aws

Guide du développeur

Amazon Timestream

Amazon Timestream: Guide du développeur

Table of Contents

À quoi sert Amazon Timestream ? LiveAnalytics

Amazon Timestream LiveAnalytics for est une base de données de séries chronologiques rapide, évolutive, entièrement gérée et spécialement conçue qui facilite le stockage et l'analyse de milliards de points de données de séries chronologiques par jour. Timestream for vous LiveAnalytics permet de gagner du temps et de réduire les coûts liés à la gestion du cycle de vie des séries chronologiques en conservant les données récentes en mémoire et en transférant les données historiques vers un niveau de stockage optimisé en termes de coûts basé sur des politiques définies par l'utilisateur. Le moteur LiveAnalytics de requête spécialement conçu par Timestream for vous permet d'accéder aux données récentes et historiques et de les analyser ensemble, sans avoir à spécifier leur emplacement. Amazon Timestream LiveAnalytics for intègre des fonctions d'analyse de séries chronologiques qui vous aident à identifier les tendances et les modèles de vos données en temps quasi réel. Timestream for LiveAnalytics fonctionne sans serveur et augmente ou diminue automatiquement pour ajuster la capacité et les performances. Comme vous n'avez pas besoin de gérer l'infrastructure sous-jacente, vous pouvez vous concentrer sur l'optimisation et le développement de vos applications.

Timestream for s'intègre LiveAnalytics également aux services couramment utilisés pour la collecte de données, la visualisation et l'apprentissage automatique. Vous pouvez envoyer des données à Amazon Timestream LiveAnalytics pour les utiliser, AWS IoT Core Amazon Kinesis, MSK Amazon et l'open source Telegraf. Vous pouvez visualiser les données à l'aide d'Amazon QuickSight, de Grafana et d'outils de business intelligence via. JDBC Vous pouvez également utiliser Amazon SageMaker avec Timestream LiveAnalytics pour l'apprentissage automatique.

Diffusion chronologique des principaux avantages LiveAnalytics

Les principaux avantages d'Amazon Timestream LiveAnalytics pour les utilisateurs sont les suivants :

- Sans serveur avec auto-scaling : avec Amazon Timestream LiveAnalytics pour, il n'y a aucun serveur à gérer et aucune capacité à provisionner. À mesure que les besoins de votre application évoluent, Timestream for évolue LiveAnalytics automatiquement pour ajuster la capacité.
- Gestion du cycle de vie des données : Amazon Timestream simplifie le processus complexe LiveAnalytics de gestion du cycle de vie des données. Il propose une hiérarchisation du stockage, avec une mémoire pour les données récentes et une mémoire magnétique pour les données historiques. Amazon Timestream automatise le transfert des données de la mémoire vers la mémoire magnétique en fonction de politiques configurables par l'utilisateur.

- Accès aux données simplifié : avec Amazon Timestream LiveAnalytics pour, vous n'avez plus besoin d'utiliser des outils disparates pour accéder aux données récentes et historiques. Le moteur de requêtes spécialement conçu par Amazon Timestream LiveAnalytics for accède aux données et les combine de manière transparente entre les niveaux de stockage sans que vous ayez à spécifier l'emplacement des données.
- Spécialement conçu pour les séries chronologiques : vous pouvez analyser rapidement les données des séries chronologiques à l'aide SQL de fonctions de séries chronologiques intégrées pour le lissage, l'approximation et l'interpolation. Timestream for prend LiveAnalytics également en charge les agrégats avancés, les fonctions de fenêtre et les types de données complexes tels que les tableaux et les lignes.
- Toujours cryptées : Amazon Timestream garantit que LiveAnalytics les données de vos séries chronologiques sont toujours cryptées, qu'elles soient au repos ou en transit. Amazon Timestream LiveAnalytics pour vous permet également de spécifier AWS KMS une clé gérée par le client CMK () pour chiffrer les données dans le magasin magnétique.
- Haute disponibilité : Amazon Timestream garantit la haute disponibilité de vos demandes d'écriture et de lecture en répliquant automatiquement les données et en allouant des ressources dans au moins 3 zones de disponibilité différentes au sein d'une même région. AWS Pour plus d'informations, consultez le contrat de niveau de service Timestream.
- Durabilité : Amazon Timestream garantit la durabilité de vos données en répliquant automatiquement les données de votre mémoire et de votre magasin magnétique dans différentes zones de disponibilité au sein d'une même région. AWS Toutes vos données sont écrites sur le disque avant de confirmer que votre demande d'écriture est terminée.

Timestream pour les cas d'utilisation LiveAnalytics

Voici quelques exemples d'une liste croissante de cas d'utilisation de Timestream pour LiveAnalytics :

- Surveillance des métriques pour améliorer les performances et la disponibilité de vos applications.
- Stockage et analyse de la télémétrie industrielle pour rationaliser la gestion et la maintenance des équipements.
- Suivi de l'interaction des utilisateurs avec une application au fil du temps.
- Stockage et analyse des données des capteurs IoT.

Commencer à utiliser Timestream pour LiveAnalytics

Nous vous recommandons de commencer par lire les sections suivantes :

- Didacticiel- Pour créer une base de données contenant des exemples d'ensembles de données et exécuter des exemples de requêtes.
- Amazon LiveAnalytics Timestream pour les concepts- Pour connaître les principes essentiels du Timestream pour les LiveAnalytics concepts.
- Accès à Timestream pour LiveAnalytics- Pour savoir comment accéder à Timestream pour LiveAnalytics utiliser la console AWS CLI, ou. API
- Quotas- Pour en savoir plus sur les quotas relatifs au nombre de Timestream pour les LiveAnalytics composants que vous pouvez provisionner.

Pour savoir comment commencer rapidement à développer des applications pour Timestream for LiveAnalytics, consultez les pages suivantes :

- À l'aide du AWS SDKs
- Référence du langage de requête

Comment ça marche

Les sections suivantes fournissent une vue d'ensemble des composants du service Amazon Timestream for Live Analytics et de la manière dont ils interagissent.

Après avoir lu cette introduction, consultez les Accès à Timestream pour LiveAnalytics sections pour savoir comment accéder à Timestream pour Live Analytics à l'aide de la console AWS CLI, ou. SDKs

Rubriques

- Amazon LiveAnalytics Timestream pour les concepts
- Architecture
- écrit
- Stockage
- Requêtes
- Requêtes planifiées
- Unité de calcul Timestream () TCU

Amazon LiveAnalytics Timestream pour les concepts

Les données de séries chronologiques sont une séquence de points de données enregistrés sur un intervalle de temps. Ce type de données est utilisé pour mesurer les événements qui changent au fil du temps. Voici quelques exemples :

- Cours des actions au fil du temps
- Mesures de température au fil du temps
- CPUutilisation d'une EC2 instance au fil du temps

Dans le cas des séries chronologiques, chaque point de données se compose d'un horodatage, d'un ou de plusieurs attributs et de l'événement qui change au fil du temps. Ces données peuvent être utilisées pour obtenir des informations sur les performances et l'état d'une application, détecter les anomalies et identifier les opportunités d'optimisation. Par exemple, les DevOps ingénieurs peuvent souhaiter consulter des données qui mesurent l'évolution des indicateurs de performance de l'infrastructure. Les fabricants voudront peut-être suivre les données des capteurs IoT qui mesurent les modifications apportées aux équipements d'une installation. Les spécialistes du marketing en ligne souhaiteront peut-être analyser les données du flux de clics qui capturent la façon dont un utilisateur navigue sur un site Web au fil du temps. Étant donné que les données de séries chronologiques sont générées à partir de sources multiples dans des volumes extrêmement élevés, elles doivent être collectées de manière rentable en temps quasi réel et nécessitent donc un stockage efficace permettant d'organiser et d'analyser les données.

Vous trouverez ci-dessous les concepts clés de Timestream pour. LiveAnalytics

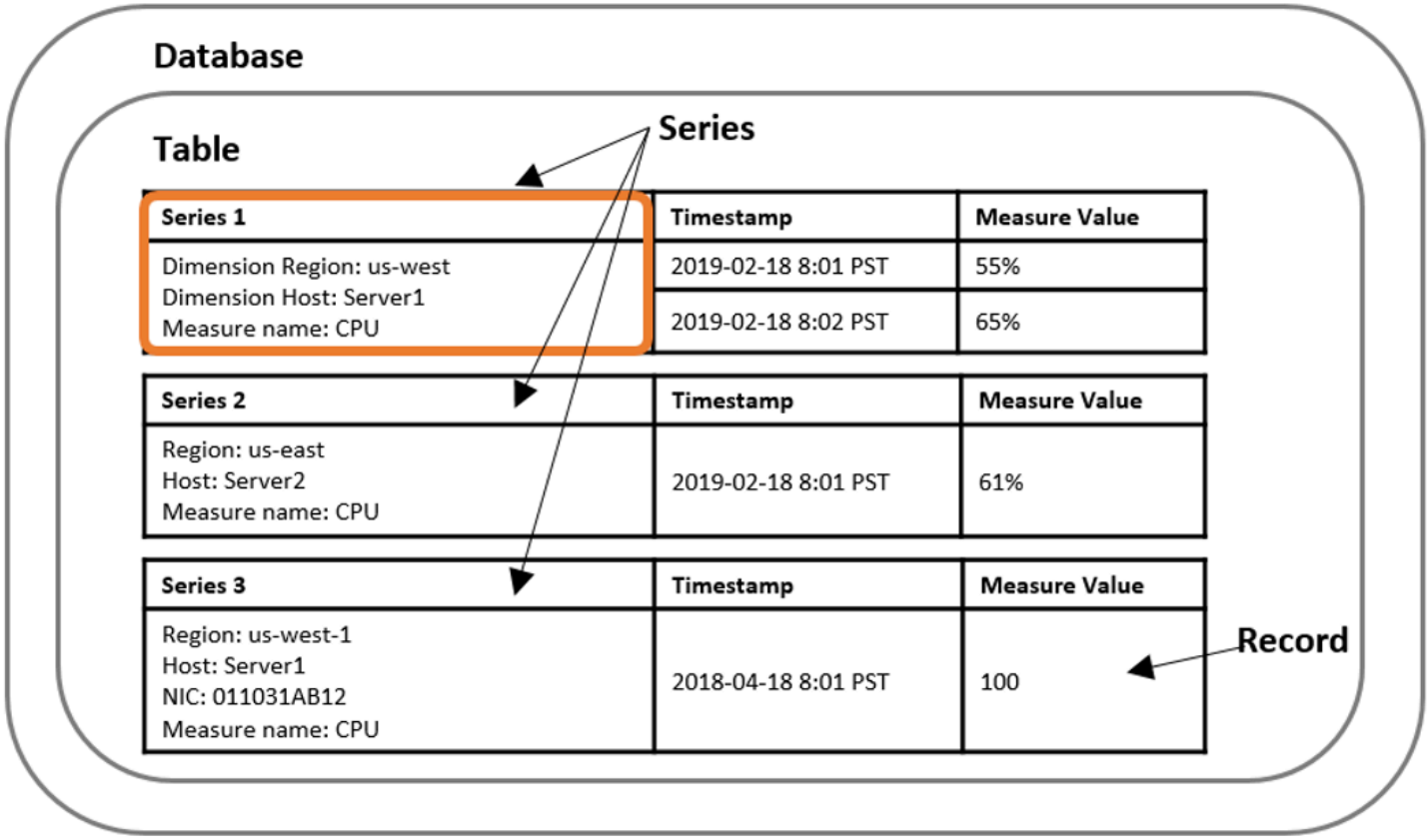
- Série chronologique : séquence d'un ou de plusieurs points de données (ou enregistrements) enregistrés sur un intervalle de temps. Les exemples incluent le cours d'une action au fil du temps, l'CPUutilisation de la mémoire d'une EC2 instance au fil du temps et la mesure de la température/pression d'un capteur IoT au fil du temps.
- Enregistrement : point de données unique dans une série chronologique.
- Dimension : attribut qui décrit les métadonnées d'une série chronologique. Une dimension se compose d’un nom de dimension et d’une valeur de dimension. Considérez les exemples suivants :
  - Lorsque vous considérez une bourse comme une dimension, le nom de la dimension est « bourse » et la valeur de la dimension est « NYSE »
  - Lorsque vous considérez une AWS région comme une dimension, le nom de la dimension est « region » et la valeur de la dimension est « us-east-1 »

  - Pour un capteur IoT, le nom de la dimension est « identifiant de l'appareil » et la valeur de dimension est « 12345 »
- Mesure : valeur réelle mesurée par l'enregistrement. Les exemples sont le cours de l'action, l'CPUutilisation de la mémoire et la lecture de la température ou de l'humidité. Les mesures se composent de noms de mesures et de valeurs de mesures. Considérez les exemples suivants :
  - Pour le cours d'une action, le nom de la mesure est « cours de l'action » et la valeur de la mesure est le cours réel de l'action à un moment donné.
  - Pour CPU l'utilisation, le nom de la mesure est « CPU utilisation » et la valeur de la mesure est l'CPUutilisation réelle.

  Les mesures peuvent être modélisées dans Timestream sous forme d'enregistrements à mesures multiples ou à LiveAnalytics mesure unique. Pour de plus amples informations, veuillez consulter Enregistrements à mesures multiples contre enregistrements à mesure unique.
- Horodatage : indique quand une mesure a été collectée pour un enregistrement donné. Timestream for LiveAnalytics prend en charge les horodatages avec une granularité en nanosecondes.
- Tableau : conteneur pour un ensemble de séries chronologiques connexes.
- Base de données : conteneur de premier niveau pour les tables.

Résumé de Timestream pour les concepts LiveAnalytics

Une base de données contient au moins 0 tables. Chaque table contient au moins 0 séries chronologiques. Chaque série chronologique consiste en une séquence d'enregistrements sur un intervalle de temps donné avec une granularité spécifiée. Chaque série chronologique peut être décrite à l'aide de ses métadonnées ou dimensions, de ses données ou mesures et de ses horodatages.

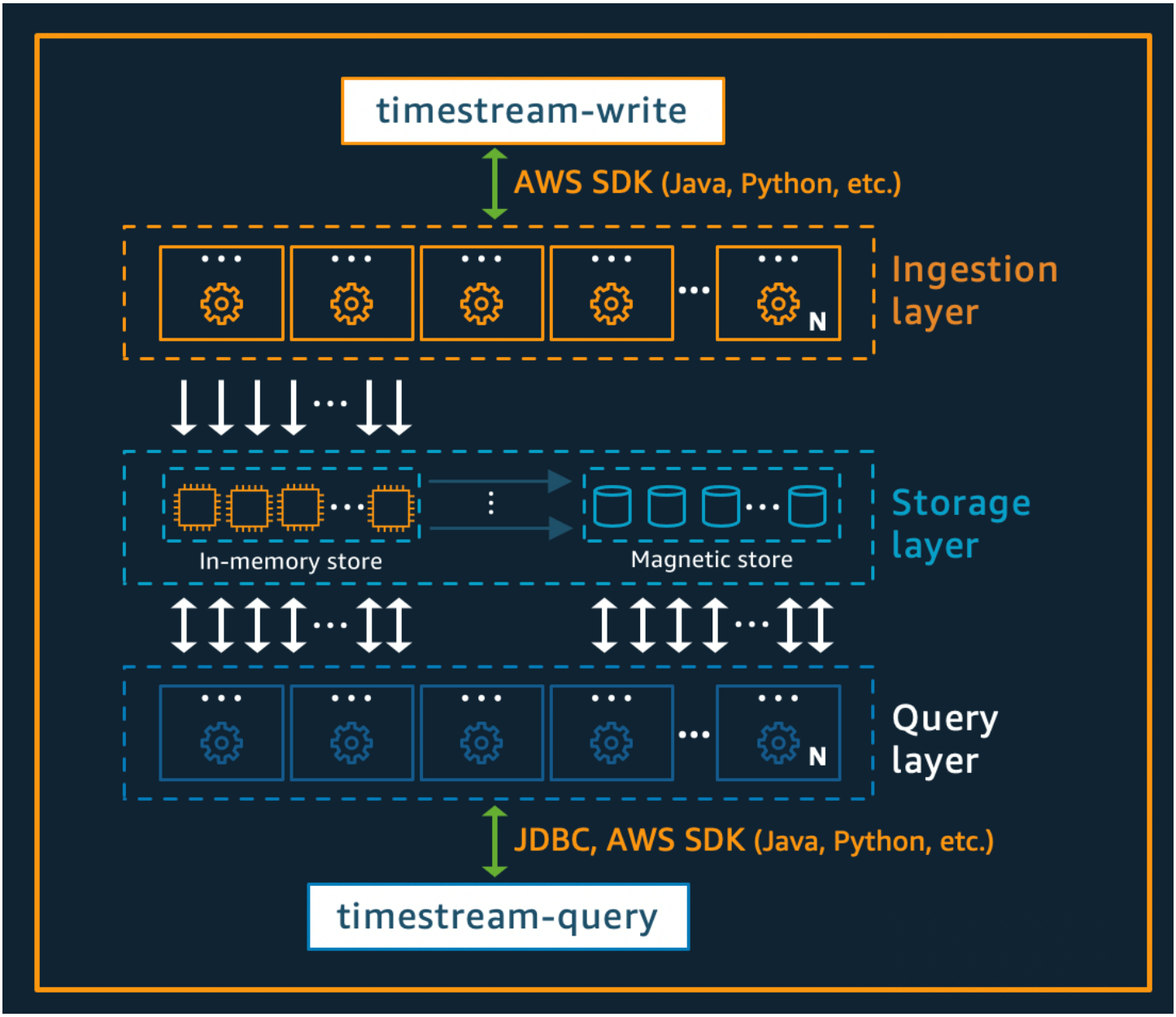


Architecture

Amazon Timestream for Live Analytics a été entièrement conçu pour collecter, stocker et traiter des séries chronologiques à grande échelle. Son architecture sans serveur prend en charge des systèmes d'ingestion de données, de stockage et de traitement des requêtes totalement découplés qui peuvent évoluer indépendamment. Cette conception simplifie chaque sous-système, ce qui permet d'atteindre une fiabilité sans faille, d'éliminer les goulots d'étranglement liés à la mise à l'échelle et de réduire les risques de défaillances du système corrélées. Chacun de ces facteurs prend de l'importance à mesure que le système évolue.

Rubriques

- Écrivez de l'architecture
- Architecture de stockage
- Architecture des requêtes
- Architecture cellulaire



Écrivez de l'architecture

Lors de l'écriture de données chronologiques, Amazon Timestream for Live Analytics achemine les données d'une table ou d'une partition vers une instance de mémoire tolérante aux pannes qui traite les écritures de données à haut débit. La mémoire assure à son tour la durabilité dans un système de stockage distinct qui réplique les données sur trois zones de disponibilité ()AZs. La réplication est basée sur le quorum, de sorte que la perte de nœuds, ou d'une zone de disponibilité complète, ne perturbe pas la disponibilité en écriture. En temps quasi réel, d'autres nœuds de stockage en mémoire se synchronisent avec les données afin de répondre aux requêtes. Les nœuds de réplication du lecteur s'étendent AZs également, afin de garantir une haute disponibilité en lecture.

Timestream for Live Analytics prend en charge l'écriture de données directement dans le magasin magnétique, pour les applications générant des données tardives à faible débit. Les données arrivées tardivement sont des données dont l'horodatage est antérieur à l'heure actuelle. À l'instar des écritures à haut débit dans la mémoire, les données écrites dans la mémoire magnétique sont répliquées sur trois unités AZs et la réplication est basée sur le quorum.

Que les données soient écrites dans la mémoire ou dans le stockage magnétique, Timestream for Live Analytics indexe et partitionne automatiquement les données avant de les enregistrer dans le stockage. Une seule table Timestream for Live Analytics peut comporter des centaines, des milliers, voire des millions de partitions. Les partitions individuelles ne communiquent pas directement entre elles et ne partagent aucune donnée (architecture sans partage). Au lieu de cela, le partitionnement d'une table est suivi via un service de suivi et d'indexation des partitions à haute disponibilité. Cela permet une autre séparation des préoccupations, conçue spécifiquement pour minimiser l'effet des défaillances dans le système et réduire considérablement la probabilité de défaillances corrélées.

Architecture de stockage

Lorsque les données sont stockées dans Timestream for Live Analytics, elles sont organisées par ordre chronologique et dans le temps en fonction des attributs contextuels écrits avec les données. Il est important de disposer d'un schéma de partitionnement qui divise « l'espace » en plus du temps pour dimensionner massivement un système de séries chronologiques. Cela est dû au fait que la plupart des données de séries chronologiques sont écrites à l'heure actuelle ou aux alentours de cette date. Par conséquent, le partitionnement basé uniquement sur le temps ne permet pas de répartir correctement le trafic d'écriture ou de permettre un élagage efficace des données au moment de la requête. C'est important pour le traitement de séries chronologiques à grande échelle, et cela a permis à Timestream for Live Analytics d'augmenter de plusieurs ordres de grandeur par rapport aux autres principaux systèmes actuels en mode sans serveur. Les partitions qui en résultent sont appelées « tuiles » car elles représentent les divisions d'un espace bidimensionnel (conçues pour être de taille similaire). Pour les tables Live Analytics, Timestream commence par une partition unique (tuile), puis se divise dans la dimension spatiale en fonction du débit. Lorsque les tuiles atteignent une certaine taille, elles se divisent dans la dimension temporelle afin d'obtenir un meilleur parallélisme de lecture à mesure que la taille des données augmente.

Timestream for Live Analytics est conçu pour gérer automatiquement le cycle de vie des données de séries chronologiques. Timestream for Live Analytics propose deux magasins de données : un stockage en mémoire et un magasin magnétique rentable. Il prend également en charge la configuration de politiques au niveau des tables pour transférer automatiquement les données entre les boutiques. Les écritures de données à haut débit entrantes arrivent dans la mémoire,

où les données sont optimisées pour les écritures, ainsi que pour les lectures effectuées à l'heure actuelle pour alimenter les requêtes de type tableau de bord et alertes. Lorsque le délai principal pour les besoins d'écriture, d'alerte et de tableau de bord est dépassé, les données peuvent circuler automatiquement de la mémoire vers la mémoire magnétique afin d'optimiser les coûts. Timestream for Live Analytics permet de définir une politique de conservation des données sur la mémoire à cette fin. Les écritures de données pour les données arrivées en retard sont directement enregistrées dans le magasin magnétique.

Une fois que les données sont disponibles dans la mémoire magnétique (en raison de l'expiration de la période de conservation de la mémoire ou en raison d'écritures directes dans la mémoire magnétique), elles sont réorganisées dans un format hautement optimisé pour les lectures de gros volumes de données. La mémoire magnétique dispose également d'une politique de conservation des données qui peut être configurée s'il existe un seuil temporel pendant lequel les données ne sont plus utiles. Lorsque les données dépassent la plage de temps définie pour la politique de conservation des mémoires magnétiques, elles sont automatiquement supprimées. Ainsi, avec Timestream for Live Analytics, outre certaines configurations, la gestion du cycle de vie des données s'effectue de manière fluide en arrière-plan.

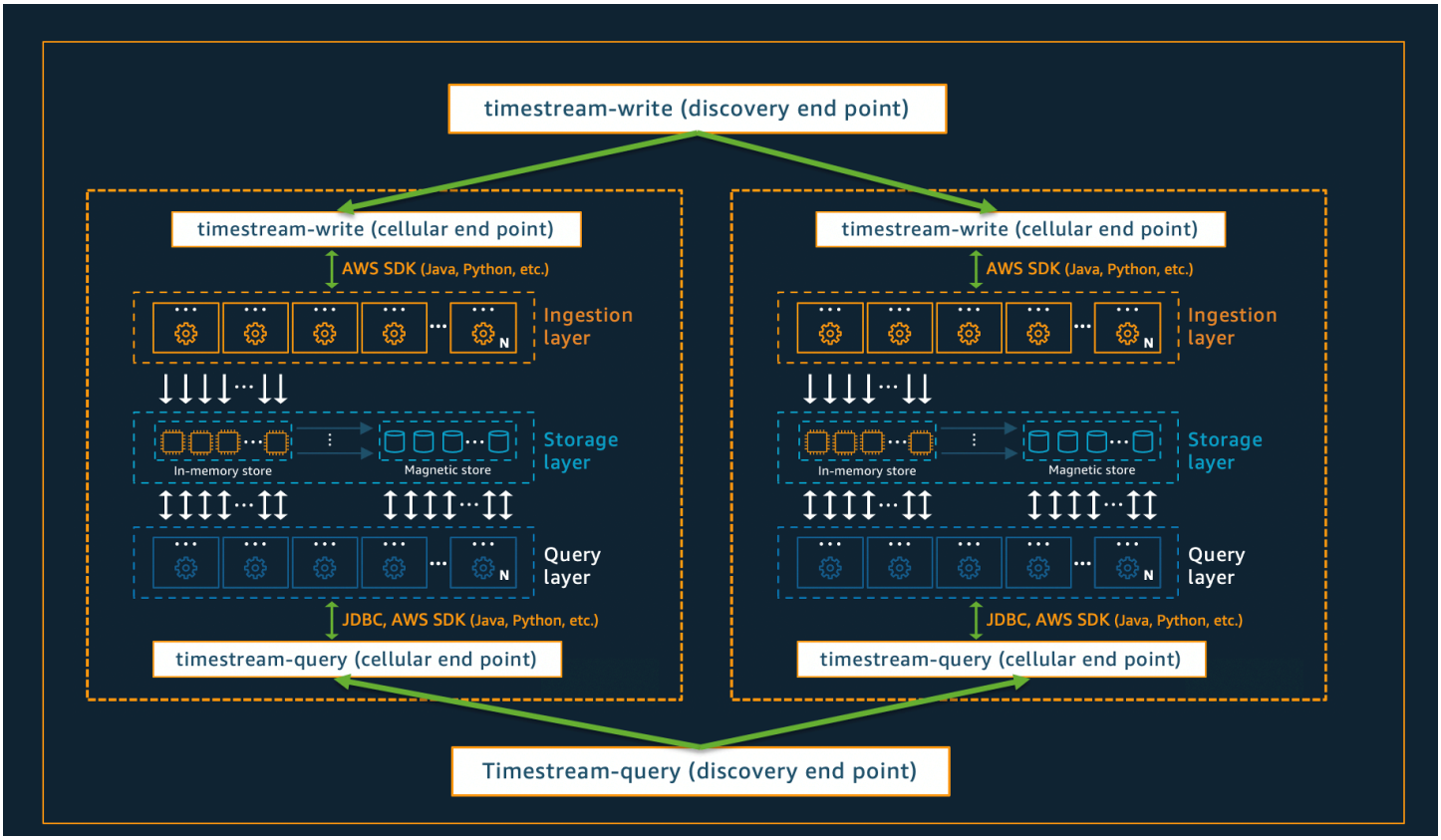
Architecture des requêtes

Les requêtes Timestream for Live Analytics sont exprimées dans une SQL grammaire qui comporte des extensions pour la prise en charge spécifique aux séries chronologiques (types de données et fonctions spécifiques aux séries chronologiques), ce qui facilite l'apprentissage pour les développeurs déjà habitués. SQL Les requêtes sont ensuite traitées par un moteur de requêtes adaptatif et distribué qui utilise les métadonnées du service de suivi et d'indexation des tuiles pour accéder et combiner facilement les données entre les magasins de données au moment où la requête est émise. Cela en fait une expérience qui trouve un écho auprès des clients, car elle réduit la plupart des complexités de Rube Goldberg en une abstraction de base de données simple et familière.

Les requêtes sont exécutées par un parc de travailleurs dédiés, le nombre de travailleurs recrutés pour exécuter une requête donnée étant déterminé par la complexité de la requête et la taille des données. Les performances des requêtes complexes sur de grands ensembles de données sont obtenues grâce à un parallélisme massif, à la fois sur le parc d'exécution des requêtes et sur les flottes de stockage du système. La capacité d'analyser rapidement et efficacement d'énormes quantités de données est l'un des principaux atouts de Timestream for Live Analytics. Une seule requête portant sur des téraoctets, voire des pétaoctets de données, peut nécessiter l'exécution simultanée de milliers de machines.

Architecture cellulaire

Pour garantir que Timestream for Live Analytics puisse offrir une évolutivité pratiquement infinie à vos applications, tout en garantissant une disponibilité de 99,99 %, le système est également conçu selon une architecture cellulaire. Plutôt que de dimensionner le système dans son ensemble, Timestream for Live Analytics se segmente en plusieurs petites copies de lui-même, appelées cellules. Cela permet de tester les cellules à grande échelle et d'éviter qu'un problème systémique dans une cellule n'affecte l'activité des autres cellules d'une région donnée. Bien que Timestream for Live Analytics soit conçu pour prendre en charge plusieurs cellules par région, considérez le scénario fictif suivant, dans lequel il y a deux cellules dans une région.



Dans le scénario décrit ci-dessus, les demandes d'ingestion de données et de requête sont d'abord traitées par le point de terminaison de découverte pour l'ingestion de données et les requêtes, respectivement. Le point de terminaison de découverte identifie ensuite la cellule contenant les données du client et dirige la demande vers le point de terminaison d'ingestion ou de requête approprié pour cette cellule. Lorsque vous utilisez leSDKs, ces tâches de gestion des terminaux sont gérées de manière transparente pour vous.

> **Note**
>
> Lorsque vous utilisez des VPC terminaux avec Timestream for Live Analytics ou que vous accédez directement aux REST API opérations de Timestream for Live Analytics, vous devez interagir directement avec les terminaux cellulaires. Pour obtenir des instructions sur la manière de procéder, consultez VPCEndpoints pour obtenir des instructions sur la façon de configurer les VPC points de terminaison, et Endpoint Discovery Pattern pour des instructions sur l'invocation directe des opérations. REST API

écrit

Vous pouvez collecter des séries chronologiques à partir d'appareils connectés, de systèmes informatiques et d'équipements industriels, et les écrire dans Timestream pour Live Analytics. Timestream for Live Analytics vous permet d'écrire des points de données d'une seule série chronologique et/ou des points de données de plusieurs séries dans une seule demande d'écriture lorsque les séries temporelles appartiennent à la même table. Pour vous faciliter la tâche, Timestream for Live Analytics propose un schéma flexible qui détecte automatiquement les noms de colonnes et les types de données de vos tables Timestream for Live Analytics en fonction des noms des dimensions et des types de données des valeurs de mesure que vous spécifiez lorsque vous appelez des écritures dans la base de données. Vous pouvez également écrire des lots de données dans Timestream pour Live Analytics.

> **Note**
>
> Timestream for Live Analytics prend en charge la sémantique de cohérence éventuelle pour les lectures. Cela signifie que lorsque vous interrogez des données immédiatement après avoir écrit un lot de données dans Timestream for Live Analytics, les résultats de la requête peuvent ne pas refléter les résultats d'une opération d'écriture récemment terminée. Les résultats peuvent également inclure des données périmées. De même, lors de l'écriture de données de séries chronologiques avec une ou plusieurs nouvelles dimensions, une requête peut renvoyer un sous-ensemble partiel de colonnes pendant une courte période. Si vous répétez ces requêtes après un court laps de temps, les résultats devraient renvoyer les données les plus récentes.

Vous pouvez écrire des données en utilisant le AWS SDKsAWS CLI, ou via AWS LambdaAWS IoT Core,Service géré Amazon pour Apache Flink,Amazon Kinesis,Amazon MSK, etTelegraf open source.

Rubriques

Types de données

Timestream for Live Analytics prend en charge les types de données suivants pour les écritures.

Type de données	Description
BIGINT	Représente un entier signé de 64 bits.
BOOLEAN	Représente les deux valeurs de vérité de la logique, à savoir vrai et faux.
DOUBLE	64 bits à précision variable implémentant la IEEE norme 754 pour l'arithmétique binaire à virgule flottante. **Note** Il existe des fonctions de langage de requête `Infinity` et des valeurs `NaN` doubles qui peuvent être utilisées dans les requêtes. Mais vous ne pouvez pas écrire ces valeurs dans Timestream.
VARCHAR	Données de caractères de longueur variable avec une longueur maximale facultative. La limite maximale est de 2 Ko.

Type de données	Description
MULTI	Type de données pour les enregistrements à mesures multiples . Ce type de données inclut une ou plusieurs mesures de type `BIGINT` `BOOLEAN`, `DOUBLE`, `VARCHAR`, et `TIMESTAMP` .
TIMESTAMP	Représente une instance dans le temps en utilisant une précision de nanosecondeUTC, en suivant le temps écoulé depuis l'époque d'Unix. Ce type de données n'est actuellement pris en charge que pour les enregistrements multi-mesures (c'est-à-dire dans les limites des valeurs de mesure du type`MULTI`). *`YYYY`*`-`*`MM`*`-`*`DD`* *`hh`*`:`*`mm`*`:`*`ss`*`.`*`sssssssss`* Écrit les horodatages de support compris entre`1970-01-01 00:00:00.000000000` . `2262-04-11 23:47:16.854775807`

Aucune définition initiale du schéma

Avant d'envoyer des données vers Amazon Timestream pour Live Analytics, vous devez créer une base de données et une table à AWS Management Console l'aide des opérations Timestream pour Live SDKs Analytics ou Timestream pour Live Analytics. API Pour plus d'informations, consultez Créer une base de données et Créer une table . Lors de la création de la table, il n'est pas nécessaire de définir le schéma dès le départ. Amazon Timestream for Live Analytics détecte automatiquement le schéma en fonction des mesures et des dimensions des points de données envoyés. Vous n'avez donc plus besoin de modifier votre schéma hors ligne pour l'adapter à l'évolution rapide de vos données de séries chronologiques.

Écrire des données (insertions et insertions)

L'opération d'écriture dans Amazon Timestream for Live Analytics vous permet d'insérer et de modifier des données. Par défaut, les écritures dans Amazon Timestream pour Live Analytics suivent la sémantique du premier auteur, selon laquelle les données sont stockées sous forme d'ajout uniquement et les enregistrements dupliqués sont rejetés. Alors que la sémantique du premier rédacteur gagne répond aux exigences de nombreuses applications de séries chronologiques, il existe des scénarios dans lesquels les applications doivent mettre à jour les enregistrements existants de manière idempotente et/ou écrire des données selon la sémantique du dernier rédacteur

gagnant, dans lesquels l'enregistrement contenant la version la plus récente est stocké dans le service. Pour répondre à ces scénarios, Amazon Timestream for Live Analytics permet de modifier les données. Upsert est une opération qui insère un enregistrement dans le système lorsqu'il n'existe pas ou met à jour l'enregistrement lorsqu'il en existe un. Lorsque l'enregistrement est mis à jour, il est mis à jour de manière idempotente.

Il n'existe aucune opération de suppression au niveau de l'enregistrement. Mais les tables et les bases de données peuvent être supprimées.

Écrire des données dans la mémoire et dans la mémoire magnétique

Amazon Timestream for Live Analytics permet d'écrire des données directement dans la mémoire et dans la mémoire magnétique. La mémoire est optimisée pour les écritures de données à haut débit et la mémoire magnétique est optimisée pour les écritures à faible débit de données d'arrivée tardive.

Les données arrivées tardivement sont des données dont l'horodatage est antérieur à l'heure actuelle et en dehors de la période de conservation de la mémoire. Vous devez explicitement activer la possibilité d'écrire des données arrivées tardivement dans la mémoire magnétique en activant les écritures dans la mémoire magnétique pour la table. Est également défini lors de la création d'une table. `MagneticStoreRejectedDataLocation` Pour écrire dans le magasin magnétique, les appelants `WriteRecords` doivent disposer des `S3:PutObject` autorisations d'accès au compartiment S3 spécifié `MagneticStoreRejectedDataLocation` lors de la création de la table. Pour plus d'informations, reportez-vous aux WriteRecordssections CreateTable, et PutObject.

Écrire des données avec des enregistrements à mesure unique et des enregistrements à mesures multiples

Amazon Timestream for Live Analytics permet d'écrire des données à l'aide de deux types d'enregistrements, à savoir les enregistrements à mesure unique et les enregistrements à mesures multiples.

Enregistrements à mesure unique

Les enregistrements à mesure unique vous permettent d'envoyer une seule mesure par enregistrement. Lorsque des données sont envoyées à Timestream pour Live Analytics à l'aide de ce format, Timestream for Live Analytics crée une ligne de tableau par enregistrement. Cela signifie que si un appareil émet 4 mesures et que chaque métrique est envoyée sous la forme d'un enregistrement de mesure unique, Timestream for Live Analytics créera 4 lignes dans le tableau pour stocker ces données, et les attributs de l'appareil seront répétés pour chaque ligne. Ce format

est recommandé lorsque vous souhaitez surveiller une seule métrique à partir d'une application ou lorsque votre application n'émet pas plusieurs métriques en même temps.

Enregistrements à mesures multiples

Avec les enregistrements à mesures multiples, vous pouvez stocker plusieurs mesures dans une seule ligne de tableau, au lieu de stocker une mesure par ligne de tableau. Les enregistrements multi-mesures vous permettent donc de migrer vos données existantes depuis des bases de données relationnelles vers Amazon Timestream for Live Analytics avec un minimum de modifications.

Vous pouvez également regrouper plus de données dans une seule demande d'écriture que dans des enregistrements à mesure unique. Cela augmente le débit et les performances d'écriture des données, tout en réduisant le coût des écritures de données. En effet, le fait de regrouper davantage de données dans une demande d'écriture permet à Amazon Timestream for Live Analytics d'identifier davantage de données reproductibles dans une seule demande d'écriture (le cas échéant) et de ne facturer qu'une seule fois pour les données répétées.

Rubriques

Enregistrements à mesures multiples

Avec les enregistrements à mesures multiples, vous pouvez stocker vos données de séries chronologiques dans un format plus compact dans la mémoire et le stockage magnétique, ce qui permet de réduire les coûts de stockage des données. En outre, le stockage de données compact permet d'écrire des requêtes plus simples pour la récupération de données, d'améliorer les performances des requêtes et de réduire le coût des requêtes.

En outre, les enregistrements à mesures multiples prennent également en charge le type de TIMESTAMP données permettant de stocker plusieurs horodatages dans un enregistrement de série chronologique. TIMESTAMPles attributs d'un enregistrement multi-mesures prennent en charge les horodatages futurs ou passés. Les enregistrements à mesures multiples contribuent donc à améliorer les performances, les coûts et la simplicité des requêtes, tout en offrant une plus grande flexibilité pour le stockage de différents types de mesures corrélées.

Avantages

Les avantages de l'utilisation d'enregistrements à mesures multiples sont les suivants.

- Performances et coûts — Les enregistrements à mesures multiples vous permettent d'écrire plusieurs mesures de séries chronologiques en une seule demande d'écriture. Cela augmente le débit d'écriture et réduit également le coût des écritures. Avec les enregistrements à mesures multiples, vous pouvez stocker les données de manière plus compacte, ce qui permet de réduire les coûts de stockage des données. Le stockage compact des données des enregistrements à mesures multiples permet de réduire le nombre de données traitées par les requêtes. Ceci est conçu pour améliorer les performances globales des requêtes et contribuer à réduire le coût des requêtes.
- Simplicité des requêtes : avec les enregistrements à mesures multiples, il n'est pas nécessaire d'écrire des expressions de table communes complexes (CTEs) dans une requête pour lire plusieurs mesures avec le même horodatage. Cela est dû au fait que les mesures sont stockées sous forme de colonnes dans une seule ligne de tableau. Les enregistrements multi-mesures permettent donc d'écrire des requêtes plus simples.
- Flexibilité de modélisation des données — Vous pouvez écrire les futurs horodatages dans Timestream pour Live Analytics en utilisant le type de TIMESTAMP données et des enregistrements à mesures multiples. Un enregistrement à mesures multiples peut avoir plusieurs attributs de type de TIMESTAMP données, en plus du champ horaire d'un enregistrement. TIMESTAMPdans un enregistrement multi-mesures, les attributs peuvent être horodatés dans le futur ou dans le passé et se comporter comme le champ horaire, sauf que Timestream for Live Analytics n'indexe pas les valeurs de type TIMESTAMP dans un enregistrement multi-mesures.

Cas d’utilisation

Vous pouvez utiliser des enregistrements de mesures multiples pour toute application de séries chronologiques qui génère plusieurs mesures à partir du même appareil à un moment donné. Voici quelques exemples d'applications.

- Une plateforme de streaming vidéo qui génère des centaines de métriques à un moment donné.
- Dispositifs médicaux qui génèrent des mesures telles que le taux d'oxygène dans le sang, la fréquence cardiaque et le pouls.
- Des équipements industriels tels que les plates-formes pétrolières qui génèrent des métriques, des capteurs de température et des capteurs météorologiques.
- Autres applications conçues avec un ou plusieurs microservices.

Exemple : surveillance des performances et de l'état d'une application de streaming vidéo

Prenons l'exemple d'une application de streaming vidéo exécutée sur 200 EC2 instances. Vous souhaitez utiliser Amazon Timestream for Live Analytics pour stocker et analyser les métriques émises par l'application, afin de comprendre les performances et l'état de votre application, d'identifier rapidement les anomalies, de résoudre les problèmes et de découvrir des opportunités d'optimisation.

Nous modéliserons ce scénario avec des enregistrements à mesure unique et des enregistrements à mesures multiples, puis comparer/contraster les deux approches. Pour chaque approche, nous formulons les hypothèses suivantes.

- Chaque EC2 instance émet quatre mesures (video_start-up_time, rebuffering_ratio, video_playback_failures et average_frame_rate) et quatre dimensions (device_id, device_type, os_version et region) par seconde.
- Vous souhaitez stocker 6 heures de données dans la mémoire et 6 mois de données dans la mémoire magnétique.
- Pour identifier les anomalies, vous avez configuré 10 requêtes exécutées toutes les minutes afin d'identifier toute activité inhabituelle au cours des dernières minutes. Vous avez également créé un tableau de bord avec huit widgets qui affichent les 6 dernières heures de données, afin que vous puissiez surveiller efficacement votre application. Ce tableau de bord est accessible à cinq utilisateurs à tout moment et est actualisé automatiquement toutes les heures.

Utilisation d'enregistrements de mesures uniques

Modélisation des données : avec des enregistrements à mesure unique, nous allons créer un enregistrement pour chacune des quatre mesures (temps de démarrage de la vidéo, taux de rebufférisation, échecs de lecture vidéo et fréquence d'images moyenne). Chaque enregistrement comportera les quatre dimensions (device_id, device_type, os_version et region) et un horodatage.

Écritures : lorsque vous écrivez des données dans Amazon Timestream pour Live Analytics, les enregistrements sont construits comme suit.

```
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);
    }
  }
```

Lorsque vous stockez des enregistrements à mesure unique, les données sont représentées logiquement comme suit.

Heure	device_id	type_d'appareil	os_version	region	nom_mesure	value_measure : :bigint	valeur_measure : double
2021-09-07 21:48:44.000000000	12345678	iPhone 11	14,8	us-east-1	video_start_time	200	
2021-09-07 21:48:44.000000000	12345678	iPhone 11	14,8	us-east-1	ratio de rebuffering_		0.5
2021-09-07	12345678	iPhone 11	14,8	us-east-1	échecs de	0	

Heure	device_id	type_d'appareil	os_version	region	nom_mesure	value_measure : :bigint	valeur_measure : double
21:48:44.000000000					lecture vidéo		
2021-09-07 21:48:44.000000000	12345678	iPhone 11	14,8	us-east-1	cadre_image moyen		0,85
2021-09-07 21:53:44.000000000	12345678	iPhone 11	14,8	us-east-1	video_start_time	500	
2021-09-07 21:53:44.000000000	12345678	iPhone 11	14,8	us-east-1	ratio de rebuffering_		1.5
2021-09-07 21:53:44.000000000	12345678	iPhone 11	14,8	us-east-1	échecs de lecture vidéo	10	
2021-09-07 21:53:44.000000000	12345678	iPhone 11	14,8	us-east-1	cadre_image moyen		0.2

Requêtes : vous pouvez écrire une requête qui récupère tous les points de données portant le même horodatage reçus au cours des 15 dernières minutes comme suit.

```
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
```

Coût de la charge de travail : Le coût de cette charge de travail est estimé à 373,23$ par mois avec des enregistrements à mesure unique

Utilisation d'enregistrements à mesures multiples

Modélisation des données : avec les enregistrements à mesures multiples, nous allons créer un enregistrement contenant les quatre mesures (heure de démarrage de la vidéo, taux de rebufférisation, échecs de lecture vidéo et fréquence d'images moyenne), les quatre dimensions (device_id, device_type, os_version et region) et un horodatage.

Écritures : lorsque vous écrivez des données dans Amazon Timestream pour Live Analytics, les enregistrements sont construits comme suit.

```
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);
    }
  }
```

Lorsque vous stockez des enregistrements à mesures multiples, les données sont représentées logiquement comme suit.

Heure	device_id	type_d'appareil	os_version	region	nom_mesure	video_start_time	ratio de rebuffering_	video_playback_failures	cadre_image moyen
2021-09-7 21:48:44 0000000	1234567	iPhone 11	14,8	us-east-1	video_metrics	200	0.5	0	0,85
2021-09-7 21:53:44 0000000	1234567	iPhone 11	14,8	us-east-1	video_metrics	500	1.5	10	0.2

Requêtes : vous pouvez écrire une requête qui récupère tous les points de données portant le même horodatage reçus au cours des 15 dernières minutes comme suit.

```
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)
```

Coût de la charge de travail : Le coût de la charge de travail est estimé à 127,43$ avec des enregistrements à mesures multiples.

> **Note**
>
> Dans ce cas, l'utilisation d'enregistrements à mesures multiples réduit les dépenses mensuelles totales estimées de 2,5 fois, le coût d'écriture des données étant réduit de 3,3 fois, le coût de stockage réduit de 3,3 fois et le coût des requêtes réduit de 1,2 fois.

Écrire des données avec un horodatage existant dans le passé ou dans le futur

Timestream for Live Analytics permet d'écrire des données avec un horodatage situé en dehors de la fenêtre de rétention de la mémoire par le biais de différents mécanismes.

- Écritures sur mémoire magnétique : vous pouvez écrire les données arrivées en retard directement dans la mémoire magnétique par le biais d'écritures sur mémoire magnétique. Pour utiliser les écritures en mémoire magnétique, vous devez d'abord activer les écritures en mémoire magnétique pour une table. Vous pouvez ensuite ingérer des données dans la table en utilisant le même mécanisme que celui utilisé pour écrire des données dans la mémoire. Amazon Timestream for Live Analytics inscrira automatiquement les données dans le magasin magnétique en fonction de son horodatage.

  > **Note**
  >
  > La write-to-read latence de la mémoire magnétique peut atteindre 6 heures, contrairement à l'écriture de données dans la mémoire, où la write-to-read latence est inférieure à la seconde.

- TIMESTAMPtype de données pour les mesures : vous pouvez utiliser le type de TIMESTAMP données pour stocker des données du passé, du présent ou du futur. Un enregistrement à mesures multiples peut avoir plusieurs attributs de type de TIMESTAMP données, en plus du champ horaire d'un enregistrement. TIMESTAMPdans un enregistrement multi-mesures, les attributs peuvent être horodatés dans le futur ou dans le passé et se comporter comme le champ horaire, sauf que Timestream for Live Analytics n'indexe pas les valeurs de type TIMESTAMP dans un enregistrement multi-mesures.

> **Note**
>
> Le type de TIMESTAMP données n'est pris en charge que pour les enregistrements à mesures multiples.

Cohérence éventuelle pour les lectures

Timestream for Live Analytics prend en charge la sémantique de cohérence éventuelle pour les lectures. Cela signifie que lorsque vous interrogez des données immédiatement après avoir écrit un lot de données dans Timestream for Live Analytics, les résultats de la requête peuvent ne pas refléter les résultats d'une opération d'écriture récemment terminée. Si vous répétez ces requêtes après un court laps de temps, les résultats devraient renvoyer les données les plus récentes.

Rédiger des écritures par lots avec WriteRecords API

Amazon Timestream for Live Analytics vous permet d'écrire des points de données issus d'une seule série chronologique et/ou des points de données issus de plusieurs séries dans une seule demande d'écriture. Le regroupement de plusieurs points de données en une seule opération d'écriture est avantageux du point de vue des performances et des coûts. Consultez écrit la section Mesurage et tarification pour plus de détails.

> **Note**
>
> Vos demandes d'écriture adressées à Timestream pour Live Analytics peuvent être limitées à mesure que Timestream for Live Analytics évolue pour s'adapter aux besoins d'ingestion de données de votre application. Si vos applications rencontrent des exceptions de limitation, vous devez continuer à envoyer des données au même débit (ou à un débit supérieur) pour permettre à Timestream for Live Analytics de s'adapter automatiquement aux besoins de votre application.

Chargement par lots

Grâce au chargement par lots pour Amazon Timestream LiveAnalytics for, vous pouvez CSV ingérer des fichiers stockés dans Amazon S3 dans Timestream par lots. Grâce à cette nouvelle fonctionnalité, vous pouvez accéder à vos données dans Timestream LiveAnalytics sans avoir à

utiliser d'autres outils ou à écrire du code personnalisé. Vous pouvez utiliser le chargement par lots pour compléter les données avec des temps d'attente flexibles, telles que les données qui ne sont pas immédiatement requises pour les requêtes ou les analyses.

Vous pouvez créer des tâches de chargement par lots à AWS CLI l'aide AWS SDKs des AWS Management Console Pour plus d'informations, consultez Utilisation du chargement par lots avec la console, En utilisant le chargement par lots avec AWS CLI et En utilisant le chargement par lots avec AWS SDKs.

Pour plus d'informations sur le chargement par lots, consultezUtilisation du chargement par lots dans Timestream pour LiveAnalytics.

Choix entre l' WriteRecords APIopération et le chargement par lots

Grâce à cette WriteRecords API opération, vous pouvez écrire les données de vos séries chronologiques de streaming dans Timestream LiveAnalytics telles qu'elles sont générées par votre système. En utilisant WriteRecords, vous pouvez ingérer en continu un seul point de données ou de petits lots de données en temps réel. Timestream for vous LiveAnalytics propose un schéma flexible qui détecte automatiquement les noms de colonnes et les types de données de votre Timestream pour les LiveAnalytics tables, en fonction des noms des dimensions et des types de données des points de données que vous spécifiez lorsque vous appelez des écritures dans la base de données.

En revanche, le chargement par lots permet l'ingestion robuste de séries chronologiques par lots à partir de fichiers sources (CSVfichiers) dans Timestream for LiveAnalytics, à l'aide d'un modèle de données que vous définissez. Voici quelques exemples d'utilisation du chargement par lots avec un fichier source : l'importation de données de séries chronologiques en masse pour l'évaluation de Timestream par le LiveAnalytics biais d'une preuve de concept, l'importation de données de séries chronologiques en masse depuis un appareil IoT resté hors ligne pendant un certain temps et la migration de données de séries chronologiques historiques d'Amazon S3 vers Timestream pour. LiveAnalytics Pour plus d'informations sur le chargement par lots, consultezUtilisation du chargement par lots dans Timestream pour LiveAnalytics.

Les deux solutions sont sécurisées, fiables et performantes.

À utiliser WriteRecords lorsque :

- Diffusion en continu de plus petites quantités (moins de 10 Mo) de données par demande.
- Remplissage de tables existantes.
- Ingestion de données à partir d'un flux de log.

- Réalisation d'analyses en temps réel.
- Nécessitant une latence plus faible.

Utilisez le chargement par lots lorsque :

- Ingestion de grandes quantités de données provenant d'Amazon S3 dans CSV des fichiers. Pour en savoir plus sur les limites, consultez Quotas.
- Remplissage de nouvelles tables, par exemple dans le cas d'une migration de données.
- Enrichissement des bases de données avec des données historiques (ingestion dans de nouvelles tables).
- Vous avez des données sources qui changent lentement ou pas du tout.
- Les temps d'attente sont flexibles car une tâche de chargement par lots peut être en attente jusqu'à ce que les ressources soient disponibles, en particulier si vous chargez une très grande quantité de données. Le chargement par lots convient aux données qui n'ont pas besoin d'être facilement accessibles pour les requêtes ou les analyses afin d'apporter plus de clarté.

Stockage

Timestream for Live Analytics stocke et organise les données de vos séries chronologiques afin d'optimiser le temps de traitement des requêtes et de réduire les coûts de stockage. Il propose une hiérarchisation du stockage des données et prend en charge deux niveaux de stockage : un stockage de mémoire et un stockage magnétique. La mémoire est optimisée pour les écritures de données à haut débit et les point-in-time requêtes rapides. Le stockage magnétique est optimisé pour les écritures de données arrivées en retard à faible débit, le stockage de données à long terme et les requêtes analytiques rapides.

Timestream for Live Analytics garantit la durabilité de vos données en répliquant automatiquement les données de votre mémoire et de votre stockage magnétique dans différentes zones de disponibilité au sein d'une même zone. Région AWS Toutes vos données sont écrites sur le disque avant de confirmer que votre demande d'écriture est terminée.

Timestream for Live Analytics vous permet de configurer des politiques de rétention afin de déplacer les données de la mémoire vers la mémoire magnétique. Lorsque les données atteignent la valeur configurée, Timestream for Live Analytics les déplace automatiquement vers le magasin magnétique. Vous pouvez également définir une valeur de rétention sur le magasin magnétique. Lorsque les données expirent hors de la mémoire magnétique, elles sont définitivement supprimées.

Par exemple, imaginez un scénario dans lequel vous configurez la mémoire pour contenir l'équivalent d'une semaine de données et la mémoire magnétique pour contenir l'équivalent d'un an de données. L'âge des données est calculé à l'aide de l'horodatage associé au point de données. Lorsque les données de la mémoire datent d'une semaine, elles sont automatiquement déplacées vers la mémoire magnétique. Elles sont ensuite conservées dans le stockage magnétique pendant un an. Lorsque les données datent d'un an, elles sont supprimées de Timestream pour Live Analytics. Les valeurs de rétention de la mémoire et de la mémoire magnétique définissent de manière cumulative la durée pendant laquelle vos données seront stockées dans Timestream pour Live Analytics. Cela signifie que pour le scénario ci-dessus, à compter de leur arrivée, les données sont stockées dans Timestream pour Live Analytics pendant une période totale d'un an et d'une semaine.

> **Note**
>
> Lorsque vous augmentez la durée de conservation de la mémoire ou de la mémoire magnétique, la modification de rétention prend effet à partir de ce moment. Par exemple, si la période de conservation de la mémoire a été définie sur 2 heures, puis modifiée à 24 heures en mettant à jour les politiques de conservation des tables, la mémoire sera capable de contenir 24 heures de données, mais sera remplie de 24 heures de données 22 heures après cette modification. Timestream for Live Analytics ne récupère pas les données de la mémoire magnétique pour alimenter la mémoire.

Pour garantir la sécurité de vos données de séries chronologiques, vos données dans Timestream for Live Analytics sont toujours cryptées par défaut. Cela s'applique aux données en transit et au repos. En outre, Timestream for Live Analytics vous permet d'utiliser des clés gérées par le client pour sécuriser vos données dans le magasin magnétique. Pour plus d'informations sur les clés gérées par le client, consultez AWS KMS keys.

Requêtes

Avec Timestream for Live Analytics, vous pouvez facilement stocker et analyser des métriques DevOps, des données de capteurs pour les applications IoT et des données de télémétrie industrielle pour la maintenance des équipements, ainsi que pour de nombreux autres cas d'utilisation. Le moteur de requête adaptatif spécialement conçu dans Timestream for Live Analytics vous permet d'accéder aux données à travers les niveaux de stockage à l'aide d'une seule instruction. SQL Il accède aux données et les combine de manière transparente entre les niveaux de stockage sans que vous ayez à spécifier l'emplacement des données. Vous pouvez l'utiliser SQL pour interroger des données dans Timestream for Live Analytics afin de récupérer des données de séries chronologiques

à partir d'une ou de plusieurs tables. Vous pouvez accéder aux informations de métadonnées des bases de données et des tables. Timestream for Live Analytics prend SQL également en charge les fonctions intégrées d'analyse des séries chronologiques. Vous pouvez vous référer à la Référence du langage de requête référence pour plus de détails.

Timestream for Live Analytics est conçu pour disposer d'une architecture d'ingestion, de stockage et de requête de données totalement découplée, dans laquelle chaque composant peut évoluer indépendamment des autres composants (ce qui lui permet d'offrir une évolutivité pratiquement infinie pour répondre aux besoins d'une application). Cela signifie que Timestream for Live Analytics ne « bascule » pas lorsque vos applications envoient des centaines de téraoctets de données par jour ou exécutent des millions de requêtes traitant de petites ou grandes quantités de données. À mesure que vos données augmentent au fil du temps, la latence des requêtes dans Timestream for Live Analytics reste pratiquement inchangée. Cela est dû au fait que l'architecture de requêtes Timestream for Live Analytics peut tirer parti d'un parallélisme massif pour traiter des volumes de données plus importants et s'adapter automatiquement aux besoins de débit de requêtes d'une application.

Modèle de données

Timestream prend en charge deux modèles de données pour les requêtes : le modèle plat et le modèle de série chronologique.

> **Note**
>
> Les données de Timestream sont stockées à l'aide du modèle plat et il s'agit du modèle par défaut pour les requêtes de données. Le modèle de série chronologique est un concept basé sur le temps de requête et est utilisé pour l'analyse de séries chronologiques.

- Modèle plat
- Modèle de série chronologique

Modèle plat

Le modèle plat est le modèle de données par défaut de Timestream pour les requêtes. Il représente les données de séries chronologiques sous forme de tableau. Les noms des dimensions, l'heure, les noms des mesures et les valeurs des mesures apparaissent sous forme de colonnes. Chaque ligne du tableau est un point de données atomique correspondant à une mesure effectuée à un

moment précis au sein d'une série chronologique. Les bases de données, les tables et les colonnes Timestream sont soumises à certaines contraintes de dénomination. Ils sont décrits dansService Limits.

Le tableau ci-dessous montre un exemple illustrant la manière dont Timestream stocke les données représentant l'CPUutilisation, l'utilisation de la mémoire et l'activité réseau des EC2 instances, lorsque les données sont envoyées sous forme d'enregistrement à mesure unique. Dans ce cas, les dimensions sont la région, la zone de disponibilité, le cloud privé virtuel et l'instance IDs des EC2 instances. Les mesures sont l'CPUutilisation, l'utilisation de la mémoire et les données réseau entrantes pour les EC2 instances. Les colonnes region, az, vpc et instance_id contiennent les valeurs des dimensions. L'heure de la colonne contient l'horodatage de chaque enregistrement. La colonne measure_name contient les noms des mesures représentées par cpu-utilization, memory_utilization et network_bytes_in. Les colonnes measure_value : :double contiennent les mesures émises sous forme de doubles (par exemple, CPU l'utilisation et l'utilisation de la mémoire). La colonne measure_value : :bigint contient les mesures émises sous forme d'entiers, par exemple les données réseau entrantes.

Heure	region	az	vpc	instance_id	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilisation du processeur	35,0	null
2019-12-04 19:00:01.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilisation du processeur	38,2	null
2019-12-04 19:00:02	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilisation du processeur	45,3	null

Heure	region	az	vpc	instance_id	nom_mesure	valeur_mesure : double	value_mesure : :bigint
000 000000							
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	54,9	null
2019-12-04 19:00:01 000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	42,6	null
2019-12-04 19:00:02 000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	33,3	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	octets_réseau	34 400	null
2019-12-04 19:00:01 000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	octets_réseau	1 500	null

Heure	region	az	vpc	instance_id	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:02.000000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	octets_réseau	6 000	null

Le tableau ci-dessous montre un exemple illustrant la manière dont Timestream stocke les données représentant l'CPUutilisation, l'utilisation de la mémoire et l'activité réseau des EC2 instances, lorsque les données sont envoyées sous forme d'enregistrement multi-mesures.

Heure	region	az	vpc	instance_id	nom_mesure	utilisation du processeur	memory_utilization	octets_réseau
2019-12-04 19:00:00.000000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcde0	metrics	35,0	54,9	34 400
2019-12-04 19:00:01.000000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcde0	metrics	38,2	42,6	1 500
2019-12-04 19:00:02.000000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcde0	metrics	45,3	33,3	6 600

Modèle de série chronologique

Le modèle de série chronologique est une construction temporelle de requête utilisée pour l'analyse de séries chronologiques. Il représente les données sous la forme d'une séquence ordonnée de paires (temps, valeur de mesure). Timestream prend en charge les fonctions de séries chronologiques telles que l'interpolation pour vous permettre de combler les lacunes de vos données. Pour utiliser ces fonctions, vous devez convertir vos données dans le modèle de série chronologique à l'aide de fonctions telles que create_time_series. Reportez-vous à Référence du langage de requête pour plus de détails.

À l'aide de l'exemple précédent de l'EC2instance, voici les données CPU d'utilisation exprimées sous forme de série chronologique.

region	az	vpc	instance_id	utilisation du processeur
us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567 890abcdef0	[{heure : 2019-12-04 19:00:00.000 000000, valeur : 35}, {heure : 2019-12-04 19:00:01.000 000000, valeur : 38.2}, {heure : 2019-12-04 19:00:02.000 000000, valeur : 45,3}]

Requêtes planifiées

La fonctionnalité de requêtes planifiées d'Amazon Timestream for Live Analytics est une solution entièrement gérée, évolutive et sans serveur permettant de calculer et de stocker des agrégats, des cumuls et d'autres formes de données prétraitées généralement utilisées pour alimenter les tableaux de bord opérationnels, les rapports commerciaux, les analyses ad hoc et d'autres applications. Les requêtes planifiées rendent les analyses en temps réel plus performantes et plus rentables, ce qui

vous permet de tirer des informations supplémentaires de vos données et de continuer à prendre de meilleures décisions commerciales.

Pour plus d'informations sur les requêtes planifiées, consultezUtilisation de requêtes planifiées dans Timestream pour LiveAnalytics.

Unité de calcul Timestream () TCU

Amazon Timestream for Live Analytics mesure la capacité de calcul qui vous est allouée pour répondre à vos besoins de requête dans l'unité de calcul Timestream (). TCU L'un TCU comprend 4 vCPUs et 16 Go de mémoire. Lorsque vous exécutez des requêtes dans Timestream for Live Analytics, le service alloue TCUs à la demande en fonction de la complexité de vos requêtes et de la quantité de données traitées. La quantité TCUs consommée par une requête détermine le coût associé.

> **Note**
>
> Toutes Comptes AWS les personnes intégrées au service après le 29 avril 2024 seront utilisées par défaut TCUs pour la tarification des requêtes.

Dans cette rubrique

MaxQuery TCU

Ce paramètre indique le nombre maximal d'unités de calcul que le service utilisera à tout moment pour répondre à vos requêtes. Pour exécuter des requêtes, vous devez définir la capacité minimale à 4TCUs. Vous pouvez définir le nombre maximum TCUs de multiples de 4, par exemple 4, 8, 16,

32, etc. Vous êtes facturé uniquement pour les ressources informatiques que vous utilisez pour votre charge de travail. Par exemple, si vous définissez le maximum TCUs sur 128, mais que vous n'en utilisez systématiquement que 8TCUs. Vous ne serez débité que pour la durée pendant laquelle vous avez utilisé le 8TCUs. La valeur par défaut `MaxQueryTCU` de votre compte est définie sur 200. Vous pouvez régler `MaxQueryTCU` de 4 à 1 000 en utilisant l'UpdateAccountSettingsAPIopération AWS Management Console ou avec le AWS SDK ou AWS CLI.

Nous vous recommandons de définir le `MaxQueryTCU` pour votre compte. La définition d'une TCU limite maximale permet de contrôler les coûts en limitant le nombre d'unités de calcul que le service peut utiliser pour la charge de travail de vos requêtes. Cela vous permet de mieux prévoir et gérer les dépenses liées à vos requêtes.

Facturation pour TCU

Chacune TCU est facturée sur une base horaire avec une granularité à la seconde et pour un minimum de 30 secondes. L'unité d'utilisation de ces unités de calcul est TCU -hour.

Lorsque vous exécutez des requêtes, vous êtes facturé pour l'TCUsutilisation pendant le temps d'exécution de la requête, mesuré en heuresTCU. Par exemple :

- Votre charge de travail en consomme 20 TCUs pendant 3 heures. 60 TCU heures (20 TCUs x 3 heures) vous sont facturées.
- Votre charge de travail en utilise 10 TCUs pendant 30 minutes, puis 20 TCUs pendant les 30 minutes suivantes. 15 TCU heures vous sont facturées (10 TCUs x 0,5 heure + 20 TCUs x 0,5 heure).

Le prix par TCU heure varie selon Région AWS. Consultez les tarifs d'Amazon Timestream pour plus de détails. À mesure que votre charge de travail augmente, le service adapte automatiquement la capacité de calcul jusqu'à la TCU limite maximale spécifiée (`MaxQueryTCU`) afin de maintenir des performances constantes. Le `MaxQueryTCU` paramètre agit comme un plafond pour la capacité de calcul que le service peut atteindre. Ce paramètre vous permet de contrôler le nombre de ressources de calcul et, par conséquent, leur coût.

Configuration TCU

Lorsque vous embarquez dans le service, la `MaxQueryTCU` limite par défaut de 200 pour chacun d'entre eux Compte AWS est fixée à 200. Vous pouvez mettre à jour cette limite selon les besoins à tout moment à l'aide de l'UpdateAccountSettingsAPIopération AWS Management Console ou avec le AWS SDK ou AWS CLI.

Si vous n'êtes pas sûr des valeurs à configurer, surveillez la `QueryTCU` métrique de votre compte. Cette métrique est disponible sur Amazon AWS Management Console et sur Amazon CloudWatch. Cette métrique donne un aperçu du nombre maximum d'utilisateurs TCUs à une granularité minute. Sur la base des données historiques et de votre estimation de la croissance future, configurez le `MaxQueryTCU` en fonction des pics d'utilisation. Nous vous recommandons d'avoir une marge de manœuvre d'au moins 4 à 16 % TCUs au-dessus de votre consommation maximale. Par exemple, si votre pic `QueryTCU` au cours des 30 derniers jours était de 128, nous vous recommandons de le régler `MaxQueryTCU` entre 132 et 144.

Estimation des unités de calcul requises

Les unités de calcul peuvent traiter les requêtes simultanément. Pour déterminer le nombre d'unités de calcul requises, prenez en compte les directives générales du tableau suivant :

Requêtes simultanées	TCUs
7	4
14	8
21	12

Note

- Il s'agit de directives générales et le nombre réel d'unités de calcul requises dépend de plusieurs facteurs, tels que :
  - La simultanéité effective des requêtes.
  - Modèles de requêtes.
  - Le nombre de partitions scannées.
  - Autres caractéristiques spécifiques à la charge de travail.
- Cette directive concerne les requêtes qui analysent des données pendant les dernières minutes ou une heure et respectent les meilleures pratiques en matière de requêtes Timestream et les directives de modélisation des données.
- Surveillez les performances de votre application et la `QueryTCU` métrique pour ajuster les unités de calcul, selon les besoins.

Quand augmenter MaxQuery TCU

Vous devriez envisager d'augmenter le `MaxQueryTCU` dans les scénarios suivants :

- Votre consommation maximale de requêtes approche ou atteint le maximum de requêtes actuellement configuréTCU. Nous vous recommandons de définir une valeur de requête maximale TCUs supérieure d'TCUau moins 4 à 16 % à votre consommation maximale.
- Vos requêtes renvoient une erreur 4xx avec le message MaxQuery TCU dépassé. Si vous prévoyez une augmentation prévue de votre charge de travail, revoyez et ajustez la requête maximale configurée TCU en conséquence.

Quand diminuer MaxQuery TCU

Vous devriez envisager de réduire la `MaxQueryTCU` valeur dans les scénarios suivants :

- Votre charge de travail suit un modèle d'utilisation prévisible et stable, et vous avez une bonne compréhension de vos besoins en matière d'utilisation du calcul. En abaissant le nombre maximal TCU de requêtes à 4 à 16 points TCU au-dessus de votre consommation maximale, vous pouvez éviter une utilisation et des coûts involontaires. Vous pouvez modifier la valeur à l'aide de UpdateAccountSettingsAPIcette opération.
- L'utilisation maximale de votre charge de travail a diminué au fil du temps, soit en raison de modifications apportées à votre application, soit en raison du comportement des utilisateurs. L'abaissement du maximum TCU peut contribuer à atténuer les coûts involontaires.

> **Note**
>
> En fonction de votre utilisation actuelle, la réduction de la modification de la TCU limite maximale peut prendre jusqu'à 24 heures pour être effective. Vous êtes facturé uniquement pour TCUs ce que vos requêtes consomment réellement. Le fait d'avoir une TCU limite maximale de requêtes plus élevée n'a aucune incidence sur vos coûts, sauf si ceux-ci TCUs sont utilisés par votre charge de travail.

Surveillance de l'utilisation à l'aide de CloudWatch métriques

Pour surveiller votre TCU utilisation, Timestream for Live Analytics fournit la CloudWatch métrique suivante :. `QueryTCU` Cette métrique indique le nombre d'unités de calcul utilisées par minute et est

émise toutes les minutes. Vous pouvez choisir de contrôler le maximum et le minimum TCUs utilisés en une minute. Vous pouvez également définir des alarmes sur cette métrique pour suivre les coûts de vos requêtes en temps réel.

Comprendre les variations dans l'utilisation des unités de calcul

Le nombre de ressources de calcul requises pour vos requêtes peut augmenter ou diminuer en fonction de plusieurs paramètres. Par exemple, le volume de données, les modèles d'ingestion de données, la latence des requêtes, la forme des requêtes, l'efficacité des requêtes et les combinaisons de requêtes utilisant des requêtes analytiques et en temps réel. Ces paramètres peuvent entraîner une augmentation ou une diminution des TCU unités requises pour votre charge de travail. Dans un état stable où ces paramètres ne changent pas, vous pouvez constater que le nombre d'unités de calcul requises pour votre charge de travail diminue. Par conséquent, cela peut réduire vos coûts mensuels.

En outre, si l'un de ces paramètres de votre charge de travail ou de vos données change, le nombre d'unités de calcul requises peut augmenter. Lorsque Timestream reçoit une requête, en fonction des partitions de données auxquelles la requête accède, Timestream décide du nombre de ressources de calcul nécessaires pour répondre efficacement à la requête.

À intervalles réguliers, en fonction de vos habitudes d'ingestion et d'accès aux requêtes, Timestream optimise la mise en page des données. Timestream effectue l'optimisation en regroupant les partitions les moins consultées en une seule partition ou en divisant une partition active en plusieurs partitions pour des raisons de performance. Par conséquent, la capacité de calcul utilisée par la même requête peut varier légèrement à différents moments.

> **Accepter d'utiliser la TCU tarification pour vos requêtes**
>
> En tant qu'utilisateur existant, vous pouvez vous inscrire une seule fois afin d'améliorer la gestion des coûts et de supprimer le nombre minimum d'octets mesurés par requête. TCUs Vous pouvez vous inscrire en utilisant l'UpdateAccountSettingsAPIopération AWS Management Console ou avec le AWS SDK ou AWS CLI. Au cours de l'APIopération, définissez le `QueryPricingModel` paramètre sur`COMPUTE_UNITS`.
> L'adoption du modèle de tarification basé sur le calcul constitue un changement irréversible.

Accès à Timestream pour LiveAnalytics

Vous pouvez accéder à Timestream pour LiveAnalytics utiliser la console CLI ou le. API Pour plus d'informations sur l'accès à Timestream pour LiveAnalytics, consultez les points suivants :

Rubriques

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d‘inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l‘utilisateur racine a accès à l‘ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l‘utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Une fois que vous vous êtes inscrit à un utilisateur administratif Compte AWS, que vous Utilisateur racine d'un compte AWS l'avez sécurisé AWS IAM Identity Center, que vous l'avez activé et que vous en avez créé un, afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur root.

   Pour obtenir des instructions, voir Activer un MFA périphérique virtuel pour votre utilisateur Compte AWS root (console) dans le guide de IAM l'utilisateur.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu'utilisateur doté d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URLidentifiant envoyé à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAMIdentity Center, consultez la section Connexion au portail AWS d'accès dans le guide de Connexion à AWS l'utilisateur.

Attribution d'un accès à d'autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme à la meilleure pratique consistant à appliquer les autorisations du moindre privilège.

Pour obtenir des instructions, consultez Création d'un ensemble d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

   Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l'utilisateur AWS IAM Identity Center .

Fournir un Timestream pour l'accès LiveAnalytics

Les autorisations requises pour accéder à Timestream pour LiveAnalytics sont déjà accordées à l'administrateur. Pour les autres utilisateurs, vous devez leur accorder un LiveAnalytics accès Timestream conformément à la politique suivante :

```
{
  "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**
>
> Pour plus d'informations sur `dbqms`, voir Actions, ressources et clés de condition pour le service de métadonnées de requête de base de données. Pour plus d'informations sur la `kms` section Actions, ressources et clés de condition pour le service de gestion des AWS clés.

Octroi d’un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
Identité de la main-d’œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées	Suivez les instructions de la section Utilisation d'informations d'identification temporair

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
	au AWS CLI AWS SDKs, ou AWS APIs.	es avec les AWS ressources du Guide de IAM l'utilisateur.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations IAM d'identif ication utilisateur dans le Guide de AWS Command Line Interface l'utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour IAM les utilisateurs dans le Guide de IAM l'utilisateur.

Utilisation de la console

Vous pouvez utiliser la console AWS de gestion de Timestream Live Analytics pour créer, modifier, supprimer, décrire et répertorier les bases de données et les tables. Vous pouvez également utiliser la console pour exécuter des requêtes.

Rubriques

- Didacticiel

- Créer une base de données
- Créer une table
- Exécuter une requête
- Création d'une requête planifiée
- Supprimer une requête planifiée
- Supprimer une table
- Supprimer une base de données
- Modifier un tableau
- Modifier une base de données

Didacticiel

Ce didacticiel explique comment créer une base de données contenant des exemples de jeux de données et exécuter des exemples de requêtes. Les exemples de jeux de données utilisés dans ce didacticiel sont fréquemment utilisés dans l'IoT et les DevOps scénarios. Le jeu de données IoT contient des séries chronologiques telles que la vitesse, l'emplacement et le chargement d'un camion, afin de rationaliser la gestion du parc et d'identifier les opportunités d'optimisation. Le DevOps jeu de données contient des mesures d'EC2instance telles que CPU l'utilisation du réseau et de la mémoire afin d'améliorer les performances et la disponibilité des applications. Voici un didacticiel vidéo pour les instructions décrites dans cette section

Procédez comme suit pour créer une base de données contenant les exemples d'ensembles de données et exécuter des exemples de requêtes à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Databases
3. Cliquez sur Créer une base de données.
4. Sur la page de création de base de données, entrez ce qui suit :
    - Choisissez la configuration —Sélectionnez un exemple de base de données.
    - Nom —Entrez le nom de base de données de votre choix.
    - Choisissez des exemples de jeux de données —Sélectionnez IoT et. DevOps
    - Cliquez sur Créer une base de données pour créer une base de données contenant deux tables : IoT et DevOps contenant des exemples de données.

5. Dans le volet de navigation, choisissez Éditeur de requêtes
6. Sélectionnez Exemples de requêtes dans le menu supérieur.
7. Cliquez sur l'un des exemples de requêtes. Cela vous ramènera à l'éditeur de requêtes avec l'éditeur rempli avec l'exemple de requête.
8. Cliquez sur Exécuter pour exécuter la requête et voir les résultats de la requête.

Créer une base de données

Procédez comme suit pour créer une base de données à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Databases
3. Cliquez sur Créer une base de données.
4. Sur la page de création de base de données, entrez ce qui suit.
   - Choisissez la configuration —Sélectionnez la base de données standard.
   - Nom —Entrez le nom de base de données de votre choix.
   - Chiffrement : choisissez une KMS clé ou utilisez l'option par défaut, dans laquelle Timestream Live Analytics créera une KMS clé dans votre compte s'il n'en existe pas déjà une.
5. Cliquez sur Créer une base de données pour créer une base de données.

Créer une table

Procédez comme suit pour créer une table à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Tables
3. Cliquez sur Créer une table.
4. Sur la page de création d'une table, entrez ce qui suit.
   - Nom de la base de données —Sélectionnez le nom de la base de données créée dansCréer une base de données .
   - Nom de table —Entrez le nom de table de votre choix.
   - Conservation de la mémoire : spécifiez la durée pendant laquelle vous souhaitez conserver les données dans la mémoire. La mémoire de stockage traite les données entrantes, y compris les

données arrivées tardivement (données dont l'horodatage est antérieur à l'heure actuelle) et est optimisée pour les requêtes rapides point-in-time.

- Conservation du stockage magnétique : spécifiez la durée pendant laquelle vous souhaitez conserver les données dans le stockage magnétique. Le magasin magnétique est conçu pour le stockage à long terme et est optimisé pour les requêtes analytiques rapides.

5. Cliquez sur Créer une table.

Exécuter une requête

Procédez comme suit pour exécuter des requêtes à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, choisissez Éditeur de requêtes
3. Dans le volet gauche, sélectionnez la base de données créée dansCréer une base de données .
4. Dans le volet gauche, sélectionnez la base de données créée dansCréer une table .
5. Dans l'éditeur de requêtes, vous pouvez exécuter une requête. Pour voir les 10 dernières lignes du tableau, exécutez :

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

6. (Facultatif) Activez Enable Insights pour obtenir des informations sur l'efficacité de vos requêtes.

Création d'une requête planifiée

Procédez comme suit pour créer une requête planifiée à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Requêtes planifiées.
3. Cliquez sur Créer une requête planifiée.
4. Dans les sections Nom de la requête et Table de destination, entrez les informations suivantes.

   - Nom —Entrez le nom de la requête.
   - Nom de la base de données —Sélectionnez le nom de la base de données créée dansCréer une base de données .
   - Nom de la table —Sélectionnez le nom de la table créée dansCréer une table .

5. Dans la section Déclaration de requête, entrez une instruction de requête valide. Cliquez ensuite sur Valider la requête.
6. Dans Modèle de table de destination, définissez le modèle pour tous les attributs non définis. Vous pouvez utiliser Visual Builder ouJSON.
7. Dans la section Exécuter le calendrier, choisissez Taux fixe ou expression Chron. Pour les expressions chronologiques, reportez-vous à Expressions de planification pour les requêtes planifiées pour plus de détails sur les expressions de planification.
8. Dans la section du SNSsujet, entrez le SNS sujet qui sera utilisé pour la notification.
9. Dans la section Rapport du journal des erreurs, entrez l'emplacement S3 qui sera utilisé pour signaler les erreurs.

   Choisissez Encryption key type (Type de clé de chiffrement).
10. Dans la section Paramètres de sécurité de la AWS KMSclé, choisissez le type de AWS KMS clé.

    Entrez le IAMrôle que Timestream for LiveAnalytics utilisera pour exécuter la requête planifiée. Reportez-vous aux exemples IAM de politique pour les requêtes planifiées pour plus de détails sur les autorisations requises et la relation de confiance pour le rôle.
11. Cliquez sur Créer une requête planifiée.

Supprimer une requête planifiée

Procédez comme suit pour supprimer ou désactiver une requête planifiée à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Requêtes planifiées
3. Sélectionnez la requête planifiée créée dansCréation d'une requête planifiée.
4. Sélectionnez Actions.
5. Choisissez Désactiver ou Supprimer.
6. Si vous avez sélectionné Supprimer, confirmez l'action et sélectionnez Supprimer.

Supprimer une table

Procédez comme suit pour supprimer une base de données à l'aide de la AWS console.

1. Ouvrez la AWS console.

2. Dans le volet de navigation, sélectionnez Tables
3. Sélectionnez la table que vous avez créée dansCréer une table .
4. Cliquez sur Delete.
5. Tapez Supprimer dans le champ de confirmation.

Supprimer une base de données

Pour supprimer une base de données à l'aide de la AWS console, procédez comme suit :

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Databases
3. Sélectionnez la base de données que vous avez créée dans Créer une base de données.
4. Cliquez sur Delete.
5. Tapez Supprimer dans le champ de confirmation.

Modifier un tableau

Procédez comme suit pour modifier un tableau à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Tables
3. Sélectionnez la table que vous avez créée dansCréer une table .
4. Cliquez sur Modifier
5. Modifiez les détails du tableau et enregistrez.

    - Conservation de la mémoire : spécifiez la durée pendant laquelle vous souhaitez conserver les données dans la mémoire. La mémoire de stockage traite les données entrantes, y compris les données arrivées tardivement (données dont l'horodatage est antérieur à l'heure actuelle) et est optimisée pour les requêtes rapides point-in-time.
    - Conservation du stockage magnétique : spécifiez la durée pendant laquelle vous souhaitez conserver les données dans le stockage magnétique. Le magasin magnétique est conçu pour le stockage à long terme et est optimisé pour les requêtes analytiques rapides.

Modifier une base de données

Procédez comme suit pour modifier une base de données à l'aide de la AWS console.

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Databases
3. Sélectionnez la base de données que vous avez créée dans Créer une base de données.
4. Cliquez sur Modifier
5. Modifiez les détails de la base de données et enregistrez.

Accès à Amazon Timestream LiveAnalytics pour utiliser le AWS CLI

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour contrôler plusieurs AWS services à partir de la ligne de commande et les automatiser par le biais de scripts. Vous pouvez utiliser le AWS CLI pour des opérations ad hoc. Vous pouvez également l'utiliser pour intégrer Amazon LiveAnalytics Timestream à des fins d'opérations dans des scripts utilitaires.

Avant de pouvoir utiliser le AWS CLI avec Timestream pour LiveAnalytics, vous devez configurer l'accès programmatique. Pour de plus amples informations, veuillez consulter Octroi d’un accès par programmation.

Pour une liste complète de toutes les commandes disponibles pour le Timestream for LiveAnalytics Query API in the AWS CLI, consultez la référence des AWS CLI commandes.

Pour une liste complète de toutes les commandes disponibles pour le Timestream pour LiveAnalytics Write API in the AWS CLI, consultez la référence des AWS CLI commandes.

Rubriques

- Téléchargement et configuration de la AWS CLI
- Utilisation du AWS CLI avec Timestream pour LiveAnalytics

Téléchargement et configuration de la AWS CLI

Il AWS CLI fonctionne sous Windows, macOS ou Linux. Pour le télécharger, l'installer et le configurer, procédez comme suit :

1. Téléchargez-le AWS CLI à l'adresse http://aws.amazon.com/cli.

2. Suivez les instructions d'installation AWS CLI et de configuration du guide AWS CLI de l'AWS Command Line Interface utilisateur.

Utilisation du AWS CLI avec Timestream pour LiveAnalytics

Le format de ligne de commande consiste en un Amazon Timestream LiveAnalytics pour le nom de l'opération, suivi des paramètres de cette opération. Le AWS CLI prend en charge une syntaxe abrégée pour les valeurs des paramètres, en plus de. JSON

Permet `help` de répertorier toutes les commandes disponibles dans Timestream pour. LiveAnalytics Par exemple :

```
aws timestream-write help
```

```
aws timestream-query help
```

Vous pouvez également utiliser `help` pour décrire une commande spécifique et en savoir plus sur son utilisation :

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

Par exemple, pour créer une base de données :

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

Pour créer une table avec les écritures magnétiques activées :

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

Pour écrire des données à l'aide d'enregistrements à mesure unique :

```
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\"}]"
```

Pour écrire des données à l'aide d'enregistrements à mesures multiples :

```
# 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
```

Pour interroger une table :

```
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)"
```

Pour créer une requête planifiée :

```
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\"\
```

```
        }\
      }"
```

À l'aide du API

En outre SDKs, Amazon Timestream LiveAnalytics pour fournit un accès REST API direct via le modèle de découverte des terminaux. Le modèle de découverte des terminaux est décrit ci-dessous, ainsi que ses cas d'utilisation.

Le modèle de découverte des terminaux

Les Timestream Live Analytics étant SDKs conçus pour fonctionner de manière transparente avec l'architecture du service, y compris la gestion et le mappage des points de terminaison du service, il est recommandé de les utiliser pour la SDKs plupart des applications. Cependant, dans certains cas, l'utilisation du modèle Timestream pour la découverte des LiveAnalytics REST API terminaux est nécessaire :

- Vous utilisez VPCendpoints (AWS PrivateLink) avec Timestream pour LiveAnalytics
- Votre application utilise un langage de programmation qui n'est pas encore pris SDK en charge
- Vous avez besoin d'un meilleur contrôle de la mise en œuvre côté client

Cette section contient des informations sur le fonctionnement du modèle de découverte des points de terminaison, comment implémenter le modèle de découverte des points de terminaison, ainsi que des notes d'utilisation. Sélectionnez un sujet ci-dessous pour en savoir plus.

Rubriques

- Comment fonctionne le modèle de découverte des terminaux
- Implémentation du modèle de découverte des terminaux

Comment fonctionne le modèle de découverte des terminaux

Timestream est construit à l'aide d'une architecture cellulaire pour garantir une meilleure évolutivité et des propriétés d'isolation du trafic. Chaque compte client étant mappé à une cellule spécifique d'une région, votre application doit utiliser les points de terminaison spécifiques à la cellule auxquels votre compte a été mappé. Lorsque vous utilisez leSDKs, ce mappage est géré de manière transparente pour vous et vous n'avez pas besoin de gérer les points de terminaison spécifiques aux cellules. Toutefois, lorsque vous accédez directement au RESTAPI, vous devrez gérer et cartographier vous-

même les points de terminaison appropriés. Ce processus, le modèle de découverte des terminaux, est décrit ci-dessous :

1. Le modèle de découverte des terminaux commence par un appel à l'`DescribeEndpoints`action (décrit dans la `DescribeEndpoints`section).
2. Le point de terminaison doit être mis en cache et réutilisé pendant la durée spécifiée par la valeur renvoyée time-to-live (TTL) (the `CachePeriodInMinutes`). Les appels au Timestream Live Analytics API peuvent ensuite être effectués pendant la durée du. TTL
3. Après l'TTLexpiration, un nouvel appel DescribeEndpoints doit être effectué pour actualiser le point de terminaison (en d'autres termes, recommencer à zéro à l'étape 1).

> **Note**
>
> La syntaxe, les paramètres et les autres informations d'utilisation de l'`DescribeEndpoints`action sont décrits dans la APIréférence. Notez que l'`DescribeEndpoints`action est disponible via les deux et qu'SDKselle est identique pour chacune d'entre elles.

Pour la mise en œuvre du modèle de découverte des terminaux, voirImplémentation du modèle de découverte des terminaux.

Implémentation du modèle de découverte des terminaux

Pour implémenter le modèle de découverte du point de terminaison, choisissez un API (écriture ou requête), créez une DescribeEndpointsdemande et utilisez le ou les points de terminaison renvoyés pendant la durée de la ou des TTL valeurs renvoyées. La procédure de mise en œuvre est décrite ci-dessous.

> **Note**
>
> Assurez-vous de bien connaître les notes d'utilisation.

Procédure de mise en œuvre

1. Obtenez le point de terminaison pour lequel API vous souhaitez passer des appels (Write ou Query) à l'aide de la `DescribeEndpoints`demande.

a. Créez une demande correspondant à l'APIobjet `DescribeEndpoints`qui vous intéresse (écriture ou requête) en utilisant l'un des deux points de terminaison décrits ci-dessous. Il n'y a aucun paramètre d'entrée pour la demande. Assurez-vous de lire les notes ci-dessous.

   Écrivez SDK :

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

   Requête SDK :

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

   Voici un exemple CLI d'appel pour une `us-east-1` région.

   ```
   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**
   >
   > L'en-tête HTTP « Host » doit également contenir le API point de terminaison. La demande échouera si l'en-tête n'est pas renseigné. Il s'agit d'une exigence standard pour toutes les requêtes HTTP /1.1. Si vous utilisez une HTTP bibliothèque compatible avec la version 1.1 ou une version ultérieure, la HTTP bibliothèque doit automatiquement remplir l'en-tête pour vous.

   > **Note**
   >
   > Substitut *<region>* avec l'identifiant de région pour la région dans laquelle la demande est faite, par ex. `us-east-1`

b. Analysez la réponse pour extraire le ou les points de terminaison et les TTL valeurs du cache. La réponse est un tableau d'un ou de plusieurs `Endpoint`objets. Chaque `Endpoint`

objet contient une adresse de point de terminaison (`Address`) et celle TTL pour ce point de terminaison (`CachePeriodInMinutes`).

2. Mettez en cache le point de terminaison jusqu'à la valeur spécifiéeTTL.
3. À l'TTLexpiration, récupérez un nouveau point de terminaison en recommençant à l'étape 1 de l'implémentation.

Remarques d'utilisation pour le modèle de découverte des terminaux

- Il s'agit de la seule action reconnue par les points de terminaison régionaux de Timestream Live Analytics. DescribeEndpoints
- La réponse contient une liste de points de terminaison auxquels effectuer des appels Timestream Live AnalyticsAPI.
- En cas de réponse satisfaisante, il doit y avoir au moins un point final dans la liste. S'il y a plusieurs points de terminaison dans la liste, chacun d'entre eux est également utilisable pour les API appels, et l'appelant peut choisir le point de terminaison à utiliser au hasard.
- Outre l'DNSadresse du point de terminaison, chaque point de terminaison de la liste spécifiera une durée de vie (TTL) autorisée pour utiliser le point de terminaison spécifié en minutes.
- Le point de terminaison doit être mis en cache et réutilisé pendant la durée spécifiée par la TTL valeur renvoyée (en minutes). Après l'TTLexpiration, un nouvel appel DescribeEndpointsdoit être effectué pour actualiser le point de terminaison à utiliser, car le point de terminaison ne fonctionnera plus après TTL son expiration.

À l'aide du AWS SDKs

Vous pouvez accéder à Amazon Timestream à l'aide du. AWS SDKs Timestream en prend en charge deux SDKs par langue, à savoir le Write SDK et le Query. SDK L'écriture SDK est utilisée pour effectuer des CRUD opérations et pour insérer les données de vos séries chronologiques dans Timestream. La requête SDK est utilisée pour interroger vos données de séries chronologiques existantes stockées dans Timestream.

Une fois que vous avez rempli les prérequis nécessaires pour votre SDK choix, vous pouvez commencer avec leExemples de code.

Rubriques

- Java

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

Java

Pour commencer à utiliser Java 1.0 SDK et Amazon Timestream, remplissez les conditions préalables décrites ci-dessous.

Une fois que vous avez rempli les prérequis nécessaires pour le JavaSDK, vous pouvez commencer avec leExemples de code.

Prérequis

Avant de commencer à utiliser Java, vous devez effectuer les opérations suivantes :

1. Suivez les instructions AWS de configuration indiquées dansAccès à Timestream pour LiveAnalytics.
2. Configurez un environnement de développement Java en téléchargeant et en installant les éléments suivants :
   - Kit de développement Java SE 8 (tel qu'Amazon Corretto 8).
   - Java IDE (comme Eclipse ou IntelliJ).

     Pour plus d'informations, consultez Getting Started with AWS SDK for Java
3. Configurez vos AWS informations d'identification et votre région pour le développement :
   - Configurez vos informations d'identification de AWS sécurité à utiliser avec le AWS SDK for Java.
   - Définissez votre AWS région pour déterminer votre flux temporel par défaut pour LiveAnalytics le point de terminaison.

Utilisation d'Apache Maven

Vous pouvez utiliser Apache Maven pour configurer et créer des AWS SDK for Java projets.

> **Note**
>
> Pour utiliser Apache Maven, assurez-vous que votre Java SDK et votre environnement d'exécution sont de version 1.8 ou supérieure.

Vous pouvez le configurer AWS SDK en tant que dépendance Maven, comme décrit dans Utilisation du SDK avec Apache Maven.

Vous pouvez exécuter la compilation et exécuter votre code source à l'aide de la commande suivante :

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

> **Note**
>
> `<your source code Main class>`est le chemin d'accès à la classe principale de votre code source Java.

Configuration de vos AWS informations d'identification

Vous devez fournir des AWS informations d'identification à votre application lors de l'exécution. AWS SDK for Java Les exemples de code présentés dans ce guide supposent que vous utilisez un fichier d' AWS informations d'identification, comme décrit dans la section Configurer les AWS informations d'identification et la région pour le développement dans le guide du AWS SDK for Java développeur.

Voici un exemple de fichier d' AWS informations d'identification nommé`~/.aws/credentials`, où le caractère tilde (~) représente votre répertoire personnel.

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

Java v2

Pour commencer à utiliser Java 2.0 SDK et Amazon Timestream, remplissez les conditions préalables décrites ci-dessous.

Une fois que vous avez rempli les prérequis nécessaires pour le Java 2.0SDK, vous pouvez commencer avec leExemples de code.

Prérequis

Avant de commencer à utiliser Java, vous devez effectuer les opérations suivantes :

1. Suivez les instructions AWS de configuration indiquées dansAccès à Timestream pour LiveAnalytics.
2. Vous pouvez le configurer AWS SDK en tant que dépendance Maven, comme décrit dans Utilisation du SDK avec Apache Maven.
3. Configurez un environnement de développement Java en téléchargeant et en installant les éléments suivants :

   - Kit de développement Java SE 8 (tel qu'Amazon Corretto 8).
   - Java IDE (comme Eclipse ou IntelliJ).

     Pour plus d'informations, consultez Getting Started with AWS SDK for Java

Utilisation d'Apache Maven

Vous pouvez utiliser Apache Maven pour configurer et créer des AWS SDK for Java projets.

> **Note**
>
> Pour utiliser Apache Maven, assurez-vous que votre Java SDK et votre environnement d'exécution sont de version 1.8 ou supérieure.

Vous pouvez le configurer AWS SDK en tant que dépendance Maven, comme décrit dans Utilisation du SDK avec Apache Maven. Les modifications nécessaires au fichier pom.xml sont décrites ici.

Vous pouvez exécuter la compilation et exécuter votre code source à l'aide de la commande suivante :

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

> **Note**
>
> `<your source code Main class>`est le chemin d'accès à la classe principale de votre code source Java.

Go

Pour commencer à utiliser Go SDK et Amazon Timestream, remplissez les conditions préalables décrites ci-dessous.

Une fois que vous avez rempli les prérequis nécessaires pour le GoSDK, vous pouvez commencer avec leExemples de code.

Prérequis

1. Téléchargez le GO SDK 1.14.
2. Configurez le GO SDK.
3. Construisez votre client.

Python

Pour commencer à utiliser Python SDK et Amazon Timestream, remplissez les conditions préalables décrites ci-dessous.

Une fois que vous avez rempli les prérequis nécessaires pour le PythonSDK, vous pouvez commencer avec leExemples de code.

Prérequis

Pour utiliser Python, installez et configurez Boto3 en suivant les instructions ici.

Node.js

Pour commencer à utiliser Node.js SDK et Amazon Timestream, remplissez les conditions préalables décrites ci-dessous.

Une fois que vous avez rempli les prérequis nécessaires pour le fichier Node.jsSDK, vous pouvez commencer avec leExemples de code.

Prérequis

Avant de commencer à utiliser Node.js, vous devez effectuer les opérations suivantes :

1. Installez Node.js.
2. Installez le AWS SDK pour JavaScript.

.NET

Pour commencer avec le. NETSDKet Amazon Timestream, remplissez les conditions préalables décrites ci-dessous.

Une fois que vous avez rempli les conditions préalables nécessaires pour le. NETSDK, vous pouvez commencer avec leExemples de code.

Prérequis

Avant de commencer avec. NET, installez les NuGet packages requis et assurez-vous que la version AWSSDK .Core est 3.3.107 ou plus récente en exécutant les commandes suivantes :

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

Mise en route

Cette section inclut un didacticiel pour vous aider à démarrer avec Amazon Timestream Live Analytics, ainsi que des instructions pour configurer un exemple d'application entièrement fonctionnel. Vous pouvez commencer à utiliser le didacticiel ou l'exemple d'application en sélectionnant l'un des liens ci-dessous.

Rubriques

- Didacticiel
- Exemple d'application

Didacticiel

Ce didacticiel explique comment créer une base de données contenant des exemples de jeux de données et exécuter des exemples de requêtes. Les exemples d'ensembles de données utilisés dans ce didacticiel sont fréquemment utilisés dans l'IoT et les DevOps scénarios. L'ensemble de données IoT contient des séries chronologiques telles que la vitesse, l'emplacement et le chargement d'un camion, afin de rationaliser la gestion du parc et d'identifier les opportunités d'optimisation. L'ensemble de DevOps données contient des métriques d'EC2instance telles que CPU l'utilisation du réseau et de la mémoire afin d'améliorer les performances et la disponibilité des applications. Voici un didacticiel vidéo pour les instructions décrites dans cette section.

Procédez comme suit pour créer une base de données contenant les exemples d'ensembles de données et exécuter des exemples de requêtes à l'aide de la AWS console :

Utilisation de la console

Procédez comme suit pour créer une base de données contenant les exemples d'ensembles de données et exécuter des exemples de requêtes à l'aide de la AWS console :

1. Ouvrez la AWS console.
2. Dans le volet de navigation, sélectionnez Databases
3. Cliquez sur Créer une base de données.
4. Sur la page de création de base de données, entrez ce qui suit :
   - Choisissez la configuration —Sélectionnez un exemple de base de données.
   - Nom —Entrez le nom de base de données de votre choix.

     > **Note**
     >
     > Après avoir créé une base de données avec des exemples de jeux de données, pour utiliser les exemples de requêtes disponibles dans la console, vous pouvez ajuster le nom de base de données référencé dans la requête pour qu'il corresponde au nom de base de données que vous entrez ici. Il existe des exemples de requêtes pour chaque combinaison d'ensembles de données d'échantillons et de types d'enregistrements de séries chronologiques.

   - Choisissez des exemples d'ensembles de données : sélectionnez IoT et. DevOps

- Choisissez le type d'enregistrements de séries chronologiques —Sélectionnez Enregistrements à mesures multiples.
- Cliquez sur Créer une base de données pour créer une base de données contenant deux tables remplies d'exemples de données. Les noms de table pour les exemples d'ensembles de données contenant des enregistrements à mesures multiples sont `DevOpsMulti` et`IoTMulti`. Les noms de table pour les exemples de jeux de données contenant des enregistrements à mesure unique sont `DevOps` et. `IoT`

5. Dans le volet de navigation, choisissez l'éditeur de requêtes
6. Sélectionnez Exemples de requêtes dans le menu supérieur.
7. Cliquez sur l'un des exemples de requêtes pour un ensemble de données que vous avez choisi lors de la création de l'exemple de base de données. Cela vous ramènera à l'éditeur de requêtes avec l'éditeur rempli avec l'exemple de requête.
8. Ajustez le nom de la base de données pour l'exemple de requête.
9. Cliquez sur Exécuter pour exécuter la requête et voir les résultats de la requête.

À l'aide du SDKs

Timestream Live Analytics fournit un exemple d'application entièrement fonctionnel qui vous montre comment créer une base de données et une table, remplir la table avec environ 126 000 lignes d'exemples de données et exécuter des exemples de requêtes. L'exemple d'application est disponible GitHubpour Java, Python, Node.js, Go et. NET.

1. Clonez les exemples d'applications du GitHub référentiel Timestream Live Analytics en suivant les instructions de. GitHub
2. Configurez le AWS SDK pour vous connecter à Amazon Timestream Live Analytics en suivant les instructions décrites dans. À l'aide du AWS SDKs
3. Compilez et exécutez l'exemple d'application en suivant les instructions ci-dessous :

   - Instructions relatives à l'exemple d'application Java.
   - Instructions pour l'exemple d'application Java v2.
   - Instructions pour l'exemple d'application Go.
   - Instructions pour l'exemple d'application Python.
   - Instructions pour l'exemple d'application Node.js.
   - Instructions pour le. NETexemple d'application.

Exemple d'application

Timestream est fourni avec un exemple d'application entièrement fonctionnel qui montre comment créer une base de données et une table, remplir la table avec environ 126 000 lignes d'échantillons de données et exécuter des exemples de requêtes. Suivez les étapes ci-dessous pour commencer à utiliser l'exemple d'application dans l'une des langues prises en charge :

Java

1. Clonez le GitHub référentiel Timestream pour obtenir des LiveAnalytics exemples d'applications en suivant les instructions de. GitHub
2. Configurez le AWS SDK pour vous connecter à Timestream en LiveAnalytics suivant les instructions décrites dans Getting Started with. Java
3. Exécutez l'exemple d'application Java en suivant les instructions décrites ici

Java v2

1. Clonez le GitHub référentiel Timestream pour obtenir des LiveAnalytics exemples d'applications en suivant les instructions de. GitHub
2. Configurez le AWS SDK pour vous connecter à Amazon Timestream en suivant LiveAnalytics les instructions décrites dans Getting Started with. Java v2
3. Exécutez l'exemple d'application Java 2.0 en suivant les instructions décrites ici

Go

1. Clonez le GitHub référentiel Timestream pour obtenir des LiveAnalytics exemples d'applications en suivant les instructions de. GitHub
2. Configurez le AWS SDK pour vous connecter à Amazon Timestream en suivant LiveAnalytics les instructions décrites dans Getting Started with. Go
3. Exécutez l'exemple d'application Go en suivant les instructions décrites ici

Python

1. Clonez le GitHub référentiel Timestream pour obtenir des LiveAnalytics exemples d'applications en suivant les instructions de. GitHub

2. Configurez le AWS SDK pour vous connecter à Amazon Timestream en suivant LiveAnalytics les instructions décrites dans. Python
3. Exécutez l'exemple d'application Python en suivant les instructions décrites ici

Node.js

1. Clonez le GitHub référentiel Timestream pour obtenir des LiveAnalytics exemples d'applications en suivant les instructions de. GitHub
2. Configurez le AWS SDK pour vous connecter à Amazon Timestream en suivant LiveAnalytics les instructions décrites dans Getting Started with. Node.js
3. Exécutez l'exemple d'application Node.js en suivant les instructions décrites ici

.NET

1. Clonez le GitHub référentiel Timestream pour obtenir des LiveAnalytics exemples d'applications en suivant les instructions de. GitHub
2. Configurez le AWS SDK pour vous connecter à Amazon Timestream en suivant LiveAnalytics les instructions décrites dans Getting Started with. .NET
3. Exécutez le. NETexemple d'application en suivant les instructions décrites ici

Exemples de code

Vous pouvez accéder à Amazon Timestream à l'aide du. AWS SDKs Timestream en prend en charge deux SDKs par langue, à savoir le Write SDK et le Query. SDK L'écriture SDK est utilisée pour effectuer des CRUD opérations et pour insérer les données de vos séries chronologiques dans Timestream. La requête SDK est utilisée pour interroger vos données de séries chronologiques existantes stockées dans Timestream. Sélectionnez un sujet dans la liste ci-dessous pour plus de détails, y compris des exemples de code pour chacun des sujets pris en chargeSDKs.

Rubriques

- SDKClient d'écriture
- SDKClient de requête
- Créer une base de données
- Décrire la base de

- Mettre à jour la base
- Supprimer la base de données
- Liste des bases de données
- Create table
- Décrire le tableau
- Mettre à jour une table
- Supprimer une table
- Répertorier des tables
- Écrire des données (insertions et insertions)
- Exécuter la requête
- Exécuter UNLOAD la requête
- Annuler la requête
- Créer une tâche de chargement par lots
- Décrire la tâche de chargement par lots
- Lister les tâches de chargement par lots
- Reprendre la tâche de chargement par lots
- Création d'une requête planifiée
- Répertorier une requête planifiée
- Décrire la requête planifiée
- Exécuter une requête planifiée
- Mettre à jour la requête planifiée
- Supprimer la requête planifiée

SDKClient d'écriture

Vous pouvez utiliser les extraits de code suivants pour créer un client Timestream pour le Write. SDK L'écriture SDK est utilisée pour effectuer des CRUD opérations et pour insérer les données de vos séries chronologiques dans Timestream.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Une importation de commande supplémentaire est présentée ici. L'`CreateDatabaseCommand`importation n'est pas requise pour créer le client.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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);
```

Nous vous recommandons d'utiliser la configuration suivante.

- Définissez le nombre SDK de nouvelles tentatives sur. `10`
- Utilisez `SDK DEFAULT_BACKOFF_STRATEGY`.
- `RequestTimeout`Réglé sur `20` secondes.
- Réglez le nombre maximum de connexions sur `5000` ou plus.

SDKClient de requête

Vous pouvez utiliser les extraits de code suivants pour créer un client Timestream pour la requête. SDK La requête SDK est utilisée pour interroger vos données de séries chronologiques existantes stockées dans Timestream.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, voir Timestream Query Client -,AWS SDK pour JavaScript la version 3.

Une importation de commande supplémentaire est présentée ici. L'`QueryCommand`importation n'est pas requise pour créer le client.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. GitHub

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

.NET

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

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

Créer une base de données

Vous pouvez utiliser les extraits de code suivants pour créer une base de données.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe CreateDatabaseCommand et CreateDatabase.

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

const params = {
    DatabaseName: "testDbFromNode"
};

const command = new CreateDatabaseCommand(params);

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

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

L'extrait de code suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }

        }
```

Décrire la base de

Vous pouvez utiliser les extraits de code suivants pour obtenir des informations sur les attributs de la base de données que vous venez de créer.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe DescribeDatabaseCommand et DescribeDatabase.

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

const params = {
    DatabaseName: "testDbFromNode"
};

const command = new DescribeDatabaseCommand(params);

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

```
}
```

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }

        }
```

Mettre à jour la base

Vous pouvez utiliser les extraits de code suivants pour mettre à jour vos bases de données.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe UpdateDatabaseCommand et UpdateDatabase.

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

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

const command = new UpdateDatabaseCommand(params);

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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}");
        }
```

Supprimer la base de données

Vous pouvez utiliser l'extrait de code suivant pour supprimer une base de données.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe DeleteDatabaseCommand et DeleteDatabase.

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

const params = {
    DatabaseName: "testDbFromNode"
};

const command = new DeleteDatabaseCommand(params);

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }
        }
```

Liste des bases de données

Vous pouvez utiliser les extraits de code suivants pour répertorier vos bases de données.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe ListDatabasesCommand et ListDatabases.

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

const params = {
    MaxResults: 15
};

const command = new ListDatabasesCommand(params);

getDatabasesList(null);

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

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

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

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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

Rubriques

- Memory Store écrit
- Magnetic Store écrit

Memory Store écrit

Vous pouvez utiliser l'extrait de code suivant pour créer une table dans laquelle les écritures magnétiques sont désactivées. Par conséquent, vous ne pouvez écrire des données que dans votre fenêtre de rétention de mémoire.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe CreateTableCommand et CreateTable.

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

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

const command = new CreateTableCommand(params);

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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 écrit

Vous pouvez utiliser l'extrait de code suivant pour créer une table dans laquelle les écritures magnétiques sont activées. Avec les écritures magnétiques, vous pouvez écrire des données à la fois dans votre fenêtre de rétention de mémoire et dans votre fenêtre de rétention de mémoire magnétique.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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());
            }

        }
```

Décrire le tableau

Vous pouvez utiliser les extraits de code suivants pour obtenir des informations sur les attributs de votre table.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe DescribeTableCommand et DescribeTable.

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

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

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

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }

        }
```

Mettre à jour une table

Vous pouvez utiliser les extraits de code suivants pour mettre à jour une table.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe UpdateTableCommand et UpdateTable.

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

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

const command = new UpdateTableCommand(params);

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }

        }
```

Supprimer une table

Vous pouvez utiliser les extraits de code suivants pour supprimer une table.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe DeleteTableCommand et DeleteTable.

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

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

const command = new DeleteTableCommand(params);

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }
        }
```

Répertorier des tables

Vous pouvez utiliser les extraits de code suivants pour répertorier les tables.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Voir également Classe ListTablesCommand et ListTables.

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

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

const command = new ListTablesCommand(params);

getTablesList(null);

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

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

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

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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}");
        }
```

Écrire des données (insertions et insertions)

Rubriques

- Écrire des lots d'enregistrements
- Écrire des lots d'enregistrements avec des attributs communs
- Upserting records
- Exemple d'attribut à mesures multiples
- Gestion des échecs d'écriture

Écrire des lots d'enregistrements

Vous pouvez utiliser les extraits de code suivants pour écrire des données dans une table Amazon Timestream. L'écriture de données par lots permet d'optimiser le coût des écritures. Pour plus d'informations, consultez Calcul du nombre d'écritures.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
    }
  }
```

Écrire des lots d'enregistrements avec des attributs communs

Si les données de vos séries chronologiques comportent des mesures et/ou des dimensions communes à de nombreux points de données, vous pouvez également utiliser la version optimisée suivante writeRecords API pour insérer des données dans Timestream pour. LiveAnalytics L'utilisation d'attributs communs avec le traitement par lots permet d'optimiser davantage le coût des écritures, comme décrit dansCalcul du nombre d'écritures.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
   }
 }
```

Upserting records

Alors que l'écriture par défaut dans Amazon Timestream suit la sémantique du premier rédacteur gagne, où les données sont stockées sous forme d'ajout uniquement et les enregistrements dupliqués sont rejetés, certaines applications nécessitent la possibilité d'écrire des données dans Amazon Timestream en utilisant la sémantique du dernier rédacteur gagnant, où l'enregistrement contenant la version la plus élevée est stocké dans le système. Certaines applications nécessitent également la possibilité de mettre à jour les enregistrements existants. Pour répondre à ces

scénarios, Amazon Timestream permet de modifier les données. Upsert est une opération qui insère un enregistrement dans le système lorsqu'il n'existe pas ou met à jour l'enregistrement lorsqu'il en existe un.

Vous pouvez modifier des enregistrements en incluant la définition d'enregistrement `Version` lors de l'envoi d'une `WriteRecords` demande. Amazon Timestream stockera l'enregistrement avec l'enregistrement le plus élevé. `Version` L'exemple de code ci-dessous montre comment vous pouvez modifier des données :

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
    }
  }
```

Exemple d'attribut à mesures multiples

Cet exemple illustre l'écriture d'attributs à plusieurs mesures. Les attributs à mesures multiples sont utiles lorsqu'un appareil ou une application que vous suivez émet plusieurs mesures ou événements au même horodatage.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
      }
    }
  }
}
```

Gestion des échecs d'écriture

Les écritures dans Amazon Timestream peuvent échouer pour une ou plusieurs des raisons suivantes :

- Il existe des enregistrements dont l'horodatage se situe en dehors de la durée de conservation de la mémoire.
- Certains enregistrements contiennent des dimensions et/ou des mesures qui dépassent les limites définies par Timestream.

- Amazon Timestream a détecté des enregistrements dupliqués. Les enregistrements sont marqués comme doublons lorsque plusieurs enregistrements ont les mêmes dimensions, horodatages et noms de mesures, mais que :
  - Les valeurs de mesure sont différentes.
  - La version n'est pas présente dans la demande ou la valeur de la version dans le nouvel enregistrement est égale ou inférieure à la valeur existante. Si Amazon Timestream rejette les données pour cette raison, `ExistingVersion` le champ du contiendra la version actuelle de `RejectedRecords` l'enregistrement telle qu'elle est stockée dans Amazon Timestream. Pour forcer une mise à jour, vous pouvez renvoyer la demande avec une version de l'enregistrement définie sur une valeur supérieure à. `ExistingVersion`

Pour plus d'informations sur les erreurs et les enregistrements rejetés, consultez la section Erreurs et RejectedRecord.

Si votre application reçoit un message `RejectedRecordsException` lorsqu'elle tente d'écrire des enregistrements dans Timestream, vous pouvez analyser les enregistrements rejetés pour en savoir plus sur les échecs d'écriture, comme indiqué ci-dessous.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
  }
```

Exécuter la requête

Rubriques

- Pagination des résultats
- Analyse des ensembles de résultats
- Accès à l'état de la requête

Pagination des résultats

Lorsque vous exécutez une requête, Timestream renvoie le jeu de résultats de manière paginée afin d'optimiser la réactivité de vos applications. L'extrait de code ci-dessous montre comment vous pouvez paginer dans le jeu de résultats. Vous devez parcourir toutes les pages du jeu de résultats jusqu'à ce que vous trouviez une valeur nulle. Les jetons de pagination expirent 3 heures après avoir été émis par Timestream pour. LiveAnalytics

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
            }
        }
```

Analyse des ensembles de résultats

Vous pouvez utiliser les extraits de code suivants pour extraire des données du jeu de résultats. Les résultats des requêtes sont accessibles jusqu'à 24 heures après la fin de la requête.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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)}]";
        }
```

Accès à l'état de la requête

Vous pouvez accéder au statut de la requête via`QueryResponse`, qui contient des informations sur la progression d'une requête, les octets analysés par une requête et les octets mesurés par une requête. Les `bytesScanned` valeurs `bytesMetered` et sont cumulatives et continuellement mises à jour lors de la pagination des résultats des requêtes. Vous pouvez utiliser ces informations pour comprendre les octets scannés par une requête individuelle et également les utiliser pour prendre certaines décisions. Par exemple, en supposant que le prix des requêtes soit de 0,01 USD par Go numérisé, vous souhaiterez peut-être annuler les requêtes supérieures à 25 USD par requête, ou X Go. L'extrait de code ci-dessous montre comment cela peut être fait.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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());
    }
}
```

Pour plus de détails sur la procédure d'annulation d'une requête, consultezAnnuler la requête.

Exécuter UNLOAD la requête

Les exemples de code suivants appellent une UNLOAD requête. Pour de plus amples informations sur UNLOAD, consultez Utilisation UNLOAD pour exporter les résultats d'une requête vers S3 depuis Timestream pour LiveAnalytics. Pour des exemples de UNLOAD requêtes, voirExemple de cas d'utilisation UNLOAD de From Timestream pour LiveAnalytics.

Rubriques

- Création et exécution d'une UNLOAD requête
- Analyse la UNLOAD réponse et obtient le nombre de lignes, le lien du manifeste et le lien des métadonnées
- Lire et analyser le contenu du manifeste
- Lire et analyser le contenu des métadonnées

Création et exécution d'une UNLOAD requête

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
    }
}
```

Analyse la UNLOAD réponse et obtient le nombre de lignes, le lien du manifeste et le lien des métadonnées

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
}
```

Lire et analyser le contenu du manifeste

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};
}
```

Lire et analyser le contenu des métadonnées

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};
}
```

Annuler la requête

Vous pouvez utiliser les extraits de code suivants pour annuler une requête.

> **Note**
>
> Ces extraits de code sont basés sur des exemples complets d'applications sur. GitHub Pour plus d'informations sur la façon de démarrer avec les exemples d'applications, consultezExemple d'application.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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);
            }
        }
```

Créer une tâche de chargement par lots

Vous pouvez utiliser les extraits de code suivants pour créer des tâches de chargement par lots.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Pour API plus de détails, voir Classe CreateBatchLoadCommand et CreateBatchLoadTask.

```
import { TimestreamWriteClient, CreateBatchLoadTaskCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-west-2", endpoint:
 "https://gamma-ingest-cell3.timestream.us-west-2.amazonaws.com" });

const params = {
    TargetDatabaseName: "BatchLoadExampleDatabase",
    TargetTableName: "BatchLoadExampleTable",
 RecordVersion: 1,
 DataModelConfiguration: {
  DataModel: {
   TimeColumn: "timestamp",
            TimeUnit: "MILLISECONDS",
            DimensionMappings: [
                {
                    SourceColumn: "registration",
                    DestinationColumn: "license"
                }
            ],
            MultiMeasureMappings: {
                TargetMultiMeasureName: "mva_measure_name",
                MultiMeasureAttributeMappings: [
                    {
                        SourceColumn: "wgt",
```

```
                        TargetMultiMeasureAttributeName: "weight",
                        MeasureValueType: "DOUBLE"
                    },
                    {
                        SourceColumn: "spd",
                        TargetMultiMeasureAttributeName: "speed",
                        MeasureValueType: "DOUBLE"
                    },
                    {
                        SourceColumn: "fuel_consumption",
                        TargetMultiMeasureAttributeName: "fuel",
                        MeasureValueType: "DOUBLE"
                    }
                ]
            }
        }
    },
 DataSourceConfiguration: {
        DataSourceS3Configuration: {
            BucketName: "test-batch-load-west-2",
            ObjectKeyPrefix: "sample.csv"
        },
        DataFormat: "CSV"
    },
    ReportConfiguration: {
        ReportS3Configuration: {
            BucketName: "test-batch-load-report-west-2",
            EncryptionOption: "SSE_S3"
        }
    }
};

const command = new CreateBatchLoadTaskCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Created batch load task ` + data.TaskId);
} catch (error) {
 console.log("Error creating table. ", error);
 throw error;
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class CreateBatchLoadTaskExample
    {
        public const string DATABASE_NAME = "<database name>";
        public const string TABLE_NAME = "<table name>";
        public const string INPUT_BUCKET = "<input bucket name>";
        public const string INPUT_OBJECT_KEY_PREFIX = "<CSV file name>";
        public const string REPORT_BUCKET = "<report bucket name>";
        public const long HT_TTL_HOURS = 24L;
        public const long CT_TTL_DAYS = 7L;
        private readonly AmazonTimestreamWriteClient writeClient;

        public CreateBatchLoadTaskExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task CreateBatchLoadTask()
        {
            try
            {
                var createBatchLoadTaskRequest = new CreateBatchLoadTaskRequest
                {
                    DataModelConfiguration = new DataModelConfiguration
                    {
                        DataModel = new DataModel
                        {
                            TimeColumn = "timestamp",
                            TimeUnit = TimeUnit.SECONDS,
                            DimensionMappings = new List<DimensionMapping>()
                            {
                                new()
                                {
                                        SourceColumn = "vehicle"
```

```
                                },
                                new()
                                {
                                        SourceColumn = "registration",
                                        DestinationColumn = "license"
                                }
                            },
                            MultiMeasureMappings = new MultiMeasureMappings
                            {
                                TargetMultiMeasureName = "mva_measure_name",
                                MultiMeasureAttributeMappings = new
List<MultiMeasureAttributeMapping>()
                                {
                                        new()
                                        {
                                                SourceColumn = "wgt",
                                                TargetMultiMeasureAttributeName =
"weight",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
                                        },
                                        new()
                                        {
                                                SourceColumn = "spd",
                                                TargetMultiMeasureAttributeName =
"speed",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
                                        },
                                        new()
                                        {
                                                SourceColumn = "fuel",
                                                TargetMultiMeasureAttributeName =
"fuel",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
                                        },
                                        new()
                                        {
                                                SourceColumn = "miles",
                                                TargetMultiMeasureAttributeName =
"miles",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
```

```
                                    }
                                }
                            }
                        }
                    },
                    DataSourceConfiguration = new DataSourceConfiguration
                    {
                        DataSourceS3Configuration = new DataSourceS3Configuration
                        {
                            BucketName = INPUT_BUCKET,
                            ObjectKeyPrefix = INPUT_OBJECT_KEY_PREFIX
                        },
                        DataFormat = "CSV"
                    },
                    ReportConfiguration = new ReportConfiguration
                    {
                        ReportS3Configuration = new ReportS3Configuration
                        {
                            BucketName = REPORT_BUCKET
                        }
                    },
                    TargetDatabaseName = DATABASE_NAME,
                    TargetTableName = TABLE_NAME
                };

                CreateBatchLoadTaskResponse response = await
 writeClient.CreateBatchLoadTaskAsync(createBatchLoadTaskRequest);
                Console.WriteLine($"Task created: " + response.TaskId);
            }
            catch (Exception e)
            {
                Console.WriteLine("Create batch load task failed:" + e.ToString());
            }
        }
    }
}
```

```
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using Amazon;
using Amazon.TimestreamQuery;
using System.Threading.Tasks;
using System;
```

```
using CommandLine;
static class Constants
{

}
namespace TimestreamDotNetSample
{
    class MainClass
    {
        public class Options
        {

        }
        public static void Main(string[] args)
        {
            Parser.Default.ParseArguments<Options>(args)
                .WithParsed<Options>(o => {
                    MainAsync().GetAwaiter().GetResult();
                });
        }

        static async Task MainAsync()
        {
            var writeClientConfig = new AmazonTimestreamWriteConfig
            {
                ServiceURL =  "<service URL>",
                Timeout = TimeSpan.FromSeconds(20),
                MaxErrorRetry = 10
            };

            var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
            var example = new CreateBatchLoadTaskExample(writeClient);
            await example.CreateBatchLoadTask();
        }
    }
}
```

Décrire la tâche de chargement par lots

Vous pouvez utiliser les extraits de code suivants pour décrire les tâches de chargement par lots.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Pour API plus de détails, voir Classe DescribeBatchLoadCommand et DescribeBatchLoadTask.

```
import { TimestreamWriteClient, DescribeBatchLoadTaskCommand } from "@aws-sdk/
client-timestream-write";
const writeClient = new TimestreamWriteClient({ region: "<region>", endpoint:
 "<endpoint>" });

const params = {
    TaskId: "<TaskId>"
};

const command = new DescribeBatchLoadTaskCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Batch load task has id ` + data.BatchLoadTaskDescription.TaskId);
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log("Batch load task doesn't exist.");
    } else {
        console.log("Describe batch load task failed.", error);
        throw error;
    }
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
```

```
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class DescribeBatchLoadTaskExample
    {
        private readonly AmazonTimestreamWriteClient writeClient;

        public DescribeBatchLoadTaskExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task DescribeBatchLoadTask(String taskId)
        {
            try
            {
                var describeBatchLoadTaskRequest = new DescribeBatchLoadTaskRequest
                {
                    TaskId = taskId
                };
                DescribeBatchLoadTaskResponse response = await
 writeClient.DescribeBatchLoadTaskAsync(describeBatchLoadTaskRequest);
                Console.WriteLine($"Task has id:
{response.BatchLoadTaskDescription.TaskId}");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Batch load task does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Describe batch load task failed:" +
 e.ToString());
            }
        }
    }
}
```

```
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using Amazon;
```

```
using Amazon.TimestreamQuery;
using System.Threading.Tasks;
using System;
using CommandLine;
static class Constants
{

}
namespace TimestreamDotNetSample
{
    class MainClass
    {
        public class Options
        {

        }
        public static void Main(string[] args)
        {
            Parser.Default.ParseArguments<Options>(args)
                .WithParsed<Options>(o => {
                    MainAsync().GetAwaiter().GetResult();
                });
        }

        static async Task MainAsync()
        {
            var writeClientConfig = new AmazonTimestreamWriteConfig
            {
                ServiceURL =  "<service URL>",
                Timeout = TimeSpan.FromSeconds(20),
                MaxErrorRetry = 10
            };

            var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
            var example = new DescribeBatchLoadTaskExample(writeClient);
            await example.DescribeBatchLoadTask("<batch load task id>");
        }
    }
}
```

Lister les tâches de chargement par lots

Vous pouvez utiliser les extraits de code suivants pour répertorier les tâches de chargement par lots.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Pour API plus de détails, voir Classe DescribeBatchLoadCommand et DescribeBatchLoadTask.

```
import { TimestreamWriteClient, ListBatchLoadTasksCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "<region>", endpoint:
 "<endpoint>" });

const params = {
    MaxResults: <15>
};

const command = new ListBatchLoadTasksCommand(params);

getBatchLoadTasksList(null);
```

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

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

        data.BatchLoadTasks.forEach(function (task) {
            console.log(task.TaskId);
        });

        if (data.NextToken) {
            return getBatchLoadTasksList(data.NextToken);
        }
    } catch (error) {
        console.log("Error while listing batch load tasks", error);
    }
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class ListBatchLoadTasksExample
    {
        private readonly AmazonTimestreamWriteClient writeClient;

        public ListBatchLoadTasksExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task ListBatchLoadTasks()
        {
            Console.WriteLine("Listing batch load tasks");
```

```
            try
            {
                var listBatchLoadTasksRequest = new ListBatchLoadTasksRequest
                {
                    MaxResults = 15
                };

                ListBatchLoadTasksResponse response = await
 writeClient.ListBatchLoadTasksAsync(listBatchLoadTasksRequest);

                PrintBatchLoadTasks(response.BatchLoadTasks);
                var nextToken = response.NextToken;

                while (nextToken != null)
                {
                    listBatchLoadTasksRequest.NextToken = nextToken;
                    response = await
 writeClient.ListBatchLoadTasksAsync(listBatchLoadTasksRequest);
                    PrintBatchLoadTasks(response.BatchLoadTasks);
                    nextToken = response.NextToken;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("List batch load tasks failed:" + e.ToString());
            }
        }

        private void PrintBatchLoadTasks(List<BatchLoadTask> tasks)
        {
            foreach (BatchLoadTask task in tasks)
                Console.WriteLine($"Task:{task.TaskId}");
        }
    }
}
```

```
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using Amazon;
using Amazon.TimestreamQuery;
using System.Threading.Tasks;
using System;
```

```
using CommandLine;
static class Constants
{

}
namespace TimestreamDotNetSample
{
    class MainClass
    {
        public class Options
        {

        }
        public static void Main(string[] args)
        {
            Parser.Default.ParseArguments<Options>(args)
                .WithParsed<Options>(o => {
                    MainAsync().GetAwaiter().GetResult();
                });
        }

        static async Task MainAsync()
        {
            var writeClientConfig = new AmazonTimestreamWriteConfig
            {
                ServiceURL =  "<service URL>",
                Timeout = TimeSpan.FromSeconds(20),
                MaxErrorRetry = 10
            };

            var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
            var example = new ListBatchLoadTasksExample(writeClient);
            await example.ListBatchLoadTasks();
        }
    }
}
```

Reprendre la tâche de chargement par lots

Vous pouvez utiliser les extraits de code suivants pour reprendre les tâches de chargement par lots.

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

L'extrait de code suivant est utilisé AWS SDK pour JavaScript la version 3. Pour plus d'informations sur l'installation du client et son utilisation, consultez Timestream Write Client - AWS SDK for JavaScript v3.

Pour API plus de détails, voir Classe CreateBatchLoadCommand et CreateBatchLoadTask.

```
import { TimestreamWriteClient, ResumeBatchLoadTaskCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "<region>", endpoint:
 "<endpoint>" });

const params = {
    TaskId: "<TaskId>"
};

const command = new ResumeBatchLoadTaskCommand(params);

try {
    const data = await writeClient.send(command);
    console.log("Resumed batch load task");
} catch (error) {
    console.log("Resume batch load task failed.", error);
 throw error;
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
```

```
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class ResumeBatchLoadTaskExample
    {
        private readonly AmazonTimestreamWriteClient writeClient;

        public ResumeBatchLoadTaskExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task ResumeBatchLoadTask(String taskId)
        {
            try
            {
                var resumeBatchLoadTaskRequest = new ResumeBatchLoadTaskRequest
                {
                    TaskId = taskId
                };
                ResumeBatchLoadTaskResponse response = await
 writeClient.ResumeBatchLoadTaskAsync(resumeBatchLoadTaskRequest);
                Console.WriteLine("Successfully resumed batch load task.");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Batch load task does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Resume batch load task failed: " + e.ToString());
            }
        }
    }
}
```

Création d'une requête planifiée

Vous pouvez utiliser les extraits de code suivants pour créer une requête planifiée avec un mappage à plusieurs mesures.

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

L'extrait de code suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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;
    }
}
```

Répertorier une requête planifiée

Vous pouvez utiliser les extraits de code suivants pour répertorier vos requêtes planifiées.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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;
    }
}
```

Décrire la requête planifiée

Vous pouvez utiliser les extraits de code suivants pour décrire une requête planifiée.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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;
    }
}
```

Exécuter une requête planifiée

Vous pouvez utiliser les extraits de code suivants pour exécuter une requête planifiée.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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;
        }
    }
```

Mettre à jour la requête planifiée

Vous pouvez utiliser les extraits de code suivants pour mettre à jour une requête planifiée.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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;
    }
}
```

Supprimer la requête planifiée

Vous pouvez utiliser les extraits de code suivants pour supprimer une requête planifiée.

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

L'extrait suivant utilise le style AWS SDK for JavaScript V2. Il est basé sur l'exemple d'application disponible sur Node.js, un exemple d'Amazon Timestream LiveAnalytics pour une application sur. 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;
    }
}
```

Utilisation du chargement par lots dans Timestream pour LiveAnalytics

Grâce au chargement par lots pour Amazon Timestream LiveAnalytics for, vous pouvez CSV ingérer des fichiers stockés dans Amazon S3 dans Timestream par lots. Grâce à cette nouvelle fonctionnalité, vous pouvez accéder à vos données dans Timestream LiveAnalytics sans avoir à utiliser d'autres outils ou à écrire du code personnalisé. Vous pouvez utiliser le chargement par lots pour compléter les données avec des temps d'attente flexibles, telles que les données qui ne sont pas immédiatement requises pour les requêtes ou les analyses.

Vous pouvez créer des tâches de chargement par lots en utilisant le AWS Management Console AWS CLI, le et le AWS SDKs. Pour plus d'informations, consultez Utilisation du chargement par lots avec la console, En utilisant le chargement par lots avec AWS CLI et En utilisant le chargement par lots avec AWS SDKs.

Outre le chargement par lots, vous pouvez écrire plusieurs enregistrements en même temps lors de l' WriteRecords APIopération. Pour obtenir des conseils sur les produits à utiliser, voirChoix entre l' WriteRecords APIopération et le chargement par lots.

Rubriques

- Concepts de chargement par lots dans Timestream
- Conditions préalables au chargement par lots
- Bonnes pratiques en matière de chargement par lots
- Préparation d'un fichier de données de chargement par lots

- Mappages de modèles de données pour le chargement par lots
- Utilisation du chargement par lots avec la console
- En utilisant le chargement par lots avec AWS CLI
- En utilisant le chargement par lots avec AWS SDKs
- Utilisation des rapports d'erreurs de chargement par lots

Concepts de chargement par lots dans Timestream

Passez en revue les concepts suivants pour mieux comprendre la fonctionnalité de chargement par lots.

Tâche de chargement par lots : tâche qui définit vos données source et votre destination dans Amazon Timestream. Vous spécifiez une configuration supplémentaire, telle que le modèle de données, lorsque vous créez la tâche de chargement par lots. Vous pouvez créer des tâches de chargement par lots via le AWS Management Console AWS CLI, et le AWS SDKs.

Destination d'importation : base de données et table de destination dans Timestream. Pour plus d'informations sur la création de bases de données et de tables, reportez-vous Créer une base de données aux sections etCréer une table .

Source de données : CSV fichier source stocké dans un compartiment S3. Pour plus d'informations sur la préparation du fichier de données, consultezPréparation d'un fichier de données de chargement par lots. Pour plus d'informations sur la tarification de S3, consultez la section Tarification d'Amazon S3.

Rapport d'erreur de chargement par lots : rapport qui stocke des informations sur les erreurs d'une tâche de chargement par lots. Vous définissez l'emplacement S3 pour les rapports d'erreur de chargement par lots dans le cadre d'une tâche de chargement par lots. Pour plus d'informations sur les informations contenues dans les rapports, consultezUtilisation des rapports d'erreurs de chargement par lots.

Mappage du modèle de données : mappage du chargement par lots pour le temps, les dimensions et les mesures depuis une source de données située dans un emplacement S3 vers un flux temporel cible pour LiveAnalytics une table. Pour de plus amples informations, veuillez consulter Mappages de modèles de données pour le chargement par lots.

Conditions préalables au chargement par lots

Voici une liste des conditions préalables à l'utilisation du chargement par lots. Pour connaître les bonnes pratiques, consultez Bonnes pratiques en matière de chargement par lots.

- Les données source de chargement par lots sont stockées dans Amazon S3 au CSV format avec en-têtes.
- Pour chaque compartiment source Amazon S3, vous devez disposer des autorisations suivantes dans une politique jointe :

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

  De même, pour chaque compartiment de sortie Amazon S3 dans lequel des rapports sont rédigés, vous devez disposer des autorisations suivantes dans une politique jointe :

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

  Par exemple :

```
{
  "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 pour les LiveAnalytics analyses CSV en mappant les informations fournies dans le modèle de données aux en-têtes. CSV Les données doivent comporter une colonne représentant l'horodatage, au moins une colonne de dimension et au moins une colonne de mesure.
- Les compartiments S3 utilisés pour le chargement par lots doivent se trouver dans la même région et provenir du même compte que le flux temporel de la LiveAnalytics table utilisé pour le chargement par lots.
- La `timestamp` colonne doit être un type de données long qui représente le temps écoulé depuis l'époque Unix. Par exemple, l'horodatage `2021-03-25T08:45:21Z` serait représenté par. `1616661921` Timestream prend en charge les secondes, les millisecondes, les microsecondes et les nanosecondes pour la précision de l'horodatage. Lorsque vous utilisez le langage de requête, vous pouvez convertir entre les formats à l'aide de fonctions telles que`to_unixtime`. Pour de plus amples informations, veuillez consulter Fonctions de date/heure.
- Timestream prend en charge le type de données de chaîne pour les valeurs de dimension. Il prend en charge les types de données longs, doubles, chaînes et booléens pour les colonnes de mesure.

Pour les limites de charge par lots et les quotas, voirChargement par lots.

Bonnes pratiques en matière de chargement par lots

Le chargement par lots fonctionne mieux (haut débit) lorsque les conditions et recommandations suivantes sont respectées :

1. CSVles fichiers soumis pour ingestion sont de petite taille, en particulier avec une taille de fichier comprise entre 100 Mo et 1 Go, afin d'améliorer le parallélisme et la vitesse d'ingestion.
2. Évitez d'ingérer simultanément des données dans la même table (par exemple en utilisant l' WriteRecords APIopération ou une requête planifiée) lorsque le chargement du lot est en cours. Cela peut entraîner des ralentissements et la tâche de chargement par lots échouera.

3. N'ajoutez, ne modifiez ou ne supprimez pas de fichiers du compartiment S3 utilisé pour le chargement par lots pendant que la tâche de chargement par lots est en cours d'exécution.
4. Ne supprimez pas ou ne révoquez pas les autorisations relatives aux tables ou à la source, et ne signalez pas les compartiments S3 contenant des tâches de chargement par lots planifiées ou en cours.
5. Lorsque vous ingérez des données avec un ensemble de valeurs de dimension à cardinalité élevée, suivez les instructions sur. Recommandations pour le partitionnement des enregistrements à mesures multiples
6. Assurez-vous de vérifier l'exactitude des données en soumettant un petit fichier. Toutes les données soumises au chargement par lots vous seront facturées, qu'elles soient correctes ou non. Pour plus d'informations sur les tarifs, consultez les tarifs d'Amazon Timestream.
7. Ne reprenez pas une tâche de chargement par lots sauf si la `ActiveMagneticStorePartitions` valeur est inférieure à 250. La tâche peut être limitée et échouer. La soumission simultanée de plusieurs tâches pour la même base de données devrait en réduire le nombre.

Les meilleures pratiques relatives à la console sont les suivantes :

1. Utilisez le générateur uniquement pour simplifier la modélisation des données qui utilise un seul nom de mesure pour les enregistrements de plusieurs mesures.
2. Pour une modélisation de données plus complexe, utilisezJSON. Par exemple, à utiliser JSON lorsque vous utilisez plusieurs noms de mesures lors de l'utilisation d'enregistrements de plusieurs mesures.

Pour un flux temporel supplémentaire consacré aux LiveAnalytics meilleures pratiques, voir. Bonnes pratiques

Préparation d'un fichier de données de chargement par lots

Un fichier de données source comporte des valeurs séparées par des délimiteurs. Le terme plus spécifique, valeurs séparées par des virgules (CSV), est utilisé de manière générique. Les séparateurs de colonnes valides incluent les virgules et les tubes. Les enregistrements sont séparés par de nouvelles lignes. Les fichiers doivent être stockés dans Amazon S3. Lorsque vous créez une nouvelle tâche de chargement par lots, l'emplacement des données source est spécifié par un ARN pour le fichier. Un fichier contient des en-têtes. Une colonne représente l'horodatage. Au moins une autre colonne représente une mesure.

Les compartiments S3 utilisés pour le chargement par lots doivent se trouver dans la même région que le flux temporel de la LiveAnalytics table utilisée pour le chargement par lots. N'ajoutez ni ne supprimez de fichiers du compartiment S3 utilisé pour le chargement par lots une fois que la tâche de chargement par lots a été soumise. Pour plus d'informations sur l'utilisation des compartiments S3, consultez Getting started with Amazon S3.

> **Note**
>
> CSVles fichiers générés par certaines applications telles qu'Excel peuvent contenir une marque d'ordre des octets (BOM) en conflit avec le codage attendu. Timestream pour les tâches de chargement LiveAnalytics par lots qui font référence à un CSV fichier avec BOM une erreur lorsqu'elles sont traitées par programmation. Pour éviter cela, vous pouvez supprimer leBOM, qui est un personnage invisible.
> Par exemple, vous pouvez enregistrer le fichier à partir d'une application telle que Notepad ++ qui vous permet de spécifier un nouveau codage. Vous pouvez également utiliser une option de programmation qui lit la première ligne, supprime le caractère de la ligne et écrit la nouvelle valeur sur la première ligne du fichier.
> Lorsque vous enregistrez à partir d'Excel, plusieurs CSV options s'offrent à vous. L'enregistrement avec une autre CSV option peut éviter le problème décrit. Mais vous devriez vérifier le résultat car une modification du codage peut affecter certains caractères.

CSVparamètres de format

Vous utilisez des caractères d'échappement lorsque vous représentez une valeur qui est par ailleurs réservée par les paramètres de format. Par exemple, si le guillemet est un guillemet double, pour représenter un guillemet double dans les données, placez le caractère d'échappement avant le guillemet double.

Pour savoir quand les spécifier lors de la création d'une tâche de chargement par lots, consultezCréation d'une tâche de chargement par lots.

Paramètre	Options
Séparateur de colonnes	(Virgule («, ») | Pipe (« | ») | Point-virgule (« ; ») | Tabulation (« /t ») | Espace vide (« "))
Personnage d'évasion	none

Paramètre	Options
Caractère de citation	Console : (Guillemet double («) | Guillemet simple ('))
Valeur nulle	Espace vide (« »)
Supprimer les espaces blancs	Console : (Non | Oui)

Mappages de modèles de données pour le chargement par lots

Ce qui suit décrit le schéma des mappages de modèles de données et donne un exemple.

Schéma de mappage des modèles de données

Syntaxe de la `CreateBatchLoadTask` demande et `BatchLoadTaskDescription` objet renvoyé par un appel pour `DescribeBatchLoadTask` inclure un `DataModelConfiguration` objet incluant le `DataModel` chargement par lots. `DataModel`définit les mappages entre les données source stockées CSV au format dans un emplacement S3 et un flux temporel cible pour la LiveAnalytics base de données et la table.

Le `TimeColumn` champ indique l'emplacement des données source pour la valeur à mapper à la `time` colonne de la table de destination dans Timestream for. LiveAnalytics `TimeUnit`Spécifie l'unité pour le`TimeColumn`, et peut être l'une des valeurs `MILLISECONDS` suivantes :`SECONDS`,`MICROSECONDS`, ou`NANOSECONDS`. Il existe également des mappages pour les dimensions et les mesures. Les mappages de dimensions sont composés de colonnes source et de champs cibles.

Pour plus d'informations, consultez DimensionMapping. Les mappages pour les mesures comportent deux options, `MixedMeasureMappings` et`MultiMeasureMappings`.

En résumé, a `DataModel` contient des mappages entre une source de données située dans un emplacement S3 et un flux temporel cible pour la LiveAnalytics table suivante.

- Heure
- Dimensions
- Mesures

Dans la mesure du possible, nous vous recommandons de mapper les données de mesure à des enregistrements de mesures multiples dans Timestream for. LiveAnalytics Pour plus d'informations sur les avantages des enregistrements à mesures multiples, consultezEnregistrements à mesures multiples.

Si plusieurs mesures des données source sont stockées sur une seule ligne, vous pouvez mapper ces mesures à des enregistrements de plusieurs mesures dans Timestream pour les utiliser. LiveAnalytics `MultiMeasureMappings` Si certaines valeurs doivent correspondre à un enregistrement à mesure unique, vous pouvez utiliser`MixedMeasureMappings`.

`MixedMeasureMappings`set `MultiMeasureMappings` les deux incluent`MultiMeasureAttributeMappings`. Les enregistrements à mesures multiples sont pris en charge, que des enregistrements à mesure unique soient nécessaires ou non.

Si seuls des enregistrements cibles à plusieurs mesures sont nécessaires dans Timestream for LiveAnalytics, vous pouvez définir des mappages de mesures dans la structure suivante.

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

> **Note**
>
> Nous vous recommandons de l'utiliser dans la `MultiMeasureMappings` mesure du possible.

Si des enregistrements cibles à mesure unique sont nécessaires dans Timestream for LiveAnalytics, vous pouvez définir des mappages de mesures dans la structure suivante.

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

```
            MultiMeasureAttributeMappings array
```

Lorsque vous l'utilisezMultiMeasureMappings, le MultiMeasureAttributeMappings tableau est toujours requis. Lorsque vous utilisez le MixedMeasureMappings tableau, si MeasureValueType c'est MULTI pour une donnéeMixedMeasureMapping, MultiMeasureAttributeMappings c'est requis pour celaMixedMeasureMapping. Dans le cas contraire, MeasureValueType indique le type de mesure pour l'enregistrement d'une seule mesure.

Quoi qu'il en soit, il existe une gamme de MultiMeasureAttributeMapping produits disponibles. Vous définissez les mappages vers les enregistrements à mesures multiples dans chacun d'eux MultiMeasureAttributeMapping comme suit :

SourceColumn

La colonne des données source qui se trouve dans Amazon S3.

TargetMultiMeasureAttributeName

Nom du nom multi-mesures cible dans la table de destination. Cette saisie est obligatoire lorsqu'elle n'MeasureNameColumnest pas fournie. Si elle MeasureNameColumn est fournie, la valeur de cette colonne est utilisée comme nom de plusieurs mesures.

MeasureValueType

L'un des DOUBLE BIGINTBOOLEAN,VARCHAR, ouTIMESTAMP.

Mappages de modèles de données avec exemple **MultiMeasureMappings**

Cet exemple illustre le mappage vers des enregistrements à mesures multiples, l'approche préférée, qui stocke chaque valeur de mesure dans une colonne dédiée. Vous pouvez télécharger un extrait CSV sur sample CSV. L'exemple contient les en-têtes suivants à mapper à une colonne cible dans un tableau Timestream for. LiveAnalytics

- time
- measure_name
- region
- location
- hostname
- memory_utilization

- cpu_utilization

Identifiez les measure_name colonnes time et du CSV fichier. Dans ce cas, elles sont directement associées au Timestream pour les colonnes de LiveAnalytics table portant le même nom.

- timedes cartes pour time
- measure_namecorrespond à measure_name (ou à la valeur que vous avez choisie)

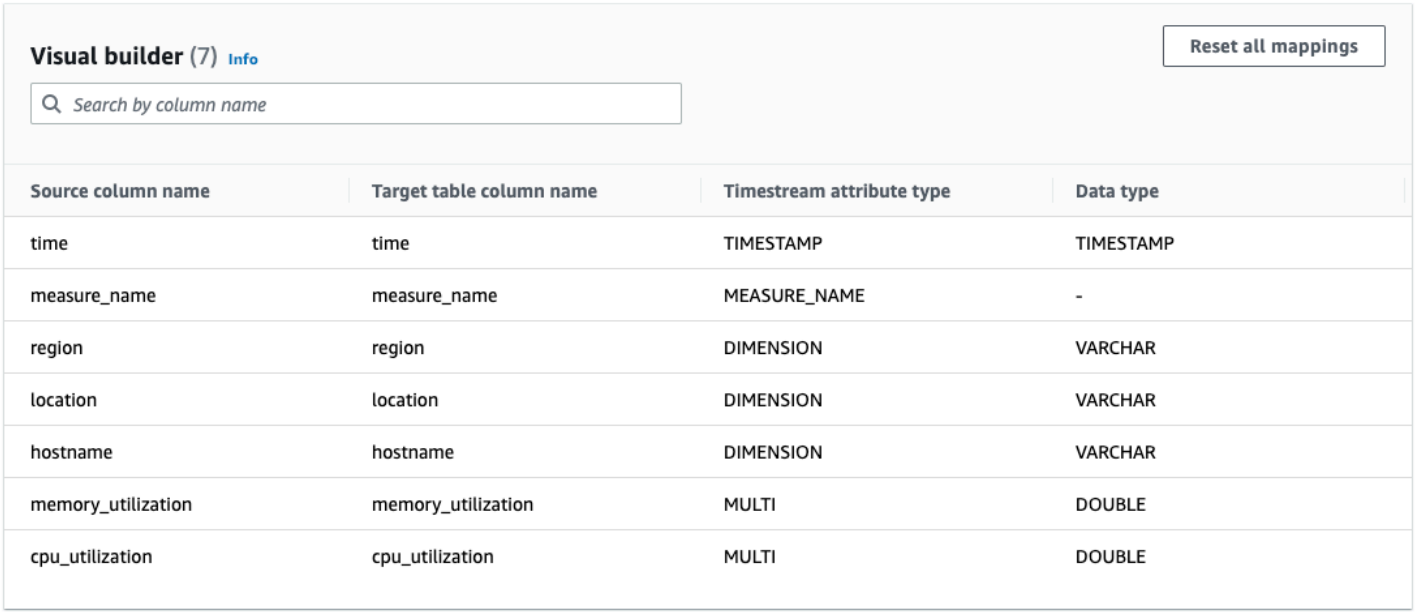
Lorsque vous utilisez leAPI, vous spécifiez time dans le TimeColumn champ une valeur d'unité de temps prise en charge, par exemple MILLISECONDS dans le TimeUnit champ. Ils correspondent au nom de la colonne Source et à l'heure d'horodatage saisis dans la console. Vous pouvez regrouper ou partitionner des enregistrements à l'aide de measure_name ce qui est défini par la MeasureNameColumn clé.

Dans l'exemple, regionlocation, et hostname sont des dimensions. Les dimensions sont cartographiées dans un tableau d'DimensionMappingobjets.

Pour les mesures, la valeur TargetMultiMeasureAttributeName deviendra une colonne dans le tableau Timestream for LiveAnalytics . Vous pouvez conserver le même nom, comme dans cet exemple. Vous pouvez également en spécifier un nouveau. MeasureValueTypeest l'un des DOUBLEBIGINT,BOOLEAN,VARCHAR, ouTIMESTAMP.

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



Mappages de modèles de données avec exemple **`MixedMeasureMappings`**

Nous vous recommandons de n'utiliser cette approche que lorsque vous devez mapper des enregistrements à mesure unique dans Timestream for. LiveAnalytics

Utilisation du chargement par lots avec la console

Voici les étapes à suivre pour utiliser le chargement par lots avec le AWS Management Console. Vous pouvez télécharger un extrait CSV sur sample CSV.

Rubriques

Accéder au chargement par lots

Suivez ces étapes pour accéder au chargement par lots à l'aide du AWS Management Console.

1. Ouvrez la console Amazon Timestream.
2. Dans le volet de navigation, choisissez Outils de gestion, puis Tâches de chargement par lots.
3. À partir de là, vous pouvez consulter la liste des tâches de chargement par lots et explorer une tâche donnée pour plus de détails. Vous pouvez également créer et reprendre des tâches.

Création d'une tâche de chargement par lots

Procédez comme suit pour créer une tâche de chargement par lots à l'aide du AWS Management Console.

1. Ouvrez la console Amazon Timestream.
2. Dans le volet de navigation, choisissez Outils de gestion, puis Tâches de chargement par lots.
3. Choisissez Créer une tâche de chargement par lots.
4. Dans Destination de l'importation, choisissez ce qui suit.

   - Base de données cible : sélectionnez le nom de la base de données créée dansCréer une base de données .
   - Table cible : sélectionnez le nom de la table créée dansCréer une table .

   Si nécessaire, vous pouvez ajouter un tableau à partir de ce panneau à l'aide du bouton Créer un nouveau tableau.
5. Dans Emplacement de la source de données S3 dans Source de données, sélectionnez le compartiment S3 dans lequel les données sources sont stockées. Utilisez le bouton Parcourir S3 pour afficher les ressources S3 auxquelles le AWS compte actif a accès ou entrez l'emplacement S3URL. La source de données doit se trouver dans la même région.

6. Dans les paramètres de format de fichier (section extensible), vous pouvez utiliser les paramètres par défaut pour analyser les données d'entrée. Vous pouvez également choisir les paramètres avancés. À partir de là, vous pouvez choisir les paramètres de CSV format et sélectionner les paramètres pour analyser les données d'entrée. Pour plus d'informations sur ces paramètres, consultezCSVparamètres de format.
7. Dans Configurer le mappage du modèle de données, configurez le modèle de données. Pour des conseils supplémentaires sur les modèles de données, voir Mappages de modèles de données pour le chargement par lots

   - Dans Cartographie du modèle de données, choisissez Mapping configuration input, puis choisissez l'une des options suivantes.
     - Générateur visuel : pour cartographier les données visuellement, choisissez TargetMultiMeasureNameou MeasureNameColumn. Ensuite, à partir de Visual Builder, mappez les colonnes.

       Visual Builder détecte et charge automatiquement les en-têtes des colonnes source à partir du fichier de source de données lorsqu'un seul CSV fichier est sélectionné comme source de données. Choisissez l'attribut et le type de données pour créer votre mappage.

       Pour plus d'informations sur l'utilisation du générateur visuel, consultezUtilisation du générateur visuel.
     - JSONéditeur : éditeur de forme libre JSON permettant de configurer votre modèle de données. Choisissez cette option si vous connaissez Timestream pour LiveAnalytics et si vous souhaitez créer des mappages de modèles de données avancés.
     - JSONfichier depuis S3 — Sélectionnez un fichier de JSON modèle que vous avez stocké dans S3. Choisissez cette option si vous avez déjà configuré un modèle de données et que vous souhaitez le réutiliser pour des chargements par lots supplémentaires.
8. Dans Emplacement S3 des journaux d'erreurs dans le rapport du journal des erreurs, sélectionnez l'emplacement S3 qui sera utilisé pour signaler les erreurs. Pour plus d'informations sur l'utilisation de ce rapport, consultezUtilisation des rapports d'erreurs de chargement par lots.
9. Pour le type de clé de chiffrement, choisissez l'une des options suivantes.

   - Clé gérée par Amazon SSE S3 (-S3) : clé de chiffrement qu'Amazon S3 crée, gère et utilise pour vous.
   - AWS KMS key (SSE-KMS) — Une clé de chiffrement protégée par AWS Key Management Service (AWS KMS).

10. Choisissez Suivant.
11. Sur la page Réviser et créer, passez en revue les paramètres et modifiez-les si nécessaire.

> **Note**
>
> Vous ne pouvez pas modifier les paramètres de la tâche de chargement par lots une fois la tâche créée. Les délais d'exécution des tâches varient en fonction de la quantité de données importées.

12. Choisissez Créer une tâche de chargement par lots.

Reprendre une tâche de chargement par lots

Lorsque vous sélectionnez une tâche de chargement par lots dont le statut est « Progression arrêtée » et qui peut toujours être reprise, vous êtes invité à reprendre la tâche. Il existe également une bannière avec un bouton Reprendre les tâches lorsque vous consultez les détails de ces tâches. Les tâches pouvant être reprises ont une date limite de reprise. Une fois cette date expirée, les tâches ne peuvent pas être reprises.

Utilisation du générateur visuel

Vous pouvez utiliser le générateur visuel pour mapper les colonnes de données source, un ou plusieurs CSV fichiers stockés dans un compartiment S3 aux colonnes de destination d'une table Timestream for LiveAnalytics .

> **Note**
>
> Votre rôle aura besoin de l'`SelectObjectContent`autorisation pour le fichier. Dans le cas contraire, vous devrez ajouter et supprimer des colonnes manuellement.

Mode de chargement automatique des colonnes sources

Timestream for LiveAnalytics peut analyser automatiquement le CSV fichier source à la recherche de noms de colonnes si vous ne spécifiez qu'un seul compartiment. Lorsqu'aucun mappage n'existe, vous pouvez choisir Importer les colonnes source.

1. Lorsque l'option Visual Builder est sélectionnée dans les paramètres d'entrée de configuration du mappage, définissez la saisie de l'heure d'horodatage. `Milliseconds`est le paramètre par défaut.
2. Cliquez sur le bouton Charger les colonnes source pour importer les en-têtes de colonne présents dans le fichier de données source. Le tableau sera renseigné avec les noms des en-têtes des colonnes source provenant du fichier de source de données.
3. Choisissez le nom de colonne de la table cible, le type d'attribut Timestream et le type de données pour chaque colonne source.

   Pour plus de détails sur ces colonnes et les valeurs possibles, consultezMappage des champs.
4. Utilisez drag-to-fill cette fonctionnalité pour définir la valeur de plusieurs colonnes à la fois.

Ajouter manuellement des colonnes source

Si vous utilisez un bucket ou un CSV préfixe et non un seulCSV, vous pouvez ajouter et supprimer des mappages de colonnes dans l'éditeur visuel à l'aide des boutons Ajouter un mappage de colonnes et Supprimer le mappage de colonnes. Il existe également un bouton pour réinitialiser les mappages.

Mappage des champs

- Nom de la colonne source : nom d'une colonne du fichier source qui représente une mesure à importer. Timestream for LiveAnalytics peut renseigner cette valeur automatiquement lorsque vous utilisez l'option Importer des colonnes source.
- Nom de colonne de la table cible : entrée facultative indiquant le nom de colonne de la mesure dans la table cible.
- Type d'attribut Timestream : type d'attribut des données de la colonne source spécifiée, par exemple. `DIMENSION`
  - TIMESTAMP— Spécifie le moment où une mesure a été collectée.
  - MULTI— Plusieurs mesures sont représentées.
  - DIMENSION— Métadonnées des séries chronologiques.
  - MEASURE_ NAME — Pour les enregistrements d'une seule mesure, il s'agit du nom de la mesure.
- Type de données : type de colonne Timestream, par exemple. `BOOLEAN`
  - BIGINT— Un entier de 64 bits.

- BOOLEAN— Les deux valeurs de vérité de la logique : vrai et faux.
- DOUBLE— Numéro à précision variable de 64 bits.
- TIMESTAMP— Une instance temporelle qui utilise une précision de l'ordre de la nanoseconde et qui suit le temps écoulé depuis l'époque d'Unix. UTC

En utilisant le chargement par lots avec AWS CLI

Configuration

Pour commencer à utiliser le chargement par lots, suivez les étapes suivantes.

1. Installez le AWS CLI en suivant les instructions fournies à l'adresseAccès à Amazon Timestream LiveAnalytics pour utiliser le AWS CLI.
2. Exécutez la commande suivante pour vérifier que les CLI commandes Timestream ont été mises à jour. Vérifiez que cela create-batch-load-task figure dans la liste.

   ```
   aws timestream-write help
   ```
3. Préparez une source de données en suivant les instructions dePréparation d'un fichier de données de chargement par lots.
4. Créez une base de données et une table en suivant les instructions deAccès à Amazon Timestream LiveAnalytics pour utiliser le AWS CLI.
5. Créez un compartiment S3 pour la sortie du rapport. Le compartiment doit se trouver dans la même région. Pour plus d'informations sur les compartiments, consultez Création, configuration et utilisation des compartiments Amazon S3.
6. Créez une tâche de chargement par lots. Pour les étapes, consultez Création d'une tâche de chargement par lots.
7. Confirmez le statut de la tâche. Pour les étapes, consultez Décrire la tâche de chargement par lots.

Création d'une tâche de chargement par lots

Vous pouvez créer une tâche de chargement par lots à l'aide de la `create-batch-load-task` commande. Lorsque vous créez une tâche de chargement par lots à l'aide duCLI, vous pouvez utiliser un JSON paramètre qui vous permet d'agréger les paramètres en un seul JSON fragment. `cli-input-json` Vous pouvez également séparer ces détails à l'aide de plusieurs autres

paramètres`data-model-configuration`, notamment`data-source-configuration`,`report-configuration`,`target-database-name`, et`target-table-name`.

Pour obtenir un exemple, veuillez consulter Exemple de création d'une tâche de chargement par lots.

Décrire la tâche de chargement par lots

Vous pouvez récupérer la description d'une tâche de chargement par lots comme suit.

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

Voici un exemple de réponse.

```
{
    "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
    }
```

```
}
```

Lister les tâches de chargement par lots

Vous pouvez répertorier les tâches de chargement par lots comme suit.

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

Une sortie apparaît comme suit.

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

Reprendre la tâche de chargement par lots

Vous pouvez reprendre une tâche de chargement par lots comme suit.

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

Une réponse peut indiquer un succès ou contenir des informations d'erreur.

Exemple de création d'une tâche de chargement par lots

Example

1. Créez un flux temporel pour la LiveAnalytics base de données nommée `BatchLoad` et une table nommée. `BatchLoadTest` Vérifiez et, si nécessaire, ajustez les valeurs pour `MemoryStoreRetentionPeriodInHours` et`MagneticStoreRetentionPeriodInDays`.

   ```
   aws timestream-write create-database --database-name BatchLoad \
   ```

```
aws timestream-write create-table --database-name BatchLoad \
--table-name BatchLoadTest \
--retention-properties "{\"MemoryStoreRetentionPeriodInHours\": 12,
 \"MagneticStoreRetentionPeriodInDays\": 100}"
```

2. À l'aide de la console, créez un compartiment S3 et copiez le `sample.csv` fichier à cet emplacement. Vous pouvez télécharger un extrait CSV sur sample CSV.
3. À l'aide de la console, créez un compartiment S3 pour que Timestream LiveAnalytics rédige un rapport si la tâche de chargement par lots aboutit à des erreurs.
4. Créez une tâche de chargement par lots. Assurez-vous de remplacer *$INPUT_BUCKET* and *$REPORT_BUCKET* avec les buckets que vous avez créés au cours des étapes précédentes.

```
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
```

La commande précédente renvoie le résultat suivant.

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

5. Vérifiez l'état d'avancement de la tâche. Assurez-vous de remplacer *$TASK_ID* avec l'identifiant de tâche renvoyé à l'étape précédente.

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

Exemple de sortie

```
{
```

```
    "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
    }
}
```

En utilisant le chargement par lots avec AWS SDKs

Pour des exemples de création, de description et de liste de tâches de chargement par lots à l'aide des sections AWS SDKsCréer une tâche de chargement par lots, voirDécrire la tâche de chargement par lots,Lister les tâches de chargement par lots, etReprendre la tâche de chargement par lots.

Utilisation des rapports d'erreurs de chargement par lots

Les tâches de chargement par lots ont l'une des valeurs d'état suivantes :

- `CREATED`(Créé) — La tâche est créée.

- `IN_PROGRESS`(En cours) — La tâche est en cours.
- `FAILED`(Échec) — La tâche est terminée. Mais une ou plusieurs erreurs ont été détectées.
- `SUCCEEDED`(Terminé) — La tâche s'est terminée sans erreur.
- `PROGRESS_STOPPED`(Progression arrêtée) — La tâche s'est arrêtée mais n'est pas terminée. Vous pouvez essayer de reprendre la tâche.
- `PENDING_RESUME`(En attente de reprise) — La tâche est en attente de reprise.

En cas d'erreur, un rapport de journal des erreurs est créé dans le compartiment S3 défini à cet effet. Les erreurs sont classées sous forme taskErrors ou fileErrors dans des tableaux distincts. Voici un exemple de rapport d'erreur.

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

Utilisation de requêtes planifiées dans Timestream pour LiveAnalytics

La fonctionnalité de requêtes planifiées d'Amazon Timestream LiveAnalytics for est une solution entièrement gérée, évolutive et sans serveur permettant de calculer et de stocker des agrégats, des

cumuls et d'autres formes de données prétraitées généralement utilisées pour les tableaux de bord opérationnels, les rapports commerciaux, les analyses ad hoc et d'autres applications. Les requêtes planifiées rendent les analyses en temps réel plus performantes et plus rentables, ce qui vous permet de tirer des informations supplémentaires de vos données et de continuer à prendre de meilleures décisions commerciales.

Avec les requêtes planifiées, vous définissez les requêtes d'analyse en temps réel qui calculent les agrégats, les cumuls et les autres opérations sur les données. Amazon Timestream exécute régulièrement et automatiquement ces requêtes et écrit de manière fiable les résultats des requêtes dans une table séparée. LiveAnalytics Les données sont généralement calculées et mises à jour dans ces tables en quelques minutes.

Vous pouvez ensuite diriger vos tableaux de bord et vos rapports vers les tables contenant des données agrégées au lieu d'interroger les tables sources considérablement plus grandes. Cela se traduit par des gains de performance et de coûts qui peuvent dépasser des ordres de grandeur. Cela est dû au fait que les tables contenant des données agrégées contiennent beaucoup moins de données que les tables sources, ce qui leur permet d'effectuer des requêtes plus rapidement et de stocker les données à moindre coût.

En outre, les tables comportant des requêtes planifiées offrent toutes les fonctionnalités existantes d'un Timestream pour LiveAnalytics une table. Par exemple, vous pouvez interroger les tables à l'aide deSQL. Vous pouvez visualiser les données stockées dans les tables à l'aide de Grafana. Vous pouvez également ingérer des données dans le tableau à l'aide d'Amazon Kinesis, AmazonMSK, AWS IoT Core et Telegraf. Vous pouvez configurer des politiques de conservation des données sur ces tables pour une gestion automatique du cycle de vie des données.

La conservation des données des tables contenant des données agrégées étant totalement découplée de celle des tables sources, vous pouvez également choisir de réduire la rétention des données des tables sources et de conserver les données agrégées pendant une durée beaucoup plus longue, à une fraction du coût de stockage des données. Les requêtes planifiées rendent les analyses en temps réel plus rapides, moins coûteuses et donc plus accessibles à un plus grand nombre de clients, afin qu'ils puissent surveiller leurs applications et prendre de meilleures décisions commerciales basées sur les données.

Rubriques

- Exemple : utilisation d'analyses en temps réel pour détecter les paiements frauduleux et prendre de meilleures décisions commerciales
- Concepts de requêtes planifiées
- Expressions de planification pour les requêtes planifiées
- Mappages de modèles de données pour les requêtes planifiées
- Messages de notification de requêtes planifiées
- Rapports d'erreur liés aux requêtes planifiées
- Modèles de requêtes planifiées et exemples

Avantages des requêtes planifiées

Les avantages des requêtes planifiées sont les suivants :

- Facilité opérationnelle — Les requêtes planifiées sont sans serveur et entièrement gérées.
- Performances et coûts : étant donné que les requêtes planifiées précalculent les agrégats, les cumuls ou les autres opérations d'analyse en temps réel de vos données et stockent les résultats dans une table, les requêtes qui accèdent aux tables remplies par des requêtes planifiées contiennent moins de données que les tables sources. Par conséquent, les requêtes exécutées sur ces tables sont plus rapides et moins coûteuses. Les tables alimentées par des calculs planifiés contiennent moins de données que leurs tables sources, ce qui contribue à réduire les coûts de stockage. Vous pouvez également conserver ces données plus longtemps dans la mémoire à une fraction du coût de conservation des données sources dans la mémoire.
- Interopérabilité — Les tables remplies par des requêtes planifiées offrent toutes les fonctionnalités existantes de Timestream pour les LiveAnalytics tables et peuvent être utilisées avec tous les services et outils compatibles avec Timestream for. LiveAnalytics Voir Travailler avec d'autres services pour plus de détails.

Cas d'utilisation des requêtes planifiées

Vous pouvez utiliser des requêtes planifiées pour des rapports commerciaux qui résument l'activité de l'utilisateur final à partir de vos applications, afin de former des modèles d'apprentissage automatique à des fins de personnalisation. Vous pouvez également utiliser des requêtes planifiées pour les alarmes qui détectent des anomalies, des intrusions sur le réseau ou des activités frauduleuses, afin de prendre des mesures correctives immédiates.

En outre, vous pouvez utiliser des requêtes planifiées pour une gouvernance des données plus efficace. Vous pouvez le faire en accordant l'accès aux tables source exclusivement aux requêtes planifiées et en fournissant à vos développeurs l'accès uniquement aux tables remplies par des requêtes planifiées. Cela permet de minimiser l'impact des requêtes involontaires de longue durée.

Exemple : utilisation d'analyses en temps réel pour détecter les paiements frauduleux et prendre de meilleures décisions commerciales

Imaginons un système de paiement qui traite les transactions envoyées à partir de plusieurs point-of-sale terminaux répartis dans les grandes villes métropolitaines des États-Unis. Vous souhaitez utiliser Amazon Timestream LiveAnalytics pour stocker et analyser les données des transactions, afin de détecter les transactions frauduleuses et d'exécuter des requêtes d'analyse en temps réel. Ces requêtes peuvent vous aider à répondre à des questions commerciales telles que l'identification des point-of-sale terminaux les plus fréquentés et les moins utilisés par heure, l'heure la plus chargée de la journée dans chaque ville et la ville enregistrant le plus grand nombre de transactions par heure.

Le système traite environ 100 000 transactions par minute. Chaque transaction stockée dans Amazon Timestream LiveAnalytics est de 100 octets. Vous avez configuré 10 requêtes exécutées toutes les minutes pour détecter différents types de paiements frauduleux. Vous avez également créé 25 requêtes qui regroupent et découpent vos données selon différentes dimensions afin de répondre à vos questions commerciales. Chacune de ces requêtes traite les données de la dernière heure.

Vous avez créé un tableau de bord pour afficher les données générées par ces requêtes. Le tableau de bord contient 25 widgets, il est actualisé toutes les heures et 10 utilisateurs y accèdent généralement à tout moment. Enfin, votre mémoire est configurée avec une période de conservation des données de 2 heures et la mémoire magnétique est configurée pour une période de conservation des données de 6 mois.

Dans ce cas, vous pouvez utiliser des requêtes d'analyse en temps réel qui recalculent les données chaque fois que le tableau de bord est consulté et actualisé, ou utiliser des tables dérivées pour le tableau de bord. Le coût des requêtes pour les tableaux de bord basés sur des requêtes d'analyse en temps réel sera de 120,70$ par mois. En revanche, le coût des requêtes de tableau de bord basées sur des tables dérivées sera de 12,27 dollars par mois (consultez Amazon Timestream pour les tarifs). LiveAnalytics Dans ce cas, l'utilisation de tables dérivées réduit le coût des requêtes d'environ 10 fois.

Concepts de requêtes planifiées

Chaîne de requête : il s'agit de la requête dont vous précalculez le résultat et que vous stockez dans un autre Timestream pour une table. LiveAnalytics Vous pouvez définir une requête planifiée en utilisant toute la SQL surface de Timestream for LiveAnalytics, ce qui vous permet d'écrire des requêtes avec des expressions tabulaires communes, des requêtes imbriquées, des fonctions de fenêtre ou tout autre type de fonction agrégée et scalaire prise en charge par Timestream pour le langage de requête. LiveAnalytics

Expression de planification : vous permet de spécifier le moment où vos instances de requête planifiées sont exécutées. Vous pouvez spécifier les expressions à l'aide d'une expression cron (telle qu'une exécution à 8 heures du matin UTC tous les jours) ou d'une expression de débit (telle qu'une exécution toutes les 10 minutes).

Configuration cible : vous permet de spécifier la manière dont vous mappez le résultat d'une requête planifiée dans la table de destination dans laquelle les résultats de cette requête planifiée seront stockés.

Configuration des notifications -Timestream pour exécuter LiveAnalytics automatiquement les instances d'une requête planifiée en fonction de votre expression de planification. Vous recevez une notification pour chaque requête exécutée sur un SNS sujet que vous configurez lorsque vous créez une requête planifiée. Cette notification indique si l'instance a été exécutée avec succès ou si elle a rencontré des erreurs. En outre, il fournit des informations telles que les octets mesurés, les données écrites dans la table cible, l'heure du prochain appel, etc.

Voici un exemple de ce type de message de notification.

```
{
    "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
        }
    }
}
```

Ce message de notification `bytesMetered` contient les octets analysés par la requête sur la table source et dataWrites les octets écrits dans la table cible.

> **Note**
>
> Si vous utilisez ces notifications par programmation, sachez que de nouveaux champs pourraient être ajoutés au message de notification à l'avenir.

Emplacement du rapport d'erreur : les requêtes planifiées s'exécutent de manière asynchrone et stockent les données dans la table cible. Si une instance rencontre des erreurs (par exemple, des données non valides qui n'ont pas pu être stockées), les enregistrements contenant des erreurs sont consignés dans un rapport d'erreurs à l'emplacement du rapport d'erreurs que vous spécifiez lors de la création d'une requête planifiée. Vous spécifiez le compartiment S3 et le préfixe de l'emplacement. Timestream for LiveAnalytics ajoute le nom de la requête planifiée et l'heure d'invocation à ce préfixe pour vous aider à identifier les erreurs associées à une instance spécifique d'une requête planifiée.

Balisage : vous pouvez éventuellement spécifier des balises que vous pouvez associer à une requête planifiée. Pour plus de détails, voir Tagging Timestream for Resources. LiveAnalytics

Exemple

Dans l'exemple suivant, vous calculez un agrégat simple à l'aide d'une requête planifiée :

```
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`- Dans cet exemple, vous remarquerez que la requête accepte un paramètre nommé spécial`@scheduled_runtime`. Il s'agit d'un paramètre spécial (de type

Timestamp) que le service définit lors de l'appel d'une instance spécifique d'une requête planifiée afin que vous puissiez contrôler de manière déterministe la plage de temps pour laquelle une instance spécifique d'une requête planifiée analyse les données de la table source. Vous pouvez l'utiliser `@scheduled_runtime` dans votre requête à n'importe quel endroit où un type d'horodatage est attendu.

Prenons un exemple où vous définissez une expression de planification : cron (0/5 * * * ? *) où la requête planifiée sera exécutée à la minute 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 de chaque heure. Pour l'instance déclenchée le 2021-12-01 00:05:00, le paramètre @scheduled_runtime est initialisé à cette valeur, de telle sorte que l'instance fonctionne actuellement sur des données comprises entre le 2021-11-30 23:55:00 et le 2021-12-01 00:06:00.

Instances dont les plages temporelles se chevauchent : comme vous le verrez dans cet exemple, deux instances suivantes d'une requête planifiée peuvent se chevaucher dans leurs plages temporelles. C'est quelque chose que vous pouvez contrôler en fonction de vos besoins, des prédicats temporels que vous spécifiez et de l'expression de planification. Dans ce cas, ce chevauchement permet à ces calculs de mettre à jour les agrégats en fonction des données dont l'arrivée a été légèrement retardée, jusqu'à 10 minutes dans cet exemple. L'exécution de la requête déclenchée le 2021-12-01 00:00:00 couvrira la plage de temps 2021-11-30 23:50:00 à 2021-12-30 00:01:00 et l'exécution de la requête déclenchée le 2021-12-01 00:05:00 couvrira la plage 2021-11-30 23:55:00 à 2021-12-01 00:06:00.

Pour garantir l'exactitude et s'assurer que les agrégats stockés dans la table cible correspondent aux agrégats calculés à partir de la table source, Timestream for LiveAnalytics garantit que le calcul du 2021-12-01 00:05:00 ne sera effectué qu'une fois le calcul du 2021-12-01 00:00:00 terminé. Les résultats de ces derniers calculs peuvent mettre à jour tout agrégat déjà matérialisé si une nouvelle valeur est générée. En interne, Timestream for LiveAnalytics utilise des versions d'enregistrement où les enregistrements générés par les dernières instances d'une requête planifiée se verront attribuer un numéro de version supérieur. Par conséquent, les agrégats calculés par l'appel au 2021-12-01 00:05:00 peuvent mettre à jour les agrégats calculés par l'appel au 2021-12-01 00:00:00, en supposant que des données plus récentes soient disponibles dans la table source.

Déclencheurs automatiques ou déclencheurs manuels - Une fois qu'une requête planifiée est créée, Timestream for exécute LiveAnalytics automatiquement les instances en fonction du calendrier spécifié. Ces déclencheurs automatisés sont entièrement gérés par le service.

Toutefois, il peut arriver que vous souhaitiez lancer manuellement certaines instances d'une requête planifiée. Par exemple, si une instance spécifique a échoué lors d'une exécution de requête, si des données sont arrivées en retard ou des mises à jour sont apparues dans la table source après

l'exécution de la planification automatique, ou si vous souhaitez mettre à jour la table cible pour les plages de temps non couvertes par les exécutions de requêtes automatisées (par exemple, pour les plages de temps avant la création d'une requête planifiée).

Vous pouvez utiliser le ExecuteScheduledQuery API pour lancer manuellement une instance spécifique d'une requête planifiée en transmettant le InvocationTime paramètre, qui est une valeur utilisée pour le paramètre @scheduled_runtime. Voici quelques points importants à prendre en compte lors de l'utilisation du ExecuteScheduledQuery API :

- Si vous déclenchez plusieurs de ces invocations, vous devez vous assurer qu'elles ne génèrent pas de chevauchement de plages temporelles. Si vous ne pouvez pas garantir que les plages de temps ne se chevauchent pas, assurez-vous que ces exécutions de requêtes sont lancées séquentiellement les unes après les autres. Si vous lancez simultanément plusieurs exécutions de requêtes dont les plages temporelles se chevauchent, vous pouvez constater des échecs déclencheurs susceptibles de provoquer des conflits de version dans les rapports d'erreur relatifs à ces exécutions de requêtes.
- Vous pouvez lancer les invocations avec n'importe quelle valeur d'horodatage pour @scheduled_runtime. Il est donc de votre responsabilité de définir les valeurs de manière appropriée afin que les plages de temps appropriées soient mises à jour dans la table cible en fonction des plages où les données ont été mises à jour dans la table source.

Expressions de planification pour les requêtes planifiées

Vous pouvez créer des requêtes planifiées selon un calendrier automatique en utilisant Amazon Timestream LiveAnalytics pour les requêtes planifiées qui utilisent des expressions cron ou rate. Toutes les requêtes planifiées utilisent le UTC fuseau horaire, et la précision minimale possible pour les plannings est de 1 minute.

Les expressions de planification peuvent être spécifiées de deux manières : cron et rate. Les expressions Cron offrent un contrôle de planification plus précis, tandis que les expressions de taux sont plus simples à exprimer mais ne disposent pas d'un contrôle précis.

Par exemple, avec une expression cron, vous pouvez définir une requête planifiée qui est déclenchée à une heure précise un certain jour de la semaine ou du mois, ou une minute spécifiée toutes les heures uniquement du lundi au vendredi, etc. En revanche, les expressions de débit lancent une requête planifiée à un rythme régulier, par exemple une fois par minute, heure ou jour, à partir de l'heure exacte à laquelle la requête planifiée est créée.

Expression cron

- Syntaxe

```
cron(fields)
```

Ces expressions se composent de six champs obligatoires qui sont séparés par des espaces.

Champ	Valeurs	Caractères génériques
Minutes	0-59	, - * /
Heures	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Mois	1-12 ou JAN - DEC	, - * /
D ay-of-week	1-7 ou SUN - SAT	, - * ? L #
Année	1970-2199	, - * /

Caractères génériques

- Le caractère générique *, * (virgule) inclut des valeurs supplémentaires. Dans le champ Mois, JANFEB, MAR inclurait les mois de janvier, février et mars.
- Le caractère générique *-* (tiret) indique les plages. Dans le champ Jour, 1-15 englobe les jours 1 à 15 du mois spécifié.
- Le caractère générique*** (astérisque) inclut toutes les valeurs du champ. Dans le champ Heures, *** inclurait toutes les heures. Vous ne pouvez pas utiliser *** à la fois dans Day-of-week les champs Day-of-month et. Si vous l'utilisez dans l'un d'entre eux, vous devez utiliser* ? * dans l'autre.
- Le caractère générique */* (barre oblique) indique les incréments. Dans le champ Minutes, vous pouvez saisir 1/10 pour spécifier toutes les 10 minutes, à partir de la première minute de l'heure (par exemple, les 11, 21 et 31 minutes, etc.).

- Le * ? Le caractère générique * (point d'interrogation) indique l'un ou l'autre. Dans le Day-of-month champ, vous pouvez saisir *7* et si vous ne vous souciez pas du jour de la semaine le 7, vous pouvez saisir* ? * sur le Day-of-week terrain.
- Le caractère générique *L* dans les Day-of-week champs Day-of-month ou indique le dernier jour du mois ou de la semaine.
- Le caractère générique W dans le Day-of-month champ indique un jour de semaine. Dans le Day-of-month champ, 3W indique le jour de la semaine le plus proche du troisième jour du mois.
- Le caractère générique *#* dans le Day-of-week champ indique une certaine occurrence du jour de la semaine spécifié dans un délai d'un mois. Par exemple, 3#2 correspond au deuxième mardi du mois : le 3 fait référence à mardi, car c'est le troisième jour de chaque semaine, et le 2 fait référence à la deuxième journée de ce type dans le mois.

> **Note**
>
> Si vous utilisez un caractère « # », vous ne pouvez définir qu'une seule expression dans le day-of-week champ. Par exemple, « 3#1,6#3 » n'est pas valide, car il est interprété comme deux expressions.

Limites

- Vous ne pouvez pas spécifier les Day-of-week champs Day-of-month et dans la même expression cron. Si vous spécifiez une valeur (ou un *) dans l'un des champs, vous devez utiliser un * ? * (point d'interrogation) dans l'autre.
- Les expressions cron qui entraînent des fréquences d'une rapidité supérieure à 1 minute ne sont pas prises en charge.

Exemples

Minutes	Heures	Jour du mois	Mois	Jour de la semaine	Année	Signification
0 USD	10	*	*	?	*	Courez à 10 h (UTC) tous les jours.

Minutes	Heures	Jour du mois	Mois	Jour de la semaine	Année	Signification
15	12	*	*	?	*	Courez à 12 h 15 (UTC) tous les jours.
0	18	?	*	MON-FRI	*	Ouvert à 18 h 00 (UTC) du lundi au vendredi.
0	8	1	*	?	*	Courez à 8 h 00 (UTC) tous les premiers jours du mois.
0/15	*	*	*	?	*	Courez toutes les 15 minutes.
0/10	*	*	*	MON-FRI	*	Ouvert toutes les 10 minutes du lundi au vendredi.

Minutes	Heures	Jour du mois	Mois	Jour de la semaine	Année	Signification
0/5	8-17	?	*	MON-FRI	*	Ouvert toutes les 5 minutes du lundi au vendredi entre 8 h 00 et 17 h 55 ()UTC.

Expressions de fréquence

- Une expression de fréquence démarre au moment où vous créez la règle d'événement planifié, puis s'exécute selon le calendrier défini. Les expressions de fréquence comportent deux champs obligatoires. Ces champs sont séparés par un espace.

  Syntaxe

  ```
  rate(value unit)
  ```

  - `value`: Un chiffre positif.
  - `unit`: unité de temps. Différentes unités sont requises pour les valeurs de 1 (par exemple, minute) et les valeurs supérieures à 1 (par exemple, minutes). Valeurs valides : minute | minutes | heure | heures | jour | jours

Mappages de modèles de données pour les requêtes planifiées

Timestream for LiveAnalytics prend en charge la modélisation flexible des données dans ses tables et cette même flexibilité s'applique aux résultats des requêtes planifiées qui sont matérialisés dans un autre Timestream for table. LiveAnalytics Avec les requêtes planifiées, vous pouvez interroger n'importe quelle table, qu'elle contienne des données dans des enregistrements à mesures multiples ou des enregistrements à mesure unique, et écrire les résultats de la requête à l'aide d'enregistrements à mesures multiples ou à mesure unique.

Vous utilisez le TargetConfiguration dans la spécification d'une requête planifiée pour mapper les résultats de la requête aux colonnes appropriées de la table dérivée de destination. Les sections suivantes décrivent les différentes manières de le spécifier TargetConfiguration pour obtenir différents modèles de données dans la table dérivée. Plus précisément, vous verrez :

- Comment écrire dans des enregistrements à mesures multiples lorsque le résultat de la requête ne porte pas de nom de mesure et que vous spécifiez le nom de la mesure cible dans le TargetConfiguration.
- Comment utiliser le nom de la mesure dans le résultat de la requête pour écrire des enregistrements de plusieurs mesures.
- Comment définir un modèle pour écrire plusieurs enregistrements avec différents attributs multi-mesures.
- Comment définir un modèle à écrire dans des enregistrements à mesure unique de la table dérivée.
- Comment interroger des enregistrements à mesure unique et/ou à mesures multiples dans une requête planifiée et faire en sorte que les résultats soient matérialisés sous forme d'enregistrement à mesure unique ou à mesure multiple, ce qui vous permet de choisir la flexibilité des modèles de données.

Exemple : nom de mesure cible pour les enregistrements de mesures multiples

Dans cet exemple, vous verrez que la requête lit les données d'une table contenant des données à mesures multiples et écrit les résultats dans une autre table à l'aide d'enregistrements à mesures multiples. Le résultat de la requête planifiée ne comporte pas de colonne de nom de mesure naturelle. Vous spécifiez ici le nom de la mesure dans le tableau dérivé à l'aide de la TargetMultiMeasureName propriété du 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"
        }
    }
}
```

Dans cet exemple, le mappage crée un enregistrement multi-mesures avec le nom de la mesure, les métriques du tableau de bord et les noms d'attributs, 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 Notez l'utilisation facultative de TargetMultiMeasureAttributeName pour renommer les colonnes de sortie de la requête en un autre nom d'attribut utilisé pour la matérialisation des résultats.

Voici le schéma de la table de destination une fois cette requête planifiée matérialisée. Comme vous pouvez le voir dans le flux temporel du type d' LiveAnalytics attribut dans le résultat suivant, les résultats sont matérialisés dans un enregistrement à plusieurs mesures portant un nom de mesure unique`dashboard-metrics`, comme indiqué dans le schéma de mesure.

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
region	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
CpuSystemP100	double	MULTI
avgMemCached	double	MULTI
min_cpu_idle_1h	double	MULTI
avg_disk_free_1h	double	MULTI
avg_disk_used_1h	double	MULTI
totalDiskWrites	double	MULTI
max_mem_used_1h	double	MULTI
min_mem_free_1h	double	MULTI
CpuUserP100	double	MULTI

Les mesures correspondantes obtenues à l'aide d'une SHOW MEASURES requête sont les suivantes.

nom_mesure	data_type	Dimensions
métriques du tableau de bord	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

Exemple : utilisation du nom de mesure issu d'une requête planifiée dans des enregistrements multi-mesures

Dans cet exemple, vous verrez une requête lire à partir d'une table contenant des enregistrements à mesure unique et matérialiser les résultats sous forme d'enregistrements à mesures multiples. Dans ce cas, le résultat de la requête planifiée comporte une colonne dont les valeurs peuvent être utilisées comme noms de mesures dans la table cible où les résultats de la requête planifiée sont matérialisés. Vous pouvez ensuite spécifier le nom de mesure pour l'enregistrement multi-mesures dans la table dérivée à l'aide de la MeasureNameColumn propriété in 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"
        }
    }
}
```

Dans cet exemple, le mappage créera des enregistrements à mesures multiples avec les attributs avg_1h, p0_1h, sum_1h, p100_1h et utilisera les valeurs de la colonne measure_name dans le résultat de la requête comme nom de mesure pour les enregistrements à mesures multiples de la table de destination. Notez également que les exemples précédents utilisent éventuellement le TargetMultiMeasureAttributeName avec un sous-ensemble de mappages pour renommer les attributs. Par exemple, min_1h a été renommé en p0_1h et max_1h est renommé en p100_1h.

Voici le schéma de la table de destination une fois cette requête planifiée matérialisée. Comme vous pouvez le voir dans le flux temporel du type d' LiveAnalytics attribut dans le résultat suivant, les résultats sont matérialisés dans un enregistrement à plusieurs mesures. Si vous examinez le schéma de mesure, neuf noms de mesures différents ont été ingérés, correspondant aux valeurs affichées dans les résultats de la requête.

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
region	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
sum_1h	double	MULTI
p100_1h	double	MULTI
p0_1h	double	MULTI
moyenne 1 h	double	MULTI

Les mesures correspondantes obtenues à l'aide d'une SHOW MEASURES requête sont les suivantes.

nom_mesure	data_type	Dimensions
cpu_idle	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
système_processeur	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
utilisateur_processeur	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
disk_free	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
disk_io_writes	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
disk_used	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
mémoire_mise en cache	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
sans mémoire	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
sans mémoire	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

Exemple : mappage des résultats sur différents enregistrements à mesures multiples avec différents attributs

L'exemple suivant montre comment vous pouvez mapper différentes colonnes du résultat de votre requête dans différents enregistrements à mesures multiples portant différents noms de mesures. Si vous voyez la définition de requête planifiée suivante, le résultat de la requête contient les colonnes suivantes : region, heure, avg_mem_cached_1h, min_mem_free_1h, max_mem_used_1h, total_disk_io_writes_1h, avg_disk_free_1h, max_cpu_user_1h, max_cpu_user_1h, max_cpu_1h système_1h, min_cpu_system_1h. `region`est mappé à la dimension, et `hour` est mappé à la colonne de temps.

La MixedMeasureMappings propriété à TargetConfiguration. TimestreamConfigurationindique comment mapper les mesures aux enregistrements de mesures multiples dans la table dérivée.

Dans cet exemple spécifique, avg_mem_cached_1h, min_mem_free_1h, max_mem_used_1h sont utilisés dans un enregistrement multi-mesures avec le nom de mesure mem_aggregates, total_disk_io_writes_1h, avg_disk_used_1h, avg_disk_free_1h sont utilisés dans un autre enregistrement multi-mesures avec le nom de mesure disk_aggregates, et enfin, max_cpu_user_1h, max_cpu_system_1h, min_cpu_system_1h sont utilisés dans un autre enregistrement multi-mesures portant le nom de mesure cpu_aggregates.

Dans ces mappages, vous pouvez également éventuellement renommer la colonne des résultats de la requête afin qu'elle ait un nom d'attribut différent dans la table de destination. TargetMultiMeasureAttributeName Par exemple, la colonne de résultat avg_mem_cached_1h est renommée en, total_disk_io_writes_1h est renommée en avgMemCached, etc. totalIOWrites

Lorsque vous définissez les mappages pour des enregistrements à mesures multiples, Timestream for LiveAnalytics inspecte chaque ligne des résultats de la requête et ignore automatiquement les valeurs des colonnes contenant des valeurs. NULL Par conséquent, dans le cas de mappages comportant plusieurs noms de mesures, si toutes les valeurs de colonne de ce groupe dans le mappage concernent NULL une ligne donnée, aucune valeur pour ce nom de mesure n'est ingérée pour cette ligne.

Par exemple, dans le mappage suivant, avg_mem_cached_1h, min_mem_free_1h et max_mem_used_1h sont mappés pour mesurer le nom mem_aggregates. Si, pour une ligne donnée du résultat de la requête, toutes ces valeurs de colonne sont les suivantesNULL, Timestream for LiveAnalytics n'ingérera pas la mesure mem_aggregates pour cette ligne. Si les neuf colonnes d'une ligne donnée le sontNULL, une erreur utilisateur sera signalée dans votre rapport d'erreur.

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

Voici le schéma de la table de destination une fois cette requête planifiée matérialisée.

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
region	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
minCpuIdle	double	MULTI
max_cpu_system_1h	double	MULTI
max_cpu_utilisateur_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

Les mesures correspondantes obtenues à l'aide d'une SHOW MEASURES requête sont les suivantes.

nom_mesure	data_type	Dimensions
cpu_aggregates	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
disk_aggregates	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
mem_aggregates	multi	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

Exemple : mappage des résultats vers des enregistrements à mesure unique avec le nom de la mesure à partir des résultats de requête

Voici un exemple de requête planifiée dont les résultats sont matérialisés dans des enregistrements à mesure unique. Dans cet exemple, le résultat de la requête contient la colonne measure_name dont les valeurs seront utilisées comme noms de mesures dans la table cible. Vous utilisez l' MixedMeasureMappings attribut dans le TargetConfiguration. TimestreamConfiguration pour spécifier le mappage de la colonne du résultat de la requête avec la mesure scalaire de la table cible.

Dans l'exemple de définition suivant, le résultat de la requête est censé contenir neuf valeurs measure_name distinctes. Vous listez tous ces noms de mesures dans le mappage et vous spécifiez la colonne à utiliser pour la valeur d'une seule mesure pour ce nom de mesure. Par exemple, dans ce mappage, si le nom de mesure memory_cached apparaît pour une ligne de résultat donnée, la valeur de la colonne avg_1h est utilisée comme valeur de la mesure lorsque les données sont écrites dans la table cible. Vous pouvez éventuellement l'utiliser TargetMeasureName pour fournir un nouveau nom de mesure pour cette valeur.

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

Voici le schéma de la table de destination une fois cette requête planifiée matérialisée. Comme vous pouvez le voir sur le schéma, la table utilise des enregistrements à mesure unique. Si vous listez le schéma de mesures de la table, vous verrez les neuf mesures écrites sur la base du mappage fourni dans la spécification.

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
region	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
valeur_mesure : double	double	MEASURE_VALUE

Les mesures correspondantes obtenues à l'aide d'une SHOW MEASURES requête sont les suivantes.

nom_mesure	data_type	Dimensions
AvgMemCached	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
MinMemFree	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
cpu_idle	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
système_processeur	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
utilisateur_processeur	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
disk_free	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
disk_used	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
maxMemUsed	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
total-disk-io-writes	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

Exemple : mappage des résultats vers des enregistrements à mesure unique avec des colonnes de résultats de requête comme noms de mesures

Dans cet exemple, vous avez une requête dont les résultats ne comportent pas de colonne de nom de mesure. Vous souhaitez plutôt utiliser le nom de la colonne du résultat de la requête comme nom de mesure lorsque vous mappez la sortie à des enregistrements à mesure unique. Plus tôt, il y avait un exemple où un résultat similaire était écrit dans un enregistrement à mesures multiples. Dans cet exemple, vous verrez comment le mapper à des enregistrements à mesure unique si cela correspond à votre scénario d'application.

Encore une fois, vous spécifiez ce mappage à l'aide de la MixedMeasureMappings propriété dans TargetConfiguration. TimestreamConfiguration. Dans l'exemple suivant, vous voyez que le résultat de la requête comporte neuf colonnes. Vous utilisez les colonnes de résultats comme noms de mesures et les valeurs comme valeurs de mesure unique.

Par exemple, pour une ligne donnée dans le résultat de la requête, le nom de colonne avg_mem_cached_1h est utilisé comme nom de colonne et valeur associée à la colonne, et avg_mem_cached_1h est utilisé comme valeur de mesure pour l'enregistrement à mesure unique. Vous pouvez également TargetMeasureName utiliser un autre nom de mesure dans la table cible. Par exemple, pour les valeurs de la colonne sum_1h, le mappage indique d'utiliser total_disk_io_writes_1h comme nom de mesure dans la table cible. Si la valeur d'une colonne est égale à cette valeurNULL, la mesure correspondante est ignorée.

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

Voici le schéma de la table de destination une fois cette requête planifiée matérialisée. Comme vous pouvez le constater, la table cible stocke des enregistrements avec des valeurs à mesure unique de type double. De même, le schéma de mesure de la table indique les neuf noms de mesures. Notez également que le nom de mesure total_disk_io_writes_1h est présent puisque le mappage a renommé sum_1h en total_disk_io_writes_1h.

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
region	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
valeur_mesure : double	double	MEASURE_VALUE

Les mesures correspondantes obtenues à l'aide d'une SHOW MEASURES requête sont les suivantes.

nom_mesure	data_type	Dimensions
avg_disk_free_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
avg_disk_used_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
avg_mem_cached_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
max_cpu_system_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
max_cpu_utilisateur_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
max_mem_used_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
min_cpu_idle_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
min_mem_free_1h	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]
total-disk-io-writes	double	[{'dimension_name' : 'région', 'data_type' : 'valeur'}]

Messages de notification de requêtes planifiées

Cette section décrit les messages envoyés par Timestream LiveAnalytics lors de la création, de la suppression, de l'exécution ou de la mise à jour de l'état d'une requête planifiée.

Nom du message de notification	Structure	Description
CreatingNotificationMessage	`CreatingNotificationMessage { String arn; NotificationType type; }`	Ce message de notification est envoyé avant l'envoi de la réponse pour`CreateScheduledQuery` . La requête planifiée est activée après l'envoi de cette notification. arn - Le ARN nom de la requête planifiée en cours de création. tapez - SCHEDULED _ QUERY _ CREATING
UpdateNotificationMessage	`UpdateNotificationMessage { String arn; NotificationType type; QueryState state; }`	Ce message de notification est envoyé lorsqu'une requête planifiée est mise à jour. Timestream for LiveAnalytics peut désactiver automatiquement la requête planifiée

Nom du message de notification	Structure	Description
		en cas d'erreur irrécupérable, telle que : • AssumeRole échec • Toute erreur 4xx rencontrée lors de la communication avec KMS lorsqu'une KMS clé gérée par le client est spécifiée. • Toute erreur 4xx rencontrée lors de l'exécution de la requête planifiée. • Toute erreur 4xx rencontrée lors de l'ingestion des résultats de la requête arn - Le nom ARN de la requête planifiée en cours de mise à jour. tapez - SCHEDULED _ QUERY _ UPDATE État - ENABLED ou DISABLED

Nom du message de notification	Structure	Description
DeleteNotificationMessage	`DeletionNotificationMessage { String arn; NotificationType type; }`	Ce message de notification est envoyé lorsqu'une requête planifiée a été supprimée. arn - Le ARN nom de la requête planifiée en cours de création. tapez - SCHEDULED _ QUERY _ DELETED

Nom du message de notification	Structure	Description

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 {`
		Ce message de notification est envoyé une fois que la requête planifiée a été exécutée et que les résultats ont été correctement ingérés.
		
		ARN- Le ARN nom de la requête planifiée en cours de suppression.
		
		NotificationType- AUTO _ TRIGGER _ SUCCESS ou MANUAL _ TRIGGER _SUCCESS.
		
		nextInvocationEpoch hDeuxièmement - La prochaine fois que Timestream for LiveAnalytics exécutera la requête planifiée.
		
		runSummary- Informations relatives à l'exécution planifiée de la requête.

Nom du message de notification	Structure	Description

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

Nom du message de notification	Structure	Description

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;`
		Ce message de notification est envoyé en cas d'échec lors de l'exécution planifiée d'une requête ou lors de l'ingestion des résultats de la requête.
		
		arn - Le ARN nom de la requête planifiée en cours d'exécution.
		
		tapez - AUTO _ TRIGGER _ FAILURE ou MANUAL _ TRIGGER _FAILURE.
		
		runSummary- Informations relatives à l'exécution planifiée de la requête.

Nom du message de notification	Structure	Description

	}
	
	S3ReportLocation {
	String bucketName;
	String objectKey;
	}	

Rapports d'erreur liés aux requêtes planifiées

Cette section décrit l'emplacement, le format et les raisons des rapports d'erreur générés par Timestream lorsque des erreurs sont détectées LiveAnalytics lors de l'exécution de requêtes planifiées.

Rubriques

- Motifs des rapports d'erreur liés aux requêtes planifiées
- Emplacement des rapports d'erreur liés aux requêtes planifiées
- Format des rapports d'erreurs liés aux requêtes planifiées
- Types d'erreurs de requêtes planifiées
- Exemple de rapports d'erreur liés aux requêtes planifiées

Motifs des rapports d'erreur liés aux requêtes planifiées

Des rapports d'erreurs sont générés pour les erreurs récupérables. Aucun rapport d'erreur n'est généré pour les erreurs irrécupérables. Timestream for LiveAnalytics peut désactiver automatiquement les requêtes planifiées lorsque des erreurs irrécupérables sont rencontrées. Il s'agit des licences suivantes :

- `AssumeRole`échec
- Toute erreur 4xx rencontrée lors de la communication avec une clé gérée par le client KMS lorsqu'une clé gérée par le client KMS est spécifiée
- Toute erreur 4xx rencontrée lors de l'exécution d'une requête planifiée
- Toute erreur 4xx rencontrée lors de l'ingestion des résultats de la requête

Pour les erreurs irréparables, Timestream for LiveAnalytics envoie une notification d'échec avec un message d'erreur irrécupérable. Une notification de mise à jour est également envoyée pour indiquer que la requête planifiée est désactivée.

Emplacement des rapports d'erreur liés aux requêtes planifiées

L'emplacement d'un rapport d'erreur de requête planifiée respecte la convention de dénomination suivante :

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

Voici un exemple de requête planifiée ARN :

```
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 indique les requêtes planifiées automatiquement planifiées par Timestream pour et LiveAnalytics *Manual* indique les requêtes planifiées déclenchées manuellement par un utilisateur via une `ExecuteScheduledQuery` API action dans Amazon Timestream LiveAnalytics for Query. Pour plus d'informations sur`ExecuteScheduledQuery`, voir ExecuteScheduledQuery.

Format des rapports d'erreurs liés aux requêtes planifiées

Les rapports d'erreur ont le JSON format suivant :

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

Types d'erreurs de requêtes planifiées

L'`Error`objet peut être de trois types :

- Erreurs d'ingestion d'enregistrements

```
{
```

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

- Erreurs d'analyse et de validation des lignes

```
{
    "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.
}
```

- Erreurs générales

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

Exemple de rapports d'erreur liés aux requêtes planifiées

Voici un exemple de rapport d'erreur produit en raison d'erreurs d'ingestion.

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

Modèles de requêtes planifiées et exemples

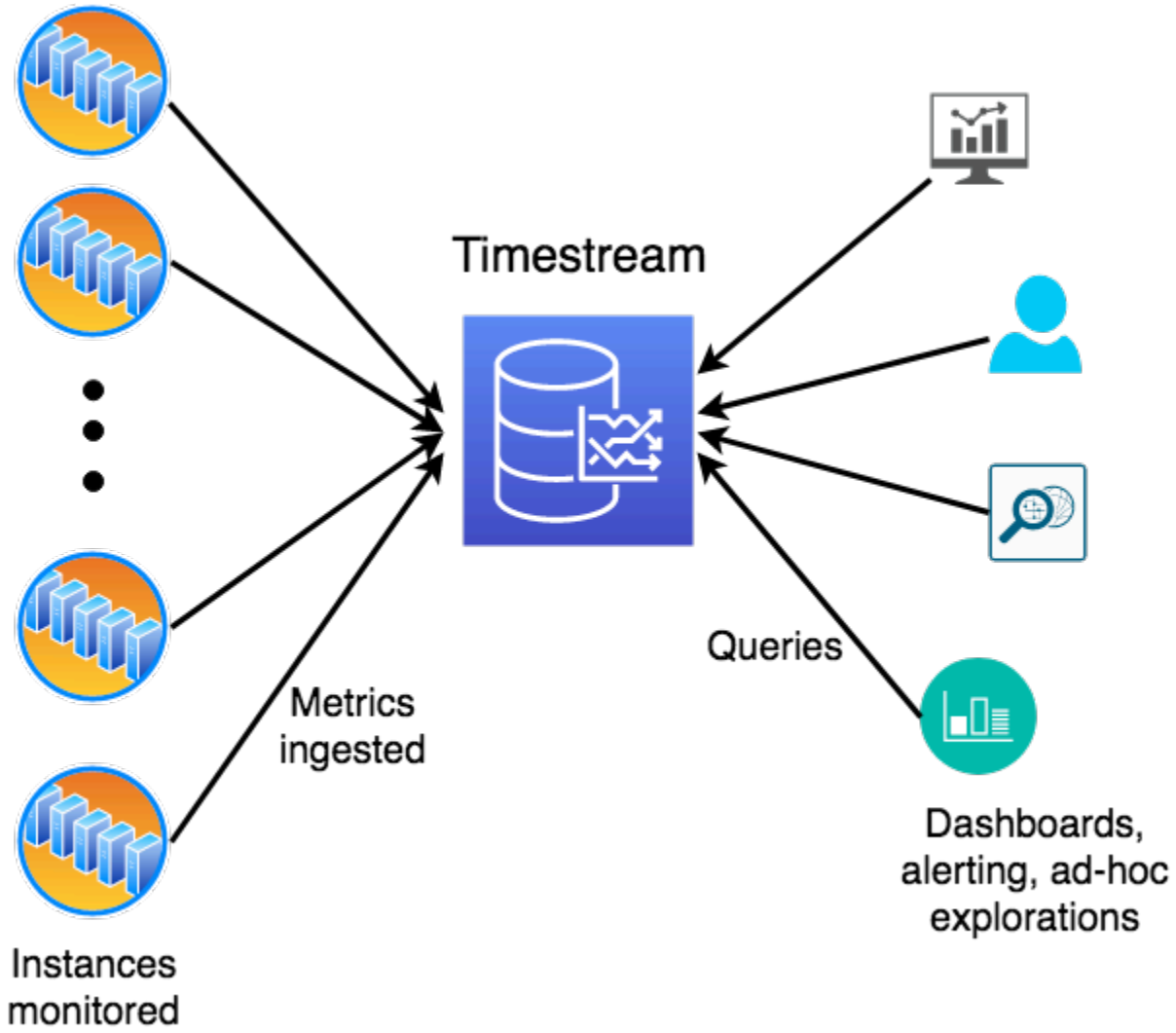
Cette section décrit les modèles d'utilisation pour les requêtes planifiées ainsi que end-to-end des exemples.

Rubriques

- Exemple de schéma de requêtes planifiées
- Modèles de requêtes planifiés
- Exemples de requêtes planifiées

Exemple de schéma de requêtes planifiées

Dans cet exemple, nous allons utiliser un exemple d'application imitant un DevOps scénario de surveillance des métriques provenant d'un grand parc de serveurs. Les utilisateurs souhaitent être alertés en cas d'utilisation anormale des ressources, créer des tableaux de bord sur le comportement et l'utilisation globaux de la flotte, et effectuer des analyses sophistiquées sur des données récentes et historiques afin de trouver des corrélations. Le schéma suivant illustre la configuration dans laquelle un ensemble d'instances surveillées émet des métriques vers Timestream pour. LiveAnalytics Un autre groupe d'utilisateurs simultanés émet des requêtes pour des alertes, des tableaux de bord ou des analyses ad hoc, dans le cadre desquelles les requêtes et l'ingestion s'exécutent en parallèle.



L'application surveillée est modélisée comme un service hautement évolutif déployé dans plusieurs régions du monde. Chaque région est ensuite subdivisée en un certain nombre d'unités d'échelle

appelées cellules qui présentent un certain niveau d'isolation en termes d'infrastructure au sein de la région. Chaque cellule est ensuite subdivisée en silos, qui représentent un niveau d'isolation logicielle. Chaque silo possède cinq microservices qui constituent une instance isolée du service. Chaque microservice possède plusieurs serveurs dotés de différents types d'instances et de versions de système d'exploitation, qui sont déployés dans trois zones de disponibilité. Ces attributs qui identifient les serveurs émettant les métriques sont modélisés sous forme de dimensions dans Timestream for. LiveAnalytics Dans cette architecture, nous avons une hiérarchie de dimensions (telles que region, cell, silo et microservice_name) et d'autres dimensions qui recoupent la hiérarchie (telles que instance_type et availability_zone).

L'application émet une variété de métriques (telles que cpu_user et memory_free) et d'événements (tels que task_completed et gc_reclaimed). Chaque métrique ou événement est associé à huit dimensions (telles que la région ou la cellule) qui identifient de manière unique le serveur qui l'émet. Les données sont écrites avec les 20 métriques stockées ensemble dans un enregistrement multi-mesures avec les métriques de nom de mesure et les 5 événements sont stockés ensemble dans un autre enregistrement multi-mesures avec les événements de nom de mesure. Le modèle de données, le schéma et la génération de données se trouvent dans le générateur de données open source. Outre le schéma et les distributions de données, le générateur de données fournit un exemple d'utilisation de plusieurs rédacteurs pour ingérer des données en parallèle, en utilisant la mise à l'échelle d'ingestion de Timestream pour LiveAnalytics ingérer des millions de mesures par seconde. Ci-dessous, nous montrons le schéma (schéma de table et de mesure) et quelques exemples de données de l'ensemble de données.

Rubriques

- Enregistrements à mesures multiples
- Enregistrements à mesure unique

Enregistrements à mesures multiples

Schéma de table

Vous trouverez ci-dessous le schéma du tableau une fois que les données sont ingérées à l'aide d'enregistrements à mesures multiples. C'est le résultat de la DESCRIBE requête. En supposant que les données soient ingérées dans une base de données raw_data et une table devops, voici la requête.

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

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
availability_zone	varchar	DIMENSION
nom_microservice	varchar	DIMENSION
nom_instance	varchar	DIMENSION
nom_processus	varchar	DIMENSION
os_version	varchar	DIMENSION
jdk_version	varchar	DIMENSION
cellule	varchar	DIMENSION
region	varchar	DIMENSION
silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
sans mémoire	double	MULTI
cpu_steal	double	MULTI
cpu_iowait	double	MULTI
utilisateur_processeur	double	MULTI
mémoire_mise en cache	double	MULTI
disk_io_reads	double	MULTI
cpu_hi	double	MULTI
latence par lecture	double	MULTI

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
octets_out du réseau	double	MULTI
cpu_idle	double	MULTI
disk_free	double	MULTI
mémoire_utilisée	double	MULTI
système_processeur	double	MULTI
descripteurs de fichiers en cours d'utilisation	double	MULTI
disk_used	double	MULTI
cpu_nice	double	MULTI
disk_io_writes	double	MULTI
cpu_si	double	MULTI
latence par écriture	double	MULTI
network_bytes_in	double	MULTI
État_final de la tâche	varchar	MULTI
gc_pause	double	MULTI
tâche_terminée	bigint	MULTI
gc_reclaimed	double	MULTI

Schéma de mesure

Vous trouverez ci-dessous le schéma de mesure renvoyé par la SHOW MEASURES requête.

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

nom_mesure	data_type	Dimensions
événements	multi	[{"data_type » « varchar », « dimension_name » « availability_zone »}, {"data_type » « varchar », « dimension_name » « microservice_name »}, {"data_type » varchar », « dimension_name » 1_instance_name «}, {" data_type » « varchar », « dimension_name » « process_name » _name "}, {" data_type »¨varchar », « dimension_name »¨jdk_vers ion "}, {" data_type »¨varchar », « dimension_name » ¨varchar », « dimension _name »¨varchar », « dimension_name »¨region «}, {" data_type »¨varchar », « dimension_name »¨silo}]
metrics	multi	[{"data_type » « varchar », « dimension_name » « availability_zone »}, {"data_type » « varchar », « dimension_name » « microservice_name »}, {"data_type » varchar », « dimension_name » 1_instance_name «}, {" data_type » .varchar », « dimension_name » _version "}, {" data_type »¨varchar »,

nom_mesure	data_type	Dimensions
		« dimension_name »¨cell "}, {" data_type »¨varchar », « dimension_name »¨region «}, {" data_type »¨varchar », « dimension_name »¨silo «}, {" data_type »¨varchar », « dimension_name »¨instance _type »}]

Exemples de données

re	C	S	a	n	n	in	o	n	jo	n	H	u	sy	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	o	d	s	É	g	tâ	m	ge	se	cla
u	u	u	u	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		z	e					1		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-							
	-	-	-1		-						1																												
	c	c			a																																		
		s			u																																		
		il			-																																		
					e																																		
					-2																																		
					c																																		
					-																																		
					si																																		
					0																																		
					m																																		
					c																																		

re	C	S	a ity	n c e	n n	ir ty	o n	n ss	jo o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n yt	o t d ré	d u d fi e c d ti	sa m	É l d la tâ	g	tâ m	ge s_tecla mé ed
u e	u e - c	u e - c s il	u e -1	a	i- z - a u - e -2 c - si 0 m c	m e	A			m	1 1 1.	5	0	3	0	0	0	0	0	9	3	5	3	2	9	5.	3	7	5	6	2				

re	C	S	a ity	n cı e	n n	ir ty	o n	n ss	jo o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n y	o t d ré	d u d fi e c d ti	sa m	É l d la tâ	g	tâ m	ge m s_ted ecla sed
u e	u e - c	u e - c s il	u e -1	a	i- z - a u - e -2 c - si 0 m c	m e	A			m	1 1 1.	4	0	4	0	0	0	0	0	9	4	5	3.	8	3	7.	6	8	5	4	8				

re	C	S	a ity	n c e	n n	ir ty	o n	n s	jo o	n e	H	u u s	s r	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n yt	o t d ré	d u d fi e c d ti	s m	É l d la tâ	g	tâ m	gs_tecla méed
u e	u e - c	u e - c s il	u e -1	a	i- z - a u - e -2 c - si 0 m c	m e	A			m	1 1 1.	3	0	5	0	0	0	0	0	4	7	2	4	4	3	8	6	2	1	2	1				

re	C	S	a ity	n c e	n n	ir ty	o n	n ss	jo o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n yt	o t d ré	d u d fi e co d ti	sa m	É l d la tâ	g	tâ m	ge mé s_tecla sed
u e	u e - c	u e - c s il	u e -1	a	i- z - a u - e -2 c - si 0 m c	m e	A			m	1 1 1.	5	0	3	0	0	0	0	0	5	3	8	5	7	1	8	6	7	6	5	3				

re	C	S	a ity	n c e	n n	ir ty	o n	n ss	jc o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n y	o t d ré	d u d fi e c d ti	sa m	É l d la tâ	g	tâ m	ge me / se_ecla sed
u e	u e - c	u e - c s il	u e -1	a	i- z - a u - e -2 c - si 0 m c			g ir	J	é s	1 1 1.																				7	S W _ R	8	3	6 4.8

re	C	S	a	n	n	in	o	n	jo	n	H	u	sy	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	o	d	sa	É	g	tâ	ge	se	cla
			ity	ce	n	ty	n	ss	o	e		us	ro			t				til	is	e	ple	rit	pé			yt	t d	du fi e c ti		l d la tâ		m	mé		ed
us e	us e- c	us e- c sil	us e -1	a	i- za - au us - e -2 co - si 0 m co			gi	Jl	és	1 1 1																				7	S IT	2	4	73	15 JL	W

re	C	S	a ity	n ci e	n n	ir ty	o n	n ss	jc o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n y	o t d ré	d u d fi e c d ti	sa m	É l d la tâ	g	tâ m	ge me s_tecla sed
u e	u e - c	u e - c s il	u e -1	a	i- za - a u - e -2 c - si 0 m c			g ir	J	é s	1 1 1.																				6	S IT T	6	2	72.3W JL

re	C	S	a ity	n cr e	n n	ir ty	o n	n ss	jo o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n yt	o t d ré	d u d fi e c d ti	sa m	É l d la tâ	g	tâ m	ge mé	s_tecla ed
u e	u e - c	u e - c s il	u e -1	a	i- z - a u - e -2 c - si 0 m c			g ir	J	é s	1 1 1.																				2	S IT T	5	1	9	9_W JL

re	C	S	a ity	n ci e	n n	ir ty	o n	n ss	jc o	n e	H	u u s	sy ro	c	c	c t	c	c	c	m til	m is e ca	d e	la p le	d ri	la p é	d	d	n y	o t d ré	d u d fi e co d ti	sa m	É l d la tâ	g	tâ m	gs_ecla med
u e	u e - c	u e - c s il	u e -1	a	i- za - a u - e -2 c - si 0 m c			g ir	J	é s	1 1 1.																				3	S W _ R	7	2	83.9

Enregistrements à mesure unique

Timestream for vous permet LiveAnalytics également d'ingérer les données avec une seule mesure par enregistrement de série chronologique. Vous trouverez ci-dessous les détails du schéma en cas d'ingestion à l'aide d'enregistrements de mesures uniques.

Schéma de table

Vous trouverez ci-dessous le schéma du tableau une fois que les données sont ingérées à l'aide d'enregistrements à mesures multiples. C'est le résultat de la DESCRIBE requête. En supposant que les données soient ingérées dans une base de données raw_data et une table devops, voici la requête.

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

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
availability_zone	varchar	DIMENSION
nom_du_microservice	varchar	DIMENSION
nom_instance	varchar	DIMENSION
nom_processus	varchar	DIMENSION
os_version	varchar	DIMENSION
jdk_version	varchar	DIMENSION
cellule	varchar	DIMENSION
region	varchar	DIMENSION
silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
time	timestamp	TIMESTAMP
valeur_mesure : double	double	MEASURE_VALUE
value_mesure : :bigint	bigint	MEASURE_VALUE
valeur_mesure : varchar	varchar	MEASURE_VALUE

Schéma de mesure

Vous trouverez ci-dessous le schéma de mesure renvoyé par la SHOW MEASURES requête.

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

nom_mesure	data_type	Dimensions
cpu_hi	double	[{'dimension_name' : 'availabi lity_zone', 'data_type' : 'valeur'} , {'dimension_name' : 'microser vice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimensi on_name' : 'os_version', 'data_type' : 'valeur'}, {'dimensi on_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_typ e', 'data_type' : 'valeur'}]
cpu_idle	double	[{'dimension_name' : 'availabi lity_zone', 'data_type' : 'valeur'} , {'dimension_name' : 'microser vice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimensi on_name' : 'os_version', 'data_type' : 'valeur'}, {'dimensi on_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'},

nom_mesure	data_type	Dimensions
		{'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]
cpu_iowait	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
cpu_nice	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
cpu_si	double	[{'dimension_name' : 'availabi lity_zone', 'data_type' : 'valeur'} , {'dimension_name' : 'microser vice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimensi on_name' : 'os_version', 'data_type' : 'valeur'}, {'dimensi on_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_typ e', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
cpu_steal	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
système_processeur	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
utilisateur_processeur	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
disk_free	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
disk_io_reads	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
disk_io_writes	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
disk_used	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
descripteurs de fichiers en cours d'utilisation	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]
gc_pause	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'varchar'}, {'dimension_name' : 'microservice_name', 'data_type' : 'varchar'}, {'dimension_name' : 'nom_de_processus', 'data_type' : 'valeur'}, {'dimension_name' : 'jdk_version', 'data_type' : 'variable'}, {'dimension_name' : 'cellule', 'data_type' : 'valeur'}, {'dimension_name' : « région », « type de donnée » : « variable »}, {'dimension_name' : 'silo', 'data_type' : 'variable'}]

nom_mesure	data_type	Dimensions
gc_reclaimed	double	[{'dimension_name' : 'availabi lity_zone', 'data_type' : 'varchar'}, {'dimension_name' : 'microservice_name', 'data_typ e' : 'varchar'}, {'dimensi on_name' : 'nom_de_p rocessus', 'data_type' : 'valeur'}, {'dimension_name' : 'jdk_version', 'data_typ e' : 'variable'}, {'dimensi on_name' : 'cellule', 'data_typ e' : 'valeur'}, {'dimensi on_name' : « région », « type de donnée » : « variable »}, {'dimension_name' : 'silo', 'data_type' : 'variable'}]
latence par lecture	double	[{'dimension_name' : 'availabi lity_zone', 'data_type' : 'valeur'} , {'dimension_name' : 'microser vice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimensi on_name' : 'os_version', 'data_type' : 'valeur'}, {'dimensi on_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_typ e', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
latence par écriture	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
mémoire_mise en cache	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
sans mémoire	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'varchar'}, {'dimension_name' : 'microservice_name', 'data_type' : 'varchar'}, {'dimension_name' : 'nom_de_processus', 'data_type' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'variable'}, {'dimension_name' : 'jdk_version', 'data_type' : 'variable'}, {'dimension_name' : 'cellule', 'data_type' : 'variable'}, {'dimension_name' ':' region ',' data_type ':' variable '}, {' dimension_name ':' silo ',' data_type ':' variable '}, {' dimension_name ':' instance_type ', 'data_ type' : 'variable'}]

nom_mesure	data_type	Dimensions
mémoire_utilisée	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
network_bytes_in	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]

nom_mesure	data_type	Dimensions
octets_out du réseau	double	[{'dimension_name' : 'availability_zone', 'data_type' : 'valeur'}, {'dimension_name' : 'microservice_name', 'data_type' : 'variable'}, {'dimension_name' : 'nom_instance', 'type de donnée' : 'variable'}, {'dimension_name' : 'os_version', 'data_type' : 'valeur'}, {'dimension_name' : 'cellule', 'type de donnée' : 'variable'}, {'dimension_name' : 'région', 'data_type' : 'variable'}, {'dimension_name' : 'silo', 'type de données' : 'instance'}, {'dimension_name' : 'ance_type', 'data_type' : 'valeur'}]
tâche_terminée	bigint	[{'dimension_name' : 'availability_zone', 'data_type' : 'varchar'}, {'dimension_name' : 'microservice_name', 'data_type' : 'varchar'}, {'dimension_name' : 'nom_de_processus', 'data_type' : 'valeur'}, {'dimension_name' : 'jdk_version', 'data_type' : 'variable'}, {'dimension_name' : 'cellule', 'data_type' : 'valeur'}, {'dimension_name' : « région », « type de donnée » : « variable »}, {'dimension_name' : 'silo', 'data_type' : 'variable'}]

nom_mesure	data_type	Dimensions
État_final de la tâche	varchar	[{'dimension_name' : 'availability_zone', 'data_type' : 'varchar'}, {'dimension_name' : 'microservice_name', 'data_type' : 'varchar'}, {'dimension_name' : 'nom_de_processus', 'data_type' : 'valeur'}, {'dimension_name' : 'jdk_version', 'data_type' : 'variable'}, {'dimension_name' : 'cellule', 'data_type' : 'valeur'}, {'dimension_name' : « région », « type de donnée » : « variable »}, {'dimension_name' : 'silo', 'data_type' : 'variable'}]

Exemples de données

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZs\ - hercu eu-west -1- cell-9 - silo-2 0000(		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	cpu_l	34:57	0,871		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
		mazo com												
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	cpu_i	34:57	3,462		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	cpu_i t	34:57	0,102		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	cpu_r	34:57	0,630		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	cpu_s	34:57	0,164		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	cpu_s	34:57	0,107		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	systè roces	34:57	0,457		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	utilisa ur_pr seur	34:57	94,20		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	disk_	34:57	72,51		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	disk_i eads	34:57	81,73		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	disk_i rites	34:57	77,11		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	disk_	34:57	89,42		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	descr urs de fichie en cours d'utili tion	34:57	30,08		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com	serve		JDK_	eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2		gc_pa	34:57	60,28		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com	serve		JDK_	eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2		gc_re med	34:57	75,28		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	laten par lectur	34:57	8,076		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	laten par écritu	34:57	58,11		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	mém ise en cache	34:57	87,56		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com	serve		JDK_	eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2		sans mém	34:57	18,95		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	sans mém	34:57	97,20		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	mém tilisée	34:57	12,37		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	netwo ytes_	34:57	31,02		
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com		AL20		eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2	r5.4xl e	octets t du résea	34:57	0,514		

availa ity_zo	nom_ crose e	nom_ nce	nom_ ssus	os_ve n	jdk_v on	Cellul	regio	Silo	instar type	nom_ e	Heure	valeu sure : doubl	value ure : :	valeur_me sure : varchar
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com	serve		JDK_	eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2		tâche minée	34:57		69	
eu-west--1	hercu	i-zaZsv - hercu eu-west -1- cell-9 - silo-2 0000(mazo com	serve		JDK_	eu-west- - cell-9	eu-west-	euwe cell-9 si lo-2		État_ l de la tâche	34:57			SUCCESS_ ITH_RESUL T

Modèles de requêtes planifiés

Dans cette section, vous trouverez quelques modèles courants d'utilisation d'Amazon Timestream LiveAnalytics pour les requêtes planifiées afin d'optimiser vos tableaux de bord afin qu'ils se chargent

plus rapidement et à moindre coût. Les exemples ci-dessous utilisent un scénario d' DevOps application pour illustrer les concepts clés qui s'appliquent aux requêtes planifiées en général, quel que soit le scénario d'application.

Les requêtes planifiées dans Timestream pour vous LiveAnalytics permettent d'exprimer vos requêtes en utilisant toute la SQL surface de Timestream pour. LiveAnalytics Votre requête peut inclure une ou plusieurs tables sources, effectuer des agrégations ou toute autre requête autorisée par le SQL langage de Timestream for, puis matérialiser les résultats LiveAnalytics de la requête dans une autre table de destination dans Timestream for. LiveAnalytics Pour faciliter l'exposition, cette section fait référence à cette table cible d'une requête planifiée en tant que table dérivée.

Les principaux points abordés dans cette section sont les suivants.

- Utilisation d'un simple agrégat au niveau du parc pour expliquer comment définir une requête planifiée et comprendre certains concepts de base.
- Comment combiner les résultats de la cible d'une requête planifiée (la table dérivée) avec les résultats de la table source pour obtenir les avantages en termes de coûts et de performances d'une requête planifiée.
- Quels sont les compromis à faire lors de la configuration de la période d'actualisation des requêtes planifiées ?
- Utilisation de requêtes planifiées pour certains scénarios courants.
  - Suivi du dernier point de données de chaque instance avant une date précise.
  - Valeurs distinctes pour une dimension à utiliser pour renseigner les variables d'un tableau de bord.
- Comment gérez-vous les données arrivées en retard dans le contexte de requêtes planifiées.
- Comment utiliser des exécutions manuelles ponctuelles pour gérer divers scénarios qui ne sont pas directement couverts par les déclencheurs automatisés pour les requêtes planifiées.

Rubriques

Scénario

Les exemples suivants utilisent un scénario de DevOps surveillance décrit dansExemple de schéma de requêtes planifiées.

Les exemples fournissent la définition de requête planifiée dans laquelle vous pouvez insérer les configurations appropriées pour savoir où recevoir les notifications d'état d'exécution pour les requêtes planifiées, où recevoir les rapports sur les erreurs rencontrées lors de l'exécution d'une requête planifiée, et le IAM rôle que la requête planifiée utilise pour effectuer ses opérations.

Vous pouvez créer ces requêtes planifiées après avoir renseigné les options précédentes, créé la table cible (ou dérivée) et les avoir exécutées via le AWSCLI. Supposons, par exemple, qu'une définition de requête planifiée soit stockée dans un fichier`scheduled_query_example.json`. Vous pouvez créer la requête à l'aide de la CLI commande.

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

Dans la commande précédente, le profil transmis à l'aide de l'option --profile doit disposer des autorisations appropriées pour créer des requêtes planifiées. Consultez la section Politiques basées sur l'identité pour les requêtes planifiées pour obtenir des instructions détaillées sur les politiques et les autorisations.

Agrégats simples au niveau de la flotte

Ce premier exemple décrit certains des concepts de base relatifs à l'utilisation de requêtes planifiées à l'aide d'un exemple simple de calcul d'agrégats au niveau du parc. À l'aide de cet exemple, vous allez apprendre ce qui suit.

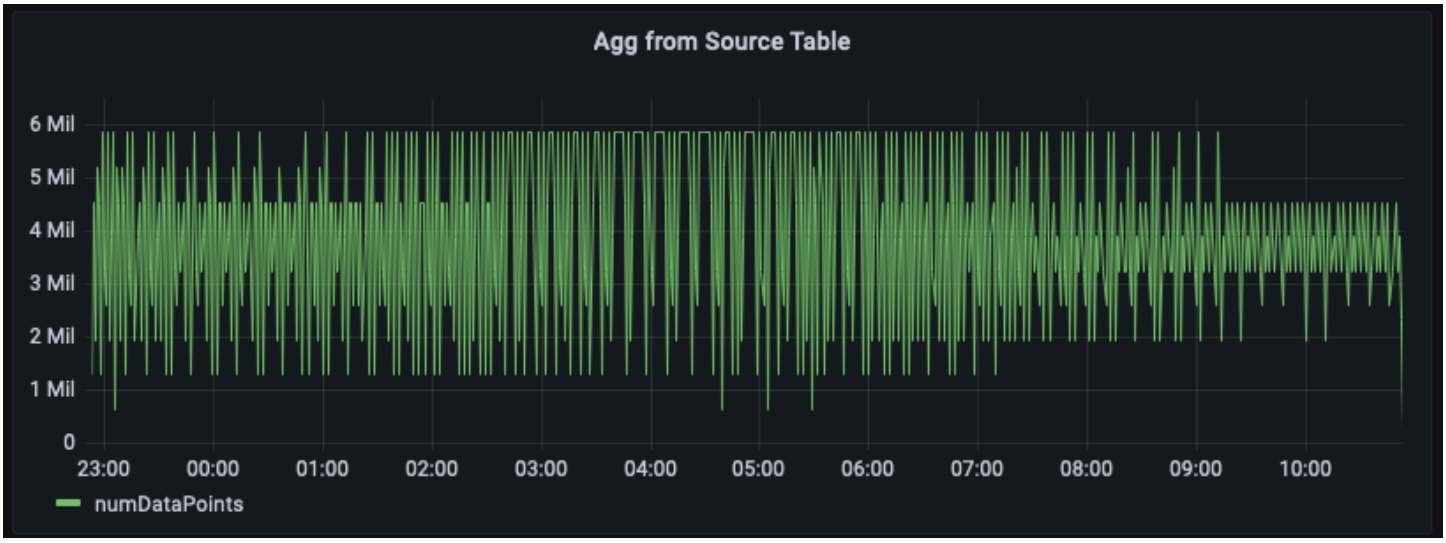
- Comment associer votre requête de tableau de bord utilisée pour obtenir des statistiques agrégées à une requête planifiée.
- Comment Timestream for LiveAnalytics gère l'exécution des différentes instances de votre requête planifiée.
- Comment faire en sorte que différentes instances de requêtes planifiées se chevauchent dans les plages temporelles et comment l'exactitude des données est maintenue dans la table cible pour garantir que votre tableau de bord utilisant les résultats de la requête planifiée vous donne des résultats correspondant au même agrégat calculé sur les données brutes.
- Comment définir la plage horaire et la cadence d'actualisation de votre requête planifiée.

- Comment suivre en libre-service les résultats des requêtes planifiées afin de les ajuster afin que la latence d'exécution des instances de requête soit dans les délais acceptables pour l'actualisation de vos tableaux de bord.

Rubriques

Agrégation à partir des tables sources

Dans cet exemple, vous suivez le nombre de métriques émises par les serveurs d'une région donnée par minute. Le graphique ci-dessous est un exemple illustrant cette série chronologique pour la région us-east-1.



Vous trouverez ci-dessous un exemple de requête pour calculer cet agrégat à partir des données brutes. Il filtre les lignes pour la région us-east-1, puis calcule la somme par minute en prenant en compte les 20 mesures (si measure_name est une métrique) ou 5 événements (si measure_name est un événement). Dans cet exemple, l'illustration graphique montre que le nombre de métriques émises varie entre 1,5 million et 6 millions par minute. Lorsque vous tracez cette série chronologique pendant plusieurs heures (les 12 dernières heures sur cette figure), cette requête sur les données brutes analyse des centaines de millions de lignes.

```
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
```

Requête planifiée pour précalculer les agrégats

Si vous souhaitez optimiser vos tableaux de bord afin de les charger plus rapidement et de réduire vos coûts en analysant moins de données, vous pouvez utiliser une requête planifiée pour précalculer ces agrégats. Les requêtes planifiées dans Timestream for vous LiveAnalytics permettent de matérialiser ces précalculs dans une autre LiveAnalytics table Timestream for, que vous pourrez ensuite utiliser pour vos tableaux de bord.

La première étape de la création d'une requête planifiée consiste à identifier la requête que vous souhaitez précalculer. Notez que le tableau de bord précédent a été dessiné pour la région us-east-1. Cependant, un autre utilisateur peut vouloir obtenir le même agrégat pour une région différente, par exemple us-west-2 ou eu-west-1. Pour éviter de créer une requête planifiée pour chacune de ces requêtes, vous pouvez précalculer l'agrégat pour chaque région et matérialiser les agrégats par région dans un autre Timestream for table. LiveAnalytics

La requête ci-dessous fournit un exemple du précalcul correspondant. Comme vous pouvez le constater, elle est similaire à l'expression de table courante grouped_data utilisée dans la requête sur les données brutes, à deux différences près : 1) elle n'utilise pas de prédicat de région, de sorte que nous pouvons utiliser une seule requête pour précalculer pour toutes les régions ; et 2) elle utilise un prédicat temporel paramétré avec un paramètre spécial @scheduled_runtime qui est expliqué en détail ci-dessous.

```
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
```

La requête précédente peut être convertie en requête planifiée à l'aide de la spécification suivante. Un nom est attribué à la requête planifiée, qui est un mnémotechnique convivial. Il inclut ensuite le QueryString, a ScheduleConfiguration, qui est une expression cron. Il indique TargetConfiguration qui fait correspondre les résultats de la requête à la table de destination dans Timestream for. LiveAnalytics Enfin, il spécifie un certain nombre d'autres configurations, telles que le NotificationConfiguration, où les notifications sont envoyées pour les exécutions individuelles de la requête, ErrorReportConfiguration où un rapport est rédigé au cas où la requête rencontrerait des erreurs, et le ScheduledQueryExecutionRoleArn, quel est le rôle utilisé pour effectuer les opérations pour la requête planifiée.

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

Dans l'exemple, le ScheduleExpression cron (0/5 * * * ? *) implique que la requête est exécutée une fois toutes les 5 minutes aux 5, 10, 15,... minutes de chaque heure de chaque jour. Ces horodatages lorsqu'une instance spécifique de cette requête est déclenchée sont ce qui se traduit par le paramètre @scheduled_runtime utilisé dans la requête. Par exemple, considérez l'instance de cette requête planifiée exécutée le 2021-12-01 00:00:00. Pour cette instance, le paramètre @scheduled_runtime est initialisé à l'horodatage 2021-12-01 00:00:00 lors de l'appel de la requête. Par conséquent, cette instance spécifique s'exécutera à l'horodatage 2021-12-01 00:00:00 et calculera les agrégats par minute entre le 2021-11-30 23:50:00 et le 2021-12-01 00:01:00. De même, l'instance suivante de cette requête est déclenchée à l'horodatage 2021-12-01 00:05:00 et dans ce cas, la requête calculera des agrégats par minute à partir de la plage horaire 2021-11-30 23:55:00 à 2021-12-01 00:06:00. Par conséquent, le paramètre @scheduled_runtime fournit une requête planifiée pour précalculer les agrégats pour les plages de temps configurées en utilisant le temps d'invocation des requêtes.

Notez que les deux instances suivantes de la requête se chevauchent dans leurs plages temporelles. C'est quelque chose que vous pouvez contrôler en fonction de vos besoins. Dans ce cas, ce chevauchement permet à ces requêtes de mettre à jour les agrégats en fonction des données dont l'arrivée a été légèrement retardée, jusqu'à 5 minutes dans cet exemple. Pour garantir l'exactitude des requêtes matérialisées, Timestream for LiveAnalytics garantit que la requête du 2021-12-01 00:05:00 ne sera exécutée qu'une fois la requête du 2021-12-01 00:00:00 terminée et que les résultats de ces dernières requêtes peuvent mettre à jour tout agrégat précédemment matérialisé en utilisant si une nouvelle valeur est générée. Par exemple, si certaines données à l'horodatage 2021-11-30 23:59:00 sont arrivées après l'exécution de la requête pour 2021-12-01 00:00:00 mais avant la requête pour 2021-12-01 00:05:00, alors l'exécution au 2021-12-01 00:05:00 recalculera les

agrégats pour la minute 2021-11-30 23:59:00 et cela entraînera la mise à jour de l'agrégat précédent avec la valeur nouvellement calculée. Vous pouvez vous fier à cette sémantique des requêtes planifiées pour trouver un compromis entre la rapidité avec laquelle vous mettez à jour vos précalculs et la manière dont vous pouvez gérer correctement certaines données en cas d'arrivée différée. D'autres considérations sont abordées ci-dessous concernant la manière dont vous pouvez concilier cette cadence d'actualisation avec la fraîcheur des données et comment vous gérez la mise à jour des agrégats pour les données qui arrivent encore plus tard ou si la source du calcul planifié contient des valeurs mises à jour qui nécessiteraient le recalcul des agrégats.

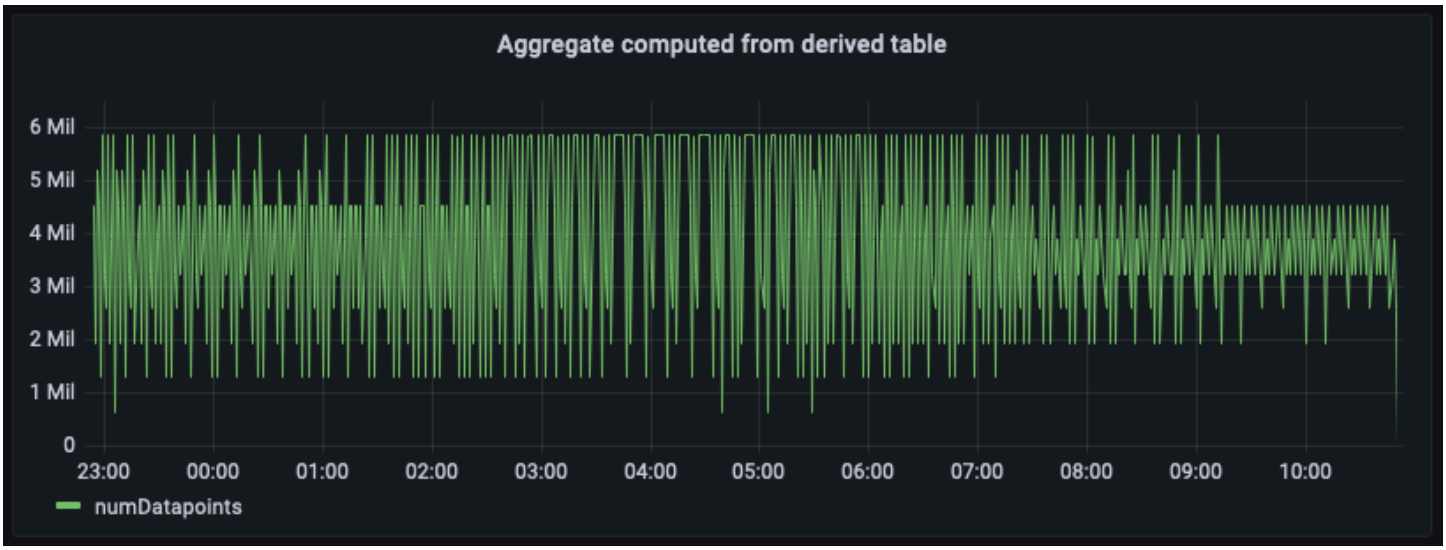
Chaque calcul planifié possède une configuration de notification dans laquelle Timestream for LiveAnalytics envoie une notification de chaque exécution d'une configuration planifiée. Vous pouvez configurer un SNS sujet pour recevoir des notifications pour chaque appel. Outre le statut de réussite ou d'échec d'une instance spécifique, elle contient également plusieurs statistiques telles que le temps d'exécution de ce calcul, le nombre d'octets analysés par le calcul et le nombre d'octets que le calcul a écrits dans sa table de destination. Vous pouvez utiliser ces statistiques pour affiner votre requête, planifier la configuration ou suivre les dépenses liées à vos requêtes planifiées. Un aspect à noter est le temps d'exécution d'une instance. Dans cet exemple, le calcul planifié est configuré pour être exécuté toutes les 5 minutes. Le temps d'exécution déterminera le délai dans lequel le précalcul sera disponible, ce qui définira également le décalage dans votre tableau de bord lorsque vous utilisez les données précalculées dans vos tableaux de bord. En outre, si ce délai est constamment supérieur à l'intervalle d'actualisation, par exemple, si le temps d'exécution est supérieur à 5 minutes pour un calcul configuré pour être actualisé toutes les 5 minutes, il est important de régler votre calcul pour qu'il s'exécute plus rapidement afin d'éviter tout retard supplémentaire dans vos tableaux de bord.

Agrégation à partir d'une table dérivée

Maintenant que vous avez configuré les requêtes planifiées et que les agrégats sont précalculés et matérialisés dans un autre flux temporel pour la LiveAnalytics table spécifiée dans la configuration cible du calcul planifié, vous pouvez utiliser les données de cette table pour écrire SQL des requêtes destinées à alimenter vos tableaux de bord. Vous trouverez ci-dessous un équivalent de la requête qui utilise les pré-agrégats matérialisés pour générer l'agrégat du nombre de points de données par minute pour 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
```



La figure précédente représente l'agrégat calculé à partir du tableau des agrégats. En comparant ce panneau avec le panneau calculé à partir des données source brutes, vous remarquerez qu'ils correspondent exactement, bien que ces agrégats soient retardés de quelques minutes, en fonction de l'intervalle d'actualisation que vous avez configuré pour le calcul planifié et du temps d'exécution de celui-ci.

Cette requête sur les données précalculées analyse les données de plusieurs ordres de grandeur en moins par rapport aux agrégats calculés sur les données sources brutes. En fonction de la granularité des agrégations, cette réduction peut facilement se traduire par une réduction de 100 fois des coûts et de la latence des requêtes. L'exécution de ce calcul planifié entraîne un coût. Toutefois, en fonction de la fréquence à laquelle ces tableaux de bord sont actualisés et du nombre d'utilisateurs qui les chargent simultanément, vous finissez par réduire considérablement vos coûts globaux en utilisant ces précalculs. Et cela s'ajoute à des temps de chargement 10 à 100 fois plus rapides pour les tableaux de bord.

Agrégation combinant des tables source et dérivées

Les tableaux de bord créés à l'aide des tables dérivées peuvent présenter un décalage. Si le scénario de votre application nécessite que les tableaux de bord contiennent les données les plus récentes, vous pouvez utiliser la puissance et la flexibilité du SQL support de Timestream for pour LiveAnalytics combiner les dernières données de la table source avec les agrégats historiques de la table dérivée afin de former une vue fusionnée. Cette vue fusionnée utilise la sémantique d'SQLunion des plages temporelles non chevauchantes de la source et de la table dérivée. Dans l'exemple ci-dessous, nous

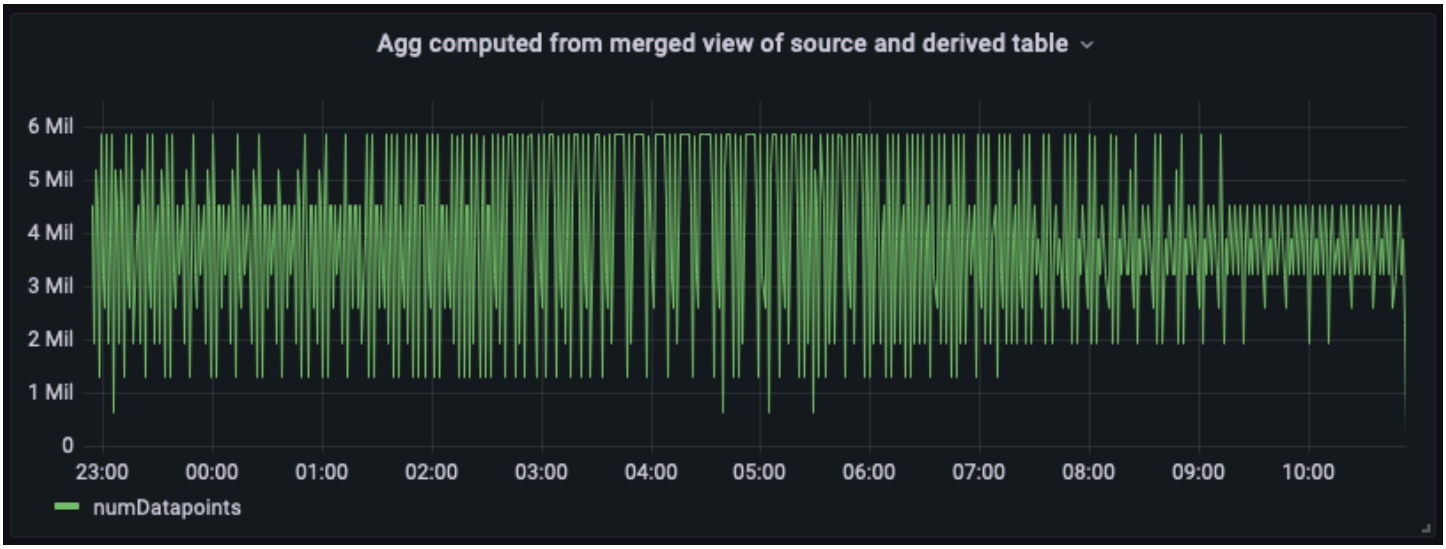
utilisons le terme « dérivé ». » per_minute_aggs_pt5m » table dérivée. Comme le calcul planifié pour cette table dérivée est actualisé toutes les 5 minutes (conformément à la spécification de l'expression de planification), la requête ci-dessous utilise les 15 dernières minutes de données de la table source et toutes les données datant de plus de 15 minutes de la table dérivée, puis réunit les résultats pour créer la vue fusionnée qui offre le meilleur des deux mondes : économie et faible latence en lisant les anciens agrégats précalculés de la table dérivée et la fraîcheur des agrégats provenant de la source pour optimiser vos cas d'utilisation de l'analyse en temps réel.

Notez que cette approche d'union aura une latence de requête légèrement plus élevée que celle consistant à interroger uniquement la table dérivée et qu'elle permettra également d'analyser un peu plus de données, car elle agrège les données brutes en temps réel pour remplir l'intervalle de temps le plus récent. Cependant, cette vue fusionnée restera nettement plus rapide et moins coûteuse que l'agrégation à la volée à partir de la table source, en particulier pour les tableaux de bord affichant des jours ou des semaines de données. Vous pouvez ajuster les plages temporelles de cet exemple en fonction des besoins d'actualisation et de tolérance aux délais de votre application.

```
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
```

Vous trouverez ci-dessous le panneau du tableau de bord avec cette vue fusionnée unifiée. Comme vous pouvez le constater, le tableau de bord est presque identique à la vue calculée à partir de la table dérivée, sauf qu'il aura le plus d' up-to-dateagrégats à l'extrémité droite.



Agrégation à partir de calculs planifiés fréquemment actualisés

En fonction de la fréquence de chargement de vos tableaux de bord et de la latence que vous souhaitez pour votre tableau de bord, il existe une autre approche pour obtenir des résultats plus récents dans votre tableau de bord : faire en sorte que le calcul planifié actualise les agrégats plus fréquemment. Par exemple, vous trouverez ci-dessous la configuration du même calcul planifié, sauf qu'il est actualisé une fois par minute (notez le calendrier express cron (0/1 * * * ? *). Avec cette configuration, la table dérivée per_minute_aggs_pt1m aura des agrégats beaucoup plus récents par rapport au scénario dans lequel le calcul spécifiait un programme d'actualisation toutes les 5 minutes.

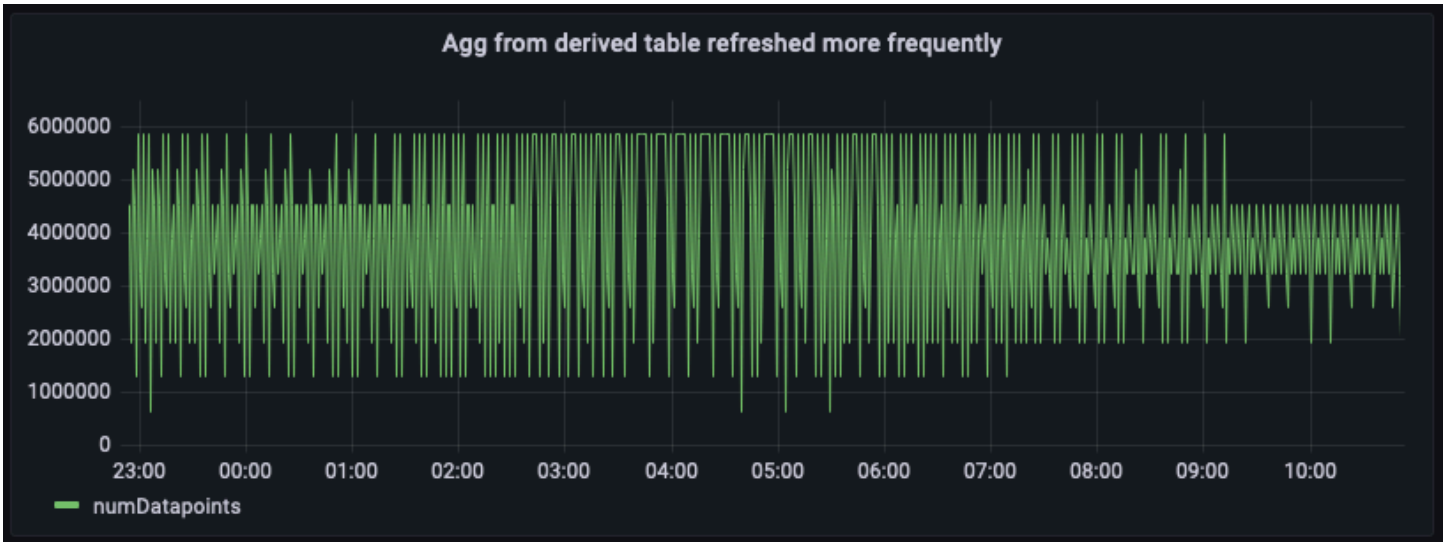
```
{
    "Name": "MultiPT1mPerMinutePerRegionMeasureCount",
    "QueryString": "SELECT region, bin(time, 1m) as minute, SUM(CASE WHEN measure_name
 = 'metrics' THEN 20 ELSE 5 END) as numDataPoints FROM raw_data.devops WHERE time
 BETWEEN @scheduled_runtime - 10m AND @scheduled_runtime + 1m GROUP BY 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
```

Comme la table dérivée contient des agrégats plus récents, vous pouvez désormais interroger directement la table dérivée per_minute_aggs_pt1m pour obtenir des agrégats plus récents, comme le montrent la requête précédente et l'instantané du tableau de bord ci-dessous.



Notez que l'actualisation du calcul planifié à un rythme plus rapide (disons 1 minute au lieu de 5 minutes) augmentera les coûts de maintenance du calcul planifié. Le message de notification pour l'exécution de chaque calcul fournit des statistiques sur la quantité de données scannée et la quantité écrite dans la table dérivée. De même, si vous utilisez la vue fusionnée pour unir la table dérivée, vous interrogez les coûts sur la vue fusionnée et la latence de chargement du tableau de bord sera plus élevée par rapport à une simple interrogation de la table dérivée. Par conséquent, l'approche que vous choisirez dépendra de la fréquence d'actualisation de vos tableaux de bord et des coûts de maintenance des requêtes planifiées. Si des dizaines d'utilisateurs actualisent les tableaux de bord une fois par minute environ, une actualisation plus fréquente de votre table dérivée se traduira probablement par une réduction globale des coûts.

Dernier point pour chaque appareil

Votre application peut vous demander de lire la dernière mesure émise par un appareil. Il peut y avoir des cas d'utilisation plus généraux pour obtenir la dernière mesure d'un appareil avant une mesure donnéedate/time or the first measurement for a device after a given date/time. Lorsque vous avez des millions d'appareils et des années de données, cette recherche peut nécessiter l'analyse de grandes quantités de données.

Vous trouverez ci-dessous un exemple de la façon dont vous pouvez utiliser des requêtes planifiées pour optimiser la recherche du dernier point émis par un appareil. Vous pouvez également utiliser le même modèle pour optimiser la première requête si votre application en a besoin.

Rubriques

Calculé à partir de la table source

Vous trouverez ci-dessous un exemple de requête pour trouver la dernière mesure émise par les services dans un déploiement spécifique (par exemple, les serveurs d'un microservice donné dans une région donnée, une cellule, un silo et une zone de disponibilité). Dans l'exemple d'application, cette requête renverra la dernière mesure pour des centaines de serveurs. Notez également que cette requête possède un prédicat temporel illimité et recherche toutes les données antérieures à un horodatage donné.

> **Note**
>
> Pour plus d'informations sur les `max_by` fonctions `max` et, consultezFonctions d’agrégation.

```
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
```

Table dérivée à précalculer à la granularité quotidienne

Vous pouvez convertir le cas d'utilisation précédent en un calcul planifié. Si les exigences de votre application sont telles que vous devrez peut-être obtenir ces valeurs pour l'ensemble de votre flotte dans plusieurs régions, cellules, silos, zones de disponibilité et microservices, vous pouvez utiliser un seul calcul de planification pour précalculer les valeurs pour l'ensemble de votre flotte. C'est la puissance des requêtes planifiées sans serveur LiveAnalytics de Timestream for, qui permet à ces requêtes de s'adapter aux exigences de dimensionnement de votre application.

Vous trouverez ci-dessous une requête permettant de précalculer le dernier point sur tous les serveurs pour un jour donné. Notez que la requête ne comporte qu'un prédicat temporel et non un prédicat sur les dimensions. Le prédicat temporel limite la requête au dernier jour à partir du moment où le calcul est déclenché en fonction de l'expression de planification spécifiée.

```
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
```

Vous trouverez ci-dessous une configuration pour le calcul planifié à l'aide de la requête précédente qui exécute cette requête à 01h00 UTC tous les jours pour calculer l'agrégat du jour précédent. L'expression de planification cron (0 1 * * ? *) contrôle ce comportement et s'exécute une heure après la fin de la journée pour prendre en compte les données arrivant jusqu'à un jour de retard.

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

Calculé à partir d'une table dérivée

Une fois que vous avez défini la table dérivée à l'aide de la configuration précédente et qu'au moins une instance de la requête planifiée a matérialisé les données dans la table dérivée, vous pouvez désormais interroger la table dérivée pour obtenir la dernière mesure. Vous trouverez ci-dessous un exemple de requête sur la table dérivée.

```
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
```

Combinaison d'une table source et d'une table dérivée

Comme dans l'exemple précédent, les données de la table dérivée ne comporteront pas les écritures les plus récentes. Par conséquent, vous pouvez à nouveau utiliser un modèle similaire à celui utilisé

précédemment pour fusionner les données de la table dérivée pour les données plus anciennes et utiliser les données source pour le conseil restant. Vous trouverez ci-dessous un exemple d'une telle requête utilisant une UNION approche similaire. Étant donné que l'exigence de l'application est de trouver la dernière mesure avant une période donnée, et que cette heure de début peut être passée, la façon dont vous écrivez cette requête consiste à utiliser l'heure indiquée, à utiliser les données source jusqu'à un jour à partir de l'heure spécifiée, puis à utiliser la table dérivée sur les anciennes données. Comme vous pouvez le voir dans l'exemple de requête ci-dessous, le prédicat temporel des données source est limité. Cela garantit un traitement efficace sur la table source qui contient un volume de données nettement plus élevé, puis le prédicat temporel illimité se trouve sur la table dérivée.

```
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
```

Le précédent n'est qu'une illustration de la manière dont vous pouvez structurer les tables dérivées. Si vous disposez de plusieurs années de données, vous pouvez utiliser plusieurs niveaux d'agrégation. Par exemple, vous pouvez avoir des agrégats mensuels en plus des agrégats quotidiens, et vous pouvez avoir des agrégats horaires avant les agrégats quotidiens. Vous pouvez donc fusionner le plus récent pour renseigner la dernière heure, l'horaire pour le dernier jour, le quotidien pour le mois dernier et le mensuel pour remplir le plus ancien. Le nombre de niveaux que vous configurez par rapport au calendrier d'actualisation dépendra de vos besoins, notamment de la fréquence à laquelle ces requêtes posent problème et du nombre d'utilisateurs qui les émettent simultanément.

Valeurs de dimension uniques

Dans certains cas d'utilisation, vous pouvez utiliser des tableaux de bord dans lesquels vous souhaitez utiliser les valeurs uniques des dimensions comme variables afin d'analyser les métriques correspondant à une tranche de données spécifique. L'instantané ci-dessous est un exemple dans lequel le tableau de bord préremplit les valeurs uniques de plusieurs dimensions telles que la région, la cellule, le silo, le microservice et la zone de disponibilité. Nous montrons ici un exemple de la façon dont vous pouvez utiliser des requêtes planifiées pour accélérer considérablement le calcul de ces valeurs distinctes de ces variables à partir des indicateurs que vous suivez.

Rubriques

- Sur les données brutes
- Pré-calculez des valeurs de dimension uniques
- Calcul des variables à partir d'une table dérivée

Sur les données brutes

Vous pouvez l'utiliser SELECT DISTINCT pour calculer les valeurs distinctes observées à partir de vos données. Par exemple, si vous souhaitez obtenir les valeurs distinctes de la région, vous pouvez utiliser la requête de ce formulaire.

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

```
ORDER BY 1
```

Vous suivez peut-être des millions d'appareils et des milliards de séries chronologiques. Cependant, dans la plupart des cas, ces variables intéressantes concernent des dimensions de cardinalité inférieures, pour lesquelles vous avez quelques valeurs, voire des dizaines. Le calcul DISTINCT à partir de données brutes peut nécessiter la numérisation de gros volumes de données.

Pré-calculez des valeurs de dimension uniques

Vous souhaitez que ces variables se chargent rapidement afin que vos tableaux de bord soient interactifs. De plus, ces variables sont souvent calculées pour chaque charge du tableau de bord. Vous souhaitez donc qu'elles soient également rentables. Vous pouvez optimiser la recherche de ces variables à l'aide de requêtes planifiées et en les matérialisant dans une table dérivée.

Tout d'abord, vous devez identifier les dimensions pour lesquelles vous devez calculer les DISTINCT valeurs ou les colonnes que vous utiliserez dans les prédicats lors du calcul de la DISTINCT valeur.

Dans cet exemple, vous pouvez constater que le tableau de bord fournit des valeurs distinctes pour les dimensions region, cell, silo, availability_zone et microservice. Vous pouvez donc utiliser la requête ci-dessous pour précalculer ces valeurs uniques.

```
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
```

Il y a quelques points importants à noter ici.

- Vous pouvez utiliser un calcul planifié pour précalculer les valeurs de nombreuses requêtes différentes. Par exemple, vous utilisez la requête précédente pour précalculer les valeurs de cinq variables différentes. Vous n'en avez donc pas besoin pour chaque variable. Vous pouvez utiliser ce même modèle pour identifier les calculs partagés entre plusieurs panneaux afin d'optimiser le nombre de requêtes planifiées que vous devez gérer.
- Les valeurs uniques des dimensions ne sont pas intrinsèquement des données de séries chronologiques. Vous convertissez donc cela en série chronologique à l'aide du @scheduled_runtime. En associant ces données au paramètre @scheduled_runtime, vous pouvez également suivre les valeurs uniques apparues à un moment donné, créant ainsi des séries chronologiques à partir de celles-ci.

- Dans l'exemple précédent, vous verrez une valeur métrique faire l'objet d'un suivi. Cet exemple utilise COUNT (*). Vous pouvez calculer d'autres agrégats significatifs si vous souhaitez les suivre pour vos tableaux de bord.

Vous trouverez ci-dessous une configuration pour un calcul planifié à l'aide de la requête précédente. Dans cet exemple, il est configuré pour être actualisé toutes les 15 minutes à l'aide de l'expression de planification 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": "******"
}
```

Calcul des variables à partir d'une table dérivée

Une fois que le calcul planifié a pré-matérialisé les valeurs uniques de la table dérivée hc_unique_dimensions_pt15m, vous pouvez utiliser la table dérivée pour calculer efficacement les valeurs uniques des dimensions. Vous trouverez ci-dessous des exemples de requêtes expliquant comment calculer les valeurs uniques et comment utiliser d'autres variables comme prédicats dans ces requêtes de valeurs uniques.

Région

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

Cellule

```
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
```

Microservices

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

Zone de disponibilité

```
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
```

Gestion des données arrivées en retard

Dans certains scénarios, les données peuvent arriver très tard, par exemple, l'heure à laquelle les données ont été ingérées dans Timestream pour LiveAnalytics est considérablement retardée par rapport à l'horodatage associé aux lignes ingérées. Dans les exemples précédents, vous avez vu comment utiliser les plages de temps définies par le paramètre @scheduled_runtime pour tenir compte de certaines données arrivées en retard. Toutefois, si vous avez des cas d'utilisation où les données peuvent être retardées de plusieurs heures ou de plusieurs jours, vous aurez peut-être besoin d'un modèle différent pour vous assurer que vos précalculs dans la table dérivée sont

correctement mis à jour afin de refléter ces données arrivées tardivement. Pour des informations générales sur les données arrivées tardivement, voir. Écrire des données (insertions et insertions)

Dans ce qui suit, vous verrez deux manières différentes de traiter ces données arrivées tardivement.

- Si vous avez des retards prévisibles dans l'arrivée de vos données, vous pouvez utiliser un autre calcul planifié « de rattrapage » pour mettre à jour vos agrégats en fonction des données arrivées en retard.
- Si vous êtes confronté à des retards imprévisibles ou à des données d'arrivée tardive occasionnelles, vous pouvez utiliser des exécutions manuelles pour mettre à jour les tables dérivées.

Cette discussion couvre les scénarios d'arrivée tardive des données. Toutefois, les mêmes principes s'appliquent aux corrections de données, lorsque vous avez modifié les données de votre table source et que vous souhaitez mettre à jour les agrégats dans vos tables dérivées.

Rubriques

- Requêtes de rattrapage planifiées
- Exécutions manuelles pour les données arrivant en retard et imprévisibles

Requêtes de rattrapage planifiées

Interrogez en agrégeant les données arrivées à temps

Vous trouverez ci-dessous un schéma vous expliquant comment vous pouvez utiliser une méthode automatisée pour mettre à jour vos agrégats si vous avez des retards prévisibles dans l'arrivée de vos données. Prenons l'un des exemples précédents de calcul planifié sur des données en temps réel ci-dessous. Ce calcul planifié actualise la table dérivée toutes les 30 minutes et prend déjà en compte les données retardées jusqu'à une heure.

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

Requête de rattrapage mettant à jour les agrégats pour les données arrivées en retard

Maintenant, si vous considérez le cas où vos données peuvent être retardées d'environ 12 heures. Vous trouverez ci-dessous une variante de la même requête. Cependant, la différence réside dans le fait qu'il calcule les agrégats sur des données retardées jusqu'à 12 heures par rapport au moment où le calcul planifié est déclenché. Par exemple, vous voyez la requête dans l'exemple ci-dessous, la plage de temps ciblée par cette requête est comprise entre 2 h et 14 h avant le déclenchement de la requête. De plus, si vous remarquez l'expression de planification cron (0, 0,12 * * ? *), il déclenchera le calcul à 00h00 UTC et 12h00 UTC tous les jours. Par conséquent, lorsque la requête est déclenchée le 2021-12-01 00:00:00, elle met à jour les agrégats entre le 2021-11-30

10:00:00 et le 2021-11-30 22:00:00. Les requêtes planifiées utilisent une sémantique ascendante similaire à celle de Timestream for, où cette requête LiveAnalytics de rattrapage met à jour les valeurs agrégées avec des valeurs plus récentes si des données arrivent en retard dans la fenêtre ou si de nouveaux agrégats sont trouvés (par exemple, un nouveau regroupement apparaît dans cet agrégat qui n'était pas présent lorsque le calcul planifié d'origine a été déclenché), puis le nouvel agrégat sera inséré dans la table dérivée. De même, lorsque la prochaine instance est déclenchée le 2021-12-01 12:00:00, cette instance mettra à jour les agrégats compris entre le 2021-11-30 22:00:00 et le 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": "******"
}
```

L'exemple précédent est une illustration en supposant que votre arrivée tardive est limitée à 12 heures et que vous pouvez mettre à jour la table dérivée une fois toutes les 12 heures pour les données arrivant après la fenêtre en temps réel. Vous pouvez adapter ce modèle pour mettre à jour votre table dérivée une fois par heure afin qu'elle reflète plus rapidement les données arrivées en retard. De même, vous pouvez adapter la plage horaire pour qu'elle soit antérieure à 12 heures, par exemple un jour, voire une semaine ou plus, afin de gérer des données prévisibles en retard.

Exécutions manuelles pour les données arrivant en retard et imprévisibles

Il peut arriver que des données arrivent en retard de façon imprévisible ou que vous ayez apporté des modifications aux données sources et mis à jour certaines valeurs après coup. Dans tous ces cas, vous pouvez déclencher manuellement des requêtes planifiées pour mettre à jour la table dérivée. Vous trouverez ci-dessous un exemple de la manière dont vous pouvez y parvenir.

Supposons que vous ayez le cas d'utilisation où le calcul est écrit dans la table dérivée dp_per_timeseries_per_hr. Vos données de base dans le tableau devops ont été mises à jour dans la plage horaire 2021-11-30 23:00:00 - 2021-12-01 00:00:00. Deux requêtes planifiées différentes peuvent être utilisées pour mettre à jour cette table dérivée : Multi PT3 0 mPerHr PerTimeseries DPCount et MultiPT12HPerHrPerTimeseriesDPCountCatchUp. Chaque calcul planifié que vous créez dans Timestream pour LiveAnalytics possède un caractère unique ARN que vous obtenez lorsque vous créez le calcul ou lorsque vous effectuez une opération de liste. Vous pouvez utiliser le ARN pour le calcul et une valeur pour le paramètre @scheduled_runtime pris par la requête pour effectuer cette opération.

Supposons que le calcul pour Multi PT3 0 mPerHr PerTimeseries DPCount comporte un ARN arn_1 et que vous souhaitez utiliser ce calcul pour mettre à jour la table dérivée. Étant donné que le calcul planifié précédent met à jour les agrégats 1 h avant et 1 heure après la valeur @scheduled_runtime, vous pouvez couvrir la plage horaire de la mise à jour (2021-11-30 23:00:00 - 2021-12-01 00:00:00) en utilisant une valeur de 2021-12-01 00:00:00 pour le paramètre @scheduled_runtime. Vous pouvez utiliser le ExecuteScheduledQuery API pour transmettre ce calcul et la ARN valeur du paramètre

temporel en secondes d'époque (inUTC) pour y parvenir. Vous trouverez ci-dessous un exemple d'utilisation du AWS CLI et vous pouvez suivre le même schéma en utilisant l'un des logiciels SDKs pris en charge par Timestream pour. LiveAnalytics

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

Dans l'exemple précédent, le profil est le AWS profil qui possède les privilèges appropriés pour effectuer cet API appel et 1638316800 correspond à la seconde époque du 2021-12-01 00:00:00. Ce déclencheur manuel se comporte presque comme le déclencheur automatique en supposant que le système ait déclenché cette invocation à la période souhaitée.

Si vous avez eu une mise à jour sur une période plus longue, disons que les données de base ont été mises à jour pour le 2021-11-30 23:00:00 - 2021-12-01 11:00:00, alors vous pouvez déclencher les requêtes précédentes plusieurs fois pour couvrir toute cette période. Par exemple, vous pouvez effectuer six exécutions différentes comme suit.

```
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
```

Les six commandes précédentes correspondent au calcul planifié invoqué le 2021-12-01 00:00:00, 2021-12-01 02:00:00, 2021-12-01 04:0:00, 2021-12-01 06:00:00, 2021-12-01 08:00:00, 2021-12-01 08:00:00, et le 2021-12-01 10:00 :

Vous pouvez également utiliser le calcul Multi PT12HPerHrPerTimeseriesDPCountCatchUp déclenché le 2021-12-01 13:00:00 pour une exécution afin de mettre à jour les agrégats pour

l'ensemble de la plage horaire de 12 heures. Par exemple, si arn_2 correspond à ce calcul, vous pouvez exécuter la commande suivante à partir de. ARN CLI

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

Il convient de noter que pour un déclenchement manuel, vous pouvez utiliser un horodatage pour le paramètre d'heure d'appel qui n'a pas besoin d'être aligné sur les horodatages de ce déclencheur automatique. Par exemple, dans l'exemple précédent, vous avez déclenché le calcul à l'heure 2021-12-01 13:00:00 même si le calendrier automatique ne se déclenche qu'aux horodatages 2021-12-01 10:00:00, 2021-12-01 12:00:00, et 2021-12-02 00:00:00. Timestream for vous LiveAnalytics offre la flexibilité de le déclencher avec les valeurs appropriées en fonction de vos opérations manuelles.

Voici quelques considérations importantes à prendre en compte lors de l'utilisation du ExecuteScheduledQuery API.

- Si vous déclenchez plusieurs de ces invocations, vous devez vous assurer qu'elles ne génèrent pas de chevauchement de plages temporelles. Par exemple, dans les exemples précédents, il y avait six invocations. Chaque invocation couvre une période de 2 heures, c'est pourquoi les horodatages d'invocation ont été étalés de deux heures chacun pour éviter tout chevauchement dans les mises à jour. Cela garantit que les données de la table dérivée se retrouvent dans un état correspondant aux agrégats de la table source. Si vous ne pouvez pas garantir que les plages de temps ne se chevauchent pas, assurez-vous que ces exécutions sont déclenchées séquentiellement les unes après les autres. Si vous déclenchez simultanément plusieurs exécutions dont les plages temporelles se chevauchent, les rapports d'erreur relatifs à ces exécutions peuvent entraîner des échecs de déclenchement susceptibles de provoquer des conflits de version. Une version est attribuée aux résultats générés par un appel de requête planifié en fonction du moment où l'appel a été déclenché. Par conséquent, les lignes générées par les nouveaux appels ont des versions supérieures. Un enregistrement de version supérieure peut remplacer un enregistrement de version inférieure. Pour les requêtes planifiées déclenchées automatiquement, Timestream for gère LiveAnalytics automatiquement les plannings afin que vous ne voyiez pas ces problèmes, même si les invocations suivantes ont des plages horaires qui se chevauchent.
- comme indiqué précédemment, vous pouvez déclencher les invocations avec n'importe quelle valeur d'horodatage pour @scheduled_runtime. Il est donc de votre responsabilité de définir les valeurs de manière appropriée afin que les plages de temps appropriées soient mises à jour

dans la table dérivée correspondant aux plages où les données ont été mises à jour dans la table source.

- Vous pouvez également utiliser ces déclencheurs manuels pour les requêtes planifiées qui sont dans l'DISABLEDétat. Cela vous permet de définir des requêtes spéciales qui ne sont pas exécutées dans un calendrier automatique, car elles sont dans l'DISABLEDétat. Vous pouvez plutôt utiliser les déclencheurs manuels qu'ils contiennent pour gérer les corrections de données ou les cas d'utilisation en retard.

Compléter les précalculs historiques

Lorsque vous créez un calcul planifié, Timestream for LiveAnalytics gère les exécutions des requêtes à l'avenir, l'actualisation étant régie par l'expression de planification que vous fournissez. En fonction de la quantité de données historiques de votre table source, vous souhaiterez peut-être mettre à jour votre table dérivée avec des agrégats correspondant aux données historiques. Vous pouvez utiliser la logique précédente pour les déclencheurs manuels afin de compléter les agrégats historiques.

Par exemple, si nous considérons la table dérivée per_timeseries_lastpoint_pt1d, le calcul planifié est mis à jour une fois par jour pour le dernier jour. Si votre table source contient une année de données, vous pouvez utiliser le ARN pour ce calcul planifié et le déclencher manuellement pour tous les jours jusqu'à un an afin que toutes les requêtes historiques soient remplies dans la table dérivée. Notez que toutes les mises en garde relatives aux déclencheurs manuels s'appliquent ici. De plus, si la table dérivée est configurée de manière à ce que l'historique d'ingestion soit écrit dans la mémoire magnétique de la table dérivée, prenez connaissance des meilleures pratiques et des limites relatives aux écritures dans la mémoire magnétique.

Exemples de requêtes planifiées

Cette section contient des exemples de la manière dont vous pouvez utiliser les requêtes planifiées de Timestream pour LiveAnalytics optimiser les coûts et les temps de chargement du tableau de bord lorsque vous visualisez des statistiques à l'échelle du parc et surveillez efficacement votre parc d'appareils. Les requêtes planifiées dans Timestream pour vous LiveAnalytics permettent d'exprimer vos requêtes en utilisant toute la SQL surface de Timestream pour. LiveAnalytics Votre requête peut inclure une ou plusieurs tables sources, effectuer des agrégations ou toute autre requête autorisée par la SQL langue de Timestream for LiveAnalytics, puis stocker les résultats de la requête dans une autre table de destination dans Timestream for. LiveAnalytics

Cette section fait référence à la table cible d'une requête planifiée en tant que table dérivée.

À titre d'exemple, nous utiliserons une DevOps application dans laquelle vous surveillez un vaste parc de serveurs déployés sur plusieurs déploiements (tels que des régions, des cellules et des silos), plusieurs microservices, et vous suivez les statistiques de l'ensemble du parc à l'aide de Timestream pour. LiveAnalytics L'exemple de schéma que nous allons utiliser est décrit dans Exemple de schéma de requêtes planifiées.

Les scénarios suivants seront décrits.

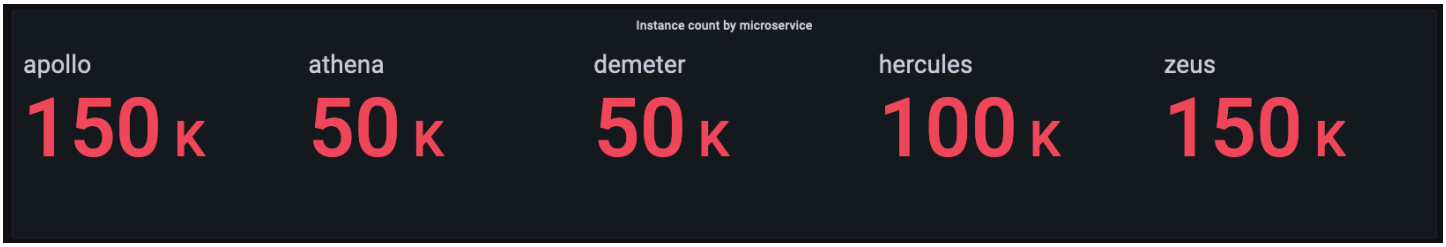
- Comment convertir un tableau de bord en traçant des statistiques agrégées à partir des données brutes que vous ingérez dans Timestream LiveAnalytics en une requête planifiée, puis comment utiliser vos agrégats précalculés pour créer un nouveau tableau de bord présentant des statistiques agrégées.
- Comment combiner des requêtes planifiées pour obtenir une vue agrégée et les données granulaires brutes, afin d'approfondir les détails. Cela vous permet de stocker et d'analyser les données brutes tout en optimisant les opérations courantes à l'échelle de votre flotte à l'aide de requêtes planifiées.
- Comment optimiser les coûts à l'aide de requêtes planifiées en identifiant les agrégats utilisés dans plusieurs tableaux de bord et en faisant en sorte que la même requête planifiée remplisse plusieurs panneaux du même tableau de bord ou de plusieurs tableaux de bord.

Rubriques

Conversion d'un tableau de bord agrégé en requête planifiée

Supposons que vous calculez les statistiques à l'échelle du parc, telles que le nombre d'hôtes dans le parc par les cinq microservices et par les six régions dans lesquelles votre service est déployé. L'instantané ci-dessous montre que 500 000 serveurs émettent des métriques, et que certaines des plus grandes régions (par exemple, us-east-1) comptent plus de 200 000 serveurs.

Le calcul de ces agrégats, qui consiste à calculer des noms d'instance distincts pour des centaines de gigaoctets de données, peut entraîner une latence des requêtes de plusieurs dizaines de secondes, en plus du coût d'analyse des données.



Requête de tableau de bord originale

L'agrégat affiché dans le panneau du tableau de bord est calculé, à partir de données brutes, à l'aide de la requête ci-dessous. La requête utilise plusieurs SQL structures, telles que des dénombrements distincts et de multiples fonctions d'agrégation.

```
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
)
```

Conversion en requête planifiée

La requête précédente peut être convertie en requête planifiée comme suit. Vous devez d'abord calculer les noms d'hôtes distincts au sein d'un déploiement donné dans une région, une cellule, un silo, une zone de disponibilité et un microservice. Vous additionnez ensuite les hôtes pour calculer

le nombre d'hôtes par heure et par microservice. En utilisant le `@scheduled_runtime` paramètre pris en charge par les requêtes planifiées, vous pouvez le recalculer pour l'heure écoulée lorsque la requête est invoquée. La `WHERE` clause `bin(@scheduled_runtime, 1h)` in the de la requête interne garantit que même si la requête est planifiée au milieu de l'heure, vous obtenez toujours les données pendant toute l'heure.

Même si la requête calcule des agrégats horaires, comme vous le verrez dans la configuration de calcul planifiée, elle est configurée pour être actualisée toutes les demi-heures afin que vous puissiez obtenir les mises à jour de votre table dérivée plus rapidement. Vous pouvez ajuster cela en fonction de vos besoins en fraîcheur, par exemple en recalculant les agrégats toutes les 15 minutes ou en les recalculant aux limites horaires.

```
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
```

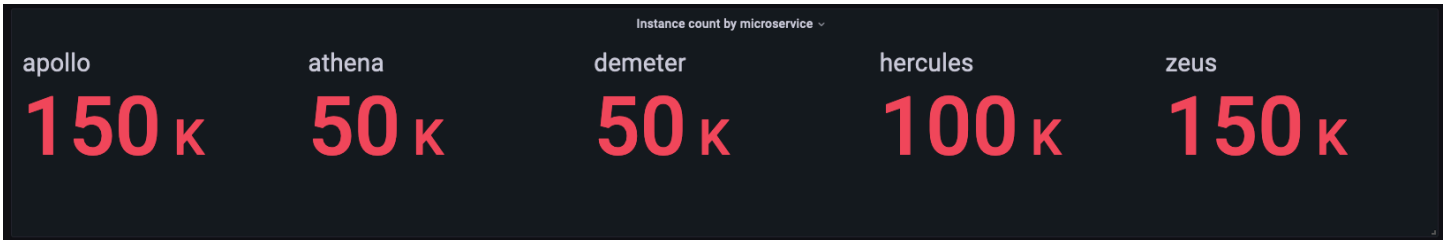
```
{
    "Name": "MultiPT30mHostCountMicroservicePerHr",
    "QueryString": "SELECT microservice_name, hour, SUM(num_instances) AS num_instances
    FROM (        SELECT microservice_name, bin(time, 1h) AS hour, COUNT(DISTINCT
 instance_name) as num_instances        FROM raw_data.devops        WHERE 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": "******"
}
```

Utilisation des résultats précalculés dans un nouveau tableau de bord

Vous allez maintenant voir comment créer votre tableau de bord de vue agrégée à l'aide de la table dérivée de la requête planifiée que vous avez créée. À partir de l'instantané du tableau de bord, vous pourrez également vérifier que les agrégats calculés à partir de la table dérivée et de la table de base correspondent également. Une fois que vous aurez créé les tableaux de bord à l'aide des tables dérivées, vous remarquerez que le temps de chargement est nettement plus rapide et que les coûts d'utilisation des tables dérivées sont réduits par rapport au calcul de ces agrégats à partir des données brutes. Vous trouverez ci-dessous un instantané du tableau de bord utilisant des données précalculées, ainsi que la requête utilisée pour afficher ce panneau à l'aide de données précalculées

stockées dans la table « dérivée ». » host_count_pt1h ». Notez que la structure de la requête est très similaire à celle utilisée dans le tableau de bord sur les données brutes, sauf qu'elle utilise la table dérivée qui calcule déjà les nombres distincts que cette requête agrège.




```
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
)
```

Utilisation de requêtes planifiées et de données brutes pour les analyses approfondies

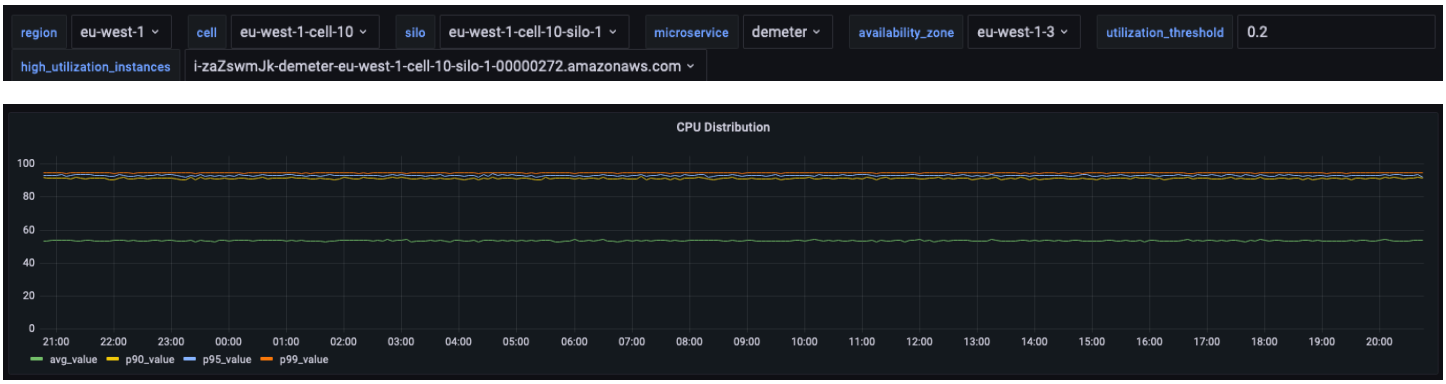
Vous pouvez utiliser les statistiques agrégées de votre flotte pour identifier les domaines nécessitant une analyse approfondie, puis utiliser les données brutes pour analyser les données granulaires afin d'obtenir des informations plus approfondies.

Dans cet exemple, vous verrez comment vous pouvez utiliser un tableau de bord agrégé pour identifier tout déploiement (un déploiement concerne un microservice donné dans une région, une cellule, un silo et une zone de disponibilité donnés) dont le taux d'CPUutilisation semble supérieur à celui des autres déploiements. Vous pouvez ensuite effectuer une analyse approfondie pour mieux

comprendre à l'aide des données brutes. Étant donné que ces analyses peuvent être peu fréquentes et n'accéder qu'aux données pertinentes pour le déploiement, vous pouvez utiliser les données brutes pour cette analyse et vous n'avez pas besoin d'utiliser de requêtes planifiées.

Analyse détaillée par déploiement

Le tableau de bord ci-dessous permet d'accéder à des statistiques plus détaillées et plus détaillées au niveau du serveur dans le cadre d'un déploiement donné. Pour vous aider à analyser les différentes parties de votre flotte, ce tableau de bord utilise des variables telles que la région, la cellule, le silo, le microservice et la zone de disponibilité. Il affiche ensuite des statistiques agrégées pour ce déploiement.



Dans la requête ci-dessous, vous pouvez voir que les valeurs choisies dans le menu déroulant des variables sont utilisées comme prédicats dans la `WHERE` clause de la requête, ce qui vous permet de vous concentrer uniquement sur les données du déploiement. Ensuite, le panneau trace les CPU métriques agrégées pour les instances de ce déploiement. Vous pouvez utiliser les données brutes pour effectuer cette analyse détaillée avec une latence de requête interactive afin d'obtenir des informations plus approfondies.

```
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
```

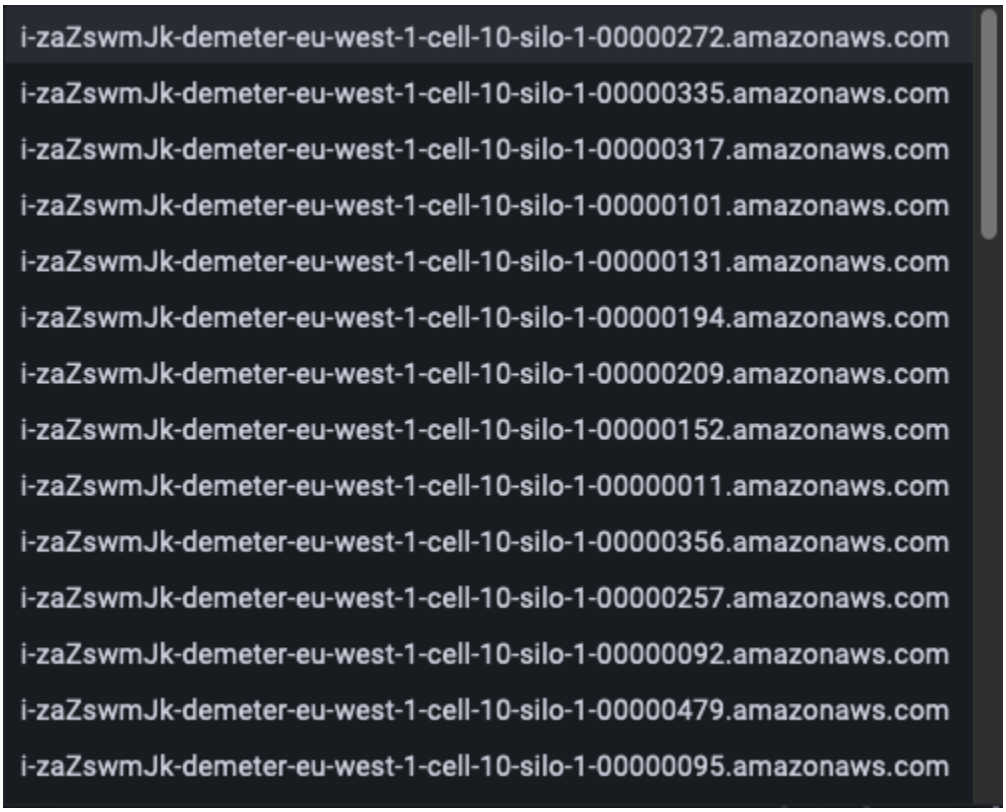
Statistiques au niveau de l'instance

Ce tableau de bord calcule également une autre variable qui répertorie également les serveurs/instances les plus CPU utilisés, triés par ordre décroissant d'utilisation. La requête utilisée pour calculer cette variable est affichée ci-dessous.

```
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
```

Dans la requête précédente, la variable est recalculée dynamiquement en fonction des valeurs choisies pour les autres variables. Une fois que la variable est renseignée pour un déploiement, vous pouvez sélectionner des instances individuelles dans la liste pour mieux visualiser les métriques de

cette instance. Vous pouvez sélectionner les différentes instances dans le menu déroulant des noms d'instance, comme le montre l'instantané ci-dessous.



Les panneaux précédents présentent les statistiques de l'instance sélectionnée et les requêtes utilisées pour récupérer ces statistiques sont présentées ci-dessous.

```
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
```

Optimisation des coûts en partageant les requêtes planifiées entre les tableaux de bord

Dans cet exemple, nous allons voir un scénario dans lequel plusieurs panneaux de tableau de bord affichent des variations d'informations similaires (recherche d'CPUhôtes élevés et fraction de flotte CPU très utilisée) et comment vous pouvez utiliser la même requête planifiée pour précalculer les résultats qui sont ensuite utilisés pour remplir plusieurs panneaux. Cette réutilisation optimise encore davantage vos coûts, car au lieu d'utiliser différentes requêtes planifiées, une pour chaque panneau, vous n'utilisez que le propriétaire.

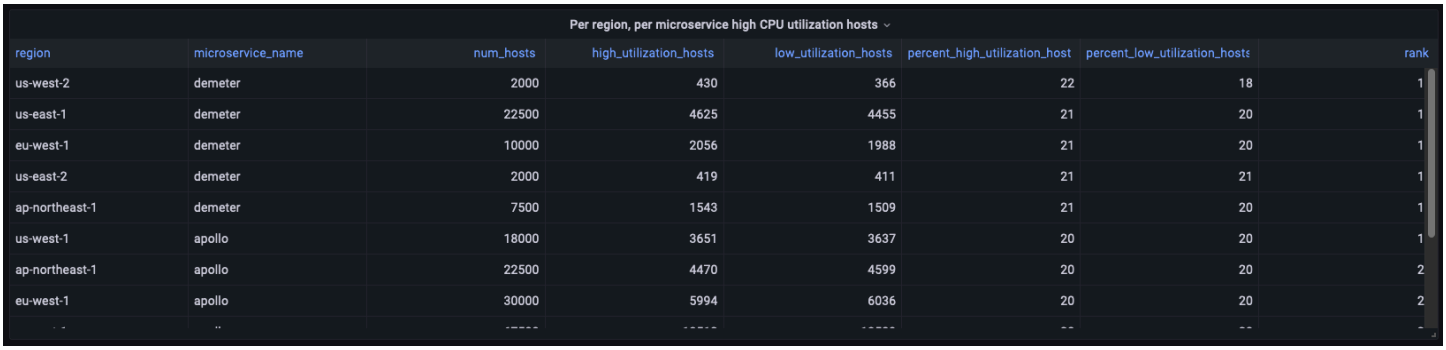
Panneaux de tableau de bord avec données brutes

CPUutilisation par région et par microservice

Le premier panneau calcule les instances dont l'utilisation moyenne est un seuil inférieur ou supérieur à CPU l'utilisation ci-dessus pour un déploiement donné au sein d'une région, d'une cellule, d'un silo, d'une zone de disponibilité et d'un microservice. CPU Il trie ensuite la région et le microservice présentant le pourcentage le plus élevé d'hôtes présentant un taux d'utilisation élevé. Il permet

d'identifier la température de fonctionnement des serveurs d'un déploiement spécifique, puis d'effectuer une analyse approfondie pour mieux comprendre les problèmes.

La requête pour le panneau démontre la flexibilité du SQL support de Timestream for pour LiveAnalytics effectuer des tâches analytiques complexes avec des expressions de table, des fonctions de fenêtre, des jointures, etc. communes.



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

Requête :

```
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
```

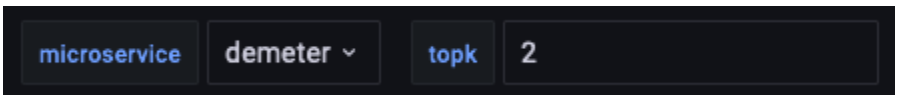
```
      AND m.microservice_name = i.microservice_name
), per_deployment_high AS (
SELECT region, microservice_name, COUNT(*) AS num_hosts, SUM(high_utilization) AS
 high_utilization_hosts, SUM(low_utilization) AS low_utilization_hosts,
    ROUND(SUM(high_utilization) * 100.0 / COUNT(*), 0) AS
 percent_high_utilization_hosts,
    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
```

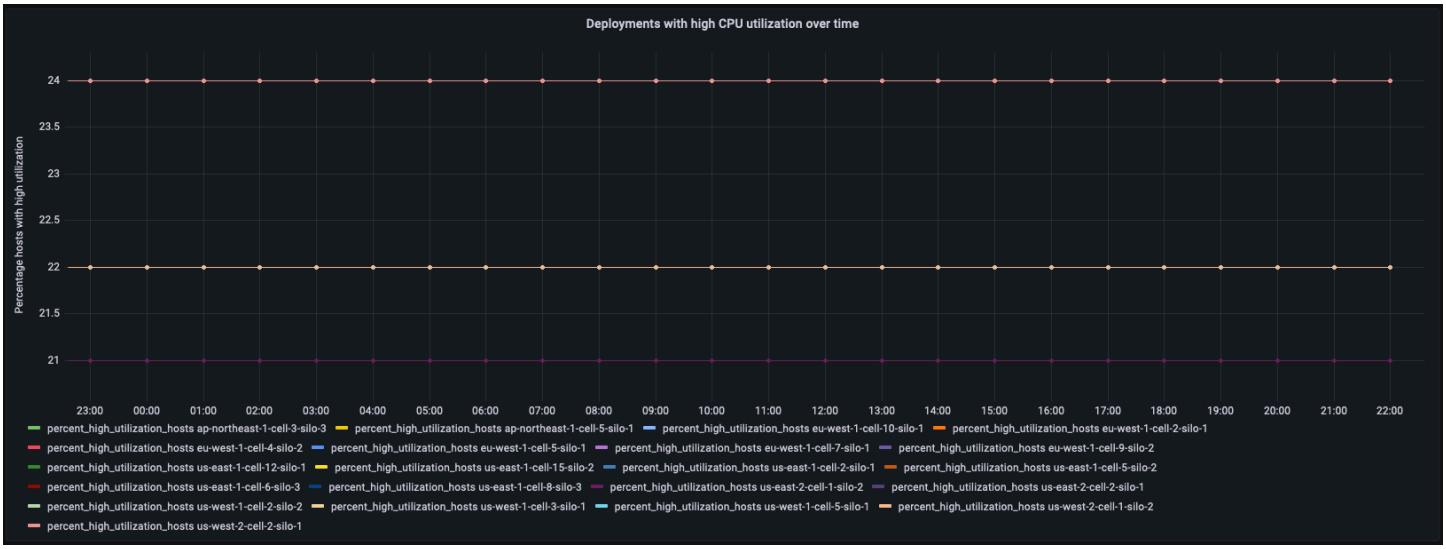
Explorez un microservice pour trouver les points chauds

Le tableau de bord suivant vous permet d'étudier plus en détail l'un des microservices afin de déterminer la région, la cellule et le silo spécifiques dans lesquels ce microservice gère quelle fraction de son parc est le plus utilisé. CPU Par exemple, dans le tableau de bord à l'échelle de la flotte, vous avez vu le microservice apparaître dans les premières positions du classement. Dans ce tableau de bord, vous souhaitez donc approfondir ce microservice.

Ce tableau de bord utilise une variable pour sélectionner le microservice à explorer, et les valeurs de la variable sont renseignées à l'aide des valeurs uniques de la dimension. Une fois que vous avez sélectionné le microservice, le reste du tableau de bord est actualisé.

Comme vous le voyez ci-dessous, le premier panneau trace le pourcentage d'hôtes dans un déploiement (une région, une cellule et un silo pour un microservice) au fil du temps, ainsi que la requête correspondante qui est utilisée pour tracer le tableau de bord. Ce diagramme identifie lui-même un déploiement spécifique présentant un pourcentage plus élevé d'hôtes présentant un taux élevéCPU.





Requête :

```
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
```

Conversion en une seule requête planifiée permettant la réutilisation

Il est important de noter qu'un calcul similaire est effectué sur les différents panneaux des deux tableaux de bord. Vous pouvez définir une requête planifiée distincte pour chaque panneau. Vous découvrirez ici comment optimiser davantage vos coûts en définissant une requête planifiée dont les résultats peuvent être utilisés pour afficher les trois panneaux.

Voici la requête qui capture les agrégats calculés et utilisés pour les différents panneaux. Vous observerez plusieurs aspects importants lors de la définition de cette requête planifiée.

- La flexibilité et la puissance de la SQL surface prises en charge par les requêtes planifiées, où vous pouvez utiliser des expressions tabulaires courantes, des jointures, des exposés de cas, etc.
- Vous pouvez utiliser une requête planifiée pour calculer les statistiques avec une granularité plus fine que celle dont un tableau de bord spécifique pourrait avoir besoin, et pour toutes les valeurs qu'un tableau de bord peut utiliser pour différentes variables. Par exemple, vous verrez que les agrégats sont calculés pour une région, une cellule, un silo et un microservice. Vous pouvez donc les combiner pour créer des agrégats au niveau de la région, ou de la région, et au niveau des

microservices. De même, la même requête calcule les agrégats pour toutes les régions, cellules, silos et microservices. Il vous permet d'appliquer des filtres sur ces colonnes afin d'obtenir les agrégats d'un sous-ensemble de valeurs. Par exemple, vous pouvez calculer les agrégats pour n'importe quelle région, par exemple us-east-1, ou pour n'importe quel microservice, par exemple demeter ou explorer un déploiement spécifique au sein d'une région, d'une cellule, d'un silo ou d'un microservice. Cette approche optimise davantage vos coûts de maintenance des agrégats précalculés.

```
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
```

Voici une définition de requête planifiée pour la requête précédente. L'expression de planification est configurée pour être actualisée toutes les 30 minutes et actualise les données jusqu'à une heure en arrière, en utilisant à nouveau la construction bin (@scheduled_runtime, 1h) pour obtenir les événements de l'heure complète. En fonction des exigences de fraîcheur de votre application, vous pouvez la configurer pour qu'elle s'actualise plus ou moins fréquemment. En utilisant WHERE time BETWEEN bin (@scheduled_runtime, 1h) - 1h AND bin (@scheduled_runtime, 1h) + 1h, nous pouvons garantir que même si vous actualisez toutes les 15 minutes, vous obtiendrez les données de l'heure complète pour l'heure en cours et l'heure précédente.

Plus tard, vous verrez comment les trois panneaux utilisent ces agrégats écrits dans la table deployment_cpu_stats_per_hr pour visualiser les métriques pertinentes pour le panneau.

```
{
    "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": {
```

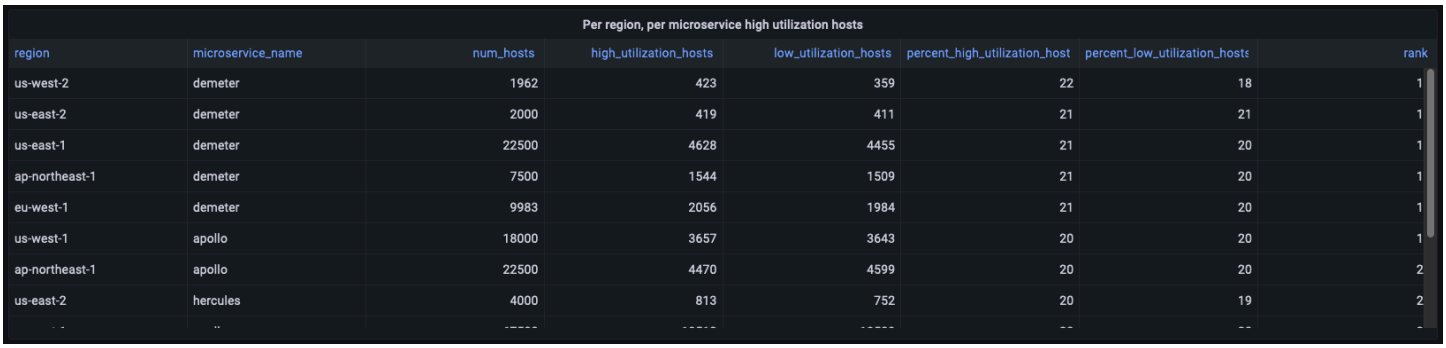
```
            "TopicArn": "******"
        }
    },
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName": "derived",
            "TableName": "deployment_cpu_stats_per_hr",
            "TimeColumn": "hour",
            "DimensionMappings": [
                {
                    "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": "******"
}
```

Tableau de bord à partir de résultats précalculés

Hôtes à taux CPU d'utilisation élevé

Pour les hôtes à forte utilisation, vous verrez comment les différents panneaux utilisent les données de deployment_cpu_stats_per_hr pour calculer les différents agrégats nécessaires aux panneaux. Par exemple, ce panneau fournit des informations au niveau régional, de sorte qu'il présente des agrégats regroupés par région et par microservice, sans filtrer aucune région ou microservice.



Per region, per microservice high utilization hosts

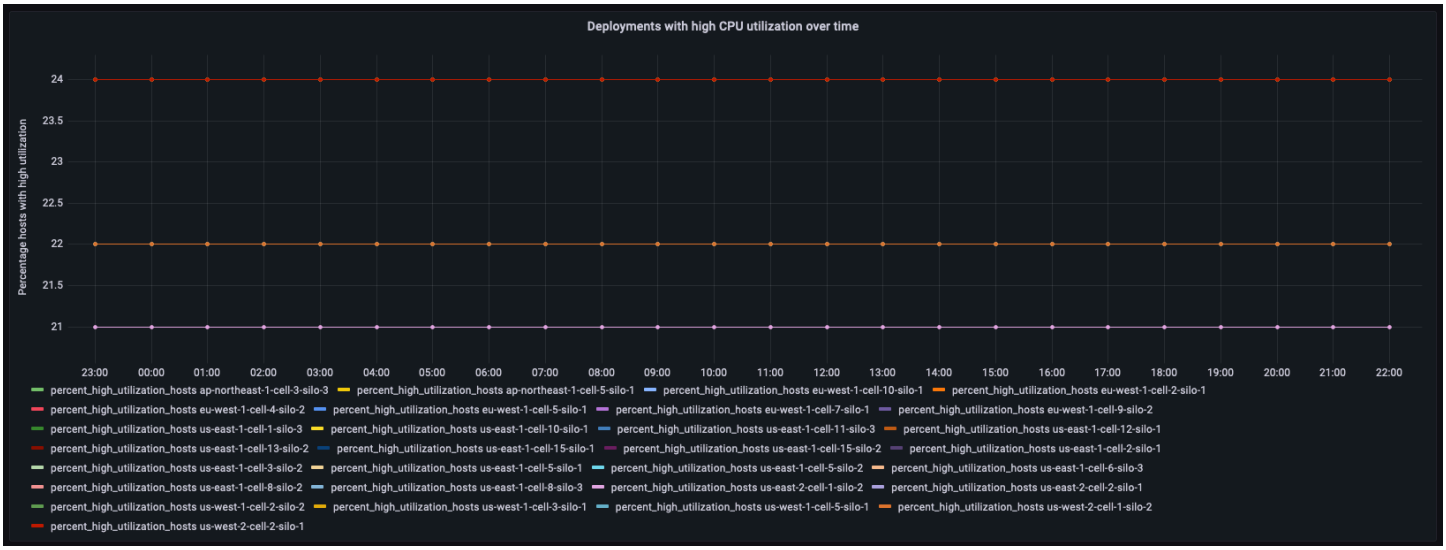
region	microservice_name	num_hosts	high_utilization_hosts	low_utilization_hosts	percent_high_utilization_host	percent_low_utilization_hosts	rank
us-west-2	demeter	1962	423	359	22	18	1
us-east-2	demeter	2000	419	411	21	21	1
us-east-1	demeter	22500	4628	4455	21	20	1
ap-northeast-1	demeter	7500	1544	1509	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
```

Explorez un microservice pour trouver les déploiements les plus CPU utilisés

L'exemple suivant utilise à nouveau la table dérivée deployment_cpu_stats_per_hr, mais applique désormais un filtre pour un microservice spécifique (demeter dans cet exemple, car il signalait des hôtes à taux d'utilisation élevé dans le tableau de bord agrégé). Ce panneau suit le pourcentage d'hôtes à taux d'CPUutilisation élevé au fil du temps.




```
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
```

Comparaison d'une requête sur une table de base avec une requête de résultats de requête planifiés

Dans cet exemple de requête Timestream, nous utilisons le schéma, les exemples de requêtes et les sorties suivants pour comparer une requête sur une table de base avec une requête sur une table dérivée des résultats de requête planifiés. Avec une requête planifiée bien planifiée, vous pouvez obtenir une table dérivée comportant moins de lignes et présentant d'autres caractéristiques susceptibles de conduire à des requêtes plus rapides que celles de la table de base d'origine.

Pour une vidéo décrivant ce scénario, voir Améliorer les performances des requêtes et réduire les coûts à l'aide de requêtes planifiées dans Amazon LiveAnalytics Timestream pour.

Pour cet exemple, nous utilisons le scénario suivant :

- Région — us-east-1
- Table de base — `"clickstream"."shopping"`
- Tableau dérivé — `"clickstream"."aggregate"`

Table de base

Le schéma de la table de base est décrit ci-dessous.

Colonne	Type	Timestream pour le type d'attribut LiveAnalytics
channel	varchar	MULTI
description	varchar	MULTI
event	varchar	DIMENSION
ip_address	varchar	DIMENSION
nom_mesure	varchar	MEASURE_NAME
produit	varchar	MULTI
product_id	varchar	MULTI
quantity	double	MULTI
query	varchar	MULTI
session_id	varchar	DIMENSION
groupe_utilisateur	varchar	DIMENSION
user_id	varchar	DIMENSION

Ce qui suit décrit les mesures de la table de base. Une table de base fait référence à une table dans Timestream sur laquelle une requête planifiée est exécutée.

- nom_mesure — `metrics`
- données — multi
- dimensions :

```
[ ( user_group, varchar ),( user_id, varchar ),( session_id, varchar ),( ip_address,
 varchar ),( event, varchar ) ]
```

Requête sur une table de base

Voici une requête ad hoc qui rassemble les dénombrements par un agrégat de 5 minutes dans un intervalle de temps donné.

```
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
```

Sortie :

```
duration:1.745 sec
Bytes scanned: 29.89 MB
Query Id: AEBQEANMHG7MHHBHCKJ3BSOE3QUGIDBGWCCP5I6J6YUW5CVJZ2M3JCJ27QRMM7A
Row count:5
```

Requête planifiée

Voici une requête planifiée qui s'exécute toutes les 5 minutes.

```
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
```

Requête sur une table dérivée

Voici une requête ad hoc sur une table dérivée. Une table dérivée fait référence à une table Timestream qui contient les résultats d'une requête planifiée.

```
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'
```

Sortie :

```
duration: 0.2960 sec
Bytes scanned: 235.00 B
QueryID: AEBQEANMHHAAQU4FFTT6CFM6UYXTL4SMLZV22MFP4KV2Z7IRVOPLOMLDD6BR33Q
Row count: 5
```

Comparison (Comparaison)

Voici une comparaison des résultats d'une requête sur une table de base avec ceux d'une requête sur une table dérivée. La même requête sur une table dérivée dont les résultats sont agrégés par le biais d'une requête planifiée s'exécute plus rapidement avec moins d'octets numérisés.

Ces résultats montrent l'intérêt d'utiliser des requêtes planifiées pour agréger des données afin d'accélérer les requêtes.

	Requête sur la table de base	Requête sur une table dérivée
Durée	1,745 secondes	0,2960 sec
Octets scannés	29,89 MB	235 octets
Nombre de lignes	5	5

Utilisation UNLOAD pour exporter les résultats d'une requête vers S3 depuis Timestream pour LiveAnalytics

Amazon Timestream vous permet actuellement LiveAnalytics d'exporter les résultats de vos requêtes vers Amazon S3 de manière rentable et sécurisée à l'aide de l'instruction. `UNLOAD` À l'aide de `UNLOAD` cette instruction, vous pouvez désormais exporter des données de séries chronologiques vers des compartiments S3 sélectionnés au format Apache Parquet ou au format Comma Separated Values (CSV), ce qui permet de stocker, de combiner et d'analyser vos données de séries chronologiques avec d'autres services en toute flexibilité. La `UNLOAD` déclaration vous

permet d'exporter les données de manière compressée, ce qui réduit les données transférées et l'espace de stockage requis. UNLOADprend également en charge le partitionnement en fonction d'attributs sélectionnés lors de l'exportation des données, ce qui améliore les performances et réduit le temps de traitement des services en aval accédant aux données. En outre, vous pouvez utiliser les clés gérées par Amazon S3 (SSE-S3) ou les AWS clés gérées par le service de gestion des clés (AWS KMS) (SSE-KMS) pour chiffrer vos données exportées.

Avantages UNLOAD de Timestream pour LiveAnalytics

Les principaux avantages de l'utilisation de UNLOAD cette déclaration sont les suivants.

- Facilité opérationnelle : grâce à UNLOAD cette instruction, vous pouvez exporter des gigaoctets de données en une seule demande de requête au format Apache Parquet ou CSV au format Apache, ce qui vous permet de sélectionner le format le mieux adapté à vos besoins de traitement en aval et de faciliter la création de lacs de données.
- Sécurisé et rentable : UNLOAD Statement permet d'exporter vos données vers un compartiment S3 de manière compressée et de chiffrer (SSE- KMS ou SSE _S3) vos données à l'aide de clés gérées par le client, réduisant ainsi les coûts de stockage des données et les protégeant contre tout accès non autorisé.
- Performance — À l'aide de UNLOAD cette instruction, vous pouvez partitionner les données lors de l'exportation vers un compartiment S3. Le partitionnement des données permet aux services en aval de traiter les données en parallèle, réduisant ainsi leur temps de traitement. En outre, les services en aval ne peuvent traiter que les données dont ils ont besoin, ce qui réduit les ressources de traitement requises et donc les coûts associés.

Cas d'utilisation UNLOAD de From Timestream pour LiveAnalytics

Vous pouvez utiliser l'UNLOADinstruction pour écrire des données dans votre compartiment S3 comme suit.

- Créez un entrepôt de données : vous pouvez exporter des gigaoctets de résultats de requêtes dans un compartiment S3 et ajouter plus facilement des séries chronologiques dans votre lac de données. Vous pouvez utiliser des services tels qu'Amazon Athena et Amazon Redshift pour combiner les données de vos séries chronologiques avec d'autres données pertinentes afin d'obtenir des informations commerciales complexes.
- Créez des pipelines de données d'intelligence artificielle et de machine learning : UNLOAD cette déclaration vous permet de créer facilement des pipelines de données pour vos modèles

d'apprentissage automatique qui accèdent aux données de séries chronologiques, ce qui facilite l'utilisation des données de séries chronologiques avec des services tels qu'Amazon SageMaker et AmazonEMR.

- Simplification ETL du traitement — L'exportation de données dans des compartiments S3 peut simplifier le processus d'extraction, de transformation, de chargement (ETL) sur les données, vous permettant ainsi d'utiliser facilement des outils ou des AWS services tiers tels que AWS Glue pour traiter et transformer les données.

UNLOADConcepts

Syntaxe

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

où se `option` trouve

```
{ 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>'
 | }
```

Paramètres

SELECTdéclaration

L'instruction de requête utilisée pour sélectionner et récupérer les données d'un ou de plusieurs Timestream pour les LiveAnalytics tables.

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

Clause TO

```
TO 's3://bucket-name/folder'
```

or

```
TO 's3://access-point-alias/folder'
```

La TO clause contenue dans l'UNLOADinstruction indique la destination de sortie des résultats de la requête. Vous devez fournir le chemin complet, y compris le nom du compartiment Amazon S3 ou Amazon S3 access-point-alias avec l'emplacement du dossier sur Amazon S3 où Timestream for LiveAnalytics écrit les objets du fichier de sortie. Le compartiment S3 doit appartenir au même compte et se trouver dans la même région. Outre le jeu de résultats de la requête, Timestream for LiveAnalytics écrit le manifeste et les fichiers de métadonnées dans le dossier de destination spécifié.

PARTITIONEDClause _BY

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

La `partitioned_by` clause est utilisée dans les requêtes pour regrouper et analyser les données à un niveau granulaire. Lorsque vous exportez les résultats de votre requête vers le compartiment S3, vous pouvez choisir de partitionner les données en fonction d'une ou de plusieurs colonnes de la requête de sélection. Lors du partitionnement des données, les données exportées sont divisées en sous-ensembles en fonction de la colonne de partition et chaque sous-ensemble est stocké dans un dossier distinct. Dans le dossier de résultats qui contient vos données exportées, un sous-dossier `folder/results/partition column = partition value/` est automatiquement créé. Notez toutefois que les colonnes partitionnées ne sont pas incluses dans le fichier de sortie.

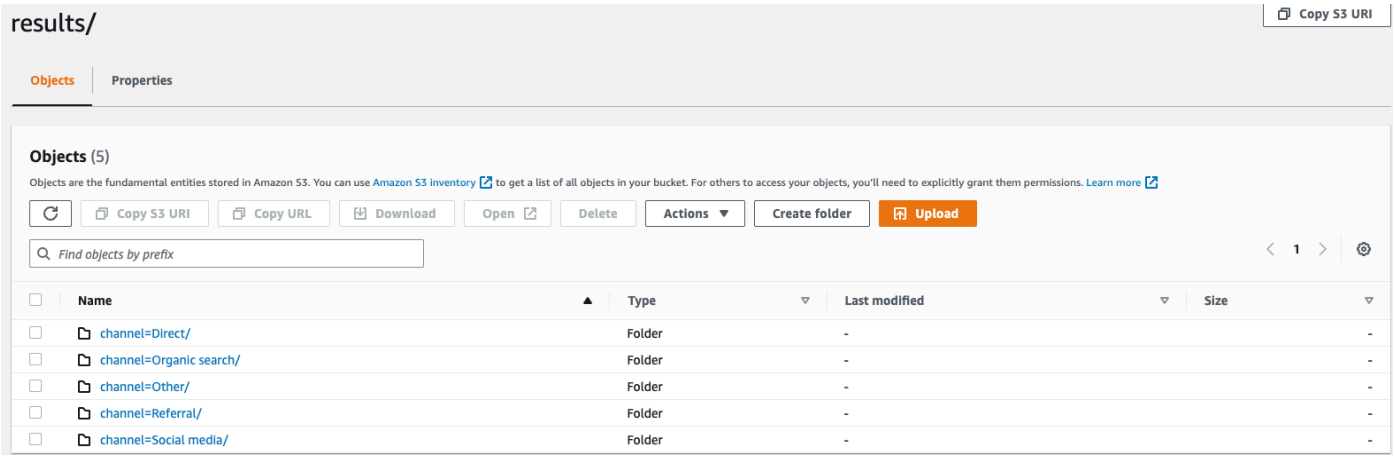
`partitioned_by`n'est pas une clause obligatoire dans la syntaxe. Si vous choisissez d'exporter les données sans partitionnement, vous pouvez exclure la clause dans la syntaxe.

Example

En supposant que vous surveillez les données du flux de clics de votre site Web et que vous disposiez de 5 canaux de trafic `direct`Social Media, à savoir`Organic Search`,`Other`, et`Referral`. Lorsque vous exportez les données, vous pouvez choisir de les partitionner à l'aide de la colonne`Channel`. Dans votre dossier de données`s3://bucketname/results`,

vous aurez cinq dossiers portant chacun leur nom de chaîne respectif. Par exemple, `s3://bucketname/results/channel=Social Media/`. dans ce dossier, vous trouverez les données de tous les clients qui ont accédé à votre site Web via le `Social Media` canal. De même, vous aurez d'autres dossiers pour les chaînes restantes.

Données exportées partitionnées par colonne de canal



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

Les mots clés permettant de spécifier le format des résultats de requête écrits dans votre compartiment S3. Vous pouvez exporter les données soit sous forme de valeur séparée par des virgules (CSV) en utilisant une virgule (,) comme séparateur par défaut, soit au format Apache Parquet, un format de stockage en colonnes ouvert efficace pour les analyses.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Vous pouvez compresser les données exportées à l'aide d'un algorithme de compression GZIP ou les décompresser en spécifiant l'`NONE`option.

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

Les fichiers de sortie sur Amazon S3 sont chiffrés à l'aide de l'option de chiffrement que vous avez sélectionnée. Outre vos données, le manifeste et les fichiers de métadonnées sont également chiffrés en fonction de l'option de chiffrement que vous avez sélectionnée. Nous prenons

actuellement en charge le KMS chiffrement SSE _S3 et SSE _. SSE_S3 est un chiffrement côté serveur, Amazon S3 chiffrant les données à l'aide d'un cryptage standard de chiffrement avancé () 256 bits. AES SSE_ KMS est un chiffrement côté serveur qui permet de chiffrer les données à l'aide de clés gérées par le client.

KMS_KEY

```
kms_key = '<string>'
```

KMSLa clé est une clé définie par le client pour chiffrer les résultats de requête exportés. KMSLa clé est gérée de manière sécurisée par le service de gestion des AWS clés (AWS KMS) et utilisée pour chiffrer les fichiers de données sur Amazon S3.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Lors de l'exportation des données au CSV format, ce champ spécifie un seul ASCII caractère utilisé pour séparer les champs du fichier de sortie, tel qu'un tube (|), une virgule (,) ou un onglet (/t). Le séparateur par défaut pour les CSV fichiers est une virgule. Si une valeur de vos données contient le délimiteur choisi, celui-ci sera mis entre guillemets. Par exemple, si la valeur de vos données contient`Time,stream`, elle sera indiquée entre guillemets comme `"Time,stream"` dans les données exportées. Les guillemets utilisés par Timestream LiveAnalytics sont des guillemets doubles («).

Évitez de spécifier le caractère de renvoi (ASCII13, hexadécimal`0D`, texte « \ r ») ou le caractère de saut de ligne (ASCII10, hexadécimal 0A, texte «\n») comme tel `FIELD_DELIMITER` si vous souhaitez inclure des en-têtes dans leCSV, car cela empêcherait de nombreux analyseurs de pouvoir analyser correctement les en-têtes dans la sortie résultante. CSV

ESCAPED_PAR

```
escaped_by = '<character>', default: (\)
```

Lors de l'exportation des données au CSV format, ce champ indique le caractère qui doit être traité comme un caractère d'échappement dans le fichier de données écrit dans le compartiment S3. L'évasion se produit dans les scénarios suivants :

1. Si la valeur elle-même contient le caractère guillemet («), elle sera échappée à l'aide d'un caractère d'échappement. Par exemple, si la valeur est`Time"stream`, où (\) est le caractère d'échappement configuré, il sera échappé en tant que`Time\"stream`.

2. Si la valeur contient le caractère d'échappement configuré, il sera échappé. Par exemple, si la valeur est`Time\stream`, elle sera ignorée en tant que`Time\\stream`.

> **Note**
>
> Si la sortie exportée contient des types de données complexes tels que des tableaux, des lignes ou des séries temporelles, elle sera sérialisée sous forme de chaîne. JSON Voici un exemple.

Type de données	Valeur réelle	Comment la valeur est échappée au CSV format [JSONchaîne sérialisée]
Tableau	`[ 23,24,25 ]`	`"[23,24,25]"`
Rangée	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\":\"hello\"}"`
Chronologique	`[ ( 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'
```

Lorsque vous exportez les données au CSV format, ce champ vous permet d'inclure les noms de colonnes dans la première ligne des fichiers de CSV données exportés.

Les valeurs acceptées sont « vrai » et « faux » et la valeur par défaut est « faux ». Les options de transformation de texte telles que `escaped_by` et `field_delimiter` s'appliquent également aux en-têtes.

> **Note**
>
> Lorsque vous incluez des en-têtes, il est important de ne pas sélectionner un caractère de renvoi (ASCII13, hexadécimal 0D, texte « \ r ») ou un caractère de saut de ligne (ASCII10, hexadécimal 0A, texte «\n») comme caractère`FIELD_DELIMITER`, car cela empêcherait de nombreux analyseurs de pouvoir analyser correctement les en-têtes dans le résultat obtenu. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Ce champ indique la taille maximale des fichiers créés par l'`UNLOAD`instruction dans Amazon S3. L'`UNLOAD`instruction peut créer plusieurs fichiers, mais la taille maximale de chaque fichier écrit sur Amazon S3 sera approximativement celle spécifiée dans ce champ.

La valeur du champ doit être comprise entre 16 Mo et 78 Go inclus. Vous pouvez le spécifier en nombre entier tel que`12GB`, ou en décimaux tels que `0.5GB` ou. `24.7MB` La valeur par défaut est de 78 Go.

La taille réelle du fichier est approximative au moment de l'écriture du fichier, de sorte que la taille maximale réelle peut ne pas être exactement égale au nombre que vous spécifiez.

Qu'est-ce qui est écrit dans mon compartiment S3 ?

Pour chaque UNLOAD requête exécutée avec succès, Timestream for LiveAnalytics écrit les résultats de la requête, le fichier de métadonnées et le fichier manifeste dans le compartiment S3. Si vous avez partitionné les données, tous les dossiers de partition se trouvent dans le dossier des résultats. Le fichier manifeste contient la liste des fichiers écrits par la UNLOAD commande. Le fichier de métadonnées contient des informations qui décrivent les caractéristiques, les propriétés et les attributs des données écrites.

Quel est le nom du fichier exporté ?

Le nom du fichier exporté contient deux composants, le premier est le QueryID et le second est un identifiant unique.

CSVfichiers

```
S3://bucket_name/results/<queryid>_<UUID>.csv
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.csv
```

CSVFichier compressé

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.gz
```

Lime à parquet

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.parquet
```

Fichiers de métadonnées et de manifeste

```
S3://bucket_name/<queryid>_<UUID>_manifest.json
S3://bucket_name/<queryid>_<UUID>_metadata.json
```

Comme les données au CSV format sont stockées au niveau du fichier, lorsque vous compressez les données lors de l'exportation vers S3, le fichier porte l'extension « .gz ». Cependant, les données de Parquet sont compressées au niveau des colonnes. Ainsi, même lorsque vous compressez les données lors de l'exportation, le fichier portera toujours l'extension .parquet.

Quelles sont les informations contenues dans chaque fichier ?

Fichier manifeste

Le fichier manifeste fournit des informations sur la liste des fichiers exportés lors de l'UNLOADexécution. Le fichier manifeste est disponible dans le compartiment S3 fourni sous le nom de fichier :`s3://<bucket_name>/<queryid>_<UUID>_manifest.json`. Le fichier manifeste contiendra l'URL des fichiers du dossier de résultats, le nombre d'enregistrements et la taille des fichiers respectifs, ainsi que les métadonnées de la requête (c'est-à-dire le nombre total d'octets et le nombre total de lignes exportés vers S3 pour la requête).

```
{
```

```
  "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"
  }
}
```

Metadonnées

Le fichier de métadonnées fournit des informations supplémentaires sur l'ensemble de données, telles que le nom de colonne, le type de colonne et le schéma. <queryid>Le fichier de métadonnées est disponible dans le compartiment S3 fourni sous le nom de fichier : S3 : //bucket_name/ _< >_metadata.json UUID

Voici un exemple de fichier de métadonnées.

```
{
```

```
    "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"
  }
}
```

Les informations de colonne partagées dans le fichier de métadonnées ont la même structure que celle `ColumnInfo` envoyée dans API la réponse aux `SELECT` requêtes.

Résultats

Le dossier des résultats contient vos données exportées au format Apache Parquet ou CSV au format Apache.

Exemple

Lorsque vous soumettez une UNL0AD requête comme ci-dessous via QueryAPI,

```
UNL0AD(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')
```

UNL0ADla réponse à la requête comportera 1 ligne* 3 colonnes. Ces 3 colonnes sont les suivantes :

- lignes de type BIGINT : indique le nombre de lignes exportées
- metadataFile de type VARCHAR - qui est le S3 URI du fichier de métadonnées exporté
- manifestFile de type VARCHAR - qui est le S3 URI du fichier manifeste exporté

Vous obtiendrez la réponse suivante de Query API :

```
{
    "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
    }
}
```

Types de données

La UNLOAD déclaration prend en charge tous les types de données du langage de requête LiveAnalytics de Timestream for décrit dans Types de données pris en charge sauf `time` et. `unknown`

Conditions préalables à UNLOAD partir de Timestream pour LiveAnalytics

Vous trouverez ci-dessous les conditions requises pour écrire des données dans S3 à l'aide UNLOAD de Timestream pour. LiveAnalytics

- Vous devez être autorisé à lire les données du Timestream pour les LiveAnalytics tables à utiliser dans une UNLOAD commande.
- Vous devez disposer d'un compartiment Amazon S3 dans la même AWS région que votre Timestream pour les LiveAnalytics ressources.

- Pour le compartiment S3 sélectionné, assurez-vous que la politique du compartiment S3 dispose également des autorisations permettant à Timestream d' LiveAnalytics exporter les données.
- Les informations d'identification utilisées pour exécuter la UNLOAD requête doivent disposer des autorisations AWS Identity and Access Management (IAM) nécessaires pour permettre LiveAnalytics à Timestream d'écrire les données dans S3. Voici un exemple de politique :

```
{
    "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>/*"
            ]
        }
    ]
}
```

Pour plus de détails sur ces autorisations d'écriture S3, reportez-vous au guide Amazon Simple Storage Service. Si vous utilisez une KMS clé pour chiffrer les données exportées, reportez-vous à la section suivante pour connaître les IAM politiques supplémentaires requises.

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

Bonnes pratiques pour UNLOAD Timestream pour LiveAnalytics

Vous trouverez ci-dessous les meilleures pratiques liées à la UNLOAD commande.

- La quantité de données pouvant être exportée vers le compartiment S3 à l'aide de la `UNLOAD` commande n'est pas limitée. Cependant, la requête expire au bout de 60 minutes et nous recommandons de ne pas exporter plus de 60 Go de données en une seule requête. Si vous devez exporter plus de 60 Go de données, répartissez le travail sur plusieurs requêtes.
- Bien que vous puissiez envoyer des milliers de demandes à S3 pour télécharger les données, il est recommandé de paralléliser les opérations d'écriture sur plusieurs préfixes S3. Reportez-vous à la documentation ici. Le débit API d'appels S3 peut être limité lorsque plusieurs lecteurs/écrivains accèdent au même dossier.
- Compte tenu de la limite de longueur de clé S3 pour définir un préfixe, nous recommandons que les noms des compartiments et des dossiers ne dépassent pas 10 à 15 caractères, en particulier lors de l'utilisation d'une `partitioned_by` clause.
- Lorsque vous recevez un 4XX ou un 5XX pour des requêtes contenant l'`UNLOAD`instruction, il est possible que des résultats partiels soient écrits dans le compartiment S3. Timestream for LiveAnalytics ne supprime aucune donnée de votre bucket. Avant d'exécuter une autre `UNLOAD` requête avec la même destination S3, nous vous recommandons de supprimer manuellement les fichiers créés par l'échec de la requête. Vous pouvez identifier les fichiers écrits par une requête ayant échoué avec le code correspondant`QueryExecutionId`. En cas d'échec des requêtes, Timestream for LiveAnalytics n'exporte pas de fichier manifeste vers le compartiment S3.
- Timestream for LiveAnalytics utilise le téléchargement en plusieurs parties pour exporter les résultats des requêtes vers S3. Lorsque vous recevez un 4XX ou un 5XX de Timestream pour des requêtes contenant un UNLOAD relevé, Timestream LiveAnalytics for LiveAnalytics fait de son mieux pour abandonner le téléchargement en plusieurs parties, mais il est possible que certaines parties incomplètes soient laissées pour compte. C'est pourquoi nous vous recommandons de configurer un nettoyage automatique des téléchargements partitionnés incomplets dans votre compartiment S3 en suivant les instructions fournies ici.

Recommandations pour accéder aux données au CSV format à l'aide d'un CSV analyseur

- `CSV`les analyseurs ne vous permettent pas d'avoir le même caractère dans le séparateur, le caractère d'échappement et le caractère entre guillemets.
- Certains CSV analyseurs ne peuvent pas interpréter les types de données complexes tels que les tableaux. Nous vous recommandons de les interpréter via JSON un désérialiseur.

Recommandations pour accéder aux données au format Parquet

1. Si votre cas d'utilisation nécessite la prise en charge de UTF -8 caractères dans le schéma, c'est-à-dire le nom de colonne, nous vous recommandons d'utiliser la bibliothèque Parquet-MR.
2. L'horodatage de vos résultats est représenté par un entier de 12 octets () INT96
3. Les séries temporelles seront représentées sous `array<row<time, value>>` la forme suivante : les autres structures imbriquées utiliseront les types de données correspondants pris en charge au format Parquet

Utilisation de la clause partition_by

- La colonne utilisée dans le `partitioned_by` champ doit être la dernière colonne de la requête de sélection. Si plusieurs colonnes sont utilisées dans le `partitioned_by` champ, les colonnes doivent être les dernières de la requête de sélection et dans le même ordre que celui utilisé dans le `partition_by` champ.
- Les valeurs de colonne utilisées pour partitionner les données (`partitioned_by`champ) ne peuvent contenir que des ASCII caractères. Alors que Timestream pour LiveAnalytics autorise UTF -8 caractères dans les valeurs, S3 ne prend en charge que les ASCII caractères en tant que clés d'objet.

Exemple de cas d'utilisation UNLOAD de From Timestream pour LiveAnalytics

Supposons que vous surveillez les statistiques de session utilisateur, les sources de trafic et les achats de produits sur votre site Web de commerce électronique. Vous utilisez Timestream LiveAnalytics pour obtenir des informations en temps réel sur le comportement des utilisateurs, les ventes de produits et effectuer des analyses marketing sur les canaux de trafic (recherche organique, réseaux sociaux, trafic direct, campagnes payantes, etc.) qui dirigent les clients vers le site Web.

Rubriques

- Exporter les données sans aucune partition
- Partitionnement des données par canal
- Partitionnement des données par événement
- Partitionnement des données par canal et par événement
- Fichiers de manifeste et de métadonnées

- Utiliser les robots d'exploration Glue pour créer le catalogue de données Glue

Exporter les données sans aucune partition

Vous souhaitez exporter les deux derniers jours de vos données au CSV format.

```
UNLOAD(SELECT user_id, ip_address, event, session_id, measure_name, time,
query, quantity, product_id, channel
FROM sample_clickstream.sample_shopping
WHERE time BETWEEN ago(2d) AND now())
TO 's3://<bucket_name>/withoutpartition'
WITH (  format='CSV',
compression='GZIP')
```

Partitionnement des données par canal

Vous souhaitez exporter les données des deux derniers jours au CSV format, mais vous souhaitez disposer des données de chaque canal de trafic dans un dossier distinct. Pour ce faire, vous devez partitionner les données à l'aide de la `channel` colonne, comme indiqué ci-dessous.

```
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')
```

Partitionnement des données par événement

Vous souhaitez exporter les données des deux derniers jours au CSV format, mais vous souhaitez disposer des données de chaque événement dans un dossier distinct. Pour ce faire, vous devez partitionner les données à l'aide de la `event` colonne, comme indiqué ci-dessous.

```
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')
```

Partitionnement des données par canal et par événement

Vous souhaitez exporter les données des deux derniers jours au CSV format, mais vous souhaitez que les données de chaque canal soient stockées dans un dossier distinct au sein du canal. Pour ce faire, vous devez partitionner les données en utilisant à la fois `event` la colonne `channel` et la colonne, comme indiqué ci-dessous.

```
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')
```

Fichiers de manifeste et de métadonnées

Fichier manifeste

Le fichier manifeste fournit des informations sur la liste des fichiers exportés lors de l'UNLOADexécution. Le fichier manifeste est disponible dans le compartiment S3 fourni sous le nom de fichier :`S3://bucket_name/<queryid>_<UUID>_manifest.json`. Le fichier manifeste contiendra l'URL des fichiers du dossier de résultats, le nombre d'enregistrements et la taille des fichiers respectifs, ainsi que les métadonnées de la requête (c'est-à-dire le nombre total d'octets et le nombre total de lignes exportés vers S3 pour la requête).

```
{
  "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"
  }
}
```

Metadonnées

Le fichier de métadonnées fournit des informations supplémentaires sur l'ensemble de données, telles que le nom de colonne, le type de colonne et le schéma. <queryid>Le fichier de métadonnées est disponible dans le compartiment S3 fourni sous le nom de fichier : S3 : //bucket_name/ _< >_metadata.json UUID

Voici un exemple de fichier de métadonnées.

```
{
    "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"
  }
}
```

Les informations de colonne partagées dans le fichier de métadonnées ont la même structure que celle `ColumnInfo` envoyée dans API la réponse aux `SELECT` requêtes.

Utiliser les robots d'exploration Glue pour créer le catalogue de données Glue

1. Connectez-vous à votre compte avec les informations d'identification d'administrateur pour la validation suivante.
2. Créez une base de données Crawler for Glue en suivant les instructions fournies ici. Veuillez noter que le dossier S3 à fournir dans la source de données doit être le dossier de `UNLOAD` résultats tel que`s3://my_timestream_unloads/results`.
3. Exécutez le crawler en suivant les instructions indiquées ici.

4. Consultez le tableau Glue.
   - Accédez à AWS Glue → Tables.
   - Vous verrez une nouvelle table créée avec le préfixe de table fourni lors de la création du crawler.
   - Vous pouvez consulter les informations relatives au schéma et à la partition en cliquant sur la vue détaillée de la table.

Vous trouverez ci-dessous d'autres AWS services et projets open source qui utilisent le catalogue de données AWS Glue.

- Amazon Athena — Pour plus d'informations, consultez la section Présentation des tables, des bases de données et des catalogues de données dans le guide de l'utilisateur d'Amazon Athena.
- Amazon Redshift Spectrum — Pour plus d'informations, consultez la section Interrogation de données externes à l'aide d'Amazon Redshift Spectrum dans le manuel Amazon Redshift Database Developer Guide.
- Amazon EMR — Pour plus d'informations, consultez la section Utiliser des politiques basées sur les ressources pour l'EMRaccès d'Amazon à AWS Glue Data Catalog dans le guide de EMR gestion Amazon.
- AWS Client Glue Data Catalog pour Apache Hive Metastore — Pour plus d'informations sur ce GitHub projet, voir Client AWS Glue Data Catalog pour Apache Hive Metastore.

Limites pour UNLOAD Timestream pour LiveAnalytics

Les limites liées à la `UNLOAD` commande sont les suivantes.

- La simultanéité des requêtes utilisant l'`UNLOAD`instruction est d'une requête par seconde (QPS). Le dépassement du taux de requêtes peut entraîner un ralentissement.
- Les requêtes contenant `UNLOAD` une instruction peuvent exporter au maximum 100 partitions par requête. Nous vous recommandons de vérifier le nombre distinct de la colonne sélectionnée avant de l'utiliser pour partitionner les données exportées.
- Les requêtes contenant `UNLOAD` une instruction arrivent à expiration au bout de 60 minutes.
- La taille maximale des fichiers créés par l'`UNLOAD`instruction dans Amazon S3 est de 78 Go.

Pour les autres limites de Timestream pour LiveAnalytics, voir Quotas

Utilisation des informations relatives aux requêtes pour optimiser les requêtes dans Amazon Timestream

Query Insights est une fonctionnalité d'optimisation des performances qui vous permet d'optimiser vos requêtes, d'améliorer leurs performances et de réduire les coûts. Grâce à Query Insights, vous pouvez évaluer l'efficacité de l'élagage basé sur les clés de partition temporelles, temporelles et spatiales de vos requêtes. À l'aide des informations sur les requêtes, vous pouvez également identifier les domaines à améliorer afin d'améliorer les performances des requêtes. En outre, grâce aux informations sur les requêtes, vous pouvez évaluer l'efficacité avec laquelle vos requêtes utilisent l'indexation basée sur le temps et sur les clés de partition pour optimiser la récupération des données. Pour optimiser les performances des requêtes, il est essentiel d'affiner les paramètres temporels et spatiaux qui régissent l'exécution des requêtes.

Rubriques

Avantages des informations sur les requêtes

Les principaux avantages de l'utilisation de Query Insights sont les suivants :

- Identification des requêtes inefficaces : Query Insights fournit des informations sur l'élagage basé sur le temps et les attributs des tables auxquelles la requête accède. Ces informations vous aident à identifier les tables auxquelles l'accès n'est pas optimal.
- Optimisation de votre modèle de données et de partitionnement : vous pouvez utiliser les informations issues des requêtes pour accéder à votre modèle de données et à votre stratégie de partitionnement et les affiner.
- Optimisation des requêtes : Query Insights met en évidence les possibilités d'utiliser les index de manière plus efficace.

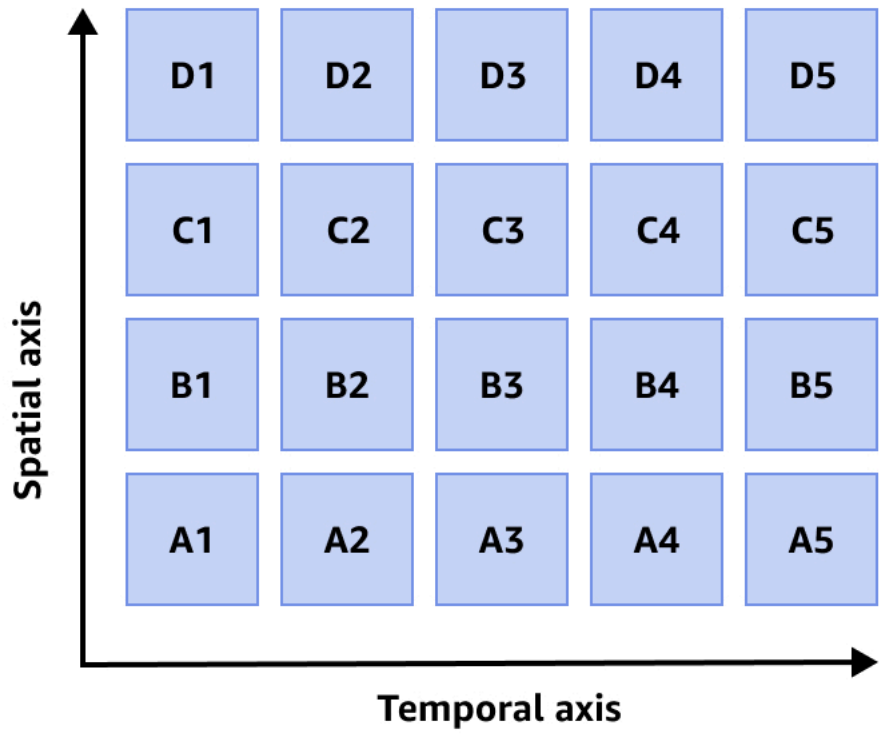
Optimisation de l'accès aux données dans Amazon Timestream

Vous pouvez optimiser les modèles d'accès aux données dans Amazon Timestream à l'aide du schéma de partitionnement Timestream ou des techniques d'organisation des données.

Rubriques

Schéma de partitionnement Timestream

Amazon Timestream utilise un schéma de partitionnement hautement évolutif dans lequel chaque table Timestream peut contenir des centaines, des milliers, voire des millions de partitions indépendantes. Un service de suivi et d'indexation des partitions à haute disponibilité gère le partitionnement, minimisant ainsi l'impact des défaillances et renforçant la résilience du système.



Organisation des données

Timestream stocke chaque point de données ingéré dans une seule partition. Lorsque vous ingérez des données dans une table Timestream, Timestream crée automatiquement des partitions en fonction des horodatages, de la clé de partition et d'autres attributs contextuels contenus dans les données. Outre le partitionnement des données dans le temps (partitionnement temporel), Timestream partitionne également les données en fonction de la clé de partitionnement sélectionnée et d'autres dimensions (partitionnement spatial). Cette approche est conçue pour répartir le trafic d'écriture et permettre un élagage efficace des données pour les requêtes.

La fonction d'analyse des requêtes fournit des informations précieuses sur l'efficacité de l'élagage de la requête, notamment la couverture spatiale et la couverture temporelle des requêtes.

Rubriques

- QuerySpatialCoverage
- QueryTemporalCoverage

QuerySpatialCoverage

La QuerySpatialCoveragemétrique fournit des informations sur la couverture spatiale de la requête exécutée et de la table présentant l'élagage spatial le plus inefficace. Ces informations peuvent vous aider à identifier les points à améliorer dans la stratégie de partitionnement afin d'améliorer l'élagage spatial. La valeur de la `QuerySpatialCoverage` métrique est comprise entre 0 et 1. Plus la valeur de la métrique est faible, plus l'élagage des requêtes sur l'axe spatial est optimal. Par exemple, une valeur de 0,1 indique que la requête scanne 10 % de l'axe spatial. La valeur 1 indique que la requête scanne 100 % de l'axe spatial.

Example Utilisation des informations issues des requêtes pour analyser la couverture spatiale d'une requête

Supposons que vous disposiez d'une base de données Timestream qui stocke des données météorologiques. Supposons que la température soit enregistrée toutes les heures par des stations météorologiques situées dans différents États des États-Unis. Imaginez que vous choisissiez `State` comme clé de partitionnement définie par le client (CDPK) pour partitionner les données par état.

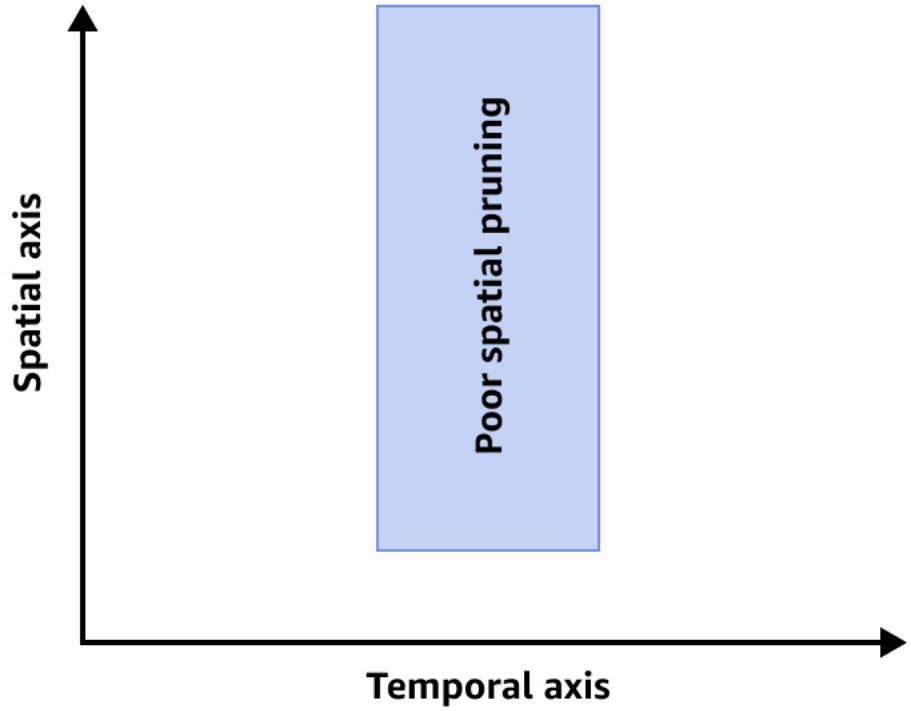
Supposons que vous exécutiez une requête pour récupérer la température moyenne de toutes les stations météorologiques de Californie entre 14 h et 16 h un jour donné. L'exemple suivant montre la requête pour ce scénario.

```
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';
```

À l'aide de la fonctionnalité Query Insights, vous pouvez analyser la couverture spatiale de la requête. Imaginez que la `QuerySpatialCoverage` métrique renvoie une valeur de 0,02. Cela signifie que la requête n'a scanné que 2 % de l'axe spatial, ce qui est efficace. Dans ce cas, la requête a réussi à affiner efficacement la plage spatiale, en ne récupérant que les données de Californie et en ignorant les données d'autres États.

Au contraire, si la `QuerySpatialCoverage` métrique renvoie une valeur de 0,8, cela indique que la requête a scanné 80 % de l'axe spatial, ce qui est moins efficace. Cela peut suggérer que la stratégie de partitionnement doit être affinée pour améliorer l'élagage spatial. Par exemple, vous pouvez sélectionner la clé de partition comme ville ou région plutôt que comme État. En analysant la `QuerySpatialCoverage` métrique, vous pouvez identifier les opportunités d'optimiser votre stratégie de partitionnement et d'améliorer les performances de vos requêtes.

L'image suivante montre un élagage spatial médiocre.



Pour améliorer l'efficacité de l'élagage spatial, vous pouvez effectuer l'une des opérations suivantes ou les deux :

- Ajoutez `measure_name` la clé de partitionnement par défaut ou utilisez les CDPK prédicats dans votre requête.
- Si vous avez déjà ajouté les attributs mentionnés au point précédent, supprimez les fonctions associées à ces attributs ou clauses, telles que`LIKE`.

QueryTemporalCoverage

La `QueryTemporalCoverage` métrique fournit des informations sur la plage temporelle analysée par la requête exécutée, y compris la table contenant la plus grande plage temporelle scannée. La valeur de la `QueryTemporalCoverage` métrique est la plage de temps représentée en nanosecondes. Plus la valeur de cette métrique est faible, plus l'élagage des requêtes sur la plage temporelle est optimal. Par exemple, une requête analysant les données des dernières minutes est plus performante qu'une requête analysant l'ensemble de la plage temporelle de la table.

Example

Supposons que vous disposiez d'une base de données Timestream qui stocke les données des capteurs IoT, avec des mesures prises chaque minute à partir d'appareils situés dans une usine de fabrication. Supposons que vous avez partitionné vos données par`device_ID`.

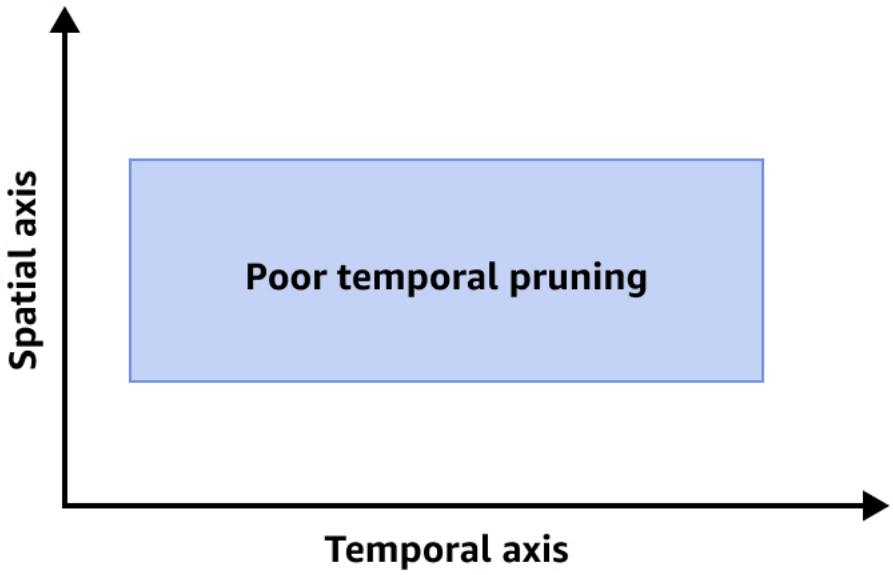
Supposons que vous exécutiez une requête pour récupérer le relevé moyen du capteur pour un appareil spécifique au cours des 30 dernières minutes. L'exemple suivant montre la requête pour ce scénario.

```
SELECT AVG(sensor_reading)
FROM "sensor_data"."factory_1"
WHERE device_id = 'DEV_123'
  AND time >= NOW() - INTERVAL 30 MINUTE and time < NOW();
```

À l'aide de la fonctionnalité Query Insights, vous pouvez analyser la plage temporelle analysée par la requête. Imaginez que la `QueryTemporalCoverage` métrique renvoie une valeur de 1800000000000 nanosecondes (30 minutes). Cela signifie que la requête n'a scanné que les 30 dernières minutes de données, ce qui représente une plage temporelle relativement étroite. C'est un bon signe car cela indique que la requête a réussi à affiner efficacement le partitionnement temporel et à récupérer uniquement les données demandées.

Au contraire, si la `QueryTemporalCoverage` métrique renvoie une valeur de 1 an en nanosecondes, cela indique que la requête a scanné une période d'un an dans le tableau, ce qui est moins efficace. Cela peut indiquer que la requête n'est pas optimisée pour l'élagage temporel, et vous pouvez l'améliorer en ajoutant des filtres temporels.

L'image suivante montre un élagage temporel médiocre.



Pour améliorer l'élagage temporel, nous vous recommandons d'effectuer l'une ou l'ensemble des opérations suivantes :

- Ajoutez les prédicats temporels manquants dans la requête et assurez-vous qu'ils élaguent la fenêtre temporelle souhaitée.
- Supprimez les fonctions`MAX()`, telles que celles relatives aux prédicats temporels.
- Ajoutez des prédicats temporels à toutes les sous-requêtes. Cela est important si vos sous-requêtes joignent de grandes tables ou exécutent des opérations complexes.

Activation de l'analyse des requêtes dans Amazon Timestream

Vous pouvez activer les informations relatives aux requêtes pour vos requêtes grâce à des informations fournies directement par le biais de la réponse à la requête. L'activation des informations sur les requêtes ne nécessite aucune infrastructure supplémentaire ni n'entraîne de coûts supplémentaires. Lorsque vous activez Query Insights, il renvoie des champs de métadonnées liés aux performances des requêtes, en plus des résultats de requête, dans le cadre de la réponse à votre requête. Vous pouvez utiliser ces informations pour ajuster vos requêtes afin d'améliorer les performances des requêtes et de réduire le coût des requêtes.

Pour plus d'informations sur l'activation des informations sur les requêtes, consultezExécuter une requête.

Pour consulter des exemples de réponses renvoyées en activant les informations sur les requêtes, consultez la section Exemples de requêtes planifiées.

> **Note**
>
> - Lorsque vous activez les informations sur les requêtes, le débit limite la requête à une requête par seconde (QPS). Pour éviter tout impact sur les performances, nous vous recommandons vivement d'activer les informations sur les requêtes uniquement pendant la phase d'évaluation de vos requêtes, avant de les déployer en production.
> - Les informations fournies dans les informations des requêtes sont finalement cohérentes, ce qui signifie qu'elles peuvent changer à mesure que de nouvelles données sont continuellement ingérées dans les tables.

Optimisation des requêtes à l'aide des informations sur les requêtes et

Supposons que vous utilisiez Amazon Timestream LiveAnalytics pour surveiller la consommation d'énergie sur différents sites. Imaginez que votre base de données comporte deux tables nommées `raw-metrics` et`aggregate-metrics`.

Le `raw-metrics` tableau contient des données énergétiques détaillées au niveau de l'appareil et contient les colonnes suivantes :

- Horodatage
- État, par exemple, Washington
- ID de l'appareil
- Consommation d'énergie

Les données de ce tableau sont collectées et stockées selon une minute-by-minute granularité. Le tableau utilise `State` commeCDPK.

Le `aggregate-metrics` tableau enregistre le résultat d'une requête planifiée pour agréger les données de consommation d'énergie de tous les appareils toutes les heures. Ce tableau contient les colonnes suivantes :

- Horodatage
- État, par exemple, Washington

- Consommation totale d'énergie

La `aggregate-metrics` table stocke ces données selon une granularité horaire. Le tableau utilise `State` commeCDPK.

Rubriques

- Consultation de la consommation d'énergie des dernières 24 heures
- Optimisation de la requête pour la plage temporelle
- Optimisation de la requête pour la couverture spatiale
- Performances de requête améliorées

Consultation de la consommation d'énergie des dernières 24 heures

Supposons que vous souhaitiez extraire l'énergie totale consommée à Washington au cours des dernières 24 heures. Pour trouver ces données, vous pouvez tirer parti des points forts des deux tableaux : `raw-metrics` et`aggregate-metrics`. Le `aggregate-metrics` tableau fournit des données de consommation d'énergie horaire pour les 23 dernières heures, tandis que le `raw-metrics` tableau fournit des données détaillées par minute pour la dernière heure. En consultant les deux tableaux, vous pouvez obtenir une image complète et précise de la consommation d'énergie à Washington au cours des dernières 24 heures.

```
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()
```

Cet exemple de requête est fourni à titre indicatif uniquement et peut ne pas fonctionner tel quel. Il est destiné à illustrer le concept, mais vous devrez peut-être le modifier pour l'adapter à votre cas d'utilisation ou à votre environnement spécifique.

Après avoir exécuté cette requête, vous remarquerez peut-être que le temps de réponse à la requête est plus lent que prévu. Pour identifier la cause première de ce problème de performance, vous pouvez utiliser la fonctionnalité Query Insights pour analyser les performances de la requête et optimiser son exécution.

L'exemple suivant montre la réponse de Query Insights.

```
queryInsightsResponse={
                QuerySpatialCoverage: {
                    Max: {
                        Value: 1.0,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/raw-metrics,
                        PartitionKey: [State]
                    }
                },
                QueryTemporalRange: {
                    Max: {
                        Value:31540000000000000 //365 days,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/aggregate-metrics
                    }
                },
                QueryTableCount: 2,
                OutputRows: 83,
                OutputBytes: 590
```

La réponse Query Insights fournit les informations suivantes :

- Plage temporelle : la requête a scanné une plage temporelle excessive de 365 jours pour la `aggregate-metrics` table. Cela indique une utilisation inefficace du filtrage temporel.
- Couverture spatiale : La requête a scanné l'ensemble de la plage spatiale (100 %) de la `raw-metrics` table. Cela suggère que le filtrage spatial n'est pas utilisé efficacement.

Si votre requête accède à plusieurs tables, query insights fournit les indicateurs de la table dont le modèle d'accès est le moins optimal.

Optimisation de la requête pour la plage temporelle

Sur la base de la réponse de la requête Insights, vous pouvez optimiser la requête en fonction de la plage temporelle, comme indiqué dans l'exemple suivant.

```
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'
```

Si vous réexécutez la `QueryInsights` commande, elle renvoie la réponse suivante.

```
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
```

Cette réponse montre que la couverture spatiale de la `aggregate-metrics` table est toujours de 100 %, ce qui est inefficace. La section suivante montre comment optimiser la requête pour la couverture spatiale.

Optimisation de la requête pour la couverture spatiale

Sur la base de la réponse à la requête Insights, vous pouvez optimiser la requête pour la couverture spatiale, comme indiqué dans l'exemple suivant.

```
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'
```

Si vous réexécutez la `QueryInsights` commande, elle renvoie la réponse suivante.

```
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
```

Performances de requête améliorées

Une fois la requête optimisée, Query Insights fournit les informations suivantes :

- L'élagage temporel de la `aggregate-metrics` table est de 23 heures. Cela indique que seules 23 heures de la plage temporelle sont scannées.
- L'élagage spatial de la `aggregate-metrics` table est de 0,02. Cela indique que seulement 2 % des données de plage spatiale de la table sont numérisées. La requête analyse une très petite partie des tables, ce qui améliore les performances et réduit l'utilisation des ressources. L'efficacité améliorée de l'élagage indique que la requête est désormais optimisée en termes de performances.

Travailler avec AWS Backup

La fonctionnalité de protection des données d'Amazon Timestream LiveAnalytics for est une solution entièrement gérée qui vous aide à répondre à vos exigences en matière de conformité réglementaire et de continuité des activités. Cette fonctionnalité est activée grâce à l'intégration native à un service de sauvegarde unifié conçu pour simplifier la création, la migration, la restauration et la suppression des sauvegardes, tout en fournissant des rapports et des audits améliorés. AWS Backup Grâce à l'intégration avec AWS Backup, vous pouvez utiliser une solution de protection des données centralisée entièrement gérée et basée sur des règles pour créer des sauvegardes immuables et gérer de manière centralisée la protection des données de votre application couvrant Timestream et les autres AWS services pris en charge par. AWS Backup

Pour utiliser cette fonctionnalité, vous devez accepter d'autoriser la protection AWS Backup de vos ressources Timestream. Les choix d'inscription s'appliquent au compte et à AWS la région spécifiques. Vous devrez donc peut-être vous inscrire dans plusieurs régions en utilisant le même compte. Pour plus d'informations sur AWS Backup, consultez le manuel du AWS Backup développeur.

Les fonctionnalités de protection des données disponibles via AWS Backup incluent les suivantes.

Sauvegardes planifiées : vous pouvez configurer des sauvegardes régulières de votre Timestream pour les LiveAnalytics tables à l'aide de plans de sauvegarde.

Copie entre comptes et entre régions : vous pouvez copier automatiquement vos sauvegardes dans un autre coffre-fort de sauvegarde d'une autre AWS région ou d'un autre compte, ce qui vous permet de répondre à vos exigences en matière de protection des données.

Hiérarchisation du stockage à froid : vous pouvez configurer vos sauvegardes pour mettre en œuvre des règles de cycle de vie afin de supprimer ou de transférer les sauvegardes vers un stockage à froid. Cela peut vous aider à optimiser vos coûts de sauvegarde.

Balises : vous pouvez étiqueter automatiquement vos sauvegardes à des fins de facturation et de répartition des coûts.

Chiffrement : vos données de sauvegarde sont stockées dans le AWS Backup coffre-fort. Cela vous permet de chiffrer et de sécuriser vos sauvegardes à l'aide d'une AWS KMS clé indépendante de votre clé de chiffrement Timestream for LiveAnalytics table.

Sauvegardes sécurisées à l'aide du WORM modèle : vous pouvez utiliser AWS Backup Vault Lock pour activer un paramètre write-once-read-many (WORM) pour vos sauvegardes. Avec AWS Backup

Vault Lock, vous pouvez ajouter une couche de défense supplémentaire qui protège les sauvegardes contre les opérations de suppression involontaires ou malveillantes, les modifications des périodes de conservation des sauvegardes et les mises à jour des paramètres du cycle de vie. Pour en savoir plus, consultez le verrouillage de coffre AWS Backup.

La fonctionnalité de protection des données est disponible dans toutes les régions. Pour en savoir plus sur cette fonctionnalité, consultez le guide du AWS Backup développeur.

Sauvegarde et restauration de tables Timestream : comment cela fonctionne

Vous pouvez créer des sauvegardes de vos tables Amazon Timestream. Cette section présente le processus de sauvegarde et de restauration.

Rubriques

- Sauvegardes
- Restaurations

Sauvegardes

Vous pouvez utiliser la fonctionnalité de sauvegarde à la demande pour créer des sauvegardes complètes de votre Amazon Timestream LiveAnalytics pour les tables. Cette section présente le processus de sauvegarde et de restauration.

Vous pouvez créer une sauvegarde de vos données Timestream selon une granularité de table. Vous pouvez lancer une sauvegarde de la table sélectionnée à l'aide de la console Timestream, ou de AWS Backup la console. SDK CLI La sauvegarde est créée de manière asynchrone et toutes les données de la table jusqu'à l'heure de lancement de la sauvegarde sont incluses dans la sauvegarde. Cependant, il est possible que certaines des données ingérées dans la table pendant que la sauvegarde est en cours soient également incluses dans la sauvegarde. Pour protéger vos données, vous pouvez créer une sauvegarde unique à la demande ou planifier une sauvegarde récurrente de votre table.

Pendant qu'une sauvegarde est en cours, vous ne pouvez pas effectuer les opérations suivantes.

- Suspendre ou annuler l'opération de sauvegarde.
- Supprimer la table source de la sauvegarde.
- Désactiver les sauvegardes d'une table si une sauvegarde de cette tables est en cours.

Une fois configuré, il AWS Backup fournit des plannings de sauvegarde automatisés, une gestion de la rétention et une gestion du cycle de vie, éliminant ainsi le besoin de scripts personnalisés et de processus manuels. Pour plus d'informations, consultez le guide du AWS Backup développeur

Tous les flux temporels pour les LiveAnalytics sauvegardes sont de nature incrémentielle, ce qui implique que la première sauvegarde d'une table est une sauvegarde complète et que chaque sauvegarde suivante de la même table est une sauvegarde incrémentielle, copiant uniquement les modifications apportées aux données depuis la dernière sauvegarde. Comme les données de Timestream for sont LiveAnalytics stockées dans un ensemble de partitions, toutes les partitions modifiées en raison de l'ingestion de nouvelles données ou de mises à jour des données existantes depuis la dernière sauvegarde sont copiées lors des sauvegardes suivantes.

Si vous utilisez Timestream pour LiveAnalytics console, les sauvegardes créées pour toutes les ressources du compte sont répertoriées dans l'onglet Sauvegardes. En outre, les sauvegardes sont également répertoriées dans les détails du tableau.

Restaurations

Vous pouvez restaurer une table à partir du Timestream pour LiveAnalytics console, ou AWS Backup consoleSDK, ou. AWS CLI Vous pouvez soit restaurer l'intégralité des données à partir de votre sauvegarde, soit configurer les paramètres de conservation des tables pour restaurer certaines données. Lorsque vous lancez une restauration, vous pouvez configurer les paramètres du tableau ci-dessous.

- Database Name
- Nom de la table
- Conservation de la mémoire
- Rétention magnétique du magasin
- Activer les écritures sur stockage magnétique
- Emplacement des journaux d'erreurs S3 (facultatif)
- IAMrôle qui AWS Backup sera assumé lors de la restauration de la sauvegarde

Les configurations précédentes sont indépendantes de la table source. Pour restaurer toutes les données de votre sauvegarde, nous vous recommandons de configurer les nouveaux paramètres de la table de telle sorte que la somme de la période de conservation de la mémoire et de la période de conservation de la mémoire magnétique soit supérieure à la différence entre l'horodatage le plus

ancien et le présent. Lorsque vous sélectionnez une sauvegarde incrémentielle à restaurer, toutes les données (incrémentielles et données complètes sous-jacentes) sont restaurées. Une fois la restauration réussie, la table est active et vous pouvez effectuer des opérations d'ingestion et/ou de requête sur la table restaurée. Toutefois, vous ne pouvez pas effectuer ces opérations pendant que la restauration est en cours. Une fois restaurée, la table est similaire à toutes les autres tables de votre compte.

Example Restaurez toutes les données à partir d'une sauvegarde

Cet exemple repose sur les hypothèses suivantes.

Horodatage le plus ancien — `August 1, 2021 0:00:00`

- Maintenant — `November 9, 2022 0:00:00`

Pour restaurer toutes les données d'une sauvegarde, entrez et comparez les valeurs comme suit.

1. Entrez Conservation de la mémoire et Conservation de la mémoire magnétique. Supposons, par exemple, ces valeurs.
   - Conservation de la mémoire : 12 heures
   - Conservation magnétique en magasin —500 jours
2. Trouvez la somme de la rétention de la mémoire et de la rétention de la mémoire magnétique.

   ```
   12 hours + (500 * 24 hours) =
   12 hours + 12,000 hours =
   12,012 hours
   ```
3. Trouvez la différence entre le plus ancien horodatage et le présent.

   ```
   November 9, 2022 0:00:00 - August 1, 2021 0:00:00 =
   465 days =
   465 * 24 hours =
   11,160 hours
   ```
4. Assurez-vous que la somme des valeurs de rétention de la deuxième étape est supérieure à la différence de temps de la troisième étape. Ajustez les durées de conservation si nécessaire.

   ```
   12,012 > 11,160
   true
   ```

Example Restaurer certaines données à partir d'une sauvegarde

Cet exemple repose sur l'hypothèse suivante.

- Maintenant — `November 9, 2022 0:00:00`

Pour restaurer uniquement certaines données d'une sauvegarde, entrez et comparez les valeurs comme suit.

1. Déterminez le premier horodatage requis. Supposons, par exemple`December 4, 2021 0:00:00`.
2. Trouvez la différence entre le premier horodatage requis et l'heure actuelle.

```
November 9, 2022 0:00:00 - December 4, 2021 0:00:00 =
340 days =
340 * 24 hours =
8,160 hours
```

3. Entrez la valeur souhaitée pour la conservation de la mémoire. Par exemple, entrez 12 heures.
4. Soustrayez la valeur de la différence à la deuxième étape.

```
8,160 hours - 12 hours =
8148 hours
```

5. Entrez cette valeur pour Magnetic store retention.

Vous pouvez copier une sauvegarde de votre flux temporel pour les données des LiveAnalytics tables dans une autre AWS région, puis la restaurer dans cette nouvelle région. Vous pouvez copier puis restaurer des sauvegardes entre les régions AWS commerciales et les régions AWS GovCloud (États-Unis). Vous payez uniquement pour les données que vous copiez depuis la région source et pour les données que vous restaurez vers une nouvelle table dans la région de destination.

Une fois le tableau restauré, vous devez configurer manuellement les éléments suivants sur le tableau restauré.

- AWS Politiques d'Identity and Access Management (IAM)
- Balises
- Requêtes planifiées

Les temps de restauration sont directement liés à la configuration de vos tables. Il s'agit notamment de la taille de vos tables, du nombre de partitions sous-jacentes, de la quantité de données restaurées dans la mémoire et d'autres variables. Lors de la planification d'une reprise après sinistre, il est recommandé de documenter régulièrement les délais moyens d'exécution des restaurations et de déterminer l'impact de ces délais sur votre objectif global de temps de restauration (RTO).

Toutes les API actions et consoles de sauvegarde et de restauration sont capturées et enregistrées à AWS CloudTrail des fins de journalisation, de surveillance continue et d'audit.

Création de sauvegardes des tables Amazon Timestream

Cette section explique comment activer AWS Backup et créer des sauvegardes planifiées et à la demande pour Amazon Timestream.

Rubriques

AWS Backup Permettre de protéger Timestream pour les données LiveAnalytics

Vous devez l'activer AWS Backup pour l'utiliser avec Timestream pour. LiveAnalytics

Pour l'activer AWS Backup dans Timestream pour LiveAnalytics console, effectuez les étapes suivantes.

1. Connectez-vous à la console AWS de gestion.
2. Une bannière contextuelle apparaît en haut de la page de votre tableau de LiveAnalytics bord Timestream pour permettre de prendre en charge Timestream AWS Backup pour les données. LiveAnalytics Sinon, dans le volet de navigation, choisissez Backups.
3. Dans la fenêtre Backup, vous verrez la bannière à activer AWS Backup. Sélectionnez Activer.

   La protection des données via AWS Backup est désormais disponible pour votre Timestream pour les LiveAnalytics tables.

Pour l'activer via AWS Backup, reportez-vous à AWS Backup la documentation relative à l'activation via la console et par programmation.

Si vous choisissez de désactiver la protection AWS Backup de votre Timestream pour les LiveAnalytics données une fois que celles-ci ont été activées, connectez-vous via la AWS Backup console et déplacez le bouton vers la gauche.

Si vous ne pouvez pas activer ou désactiver les AWS Backup fonctionnalités, votre AWS administrateur devra peut-être effectuer ces actions.

Création de sauvegardes à la demande

Pour créer une sauvegarde à la demande d'un flux temporel pour une LiveAnalytics table, procédez comme suit.

1. Connectez-vous à la console AWS de gestion.
2. Dans le volet de navigation sur le côté gauche de la console, choisissez Sauvegardes.
3. Choisissez Create on-demand backup (Créer une sauvegarde à la demande).
4. Continuez à sélectionner les paramètres dans la fenêtre de sauvegarde.
5. Vous pouvez soit créer une sauvegarde maintenant, soit lancer une sauvegarde immédiatement, soit sélectionner une fenêtre de sauvegarde pour démarrer la sauvegarde.
6. Sélectionnez la politique de gestion du cycle de vie de votre sauvegarde. Vous pouvez transférer vos données de sauvegarde vers un stockage à froid où vous devez conserver la sauvegarde pendant au moins 90 jours. Vous pouvez définir la période de conservation requise pour votre sauvegarde. Vous pouvez soit sélectionner un coffre-fort existant, soit sélectionner « créer un nouveau coffre-fort de sauvegarde » pour accéder à AWS Backup la console et créer un nouveau coffre-fort de sauvegarde. **<documentation link on creating a new backup vault here>**
7. Sélectionnez le IAM rôle approprié.
8. Si vous souhaitez affecter une ou plusieurs balises à votre sauvegarde à la demande, entrez une clé et une valeur facultative, puis choisissez Add tag (Ajouter une balise).
9. Choisissez de créer une sauvegarde à la demande. Cela vous amène à la page Backup, où vous verrez une liste de tâches.
10. Choisissez l'ID de tâche de sauvegarde de la ressource que vous avez choisi de sauvegarder pour afficher les détails de cette tâche.

Sauvegardes planifiées

Pour planifier une sauvegarde, reportez-vous à la section Création d'une sauvegarde planifiée.

Restauration d'une sauvegarde d'une table Amazon Timestream

Cette section explique comment restaurer une sauvegarde d'une table Amazon Timestream.

Rubriques

Restaurer un flux temporel pour une table à partir de LiveAnalytics AWS Backup

Pour restaurer votre Timestream pour LiveAnalytics table AWS Backup en utilisant Timestream pour LiveAnalytics console, procédez comme suit.

1. Connectez-vous à la console AWS de gestion.
2. Dans le volet de navigation sur le côté gauche de la console, choisissez Sauvegardes.
3. Pour restaurer une ressource, cliquez sur le bouton radio situé à côté de l'ID du point de récupération de la ressource. Dans le coin supérieur droit du volet, choisissez Restaurer.
4. Entrez les paramètres de configuration de la table, à savoir le nom de la base de données et le nom de la table. Notez que le nom de la table restaurée doit être différent du nom de la table source d'origine.
5. Configurez les paramètres de conservation de la mémoire et de la mémoire magnétique.
6. Dans le champ Rôle de restauration, choisissez le IAM rôle qui AWS Backup sera assumé pour cette restauration.
7. Choisissez Restore backup. Un message en haut de la page fournit des informations sur la tâche de restauration.

> **Note**
>
> La restauration de l'intégralité de la sauvegarde vous est facturée, quelles que soient la mémoire configurée et les périodes de conservation du stockage magnétique. Toutefois, une fois la restauration terminée, votre table restaurée ne contiendra les données que pendant les périodes de conservation configurées.

Restaurer le flux temporel d'une LiveAnalytics table vers une autre région ou un autre compte

Pour restaurer le flux temporel d'une LiveAnalytics table dans une autre région ou un autre compte, vous devez d'abord copier la sauvegarde dans cette nouvelle région ou ce nouveau compte. Pour pouvoir effectuer une copie vers un autre compte, ce compte doit d'abord vous accorder l'autorisation. Une fois que vous avez copié votre Timestream pour le LiveAnalytics sauvegarder dans la nouvelle région ou le nouveau compte, vous pouvez le restaurer en suivant le processus décrit dans la section précédente.

Copier une sauvegarde d'une table Amazon Timestream

Vous pouvez effectuer une copie d'une sauvegarde en cours. Vous pouvez copier des sauvegardes vers plusieurs AWS comptes ou AWS régions à la demande ou automatiquement dans le cadre d'un plan de sauvegarde planifié. La réplication entre régions est particulièrement utile en cas d'exigences de continuité d'activité ou de conformité pour stocker les sauvegardes à une distance minimale de vos données de production.

Les sauvegardes entre comptes sont utiles pour copier en toute sécurité vos sauvegardes vers un ou plusieurs comptes AWS de votre organisation pour des raisons opérationnelles ou de sécurité. Si votre sauvegarde d'origine est supprimée par inadvertance, vous pouvez copier la sauvegarde de son compte de destination vers son compte source, puis démarrer la restauration. Avant de pouvoir effectuer cette opération, vous devez disposer de deux comptes appartenant à la même organisation dans le service Organizations et des autorisations requises pour ces comptes. Lorsque vous copiez une sauvegarde incrémentielle dans un autre compte ou une autre région, la sauvegarde complète associée est également copiée.

Les copies héritent de la configuration de la sauvegarde source, sauf indication contraire de votre part. Il y a une exception. Si vous spécifiez que votre nouvelle copie doit « ne jamais » expirer. Avec ce paramètre, la nouvelle copie hérite toujours de sa date d'expiration source. Si vous souhaitez que la nouvelle copie de votre sauvegarde soit permanente, définissez vos sauvegardes source pour qu'elles n'expirent jamais ou spécifiez que votre nouvelle copie expire 100 ans après sa création.

Pour copier une sauvegarde depuis la console Timestream, procédez comme suit.

1. Connectez-vous à la console AWS de gestion.
2. Dans le volet de navigation sur le côté gauche de la console, choisissez Sauvegardes.
3. Cliquez sur le bouton radio situé à côté de l'ID du point de récupération de la ressource. Dans le coin supérieur droit du volet, sélectionnez Actions, puis Copier.

4. Sélectionnez Continuer vers la AWS sauvegarde et suivez les étapes de sauvegarde entre comptes.

La copie de sauvegardes à la demande et planifiées entre comptes et régions n'est pas prise en charge de manière native dans la LiveAnalytics console Timestream pour le moment et vous devez naviguer AWS Backup pour effectuer l'opération.

Suppression de sauvegardes

Cette section explique comment supprimer une sauvegarde d'un Timestream pour LiveAnalytics une table.

Pour supprimer une sauvegarde de la console Timestream, procédez comme suit.

1. Connectez-vous à la console AWS de gestion.
2. Dans le volet de navigation sur le côté gauche de la console, choisissez Sauvegardes.
3. Cliquez sur le bouton radio situé à côté de l'ID du point de récupération de la ressource. Dans le coin supérieur droit du volet, sélectionnez Actions, puis choisissez Supprimer.
4. Sélectionnez Continuer vers la AWS sauvegarde et suivez les étapes de suppression des sauvegardes décrites dans Supprimer des sauvegardes.

> **Note**
>
> Lorsque vous supprimez une sauvegarde incrémentielle, seule la sauvegarde incrémentielle est supprimée et la sauvegarde complète sous-jacente n'est pas supprimée.

Quota et limites

AWS Backup limite les sauvegardes à une sauvegarde simultanée par ressource. Par conséquent, les demandes de sauvegarde planifiées ou à la demande supplémentaires pour la ressource sont mises en file d'attente et ne démarrent qu'une fois la tâche de sauvegarde existante terminée. Si la tâche de sauvegarde n'est pas démarrée ou terminée dans la fenêtre de sauvegarde, la demande échoue. Pour plus d'informations sur AWS Backup les limites, consultez la section Limites AWS de sauvegarde dans le AWS Backup Developer Guide.

Lorsque vous créez une sauvegarde, vous pouvez exécuter jusqu'à quatre sauvegardes simultanées par compte. De même, vous pouvez exécuter une restauration simultanée par compte. Lorsque vous lancez plus de quatre tâches de sauvegarde simultanément, seules quatre tâches de sauvegarde sont lancées et les tâches restantes sont régulièrement réessayées. Une fois lancée, si la tâche de sauvegarde n'est pas terminée pendant la durée de la fenêtre de sauvegarde configurée, la tâche de sauvegarde échoue. Si la tâche de sauvegarde qui a échoué est une sauvegarde à la demande, vous pouvez réessayer la sauvegarde et, pour les sauvegardes planifiées, la tâche est tentée selon le calendrier suivant.

Clés de partition définies par le client

Amazon Timestream LiveAnalytics pour les clés de partition définies par le client est une fonctionnalité de Timestream qui permet aux clients de définir leurs propres clés de partition LiveAnalytics pour leurs tables. Le partitionnement est une technique utilisée pour distribuer les données sur plusieurs unités de stockage physiques, ce qui permet une extraction des données plus rapide et plus efficace. Grâce aux clés de partition définies par le client, les clients peuvent créer un schéma de partitionnement mieux adapté à leurs modèles de requêtes et à leurs cas d'utilisation.

Avec Timestream pour les clés de partition LiveAnalytics définies par le client, les clients peuvent choisir le nom d'une dimension comme clé de partition pour leurs tables. Cela permet une plus grande flexibilité dans la définition du schéma de partitionnement de leurs données. En sélectionnant la bonne clé de partition, les clients peuvent optimiser leur modèle de données, améliorer les performances de leurs requêtes et réduire la latence des requêtes.

Rubriques

Utilisation de clés de partition définies par le client

Si vous avez un modèle de requête bien défini avec des dimensions de cardinalité élevées et que vous avez besoin d'une faible latence de requête, un Timestream pour une clé de partition LiveAnalytics définie par le client peut être un outil utile pour améliorer votre modèle de données. Par exemple, si vous êtes une entreprise de vente au détail qui suit les interactions des clients sur votre site Web, les principaux modèles d'accès seront probablement basés sur l'identifiant du client et l'horodatage. En définissant l'ID client comme clé de partition, vos données peuvent être distribuées de manière uniforme, ce qui permet de réduire le temps de latence et d'améliorer au final l'expérience utilisateur.

Autre exemple : le secteur de la santé, où les appareils portables collectent des données de capteurs pour suivre les signes vitaux des patients. Le principal modèle d'accès serait basé sur l'identifiant de l'appareil et l'horodatage, avec une cardinalité élevée sur les deux dimensions. En définissant l'ID du périphérique comme clé de partition, vous pouvez optimiser l'exécution de vos requêtes et garantir des performances de requête soutenues à long terme.

En résumé, Timestream pour les clés de partition LiveAnalytics définies par le client est particulièrement utile lorsque vous avez un modèle de requête clair, des dimensions de cardinalité élevées et que vous avez besoin d'une faible latence pour vos requêtes. En définissant une clé de partition qui s'aligne sur votre modèle de requête, vous pouvez optimiser l'exécution de vos requêtes et garantir des performances durables à long terme.

Commencer à utiliser les clés de partition définies par le client

Dans la console, choisissez Tables et créez une nouvelle table. Vous pouvez également utiliser an SDK pour accéder à l'`CreateTable`action permettant de créer de nouvelles tables pouvant inclure une clé de partition définie par le client.

Création d'une table avec une clé de partition de type dimension

Vous pouvez utiliser les extraits de code suivants pour créer une table avec une clé de partition de type dimension.

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)
```

Vérification de la configuration du schéma de partitionnement

Vous pouvez vérifier le schéma de configuration d'une table pour le partitionnement de plusieurs manières. Dans la console, choisissez Bases de données et choisissez la table à vérifier. Vous pouvez également utiliser un SDK pour accéder à l'`DescribeTable`action.

Décrire une table avec une clé de partition

Vous pouvez utiliser les extraits de code suivants pour décrire une table avec une clé de partition.

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;
       }
   }
```

Voici un exemple de sortie.

1. La table possède une clé de partition de type dimension

   ```
   [{Type: DIMENSION,Name: hostId,EnforcementInRecord: OPTIONAL}]
   ```

2. La table contient un nom de mesure, un type de clé de partition

   ```
   [{Type: MEASURE,}]
   ```

3. Obtenir une clé de partition composite à partir d'une table créée sans spécifier de clé de partition composite

```
[{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;
       }
   }
```

Voici un exemple de sortie.

1. La table possède une clé de partition de type dimension

```
[PartitionKey(Type=DIMENSION, Name=hostId, EnforcementInRecord=OPTIONAL)]
```

2. La table contient un nom de mesure, un type de clé de partition

```
[PartitionKey(Type=MEASURE)]
```

3. Obtenir une clé de partition composite à partir d'une table créée sans spécifier de clé de partition composite retournera

```
[PartitionKey(Type=MEASURE)]
```

Go v1

```
    <tablistentry>
     <tabname> Go </tabname>
     <tabcontent>
      <programlisting language="go"></programlisting>
     </tabcontent>
    </tablistentry>
```

Voici un exemple de sortie.

```
{
  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
   }
```

Voici un exemple de sortie.

```
{
  "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)
```

Voici un exemple de sortie.

1. La table possède une clé de partition de type dimension

```
[{'CompositePartitionKey': [{'Type': 'DIMENSION', 'Name': 'hostId',
 'EnforcementInRecord': 'OPTIONAL'}]}]
```

2. La table contient un nom de mesure, un type de clé de partition

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

3. Obtenir une clé de partition composite à partir d'une table créée sans spécifier de clé de partition composite

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

Mise à jour de la configuration du schéma de partitionnement

Vous pouvez mettre à jour la configuration des tables pour le schéma de partitionnement à l'SDKaide d'une `UpdateTable` action avec accès.

Mettre à jour une table avec une clé de partition

Vous pouvez utiliser les extraits de code suivants pour mettre à jour une table avec une clé de partition.

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)
```

Avantages des clés de partition définies par le client

Performances de requête améliorées : les clés de partition définies par le client vous permettent d'optimiser l'exécution de vos requêtes et d'améliorer les performances globales des requêtes. En définissant des clés de partition qui correspondent à vos modèles de requêtes, vous pouvez minimiser l'analyse des données et optimiser l'élagage des données, ce qui réduit la latence des requêtes.

Meilleure prévisibilité des performances à long terme : les clés de partition définies par le client permettent aux clients de répartir les données de manière uniforme sur les partitions, améliorant ainsi l'efficacité de la gestion des données. Cela garantira que les performances de vos requêtes restent stables au fur et à mesure que les données stockées évoluent au fil du temps.

Limitations des clés de partition définies par le client

En tant que Timestream pour LiveAnalytics l'utilisateur, il est important de garder à l'esprit les limites liées à une clé de partition client. Tout d'abord, cela nécessite une bonne compréhension de votre charge de travail et de vos modèles de requêtes. Cela signifie que vous devez avoir une idée précise des dimensions les plus fréquemment utilisées comme principales conditions de filtrage dans les requêtes et avoir une cardinalité élevée pour tirer le meilleur parti des clés de partition.

Deuxièmement, les clés de partition doivent être définies au moment de la création de la table et ne peuvent pas être ajoutées aux tables existantes. Cela signifie que vous devez examiner attentivement votre stratégie de partitionnement avant de créer une table afin de vous assurer qu'elle correspond aux besoins de votre entreprise.

Enfin, il est important de noter qu'une fois la table créée, vous ne pouvez pas modifier la clé de partition par la suite. Cela signifie que vous devez tester et évaluer minutieusement votre stratégie de partitionnement avant de vous y engager. Compte tenu de ces limites, la clé de partition définie par le client de Timestream peut améliorer considérablement les performances des requêtes et la satisfaction à long terme.

Clés de partition définies par le client et faibles dimensions de cardinalité

Si vous décidez d'utiliser une clé de partition à très faible cardinalité, telle qu'une région ou un état spécifique, il est important de noter que les données relatives à d'autres entités`customerID`, telles que, et d'autres`ProductCategory`, peuvent finir par être réparties sur un trop grand nombre de partitions, parfois avec peu ou pas de données présentes. Cela peut entraîner une exécution inefficace des requêtes et une baisse des performances.

Pour éviter cela, nous vous recommandons de choisir des dimensions qui font non seulement partie de votre principale condition de filtrage, mais qui présentent également une cardinalité plus élevée. Cela permettra de garantir une répartition uniforme des données entre les partitions et d'améliorer les performances des requêtes.

Création de clés de partition pour les tables existantes

Si vous avez déjà des tables dans Timestream for LiveAnalytics et que vous souhaitez utiliser des clés de partition définies par le client, vous devez migrer vos données vers une nouvelle table avec la définition de schéma de partitionnement souhaitée. Cela peut être fait en exportant vers S3 et en chargeant par lots ensemble, ce qui implique d'exporter les données de la table existante vers S3,

de modifier les données pour inclure la clé de partition (si nécessaire) et d'ajouter des en-têtes à vos CSV fichiers, puis d'importer les données dans une nouvelle table avec le schéma de partitionnement souhaité défini. N'oubliez pas que cette méthode peut être longue et coûteuse, en particulier pour les grandes tables.

Vous pouvez également utiliser des requêtes planifiées pour migrer vos données vers une nouvelle table avec le schéma de partitionnement souhaité. Cette méthode implique la création d'une requête planifiée qui lit la table existante et écrit dans la nouvelle table. La requête planifiée peut être configurée pour s'exécuter régulièrement jusqu'à ce que toutes les données aient été migrées. N'oubliez pas que la lecture et l'écriture des données vous seront facturées pendant le processus de migration.

Timestream pour la validation LiveAnalytics du schéma avec des clés de partition composites personnalisées

La validation du schéma dans Timestream for LiveAnalytics permet de garantir que les données ingérées dans la base de données sont conformes au schéma spécifié, de minimiser les erreurs d'ingestion et d'améliorer la qualité des données. La validation du schéma est particulièrement utile lors de l'adoption d'une clé de partition définie par le client dans le but d'optimiser les performances de vos requêtes.

Qu'est-ce que Timestream pour la validation du LiveAnalytics schéma avec des clés de partition définies par le client ?

Le flux temporel pour la validation LiveAnalytics du schéma est une fonctionnalité qui valide les données ingérées dans une LiveAnalytics table Timestream pour la base d'un schéma prédéfini. Ce schéma définit le modèle de données, y compris la clé de partition, les types de données et les contraintes pour les enregistrements insérés.

Lorsque vous utilisez une clé de partition définie par le client, la validation du schéma devient encore plus cruciale. Les clés de partition vous permettent de spécifier une clé de partition, qui détermine la manière dont vos données sont stockées dans Timestream for. LiveAnalytics En validant les données entrantes par rapport au schéma à l'aide d'une clé de partition personnalisée, vous pouvez renforcer la cohérence des données, détecter les erreurs à un stade précoce et améliorer la qualité globale des données stockées dans Timestream for. LiveAnalytics

Comment utiliser Timestream pour la validation LiveAnalytics du schéma avec des clés de partition composites personnalisées

Pour utiliser Timestream pour la validation LiveAnalytics du schéma avec des clés de partition composites personnalisées, procédez comme suit :

Réfléchissez à ce à quoi ressembleront vos modèles de requête : pour choisir et définir correctement le schéma de votre Timestream for LiveAnalytics table, vous devez commencer par les exigences de vos requêtes.

Spécifiez des clés de partition composites personnalisées : lors de la création de la table, spécifiez une clé de partition personnalisée. Cette clé détermine l'attribut qui sera utilisé pour partitionner les données de la table. Vous pouvez choisir entre des clés de dimension et des clés de mesure pour le partitionnement. Une clé de dimension partitionne les données en fonction du nom d'une dimension, tandis qu'une clé de mesure partitionne les données en fonction du nom de la mesure.

Définissez des niveaux d'application : pour garantir un partitionnement des données approprié et les avantages qui en découlent, Amazon LiveAnalytics Timestream vous permet de définir des niveaux d'application pour chaque clé de partition de votre schéma. Le niveau d'application détermine si la dimension de la clé de partition est obligatoire ou facultative lors de l'ingestion d'enregistrements. Vous pouvez choisir entre deux options : `REQUIRED` ce qui signifie que la clé de partition doit être présente dans l'enregistrement ingéré et`OPTIONAL`, ce qui signifie que la clé de partition n'a pas besoin d'être présente. Il est recommandé d'utiliser le niveau d'`REQUIRED`application lorsque vous utilisez une partition définie par le client afin de garantir que vos données sont correctement partitionnées et que vous bénéficiez de tous les avantages de cette fonctionnalité. En outre, vous pouvez modifier la configuration du niveau d'application à tout moment après la création du schéma afin de l'adapter à vos exigences en matière d'ingestion de données.

Ingestion de données : lors de l'ingestion de données dans la LiveAnalytics table Timestream for, le processus de validation du schéma vérifie les enregistrements par rapport au schéma défini à l'aide de clés de partition composites personnalisées. Si les enregistrements ne respectent pas le schéma, Timestream for LiveAnalytics renverra une erreur de validation.

Gérer les erreurs de validation : en cas d'erreur de validation, Timestream for LiveAnalytics renverra a `ValidationException` ou a`RejectedRecordsException`, selon le type d'erreur. Assurez-vous de gérer ces exceptions dans votre application et de prendre les mesures appropriées, telles que la correction des enregistrements incorrects et une nouvelle tentative d'ingestion.

Mettre à jour les niveaux d'application : si nécessaire, vous pouvez mettre à jour le niveau d'application des clés de partition après la création de la table à l'aide de `UpdateTable` cette action. Cependant, il est important de noter que certains aspects de la configuration de la clé de partition, tels que le nom et le type, ne peuvent pas être modifiés après la création de la table. Si vous modifiez le niveau d'application de `REQUIRED` à`OPTIONAL`, tous les enregistrements seront acceptés indépendamment de la présence de l'attribut sélectionné comme clé de partition définie par le client. À l'inverse, si vous modifiez le niveau d'application de `OPTIONAL` à`REQUIRED`, vous pouvez commencer à voir des erreurs d'écriture 4xx pour les enregistrements qui ne répondent pas à cette condition. Il est donc essentiel de choisir le niveau d'application approprié à votre cas d'utilisation lors de la création de votre table, en fonction des exigences de partitionnement de vos données.

Quand utiliser Timestream pour la validation LiveAnalytics du schéma avec des clés de partition composites personnalisées

Le flux temporel pour la validation LiveAnalytics du schéma avec des clés de partition composites personnalisées doit être utilisé dans les scénarios où la cohérence, la qualité et le partitionnement optimisé des données sont essentiels. En appliquant un schéma lors de l'ingestion des données, vous pouvez éviter les erreurs et les incohérences susceptibles d'entraîner une analyse incorrecte ou la perte d'informations précieuses.

Interaction avec les tâches de chargement par lots

Lorsque vous configurez une tâche de chargement par lots pour importer des données dans une table avec une clé de partition définie par le client, plusieurs scénarios peuvent affecter le processus :

1. Si le niveau d'application est défini sur`OPTIONAL`, une alerte sera affichée sur la console pendant le flux de création si la clé de partition n'est pas mappée lors de la configuration de la tâche. Cette alerte ne s'affiche pas lorsque vous utilisez le API ouCLI.
2. Si le niveau d'application est défini sur`REQUIRED`, la création de tâche sera rejetée sauf si la clé de partition est mappée à une colonne de données source.
3. Si le niveau d'application est modifié `REQUIRED` après la création de la tâche, celle-ci continuera à s'exécuter, mais tous les enregistrements ne disposant pas du mappage approprié pour la clé de partition seront rejetés avec une erreur 4xx.

Interaction avec une requête planifiée

Lorsque vous configurez une tâche de requête planifiée pour calculer et stocker des agrégats, des cumuls et d'autres formes de données prétraitées dans une table avec une clé de partition définie par le client, certains scénarios peuvent affecter le processus :

1. Si le niveau d'application est défini sur`OPTIONAL`, une alerte s'affiche si la clé de partition n'est pas mappée lors de la configuration de la tâche. Cette alerte ne s'affiche pas lorsque vous utilisez le API ouCLI.
2. Si le niveau d'application est défini sur`REQUIRED`, la création de tâche sera rejetée sauf si la clé de partition est mappée à une colonne de données source.
3. Si le niveau d'application est modifié `REQUIRED` après la création de la tâche et que les résultats de la requête planifiée ne contiennent pas la dimension de clé de partition, toutes les itérations suivantes de la tâche échoueront.

Ajout d'identifications et d'étiquettes aux ressources

Vous pouvez étiqueter les ressources sur Amazon Timestream à l'aide LiveAnalytics de balises. Les balises vous permettent de classer vos ressources de différentes manières, par exemple en fonction de leur objectif, de leur propriétaire, de leur environnement ou d'autres critères. Les balises vous permettent d'effectuer les actions suivantes :

- Identifier rapidement une ressource en fonction des balises que vous lui avez attribuées.
- Consultez AWS les factures ventilées par tags.

Le balisage est pris en charge par AWS des services tels qu'Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream LiveAnalytics for, etc. Un balisage efficace peut fournir un aperçu sur les coûts en vous permettant de créer des rapports sur les services associés à une balise spécifique.

Pour commencer le balisage, procédez comme suit :

1. Comprenez les restrictions relatives au balisage.
2. Créez des balises à l'aide des opérations de balisage.

En dernier lieu, il est conseillé de suivre les politique de balisage optimales. Pour plus d'informations, consultez Politiques d'étiquetage AWS.

Restrictions de balisage

Chaque étiquette est constituée d'une clé et d'une valeur, que vous définissez. Les restrictions suivantes s'appliquent :

- Chaque flux temporel d'une LiveAnalytics table ne peut comporter qu'une seule balise avec la même clé. Si vous essayez d'ajouter une balise existante, la valeur de la balise existante est mise à jour avec la nouvelle valeur.
- Une valeur agit comme un descripteur au sein d'une catégorie de balises. Dans Timestream, LiveAnalytics la valeur ne peut pas être vide ou nulle.
- Les clés et valeurs de balise sont sensibles à la casse.
- La longueur de clé maximale est de 128 caractères Unicode.
- La longueur de valeur maximale est de 256 caractères Unicode.
- Les caractères autorisés sont les lettres, les espaces blancs et les chiffres, ainsi que les caractères spéciaux suivants : `+ - = . _ : /`
- Le nombre maximum d'identifications par ressource est de 50.
- AWS-les noms et valeurs des balises assignés reçoivent automatiquement le `aws:` préfixe, que vous ne pouvez pas attribuer. AWS-les noms de balises attribués ne sont pas pris en compte dans la limite de 50 balises. Les noms des balises attribuées par l'utilisateur ont le préfixe `user:` dans le rapport de répartition des coûts.
- Vous ne pouvez pas antidater l'application d'une balise.

Opérations de balisage

Vous pouvez ajouter, répertorier, modifier ou supprimer des balises pour les bases de données et les tables à l'aide d'Amazon Timestream LiveAnalytics pour console, du langage de requête ou AWS Command Line Interface du ().AWS CLI

Rubriques

- Ajout de balises à des bases de données et à des tables nouvelles ou existantes à l'aide de la console

Ajout de balises à des bases de données et à des tables nouvelles ou existantes à l'aide de la console

Vous pouvez utiliser la LiveAnalytics console Timestream pour ajouter des balises aux nouvelles bases de données, tables et requêtes planifiées lorsque vous les créez. Vous pouvez également ajouter, répertorier, modifier ou supprimer des balises pour les tables existantes.

Pour baliser les bases de données lors de leur création (console)

1. Ouvrez la console Timestream à l'adresse https://console.aws.amazon.com /timestream.
2. Dans le volet de navigation, choisissez Databases, puis Create database.
3. Sur la page Créer une base de données, attribuez un nom à la base de données. Entrez une clé et une valeur pour la balise, puis choisissez Ajouter une nouvelle balise.
4. Choisissez Créer une base de données.

Pour baliser les tables lors de leur création (console)

1. Ouvrez la console Timestream à l'adresse https://console.aws.amazon.com /timestream.
2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).
3. Sur la page Créer un flux temporel pour une LiveAnalytics table, attribuez un nom à la table. Entrez une clé et une valeur pour la balise, puis choisissez Ajouter une nouvelle balise.
4. Choisissez Créer un tableau.

Pour baliser les requêtes planifiées lors de leur création (console)

1. Ouvrez la console Timestream à l'adresse https://console.aws.amazon.com /timestream.
2. Dans le volet de navigation, choisissez Requêtes planifiées, puis choisissez Créer une requête planifiée.
3. À l'étape 3. Configurez la page des paramètres de requête, choisissez Ajouter une nouvelle balise. Saisissez une clé et une valeur pour la balise. Choisissez Ajouter un nouveau tag pour ajouter des tags supplémentaires.
4. Choisissez Suivant.

Pour baliser des ressources existantes (console)

1. Ouvrez la console Timestream à l'adresse https://console.aws.amazon.com /timestream.
2. Dans le volet de navigation, sélectionnez Bases de données, Tables ou Requêtes planifiées.
3. Choisissez une base de données ou une table dans la liste. Choisissez ensuite Gérer les balises pour ajouter, modifier ou supprimer vos balises.

Pour plus d'informations sur la structure des balises, consultez Restrictions de balisage.

Sécurité dans Timestream pour LiveAnalytics

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. L'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Timestream pour LiveAnalytics, consultez la section AWS Services concernés par programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre organisation, et la législation et la réglementation applicables.

Cette documentation vous aidera à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de Timestream pour. LiveAnalytics Les rubriques suivantes expliquent comment configurer Timestream pour atteindre vos objectifs LiveAnalytics de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui peuvent vous aider à surveiller et à sécuriser votre Timestream pour les LiveAnalytics ressources.

Rubriques

- Protection des données dans Timestream pour LiveAnalytics

- Gestion des identités et des accès pour Amazon Timestream pour LiveAnalytics
- Enregistrement et surveillance dans Timestream pour LiveAnalytics
- Résilience dans Amazon Timestream Live Analytics
- Sécurité de l'infrastructure dans Amazon Timestream Live Analytics
- Analyse de configuration et de vulnérabilité dans Timestream
- Réponse aux incidents dans Timestream pour LiveAnalytics
- VPCpoints de terminaison ()AWS PrivateLink
- Bonnes pratiques de sécurité pour Amazon Timestream pour LiveAnalytics

Protection des données dans Timestream pour LiveAnalytics

Le modèle de responsabilité AWS partagée s'applique à la protection des données dans Amazon Timestream Live Analytics. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la section Confidentialité des données FAQ. Pour plus d'informations sur la protection des données en Europe, consultez le modèle de responsabilitéAWS partagée et le billet de GDPR blog sur le blog sur la AWS sécurité.

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) pour chaque compte.
- UtilisezSSL/TLSpour communiquer avec les AWS ressources. Nous avons besoin de la TLS version 1.2 et recommandons la TLS version 1.3.
- Configuration API et journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.

- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de FIPS 140 à 3 modules cryptographiques validés pour accéder AWS via une interface de ligne de commande ou unAPI, utilisez un point de terminaison. FIPS Pour plus d'informations sur les FIPS points de terminaison disponibles, voir Federal Information Processing Standard (FIPS) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Timestream Live Analytics ou autre à Services AWS l'aide de la console, API AWS CLI, ou. AWS SDKs Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez un URL à un serveur externe, nous vous recommandons vivement de ne pas y inclure d'informations d'identification URL pour valider votre demande auprès de ce serveur.

Pour obtenir des informations plus détaillées sur Timestream concernant des sujets liés à la protection des LiveAnalytics données tels que le chiffrement au repos et la gestion des clés, sélectionnez l'un des sujets disponibles ci-dessous.

Rubriques

- Chiffrement au repos
- Chiffrement en transit
- Gestion des clés

Chiffrement au repos

Timestream pour le LiveAnalytics chiffrement au repos améliore la sécurité en chiffrant toutes vos données au repos à l'aide des clés de chiffrement stockées dans AWS Key Management Service ().AWS KMS Cette fonctionnalité réduit la lourdeur opérationnelle et la complexité induites par la protection des données sensibles. Le chiffrement au repos vous permet de créer des applications sensibles en matière de sécurité qui sont conformes aux exigences réglementaires et de chiffrement strictes.

- Le chiffrement est activé par défaut sur votre LiveAnalytics base de données Timestream for et ne peut pas être désactivé. L'algorithme de chiffrement standard AES -256 est l'algorithme de chiffrement par défaut utilisé.
- AWS KMS est requis pour le chiffrement au repos dans Timestream pour. LiveAnalytics
- Vous ne pouvez pas chiffrer uniquement un sous-ensemble d'éléments dans une table.
- Vous n'avez pas besoin de modifier vos applications clientes de base de données pour utiliser le chiffrement.

Si vous ne fournissez pas de clé, Timestream for LiveAnalytics crée et utilise une AWS KMS clé nommée `alias/aws/timestream` dans votre compte.

Vous pouvez utiliser votre propre clé gérée par le client KMS pour chiffrer les données de votre Timestream. LiveAnalytics Pour plus d'informations sur les touches dans Timestream for LiveAnalytics, voir. Gestion des clés

Timestream for LiveAnalytics stocke vos données sur deux niveaux de stockage, mémoire et stockage magnétique. Les données de la mémoire sont cryptées à l'aide d'une clé de LiveAnalytics service Timestream for. Les données du magasin magnétique sont cryptées à l'aide de votre AWS KMS clé.

Le service Timestream Query nécessite des informations d'identification pour accéder à vos données. Ces informations d'identification sont cryptées à l'aide de votre KMS clé.

> **Note**
>
> Timestream for LiveAnalytics n'appelle pas toutes les opérations AWS KMS de déchiffrement. Au lieu de cela, il conserve un cache local de clés pendant 5 minutes avec du trafic actif. Toute modification d'autorisation est propagée par le biais du Timestream pour le LiveAnalytics système avec une cohérence finale dans un délai maximum de 5 minutes.

Chiffrement en transit

Toutes vos données Timestream Live Analytics sont cryptées pendant leur transfert. Par défaut, toutes les communications à destination et en provenance de Timestream pour LiveAnalytics sont protégées à l'aide du chiffrement Transport Layer Security (TLS).

Gestion des clés

Vous pouvez gérer les clés pour Amazon Timestream Live Analytics à l'aide AWS du service de gestion des clés ().AWS KMS Timestream Live Analytics nécessite l'utilisation de KMS pour crypter vos données. Vous disposez des options suivantes pour la gestion des clés, en fonction du niveau de contrôle dont vous avez besoin sur vos clés :

Ressources de base de données et de tables

- Clé gérée par Timestream Live Analytics : si vous ne fournissez pas de clé, Timestream Live Analytics en créera une en utilisant. `alias/aws/timestream` KMS
- Clé gérée par le KMS client : les clés gérées par le client sont prises en charge. Choisissez cette option si vous souhaitez mieux contrôler les autorisations et le cycle de vie de vos clés, notamment la possibilité de les faire alterner automatiquement sur une base annuelle.

Ressource de requête planifiée

- Clé appartenant à Timestream Live Analytics : si vous ne fournissez pas de clé, Timestream Live Analytics utilisera sa propre clé pour chiffrer la ressource Query. Cette KMS clé est présente dans le compte Timestream. Consultez les clés AWS détenues dans le guide du KMS développeur pour plus de détails.
- Clé gérée par le KMS client : les clés gérées par le client sont prises en charge. Choisissez cette option si vous souhaitez mieux contrôler les autorisations et le cycle de vie de vos clés, notamment la possibilité de les faire alterner automatiquement sur une base annuelle.

KMSles clés d'un magasin de clés externe (XKS) ne sont pas prises en charge.

Gestion des identités et des accès pour Amazon Timestream pour LiveAnalytics

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. IAMles administrateurs contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser Timestream pour les ressources. LiveAnalytics IAMest un Service AWS ventilateur que vous pouvez utiliser sans frais supplémentaires.

Rubriques

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans Timestream pour. LiveAnalytics

Utilisateur du service : si vous utilisez le Timestream for LiveAnalytics Service pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez Timestream pour les LiveAnalytics fonctionnalités dans le cadre de votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans Timestream pour LiveAnalytics, consultez. Résolution des problèmes liés à Amazon Timestream en matière LiveAnalytics d'identité et d'accès

Administrateur du service — Si vous êtes responsable de Timestream pour les LiveAnalytics ressources de votre entreprise, vous avez probablement un accès complet à Timestream pour. LiveAnalytics C'est à vous de déterminer le flux temporel des LiveAnalytics fonctionnalités et des ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite envoyer des demandes à votre IAM administrateur pour modifier les autorisations des utilisateurs de votre service. Consultez les informations de cette page pour comprendre les concepts de base deIAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM Timestream pour LiveAnalytics, voir. Comment fonctionne Amazon Timestream for LiveAnalytics avec IAM

IAMadministrateur — Si vous êtes IAM administrateur, vous souhaiterez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à Timestream pour. LiveAnalytics Pour voir un exemple de Timestream pour les politiques LiveAnalytics basées sur l'identité que vous pouvez utiliser dans, voir. IAM Amazon Timestream LiveAnalytics pour des exemples de politiques basées sur l'identité

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant que Utilisateur racine d'un compte AWS, en tant qu'IAMutilisateur ou en assumant un IAM rôle.

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAMIdentity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez en tant qu'identité fédérée, votre administrateur a préalablement configuré la fédération d'identité à l'aide de IAM rôles. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la version 4 de AWS Signature pour les API demandes dans le guide de IAM l'utilisateur.

Quelle que soit la méthode d’authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, voir Authentification multifactorielle dans le guide de l'AWS IAM Identity Center utilisateur et Authentification AWS multifactorielle IAM dans le guide de l'IAMutilisateur.

Utilisateurs et groupes IAM

Un IAMutilisateur est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des IAM utilisateurs dotés d'informations d'identification à long terme, telles que des mots de passe et des clés d'accès. Toutefois, si vous avez des cas d'utilisation spécifiques qui nécessitent des informations d'identification à long terme auprès des IAM utilisateurs, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, voir Rotation régulière des clés d'accès

pour les cas d'utilisation nécessitant des informations d'identification à long terme dans le Guide de IAM l'utilisateur.

Un IAMgroupe est une identité qui définit un ensemble d'IAMutilisateurs. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer IAM des ressources.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez la section Cas d'utilisation pour IAM les utilisateurs dans le Guide de IAM l'utilisateur.

IAMrôles

Un IAMrôle est une identité au sein de Compte AWS vous dotée d'autorisations spécifiques. Il est similaire à un IAM utilisateur, mais n'est pas associé à une personne en particulier. Pour assumer temporairement un IAM rôle dans le AWS Management Console, vous pouvez passer d'un rôle d'utilisateur à un IAM rôle (console). Vous pouvez assumer un rôle en appelant une AWS API opération AWS CLI or ou en utilisant une option personnaliséeURL. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez la section Méthodes pour assumer un rôle dans le Guide de IAM l'utilisateur.

IAMles rôles dotés d'informations d'identification temporaires sont utiles dans les situations suivantes :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour plus d'informations sur les rôles pour la fédération, voir Création d'un rôle pour un fournisseur d'identité tiers dans le guide de IAM l'utilisateur. Si vous utilisez IAM Identity Center, vous configurez un ensemble d'autorisations. Pour contrôler les accès auxquels vos identités peuvent accéder après leur authentification, IAM Identity Center met en corrélation l'ensemble d'autorisations avec un rôle dans. IAM Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations IAM utilisateur temporaires : un IAM utilisateur ou un rôle peut assumer un IAM rôle afin d'obtenir temporairement différentes autorisations pour une tâche spécifique.

- Accès entre comptes : vous pouvez utiliser un IAM rôle pour autoriser une personne (un mandant fiable) d'un autre compte à accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour connaître la différence entre les rôles et les politiques basées sur les ressources pour l'accès entre comptes, voir Accès aux ressources entre comptes IAM dans le guide de l'IAMutilisateur.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès transmises (FAS) — Lorsque vous utilisez un IAM utilisateur ou un rôle pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FASutilise les autorisations du principal appelant an Service AWS, combinées à la demande Service AWS pour adresser des demandes aux services en aval. FASles demandes ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur les politiques relatives FAS aux demandes, consultez la section Transférer les sessions d'accès.
  - Rôle de service — Un rôle de service est un IAMrôle qu'un service assume pour effectuer des actions en votre nom. Un IAM administrateur peut créer, modifier et supprimer un rôle de service de l'intérieurIAM. Pour plus d'informations, consultez la section Création d'un rôle auquel déléguer des autorisations Service AWS dans le Guide de IAM l'utilisateur.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés au service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un IAM administrateur peut consulter, mais pas modifier les autorisations pour les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un IAM rôle pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui font AWS CLI des AWS API demandes. Cela est préférable au stockage des clés d'accès dans l'EC2instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l'EC2instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez la section Utilisation

d'un IAM rôle pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le Guide de IAM l'utilisateur.

Gestion des accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de JSON documents. Pour plus d'informations sur la structure et le contenu des documents de JSON politique, voir Présentation des JSON politiques dans le guide de IAM l'utilisateur.

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C’est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d’aucune autorisation. Pour autoriser les utilisateurs à effectuer des actions sur les ressources dont ils ont besoin, un IAM administrateur peut créer des IAM politiques. L'administrateur peut ensuite ajouter les IAM politiques aux rôles, et les utilisateurs peuvent assumer les rôles.

IAMles politiques définissent les autorisations pour une action, quelle que soit la méthode que vous utilisez pour effectuer l'opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle auprès du AWS Management Console AWS CLI, ou du AWS API.

Politiques basées sur l’identité

Les politiques basées sur l'identité sont JSON des documents de politique d'autorisation que vous pouvez joindre à une identité, telle qu'un IAM utilisateur, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour savoir comment créer une politique basée sur l'identité, voir Définir des IAM autorisations personnalisées avec des politiques gérées par le client dans le Guide de l'IAMutilisateur.

Les politiques basées sur l’identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou

rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour savoir comment choisir entre une politique gérée ou une politique intégrée, voir Choisir entre les politiques gérées et les politiques intégrées dans le Guide de l'IAMutilisateur.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents JSON de stratégie que vous attachez à une ressource. Les politiques de confiance dans les IAM rôles et les politiques relatives aux compartiments Amazon S3 sont des exemples de politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser de politiques AWS gérées depuis une IAM stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLssont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format du document JSON de stratégie.

Amazon S3 et Amazon VPC sont des exemples de services compatiblesACLs. AWS WAF Pour en savoir plusACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limites d'autorisations — Une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une IAM entité (IAMutilisateur ou rôle). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur

l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, voir Limites d'autorisations pour les IAM entités dans le Guide de IAM l'utilisateur.

- Politiques de contrôle des services (SCPs) : SCPs JSON politiques qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Les SCP limites d'autorisations pour les entités présentes dans les comptes des membres, y compris chacune d'entre elles Utilisateur racine d'un compte AWS. Pour plus d'informations sur les OrganizationsSCPs, voir Politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez la section Politiques de session dans le guide de IAM l'utilisateur.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de IAM l'utilisateur.

Comment fonctionne Amazon Timestream for LiveAnalytics avec IAM

Avant de gérer l'IAMaccès à Timestream pour LiveAnalytics, vous devez connaître les IAM fonctionnalités disponibles avec Timestream for. LiveAnalytics Pour obtenir une vue d'ensemble du fonctionnement de Timestream for LiveAnalytics et AWS des autres servicesIAM, consultez la section AWS Services qui fonctionnent avec IAM dans le guide de l'IAMutilisateur.

Rubriques

- Timestream pour les politiques basées sur l'identité LiveAnalytics
- Timestream pour les politiques basées sur les ressources LiveAnalytics
- Autorisation basée sur le Timestream pour les tags LiveAnalytics
- Diffusion chronologique des rôles LiveAnalytics IAM

Timestream pour les politiques basées sur l'identité LiveAnalytics

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier les actions et les ressources autorisées ou refusées ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Timestream for LiveAnalytics prend en charge des actions et des ressources spécifiques, ainsi que des clés de condition. Pour en savoir plus sur tous les éléments que vous utilisez dans une JSON politique, consultez la section Référence des éléments de IAM JSON stratégie dans le guide de IAM l'utilisateur.

Actions

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'`Action`élément d'une JSON politique décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès dans une politique. Les actions de stratégie portent généralement le même nom que l' AWS APIopération associée. Il existe certaines exceptions, telles que les actions avec autorisation uniquement qui n'ont pas d'opération correspondante. API Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Vous pouvez spécifier les actions suivantes dans l'élément Action d'une déclaration de IAM politique. Utilisez des politiques pour accorder des autorisations permettant d'effectuer une opération dansAWS. Lorsque vous utilisez une action dans une politique, vous autorisez ou refusez généralement l'accès à l'APIopération, à la CLI commande ou à la SQL commande portant le même nom.

Dans certains cas, une seule action contrôle l'accès à une API opération ainsi qu'à une SQL commande. D'autres opérations, quant à elles, requièrent plusieurs actions différentes.

Pour obtenir la liste des options Timestream prises en charge, consultez le tableau ci-dessous : LiveAnalytics `Action`

> **Note**
>
> Pour toutes les données spécifiques à une base de données`Actions`, vous pouvez spécifier une base de données afin de limiter l'action ARN à une base de données particulière.

Actions	Description	Niveau d'accès	Types de ressources (*obligatoire)
DescribeEndpoints	Renvoie le point de terminaison Timestream auquel les demandes suivantes doivent être adressées.	Tous	*
Select	Exécutez des requêtes sur Timestream qui sélectionnent les données d'une ou de plusieurs tables. Voir cette note pour une explication détaillée	Lecture	tableau*
CancelQuery	Annulez une requête.	Lecture	*
ListTables	Obtenez la liste des tables.	Liste	base de données*
ListDatabases	Obtenez la liste des bases de données.	Liste	*
ListMeasures	Obtenez la liste des mesures.	Lecture	tableau*

Actions	Description	Niveau d'accès	Types de ressources (*obligatoire)
DescribeTable	Obtenez la description du tableau.	Lecture	tableau*
DescribeDatabase	Obtenez la description de la base de données.	Lecture	base de données*
SelectValues	Exécutez des requêtes qui ne nécessitent pas la spécification d'une ressource particulière. Consultez cette note pour une explication détaillée.	Lecture	*
WriteRecords	Insérez des données dans Timestream.	Écrire	tableau*
CreateTable	Créer une table .	Écrire	base de données*
CreateDatabase	Créez une base de données.	Écrire	*
DeleteDatabase	Supprimez une base de données.	Écrire	*
UpdateDatabase	Mettez à jour une base de données.	Écrire	*
DeleteTable	Supprimez un tableau.	Écrire	base de données*
UpdateTable	Mettez à jour un tableau.	Écrire	base de données*

SelectValues contre sélectionner :

`SelectValues`est un `Action` qui est utilisé pour les requêtes ne nécessitant aucune ressource. Voici un exemple de requête ne nécessitant aucune ressource :

```
SELECT 1
```

Notez que cette requête ne fait pas référence à un flux temporel spécifique pour LiveAnalytics une ressource. Prenons un autre exemple :

```
SELECT now()
```

Cette requête renvoie l'horodatage actuel à l'aide de la `now()` fonction, mais aucune ressource ne doit être spécifiée. `SelectValues`est souvent utilisé pour les tests, de sorte que Timestream for LiveAnalytics peut exécuter des requêtes sans ressources. Maintenant, considérez une `Select` requête :

```
SELECT * FROM database.table
```

Ce type de requête nécessite une ressource, en particulier un flux temporel pour LiveAnalytics `table`, afin que les données spécifiées puissent être extraites de la table.

Ressources

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Resource` JSON de stratégie indique le ou les objets auxquels s'applique l'action. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de spécifier une ressource en utilisant son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

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

Dans Timestream pour les LiveAnalytics bases de données et les tables, vous pouvez utiliser l'`Resource`IAMélément autorisations.

La ressource Timestream pour la LiveAnalytics base de données est la suivante : ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}
```

La ressource Timestream for LiveAnalytics table est la suivante : ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}/table/
${TableName}
```

Pour plus d'informations sur le format deARNs, consultez Amazon Resource Names (ARNs) et AWS Service Namespaces.

Par exemple, pour spécifier le `database` keyspace dans votre relevé, utilisez ce qui suit : ARN

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/mydatabase"
```

Pour spécifier toutes les bases de données appartenant à un compte spécifique, utilisez le caractère générique (*) :

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/*"
```

Certaines diffusions temporelles pour des LiveAnalytics actions, telles que celles relatives à la création de ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

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

Clés de condition

Timestream for LiveAnalytics ne fournit aucune clé de condition spécifique à un service, mais il prend en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, voir Clés contextuelles de condition AWS globale dans le guide de IAM l'utilisateur.

Exemples

Pour consulter des exemples de Timestream pour les politiques LiveAnalytics basées sur l'identité, voir. Amazon Timestream LiveAnalytics pour des exemples de politiques basées sur l'identité

Timestream pour les politiques basées sur les ressources LiveAnalytics

Timestream for LiveAnalytics ne prend pas en charge les politiques basées sur les ressources. Pour afficher un exemple de page de stratégie basée sur les ressources détaillée, consultez https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Autorisation basée sur le Timestream pour les tags LiveAnalytics

Vous pouvez gérer l'accès à votre Timestream pour les LiveAnalytics ressources à l'aide de balises. Pour gérer l'accès aux ressources basé sur des balises, vous devez fournir les informations de balise dans l'élément Condition d'une stratégie utilisant les clés de condition `timestream:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Pour plus d'informations sur le balisage de Timestream pour les LiveAnalytics ressources, consultez. the section called “Balisage de ressources ”

Pour visualiser un exemple de stratégie basée sur l'identité permettant de limiter l'accès à une ressource en fonction des balises de cette ressource, veuillez consulter Timestream pour l'accès aux LiveAnalytics ressources en fonction des balises.

Diffusion chronologique des rôles LiveAnalytics IAM

Un IAMrôle est une entité de votre AWS compte dotée d'autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec Timestream pour LiveAnalytics

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à la fédération, assumer un IAM rôle ou assumer un rôle entre comptes. Vous obtenez des informations d'identification de sécurité temporaires en appelant AWS STS API des opérations telles que AssumeRoleou GetFederationToken.

Rôles liés à un service

Timestream for LiveAnalytics ne prend pas en charge les rôles liés à un service.

Rôles de service

Timestream for LiveAnalytics ne prend pas en charge les rôles de service.

AWS politiques gérées pour Amazon Timestream Live Analytics

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou que de nouvelles API opérations sont disponibles pour les services existants.

Pour plus d'informations, consultez la section Politiques AWS gérées dans le Guide de IAM l'utilisateur.

Rubriques

- AWS politique gérée : AmazonTimestreamReadOnlyAccess
- AWS politique gérée : AmazonTimestreamConsoleFullAccess
- AWS politique gérée : AmazonTimestreamFullAccess
- Mises à jour des politiques gérées par Timestream Live Analytics AWS

AWS politique gérée : AmazonTimestreamReadOnlyAccess

Vous pouvez vous associer `AmazonTimestreamReadOnlyAccess` à vos utilisateurs, groupes et rôles. La politique fournit un accès en lecture seule à Amazon Timestream.

Détails de l’autorisation

Cette politique inclut l'autorisation suivante :

- `Amazon Timestream`— Fournit un accès en lecture seule à Amazon Timestream. Cette politique accorde également l'autorisation d'annuler toute requête en cours d'exécution.

Pour consulter cette politique dans son JSON format, voir AmazonTimestreamReadOnlyAccess.

AWS politique gérée : AmazonTimestreamConsoleFullAccess

Vous pouvez vous associer `AmazonTimestreamConsoleFullAccess` à vos utilisateurs, groupes et rôles.

La politique fournit un accès complet pour gérer Amazon Timestream à l'aide du. AWS Management Console Cette politique accorde également des autorisations pour certaines AWS KMS opérations et pour gérer vos requêtes enregistrées.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `Amazon Timestream`— Accorde aux principaux un accès complet à Amazon Timestream.
- `AWS KMS`— Permet aux principaux de répertorier les alias et de décrire les clés.
- `Amazon S3`— Permet aux principaux de répertorier tous les compartiments Amazon S3.
- `Amazon SNS`— Permet aux directeurs de répertorier les SNS sujets Amazon.
- `IAM`— Permet aux principaux de répertorier les IAM rôles.
- `DBQMS` : permet aux principaux d'accéder, de créer, de supprimer, de décrire et de mettre à jour des requêtes. Le service de métadonnées de requête de base de données (dbqms) est un service interne uniquement. Il fournit vos requêtes récentes et enregistrées pour l'éditeur de requêtes sur le AWS Management Console for multiple Services AWS, y compris Amazon Timestream.

Pour consulter cette politique dans son JSON format, voir AmazonTimestreamConsoleFullAccess.

AWS politique gérée : AmazonTimestreamFullAccess

Vous pouvez vous associer `AmazonTimestreamFullAccess` à vos utilisateurs, groupes et rôles.

La politique fournit un accès complet à Amazon Timestream. Cette politique accorde également des autorisations pour certaines AWS KMS opérations.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `Amazon Timestream`— Accorde aux principaux un accès complet à Amazon Timestream.
- `AWS KMS`— Permet aux principaux de répertorier les alias et de décrire les clés.
- `Amazon S3`— Permet aux principaux de répertorier tous les compartiments Amazon S3.

Pour consulter cette politique dans son JSON format, voir AmazonTimestreamFullAccess.

Mises à jour des politiques gérées par Timestream Live Analytics AWS

Consultez les détails des mises à jour des politiques AWS gérées pour Timestream Live Analytics depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au RSS flux sur la page d'historique des documents Timestream Live Analytics.

Modification	Description	Date
AmazonTimestreamReadOnlyAccess – Mise à jour d’une stratégie existante	L'`timestream:DescribeAccountSettings` action a été ajoutée à la politique `AmazonTimestreamReadOnlyAccess` gérée existante. Cette action est utilisée pour décrire Compte AWS les paramètres. Timestream Live Analytics a également mis à jour cette politique gérée en ajoutant un `Sid` champ. La mise à jour de la politique n'a aucun impact sur l'utilisation de la politique `AmazonTimestreamReadOnlyAccess` gérée.	3 juin 2024

Modification	Description	Date
AmazonTimestreamReadOnlyAccess – Mise à jour d'une politique existante	Les `timestream:ListBatchLoadTasks` actions `timestream:DescribeBatchLoadTask` et ont été ajoutées à la politique `AmazonTimestreamReadOnlyAccess` gérée existante. Ces actions sont utilisées lors de la liste et de la description des tâches de chargement par lots. La mise à jour de la politique n'a aucun impact sur l'utilisation de la politique `AmazonTimestreamReadOnlyAccess` gérée.	24 février 2023
AmazonTimestreamReadOnlyAccess – Mise à jour d'une politique existante	Les `timestream:ListScheduledQueries` actions `timestream:DescribeScheduledQuery` et ont été ajoutées à la politique `AmazonTimestreamReadOnlyAccess` gérée existante. Ces actions sont utilisées lors de la liste et de la description des requêtes planifiées existantes. La mise à jour de la politique n'a aucun impact sur l'utilisation de la politique `AmazonTimestreamReadOnlyAccess` gérée.	29 novembre 2021

Modification	Description	Date
AmazonTimestreamConsoleFullAccess – Mise à jour d'une politique existante	L'`s3:ListAllMyBuckets` action a été ajoutée à la politique `AmazonTimestreamConsoleFullAccess` gérée existante. Cette action est utilisée lorsque vous spécifiez un compartiment Amazon S3 pour Timestream afin de consigner les erreurs d'écriture dans le magasin magnétique. La mise à jour de la politique n'a aucun impact sur l'utilisation de la politique `AmazonTimestreamConsoleFullAccess` gérée.	29 novembre 2021
AmazonTimestreamFullAccess – Mise à jour d'une politique existante	L'`s3:ListAllMyBuckets` action a été ajoutée à la politique `AmazonTimestreamFullAccess` gérée existante. Cette action est utilisée lorsque vous spécifiez un compartiment Amazon S3 pour Timestream afin de consigner les erreurs d'écriture dans le magasin magnétique. La mise à jour de la politique n'a aucun impact sur l'utilisation de la politique `AmazonTimestreamFullAccess` gérée.	29 novembre 2021

Modification	Description	Date
AmazonTimestreamConsoleFullAccess – Mise à jour d'une politique existante	Les actions redondantes ont été supprimées de la politique `AmazonTimestreamConsoleFullAccess` gérée existante. Auparavant, cette politique incluait une action `dbqms:DescribeQueryHistory` redondante. La politique mise à jour supprime l'action redondante. La mise à jour de la politique n'a aucun impact sur l'utilisation de la politique `AmazonTimestreamConsoleFullAccess` gérée.	23 avril 2021
Timestream Live Analytics a commencé à suivre les modifications	Timestream Live Analytics a commencé à suivre les modifications apportées à ses politiques AWS gérées.	21 avril 2021

Amazon Timestream LiveAnalytics pour des exemples de politiques basées sur l'identité

Par défaut, IAM les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier Timestream pour les LiveAnalytics ressources. Ils ne peuvent pas non plus effectuer de tâches à l'aide du AWS Management Console CQLSH AWS CLI,, ou AWS API. Un IAM administrateur doit créer des IAM politiques qui accordent aux utilisateurs et aux rôles l'autorisation d'effectuer des API opérations spécifiques sur les ressources spécifiques dont ils ont besoin. L'administrateur doit ensuite associer ces politiques aux IAM utilisateurs ou aux groupes qui ont besoin de ces autorisations.

Pour savoir comment créer une politique IAM basée sur l'identité à l'aide de ces exemples de documents de JSON stratégie, voir Création de politiques dans l'JSONonglet du guide de l'IAMutilisateur.

Rubriques

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer Timestream pour les LiveAnalytics ressources de votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez les politiques AWS gérées ou les politiques AWS gérées pour les fonctions professionnelles dans le Guide de IAM l'utilisateur.
- Appliquer les autorisations du moindre privilège : lorsque vous définissez des autorisations à IAM l'aide de politiques, accordez uniquement les autorisations nécessaires à l'exécution d'une tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation IAM pour appliquer des autorisations, consultez la section Politiques et autorisations du Guide de IAM l'utilisateur. IAM
- Utilisez des conditions dans IAM les politiques pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques pour limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez rédiger une condition de politique pour spécifier que toutes les demandes doivent être envoyées en utilisantSSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, voir Éléments IAM JSON de politique : Condition dans le guide de IAM l'utilisateur.

- Utilisez IAM Access Analyzer pour valider vos IAM politiques afin de garantir des autorisations sécurisées et fonctionnelles. IAM Access Analyzer valide les politiques nouvelles et existantes afin qu'elles soient conformes au langage des IAM politiques (JSON) et IAM aux meilleures pratiques. IAMAccess Analyzer fournit plus de 100 vérifications des politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez la section Valider les politiques avec IAM Access Analyzer dans le guide de l'IAMutilisateur.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des IAM utilisateurs ou un utilisateur root Compte AWS, activez-le MFA pour une sécurité supplémentaire. Pour exiger le MFA moment où les API opérations sont appelées, ajoutez MFA des conditions à vos politiques. Pour plus d'informations, consultez la section APIAccès sécurisé avec MFA dans le guide de IAM l'utilisateur.

Pour plus d'informations sur les meilleures pratiques en matière de sécuritéIAM, consultez la section Bonnes pratiques en matière de sécurité IAM dans le Guide de IAM l'utilisateur.

Utilisation du Timestream pour console LiveAnalytics

Timestream for LiveAnalytics ne nécessite pas d'autorisations spécifiques pour accéder à Amazon Timestream pour console. LiveAnalytics Vous devez disposer d'au moins des autorisations en lecture seule pour répertorier et afficher les informations relatives au flux temporel des LiveAnalytics ressources de votre compte. AWS Si vous créez une politique basée sur l'identité qui est plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les entités (IAMutilisateurs ou rôles) dotées de cette politique.

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux IAM utilisateurs de consulter les politiques intégrées et gérées associées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide du 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": "*"
        }
    ]
}
```

Opérations courantes dans Timestream pour LiveAnalytics

Vous trouverez ci-dessous des exemples de IAM politiques qui autorisent les opérations courantes dans le Timestream for LiveAnalytics Service.

Rubriques

- Autoriser toutes les opérations
- Autoriser SELECT les opérations
- Autoriser SELECT les opérations sur plusieurs ressources
- Autoriser les opérations de métadonnées
- Autoriser INSERT les opérations
- Autoriser CRUD les opérations
- Annuler les requêtes et sélectionner les données sans spécifier de ressources
- Création, description, suppression et description d'une base de données

- Limiter les bases de données répertoriées par tag {"Owner": "${username}"}
- Répertorier toutes les tables d'une base de données
- Création, description, suppression, mise à jour et sélection dans un tableau
- Limiter une requête par table

Autoriser toutes les opérations

Voici un exemple de politique qui autorise toutes les opérations dans Timestream pour. LiveAnalytics

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Autoriser SELECT les opérations

L'exemple de politique suivant autorise les requêtes de `SELECT` type -style sur une ressource spécifique.

> **Note**
>
> `<account_ID>`Remplacez-le par votre identifiant de compte 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": "*"
        }
    ]
}
```

Autoriser SELECT les opérations sur plusieurs ressources

L'exemple de politique suivant autorise les requêtes de type « `SELECT` -style » sur plusieurs ressources.

> **Note**
>
> `<account_ID>`Remplacez-le par votre identifiant de compte 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": "*"
        }
    ]
}
```

Autoriser les opérations de métadonnées

L'exemple de politique suivant permet à l'utilisateur d'effectuer des requêtes de métadonnées, mais ne l'autorise pas à effectuer des opérations de lecture ou d'écriture de données réelles dans 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": "*"
        }
    ]
}
```

Autoriser INSERT les opérations

L'exemple de politique suivant permet à un utilisateur d'effectuer une `INSERT` opération sur `database/sampleDB/table/DevOps` un compte`<account_id>`.

> **Note**
>
> `<account_ID>`Remplacez-le par votre identifiant de compte 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"
        }
    ]
}
```

Autoriser CRUD les opérations

L'exemple de politique suivant permet à un utilisateur d'effectuer des CRUD opérations dans Timestream pour. 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": "*"
        }
    ]
}
```

Annuler les requêtes et sélectionner les données sans spécifier de ressources

L'exemple de politique suivant permet à un utilisateur d'annuler des requêtes et d'exécuter des `Select` requêtes sur des données qui ne nécessitent pas de spécification de ressource :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:SelectValues",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Création, description, suppression et description d'une base de données

L'exemple de politique suivant permet à un utilisateur de créer, de décrire, de supprimer et de décrire une base de données `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"
        }
    ]
}
```

Limiter les bases de données répertoriées par tag **`{"Owner": "${username}"}`**

L'exemple de politique suivant permet à un utilisateur de répertorier toutes les bases de données étiquetées avec une paire clé-valeur `{"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}"
                }
            }
        }
    ]
}
```

Répertorier toutes les tables d'une base de données

Exemple de politique suivant pour répertorier toutes les tables de la base de données `sampleDB` :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:ListTables"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/"
        }
    ]
}
```

Création, description, suppression, mise à jour et sélection dans un tableau

L'exemple de politique suivant permet à un utilisateur de créer des tables, de décrire des tables, de supprimer des tables, de mettre à jour des tables et d'effectuer des `Select` requêtes sur une table `DevOps` dans une base de données `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"
        }
    ]
}
```

Limiter une requête par table

L'exemple de politique suivant permet à un utilisateur d'interroger toutes les tables sauf celles de `DevOps` la base de données `sampleDB` :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:Select"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/
table/*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "timestream:Select"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/
table/DevOps"
        }
    ]
}
```

Timestream pour l'accès aux LiveAnalytics ressources en fonction des balises

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès à Timestream pour les LiveAnalytics ressources en fonction de balises. Cette section fournit quelques exemples.

L'exemple suivant montre comment créer une stratégie qui accorde des autorisations à un utilisateur pour afficher une table si son `0wner` contient la valeur du nom d'utilisateur de cet utilisateur.

```
{
    "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}"
```

```
                }
            }
        }
    ]
}
```

Vous pouvez associer cette politique aux IAM utilisateurs de votre compte. Si un utilisateur nommé `richard-roe` tente d'afficher le flux temporel d'une LiveAnalytics table, la table doit être `Owner=richard-roe` balisée ou. `owner=richard-roe` Dans le cas contraire, l'utilisateur se voit refuser l'accès. La clé de condition d'étiquette `Owner` correspond à la fois à `Owner` et à `owner`, car les noms de clé de condition ne sont pas sensibles à la casse. Pour plus d'informations, voir Éléments IAM JSON de politique : condition dans le guide de IAM l'utilisateur.

La politique suivante autorise un utilisateur à créer des tables avec des balises si la balise transmise dans la demande comporte une clé `Owner` et une valeur `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}"
                }
            }
        }
    ]
}
```

La politique ci-dessous permet l'utilisation du `DescribeDatabase` API sur toute base de données dont la `env` balise est définie sur l'une `dev` ou l'autre des valeurs `test` suivantes :

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

```
  ]
}
```

Cette politique utilise une `Condition` clé pour autoriser une balise contenant la clé `env` et une valeur de `testqa`, ou `dev` à être ajoutée à une ressource.

Requêtes planifiées

Répertorier, supprimer, mettre à jour, exécuter ScheduledQuery

L'exemple de politique suivant permet à un utilisateur de répertorier, de supprimer, de mettre à jour et d'exécuter des requêtes planifiées.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DeleteScheduledQuery",
                "timestream:ExecuteScheduledQuery",
                "timestream:UpdateScheduledQuery",
                "timestream:ListScheduledQueries",
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*"
        }
    ]
}
```

CreateScheduledQuery à l'aide d'une KMS clé gérée par le client

L'exemple de politique suivant permet à un utilisateur de créer une requête planifiée chiffrée à l'aide d'une KMS clé gérée par le client ; *<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 à l'aide d'une KMS clé gérée par le client

L'exemple de politique suivant permet à un utilisateur de décrire une requête planifiée créée à l'aide d'une KMS clé gérée par le client ; *<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"
        }
    ]
}
```

Autorisations relatives au rôle d'exécution (utilisation d'une KMS clé gérée par le client pour les requêtes planifiées et SSE KMS pour les rapports d'erreur)

Associez l'exemple de politique suivant au IAM rôle spécifié dans le `ScheduledQueryExecutionRoleArn` paramètre, `CreateScheduledQuery` API qui utilise une KMS clé gérée par le client pour le chiffrement des requêtes planifiées et un `SSE-KMS` cryptage pour les rapports d'erreur.

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

Relation de confiance entre les rôles d'exécution

Voici la relation de confiance pour le IAM rôle spécifié dans le `ScheduledQueryExecutionRoleArn` paramètre du `CreateScheduledQuery`API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "timestream.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Autoriser l'accès à toutes les requêtes planifiées créées dans un compte

Associez l'exemple de politique suivant au IAM rôle spécifié dans le `ScheduledQueryExecutionRoleArn` paramètre, du `CreateScheduledQuery`API, pour autoriser l'accès à toutes les requêtes planifiées créées dans le compte *`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/*"
                }
            }
        }
    ]
}
```

Autoriser l'accès à toutes les requêtes planifiées portant un nom spécifique

Associez l'exemple de politique suivant au IAM rôle spécifié dans le `ScheduledQueryExecutionRoleArn` paramètre, du `CreateScheduledQuery`API, pour autoriser l'accès à toutes les requêtes planifiées dont le nom commence par *`Scheduled_Query_Name`*, dans le compte *`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*"
                }
            }
        }
    ]
}
```

Résolution des problèmes liés à Amazon Timestream en matière LiveAnalytics d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous utilisez Timestream pour LiveAnalytics et. IAM

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Timestream pour LiveAnalytics
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mon flux chronologique pour les ressources LiveAnalytics

Je ne suis pas autorisé à effectuer une action dans Timestream pour LiveAnalytics

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit lorsque l'`mateojackson`IAMutilisateur essaie d'utiliser la console pour afficher les détails d'un *`table`* mais n'est pas `timestream:`*`Select`* autorisé à accéder à la table.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream:Select on resource: mytable
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`mytable`* à l'aide de l'action `timestream:`*`Select`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Timestream pour. LiveAnalytics

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un IAM utilisateur nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Timestream for. LiveAnalytics Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mon flux chronologique pour les ressources LiveAnalytics

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez

spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si Timestream for LiveAnalytics prend en charge ces fonctionnalités, consultez. Comment fonctionne Amazon Timestream for LiveAnalytics avec IAM
- Pour savoir comment donner accès à vos ressources sur un site Comptes AWS qui vous appartient, consultez la section Fournir l'accès à un IAM utilisateur dans un autre site Compte AWS que vous possédez dans le Guide de IAM l'utilisateur.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir l'accès à des ressources Comptes AWS détenues par des tiers dans le Guide de IAM l'utilisateur.
- Pour savoir comment fournir un accès via la fédération d'identité, consultez la section Fournir un accès aux utilisateurs authentifiés de manière externe (fédération d'identité) dans le guide de l'IAMutilisateur.
- Pour connaître la différence entre l'utilisation de rôles et de politiques basées sur les ressources pour l'accès entre comptes, voir Accès aux ressources entre comptes IAM dans le guide de l'IAMutilisateur.

Enregistrement et surveillance dans Timestream pour LiveAnalytics

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances de Timestream pour LiveAnalytics et de vos AWS solutions. Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant. Toutefois, avant de commencer à surveiller Timestream pour LiveAnalytics, vous devez créer un plan de surveillance comprenant des réponses aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- À quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?

- Qui doit être informé en cas de problème ?

L'étape suivante consiste à établir une base de référence pour un Timestream normal pour les LiveAnalytics performances dans votre environnement, en mesurant les performances à différents moments et dans différentes conditions de charge. Lorsque vous surveillez Timestream LiveAnalytics, stockez les données de surveillance historiques afin de pouvoir les comparer aux données de performance actuelles, d'identifier les modèles de performances normaux et les anomalies de performance, et de concevoir des méthodes pour résoudre les problèmes.

Pour établir une référence, vous devez, au Moins, superviser les éléments suivants :

- Les erreurs système, de sorte que vous puissiez déterminer si des demandes ont entraîné une erreur.

Rubriques

Outils de surveillance

AWS fournit divers outils que vous pouvez utiliser pour surveiller Timestream. LiveAnalytics Vous pouvez configurer certains outils pour qu'ils effectuent la supervision automatiquement, tandis que d'autres nécessitent une intervention manuelle. Nous vous recommandons d'automatiser le plus possible les tâches de supervision.

Rubriques

Outils de surveillance automatique

Vous pouvez utiliser les outils de surveillance automatique suivants pour surveiller Timestream LiveAnalytics et signaler tout problème :

- Amazon CloudWatch Alarms : surveillez une seule métrique sur une période que vous spécifiez et effectuez une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil

donné sur un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon Simple Notification Service (AmazonSNS) ou à une politique Amazon EC2 Auto Scaling. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier ; l'état doit avoir changé et être maintenu pendant un certain nombre de périodes. Pour de plus amples informations, veuillez consulter Surveillance avec Amazon CloudWatch.

Outils de surveillance manuelle

Un autre élément important de la surveillance de Timestream LiveAnalytics consiste à surveiller manuellement les éléments non couverts par les CloudWatch alarmes. Le Timestream pour LiveAnalytics, CloudWatch Trusted Advisor, et les autres AWS Management Console tableaux de bord fournissent une at-a-glance vue de l'état de votre environnement. AWS

- La page CloudWatch d'accueil affiche les informations suivantes :
  - Alarmes et statuts en cours
  - Graphiques des alarmes et des ressources
  - Statut d'intégrité du service

  En outre, vous pouvez utiliser CloudWatch pour effectuer les opérations suivantes :
  - Créer des tableaux de bord personnalisés pour surveiller les services de votre choix
  - Représenter graphiquement les données de métriques pour résoudre les problèmes et découvrir les tendances
  - Recherchez et parcourez tous les indicateurs de vos AWS ressources
  - Créer et modifier des alarmes pour être informé des problèmes

Enregistrement du flux temporel des appels LiveAnalytics API avec AWS CloudTrail

Timestream for LiveAnalytics est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Timestream for. LiveAnalytics CloudTrail capture les API appels du Data Definition Language (DDL) pour Timestream LiveAnalytics sous forme d'événements. Les appels capturés incluent les appels provenant du Timestream pour la LiveAnalytics console et les appels codés vers le Timestream pour les opérations. LiveAnalytics API Si vous créez un suivi, vous pouvez activer la diffusion continue des CloudTrail événements vers un bucket Amazon Simple Storage Service (Amazon S3), y compris les événements pour Timestream for. LiveAnalytics Si vous ne configurez pas de suivi, vous pouvez

toujours consulter les événements les plus récents sur la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Timestream LiveAnalytics, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Diffusion chronologique des informations en LiveAnalytics CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans Timestream for LiveAnalytics, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS . Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

> **Warning**
>
> Actuellement, Timestream for LiveAnalytics génère des CloudTrail événements pour l'ensemble de la gestion et des `Query` API opérations, mais ne génère pas d'événements pour `WriteRecords` et. `DescribeEndpoints` APIs

Pour un enregistrement continu des événements de votre AWS compte, y compris les événements pour Timestream for LiveAnalytics, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un parcours dans la console, celui-ci s'applique à toutes les AWS régions. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence.

Pour plus d’informations, consultez les rubriques suivantes dans le AWS CloudTrail Guide de l’utilisateur :

- Présentation de la création d’un journal d’activité
- CloudTrail Services et intégrations pris en charge
- Configuration des SNS notifications Amazon pour CloudTrail
- Réception de fichiers CloudTrail journaux provenant de plusieurs régions
- Réception de fichiers CloudTrail journaux provenant de plusieurs comptes

- Journalisation des événements liés aux données

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou AWS Identity and Access Management (IAM)
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré
- Si la demande a été faite par un autre AWS service

Pour plus d'informations, consultez l'CloudTrail userIdentityélément.

Pour les `Query` API événements :

- Créez un parcours qui reçoit tous les événements ou sélectionnez des événements avec Timestream pour le type `AWS::Timestream::Database` de LiveAnalytics ressource ou. `AWS::Timestream::Table`
- `Query`APIIles demandes qui n'accèdent à aucune base de données ou table ou qui entraînent une exception de validation en raison d'une chaîne de requête mal formée sont enregistrées CloudTrail avec un type de ressource `AWS::Timestream::Database` et une ARN valeur de :

  ```
  arn:aws:timestream:(region):(accountId):database/NO_RESOURCE_ACCESSED
  ```

  Ces événements sont organisés uniquement sur les sentiers qui accueillent des événements avec un type de ressource`AWS::Timestream::Database`.

Résilience dans Amazon Timestream Live Analytics

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Pour plus d'informations sur les fonctionnalités de protection des données pour Timestream disponibles via AWS Backup, voir. Travailler avec AWS Backup

Sécurité de l'infrastructure dans Amazon Timestream Live Analytics

En tant que service géré, Amazon Timestream Live Analytics est protégé par les procédures de sécurité AWS du réseau mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez les API appels AWS publiés pour accéder à Timestream Live Analytics via le réseau. Les clients doivent prendre en charge Transport Layer Security (TLS) 1.0 ou version ultérieure. Nous recommandons la version TLS 1.2 ou une version ultérieure. Les clients doivent également prendre en charge les suites de chiffrement parfaitement confidentielles (), telles que Ephemeral Diffie-Hellman (PFS) ou Elliptic Curve Ephemeral Diffie-Hellman (DHE). ECDHE La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un identifiant de clé d'accès et d'une clé d'accès secrète associés à un IAM principal. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Timestream Live Analytics est conçu de telle sorte que votre trafic soit isolé par rapport à la AWS région spécifique dans laquelle réside votre instance Timestream Live Analytics.

Analyse de configuration et de vulnérabilité dans Timestream

La configuration et les contrôles informatiques sont une responsabilité partagée entre vous AWS et vous, notre client. Pour plus d'informations, consultez le modèle de responsabilité AWS partagée. Outre le modèle de responsabilité partagée, Timestream for LiveAnalytics users doit tenir compte des points suivants :

- Il est de la responsabilité du client de corriger ses applications clients avec les dépendances côté client pertinentes.
- Les clients devraient envisager des tests de pénétration le cas échéant (voir https://aws.amazon.com/security/tests de pénétration/.)

Réponse aux incidents dans Timestream pour LiveAnalytics

Les incidents liés au service Amazon Timestream LiveAnalytics sont signalés dans le Personal Health Dashboard. Pour en savoir plus sur le tableau de bord, cliquez AWS Health ici.

Timestream pour les LiveAnalytics aides à l'utilisation de rapports. AWS CloudTrail Pour de plus amples informations, veuillez consulter Enregistrement du flux temporel des appels LiveAnalytics API avec AWS CloudTrail.

VPCpoints de terminaison ()AWS PrivateLink

Vous pouvez établir une connexion privée entre votre compte VPC et Amazon Timestream en créant un point de LiveAnalytics terminaison d'interface. VPC Les points de terminaison de l'interface sont alimentés par AWS PrivateLinkune technologie qui vous permet d'accéder à Timestream en privé LiveAnalytics APIs sans passerelle Internet, NAT appareil, VPN connexion ou connexion Direct AWS Connect. Les instances de votre VPC ordinateur n'ont pas besoin d'adresses IP publiques pour communiquer avec Timestream pour. LiveAnalytics APIs Le trafic entre vous VPC et Timestream pour LiveAnalytics ne quitte pas le réseau Amazon.

Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux. Pour plus d'informations sur les VPC points de terminaison d'interface, consultez la section VPCpoints de terminaison d'interface (AWS PrivateLink) dans le guide de VPCl'utilisateur Amazon.

Pour commencer à utiliser Timestream pour LiveAnalytics et les VPC points de terminaison, nous avons fourni des informations sur les considérations spécifiques relatives à Timestream pour les points de VPC terminaison, à la création d'un point de VPC terminaison d'interface pour Timestream for, à la création d'une politique de point de VPC terminaison pour LiveAnalytics Timestream for et à l'utilisation du client Timestream (pour LiveAnalytics Write ou Query) LiveAnalytics avec les points de terminaison. SDK VPC

Rubriques

- Comment les VPC terminaux fonctionnent avec Timestream
- Création d'un point de VPC terminaison d'interface pour Timestream pour LiveAnalytics
- Création d'une politique de VPC point de terminaison pour Timestream pour LiveAnalytics

Comment les VPC terminaux fonctionnent avec Timestream

Lorsque vous créez un VPC point de terminaison pour accéder à la requête Timestream Write ou TimestreamSDK, toutes les demandes sont acheminées vers les points de terminaison du réseau Amazon et n'accèdent pas à l'Internet public. Plus précisément, vos demandes sont acheminées vers les points de terminaison d'écriture et de requête de la cellule vers laquelle votre compte a été mappé pour une région donnée. Pour en savoir plus sur l'architecture cellulaire de Timestream et les points de terminaison spécifiques aux cellules, vous pouvez vous référer à. Architecture cellulaire Supposons, par exemple, que votre compte ait été mappé à `cell1` in `us-west-2` et que vous ayez configuré des points de terminaison d'VPCinterface pour les écritures (`ingest-cell1.timestream.us-west-2.amazonaws.com`) et les requêtes (`query-cell1.timestream.us-west-2.amazonaws.com`). Dans ce cas, toutes les demandes d'écriture envoyées via ces points de terminaison resteront entièrement sur le réseau Amazon et n'auront pas accès à l'Internet public.

Considérations relatives aux points de terminaison Timestream VPC

Tenez compte des points suivants lors de la création d'un VPC point de terminaison pour Timestream :

- Avant de configurer un point de VPC terminaison d'interface pour Timestream for LiveAnalytics, assurez-vous de consulter les propriétés et les limites du point de terminaison d'interface dans le guide de VPCl'utilisateur Amazon.
- Timestream pour LiveAnalytics les supports qui appellent toutes ses API actions depuis votre. VPC
- VPCles politiques relatives aux terminaux sont prises en charge pour Timestream for. LiveAnalytics Par défaut, l'accès complet à Timestream pour LiveAnalytics est autorisé via le point de terminaison. Pour plus d'informations, consultez la section Contrôle de l'accès aux services avec des VPC points de terminaison dans le guide de VPC l'utilisateur Amazon.
- En raison de l'architecture de Timestream, l'accès aux actions Write et Query nécessite la création de deux points de terminaison d'VPCinterface, un pour chacun. SDK En outre, vous devez spécifier un point de terminaison de cellule (vous ne pourrez créer un point de terminaison que pour la cellule Timestream à laquelle vous êtes mappé). Vous trouverez des informations détaillées dans la LiveAnalytics section Créer un point de VPC terminaison d'interface pour Timestream de ce guide.

Maintenant que vous comprenez comment Timestream for LiveAnalytics fonctionne avec les VPC points de terminaison, créez un point de VPC terminaison d'interface pour Timestream for. LiveAnalytics

Création d'un point de VPC terminaison d'interface pour Timestream pour LiveAnalytics

Vous pouvez créer un point de VPCterminaison d'interface pour le LiveAnalytics service Timestream for à l'aide de la VPC console Amazon ou du AWS Command Line Interface ()AWS CLI. Pour créer un VPC point de terminaison pour Timestream, suivez les étapes spécifiques à Timestream décrites ci-dessous.

> **Note**
>
> Avant de suivre les étapes ci-dessous, assurez-vous de bien comprendre les considérations spécifiques relatives aux points de terminaison Timestream VPC.

Création d'un nom de service de VPC point de terminaison à l'aide de votre cellule Timestream

En raison de l'architecture unique de Timestream, des points de terminaison d'VPCinterface distincts doivent être créés pour chacun SDK (écriture et requête). En outre, vous devez spécifier un point de terminaison de cellule Timestream (vous ne pourrez créer un point de terminaison que pour la cellule Timestream à laquelle vous êtes mappé). Pour utiliser les VPC points de terminaison d'interface pour vous connecter directement à Timestream depuis votre ordinateurVPC, procédez comme suit :

1. Tout d'abord, recherchez un point de terminaison cellulaire Timestream disponible. Pour trouver un point de terminaison de cellule disponible, utilisez l'`DescribeEndpoints`action (disponible à la fois via Write et QueryAPIs) pour répertorier les points de terminaison de cellule disponibles dans votre compte Timestream. Consultez l'exemple pour plus de détails.
2. Une fois que vous avez sélectionné un point de terminaison de cellule à utiliser, créez une chaîne de point de terminaison d'VPCinterface pour Timestream Write ou Query : API
   - Pour le Write API :

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

   - Pour la requête API :

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

où *<region>* est un code de AWS région valide et *<cell>* est l'une des adresses de point de terminaison de la cellule (telle que `cell1` ou`cell2`) renvoyée dans l'objet Endpoints par l'DescribeEndpoints action. Consultez l'exemple pour plus de détails.

3. Maintenant que vous avez créé un nom de service de VPC point de terminaison, créez un point de terminaison d'interface. Lorsqu'on vous demande de fournir un nom de service de point de VPC terminaison, utilisez le nom de service de VPC point de terminaison que vous avez créé à l'étape 2.

Exemple : construction du nom de votre service de VPC point de terminaison

Dans l'exemple suivant, l'`DescribeEndpoints`action est exécutée à l' AWS CLIaide de la commande Write API in the `us-west-2` region :

```
aws timestream-write describe-endpoints --region us-west-2
```

Cette commande renverra le résultat suivant :

```
{
    "Endpoints": [
        {
            "Address": "ingest-cell1.timestream.us-west-2.amazonaws.com",
            "CachePeriodInMinutes": 1440
        }
    ]
}
```

Dans ce cas, *cell1* est le *<cell>* , et *us-west-2* est le *<region>*. Ainsi, le nom du service de VPC point de terminaison qui en résultera ressemblera à :

```
com.amazonaws.us-west-2.timestream.ingest-cell1
```

Maintenant que vous avez créé un point de VPC terminaison d'interface pour Timestream for LiveAnalytics, créez une politique de point de VPC terminaison pour Timestream for. LiveAnalytics

Création d'une politique de VPC point de terminaison pour Timestream pour LiveAnalytics

Vous pouvez associer une politique de point de terminaison à votre VPC point de terminaison qui contrôle l'accès à Timestream pour. LiveAnalytics La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez la section Contrôle de l'accès aux services avec des VPC points de terminaison dans le guide de VPC l'utilisateur Amazon.

Exemple : politique de VPC point de terminaison pour Timestream pour les actions LiveAnalytics

Voici un exemple de politique de point de terminaison pour Timestream for. LiveAnalytics Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès au flux temporel répertorié pour les LiveAnalytics actions (dans ce cas, `ListDatabases`) pour tous les principaux sur toutes les ressources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "timestream:ListDatabases"
         ],
         "Resource":"*"
      }
   ]
}
```

Bonnes pratiques de sécurité pour Amazon Timestream pour LiveAnalytics

Amazon Timestream LiveAnalytics for fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques

peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Rubriques

- Timestream pour les meilleures pratiques de LiveAnalytics sécurité préventive

Timestream pour les meilleures pratiques de LiveAnalytics sécurité préventive

Les meilleures pratiques suivantes peuvent vous aider à anticiper et à prévenir les incidents de sécurité dans Timestream for. LiveAnalytics

Chiffrement au repos

Timestream pour LiveAnalytics chiffrer au repos toutes les données utilisateur stockées dans des tables à l'aide des clés de chiffrement stockées dans AWS Key Management Service ().AWS KMS Cela fournit une couche supplémentaire de protection des données en sécurisant vos données contre tout accès non autorisé au stockage sous-jacent.

Timestream for LiveAnalytics utilise une clé par défaut de service unique (AWS détenueCMK) pour chiffrer toutes vos tables. Si cette clé n'existe pas, elle est créée pour vous. Les clés par défaut du service ne peuvent pas être désactivées. Pour plus d'informations, consultez Timestream pour le LiveAnalytics chiffrement au repos.

Utiliser IAM des rôles pour authentifier l'accès à Timestream pour LiveAnalytics

Pour que les utilisateurs, applications et autres AWS services puissent accéder à Timestream LiveAnalytics, ils doivent inclure des AWS informations d'identification valides dans leurs AWS API demandes. Vous ne devez pas stocker les AWS informations d'identification directement dans l'application ou l'EC2instance. Il s'agit d'informations d'identification à long terme qui ne font pas l'objet d'une rotation automatique, et dont la compromission pourrait avoir un impact considérable sur l'activité. Un IAM rôle vous permet d'obtenir des clés d'accès temporaires qui peuvent être utilisées pour accéder aux AWS services et aux ressources.

Pour plus d'informations, consultez Rôles IAM.

Utiliser IAM les politiques de Timestream pour LiveAnalytics l'autorisation de base

Lorsque vous accordez des autorisations, vous décidez qui les obtient, pour quel Timestream LiveAnalytics APIs ils obtiennent des autorisations et quelles actions spécifiques vous souhaitez autoriser sur ces ressources. L'implémentation d'un privilège minimum est la clé de la réduction des risques de sécurité et de l'impact potentiel d'erreurs ou d'actes de malveillance.

Associez des politiques d'autorisation aux IAM identités (c'est-à-dire aux utilisateurs, aux groupes et aux rôles) et accordez ainsi des autorisations pour effectuer des opérations sur Timestream pour les LiveAnalytics ressources.

Pour ce faire, utilisez les ressources suivantes :

- AWS politiques gérées (prédéfinies)
- Politiques gérées par le client
- Autorisation basée sur des balises

Envisager un chiffrement côté client

Si vous stockez des données sensibles ou confidentielles dans Timestream for LiveAnalytics, vous souhaiterez peut-être chiffrer ces données le plus près possible de leur origine afin de les protéger tout au long de leur cycle de vie. Le chiffrement de vos données sensibles en transit et au repos permet de garantir que vos données en texte brut ne sont pas accessibles à des tiers.

Utilisation d'autres services

Amazon Timestream LiveAnalytics for s'intègre à de nombreux services et outils AWS tiers populaires. Actuellement, Timestream for LiveAnalytics prend en charge les intégrations avec les éléments suivants :

Rubriques

- Amazon DynamoDB
- AWS Lambda
- AWS IoT Core
- Service géré Amazon pour Apache Flink
- Amazon Kinesis
- Amazon MQ
- Amazon MSK
- Amazon QuickSight
- Amazon SageMaker
- Amazon SQS
- Utilisation DBeaver pour travailler avec Amazon Timestream

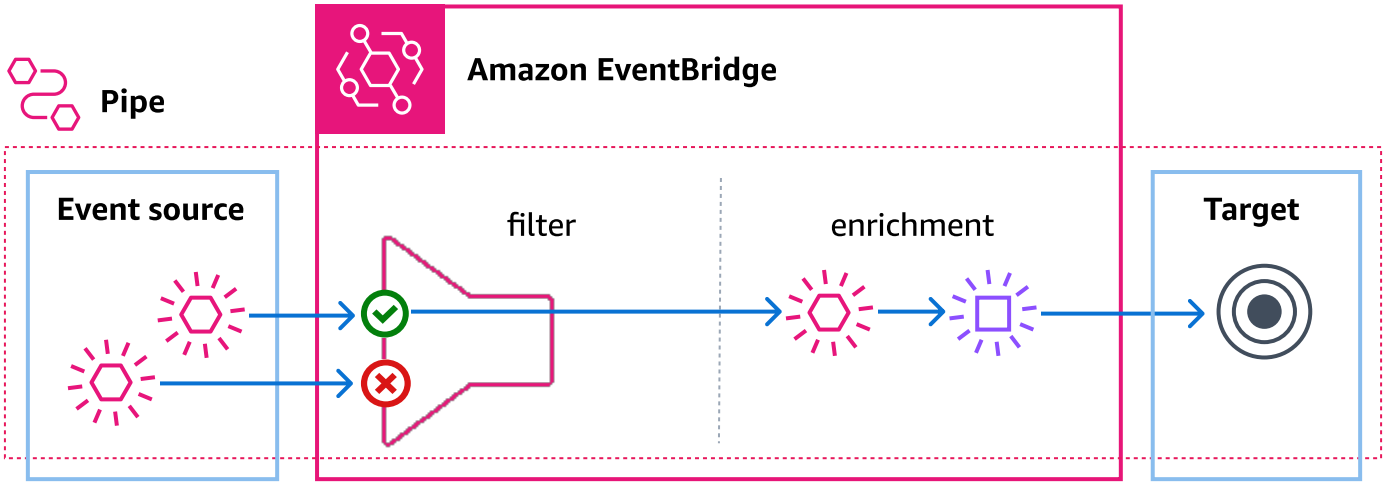
- Grafana
- Utilisation SquaredUp pour travailler avec Amazon Timestream
- Telegraf open source
- JDBC
- ODBC
- VPCpoints de terminaison ()AWS PrivateLink

Amazon DynamoDB

Utilisation de EventBridge Pipes pour envoyer des données DynamoDB à Timestream

Vous pouvez utiliser EventBridge Pipes pour envoyer des données d'un flux DynamoDB vers une table Amazon Timestream for. LiveAnalytics

Les tubes sont destinés aux point-to-point intégrations entre les sources et les cibles prises en charge, avec la prise en charge des transformations avancées et de l'enrichissement. Pipes réduit le besoin de connaissances spécialisées et de code d'intégration lors du développement d'architectures pilotées par des événements. Pour configurer un canal, vous devez choisir la source, ajouter un filtrage facultatif, définir un enrichissement facultatif et choisir la cible pour les données d'événement.



Pour plus d'informations sur les EventBridge tuyaux, voir EventBridge Tuyaux dans le guide de EventBridge l'utilisateur. Pour plus d'informations sur la configuration d'un canal pour transmettre des événements à un tableau Amazon Timestream LiveAnalytics for, EventBridge consultez la section Caractéristiques de la cible des canaux.

AWS Lambda

Vous pouvez créer des fonctions Lambda qui interagissent avec Timestream pour. LiveAnalytics Par exemple, vous pouvez créer une fonction Lambda qui s'exécute à intervalles réguliers pour exécuter une requête sur Timestream et envoyer une SNS notification en fonction des résultats de la requête répondant à un ou plusieurs critères. Pour en savoir plus sur Lambda, consultez la documentation Lambda AWS .

Rubriques

- Création de fonctions AWS Lambda à l'aide d'Amazon Timestream pour Python LiveAnalytics
- Créez des fonctions AWS Lambda à l'aide d'Amazon Timestream pour avec LiveAnalytics JavaScript
- Créez des fonctions AWS Lambda à l'aide d'Amazon Timestream pour with Go LiveAnalytics
- Création de fonctions AWS Lambda à l'aide d'Amazon Timestream pour C# LiveAnalytics

Création de fonctions AWS Lambda à l'aide d'Amazon Timestream pour Python LiveAnalytics

Pour créer des fonctions AWS Lambda à l'aide d'Amazon Timestream LiveAnalytics pour Python, suivez les étapes ci-dessous.

1. Créez un IAM rôle que Lambda assumera et qui accordera les autorisations requises pour accéder au service Timestream, comme indiqué dans. Fournir un Timestream pour l'accès LiveAnalytics
2. Modifiez la relation de confiance du IAM rôle pour ajouter le service Lambda. Vous pouvez utiliser les commandes ci-dessous pour mettre à jour un rôle existant afin que AWS Lambda puisse l'assumer :

    a. Créez le document de politique de confiance :

```
cat > Lambda-Role-Trust-Policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
```

```
          "lambda.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Mettez à jour le rôle de l'étape précédente avec le document de confiance

```
aws iam update-assume-role-policy --role-name <name_of_the_role_from_step_1> --
policy-document file://Lambda-Role-Trust-Policy.json
```

Les références connexes se trouvent à TimestreamWriteet TimestreamQuery.

Créez des fonctions AWS Lambda à l'aide d'Amazon Timestream pour avec LiveAnalytics JavaScript

Pour créer des fonctions AWS Lambda à l'aide d'Amazon Timestream LiveAnalytics for JavaScript with, suivez les instructions décrites ici.

Les références connexes se trouvent sur Timestream Write Client - AWS SDK pour la JavaScript v3 et Timestream Query Client - pour la v3. AWS SDK JavaScript

Créez des fonctions AWS Lambda à l'aide d'Amazon Timestream pour with Go LiveAnalytics

Pour créer des fonctions AWS Lambda à l'aide d'Amazon Timestream LiveAnalytics pour with Go, suivez les instructions décrites ici.

Les références associées se trouvent sur timestreamwrite et timestreamquery.

Création de fonctions AWS Lambda à l'aide d'Amazon Timestream pour C# LiveAnalytics

Pour créer des fonctions AWS Lambda à l'aide d'Amazon Timestream LiveAnalytics pour C#, suivez les instructions décrites ici.

Les références associées se trouvent sur Amazon. TimestreamWriteet Amazon. TimestreamQuery.

AWS IoT Core

Vous pouvez collecter des données à partir d'appareils IoT à l'aide d'AWS IoT Core et les acheminer vers Amazon Timestream via des actions de règles IoT Core. AWS Les actions relatives aux règles de l'IoT spécifient ce qu'il faut faire lorsqu'une règle est déclenchée. Vous pouvez définir des actions pour envoyer des données vers une table Amazon Timestream, une base de données Amazon DynamoDB et appeler une fonction Lambda. AWS

L'action Timestream dans IoT Rules est utilisée pour insérer les données des messages entrants directement dans Timestream. L'action analyse les résultats de la SQL déclaration IoT Core et stocke les données dans Timestream. Les noms des champs du jeu de SQL résultats renvoyé sont utilisés comme measure : :name et la valeur du champ est la measure : :value.

Par exemple, considérez l'SQLinstruction et la charge utile de l'exemple de message :

```
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
}
```

Si une action de règle IoT Core pour Timestream est créée avec l'SQLinstruction ci-dessus, deux enregistrements seront ajoutés à Timestream avec les noms de mesure température et humidité et les valeurs de mesure de 24,04 et 43,605, respectivement.

Vous pouvez modifier le nom de mesure d'un enregistrement ajouté à Timestream en utilisant l'opérateur AS dans l'SELECTinstruction. La SQL déclaration ci-dessous créera un enregistrement avec le nom du message temp au lieu de temperature.

Le type de données de la mesure est déduit du type de données de la valeur de la charge utile du message. JSONles types de données tels que entier, double, booléen et chaîne sont mappés aux types de données Timestream deBIGINT,, DOUBLE et respectivement. BOOLEAN VARCHAR Les données peuvent également être forcées vers des types de données spécifiques à l'aide de la fonction cast (). Vous pouvez spécifier l'horodatage de la mesure. Si l'horodatage est laissé vide, l'heure à laquelle l'entrée a été traitée est utilisée.

Vous pouvez vous référer à la documentation relative aux règles et aux actions de Timestream pour plus de détails.

Pour créer une action de règle IoT Core afin de stocker des données dans Timestream, suivez les étapes ci-dessous :

Rubriques

- Prérequis
- Utilisation de la console
- À l'aide du CLI
- Exemple d'application
- Tutoriel vidéo

Prérequis

1. Créez une base de données dans Amazon Timestream en suivant les instructions décrites dans. Créer une base de données
2. Créez un tableau dans Amazon Timestream en suivant les instructions décrites dans. Créer une table

Utilisation de la console

1. Utilisez la console AWS de gestion pour AWS IoT Core pour créer une règle en cliquant sur Gérer > Routage des messages > Règles, puis sur Créer une règle.
2. Définissez le nom de la règle sur le SQL nom de votre choix et sur le texte ci-dessous

```
SELECT temperature as temp, humidity from 'iot/topic'
```

3. Sélectionnez Timestream dans la liste des actions

4. Spécifiez les noms de la base de données, de la table et des dimensions Timestream, ainsi que le rôle pour écrire les données dans Timestream. Si le rôle n'existe pas, vous pouvez en créer un en cliquant sur Créer des rôles
5. Pour tester la règle, suivez les instructions indiquées ici.

À l'aide du CLI

Si vous n'avez pas installé l'interface de ligne de AWS commande (AWS CLI), faites-le à partir d'ici.

1. Enregistrez la charge utile de la règle suivante dans un JSON fichier appelé timestream_rule.json. Remplacez *arn:aws:iam::123456789012:role/TimestreamRole* avec votre rôle arn qui permet à AWS l'IoT d'accéder aux données de stockage dans 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. Créez une règle de rubrique à l'aide de la commande suivante

```
aws iot create-topic-rule --rule-name timestream_test --topic-rule-payload file://
<path/to/timestream_rule.json> --region us-east-1
```

3. Récupérez les détails de la règle du sujet à l'aide de la commande suivante

```
aws iot get-topic-rule --rule-name timestream_test
```

4. Enregistrez la charge utile du message suivant dans un fichier appelé 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. Testez la règle à l'aide de la commande suivante

```
aws iot-data publish --topic 'iot/topic' --payload file://<path/to/
timestream_msg.json>
```

Exemple d'application

Pour vous aider à commencer à utiliser Timestream avec AWS IoT Core, nous avons créé un exemple d'application entièrement fonctionnel qui crée les artefacts nécessaires dans AWS IoT Core et Timestream pour créer une règle de sujet, ainsi qu'un exemple d'application pour publier des données sur le sujet.

1. Clonez le GitHub référentiel de l'exemple d'application pour l'intégration de l' AWS IoT Core en suivant les instructions de GitHub

2. Suivez les instructions du README pour utiliser un AWS CloudFormation modèle afin de créer les artefacts nécessaires dans Amazon Timestream AWS et IoT Core et pour publier des exemples de messages sur le sujet.

Tutoriel vidéo

Cette vidéo explique comment IoT Core fonctionne avec Timestream.

Service géré Amazon pour Apache Flink

Vous pouvez utiliser Apache Flink pour transférer vos données de séries chronologiques depuis Amazon Managed Service pour Apache FlinkMSK, Amazon, Apache Kafka et d'autres technologies de streaming directement vers Amazon Timestream pour. LiveAnalytics Nous avons créé un exemple de connecteur de données Apache Flink pour Timestream. Nous avons également créé un exemple d'application pour envoyer des données à Amazon Kinesis afin que les données puissent être transmises de Kinesis à Managed Service pour Apache Flink, puis à Amazon Timestream. Tous ces artefacts sont à votre disposition dans GitHub. Ce didacticiel vidéo décrit la configuration.

> **Note**
>
> Java 11 est la version recommandée pour utiliser le service géré pour l'application Apache Flink. Si vous disposez de plusieurs versions de Java, assurez-vous d'exporter Java 11 vers votre variable d'HOMEenvironnement JAVA _.

Rubriques

- Exemple d'application
- Tutoriel vidéo

Exemple d'application

Pour commencer, suivez la procédure ci-dessous :

1. Créez une base de données dans Timestream avec le nom en `kdaflink` suivant les instructions décrites dans Créer une base de données
2. Créez une table dans Timestream avec le nom en `kinesisdata1` suivant les instructions décrites dans Créer une table

3. Créez un flux de données Amazon Kinesis portant le nom en `TimestreamTestStream` suivant les instructions décrites dans Création d'un flux
4. Clonez le GitHub référentiel pour le connecteur de données Apache Flink pour Timestream en suivant les instructions de GitHub
5. Pour compiler, exécuter et utiliser l'exemple d'application, suivez les instructions du connecteur de données d'exemple Apache Flink README
6. Compilez l'application Managed Service for Apache Flink en suivant les instructions de compilation du code de l'application
7. Téléchargez le fichier binaire de l'application Managed Service for Apache Flink en suivant les instructions pour télécharger le code de streaming Apache Flink

   a. Après avoir cliqué sur Créer une application, cliquez sur le lien du IAM rôle de l'application

   b. Joignez les IAM politiques pour AmazonKinesisReadOnlyAccesset AmazonTimestreamFullAccess.

   > **Note**
   >
   > Les IAM politiques ci-dessus ne sont pas limitées à des ressources spécifiques et ne sont pas adaptées à une utilisation en production. Pour un système de production, envisagez d'utiliser des politiques qui limitent l'accès à des ressources spécifiques.

8. Clonez le GitHub référentiel pour l'exemple d'application écrivant des données dans Kinesis en suivant les instructions de GitHub
9. Suivez les instructions du READMEpour exécuter l'exemple d'application permettant d'écrire des données dans Kinesis
10. Exécutez une ou plusieurs requêtes dans Timestream pour vous assurer que les données sont envoyées de Kinesis à Managed Service for Apache Flink à Timestream en suivant les instructions de Créer une table

Tutoriel vidéo

Cette vidéo explique comment utiliser Timestream avec Managed Service pour Apache Flink.

Amazon Kinesis

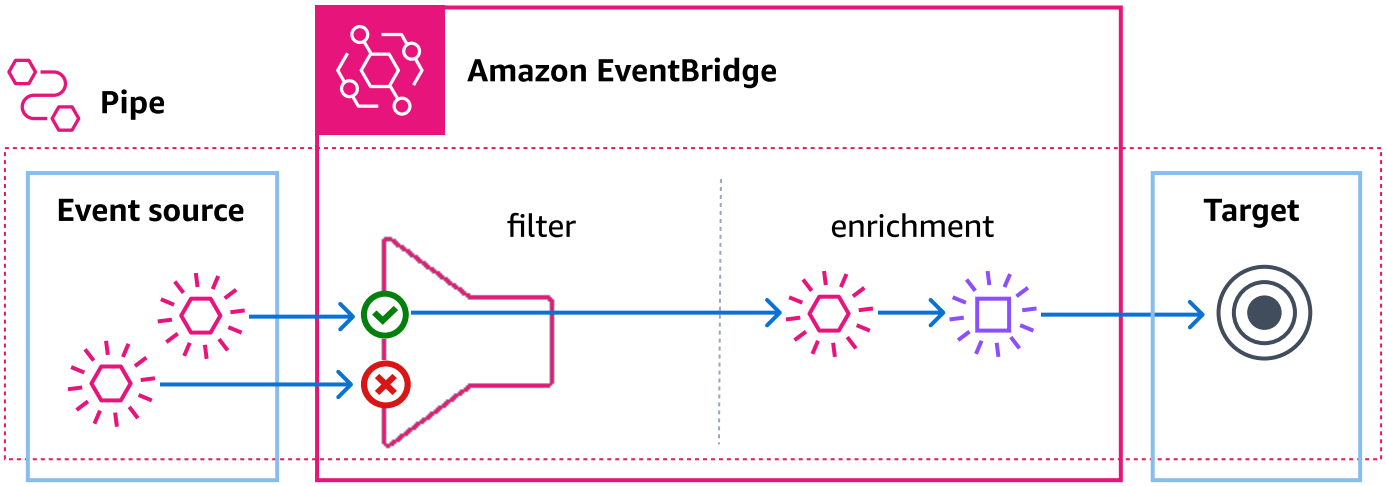
En utilisant Service géré Amazon pour Apache Flink

Vous pouvez envoyer des données de Kinesis Data Streams à Timestream LiveAnalytics pour utiliser le connecteur de données d' Timestream exemple pour Managed Service for Apache Flink. Reportez-vous Service géré Amazon pour Apache Flink à Apache Flink pour plus d'informations.

Utilisation de EventBridge Pipes pour envoyer des données Kinesis à Timestream

Vous pouvez utiliser EventBridge Pipes pour envoyer des données d'un flux Kinesis vers une table Amazon Timestream for. LiveAnalytics

Les tubes sont destinés aux point-to-point intégrations entre les sources et les cibles prises en charge, avec la prise en charge des transformations avancées et de l'enrichissement. Pipes réduit le besoin de connaissances spécialisées et de code d'intégration lors du développement d'architectures pilotées par les événements. Pour configurer un canal, vous devez choisir la source, ajouter un filtrage facultatif, définir un enrichissement facultatif et choisir la cible pour les données d'événement.



Cette intégration vous permet de tirer parti de la puissance des capacités Timestream d'analyse des séries chronologiques de données, tout en simplifiant votre pipeline d'ingestion de données.

L'utilisation de EventBridge Pipes with Timestream offre les avantages suivants :

- Ingestion des données en temps réel : diffusez les données de Kinesis directement vers Timestream pour permettre des analyses et une LiveAnalytics surveillance en temps réel.

- Intégration fluide : utilisez EventBridge Pipes pour gérer le flux de données sans avoir besoin d'intégrations personnalisées complexes.
- Filtrage et transformation améliorés : filtrez ou transformez les enregistrements Kinesis avant leur stockage afin de répondre Timestream à vos besoins spécifiques en matière de traitement des données.
- Évolutivité : gérez des flux de données à haut débit et garantissez un traitement efficace des données grâce aux fonctionnalités intégrées de parallélisme et de traitement par lots.

Configuration

Pour configurer un EventBridge canal vers lequel diffuser des données depuis Kinesis Timestream, procédez comme suit :

1. Créez un flux Kinesis

   Assurez-vous de disposer d'un flux de données Kinesis actif à partir duquel vous souhaitez ingérer des données.
2. Création d'une Timestream base de données et d'une table

   Configurez votre Timestream base de données et votre table dans laquelle les données seront stockées.
3. Configurez le EventBridge tuyau :
   - Source : sélectionnez votre flux Kinesis comme source.
   - Cible : Choisissez Timestream comme cible.
   - Paramètres de traitement par lots : définissez la fenêtre de traitement par lots et la taille du lot pour optimiser le traitement des données et réduire la latence.

> **Important**
>
> Lors de la configuration d'un canal, nous vous recommandons de tester l'exactitude de toutes les configurations en ingérant quelques enregistrements. Veuillez noter que la création réussie d'un tuyau ne garantit pas que le pipeline est correct et que les données circuleront sans erreur. Certaines erreurs d'exécution, telles qu'une table incorrecte, un paramètre de chemin dynamique incorrect ou un Timestream enregistrement non valide après l'application du mappage, seront découvertes lorsque les données réelles circuleront dans le canal.

Les configurations suivantes déterminent la vitesse à laquelle les données sont ingérées :

- BatchSize: taille maximale du lot qui sera envoyé à Timestream pour. LiveAnalytics Plage : 0 à 100. Il est recommandé de conserver cette valeur à 100 pour obtenir un débit maximal.
- MaximumBatchingWindowInSeconds: le temps d'attente maximal batchSize avant que le lot ne soit envoyé à Timestream pour LiveAnalytics la cible. En fonction du taux d'événements entrants, cette configuration décidera du délai d'ingestion. Il est recommandé de conserver cette valeur < 10 s pour continuer à envoyer les données Timestream en temps quasi réel.
- ParallelizationFactor: le nombre de lots à traiter simultanément à partir de chaque partition. Il est recommandé d'utiliser la valeur maximale de 10 pour obtenir un débit maximal et une ingestion en temps quasi réel.

  Si votre flux est lu par plusieurs cibles, utilisez un ventilateur amélioré pour fournir un consommateur dédié à votre chaîne afin d'atteindre un débit élevé. Pour plus d'informations, consultez la section Développement d'utilisateurs optimisés grâce Kinesis Data Streams API au guide de l'Kinesis Data Streams utilisateur.

> **Note**
>
> Le débit maximal pouvant être atteint est limité par les exécutions de canaux simultanées par compte.

La configuration suivante garantit la prévention des pertes de données :

- DeadLetterConfig: Il est recommandé de toujours configurer DeadLetterConfig pour éviter toute perte de données dans les cas où les événements n'ont pas pu être ingérés dans Timestream LiveAnalytics en raison d'erreurs utilisateur.

Optimisez les performances de votre canal à l'aide des paramètres de configuration suivants, qui permettent d'éviter que les enregistrements ne provoquent des ralentissements ou des blocages.

- MaximumRecordAgeInSeconds: Les enregistrements plus anciens ne seront pas traités et seront directement déplacés versDLQ. Nous recommandons de définir cette valeur pour qu'elle ne soit pas supérieure à la période de rétention de mémoire configurée pour la Timestream table cible.
- MaximumRetryAttempts: nombre de nouvelles tentatives pour un enregistrement avant que celui-ci ne soit envoyé à DeadLetterQueue. Il est recommandé de le configurer à 10. Cela devrait

permettre de résoudre les problèmes transitoires. En cas de problème persistant, l'enregistrement sera déplacé vers le reste du flux DeadLetterQueue et débloquera le reste du flux.

- OnPartialBatchItemFailure: Pour les sources qui prennent en charge le traitement partiel par lots, nous vous recommandons de l'activer et de le configurer sous la forme AUTOMATIC _ BISECT pour réessayer les enregistrements ayant échoué avant de les déposer/de les envoyer à. DLQ

Exemple de configuration

Voici un exemple de configuration d'un EventBridge canal pour diffuser les données d'un flux Kinesis vers une Timestream table :

Example IAM mises à jour des politiques pour 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 Configuration du flux 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 configuration cible

```
{
    "Target": "arn:aws:timestream:us-east-1:123456789012:database/my-database/table/my-
table",
    "TargetParameters": {
        "TimestreamParameters": {
            "DimensionMappings": [
                {
                    "DimensionName": "sensor_id",
                    "DimensionValue": "$.data.device_id",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "DimensionName": "sensor_type",
                    "DimensionValue": "$.data.sensor_type",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "DimensionName": "sensor_location",
                    "DimensionValue": "$.data.sensor_loc",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": [
                {
                    "MultiMeasureName": "readings",
                    "MultiMeasureAttributeMappings": [
                        {
```

```
                            "MultiMeasureAttributeName": "temperature",
                            "MeasureValue": "$.data.temperature",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "MultiMeasureAttributeName": "humidity",
                            "MeasureValue": "$.data.humidity",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "MultiMeasureAttributeName": "pressure",
                            "MeasureValue": "$.data.pressure",
                            "MeasureValueType": "DOUBLE"
                        }
                    ]
                }
            ],
            "SingleMeasureMappings": [],
            "TimeFieldType": "TIMESTAMP_FORMAT",
            "TimestampFormat": "yyyy-MM-dd HH:mm:ss.SSS",
            "TimeValue": "$.data.time",
            "VersionValue": "$.approximateArrivalTimestamp"
        }
    }
}
```

Transformation de l'événement

EventBridge Les canaux vous permettent de transformer les données avant qu'elles ne les atteignent Timestream. Vous pouvez définir des règles de transformation pour modifier les Kinesis enregistrements entrants, par exemple en modifiant les noms de champs.

Supposons que votre Kinesis flux contienne des données de température et d'humidité. Vous pouvez utiliser une EventBridge transformation pour renommer ces champs avant de les y insérer. Timestream

Bonnes pratiques

Traitement par lots et mise en mémoire tampon

- Configurez la fenêtre et la taille de traitement par lots afin d'équilibrer la latence d'écriture et l'efficacité du traitement.

- Utilisez une fenêtre de traitement par lots pour accumuler suffisamment de données avant le traitement, ce qui réduit les frais liés aux petits lots fréquents.

Traitement parallèle

Utilisez ce ParallelizationFactorparamètre pour augmenter la simultanéité, en particulier pour les flux à haut débit. Cela garantit que plusieurs lots de chaque partition peuvent être traités simultanément.

Transformation des données

Tirez parti des capacités de transformation de EventBridge Pipes pour filtrer et améliorer les enregistrements avant de les stocker Timestream. Cela peut vous aider à aligner les données sur vos exigences analytiques.

Sécurité

- Assurez-vous que les IAM rôles utilisés pour EventBridge Pipes disposent des autorisations nécessaires pour lire Kinesis et écrire Timestream.
- Utilisez des mesures de cryptage et de contrôle d'accès pour sécuriser les données en transit et au repos.

Défaillances de débogage

- Désactivation automatique des canalisations

  Les canaux seront automatiquement désactivés dans environ 2 heures si la cible n'existe pas ou présente des problèmes d'autorisation
- Throttles

  Les tuyaux ont la capacité de s'arrêter automatiquement et de réessayer jusqu'à ce que les gaz soient réduits.
- Activation des journaux

  Nous vous recommandons d'activer les journaux au ERROR niveau 1 et d'inclure les données d'exécution pour mieux comprendre les échecs. En cas de panne, ces journaux contiendront des informations request/response sent/received de Timestream. Cela vous permet de comprendre l'erreur associée et, si nécessaire, de retraiter les enregistrements après l'avoir corrigée.

Surveillance

Nous vous recommandons de configurer des alarmes sur les points suivants afin de détecter tout problème lié au flux de données :

- Âge maximal de l'enregistrement dans la source
  - `GetRecords.IteratorAgeMilliseconds`
- Mesures de défaillance dans les canalisations
  - `ExecutionFailed`
  - `TargetStageFailed`
- Timestream APIErreurs d'écriture
  - `UserErrors`

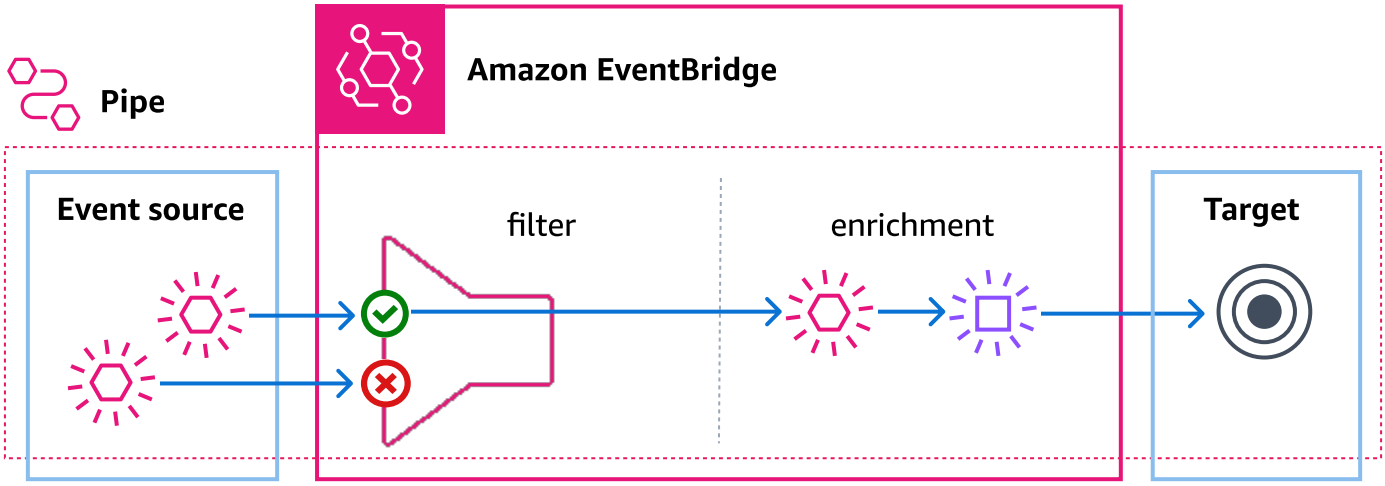
Pour des mesures de surveillance supplémentaires, consultez la section Surveillance EventBridge dans le guide de EventBridge l'utilisateur.

Amazon MQ

Utilisation de EventBridge Pipes pour envoyer des données Amazon MQ à Timestream

Vous pouvez utiliser EventBridge Pipes pour envoyer des données d'un courtier Amazon MQ vers une table Amazon Timestream for. LiveAnalytics

Les tubes sont destinés aux point-to-point intégrations entre les sources et les cibles prises en charge, avec la prise en charge des transformations avancées et de l'enrichissement. Pipes réduit le besoin de connaissances spécialisées et de code d'intégration lors du développement d'architectures pilotées par des événements. Pour configurer un canal, vous devez choisir la source, ajouter un filtrage facultatif, définir un enrichissement facultatif et choisir la cible pour les données d'événement.



Pour plus d'informations sur les EventBridge tuyaux, voir EventBridge Tuyaux dans le guide de EventBridge l'utilisateur. Pour plus d'informations sur la configuration d'un canal pour transmettre des événements à un tableau Amazon Timestream LiveAnalytics for, EventBridge consultez la section Caractéristiques de la cible des canaux.

Amazon MSK

Utilisation du service géré pour Apache Flink pour envoyer des Amazon MSK données à Timestream pour LiveAnalytics

Vous pouvez envoyer des données de Amazon MSK à Timestream en créant un connecteur de données similaire à l'exemple de connecteur de Timestream données pour Managed Service for Apache Flink. Pour plus d'informations, consultez la section Service géré Amazon pour Apache Flink.

Utiliser Kafka Connect pour envoyer des MSK données Amazon à Timestream pour LiveAnalytics

Vous pouvez utiliser Kafka Connect pour ingérer les données de vos séries chronologiques Amazon MSK directement dans Timestream for. LiveAnalytics

Nous avons créé un exemple de connecteur Kafka Sink pour Timestream. Nous avons également créé un exemple de plan de jMeter test Apache pour publier des données sur un sujet Kafka, afin que les données puissent être transmises du sujet via le connecteur Timestream Kafka Sink vers un flux temporel pour une table. LiveAnalytics Tous ces artefacts sont disponibles sur GitHub.

> **Note**
>
> Java 11 est la version recommandée pour utiliser le connecteur Timestream Kafka Sink. Si vous disposez de plusieurs versions de Java, assurez-vous d'exporter Java 11 vers votre variable d'HOMEenvironnement JAVA _.

Création d'un exemple d'application

Pour commencer, suivez la procédure ci-dessous.

1. Dans Timestream for LiveAnalytics, créez une base de données portant le nom. `kafkastream`

   Consultez la procédure ??? pour obtenir des instructions détaillées.
2. Dans Timestream for LiveAnalytics, créez une table portant le nom. `purchase_history`

   Consultez la procédure ??? pour obtenir des instructions détaillées.
3. Suivez les instructions partagées dans le pour créer les éléments suivants :, et.

   - Un Amazon MSK cluster
   - Une Amazon EC2 instance configurée en tant que machine cliente du producteur Kafka
   - Un sujet de Kafka

   Consultez les prérequis du projet kafka_ingestor pour des instructions détaillées.
4. Clonez le référentiel Timestream Kafka Sink Connector.

   Consultez la section Clonage d'un dépôt GitHub pour obtenir des instructions détaillées.
5. Compilez le code du plugin.

   Voir Connector - Build from source on GitHub pour des instructions détaillées.
6. Téléchargez les fichiers suivants dans un compartiment S3 : en suivant les instructions décrites dans.

   - Le fichier jar (kafka-connector-timestream-> VERSION <- jar-with-dependencies .jar) du répertoire `/target`
   - Exemple de fichier de schéma JSON,`purchase_history.json`.

Consultez la section Chargement d'objets dans le guide de Amazon S3 l'utilisateur pour obtenir des instructions détaillées.

7. Créez deux VPC points de terminaison. Ces points de terminaison seraient utilisés par le MSK connecteur pour accéder aux ressources à l'aide AWS PrivateLink de.

   - Un pour accéder au Amazon S3 compartiment
   - Un pour accéder au Timestream de la table. LiveAnalytics

   Voir VPCEndpoints pour des instructions détaillées.

8. Créez un plugin personnalisé avec le fichier jar téléchargé.

   Consultez la section Plugins dans le guide du Amazon MSK développeur pour obtenir des instructions détaillées.

9. Créez une configuration de travail personnalisée avec le JSON contenu décrit dans Paramètres de configuration de travail. En suivant les instructions décrites dans

   Consultez la section Création d'une configuration de travail personnalisée dans le Guide du Amazon MSK développeur pour obtenir des instructions détaillées.

10. Créez un IAM rôle d'exécution de service.

    Reportez-vous à la section IAM Service Role pour obtenir des instructions détaillées.

11. Créez un Amazon MSK connecteur avec le plugin personnalisé, la configuration de travail personnalisée et le IAM rôle d'exécution de service créés dans les étapes précédentes et avec l'exemple de configuration du connecteur.

    Reportez-vous à la section Création d'un connecteur dans le guide du Amazon MSK développeur pour obtenir des instructions détaillées.

    Assurez-vous de mettre à jour les valeurs des paramètres de configuration ci-dessous avec les valeurs respectives. Voir Paramètres de configuration du connecteur pour plus de détails.

    - `aws.region`
    - `timestream.schema.s3.bucket.name`
    - `timestream.ingestion.endpoint`

La création du connecteur prend 5 à 10 minutes. Le pipeline est prêt lorsque son statut passe à`Running`.

12. Publiez un flux continu de messages pour écrire des données sur le sujet Kafka créé.

    Consultez la section Comment l'utiliser pour obtenir des instructions détaillées.

13. Exécutez une ou plusieurs requêtes pour vous assurer que les données sont envoyées depuis la Amazon MSK table MSK Connect to the Timestream for LiveAnalytics .

    Consultez la procédure ??? pour obtenir des instructions détaillées.

Ressources supplémentaires

Le blog Real-time serverless data ingestion from your Kafka clusters into Timestream for using LiveAnalytics Kafka Connect explique la configuration d'un end-to-end pipeline à l'aide du connecteur Timestream for LiveAnalytics Kafka Sink, en partant d'une machine cliente du producteur Kafka qui utilise le plan de jMeter test Apache pour publier des milliers d'exemples de messages sur un sujet Kafka pour vérifier les enregistrements ingérés dans un Timestream pour une table. LiveAnalytics

Amazon QuickSight

Vous pouvez utiliser Amazon QuickSight pour analyser et publier des tableaux de bord contenant vos données Amazon Timestream. Cette section décrit comment créer une nouvelle connexion à une source de QuickSight données, modifier les autorisations, créer de nouveaux ensembles de données et effectuer une analyse. Ce didacticiel vidéo explique comment travailler avec Timestream et Amazon. QuickSight

> **Note**
>
> Tous les ensembles de données d'Amazon QuickSight sont en lecture seule. Vous ne pouvez pas modifier vos données réelles dans Timestream en utilisant Amazon QuickSight pour supprimer la source de données, le jeu de données ou les champs.

Rubriques

- Accès à Amazon Timestream depuis QuickSight
- Création d'une nouvelle connexion à une source de QuickSight données pour Timestream

- Modifier les autorisations pour la connexion à la source de QuickSight données pour Timestream
- Création d'un nouveau QuickSight jeu de données pour Timestream
- Créer une nouvelle analyse pour Timestream
- Tutoriel vidéo

Accès à Amazon Timestream depuis QuickSight

Avant de continuer, Amazon QuickSight doit être autorisé à se connecter à Amazon Timestream. Si les connexions ne sont pas activées, vous recevrez un message d'erreur lorsque vous tenterez de vous connecter. Un QuickSight administrateur peut autoriser les connexions aux AWS ressources. Pour autoriser une connexion depuis QuickSight Timestream, suivez la procédure décrite dans Using Other AWS Services : Scoping Down Access, en choisissant Amazon Timestream à l'étape 5.

Création d'une nouvelle connexion à une source de QuickSight données pour Timestream

> **Note**
>
> La connexion entre Amazon QuickSight et Amazon Timestream est cryptée en transit à l'SSLaide TLS de (1.2). Vous ne pouvez pas créer de connexion non cryptée.

1. Assurez-vous d'avoir configuré les autorisations appropriées pour qu'Amazon puisse accéder QuickSight à Amazon Timestream, comme décrit dans. Accès à Amazon Timestream depuis QuickSight
2. Commencez par créer un nouveau jeu de données. Choisissez Datasets dans le volet de navigation, puis choisissez New Dataset.
3. Sélectionnez la carte de source de données Timestream.
4. Dans Nom de la source de données, entrez un nom pour votre connexion à la source de données Timestream, par exemple. `US Timestream Data`

   > **Note**
   >
   > Comme vous pouvez créer de nombreux jeux de données à partir d'une connexion à Timestream, il est préférable de choisir un nom simple.

5. Choisissez Valider la connexion pour vérifier la bonne connexion à Timestream.

> **Note**
>
> Valider la connexion valide uniquement le fait que vous pouvez vous connecter. Cependant, il ne valide pas une table ou une requête spécifique.

6. Pour continuer, choisissez Créer une source de données.
7. Pour Base de données, choisissez Sélectionner... pour consulter la liste des options disponibles. Choisissez celui que vous souhaitez utiliser.
8. Choisissez Sélectionner pour continuer.
9. Sélectionnez l'une des méthodes suivantes :

   - Pour importer vos données dans le QuickSight moteur en mémoire (appeléSPICE), choisissez Importer vers SPICE pour des analyses plus rapides.
   - QuickSight Pour permettre d'exécuter une requête sur vos données chaque fois que vous actualisez le jeu de données ou que vous utilisez l'analyse ou le tableau de bord, choisissez Directly query your data.
10. Choisissez Modifier/apercevoir, puis Enregistrer pour enregistrer votre jeu de données et le fermer.

Modifier les autorisations pour la connexion à la source de QuickSight données pour Timestream

La procédure suivante décrit comment afficher, ajouter et révoquer des autorisations pour d'autres QuickSight utilisateurs afin qu'ils puissent accéder à la même source de données Timestream. Les utilisateurs doivent être des utilisateurs actifs QuickSight avant que vous puissiez les ajouter.

> **Note**
>
> Dans QuickSight, les sources de données ont deux niveaux d'autorisation : utilisateur et propriétaire.
>
> - Choisissez un utilisateur pour autoriser l'accès en lecture.
> - Choisissez le propriétaire pour autoriser cet utilisateur à modifier, partager ou supprimer cette source de QuickSight données.

1. Assurez-vous d'avoir configuré les autorisations appropriées pour qu'Amazon puisse accéder QuickSight à Amazon Timestream, comme décrit dans. Accès à Amazon Timestream depuis QuickSight
2. Choisissez Jeux de données sur la gauche, puis faites défiler l'écran vers le bas pour trouver la carte de source de données pour votre connexion Timestream. Par exemple, `US Timestream Data`.
3. Choisissez la carte `Timestream` de source de données.
4. Sélectionnez `Share data source`. La liste des autorisations actuelles s'affiche.
5. (Facultatif) Pour modifier les autorisations, vous pouvez choisir `user` ou`owner`.
6. (Facultatif) Pour révoquer les autorisations, choisissez`Revoke access`. Les personnes que vous révoquez ne peuvent pas créer de nouveaux ensembles de données à partir de cette source de données. Toutefois, leurs ensembles de données existants auront toujours accès à cette source de données.
7. Pour ajouter des autorisations, choisissez`Invite users`, puis suivez ces étapes pour ajouter un utilisateur :

   a. Ajoutez des personnes pour leur permettre d'utiliser la même source de données.

   b. Pour chacune d'entre elles, choisissez `Permission` celle que vous souhaitez appliquer.
8. Lorsque vous avez terminé, choisissez `Close`.

Création d'un nouveau QuickSight jeu de données pour Timestream

1. Assurez-vous d'avoir configuré les autorisations appropriées pour qu'Amazon puisse accéder QuickSight à Amazon Timestream, comme décrit dans. Accès à Amazon Timestream depuis QuickSight
2. Choisissez Jeux de données sur la gauche, puis faites défiler l'écran vers le bas pour trouver la carte de source de données pour votre connexion Timestream. Si vous avez de nombreuses sources de données, vous pouvez utiliser la barre de recherche en haut de la page pour les trouver dont le nom correspond partiellement à celui des sources.
3. Choisissez la carte de source de données Timestream. Choisissez ensuite Créer un ensemble de données.
4. Pour Base de données, cliquez sur Sélectionner pour afficher la liste des options disponibles. Choisissez la base de données que vous souhaitez utiliser.
5. Pour Tableaux, sélectionnez le tableau que vous souhaitez utiliser.

6. Choisissez Modifier/apercevoir.
7. (Facultatif) Pour ajouter des données supplémentaires, choisissez Ajouter des données en haut à droite.
   a. Choisissez Changer de source de données, puis choisissez une autre source de données.
   b. Suivez les instructions de l'interface utilisateur et terminez l'ajout de données.
   c. Après avoir ajouté de nouvelles données au même jeu de données, sélectionnez Configurer cette jointure (les deux points rouges). Configurez une jointure pour chaque tableau supplémentaire.
   d. Pour ajouter des champs calculés, Sélectionnez Ajouter un champ calculé.
   e. Pour utiliser Sagemaker, choisissez Augmenter avec. SageMaker Cette option n'est disponible que dans QuickSight l'édition Enterprise.
   f. Décochez les champs que vous souhaitez omettre.
   g. Mettez à jour les types de données que vous souhaitez modifier.
8. Lorsque vous avez terminé, sélectionnez Enregistrer pour enregistrer et fermer le jeu de données.

Créer une nouvelle analyse pour Timestream

1. Assurez-vous d'avoir configuré les autorisations appropriées pour qu'Amazon puisse accéder QuickSight à Amazon Timestream, comme décrit dans. Accès à Amazon Timestream depuis QuickSight
2. Choisissez Analyses sur la gauche.
3. Sélectionnez l'une des méthodes suivantes :
   - Pour créer une nouvelle analyse, choisissez Nouvelle analyse en haut à droite.
   - Pour ajouter le jeu de données Timestream à une analyse existante, ouvrez l'analyse que vous souhaitez modifier. Cliquez sur l'icône en forme de crayon en haut à gauche, puis sur Ajouter un ensemble de données.
4. Commencez la première visualisation des données en choisissant les champs sur la gauche.
5. Pour plus d'informations, consultez Travailler avec des analyses - Amazon QuickSight

Tutoriel vidéo

Cette vidéo explique comment Amazon QuickSight travaille avec Timestream.

Amazon SageMaker

Vous pouvez utiliser Amazon SageMaker Notebooks pour intégrer vos modèles de machine learning à Amazon Timestream. Pour vous aider à démarrer, nous avons créé un exemple de SageMaker bloc-notes qui traite les données de Timestream. Les données sont insérées dans Timestream à partir d'une application Python multithread qui envoie des données en continu. Le code source de l'exemple de SageMaker bloc-notes et de l'exemple d'application Python est disponible dans GitHub.

1. Créez une base de données et une table en suivant les instructions décrites dans Créer une base de données et Créer une table
2. Clonez le GitHub référentiel pour l'exemple d'application Python multithread en suivant les instructions de GitHub
3. Clonez le GitHub référentiel pour l'exemple de SageMaker bloc-notes Timestream en suivant les instructions de. GitHub
4. Exécutez l'application pour ingérer en continu des données dans Timestream en suivant les instructions du README
5. Suivez les instructions pour créer un compartiment Amazon S3 pour Amazon SageMaker , comme décrit ici.
6. Créez une SageMaker instance Amazon avec le dernier boto3 installé : en plus des instructions décrites ici, suivez les étapes ci-dessous :

    a. Sur la page Créer une instance de bloc-notes, cliquez sur Configuration supplémentaire

    b. Cliquez sur Configuration du cycle de vie (facultatif) et sélectionnez Créer une nouvelle configuration du cycle de vie

    c. Dans la boîte de l'assistant de création de configuration du cycle de vie, procédez comme suit :

        i. Entrez le nom de votre choix pour la configuration, par ex. `on-start`

        ii. Dans le script Start Notebook, copiez-collez le contenu du script depuis Github

        iii. Remplacez `PACKAGE=scipy` par `PACKAGE=boto3` dans le script collé.

7. Cliquez sur Créer une configuration

8. Accédez au IAM service dans la console de AWS gestion et recherchez le rôle SageMaker d'exécution nouvellement créé pour l'instance de bloc-notes.
9. Attachez la IAM politique `AmazonTimestreamFullAccess` pour au rôle d'exécution.

> **Note**
>
> La `AmazonTimestreamFullAccess` IAM politique n'est pas limitée à des ressources spécifiques et n'est pas adaptée à une utilisation en production. Pour un système de production, envisagez d'utiliser des politiques qui limitent l'accès à des ressources spécifiques.

10. Lorsque le statut de l'instance de bloc-notes est défini InService, choisissez Open Jupyter pour lancer un SageMaker bloc-notes pour l'instance
11. Téléchargez les fichiers `timestreamquery.py` et placez-les `Timestream_SageMaker_Demo.ipynb` dans le bloc-notes en sélectionnant le bouton Télécharger
12. Choisissez `Timestream_SageMaker_Demo.ipynb`

> **Note**
>
> Si vous voyez une fenêtre contextuelle indiquant Kernel not found, choisissez conda_python3 et cliquez sur Set Kernel.

13. Modifiez`DB_NAME`, `TABLE_NAMEbucket`, et `ENDPOINT` pour qu'ils correspondent au nom de la base de données, au nom de la table, au nom du compartiment S3 et à la région des modèles d'entraînement.
14. Choisissez l'icône de jeu pour exécuter les cellules individuelles
15. Lorsque vous arrivez dans la celluleLeverage `Timestream to find hosts with average CPU utilization across the fleet`, assurez-vous que la sortie renvoie au moins 2 noms d'hôtes.

> **Note**
>
> Si la sortie contient moins de 2 noms d'hôtes, vous devrez peut-être réexécuter l'exemple d'application Python ingérant des données dans Timestream avec un plus grand nombre de threads et une plus grande échelle d'hôte.

16. Lorsque vous arrivez dans la celluleTrain a Random Cut Forest (RCF) model using the CPU utilization history, modifiez le train_instance_type en fonction des besoins en ressources pour votre poste de formation
17. Lorsque vous arrivez à la celluleDeploy the model for inference, modifiez le en instance_type fonction des besoins en ressources pour votre tâche d'inférence

> **Note**
>
> L'entraînement du modèle peut prendre quelques minutes. Lorsque la formation est terminée, le message Terminé - Tâche de formation terminée s'affiche dans la sortie de la cellule.

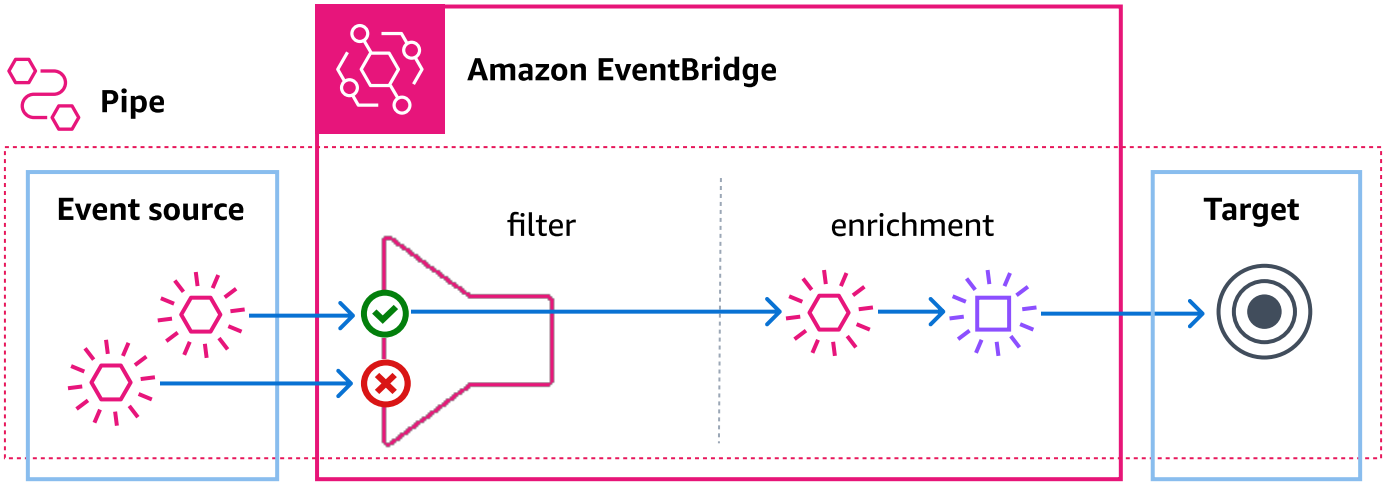
18. Exécutez la cellule Stop and delete the endpoint pour nettoyer les ressources. Vous pouvez également arrêter et supprimer l'instance de la SageMaker console

Amazon SQS

Utiliser EventBridge Pipes pour envoyer SQS des données Amazon à Timestream

Vous pouvez utiliser EventBridge Pipes pour envoyer des données d'une SQS file d'attente Amazon vers une table Amazon Timestream for LiveAnalytics .

Les tubes sont destinés aux point-to-point intégrations entre les sources et les cibles prises en charge, avec la prise en charge des transformations avancées et de l'enrichissement. Pipes réduit le besoin de connaissances spécialisées et de code d'intégration lors du développement d'architectures pilotées par des événements. Pour configurer un canal, vous devez choisir la source, ajouter un filtrage facultatif, définir un enrichissement facultatif et choisir la cible pour les données d'événement.



Pour plus d'informations sur les EventBridge tuyaux, voir EventBridge Tuyaux dans le guide de EventBridge l'utilisateur. Pour plus d'informations sur la configuration d'un canal pour transmettre des événements à un tableau Amazon Timestream LiveAnalytics for, EventBridge consultez la section Caractéristiques de la cible des canaux.

Utilisation DBeaver pour travailler avec Amazon Timestream

DBeaverest un SQL client universel gratuit qui peut être utilisé pour gérer n'importe quelle base de données dotée d'un JDBC pilote. Il est largement utilisé par les développeurs et les administrateurs de bases de données en raison de ses fonctionnalités robustes de visualisation, d'édition et de gestion des données.

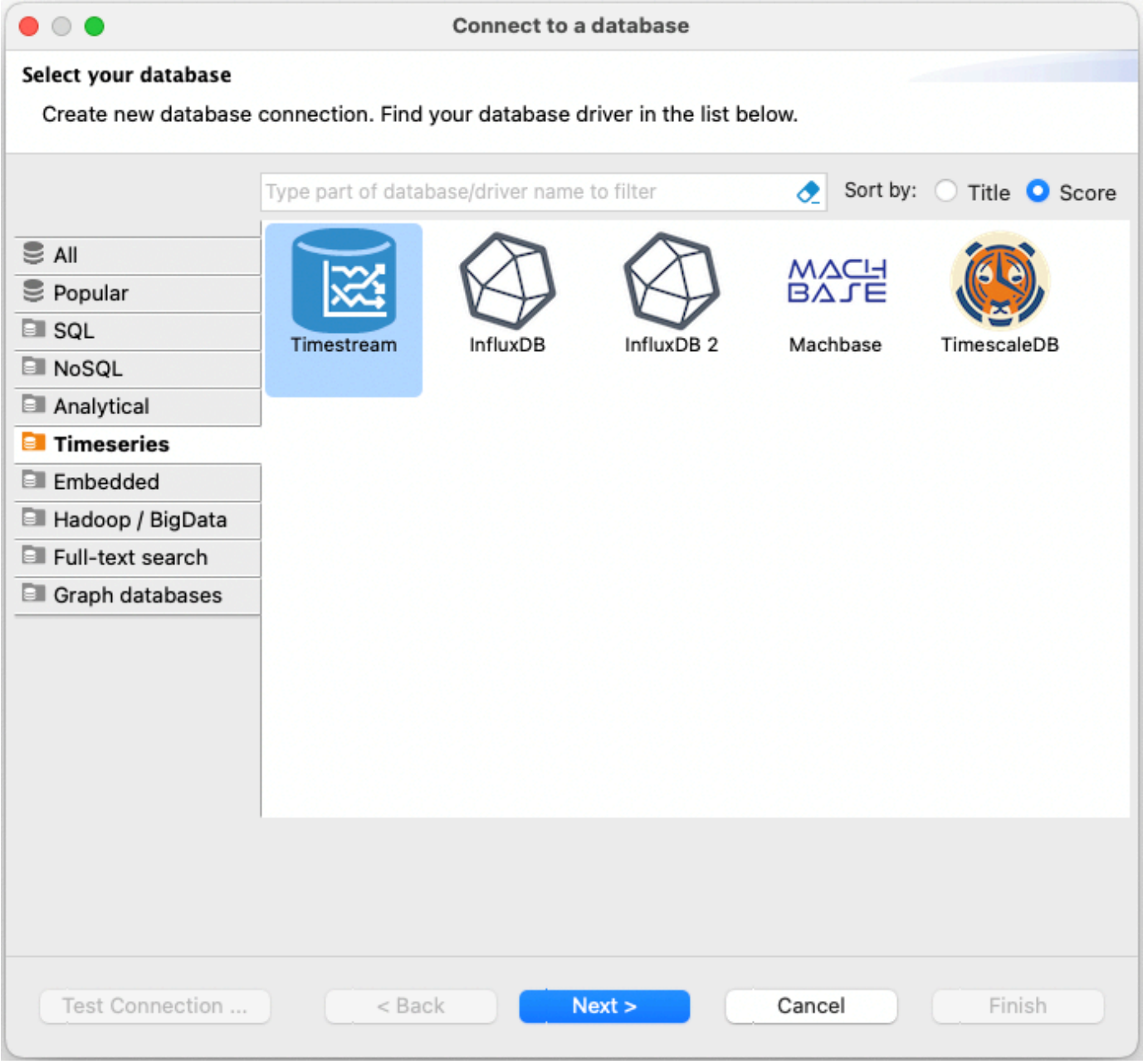
Grâce DBeaver aux options de connectivité au cloud, vous pouvez vous connecter DBeaver à Amazon Timestream de manière native. DBeaverfournit une interface complète et intuitive permettant de travailler avec des données de séries chronologiques directement depuis une DBeaver application. À l'aide de vos informations d'identification, il vous donne également un accès complet à toutes les requêtes que vous pourriez exécuter à partir d'une autre interface de requête. Il vous permet même de créer des graphiques pour mieux comprendre et visualiser les résultats des requêtes.

Configuration DBeaver pour fonctionner avec Timestream

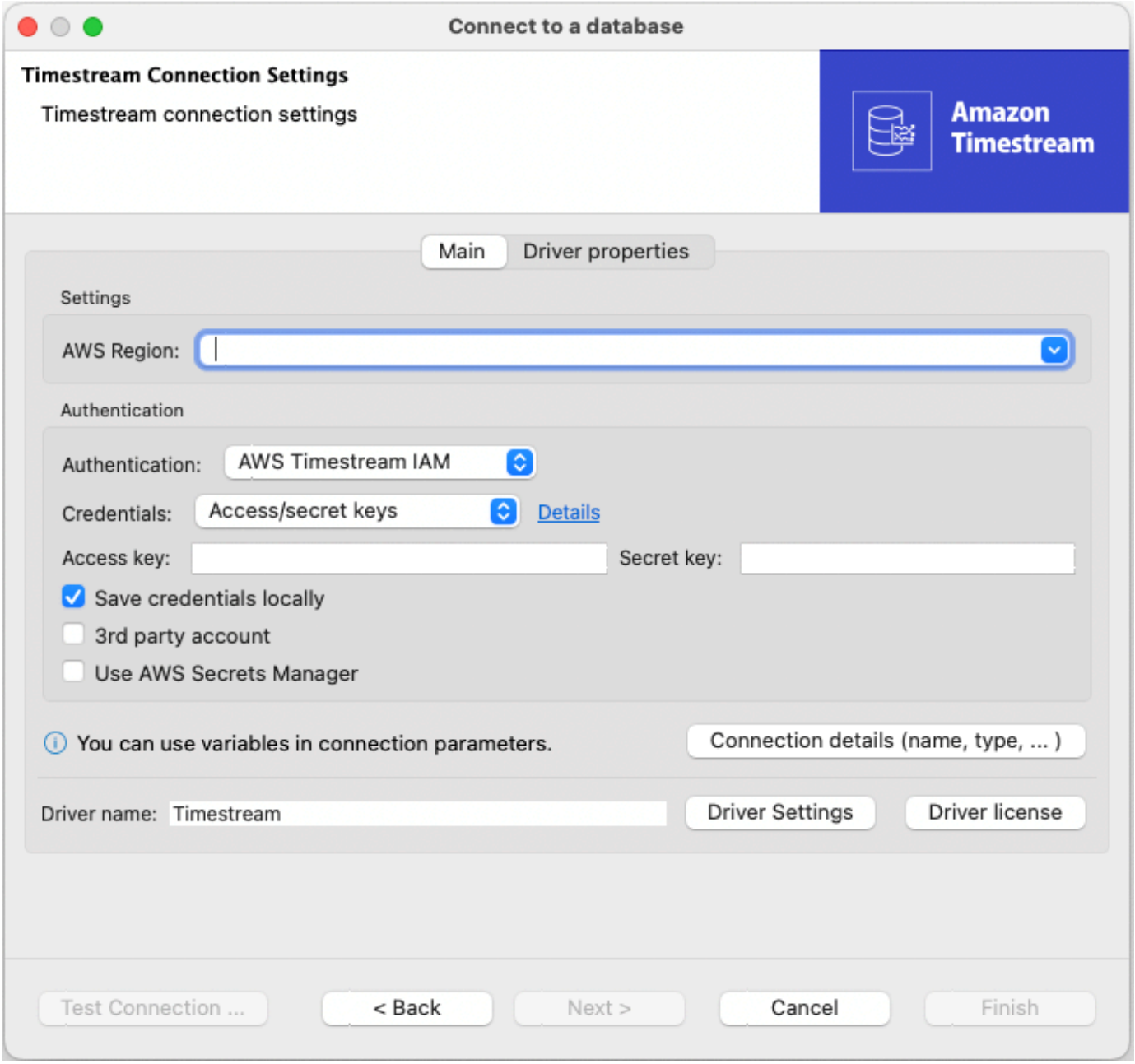
Procédez comme suit pour configurer Timestream DBeaver afin qu'il fonctionne avec Timestream :

1. Téléchargez et installez DBeaver sur votre ordinateur local.

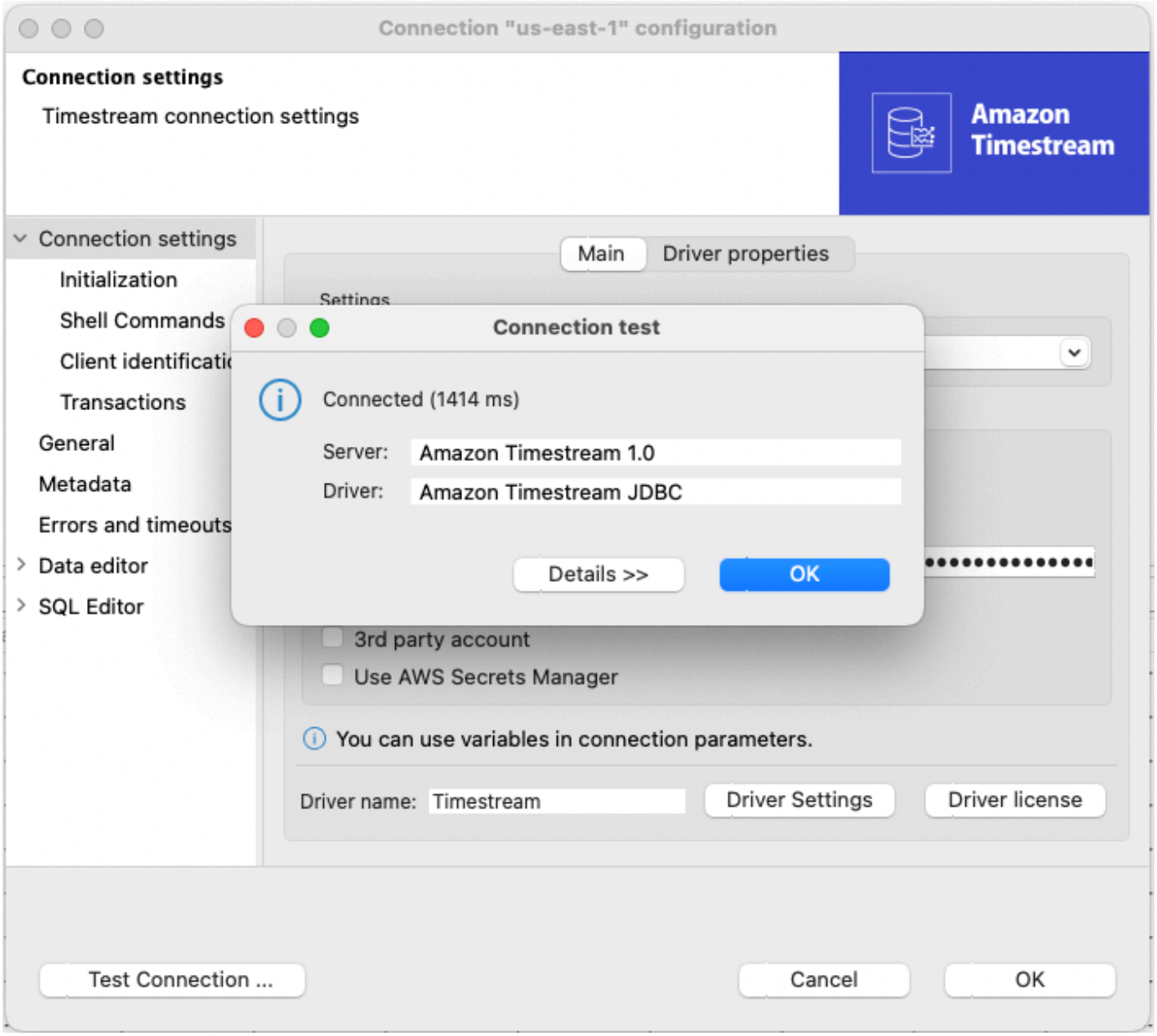
2. LancezDBeaver, accédez à la zone de sélection de la base de données, choisissez Timeseries dans le volet gauche, puis sélectionnez l'icône Timestream dans le volet droit :



3. Dans la fenêtre Paramètres de connexion Timestream, entrez toutes les informations nécessaires pour vous connecter à votre base de données Amazon Timestream. Assurez-vous que les clés utilisateur que vous entrez disposent des autorisations nécessaires pour accéder à votre base de données Timestream. Veillez également à ce que les informations et les clés que vous DBeaver saisissez restent confidentielles, comme pour toute information sensible.



4. Testez la connexion pour vous assurer que tout est correctement configuré :



5. Si le test de connexion est réussi, vous pouvez désormais interagir avec votre base de données Amazon Timestream comme vous le feriez avec n'importe quelle autre base de données. DBeaver Par exemple, vous pouvez accéder à l'SQLéditeur ou à la vue du diagramme ER pour exécuter des requêtes :

6. DBeaverfournit également de puissants outils de visualisation des données. Pour les utiliser, lancez votre requête, puis sélectionnez l'icône graphique pour visualiser le jeu de résultats. L'outil graphique peut vous aider à mieux comprendre les tendances des données au fil du temps.

Associer Amazon Timestream à Amazon DBeaver Timestream crée un environnement efficace pour gérer les données de séries chronologiques. Vous pouvez l'intégrer facilement à votre flux de travail existant pour améliorer la productivité et l'efficacité.

Grafana

Vous pouvez visualiser les données de vos séries chronologiques et créer des alertes à l'aide de Grafana. Pour vous aider à démarrer avec la visualisation des données, nous avons créé un exemple de tableau de bord dans Grafana qui permet de visualiser les données envoyées à Timestream depuis une application Python, ainsi qu'un didacticiel vidéo décrivant la configuration.

Rubriques

Exemple d'application

1. Créez une base de données et une table dans Timestream en suivant les instructions décrites dans Créer une base de données pour plus d'informations.

   > **Note**
   >
   > Le nom de base de données et le nom de table par défaut pour le tableau de bord Grafana sont respectivement définis sur GrafanaDB. grafanaTable Utilisez ces noms pour minimiser la configuration.

2. Installation de Python 3.7 ou supérieur
3. Installation et configuration du Python Timestream SDK
4. Clonez le GitHub référentiel pour l'application Python multithread en ingérant continuellement des données dans Timestream en suivant les instructions de GitHub
5. Exécutez l'application pour ingérer en continu des données dans Timestream en suivant les instructions du README
6. Terminez la mise en route avec Amazon Managed Grafana ou terminez l'installation de Grafana.
7. Si vous installez Grafana au lieu d'utiliser Amazon Managed Grafana, terminez l'installation du plugin Timestream pour Grafana.

8. Ouvrez le tableau de bord Grafana à l'aide du navigateur de votre choix. Si vous avez installé Grafana localement, vous pouvez suivre les instructions décrites dans la documentation de Grafana pour vous connecter
9. Après avoir lancé Grafana, allez dans Datasources, cliquez sur Ajouter une source de données, recherchez Timestream et sélectionnez la source de données Timestream
10. Configurez le fournisseur d'authentification et la région, puis cliquez sur Enregistrer et tester
11. Définissez les macros par défaut
    a. Définissez $__database sur le nom de votre base de données Timestream (par exemple GrafanaDB)
    b. Définissez $__table sur le nom de votre table Timestream (par exemple) grafanaTable
    c. Définissez $__measure sur la mesure la plus couramment utilisée dans le tableau
12. Cliquez sur Enregistrer et tester
13. Cliquez sur l'onglet Tableaux de bord
14. Cliquez sur Importer pour importer le tableau de bord
15. Double-cliquez sur le tableau de bord de l'exemple d'application
16. Cliquez sur les paramètres du tableau de bord
17. Sélectionnez les variables
18. Modifier dbName et tableName faire correspondre les noms de la base de données et de la table Timestream
19. Cliquez sur Enregistrer
20. Actualiser le tableau de bord
21. Pour créer des alertes, suivez les instructions décrites dans la documentation de Grafana pour créer une règle d'alerte gérée par Grafana
22. Pour résoudre les problèmes liés aux alertes, suivez les instructions décrites dans la documentation de Grafana pour le dépannage
23. Pour plus d'informations, consultez la documentation de Grafana

Tutoriel vidéo

Cette vidéo explique comment Grafana fonctionne avec Timestream.

Utilisation SquaredUp pour travailler avec Amazon Timestream

SquaredUpest une plateforme d'observabilité qui s'intègre à Amazon Timestream. Vous pouvez utiliser SquaredUp le concepteur de tableau de bord intuitif pour visualiser, analyser et surveiller vos données chronologiques. Les tableaux de bord peuvent être partagés en public ou en privé, et des canaux de notification peuvent être créés pour vous avertir lorsque l'état de santé d'un moniteur change.

Utilisation SquaredUp avec Amazon Timestream

1. Inscrivez-vous SquaredUpet commencez gratuitement.
2. Ajoutez une source AWS de données.
3. Créez une vignette de tableau de bord qui utilise le flux de données Timestream Query.
4. Activez éventuellement la surveillance de la vignette, créez un canal de notification ou partagez le tableau de bord en public ou en privé.
5. Créez éventuellement d'autres vignettes pour afficher vos données Timestream aux côtés des données de vos autres outils de surveillance et d'observabilité.

Telegraf open source

Vous pouvez utiliser le plugin Timestream for LiveAnalytics output pour Telegraf afin d'écrire des métriques dans Timestream directement à partir de Telegraf open source. LiveAnalytics

Cette section explique comment installer Telegraf avec le plugin Timestream for LiveAnalytics output, comment exécuter Telegraf avec le plugin Timestream for LiveAnalytics output et comment Telegraf open source fonctionne avec Timestream for. LiveAnalytics

Rubriques

- Installation de Telegraf avec le plugin Timestream for output LiveAnalytics
- Exécution de Telegraf avec le plugin Timestream pour la sortie LiveAnalytics
- Mappage des métriques Telegraf/InfluxDB au Timestream du modèle LiveAnalytics

Installation de Telegraf avec le plugin Timestream for output LiveAnalytics

Depuis la version 1.16, le plugin Timestream for LiveAnalytics output est disponible dans la version officielle de Telegraf. Pour installer le plugin de sortie sur la plupart des principaux systèmes

d'exploitation, suivez les étapes décrites dans la documentation InfluxData Telegraf. Pour l'installer sur le système d'exploitation Amazon Linux 2, suivez les instructions ci-dessous.

Installation de Telegraf avec le LiveAnalytics plugin Timestream for output sur Amazon Linux 2

Pour installer Telegraf avec le plug-in de sortie Timestream sur Amazon Linux 2, effectuez les étapes suivantes.

1. Installez Telegraf à l'aide du gestionnaire de yum paquets.

```
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. Exécutez la commande suivante.

```
sudo sed -i "s/\$releasever/$(rpm -E %{rhel})/g" /etc/yum.repos.d/influxdb.repo
```

3. Installez et démarrez Telegraf.

```
sudo yum install telegraf
sudo service telegraf start
```

Exécution de Telegraf avec le plugin Timestream pour la sortie LiveAnalytics

Vous pouvez suivre les instructions ci-dessous pour exécuter Telegraf avec le plugin Timestream for. LiveAnalytics

1. Générez un exemple de configuration à l'aide de Telegraf.

```
telegraf --section-filter agent:inputs:outputs --input-filter cpu:mem --output-
filter timestream config > example.config
```

2. Créez une base de données dans Timestream à l'aide de la console de gestion CLI, ou. SDKs

3. Dans le `example.config` fichier, ajoutez le nom de votre base de données en modifiant la clé suivante sous la `[[outputs.timestream]]` section.

```
database_name = "yourDatabaseNameHere"
```

4. Par défaut, Telegraf créera une table. Si vous souhaitez créer une table manuellement, définissez `create_table_if_not_exists` `false` et suivez les instructions pour créer une table à l'aide de la console de gestion CLI, ou SDKs.

5. Dans le fichier exemple.config, configurez les informations d'identification dans la `[[outputs.timestream]]` section. Les informations d'identification doivent permettre les opérations suivantes.

```
timestream:DescribeEndpoints
timestream:WriteRecords
```

> **Note**
>
> Si vous laissez ce `create_table_if_not_exists` paramètre défini sur`true`, incluez :
>
> ```
> timestream:CreateTable
> ```

> **Note**
>
> Si vous définissez cette `describe_database_on_start` option`true`, incluez les éléments suivants.
>
> ```
> timestream:DescribeDatabase
> ```

6. Vous pouvez modifier le reste de la configuration selon vos préférences.

7. Lorsque vous avez terminé de modifier le fichier de configuration, lancez Telegraf avec ce qui suit.

```
./telegraf --config example.config
```

8. Les métriques devraient apparaître au bout de quelques secondes, selon la configuration de votre agent. Vous devriez également voir les nouvelles tables, cpu et mem, dans la console Timestream.

Mappage des métriques Telegraf/InfluxDB au Timestream du modèle LiveAnalytics

Lorsque vous écrivez des données de Telegraf vers Timestream for LiveAnalytics, les données sont mappées comme suit.

- L'horodatage est écrit sous forme de champ horaire.
- Les balises sont écrites sous forme de dimensions.
- Les champs sont écrits sous forme de mesures.
- Les mesures sont principalement écrites sous forme de noms de tables (plus d'informations à ce sujet ci-dessous).

Le plugin Timestream for LiveAnalytics output pour Telegraf offre plusieurs options pour organiser et stocker les données dans Timestream for. LiveAnalytics Cela peut être décrit par un exemple qui commence par les données au format de protocole de ligne.

```
weather,location=us-midwest,season=summer temperature=82,humidity=71
1465839830100400200 airquality,location=us-west no2=5,pm25=16
1465839830100400200
```

Ce qui suit décrit les données.

- Les noms des mesures sont `weather` et`airquality`.
- Les tags sont `location` et`season`.
- Les champs sont `temperature`,`humidity`,`no2`, et`pm25`.

Rubriques

- Stockage des données dans plusieurs tables
- Stockage des données dans une seule table

Stockage des données dans plusieurs tables

Vous pouvez choisir de créer un tableau distinct par mesure et de stocker chaque champ dans une ligne distincte par tableau.

La configuration est`mapping_mode = "multi-table"`.

- L' LiveAnalytics adaptateur Timestream for créera deux tables, à savoir, et`weather`. `airquality`
- Chaque ligne du tableau ne contiendra qu'un seul champ.

Le Timestream qui en résultera pour LiveAnalytics les tables `weather` et`airquality`, ressemblera à ceci.

`weather`

time	location	saison	nom_mesure	value_mes ure : :bigint
13/06/2016 17:43:50	Midwest américain	été	temperature	82
13/06/2016 17:43:50	Midwest américain	été	humidité	71

`airquality`

time	location	nom_mesure	value_mesure : :bigint
13/06/2016 17:43:50	Midwest américain	n° 2	5
13/06/2016 17:43:50	Midwest américain	pm25	16

Stockage des données dans une seule table

Vous pouvez choisir de stocker toutes les mesures dans un seul tableau et de stocker chaque champ dans une ligne de tableau distincte.

La configuration est`mapping_mode = "single-table"`. Il existe deux configurations supplémentaires lors de l'utilisation`single-table`, `single_table_name` et`single_table_dimension_name_for_telegraf_measurement_name`.

- Le plugin Timestream for LiveAnalytics output créera une seule table avec un nom *<single_table_name>* qui inclut un *<single_table_dimension_name_for_telegraf_measurement_name>* colonne.
- La table peut contenir plusieurs champs dans une seule ligne.

Le Timestream qui en résultera pour le LiveAnalytics tableau ressemblera à ceci.

`weather`

time	location	saison	***<single_table_dimension_name_for_telegraf_measurement_name>***	nom_mesure	value_mesure : :bigint
13/06/2016 17:43:50	Midwest américain	été	météo	temperature	82
13/06/2016 17:43:50	Midwest américain	été	météo	humidité	71
13/06/2016 17:43:50	Midwest américain	été	qualité de l'air	n° 2	5
13/06/2016 17:43:50	Midwest américain	été	météo	pm25	16

JDBC

Vous pouvez utiliser une connexion Java Database Connectivity (JDBC) pour connecter Timestream LiveAnalytics à vos outils de business intelligence et à d'autres applications, telles que SQL

Workbench. Le Timestream for LiveAnalytics JDBC driver est actuellement compatible SSO avec Okta et Microsoft Azure AD.

Rubriques

- Configuration du JDBC pilote pour Timestream pour LiveAnalytics
- Propriétés de connexion
- JDBCURLexemples
- Configuration de Timestream pour l'authentification LiveAnalytics JDBC unique avec Okta
- Configuration de Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD

Configuration du JDBC pilote pour Timestream pour LiveAnalytics

Suivez les étapes ci-dessous pour configurer le JDBC pilote.

Rubriques

- Timestream pour le chauffeur LiveAnalytics JDBC JARs
- Diffusion chronologique pour la classe et LiveAnalytics JDBC le format du pilote URL
- Exemple d'application

Timestream pour le chauffeur LiveAnalytics JDBC JARs

Vous pouvez obtenir le Timestream pour le LiveAnalytics JDBC pilote par téléchargement direct ou en ajoutant le pilote en tant que dépendance Maven.

- En téléchargement direct :. Pour télécharger directement le Timestream pour le LiveAnalytics JDBC pilote, procédez comme suit :

  1. Accédez à https://github.com/awslabs/amazon-timestream-driver-jdbc/releases
  2. Vous pouvez utiliser `amazon-timestream-jdbc-1.0.1-shaded.jar` directement avec vos outils et applications de business intelligence
  3. `amazon-timestream-jdbc-1.0.1-javadoc.jar`Téléchargez-le dans le répertoire de votre choix.
  4. Dans le répertoire où vous avez effectué le téléchargement`amazon-timestream-jdbc-1.0.1-javadoc.jar`, exécutez la commande suivante pour extraire les fichiers Javadoc HTML :

```
jar -xvf amazon-timestream-jdbc-1.0.1-javadoc.jar
```

- En tant que dépendance Maven : pour ajouter le Timestream for LiveAnalytics JDBC driver en tant que dépendance Maven, procédez comme suit :

  1. Accédez au `pom.xml` fichier de votre application et ouvrez-le dans l'éditeur de votre choix.
  2. Ajoutez le JDBC pilote en tant que dépendance dans le `pom.xml` fichier de votre application :

```
<!-- 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>
```

Diffusion chronologique pour la classe et LiveAnalytics JDBC le format du pilote URL

La classe de pilote pour Timestream for LiveAnalytics JDBC driver est la suivante :

```
software.amazon.timestream.jdbc.TimestreamDriver
```

Le JDBC pilote Timestream nécessite le format suivant : JDBC URL

```
jdbc:timestream:
```

Pour spécifier les propriétés de base de données par le biais du JDBCURL, utilisez le URL format suivant :

```
jdbc:timestream://
```

Exemple d'application

Pour vous aider à commencer à utiliser Timestream for LiveAnalytics withJDBC, nous avons créé un exemple d'application entièrement fonctionnel dans. GitHub

1. Créez une base de données avec des exemples de données en suivant les instructions décrites ici.

2. Clonez le GitHub référentiel pour l'exemple d'application en JDBC suivant les instructions de GitHub.
3. Suivez les instructions du READMEpour commencer à utiliser l'exemple d'application.

Propriétés de connexion

Le Timestream for LiveAnalytics JDBC driver prend en charge les options suivantes :

Rubriques

- Options d'authentification de base
- Option standard d'informations sur le client
- Option de configuration du pilote
- SDKoption
- Option de configuration du terminal
- Options du fournisseur d'informations d'identification
- SAMLoptions d'authentification basées sur Okta
- SAMLoptions d'authentification basées sur Azure AD

> **Note**
>
> Si aucune des propriétés n'est fournie, le Timestream for LiveAnalytics JDBC driver utilisera la chaîne d'informations d'identification par défaut pour charger les informations d'identification.

> **Note**
>
> Toutes les clés de propriété distinguent les majuscules et minuscules.

Options d'authentification de base

Le tableau suivant décrit les options d'authentification de base disponibles.

Option	Description	Par défaut
AccessKeyId	L' AWS identifiant de la clé d'accès utilisateur.	NONE
SecretAccessKey	La clé d'accès secrète de l' AWS utilisateur.	NONE
SessionToken	Le jeton de session temporaire requis pour accéder à une base de données avec l'authentification multifactorielle (MFA) activée.	NONE

Option standard d'informations sur le client

Le tableau suivant décrit l'option d'informations client standard.

Option	Description	Par défaut
ApplicationName	Nom de l'application qui utilise actuellement la connexion . `ApplicationName` est utilisé à des fins de débogage et ne sera pas communiqué au Timestream à des fins de maintenance. LiveAnalytics	Le nom de l'application détecté par le pilote.

Option de configuration du pilote

Le tableau suivant décrit l'option de configuration du pilote.

Option	Description	Par défaut
EnableMetaDataPreparedStatement	Permet à Timestream pour LiveAnalytics JDBC le pilote de renvoyer des métadonné	FALSE

Option	Description	Par défaut
	esPreparedStatements , mais cela entraînera un coût supplémentaire pour Timestream lors de la récupération des métadonnées. LiveAnalytics	
Région	Région de la base de données.	us-east-1

SDKoption

Le tableau suivant décrit l'SDKoption.

Option	Description	Par défaut
RequestTimeout	Le temps, en millisecondes, pendant lequel une demande de AWS SDK requête est attendue avant l'expiration du délai imparti. Une valeur non positive désactive le délai d'expiration de la demande.	0
SocketTimeout	Durée en millisecondes pendant laquelle les données AWS SDK seront transférées via une connexion ouverte avant l'expiration du délai imparti. La valeur ne doit pas être négative. La valeur 0 désactive le délai d'expiration du socket.	50000
MaxRetryCountClient	Nombre maximal de tentatives pour les erreurs réessayables	NONE

Option	Description	Par défaut
	contenant 5XX codes d'erreur dans le. SDK La valeur ne doit pas être négative.	
MaxConnections	Le nombre maximum de HTTP connexions ouvertes simultanément au Timestream pour le service. LiveAnalytics La valeur doit être positive.	50

Option de configuration du terminal

Le tableau suivant décrit l'option de configuration du point de terminaison.

Option	Description	Par défaut
Point de terminaison	Le point de terminaison du service Timestream for LiveAnalytics .	NONE

Options du fournisseur d'informations d'identification

Le tableau suivant décrit les options de fournisseur d'informations d'identification disponibles.

Option	Description	Par défaut
AwsCredentialsProviderClass	L'un des `PropertiesFileCredentialsProvider` ou `InstanceProfileCredentialsProvider` à utiliser pour l'authentification.	NONE
CustomCredentialsFilePath	Le chemin d'accès à un fichier de propriétés contenant	NONE

Option	Description	Par défaut
	les informations d'identification AWS de sécurité `accessKey` et `secretKey`. Ceci n'est obligatoire que s'`AwsCredentialsProviderClass` il est spécifié comme `PropertiesFileCredentialsProvider`.	

SAMLoptions d'authentification basées sur Okta

Le tableau suivant décrit les options d'authentification SAML basées disponibles pour Okta.

Option	Description	Par défaut
IdpName	Le nom du fournisseur d'identité (Idp) à utiliser pour l'authentification SAML basée. L'un des `Okta` ou `AzureAD`	NONE
IdpHost	Le nom d'hôte de l'Idp spécifié.	NONE
IdpUserName	Le nom d'utilisateur du compte Idp spécifié.	NONE
IdpPassword	Le mot de passe du compte Idp spécifié.	NONE
OktaApplicationID	L'identifiant unique fourni par Okta associé au Timestream pour l'application. LiveAnalytics `AppId`se trouve dans le `entityID` champ prévu à cet effet dans les métadonné	NONE

Option	Description	Par défaut
	es de l'application. Prenons l'exemple suivant : `entityID = http://www.okta.com//IdpAppID`	
Rôle ARN	Le nom de ressource Amazon (ARN) du rôle assumé par l'appelant.	NONE
Idp ARN	Le nom de ressource Amazon (ARN) du SAML fournisseur IAM qui décrit l'Idp.	NONE

SAMLoptions d'authentification basées sur Azure AD

Le tableau suivant décrit les options d'authentification SAML basées disponibles pour Azure AD.

Option	Description	Par défaut
IdpName	Le nom du fournisseur d'identité (Idp) à utiliser pour l'authentification SAML basée. L'un des `Okta` ou `AzureAD`	NONE
IdpHost	Le nom d'hôte de l'Idp spécifié.	NONE
IdpUserName	Le nom d'utilisateur du compte Idp spécifié.	NONE
IdpPassword	Le mot de passe du compte Idp spécifié.	NONE
AADApplicationID	L'identifiant unique de l'application enregistrée sur Azure AD.	NONE

Option	Description	Par défaut
AADClientSecret	Le secret client associé à l'application enregistrée sur Azure AD est utilisé pour autoriser l'extraction de jetons.	NONE
AADTenant	L'identifiant du locataire Azure AD.	NONE
Idp ARN	Le nom de ressource Amazon (ARN) du SAML fournisseur IAM qui décrit l'Idp.	NONE

JDBCURLexemples

Cette section décrit comment créer une JDBC connexion URL et fournit des exemples. Pour définir les propriétés de connexion facultatives, utilisez le URL format suivant :

```
jdbc:timestream://PropertyName1=value1;PropertyName2=value2...
```

> **Note**
>
> Toutes les propriétés de connexion sont facultatives. Toutes les clés de propriété distinguent les majuscules et minuscules.

Vous trouverez ci-dessous quelques exemples de JDBC connexionURLs.

Exemple avec les options d'authentification de base et la région :

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
east-1
```

Exemple avec les informations sur le client, la région et SDK les options :

```
jdbc:timestream://ApplicationName=MyApp;Region=us-
east-1;MaxRetryCountClient=10;MaxConnections=5000;RequestTimeout=20000
```

Connectez-vous à l'aide de la chaîne de fournisseurs d'informations d'identification par défaut avec les AWS informations d'identification définies dans les variables d'environnement :

```
jdbc:timestream
```

Connectez-vous à l'aide de la chaîne de fournisseurs d'informations d'identification par défaut avec les AWS informations d'identification définies dans la connexion : URL

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
```

Connectez-vous à l'aide PropertiesFileCredentialsProvider de la méthode d'authentification suivante :

```
jdbc:timestream://
AwsCredentialsProviderClass=PropertiesFileCredentialsProvider;CustomCredentialsFilePath=<pat
 to properties file>
```

Connectez-vous à l'aide InstanceProfileCredentialsProvider de la méthode d'authentification suivante :

```
jdbc:timestream://AwsCredentialsProviderClass=InstanceProfileCredentialsProvider
```

Connectez-vous en utilisant les informations d'identification Okta comme méthode d'authentification :

```
jdbc:timestream://
IdpName=Okta;IdpHost=<host>;IdpUserName=<name>;IdpPassword=<password>;OktaApplicationID=<id>
```

Connectez-vous en utilisant les informations d'identification Azure AD comme méthode d'authentification :

```
jdbc:timestream://
IdpName=AzureAD;IdpUserName=<name>;IdpPassword=<password>;AADApplicationID=<id>;AADClientSec
```

Connectez-vous à un point de terminaison spécifique :

```
jdbc:timestream://Endpoint=abc.us-east-1.amazonaws.com;Region=us-east-1
```

Configuration de Timestream pour l'authentification LiveAnalytics JDBC unique avec Okta

Timestream for LiveAnalytics prend en charge Timestream pour l'authentification LiveAnalytics JDBC unique avec Okta. Pour utiliser Timestream pour l'authentification LiveAnalytics JDBC unique avec Okta, complétez chacune des sections répertoriées ci-dessous.

Rubriques

- Prérequis
- AWS fédération de comptes dans Okta
- Configuration d'Okta pour SAML

Prérequis

Assurez-vous de remplir les conditions préalables suivantes avant d'utiliser le Timestream pour l'authentification LiveAnalytics JDBC unique avec Okta :

- Autorisations d'administrateur AWS pour créer le fournisseur d'identité et les rôles.
- Un compte Okta (Accédez à https://www.okta.com/login/pour créer un compte).
- Accès à Amazon LiveAnalytics Timestream pour.

Maintenant que vous avez rempli les prérequis, vous pouvez passer àAWS fédération de comptes dans Okta.

AWS fédération de comptes dans Okta

Le Timestream pour le LiveAnalytics JDBC chauffeur prend en charge la fédération de AWS comptes dans Okta. Pour configurer la fédération de AWS comptes dans Okta, procédez comme suit :

1. Connectez-vous au tableau de bord Okta Admin à l'aide des méthodes suivantes : URL

```
https://<company-domain-name>-admin.okta.com/admin/apps/active
```

> **Note**
>
> Remplacez < company-domain-name > par votre nom de domaine.

2. Une fois la connexion établie, choisissez Ajouter une application et recherchez AWS Account Federation.
3. Choisissez Ajouter
4. Remplacez le nom URL d'utilisateur par le nom d'utilisateur appropriéURL.
5. Choisissez Next (Suivant)
6. Choisissez SAML2.0 comme méthode de connexion
7. Choisissez les métadonnées du fournisseur d'identité pour ouvrir le XML fichier de métadonnées. Enregistrez le fichier au niveau local.
8. Laissez toutes les autres options de configuration vides.
9. Choisissez Terminé.

Maintenant que vous avez terminé la fédération de AWS comptes dans Okta, vous pouvez passer àConfiguration d'Okta pour SAML.

Configuration d'Okta pour SAML

1. Choisissez l'onglet Sign On (Se connecter). Choisissez la vue.
2. Cliquez sur le bouton Instructions de configuration dans la section Paramètres.

Trouver le document de métadonnées Okta

1. Pour trouver le document, rendez-vous sur :

```
https://<domain>-admin.okta.com/admin/apps/active
```

> **Note**
>
> <domain>est le nom de domaine unique de votre compte Okta.

2. Choisissez l'application AWS Account Federation
3. Choisissez l'onglet Se connecter

Configuration de Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD

Timestream for LiveAnalytics prend en charge Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD. Pour utiliser Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD, complétez chacune des sections répertoriées ci-dessous.

Rubriques

Prérequis

Assurez-vous de remplir les conditions préalables suivantes avant d'utiliser le Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD :

- Autorisations d'administrateur AWS pour créer le fournisseur d'identité et les rôles.
- Un compte Azure Active Directory (Accédez à https://azure.microsoft.com/en-ca/services/active-directory/ pour créer un compte)
- Accès à Amazon LiveAnalytics Timestream pour.

Configuration d'Azure AD

1. Connectez-vous au portail Azure
2. Choisissez Azure Active Directory dans la liste des services Azure. Cela redirigera vers la page du répertoire par défaut.
3. Choisissez Applications d'entreprise dans la section Gérer de la barre latérale
4. Choisissez + Nouvelle application.
5. Recherchez et sélectionnez Amazon Web Services.
6. Choisissez Single Sign-On dans la section Gérer de la barre latérale
7. Choisissez SAMLcomme méthode d'authentification unique
8. Dans la section SAML Configuration de base, entrez les informations suivantes URL pour l'identifiant et pour la réponse URL :

```
https://signin.aws.amazon.com/saml
```

9. Choisissez Enregistrer.
10. Téléchargez les métadonnées de fédération XML dans la section Certificat de SAML signature. Cela sera utilisé ultérieurement lors de la création du fournisseur IAM d'identité
11. Retournez à la page du répertoire par défaut et choisissez Inscriptions d'applications sous Gérer.
12. Choisissez Timestream pour dans la LiveAnalytics section Toutes les applications. La page sera redirigée vers la page d'aperçu de l'application

> **Note**
>
> Notez l'ID de l'application (client) et l'ID du répertoire (tenant). Ces valeurs sont obligatoires lors de la création d'une connexion.

13. Choisissez des certificats et des secrets
14. Sous Secrets client, créez un nouveau secret client avec + Nouveau secret client.

> **Note**
>
> Notez le secret client généré, car il est obligatoire lors de la création d'une connexion à Timestream pour. LiveAnalytics

15. Dans la barre latérale, sous Gérer, sélectionnez autorisations API
16. Dans les autorisations configurées, utilisez Ajouter une autorisation pour accorder à Azure AD l'autorisation de se connecter à Timestream pour. LiveAnalytics Choisissez Microsoft Graph sur la page Demande API d'autorisations.
17. Choisissez Autorisations déléguées et sélectionnez l'autorisation User.Read
18. Choisissez Ajouter des autorisations
19. Choisissez Accorder le consentement de l'administrateur pour le répertoire par défaut

Configuration du fournisseur IAM d'identité et des rôles dans AWS

Complétez chaque section ci-dessous IAM pour configurer Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD :

Rubriques

- Création d'un fournisseur SAML d'identité
- créer un rôle IAM ;
- Création d'une IAM politique
- Allouer

Création d'un fournisseur SAML d'identité

Pour créer un fournisseur SAML d'identité pour le Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD, procédez comme suit :

1. Connectez-vous à la console AWS de gestion
2. Choisissez Services et sélectionnez la IAMsection Sécurité, identité et conformité
3. Choisissez les fournisseurs d'identité sous Gestion des accès
4. Choisissez Create Provider et choisissez SAMLcomme type de fournisseur. Entrez le nom du fournisseur. Cet exemple utilisera zureADProvider A.
5. Téléchargez le XML fichier de métadonnées de fédération précédemment téléchargé
6. Choisissez Next, puis Create.
7. Une fois terminée, la page sera redirigée vers la page des fournisseurs d'identité

créer un rôle IAM ;

Pour créer un IAM rôle pour le Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD, procédez comme suit :

1. Dans la barre latérale, sélectionnez Rôles sous Gestion des accès
2. Sélectionnez Créer un rôle.
3. Choisissez la fédération SAML 2.0 comme entité de confiance
4. Choisissez le fournisseur Azure AD
5. Choisissez Autoriser l'accès à la programmation et à AWS la console de gestion
6. Choisissez Suivant : Autorisations
7. Joignez des politiques d'autorisation ou passez à Next:Tags
8. Ajoutez des balises facultatives ou passez à Next:Review
9. Entrez un nom de rôle. Cet exemple utilisera A zureSAMLRole

10. Fournir une description du rôle
11. Choisissez Créer un rôle pour terminer

Création d'une IAM politique

Pour créer une IAM politique pour le Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD, procédez comme suit :

1. Dans la barre latérale, choisissez Politiques sous Gestion des accès
2. Choisissez Créer une politique et sélectionnez l'JSONonglet
3. Ajoutez la politique suivante

```
{
"Version": "2012-10-17",
"Statement": [
    {
          "Effect": "Allow",
          "Action": [
                "iam:ListRoles",
                "iam:ListAccountAliases"
           ],
           "Resource": "*"
      }
]
}
```

4. Choisir Create policy (Créer une stratégie)
5. Entrez un nom de stratégie. Cet exemple utilisera TimestreamAccessPolicy.
6. Choisissez Créer une politique
7. Dans la barre latérale, sélectionnez Rôles sous Gestion des accès.
8. Choisissez le rôle Azure AD créé précédemment et choisissez Attacher des politiques sous Autorisations.
9. Sélectionnez la politique d'accès créée précédemment.

Allouer

Pour configurer le fournisseur d'identité pour Timestream pour l'authentification LiveAnalytics JDBC unique avec Microsoft Azure AD, procédez comme suit :

1. Retourner au portail Azure
2. Choisissez Azure Active Directory dans la liste des services Azure. Cela redirigera vers la page du répertoire par défaut
3. Choisissez Applications d'entreprise dans la section Gérer de la barre latérale
4. Choisissez Provisioning
5. Choisissez le mode automatique pour la méthode de provisionnement
6. Sous Identifiants d'administrateur, entrez votre AwsAccessKeyidentifiant pour clientsecret et SecretAccessKeypour Secret Token
7. Définissez le statut de provisionnement sur Activé
8. Choisissez Enregistrer. Cela permet à Azure AD de charger les IAM rôles nécessaires
9. Une fois l'état du cycle en cours terminé, choisissez Utilisateurs et groupes dans la barre latérale
10. Choisissez + Ajouter un utilisateur
11. Choisissez l'utilisateur Azure AD pour lequel vous souhaitez donner accès à Timestream pour LiveAnalytics
12. Choisissez le rôle IAM Azure AD et le fournisseur d'identité Azure correspondant créé dans AWS
13. Choisissez Attribuer

ODBC

Le ODBCpilote open source d'Amazon LiveAnalytics Timestream for SQL fournit une interface relationnelle à Timestream for pour les développeurs et permet la connectivité à LiveAnalytics partir d'outils de business intelligence (BI) tels que Power BI Desktop et Microsoft Excel. Le LiveAnalytics ODBC pilote Timestream for est actuellement disponible sous Windows, macOS et Linux, et est également compatible SSO avec Okta et Microsoft Azure Active Directory (AD).

Pour plus d'informations, consultez Amazon Timestream LiveAnalytics ODBC pour obtenir de la documentation sur les pilotes. GitHub

Rubriques

- Configuration du Timestream pour le conducteur LiveAnalytics ODBC
- Syntaxe et options de la chaîne de connexion pour le ODBC pilote
- Exemples de chaînes de connexion pour le Timestream for driver LiveAnalytics ODBC
- Résolution des problèmes de connexion avec le ODBC pilote

Configuration du Timestream pour le conducteur LiveAnalytics ODBC

Configurez l'accès à Timestream LiveAnalytics dans votre compte AWS

Si vous n'avez pas encore configuré votre AWS compte pour qu'il fonctionne avec Timestream for LiveAnalytics, suivez les instructions figurant dans. Accès à Timestream pour LiveAnalytics

Installez le ODBC pilote sur votre système

Téléchargez le programme d'installation du ODBC pilote Timestream approprié pour votre système depuis le ODBC GitHubréférentiel, et suivez les instructions d'installation qui s'appliquent à votre système :.

- Guide d'installation de Windows
- Guide d'installation de macOS
- Guide d'installation de Linux

Définissez un nom de source de données (DSN) pour le ODBC pilote

Suivez les instructions du guide DSN de configuration de votre système :

- DSNConfiguration de Windows
- DSNConfiguration de macOS
- DSNConfiguration de Linux

Configurez votre application de Business Intelligence (BI) pour qu'elle fonctionne avec le ODBC conducteur

Voici les instructions pour configurer plusieurs applications de BI courantes pour qu'elles fonctionnent avec le ODBC pilote :

- Configuration de Microsoft Power BI
- Configuration de Microsoft Excel
- Configuration de Tableau

Pour d'autres applications

Syntaxe et options de la chaîne de connexion pour le ODBC pilote

La syntaxe permettant de spécifier les options de chaîne de connexion pour le ODBC pilote est la suivante :

```
DRIVER={Amazon Timestream ODBC Driver};(option)=(value);
```

Les options disponibles sont les suivantes :

Options de connexion du pilote

- **`Driver`**(obligatoire) — Le pilote utilisé avecODBC.

  La valeur par défaut est Amazon Timestream.
- **`DSN`**— Le nom de la source de données (DSN) à utiliser pour configurer la connexion.

  L'argument par défaut est `NONE`.
- **`Auth`**— Le mode d'authentification. Il doit s'agir de l'une des options suivantes :
  - `AWS_PROFILE`— Utilisez la chaîne d'informations d'identification par défaut.
  - `IAM`— Utilisez les AWS IAM informations d'identification.
  - `AAD`— Utilisez le fournisseur d'identité Azure Active Directory (AD).
  - `OKTA`— Utilisez le fournisseur d'identité Okta.

  L'argument par défaut est `AWS_PROFILE`.

Options de configuration des terminaux

- **`EndpointOverride`**— La dérogation du point de terminaison pour le service Timestream. LiveAnalytics Il s'agit d'une option avancée qui remplace la région. Par exemple :

  ```
  query-cell2.timestream.us-east-1.amazonaws.com
  ```
- **`Region`**— La région de signature pour le Timestream pour le point de terminaison de LiveAnalytics service.

  L'argument par défaut est `us-east-1`.

Option de fournisseur d'informations d'identification

- **ProfileName**— Le nom du profil dans le fichier de AWS configuration.

  L'argument par défaut est NONE.

AWS IAMoptions d'authentification

- **UID**ou **AccessKeyId**— L'identifiant de la clé d'accès AWS utilisateur. Si UID les deux AccessKeyId sont fournis dans la chaîne de connexion, la UID valeur sera utilisée sauf si elle est vide.

  L'argument par défaut est NONE.
- **PWD**ou **SecretKey**— La clé d'accès secrète de l'AWSutilisateur. Si PWD les deux SecretKey sont fournis dans la chaîne de connexion, la PWD valeur with sera utilisée sauf si elle est vide.

  L'argument par défaut est NONE.
- **SessionToken**— Le jeton de session temporaire requis pour accéder à une base de données avec l'authentification multifactorielle (MFA) activée. N'incluez pas de traînée = dans l'entrée.

  L'argument par défaut est NONE.

SAMLoptions d'authentification basées sur Okta

- **IdPHost**— Le nom d'hôte de l'IdP spécifié.

  L'argument par défaut est NONE.
- **UID**ou **IdPUserName**— Le nom d'utilisateur du compte IdP spécifié. Si UID les deux IdPUserName sont fournis dans la chaîne de connexion, la UID valeur sera utilisée sauf si elle est vide.

  L'argument par défaut est NONE.
- **PWD**ou **IdPPassword**— Le mot de passe du compte IdP spécifié. Si PWD les deux IdPPassword sont fournis dans la chaîne de connexion, la PWD valeur sera utilisée sauf si elle est vide.

  L'argument par défaut est NONE.

- **OktaApplicationID**— L'identifiant unique fourni par Okta associé au Timestream de l'application. LiveAnalytics L'identifiant de l'application (AppId) se trouve dans le `entityID` champ fourni dans les métadonnées de l'application. Voici un exemple :

  ```
  entityID="http://www.okta.com//(IdPAppID)
  ```

  L'argument par défaut est `NONE`.
- **RoleARN**— Le nom de ressource Amazon (ARN) du rôle assumé par l'appelant.

  L'argument par défaut est `NONE`.
- **IdPARN**— Le nom de ressource Amazon (ARN) du SAML fournisseur IAM qui décrit l'IdP.

  L'argument par défaut est `NONE`.

SAMLoptions d'authentification basées sur Azure Active Directory

- **UID**ou **IdPUserName**— Le nom d'utilisateur du compte IdP spécifié.

  L'argument par défaut est `NONE`.
- **PWD**ou **IdPPassword**— Le mot de passe du compte IdP spécifié.

  L'argument par défaut est `NONE`.
- **AADApplicationID**— L'identifiant unique de l'application enregistrée sur Azure AD.

  L'argument par défaut est `NONE`.
- **AADClientSecret**— Le secret client associé à l'application enregistrée sur Azure AD est utilisé pour autoriser l'extraction de jetons.

  L'argument par défaut est `NONE`.
- **AADTenant**— L'identifiant du locataire Azure AD.

  L'argument par défaut est `NONE`.
- **RoleARN**— Le nom de ressource Amazon (ARN) du rôle assumé par l'appelant.

  L'argument par défaut est `NONE`.
- **IdPARN**— Le nom de ressource Amazon (ARN) du SAML fournisseur IAM qui décrit l'IdP.

  L'argument par défaut est `NONE`.

AWS SDKOptions (avancées)

- **`RequestTimeout`**— Durée en millisecondes pendant laquelle le délai d' AWS SDKattente d'une demande de requête est expiré. Toute valeur non positive désactive le délai d'expiration de la demande.

  L'argument par défaut est `3000`.

- **`ConnectionTimeout`**— Durée en millisecondes pendant laquelle les AWS SDK données doivent être transférées via une connexion ouverte avant l'expiration du délai imparti. La valeur 0 désactive le délai d'expiration de la connexion. Cette valeur ne doit pas être négative.

  L'argument par défaut est `1000`.

- **`MaxRetryCountClient`**— Le nombre maximal de tentatives pour les erreurs réessayables contenant 5xx codes d'erreur dans le. SDK La valeur ne doit pas être négative.

  L'argument par défaut est `0`.

- **`MaxConnections`**— Le nombre maximum de HTTP connexions ouvertes simultanément autorisées au service Timestream. La valeur doit être positive.

  L'argument par défaut est `25`.

ODBCOptions de journalisation du pilote

- **`LogLevel`**— Le niveau de journalisation pour la journalisation du pilote. Doit être l'une des valeurs suivantes :
  - (0OFF).
  - (1ERROR).
  - (2WARNING).
  - (3INFO).
  - (4DEBUG).

  La valeur par défaut est 1 (ERROR).

  Attention : des informations personnelles peuvent être enregistrées par le conducteur lors de l'utilisation du mode DEBUG journalisation.

- **`LogOutput`**— Dossier dans lequel le fichier journal doit être stocké.

La valeur par défaut est :

- Windows :`%USERPROFILE%`, ou s'il n'est pas disponible,`%HOMEDRIVE%%HOMEPATH%`.
- macOS et Linux : `$HOME` ou s'il n'est pas disponible, le champ `pw_dir` de la fonction `getpwuid(getuid())` renvoie la valeur.

SDKoptions de journalisation

Le niveau de AWS SDK journalisation est distinct du flux temporel pour le niveau de journalisation LiveAnalytics ODBC du pilote. Le réglage de l'un n'a aucune incidence sur l'autre.

Le niveau de SDK journalisation est défini à l'aide de la variable d'environnement`TS_AWS_LOG_LEVEL`. Les valeurs valides sont :

- `OFF`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`
- `FATAL`

Si `TS_AWS_LOG_LEVEL` ce n'est pas le cas, le niveau de SDK journalisation est défini sur la valeur par défaut, qui est`WARN`.

Connexion via un proxy

Le ODBC pilote prend en charge la connexion à Amazon Timestream via LiveAnalytics un proxy. Pour utiliser cette fonctionnalité, configurez les variables d'environnement suivantes en fonction des paramètres de votre proxy :

- **`TS_PROXY_HOST`**— l'hôte proxy.
- **`TS_PROXY_PORT`**— Le numéro de port du proxy.
- **`TS_PROXY_SCHEME`**— Le schéma de proxy, `http` soit`https`.
- **`TS_PROXY_USER`**— Le nom d'utilisateur pour l'authentification par proxy.
- **`TS_PROXY_PASSWORD`**— Le mot de passe utilisateur pour l'authentification par proxy.

- **TS_PROXY_SSL_CERT_PATH**— Le fichier de SSL certificat à utiliser pour se connecter à un HTTPS proxy.
- **TS_PROXY_SSL_CERT_TYPE**— Type du SSL certificat du client proxy.
- **TS_PROXY_SSL_KEY_PATH**— Le fichier de clé privée à utiliser pour se connecter à un HTTPS proxy.
- **TS_PROXY_SSL_KEY_TYPE**— Type de fichier de clé privée utilisé pour se connecter à un HTTPS proxy.
- **TS_PROXY_SSL_KEY_PASSWORD**— Phrase secrète du fichier de clé privée utilisé pour se connecter à un proxy. HTTPS

Exemples de chaînes de connexion pour le Timestream for driver LiveAnalytics ODBC

Exemple de connexion au ODBC pilote avec des IAM informations d'identification

```
Driver={Amazon Timestream ODBC Driver};Auth=IAM;AccessKeyId=(your access key
 ID);secretKey=(your secret key);SessionToken=(your session token);Region=us-east-2;
```

Exemple de connexion au ODBC pilote avec un profil

```
Driver={Amazon Timestream ODBC Driver};ProfileName=(the profile name);region=us-west-2;
```

Le pilote tentera de se connecter en utilisant les informations d'identification fournies dans`~/.aws/credentials`, ou si un fichier est spécifié dans la variable d'environnement`AWS_SHARED_CREDENTIALS_FILE`, en utilisant les informations d'identification contenues dans ce fichier.

Exemple de connexion au ODBC pilote avec 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);
```

Exemple de connexion au ODBC pilote avec 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);
```

Exemple de connexion au ODBC pilote avec un point de terminaison spécifié et un niveau de journalisation 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;
```

Résolution des problèmes de connexion avec le ODBC pilote

> **Note**
>
> Lorsque le nom d'utilisateur et le mot de passe sont déjà spécifiés dans leDSN, il n'est pas nécessaire de les spécifier à nouveau lorsque le gestionnaire de ODBC pilotes les demande.

Un code d'erreur, `01S02` avec un message, `Re-writing (connection string option) (have you specified it several times?` se produit lorsqu'une option de chaîne de connexion est passée plusieurs fois dans la chaîne de connexion. Le fait de spécifier une option plusieurs fois génère une erreur. Lorsque vous établissez une connexion avec a DSN et une chaîne de connexion, si une option de connexion est déjà spécifiée dans leDSN, ne la spécifiez pas à nouveau dans la chaîne de connexion.

VPCpoints de terminaison ()AWS PrivateLink

Vous pouvez établir une connexion privée entre votre compte VPC et Amazon Timestream en créant un point de LiveAnalytics terminaison d'interface. VPC Pour de plus amples informations, veuillez consulter VPCpoints de terminaison ()AWS PrivateLink.

Bonnes pratiques

Pour profiter pleinement des avantages d'Amazon Timestream LiveAnalytics pour, suivez les meilleures pratiques décrites ci-dessous.

> **Note**
>
> Lorsque vous exécutez proof-of-concept des applications, tenez compte de la quantité de données que votre application accumulera en quelques mois ou années tout en évaluant les performances et l'échelle de Timestream. LiveAnalytics À mesure que vos données augmentent au fil du temps, vous remarquerez que les performances de Timestream pour

la plupart LiveAnalytics restent inchangées, car son architecture sans serveur peut tirer parti d'un énorme parallélisme pour traiter des volumes de données plus importants et s'adapter automatiquement aux besoins de votre application.

Rubriques

Modélisation des données

Amazon Timestream LiveAnalytics for est conçu pour collecter, stocker et analyser des séries chronologiques provenant d'applications et d'appareils émettant une séquence de données horodatée. Pour des performances optimales, les données envoyées à Timestream LiveAnalytics doivent avoir des caractéristiques temporelles et le temps doit être une composante essentielle des données.

Timestream for vous LiveAnalytics offre la flexibilité de modéliser vos données de différentes manières en fonction des exigences de votre application. Dans cette section, nous abordons plusieurs de ces modèles et vous donnons des directives pour optimiser vos coûts et vos performances. Familiarisez-vous avec Timestream pour des LiveAnalytics concepts tels que les dimensions et les mesures. Dans cette section, vous en apprendrez davantage sur :

Lorsque vous décidez de créer une seule table ou plusieurs tables pour stocker les données, tenez compte des points suivants :

- Quelles données placer dans la même table ou lorsque vous souhaitez séparer les données entre plusieurs tables et bases de données.
- Comment choisir entre Timestream pour les enregistrements à LiveAnalytics mesures multiples ou les enregistrements à mesure unique, et les avantages de la modélisation à l'aide

d'enregistrements à mesures multiples, en particulier lorsque votre application suit plusieurs mesures en même temps instantanément.

- Quels attributs modéliser sous forme de dimensions ou de mesures.
- Comment utiliser efficacement les attributs des noms de mesures pour optimiser la latence de vos requêtes.

Rubriques

Table unique ou tables multiples

Lorsque vous modélisez vos données dans une application, un autre aspect important est de savoir comment les modéliser dans des tables et des bases de données. Les bases de données et les tables de Timestream for LiveAnalytics sont des abstractions destinées au contrôle d'accès, à la spécification des KMS clés, aux périodes de conservation, etc. Timestream permet de partitionner LiveAnalytics automatiquement vos données et est conçu pour adapter les ressources en fonction de l'ingestion, du stockage, de la charge des requêtes et des exigences de vos applications.

Dans Timestream for, un tableau LiveAnalytics peut prendre en charge des pétaoctets de données stockées, des dizaines de gigaoctets par seconde d'écriture de données, et les requêtes peuvent en traiter des centaines par heure. TBs Les requêtes de Timestream for LiveAnalytics peuvent couvrir plusieurs tables et bases de données, fournissant des jointures et des unions pour fournir un accès fluide à vos données sur plusieurs tables et bases de données. L'échelle des données ou le volume de demandes ne sont donc généralement pas la principale préoccupation lorsque vous décidez comment organiser vos données dans Timestream for. LiveAnalytics Vous trouverez ci-dessous quelques points importants à prendre en compte lors du choix des données à colocaliser dans la même table par rapport à des tables différentes ou à des tables de différentes bases de données.

- Les politiques de conservation des données (conservation de la mémoire, conservation de la mémoire magnétique, etc.) sont prises en charge au niveau de la granularité d'une table. Par conséquent, les données qui nécessitent des politiques de conservation différentes doivent figurer dans des tables différentes.

- AWS KMS les clés utilisées pour chiffrer vos données sont configurées au niveau de la base de données. Par conséquent, les différentes exigences en matière de clés de chiffrement impliquent que les données devront se trouver dans différentes bases de données.
- Timestream for LiveAnalytics prend en charge le contrôle d'accès basé sur les ressources au niveau de la granularité des tables et des bases de données. Tenez compte de vos exigences en matière de contrôle d'accès lorsque vous décidez quelles données vous souhaitez écrire dans la même table ou dans des tables différentes.
- Tenez compte des limites relatives au nombre de dimensions, de noms de mesures et de noms d'attributs à mesures multiples lorsque vous décidez quelles données sont stockées dans quelle table.
- Tenez compte de la charge de travail de vos requêtes et de vos modèles d'accès lorsque vous décidez de la manière dont vous organisez vos données, car la latence des requêtes et la facilité de rédaction de vos requêtes en dépendront.
  - Si vous stockez les données que vous demandez fréquemment dans la même table, cela facilitera généralement la rédaction de vos requêtes, ce qui vous évitera d'avoir à écrire des jointures, des unions ou des expressions de table communes. Cela se traduit également généralement par une réduction de la latence des requêtes. Vous pouvez utiliser des prédicats sur les dimensions et les noms de mesures pour filtrer les données pertinentes pour les requêtes.

    Par exemple, imaginez le cas où vous stockez des données provenant d'appareils situés sur six continents. Si vos requêtes accèdent fréquemment à des données provenant de plusieurs continents pour obtenir une vue globale agrégée, le stockage des données de ces continents dans la même table facilitera la rédaction des requêtes. D'autre part, si vous stockez des données sur différentes tables, vous pouvez toujours combiner les données dans la même requête, mais vous devrez écrire une requête pour unir les données des différentes tables.
  - Timestream for LiveAnalytics utilise le partitionnement et l'indexation adaptatifs de vos données. Les requêtes ne sont donc facturées que pour les données pertinentes à vos requêtes. Par exemple, si vous avez une table stockant les données d'un million d'appareils sur six continents, si votre requête contient des prédicats du type « `WHERE device_id = 'abcdef'` ou »`WHERE continent = 'North America'`, les requêtes ne sont facturées que pour les données relatives à l'appareil ou au continent.
  - Dans la mesure du possible, si vous utilisez le nom de la mesure pour séparer les données d'une même table qui ne sont pas émises au même moment ou qui ne sont pas fréquemment demandées, `WHERE measure_name = 'cpu'` en utilisant des prédicats tels que ceux de votre requête, vous bénéficiez non seulement des avantages de la mesure, mais Timestream for

LiveAnalytics peut également éliminer efficacement les partitions dont le nom de mesure n'est pas utilisé dans votre prédicat de requête. Cela vous permet de stocker des données associées avec différents noms de mesures dans la même table sans impact sur la latence ou les coûts des requêtes, et évite de répartir les données dans plusieurs tables. Le nom de la mesure est essentiellement utilisé pour partitionner les données et supprimer les partitions non pertinentes pour la requête.

Enregistrements à mesures multiples contre enregistrements à mesure unique

Timestream for vous LiveAnalytics permet d'écrire des données avec plusieurs mesures par enregistrement (mesures multiples) ou une seule mesure par enregistrement (mesure unique).

Enregistrements à mesures multiples

Dans de nombreux cas d'utilisation, un appareil ou une application que vous suivez peut émettre plusieurs mesures ou événements au même horodatage. Dans de tels cas, vous pouvez stocker toutes les métriques émises au même horodatage dans le même enregistrement multi-mesures. C'est-à-dire que toutes les mesures stockées dans le même enregistrement multi-mesures apparaissent sous forme de colonnes différentes dans la même ligne de données.

Supposons, par exemple, que votre application émet des métriques telles que cpu, memory, disk_iops à partir d'un périphérique mesuré au même moment. Voici un exemple d'une telle table où plusieurs métriques émises simultanément sont stockées dans la même ligne. Vous verrez alors deux hôtes émettre les métriques une fois par seconde.

Hostname	nom_mesure	Heure	cpu	Mémoire	disk_iops
Hôte-24GJU	metrics	2021-12-01 19:00:00	35	54,9	38,2
Hôte-24GJU	metrics	2021-12-01 19:00:01	36	58	39
Hôte - 28 GJU	metrics	2021-12-01 19:00:00	15	55	92
Hôte - 28 GJU	metrics	2021-12-01 19:00:01	16	50	40

Enregistrements à mesure unique

Les enregistrements de mesures uniques conviennent lorsque vos appareils émettent des mesures différentes à différentes périodes, ou lorsque vous utilisez une logique de traitement personnalisée qui émet metrics/events at different time periods (for instance, when a device's reading/state des modifications). Comme chaque mesure possède un horodatage unique, les mesures peuvent être stockées dans leurs propres enregistrements dans Timestream for. LiveAnalytics Prenons l'exemple d'un capteur IoT, qui suit la température et l'humidité du sol, qui émet un enregistrement uniquement lorsqu'il détecte un changement par rapport à l'entrée signalée précédemment. L'exemple suivant fournit un exemple de telles données émises à l'aide d'enregistrements de mesures uniques.

device_id	nom_mesure	Heure	valeur_mesure : double	value_mesure : :bigint
sensor-sea478	temperature	2021-12-01 19:22:32	35	NULL
sensor-sea478	temperature	2021-12-01 18:07:51	36	NULL
sensor-sea478	humidité	2021-12-01 19:05:30	NULL	21
sensor-sea478	humidité	2021-12-01 19:00:01	NULL	23

Comparaison des enregistrements à mesure unique et à mesures multiples

Timestream for vous LiveAnalytics offre la flexibilité de modéliser vos données sous forme d'enregistrements à mesure unique ou à mesures multiples en fonction des exigences et des caractéristiques de votre application. Une seule table peut stocker à la fois des enregistrements de mesures uniques et de plusieurs mesures, si les exigences de votre application le souhaitent. En général, lorsque votre application émet plusieurs mesures/événements simultanément, il est généralement recommandé de modéliser les données sous forme d'enregistrements multi-mesures pour un accès performant aux données et un stockage rentable des données.

Par exemple, si vous considérez un cas d'DevOps utilisation pour suivre les métriques et les événements provenant de centaines de milliers de serveurs, chaque serveur émet périodiquement

20 métriques et 5 événements, les événements et les métriques étant émis en même temps instantanément. Ces données peuvent être modélisées à l'aide d'enregistrements à mesures uniques ou à l'aide d'enregistrements à mesures multiples (voir le générateur de données open source pour le schéma obtenu). Dans ce cas d'utilisation, la modélisation des données à l'aide d'enregistrements à mesures multiples par rapport à des enregistrements à mesures uniques donne les résultats suivants :

- Mesure de l'ingestion - Les enregistrements à mesures multiples permettent de réduire d'environ 40 % le nombre d'octets d'ingestion écrits.
- Traitement par lots d'ingestion - Les enregistrements à mesures multiples entraînent l'envoi de lots de données plus importants, ce qui signifie que les clients ont besoin de moins de threads et de moins de threads CPU pour traiter l'ingestion.
- Mesure du stockage - Les enregistrements à mesures multiples réduisent d'environ 8 fois la capacité de stockage, ce qui se traduit par des économies de stockage importantes à la fois en termes de mémoire et de stockage magnétique.
- Latence des requêtes : les enregistrements à mesures multiples réduisent la latence des requêtes pour la plupart des types de requêtes par rapport aux enregistrements à mesure unique.
- Nombre d'octets mesurés par requête : pour les requêtes analysant moins de 10 Mo de données, les enregistrements à mesure unique et à mesures multiples sont comparables. Pour les requêtes accédant à une seule mesure et scannant des données de plus de 10 Mo, les enregistrements d'une seule mesure se traduisent généralement par une réduction du nombre d'octets mesurés. Pour les requêtes faisant référence à 3 mesures ou plus, les enregistrements à mesures multiples se traduisent par une réduction du nombre d'octets mesurés.
- Facilité d'expression de requêtes à mesures multiples : lorsque vos requêtes font référence à plusieurs mesures, la modélisation de vos données à l'aide d'enregistrements à mesures multiples permet d'écrire des requêtes plus simples et plus compactes.

Les facteurs précédents varieront en fonction du nombre de mesures que vous suivez, du nombre de dimensions de vos données, etc. Bien que l'exemple précédent fournisse des données concrètes, nous constatons que dans de nombreux scénarios d'application et cas d'utilisation, si votre application émet plusieurs mesures au même instant, le stockage des données sous forme d'enregistrements de mesures multiples est plus efficace. De plus, les enregistrements à mesures multiples vous offrent la flexibilité des types de données et le stockage de plusieurs autres valeurs en tant que contexte (par exemple, le stockage de la demande IDs et des horodatages supplémentaires, dont il sera question plus loin).

Notez qu'un enregistrement multi-mesures peut également modéliser des mesures éparses, comme dans l'exemple précédent pour les enregistrements de mesures uniques : vous pouvez utiliser le nom de la mesure pour stocker le nom de la mesure et utiliser un nom d'attribut multi-mesures générique, tel que value_double pour stocker les mesures, value_bigint pour stocker les `DOUBLE` mesures, value_timestamp pour stocker des valeurs supplémentaires, etc. `BIGINT` `TIMESTAMP`

Dimensions et mesures

Un tableau dans Timestream vous LiveAnalytics permet de stocker des dimensions (identifiant les attributs de l'appareil/des données que vous stockez) et des mesures (les mesures/valeurs que vous suivez). Consultez Timestream pour les concepts pour plus de détails. LiveAnalytics Lorsque vous modélisez votre application sur Timestream pour LiveAnalytics, la façon dont vous mappez vos données en dimensions et en mesures a un impact sur votre ingestion et la latence des requêtes. Vous trouverez ci-dessous des directives sur la manière de modéliser vos données sous forme de dimensions et de mesures que vous pouvez appliquer à votre cas d'utilisation.

Choix des dimensions

Les données qui identifient la source qui envoie les données de série chronologique s'adaptent naturellement aux dimensions, qui sont des attributs qui ne changent pas au fil du temps. Par exemple, si un serveur émet des métriques, les attributs identifiant le serveur, tels que le nom d'hôte, la région, le rack, la zone de disponibilité, sont candidats aux dimensions. De même, pour un appareil IoT doté de plusieurs capteurs signalant des données chronologiques, l'identifiant de l'appareil, l'identifiant du capteur, etc. sont candidats pour les dimensions.

Si vous écrivez des données sous forme d'enregistrements à mesures multiples, les dimensions et les attributs à mesures multiples apparaissent sous forme de colonnes dans le tableau lorsque vous effectuez DESCRIBE ou exécutez une SELECT instruction sur le tableau. Ainsi, lorsque vous rédigez vos requêtes, vous pouvez librement utiliser les dimensions et les mesures de la même requête. Toutefois, lorsque vous créez votre enregistrement d'écriture pour ingérer des données, gardez à l'esprit les points suivants lorsque vous choisissez les attributs spécifiés sous forme de dimensions et ceux qui sont des valeurs de mesure :

- Les noms des dimensions, les valeurs, le nom des mesures et l'horodatage identifient de manière unique les données des séries chronologiques. Timestream for LiveAnalytics utilise cet identifiant unique pour dédupliquer automatiquement les données. En d'autres termes, si Timestream pour LiveAnalytics reçoit deux points de données portant les mêmes valeurs de nom de dimension, de valeurs de dimension, de nom de mesure et d'horodatage, si les valeurs ont le même numéro de version, alors Timestream pour les dédoublons. LiveAnalytics Si la nouvelle demande d'écriture

possède une version inférieure aux données déjà existantes dans Timestream pour LiveAnalytics, la demande d'écriture est rejetée. Si la nouvelle demande d'écriture possède une version supérieure, la nouvelle valeur remplace l'ancienne valeur. Par conséquent, la façon dont vous choisissez vos valeurs de dimension aura un impact sur ce comportement de déduplication.

- Les noms et valeurs des dimensions ne peuvent pas être mis à jour, la valeur de mesure peut l'être. Ainsi, toutes les données susceptibles de nécessiter des mises à jour sont mieux modélisées sous forme de valeurs de mesure. Par exemple, si vous avez une machine dans l'usine dont la couleur peut changer, vous pouvez modéliser la couleur sous forme de valeur de mesure, sauf si vous souhaitez également utiliser la couleur comme attribut d'identification nécessaire à la déduplication. En d'autres termes, les valeurs de mesure peuvent être utilisées pour stocker des attributs qui ne changent que lentement au fil du temps.

Notez qu'un tableau dans Timestream pour LiveAnalytics ne limite pas le nombre de combinaisons uniques de noms et de valeurs de dimension. Par exemple, des milliards de combinaisons de valeurs uniques de ce type peuvent être stockées dans une table. Cependant, comme vous le verrez dans les exemples suivants, un choix judicieux des dimensions et des mesures peut optimiser considérablement la latence de vos demandes, en particulier pour les requêtes.

Dimensions uniques IDs

Si le scénario de votre application nécessite que vous stockiez un identifiant unique pour chaque point de données (par exemple, un identifiant de demande, un identifiant de transaction ou un identifiant de corrélation), la modélisation de l'attribut ID en tant que valeur de mesure améliorera considérablement la latence des requêtes. Lorsque vous modélisez vos données à l'aide d'enregistrements à mesures multiples, l'ID apparaît sur la même ligne en contexte avec vos autres dimensions et données de séries chronologiques, afin que vos requêtes puissent continuer à les utiliser efficacement. Par exemple, si l'on considère un cas d'DevOps utilisation où chaque point de données émis par un serveur possède un attribut d'ID de demande unique, la modélisation de l'ID de demande sous forme de valeur de mesure permet de réduire la latence des requêtes jusqu'à 4 fois plus faible pour différents types de requêtes, par opposition à la modélisation de l'ID de demande unique en tant que dimension.

Vous pouvez utiliser la même analogie pour des attributs qui ne sont pas totalement uniques pour chaque point de données, mais qui comportent des centaines de milliers ou des millions de valeurs uniques. Vous pouvez modéliser ces attributs sous forme de dimensions ou de valeurs de mesure. Vous souhaiterez le modéliser sous forme de dimension si les valeurs sont nécessaires à la déduplication sur le chemin d'écriture, comme indiqué précédemment, ou si vous l'utilisez souvent

comme prédicat (par exemple, dans la `WHERE` clause avec un prédicat d'égalité sur une valeur de cet attribut, telle que l'`device_id = ‘abcde’`endroit où votre application suit des millions d'appareils) dans vos requêtes.

Richesse des types de données avec des enregistrements à mesures multiples

Les enregistrements à mesures multiples vous offrent la flexibilité nécessaire pour modéliser efficacement vos données. Les données que vous stockez dans un enregistrement multi-mesures apparaissent sous forme de colonnes dans le tableau, à l'instar des dimensions, ce qui permet de demander des dimensions et des valeurs de mesure avec la même facilité. Vous avez observé certains de ces modèles dans les exemples présentés précédemment. Vous trouverez ci-dessous des modèles supplémentaires permettant d'utiliser efficacement les enregistrements à mesures multiples afin de répondre aux cas d'utilisation de votre application.

Les enregistrements à mesures multiples prennent en charge les attributs des types de données `DOUBLE`,`BIGINT`,`VARCHAR`,`BOOLEAN`, et`TIMESTAMP`. Par conséquent, ils correspondent naturellement à différents types d'attributs :

- Informations de localisation : par exemple, si vous souhaitez suivre l'emplacement (exprimé sous forme de latitude et de longitude), le modéliser sous la forme d'un attribut à mesures multiples réduira le temps de latence des requêtes par rapport au stockage sous forme de `VARCHAR` dimensions, en particulier lorsque vous avez des prédicats sur la latitude et les longitudes.
- Plusieurs horodatages dans un enregistrement : si le scénario de votre application vous oblige à suivre plusieurs horodatages pour un enregistrement de série chronologique, vous pouvez les modéliser sous forme d'attributs supplémentaires dans l'enregistrement multi-mesures. Ce modèle peut être utilisé pour stocker des données avec des horodatages futurs ou passés. Notez que chaque enregistrement utilisera toujours l'horodatage dans la colonne de temps pour partitionner, indexer et identifier un enregistrement de manière unique.

En particulier, si vous avez des données numériques ou des horodatages sur lesquels vous avez des prédicats dans la requête, la modélisation de ces attributs sous forme d'attributs à mesures multiples, par opposition à des dimensions, réduira le temps de latence des requêtes. En effet, lorsque vous modélisez de telles données à l'aide des types de données riches pris en charge dans les enregistrements à mesures multiples, vous pouvez exprimer les prédicats à l'aide de types de données natifs au lieu de convertir des valeurs en un autre type de VARCHAR données si vous avez modélisé ces données sous forme de dimensions.

Utilisation du nom de la mesure avec des enregistrements de plusieurs mesures

Les tables de Timestream prennent en LiveAnalytics charge un attribut spécial (ou une colonne) appelé nom de mesure. Vous spécifiez une valeur pour cet attribut pour chaque enregistrement pour lequel vous écrivez dans Timestream. LiveAnalytics Pour les enregistrements d'une seule mesure, il est naturel d'utiliser le nom de votre métrique (tel que cpu, mémoire pour les métriques du serveur, ou température, pression pour les métriques du capteur). Lorsque vous utilisez des enregistrements multi-mesures, étant donné que les attributs d'un enregistrement multi-mesures sont nommés (et ces noms deviennent des noms de colonnes dans la table), cpu, mémoire ou température, la pression peut devenir des noms d'attributs multi-mesures. Il est donc naturel de se demander comment utiliser efficacement le nom de la mesure.

Timestream for LiveAnalytics utilise les valeurs de l'attribut du nom de la mesure pour partitionner et indexer les données. Par conséquent, si une table possède plusieurs noms de mesures différents et si les requêtes utilisent ces valeurs comme prédicats de requête, Timestream for LiveAnalytics peut utiliser son partitionnement et son indexation personnalisés pour éliminer les données qui ne sont pas pertinentes pour les requêtes. Par exemple, si votre table contient des noms de mesures de processeur et de mémoire, et que votre requête contient un prédicat`WHERE measure_name = 'cpu'`, Timestream for LiveAnalytics peut efficacement affiner les données pour les noms de mesures qui ne sont pas pertinents pour la requête, par exemple les lignes contenant une mémoire de nom de mesure dans cet exemple. Cet élagage s'applique même lorsque vous utilisez des noms de mesures avec des enregistrements de plusieurs mesures. Vous pouvez utiliser efficacement l'attribut de nom de mesure comme attribut de partitionnement pour une table. Le nom de la mesure ainsi que les noms et valeurs des dimensions, ainsi que le temps, sont utilisés pour partitionner les données dans un flux temporel pour LiveAnalytics une table. Tenez compte des limites du nombre de noms de mesures uniques autorisés dans un flux temporel pour LiveAnalytics une table. Notez également qu'un nom de mesure est également associé à un type de données de valeur de mesure. Par exemple, un seul nom de mesure ne peut être associé qu'à un seul type de valeur de mesure. Ce type peut être l'un des `DOUBLE` suivants : `BIGINT``BOOLEAN`,`VARCHAR`, et`MULTI`. Les enregistrements de plusieurs mesures stockés avec un nom de mesure auront le type de données comme `MULTI` suit. Étant donné qu'un seul enregistrement multi-mesures peut stocker plusieurs métriques avec différents types de données (`DOUBLE`,`BIGINT`, `VARCHAR``BOOLEAN`, et`TIMESTAMP`), vous pouvez associer des données de différents types dans un enregistrement multi-mesures.

Les sections suivantes décrivent quelques exemples de la manière dont l'attribut de nom de mesure peut être utilisé efficacement pour regrouper différents types de données dans la même table.

Les capteurs IoT signalent la qualité et la valeur

Supposons que vous disposiez d'une application surveillant les données provenant de capteurs IoT. Chaque capteur suit différentes mesures, telles que la température et la pression. Outre les valeurs réelles, les capteurs signalent également la qualité des mesures, qui est une mesure de la précision de la lecture et une unité de mesure. Comme la qualité, l'unité et la valeur sont émises ensemble, elles peuvent être modélisées sous forme d'enregistrements à mesures multiples, comme le montre l'exemple de données ci-dessous où device_id est une dimension, la qualité, la valeur et l'unité sont des attributs à mesures multiples :

device_id	nom_mesure	Heure	Qualité	Valeur	Unité
sensor-sea478	temperature	2021-12-01 19:22:32	92	35	c
sensor-sea478	temperature	2021-12-01 18:07:51	93	34	c
sensor-sea478	pressure (pression)	2021-12-01 19:05:30	98	31	psi
sensor-sea478	pressure (pression)	2021-12-01 19:00:01	24	132	psi

Cette approche vous permet de combiner les avantages des enregistrements multi-mesures avec le partitionnement et l'élagage des données en utilisant les valeurs du nom de la mesure. Si les requêtes font référence à une seule mesure, par exemple la température, vous pouvez inclure un prédicat de nom de mesure dans la requête. Voici un exemple d'une telle requête, qui projette également l'unité pour les mesures dont la qualité est supérieure à 90.

```
SELECT device_id, time, value AS temperature, unit
FROM db.table
WHERE time > ago(1h)
    AND measure_name = 'temperature'
    AND quality > 90
```

L'utilisation du prédicat measure_name sur la requête permet LiveAnalytics à Timestream de supprimer efficacement les partitions et les données qui ne sont pas pertinentes pour la requête, améliorant ainsi la latence de votre requête.

Il est également possible de stocker toutes les métriques dans le même enregistrement multi-mesures si toutes les métriques sont émises au même horodatage et/ou si plusieurs métriques sont interrogées ensemble dans la même requête. Par exemple, vous pouvez créer un enregistrement multi-mesures avec les attributs temperature_quality, temperature_value, temperature_unit, pressure_quality, pressure_value, pressure_unit, etc. La plupart des points abordés précédemment concernant la modélisation des données à l'aide d'enregistrements à mesures uniques ou à mesures multiples s'appliquent à votre décision quant à la manière de modéliser les données. Tenez compte de vos modèles d'accès aux requêtes et de la manière dont vos données sont générées pour choisir un modèle qui optimise les coûts, l'ingestion et la latence des requêtes, ainsi que la facilité de rédaction de vos requêtes.

Différents types de métriques dans le même tableau

Un autre cas d'utilisation dans lequel vous pouvez combiner des enregistrements de plusieurs mesures avec des valeurs de nom de mesure consiste à modéliser différents types de données émis indépendamment par le même appareil. Prenons le cas d'utilisation de la DevOps surveillance : les serveurs émettent deux types de données : des métriques émises régulièrement et des événements irréguliers. Le schéma décrit dans le générateur de données modélisant un cas d' DevOps utilisation est un exemple de cette approche. Dans ce cas, vous pouvez stocker les différents types de données émises par le même serveur dans la même table en utilisant différents noms de mesures. Par exemple, toutes les métriques émises en même temps sont stockées avec les métriques de nom de mesure. Tous les événements émis à un instant différent de celui des métriques sont stockés avec les événements de nom de mesure. Le schéma de mesure de la table (par exemple, le résultat de la `SHOW MEASURES` requête) est le suivant :

nom_mesure	data_type	Dimensions
événements	multi	[{"data_type » « varchar », « dimension_name » « availability_zone »}, {"data_type » « varchar », « dimension_name » « microservice_name »}, {"data_type » varchar », « dimension_name » 1_instance_name «}, {" data_type » « varchar », « dimension_name » _name

nom_mesure	data_type	Dimensions
		"}, {" data_type »¨varchar », « dimension_name »¨jdk_version "}, {" data_type »¨varchar », « dimension_name » ¨varchar », « dimension _name »¨varchar », « dimension_name »¨region «}, {" data_type »¨varchar », « dimension_name »¨silo}]
metrics	multi	[{"data_type » « varchar », « dimension_name » « availability_zone »}, {"data_type » « varchar », « dimension_name » « microservice_name »}, {"data_type » varchar », « dimension_name » 1_instance_name «}, {" data_type » .varchar », « dimension_name » _version "}, {" data_type »¨varchar », « dimension_name »¨cell "}, {" data_type »¨varchar », « dimension_name »¨region «}, {" data_type »¨varchar », « dimension_name »¨silo «}, {" data_type »¨varchar », « dimension_name »¨instance _type}]

Dans ce cas, vous pouvez constater que les événements et les métriques ont également des ensembles de dimensions différents, les événements ayant des dimensions différentes jdk_version et process_name tandis que les métriques ont les dimensions instance_type et os_version.

L'utilisation de différents noms de mesures vous permet d'écrire des requêtes avec des prédicats, par exemple `WHERE measure_name = 'metrics'` pour obtenir uniquement les métriques. De plus, le fait d'avoir toutes les données émises par la même instance dans la même table implique que vous pouvez également écrire une requête plus simple avec le prédicat instance_name pour obtenir toutes les données de cette instance. Par exemple, un prédicat du formulaire `WHERE instance_name = ‘instance-1234’` sans prédicat measure_name renverra toutes les données d'une instance de serveur spécifique.

Recommandations pour le partitionnement des enregistrements à mesures multiples

> **Important**
>
> Cette section est obsolète !
> Ces recommandations sont périmées. Le partitionnement est désormais mieux contrôlé à l'aide de clés de partition définies par le client.

Nous avons constaté que l'écosystème des séries chronologiques comporte un nombre croissant de charges de travail qui nécessitent d'ingérer et de stocker d'énormes quantités de données, tout en nécessitant des réponses à des requêtes à faible latence lors de l'accès aux données via un ensemble de valeurs de dimension à cardinalité élevée.

En raison de ces caractéristiques, les recommandations de cette section seront utiles pour les charges de travail des clients présentant les caractéristiques suivantes.

- Vous avez adopté ou souhaitez adopter des enregistrements à mesures multiples.
- Attendez-vous à recevoir dans le système un volume élevé de données qui seront stockées pendant de longues périodes.
- Exiger des temps de réponse à faible latence pour leurs principaux modèles d'accès (requêtes).
- Sachez que les modèles de requêtes les plus importants impliquent une condition de filtrage quelconque dans le prédicat. Cette condition de filtrage est basée sur une dimension de cardinalité élevée. Par exemple, considérez les événements ou les agrégations par UserId DeviceId, ServerID, host-name, etc.

Dans ces cas, un nom unique pour toutes les mesures multi-mesures ne sera pas utile, car notre moteur utilise un nom multi-mesures pour partitionner les données et le fait d'avoir une valeur unique limite l'avantage de partition que vous obtenez. Le partitionnement de ces enregistrements

repose principalement sur deux dimensions. Supposons que le temps soit indiqué sur l'axe X, qu'il y ait un hachage des noms de dimension et que le temps soit indiqué `measure_name` sur l'axe Y. `measure_name`Dans ces cas, cela fonctionne presque comme une clé de partitionnement.

Notre recommandation est la suivante.

- Lorsque vous modélisez vos données pour des cas d'utilisation tels que celui `measure_name` que nous avons mentionné, utilisez un dérivé direct de votre modèle d'accès aux requêtes principal. Par exemple :
  - Votre cas d'utilisation nécessite de suivre les performances des applications et la QoE du point de vue de l'utilisateur final. Il peut également s'agir de mesures de suivi pour un seul serveur ou appareil IoT.
  - Si vous interrogez et filtrez par UserId, vous devez, au moment de l'ingestion, trouver le meilleur moyen de vous associer `measure_name` à UserId.
  - Étant donné qu'une table à plusieurs mesures ne peut contenir que 8 192 noms de mesures différents, la formule adoptée ne doit pas générer plus de 8 192 valeurs différentes.
- Une approche que nous avons appliquée avec succès pour les valeurs de chaîne consiste à appliquer un algorithme de hachage à la valeur de chaîne. Effectuez ensuite l'opération modulo avec la valeur absolue du résultat du hachage et 8192.

```
measure_name = getMeasureName(UserId)
int getMeasureName(value) {
    hash_value =  abs(hash(value))
    return hash_value % 8192
}
```

- Nous avons également ajouté `abs()` la suppression du signe, éliminant ainsi la possibilité que les valeurs soient comprises entre -8192 et 8192. Cela doit être effectué avant l'opération du modulo.
- En utilisant cette méthode, vos requêtes peuvent être exécutées en une fraction du temps qu'il faudrait pour exécuter un modèle de données non partitionné.
- Lorsque vous interrogez les données, assurez-vous d'inclure une condition de filtrage dans le prédicat qui utilise la nouvelle valeur dérivée du measure_name. Par exemple :
  -

```
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))
```

- Cela réduira le nombre total de partitions numérisées pour obtenir des données qui se traduiront par des requêtes plus rapides au fil du temps.

N'oubliez pas que si vous souhaitez bénéficier des avantages de ce schéma de partition, le hachage doit être calculé côté client et transmis à Timestream LiveAnalytics sous forme de valeur statique au moteur de requête. L'exemple précédent fournit un moyen de valider que le hachage généré peut être résolu par le moteur en cas de besoin.

time	host_name	location	type_serveur	cpu_usage	mémoire_disponible	temp du processeur
05.09-07 21:48:44.000000000	hôte-1235	us-east1	5,8 xl	55	16,2	78
RK09-07 21:48:44.000000000	hôte-3587	us-west1	5,8 xl	62	18.1	81
2_09-07 21:48:45 000 000000	hôte-258743	central de l'UE	5,8 xl	88	9,4	91
05.09-07 21:48:45.000000000	hôte-35654	us-east2	5,8 xl	29	24	54
RK09.07 21:48:45,000000000	hôte-254	us-west1	5,8 xl	44	32	48

Pour générer le résultat associé `measure_name` conformément à nos recommandations, il existe deux voies qui dépendent de votre schéma d'ingestion.

1. Pour l'ingestion par lots de données historiques : vous pouvez ajouter la transformation à votre code d'écriture si vous utilisez votre propre code pour le traitement par lots.

   En s'appuyant sur l'exemple précédent.

```
        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);

        }
```

   Sortie

```
host-1235 - 6445
host-3587 - 6399
host-258743 - 640
host-35654 - 2093
host-254 - 7051
```

   Ensemble de données résultant

time	host_name	location	nom_mesure	type_serveur	cpu_usage	mémoire_disponible	temp du processeur
05.09-07 21:48:44.000000000	hôte-1235	us-east1	6445	5,8 xl	55	16,2	78

time	host_name	location	nom_mesure	type_serveur	cpu_usage	mémoire_disponible	temp du processeur
RK09-07 21:48:44.000000000	hôte-3587	us-west1	6399	5,8 xl	62	18.1	81
2_09-07 21:48:45 000 000000	hôte-258743	central de l'UE	640	5,8 xl	88	9,4	91
05.09-07 21:48:45.000000000	hôte-35654	us-east2	2093	5,8 xl	29	24	54
RK09.07 21:48:45,000000000	hôte-254	us-west1	7051	5,8 xl	44	32	48

2. Pour une ingestion en temps réel : vous devez générer le fichier `measure_name` en vol au fur et à mesure que les données arrivent.

Dans les deux cas, nous vous recommandons de tester votre algorithme de génération de hachage aux deux extrémités (ingestion et interrogation) pour vous assurer que vous obtenez les mêmes résultats.

Voici quelques exemples de code sur `host_name` lesquels générer la valeur hachée.

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 dépendance dans Maven

```
        <dependency>
            <groupId>net.jpountz.lz4</groupId>
            <artifactId>lz4</artifactId>
            <version>1.3.0</version>
        </dependency>
```

Sécurité

- Pour un accès continu à Timestream for LiveAnalytics, assurez-vous que les clés de chiffrement sont sécurisées et ne sont pas révoquées ou rendues inaccessibles.
- Surveillez les journaux d'APIaccès depuis AWS CloudTrail. Auditez et révoquez tout modèle d'accès anormal provenant d'utilisateurs non autorisés.
- Suivez les directives supplémentaires décrites dansBonnes pratiques de sécurité pour Amazon Timestream pour LiveAnalytics.

Configuration d'Amazon Timestream pour LiveAnalytics

Configurez la période de conservation des données pour la mémoire et la mémoire magnétique en fonction des exigences en matière de traitement des données, de stockage, de performance des requêtes et de coûts.

- Définissez la durée de conservation des données de la mémoire en fonction des exigences de votre application en matière de traitement des données arrivant tardivement. Les données arrivées tardivement sont des données entrantes dont l'horodatage est antérieur à l'heure actuelle. Il est émis par des ressources qui regroupent les événements pendant un certain temps avant d'envoyer les données à Timestream LiveAnalytics, ainsi que par des ressources présentant une connectivité intermittente, par exemple un capteur IoT connecté par intermittence.
- Si vous vous attendez à ce que les données arrivant en retard arrivent parfois avec un horodatage antérieur à la durée de conservation de la mémoire, vous devez activer les écritures magnétiques pour votre table. Une fois que vous avez configuré un tableau, celui-ci accepte les données horodatées avant la date de conservation de votre mémoire, mais MagneticStoreWritesProperties pendant la période de conservation de votre mémoire magnétique. EnableMagneticStoreWrites

- Tenez compte des caractéristiques des requêtes que vous prévoyez d'exécuter sur Timestream, LiveAnalytics telles que les types de requêtes, la fréquence, la plage de temps et les exigences de performance. Cela est dû au fait que la mémoire et la mémoire magnétique sont optimisées pour différents scénarios. La mémoire est optimisée pour les point-in-time requêtes rapides qui traitent de petites quantités de données récentes envoyées à Timestream pour. LiveAnalytics Le magasin magnétique est optimisé pour les requêtes analytiques rapides qui traitent des volumes moyens à importants de données envoyés à Timestream pour. LiveAnalytics
- La durée de conservation de vos données doit également être influencée par les exigences de coût de votre système.

  Par exemple, imaginez un scénario dans lequel le seuil de réception tardive des données pour votre application est de 2 heures et où vos applications envoient de nombreuses requêtes traitant l'équivalent d'une journée, d'une semaine ou d'un mois de données. Dans ce cas, vous souhaiterez peut-être configurer une période de rétention plus courte pour la mémoire (2 à 3 heures) et permettre à davantage de données de circuler vers la mémoire magnétique, étant donné que la mémoire magnétique est optimisée pour les requêtes analytiques rapides.

Comprenez l'impact de l'augmentation ou de la diminution de la durée de conservation des données de la mémoire et de la mémoire magnétique d'une table existante.

- Lorsque vous réduisez la durée de conservation de la mémoire, les données sont déplacées de la mémoire vers la mémoire magnétique, et ce transfert de données est permanent. Timestream for LiveAnalytics ne récupère pas les données de la mémoire magnétique pour alimenter la mémoire. Lorsque vous réduisez la durée de conservation de la mémoire magnétique, les données sont supprimées du système et la suppression des données est permanente.
- Lorsque vous augmentez la durée de conservation de la mémoire ou de la mémoire magnétique, la modification prend effet pour les données envoyées à Timestream à LiveAnalytics partir de ce moment. Timestream for LiveAnalytics ne récupère pas les données de la mémoire magnétique pour alimenter la mémoire. Par exemple, si la période de conservation de la mémoire a été initialement fixée à 2 heures puis augmentée à 24 heures, il faudra 22 heures pour que la mémoire contienne 24 heures de données.

écrit

- Assurez-vous que l'horodatage des données entrantes n'est pas antérieur à la durée de conservation des données configurée pour la mémoire et au plus tard à la future période

d'ingestion définie dans. Quotas L'envoi de données avec un horodatage en dehors de ces limites entraînera le rejet des données par Timestream, LiveAnalytics sauf si vous activez les écritures par stockage magnétique pour votre table. Si vous activez les écritures en mémoire magnétique, assurez-vous que l'horodatage des données entrantes n'est pas antérieur à la durée de conservation des données configurée pour la mémoire magnétique.

- Si vous vous attendez à ce que les données arrivent en retard, activez le stockage magnétique pour votre table. Cela permettra l'ingestion de données dont l'horodatage se situe en dehors de la période de conservation de votre mémoire, mais toujours pendant la période de conservation de votre mémoire magnétique. Vous pouvez le définir en mettant à jour l'`EnableMagneticStoreWrites`indicateur dans le `MagneticStoreWritesProperties` tableau. Cette propriété est fausse par défaut. Notez que les écritures dans le magasin magnétique ne seront pas immédiatement disponibles pour les requêtes. Ils seront disponibles dans les 6 heures.
- Ciblez les charges de travail à haut débit sur la mémoire en vous assurant que les horodatages des données ingérées respectent les limites de rétention de la mémoire. Les écritures dans la mémoire magnétique sont limitées à un nombre maximum de partitions de mémoire magnétique actives qui peuvent être ingérées simultanément pour une base de données. Vous pouvez voir cette `ActiveMagneticStorePartitions` métrique dans CloudWatch. Pour réduire les cloisons magnétiques actives, essayez de réduire le nombre de séries et la durée pendant laquelle vous ingérez simultanément des unités de stockage magnétiques.
- Lorsque vous envoyez des données à Timestream pour LiveAnalytics, regroupez plusieurs enregistrements en une seule demande afin d'optimiser les performances d'ingestion des données.
  - Il est avantageux de regrouper les enregistrements de la même série chronologique et les enregistrements portant le même nom de mesure.
  - Batch autant d'enregistrements que possible dans une seule demande, à condition que les demandes respectent les limites de service définies dansQuotas.
  - Utilisez des attributs communs dans la mesure du possible pour réduire les coûts de transfert et d'ingestion des données. Pour plus d'informations, consultez WriteRecords API.
- Si vous rencontrez des défaillances partielles côté client lors de l'écriture de données dans Timestream pour LiveAnalytics, vous pouvez renvoyer le lot d'enregistrements dont l'ingestion a échoué après avoir résolu la cause du rejet.
- Les données classées par horodatage présentent de meilleures performances d'écriture.
- Amazon Timestream LiveAnalytics for est conçu pour s'adapter automatiquement aux besoins de votre application. Lorsque Timestream for LiveAnalytics Notices augmente le nombre de demandes d'écriture provenant de votre application, celle-ci peut être confrontée à un certain

niveau de limitation de la mémoire initiale. Si votre application est confrontée à une limitation de la mémoire, continuez à envoyer des données à Timestream LiveAnalytics au même débit (ou à un débit accru) pour permettre à Timestream de s'adapter automatiquement LiveAnalytics aux besoins de votre application. Si vous constatez un ralentissement de la réserve magnétique, vous devez diminuer le taux d'ingestion de la réserve magnétique jusqu'à ce que votre nombre `ActiveMagneticStorePartitions` diminue.

Chargement par lots

Les meilleures pratiques en matière de chargement par lots sont décrites dansBonnes pratiques en matière de chargement par lots.

Requêtes

Vous trouverez ci-dessous des suggestions de bonnes pratiques pour les requêtes avec Amazon LiveAnalytics Timestream pour.

- N'incluez que les noms de mesures et de dimensions essentiels à la requête. L'ajout de colonnes superflues augmentera les analyses de données, ce qui aura un impact sur les performances des requêtes.
- Avant de déployer votre requête en production, nous vous recommandons de consulter les informations relatives aux requêtes pour vous assurer que l'élagage spatial et temporel est optimal. Pour de plus amples informations, veuillez consulter Utilisation des informations relatives aux requêtes pour optimiser les requêtes dans Amazon Timestream.
- Dans la mesure du possible, transférez le calcul des données vers Timestream pour LiveAnalytics utiliser les agrégats intégrés et les fonctions scalaires dans la clause et la SELECT clause, le cas échéant, afin d'améliorer les WHERE performances des requêtes et de réduire les coûts. Consultez SELECT et Fonctions d'agrégation.
- Dans la mesure du possible, utilisez des fonctions approximatives. Par exemple, utilisez APPROX _ DISTINCT au lieu de COUNT (DISTINCTcolumn_name) pour optimiser les performances des requêtes et réduire le coût des requêtes. Consultez Fonctions d'agrégation.
- Utilisez une CASE expression pour effectuer des agrégations complexes au lieu de sélectionner plusieurs fois dans la même table. Consultez La CASE déclaration.
- Dans la mesure du possible, incluez une plage de temps dans la WHERE clause de votre requête. Cela permet d'optimiser les performances et les coûts des requêtes. Par exemple, si vous n'avez

besoin que de la dernière heure de données dans votre ensemble de données, incluez un prédicat temporel tel que time > ago (1h). Consultez SELECT et Intervalle et durée.

- Lorsqu'une requête accède à un sous-ensemble de mesures d'une table, incluez toujours les noms des mesures dans la WHERE clause de la requête.
- Dans la mesure du possible, utilisez l'opérateur d'égalité lorsque vous comparez des dimensions et des mesures dans la WHERE clause d'une requête. Un prédicat d'égalité sur les dimensions et les noms de mesures permet d'améliorer les performances des requêtes et de réduire les coûts des requêtes.
- Dans la mesure du possible, évitez d'utiliser des fonctions dans la WHERE clause pour optimiser les coûts.
- Évitez d'utiliser LIKE la clause plusieurs fois. Utilisez plutôt des expressions régulières lorsque vous filtrez plusieurs valeurs sur une colonne de chaîne. Consultez Fonctions d'expression régulière.
- Utilisez uniquement les colonnes nécessaires dans la clause GROUP BY d'une requête.
- Si le résultat de la requête doit être dans un ordre spécifique, spécifiez explicitement cet ordre dans la clause ORDER BY de la requête la plus éloignée. Si le résultat de votre requête ne nécessite pas de classement, évitez d'utiliser une clause ORDER BY pour améliorer les performances de la requête.
- Utilisez une LIMIT clause si vous n'avez besoin que des N premières lignes de votre requête.
- Si vous utilisez une clause ORDER BY pour examiner les valeurs N supérieures ou inférieures, utilisez une LIMIT clause pour réduire les coûts de requête.
- Utilisez le jeton de pagination contenu dans la réponse renvoyée pour récupérer les résultats de la requête. Pour plus d'informations, consultez Requête.
- Si vous avez commencé à exécuter une requête et que vous vous rendez compte que celle-ci ne renverra pas les résultats que vous recherchez, annulez-la pour réduire les coûts. Pour plus d'informations, consultez CancelQuery.
- Si votre application connaît un ralentissement, continuez à envoyer des données à Amazon Timestream au même rythme LiveAnalytics pour permettre à Amazon Timestream de s'adapter automatiquement aux besoins de débit de LiveAnalytics requêtes de votre application.
- Si les exigences de simultanéité des requêtes de vos applications dépassent les limites par défaut de Timestream pour LiveAnalytics, contactez AWS Support pour obtenir des augmentations de limite.

Requêtes planifiées

Les requêtes planifiées vous aident à optimiser vos tableaux de bord en précalculant certaines statistiques agrégées à l'échelle du parc. Il est donc naturel de se demander comment prendre votre cas d'utilisation pour identifier les résultats à précalculer et comment utiliser ces résultats stockés dans une table dérivée pour créer votre tableau de bord. La première étape de ce processus consiste à identifier les panneaux à précalculer. Voici quelques directives de haut niveau :

- Tenez compte des octets scannés par les requêtes utilisées pour remplir les panneaux, de la fréquence de rechargement des tableaux de bord et du nombre d'utilisateurs simultanés susceptibles de charger ces tableaux de bord. Vous devez commencer par les tableaux de bord les plus fréquemment chargés et analyser des quantités importantes de données. Les deux premiers tableaux de bord de l'exemple de tableau de bord agrégé ainsi que le tableau de bord agrégé de l'exemple de hiérarchisation sont de bons exemples de tels tableaux de bord.
- Déterminez quels calculs sont utilisés de manière répétée. Bien qu'il soit possible de créer une requête planifiée pour chaque panneau et chaque valeur variable utilisée dans le panneau, vous pouvez optimiser considérablement vos coûts et le nombre de requêtes planifiées en recherchant des moyens d'utiliser un seul calcul pour précalculer les données nécessaires à plusieurs panneaux.
- Tenez compte de la fréquence de vos requêtes planifiées pour actualiser les résultats matérialisés dans la table dérivée. Vous souhaiteriez analyser la fréquence d'actualisation d'un tableau de bord par rapport à la fenêtre temporelle demandée dans un tableau de bord par rapport au binning utilisé dans le précalcul ainsi qu'aux panneaux des tableaux de bord. Par exemple, si un tableau de bord qui trace les agrégats horaires des derniers jours n'est actualisé qu'une fois toutes les quelques heures, vous souhaiterez peut-être configurer vos requêtes planifiées pour qu'elles ne soient actualisées qu'une fois toutes les 30 minutes ou toutes les heures. En revanche, si vous disposez d'un tableau de bord qui trace des agrégats par minute et qui est actualisé toutes les minutes environ, vous souhaiterez que vos requêtes planifiées actualisent les résultats toutes les minutes ou toutes les quelques minutes.
- Déterminez quels modèles de requêtes peuvent être optimisés davantage (du point de vue du coût des requêtes et de la latence des requêtes) à l'aide de requêtes planifiées. Par exemple, lorsque vous calculez les valeurs de dimension uniques fréquemment utilisées comme variables dans les tableaux de bord, ou lorsque vous renvoyez le dernier point de données émis par un capteur ou le premier point de données émis par un capteur après une certaine date, etc. Certains de ces exemples de modèles sont présentés dans ce guide.

Les considérations précédentes auront un impact significatif sur vos économies lorsque vous déplacerez votre tableau de bord pour interroger les tables dérivées, l'actualité des données de vos tableaux de bord et les coûts engendrés par les requêtes planifiées.

Applications clientes et intégrations prises en charge

Exécutez votre application client depuis la même région que Timestream afin de réduire les LiveAnalytics latences du réseau et les coûts de transfert de données. Pour plus d'informations sur l'utilisation d'autres services, consultezUtilisation d'autres services . Voici d'autres liens utiles.

- Meilleures pratiques en matière AWS de développement avec le AWS SDK for Java
- Bonnes pratiques d'utilisation des AWS Lambda fonctions
- Bonnes pratiques pour Amazon Managed Service pour Apache Flink
- Meilleures pratiques pour créer des tableaux de bord dans Grafana

Général

- Assurez-vous de suivre le cadre The AWS Well-Architected lorsque vous utilisez Timestream pour. LiveAnalytics Ce livre blanc fournit des conseils sur les meilleures pratiques en matière d'excellence opérationnelle, de sécurité, de fiabilité, d'efficacité des performances et d'optimisation des coûts.

Mesurage et optimisation des coûts

Avec Amazon Timestream LiveAnalytics pour, vous ne payez que pour ce que vous utilisez. Diffusion temporelle des LiveAnalytics compteurs séparément pour les écritures, les données stockées et les données numérisées par des requêtes. Le prix de chaque dimension de mesure est indiqué sur la page de tarification. Vous pouvez estimer votre facture mensuelle à l'aide du calculateur de prix Amazon Timestream LiveAnalytics for Pricing.

Cette section décrit le fonctionnement de la mesure pour les écritures, le stockage et les requêtes dans Timestream for. LiveAnalytics Des exemples de scénarios et de calculs sont également fournis. En outre, une liste des meilleures pratiques en matière d'optimisation des coûts est incluse. Vous pouvez sélectionner un sujet ci-dessous :

Rubriques

écrit

La taille d'écriture de chaque événement de série chronologique est calculée comme la somme de la taille de l'horodatage et d'un ou plusieurs noms de dimension, valeurs de dimension, noms de mesures et valeurs de mesure. La taille de l'horodatage est de 8 octets. La taille des noms de dimension, des valeurs de dimension et des noms de mesures correspond à la longueur des UTF -8 octets codés de la chaîne représentant chaque nom de dimension, valeur de dimension et nom de mesure. La taille de la valeur de mesure dépend du type de données. Il s'agit de 1 octet pour le type de données booléen, de 8 octets pour bigint et double, et de la longueur des UTF -8 octets codés pour les chaînes. Chaque écriture est comptée en unités de 1 KiB.

Deux exemples de calculs sont fournis ci-dessous :

Rubriques

Calcul de la taille d'écriture d'un événement de série chronologique

Prenons l'exemple d'un événement chronologique représentant l'CPUutilisation d'une EC2 instance, comme indiqué ci-dessous :

Heure	region	az	vpc	Hostname	nom_mesure	valeur_mesure : double
1602983435238563000	us-east-1	1d	vpc-1a2b3c4d	Hôte-24GJU	utilisation du processeur	35,0

La taille d'écriture de l'événement de série chronologique peut être calculée comme suit :

- temps = 8 octets
- première dimension = 15 octets (`region+us-east-1`)
- deuxième dimension = 4 octets (`az+1d`)
- troisième dimension = 15 octets (`vpc+vpc-1a2b3c4d`)
- quatrième dimension = 18 octets (`hostname+host-24Gju`)
- nom de la mesure = 15 octets (`cpu_utilization`)
- valeur de la mesure = 8 octets

Taille d'écriture de l'événement chronologique = 83 octets

Calcul du nombre d'écritures

Supposons maintenant que 100 EC2 instances, similaires à l'instance décrite dansCalcul de la taille d'écriture d'un événement de série chronologique, émettent des métriques toutes les 5 secondes. Le nombre total d'écritures mensuelles pour les EC2 instances variera en fonction du nombre d'événements de série chronologique existants par écriture et de l'utilisation d'attributs communs lors du traitement par lots d'événements de séries chronologiques. Un exemple de calcul du total des écritures mensuelles est fourni pour chacun des scénarios suivants :

Rubriques

- Un événement chronologique par écrit
- Regrouper les événements d'une série chronologique dans un écrit
- Regrouper les événements de séries chronologiques et utiliser des attributs communs lors d'une écriture

Un événement chronologique par écrit

Si chaque écriture ne contient qu'un seul événement de série chronologique, le total des écritures mensuelles est calculé comme suit :

- 100 événements de séries chronologiques = 100 écritures toutes les 5 secondes
- x 12 écritures/minute = 1 200 écritures
- x 60 minutes/heure = 72 000 écritures

- x 24 heures/jour = 1 728 000 écritures
- x 30 jours/mois = 51 840 000 écritures

Nombre total d'écritures mensuelles = 51 840 000

Regrouper les événements d'une série chronologique dans un écrit

Étant donné que chaque écriture est mesurée en unités de 1 Ko, une écriture peut contenir un lot de 12 événements chronologiques (998 octets) et le total des écritures mensuelles est calculé comme suit :

- 100 événements de séries chronologiques = 9 écritures (12 événements de séries chronologiques par écriture) toutes les 5 secondes
- x 12 écritures/minute = 108 écritures
- x 60 minutes/heure = 6 480 écritures
- x 24 heures/jour = 155 520 écritures
- x 30 jours/mois = 4 665 600 écritures

Nombre total d'écritures mensuelles = 4 665 600

Regrouper les événements de séries chronologiques et utiliser des attributs communs lors d'une écriture

Si la région, l'az, le vpc et le nom de la mesure sont communs à 100 EC2 instances, les valeurs communes peuvent être spécifiées une seule fois par écriture et sont appelées attributs communs. Dans ce cas, la taille des attributs communs est de 52 octets et celle des événements de la série chronologique est de 27 octets. Étant donné que chaque écriture est mesurée en unités de 1 KiB, une écriture peut contenir 36 événements chronologiques et des attributs communs, et le total des écritures mensuelles est calculé comme suit :

- 100 événements de séries chronologiques = 3 écritures (36 événements de séries chronologiques par écriture) toutes les 5 secondes
- x 12 écritures/minute = 36 écritures
- x 60 minutes/heure = 2 160 écritures
- x 24 heures/jour = 51 840 écritures
- x 30 jours/mois = 1 555 200 écritures

Nombre total d'écritures mensuelles = 1 555 200

> **Note**
>
> En raison de l'utilisation du traitement par lots, des attributs communs et de l'arrondissement des écritures par unités de 1 Ko, la taille de stockage des événements de série chronologique peut être différente de la taille d'écriture.

Stockage

La taille de stockage de chaque événement de série chronologique dans la mémoire et dans la mémoire magnétique est calculée comme la somme de la taille de l'horodatage, des noms de dimensions, des valeurs de dimension, des noms de mesure et des valeurs de mesure. La taille de l'horodatage est de 8 octets. La taille des noms de dimension, des valeurs de dimension et des noms de mesure correspond à la longueur des UTF -8 octets codés de chaque chaîne représentant le nom de dimension, la valeur de dimension et le nom de mesure. La taille de la valeur de mesure dépend du type de données. Il s'agit de 1 octet pour les types de données booléens, de 8 octets pour bigint et double, et de la longueur des UTF -8 octets codés pour les chaînes. Chaque mesure est stockée sous la forme d'un enregistrement distinct dans Amazon Timestream LiveAnalytics, c'est-à-dire que si votre événement de série chronologique comporte quatre mesures, quatre enregistrements seront stockés pour cet événement de série chronologique.

En prenant l'exemple de l'événement de série chronologique représentant l'CPUutilisation d'une EC2 instance (voirCalcul de la taille d'écriture d'un événement de série chronologique), la taille de stockage de l'événement de série chronologique est calculée comme suit :

- temps = 8 octets
- première dimension = 15 octets (`region+us-east-1`)
- deuxième dimension = 4 octets (`az+1d`)
- troisième dimension = 15 octets (`vpc+vpc-1a2b3c4d`)
- quatrième dimension = 18 octets (`hostname+host-24Gju`)
- nom de la mesure = 15 octets (`cpu_utilization`)
- valeur de la mesure = 8 octets

Taille de stockage de l'événement chronologique = 83 octets

> **Note**
>
> La mémoire mémoire est mesurée en Go par heure et la mémoire magnétique est mesurée en Go par mois.

Requêtes

Les requêtes sont facturées en fonction de la durée des unités de calcul Timestream (TCUs) utilisées par votre application en TCU heures, comme indiqué sur la page de tarification d'Amazon Timestream. Le moteur de requêtes Amazon Timestream LiveAnalytics for élimine les données non pertinentes lors du traitement d'une requête. Les requêtes comportant des projections et des prédicats, notamment des plages temporelles, des noms de mesures et/ou des noms de dimensions, permettent au moteur de traitement des requêtes d'élaguer une quantité importante de données et de réduire les coûts des requêtes.

Optimisation des coûts

Pour optimiser le coût des écritures, du stockage et des requêtes, appliquez les meilleures pratiques suivantes avec Amazon LiveAnalytics Timestream pour :

- Batch par lots de plusieurs événements chronologiques par écriture afin de réduire le nombre de demandes d'écriture.
- Envisagez d'utiliser des enregistrements à mesures multiples, qui vous permettent d'écrire plusieurs mesures de séries chronologiques en une seule demande d'écriture et de stocker vos données de manière plus compacte. Cela réduit le nombre de demandes d'écriture ainsi que les coûts de stockage des données et de requêtes.
- Utilisez des attributs communs avec le traitement par lots pour regrouper un plus grand nombre d'événements chronologiques par écriture afin de réduire davantage le nombre de demandes d'écriture.
- Définissez la durée de conservation des données de la mémoire en fonction des exigences de votre application en matière de traitement des données arrivant tardivement. Les données arrivées tardivement sont des données entrantes dont l'horodatage est antérieur à l'heure actuelle et en dehors de la période de conservation de la mémoire.
- Configurez la conservation des données du magasin magnétique en fonction de vos besoins de stockage de données à long terme.

- Lorsque vous rédigez des requêtes, n'incluez que les noms de mesure et de dimension essentiels à la requête. L'ajout de colonnes superflues augmentera les analyses de données et, par conséquent, le coût des requêtes. Nous vous recommandons de consulter les informations des requêtes pour évaluer l'efficacité d'élagage des dimensions et mesures incluses.
- Dans la mesure du possible, incluez une plage de temps dans la WHERE clause de votre requête. Par exemple, si vous n'avez besoin que de la dernière heure de données dans votre ensemble de données, incluez un prédicat temporel tel `quetime > ago(1h)`.
- Lorsqu'une requête accède à un sous-ensemble de mesures d'une table, incluez toujours les noms des mesures dans la WHERE clause de la requête.
- Si vous avez commencé à exécuter une requête et que vous vous rendez compte que celle-ci ne renverra pas les résultats que vous recherchez, annulez la requête pour économiser sur les coûts.

Surveillance avec Amazon CloudWatch

Vous pouvez surveiller Timestream pour utiliser LiveAnalytics Amazon CloudWatch, qui collecte et traite les données brutes de Timestream pour LiveAnalytics en faire des métriques lisibles. near-real-time Ces statistiques sont enregistrées pour une durée de deux semaines ; par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web s'exécute. Par défaut, le Timestream pour les données LiveAnalytics métriques est automatiquement envoyé par tranches d'une CloudWatch minute ou de 15 minutes. Pour plus d'informations, consultez Qu'est-ce qu'Amazon CloudWatch ? dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

- Comment utiliser Timestream pour les LiveAnalytics statistiques ?
- Timestream pour les LiveAnalytics métriques et les dimensions
- Création d' CloudWatch alarmes pour surveiller Timestream LiveAnalytics

Comment utiliser Timestream pour les LiveAnalytics statistiques ?

Les métriques rapportées par Timestream LiveAnalytics fournissent des informations que vous pouvez analyser de différentes manières. La liste suivante présente certaines utilisations courantes des métriques. Voici quelques suggestions pour vous aider à démarrer, qui ne forment pas une liste exhaustive.

Comment puis-je ?	Métriques pertinentes
How can I determine if any system errors occurred?	Vous pouvez superviser SystemErrors pour déterminer si des demandes ont entraîné un code erreur du serveur. En règle générale, cette métrique doit être égale à zéro. Si tel n'est pas le cas, vous devez enquêter.
How can I monitor the amount of data in the memory store?	Vous pouvez effectuer une surveillance MemoryCumulativeBytesMetered pendant la période spécifiée afin de contrôler la quantité de données stockées dans la mémoire en octets. Cette métrique est émise toutes les heures et vous pouvez suivre les octets stockés sur un compte ainsi que la granularité de la base de données. La mémoire stockée est mesurée en Go par heure (le coût du stockage de 1 Go de données pendant une heure). Ainsi, en multipliant la valeur horaire de MemoryCumulativeBytesMetered par la tarification en Go par heure dans votre région, vous obtiendrez le coût horaire encouru. Dimensions : opération (stockage) DatabaseName, nom de la métrique
How can I monitor the amount of data in the magnetic store?	Vous pouvez effectuer une surveillance MagneticCumulativeBytesMetered pendant la période spécifiée pour surveiller la quantité de données stockées dans le stockage magnétique en octets. Cette métrique est émise toutes les heures et vous pouvez suivre les octets stockés sur un compte ainsi que la granularité de la base de données. La mémoire stockée est mesurée en Go par mois (le coût du stockage de 1 Go de données pendant un mois). Ainsi, en multipliant la valeur horaire de MagneticCumulativeBytesMetered par le prix en Go par mois dans votre région, vous obtiendrez le coût horaire encouru. Par exemple, si la valeur de MagneticCumulativeBytesMetered est de 107374182400 octets (100 Go), le prix horaire de 1 Go de données dans Magnetic Store = (0,03) (prix us-east-1)/(30,4*24). En multipliant cette

Comment puis-je ?	Métriques pertinentes
	valeur par le nombre `MagneticCumulativeBytesMetered` en Go, vous obtiendrez ~0,004 $ pour cette heure. Dimensions : opération (stockage) DatabaseName, nom de la métrique
`How can I monitor the data scanned by queries?`	Vous pouvez effectuer une surveillance pendant la `CumulativeBytesMetered` période spécifiée afin de surveiller les données scannées par les requêtes (en octets) envoyées à Timestream pour. LiveAnalytics Cette métrique est émise après l'exécution de la requête et vous pouvez suivre les données numérisées selon la granularité du compte et de la base de données. Vous pouvez calculer le coût de la requête pour une période donnée en multipliant la valeur de la métrique par le prix par Go numérisé dans votre région. Les octets analysés par les requêtes planifiées sont pris en compte dans cette métrique. Dimensions : opération (requête) DatabaseName, nom de la métrique

Comment puis-je ?	Métriques pertinentes
How can I monitor the data scanned by scheduled queries?	Vous pouvez effectuer une surveillance pendant la `CumulativeBytesMetered` période spécifiée afin de surveiller les données scannées par des requêtes planifiées (en octets) exécutées par Timestream pour. LiveAnalytics Cette métrique est émise après l'exécution de la requête et vous pouvez suivre les données numérisées selon la granularité du compte et de la base de données. Vous pouvez calculer le coût de la requête pour une période donnée en multipliant la valeur de la métrique par le prix par Go numérisé dans votre région. **Note** Les octets mesurés sont également pris en compte dans la requête`CumulativeBytesMetered` . Dimensions : Operation (TriggeredScheduledQuery) DatabaseName, Nom de la métrique

Comment puis-je ?	Métriques pertinentes
How can I monitor the number of records ingested?	Vous pouvez effectuer une surveillance NumberOfRecords pendant la période spécifiée pour contrôler le nombre d'enregis trements ingérés. Vous pouvez suivre les octets stockés sur un compte ainsi que la granularité de la base de données. Vous pouvez également utiliser cette métrique pour surveiller les écritures effectuées par les requêtes planifiées lorsque les résultats des requêtes sont écrits dans une table séparée. Lorsque vous utilisez le WriteRecords API, la métrique est émise pour chaque WriteRecords demande, la dimension CloudWatch Operation étantWriteRecords . Lorsque vous utilisez le BatchLoad ou ScheduledQuery APIs, la métrique est émise à des intervalles déterminés par le service jusqu'à la fin de la tâche. La dimension d' CloudWatch opération pour cette métrique est BatchLoad soitScheduledQuery , API soit, selon la mesure utilisée. Dimensions : opération (WriteRecords BatchLoad, ou Scheduled Query) DatabaseName, nom de la métrique

Comment puis-je ?	Métriques pertinentes
How can I monitor the cost of records ingested?	Vous pouvez surveiller CumulativeBytesMetered pour contrôler le nombre d'octets ingérés qui entraînent des coûts. Vous pouvez suivre les octets stockés sur un compte ainsi que la granularité de la base de données. Les enregistrements ingérés sont mesurés en octets cumulés. En multipliant la valeur du prix CumulativeBytesMetered par Writes dans votre région, vous obtenez le coût d'ingestion encouru. Lorsque vous utilisez le WriteRecords API, cette métrique est émise pour chaque WriteRecords demande, la dimension CloudWatch Operation étantWriteRecords . Lorsque vous utilisez le BatchLoad ou ScheduledQuery API, la métrique est émise à des intervalles déterminés par le service jusqu'à la fin de la tâche. La dimension CloudWatch d'opération pour cette métrique est BatchLoad ou ScheduledQuery dépend de celle qui API est utilisée. Dimensions : opération (WriteRecords BatchLoad, ou Scheduled Query) DatabaseName, nom de la métrique
How can I monitor the Timestream Compute Units (TCUs) used in my account?	Vous pouvez effectuer une surveillance QueryTCU sur la période spécifiée afin de surveiller les unités de calcul consommées pour la charge de travail des requêtes dans le compte. Cette métrique est émise avec des unités de calcul maximales et minimales pour chaque minute pendant la charge de travail active des requêtes depuis le compte. Unités Count: Statistiques valides : minimum, maximum Métrique : ResourceCount Service: Timestream Dimensioni :Resource: QueryTCU,Type: Resource, Class: OnDemand

Timestream pour les LiveAnalytics métriques et les dimensions

Lorsque vous interagissez avec Timestream pour LiveAnalytics, il envoie les métriques et dimensions suivantes à Amazon. CloudWatch Toutes les métriques sont agrégées et signalées chaque minute. Vous pouvez utiliser les procédures suivantes pour afficher les statistiques de Timestream pour. LiveAnalytics

Pour afficher les métriques à l'aide de la CloudWatch console

Les métriques sont d'abord regroupées par espace de noms de service, puis par les différentes combinaisons de dimension au sein de chaque espace de noms.

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Si nécessaire, changez la région. Dans la barre de navigation, choisissez la région dans laquelle se trouvent vos AWS ressources. Pour plus d'informations, consultez Points de terminaison du service AWS.
3. Dans le panneau de navigation, sélectionnez Métriques.
4. Sous l'onglet Toutes les métriques, choisissez `AWS/Timestream for LiveAnalytics`..

Pour consulter les statistiques à l'aide du AWS CLI

- À partir d'une invite de commande, utilisez la commande suivante :

```
aws cloudwatch list-metrics --namespace "AWS/Timestream"
```

Dimensions pour Timestream pour les métriques LiveAnalytics

Les métriques de Timestream for LiveAnalytics sont qualifiées par les valeurs du compte, du nom de la table ou de l'opération. Vous pouvez utiliser la CloudWatch console pour récupérer les LiveAnalytics données Timestream selon l'une des dimensions indiquées dans le tableau suivant :

Dimension	Description
`DatabaseName`	Cette dimension limite les données à un flux temporel spécifiqu e pour LiveAnalytics la base de données. Cette valeur peut être

Dimension	Description
	n'importe quelle base de données de la région actuelle et du AWS compte courant
`Operation`	Cette dimension limite les données à l'un des flux temporels pour les LiveAnalytics opérations, telles que`Storage`, `WriteRecords BatchLoad` , ou. `ScheduledQuery` Consultez le Timestream for LiveAnalytics Query API Reference pour une liste des valeurs disponibles.
`TableName`	Cette dimension limite les données à une table spécifique dans une base de données Timestream for LiveAnalyticss .

> **Important**
>
> `CumulativeBytesMetered`, `UserErrors` et `SystemErrors` les métriques n'ont que la `Operation` dimension. `SuccessfulRequestLatency`les métriques ont toujours `Operation` une dimension, mais peuvent également avoir les `TableName` dimensions `DatabaseName` et, selon la valeur de`Operation`. Cela est dû au fait que Timestream pour les opérations LiveAnalytics au niveau des tables possède `DatabaseName` et en `TableName` tant que dimensions, mais pas les opérations au niveau du compte.

Timestream pour les métriques LiveAnalytics

> **Note**
>
> Amazon CloudWatch agrège tous les flux temporels suivants pour les LiveAnalytics métriques à des intervalles d'une minute.

Métriques générales

Métrique	Description
`SuccessfulRequestLatency`	Les demandes réussies adressées à Timestrea m LiveAnalytics pendant la période spécifiée.

Métrique	Description
	SuccessfulRequestLatency peut fournir deux types d'informations différents : • Le temps écoulé pour les demandes réussies (minimum, maximum, somme ou moyenne). • Nombre de requêtes réussies (SampleCount). SuccessfulRequestLatency reflète uniquement l'activité dans Timestream pour LiveAnalytics et ne prend pas en compte la latence du réseau ou l'activité côté client. Unités `Milliseconds` : Dimensions • `DatabaseName` • `TableName` • `Operation` Statistiques valides : • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`

Rédaction et mesures de stockage

Métrique	Description
`MagneticStoreRejectedRecordCount`	Le nombre d'enregistrements écrits en mémoire magnétique qui ont été rejetés de manière asynchrone. Cela peut se produire si le nouvel enregistrement possède une version inférieur e à la version actuelle ou si le nouvel enregistr ement possède une version égale à la version actuelle mais contient des données différentes. Unités `Count`: Dimensions • `DatabaseName` • `TableName` • `Operation` Statistiques valides : • `Sum` • `SampleCount`
`MagneticStoreRejectedUpload UserFailures`	Le nombre de rapports d'enregistrement rejetés par Magnetic Store qui n'ont pas été télécharg és en raison d'erreurs de l'utilisateur. Cela peut être dû à IAM des autorisations mal configurées ou à la suppression d'un compartiment S3. Unités `Count`: Dimensions • `DatabaseName` • `TableName` • `Operation`

Métrique	Description
	Statistiques valides : • `Sum` • `SampleCount`
`MagneticStoreRejectedUploadSystemFailures`	Le nombre de rapports d'enregistrement rejetés par Magnetic Store qui n'ont pas été télécharg és en raison d'erreurs du système. Unités `Count`: Dimensions • `DatabaseName` • `TableName` • `Operation` Statistiques valides : • `Sum` • `SampleCount`

Métrique	Description
`ActiveMagneticStorePartitions`	Le nombre de partitions de stockage magnétiques ingérant activement des données à un moment donné. Unités `Count`: Dimensions • `DatabaseName` • `Operation` Statistiques valides : • `Sum` • `SampleCount`

Métrique	Description
`MagneticStorePendingRecordsLatency`	Les plus anciens écrivent dans une mémoire magnétique qui n'est pas disponible pour les requêtes. Les dossiers enregistrés dans le magasin magnétique pourront être consultés dans les 6 heures. Unités `Milliseconds` : Dimensions • `DatabaseName` • `TableName` • `Operation` Statistiques valides : • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`

Métrique	Description
`MemoryCumulativeBytesMetered`	La quantité de données stockées dans la mémoire, en octets Unités : `Bytes` Dimensions : `Operation` Statistiques valides : • `Average`
`MagneticCumulativeBytesMetered`	La quantité de données stockées dans le magasin magnétique, en octets Unités : `Bytes` Dimensions : `Operation` Statistiques valides : • `Average`
`CumulativeBytesMetered`	La quantité de données mesurée par ingestion dans Timestream LiveAnalytics, en octets. Unités : `Bytes` Dimensions : `Operation` Statistiques valides : `Sum`
`NumberOfRecords`	Le nombre d'enregistrements ingérés dans Timestream pour. LiveAnalytics Unités : `Count` Dimensions : `Operation` Statistiques valides : `Sum`

Métriques des requêtes

Métrique	Description
`CumulativeBytesMetered`	La quantité de données numérisées par les requêtes envoyées à Timestream LiveAnaly tics, en octets. Unités : `Bytes` Dimensions : `Operation` Statistiques valides : • `Sum`
`ResourceCount`	Les unités de calcul Timestream (TCUs) consommées pour la charge de travail des requêtes dans le compte. Cette métrique est émise avec des unités de calcul maximales et minimales pour chaque minute pendant la charge de travail active des requêtes depuis le compte. Unités `Count`: Statistiques valides : minimum, maximum `Service: Timestream` Dimension i :`Resource: QueryTCU`,`Type: Resource`, `Class: OnDemand`

Métriques d'erreur

Métrique	Description
`SystemErrors`	Les demandes adressées à Timestream pour LiveAnalytics cela génèrent un SystemErr or pendant la période spécifiée. A indique

Métrique	Description
	SystemError généralement une erreur de service interne. Unités : `Count` Dimensions : `Operation` Statistiques valides : • `Sum` • `SampleCount`
`UserErrors`	Les demandes adressées à Timestream pour LiveAnalytics cela génèrent une InvalidRequest erreur pendant la période spécifiée. Un indique InvalidRequest généralement une erreur côté client, telle qu'une combinaison de paramètre s non valide, une tentative de mise à jour d'une table inexistante ou une signature de demande incorrecte. UserErrors représente l'ensemble des demandes non valides pour la AWS région actuelle et le AWS compte courant. Unités : `Count` Dimensions : `Operation` Statistiques valides : • `Sum` • `SampleCount`

> **Important**
>
> Certaines statistiques, telles que `Average` ou `Sum`, s'appliquent à chaque métrique. Cependant, toutes ces valeurs sont disponibles via le Timestream pour LiveAnalytics console, ou à l'aide de la CloudWatch console AWS CLI, ou AWS SDKs pour toutes les métriques.

Création d' CloudWatch alarmes pour surveiller Timestream LiveAnalytics

Vous pouvez créer une CloudWatch alarme Amazon pour Timestream LiveAnalytics qui envoie un message Amazon Simple Notification Service SNS (Amazon) lorsque l'alarme change d'état. Une alarme surveille une seule métrique pendant la période que vous spécifiez. Elle réalise une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. L'action est une notification envoyée à une SNS rubrique Amazon ou à une politique Auto Scaling.

Les alarmes déclenchent des actions uniquement pour les changements d'état prolongés. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier. L'état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié.

Pour plus d'informations sur la création d' CloudWatch alarmes, consultez la section Utilisation d'Amazon CloudWatch Alarms dans le guide de CloudWatch l'utilisateur Amazon.

Résolution des problèmes

Cette section contient des informations sur la résolution des problèmes liés à Timestream pour. LiveAnalytics

Rubriques

- Manipulation des WriteRecords accélérateurs
- Gestion des enregistrements rejetés
- Résolution des problèmes UNLOAD depuis Timestream pour LiveAnalytics
- Diffusion chronologique des codes d'erreur LiveAnalytics spécifiques

Manipulation des WriteRecords accélérateurs

Les demandes d'écriture de votre mémoire adressées à Timestream peuvent être limitées à mesure que Timestream évolue pour s'adapter aux besoins d'ingestion de données de votre application. Si vos applications rencontrent des exceptions de limitation, vous devez continuer à envoyer des données au même débit (ou à un débit supérieur) pour permettre à Timestream de s'adapter automatiquement aux besoins de votre application.

Les demandes d'écriture de votre magasin magnétique adressées à Timestream peuvent être limitées si la limite maximale de partitions magnétiques est ingérée. Vous verrez un message d'accélération vous demandant de vérifier la métrique `ActiveMagneticStorePartitions` Cloudwatch pour cette base de données. La résolution de cet accélérateur peut prendre jusqu'à 6 heures. Pour éviter ce ralentissement, vous devez utiliser le stockage de mémoire pour toute charge de travail d'ingestion de débit élevé. Pour l'ingestion de magasins magnétiques, vous pouvez cibler l'ingestion dans un plus petit nombre de partitions en limitant le nombre de séries et la durée pendant lesquelles vous ingérez

Pour plus d'informations sur les meilleures pratiques en matière d'ingestion de données, consultezécrit.

Gestion des enregistrements rejetés

Si Timestream rejette des enregistrements, vous recevrez un document `RejectedRecordsException` contenant les détails du rejet. Reportez-vous à la section Gestion des échecs d'écriture pour plus d'informations sur la façon d'extraire ces informations de la WriteRecords réponse.

Tous les refus seront inclus dans cette réponse, à l'exception des mises à jour du magasin magnétique où la version du nouvel enregistrement est inférieure ou égale à la version de l'enregistrement existant. Dans ce cas, Timestream ne mettra pas à jour l'enregistrement existant contenant la version supérieure. Timestream rejettera le nouvel enregistrement dont la version est inférieure ou égale et enregistrera ces erreurs de manière asynchrone dans votre compartiment S3. Pour recevoir ces rapports d'erreur asynchrones, vous devez définir la `MagneticStoreRejectedDataLocation` propriété dans votre `MagneticStoreWriteProperties` table.

Résolution des problèmes UNLOAD depuis Timestream pour LiveAnalytics

Vous trouverez ci-dessous des instructions pour le dépannage lié à la UNLOAD commande.

Catégorie	Message d'erreur	Comment résoudre les problèmes
Longueur de la clé S3	UNLOADla clé du fichier de résultats lors de l'utilisation du préfixe S3 [%s] fourni dans la destination dépassera la longueur de clé S3 autorisée . Consultez la documentation pour plus de détails.	Lors de l'exportation des résultats de requête à l'aide de l'UNLOADinstruction, la longueur de la clé S3, composée de la somme de la longueur du nom du compartiment S3 et du préfixe, dépasse la longueur de clé S3 maximale prise en charge. Nous vous recommandons de réduire la longueur de votre préfixe ou de votre nom de compartiment.
	UNLOADla clé du fichier de résultats lors de l'utilisation de partitioned_by [%s] dépassera la longueur de clé autorisée dans S3. Consultez la documentation pour plus de détails.	Lorsque vous exportez les résultats d'une requête à l'aide de l'UNLOADinstruction, la longueur de la clé S3 à l'aide de la colonne partitioned_by dépasse la longueur de clé S3 maximale prise en charge. Nous vous recommandons de partitionner avec une autre colonne ou de réduire la longueur de la colonne partitioned_column (si possible).
	UNLOADla clé du fichier de résultats lorsque vous utilisez le préfixe S3 [%s] avec le partitioned_by [%s] dépassera la longueur de clé S3 autorisée. Consultez la	Lors de l'exportation des résultats de requête à l'aide de l'UNLOADinstruction, la longueur de la clé S3, composée de la somme de la longueur du nom

Catégorie	Message d’erreur	Comment résoudre les problèmes
	documentation pour plus de détails.	du compartiment S3, du préfixe et du nom de colonne partitioned_by, dépasse la longueur de clé S3 maximale prise en charge. Nous vous recommandons de réduire le préfixe, la longueur du nom du bucket ou d'utiliser une autre colonne pour partitionner vos données.
	La clé d'objet S3 générée : %s est trop longue. Consultez la documentation pour plus de détails.	Lors du traitement de votre requête à l'aide de l'UNLOADinstruction, l'une des valeurs de la colonne partition née dépasse la longueur de clé S3 maximale prise en charge. La colonne et la valeur de la partition se trouvent dans la clé d'objet générée.

Catégorie	Message d’erreur	Comment résoudre les problèmes
accélérateurs S3	Nous avons détecté qu'Amazon S3 limitait la commande write fromUNLOAD. Consultez la documentation Amazon Timestream pour plus d'informations	Reportez-vous à la documentation S3 ici. Le débit API d'appels S3 peut être limité lorsque plusieurs lecteurs/écrivains accèdent au même dossier. Vérifiez le volume d'appels dans le compartiment fourni. Si vous utilisez le même compartiment pour plusieurs UNLOAD requêtes simultanées, essayez d'utiliser différents compartiments pour les mêmes requêtes. Si vous utilisez le même compartiment pour plusieurs opérations autres que Timestream pour LiveAnalytics UNLOAD, envisagez de déplacer les UNLOAD résultats vers un compartiment distinct.

Diffusion chronologique des codes d'erreur LiveAnalytics spécifiques

Cette section contient les codes d'erreur spécifiques à Timestream pour. LiveAnalytics

Diffusion chronologique des erreurs d'écriture LiveAnalytics API

InternalServerException

- HTTPCode de statut : 500

ThrottlingException

- HTTPCode de statut : 429

ValidationException

HTTPCode de statut : 400

ConflictException

HTTPCode de statut : 409

AccessDeniedException

Vous ne disposez pas d’un accès suffisant pour effectuer cette action.

HTTPCode de statut : 403

ServiceQuotaExceededException

HTTPCode de statut : 402

ResourceNotFoundException

HTTPCode de statut : 404

RejectedRecordsException

HTTPCode de statut : 419

InvalidEndpointException

HTTPCode de statut : 421

Diffusion chronologique des erreurs de requête LiveAnalytics API

ValidationException

HTTPCode de statut : 400

QueryExecutionException

HTTPCode de statut : 400

ConflictException

HTTPCode de statut : 409

ThrottlingException

HTTPCode de statut : 429

InternalServerException

HTTPCode de statut : 500

InvalidEndpointException

HTTPCode de statut : 421

Quotas

Cette rubrique décrit les quotas actuels, également appelés limites, au sein d'Amazon LiveAnalytics Timestream pour. Chaque quota s'applique par région, sauf indication contraire.

Rubriques

Quotas par défaut

Le tableau suivant contient le flux temporel des LiveAnalytics quotas et les valeurs par défaut.

displayName	Description	defaultValue
Bases de données par compte	Le nombre maximum de bases de données que vous pouvez créer par Compte AWS.	500

displayName	Description	defaultValue
Tableaux par compte	Le nombre maximum de tables que vous pouvez créer par Compte AWS.	50000
Taux d'accélérateur pour CRUD APIs	Le nombre maximum de demandes de Create/Update/List/Describe/Delete database/table/scheduled API requêtes autorisées par seconde et par compte, dans la région actuelle.	1
Requêtes planifiées par compte	Le nombre maximum de requêtes planifiées que vous pouvez créer par Compte AWS.	10 000
Nombre maximal de cloisons de stockage magnétiques actives	Le nombre maximal de partitions de stockage magnétiques actives par base de données. Une cloison peut rester active jusqu'à six heures après avoir été ingérée.	250
maxQueryTCU	Le nombre maximum de requêtes TCUs que vous pouvez définir pour votre compte.	1 000

Service Limits

Le tableau suivant contient le flux temporel des limites de LiveAnalytics service et les valeurs par défaut. Pour modifier la conservation des données d'une table depuis la console, voir Modifier une table.

displayName	Description	defaultValue
Période d'ingestion future en minutes	Le délai maximum (en minutes) pour les données de vos séries chronologiques par rapport à l'heure actuelle du système. Par exemple, si la future période d'ingestion est de 15 minutes, Timestream for LiveAnalytics acceptera les données jusqu'à 15 minutes en avance sur l'heure actuelle du système.	15
Durée minimale de conservation de la mémoire stockée en heures	Durée minimale (en heures) pendant laquelle les données doivent être conservées dans la mémoire par table.	1
Durée maximale de conservation de la mémoire stockée en heures	Durée maximale (en heures) pendant laquelle les données peuvent être conservées dans la mémoire par table.	8766
Période de conservation minimale pour le stockage magnétique en jours	Durée minimale (en jours) pendant laquelle les données doivent être conservées dans le magasin magnétique par table.	1
Durée de conservation maximale pour le stockage magnétique en jours	Durée maximale (en jours) pendant laquelle les données peuvent être conservées dans le magasin magnétique. Cette valeur équivaut à 200 ans.	73000

displayName	Description	defaultValue
Période de conservation par défaut pour le stockage magnétique en jours	Valeur par défaut (en jours) pour laquelle les données sont conservées dans le magasin magnétique par table. Cette valeur équivaut à 200 ans.	73000
Durée de conservation par défaut pour la mémoire stockée en heures	Durée par défaut (en heures) pendant laquelle les données sont conservées dans la mémoire.	6
Dimensions par table	Le nombre maximum de dimensions par table.	128
Noms des mesures par table	Le nombre maximum de noms de mesures uniques par table.	8192
Nom de la dimension, valeur de dimension, taille de la paire par série	Taille maximale de la paire nom de dimension et valeur de dimension par série.	2 kilo-octets
Maximum record size (Taille d’enregistrement maximale)	Taille maximale d'un enregistrement.	2 kilo-octets
Enregistrements par WriteRecords API demande	Le nombre maximum d'enregistrements dans une WriteRecords API demande.	100
Longueur du nom de dimension	Le nombre maximal d'octets pour le nom d'une dimension.	60 octets
Longueur du nom de la mesure	Le nombre maximal d'octets pour le nom d'une mesure.	256 bytes
Longueur du nom de la base de données	Le nombre maximal d'octets pour un nom de base de données.	256 bytes

displayName	Description	defaultValue
Longueur du nom du tableau	Le nombre maximal d'octets pour le nom d'une table.	256 bytes
QueryString longueur en KiB	Longueur maximale (en KiB) d'une chaîne de requête en UTF -8 caractères codés pour une requête.	256
Durée d'exécution des requêtes en heures	Durée d'exécution maximale (en heures) d'une requête. Les requêtes qui prennent plus de temps expireront.	1
Informations sur les requêtes	Le nombre maximum de API demandes de requête autorisées avec des informations sur les requêtes activées par seconde et par compte, dans la région actuelle.	1
Taille des métadonnées pour le résultat de la requête	Taille maximale des métadonnées pour le résultat d'une requête.	100 kilo-octets
Taille des données pour le résultat de la requête	Taille maximale des données pour le résultat d'une requête.	5 gigaoctets
Mesures par enregistrement multi-mesures	Le nombre maximum de mesures par enregistrement multi-mesures.	256
Taille de la valeur de mesure par enregistrement multi-mesures	Taille maximale des valeurs de mesure par enregistrement multi-mesures.	2048

displayName	Description	defaultValue
Mesures uniques sur des enregistrements à plusieurs mesures par table	Les mesures uniques de tous les enregistrements multi-mesures définis dans une seule table.	1 024
Unités de calcul Timestream (TCUs) par compte	Le maximum par défaut TCUs par compte.	200

Types de données pris en charge

Le tableau suivant décrit les types de données pris en charge pour les valeurs de mesure et de dimension.

Description	Timestream au service de la valeur LiveAnalytics
Types de données pris en charge pour les valeurs de mesure.	Grand int, double, chaîne, booléen, horodatage MULTI
Types de données pris en charge pour les valeurs de dimension.	Chaîne

Chargement par lots

Les quotas actuels, également appelés limites, pour le chargement par lots sont les suivants.

Description	Timestream au service de la valeur LiveAnalytics
Taille maximale de la tâche de chargement par lots	La taille maximale des tâches de chargement par lots ne peut pas dépasser 100 Go.

Description	Timestream au service de la valeur LiveAnalytics
Quantité de fichiers	Une tâche de chargement par lots ne peut pas comporter plus de 100 fichiers.
Taille maximale du fichier	La taille de fichier maximale d'une tâche de chargement par lots ne peut pas dépasser 5 Go.
CSVtaille de ligne du fichier	Une ligne d'un CSV fichier ne peut pas dépasser 16 Mo. Il s'agit d'une limite stricte qui ne peut pas être augmentée.
Tâches de chargement par lots actives	Une table ne peut pas contenir plus de 5 tâches de chargement par lots actives et un compte ne peut pas avoir plus de 10 tâches de chargement par lots actives. Timestream for LiveAnalytics limitera les nouvelles tâches de chargement par lots jusqu'à ce que davantage de ressources soient disponibles.

Contraintes d’affectation de noms

Le tableau suivant décrit les contraintes de dénomination.

Description	Timestream au service de la valeur LiveAnalytics
Longueur maximale du nom d'une dimension.	60 octets
Longueur maximale du nom d'une mesure.	256 bytes
Longueur maximale d'un nom de table ou d'un nom de base de données.	256 bytes
Nom de table et de base de données	• Nous vous recommandons de ne pas l'utiliserIdentifiants du système. • Peut contenir a-z A-Z 0-9 _ (trait de soulignement) - (tiret). (point).

Description	Timestream au service de la valeur LiveAnalytics
	• Tous les noms doivent être codés en UTF -8, en distinguant majuscules et minuscules. **Note** Les noms de table et de base de données sont comparés à l'aide d'une représentation binaire UTF -8. Cela signifie que la comparaison des ASCII caractères fait la distinction majuscules/minuscules.
Nom de la mesure	• Ne doit pas contenir « : » Identifiants du système ni deux points. • Ne doit pas commencer par un préfixe réservé (`ts_`,`measure_value`). **Note** Les noms de table et de base de données sont comparés à l'aide d'une représentation binaire UTF -8. Cela signifie que la comparaison des ASCII caractères fait la distinction majuscules/minuscules.

Description	Timestream au service de la valeur LiveAnalytics
Nom de la dimension	• Ne doit pas contenir Identifiants du système de deux points « : » ou de guillemets («). • Ne doit pas commencer par un préfixe réservé (`ts_`,`measure_value`). • Ne doit pas contenir les caractères Unicode [0,31] listés ici ou « \ u2028 » ou « \ u2029 ». **Note** Les noms des dimensions et des mesures sont comparés à l'aide d'une représentation binaire UTF -8. Cela signifie que la comparaison des ASCII caractères fait la distinction majuscules/minuscules.
Tous les noms de colonnes	Les noms de colonnes ne peuvent pas être dupliqués. Étant donné que les enregistrements multi-mesures représentent les dimensions et les mesures sous forme de colonnes, le nom d'une dimension ne peut pas être identique à celui d'une mesure. Les noms sont sensibles à la casse.

Mots-clés réservés

Tous les mots suivants sont des mots clés réservés :

- 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

Identifiants du système

Nous réservons les noms de colonne « measure_value », « ts_non_existent_col » et « time » à Timestream pour les identifiants du système. LiveAnalytics De plus, les noms de colonnes ne peuvent pas commencer par « ts_ » ou « measure_name ». Les identificateurs du système distinguent les majuscules et minuscules. Identifiants comparés à l'aide d'une représentation binaire UTF -8. Cela signifie que la comparaison des identifiants fait la distinction majuscules/minuscules.

> **Note**
>
> Les identifiants du système ne peuvent pas être utilisés pour les noms de dimensions ou de mesures. Nous vous recommandons de ne pas utiliser d'identificateurs système pour les noms de base de données ou de tables.

UNLOAD

Pour connaître les limites liées à la `UNLOAD` commande, consultez la section Utilisation UNLOAD pour exporter les résultats d'une requête vers S3 depuis Timestream.

Référence du langage de requête

> **Note**
>
> Cette référence sur le langage de requête inclut la documentation tierce suivante de la Trino Software Foundation (anciennement Presto Software Foundation), qui est sous licence Apache, version 2.0. Vous ne pouvez pas utiliser ce fichier si ce n'est conformément à cette licence. Pour obtenir une copie de la licence Apache, version 2.0, consultez le site Web d'Apache.

Timestream for LiveAnalytics prend en charge un langage de requête riche pour travailler avec vos données. Vous pouvez voir les types de données, les opérateurs, les fonctions et les constructions disponibles ci-dessous.

Vous pouvez également commencer immédiatement avec le langage de requête de Timestream dans la Exemples de requêtes section.

Rubriques

Types de données pris en charge

Le langage de requête LiveAnalytics de Timestream for prend en charge les types de données suivants.

> **Note**
>
> Les types de données pris en charge pour les écritures sont décrits dans la section Types de données.

Type de données	Description
`int`	Représente un entier de 32 bits.
`bigint`	Représente un entier signé de 64 bits.
`boolean`	L'une des deux valeurs de vérité de la logique, `True` et`False`.
`double`	Représente un type de données à précision variable de 64 bits. Implémente la IEEEnorme 754 pour l'arithmétique binaire à virgule flottante. **Note** Le langage de requête est destiné à la lecture des données. Il existe des fonctions `Infinity` et des valeurs `NaN` doubles qui peuvent être utilisées dans les requêtes. Mais vous ne pouvez pas écrire ces valeurs dans Timestream.
`varchar`	Données de caractères de longueur variable avec une taille maximale de 2 Ko.
`array[`*T*`,...]`	Contient un ou plusieurs éléments d'un type de données spécifié *T*Où : *T* peut être n'importe quel type de données pris en charge dans Timestream.
`row(`*T*`,...)`	Contient un ou plusieurs champs nommés de type de données *T*. Les champs peuvent être de n'importe quel type de données pris en charge par Timestream et sont accessibles à l'aide de l'opérateur de référence des champs à points :

Type de données	Description
	`.`
`date`	Représente une date dans le formulaire*YYYY*`-`*MM*`-`*DD*. où *YYYY* c'est l'année, *MM* est le mois, et *DD* est le jour, respectivement. La plage prise en charge est comprise entre `1970-01-01` et`2262-04-11` . Exemple : `1971-02-03`
`time`	Représente l'heure de la journée UTC. Le `time` type de données est représenté sous la forme *HH*`.`*MM*`.`*SS*`.`*sssssssss* `.` Supporte la précision de la nanoseconde. Exemple : `17:02:07.496000000`
`timestamp`	Représente une instance dans le temps en utilisant une précision de nanoseconde. UTC *YYYY*`-`*MM*`-`*DD* *hh*`:`*mm*`:`*ss*`.`*sssssssss* La requête prend en charge les horodatages compris entre`1677-09-21 00:12:44.000000000` . `2262-04-11 23:47:16.854775807`

Type de données	Description

`interval`	Représente un intervalle de temps sous forme de chaîne littérale*X**t*, composé de deux parties, *X* and *t*.
	
	X est une valeur numérique supérieure ou égale à`0`, et *t* est une unité de temps telle que la seconde ou l'heure. L'unité n'est pas pluralisée. L'unité de temps *t* doit être l'un des littéraux de chaîne suivants :
	
	• `nanosecond`
	• `microsecond`
	• `millisecond`
	• `second`
	• `minute`
	• `hour`
	• `day`
	• `ns`(identique à`nanosecond`)
	• `us`(identique à`microsecond`)
	• `ms`(identique à`millisecond`)
	• `s`(identique à`second`)
	• `m`(identique à`minute`)
	• h(identique à`hour`)
	• d(identique à`day`)
	
	Exemples :
	
	`17s`
	
	`12second`
	
	`21hour`

Type de données	Description
	`2d`
`timeseries[row(timestamp, T,...)]`	Représente les valeurs d'une mesure enregistrée sur un intervalle de temps sous `array` forme d'`row`objets. Chacun `row` contient une `timestamp` et une ou plusieurs valeurs de mesure de type de données *T*Où : *T* peut être l'un des suivants : `bigint``boolean`,`double`, ou`varchar`. Les lignes sont triées par ordre croissant par. `timestamp` Le type de données de série chronologique représente les valeurs d'une mesure au fil du temps.
`unknown`	Représente des données nulles.

Fonctionnalité de série chronologique intégrée

Timestream for LiveAnalytics fournit une fonctionnalité de série chronologique intégrée qui traite les données de séries chronologiques comme un concept de premier ordre.

La fonctionnalité de série chronologique intégrée peut être divisée en deux catégories : les vues et les fonctions.

Vous pouvez en savoir plus sur chaque construction ci-dessous.

Rubriques

- Vues des séries chronologiques
- Fonctions de séries chronologiques

Vues des séries chronologiques

Timestream for LiveAnalytics prend en charge les fonctions suivantes pour transformer vos données en type de `timeseries` données :

Rubriques

- CREATE_TIME_SERIES
- UNNEST

CREATE_TIME_SERIES

CREATE_ TIME _ SERIES est une fonction d'agrégation qui prend toutes les mesures brutes d'une série chronologique (valeurs temporelles et valeurs de mesure) et renvoie un type de données de série chronologique. La syntaxe de cette fonction est la suivante :

```
CREATE_TIME_SERIES(time, measure_value::<data_type>)
```

où `<data_type>` est le type de données de la valeur de mesure et peut être bigint, boolean, double ou varchar. Le second paramètre ne peut pas être nul.

Tenez compte de l'CPUutilisation des EC2 instances stockées dans une table nommée metrics, comme indiqué ci-dessous :

Heure	region	az	vpc	instance_id	nom_mesure	valeur_mesure : double
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilisation du processeur	35,0
2019-12-04 19:00:01.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilisation du processeur	38,2
2019-12-04 19:00:02.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilisation du processeur	45,3
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef1	utilisation du processeur	54,1

Heure	region	az	vpc	instance_id	nom_mesure	valeur_mesure : double
2019-12-04 19:00:01 000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef1	utilisation du processeur	42,5
2019-12-04 19:00:02 000 000000	us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef1	utilisation du processeur	33,7

Exécution de la requête :

```
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
```

renverra toutes les séries qui ont `cpu_utilization` comme valeur de mesure. Dans ce cas, nous avons deux séries :

region	az	vpc	instance_id	utilisation du processeur
us-east-1	us-east-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,

region	az	vpc	instance_id	utilisation du processeur
				measure_value : :double : 38.2}, {time : 2019-12-04 19:00:02.000 000000, measure_value : :double : 45,3}]
us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567890abcdef1	[{time : 2019-12-04 19:00:00.000 000000, measure_value : :double : 35.1}, {time : 2019-12-04 19:00:01.000 000000, measure_value : :double : 38.5}, {time : 2019-12-04 19:00:02.000 000000, measure_value : :double : 45,7}]

UNNEST

UNNESTest une fonction de table qui vous permet de transformer `timeseries` les données en modèle plat. La syntaxe est la suivante :

UNNESTtimeseriestransforme a en deux colonnes, à savoir, time etvalue. Vous pouvez également utiliser des alias UNNEST comme indiqué ci-dessous :

```
UNNEST(timeseries) AS <alias_name> (time_alias, value_alias)
```

où <alias_name> est l'alias de la table plate, time_alias l'alias de la time colonne et value_alias l'alias de la value colonne.

Par exemple, imaginez le scénario dans lequel certaines EC2 instances de votre parc sont configurées pour émettre des métriques à un intervalle de 5 secondes, d'autres à un intervalle de 15 secondes, et vous avez besoin des métriques moyennes pour toutes les instances avec une granularité de 10 secondes au cours des 6 dernières heures. Pour obtenir ces données, vous transformez vos indicateurs en modèle de série chronologique à l'aide de CREATE_ TIME _ SERIES. Vous pouvez ensuite utiliser INTERPOLATE_ LINEAR pour obtenir les valeurs manquantes avec une granularité de 10 secondes. Ensuite, vous retransformez les données dans le modèle plat en utilisant UNNEST, puis utilisez AVGpour obtenir les mesures moyennes sur toutes les instances.

```
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
```

La requête ci-dessus illustre l'utilisation de UNNESTavec un alias. Vous trouverez ci-dessous un exemple de la même requête sans utiliser d'alias pour 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
```

Fonctions de séries chronologiques

Amazon Timestream LiveAnalytics for prend en charge les fonctions de séries chronologiques, telles que les dérivées, les intégrales et les corrélations, entre autres, pour obtenir des informations plus approfondies à partir de vos données de séries chronologiques. Cette section fournit des informations d'utilisation pour chacune de ces fonctions, ainsi que des exemples de requêtes. Sélectionnez un sujet ci-dessous pour en savoir plus.

Rubriques

Fonctions d'interpolation

Si les données de votre série chronologique ne contiennent pas de valeurs pour des événements à certains moments, vous pouvez estimer les valeurs de ces événements manquants à l'aide d'une interpolation. Amazon Timestream prend en charge quatre variantes d'interpolation : l'interpolation linéaire, l'interpolation par spline cubique, l'interpolation de la dernière observation reportée (locf) et l'interpolation constante. Cette section fournit des informations d'utilisation du Timestream pour les fonctions LiveAnalytics d'interpolation, ainsi que des exemples de requêtes.

Informations d'utilisation

Fonction	Type de données de sortie	Description
`interpolate_linear (timeseries, array[timestamp])`	séries chronologiques	Complète les données manquantes à l'aide d'une interpolation linéaire.
`interpolate_linear (timeseries, timestamp)`	double	Complète les données manquantes à l'aide d'une interpolation linéaire.
`interpolate_spline _cubic(timeseries, array[timestamp])`	séries chronologiques	Complète les données manquantes à l'aide d'une interpolation par splines cubiques.
`interpolate_spline _cubic(timeseries, timestamp)`	double	Complète les données manquantes à l'aide d'une interpolation par splines cubiques.
`interpolate_locf(t imeseries, array[tim estamp])`	séries chronologiques	Complète les données manquantes en utilisant la dernière valeur échantillonnée.
`interpolate_locf(t imeseries, timestamp )`	double	Complète les données manquantes en utilisant la dernière valeur échantillonnée.
`interpolate_fill(t imeseries, array[tim estamp], double)`	séries chronologiques	Complète les données manquantes à l'aide d'une valeur constante.
`interpolate_fill(t imeseries, timestamp , double)`	double	Complète les données manquantes à l'aide d'une valeur constante.

Exemples de requêtes

Example

Déterminez l'CPUutilisation moyenne regroupée à intervalles de 30 secondes pour un EC2 hôte spécifique au cours des 2 dernières heures, en remplissant les valeurs manquantes à l'aide d'une interpolation linéaire :

```
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

Trouvez l'CPUutilisation moyenne regroupée à intervalles de 30 secondes pour un EC2 hôte spécifique au cours des 2 dernières heures, en remplissant les valeurs manquantes à l'aide d'une interpolation basée sur la dernière observation reportée :

```
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)
```

Fonctions dérivées

Les dérivés sont utilisés pour calculer le taux de variation d'une métrique donnée et peuvent être utilisés pour répondre de manière proactive à un événement. Supposons, par exemple, que vous calculiez la dérivée de CPU l'utilisation des EC2 instances au cours des 5 dernières minutes et que vous remarquiez une dérivée positive significative. Cela peut indiquer une augmentation de la charge de travail. Vous pouvez donc décider de créer davantage d'EC2instances pour mieux gérer votre charge de travail.

Amazon Timestream prend en charge deux variantes de fonctions dérivées. Cette section fournit des informations d'utilisation du Timestream pour les fonctions LiveAnalytics dérivées, ainsi que des exemples de requêtes.

Informations d'utilisation

Fonction	Type de données de sortie	Description
`derivative_linear( timeseries, interval)`	séries chronologiques	Calcule la dérivée de chaque point du `timeseries` pour le spécifié`interval`.
`non_negative_deriv ative_linear(times eries, interval)`	séries chronologiques	Identique`derivativ e_linear(timeserie s, interval)` , mais ne renvoie que des valeurs positives.

Exemples de requêtes

Example

Trouvez le taux de variation de l'CPUutilisation toutes les 5 minutes au cours de la dernière heure :

```
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

Calculez le taux d'augmentation des erreurs générées par un ou plusieurs microservices :

```
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
```

Fonctions intégrales

Vous pouvez utiliser des intégrales pour déterminer l'aire sous la courbe par unité de temps pour les événements de votre série chronologique. Supposons par exemple que vous suiviez le volume de demandes reçues par votre application par unité de temps. Dans ce scénario, vous pouvez utiliser la fonction intégrale pour déterminer le volume total de demandes traitées par intervalle spécifié sur une période donnée.

Amazon Timestream prend en charge une variante de fonctions intégrales. Cette section fournit des informations d'utilisation du Timestream pour la fonction LiveAnalytics intégrale, ainsi que des exemples de requêtes.

Informations d'utilisation

Fonction	Type de données de sortie	Description

`integral_trapezoidal(timeseries(double))`
 	
`integral_trapezoidal(timeseries(double), interval day to second)`
 	
`integral_trapezoidal(timeseries(bigint))`
 	
`integral_trapezoidal(timeseries(bigint), interval day to second)`
 	
`integral_trapezoidal(timeseries(integer), interval day to second)`
 	
`integral_trapezoidal(timeseries(integer))`
	double	Correspond approximativement à l'intégrale spécifiée `interval day to second` pour la valeur `timeseries` fournie, en utilisant la règle du trapèze. Le paramètre d'intervalle entre le jour et la seconde est facultatif et la valeur par défaut est`1s`. Pour plus d'informations sur les intervalles, consultezIntervalle et durée.

Exemples de requêtes

Example

Calculez le volume total de demandes traitées toutes les cinq minutes au cours de la dernière heure par un hôte spécifique :

```
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
```

Fonctions de corrélation

Pour deux séries chronologiques de longueur similaire, les fonctions de corrélation fournissent un coefficient de corrélation, qui explique l'évolution des deux séries chronologiques dans le temps. Le coefficient de corrélation est compris entre `-1.0` et`1.0`. `-1.0`indique que les deux séries chronologiques évoluent dans des directions opposées au même rythme, alors qu'`1.0`indique que les deux séries chronologiques évoluent dans la même direction au même rythme. La valeur de `0` indique l'absence de corrélation entre les deux séries chronologiques. Par exemple, si le prix du pétrole augmente et que le cours de l'action d'une société pétrolière augmente, la tendance de la hausse du prix du pétrole et celle de la société pétrolière auront un coefficient de corrélation positif. Un coefficient de corrélation positif élevé indiquerait que les deux prix évoluent au même rythme. De même, le coefficient de corrélation entre les cours des obligations et les rendements obligataires est négatif, ce qui indique que ces deux valeurs évoluent dans le sens inverse au fil du temps.

Amazon Timestream prend en charge deux variantes de fonctions de corrélation. Cette section fournit des informations d'utilisation du Timestream pour les fonctions de LiveAnalytics corrélation, ainsi que des exemples de requêtes.

Informations d'utilisation

Fonction	Type de données de sortie	Description
`correlate_pearson( timeseries, timeseries)`	double	Calcule le coefficient de corrélation de Pearson pour les deux. `timeseries` Les séries temporelles doivent avoir les mêmes horodatages.
`correlate_spearman (timeseries, timeseries)`	double	Calcule le coefficient de corrélation de Spearman pour les deux. `timeseries` Les

Fonction	Type de données de sortie	Description
		séries temporelles doivent avoir les mêmes horodatages.

Exemples de requêtes

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
```

Fonctions de filtrage et de réduction

Amazon Timestream prend en charge des fonctions permettant de filtrer et de réduire les opérations sur les données de séries chronologiques. Cette section fournit des informations d'utilisation du Timestream pour les fonctions de LiveAnalytics filtrage et de réduction, ainsi que des exemples de requêtes.

Informations d'utilisation

Fonction	Type de données de sortie	Description
`filter(timeseries(T), function(T, Boolean))`	série chronologique (T)	Construit une série chronologique à partir d'une série chronologique en entrée, en utilisant les valeurs pour lesquelles le passé `function` renvoie`true`.
`reduce(timeseries(T), initialState S, inputFunction(S, T, S), outputFunction(S, R))`	R	Renvoie une valeur unique, réduite par rapport à la série chronologique. Le `inputFunction` sera invoqué sur chaque élément des séries temporelles dans l'ordre. En plus de prendre l'élément actuel, inputFunction prend l'état actuel (initialement`initialState`) et renvoie le nouvel état. Le `outputFunction` sera invoqué pour transformer l'état final en valeur de résultat. Il `outputFunction` peut s'agir d'une fonction d'identité.

Exemples de requêtes

Example

Construisez une série chronologique d'CPUutilisation d'un hôte et de points de filtrage avec une mesure supérieure à 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

Construisez une série chronologique d'CPUutilisation d'un hôte et déterminez la somme au carré des mesures :

```
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

Construisez une série CPU chronologique d'utilisation d'un hôte et déterminez la fraction d'échantillons supérieurs au CPU seuil :

```
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
```

SQLsoutien

Timestream for LiveAnalytics prend en charge certaines constructions courantesSQL. Vous pouvez en lire plus ci-dessous.

Rubriques

- SELECT
- Support des sous-requêtes
- SHOWdéclarations
- DESCRIBEdéclarations
- UNLOAD

SELECT

SELECTles instructions peuvent être utilisées pour récupérer des données d'une ou de plusieurs tables. Le langage de requête de Timestream prend en charge la syntaxe suivante pour SELECTles instructions :

```
[ 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 ] ]
```

où

- `function (expression)`est l'une des fonctions de fenêtre prises en charge.
- `partition_expr_list`c'est :

```
expression | column_name [, expr_list ]
```

- `order_list`c'est :

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

- `frame_clause`c'est :

```
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`est l'un des suivants :

```
table_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
from_item join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]
```

- `join_type`est l'un des suivants :

```
[ INNER ] JOIN
LEFT [ OUTER ] JOIN
RIGHT [ OUTER ] JOIN
```

```
FULL [ OUTER ] JOIN
```

- `grouping_element`est l'un des suivants :

```
()
expression
```

Support des sous-requêtes

Timestream prend en charge les sous-requêtes et les prédicats. `EXISTS` `IN` Le `EXISTS` prédicat détermine si une sous-requête renvoie des lignes. Le `IN` prédicat détermine si les valeurs produites par la sous-requête correspondent aux valeurs ou à l'expression de la clause IN. Le langage de requête Timestream prend en charge les sous-requêtes corrélées et autres.

```
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

SHOWdéclarations

Vous pouvez consulter toutes les bases de données d'un compte à l'aide du `SHOW DATABASES` relevé. La syntaxe est la suivante :

```
SHOW DATABASES [LIKE pattern]
```

où la `LIKE` clause peut être utilisée pour filtrer les noms de base de données.

Vous pouvez consulter tous les tableaux d'un compte à l'aide du `SHOW TABLES` relevé. La syntaxe est la suivante :

```
SHOW TABLES [FROM database] [LIKE pattern]
```

où la `FROM` clause peut être utilisée pour filtrer les noms de base de données et la `LIKE` clause pour filtrer les noms de tables.

Vous pouvez afficher toutes les mesures d'un tableau à l'aide de l'`SHOW MEASURES`instruction. La syntaxe est la suivante :

```
SHOW MEASURES FROM database.table [LIKE pattern]
```

où la `FROM` clause sera utilisée pour spécifier le nom de la base de données et de la table et où la `LIKE` clause peut être utilisée pour filtrer les noms des mesures.

DESCRIBEdéclarations

Vous pouvez afficher les métadonnées d'une table à l'aide de l'`DESCRIBE`instruction. La syntaxe est la suivante :

```
DESCRIBE database.table
```

où `table` contient le nom de la table. L'instruction describe renvoie les noms des colonnes et les types de données de la table.

UNLOAD

Timestream for LiveAnalytics prend en charge une `UNLOAD` commande en tant qu'extension de son SQL support. Les types de données pris en charge par `UNLOAD` sont décrits dansTypes de données pris en charge. Les `unknown` types `time` et ne s'appliquent pas à`UNLOAD`.

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

où se trouve l'option

```
{ 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>'
}
```

SELECTdéclaration

L'instruction de requête utilisée pour sélectionner et récupérer les données d'un ou de plusieurs Timestream pour les LiveAnalytics tables.

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

Clause TO

```
TO 's3://bucket-name/folder'
```

or

```
TO 's3://access-point-alias/folder'
```

La T0 clause contenue dans l'UNL0ADinstruction indique la destination de sortie des résultats de la requête. Vous devez fournir le chemin complet, y compris le nom du compartiment Amazon S3 ou Amazon S3 access-point-alias avec l'emplacement du dossier sur Amazon S3 où Timestream for LiveAnalytics écrit les objets du fichier de sortie. Le compartiment S3 doit appartenir au même compte et se trouver dans la même région. Outre le jeu de résultats de la requête, Timestream for LiveAnalytics écrit le manifeste et les fichiers de métadonnées dans le dossier de destination spécifié.

PARTITIONEDClause _BY

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

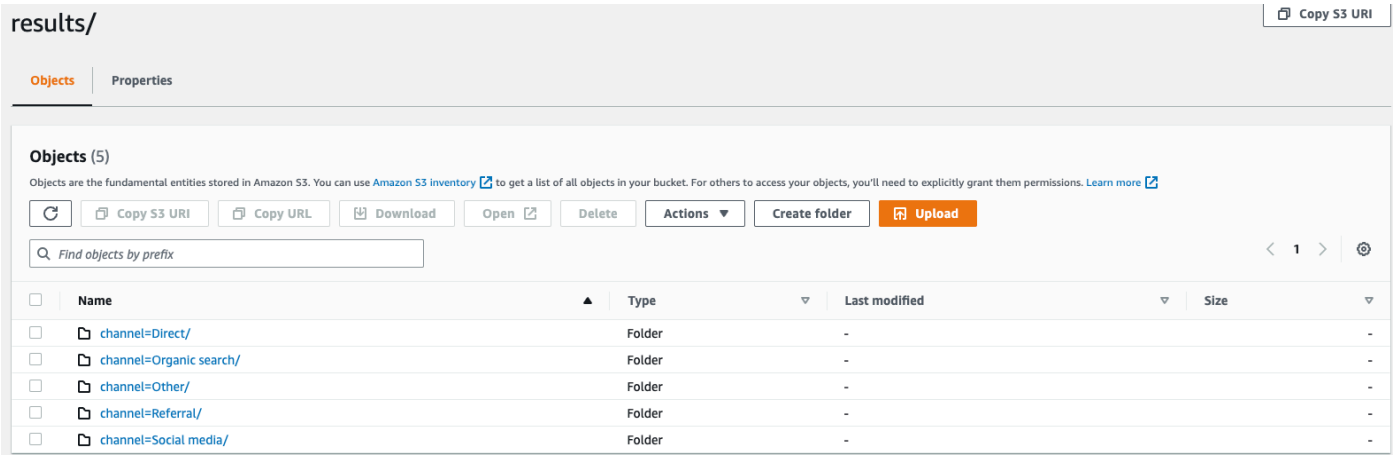
La `partitioned_by` clause est utilisée dans les requêtes pour regrouper et analyser les données à un niveau granulaire. Lorsque vous exportez les résultats de votre requête vers le compartiment S3, vous pouvez choisir de partitionner les données en fonction d'une ou de plusieurs colonnes de la requête de sélection. Lors du partitionnement des données, les données exportées sont divisées en sous-ensembles en fonction de la colonne de partition et chaque sous-ensemble est stocké dans un dossier distinct. Dans le dossier de résultats qui contient vos données exportées, un sous-dossier `folder/results/partition column = partition value/` est automatiquement créé. Notez toutefois que les colonnes partitionnées ne sont pas incluses dans le fichier de sortie.

`partitioned_by`n'est pas une clause obligatoire dans la syntaxe. Si vous choisissez d'exporter les données sans partitionnement, vous pouvez exclure la clause dans la syntaxe.

Example

En supposant que vous surveillez les données de navigation de votre site Web et que vous disposiez de 5 canaux de traficdirect, à savoir `Social MediaOrganic Search`,`0ther`, et`Referral`. Lorsque vous exportez les données, vous pouvez choisir de les partitionner à l'aide de la colonne`Channel`. Dans votre dossier de données`s3://bucketname/results`, vous aurez cinq dossiers portant chacun leur nom de chaîne respectif. Par exemple, `s3://bucketname/results/channel=Social Media/.` dans ce dossier, vous trouverez les données de tous les clients qui ont accédé à votre site Web via le `Social Media` canal. De même, vous aurez d'autres dossiers pour les chaînes restantes.

Données exportées partitionnées par colonne de canal



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

Les mots clés permettant de spécifier le format des résultats de requête écrits dans votre compartiment S3. Vous pouvez exporter les données soit sous forme de valeur séparée par des virgules (CSV) en utilisant une virgule (,) comme séparateur par défaut, soit au format Apache Parquet, un format de stockage en colonnes ouvert efficace pour les analyses.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Vous pouvez compresser les données exportées à l'aide d'un algorithme de compression GZIP ou les décompresser en spécifiant l'NONEoption.

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

Les fichiers de sortie sur Amazon S3 sont chiffrés à l'aide de l'option de chiffrement que vous avez sélectionnée. Outre vos données, le manifeste et les fichiers de métadonnées sont également chiffrés en fonction de l'option de chiffrement que vous avez sélectionnée. Nous prenons actuellement en charge le KMS chiffrement SSE _S3 et SSE _. SSE_S3 est un chiffrement côté serveur, Amazon S3 chiffrant les données à l'aide d'un cryptage standard de chiffrement avancé () 256 bits. AES SSE_ KMS est un chiffrement côté serveur qui permet de chiffrer les données à l'aide de clés gérées par le client.

KMS_KEY

```
kms_key = '<string>'
```

KMSLa clé est une clé définie par le client pour chiffrer les résultats de requête exportés. KMSLa clé est gérée de manière sécurisée par le service de gestion des AWS clés (AWS KMS) et utilisée pour chiffrer les fichiers de données sur Amazon S3.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Lors de l'exportation des données au CSV format, ce champ spécifie un seul ASCII caractère utilisé pour séparer les champs du fichier de sortie, tel qu'un tube (|), une virgule (,) ou un onglet (/t). Le séparateur par défaut pour les CSV fichiers est une virgule. Si une valeur de vos données contient le délimiteur choisi, celui-ci sera mis entre guillemets. Par exemple, si la valeur de vos données contient`Time,stream`, elle sera indiquée entre guillemets comme `"Time,stream"` dans les données exportées. Les guillemets utilisés par Timestream LiveAnalytics sont des guillemets doubles («).

Évitez de spécifier le caractère de renvoi (ASCII13, hexadécimal`0D`, texte « \ r ») ou le caractère de saut de ligne (ASCII10, hexadécimal 0A, texte «\n») comme tel `FIELD_DELIMITER` si vous souhaitez inclure des en-têtes dans leCSV, car cela empêcherait de nombreux analyseurs de pouvoir analyser correctement les en-têtes dans la sortie résultante. CSV

ESCAPED_PAR

```
escaped_by = '<character>', default: (\)
```

Lors de l'exportation des données au CSV format, ce champ indique le caractère qui doit être traité comme un caractère d'échappement dans le fichier de données écrit dans le compartiment S3. L'évasion se produit dans les scénarios suivants :

1. Si la valeur elle-même contient le caractère guillemet («), elle sera échappée à l'aide d'un caractère d'échappement. Par exemple, si la valeur est`Time"stream`, où (\) est le caractère d'échappement configuré, il sera échappé en tant que`Time\"stream`.
2. Si la valeur contient le caractère d'échappement configuré, il sera échappé. Par exemple, si la valeur est`Time\stream`, elle sera ignorée en tant que`Time\\stream`.

> **Note**
>
> Si la sortie exportée contient des types de données complexes tels que des tableaux, des lignes ou des séries temporelles, elle sera sérialisée sous forme de chaîne. JSON Voici un exemple.

Type de données	Valeur réelle	Comment la valeur est échappée au CSV format [JSONchaîne sérialisée]
Tableau	`[ 23,24,25 ]`	`"[23,24,25]"`
Rangée	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\":\"hello\"}"`
Chronologique	`[ ( 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'
```

Lorsque vous exportez les données au CSV format, ce champ vous permet d'inclure les noms de colonnes dans la première ligne des fichiers de CSV données exportés.

Les valeurs acceptées sont « vrai » et « faux » et la valeur par défaut est « faux ». Les options de transformation de texte telles que `escaped_by` et `field_delimiter` s'appliquent également aux en-têtes.

> **Note**
>
> Lorsque vous incluez des en-têtes, il est important de ne pas sélectionner un caractère de renvoi (ASCII13, hexadécimal 0D, texte « \ r ») ou un caractère de saut de ligne (ASCII10, hexadécimal 0A, texte «\n») comme caractère`FIELD_DELIMITER`, car cela empêcherait de nombreux analyseurs de pouvoir analyser correctement les en-têtes dans le résultat obtenu. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Ce champ indique la taille maximale des fichiers créés par l'`UNLOAD`instruction dans Amazon S3. L'`UNLOAD`instruction peut créer plusieurs fichiers, mais la taille maximale de chaque fichier écrit sur Amazon S3 sera approximativement celle spécifiée dans ce champ.

La valeur du champ doit être comprise entre 16 Mo et 78 Go inclus. Vous pouvez le spécifier en nombre entier tel que`12GB`, ou en décimaux tels que `0.5GB` ou. `24.7MB` La valeur par défaut est de 78 Go.

La taille réelle du fichier est approximative au moment de l'écriture du fichier, de sorte que la taille maximale réelle peut ne pas être exactement égale au nombre que vous spécifiez.

Opérateurs logiques

Timestream for LiveAnalytics prend en charge les opérateurs logiques suivants.

Opérateur	Description	Exemple
AND	Vrai si les deux valeurs sont vraies	un AND b
OU	Vrai si l'une des valeurs est vraie	a OU b
NOT	Vrai si la valeur est fausse	NOTun

- Le résultat d'une AND comparaison peut être NULL si l'un ou les deux côtés de l'expression le sontNULL.
- Si au moins un côté d'un AND opérateur est, FALSE l'expression est évaluée àFALSE.
- Le résultat d'une OR comparaison peut être NULL si l'un ou les deux côtés de l'expression le sontNULL.
- Si au moins un côté d'un OR opérateur est, TRUE l'expression est évaluée àTRUE.
- NULLC'est le complément logiqueNULL.

Le tableau de vérité suivant illustre le traitement de NULL in AND et OR :

A	B	A et b	A ou b
null	null	null	null
false	null	false	null
null	false	false	null
true	null	null	true
null	true	null	vrai
false	false	false	false
vrai	false	false	vrai
false	vrai	false	true
true	true	true	true

Le tableau de vérité suivant illustre le traitement de NULL in NOT :

A	Ce n'est pas un
null	null
vrai	false

A	Ce n'est pas un
false	true

Opérateurs de comparaison

Timestream for LiveAnalytics prend en charge les opérateurs de comparaison suivants.

Opérateur	Description
<	Inférieur à
>	Supérieure à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
=	Égal à
<>	Non égal à
!=	Non égal à

Note

- L'`BETWEEN`opérateur vérifie si une valeur se situe dans une plage spécifiée. La syntaxe est la suivante :

  ```
  BETWEEN min AND max
  ```

  La présence de la `NOT BETWEEN` mention `NULL` in a `BETWEEN` or fera en sorte que l'instruction soit évaluée à`NULL`.
- `IS NULL` et `IS NOT NULL` les opérateurs vérifient si une valeur est nulle (non définie). L'utilisation `NULL` de with `IS NULL` donne la valeur true.
- DansSQL, une `NULL` valeur signifie une valeur inconnue.

Fonctions de comparaison

Timestream for LiveAnalytics prend en charge les fonctions de comparaison suivantes.

Rubriques

le plus grand ()

La fonction greatest () renvoie la plus grande des valeurs fournies. Elle renvoie `NULL` si l'une des valeurs fournies l'est`NULL`. La syntaxe est la suivante.

```
greatest(value1, value2, ..., valueN)
```

le moins ()

La fonction least () renvoie la plus petite des valeurs fournies. Elle renvoie `NULL` si l'une des valeurs fournies l'est`NULL`. La syntaxe est la suivante.

```
least(value1, value2, ..., valueN)
```

ALL(), ANY () et SOME ()

Les `ALL` `SOME` quantificateurs `ANY` et peuvent être utilisés conjointement avec les opérateurs de comparaison de la manière suivante.

Expression	Signification
A = ALL (...)	Est considéré comme vrai lorsque A est égal à toutes les valeurs.
UN <> ALL (...)	Est considéré comme vrai lorsque A ne correspond à aucune valeur.
UN < ALL (...)	Est considéré comme vrai lorsque A est inférieur à la plus petite valeur.

Expression	Signification
A = ANY (...)	Est considéré comme vrai lorsque A est égal à l'une des valeurs.
UN <> ANY (...)	Est considéré comme vrai lorsque A ne correspond pas à une ou plusieurs valeurs.
UN < ANY (...)	Est considéré comme vrai lorsque A est inférieur à la valeur la plus élevée.

Exemples et notes d'utilisation

> **Note**
>
> Lorsque vous utilisez `ANY` ou `ALLSOME`, le mot clé `VALUES` doit être utilisé si les valeurs de comparaison sont une liste de littéraux.

Exemple : **`ANY()`**

Voici un exemple de requête `ANY()` dans une instruction de requête.

```
SELECT 11.7 = ANY (VALUES 12.0, 13.5, 11.7)
```

Une syntaxe alternative pour la même opération est la suivante.

```
SELECT 11.7 = ANY (SELECT 12.0 UNION ALL SELECT 13.5 UNION ALL SELECT 11.7)
```

Dans ce cas, est `ANY()` évalué à. `True`

Exemple : **`ALL()`**

Voici un exemple de requête `ALL()` dans une instruction de requête.

```
SELECT 17 < ALL (VALUES 19, 20, 15);
```

Une syntaxe alternative pour la même opération est la suivante.

```
SELECT 17 < ALL (SELECT 19 UNION ALL SELECT 20 UNION ALL SELECT 15);
```

Dans ce cas, est `ALL()` évalué à. `False`

Exemple : **`SOME()`**

Voici un exemple de requête `SOME()` dans une instruction de requête.

```
SELECT 50 >= SOME (VALUES 53, 77, 27);
```

Une syntaxe alternative pour la même opération est la suivante.

```
SELECT 50 >= SOME (SELECT 53 UNION ALL SELECT 77 UNION ALL SELECT 27);
```

Dans ce cas, est `SOME()` évalué à. `True`

Expressions conditionnelles

Timestream for LiveAnalytics prend en charge les expressions conditionnelles suivantes.

Rubriques

- La CASE déclaration
- La déclaration IF
- La COALESCE déclaration
- La NULLIF déclaration
- La TRY déclaration

La CASE déclaration

L'CASEinstruction recherche chaque expression de valeur de gauche à droite jusqu'à ce qu'elle en trouve une égale`expression`. S'il trouve une correspondance, le résultat correspondant à la valeur correspondante est renvoyé. Si aucune correspondance n'est trouvée, le résultat de la `ELSE` clause est renvoyé s'il existe ; dans le cas contraire, il `null` est renvoyé. La syntaxe est la suivante :

```
CASE expression
    WHEN value THEN result
```

```
    [ WHEN ... ]
    [ ELSE result ]
END
```

Timestream prend également en charge la syntaxe suivante pour CASEles instructions. Dans cette syntaxe, le formulaire « recherché » évalue chaque condition booléenne de gauche à droite jusqu'à ce qu'une condition le soit `true` et renvoie le résultat correspondant. Si aucune condition ne l'est`true`, le résultat de la `ELSE` clause est renvoyé s'il existe ; dans le cas contraire, il `null` est renvoyé. Voir ci-dessous pour la syntaxe alternative :

```
CASE
    WHEN condition THEN result
    [ WHEN ... ]
    [ ELSE result ]
END
```

La déclaration IF

L'instruction IF évalue une condition comme vraie ou fausse et renvoie la valeur appropriée. Timestream prend en charge les deux représentations syntaxiques suivantes pour IF :

```
if(condition, true_value)
```

Cette syntaxe évalue et renvoie `true_value` si la condition est `true` ; dans le cas contraire, elle `null` est renvoyée et n'`true_value`est pas évaluée.

```
if(condition, true_value, false_value)
```

Cette syntaxe évalue et renvoie `true_value` si la condition est`true`, sinon évalue et renvoie. `false_value`

Exemples

```
SELECT
  if(true, 'example 1'),
  if(false, 'example 2'),
  if(true, 'example 3 true', 'example 3 false'),
  if(false, 'example 4 true', 'example 4 false')
```

_col0	_col1	_col2	_col3
example 1	- null	example 3 true	example 4 false

La COALESCE déclaration

COALESCErenvoie la première valeur non nulle d'une liste d'arguments. La syntaxe est la suivante :

```
coalesce(value1, value2[,...])
```

La NULLIF déclaration

L'instruction IF évalue une condition comme vraie ou fausse et renvoie la valeur appropriée. Timestream prend en charge les deux représentations syntaxiques suivantes pour IF :

NULLIFrenvoie null s'il `value1` est égal `value2` ; dans le cas contraire, il renvoie la valeur`value1`. La syntaxe est la suivante :

```
nullif(value1, value2)
```

La TRY déclaration

La TRYfonction évalue une expression et gère certains types d'erreurs en `null` renvoyant. La syntaxe est la suivante :

```
try(expression)
```

Fonctions de conversion

Timestream for LiveAnalytics prend en charge les fonctions de conversion suivantes.

Rubriques

- cast()
- try_cast ()

cast()

La syntaxe de la fonction cast permettant de convertir explicitement une valeur en type est la suivante.

```
cast(value AS type)
```

try_cast ()

Timestream for prend LiveAnalytics également en charge la fonction try_cast qui est similaire à cast mais renvoie null en cas d'échec du cast. La syntaxe est la suivante.

```
try_cast(value AS type)
```

Operateurs mathématiques

Timestream for LiveAnalytics prend en charge les opérateurs mathématiques suivants.

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division (la division entière effectue la troncature)
%	Module (reste)

Fonctions mathématiques

Timestream for LiveAnalytics prend en charge les fonctions mathématiques suivantes.

Fonction	Type de données de sortie	Description
abdos (x)	[identique à l'entrée]	Renvoie la valeur absolue de x.
cbrt (x)	double	Renvoie la racine cubique de x.
plafond (x) ou plafond (x)	[identique à l'entrée]	Renvoie x arrondi au nombre entier le plus proche.
degrés (x)	double	Convertit l'angle x en radians en degrés.
e ()	double	Renvoie le nombre d'Euler constant.
exp (x)	double	Renvoie le nombre d'Euler élevé à la puissance x.
étage (x)	[identique à l'entrée]	Renvoie x arrondi à l'entier inférieur le plus proche.
from_base (chaîne, base)	bigint	Renvoie la valeur d'une chaîne interprétée comme un nombre de base de base.
ln (x)	double	Renvoie le logarithme naturel de x.
journal 2 (x)	double	Renvoie le logarithme en base 2 de x.
journal 10 (x)	double	Renvoie le logarithme en base 10 de x.
mode (n, m)	[identique à l'entrée]	Renvoie le module (reste) de n divisé par m.

Fonction	Type de données de sortie	Description
épi ()	double	Renvoie la constante Pi.
pow (x, p) ou puissance (x, p)	double	Renvoie x élevé à la puissance p.
radians (x)	double	Convertit l'angle x en degrés en radians.
rand () ou random ()	double	Renvoie une valeur pseudo-al éatoire comprise entre 0,0 et 1,0.
aléatoire (n)	[identique à l'entrée]	Renvoie un nombre pseudo-aléatoire compris entre 0 et n (exclusif).
rond (x)	[identique à l'entrée]	Renvoie x arrondi à l'entier le plus proche.
rond (x, d)	[identique à l'entrée]	Renvoie x arrondi à d décimales.

Fonction	Type de données de sortie	Description
signe (x)	[identique à l'entrée]	Renvoie la fonction signum de x, c'est-à-dire : • 0 si l'argument est 0 • 1 si l'argument est supérieur à 0 • -1 si l'argument est inférieur à 0. Pour les arguments doubles, la fonction renvoie également : • NaN si l'argument est NaN • 1 si l'argument est +Infinity • -1 si l'argument est -Infinity.
carré (x)	double	Renvoie la racine carrée de x.
to_base (x, radix)	varchar	Renvoie la représentation en base de x.
tronquer (x)	double	Renvoie x arrondi à un entier en supprimant les chiffres après la virgule décimale.
macos (x)	double	Renvoie l'arc cosinus de x.
ASIN (x)	double	Renvoie l'arc sinusoïdal de x.
Satan (x)	double	Renvoie la tangente de l'arc de x.
atan2 (y, x)	double	Renvoie la tangente de l'arc de y/ x.
cos (x)	double	Renvoie le cosinus de x.

Fonction	Type de données de sortie	Description
douillet (x)	double	Renvoie le cosinus hyperboli que de x.
péché (x)	double	Renvoie le sinus de x.
beige (x)	double	Renvoie la tangente de x.
bronzage (x)	double	Renvoie la tangente hyperboli que de x.
infini ()	double	Renvoie la constante représentant l'infini positif.
is_fini (x)	boolean	Déterminez si x est fini.
is_infini (x)	boolean	Déterminez si x est infini.
is_nan (x)	boolean	Déterminez si x est not-a-num ber.
homme (1)	double	Renvoie la constante représentant not-a-number.

Opérateurs de chaîne

Timestream for LiveAnalytics permet à l'| |opérateur de concaténer une ou plusieurs chaînes.

Fonctions de chaîne

> **Note**
>
> Le type de données d'entrée de ces fonctions est supposé être varchar, sauf indication contraire.

Fonction	Type de données de sortie	Description
chr (n)	varchar	Renvoie le point de code Unicode n sous forme de varchar.
point de code (x)	entier	Renvoie le point de code Unicode du seul caractère de str.
concat (x1,..., xN)	varchar	Renvoie la concaténation de x1, x2,..., xN.
hamming_distance (x1, x2)	bigint	Renvoie la distance de Hamming de x1 et x2, c'est-à-dire le nombre de positions auxquelles les caractères correspondants sont différents. Notez que les deux entrées varchar doivent avoir la même longueur.
longueur (x)	bigint	Renvoie la longueur de x en caractères.
levenshtein_distance (x1, x2)	bigint	Renvoie la distance d'édition Levenshtein de x1 et x2, c'est-à-dire le nombre minimum de modifications d'un seul caractère (insertions, suppressions ou substitutions) nécessaires pour transformer x1 en x2.
inférieur (x)	varchar	Convertit x en minuscules.
charge (x1, taille bigint, x2)	varchar	Le bloc-notes gauche est x1 pour dimensionner les

Fonction	Type de données de sortie	Description
		caractères avec x2. Si la taille est inférieure à la longueur de x1, le résultat est tronqué en caractères de taille. La taille ne doit pas être négative et x2 ne doit pas être vide.
Ultrim (x)	varchar	Supprime les espaces blancs principaux de x.
remplacer (x1, x2)	varchar	Supprime toutes les instances de x2 de x1.
remplacer (x1, x2, x3)	varchar	Remplace toutes les instances de x2 par x3 dans x1.
Inverser (x)	varchar	Renvoie x avec les caractères dans l'ordre inverse.
route (x1, taille bigint, x2)	varchar	Le clavier droit contient x1 pour dimensionner les caractères avec x2. Si la taille est inférieure à la longueur de x1, le résultat est tronqué en caractères de taille. La taille ne doit pas être négative et x2 ne doit pas être vide.
garniture (x)	varchar	Supprime les espaces blancs de fin de x.
scindé (x1, x2)	array(varchar)	Divise x1 sur le délimiteur x2 et renvoie un tableau.

Fonction	Type de données de sortie	Description
split (x1, x2, limite bigint)	array(varchar)	Divise x1 sur le délimiteur x2 et renvoie un tableau. Le dernier élément du tableau contient toujours tout ce qui reste dans la limite x1. Ce doit être un nombre positif.
split_part (x1, x2, bigint pos)	varchar	Divise x1 sur le délimiteur x2 et renvoie le champ varchar à pos. Les index de champs commencent par 1. Si pos est supérieur au nombre de champs, la valeur null est renvoyée.
sangles (x1, x2)	bigint	Renvoie la position de départ de la première instance de x2 dans x1. Les positions commencent par 1. S'il n'est pas trouvé, 0 est renvoyé.
strpos (x1, x2, instance de bigint)	bigint	Renvoie la position de la nième instance de x2 dans x1. L'instance doit être un nombre positif. Les positions commencent par 1. S'il n'est pas trouvé, 0 est renvoyé.
strapos (x1, x2)	bigint	Renvoie la position de départ de la dernière instance de x2 dans x1. Les positions commencent par 1. S'il n'est pas trouvé, 0 est renvoyé.

Fonction	Type de données de sortie	Description
strrpos (x1, x2, instance de bigint)	bigint	Renvoie la position de la nième instance de x2 dans x1 à partir de la fin de x1. L'instance doit être un nombre positif. Les positions commencent par 1. S'il n'est pas trouvé, 0 est renvoyé.
position (x2 IN x1)	bigint	Renvoie la position de départ de la première instance de x2 dans x1. Les positions commencent par 1. S'il n'est pas trouvé, 0 est renvoyé.
substr (x, bigint start)	varchar	Renvoie le reste de x à partir de la position de départ. Les positions commencent par 1. Une position de départ négative est interprétée comme étant relative à la fin de x.
substr (x, bigint start, bigint len)	varchar	Renvoie une sous-chaîne à partir de x de longueur len à partir de la position de départ. Les positions commencent par 1. Une position de départ négative est interprétée comme étant relative à la fin de x.
garniture (x)	varchar	Supprime les espaces blancs de début et de fin de x.
supérieur (x)	varchar	Convertit x en majuscules.

Opérateurs de réseaux

Timestream for LiveAnalytics prend en charge les opérateurs de tableau suivants.

Opérateur	Description
[]	Accédez à un élément d'un tableau dont le premier index commence à 1.
||	Concatène un tableau avec un autre tableau ou élément du même type.

Fonctions de tableau

Timestream for LiveAnalytics prend en charge les fonctions de tableau suivantes.

Fonction	Type de données de sortie	Description
tableau distinct (x)	array	Supprimez les valeurs dupliquées du tableau x. `SELECT array_distinct(ARRAY[1,2,2,3])` Exemple de résultat : `[ 1,2,3 ]`
array_intersect (x, y)	array	Renvoie un tableau des éléments situés à l'intersection de x et y, sans doublons. `SELECT array_intersect(ARRAY[1,2,3], ARRAY[3,4,5])` Exemple de résultat : `[ 3 ]`

Fonction	Type de données de sortie	Description
array_union (x, y)	array	Renvoie un tableau des éléments dans l'union de x et y, sans doublons. `SELECT array_union(ARRAY[1,2,3], ARRAY[3,4,5])` Exemple de résultat : `[ 1,2,3,4,5 ]`
array_except (x, y)	array	Renvoie un tableau d'éléments en x mais pas en y, sans doublons. `SELECT array_except(ARRAY[1,2,3], ARRAY[3,4,5])` Exemple de résultat : `[ 1,2 ]`
array_join (x, délimiteur, remplacement nul)	varchar	Concatène les éléments du tableau donné à l'aide du délimiteur et d'une chaîne facultative pour remplacer les valeurs nulles. `SELECT array_join(ARRAY[1,2,3], ';', '')` Exemple de résultat : `1;2;3`

Fonction	Type de données de sortie	Description
array_max (x)	identique aux éléments du tableau	Renvoie la valeur maximale du tableau d'entrée. `SELECT array_max (ARRAY[1,2,3])` Exemple de résultat : 3
table_min (x)	identique aux éléments du tableau	Renvoie la valeur minimale du tableau d'entrée. `SELECT array_min (ARRAY[1,2,3])` Exemple de résultat : 1
array_position (x, élément)	bigint	Renvoie la position de la première occurrence de l'élément dans le tableau x (ou 0 s'il est introuvable). `SELECT array_pos ition(ARRAY[3,4,5,9], 5)` Exemple de résultat : 3
array_remove (x, élément)	array	Supprimez tous les éléments égaux du tableau x. `SELECT array_rem ove(ARRAY[3,4,5,9], 4)` Exemple de résultat : `[ 3,5,9 ]`

Fonction	Type de données de sortie	Description
array_sort (x)	array	Trie et renvoie le tableau x. Les éléments de x doivent être commandables. Les éléments nuls seront placés à la fin du tableau renvoyé. `SELECT array_sor t(ARRAY[6,8,2,9,3])` Exemple de résultat : `[ 2,3,6,8,9 ]`
arrays_overlap (x, y)	boolean	Teste si les tableaux x et y ont des éléments non nuls en commun. Renvoie null s'il n'y a aucun élément non nul en commun mais que l'un ou l'autre des tableaux contient des valeurs nulles. `SELECT arrays_ov erlap(ARRAY[6,8,2, 9,3], ARRAY[6,8])` Exemple de résultat : `true`
cardinalité (x)	bigint	Renvoie la taille du tableau x. `SELECT cardinali ty(ARRAY[6,8,2,9,3])` Exemple de résultat : 5

Fonction	Type de données de sortie	Description
concat (tableau1, matrice2,..., arrayN)	array	Concatène les tableaux array1, array2,..., arrayN. `SELECT concat(ARRAY[6,8,2,9,3], ARRAY[11,32], ARRAY[6,8,2,0,14])` Exemple de résultat : `[ 6,8,2,9,3,11,32,6,8,2,0,14 ]`
element_at (tableau (E), index)	E	Renvoie l'élément du tableau à un index donné. Si index est inférieur à 0, element_at accède aux éléments du dernier au premier. `SELECT element_at(ARRAY[6,8,2,9,3], 1)` Exemple de résultat : 6
répéter (élément, nombre)	array	Répétez l'élément pour le comptage des fois. `SELECT repeat(1, 3)` Exemple de résultat : `[ 1,1,1 ]`

Fonction	Type de données de sortie	Description

inverse (x)	array	Renvoie un tableau dont l'ordre est inversé du tableau x.
		`SELECT reverse(ARRAY[6,8,2,9,3])`
		Exemple de résultat :
		`[ 3,9,2,8,6 ]`

séquence (démarrage, arrêt)	tableau (bigint)	Générez une séquence d'entiers du début à la fin, en l'incrémentant de 1 si le début est inférieur ou égal à l'arrêt, sinon -1.
		`SELECT sequence(3, 8)`
		Exemple de résultat :
		`[ 3,4,5,6,7,8 ]`

séquence (démarrage, arrêt, étape)	tableau (bigint)	Générez une séquence d'entiers du début à la fin, en les incrémentant pas à pas.
		`SELECT sequence(3, 15, 2)`
		Exemple de résultat :
		`[ 3,5,7,9,11,13,15 ]`

Fonction	Type de données de sortie	Description

séquence (démarrage, arrêt)	tableau (horodatage)	Générez une séquence d'horodatages allant de la date de début à la date de fin, en l'incrémentant d'un jour.
		
		`SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-06 19:26:12.941000000', 1d)`
		
		Exemple de résultat :
		`[ 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 ]`

Fonction	Type de données de sortie	Description

séquence (démarrage, arrêt, étape)	tableau (horodatage)	Générez une séquence d'horodatages du début à la fin, en incrémentant étape par étape. Le type de données de l'étape est intervalle.
		`SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-10 19:26:12.941000000', 2d)`
		Exemple de résultat :
		`[ 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 ]`

mélanger (x)	array	Génère une permutation aléatoire du tableau x donné.
		`SELECT shuffle(ARRAY[6,8,2,9,3])`
		Exemple de résultat :
		`[ 6,3,2,9,8 ]`

Fonction	Type de données de sortie	Description
tranche (x, début, longueur)	array	Sous-ensembles le tableau x à partir du début de l'index (ou à partir de la fin si le début est négatif) avec une longueur de longueur. `SELECT slice(ARRAY[6,8,2,9,3], 1, 3)` Exemple de résultat : `[ 6,8,2 ]`
zip (tableau1, tableau2 [,...])	tableau (ligne)	Fusionne les tableaux donnés, élément par élément, en un seul tableau de lignes. Si les arguments ont une longueur inégale, les valeurs manquantes sont remplies`NULL`. `SELECT zip(ARRAY[6,8,2,9,3], ARRAY[15, 24])` Exemple de résultat : `[ ( 6, 15 ),( 8, 24 ),( 2, - ),( 9, - ),( 3, - ) ]`

Fonctions bitwise

Timestream for LiveAnalytics prend en charge les fonctions binaires suivantes.

Fonction	Type de données de sortie	Description

bit_count (bigint, bigint)	bigint (complément à deux)	Renvoie le nombre de bits dans le premier paramètre bigint où le second paramètre est un entier signé par un bit tel que 8 ou 64.
		`SELECT bit_count(19, 8)`
		Exemple de résultat : `3`
		`SELECT bit_count(19, 2)`
		Exemple de résultat : `Number must be represent able with the bits specified. 19 can not be represented with 2 bits`

bitwise_and (bigint, bigint)	bigint (complément à deux)	Renvoie le nombre de paramètres bigint au niveau AND du bit. `SELECT bitwise_and(12, 7)` Exemple de résultat : `4`
bitwise_not (bigint)	bigint (complément à deux)	Renvoie le paramètre bigint au niveau NOT du bit. `SELECT bitwise_not(12)` Exemple de résultat : `-13`

Fonction	Type de données de sortie	Description
bitwise_or (bigint, bigint)	bigint (complément à deux)	Renvoie le OR bit à bit des paramètres bigint. `SELECT bitwise_or(12, 7)` Exemple de résultat : 15
bitwise_xor (bigint, bigint)	bigint (complément à deux)	Renvoie le nombre de paramètres bigint au niveau XOR du bit. `SELECT bitwise_xor(12, 7)` Exemple de résultat : 11

Fonctions d'expression régulière

Les fonctions d'expression régulière dans Timestream prennent en LiveAnalytics charge la syntaxe des modèles Java. Timestream for LiveAnalytics prend en charge les fonctions d'expression régulière suivantes.

Fonction	Type de données de sortie	Description
regexp_extract_all (chaîne, modèle)	array(varchar)	Renvoie la ou les sous-chaî nes correspondant au modèle d'expression régulière dans la chaîne. `SELECT regexp_ex tract_all('example expect complex', 'ex \w')`

Fonction	Type de données de sortie	Description
		Exemple de résultat : `[ exa,exp ]`
regexp_extract_all (chaîne, modèle, groupe)	array(varchar)	Trouve toutes les occurrences du modèle d'expression régulière dans une chaîne et renvoie le groupe de numéros du groupe de capture. `SELECT regexp_extract_all('example expect complex', '(ex)(\w)', 2)` Exemple de résultat : `[ a,p ]`
regexp_extract (chaîne, modèle)	varchar	Renvoie la première sous-chaîne correspondant au modèle d'expression régulière dans la chaîne. `SELECT regexp_extract('example expect', 'ex\w')` Exemple de résultat : `exa`

Fonction	Type de données de sortie	Description
regexp_extract (chaîne, modèle, groupe)	varchar	Trouve la première occurrenc e du modèle d'expression régulière dans une chaîne et renvoie le groupe de numéros du groupe de capture. `SELECT regexp_ex tract('example expect', '(ex)(\w)', 2)` Exemple de résultat : a
regexp_like (chaîne, motif)	boolean	Évalue le modèle d'express ion régulière et détermine s'il est contenu dans une chaîne. Cette fonction est similaire à l'LIKEopérateur, sauf que le modèle doit uniquement être contenu dans la chaîne, au lieu de devoir correspondre à l'ensemble de la chaîne. En d'autres termes, cela effectue une opération de contenu plutôt qu'une opération de correspondance. Vous pouvez faire correspondre la chaîne entière en ancrant le motif en utilisant ^ et $. `SELECT regexp_li ke('example', 'ex')` Exemple de résultat : `true`

Fonction	Type de données de sortie	Description
regexp_replace (chaîne, modèle)	varchar	Supprime toutes les instances de la sous-chaîne correspon dant au modèle d'expression régulière de la chaîne. `SELECT regexp_re place('example expect', 'expect')` Exemple de résultat : `example`
regexp_replace (chaîne, modèle, remplacement)	varchar	Remplace chaque instance de la sous-chaîne correspon dant au modèle regex dans la chaîne par un remplacem ent. Les groupes de capture peuvent être référencés en remplacement en utilisant $g pour un groupe numéroté ou $ {name} pour un groupe nommé. Un signe dollar ($) peut être inclus dans le remplacement en le remplaçant par une barre oblique inverse (\ $). `SELECT regexp_re place('example expect', 'expect', 'surprise')` Exemple de résultat : `example surprise`

Fonction	Type de données de sortie	Description
regexp_replace (chaîne, modèle, fonction)	varchar	Remplace chaque instance de la sous-chaîne correspondant au modèle d'expression régulière dans une chaîne à l'aide d'une fonction. La fonction d'expression lambda est invoquée pour chaque correspondance, les groupes de capture étant transmis sous forme de tableau. La capture des numéros de groupe commence par un ; il n'existe aucun groupe pour l'ensemble de la correspondance (si nécessaire, entourez l'expression entière de parenthèses). `SELECT regexp_replace('example', '(\w)', x -> upper(x[1]))` Exemple de résultat : `EXAMPLE`

Fonction	Type de données de sortie	Description
regexp_split (chaîne, modèle)	array(varchar)	Divise la chaîne en utilisant le modèle d'expression régulière et renvoie un tableau. Les chaînes vides de fin sont conservées. `SELECT regexp_split('example', 'x')` Exemple de résultat : `[ e,ample ]`

Opérateurs de date et d'heure

> **Note**
>
> Timestream for LiveAnalytics ne prend pas en charge les valeurs temporelles négatives. Toute opération entraînant une durée négative entraîne une erreur.

Timestream for LiveAnalytics prend en charge les opérations suivantes sur `timestampsdates`, et. `intervals`

Opérateur	Description
+	Addition
-	Soustraction

Rubriques

- Opérations
- Addition
- Soustraction

Opérations

Le type de résultat d'une opération est basé sur les opérandes. Des littéraux d'intervalle tels que `1day` et `3s` peuvent être utilisés.

```
SELECT date '2022-05-21' + interval '2' day
```

```
SELECT date '2022-05-21' + 2d
```

```
SELECT date '2022-05-21' + 2day
```

Exemple de résultat pour chacun d'entre eux : `2022-05-23`

Les unités d'intervalle incluent `second` `minutehour`,`day`,`week`,`month`, et`year`. Mais dans certains cas, tous ne sont pas applicables. Par exemple, les secondes, les minutes et les heures ne peuvent pas être ajoutées ou soustraites à une date.

```
SELECT interval '4' year + interval '2' month
```

Exemple de résultat : `4-2`

```
SELECT typeof(interval '4' year + interval '2' month)
```

Exemple de résultat : `interval year to month`

Le type de résultat des opérations d'intervalle peut être `'interval year to month'` ou `'interval day to second'` dépendre des opérandes. Les intervalles peuvent être ajoutés ou soustraits de `dates` et`timestamps`. Mais un `date` ou `timestamp` ne peut pas être ajouté ou soustrait d'un `date` ou`timestamp`. Pour trouver les intervalles ou les durées liés aux dates ou aux horodatages, voir `date_diff` et les fonctions associées dans. Intervalle et durée

Addition

Example

```
SELECT date '2022-05-21' + interval '2' day
```

Exemple de résultat : `2022-05-23`

Example

```
SELECT typeof(date '2022-05-21' + interval '2' day)
```

Exemple de résultat : `date`

Example

```
SELECT interval '2' year + interval '4' month
```

Exemple de résultat : `2-4`

Example

```
SELECT typeof(interval '2' year + interval '4' month)
```

Exemple de résultat : `interval year to month`

Soustraction

Example

```
SELECT timestamp '2022-06-17 01:00' - interval '7' hour
```

Exemple de résultat : `2022-06-16 18:00:00.000000000`

Example

```
SELECT typeof(timestamp '2022-06-17 01:00' - interval '7' hour)
```

Exemple de résultat : `timestamp`

Example

```
SELECT interval '6' day - interval '4' hour
```

Exemple de résultat : `5 20:00:00.000000000`

Example

```
SELECT typeof(interval '6' day - interval '4' hour)
```

Exemple de résultat : `interval day to second`

Fonctions de date/heure

> **Note**
>
> Timestream for LiveAnalytics ne prend pas en charge les valeurs temporelles négatives. Toute opération entraînant une durée négative entraîne une erreur.

Timestream pour LiveAnalytics utilise le UTC fuseau horaire pour la date et l'heure. Timestream prend en charge les fonctions suivantes pour la date et l'heure.

Rubriques

Généralités et conversion

Timestream for LiveAnalytics prend en charge les fonctions générales et de conversion suivantes pour la date et l'heure.

Fonction	Type de données de sortie	Description
date_actuelle	date	Renvoie la date actuelle enUTC. Aucune parenthèse n'est utilisée. `SELECT current_date` Exemple de résultat : `2022-07-07`

Fonction	Type de données de sortie	Description
		Note Il s'agit également d'un mot clé réservé. Pour obtenir la liste des mots clés réservés, consultezMots-clés réservés.
heure_actuelle	time	Renvoie l'heure actuelleU TC. Aucune parenthèse n'est utilisée. `SELECT current_time` Exemple de résultat : `17:41:52.827000000` **Note** Il s'agit également d'un mot clé réservé. Pour obtenir la liste des mots clés réservés, consultezMots-clés réservés.

Fonction	Type de données de sortie	Description
current_timestamp ou now ()	timestamp	Renvoie l'horodatage actuel en. UTC `SELECT current_timestamp` Exemple de résultat : `2022-07-07 17:42:32.939000000` **Note** Il s'agit également d'un mot clé réservé. Pour obtenir la liste des mots clés réservés, consultezMots-clés réservés.
fuseau horaire actuel ()	varchar La valeur sera «UTC. »	Timestream utilise le UTC fuseau horaire pour la date et l'heure. `SELECT current_timezone()` Exemple de résultat : UTC
date (varchar (x)), date (horodatage)	date	`SELECT date(TIMESTAMP '2022-07-07 17:44:43.771000000')` Exemple de résultat : `2022-07-07`

Fonction	Type de données de sortie	Description
dernier_jour_de_mois (horodatage), dernier_jour_de_mois (date)	date	`SELECT last_day_of_month(TIMESTAMP '2022-07-07 17:44:43.771000000')` Exemple de résultat : `2022-07-31`
from_iso8601_timestamp (chaîne de caractères)	timestamp	Analyse l'horodatage ISO 8601 au format d'horodatage interne. `SELECT from_iso8601_timestamp('2022-06-17T08:04:05.000000000+05:00')` Exemple de résultat : `2022-06-17 03:04:05.000000000`
from_iso8601_date (chaîne de caractères)	date	Analyse la chaîne de date ISO 8601 au format d'horodatage interne pour UTC 00:00:00 de la date spécifiée. `SELECT from_iso8601_date('2022-07-17')` Exemple de résultat : `2022-07-17`

Fonction	Type de données de sortie	Description
to_iso8601 (horodatage), to_iso8601 (date)	varchar	Renvoie une chaîne au format ISO 8601 pour l'entrée. `SELECT to_iso8601(from_iso8601_date('2022-06-17'))` Exemple de résultat : `2022-06-17`
from_milliseconds (bigint)	timestamp	`SELECT from_milliseconds(1)` Exemple de résultat : `1970-01-01 00:00:00.001000000`
from_nanoseconds (bigint)	timestamp	`select from_nanoseconds(300000001)` Exemple de résultat : `1970-01-01 00:00:00.300000001`
from_unixtime (double)	timestamp	Renvoie un horodatage qui correspond à l'unixtime fourni. `SELECT from_unixtime(1)` Exemple de résultat : `1970-01-01 00:00:01.000000000`

Fonction	Type de données de sortie	Description
heure locale	time	Renvoie l'heure actuelleU TC. Aucune parenthèse n'est utilisée. `SELECT localtime` Exemple de résultat : `17:58:22.654000000` **Note** Il s'agit également d'un mot clé réservé. Pour obtenir la liste des mots clés réservés, consultezMots-clés réservés.

Fonction	Type de données de sortie	Description
horodatage local	timestamp	Renvoie l'horodatage actuel en. UTC Aucune parenthèse n'est utilisée. `SELECT localtimestamp` Exemple de résultat : `2022-07-07 17:59:04.368000000` **Note** Il s'agit également d'un mot clé réservé. Pour obtenir la liste des mots clés réservés, consultezMots-clés réservés.

to_milliseconds (intervalle jour par seconde), to_milliseconds (horodatage)	bigint	`SELECT to_milliseconds(INTERVAL '2' DAY + INTERVAL '3' HOUR)`
		Exemple de résultat : `183600000`
		`SELECT to_milliseconds(TIMESTAMP '2022-06-17 17:44:43.771000000')`
		Exemple de résultat : `1655487883771`

Fonction	Type de données de sortie	Description

to_nanoseconds (intervalle jour par seconde), to_nanose conds (horodatage)	bigint	`SELECT to_nanose conds(INTERVAL '2' DAY + INTERVAL '3' HOUR)`
		
		Exemple de résultat : `183600000000000`
		
		`SELECT to_nanose conds(TIMESTAMP '2022-06-17 17:44:43. 771000678')`
		
		Exemple de résultat : `1655487883771000678`

to_unixtime (horodatage)	double	Renvoie unixtime pour l'horodatage fourni. `SELECT to_unixti me('2022-06-17 17:44:43.771000000')` Exemple de résultat : `1.6554878837710001E9`

Fonction	Type de données de sortie	Description
date_trunc (unité, horodatage)	timestamp	Renvoie l'horodatage tronqué en unité, où l'unité est [seconde, minute, heure, jour, semaine, mois, trimestre ou année]. `SELECT date_trun c('minute', TIMESTAMP '2022-06-17 17:44:43. 771000000')` Exemple de résultat : `2022-06-17 17:44:00. 000000000`

Intervalle et durée

Timestream for LiveAnalytics prend en charge les fonctions d'intervalle et de durée suivantes pour la date et l'heure.

Fonction	Type de données de sortie	Description
date_add (unité, bigint, date), date_add (unité, bigint, heure), date_add (varchar (x), bigint, horodatage)	timestamp	Ajoute un bigint d'unités, l'unité étant [seconde, minute, heure, jour, semaine, mois, trimestre ou année]. `SELECT date_add('hour', 9, TIMESTAMP '2022-06-17 00:00:00')` Exemple de résultat : `2022-06-17 09:00:00. 000000000`

Fonction	Type de données de sortie	Description
date_diff (unité, date, date), date_diff (unité, heure, heure), date_diff (unité, horodatage, horodatage)	bigint	Renvoie une différence dont l'unité est [seconde, minute, heure, jour, semaine, mois, trimestre ou année]. `SELECT date_diff('day', DATE '2020-03-01', DATE '2020-03-02')` Exemple de résultat : 1

parse_duration (chaîne de caractères)	interval	Analyse la chaîne d'entrée pour renvoyer un `interval` équivalent.
		`SELECT parse_duration('42.8ms')`
		Exemple de résultat : `0 00:00:00.042800000`
		`SELECT typeof(parse_duration('42.8ms'))`
		Exemple de résultat : `interval day to second`

Fonction	Type de données de sortie	Description
bin (horodatage, intervalle)	timestamp	Arrondit la valeur entière du `timestamp` paramètre au multiple le plus proche de la valeur entière du `interval` paramètre. La signification de cette valeur de retour n'est peut-être pas évidente. Il est calculé à l'aide de l'arithmétique des entiers en divisant d'abord le nombre entier d'horodatage par l'entier d'intervalle, puis en multipliant le résultat par l'entier d'interva lle. En gardant à l'esprit qu'un horodatage indique un UTC point dans le temps en tant que nombre de fractions de seconde écoulées depuis l'POSIXépoque (1er janvier 1970), la valeur renvoyée s'alignera rarement sur les unités du calendrier. Par exemple, si vous spécifiez un intervalle de 30 jours, tous les jours écoulés depuis l'époque sont divisés en tranches de 30 jours, et le début de la dernière tranche de 30 jours est renvoyé, ce qui n'a aucun rapport avec les mois civils. Voici quelques exemples :

Fonction	Type de données de sortie	Description

		`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`

il y a (intervalle)	timestamp	Renvoie la valeur correspondant à `interval` current_timestamp. `SELECT ago(1d)` Exemple de résultat : `2022-07-06 21:08:53.245000000`
littéraux d'intervalle tels que 1h, 1d et 30m	interval	Les littéraux d'intervalle sont pratiques pour parse_duration (chaîne). Par exemple, `1d` est identique à `parse_duration('1d')` . Cela permet d'utiliser les littéraux partout où un intervalle est utilisé. Par exemple : `ago(1d)` et `bin(<timestamp> , 1m)`.

Certains littéraux d'intervalle servent de raccourci pour parse_duration. Par exemple,`parse_duration('1day')`, `1day`parse_duration('1d'), et `1d` chaque retour `1 00:00:00.000000000` où se trouve le type`interval day to second`. L'espace est autorisé dans le format fourni à`parse_duration`. Par exemple, les retours sont `parse_duration('1day')` également`00:00:00.000000000`. Mais ce n'`1 day`est pas un intervalle littéral.

Les unités associées `interval day to second` sont ns, nanoseconde, us, microseconde, ms, milliseconde, s, seconde, m, minute, h, heure, d et jour.

Il y en a également`interval year to month`. Les unités associées à l'intervalle d'une année à l'autre sont y, année et mois. Par exemple, les `SELECT 1year` retours`1-0`. `SELECT 12month`revient également`1-0`. `SELECT 8month`retours`0-8`.

Bien que l'unité de `quarter` soit également disponible pour certaines fonctions telles que `date_trunc` et`date_add`, elle n'`quarter`est pas disponible dans le cadre d'un intervalle littéral.

Formatage et analyse

Timestream for LiveAnalytics prend en charge les fonctions de formatage et d'analyse suivantes pour la date et l'heure.

Fonction	Type de données de sortie	Description
date_format (horodatage, varchar (x))	varchar	Pour plus d'informations sur les spécificateurs de format utilisés par cette fonction, voir # 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')` Exemple de résultat : `2019-10-20 10:20:20`

Fonction	Type de données de sortie	Description
date_parse (varchar (x), varchar (y))	timestamp	Pour plus d'informations sur les spécificateurs de format utilisés par cette fonction, voir # 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')` Exemple de résultat : `2019-10-20 10:20:20.000000000`
format_datetime (horodatage, varchar (x))	varchar	Pour plus d'informations sur la chaîne de format utilisée par cette fonction, consultez 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')` Exemple de résultat : `1968-01-13 12`

Fonction	Type de données de sortie	Description
parse_datetime (varchar (x), varchar (y))	timestamp	Pour plus d'informations sur la chaîne de format utilisée par cette fonction, consultez 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')` Exemple de résultat : `2019-12-29 18:10:00.000000000`

Extraction

Timestream for LiveAnalytics prend en charge les fonctions d'extraction suivantes pour la date et l'heure. La fonction d'extraction est à la base des autres fonctions pratiques.

Fonction	Type de données de sortie	Description
extract	bigint	Extrait un champ d'un horodatage, où le champ est l'un des suivants [YEAR,,,QUARTER, _OF_ MONTH WEEKDAY, DAY _OF_MONTH,, DAY _OF_WEEK, DAY _OF_DOW,, YEAR,DOY, YEAR ou]. WEEK YOW HOUR MINUTE SECOND

Fonction	Type de données de sortie	Description
		`SELECT extract(YEAR FROM '2019-10-12 23:10:34.000000000')` Exemple de résultat : 2019
jour (horodatage), jour (date), jour (intervalle de jour à seconde)	bigint	`SELECT day('2019-10-12 23:10:34.000000000')` Exemple de résultat : 12
jour_of_month (horodatage), day_of_month (date), day_of_month (intervalle d'un jour à une seconde)	bigint	`SELECT day_of_month('2019-10-12 23:10:34.000000000')` Exemple de résultat : 12
jour_de_semaine (horodatage), jour_de_semaine (date)	bigint	`SELECT day_of_week('2019-10-12 23:10:34.000000000')` Exemple de résultat : 6
jour_de_année (horodatage), jour_de_année (date)	bigint	`SELECT day_of_year('2019-10-12 23:10:34.000000000')` Exemple de résultat : 285
dow (horodatage), dow (date)	bigint	Alias pour day_of_week
doy (horodatage), doy (date)	bigint	Alias pour day_of_year

Fonction	Type de données de sortie	Description
heure (horodatage), heure (heure), heure (intervalle du jour à la seconde)	bigint	`SELECT hour('2019-10-12 23:10:34.000000000')` Exemple de résultat : `23`
milliseconde (horodatage), milliseconde (heure), milliseconde (intervalle d'un jour à une seconde)	bigint	`SELECT millisecond('2019-10-12 23:10:34.000000000')` Exemple de résultat : `0`
minute (horodatage), minute (heure), minute (intervalle du jour à la seconde)	bigint	`SELECT minute('2019-10-12 23:10:34.000000000')` Exemple de résultat : `10`
mois (horodatage), mois (date), mois (intervalle d'une année à l'autre)	bigint	`SELECT month('2019-10-12 23:10:34.000000000')` Exemple de résultat : `10`
nanoseconde (horodatage), nanoseconde (temps), nanoseconde (intervalle entre un jour et une seconde)	bigint	`SELECT nanosecond(current_timestamp)` Exemple de résultat : `162000000`
trimestre (horodatage), trimestre (date)	bigint	`SELECT quarter('2019-10-12 23:10:34.000000000')` Exemple de résultat : `4`

Fonction	Type de données de sortie	Description
seconde (horodatage), seconde (heure), seconde (intervalle de jour à seconde)	bigint	`SELECT second('2019-10-12 23:10:34.000000000')` Exemple de résultat : `34`
semaine (horodatage), semaine (date)	bigint	`SELECT week('2019-10-12 23:10:34.000000000')` Exemple de résultat : `41`
semaine_de_année (horodatage), semaine_de_année (date)	bigint	Alias pour la semaine
année (horodatage), année (date), année (intervalle d'une année à l'autre)	bigint	`SELECT year('2019-10-12 23:10:34.000000000')` Exemple de résultat : `2019`
année_de_semaine (horodatage), année_de_semaine (date)	bigint	`SELECT year_of_week('2019-10-12 23:10:34.000000000')` Exemple de résultat : `2019`
yow (horodatage), yow (date)	bigint	Alias pour year_of_week

Fonctions d’agrégation

Timestream for LiveAnalytics prend en charge les fonctions d'agrégation suivantes.

Fonction	Type de données de sortie	Description
arbitraire (x)	[identique à l'entrée]	Renvoie une valeur arbitrair e non nulle de x, s'il en existe une. `SELECT arbitrary(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : 1
array_agg (x)	array< [identique à l'entrée]	Renvoie un tableau créé à partir des éléments x d'entrée. `SELECT array_agg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `[ 1,2,3,4 ]`
moyenne (x)	double	Renvoie la moyenne (moyenne arithmétique) de toutes les valeurs d'entrée. `SELECT avg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `2.5`
bool_and (boolean) every (booléen)	boolean	Renvoie TRUE si chaque valeur d'entrée l'estTRUE, sinonFALSE. `SELECT bool_and(t.c) FROM (VALUES true,`

Fonction	Type de données de sortie	Description
		`true, false, true) AS t(c)` Exemple de résultat : `false`
bool_or (booléen)	boolean	Renvoie TRUE si une valeur d'entrée l'estTRUE, sinonFALSE. `SELECT bool_or(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemple de résultat : `true`
compte (*) compte (x)	bigint	count (*) renvoie le nombre de lignes d'entrée. count (x) renvoie le nombre de valeurs d'entrée non nulles. `SELECT count(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemple de résultat : 4
nombre_if (x)	bigint	Renvoie le nombre de valeurs TRUE d'entrée. `SELECT count_if(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemple de résultat : 3

Fonction	Type de données de sortie	Description
moyenne géométrique (x)	double	Renvoie la moyenne géométrique de toutes les valeurs d'entrée. `SELECT geometric _mean(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `2.213363839400643`
max_by (x, y)	[identique à x]	Renvoie la valeur de x associée à la valeur maximale de y sur toutes les valeurs d'entrée. `SELECT max_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemple de résultat : d

Fonction	Type de données de sortie	Description
max_by (x, y, n)	réseau< [same as x] >	Renvoie n valeurs de x associées à la plus grande valeur n de toutes les valeurs d'entrée de y par ordre décroissant de y. `SELECT max_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemple de résultat : `[ d,c ]`
min_by (x, y)	[identique à x]	Renvoie la valeur de x associée à la valeur minimale de y sur toutes les valeurs d'entrée. `SELECT min_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemple de résultat : a

Fonction	Type de données de sortie	Description
min_by (x, y, n)	réseau< [same as x] >	Renvoie n valeurs de x associées à la n plus petite de toutes les valeurs d'entrée de y dans l'ordre croissant de y. `SELECT min_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemple de résultat : `[ a,b ]`
maximum (x)	[identique à l'entrée]	Renvoie la valeur maximale de toutes les valeurs d'entrée. `SELECT max(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : 4
maximum (x, n)	réseau< [same as x] >	Renvoie n valeurs les plus élevées de toutes les valeurs d'entrée de x. `SELECT max(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `[ 4,3 ]`

Fonction	Type de données de sortie	Description
minute (x)	[identique à l'entrée]	Renvoie la valeur minimale de toutes les valeurs d'entrée. `SELECT min(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : 1
minute (x, n)	réseau< [same as x] >	Renvoie n plus petites valeurs de toutes les valeurs d'entrée de x. `SELECT min(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `[ 1,2 ]`
somme (x)	[identique à l'entrée]	Renvoie la somme de toutes les valeurs d'entrée. `SELECT sum(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `10`

Fonction	Type de données de sortie	Description
bitwise_et_agg (x)	bigint	Renvoie le bit à bit AND de toutes les valeurs d'entrée dans une représentation complémentaire à 2 s. `SELECT bitwise_and_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Exemple de résultat : 1
bitwise_or_agg (x)	bigint	Renvoie le OR au niveau du bit de toutes les valeurs d'entrée dans une représentation complémentaire à 2 s. `SELECT bitwise_or_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Exemple de résultat : -3

Fonction	Type de données de sortie	Description
approx_distinct (x)	bigint	Renvoie le nombre approximatif de valeurs d'entrée distinctes. Cette fonction fournit une approximation de count (DISTINCTx). Zéro est renvoyé si toutes les valeurs d'entrée sont nulles. Cette fonction doit produire une erreur type de 2,3 %, qui est l'écart type de la distribution des erreurs (approximativement normale) sur tous les ensembles possibles. Cela ne garantit pas une limite supérieure de l'erreur pour un ensemble d'entrées spécifique. `SELECT approx_distinct(t.c) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Exemple de résultat : 5

Fonction	Type de données de sortie	Description
approx_distinct (x, e)	bigint	Renvoie le nombre approximatif de valeurs d'entrée distinctes. Cette fonction fournit une approximation de count (DISTINCTx). Zéro est renvoyé si toutes les valeurs d'entrée sont nulles. Cette fonction doit produire une erreur type ne dépassant pas e, qui est l'écart type de la distribution d'erreur (approximativement normale) sur tous les ensembles possibles . Cela ne garantit pas une limite supérieure de l'erreur pour un ensemble d'entrées spécifique. L'implémentation actuelle de cette fonction nécessite que e soit compris entre [0,0040625, 0,26000]. `SELECT approx_distinct(t.c, 0.2) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Exemple de résultat : 5

Fonction	Type de données de sortie	Description
approx_percentile (x, pourcentage)	[identique à x]	Renvoie le percentile approximatif pour toutes les valeurs d'entrée de x au pourcentage donné. La valeur du pourcentage doit être comprise entre zéro et un et doit être constante pour toutes les lignes d'entrée. `SELECT approx_percentile(t.c, 0.4) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : 2
approx_percentile (x, pourcentages)	réseau< [same as x] >	Renvoie le percentile approximatif pour toutes les valeurs d'entrée de x pour chacun des pourcentages spécifiés. Chaque élément du tableau des pourcentages doit être compris entre zéro et un, et le tableau doit être constant pour toutes les lignes d'entrée. `SELECT approx_percentile(t.c, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `[ 1,4,4 ]`

Fonction	Type de données de sortie	Description
approx_percentile (x, w, pourcentage)	[identique à x]	Renvoie le percentile pondéré approximatif pour toutes les valeurs d'entrée de x en utilisant le poids par article w au pourcentage p. Le poids doit être une valeur entière d'au moins un. Il s'agit en fait d'un nombre de réplicati ons pour la valeur x dans l'ensemble de percentiles. La valeur de p doit être comprise entre zéro et un et doit être constante pour toutes les lignes d'entrée. `SELECT approx_pe rcentile(t.c, 1, 0.1) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : 1

Fonction	Type de données de sortie	Description
approx_percentile (x, w, pourcentages)	réseau< [same as x] >	Renvoie le percentile pondéré approximatif pour toutes les valeurs d'entrée de x en utilisant le poids par article w à chacun des pourcenta ges indiqués dans le tableau. Le poids doit être une valeur entière d'au moins un. Il s'agit en fait d'un nombre de réplicati ons pour la valeur x dans l'ensemble de percentiles. Chaque élément du tableau doit être compris entre zéro et un, et le tableau doit être constant pour toutes les lignes d'entrée. `SELECT approx_pe rcentile(t.c, 1, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : `[ 1,4,4 ]`

Fonction	Type de données de sortie	Description
approx_percentile (x, w, pourcentage, précision)	[identique à x]	Renvoie le percentile pondéré approximatif pour toutes les valeurs d'entrée de x en utilisant le poids par article w au pourcentage p, avec une erreur de précision de classement maximale. Le poids doit être une valeur entière d'au moins un. Il s'agit en fait d'un nombre de réplications pour la valeur x dans l'ensemble de percentiles. La valeur de p doit être comprise entre zéro et un et doit être constante pour toutes les lignes d'entrée. La précision doit être une valeur supérieure à zéro et inférieure à un, et elle doit être constante pour toutes les lignes en entrée. `SELECT approx_percentile(t.c, 1, 0.1, 0.5) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemple de résultat : 1

Fonction	Type de données de sortie	Description
corr (y, x)	double	Renvoie le coefficient de corrélation des valeurs d'entrée. `SELECT corr(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemple de résultat : `1.0`
covar_pop (y, x)	double	Renvoie la covariance des valeurs d'entrée dans la population. `SELECT covar_pop(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemple de résultat : `1.25`
covar_samp (y, x)	double	Renvoie la covariance de l'échantillon des valeurs d'entrée. `SELECT covar_samp(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemple de résultat : `1.6666666666666667`

Fonction	Type de données de sortie	Description
reg_interception (y, x)	double	Renvoie l'intersection par régression linéaire des valeurs d'entrée. y est la valeur dépendante. x est la valeur indépendante. `SELECT regr_intercept(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemple de résultat : `0.0`
reg_slope (y, x)	double	Renvoie la pente de régression linéaire des valeurs d'entrée. y est la valeur dépendante. x est la valeur indépendante. `SELECT regr_slope(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemple de résultat : `1.0`
asymétrie (x)	double	Renvoie l'asymétrie de toutes les valeurs d'entrée. `SELECT skewness(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemple de résultat : `0.8978957037987335`

Fonction	Type de données de sortie	Description
stddev_pop (x)	double	Renvoie l'écart type de la population de toutes les valeurs d'entrée. `SELECT stddev_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemple de résultat : `2.4166091947189146`
stddev_samp (x) stddev (x)	double	Renvoie l'écart type de l'échantillon de toutes les valeurs d'entrée. `SELECT stddev_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemple de résultat : `2.701851217221259`
var_pop (x)	double	Renvoie la variance de population de toutes les valeurs d'entrée. `SELECT var_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemple de résultat : `5.840000000000001`

Fonction	Type de données de sortie	Description
var_samp (x) variance (x)	double	Renvoie la variance d'échantillon de toutes les valeurs d'entrée. `SELECT var_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemple de résultat : `7.300000000000001`

Fonctions de fenêtrage

Les fonctions de fenêtre effectuent des calculs sur les lignes du résultat de la requête. Ils s'exécutent après la HAVING clause mais avant la clause ORDER BY. L'appel d'une fonction de fenêtre nécessite une syntaxe spéciale utilisant la OVER clause pour spécifier la fenêtre. Une fenêtre comporte trois éléments :

- Spécification de partition, qui sépare les lignes d'entrée en différentes partitions. Cela est analogue à la façon dont la clause GROUP BY sépare les lignes en différents groupes pour les fonctions d'agrégation.
- La spécification d'ordre, qui détermine l'ordre dans lequel les lignes d'entrée seront traitées par la fonction de fenêtre.
- Le cadre de fenêtre, qui indique une fenêtre coulissante de lignes à traiter par la fonction pour une ligne donnée. Si le cadre n'est pas spécifié, sa valeur par défaut est identique RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW à RANGE UNBOUNDEDPRECEDING. Ce cadre contient toutes les lignes depuis le début de la partition jusqu'au dernier pair de la ligne en cours.

Toutes les fonctions d'agrégation peuvent être utilisées comme fonctions de fenêtre en ajoutant la OVER clause. La fonction d'agrégation est calculée pour chaque ligne au-dessus des lignes du cadre de fenêtre de la ligne en cours. Outre les fonctions d'agrégation, Timestream for LiveAnalytics prend en charge les fonctions de classement et de valeur suivantes.

Fonction	Type de données de sortie	Description
cume_dist ()	bigint	Renvoie la distribution cumulée d'une valeur dans un groupe de valeurs. Le résultat est le nombre de lignes précédant ou correspondant à la ligne dans l'ordre des fenêtres de la partition de fenêtre divisé par le nombre total de lignes de la partition de fenêtre. Ainsi, toutes les valeurs d'égalité figurant dans l'ordre seront évaluées à la même valeur de distribution.
dense_rank ()	bigint	Renvoie le rang d'une valeur dans un groupe de valeurs. Ceci est similaire à rank (), sauf que les valeurs d'égalité ne produisent pas de lacunes dans la séquence.
Titre (n)	bigint	Divise les lignes de chaque partition de fenêtre en n compartiments allant de 1 à n au maximum. Les valeurs des compartiments différeront d'au plus 1. Si le nombre de lignes de la partition ne correspond pas uniformément au nombre de compartiments, les valeurs restantes sont distribuées une par compartiment, en commençant par le premier compartiment.

Fonction	Type de données de sortie	Description
classement en pourcentage ()	double	Renvoie le classement en pourcentage d'une valeur dans un groupe de valeurs. Le résultat est (r - 1)/(n - 1) où r est le rang () de la ligne et n est le nombre total de lignes dans la partition de fenêtre.
rang ()	bigint	Renvoie le rang d'une valeur dans un groupe de valeurs. Le rang est égal à un plus le nombre de lignes précédant la ligne qui ne correspon dent pas à la ligne. Ainsi, les valeurs d'égalité dans l'ordre produiront des écarts dans la séquence. Le classement est effectué pour chaque partition de fenêtre.
numéro_ligne ()	bigint	Renvoie un numéro séquentie l unique pour chaque ligne, en commençant par un, en fonction de l'ordre des lignes dans la partition de fenêtre.
première_valeur (x)	[identique à l'entrée]	Renvoie la première valeur de la fenêtre. Cette fonction est limitée au cadre de la fenêtre. La fonction prend une expression ou une cible comme paramètre.

Fonction	Type de données de sortie	Description
dernière_valeur (x)	[identique à l'entrée]	Renvoie la dernière valeur de la fenêtre. Cette fonction est limitée au cadre de la fenêtre. La fonction prend une expression ou une cible comme paramètre.
nth_value (x, décalage)	[identique à l'entrée]	Renvoie la valeur au décalage spécifié par rapport au début de la fenêtre. Les décalages commencent à 1. Le décalage peut être n'importe quelle expression scalaire. Si le décalage est nul ou supérieur au nombre de valeurs de la fenêtre, nul est renvoyé. Le fait que le décalage soit nul ou négatif est une erreur. La fonction prend une expressio n ou une cible comme premier paramètre.

Fonction	Type de données de sortie	Description
plomb (x [, offset [, valeur_défaut]])	[identique à l'entrée]	Renvoie la valeur des lignes décalées après la ligne actuelle de la fenêtre. Les décalages commencent à 0, qui correspond à la ligne actuelle. Le décalage peut être n'importe quelle expression scalaire. Le décalage par défaut est 1. Si le décalage est nul ou supérieur à celui de la fenêtre, la valeur par défaut est renvoyée, ou si elle n'est pas spécifiée, null est renvoyée. La fonction prend une expression ou une cible comme premier paramètre.
décalage (x [, offset [, valeur_défaut]])	[identique à l'entrée]	Renvoie la valeur des lignes décalées avant la ligne actuelle de la fenêtre Les décalages commencent à 0, qui est la ligne en cours. Le décalage peut être n'importe quelle expression scalaire. Le décalage par défaut est 1. Si le décalage est nul ou supérieur à celui de la fenêtre, la valeur par défaut est renvoyée, ou si elle n'est pas spécifiée, null est renvoyée. La fonction prend une expression ou une cible comme premier paramètre.

Exemples de requêtes

Cette section inclut des exemples de cas d'utilisation du langage de requête LiveAnalytics de Timestream for.

Rubriques

Requêtes simples

Ce qui suit permet d'obtenir les 10 derniers points de données ajoutés à une table.

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

Ce qui suit permet d'obtenir les 5 points de données les plus anciens pour une mesure spécifique.

```
SELECT * FROM <database_name>.<table_name>
WHERE measure_name = '<measure_name>'
ORDER BY time ASC
LIMIT 5
```

Ce qui suit fonctionne avec des horodatages à granularité nanoseconde.

```
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
```

Les valeurs de mesure pour les enregistrements à mesures multiples sont identifiées par le nom de colonne. Les valeurs de mesure pour les enregistrements à mesure unique sont identifiées par`measure_value::<data_type>`, où se `<data_type>` trouve l'un des `double`

`bigint`, `boolean`, ou `varchar` comme décrit dans[Types de données pris en charge](). Pour plus d'informations sur la façon dont les valeurs de mesure sont modélisées, voir [Table unique ou tables multiples]().

Ce qui suit récupère les valeurs d'une mesure appelée `speed` à partir d'enregistrements de plusieurs mesures avec un `measure_name` de. `IoTMulti-stats`

```
SELECT speed FROM <database_name>.<table_name> where measure_name = 'IoTMulti-stats'
```

Ce qui suit récupère `double` les valeurs des enregistrements à mesure unique avec un `measure_name` de. `load`

```
SELECT measure_value::double FROM <database_name>.<table_name> WHERE measure_name =
 'load'
```

Requêtes avec fonctions de séries chronologiques

Rubriques

- [Exemple de jeu de données et de requêtes]()

Exemple de jeu de données et de requêtes

Vous pouvez utiliser Timestream LiveAnalytics pour comprendre et améliorer les performances et la disponibilité de vos services et applications. Vous trouverez ci-dessous un exemple de table et des exemples de requêtes exécutées sur cette table.

La table `ec2_metrics` stocke les données de télémétrie, telles que CPU l'utilisation et d'autres mesures provenant EC2 des instances. Vous pouvez consulter le tableau ci-dessous.

Heure	region	az	Hostname	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	frontend 01	utilisation du processeur	35,1	null

Heure	region	az	Hostname	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	frontend 01	memory_utilization	55,3	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	frontend 01	network_bytes_in	null	1 500
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	frontend 01	octets_out du réseau	null	6 700
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	frontend 02	utilisation du processeur	38,5	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	frontend 02	memory_utilization	58,4	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	frontend 02	network_bytes_in	null	23 000
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	frontend 02	octets_out du réseau	null	12 000

Heure	region	az	Hostname	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend 03	utilisation du processeur	45.0	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend 03	memory_utilization	65,8	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend 03	network_bytes_in	null	15 000
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend 03	octets_out du réseau	null	836 000
2019-12-04 19:00:05.000 000000	us-east-1	us-east-1a	frontend 01	utilisation du processeur	55,2	null
2019-12-04 19:00:05.000 000000	us-east-1	us-east-1a	frontend 01	memory_utilization	75.0	null
2019-12-04 19:00:05.000 000000	us-east-1	us-east-1a	frontend 01	network_bytes_in	null	1 245

Heure	region	az	Hostname	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	frontend 01	octets_out du réseau	null	68 432
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend 02	utilisation du processeur	65,6	null
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend 02	memory_utilization	85,3	null
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend 02	network_bytes_in	null	1 245
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend 02	octets_out du réseau	null	68 432
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend 03	utilisation du processeur	12.1	null
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend 03	memory_utilization	32,0	null

Heure	region	az	Hostname	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend 03	network_bytes_in	null	1 400
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend 03	octets_out du réseau	null	345
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend 01	utilisation du processeur	15,3	null
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend 01	memory_utilization	35,4	null
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend 01	network_bytes_in	null	23
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend 01	octets_out du réseau	null	0
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend 02	utilisation du processeur	44.0	null

Heure	region	az	Hostname	nom_mesure	valeur_mesure : double	value_mesure : :bigint
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend 02	memory_utilization	64,2	null
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend 02	network_bytes_in	null	1 450
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend 02	octets_out du réseau	null	200
2019-12-04 19:00:40.000000000	us-east-1	us-east-1c	frontend 03	utilisation du processeur	66,4	null
2019-12-04 19:00:40.000000000	us-east-1	us-east-1c	frontend 03	memory_utilization	86,3	null
2019-12-04 19:00:40.000000000	us-east-1	us-east-1c	frontend 03	network_bytes_in	null	300
2019-12-04 19:00:40.000000000	us-east-1	us-east-1c	frontend 03	octets_out du réseau	null	423

Trouvez l'CPUutilisation moyenne, p90, p95 et p99 pour un EC2 hôte spécifique au cours des 2 dernières heures :

```
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
```

Identifiez les EC2 hôtes dont le taux d'CPUutilisation est supérieur de 10 % ou plus à l'CPUutilisation moyenne de l'ensemble du parc au cours des 2 dernières heures :

```
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
```

Trouvez le CPU taux d'utilisation moyen regroupé à intervalles de 30 secondes pour un EC2 hôte spécifique au cours des 2 dernières heures :

```
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
```

Déterminez l'CPUutilisation moyenne regroupée à intervalles de 30 secondes pour un EC2 hôte spécifique au cours des 2 dernières heures, en remplissant les valeurs manquantes à l'aide d'une interpolation linéaire :

```
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)
```

Trouvez l'CPUutilisation moyenne regroupée à intervalles de 30 secondes pour un EC2 hôte spécifique au cours des 2 dernières heures, en remplissant les valeurs manquantes à l'aide d'une interpolation basée sur la dernière observation reportée :

```
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)
```

Requêtes avec fonctions d'agrégation

Vous trouverez ci-dessous un exemple de jeu de données pour un scénario IoT illustrant des requêtes avec des fonctions agrégées.

Rubriques

- Exemple de données
- Exemples de requêtes

Exemple de données

Timestream vous permet de stocker et d'analyser les données des capteurs IoT telles que l'emplacement, la consommation de carburant, la vitesse et la capacité de charge d'une ou de plusieurs flottes de camions afin de permettre une gestion efficace de la flotte. Vous trouverez ci-dessous le schéma et certaines des données d'une table iot_trucks qui stocke des données télémétriques telles que l'emplacement, la consommation de carburant, la vitesse et la capacité de charge des camions.

Heure	truck_id	Marque	Modèle	Flotte	capacité_carburan	capacité_charge	nom_mesure	valeur_mesure : double	valeur_mesure : varchar
2019-12-4 19:00:00	1234567	GMC	Astro	Alpha	100	500	lecture de carburan	65,2	null

Heure	truck_id	Marque	Modèle	Flotte	capacité_carburan	capacité_charge	nom_mesure	valeur_mesure : double	valeur_mesure : varchar
000 000000									
2019-12-4 19:00:00 000 000000	1234567	GMC	Astro	Alpha	100	500	charge	400,0	null
2019-12-4 19:00:00 000 000000	1234567	GMC	Astro	Alpha	100	500	speed	90,2	null
2019-12-4 19:00:00 000 000000	1234567	GMC	Astro	Alpha	100	500	location	null	47,6062 M, 122,3321 W
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alpha	150	1 000	lecture de carburan	10.1	null
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alpha	150	1 000	charge	950,3	null

Heure	truck_id	Marque	Modèle	Flotte	capacité_carburan	capacité_charge	nom_me e	valeur_m sure : double	valeur_me sure : varchar
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alpha	150	1 000	speed	50,8	null
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alpha	150	1 000	location	null	40,7128 degrés N, 74,0060 degrés W

Exemples de requêtes

Obtenez une liste de tous les attributs et valeurs des capteurs surveillés pour chaque camion de la flotte.

```
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
```

Obtenez le dernier relevé de consommation de carburant de chaque camion de la flotte au cours des dernières 24 heures.

```
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
```

Identifiez les camions qui ont consommé peu de carburant (moins de 10 %) au cours des dernières 48 heures :

```
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
)
```

Trouvez la charge moyenne et la vitesse maximale de chaque camion au cours de la semaine écoulée :

```
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
```

Découvrez l'efficacité de chargement de chaque camion au cours de la semaine écoulée :

```
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
```

APIréférence

Cette section contient la documentation de API référence pour Amazon Timestream.

Timestream en a deux APIs : Query et Write.

- L'écriture vous API permet d'effectuer des opérations telles que la création de tables, le balisage des ressources et l'écriture d'enregistrements dans Timestream.
- La requête vous API permet d'effectuer des opérations de requête.

> **Note**
>
> Les deux APIs incluent l' DescribeEndpoints action. Pour Query et Write, les DescribeEndpoints actions sont identiques.

Vous pouvez en savoir plus sur chacune d'entre elles API ci-dessous, ainsi que sur les types de données, les erreurs courantes et les paramètres.

> **Note**
>
> Pour les codes d'erreur communs à tous les AWS services, consultez la section AWS Support.

Rubriques

Actions

Les actions suivantes sont prises en charge par Amazon Timestream Write :

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase
- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks
- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase

- UpdateTable
- WriteRecords

Les actions suivantes sont prises en charge par Amazon Timestream Query :

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

Amazon Timestream Write

Les actions suivantes sont prises en charge par Amazon Timestream Write :

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase

- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks
- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase
- UpdateTable
- WriteRecords

CreateBatchLoadTask

Service : Amazon Timestream Write

Crée une nouvelle tâche de chargement par lots Timestream. Une tâche de chargement par lots traite les données d'une CSV source située dans un emplacement S3 et les écrit dans une table Timestream. Un mappage de la source à la cible est défini dans une tâche de chargement par lots. Les erreurs et les événements sont consignés dans un rapport sur un site S3. Pour le rapport, si la AWS KMS clé n'est pas spécifiée, le rapport sera chiffré avec une clé gérée S3 lorsque `SSE_S3` c'est possible. Dans le cas contraire, une erreur est générée. Pour en savoir plus, consultez Clés gérées par AWS. Des quotas de service s'appliquent. Pour plus de détails, consultez l'exemple de code.

Syntaxe de la requête

```
{
   "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"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ClientToken

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 64.

Obligatoire : non

DataModelConfiguration

Type : objet DataModelConfiguration

Obligatoire : non

DataSourceConfiguration

Définit les détails de configuration de la source de données pour une tâche de chargement par lots.

Type : objet DataSourceConfiguration

Obligatoire : oui

RecordVersion

Type : long

Obligatoire : non

ReportConfiguration

Configuration du rapport pour une tâche de chargement par lots. Il contient des informations sur l'emplacement de stockage des rapports d'erreur.

Type : objet ReportConfiguration

Obligatoire : oui

TargetDatabaseName

Base de données Timestream cible pour une tâche de chargement par lots.

Type : chaîne

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

TargetTableName

Table Timestream cible pour une tâche de chargement par lots.

Type : chaîne

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Syntaxe de la réponse

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

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

TaskId

ID de la tâche de chargement par lots.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 32.

Modèle : `[A-Z0-9]+`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

ConflictException

Timestream n'a pas pu traiter cette demande car elle contient une ressource qui existe déjà.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ServiceQuotaExceededException

Le quota de ressources d'instance a été dépassé pour ce compte.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

CreateDatabase
Service : Amazon Timestream Write

Crée une nouvelle base de données Timestream. Si la AWS KMS clé n'est pas spécifiée, la base de données sera chiffrée à l'aide d'une AWS KMS clé gérée par Timestream située dans votre compte. Pour en savoir plus, consultez Clés gérées par AWS. Des quotas de service s'appliquent. Pour plus de détails, consultez l'exemple de code.

Syntaxe de la requête

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 256.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

KmsKeyId

La AWS KMS clé de la base de données. Si la AWS KMS clé n'est pas spécifiée, la base de données sera chiffrée à l'aide d'une AWS KMS clé gérée par Timestream située dans votre compte. Pour en savoir plus, consultez Clés gérées par AWS.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

Tags

Liste de paires clé-valeur pour étiqueter la table.

Type : tableau d'objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Obligatoire : non

Syntaxe de la réponse

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Database

La nouvelle base de données Timestream.

Type : objet Database

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

ConflictException

Timestream n'a pas pu traiter cette demande car elle contient une ressource qui existe déjà.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ServiceQuotaExceededException

Le quota de ressources d'instance a été dépassé pour ce compte.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

CreateTable

Service : Amazon Timestream Write

Ajoute une nouvelle table à une base de données existante de votre compte. Dans un Compte AWS, les noms de table doivent être au moins uniques dans chaque région s'ils se trouvent dans la même base de données. Vous pouvez avoir des noms de table identiques dans la même région si les tables se trouvent dans des bases de données distinctes. Lors de la création de la table, vous devez spécifier le nom de la table, le nom de la base de données et les propriétés de conservation. Des quotas de service s'appliquent. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "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"
```

```
        }
    ]
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte le JSON format des données suivantes.

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 256.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

MagneticStoreWriteProperties

Contient les propriétés à définir sur la table lors de l'activation des écritures sur stockage magnétique.

Type : objet MagneticStoreWriteProperties

Obligatoire : non

RetentionProperties

Durée pendant laquelle vos données chronologiques doivent être stockées dans la mémoire et dans la mémoire magnétique.

Type : objet RetentionProperties

Obligatoire : non

Schema

Schéma de la table.

Type : objet Schema

Obligatoire : non

TableName

Nom de la table Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 256.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Tags

Liste de paires clé-valeur pour étiqueter la table.

Type : tableau d'objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Obligatoire : non

Syntaxe de la réponse

```
{
   "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"
      }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Table

La table Timestream nouvellement créée.

Type : objet Table

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

ConflictException

Timestream n'a pas pu traiter cette demande car elle contient une ressource qui existe déjà.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ServiceQuotaExceededException

Le quota de ressources d'instance a été dépassé pour ce compte.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DeleteDatabase
Service : Amazon Timestream Write

Supprime une base de données Timestream donnée. Il s'agit d'une opération irréversible. Après la suppression d'une base de données, les données chronologiques de ses tables ne peuvent pas être récupérées.

> **Note**
>
> Toutes les tables de la base de données doivent d'abord être supprimées, sinon une ValidationException erreur sera générée.
> En raison de la nature des nouvelles tentatives distribuées, l'opération peut renvoyer soit un succès, soit un ResourceNotFoundException. Les clients doivent les considérer comme équivalents.

Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "DatabaseName": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données Timestream à supprimer.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DeleteTable
Service : Amazon Timestream Write

Supprime une table Timestream donnée. Il s'agit d'une opération irréversible. Après la suppression d'une table de base de données Timestream, les données chronologiques stockées dans la table ne peuvent pas être récupérées.

> **Note**
>
> En raison de la nature des nouvelles tentatives distribuées, l'opération peut renvoyer soit un succès, soit un ResourceNotFoundException. Les clients doivent les considérer comme équivalents.

Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données dans laquelle la base de données Timestream doit être supprimée.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

TableName

Nom de la table Timestream à supprimer.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeBatchLoadTask
Service : Amazon Timestream Write

Renvoie des informations sur la tâche de chargement par lots, notamment les configurations, les mappages, la progression et d'autres détails. Des quotas de service s'appliquent. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

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

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

TaskId

ID de la tâche de chargement par lots.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 32.

Modèle : `[A-Z0-9]+`

Obligatoire : oui

Syntaxe de la réponse

```
{
   "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"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

BatchLoadTaskDescription

Description de la tâche de chargement par lots.

Type : objet BatchLoadTaskDescription

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS

- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeDatabase

Service : Amazon Timestream Write

Renvoie des informations sur la base de données, notamment le nom de la base de données, l'heure à laquelle elle a été créée et le nombre total de tables trouvées dans la base de données. Des quotas de service s'appliquent. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "DatabaseName": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Syntaxe de la réponse

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
```

```
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Database

Nom de la table Timestream.

Type : objet Database

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeEndpoints
Service : Amazon Timestream Write

Renvoie une liste des points de terminaison disponibles pour effectuer des appels TimestreamAPI. Cette API opération est disponible à la fois via Write et QueryAPIs.

Les Timestream étant SDKs conçus pour fonctionner de manière transparente avec l'architecture du service, y compris la gestion et le mappage des points de terminaison du service, nous vous déconseillons d'utiliser cette opération sauf si : API

- Vous utilisez VPCendpoints (AWS PrivateLink) avec Timestream
- Votre application utilise un langage de programmation qui n'est pas encore pris SDK en charge
- Vous avez besoin d'un meilleur contrôle de la mise en œuvre côté client

Pour des informations détaillées sur comment et quand utiliser et implémenter DescribeEndpoints, consultez The Endpoint Discovery Pattern.

Syntaxe de la réponse

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Endpoints

Un `Endpoints` objet est renvoyé lorsqu'une `DescribeEndpoints` demande est faite.

Type : tableau d’objets Endpoint

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeTable

Service : Amazon Timestream Write

Renvoie des informations sur la table, notamment le nom de la table, le nom de la base de données, la durée de conservation de la mémoire et de la mémoire magnétique. Des quotas de service s'appliquent. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

TableName

Nom de la table Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Syntaxe de la réponse

```
{
```

```
   "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"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Table

Le tableau Timestream.

Type : objet Table

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ListBatchLoadTasks
Service : Amazon Timestream Write

Fournit une liste des tâches de chargement par lots, ainsi que leur nom, leur statut, la date à laquelle la tâche peut être reprise et d'autres détails. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "MaxResults": number,
   "NextToken": "string",
   "TaskStatus": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

MaxResults

Le nombre total d'éléments à renvoyer dans la sortie. Si le nombre total d'éléments disponibles est supérieur à la valeur spécifiée, un NextToken est fourni dans la sortie. Pour reprendre la pagination, indiquez la NextToken valeur comme argument d'un appel ultérieurAPI.

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale fixée à 100.

Obligatoire : non

NextToken

Jeton permettant de spécifier où commencer la pagination. C'est le résultat NextToken d'une réponse tronquée précédemment.

Type : chaîne

Obligatoire : non

TaskStatus

État de la tâche de chargement par lots.

Type : String

Valeurs valides : `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Obligatoire : non

Syntaxe de la réponse

```
{
   "BatchLoadTasks": [
      {
         "CreationTime": number,
         "DatabaseName": "string",
         "LastUpdatedTime": number,
         "ResumableUntil": number,
         "TableName": "string",
         "TaskId": "string",
         "TaskStatus": "string"
      }
   ],
   "NextToken": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

BatchLoadTasks

Liste des détails des tâches de chargement par lots.

Type : tableau d’objets BatchLoadTask

NextToken

Jeton permettant de spécifier où commencer la pagination. Fournissez le suivant ListBatchLoadTasksRequest.

Type : String

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET

- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ListDatabases
Service : Amazon Timestream Write

Renvoie la liste de vos bases de données Timestream. Des quotas de service s'appliquent. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

MaxResults

Le nombre total d'éléments à renvoyer dans la sortie. Si le nombre total d'éléments disponibles est supérieur à la valeur spécifiée, un NextToken est fourni dans la sortie. Pour reprendre la pagination, indiquez la NextToken valeur comme argument d'un appel ultérieurAPI.

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale de 20.

Obligatoire : non

NextToken

Le jeton de pagination. Pour reprendre la pagination, indiquez la NextToken valeur comme argument d'un appel ultérieurAPI.

Type : chaîne

Obligatoire : non

Syntaxe de la réponse

```
{
   "Databases": [
      {
         "Arn": "string",
         "CreationTime": number,
         "DatabaseName": "string",
         "KmsKeyId": "string",
         "LastUpdatedTime": number,
         "TableCount": number
      }
   ],
   "NextToken": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Databases

Liste des noms de bases de données.

Type : tableau d'objets Database

NextToken

Le jeton de pagination. Ce paramètre est renvoyé lorsque la réponse est tronquée.

Type : String

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python

- AWS SDKpour Ruby V3

ListTables

Service : Amazon Timestream Write

Fournit une liste de tables, ainsi que le nom, le statut et les propriétés de rétention de chaque table. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "DatabaseName": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : non

MaxResults

Le nombre total d'éléments à renvoyer dans la sortie. Si le nombre total d'éléments disponibles est supérieur à la valeur spécifiée, un NextToken est fourni dans la sortie. Pour reprendre la pagination, indiquez la NextToken valeur comme argument d'un appel ultérieurAPI.

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale de 20.

Obligatoire : non

NextToken

Le jeton de pagination. Pour reprendre la pagination, indiquez la NextToken valeur comme argument d'un appel ultérieurAPI.

Type : chaîne

Obligatoire : non

Syntaxe de la réponse

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

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

NextToken

Jeton permettant de spécifier où commencer la pagination. C'est le résultat NextToken d'une réponse tronquée précédemment.

Type : String

Tables

Une liste de tables.

Type : tableau d’objets Table

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ListTagsForResource
Service : Amazon Timestream Write

Répertorie tous les tags d'une ressource Timestream.

Syntaxe de la requête

```
{
   "ResourceARN": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ResourceARN

La ressource Timestream avec les balises à répertorier. Cette valeur est un nom de ressource Amazon (ARN).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 1011.

Obligatoire : oui

Syntaxe de la réponse

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Tags

Les balises actuellement associées à la ressource Timestream.

Type : tableau d'objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ResumeBatchLoadTask
Service : Amazon Timestream Write

Syntaxe de la requête

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

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

TaskId

ID de la tâche de chargement par lots à reprendre.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 32.

Modèle : `[A-Z0-9]+`

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2

- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

TagResource
Service : Amazon Timestream Write

Associe un ensemble de balises à une ressource Timestream. Vous pouvez ensuite activer ces balises définies par l'utilisateur afin qu'elles apparaissent sur la console Billing and Cost Management pour le suivi de la répartition des coûts.

Syntaxe de la requête

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte le JSON format des données suivantes.

ResourceARN

Identifie la ressource Timestream à laquelle les balises doivent être ajoutées. Cette valeur est un nom de ressource Amazon (ARN).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 1011.

Obligatoire : oui

Tags

Les balises à attribuer à la ressource Timestream.

Type : tableau d’objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ServiceQuotaExceededException

Le quota de ressources d'instance a été dépassé pour ce compte.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

UntagResource
Service : Amazon Timestream Write

Supprime l'association de balises d'une ressource Timestream.

Syntaxe de la requête

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ResourceARN

La ressource Timestream dont les balises seront supprimées. Cette valeur est un nom de ressource Amazon (ARN).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 1011.

Obligatoire : oui

TagKeys

Liste des clés de tags. Les balises existantes de la ressource dont les clés sont membres de cette liste seront supprimées de la ressource Timestream.

Type : tableau de chaînes

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 128.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ServiceQuotaExceededException

Le quota de ressources d'instance a été dépassé pour ce compte.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

UpdateDatabase
Service : Amazon Timestream Write

Modifie la AWS KMS clé d'une base de données existante. Lors de la mise à jour de la base de données, vous devez spécifier le nom de la base de données et l'identifiant de la nouvelle AWS KMS clé à utiliser (`KmsKeyId`). S'il y a des `UpdateDatabase` demandes simultanées, le premier rédacteur gagne.

Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

KmsKeyId

Identifiant de la nouvelle AWS KMS clé (`KmsKeyId`) à utiliser pour chiffrer les données stockées dans la base de données. Si le contenu `KmsKeyId` actuellement enregistré dans la base de données est le même que celui `KmsKeyId` indiqué dans la demande, aucune mise à jour ne sera effectuée.

Vous pouvez le spécifier à l'`KmsKeyId`aide de l'une des options suivantes :

- ID de clé : `1234abcd-12ab-34cd-56ef-1234567890ab`
- Clé ARN : `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nom d'alias : `alias/ExampleAlias`
- Pseudonyme ARN : `arn:aws:kms:us-east-1:111122223333:alias/ExampleAlias`

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Syntaxe de la réponse

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Database

Un conteneur de haut niveau pour une table. Les bases de données et les tables sont les concepts de gestion fondamentaux d'Amazon Timestream. Toutes les tables d'une base de données sont chiffrées avec la même AWS KMS clé.

Type : objet Database

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ServiceQuotaExceededException

Le quota de ressources d'instance a été dépassé pour ce compte.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

UpdateTable

Service : Amazon Timestream Write

Modifie la durée de conservation de la mémoire et de la mémoire magnétique de votre table Timestream. Notez que la modification de la durée de conservation prend effet immédiatement. Par exemple, si la période de conservation de la mémoire a été initialement fixée à 2 heures, puis modifiée à 24 heures, la mémoire sera capable de contenir 24 heures de données, mais sera remplie de 24 heures de données 22 heures après cette modification. Timestream ne récupère pas les données de la mémoire magnétique pour alimenter la mémoire.

Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "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"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

MagneticStoreWriteProperties

Contient les propriétés à définir sur la table lors de l'activation des écritures sur stockage magnétique.

Type : objet MagneticStoreWriteProperties

Obligatoire : non

RetentionProperties

Durée de conservation de la mémoire et de la mémoire magnétique.

Type : objet RetentionProperties

Obligatoire : non

Schema

Schéma de la table.

Type : objet Schema

Obligatoire : non

TableName

Nom de la table Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Syntaxe de la réponse

```
{
   "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"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Table

La table Timestream mise à jour.

Type : objet Table

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

WriteRecords
Service : Amazon Timestream Write

Vous permet d'écrire les données de vos séries chronologiques dans Timestream. Vous pouvez spécifier un point de données unique ou un lot de points de données à insérer dans le système. Timestream vous propose un schéma flexible qui détecte automatiquement les noms de colonnes et les types de données de vos tables Timestream en fonction des noms de dimension et des types de données des points de données que vous spécifiez lorsque vous appelez des écritures dans la base de données.

Timestream prend en charge la cohérence éventuelle de la sémantique de lecture. Cela signifie que lorsque vous interrogez des données immédiatement après avoir écrit un lot de données dans Timestream, les résultats de la requête peuvent ne pas refléter les résultats d'une opération d'écriture récemment terminée. Les résultats peuvent également inclure des données périmées. Si vous répétez la requête après un court laps de temps, les résultats devraient renvoyer les données les plus récentes. Des quotas de service s'appliquent.

Voir l'exemple de code pour plus de détails.

Upserts

Vous pouvez utiliser le `Version` paramètre dans une `WriteRecords` demande pour mettre à jour des points de données. Timestream enregistre un numéro de version pour chaque enregistrement. `Version`par défaut, c'est `1` quand il n'est pas spécifié pour l'enregistrement dans la demande. Timestream met à jour la valeur de mesure d'un enregistrement existant ainsi que sa valeur `Version` lorsqu'il reçoit une demande d'écriture avec un `Version` nombre plus élevé pour cet enregistrement. Lorsqu'il reçoit une demande de mise à jour dont la valeur de mesure est identique à celle de l'enregistrement existant, Timestream est toujours mis à jour`Version`, si elle est supérieure à la valeur existante de. `Version` Vous pouvez mettre à jour un point de données autant de fois que vous le souhaitez, à condition que la valeur de augmente `Version` continuellement.

Supposons, par exemple, que vous écriviez un nouvel enregistrement sans l'indiquer `Version` dans la demande. Timestream enregistre cet enregistrement et le définit sur`Version`. `1` Supposons maintenant que vous essayiez de mettre à jour cet enregistrement avec une `WriteRecords` demande du même enregistrement avec une valeur de mesure différente mais que, comme auparavant, vous ne la fournissez pas`Version`. Dans ce cas, Timestream rejettera cette mise à jour avec un `RejectedRecordsException` car la version de l'enregistrement mis à jour n'est pas supérieure à la valeur existante de Version.

Toutefois, si vous renvoyez la demande de mise à jour avec `Version` défini sur2, Timestream réussira alors à mettre à jour la valeur de l'enregistrement, qui `Version` sera définie sur. 2 Supposons ensuite que vous ayez envoyé une `WriteRecords` demande avec le même enregistrement et une valeur de mesure identique, mais avec la valeur `Version` définie sur3. Dans ce cas, Timestream sera uniquement mis à jour `Version` vers. 3 Toute mise à jour ultérieure devra envoyer un numéro de version supérieur à3, sinon les demandes de mise à jour recevront un`RejectedRecordsException`.

Syntaxe de la requête

```
{
   "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"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

CommonAttributes

Un enregistrement qui contient les attributs communs de mesure, de dimension, de temps et de version partagés entre tous les enregistrements de la demande. Les attributs de mesure et de dimension spécifiés seront fusionnés avec les attributs de mesure et de dimension dans l'objet records lorsque les données seront écrites dans Timestream. Les dimensions peuvent ne pas se chevaucher, sinon un `ValidationException` sera projeté. En d'autres termes, un enregistrement doit contenir des dimensions portant des noms uniques.

Type : objet Record

Obligatoire : non

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Records

Tableau d'enregistrements contenant les attributs uniques de mesure, de dimension, de temps et de version pour chaque point de données de série chronologique.

Type : tableau d'objets Record

Membres du tableau : Nombre minimum de 1 élément. Nombre maximal de 100 éléments.

Obligatoire : oui

TableName

Nom de la table Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : oui

Syntaxe de la réponse

```
{
   "RecordsIngested": {
      "MagneticStore": number,
      "MemoryStore": number,
      "Total": number
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

RecordsIngested

Informations sur les enregistrements ingérés par cette demande.

Type : objet RecordsIngested

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Timestream n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 500

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

RejectedRecordsException

WriteRecords déclencherait cette exception dans les cas suivants :

- Enregistrements contenant des données dupliquées contenant plusieurs enregistrements ayant les mêmes dimensions, horodatages et noms de mesures, mais :
  - Les valeurs de mesure sont différentes
  - La version n'est pas présente dans la demande ou la valeur de la version dans le nouvel enregistrement est égale ou inférieure à la valeur existante

  Dans ce cas, si Timestream rejette les données, le `ExistingVersion` champ de `RejectedRecords` réponse indiquera la version de l'enregistrement en cours. Pour forcer une mise à jour, vous pouvez renvoyer la demande avec une version de l'enregistrement définie sur une valeur supérieure à. `ExistingVersion`
- Enregistrements dont l'horodatage se situe en dehors de la durée de conservation de la mémoire.

- Enregistrements dont les dimensions ou les mesures dépassent les limites définies par Timestream.

Pour plus d'informations, consultez la section Quotas dans le guide du développeur Amazon Timestream.

HTTPCode de statut : 400

ResourceNotFoundException

L'opération a tenté d'accéder à une ressource inexistante. La ressource n'est peut-être pas spécifiée correctement ou son statut ne l'est peut-être pasACTIVE.

HTTPCode de statut : 400

ThrottlingException

Trop de demandes ont été effectuées par un utilisateur et elles ont dépassé les quotas de service. La demande a été limitée.

HTTPCode de statut : 400

ValidationException

Une demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python

- AWS SDKpour Ruby V3

Requête Amazon Timestream

Les actions suivantes sont prises en charge par Amazon Timestream Query :

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

CancelQuery
Service : Amazon Timestream Query

Annule une requête qui a été émise. L'annulation n'est accordée que si l'exécution de la requête n'est pas terminée avant l'émission de la demande d'annulation. L'annulation étant une opération idempotente, les demandes d'annulation suivantes renverront un`CancellationMessage`, indiquant que la requête a déjà été annulée. Voir l'exemple de code pour plus de détails.

Syntaxe de la requête

```
{
   "QueryId": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

QueryId

L'ID de la requête qui doit être annulée. `QueryID`est renvoyé dans le cadre du résultat de la requête.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 64.

Modèle : `[a-zA-Z0-9]+`

Obligatoire : oui

Syntaxe de la réponse

```
{
   "CancellationMessage": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

CancellationMessage

A `CancellationMessage` est renvoyé lorsqu'une `CancelQuery` demande pour la requête spécifiée par `QueryId` a déjà été émise.

Type : String

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

CreateScheduledQuery

Service : Amazon Timestream Query

Créez une requête planifiée qui sera exécutée en votre nom selon la planification configurée. Timestream endosse le rôle d'exécution fourni dans le cadre du paramètre `ScheduledQueryExecutionRoleArn` pour exécuter la requête. Vous pouvez utiliser le paramètre `NotificationConfiguration` pour configurer la notification pour vos opérations de requêtes planifiées.

Syntaxe de la requête

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

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ClientToken

L'utilisation de a ClientToken rend l'appel à CreateScheduledQuery idempotent, en d'autres termes, le fait de faire la même demande à plusieurs reprises produira le même résultat. Le fait de faire plusieurs CreateScheduledQuery demandes identiques a le même effet qu'une seule demande.

- Si elle CreateScheduledQuery est appelée sans un`ClientToken`, la requête SDK génère un `ClientToken` en votre nom.
- Après 8 heures, toute demande avec le même `ClientToken` est considérée comme une nouvelle demande.

Type : String

Contraintes de longueur : longueur minimale de 32. Longueur maximale de 128.

Obligatoire : non

ErrorReportConfiguration

Configuration pour le signalement d'erreurs. Des rapports d'erreurs seront générés lorsqu'un problème est rencontré au moment de l'écriture des résultats de la requête.

Type : objet ErrorReportConfiguration

Obligatoire : oui

KmsKeyId

La KMS clé Amazon utilisée pour chiffrer la ressource de requête planifiée, au repos. Si la KMS clé Amazon n'est pas spécifiée, la ressource de requête planifiée sera chiffrée avec une clé Amazon KMS appartenant à Timestream. Pour spécifier une KMS clé, utilisez son identifiant, sa cléARN, son nom d'alias ou son aliasARN. Lorsque vous utilisez un nom d'alias, préfixez-le avec alias/.

S'il est ErrorReportConfiguration utilisé `SSE_KMS` comme type de chiffrement, le même KmsKeyId est utilisé pour chiffrer le rapport d'erreur au repos.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

Name

Nom de la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 64.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : oui

NotificationConfiguration

Configuration de la notification pour la requête planifiée. Une notification est envoyée par Timestream lorsque l'exécution d'une requête se termine, lorsque l'état est mis à jour ou lorsque vous la supprimez.

Type : objet NotificationConfiguration

Obligatoire : oui

QueryString

La chaîne de requête à exécuter. Les noms de paramètres peuvent être spécifiés dans le caractère `@` de la chaîne de requête suivi d'un identifiant. Le paramètre nommé `@scheduled_runtime` est réservé et peut être utilisé dans la requête pour obtenir l'heure à laquelle l'exécution de la requête est planifiée.

L'horodatage calculé en fonction du ScheduleConfiguration paramètre sera la valeur du `@scheduled_runtime` paramètre pour chaque requête exécutée. Par exemple, imaginez une instance d'une requête planifiée s'exécutant le 01/12/2021 à minuit. Pour cette instance, le paramètre `@scheduled_runtime` est initialisé à l'horodatage 01/12/2021 à minuit lors de l'appel de la requête.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximum de 262 144.

Obligatoire : oui

ScheduleConfiguration

Configuration du planning pour la requête.

Type : objet ScheduleConfiguration

Obligatoire : oui

ScheduledQueryExecutionRoleArn

ARNPour le IAM rôle que Timestream assumera lors de l'exécution de la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Tags

Une liste de paires clé-valeur pour étiqueter la requête planifiée.

Type : tableau d'objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Obligatoire : non

TargetConfiguration

Configuration utilisée pour écrire le résultat d'une requête.

Type : objet TargetConfiguration

Obligatoire : non

Syntaxe de la réponse

```
{
   "Arn": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Arn

ARNpour la requête planifiée créée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

ConflictException

Impossible d'interroger les résultats d'une requête annulée.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ServiceQuotaExceededException

Vous avez dépassé le quota de service.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DeleteScheduledQuery
Service : Amazon Timestream Query

Supprime une requête planifiée donnée. Il s'agit d'une opération irréversible.

Syntaxe de la requête

```
{
   "ScheduledQueryArn": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ScheduledQueryArn

L'ARN de la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3

- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeAccountSettings
Service : Amazon Timestream Query

Décrit les paramètres de votre compte, notamment le modèle de tarification des requêtes et le maximum configuré que TCUs le service peut utiliser pour votre charge de travail de requêtes.

Vous êtes facturé uniquement pour la durée des unités de calcul utilisées pour vos charges de travail.

Syntaxe de la réponse

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

MaxQueryTCU

Le nombre maximum d'unités de calcul Timestream (TCUs) que le service utilisera à tout moment pour répondre à vos requêtes.

Type : entier

QueryPricingModel

Le modèle de tarification pour les requêtes relatives à votre compte.

Type : String

Valeurs valides : `BYTES_SCANNED | COMPUTE_UNITS`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeEndpoints
Service : Amazon Timestream Query

DescribeEndpoints renvoie une liste des points de terminaison disponibles pour effectuer des appels TimestreamAPI. Ceci API est disponible à la fois via Write et Query.

Étant donné que les Timestream SDKs sont conçus pour fonctionner de manière transparente avec l'architecture du service, y compris la gestion et le mappage des points de terminaison du service, il n'est pas recommandé de les utiliser sauf si : API

- Vous utilisez VPCendpoints (AWS PrivateLink) avec Timestream
- Votre application utilise un langage de programmation qui n'est pas encore pris SDK en charge
- Vous avez besoin d'un meilleur contrôle de la mise en œuvre côté client

Pour des informations détaillées sur comment et quand utiliser et implémenter DescribeEndpoints, consultez The Endpoint Discovery Pattern.

Syntaxe de la réponse

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Endpoints

Un `Endpoints` objet est renvoyé lorsqu'une `DescribeEndpoints` demande est faite.

Type : tableau d’objets Endpoint

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

DescribeScheduledQuery
Service : Amazon Timestream Query

Fournit des informations détaillées sur une requête planifiée.

Syntaxe de la requête

```
{
   "ScheduledQueryArn": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ScheduledQueryArn

L'ARN de la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Syntaxe de la réponse

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

```
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

ScheduledQuery

La requête planifiée.

Type : objet ScheduledQueryDescription

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ExecuteScheduledQuery
Service : Amazon Timestream Query

Vous pouvez l'utiliser API pour exécuter une requête planifiée manuellement.

Si vous l'avez activé`QueryInsights`, cela renvoie API également des informations et des statistiques relatives à la requête que vous avez exécutée dans le cadre d'une SNS notification Amazon. `QueryInsights`aide à optimiser les performances de votre requête. Pour en savoir plus`QueryInsights`, consultez la section Utilisation des informations relatives aux requêtes pour optimiser les requêtes dans Amazon Timestream.

Syntaxe de la requête

```
{
   "ClientToken": "string",
   "InvocationTime": number,
   "QueryInsights": {
      "Mode": "string"
   },
   "ScheduledQueryArn": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ClientToken

Non utilisé.

Type : String

Contraintes de longueur : longueur minimale de 32. Longueur maximale de 128.

Obligatoire : non

InvocationTime

L'horodatage est entré. UTC La requête sera exécutée comme si elle avait été invoquée à cet horodatage.

Type : Timestamp

Obligatoire : oui

QueryInsights

Encapsule les paramètres à activer`QueryInsights`.

Activation des informations et des statistiques sur les `QueryInsights` retours dans le cadre de la SNS notification Amazon relative à la requête que vous avez exécutée. Vous pouvez l'utiliser `QueryInsights` pour ajuster les performances et le coût de vos requêtes.

Type : objet ScheduledQueryInsights

Obligatoire : non

ScheduledQueryArn

ARNde la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

Exemples

Message de notification de requête planifiée pour le CONTROL mode ENABLED WITH _ RATE _ _

L'exemple suivant montre un message de notification de requête planifiée réussie pour le `ENABLED_WITH_RATE_CONTROL` mode du `QueryInsights` paramètre.

```
"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
        }
    }
}
```

Message de notification de requête planifiée pour le DISABLED mode

L'exemple suivant montre un message de notification de requête planifiée réussie pour le `DISABLED` mode du `QueryInsights` paramètre.

```
"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
        }
    }
}
```

Message de notification d'échec pour le CONTROL mode ENABLED WITH _ RATE _ _

L'exemple suivant montre un message de notification de requête planifiée ayant échoué pour le `ENABLED_WITH_RATE_CONTROL` mode du `QueryInsights` paramètre.

```
"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"
    }
}
```

Message de notification d'échec pour le DISABLED mode

L'exemple suivant montre un message de notification de requête planifiée ayant échoué pour le `DISABLED` mode du `QueryInsights` paramètre.

```
"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"
    }
}
```

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2

- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ListScheduledQueries
Service : Amazon Timestream Query

Obtient une liste de toutes les requêtes planifiées dans le compte Amazon et la région de l'appelant. `ListScheduledQueries`est finalement cohérent.

Syntaxe de la requête

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

MaxResults

Le nombre maximum d'éléments à renvoyer dans la sortie. Si le nombre total d'éléments disponibles est supérieur à la valeur spécifiée, un `NextToken` est fourni dans la sortie. Pour reprendre la pagination, fournissez la `NextToken` valeur comme argument de l'appel suivant à`ListScheduledQueriesRequest`.

Type : entier

Plage valide : valeur minimum de 1. La valeur maximale est 1 000.

Obligatoire : non

NextToken

Un jeton de pagination pour reprendre la pagination.

Type : chaîne

Obligatoire : non

Syntaxe de la réponse

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

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

NextToken

Jeton permettant de spécifier où commencer la pagination. Il s'agit du NextToken résultat d'une réponse tronquée précédemment.

Type : String

ScheduledQueries

Liste de requêtes planifiées.

Type : tableau d'objets ScheduledQuery

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

ListTagsForResource
Service : Amazon Timestream Query

Répertoriez tous les tags d'une ressource de requête Timestream.

Syntaxe de la requête

```
{
   "MaxResults": number,
   "NextToken": "string",
   "ResourceARN": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

MaxResults

Le nombre maximum de balises à renvoyer.

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale fixée à 200.

Obligatoire : non

NextToken

Un jeton de pagination pour reprendre la pagination.

Type : chaîne

Obligatoire : non

ResourceARN

La ressource Timestream avec les balises à répertorier. Cette valeur est un nom de ressource Amazon (ARN).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Syntaxe de la réponse

```
{
   "NextToken": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

NextToken

Un jeton de pagination permettant de reprendre la pagination avec un appel ultérieur à. `ListTagsForResourceResponse`

Type : String

Tags

Les balises actuellement associées à la ressource Timestream.

Type : tableau d'objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

PrepareQuery
Service : Amazon Timestream Query

Opération synchrone qui vous permet de soumettre une requête avec des paramètres à stocker par Timestream pour une exécution ultérieure. Timestream ne prend en charge l'utilisation de cette opération que si elle est `ValidateOnly` définie sur. `true`

Syntaxe de la requête

```
{
   "QueryString": "string",
   "ValidateOnly": boolean
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

QueryString

Chaîne de requête Timestream que vous souhaitez utiliser comme instruction préparée. Les noms de paramètres peuvent être spécifiés dans le caractère @ de la chaîne de requête suivi d'un identifiant.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximum de 262 144.

Obligatoire : oui

ValidateOnly

En définissant cette valeur sur`true`, Timestream validera uniquement que la chaîne de requête est une requête Timestream valide, et ne stockera pas la requête préparée pour une utilisation ultérieure.

Type : booléen

Obligatoire : non

Syntaxe de la réponse

```
{
   "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"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

Columns

Liste des colonnes de SELECT clauses de la chaîne de requête soumise.

Type : tableau d'objets SelectColumn

Parameters

Liste des paramètres utilisés dans la chaîne de requête soumise.

Type : tableau d'objets ParameterMapping

QueryString

Chaîne de requête que vous souhaitez préparer.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximum de 262 144.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

Query
Service : Amazon Timestream Query

`Query`est une opération synchrone qui vous permet d'exécuter une requête sur vos données Amazon Timestream.

Si vous l'avez activé`QueryInsights`, cela renvoie API également des informations et des mesures relatives à la requête que vous avez exécutée. `QueryInsights`saide à optimiser les performances de votre requête. Pour en savoir plus`QueryInsights`, consultez la section Utilisation des informations relatives aux requêtes pour optimiser les requêtes dans Amazon Timestream.

> **Note**
>
> Le nombre maximum de `Query` API demandes que vous êtes autorisé à effectuer lorsque `QueryInsights` cette option est activée est d'une requête par seconde (QPS). Si vous dépassez ce taux de requêtes, cela peut entraîner un ralentissement.

`Query`expirera au bout de 60 secondes. Vous devez mettre à jour le délai d'expiration par défaut dans le SDK pour prendre en charge un délai d'attente de 60 secondes. Consultez l'exemple de code pour plus de détails.

Votre demande de requête échouera dans les cas suivants :

- Si vous soumettez une `Query` demande avec le même jeton client en dehors de la fenêtre d'idempuissance de 5 minutes.
- Si vous soumettez une `Query` demande avec le même jeton client, mais que vous modifiez d'autres paramètres, dans le délai d'idempotentie de 5 minutes.
- Si la taille de la ligne (y compris les métadonnées de la requête) dépasse 1 Mo, la requête échouera avec le message d'erreur suivant :

  ```
  Query aborted as max page response size has been exceeded by the output result row
  ```

- Si le IAM principal de l'initiateur de requête et le lecteur de résultats ne sont pas identiques et/ou si l'initiateur de requête et le lecteur de résultats n'ont pas la même chaîne de requête dans les demandes de requête, la requête échouera avec une `Invalid pagination token` erreur.

Syntaxe de la requête

```
{
   "ClientToken": "string",
   "MaxRows": number,
   "NextToken": "string",
   "QueryInsights": {
      "Mode": "string"
   },
   "QueryString": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte le JSON format des données suivantes.

ClientToken

Chaîne unique distinguant majuscules et minuscules de 64 ASCII caractères maximum spécifiée lorsqu'une `Query` demande est faite. Fournir un `ClientToken` fait passer l'appel à `Query` idempotent. Cela signifie que l'exécution répétée de la même requête produira le même résultat. En d'autres termes, le fait de faire plusieurs `Query` demandes identiques a le même effet qu'une seule demande. Lorsque vous l'utilisez `ClientToken` dans une requête, notez les points suivants :

- Si la requête API est instanciée sans a`ClientToken`, elle en SDK génère un en votre `ClientToken` nom.
- Si l'`Query`invocation ne contient que le `ClientToken` mais n'inclut pas un`NextToken`, cet appel de `Query` est supposé être une nouvelle requête exécutée.
- Si l'invocation contient`NextToken`, cette invocation particulière est supposée être une invocation ultérieure d'un appel antérieur à la requêteAPI, et un ensemble de résultats est renvoyé.
- Après 4 heures, toute demande contenant la même chose `ClientToken` est traitée comme une nouvelle demande.

Type : String

Contraintes de longueur : longueur minimale de 32. Longueur maximale de 128.

Obligatoire : non

MaxRows

Le nombre total de lignes à renvoyer dans la `Query` sortie. L'exécution initiale de `Query` avec une `MaxRows` valeur spécifiée renverra le jeu de résultats de la requête dans deux cas :

- La taille du résultat est inférieure à`1MB`.
- Le nombre de lignes du jeu de résultats est inférieur à la valeur de`maxRows`.

Sinon, l'appel initial de renvoie `Query` uniquement a`NextToken`, qui peut ensuite être utilisé lors des appels suivants pour récupérer le jeu de résultats. Pour reprendre la pagination, entrez la `NextToken` valeur dans la commande suivante.

Si la taille de la ligne est grande (par exemple, une ligne comporte de nombreuses colonnes), Timestream peut renvoyer moins de lignes afin d'empêcher la taille de réponse de dépasser la limite de 1 Mo. Si `MaxRows` ce n'est pas le cas, Timestream enverra le nombre de lignes nécessaire pour respecter la limite de 1 Mo.

Type : entier

Plage valide : valeur minimum de 1. La valeur maximale est 1 000.

Obligatoire : non

NextToken

Un jeton de pagination utilisé pour renvoyer un ensemble de résultats. Lorsque le `Query` API est invoqué en utilisant`NextToken`, cet appel particulier est supposé être une invocation ultérieure d'un appel précédent à`Query`, et un ensemble de résultats est renvoyé. Toutefois, si l'`Query`invocation ne contient que le`ClientToken`, cette invocation de `Query` est supposée être une nouvelle requête exécutée.

Notez les points suivants lors de l'utilisation NextToken dans une requête :

- Un jeton de pagination peut être utilisé pour un maximum de cinq `Query` invocations, OU pour une durée maximale d'une heure, selon la première éventualité.
- L'utilisation de la même option `NextToken` renverra le même ensemble d'enregistrements. Pour continuer à paginer dans le jeu de résultats, vous devez utiliser le plus récent. `nextToken`
- Supposons qu'`Query`un appel renvoie deux `NextToken` valeurs, `TokenA` et`TokenB`. S'il `TokenB` est utilisé lors d'un appel ultérieur, `Query` il `TokenA` est alors invalidé et ne peut pas être réutilisé.

- Pour demander un ensemble de résultats antérieur à partir d'une requête après le début de la pagination, vous devez réinvoquer la requête. API
- La dernière `NextToken` doit être utilisée pour paginer jusqu'à ce qu'elle `null` soit renvoyée, après quoi une nouvelle `NextToken` doit être utilisée.
- Si le IAM principal de l'initiateur de requête et le lecteur de résultats ne sont pas identiques et/ou si l'initiateur de requête et le lecteur de résultats n'ont pas la même chaîne de requête dans les demandes de requête, la requête échouera avec une `Invalid pagination token` erreur.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

QueryInsights

Encapsule les paramètres à activer`QueryInsights`.

L'activation `QueryInsights` renvoie des informations et des métriques en plus des résultats de requête pour la requête que vous avez exécutée. Vous pouvez l'utiliser `QueryInsights` pour optimiser les performances de vos requêtes.

Type : objet QueryInsights

Obligatoire : non

QueryString

Requête à exécuter par Timestream.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximum de 262 144.

Obligatoire : oui

Syntaxe de la réponse

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

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

ColumnInfo

Les types de données de colonne du jeu de résultats renvoyé.

Type : tableau d’objets ColumnInfo

NextToken

Un jeton de pagination qui peut être réutilisé lors d'un `Query` appel pour obtenir la prochaine série de résultats.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

QueryId

Un identifiant unique pour la requête donnée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 64.

Modèle : `[a-zA-Z0-9]+`

QueryInsightsResponse

Encapsule `QueryInsights` contenant des informations et des mesures relatives à la requête que vous avez exécutée.

Type : objet QueryInsightsResponse

QueryStatus

Informations sur l'état de la requête, y compris la progression et les octets analysés.

Type : objet QueryStatus

Rows

Les lignes du jeu de résultats renvoyées par la requête.

Type : tableau d’objets Row

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

ConflictException

Impossible d'interroger les résultats d'une requête annulée.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

QueryExecutionException

Timestream n'a pas pu exécuter correctement la requête.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

TagResource
Service : Amazon Timestream Query

Associez un ensemble de balises à une ressource Timestream. Vous pouvez ensuite activer ces balises définies par l'utilisateur afin qu'elles apparaissent sur la console Billing and Cost Management pour le suivi de la répartition des coûts.

Syntaxe de la requête

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ResourceARN

Identifie la ressource Timestream à laquelle les balises doivent être ajoutées. Cette valeur est un nom de ressource Amazon (ARN).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Tags

Les balises à attribuer à la ressource Timestream.

Type : tableau d’objets Tag

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ServiceQuotaExceededException

Vous avez dépassé le quota de service.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

UntagResource
Service : Amazon Timestream Query

Supprime l'association de balises d'une ressource de requête Timestream.

Syntaxe de la requête

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ResourceARN

La ressource Timestream dont les balises seront supprimées. Cette valeur est un nom de ressource Amazon (ARN).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

TagKeys

Une liste de clés de tags. Les balises existantes de la ressource dont les clés sont membres de cette liste seront supprimées de la ressource Timestream.

Type : tableau de chaînes

Membres du tableau : nombre minimum de 0 élément. Nombre maximum de 200 éléments.

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 128.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2

- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

UpdateAccountSettings
Service : Amazon Timestream Query

Transfère votre compte pour l'utiliser TCUs pour la tarification des requêtes et modifie le nombre maximal d'unités de calcul des requêtes que vous avez configurées. Si vous réduisez la valeur de `MaxQueryTCU` à la configuration souhaitée, la nouvelle valeur peut prendre jusqu'à 24 heures pour être effective.

> **Note**
>
> Une fois que vous avez transféré votre compte à des TCUs fins de tarification des requêtes, vous ne pouvez plus passer à l'utilisation d'octets scannés pour la tarification des requêtes.

Syntaxe de la requête

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

MaxQueryTCU

Le nombre maximum d'unités de calcul que le service utilisera à tout moment pour répondre à vos requêtes. Pour exécuter des requêtes, vous devez définir une capacité minimale de 4TCU. Vous pouvez définir le nombre maximum TCU de multiples de 4, par exemple 4, 8, 16, 32, etc.

La valeur maximale prise en charge `MaxQueryTCU` est de 1 000. Pour demander une augmentation de cette limite souple, contactez le AWS Support. Pour plus d'informations sur le quota par défaut pour maxQueryTCU, consultez la section Quotas par défaut.

Type : entier

Obligatoire : non

QueryPricingModel

Le modèle de tarification pour les requêtes relatives à un compte.

> **Note**
>
> Le `QueryPricingModel` paramètre est utilisé par plusieurs opérations Timestream ; toutefois, l'`UpdateAccountSettings`APIopération ne reconnaît aucune valeur autre que. `COMPUTE_UNITS`

Type : String

Valeurs valides : `BYTES_SCANNED | COMPUTE_UNITS`

Obligatoire : non

Syntaxe de la réponse

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées sous JSON forme formatée par le service.

MaxQueryTCU

Le nombre maximal configuré d'unités de calcul que le service utilisera à tout moment pour répondre à vos requêtes.

Type : entier

QueryPricingModel

Le modèle de tarification d'un compte.

Type : String

Valeurs valides : `BYTES_SCANNED | COMPUTE_UNITS`

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS

- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

UpdateScheduledQuery
Service : Amazon Timestream Query

Mettez à jour une requête planifiée.

Syntaxe de la requête

```
{
   "ScheduledQueryArn": "string",
   "State": "string"
}
```

Paramètres de demande

Pour plus d'informations sur les paramètres courants pour toutes les actions, consultez Paramètres courants.

La demande accepte les données suivantes au JSON format suivant.

ScheduledQueryArn

ARNde la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

State

État de la requête planifiée.

Type : String

Valeurs valides : `ENABLED | DISABLED`

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un HTTP corps vide.

Erreurs

Pour plus d'informations sur les erreurs courantes pour toutes les actions, consultez Erreurs courantes.

AccessDeniedException

Vous n'êtes pas autorisé à effectuer cette action.

HTTPCode de statut : 400

InternalServerException

Le service n'a pas pu traiter complètement cette demande en raison d'une erreur interne du serveur.

HTTPCode de statut : 400

InvalidEndpointException

Le point de terminaison demandé n'était pas valide.

HTTPCode de statut : 400

ResourceNotFoundException

La ressource demandée est introuvable.

HTTPCode de statut : 400

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationException

Demande non valide ou mal formée.

HTTPCode de statut : 400

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- Interface de ligne de commande AWS
- AWS SDKpour. NET
- AWS SDKpour C++
- AWS SDKpour Go v2
- AWS SDKpour Java V2
- AWS SDKpour JavaScript V3
- AWS SDKpour PHP V3
- AWS SDKpour Python
- AWS SDKpour Ruby V3

Types de données

Les types de données suivants sont pris en charge par 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

Les types de données suivants sont pris en charge par Amazon Timestream Query :

- ColumnInfo
- Datum
- DimensionMapping
- Endpoint
- ErrorReportConfiguration
- ErrorReportLocation
- ExecutionStats
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- NotificationConfiguration
- ParameterMapping
- QueryInsights
- QueryInsightsResponse
- QuerySpatialCoverage
- QuerySpatialCoverageMax

- QueryStatus
- QueryTemporalRange
- QueryTemporalRangeMax
- Row
- S3Configuration
- S3ReportLocation
- ScheduleConfiguration
- ScheduledQuery
- ScheduledQueryDescription
- ScheduledQueryInsights
- ScheduledQueryInsightsResponse
- ScheduledQueryRunSummary
- SelectColumn
- SnsConfiguration
- Tag
- TargetConfiguration
- TargetDestination
- TimeSeriesDataPoint
- TimestreamConfiguration
- TimestreamDestination
- Type

Amazon Timestream Write

Les types de données suivants sont pris en charge par 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
Service : Amazon Timestream Write

Informations sur la progression d'une tâche de chargement par lots.

Table des matières

BytesMetered

Type : long

Obligatoire : non

FileFailures

Type : long

Obligatoire : non

ParseFailures

Type : long

Obligatoire : non

RecordIngestionFailures

Type : long

Obligatoire : non

RecordsIngested

Type : long

Obligatoire : non

RecordsProcessed

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

BatchLoadTask

Service : Amazon Timestream Write

Détails relatifs à une tâche de chargement par lots.

Table des matières

CreationTime

Heure à laquelle la tâche de chargement par lots Timestream a été créée.

Type : Timestamp

Obligatoire : non

DatabaseName

Nom de base de données pour la base de données dans laquelle une tâche de chargement par lots charge des données.

Type : chaîne

Obligatoire : non

LastUpdatedTime

Heure à laquelle la tâche de chargement par lots Timestream a été mise à jour pour la dernière fois.

Type : Timestamp

Obligatoire : non

ResumableUntil

Type : Timestamp

Obligatoire : non

TableName

Nom de la table dans laquelle une tâche de chargement par lots charge des données.

Type : chaîne

Obligatoire : non

TaskId

ID de la tâche de chargement par lots.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 32.

Modèle : `[A-Z0-9]+`

Obligatoire : non

TaskStatus

État de la tâche de chargement par lots.

Type : String

Valeurs valides : `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

BatchLoadTaskDescription
Service : Amazon Timestream Write

Détails relatifs à une tâche de chargement par lots.

Table des matières

CreationTime

Heure à laquelle la tâche de chargement par lots Timestream a été créée.

Type : Timestamp

Obligatoire : non

DataModelConfiguration

Configuration du modèle de données pour une tâche de chargement par lots. Il contient des informations sur l'emplacement de stockage d'un modèle de données pour une tâche de chargement par lots.

Type : objet DataModelConfiguration

Obligatoire : non

DataSourceConfiguration

Détails de configuration concernant la source de données pour une tâche de chargement par lots.

Type : objet DataSourceConfiguration

Obligatoire : non

ErrorMessage

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

LastUpdatedTime

Heure à laquelle la tâche de chargement par lots Timestream a été mise à jour pour la dernière fois.

Type : Timestamp

Obligatoire : non

ProgressReport

Type : objet BatchLoadProgressReport

Obligatoire : non

RecordVersion

Type : long

Obligatoire : non

ReportConfiguration

Configuration du rapport pour une tâche de chargement par lots. Il contient des informations sur l'emplacement de stockage des rapports d'erreur.

Type : objet ReportConfiguration

Obligatoire : non

ResumableUntil

Type : Timestamp

Obligatoire : non

TargetDatabaseName

Type : chaîne

Obligatoire : non

TargetTableName

Type : chaîne

Obligatoire : non

TaskId

ID de la tâche de chargement par lots.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 32.

Modèle : `[A-Z0-9]+`

Obligatoire : non

TaskStatus

État de la tâche de chargement par lots.

Type : String

Valeurs valides : `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

CsvConfiguration
Service : Amazon Timestream Write

Format de données délimité dans lequel le séparateur de colonnes peut être une virgule et le séparateur d'enregistrements un caractère de nouvelle ligne.

Table des matières

ColumnSeparator

Le séparateur de colonnes peut être une virgule (','), un tube ('|), un point-virgule (';'), un tabulation ('/t') ou un espace vide (").

Type : String

Contraintes de longueur : longueur fixe de 1.

Obligatoire : non

EscapeChar

Le personnage d'évasion peut être l'un des

Type : String

Contraintes de longueur : longueur fixe de 1.

Obligatoire : non

NullValue

Il peut s'agir d'un espace vide (« »).

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 256.

Obligatoire : non

QuoteChar

Il peut s'agir d'un guillemet simple (') ou d'un guillemet double («).

Type : String

Contraintes de longueur : longueur fixe de 1.

Obligatoire : non

TrimWhiteSpace

Spécifie de couper les espaces blancs de début et de fin.

Type : booléen

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Database
Service : Amazon Timestream Write

Un conteneur de haut niveau pour une table. Les bases de données et les tables sont les concepts de gestion fondamentaux d'Amazon Timestream. Toutes les tables d'une base de données sont chiffrées avec la même AWS KMS clé.

Table des matières

Arn

Le nom de la ressource Amazon qui identifie de manière unique cette base de données.

Type : chaîne

Obligatoire : non

CreationTime

Heure à laquelle la base de données a été créée, calculée à partir de l'époque Unix.

Type : Timestamp

Obligatoire : non

DatabaseName

Nom de la base de données Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : non

KmsKeyId

Identifiant de la AWS KMS clé utilisée pour chiffrer les données stockées dans la base de données.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

LastUpdatedTime

Dernière mise à jour de cette base de données.

Type : Timestamp

Obligatoire : non

TableCount

Nombre total de tables trouvées dans une base de données Timestream.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DataModel

Service : Amazon Timestream Write

Modèle de données pour une tâche de chargement par lots.

Table des matières

DimensionMappings

Mappages source-cible pour les dimensions.

Type : tableau d'objets DimensionMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : oui

MeasureNameColumn

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 256.

Obligatoire : non

MixedMeasureMappings

Mappages source-cible pour les mesures.

Type : tableau d'objets MixedMeasureMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : non

MultiMeasureMappings

Mappages source-cible pour les enregistrements à mesures multiples.

Type : objet MultiMeasureMappings

Obligatoire : non

TimeColumn

Colonne source à mapper à l'heure.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 256.

Obligatoire : non

TimeUnit

La granularité de l'unité d'horodatage. Il indique si la valeur temporelle est en secondes, en millisecondes, en nanosecondes ou en d'autres valeurs prises en charge. La valeur par défaut est `MILLISECONDS`.

Type : String

Valeurs valides : `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DataModelConfiguration
Service : Amazon Timestream Write

Table des matières

DataModel

Type : objet DataModel

Obligatoire : non

DataModelS3Configuration

Type : objet DataModelS3Configuration

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DataModelS3Configuration
Service : Amazon Timestream Write

Table des matières

BucketName

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 63.

Modèle : `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obligatoire : non

ObjectKey

Type : String

Contraintes de longueur : Longueur minimum de 1. Longueur maximum de 1024.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DataSourceConfiguration
Service : Amazon Timestream Write

Définit les détails de configuration de la source de données.

Table des matières

DataFormat

C'est actuellement le casCSV.

Type : String

Valeurs valides : `CSV`

Obligatoire : oui

DataSourceS3Configuration

Configuration d'un emplacement S3 pour un fichier contenant des données à charger.

Type : objet DataSourceS3Configuration

Obligatoire : oui

CsvConfiguration

Format de données délimité dans lequel le séparateur de colonnes peut être une virgule et le séparateur d'enregistrements un caractère de nouvelle ligne.

Type : objet CsvConfiguration

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DataSourceS3Configuration
Service : Amazon Timestream Write

Table des matières

BucketName

Nom du compartiment S3 client.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 63.

Modèle : `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obligatoire : oui

ObjectKeyPrefix

Type : String

Contraintes de longueur : Longueur minimum de 1. Longueur maximum de 1024.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Dimension

Service : Amazon Timestream Write

Représente les attributs de métadonnées de la série chronologique. Par exemple, le nom et la zone de disponibilité d'une EC2 instance ou le nom du fabricant d'une éolienne sont des dimensions.

Table des matières

Name

La dimension représente les attributs de métadonnées de la série chronologique. Par exemple, le nom et la zone de disponibilité d'une EC2 instance ou le nom du fabricant d'une éolienne sont des dimensions.

Pour les restrictions relatives aux noms de dimension, consultez la section Contraintes de dénomination.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 60.

Obligatoire : oui

Value

Valeur de la dimension.

Type : String

Obligatoire : oui

DimensionValueType

Type de données de la dimension pour le point de données de la série chronologique.

Type : String

Valeurs valides : `VARCHAR`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DimensionMapping
Service : Amazon Timestream Write

Table des matières

DestinationColumn

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

SourceColumn

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Endpoint
Service : Amazon Timestream Write

Représente un point de terminaison disponible contre lequel effectuer des API appels, ainsi que le point de terminaison TTL pour ce point de terminaison.

Table des matières

Address

Une adresse de point de terminaison.

Type : String

Obligatoire : oui

CachePeriodInMinutes

Le TTL pour le point de terminaison, en minutes.

Type : long

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MagneticStoreRejectedDataLocation
Service : Amazon Timestream Write

Emplacement où écrire les rapports d'erreur pour les enregistrements rejetés, de manière asynchrone, lors des écritures sur stockage magnétique.

Table des matières

S3Configuration

Configuration d'un emplacement S3 où écrire les rapports d'erreur pour les enregistrements rejetés, de manière asynchrone, lors des écritures sur stockage magnétique.

Type : objet S3Configuration

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MagneticStoreWriteProperties
Service : Amazon Timestream Write

Ensemble de propriétés d'une table permettant de configurer des écritures sur stockage magnétique.

Table des matières

EnableMagneticStoreWrites

Indicateur pour activer les écritures sur stockage magnétique.

Type : booléen

Obligatoire : oui

MagneticStoreRejectedDataLocation

Emplacement où écrire les rapports d'erreur pour les enregistrements rejetés de manière asynchrone lors des écritures sur stockage magnétique.

Type : objet MagneticStoreRejectedDataLocation

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MeasureValue
Service : Amazon Timestream Write

Représente l'attribut de données de la série chronologique. Par exemple, l'CPUutilisation d'une EC2 instance ou RPM d'une éolienne sont des mesures. MeasureValue possède à la fois un nom et une valeur.

MeasureValue n'est autorisé que pour le type`MULTI`. En utilisant le `MULTI` type, vous pouvez transmettre plusieurs attributs de données associés à la même série chronologique dans un seul enregistrement

Table des matières

Name

Le nom du MeasureValue.

Pour les restrictions relatives aux MeasureValue noms, consultez la section Contraintes de dénomination dans le manuel Amazon Timestream Developer Guide.

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : oui

Type

Contient le type de données du point MeasureValue de données de la série chronologique.

Type : String

Valeurs valides : `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Obligatoire : oui

Value

La valeur du MeasureValue. Pour plus d'informations, consultez la section Types de données.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MixedMeasureMapping
Service : Amazon Timestream Write

Table des matières

MeasureValueType

Type : String

Valeurs valides : `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Obligatoire : oui

MeasureName

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

MultiMeasureAttributeMappings

Type : tableau d'objets MultiMeasureAttributeMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : non

SourceColumn

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

TargetMeasureName

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MultiMeasureAttributeMapping
Service : Amazon Timestream Write

Table des matières

SourceColumn

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : oui

MeasureValueType

Type : String

Valeurs valides : `DOUBLE | BIGINT | BOOLEAN | VARCHAR | TIMESTAMP`

Obligatoire : non

TargetMultiMeasureAttributeName

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MultiMeasureMappings
Service : Amazon Timestream Write

Table des matières

MultiMeasureAttributeMappings

Type : tableau d’objets MultiMeasureAttributeMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : oui

TargetMultiMeasureName

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

PartitionKey
Service : Amazon Timestream Write

Attribut utilisé pour partitionner les données d'une table. Une clé de dimension partitionne les données en utilisant les valeurs de la dimension spécifiée par le nom de dimension comme clé de partition, tandis qu'une clé de mesure partitionne les données en utilisant les noms de mesures (valeurs de la colonne « measure_name »).

Table des matières

Type

Type de clé de partition. Les options sont DIMENSION (clé de dimension) et MEASURE (clé de mesure).

Type : String

Valeurs valides : `DIMENSION | MEASURE`

Obligatoire : oui

EnforcementInRecord

Le niveau d'application pour la spécification d'une clé de dimension dans les enregistrements ingérés. Les options sont REQUIRED (la clé de dimension doit être spécifiée) et OPTIONAL (la clé de dimension n'a pas besoin d'être spécifiée).

Type : String

Valeurs valides : `REQUIRED | OPTIONAL`

Obligatoire : non

Name

Nom de l'attribut utilisé pour une clé de dimension.

Type : String

Contraintes de longueur : longueur minimum de 1.

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Record
Service : Amazon Timestream Write

Représente un point de données de série chronologique écrit dans Timestream. Chaque enregistrement contient un tableau de dimensions. Les dimensions représentent les attributs de métadonnées d'un point de données chronologique, tels que le nom de l'instance ou la zone de disponibilité d'une EC2 instance. Un enregistrement contient également le nom de la mesure, qui est le nom de la mesure collectée (par exemple, l'CPUutilisation d'une EC2 instance). En outre, un enregistrement contient la valeur de mesure et le type de valeur, qui est le type de données de la valeur de mesure. L'enregistrement contient également l'horodatage de la collecte de la mesure et l'unité d'horodatage, qui représente la granularité de l'horodatage.

Les enregistrements comportent un `Version` champ de 64 bits `long` que vous pouvez utiliser pour mettre à jour les points de données. Les écritures d'un enregistrement dupliqué avec la même dimension, le même horodatage et le même nom de mesure, mais avec une valeur de mesure différente, ne réussiront que si l'`Version`attribut de l'enregistrement dans la demande d'écriture est supérieur à celui de l'enregistrement existant. La valeur par défaut de Timestream est `Version` de `1` pour les enregistrements sans le champ. `Version`

Table des matières

Dimensions

Contient la liste des dimensions des points de données de séries chronologiques.

Type : tableau d’objets Dimension

Membres du tableau : nombre maximum de 128 éléments.

Obligatoire : non

MeasureName

La mesure représente l'attribut de données de la série chronologique. Par exemple, l'CPUutilisation d'une EC2 instance ou RPM d'une éolienne sont des mesures.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 256.

Obligatoire : non

MeasureValue

Contient la valeur de mesure pour le point de données de la série chronologique.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

MeasureValues

Contient la liste des quatre points MeasureValue de données de séries chronologiques.

Ceci n'est autorisé que pour le type`MULTI`. Pour les valeurs scalaires, utilisez directement `MeasureValue` l'attribut de l'enregistrement.

Type : tableau d’objets MeasureValue

Obligatoire : non

MeasureValueType

Contient le type de données de la valeur de mesure pour le point de données de la série chronologique. Le type par défaut est`DOUBLE`. Pour plus d'informations, consultez la section Types de données.

Type : String

Valeurs valides : `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Obligatoire : non

Time

Contient l'heure à laquelle la valeur de mesure pour le point de données a été collectée. La valeur du temps plus l'unité indique le temps écoulé depuis l'époque. Par exemple, si la valeur temporelle est `12345` et l'unité l'est`ms`, cela signifie que le temps s'est écoulé `12345 ms` depuis l'époque.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 256.

Obligatoire : non

TimeUnit

La granularité de l'unité d'horodatage. Il indique si la valeur temporelle est en secondes, en millisecondes, en nanosecondes ou en d'autres valeurs prises en charge. La valeur par défaut est `MILLISECONDS`.

Type : String

Valeurs valides : `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Obligatoire : non

Version

Attribut 64 bits utilisé pour les mises à jour des enregistrements. Rédiger des demandes pour des données dupliquées avec un numéro de version supérieur mettra à jour la valeur et la version de mesure existantes. Dans les cas où la valeur de mesure est identique, elle `Version` sera toujours mise à jour. La valeur par défaut est 1.

> **Note**
>
> `Version`doit être 1 égal ou supérieur, sinon vous recevrez un `ValidationException` message d'erreur.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

RecordsIngested
Service : Amazon Timestream Write

Informations sur les enregistrements ingérés par cette demande.

Table des matières

MagneticStore

Nombre d'enregistrements ingérés dans le magasin magnétique.

Type : entier

Obligatoire : non

MemoryStore

Nombre d'enregistrements ingérés dans la mémoire.

Type : entier

Obligatoire : non

Total

Nombre total d'enregistrements correctement ingérés.

Type : entier

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

RejectedRecord
Service : Amazon Timestream Write

Représente les enregistrements qui n'ont pas été correctement insérés dans Timestream en raison de problèmes de validation des données qui doivent être résolus avant de réinsérer des données de séries chronologiques dans le système.

Table des matières

ExistingVersion

Version existante de l'enregistrement. Cette valeur est renseignée dans les scénarios où un enregistrement identique existe avec une version supérieure à celle de la demande d'écriture.

Type : long

Obligatoire : non

Reason

La raison pour laquelle un enregistrement n'a pas été correctement inséré dans Timestream. Les causes possibles d'échec sont les suivantes :

- Enregistrements contenant des données dupliquées contenant plusieurs enregistrements ayant les mêmes dimensions, horodatages et noms de mesures, mais :
  - Les valeurs de mesure sont différentes
  - La version n'est pas présente dans la demande, ou la valeur de la version dans le nouvel enregistrement est égale ou inférieure à la valeur existante

  Si Timestream rejette les données pour ce cas, le `ExistingVersion` champ de `RejectedRecords` réponse indiquera la version de l'enregistrement actuel. Pour forcer une mise à jour, vous pouvez renvoyer la demande avec une version de l'enregistrement définie sur une valeur supérieure à. `ExistingVersion`
- Enregistrements dont l'horodatage se situe en dehors de la durée de conservation de la mémoire.

  > **Note**
  >
  > Lorsque la fenêtre de rétention est mise à jour, vous recevrez une `RejectedRecords` exception si vous essayez immédiatement d'ingérer des données dans la nouvelle fenêtre. Pour éviter une `RejectedRecords` exception, attendez la fin de la nouvelle

fenêtre pour ingérer de nouvelles données. Pour plus d'informations, consultez les meilleures pratiques de configuration de Timestream et l'explication du fonctionnement du stockage dans Timestream.

- Enregistrements dont les dimensions ou les mesures dépassent les limites définies par Timestream.

Pour de plus amples informations, veuillez consulter Access Management dans le Manuel du développeur Timestream.

Type : chaîne

Obligatoire : non

RecordIndex

L'index de l'enregistrement dans la demande d'entrée pour WriteRecords. Les index commencent par 0.

Type : entier

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ReportConfiguration
Service : Amazon Timestream Write

Configuration du rapport pour une tâche de chargement par lots. Il contient des informations sur l'emplacement de stockage des rapports d'erreur.

Table des matières

ReportS3Configuration

Configuration d'un emplacement S3 pour écrire des rapports d'erreur et des événements pour un chargement par lots.

Type : objet ReportS3Configuration

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ReportS3Configuration
Service : Amazon Timestream Write

Table des matières

BucketName

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 63.

Modèle : `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obligatoire : oui

EncryptionOption

Type : String

Valeurs valides : `SSE_S3 | SSE_KMS`

Obligatoire : non

KmsKeyId

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

ObjectKeyPrefix

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 928.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

RetentionProperties
Service : Amazon Timestream Write

Les propriétés de rétention indiquent la durée pendant laquelle les données de séries temporelles doivent être stockées dans le stockage magnétique et le stockage en mémoire.

Table des matières

MagneticStoreRetentionPeriodInDays

Durée pendant laquelle les données doivent être stockées dans le stockage magnétique.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 73 000.

Obligatoire : oui

MemoryStoreRetentionPeriodInHours

Durée pendant laquelle les données doivent être stockées dans le stockage en mémoire.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 8766.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

S3Configuration

Service : Amazon Timestream Write

Configuration qui spécifie un emplacement S3.

Table des matières

BucketName

Nom du compartiment S3 client.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 63.

Modèle : `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obligatoire : non

EncryptionOption

Option de chiffrement pour l'emplacement S3 client. Les options sont le chiffrement côté serveur S3 avec une clé gérée S3 ou une clé AWS gérée.

Type : String

Valeurs valides : `SSE_S3 | SSE_KMS`

Obligatoire : non

KmsKeyId

L'ID de AWS KMS clé de l'emplacement S3 du client lors du chiffrement à l'aide d'une clé AWS gérée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

ObjectKeyPrefix

Aperçu de la clé d'objet pour l'emplacement S3 client.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 928.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Schema

Service : Amazon Timestream Write

Un schéma indique le modèle de données attendu de la table.

Table des matières

CompositePartitionKey

Liste non vide de clés de partition définissant les attributs utilisés pour partitionner les données de la table. L'ordre de la liste détermine la hiérarchie des partitions. Le nom et le type de chaque clé de partition ainsi que l'ordre des clés de partition ne peuvent pas être modifiés une fois la table créée. Cependant, le niveau d'application de chaque clé de partition peut être modifié.

Type : tableau d’objets PartitionKey

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Table

Service : Amazon Timestream Write

Représente une table de base de données dans Timestream. Les tableaux contiennent une ou plusieurs séries chronologiques connexes. Vous pouvez modifier la durée de conservation de la mémoire et de la mémoire magnétique d'une table.

Table des matières

Arn

Le nom de la ressource Amazon qui identifie de manière unique cette table.

Type : chaîne

Obligatoire : non

CreationTime

Heure à laquelle la table Timestream a été créée.

Type : Timestamp

Obligatoire : non

DatabaseName

Nom de la base de données Timestream qui contient cette table.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : non

LastUpdatedTime

Heure à laquelle la table Timestream a été mise à jour pour la dernière fois.

Type : Timestamp

Obligatoire : non

MagneticStoreWriteProperties

Contient les propriétés à définir sur la table lors de l'activation des écritures sur stockage magnétique.

Type : objet MagneticStoreWriteProperties

Obligatoire : non

RetentionProperties

Durée de conservation pour le stockage en mémoire et le stockage magnétique.

Type : objet RetentionProperties

Obligatoire : non

Schema

Schéma de la table.

Type : objet Schema

Obligatoire : non

TableName

Nom de la table Timestream.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximale de 256.

Obligatoire : non

TableStatus

État actuel de la table :

- `DELETING`- La table est en cours de suppression.
- `ACTIVE`- La table est prête à être utilisée.

Type : String

Valeurs valides : `ACTIVE | DELETING | RESTORING`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Tag
Service : Amazon Timestream Write

Une balise est une étiquette que vous attribuez aux 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 tables d'une base de données Timestream, par exemple, par objectif, propriétaire ou environnement.

Table des matières

Key

Clé de la balise. Les clés de balises sont sensibles à la casse.

Type : String

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Obligatoire : oui

Value

Valeur de la balise. Les valeurs des balises font la distinction majuscules/majuscules et peuvent être nulles.

Type : String

Contraintes de longueur : Longueur minimum de 0. Longueur maximale de 256.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Requête Amazon Timestream

Les types de données suivants sont pris en charge par 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
Service : Amazon Timestream Query

Contient les métadonnées pour les résultats des requêtes, tels que les noms des colonnes, les types de données et d'autres attributs.

Table des matières

Type

Type de données de la colonne du jeu de résultats. Le type de données peut être scalaire ou complexe. Les types de données scalaires sont des entiers, des chaînes, des doubles, des booléens, etc. Les types de données complexes sont des types tels que des tableaux, des lignes, etc.

Type : objet Type

Obligatoire : oui

Name

Nom de la colonne du jeu de résultats. Le nom du jeu de résultats est disponible pour les colonnes de tous les types de données, à l'exception des tableaux.

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Datum

Service : Amazon Timestream Query

Datum représente un point de données unique dans le résultat d'une requête.

Table des matières

ArrayValue

Indique si le point de données est un tableau.

Type : tableau d’objets Datum

Obligatoire : non

NullValue

Indique si le point de données est nul.

Type : booléen

Obligatoire : non

RowValue

Indique si le point de données est une ligne.

Type : objet Row

Obligatoire : non

ScalarValue

Indique si le point de données est une valeur scalaire telle qu'un entier, une chaîne, un double ou un booléen.

Type : chaîne

Obligatoire : non

TimeSeriesValue

Indique si le point de données est un type de données de série chronologique.

Type : tableau d’objets TimeSeriesDataPoint

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

DimensionMapping
Service : Amazon Timestream Query

Ce type est utilisé pour mapper la ou les colonnes du résultat de la requête à une dimension de la table de destination.

Table des matières

DimensionValueType

Type pour la dimension.

Type : String

Valeurs valides : `VARCHAR`

Obligatoire : oui

Name

Nom de colonne issu du résultat de la requête.

Type : String

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Endpoint
Service : Amazon Timestream Query

Représente un point de terminaison disponible contre lequel effectuer des API appels, ainsi que le point de terminaison TTL pour ce point de terminaison.

Table des matières

Address

Une adresse de point de terminaison.

Type : String

Obligatoire : oui

CachePeriodInMinutes

Le TTL pour le point de terminaison, en minutes.

Type : long

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ErrorReportConfiguration
Service : Amazon Timestream Query

Configuration requise pour le signalement des erreurs.

Table des matières

S3Configuration

La configuration S3 pour les rapports d'erreurs.

Type : objet S3Configuration

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ErrorReportLocation
Service : Amazon Timestream Query

Il contient l'emplacement du rapport d'erreur pour un seul appel de requête planifié.

Table des matières

S3ReportLocation

Emplacement S3 où les rapports d'erreur sont rédigés.

Type : objet S3ReportLocation

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ExecutionStats

Service : Amazon Timestream Query

Statistiques pour une seule exécution de requête planifiée.

Table des matières

BytesMetered

Octets mesurés pour une seule exécution de requête planifiée.

Type : long

Obligatoire : non

CumulativeBytesScanned

Octets analysés pour une seule exécution de requête planifiée.

Type : long

Obligatoire : non

DataWrites

Les écritures de données sont mesurées pour les enregistrements ingérés lors d'une seule exécution de requête planifiée.

Type : long

Obligatoire : non

ExecutionTimeInMillis

Durée totale, mesurée en millisecondes, nécessaire à l'exécution de la requête planifiée.

Type : long

Obligatoire : non

QueryResultRows

Nombre de lignes présentes dans le résultat de l'exécution d'une requête avant son ingestion vers la source de données de destination.

Type : long

Obligatoire : non

RecordsIngested

Nombre d'enregistrements ingérés pour une seule exécution de requête planifiée.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MixedMeasureMapping

Service : Amazon Timestream Query

MixedMeasureMappings sont des mappages qui peuvent être utilisés pour intégrer des données dans un mélange de mesures restreintes et de mesures multiples dans le tableau dérivé.

Table des matières

MeasureValueType

Type de la valeur à partir de laquelle la lecture doit être effectuéesourceColumn. Si le mappage est pourMULTI, utilisez MeasureValueType. MULTI.

Type : String

Valeurs valides : `BIGINT | BOOLEAN | DOUBLE | VARCHAR | MULTI`

Obligatoire : oui

MeasureName

Fait référence à la valeur de measure_name dans une ligne de résultat. Ce champ est obligatoire s'il MeasureNameColumn est fourni.

Type : chaîne

Obligatoire : non

MultiMeasureAttributeMappings

Obligatoire quand measureValueType c'est le casMULTI. Mappages d'attributs pour les mesures MULTI de valeur.

Type : tableau d’objets MultiMeasureAttributeMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : non

SourceColumn

Ce champ fait référence à la colonne source à partir de laquelle la valeur de mesure doit être lue pour la matérialisation des résultats.

Type : chaîne

Obligatoire : non

TargetMeasureName

Nom de la mesure cible à utiliser. S'il n'est pas fourni, le nom de la mesure cible par défaut serait measure-name s'il est fourni, ou sourceColumn autrement.

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MultiMeasureAttributeMapping
Service : Amazon Timestream Query

Mappage d'attributs pour les mesures de MULTI valeur.

Table des matières

MeasureValueType

Type de l'attribut à lire depuis la colonne source.

Type : String

Valeurs valides : `BIGINT | BOOLEAN | DOUBLE | VARCHAR | TIMESTAMP`

Obligatoire : oui

SourceColumn

Colonne source à partir de laquelle la valeur d'attribut doit être lue.

Type : String

Obligatoire : oui

TargetMultiMeasureAttributeName

Nom personnalisé à utiliser comme nom d'attribut dans la table dérivée. S'il n'est pas fourni, le nom de la colonne source sera utilisé.

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

MultiMeasureMappings
Service : Amazon Timestream Query

Un seul MixedMeasureMappings des deux MultiMeasureMappings doit être fourni. MultiMeasureMappings peut être utilisé pour ingérer des données sous forme de mesures multiples dans la table dérivée.

Table des matières

MultiMeasureAttributeMappings

Obligatoire. Mappages d'attributs à utiliser pour mapper les résultats des requêtes afin d'intégrer des données pour des attributs multi-mesures.

Type : tableau d’objets MultiMeasureAttributeMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : oui

TargetMultiMeasureName

Le nom de la multi-mesure cible dans la table dérivée. Cette saisie est obligatoire lorsqu'elle n' measureNameColumn est pas fournie. Si elle MeasureNameColumn est fournie, la valeur de cette colonne sera utilisée comme nom multi-mesures.

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

NotificationConfiguration
Service : Amazon Timestream Query

Configuration de la notification pour une requête planifiée. Une notification est envoyée par Timestream lorsqu'une requête planifiée est créée, lorsque son état est mis à jour ou lorsque vous la supprimez.

Table des matières

SnsConfiguration

Détails sur la SNS configuration.

Type : objet SnsConfiguration

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ParameterMapping
Service : Amazon Timestream Query

Cartographie des paramètres nommés.

Table des matières

Name

Nom du paramètre.

Type : String

Obligatoire : oui

Type

Contient le type de données d'une colonne dans un jeu de résultats de requête. Le type de données peut être scalaire ou complexe. Les types de données scalaires pris en charge sont les entiers, les booléens, les chaînes, les doubles, les horodatages, les dates, les heures et les intervalles. Les types de données complexes pris en charge sont les tableaux, les lignes et les séries temporelles.

Type : objet Type

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QueryInsights
Service : Amazon Timestream Query

`QueryInsights`est une fonctionnalité de réglage des performances qui vous permet d'optimiser vos requêtes, de réduire les coûts et d'améliorer les performances. Vous pouvez ainsi évaluer l'efficacité de l'élagage de vos requêtes et identifier les domaines à améliorer pour améliorer les performances des requêtes. `QueryInsights` Vous pouvez également analyser l'efficacité de vos requêtes en termes d'élagage temporel et spatial, et identifier les opportunités d'amélioration des performances. `QueryInsights` Plus précisément, vous pouvez évaluer dans quelle mesure vos requêtes utilisent les stratégies d'indexation basées sur le temps et les clés de partition pour optimiser la récupération des données. Pour optimiser les performances des requêtes, il est essentiel de peaufiner les paramètres temporels et spatiaux qui régissent l'exécution des requêtes.

Les indicateurs clés fournis par `QueryInsights` sont `QuerySpatialCoverage` et`QueryTemporalRange`. `QuerySpatialCoverage`indique la partie de l'axe spatial scannée par la requête, les valeurs les plus faibles étant plus efficaces. `QueryTemporalRange`indique la plage de temps numérisée, les plages plus étroites étant plus performantes.

Les avantages de QueryInsights

Les principaux avantages de l'utilisation sont les `QueryInsights` suivants :

- Identification des requêtes inefficaces : `QueryInsights` fournit des informations sur l'élagage basé sur le temps et les attributs des tables auxquelles la requête accède. Ces informations vous aident à identifier les tables auxquelles l'accès n'est pas optimal.
- Optimisation de votre modèle de données et de partitionnement : vous pouvez utiliser les `QueryInsights` informations pour accéder à votre modèle de données et à votre stratégie de partitionnement et les affiner.
- Réglage des requêtes : `QueryInsights` met en évidence les possibilités d'utiliser les index de manière plus efficace.

> **Note**
>
> Le nombre maximum de `Query` API demandes que vous êtes autorisé à effectuer lorsque `QueryInsights` cette option est activée est d'une requête par seconde (QPS). Si vous dépassez ce taux de requêtes, cela peut entraîner un ralentissement.

Table des matières

Mode

Fournit les modes suivants à activer `QueryInsights` :

- `ENABLED_WITH_RATE_CONTROL`— Active `QueryInsights` le traitement des requêtes. Ce mode inclut également un mécanisme de contrôle du débit, qui limite la `QueryInsights` fonctionnalité à une requête par seconde (QPS).
- `DISABLED`— Désactive. `QueryInsights`

Type : String

Valeurs valides : `ENABLED_WITH_RATE_CONTROL | DISABLED`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QueryInsightsResponse
Service : Amazon Timestream Query

Fournit diverses informations et mesures relatives à la requête que vous avez exécutée.

Table des matières

OutputBytes

Indique la taille du jeu de résultats de requête en octets. Vous pouvez utiliser ces données pour vérifier si le jeu de résultats a changé dans le cadre de l'exercice de réglage des requêtes.

Type : long

Obligatoire : non

OutputRows

Indique le nombre total de lignes renvoyées dans le cadre du jeu de résultats de la requête. Vous pouvez utiliser ces données pour vérifier si le nombre de lignes du jeu de résultats a changé dans le cadre de l'exercice de réglage des requêtes.

Type : long

Obligatoire : non

QuerySpatialCoverage

Fournit des informations sur la couverture spatiale de la requête, y compris la table avec un élagage spatial sous-optimal (maximum). Ces informations peuvent vous aider à identifier les points à améliorer dans votre stratégie de partitionnement afin d'améliorer l'élagage spatial.

Type : objet QuerySpatialCoverage

Obligatoire : non

QueryTableCount

Indique le nombre de tables contenues dans la requête.

Type : long

Obligatoire : non

QueryTemporalRange

Fournit des informations sur la plage temporelle de la requête, y compris la table présentant la plage de temps (maximale) la plus grande. Voici quelques-unes des options possibles pour optimiser l'élagage basé sur le temps :

- Ajoutez les prédicats temporels manquants.
- Supprimez les fonctions associées aux prédicats temporels.
- Ajoutez des prédicats temporels à toutes les sous-requêtes.

Type : objet QueryTemporalRange

Obligatoire : non

UnloadPartitionCount

Indique les partitions créées par l'`Unload`opération.

Type : long

Obligatoire : non

UnloadWrittenBytes

Indique la taille, en octets, écrite par l'`Unload`opération.

Type : long

Obligatoire : non

UnloadWrittenRows

Indique les lignes écrites par la `Unload` requête.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++

- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QuerySpatialCoverage
Service : Amazon Timestream Query

Fournit des informations sur la couverture spatiale de la requête, y compris la table avec un élagage spatial sous-optimal (maximum). Ces informations peuvent vous aider à identifier les points à améliorer dans votre stratégie de partitionnement afin d'améliorer l'élagage spatial.

Par exemple, vous pouvez effectuer les opérations suivantes avec les `QuerySpatialCoverage` informations :

- Ajoutez measure_name ou utilisez des prédicats de clé de partition () définis par le client. CDPK
- Si vous avez déjà effectué l'action précédente, supprimez les fonctions ou les clauses qui les entourent, telles que`LIKE`.

Table des matières

Max

Fournit des informations sur la couverture spatiale de la requête exécutée et de la table présentant l'élagage spatial le plus inefficace.

- `Value`— Le ratio maximal de couverture spatiale.
- `TableArn`— Le nom de la ressource Amazon (ARN) de la table dont l'élagage spatial n'est pas optimal.
- `PartitionKey`— La clé de partition utilisée pour le partitionnement, qui peut être une valeur par défaut `measure_name` ou uneCDPK.

Type : objet QuerySpatialCoverageMax

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QuerySpatialCoverageMax
Service : Amazon Timestream Query

Fournit des informations sur la table dont la plage spatiale est la moins optimale analysée par votre requête.

Table des matières

PartitionKey

La clé de partition utilisée pour le partitionnement, qui peut être une clé de partition par défaut `measure_name` ou définie par le client.

Type : tableau de chaînes

Obligatoire : non

TableArn

Nom de la ressource Amazon (ARN) de la table présentant l'élagage spatial le moins optimal.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

Value

Le ratio maximal de couverture spatiale.

Type : double

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QueryStatus

Service : Amazon Timestream Query

Informations sur l'état de la requête, y compris la progression et les octets analysés.

Table des matières

CumulativeBytesMetered

La quantité de données numérisées par la requête en octets qui vous sera facturée. Il s'agit d'une somme cumulée qui représente la quantité totale de données qui vous sera facturée depuis le début de la requête. Les frais ne sont appliqués qu'une seule fois et sont appliqués soit lorsque la requête est terminée, soit lorsque la requête est annulée.

Type : long

Obligatoire : non

CumulativeBytesScanned

Quantité de données scannée par la requête en octets. Il s'agit d'une somme cumulée qui représente le nombre total d'octets analysés depuis le lancement de la requête.

Type : long

Obligatoire : non

ProgressPercentage

La progression de la requête, exprimée en pourcentage.

Type : double

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QueryTemporalRange
Service : Amazon Timestream Query

Fournit des informations sur la plage temporelle de la requête, y compris la table présentant la plage de temps (maximale) la plus grande.

Table des matières

Max

Encapsule les propriétés suivantes qui fournissent des informations sur la table de performances la moins optimale sur l'axe temporel :

- `Value`— Durée maximale en nanosecondes entre le début et la fin de la requête.
- `TableArn`— Le nom de la ressource Amazon (ARN) de la table interrogée avec la plage de temps la plus longue.

Type : objet QueryTemporalRangeMax

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

QueryTemporalRangeMax
Service : Amazon Timestream Query

Fournit des informations sur le tableau présentant l'élagage temporel le moins optimal analysé par votre requête.

Table des matières

TableArn

Nom de la ressource Amazon (ARN) de la table interrogée avec la plage de temps la plus longue.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

Value

Durée maximale en nanosecondes entre le début et la fin de la requête.

Type : long

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Row
Service : Amazon Timestream Query

Représente une seule ligne dans les résultats de la requête.

Table des matières

Data

Liste des points de données sur une seule ligne du jeu de résultats.

Type : tableau d’objets Datum

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

S3Configuration
Service : Amazon Timestream Query

Détails sur l'emplacement S3 pour les rapports d'erreurs résultant de l'exécution d'une requête.

Table des matières

BucketName

Nom du compartiment S3 sous lequel les rapports d'erreurs seront créés.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 63.

Modèle : `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obligatoire : oui

EncryptionOption

Options de chiffrement au repos pour les rapports d'erreurs. Si aucune option de chiffrement n'est spécifiée, Timestream choisira SSE _S3 par défaut.

Type : String

Valeurs valides : `SSE_S3 | SSE_KMS`

Obligatoire : non

ObjectKeyPrefix

Préfixe pour la clé du rapport d'erreurs. Timestream ajoute par défaut le préfixe suivant au chemin d'accès au rapport d'erreurs.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 896.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

S3ReportLocation
Service : Amazon Timestream Query

Emplacement du rapport S3 pour l'exécution planifiée de la requête.

Table des matières

BucketName

Nom du compartiment S3.

Type : String

Contraintes de longueur : Longueur minimum de 3. Longueur maximum de 63.

Modèle : `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obligatoire : non

ObjectKey

Clé S3

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ScheduleConfiguration
Service : Amazon Timestream Query

Configuration de la planification de la requête.

Table des matières

ScheduleExpression

Une expression qui indique quand déclencher l'exécution planifiée de la requête. Il peut s'agir d'une expression cron ou d'une expression rate.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 256.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ScheduledQuery

Service : Amazon Timestream Query

Requête planifiée

Table des matières

Arn

Le nom de la ressource Amazon.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Name

Nom de la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 64.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : oui

State

État de la requête planifiée.

Type : String

Valeurs valides : `ENABLED | DISABLED`

Obligatoire : oui

CreationTime

Heure de création de la requête planifiée.

Type : Timestamp

Obligatoire : non

ErrorReportConfiguration

Configuration pour le signalement des erreurs de requête planifiées.

Type : objet ErrorReportConfiguration

Obligatoire : non

LastRunStatus

État de la dernière requête planifiée exécutée.

Type : String

Valeurs valides : `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Obligatoire : non

NextInvocationTime

La prochaine fois que la requête planifiée sera exécutée.

Type : Timestamp

Obligatoire : non

PreviousInvocationTime

La dernière fois que la requête planifiée a été exécutée.

Type : Timestamp

Obligatoire : non

TargetDestination

Source de données cible dans laquelle le résultat final de la requête planifiée sera écrit.

Type : objet TargetDestination

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ScheduledQueryDescription
Service : Amazon Timestream Query

Structure qui décrit la requête planifiée.

Table des matières

Arn

Requête planifiéeARN.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

Name

Nom de la requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 64.

Modèle : `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obligatoire : oui

NotificationConfiguration

Configuration des notifications.

Type : objet NotificationConfiguration

Obligatoire : oui

QueryString

Requête à exécuter.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximum de 262 144.

Obligatoire : oui

ScheduleConfiguration

Configuration d'une planification.

Type : objet ScheduleConfiguration

Obligatoire : oui

State

État de la requête planifiée.

Type : String

Valeurs valides : `ENABLED | DISABLED`

Obligatoire : oui

CreationTime

Heure de création de la requête planifiée.

Type : Timestamp

Obligatoire : non

ErrorReportConfiguration

Configuration du rapport d'erreur pour la requête planifiée.

Type : objet ErrorReportConfiguration

Obligatoire : non

KmsKeyId

Une KMS clé fournie par le client est utilisée pour chiffrer la ressource de requête planifiée.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

LastRunSummary

Résumé de l'exécution de la dernière requête planifiée exécutée.

Type : objet ScheduledQueryRunSummary

Obligatoire : non

NextInvocationTime

La prochaine fois que l'exécution de la requête planifiée est planifiée.

Type : Timestamp

Obligatoire : non

PreviousInvocationTime

La dernière fois que la requête a été exécutée.

Type : Timestamp

Obligatoire : non

RecentlyFailedRuns

Résumé de l'exécution des cinq dernières exécutions de requêtes planifiées ayant échoué.

Type : tableau d’objets ScheduledQueryRunSummary

Obligatoire : non

ScheduledQueryExecutionRoleArn

IAMrôle que Timestream utilise pour exécuter la requête de planification.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : non

TargetConfiguration

Configuration planifiée du magasin cible de requêtes.

Type : objet TargetConfiguration

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ScheduledQueryInsights
Service : Amazon Timestream Query

Encapsule les paramètres à activer `QueryInsights` sur un`ExecuteScheduledQueryRequest`.

Table des matières

Mode

Fournit les modes suivants à activer `ScheduledQueryInsights` :

- `ENABLED_WITH_RATE_CONTROL`— Active `ScheduledQueryInsights` le traitement des requêtes. Ce mode inclut également un mécanisme de contrôle du débit, qui limite la `QueryInsights` fonctionnalité à une requête par seconde (QPS).
- `DISABLED`— Désactive. `ScheduledQueryInsights`

Type : String

Valeurs valides : `ENABLED_WITH_RATE_CONTROL | DISABLED`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ScheduledQueryInsightsResponse
Service : Amazon Timestream Query

Fournit diverses informations et mesures liées à `ExecuteScheduledQueryRequest` ce qui a été exécuté.

Table des matières

OutputBytes

Indique la taille du jeu de résultats de requête en octets. Vous pouvez utiliser ces données pour vérifier si le jeu de résultats a changé dans le cadre de l'exercice de réglage des requêtes.

Type : long

Obligatoire : non

OutputRows

Indique le nombre total de lignes renvoyées dans le cadre du jeu de résultats de la requête. Vous pouvez utiliser ces données pour vérifier si le nombre de lignes du jeu de résultats a changé dans le cadre de l'exercice de réglage des requêtes.

Type : long

Obligatoire : non

QuerySpatialCoverage

Fournit des informations sur la couverture spatiale de la requête, y compris la table avec un élagage spatial sous-optimal (maximum). Ces informations peuvent vous aider à identifier les points à améliorer dans votre stratégie de partitionnement afin d'améliorer l'élagage spatial.

Type : objet QuerySpatialCoverage

Obligatoire : non

QueryTableCount

Indique le nombre de tables contenues dans la requête.

Type : long

Obligatoire : non

QueryTemporalRange

Fournit des informations sur la plage temporelle de la requête, y compris la table présentant la plage de temps (maximale) la plus grande. Voici quelques-unes des options possibles pour optimiser l'élagage basé sur le temps :

- Ajoutez les prédicats temporels manquants.
- Supprimez les fonctions associées aux prédicats temporels.
- Ajoutez des prédicats temporels à toutes les sous-requêtes.

Type : objet QueryTemporalRange

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

ScheduledQueryRunSummary
Service : Amazon Timestream Query

Récapitulatif de l'exécution de la requête planifiée

Table des matières

ErrorReportLocation

Emplacement S3 pour le rapport d'erreur.

Type : objet ErrorReportLocation

Obligatoire : non

ExecutionStats

Statistiques d'exécution pour une exécution planifiée.

Type : objet ExecutionStats

Obligatoire : non

FailureReason

Message d'erreur pour la requête planifiée en cas d'échec. Vous devrez peut-être consulter le rapport d'erreur pour obtenir des raisons d'erreur plus détaillées.

Type : chaîne

Obligatoire : non

InvocationTime

InvocationTime pour cette course. Il s'agit de l'heure à laquelle l'exécution de la requête est planifiée. Le paramètre `@scheduled_runtime` peut être utilisé dans la requête pour obtenir la valeur.

Type : Timestamp

Obligatoire : non

QueryInsightsResponse

Fournit diverses informations et mesures relatives au résumé de l'exécution de la requête planifiée.

Type : objet ScheduledQueryInsightsResponse

Obligatoire : non

RunStatus

État de l'exécution d'une requête planifiée.

Type : String

Valeurs valides : `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Obligatoire : non

TriggerTime

Heure réelle à laquelle la requête a été exécutée.

Type : Timestamp

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

SelectColumn
Service : Amazon Timestream Query

Détails de la colonne renvoyée par la requête.

Table des matières

Aliased

Vrai, si le nom de colonne a été aliasé par la requête. Faux dans le cas contraire.

Type : booléen

Obligatoire : non

DatabaseName

Base de données contenant cette colonne.

Type : chaîne

Obligatoire : non

Name

Nom de la colonne.

Type : chaîne

Obligatoire : non

TableName

Table de la base de données contenant cette colonne.

Type : chaîne

Obligatoire : non

Type

Contient le type de données d'une colonne dans un jeu de résultats de requête. Le type de données peut être scalaire ou complexe. Les types de données scalaires pris en charge sont les entiers, les booléens, les chaînes, les doubles, les horodatages, les dates, les heures et les intervalles. Les types de données complexes pris en charge sont les tableaux, les lignes et les séries temporelles.

Type : objet Type

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

SnsConfiguration
Service : Amazon Timestream Query

Des informations à SNS ce sujet sont nécessaires pour envoyer la notification.

Table des matières

TopicArn

SNSsujet ARN auquel les notifications d'état des requêtes planifiées seront envoyées.

Type : String

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Tag
Service : Amazon Timestream Query

Une balise est une étiquette que vous attribuez aux 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 tables d'une base de données Timestream, par exemple, par objectif, propriétaire ou environnement.

Table des matières

Key

Clé de la balise. Les clés de balises sont sensibles à la casse.

Type : String

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Obligatoire : oui

Value

Valeur de la balise. Les valeurs des balises distinguent les majuscules et minuscules et peuvent être nulles.

Type : String

Contraintes de longueur : Longueur minimum de 0. Longueur maximale de 256.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

TargetConfiguration
Service : Amazon Timestream Query

Configuration utilisée pour écrire la sortie d'une requête.

Table des matières

TimestreamConfiguration

Configuration requise pour écrire des données dans la base de données et la table Timestream.

Type : objet TimestreamConfiguration

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

TargetDestination
Service : Amazon Timestream Query

Détails de destination pour écrire des données pour une source de données cible. La source de données actuellement prise en charge est Timestream.

Table des matières

TimestreamDestination

Détails de destination des résultats de la requête pour la source de données Timestream.

Type : objet TimestreamDestination

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

TimeSeriesDataPoint
Service : Amazon Timestream Query

Le type de données de série chronologique représente les valeurs d'une mesure au fil du temps. Une série chronologique est un ensemble de lignes d'horodatages et de valeurs de mesure, les lignes étant triées par ordre croissant de temps. A TimeSeriesDataPoint est un point de données unique de la série chronologique. Il représente un tuple de (temps, valeur de mesure) dans une série chronologique.

Table des matières

Time

Horodatage auquel la valeur de mesure a été collectée.

Type : String

Obligatoire : oui

Value

La valeur de mesure pour le point de données.

Type : objet Datum

Obligatoire : oui

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

TimestreamConfiguration
Service : Amazon Timestream Query

Configuration pour écrire des données dans la base de données et la table Timestream. Cette configuration permet à l'utilisateur de mapper les colonnes de sélection du résultat de la requête dans les colonnes de la table de destination.

Table des matières

DatabaseName

Nom de la base de données Timestream dans laquelle le résultat de la requête sera écrit.

Type : String

Obligatoire : oui

DimensionMappings

Cela permet de mapper la ou les colonnes du résultat de la requête à la dimension de la table de destination.

Type : tableau d’objets DimensionMapping

Obligatoire : oui

TableName

Nom de la table Timestream dans laquelle le résultat de la requête sera écrit. La table doit se trouver dans la même base de données que celle fournie dans la configuration de Timestream.

Type : String

Obligatoire : oui

TimeColumn

Colonne du résultat de la requête qui doit être utilisée comme colonne de temps dans la table de destination. Le type de colonne pour cela doit êtreTIMESTAMP.

Type : String

Obligatoire : oui

MeasureNameColumn

Nom de la colonne de mesure.

Type : chaîne

Obligatoire : non

MixedMeasureMappings

Spécifie comment mapper les mesures aux enregistrements multi-mesures.

Type : tableau d’objets MixedMeasureMapping

Membres du tableau : Nombre minimum de 1 élément.

Obligatoire : non

MultiMeasureMappings

Mappages multi-mesures.

Type : objet MultiMeasureMappings

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

TimestreamDestination
Service : Amazon Timestream Query

Destination pour la requête planifiée.

Table des matières

DatabaseName

Nom de la base de données Timestream.

Type : chaîne

Obligatoire : non

TableName

Nom de la table Timestream.

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Type
Service : Amazon Timestream Query

Contient le type de données d'une colonne dans un jeu de résultats de requête. Le type de données peut être scalaire ou complexe. Les types de données scalaires pris en charge sont les entiers, les booléens, les chaînes, les doubles, les horodatages, les dates, les heures et les intervalles. Les types de données complexes pris en charge sont les tableaux, les lignes et les séries temporelles.

Table des matières

ArrayColumnInfo

Indique si la colonne est un tableau.

Type : objet ColumnInfo

Obligatoire : non

RowColumnInfo

Indique si la colonne est une ligne.

Type : tableau d’objets ColumnInfo

Obligatoire : non

ScalarType

Indique si la colonne est de type chaîne, entier, booléen, double, horodatage, date, heure. Pour plus d'informations, consultez la section Types de données pris en charge.

Type : String

Valeurs valides : `VARCHAR | BOOLEAN | BIGINT | DOUBLE | TIMESTAMP | DATE | TIME | INTERVAL_DAY_TO_SECOND | INTERVAL_YEAR_TO_MONTH | UNKNOWN | INTEGER`

Obligatoire : non

TimeSeriesMeasureValueColumnInfo

Indique si la colonne est un type de données de série chronologique.

Type : objet ColumnInfo

Obligatoire : non

consultez aussi

Pour plus d'informations sur son utilisation API dans l'une des langues spécifiques AWS SDKs, consultez ce qui suit :

- AWS SDKpour C++
- AWS SDKpour Java V2
- AWS SDKpour Ruby V3

Erreurs courantes

Cette section répertorie les erreurs communes aux API actions de tous les AWS services. Pour les erreurs spécifiques à une API action pour ce service, consultez la rubrique consacrée à cette API action.

AccessDeniedException

Vous ne disposez pas d’un accès suffisant pour effectuer cette action.

HTTPCode de statut : 400

IncompleteSignature

La signature de la demande n'est pas conforme aux AWS normes.

HTTPCode de statut : 400

InternalFailure

Le traitement de la demande a échoué en raison d'une erreur, d'une exception ou d'un échec inconnu.

HTTPCode de statut : 500

InvalidAction

L'action ou l'opération demandée n'est pas valide. Vérifiez que l’action est entrée correctement.

HTTPCode de statut : 400

InvalidClientTokenId

Le certificat X.509 ou AWS l'ID de clé d'accès fourni n'existe pas dans nos archives.

HTTPCode de statut : 403

NotAuthorized

Vous ne disposez pas de l’autorisation nécessaire pour effectuer cette action.

HTTPCode de statut : 400

OptInRequired

L'ID de clé d' AWS accès nécessite un abonnement au service.

HTTPCode de statut : 403

RequestExpired

La demande est parvenue au service plus de 15 minutes après l'horodatage sur la demande ou plus de 15 minutes après la date d'expiration de la demande (par exemple pour les demandes pré-signéesURLs), ou le horodatage sur la demande est daté dans plus de 15 minutes dans le futur.

HTTPCode de statut : 400

ServiceUnavailable

La demande a échoué en raison d’une défaillance temporaire du serveur.

HTTPCode de statut : 503

ThrottlingException

La demande a été refusée suite à une limitation des demandes.

HTTPCode de statut : 400

ValidationError

L'entrée ne satisfait pas les contraintes spécifiées par un AWS service.

HTTPCode de statut : 400

Paramètres communs

La liste suivante contient les paramètres que toutes les actions utilisent pour signer les demandes Signature Version 4 à l'aide d'une chaîne de requête. Tous les paramètres spécifiques d'une action particulière sont énumérées dans le sujet consacré à cette action. Pour plus d'informations sur la

version 4 de Signature, consultez la section AWS APIDemandes de signature dans le guide de IAM l'utilisateur.

Action

Action à effectuer.

Type : chaîne

Obligatoire : oui

Version

APIVersion pour laquelle la demande est écrite, exprimée dans le format YYYY-MM-DD.

Type : chaîne

Obligatoire : oui

X-Amz-Algorithm

Algorithme de hachage que vous avez utilisé pour créer la signature de la demande.

Condition : Spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête HTTP d'autorisation.

Type : chaîne

Valeurs valides : `AWS4-HMAC-SHA256`

Obligatoire : Conditionnelle

X-Amz-Credential

Valeur de la portée des informations d'identification, qui est une chaîne incluant votre clé d'accès, la date, la région cible, le service demandé et une chaîne de terminaison (« aws4_request »). La valeur est exprimée au format suivant : access_key//region YYYYMMDD/service /aws4_request.

Pour plus d'informations, voir Création d'une AWS API demande signée dans le Guide de IAM l'utilisateur.

Condition : Spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête HTTP d'autorisation.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-Date

La date utilisée pour créer la signature. Le format doit être le format de base ISO 8601 (YYYYMMDD« T HHMMSS » « Z »). Par exemple, la date et l'heure suivantes sont une X-Amz-Date valeur valide :`20120325T120000Z`.

Condition : X-Amz-Date est facultative pour toutes les demandes ; elle peut être utilisée pour remplacer la date utilisée pour signer les demandes. Si l'en-tête Date est spécifié au format de base ISO 8601, X-Amz-Date il n'est pas obligatoire. Lorsqu'il X-Amz-Date est utilisé, il remplace toujours la valeur de l'en-tête Date. Pour plus d'informations, consultez la section Éléments d'une signature de AWS API demande dans le Guide de IAM l'utilisateur.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-Security-Token

Le jeton de sécurité temporaire obtenu par un appel à AWS Security Token Service (AWS STS). Pour obtenir la liste des services qui prennent en charge les informations d'identification de sécurité temporaires AWS STS fournies par, consultez Services AWS le guide de IAM l'IAMutilisateur avec lequel ils fonctionnent.

Condition : Si vous utilisez des informations d'identification de sécurité temporaires provenant de AWS STS, vous devez inclure le jeton de sécurité.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-Signature

Spécifie la signature codée en hexadécimal qui a été calculée à partir de la chaîne à signer et de la clé de signature dérivée.

Condition : Spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête HTTP d'autorisation.

Type : chaîne

Obligatoire : Conditionnelle

X-Amz-SignedHeaders

Spécifie tous HTTP les en-têtes inclus dans la demande canonique. Pour plus d'informations sur la spécification d'en-têtes signés, voir Création d'une AWS API demande signée dans le Guide de l'IAMutilisateur.

Condition : Spécifiez ce paramètre lorsque vous incluez des informations d'authentification dans une chaîne de requête plutôt que dans l'en-tête HTTP d'autorisation.

Type : chaîne

Obligatoire : Conditionnelle

Historique du document

Modification	Description	Date
Mise à jour relative à la documentation uniquement	Mise à jour de la rubrique Quotas pour séparer les quotas par défaut des limites du système.	22 octobre 2024
Amazon Timestream prend désormais en charge l'analyse des requêtes	Timestream inclut désormais la prise en charge de la fonctionnalité d'analyse des requêtes qui vous aide à optimiser vos requêtes, à améliorer leurs performances et à réduire les coûts.	22 octobre 2024
Amazon Timestream pour InfluxDB met à jour une politique existante.	Amazon Timestream for InfluxDB a ajouté l'action à `ec2:DescribeRouteTables` la politique gérée `AmazonTimestreamInfluxDBFullAccess` existante pour décrire vos tables de routage	8 octobre 2024

AmazonTimestreamReadOnlyAccess — Mise à jour d'une politique existante	Timestream for LiveAnalytics a ajouté l'DescribeAccountSettings autorisation à la politique AmazonTimestreamReadOnlyAccess gérée pour décrire Compte AWS les paramètres.	3 juin 2024
Amazon Timestream prend actuellement en charge LiveAnalytics les unités de calcul Timestream () TCUs	Amazon Timestream inclut actuellement la prise en charge LiveAnalytics des unités de calcul Timestream TCUs () afin de mesurer la capacité de calcul allouée aux besoins de vos requêtes.	29 avril 2024
Nouvelles politiques ajoutées	Amazon Timestream for InfluxDB a ajouté deux nouvelles politiques : l'une permet au service de gérer les interfaces réseau et les groupes de sécurité de votre compte. Pour plus d'informations, consultez AmazonTimestreamInfluxDBServiceRolePolicy. Un autre qui fournit un accès administratif complet pour créer, mettre à jour, supprimer et répertorier les instances Amazon Timestream InfluxDB, ainsi que pour créer et répertorier des groupes de paramètres. Pour plus d'informations, consultez AmazonTimestreamInfluxDBFullAccess.	14 mars 2024

Amazon Timestream pour InfluxDB est désormais disponible pour tous.	Cette documentation couvre la version initiale d'Amazon Timestream pour InfluxDB.	14 mars 2024
Les événements Amazon Timestream LiveAnalytics for Query sont disponibles dans AWS CloudTrail	Amazon Timestream publie pour l'instant LiveAnalytics les événements relatifs aux données de API requête sur. AWS CloudTrail Les clients peuvent auditer toutes les API demandes de requête effectuées dans leurs AWS comptes et consulter des informations telles que l'IAMutil isateur/le rôle qui a fait la demande, la date à laquelle la demande a été faite, les bases de données et les tables interrogées, ainsi que l'ID de requête de la demande.	12 septembre 2023
Amazon Timestream pour LiveAnalytics UNLOAD	Amazon Timestream UNLOAD prend actuellement en charge l'exportation LiveAnalytics des résultats des requêtes vers S3.	12 mai 2023
Amazon Timestream LiveAnalytics pour la mise à jour d'une politique existante.	Des autorisations de chargement par lots ont été ajoutées à une politique gérée.	24 février 2023
Amazon Timestream LiveAnalytics pour le chargement par lots.	Amazon Timestream prend actuellement en charge la fonctionnalité LiveAnalytics de chargement par lots.	24 février 2023

Amazon Timestream LiveAnalytics prend actuellement en charge. AWS Backup	Amazon Timestream LiveAnalytics prend actuellement en charge. AWS Backup	14 décembre 2022
Amazon Timestream LiveAnalytics pour les mises à jour des politiques gérées AWS	De nouvelles informations sur les politiques AWS gérées et Amazon Timestream LiveAnalytics pour, notamment des mises à jour des politiques gérées existantes.	29 novembre 2021
Amazon Timestream LiveAnalytics pour prend en charge les requêtes planifiées	Amazon Timestream prend actuellement en charge LiveAnalytics l'exécution d'une requête en votre nom, selon un calendrier.	29 novembre 2021
Amazon Timestream LiveAnalytics pour les supports Magnetic Store.	Amazon Timestream prend actuellement en charge l'utilisation du stockage magnétique LiveAnalytics pour vos écritures de table.	29 novembre 2021
Amazon Timestream LiveAnalytics pour les enregistrements à mesures multiples.	Amazon Timestream prend actuellement en charge un format plus compact LiveAnalytics pour stocker vos données de séries chronologiques.	29 novembre 2021
Amazon Timestream LiveAnalytics pour les mises à jour des politiques gérées AWS	De nouvelles informations sur les politiques AWS gérées et Amazon Timestream LiveAnalytics pour, notamment des mises à jour des politiques gérées existantes.	24 mai 2021

Amazon Timestream LiveAnalytics for est désormais disponible dans la région Europe (Francfort).	Amazon Timestream LiveAnalytics pour est désormais généralement disponible dans la région Europe (Francfort) (). `eu-central-1`	23 avril 2021
Amazon Timestream LiveAnalytics for is VPC prend désormais en charge les points de terminaison ().AWS PrivateLink	Amazon Timestream prend actuellement en charge l'utilisation LiveAnalytics de points VPC de terminaison ().AWS PrivateLink	23 mars 2021
Amazon Timestream prend désormais en charge les requêtes entre tables.	Vous pouvez utiliser Amazon Timestream LiveAnalytics pour exécuter des requêtes entre tables.	10 février 2021
Amazon Timestream prend actuellement en charge LiveAnalytics les statistiques d'exécution des requêtes améliorées.	Amazon Timestream prend actuellement en charge LiveAnalytics les statistiques d'exécution des requêtes améliorées, telles que la quantité de données numérisées.	10 février 2021
Amazon Timestream prend actuellement en charge les fonctions avancées LiveAnalytics de séries chronologiques.	Vous pouvez utiliser Amazon Timestream LiveAnalytics pour SQL exécuter des requêtes avec des fonctions de séries chronologiques avancées, telles que des dérivées, des intégrales et des corrélations.	10 février 2021

Amazon Timestream LiveAnalytics for est HIPAA désormais conformeISO. PCI	Vous pouvez désormais utiliser Amazon Timestream LiveAnalytics pour les charges de travail qui HIPAA nécessite nt une infrastructure conformeI SO. PCI	27 janvier 2021
Amazon Timestream prend actuellement en charge les logiciels open source LiveAnalytics Telegraf et Grafana.	Vous pouvez désormais utiliser Telegraf, l'agent serveur open source piloté par des plugins pour collecter et générer des rapports de statistiques, et Grafana, la plateforme d'analyse et de surveillance open source pour les bases de données, avec Amazon Timestream pour. LiveAnalytics	25 novembre 2020
Amazon Timestream LiveAnalytics pour est désormais disponible pour tous.	Cette documentation couvre la version initiale d'Amazon LiveAnalytics Timestream pour.	30 septembre 2020

Qu'est-ce que Timestream pour InfluxDB ?

Amazon Timestream pour InfluxDB est un moteur de base de données chronologique géré qui permet aux développeurs d'applications DevOps et aux équipes d'exécuter facilement des bases de données InfluxDB pour des applications de séries chronologiques en temps réel utilisant l'open source. AWS APIs Avec Amazon Timestream pour InfluxDB, il est facile de configurer, d'exploiter et de dimensionner des charges de travail chronologiques capables de répondre à des requêtes avec un temps de réponse à un chiffre en millisecondes.

Amazon Timestream for InfluxDB vous donne accès aux fonctionnalités de la version open source familière d'InfluxDB sur sa branche 2.x. Cela signifie que le code, les applications et les outils que vous utilisez déjà aujourd'hui avec vos bases de données open source InfluxDB existantes devraient fonctionner parfaitement avec Amazon Timestream pour InfluxDB. Amazon Timestream for InfluxDB peut sauvegarder automatiquement votre base de données et maintenir votre logiciel de base de données à jour avec la dernière version. En outre, Amazon Timestream pour InfluxDB facilite l'utilisation de la réplication afin d'améliorer la disponibilité des bases de données et la durabilité des données. Comme pour tous les AWS services, aucun investissement initial n'est requis et vous ne payez que pour les ressources que vous utilisez.

Instances de base de données

Une instance de bases de données est un environnement de base de données isolé s'exécutant dans le cloud. Il s'agit de l'élément de base d'Amazon Timestream pour InfluxDB. Une instance de base de données peut contenir plusieurs bases de données créées par l'utilisateur (ou des organisations et des compartiments dans le cas des bases de données InfluxDb 2.x) et est accessible à l'aide des mêmes outils et applications clients que ceux que vous pourriez utiliser pour accéder à une instance InfluxDB autonome et autogérée. Les instances de base de données sont simples à créer et à modifier à l'aide des outils de ligne de AWS commande, des opérations Amazon Timestream API InfluxDB ou du. AWS Management Console

> **Note**
>
> Amazon Timestream pour InfluxDB prend en charge l'accès aux bases de données à l'aide des opérations Influx et de l'APIinterface utilisateur Influx. Amazon Timestream pour InfluxDB n'autorise pas l'accès direct à l'hôte.

Vous pouvez avoir jusqu'à 40 instances Amazon Timestream pour InfluxDB.

Chaque instance de base de données possède un nom d'instance de base de données. Ce nom fourni par le client identifie de manière unique l'instance de base de données lors de l'interaction avec Amazon Timestream pour InfluxDB et ses commandes. API AWS CLI Le nom de l'instance de base de données doit être unique pour ce client dans une AWS région.

Le nom de l'instance de base de données fait partie du DNS nom d'hôte alloué à votre instance par Timestream pour InfluxDB. Par exemple, si vous spécifiez influxdb1 comme nom d'instance de base de données, Timestream allouera automatiquement un DNS point de terminaison à votre instance. Voici un exemple de point de terminaison : où `influxdb1` est le nom de votre instance. `influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`

Dans l'exemple de point de terminaison`influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`, la chaîne `3ksj4dla5nfjhi` est un identifiant de compte unique généré par AWS. L'identifiant indiqué `3ksj4dla5nfjhi` dans l'exemple ne change pas pour le compte spécifié dans une région donnée. Par conséquent, toutes vos instances de base de données créées par ce compte partagent le même identifiant fixe. Tenez compte des caractéristiques suivantes de cet identifiant fixe :

- Actuellement, Timestream for InfluxDB ne prend pas en charge le renommage des instances de base de données.
- Si vous supprimez et recréez une instance de base de données avec le même identifiant d'instance de base de données, le point de terminaison est le même.
- Si vous utilisez le même compte pour créer une instance de base de données dans une autre région, l'identifiant généré en interne est différent, car la région est différente, comme dans `influxdb2.4a3j5du5ks7md2.us-west-1.timestream-influxdb.amazonaws.com`.

Chaque instance de base de données ne prend en charge qu'un seul moteur de base de données Timestream pour InfluxDB.

Lors de la création d'une instance de base de données, InfluxDB exige qu'un nom d'organisation soit spécifié. Une instance de base de données peut héberger plusieurs organisations et plusieurs buckets associés à chaque organisation.

Amazon Timestream for InfluxDB vous permet de créer un compte utilisateur principal et un mot de passe pour votre instance de base de données dans le cadre du processus de création. Cet utilisateur principal est autorisé à créer des organisations, des compartiments et à effectuer des

opérations de lecture, d'écriture, de suppression et d'insertion sur vos données. Vous pourrez également accéder à InfluxUI et récupérer votre jeton d'opérateur lors de votre première connexion. À partir de là, vous pourrez également gérer tous vos jetons d'accès. Vous devez définir le mot de passe de l'utilisateur principal lorsque vous créez une instance de base de données, mais vous pouvez le modifier à tout moment à l'aide de l'InfluxAPI, de l'Influx CLI ou de l'InfluxUI.

Classes d'instances de base de données

La classe d'instance de base de données détermine le calcul et la capacité de mémoire d'une instance de base de données `fi UbfkyxDB` Amazon Timestream. La classe d'instance de base de données dont vous avez besoin varie selon vos exigences en mémoire et en puissance de traitement.

Une classe d'instance de base de données comprend à la fois le type de classe d'instance de base de données et la taille. Par exemple, `db.influx` est un type de classe d'instance de base de données optimisé pour la mémoire adapté aux exigences de mémoire à hautes performances liées à l'exécution InfluxDb de charges de travail. Dans le type de classe d'`db.influx`instance, `db.influx.2xlarge` se trouve une classe d'instance de base de données. La taille de cette classe est 2 fois plus grande.

Pour plus d'informations sur la tarification des classes d'instances, consultez Amazon Timestream pour connaître les tarifs d'InfluxDB.

Types de classes d'instance de base de données

Amazon Timestream for InfluxDB prend en charge les classes d'instance de base de données pour le cas d'utilisation suivant, optimisées pour les cas d'utilisation d'InfluxDB.

- **`db.influx`**—Ces classes d'instance sont idéales pour exécuter des charges de travail gourmandes en mémoire dans des bases de données InfluxDB open source

Spécifications matérielles pour les classes d'instances de base de données

La terminologie suivante décrit les spécifications matérielles des classes d'instances de base de données :

- v CPU

  Nombre d'unités centrales virtuelles (CPUs). Un virtuel CPU est une unité de capacité que vous pouvez utiliser pour comparer les classes d'instances de base de données.

- Mémoire (Gio)

  LeRAM, en gibioctets, alloué à l'instance de base de données. Il existe souvent un rapport constant entre la mémoire et CPU v. Prenons l'exemple de la classe d'instance db.influx, dont le CPU ratio mémoire/v est similaire à celui de la classe d'instance EC2 r7g.

- Optimisé pour les flux

  Une instance de base de données utilise une pile de configuration optimisée et fournit une capacité supplémentaire dédiée aux I/O. Cette optimisation offre les meilleures performances en minimisant les conflits entre les I/O et le trafic en provenance de votre instance.

- Bande passante du réseau

  Vitesse du réseau par rapport à d'autres classes d'instance de base de données. Dans le tableau suivant, vous trouverez des informations matérielles sur les classes d'instance Amazon Timestream pour InfluxDB.

Classe Instances	v CPU	Mémoire (Gio)	Storage Type	Bande passante réseau (Gbit/s)
db.influx.medium	1	8	Influx IOPS inclus	10
db.influx.large	2	16	Influx IOPS inclus	10
db.influx.xlarge	4	32	Influx IOPS inclus	10
db.influx.2xlarge	8	64	Influx IOPS inclus	10
db.influx.4xlarge	16	128	Influx IOPS inclus	10

Classe Instances	v CPU	Mémoire (Gio)	Storage Type	Bande passante réseau (Gbit/s)
db.influx x 8 x large	32	256	Influx IOPS inclus	12
db.influx 12 x large	48	384	Influx IOPS inclus	20
db.influx 16 x large	64	512	Influx IOPS inclus	25

Stockage d'instance InfluxDB

Les instances de base de données pour Amazon Timestream pour InfluxDB utilisent les volumes IOPS Influx Included pour les bases de données et le stockage des journaux.

Dans certains cas, la charge de travail de votre base de données peut ne pas atteindre 100 % de la charge IOPS que vous avez provisionnée. Pour de plus amples informations, veuillez consulter Autres facteurs ayant un impact sur les performances de stockage. Pour plus d'informations sur la tarification du stockage Timestream pour InfluxDB, consultez les tarifs d'Amazon Timestream.

Amazon Timestream pour les types de stockage InfluxDB

Amazon Timestream pour InfluxDB prend en charge un type de stockage, Influx Included. IOPS Vous pouvez créer un Timestream pour les instances InfluxDB avec un maximum de 16 tebioctets (TiB) de stockage.

Voici une brève description du type de stockage disponible :

- Stockage inclus dans le flux d'E/S : les performances de stockage sont la combinaison des opérations d'E/S par seconde (IOPS) et de la rapidité avec laquelle le volume de stockage peut effectuer des lectures et des écritures (débit de stockage). Sur les volumes de stockage IOPS inclus dans Influx, Amazon Timestream pour InfluxDB fournit 3 niveaux de stockage préconfigurés avec un débit IOPS optimal et requis pour différents types de charges de travail.

Dimensionnement de l'instance InfluxDB

La configuration optimale d'une instance Timestream pour InfluxDB dépend de nombreux facteurs, notamment le taux d'ingestion, la taille des lots, la cardinalité des séries chronologiques, les requêtes simultanées et les types de requêtes. Afin de fournir des recommandations de dimensionnement, nous nous concentrons sur une charge de travail exemplaire présentant les caractéristiques suivantes :

- Les données sont collectées et écrites par une flotte d'agents Telegraf qui collectent le système, la mémoireCPU, le disque, les E/S, etc. à partir d'un centre de données.

  Chaque demande d'écriture contient 5 000 lignes.
- Les types de requêtes exécutées sur le système sont classés dans la catégorie des requêtes « de complexité modérée ». Cette catégorie de requêtes présente les caractéristiques suivantes :
  - Possède plusieurs fonctions et une ou deux expressions régulières
  - Peut également comporter des clauses de regroupement ou un échantillon sur une période de plusieurs semaines.
  - L'exécution prend généralement de quelques centaines à quelques milliers de millisecondes.
  - CPUfavorise principalement les performances des requêtes.

Classe d'instance	Storage Type	Écritures (lignes par seconde)	Lectures (requêtes par seconde)
db.influx.large	Influx IO inclus 3K	~50 000	<10
db.influx.2xlarge	Influx IO inclus 3K	~150 000	<25
db.influx.4xlarge	Influx IO inclus 3K	~200 000	~25
db.influx.4xlarge	Influx IO inclus 12K	~250 000	~35
db.influx x 8 x large	Influx IO inclus 12K	~500 000	~50
db.influx 12 x large	Influx IO inclus 12K	<750 000	<100

AWS Régions et zones de disponibilité

Les ressources de cloud computing Amazon sont hébergées dans plusieurs emplacements à travers le monde. Ces emplacements sont composés de AWS régions et de zones de disponibilité. Chaque AWS région est une zone géographique distincte. Chaque AWS région possède plusieurs emplacements isolés appelés zones de disponibilité.

> **Note**
>
> Pour plus d'informations sur la recherche des zones de disponibilité d'une AWS région, consultez Régions et zones dans le guide de EC2 l'utilisateur Amazon.

Amazon Timestream pour InfluxDB vous permet de placer des ressources, telles que des instances de base de données, et des données à plusieurs endroits.

Amazon exploite state-of-the-art des centres de données hautement disponibles. Bien qu'elles soient rares, des pannes touchant la disponibilité des instances de base de données se trouvant au même emplacement peuvent se produire. Si vous hébergez toutes vos instances de base de données dans un seul emplacement touché par une panne de ce type, aucune de vos instances de base de données ne sera disponible.

Il est important de se rappeler que chaque AWS région est totalement indépendante. Toute activité Amazon Timestream for InfluxDB que vous initiez (par exemple, la création d'instances de base de données ou la liste des instances de base de données disponibles) s'exécute uniquement dans votre région par défaut actuelle. AWS La région AWS par defaut peut être modifiée dans la console ou en définissant la variable d'environnement `AWS_DEFAULT_REGION`. Il peut également être remplacé en utilisant le `--region` paramètre avec le AWS Command Line Interface ()AWS CLI. Pour plus d'informations, consultez la section Configuration des AWS Command Line Interface, en particulier les sections relatives aux variables d'environnement et aux options de ligne de commande.

Pour créer ou utiliser une instance de base de données Amazon Timestream pour InfluxDB dans une région AWS spécifique, utilisez le point de terminaison de service régional correspondant.

AWS Disponibilité de la région

Pour plus d'informations sur AWS les régions dans lesquelles Amazon Timestream pour InfluxDB est actuellement disponible et sur le point de terminaison de chaque région, consultez la section Points de terminaison et quotas Amazon Timestream.

AWS Design des régions

Chaque AWS région est conçue pour être isolée des autres AWS régions. Cette conception permet d'atteindre la plus grande tolérance aux pannes possible et une stabilité optimale.

Lorsque vous consultez vos ressources, seules les ressources liées à la AWS région que vous avez spécifiée s'affichent. Cela est dû au fait que les AWS régions sont isolées les unes des autres et que nous ne répliquons pas automatiquement les ressources entre AWS les régions.

AWS Zones de disponibilité

Lorsque vous créez une instance de base de données, Amazon Timestream pour InfluxDB en choisit une pour vous de manière aléatoire en fonction de la configuration de votre sous-réseau. Une zone de disponibilité est représentée par un code de AWS région suivi d'une lettre d'identification (par exemple,`us-east-1a`).

Utilisez la EC2 commande Amazon `describe-availability` -zones comme suit pour décrire les zones de disponibilité activées pour votre compte dans la région spécifiée.

```
aws ec2 describe-availability-zones --region region-name
```

Par exemple, pour décrire les zones de disponibilité de la région USA Est (Virginie du Nord) (us-east-1) activées pour votre compte, exécutez la commande suivante :

```
aws ec2 describe-availability-zones --region us-east-1
```

Vous ne pouvez pas choisir les zones de disponibilité pour les instances de base de données principale et secondaire dans un déploiement de base de données multi-AZ. Amazon Timesteram for InfluxDB les choisit pour vous de manière aléatoire. Pour plus d'informations sur les déploiements multi-AZ, voir. Configuration et gestion d'un déploiement multi-AZ

Facturation des instances de base de données pour Amazon Timestream pour InfluxDB

Les instances Amazon Timestream pour InfluxDB sont facturées sur la base des composants suivants :

- Heures d'instance de base de données (par heure) : en fonction de la classe d'instance de base de données de l'instance de base de données, par exemple db.influx.large. La tarification est

indiquée selon une base horaire, mais les factures sont calculées à la seconde près et affichent les heures sous une forme décimale. L'utilisation d'Amazon Timestream pour InfluxDB est facturée par tranches d'une seconde, avec un minimum de 10 minutes. Pour plus d'informations, consultez la section Classes d'Classes d'instances de base de données instances de base de données.

- Stockage (par GiB par mois) : capacité de stockage que vous avez allouée à votre instance de base de données. Pour de plus amples informations, veuillez consulter Stockage d'instance InfluxDB.
- Transfert de données (par Go) — Transfert de données vers et depuis votre instance de base de données depuis ou vers Internet et d'autres AWS régions.

Pour obtenir des informations sur les tarifs d'Amazon Timestream pour InfluxDB, consultez la page de tarification d'Amazon Timestream pour InfluxDB.

Configuration d'Amazon Timestream pour InfluxDB

Avant d'utiliser Amazon Timestream pour InfluxDB pour la première fois, effectuez les tâches suivantes :

Si vous avez déjà un AWS compte, connaissez les exigences d'Amazon Timestream pour InfluxDB, et préférez utiliser les valeurs par défaut et IAM Getting VPC Commencer à utiliser Timestream pour InfluxDB Started with Amazon Timestream for InfluxDB.

Créez un AWS compte

Si vous n'avez pas de AWS compte, suivez les étapes ci-dessous pour en créer un.

Pour créer un AWS compte

- Accédez à la page de AWS connexion.
- Choisissez Créer un nouveau compte, puis suivez les instructions.

> **Note**
>
> Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous créez un AWS compte, un AWS utilisateur root est créé. L'utilisateur root a accès à tous les AWS services et ressources du compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur administratif, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Gestion des utilisateurs

Création d'un utilisateur administratif

Création d'un utilisateur administratif

Après avoir créé un AWS compte, créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre AWS compte (utilisateur root)

Connectez-vous à la console de AWS gestion en tant que propriétaire du compte en choisissant Utilisateur root et en saisissant l'adresse e-mail de votre AWS compte. Sur la page suivante, saisissez votre mot de passe. Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur root, consultez la section Connexion en tant qu'utilisateur root dans le Guide de l'utilisateur de AWS connexion

Activez l'authentification multifactorielle (MFA) pour votre utilisateur root. Pour obtenir des instructions, voir Activer un MFA périphérique virtuel pour l'utilisateur root de votre AWS compte (console) dans le guide de IAM l'utilisateur.

Accorder un accès programmatique

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes :

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
Identité du personnel (utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions relatives à l'interface que vous souhaitez utiliser.* Pour le AWS CLI, voir Configuration de l'Identity Center AWS CLI pour utiliser AWS IAM Identity Center dans le Guide de l’utilisateur de l’interface de ligne de commande AWS .* Pour AWS SDKs les outils et AWS APIs, voir IAMAuthentification Identity Center dans le AWSSDKset guide de référence sur les outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLISDKs, etAPIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec les AWS ressources du Guide de IAM l'utilisateur.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer les demandes programmatiques adressées aux AWS CLISDKs, etAPIs.	En suivant les instructions relatives à l'interface que vous souhaitez utiliser.Pour le AWS CLI, voir

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
		Authentification à l'aide des informations IAM d'identification utilisateur dans le AWS Guide de l'utilisateur de l'interface de ligne de commande .Pour AWS SDKs et outils, voir Authentifiez-vous à l'aide d'informations d'identification dans le AWS SDKsGuide de référence et d'outils . Pour AWS APIs, voir Gestion des clés d'accès pour IAM les utilisateurs dans le IAM Guide de l’utilisateur.

Déterminer les exigences

L'élément de base d'Amazon Timestream for Influx est l'instance de base de données. Dans une instance de base de données, vous créez vos buckets. Une instance de base de données fournit une adresse réseau appelée point de terminaison. Vos applications utilisent ce point de terminaison pour se connecter à votre instance de base de données. Vous accéderez également à votre InfluxUI en utilisant ce même point de terminaison depuis votre navigateur. Lorsque vous créez une instance de base de données, vous spécifiez des détails tels que le stockage, la mémoire, le moteur et la version de base de données, la configuration réseau et la sécurité. Votre contrôlez l'accès réseau à une instance de base de données via un groupe de sécurité.

Avant de créer une instance de base de données et un groupe de sécurité, vous devez connaître les besoins en termes d'instances de base de données et de réseau. Voici quelques éléments importants à prendre en compte :

- Besoins en ressources — Quelles sont les exigences en matière de mémoire et de processeur pour votre application ou votre service ? Vous utilisez ces paramètres pour déterminer plus facilement la classe d'instance de base de données à utiliser. Pour les spécifications relatives aux classes d'instance de base de données, consultez la section Classes d'instance de base de données.
- VPCet groupe de sécurité : votre instance de base de données se trouvera très probablement dans un cloud privé virtuel (VPC). Pour vous connecter à votre instance de base de données, vous devez configurer des règles de groupes de sécurité. Ces règles sont configurées différemment en fonction du type d'appareil que VPC vous utilisez et de la manière dont vous l'utilisez. Par exemple, vous pouvez utiliser : une valeur par défaut VPC ou définie par l'utilisateurVPC.

  La liste suivante décrit les règles applicables à chaque VPC option :

  - Par défaut VPC : si votre AWS compte possède une valeur par défaut VPC dans la AWS région actuelle, celle-ci VPC est configurée pour prendre en charge les instances de base de données. Si vous spécifiez la valeur par défaut VPC lorsque vous créez l'instance de base de données, assurez-vous de créer un groupe de VPC sécurité qui autorise les connexions entre l'application ou le service et l'instance de base de données Amazon Timestream for InfluxDB. Utilisez l'option Groupe de sécurité sur la VPC console ou AWS CLI pour créer des groupes VPC de sécurité. Pour plus d'informations, consultez Étape 3 : Création d'un groupe VPC de sécurité.
- Défini par l'utilisateur VPC — Si vous souhaitez spécifier une instance définie par l'utilisateur VPC lorsque vous créez une instance de base de données, tenez compte des points suivants :
  - Assurez-vous de créer un groupe de VPC sécurité qui autorise les connexions entre l'application ou le service et l'instance de base de données Amazon Timestream for InfluxDB. Utilisez l'option Groupe de sécurité sur la VPC console ou AWS CLI pour créer des groupes VPC de sécurité. Pour plus d'informations, voir Étape 3 : Création d'un groupe VPC de sécurité.
  - Ils VPC doivent répondre à certaines exigences pour héberger des instances de base de données, par exemple avoir au moins deux sous-réseaux, chacun dans une zone de disponibilité distincte. Pour plus d'informations, consultez Amazon VPC VPCs et Amazon Timestream pour InfluxDB.
- Haute disponibilité : avez-vous besoin d'une assistance en cas de basculement ? Sur Amazon Timestream pour InfluxDB, un déploiement multi-AZ crée une instance de base de données principale et une instance de base de données de secours secondaire dans une autre zone de

disponibilité pour la prise en charge du basculement. Nous recommandons les déploiements multi-AZ pour les charges de travail de production afin de maintenir une haute disponibilité. À des fins de développement et de test, vous pouvez utiliser un déploiement qui n'est pas multi-AZ. Pour de plus amples informations, veuillez consulter Déploiements d'instances de base de données multi-AZ.

- IAMpolitiques — Votre AWS compte dispose-t-il de politiques qui accordent les autorisations nécessaires pour effectuer des opérations Amazon Timestream pour InfluxDB ? Si vous vous connectez à AWS l'aide IAM d'informations d'identification, votre IAM compte doit disposer de IAM politiques accordant les autorisations requises pour effectuer les opérations du plan de contrôle Amazon Timestream for InfluxDB. Pour de plus amples informations, veuillez consulter Identity and Access Management pour Amazon Timestream pour InfluxDB.
- Ports ouverts : sur quel port TCP /IP votre base de données écoute-t-elle ? Dans certaines entreprises, les pare-feu peuvent bloquer les connexions vers le port par défaut de votre moteur de base de données. La valeur par défaut pour Timestream pour InfluxDB est 8086.
- AWS Région — Dans quelle AWS région souhaitez-vous disposer de votre base de données ? La proximité entre votre base de données et votre application ou le service Web service permet de réduire la latence du réseau. Pour de plus amples informations, veuillez consulter AWS Régions et zones de disponibilité.
- Sous-système de disque de base de données : quels sont vos besoins en matière de stockage ? Amazon Timestream pour InfluxDB fournit trois configurations pour ce type de stockage Influx Included : IOPS
  - Influx Io inclus (3 kIOPS) SSD
  - Influx IO inclus 12k IOPS () SSD
  - Influx Io inclus (25 000€IOPS) SSD

  Pour plus d'informations sur Amazon Timestream pour le stockage InfluxDB, consultez Amazon Timestream pour le stockage d'instances de base de données InfluxDB. Lorsque vous disposez de toutes les informations nécessaires pour créer le groupe de sécurité et l'instance de base de données, passez à l'étape suivante.

Donnez accès à votre instance de base VPC de données dans votre

VPCles groupes de sécurité fournissent un accès aux instances de base de données dans unVPC. Ils font office de pare-feu pour l'instance de base de données associée, en contrôlant le trafic entrant et le trafic sortant au niveau de l'instance de base de données. Par défaut, les instances de base de

données sont créées avec un pare-feu et un groupe de sécurité par défaut protégeant l'instance de base de données.

Avant de pouvoir vous connecter à votre instance de base de données, vous devez ajouter des règles à un groupe de sécurité qui vous permettent de vous connecter. Utilisez vos informations réseau et de configuration pour créer les règles autorisant l'accès à votre instance de base de données.

Supposons, par exemple, qu'une application accède à une base de données sur votre instance de base de données dans unVPC. Dans ce cas, vous devez ajouter une TCP règle personnalisée qui spécifie la plage de ports et les adresses IP que votre application utilise pour accéder à la base de données. Si vous avez une application sur une EC2 instance Amazon, vous pouvez utiliser le groupe de sécurité que vous avez configuré pour l'EC2instance Amazon.

Création d'un groupe de sécurité pour VPC l'accès

Pour créer un groupe VPC de sécurité, connectez-vous au AWS Management Console et choisissez VPC.

> **Note**
>
> Assurez-vous que vous êtes dans la VPC console, et non dans la console Amazon Timesteam for InfluxDB.

- Dans le coin supérieur droit du AWS Management Console, choisissez la AWSrégion dans laquelle vous souhaitez créer votre groupe de VPC sécurité et votre instance de base de données. Dans la liste des VPC ressources Amazon pour cette AWS région, vous devriez voir au moins un ou VPC plusieurs sous-réseaux. Si ce n'est pas le cas, il n'y a pas de valeur par défaut VPC dans cette AWS région. .
- Dans le panneau de navigation, choisissez Security Groups (Groupes de sécurité).
- Sélectionnez Create security group (Créer un groupe de sécurité).
- Dans la section Détails de base de la page du groupe de sécurité, entrez le nom et la description du groupe de sécurité. Pour VPC, choisissez VPC celui dans lequel vous souhaitez créer votre instance de base de données.
- Dans Inbound rules (Règles entrantes), choisissez Add rule (Ajouter une règle).
  - Dans Type, choisissez Personnalisé TCP.

- Pour Source, choisissez un nom de groupe de sécurité ou entrez la plage d'adresses IP (CIDRvaleur) à partir de laquelle vous accédez à l'instance de base de données. Si vous choisissez Mon IP, l'accès à l'instance de base de données est autorisé à partir de l'adresse IP détectée dans votre navigateur.

  Pour Source, choisissez un nom de groupe de sécurité ou saisissez la plage d'adresses IP (CIDRvaleur) à partir de laquelle vous accédez à l'instance de base de données. Si vous choisissez Mon IP, l'accès à l'instance de base de données est autorisé à partir de l'adresse IP détectée dans votre navigateur.
- (Facultatif) Dans Outbound rules (Règles sortantes), ajoutez des règles pour le trafic sortant. Par défaut, tous les trafics sortant sont autorisés.
- Sélectionnez Créer un groupe de sécurité.

Vous pouvez utiliser ce groupe VPC de sécurité comme groupe de sécurité pour votre instance de base de données lorsque vous la créez.

> **Note**
>
> Si vous utilisez une valeur par défaut VPC, un groupe de sous-réseaux par défaut couvrant tous les sous-réseaux est créé pour vous. VPC Lorsque vous créez une instance de base de données, vous pouvez choisir l'eiifccntf par défaut VPC et choisir la valeur par défaut pour le groupe de sous-réseaux de base de données.

Une fois que vous avez terminé les exigences de configuration, vous pouvez créer une instance de base de données en utilisant votre configuration et votre groupe de sécurité. Pour ce faire, suivez les instructions dans Création d'une instance de base de données.

Commencer à utiliser Timestream pour InfluxDB

Dans les exemples suivants, vous découvrirez comment créer et vous connecter à une instance de base de données à l'aide d'Amazon Timestream for InfluxDB Service.

> **Note**
>
> Vous devez réaliser les tâches de la section Configuration d'Amazon Timestream pour InfluxDB avant de créer une instance de base de données ou de vous y connecter.

Rubriques

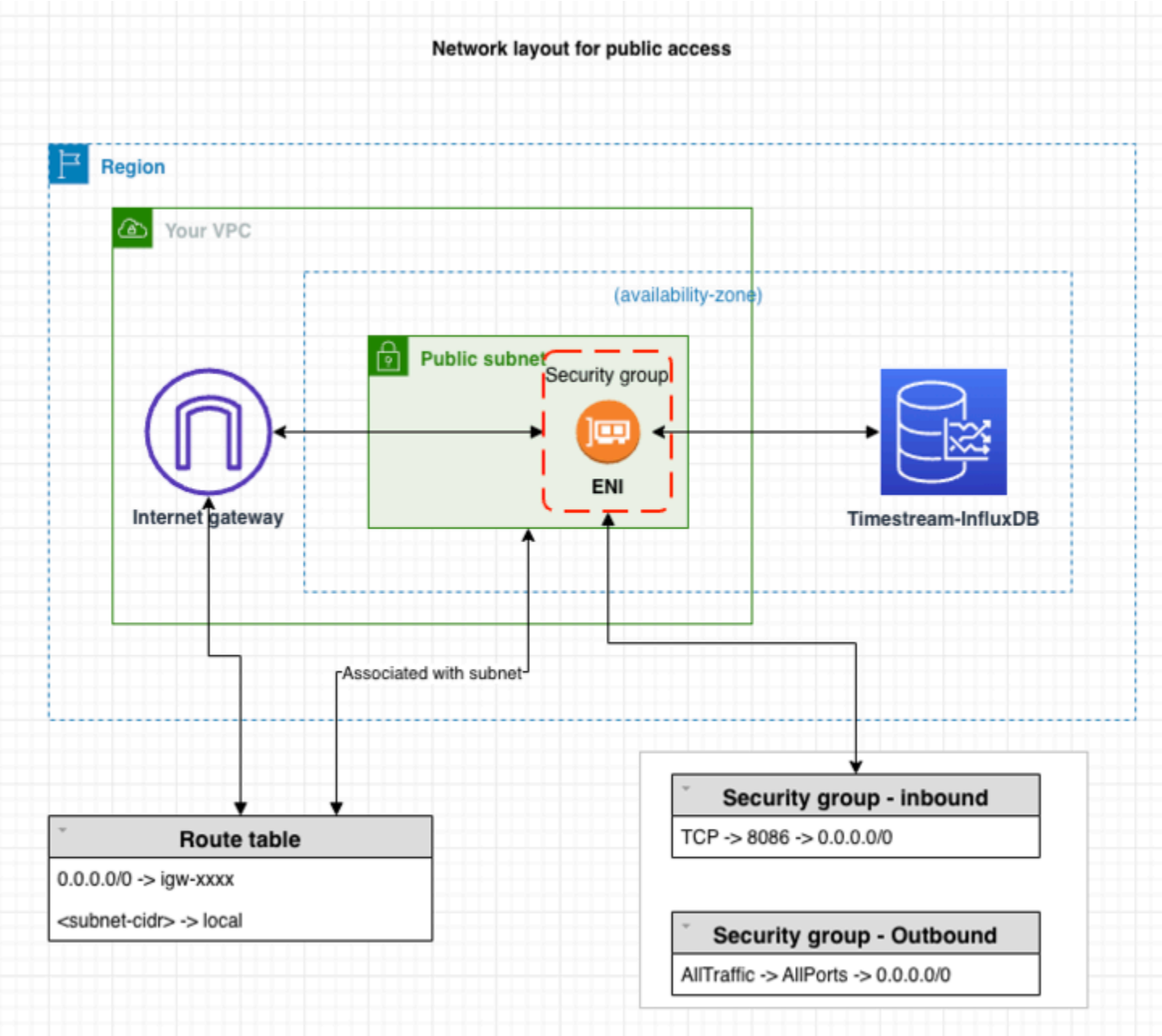
Création et connexion à une instance Timestream pour InfluxDB

Ce didacticiel crée une EC2 instance Amazon et une instance de base de données Amazon Timestream pour InfluxDB. Le didacticiel vous montre comment écrire des données sur l'instance de base de données à partir de l'EC2instance à l'aide du client Telegraf. À titre de bonne pratique, ce didacticiel crée une instance de base de données privée dans un cloud privé virtuel (VPC). Dans la plupart des cas, d'autres ressources présentes dans la même instanceVPC, telles que EC2 des instances, peuvent accéder à l'instance de base de données, mais les ressources extérieures ne VPC peuvent pas y accéder.

Une fois le didacticiel terminé, il existe un sous-réseau public et un sous-réseau privé dans chaque zone de disponibilité de votreVPC. Dans une zone de disponibilité, l'EC2instance se trouve dans le sous-réseau public et l'instance de base de données dans le sous-réseau privé.

> **Note**
>
> La création d'un AWS compte est gratuite. Cependant, en suivant ce didacticiel, les AWS ressources que vous utilisez peuvent vous coûter cher. Vous pouvez supprimer ces ressources après avoir terminé le didacticiel si elles ne sont plus nécessaires.

Le schéma suivant montre la configuration lorsque l'accessibilité est publique.



Warning

Nous ne recommandons pas d'utiliser 0.0.0.0/0 pour l'HTTPaccès, car vous permettez à toutes les adresses IP d'accéder à votre instance publique InfluxDB via. HTTP Cette approche n'est même pas acceptable pendant une courte période dans un environnement de test. Autorisez uniquement une adresse IP ou une plage d'adresses spécifique pour accéder à vos instances InfluxDB en utilisant HTTP Being for WebUI ou Access. API

Ce didacticiel crée une instance de base de données exécutant InfluxDB avec le. AWS Management Console Nous nous concentrerons uniquement sur la taille de l'instance de base de données et sur l'identifiant de l'instance de base de données. Nous utiliserons les paramètres par défaut pour les autres options de configuration. L'instance de base de données créée par cet exemple sera privée.

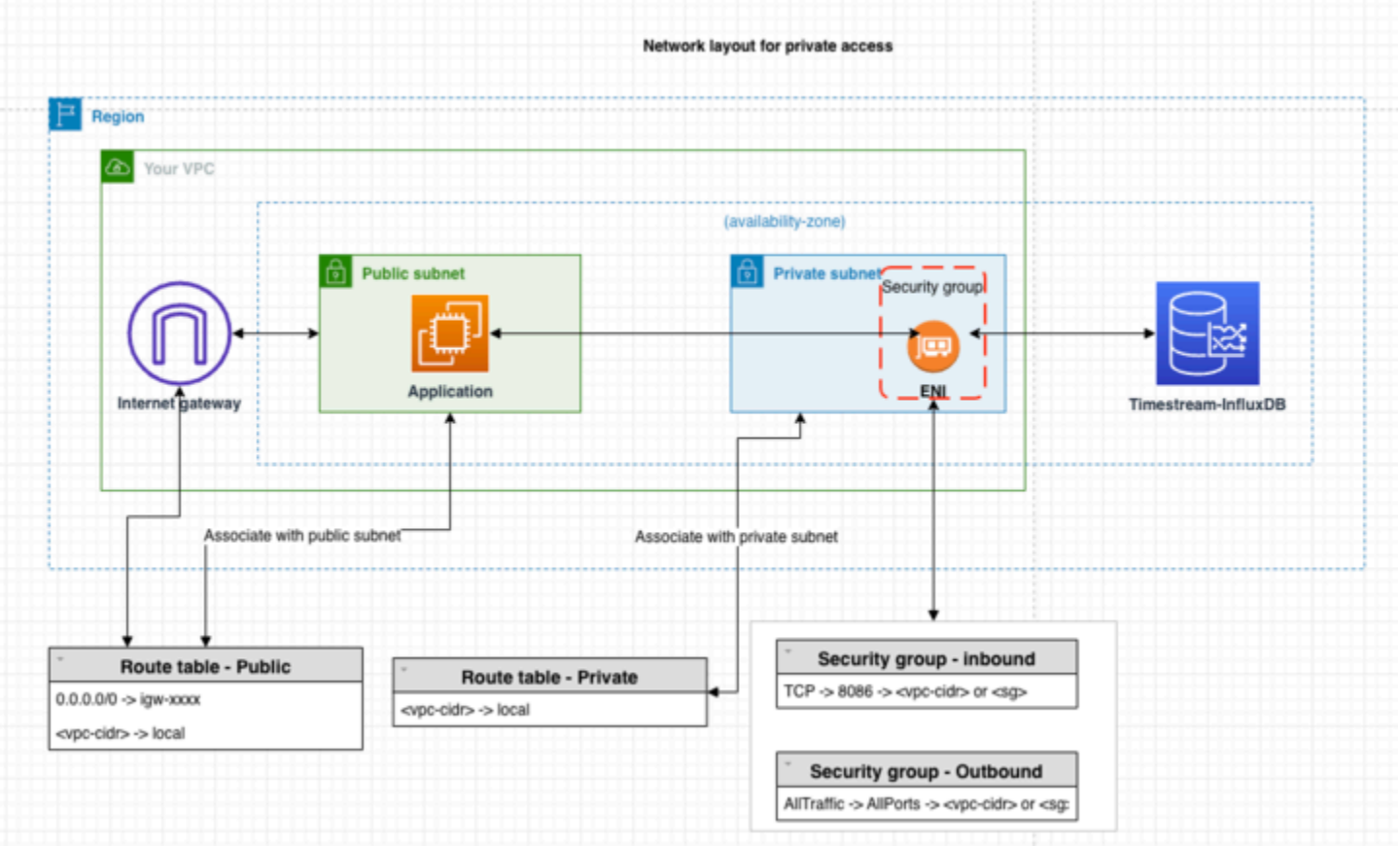
Les autres paramètres que vous pouvez configurer incluent la disponibilité, la sécurité et la journalisation. Pour créer une instance de base de données publique, vous devez choisir de rendre votre instance « accessible au public » dans la section Configuration de la connectivité. Pour plus d'informations sur la création d'instances de base de données, consultezCréation d'une instance de base de données..

Si votre instance n'est pas accessible au public, procédez comme suit :

- Créez un hôte sur VPC l'instance via lequel vous pouvez canaliser le trafic.
- Configurez le tunneling SSH vers l'instance. Pour plus d'informations, consultez Amazon EC2 instance port forwarding with AWS Systems Manager
- Pour que le certificat fonctionne, ajoutez la ligne suivante au `/etc/hosts` fichier de votre machine cliente : `127.0.0.1` Il s'agit de l'DNSadresse de votre instance.
- Connectez-vous à votre instance à l'aide du nom de domaine complet, par exemple https : //< DNS >:8086.

> **Note**
>
> Localhost n'est pas en mesure de valider le certificat car localhost n'en fait pas partie. SAN

Le schéma suivant montre la configuration lorsque l'accessibilité est privée :



Prérequis

Avant de commencer, suivez les étapes détaillées dans les sections suivantes :

- Ouvrez un AWS compte.
- Créez un utilisateur administratif.

Étape 1 : créer une EC2 instance Amazon

Créez une EC2 instance Amazon que vous utiliserez pour vous connecter à votre base de données.

1. Connectez-vous à la EC2 console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le coin supérieur droit de AWS Management Console, choisissez la AWS région dans laquelle vous souhaitez créer l'EC2instance.
3. Choisissez EC2Dashboard, puis Launch instance.

4. Lorsque la page Lancer une instance s'ouvre, choisissez les paramètres suivants sur la page Lancer une instance.

   a. Sous Nom et balises, dans Nom, saisissez ec2-database-connect.

   b. Sous Images d'applications et de systèmes d'exploitation (Amazon Machine Image), choisissez Amazon Linux, puis Amazon Linux 2023AMI. Conservez les sélections par défaut pour les autres choix.

   c. Sous Instance type (Type d'instance), choisissez t2.micro.

   d. Sous Key pair (login) [Paire de clés (connexion)], choisissez une valeur Key pair name (Nom de paire de clés) pour utiliser une paire de clés existante. Pour créer une nouvelle paire de clés pour l'EC2instance Amazon, choisissez Create new key pair, puis utilisez la fenêtre Create key Pair pour la créer. Pour plus d'informations sur la création d'une nouvelle paire de clés, consultez la section Créer une paire de clés dans le Guide de EC2 l'utilisateur Amazon pour les instances Linux.

   e. Pour Autoriser SSH le trafic dans les paramètres réseau, choisissez la source des SSH connexions à l'EC2instance. Vous pouvez choisir Mon adresse IP si l'adresse IP affichée est correcte pour les SSH connexions. Sinon, vous pouvez déterminer l'adresse IP à utiliser pour vous connecter à EC2 des instances à VPC l'aide de Secure Shell (SSH). Pour déterminer votre adresse IP publique, dans une autre fenêtre ou un autre onglet du navigateur, vous pouvez utiliser le service à l'adresse https://checkip.amazonaws.com. Exemple d'adresse IP : 192.0.2.1/32. Dans de nombreux cas, vous pouvez vous connecter via un fournisseur de services Internet (ISP) ou derrière votre pare-feu sans adresse IP statique. Si tel est le cas, assurez-vous de déterminer la plage d'adresses IP utilisées par les ordinateurs clients.

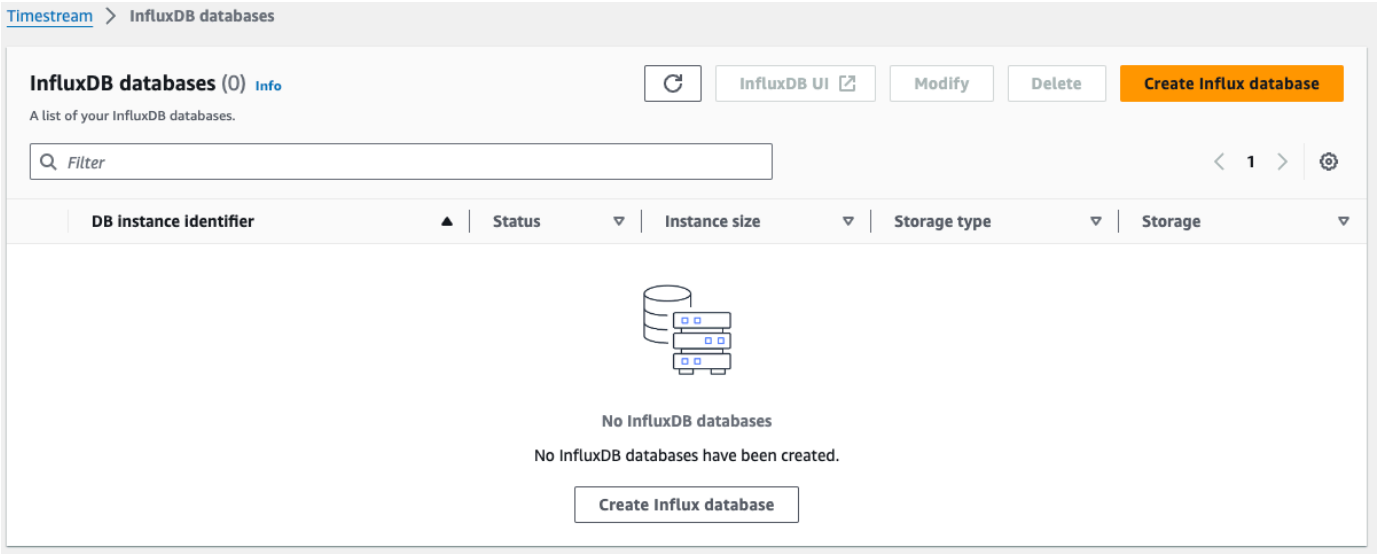
      > ⚠ Warning
      >
      > Nous vous déconseillons d'utiliser 0.0.0.0/0 pour l'SSHaccès, car vous permettez à toutes les adresses IP d'accéder à vos instances publiques EC2 en utilisant. SSH Cette approche n'est même pas acceptable pendant une courte période dans un environnement de test. N'autorisez qu'une adresse IP ou une plage d'adresses spécifiques à accéder à vos EC2 instancesSSH.

Étape 2 : créer une instance de base de données InfluxDB

L'élément de base d'Amazon Timestream pour InfluxDB est l'instance de base de données. C'est dans cet environnement que vous exécutez vos bases de données InfluxDB.

Dans cet exemple, vous allez créer une instance de base de données exécutant le moteur de base de données InfluxDB avec une classe d'instance de base de données db.influx.large.

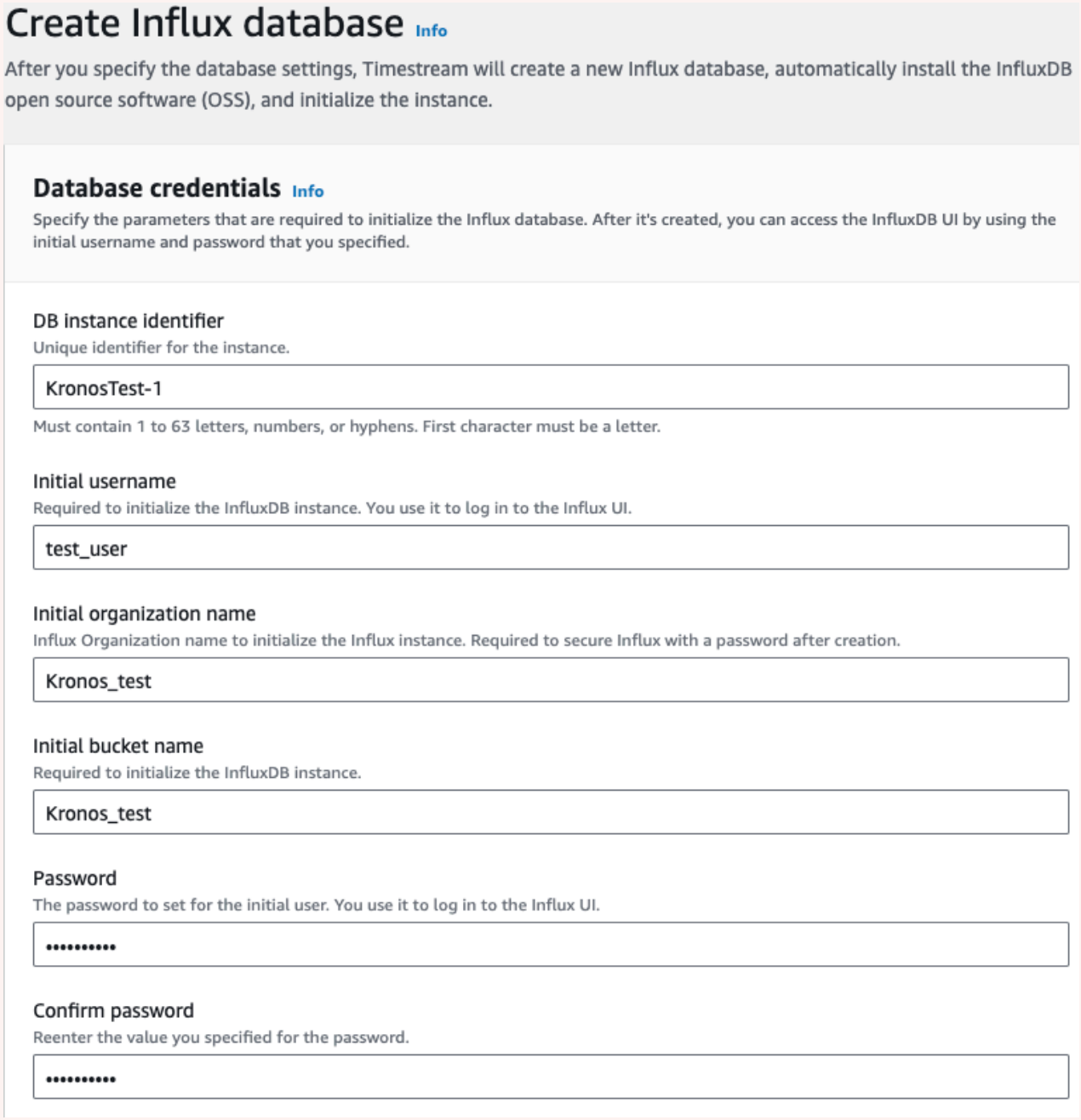
1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/timestream/
2. Dans le coin supérieur droit de la console Amazon Timestream pour InfluxDB, choisissez la région dans laquelle vous souhaitez créer AWS l'instance de base de données.
3. Dans le volet de navigation, choisissez InfluxDB Databases.
4. Choisissez Create Influx database.



5. Pour l'identifiant d'instance de base de données, entrez KronosTest -1.
6. Fournissez les paramètres de configuration de base d'InfluxDB : nom d'utilisateur, organisation, nom du compartiment et mot de passe.

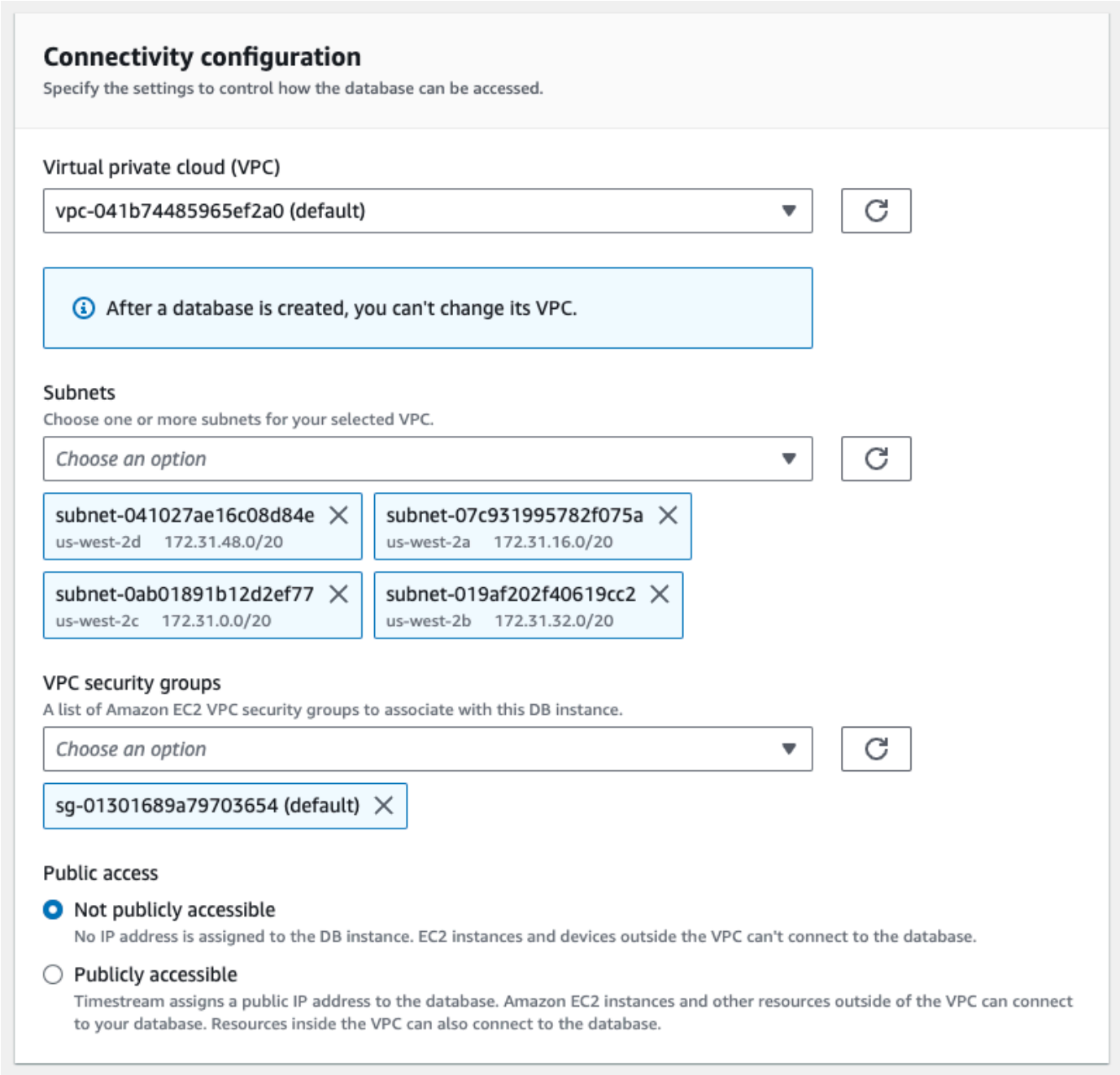
> **Important**
>
> Vous ne pourrez plus consulter le mot de passe utilisateur. Vous ne pourrez pas accéder à votre instance et obtenir un jeton d'opérateur sans votre mot de passe. Si vous ne l'enregistrez pas, il sera peut-être nécessaire de le modifier. Consultez Création d'un nouveau jeton d'opérateur pour votre instance InfluxDB.

Si vous devez modifier le mot de passe utilisateur une fois que l'instance de base de données est disponible, vous pouvez modifier l'instance de base de données pour le faire. Pour plus d'informations sur la modification d'une instance de base de données, veuillez consulter Mise à jour des instances de base de données.



7. Pour la classe d'instance de base de données, sélectionnez db.influx.large.
8. Pour la classe de stockage de base de données, sélectionnez influx IOPS Included 3K.

9. Configurez vos journaux. Pour de plus amples informations, veuillez consulter Configuration pour afficher les journaux InfluxDB sur les instances Timestream Influxdb.
10. Dans la section Configuration de la connectivité, assurez-vous que votre instance InfluxDB se trouve dans le même sous-réseau que votre instance nouvellement créée. EC2



11. Choisissez Create Influx database.
12. Dans la liste des bases de données, choisissez le nom de votre nouvelle instance InfluxDB pour afficher ses détails. L'instance de base de données a le statut Creating jusqu'à ce qu'elle soit prête à être utilisée.

Vous pouvez vous connecter à l'instance de base de données lorsque le statut passe à Disponible. En fonction de la quantité de stockage et de la classe d'instance de base de données, la mise à disposition de la nouvelle instance peut prendre jusqu'à 20 minutes.

> **Important**
>
> Pour le moment, vous ne pouvez pas modifier la configuration de calcul (types d'instances) et de stockage (types de stockage) des instances existantes.

Étape 3 : Envoyer des données Telegraf à votre instance InfluxDB

Vous pouvez maintenant commencer à envoyer des données de télémétrie à votre instance de base de données InfluxDB à l'aide de l'agent Telegraf. Dans cet exemple, vous allez installer et configurer un agent Telegraf pour envoyer des mesures de performance à votre instance de base de données InfluxDB.

1. Trouvez le point de terminaison (DNSnom) et le numéro de port de votre instance de base de données.

   a. Connectez-vous à la console de AWS gestion et ouvrez la console Amazon Timestream à l'adresse. https://console.aws.amazon.com/timestream/

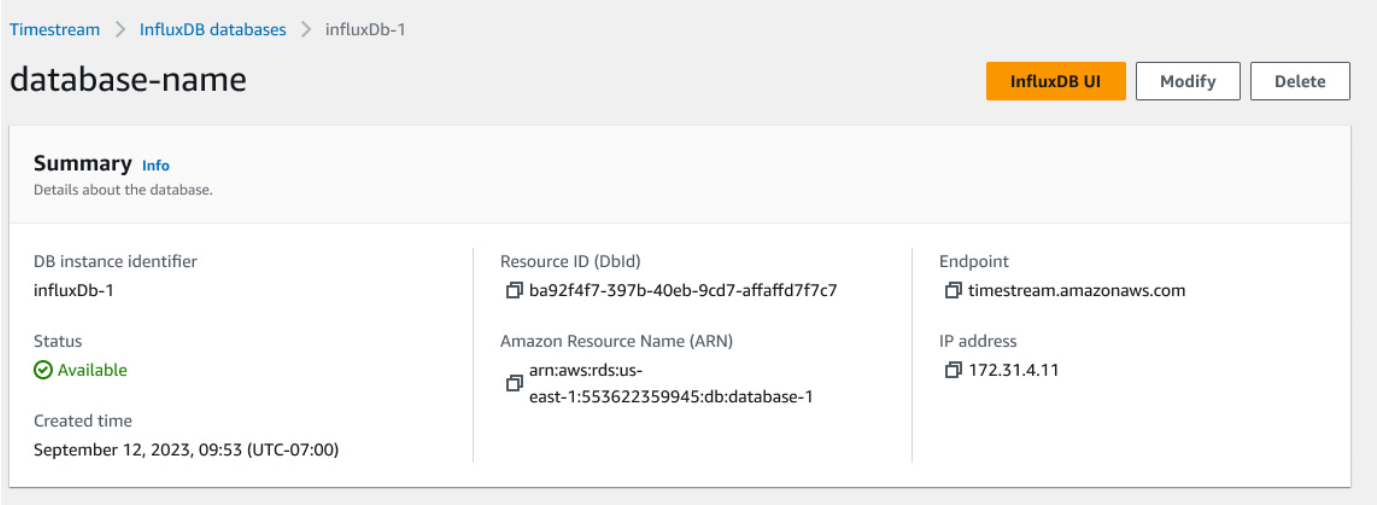
   b. Dans le coin supérieur droit de la console Amazon Timestream, choisissez la région pour l'instance de base AWS de données.

   c. Dans le volet de navigation, choisissez InfluxDB Databases.

   d. Choisissez le nom de l'instance de base de données InfluxDB pour afficher ses détails.

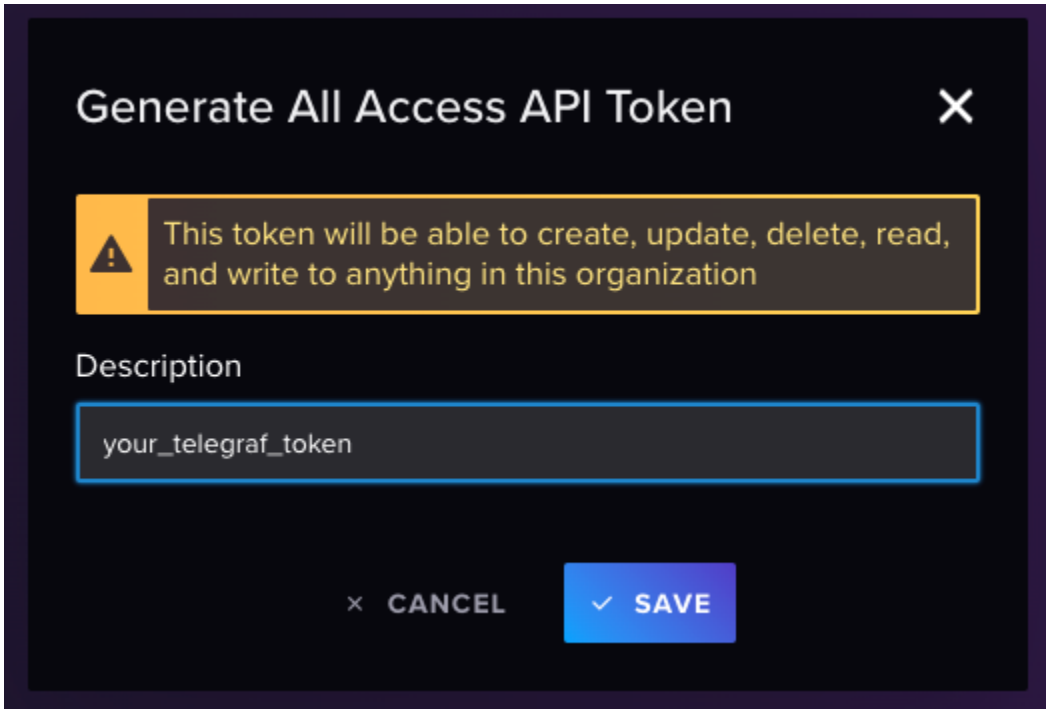
   e. Dans la section Résumé, copiez le point de terminaison. Notez également le numéro du port. Vous avez besoin du point de terminaison et du numéro de port pour vous connecter à l'instance de base de données (le numéro de port par défaut pour InfluxDB est 8086).

2. Ensuite, sélectionnez InfluxDB UI.



3. Cela ouvrira une nouvelle fenêtre de navigateur dans laquelle vous devriez voir une invite de connexion. Entrez les informations d'identification que vous avez utilisées précédemment pour créer votre instance InfluxDB Db.
4. Dans le volet de navigation, cliquez sur la flèche et sélectionnez APITokens.
5. Pour ce test, générez un jeton All Access.

> **Note**
>
> Pour les scénarios de production, nous vous recommandons de créer des jetons avec un accès spécifique aux compartiments requis conçus pour les besoins spécifiques de Telegraf.



6. Votre jeton apparaîtra à l'écran.

> **Important**
>
> Assurez-vous de copier et d'enregistrer le jeton, car vous ne pourrez plus l'afficher.

7. Connectez-vous à l'EC2instance que vous avez créée précédemment en suivant les étapes décrites dans la section Connexion à votre instance Linux dans le Guide de l'EC2utilisateur Amazon pour les instances Linux.

   Nous vous recommandons de vous connecter à votre EC2 instance à l'aide deSSH. Si l'utilitaire SSH client est installé sous Windows, Linux ou Mac, vous pouvez vous connecter à l'instance à l'aide du format de commande suivant :

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Supposons, par exemple, qu'`ec2-database-connect-key-pair.pem`il soit stocké `/dir1` sous Linux et que le public IPv4 DNS de votre EC2 instance l'est`ec2-12-345-678-90.compute-1.amazonaws.com`. Votre SSH commande se présenterait comme suit :

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

8. Installez la dernière version de Telegraf sur votre instance. Pour ce faire, utilisez la commande suivante :

```
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. Configurez votre instance Telegraf.

> **Note**
>
> Si telegraf.conf n'existe pas ou contient une `timestream` section, vous pouvez en générer une avec :
>
> ```
> telegraf –section-filter agent:inputs:outputs –input-filter cpu:mem –output-
> filter timestream config > telegraf.conf
> ```

a. Modifiez le fichier de configuration généralement situé à l'adresse`/etc/telegraf`.

```
sudo nano /etc/telegraf/telegraf.conf
```

b. Configurez les entrées de base pourCPU, MEM etDISK.

```
[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false
```

```
[[inputs.mem]]

[[inputs.disk]]
  ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
```

c. Configurez le plugin Output pour envoyer des données à votre instance de base de données InfluxDB et enregistrer vos modifications.

```
[[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. Configurez la cible 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. Activez et démarrez le service Telegraf.

```
$ sudo systemctl enable telegraf
$ sudo systemctl start telegraf
```

Étape 4 : Supprimer l'EC2instance Amazon et l'instance de base de données InfluxDB

Après avoir exploré les données générées par Telegraff à l'aide de votre instance de base de données InfluxDB avec InfluxUI, supprimez à la fois vos instances de base de données InfluxDB EC2 et vos instances de base de données InfluxDB afin qu'elles ne vous soient plus facturées.

Pour supprimer l'EC2instance :

1. Connectez-vous à la EC2 console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le panneau de navigation, choisissez Instances.
3. Sélectionnez l'EC2instance, choisissez État de l'instance et Terminate instance.
4. Choisissez Résilier lorsque vous êtes invité à confirmer.

Pour plus d'informations sur la suppression d'une EC2 instance, consultez Résilier votre instance dans le guide de EC2 l'utilisateur Amazon.

Pour supprimer l'instance de base de données sans capture d'écran de base de données finale :

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/timestream/
2. Dans le volet de navigation, choisissez InfluxDB Databases.
3. Choisissez l'instance de base de données que vous voulez supprimer.
4. Pour Actions, choisissez Supprimer.
5. Terminez la confirmation et choisissez Supprimer.

(Facultatif) Connectez-vous à votre instance de base de données à l'aide d'Amazon Managed Grafana

Vous pouvez utiliser Amazon Managed Grafana pour créer des tableaux de bord et surveiller les performances de vos EC2 instances à l'aide d'Amazon Timestream pour InfluxDB. Amazon Managed Grafana est un service entièrement géré pour Grafana, une plateforme d'analyse open source populaire qui vous permet d'interroger, de visualiser et d'émettre des alertes sur vos indicateurs, journaux et traces.

Création d'un nouveau jeton d'opérateur pour votre instance InfluxDB

Si vous devez obtenir le jeton d'opérateur pour votre nouvelle instance InfluxDB, effectuez les étapes suivantes :

1. Pour changer votre jeton d'opérateur, nous vous recommandons d'utiliser l'InfluxCLI. Pour obtenir des instructions, veuillez consulter : Installer et utiliser l'influx CLI.
2. Configurez votre `--username-password` appareil CLI à utiliser pour pouvoir créer l'opérateur :

```
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. Créez votre nouveau jeton d'opérateur. Il vous sera demandé votre mot de passe pour confirmer cette étape.

```
influx auth create --org [YOURORG] --operator
```

> **Important**
>
> Une fois qu'un nouveau jeton d'opérateur a été créé, vous devez mettre à jour tous les clients qui utilisent actuellement l'ancien.

Migration des données d'InfluxDB autogéré vers Timestream pour InfluxDB

Le script de migration Influx est un script Python qui migre les données entre les OSS instances d'InfluxDB, que ces instances soient gérées par AWS ou non.

InfluxDB est une base de données de séries chronologiques. InfluxDB contient des points, qui contiennent un certain nombre de paires clé-valeur et un horodatage. Lorsque les points sont regroupés par paires clé-valeur, ils forment une série. Une série est regroupée par un identifiant de chaîne appelé mesure. InfluxDB est souvent utilisé pour la surveillance des opérations, les IOT données et les analyses. Un bucket est une sorte de conteneur au sein d'InfluxDB pour stocker des données. AWS-managed InfluxDB est InfluxDB au sein de l'écosystème. AWS InfluxDB fournit

l'InfluxDB v2 API pour accéder aux données et apporter des modifications à la base de données. L'InfluxDB v2 API est ce que le script de migration Influx utilise pour migrer les données.

- Le script de migration Influx peut faire migrer les buckets et leurs métadonnées, migrer tous les buckets de toutes les organisations ou effectuer une migration complète, qui remplace toutes les données de l'instance de destination.
- Le script sauvegarde les données de l'instance source localement, quel que soit le système qui exécute le script, puis restaure les données sur l'instance de destination. Les données sont conservées dans les<timestamp></timestamp> répertoires code>influxdb-backup-, un pour chaque migration.
- Le script fournit un certain nombre d'options et de configurations, notamment le montage de compartiments S3 pour limiter l'utilisation du stockage local pendant la migration et le choix des organisations à utiliser pendant la migration.

Rubriques

Préparation

La migration des données pour InfluxDB est réalisée à l'aide d'un script Python qui utilise les CLI fonctionnalités d'InfluxDB et l'InfluxDB v2. API L'exécution du script de migration nécessite la configuration d'environnement suivante :

- Versions prises en charge : Une version minimale de 2.3 d'InfluxDB et Influx CLI est prise en charge.
- Variables d'environnement des jetons
  - Créez la variable d'environnement `INFLUX_SRC_TOKEN` contenant le jeton pour votre instance InfluxDB source.
  - Créez la variable d'environnement `INFLUX_DEST_TOKEN` contenant le jeton pour votre instance InfluxDB de destination.
- Python 3
  - Vérifiez l'installation : `python3 --version`

  - Si ce n'est pas le cas, installez-le depuis le site Web de Python. Version 3.7 minimale requise. Sous Windows, l'alias Python 3 par défaut est simplement python.
  - Les requêtes du module Python sont obligatoires. Installez-le avec : `shell python3 -m pip install requests`
  - TTheLe module Python influxdb_client est requis. Installez-le avec : `shell python3 -m pip install influxdb_client`
- InfluxDB CLI
  - Confirmez l'installation : `influx version`
  - S'il n'est pas installé, suivez le guide d'installation de la documentation InfluxDB.

    Ajoutez un afflux à votre $PATH.
- Outils de montage S3 (en option)

  Lorsque le montage S3 est utilisé, tous les fichiers de sauvegarde sont stockés dans un compartiment S3 défini par l'utilisateur. Le montage S3 peut être utile pour économiser de l'espace sur la machine d'exécution ou lorsque des fichiers de sauvegarde doivent être partagés. Si le montage S3 n'est pas utilisé, en omettant `--s3-bucket` cette option, un `influxdb-backup-<millisecond timestamp>` répertoire local sera créé pour stocker les fichiers de sauvegarde dans le même répertoire que celui dans lequel le script a été exécuté.

  Pour Linux : mountpoint-s3.

  Pour Windows : rclone (une configuration préalable de rclone est nécessaire).
- Espace disque
  - Le processus de migration crée automatiquement des répertoires uniques pour stocker des ensembles de fichiers de sauvegarde et conserve ces répertoires de sauvegarde dans S3 ou sur le système de fichiers local, en fonction des arguments du programme fournis.
  - Assurez-vous qu'il y a suffisamment d'espace disque pour la sauvegarde de la base de données. Idéalement, doublez la taille de la base de données InfluxDB existante si vous choisissez d'omettre l'`--s3-bucket`option et d'utiliser le stockage local pour la sauvegarde et la restauration.
  - Vérifiez l'espace avec `df -h (UNIX/Linux)` ou en vérifiant les propriétés du lecteur sous Windows.
- Connexion directe

Assurez-vous qu'une connexion réseau directe existe entre le système exécutant le script de migration et les systèmes source et de destination. `influx ping --host <host>`est un moyen de vérifier une connexion directe.

Comment utiliser le script

Voici un exemple simple d'exécution du script :

```
python3 influx_migration.py --src-host <source host> --src-bucket <source bucket> --
dest-host <destination host>
```

Qui fait migrer un seul bucket.

Toutes les options peuvent être consultées en exécutant :

```
python3 influx_migration.py -h
```

Utilisation

```
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]
```

Options

- -confirm-full (facultatif) : l'utilisation de `--full` without `--csv` remplacera tous les jetons, utilisateurs, compartiments, tableaux de bord et toute autre donnée clé-valeur de la base de données de destination par les jetons, utilisateurs, compartiments, tableaux de bord et toute autre donnée clé-valeur de la base de données source. `--full`avec migre `--csv` uniquement toutes les métadonnées du bucket et du bucket, y compris les organisations du bucket. Cette option (`--confirm-full`) confirmera une migration complète et se poursuivra sans intervention de l'utilisateur. Si cette option n'est pas fournie, qu'`--full`elle a été fournie et `--csv` non fournie, le script s'interrompra pour s'exécuter et attendra la confirmation de l'utilisateur. Il s'agit d'une action critique, procédez avec prudence. La valeur par défaut est false.
- -csv (facultatif) : s'il faut utiliser des fichiers csv pour la sauvegarde et la restauration. S'il `--full` est également adopté, tous les compartiments définis par l'utilisateur de toutes les organisations seront migrés, et non les compartiments système, les utilisateurs, les jetons ou les tableaux

de bord. Si une organisation unique est souhaitée pour tous les compartiments du serveur de destination au lieu de leurs organisations source existantes, utilisez. `--dest-org`

- -dest-bucket DEST _ BUCKET (facultatif) : le nom du bucket InfluxDB sur le serveur de destination ne doit pas être un bucket déjà existant. La valeur par défaut est la valeur de `--src-bucket` ou None si elle `--src-bucket` n'est pas fournie.
- -dest-host DEST _ HOST : L'hôte du serveur de destination. Exemple : http://localhost:8086.
- -dest-org DEST _ ORG (facultatif) : nom de l'organisation vers laquelle restaurer les buckets sur le serveur de destination. Si cette option est omise, tous les buckets migrés depuis le serveur source conserveront leur organisation d'origine et les buckets migrés risquent de ne pas être visibles sur le serveur de destination sans création ou changement d'organisation. Cette valeur sera utilisée dans toutes les formes de restauration, qu'il s'agisse d'un bucket unique, d'une migration complète ou de toute migration utilisant des fichiers CSV pour la sauvegarde et la restauration.
- -dir-name DIR _ NAME (facultatif) : nom du répertoire de sauvegarde à créer. La valeur par défaut est `influxdb-backup-<timestamp>`. Il ne doit pas déjà exister.
- -full (facultatif) : s'il faut effectuer une restauration complète, en remplaçant toutes les données du serveur de destination par toutes les données du serveur source de toutes les organisations, y compris toutes les données clé-valeur telles que les jetons, les tableaux de bord, les utilisateurs, etc. Dérogations `--src-bucket` et. `--dest-bucket` S'il est utilisé avec`--csv`, migre uniquement les données et les métadonnées des buckets. La valeur par défaut est false.
- h, --help : affiche le message d'aide et sort.
- -log-level LOG _ LEVEL (facultatif) : niveau de journalisation à utiliser lors de l'exécution. Les options sont debug, error et info. La valeur par défaut est info.
- - retry-restore-dir RETRY _ RESTORE _ DIR (facultatif) : répertoire à utiliser pour la restauration en cas d'échec d'une restauration précédente, ignorera la sauvegarde et la création de répertoire, échouera si le répertoire n'existe pas, peut être un répertoire dans un compartiment S3. Si une restauration échoue, le chemin du répertoire de sauvegarde qui peut être utilisé pour la restauration sera indiqué par rapport au répertoire en cours. Les compartiments S3 seront sous la forme`influxdb-backups/<s3 bucket>/<backup directory>`. Le nom du répertoire de sauvegarde par défaut est`influxdb-backup-<timestamp>`.
- -s3-bucket S3_ BUCKET (facultatif) : nom du compartiment S3 à utiliser pour stocker les fichiers de sauvegarde. Sous Linux, il s'agit simplement du nom du compartiment S3, par exemple`my-bucket`, donné `AWS_ACCESS_KEY_ID` et les variables d'`AWS_SECRET_ACCESS_KEY`environnement ont été définies ou `${HOME}/.aws/credentials` existent. Sous Windows, il s'agit de la télécommande et du nom du bucket `rclone` configurés, par exemple`my-remote:my-bucket`. Tous les fichiers de sauvegarde seront conservés dans le

compartiment S3 après la migration dans un `influxdb-backups-<timestamp>` répertoire créé. Un répertoire de montage temporaire nommé `influx-backups` sera créé dans le répertoire à partir duquel ce script est exécuté. S'ils ne sont pas fournis, tous les fichiers de sauvegarde seront stockés localement dans un `influxdb-backups-<timestamp>` répertoire créé à partir duquel ce script est exécuté.

- -skip-verify (facultatif) : ignore TLS la vérification du certificat.
- -src-bucket SRC _ BUCKET (facultatif) : nom du bucket InfluxDB sur le serveur source. Si ce n'est pas le cas, il `--full` doit être fourni.
- -src-host SRC _ HOST (facultatif) : hôte du serveur source. La valeur par défaut est http://localhost:8086.

Comme indiqué précédemment, `mountpoint-s3` `rclone` elles sont nécessaires si `--s3-bucket` elles doivent être utilisées, mais peuvent être ignorées si l'utilisateur ne fournit pas de valeur pour`--s3-bucket`, auquel cas les fichiers de sauvegarde seront stockés localement dans un répertoire unique.

Présentation de la migration

Après avoir rempli les conditions préalables :

1. Exécuter le script de migration : à l'aide d'une application de terminal de votre choix, exécutez le script Python pour transférer les données de l'instance InfluxDB source vers l'instance InfluxDB de destination.
2. Fournir des informations d'identification : indiquez les adresses et les ports des hôtes en tant qu'CLIoptions.
3. Vérifier les données : assurez-vous que les données sont correctement transférées en :

    a. Utilisation de l'interface utilisateur InfluxDB et inspection des buckets.

    b. Répertorier les seaux avec`influx bucket list -t <destination token> --host <destination host address> --skip-verify`.

    c. Utilisation `influx v1 shell -t <destination token> --host <destination host address> --skip-verify` et exécution `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>` pour vérifier que le nombre correct d'enregistrements ont été migrés.

Example Exemple de course

1. Ouvrez l'application pour terminal de votre choix et assurez-vous que les prérequis requis sont correctement installés :

```
~ > 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. Accédez au script de migration :

```
~ > 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. Préparez les informations suivantes :

   a. Nom du compartiment source à migrer.

   b. (Facultatif) Choisissez un nouveau nom de compartiment pour le compartiment migré sur le serveur de destination.

   c. Jeton racine pour les instances d'afflux source et de destination.

   d. Adresse hôte des instances d'afflux source et de destination.

   e. (Facultatif) Nom et informations d'identification du compartiment S3 ; les AWS Command Line Interface informations d'identification doivent être définies dans les variables d'environnement du système d'exploitation.

```
# AWS credentials (for timestream testing)
    export AWS_ACCESS_KEY_ID="xxx"
    export AWS_SECRET_ACCESS_KEY="xxx"
```

   f. Construisez la commande comme suit :

```
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. Exécutez le 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. Attendez la fin de l'exécution du script.

i. Vérifiez l'intégrité des données du compartiment récemment migré. `performance.txt` Ce fichier, situé dans le répertoire où le script a été exécuté, contient des informations de base sur la durée de chaque étape.

Scénarios de migration

Example Exemple 1 : migration simple à l'aide du stockage local

Vous souhaitez migrer un seul compartiment, le compartiment principal, du serveur source (`http://localhost:8086`) vers un serveur de destination. (`http://dest-server-address:8086`)

Après avoir vérifié que vous avez TCP accès (pour l'HTTPaccès) aux deux machines hébergeant les instances InfluxDB sur le port 8086, que vous disposez à la fois de jetons source et de destination et que vous les avez stockés en tant que variables d'environnement `INFLUX_SRC_TOKEN` et`INFLUX_DEST_TOKEN`, respectivement, pour plus de sécurité :

```
python3 influx_migration.py --src-bucket primary-bucket --src-host http://
localhost:8086 --dest-host http://dest-server-address:8086
```

La sortie doit ressembler à ce qui suit :

```
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
```

Le répertoire `influxdb-backup-<timestamp>` sera créé et stocké dans le répertoire à partir duquel le script a été exécuté, contenant les fichiers de sauvegarde.

Example Exemple 2 : migration complète à l'aide du stockage local et de la journalisation du débogage

Comme ci-dessus, sauf que vous souhaitez migrer tous les buckets, jetons, utilisateurs et tableaux de bord, supprimer les buckets sur le serveur de destination et procéder sans confirmation par l'utilisateur d'une migration complète de la base de données à l'aide de cette option. `--confirm-full` Vous souhaitez également connaître les mesures de performance afin d'activer la journalisation du débogage.

```
python3 influx_migration.py --full --confirm-full --src-host http://localhost:8086 --
dest-host http://dest-server-address:8086 --log-level debug
```

La sortie doit ressembler à ce qui suit :

```
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 Exemple 3 : utilisation de la migration complèteCSV, organisation de destination et compartiment S3

Identique à l'exemple précédent, mais en utilisant Linux ou Mac et en stockant les fichiers dans le compartiment S3,`my-s3-bucket`. Cela permet d'éviter que les fichiers de sauvegarde ne surchargent la capacité de stockage locale.

```
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
```

La sortie doit ressembler à ce qui suit :

```
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
```

Configuration d'une instance de base de données

Cette section explique comment configurer votre instance de base de données Amazon Timestream pour InfluxDB. Avant de créer une instance de base de données, choisissez la classe d'instance de

base de données qui exécutera l'instance de base de données. Décidez également où l'instance de base de données sera exécutée en choisissant une AWS région. Créez ensuite l'instance de base de données.

Vous pouvez configurer une instance de base de données avec un groupe de paramètres de base de données. Un groupe de paramètres de base de données agit comme un conteneur pour les valeurs de configuration du moteur appliquées à une ou plusieurs instances de base de données.

Les paramètres disponibles dépendent du moteur de base de données et de la version du moteur de base de données. Vous pouvez spécifier un groupe de paramètres de base de données lorsque vous créez une instance de base de données. Vous pouvez également modifier une instance de base de données pour les spécifier.

> **⚠ Important**
>
> Pour le moment, vous ne pouvez pas modifier la configuration de calcul (types d'instances) et de stockage (types de stockage) des instances existantes.

Création d'une instance de base de données

Utilisation de la console

1. Connectez-vous AWS Management Console et ouvrez Amazon Timestream pour InfluxDB.
2. Dans le coin supérieur droit de la console Amazon Timestream pour InfluxDB, choisissez la région dans laquelle vous souhaitez créer AWS l'instance de base de données.
3. Dans le volet de navigation, choisissez InfluxDB Databases.
4. Choisissez Create Influx database.
5. Pour DB Instance Identifier, entrez un nom qui identifiera votre instance.
6. Fournissez les paramètres de configuration de base d'InfluxDB : nom d'utilisateur, organisation, nom du compartiment et mot de passe.

   > **⚠ Important**
   >
   > Votre nom d'utilisateur, votre organisation, le nom de votre bucket et votre mot de passe seront enregistrés sous forme de secret dans AWS Secrets Manager qui sera créé pour votre compte.

Si vous devez modifier le mot de passe utilisateur une fois que l'instance de base de données est disponible, vous pouvez le modifier à l'aide de l'Influx CLI.

7.

8. Pour la classe d'instance de base de données, sélectionnez une taille d'instance mieux adaptée à vos besoins en matière de charge de travail.
9. Pour la classe de stockage de base de données, sélectionnez une classe de stockage adaptée à vos besoins. Dans tous les cas, il vous suffira de configurer le stockage alloué.
10. Dans la section Configuration de la connectivité, assurez-vous que votre instance InfluxDB se trouve dans le même sous-réseau que vos nouveaux clients qui ont besoin d'une connectivité à votre instance de base de données Timestream for InfluxDB. Vous pouvez également choisir de rendre votre instance de base de données accessible au public.
11. Choisissez Create Influx database.
12. Dans la liste des bases de données, choisissez le nom de votre nouvelle instance InfluxDB pour afficher ses détails. L'instance de base de données a le statut Creating jusqu'à ce qu'elle soit prête à être utilisée.
13. Lorsque l’état passe à Available (Disponible), vous pouvez vous connecter à l’instance de base de données. En fonction de la quantité de stockage et de la classe d’instance de base de données, la mise à disposition de la nouvelle instance peut prendre jusqu’à 20 minutes.

À l'aide du CLI

Pour créer une instance de base de données à l'aide de AWS Command Line Interface, appelez la `create-db-instance` commande avec les paramètres suivants :

```
--name
--vpc-subnet-ids
--vpc-security-group-ids
--db-instance-type
--db-storage-type
--username
--organization
--password
--allocated-storage
```

Pour obtenir des informations sur chaque paramètre, consultez Paramètres des instances de base de données.

Example Exemple : utilisation des configurations de moteur par défaut

Pour 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
```

Pour 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
```

À l'aide du API

Pour créer une instance de base de données à l'aide de AWS Command Line Interface, appelez la `CreateDBInstance` commande avec les paramètres suivants :

Pour obtenir des informations sur chaque paramètre, consultez Paramètres des instances de base de données.

> **Important**
>
> Partie de l'objet de DBInstance réponse pour lequel vous recevez un influxAuthParametersSecretArn. Cela gardera un ARN SecretsManager secret sur votre compte. Il ne sera renseigné qu'une fois que vos instances de base de données InfluxDB

seront disponibles. Le secret contient les paramètres d'authentification des flux fournis au cours du `CreateDbInstance` processus. Il s'agit d'une READONLY copie, car aucun updates/modifications/deletions élément de ce secret n'a d'impact sur l'instance de base de données créée. Si vous supprimez ce secret, notre API réponse fera toujours référence au secret supprimé ARN.

Une fois que vous avez terminé de créer votre instance de base de données Timestream pour InfluxDB, nous vous recommandons de télécharger, d'installer et de configurer l'Influx. CLI

L'influx CLI fournit un moyen simple d'interagir avec InfluxDB à partir d'une ligne de commande. Pour des instructions détaillées d'installation et de configuration, voir Utiliser l'afflux CLI.

Paramètres des instances de base de données

Vous pouvez créer une instance de base de données à l'aide de la console, de la `create-db-instance` CLI commande ou du `CreateDBInstance` Timestream pour l'opération API InfluxDB.

Le tableau suivant fournit des détails sur les paramètres que vous choisissez lorsque vous créez une instance de base de données.

Paramètre de la console	Description	CLIoption et paramètre Timestream API
Stockage alloué	Quantité de stockage à allouer pour votre instance de base de données (en gibioctets). Dans certains cas, allouer une quantité de stockage pour votre instance de base de données supérieure à la taille de votre base de données permet d'améliorer les performan ces d'I/O. Pour de plus amples informations, veuillez consulter Stockage d'instance InfluxDB.	CLI: `allocated -storage` API: `allocated storage`
Nom du compartim ent	Nom du compartiment pour initialiser l'instance InfluxDb	CLI: `bucket` API: `bucket`

Paramètre de la console	Description	CLIoption et paramètre Timestream API
Type d'instance de base de données	Configuration pour votre instance de base de données. Par exemple, une classe d'instance de base de données db.influx.large possède 16 GiB de mémoire, dont 2 sont optimisées pour la mémoire. vCPUs Si possible, choisissez un type d'instance de base de données suffisamment grand pour qu'un ensemble de travail de requête typique puisse être conservé en mémoire. Lorsque les ensembles de travail sont en mémoire, le système peut éviter d'écrire sur le disque, ce qui améliore les performances. Pour de plus amples informations, veuillez consulter Types de classes d'instance de base de données.	CLI: `db-instance-type` API: `Dbinstancetype`
Identifiant d'instance de base de données	Nom de votre instance de base de données. Nommez vos instances de base de données de la même façon que vos serveurs sur site. L'identifiant de votre instance de base de données peut contenir jusqu'à 63 caractères alphanumériques et doit être unique pour votre compte dans la AWS région que vous avez choisie.	CLI: `db-instance-identifier` API: `Dbinstanceidentifier`
Groupe de paramètres de base de données	Groupe de paramètres pour l'instance de base de données. Vous pouvez soit choisir le groupe de paramètres par défaut, soit créer un groupe de paramètres personnalisé. Pour plus d'informations, consultez Utilisation des groupes de paramètres DB..	CLI: `db-parameter-group-name` API: `DBParameterGroupName`

Paramètre de la console	Description	CLIoption et paramètre Timestream API
Paramètre de livraison du journal	Le nom du compartiment S3 dans lequel les journaux InfluxDB seront stockés.	CLI: `LogDeliveryConfiguration` API: `log-delivery-configuration`
déploiement multi-AZ	Create a standby instance (Créer une instance de secours) permet de créer un réplica secondaire passif de votre instance de base de données dans une autre zone de disponibilité pour la prise en charge du basculement. Nous recommandons Multi-AZ pour les charges de travail de production afin de maintenir une haute disponibilité. Pour le développement et les tests, vous pouvez choisir Do not create a standby instance (Ne pas créer d'instance de secours). Pour de plus amples informations, veuillez consulter Configuration et gestion d'un déploiement multi-AZ.	CLI: `MultiAz` API: `multi-az`
Mot de passe	Ce sera votre mot de passe d'utilisation principal pour initialiser votre instance InfluxDB Db. Vous utiliserez ce mot de passe pour vous connecter à InfluxUI afin d'obtenir votre jeton d'opérateur.	CLI: `password` API: `password`

Paramètre de la console	Description	CLIoption et paramètre Timestream API
Accès public	Oui, pour attribuer à l'instance de base de données une adresse IP publique, ce qui signifie qu'elle est accessible en dehors deVPC. Pour être accessible au public, l'instance de base de données doit également se trouver dans un sous-réseau public duVPC. Non pour rendre l'instance de base de données accessible uniquement depuis l'intérieur duVPC. Pour se connecter à une instance de base de données depuis l'extérieur de celle-ciVPC, l'instance de base de données doit être accessible au public. En outre, l'accès doit être accordé en utilisant les règles entrantes du groupe de sécurité de l'instance de base de données. En outre, d'autres exigences doivent être respectées.	CLI: `publicly-accessible` API: `PubliclyAccessible`
Storage Type	Le type de stockage de votre instance de base de données Vous pouvez choisir entre 3 types différents : flux provisionné, stockage IOPS inclus, en fonction de vos besoins en matière de charge de travail : * Influx IOPS inclus 3000 IOPS * Afflux IOPS inclus : 12 000 IOPS * INflux IOPS Inclus 16000 IOPS Pour de plus amples informations, veuillez consulter Stockage d'instance InfluxDB.	CLI: `db-storage-type` API: `DbStorageType`

Paramètre de la console	Description	CLIoption et paramètre Timestream API
Nom d'utilisateur initial	Ce sera l'utilisateur principal avec lequel initialiser votre instance de base de données InfluxDB. Vous utiliserez ce nom d'utilisateur pour vous connecter à InfluxUI afin d'obtenir votre jeton d'opérateur.	CLI: `username` API: `Username`
Sous-réseaux	Un sous-réseau vpc à associer à cette instance de base de données.	CLI: `vpc-subnet-ids` API: `VPCSubnetIds`
VPCGroupe de sécurité (pare-feu)	Groupe de sécurité à associer à l'instance de base de données.	CLI: `vpc-security-group-ids` API: `VPCSecurityGroupIds`

Connexion à une instance de base de données Amazon Timestream pour InfluxDB

Avant de pouvoir vous connecter à une instance de base de données, vous devez créer l'instance de base de données. Pour plus d'informations, veuillez consulter Création d'une instance de base de données. Une fois qu'Amazon Timestream a approvisionné votre instance de base de données, utilisez API InfluxDB, CLI Influx ou tout autre client ou utilitaire compatible permettant à InfluxDB de vous connecter à l'instance de base de données.

Rubriques

- Recherche des informations de connexion pour une instance de base de données Amazon Timestream pour InfluxDB
- Options d'authentification de base de données
- Utilisation des groupes de paramètres

Recherche des informations de connexion pour une instance de base de données Amazon Timestream pour InfluxDB

Les informations de connexion d'une instance de base de données incluent son point de terminaison, son port, son nom d'utilisateur, son mot de passe et un jeton d'accès valide, tel que l'opérateur ou le jeton d'accès complet. Par exemple, pour une instance de base de données InfluxDB, supposons que la valeur du point de terminaison soit. `influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com` Dans ce cas, la valeur du port est 8086 et l'utilisateur de la base de données est admin. Compte tenu de ces informations, vous spécifiez les valeurs suivantes dans une chaîne de connexion :

- Pour l'hôte ou le nom ou le DNS nom d'hôte, spécifie`influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com`.
- Pour le port, spécifiez 8086.
- Pour utilisateur, spécifiez admin.
- Pour le mot de passe, spécifiez celui que vous avez fourni lors de la création de votre instance de base de données.

> **Important**
>
> Lorsque vous avez créé votre instance Timestream pour InfluxDB Db, vous recevez une partie de l'objet de DBInstance réponse. `influxAuthParametersSecretArn` Cela contiendra un code SecretsManager secret sur votre compte. Il ne sera renseigné qu'une fois que vos instances de base de données InfluxDB seront disponibles. Le secret contient les paramètres d'authentification des flux fournis au cours du `CreateDbInstance` processus. Il s'agit d'une READONLY copie, car aucun updates/modifications/deletions élément de ce secret n'a d'impact sur l'instance de base de données créée. Si vous supprimez ce secret, notre API réponse fera toujours référence à l'ARN du secret supprimé.

Le point de terminaison est unique pour chaque instance de base de données, et les valeurs du port et de l'utilisateur peuvent varier. Pour vous connecter à une instance de base de données, vous pouvez utiliser InfluxCLI, Influx API ou tout autre client compatible avec InfluxDB.

Pour trouver les informations de connexion d'une instance de base de données, utilisez la console AWS de gestion. Vous pouvez également utiliser la AWS commande Command Line Interface (AWS CLI) `describe-db-instances` ou l'opération API `GetDBInstance` Timestream-InfluxDB.

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez InfluxDB Databases pour afficher la liste de vos instances de base de données.
3. Choisissez le nom de l'instance de base de données pour afficher ses détails.
4. Dans la section Résumé, copiez le point de terminaison. Notez également le numéro du port. Vous avez besoin du point de terminaison et du numéro de port pour vous connecter à l'instance de base de données.

Si vous avez besoin des informations relatives au nom d'utilisateur et au mot de passe, cliquez sur l'onglet Détails de configuration, puis sur le `influxAuthParametersSecretArn` pour accéder à votre Secrets Manager.

En utilisant le CLI

- Pour trouver les informations de connexion pour une instance de base de données InfluxDB à l'aide de AWS CLI, appelez la get-db-instance commande. Dans l'appel, demandez l'ID, le point de terminaison, le port et le influxAuthParameters secret de l'instance de base de données.

  Pour Linux, macOS ou Unix :

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

  Pour Windows :

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

  Votre sortie doit ressembler à ce qui suit. Pour accéder aux informations du nom d'utilisateur, vous devez vérifier le`InfluxAuthParameterSecret`.

  ```
  [
      [
          "mydb",
          "mydb-123456789012.us-east-1.timestream-influxdb.amazonaws.com",
          8086,
  ```

```
    ]
]
```

Création de jetons d'accès

Avec ces informations, vous allez pouvoir vous connecter à votre instance pour récupérer ou créer vos jetons d'accès. Il existe plusieurs moyens d'y parvenir :

En utilisant le CLI

1. Si ce n'est pas déjà fait, téléchargez, installez et configurez l'influx CLI.
2. Lors de la configuration de votre CLI configuration Influx, utilisez `--username-password` pour vous authentifier.

   ```
   influx config create --config-name YOUR_CONFIG_NAME  --host-url "https://
   yourinstance.timestream-influxdb.amazonaws.com:8086"  --org yourorg --username-
   password admin --active
   ```

3. Utilisez la commande influx auth create pour recréer votre jeton d'opérateur. Tenez compte du fait que ce processus invalidera l'ancien jeton d'opérateur.

   ```
   influx auth create --org kronos --operator
   ```

4. Une fois que vous avez le jeton d'opérateur, vous pouvez utiliser la commande influx auth list pour afficher les jetons de tous vos jetons. Vous pouvez utiliser la commande influx auth create pour créer un jeton d'accès universel.

> **Important**
>
> Vous devrez d'abord effectuer cette étape pour obtenir votre jeton d'opérateur, pour pouvoir ensuite créer de nouveaux jetons à l'aide de l'InfluxDB ouAPI. CLI

Utilisation de l'interface utilisateur Influx

1. Accédez à votre instance Timestream for InfluxDB à l'aide du point de terminaison créé pour vous connecter et accéder à l'interface utilisateur d'InfluxDB. Vous devrez utiliser le nom d'utilisateur et le mot de passe utilisés pour créer votre instance de base de données InfluxDB.

Vous pouvez récupérer ces informations à partir de `influxAuthParametersSecretArn` celles spécifiées dans l'objet de réponse du`CreateDbInstance`.

Vous pouvez également ouvrir l'InfluxUI depuis la console de gestion Timestream for InfluxDB :

a. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/timestream/

b. Dans le coin supérieur droit de la console Amazon Timestream pour InfluxDB, choisissez la région dans laquelle vous avez créé l'instance de base AWS de données.

c. Dans la liste des bases de données, choisissez le nom de votre instance InfluxDB pour afficher ses détails. Dans le coin supérieur droit, choisissez Open Influx UI.

2. Une fois connecté à votre InfluxUI, accédez à Load Data puis APITokens en utilisant la barre de navigation de gauche.
3. Choisissez +, GENERATE API TOKEN puis sélectionnez All Access API Token.
4. Entrez une description pour le API jeton et choisissez SAVE.
5. Copiez le jeton généré et stockez-le pour le conserver en lieu sûr.

> **Important**
>
> Lors de la création de jetons à partir de l'InfluxUI, les jetons nouvellement créés ne seront affichés qu'une seule fois. Assurez-vous de les copier, sinon vous devrez les recréer.

Utilisation de l'InfluxDB API

- Envoyez une demande au point de API `/api/v2/authorizations` terminaison InfluxDB à l'aide de la méthode POST request.

  Joignez les éléments suivants à votre demande :

  a. En-têtes :

     i. Autorisation : jeton < INFLUX _ OPERATOR _ TOKEN >

     ii. Type de contenu : application/json

  b. Corps de la requête : JSON corps présentant les propriétés suivantes :

     i. statut : « actif »

ii. description : description API du jeton

iii. OrgID : ID d'organisation InfluxDB

iv. permissions : tableau d'objets où chaque objet représente des autorisations pour un type de ressource InfluxDB ou une ressource spécifique. Chaque autorisation contient les propriétés suivantes :

A. action : « lire » ou « écrire »

B. ressource : JSON objet qui représente la ressource InfluxDB à laquelle accorder l'autorisation. Chaque ressource contient au moins la propriété suivante : OrgID : ID d'organisation InfluxDB

C. type : type de ressource. Pour plus d'informations sur les types de ressources InfluxDB existants, utilisez the /api/v2/resources endpoint.

L'exemple suivant utilise `curl` l'InfluxDB API pour générer un jeton d'accès universel :

```
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"}}
    ]
  }
'
```

Options d'authentification de base de données

Amazon Timestream pour InfluxDB prend en charge les méthodes suivantes pour authentifier les utilisateurs de base de données :

- Authentification par mot de passe, – Votre instance de base de données effectue toute l'administration des comptes d'utilisateurs. Vous créez des utilisateurs, spécifiez des mots de passe et administrez des jetons à l'aide d'InfluxUI, Influx CLI ou influx. API

- Authentification par jeton : votre instance de base de données effectue toute l'administration des comptes utilisateurs. Vous pouvez créer des utilisateurs, spécifier un mot de passe et administrer des jetons à l'aide de votre jeton d'opérateur à l'aide de l'Influx CLI et de l'InfluxAPI.

Connexions chiffrées

Vous pouvez utiliser Secure Socket Layer (SSL) ou Transport Layer Security (TLS) depuis votre application pour chiffrer une connexion à une instance de base de données. Les certificats nécessaires à la prise de TLS contact entre InfluxDB et les applications créées et gérées par le service Kronos. Lorsque le certificat est renouvelé, l'instance est automatiquement mise à jour avec la dernière version sans intervention de l'utilisateur.

Utilisation des groupes de paramètres

Les paramètres de base de données spécifient comment la base de données est configurée. Par exemple, les paramètres de base de données peuvent spécifier la quantité de ressources, telles que la mémoire, à allouer à une base de données.

Vous gérez la configuration de votre base de données en associant vos instances de base de données à des groupes de paramètres. Amazon Timestream pour InfluxDB définit des groupes de paramètres avec des paramètres par défaut. Vous pouvez également définir vos propres groupes de paramètres à l'aide de paramètres personnalisés.

Présentation des groupes de paramètres

Un groupe de paramètres de base de données sert de conteneur pour les valeurs de configuration du moteur qui sont appliquées à une ou plusieurs instances de base de données.

Rubriques

- Groupes de paramètres par défaut et personnalisés
- Création d'un groupe de paramètres de bases de données
- Paramètres d'instance de bases de données statiques et dynamiques
- Paramètres et valeurs de paramètres pris en charge

Groupes de paramètres par défaut et personnalisés

Les instances de base de données utilisent des groupes de paramètres de base de données. Les sections suivantes décrivent la configuration et la gestion des groupes de paramètres d'une instance de base de données.

Création d'un groupe de paramètres de bases de données

Vous pouvez créer un nouveau groupe de paramètres de base de données à l'aide du AWS Management Console AWS Command Line Interface, du ou du TimestreamAPI.

Les limites suivantes s'appliquent aux noms de groupes de paramètres de base de données :

- Ces noms doivent comporter entre 1 et 255 lettres, chiffres ou traits d'union.
- Les noms des groupes de paramètres par défaut peuvent inclure un point, par exemple `default.InfluxDB.2.7`. Toutefois, les noms de groupes de paramètres personnalisés ne peuvent pas inclure de point.
- Le premier caractère doit être une lettre.
- Le nom ne peut pas commencer par « dbpg- »
- Les noms ne peuvent pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.
- Si vous créez une instance de base de données sans spécifier de groupe de paramètres de base de données, l'instance de base de données utilise les valeurs par défaut du moteur InfluxDB.

Vous ne pouvez pas modifier les valeurs de paramètre d'un groupe de paramètres de base de données par défaut. Au lieu de cela, vous pouvez effectuer les actions suivantes :

1. Créez un groupe de paramètres.
2. Modifiez les paramètres souhaités. Il n'est pas possible de modifier tous les paramètres du moteur de base de données dans un groupe de paramètres.
3. Mettez à jour votre instance de base de données pour utiliser le groupe de paramètres personnalisé. Pour plus d'informations sur la mise à jour d'une instance de base de données, consultezMise à jour des instances de base de données.

> **Note**
>
> Si vous avez modifié votre instance de base de données pour utiliser un groupe de paramètres personnalisé et que vous démarrez l'instance de base de données, Amazon Timestream for InfluxDB redémarre automatiquement l'instance de base de données dans le cadre du processus de démarrage.
> Actuellement, vous ne pouvez pas modifier les groupes de paramètres personnalisés une fois qu'ils ont été créés. Si vous devez modifier un paramètre, vous devez créer un nouveau groupe de paramètres personnalisé et l'attribuer aux instances qui nécessitent cette modification de configuration. Si vous mettez à jour une instance de base de données existante pour attribuer un nouveau groupe de paramètres, celui-ci sera toujours appliqué immédiatement et redémarrera votre instance.

Paramètres d'instance de bases de données statiques et dynamiques

Les paramètres de l'instance de base de données InfluxDB sont toujours statiques. Ils se comportent comme suit :

Lorsque vous modifiez un paramètre statique, enregistrez le groupe de paramètres de base de données et que vous l'attribuez à une instance, la modification du paramètre prend effet automatiquement après le redémarrage de l'instance.

Lorsque vous associez un nouveau groupe de paramètres de base de données à une instance de base de données, Timestream applique les paramètres statiques modifiés uniquement après le redémarrage de l'instance de base de données. Actuellement, la seule option est d'appliquer immédiatement.

Pour de plus amples informations sur la modification du groupe de paramètres de base de données, veuillez consulter Mise à jour des instances de base de données.

Paramètres et valeurs de paramètres pris en charge

Pour déterminer les paramètres pris en charge pour votre instance de base de données, consultez les paramètres du groupe de paramètres de base de données utilisé par l'instance de base de données. Pour plus d'informations, consultez la section Affichage des valeurs des paramètres d'un groupe de paramètres de base de données.

Pour plus d'informations sur tous les paramètres pris en charge par la version open source d'InfluxDB, consultez les options de configuration d'InfluxDB. Actuellement, vous ne pouvez modifier que les paramètres InfluxDB suivants :

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
flux-log-enabled	Inclure une option pour afficher les journaux détaillés des requêtes Flux	FALSE	true, false	N/A	
niveau du journal	Enregistrez le niveau de sortie. InfluxDB produit des entrées de journal avec des niveaux de gravité supérieurs ou égaux au niveau spécifié.	info	debug, info, erreur	N/A	
aucune tâche	Nombre de requêtes autorisées à être exécutées simultanément. La valeur 0 autorise	FALSE	true, false	N/A	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
	un nombre illimité de requêtes simultanées.				
simultanéité des requêtes	Désactivez le planificateur de tâches. Si des tâches problémat iques empêchent le démarrage d'InfluxDB, utilisez cette option pour démarrer InfluxDB sans planifier ni exécuter de tâches.	1 024		N/A	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
query-queue-size	Nombre maximum de requêtes autorisées dans la file d'exécution. Lorsque la limite de file d'attente est atteinte, les nouvelles requêtes sont rejetées. La valeur 0 autorise un nombre illimité de requêtes dans la file d'attente.	1 024		N/A	
type de traçage	Activez le suivi dans InfluxDB et spécifiez le type de suivi. Le suivi est désactivé par défaut.	""	bûche, jaeger	N/A	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
métriques désactivées	Désactivez le point de terminaison HTTP / metrics qui expose les métriques internes d'InfluxDB.	FALSE		N/A	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
http-idle-timeout	Durée maximale pendant laquelle le serveur doit maintenir les connexions établies en attente de nouvelles demandes. Réglé sur `0` pour qu'il n'y ait pas de délai d'expiration.	30 minutes	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
http-read-header-timeout	Durée maximale pendant laquelle le serveur doit essayer de lire HTTP les en-têtes des nouvelles demandes. Réglé sur `0` pour ne pas avoir de temps mort.	10 s	Durée avec unité `hoursminutes nds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203 685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
http-read-timeout	Durée maximale pendant laquelle le serveur doit essayer de lire l'intégralité des nouvelles demandes. Réglé sur 0 pour ne pas avoir de temps mort.	0	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
http-write-timeout	Durée maximale que le serveur doit consacrer au traitement et à la réponse aux demandes d'écriture. Réglé sur `0` pour ne pas avoir de temps mort.	0	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203 685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
influxql-max-select-buckets	Nombre maximum de groupes par tranches temporelles qu'une `SELECT` instruction peut créer. `0`permet un nombre illimité de seaux.	0	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
influxql-max-select-point	Nombre maximum de points qu'une `SELECT` déclaration peut traiter. `0`permet un nombre illimité de points. InfluxDB vérifie le nombre de points toutes les secondes (les requêtes dépassant le maximum ne sont donc pas immédiatement abandonnées).	0	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
influxql-max-select-series	Nombre maximum de séries qu'un `SELECT` relevé peut renvoyer. `0`permet un nombre illimité de séries.	0	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	
prof-disabled	Désactivez le `/debug/pprof` HTTP point de terminaison. Ce point de terminaison fournit des données de profilage d'exécution et peut être utile lors du débogage.	FALSE	Booléen	N/A	
query-initial-memory-bytes	Octets de mémoire initiaux alloués pour une requête.	0	Long	Minimum : 0 Maximum : query-memory-bytes	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
query-max-memory-bytes	Nombre total maximal d'octets de mémoire autorisés pour les requêtes.	0	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	
query-memory-bytes	Spécifie la durée de vie (TTL) en minutes pour les sessions utilisateur nouvellement créées.	0	Long	Minimum : 0 Maximum : 2 147 483 647	Doit être supérieur ou égal à query-initial-memory-bytes.
durée de la session	Spécifie la durée de vie (TTL) en minutes pour les sessions utilisateur nouvellement créées.	60	Entier	Minimum : 0 Maximum : 2880	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
session-renew-disabled	Désactive l'extension automatique de la session d'un utilisateur TTL à chaque demande. Par défaut, chaque demande définit le délai d'expiration de la session à cinq minutes à partir de maintenant. Lorsqu'elle est désactivée, les sessions expirent après la durée de session spécifiée et l'utilisateur est redirigé vers la page de connexion, même s'il est	FALSE	Booléen	N/A	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
	récemment actif.				
storage-cache-max-memory-taille	Taille maximale (en octets) que le cache d'une partition peut atteindre avant qu'il ne commence à rejeter des écritures.	1073741824	Long	Minimum : 0 Maximum : 549755813888	Doit être inférieure à la capacité de mémoire totale de l'instance. Nous vous recommandons de le régler à une valeur inférieure à 15 % de la capacité de mémoire totale.
storage-cache-snapshot-memory-taille	Taille (en octets) à laquelle le moteur de stockage créera un instantané du cache et l'écrira dans un TSM fichier afin de libérer de la mémoire.	26214400	Long	Minimum : 0 Maximum : 549755813888	Doit être inférieur à storage-cache-max-memory - size.

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-cache-snapshot-write-durée du froid	Durée pendant laquelle le moteur de stockage capture le cache et l'écrit dans un nouveau TSM fichier si la partition n'a pas reçu d'écritures ou de suppressions.	100ms	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203 685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-compact-full-write-durée du froid	Durée pendant laquelle le moteur de stockage compactera tous les TSM fichiers d'une partition s'il n'a pas reçu d'écritures ou de suppressions.	40h00s	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-compact-throughput-burst	Limite de débit (en octets par seconde) que les TSM compactages peuvent écrire sur le disque.	50331648	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-max-concurrent-compactions	Nombre maximum de compactages complets et plats pouvant être exécutés simultanément. Une valeur de correspond à 0 à 50 % d'`runtime.GOMAXPROCS(0)` utilisation au moment de l'exécution. Tout nombre supérieur à zéro limite les compactages à cette valeur. Ce paramètre ne s'applique pas à la capture instantanée du cache.	0	Entier	Minimum : 0 Maximum : 64	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-max-index-log-taille du fichier	Taille (en octets) à laquelle un fichier journal d'écriture anticipée (WAL) d'index sera compacté en fichier d'index. Des tailles inférieures accéléreront le compactage des fichiers journaux et réduiront l'utilisation du tas au détriment du débit d'écriture.	1048576	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	
storage-no-validate-field-taille	Ignorez la validation de la taille des champs sur les demandes d'écriture entrantes.	FALSE	Booléen	N/A	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-retention-check-interval	Intervalle entre les contrôles de mise en œuvre de la politique de rétention.	300 ms	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	N/A	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203685

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-series-file-max-concurrent-snapshot-compactions	Nombre maximal de compactages de snapshots pouvant être exécutés simultanément sur toutes les partitions de série d'une base de données.	0	Entier	Minimum : 0 Maximum : 64	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-series-id-set-taille du cache	Taille du cache interne utilisé dans l'TSIindex pour stocker les résultats de séries précédemment calculés. Les résultats mis en cache sont renvoyés rapidement au lieu de devoir être recalculés lors de l'exécution d'une requête ultérieure avec le même prédicat clé/valeur de balise. La définition de cette valeur sur 0 désactivera le cache et risque de réduire les	100	Long	Minimum : 0 Maximum : 9 223 372 036 854 775 807	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
	performances des requêtes.				
storage-wal-max-concurrent-écrit	Nombre maximum d'écritures à effectuer simultanément dans le WAL répertoire.	0	Entier	Minimum : 0 Maximum : 256	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
storage-wal-max-write-retard	Durée maximale pendant laquelle une demande d'écriture WAL dans le répertoire peut attendre lorsque le nombre maximal d'écritures actives simultanées WAL dans le répertoire est atteint. Réglez sur `0` pour désactiver le délai d'expiration.	10 m	Durée avec unité `hoursminutesnds` . Exemple : `durationType=minutes,value=10`	Heures : -Minimum : 0 -Maximum : 256 205 Procès-verbal : -Minimum : 0 -Maximal : 15372 286 Secondes : -Minimum : 0 -Maximum : 922337203 Millisecondes : -Minimum : 0 -Maximum : 922337203 685	

Paramètre	Description	Valeur par défaut	Valeur	Plage valide	Remarque
interface utilisateur désactivée	Désactivez l'interface utilisateur (UI) InfluxDB. L'interface utilisateur est activée par défaut.	FALSE	Booléen	N/A	

La configuration incorrecte de paramètres dans un groupe de paramètres peut avoir des effets contraires involontaires, dont une dégradation de la performance et une instabilité du système. Soyez toujours prudent lorsque vous modifiez les paramètres de base de données. Essayez de modifier les paramètres de groupe de paramètres sur une instance de base de données de test avant d'appliquer ces modifications de groupe de paramètres à une instance de base de données de production.

Utilisation des groupes de paramètres DB

Les instances de base de données utilisent des groupes de paramètres de base de données. Les sections suivantes décrivent la configuration et la gestion des groupes de paramètres d'une instance de base de données.

Rubriques

- Création d'un groupe de paramètres de bases de données
- Association d'un groupe de paramètres de base de données à une instance de base de données
- Liste des groupes de paramètres de bases de données
- Affichage des valeurs de paramètres pour un groupe de paramètres de bases de données

Création d'un groupe de paramètres de bases de données

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.

2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Choisissez Créer un groupe de paramètres.
4. Dans la zone Nom du groupe, entrez le nom du nouveau groupe de paramètres de base de données.
5. Dans la zone Description, entrez une description pour le nouveau groupe de paramètres de base de données.
6. Choisissez les paramètres à modifier et appliquez les valeurs souhaitées. Pour plus d'informations sur les paramètres pris en charge, consultezParamètres et valeurs de paramètres pris en charge.
7. Choisissez Save (Enregistrer).

En utilisant le AWS Command Line Interface

- Pour créer un groupe de paramètres de base de données à l'aide de AWS CLI, appelez la `create-db-parameter-group` commande avec les paramètres suivants :

```
--db-parameter-group-name <value>
--description <value>
--endpoint_url <value>
--region <value>
--parameters (list) (string)
```

Example Exemple

Pour obtenir des informations sur chaque paramètre, consultez Paramètres des instances de base de données. Cet exemple utilise les configurations de moteur par défaut.

```
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
```

Association d'un groupe de paramètres de base de données à une instance de base de données

Vous pouvez créer vos propres groupes de paramètres de base de données avec des paramètres personnalisés. Vous pouvez associer un groupe de paramètres de base de données à une instance de base de données en utilisant le AWS Management Console, le ou le AWS Command Line Interface API Timestream-InfluxDB. Vous pouvez le faire lorsque vous créez ou modifiez une instance de base de données.

Pour plus d'informations sur la création d'un groupe de paramètres de base de données, consultez Création d'un groupe de paramètres de bases de données. Pour de plus amples informations sur la création d'une instance de base de données, veuillez consulter Création d'une instance de base de données. Pour plus d'informations sur la modification d'une instance de base de données, consultezMise à jour des instances de base de données..

> **Note**
>
> Lorsque vous associez un nouveau groupe de paramètres de base de données à une instance de base de données, les paramètres statiques modifiés ne sont appliqués qu'après le redémarrage de l'instance de base de données. Actuellement, seule l'application immédiate est prise en charge. Timestream pour InfluxDB ne prend en charge que les paramètres statiques.

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez InfluxDB Databases, puis choisissez l'instance de base de données que vous souhaitez modifier.
3. Choisissez Mettre à jour. La page Mettre à jour l'instance de base de données s'affiche.
4. Modifiez le paramètre DB parameter group (groupe de paramètres de base de données).
5. Choisissez Continuer et vérifiez le récapitulatif des modifications.
6. Actuellement, seule la fonctionnalité Appliquer immédiatement est prise en charge. Cette option peut provoquer une panne dans certains cas car elle redémarrera votre instance de base de données.

7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Update DB instance pour enregistrer vos modifications et les appliquer. Vous pouvez également sélectionner Retour pour revoir vos modifications ou Annuler pour les annuler.

À l'aide du AWS Command Line Interface

Pour 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 }}"
```

Pour 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 }}"
```

Liste des groupes de paramètres de bases de données

Vous pouvez répertorier les groupes de paramètres de base de données que vous avez créés pour votre AWS compte.

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez Groupes de paramètres.
3. Les groupes de paramètres DB s'affichent dans une liste.

À l'aide du AWS Command Line Interface

Pour répertorier tous les groupes de paramètres de base de données d'un AWS compte, utilisez la AWS Command Line Interface `list-db-parameter-groups` commande.

```
aws timestream-influxdb list-db-parameter-groups --region region
```

Pour renvoyer des groupes de paramètres de base de données spécifiques pour un AWS compte, utilisez la AWS Command Line Interface `get-db-parameter-group` commande.

```
aws timestream-influxdb get-db-parameter-group --region region --identifier identifier
```

Affichage des valeurs de paramètres pour un groupe de paramètres de bases de données

Vous pouvez obtenir une liste de tous les paramètres dans un groupe de paramètres DB et de leurs valeurs.

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez Groupes de paramètres.
3. Les groupes de paramètres DB s'affichent dans une liste.
4. Choisissez le nom du groupe de paramètres pour consulter la liste des paramètres associée.

À l'aide du AWS Command Line Interface

Pour afficher les valeurs des paramètres d'un groupe de paramètres de base de données, utilisez la AWS Command Line Interface `get-db-parameters` commande avec le paramètre obligatoire suivant.

```
--db-parameter-group-name
```

À l'aide du API

Pour afficher les valeurs des paramètres d'un groupe de paramètres de base de données, utilisez la API `GetDBParameters` commande Timestream avec le paramètre obligatoire suivant.

```
DBParameterGroupName
```

Gestion des instances de base de données

Cette section couvre divers aspects de la gestion de Timestream pour l'instance InfluxDB afin de garantir des performances, une disponibilité et des capacités de surveillance optimales. Il fournit des conseils sur la mise à jour des configurations de vos instances de base de données, la gestion des déploiements multi-AZ et les processus de basculement. Il explique également comment supprimer des instances de base de données et configurer l'affichage des journaux pour vos instances InfluxDB.

Rubriques

Mise à jour des instances de base de données

Vous pouvez mettre à jour les paramètres de configuration suivants de votre instance Timestream for InfluxDB :

- Classe d'instance
- Type de déploiement
- Groupe de paramètres
- Configuration de la livraison des journaux

> **Important**
>
> Nous vous recommandons de tester toutes les modifications sur une instance de test avant de modifier l'instance de production afin de comprendre leur impact, en particulier lors de la mise à niveau des versions de base de données. Vérifiez l'impact sur votre base de données et vos applications avant de mettre à jour les paramètres. Certaines modifications nécessitent le redémarrage de l'instance de base de données, ce qui entraîne une interruption de service.

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez InfluxDB Databases, puis choisissez l'instance de base de données que vous souhaitez modifier.
3. Sélectionnez Modifier.
4. Sur la page Modifier une instance de base de données, apportez les modifications souhaitées.
5. Choisissez Continuer et vérifiez le récapitulatif des modifications.
6. Choisissez Suivant.
7. Passez en revue vos modifications.
8. Choisissez Modifier l'instance pour appliquer vos modifications.

> **Note**
>
> Ces modifications nécessitent un redémarrage de l'instance de base de données Influx et peuvent provoquer une panne dans certains cas.

À l'aide du AWS Command Line Interface

Pour mettre à jour une instance de base de données à l'aide de AWS Command Line Interface, appelez la `update-db-instance` commande. Spécifiez l'identifiant d'instance de base de données et les valeurs des options que vous souhaitez modifier. Pour plus d'informations sur chaque option, veuillez consulter Paramètres des instances de base de données.

Example Exemple

Le code suivant modifie mydbinstance en définissant un dbparametergroup différent. Les modifications sont appliquées immédiatement.

Pour 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
```

Pour 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
```

Entretien d'une instance de base de données

Amazon Timestream pour InfluxDB effectue régulièrement la maintenance des ressources d'InfluxDB sur Amazon Timestream. La maintenance implique le plus souvent des mises à jour des ressources suivantes de votre instance de base de données :

- Matériel sous-jacent
- Système d'exploitation (SE) sous-jacent
- Version du moteur de base de données

Les mises à jour du système d'exploitation se produisent le plus souvent pour des raisons de sécurité.

Certains éléments de maintenance nécessitent qu'Amazon Timestream pour InfluxDB mette votre instance de base de données hors ligne pendant une courte période. Parmi les éléments de maintenance qui nécessitent qu'une ressource soit hors ligne figure l'application obligatoire de correctifs au système d'exploitation ou à la base de données. Les mises à jour correctives obligatoires sont planifiées automatiquement uniquement pour les correctifs associés à la sécurité et à la fiabilité de l'instance. Ce type d'application de correctifs est peu fréquent, généralement une fois tous les quelques mois. Cela nécessite rarement plus d'une fraction de votre fenêtre de maintenance.

- Les fenêtres de maintenance sont configurées pour avoir lieu tous les jours entre 12 h 00 et 4 h 00, heure locale, dans la région dans laquelle votre instance est hébergée.
- Les ressources des clients peuvent être corrigées une fois par semaine au cours de l'une des sept fenêtres de maintenance de la semaine.

Suppression d'une instance DB

La suppression d'une instance de base de données a un effet sur la capacité de restauration de l'instance et sur la disponibilité des snapshots. Examinez les problèmes suivants :

- Si vous souhaitez supprimer toutes les ressources Timestream pour InfluxDB, notez que les ressources des instances de base de données entraînent des frais de facturation.
- Lorsque le statut d'une instance de base de données est supprimé, la valeur de son certificat CA n'apparaît pas dans la console Timestream for InfluxDB ni dans la sortie pour AWS Command Line Interface les commandes ou les opérations Timestream. API
- Le temps nécessaire pour supprimer une instance de base de données varie en fonction de la quantité de données supprimées et de la prise d'un instantané final.

Vous pouvez supprimer une instance de base de données à l'aide du AWS Management Console AWS Command Line Interface, du ou du TimestreamAPI. Vous devez fournir le nom de l'instance de base de données :

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez InfluxDB Databases, puis choisissez l'instance de base de données que vous souhaitez supprimer.
3. Sélectionnez Delete (Supprimer).
4. Entrez « Confirmer » dans le champ.
5. Sélectionnez Delete (Supprimer).

À l'aide du AWS Command Line Interface

Pour trouver l'instance IDs des instances de base de données dans votre compte, appelez la `list-db-instances` commande suivante :

```
aws timestream-influxdb list-db-instances \
--endpoint-url YOUR_ENDPOINT \
--region YOUR_REGION
```

Pour supprimer une instance de base de données à l'aide de AWSCLI, appelez la `delete-db-instance` commande avec les options suivantes :

```
aws timestream-influxdb list-db-instances \
--identifier YOUR_DB_INSTANCE \
```

Example Exemple

Pour Linux, macOS ou Unix :

```
aws timestream-influxdb delete-db-instance \
    --identifier mydbinstance
```

Pour Windows :

```
aws timestream-influxdb delete-db-instance ^
    --identifier mydbinstance
```

Déploiements d'instances de base de données multi-AZ

Amazon Timestream for InfluxDB fournit une haute disponibilité et une prise en charge du basculement pour les instances de base de données utilisant des déploiements multi-AZ avec une seule instance de base de données de secours. Ce type de déploiement est appelé déploiement d'instance de base de données multi-AZ. Amazon Timestream pour InfluxDB utilise la technologie de basculement Amazon.

Dans le cadre d'un déploiement d'instance de base de données multi-AZ, Amazon Timestream approvisionne et gère automatiquement une réplique de secours synchrone dans une autre zone de disponibilité. L'instance de base de données primaire est répliquée de manière synchrone dans les zones de disponibilité sur un réplica de secours afin d'assurer une redondance des données. L'exécution d'une instance de base de données à haute disponibilité peut améliorer la disponibilité en cas de défaillance de l'instance de base de données et d'interruption de la zone de disponibilité. Pour plus d'informations sur les zones de disponibilité, consultez AWS Régions et zones de disponibilité.

> **Note**
>
> L'option de haute disponibilité n'est pas une solution de mise à l'échelle pour les scénarios de lecture seule. Vous ne pouvez pas utiliser un réplica de secours pour traiter le trafic en lecture.

À l'aide de la console Amazon Timestream, vous pouvez créer un déploiement d'instance de base de données multi-AZ en spécifiant simplement l'option Créer une instance de secours dans la section Configuration de la disponibilité et de la durabilité lors de la création d'une instance de base de données. Vous pouvez également spécifier un déploiement d'instance de base de données multi-AZ avec Amazon Timestream AWS Command Line Interface ou Amazon TimestreamAPI. Utilisez la CLI commande `create-db-instance` ou, ou l'`CreateDBInstance`APIopération.

Les instances de base de données qui utilisent des déploiements d'instance de base de données multi-AZ peuvent avoir une latence d'écriture et de validation accrue par rapport à un déploiement mono-AZ. Cela peut se produire en raison de la réplication de données synchrone qui se produit. La latence peut changer si votre déploiement bascule vers la réplique de secours, même si elle AWS est conçue avec une connectivité réseau à faible latence entre les zones de disponibilité. Pour les charges de travail de production, nous vous recommandons d'utiliser le stockage IOPS inclus de 12 ou 16 Ko IOPS pour des performances rapides et constantes. Pour plus d'informations sur les classes d'instance de base de données, veuillez consulter Classes d'instances de base de données.

Configuration et gestion d'un déploiement multi-AZ

Timestream pour les déploiements multi-AZ d'InfluxDB ne peut avoir qu'un seul mode de veille. Lorsque le déploiement comporte une instance de base de données de secours, il s'agit d'un déploiement d'instance de base de données multi-AZ. Un déploiement d'instance de base de données multi-AZ comporte une instance de base de données de secours qui prend en charge le basculement, mais qui ne traite pas le trafic en lecture.

> **Important**
>
> Votre instance doit être associée à au moins deux sous-réseaux pour exécuter des mises à jour mono-AZ ou multi-AZ. Une fois l'instance créée, vous ne pouvez pas modifier son mode de déploiement du mode mono-AZ au mode multi-AZ.

Vous pouvez utiliser le AWS Management Console pour déterminer si votre instance de base de données est un déploiement mono-AZ ou multi-AZ.

En utilisant le AWS Management Console

1. Connectez-vous à la console Amazon Timestream pour InfluxDB AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez InfluxDB Databases, puis choisissez DB identifier.

Un déploiement d'instance de base de données Multi-AZ présente les caractéristiques suivantes :

- Il n'y a qu'une seule ligne pour l'instance de base de données.
- La valeur pour Role (Rôle) est Instance (Instance) ou Primary (Principal).
- La valeur pour Multi-AZ est Yes (Oui).

Processus de basculement pour Amazon Timestream

Si une interruption planifiée ou imprévue de votre instance de base de données résulte d'un défaut d'infrastructure, Amazon Timestream for InfluxDB passe automatiquement à une réplique de secours dans une autre zone de disponibilité si vous avez activé le mode multi-AZ. La durée du basculement dépend de l'activité de la base de données et d'autres conditions au moment où l'instance de base de données primaire est devenue indisponible. Les durées de basculement oscillent généralement entre 60 et 120 secondes. Cependant, les transactions importantes ou les processus de récupération longs peuvent augmenter le temps de basculement. Lorsque le basculement est terminé, la console Timestream peut prendre plus de temps pour refléter la nouvelle zone de disponibilité.

> **Note**
>
> Amazon Timestream gère automatiquement les basculements afin que vous puissiez reprendre les opérations de base de données le plus rapidement possible sans intervention administrative. L'instance de base de données primaire bascule automatiquement vers le réplica de secours si l'une des conditions décrites dans le tableau suivant se produit :

Raison du basculement	Description
Le système d'exploitation sous-jacent à l'instance de base de données Timestream est corrigé dans le cadre d'une opération hors ligne.	Un basculement a été déclenché pendant la fenêtre de maintenance d'un correctif du système d'exploitation ou d'une mise à jour de sécurité.
L'hôte principal de l'instance Timestream Multi-AZ n'est pas fonctionnel.	Le déploiement d'instance de base de données multi-AZ a détecté une instance de base de données primaire déficiente et a opéré un basculement.

Raison du basculement	Description
L'hôte principal de l'instance Timestream Multi-AZ est inaccessible en raison d'une perte de connectivité réseau.	La surveillance du flux temporel a détecté une défaillance de l'accessibilité réseau de l'instanc e de base de données principale et a déclenché un basculement.
L'instance Timestream a été modifiée par le client.	Une modification de l'instance de base de données Timesteam for InfluxDB a déclenché un basculement. Pour de plus amples informati ons, veuillez consulter Mise à jour des instances de base de données.
L'instance principale Multi-AZ de Timestream est occupée et ne répond pas.	L'instance de base de données primaire ne répond pas. Nous vous recommandons de procéder comme suit :* Examinez l'événeme nt pour détecter une utilisation excessive CPU de la mémoire ou de l'espace d'échange. * Évaluez votre charge de travail pour détermine r si vous utilisez la classe d'instance de base de données appropriée. Pour plus d'informa tions, consultez Classes d'instances de base de données.
Le volume de stockage sous-jacent à l'hôte principal de l'instance Multi-AZ de Timestream est tombé en panne.	Le déploiement d'instance de base de données multi-AZ a détecté un problème de stockage sur l'instance de base de données primaire et a opéré un basculement.

Configuration JVM TTL des recherches DNS de noms

Le mécanisme de basculement modifie automatiquement l'enregistrement du système de noms de domaine (DNS) de l'instance de base de données pour qu'il pointe vers l'instance de base de données de secours. Par conséquent, vous devez rétablir toutes les connexions existantes à votre instance de base de données. Dans un environnement de machine virtuelle Java (JVM), en raison du fonctionnement du mécanisme de DNS mise en cache Java, il se peut que vous deviez reconfigurer JVM les paramètres.

Les recherches de DNS noms de JVM caches. Lorsque le JVM convertit un nom d'hôte en adresse IP, il met l'adresse IP en cache pendant une période spécifiée, connue sous le nom de time-to-live(TTL).

Étant donné que les AWS ressources utilisent des entrées de DNS nom qui changent parfois, nous vous recommandons de configurer votre nom JVM avec une TTL valeur ne dépassant pas 60 secondes. Cela garantit que lorsque l'adresse IP d'une ressource change, votre application peut recevoir et utiliser la nouvelle adresse IP de la ressource en demandant le. DNS

Sur certaines configurations Java, la JVM valeur par défaut TTL est définie de manière à ne jamais actualiser les DNS entrées avant JVM le redémarrage. Ainsi, si l'adresse IP d'une AWS ressource change alors que votre application est toujours en cours d'exécution, elle ne peut pas utiliser cette ressource tant que vous ne l'avez pas redémarrée manuellement JVM et que les informations IP mises en cache ne sont pas actualisées. Dans ce cas, il est essentiel de définir le JVM « s » TTL afin qu'il actualise régulièrement ses informations IP mises en cache.

Vous pouvez obtenir la JVM valeur TTL par défaut en récupérant la valeur de la `networkaddress.cache.ttl` propriété :

```
String ttl = java.security.Security.getProperty("networkaddress.cache.ttl");
```

> **Note**
>
> La valeur par défaut TTL peut varier en fonction de la version de votre ordinateur JVM et de l'installation ou non d'un gestionnaire de sécurité. Beaucoup JVMs fournissent une valeur par défaut TTL inférieure à 60 secondes. Si vous utilisez un tel gestionnaire JVM sans utiliser de gestionnaire de sécurité, vous pouvez ignorer le reste de cette rubrique.
> Pour modifier les 'TTL, définissez JVM la valeur de la propriété networkaddress.cache.ttl. Utilisez l'une des méthodes suivantes selon vos besoins :
>
> - Pour définir la valeur de propriété de manière globale pour toutes les applications utilisant leJVM, définissez `networkaddress.cache.ttl` dans le `$JAVA_HOME/jre/lib/security/java.security` fichier.
>
>   ```
>   networkaddress.cache.ttl=60
>   ```
>
> - Pour définir la propriété localement pour votre application uniquement, définissez `networkaddress.cache.ttl` dans le code d'initialisation de votre application avant que les connexions réseau ne soient établies.

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

Configuration pour afficher les journaux InfluxDB sur les instances Timestream Influxdb

Par défaut, InfluxDB génère des journaux qui sont envoyés à stdout. Pour plus d'informations, voir Gérer les journaux InfluxDB

Pour afficher les journaux InfluxDB générés à partir de l'instance que vous avez créée via Timestream InfluxDB, nous vous offrons la possibilité de fournir des journaux horaires. Ces journaux seront envoyés dans un compartiment S3 spécifié que vous devez créer avant de créer votre instance.

- Avant de créer l'instance, le compartiment Amazon S3 fourni doit également autoriser Timestream-InfluxDB à envoyer des journaux à ce compartiment en fournissant une politique de bucket avec Timestream InfluxDB Service Principal comme suit (remplacez *{BUCKET_NAME}* avec le nom réel de votre compartiment 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/*"
        }
    ]
}
```

- Le bucket fourni doit se trouver dans le même compte et dans la même région que l'instance Timestream InfluxDB que vous avez créée

  Voici la commande que vous pouvez appeler pour qu'une instance reçoive les journaux d'afflux :

```
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
```

Voici le format de ce paramètre.

```
-- log-delivery-configuration
{
    "S3Configuration": {
      "BucketName": "string",
      "Enabled": true|false
    }
}
```

- Ce champ n'est pas obligatoire et la journalisation n'est pas activée par défaut.
- Ne pas définir ce champ revient à ne pas activer les journaux.
- Les journaux seront envoyés au compartiment spécifié avec un préfixe de`InfluxLogs/`.
- Après avoir créé l'instance, vous pouvez modifier la configuration de livraison du journal à l'aide de la `update-db-instance` API commande.

InfluxDB propose différents types de journaux. Ceux-ci peuvent être configurés en définissant les paramètres InfluxDB. Utilisez les paramètres flux-log-enabled et au niveau du journal pour configurer le type de journaux émis par l'instance. Pour de plus amples informations, veuillez consulter Paramètres et valeurs de paramètres pris en charge.

Ajout d'identifications et d'étiquettes aux ressources

Vous pouvez étiqueter Amazon Timestream pour les ressources InfluxDB à l'aide de balises. Les balises vous permettent de classer vos ressources de différentes manières, par exemple en fonction

de leur objectif, de leur propriétaire, de leur environnement ou d'autres critères. Les balises vous permettent d'effectuer les actions suivantes :

- Identifier rapidement une ressource en fonction des balises que vous lui avez attribuées.
- Consultez AWS les factures ventilées par tags.

Le balisage est pris en charge par AWS des services tels qu'Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream pour InfluxDB, etc. Un balisage efficace peut fournir un aperçu sur les coûts en vous permettant de créer des rapports sur les services associés à une balise spécifique.

En dernier lieu, il est conseillé de suivre les politique de balisage optimales. Pour plus d'informations, consultez Politiques d'étiquetage AWS.

Restrictions de balisage

Chaque étiquette est constituée d'une clé et d'une valeur, que vous définissez. Les restrictions suivantes s'appliquent :

- Chaque instance de base de données Timestream for InfluxDB ne peut avoir qu'une seule balise avec la même clé. Si vous essayez d'ajouter une balise existante, la valeur de la balise existante est mise à jour avec la nouvelle valeur.
- Une valeur agit comme un descripteur au sein d'une catégorie de balises. Dans Timestream pour InfluxDB, la valeur ne peut pas être vide ou nulle.
- Les clés et valeurs de balise sont sensibles à la casse.
- La longueur de clé maximale est de 128 caractères Unicode.
- La longueur de valeur maximale est de 256 caractères Unicode.
- Les caractères autorisés sont les lettres, les espaces blancs et les chiffres, ainsi que les caractères spéciaux suivants : `+ - = . _ : /`
- Le nombre maximum d'identifications par ressource est de 50.
- AWS-les noms et valeurs des balises assignés reçoivent automatiquement le `aws:` préfixe, que vous ne pouvez pas attribuer. AWS-les noms de balises attribués ne sont pas pris en compte dans la limite de 50 balises. Les noms des balises attribuées par l'utilisateur ont le préfixe `user:` dans le rapport de répartition des coûts.
- Vous ne pouvez pas antidater l'application d'une balise.

Meilleures pratiques de sécurité pour Timestream pour InfluxDB

Optimisez les écritures dans InfluxDB

Comme toute autre base de données de séries chronologiques, InfluxDB est conçue pour pouvoir ingérer et traiter des données en temps réel. Pour que le système fonctionne au mieux, nous recommandons les optimisations suivantes lors de l'écriture de données dans InfluxDB :

- Écritures par lots : Lorsque vous écrivez des données dans InfluxDB, écrivez les données par lots afin de minimiser la surcharge réseau liée à chaque demande d'écriture. La taille de lot optimale est de 5 000 lignes de protocole de ligne par demande d'écriture. Pour écrire plusieurs lignes dans une requête, chaque ligne du protocole de ligne doit être délimitée par une nouvelle ligne (\n).
- Trier les balises par clé : Avant d'écrire des points de données dans InfluxDB, triez les balises par clé dans l'ordre lexicographique.

```
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
```

- Utilisez la précision temporelle la plus grossière possible : — InfluxDB écrit les données avec une précision de nanoseconde, mais si vos données ne sont pas collectées en nanosecondes, il n'est pas nécessaire d'écrire à cette précision. Pour de meilleures performances, utilisez la précision la plus grossière possible pour les horodatages. Vous pouvez spécifier la précision d'écriture dans les cas suivants :
  - Lorsque vous utilisez le, SDK vous pouvez spécifier l'attribut time de votre point WritePrecision lors de la définition. Pour plus d'informations sur les bibliothèques clientes InfluxDB, consultez la documentation InfluxDB.
  - Lorsque vous utilisez Telegraf, vous configurez la précision temporelle dans la configuration de l'agent Telegraf. La précision est spécifiée sous la forme d'un intervalle avec une unité entière + (par exemple, 0s,10ms,2us,4s). Les unités de temps valides sont « ns », « us », « ms » et « s ».

    ```
    [agent]
     interval ="10s"
     metric_batch_size="5000"
     precision = "0s"
    ```

- Utiliser la compression gzip : — Utilisez la compression gzip pour accélérer les écritures dans InfluxDB et réduire la bande passante réseau. Les tests de performance ont montré une amélioration de la vitesse jusqu'à 5 fois lorsque les données sont compressées.
  - Lorsque vous utilisez Telegraf, dans la configuration du plugin de sortie InfluxDB_v2 dans votre fichier telegraf.conf, définissez l'option content_encoding sur gzip :

    ```
    [[outputs.influxdb_v2]]
      urls = ["http://localhost:8086"]
      # ...
      content_encoding = "gzip"
    ```

  - Lorsque vous utilisez des bibliothèques clientes, chaque bibliothèque cliente InfluxDB fournit des options pour compresser les demandes d'écriture ou applique la compression par défaut. La méthode d'activation de la compression est différente pour chaque bibliothèque. Pour des instructions spécifiques, consultez la documentation InfluxDB
  - Lorsque vous utilisez le point de API `/api/v2/write` terminaison InfluxDB pour écrire des données, compressez-les avec gzip et définissez l'en-tête Content-Encoding sur gzip.

Conception axée sur la performance

Concevez votre schéma pour des requêtes plus simples et plus performantes. Les directives suivantes garantiront que votre schéma sera facile à interroger et optimiseront les performances des requêtes :

- Conception adaptée aux requêtes : choisissez des mesures, des clés de balise et des clés de champ faciles à interroger. Pour atteindre cet objectif, suivez les principes suivants :
  - Utilisez des mesures portant un nom simple et décrivant le schéma avec précision.
  - Évitez d'utiliser le même nom pour une clé de balise et une clé de champ dans le même schéma.
  - Évitez d'utiliser des mots clés Flux réservés et des caractères spéciaux dans les clés de balise et de champ.
  - Les balises stockent des métadonnées qui décrivent les champs et sont communes à de nombreux points de données.
  - Les champs stockent des données uniques ou très variables, généralement des points de données numériques.

  - Les mesures et les clés ne doivent pas contenir de données, mais être utilisées pour agréger ou décrire des données. Les données seront stockées sous forme de balises et de valeurs de champs.

- Gardez le contrôle de votre cardinalité des séries chronologiques La cardinalité élevée des séries est l'une des principales causes de la diminution des performances d'écriture et de lecture dans InfluxDB. Dans le contexte d'InfluxDB, la cardinalité élevée fait référence à la présence d'un très grand nombre de valeurs de balises uniques. Les valeurs des balises sont indexées dans InfluxDB, ce qui signifie qu'un très grand nombre de valeurs uniques générera un index plus important, ce qui peut ralentir l'ingestion de données et les performances des requêtes.

  Pour mieux comprendre et résoudre les problèmes potentiels liés à une cardinalité élevée, vous pouvez suivre les étapes suivantes :

  - Comprendre les causes d'une cardinalité élevée
  - Mesurez la cardinalité de vos seaux
  - Prendre des mesures pour résoudre le problème de la cardinalité élevée

- Causes de la cardinalité élevée des séries InfluxDB indexe les données en fonction des mesures et des balises pour accélérer la lecture des données. Chaque ensemble d'éléments de données indexés forme une clé de série. Les balises contenant des informations très variablesIDs, telles que des chaînes uniques, des hachages et des chaînes aléatoires, génèrent un grand nombre de séries, également appelées cardinalité de série élevée. La cardinalité élevée des séries est le principal moteur de l'utilisation élevée de la mémoire dans InfluxDB.
- Mesurer la cardinalité des séries Si vous rencontrez des ralentissements de performances ou si vous constatez une utilisation croissante de la mémoire dans votre instance Timestream for InfluxDB, nous vous recommandons de mesurer la cardinalité des séries de vos buckets.

  InfluxDB fournit des fonctions qui vous permettent de mesurer la cardinalité des séries à la fois dans Flux et InfluxQL.

  - Dans Flux, utilisez la fonction `influxdb.cardinality()`
  - Dans FluxQL, utilisez la commande `SHOW SERIES CARDINALITY`

  Dans les deux cas, le moteur renverra le nombre de clés de série uniques présentes dans vos données. N'oubliez pas qu'il n'est pas recommandé d'avoir plus de 10 millions de clés de série sur l'une de vos instances Timestream pour InfluxDB.

- Causes de la cardinalité élevée des séries Si vous constatez que l'un de vos compartiments présente une cardinalité élevée, vous pouvez prendre quelques mesures pour y remédier :

- Vérifiez vos balises : assurez-vous que vos charges de travail ne génèrent pas de cas où les balises ont des valeurs uniques pour la plupart des entrées. Cela peut se produire dans les cas où le nombre de valeurs de balises uniques augmente toujours avec le temps, ou si des messages de type journal sont écrits dans la base de données où chaque message comporterait une combinaison unique d'horodatage, de balises, etc. Vous pouvez utiliser le code Flux suivant pour vous aider à déterminer quels tags contribuent le plus à vos problèmes de cardinalité élevée :

```
// 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")
```

Si votre cardinalité est très élevée, la requête ci-dessus peut expirer. Si vous rencontrez un délai d'attente, exécutez les requêtes ci-dessous, une par une.

Générez une liste de tags :

```
// Generate a list of tagsimport "influxdata/influxdb/schema"

schema.tagKeys(bucket: "example-bucket")
```

Comptez les valeurs de balise uniques pour chaque balise :

```
// 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()
```

Nous vous recommandons de les exécuter à différents moments pour identifier le tag qui croît le plus rapidement.

- Améliorez votre schéma : suivez les recommandations de modélisation décrites dans notreMeilleures pratiques de sécurité pour Timestream pour InfluxDB.
- Supprimez ou agrégez les anciennes données pour réduire la cardinalité : déterminez si vos cas d'utilisation nécessitent ou non toutes les données à l'origine de vos problèmes de cardinalité élevée. Si ces données ne sont plus nécessaires ou si vous n'y accédez plus fréquemment, vous pouvez les agréger, les supprimer ou les exporter vers un autre moteur tel que Timestream for Live Analytics pour un stockage et une analyse à long terme.

Résolution des problèmes

Avertissement indiquant que la version « dev » n'est pas reconnue

L'avertissement « WARN Impossible d'analyser la version « dev » signalée par le serveur, en supposant que les dernières sauvegardes/restaurations APIs soient prises en charge » peut s'afficher pendant la migration. Cet avertissement peut être ignoré.

La migration a échoué pendant la phase de restauration

En cas d'échec de la migration pendant la phase de restauration, les utilisateurs peuvent utiliser le `--retry-restore-dir` drapeau pour tenter à nouveau la restauration. Utilisez l'`--retry-restore-dir`indicateur indiquant le chemin d'accès à un répertoire précédemment sauvegardé pour sauter l'étape de sauvegarde et réessayer l'étape de restauration. Le répertoire de sauvegarde créé et utilisé pour une migration sera indiqué en cas d'échec de la migration lors de la restauration.

Les raisons possibles de l'échec d'une restauration sont les suivantes :

- Jeton de destination InfluxDB non valide — Un bucket existant dans l'instance de destination avec le même nom que dans l'instance source. Pour les migrations de compartiments individuels, utilisez l'`--dest-bucket`option permettant de définir un nom unique pour le compartiment migré.

- Défaillance de connectivité, que ce soit avec les hôtes source ou de destination, ou avec un compartiment S3 en option.

Directives opérationnelles de base d'Amazon Timestream pour InfluxDB

Vous trouverez ci-dessous les directives opérationnelles de base que tout le monde doit suivre lorsqu'il travaille avec Amazon Timestream pour InfluxDB. Notez que l'accord de niveau de service Amazon Timestream pour InfluxDB exige que vous suiviez les directives suivantes :

- Utilisez des métriques pour surveiller votre mémoire et votre utilisation du stockage. CPU Vous pouvez configurer Amazon CloudWatch pour qu'il vous avertisse lorsque les habitudes d'utilisation changent ou lorsque vous approchez de la capacité de votre déploiement. De cette façon, vous pouvez maintenir les performances et la disponibilité du système.
- Augmentez la capacité de votre instance de base de données lorsque vous atteignez la limite de stockage. Vous devrez disposer de capacités de mémoire et de stockage supplémentaires pour vous adapter aux hausses imprévues des besoins de vos applications. N'oubliez pas qu'à ce stade, vous devrez créer une nouvelle instance et migrer vos données pour y parvenir.
- Si la charge de travail de votre base de données exige plus d'I/O que vous avez provisionné, la récupération suite à un basculement ou un échec de la base de données est lente. Pour augmenter la capacité d'I/O d'une instance de base de données, procédez comme suit :
  - Migrez vers une autre instance de base de données dotée d'une capacité d'E/S supérieure.
  - Si vous utilisez déjà le stockage IOPS inclus dans Influx, configurez un type de stockage avec un niveau de stockage supérieur IOPS inclus.
- Si votre application cliente met en cache les données du Domain Name Service (DNS) de vos instances de base de données, définissez une valeur time-to-live (TTL) inférieure à 30 secondes. L'adresse IP sous-jacente d'une instance de base de données peut changer après un basculement. La mise en cache prolongée DNS des données peut donc entraîner des échecs de connexion. Il se peut que votre application essaie de se connecter à une adresse IP qui n'est plus en service.

RAMRecommandations relatives aux instances de base de

L'une des meilleures pratiques d'Amazon Timestream pour les performances d'InfluxDB consiste à en allouer RAM suffisamment pour que votre espace de travail réside presque entièrement en mémoire. L'ensemble de travail est les données et les index fréquemment utilisés sur votre instance. Plus vous utilisez l'instance de base de données, plus l'ensemble de travail se développera.

Sécurité dans Timestream pour InfluxDB

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. L'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Timestream pour InfluxDB, voir AWS Services concernés par programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre organisation, et la législation et la réglementation applicables.

Cette documentation vous aidera à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de Timestream pour InfluxDB. Les rubriques suivantes vous montrent comment configurer Timestream pour InfluxDB afin d'atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui peuvent vous aider à surveiller et à sécuriser votre Timestream pour les ressources InfluxDB.

Rubriques

- Analyse de configuration et de vulnérabilité dans Timestream pour InfluxDB
- Réponse aux incidents dans Timestream pour InfluxDB
- Amazon Timestream pour API InfluxDB et points de terminaison d'interface () VPC AWS PrivateLink
- Meilleures pratiques de sécurité pour Timestream pour InfluxDB

Présentation

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon Timestream pour InfluxDB. Les rubriques suivantes vous montrent comment configurer Amazon Timestream pour InfluxDB afin d'atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser votre Amazon Timestream pour les ressources InfluxDB.

Vous pouvez gérer l'accès à votre Amazon Timestream pour les ressources InfluxDB et à vos bases de données sur une instance de base de données. La méthode que vous utilisez pour gérer l'accès dépend du type de tâche que l'utilisateur doit effectuer avec Amazon Timestream pour InfluxDB :

- Exécutez votre instance de base de données dans un cloud privé virtuel (VPC) basé sur le VPC service Amazon pour le contrôle d'accès au réseau.
- Utilisez les politiques AWS Identity and Access Management (IAM) pour attribuer des autorisations qui déterminent qui est autorisé à gérer les ressources Amazon Timestream pour InfluxDB. Par exemple, vous pouvez l'utiliser IAM pour déterminer qui est autorisé à créer, décrire, modifier et supprimer des instances de base de données, à étiqueter des ressources ou à modifier des groupes de sécurité.
- Utilisez des groupes de sécurité pour contrôler les adresses IP ou les EC2 instances Amazon qui peuvent se connecter à vos bases de données sur une instance de base de données. Lorsque vous créez une instance de base de données pour la première fois, elle n'est accessible que par le biais de règles spécifiées par un groupe de sécurité associé.
- Utilisez des connexions Secure Socket Layer (SSL) ou Transport Layer Security (TLS) avec vos instances de base de données.
- Utilisez les fonctionnalités de sécurité de votre moteur InfluxDB pour contrôler qui peut se connecter aux bases de données sur une instance de base de données. Ces fonctions agissent comme si la base de données se trouvait sur votre réseau local. Pour de plus amples informations, veuillez consulter Sécurité dans Timestream pour InfluxDB.

> **Note**
>
> Vous devez uniquement configurer la sécurité de vos cas d'utilisation. Il n'est pas nécessaire de configurer l'accès sécurisé pour les processus gérés par Amazon Timestream for InfluxDB. Cela inclut, par exemple, la création de sauvegardes et la réplication de données entre une instance de base de données principale et un réplica en lecture, et d'autres processus.

Rubriques

Sûreté générale

Rubriques

Autorisations

Les utilisateurs d'InfluxDB doivent bénéficier d'autorisations de moindre privilège. Seuls les jetons accordés à des utilisateurs spécifiques, au lieu des jetons d'opérateur, doivent être utilisés lors de la migration.

Timestream for InfluxDB utilise des autorisations pour contrôler IAM les autorisations des utilisateurs. Nous recommandons aux utilisateurs d'avoir accès aux actions et aux ressources spécifiques dont ils ont besoin. Pour plus d'informations, voir Accorder un accès avec le moindre privilège.

Accès réseau

Le script de migration Influx peut fonctionner localement, faisant migrer les données entre deux instances InfluxDB sur le même système, mais il est supposé que le principal cas d'utilisation des migrations sera la migration des données sur le réseau, qu'il s'agisse d'un réseau local ou public. Cela s'accompagne de considérations de sécurité. Le script de migration Influx vérifiera, par défaut,

les TLS certificats des instances TLS activées : nous recommandons aux utilisateurs de les activer TLS dans leurs instances InfluxDB et de ne pas utiliser l'`--skip-verify`option pour le script.

Nous vous recommandons d'utiliser une liste d'autorisation pour limiter le trafic réseau à des sources auxquelles vous vous attendez. Vous pouvez le faire en limitant le trafic réseau aux instances InfluxDB uniquement à partir de sources connues. IPs

Dépendances

Les dernières versions majeures de toutes les dépendances doivent être utilisées, y compris InfluxCLI, InfluxDB, Python, le module Requests et les dépendances facultatives telles que `mountpoint-s3` et. `rclone`

Compartiments S3

Si les compartiments S3 sont utilisés comme stockage temporaire pour la migration, nous vous recommandons d'activerTLS, de versionner et de désactiver l'accès public.

Utilisation de compartiments S3 pour la migration

1. Ouvrez le AWS Management Console, accédez à Amazon Simple Storage Service, puis choisissez Buckets.
2. Choisissez le bucket que vous souhaitez utiliser.
3. Choisissez l’onglet Permissions (Autorisations).
4. Sous Block public access (bucket settings) (Bloquer l'accès public (paramètres de compartiment)), choisissez Edit (Modifier).
5. Cochez Bloquer tout accès public.
6. Sélectionnez Enregistrer les modifications.
7. Sous Politique de compartiment, choisissez Modifier.
8. Entrez le texte suivant, en le <example-bucket>remplaçant par le nom de votre compartiment, pour imposer l'utilisation de la TLS version 1.2 ou ultérieure pour les connexions :

```
{
    "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. Sélectionnez Enregistrer les modifications.
10. Choisissez l'onglet Propriétés.
11. Sous Bucket Versioning (Gestion des versions de compartiment), choisissez Edit (Modifier).
12. Cochez Activer.
13. Sélectionnez Enregistrer les modifications.

Pour plus d'informations sur les meilleures pratiques de sécurité des compartiments Amazon S3, consultez la section Meilleures pratiques de sécurité pour Amazon Simple Storage Service.

Authentification de base de données avec Amazon Timestream pour InfluxDB

Amazon Timestream for InfluxDB prend en charge deux méthodes pour authentifier les utilisateurs de la base de données.

L'authentification de la base de données par mot de passe et jeton d'accès utilise différentes méthodes d'authentification auprès de la base de données. Par conséquent, un utilisateur spécifique peut se connecter à une base de données en utilisant une seule méthode d'authentification. Dans les deux cas, InfluxDB effectue toute l'administration des comptes utilisateurs et API des jetons.

Authentification par mot de passe

Au cours du processus de création de l'instance de base de données InfluxDB, vous avez créé une organisation, un utilisateur et un mot de passe. L'utilisateur est autorisé à tout gérer dans votre instance de base de données Timestream for InfluxDB. Avec cette combinaison de nom d'utilisateur et de mot de passe, vous pourrez accéder à LogIn votre instance à l'aide de l'InfluxUI et également utiliser l'Influx CLI pour générer un jeton d'opérateur.

Un jeton d'opérateur est nécessaire pour créer des utilisateurs, supprimer des buckets, des organisations, etc. Pour de plus amples informations, veuillez consulter Options d'authentification de base de données.

APIjetons

APILes jetons InfluxDB garantissent une interaction sécurisée entre InfluxDB et les outils externes tels que les clients ou les applications. Un API jeton appartient à un utilisateur spécifique et identifie les autorisations InfluxDB au sein de l'organisation de l'utilisateur.

Il existe trois types de API jetons dans InfluxDB :

- Jeton d'opérateur : accorde un accès complet en lecture et en écriture à toutes les organisations et à toutes les ressources de l'organisation dans InfluxDB 2.xOSS. Certaines opérations, par exemple la récupération de la configuration du serveur, nécessitent des autorisations d'opérateur. Pour créer un jeton d'opérateur manuellement avec l'interface utilisateur InfluxDB `api/v2`API, ou Influx une CLI fois le processus de configuration terminé, vous devez utiliser un jeton d'opérateur existant ou votre nom d'utilisateur et votre mot de passe. Pour créer un nouveau jeton d'opérateur sans utiliser un jeton existant, consultez l'authentification CLI de restauration influxd.

  > **Important**
  >
  > Étant donné que les jetons d'opérateur ont un accès complet en lecture et en écriture à toutes les organisations de la base de données, nous vous recommandons de créer un jeton All-Access pour chaque organisation et de les utiliser pour gérer InfluxDB. Cela permet d'éviter les interactions accidentelles entre les organisations.

- APIJeton d'accès illimité : accorde un accès complet en lecture et en écriture à toutes les ressources d'une organisation.
- Jetons de lecture/écriture : octroie un accès en lecture, un accès en écriture ou les deux à des compartiments spécifiques d'une organisation.

Tous les InfluxDb jetons sont des jetons à longue durée de vie sans date d'expiration définie. Il n'est donc pas recommandé d'utiliser votre opérateur ou tous les jetons d'accès pour envoyer des données de surveillance provenant de vos clients ou agents Telegraf, ni de les intégrer dans vos applications de tableau de bord. Pour ces applications, créez des jetons de lecture/écriture avec uniquement les autorisations nécessaires pour effectuer le travail. Pour plus d'informations sur la création d'un jeton InfluxDB, consultez Créer un jeton.

Secrets

Les jetons d'opérateur InfluxDB sont générés lors de la configuration de l'instance ; d'autres types de jetons, tels que les jetons d'accès complet et les jetons de lecture/écriture, peuvent être créés à l'aide de la fonction de rotation multi-utilisateurs InfluxCLI, Influx v2 API ou Timestream for InfluxDB. Consultez Gérer les API jetons pour savoir comment générer, afficher, attribuer et supprimer des jetons.

Nous vous recommandons de faire pivoter Timestream pour les jetons InfluxDB qui utilisent AWS Secrets Manager et stockent souvent des jetons via des variables d'environnement. Voir Utiliser des jetons pour l'utilisation de jetons dans les variables d'environnement et Transformer le secret pour savoir comment faire pivoter Timestream pour les utilisateurs et les jetons d'InfluxDB.

Voir aussi :

- Sécurité de l'infrastructure dans Amazon Timestream pour InfluxDB
- Meilleures pratiques de sécurité pour Timestream pour InfluxDB

Comment Amazon Timestream pour InfluxDB utilise les secrets

Timestream for InfluxDB prend en charge l'authentification par nom d'utilisateur et mot de passe via l'interface utilisateur, ainsi que les informations d'identification par jeton pour les connexions client et application avec le moindre privilège. Les utilisateurs de Timestream for InfluxDB ont des `allAccess` autorisations au sein de leur organisation, tandis que les jetons peuvent avoir n'importe quel ensemble d'autorisations. Conformément aux meilleures pratiques API en matière de gestion sécurisée des jetons, des utilisateurs doivent être créés pour gérer les jetons afin d'obtenir un accès précis au sein d'une organisation. Des informations supplémentaires sur les meilleures pratiques d'administration avec Timestream pour InfluxDB sont disponibles dans la documentation Influxdata.

AWS Secrets Manager est un service de stockage secret que vous pouvez utiliser pour protéger les informations d'identification, API les clés et autres informations secrètes de base de données.

Ensuite, dans votre code, vous pouvez remplacer les informations d'identification codées en dur par un API appel à Secrets Manager. Cela permet de s'assurer que le secret ne peut pas être compromis par quelqu'un qui examine votre code, car le secret n'existe pas. Pour une présentation de Secrets Manager, voir Qu'est-ce que AWS Secrets Manager ?

Lorsque vous créez une instance de base de données, Timestream for InfluxDB crée automatiquement un secret d'administration que vous pouvez utiliser avec la fonction de rotation multi-utilisateurs. AWS Lambda Afin de faire pivoter Timestream pour les utilisateurs et les jetons d'InfluxDB, vous devez créer un nouveau secret à la main pour chaque utilisateur ou jeton que vous souhaitez faire pivoter. Chaque secret peut être configuré pour alterner selon un calendrier à l'aide d'une fonction Lambda. Le processus de configuration d'un nouveau secret rotatif consiste à télécharger le code de la fonction Lambda, à configurer le rôle Lambda, à définir le nouveau secret et à configurer le calendrier de rotation du secret.

Qu'y a-t-il dans le secret

Lorsque vous stockez les informations d'identification de l'utilisateur Timestream for InfluxDB dans le secret, utilisez le format suivant.

Utilisateur unique :

```
{
  "engine": "<required: must be set to 'timestream-influxdb'>",
  "username": "<required: username>",
  "password": "<required: password>",
  "dbIdentifier": "<required: DB identifier>"
}
```

Lorsque vous créez une instance Timestream pour InfluxDB, un secret d'administration est automatiquement stocké dans Secrets Manager avec les informations d'identification à utiliser avec la fonction Lambda multi-utilisateurs. Définissez le sur `adminSecretArn` la `Authentication Properties Secret Manager ARN` valeur trouvée sur la page de résumé de l'instance de base ARN de données ou sur le secret d'administration. Pour créer un nouveau secret d'administrateur, vous devez déjà disposer des informations d'identification associées et les informations d'identification doivent disposer de privilèges d'administrateur.

Lorsque vous stockez les informations d'identification du jeton Timestream for InfluxDB dans le secret, utilisez le format suivant.

Multi-utilisateur :

```
{
  "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']>"
}
```

Lorsque vous stockez les informations d'identification de l'administrateur Timestream for InfluxDB dans le secret, utilisez le format suivant :

Secret d'administration :

```
{
  "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>"
}
```

Pour activer la rotation automatique du secret, celui-ci doit être dans la bonne JSON structure. Découvrez Transformer le secret comment faire pivoter Timestream pour les secrets d'InfluxDB.

Modification du secret

Les informations d'identification générées lors du processus de création de l'instance Timestream for InfluxDB sont stockées dans un secret Secrets Manager de votre compte. L'objet de GetDbInstanceréponse contient un `influxAuthParametersSecretArn` code contenant le nom de ressource Amazon (ARN) associé à ce secret. Le secret ne sera renseigné qu'une fois que votre instance Timestream for InfluxDB sera disponible. Il s'agit d'une READONLY copie, car aucun updates/modifications/deletions élément de ce secret n'a d'impact sur l'instance de base de données créée. Si vous supprimez ce secret, la APIréponse fera toujours référence au secret supprimé ARN.

Pour créer un nouveau jeton dans l'instance Timestream for InfluxDB plutôt que de stocker les informations d'identification des jetons existants, vous pouvez créer des jetons non opérateurs en laissant la `token` valeur vide dans le secret et en utilisant la fonction de rotation multi-utilisateurs avec la variable d'environnement Lambda `AUTHENTICATION_CREATION_ENABLED` définie sur. `true` Si vous créez un nouveau jeton, les autorisations définies dans le secret sont attribuées au jeton et ne peuvent pas être modifiées après la première rotation réussie. Pour plus d'informations sur la rotation des secrets, consultez Rotating AWS Secrets Manager Secrets Manager.

Si un secret est supprimé, l'utilisateur ou le jeton associé dans l'instance Timestream for InfluxDB ne sera pas supprimé.

Transformer le secret

Vous utilisez les fonctions Lambda de rotation Timestream for InfluxDB pour un ou plusieurs utilisateurs pour faire pivoter Timestream pour les informations d'identification des utilisateurs et des jetons InfluxDB. Utilisez la fonction Lambda mono-utilisateur pour faire pivoter les informations d'identification utilisateur pour votre instance Timestream for InfluxDB, et utilisez la fonction Lambda multi-utilisateurs pour faire pivoter les informations d'identification des jetons pour votre instance Timestream for InfluxDB.

La rotation des utilisateurs et des jetons à l'aide des fonctions Lambda mono-utilisateur et multi-utilisateurs est facultative. Le flux temporel des informations d'identification InfluxDB n'expire jamais et toute information d'identification exposée présente un risque d'actions malveillantes contre votre instance de base de données. L'avantage de la rotation de Timestream pour les informations d'identification InfluxDB avec Secrets Manager est une couche de sécurité supplémentaire qui limite le vecteur d'attaque des informations d'identification exposées à la fenêtre temporelle précédant le prochain cycle de rotation. Si aucun mécanisme de rotation n'est en place pour votre instance de base de données, toutes les informations d'identification exposées seront valides jusqu'à ce qu'elles soient supprimées manuellement.

Vous pouvez configurer Secrets Manager pour qu'il fasse automatiquement pivoter les secrets selon un calendrier que vous spécifiez. Cela vous permet de remplacer les secrets à long terme par ceux à court terme, ce qui réduit considérablement le risque de mise en péril. Pour plus d'informations sur la rotation des secrets avec Secrets Manager, voir Rotate Secrets Manager Secrets Secrets AWS Secrets Secrets Secrets Secrets Secrets Secrets.

Rotation des utilisateurs

Lorsque vous effectuez une rotation d'utilisateurs à l'aide de la fonction Lambda mono-utilisateur, un nouveau mot de passe aléatoire est attribué à l'utilisateur après chaque rotation. Pour plus

d'informations sur la façon d'activer la rotation automatique, voir Configurer la rotation automatique pour les secrets non liés à la base de données AWS Secrets Manager.

Rotation des secrets d'administration

Pour faire pivoter un secret d'administration, vous utilisez la fonction de rotation mono-utilisateur. Vous devez ajouter les `dbIdentifier` valeurs `engine` et au secret, car ces valeurs ne sont pas automatiquement renseignées lors de l'initialisation de la base de données. Consultez Qu'y a-t-il dans le secret le modèle secret complet.

Pour localiser un secret d'administration pour une instance Timestream for InfluxDB, vous utilisez le secret d'administration de la page de résumé ARN de l'instance Timestream for InfluxDB. Il est recommandé de faire pivoter tous les secrets d'administration de Timestream pour InfluxDB, car les utilisateurs administrateurs ont des autorisations élevées pour l'instance Timestream for InfluxDB.

Fonction de rotation Lambda

Vous pouvez faire pivoter un flux temporel pour un utilisateur InfluxDB à l'aide de la fonction de rotation utilisateur unique en utilisant le code Qu'y a-t-il dans le secret avec un nouveau secret et en ajoutant les champs obligatoires pour votre utilisateur Timestream pour InfluxDB. Pour plus d'informations sur les fonctions Lambda de rotation secrètes, voir Rotation par fonction Lambda.

La fonction de rotation mono-utilisateur s'authentifie auprès de l'instance de base de données Timestream for InfluxDB à l'aide des informations d'identification définies dans le secret, puis génère un nouveau mot de passe aléatoire et définit le nouveau mot de passe pour l'utilisateur. Pour plus d'informations sur les fonctions Lambda de rotation secrètes, voir Rotation par fonction Lambda.

Autorisations du rôle d'exécution de la fonction Lambda

Utilisez la IAM politique suivante comme rôle pour la fonction Lambda mono-utilisateur. La politique donne à la fonction Lambda les autorisations requises pour effectuer une rotation secrète pour Timestream pour les utilisateurs d'InfluxDB.

Remplacez tous les éléments listés ci-dessous dans la IAM politique par les valeurs de votre AWS compte :

- {rotating_secret_arn} — Le secret faisant l'objet ARN de la rotation se trouve dans les détails du secret du Secrets Manager.
- {db_instance_arn} — Le flux temporel pour l'instance InfluxDB se trouve sur la page de résumé du flux temporel pour l'instance ARN 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"
        }
    ]
}
```

Jetons rotatifs

Vous pouvez faire pivoter un jeton Timestream pour InfluxDB à l'aide de la fonction de rotation multi-utilisateurs en utilisant le code Qu'y a-t-il dans le secret avec un nouveau secret et en ajoutant les champs obligatoires pour votre jeton Timestream for InfluxDB. Pour plus d'informations sur les fonctions Lambda de rotation secrètes, voir Rotation par fonction Lambda.

Vous pouvez faire pivoter un jeton Timestream pour InfluxDB en utilisant la fonction Lambda multi-utilisateurs Timestream for InfluxDB. Définissez la variable d'`AUTHENTICATION_CREATION_ENABLED`environnement sur `true` dans la configuration Lambda pour permettre la création de jetons. Pour créer un nouveau jeton, utilisez le Qu'y a-t-il dans le secret pour votre valeur secrète. Omettez la paire `token` clé-valeur dans le nouveau secret et définissez

la `type` valeur sur`allAccess`, ou définissez les autorisations spécifiques et définissez le type sur. `custom` La fonction de rotation créera un nouveau jeton lors du premier cycle de rotation. Vous ne pouvez pas modifier les autorisations du jeton en modifiant le secret après la rotation et toutes les rotations suivantes utiliseront les autorisations définies dans l'instance de base de données.

Fonction de rotation Lambda

La fonction de rotation multi-utilisateurs fait pivoter les informations d'identification du jeton en créant un nouveau jeton d'autorisation identique à l'aide des informations d'identification figurant dans le secret d'administration. La fonction Lambda valide la valeur du jeton dans le secret avant de créer le jeton de remplacement, de stocker la nouvelle valeur du jeton dans le secret et de supprimer l'ancien jeton. Si la fonction Lambda crée un nouveau jeton, elle validera d'abord que la variable d'`AUTHENTICATION_CREATION_ENABLED`Environnement est définie sur`true`, qu'il n'y a aucune valeur de jeton dans le secret et que le type de jeton n'est pas un opérateur de type.

Autorisations du rôle d'exécution de la fonction Lambda

Utilisez la IAM politique suivante comme rôle pour la fonction Lambda multi-utilisateurs. La politique donne à la fonction Lambda les autorisations requises pour effectuer une rotation secrète pour Timestream pour les jetons InfluxDB.

Remplacez tous les éléments listés ci-dessous dans la IAM politique par les valeurs de votre AWS compte :

- {rotating_secret_arn} — Le secret faisant l'objet ARN de la rotation se trouve dans les détails du secret du Secrets Manager.
- {authentication_properties_admin_secret_arn} — Le secret d'administration Timestream for InfluxDB se trouve sur la page de résumé du Timestream for InfluxDB instance. ARN
- {db_instance_arn} — Le flux temporel pour l'instance InfluxDB se trouve sur la page de résumé du flux temporel pour l'instance ARN 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"
        }
    ]
}
```

Protection des données dans Timestream pour InfluxDB

Le modèle de responsabilité AWS partagée Le modèle s'applique à la protection des données dans Amazon Timestream pour InfluxDB. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la section Confidentialité des données FAQ. Pour plus d'informations sur la protection des données en Europe, consultez le modèle de responsabilitéAWS partagée et le billet de GDPR blog sur le blog sur la AWS sécurité.

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity

Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) pour chaque compte.
- UtilisezSSL/TLSpour communiquer avec les AWS ressources. Nous avons besoin de la TLS version 1.2 et recommandons la TLS version 1.3.
- Configuration API et journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de FIPS 140 à 3 modules cryptographiques validés pour accéder AWS via une interface de ligne de commande ou unAPI, utilisez un point de terminaison. FIPS Pour plus d'informations sur les FIPS points de terminaison disponibles, voir Federal Information Processing Standard (FIPS) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Timestream pour InfluxDB ou autre à Services AWS l'aide de la console,, API ou. AWS CLI AWS SDKs Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez un URL à un serveur externe, nous vous recommandons vivement de ne pas inclure d'informations d'identification dans le URL afin de valider votre demande auprès de ce serveur.

Pour des informations plus détaillées sur les sujets relatifs à la protection des données de Timestream for InfluxDB, tels que le chiffrement au repos et la gestion des clés, sélectionnez l'un des sujets disponibles ci-dessous.

Rubriques

- Chiffrement au repos
- Chiffrement en transit

Chiffrement au repos

Timestream for InfluxDB Encryption at rest fournit une sécurité renforcée en cryptant toutes vos données au repos à l'aide des clés de chiffrement stockées dans ().AWS Key Management ServiceAWS KMS Cette fonctionnalité réduit la lourdeur opérationnelle et la complexité induites par la protection des données sensibles. Le chiffrement au repos vous permet de créer des applications sensibles en matière de sécurité qui sont conformes aux exigences réglementaires et de chiffrement strictes.

- Le chiffrement est activé par défaut sur votre instance de base de données Timestream for InfluxDB et ne peut pas être désactivé. L'algorithme de chiffrement standard AES -256 est l'algorithme de chiffrement par défaut utilisé.
- AWS KMS est utilisé pour le chiffrement au repos dans Timestream pour InfluxDB.
- Il n'est pas nécessaire de modifier les applications clientes de votre instance de base de données pour utiliser le chiffrement.

Chiffrement en transit

Toutes vos données Timestream pour InfluxDB sont cryptées en transit. Par défaut, toutes les communications à destination et en provenance de Timestream pour InfluxDB sont protégées à l'aide du cryptage Transport Layer Security (). TLS

Le trafic à destination et en provenance d'Amazon Timestream pour InfluxDB est sécurisé à l'aide TLS des versions 1.2 ou 1.3 prises en charge.

Identity and Access Management pour Amazon Timestream pour InfluxDB

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. IAMles administrateurs contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser Timestream pour les ressources InfluxDB. IAMest un Service AWS ventilateur que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Authentification par des identités
- Gestion des accès à l’aide de politiques

- Comment fonctionne Amazon Timestream pour InfluxDB avec IAM
- Exemples de politiques basées sur l'identité pour Amazon Timestream pour InfluxDB
- Résolution des problèmes liés à l'identité et à l'accès à Amazon Timestream pour InfluxDB
- Contrôle de l'accès à une instance de base de données dans un VPC
- Utilisation de rôles liés à un service pour Amazon Timestream pour InfluxDB
- AWS politiques gérées pour Amazon Timestream pour InfluxDB
- Connexion à Timestream pour InfluxDB via un point de terminaison VPC

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant que Utilisateur racine d'un compte AWS, en tant qu'IAMutilisateur ou en assumant un IAM rôle.

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAMIdentity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez en tant qu'identité fédérée, votre administrateur a préalablement configuré la fédération d'identité à l'aide de IAM rôles. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la version 4 de AWS Signature pour les API demandes dans le guide de IAM l'utilisateur.

Quelle que soit la méthode d’authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir

plus, voir Authentification multifactorielle dans le guide de l'AWS IAM Identity Center utilisateur et Authentification AWS multifactorielle IAM dans le guide de l'IAMutilisateur.

Utilisateurs et groupes IAM

Un IAMutilisateur est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des IAM utilisateurs dotés d'informations d'identification à long terme, telles que des mots de passe et des clés d'accès. Toutefois, si vous avez des cas d'utilisation spécifiques qui nécessitent des informations d'identification à long terme auprès des IAM utilisateurs, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, voir Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification à long terme dans le Guide de IAM l'utilisateur.

Un IAMgroupe est une identité qui définit un ensemble d'IAMutilisateurs. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d’utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer IAM des ressources.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu’un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d’informations d’identification permanentes, mais les rôles fournissent des informations d’identification temporaires. Pour en savoir plus, consultez la section Cas d'utilisation pour IAM les utilisateurs dans le Guide de IAM l'utilisateur.

IAMrôles

Un IAMrôle est une identité au sein de Compte AWS vous dotée d'autorisations spécifiques. Il est similaire à un IAM utilisateur, mais n'est pas associé à une personne en particulier. Pour assumer temporairement un IAM rôle dans le AWS Management Console, vous pouvez passer d'un rôle d'utilisateur à un IAM rôle (console). Vous pouvez assumer un rôle en appelant une AWS API opération AWS CLI or ou en utilisant une option personnaliséeURL. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez la section Méthodes pour assumer un rôle dans le Guide de IAM l'utilisateur.

IAMles rôles dotés d'informations d'identification temporaires sont utiles dans les situations suivantes :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour plus d'informations sur les rôles pour la fédération, voir Création d'un rôle pour un fournisseur d'identité tiers dans le guide de IAM l'utilisateur. Si vous utilisez IAM Identity Center, vous configurez un ensemble d'autorisations. Pour contrôler les accès auxquels vos identités peuvent accéder après leur authentification, IAM Identity Center met en corrélation l'ensemble d'autorisations avec un rôle dans. IAM Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations IAM utilisateur temporaires : un IAM utilisateur ou un rôle peut assumer un IAM rôle afin d'obtenir temporairement différentes autorisations pour une tâche spécifique.
- Accès entre comptes : vous pouvez utiliser un IAM rôle pour autoriser une personne (un mandant fiable) d'un autre compte à accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour connaître la différence entre les rôles et les politiques basées sur les ressources pour l'accès entre comptes, voir Accès aux ressources entre comptes IAM dans le guide de l'IAMutilisateur.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès transmises (FAS) — Lorsque vous utilisez un IAM utilisateur ou un rôle pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FASutilise les autorisations du principal appelant an Service AWS, combinées à la demande Service AWS pour adresser des demandes aux services en aval. FASles demandes ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur les politiques relatives FAS aux demandes, consultez la section Transférer les sessions d'accès.
  - Rôle de service — Un rôle de service est un IAMrôle qu'un service assume pour effectuer des actions en votre nom. Un IAM administrateur peut créer, modifier et supprimer un rôle de service de l'intérieurIAM. Pour plus d'informations, consultez la section Création d'un rôle auquel déléguer des autorisations Service AWS dans le Guide de IAM l'utilisateur.

  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un IAM administrateur peut consulter, mais pas modifier les autorisations pour les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un IAM rôle pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui font AWS CLI des AWS API demandes. Cela est préférable au stockage des clés d'accès dans l'EC2instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l'EC2instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez la section Utilisation d'un IAM rôle pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le Guide de IAM l'utilisateur.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de JSON documents. Pour plus d'informations sur la structure et le contenu des documents de JSON politique, voir Présentation des JSON politiques dans le guide de IAM l'utilisateur.

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour autoriser les utilisateurs à effectuer des actions sur les ressources dont ils ont besoin, un IAM administrateur peut créer des IAM politiques. L'administrateur peut ensuite ajouter les IAM politiques aux rôles, et les utilisateurs peuvent assumer les rôles.

IAMles politiques définissent les autorisations pour une action, quelle que soit la méthode que vous utilisez pour effectuer l'opération. Par exemple, supposons que vous disposiez d'une politique qui

autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle auprès du AWS Management Console AWS CLI, ou du AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont JSON des documents de politique d'autorisation que vous pouvez joindre à une identité, telle qu'un IAM utilisateur, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour savoir comment créer une politique basée sur l'identité, voir Définir des IAM autorisations personnalisées avec des politiques gérées par le client dans le Guide de l'IAMutilisateur.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour savoir comment choisir entre une politique gérée ou une politique intégrée, voir Choisir entre les politiques gérées et les politiques intégrées dans le Guide de l'IAMutilisateur.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents JSON de stratégie que vous attachez à une ressource. Les politiques de confiance dans les IAM rôles et les politiques relatives aux compartiments Amazon S3 sont des exemples de politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser de politiques AWS gérées depuis une IAM stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLssont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format du document JSON de stratégie.

Amazon S3 et Amazon VPC sont des exemples de services compatiblesACLs. AWS WAF Pour en savoir plusACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limites d'autorisations — Une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une IAM entité (IAMutilisateur ou rôle). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez la section Limites d'autorisations pour les IAM entités dans le Guide de IAM l'utilisateur.
- Politiques de contrôle des services (SCPs) : SCPs JSON politiques qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Les SCP limites d'autorisations pour les entités présentes dans les comptes des membres, y compris chacune d'entre elles Utilisateur racine d'un compte AWS. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez la section Politiques de session dans le guide de IAM l'utilisateur.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de IAM l'utilisateur.

Comment fonctionne Amazon Timestream pour InfluxDB avec IAM

IAMfonctionnalités que vous pouvez utiliser avec Amazon Timestream pour InfluxDB

IAMfonctionnalité	Timestream pour le support d'InfluxDB
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition d'une politique	Non
ACLs	Non
ABAC(balises dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Fonctions du service	Non
Rôles liés à un service	Oui

Pour obtenir une vue d'ensemble de la façon dont Timestream for InfluxDB et les autres AWS services fonctionnent avec la plupart des IAM fonctionnalités, consultez les AWS services compatibles IAM dans le guide de l'utilisateur. IAM

Politiques basées sur l'identité pour Timestream for InfluxDB

Prend en charge les politiques basées sur l’identité : oui

Les politiques basées sur l'identité sont JSON des documents de politique d'autorisation que vous pouvez joindre à une identité, telle qu'un IAM utilisateur, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour savoir comment créer une politique basée sur l'identité, voir Définir des IAM autorisations personnalisées avec des politiques gérées par le client dans le Guide de l'IAMutilisateur.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier les actions et les ressources autorisées ou refusées ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité, car celle-ci s’applique à l’utilisateur ou au rôle auquel elle est attachée. Pour en savoir plus sur tous les éléments que vous pouvez utiliser dans une JSON politique, consultez la référence aux éléments de IAM JSON politique dans le Guide de IAM l'utilisateur.

Exemples de politiques basées sur l'identité pour Timestream pour InfluxDB

Pour voir des exemples de politiques basées sur l'identité Timestream pour InfluxDB, voir.. Exemples de politiques basées sur l'identité pour Amazon Timestream pour InfluxDB

Politiques basées sur les ressources dans Timestream pour InfluxDB

Prend en charge les politiques basées sur les ressources : non

Les politiques basées sur les ressources sont des documents JSON de stratégie que vous attachez à une ressource. Les politiques de confiance dans les IAM rôles et les politiques relatives aux compartiments Amazon S3 sont des exemples de politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l’accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d’actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour activer l'accès entre comptes, vous pouvez spécifier un compte entier ou IAM des entités d'un autre compte comme principal dans une politique basée sur les ressources. L’ajout d’un principal entre comptes à une politique basée sur les ressources ne représente qu’une partie de l’instauration

de la relation d'approbation. Lorsque le principal et la ressource sont différents Comptes AWS, un IAM administrateur du compte de confiance doit également accorder à l'entité principale (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, voir Accès aux ressources entre comptes IAM dans le Guide de IAM l'utilisateur.

Actions politiques pour Timestream for InfluxDB

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'`Action`élément d'une JSON politique décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès dans une politique. Les actions de stratégie portent généralement le même nom que l' AWS APIopération associée. Il existe certaines exceptions, telles que les actions avec autorisation uniquement qui n'ont pas d'opération correspondante. API Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des actions Timestream pour InfluxDB, consultez la section Actions définies par Amazon Timestream pour InfluxDB dans le Service Authorization Reference.

Les actions de politique dans Timestream pour InfluxDB utilisent le préfixe suivant avant l'action :

```
timestream-influxdb
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "timestream-influxdb:action1",
      "timestream-influxdb:action2"
         ]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "timestream-influxdb:Describe*"
```

Ressources politiques pour Timestream pour InfluxDB

Prend en charge les ressources de politique : oui

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Resource` JSON de stratégie indique le ou les objets auxquels s'applique l'action. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de spécifier une ressource en utilisant son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

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

Pour consulter la liste des types de ressources Timestream pour InfluxDB et leurs caractéristiquesARNs, consultez la section Ressources définies par Amazon Timestream pour InfluxDB dans la référence d'autorisation de service. Pour savoir avec quelles actions vous pouvez spécifier pour chaque ressource, consultez Actions définies par Amazon Timestream pour InfluxDB. ARN

Clés de condition de politique pour Timestream pour InfluxDB

Prend en charge les clés de condition de politique spécifiques au service : Non

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez autoriser un IAM utilisateur à accéder à une ressource uniquement si celle-ci est étiquetée avec son nom IAM d'utilisateur. Pour plus d'informations, consultez IAMla section Éléments de politique : variables et balises dans le Guide de IAM l'utilisateur.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés contextuelles de condition AWS globales dans le guide de IAM l'utilisateur.

Listes de contrôle d'accès (ACLs) dans Timestream pour InfluxDB

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLssont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format du document JSON de stratégie.

Contrôle d'accès basé sur les attributs (ABAC) avec Timestream pour InfluxDB

Supports ABAC (balises dans les politiques) : Oui

Le contrôle d'accès basé sur les attributs (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises à IAM des entités (utilisateurs ou rôles) et à de nombreuses AWS ressources. Le balisage des entités et des ressources est la première étape deABAC. Vous concevez ensuite des ABAC politiques pour autoriser les opérations lorsque le tag du principal correspond à celui de la ressource à laquelle il essaie d'accéder.

ABACest utile dans les environnements qui se développent rapidement et aide dans les situations où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informationsABAC, voir Définir des autorisations avec ABAC autorisation dans le Guide de IAM l'utilisateur. Pour consulter un didacticiel présentant les étapes de configurationABAC, voir Utiliser le contrôle d'accès basé sur les attributs (ABAC) dans le guide de l'IAMutilisateur.

Utilisation d'informations d'identification temporaires avec Timestream pour InfluxDB

Prend en charge les informations d'identification temporaires : oui

Certains Services AWS ne fonctionnent pas lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation IAM dans le Guide de IAM l'utilisateur.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d'identification temporaires lorsque vous vous connectez à la console en tant qu'utilisateur, puis changez de rôle. Pour plus d'informations sur le changement de rôle, voir Passer d'un utilisateur à un IAM rôle (console) dans le guide de IAM l'utilisateur.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide du AWS CLI ou AWS API. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez la section Informations d'identification de sécurité temporaires dans IAM.

Autorisations principales interservices pour Timestream pour InfluxDB

Prend en charge les sessions d'accès transféré (FAS) : Oui

Lorsque vous utilisez un IAM utilisateur ou un rôle pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FASutilise les autorisations du principal appelant an Service AWS, combinées à la demande Service AWS pour adresser des demandes aux services en aval. FASles demandes ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur les politiques relatives FAS aux demandes, consultez la section Transférer les sessions d'accès.

Rôles de service pour Timestream pour InfluxDB

Supporte les rôles de service : Non

Un rôle de service est un IAMrôle qu'un service assume pour effectuer des actions en votre nom. Un IAM administrateur peut créer, modifier et supprimer un rôle de service de l'intérieurIAM. Pour plus d'informations, consultez la section Création d'un rôle auquel déléguer des autorisations Service AWS dans le Guide de IAM l'utilisateur.

> **Warning**
>
> La modification des autorisations pour un rôle de service peut interrompre la fonctionnalité Timestream for InfluxDB. Modifiez les rôles de service uniquement lorsque Timestream for InfluxDB fournit des instructions à cet effet.

Rôles liés à un service pour Timestream for InfluxDB

Prend en charge les rôles liés aux services : Oui

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un IAM administrateur peut consulter, mais pas modifier les autorisations pour les rôles liés à un service.

Pour plus de détails sur la création ou la gestion des rôles liés à un service, consultez la section AWS Services compatibles avec. IAM Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur l'identité pour Amazon Timestream pour InfluxDB

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier Timestream pour les ressources InfluxDB. Ils ne peuvent pas non plus effectuer de tâches en utilisant le AWS Management Console, AWS Command Line Interface (AWS CLI) ou AWS API. Pour autoriser les utilisateurs à effectuer des actions sur les ressources dont ils ont besoin, un IAM administrateur peut créer des IAM politiques. L'administrateur peut ensuite ajouter les IAM politiques aux rôles, et les utilisateurs peuvent assumer les rôles.

Pour savoir comment créer une politique IAM basée sur l'identité à l'aide de ces exemples de documents de JSON stratégie, voir Créer des IAM politiques (console) dans le guide de l'IAMutilisateur.

Pour plus de détails sur les actions et les types de ressources définis par Timestream pour InfluxDB, y compris le format du ARNs pour chacun des types de ressources, voir Actions, ressources et clés de condition pour Amazon Timestream pour InfluxDB dans la référence d'autorisation de service.

Rubriques

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Timestream pour InfluxDB dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à

vos cas d'utilisation. Pour plus d'informations, consultez les politiques AWS gérées ou les politiques AWS gérées pour les fonctions professionnelles dans le Guide de IAM l'utilisateur.

- Appliquer les autorisations du moindre privilège : lorsque vous définissez des autorisations à IAM l'aide de politiques, accordez uniquement les autorisations nécessaires à l'exécution d'une tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation IAM pour appliquer des autorisations, consultez la section Politiques et autorisations du Guide de IAM l'utilisateur. IAM
- Utilisez des conditions dans IAM les politiques pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques pour limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez rédiger une condition de politique pour spécifier que toutes les demandes doivent être envoyées en utilisantSSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, voir Éléments IAM JSON de politique : Condition dans le guide de IAM l'utilisateur.
- Utilisez IAM Access Analyzer pour valider vos IAM politiques afin de garantir des autorisations sécurisées et fonctionnelles. IAM Access Analyzer valide les politiques nouvelles et existantes afin qu'elles soient conformes au langage des IAM politiques (JSON) et IAM aux meilleures pratiques. IAMAccess Analyzer fournit plus de 100 vérifications des politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez la section Valider les politiques avec IAM Access Analyzer dans le guide de l'IAMutilisateur.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des IAM utilisateurs ou un utilisateur root Compte AWS, activez-le MFA pour une sécurité supplémentaire. Pour exiger le MFA moment où les API opérations sont appelées, ajoutez MFA des conditions à vos politiques. Pour plus d'informations, consultez la section APIAccès sécurisé avec MFA dans le guide de IAM l'utilisateur.

Pour plus d'informations sur les meilleures pratiques en matière de sécuritéIAM, consultez la section Bonnes pratiques en matière de sécurité IAM dans le Guide de IAM l'utilisateur.

Utilisation de la console Timestream pour InfluxDB

Pour accéder à la console Amazon Timestream for InfluxDB, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails du flux temporel des ressources InfluxDB dans votre. Compte AWS Si vous créez une

politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui passent des appels uniquement vers le AWS CLI ou le AWS API. Au lieu de cela, autorisez uniquement l'accès aux actions correspondant à l'APIopération qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la console Timestream for InfluxDB, attachez également le Timestream for `ConsoleAccess` InfluxDB ou la politique gérée aux entités. `ReadOnly` AWS Pour plus d'informations, consultez la section Ajouter des autorisations à un utilisateur dans le Guide de IAM l'utilisateur.

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux IAM utilisateurs de consulter les politiques intégrées et gérées associées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide du 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": "*"
        }
    ]
}
```

Accès à un compartiment Amazon S3

Dans cet exemple, vous souhaitez accorder à un IAM utilisateur de votre AWS compte l'accès à l'un de vos compartiments Amazon S3. `examplebucket` Vous souhaitez également autoriser l'utilisateur à ajouter, mettre à jour et supprimer des objets.

En plus de l'octroi des autorisations `s3:PutObject`, `s3:GetObject` et `s3:DeleteObject` à l'utilisateur, la stratégie octroie aussi les autorisations `s3:ListAllMyBuckets`, `s3:GetBucketLocation` et `s3:ListBucket`. Ces conditions supplémentaires sont requises par la console. De la même manière, les actions `s3:PutObjectAcl` et `s3:GetObjectAcl` sont nécessaires pour que les objets puissent être copiés, coupés et collés dans la console. Pour un exemple de procédure pas à pas qui accorde des autorisations aux utilisateurs et les teste à l'aide de la console, consultez Un exemple de procédure pas à pas : utilisation de politiques utilisateur pour contrôler l'accès à votre compartiment.

```
{
   "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/*"
      }
   ]
}
```

Autoriser toutes les opérations

Voici un exemple de politique qui autorise toutes les opérations dans Timestream pour InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream-influxdb:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Création, description, suppression et mise à jour d'une instance de base de données

L'exemple de politique suivant permet à un utilisateur de créer, de décrire, de supprimer et de mettre à jour une instance de base de données `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"
        }
    ]
}
```

Résolution des problèmes liés à l'identité et à l'accès à Amazon Timestream pour InfluxDB

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pourriez rencontrer lorsque vous travaillez avec Timestream pour InfluxDB et. IAM

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Timestream pour InfluxDB
- Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Timestream pour InfluxDB

Je ne suis pas autorisé à effectuer une action dans Timestream pour InfluxDB

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

L'exemple d'erreur suivant se produit quand l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `timestream-influxdb:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream-influxdb:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *my-example-widget* à l'aide de l'action `timestream-influxdb:GetWidget`.

Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Timestream pour InfluxDB

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Contrôle de l'accès à une instance de base de données dans un VPC
- Pour savoir si Timestream pour InfluxDB prend en charge ces fonctionnalités, consultez Comment fonctionne Amazon Timestream pour InfluxDB. IAM
- Pour savoir comment donner accès à vos ressources sur les AWS comptes que vous possédez, consultez la section Fournir l'accès à un IAM utilisateur sur un autre AWS compte que vous possédez dans le Guide de IAM l'utilisateur.
- Pour savoir comment fournir l'accès à vos ressources à des AWS comptes tiers, consultez la section Fournir un accès aux AWS comptes détenus par des tiers dans le Guide de IAM l'utilisateur.
- Pour savoir comment fournir un accès via la fédération d'identité, consultez la section Fournir un accès aux utilisateurs authentifiés de manière externe (fédération d'identité) dans le guide de l'IAMutilisateur.
- Pour connaître la différence entre l'utilisation de rôles et de politiques basées sur les ressources pour l'accès entre comptes, consultez la section En quoi les IAM rôles diffèrent des politiques basées sur les ressources dans le Guide de l'utilisateur. IAM

Contrôle de l'accès à une instance de base de données dans un VPC

À l'aide d'Amazon Virtual Private Cloud (AmazonVPC), vous pouvez lancer AWS des ressources, telles qu'Amazon Timestream pour les instances de base de données InfluxDB, dans un cloud privé virtuel (). VPC Lorsque vous utilisez AmazonVPC, vous contrôlez votre environnement réseau

virtuel. Vous pouvez choisir votre propre plage d'adresses IP, créer des sous-réseaux et configurer le routage et les listes de contrôle d'accès.

Un groupe VPC de sécurité contrôle l'accès aux instances de base de données au sein d'unVPC. Chaque règle VPC de groupe de sécurité permet à une source spécifique d'accéder à une instance de base de données associée à ce groupe VPC de sécurité. VPC La source peut être une plage d'adresses (par exemple, 203.0.113.0/24) ou un autre groupe de sécurité. VPC En spécifiant un groupe VPC de sécurité comme source, vous autorisez le trafic entrant en provenance de toutes les instances (généralement des serveurs d'applications) qui utilisent le groupe VPC de sécurité source. Avant de tenter de vous connecter à votre instance de base de données, configurez-la en VPC fonction de votre cas d'utilisation. Les scénarios suivants sont courants pour accéder à une instance de base de données dans un VPC :

Une instance de base de données dans une EC2 instance VPC accessible par Amazon dans le même VPC

> Une utilisation courante d'une instance de base de données dans un VPC est de partager des données avec un serveur d'applications qui s'exécute dans une EC2 instance de ce mêmeVPC. L'EC2instance peut exécuter un serveur Web avec une application qui interagit avec l'instance de base de données.

Une instance de base de données dans un VPC accès par une EC2 instance dans un autre VPC

> Dans certains cas, votre instance de base de données se trouve dans une EC2 instance VPC différente de celle que vous utilisez pour y accéder. Si tel est le cas, vous pouvez utiliser le VPC peering pour accéder à l'instance de base de données.

Une instance de base de données dans une application cliente VPC accessible via Internet

> Pour accéder à une instance de base de données VPC depuis une application cliente via Internet, vous configurez une VPC avec un seul sous-réseau public et utilisez les sous-réseaux publics pour créer l'instance de base de données. Vous configurez également une passerelle Internet dans le VPC pour permettre la communication sur Internet. Pour se connecter à une instance de base de données depuis l'extérieur de celle-ciVPC, l'instance de base de données doit être accessible au public. En outre, l'accès doit être accordé en utilisant les règles entrantes du groupe de sécurité de l'instance de base de données, et d'autres exigences doivent être satisfaites.

Pour plus d'informations sur les groupes VPC de sécurité, consultez la section Groupes de sécurité dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.

Pour plus de détails sur la façon de se connecter à une instance de base de données Timestream pour InfluxDB, consultez. Connexion à une instance de base de données Amazon Timestream pour InfluxDB

Scénario de groupes de sécurité

Une utilisation courante d'une instance de base de données dans un VPC est de partager des données avec un serveur d'applications exécuté dans une EC2 instance Amazon de la même instanceVPC, à laquelle accède une application cliente extérieure àVPC. Dans ce scénario, vous utilisez le Timestream pour InfluxDB et les VPC pages du AWS Management Console ou le Timestream pour InfluxDB, ainsi que les EC2 API opérations pour créer les instances et les groupes de sécurité nécessaires :

1. Créez un groupe VPC de sécurité (par exemple`sg-0123ec2example`) et définissez des règles entrantes qui utilisent les adresses IP de l'application cliente comme source. Ce groupe de sécurité permet à votre application cliente de se connecter aux EC2 instances d'une instance VPC qui utilise ce groupe de sécurité.
2. Créez une EC2 instance pour l'application et EC2 ajoutez-la au groupe VPC de sécurité (`sg-0123ec2example`) que vous avez créé à l'étape précédente.
3. Créez un deuxième groupe VPC de sécurité (par exemple,`sg-6789rdsexample`) et créez une nouvelle règle en spécifiant le groupe de VPC sécurité que vous avez créé à l'étape 1 (`sg-0123ec2example`) comme source.
4. Créez une nouvelle instance de base de données et ajoutez-la au groupe VPC de sécurité (`sg-6789rdsexample`) que vous avez créé à l'étape précédente. Lorsque vous créez la base de données, utilisez le même numéro de port que celui spécifié pour la règle VPC de groupe de sécurité (`sg-6789rdsexample`) que vous avez créée à l'étape 3.

Création d'un groupe VPC de sécurité

Vous pouvez créer un groupe VPC de sécurité pour une instance de base de données à l'aide de la VPC console. Pour plus d'informations sur la création d'un groupe de sécurité, consultez la section Groupes de sécurité dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.

Association d'un groupe de sécurité à une instance de base de données

Vous pouvez associer un groupe de sécurité à une instance de base de données en utilisant Update sur la console Timestream pour InfluxDB, le `UpdateDBInstance` Timestream pour InfluxDB ou la commande. API `update-db-instance` AWS CLI

L'CLIexemple suivant associe un groupe de VPC sécurité spécifique et supprime les groupes de sécurité de base de données de l'instance de base de données.

```
aws timestream-influxdb update-db-instance --identifier dbName --vpc-security-group-
ids sg-ID
```

Pour savoir comment modifier une instance de base de données, consultez Mise à jour des instances de base de données.

Utilisation de rôles liés à un service pour Amazon Timestream pour InfluxDB

Amazon Timestream pour InfluxDB utilise () des rôles liés AWS Identity and Access Management à un service. IAM Un rôle lié à un service est un type unique de IAM rôle directement lié à un AWS service, tel qu'Amazon Timestream pour InfluxDB. Les rôles liés au service Amazon Timestream pour InfluxDB sont prédéfinis par Amazon Timestream pour InfluxDB. Elles incluent toutes les autorisations dont le service a besoin pour appeler des AWS services au nom de vos instances de base de données.

Un rôle lié à un service facilite la configuration d'Amazon Timestream pour InfluxDB, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Les rôles existent déjà dans votre AWS compte mais sont liés à Amazon Timestream pour les cas d'utilisation d'InfluxDB et disposent d'autorisations prédéfinies. Seul Amazon Timestream pour InfluxDB peut assumer ces rôles, et seuls ces rôles peuvent utiliser la politique d'autorisation prédéfinie. Vous pouvez supprimer les rôles uniquement après la suppression préalable de leurs ressources connexes. Cela protège votre Amazon Timestream pour les ressources InfluxDB, car vous ne pouvez pas supprimer par inadvertance les autorisations nécessaires pour accéder aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez la section AWS Services compatibles avec IAM et recherchez les services dont la valeur est Oui dans la colonne Rôle lié au service. Choisissez un Oui ayant un lien permettant de consulter les détails du rôle pour ce service.

Table des matières

Autorisations de rôle liées à un service pour Amazon Timestream pour InfluxDB

Amazon Timestream for InfluxDB utilise le rôle lié à un service AmazonTimestreamInfluxDBServiceRolePolicynommé — Cette politique permet à Timestream for InfluxDB de gérer les ressources en votre nom selon les besoins de la gestion de AWS vos clusters.

La politique d'autorisation des rôles AmazonTimestreamInflux DBServiceRolePolicy liés au service permet à Amazon Timestream pour InfluxDB d'effectuer les actions suivantes sur les ressources spécifiées :

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

Pour autoriser une IAM entité à créer des rôles liés à un AmazonTimestreamInflux DBServiceRolePolicy service

Ajoutez la déclaration de politique suivante aux autorisations pour cette IAM entité :

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

Pour autoriser une IAM entité à supprimer des rôles liés à un AmazonTimestreamInflux DBServiceRolePolicy service

Ajoutez la déclaration de politique suivante aux autorisations pour cette IAM entité :

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

Vous pouvez également utiliser une politique AWS gérée pour fournir un accès complet à Amazon Timestream pour InfluxDB.

Création d'un rôle lié à un service () IAM

Vous n’avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez une instance de base de données, Amazon Timestream pour InfluxDB crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez une instance

de base de données, Amazon Timestream pour InfluxDB crée à nouveau le rôle lié au service pour vous.

Modification de la description d'un rôle lié à un service pour Amazon Timestream pour InfluxDB

Amazon Timestream pour InfluxDB ne vous permet pas de modifier le rôle lié au service. AmazonTimestreamInflux DBServiceRolePolicy Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Vous pouvez toutefois modifier la description du rôle à l'aide deIAM.

Modification de la description d'un rôle lié à un service (console) IAM

Vous pouvez utiliser la IAM console pour modifier la description d'un rôle lié à un service.

Pour modifier la description d'un rôle lié à un service (console)

1. Dans le volet de navigation gauche de la IAM console, sélectionnez Rôles.
2. Choisissez le nom du rôle à modifier.
3. A l'extrême droite de Description du rôle, choisissez Edit (Modifier).
4. Saisissez une nouvelle description dans la zone et choisissez Save (Enregistrer).

Modification de la description d'un rôle lié à un service () IAM CLI

Vous pouvez utiliser IAM les opérations du AWS Command Line Interface pour modifier la description d'un rôle lié à un service.

Pour modifier la description d'un rôle lié à un service () CLI

1. (Facultatif) Pour afficher la description actuelle d'un rôle, utilisez l'IAMopération AWS CLI for`get-role`.

   Example

   ```
   $ aws iam get-role --role-name AmazonTimestreamInfluxDBServiceRolePolicy
   ```

   Utilisez le nom du rôle, et non leARN, pour faire référence aux rôles associés aux CLI opérations. Par exemple, si un rôle présente les caractéristiques suivantes ARN :`arn:aws:iam::123456789012:role/myrole`, faites référence au rôle en tant que**`myrole`**.

2. Pour mettre à jour la description d'un rôle lié à un service, utilisez l'opération AWS CLI forIAM. update-role-description

   Linux et macOS

   ```
   $ aws iam update-role-description \
       --role-name AmazonTimestreamInfluxDBServiceRolePolicy \
       --description "new description"
   ```

   Windows

   ```
   $ aws iam update-role-description ^
       --role-name AmazonTimestreamInfluxDBServiceRolePolicy ^
       --description "new description"
   ```

Modification de la description d'un rôle lié à un service () IAM API

Vous pouvez utiliser le IAM API pour modifier la description d'un rôle lié à un service.

Pour modifier la description d'un rôle lié à un service () API

1. (Facultatif) Pour afficher la description actuelle d'un rôle, utilisez l'IAMAPIopération GetRole.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. Pour mettre à jour la description d'un rôle, utilisez l'IAMAPIopération UpdateRoleDescription.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
      &Version=2010-05-08
      &Description="New description"
   ```

Supprimer un rôle lié à un service pour Amazon Timestream pour InfluxDB

Si vous n’avez plus besoin d’utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n’avez aucune entité inutilisée qui n’est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer.

Amazon Timestream pour InfluxDB ne supprime pas pour vous le rôle lié au service.

Nettoyage d'un rôle lié à un service

Avant de pouvoir supprimer un rôle lié IAM à un service, vérifiez d'abord qu'aucune ressource (cluster) n'est associée au rôle.

Pour vérifier si le rôle lié à un service possède une session active dans la console IAM

1. Connectez-vous à la IAM console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation gauche de la IAM console, sélectionnez Rôles. Choisissez ensuite le nom (et non la case à cocher) du AmazonTimestreamInflux DBServiceRolePolicy rôle.
3. Sur la page Récapitulatif du rôle sélectionné, choisissez l'onglet Access Advisor.
4. Dans l'onglet Access Advisor, consultez l'activité récente pour le rôle lié à un service.

Supprimer un rôle lié à un service (console) IAM

Vous pouvez utiliser la IAM console pour supprimer un rôle lié à un service.

Pour supprimer un rôle lié à un service (console)

1. Connectez-vous à la IAM console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation gauche de la IAM console, sélectionnez Rôles. Cochez ensuite la case en regard du nom du rôle que vous souhaitez supprimer, sans sélectionner le nom ou la ligne.
3. Pour les actions sur les Rôle en haut de la page, sélectionnez Supprimer.
4. Sur la page de confirmation, passez en revue les données du dernier accès au service, qui indiquent la date à laquelle chacun des rôles sélectionnés a accédé à un AWS service pour la

dernière fois. Cela vous permet de confirmer si le rôle est actif actuellement. Si vous souhaitez continuer, sélectionnez Oui, supprimer pour lancer la tâche de suppression du rôle.

5. Regardez les notifications de la IAM console pour suivre la progression de la suppression des rôles liés au service. La suppression du rôle IAM lié au service étant asynchrone, une fois que vous avez soumis le rôle pour suppression, la tâche de suppression peut réussir ou échouer. Si la tâche échoue, vous pouvez choisir View details (Afficher les détails) ou View Resources (Afficher les ressources) à partir des notifications pour connaître le motif de l'échec de la suppression.

Supprimer un rôle lié à un service () IAM CLI

Vous pouvez utiliser IAM les opérations du AWS Command Line Interface pour supprimer un rôle lié à un service.

Pour supprimer un rôle lié à un service () CLI

1. Si vous ne connaissez pas le nom du rôle lié à un service que vous souhaitez supprimer, saisissez la commande suivante. Cette commande répertorie les rôles et leurs noms de ressources Amazon (ARNs) dans votre compte.

```
$ aws iam get-role --role-name role-name
```

   Utilisez le nom du rôle, et non leARN, pour faire référence aux rôles associés aux CLI opérations. Par exemple, si un rôle possède le ARN`arn:aws:iam::123456789012:role/myrole`, vous l'appelez**`myrole`**.

2. Dans la mesure où un rôle lié à un service ne peut pas être supprimé s'il est utilisé ou si des ressources lui sont associées, vous devez envoyer une demande de suppression. Cette demande peut être refusée si ces conditions ne sont pas satisfaites. Vous devez capturer le `deletion-task-id` de la réponse afin de vérifier l'état de la tâche de suppression. Saisissez la commande suivante pour envoyer une demande de suppression d'un rôle lié à un service.

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. Tapez la commande suivante pour vérifier l'état de la tâche de suppression.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

L'état de la tâche de suppression peut être `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` ou `FAILED`. Si la suppression échoue, l'appel renvoie le motif de l'échec, afin que vous puissiez apporter une solution.

Supprimer un rôle lié à un service () IAM API

Vous pouvez utiliser le IAM API pour supprimer un rôle lié à un service.

Pour supprimer un rôle lié à un service () API

1. Pour soumettre une demande de suppression pour un rôle lié à un service, appelez DeleteServiceLinkedRole. Dans la demande, spécifiez un nom de rôle.

   Dans la mesure où un rôle lié à un service ne peut pas être supprimé s'il est utilisé ou si des ressources lui sont associées, vous devez envoyer une demande de suppression. Cette demande peut être refusée si ces conditions ne sont pas satisfaites. Vous devez capturer le `DeletionTaskId` de la réponse afin de vérifier l'état de la tâche de suppression.
2. Pour vérifier l'état de la suppression, appelez GetServiceLinkedRoleDeletionStatus. Dans la demande, spécifiez le`DeletionTaskId`.

   L'état de la tâche de suppression peut être `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` ou `FAILED`. Si la suppression échoue, l'appel renvoie le motif de l'échec, afin que vous puissiez apporter une solution.

Régions prises en charge pour Amazon Timestream pour les rôles liés au service InfluxDB

Amazon Timestream for InfluxDB prend en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Points de terminaison du service AWS.

AWS politiques gérées pour Amazon Timestream pour InfluxDB

Pour ajouter des autorisations aux utilisateurs, aux groupes et aux rôles, il est plus facile d'utiliser des politiques AWS gérées que de les rédiger vous-même. Il faut du temps et de l'expertise pour créer des politiques gérées par les IAM clients qui fournissent à votre équipe uniquement les autorisations

dont elle a besoin. Pour démarrer rapidement, vous pouvez utiliser nos politiques AWS gérées. Ces politiques couvrent les cas d'utilisation courants et sont disponibles dans votre AWS compte. Pour plus d'informations sur les politiques AWS gérées, voir les politiques AWS gérées dans le Guide de IAM l'utilisateur.

AWS les services maintiennent et mettent à jour les politiques AWS gérées. Vous ne pouvez pas modifier les autorisations dans les politiques AWS gérées. Les services ajoutent occasionnellement des autorisations à une politique gérée par AWS pour prendre en charge de nouvelles fonctionnalités. Ce type de mise à jour affecte toutes les identités (utilisateurs, groupes et rôles) auxquelles la politique est attachée. Les services sont très susceptibles de mettre à jour une politique gérée par AWS quand une nouvelle fonctionnalité est lancée ou quand de nouvelles opérations sont disponibles. Les services ne suppriment pas les autorisations d'une politique AWS gérée. Les mises à jour des politiques n'endommageront donc pas vos autorisations existantes.

En outre, AWS prend en charge les politiques gérées pour les fonctions professionnelles qui couvrent plusieurs services. Par exemple, la politique ReadOnlyAccess AWS gérée fournit un accès en lecture seule à tous les AWS services et ressources. Lorsqu'un service lance une nouvelle fonctionnalité, il AWS ajoute des autorisations en lecture seule pour les nouvelles opérations et ressources. Pour obtenir une liste et une description des politiques relatives aux fonctions de travail, voir les politiques AWS gérées pour les fonctions de travail dans le Guide de IAM l'utilisateur.

AWS politique gérée : AmazonTimestreamInflux DBServiceRolePolicy

Vous ne pouvez pas associer la politique AmazonTimestreamInflux DBServiceRolePolicy AWS gérée aux identités de votre compte. Cette politique fait partie du rôle lié au service AWS TimestreamforInflux de base de données. Ce rôle permet au service de gérer les interfaces réseau et les groupes de sécurité de votre compte.

Timestream for InfluxDB utilise les autorisations de cette politique pour gérer les groupes de EC2 sécurité et les interfaces réseau. Cela est nécessaire pour gérer Timestream pour les instances de base de données InfluxDB.

Pour consulter cette politique dans son JSON format, voir AmazonTimestreamInfluxDBServiceRolePolicy.

AWS-politiques gérées pour Amazon Timestream pour InfluxDB

AWS répond à de nombreux cas d'utilisation courants en fournissant des IAM politiques autonomes créées et administrées par AWS. Les politiques gérées octroient les autorisations requises dans les cas d'utilisation courants et vous évitent d'avoir à réfléchir aux autorisations qui sont requises. Pour plus d'informations, consultez la section Politiques AWS gérées dans le guide de IAM l'utilisateur.

Les politiques AWS gérées suivantes, que vous pouvez associer aux utilisateurs de votre compte, sont spécifiques à Timestream pour InfluxDB :

AmazonTimestreamInfluxDBFullAccess

Vous pouvez associer la `AmazonTimestreamInfluxDBFullAccess` politique à votre IAM identité. Cette politique accorde des autorisations administratives qui permettent un accès complet à toutes les ressources Timestream for InfluxDB.

Vous pouvez également créer vos propres IAM politiques personnalisées pour autoriser Amazon Timestream à effectuer des actions InfluxDB. API Vous pouvez associer ces politiques personnalisées aux IAM utilisateurs ou aux groupes qui ont besoin de ces autorisations.

Pour consulter cette politique dans son JSON format, voir AmazonTimestreamInfluxDBFullAccess.

Timestream pour les mises à jour des politiques gérées par InfluxDB AWS

Consultez les détails des mises à jour des politiques AWS gérées pour Timestream for InfluxDB depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au RSS flux sur la page d'historique du document Timestream for InfluxDB.

Modification	Description	Date
AmazonTimestreamInfluxDBFullAccess – Mise à jour d'une stratégie existante	L'`ec2:DescribeRouteTables` action a été ajoutée à la politique `AmazonTimestreamInfluxDBFul`	10/08/2024

Modification	Description	Date
	`lAccess` gérée existante. Cette action est utilisée pour décrire vos tables de routage.	
AWS politique gérée : AmazonTimestreamInfluxDBServiceRolePolicy : nouvelle politique	Amazon Timestream for InfluxDB a ajouté une nouvelle politique qui permet au service de gérer les interfaces réseau et les groupes de sécurité de votre compte.	14/03/2024
AmazonTimestreamInfluxDBFullAccess : nouvelle politique	Amazon Timestream for InfluxDB a ajouté une nouvelle politique fournissant un accès administratif complet pour créer, mettre à jour, supprimer et répertorier les instances Amazon Timestream InfluxDB ainsi que pour créer et répertorier des groupes de paramètres.	14/03/2024

Connexion à Timestream pour InfluxDB via un point de terminaison VPC

Vous pouvez vous connecter directement à Timestream pour InfluxDB via un point de terminaison d'interface privé dans votre cloud privé virtuel (). VPC Lorsque vous utilisez un point de VPC terminaison d'interface, la communication entre vous VPC et Timestream pour InfluxDB s'effectue entièrement au sein du réseau. AWS

Timestream for InfluxDB prend en charge les points de terminaison Amazon Virtual Private Cloud (AmazonVPC) alimentés par. AWS PrivateLink Chaque VPC point de terminaison est représenté par une ou plusieurs interfaces réseau élastiques (ENIs) avec des adresses IP privées dans vos VPC sous-réseaux.

Le point de VPC terminaison de l'interface vous connecte VPC directement à Timestream pour InfluxDB sans passerelle Internet, NAT appareil, VPN connexion ou connexion. AWS Direct Connect

Les instances de votre navigateur n'ont VPC pas besoin d'adresses IP publiques pour communiquer avec Timestream for InfluxDB.

Régions

Timestream for InfluxDB prend en charge les VPC points de VPC terminaison et les politiques de point de terminaison dans tous les cas où Timestream for InfluxDB est pris Régions AWS en charge.

Rubriques

- Considérations relatives à Timestream pour les points de terminaison InfluxDB VPC
- Création d'un VPC point de terminaison pour Timestream pour InfluxDB
- Connexion à un Timestream pour le point de terminaison InfluxDB VPC
- Contrôle de l'accès à un VPC point de terminaison
- Utilisation d'un VPC point de terminaison dans une déclaration de politique
- Enregistrement de votre VPC terminal

Considérations relatives à Timestream pour les points de terminaison InfluxDB VPC

Avant de configurer un point de VPC terminaison d'interface pour Timestream for InfluxDB, consultez la rubrique Propriétés et limites du point de terminaison d'interface dans le Guide.AWS PrivateLink

Le support Timestream pour InfluxDB pour un VPC point de terminaison inclut les éléments suivants.

- Vous pouvez utiliser votre VPC point de terminaison pour appeler toutes les opérations Timestream for InfluxDB depuis API votre. VPC
- Vous pouvez utiliser AWS CloudTrail les journaux pour auditer votre utilisation de Timestream pour les ressources InfluxDB via le point de terminaison. VPC Pour plus de détails, consultez Enregistrement de votre VPC terminal.

Création d'un VPC point de terminaison pour Timestream pour InfluxDB

Vous pouvez créer un VPC point de terminaison pour Timestream for InfluxDB à l'aide de la console Amazon VPC ou d'Amazon. VPC API Pour plus d’informations, consultez Création d’un point de terminaison d’interface dans le Guide AWS PrivateLink .

- Pour créer un VPC point de terminaison pour Timestream for InfluxDB, utilisez le nom de service suivant :

```
com.amazonaws.region.timestream-influxdb
```

Par exemple, dans la région USA Ouest (Oregon) (`us-west-2`), le nom du service serait :

```
com.amazonaws.us-west-2.timestream-influxdb
```

Pour faciliter l'utilisation du VPC point de terminaison, vous pouvez activer un DNSnom privé pour votre VPC point de terminaison. Si vous sélectionnez l'option Activer le DNS nom, le flux temporel standard pour le nom d'DNShôte InfluxDB est résolu vers votre point de terminaison. VPC Par exemple, `https://timestream-influxdb.us-west-2.amazonaws.com` serait résolu en un point de VPC terminaison connecté au nom du service`com.amazonaws.us-west-2.timestream-influxdb`.

Cette option facilite l'utilisation du VPC point de terminaison. Le AWS SDKs et AWS CLI utilisez le Timestream standard pour le DNS nom d'hôte InfluxDB par défaut, vous n'avez donc pas besoin de spécifier le point de VPC terminaison URL dans les applications et les commandes.

Pour de plus amples informations, veuillez consulter Accès à un service via un point de terminaison d'interface dans le Guide AWS PrivateLink .

Connexion à un Timestream pour le point de terminaison InfluxDB VPC

Vous pouvez vous connecter à Timestream pour InfluxDB via le VPC point de terminaison en utilisant un AWS SDK, le ou. AWS CLI AWS Tools for PowerShell Pour spécifier le VPC point de terminaison, utilisez son DNS nom.

Si vous avez activé les noms d'hôte privés lors de la création de votre VPC point de terminaison, vous n'avez pas besoin de spécifier le VPC point de terminaison URL dans vos CLI commandes ou dans la configuration de l'application. Le Timestream standard pour le nom d'DNShôte InfluxDB est résolu vers votre point de terminaison. VPC Le AWS CLI et SDKs utilisez ce nom d'hôte par défaut, afin que vous puissiez commencer à utiliser le VPC point de terminaison pour vous connecter à un point de terminaison régional Timestream for InfluxDB sans rien modifier dans vos scripts et applications.

Pour utiliser des noms d'hôte privés, les `enableDnsSupport` attributs `enableDnsHostnames` et de votre nom VPC doivent être définis sur. `true` Pour définir ces attributs, utilisez l'ModifyVpcAttributeopération. Pour plus de détails, consultez la section Afficher et mettre à jour vos DNS attributs VPC dans le guide de VPC l'utilisateur Amazon.

Contrôle de l'accès à un VPC point de terminaison

Pour contrôler l'accès à votre VPC point de terminaison pour Timestream for InfluxDB, associez une politique de point de terminaison à votre point de VPC terminaison. VPC La politique de point de terminaison détermine si les principaux peuvent utiliser le VPC point de terminaison pour appeler Timestream pour les opérations InfluxDB sur Timestream pour les ressources InfluxDB.

Vous pouvez créer une politique de point de VPC terminaison lorsque vous créez votre point de terminaison, et vous pouvez modifier la politique de VPC point de terminaison à tout moment. Utilisez la console VPC de gestion ou les ModifyVpcEndpointopérations CreateVpcEndpointor. Vous pouvez également créer et modifier une politique de point de VPC terminaison à l'aide d'un AWS CloudFormation modèle. Pour obtenir de l'aide sur l'utilisation de la console de VPC gestion, consultez les sections Création d'un point de terminaison d'interface et Modification d'un point de terminaison d'interface dans le AWS PrivateLink Guide.

> **Note**
>
> Timestream for InfluxDB prend en charge les politiques relatives aux VPC terminaux à compter de juillet 2020. VPCles points de terminaison Timestream for InfluxDB créés avant cette date ont la politique de point de VPC terminaison par défaut, mais vous pouvez la modifier à tout moment.

Rubriques

- À propos des politiques relatives aux VPC terminaux
- Politique relative aux VPC terminaux par défaut
- Création d'une politique de point de VPC terminaison
- Afficher une politique de point de VPC terminaison

À propos des politiques relatives aux VPC terminaux

Pour qu'une demande Timestream for InfluxDB qui utilise un VPC point de terminaison soit réussie, le principal a besoin d'autorisations provenant de deux sources :

- Une IAMpolitique doit autoriser le principal à appeler l'opération sur la ressource.
- Une politique de VPC point de terminaison doit donner au principal l'autorisation d'utiliser le point de terminaison pour effectuer la demande.

Politique relative aux VPC terminaux par défaut

Chaque VPC point de VPC terminaison possède une politique de point de terminaison, mais vous n'êtes pas obligé de la spécifier. Si vous ne spécifiez pas de politique, la politique de point de terminaison par défaut autorise toutes les opérations effectuées par tous les principaux sur toutes les ressources du point de terminaison.

Cependant, pour les ressources Timestream for InfluxDB, le principal doit également être autorisé à appeler l'opération à partir d'une IAMpolitique. Par conséquent, dans la pratique, la politique par défaut indique que si un principal est autorisé à appeler une opération sur une ressource, il peut également l'appeler en utilisant le point de terminaison.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

Pour permettre aux principaux d'utiliser le VPC point de terminaison uniquement pour un sous-ensemble de leurs opérations autorisées, créez ou mettez à jour la politique du VPC point de terminaison.

Création d'une politique de point de VPC terminaison

Une politique de VPC point de terminaison détermine si un principal est autorisé à utiliser le VPC point de terminaison pour effectuer des opérations sur une ressource. Pour les ressources Timestream for InfluxDB, le principal doit également être autorisé à effectuer les opérations à partir d'une politique, IAM

Chaque déclaration de politique relative aux VPC terminaux nécessite les éléments suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

La déclaration de politique ne précise pas le VPC point de terminaison. Elle s'applique plutôt à tout VPC point de terminaison auquel la politique est attachée. Pour plus d'informations, consultez la section Contrôle de l'accès aux services avec des VPC points de terminaison dans le guide de VPC l'utilisateur Amazon.

AWS CloudTrail enregistre toutes les opérations qui utilisent le VPC point de terminaison.

Afficher une politique de point de VPC terminaison

Pour consulter la politique de VPC point de terminaison d'un point de terminaison, utilisez la console de VPC gestion ou l'DescribeVpcEndpointsopération.

La AWS CLI commande suivante obtient la politique du point de terminaison avec l'ID de point de VPC terminaison spécifié.

Avant d'utiliser cette commande, remplacez l'exemple d'ID de point de terminaison d'exemple par un ID valide provenant de votre compte.

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpc-endpoint-id`].[PolicyDocument]'
--output text
```

Utilisation d'un VPC point de terminaison dans une déclaration de politique

Vous pouvez contrôler l'accès à Timestream pour les ressources et les opérations d'InfluxDB lorsque la demande provient VPC ou utilise un point de terminaison. VPC Pour ce faire, utilisez l'une des clés de condition globales suivantes dans une IAMpolitique.

- Utilisez la clé de `aws:sourceVpce` condition pour accorder ou restreindre l'accès en fonction du VPC point de terminaison.
- Utilisez la clé de `aws:sourceVpc` condition pour accorder ou restreindre l'accès en fonction de VPC celle qui héberge le point de terminaison privé.

> **Note**
>
> Soyez prudent lorsque vous créez des politiques clés et IAM des politiques basées sur votre VPC terminal. Si une déclaration de politique exige que les demandes proviennent d'un point de VPC terminaison VPC ou d'un point de terminaison en particulier, les demandes

provenant de AWS services intégrés qui utilisent une ressource Timestream for InfluxDB en votre nom peuvent échouer.
De plus, la clé de `aws:sourceIP` condition n'est pas efficace lorsque la demande provient d'un point de VPCterminaison Amazon. Pour limiter les demandes à un VPC point de terminaison, utilisez les clés de `aws:sourceVpc` condition `aws:sourceVpce` ou. Pour plus d'informations, consultez la section Gestion des identités et des accès pour les VPC terminaux et les services de point de VPC terminaison dans le AWS PrivateLink Guide.

Vous pouvez utiliser ces clés de condition globales pour contrôler l'accès à des opérations telles CreateDbInstanceque celles qui ne dépendent d'aucune ressource en particulier.

Enregistrement de votre VPC terminal

AWS CloudTrail enregistre toutes les opérations qui utilisent le VPC point de terminaison. Lorsqu'une demande adressée à Timestream pour InfluxDB utilise un VPC point de terminaison, l'ID du point de VPC terminaison apparaît dans l'entrée du AWS CloudTrail journal qui enregistre la demande. Vous pouvez utiliser l'identifiant du point de terminaison pour auditer l'utilisation de votre point de terminaison Timestream pour VPC InfluxDB.

Cependant, vos CloudTrail journaux n'incluent pas les opérations demandées par les principaux sur d'autres comptes ni les demandes de Timestream pour les opérations InfluxDB sur Timestream pour les ressources InfluxDB et les alias sur d'autres comptes. De plus, pour vous protégerVPC, les demandes refusées par une politique de point de VPC terminaison, mais qui auraient autrement été autorisées, ne sont pas enregistrées dans AWS CloudTrail.

Journalisation et surveillance dans Timestream pour InfluxDB

La surveillance est un élément important du maintien de la fiabilité, de la disponibilité et des performances de Timestream pour InfluxDB et vos solutions. AWS Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant. Toutefois, avant de commencer à surveiller Timestream pour InfluxDB, vous devez créer un plan de surveillance qui inclut les réponses aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- À quelle fréquence les ressources doivent-elles être surveillées ?

- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

L'étape suivante consiste à établir une base de référence pour les performances Timestream normales d'InfluxDB dans votre environnement, en mesurant les performances à différents moments et dans différentes conditions de charge. Lorsque vous surveillez Timestream pour InfluxDB, stockez les données de surveillance historiques afin de pouvoir les comparer aux données de performance actuelles, d'identifier les modèles de performances normaux et les anomalies de performance, et de concevoir des méthodes pour résoudre les problèmes.

Pour établir une référence, vous devez, au Moins, superviser les éléments suivants :

- Les erreurs système, de sorte que vous puissiez déterminer si des demandes ont entraîné une erreur.

Rubriques

Outils de surveillance

AWS fournit divers outils que vous pouvez utiliser pour surveiller Timestream pour InfluxDB. Vous pouvez configurer certains outils pour qu'ils effectuent la supervision automatiquement, tandis que d'autres nécessitent une intervention manuelle. Nous vous recommandons d'automatiser le plus possible les tâches de supervision.

Rubriques

Outils de surveillance automatique

Vous pouvez utiliser les outils de surveillance automatique suivants pour regarder Timestream pour InfluxDB et signaler tout problème :

- Amazon CloudWatch Alarms : surveillez une seule métrique sur une période que vous spécifiez et effectuez une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon Simple Notification Service (AmazonSNS) ou à une politique Amazon EC2 Auto Scaling. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier ; l'état doit avoir changé et être maintenu pendant un certain nombre de périodes. Pour de plus amples informations, veuillez consulter Surveillance avec Amazon CloudWatch.

Outils de surveillance manuelle

Un autre élément important de la surveillance de Timestream pour InfluxDB consiste à surveiller manuellement les éléments que les CloudWatch alarmes ne couvrent pas. Le Timestream pour InfluxDB, CloudWatch Trusted Advisor, et d'autres AWS Management Console tableaux de bord fournissent une at-a-glance vue de l'état de votre environnement. AWS

- La page CloudWatch d'accueil affiche les informations suivantes :
  - Alarmes et statuts en cours
  - Graphiques des alarmes et des ressources
  - Statut d'intégrité du service

  En outre, vous pouvez utiliser CloudWatch pour effectuer les opérations suivantes :
  - Créer des tableaux de bord personnalisés pour surveiller les services de votre choix
  - Représenter graphiquement les données de métriques pour résoudre les problèmes et découvrir les tendances
  - Recherchez et parcourez tous les indicateurs de vos AWS ressources
  - Créer et modifier des alarmes pour être informé des problèmes

Enregistrement du flux temporel pour les appels API InfluxDB avec AWS CloudTrail

Timestream for InfluxDB est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Timestream pour InfluxDB. CloudTrail capture les API appels du Data Definition Language (DDL) pour Timestream pour InfluxDB sous forme d'événements. Les appels capturés incluent les appels depuis la console Timestream pour InfluxDB et les appels de code vers le Timestream pour les opérations InfluxDB. API Si vous créez un suivi, vous pouvez activer la diffusion continue des CloudTrail événements vers un bucket

Amazon Simple Storage Service (Amazon S3), y compris les événements pour Timestream for InfluxDB. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents sur la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Timestream pour InfluxDB, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Diffusion chronologique des informations d'InfluxDB dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans Timestream pour InfluxDB, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS . Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris les événements de Timestream pour InfluxDB, créez une trace. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un parcours dans la console, celui-ci s'applique à toutes les AWS régions. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence.

Pour plus d’informations, consultez les rubriques suivantes dans le AWS CloudTrail Guide de l’utilisateur :

- Présentation de la création d’un journal d’activité
- CloudTrail Services et intégrations pris en charge
- Configuration des SNS notifications Amazon pour CloudTrail
- Réception de fichiers CloudTrail journaux provenant de plusieurs régions
- Réception de fichiers CloudTrail journaux provenant de plusieurs comptes
- Journalisation des événements liés aux données

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l’identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou AWS Identity and Access Management (IAM)
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré
- Si la demande a été faite par un autre AWS service

Pour plus d'informations, consultez l'CloudTrail userIdentityélément.

Validation de conformité pour Amazon Timestream pour InfluxDB

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon Timestream pour InfluxDB dans le cadre de plusieurs programmes de conformité. AWS Tel est le cas des éléments suivants :

- GDPR
- HIPAA
- PCI
- SOC

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité : ces guides de déploiement abordent les considérations architecturales et fournissent des étapes pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- Architecture axée sur la HIPAA sécurité et la conformité sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent AWS créer HIPAA des applications éligibles.

> **Note**
>
> Tous ne Services AWS sont pas HIPAA éligibles. Pour plus d'informations, consultez la référence des services HIPAA éligibles.

- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et reprennent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité PCIDSS, par exemple en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Résilience dans Amazon Timestream pour InfluxDB

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones

de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Amazon Timestream for InfluxDB effectue régulièrement des sauvegardes internes et les conserve pendant 24 heures pour garantir la disponibilité et la durabilité. Les instantanés sont pris lors des suppressions et conservés pendant 30 jours pour permettre les restaurations. Pour y accéder ou les utiliser, déposez un ticket auprès du AWS support.

Vous pouvez créer votre instance avec des fonctionnalités de restauration multi-AZ. Pour plus d'informations, voir Déploiements d'instances de bases de données multi-AZ.

Sécurité de l'infrastructure dans Amazon Timestream pour InfluxDB

En tant que service géré, Amazon Timestream pour InfluxDB est protégé par les procédures de sécurité AWS du réseau mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des API appels de plan de contrôle AWS publiés pour accéder à Timestream pour InfluxDB via le réseau. Pour plus d'informations, voir Plans de contrôle et plans de données. Les clients doivent prendre en charge Transport Layer Security (TLS) 1.2 ou version ultérieure. Nous recommandons la TLS version 1.2 ou 1.3. Les clients doivent également prendre en charge les suites de chiffrement parfaitement confidentielles (), telles que Ephemeral Diffie-Hellman (PFS) ou Elliptic Curve Ephemeral Diffie-Hellman (DHE). ECDHE La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un identifiant de clé d'accès et d'une clé d'accès secrète associés à un IAM principal. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Timestream for InfluxDB est conçu de telle sorte que votre trafic soit isolé dans la AWS région spécifique dans laquelle réside votre instance Timestream for InfluxDB.

Groupes de sécurité

Les groupes de sécurité contrôlent l'accès dont dispose le trafic entrant et sortant d'une instance de base de données. Par défaut, l'accès au réseau est désactivé sur une instance de base de données. Vous pouvez spécifier des règles dans un groupe de sécurité qui autorisent l'accès depuis une plage d'adresses IP, un port ou un groupe de sécurité. Une fois les règles de trafic entrant configurées, les mêmes règles s'appliquent à toutes les instances de base de données qui sont associées à ce groupe de sécurité.

Pour de plus amples informations, veuillez consulter Contrôle de l'accès à une instance de base de données dans un VPC.

Analyse de configuration et de vulnérabilité dans Timestream pour InfluxDB

La configuration et les contrôles informatiques sont une responsabilité partagée entre vous AWS et vous, notre client. Pour plus d'informations, consultez le modèle de responsabilité AWS partagée. Outre le modèle de responsabilité partagée, les utilisateurs de Timestream for InfluxDB doivent être conscients des points suivants :

- Il est de la responsabilité du client de corriger ses applications clients avec les dépendances côté client pertinentes.
- Les clients devraient envisager des tests de pénétration le cas échéant (voir https://aws.amazon.com/security/tests de pénétration/.)

Réponse aux incidents dans Timestream pour InfluxDB

Les incidents liés au service Amazon Timestream for InfluxDB sont signalés dans le Personal Health Dashboard. Pour en savoir plus sur le tableau de bord, cliquez AWS Health ici.

Timestream for InfluxDB prend en charge l'utilisation de rapports. AWS CloudTrail Pour de plus amples informations, veuillez consulter Enregistrement du flux temporel pour les appels API InfluxDB avec AWS CloudTrail.

Amazon Timestream pour API InfluxDB et points de terminaison d'interface () VPC AWS PrivateLink

Vous pouvez établir une connexion privée entre vous VPC et Amazon Timestream pour les points de terminaison du API plan de contrôle InfluxDB en créant un point de terminaison d'interface. VPC

Les points de terminaison de l'interface sont alimentés par AWS PrivateLink. AWS PrivateLink vous permet d'accéder en privé à Amazon Timestream pour les opérations API InfluxDB sans passerelle InternetNAT, appareil, connexion ou VPN connexion Direct Connect. AWS

Les instances de votre choix VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec Amazon Timestream pour les points de terminaison InfluxDB. API Vos instances n'ont pas non plus besoin d'adresses IP publiques pour utiliser le Timestream disponible pour les opérations API InfluxDB. Le trafic entre vous VPC et Amazon Timestream pour InfluxDB ne quitte pas le réseau Amazon. Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux. Pour plus d'informations sur les interfaces réseau élastiques, consultez la section relative aux interfaces réseau élastiques dans le guide de EC2 l'utilisateur Amazon.

- Pour plus d'informations sur les VPC points de terminaison, consultez Interface VPC endpoints (AWS PrivateLink) dans le guide de VPCl'utilisateur Amazon.
- Pour plus d'informations sur Timestream pour les opérations InfluxDB, voir Timestream pour API les opérations InfluxDB. API

Après avoir créé un point de VPC terminaison d'interface, si vous activez les DNS noms d'hôte privés pour le point de terminaison, c'est le Timestream par défaut pour le point de terminaison InfluxDB (https://timestream-influxb).*Region*.amazonaws.com) se résout sur votre point de terminaison. VPC Si vous n'activez pas les DNS noms d'hôte privés, Amazon VPC fournit un nom de point de DNS terminaison que vous pouvez utiliser au format suivant :

```
VPC_Endpoint_ID.timestream-influxb.Region.vpce.amazonaws.com
```

Pour plus d'informations, consultez Interface VPC Endpoints (AWS PrivateLink) dans le guide de l'VPCutilisateur Amazon. Timestream for InfluxDB permet d'appeler toutes ses API actions dans votre. VPC

> **Note**
>
> DNSLes noms d'hôte privés ne peuvent être activés que pour un seul point de VPC terminaison dans leVPC. Si vous souhaitez créer un point de VPC terminaison supplémentaire, le DNS nom d'hôte privé doit être désactivé pour celui-ci.

Considérations relatives aux VPC terminaux

Avant de configurer un point de VPC terminaison d'interface pour Amazon Timestream pour les points de terminaison API InfluxDB, assurez-vous de consulter les propriétés et les limites du point de terminaison d'interface dans le guide de l'utilisateur Amazon. VPC Toutes les API opérations Timestream pour InfluxDB pertinentes pour la gestion des ressources Amazon Timestream pour InfluxDB sont disponibles auprès de vous. VPC AWS PrivateLink VPCles politiques de point de terminaison sont prises en charge pour Timestream pour les points de terminaison InfluxDBAPI. Par défaut, l'accès complet à Timestream pour les API opérations InfluxDB est autorisé via le point de terminaison. Pour plus d'informations, consultez la section Contrôle de l'accès aux services avec des VPC points de terminaison dans le guide de VPC l'utilisateur Amazon.

Création d'un point de VPC terminaison d'interface pour le Timestream pour InfluxDB API

Vous pouvez créer un VPC point de terminaison pour Amazon Timestream pour API InfluxDB à l'aide de la console Amazon ou duVPC. AWS CLI Pour plus d'informations, consultez la section Création d'un point de terminaison d'interface dans le guide de VPC l'utilisateur Amazon.

Après avoir créé un point de VPC terminaison d'interface, vous pouvez activer les noms d'DNShôtes privés pour le point de terminaison. Lorsque vous le faites, le point de terminaison Amazon Timestream pour InfluxDB par défaut (https://timestream-influxb.*Region*.amazonaws.com) se résout sur votre point de terminaison. VPC Pour plus d'informations, consultez la section Accès à un service via un point de terminaison d'interface dans le guide de VPC l'utilisateur Amazon.

Création d'une politique de VPC point de terminaison pour Amazon Timestream pour InfluxDB API

Vous pouvez associer une politique de point de terminaison à votre VPC point de terminaison qui contrôle l'accès au Timestream pour InfluxDB. API La stratégie spécifie les éléments suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez la section Contrôle de l'accès aux services avec des VPC points de terminaison dans le guide de VPC l'utilisateur Amazon.

Example VPCpolitique de point de terminaison pour les actions Timestream for InfluxDB API

Voici un exemple de politique de point de terminaison pour le Timestream pour InfluxDB. API Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès au Timestream répertorié pour les API actions InfluxDB à tous les principaux sur toutes les ressources.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "timestream-influxb:CreateDbInstance",
   "timestream-influxb:UpdateDbInstance"
  ],
  "Resource": "*"
 }]
}
```

Example VPCpolitique de point de terminaison qui refuse tout accès à partir d'un AWS compte spécifié

La politique de point de VPC terminaison suivante refuse le AWS compte *123456789012* tous les accès aux ressources à l'aide du point de terminaison. La politique autorise toutes les actions provenant d'autres comptes.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
```

```
  ]
}
```

Meilleures pratiques de sécurité pour Timestream pour InfluxDB

Amazon Timestream for InfluxDB fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lorsque vous développez et mettez en œuvre vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Implémentation d'un accès sur la base du moindre privilège

Lorsque vous accordez des autorisations, vous décidez qui obtient quelles autorisations à quel Timestream pour les ressources InfluxDB. Vous activez des actions spécifiques que vous souhaitez autoriser sur ces ressources. Par conséquent, vous devez accorder uniquement les autorisations qui sont requises pour exécuter une tâche. L'implémentation d'un accès sur la base du moindre privilège est fondamentale pour réduire les risques en matière de sécurité et l'impact que pourraient avoir des d'erreurs ou des actes de malveillance.

Utiliser IAM les rôles

Les applications productrices et clientes doivent disposer d'informations d'identification valides pour accéder à Timestream pour les instances de base de données InfluxDB. Vous ne devez pas stocker les AWS informations d'identification directement dans une application cliente ou dans un compartiment Amazon S3. Il s'agit d'autorisations à long terme qui ne font pas automatiquement l'objet d'une rotation et qui pourraient avoir un impact commercial important si elles étaient compromises.

Vous devez plutôt utiliser un IAM rôle pour gérer les informations d'identification temporaires permettant à vos applications productrices et clientes d'accéder à Timestream pour les instances de base de données InfluxDB. Lorsque vous utilisez un rôle, vous n'avez pas à utiliser d'informations d'identification à long terme (par exemple, un nom d'utilisateur et un mot de passe ou des clés d'accès) pour accéder à d'autres ressources.

Pour plus d'informations, consultez les rubriques suivantes du guide de l'IAMutilisateur :

- IAMRôles
- Scénarios courants pour les rôles : utilisateurs, applications et services.

Utilisez des comptes AWS Identity and Access Management (IAM) pour contrôler l'accès à Amazon Timestream pour les opérations API InfluxDB, en particulier les opérations qui créent, modifient ou suppriment les ressources Amazon Timestream pour InfluxDB. Ces ressources incluent les instances de base de données, les groupes de sécurité et les groupes de paramètres.

- Créez un utilisateur individuel pour chaque personne qui gère les ressources Amazon Timestream pour InfluxDB, y compris vous-même. N'utilisez pas les informations d'identification AWS root pour gérer les ressources Amazon Timestream for InfluxDB.
- Accordez à chaque utilisateur un ensemble minimum d'autorisations requises pour exécuter ses tâches.
- Utilisez IAM des groupes pour gérer efficacement les autorisations de plusieurs utilisateurs.
- Procédez à une rotation régulière des informations d'identification IAM.
- Configurez AWS Secrets Manager pour faire automatiquement pivoter les secrets d'Amazon Timestream pour InfluxDB. Pour plus d'informations, consultez Rotation de vos AWS secrets Secrets Manager dans le Guide de l'utilisateur de AWS Secrets Manager. Vous pouvez également récupérer les informations d'identification depuis AWS Secrets Manager par programmation. Pour plus d'informations, consultez la section Récupération de la valeur secrète dans le Guide de l'utilisateur de AWS Secrets Manager.
- Sécurisez votre Timestream pour les API jetons d'afflux InfluxDB en utilisant le. APIjetons

Implémentation d'un chiffrement côté serveur dans des ressources dépendantes

Les données au repos et les données en transit peuvent être cryptées dans Timestream pour InfluxDB. Pour de plus amples informations, veuillez consulter Chiffrement en transit.

CloudTrail À utiliser pour surveiller les API appels

Timestream for InfluxDB est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Timestream pour InfluxDB.

À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Timestream pour InfluxDB, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour de plus amples informations, veuillez consulter the section called “Enregistrement du flux temporel des appels LiveAnalytics API avec AWS CloudTrail”.

Amazon Timestream pour InfluxDB prend en charge les événements du plan de contrôle, mais pas le CloudTrail plan de données. Pour plus d'informations, voir Plans de contrôle et plans de données.

Accessible publiquement

Lorsque vous lancez une instance de base de données dans un cloud privé virtuel (VPC) basé sur le VPC service Amazon, vous pouvez activer ou désactiver l'accessibilité publique pour cette instance de base de données. Pour indiquer si le DNS nom de l'instance de base de données que vous créez correspond à une adresse IP publique, vous utilisez le paramètre d'accessibilité public. En utilisant ce paramètre, vous pouvez indiquer s'il existe un accès public à l'instance de base de données

Si votre instance de base de données se trouve dans un VPC mais n'est pas accessible au public, vous pouvez également utiliser une AWS Site-to-Site VPN connexion ou une connexion AWS Direct Connect pour y accéder depuis un réseau privé.

Si votre instance de base de données est accessible au public, veillez à prendre des mesures pour prévenir ou atténuer les menaces liées au déni de service. Pour plus d'informations, consultez les sections Introduction aux attaques par déni de service et Protection des réseaux.

APIréférence

Pour obtenir la liste complète et les détails d'Amazon Timestream pour InfluxDBAPIs, consultez Amazon Timestream pour InfluxDB. APIs

Pour les codes d'erreur communs à tous les AWS services, consultez la section AWS Support.

Historique du document

Modification	Description	Date
Mise à jour relative à la documentation uniquement	Mise à jour de la rubrique Quotas pour séparer les quotas par défaut des limites du système.	22 octobre 2024
Amazon Timestream prend désormais en charge l'analyse des requêtes	Timestream inclut désormais la prise en charge de la fonctionnalité d'analyse des	22 octobre 2024

	requêtes qui vous aide à optimiser vos requêtes, à améliorer leurs performances et à réduire les coûts.	
Amazon Timestream pour InfluxDB met à jour une politique existante.	Amazon Timestream for InfluxDB a ajouté l'action à `ec2:DescribeRouteTables` la politique gérée `AmazonTimestreamInfluxDBFullAccess` existante pour décrire vos tables de routage	8 octobre 2024
`AmazonTimestreamReadOnlyAccess` — Mise à jour d'une politique existante	Timestream for LiveAnalytics a ajouté l'`DescribeAccountSettings` autorisation à la politique `AmazonTimestreamReadOnlyAccess` gérée pour décrire Compte AWS les paramètres.	3 juin 2024
Amazon Timestream prend actuellement en charge LiveAnalytics les unités de calcul Timestream () TCUs	Amazon Timestream inclut actuellement la prise en charge LiveAnalytics des unités de calcul Timestream TCUs () afin de mesurer la capacité de calcul allouée aux besoins de vos requêtes.	29 avril 2024

Nouvelles politiques ajoutées	Amazon Timestream for InfluxDB a ajouté deux nouvelles politiques : l'une permet au service de gérer les interfaces réseau et les groupes de sécurité de votre compte. Pour plus d'informations, consultez AmazonTimestreamInfluxDBServiceRolePolicy. Un autre qui fournit un accès administratif complet pour créer, mettre à jour, supprimer et répertorier les instances Amazon Timestream InfluxDB, ainsi que pour créer et répertorier des groupes de paramètres. Pour plus d'informations, consultez AmazonTimestreamInfluxDBFullAccess.	14 mars 2024
Amazon Timestream pour InfluxDB est désormais disponible pour tous.	Cette documentation couvre la version initiale d'Amazon Timestream pour InfluxDB.	14 mars 2024

Les événements Amazon Timestream LiveAnalytics for Query sont disponibles dans AWS CloudTrail	Amazon Timestream publie pour l'instant LiveAnalytics les événements relatifs aux données de API requête sur. AWS CloudTrail Les clients peuvent auditer toutes les API demandes de requête effectuées dans leurs AWS comptes et consulter des informations telles que l'IAMutilisateur/le rôle qui a fait la demande, la date à laquelle la demande a été faite, les bases de données et les tables interrogées, ainsi que l'ID de requête de la demande.	12 septembre 2023
Amazon Timestream pour LiveAnalytics UNLOAD	Amazon Timestream UNLOAD prend actuellement en charge l'exportation LiveAnalytics des résultats des requêtes vers S3.	12 mai 2023
Amazon Timestream LiveAnalytics pour la mise à jour d'une politique existante.	Autorisations de chargement par lots ajoutées à une politique gérée.	24 février 2023
Amazon Timestream LiveAnalytics pour le chargement par lots.	Amazon Timestream prend actuellement en charge la fonctionnalité LiveAnalytics de chargement par lots.	24 février 2023
Amazon Timestream LiveAnalytics prend actuellement en charge. AWS Backup	Amazon Timestream LiveAnalytics prend actuellement en charge. AWS Backup	14 décembre 2022

Amazon Timestream LiveAnalytics pour les mises à jour des politiques gérées AWS	De nouvelles informations sur les politiques AWS gérées et Amazon Timestream LiveAnalytics pour, notamment des mises à jour des politiques gérées existantes.	29 novembre 2021
Amazon Timestream LiveAnalytics pour prend en charge les requêtes planifiées	Amazon Timestream prend actuellement en charge LiveAnalytics l'exécution d'une requête en votre nom, selon un calendrier.	29 novembre 2021
Amazon Timestream LiveAnalytics pour les supports Magnetic Store.	Amazon Timestream prend actuellement en charge l'utilisation du stockage magnétique LiveAnalytics pour vos écritures de table.	29 novembre 2021
Amazon Timestream LiveAnalytics pour les enregistrements à mesures multiples.	Amazon Timestream prend actuellement en charge un format plus compact LiveAnalytics pour stocker vos données de séries chronologiques.	29 novembre 2021
Amazon Timestream LiveAnalytics pour les mises à jour des politiques gérées AWS	De nouvelles informations sur les politiques AWS gérées et Amazon Timestream LiveAnalytics pour, notamment des mises à jour des politiques gérées existantes.	24 mai 2021

Amazon Timestream LiveAnalytics for est désormais disponible dans la région Europe (Francfort).	Amazon Timestream LiveAnalytics pour est désormais généralement disponible dans la région Europe (Francfort) (). `eu-central-1`	23 avril 2021
Amazon Timestream LiveAnalytics for is VPC prend désormais en charge les points de terminaison ().AWS PrivateLink	Amazon Timestream prend actuellement en charge l'utilisation LiveAnalytics de points VPC de terminaison ().AWS PrivateLink	23 mars 2021
Amazon Timestream prend désormais en charge les requêtes entre tables.	Vous pouvez utiliser Amazon Timestream LiveAnalytics pour exécuter des requêtes entre tables.	10 février 2021
Amazon Timestream prend actuellement en charge LiveAnalytics les statistiques d'exécution des requêtes améliorées.	Amazon Timestream prend actuellement en charge LiveAnalytics les statistiques d'exécution des requêtes améliorées, telles que la quantité de données numérisées.	10 février 2021
Amazon Timestream prend actuellement en charge les fonctions avancées LiveAnalytics de séries chronologiques.	Vous pouvez utiliser Amazon Timestream LiveAnalytics pour SQL exécuter des requêtes avec des fonctions de séries chronologiques avancées, telles que des dérivées, des intégrales et des corrélations.	10 février 2021

Amazon Timestream LiveAnalytics for est HIPAA désormais conformeISO. PCI	Vous pouvez désormais utiliser Amazon Timestream LiveAnalytics pour les charges de travail qui HIPAA nécessite nt une infrastructure conformeI SO. PCI	27 janvier 2021
Amazon Timestream prend actuellement en charge les logiciels open source LiveAnalytics Telegraf et Grafana.	Vous pouvez désormais utiliser Telegraf, l'agent serveur open source piloté par des plugins pour collecter et générer des rapports de statistiques, et Grafana, la plateforme d'analyse et de surveillance open source pour les bases de données, avec Amazon Timestream pour. LiveAnalytics	25 novembre 2020
Amazon Timestream LiveAnalytics pour est désormais disponible pour tous.	Cette documentation couvre la version initiale d'Amazon LiveAnalytics Timestream pour.	30 septembre 2020

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.