aws

Guide du développeur SQL

Manuel du développeur des applications Amazon Kinesis Data Analytics pour SQL

Manuel du développeur des applications Amazon Kinesis Data Analytics pour SQL: Guide du développeur SQL

Table of Contents

Pour les nouveaux projets, nous vous recommandons d’utiliser le nouveau service géré pour Apache Flink Studio plutôt que les applications Kinesis Data Analytics pour SQL. Le service géré pour Apache Flink Studio allie facilité d’utilisation et capacités analytiques avancées, ce qui vous permet de créer des applications sophistiquées de traitement des flux en quelques minutes.

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.

Qu’est-ce qu’une application Amazon Kinesis Data Analytics pour SQL ?

Les applications Amazon Kinesis Data Analytics pour SQL vous permettent de traiter et d’analyser les données de streaming à l’aide du langage SQL standard. Le service vous permet de créer et d'exécuter rapidement un code SQL puissant sur des sources de diffusion pour effectuer des analyses de séries chronologiques, alimenter des tableaux de bord en temps réel et créer des métriques en temps réel.

Pour commencer avec Kinesis Data Analytics, vous créez une application Kinesis Data Analytics qui lit et traite en continu des données de streaming. Le service prend en charge l'ingestion de données provenant des sources de streaming Amazon Kinesis Data Streams et Amazon Data Firehose. Vous créez ensuite votre code SQL à l'aide de l'éditeur interactif et vous le testez avec des données de diffusion. Vous pouvez également configurer des destinations dans lesquelles vous souhaitez que Kinesis Data Analytics envoie les résultats.

Kinesis Data Analytics prend en charge Amazon Data Firehose (Amazon S3, Amazon Redshift, Amazon Service et Splunk) et OpenSearch Amazon AWS LambdaKinesis Data Streams comme destinations.

Quand dois-je utiliser Amazon Kinesis Data Analytics ?

Amazon Kinesis Data Analytics vous permet de créer rapidement du code SQL qui lit, traite et stocke en continu des données presque en temps réel. A l'aide de requêtes SQL standard sur les données de diffusion, vous pouvez créer des applications qui transforment vos données et en tirent des informations. Voici quelques exemples de scénarios pour l’utilisation de Kinesis Data Analytics :

- Générer des analyses chronologiques : vous pouvez calculer des métriques sur des fenêtres temporelles, puis transmettre des valeurs à Amazon S3 ou Amazon Redshift via un flux de diffusion de données Kinesis.
- Alimenter des tableaux de bord en temps réel : vous pouvez envoyer des résultats de données de streaming regroupés et traités en aval pour alimenter des tableaux de bord en temps réel.
- Créer des métriques en temps réel : vous pouvez créer des métriques et des déclencheurs personnalisés à utiliser pour une surveillance, des notifications et des alarmes en temps réel.

Pour plus d'informations sur les éléments du langage SQL pris en charge par Kinesis Data Analytics, consultez la Référence SQL Amazon Kinesis Data Analytics.

Vous utilisez Amazon Kinesis Data Analytics pour la première fois ?

Si vous utilisez Amazon Kinesis Data Analytics pour la première fois, nous vous recommandons de lire les sections suivantes dans l'ordre :

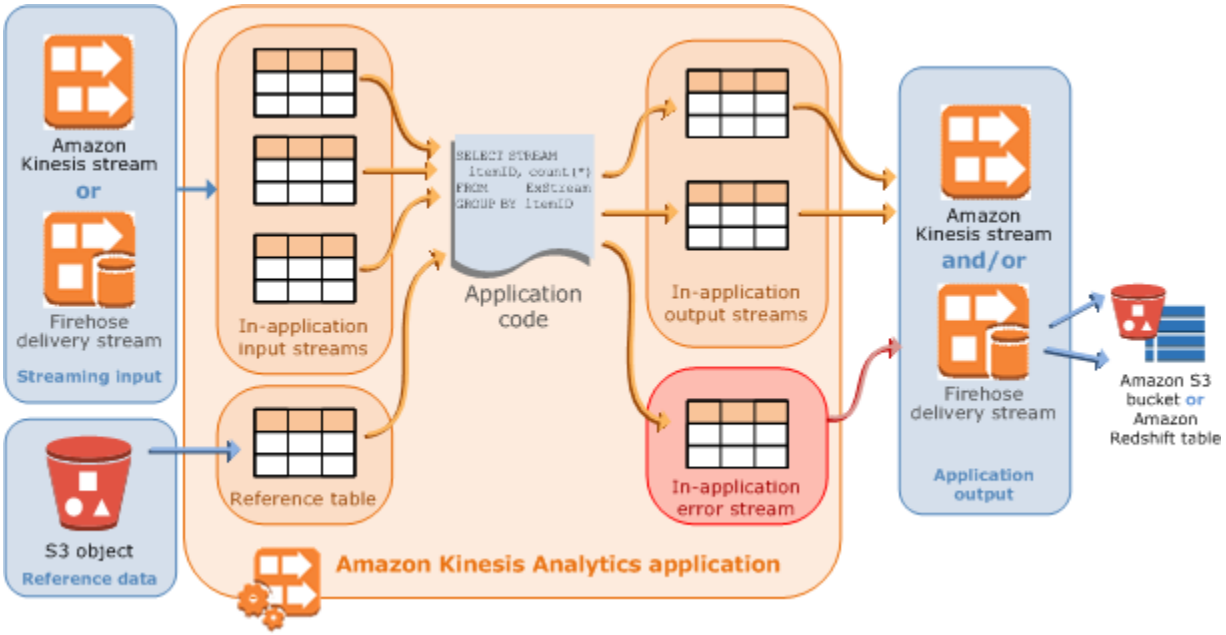
1. Consultez la section « Fonctionnement » de ce guide. Cette section présente les différents composants Kinesis Data Analytics que vous utilisez pour créer end-to-end une expérience. Pour de plus amples informations, veuillez consulter Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement.
2. Essayez les exercices Démarrez. Pour de plus amples informations, veuillez consulter Mise en route avec les applications Amazon Kinesis Data Analytics pour SQL.
3. Explorez les concepts du code SQL de streaming. Pour de plus amples informations, veuillez consulter Concepts du code SQL de streaming.
4. Essayez des exemples supplémentaires. Pour de plus amples informations, veuillez consulter Exemples de Kinesis Data Analytics pour SQL.

Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d'informations, consultez Limites .

Une application est la ressource principale d'Amazon Kinesis Data Analytics que vous pouvez créer dans votre compte. Vous pouvez créer et gérer des applications à l'aide de l' AWS Management Console API ou de l'API Kinesis Data Analytics. Kinesis Data Analytics fournit des opérations d'API pour gérer les applications. Pour obtenir une liste d'opérations d'API, consultez Actions.

Les applications Kinesis Data Analytics lisent et traitent en continu les données de streaming en temps réel. Vous écrivez le code d'application à l'aide de SQL pour traiter les données de diffusion entrantes et produire la sortie. Kinesis Data Analytics écrit ensuite la sortie vers une destination configurée. Le graphique suivant illustre une architecture d'applications typique.



Chaque application possède un nom, une description, un ID de version et un statut. Amazon Kinesis Data Analytics attribue un ID de version lorsque vous créez une application pour la première fois. Cet

ID de version est mis à jour lorsque vous mettez à jour une configuration d'application. Par exemple, si vous ajoutez une configuration d'entrée, si vous ajoutez ou supprimez une source de données de référence, si vous ajoutez ou supprimez une configuration de sortie, ou si vous mettez à jour le code d'application, Kinesis Data Analytics met à jour l'ID de version de l'application actuel. Kinesis Data Analytics gère également les horodatages des moments où une application a été créée et mise à jour pour la dernière fois.

En plus de ces propriétés de base, chaque application comprend les éléments suivants :

- Entrée : La source de streaming pour votre application. Vous pouvez sélectionner un flux de données Kinesis ou un flux de diffusion de données Firehose comme source de diffusion. Dans la configuration d'entrée, vous mappez la source de diffusion à un flux d'entrée intégré à l'application. Le flux intégré à l'application s'apparente à une table mise à jour en continu sur laquelle vous pouvez effectuer les opérations `SELECT` et `INSERT SQL`. Dans votre code d'application, vous pouvez créer des flux intégrés à l'application supplémentaires pour stocker des résultats de requête intermédiaires.

  Vous pouvez partitionner le cas échéant une source de diffusion unique en plusieurs flux d'entrée intégrés à l'application pour améliorer le débit. Pour plus d'informations, consultez Limites et Configuration de l'entrée de l'application.

  Amazon Kinesis Data Analytics fournit une colonne d'horodatage dans chaque flux d'application, appelée Horodatages et colonne ROWTIME. Vous pouvez utiliser cette colonne dans des requêtes à fenêtres temporelles. Pour de plus amples informations, veuillez consulter Requêtes à fenêtres.

  Vous pouvez configurer une source de données de référence pour enrichir votre flux de données d'entrée au sein de l'application. Celle-ci crée une table de référence intégrée à l'application. Vous devez stocker vos données de référence en tant qu'objet dans votre compartiment S3. Lorsque l'application démarre, Amazon Kinesis Data Analytics lit l'objet Amazon S3 et crée une table intégrée à l'application. Pour de plus amples informations, veuillez consulter Configuration de l'entrée de l'application.

- Code d'application : Ensemble d'instructions SQL qui traite une entrée et produit une sortie. Vous pouvez écrire des instructions SQL sur des flux et des tables de référence intégrés à l'application. Vous pouvez également écrire des requêtes JOIN pour combiner des données provenant de ces deux sources.

  Pour plus d'informations sur les éléments du langage SQL pris en charge par Kinesis Data Analytics, consultez la Référence SQL Amazon Kinesis Data Analytics.

  Dans sa forme la plus simple, le code d'application peut être une instruction SQL unique qui effectue une sélection à partir d'une entrée de diffusion et insère les résultats dans une sortie de diffusion. Il peut également s'agir d'un ensemble d'instructions SQL où la sortie d'un flux alimente l'entrée de l'instruction SQL suivante. En outre, vous pouvez écrire un code d'application pour scinder un flux d'entrée en plusieurs flux. Vous pouvez ensuite appliquer des requêtes supplémentaires pour traiter ces flux. Pour de plus amples informations, veuillez consulter Code d'application.

- Sortie : Dans le code d'application, les résultats d'une requête sont transmis à des flux intégrés à l'application. Dans votre code d'application, vous pouvez créer un ou plusieurs flux intégrés à l'application pour stocker des résultats intermédiaires. Vous pouvez ensuite configurer, le cas échéant, une sortie d'application pour conserver des données des flux intégrés à l'application qui contiennent votre sortie d'application (également appelés flux de sortie intégrés à l'application) dans des destinations externes. Les destinations externes peuvent être un flux de diffusion Firehose ou un flux de données Kinesis. Notez les points suivants à propos de ces destinations :
  - Vous pouvez configurer un flux de diffusion Firehose pour écrire les résultats sur Amazon S3, Amazon Redshift ou OpenSearch Amazon Service (ServiceOpenSearch).
  - Vous pouvez également écrire une sortie d'application dans une destination personnalisée, au lieu d'Amazon S3 ou Amazon Redshift. Pour ce faire, vous devez spécifier un flux de données Kinesis comme destination dans votre configuration de sortie. Ensuite, vous configurez AWS Lambda pour interroger le flux et appeler votre fonction Lambda. Le code de votre fonction Lambda reçoit des données de flux comme entrée. Dans le code de votre fonction Lambda,

vous pouvez écrire les données entrantes dans votre destination personnalisée. Pour plus d'informations, consultez Utilisation AWS Lambda avec Amazon Kinesis Data Analytics.

Pour de plus amples informations, veuillez consulter Configuration de la sortie d'application.

En outre, notez les éléments suivants :

- Amazon Kinesis Data Analytics a besoin d'autorisations pour lire les enregistrements à partir d'une source de streaming et écrire la sortie d'application dans les destinations externes. Vous utilisez des rôles IAM pour accorder ces autorisations.

- Kinesis Data Analytics fournit automatiquement un flux d'erreurs intégré à l'application pour chaque application. Si votre application rencontre des problèmes lors du traitement de certains enregistrements (par exemple en raison d'une incompatibilité de type ou d'une arrivée tardive), ces enregistrements seront écrits dans le flux d'erreurs. Vous pouvez configurer la sortie d'application pour demander à Kinesis Data Analytics de conserver les données du flux d'erreurs dans une destination externe pour une évaluation plus approfondie. Pour de plus amples informations, veuillez consulter Gestion des erreurs.

- Amazon Kinesis Data Analytics s'assure que vos enregistrements de sortie d'application sont écrits dans la destination configurée. Il utilise un modèle de traitement et de diffusion « au moins une fois », même en cas d'interruption de l'application. Pour de plus amples informations, veuillez consulter Modèle de diffusion pour la conservation de la sortie d'application dans une destination externe.

Rubriques

Configuration de l'entrée de l'application

Votre application Amazon Kinesis Data Analytics peut recevoir une entrée d'une seule source de streaming et, le cas échéant, utiliser une source de données de référence. Pour de plus amples informations, veuillez consulter Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement. Les sections de cette rubrique décrivent les sources d'entrée d'une application.

Rubriques

Configuration d'une source de diffusion

Lorsque vous créez une application, vous spécifiez une source de streaming. Vous pouvez également modifier une entrée après avoir créé l'application. Amazon Kinesis Data Analytics prend en charge les sources de streaming suivantes pour votre application :

- Un flux de données Kinesis
- Un flux de livraison Firehose

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Les clients existants qui utilisent les applications Kinesis Data Analytics pour SQL avec `KinesisFirehoseInput` peuvent continuer à ajouter des applications avec `KinesisFirehoseInput` au sein d'un compte existant à l'aide de Kinesis Data Analytics. Si vous êtes déjà client et que vous souhaitez créer un compte avec les applications Kinesis

Data Analytics pour SQL avec `KinesisFirehoseInput`, vous pouvez créer un cas via le formulaire d'augmentation de limite de service. Pour plus d'informations, consultez le CentreAWS Support. Nous vous recommandons de toujours tester les nouvelles applications avant de les mettre en production.

> **Note**
>
> Si le flux de données Kinesis est chiffré, Kinesis Data Analytics accède aux données du flux chiffré de façon transparente sans qu'aucune configuration supplémentaire ne soit nécessaire. Kinesis Data Analytics ne stocke pas les données non chiffrées lues depuis Kinesis Data Streams. Pour plus d'informations, consultez la page Qu'est-ce que le chiffrement côté serveur pour Kinesis Streams Data ?.

Kinesis Data Analytics interroge en continu la source de streaming pour les nouvelles données et intègre celles-ci dans des flux intégrés à l'application en fonction de la configuration d'entrée.

> **Note**
>
> L'ajout d'un flux Kinesis en tant qu'entrée de votre application n'affecte pas les données du flux. Si une autre ressource, telle qu'un flux de diffusion Firehose, accédait également au même flux Kinesis, le flux de diffusion Firehose et l'application Kinesis Data Analytics recevraient les mêmes données. Toutefois, le débit et la limitation peuvent être affectés.

Votre code d'application peut interroger le flux intégré à l'application. Dans le cadre de la configuration d'entrée, vous fournissez les éléments suivants :

- Source de streaming : Vous fournissez l'Amazon Resource Name (ARN) du flux et un rôle IAM que Kinesis Data Analytics peut assumer pour accéder au flux en votre nom.
- Préfixe du nom de flux intégré à l'application : Lorsque vous démarrez l'application, Kinesis Data Analytics crée le flux intégré à l'application spécifié. Dans votre code d'application, vous accédez au flux intégré à l'application à l'aide de ce nom.

  Vous pouvez mapper le cas échéant une source de diffusion à plusieurs flux intégrés à l'application. Pour de plus amples informations, veuillez consulter Limites. Dans ce cas, Amazon Kinesis Data Analytics crée le nombre spécifié de flux intégrés à l'application avec des noms

comme suit : *prefix*_001, *prefix*_002 et *prefix*_003. Par défaut, Kinesis Data Analytics mappe la source de streaming à un flux intégré à l'application appelé *prefix*_001.

Une limite est appliqué au rythme auquel vous pouvez insérer des lignes dans un flux intégré à l'application. Par conséquent, Kinesis Data Analytics prend en charge plusieurs flux intégrés à l'application pour vous permettre de transmettre des enregistrements dans votre application à un rythme beaucoup plus rapide. Si vous constatez que votre application n'arrive pas à suivre le rythme des données de la source de diffusion, vous pouvez ajouter des unités de parallélisme pour améliorer les performances.

- Schéma de mappage : Vous décrivez le format d'enregistrement (JSON, CSV) dans la source de streaming. Vous décrivez également la façon dont chaque enregistrement du flux est mappé à des colonnes dans le flux intégré à l'application qui est créé. C'est là où vous fournissez des noms de colonnes et des types de données.

> **Note**
>
> Kinesis Data Analytics place les identifiants (nom du flux et noms de colonnes) entre guillemets lors de la création du flux intégré à l'application. Lors de l'interrogation de ce flux et des colonnes, vous devez les spécifier entre guillemets en utilisant exactement la même casse (avec les mêmes lettres minuscules et majuscules). Pour plus d'informations sur les identifiants, consultez la section Identifiants dans le manuel Référence SQL du service géré Amazon pour Apache Flink SQL.

Vous pouvez créer une application et configurer des entrées dans la console Amazon Kinesis Data Analytics. La console effectue ensuite les appels d'API nécessaires. Vous pouvez configurer l'entrée d'application lorsque vous créez une nouvelle API d'application ou ajouter la configuration d'entrée à une application existante. Pour plus d'informations, consultez CreateApplication et AddApplicationInput. Voici la partie de configuration d'entrée du corps de requête d'API `Createapplication` :

```
"Inputs": [
      {
          "InputSchema": {
              "RecordColumns": [
                  {
                      "Mapping": "string",
                      "Name": "string",
```

```
                    "SqlType": "string"
                }
            ],
            "RecordEncoding": "string",
            "RecordFormat": {
                "MappingParameters": {
                    "CSVMappingParameters": {
                        "RecordColumnDelimiter": "string",
                        "RecordRowDelimiter": "string"
                    },
                    "JSONMappingParameters": {
                        "RecordRowPath": "string"
                    }
                },
                "RecordFormatType": "string"
            }
        },
        "KinesisFirehoseInput": {
            "ResourceARN": "string",
            "RoleARN": "string"
        },
        "KinesisStreamsInput": {
            "ResourceARN": "string",
            "RoleARN": "string"
        },
        "Name": "string"
    }
]
```

Configuration d'une source de référence

Vous pouvez également ajouter le cas échéant une source de données de référence à une application existante pour enrichir les données en provenance de sources de diffusion. Vous devez stocker les données de référence en tant qu'objet dans votre compartiment Amazon S3. Lorsque l'application démarre, Amazon Kinesis Data Analytics lit l'objet Amazon S3 et crée une table de référence intégrée à l'application. Votre code d'application peut ensuite la joindre à un flux intégré à l'application.

Vous stockez les données de référence dans l'objet Amazon S3 à l'aide des formats pris en charge (CSV, JSON). Par exemple, supposons que votre application effectue des analyses sur des ordres de bourse. Supposons que le format d'enregistrement suivant soit utilisé sur la source de diffusion :

```
Ticker, SalePrice, OrderId

AMZN     $700        1003
XYZ      $250        1004
...
```

Dans ce cas, vous pouvez envisager de gérer une source de données de référence afin de fournir des détails pour chaque symbole boursier, comme un nom de société.

```
Ticker, Company
AMZN, Amazon
XYZ, SomeCompany
...
```

Vous pouvez ajouter une source de données de référence d'application à l'aide de l'API ou de la console. Amazon Kinesis Data Analytics fournit les actions d'API suivantes pour gérer les sources de données de référence :

- AddApplicationReferenceDataSource
- UpdateApplication

Pour plus d'informations sur l'ajout de données de référence; consultez Exemple : ajout de données de référence à une application Kinesis Data Analytics.

Notez ce qui suit :

- Si l'application est en cours d'exécution, Kinesis Data Analytics crée un tableau de référence intégré à l'application, puis charge les données de référence immédiatement.
- Si l'application n'est pas en cours d'exécution (par exemple, elle est à l'état prêt), Kinesis Data Analytics enregistre uniquement la configuration d'entrée mise à jour. Lorsque l'application commence à s'exécuter, Kinesis Data Analytics charge les données de référence dans votre application en tant que tableau.

Supposons que vous souhaitez actualiser les données après que Kinesis Data Analytics a créé le tableau de référence intégré à l'application. Vous avez peut-être mis à jour l'objet Amazon S3, ou vous souhaitez utiliser un autre objet Amazon S3. Dans ce cas, vous pouvez soit appeler explicitement UpdateApplication, soit choisir Actions, Synchroniser le tableau de données de

référence dans la console. Kinesis Data Analytics n'actualise pas automatiquement le tableau de référence intégré à l'application.

Une limite est appliquée à la taille de l'objet Amazon S3 que vous pouvez créer comme source de données de référence. Pour de plus amples informations, veuillez consulter Limites. Si la taille de l'objet dépasse la limite, Kinesis Data Analytics ne peut pas charger les données. L'état de l'application s'affiche comme étant en cours d'exécution, mais les données ne sont pas lues.

Lorsque vous ajoutez une source de données de référence, vous fournissez les informations suivantes :

- Nom du compartiment S3 et de la clé d'objet : En plus du nom du compartiment et de la clé d'objet, vous fournissez également un rôle IAM que Kinesis Data Analytics peut assumer pour lire l'objet en votre nom.
- Nom du tableau de référence intégré à l'application : Kinesis Data Analytics crée ce tableau intégré à l'application et le remplit en lisant l'objet Amazon S3. Il s'agit du nom de table que vous spécifiez dans votre code d'application.
- Schéma de mappage : vous décrivez le format d'enregistrement (JSON, CSV), l'encodage des données stockées dans l'objet Amazon S3. Vous décrivez également la façon dont chaque élément de données est mappé aux colonnes de la table de référence intégrée à l'application.

Voici le corps de la demande dans la demande d'API `AddApplicationReferenceDataSource`.

```
{
    "applicationName": "string",
    "CurrentapplicationVersionId": number,
    "ReferenceDataSource": {
        "ReferenceSchema": {
            "RecordColumns": [
                {
                    "IsDropped": boolean,
                    "Mapping": "string",
                    "Name": "string",
                    "SqlType": "string"
                }
            ],
            "RecordEncoding": "string",
            "RecordFormat": {
                "MappingParameters": {
                    "CSVMappingParameters": {
```

```
                        "RecordColumnDelimiter": "string",
                        "RecordRowDelimiter": "string"
                    },
                    "JSONMappingParameters": {
                        "RecordRowPath": "string"
                    }
                },
                "RecordFormatType": "string"
            }
        },
        "S3ReferenceDataSource": {
            "BucketARN": "string",
            "FileKey": "string",
            "ReferenceRoleARN": "string"
        },
        "TableName": "string"
    }
}
```

Utilisation de JSONPath

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d'informations, consultez Limites .

JSONPath est une méthode normalisée d'interrogation des éléments d'un objet JSON. JSONPath utilise des expressions de chemin pour parcourir les éléments, les éléments imbriqués et les tableaux d'un document JSON. Pour de plus amples informations sur JSON, veuillez consulter Introducing JSON.

Amazon Kinesis Data Analytics utilise des expressions JSONPath dans le schéma source de l'application pour identifier les éléments de données dans une source de streaming qui contient des données au format JSON.

Pour plus d'informations sur la façon de mapper les données en streaming sur le flux d'entrée de l'application, consultez the section called "Mappage d'éléments d'une source de streaming à des colonnes d'entrée SQL".

Accès à des éléments JSON avec JSONPath

Vous découvrirez ci-dessous comment utiliser des expressions JSONPath pour accéder à divers éléments de données au format JSON. Pour les exemples de cette section, supposons que le flux source contient l'enregistrement JSON suivant :

```
{
  "customerName":"John Doe",
  "address":
  {
    "streetAddress":
    [
      "number":"123",
      "street":"AnyStreet"
    ],
    "city":"Anytown"
  }
  "orders":
  [
    { "orderId":"23284", "itemName":"Widget", "itemPrice":"33.99" },
    { "orderId":"63122", "itemName":"Gadget", "itemPrice":"22.50" },
    { "orderId":"77284", "itemName":"Sprocket", "itemPrice":"12.00" }
  ]
}
```

Accès à des éléments JSON

Pour interroger un élément dans des données JSON à l'aide de JSONPath, utilisez la syntaxe suivante. Ici, $ représente la racine de la hiérarchie des données et `elementName` est le nom du nœud de l'élément à interroger.

```
$.elementName
```

L'expression suivante interroge l'élément `customerName` de l'exemple JSON précédent.

```
$.customerName
```

L'expression précédente retourne les informations suivantes à partir de l'enregistrement JSON précédent.

```
John Doe
```

> **Note**
>
> Les expressions de chemin sont sensibles à la casse. L'expression `$.customername` retourne `null` à partir de l'exemple JSON précédent.

> **Note**
>
> Si aucun élément n'apparaît à l'emplacement spécifié dans l'expression de chemin, l'expression retourne `null`. L'expression suivante retourne `null` à partir de l'exemple JSON précédent, car il n'y a aucun élément correspondant.
>
> ```
> $.customerId
> ```

Accès aux éléments JSON imbriqués

Pour interroger un élément JSON imbriqué, utilisez la syntaxe suivante.

```
$.parentElement.element
```

L'expression suivante interroge l'élément `city` de l'exemple JSON précédent.

```
$.address.city
```

L'expression précédente retourne les informations suivantes à partir de l'enregistrement JSON précédent.

```
Anytown
```

Vous pouvez interroger d'autres niveaux de sous-éléments à l'aide de la syntaxe suivante.

```
$.parentElement.element.subElement
```

L'expression suivante interroge l'élément `street` de l'exemple JSON précédent.

```
$.address.streetAddress.street
```

L'expression précédente retourne les informations suivantes à partir de l'enregistrement JSON précédent.

```
AnyStreet
```

Accès aux tableaux

Vous pouvez accéder aux données dans un tableau JSON de la manière suivante :

- Récupérer tous les éléments du tableau sous la forme d'une ligne unique.
- Récupérer chaque élément du tableau sous la forme d'une ligne distincte.

Récupérer tous les éléments du tableau dans une ligne unique.

Pour interroger tout le contenu d'un tableau sur une seule ligne, utilisez la syntaxe suivante.

```
$.arrayObject[0:]
```

L'expression suivante interroge tout le contenu de l'élément `orders` de l'exemple JSON précédent utilisé dans cette section. Elle retourne le contenu du tableau dans une seule colonne et sur une seule ligne.

```
$.orders[0:]
```

L'expression précédente retourne les éléments suivants de l'exemple JSON utilisé dans cette section.

```
[{"orderId":"23284","itemName":"Widget","itemPrice":"33.99"},
{"orderId":"61322","itemName":"Gadget","itemPrice":"22.50"},
{"orderId":"77284","itemName":"Sprocket","itemPrice":"12.00"}]
```

Récupérez tous les éléments dans un tableau dans des lignes séparées

Pour interroger chaque élément d'un tableau sur une ligne distincte, utilisez la syntaxe suivante.

```
$.arrayObject[0:].element
```

L'expression suivante interroge les éléments `orderId` de l'exemple JSON précédent et retourne chaque élément du tableau sur une ligne distincte.

```
$.orders[0:].orderId
```

L'expression précédente retourne les informations suivantes à partir de l'enregistrement JSON précédent, chaque élément de données étant retourné sur une ligne distincte.

23284
63122
77284

> **Note**
>
> Si un schéma interrogeant des éléments de tableau comporte des expressions interrogeant des éléments autres, ces derniers sont répétés pour chaque élément du tableau. Par exemple, supposons qu'un schéma de l'exemple JSON précédent inclut les expressions suivantes :
>
> - $.customerName
> - $.orders[0:].orderId
>
> Dans ce cas, les lignes de données retournées de l'exemple d'élément de flux d'entrée ressemble à ce qui suit, l'élément `name` étant répété pour chaque élément `orderId`.
>
> | | |
> |---|---|
> | Jean Dupont | 23284 |
> | Jean Dupont | 63122 |
> | Jean Dupont | 77284 |

> **Note**
>
> Les limitations suivantes s'appliquent aux expressions de tableau dans Amazon Kinesis Data Analytics :
>
> - Seul un niveau de déréférencement est pris en charge dans une expression de tableau. Le format d'expression suivant n'est pas pris en charge.
>
> ```
> $.arrayObject[0:].element[0:].subElement
> ```
>
> - Seul un tableau peut être mis à plat dans un schéma. Plusieurs tableaux peuvent être référencés/retournés sur une ligne contenant tous les éléments du tableau. Cependant, seul un tableau peut avoir chacun de ses éléments retournés sur des lignes distinctes.
>
> Un schéma contenant des éléments au format suivant est valide. Ce format retourne le contenu du second tableau dans une seule colonne, répétée pour chaque élément du premier tableau.
>
> ```
> $.arrayObjectOne[0:].element
> $.arrayObjectTwo[0:]
> ```
>
> Un schéma contenant des éléments au format suivant n'est pas valide.
>
> ```
> $.arrayObjectOne[0:].element
> $.arrayObjectTwo[0:].element
> ```

Autres considérations

Voici quelques considérations supplémentaires concernant l'utilisation de JSONPath :

- Si aucun élément de l'expression JSONPath n'accède aux tableaux dans le schéma d'application, une ligne est créée dans le flux d'entrée de l'application pour chaque enregistrement JSON traité.
- Lorsqu'un tableau est mis à plat (c'est-à-dire que ses éléments sont retournés sous la forme de lignes individuelles), tous les éléments manquants retournent une valeur nulle créée dans le flux intégré à l'application.

- Un tableau est toujours mis à plat sur une ligne minimum. Si aucune valeur n'est retournée (c'est-à-dire, si le tableau est vide ou si aucun de ses éléments n'est interrogé), une seule ligne comportant toutes les valeurs nulles est retournée.

  L'expression suivante retourne les enregistrements de l'exemple JSON précédent qui ont une valeur nulle, car il n'existe aucun élément correspondant à l'emplacement spécifié.

  ```
  $.orders[0:].itemId
  ```

  L'expression précédente retourne les éléments suivants de l'exemple JSON précédent.

null
null
null

Rubriques connexes

- Présentation de JSON

Mappage d'éléments d'une source de streaming à des colonnes d'entrée SQL

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d'informations, consultez Limites .

Avec Amazon Kinesis Data Analytics, vous pouvez traiter et analyser des données de streaming au format JSON ou CSV à l'aide du langage SQL standard.

- Pour traiter et analyser des données de streaming au format CSV, vous devez attribuer des noms de colonne et des types de données aux colonnes du flux d'entrée. Votre application importe une colonne du flux d'entrée par définition de colonne, dans l'ordre.

  Vous n'avez pas besoin d'inclure toutes les colonnes du flux d'entrée de l'application, mais vous ne pouvez pas ignorer les colonnes du flux source. Par exemple, vous pouvez importer les trois premières colonnes d'un flux d'entrée contenant cinq éléments, mais vous ne pouvez pas importer uniquement les colonnes 1, 2 et 4.
- Pour traiter et analyser des données de streaming au format JSON, vous utilisez des expressions JSONPath pour mapper des éléments JSON d'une source de streaming à des colonnes SQL d'un flux d'entrée. Pour de plus amples informations sur l'utilisation de JSONPath avec Amazon Kinesis Data Analytics, consultez Utilisation de JSONPath. Les colonnes de la table SQL possèdent des types de données qui sont mappées à partir de types JSON. Pour connaître les types de données pris en charge, consultez Types de données. Pour plus d'informations sur la conversion des données JSON en données SQL, consultez Mappage de types de données JSON à des types de données SQL.

Pour plus d'informations sur la configuration des flux d'entrée, consultez Configuration de l'entrée de l'application.

Mappage de données JSON à des colonnes SQL

Vous pouvez mapper des éléments JSON à des colonnes d'entrée à l'aide de l'API AWS Management Console ou de l'API Kinesis Data Analytics.

- Pour mapper des éléments à des colonnes à l'aide de la console, consultez Utilisation de l'éditeur de schéma.
- Pour mapper des éléments à des colonnes à l'aide de l'API Kinesis Data Analytics, reportez-vous à la section suivante.

Pour mapper des éléments JSON à des colonnes du flux d'entrée intégré à l'application, vous avez besoin d'un schéma avec les informations suivantes pour chaque colonne :

- Expression source : Expression JSONPath qui identifie l'emplacement des données pour la colonne.
- Nom de la colonne : Nom que vos requêtes SQL utilisent pour référencer les données.
- Type de données : Type de données SQL de la colonne.

Utilisation de l’API

Pour mapper des éléments d’une source de streaming à des colonnes d’entrée, vous pouvez utiliser l’action CreateApplication de l’API Kinesis Data Analytics. Pour créer le flux intégré à l'application, spécifiez un schéma pour transformer vos données en une version schématisée utilisée dans SQL. L'action CreateApplication configure votre application pour qu'elle reçoive des entrées d'une source de streaming. Pour mapper des éléments JSON ou des colonnes CSV à des colonnes SQL, vous créez un objet RecordColumn dans le tableau SourceSchema `RecordColumns`. L'objet RecordColumn présente le schéma suivant:

```
{
    "Mapping": "String",
    "Name": "String",
    "SqlType": "String"
}
```

Les champs de l'objet RecordColumn possèdent les valeurs suivantes :

- `Mapping` : Expression JSONPath qui identifie l'emplacement des données dans l'enregistrement du flux d'entrée. Cette valeur n'est pas disponible pour le schéma d'entrée d'un flux source au format CSV.
- `Name` : Nom de la colonne dans le flux de données SQL intégré à l'application.
- `SqlType` : Type des données du flux de données SQL intégré à l'application.

Exemple de schéma d'entrée JSON

L'exemple suivant présente le format de la valeur `InputSchema` pour un schéma JSON.

```
"InputSchema": {
    "RecordColumns": [
        {
            "SqlType": "VARCHAR(4)",
            "Name": "TICKER_SYMBOL",
            "Mapping": "$.TICKER_SYMBOL"
        },
        {
            "SqlType": "VARCHAR(16)",
            "Name": "SECTOR",
```

```
            "Mapping": "$.SECTOR"
        },
        {
            "SqlType": "TINYINT",
            "Name": "CHANGE",
            "Mapping": "$.CHANGE"
        },
        {
            "SqlType": "DECIMAL(5,2)",
            "Name": "PRICE",
            "Mapping": "$.PRICE"
        }
    ],
    "RecordFormat": {
        "MappingParameters": {
            "JSONMappingParameters": {
                "RecordRowPath": "$"
            }
        },
        "RecordFormatType": "JSON"
    },
    "RecordEncoding": "UTF-8"
}
```

Exemple de schéma d'entrée CSV

L'exemple suivant présente le format de la valeur `InputSchema` pour un schéma CSV.

```
"InputSchema": {
    "RecordColumns": [
        {
            "SqlType": "VARCHAR(16)",
            "Name": "LastName"
        },
        {
            "SqlType": "VARCHAR(16)",
            "Name": "FirstName"
        },
        {
            "SqlType": "INTEGER",
             "Name": "CustomerId"
        }
    ],
```

```
    "RecordFormat": {
        "MappingParameters": {
            "CSVMappingParameters": {
                "RecordColumnDelimiter": ",",
                "RecordRowDelimiter": "\n"
            }
        },
        "RecordFormatType": "CSV"
    },
    "RecordEncoding": "UTF-8"
}
```

Mappage de types de données JSON à des types de données SQL

Les types de données JSON sont convertis en types de données SQL correspondants selon le schéma d'entrée de l'application. Pour plus d'informations sur les types de données SQL pris en charge, consultez Types de données. Amazon Kinesis Data Analytics convertit les types de données JSON en types de données SQL selon les règles suivantes.

Littéral null

Un littéral null dans le flux d'entrée JSON (c'est-à-dire `"City":null`) est converti en null SQL, quel que soit le type de données de destination.

Littéral booléen

Un littéral booléen dans le flux d'entrée JSON (c'est-à-dire `"Contacted":true`) est converti en données SQL comme suit :

- Numérique (DECIMAL, INT, etc.) : `true` est converti en 1 ; `false` est converti en 0.
- Binaire (BINARY ou VARBINARY) :
  - `true` : Le résultat possède l'ensemble de bits le plus faible, les autres bits sont effacés.
  - `false` : Tous les bits sont effacés.

  La conversion en données VARBINARY génère une valeur de 1 octet de longueur.
- BOOLEEN : Conversion dans la valeur BOOLEENNE SQL correspondante.
- Caractère (CHAR ou VARCHAR) : Conversion dans la valeur de chaîne correspondante (`true` ou `false`). La valeur est tronquée pour s'adapter à la longueur du champ.

- Date/heure (DATE, TIME ou TIMESTAMP) : La conversion échoue et une erreur de forçage de type est écrite dans le flux d'erreurs.

Nombre

Un littéral numérique dans le flux d'entrée JSON (c'est-à-dire `"CustomerId":67321`) est converti en données SQL comme suit :

- Numérique (DECIMAL, INT, etc.) : Conversion directe. Si la valeur convertie dépasse la taille ou la précision du type de données cible (c'est-à-dire, conversion de `123.4` en INT), la conversion échoue et une erreur de forçage du type est écrite dans le flux d'erreurs.
- Binaire (BINARY or VARBINARY) : La conversion échoue et une erreur de forçage du type est écrite dans le flux d'erreurs.
- BOOLEAN:
  - `0` : Conversion `false`.
  - Tous les autres nombres : Conversion en `true`.
- Caractère (CHAR ou VARCHAR) : Conversion en une représentation de chaîne du nombre.
- Date/heure (DATE, TIME ou TIMESTAMP) : La conversion échoue et une erreur de forçage de type est écrite dans le flux d'erreurs.

Chaîne

Une valeur de chaîne dans le flux d'entrée JSON (c'est-à-dire `"CustomerName":"John Doe"`) est convertie en données SQL comme suit :

- Numérique (DECIMAL, INT, etc.) : Amazon Kinesis Data Analytics tente de convertir la valeur dans le type de données cible. Si la valeur ne peut être convertie, la conversion échoue et une erreur de forçage du type est écrite dans le flux d'erreurs.
- Binaire (BINARY ou VARBINARY) : Si la chaîne source est un littéral binaire valide (c'est-à-dire `X'3F67A23A'`, avec un nombre pair de f), la valeur est convertie dans le type de données cible. Dans le cas contraire, la conversion échoue et une erreur de forçage du type est écrite dans le flux d'erreurs.
- BOOLEEN : Si la chaîne source est `"true"`, conversion en `true`. Cette comparaison n'est pas sensible à la casse. Sinon, conversion en `false`.

- Caractère (CHAR ou VARCHAR) : Conversion dans la valeur de chaîne de l'entrée. Si la valeur est plus longue que le type de données cible, elle est tronquée et aucune erreur n'est écrite dans le flux d'erreurs.
- Date/heure (DATE, TIME ou TIMESTAMP) : Si la chaîne source est dans un format convertible dans la valeur cible, la valeur est convertie. Dans le cas contraire, la conversion échoue et une erreur de forçage du type est écrite dans le flux d'erreurs.

  Les formats de date et d'heure sont les suivants :

  - "1992-02-14"
  - "1992-02-14 18:35:44.0"

Tableau ou objet

Un tableau ou un objet dans le flux d'entrée JSON est converti en données SQL comme suit :

- Caractère (CHAR ou VARCHAR) : Convertit en texte source du tableau ou de l'objet. veuillez consulter Accès aux tableaux.
- Tous les autres types de données : La conversion échoue et une erreur de forçage du type est écrite dans le flux d'erreurs.

Pour voir un exemple de tableau JSON, consultez Utilisation de JSONPath.

Rubriques connexes

- Configuration de l'entrée de l'application
- Types de données
- Utilisation de l'éditeur de schéma
- CreateApplication
- RecordColumn
- SourceSchema

Utilisation de la fonction de découverte de schéma sur des données de diffusion

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d'informations, consultez Limites .

Fournir un schéma d'entrée qui décrit la façon dont les enregistrements de l'entrée de diffusion sont mappés à un flux intégré à l'application peut s'avérer fastidieux et source d'erreurs. Vous pouvez utiliser l'API DiscoverInputSchema (appelée API de découverte) pour déduire un schéma. En utilisant des échantillons aléatoires d'enregistrements de la source de diffusion, l'API peut déduire un schéma (autrement dit, les noms de colonnes, les types de données et la position de l'élément de données dans les données entrantes).

> **Note**
>
> Pour utiliser l'API de découverte pour générer un schéma à partir d'un fichier stocké dans Amazon S3, consultez Utilisation de la fonction de découverte de schéma sur des données statiques.

La console utilise l'API de découverte pour générer un schéma pour une source de streaming spécifiée. A l'aide de la console, vous pouvez également mettre le schéma à jour, y compris ajouter, supprimer ou renommer des colonnes et modifier des types de données. Cependant, apportez les modifications avec soin pour veiller à ne pas créer un schéma non valide.

Une fois que vous avez finalisé le schéma de votre flux intégré à l'application, vous pouvez modifier des valeurs de chaîne et de date/heure. Vous pouvez utiliser ces fonctions dans votre code d'application lorsque vous utilisez des lignes dans le flux intégré à l'application qui en résulte. Pour de plus amples informations, veuillez consulter Exemple : transformation DateTime des valeurs.

Attribution de noms de colonnes pendant la découverte de schéma

Pendant la découverte de schéma, Amazon Kinesis Data Analytics tente de conserver autant que possible le nom d'origine de la colonne de la source d'entrée de streaming, sauf dans les cas suivants :

- Le nom de la colonne du flux source est un mot réservé SQL, par exemple `TIMESTAMP`, `USER`, `VALUES` ou `YEAR`.
- Le nom de la colonne du flux source contient des caractères non pris en charge. Seuls les lettres, les chiffres et les caractères de soulignement (_) sont pris en charge.
- Le nom de la colonne du flux source commence par un chiffre.
- Le nom de la colonne du flux source comporte plus de 100 caractères.

Si une colonne est renommée, son nouveau nom commence par `COL_`. Dans certains cas, aucun nom de colonne d'origine ne peut être conservé, par exemple si le nom complet est composé de caractères non pris en charge. Dans ce cas, la colonne est nommée `COL_#`, # correspondant au chiffre de l'emplacement de la colonne dans l'ordre des colonnes.

Une fois la découverte terminée, vous pouvez mettre le schéma à jour à l'aide de la console en ajoutant, supprimant ou renommant des colonnes ou en modifiant les types ou la taille des données.

Exemples de noms de colonnes suggérés lors de la découverte

Nom de la colonne dans le flux source	Nom de colonne suggéré lors de la découverte
USER	COL_USER
USER@DOMAIN	COL_USERDOMAIN
@@	COL_0

Problèmes liés à la découverte d'un schéma

Que se passe-t-il si Kinesis Data Analytics ne déduit pas un schéma pour une source de streaming donnée ?

Kinesis Data Analytics déduit votre schéma pour les formats courants, tels que CSV et JSON, qui sont codés en UTF-8. Kinesis Data Analytics prend en charge tous les enregistrements codés en

UTF-8 (y compris le texte brut, comme des journaux d'applications, et des enregistrements) avec un délimiteur de ligne et de colonne personnalisé. Si Kinesis Data Analytics ne déduit pas un schéma, vous pouvez définir un schéma manuellement à l'aide de l'éditeur de schéma dans la console (ou à l'aide de l'API).

Si vous vos données ne suivent pas un modèle (que vous pouvez spécifier à l'aide de l'éditeur de schéma), vous pouvez définir un schéma en tant que colonne unique de type VARCHAR(N), où N est le plus grand nombre de caractères que vous prévoyez que votre enregistrement puisse inclure. A partir de là, vous pouvez utiliser une manipulation de chaîne et de date/heure pour structurer vos données une fois que celles-ci sont dans un flux intégré à l'application. Pour obtenir des exemples, consultez Exemple : transformation DateTime des valeurs.

Utilisation de la fonction de découverte de schéma sur des données statiques

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d'informations, consultez Limites .

La fonction de découverte de schéma peut générer un schéma à partir des données dans un flux ou des données dans un fichier statique qui est stocké dans un compartiment Amazon S3. Supposons que vous souhaitez générer un schéma pour une application Kinesis Data Analytics à des fins de référence ou lorsque des données de streaming ne sont pas disponibles. Vous pouvez utiliser la fonction de découverte de schéma sur un fichier statique qui contient un échantillon de données au format attendu de vos données de diffusion ou de référence. Kinesis Data Analytics peut exécuter une découverte de schéma sur des exemples de données provenant d'un fichier JSON ou CSV stocké dans un compartiment Amazon S3. L'utilisation de la découverte de schéma sur un fichier de données fait appel soit à la console, soit à l'API DiscoverInputSchema avec le paramètre `S3Configuration` spécifié.

Exécution de la découverte de schéma à l'aide de la console

Pour exécuter la découverte sur un fichier statique à l'aide de la console, procédez comme suit :

1. Ajoutez un objet de données de référence à un compartiment S3.

2. Choisissez Connecter des données de référence dans la page principale de l'application dans la console Kinesis Data Analytics.
3. Fournissez les données relatives au compartiment, au chemin d'accès et au rôle IAM pour pouvoir accéder à l'objet Amazon S3 contenant les données de référence.
4. Choisissez Discover schema (Découvrir le schéma).

Pour plus d'informations sur l'ajout des données de référence et la découverte du schéma dans la console, consultez Exemple : ajout de données de référence à une application Kinesis Data Analytics.

Exécution de la découverte de schéma à l'aide de l'API

Pour exécuter la découverte sur un fichier statique à l'aide de l'API, vous devez fournir l'API avec une structure `S3Configuration` à l'aide des informations suivantes :

- `BucketARN` : l'Amazon Resource Name (ARN) du compartiment Amazon S3 qui contient le fichier. Pour le format d'un ARN de compartiment Amazon S3, consultez Amazon Resource Name (ARN) et Espaces de noms Amazon : Amazon Simple Storage Service (Amazon S3).
- `RoleARN` : l'ARN d'un rôleIAM avec la stratégie `AmazonS3ReadOnlyAccess`. Pour plus d'informations sur comment ajouter une stratégie à un rôle, consultez Modification d'un rôle.
- `FileKey` : le nom de fichier de l'objet.

Pour générer un schéma à partir d'un objet Amazon S3 à l'aide de l'API **`DiscoverInputSchema`**

1. Assurez-vous que vous disposez de la AWS CLI configuration requise. Pour plus d'informations, consultez Étape 2 : configurer le AWS Command Line Interface (AWS CLI) dans la section Mise en route.
2. Créez un fichier nommé `data.csv` avec le contenu suivant :

```
year,month,state,producer_type,energy_source,units,consumption
2001,1,AK,TotalElectricPowerIndustry,Coal,ShortTons,47615
2001,1,AK,ElectricGeneratorsElectricUtilities,Coal,ShortTons,16535
2001,1,AK,CombinedHeatandPowerElectricPower,Coal,ShortTons,22890
2001,1,AL,TotalElectricPowerIndustry,Coal,ShortTons,3020601
2001,1,AL,ElectricGeneratorsElectricUtilities,Coal,ShortTons,2987681
```

3. Connectez-vous à la console Amazon S3 à l'adresse https://console.aws.amazon.com/s3/.

4. Créez un compartiment Amazon S3 et chargez le fichier `data.csv` que vous avez créé. Notez l'ARN du compartiment créé. Pour plus d'informations sur la création d'un compartiment Amazon S3 et le chargement d'un fichier, consultez Démarrez avec Amazon Simple Storage Service.

5. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/. Créez un rôle avec la stratégie `AmazonS3ReadOnlyAccess`. Notez l'ARN du nouveau rôle. Pour plus d'informations sur la création d'un rôle, consultez Création d'un rôle pour déléguer des autorisations à un service Amazon. Pour plus d'informations sur comment ajouter une stratégie à un rôle, consultez Modification d'un rôle.

6. Exécutez la `DiscoverInputSchema` commande suivante dans le AWS CLI, en remplaçant les ARN par votre compartiment Amazon S3 et votre rôle IAM :

```
$aws kinesisanalytics discover-input-schema --s3-configuration '{ "RoleARN":
 "arn:aws:iam::123456789012:role/service-role/your-IAM-role", "BucketARN":
 "arn:aws:s3:::your-bucket-name", "FileKey": "data.csv" }'
```

7. La réponse ressemble à ce qui suit :

```
{
    "InputSchema": {
        "RecordEncoding": "UTF-8",
        "RecordColumns": [
            {
                "SqlType": "INTEGER",
                "Name": "COL_year"
            },
            {
                "SqlType": "INTEGER",
                "Name": "COL_month"
            },
            {
                "SqlType": "VARCHAR(4)",
                "Name": "state"
            },
            {
                "SqlType": "VARCHAR(64)",
                "Name": "producer_type"
            },
            {
                "SqlType": "VARCHAR(4)",
                "Name": "energy_source"
            },
```

```
            {
                "SqlType": "VARCHAR(16)",
                "Name": "units"
            },
            {
                "SqlType": "INTEGER",
                "Name": "consumption"
            }
        ],
        "RecordFormat": {
            "RecordFormatType": "CSV",
            "MappingParameters": {
                "CSVMappingParameters": {
                    "RecordRowDelimiter": "\r\n",
                    "RecordColumnDelimiter": ","
                }
            }
        }
    },
    "RawInputRecords": [
        "year,month,state,producer_type,energy_source,units,consumption
\r\n2001,1,AK,TotalElectricPowerIndustry,Coal,ShortTons,47615\r
\n2001,1,AK,ElectricGeneratorsElectricUtilities,Coal,ShortTons,16535\r
\n2001,1,AK,CombinedHeatandPowerElectricPower,Coal,ShortTons,22890\r
\n2001,1,AL,TotalElectricPowerIndustry,Coal,ShortTons,3020601\r
\n2001,1,AL,ElectricGeneratorsElectricUtilities,Coal,ShortTons,2987681"
    ],
    "ParsedInputRecords": [
        [
            null,
            null,
            "state",
            "producer_type",
            "energy_source",
            "units",
            null
        ],
        [
            "2001",
            "1",
            "AK",
            "TotalElectricPowerIndustry",
            "Coal",
            "ShortTons",
```

```
            "47615"
        ],
        [
            "2001",
            "1",
            "AK",
            "ElectricGeneratorsElectricUtilities",
            "Coal",
            "ShortTons",
            "16535"
        ],
        [
            "2001",
            "1",
            "AK",
            "CombinedHeatandPowerElectricPower",
            "Coal",
            "ShortTons",
            "22890"
        ],
        [
            "2001",
            "1",
            "AL",
            "TotalElectricPowerIndustry",
            "Coal",
            "ShortTons",
            "3020601"
        ],
        [
            "2001",
            "1",
            "AL",
            "ElectricGeneratorsElectricUtilities",
            "Coal",
            "ShortTons",
            "2987681"
        ]
    ]
}
```

Prétraitement des données à l’aide d’une fonction Lambda

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n’utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d’informations, consultez Limites .

Si les données de votre flux nécessitent une conversion de format, une transformation, un enrichissement ou un filtrage, vous pouvez les prétraiter à l'aide d'une AWS Lambda fonction. Vous pouvez effectuer cette opération avant que le code SQL de votre application s'exécute ou avant que votre application crée un schéma à partir de votre flux de données.

L’utilisation d’une fonction Lambda de prétraitement des enregistrements est utile dans les cas suivants :

- Transformation d’enregistrements à partir d’autres formats (comme KPL ou GZIP) en formats pouvant être analysés par Kinesis Data Analytics. Kinesis Data Analytics prend actuellement en charge les formats de données JSON ou CSV.
- Développement des données dans un format qui est plus accessible pour des opérations telles que le regroupement ou la détection des anomalies. Par exemple, si plusieurs valeurs de données sont stockées ensemble dans une chaîne, vous pouvez développer les données en colonnes distinctes.
- Enrichissement des données avec d’autres services Amazon, comme l’extrapolation ou la correction d’erreurs.
- Application de transformation de chaîne complexe à des champs d'enregistrement.
- Filtrage de données pour nettoyer les données.

L'utilisation d'une fonction Lambda pour le prétraitement des enregistrements

Lorsque vous créez votre application Kinesis Data Analytics, vous activez le prétraitement Lambda sur la page Se connecter à une source.

Pour utiliser une fonction Lambda pour prétraiter des enregistrements dans une application Kinesis Data Analytics

1. Connectez-vous à la console Managed Service for Apache Flink AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Sur la page Se connecter à une source pour votre application, choisissez Activé dans la section Prétraitement d'enregistrements avec AWS Lambda.
3. Pour utiliser une fonction Lambda que vous avez déjà créée, choisissez la fonction dans la liste déroulante Fonction Lambda.
4. Pour créer une nouvelle fonction Lambda à partir de l'un des modèles de prétraitement Lambda, choisissez le modèle dans la liste déroulante. Ensuite, choisissez Afficher <nom_modèle> dans Lambda pour modifier la fonction.
5. Pour créer une nouvelle fonction Lambda, choisissez Créer. Pour plus d'informations sur la création d'une fonction Lambda, consultez les sections Créer une fonction HelloWorld Lambda et Explorez la console dans le manuel du développeur.AWS Lambda
6. Choisissez la version de la fonction Lambda à utiliser. Pour utiliser la dernière version, choisissez $LATEST.

Lorsque vous choisissez ou créez une fonction Lambda pour le prétraitement d'enregistrements, les enregistrements sont prétraités avant que le code SQL de votre application ne s'exécute ou que votre application ne génère un schéma à partir des enregistrements.

Autorisations de prétraitement Lambda

Pour utiliser le prétraitement Lambda, le rôle IAM de l'application a besoin de la stratégie d'autorisations suivante :

```
    {
      "Sid": "UseLambdaFunction",
      "Effect": "Allow",
      "Action": [
          "lambda:InvokeFunction",
          "lambda:GetFunctionConfiguration"
      ],
      "Resource": "<FunctionARN>"
  }
```

Métriques de prétraitement Lambda

Vous pouvez utiliser Amazon CloudWatch pour surveiller le nombre d'appels Lambda, le nombre d'octets traités, les réussites et les échecs, etc. Pour plus d'informations sur CloudWatch les métriques émises par le prétraitement Lambda de Kinesis Data Analytics, consultez Amazon Kinesis Analytics Metrics.

Utilisation AWS Lambda avec la bibliothèque Kinesis Producer

La bibliothèque producteur Kinesis (KPL) regroupe de petits enregistrements formatés par l'utilisateur en enregistrements plus volumineux allant jusqu'à 1 Mo afin de mieux utiliser le débit Amazon Kinesis Data Streams. La bibliothèque Kinesis Client Library (KCL) pour Java prend en charge la désagrégation de ces enregistrements. Cependant, vous devez utiliser un module spécial pour désagréger les enregistrements lorsque vous les utilisez AWS Lambda en tant que consommateur de vos streams.

Pour obtenir le code de projet et les instructions nécessaires, consultez les modules de désagrégation de la bibliothèque Kinesis Producer pour plus d'informations. AWS LambdaGitHub Vous pouvez utiliser les composants de ce projet pour traiter des données sérialisées KPL AWS Lambda dans Java, Node.js et Python. Vous pouvez également utiliser ces composants dans le cadre d’une application KCL multi-lang.

Modèle de données d'entrée d'événement modèle de réponse d'enregistrement pour le prétraitement des données

Pour prétraiter des enregistrements, votre fonction Lambda doit être conforme aux modèles de données d’entrée d’événement et de réponse d’enregistrement imposés.

Modèle de données d'entrée d'événement

Kinesis Data Analytics lit en permanence les données de votre flux de données Kinesis ou de votre flux de diffusion Firehose. Pour chaque lot d’enregistrements qu’il récupère, le service gère la manière dont chaque lot est transmis à votre fonction Lambda. Votre fonction reçoit une liste d'enregistrements en entrée. Au sein de votre fonction, vous effectuez une itération dans la liste et vous appliquez votre logique métier pour réaliser vos exigences de prétraitement (par exemple, conversion du format de données ou enrichissement).

Le modèle d'entrée de votre fonction de prétraitement varie légèrement selon que les données proviennent d'un flux de données Kinesis ou d'un flux de diffusion Firehose.

Si la source est un flux de diffusion Firehose, le modèle de données d'entrée des événements est le suivant :

Modèle de données de demande Kinesis Data Firehose

Champ	Description
`invocationId`	ID d'appel Lambda (GUID aléatoire).
`applicationArn`	Amazon Resource Name (ARN) de l'application Kinesis Data Analytics
`streamArn`	ARN du flux de diffusion
enregistrements	

Champ	Description	
`recordId`	ID d'enregistrement (GUID aléatoire)	
`kinesisFirehoseRecordMetadata`	<table><tr><th>Champ</th><th>Description</th></tr><tr><td>`approximateArrivalTimestamp`</td><td>Heure d'arrivée approximative de l'enregistrement de flux de diffusion</td></tr></table>	
`data`	Charge utile d'enregistrement source codée en base64	

L'exemple suivant montre l'entrée d'un flux de diffusion Firehose :

```
{
   "invocationId":"00540a87-5050-496a-84e4-e7d92bbaf5e2",
   "applicationArn":"arn:aws:kinesisanalytics:us-east-1:12345678911:application/lambda-test",
   "streamArn":"arn:aws:firehose:us-east-1:AAAAAAAAAAAA:deliverystream/lambda-test",
   "records":[
```

```
      {
         "recordId":"495726722236655144228052469266569546309724860595358924 82",
         "data":"aGVsbG8gd29ybGQ=",
         "kinesisFirehoseRecordMetadata":{
            "approximateArrivalTimestamp":1520280173
         }
      }
   ]
}
```

Si la source est un flux de données Kinesis, le modèle de données d'entrée d'événement se présente comme suit :

Modèle de données de demande de flux Kinesis.

Champ	Description
`invocationId`	ID d'appel Lambda (GUID aléatoire).
`applicationArn`	ARN de l'application Kinesis Data Analytics
`streamArn`	ARN du flux de diffusion
enregistrements	

Champ	Description
`recordId`	ID d'enregistrement basé sur le numéro de séquence d'enregistrement Kinesis
`kinesisStreamRecordMetadata`	

Champ	Description
`sequenceNumber`	Numéro de séquence de l'enregistrement de flux Kinesis

Champ	Description
Champ	Description
Champ	Description
partitionKey	Clé de partition de l'enregistrement de flux Kinesis
shardId	ShardId de l'enregistrement de flux Kinesis
approximateArrivalTimestamp	Heure d'arrivée approximative de l'enregistrement de flux de diffusion
data	Charge utile d'enregistrement source codée en base64

L'exemple suivant montre l'entrée d'un flux de données Kinesis :

```
{
  "invocationId": "00540a87-5050-496a-84e4-e7d92bbaf5e2",
  "applicationArn": "arn:aws:kinesisanalytics:us-east-1:12345678911:application/lambda-test",
  "streamArn": "arn:aws:kinesis:us-east-1:AAAAAAAAAAAA:stream/lambda-test",
  "records": [
    {
      "recordId": "49572672223665514422805246926656954630972486059535892482",
      "data": "aGVsbG8gd29ybGQ=",
      "kinesisStreamRecordMetadata":{
            "shardId" :"shardId-000000000003",
            "partitionKey":"7400791606",

 "sequenceNumber":"49572672223665514422805246926656954630972486059535892482",
```

```
            "approximateArrivalTimestamp":1520280173
         }
      }
   ]
}
```

Modèle de réponse d'enregistrement

Tous les enregistrements renvoyés à partir de votre fonction de prétraitement Lambda (avec les ID d'enregistrement) qui sont envoyés à la fonction Lambda doit être renvoyés. Ils doivent contenir les paramètres suivants. Sinon, Kinesis Data Analytics les rejette et les traite comme un échec de prétraitement des données. La partie charge utile des données de l'enregistrement peut être transformée pour réaliser les exigences de prétraitement.

Modèle de données de réponse

enregistrements

Champ	Description
`recordId`	L'ID d'enregistrement est transmis depuis Kinesis Data Analytics vers Lambda pendant l'invocation. L'enregistrement transformé doit comporter le même ID d'enregistrement. La moindre incohérence entre l'ID de l'enregistrement initial et l'ID de l'enregistrement transformé est traitée comme un échec du prétraitement des données.
`result`	État de la transformation de données de l'enregistrement. Les valeurs possibles sont : • `Ok` : L'enregistrement a été transformé avec succès. Kinesis Data Analytics reçoit l'enregistrement pour le traitement SQL. • `Dropped` : L'enregistrement a été supprimé volontairement par votre logique de traitement. Kinesis Data Analytics supprime l'enregistrement du traitement SQL. Le champ de charge utile de données est facultatif pour un enregistrement `Dropped`.

Champ	Description
	• `ProcessingFailed` : L'enregistrement n'a pas pu être transformé. Kinesis Data Analytics le considère comme non traité avec succès par votre fonction Lambda et écrit une erreur dans le flux d'erreur. Pour plus d'informations sur le flux d'erreur, consultez Gestion des erreurs. Le champ de charge utile de données est facultatif pour un enregistr ement `ProcessingFailed` .
`data`	Charge utile des données transformées, d'après l'encodag e en base64. Chaque charge utile de données peut contenir plusieurs documents JSON si le format de données d'ingestion de l'application est JSON. Ou chaque charge utile de données peut contenir plusieurs lignes CSV (avec un délimiteur de ligne indiqué dans chaque ligne) si le format de données d'ingesti on de l'application est CSV. Le service Kinesis Data Analytics analyse et traite les données correctement avec plusieurs documents JSON ou lignes CSV dans la même charge utile de données.

L'exemple suivant montre la sortie d'une fonction Lambda :

```
{
  "records": [
    {
      "recordId": "49572672223665514422805246926656954630972486059535892482",
      "result": "Ok",
      "data": "SEVMTE8gV09STEQ="
    }
  ]
}
```

Échecs courants du prétraitement des données

Voici les raisons courantes pour lesquelles le prétraitement peut échouer.

- Tous les enregistrements (avec ID d'enregistrement) d'un lot qui sont envoyés à la fonction Lambda ne sont pas renvoyés au service Kinesis Data Analytics.
- L'ID d'enregistrement ou le champ de charge utile de données est manquant dans la réponse. Le champ de charge utile de données est facultatif pour un enregistrement `Dropped` ou `ProcessingFailed`.
- Les délais d'expiration de la fonction Lambda ne sont pas suffisants pour prétraiter les données.
- La réponse de la fonction Lambda dépasse les limites de réponse imposées par le service AWS Lambda .

En cas d'échec de prétraitement des données, Kinesis Data Analytics continue de relancer les invocations Lambda sur le même ensemble d'enregistrements jusqu'à ce que cela aboutisse. Vous pouvez surveiller les CloudWatch indicateurs suivants pour mieux comprendre les défaillances.

- `MillisBehindLatest` de l'application Kinesis Data Analytics : indique le retard d'une application pour la lecture de la source de streaming.
- Indicateurs de `InputPreprocessing` CloudWatch l'application Kinesis Data Analytics : indiquent le nombre de réussites et d'échecs, entre autres statistiques. Pour plus d'informations, consultez Métriques Amazon Kinesis Analytics.
- AWS Lambda CloudWatch métriques et journaux des fonctions.

Création de fonctions Lambda pour le prétraitement

Votre application Amazon Kinesis Data Analytics peut utiliser des fonctions Lambda pour le prétraitement des enregistrements à mesure que ceux-ci sont reçus dans l'application. Kinesis Data Analytics fournit dans la console les modèles suivants à utiliser comme point de départ pour le prétraitement de vos données.

Rubriques

- Création d'une fonction Lambda de prétraitement dans Node.js
- Création d'une fonction Lambda de prétraitement dans Python
- Création d'une fonction Lambda de prétraitement dans Java
- Création d'une fonction Lambda de prétraitement dans .NET

Création d’une fonction Lambda de prétraitement dans Node.js

Les modèles suivants pour créer une fonction Lambda de prétraitement dans Node.js sont disponibles dans la console Kinesis Data Analytics :

Plan Lambda	Langage et version	Description
Traitement d’entrée General Kinesis Data Analytics	Node.js 6.10	Un préprocesseur d’enregistrement Kinesis Data Analytics qui reçoit des enregistrements JSON ou CSV en entrée, puis les renvoie avec un statut de traitement. Utilisez ce processeur comme point de départ pour une logique de transformation personnalisée.
Traitement d'entrée compressée	Node.js 6.10	Un processeur d’enregistrements Kinesis Data Analytics qui reçoit des enregistrements JSON ou CSV compressés (GZIP ou compressés via Deflate) en entrée et renvoie des enregistrements décompressés avec un statut de traitement.

Création d’une fonction Lambda de prétraitement dans Python

Les modèles suivants pour créer une fonction Lambda de prétraitement dans Python sont disponibles dans la console :

Plan Lambda	Langage et version	Description
Traitement d'entrée General Kinesis Analytics	Python 2.7	Un préprocesseur d’enregistrement Kinesis Data Analytics qui reçoit des enregistrements JSON ou CSV en entrée, puis les renvoie avec un statut de traitement. Utilisez ce processeur comme point de départ pour une logique de transformation personnalisée.
Traitement d'entrée KPL	Python 2.7	Un processeur d’enregistrements Kinesis Data Analytics qui reçoit des regroupements

Plan Lambda	Langage et version	Description
		KPL (Kinesis Producer Library) d'enregis trements JSON ou CSV en entrée et renvoie des enregistrements désagrégés avec un statut de traitement.

Création d'une fonction Lambda de prétraitement dans Java

Pour créer une fonction Lambda dans Java pour prétraiter des enregistrements, utilisez les classes d'événements Java.

Le code suivant illustre un exemple de fonction Lambda qui prétraite des enregistrements à l'aide de Java :

```
public class LambdaFunctionHandler implements
        RequestHandler<KinesisAnalyticsStreamsInputPreprocessingEvent,
 KinesisAnalyticsInputPreprocessingResponse> {

    @Override
    public KinesisAnalyticsInputPreprocessingResponse handleRequest(
            KinesisAnalyticsStreamsInputPreprocessingEvent event, Context context) {
        context.getLogger().log("InvocatonId is : " + event.invocationId);
        context.getLogger().log("StreamArn is : " + event.streamArn);
        context.getLogger().log("ApplicationArn is : " + event.applicationArn);

        List<KinesisAnalyticsInputPreprocessingResponse.Record> records = new
 ArrayList<KinesisAnalyticsInputPreprocessingResponse.Record>();
        KinesisAnalyticsInputPreprocessingResponse response = new
 KinesisAnalyticsInputPreprocessingResponse(records);

        event.records.stream().forEach(record -> {
            context.getLogger().log("recordId is : " + record.recordId);
            context.getLogger().log("record aat is :" +
 record.kinesisStreamRecordMetadata.approximateArrivalTimestamp);
             // Add your record.data pre-processing logic here.

            // response.records.add(new Record(record.recordId,
 KinesisAnalyticsInputPreprocessingResult.Ok, <preprocessedrecordData>));
        });
        return response;
    }
```

```
}
```

Création d'une fonction Lambda de prétraitement dans .NET

Pour créer une fonction Lambda dans .NET pour prétraiter des enregistrements, utilisez les classes d'événements .NET.

Le code suivant illustre un exemple de fonction Lambda qui prétraite des enregistrements à l'aide de C# :

```
public class Function
    {
        public KinesisAnalyticsInputPreprocessingResponse
 FunctionHandler(KinesisAnalyticsStreamsInputPreprocessingEvent evnt, ILambdaContext
 context)
        {
            context.Logger.LogLine($"InvocationId: {evnt.InvocationId}");
            context.Logger.LogLine($"StreamArn: {evnt.StreamArn}");
            context.Logger.LogLine($"ApplicationArn: {evnt.ApplicationArn}");

            var response = new KinesisAnalyticsInputPreprocessingResponse
            {
                Records = new List<KinesisAnalyticsInputPreprocessingResponse.Record>()
            };

            foreach (var record in evnt.Records)
            {
                context.Logger.LogLine($"\tRecordId: {record.RecordId}");
                context.Logger.LogLine($"\tShardId: {record.RecordMetadata.ShardId}");
                context.Logger.LogLine($"\tPartitionKey:
 {record.RecordMetadata.PartitionKey}");
                context.Logger.LogLine($"\tRecord ApproximateArrivalTime:
 {record.RecordMetadata.ApproximateArrivalTimestamp}");
                context.Logger.LogLine($"\tData: {record.DecodeData()}");

                // Add your record preprocessig logic here.

                var preprocessedRecord = new
 KinesisAnalyticsInputPreprocessingResponse.Record
                {
                    RecordId = record.RecordId,
                    Result = KinesisAnalyticsInputPreprocessingResponse.OK
```

```
                };
                preprocessedRecord.EncodeData(record.DecodeData().ToUpperInvariant());
                response.Records.Add(preprocessedRecord);
            }
            return response;
        }
    }
```

Pour plus d'informations sur la création de fonctions Lambda pour le prétraitement et les destinations dans .NET, consultez `Amazon.Lambda.KinesisAnalyticsEvents`.

Mise en parallèle des flux d'entrée pour un débit accru

> **Note**
>
> Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Pour plus d'informations, consultez Limites .

Les applications Amazon Kinesis Data Analytics peuvent prendre en charge plusieurs flux d'entrée intégrés à l'application pour mettre une application à l'échelle au-delà du débit d'un flux d'entrée intégré. Pour plus d'informations sur les flux d'entrée intégrés à l'application, consultez Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement.

Dans presque tous les cas, Amazon Kinesis Data Analytics adapte votre application pour gérer la capacité des flux Kinesis ou des flux sources Firehose qui alimentent votre application. Toutefois, si le débit de votre flux source est supérieur au débit d'un flux d'entrée intégré à l'application, vous pouvez augmenter sensiblement le nombre de flux d'entrée que votre application utilise. Pour ce faire, vous devez utiliser le paramètre `InputParallelism`.

Lorsque le paramètre `InputParallelism` est supérieur à un, Amazon Kinesis Data Analytics répartit uniformément les partitions de votre flux source entre les flux intégrés à l'application. Par exemple, si votre flux source possède 50 partitions et que vous avez défini `InputParallelism` sur 2, chaque flux d'entrée intégré à l'application reçoit les entrées de 25 partitions du flux source.

Si vous augmentez le nombre de flux, votre application doit accéder explicitement aux données de chaque flux. Pour plus d'informations sur l'accès aux différents flux intégrés à l'application dans votre code, consultez Accès à plusieurs flux intégrés à l'application au sein de votre application Amazon Kinesis Data Analytics.

Bien que les fragments de flux Kinesis Data Streams et Firehose soient tous deux répartis de la même manière entre les flux intégrés à l'application, ils apparaissent différemment dans votre application :

- Les enregistrements d'un flux de données Kinesis incluent un champ `shard_id` qui peut servir à identifier la partition source de l'enregistrement.
- Les enregistrements d'un flux de diffusion Firehose n'incluent pas de champ identifiant la partition ou la partition source de l'enregistrement. Cela est dû au fait que Firehose extrait ces informations de votre application.

Evaluation de l'augmentation ou non de votre nombre de flux d'entrée intégrés à l'application

Dans la plupart des cas, un flux d'entrée intégré à l'application peut traiter le débit d'un flux source, selon la complexité et la taille des données des flux d'entrée. Pour déterminer s'il est nécessaire d'augmenter le nombre de flux d'entrée intégrés à l'application, vous pouvez surveiller les `MillisBehindLatest` métriques `InputBytes` et sur Amazon CloudWatch.

Si la métrique `InputBytes` est supérieure à 100 Mo/s (ou si vous pensez qu'elle sera supérieure à ce débit), elle peut provoquer une augmentation dans `MillisBehindLatest` et accroître l'impact des problèmes d'application. Pour résoudre ce problème, il est recommandé de faire les choix de langues suivants pour votre application :

- Utilisez plusieurs flux et applications Kinesis Data Analytics pour SQL si les besoins de mise à l'échelle de votre application sont supérieurs à 100 Mo/seconde.
- Utilisez les applications Kinesis Data Analytics pour Java si vous voulez utiliser un seul flux et une application.

Si la métrique `MillisBehindLatest` possède l'une des caractéristiques suivantes, vous devez augmenter le paramètre `InputParallelism` de votre application :

- La métrique `MillisBehindLatest` augmente progressivement, ce qui indique que votre application est en retard par rapport aux dernières données du flux.
- La métrique `MillisBehindLatest` est systématiquement supérieure à 1 000 (une seconde).

Vous n'avez pas besoin d'augmenter le paramètre `InputParallelism` de votre application si :

- La métrique `MillisBehindLatest` baisse progressivement, ce qui indique que votre application devance les dernières données du flux.
- La métrique `MillisBehindLatest` est systématiquement inférieure à 1 000 (une seconde).

Pour plus d'informations sur l'utilisation CloudWatch, consultez le guide de CloudWatch l'utilisateur.

Implémentation de plusieurs flux d'entrée intégrés à l'application

Vous pouvez définir le nombre de flux d'entrée intégrés à l'application lorsqu'une application est créée à l'aide de CreateApplication. Ce nombre est défini une fois que l'application a été créée à l'aide de UpdateApplication.

> **Note**
>
> Vous ne pouvez définir le paramètre `InputParallelism` qu'à l'aide de l'API Amazon Kinesis Data Analytics ou de l' AWS CLI. Vous ne pouvez pas définir ce paramètre à l'aide du AWS Management Console. Pour plus d'informations sur la configuration du AWS CLI, voirÉtape 2 : configurer le AWS Command Line Interface (AWS CLI).

Définition du nombre de flux d'entrée d'une nouvelle application

L'exemple suivant montre comment utiliser l'action d'API `CreateApplication` pour définir le nombre de flux d'entrée d'une nouvelle application sur 2.

Pour plus d'informations sur `CreateApplication`, consultez CreateApplication.

```
{
   "ApplicationCode": "<The SQL code the new application will run on the input
 stream>",
   "ApplicationDescription": "<A friendly description for the new application>",
   "ApplicationName": "<The name for the new application>",
   "Inputs": [
    {
      "InputId": "ID for the new input stream",
      "InputParallelism": {
        "Count": 2
    }],
   "Outputs": [ ... ],
 }]
```

```
}
```

Définition du nombre de flux d'entrée d'une application existante

L'exemple suivant montre comment utiliser l'action d'API `UpdateApplication` pour définir le nombre de flux d'entrée d'une application existante sur 2.

Pour plus d'informations sur `Update_Application`, consultez UpdateApplication.

```
{
   "InputUpdates": [
      {
         "InputId": "yourInputId",
         "InputParallelismUpdate": {
            "CountUpdate": 2
         }
      }
   ],
}
```

Accès à plusieurs flux intégrés à l'application au sein de votre application Amazon Kinesis Data Analytics

Pour utiliser plusieurs flux d'entrée intégrés à l'application au sein de votre application, vous devez sélectionner explicitement différents flux. L'exemple de code suivant montre comment interroger plusieurs flux d'entrée intégrés à l'application créés dans le didacticiel de mise en route.

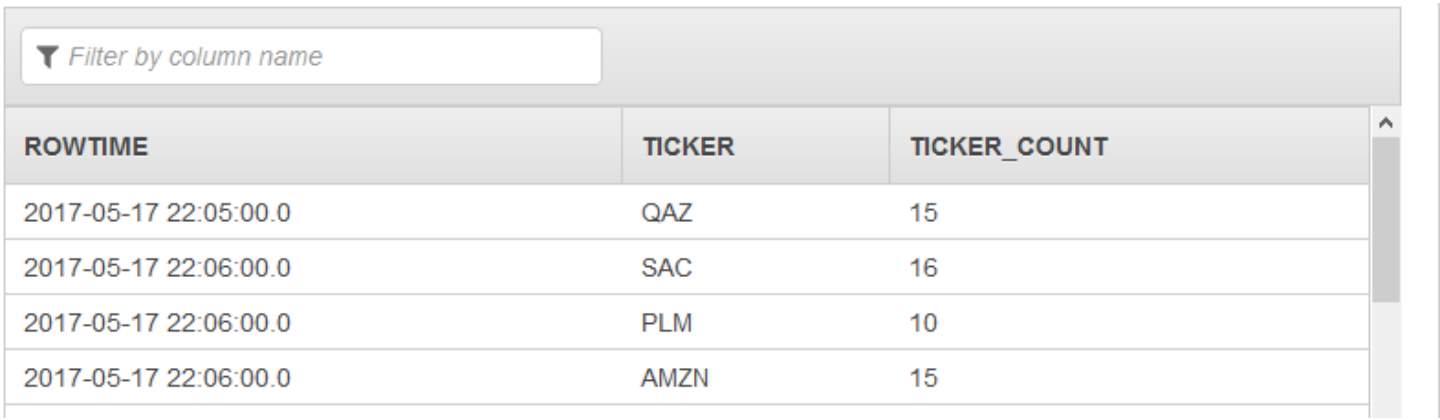
Dans l'exemple suivant, chaque flux source est d'abord agrégé avec COUNT avant d'être combiné en un seul flux intégré à l'application, appelé `in_application_stream001`. L'agrégation des flux source en amont vous permet de vous assurer que les flux combinés peuvent gérer le trafic de plusieurs flux sans être surchargés.

> **Note**
>
> Pour exécuter cet exemple et obtenir des résultats des deux flux d'entrée intégrés à l'application, vous devez mettre à jour le nombre de partitions dans votre flux source et le paramètre `InputParallelism` dans votre application.

```
CREATE OR REPLACE STREAM in_application_stream_001 (
```

```
    ticker VARCHAR(64),
    ticker_count INTEGER
);

CREATE OR REPLACE PUMP pump001 AS
INSERT INTO in_application_stream_001
SELECT STREAM ticker_symbol, COUNT(ticker_symbol)
FROM source_sql_stream_001
GROUP BY STEP(source_sql_stream_001.rowtime BY INTERVAL '60' SECOND),
    ticker_symbol;

CREATE OR REPLACE PUMP pump002 AS
INSERT INTO in_application_stream_001
SELECT STREAM ticker_symbol, COUNT(ticker_symbol)
FROM source_sql_stream_002
GROUP BY STEP(source_sql_stream_002.rowtime BY INTERVAL '60' SECOND),
    ticker_symbol;
```

L'exemple de code précédent produit dans `in_application_stream001` un résultat similaire à ce qui suit :



Considérations supplémentaires

Si vous utilisez plusieurs flux d'entrée, vous devez savoir que :

- Le nombre maximal de flux d'entrée intégrés à l'application est de 64.
- Les flux d'entrée intégrés à l'application sont répartis uniformément entre les partitions du flux d'entrée de l'application.
- Les gains de performances résultant de l'ajout de flux intégrés à l'application ne sont pas linéaires. Ainsi, le fait de doubler le nombre de flux intégrés à l'application ne double pas le débit. Avec

une taille de ligne classique, chaque flux intégré à l'application peut atteindre un débit de 5 000 à 15 000 lignes par seconde. En passant le nombre de flux intégrés à l'application à 10, vous pouvez obtenir un débit de 20 000 à 30 000 lignes par seconde. La vitesse du débit dépend du nombre, des types et de la taille des données des champs dans le flux d'entrée.

- Certaines fonctions d'agrégation (comme AVG) peuvent générer des résultats inattendus en cas de répartition des flux d'entrée entre plusieurs partitions. Etant donné que vous devez agréger toutes les partitions avant de les rassembler en un flux agrégé, les résultats peuvent être dirigés vers le flux contenant le plus d'enregistrements, quel qu'il soit.
- Si votre application continue d'offrir des performances médiocres (ce qui est reflété par une métrique `MillisBehindLatest` élevée) lorsque vous augmentez le nombre de flux d'entrée, vous avez peut-être atteint votre limite d'unités de traitement Kinesis (KPU). Pour de plus amples informations, veuillez consulter Dimensionnement automatique des applications pour augmenter le débit.

Code d'application

Le code d'application est un ensemble d'instructions SQL qui traite une entrée et produit une sortie. Ces instructions SQL fonctionnent sur des flux et des tables de référence intégrés à l'application. Pour de plus amples informations, veuillez consulter Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement.

Pour plus d'informations sur les éléments du langage SQL pris en charge par Kinesis Data Analytics, consultez la Référence SQL Amazon Kinesis Data Analytics.

Dans les bases de données relationnelles, vous gérez des tables à l'aide d'instructions INSERT pour ajouter des enregistrements et de l'instruction SELECT pour interroger les données. Dans Amazon Kinesis Data Analytics, vous travaillez avec des flux. Vous pouvez écrire une instruction SQL pour interroger ces flux. Les résultats de l'interrogation d'un flux intégré à l'application sont toujours envoyés à un autre flux intégré à l'application. Lorsque vous effectuez des analyses complexes, vous pouvez créer plusieurs flux intégrés à l'application pour conserver les résultats d'analyses intermédiaires. Enfin, vous configurez la sortie d'application pour conserver les résultats de l'analyse finale (à partir d'un ou de plusieurs flux intégrés à l'application) dans des destinations externes. Pour résumer, voici un modèle typique pour écrire du code d'application :

- L'instruction SELECT est toujours utilisée dans le contexte d'une instruction INSERT. En d'autres termes, lorsque vous sélectionnez des lignes, vous insérez des résultats dans un autre flux intégré à l'application.

- L'instruction INSERT est toujours utilisée dans le contexte d'une pompe. En d'autres termes, vous utilisez des pompes pour écrire dans un flux intégré à l'application.

L'exemple de code d'application suivant lit des enregistrements d'un flux intégré à l'application (SOURCE_SQL_STREAM_001) et écrit dans un autre flux intégré à l'application (DESTINATION_SQL_STREAM). Vous pouvez insérer des enregistrements dans des flux intégrés à l'application à l'aide de pompes, comme illustré ci-après :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ticker_symbol VARCHAR(4),
                                                   change DOUBLE,
                                                   price DOUBLE);
-- Create a pump and insert into output stream.
CREATE OR REPLACE PUMP "STREAM_PUMP" AS

  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM ticker_symbol, change,price
    FROM   "SOURCE_SQL_STREAM_001";
```

> **Note**
>
> Les identificateurs que vous spécifiez pour les noms de flux et de colonne suivent les conventions SQL standard. Par exemple, si vous placez un identificateur entre guillemets, celui-ci sera sensible à la casse. Sinon, l'identificateur apparaît en majuscules par défaut. Pour plus d'informations sur les identifiants, consultez la section Identifiants dans le manuel Référence SQL du service géré Amazon pour Apache Flink SQL.

Votre code d'application peut être constitué de nombreuses instructions SQL. Par exemple :

- Vous pouvez écrire des requêtes SQL de manière séquentielle de sorte que le résultat d'une instruction SQL soit intégré dans l'instruction SQL suivante.
- Vous pouvez également écrire des requêtes SQL qui s'exécutent indépendamment les unes des autres. Par exemple, vous pouvez écrire deux instructions SQL qui interrogent le même flux intégré à l'application, mais qui envoient une sortie dans des flux intégrés à l'application différents. Vous pouvez ensuite interroger indépendamment les flux intégrés à l'application nouvellement créés.

Vous pouvez créer des flux intégrés à l'application pour enregistrer des résultats intermédiaires. Vous insérez les données dans des flux intégrés à l'application à l'aide de pompes. Pour de plus amples informations, veuillez consulter Flux et pompes intégrés à l'application.

Si vous ajoutez une table de référence intégrée à l'application, vous pouvez écrire du code SQL pour joindre des données de flux et de tables de référence intégrés à l'application. Pour de plus amples informations, veuillez consulter Exemple : ajout de données de référence à une application Kinesis Data Analytics.

Selon la configuration de sortie de l'application, Amazon Kinesis Data Analytics écrit les données de flux intégrés à l'application spécifiques dans la destination externe. Assurez-vous que votre code d'application écrit dans les flux intégrés à l'application spécifiés dans la configuration de sortie.

Pour plus d'informations, consultez les rubriques suivantes :

- Concepts du code SQL de streaming
- Référence SQL Amazon Kinesis Data Analytics

Configuration de la sortie d'application

Dans le code de votre application, vous écrivez la sortie d'instructions SQL dans un ou plusieurs flux intégrés à l'application. Vous pouvez éventuellement ajouter une configuration de sortie à votre application pour conserver tout ce qui est écrit dans un flux intégré à l'application vers une destination externe, telle qu'un flux de données Amazon Kinesis, un flux de diffusion Firehose ou une fonction. AWS Lambda

Le nombre de destinations externes que vous pouvez utiliser pour conserver la sortie d'une application est limité. Pour de plus amples informations, veuillez consulter Limites.

> **Note**
>
> Nous vous recommandons d'utiliser une destination externe pour conserver les données du flux d'erreurs intégré à l'application pour pouvoir examiner les erreurs.

Dans chacune de ces configurations de sortie, vous fournissez les éléments suivants :

- Nom du flux intégré à l'application : flux que vous souhaitez conserver dans une destination externe.

  Kinesis Data Analytics recherche le flux intégré à l'application que vous avez spécifié dans la configuration de sortie. (Le nom du flux est sensible à la casse et doit correspondre exactement). Assurez-vous que votre code d'application crée ce flux intégré à l'application.
- Destination externe : vous pouvez conserver les données dans un flux de données Kinesis, un flux de diffusion Firehose ou une fonction Lambda. Vous fournissez le nom Amazon Resource Name (ARN) du flux ou de la fonction. Vous pouvez également fournir un rôle IAM que Kinesis Data Analytics peut endosser pour écrire dans le flux ou la fonction en votre nom. Vous décrivez également le format d'enregistrement (JSON, CSV) que Kinesis Data Analytics doit utiliser lors de l'écriture dans la destination externe.

Si Kinesis Data Analytics ne peut pas écrire dans la destination de streaming ou Lambda, il continue d'essayer indéfiniment. Cela crée une pression de retour qui peut amener votre application à prendre du retard. Si ce problème n'est pas résolu, votre application finira par arrêter de traiter de nouvelles données. Vous pouvez surveiller les Métriques de données Kinesis Analytics et définir des alarmes d'échec. Pour plus d'informations sur les métriques et les alarmes, consultez les sections Utilisation d'Amazon CloudWatch Metrics et Création d' CloudWatchalarmes Amazon.

Vous pouvez configurer la sortie d'application à l'aide de l' AWS Management Console. La console effectue l'appel d'API pour enregistrer la configuration.

Création d'une sortie à l'aide du AWS CLI

Cette section explique comment créer la section `Outputs` du corps de la requête pour une opération `CreateApplication` ou `AddApplicationOutput`.

Création d'une sortie de flux Kinesis

Le fragment JSON suivant montre la section `Outputs` dans le corps de la requête `CreateApplication` qui permet de créer une destination de flux de données Amazon Kinesis.

```
"Outputs": [
   {
       "DestinationSchema": {
           "RecordFormatType": "string"
       },
```

```
        "KinesisStreamsOutput": {
            "ResourceARN": "string",
            "RoleARN": "string"
        },
        "Name": "string"
    }

]
```

Création d'une sortie Firehose Delivery Stream

Le fragment JSON suivant montre la `Outputs` section du corps de la `CreateApplication` demande permettant de créer une destination de flux de diffusion Amazon Data Firehose.

```
"Outputs": [
    {
        "DestinationSchema": {
            "RecordFormatType": "string"
        },
        "KinesisFirehoseOutput": {
            "ResourceARN": "string",
            "RoleARN": "string"
        },
        "Name": "string"
    }
]
```

Création d’une sortie de fonction Lambda

Le fragment JSON suivant montre la `Outputs` section du corps de la `CreateApplication` demande permettant de créer une destination de AWS Lambda fonction.

```
"Outputs": [
    {
        "DestinationSchema": {
            "RecordFormatType": "string"
        },
        "LambdaOutput": {
            "ResourceARN": "string",
            "RoleARN": "string"
        },
        "Name": "string"
```

```
    }
]
```

Utilisation d'une fonction Lambda en tant que sortie

L'utilisation AWS Lambda comme destination vous permet d'effectuer plus facilement le post-traitement de vos résultats SQL avant de les envoyer vers une destination finale. Les tâches de post-traitement courantes sont les suivantes :

- Agrégation de plusieurs lignes dans un seul enregistrement
- Combinaison des résultats actuels avec des résultats antérieurs afin de mieux gérer les données tardives
- Diffusion vers différentes destinations en fonction du type d'information
- Traduction des formats d'enregistrement (par exemple, traduction en Protobuf)
- Manipulation ou transformation des chaînes
- Enrichissement des données après traitement analytique
- Traitement personnalisé pour les cas d'utilisation de géolocalisation
- Chiffrement des données

Les fonctions Lambda peuvent fournir des informations analytiques à divers AWS services et à d'autres destinations, notamment les suivantes :

- Amazon Simple Storage Service (Amazon S3)
- API personnalisées
- Amazon DynamoDB
- Apache Aurora
- Amazon Redshift
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon CloudWatch

Pour plus d’informations sur la création d’applications Lambda, consultez Mise en route avec AWS Lambda.

Rubriques

Autorisations de la fonction Lambda utilisée en tant que sortie

Pour utiliser Lambda comme sortie, le rôle IAM de sortie Lambda de l’application a besoin de la stratégie d’autorisations suivante :

```
{
   "Sid": "UseLambdaFunction",
   "Effect": "Allow",
   "Action": [
       "lambda:InvokeFunction",
       "lambda:GetFunctionConfiguration"
   ],
   "Resource": "FunctionARN"
}
```

Métriques de la fonction Lambda utilisée en tant que sortie

Vous utilisez Amazon CloudWatch pour surveiller le nombre d'octets envoyés, les réussites et les échecs, etc. Pour plus d'informations sur CloudWatch les métriques émises par Kinesis Data Analytics à l'aide de Lambda comme sortie, consultez Amazon Kinesis Analytics Metrics.

Modèle de données d'entrée d'événement et modèle de réponse d'enregistrement pour une fonction Lambda utilisée comme sortie

Pour envoyer des enregistrements de sortie Kinesis Data Analytics, votre fonction Lambda doit être conforme aux modèles de données d’entrée d’événement et de réponse d’enregistrement imposés.

Modèle de données d'entrée d'événement

Kinesis Data Analytics envoie continuellement les enregistrements de sortie de l'application à la fonction Lambda en tant que sortie en suivant le modèle de requête ci-dessous. Dans votre fonction, vous effectuez une itération dans la liste et appliquez votre logique métier pour réaliser vos exigences de sortie (par exemple, transformation des données avant envoi vers une destination finale).

Champ	Description
`invocationId`	ID d'appel Lambda (GUID aléatoire).
`applicationArn`	Amazon Resource Name (ARN) de l'application Kinesis Data Analytics.
enregistrements	

Champ	Description
`recordId`	ID d'enregistrement (GUID aléatoire)
`lambdaDeliveryRecordMetadata`	

Champ	Description
`retryHint`	Nombre de nouvelles tentatives de diffusion

data	Charge utile d'enregistrement de sortie codée en base64

> **Note**
>
> Le `retryHint` est une valeur qui augmente à chaque échec de diffusion. Cette valeur n'est pas conservée durablement et est réinitialisée en cas d'interruption de l'application.

Modèle de réponse d'enregistrement

Chaque enregistrement envoyé à votre fonction Lambda en tant que sortie (avec des ID d'enregistrements) doit être confirmé par `Ok` ou `DeliveryFailed` et doit contenir les paramètres suivants. À défaut, Kinesis Data Analytics les traite comme un échec de diffusion.

enregistrements

Champ	Description
`recordId`	L'ID d'enregistrement est transmis depuis Kinesis Data Analytics vers Lambda pendant l'invocation. La moindre incohérence entre l'ID de l'enregistrement initial et l'ID de l'enregistrement conformé est traitée comme un échec de diffusion.
`result`	État de la diffusion de l'enregistrement. Les valeurs admises sont les suivantes : • `Ok` : l'enregistrement a bien été transformé et envoyé vers la destination finale. Kinesis Data Analytics reçoit l'enregis trement pour le traitement SQL. • `DeliveryFailed` : l'enregistrement n'a pas pu être envoyé vers la destination finale par la fonction Lambda en tant que sortie. Kinesis Data Analytics tente en continu de renvoyer les enregistrements qui n'ont pas pu être diffusés vers la fonction Lambda en tant que sortie.

Fréquence d'appel de sortie Lambda

Une application Kinesis Data Analytics place les enregistrements de sortie dans la mémoire tampon et appelle fréquemment la fonction de destination AWS Lambda .

- Si des enregistrements sont émis vers le flux intégré à l'application de destination au sein de l'application d'analyse de données sous forme de fenêtre de basculement, la fonction de AWS Lambda destination est invoquée par déclencheur de fenêtre de basculement. Par exemple, si une

fenêtre bascule de 60 secondes est utilisée pour transmettre les enregistrements vers le flux de destination intégré à l'application, la fonction Lambda est appelée une fois toutes les 60 secondes.

- Si des enregistrements sont émis vers le flux de destination intégré à l'application au sein de l'application sous la forme d'une requête continue ou d'une fenêtre défilante, la fonction Lambda de destination est appelée environ une fois par seconde.

> **Note**
>
> Les limites de taille de la charge utile pour les demandes d'invocation par fonction Lambda s'appliquent. Si ces limites sont dépassées, les enregistrements de sortie sont fractionnés et envoyés sur plusieurs appels de fonction Lambda.

Ajout d'une fonction Lambda pour une utilisation en tant que sortie

La procédure suivante explique comment ajouter une fonction Lambda en tant que sortie pour une application Kinesis Data Analytics.

1. Connectez-vous à la console Managed Service for Apache Flink AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez l'application dans la liste, puis sélectionnez Application details.
3. Dans la section Destination, choisissez Connect new destination.
4. Pour l'élément Destination, choisissez la fonctionAWS Lambda .
5. Dans la section Diffuser des enregistrements vers AWS Lambda, choisissez une fonction et une version Lambda existantes ou cliquez sur Créer.
6. Si vous créez une nouvelle fonction Lambda, procédez comme suit :

    a. Choisissez l'un des modèles fournis. Pour plus d'informations, veuillez consulter Création de fonctions Lambda pour des destinations d'application.

    b. La page Create Function (Créer une fonction) s'ouvre dans un nouvel onglet du navigateur web. Dans la zone Nom, attribuez à la fonction un nom significatif (par exemple, **`myLambdaFunction`**).

    c. Mettez à jour le modèle avec la fonctionnalité de post-traitement pour votre application. Pour plus d'informations sur la création d'une fonction Lambda, consultez Mise en route dans le Manuel du développeurAWS Lambda .

d. Dans la console Kinesis Data Analytics, dans la liste de fonctions Lambda, choisissez la fonction Lambda que vous venez de créer. Sélectionnez $LATEST pour la version de la fonction Lambda.

7. Dans la section In-application stream, sélectionnez Choose an existing in-application stream. Dans le champ In-application stream name, choisissez le flux de sortie de votre application. Les résultats du flux de sortie sélectionné sont envoyés à la fonction de sortie Lambda.
8. Conservez les valeurs par défaut pour les autres champs du formulaire, puis cliquez sur Save and continue.

Votre application envoie désormais à votre fonction Lambda les enregistrements du flux intégré à l'application. Vous pouvez consulter les résultats du modèle par défaut dans la CloudWatch console Amazon. Surveillez la métrique `AWS/KinesisAnalytics/LambdaDelivery.OkRecords` pour voir le nombre d'enregistrements envoyés à la fonction Lambda.

Échecs courants associés à l'utilisation de Lambda en tant que sortie

Voici quelques raisons courantes qui peuvent expliquer un échec de diffusion vers une fonction Lambda.

- Tous les enregistrements (avec ID d'enregistrement) d'un lot qui sont envoyés à la fonction Lambda ne sont pas renvoyés au service Kinesis Data Analytics.
- L'ID d'enregistrement ou le champ d'état est manquant dans la réponse.
- Les délais d'expiration de la fonction Lambda ne sont pas suffisants pour réaliser la logique métier au sein de la fonction Lambda.
- La logique métier au sein de la fonction Lambda ne repère pas toutes les erreurs, ce qui se traduit par une expiration du délai d'attente et par une pression de retour en raison d'exceptions non traitées. C'est ce que l'on appelle communément les messages « poison pill ».

En cas d'échec de diffusion des données, Kinesis Data Analytics continue de relancer les invocations Lambda sur le même ensemble d'enregistrements jusqu'à ce que cela aboutisse. Pour mieux comprendre les défaillances, vous pouvez surveiller les CloudWatch indicateurs suivants :

- L'application Lambda de l'application Kinesis Data Analytics en tant que métriques CloudWatch de sortie : indique le nombre de réussites et d'échecs, entre autres statistiques. Pour plus d'informations, consultez Métriques Amazon Kinesis Analytics.
- AWS Lambda CloudWatch métriques et journaux des fonctions.

Création de fonctions Lambda pour des destinations d'application

Votre application Kinesis Data Analytics peut AWS Lambda utiliser des fonctions comme sortie. Kinesis Data Analytics fournit des modèles de création de fonctions Lambda qui peuvent être utilisées comme destination pour vos applications. Utilisez ces modèles comme point de départ pour la sortie de post-traitement de votre application.

Rubriques

Création d'une destination de fonction Lambda dans Node.js

Le modèle suivant pour créer une fonction Lambda de destination dans Node.js est disponible dans la console :

Plan de la fonction Lambda utilisée en tant que sortie	Langage et version	Description
`kinesis-analytics-output`	Node.js 12.x	Envoyez des enregistrements de sortie d'une application Kinesis Data Analytics vers une destination personnalisée.

Création d'une destination de fonction Lambda dans Python

Les modèles suivants pour créer une fonction Lambda de destination dans Python sont disponibles dans la console :

Plan de la fonction Lambda utilisée en tant que sortie	Langage et version	Description
`kinesis-analytics-output-sns`	Python 2.7	Envoyez des enregistrements de sortie d'une application

Plan de la fonction Lambda utilisée en tant que sortie	Langage et version	Description
		Kinesis Data Analytics vers Amazon SNS.
`kinesis-analytics-output-ddb`	Python 2.7	Envoyez des enregistrements de sortie d'une application Kinesis Data Analytics vers Amazon DynamoDB.

Création d'une destination de fonction Lambda dans Java

Pour créer une fonction Lambda de destination dans Java, utilisez les classes d'événements Java.

Le code suivant illustre un exemple de fonction Lambda de destination utilisant Java :

```
public class LambdaFunctionHandler
        implements RequestHandler<KinesisAnalyticsOutputDeliveryEvent,
 KinesisAnalyticsOutputDeliveryResponse> {

    @Override
    public KinesisAnalyticsOutputDeliveryResponse
 handleRequest(KinesisAnalyticsOutputDeliveryEvent event,
            Context context) {
        context.getLogger().log("InvocatonId is : " + event.invocationId);
        context.getLogger().log("ApplicationArn is : " + event.applicationArn);

        List<KinesisAnalyticsOutputDeliveryResponse.Record> records = new
 ArrayList<KinesisAnalyticsOutputDeliveryResponse.Record>();
        KinesisAnalyticsOutputDeliveryResponse response = new
 KinesisAnalyticsOutputDeliveryResponse(records);

        event.records.stream().forEach(record -> {
            context.getLogger().log("recordId is : " + record.recordId);
            context.getLogger().log("record retryHint is :" +
 record.lambdaDeliveryRecordMetadata.retryHint);
            // Add logic here to transform and send the record to final destination of
 your choice.
            response.records.add(new Record(record.recordId,
 KinesisAnalyticsOutputDeliveryResponse.Result.Ok));
        });
```

```
        return response;
    }

}
```

Création d'une destination de fonction Lambda dans .NET

Pour créer une fonction Lambda de destination dans .NET, utilisez les classes d'événements .NET.

Le code suivant illustre un exemple de fonction Lambda de destination utilisant C# :

```
public class Function
    {
        public KinesisAnalyticsOutputDeliveryResponse
 FunctionHandler(KinesisAnalyticsOutputDeliveryEvent evnt, ILambdaContext context)
        {
            context.Logger.LogLine($"InvocationId: {evnt.InvocationId}");
            context.Logger.LogLine($"ApplicationArn: {evnt.ApplicationArn}");

            var response = new KinesisAnalyticsOutputDeliveryResponse
            {
                Records = new List<KinesisAnalyticsOutputDeliveryResponse.Record>()
            };

            foreach (var record in evnt.Records)
            {
                context.Logger.LogLine($"\tRecordId: {record.RecordId}");
                context.Logger.LogLine($"\tRetryHint:
 {record.RecordMetadata.RetryHint}");
                context.Logger.LogLine($"\tData: {record.DecodeData()}");

                // Add logic here to send to the record to final destination of your
 choice.

                var deliveredRecord = new KinesisAnalyticsOutputDeliveryResponse.Record
                {
                    RecordId = record.RecordId,
                    Result = KinesisAnalyticsOutputDeliveryResponse.OK
                };
                response.Records.Add(deliveredRecord);
            }
            return response;
        }
```

```
    }
```

Pour plus d'informations sur la création de fonctions Lambda pour le prétraitement et les destinations dans .NET, consultez `Amazon.Lambda.KinesisAnalyticsEvents`.

Modèle de diffusion pour la conservation de la sortie d'application dans une destination externe

Amazon Kinesis Data Analytics utilise un modèle de diffusion « au moins une fois » pour la sortie d'application vers les destinations configurées. Lorsqu'une application est en cour d'exécution, Kinesis Data Analytics prend des points de contrôle internes. Ces points de contrôle sont des points dans le temps où les enregistrements de sortie ont été envoyés vers les destinations sans perte de données. Le service utilise les points de contrôle si nécessaire pour s'assurer que la sortie de votre application est envoyée au moins une fois aux destinations configurées.

Dans une situation normale, votre application traite les données entrantes en continu. Kinesis Data Analytics écrit la sortie sur les destinations configurées, telles qu'un flux de données Kinesis ou un flux de diffusion Firehose. Toutefois, votre application peut parfois être interrompue, par exemple :

- Vous choisissez d'arrêter votre application et de la redémarrer ultérieurement.
- Vous supprimez le rôle IAM dont Kinesis Data Analytics a besoin pour écrire la sortie de votre application vers la destination configurée. Sans le rôle IAM, Kinesis Data Analytics n'est pas autorisé à écrire vers la destination externe en votre nom.
- Une indisponibilité du réseau ou d'autres défaillances de service internes entraînent l'arrêt temporaire de votre application.

Lorsque votre application redémarre, Kinesis Data Analytics veille à ce qu'elle continue de traiter et d'écrire la sortie à partir d'un point avant ou correspondant au moment où la défaillance s'est produite. Cela permet de garantir que rien ne manque dans la sortie de votre application envoyée vers les destinations configurées.

Supposons que vous avez configuré plusieurs destinations depuis le même flux intégré à l'application. Une fois que l'application s'est rétablie de la défaillance, Kinesis Data Analytics reprend l'envoi de la sortie vers les destinations configurées pour conservation à partir du dernier enregistrement qui a été envoyé vers la destination plus lente. De ce fait, le même enregistrement de sortie peut être envoyé plusieurs fois vers d'autres destinations. Dans ce cas, vous devez traiter les duplications potentielles dans la destination en externe.

Gestion des erreurs

Amazon Kinesis Data Analytics vous renvoie les erreurs d'API ou de code SQL directement. Pour plus d'informations sur les opérations d'API, consultez Actions. Pour plus d'informations sur la gestion des erreurs SQL, consultez le manuel Référence SQL Amazon Kinesis Data Analytics.

Amazon Kinesis Data Analytics signale les erreurs d'exécution à l'aide d'un flux d'erreurs intégré à l'application appelé `error_stream`.

Signalement des erreurs à l'aide d'un flux d'erreurs intégré à l'application

Amazon Kinesis Data Analytics signale les erreurs d'exécution dans le flux d'erreurs intégré à l'application appelé `error_stream`. Voici quelques exemples d'erreurs possibles :

- Une lecture d'enregistrement à partir de la source de diffusion n'est pas conforme au schéma d'entrée.
- Le code de votre application spécifie une division par zéro.
- Les lignes sont ne sont pas dans l'ordre (par exemple, un enregistrement s'affiche dans le flux avec une valeur `ROWTIME` modifiée par l'utilisateur à cause de laquelle un enregistrement n'est plus dans l'ordre).
- Les données du flux source ne peuvent être converties dans le type de données spécifié dans le schéma (erreur de forçage du type). Pour plus d'informations sur les types de données convertibles, consultez Mappage de types de données JSON à des types de données SQL.

Nous vous recommandons de gérer ces erreurs par programmation dans votre code SQL ou de conserver les données dans le flux d'erreurs sur une destination externe. Cela nécessite que vous ajoutiez la configuration de sortie (voir Configuration de la sortie d'application) à votre application. Pour accéder à un exemple de fonctionnement du flux d'erreurs intégré à l'application, consultez Exemple : Exploration du flux d'erreurs intégré à l'application.

> **Note**
>
> Votre application Kinesis Data Analytics ne peut pas accéder ou modifier le flux d'erreurs par programmation, car le flux d'erreurs est créé à l'aide du compte système. Vous devez utiliser la sortie d'erreur pour déterminer les erreurs que votre application peut rencontrer.

Vous écrivez ensuite le code SQL de votre application afin de gérer les conditions d'erreur anticipées.

Schéma du flux d'erreurs

Le flux d'erreurs présente le schéma suivant :

Champ	Type de données	Remarques
ERROR_TIME	TIMESTAMP	Heure à laquelle l'erreur s'est produite
ERROR_LEVEL	VARCHAR(10)	
ERROR_NAME	VARCHAR(32)	
MESSAGE	VARCHAR(4096)	
DATA_ROWTIME	TIMESTAMP	Valeur rowtime de l'enregis trement entrant
DATA_ROW	VARCHAR(49152)	Données encodées en hexadécimal dans la ligne d'origine. Vous pouvez utiliser des bibliothèques standard pour décoder cette valeur en hexadécodage ou des ressources Web telles que ce convertisseur du format hexadécimal vers le format chaîne.
PUMP_NAME	VARCHAR(128)	Pompe d'origine, telle que définie avec `CREATE PUMP`

Dimensionnement automatique des applications pour augmenter le débit

Amazon Kinesis Data Analytics effectue une mise à l'échelle basée sur Elastic de votre application pour répondre au débit des données de votre flux source et à la complexité de la requête pour la plupart des scénarios. Kinesis Data Analytics alloue de la capacité sous la forme d'unités de traitement Kinesis (KPU). Une seule unité KPU vous fournit la mémoire (4 GB), ainsi que les ressources de calcul et de mise en réseau correspondantes.

La limite par défaut est de 64 pour les KPU pour votre application. Pour trouver des instructions pour demander une augmentation de cette limite, consultez Pour demander une augmentation de limite dans Limites relatives au service Amazon.

Utilisation du balisage

Cette section décrit comment ajouter des balises de métadonnées clé-valeur à des applications Kinesis Data Analytics. Ces balises peuvent être utilisées aux fins suivantes :

- Déterminer la facturation pour les applications Kinesis Data Analytics individuelles. Pour plus d'informations, consultez Utilisation des balises de répartition des coûts dans le Guide de l'utilisateur AWS Billing and Cost Management.
- Contrôle de l'accès aux ressources d'application basé sur des balises. Pour de plus amples informations, veuillez consulter Contrôle de l'accès à l'aide de balises dans le Guide de l'utilisateur.
- À des fins définies par l'utilisateur. Vous pouvez définir des fonctionnalités d'application en fonction de la présence de balises utilisateur.

Notez les informations suivantes concernant le balisage :

- Le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50.
- Si une action inclut une liste de balises qui comporte des valeurs `Key` en double, le service émet une `InvalidArgumentException`.

Cette rubrique contient les sections suivantes :

- Ajout de balises lorsqu'une application est créée

- Ajout ou mise à jour des balises pour une application existante
- Répertorier les balises d'une application
- Suppression des balises d'une application

Ajout de balises lorsqu'une application est créée

Vous pouvez ajouter des balises lors de la création d'une application à l'aide du paramètre `tags` de l'action CreateApplication.

L'exemple de demande suivant illustre le nœud `Tags` pour une demande `CreateApplication` :

```
"Tags": [
    {
        "Key": "Key1",
        "Value": "Value1"
    },
    {
        "Key": "Key2",
        "Value": "Value2"
    }
]
```

Ajout ou mise à jour des balises pour une application existante

Vous pouvez ajouter des balises à une application à l'aide de l'action TagResource. Vous ne pouvez pas ajouter de balises à une application à l'aide de l'action UpdateApplication.

Pour mettre à jour une balise existante, ajoutez une balise avec la même clé que la balise existante.

L'exemple de demande suivant pour l'action `TagResource` ajoute de nouvelles balises ou met à jour des balises existantes :

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "NewTagKey",
         "Value": "NewTagValue"
      },
      {
```

```
         "Key": "ExistingKeyOfTagToUpdate",
         "Value": "NewValueForExistingTag"
      }
   ]
}
```

Répertorier les balises d'une application

Pour répertorier les balises existantes, utilisez l'action ListTagsForResource.

L'exemple de demande suivant pour l'action `ListTagsForResource` répertorie les balises pour une application :

```
{
   "ResourceARN": "arn:aws:kinesisanalytics:us-west-2:012345678901:application/
MyApplication"
}
```

Suppression des balises d'une application

Pour supprimer des balises d'une application, vous devez utiliser l'action UntagResource.

L'exemple de requête suivant pour l'action `UntagResource` supprime des balises d'une application :

```
{
   "ResourceARN": "arn:aws:kinesisanalytics:us-west-2:012345678901:application/
MyApplication",
   "TagKeys": [ "KeyOfFirstTagToRemove", "KeyOfSecondTagToRemove" ]
}
```

Mise en route avec les applications Amazon Kinesis Data Analytics pour SQL

Vous trouverez ci-dessous des rubriques pour vous aider à vous familiariser avec les applications Amazon Kinesis Data Analytics pour SQL. Si vous n'êtes pas encore familiarisé avec les applications Kinesis Data Analytics pour SQL, nous vous recommandons de consulter les concepts et la terminologie présentés dans Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement avant d'exécuter les étapes de la section Mise en route.

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 https://portal.aws.amazon.com/billing/signup.
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. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et

utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

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

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif 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 racine.

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

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.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée 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'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Étape 1 : configuration d'un compte et création d'un administrateur

Avant d'utiliser Amazon Kinesis Data Analytics pour la première fois, exécutez les tâches suivantes :

1. Inscrivez-vous pour AWS
2. Créer un utilisateur IAM

Inscrivez-vous pour AWS

Lorsque vous vous inscrivez à Amazon Web Services, vous êtes automatiquement Compte AWS inscrit à tous les services AWS, y compris Amazon Kinesis Data Analytics. Seuls les services que vous utilisez vous sont facturés.

Avec Kinesis Data Analytics, vous ne payez que les ressources que vous utilisez. Si vous êtes un nouveau AWS client, vous pouvez commencer à utiliser Kinesis Data Analytics gratuitement. Pour plus d'informations, consultez la page Niveau d'offre gratuite d'AWS.

Si vous en avez déjà un Compte AWS, passez à la tâche suivante. Si vous n'en avez pas Compte AWS, suivez les étapes de la procédure suivante pour en créer un.

Pour créer un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
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. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

Notez votre Compte AWS identifiant car vous en aurez besoin pour la prochaine tâche.

Créer un utilisateur IAM

Les services tels qu'Amazon Kinesis Data Analytics nécessitent que vous fournissiez des informations d'identification lorsque vous y accédez afin que le service puisse déterminer si vous êtes autorisé à accéder aux ressources détenues par ce service. AWS La console exige votre mot de passe. Vous pouvez créer des clés d'accès pour accéder Compte AWS à l'API AWS CLI or. Cependant, nous vous déconseillons d'accéder à AWS l'aide des informations d'identification de votre Compte AWS. À la place, nous vous recommandons d'utiliser AWS Identity and Access Management (IAM). Créez un utilisateur IAM, ajoutez-le à un groupe IAM avec des autorisations administratives, puis attribuez-lui ces autorisations. Vous pouvez ensuite y accéder à AWS l'aide d'une URL spéciale et des informations d'identification de cet utilisateur IAM.

Si vous vous êtes inscrit AWS, mais que vous n'avez pas créé d'utilisateur IAM pour vous-même, vous pouvez en créer un à l'aide de la console IAM.

Les exercices de mise en route de ce guide présument que l'utilisateur (`adminuser`) dispose de privilèges d'administrateur. Suivez la procédure pour créer `adminuser` dans votre compte.

Pour créer un administrateur et vous connecter à la console

1. Créez un administrateur appelé `adminuser` dans votre Compte AWS. Pour obtenir des instructions, veuillez consulter Création de votre premier groupe d'utilisateurs et d'administrateurs IAM dans le Guide de l'utilisateur IAM.
2. Un utilisateur peut se connecter à l' AWS Management Console aide d'une URL spéciale. Pour plus d’informations, consultez Comment les utilisateurs se connectent à votre compte dans le Guide de l’utilisateur IAM.

Pour plus d’informations sur IAM, consultez les ressources suivantes :

- AWS Identity and Access Management (JE SUIS)
- Prise en main
- Guide de l’utilisateur IAM

Étape suivante

Étape 2 : configurer le AWS Command Line Interface (AWS CLI)

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 https://portal.aws.amazon.com/billing/signup.
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. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

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

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif 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 racine.

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

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.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée 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'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Étape 2 : configurer le AWS Command Line Interface (AWS CLI)

Suivez les étapes pour télécharger et configurer le AWS Command Line Interface (AWS CLI).

> **Important**
>
> Vous n'en avez pas besoin AWS CLI pour effectuer les étapes de l'exercice Getting Started. Cependant, quelques-uns des exercices de ce guide l'utilisent l' AWS CLI. Vous pouvez ignorer cette étape et passer à Étape 3 : Création de votre première application Amazon Kinesis Data Analytics, puis la configurer AWS CLI ultérieurement lorsque vous en aurez besoin.

Pour configurer le AWS CLI

1. Téléchargez et configurez l'interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'AWS Command Line Interface :

   - Configuration avec le AWS Command Line Interface
   - Configuration du AWS Command Line Interface

2. Ajoutez un profil nommé pour l'utilisateur administrateur dans le fichier de AWS CLI configuration. Vous utilisez ce profil lors de l'exécution des AWS CLI commandes. Pour plus d'informations sur les profils nommés, consultez la rubrique Profils nommés dans le Guide de l'utilisateur AWS Command Line Interface .

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   Pour obtenir la liste des régions disponibles Régions AWS, consultez la section Régions et points de terminaison dans le Référence générale d'Amazon Web Services.

3. Vérifiez la configuration en saisissant la commande d'aide suivante à l'invite de commande :

   ```
   aws help
   ```

Étape suivante

Étape 3 : Création de votre première application Amazon Kinesis Data Analytics

Étape 3 : Création de votre première application Amazon Kinesis Data Analytics

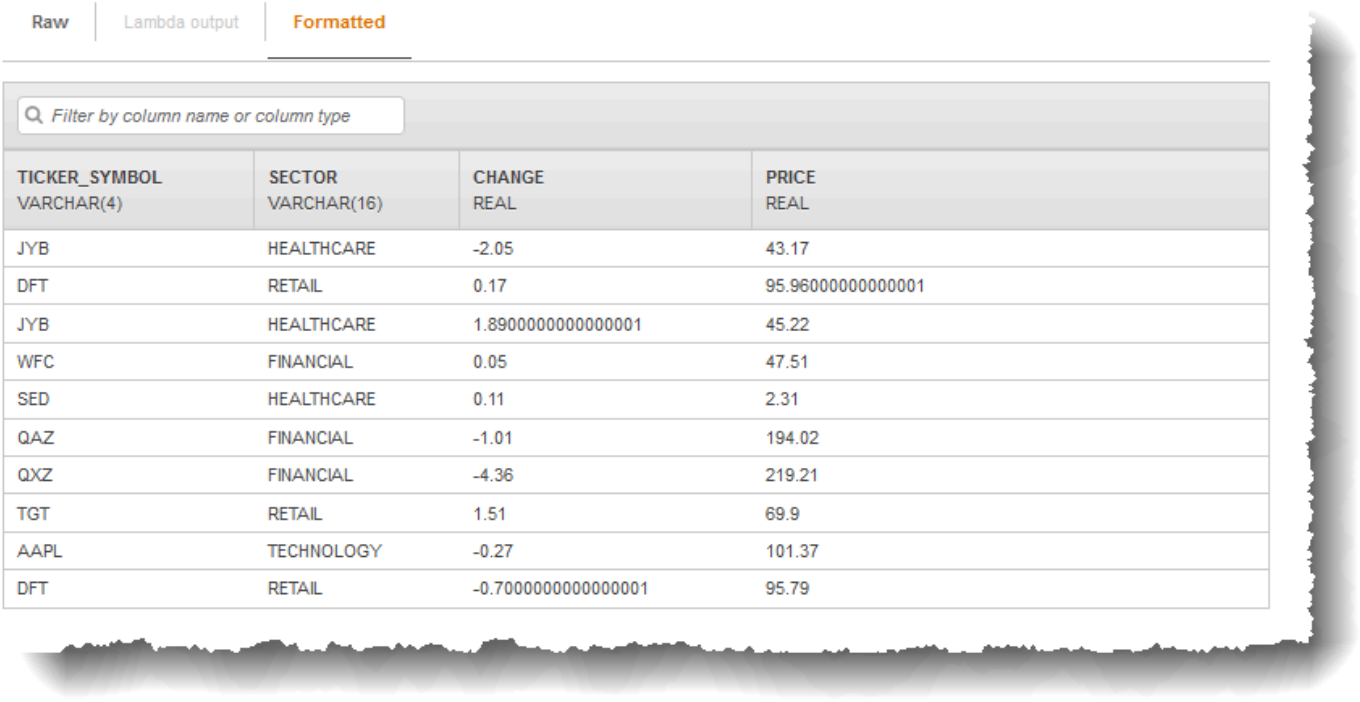
En suivant les étapes de cette section, vous pouvez créer votre première application Kinesis Data Analytics à l'aide de la console.

> **Note**
>
> Nous vous conseillons de consulter Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement avant d'effectuer l'exercice de mise en route.

Pour cet exercice de mise en route, vous pouvez utiliser la console avec le flux de démonstration ou les modèles avec code d'application.

- Si vous choisissez d'utiliser le flux de démonstration, la console crée dans votre compte un flux de données Kinesis appelé `kinesis-analytics-demo-stream`.

  Une application Kinesis Data Analytics requiert une source de streaming. Pour cette source, plusieurs exemples SQL de ce guide utilisent le flux de démonstration `kinesis-analytics-demo-stream`. La console exécute également un script qui ajoute en continu des exemples de données (enregistrements de simulations de transactions boursières) à ce flux, comme illustré ci-dessous.



TICKER_SYMBOL VARCHAR(4)	SECTOR VARCHAR(16)	CHANGE REAL	PRICE REAL
JYB	HEALTHCARE	-2.05	43.17
DFT	RETAIL	0.17	95.96000000000001
JYB	HEALTHCARE	1.8900000000000001	45.22
WFC	FINANCIAL	0.05	47.51
SED	HEALTHCARE	0.11	2.31
QAZ	FINANCIAL	-1.01	194.02
QXZ	FINANCIAL	-4.36	219.21
TGT	RETAIL	1.51	69.9
AAPL	TECHNOLOGY	-0.27	101.37
DFT	RETAIL	-0.7000000000000001	95.79

  Dans cet exercice, vous pouvez utiliser `kinesis-analytics-demo-stream` comme source de streaming pour votre application.

> **Note**
>
> Le flux de démonstration reste dans votre compte. Vous pouvez l'utiliser pour tester d'autres exemples de ce guide. Toutefois, lorsque vous quittez la console, le script que la console utilise arrête de remplir les données. Si nécessaire, la console offre la possibilité de recommencer à remplir le flux.

- Si vous choisissez d'utiliser les modèles avec exemple de code d'application, vous utilisez le code du modèle fourni par la console pour effectuer des analyses simples sur le flux de démonstration.

Utilisez ces fonctions pour configurer rapidement votre première application comme suit :

1. Créer une application : vous devez uniquement fournir un nom. La console crée l'application et le service définit l'état de l'application sur `READY`.

2. Configurer l'entrée : vous ajoutez d'abord une source de streaming, le flux de démonstration. Vous devez créer un flux de démonstration dans la console pour pouvoir utiliser celui-ci. La console prélève ensuite un échantillon aléatoire d'enregistrements dans le flux de démonstration et déduit un schéma pour le flux d'entrée intégré à l'application qui est créé. La console nomme le flux intégré à l'application `SOURCE_SQL_STREAM_001`.

   La console utilise l'API de découverte pour déduire le schéma. Si nécessaire, vous pouvez modifier le schéma déduit. Pour plus d'informations, consultez DiscoverInputSchema. Kinesis Data Analytics utilise ce schéma pour créer un flux intégré à l'application.

   Lorsque vous démarrez l'application, Kinesis Data Analytics lit le flux de démonstration en continu en votre nom et insère des lignes dans le flux d'entrée intégré à l'application `SOURCE_SQL_STREAM_001`.

3. Spécifiez le code d'application : Vous utilisez un modèle (appelé Continuous filter) qui fournit le code suivant :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM"
```

```
  (symbol VARCHAR(4), sector VARCHAR(12), CHANGE DOUBLE, price DOUBLE);

-- Create pump to insert into output.
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM ticker_symbol, sector, CHANGE, price
      FROM "SOURCE_SQL_STREAM_001"
      WHERE sector SIMILAR TO '%TECH%';
```

Le code d'application interroge le flux intégré à l'application `SOURCE_SQL_STREAM_001`. Puis, il insère les lignes résultantes dans un autre flux intégré à l'application, `DESTINATION_SQL_STREAM`, à l'aide de pompes. Pour plus d'informations sur ce modèle de codage, consultez Code d'application.

Pour plus d'informations sur les éléments du langage SQL pris en charge par Kinesis Data Analytics, consultez la Référence SQL Amazon Kinesis Data Analytics.

4. Configuration de la sortie : dans cet exercice, vous ne configurez pas n'importe quelle sortie. En d'autres termes, vous ne conservez pas les données du flux intégré à l'application que votre application crée dans n'importe quelle destination externe. Au lieu de cela, vous vérifiez les résultats de la requête dans la console. D'autres exemples de ce guide vous expliquent comment configurer la sortie. Pour obtenir un exemple, consultez Exemple : Création d'alertes simples.

> **Important**
>
> L'exercice utilise la région USA Est (Virginie du Nord) (us-east-1) pour configurer l'application. Vous pouvez utiliser n'importe lequel des modèles pris en charge Régions AWS.

Étape suivante

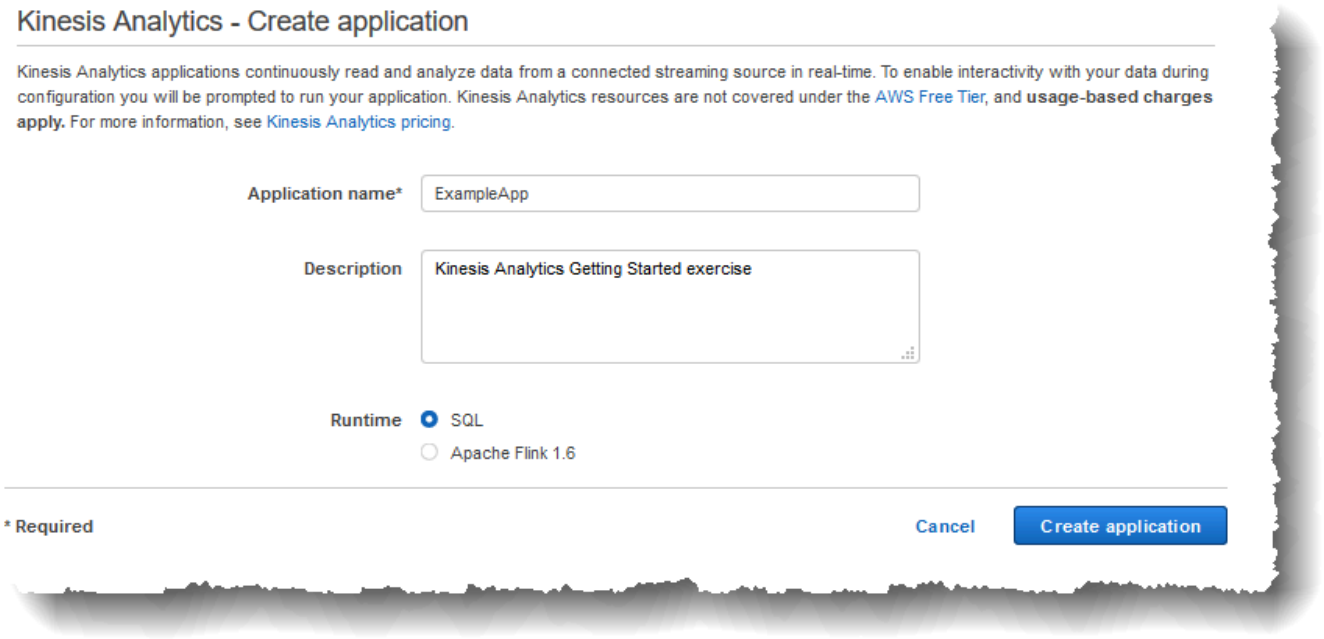
Étape 3.1 : Créer une application

Étape 3.1 : Créer une application

Dans cette section, vous allez créer une application Amazon Kinesis Data Analytics. Vous configurez l'entrée de l'application dans l'étape suivante.

Pour créer une application d'analyse de données

1. Connectez-vous à la console Managed Service for Apache Flink AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Créer une application.
3. Sur la page Créer une application, saisissez un nom d'application, entrez une description, sélectionnez SQL pour le paramètre Runtime (Exécution), puis sélectionnez Create application (Créer une application).



Une application Kinesis Data Analytics est ainsi créée à l'état READY. La console affiche le hub d'applications où vous pouvez configurer l'entrée et la sortie.

> **Note**
>
> Pour créer une application, l'opération CreateApplication a besoin uniquement du nom de l'application. Vous pouvez ajouter une configuration d'entrée et de sortie une fois que vous avez créé une application dans la console.

Dans l'étape suivante, vous configurez une entrée pour l'application. Dans la configuration d'entrée, vous ajoutez la source de données de diffusion à l'application et vous découvrez un

schéma pour un flux d'entrée intégré à l'application en prélevant un échantillon de données dans la source de diffusion.

Étape suivante

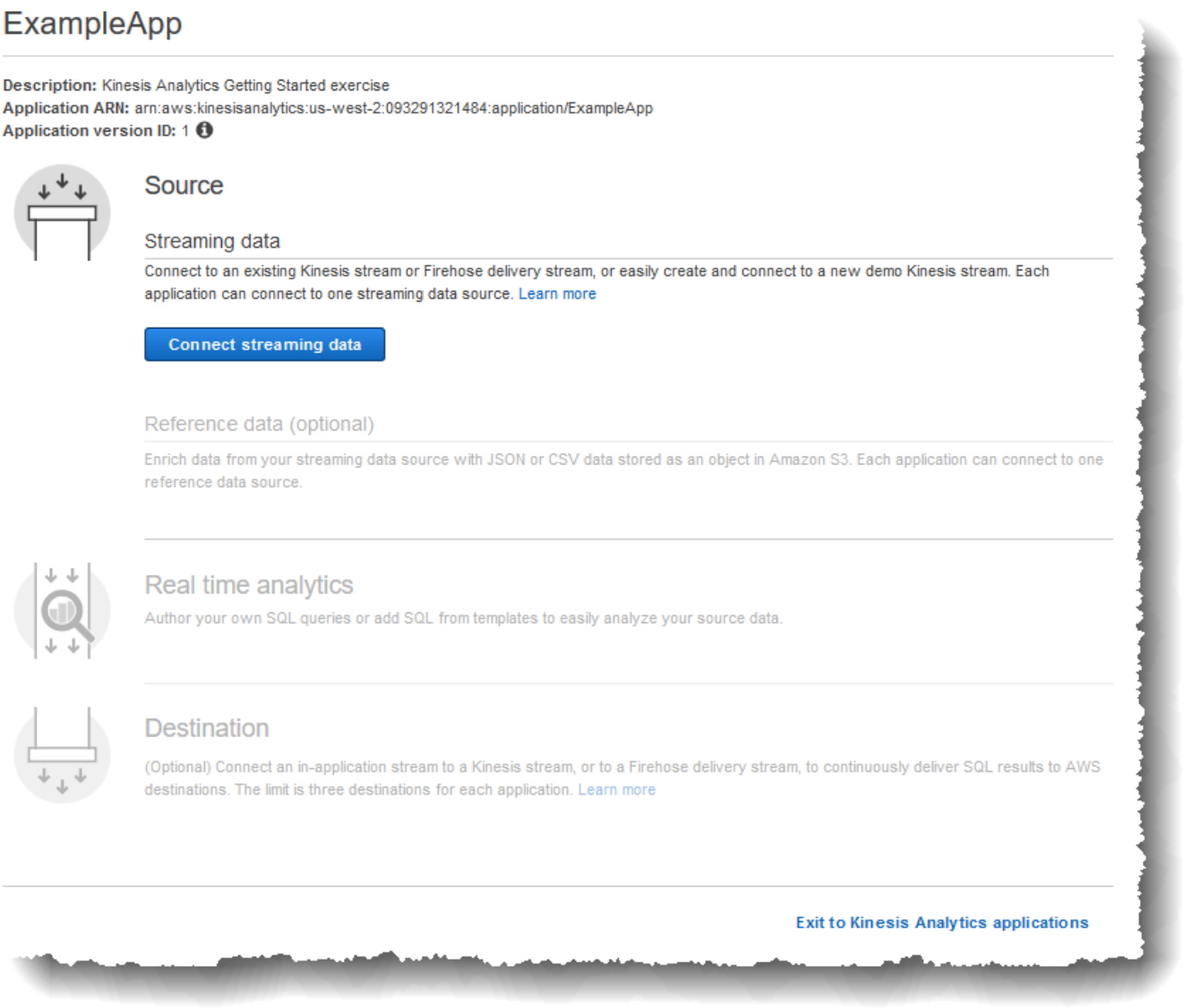
Étape 3.2 : Configuration de l'entrée

Étape 3.2 : Configuration de l'entrée

Votre application a besoin d'une source de diffusion. Pour vous aider à démarrer, la console peut créer un flux de démonstration (appelé `kinesis-analytics-demo-stream`). La console exécute également un script qui remplit des enregistrements dans le flux.

Pour ajouter une source de streaming à votre application

1. Sur la page de hub des applications de la console, sélectionnez Connect streaming data (Connecter les données de streaming).



2. Sur la page qui s'affiche, vérifiez les éléments suivants :

   - La section Source dans laquelle vous spécifiez une source de streaming pour votre application. Vous pouvez sélectionner une source de diffusion existante ou en créer une. Dans cet exercice, vous créez un nouveau flux, le flux de démonstration.

     Par défaut, la console nomme le flux d'entrée intégré à l'application créé `INPUT_SQL_STREAM_001`. Pour cet exercice, conservez ce nom tel qu'il apparaît.

- Nom de référence du flux : ce champ affiche le nom du flux d'entrée intégré à l'application qui est créé, `SOURCE_SQL_STREAM_001`. Vous pouvez le modifier, mais conservez-le pour cet exercice.

  Dans la configuration d'entrée, vous mappez le flux démonstration à un flux d'entrée intégré à l'application qui est créé. Lorsque vous démarrez l'application, Amazon Kinesis Data Analytics lit en continu le flux de démonstration et insère des lignes dans le flux d'entrée intégré à l'application. Vous interrogez ce flux d'entrée intégré à l'application dans votre code d'application.

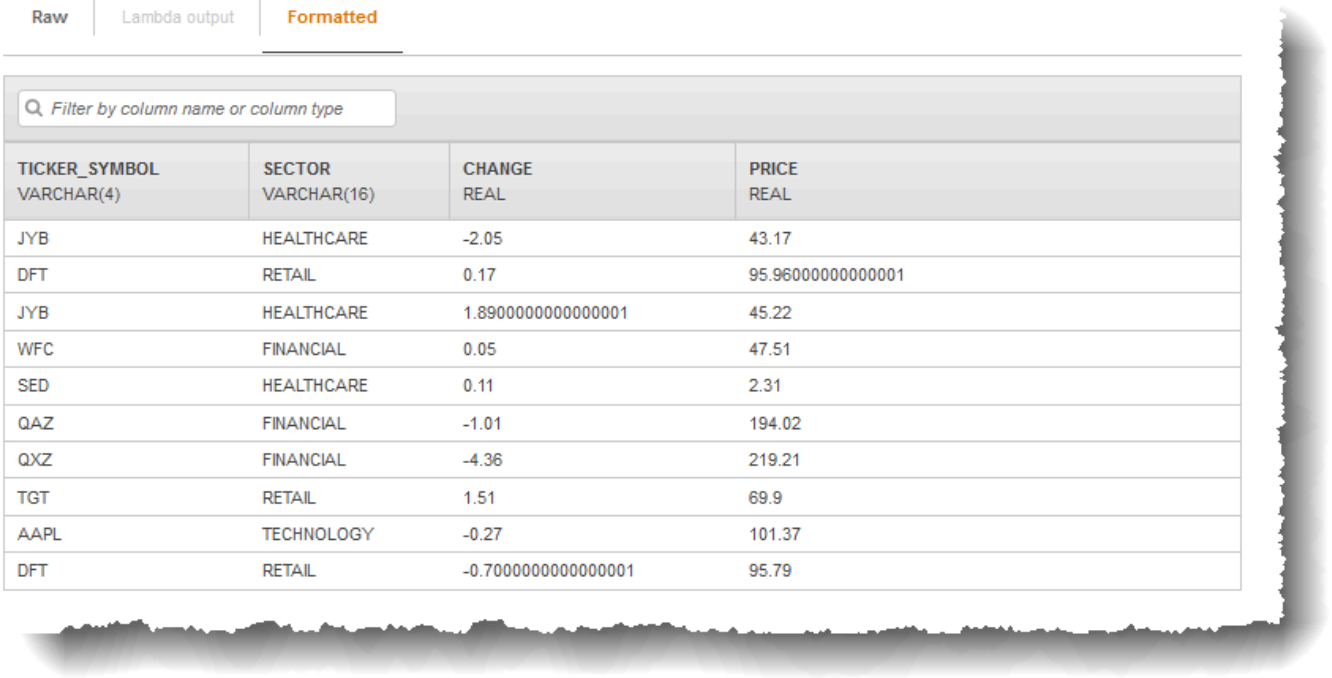
- Prétraitement des enregistrements avec AWS Lambda : cette option vous permet de spécifier une AWS Lambda expression qui modifie les enregistrements du flux d'entrée avant l'exécution du code de votre application. Dans le cadre de cet exercice, laissez l'option Disabled sélectionnée. Pour plus d'informations sur le prétraitement Lambda, consultez Prétraitement des données à l'aide d'une fonction Lambda.

Une fois que vous avez fourni toutes les informations de cette page, la console envoie une demande de mise à jour (voir UpdateApplication) pour ajouter la configuration d'entrée à l'application.

3. Sur la page Source , choisissez Configure a new stream.
4. Choisissez Create demo stream. La console configure l'entrée de l'application en effectuant les opérations suivantes :

   - La console crée un flux de données Kinesis appelé `kinesis-analytics-demo-stream`.
   - La console remplit le flux avec des exemples de données de symbole boursier.
   - A l'aide de l'action DiscoverInputSchema, la console déduit un schéma en lisant des échantillons d'enregistrements du flux. Le schéma déduit est celui du flux d'entrée intégré à l'application qui est créé. Pour plus d'informations, consultez Configuration de l'entrée de l'application.
   - La console affiche le schéma déduit et les exemples de données qu'elle lit à partir de la source de streaming pour déduire le schéma.

   La console affiche les exemples d'enregistrements de la source de diffusion.



Les informations suivantes apparaissent sur la page Stream sample de la console :

- L'onglet Raw stream sample affiche les enregistrements de flux bruts échantillonnés par l'action d'API DiscoverInputSchema pour déduire le schéma.
- L'onglet Formatted stream sample affiche la version sous forme de tableau des données de l'onglet Raw stream sample.
- Si vous choisissez Edit schema, vous pouvez modifier le schéma déduit. Pour cet exercice, ne changez pas le schéma déduit. Pour plus d'informations sur l'édition d'un schéma, consultez Utilisation de l'éditeur de schéma.

  Si vous choisissez Rediscover schema, vous pouvez demander à la console d'exécuter de nouveau DiscoverInputSchema et de déduire le schéma.

5. Choisissez Save and continue (Enregistrer et continuer).

   Vous disposez maintenant d'une application avec une configuration d'entrée ajoutée à celle-ci. Dans l'étape suivante, vous ajoutez du code SQL pour effectuer certaines analyses sur le flux d'entrée de données intégré à l'application.

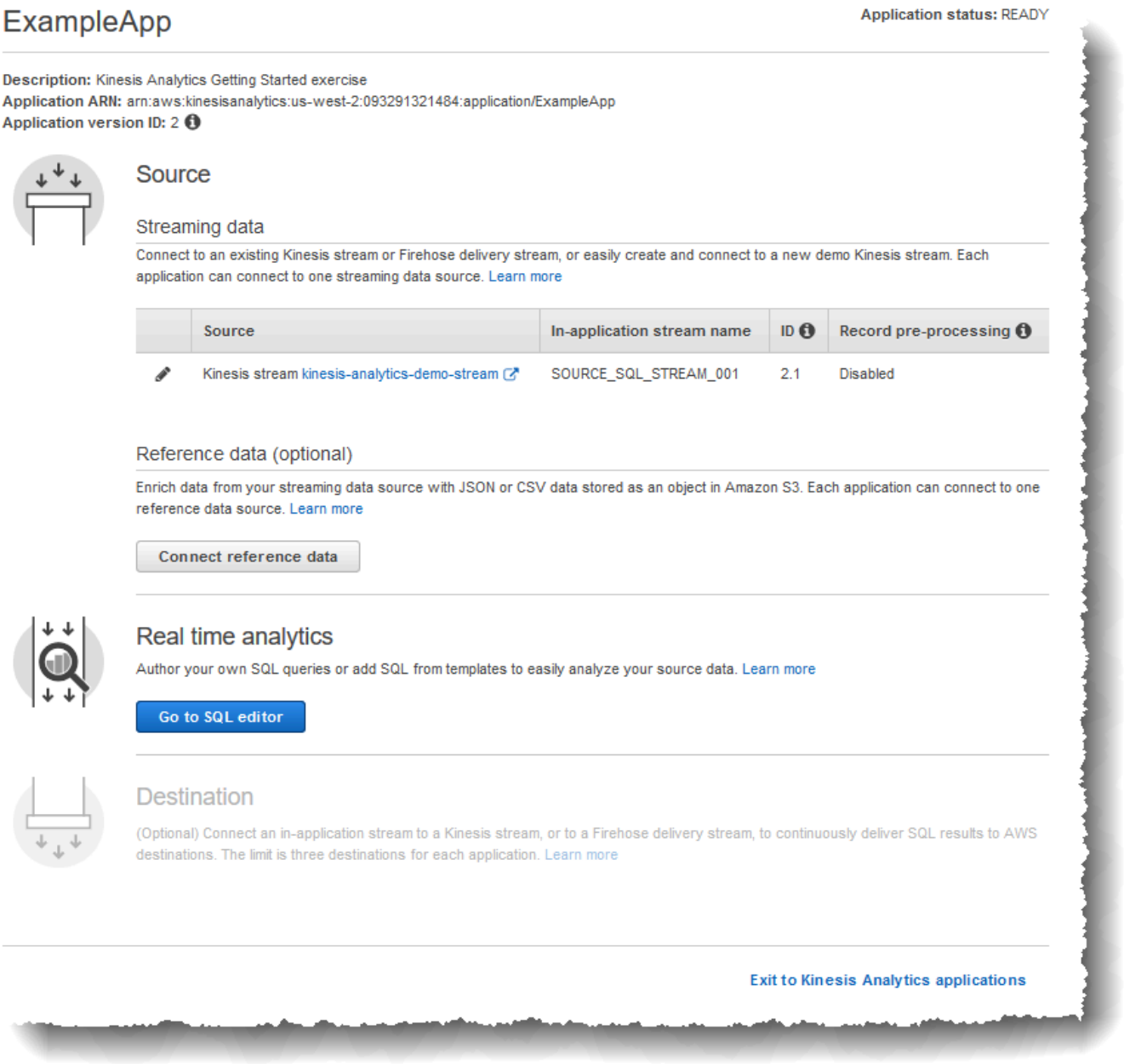
Étape suivante

[Étape 3.3 : Ajout d'analyses en temps réel (ajout du code d'application)]

Étape 3.3 : Ajout d'analyses en temps réel (ajout du code d'application)

Vous pouvez écrire vos propres requêtes SQL sur le flux intégré à l'application, mais pour l'étape suivante, vous utilisez l'un des modèles avec exemple de code.

1. Sur la page du hub d'applications, choisissez Go to SQL editor.



2. Dans le champ Voulez-vous commencer à courir "ExampleApp« ? boîte de dialogue, choisissez Oui, démarrer l'application.

   La console envoie une demande pour démarrer l'application (voir StartApplication), puis la page de l'éditeur SQL s'affiche.

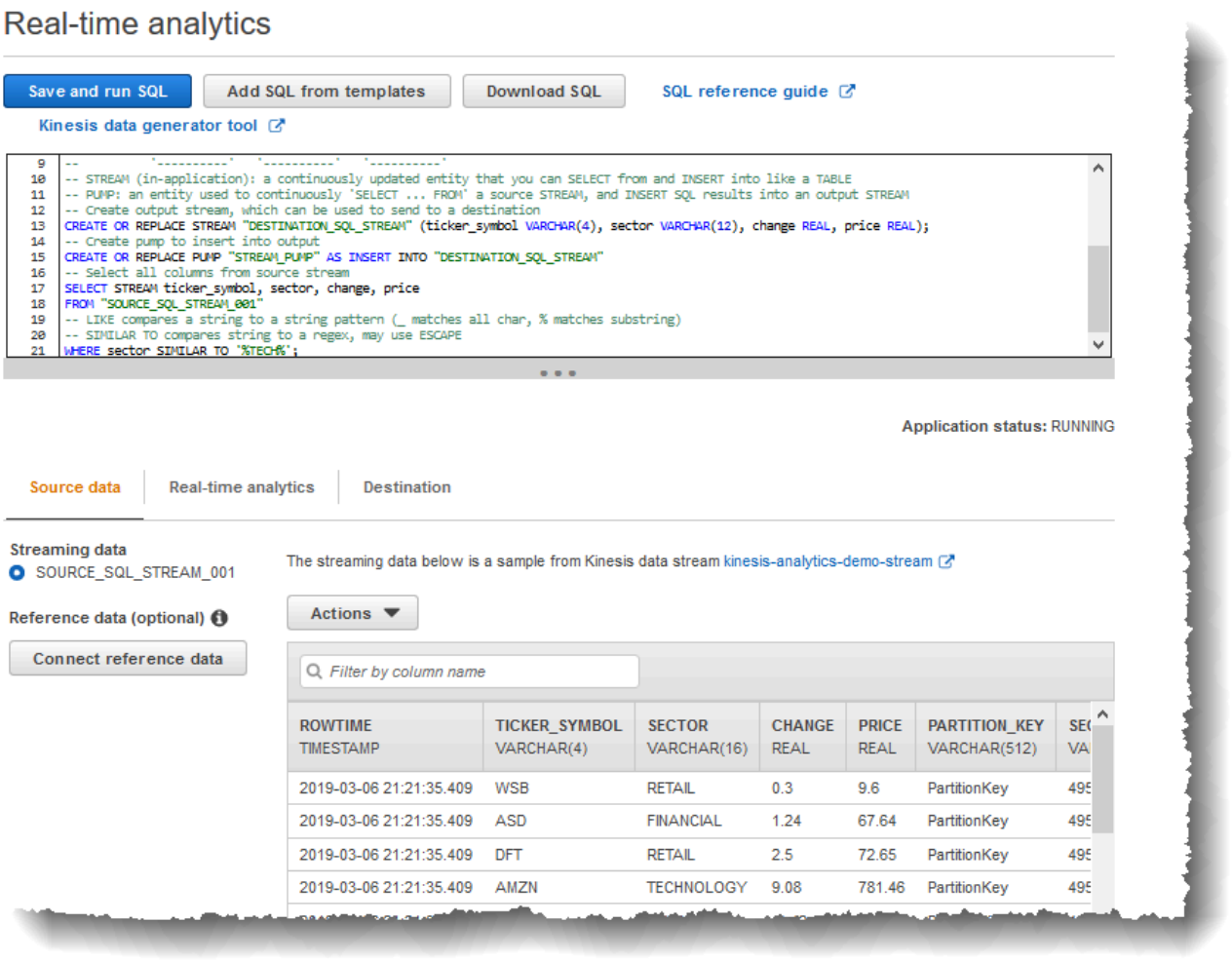
3. La console ouvre la page de l'éditeur SQL. Vérifiez la page, y compris les boutons (Add SQL from templates, Save and run SQL) et les différents onglets.
4. Dans l'éditeur SQL, choisissez Add SQL from templates.
5. Dans la liste des modèles disponibles, choisissez Continuous filter. L'exemple de code lit les données d'un flux intégré à l'application (la clause `WHERE` filtre les lignes) et les insère dans un autre flux intégré à l'application comme suit :

   - Il crée le flux de données intégré à l'application `DESTINATION_SQL_STREAM`.
   - Il crée une pompe `STREAM_PUMP` et l'utilise pour sélectionner des lignes dans `SOURCE_SQL_STREAM_001` et les insérer dans `DESTINATION_SQL_STREAM`.

6. Choisissez Add this SQL to editor.
7. Testez le code d'application comme suit :

   N'oubliez pas que vous avez déjà démarré l'application (l'état est RUNNING). Par conséquent, Amazon Kinesis Data Analytics lit déjà en continu la source de streaming et ajoute des lignes dans le flux intégré à l'application `SOURCE_SQL_STREAM_001`.

   a. Dans l'éditeur SQL, choisissez Save and run SQL. La console envoie d'abord la demande de mise à jour pour enregistrer le code d'application. Ensuite, le code s'exécute en continu.
   b. Vous pouvez voir les résultats dans l'onglet Real-time analytics.



L'éditeur SQL comporte les onglets suivants :

- L'onglet Source data affiche un flux d'entrée intégré à l'application qui est mappé à la source de diffusion. Choisissez le flux intégré à l'application. Vous pouvez alors voir des données entrantes. Vous remarquerez les colonnes supplémentaires dans le flux d'entrée intégré à l'application qui n'étaient pas spécifiées dans la configuration d'entrée. Celles-ci incluent les colonnes d'horodatage suivantes :

  - ROWTIME : chaque ligne d'un flux intégré à l'application comporte une colonne spéciale appelée `ROWTIME`. Cette colonne est l'horodatage du moment où Amazon Kinesis Data Analytics a inséré la ligne dans le premier flux intégré à l'application (le flux mappé à la source de streaming).

- Approximate_Arrival_Time : chaque enregistrement Kinesis Data Analytics inclut une valeur appelée `Approximate_Arrival_Time`. Cette valeur est l'horodatage d'arrivée approximatif défini lorsque la source de streaming reçoit et stocke l'enregistrement avec succès. Quand Kinesis Data Analytics lit les enregistrements d'une source de streaming, il extrait cette colonne et l'insère dans le flux d'entrée intégré à l'application.

Ces valeurs d'horodatage sont utiles dans les requêtes à fenêtres temporelles. Pour plus d'informations, consultez Requêtes à fenêtres.

- L'onglet Real-time analytics affiche tous les autres flux intégrés à l'application créés par votre code d'application. Il inclut également le flux d'erreurs. Kinesis Data Analytics envoie les lignes qu'il ne peut pas traiter dans le flux d'erreurs. Pour plus d'informations, consultez Gestion des erreurs.

Choisissez `DESTINATION_SQL_STREAM` pour afficher les lignes que votre code d'application a insérées. Vous remarquerez les colonnes supplémentaires que votre code d'application n'a pas créées. Celles-ci incluent la colonne d'horodatage `ROWTIME`. Kinesis Data Analytics copie simplement ces valeurs depuis la source (`SOURCE_SQL_STREAM_001`).

- L'onglet Destination affiche la destination externe où Kinesis Data Analytics écrit les résultats de requête. Vous n'avez pas encore configuré de destination externe pour la sortie de votre application.

Étape suivante

Étape 3.4 : (Facultatif) Mise à jour du code d'application

Étape 3.4 : (Facultatif) Mise à jour du code d'application

Dans cette étape, vous découvrez comment mettre à jour le code d'application.

Pour mettre à jour votre code d'application

1. Créez un autre flux intégré à l'application comme suit :

   - Créez un autre flux intégré à l'application appelé `DESTINATION_SQL_STREAM_2`.
   - Créez une pompe, puis utilisez-la pour insérer des lignes dans le flux nouvellement créé en sélectionnant des lignes dans le flux `DESTINATION_SQL_STREAM`.

   Dans l'éditeur SQL, ajoutez le code suivant au code d'application existant :

   ```
   CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM_2"
              (ticker_symbol VARCHAR(4),
               change        DOUBLE,
               price         DOUBLE);

   CREATE OR REPLACE PUMP "STREAM_PUMP_2" AS
      INSERT INTO "DESTINATION_SQL_STREAM_2"
         SELECT STREAM ticker_symbol, change, price
         FROM   "DESTINATION_SQL_STREAM";
   ```

   Enregistrez et exécutez le code. Des flux intégrés à l'application supplémentaires apparaissent dans l'onglet Real-time analytics.

2. Créez deux flux intégrés à l'application. Filtrez les lignes dans `SOURCE_SQL_STREAM_001` en fonction du symbole boursier, puis insérez-les dans ces flux distincts.

   Ajoutez les instructions SQL suivantes à votre code d'application :

   ```
   CREATE OR REPLACE STREAM "AMZN_STREAM"
              (ticker_symbol VARCHAR(4),
               change        DOUBLE,
               price         DOUBLE);

   CREATE OR REPLACE PUMP "AMZN_PUMP" AS
      INSERT INTO "AMZN_STREAM"
         SELECT STREAM ticker_symbol, change, price
         FROM   "SOURCE_SQL_STREAM_001"
         WHERE  ticker_symbol SIMILAR TO '%AMZN%';

   CREATE OR REPLACE STREAM "TGT_STREAM"
              (ticker_symbol VARCHAR(4),
   ```

```
            change        DOUBLE,
            price         DOUBLE);

CREATE OR REPLACE PUMP "TGT_PUMP" AS
   INSERT INTO "TGT_STREAM"
      SELECT STREAM ticker_symbol, change, price
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  ticker_symbol SIMILAR TO '%TGT%';
```

Enregistrez et exécutez le code. Vous remarquerez des flux intégrés à l'application supplémentaires dans l'onglet Real-time analytics.

Vous disposez maintenant de votre première application Amazon Kinesis Data Analytics opérationnelle. Dans cet exercice, vous avez effectué les opérations suivantes :

- Création de votre première application Kinesis Data Analytics.

- Vous avez configuré l'entrée d'application qui identifiait le flux de démonstration comme source de diffusion et vous l'avez mappée à un flux intégré à l'application (`SOURCE_SQL_STREAM_001`) qui a été créé. Kinesis Data Analytics lit en continu le flux de démonstration et insère des enregistrements dans le flux intégré à l'application.

- Votre code d'application a interrogé le flux `SOURCE_SQL_STREAM_001` et a écrit la sortie dans un autre flux intégré à l'application, appelé `DESTINATION_SQL_STREAM`.

Vous pouvez maintenant configurer le cas échéant la sortie d'application pour écrire celle-ci dans une destination externe. En d'autres termes, vous pouvez configurer la sortie de l'application pour qu'elle écrive des enregistrements du flux `DESTINATION_SQL_STREAM` dans une destination externe. Pour cet exercice, cette étape est facultative. Pour apprendre à configurer la destination, allez à l'étape suivante.

Étape suivante

Étape 4 (facultatif) Modification du schéma et du code SQL à l'aide de la console.

Étape 4 (facultatif) Modification du schéma et du code SQL à l'aide de la console

Vous trouverez ci-après des informations sur la façon de modifier un schéma déduit et le code SQL d'Amazon Kinesis Data Analytics. Pour ce faire, vous pouvez utiliser l'éditeur de schéma et l'éditeur SQL, qui font partie de la console Kinesis Data Analytics.

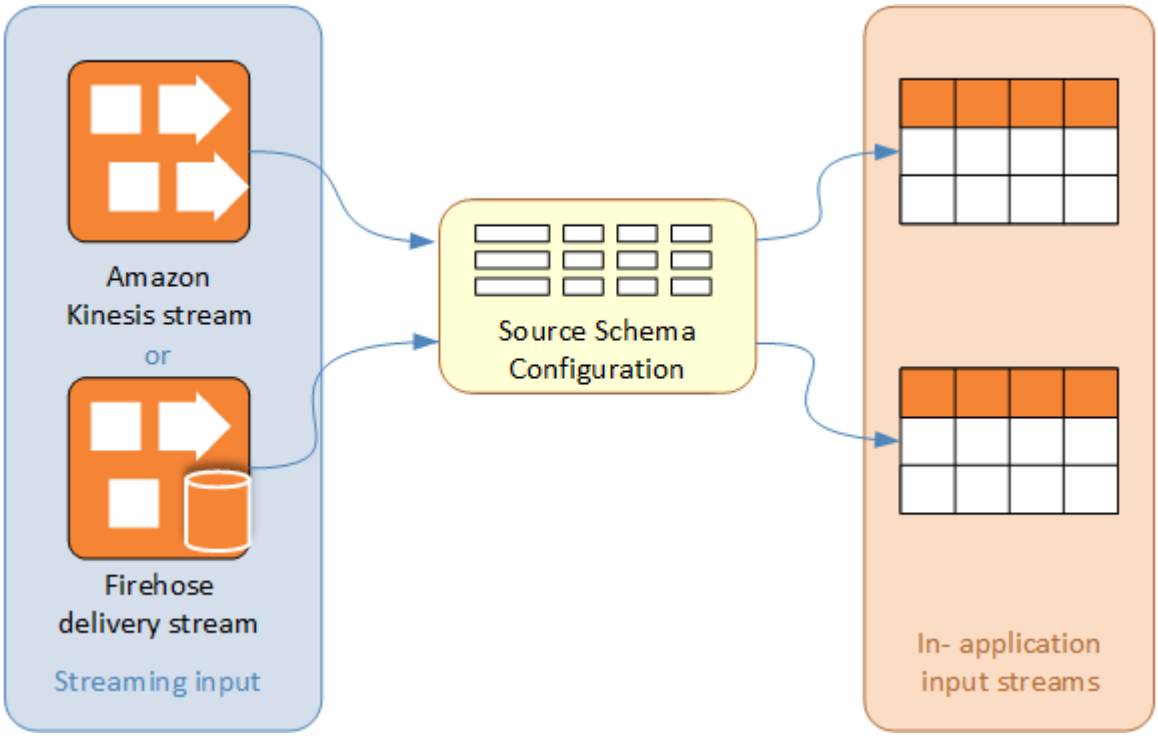
> **Note**
>
> Pour accéder à des exemples de données le rôle de l'utilisateur de connexion doit disposer de l'autorisation `kinesisanalytics:GetApplicationState`. Pour plus d'informations sur les autorisations de l'application Kinesis Data Analytics, consultez Présentation de la gestion de l'accès.

Rubriques

- Utilisation de l'éditeur de schéma
- Utilisation de l'éditeur SQL

Utilisation de l'éditeur de schéma

Le schéma du flux d'entrée d'une application Amazon Kinesis Data Analytics définit la manière dont les données du flux sont mises à la disposition des requêtes SQL dans l'application.



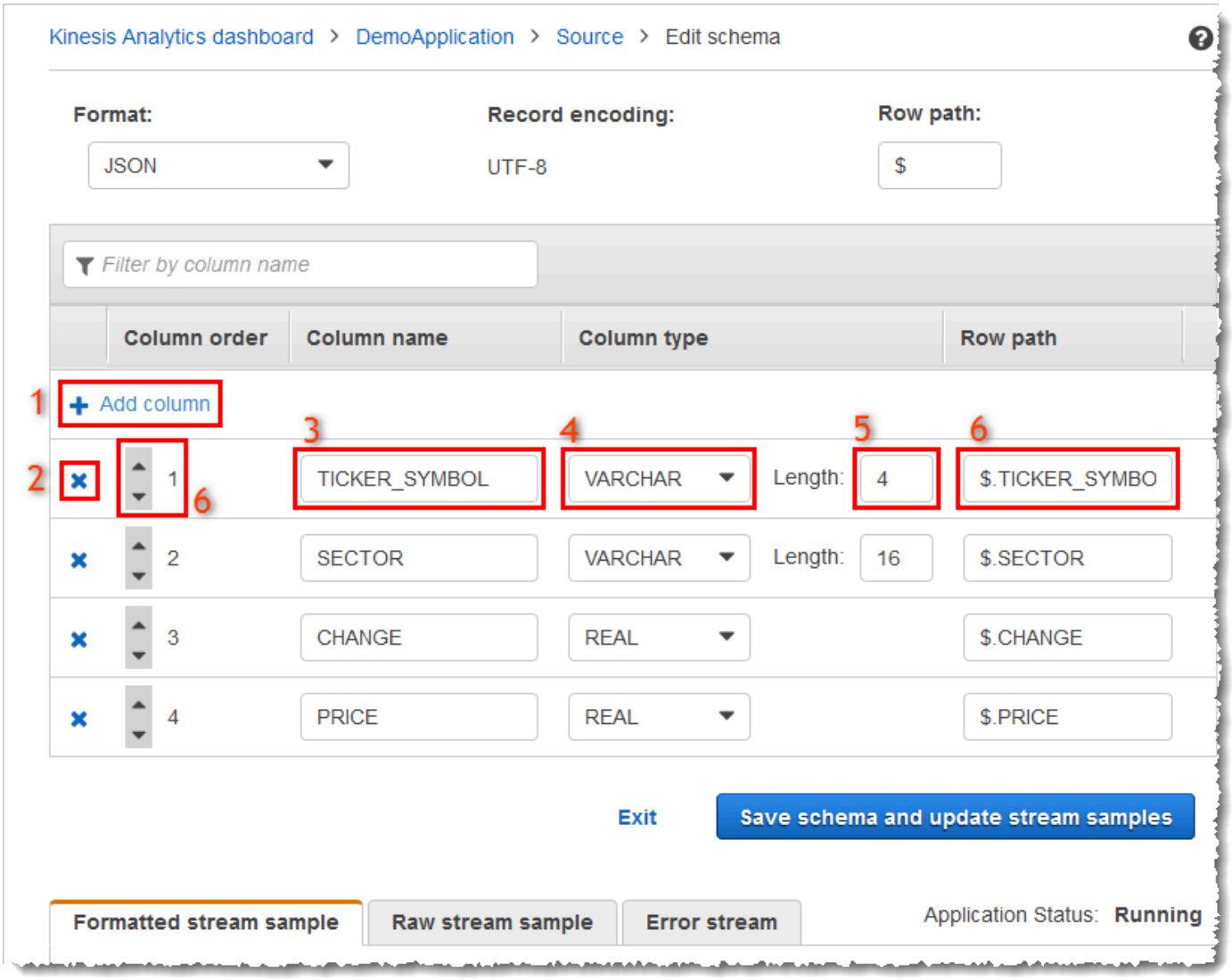
Le schéma contient des critères de sélection permettant de déterminer la partie de l'entrée de streaming qui est transformée en colonne dans le flux d'entrée intégré à l'application. Cette entrée peut être l'une des suivantes :

- Une expression JSONPath pour les flux d'entrée JSON. JSONPath est un outil d'interrogation des données JSON.
- Un numéro de colonne pour les flux d'entrée au format CSV.
- Un nom de colonne et un type de données SQL pour présenter les données dans le flux de données intégré à l'application. Le type de données contient également une longueur pour les données de caractères ou les données binaires.

La console tente de générer le schéma à l'aide de DiscoverInputSchema. Si la découverte de schéma échoue ou retourne un schéma incorrect ou incomplet, vous devez modifier le schéma manuellement à l'aide de l'éditeur de schéma.

Ecran principal de l'éditeur de schéma

La capture d'écran suivante montre l'écran principal de l'éditeur de schéma.



Vous pouvez appliquer les modifications suivantes au schéma :

- Ajouter une colonne (1) : Il se peut que vous ayez besoin d'ajouter une colonne de données si un élément de données n'est pas détecté automatiquement.
- Supprimer une colonne (2) : Vous pouvez exclure des données du flux source si votre application ne les exige pas. Cette exclusion n'affecte pas les données du flux source. Si des données sont exclues, elles ne sont tout simplement pas mises à la disposition de l'application.
- Renommer une colonne (3) : Un nom de colonne ne peut pas être vide, doit comporter plus d'un caractère et ne doit pas contenir de mots réservés SQL. Il doit également respecter les critères de dénomination des identifiants ordinaires pour SQL : Le nom doit commencer par une lettre et contenir uniquement des lettres, des traits de soulignement et des chiffres.

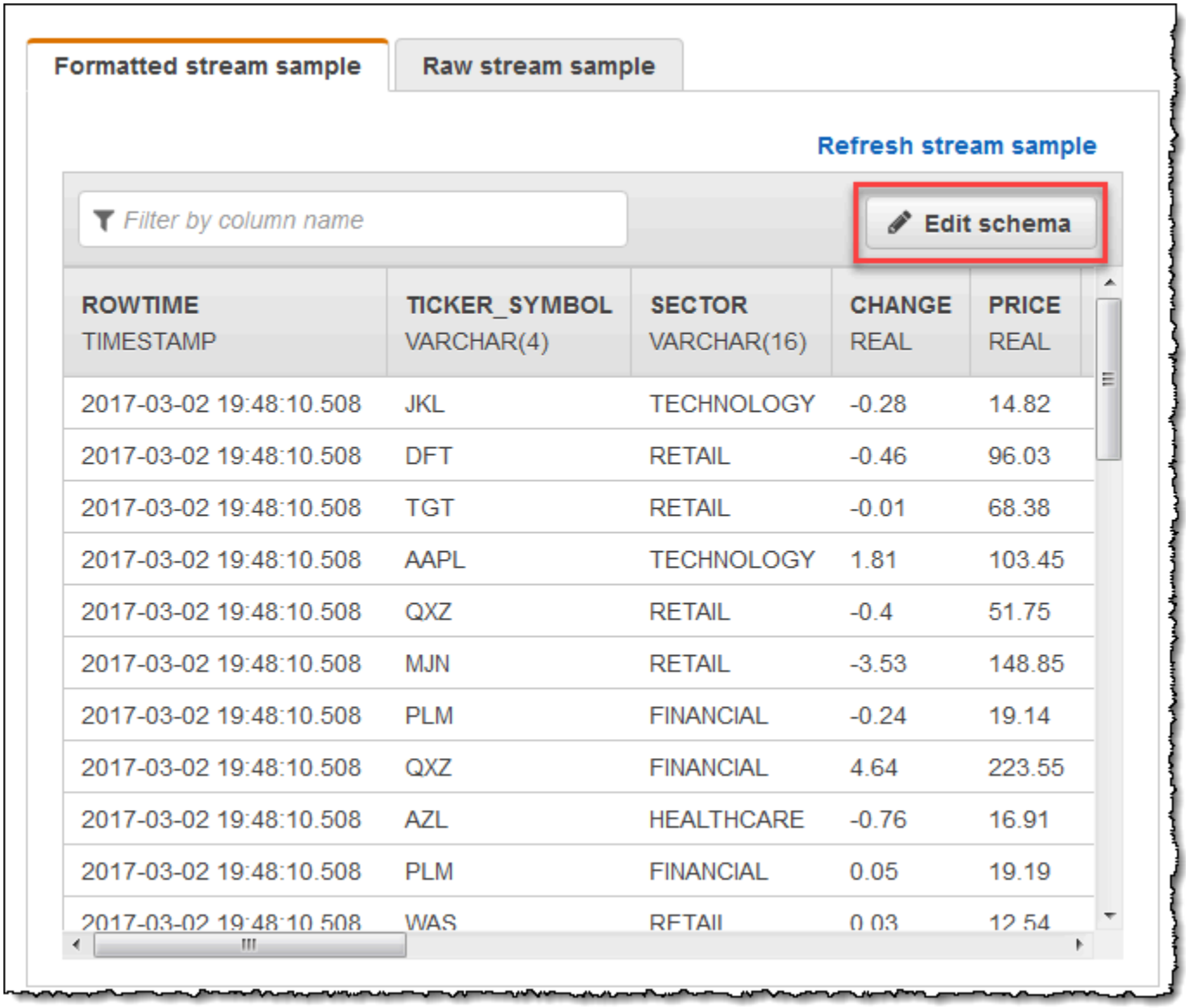
- Changer le type de données (4) ou la longueur (5) d'une colonne : Vous pouvez spécifier un type de données compatible pour une colonne. Si vous spécifiez un type de données incompatible, soit la colonne contiendra la valeur NULL soit le flux intégré à l'application n'est pas du tout rempli. Dans ce dernier cas, les erreurs seront écrites dans le flux d'erreurs. Si vous spécifiez une longueur pour une colonne qui est trop petite, les données entrantes seront tronquées.
- Modifier les critères de sélection d'une colonne (6) : Vous pouvez modifier l'expression JSONPath ou l'ordre des colonnes CSV pour déterminer la source des données d'une colonne. Pour modifier les critères de sélection d'un schéma JSON, saisissez une nouvelle valeur pour l'expression de chemin de la ligne. Un schéma CSV utilise l'ordre des colonnes comme critères de sélection. Pour modifier les critères de sélection d'un schéma CSV, changez l'ordre des colonnes.

Modification du schéma d'une source de streaming

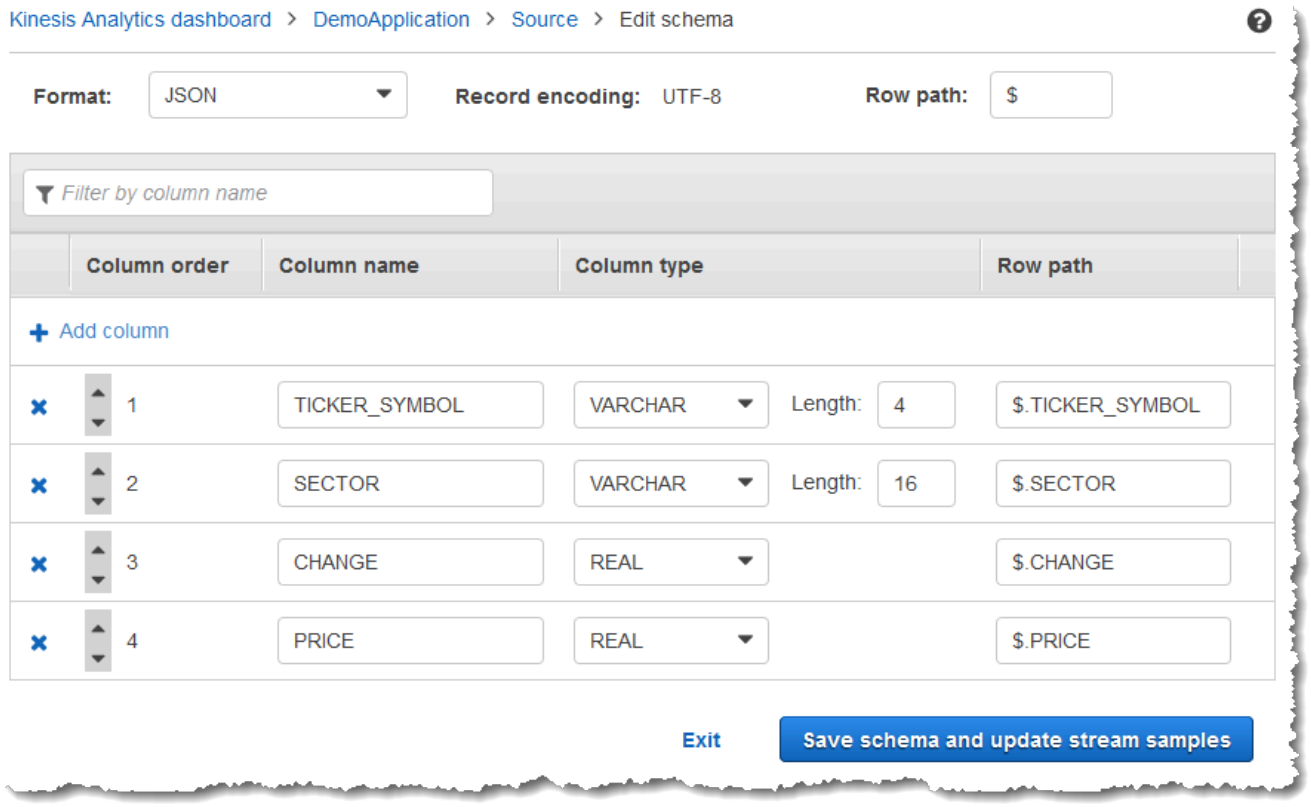
Si vous avez besoin de modifier le schéma d'une source de streaming, procédez comme suit.

Pour modifier le schéma d'une source de streaming

1. Sur la page Source, choisissez Edit schema.



2. Sur la page Edit schema, modifiez le schéma source.



3. Sous Format, choisissez JSON ou CSV. Pour le format JSON ou CSV, l'encodage pris en charge est ISO 8859-1.

Pour plus d'informations sur la modification du schéma JSON ou CSV, consultez les procédures des sections suivantes.

Modification d'un schéma JSON

Vous pouvez modifier un schéma JSON en procédant comme suit.

Pour modifier un schéma JSON

1. Dans l'éditeur de schéma, choisissez Add column pour ajouter une colonne.

   Une nouvelle colonne s'affiche en première position. Pour modifier l'ordre des colonnes, choisissez les flèches haut et bas en regard du nom de la colonne.

   Fournissez les informations suivantes pour la nouvelle colonne :

   - Sous Column name, saisissez un nom.

Un nom de colonne ne peut pas être vide, doit comporter plus d'un caractère et ne doit pas contenir de mots réservés SQL. Il doit également respecter les critères de dénomination des identifiants ordinaires pour SQL : Le nom doit commencer par une lettre et contenir uniquement des lettres, des traits de soulignement et des chiffres.

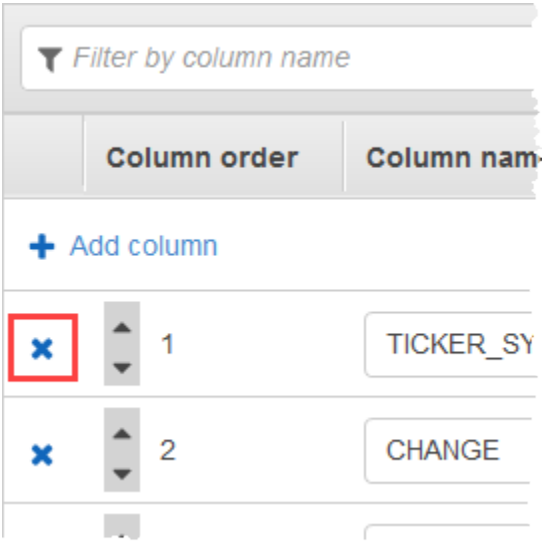
- Sous Column type, saisissez un type de données SQL.

  Le type de colonne peut être n'importe quel type de données SQL pris en charge. Si le nouveau type de données est CHAR, VARBINARY ou VARCHAR, indiquez une longueur de données sous Length. Pour plus d'informations, consultez Types de données.

- Sous Row path, indiquez un chemin de ligne. Un chemin de ligne est une expression JSONPath valide qui est mappée à un élément JSON.

> **Note**
>
> La valeur Row path de base est le chemin vers le parent de premier niveau qui contient les données à importer. Cette valeur est $ par défaut. Pour plus d'informations, consultez `RecordRowPath` dans `JSONMappingParameters`.

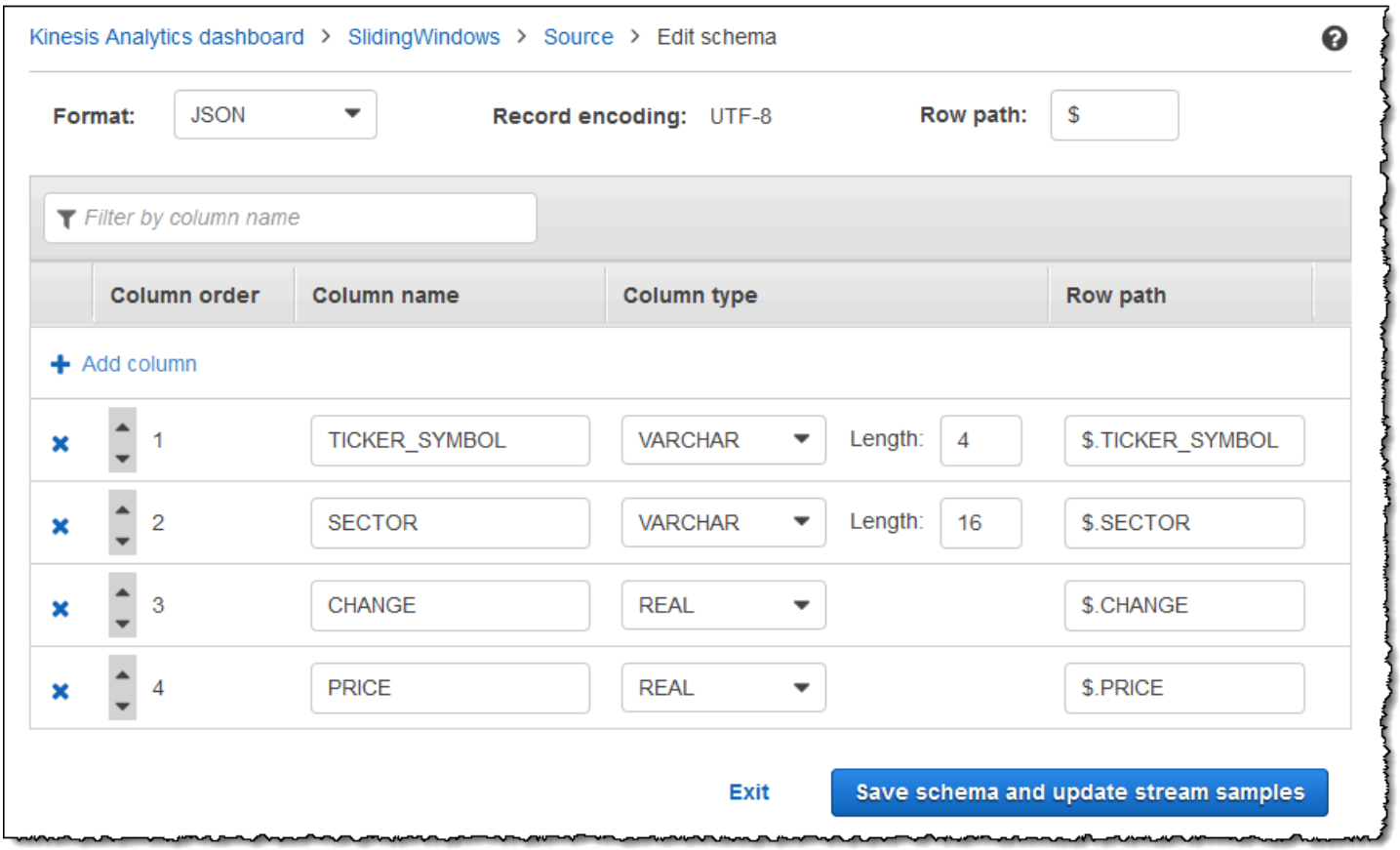
2. Pour supprimer une colonne, choisissez l'icône X en regard du numéro de la colonne.



3. Pour renommer une colonne, tapez son nouveau nom sous Nom de la colonne. Le nouveau nom de la colonne ne peut pas être vide, doit comporter plus d'un caractère et ne doit pas contenir de mots réservés SQL. Il doit également respecter les critères de dénomination des identifiants ordinaires pour SQL : Le nom doit commencer par une lettre et contenir uniquement des lettres, des traits de soulignement et des chiffres.

4. Pour changer le type de données d'une colonne, choisissez un nouveau type de données sous Column type. Si le nouveau type de données est `CHAR`, `VARBINARY` ou `VARCHAR`, indiquez une longueur de données sous Length (Longueur). Pour plus d'informations, consultez Types de données.
5. Choisissez Save schema and update stream pour enregistrer vos modifications.

Le schéma modifié s'affiche dans l'éditeur et ressemble à l'exemple suivant.



Si votre schéma comporte plusieurs lignes, vous pouvez les filtrer à l'aide de Filter by column name. Par exemple, pour modifier les noms des colonnes débutant par `P` (colonne `Price` par exemple), saisissez `P` dans la zone Filter by column name (Filtrer par nom de colonne).

Modification d'un schéma CSV

Vous pouvez modifier un schéma CSV en procédant comme suit.

Pour modifier un schéma CSV

1. Dans l'éditeur de schéma, sous Row delimiter, choisissez le séparateur utilisé par votre flux de données entrantes. Il s'agit du séparateur utilisé entre les enregistrements de données de votre flux (par exemple, un caractère de nouvelle ligne).
2. Sous Column delimiter, choisissez le séparateur utilisé par votre flux de données entrantes. Il s'agit du séparateur utilisé entre les champs de données de votre flux (par exemple, une virgule).
3. Pour ajouter une colonne, choisissez Add column.

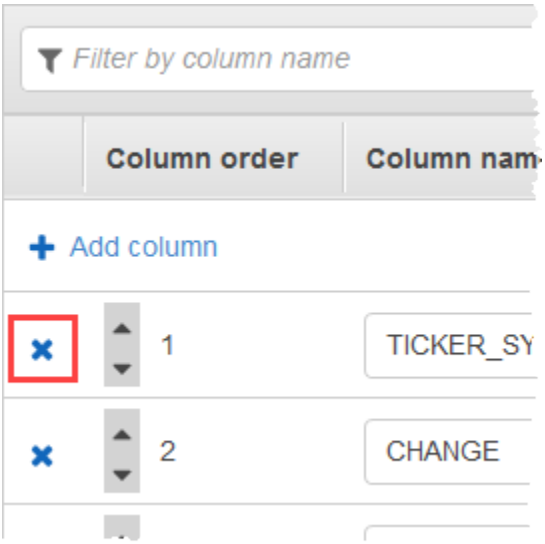
   Une nouvelle colonne s'affiche en première position. Pour modifier l'ordre des colonnes, choisissez les flèches haut et bas en regard du nom de la colonne.

   Fournissez les informations suivantes pour la nouvelle colonne :

   - Sous Nom de la colonne, saisissez un nom.

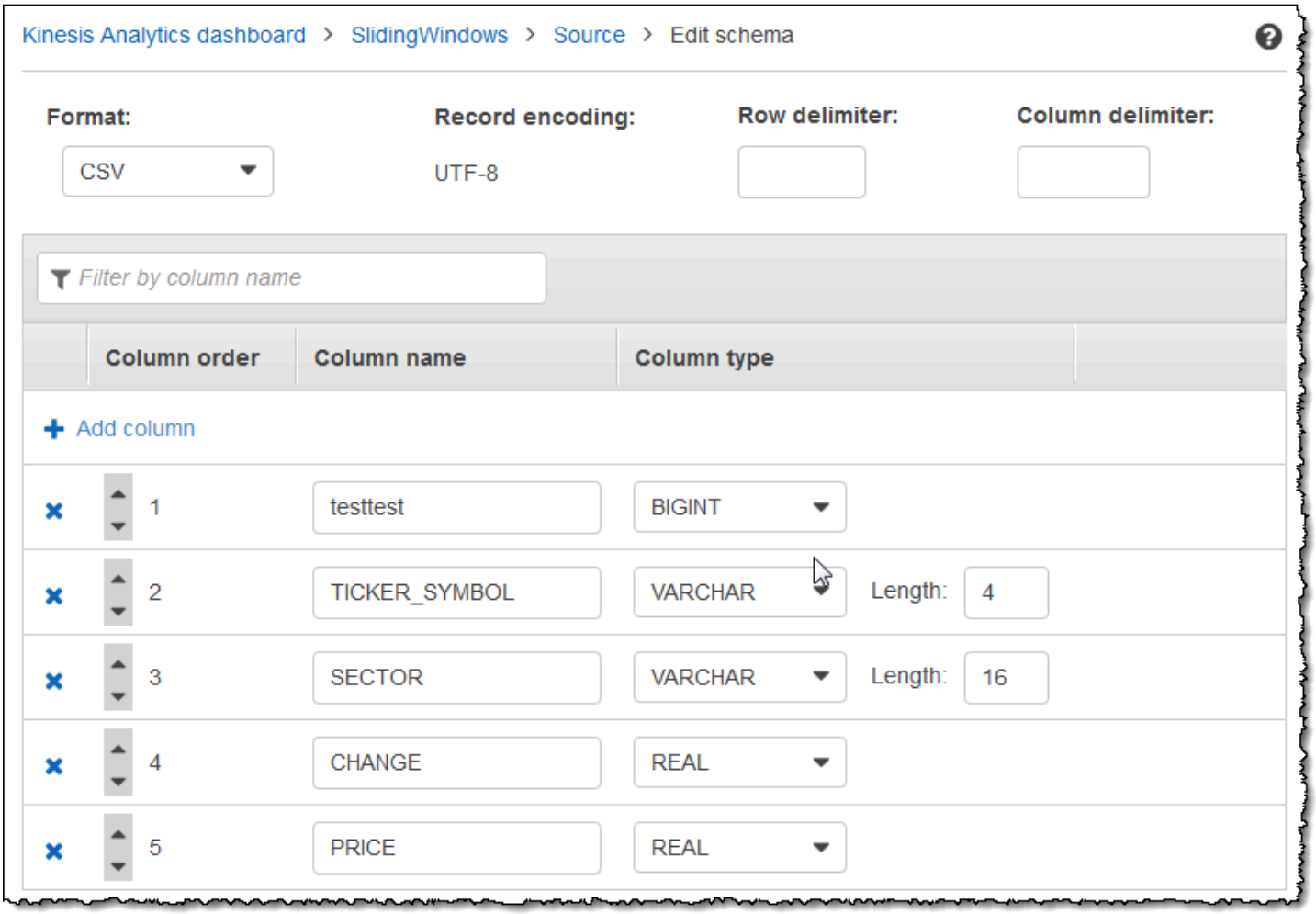
     Un nom de colonne ne peut pas être vide, doit comporter plus d'un caractère et ne doit pas contenir de mots réservés SQL. Il doit également respecter les critères de dénomination des identifiants ordinaires pour SQL : Le nom doit commencer par une lettre et contenir uniquement des lettres, des traits de soulignement et des chiffres.
   - Sous Type de colonne, saisissez un type de données SQL.

     Le type de colonne peut être n'importe quel type de données SQL pris en charge. Si le nouveau type de données est CHAR, VARBINARY ou VARCHAR, indiquez une longueur de données sous Length. Pour plus d'informations, consultez Types de données.
4. Pour supprimer une colonne, choisissez l'icône X en regard du numéro de la colonne.



5. Pour renommer une colonne, tapez son nouveau nom sous Nom de la colonne. Le nouveau nom de la colonne ne peut pas être vide, doit comporter plus d'un caractère et ne doit pas contenir de mots réservés SQL. Il doit également respecter les critères de dénomination des identifiants ordinaires pour SQL : Le nom doit commencer par une lettre et contenir uniquement des lettres, des traits de soulignement et des chiffres.
6. Pour changer le type de données d'une colonne, choisissez un nouveau type de données sous Column type. Si le nouveau type de données est CHAR, VARBINARY ou VARCHAR, indiquez une longueur de données sous Length. Pour plus d'informations, consultez Types de données.
7. Choisissez Save schema and update stream pour enregistrer vos modifications.

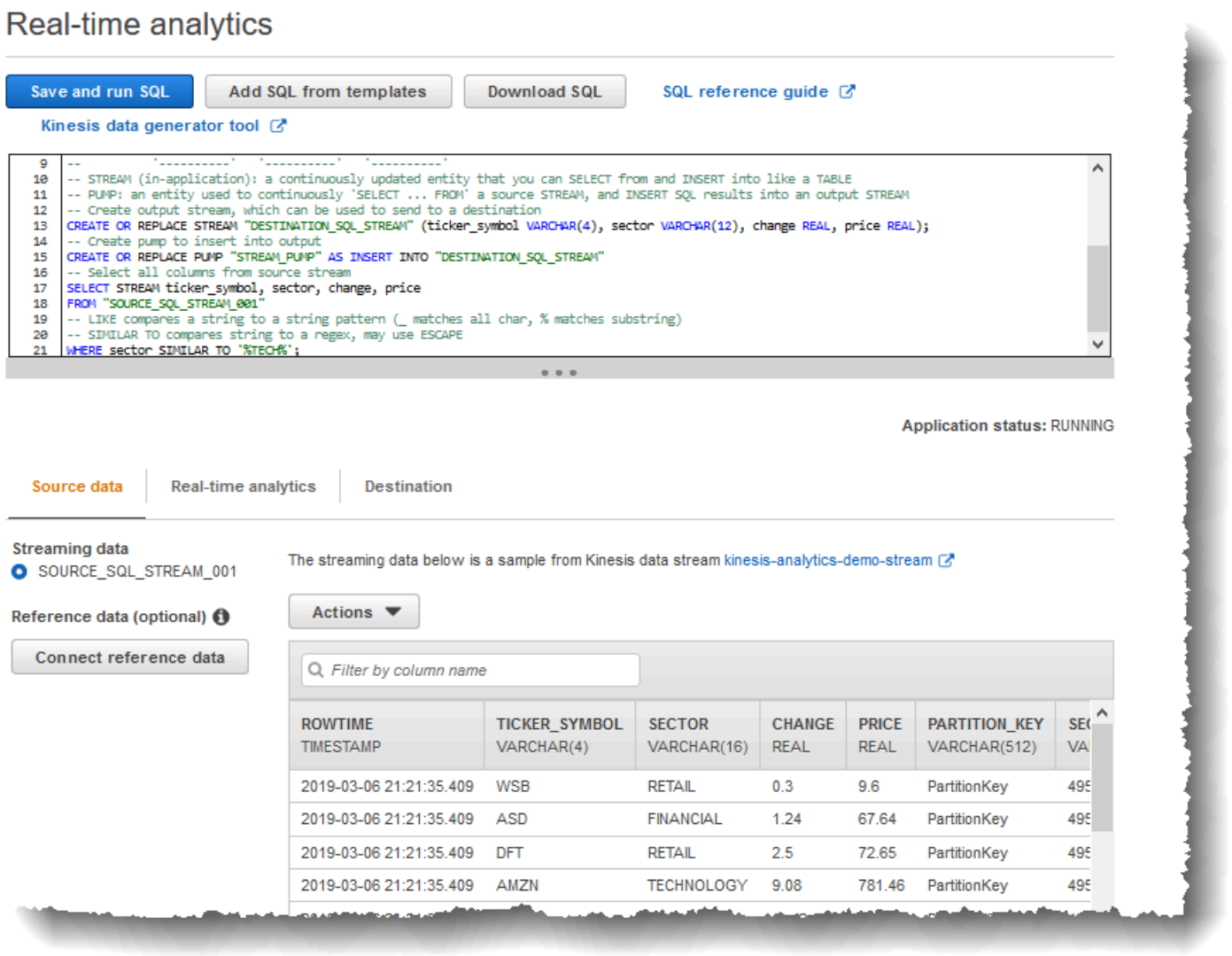
Le schéma modifié s'affiche dans l'éditeur et ressemble à l'exemple suivant.



Si votre schéma comporte plusieurs lignes, vous pouvez les filtrer à l'aide de Filter by column name. Par exemple, pour modifier les noms des colonnes débutant par P (colonne `Price` par exemple), saisissez P dans la zone Filter by column name (Filtrer par nom de colonne).

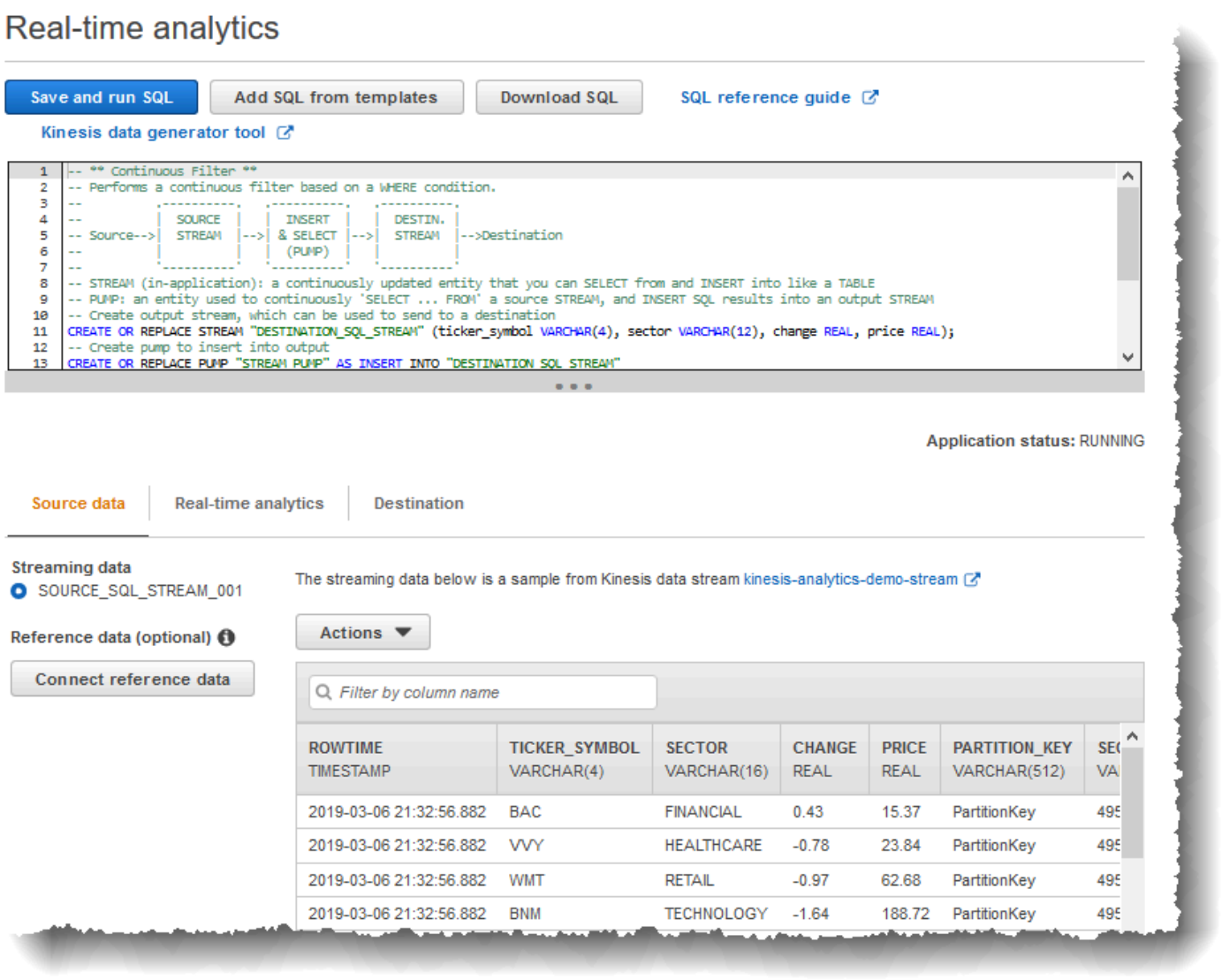
Utilisation de l'éditeur SQL

Vous trouverez ci-dessous des informations sur les sections de l'éditeur SQL et sur leur fonctionnement. Dans l'éditeur SQL, vous pouvez soit créer votre propre code vous-même soit choisir Add SQL from templates. Un modèle SQL est un exemple de code SQL que vous pouvez utiliser pour écrire des applications Amazon Kinesis Data Analytics courantes. Les exemples d'applications de ce guide utilisent certains de ces modèles. Pour plus d'informations, consultez Exemples de Kinesis Data Analytics pour SQL.



Onglet Source Data

L'onglet Source data identifie une source de streaming. Il identifie également le flux d'entrée intégré à l'application auquel celle-ci est mappée et fournit la configuration d'entrée de l'application.



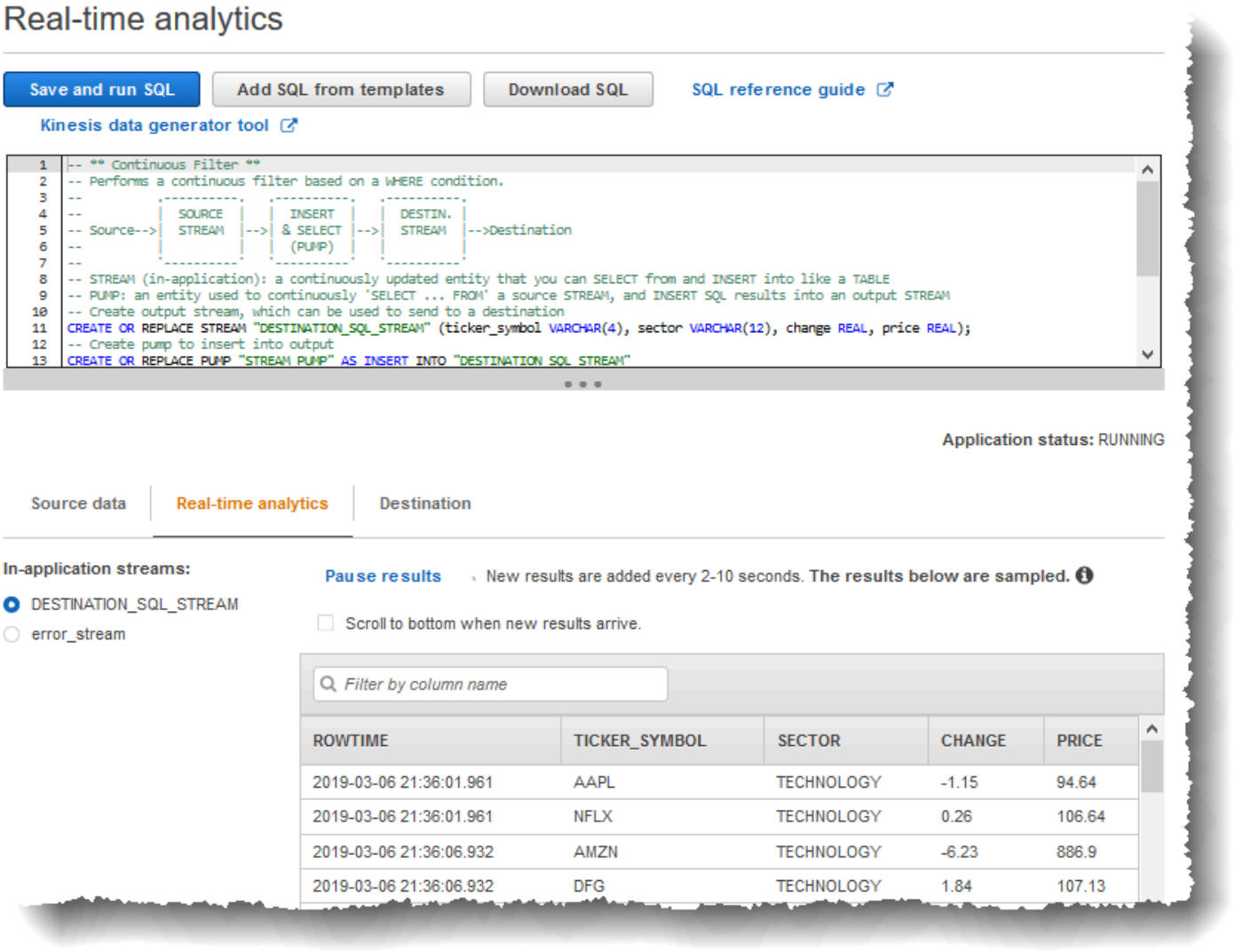
Amazon Kinesis Data Analytics fournit les colonnes d'horodatage suivantes. Vous n'avez donc pas besoin d'indiquer un mappage explicite dans votre configuration d'entrée :

- ROWTIME : chaque ligne d'un flux intégré à l'application comporte une colonne spéciale appelée `ROWTIME`. Cette colonne est l'horodatage du moment où Kinesis Data Analytics a inséré la ligne dans le premier flux intégré à l'application.
- Approximate_Arrival_Time : les enregistrements de votre source de streaming incluent la colonne `Approximate_Arrival_Timestamp`. Il s'agit de l'horodatage d'arrivée approximatif défini lorsque la source de streaming reçoit et stocke l'enregistrement associé avec succès. Kinesis Data Analytics extrait cette colonne et l'insère dans le flux d'entrée intégré à l'application en tant que valeur `Approximate_Arrival_Time`. Amazon Kinesis Data Analytics fournit cette colonne uniquement dans le flux d'entrée intégré à l'application qui est mappé à la source de streaming.

Ces valeurs d'horodatage sont utiles dans les requêtes à fenêtres temporelles. Pour plus d'informations, consultez Requêtes à fenêtres.

Onglet Real-Time Analytics

L'onglet Real-time analytics affiche tous les flux intégrés à l'application créés par votre code d'application. Ce groupe de flux inclut également le flux d'erreurs (`error_stream`) qu'Amazon Kinesis Data Analytics fournit pour toutes les applications.



Onglet Destination

L'onglet Destination vous permet de configurer la sortie de l'application pour conserver les flux intégrés à l'application dans des destinations externes. Vous pouvez configurer la sortie pour

conserver les données de n'importe quel flux intégré à l'application dans des destinations externes. Pour plus d'informations, voir Configuration de la sortie d'application.

Concepts du code SQL de streaming

Amazon Kinesis Data Analytics implémente la norme SQL ANSI 2008 avec des extensions. Ces extensions vous permettent de traiter des données de diffusion. Les rubriques suivantes couvrent des concepts clés du code SQL de diffusion.

Rubriques

Flux et pompes intégrés à l'application

Lorsque vous configurez l'entrée d'application, vous mappez une source de diffusion à un flux intégré à l'application, qui est créé. Les données arrivent en continu depuis la source de diffusion vers le flux intégré à l'application. Un flux intégré à l'application fonctionne comme une table que vous pouvez interroger à l'aide d'instructions SQL, mais on l'appelle « flux », car il s'agit d'un flux de données en continu.

> **Note**
>
> Ne confondez pas les flux intégrés à l'application avec les flux de données Amazon Kinesis et les flux de diffusion Firehose. Les flux intégrés à l’application n’existent que dans le contexte d’une application Amazon Kinesis Data Analytics. Les flux de données Kinesis et les flux de diffusion Firehose existent indépendamment de votre application. Vous pouvez les configurer comme source de diffusion dans votre configuration d'entrée d'application ou comme destination dans la configuration de sortie.

Vous pouvez également créer autant de flux intégrés à l'application supplémentaires que nécessaire pour stocker des résultats de requête intermédiaires. La création d'un flux intégré à l'application est

un processus en deux étapes. D'abord, vous créez un flux intégré à l'application, puis vous pompez des données dans celui-ci. Par exemple, supposons que la configuration d'entrée de votre application crée un flux intégré à l'application appelé `INPUTSTREAM`. Dans l'exemple suivant, vous créez un autre flux (`TEMPSTREAM`), puis vous pompez des données depuis `INPUTSTREAM` dans ce flux.

1. Créez un flux intégré à l'application (`TEMPSTREAM`) avec trois colonnes, comme illustré ci-après :

   ```
   CREATE OR REPLACE STREAM "TEMPSTREAM" (
      "column1" BIGINT NOT NULL,
      "column2" INTEGER,
      "column3" VARCHAR(64));
   ```

   Les noms de colonne sont spécifiés entre guillemets, ce qui les rend sensibles à la casse. Pour plus d'informations, consultez Identificateurs dans le manuel Référence SQL Amazon Kinesis Data Analytics.

2. Insérez des données dans le flux à l'aide d'une pompe. Une pompe est une requête d'insertion s'exécutant en continu qui insère des données d'un flux intégré à l'application vers un autre flux intégré à l'application. L'instruction suivante crée une pompe (`SAMPLEPUMP`) et insère des données dans le flux `TEMPSTREAM` en sélectionnant des enregistrements d'un autre flux (`INPUTSTREAM`).

   ```
   CREATE OR REPLACE PUMP "SAMPLEPUMP" AS
   INSERT INTO "TEMPSTREAM" ("column1",
                             "column2",
                             "column3")
   SELECT STREAM inputcolumn1,
                 inputcolumn2,
                 inputcolumn3
   FROM "INPUTSTREAM";
   ```

Plusieurs enregistreurs peuvent effectuer des insertions dans un flux intégré à l'application et plusieurs lecteurs peuvent être sélectionnés depuis le flux. Vous pouvez considérer un flux intégré à l'application comme l'implémentation d'un paradigme de messagerie de publication/abonnement. Dans ce paradigme, la ligne de données, y compris l'heure de création et l'heure de réception, peut être traitée, interprétée et transmise par une cascade d'instructions SQL de diffusion, sans devoir être stockée dans un SGBDR traditionnel.

Après avoir créé un flux intégré à l'application, vous pouvez exécuter des requêtes SQL normales.

> **Note**
>
> Lors de l'interrogation de flux, la plupart des instructions SQL sont bornées à l'aide d'une fenêtre basée sur des lignes ou d'une fenêtre temporelle. Pour de plus amples informations, veuillez consulter Requêtes à fenêtres.

Vous pouvez également joindre des flux. Pour obtenir des exemples de jointure de flux, consultez Opérations de diffusion de données : Jointures de flux.

Horodatages et colonne ROWTIME

Les flux intégrés à l'application incluent une colonne spéciale appelée `ROWTIME`. Celle-ci stocke un horodatage quand Amazon Kinesis Data Analytics insère une ligne dans le premier flux intégré à l'application. `ROWTIME` reflète l'horodatage du moment où Amazon Kinesis Data Analytics a inséré un enregistrement dans le premier flux intégré à l'application après la lecture de la source de streaming. Cette valeur `ROWTIME` est ensuite gérée tout au long de votre application.

> **Note**
>
> Lorsque vous pompez des enregistrements d'un flux intégré à l'application dans un autre, vous n'avez pas besoin de copier explicitement la colonne `ROWTIME` ; Amazon Kinesis Data Analytics copie cette colonne pour vous.

Amazon Kinesis Data Analytics s'assure que les valeurs `ROWTIME` augmentent de façon monotone. Vous utilisez cet horodatage dans des requêtes à fenêtres temporelles. Pour de plus amples informations, veuillez consulter Requêtes à fenêtres.

Vous pouvez accéder à la colonne ROWTIME dans votre instruction `SELECT` comme à toutes les autres colonnes dans votre flux intégré à l'application. Par exemple :

```
SELECT STREAM ROWTIME,
              some_col_1,
              some_col_2
FROM  SOURCE_SQL_STREAM_001
```

Présentation des différents types d'heure dans les analyses de diffusion

En plus de `ROWTIME`, il existe d'autres types d'heure dans les applications de diffusion en temps réel. Il s'agit des types suivants :

- Heure de l’événement : Horodatage du moment où l’événement s’est produit. Ce type d'heure est parfois appelé heure côté client. Il est souvent préférable d'utiliser cette heure dans les analyses car il s'agit du moment où un événement s'est produit. Cependant, de nombreuses sources d'événements, telles que les téléphones mobiles et les clients web, n'ont pas horloges fiables, ce qui peut entraîner des heures inexactes. En outre, des problèmes de connectivité peuvent provoquer le fait que des enregistrements figurant dans un flux ne sont pas dans le même ordre que celui où les événements se sont produits.

- Heure d’intégration : Horodatage auquel l’enregistrement a été ajouté à la source de streaming. Amazon Kinesis Data Streams inclut un champ appelé `APPROXIMATE_ARRIVAL_TIME` dans chaque enregistrement qui fournit cet horodatage. Ce type d'heure est parfois appelé heure côté serveur. Cette heure d'intégration est souvent une approximation proche de l'heure de l'événement. Si un retard a lieu dans l'intégration d'enregistrement dans le flux, cela peut conduire à des inexactitudes, qui sont généralement rares. De plus, l'heure d'intégration est rarement dans le mauvais ordre, mais cela peut se produire en raison de la nature distribuée des données de diffusion. Par conséquent, l'heure d'intégration est la réflexion la plus exacte et dans l'ordre de l'heure de l'événement.

- Heure de traitement : Horodatage auquel Amazon Kinesis Data Analytics insère une ligne dans le premier flux intégré à l’application. Amazon Kinesis Data Analytics fournit cet horodatage dans la colonne `ROWTIME` qui existe dans chaque flux intégré à l’application. L’heure de traitement augmente toujours de façon monotone. Mais elle ne sera pas exacte si votre application est en retard. (Si une application est en retard, l'heure de traitement ne reflète pas avec précision l'heure de l'événement). Cette valeur `ROWTIME` est très précise en ce qui concerne l'horloge, mais elle peut ne pas correspondre à l'heure où l'événement s'est réellement produit.

L'utilisation de chacun de ces types d'heure dans des requêtes à fenêtres temporelles présente des avantages et des inconvénients. Nous vous recommandons de sélectionner un ou plusieurs de ces types d'heure, et une stratégie pour traiter les inconvénients correspondant à votre scénario d'utilisation.

> **Note**
>
> Si vous utilisez des fenêtres basées sur les lignes, le temps ne constitue pas un problème et vous pouvez ignorer cette section.

Nous vous recommandons une stratégie à deux fenêtres qui utilise deux types d'heure, `ROWTIME` et l'un des autres types d'heure (heure d'intégration ou de l'événement).

- Utilisez `ROWTIME`comme première fenêtre, qui contrôle la fréquence à laquelle la requête émet les résultats, comme illustré dans l'exemple suivant. Cette valeur n'est pas utilisée comme une heure logique.
- Utilisez l'un des autres types d'heure comme heure logique à associer à vos analyses. Cette heure représente le moment où l'événement s'est produit. Dans l'exemple suivant, l'objectif de l'analyse est de regrouper les enregistrements et de renvoyer un comptage par symbole boursier.

L'avantage de cette stratégie est qu'elle peut utiliser une heure qui représente le moment où l'événement s'est produit. Elle peut correctement procéder au traitement quand votre application est en retard ou quand des événements arrivent dans le mauvais ordre. Si l'application est en retard pour insérer des enregistrements dans le flux intégré à l'application, les enregistrements sont quand-même regroupés par heure logique dans la deuxième fenêtre. La requête utilise `ROWTIME` pour garantir l'ordre de traitement. Les enregistrements qui sont en retard (l'horodatage d'intégration affiche une valeur antérieure à celle de `ROWTIME`) sont également traités avec succès.

Envisagez d'exécuter la requête suivante sur le flux de démonstration utilisé dans l'exercice de mise en route. La requête utilise la clause `GROUP  BY` et renvoie un nombre de symboles boursiers dans une fenêtre bascule d'une minute.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM"
    ("ingest_time"    timestamp,
    "APPROXIMATE_ARRIVAL_TIME" timestamp,
    "ticker_symbol"  VARCHAR(12),
    "symbol_count"        integer);


CREATE OR REPLACE PUMP "STREAM_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND) AS
 "ingest_time",
```

```
        STEP("SOURCE_SQL_STREAM_001".APPROXIMATE_ARRIVAL_TIME BY INTERVAL '60' SECOND)
 AS "APPROXIMATE_ARRIVAL_TIME",
        "TICKER_SYMBOL",
        COUNT(*) AS "symbol_count"
    FROM "SOURCE_SQL_STREAM_001"
    GROUP BY "TICKER_SYMBOL",
        STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND),
        STEP("SOURCE_SQL_STREAM_001".APPROXIMATE_ARRIVAL_TIME BY INTERVAL '60' SECOND);
```

Dans `GROUP BY`, vous regroupez d'abord les enregistrements en fonction de `ROWTIME` dans une fenêtre d'une minute, puis par `APPROXIMATE_ARRIVAL_TIME`.

Les valeurs d'horodatage dans le résultat sont arrondies à l'intervalle de 60 secondes inférieur le plus proche. Le résultat du premier groupe renvoyé par la requête affiche les enregistrements dans la première minute. Le deuxième groupe de résultats renvoyé affiche les enregistrements de la minute suivante en fonction de `ROWTIME`. Le dernier enregistrement indique que l'application était en retard pour insérer l'enregistrement dans le flux intégré à l'application (il affiche une valeur `ROWTIME` en retard par rapport à l'horodatage d'intégration).

```
ROWTIME                 INGEST_TIME     TICKER_SYMBOL  SYMBOL_COUNT

--First one minute window.
2016-07-19 17:05:00.0    2016-07-19 17:05:00.0    ABC      10
2016-07-19 17:05:00.0    2016-07-19 17:05:00.0    DEF      15
2016-07-19 17:05:00.0    2016-07-19 17:05:00.0    XYZ      6
--Second one minute window.
2016-07-19 17:06:00.0    2016-07-19 17:06:00.0    ABC      11
2016-07-19 17:06:00.0    2016-07-19 17:06:00.0    DEF      11
2016-07-19 17:06:00.0    2016-07-19 17:05:00.0    XYZ      1  ***

***late-arriving record, instead of appearing in the result of the
first 1-minute windows (based on ingest_time, it is in the result
of the second 1-minute window.
```

Vous pouvez combiner les résultats pour obtenir un comptage exact final par minute en transmettant les résultats à une base de données en aval. Par exemple, vous pouvez configurer la sortie de l'application pour conserver les résultats dans un flux de diffusion Firehose qui peut écrire dans une table Amazon Redshift. Une fois que les résultats sont dans une table Amazon Redshift, vous pouvez interroger la table pour calculer le groupe de comptage total par `Ticker_Symbol`. Dans la cas d'XYZ, le total est exact (6+1), même si un enregistrement est arrivé en retard.

Requêtes continues

Une requête sur un flux s'exécute en continu sur les données de diffusion. Cette exécution en continu autorise des scénarios, comme la possibilité pour des applications d'interroger un flux et de générer des alertes en continu.

Dans l'exercice de mise en route, vous avez un flux intégré à l'application appelé `SOURCE_SQL_STREAM_001`. Il reçoit en continu des prix d'action en provenance d'un flux de démonstration (flux de données Kinesis). Le schéma est le suivant :

```
(TICKER_SYMBOL VARCHAR(4),
 SECTOR varchar(16),
 CHANGE REAL,
 PRICE REAL)
```

Supposons que vous soyez intéressé par les changements de prix d'action supérieurs à 15 %. Vous pouvez utiliser la requête suivante dans votre code d'application. Cette requête s'exécute en continu et émet des enregistrements lorsqu'un changement de prix d'une action supérieur à 15 % est détecté.

```
SELECT STREAM TICKER_SYMBOL, PRICE
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  (ABS((CHANGE / (PRICE-CHANGE)) * 100)) > 15
```

Utilisez la procédure suivante pour configurer une application Amazon Kinesis Data Analytics et tester cette requête.

Pour tester la requête

1. Créez une application en suivant l'exercice de mise en route.
2. Remplacez l'instruction `SELECT` dans le code d'application par la requête `SELECT` précédente. Le code d'application résultant est présenté ci-après :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ticker_symbol VARCHAR(4),
                                                   price DOUBLE);
-- CREATE OR REPLACE PUMP to insert into output
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM TICKER_SYMBOL,
                    PRICE
```

```
FROM   "SOURCE_SQL_STREAM_001"
WHERE  (ABS((CHANGE / (PRICE-CHANGE)) * 100)) > 15;
```

Requêtes à fenêtres

Les requêtes SQL dans votre code d'application s'exécutent en continu sur des flux intégrés à l'application. Un flux intégré à l'application représente des données illimitées qui circulent en continu dans votre application. Par conséquent, pour obtenir des ensembles de résultats de cette entrée continuellement mise à jour, vous bornez souvent des requêtes à l'aide d'une fenêtre définie en termes de temps ou de lignes. Ces requêtes sont également appelées SQL à fenêtres.

Pour une requête à fenêtres temporelle, vous spécifiez la durée de la fenêtre (par exemple, une fenêtre d'une minute). Cela nécessite une colonne d'horodatage dans votre flux intégré à l'application qui augmente de façon monotone. (L'horodatage d'une nouvelle ligne est supérieur ou égal à celui de la ligne précédente.) Amazon Kinesis Data Analytics fournit cette colonne d'horodatage appelée `ROWTIME` pour chaque flux intégré à l'application. Vous pouvez utiliser cette colonne lors de la spécification de requêtes temporelles. Pour votre application, vous pouvez choisir une autre option d'horodatage. Pour de plus amples informations, veuillez consulter Horodatages et colonne ROWTIME.

Pour une requête à fenêtres basées sur les lignes, vous spécifiez la longueur de la fenêtre en termes de nombre de lignes.

Vous pouvez spécifier une requête pour traiter des enregistrements dans une fenêtre bascule, une fenêtre défilante ou une fenêtre stagger, en fonction des besoins de votre application. Kinesis Data Analytics prend en charge les types de fenêtres suivants :

- Stagger Windows : requête qui regroupe les données à l'aide de fenêtres clés basées sur la durée qui s'ouvrent à mesure que les données arrivent. Les clés permettent le chevauchement de plusieurs fenêtres. Il s'agit de la méthode recommandée pour agréger les données à l'aide de fenêtres temporelles, car les fenêtres Stagger réduisent les retards ou les out-of-order données par rapport aux fenêtres Tumbling.
- Fenêtres bascules : requête qui regroupe les données à l'aide de différentes fenêtres basées sur la durée qui s'ouvrent et se ferment à intervalles réguliers.
- Fenêtres défilantes : requête qui regroupe les données en continu, à l'aide d'un intervalle de temps fixe ou d'un intervalle rowcount.

Stagger Windows

Les fenêtres stagger sont une méthode de fenêtrage adaptée à l'analyse des groupes de données qui arrivent à intervalles irréguliers. Elle est idéale pour les cas d'utilisation d'analyse de données chronologiques, comme un ensemble de ventes liées ou d'enregistrements de journaux.

Par exemple, les journaux de flux VPC ont une fenêtre de capture d'environ 10 minutes. Mais elle peut atteindre 15 minutes si vous regroupez des données sur le client. Les fenêtres stagger sont idéales pour agréger ces journaux à des fins d'analyse.

Stagger Windows résout le problème d'enregistrements connexes n'entrant pas dans la même fenêtre de temps, comme lors de l'utilisation de fenêtres à bascule.

Résultats partiels avec les fenêtres à bascule

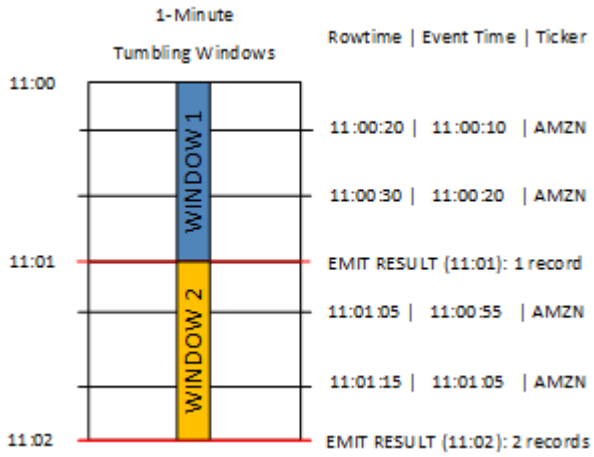
L'utilisation de Fenêtres bascules pour l'agrégation de données tardives ou obsolètes présente certaines limites.

Si les fenêtres bascules sont utilisées pour analyser les groupes de données connexes, les enregistrements individuels peuvent entrer dans des fenêtres distinctes. Par conséquent, les résultats partiels de chaque fenêtre doivent être combinés ultérieurement pour obtenir les résultats complets pour chaque groupe d'enregistrements.

Dans la requête de fenêtre à bascule suivante, les enregistrements sont regroupés en fenêtres par valeur rowtime, heure d'événement et symbole boursier :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    TICKER_SYMBOL VARCHAR(4),
    EVENT_TIME timestamp,
    TICKER_COUNT     DOUBLE);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM
        TICKER_SYMBOL,
        FLOOR(EVENT_TIME TO MINUTE),
        COUNT(TICKER_SYMBOL) AS TICKER_COUNT
    FROM "SOURCE_SQL_STREAM_001"
    GROUP BY ticker_symbol, FLOOR(EVENT_TIME TO MINUTE),
 STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '1' MINUTE);
```

Dans le schéma suivant, une application compte le nombre de transactions qu'elle reçoit, en fonction de la date à laquelle l'échange s'est produit (événement), avec une minute de granularité. L'application peut utiliser une fenêtre à bascule pour le regroupement des données en fonction de la chronologie et de l'événement. L'application reçoit quatre enregistrements qui arrivent les uns après les autres dans un délai d'une minute. Elle regroupe les enregistrements par valeur rowtime, heure d'événement et symbole boursier. Comme certains enregistrements arrivent après la fin de la première fenêtre à bascule, les enregistrements n'interviennent pas dans la même fenêtre à bascule d'une minute.



Le schéma précédent comporte les événements suivants.

ROWTIME	EVENT_TIME	TICKER_SYMBOL
11:00:20	11:00:10	AMZN
11:00:30	11:00:20	AMZN
11:01:05	11:00:55	AMZN
11:01:15	11:01:05	AMZN

L'ensemble de résultats de l'application de fenêtre à bascule se présente comme suit.

ROWTIME	EVENT_TIME	TICKER_SYMBOL	COUNT
11:01:00	11:00:00	AMZN	2

ROWTIME	EVENT_TIME	TICKER_SYMBOL	COUNT
11:02:00	11:00:00	AMZN	1
11:02:00	11:01:00	AMZN	1

Dans le jeu de résultats précédent, trois résultats sont renvoyée :

- Un enregistrement avec un `ROWTIME` de 11:01:00 qui regroupe les deux premiers enregistrements.
- Un enregistrement à 11:02:00 qui regroupe uniquement le troisième enregistrement. Cet enregistrement possède un `ROWTIME` dans la deuxième fenêtre, mais un `EVENT_TIME` dans la première fenêtre.
- Un enregistrement à 11:02:00 qui regroupe uniquement le quatrième enregistrement.

Pour analyser les résultats complets, les enregistrements doivent être regroupés dans le magasin de persistance. Cela ajoute de la complexité et des exigences en matière de traitement à l'application.

Résultats complets avec Stagger Windows

Pour améliorer la précision de l'analyse des enregistrements de données temporelles, Kinesis Data Analytics offre un nouveau type de fenêtre appelé fenêtre Stagger. Dans ce type de fenêtre, les fenêtres s'ouvrent lorsque le premier événement correspondant à la clé de partition se produit, et non pas sur un intervalle de temps fixe. Les fenêtres se ferment en fonction de l'âge spécifié, qui est mesuré à partir du moment où la fenêtre s'est ouverte.

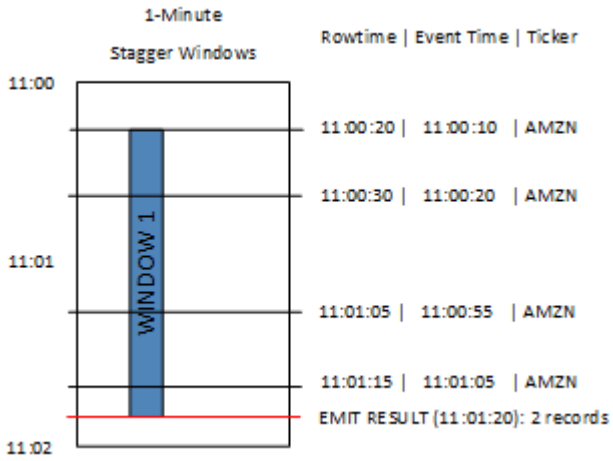
Une fenêtre stagger est une fenêtre distincte limitée dans le temps pour chaque regroupement clé d'une clause de fenêtre. L'application regroupe chaque résultat de la clause de fenêtre à l'intérieur de sa propre fenêtre temporelle, au lieu d'utiliser une seule fenêtre pour tous les résultats.

Dans la requête de fenêtre stagger suivante, les enregistrements sont regroupés en fenêtres par rowtime, heure d'événement et symbole boursier :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    ticker_symbol    VARCHAR(4),
    event_time       TIMESTAMP,
    ticker_count     DOUBLE);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
```

```
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM
        TICKER_SYMBOL,
        FLOOR(EVENT_TIME TO MINUTE),
        COUNT(TICKER_SYMBOL) AS ticker_count
    FROM "SOURCE_SQL_STREAM_001"
    WINDOWED BY STAGGER (
            PARTITION BY FLOOR(EVENT_TIME TO MINUTE), TICKER_SYMBOL RANGE INTERVAL '1'
 MINUTE);
```

Dans le schéma suivant, les enregistrements sont regroupés en fenêtres stagger par heure d'événement et symbole boursier :



Le schéma précédent comporte les événements suivants, qui sont les mêmes événements que ceux analysés par l'application de fenêtre à bascule :

ROWTIME	EVENT_TIME	TICKER_SYMBOL
11:00:20	11:00:10	AMZN
11:00:30	11:00:20	AMZN
11:01:05	11:00:55	AMZN
11:01:15	11:01:05	AMZN

L'ensemble de résultats de l'application de fenêtre stagger se présente comme suit.

ROWTIME	EVENT_TIME	TICKER_SYMBOL	Nombre
11:01:20	11:00:00	AMZN	3
11:02:15	11:01:00	AMZN	1

Les enregistrements retournés regroupent les trois premiers enregistrements d'entrée. Les enregistrements sont regroupés par fenêtres stagger de 1 minute. La fenêtre stagger démarre lorsque l'application reçoit le premier enregistrement AMZN (avec un `ROWTIME` de 11:00:20). Lorsque la fenêtre stagger de 1 minute expire à 11:01:20, un enregistrement avec les résultats qui se trouvent dans la fenêtre stagger (en fonction de `ROWTIME` et d'`EVENT_TIME`) est écrit dans le flux de sortie. À l'aide d'une fenêtre stagger, tous les enregistrements avec un `ROWTIME` et un `EVENT_TIME` dans une fenêtre d'une minute sont émis en un seul résultat.

Le dernier enregistrement (avec un `EVENT_TIME` une fois passée l'agrégation d'une minute) est regroupé séparément. En effet, `EVENT_TIME` est l'une des clés de partition utilisées pour séparer les enregistrements en jeux de résultats, et la clé de partition `EVENT_TIME` pour la première fenêtre est `11:00`.

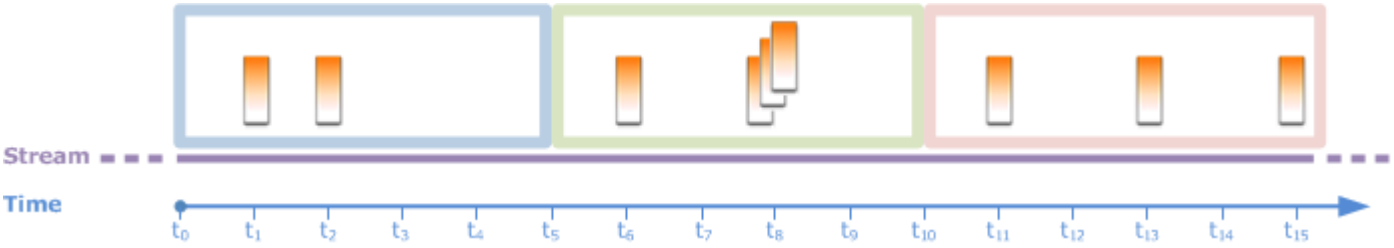
La syntaxe d'une fenêtre stagger est définie dans une clause spéciale, `WINDOWED BY`. Cette clause est utilisée au lieu de la clause `GROUP BY` pour les regroupements de streaming. La clause s'affiche immédiatement après la clause `WHERE` facultative et avant la clause `HAVING`.

La fenêtre stagger est définie par la clause `WINDOWED BY` et accepte deux paramètres : les clés de partition et la longueur de la fenêtre. Les clés de partition partitionnent les flux de données entrantes et définissent à quel moment la fenêtre s'ouvre. Une fenêtre stagger s'ouvre lorsque le premier événement avec une clé de partition unique s'affiche dans le flux. La fenêtre stagger se ferme après une période définie, par la longueur de la fenêtre. Cette syntaxe est illustrée dans l'exemple de code suivant.

```
...
FROM <stream-name>
WHERE <... optional statements...>
WINDOWED BY STAGGER(
 PARTITION BY <partition key(s)>
 RANGE INTERVAL <window length, interval>
);
```

Fenêtres bascules (regroupements à l'aide de GROUP BY)

Lorsqu'une requête à fenêtres traite chaque fenêtre sans chevauchement, la fenêtre est appelée fenêtre bascule. Dans ce cas, chaque enregistrement d'un flux intégré à l'application appartient à une fenêtre spécifique. Il n'est traité qu'une fois (lorsque la requête traite la fenêtre à laquelle l'enregistrement appartient).



Par exemple, une requête de regroupement à l'aide d'une clause `GROUP BY` traite des lignes dans une fenêtre bascule. Le flux de démonstration de l'exercice de mise en route reçoit des données de prix d'action qui sont mappées au flux intégré à l'application `SOURCE_SQL_STREAM_001` dans votre application. Ce flux a le schéma suivant :

```
(TICKER_SYMBOL VARCHAR(4),
 SECTOR varchar(16),
 CHANGE REAL,
 PRICE REAL)
```

Dans votre code d'application, supposons que vous vouliez trouver des prix agrégés (minimum, maximum) pour chaque symbole boursier sur une fenêtre d'une minute. Vous pouvez utiliser la requête suivante.

```
SELECT STREAM ROWTIME,
              Ticker_Symbol,
              MIN(Price) AS Price,
              MAX(Price) AS Price
FROM     "SOURCE_SQL_STREAM_001"
GROUP BY Ticker_Symbol,
         STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

La requête précédente est un exemple de requête à fenêtres temporelle. La requête regroupe les enregistrements par valeurs `ROWTIME`. Une génération de rapports sur une base par minute, la fonction `STEP` arrondit vers le bas les valeurs `ROWTIME` à la minute la plus proche.

> **Note**
>
> Vous pouvez également utiliser la fonction FLOOR pour regrouper les enregistrements dans des fenêtres. Cependant, FLOOR peut uniquement arrondir les valeurs de temps à l'unité de temps entière inférieure (heure, minute, seconde, etc.). La fonction STEP est recommandée pour regrouper les enregistrements dans des fenêtres bascules, car elle peut arrondir des valeurs à un intervalle inférieur arbitraire, par exemple 30 secondes.

Cette requête est un exemple de fenêtre sans chevauchement (bascule). La clause GROUP BY regroupe les enregistrements dans une fenêtre d'une minute et chaque enregistrement appartient à une fenêtre spécifique (pas de chevauchement). La requête émet un enregistrement de sortie par minute, fournissant le prix d'un symbole boursier minimum/maximum enregistré dans la minute spécifique. Ce type de requête est utile pour générer des rapports périodiques à partir du flux de données d'entrée. Dans cet exemple, des rapports sont générés toutes les minutes.

Pour tester la requête

1. Configurez une application en suivant l'exercice de mise en route.
2. Remplacez l'instruction SELECT dans le code d'application par la requête SELECT précédente. Le code d'application résultant est présenté ci-après :

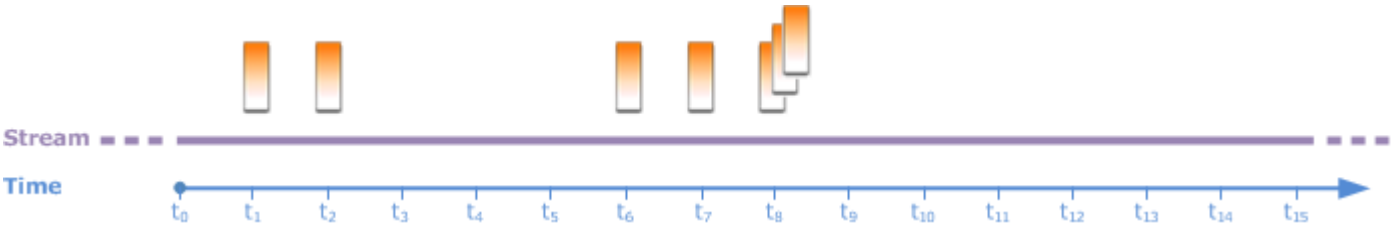
```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                                    ticker_symbol VARCHAR(4),
                                    Min_Price     DOUBLE,
                                    Max_Price     DOUBLE);
-- CREATE OR REPLACE PUMP to insert into output
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM Ticker_Symbol,
                  MIN(Price) AS Min_Price,
                  MAX(Price) AS Max_Price
    FROM    "SOURCE_SQL_STREAM_001"
    GROUP BY Ticker_Symbol,
             STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

Fenêtres défilantes

Au lieu de regrouper des enregistrements à l'aide de `GROUP BY`, vous pouvez définir une fenêtre temporelle ou basées sur les lignes. Pour ce faire, vous pouvez ajouter une clause `WINDOW` explicite.

Dans ce cas, au fur et à mesure que la fenêtre défile dans le temps, Amazon Kinesis Data Analytics émet une sortie lorsque de nouveaux enregistrements apparaissent dans le flux. Kinesis Data Analytics émet cette sortie en traitant les lignes de la fenêtre. Les fenêtres peuvent se chevaucher dans ce type de traitement et un enregistrement peut faire partie de plusieurs fenêtres et être traité avec chaque fenêtre. L'exemple suivant illustre une fenêtre défilante.

Prenons l'exemple d'une requête simple qui compte les enregistrements du flux. Pour cet exemple, supposons une fenêtre de 5 secondes. Dans l'exemple de flux suivant, de nouveaux enregistrements arrivent aux instants t_1, t_2, t_6 et t_7, et trois enregistrements à l'instant t_8 secondes.



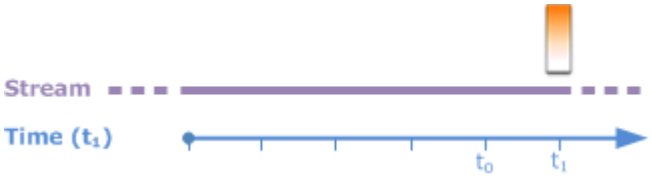
Gardez à l'esprit les points suivants :

- Pour cet exemple, supposons une fenêtre de 5 secondes. Cette fenêtre défile en continu avec le temps.
- Pour chaque ligne qui entre dans une fenêtre, une ligne de sortie est émise par la fenêtre défilante. Peu de temps après le démarrage de l'application, vous voyez la requête émettre une sortie pour chaque nouvel enregistrement qui s'affiche dans le flux, même si la fenêtre de 5 secondes ne s'est pas encore écoulée. Par exemple, la requête émet une sortie lorsqu'un enregistrement apparaît à la première et à la deuxième secondes. Puis, elle traite les enregistrements dans la fenêtre de 5 secondes.
- Les fenêtres défilent avec le temps. Si un ancien enregistrement du flux se trouve en dehors de la fenêtre, la requête n'émet aucune sortie, à moins qu'un nouvel enregistrement apparaisse dans le flux dans cette fenêtre de 5 secondes.

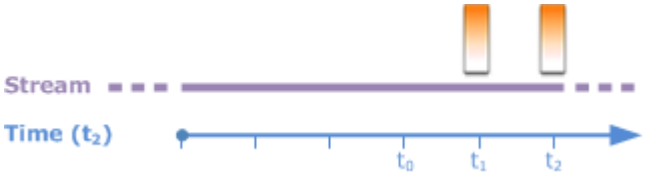
Supposons que la requête commence à s'exécuter à t_0. Les actions suivantes se produisent :

1. À l'instant t_0, la requête démarre. La requête n'émet aucune sortie (valeur de comptage), car il n'y a aucun enregistrement à cet instant.

2. À l'instant t_1, un nouvel enregistrement apparaît dans le flux et la requête émet la valeur de comptage 1.

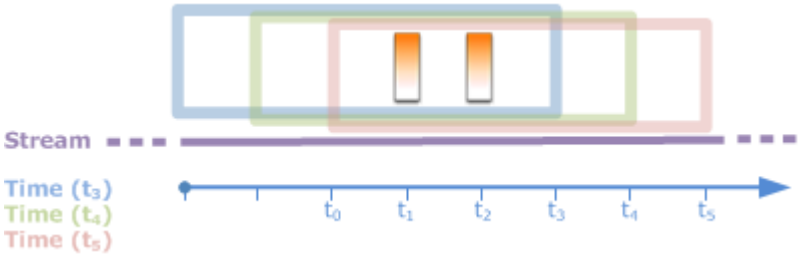


3. À l'instant t_2, un autre enregistrement apparaît, et la requête émet le nombre 2.

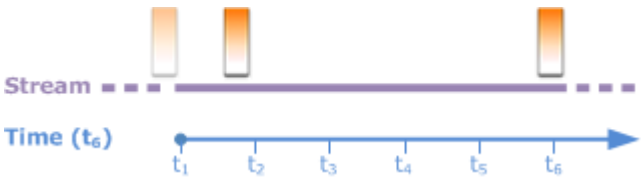


4. La fenêtre de 5 secondes défile avec le temps :
   - À t_3, la fenêtre défilante t_3 à t_0
   - À t4 (fenêtre défilante t_4 à t_0)
   - À t_5, la fenêtre défilante t_5 à t_0

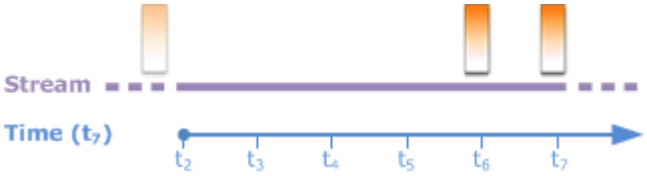
   À tous ces instants, la fenêtre de 5 secondes comporte les mêmes enregistrement, sil n'y a aucun nouvel enregistrement. La requête n'émet donc aucune sortie.



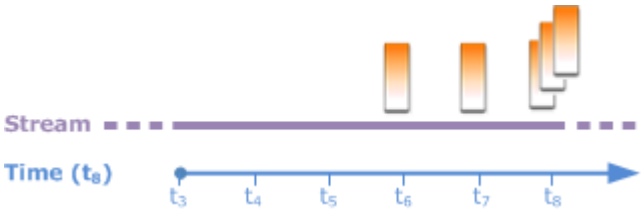
5. À l'instant t_6, la fenêtre de 5 secondes est (t_6 à t_1). La requête détecte un nouvel enregistrement à l'instant t_6 et émet donc la sortie 2. L'enregistrement à t_1 n'est plus dans la fenêtre et n'est donc pas comptabilisé.



6. À l'instant t_7, la fenêtre de 5 secondes est t_7 à t_2. La requête détecte un nouvel enregistrement à l'instant t_7 et émet donc la sortie 2. L'enregistrement à t_2 n'est plus dans la fenêtre de 5 secondes et n'est donc pas comptabilisé.



7. À l'instant t_8, la fenêtre de 5 secondes est t_8 à t_3. La requête détecte trois nouveaux enregistrements et comptabilise donc 5 enregistrements.



Pour résumer, la fenêtre est de taille fixe et défile avec le temps. La requête émet une sortie lorsque de nouveaux enregistrements apparaissent.

> **Note**
>
> Nous vous recommandons d'utiliser une fenêtre coulissante d'une heure maximum. Si vous utilisez une fenêtre plus longue, le redémarrage de l'application sera plus long après la maintenance standard du système. En effet, la source de données doit de nouveau être lue à partir du flux.

Voici des exemples de requêtes qui utilisent la clause `WINDOW` pour définir des fenêtres et effectuer des regroupements. Comme les requêtes ne spécifient pas `GROUP BY`, elles utilisent l'approche de fenêtre défilante pour traiter les enregistrements du flux.

Exemple 1 : Traitement d'un flux à l'aide d'une fenêtre défilante de 1 minute

Prenons le flux de démonstration de l'exercice de mise en route qui remplit le flux intégré à l'application `SOURCE_SQL_STREAM_001`. Voici le schéma.

```
(TICKER_SYMBOL VARCHAR(4),
```

```
  SECTOR varchar(16),
  CHANGE REAL,
  PRICE REAL)
```

Supposons que vous vouliez que votre application effectue des regroupements à l'aide d'une fenêtre défilante de 1 minute. En d'autres termes, pour chaque nouvel enregistrement qui apparaît dans le flux, vous souhaitez que l'application émette une sortie en appliquant des regroupements aux enregistrements de la fenêtre de 1 minute précédente.

Vous pouvez utiliser la requête à fenêtres temporelle suivante. La requête utilise la clause `WINDOW` pour définir l'intervalle de 1 minute. `PARTITION BY` dans la clause `WINDOW` regroupe les enregistrements par valeurs de symbole boursier dans la fenêtre défilante.

```
SELECT STREAM ticker_symbol,
              MIN(Price) OVER W1 AS Min_Price,
              MAX(Price) OVER W1 AS Max_Price,
              AVG(Price) OVER W1 AS Avg_Price
FROM   "SOURCE_SQL_STREAM_001"
WINDOW W1 AS (
   PARTITION BY ticker_symbol
   RANGE INTERVAL '1' MINUTE PRECEDING);
```

Pour tester la requête

1. Configurez une application en suivant l'exercice de mise en route.
2. Remplacez l'instruction `SELECT` dans le code d'application par la requête `SELECT` précédente. Le code d'application résultant est le suivant.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                        ticker_symbol VARCHAR(10),
                        Min_Price     double,
                        Max_Price     double,
                        Avg_Price     double);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
     SELECT STREAM ticker_symbol,
                   MIN(Price) OVER W1 AS Min_Price,
                   MAX(Price) OVER W1 AS Max_Price,
                   AVG(Price) OVER W1 AS Avg_Price
     FROM   "SOURCE_SQL_STREAM_001"
     WINDOW W1 AS (
```

```
        PARTITION BY ticker_symbol
        RANGE INTERVAL '1' MINUTE PRECEDING);
```

Exemple 2 : Requête appliquant des regroupements sur une fenêtre défilante

La requête suivante sur le flux de démonstration retourne la variation moyenne en pourcentage du prix de chaque symbole boursier dans une fenêtre de 10 secondes.

```
SELECT STREAM Ticker_Symbol,
             AVG(Change / (Price - Change)) over W1 as Avg_Percent_Change
FROM "SOURCE_SQL_STREAM_001"
WINDOW W1 AS (
        PARTITION BY ticker_symbol
        RANGE INTERVAL '10' SECOND PRECEDING);
```

Pour tester la requête

1. Configurez une application en suivant l'exercice de mise en route.
2. Remplacez l'instruction SELECT dans le code d'application par la requête SELECT précédente. Le code d'application résultant est le suivant.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                            ticker_symbol VARCHAR(10),
                            Avg_Percent_Change double);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM Ticker_Symbol,
                    AVG(Change / (Price - Change)) over W1 as Avg_Percent_Change
      FROM "SOURCE_SQL_STREAM_001"
      WINDOW W1 AS (
              PARTITION BY ticker_symbol
              RANGE INTERVAL '10' SECOND PRECEDING);
```

Exemple 3 : Interrogation des données à partir de plusieurs fenêtres défilantes sur le même flux

Vous pouvez écrire des requêtes pour émettre une sortie dans laquelle chaque valeur de colonne est calculée à l'aide de différentes fenêtres défilantes définies sur le même flux.

Dans l'exemple suivant, la requête émet une sortie pour le symbole, le prix, a2 et a10. Il émet une sortie pour les symboles boursiers dont la moyenne mobile sur deux lignes croise la moyenne mobile sur dix lignes. Les valeurs de colonne `a2` et `a10` proviennent de fenêtres défilantes à deux et dix lignes.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                           ticker_symbol      VARCHAR(12),
                           price              double,
                           average_last2rows  double,
                           average_last10rows double);

CREATE OR REPLACE PUMP "myPump" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM ticker_symbol,
              price,
              avg(price) over last2rows,
              avg(price) over last10rows
FROM SOURCE_SQL_STREAM_001
WINDOW
    last2rows AS (PARTITION BY ticker_symbol ROWS 2 PRECEDING),
    last10rows AS (PARTITION BY ticker_symbol ROWS 10 PRECEDING);
```

Pour tester cette requête sur le flux de démonstration, suivez la procédure de test décrite dans Exemple 1.

Opérations de diffusion de données : Jointures de flux

Vous pouvez disposer de plusieurs flux intégrés à l'application dans votre application. Vous pouvez écrire des requêtes `JOIN` pour corréler les données qui arrivent dans ces flux. Par exemple, supposons que vous ayez les flux intégrés à l'application suivants :

- OrderStream— Reçoit les commandes de stock passées.

```
(orderId SqlType, ticker SqlType, amount SqlType, ROWTIME TimeStamp)
```

- TradeStream— Reçoit les transactions boursières qui en résultent pour ces ordres.

```
(tradeId SqlType, orderId SqlType, ticker SqlType, amount SqlType, ticker SqlType,
 amount SqlType, ROWTIME TimeStamp)
```

Voici des exemples de requête `JOIN` qui corrèlent les données de ces flux.

Exemple 1 : Génération de rapport en cas d'opérations à moins d'une minute du placement de l'ordre

Dans cet exemple, votre requête joint les deux flux `OrderStream` et `TradeStream`. Cependant, comme nous voulons uniquement les opérations placées dans un délai d'une minute après les ordres, la requête définit la fenêtre de 1 minute sur le flux `TradeStream`. Pour plus d'informations sur les requêtes à fenêtres, consultez Fenêtres défilantes.

```
SELECT STREAM
     ROWTIME,
     o.orderId, o.ticker, o.amount AS orderAmount,
     t.amount AS tradeAmount
FROM OrderStream AS o
JOIN TradeStream OVER (RANGE INTERVAL '1' MINUTE PRECEDING) AS t
ON   o.orderId = t.orderId;
```

Vous pouvez définir explicitement la fenêtre en utilisant la clause `WINDOW` et en écrivant la requête précédente comme suit :

```
SELECT STREAM
    ROWTIME,
    o.orderId, o.ticker, o.amount AS orderAmount,
    t.amount AS tradeAmount
FROM OrderStream AS o
JOIN TradeStream OVER t
ON o.orderId = t.orderId
WINDOW t AS
    (RANGE INTERVAL '1' MINUTE PRECEDING)
```

Lorsque vous incluez cette requête dans votre code d'application, le code d'application s'exécute en continu. Pour chaque enregistrement qui arrive dans le flux `OrderStream`, l'application émet une sortie si des opérations ont lieu dans la fenêtre de 1 minute suivant l'ordre.

La jointure dans la requête précédente est une jointure interne où la requête émet des enregistrements dans `OrderStream` pour lesquels il existe un enregistrement correspondant dans `TradeStream` (et vice versa). A l'aide d'une jointure externe, vous pouvez créer un autre scénario intéressant. Supposons que vous vouliez les ordres de bourse pour lesquels aucune opération n'a lieu dans un délai d'une minute après le placement de l'ordre de bourse, et que des opérations soient

signalées dans la même fenêtre, mais pour certains autres ordres. Il s'agit d'un exemple de jointure externe.

```
SELECT STREAM
    ROWTIME,
    o.orderId, o.ticker, o.amount AS orderAmount,
    t.ticker, t.tradeId, t.amount AS tradeAmount,
FROM OrderStream AS o
LEFT OUTER JOIN TradeStream OVER (RANGE INTERVAL '1' MINUTE PRECEDING) AS t
ON    o.orderId = t.orderId;
```

Exemples de migration vers le service géré pour Apache Flink Studio

Les exemples suivants montrent comment migrer les applications Kinesis Data Analytics pour SQL vers le service géré pour Apache Flink Studio.

Réplication des requêtes Kinesis Data Analytics pour SQL dans un service géré pour Apache Flink Studio

> **Warning**
>
> Pour les nouveaux projets, nous vous recommandons d'utiliser le service géré pour Apache Flink Studio plutôt que les requêtes Kinesis Data Analytics pour SQL. Le service géré pour Apache Flink Studio allie facilité d'utilisation et capacités analytiques avancées, ce qui vous permet de créer des applications sophistiquées de traitement des flux en quelques minutes.

Cette section fournit des conversions de requêtes que vous pouvez utiliser pour les cas d'utilisation courants lors de la migration de vos charges de travail vers le service géré pour Apache Flink Studio ou le service géré pour Apache Flink.

> **Note**
>
> Le service géré pour Apache Flink et le service géré pour Apache Flink Studio proposent des fonctionnalités avancées de traitement des flux de données qui ne sont pas disponibles dans les applications Kinesis Data Analytics basées sur SQL. Cela comprend notamment la sémantique de traitement unique, les fenêtres temporelles d'événements, l'extensibilité grâce à des fonctions définies par l'utilisateur et à des intégrations personnalisées, la prise en charge du langage impératif, l'état durable des applications, la mise à l'échelle horizontale, la prise en charge de plusieurs sources de données, les intégrations extensibles, etc. Ces fonctionnalités sont essentielles pour garantir l'exactitude, l'exhaustivité, la cohérence et la fiabilité du traitement des flux de données.

Avant d'explorer ces exemples, nous vous recommandons de consulter d'abord l'article Utilisation d'un bloc-notes Studio avec un service géré pour Apache Flink.

Rubriques

Recréation des requêtes Kinesis Data Analytics pour SQL dans un service géré pour Apache Flink Studio

Le tableau suivant fournit les conversions des requêtes courantes de l'application Kinesis Data Analytics basée sur SQL vers le service géré pour Apache Flink Studio.

Application à plusieurs étapes

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "IN_APP_STREAM_001" (
   ingest_time TIMESTAMP,
   ticker_symbol VARCHAR(4),
   sector VARCHAR(16), price REAL, change REAL);
CREATE
OR REPLACE PUMP "STREAM_PUMP_001" AS
INSERT INTO
   "IN_APP_STREAM_001"
   SELECT
      STREAM APPROXIMATE_ARRIVAL_TIME,
      ticker_symbol,
      sector,
      price,
      change FROM "SOURCE_SQL_STREAM_001";
-- Second in-app stream and pump
CREATE
OR REPLACE STREAM "IN_APP_STREAM_02" (ingest_time TIMESTAMP,
   ticker_symbol VARCHAR(4),
   sector VARCHAR(16),
   price REAL,
   change REAL);
CREATE
OR REPLACE PUMP "STREAM_PUMP_02" AS
INSERT INTO
   "IN_APP_STREAM_02"
   SELECT
```

```
      STREAM ingest_time,
      ticker_symbol,
      sector,
      price,
      change FROM "IN_APP_STREAM_001";
-- Destination in-app stream and third pump
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ingest_time TIMESTAMP,
   ticker_symbol VARCHAR(4),
   sector VARCHAR(16),
   price REAL,
   change REAL);
CREATE
OR REPLACE PUMP "STREAM_PUMP_03" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM ingest_time,
      ticker_symbol,
      sector,
      price,
      change FROM "IN_APP_STREAM_02";
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001;

CREATE TABLE SOURCE_SQL_STREAM_001 (TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(16),
   PRICE DOUBLE,
   CHANGE DOUBLE,
   APPROXIMATE_ARRIVAL_TIME TIMESTAMP(3) METADATA

FROM
   'timestamp' VIRTUAL,
   WATERMARK FOR APPROXIMATE_ARRIVAL_TIME AS APPROXIMATE_ARRIVAL_TIME - INTERVAL '1'
 SECOND )
   PARTITIONED BY (TICKER_SYMBOL) WITH (
      'connector' = 'kinesis',
      'stream' = 'kinesis-analytics-demo-stream',
      'aws.region' = 'us-east-1',
      'scan.stream.initpos' = 'LATEST',
      'format' = 'json',
```

```
      'json.timestamp-format.standard' = 'ISO-8601');
DROP TABLE IF EXISTS IN_APP_STREAM_001;

CREATE TABLE IN_APP_STREAM_001 (
   INGEST_TIME TIMESTAMP,
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(16),
   PRICE DOUBLE,
   CHANGE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
      'connector' = 'kinesis',
      'stream' = 'IN_APP_STREAM_001',
      'aws.region' = 'us-east-1',
      'scan.stream.initpos' = 'LATEST',
      'format' = 'json',
      'json.timestamp-format.standard' = 'ISO-8601');

DROP TABLE IF EXISTS IN_APP_STREAM_02;

CREATE TABLE IN_APP_STREAM_02 (
   INGEST_TIME TIMESTAMP,
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(16),
   PRICE DOUBLE,
   CHANGE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'IN_APP_STREAM_02',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');

DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;

CREATE TABLE DESTINATION_SQL_STREAM (
   INGEST_TIME TIMESTAMP, TICKER_SYMBOL VARCHAR(4), SECTOR VARCHAR(16),
   PRICE DOUBLE, CHANGE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'DESTINATION_SQL_STREAM',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
```

```
   'json.timestamp-format.standard' = 'ISO-8601');


Query 2 - % flink.ssql(type =
update
)
   INSERT INTO
      IN_APP_STREAM_001
      SELECT
         APPROXIMATE_ARRIVAL_TIME AS INGEST_TIME,
         TICKER_SYMBOL,
         SECTOR,
         PRICE,
         CHANGE
      FROM
         SOURCE_SQL_STREAM_001;


Query 3 - % flink.ssql(type =
update
)
   INSERT INTO
      IN_APP_STREAM_02
      SELECT
         INGEST_TIME,
         TICKER_SYMBOL,
         SECTOR,
         PRICE,
         CHANGE
      FROM
         IN_APP_STREAM_001;


Query 4 - % flink.ssql(type =
update
)
   INSERT INTO
      DESTINATION_SQL_STREAM
      SELECT
         INGEST_TIME,
         TICKER_SYMBOL,
         SECTOR,
         PRICE,
         CHANGE
```

```
      FROM
         IN_APP_STREAM_02;
```

Transformation de valeurs DateTime

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   TICKER VARCHAR(4),
   event_time TIMESTAMP,
   five_minutes_before TIMESTAMP,
   event_unix_timestamp BIGINT,
   event_timestamp_as_char VARCHAR(50),
   event_second INTEGER);

CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM TICKER,
      EVENT_TIME,
      EVENT_TIME - INTERVAL '5' MINUTE,
      UNIX_TIMESTAMP(EVENT_TIME),
      TIMESTAMP_TO_CHAR('yyyy-MM-dd hh:mm:ss', EVENT_TIME),
      EXTRACT(SECOND
   FROM
      EVENT_TIME)
   FROM
      "SOURCE_SQL_STREAM_001"
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER VARCHAR(4),
   EVENT_TIME TIMESTAMP(3),
   FIVE_MINUTES_BEFORE TIMESTAMP(3),
   EVENT_UNIX_TIMESTAMP INT,
   EVENT_TIMESTAMP_AS_CHAR VARCHAR(50),
```

```
   EVENT_SECOND INT)

PARTITIONED BY (TICKER) WITH (
   'connector' = 'kinesis', 'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601')

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         TICKER,
         EVENT_TIME,
         EVENT_TIME - INTERVAL '5' MINUTE AS FIVE_MINUTES_BEFORE,
         UNIX_TIMESTAMP() AS EVENT_UNIX_TIMESTAMP,
         DATE_FORMAT(EVENT_TIME, 'yyyy-MM-dd hh:mm:ss') AS EVENT_TIMESTAMP_AS_CHAR,
         EXTRACT(SECOND
      FROM
         EVENT_TIME) AS EVENT_SECOND
      FROM
         DESTINATION_SQL_STREAM;
```

Alertes simples

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM"(
   ticker_symbol VARCHAR(4),
   sector VARCHAR(12),
   change DOUBLE,
   price DOUBLE);

CREATE
OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT
   STREAM ticker_symbol,
   sector,
   change,
   price
```

```
FROM
   "SOURCE_SQL_STREAM_001"
WHERE
   (
      ABS(Change / (Price - Change)) * 100
   )
   > 1
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;

CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(4),
   CHANGE DOUBLE,
   PRICE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');

Query 2 - % flink.ssql(type =
update
)
   SELECT
      TICKER_SYMBOL,
      SECTOR,
      CHANGE,
      PRICE
   FROM
      DESTINATION_SQL_STREAM
   WHERE
      (
         ABS(CHANGE / (PRICE - CHANGE)) * 100
      )
      > 1;
```

Alertes limitées

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "CHANGE_STREAM"(
   ticker_symbol VARCHAR(4),
   sector VARCHAR(12),
   change DOUBLE,
   price DOUBLE);

CREATE
OR REPLACE PUMP "change_pump" AS INSERT INTO "CHANGE_STREAM"
SELECT
   STREAM ticker_symbol,
   sector,
   change,
   price
FROM "SOURCE_SQL_STREAM_001"
WHERE
   (
      ABS(Change / (Price - Change)) * 100
   )
   > 1;
-- ** Trigger Count and Limit **
-- Counts "triggers" or those values that evaluated true against the previous where
 clause
-- Then provides its own limit on the number of triggers per hour per ticker symbol
 to what is specified in the WHERE clause

CREATE
OR REPLACE STREAM TRIGGER_COUNT_STREAM (
   ticker_symbol VARCHAR(4),
   change REAL,
   trigger_count INTEGER);

CREATE
OR REPLACE PUMP trigger_count_pump AS
INSERT INTO
   TRIGGER_COUNT_STREAMSELECT STREAM ticker_symbol,
   change,
   trigger_count
FROM
   (
```

```
      SELECT
         STREAM ticker_symbol,
         change,
         COUNT(*) OVER W1 as trigger_countFROM "CHANGE_STREAM" --window to perform
 aggregations over last minute to keep track of triggers
         WINDOW W1 AS
         (
            PARTITION BY ticker_symbol RANGE INTERVAL '1' MINUTE PRECEDING
         )
   )
WHERE
   trigger_count >= 1;
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;

CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(4),
   CHANGE DOUBLE, PRICE DOUBLE,
   EVENT_TIME AS PROCTIME())
PARTITIONED BY (TICKER_SYMBOL)
WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');
DROP TABLE IF EXISTS TRIGGER_COUNT_STREAM;
CREATE TABLE TRIGGER_COUNT_STREAM (
   TICKER_SYMBOL VARCHAR(4),
   CHANGE DOUBLE,
   TRIGGER_COUNT INT)
PARTITIONED BY (TICKER_SYMBOL);

Query 2 - % flink.ssql(type =
update
)
   SELECT
```

```
      TICKER_SYMBOL,
      SECTOR,
      CHANGE,
      PRICE
   FROM
      DESTINATION_SQL_STREAM
   WHERE
      (
         ABS(CHANGE / (PRICE - CHANGE)) * 100
      )
      > 1;

Query 3 - % flink.ssql(type =
update
)
   SELECT *
   FROM(
         SELECT
            TICKER_SYMBOL,
            CHANGE,
            COUNT(*) AS TRIGGER_COUNT
         FROM
            DESTINATION_SQL_STREAM
         GROUP BY
            TUMBLE(EVENT_TIME, INTERVAL '1' MINUTE),
            TICKER_SYMBOL,
            CHANGE
      )
   WHERE
      TRIGGER_COUNT > 1;
```

Regroupement d’une partie des résultats à partir d’une requête

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "CALC_COUNT_SQL_STREAM"(
   TICKER VARCHAR(4),
   TRADETIME TIMESTAMP,
   TICKERCOUNT DOUBLE);

CREATE
```

```
OR REPLACE STREAM "DESTINATION_SQL_STREAM"(
   TICKER VARCHAR(4),
   TRADETIME TIMESTAMP,
   TICKERCOUNT DOUBLE);

CREATE PUMP "CALC_COUNT_SQL_PUMP_001" AS
INSERT INTO
   "CALC_COUNT_SQL_STREAM"(
   "TICKER",
   "TRADETIME",
   "TICKERCOUNT")
   SELECT
      STREAM "TICKER_SYMBOL",
      STEP("SOURCE_SQL_STREAM_001",
      "ROWTIME" BY INTERVAL '1' MINUTE) as "TradeTime",
      COUNT(*) AS "TickerCount "
   FROM
      "SOURCE_SQL_STREAM_001"
   GROUP BY
      STEP("SOURCE_SQL_STREAM_001". ROWTIME BY INTERVAL '1' MINUTE),
      STEP("SOURCE_SQL_STREAM_001"." APPROXIMATE_ARRIVAL_TIME" BY INTERVAL '1'
 MINUTE),
      TICKER_SYMBOL;
CREATE PUMP "AGGREGATED_SQL_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM" (
   "TICKER",
   "TRADETIME",
   "TICKERCOUNT")
   SELECT
      STREAM "TICKER",
      "TRADETIME",
      SUM("TICKERCOUNT") OVER W1 AS "TICKERCOUNT"
   FROM
      "CALC_COUNT_SQL_STREAM" WINDOW W1 AS
      (
         PARTITION BY "TRADETIME" RANGE INTERVAL '10' MINUTE PRECEDING
      )
;
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
```

```
update
) DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001;
CREATE TABLE SOURCE_SQL_STREAM_001 (
   TICKER_SYMBOL VARCHAR(4),
   TRADETIME AS PROCTIME(),
   APPROXIMATE_ARRIVAL_TIME TIMESTAMP(3) METADATA
FROM
   'timestamp' VIRTUAL,
   WATERMARK FOR APPROXIMATE_ARRIVAL_TIME AS APPROXIMATE_ARRIVAL_TIME - INTERVAL '1'
 SECOND)
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');
DROP TABLE IF EXISTS CALC_COUNT_SQL_STREAM;
CREATE TABLE CALC_COUNT_SQL_STREAM (
   TICKER VARCHAR(4),
   TRADETIME TIMESTAMP(3),
   WATERMARK FOR TRADETIME AS TRADETIME - INTERVAL '1' SECOND,
   TICKERCOUNT BIGINT NOT NULL ) PARTITIONED BY (TICKER) WITH (
      'connector' = 'kinesis',
      'stream' = 'CALC_COUNT_SQL_STREAM',
      'aws.region' = 'us-east-1',
      'scan.stream.initpos' = 'LATEST',
      'format' = 'csv');
DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;
CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER VARCHAR(4),
   TRADETIME TIMESTAMP(3),
   WATERMARK FOR TRADETIME AS TRADETIME - INTERVAL '1' SECOND,
   TICKERCOUNT BIGINT NOT NULL )
   PARTITIONED BY (TICKER) WITH ('connector' = 'kinesis',
      'stream' = 'DESTINATION_SQL_STREAM',
      'aws.region' = 'us-east-1',
      'scan.stream.initpos' = 'LATEST',
      'format' = 'csv');

Query 2 - % flink.ssql(type =
update
)
   INSERT INTO
```

```
      CALC_COUNT_SQL_STREAM
      SELECT
         TICKER,
         TO_TIMESTAMP(TRADETIME, 'yyyy-MM-dd HH:mm:ss') AS TRADETIME,
         TICKERCOUNT
      FROM
         (
            SELECT
               TICKER_SYMBOL AS TICKER,
               DATE_FORMAT(TRADETIME, 'yyyy-MM-dd HH:mm:00') AS TRADETIME,
               COUNT(*) AS TICKERCOUNT
            FROM
               SOURCE_SQL_STREAM_001
            GROUP BY
               TUMBLE(TRADETIME, INTERVAL '1' MINUTE),
               DATE_FORMAT(TRADETIME, 'yyyy-MM-dd HH:mm:00'),
               DATE_FORMAT(APPROXIMATE_ARRIVAL_TIME, 'yyyy-MM-dd HH:mm:00'),
               TICKER_SYMBOL
         )
;

Query 3 - % flink.ssql(type =
update
)
   SELECT
      *
   FROM
      CALC_COUNT_SQL_STREAM;

Query 4 - % flink.ssql(type =
update
)
   INSERT INTO
      DESTINATION_SQL_STREAM
      SELECT
         TICKER,
         TRADETIME,
         SUM(TICKERCOUNT) OVER W1 AS TICKERCOUNT
      FROM
         CALC_COUNT_SQL_STREAM WINDOW W1 AS
         (
            PARTITION BY TICKER
         ORDER BY
            TRADETIME RANGE INTERVAL '10' MINUTE PRECEDING
```

```
         )
;

Query 5 - % flink.ssql(type =
update
)
   SELECT
      *
   FROM
      DESTINATION_SQL_STREAM;
```

Transformation de valeurs de chaîne

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM for cleaned up referrerCREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" ( "ingest_time" TIMESTAMP, "referrer"
 VARCHAR(32));
CREATE
OR REPLACE PUMP "myPUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT
   STREAM "APPROXIMATE_ARRIVAL_TIME",
   SUBSTRING("referrer", 12,
   (
      POSITION('.com' IN "referrer") - POSITION('www.' IN "referrer") - 4
   )
)
FROM
   "SOURCE_SQL_STREAM_001";
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   referrer VARCHAR(32),
   ingest_time AS PROCTIME() ) PARTITIONED BY (referrer)
WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
```

```
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601')

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         ingest_time,
         substring(referrer, 12, 6) as referrer
      FROM
         DESTINATION_SQL_STREAM;
```

Remplacement d'une sous-chaîne à l'aide de Regex

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM for cleaned up referrerCREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" ( "ingest_time" TIMESTAMP, "referrer"
 VARCHAR(32));
CREATE
OR REPLACE PUMP "myPUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT
   STREAM "APPROXIMATE_ARRIVAL_TIME",
   REGEX_REPLACE("REFERRER", 'http://', 'https://', 1, 0)
FROM
   "SOURCE_SQL_STREAM_001";
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   referrer VARCHAR(32),
   ingest_time AS PROCTIME())
PARTITIONED BY (referrer) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
```

```
   'json.timestamp-format.standard' = 'ISO-8601')

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         ingest_time,
         REGEXP_REPLACE(referrer, 'http', 'https') as referrer
      FROM
         DESTINATION_SQL_STREAM;
```

Analyse du journal Regex

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM"(
   sector VARCHAR(24),
   match1 VARCHAR(24),
   match2 VARCHAR(24));
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM T.SECTOR,
      T.REC.COLUMN1,
      T.REC.COLUMN2
   FROM
      (
         SELECT
            STREAM SECTOR,
            REGEX_LOG_PARSE(SECTOR, '.*([E].).*([R].*)') AS REC
         FROM
            SOURCE_SQL_STREAM_001
      )
      AS T;
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
```

```
) CREATE TABLE DESTINATION_SQL_STREAM (
   CHANGE DOUBLE, PRICE DOUBLE,
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(16))
PARTITIONED BY (SECTOR) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601')

Query 2 - % flink.ssql(type =
   update
)
SELECT
   *
FROM
   (
      SELECT
         SECTOR,
         REGEXP_EXTRACT(SECTOR, '.([E].).([R].)', 1) AS MATCH1,
         REGEXP_EXTRACT(SECTOR, '.([E].).([R].)', 2) AS MATCH2
      FROM
         DESTINATION_SQL_STREAM
   )
WHERE
   MATCH1 IS NOT NULL
   AND MATCH2 IS NOT NULL;
```

Transformation de valeurs DateTime

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   TICKER VARCHAR(4),
   event_time TIMESTAMP,
   five_minutes_before TIMESTAMP,
   event_unix_timestamp BIGINT,
   event_timestamp_as_char VARCHAR(50),
   event_second INTEGER);
```

```
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM TICKER,
      EVENT_TIME,
      EVENT_TIME - INTERVAL '5' MINUTE,
      UNIX_TIMESTAMP(EVENT_TIME),
      TIMESTAMP_TO_CHAR('yyyy-MM-dd hh:mm:ss', EVENT_TIME),
      EXTRACT(SECOND
   FROM
      EVENT_TIME)
   FROM
      "SOURCE_SQL_STREAM_001"
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER VARCHAR(4),
   EVENT_TIME TIMESTAMP(3),
   FIVE_MINUTES_BEFORE TIMESTAMP(3),
   EVENT_UNIX_TIMESTAMP INT,
   EVENT_TIMESTAMP_AS_CHAR VARCHAR(50),
   EVENT_SECOND INT) PARTITIONED BY (TICKER)
WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601')

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         TICKER,
         EVENT_TIME,
         EVENT_TIME - INTERVAL '5' MINUTE AS FIVE_MINUTES_BEFORE,
         UNIX_TIMESTAMP() AS EVENT_UNIX_TIMESTAMP,
```

```
         DATE_FORMAT(EVENT_TIME, 'yyyy-MM-dd hh:mm:ss') AS EVENT_TIMESTAMP_AS_CHAR,
         EXTRACT(SECOND
      FROM
         EVENT_TIME) AS EVENT_SECOND
      FROM
         DESTINATION_SQL_STREAM;
```

Fenêtres et regroupement

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   event_time TIMESTAMP,
   ticker_symbol VARCHAR(4),
   ticker_count INTEGER);
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM EVENT_TIME,
      TICKER,
      COUNT(TICKER) AS ticker_count
   FROM
      "SOURCE_SQL_STREAM_001" WINDOWED BY STAGGER ( PARTITION BY
         TICKER,
         EVENT_TIME RANGE INTERVAL '1' MINUTE);
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   EVENT_TIME TIMESTAMP(3),
   WATERMARK FOR EVENT_TIME AS EVENT_TIME - INTERVAL '60' SECOND,
   TICKER VARCHAR(4),
   TICKER_COUNT INT) PARTITIONED BY (TICKER)
WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
```

```
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json'

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         EVENT_TIME,
         TICKER, COUNT(TICKER) AS ticker_count
      FROM
         DESTINATION_SQL_STREAM
      GROUP BY
         TUMBLE(EVENT_TIME,
         INTERVAL '60' second),
         EVENT_TIME, TICKER;
```

Fenêtre bascule utilisant Rowtime

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM"(
   TICKER VARCHAR(4),
   MIN_PRICE REAL,
   MAX_PRICE REAL);
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM TICKER,
      MIN(PRICE),
      MAX(PRICE)
   FROM
      "SOURCE_SQL_STREAM_001"
   GROUP BY
      TICKER,
      STEP("SOURCE_SQL_STREAM_001".
            ROWTIME BY INTERVAL '60' SECOND);
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   ticker VARCHAR(4),
   price DOUBLE,
   event_time VARCHAR(32),
   processing_time AS PROCTIME())
PARTITIONED BY (ticker) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601')

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         ticker,
         min(price) AS MIN_PRICE,
         max(price) AS MAX_PRICE
      FROM
         DESTINATION_SQL_STREAM
      GROUP BY
         TUMBLE(processing_time, INTERVAL '60' second),
         ticker;
```

Extraction des valeurs les plus fréquentes (TOP_K_ITEMS_TUMBLING)

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "CALC_COUNT_SQL_STREAM"(TICKER VARCHAR(4),
   TRADETIME TIMESTAMP,
   TICKERCOUNT DOUBLE);
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM"(
   TICKER VARCHAR(4),
   TRADETIME TIMESTAMP,
```

```
   TICKERCOUNT DOUBLE);
CREATE PUMP "CALC_COUNT_SQL_PUMP_001" AS INSERT INTO "CALC_COUNT_SQL_STREAM" (
   "TICKER",
   "TRADETIME",
   "TICKERCOUNT")
SELECT
   STREAM"TICKER_SYMBOL",
   STEP("SOURCE_SQL_STREAM_001"."ROWTIME" BY INTERVAL '1' MINUTE) as "TradeTime",
   COUNT(*) AS "TickerCount"
FROM
   "SOURCE_SQL_STREAM_001"
GROUP BY STEP("SOURCE_SQL_STREAM_001".
   ROWTIME BY INTERVAL '1' MINUTE),
   STEP("SOURCE_SQL_STREAM_001".
      "APPROXIMATE_ARRIVAL_TIME" BY INTERVAL '1' MINUTE),
   TICKER_SYMBOL;
CREATE PUMP "AGGREGATED_SQL_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM" (
   "TICKER",
   "TRADETIME",
   "TICKERCOUNT")
SELECT
   STREAM "TICKER",
   "TRADETIME",
   SUM("TICKERCOUNT") OVER W1 AS "TICKERCOUNT"
FROM
   "CALC_COUNT_SQL_STREAM" WINDOW W1 AS
   (
      PARTITION BY "TRADETIME" RANGE INTERVAL '10' MINUTE PRECEDING
   )
;
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;
CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER VARCHAR(4),
   EVENT_TIME TIMESTAMP(3),
   WATERMARK FOR EVENT_TIME AS EVENT_TIME - INTERVAL '1' SECONDS )
PARTITIONED BY (TICKER) WITH (
   'connector' = 'kinesis', 'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
```

```
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');

Query 2 - % flink.ssql(type =
update
)
   SELECT
      *
   FROM
      (
         SELECT
            TICKER,
            COUNT(*) as MOST_FREQUENT_VALUES,
            ROW_NUMBER() OVER (PARTITION BY TICKER
         ORDER BY
            TICKER DESC) AS row_num
         FROM
            DESTINATION_SQL_STREAM
         GROUP BY
            TUMBLE(EVENT_TIME, INTERVAL '1' MINUTE),
            TICKER
      )
   WHERE
      row_num <= 5;
```

Éléments Top-K approximatifs

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ITEM VARCHAR(1024), ITEM_COUNT DOUBLE);
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM ITEM,
      ITEM_COUNT
   FROM
      TABLE(TOP_K_ITEMS_TUMBLING(CURSOR(
      SELECT
```

```
         STREAM *
      FROM
         "SOURCE_SQL_STREAM_001"), 'column1', -- name of column in single quotes10,
 -- number of top items60 -- tumbling window size in seconds));
```

Managed Service for Apache Flink Studio

```
%flinkssql
DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001
CREATE TABLE SOURCE_SQL_STREAM_001 ( TS TIMESTAMP(3), WATERMARK FOR TS as TS -
 INTERVAL '5' SECOND, ITEM VARCHAR(1024),
PRICE DOUBLE)
   WITH ( 'connector' = 'kinesis', 'stream' = 'SOURCE_SQL_STREAM_001',
'aws.region' = 'us-east-1', 'scan.stream.initpos' = 'LATEST', 'format' = 'json',
'json.timestamp-format.standard' = 'ISO-8601');


%flink.ssql(type=update)
SELECT
   *
FROM
   (
      SELECT
         *,
         ROW_NUMBER() OVER (PARTITION BY AGG_WINDOW
      ORDER BY
         ITEM_COUNT DESC) as rownum
      FROM
         (
            select
               AGG_WINDOW,
               ITEM,
               ITEM_COUNT
            from
               (
                  select
                     TUMBLE_ROWTIME(TS, INTERVAL '60' SECONDS) as AGG_WINDOW,
                     ITEM,
                     count(*) as ITEM_COUNT
                  FROM
                     SOURCE_SQL_STREAM_001
                  GROUP BY
```

```
                    TUMBLE(TS, INTERVAL '60' SECONDS),
                    ITEM
                )
        )
    )
where
    rownum <= 3
```

Analyse de journaux web (fonction W3C_LOG_PARSE)

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" ( column1 VARCHAR(16),
    column2 VARCHAR(16),
    column3 VARCHAR(16),
    column4 VARCHAR(16),
    column5 VARCHAR(16),
    column6 VARCHAR(16),
    column7 VARCHAR(16));
CREATE
OR REPLACE PUMP "myPUMP" ASINSERT INTO "DESTINATION_SQL_STREAM"
SELECT
    STREAM l.r.COLUMN1,
    l.r.COLUMN2,
    l.r.COLUMN3,
    l.r.COLUMN4,
    l.r.COLUMN5,
    l.r.COLUMN6,
    l.r.COLUMN7
FROM
    (
       SELECT
          STREAM W3C_LOG_PARSE("log", 'COMMON')
       FROM
          "SOURCE_SQL_STREAM_001"
    )
    AS l(r);
```

Managed Service for Apache Flink Studio

```
%flink.ssql(type=update)
```

```
DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001 CREATE TABLE SOURCE_SQL_STREAM_001 ( log
 VARCHAR(1024))
   WITH ( 'connector' = 'kinesis',
          'stream' = 'SOURCE_SQL_STREAM_001',
          'aws.region' = 'us-east-1',
          'scan.stream.initpos' = 'LATEST',
          'format' = 'json',
          'json.timestamp-format.standard' = 'ISO-8601');

% flink.ssql(type=update)
   select
      SPLIT_INDEX(log, ' ', 0),
      SPLIT_INDEX(log, ' ', 1),
      SPLIT_INDEX(log, ' ', 2),
      SPLIT_INDEX(log, ' ', 3),
      SPLIT_INDEX(log, ' ', 4),
      SPLIT_INDEX(log, ' ', 5),
      SPLIT_INDEX(log, ' ', 6)
   from
      SOURCE_SQL_STREAM_001;
```

Fractionnement de chaînes en plusieurs champs (fonction VARIABLE_COLUMN_LOG_PARSE)

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM"( "column_A" VARCHAR(16),
   "column_B" VARCHAR(16),
   "column_C" VARCHAR(16),
   "COL_1" VARCHAR(16),
   "COL_2" VARCHAR(16),
   "COL_3" VARCHAR(16));
CREATE
OR REPLACE PUMP "SECOND_STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT
   STREAM t."Col_A",
   t."Col_B",
   t."Col_C",
   t.r."COL_1",
   t.r."COL_2",
   t.r."COL_3"
FROM
```

```
   (
      SELECT
         STREAM "Col_A",
         "Col_B",
         "Col_C",
         VARIABLE_COLUMN_LOG_PARSE ("Col_E_Unstructured",
         'COL_1 TYPE VARCHAR(16),
         COL_2 TYPE VARCHAR(16),
         COL_3 TYPE VARCHAR(16)', ',') AS r
      FROM
         "SOURCE_SQL_STREAM_001"
   )
   as t;
```

Managed Service for Apache Flink Studio

```
%flink.ssql(type=update)
DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001 CREATE TABLE SOURCE_SQL_STREAM_001 ( log
 VARCHAR(1024))
   WITH ( 'connector' = 'kinesis',
          'stream' = 'SOURCE_SQL_STREAM_001',
          'aws.region' = 'us-east-1',
          'scan.stream.initpos' = 'LATEST',
          'format' = 'json',
          'json.timestamp-format.standard' = 'ISO-8601');

% flink.ssql(type=update)
   select
      SPLIT_INDEX(log, ' ', 0),
      SPLIT_INDEX(log, ' ', 1),
      SPLIT_INDEX(log, ' ', 2),
      SPLIT_INDEX(log, ' ', 3),
      SPLIT_INDEX(log, ' ', 4),
      SPLIT_INDEX(log, ' ', 5)
)
from
   SOURCE_SQL_STREAM_001;
```

Jointures

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   ticker_symbol VARCHAR(4),
   "Company" varchar(20),
   sector VARCHAR(12),
   change DOUBLE,
   price DOUBLE);
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM ticker_symbol,
      "c"."Company",
      sector,
      change,
      priceFROM "SOURCE_SQL_STREAM_001"
      LEFT JOIN
         "CompanyName" as "c"
         ON "SOURCE_SQL_STREAM_001".ticker_symbol = "c"."Ticker";
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(12),
   CHANGE INT,
   PRICE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');

Query 2 - CREATE TABLE CompanyName (
```

```
   Ticker VARCHAR(4),
   Company VARCHAR(4)) WITH (
      'connector' = 'filesystem',
      'path' = 's3://kda-demo-sample/TickerReference.csv',
      'format' = 'csv' );

Query 3 - % flink.ssql(type =
update
)
   SELECT
      TICKER_SYMBOL,
      c.Company,
      SECTOR,
      CHANGE,
      PRICE
   FROM
      DESTINATION_SQL_STREAM
      LEFT JOIN
         CompanyName as c
         ON DESTINATION_SQL_STREAM.TICKER_SYMBOL = c.Ticker;
```

Erreurs

SQL-based Kinesis Data Analytics application

```
SELECT
   STREAM ticker_symbol,
   sector,
   change,
   (
      price / 0
   )
   as ProblemColumnFROM "SOURCE_SQL_STREAM_001"
WHERE
   sector SIMILAR TO '%TECH%';
```

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;
CREATE TABLE DESTINATION_SQL_STREAM (
```

```
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(16),
   CHANGE DOUBLE,
   PRICE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');

Query 2 - % flink.pyflink @udf(input_types = [DataTypes.BIGINT()],
   result_type = DataTypes.BIGINT()) def DivideByZero(price): try: price / 0
except
: return - 1 st_env.register_function("DivideByZero",
   DivideByZero)

   Query 3 - % flink.ssql(type =
update
)
   SELECT
      CURRENT_TIMESTAMP AS ERROR_TIME,
      *
   FROM
      (
         SELECT
            TICKER_SYMBOL,
            SECTOR,
            CHANGE,
            DivideByZero(PRICE) as ErrorColumn
         FROM
            DESTINATION_SQL_STREAM
         WHERE
            SECTOR SIMILAR TO '%TECH%'
      )
      AS ERROR_STREAM;
```

Migration des charges de travail Random Cut Forest

Si vous souhaitez déplacer des charges de travail utilisant Random Cut Forest de Kinesis Analytics pour SQL vers le service géré pour Apache Flink, ce billet de blog AWS explique comment utiliser le

service géré pour Apache Flink afin d'exécuter un algorithme RCF en ligne en vue de détecter des anomalies.

Remplacement de Kinesis Data Firehose en tant que source par Kinesis Data Streams

Consultez Converting-KDASQL-KDAStudio/ pour un didacticiel complet.

Dans l'exercice suivant, vous allez modifier votre flux de données afin d'utiliser le service géré Amazon pour Apache Flink Studio. Cela impliquera également de passer d'Amazon Kinesis Data Firehose à Amazon Kinesis Data Streams.

Nous partageons d'abord une architecture KDA basée sur SQL typique, avant de montrer comment la remplacer à l'aide du service géré Amazon pour Apache Flink Studio et Amazon Kinesis Data Streams. Vous pouvez également lancer le modèle AWS CloudFormation ici :

Amazon Kinesis Data Analytics basée sur SQL et Amazon Kinesis Data Firehose

Voici le flux architectural de l'application Amazon Kinesis Data Analytics basée sur SQL :

Nous examinons d'abord la configuration d'une application héritée Amazon Kinesis Data Analytics basée sur SQL et Amazon Kinesis Data Firehose. Le cas d'utilisation concerne un marché boursier sur lequel des données commerciales, notamment des données de symbole boursier et de cours des actions, sont transmises depuis des sources externes aux systèmes Amazon Kinesis. Amazon Kinesis Data Analytics pour SQL utilise le flux d'entrée pour exécuter des requêtes à fenêtres, telles que la fenêtre bascule, afin de déterminer le volume des transactions et le cours `min`, `max` et `average` des transactions sur une fenêtre d'une minute pour chaque symbole boursier.

Amazon Kinesis Data Analytics pour SQL est configuré pour ingérer les données de l'API Amazon Kinesis Data Firehose. Après le traitement, Amazon Kinesis Data Analytics pour SQL envoie les données traitées vers un autre Amazon Kinesis Data Firehose, qui enregistre ensuite le résultat dans un compartiment Amazon S3.

Dans ce cas, vous utilisez Amazon Kinesis Data Generator. Amazon Kinesis Data Generator vous permet d'envoyer des données de test à vos flux de diffusion Amazon Kinesis Data Streams ou Amazon Kinesis Data Firehose. Pour commencer, veuillez suivre les instructions ici. Utilisez le modèle AWS CloudFormation ici à la place de celui fourni dans les instructions :.

Une fois le modèle AWS CloudFormation exécuté, la section de sortie fournit l'URL d'Amazon Kinesis Data Generator. Connectez-vous au portail à l'aide de l'identifiant utilisateur et du mot de passe Cognito que vous avez définis ici. Sélectionnez la région et le nom du flux cible. Pour l'état actuel, choisissez les flux de diffusion Amazon Kinesis Data Firehose. Pour l'état actuel, choisissez le nom de flux Amazon Kinesis Data Firehose. Vous pouvez créer plusieurs modèles, en fonction de vos besoins, et tester le modèle à l'aide du bouton Tester le modèle avant de l'envoyer au flux cible.

Vous trouverez ci-dessous un exemple de charge utile utilisant Amazon Kinesis Data Generator. Le générateur de données cible les flux Amazon Kinesis Firehose en entrée pour diffuser les données en continu. Le client Amazon Kinesis SDK peut également envoyer des données provenant d'autres producteurs.

```
2023-02-17 09:28:07.763,"AAPL",5032023-02-17 09:28:07.763,
"AMZN",3352023-02-17 09:28:07.763,
"GOOGL",1852023-02-17 09:28:07.763,
"AAPL",11162023-02-17 09:28:07.763,
"GOOGL",1582
```

Le code JSON suivant est utilisé pour générer une série aléatoire de date et heure de transaction, de symbole boursier et de cours d'action :

```
date.now(YYYY-MM-DD HH:mm:ss.SSS),
"random.arrayElement(["AAPL","AMZN","MSFT","META","GOOGL"])",
random.number(2000)
```

Une fois que vous avez choisi Envoyer les données, le générateur commence à envoyer des données fictives.

Les systèmes externes transmettent les données à Amazon Kinesis Data Firehose. Grâce aux applications Amazon Kinesis Data Analytics pour SQL, vous pouvez analyser les données de streaming à l'aide du langage SQL standard. Ce service vous permet de créer et d'exécuter rapidement un code sur des sources de streaming afin d'effectuer des analyses de séries chronologiques, d'alimenter des tableaux de bord en temps réel et de générer des métriques en temps réel. Les applications Amazon Kinesis Data Analytics pour SQL peuvent créer un flux de

destination à partir de requêtes SQL sur le flux d'entrée et envoyer le flux de destination à un autre Amazon Kinesis Data Firehose. L'Amazon Kinesis Data Firehose de destination peut envoyer les données analytiques à Amazon S3 en dernière étape.

Le code hérité d'Amazon Kinesis Data Analytics pour SQL est basé sur une extension de la norme SQL.

Vous utilisez la requête suivante dans Amazon Kinesis Data Analytics pour SQL. Vous créez d'abord un flux de destination pour la sortie de requête. Ensuite, vous devez utiliser PUMP, un objet de référentiel Amazon Kinesis Data Analytics (une extension de la norme SQL) qui fournit une fonctionnalité de requête `INSERT INTO stream SELECT ... FROM` continue, pour saisir les résultats d'une requête en continu dans un flux nommé.

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (EVENT_TIME TIMESTAMP,
INGEST_TIME TIMESTAMP,
TICKER VARCHAR(16),
VOLUME BIGINT,
AVG_PRICE DOUBLE,
MIN_PRICE DOUBLE,
MAX_PRICE DOUBLE);

CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM STEP("SOURCE_SQL_STREAM_001"."tradeTimestamp" BY INTERVAL '60' SECOND) AS
 EVENT_TIME,
      STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND) AS
 "STREAM_INGEST_TIME",
      "ticker",
       COUNT(*) AS VOLUME,
      AVG("tradePrice") AS AVG_PRICE,
      MIN("tradePrice") AS MIN_PRICE,
      MAX("tradePrice") AS MAX_PRICEFROM "SOURCE_SQL_STREAM_001"
   GROUP BY
      "ticker",
      STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND),
      STEP("SOURCE_SQL_STREAM_001"."tradeTimestamp" BY INTERVAL '60' SECOND);
```

Le code SQL ci-dessus utilise deux fenêtres temporelles : `tradeTimestamp` qui provient de la charge utile du flux entrant et `ROWTIME.tradeTimestamp` également appelée `Event Time` ou `client-side time`. Il est souvent préférable d'utiliser cette heure dans les analyses car il s'agit du moment où un événement s'est produit. Cependant, de nombreuses sources d'événements, telles que les téléphones mobiles et les clients web, n'ont pas horloges fiables, ce qui peut entraîner des heures inexactes. En outre, des problèmes de connectivité peuvent provoquer le fait que des enregistrements figurant dans un flux ne sont pas dans le même ordre que celui où les événements se sont produits.

Les flux intégrés à l'application incluent également une colonne spéciale appelée `ROWTIME`. Celle-ci stocke un horodatage quand Amazon Kinesis Data Analytics insère une ligne dans le premier flux intégré à l'application. `ROWTIME` reflète l'horodatage du moment où Amazon Kinesis Data Analytics a inséré un enregistrement dans le premier flux intégré à l'application après la lecture de la source de streaming. Cette valeur `ROWTIME` est ensuite gérée tout au long de votre application.

Le code SQL détermine le nombre de symboles sous forme de `volume`, et de cours `min`, `max` et `average` sur un intervalle de 60 secondes.

L'utilisation de chacun de ces types d'heure dans des requêtes à fenêtres temporelles présente des avantages et des inconvénients. Sélectionnez un ou plusieurs de ces types d'heure, et une stratégie pour traiter les inconvénients pertinents en fonction du scénario d'utilisation.

Une stratégie à deux fenêtres utilise deux types d'heure, `ROWTIME` et l'un des autres types d'heure, comme l'heure de l'événement.

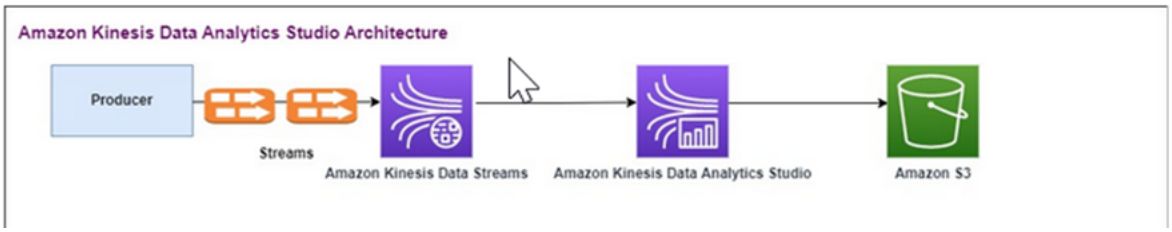
- Utilisez `ROWTIME`comme première fenêtre, qui contrôle la fréquence à laquelle la requête émet les résultats, comme illustré dans l'exemple suivant. Cette valeur n'est pas utilisée comme une heure logique.
- Utilisez l'un des autres types d'heure comme heure logique à associer à vos analyses. Cette heure représente le moment où l'événement s'est produit. Dans l'exemple suivant, l'objectif de l'analyse est de regrouper les enregistrements et de renvoyer un comptage par symbole boursier.

Service géré Amazon pour Apache Flink Studio

Dans l'architecture mise à jour, vous remplacez Amazon Kinesis Data Firehose par Amazon Kinesis Data Streams. Les applications Amazon Kinesis Data Analytics pour SQL sont remplacées par le service géré Amazon pour Apache Flink Studio. Le code Apache Flink est exécuté de manière interactive dans un bloc-notes Apache Zeppelin. Le service géré Amazon pour Apache Flink Studio

envoie les données commerciales agrégées vers un compartiment Amazon S3 en vue de leur stockage. Les étapes sont indiquées ci-dessous :

Voici le flux architectural du service géré Amazon pour Apache Flink Studio :



Créer un flux de données Kinesis

Pour créer un flux de données avec la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Dans la barre de navigation, développez le sélecteur de région et choisissez une région.
3. Choisissez Create data stream (Créer un flux de données).
4. Sur la page Créer un flux Kinesis, saisissez le nom de votre flux de données, puis acceptez le mode de capacité à la demande par défaut.

   En mode à la demande, vous pouvez ensuite choisir Créer un flux Kinesis pour créer votre flux de données.

   Dans la page Flux Kinesis, la valeur Statut de votre flux est En création lorsque le flux est en cours de création. Lorsque le flux est à prêt à être utilisé, le statut passe à Actif.
5. Choisissez le nom de votre flux. La page Détails du flux affiche un récapitulatif de la configuration de flux, ainsi que des informations de surveillance.
6. Dans Amazon Kinesis Data Generator, remplacez le flux ou le flux de diffusion par le nouvel Amazon Kinesis Data Streams : TRADE_SOURCE_STREAM.

   Le format JSON et la charge utile seront les mêmes que ceux que vous avez utilisés pour Amazon Kinesis Data Analytics pour SQL. Utilisez Amazon Kinesis Data Generator pour produire des exemples de données de charge utile commerciales et ciblez le flux de données TRADE_SOURCE_STREAM pour cet exercice :

```
{{date.now(YYYY-MM-DD HH:mm:ss.SSS)}},
"{{random.arrayElement(["AAPL","AMZN","MSFT","META","GOOGL"])}}",
```

```
{{random.number(2000)}}
```

7. Sur l'AWS Management Console, accédez au service géré pour Apache Flink et sélectionnez Créer une application.
8. Dans le panneau de navigation à gauche, sélectionnez Blocs-notes Studio, puis Créer un bloc-notes Studio.
9. Saisissez un nom pour le bloc-notes Studio.
10. Sous Base de données AWS Glue, indiquez une base de données AWS Glue existante qui définira les métadonnées de vos sources et destinations. Si vous n'avez pas de base de données AWS Glue, sélectionnez Créer et procédez comme suit :

    a. Dans la console AWS Glue, sélectionnez Bases de données sous Catalogue de données dans le menu de gauche.
    b. Sélectionnez Créer une base de données.
    c. Sur la page Créer une base de données, saisissez un nom pour la base de données. Dans la section Emplacement - facultatif, choisissez Parcourir Amazon S3 et sélectionnez le compartiment Amazon S3. Si vous n'avez pas de compartiment Amazon S3 déjà configuré, vous pouvez ignorer cette étape et y revenir plus tard.
    d. (Facultatif). Saisissez une description pour la base de données.
    e. Choisissez Create database (Créer une base de données).
11. Choisissez Créer un bloc-notes.
12. Une fois votre bloc-notes créé, choisissez Exécuter.
13. Une fois le bloc-notes démarré avec succès, lancez un bloc-notes Zeppelin en sélectionnant Ouvrir dans Apache Zeppelin.
14. Sur la page Bloc-notes Zeppelin, choisissez Créer une nouvelle note et nommez-la MarketDataFeed.

Le code SQL Flink est expliqué ci-dessous, mais voici d'abord à quoi ressemble un écran de bloc-notes Zeppelin. Chaque fenêtre du bloc-notes est un bloc de code distinct ; elles peuvent être exécutées une par une.

Code de service géré Amazon pour Apache Flink Studio

Le service géré Amazon pour Apache Flink Studio utilise les blocs-notes Zeppelin pour exécuter le code. Pour cet exemple, le mappage est effectué avec du code SSQL basé sur Apache Flink 1.13. Le code du bloc-notes Zeppelin est affiché en dessous un bloc à la fois.

Avant d'exécuter du code dans votre bloc-notes Zeppelin, les commandes de configuration Flink doivent être exécutées. Si vous devez modifier un paramètre de configuration après l'exécution du code (SSQL, Python ou Scala), vous devez arrêter et redémarrer votre bloc-notes. Dans cet exemple, vous devez définir un point de contrôle. Un point de contrôle est nécessaire pour diffuser des données vers un fichier dans Amazon S3. Cela permet de transférer vers un fichier la diffusion de données vers Amazon S3. L'instruction ci-dessous définit l'intervalle à 5 000 millisecondes.

```
%flink.conf
execution.checkpointing.interval 5000
```

`%flink.conf` indique que ce bloc est constitué d'instructions de configuration. Pour plus d'informations sur la configuration de Flink, y compris le point de contrôle, consultez Point de contrôle Apache Flink.

La table d'entrée pour la source Amazon Kinesis Data Streams est créée avec le code SSQL Flink ci-dessous. Notez que le champ `TRADE_TIME` enregistre la date et l'heure de création par le générateur de données.

```
%flink.ssql

DROP TABLE IF EXISTS TRADE_SOURCE_STREAM;
CREATE TABLE TRADE_SOURCE_STREAM (--`arrival_time` TIMESTAMP(3) METADATA FROM
 'timestamp' VIRTUAL,
TRADE_TIME TIMESTAMP(3),
WATERMARK FOR TRADE_TIME as TRADE_TIME - INTERVAL '5' SECOND,TICKER STRING,PRICE
 DOUBLE,
STATUS STRING)WITH ('connector' = 'kinesis','stream' = 'TRADE_SOURCE_STREAM',
'aws.region' = 'us-east-1','scan.stream.initpos' = 'LATEST','format' = 'csv');
```

Vous pouvez afficher le flux d'entrée avec cette instruction :

```
%flink.ssql(type=update)-- testing the source stream

select * from TRADE_SOURCE_STREAM;
```

Avant d'envoyer les données agrégées à Amazon S3, vous pouvez les afficher directement dans le service géré Amazon pour Apache Flink Studio à l'aide d'une requête de sélection à fenêtres bascules. Cela agrège les données commerciales dans des fenêtres temporelles d'une minute. Notez que l'instruction %flink.ssql doit avoir une désignation (type=update) :

```
%flink.ssql(type=update)

select TUMBLE_ROWTIME(TRADE_TIME,
INTERVAL '1' MINUTE) as TRADE_WINDOW,
TICKER, COUNT(*) as VOLUME,
AVG(PRICE) as AVG_PRICE,
MIN(PRICE) as MIN_PRICE,
MAX(PRICE) as MAX_PRICE FROM TRADE_SOURCE_STREAMGROUP BY TUMBLE(TRADE_TIME, INTERVAL
 '1' MINUTE), TICKER;
```

Vous pouvez ensuite créer une table de destination dans Amazon S3. Vous devez utiliser un filigrane. Un filigrane est une mesure de progression qui indique un instant donné à partir duquel vous êtes certain qu'aucun autre événement retardé ne se produira. L'objectif du filigrane est de prendre en compte les arrivées tardives. L'intervalle `'5' Second` permet aux transactions d'entrer dans le flux Amazon Kinesis Data Streams avec 5 secondes de retard et d'être incluses si elles sont horodatées dans cette fenêtre. Pour plus d'informations, consultez Générations de filigranes.

```
%flink.ssql(type=update)

DROP TABLE IF EXISTS TRADE_DESTINATION_S3;
CREATE TABLE TRADE_DESTINATION_S3 (
TRADE_WINDOW_START TIMESTAMP(3),
WATERMARK FOR TRADE_WINDOW_START as TRADE_WINDOW_START - INTERVAL '5' SECOND,
TICKER STRING,
VOLUME BIGINT,
AVG_PRICE DOUBLE,
MIN_PRICE DOUBLE,
MAX_PRICE DOUBLE)
WITH ('connector' = 'filesystem','path' = 's3://trade-destination/','format' = 'csv');
```

Cette instruction insère les données dans le code `TRADE_DESTINATION_S3`. `TUMPLE_ROWTIME` correspond à l'horodatage de la limite supérieure incluse de la fenêtre bascule.

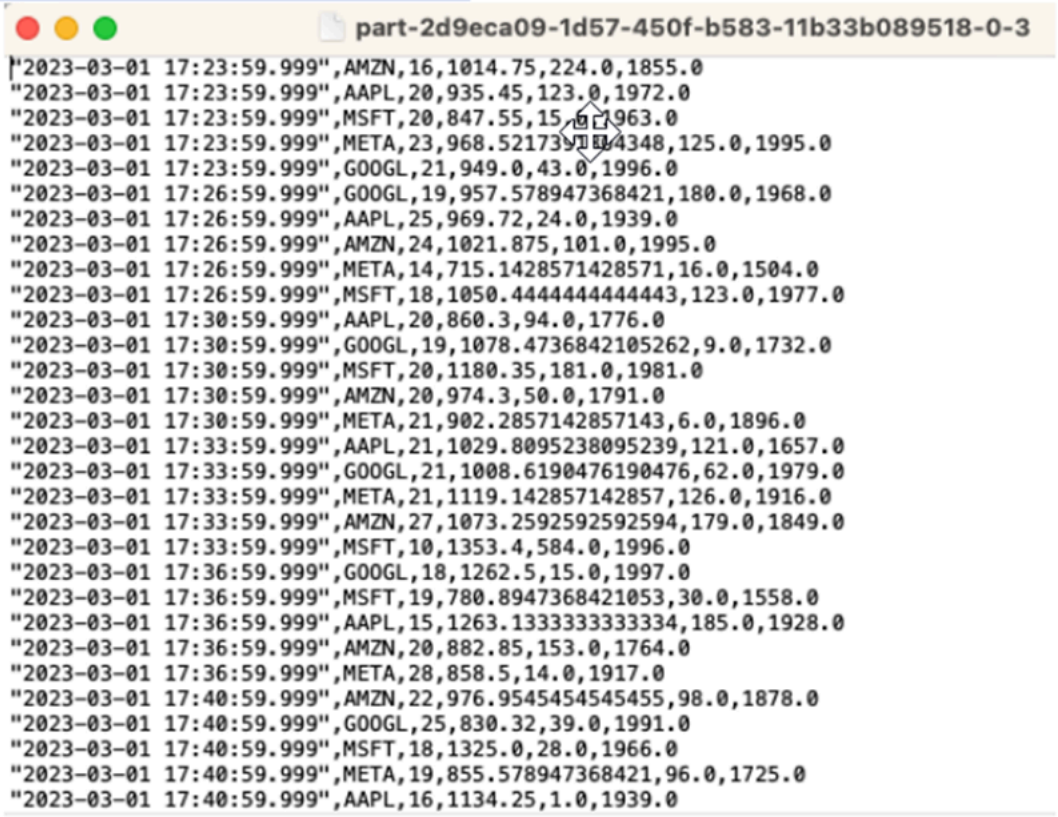
```
%flink.ssql(type=update)

insert into TRADE_DESTINATION_S3
select TUMBLE_ROWTIME(TRADE_TIME,
INTERVAL '1' MINUTE),
TICKER, COUNT(*) as VOLUME,
AVG(PRICE) as AVG_PRICE,
MIN(PRICE) as MIN_PRICE,
```

```
MAX(PRICE) as MAX_PRICE FROM TRADE_SOURCE_STREAM
GROUP BY TUMBLE(TRADE_TIME, INTERVAL '1' MINUTE), TICKER;
```

Exécutez votre instruction pendant 10 à 20 minutes afin d'accumuler des données dans Amazon S3. Puis annulez l'instruction.

Cela ferme le fichier dans Amazon S3 afin qu'il soit consultable.

Voici à quoi ressemble le contenu :



Vous pouvez utiliser le modèle AWS CloudFormation pour créer l'infrastructure.

AWS CloudFormation créera les ressources suivantes dans votre compte AWS :

- Amazon Kinesis Data Streams
- Service géré Amazon pour Apache Flink Studio
- Base de données Amazon Glue
- Compartiment Amazon S3
- Rôles et politiques IAM du service géré Amazon pour Apache Flink Studio pour accéder aux ressources appropriées

Importez le bloc-notes et remplacez le nom du compartiment Amazon S3 par le nouveau compartiment Amazon S3 créé par AWS CloudFormation.

```
%flink.ssql(type=update)

DROP TABLE IF EXISTS  TRADE_DESTINATION_S3;
CREATE TABLE TRADE_DESTINATION_S3 (
TRADE_WINDOW_START TIMESTAMP(3),
WATERMARK FOR TRADE_WINDOW_START as TRADE_WINDOW_START - INTERVAL '5' SECOND,
TICKER STRING,
VOLUME BIGINT,
AVG_PRICE DOUBLE,
MIN_PRICE DOUBLE,
MAX_PRICE DOUBLE)
WITH ('connector' = 'filesystem','path' = 's3://kda-studio-test-stack-markettradinganalyticsc[redacted]/','format' = 'csv');
```

Voir plus

Voici quelques ressources supplémentaires que vous pouvez utiliser pour en savoir plus sur l'utilisation du service géré pour Apache Flink Studio :

- Manuel du développeur relatifs aux blocs-notes du service géré pour Apache Flink Studio
- Documentation Apache Flink 1.13
- Atelier Service géré Amazon pour Apache Flink Studio
- Fenêtrage Apache Flink
- Guide du développeur Amazon Kinesis Data Analytics — Rédaction depuis un flux Kinesis Data Analytics vers un compartiment S3

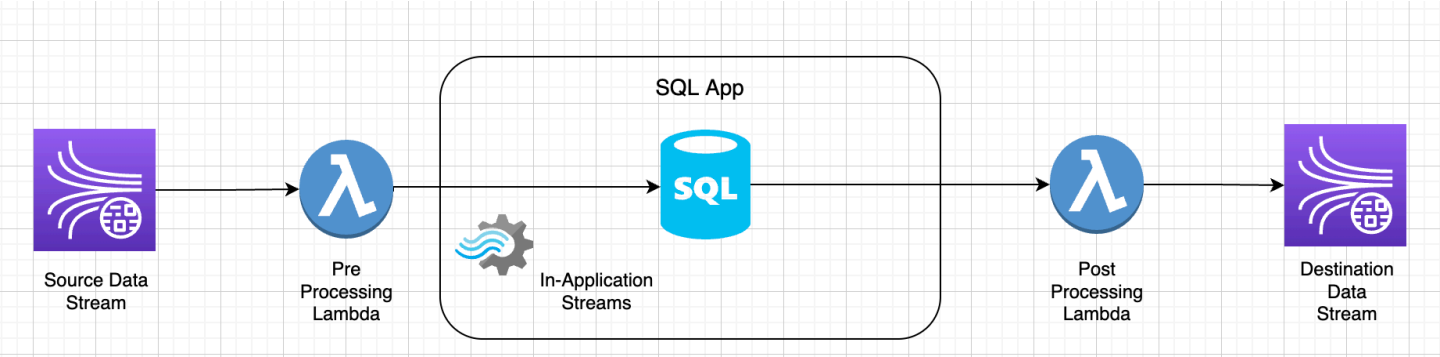
Utilisation des fonctions définies par l'utilisateur (UDF)

L'objectif de ce modèle est de montrer comment utiliser les UDF dans les blocs-notes Zeppelin de Kinesis Data Analytics-Studio pour traiter les données dans le flux Kinesis. Le service géré pour Apache Flink Srudio utilise Apache Flink pour fournir des capacités analytiques avancées, dont notamment la sémantique de traitement unique, les fenêtres temporelles d'événements, l'extensibilité grâce à des fonctions définies par l'utilisateur et à des intégrations clients, la prise en charge du langage impératif, l'état durable des applications, la mise à l'échelle horizontale, la prise en charge de plusieurs sources de données, les intégrations extensibles, etc. Ces fonctionnalités sont essentielles pour garantir l'exactitude, l'exhaustivité, la cohérence et la fiabilité du traitement des flux de données et ne sont pas disponibles avec Amazon Kinesis Data Analytics pour SQL.

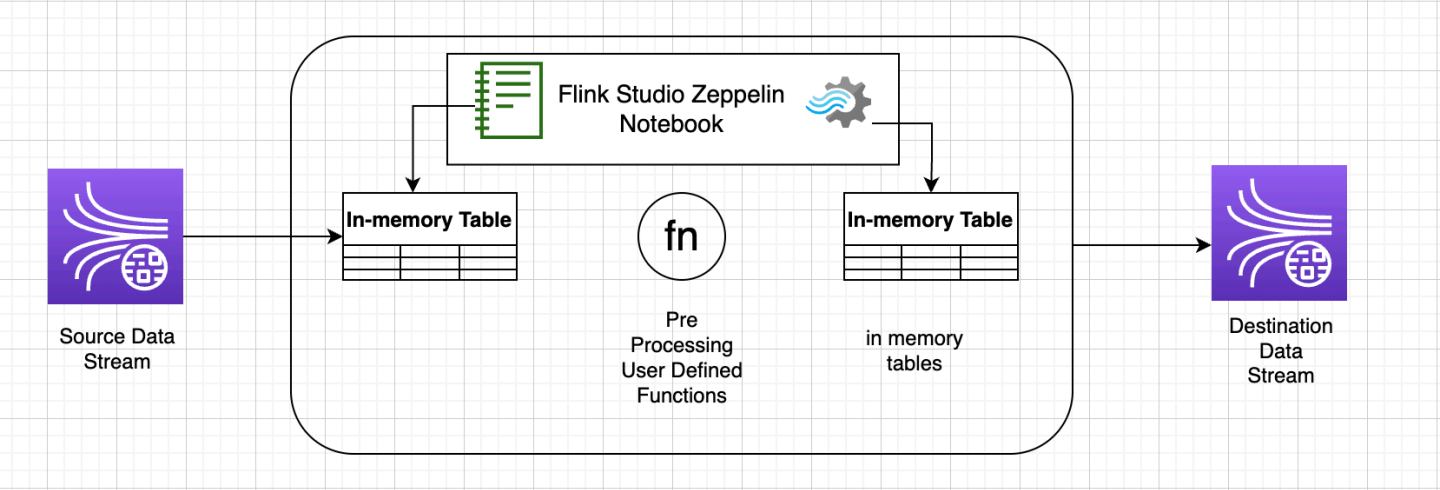
Dans cet exemple d'application, nous allons montrer comment utiliser les UDF dans le bloc-notes Zeppelin de KDA-Studio pour traiter les données dans le flux Kinesis. Les blocs-notes Studio pour Kinesis Data Analytics vous permettent d'interroger des flux de données de manière interactive en temps réel, et de créer et d'exécuter facilement des applications de traitement de flux à l'aide des

normes SQL, Python et Scala. En quelques clics dans l'AWS Management Console, vous pouvez lancer un bloc-notes sans serveur pour interroger des flux de données et obtenir des résultats en quelques secondes. Pour plus d'informations, consultez Utilisation d'un bloc-notes Studio avec Kinesis Data Analytics pour Apache Flink.

Fonctions Lambda utilisées pour le prétraitement et le post-traitement des données dans les applications KDA-SQL :



Fonctions définies par l'utilisateur pour le prétraitement et le post-traitement des données à l'aide des blocs-notes Zeppelin KDA-Studio



fonctions définies par l'utilisateur (UDF)

Pour réutiliser une logique métier courante dans un opérateur, il peut être utile de référencer une fonction définie par l'utilisateur afin de transformer votre flux de données. Vous pouvez le faire soit dans le bloc-notes du service géré pour Apache Flink Studio, soit sous forme de fichier jar d'application référencé en externe. L'utilisation de fonctions définies par l'utilisateur peut simplifier les transformations ou les enrichissements de données que vous pouvez effectuer sur des données de streaming.

Dans votre bloc-notes, vous ferez référence à un simple fichier jar d'application Java doté de fonctionnalités permettant d'anonymiser les numéros de téléphone personnels. Vous pouvez également écrire des UDF en Python ou Scala à utiliser dans le bloc-notes. Nous avons choisi un fichier jar d'application Java pour mettre en évidence la fonctionnalité d'importation d'un fichier jar d'application dans un bloc-notes Pyflink.

Configuration de l'environnement

Pour suivre ce guide et interagir avec vos données de streaming, vous allez utiliser un script AWS CloudFormation afin de lancer les ressources suivantes :

- Flux de données sources et cibles Kinesis
- Base de données Glue
- Rôle IAM
- Service géré pour l'application Apache Flink Studio
- Fonction Lambda pour démarrer le service géré pour l'application Apache Flink Studio
- Rôle Lambda pour exécuter la fonction Lambda ci-dessus
- Ressource personnalisée pour appeler la fonction Lambda

Téléchargez le modèle AWS CloudFormation ici.

Créez la pile AWS CloudFormation.

1. Accédez à la AWS Management Console et choisissez CloudFormation dans la liste des services.
2. Sur la page CloudFormation, choisissez Piles, puis Créer une pile avec de nouvelles ressources (standard).
3. Sur la page Créer une pile, choisissez Télécharger un fichier modèle, puis choisissez le fichier `kda-flink-udf.yml` que vous avez téléchargé précédemment. Chargez le fichier, puis sélectionnez Suivant.
4. Donnez un nom au modèle, par exemple pour `kinesis-UDF` afin qu'il soit facile à mémoriser, et mettez à jour les paramètres d'entrée tels que input-stream si vous souhaitez un autre nom. Choisissez Next (Suivant).
5. Sur la page Configurer les options de pile, ajoutez des balises si vous le souhaitez, puis choisissez Suivant.

6. Sur la page Révision, cochez les cases permettant la création de ressources IAM, puis choisissez Envoyer.

Le lancement de la pile AWS CloudFormation peut prendre de 10 à 15 minutes, en fonction de la région dans laquelle vous le lancez. Une fois que vous voyez le statut `CREATE_COMPLETE` pour l'ensemble de la pile, vous êtes prêt à continuer.

Utilisation du bloc-notes du service géré pour Apache Flink Studio

Les blocs-notes Studio pour Kinesis Data Analytics vous permettent d'interroger des flux de données de manière interactive en temps réel, et de créer et d'exécuter facilement des applications de traitement de flux à l'aide des normes SQL, Python et Scala. En quelques clics dans l'AWS Management Console, vous pouvez lancer un bloc-notes sans serveur pour interroger des flux de données et obtenir des résultats en quelques secondes.

Un bloc-notes est un environnement de développement basé sur le Web. Grâce aux blocs-notes, vous bénéficiez d'une expérience de développement interactive simple associée aux capacités avancées de traitement des flux de données fournies par Apache Flink. Les blocs-notes Studio utilisent des blocs-notes alimentés par Apache Zeppelin et se servent d'Apache Flink comme moteur de traitement des flux. Les blocs-notes Studio combinent parfaitement ces technologies pour rendre les analyses avancées sur les flux de données accessibles aux développeurs de tous niveaux de compétences.

Apache Zeppelin fournit à vos blocs-notes Studio une suite complète d'outils d'analyse, y compris les outils suivants :

- Visualisation des données
- Exportation de données dans des fichiers
- Contrôle du format de sortie pour une analyse simplifiée

Utilisation du bloc-notes

1. Accédez à l'AWS Management Console et choisissez Amazon Kinesis dans la liste des services.
2. Dans le volet de navigation de gauche, choisissez Applications d'analyse, puis Blocs-notes Studio.
3. Vérifiez que le bloc-notes KinesisDataAnalyticsStudio est en cours d'exécution.
4. Sélectionnez le bloc-notes, puis choisissez Ouvrir dans Apache Zeppelin.

5. Téléchargez le fichier Data Producer Zeppelin Notebook que vous utiliserez pour lire et charger des données dans le flux Kinesis.
6. Importez le bloc-notes Zeppelin `Data Producer`. Assurez-vous de modifier l'entrée `STREAM_NAME` et `REGION` dans le code du bloc-notes. Le nom du flux d'entrée se trouve dans la sortie de pile AWS CloudFormation.
7. Exécutez le bloc-notes Data Producer en cliquant sur le bouton Exécuter ce paragraphe pour insérer des exemples de données dans le flux de données Kinesis d'entrée.
8. Pendant le chargement des exemples de données, téléchargez le bloc-notes MaskPhoneNumber-Interactive, qui lira les données d'entrée, anonymisera les numéros de téléphone du flux d'entrée et stockera les données anonymisées dans le flux de sortie.
9. Importez le bloc-notes Zeppelin `MaskPhoneNumber-interactive`.
10. Exécutez chaque paragraphe du bloc-notes.

    a. Au paragraphe 1, vous importez une fonction définie par l'utilisateur pour anonymiser les numéros de téléphone.

    ```
    %flink(parallelism=1)
    import com.mycompany.app.MaskPhoneNumber
    stenv.registerFunction("MaskPhoneNumber", new MaskPhoneNumber())
    ```

    b. Dans le paragraphe suivant, vous créez une table en mémoire pour lire les données du flux d'entrée. Assurez-vous que le nom du flux et la région AWS sont corrects.

    ```
    %flink.ssql(type=update)

    DROP TABLE IF EXISTS customer_reviews;

    CREATE TABLE customer_reviews (
    customer_id VARCHAR,
    product VARCHAR,
    review VARCHAR,
    phone VARCHAR
    )
    WITH (
    'connector' = 'kinesis',
    'stream' = 'KinesisUDFSampleInputStream',
    'aws.region' = 'us-east-1',
    'scan.stream.initpos' = 'LATEST',
    'format' = 'json');
    ```

c. Vérifiez si les données sont chargées dans la table en mémoire.

```
%flink.ssql(type=update)
select * from customer_reviews
```

d. Appelez la fonction définie par l'utilisateur pour anonymiser le numéro de téléphone.

```
%flink.ssql(type=update)
select customer_id, product, review, MaskPhoneNumber('mask_phone', phone) as
 phoneNumber from customer_reviews
```

e. Maintenant que les numéros de téléphone sont masqués, créez une vue avec un numéro masqué.

```
%flink.ssql(type=update)

DROP VIEW IF EXISTS sentiments_view;

CREATE VIEW
    sentiments_view
AS
  select customer_id, product, review, MaskPhoneNumber('mask_phone', phone) as
 phoneNumber from customer_reviews
```

f. Vérifiez les données.

```
%flink.ssql(type=update)
select * from sentiments_view
```

g. Créez une table en mémoire pour le flux Kinesis en sortie. Assurez-vous que le nom du flux et la région AWS sont corrects.

```
%flink.ssql(type=update)

DROP TABLE IF EXISTS customer_reviews_stream_table;

CREATE TABLE customer_reviews_stream_table (
customer_id VARCHAR,
product VARCHAR,
review VARCHAR,
phoneNumber varchar
)
```

```
WITH (
'connector' = 'kinesis',
'stream' = 'KinesisUDFSampleOutputStream',
'aws.region' = 'us-east-1',
'scan.stream.initpos' = 'TRIM_HORIZON',
'format' = 'json');
```

h. Insérez les enregistrements mis à jour dans le flux Kinesis cible.

```
%flink.ssql(type=update)
INSERT INTO customer_reviews_stream_table
SELECT customer_id, product, review, phoneNumber
FROM sentiments_view
```

i. Affichez et vérifiez les données du flux Kinesis cible.

```
%flink.ssql(type=update)
select * from customer_reviews_stream_table
```

Promotion d'un bloc-notes en tant qu'application

Maintenant que vous avez testé le code de votre bloc-notes de manière interactive, vous allez le déployer en tant qu'application de streaming durable. Vous devez d'abord modifier la configuration de l'application pour indiquer l'emplacement de votre code dans Amazon S3.

1. Dans l'AWS Management Console, choisissez votre bloc-notes et dans Déployer en tant que configuration d'application - facultatif, sélectionnez Modifier.
2. Sous Destination du code dans Amazon S3, choisissez le compartiment Amazon S3 créé par les scripts AWS CloudFormation. Ce processus peut prendre quelques minutes.
3. Vous ne pourrez pas promouvoir la note telle quelle. Si vous essayez, vous obtenez une erreur car les instructions `Select` ne sont pas prises en charge. Pour éviter ce problème, téléchargez le bloc-notes Zeppelin MaskPhoneNumber-Streaming.
4. Importez le bloc-notes Zeppelin `MaskPhoneNumber-streaming`.
5. Ouvrez la note et choisissez Actions pour KinesisDataAnalyticsStudio.
6. Choisissez Générer MaskPhoneNumber-Streaming et exporter vers S3. Assurez-vous de modifier le nom d'application et de ne pas inclure de caractères spéciaux.
7. Choisissez Générer et exporter. La configuration de l'application de streaming prendra quelques minutes.

8. Une fois la génération terminée, choisissez Déployer à l'aide de la console AWS.
9. Sur la page suivante, vérifiez les paramètres et assurez-vous de choisir le rôle IAM approprié. Ensuite, choisissez Créer une application de streaming.
10. Après quelques minutes, vous verrez un message indiquant que l'application de streaming a été créée avec succès.

Pour plus d'informations sur le déploiement d'applications à état durable et avec limitations, consultez Déploiement en tant qu'application à état durable.

Nettoyage

Si vous le souhaitez, vous pouvez à présent désinstaller la pile AWS CloudFormation. Cela supprimera tous les services que vous avez configurés précédemment.

Exemples de Kinesis Data Analytics pour SQL

Cette section fournit des exemples de création et d'utilisation d'applications dans Amazon Kinesis Data Analytics. Ils comprennent un exemple de code et des instructions détaillées pour vous aider à créer des applications Kinesis Data Analytics et tester vos résultats.

Avant d'explorer ces exemples, nous vous recommandons de consulter Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement et Mise en route avec les applications Amazon Kinesis Data Analytics pour SQL.

Rubriques

Exemples : Transformation de données

Parfois, votre code d'application doit prétraiter les enregistrements entrants avant d'exécuter des analyses dans Amazon Kinesis Data Analytics. Différentes raisons peuvent entraîner ce prétraitement, par exemple quand des enregistrements sont non conformes aux formats d'enregistrement pris en charge, ce qui peut se traduire par des colonnes non normalisées dans les flux d'entrée intégrés à l'application.

Cette section fournit des exemples montrant comment utiliser les fonctions de chaîne disponibles pour normaliser des données, comment extraire les informations dont vous avez besoin de colonnes de chaîne, etc. La section fait également référence à des fonctions de date et d'heure pouvant vous être utiles.

Prétraitement des flux avec Lambda

Pour plus d'informations sur le prétraitement des flux avec AWS Lambda, consultezPrétraitement des données à l'aide d'une fonction Lambda.

Rubriques

Exemples : Transformation de valeurs de chaîne

Amazon Kinesis Data Analytics prend en charge des formats tels que JSON et CSV pour des enregistrements d'une source de streaming. Pour plus de détails, veuillez consulter RecordFormat. Ces enregistrements sont alors mappés à un flux intégré à l'application, selon la configuration d'entrée. Pour plus de détails, veuillez consulter Configuration de l'entrée de l'application. La configuration d'entrée spécifie la façon dont des champs d'enregistrement de la source de diffusion sont mappés à des colonnes d'un flux intégré à l'application.

Ce mappage fonctionne lorsque des enregistrements de la source de diffusion suivent les formats pris en charge, ce qui se traduit par un flux intégré à l'application avec des données normalisées. Mais, que ce passe-t-il si les données de votre source de diffusion ne sont pas conforme aux normes prises en charge ? Par exemple, que ce passe-t-il si votre source de diffusion contient des données telles que des données de flux de clics, des capteurs IoT et des journaux d'application ?

Prenez en compte les exemples suivants :

- La source de diffusion contient des journaux d'application : Les journaux d'application suivent le format de journal Apache standard et sont écrits dans le flux à l'aide du format JSON.

  ```
  {
     "Log":"192.168.254.30 - John [24/May/2004:22:01:02 -0700] "GET /icons/
  apache_pb.gif HTTP/1.1" 304 0"
  }
  ```

  Pour plus d'informations sur le format de journal Apache standard, consultez Fichiers journaux sur le site Web d'Apache.

- La source de streaming contient des données semi-structurées : L'exemple suivant montre deux enregistrements. La valeur du champ `Col_E_Unstructured` est une série de valeurs séparées par des virgules. Il existe cinq colonnes. Les quatre premières ont des valeurs de type chaîne et la dernière colonne contient des valeurs séparées par des virgules.

```
{ "Col_A" : "string",
  "Col_B" : "string",
  "Col_C" : "string",
  "Col_D" : "string",
  "Col_E_Unstructured" : "value,value,value,value"}

{ "Col_A" : "string",
  "Col_B" : "string",
  "Col_C" : "string",
  "Col_D" : "string",
  "Col_E_Unstructured" : "value,value,value,value"}
```

- Les enregistrements de votre source de diffusion contiennent des URL et vous avez besoin d'une partie du nom de domaine URL pour les analyses.

```
{ "referrer" : "http://www.amazon.com"}
{ "referrer" : "http://www.stackoverflow.com" }
```

Dans de tels cas, le processus en deux étapes suivant fonctionne généralement pour la création de flux intégrés à l'application contenant des données normalisées :

1. Configurez l'entrée d'application pour mapper le champ non structuré à une colonne de type `VARCHAR(N)` dans le flux d'entrée intégré à l'application qui est créé.
2. Dans votre code d'application, utilisez des fonctions de chaîne pour scinder cette colonne unique en plusieurs colonnes, puis enregistrez les lignes dans un autre flux intégré à l'application. Ce flux intégré à l'application créé par votre code d'application contiendra des données normalisées. Vous pouvez alors exécuter des analyses sur ce flux intégré à l'application.

Amazon Kinesis Data Analytics fournit les opérations de chaîne, les fonctions SQL standard et les extensions de la norme SQL suivante pour gérer les colonnes de chaîne :

- Opérateurs de chaînes : Les opérateurs tels que `LIKE` et `SIMILAR` sont utiles pour la comparaison de chaînes. Pour plus d'informations, consultez la section Opérateurs de chaîne dans le manuel Référence SQL du service géré Amazon pour Apache Flink.
- Fonctions SQL : Les fonctions suivantes sont utiles lors de la manipulation de chaînes individuelles. Pour plus d'informations, consultez la section Fonctions de chaîne et de recherche dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

- `CHAR_LENGTH` – Fournit la longueur d'une chaîne.
- `INITCAP`– Renvoie une version convertie de la chaîne en entrée de telle sorte que le premier caractère de chaque mot délimité par un espace soit en majuscules et que tous les autres caractères soient en minuscules.
- `LOWER/UPPER` – Convertit une chaîne en minuscules ou en majuscules.
- `OVERLAY` – Remplace une partie du premier argument de chaîne (chaîne d'origine) par le deuxième argument de chaîne (chaîne de remplacement).
- `POSITION` : Recherche une chaîne dans une autre chaîne.
- `REGEX_REPLACE` – Remplace une sous-chaîne par une autre sous-chaîne.
- `SUBSTRING` : Extrait une partie d'une chaîne source à partir d'une position spécifique.
- `TRIM` – Supprime les instances du caractère spécifié depuis le début ou la fin de la chaîne source.

- Extensions SQL : Ces extensions sont utiles pour travailler avec des chaînes non structurées, telles que des journaux et des URI. Pour plus d'informations, consultez la section Fonctions d'analyse de journal dans le manuel Référence SQL du service géré Amazon pour Apache Flink.
  - `FAST_REGEX_LOG_PARSER` : Fonctionne comme l'analyseur d'expressions régulières, mais accepte plusieurs raccourcis pour fournir des résultats plus rapides. Par exemple, l'analyseur d'expression régulière rapide s'arrête à la première correspondance trouvée (processus appelé sémantique paresseuse).
  - `FIXED_COLUMN_LOG_PARSE` – Analyse des champs à largeur fixe et les convertit automatiquement aux types SQL donnés.
  - `REGEX_LOG_PARSE` – Analyse une chaîne selon des modèles d'expression régulière Java par défaut.
  - `SYS_LOG_PARSE` – Traite les entrées généralement trouvées dans les journaux système UNIX/Linux.
  - `VARIABLE_COLUMN_LOG_PARSE` – Scinde une chaîne d'entrée en champs séparés par un caractère délimiteur ou une chaîne de délimiteur.
  - `W3C_LOG_PARSE` – Peut être utilisé pour formater rapidement des journaux Apache.

Pour obtenir des exemples utilisant ces fonctions, consultez les rubriques suivantes :

Rubriques

- Exemple : Extraction d'une partie d'une chaîne (fonction SUBSTRING)

- Exemple : Remplacement d'une sous-chaîne à l'aide de Regex (fonction REGEX_REPLACE)
- Exemple : Analyse de chaînes de journal en fonction d'expressions régulières (fonction REGEX_LOG_PARSE)
- Exemple : Analyse de journaux web (fonction W3C_LOG_PARSE)
- Exemple : Fractionnement de chaînes en plusieurs champs (fonction VARIABLE_COLUMN_LOG_PARSE)

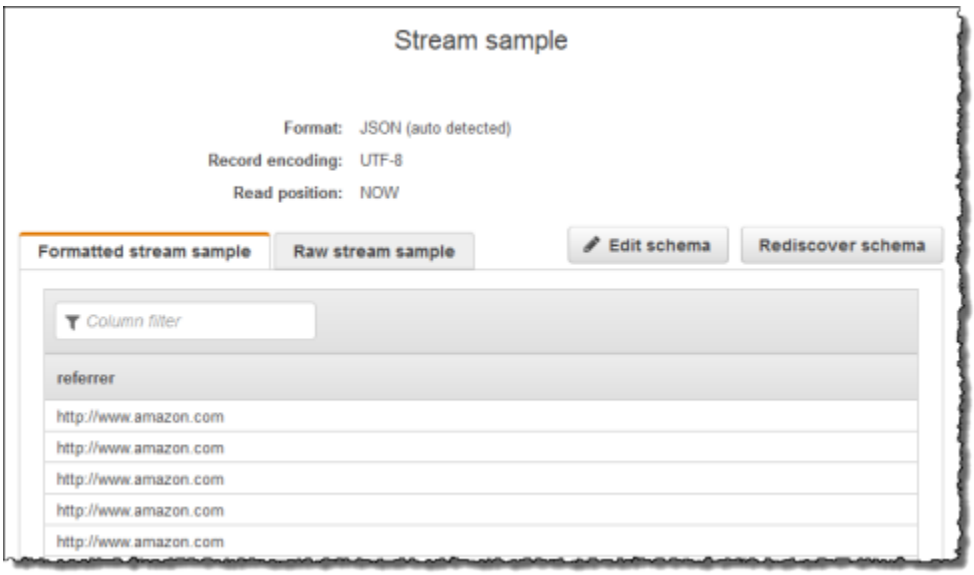
Exemple : Extraction d'une partie d'une chaîne (fonction SUBSTRING)

Cet exemple utilise la fonction `SUBSTRING` pour transformer une chaîne dans Amazon Kinesis Data Analytics. La fonction `SUBSTRING` extrait une partie d'une chaîne source à partir d'une position spécifique. Pour plus d'informations, consultez la section SUBSTRING dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

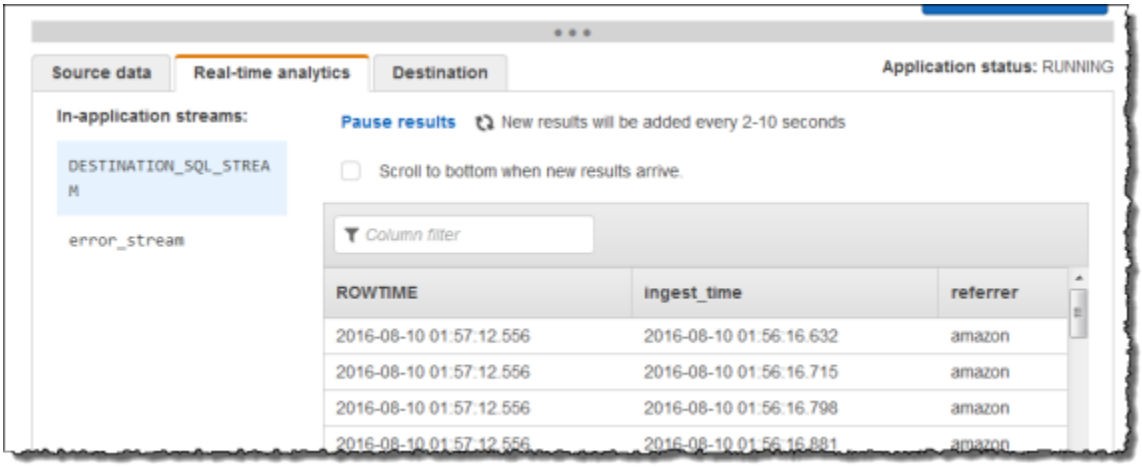
Dans cet exemple, vous écrivez les enregistrements suivants dans un flux Amazon Kinesis.

```
{ "REFERRER" : "http://www.amazon.com" }
{ "REFERRER" : "http://www.amazon.com"}
{ "REFERRER" : "http://www.amazon.com"}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans la console, à l'aide du flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements de la source de diffusion et déduit un schéma intégré à l'application avec une colonne (`REFERRER`), comme illustré.



Vous pouvez ensuite utiliser le code de l'application avec la fonction `SUBSTRING` pour analyser la chaîne d'URL afin de récupérer le nom de la société. Vous devez ensuite insérer les données obtenues dans un autre flux de l'application, comme indiqué ci-dessous :



Rubriques

- Étape 1 : Création d’un flux de données Kinesis
- Étape 2 : Création d’une application Kinesis Data Analytics

Étape 1 : Création d’un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements de journaux comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Exécutez le code Python suivant pour remplir les exemples d'enregistrements de journal. Ce code simple écrit en continu le même enregistrement de journal dans le flux.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"REFERRER": "http://www.amazon.com"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Ensuite, créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.

2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion).
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez l'option de création d'un rôle IAM.

   c. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit ne comporte qu'une seule colonne.

   d. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
-- CREATE OR REPLACE STREAM for cleaned up referrer
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    "ingest_time" TIMESTAMP,
    "referrer" VARCHAR(32));

CREATE OR REPLACE PUMP "myPUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM
         "APPROXIMATE_ARRIVAL_TIME",
         SUBSTRING("referrer", 12, (POSITION('.com' IN "referrer") -
 POSITION('www.' IN "referrer") - 4))
      FROM "SOURCE_SQL_STREAM_001";
```

   b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

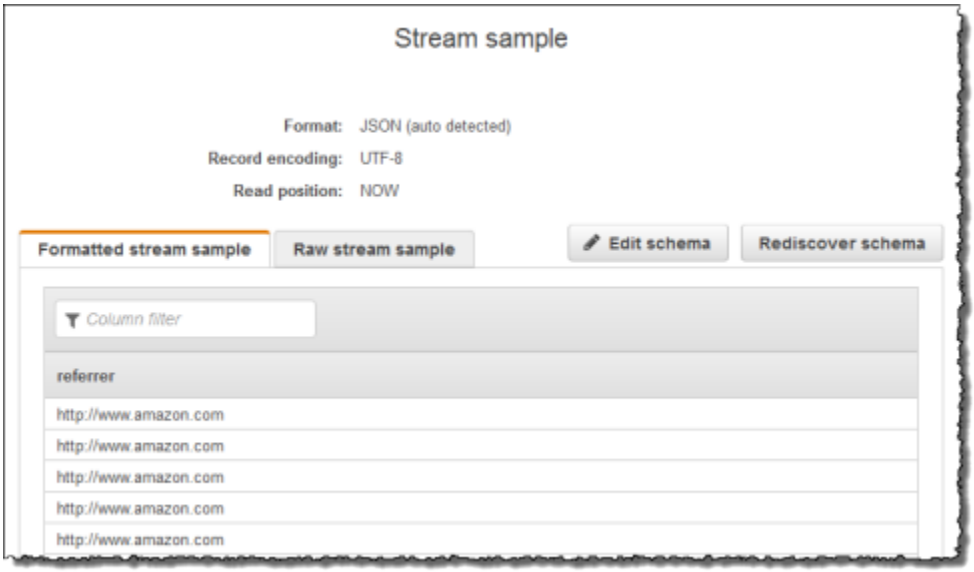
Exemple : Remplacement d'une sous-chaîne à l'aide de Regex (fonction REGEX_REPLACE)

Cet exemple utilise la fonction `REGEX_REPLACE` pour transformer une chaîne dans Amazon Kinesis Data Analytics. `REGEX_REPLACE` remplace une sous-chaîne par une autre sous-chaîne. Pour plus d'informations, consultez la section REGEX_REPLACE dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

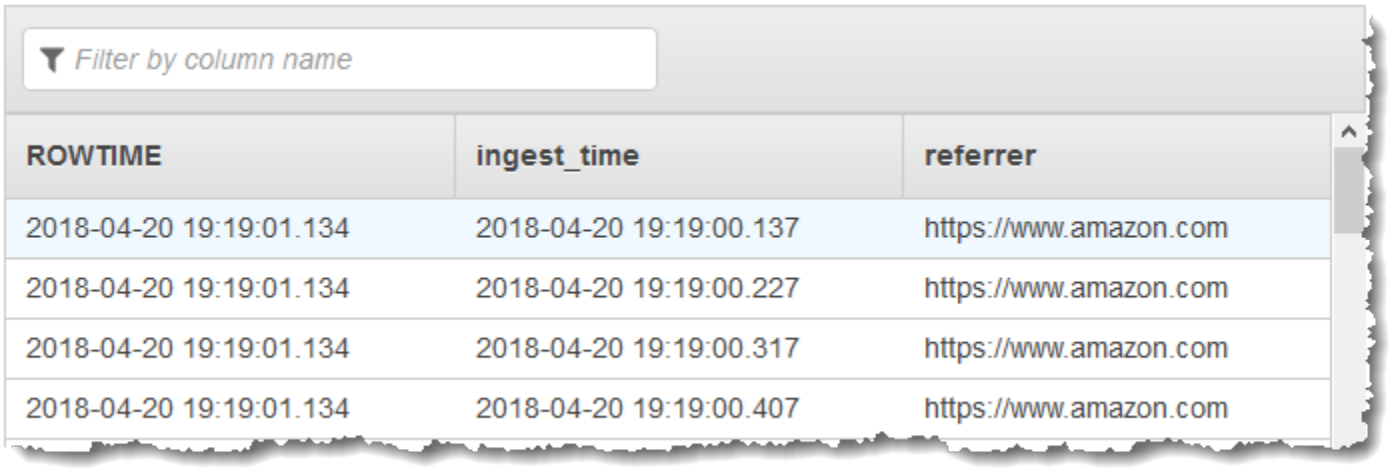
Dans cet exemple, vous écrivez les enregistrements suivants dans un flux Amazon Kinesis :

```
{ "REFERRER" : "http://www.amazon.com" }
{ "REFERRER" : "http://www.amazon.com"}
{ "REFERRER" : "http://www.amazon.com"}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans la console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements de la source de diffusion et déduit un schéma intégré à l'application avec une colonne (REFERRER), comme illustré.



Vous utilisez ensuite le code d'application avec la fonction `REGEX_REPLACE` pour convertir l'URL afin d'utiliser `https://` au lieu de `http://`. Vous insérez les données résultantes dans un autre flux de l'application, comme indiqué ci-dessous :



Rubriques

Étape 1 : Création d'un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements de journaux comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Exécutez le code Python suivant pour remplir les exemples d'enregistrements de journal. Ce code simple écrit en continu le même enregistrement de journal dans le flux.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
```

```
    return {"REFERRER": "http://www.amazon.com"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Ensuite, créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion).
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :
   a. Choisissez le flux que vous avez créé dans la section précédente.
   b. Choisissez l'option de création d'un rôle IAM.
   c. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit ne comporte qu'une seule colonne.
   d. Choisissez Save and continue (Enregistrer et continuer).
5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.

6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

    a. Copiez le code d'application suivant et collez-le dans l'éditeur :

```
-- CREATE OR REPLACE STREAM for cleaned up referrer
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    "ingest_time" TIMESTAMP,
    "referrer" VARCHAR(32));

CREATE OR REPLACE PUMP "myPUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM
         "APPROXIMATE_ARRIVAL_TIME",
         REGEX_REPLACE("REFERRER", 'http://', 'https://', 1, 0)
      FROM "SOURCE_SQL_STREAM_001";
```

    b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Exemple : Analyse de chaînes de journal en fonction d'expressions régulières (fonction REGEX_LOG_PARSE)

Cet exemple utilise la fonction `REGEX_LOG_PARSE` pour transformer une chaîne dans Amazon Kinesis Data Analytics. `REGEX_LOG_PARSE` analyse une chaîne selon des modèles d'expression régulière Java par défaut. Pour plus d'informations, consultez la section REGEX_LOG_PARSE dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

Dans cet exemple, vous écrivez les enregistrements suivants dans un flux Amazon Kinesis :

```
{"LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] \"GET /index.php HTTP/1.1\"
 200 125 \"-\" \"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0\""}
{"LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] \"GET /index.php HTTP/1.1\"
 200 125 \"-\" \"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0\""}
{"LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] \"GET /index.php HTTP/1.1\"
 200 125 \"-\" \"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0\""}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans la console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements

de la source de diffusion et déduit un schéma intégré à l'application avec une colonne (LOGENTRY), comme illustré ci-après :

ROWTIME TIMESTAMP	LOGENTRY VARCHAR(256)
2018-05-09 18:12:18.552	203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "GET /index.php HTTP/1.1"
2018-05-09 18:12:18.552	203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "GET /index.php HTTP/1.1"
2018-05-09 18:12:18.552	203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "GET /index.php HTTP/1.1"
2018-05-09 18:12:18.552	203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "GET /index.php HTTP/1.1"

Vous pouvez ensuite utiliser le code de l'application avec la fonction `REGEX_LOG_PARSE` pour analyser la chaîne de journal afin de récupérer les éléments de données. Vous insérez les données obtenues dans un autre flux intégré à l'application, comme indiqué dans la capture d'écran suivante :

Filter by column name

ROWTIME	LOGENTRY	MATCH1	MATCH2
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [

Rubriques

Étape 1 : Création d’un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements de journaux comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Exécutez le code Python suivant pour remplir les exemples d'enregistrements de journal. Ce code simple écrit en continu le même enregistrement de journal dans le flux.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "
        '"GET /index.php HTTP/1.1" 200 125 "-" '
        '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d’une application Kinesis Data Analytics

Ensuite, créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application et spécifiez un nom d'application.
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion).
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez l'option de création d'un rôle IAM.

   c. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit ne comporte qu'une seule colonne.

   d. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (logentry VARCHAR(24), match1
 VARCHAR(24), match2 VARCHAR(24));

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM T.LOGENTRY, T.REC.COLUMN1, T.REC.COLUMN2
    FROM
         (SELECT STREAM LOGENTRY,
             REGEX_LOG_PARSE(LOGENTRY, '(\w.+) (\d.+) (\w.+) (\w.+)') AS REC
             FROM SOURCE_SQL_STREAM_001) AS T;
```

   b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

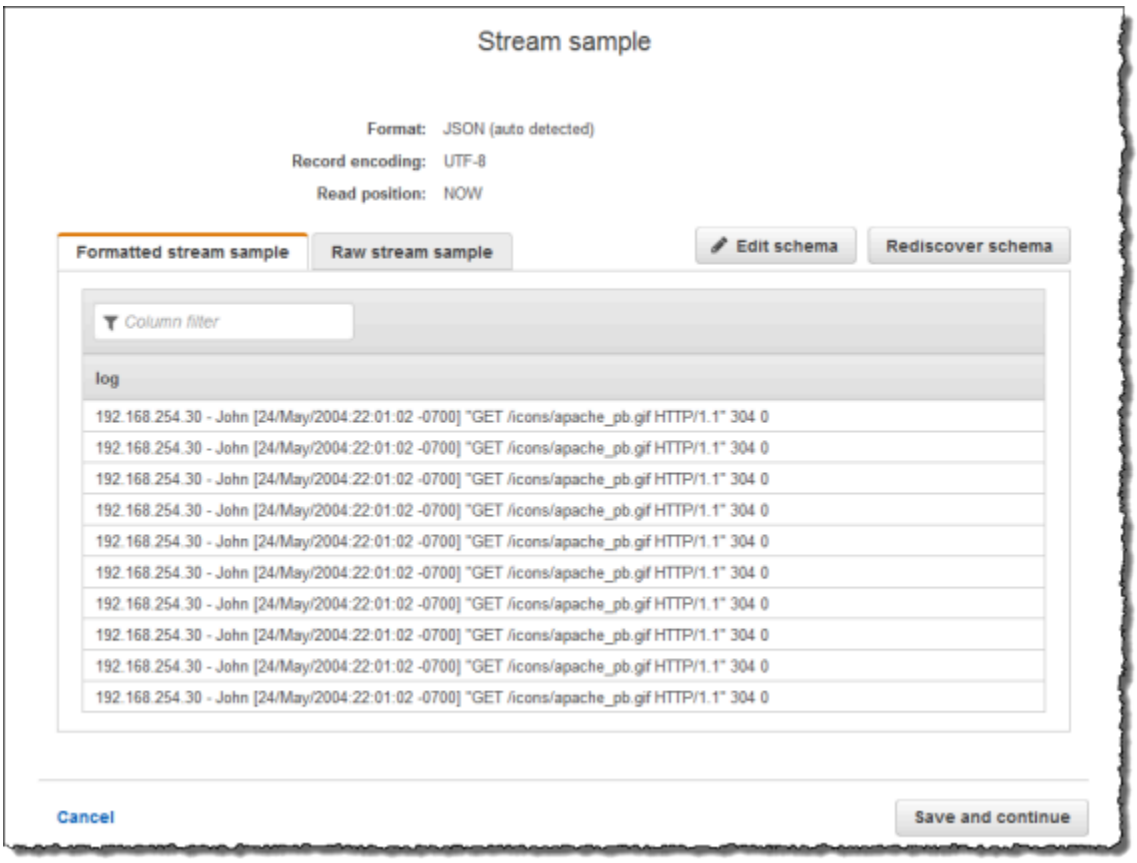
Exemple : Analyse de journaux web (fonction W3C_LOG_PARSE)

Cet exemple utilise la fonction `W3C_LOG_PARSE` pour transformer une chaîne dans Amazon Kinesis Data Analytics. Vous pouvez utiliser `W3C_LOG_PARSE` pour formater rapidement des journaux Apache. Pour plus d'informations, consultez la section W3C_LOG_PARSE dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

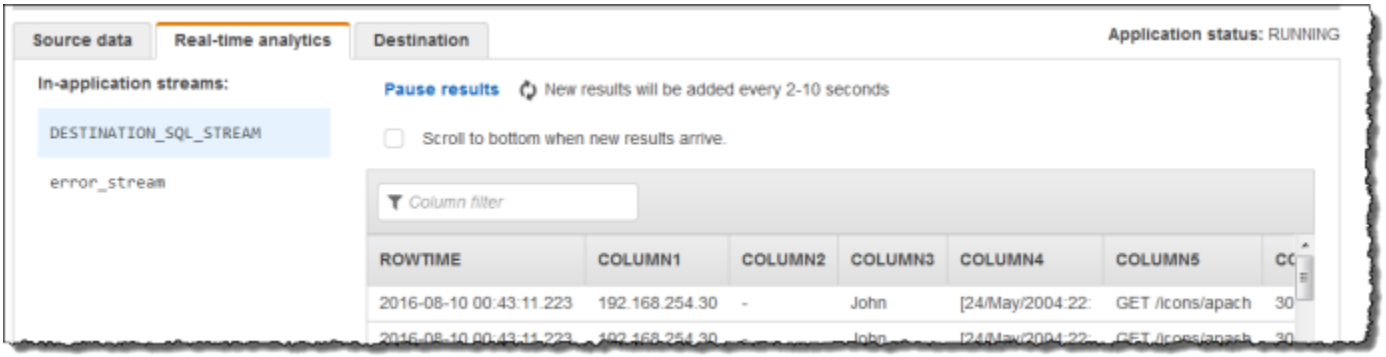
Dans cet exemple, vous écrivez des enregistrements de journaux dans un flux de données Amazon Kinesis. Les exemples de journaux affichés sont les suivants :

```
{"Log":"192.168.254.30 - John [24/May/2004:22:01:02 -0700] "GET /icons/apache_pba.gif
 HTTP/1.1" 304 0"}
{"Log":"192.168.254.30 - John [24/May/2004:22:01:03 -0700] "GET /icons/apache_pbb.gif
 HTTP/1.1" 304 0"}
{"Log":"192.168.254.30 - John [24/May/2004:22:01:04 -0700] "GET /icons/apache_pbc.gif
 HTTP/1.1" 304 0"}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans la console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements de la source de diffusion et déduit un schéma intégré à l'application avec une colonne (log), comme illustré ci-après :



Ensuite, vous utilisez le code d'application avec la fonction `W3C_LOG_PARSE` pour analyser le journal et créer un autre flux intégré à l'application avec différents champs de journal dans des colonnes distinctes, comme illustré ci-après :



Rubriques

- [Étape 1 : Création d’un flux de données Kinesis]
- [Étape 2 : Création d’une application Kinesis Data Analytics]

Étape 1 : Création d'un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements de journaux comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Exécutez le code Python suivant pour remplir les exemples d'enregistrements de journal. Ce code simple écrit en continu le même enregistrement de journal dans le flux.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "log": "192.168.254.30 - John [24/May/2004:22:01:02 -0700] "
        '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion).
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :
   a. Choisissez le flux que vous avez créé dans la section précédente.
   b. Choisissez l'option de création d'un rôle IAM.
   c. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit ne comporte qu'une seule colonne.
   d. Choisissez Save and continue (Enregistrer et continuer).
5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :
   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
column1 VARCHAR(16),
column2 VARCHAR(16),
column3 VARCHAR(16),
column4 VARCHAR(16),
column5 VARCHAR(16),
column6 VARCHAR(16),
column7 VARCHAR(16));

CREATE OR REPLACE PUMP "myPUMP" AS
INSERT INTO "DESTINATION_SQL_STREAM"
        SELECT STREAM
```

```
            l.r.COLUMN1,
            l.r.COLUMN2,
            l.r.COLUMN3,
            l.r.COLUMN4,
            l.r.COLUMN5,
            l.r.COLUMN6,
            l.r.COLUMN7
        FROM (SELECT STREAM W3C_LOG_PARSE("log", 'COMMON')
              FROM "SOURCE_SQL_STREAM_001") AS l(r);
```

b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

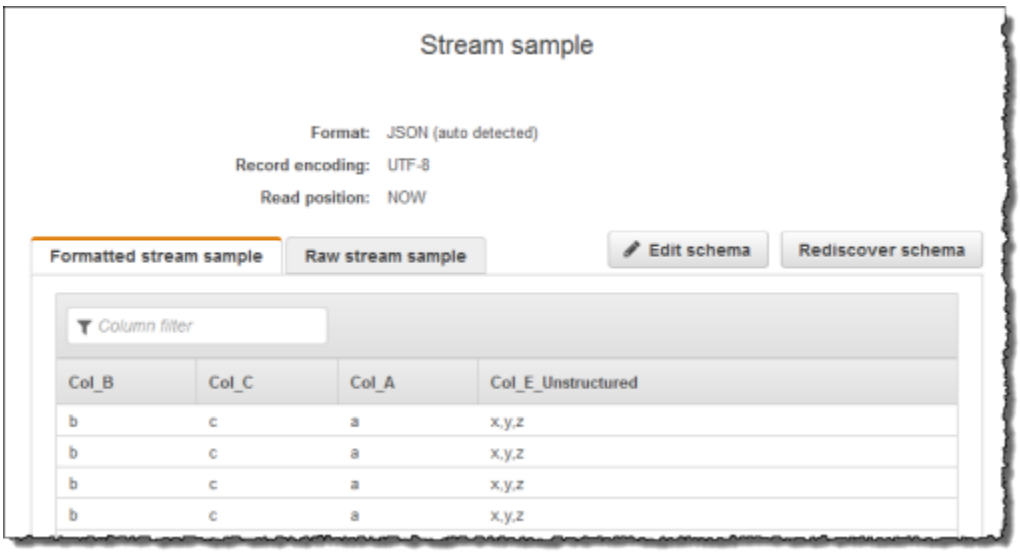
Exemple : Fractionnement de chaînes en plusieurs champs (fonction VARIABLE_COLUMN_LOG_PARSE)

Cet exemple utilise la fonction `VARIABLE_COLUMN_LOG_PARSE` pour manipuler des chaînes dans Kinesis Data Analytics. `VARIABLE_COLUMN_LOG_PARSE` fractionne une chaîne d'entrée en champs séparés par un caractère délimiteur ou une chaîne de délimiteur. Pour plus d'informations, consultez la section VARIABLE_COLUMN_LOG_PARSE dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

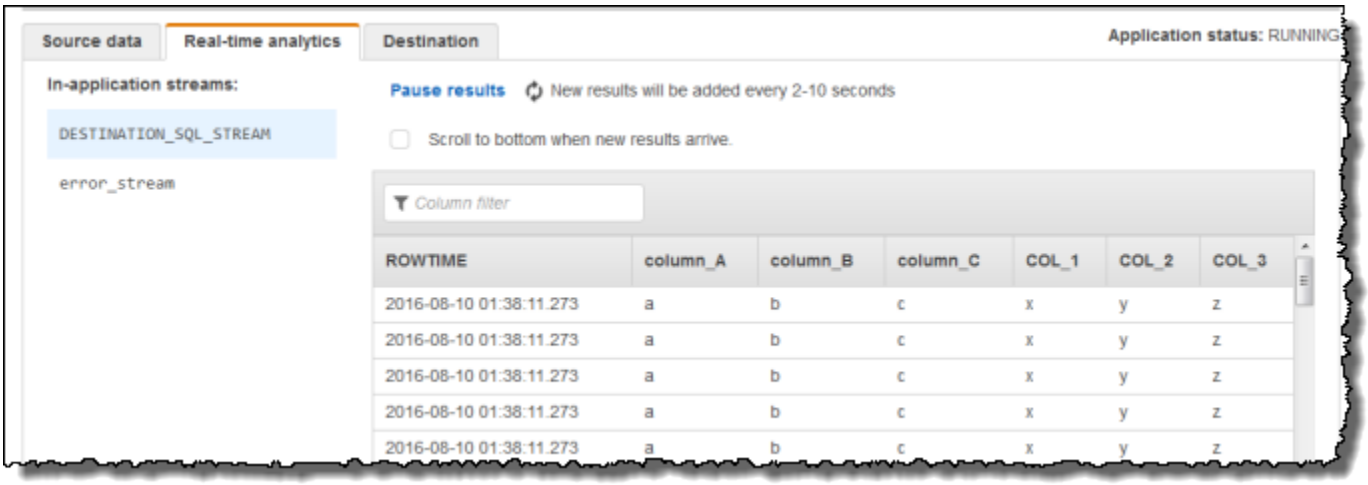
Dans cet exemple, vous écrivez des enregistrements semi-structurés dans un flux de données Amazon Kinesis. Les exemples d'enregistrements sont les suivants :

```
{ "Col_A" : "string",
  "Col_B" : "string",
  "Col_C" : "string",
  "Col_D_Unstructured" : "value,value,value,value"}
{ "Col_A" : "string",
  "Col_B" : "string",
  "Col_C" : "string",
  "Col_D_Unstructured" : "value,value,value,value"}
```

Vous créez ensuite une application Kinesis Data Analytics dans la console, à l'aide du flux Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements de la source de diffusion et déduit un schéma intégré à l'application avec quatre colonnes, comme illustré ci-après :



Vous utilisez ensuite le code d'application avec la fonction `VARIABLE_COLUMN_LOG_PARSE` pour analyser les valeurs séparées par des virgules et insérer des lignes normalisées dans un autre flux intégré à l'application, comme illustré ci-après :



Rubriques

- Étape 1 : Création d’un flux de données Kinesis
- Étape 2 : Création d’une application Kinesis Data Analytics

Étape 1 : Création d’un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements de journaux comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Exécutez le code Python suivant pour remplir les exemples d'enregistrements de journal. Ce code simple écrit en continu le même enregistrement de journal dans le flux.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"Col_A": "a", "Col_B": "b", "Col_C": "c", "Col_E_Unstructured":
 "x,y,z"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d’une application Kinesis Data Analytics

Créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion).
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez l'option de création d'un rôle IAM.

   c. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements utilisés pour déduire le schéma pour le flux intégré à l'application créé. Notez que le schéma déduit ne comporte qu'une seule colonne.

   d. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code de l'application, puis vérifiez les résultats :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur:

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM"(
            "column_A" VARCHAR(16),
            "column_B" VARCHAR(16),
            "column_C" VARCHAR(16),
            "COL_1" VARCHAR(16),
            "COL_2" VARCHAR(16),
            "COL_3" VARCHAR(16));

CREATE OR REPLACE PUMP "SECOND_STREAM_PUMP" AS
INSERT INTO "DESTINATION_SQL_STREAM"
   SELECT STREAM  t."Col_A", t."Col_B", t."Col_C",
                  t.r."COL_1", t.r."COL_2", t.r."COL_3"
   FROM (SELECT STREAM
           "Col_A", "Col_B", "Col_C",
           VARIABLE_COLUMN_LOG_PARSE ("Col_E_Unstructured",
```

```
                                        'COL_1 TYPE VARCHAR(16), COL_2 TYPE
VARCHAR(16), COL_3 TYPE VARCHAR(16)',
                                        ',') AS r
        FROM "SOURCE_SQL_STREAM_001") as t;
```

b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Exemple : transformation DateTime des valeurs

Amazon Kinesis Data Analytics prend en charge la conversion de colonnes en horodatages. Par exemple, vous pouvez utiliser votre propre horodatage dans le cadre d'une clause `GROUP BY` en tant qu'autre fenêtre temporelle, en plus de la colonne `ROWTIME`. Kinesis Data Analytics fournit des opérations et des fonctions SQL pour gérer les champs de date et d'heure.

- Opérateurs de date et d'heure : Vous pouvez effectuer des opérations arithmétiques sur les dates, les heures et les types de données d'intervalle. Pour plus d'informations, consultez la section Opérateurs de date, d'horodatage et d'intervalle dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

- Fonctions SQL : Ces fonctions incluent les fonctions suivantes. Pour plus d'informations, consultez la section Fonctions de date et d'heure dans le manuel Référence SQL du service géré Amazon pour Apache Flink.
  - `EXTRACT()` : Extrait un champ d'une expression de date, d'heure, d'horodatage ou d'intervalle.
  - `CURRENT_TIME` : Renvoie l'heure à laquelle la requête s'exécute (UTC).
  - `CURRENT_DATE` : Renvoie la date à laquelle la requête s'exécute (UTC).
  - `CURRENT_TIMESTAMP` : Renvoie l'horodatage du moment où lorsque la requête s'exécute (UTC).
  - `LOCALTIME` : Renvoie l'heure en cours à laquelle la requête s'exécute telle que définie par l'environnement d'exécution de Kinesis Data Analytics (UTC).
  - `LOCALTIMESTAMP` : Renvoie l'horodatage actuel, tel que défini par l'environnement d'exécution de Kinesis Data Analytics (UTC).

- Extensions SQL : Ces extensions incluent les extensions suivantes. Pour plus d'informations, consultez les sections Fonctions de date et d'heure et Fonctions de conversion de date et d'heure dans le manuel Référence SQL du service géré Amazon pour Apache Flink.
  - `CURRENT_ROW_TIMESTAMP` : Renvoie un nouvel horodatage pour chaque ligne du flux.
  - `TSDIFF` : Renvoie la différence entre deux horodatages en millisecondes.
  - `CHAR_TO_DATE` : Convertit une chaîne en date.
  - `CHAR_TO_TIME` : Convertit une chaîne en heure.
  - `CHAR_TO_TIMESTAMP` : Convertit une chaîne en horodatage.
  - `DATE_TO_CHAR` : Convertit une date en chaîne.
  - `TIME_TO_CHAR` : Convertit une heure en chaîne.
  - `TIMESTAMP_TO_CHAR` : Convertit un horodatage en chaîne.

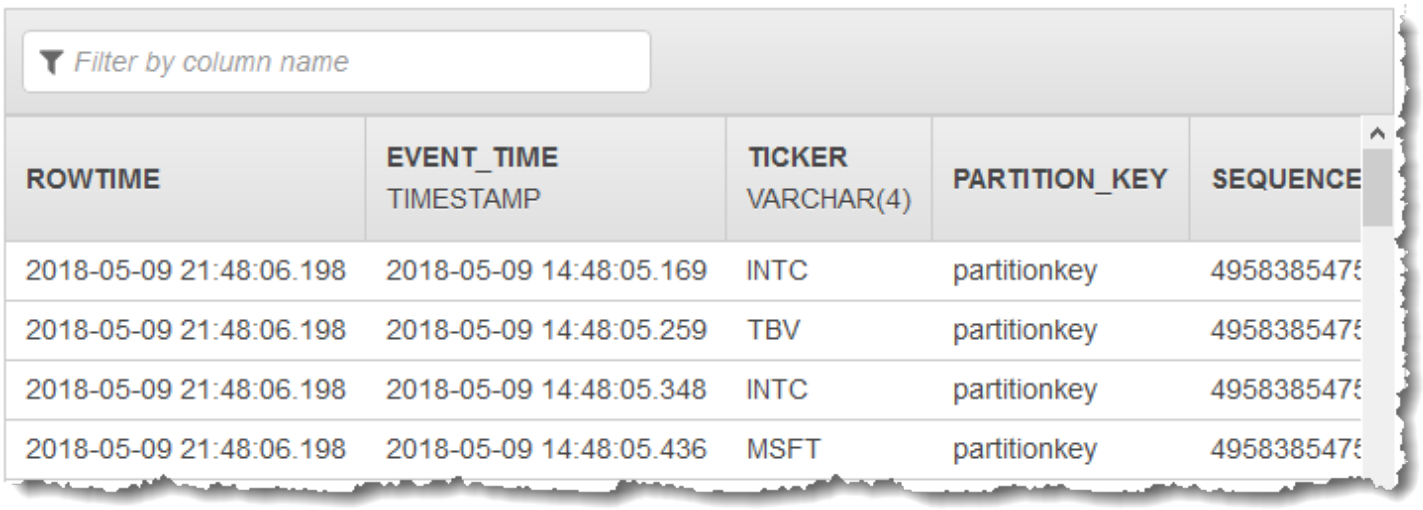
La plupart des fonctions SQL précédentes utilisent un format pour convertir les colonnes. Le format est flexible. Par exemple, vous pouvez spécifier le format `yyyy-MM-dd hh:mm:ss` pour convertir une chaîne d'entrée `2009-09-16 03:15:24` en horodatage. Pour plus d'informations, consultez la section Char vers Timestamp(Sys) dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

Exemples : Transformation de dates

Dans cet exemple, vous écrivez les enregistrements suivants dans un flux de données Amazon Kinesis.

```
{"EVENT_TIME": "2018-05-09T12:50:41.337510", "TICKER": "AAPL"}
{"EVENT_TIME": "2018-05-09T12:50:41.427227", "TICKER": "MSFT"}
{"EVENT_TIME": "2018-05-09T12:50:41.520549", "TICKER": "INTC"}
{"EVENT_TIME": "2018-05-09T12:50:41.610145", "TICKER": "MSFT"}
{"EVENT_TIME": "2018-05-09T12:50:41.704395", "TICKER": "AAPL"}
...
```

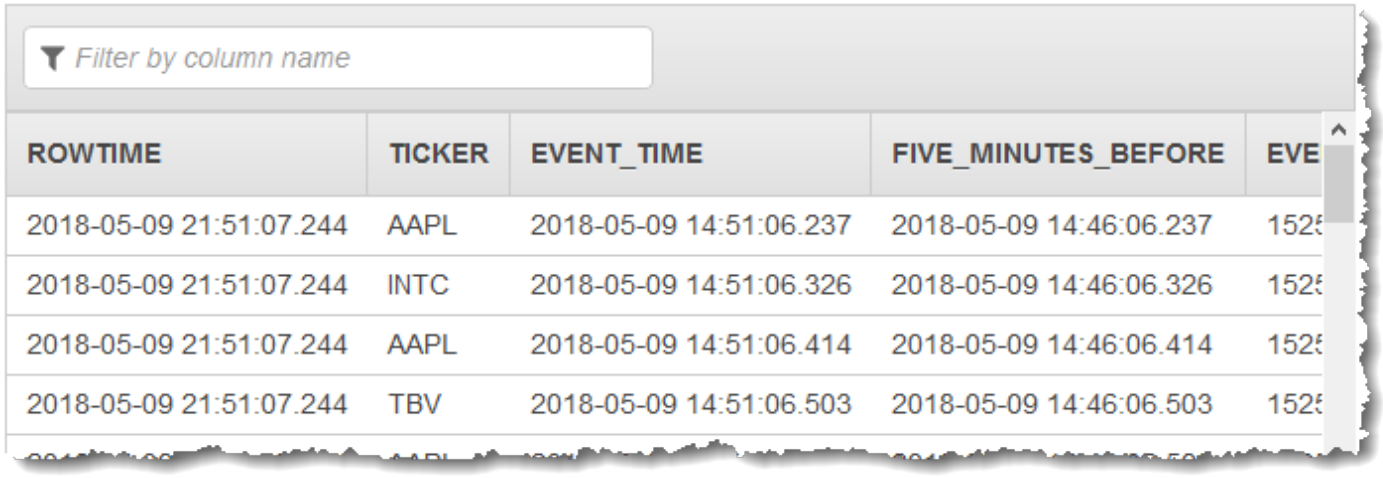
Vous créez ensuite une application Kinesis Data Analytics dans la console, avec le flux Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements de la source de diffusion et déduit un schéma intégré à l'application avec une deux colonnes (`EVENT_TIME` et `TICKER`), comme illustré.



Filter by column name

ROWTIME	EVENT_TIME TIMESTAMP	TICKER VARCHAR(4)	PARTITION_KEY	SEQUENCE
2018-05-09 21:48:06.198	2018-05-09 14:48:05.169	INTC	partitionkey	4958385475
2018-05-09 21:48:06.198	2018-05-09 14:48:05.259	TBV	partitionkey	4958385475
2018-05-09 21:48:06.198	2018-05-09 14:48:05.348	INTC	partitionkey	4958385475
2018-05-09 21:48:06.198	2018-05-09 14:48:05.436	MSFT	partitionkey	4958385475

Vous utilisez ensuite le code d'application avec des fonctions SQL pour convertir le champ d'horodatage `EVENT_TIME` de différentes manières. Puis vous insérez les données obtenues dans un autre flux intégré à l'application, comme indiqué dans la capture d'écran suivante :



Filter by column name

ROWTIME	TICKER	EVENT_TIME	FIVE_MINUTES_BEFORE	EVE
2018-05-09 21:51:07.244	AAPL	2018-05-09 14:51:06.237	2018-05-09 14:46:06.237	1525
2018-05-09 21:51:07.244	INTC	2018-05-09 14:51:06.326	2018-05-09 14:46:06.326	1525
2018-05-09 21:51:07.244	AAPL	2018-05-09 14:51:06.414	2018-05-09 14:46:06.414	1525
2018-05-09 21:51:07.244	TBV	2018-05-09 14:51:06.503	2018-05-09 14:46:06.503	1525

Étape 1 : Création d’un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez-le avec des enregistrements d’heure d’événement et de symbole boursier comme suit :

1. Connectez-vous à la console Kinesis AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.

3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition.
4. Exécutez le code Python suivant pour remplir le flux avec des exemples de données. Ce code simple écrit un enregistrement en continu avec un symbole boursier aléatoire et l'horodatage actuel dans le flux.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d’une application Amazon Kinesis Data Analytics

Créez une application comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion) pour vous connecter à la source.
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez de créer un rôle IAM.

   c. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements qui sont utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit comporte deux colonnes.

   d. Choisissez Edit schema (Modifier le schéma). Remplacez le Column type (Type de colonne) de la colonne EVENT_TIME par `TIMESTAMP`.

   e. Choisissez Save schema and update stream samples (Enregistrer le schéma et mettre à jour les exemples de flux). Une fois que la console a enregistré le schéma, choisissez Exit (Quitter).

   f. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    TICKER VARCHAR(4),
    event_time TIMESTAMP,
    five_minutes_before TIMESTAMP,
    event_unix_timestamp BIGINT,
    event_timestamp_as_char VARCHAR(50),
    event_second INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
```

```
SELECT STREAM
    TICKER,
    EVENT_TIME,
    EVENT_TIME - INTERVAL '5' MINUTE,
    UNIX_TIMESTAMP(EVENT_TIME),
    TIMESTAMP_TO_CHAR('yyyy-MM-dd hh:mm:ss', EVENT_TIME),
    EXTRACT(SECOND FROM EVENT_TIME)
FROM "SOURCE_SQL_STREAM_001"
```

b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Exemple : Transformation de plusieurs types de données

Une exigence commune des applications d'extraction, de transformation et de chargement (ETL) est de pouvoir traiter plusieurs types d'enregistrement sur une source de diffusion. Vous pouvez créer des applications Kinesis Data Analytics pour traiter ces types de sources de streaming. Procédez comme suit :

1. D'abord, vous mappez la source de streaming à un flux d'entrée intégré à l'application, similaire à toutes les autres applications Kinesis Data Analytics.
2. Ensuite, dans votre code d'application, vous écrivez des instructions SQL pour récupérer des lignes de types spécifiques depuis le flux d'entrée intégré à l'application. Puis vous les insérez dans des flux intégrés à l'application distincts. (Vous pouvez créer des flux intégrés à l'application supplémentaires dans votre code d'application.)

Dans cet exercice, vous disposez d'une source de diffusion qui reçoit des enregistrements de deux types (`Order` et `Trade`). Il s'agit d'ordres de bourse et des transactions correspondantes. Pour chaque ordre, il peut y avoir zéro ou plusieurs transactions. Des exemples d'enregistrements de chaque type sont illustrés ci-après :

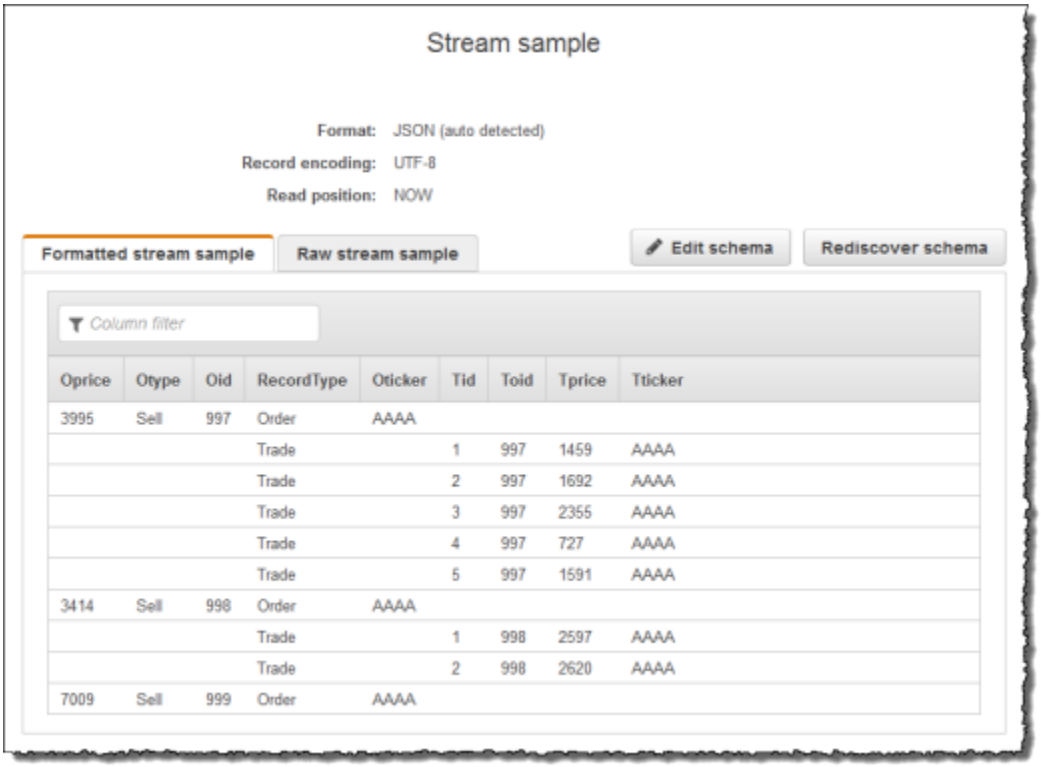
Enregistrement d'ordre

```
{"RecordType": "Order", "Oprice": 9047, "Otype": "Sell", "Oid": 3811, "Oticker":
 "AAAA"}
```

Enregistrement de transaction

```
{"RecordType": "Trade", "Tid": 1, "Toid": 3812, "Tprice": 2089, "Tticker": "BBBB"}
```

Lorsque vous créez une application à l'aide de AWS Management Console, la console affiche le schéma déduit suivant pour le flux d'entrée intégré à l'application créé. Par défaut, la console nomme ce flux intégré à l'application `SOURCE_SQL_STREAM_001`.



Oprice	Otype	Oid	RecordType	Oticker	Tid	Toid	Tprice	Tticker
3995	Sell	997	Order	AAAA				
			Trade		1	997	1459	AAAA
			Trade		2	997	1692	AAAA
			Trade		3	997	2355	AAAA
			Trade		4	997	727	AAAA
			Trade		5	997	1591	AAAA
3414	Sell	998	Order	AAAA				
			Trade		1	998	2597	AAAA
			Trade		2	998	2620	AAAA
7009	Sell	999	Order	AAAA				

Lorsque vous enregistrez la configuration, Amazon Kinesis Data Analytics lit en continu les données de la source de streaming et insère des lignes dans le flux intégré à l'application. Vous pouvez maintenant exécuter des analyses sur les données du flux intégré à l'application.

Dans le code d'application de cet exemple, vous créez d'abord deux autres flux intégrés à l'application : `Order_Stream` et `Trade_Stream`. Vous pouvez alors filtrer les lignes du flux `SOURCE_SQL_STREAM_001` en fonction du type d'enregistrement et les insérer dans les flux nouvellement créées à l'aide de pompes. Pour plus d'informations sur ce modèle de codage, consultez Code d'application.

1. Filtrer les lignes d'ordre et de transaction dans des flux intégrés à l'application distincts:

    a. Filtrez les enregistrements d'ordre dans le flux `SOURCE_SQL_STREAM_001`, puis enregistrez les ordres dans le flux `Order_Stream`.

```
--Create Order_Stream.
CREATE OR REPLACE STREAM "Order_Stream"
           (
            order_id     integer,
            order_type   varchar(10),
            ticker       varchar(4),
            order_price  DOUBLE,
            record_type  varchar(10)
            );

CREATE OR REPLACE PUMP "Order_Pump" AS
   INSERT INTO "Order_Stream"
      SELECT STREAM oid, otype,oticker, oprice, recordtype
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  recordtype = 'Order';
```

b. Filtrez les enregistrements de transaction dans le flux `SOURCE_SQL_STREAM_001`, puis enregistrez les ordres dans le flux `Trade_Stream`.

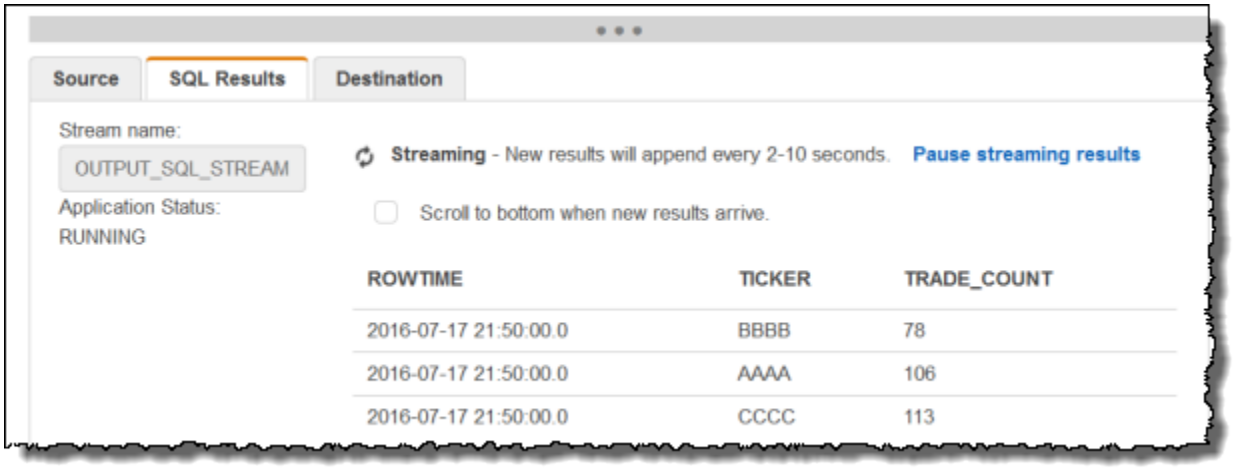
```
--Create Trade_Stream.
CREATE OR REPLACE STREAM "Trade_Stream"
           (trade_id     integer,
            order_id     integer,
            trade_price  DOUBLE,
            ticker       varchar(4),
            record_type  varchar(10)
            );

CREATE OR REPLACE PUMP "Trade_Pump" AS
   INSERT INTO "Trade_Stream"
      SELECT STREAM tid, toid, tprice, tticker, recordtype
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  recordtype = 'Trade';
```

2. Vous pouvez maintenant exécuter des analyses supplémentaires sur ces flux. Dans cet exemple, vous comptez le nombre de transactions par symbole boursier dans une fenêtre bascule d'une minute et enregistrez les résultats dans un autre flux, `DESTINATION_SQL_STREAM`.

```
--do some analytics on the Trade_Stream and Order_Stream.
-- To see results in console you must write to OPUT_SQL_STREAM.

CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
```

```
            ticker  varchar(4),
            trade_count   integer
            );

CREATE OR REPLACE PUMP "Output_Pump" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM ticker, count(*) as trade_count
      FROM   "Trade_Stream"
      GROUP BY ticker,
               FLOOR("Trade_Stream".ROWTIME TO MINUTE);
```

Vous voyez le résultat, comme illustré ci-après :



Rubriques

- Étape 1 : Préparation des données
- Étape 2 : Création de l'application

Étape suivante

Étape 1 : Préparation des données

Étape 1 : Préparation des données

Dans cette section, vous allez créer un flux de données Kinesis, puis remplir des enregistrements d'ordre et de transaction dans le flux. Il s'agit de votre source de diffusion pour l'application que vous créerez dans l'étape suivante.

Rubriques

Étape 1.1 : Création d'une source de diffusion

Vous pouvez créer un flux de données Kinesis à l'aide de la console ou de l' AWS CLI. L'exemple utilise `OrdersAndTradesStream` comme nom de flux.

- Utilisation de la console : connectez-vous à la console Kinesis AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/kinesis. Choisissez Data Streams, puis créez un flux de données avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
- À l'aide de AWS CLI— Utilisez la `create-stream` AWS CLI commande Kinesis suivante pour créer le flux :

```
$ aws kinesis create-stream \
--stream-name OrdersAndTradesStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

Étape 1.2 : Remplissage de la source de diffusion

Exécutez le script Python suivant pour remplir des exemples d'enregistrements dans `OrdersAndTradesStream`. Si vous avez créé le flux avec un autre nom, mettez à jour le code Python en conséquence.

1. Installez Python et `pip`.

   Pour plus d'informations sur l'installation de Python, consultez le site web Python.

   Vous pouvez installer des dépendances à l'aide de pip. Pour plus d'informations sur l'installation de pip, consultez Installation sur le site web de pip.
2. Exécutez le code Python suivant. La commande `put-record` dans le code écrit les enregistrements JSON dans le flux.

```
import json
```

```
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"


def get_order(order_id, ticker):
    return {
        "RecordType": "Order",
        "Oid": order_id,
        "Oticker": ticker,
        "Oprice": random.randint(500, 10000),
        "Otype": "Sell",
    }


def get_trade(order_id, trade_id, ticker):
    return {
        "RecordType": "Trade",
        "Tid": trade_id,
        "Toid": order_id,
        "Tticker": ticker,
        "Tprice": random.randint(0, 3000),
    }


def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
        ticker = random.choice(["AAAA", "BBBB", "CCCC"])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
 PartitionKey=PARTITION_KEY
        )
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
            print(trade)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(trade),
                PartitionKey=PARTITION_KEY,
```

```
            )
        order_id += 1


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape suivante

Étape 2 : Création de l'application

Étape 2 : Création de l'application

Dans cette section, vous allez créer une application Kinesis Data Analytics. Vous mettez ensuite à jour l'application en ajoutant une configuration d'entrée qui mappe la source de diffusion que vous avez créée dans la section précédente à un flux d'entrée intégré à l'application.

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Créer une application. Cet exemple utilise le nom de l'application **`ProcessMultipleRecordTypes`**.
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion) pour vous connecter à la source.
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :
   a. Choisissez le flux que vous avez créé dans Étape 1 : Préparation des données.
   b. Choisissez de créer un rôle IAM.
   c. Attendez que la console affiche le schéma déduit et les exemples d'enregistrements qui sont utilisés pour déduire le schéma pour le flux intégré à l'application créé.
   d. Choisissez Save and continue (Enregistrer et continuer).
5. Dans le hub d'applications, choisissez Go to SQL editor. Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code de l'application et vérifiez les résultats :
   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
--Create Order_Stream.
```

```
CREATE OR REPLACE STREAM "Order_Stream"
           (
            "order_id"     integer,
            "order_type"   varchar(10),
            "ticker"       varchar(4),
            "order_price"  DOUBLE,
            "record_type"  varchar(10)
            );

CREATE OR REPLACE PUMP "Order_Pump" AS
   INSERT INTO "Order_Stream"
      SELECT STREAM "Oid", "Otype","Oticker", "Oprice", "RecordType"
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  "RecordType" = 'Order';
--****************************************
--Create Trade_Stream.
CREATE OR REPLACE STREAM "Trade_Stream"
           ("trade_id"     integer,
            "order_id"     integer,
            "trade_price"  DOUBLE,
            "ticker"       varchar(4),
            "record_type"  varchar(10)
            );

CREATE OR REPLACE PUMP "Trade_Pump" AS
   INSERT INTO "Trade_Stream"
      SELECT STREAM "Tid", "Toid", "Tprice", "Tticker", "RecordType"
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  "RecordType" = 'Trade';
--************************************************************
--do some analytics on the Trade_Stream and Order_Stream.
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
            "ticker"  varchar(4),
            "trade_count"   integer
            );

CREATE OR REPLACE PUMP "Output_Pump" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM "ticker", count(*) as trade_count
      FROM   "Trade_Stream"
      GROUP BY "ticker",
                FLOOR("Trade_Stream".ROWTIME TO MINUTE);
```

b. Choisissez Save and run SQL (Enregistrer et exécuter SQL). Choisissez l'onglet Real-time analytics (Analyse en temps réel) pour voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Étape suivante

Vous pouvez configurer la sortie de l'application pour conserver les résultats vers une destination externe, telle qu'un autre flux Kinesis ou un flux de diffusion de données Firehose.

Exemples : Fenêtres et Regroupement

Cette section fournit des exemples d'applications Amazon Kinesis Data Analytics utilisant des requêtes à fenêtres et des requêtes de regroupement. (Pour plus d'informations, consultez Requêtes à fenêtres.) Chaque exemple fournit des instructions détaillées et un exemple de code pour configurer l'application Kinesis Data Analytics.

Rubriques

- Exemple : Stagger Window
- Exemple : fenêtre bascule utilisant ROWTIME
- Exemple : fenêtre bascule utilisant un horodatage d'événement
- Exemple : extraction des valeurs les plus fréquentes (TOP_K_ITEMS_TUMBLING)
- Exemple : regroupement d'une partie des résultats à partir d'une requête

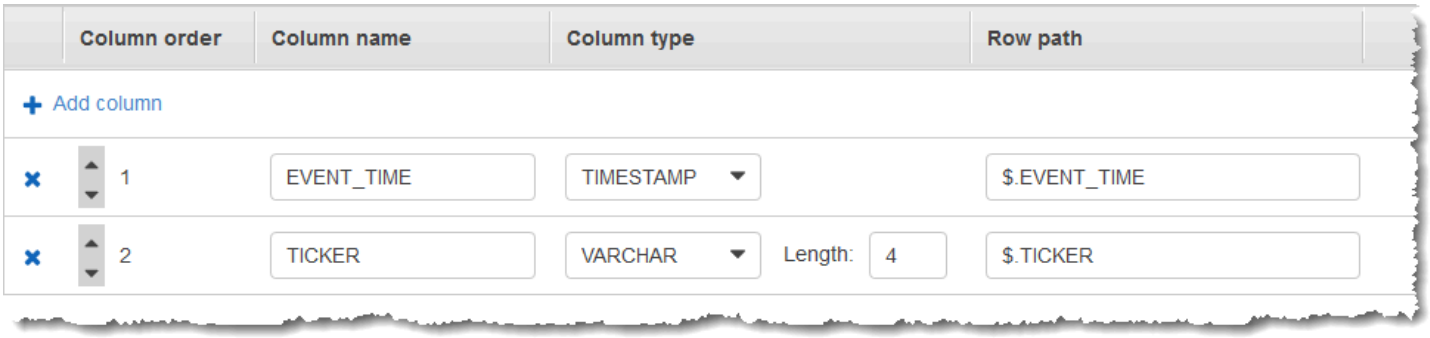
Exemple : Stagger Window

Lorsqu'une requête de fenêtre traite des fenêtres distinctes pour chaque clé de partition unique, à compter du moment où les données avec la clé correspondante arrivent, la fenêtre est appelée fenêtre stagger. Pour plus de détails, veuillez consulter Stagger Windows. Cet exemple Amazon Kinesis Data Analytics exemple utilise les colonnes EVENT_TIME et TICKER pour créer des fenêtres stagger. Le flux source contient des groupes de six enregistrements avec des valeurs EVENT_TIME et TICKER identiques qui se produisent au sein d'une période d'une minute, mais pas nécessairement avec la même valeur de minute (par exemple, `18:41:xx`).

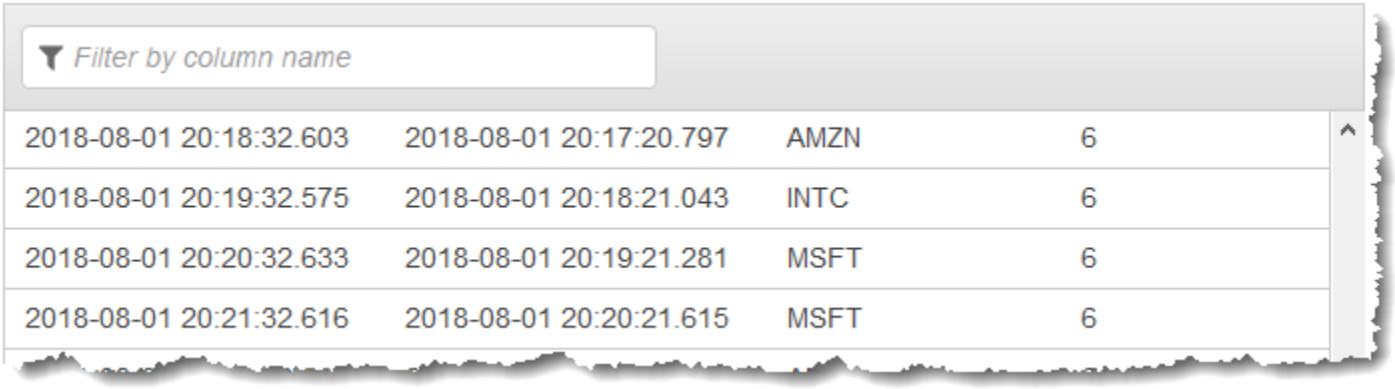
Dans cet exemple, vous écrivez les enregistrements suivants dans un flux de données Kinesis aux heures suivantes. Le script n'écrit pas les temps dans le flux, mais le temps auquel l'enregistrement est intégré par l'application est écrit dans le champ `ROWTIME` :

```
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:17:30
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:17:40
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:17:50
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:18:00
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:18:10
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:18:21
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:18:31
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:18:41
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:18:51
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:19:01
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:19:11
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:19:21
...
```

Vous créez ensuite une application Kinesis Data Analytics dans l'AWS Management Console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements sur la source de streaming et en déduit un schéma intégré à l'application avec deux colonnes (`EVENT_TIME` et `TICKER`), comme illustré ci-dessous.



Vous utilisez le code de l'application à l'aide de la fonction `COUNT` pour créer un regroupement des données avec fenêtres. Vous insérez ensuite les données obtenues dans un autre flux intégré à l'application, comme indiqué dans la capture d'écran suivante :



Dans la procédure suivante, vous créez une application Kinesis Data Analytics qui regroupe les valeurs dans le flux d'entrée dans une fenêtre stagger basée sur EVENT_TIME et TICKER.

Rubriques

- Étape 1 : Création d'un flux de données Kinesis
- Étape 2 : Création d'une application Kinesis Data Analytics

Étape 1 : Création d'un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Pour écrire des enregistrements sur un flux de données Kinesis dans un environnement de production, nous vous recommandons d'utiliser Kinesis Producer Library ou les API de flux de données Kinesis. Pour plus de simplicité, cet exemple utilise le script Python ci-dessous pour générer des enregistrements. Exécutez le code pour remplir les exemples d'enregistrements du symbole boursier. Ce code simple écrit en continu un groupe de six enregistrements avec les mêmes valeurs `EVENT_TIME` et symbole boursier aléatoires dans le flux, durant une minute. Laissez le script s'exécuter pour pouvoir générer le schéma d'application lors d'une étape ultérieure.

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
    return {
        "EVENT_TIME": event_time.isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and ticker
        for _ in range(6):
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey",
            )
            time.sleep(10)


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).

3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion) pour vous connecter à la source.
4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.
   b. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements qui sont utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit comporte deux colonnes.
   c. Choisissez Edit schema (Modifier le schéma). Remplacez le Column type (Type de colonne) de la colonne EVENT_TIME par `TIMESTAMP`.
   d. Choisissez Save schema and update stream samples (Enregistrer le schéma et mettre à jour les exemples de flux). Une fois que la console a enregistré le schéma, choisissez Exit (Quitter).
   e. Choisissez Save and continue (Enregistrer et continuer).
5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.
6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    event_time TIMESTAMP,
    ticker_symbol    VARCHAR(4),
    ticker_count     INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM
        EVENT_TIME,
        TICKER,
        COUNT(TICKER) AS ticker_count
    FROM "SOURCE_SQL_STREAM_001"
    WINDOWED BY STAGGER (
            PARTITION BY TICKER, EVENT_TIME RANGE INTERVAL '1' MINUTE);
```

   b. Choisissez Save and run SQL (Enregistrer et exécuter SQL).

Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

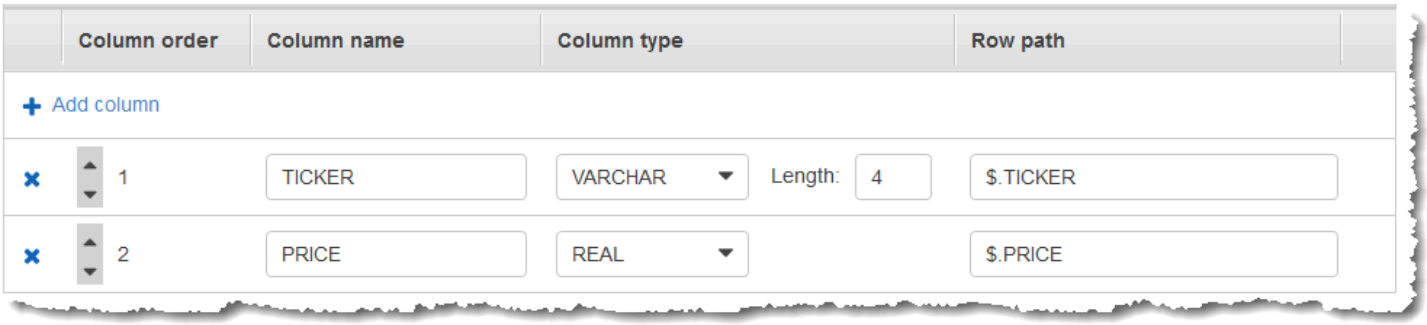
Exemple : fenêtre bascule utilisant ROWTIME

Lorsqu'une requête à fenêtres traite chaque fenêtre sans chevauchement, la fenêtre est appelée fenêtre bascule. Pour plus de détails, veuillez consulter Fenêtres bascules (regroupements à l'aide de GROUP BY). Cet exemple Amazon Kinesis Data Analytics utilise la colonne `ROWTIME` pour créer des fenêtres bascules. La colonne `ROWTIME` représente le moment où l'enregistrement a été lu par l'application.

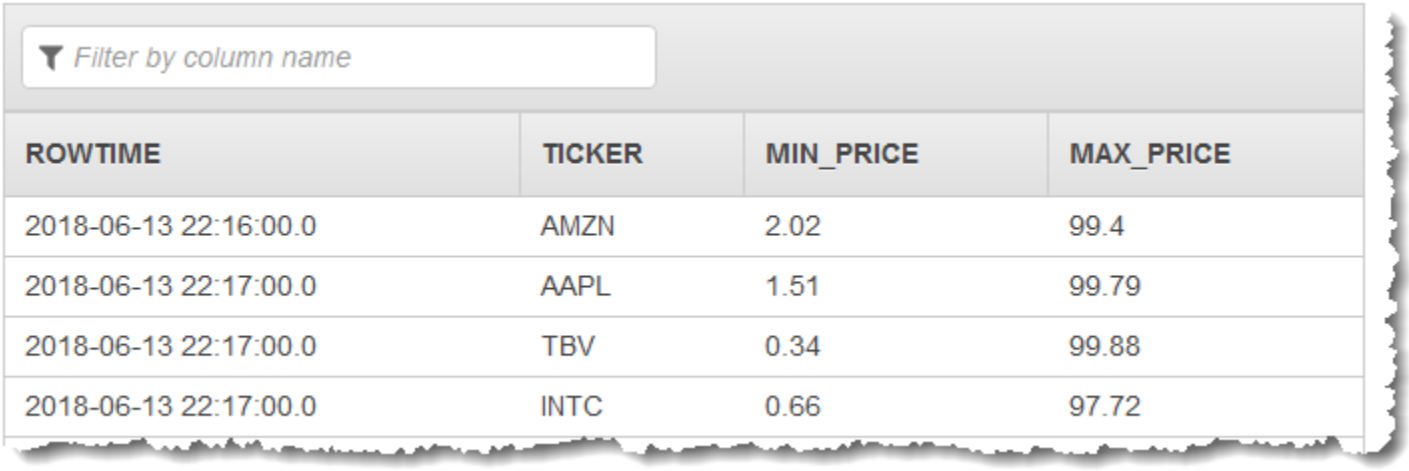
Dans cet exemple, vous écrivez les enregistrements suivants dans un flux de données Kinesis.

```
{"TICKER": "TBV", "PRICE": 33.11}
{"TICKER": "INTC", "PRICE": 62.04}
{"TICKER": "MSFT", "PRICE": 40.97}
{"TICKER": "AMZN", "PRICE": 27.9}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans l'AWS Management Console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements sur la source de streaming et en déduit un schéma intégré à l'application avec deux colonnes (`TICKER` et `PRICE`), comme illustré ci-dessous.



Vous utilisez le code de l'application à l'aide du `MIN` et des fonctions `MAX` pour créer un regroupement des données avec fenêtres. Vous insérez ensuite les données obtenues dans un autre flux intégré à l'application, comme indiqué dans la capture d'écran suivante :


Filter by column name

ROWTIME	TICKER	MIN_PRICE	MAX_PRICE
2018-06-13 22:16:00.0	AMZN	2.02	99.4
2018-06-13 22:17:00.0	AAPL	1.51	99.79
2018-06-13 22:17:00.0	TBV	0.34	99.88
2018-06-13 22:17:00.0	INTC	0.66	97.72

Dans la procédure suivante, vous créez une application Kinesis Data Analytics qui regroupe les valeurs dans le flux d'entrée dans une fenêtre bascule basée sur ROWTIME.

Rubriques

- Étape 1 : Création d'un flux de données Kinesis
- Étape 2 : Création d'une application Kinesis Data Analytics

Étape 1 : Création d'un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Pour écrire des enregistrements sur un flux de données Kinesis dans un environnement de production, nous vous recommandons d'utiliser Kinesis Client Library ou les API de flux de données Kinesis. Pour plus de simplicité, cet exemple utilise le script Python ci-dessous pour générer des enregistrements. Exécutez le code pour remplir les exemples d'enregistrements du symbole boursier. Ce code simple écrit de façon continue un enregistrement du symbole boursier aléatoire dans le flux. Laissez le script s'exécuter pour pouvoir générer le schéma d'application lors d'une étape ultérieure.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Créer une application, saisissez un nom d'application, puis sélectionnez Créer une application.
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion) pour vous connecter à la source.

4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements qui sont utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit comporte deux colonnes.

   c. Choisissez Save schema and update stream samples (Enregistrer le schéma et mettre à jour les exemples de flux). Une fois que la console a enregistré le schéma, choisissez Exit (Quitter).

   d. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.

6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (TICKER VARCHAR(4), MIN_PRICE
 REAL, MAX_PRICE REAL);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM TICKER, MIN(PRICE), MAX(PRICE)
        FROM "SOURCE_SQL_STREAM_001"
        GROUP BY TICKER,
            STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

   b. Choisissez Save and run SQL (Enregistrer et exécuter SQL).

      Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Exemple : fenêtre bascule utilisant un horodatage d'événement

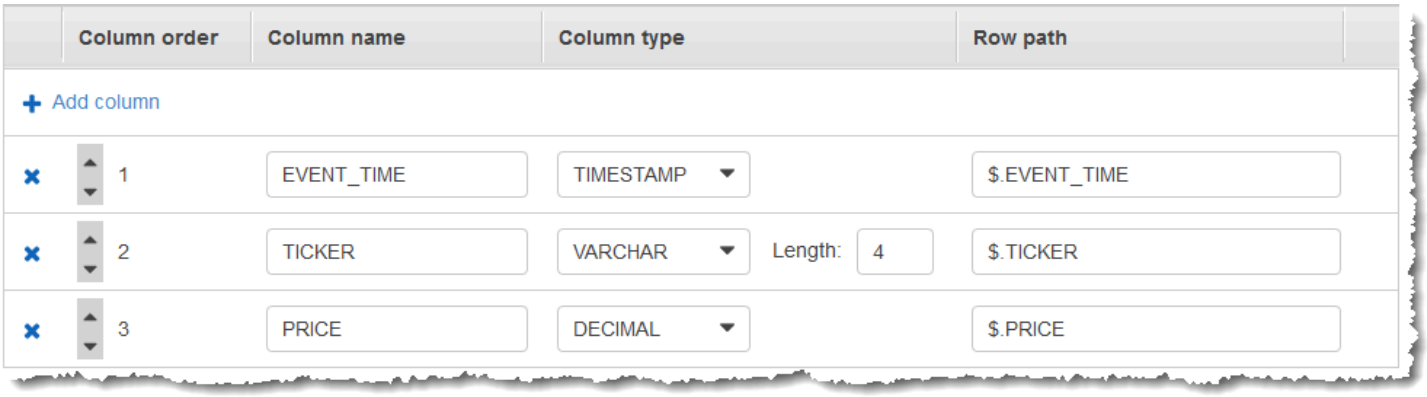
Lorsqu'une requête à fenêtres traite chaque fenêtre sans chevauchement, la fenêtre est appelée fenêtre bascule. Pour plus de détails, veuillez consulter Fenêtres bascules (regroupements à l'aide de GROUP BY). Cet exemple Amazon Kinesis Data Analytics illustre une fenêtre bascule qui utilise un horodatage d'événement, c'est-à-dire un horodatage créé par l'utilisateur et qui est compris dans

les données de streaming. Il utilise cette approche plutôt que ROWTIME, qui est un horodatage que Kinesis Data Analytics crée lorsque l'application reçoit l'enregistrement. Vous devrez utiliser un horodatage d'événement dans les données de streaming si vous souhaitez créer un regroupement basé sur le moment où un événement s'est produit, plutôt que sur le moment où il a été reçu par l'application. Dans cet exemple, la valeur `ROWTIME` déclenche le regroupement chaque minute, et les enregistrements sont regroupés à la fois par `ROWTIME` et par heure d'événement.

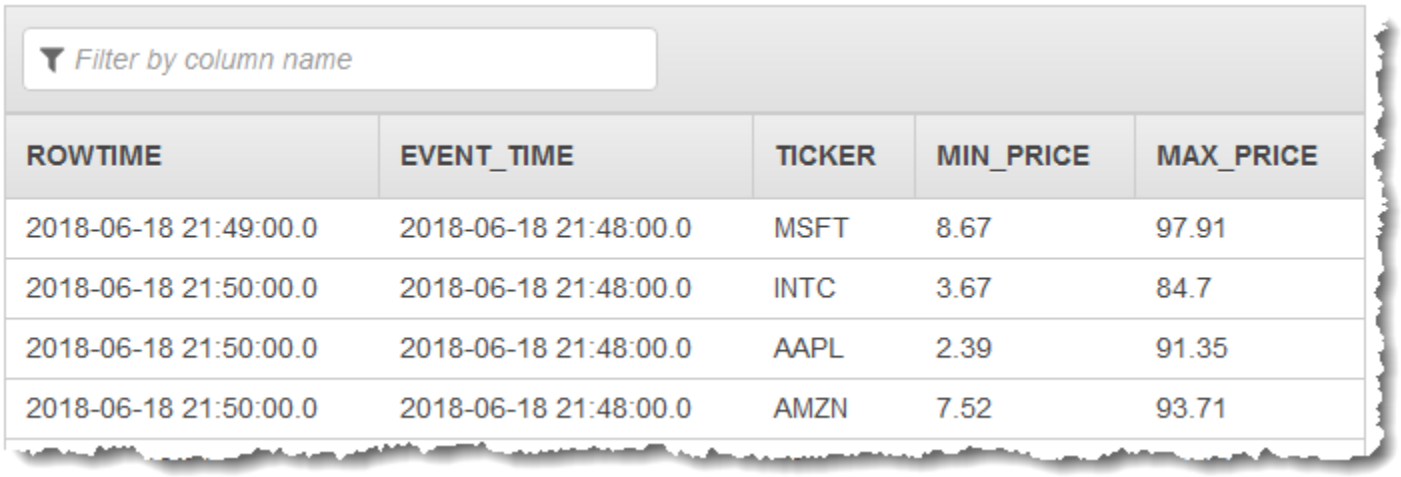
Dans cet exemple, vous écrivez les enregistrements suivants dans un flux Amazon Kinesis. La valeur `EVENT_TIME` est fixée à 5 secondes dans le passé afin de simuler un délai de transmission et de traitement susceptible de créer un retard entre le moment où l'événement s'est produit et le moment où l'enregistrement est transféré vers Kinesis Data Analytics.

```
{"EVENT_TIME": "2018-06-13T14:11:05.766191", "TICKER": "TBV", "PRICE": 43.65}
{"EVENT_TIME": "2018-06-13T14:11:05.848967", "TICKER": "AMZN", "PRICE": 35.61}
{"EVENT_TIME": "2018-06-13T14:11:05.931871", "TICKER": "MSFT", "PRICE": 73.48}
{"EVENT_TIME": "2018-06-13T14:11:06.014845", "TICKER": "AMZN", "PRICE": 18.64}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans l'AWS Management Console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements sur la source de streaming et en déduit un schéma intégré à l'application avec trois colonnes (`EVENT_TIME`, `TICKER`, et `PRICE`) comme illustré ci-dessous.



Vous utilisez le code de l'application à l'aide du `MIN` et des fonctions `MAX` pour créer un regroupement des données avec fenêtres. Vous insérez ensuite les données obtenues dans un autre flux intégré à l'application, comme indiqué dans la capture d'écran suivante :


Filter by column name

ROWTIME	EVENT_TIME	TICKER	MIN_PRICE	MAX_PRICE
2018-06-18 21:49:00.0	2018-06-18 21:48:00.0	MSFT	8.67	97.91
2018-06-18 21:50:00.0	2018-06-18 21:48:00.0	INTC	3.67	84.7
2018-06-18 21:50:00.0	2018-06-18 21:48:00.0	AAPL	2.39	91.35
2018-06-18 21:50:00.0	2018-06-18 21:48:00.0	AMZN	7.52	93.71

Dans la procédure suivante, vous créez une application Kinesis Data Analytics qui regroupe les valeurs dans le flux d'entrée dans une fenêtre bascule basée sur une heure d'événement.

Rubriques

- Étape 1 : Création d'un flux de données Kinesis
- Étape 2 : Création d'une application Kinesis Data Analytics

Étape 1 : Création d'un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Pour écrire des enregistrements sur un flux de données Kinesis dans un environnement de production, nous vous recommandons d'utiliser Kinesis Client Library ou les API de flux de données Kinesis. Pour plus de simplicité, cet exemple utilise le script Python ci-dessous pour générer des enregistrements. Exécutez le code pour remplir les exemples d'enregistrements du symbole boursier. Ce code simple écrit de façon continue un enregistrement du symbole boursier aléatoire dans le flux. Laissez le script s'exécuter pour pouvoir générer le schéma d'application lors d'une étape ultérieure.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Créer une application, saisissez un nom d'application, puis sélectionnez Créer une application.
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion) pour vous connecter à la source.

4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements qui sont utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit comporte trois colonnes.

   c. Choisissez Edit schema (Modifier le schéma). Remplacez le Column type (Type de colonne) de la colonne EVENT_TIME par `TIMESTAMP`.

   d. Choisissez Save schema and update stream samples (Enregistrer le schéma et mettre à jour les exemples de flux). Une fois que la console a enregistré le schéma, choisissez Exit (Quitter).

   e. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.

6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (EVENT_TIME timestamp, TICKER
 VARCHAR(4), min_price REAL, max_price REAL);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM STEP("SOURCE_SQL_STREAM_001".EVENT_TIME BY INTERVAL '60'
 SECOND),
        TICKER,
         MIN(PRICE) AS MIN_PRICE,
         MAX(PRICE) AS MAX_PRICE
    FROM    "SOURCE_SQL_STREAM_001"
    GROUP BY TICKER,
             STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND),
             STEP("SOURCE_SQL_STREAM_001".EVENT_TIME BY INTERVAL '60' SECOND);
```

   b. Choisissez Save and run SQL (Enregistrer et exécuter SQL).

      Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Exemple : extraction des valeurs les plus fréquentes (TOP_K_ITEMS_TUMBLING)

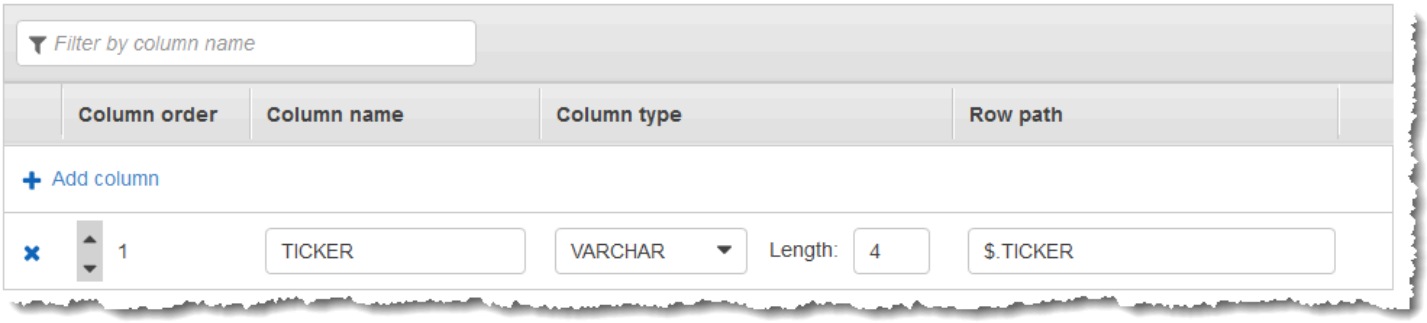
Cet exemple Amazon Kinesis Data Analytics montre comment utiliser la fonction `TOP_K_ITEMS_TUMBLING` pour extraire les valeurs les plus fréquentes dans une fenêtre bascule. Pour plus d'informations, consultez la section Fonction `TOP_K_ITEMS_TUMBLING` dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

La fonction `TOP_K_ITEMS_TUMBLING` est utile pour regrouper des dizaines ou des centaines de milliers de clés et si vous souhaitez réduire l'utilisation de vos ressources. La fonction produit le même résultat qu'un regroupement avec les clauses `GROUP BY` et `ORDER BY` .

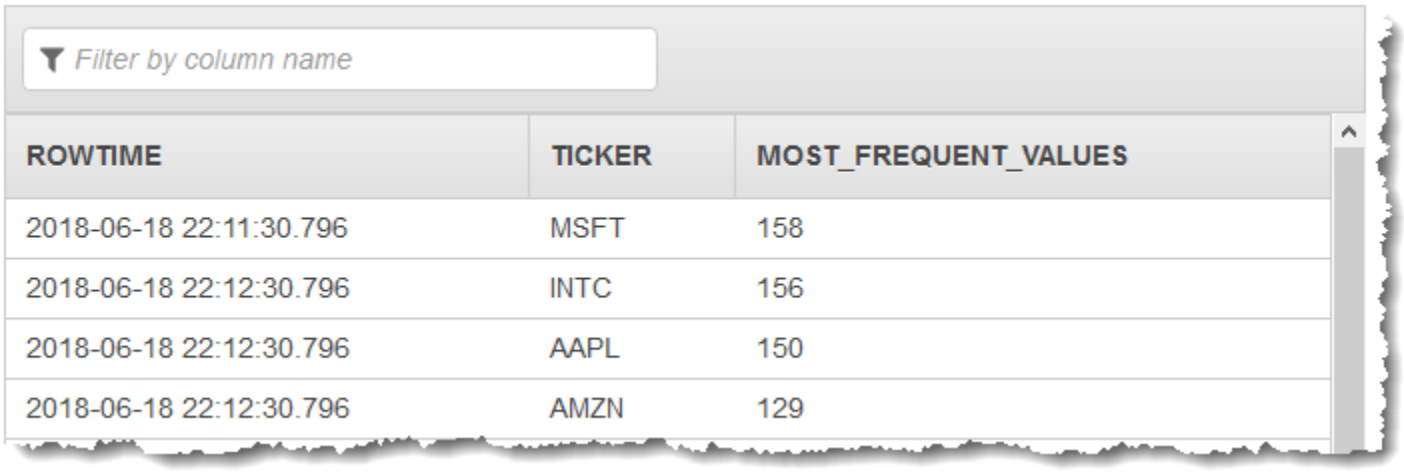
Dans cet exemple, vous écrivez les enregistrements suivants dans un flux Amazon Kinesis :

```
{"TICKER": "TBV"}
{"TICKER": "INTC"}
{"TICKER": "MSFT"}
{"TICKER": "AMZN"}
...
```

Vous créez ensuite une application Kinesis Data Analytics dans l'AWS Management Console, avec le flux de données Kinesis comme source de streaming. Le processus de découverte lit les exemples d'enregistrements de la source de streaming et en déduit un schéma intégré à l'application avec une colonne (`TICKER`), comme illustré ci-dessous.



Vous utilisez le code de l'application à l'aide de la fonction `TOP_K_VALUES_TUMBLING` pour créer un regroupement des données avec fenêtres. Vous insérez ensuite les données obtenues dans un autre flux intégré à l'application, comme indiqué dans la capture d'écran suivante :



Dans la procédure suivante, vous créez une application Kinesis Data Analytics qui extrait les valeurs les plus fréquentes dans le flux d'entrée.

Rubriques

- Étape 1 : Création d'un flux de données Kinesis
- Étape 2 : Création d'une application Kinesis Data Analytics

Étape 1 : Création d'un flux de données Kinesis

Créez un flux de données Amazon Kinesis et remplissez les enregistrements comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une seule partition. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
4. Pour écrire des enregistrements sur un flux de données Kinesis dans un environnement de production, nous vous recommandons d'utiliser Kinesis Client Library ou les API de flux de données Kinesis. Pour plus de simplicité, cet exemple utilise le script Python ci-dessous pour générer des enregistrements. Exécutez le code pour remplir les exemples d'enregistrements du symbole boursier. Ce code simple écrit de façon continue un enregistrement du symbole boursier aléatoire dans le flux. Laissez le script s'exécuter pour pouvoir générer le schéma d'application lors d'une étape ultérieure.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape 2 : Création d'une application Kinesis Data Analytics

Créez une application Kinesis Data Analytics comme suit :

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Choisissez Create application (Créer une application), saisissez un nom d'application, puis sélectionnez Create application (Créer une application).
3. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion) pour vous connecter à la source.

4. Sur la page Connect to source (Se connecter à la source), procédez comme suit :

   a. Choisissez le flux que vous avez créé dans la section précédente.

   b. Choisissez Discover schema (Découvrir le schéma). Attendez que la console affiche le schéma déduit et les exemples d'enregistrements qui sont utilisés pour déduire le schéma pour le flux intégré à l'application créé. Le schéma déduit comporte une colonne.

   c. Choisissez Save schema and update stream samples (Enregistrer le schéma et mettre à jour les exemples de flux). Une fois que la console a enregistré le schéma, choisissez Exit (Quitter).

   d. Choisissez Save and continue (Enregistrer et continuer).

5. Sur la page de détails de l'application, choisissez Go to SQL editor (Accéder à l'éditeur SQL). Pour lancer l'application, choisissez Yes, start application (Oui, démarrer l'application) dans la boîte de dialogue qui s'affiche.

6. Dans l'éditeur SQL, écrivez le code d'application et vérifiez les résultats comme suit :

   a. Copiez le code d'application suivant et collez-le dans l'éditeur:

```
CREATE OR REPLACE STREAM DESTINATION_SQL_STREAM (
  "TICKER" VARCHAR(4),
  "MOST_FREQUENT_VALUES" BIGINT
);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM *
        FROM TABLE (TOP_K_ITEMS_TUMBLING(
            CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001"),
            'TICKER',         -- name of column in single quotes
            5,                      -- number of the most frequently occurring
 values
            60                      -- tumbling window size in seconds
            )
        );
```

   b. Choisissez Save and run SQL (Enregistrer et exécuter SQL).

      Dans l'onglet Real-time analytics (Analyse en temps réel), vous pouvez voir tous les flux intégrés à l'application que l'application a créés et vérifier les données.

Exemple : regroupement d'une partie des résultats à partir d'une requête

Si un flux de données Amazon Kinesis contient des enregistrements qui disposent d'une heure d'événement ne correspondant pas exactement à l'heure de l'intégration, une sélection de résultats dans une fenêtre bascule contiendra, dans la fenêtre, les enregistrements qui sont arrivés mais qui ne se sont pas nécessairement produits. Dans ce cas, la fenêtre bascule contient uniquement une partie des résultats que vous souhaitez. Vous pouvez utiliser plusieurs approches pour résoudre ce problème :

- Utilisez uniquement une fenêtre bascule et regroupez une partie des résultats dans le traitement ultérieur via une base de données ou un entrepôt de données utilisant des opérations de « mises à jour/insertions ». Cette approche est efficace dans le traitement d'une application. Elle gère les données tardives indéfiniment pour les opérateurs d'agrégation (`sum`, `min`, `max`, etc.). L'inconvénient de cette approche est que vous devez développer et maintenir une logique d'application supplémentaire dans la couche de base de données.
- Utilisez une fenêtre bascule et défilante qui produit une partie des résultats tôt, mais continue également à produire des résultats complets au cours de la période de la fenêtre défilante. Cette approche gère les données tardives grâce à un remplacement plutôt qu'une « mise à jour/insertion » afin qu'aucune autre logique d'application n'ait besoin d'être ajoutée dans la couche de base de données. L'inconvénient de cette approche est qu'elle utilise plus d'unités de traitement Kinesis (KPU) et produit toujours deux résultats, ce qui peut ne pas fonctionner dans certains cas d'utilisation.

Pour plus d'informations sur les fenêtres bascules et défilantes, consultez Requêtes à fenêtres.

Dans la procédure suivante, le regroupement de la fenêtre bascule génère deux résultats partiels (envoyés au `CALC_COUNT_SQL_STREAM` dans le flux intégré à l'application) qui doivent être combinés pour produire un résultat final. L'application génère ensuite un second regroupement (envoyé au `DESTINATION_SQL_STREAM` dans le flux intégré à l'application) qui combine les deux résultats partiels.

Pour créer une application qui regroupe une partie des résultats à l'aide d'une heure d'évènement

1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.

2. Choisissez Data Analytics (Analyse des données) dans le volet de navigation. Créez une application Kinesis Data Analytics comme décrit dans le didacticiel Mise en route avec les applications Amazon Kinesis Data Analytics pour SQL.

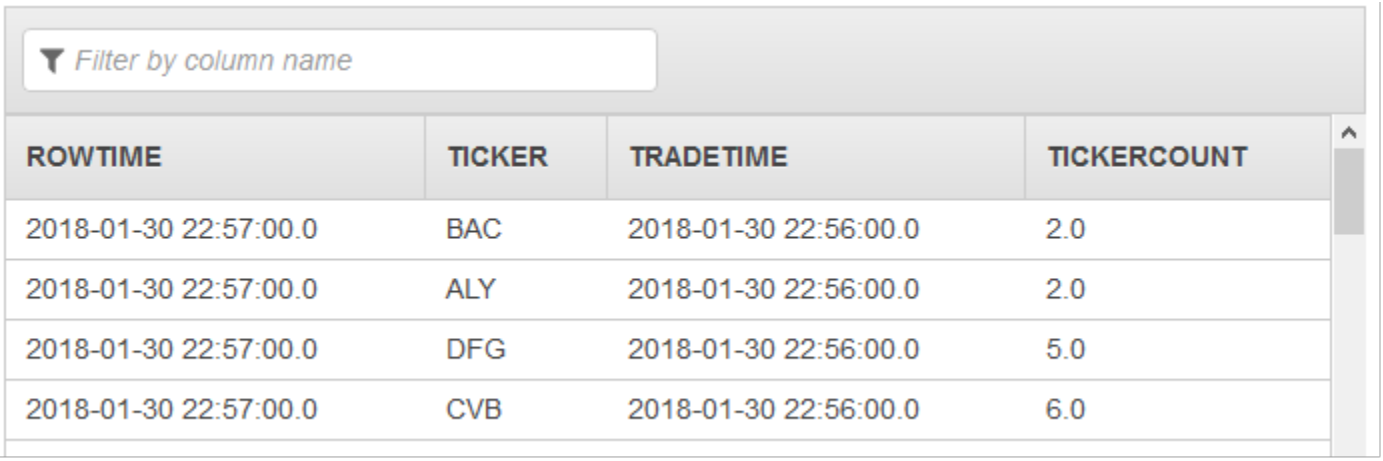
3. Dans l'éditeur SQL, remplacez le code d'application par les éléments suivants :

```
CREATE OR REPLACE STREAM "CALC_COUNT_SQL_STREAM"
    (TICKER      VARCHAR(4),
    TRADETIME   TIMESTAMP,
    TICKERCOUNT       DOUBLE);

CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM"
    (TICKER      VARCHAR(4),
    TRADETIME   TIMESTAMP,
    TICKERCOUNT       DOUBLE);

CREATE PUMP "CALC_COUNT_SQL_PUMP_001" AS
    INSERT INTO "CALC_COUNT_SQL_STREAM" ("TICKER","TRADETIME", "TICKERCOUNT")
    SELECT STREAM
        "TICKER_SYMBOL",
        STEP("SOURCE_SQL_STREAM_001"."ROWTIME" BY INTERVAL '1' MINUTE) as
 "TradeTime",
        COUNT(*) AS "TickerCount"
    FROM "SOURCE_SQL_STREAM_001"
    GROUP BY
        STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '1' MINUTE),
        STEP("SOURCE_SQL_STREAM_001"."APPROXIMATE_ARRIVAL_TIME" BY INTERVAL '1'
 MINUTE),
        TICKER_SYMBOL;

CREATE PUMP "AGGREGATED_SQL_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM" ("TICKER","TRADETIME", "TICKERCOUNT")
    SELECT STREAM
        "TICKER",
        "TRADETIME",
        SUM("TICKERCOUNT") OVER W1 AS "TICKERCOUNT"
    FROM "CALC_COUNT_SQL_STREAM"
    WINDOW W1 AS (PARTITION BY "TRADETIME" RANGE INTERVAL '10' MINUTE PRECEDING);
```

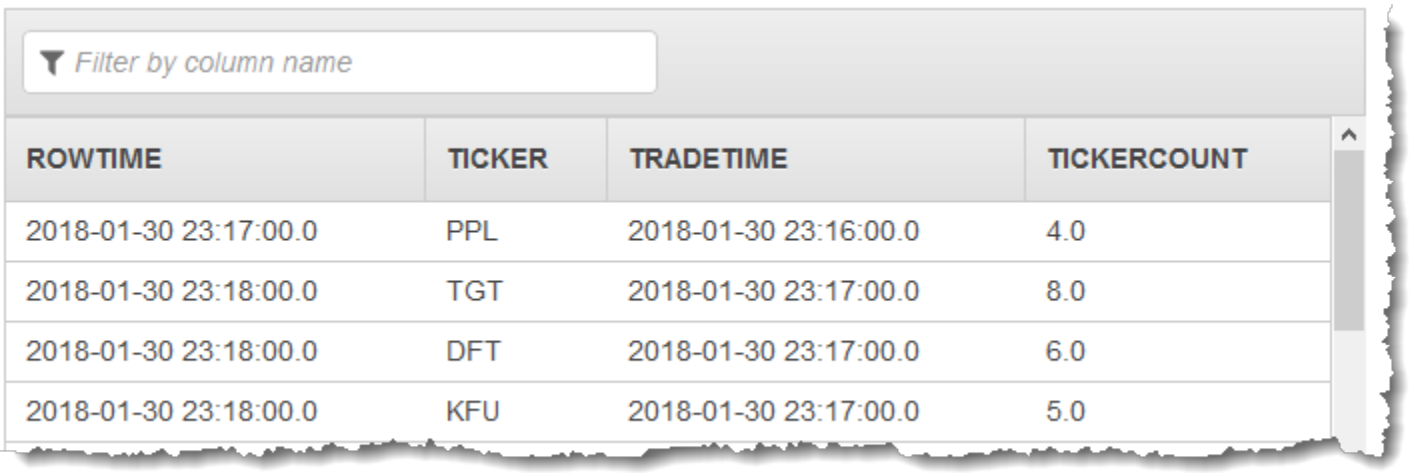
L'instruction SELECT du code d'application filtre les lignes de SOURCE_SQL_STREAM_001 pour obtenir les changements de cours d'action supérieurs à 1 % et insère ces lignes dans un autre flux intégré à l'application, CHANGE_STREAM, à l'aide d'une pompe.

4. Choisissez Save and run SQL (Enregistrer et exécuter SQL).

La première pompe génère un flux vers `CALC_COUNT_SQL_STREAM` similaire à ce qui suit. Notez que l'ensemble des résultats est incomplet :



La deuxième pompe génère ensuite un flux vers `DESTINATION_SQL_STREAM` contenant l'ensemble des résultats :



Exemples : Jointures

Cette section fournit des exemples d'applications Kinesis Data Analytics qui utilisent des requêtes de jointure. Chaque exemple fournit des instructions détaillées pour configurer et tester votre application Kinesis Data Analytics.

Rubriques

- Exemple : ajout de données de référence à une application Kinesis Data Analytics

Exemple : ajout de données de référence à une application Kinesis Data Analytics

Dans cet exercice, vous allez ajouter des données de référence à une application Kinesis Data Analytics existante. Pour plus d'informations sur les données de référence, consultez le rubrique suivante :

- Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement
- Configuration de l'entrée de l'application

Dans cet exercice, vous allez ajouter des données de référence à l'application que vous avez créée dans l'exercice de mise en route de Kinesis Data Analytics. Les données de référence fournissent le nom de la société pour chaque symbole boursier, par exemple :

```
Ticker, Company
AMZN,Amazon
ASD, SomeCompanyA
MMB, SomeCompanyB
WAS,  SomeCompanyC
```

Suivez d'abord les étapes de l'exercice de mise en route pour créer une application de démarrage. Puis suivez ces étapes pour configurer et ajouter des données de référence à votre application :

1. Préparation des données
   - Stockez les données de référence précédentes sous forme d'objet dans Amazon Simple Storage Service (Amazon S3).
   - Créez un rôle IAM pouvant être assumé par Kinesis Data Analytics pour lire l'objet Amazon S3 en votre nom.
2. Ajoutez la source des données de référence à votre application.

   Kinesis Data Analytics lit l'objet Amazon S3 et crée une table de référence intégrée à l'application que vous pouvez interroger dans votre code d'application.
3. Testez le code.

   Dans votre code d'application, vous allez écrire une requête de jointure pour joindre le flux intégré à l'application à la table de référence intégrée à l'application afin d'obtenir le nom de la société pour chaque symbole boursier.

Rubriques

Étape 1 : Préparation

Dans cette section, vous stockez des exemples de données de référence en tant qu'objet dans un compartiment Amazon S3. Vous créez également un rôle IAM pouvant être assumé par Kinesis Data Analytics pour lire l'objet en votre nom.

Stockage des données de référence en tant qu'objet Amazon S3

Dans cette étape, vous stockez les exemples de données de référence en tant qu'objet Amazon S3.

1. Ouvrez un éditeur de texte, ajoutez les données suivantes et enregistrez le fichier sous le nom `TickerReference.csv`.

   ```
   Ticker, Company
   AMZN,Amazon
   ASD, SomeCompanyA
   MMB, SomeCompanyB
   WAS,  SomeCompanyC
   ```

2. Chargez le fichier `TickerReference.csv` dans votre compartiment S3. Pour en savoir plus, consultez Téléchargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

Créer un rôle IAM

Créez ensuite un rôle IAM que Kinesis Data Analytics peut assumer et lisez l'objet Amazon S3.

1. Dans AWS Identity and Access Management (IAM), créez un rôle IAM nommé **`KinesisAnalytics-ReadS3Object`**. Pour créer le rôle, suivez les instructions de Création d'un rôle pour déléguer des autorisations à un service Amazon (AWS Management Console) dans le Guide de l'utilisateur IAM.

   Dans la console IAM, spécifiez les valeurs suivantes :

- Dans Sélectionner le type de rôle, choisissez AWS Lambda. Après avoir créé le rôle, vous allez modifier la stratégie d'approbation pour autoriser Kinesis Data Analytics (pas AWS Lambda) à assumer le rôle.
- N'attachez pas de stratégie sur la page Attach Policy.

2. Mettez à jour les stratégies de rôle IAM :

   a. Dans la console IAM, choisissez le rôle que vous avez créé.

   b. Dans l'onglet Relations d'approbation, mettez à jour la stratégie d'approbation pour accorder à Kinesis Data Analytics des autorisations pour assumer le rôle. La stratégie d'approbation est présentée ci-après :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   c. Dans l'onglet Autorisations, attachez une stratégie gérée par Amazon appelée AmazonS3ReadOnlyAccess. Vous accordez ainsi au rôle les autorisations pour lire un objet Amazon S3. Cette stratégie est présentée ci-après :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

Étape 2 : Ajout de la source de données de référence à la configuration d'application

Dans cette étape, vous ajoutez une source de données de référence à la configuration de votre application. Pour commencer, vous avez besoin des informations suivantes :

- Le nom de votre compartiment S3 et le nom de la clé d'objet
- L'Amazon Resource Name (ARN) du rôle IAM

1. Dans la page principale de l'application, choisissez Connect reference data (Connecter des données de référence).
2. Sur la page Connecter une source de données de référence, choisissez le compartiment Amazon S3 contenant vos objets de données de référence, puis saisissez le nom de clé de l'objet.
3. Saisissez **`CompanyName`** comme nom du tableau de référence intégré à l'application.
4. Dans la section Access to chosen resources (Accéder aux ressources sélectionnées), choisissez Choose from IAM roles that Kinesis Analytics can assume (Choisir depuis des rôles IAM que les analyses Kinesis peuvent prendre en charge) et sélectionnez le rôle IAM KinesisAnalytics-ReadS3Object que vous avez créé dans la section précédente.
5. Choisissez Discover schema (Découvrir le schéma). La console détecte deux colonnes dans les données de référence.
6. Choisissez Enregistrer et fermer.

Étape 3 : Test : Interrogation de la table de référence intégrée à l'application

Vous pouvez maintenant interroger la table de référence intégrée à l'application, `CompanyName`. Vous pouvez utiliser les informations de référence pour enrichir votre application en joignant les données de prix des actions à la table de référence. Le résultat indique le nom de l'entreprise.

1. Remplacez le code de votre application par ce qui suit. La requête joint le flux d'entrée intégré à l'application à la table de référence intégrée à l'application. Le code d'application écrit les résultats dans un autre flux intégré à l'application, `DESTINATION_SQL_STREAM`.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ticker_symbol VARCHAR(4),
 "Company" varchar(20), sector VARCHAR(12), change DOUBLE, price DOUBLE);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
   SELECT STREAM ticker_symbol, "c"."Company", sector, change, price
   FROM "SOURCE_SQL_STREAM_001" LEFT JOIN "CompanyName" as "c"
   ON "SOURCE_SQL_STREAM_001".ticker_symbol = "c"."Ticker";
```

2. Vérifiez que la sortie de l'application s'affiche dans l'onglet SQLResults. Assurez-vous que certaines des lignes affichent des noms de société (vos exemples de données de référence n'ont pas tous les noms de société).

Exemples : Apprentissage automatique (Machine Learning)

Cette section fournit des exemples d'applications Amazon Kinesis Data Analytics qui utilisent des requêtes d'apprentissage automatique (Machine Learning). Les requêtes d'apprentissage automatique (Machine Learning) effectuent des analyses complexes sur les données en s'appuyant sur l'historique des données du flux pour rechercher des modèles inhabituels. Les exemples fournissent des instructions détaillées pour configurer et tester votre application Kinesis Data Analytics.

Rubriques

- Exemple : Détection d'anomalies de données sur un flux (fonction RANDOM_CUT_FOREST)
- Exemple : détection d'anomalies de données et message d'explication (fonction RANDOM_CUT_FOREST_WITH_EXPLANATION)
- Exemple : Détection des points chauds sur un flux (fonction HOTSPOTS)

Exemple : Détection d'anomalies de données sur un flux (fonction RANDOM_CUT_FOREST)

Amazon Kinesis Data Analytics fournit une fonction (`RANDOM_CUT_FOREST`) qui peut attribuer un score d'anomalie à chaque enregistrement en fonction de valeurs dans les colonnes numériques. Pour plus d'informations, consultez la section Fonction `RANDOM_CUT_FOREST` dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

Dans cet exercice, vous allez écrire du code d'application pour attribuer un score d'anomalie à des enregistrements sur la source de diffusion de votre application. Pour configurer l'application, procédez comme suit :

1. Configurer une source de streaming : vous configurez un flux de données Kinesis et écrivez des échantillons de données `heartRate`, comme illustré ci-après :

   ```
   {"heartRate": 60, "rateType":"NORMAL"}
   ...
   {"heartRate": 180, "rateType":"HIGH"}
   ```

   La procédure fournit un script Python qui vous permet de remplir le flux. Les valeurs `heartRate` sont générées de façon aléatoire, avec 99 % des enregistrements ayant des valeurs `heartRate` comprises entre 60 et 100, et seulement 1 % ayant des valeurs `heartRate` entre 150 et 200. Les enregistrements avec des valeurs `heartRate` entre 150 et 200 sont donc des anomalies.
2. Configurer l'entrée : à l'aide de la console, vous créez une application Kinesis Data Analytics et configurez l'entrée de l'application en mappant la source de streaming à un flux intégré à l'application (`SOURCE_SQL_STREAM_001`). Lorsque l'application démarre, Kinesis Data Analytics lit en continu la source de streaming et insère des enregistrements dans le flux intégré à l'application.
3. Spécifiez le code d'application – L'exemple utilise le code d'application suivant :

   ```
   --Creates a temporary stream.
   CREATE OR REPLACE STREAM "TEMP_STREAM" (
            "heartRate"        INTEGER,
            "rateType"         varchar(20),
            "ANOMALY_SCORE"    DOUBLE);

   --Creates another stream for application output.
   CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
            "heartRate"        INTEGER,
            "rateType"         varchar(20),
            "ANOMALY_SCORE"    DOUBLE);

   -- Compute an anomaly score for each record in the input stream
   -- using Random Cut Forest
   CREATE OR REPLACE PUMP "STREAM_PUMP" AS
      INSERT INTO "TEMP_STREAM"
         SELECT STREAM "heartRate", "rateType", ANOMALY_SCORE
         FROM TABLE(RANDOM_CUT_FOREST(
   ```

```
                CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001")));

-- Sort records by descending anomaly score, insert into output stream
CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM * FROM "TEMP_STREAM"
      ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE DESC;
```

Le code lit les lignes dans le flux `SOURCE_SQL_STREAM_001`, attribue un score d'anomalie et écrit les lignes résultantes dans une autre flux intégré à l'application (`TEMP_STREAM`). Le code d'application trie ensuite les enregistrements dans le flux `TEMP_STREAM` et enregistre les résultats dans un autre flux intégré à l'application (`DESTINATION_SQL_STREAM`). Vous utilisez des pompes pour insérer des lignes dans les flux intégrés à l'application. Pour de plus amples informations, veuillez consulter Flux et pompes intégrés à l'application.

4. Configurer la sortie : vous configurez la sortie de l'application pour conserver les données du flux `DESTINATION_SQL_STREAM` dans une destination externe qui est un autre flux de données Kinesis. Les opérations visant à examiner les scores d'anomalie qui sont attribués à chaque enregistrement et à déterminer quel score indique une anomalie (et que vous avez besoin d'être alerté) sont externes à l'application. Vous pouvez utiliser une fonction AWS Lambda pour traiter ces scores d'anomalie et configurer des alertes.

L'exercice utilise la région USA Est (Virginie du Nord) (`us-east-1`) pour créer ces flux et votre application. Si vous utilisez une autre région, vous devez mettre à jour le code en conséquence.

Rubriques

- Étape 1 : Préparation
- Étape 2 : Créer une application
- Étape 3 : Configuration de la sortie de l'application
- Étape 4 : Vérification de la sortie

Étape suivante

Étape 1 : Préparation

Étape 1 : Préparation

Avant de créer une application d'analyse de données Amazon Kinesis Data Analytics pour cet exercice, vous devez créer deux flux de données Kinesis. Configurez l'un des flux en tant que source de streaming pour votre application et l'autre flux en tant que destination où Kinesis Data Analytics conserve la sortie de votre application.

Rubriques

Étape 1.1 : Création des flux de données d'entrée et de sortie

Dans cette section, vous créez deux flux Kinesis : `ExampleInputStream` et `ExampleOutputStream`. Vous pouvez créer ces flux avec la AWS Management Console ou l'AWS CLI.

- Pour utiliser la console

  1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
  2. Choisissez Create data stream (Créer un flux de données). Créez un flux avec une partition nommée `ExampleInputStream`. Pour de plus amples informations, consultez Créer un flux dans le Guide du développeur Amazon Kinesis Data Streams.
  3. Répétez l'étape précédente, en créant un flux avec une seule partition nommée `ExampleOutputStream`.

- Pour utiliser l'AWS CLI

  1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

2. Exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`. Cette commande crée le deuxième flux que l'application utilisera pour écrire la sortie.

Étape 1.2 : Ecriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette étape, vous exécutez du code Python pour générer en continu des exemples d'enregistrements et les écrire dans le flux `ExampleInputStream`.

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

1. Installez Python et `pip`.

   Pour plus d'informations sur l'installation de Python, consultez le site web Python.

   Vous pouvez installer des dépendances à l'aide de pip. Pour plus d'informations sur l'installation de pip, consultez Installation sur le site web de pip.

2. Exécutez le code Python suivant. La commande `put-record` dans le code écrit les enregistrements JSON dans le flux.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class RateType(Enum):
    normal = "NORMAL"
    high = "HIGH"


def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
```

```
        raise TypeError
    return {"heartRate": rate, "rateType": rate_type.value}


def generate(stream_name, kinesis_client, output=True):
    while True:
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape suivante

Étape 2 : Créer une application

Étape 2 : Créer une application

Dans cette section, vous allez créer une application Amazon Kinesis Data Analytics comme suit :

- Configurez l'entrée de l'application pour utiliser le flux de données Kinesis que vous avez créé dans the section called "Étape 1 : Préparation" comme source de streaming.
- Utilisez le modèle Anomaly Detection (Détection des anomalies) dans la console.

Pour créer une application

1. Suivez les étapes 1, 2 et 3 de l'exercice Kinesis Data Analytics Mise en route (voir Étape 3.1 : Créer une application).
    - Dans la configuration de la source, procédez de la façon suivante :
        - Spécifiez la source de diffusion que vous avez créée dans la section précédente.

- Une fois que la console a déduit le schéma, modifiez celui-ci et définissez le type de colonne `heartRate` sur `INTEGER`.

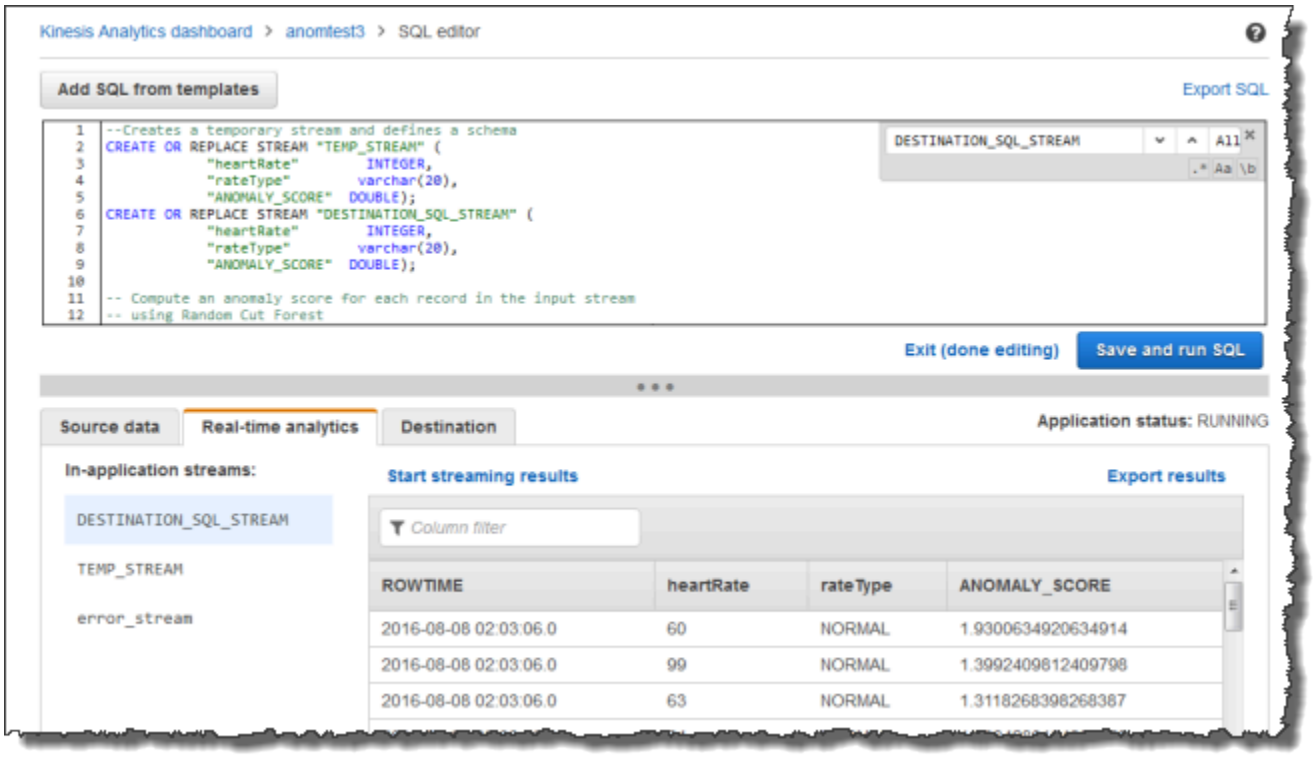
  La plupart des valeurs de taux de cœur sont normales et le processus de découverte va très probablement attribuer le type `TINYINT` à cette colonne. Mais un faible pourcentage de valeurs présente un taux de cœur élevé. Si ces valeurs élevées ne correspondent au type `TINYINT`, Kinesis Data Analytics envoie ces lignes vers un flux d'erreurs. Mettez à jour le type de données en `INTEGER` pour qu'il puisse recevoir toutes les données de taux de cœur générées.

- Utilisez le modèle Anomaly Detection (Détection des anomalies) dans la console. Mettez à jour ensuite le code du modèle pour fournir le nom de colonne approprié.

2. Mettez à jour le code d'application en fournissant des noms de colonne. Le code d'application résultant apparaît ci-dessous (collez ce code dans l'éditeur SQL) :

```
--Creates a temporary stream.
CREATE OR REPLACE STREAM "TEMP_STREAM" (
         "heartRate"        INTEGER,
         "rateType"         varchar(20),
         "ANOMALY_SCORE"    DOUBLE);

--Creates another stream for application output.
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
         "heartRate"        INTEGER,
         "rateType"         varchar(20),
         "ANOMALY_SCORE"    DOUBLE);

-- Compute an anomaly score for each record in the input stream
-- using Random Cut Forest
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "TEMP_STREAM"
      SELECT STREAM "heartRate", "rateType", ANOMALY_SCORE
      FROM TABLE(RANDOM_CUT_FOREST(
              CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001")));

-- Sort records by descending anomaly score, insert into output stream
CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM * FROM "TEMP_STREAM"
      ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE DESC;
```

3. Exécutez le code SQL et vérifiez les résultats dans la console Kinesis Data Analytics :



Étape suivante

[Étape 3 : Configuration de la sortie de l'application]

Étape 3 : Configuration de la sortie de l'application

Une fois que the section called “Étape 2 : Créer une application” est terminé, le code d'application lit des données de taux de cœur à partir d'une source de diffusion et attribue un score d'anomalie à chacune d'entre elles.

Vous pouvez maintenant envoyer les résultats de l’application depuis le flux intégré à l’application vers une destination externe, qui est un autre flux de données Kinesis (`OutputStreamTestingAnomalyScores`). Vous pouvez ensuite analyser les scores d'anomalie et déterminer quel taux de cœur est anormal. Vous pouvez ensuite étendre encore cette application pour générer des alertes.

Suivez les étapes ci-dessous pour configurer la sortie de l'application :

1. Ouvrez la console Amazon Kinesis Data Analytics. Dans l'éditeur SQL, choisissez Destination ou Add a destination dans le tableau de bord d'application.
2. Sur la page Connect to destination (Se connecter à une destination), choisissez le flux `OutputStreamTestingAnomalyScores` que vous avez créé dans la section précédente.

   Maintenant, vous disposez d'une destination externe où Amazon Kinesis Data Analytics conserve tous les enregistrements que votre application écrit dans le flux intégré à l'application `DESTINATION_SQL_STREAM`.
3. Vous pouvez configurer AWS Lambda le cas échéant pour surveiller le flux `OutputStreamTestingAnomalyScores` et vous envoyer des alertes. Pour obtenir des instructions, consultez Prétraitement des données à l'aide d'une fonction Lambda. Si vous ne définissez pas d'alertes, vous pouvez vérifier les enregistrements écrits par Kinesis Data Analytics dans la destination externe, ce qui représente le flux de données Kinesis `OutputStreamTestingAnomalyScores`, comme décrit dans Étape 4 : Vérification de la sortie.

Étape suivante

Étape 4 : Vérification de la sortie

Étape 4 : Vérification de la sortie

Après avoir configuré la sortie de l'application dans the section called "Étape 3 : Configuration de la sortie de l'application", utilisez les commandes de l'AWS CLI suivantes pour lire les enregistrements dans le flux de destination qui est écrit par l'application :

1. Exécutez la commande `get-shard-iterator` pour obtenir un pointeur vers les données du flux de sortie.

   ```
   aws kinesis get-shard-iterator \
   --shard-id shardId-000000000000 \
   --shard-iterator-type TRIM_HORIZON \
   --stream-name OutputStreamTestingAnomalyScores \
   --region us-east-1 \
   --profile adminuser
   ```

   Vous obtenez une réponse comportant une valeur d'itérateur de partition, comme illustré dans l'exemple de réponse suivant :

```
{
    "ShardIterator":
    "shard-iterator-value"   }
```

Copiez la valeur d'itérateur de partition.

2. Exécutez la commande AWS CLI`get-records`.

```
aws kinesis get-records \
--shard-iterator shared-iterator-value \
--region us-east-1 \
--profile adminuser
```

La commande renvoie une page d'enregistrements et un autre itérateur de partition que vous pouvez utiliser dans la commande `get-records` suivante pour extraire l'ensemble d'enregistrements suivant.

Exemple : détection d'anomalies de données et message d'explication (fonction RANDOM_CUT_FOREST_WITH_EXPLANATION)

Amazon Kinesis Data Analytics fournit la fonction `RANDOM_CUT_FOREST_WITH_EXPLANATION`, qui attribue un score d'anomalie à chaque enregistrement en fonction de valeurs dans les colonnes numériques. La fonction fournit également une explication de l'anomalie. Pour plus d'informations, consultez RANDOM_CUT_FOREST_WITH_EXPLANATION dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

Dans cet exercice, vous allez écrire du code d'application pour obtenir des scores d'anomalie pour des enregistrements dans la source de diffusion de votre application. Vous allez également obtenir une explication pour chaque anomalie.

Rubriques

- Étape 1 : Préparation des données
- Étape 2 : Création d'une application d'analyse
- Étape 3 : Évaluation des résultats

Première étape

Étape 1 : Préparation des données

Étape 1 : Préparation des données

Avant de créer une application Amazon Kinesis Data Analytics pour cet exemple, vous devez créer un flux de données Kinesis que vous allez utiliser comme source de streaming pour votre application. Vous pouvez également exécuter un code Python pour écrire des données simulées de pression artérielle dans le flux.

Rubriques

- Étape 1.1 : Création d'un flux de données Kinesis
- Étape 1.2 : Ecriture d'exemples d'enregistrements dans le flux d'entrée

Étape 1.1 : Création d'un flux de données Kinesis

Dans cette section, vous allez créer un flux de données Kinesis nommé `ExampleInputStream`. Vous pouvez créer ce flux de données à l'aide de l'AWS Management Console ou de l'AWS CLI.

- Pour utiliser la console :
  1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
  2. Choisissez Data Streams (Flux de données) dans le volet de navigation. Ensuite, choisissez Create Kinesis stream (Créer un flux Kinesis).
  3. Pour le nom, saisissez **`ExampleInputStream`**. Pour le nombre de partitions, saisissez **`1`**.
- Vous pouvez également utiliser l'AWS CLI pour créer le flux de données à l'aide de la commande suivante :

```
$ aws kinesis create-stream --stream-name ExampleInputStream --shard-count 1
```

Étape 1.2 : Ecriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette étape, vous exécutez du code Python pour générer en continu des exemples d'enregistrements et les écrire dans le flux de données que vous avez créé.

1. Installez Python et pip.

   Pour plus d'informations sur l'installation de Python, consultez le site Python.

Vous pouvez installer des dépendances à l'aide de pip. Pour plus d'informations sur l'installation de pip, consultez la section Installation dans la documentation de pip.

2. Exécutez le code Python suivant. Vous pouvez modifier la région en choisissant celle que vous souhaitez utiliser pour cet exemple. La commande `put-record` dans le code écrit les enregistrements JSON dans le flux.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class PressureType(Enum):
    low = "LOW"
    normal = "NORMAL"
    high = "HIGH"


def get_blood_pressure(pressure_type):
    pressure = {"BloodPressureLevel": pressure_type.value}
    if pressure_type == PressureType.low:
        pressure["Systolic"] = random.randint(50, 80)
        pressure["Diastolic"] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure["Systolic"] = random.randint(90, 120)
        pressure["Diastolic"] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure["Systolic"] = random.randint(130, 200)
        pressure["Diastolic"] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure


def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
            PressureType.low
```

```
            if rnd < 0.005
            else PressureType.high
            if rnd > 0.995
            else PressureType.normal
        )
        blood_pressure = get_blood_pressure(pressure_type)
        print(blood_pressure)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(blood_pressure),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Étape suivante

Étape 2 : Création d'une application d'analyse

Étape 2 : Création d'une application d'analyse

Dans cette section, vous allez créer une application Amazon Kinesis Data Analytics et la configurer pour utiliser le flux de données Kinesis que vous avez créé en tant que source de streaming dans the section called “Étape 1 : Préparation des données”. Vous pouvez ensuite exécuter le code de l'application qui utilise la fonction `RANDOM_CUT_FOREST_WITH_EXPLANATION`.

Pour créer une application

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez Data Analytics (Analyse des données) dans le volet de navigation, puis Create application (Créer une application).
3. Attribuez un nom et une description (facultatif) à votre application, puis sélectionnez Create application
4. Cliquez sur Connect streaming data (Connecter des données de diffusion), puis sélectionnez ExampleInputStream dans la liste.

5. Sélectionnez Discover schema et vérifiez que `Systolic` et `Diastolic` apparaissent sous la forme de colonnes `INTEGER`. S'ils sont associés à un autre type, sélectionnez Edit schema et attribuez-leur le type `INTEGER`.
6. Sous Real time analytics, sélectionnez Go to SQL editor. À l'invite, choisissez d'exécuter votre application.
7. Collez le code suivant dans l'éditeur SQL, puis choisissez Save and run SQL.

```
--Creates a temporary stream.
CREATE OR REPLACE STREAM "TEMP_STREAM" (
         "Systolic"                 INTEGER,
         "Diastolic"                INTEGER,
         "BloodPressureLevel"       varchar(20),
         "ANOMALY_SCORE"            DOUBLE,
         "ANOMALY_EXPLANATION"      varchar(512));

--Creates another stream for application output.
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
         "Systolic"                 INTEGER,
         "Diastolic"                INTEGER,
         "BloodPressureLevel"       varchar(20),
         "ANOMALY_SCORE"            DOUBLE,
         "ANOMALY_EXPLANATION"      varchar(512));

-- Compute an anomaly score with explanation for each record in the input stream
-- using RANDOM_CUT_FOREST_WITH_EXPLANATION
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "TEMP_STREAM"
      SELECT STREAM "Systolic", "Diastolic", "BloodPressureLevel", ANOMALY_SCORE,
 ANOMALY_EXPLANATION
      FROM TABLE(RANDOM_CUT_FOREST_WITH_EXPLANATION(
              CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001"), 100, 256,
 100000, 1, true));

-- Sort records by descending anomaly score, insert into output stream
CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM * FROM "TEMP_STREAM"
      ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE DESC;
```

Étape suivante

Étape 3 : Évaluation des résultats

Étape 3 : Évaluation des résultats

Lorsque vous exécutez le code SQL pour cet exemple, vous devez d'abord voir des lignes avec un score d'anomalie égal à zéro. Cela se produit au cours de la phase d'apprentissage initiale. Vous obtenez ensuite des résultats similaires à ce qui suit:

```
ROWTIME SYSTOLIC DIASTOLIC BLOODPRESSURELEVEL ANOMALY_SCORE ANOMALY_EXPLANATION
27:49.0 101      66        NORMAL             0.711460417   {"Systolic":
{"DIRECTION":"LOW","STRENGTH":"0.0922","ATTRIBUTION_SCORE":"0.3792"},"Diastolic":
{"DIRECTION":"HIGH","STRENGTH":"0.0210","ATTRIBUTION_SCORE":"0.3323"}}
27:50.0 144      123       HIGH               3.855851061   {"Systolic":
{"DIRECTION":"HIGH","STRENGTH":"0.8567","ATTRIBUTION_SCORE":"1.7447"},"Diastolic":
{"DIRECTION":"HIGH","STRENGTH":"7.0982","ATTRIBUTION_SCORE":"2.1111"}}
27:50.0 113      69        NORMAL             0.740069409   {"Systolic":
{"DIRECTION":"LOW","STRENGTH":"0.0549","ATTRIBUTION_SCORE":"0.3750"},"Diastolic":
{"DIRECTION":"LOW","STRENGTH":"0.0394","ATTRIBUTION_SCORE":"0.3650"}}
27:50.0 105      64        NORMAL             0.739644157   {"Systolic":
{"DIRECTION":"HIGH","STRENGTH":"0.0245","ATTRIBUTION_SCORE":"0.3667"},"Diastolic":
{"DIRECTION":"LOW","STRENGTH":"0.0524","ATTRIBUTION_SCORE":"0.3729"}}
27:50.0 100      65        NORMAL             0.736993425   {"Systolic":
{"DIRECTION":"HIGH","STRENGTH":"0.0203","ATTRIBUTION_SCORE":"0.3516"},"Diastolic":
{"DIRECTION":"LOW","STRENGTH":"0.0454","ATTRIBUTION_SCORE":"0.3854"}}
27:50.0 108      69        NORMAL             0.733767202   {"Systolic":
{"DIRECTION":"LOW","STRENGTH":"0.0974","ATTRIBUTION_SCORE":"0.3961"},"Diastolic":
{"DIRECTION":"LOW","STRENGTH":"0.0189","ATTRIBUTION_SCORE":"0.3377"}}
```

- L'algorithme de la fonction `RANDOM_CUT_FOREST_WITH_EXPLANATION` voit que les colonnes `Systolic` et `Diastolic` sont au format numérique et les utilise en tant qu'entrées.
- La colonne `BloodPressureLevel` contient des données texte et n'est donc pas prise en compte par l'algorithme. Cette colonne est une simple aide visuelle qui, dans cet exemple, vous permet d'identifier rapidement les différents niveaux de pression artérielle (normal, élevé et faible).
- Dans la colonne `ANOMALY_SCORE`, les enregistrements associés à des scores plus élevés sont anormaux. Avec une score d'anomalie de 3,855851061, le deuxième enregistrement dans cet exemple d'ensemble de résultats est le plus anormal.
- Pour comprendre dans quelle mesure chaque colonne numérique prise en compte par l'algorithme contribue au score d'anomalie, consultez le champ JSON nommé `ATTRIBUTION_SCORE` dans la colonne `ANOMALY_SCORE`. Dans le cas de la deuxième ligne de cet exemple d'ensemble de

résultats, les colonnes `Systolic` et `Diastolic` contribuent au score d'anomalie selon une proportion de 1.7447:2 .1111. En d'autres termes, 45 % de l'explication du score d'anomalie est imputable à la valeur systolique, le reste étant attribué à la valeur diastolique.

- Pour déterminer la direction dans laquelle le point représenté par la seconde ligne de cet exemple est anormal, consultez le champ JSON nommé `DIRECTION`. Les valeurs systolique et diastolique sont marquées en tant que `HIGH` dans ce cas. Pour déterminer le degré de confiance quant à l'exactitude de ces directions, consultez le champ JSON nommé `STRENGTH`. Dans cet exemple, l'algorithme a d'autant plus confiance que la valeur diastolique est élevée. En effet, la valeur normale du relevé diastolique est généralement comprise dans la plage 60 à 80, ce qui signifie qu'une valeur de 123 est bien plus élevée que prévu.

Exemple : Détection des points chauds sur un flux (fonction HOTSPOTS)

Amazon Kinesis Data Analytics fournit la fonction `HOTSPOTS` qui peut rechercher et renvoyer des informations sur les régions relativement denses de vos données. Pour plus d'informations, consultez HOTSPOTS dans le manuel Référence SQL du service géré Amazon pour Apache Flink.

Dans cet exercice, vous allez écrire du code d'application pour localiser les points chauds sur la source de streaming de votre application. Pour configurer l'application, exécutez les étapes suivantes :

1. Configurer une source de streaming : vous configurez un flux Kinesis et écrivez des exemples de données de coordonnées, comme illustré ci-après :

   ```
   {"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
   {"x": 0.722248626528026, "y": 4.648868803193405, "is_hot": "Y"}
   ```

   L'exemple fournit un script Python qui vous permet de remplir le flux. Les valeurs x et y sont générées de façon aléatoire, avec des enregistrements regroupés autour de certains emplacements.

   Le champ `is_hot` est fourni en tant qu'indicateur si le script a généré intentionnellement la valeur dans le cadre d'un point chaud. Cela peut vous aider à évaluer si la fonction de détection des points chauds fonctionne correctement.
2. Créer l'application : avec AWS Management Console, vous pouvez alors créer une application Kinesis Data Analytics. Configurez l'entrée d'application en mappant la source de streaming sur un flux intégré à l'application (`SOURCE_SQL_STREAM_001`). Lorsque l'application démarre,

Kinesis Data Analytics lit en continu la source de streaming et insère des enregistrements dans le flux intégré à l'application.

Dans cet exercice, vous utilisez le code suivant pour l'application :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    "x" DOUBLE,
    "y" DOUBLE,
    "is_hot" VARCHAR(4),
    HOTSPOTS_RESULT VARCHAR(10000)
);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT "x", "y", "is_hot", "HOTSPOTS_RESULT"
    FROM TABLE (
        HOTSPOTS(
            CURSOR(SELECT STREAM "x", "y", "is_hot" FROM "SOURCE_SQL_STREAM_001"),
            1000,
            0.2,
            17)
    );
```

Le code lit les lignes dans le flux `SOURCE_SQL_STREAM_001`, l'analyse pour rechercher les points chauds significatifs et écrit les données résultantes dans une autre flux intégré à l'application (`DESTINATION_SQL_STREAM`). Vous utilisez des pompes pour insérer des lignes dans les flux intégrés à l'application. Pour de plus amples informations, veuillez consulter Flux et pompes intégrés à l'application.

3. Configurer la sortie : vous configurez la sortie de l'application pour envoyer des données depuis l'application vers une destination externe qui est un autre flux de données Kinesis. Vérifiez les scores de point chaud et identifiez ceux qui indiquent qu'un point chaud a eu lieu (et que vous devez être alerté). Vous pouvez utiliser une fonction AWS Lambda pour traiter plus en détails les informations sur les points chauds et configurer des alertes.
4. Vérifiez la sortie – L'exemple inclut une application JavaScript qui lit les données du flux de sortie et les affiche de façon graphique pour que vous puissiez consulter les points chauds que l'application génère en temps réel.

L'exercice utilise la région USA Ouest (Oregon) (`us-west-2`) pour créer ces flux et votre application. Si vous utilisez une autre région, mettez à jour le code en conséquence.

Rubriques

Étape 1 : Créer les flux d'entrée et de sortie

Avant de créer une application Amazon Kinesis Data Analytics pour l'exemple pour les points chauds, vous créez deux flux de données Kinesis. Configurez l'un des flux en tant que source de streaming pour votre application et l'autre flux en tant que destination où Kinesis Data Analytics conserve la sortie de votre application.

Rubriques

Étape 1.1 : Création des flux de données Kinesis

Dans cette section, vous créez deux flux de données Kinesis : `ExampleInputStream` et `ExampleOutputStream`.

Créez ces flux de données à l'aide de la console ou de l'AWS CLI.

- Pour créer les flux de données à l'aide de la console :

  1. Connectez-vous à la AWS Management Console et ouvrez la console Kinesis à partir de l'adresse https://console.aws.amazon.com/kinesis.
  2. Choisissez Data Streams (Flux de données) dans le volet de navigation.
  3. Choisissez Create Kinesis stream (Créer un flux Kinesis), puis créez un flux avec une partition nommée `ExampleInputStream`.
  4. Répétez l'étape précédente, en créant un flux avec une seule partition nommée `ExampleOutputStream`.

- Pour créer des flux de données à l'aide de l'AWS CLI :
  - Créez des flux (`ExampleInputStream` et `ExampleOutputStream`) à l'aide de la commande AWS CLI `create-stream` Kinesis suivante. Pour créer le deuxième flux que l'application utilisera pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser

$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Étape 1.2 : Ecriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette étape, vous exécutez du code Python pour générer en continu des exemples d'enregistrements et les écrire dans le flux `ExampleInputStream`.

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626580026, "y": 4.648868803193405, "is_hot": "Y"}
```

1. Installez Python et `pip`.

   Pour plus d'informations sur l'installation de Python, consultez le site web Python.

   Vous pouvez installer des dépendances à l'aide de pip. Pour plus d'informations sur l'installation de pip, consultez Installation sur le site web de pip.
2. Exécutez le code Python suivant. Ce code effectue les opérations suivantes :
   - Génère un point chaud potentiel quelque part dans le plan (X, Y).

- Génère un ensemble de 1 000 points pour chaque point chaud. Parmi ces points, 20 % sont regroupés autour du point chaud. Les autres sont générés de façon aléatoire dans l'ensemble de l'espace.
- La commande `put-record` écrit les enregistrements JSON dans le flux.

> **Important**
>
> Ne chargez pas ce fichier sur un serveur web, car il contient vos informations d'identification AWS.

```
import json
from pprint import pprint
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_hotspot(field, spot_size):
    hotspot = {
        "left": field["left"] + random.random() * (field["width"] - spot_size),
        "width": spot_size,
        "top": field["top"] + random.random() * (field["height"] - spot_size),
        "height": spot_size,
    }
    return hotspot


def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        "x": rectangle["left"] + random.random() * rectangle["width"],
        "y": rectangle["top"] + random.random() * rectangle["height"],
        "is_hot": "Y" if rectangle is hotspot else "N",
    }
    return {"Data": json.dumps(point), "PartitionKey": "partition_key"}
```

```
def generate(
    stream_name, field, hotspot_size, hotspot_weight, batch_size, kinesis_client
):
    """
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)
        ]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)


if __name__ == "__main__":
    generate(
        stream_name=STREAM_NAME,
        field={"left": 0, "width": 10, "top": 0, "height": 10},
        hotspot_size=1,
        hotspot_weight=0.2,
        batch_size=10,
        kinesis_client=boto3.client("kinesis"),
    )
```

Étape suivante

[Étape 2 : Création d'une application Kinesis Data Analytics]

Étape 2 : Création d'une application Kinesis Data Analytics

Dans cette section de l'exemple pour les points chauds, vous créez une application Kinesis Data Analytics en procédant comme suit :

- Configurez l'entrée de l'application pour utiliser le flux de données Kinesis que vous avez créé comme source de streaming à l'étape 1.
- Utilisez le code d'application fourni dans l'AWS Management Console.

Pour créer une application

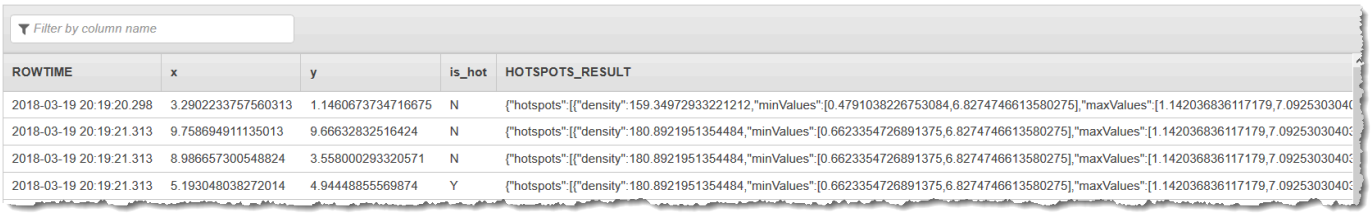
1. Créez une application Kinesis Data Analytics en suivant les étapes 1, 2 et 3 de l'exercice Mise en route (voir Étape 3.1 : Créer une application).

   Dans la configuration de la source, procédez de la façon suivante :

   - Spécifiez la source de streaming que vous avez créée dans the section called "Étape 1 : Créer les flux".
   - Une fois que la console a déduit le schéma, modifiez celui-ci. Assurez-vous que les types de colonne `x` et `yDOUBLE` sont définis sur , et que le type de colonne `IS_HOT` est défini sur `VARCHAR`.
2. Utilisez le code d'application suivant (vous pouvez coller ce code dans l'éditeur SQL) :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    "x" DOUBLE,
    "y" DOUBLE,
    "is_hot" VARCHAR(4),
    HOTSPOTS_RESULT VARCHAR(10000)
);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT "x", "y", "is_hot", "HOTSPOTS_RESULT"
    FROM TABLE (
        HOTSPOTS(
            CURSOR(SELECT STREAM "x", "y", "is_hot" FROM "SOURCE_SQL_STREAM_001"),
            1000,
            0.2,
            17)
    );
```

3. Exécutez le code SQL et vérifiez les résultats.



Étape suivante

[Étape 3 : Configurer la sortie de l'application](#)

Étape 3 : Configurer la sortie de l'application

À ce stade de l'exemple pour les points chauds, vous avez un code d'application Amazon Kinesis Data Analytics qui identifie des points chauds significatifs d'une source de streaming et qui attribue un score de chaleur à chacun.

Vous pouvez maintenant envoyer les résultats de l'application depuis le flux intégré à l'application vers une destination externe, qui est un autre flux de données Kinesis (`ExampleOutputStream`). Vous pouvez ensuite analyser les scores de points chauds et déterminer un seuil approprié pour la chaleur des points chauds. Vous pouvez étendre encore cette application pour générer des alertes.

Pour configurer la sortie de l'application

1. Ouvrez la console Kinesis Data Analytics à l'adresse https://console.aws.amazon.com/kinesisanalytics.
2. Dans l'éditeur SQL, choisissez Destination ou Add a destination dans le tableau de bord d'application.
3. Sur la page Add a destination (Ajouter une destination), choisissez Select from your streams (Choisissez parmi vos flux). Choisissez ensuite le flux `ExampleOutputStream` que vous avez créé dans la section précédente.

   Maintenant, vous disposez d'une destination externe où Amazon Kinesis Data Analytics conserve tous les enregistrements que votre application écrit dans le flux intégré à l'application `DESTINATION_SQL_STREAM`.
4. Vous pouvez configurer AWS Lambda le cas échéant pour surveiller le flux `ExampleOutputStream` et vous envoyer des alertes. Pour de plus amples informations,

veuillez consulter Utilisation d'une fonction Lambda en tant que sortie. Vous pouvez également vérifier les enregistrements écrits par Kinesis Data Analytics dans la destination externe, qui est le flux Kinesis `ExampleOutputStream`, comme décrit dans Étape 4 : Vérifier la sortie de l'application.

Étape suivante

Étape 4 : Vérifier la sortie de l'application

Étape 4 : Vérifier la sortie de l'application

Dans cette section de l'exemple pour les points chauds, vous configurez une application web qui affiche des informations sur les points chauds dans un contrôle SVG (Scalable Vector Graphics).

1. Créez un fichier nommé `index.html` avec le contenu suivant :

```
<!doctype html>
<html lang=en>
<head>
    <meta charset=utf-8>
    <title>hotspots viewer</title>

    <style>
    #visualization {
      display: block;
      margin: auto;
    }

    .point {
      opacity: 0.2;
    }

    .hot {
      fill: red;
    }

    .cold {
      fill: blue;
    }

    .hotspot {
      stroke: black;
```

```
      stroke-opacity: 0.8;
      stroke-width: 1;
      fill: none;
    }
    </style>
    <script src="https://sdk.amazonaws.com/js/aws-sdk-2.202.0.min.js"></script>
    <script src="https://d3js.org/d3.v4.min.js"></script>
</head>
<body>
<svg id="visualization" width="600" height="600"></svg>
<script src="hotspots_viewer.js"></script>
</body>
</html>
```

2. Créez un fichier dans le même répertoire, nommé `hotspots_viewer.js` avec le contenu suivant. Fournissez votre , les informations d'identification et le nom du flux de sortie dans les variables fournies.

```
// Visualize example output from the Kinesis Analytics hotspot detection algorithm.
// This script assumes that the output stream has a single shard.

// Modify this section to reflect your AWS configuration
var awsRegion = "",        // The  where your Kinesis Analytics application is
 configured.
    accessKeyId = "",      // Your Access Key ID
    secretAccessKey = "",  // Your Secret Access Key
    outputStream = "";     // The name of the Kinesis Stream where the output from
 the HOTSPOTS function is being written

// The variables in this section should reflect way input data was generated and
 the parameters that the HOTSPOTS
// function was called with.
var windowSize = 1000, // The window size used for hotspot detection
    minimumDensity = 40,  // A filter applied to returned hotspots before
 visualization
    xRange = [0, 10],  // The range of values to display on the x-axis
    yRange = [0, 10];  // The range of values to display on the y-axis

////////////////////////////////////////////////////////////////////////////////
// D3 setup
////////////////////////////////////////////////////////////////////////////////
```

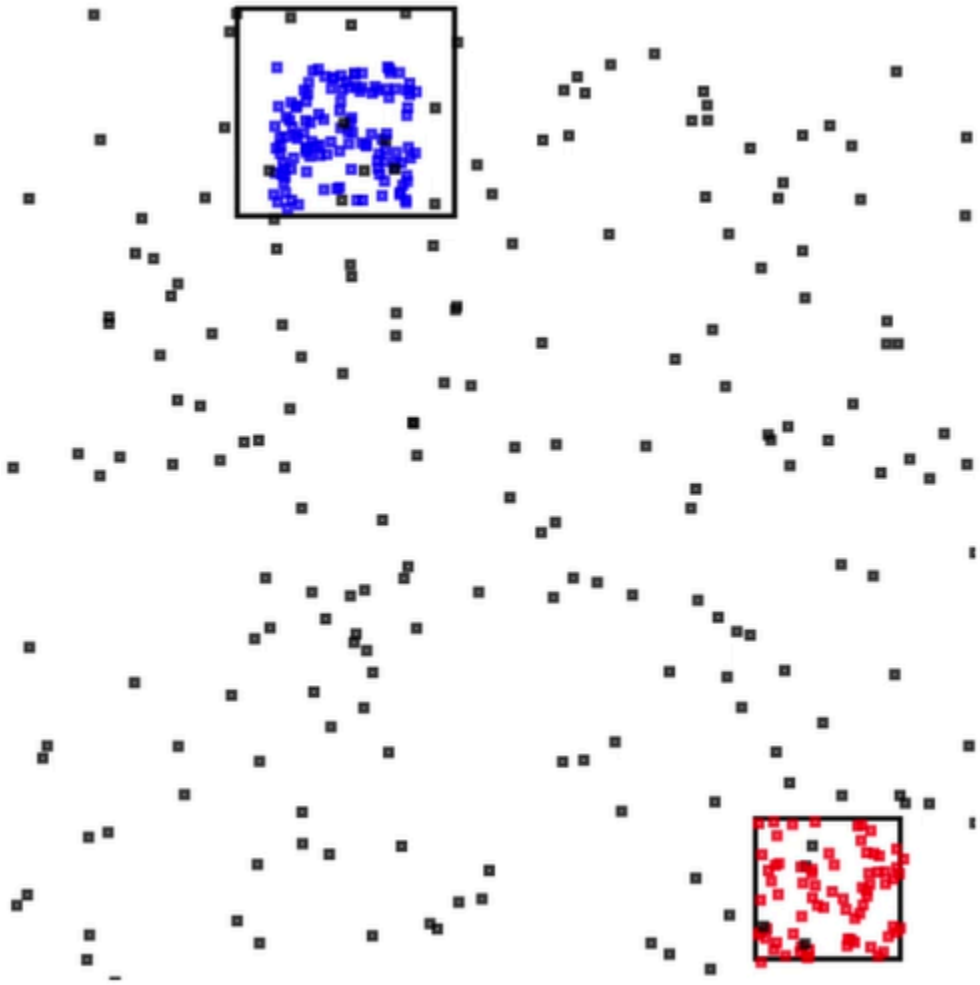
```
var svg = d3.select("svg"),
    margin = {"top": 20, "right": 20, "bottom": 20, "left": 20},
    graphWidth = +svg.attr("width") - margin.left - margin.right,
    graphHeight = +svg.attr("height") - margin.top - margin.bottom;

// Return the linear function that maps the segment [a, b] to the segment [c, d].
function linearScale(a, b, c, d) {
    var m = (d - c) / (b - a);
    return function(x) {
        return c + m * (x - a);
    };
}

// helper functions to extract the x-value from a stream record and scale it for
 output
var xValue = function(r) { return r.x; },
    xScale = linearScale(xRange[0], xRange[1], 0, graphWidth),
    xMap = function(r) { return xScale(xValue(r)); };

// helper functions to extract the y-value from a stream record and scale it for
 output
var yValue = function(r) { return r.y; },
    yScale = linearScale(yRange[0], yRange[1], 0, graphHeight),
    yMap = function(r) { return yScale(yValue(r)); };

// a helper function that assigns a CSS class to a point based on whether it was
 generated as part of a hotspot
var classMap = function(r) { return r.is_hot == "Y" ? "point hot" : "point
 cold"; };

var g = svg.append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

function update(records, hotspots) {

    var points = g.selectAll("circle")
        .data(records, function(r) { return r.dataIndex; });

    points.enter().append("circle")
        .attr("class", classMap)
        .attr("r", 3)
        .attr("cx", xMap)
        .attr("cy", yMap);
```

```
    points.exit().remove();

    if (hotspots) {
        var boxes = g.selectAll("rect").data(hotspots);

        boxes.enter().append("rect")
            .merge(boxes)
            .attr("class", "hotspot")
            .attr("x", function(h) { return xScale(h.minValues[0]); })
            .attr("y", function(h) { return yScale(h.minValues[1]); })
            .attr("width", function(h) { return xScale(h.maxValues[0]) -
 xScale(h.minValues[0]); })
            .attr("height", function(h) { return yScale(h.maxValues[1]) -
 yScale(h.minValues[1]); });

        boxes.exit().remove();
    }
}

////////////////////////////////////////////////////////////////////////////
// Use the AWS SDK to pull output records from Kinesis and update the visualization
////////////////////////////////////////////////////////////////////////////

var kinesis = new AWS.Kinesis({
    "region": awsRegion,
    "accessKeyId": accessKeyId,
    "secretAccessKey": secretAccessKey
});

var textDecoder = new TextDecoder("utf-8");

// Decode an output record into an object and assign it an index value
function decodeRecord(record, recordIndex) {
    var record = JSON.parse(textDecoder.decode(record.Data));
    var hotspots_result = JSON.parse(record.HOTSPOTS_RESULT);
    record.hotspots = hotspots_result.hotspots
        .filter(function(hotspot) { return hotspot.density >= minimumDensity});
    record.index = recordIndex
    return record;
}

// Fetch a new records from the shard iterator, append them to records, and update
 the visualization
```

```
function getRecordsAndUpdateVisualization(shardIterator, records, lastRecordIndex)
 {
    kinesis.getRecords({
        "ShardIterator": shardIterator
    }, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            return;
        }

        var newRecords = data.Records.map(function(raw) { return decodeRecord(raw,
 ++lastRecordIndex); });
        newRecords.forEach(function(record) { records.push(record); });

        var hotspots = null;
        if (newRecords.length > 0) {
            hotspots = newRecords[newRecords.length - 1].hotspots;
        }

        while (records.length > windowSize) {
            records.shift();
        }

        update(records, hotspots);

        getRecordsAndUpdateVisualization(data.NextShardIterator, records,
 lastRecordIndex);
    });
}

// Get a shard iterator for the output stream and begin updating the visualization.
 Note that this script will only
// read records from the first shard in the stream.
function init() {
    kinesis.describeStream({
        "StreamName": outputStream
    }, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            return;
        }

        var shardId = data.StreamDescription.Shards[0].ShardId;
```

```
        kinesis.getShardIterator({
            "StreamName": outputStream,
            "ShardId": shardId,
            "ShardIteratorType": "LATEST"
        }, function(err, data) {
            if (err) {
                console.log(err, err.stack);
                return;
            }
            getRecordsAndUpdateVisualization(data.ShardIterator, [], 0);
        })
    });
}

// Start the visualization
init();
```

3. Avec le code Python de l'exécution de la première section, ouvrez `index.html` dans un navigateur web. Les informations sur le point chaud s'affichent sur la page, comme illustré ci-après.



Exemples : Alertes et erreurs

Cette section fournit des exemples d'applications Kinesis Data Analytics qui utilisent des alertes et des erreurs. Chaque exemple fournit des instructions détaillées et du code pour vous aider à configurer et tester votre application Kinesis Data Analytics.

Rubriques

Exemple : Création d'alertes simples

Dans l'application Kinesis Data Analytics, la requête est exécutée en continu sur le flux intégré à l'application qui est créé sur le flux de démonstration. Pour de plus amples informations, veuillez consulter Requêtes continues.

Si des lignes montrent qu'un changement de cours d'une action est supérieur à 1 %, ces lignes sont insérées dans un autre flux intégré à l'application. Dans l'exercice, vous pouvez configurer la sortie de l'application pour conserver les résultats dans une destination externe. Vous pouvez alors étudier les résultats plus en détail. Par exemple, vous pouvez utiliser une fonction AWS Lambda pour traiter les enregistrements et vous envoyer des alertes.

Pour créer une application d'alertes simples

1. Créez une application d'analyse comme décrit dans l'exercice de mise en route de Kinesis Data Analytics.
2. Dans l'éditeur SQL dans Kinesis Data Analytics, remplacez le code d'application par les éléments suivants :

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM"
           (ticker_symbol VARCHAR(4),
            sector        VARCHAR(12),
            change        DOUBLE,
            price         DOUBLE);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM ticker_symbol, sector, change, price
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  (ABS(Change / (Price - Change)) * 100) > 1;
```

   L' instruction `SELECT` du code d'application filtre les lignes de `SOURCE_SQL_STREAM_001` pour obtenir les changements de cours d'action supérieurs à 1 %. Elle insère ensuite ces lignes dans un autre flux intégré à l'application, `DESTINATION_SQL_STREAM`, à l'aide d'une pompe. Pour plus d'informations sur le modèle de codage expliquant l'utilisation de pompes pour insérer des lignes dans des flux intégrés à l'application, consultez Code d'application.
3. Choisissez Save and run SQL (Enregistrer et exécuter SQL).
4. Ajoutez une destination. Pour ce faire, vous pouvez choisir l'onglet Destination dans l'éditeur SQL ou Add a destination (Ajouter une destination) dans la page de détails de l'application.

a. Dans l'éditeur SQL, choisissez l'onglet Destination, puis Connect to a destination (Se connecter à une destination).

   Sur la page Connect to destination (Se connecter à une destination), choisissez Create New (Créer Nouveau).

b. Choisissez Go to Kinesis Streams.

c. Dans la console Amazon Kinesis Data Streams, créez un nouveau flux Kinesis (par exemple, `gs-destination`) avec une partition. Attendez que l'état du flux soit ACTIVE.

d. Revenez à la console Kinesis Data Analytics. Sur la page Connect to destination (Se connecter à une destination), choisissez le flux que vous avez créé.

   Si le flux n'apparaît pas, actualisez la page.

e. Choisissez Save and continue (Enregistrer et continuer).

Maintenant, vous disposez d'une destination externe, un flux de données Kinesis, où Kinesis Data Analytics conserve la sortie de votre application dans le flux intégré à l'application `DESTINATION_SQL_STREAM`.

5. Configurez AWS Lambda pour surveiller le flux Kinesis que vous avez créé et appelez une fonction Lambda.

   Pour obtenir des instructions, consultez Prétraitement des données à l'aide d'une fonction Lambda.

Exemple : Création d'alertes limitées

Dans l'application Kinesis Data Analytics, la requête est exécutée en continu sur le flux intégré à l'application créé sur le flux de démonstration. Pour de plus amples informations, veuillez consulter Requêtes continues. Si des lignes montrent que le changement de cours d'une action est supérieur à 1 %, ces lignes sont insérées dans un autre flux intégré à l'application. L'application limite les alertes de sorte qu'une alerte est envoyée immédiatement lorsque le cours d'une action change. Toutefois, pas plus d'une alerte par minute par symbole boursier est envoyée au flux intégré à l'application.

Pour créer une application d'alertes limitées

1. Créez une application Kinesis Data Analytics comme décrit dans l'exercice de mise en route de Kinesis Data Analytics.

2. Dans l'éditeur SQL dans Kinesis Data Analytics, remplacez le code d'application par les éléments suivants :

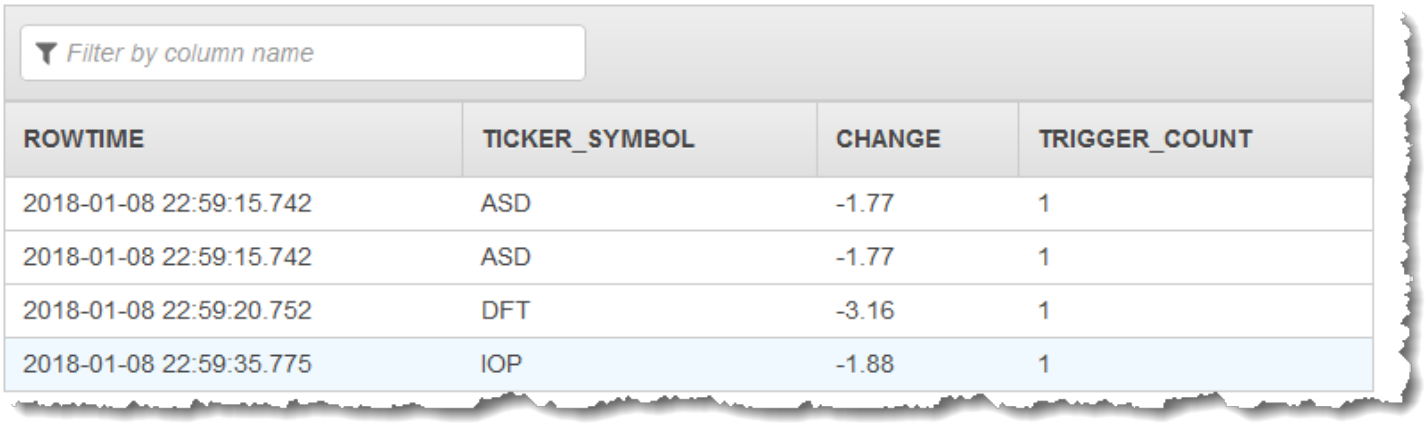
```
CREATE OR REPLACE STREAM "CHANGE_STREAM"
           (ticker_symbol VARCHAR(4),
            sector        VARCHAR(12),
            change        DOUBLE,
            price         DOUBLE);

CREATE OR REPLACE PUMP "change_pump" AS
   INSERT INTO "CHANGE_STREAM"
      SELECT STREAM ticker_symbol, sector, change, price
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  (ABS(Change / (Price - Change)) * 100) > 1;

-- ** Trigger Count and Limit **
-- Counts "triggers" or those values that evaluated true against the previous where
 clause
-- Then provides its own limit on the number of triggers per hour per ticker symbol
 to what
-- is specified in the WHERE clause

CREATE OR REPLACE STREAM TRIGGER_COUNT_STREAM (
   ticker_symbol VARCHAR(4),
   change REAL,
   trigger_count INTEGER);

CREATE OR REPLACE PUMP trigger_count_pump AS INSERT INTO TRIGGER_COUNT_STREAM
SELECT STREAM ticker_symbol, change, trigger_count
FROM (
    SELECT STREAM ticker_symbol, change, COUNT(*) OVER W1 as trigger_count
    FROM "CHANGE_STREAM"
    --window to perform aggregations over last minute to keep track of triggers
    WINDOW W1 AS (PARTITION BY ticker_symbol RANGE INTERVAL '1' MINUTE PRECEDING)
)
WHERE trigger_count >= 1;
```

L'instruction SELECT du code d'application filtre les lignes de SOURCE_SQL_STREAM_001 pour obtenir les changements de cours d'action supérieurs à 1 % et insère ces lignes dans un autre flux intégré à l'application, CHANGE_STREAM, à l'aide d'une pompe.

L'application crée alors un second flux nommé `TRIGGER_COUNT_STREAM` pour les alertes limitées. Une deuxième requête sélectionne des enregistrements dans une fenêtre qui avance à chaque fois qu'un enregistrement y est admis, de telle sorte qu'un seul enregistrement par symbole boursier par minute est écrit dans le flux.

3. Choisissez Save and run SQL (Enregistrer et exécuter SQL).

L'exemple génère dans `TRIGGER_COUNT_STREAM` un flux similaire à ce qui suit :



Exemple : Exploration du flux d'erreurs intégré à l'application

Amazon Kinesis Data Analytics fournit un flux d’erreurs intégré à l’application pour chaque application que vous créez. Les lignes que votre application ne peut pas traiter sont envoyées à ce flux d'erreurs. Vous pouvez envisager de conserver les données du flux d'erreurs dans une destination externe pour pouvoir les examiner.

Vous effectuez les exercices suivants sur la console. Dans ces exemples, vous introduisez des erreurs dans la configuration d'entrée en modifiant le schéma qui est déduit par le processus de découverte, puis vous vérifiez les lignes envoyées au flux d'erreurs.

Rubriques

- Introduction d'une erreur d'analyse
- Introduction d'une erreur de division par zéro

Introduction d'une erreur d'analyse

Dans cet exercice, vous introduisez une erreur d'analyse.

1. Créez une application Kinesis Data Analytics comme décrit dans l'exercice de mise en route de Kinesis Data Analytics.
2. Sur la page de détails de l'application, choisissez Connect streaming data (Connecter des données de diffusion).
3. Si vous avez suivi l'exercice de mise en route, vous disposez d'un flux de démonstration (`kinesis-analytics-demo-stream`) dans votre compte. Sur la page Connect to source (Se connecter à la source), choisissez ce flux de démonstration.
4. Kinesis Data Analytics prélève un échantillon dans le flux de démonstration pour déduire un schéma pour le flux d'entrée intégré à l'application qu'il crée. La console affiche le schéma déduit et les échantillons de données dans l'onglet Formatted stream sample.
5. Ensuite, éditez le schéma et modifiez le type de colonne pour introduire l'erreur d'analyse. Choisissez Edit schema (Modifier le schéma).
6. Remplacez le type de colonne `TICKER_SYMBOL VARCHAR(4)` par `INTEGER`.

   Le type de colonne du schéma intégré à l'application qui a été créé n'est pas valide, ce qui signifie que Kinesis Data Analytics ne peut pas introduire de données dans le flux intégré à l'application. Au lieu de cela, le service envoie les lignes dans le flux d'erreurs.
7. Choisissez Save schema.
8. Choisissez Refresh schema samples.

   Notez qu'il n'y a pas de lignes dans l'exemple Formatted stream. Toutefois, l'onglet Error stream affiche des données avec un message d'erreur. L'onglet Error stream montre les données envoyées aux flux d'erreurs intégré à l'application.

   Comme vous avez modifié le type de données de colonne, Kinesis Data Analytics n'a pas pu importer les données dans le flux d'entrée intégré à l'application. Au lieu de cela, le service envoie les données dans le flux d'erreurs.

Introduction d'une erreur de division par zéro

Dans le cadre de cet exercice, vous allez mettre à jour le code de l'application pour introduire une erreur d'exécution (division par zéro). Notez qu'Amazon Kinesis Data Analytics envoie les lignes obtenues sur le flux d'erreurs intégré à l'application, et non sur le flux intégré à l'application `DESTINATION_SQL_STREAM` dans lequel les résultats doivent logiquement être écrits.

1. Créez une application Kinesis Data Analytics comme décrit dans l'exercice de mise en route de Kinesis Data Analytics.

   Vérifiez les résultats dans l'onglet Real-time analytics en procédant comme suit :

   -

2. Mettez à jour l'instruction `SELECT` dans le code d'application pour introduire une division par zéro, par exemple :

   ```
   SELECT STREAM ticker_symbol, sector, change, (price / 0) as ProblemColumn
   FROM "SOURCE_SQL_STREAM_001"
   WHERE sector SIMILAR TO '%TECH%';
   ```

3. Exécutez l'application.

   En raison de l'erreur d'exécution de division par zéro, au lieu d'écrire les résultats dans le flux `DESTINATION_SQL_STREAM`, Kinesis Data Analytics envoie les lignes dans le flux d'erreurs intégré à l'application. Dans l'onglet Real-time analytics (Analyse en temps réel), choisissez le flux d'erreurs. Vous pouvez alors voir les lignes du flux d'erreurs intégré à l'application.

Exemples : Accélérateurs de solution

Le AWSsite de solutions dispose de modèles AWS CloudFormation que vous pouvez utiliser pour créer rapidement des solutions de données de streaming complètes.

Les modèles suivants sont disponibles :

Informations en temps réel sur l'activité de Compte AWS

Cette solution enregistre et affiche des métriques d'accès et d'utilisation des ressources pour votre ou vos Compte AWS en temps réel. Pour plus d'informations, consultez Informations en temps réel sur l'activité de Compte AWS.

Surveillance des appareils AWS IoT en temps réel avec Kinesis Data Analytics

Cette solution collecte, traite, analyse et affiche les données de connectivité et d'activité d'appareil IoT en temps réel. Pour plus d'informations, consultez Surveillance des appareils AWS IoT en temps réel avec Kinesis Data Analytics.

Analyse web en temps réel avec Kinesis Data Analytics

Cette solution collecte, traite, analyse et affiche les données de flux de clics sur les sites web en temps réel. Pour plus d'informations, consultez Analyse web en temps réel avec Kinesis Data Analytics.

Amazon Connected Vehicle Solution

Cette solution collecte, traite, analyse et affiche les données IoT de véhicules en temps réel. Pour plus d'informations, consultez Amazon Connected Vehicle Solution.

Sécurité dans

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficierez 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, consultez Services AWS concernés par le 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 ainsi que les lois et réglementations applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de . Les rubriques suivantes expliquent comment configurer pour répondre à vos objectifs de sécurité et de conformité. Vous pouvez également apprendre à utiliser d'autres services Amazon qui vous permettent de surveiller et de sécuriser vos ressources.

Rubriques

Protection des données dans les applications Amazon Kinesis Data Analytics pour SQL

Vous pouvez protéger vos données à l'aide des outils fournis par AWS. Kinesis Data Analytics peut fonctionner avec des services qui prennent en charge le chiffrement des données, notamment Kinesis Data Streams, Firehose et Amazon S3.

Chiffrement des données dans Kinesis Data Analytics

Chiffrement au repos

Notez les éléments suivants à propos du chiffrement des données au repos avec Kinesis Data Analytics :

- Vous pouvez chiffrer les données du flux de données Kinesis entrant à l'aide de. StartStreamEncryption Pour plus d'informations, consultez Qu'est-ce que le chiffrement côté serveur pour Kinesis Streams Data ?.
- Les données de sortie peuvent être chiffrées au repos à l'aide de Firehose pour stocker les données dans un compartiment Amazon S3 chiffré. Vous pouvez spécifier la clé de chiffrement que le compartiment Amazon S3 utilise. Pour plus d'informations, consultez Protection des données à l'aide du chiffrement côté serveur avec des clés gérées par KMS (SSE-KMS).
- Le code de votre application est chiffré au repos.
- Les données de référence de votre application sont chiffrées au repos.

Chiffrement en transit

Kinesis Data Analytics chiffre toutes les données en transit. Le chiffrement en transit est activé pour toutes les applications Kinesis Data Analytics et ne peut pas être désactivé.

Kinesis Data Analytics chiffre les données en transit dans les scénarios suivants :

- Données en transit de Kinesis Data Streams vers Kinesis Data Analytics.
- Données en transit entre les composants internes dans Kinesis Data Analytics.
- Données en transit entre Kinesis Data Analytics et Firehose.

Gestion des clés

Le chiffrement des données dans Kinesis Data Analytics utilise des clés gérées par le service. Les clés gérées par le client ne sont pas prises en charge.

Gestion des identités et des accès dans Kinesis Data Analytics

Amazon Kinesis Data Analytics a besoin d'autorisations pour lire les enregistrements provenant d'une source de diffusion que vous spécifiez dans la configuration d'entrée de votre application. Amazon Kinesis Data Analytics a également besoin d'autorisations pour écrire les résultats de votre application dans les flux que vous spécifiez dans la configuration de sortie de votre application.

Vous pouvez accorder ces autorisations en créant un rôle IAM qu'Amazon Kinesis Data Analytics peut endosser. Les autorisations que vous accordez à ce rôle déterminent ce qu'Amazon Kinesis Data Analytics peut faire lorsque le service assume ce rôle.

> **Note**
>
> Les informations de cette section sont utiles si vous voulez créer un rôle IAM vous-même. Lorsque vous créez une application dans la console Amazon Kinesis Data Analytics, celle-ci peut créer un rôle IAM pour vous à ce moment-là. La console utilise la convention d'attribution des noms suivante pour les rôles IAM qu'elle crée :
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> Une fois le rôle créé, vous pouvez le vérifier, ainsi que les stratégies attachées dans la console IAM.

Deux stratégies sont attachées à chaque rôle IAM. Dans la stratégie d'approbation, vous spécifiez qui peut endosser le rôle. Dans la stratégie d'autorisations (il peut y en avoir plusieurs), vous spécifiez les autorisations que vous voulez accorder à ce rôle. Les sections suivantes décrivent ces stratégies que vous pouvez utiliser lorsque vous créez un rôle IAM.

Stratégie d’approbation

Pour accorder à Amazon Kinesis Data Analytics des autorisations pour assumer un rôle afin d’accéder à une source de streaming ou de référence, vous pouvez attacher la stratégie d’approbation suivante à un rôle IAM :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Stratégie d’autorisations

Si vous créez un rôle IAM pour permettre à Amazon Kinesis Data Analytics de lire à partir d’une source de streaming d’une application, vous devez accorder des autorisations pour les actions de lecture pertinentes. En fonction de votre source (par exemple, un flux Kinesis, un flux de diffusion Firehose ou une source de référence dans un compartiment Amazon S3), vous pouvez joindre la politique d'autorisation suivante.

Stratégie d’autorisations pour la lecture d’un flux Kinesis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
```

```
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/inputStreamName"
            ]
        }
    ]
}
```

Politique d'autorisation pour lire un stream de diffusion Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:Get*"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-
id:deliverystream/inputFirehoseName"
            ]
        }
    ]
}
```

> **Note**
>
> L'autorisation `firehose:Get*` fait référence à une méthode accesseur interne qui utilise Kinesis Data Analytics pour accéder au flux. Il n'y a pas d'accès public pour un stream de diffusion Firehose.

Si vous demandez à Amazon Kinesis Data Analytics d'écrire la sortie dans des destinations externes dans la configuration de sortie de votre application, vous devez accorder les autorisations suivantes au rôle IAM.

Stratégie d'autorisations pour l'écriture dans un flux Kinesis

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/output-stream-name"
            ]
        }
    ]
}
```

Stratégie d'autorisations pour l'écriture dans un flux de diffusion Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-id:deliverystream/output-
firehose-name"
            ]
        }
    ]
}
```

Stratégie d’autorisations pour la lecture d’une source de données de référence à partir d’un compartiment Amazon S3

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

Prévention du problème de l'adjoint confus entre services

Dans AWS, l'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé pour agir sur les ressources d'un autre client, même s'il ne devrait pas avoir les autorisations appropriées, ce qui se traduit par un problème d'adjoint confus.

Pour éviter toute confusion chez les adjoints, AWS fournit des outils qui vous aident à protéger vos données pour tous les services en utilisant des responsables de service qui ont eu accès aux ressources de votre compte. Cette section se concentre sur la prévention du problème de l'adjoint confus entre services spécifique à Kinesis Data Analytics. Cependant, vous pouvez en savoir plus à ce sujet dans la section Le problème de l'adjoint confus du Guide de l'utilisateur IAM.

Dans le contexte de Kinesis Data Analytics for SQL, nous vous recommandons d'utiliser les clés de contexte global aws SourceArn SourceAccount : et aws : dans votre politique de confiance en matière de rôle afin de limiter l'accès au rôle aux seules demandes générées par les ressources attendues.

Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser l'association d'une ressource de ce compte à l'utilisation interservices.

La valeur de la clé `aws:SourceArn` doit être l'ARN de la ressource utilisée par Kinesis Data Analytics, qui est spécifié au format suivant : `arn:aws:kinesisanalytics:region:account:resource`.

Le moyen le plus efficace de se protéger contre le problème d'adjoint confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource.

Si vous ne connaissez pas l'ARN complet de la ressource ou si vous indiquez plusieurs ressources, utilisez la clé `aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple : `arn:aws:kinesisanalytics::111122223333:*`.

Bien que la plupart des actions de l'API Kinesis Data Analytics for SQL AddApplicationInput, CreateApplicationtelles que DeleteApplicationet soient effectuées dans le contexte d'applications spécifiques, DiscoverInputSchemaelles ne sont exécutées dans le contexte d'aucune application. Cela signifie que le rôle utilisé dans cette action ne doit pas spécifier complètement une ressource dans la clé de condition `SourceArn`. Voici un exemple d'utilisation d'un ARN générique :

```
{
   ...
   "ArnLike":{
      "aws:SourceArn":"arn:aws:kinesisanalytics:us-east-1:123456789012:*"
   }
   ...
}
```

Le rôle par défaut généré par Kinesis Data Analytics pour SQL utilise ce caractère générique. Cela garantit un fonctionnement transparent de la détection du schéma d'entrée dans l'expérience de la console. Cependant, nous vous recommandons de modifier la stratégie d'approbation afin d'utiliser un ARN complet après avoir détecté le schéma afin de mettre en œuvre une atténuation complète de l'adjoint confus.

Les politiques relatives aux rôles que vous fournissez à Kinesis Data Analytics ainsi que les politiques de confiance relatives aux rôles générés pour vous peuvent utiliser les clés de condition aws SourceArn : et aws SourceAccount :.

Afin de vous protéger contre le problème d'adjoint confus, effectuez les tâches suivantes :

Pour lutter contre le problème de l'adjoint confus

1. Connectez-vous à la console de AWS gestion et ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Rôles, puis choisissez le rôle que vous souhaitez modifier.
3. Choisissez Modifier la politique.
4. Sur la page Modifier la politique d'approbation, remplacez la politique JSON par défaut par une stratégie qui utilise l'une des clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` ou les deux. Voir l'exemple de stratégie suivant :

5. Choisissez Mettre à jour une politique.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"kinesisanalytics.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"Account ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:kinesisanalytics:us-
east-1:123456789012:application/my-app"
            }
         }
      }
   ]
}
```

Authentification et contrôle d’accès pour

L'accès à requiert des informations d'identifications. Ces informations d'identification doivent être autorisées à accéder à AWS des ressources, telles qu'une application ou une instance Amazon Elastic Compute Cloud (Amazon EC2). Les sections suivantes décrivent comment utiliser AWS Identity and Access Management (IAM) et pour contribuer à sécuriser l'accès à vos ressources.

Contrôle d'accès

Vous pouvez avoir des informations d'identification valides pour authentifier vos demandes, mais à moins d'avoir les autorisations requises, vous ne pouvez pas créer de ressources ni accéder à de telles ressources. Par exemple, vous devez disposer d'autorisations pour créer une application .

Les sections suivantes décrivent comment gérer les autorisations pour . Nous vous recommandons de lire d’abord la présentation.

- Présentation de la gestion des autorisations d'accès à vos ressources

- Utilisation des stratégies basées sur une identité (Politiques IAM) pour
- Autorisations d’API : référence des actions, des autorisations et des ressources

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 qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

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 (IAM Identity 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 avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d’identités avec des rôles IAM. 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 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 section Signature des demandes AWS d'API dans le guide de l'utilisateur IAM.

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, consultez Authentification multifactorielle dans le Guide de l’utilisateur AWS IAM Identity Center et Utilisation de l’authentification multifactorielle (MFA) dans l’interface AWS dans le Guide de l’utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez Tâches nécessitant des informations d'identification d'utilisateur root dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus

d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. 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 avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

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 Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- 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 obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez la rubrique Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'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 en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

- 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 direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM 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. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS 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 la politique relative à la transmission de demandes FAS, consultez Sessions de transmission d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.
  - 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 répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de 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 qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations,

consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Présentation de la gestion des autorisations d'accès à vos ressources

> **Warning**
>
> Pour les nouveaux projets, nous vous recommandons d'utiliser le nouveau service géré pour Apache Flink Studio plutôt que les applications Kinesis Data Analytics pour SQL. Le service géré pour Apache Flink Studio allie facilité d'utilisation et capacités analytiques avancées, ce qui vous permet de créer des applications sophistiquées de traitement des flux en quelques minutes.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .
- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Pour plus d'informations, voir la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) du Guide de l'utilisateur IAM.
- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM du Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

> **Note**
>
> Un administrateur de compte (ou utilisateur administrateur) est un utilisateur doté des privilèges d'administrateur. Pour plus d'informations, consultez Bonnes pratiques IAM dans le Guide de l'utilisateur IAM.

Rubriques

Ressources et opérations

La ressource principale est une application. Dans une politique, vous utilisez un Amazon Resource Name (ARN) pour identifier la ressource à laquelle la politique s'applique.

Ces ressources ont des noms ARN (Amazon Resource Name) uniques qui leur sont associés, comme cela est illustré dans le tableau suivant.

Type de ressource	Format ARN
Application	`arn:aws:kinesisanalytics:` *`region`*`:`*`account-id`*`:application/`*`application-name`*

fournit un ensemble d'opérations pour utiliser les ressources. Pour obtenir la liste des opérations disponibles, consultez Actions.

Présentation de la propriété des ressources

Il Compte AWS est propriétaire des ressources créées dans le compte, quelle que soit la personne qui les a créées. Plus précisément, le propriétaire Compte AWS de la ressource est l'entité principale (c'est-à-dire le compte root, un utilisateur ou un rôle IAM) qui authentifie la demande de création de ressource. Les exemples suivants illustrent comment cela fonctionne :

- Si vous utilisez les informations d'identification de votre compte root Compte AWS pour créer une application, vous Compte AWS êtes le propriétaire de la ressource. (Dans , la ressource est une application.)
- Si vous créez un utilisateur dans votre compte Compte AWS et que vous lui accordez l'autorisation de créer une application, celui-ci peut créer une application. Cependant, c'est à vous Compte AWS, à laquelle appartient l'utilisateur, que appartient la ressource de l'application. Nous vous recommandons vivement d’accorder des autorisations aux rôles et non aux utilisateurs.
- Si vous créez un rôle IAM Compte AWS avec les autorisations nécessaires pour créer une application, toute personne pouvant assumer ce rôle peut créer une application. Vous Compte AWS, à laquelle appartient l'utilisateur, êtes propriétaire de la ressource de l'application.

Gestion de l'accès aux ressources

Une politique d'autorisation décrit qui a accès à quoi. La section suivante explique les options disponibles pour créer des politiques d'autorisations.

> **Note**
>
> Cette section décrit l'utilisation d'IAM dans le contexte d' . Elle ne fournit pas d'informations détaillées sur le service IAM. Pour une documentation complète sur IAM, veuillez consulter Qu'est-ce qu'IAM ? dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la syntaxe et les descriptions des politiques IAM, veuillez consulter Référence de politique JSON IAM dans le Guide de l'utilisateur IAM.

Les politiques qui sont associées à une identité IAM sont appelées des politiques basées sur l'identité (politiques IAM). Les stratégies qui sont associées à une ressource sont appelées des stratégies basées sur les ressources. prend en charge uniquement les stratégies basées sur l’identité (politiques IAM).

Rubriques

- Politiques basées sur une identité (politiques IAM)
- Politiques basées sur une ressource

Politiques basées sur une identité (politiques IAM)

Vous pouvez attacher des politiques à des identités IAM. Par exemple, vous pouvez effectuer les opérations suivantes :

- Attacher une stratégie d'autorisations à un utilisateur ou à un groupe de votre compte : pour autoriser un utilisateur à créer une ressource, comme une application, vous pouvez attacher une stratégie d'autorisations à un utilisateur ou à un groupe auquel l'utilisateur appartient.
- Attacher une politique d'autorisations à un rôle (accorder des autorisations entre comptes) : vous pouvez attacher une politique d'autorisation basée sur une identité à un rôle IAM afin d'accorder des autorisations entre comptes. Par exemple, l'administrateur du compte A peut créer un rôle pour accorder des autorisations entre comptes à un autre Compte AWS (par exemple, le compte B) ou à un service Amazon comme suit :
  1. L'administrateur du compte A crée un rôle IAM et attache une politique d'autorisation à ce rôle qui accorde des autorisations sur les ressources dans le compte A.
  2. L'administrateur du compte A lie une politique d'approbation au rôle identifiant le compte B comme principal pouvant assumer ce rôle.
  3. L'administrateur du compte B peut alors déléguer les autorisations pour affecter ce rôle à tous les utilisateurs figurant dans le compte B. Les utilisateurs du compte B sont ainsi autorisés à créer des ressources ou à y accéder dans le compte A. Le principal dans la stratégie d'approbation peut également être un principal de service Amazon si vous souhaitez accorder à un service Amazon des autorisations pour assumer ce rôle.

  Pour en savoir plus sur l'utilisation d'IAM pour déléguer des autorisations, consultez Gestion des accès dans le Guide de l'utilisateur IAM.

Voici un exemple de stratégie qui autorise l'action `kinesisanalytics:CreateApplication` , ce qui est nécessaire pour créer une application.

> **Note**
>
> Voici un exemple de stratégie de présentation. Lorsque vous associez la politique à l'utilisateur, celui-ci pourra créer une application à l'aide du AWS CLI AWS SDK. Mais l'utilisateur aura besoin d'autres autorisations pour configurer l'entrée et sortie. En outre, l'utilisateur aura besoin d'autorisations supplémentaires lors de l'utilisation de la console. Les sections suivantes fournissent de plus amples informations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Pour plus d'informations sur l'utilisation des stratégies basées sur l'identité avec , voir Utilisation des stratégies basées sur une identité (Politiques IAM) pour . Pour de plus amples informations sur les utilisateurs, les groupes, les rôles et les autorisations, consultez Identités (utilisateurs, groupes et rôles) dans le Guide de l'utilisateur IAM.

Politiques basées sur une ressource

D'autres services, tels qu'Amazon S3, prennent également en charge les politiques d'autorisation basées sur une ressource. Par exemple, vous pouvez attacher une politique à un compartiment S3 pour gérer les autorisations d'accès à ce compartiment. ne prend pas en charge les politiques basées sur une ressource.

Spécification des éléments d'une politique : actions, effets et principaux

Pour chaque ressource , le service définit un ensemble d'opérations d'API. Pour accorder des autorisations pour ces opérations d'API, définit un ensemble d'actions que vous pouvez spécifier dans une politique. Certaines opérations d'API peuvent exiger des autorisations pour plusieurs actions afin de réaliser l'opération d'API. Pour plus d'informations sur les ressources et les opérations de l'API, consultez Ressources et opérations et Actions.

Voici les éléments les plus élémentaires d'une politique :

- Ressource : vous utilisez un nom Amazon Resource Name (ARN) pour identifier la ressource à laquelle s'applique la politique. Pour de plus amples informations, veuillez consulter Ressources et opérations.

- Action : vous utilisez des mots clés d'action pour identifier les opérations de ressource que vous voulez accorder ou refuser. Par exemple, vous pouvez utiliser `create` pour autoriser les utilisateurs à créer une application.
- Effet - Vous spécifiez l'effet produit, autorisation ou refus, lorsque l'utilisateur demande l'action spécifique. Si vous n'accordez pas explicitement l'accès pour (autoriser) une ressource, l'accès est implicitement refusé. Vous pouvez aussi explicitement refuser l'accès à une ressource, ce que vous pouvez faire afin de vous assurer qu'un utilisateur n'y a pas accès, même si une politique différente accorde l'accès.
- Principal – dans les politiques basées sur une identité (politiques IAM), l'utilisateur auquel la politique est attachée est le principal implicite. Pour les politiques basées sur une ressource, vous spécifiez l'utilisateur, le compte, le service ou une autre entité qui doit recevoir les autorisations (s'applique uniquement aux politiques basées sur une ressource). ne prend pas en charge les politiques basées sur une ressource.

Pour en savoir plus sur la syntaxe et les descriptions des politiques IAM, veuillez consulter Référence de politique JSON IAM dans le Guide de l'utilisateur IAM.

Pour obtenir une des tableaux présentant toutes les opérations d'API, ainsi que les ressources auxquelles elles s'appliquent, consultez Autorisations d'API : référence des actions, des autorisations et des ressources.

Spécification de conditions dans une politique

Lorsque vous accordez des autorisations, vous pouvez utiliser le langage de la politique d'accès pour spécifier les conditions définissant quand une politique doit prendre effet. Par exemple, il est possible d'appliquer une politique après seulement une date spécifique. Pour plus d'informations sur la spécification de conditions dans un langage de politique, consultez Condition dans le Guide de l'utilisateur IAM.

Pour exprimer des conditions, vous utilisez des clés de condition prédéfinies. Il n'existe pas de clés de condition spécifiques à . Cependant, il existe des AWS clés de condition larges que vous pouvez utiliser le cas échéant. Pour obtenir la liste complète des touches AWS-wide, consultez la section Clés disponibles pour les conditions dans le guide de l'utilisateur IAM.

Utilisation des stratégies basées sur une identité (Politiques IAM) pour

Les exemples suivants de stratégies basées sur les identités illustrent comment un administrateur de compte peut attacher des stratégies d'autorisation à des identités IAM (autrement dit, des utilisateurs,

des groupes et des rôles) et ainsi accorder des autorisations pour effectuer des opérations sur les ressources.

> **Important**
>
> Nous vous recommandons tout d'abord d'examiner les rubriques de présentation qui détaillent les concepts de base et les options disponibles pour gérer l'accès à vos ressources . Pour plus d'informations, consultez Présentation de la gestion des autorisations d'accès à vos ressources .

Rubriques

- Autorisations requises pour utiliser la console
- Stratégies (prédéfinies) gérées par Amazon pour
- Exemples de politiques gérées par le client

Un exemple de politique d'autorisation est exposé ci-dessous.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

La stratégie comporte une instruction :

- La première instruction accorde les autorisations pour une action (`kinesisanalytics:CreateApplication`) au niveau d'une ressource en utilisant l'Amazon

Resource Name (ARN) de l'application. Dans ce cas, l'ARN spécifie un caractère générique (*) pour indiquer que l'autorisation est accordée pour n'importe quelle ressource.

Pour visualiser un tableau répertoriant toutes les opérations d'API et les ressources auxquelles elles s'appliquent, voir Autorisations d'API : référence des actions, des autorisations et des ressources.

Autorisations requises pour utiliser la console

Pour qu'un utilisateur puisse utiliser la console , vous devez lui accorder les autorisations nécessaires. Par exemple, si vous voulez autoriser l'utilisateur à créer une application, vous devez lui accorder des autorisations qui lui montrent les sources de streaming dans le compte pour qu'il puisse configurer l'entrée et la sortie au niveau de la console.

Nous vous recommandons la procédure suivante :

- Utilisez les stratégies gérées par Amazon pour accorder des autorisations utilisateur. Pour obtenir les stratégies disponibles, consultez Stratégies (prédéfinies) gérées par Amazon pour .
- Créez des stratégies personnalisées. Dans ce cas, nous vous recommandons de consulter l'exemple fourni dans cette section. Pour plus d'informations, consultez Exemples de politiques gérées par le client.

Stratégies (prédéfinies) gérées par Amazon pour

AWS répond à de nombreux cas d'utilisation courants en fournissant des politiques IAM autonomes créées et administrées par. AWS Ces stratégies gérées par Amazon octroient les autorisations requises dans les cas d'utilisation courants, afin de vous épargner les réflexions sur les autorisations requises. Pour plus d'informations, consultez Stratégies gérées par Amazon dans le Guide de l'utilisateur IAM.

Les stratégies gérées par Amazon suivantes, que vous pouvez attacher aux utilisateurs de votre compte, sont propres à :

- **`AmazonKinesisAnalyticsReadOnly`** : accorde des autorisations pour des actions qui permettent à un utilisateur de répertorier les applications et de vérifier la configuration d'entrée/sortie. Il accorde également des autorisations qui permettent à un utilisateur de consulter la liste des flux Kinesis et des flux de diffusion Firehose. Comme l'application est en cours d'exécution,

l'utilisateur peut afficher les données source et les résultats des analyses en temps réel sur la console.

- **`AmazonKinesisAnalyticsFullAccess`** : accorde les autorisations pour toutes les actions et toutes les autres autorisations qui permettent à l'utilisateur de créer et gérer des applications. Toutefois, notez les points suivants :

  - Ces autorisations ne sont pas suffisantes si l'utilisateur souhaite créer un nouveau rôle IAM dans la console (ces autorisations permettent à l'utilisateur de sélectionner un rôle existant). Si vous souhaitez que l'utilisateur puisse créer un rôle IAM dans la console, ajoutez la stratégie gérée par Amazon `IAMFullAccess`.

  - Un utilisateur doit avoir l'autorisation pour l'action `iam:PassRole` afin de spécifier un rôle IAM lors de la configuration d'une application. Cette stratégie gérée par Amazon accorde à l'utilisateur l'autorisation pour l'action `iam:PassRole` uniquement sur les rôles IAM qui commencent par le préfixe `service-role/kinesis-analytics`.

    Si l'utilisateur souhaite configurer l'application avec un rôle qui n'a pas ce préfixe, vous devez d'abord accorder explicitement l'autorisation utilisateur pour l'action `iam:PassRole` sur le rôle spécifique.

Vous pouvez également créer vos propres politiques IAM personnalisées afin d'accorder des autorisations pour les actions et les ressources . Vous pouvez attacher ces stratégies personnalisées aux utilisateurs ou groupes qui nécessitent ces autorisations.

Exemples de politiques gérées par le client

Cette section fournit un ensemble d'exemples de stratégies que vous pouvez associer à un utilisateur. Si vous créez des stratégies pour la première fois, nous vous recommandons de commencer par créer un utilisateur dans votre compte, puis de lui associer les stratégies dans l'ordre décrit dans les étapes de cette section. Vous pouvez alors utiliser la console pour vérifier les effets de chaque stratégie lorsque vous l'attachez à l'utilisateur.

Au départ, l'utilisateur ne dispose pas des autorisations requises et ne peut donc exécuter aucune action dans la console. À mesure que vous lui associez des stratégies, vous pouvez vérifier que l'utilisateur peut exécuter diverses actions dans la console.

Nous vous recommandons d'utiliser deux fenêtres de navigateur. Dans une fenêtre, créez l'utilisateur et accordez des autorisations. Dans l'autre, connectez-vous à l' AWS Management Console aide des informations d'identification de l'utilisateur et vérifiez les autorisations que vous leur accordez.

Pour des exemples qui illustrent comment créer un rôle IAM que vous pouvez utiliser comme rôle d’exécution pour votre application, consultez Création de rôles IAM dans le Guide de l’utilisateur IAM.

Étapes d'exemple

- Étape 1 : Création d'un utilisateur IAM
- Étape 2 : Octroyer des autorisations à l’utilisateur pour des actions qui sont pas spécifiques à
- Étape 3 : Permettre à l'utilisateur d'afficher une liste d'applications et de voir des détails
- Étape 4 : Permettre à l'utilisateur de démarrer une application spécifique
- Étape 5 : Permettre à l'utilisateur de créer une application
- Étape 6 : Autoriser l’application à utiliser le prétraitement Lambda

Étape 1 : Création d'un utilisateur IAM

Vous devez d’abord créer un utilisateur, l’ajouter à un groupe IAM possédant des autorisations d’administration, puis lui attribuer ces autorisations. Vous pouvez ensuite accéder à AWS l'aide d'une URL spéciale et des informations d'identification de cet utilisateur.

Pour obtenir des instructions, veuillez consulter Création de votre premier groupe d'utilisateurs et d'administrateurs IAM dans le Guide de l'utilisateur IAM.

Étape 2 : Octroyer des autorisations à l’utilisateur pour des actions qui sont pas spécifiques à

Tout d'abord, nous devons accorder une autorisation à l'utilisateur pour toutes les actions qui ne sont pas spécifiques à et dont l'utilisateur aura besoin lorsqu'il utilise des applications . Il s'agit notamment des autorisations pour travailler avec les flux (actions Amazon Kinesis Data Streams, actions Amazon Data Firehose) et des autorisations pour les actions. CloudWatch Attachez la stratégie suivante à l'utilisateur.

Vous devez mettre à jour la stratégie en fournissant un nom de rôle IAM pour lequel vous voulez accorder l'autorisation `iam:PassRole` ou spécifier un caractère générique (*) indiquant tous les rôles

IAM. Il ne s'agit pas d'une pratique sécurisée ; cependant, vous pouvez ne pas disposer d'un rôle IAM créé lors de ce test.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:CreateStream",
                "kinesis:DeleteStream",
                "kinesis:DescribeStream",
                "kinesis:ListStreams",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:GetLogEvents",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListPolicyVersions",
```

```
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/service-role/role-name"
        }
    ]
}
```

Étape 3 : Permettre à l'utilisateur d'afficher une liste d'applications et de voir des détails

La stratégie suivante accorde à un utilisateur les autorisations suivantes :

- L'autorisation pour l'action `kinesisanalytics:ListApplications` permettant à l'utilisateur d'afficher la liste des applications. Notez qu'il s'agit d'un appel d'API de niveau service ; vous spécifiez donc « * » comme valeur de `Resource`.
- L'autorisation pour l'action `kinesisanalytics:DescribeApplication` pour pouvoir obtenir des informations sur les applications.

Ajoutez cette stratégie à l'utilisateur.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:ListApplications"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:DescribeApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/*"
        }
```

```
    ]
}
```

Vérifiez ces autorisations en vous connectant à la console à l'aide des informations d'identification de l'utilisateur.

Étape 4 : Permettre à l'utilisateur de démarrer une application spécifique

Si vous voulez que l'utilisateur puisse démarrer l'une des applications existantes, vous pouvez lui attacher la stratégie suivante. Elle fournit l'autorisation pour l'action `kinesisanalytics:StartApplication`. Vous devez mettre à jour la politique en fournissant votre identifiant de compte, votre AWS région et le nom de l'application.

```
{
    "Version": "2012-10-17",
    "Statement": [
                {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

Étape 5 : Permettre à l'utilisateur de créer une application

Si vous voulez que l'utilisateur puisse créer une application , vous pouvez ensuite lui attacher la stratégie suivante. Vous devez mettre à jour la politique et fournir une AWS région, votre identifiant de compte et soit un nom d'application spécifique que vous souhaitez que l'utilisateur crée, soit un « * » afin que l'utilisateur puisse spécifier n'importe quel nom d'application (et ainsi créer plusieurs applications).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
```

```
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication",
                "kinesisanalytics:UpdateApplication",
                "kinesisanalytics:AddApplicationInput",
                "kinesisanalytics:AddApplicationOutput"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

Étape 6 : Autoriser l'application à utiliser le prétraitement Lambda

Si vous voulez que l'application puisse utiliser le prétraitement Lambda, attachez la stratégie suivante au rôle.

```
    {
      "Sid": "UseLambdaFunction",
      "Effect": "Allow",
      "Action": [
          "lambda:InvokeFunction",
          "lambda:GetFunctionConfiguration"
      ],
      "Resource": "<FunctionARN>"
  }
```

Autorisations d'API : référence des actions, des autorisations et des ressources

Lorsque vous configurez Contrôle d'accès et que vous créez une stratégie d'autorisation que vous pouvez attacher à une identité IAM (stratégies basées sur une identité), vous pouvez utiliser la de tableaux ci-dessous comme référence. Le chaque opération d'API, les actions correspondantes pour

lesquelles vous pouvez accorder des autorisations pour effectuer l'action et la AWS ressource pour laquelle vous pouvez accorder les autorisations. Vous spécifiez les actions dans le champ `Action` de la politique ainsi que la valeur des ressources dans le champ `Resource` de la politique.

Vous pouvez utiliser des AWS clés de condition larges dans vos polices pour exprimer des conditions. Pour obtenir la liste complète des touches AWS-wide, consultez la section Clés disponibles dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour indiquer une action, utilisez le préfixe `kinesisanalytics` suivi du nom de l'opération d'API (par exemple, `kinesisanalytics:AddApplicationInput`).

API et autorisations requises pour les actions

Opération d'API :

Autorisations requises (Action d'API) :

Ressources :

API et autorisations requises pour les actions

API Amazon RDS et autorisations requises pour les actions

Opération d'API : AddApplicationInput

Action : `kinesisanalytics:AddApplicationInput`

Ressources :

`arn:aws:kinesisanalytics:` *`region`*`:`*`accountId`*`:application/`*`application-name`*

GetApplicationState

La console utilise une méthode interne appelée `GetApplicationState` pour échantillonner les données d'application ou y accéder. Votre application de service doit disposer des autorisations

permettant à l'API interne `kinesisanalytics:GetApplicationState` d'échantillonner les données d'application ou d'y accéder via l' AWS Management Console.

Surveillance

fournit des fonctionnalités de surveillance pour vos applications. Pour plus d'informations, consultez Surveillance.

Validation de conformité pour les applications Amazon Kinesis Data Analytics pour SQL

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon Kinesis Data Analytics dans le cadre de AWS plusieurs programmes de conformité. Il s'agit notamment des certifications SOC, PCI, HIPAA.

Pour obtenir la liste des AWS services concernés par des programmes de conformité spécifiques, consultez Amazon Services concernés par programme de conformité. Pour obtenir des informations générales, consultez Programmes de conformitéAWS.

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, consultez la section Téléchargement de rapports dans AWS Artifact.

Votre responsabilité de conformité lors de l'utilisation de Kinesis Data Analytics est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise, ainsi que par la législation et la réglementation applicables. Si votre utilisation de Kinesis Data Analytics est soumise à la conformité aux normes HIPAA ou PCI, AWS fournit des ressources pour vous aider :

- 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 la sécurité et la conformité sur. AWS
- Livre blanc sur l'architecture pour la sécurité et la conformité HIPAA — Ce livre blanc décrit comment les entreprises peuvent créer des applications conformes à la loi HIPAA. AWS
- AWS Ressources relatives à la conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Config— Ce AWS 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— Ce AWS service fournit une vue complète de l'état de votre sécurité interne, AWS ce qui vous permet de vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité.

Résilience dans Amazon Kinesis Data 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 de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives 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.

Outre l'infrastructure AWS mondiale, Kinesis Data Analytics propose plusieurs fonctionnalités pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Reprise après sinistre

Kinesis Data Analytics s’exécute en mode sans serveur et s’occupe des dégradations de l’hôte, de la disponibilité des zones de disponibilité et d’autres problèmes liés à l’infrastructure en effectuant une migration automatique. Lorsque cela se produit, Kinesis Data Analytics permet de s’assurer que l’application est traitée sans perte de données. Pour plus d’informations, consultez Modèle de diffusion pour la conservation de la sortie d'application dans une destination externe.

Sécurité de l’infrastructure dans les applications Kinesis Data Analytics pour SQL

En tant que service géré, Amazon Kinesis Data Analytics est protégé par AWS les procédures de sécurité du réseau mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Kinesis Data Analytics via le réseau. Les clients doivent prendre en charge le protocole TLS (Transport Layer Security) 1.2 ou version

ultérieure. Les clients doivent aussi prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (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 ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Bonnes pratiques de sécurité pour Kinesis Data Analytics

Amazon Kinesis Data Analytics fournit différentes fonctionnalités de sécurité à prendre en compte lorsque vous développez et implémentez 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.

Utilisation de rôles IAM pour accéder à d'autres services Amazon

Votre application Kinesis Data Analytics doit disposer d'informations d'identification valides pour accéder aux ressources d'autres services, tels que les flux de données Kinesis, les flux de diffusion Firehose ou les buckets Amazon S3. Vous ne devez pas stocker les AWS informations d'identification directement dans l'application 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 rôle IAM pour gérer des informations d'identification temporaires afin que votre application accède à d'autres ressources. Lorsque vous utilisez un rôle, vous n'avez pas à utiliser d'informations d'identification à long terme pour accéder à d'autres ressources.

Pour plus d'informations, consultez les rubriques suivantes dans le Guide de l'utilisateur IAM :

- Rôles IAM
- Scénarios courants pour les rôles : utilisateurs, applications et services.

Implémentation d'un chiffrement côté serveur dans des ressources dépendantes

Les données au repos et les données en transit sont chiffrées dans Kinesis Data Analytics, et ce chiffrement ne peut pas être désactivé. Vous devez implémenter le chiffrement côté serveur dans vos ressources dépendantes, telles que les flux de données Kinesis, les flux de diffusion Firehose et les compartiments Amazon S3. Pour plus d'informations sur l'implémentation du chiffrement côté serveur dans les ressources dépendantes, reportez-vous à Protection des données.

CloudTrail À utiliser pour surveiller les appels d'API

Kinesis Data Analytics est intégré AWS CloudTrailà un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un service Amazon dans Kinesis Data Analytics.

À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande envoyée à Kinesis Data Analytics, l'adresse IP à partir de laquelle la demande a été effectuée, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires.

Pour plus d’informations, consultez the section called “Utiliser AWS CloudTrail”.

Surveillance de pour les applications SQL

La surveillance est essentielle pour assurer la fiabilité, la disponibilité et les performances d' et de votre application . Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une panne multipoint, le cas échéant. Avant de commencer la surveillance d', toutefois, vous devez créer un plan de surveillance qui contient 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 référence de performances normales d'un dans votre environnement, en mesurant la performance à divers moments et dans diverses conditions de charge. Lorsque vous surveillez , vous pouvez stocker des données de surveillance historiques. Vous pouvez alors les comparer avec les données de performances actuelles, identifier des modèles de performances normales et des anomalies de performances et concevoir des méthodes pour les résoudre.

Avec , vous surveillez l'application. L'application traite les flux de données (entrée ou sortie), qui incluent tous deux des identifiants que vous pouvez utiliser pour affiner votre recherche sur CloudWatch les journaux. Pour plus d'informations sur la façon dont traite des flux de données, consultez Applications Amazon Kinesis Data Analytics pour SQL : fonctionnement.

La métrique la plus importante est `millisBehindLatest` qui indique le retard de lecture de votre application pour la source de diffusion. Dans un cas typique, le nombre de millisecondes doit être à égal à zéro ou proche de zéro. Il est fréquent que de courts pics aient lieu, ce qui apparaît comme une augmentation de la valeur `millisBehindLatest`.

Nous vous recommandons de configurer une CloudWatch alarme qui se déclenche lorsque l'application est en retard de plus d'une heure après avoir lu la source du streaming. Pour certains cas d'utilisation qui nécessitent un traitement pratiquement en temps réel, par exemple, l'émission

de données vers une application active, vous pouvez choisir de régler l'alarme sur une valeur plus basse, comme cinq minutes.

Rubriques

Outils de supervision

AWS fournit divers outils que vous pouvez utiliser pour surveiller. 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.

Outils de surveillance automatique

Vous pouvez utiliser les outils de surveillance automatique pour surveiller et signaler en cas de 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 (Amazon SNS) 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.
- Amazon CloudWatch Logs — Surveillez, stockez et accédez à vos fichiers journaux depuis AWS CloudTrail ou d'autres sources. Pour plus d'informations, consultez la section Monitoring Log Files dans le guide de CloudWatch l'utilisateur Amazon.
- Amazon CloudWatch Events : associez les événements et acheminez-les vers une ou plusieurs fonctions ou flux cibles afin d'apporter des modifications, de recueillir des informations d'état et de prendre des mesures correctives. Pour plus d'informations, consultez la section Qu'est-ce qu'Amazon CloudWatch Events dans le guide de CloudWatch l'utilisateur Amazon.
- AWS CloudTrail Surveillance des journaux : partagez les fichiers journaux entre les comptes, surveillez les fichiers CloudTrail CloudWatch journaux en temps réel en les envoyant à Logs,

écrivez des applications de traitement des journaux en Java et vérifiez que vos fichiers journaux n'ont pas changé après leur livraison par CloudTrail. Pour plus d'informations, consultez la section Utilisation des fichiers CloudTrail journaux dans le guide deAWS CloudTrail l'utilisateur.

Outils de surveillance manuelle

Une autre partie importante de la surveillance consiste à surveiller manuellement les éléments non couverts par les CloudWatch alarmes. Le tableau de bord CloudWatch Trusted Advisor,, et les autres AWS Management Console tableaux de bord fournissent une at-a-glance vue de l'état de votre AWS environnement.

- 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
  - Rechercher et parcourir toutes vos métriques
  - Créer et modifier des alarmes pour être informé des problèmes
- AWS Trusted Advisor peut vous aider à surveiller votre activité afin d'améliorer les performances, la fiabilité, la sécurité et la rentabilité. Quatre contrôles Trusted Advisor sont disponibles pour tous les utilisateurs. Plus de 50 contrôles sont disponibles pour les utilisateurs avec un plan de support Business ou Enterprise. Pour de plus amples informations, veuillez consulter AWS Trusted Advisor.

Surveillance avec Amazon CloudWatch

Vous pouvez surveiller les applications à l'aide d'Amazon CloudWatch. CloudWatch collecte et traite les données brutes pour en faire des indicateurs lisibles en temps quasi réel. Ces statistiques sont conservées pour une durée de deux semaines. 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, les données métriques sont automatiquement envoyées à CloudWatch. Pour

plus d'informations, consultez Qu'est-ce qu'Amazon CloudWatch ? dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

Métriques et dimensions

L'espace de noms `AWS/KinesisAnalytics` inclut les métriques suivantes.

Métrique	Description
`Bytes`	Nombre d'octets lus (par flux d'entrée) ou écrits (par flux de sortie). Niveaux : par flux d'entrée et par flux de sortie
`KPUs`	Nombre moyen d’unités de traitement Kinesis qui sont utilisées pour exécuter votre application de traitement de flux. Le nombre moyen d'unités de traitement Kinesis (KPU) utilisées chaque heure détermine la facturation pour votre application. Niveaux : niveau application
`MillisBehindLatest`	Indique le retard de lecture d'une application, par rapport à l'heure actuelle, pour la source de diffusion. Niveaux : niveau application
`Records`	Nombre d'enregistrements lus (par flux d'entrée) ou écrits (par flux de sortie). Niveaux : par flux d'entrée et par flux de sortie

Métrique	Description
`Success`	1 pour chaque tentative de remise avec succès à la destination configurée pour votre application ; 0 pour chaque tentative de remise ayant échoué. La valeur moyenne de cette métrique indique le nombre de remises réussies qui ont été effectuées. Niveaux : par destination.
`InputProcessing.Duration`	Le temps nécessaire pour chaque appel de AWS Lambda fonction effectué par. Niveaux : par flux d'entrée
`InputProcessing.OkRecords`	Nombre d’enregistrements renvoyés par une fonction Lambda qui ont été marqués avec le statut `Ok`. Niveaux : par flux d'entrée
`InputProcessing.OkBytes`	Somme d’octets des enregistrements renvoyés par une fonction Lambda qui ont été marqués avec le statut `Ok`. Niveaux : par flux d'entrée
`InputProcessing.DroppedRecords`	Nombre d’enregistrements renvoyés par une fonction Lambda qui ont été marqués avec le statut `Dropped`. Niveaux : par flux d'entrée
`InputProcessing.ProcessingFailedRecords`	Nombre d’enregistrements renvoyés par une fonction Lambda qui ont été marqués avec le statut `ProcessingFailed` . Niveaux : par flux d'entrée
`InputProcessing.Success`	Nombre d’appels Lambda réussis par . Niveaux : par flux d'entrée

Métrique	Description
`LambdaDelivery.OkRecords`	Nombre d'enregistrements renvoyés par une fonction Lambda qui ont été marqués avec le statut `Ok`. Niveaux : par destination Lambda
`LambdaDelivery.DeliveryFailedRecords`	Nombre d'enregistrements renvoyés par une fonction Lambda qui ont été marqués avec le statut `DeliveryFailed` . Niveaux : par destination Lambda
`LambdaDelivery.Duration`	Durée de l'appel de chaque fonction Lambda exécutée par . Niveaux : par destination Lambda

fournit des métriques pour les dimensions suivantes.

Dimension	Description
`Flow`	Par flux d'entrée : entrée Par flux de sortie : sortie
`Id`	Par flux d'entrée : ID d'entrée Par flux de sortie : ID de sortie

Affichage des métriques et dimensions

Lorsque votre application traite des flux de données, envoie les métriques et dimensions suivantes à CloudWatch. Vous pouvez utiliser les procédures suivantes pour afficher les métriques d'.

Dans la console, les métriques sont d'abord regroupées par espace de noms de service, puis par les combinaisons de dimension au sein de chaque espace de noms.

Pour afficher les métriques à l'aide de la CloudWatch console

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Metrics (Métriques).
3. Dans le volet CloudWatch Mesures par catégorie pour, choisissez une catégorie de mesures.
4. Dans le volet supérieur, faites défiler l'écran pour afficher la liste complète des métriques.

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/KinesisAnalytics" --region region
```

Les métriques sont collectées aux niveaux suivants :

- Application
- Flux d'entrée
- Flux de sortie

Création d' CloudWatch alarmes à surveiller

Vous pouvez créer une CloudWatch alarme Amazon qui envoie un message Amazon SNS lorsque l'alarme change d'état. Une alarme surveille une métrique sur 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 rubrique Amazon SNS ou à une politique Auto Scaling.

Les alertes invoquent les actions pour les changements d'état soutenus uniquement. Pour qu'une CloudWatch alarme déclenche une action, l'état doit avoir changé et être maintenu pendant une durée spécifiée.

Vous pouvez définir des alarmes à l'aide de CloudWatch l'API AWS Management ConsoleCloudWatch AWS CLI, ou, comme décrit ci-dessous.

Pour régler une alarme à l'aide de la CloudWatch console

1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Sélectionnez Create Alarm (Créer une alerte). L'Assistant Créer une alarme s'affiche.
3. Sélectionnez Kinesis Analytics Metrics (Métriques Kinesis Analytics). Faites défiler les métriques jusqu'à trouver celle sur laquelle vous souhaitez placer une alarme.

   Pour afficher seulement les métriques , recherchez l'ID de votre système de fichiers. Sélectionnez la métrique sur laquelle créer une alarme, puis sélectionnez Suivant.
4. Saisissez les valeurs Nom, Description et Lorsque de la métrique.
5. Si vous souhaitez vous CloudWatch envoyer un e-mail lorsque l'état d'alarme est atteint, dans le champ Chaque fois que cette alarme est :, choisissez State is ALARM. Dans le champ Send notification to:, choisissez une rubrique SNS existante. Si vous sélectionnez Create topic, vous pouvez définir le nom d'une nouvelle liste d'abonnement par e-mail et les adresses e-mail pour cette liste. La liste est enregistrée et s'affiche dans le champ des alarmes futures.

   > **Note**
   >
   > Si vous utilisez Créer la rubrique pour créer une nouvelle rubrique Amazon SNS, les adresses e-mail doivent être vérifiées avant de pouvoir recevoir des notifications. Les e-mails sont envoyés uniquement lorsque l'alarme passe à un état défini. Si ce changement d'état de l'alarme se produit avant la vérification des adresses e-mail, ces dernières ne reçoivent pas de notification.
6. Dans la sectionAperçu de l'alarme, prévisualisez l'alarme que vous êtes sur le point de créer.
7. Choisissez Créer une alarme pour créer l'alarme.

Pour définir une alarme à l'aide de la CloudWatch CLI

- Appelez `mon-put-metric-alarm`. Pour plus d'informations, consultez le manuel Amazon CloudWatch CLI Reference.

Pour configurer une alarme à l'aide de l' CloudWatch API

- Appelez `PutMetricAlarm`. Pour plus d'informations, consultez le Amazon CloudWatch API Reference.

Utilisation d'Amazon CloudWatch Logs

Si une application est mal configurée, elle peut se lancer lors du démarrage de l'application. Ou elle peut mettre à jour, mais sans traiter les données dans le flux d'entrée intégré à l'application. En ajoutant une option de CloudWatch journalisation à l'application, vous pouvez surveiller les problèmes de configuration de l'application.

peut générer des erreurs de configuration dans les conditions suivantes :

- Le flux de données Kinesis utilisé pour l’entrée n’existe pas.
- Le flux de diffusion Amazon Data Firehose utilisé pour les entrées n'existe pas.
- Le compartiment Amazon S3 utilisé comme source de données de référence n’existe pas.
- Le fichier spécifié dans la source de données de référence du compartiment S3 n'existe pas.
- La ressource appropriée n'est pas définie dans le rôle AWS Identity and Access Management (IAM) qui gère les autorisations associées.
- L'autorisation n'est pas correctement définie dans le rôle IAM qui gère les autorisations connexes.
- n’est pas autorisé à assumer le rôle IAM qui gère les autorisations connexes.

Pour plus d'informations sur Amazon CloudWatch, consultez le guide de CloudWatch l'utilisateur Amazon.

Ajouter l'action PutLogEvents politique

a besoin d'autorisations pour écrire des erreurs de configuration dans CloudWatch. Vous pouvez ajouter ces autorisations au rôle IAM assumé par , comme décrit ci-dessous. Pour plus d’informations sur l’utilisation d’un rôle IAM pour , consultez Gestion des identités et des accès dans Kinesis Data Analytics.

Stratégie d'approbation

Pour autoriser à assumer un rôle IAM, vous pouvez attacher à ce dernier la stratégie d'approbation suivante.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Stratégie d'autorisations

Pour accorder à une application l'autorisation d'écrire les événements CloudWatch du journal dans une ressource, vous pouvez utiliser la politique d'autorisation IAM suivante.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt0123456789000",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:log-
stream:my-log-stream*"
            ]
        }
    ]
}
```

Ajout de la surveillance des erreurs de configuration

Utilisez les actions d'API suivantes pour ajouter une option de CloudWatch journal à une application nouvelle ou existante ou pour modifier une option de journal pour une application existante.

> **Note**
>
> Pour le moment, vous ne pouvez ajouter une option de CloudWatch journalisation à une application qu'à l'aide d'actions d'API. Vous ne pouvez pas ajouter d'options de CloudWatch journalisation à l'aide de la console.

Ajouter une option de CloudWatch journal lors de la création d'une application

L'exemple de code suivant montre comment utiliser l'`CreateApplication`action pour ajouter une option de CloudWatch journal lorsque vous créez une application. Pour plus d'informations sur `Create_Application`, consultez CreateApplication.

```
{
   "ApplicationCode": "<The SQL code the new application will run on the input
 stream>",
   "ApplicationDescription": "<A friendly description for the new application>",
   "ApplicationName": "<The name for the new application>",
   "Inputs": [ ... ],
   "Outputs": [ ... ],
   "CloudWatchLoggingOptions": [{
      "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add
 to the new application>",
      "RoleARN": "<ARN of the role to use to access the log>"
 }]
}
```

Ajouter une option de CloudWatch journalisation à une application existante

L'exemple de code suivant montre comment utiliser l'action `AddApplicationCloudWatchLoggingOption` pour ajouter une option de journal CloudWatch à une application existante. Pour plus d'informations sur `AddApplicationCloudWatchLoggingOption`, consultez AddApplicationCloudWatchLoggingOption.

```
{
   "ApplicationName": "<Name of the application to add the log option to>",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "<ARN of the log stream to add to the application>",
      "RoleARN": "<ARN of the role to use to access the log>"
   },
   "CurrentApplicationVersionId": <Version of the application to add the log to>
}
```

Mettre à jour une option de CloudWatch journal existante

L'exemple de code suivant montre comment utiliser l'`UpdateApplication`action pour modifier une option de CloudWatch journal existante. Pour plus d'informations sur `UpdateApplication`, consultez UpdateApplication.

```
{
   "ApplicationName": "<Name of the application to update the log option for>",
   "ApplicationUpdate": {
      "CloudWatchLoggingOptionUpdates": [
         {
            "CloudWatchLoggingOptionId": "<ID of the logging option to modify>",
            "LogStreamARNUpdate": "<ARN of the new log stream to use>",
            "RoleARNUpdate": "<ARN of the new role to use to access the log stream>"
         }
      ],
   },
   "CurrentApplicationVersionId": <ID of the application version to modify>
}
```

Supprimer une option de CloudWatch journalisation d'une application

L'exemple de code suivant montre comment utiliser l'`DeleteApplicationCloudWatchLoggingOption`action pour supprimer une option de CloudWatch journal existante. Pour plus d'informations sur `DeleteApplicationCloudWatchLoggingOption`, consultez DeleteApplicationCloudWatchLoggingOption.

```
{
   "ApplicationName": "<Name of application to delete log option from>",
   "CloudWatchLoggingOptionId": "<ID of the application log option to delete>",
   "CurrentApplicationVersionId": <Version of the application to delete the log option
 from>
}
```

Erreurs de configuration

Les sections suivantes contiennent des informations détaillées sur les erreurs que vous pourriez rencontrer dans Amazon CloudWatch Logs en raison d'une application mal configurée.

Format du message d'erreur

Les messages d'erreur générés par la mauvaise configuration d'une application s'affichent dans le format suivant.

```
{
"applicationARN": "string",
"applicationVersionId": integer,
"messageType": "ERROR",
"message": "string",
"inputId": "string",
"referenceId": "string",
"errorCode": "string"
"messageSchemaVersion": "integer",
}
```

Les champs d'un message d'erreur contiennent les informations suivantes :

- `applicationARN` : Le nom ARN (Amazon Resource Name) de l'application en cause, par exemple : `arn:aws:kinesisanalytics:us-east-1:112233445566:application/sampleApp`
- `applicationVersionId` : Version de l'application lorsque l'erreur s'est produite. Pour de plus amples informations, veuillez consulter ApplicationDetail.
- `messageType` : le type de message. A l'heure actuelle, ce type peut être uniquement `ERROR`.
- `message` : détails de l'erreur, par exemple :

  ```
  There is a problem related to the configuration of your input. Please check that the
   resource exists, the role has the correct permissions to access the resource and
   that Kinesis Analytics can assume the role provided.
  ```

- `inputId` : ID associé à l'entrée de l'application. Cette valeur ne s'affiche que si cette entrée est la cause de l'erreur. Elle ne s'affiche pas si `referenceId` est renseigné. Pour de plus amples informations, veuillez consulter DescribeApplication.

- `referenceId` : l'ID associé à la source de données de référence de l'application. Cette valeur ne s'affiche que si cette source est la cause de l'erreur. Elle ne s'affiche pas si `inputId` est renseigné. Pour de plus amples informations, veuillez consulter DescribeApplication.
- `errorCode` : l'identificateur de l'erreur. Cet ID est `InputError` ou `ReferenceDataError`.
- `messageSchemaVersion` : une valeur qui spécifie la version du schéma du message, actuellement 1. Vous pouvez consulter cette valeur pour savoir si le schéma du message d'erreur a été mis à jour.

Erreurs

Les erreurs susceptibles d'apparaître dans CloudWatch Logs for sont les suivantes.

La ressource n'existe pas

Si un ARN est spécifié pour un flux d'entrée Kinesis qui n'existe pas, mais que l'ARN est syntaxiquement correct, une erreur similaire à l'exemple suivant est générée.

```
{
"applicationARN": "arn:aws:kinesisanalytics:us-east-1:112233445566:application/
sampleApp",
"applicationVersionId": "5",
 "messageType": "ERROR",
 "message": "There is a problem related to the configuration of your input. Please
 check that the resource exists, the role has the correct permissions to access the
 resource and that Kinesis Analytics can assume the role provided.",
 "inputId":"1.1",
 "errorCode": "InputError",
 "messageSchemaVersion": "1"
}
```

Si une clé de fichier Amazon S3 incorrecte est utilisée pour les données de référence, une erreur similaire à l'exemple suivant est générée.

```
{
 "applicationARN": "arn:aws:kinesisanalytics:us-east-1:112233445566:application/
sampleApp",
 "applicationVersionId": "5",
 "messageType": "ERROR",
 "message": "There is a problem related to the configuration of your reference data.
 Please check that the bucket and the file exist, the role has the correct permissions
 to access these resources and that Kinesis Analytics can assume the role provided.",
```

```
 "referenceId":"1.1",
 "errorCode": "ReferenceDataError",
 "messageSchemaVersion": "1"
}
```

Le rôle n'existe pas

Si un ARN est spécifié pour un rôle d'entrée IAM qui n'existe pas, mais que l'ARN est syntaxiquement correct, une erreur similaire à l'exemple suivant est générée.

```
{
 "applicationARN": "arn:aws:kinesisanalytics:us-east-1:112233445566:application/
sampleApp",
 "applicationVersionId": "5",
 "messageType": "ERROR",
 "message": "There is a problem related to the configuration of your input. Please
 check that the resource exists, the role has the correct permissions to access the
 resource and that Kinesis Analytics can assume the role provided.",
 "inputId":null,
 "errorCode": "InputError",
 "messageSchemaVersion": "1"
}
```

Le rôle n'est pas autorisé à accéder aux ressources

En cas d’utilisation d’un rôle d’entrée qui n’est pas autorisé à accéder aux ressources d’entrée, comme un flux source Kinesis, une erreur similaire à l’exemple suivant est générée.

```
{
 "applicationARN": "arn:aws:kinesisanalytics:us-east-1:112233445566:application/
sampleApp",
 "applicationVersionId": "5",
 "messageType": "ERROR",
 "message": "There is a problem related to the configuration of your input. Please
 check that the resource exists, the role has the correct permissions to access the
 resource and that Kinesis Analytics can assume the role provided.",
 "inputId":null,
 "errorCode": "InputError",
 "messageSchemaVersion": "1"
}
```

Journalisation des appels d’API avec AWS CloudTrail

est intégré avec AWS CloudTrail, un service qui fournit un enregistrement des actions prises par un utilisateur, un rôle ou un service AWS dans . CloudTrail capture les appels d'API vers en tant qu'événements. Les appels capturés incluent des appels de la console et les appels de code vers les opérations d'API . Si vous créez un journal d'activité, vous pouvez activer la livraison continue d'événements CloudTrail à un compartiment Amazon S3, y compris des événements pour . Si vous ne configurez pas de journal de suivi, vous pouvez toujours afficher les événements les plus récents dans la console CloudTrail dans Event history (Historique des événements). Avec les informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été envoyée à l', ainsi que l'adresse IP, l'auteur et date de la demande, ainsi que d'autres détails.

Pour en savoir plus sur CloudTrail, consultez le Guide de l'utilisateur AWS CloudTrail.

Informations dans CloudTrail

CloudTrail est activé dans votre compte AWS lors de sa création. Lorsqu’une activité a lieu dans , cette activité est enregistrée dans un événement CloudTrail avec d’autres AWS événements de service dans Historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS. Pour plus d'informations, veuillez consulter Affichage des événements avec l'historique des événements CloudTrail.

Pour enregistrer en continu les événements dans votre compte AWS, y compris les événements d’, créez un journal d’activité. Un journal d'activité permet à CloudTrail de distribuer les fichiers journaux vers Simple Storage Service (Amazon S3) bucket. Par défaut, lorsque vous créez un journal d'activité dans la console, il s'applique à toutes les régions . Le journal de suivi consigne les événements de toutes les Régions dans la partition AWS et livre les fichiers journaux dans le compartiment Amazon S3 de votre choix. En outre, vous pouvez configurer d'autres services AWS pour analyser et agir sur les données d'événements collectées dans les journaux CloudTrail. Pour en savoir plus, consultez les ressources suivantes :

- Présentation de la création d'un journal d'activité
- Intégrations et services pris en charge par CloudTrail
- Configuration des Notifications de Amazon SNS pour CloudTrail
- Réception des fichiers journaux CloudTrail de plusieurs régions et Réception des fichiers journaux CloudTrail de plusieurs comptes

Toutes les actions sont consignées par CloudTrail et documentées dans la Référence des API. À titre d'exemple, les appels vers les actions `CreateApplication` et `UpdateApplication` génèrent des entrées dans les fichiers journaux CloudTrail.

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 :

- Si la requête a été effectuée avec les informations d'identification utilisateur ou Utilisateur racine d'un compte AWS.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre service AWS.

Pour plus d'informations, consultez Élément userIdentity CloudTrail.

Présentation des entrées des fichiers journaux

Un journal d'activité est une configuration qui permet d'envoyer les événements dans des fichiers journaux à un compartiment Amazon S3 que vous spécifiez. Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs entrées. Un événement représente une demande unique provenant de n'importe quelle source et comprend des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la requête, etc. Les fichiers journaux CloudTrail ne constituent pas une série ordonnée retraçant les appels d'API publics. Ils ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre les actions AddApplicationCloudWatchLoggingOption et DescribeApplication.

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-14T01:03:00Z",
```

```
            "eventSource": "kinesisanalytics.amazonaws.com",
            "eventName": "AddApplicationCloudWatchLoggingOption",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "currentApplicationVersionId": 1,
                "cloudWatchLoggingOption": {
                    "roleARN": "arn:aws:iam::012345678910:role/cloudtrail_test",
                    "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:sql-cloudwatch"
                },
                "applicationName": "cloudtrail-test"
            },
            "responseElements": null,
            "requestID": "e897cd34-45f4-11e9-8912-e52573a36cd9",
            "eventID": "57fe50e9-c764-47c3-a0aa-d0c271fa1cbb",
            "eventType": "AwsApiCall",
            "apiVersion": "2015-08-14",
            "recipientAccountId": "303967445486"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-14T05:37:20Z",
            "eventSource": "kinesisanalytics.amazonaws.com",
            "eventName": "DescribeApplication",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "cloudtrail-test"
            },
            "responseElements": null,
            "requestID": "3b74eb29-461b-11e9-a645-fb677e53d147",
            "eventID": "750d0def-17b6-4c20-ba45-06d9d45e87ee",
            "eventType": "AwsApiCall",
```

```
            "apiVersion": "2015-08-14",
            "recipientAccountId": "012345678910"
        }
    ]
}
```

Limites

Lorsque vous utilisez les applications Amazon Kinesis Data Analytics pour SQL, notez les limites suivantes :

- Kinesis Data Analytics pour SQL est disponible dans les régions AWS suivantes : USA Est (Ohio), USA Est (Virginie du Nord), USA Ouest (Oregon), Canada (Centre), Europe (Paris), Europe (Irlande), Europe (Francfort), Europe (Londres), Asie-Pacifique (Hong Kong), Asie-Pacifique (Mumbai), Asie-Pacifique (Sydney), Asie-Pacifique (Singapour), Asie-Pacifique (Séoul), Asie-Pacifique (Tokyo), Amérique du Sud (São Paulo), AWS GovCloud (USA Est) et AWS GovCloud (USA Ouest). Nous n'avons pas l'intention de lancer Kinesis Data Analytics pour SQL dans d'autres régions AWS.
- Après le 28 juin 2023, vous ne pourrez plus créer de nouvelles applications Kinesis Data Analytics pour SQL à l'aide de la console de gestion AWS si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Si vous avez créé une application Kinesis Data Analytics pour SQL avant le 28 juin 2023, aucun changement n'est apporté à la façon dont vous créez et exécutez les applications aujourd'hui dans une région AWS où vous utilisez déjà Kinesis Data Analytics pour SQL. Toutefois, vous ne pourrez plus créer de nouvelles applications à l'aide de la console AWS dans une région où vous n'utilisez pas Kinesis Data Analytics pour SQL.
- Après le 12 septembre 2023, vous ne pourrez plus créer de nouvelles applications en utilisant Kinesis Data Firehose comme source si vous n'utilisez pas déjà Kinesis Data Analytics pour SQL. Les clients existants qui utilisent les applications Kinesis Data Analytics pour SQL avec `KinesisFirehoseInput` peuvent continuer à ajouter des applications avec `KinesisFirehoseInput` au sein d'un compte existant à l'aide de Kinesis Data Analytics. Si vous êtes déjà client et que vous souhaitez créer un compte avec les applications Kinesis Data Analytics pour SQL avec `KinesisFirehoseInput`, vous pouvez ouvrir un cas de support. Pour plus d'informations, consultez le Centre AWS Support.
- La taille d'une ligne dans un flux intégré à l'application est limitée à 512 Ko. Kinesis Data Analytics utilise jusqu'à 1 Ko pour stocker les métadonnées. Ces métadonnées sont comptabilisées dans la limite de ligne.
- Le code SQL dans une application est limité à 100 Ko.

- La fenêtre la plus longue que nous recommandons pour une requête à fenêtres est d'une heure. Les flux intégrés à l'application sont stockés dans un stockage volatile et des interruptions inattendues de l'application entraînent la reconstruction du flux par cette dernière à partir des données source dans le stockage volatile.

- Le débit le plus élevé que nous recommandons pour un flux intégré à une application unique est compris entre 2 et 20 Mo/s, en fonction de la complexité de la requête de l'application.

- Vous pouvez créer jusqu'à 50 applications Kinesis Data Analytics par région AWS dans votre compte. Vous pouvez demander des applications supplémentaires via le formulaire d'augmentation de limite de service. Pour plus d'informations, consultez le Centre AWS Support.

- Le débit de streaming maximal qu'une seule application Kinesis Data Analytics pour SQL peut traiter est d'environ 100 Mo/sec. Cela suppose que vous avez augmenté le nombre de flux intégrés à l'application jusqu'à la valeur maximale de 64 et que vous avez augmenté votre limite de KPU au-delà de 8 (voir la limite suivante pour plus de détails). Si votre application doit traiter plus de 100 Mo/s d'entrée, procédez de l'une des façons suivantes :
  - Utiliser plusieurs occurrences d'applications Kinesis Data Analytics pour SQL pour traiter l'entrée
  - Utilisez le service géré pour les applications Apache Flink pour Java si vous souhaitez utiliser un seul flux et une seule application.

  > **Note**
  >
  > Nous vous conseillons de revoir régulièrement la métrique `InputProcessing.OkBytes` de votre application afin de pouvoir planifier à l'avance l'utilisation de plusieurs applications SQL ou de migrer vers le service géré pour les applications Apache Flink pour Java si le débit d'entrée prévu de votre application dépasse 100 Mo/sec. Nous vous conseillons également de créer une alarme CloudWatch sur `InputProcessing.OkBytes` afin que vous soyez averti lorsque votre application approche de la limite de débit d'entrée. Cela peut être utile car vous pouvez mettre à jour votre requête d'application pour obtenir un débit plus élevé, évitant ainsi la surcharge et les retards dans les analyses. Pour plus d'informations, consultez Dépannage. L'alarme peut également s'avérer utile si vous disposez d'un mécanisme permettant de réduire le débit en amont.

- Le nombre d'unités de traitement Kinesis (KPU) est limité à huit. Pour trouver des instructions pour demander une augmentation de cette limite, consultez Pour demander une augmentation de limite dans Limites relatives au service Amazon.

  Avec Kinesis Data Analytics, vous ne payez que les ressources que vous utilisez. Vous êtes facturé selon un taux horaire basé sur le nombre moyen de d'unités KPU utilisées pour exécuter votre application de traitement de flux. Une seule unité KPU vous fournit 1 vCPU et 4 Go de mémoire.

- Chaque application peut comporter une source de diffusion et une source de données de référence au maximum.

- Vous pouvez configurer jusqu'à trois destinations pour votre application Kinesis Data Analytics. Nous vous recommandons d'utiliser l'une de ces destinations pour conserver les données du flux d'erreurs intégré à l'application.

- L'objet Amazon S3 qui stocke les données de référence peut avoir une taille de 1 Go maximum.

- Si vous modifiez les données de référence stockées dans le compartiment S3 après les avoir chargées dans une table intégrée à l'application, vous devez utiliser l'opération UpdateApplication (à l'aide de l'API ou de l'AWS CLI) pour actualiser les données de la table intégrée à l'application. Actuellement, AWS Management Console ne prend pas en charge l'actualisation des données de référence dans votre application.

- Actuellement, Kinesis Data Analytics ne prend pas en charge les données générées par l'Amazon Kinesis Producer Library (KPL).

- Vous pouvez attribuer jusqu'à 50 balises par application.

Bonnes pratiques

Cette section décrit les bonnes pratiques pour la gestion d'applications Amazon Kinesis Data Analytics.

Rubriques

Gestion d'applications

Lorsque vous gérez des applications Amazon Kinesis Data Analytics, suivez ces bonnes pratiques :

- Configurez les CloudWatch alarmes Amazon : vous pouvez utiliser les CloudWatch métriques fournies par Kinesis Data Analytics pour surveiller les éléments suivants :
  - Octets d'entrée et enregistrements d'entrée (nombre d'octets et d'enregistrements qui entrent dans l'application)
  - Octets de sortie et enregistrements de sortie
  - `MillisBehindLatest` (retard de l'application pour la lecture de la source de diffusion)

  Nous vous recommandons de configurer au moins deux CloudWatch alarmes sur les métriques suivantes pour vos applications en production :
  - `MillisBehindLatest` – Pour la plupart des cas, nous vous recommandons de définir cette alarme pour se déclencher lorsque votre application a 1 heure de retard par rapport aux données les plus récentes, pour une moyenne d'1 minute. Pour les applications nécessitant moins end-to-end de traitement, vous pouvez régler ce paramètre sur une tolérance inférieure. Cette alarme peut vous aider à garantir que votre application lit les données les plus récentes.

- Pour éviter d'obtenir l'exception `ReadProvisionedThroughputException`, limitez le nombre d'applications de production lisant à partir du même flux de données Kinesis à deux applications.

  > **Note**
  >
  > Dans ce cas, le terme application fait référence à n'importe quelle application pouvant lire à partir de la source de diffusion. Seule une application Kinesis Data Analytics peut lire un flux de diffusion Firehose. Cependant, de nombreuses applications peuvent lire à partir d'un flux de données Kinesis, comme une application Kinesis Data Analytics ou. AWS Lambda La limite de nombre d'applications recommandée fait référence à toutes les applications que vous configurez pour lire à partir d'une source de diffusion.

  Amazon Kinesis Data Analytics lit une source de streaming environ une fois par seconde par application. Toutefois, une application qui prend du retard peut lire des données à un rythme plus rapide pour rattraper son retard. Pour autoriser un débit adéquat permettant aux applications de rattraper leur retard, limitez le nombre d'applications lisant la même source de données.

- Limitez le nombre d'applications de production lisant le même flux de diffusion Firehose à une seule application.

  Un flux de diffusion Firehose peut écrire vers des destinations telles qu'Amazon S3 et Amazon Redshift. Il peut également tenir lieu de source de streaming pour votre application Kinesis Data Analytics. Par conséquent, nous vous recommandons de ne pas configurer plusieurs applications Kinesis Data Analytics par flux de diffusion Firehose. Vous vous assurez ainsi que le flux de diffusion peut également diffuser vers d'autres destinations.

Mise à l'échelle des applications

Configurez votre application en fonction de l'évolution de vos besoins en augmentant de manière proactive le nombre de flux intégrés à l'application d'entrée sur la base de la valeur par défaut (un). Il est recommandé d'utiliser les options de langues suivantes en fonction du débit de votre application :

- Utilisez plusieurs flux et applications Kinesis Data Analytics pour SQL si les besoins de mise à l'échelle de votre application sont supérieurs à 100 Mo/seconde.
- Utilisez le service géré pour les applications Apache Flink si vous souhaitez utiliser un seul flux et une seule application.

> **Note**
>
> Nous vous conseillons de revoir régulièrement la métrique `InputProcessing.0kBytes` de votre application afin de pouvoir planifier à l'avance l'utilisation de plusieurs applications SQL ou de migrer vers managed-flink/latest/java/ si le débit d'entrée prévu de votre application dépasse 100 Mo/s.

Surveillance des applications

Nous vous conseillons de créer une CloudWatch alarme `InputProcessing.0kBytes` afin que vous soyez averti lorsque votre application approche de la limite de débit d'entrée. Cela peut être utile car vous pouvez mettre à jour votre requête d'application pour obtenir un débit plus élevé, évitant ainsi la surcharge et les retards dans les analyses. Pour plus d'informations, consultez Dépannage. Cela peut également s'avérer utile si vous disposez d'un mécanisme permettant de réduire le débit en amont.

- Le débit le plus élevé que nous recommandons pour un flux intégré à une application unique est compris entre 2 et 20 Mo/s, en fonction de la complexité de la requête de l'application.
- Le débit de streaming maximal qu'une seule application Kinesis Data Analytics pour SQL peut traiter est d'environ 100 Mo/s. Cela suppose que vous avez augmenté le nombre de flux intégrés à l'application jusqu'à la valeur maximale de 64 et que vous avez augmenté votre limite de KPU au-delà de 8. Pour plus d'informations, consultez Limites .

> **Note**
>
> Nous vous conseillons de revoir régulièrement la métrique `InputProcessing.0kBytes` de votre application afin de pouvoir planifier à l'avance l'utilisation de plusieurs applications SQL ou de migrer vers managed-flink/latest/java/ si le débit d'entrée prévu de votre application dépasse 100 Mo/s.

Définition d'un schéma d'entrée

Lorsque vous configurez l'entrée d'application dans la console, vous spécifiez tout d'abord une source de diffusion. La console utilise ensuite l'API de découverte (voir DiscoverInputSchema) pour déduire un schéma en prélevant des enregistrements sur la source de diffusion. Le schéma, entre autres, définit les noms et les types de données des colonnes dans le flux intégré à l'application résultant. La console affiche le schéma. Nous vous recommandons de procéder comme suit avec ce schéma déduit :

- Testez le schéma déduit de façon appropriée. Le processus de découverte utilise uniquement un échantillon d'enregistrements de la source de diffusion pour déduire un schéma. Si votre source de diffusion comporte de nombreux types d'enregistrement, l'API de découverte peut ne pas avoir prélevé un ou plusieurs types d'enregistrement. Cette situation peut se traduire par un schéma qui ne reflète pas exactement les données de la source de diffusion.

  Lorsque votre application démarre, ces types d'enregistrements manqués peuvent se traduire par des erreurs d'analyse. Amazon Kinesis Data Analytics envoie ces enregistrements au flux d'erreurs intégré à l'application. Pour réduire ces erreurs d'analyse, nous vous recommandons de tester interactivement le schéma déduit dans la console et de surveiller le flux intégré à l'application pour guetter les enregistrements manqués.

- L'API Kinesis Data Analytics ne prend pas en charge la spécification de la contrainte `NOT NULL` sur les colonnes dans la configuration d'entrée. Si vous souhaitez appliquer des contraintes `NOT NULL` sur des colonnes de votre flux intégré à l'application, créez ces flux à l'aide de votre code d'application. Vous pouvez ensuite copier les données d'un flux intégré à l'application vers un autre ; la contrainte est alors appliquée.

  Toute tentative d'insertion de lignes avec des valeurs `NULL` lorsqu'une valeur est obligatoire entraîne une erreur. Kinesis Data Analytics envoie ces erreurs au flux d'erreurs intégré à l'application.

- Assouplissez les types de données déduits par le processus de découverte. Le processus de découverte recommande des colonnes et des types de données basés sur un échantillonnage d'enregistrements de la source de diffusion. Nous vous recommandons de vérifier avec soin ces types de données et d'envisager de les assouplir pour couvrir tous les cas d'enregistrements possibles dans votre entrée. Cela permet de réduire les erreurs d'analyse dans l'application

pendant son exécution. Par exemple, si un schéma déduit `SMALLINT` comme type de colonne, vous pouvez peut-être envisager de le remplacer par un type `INTEGER`.

- Utilisez des fonctions SQL dans votre code d'application pour traiter les données ou les colonnes non structurées. Vous pouvez avoir des données ou des colonnes non structurées, comme des données de journal, dans votre entrée. Pour obtenir des exemples, consultez Exemple : transformation DateTime des valeurs. Une approche pour traiter ce type de données consiste à définir le schéma avec une seule colonne de type `VARCHAR(N)`, où `N` est la plus grande ligne possible que vous vous attendez à voir dans votre flux. Dans votre code d'application, vous pouvez ensuite lire les enregistrements entrants, puis utiliser les fonctions `String` et `Date Time` pour analyser et schématiser les données brutes.

- Veillez à traiter complètement des données de source de diffusion contenant une imbrication de plus de deux niveaux. Lorsque les données source sont au format JSON, vous pouvez avoir une imbrication. L'API de découverte déduit un schéma qui aplatit un niveau d'imbrication. Pour deux niveaux d'imbrication, l'API de découverte essaie également d'aplatir ces niveaux. Au-delà de deux niveaux d'imbrication, la prise en charge de l'aplatissement est limitée. Pour pouvoir traiter complètement l'imbrication, vous devez modifier manuellement le schéma selon vos besoins. Pour ce faire, utilisez l'une des stratégies suivantes :

  - Utilisez le chemin de ligne JSON pour extraire de façon sélective les paires clé-valeur requises pour votre application. Un chemin de ligne JSON fournit un pointeur vers la paire clé-valeur spécifique que vous souhaitez insérer dans votre application. Vous pouvez effectuer ceci pour n'importe quel niveau de l'imbrication.
  - Utilisez le chemin de ligne JSON pour extraire de façon sélective des objets JSON complexes, puis utilisez des fonctions de manipulation de chaîne dans votre code d'application pour extraire les données spécifiques dont vous avez besoin.

Connexion à des sorties

Nous recommandons que toutes les applications disposent d'au moins deux sorties:

- Utilisez la première destination pour insérer les résultats de vos requêtes SQL.

- Utilisez la deuxième destination pour insérer l'intégralité du flux d'erreurs et l'envoyer vers un compartiment S3 via un flux de diffusion Firehose.

Création de code d'application

Nous vous recommandons la procédure suivante :

- Dans votre instruction SQL, ne spécifiez pas une fenêtre temporelle plus longue qu'une heure pour les raisons suivantes :
  - Parfois, une application doit être redémarrée, parce que vous l'avez mise à jour ou pour des raisons internes à Kinesis Data Analytics. Lorsqu'elle redémarre, toutes les données incluses dans la fenêtre doivent être lues à nouveau à partir de la source de données de diffusion. Cela prend du temps avant que Kinesis Data Analytics puisse émettre une sortie pour cette fenêtre.
  - Kinesis Data Analytics doit gérer tout ce qui concerne l'état de l'application, y compris des données pertinentes pour la durée. Cela consomme d'importantes unités de traitement Kinesis Data Analytics.
- Lors du développement, conservez une durée de fenêtre réduite dans vos requêtes SQL pour pouvoir voir les résultats plus rapidement. Lorsque vous déployez l'application vers votre environnement de production, vous pouvez définir la durée de la fenêtre de façon appropriée.
- Au lieu d'utiliser une seule instruction SQL complexe, envisagez de diviser celle-ci en plusieurs instructions, en enregistrant les résultats dans des flux intégrés à l'application intermédiaires. Cela peut vous aider à déboguer plus rapidement.
- Lorsque vous utilisez des fenêtres bascules, nous vous recommandons d'utiliser deux fenêtres, une pour l'heure de traitement et une pour l'heure logique (heure d'intégration ou heure de l'événement). Pour plus d'informations, consultez Horodatages et colonne ROWTIME.

Test des applications

Lorsque vous modifiez le schéma ou le code d'application pour votre application Kinesis Data Analytics, nous vous recommandons d'utiliser une application de test pour vérifier vos modifications avant de les déployer en production.

Configuration d'une application de test

Vous pouvez configurer une application de test via la console ou à l'aide d'un modèle AWS CloudFormation . L'utilisation d'un AWS CloudFormation modèle permet de garantir que les modifications de code que vous apportez à l'application de test et à votre application en ligne sont cohérentes.

Lorsque vous configurez une application de test, vous pouvez connecter l'application à vos données en temps réel ou remplir un flux avec des données fictives pour les tests. Nous vous recommandons deux méthodes pour remplir un flux avec des données fictives :

- Utilisez le Kinesis Data Generator (KDG). Le générateur KDG utilise un modèle de données pour envoyer des données aléatoires à un flux Kinesis. Le générateur KDG est simple à utiliser, mais il n'est pas adapté pour tester des relations complexes entre des éléments de données, comme pour les applications qui détectent les points chauds ou les anomalies dans les données.
- Utilisez une application Python personnalisée pour envoyer les données plus complexes dans un flux de données Kinesis. Une application Python peut générer des relations complexes entre des éléments de données, comme des points chaud ou des anomalies. Pour obtenir un exemple d'application Python qui envoie les données regroupées dans un point chaud de données, consultez Exemple : Détection des points chauds sur un flux (fonction HOTSPOTS).

Lorsque vous exécutez votre application de test, visualisez vos résultats à l'aide d'une destination (telle qu'un flux de diffusion Firehose vers une base de données Amazon Redshift) au lieu de consulter votre flux intégré à l'application sur la console. Les données qui s'affichent dans la console constituent un échantillonnage du flux et ne contiennent pas tous les enregistrements.

Test des modifications de schéma

Lorsque vous modifiez le schéma du flux d'entrée d'une application, utilisez votre application de test pour vérifier que les conditions suivantes sont réunies :

- Les données de votre flux sont converties dans le type de données approprié. Par exemple, assurez-vous que les données datetime ne sont pas insérées dans l'application sous la forme d'une chaîne.
- Les données sont analysées et converties dans le type de données que vous souhaitez. Si des erreurs d'analyse ou de conversion se produisent, vous pouvez les afficher dans la console ou attribuer une destination au flux d'erreurs et afficher les erreurs dans la destination.

- Les champs de données pour les données de type caractère ont une longueur suffisante et l'application ne tronque pas ces données. Vous pouvez contrôler les enregistrements de données dans votre magasin de destination pour vérifier que les données de votre application ne sont pas tronquées.

Test des modifications de code

Le test des modifications apportées à votre code SQL requiert des connaissances du domaine de votre application. Vous devez pouvoir déterminer quelle sortie doit être testée et comment une sortie correcte doit se présenter. Pour connaître les zones à problème potentiel à vérifier lors de la modification du code SQL de votre application, consultez Résolution des problèmes liés aux applications Amazon Kinesis Data Analytics pour SQL.

Résolution des problèmes liés aux applications Amazon Kinesis Data Analytics pour SQL

La documentation suivante peut vous aider à résoudre les problèmes que vous pouvez rencontrer avec les applications Amazon Kinesis Data Analytics pour SQL.

Rubriques

Applications à l’arrêt

- Qu’est-ce qu’une application Kinesis Data Analytics pour SQL à l’arrêt ?

  Une application à l’arrêt est une application dont nous avons constaté qu’elle ne traitait aucun enregistrement sur un période minimum de trois mois. Cela signifie que les clients payent pour des ressources Kinesis Data Analytics pour SQL qu’ils n’utilisent pas.

- Quand AWS commencera-t-il à arrêter les applications inactives ?

  AWS commencera à arrêter les applications inactives le 14 2023 et ce jusqu’au 21 novembre 2023. Nous arrêterons les applications inactives dans le fuseau horaire des heures de bureau de la région concernée.

- Les applications Kinesis Data Analytics pour SQL à l’arrêt peuvent-elles être redémarrées ?

  Oui. Si vous avez besoin de redémarrer votre application, vous pouvez le faire normalement. Il n’est pas nécessaire de soumettre un ticket de support.

- Si AWS arrête une application inactive, certains des résultats de ma requête seront-ils également supprimés ?

  Non. Premièrement, votre application étant inactive, elle ne traite pas les requêtes. Deuxièmement, les résultats de vos requêtes ne sont pas stockés dans Kinesis Data Analytics pour SQL. Vous configurez votre application Kinesis Data Analytics pour SQL avec une destination réceptrice vers laquelle les résultats de ses calculs sont envoyés (par exemple, dans Amazon S3 ou un autre flux de données). Ainsi, vous conservez l'entière propriété de vos données et celles-ci resteront récupérables selon les conditions de ce service de stockage.

- Que dois-je faire si je ne souhaite pas que mon application soit arrêtée ?

  Vous pouvez envoyer un e-mail à l'équipe de service (kda-sql-questions@amazon.com) pour demander que les applications ne soient pas arrêtées, avant le 10 novembre 2023. L'e-mail doit inclure votre ID de compte et l'ARN de votre application.

Impossible d'exécuter le code SQL

Si vous avez besoin de savoir comment obtenir une instruction SQL particulière qui fonctionne correctement, vous disposez de plusieurs ressources lorsque vous utilisez Kinesis Data Analytics :

- Pour plus d'informations sur les instructions SQL, consultez Exemples de Kinesis Data Analytics pour SQL. Cette section fournit plusieurs exemples SQL que vous pouvez utiliser.
- La référence SQL Amazon Kinesis Data Analytics fournit un guide détaillé sur la création d'instructions SQL de streaming.
- Si vous continuez à rencontrer des problèmes, nous vous recommandons de poser une question sur les Forums Kinesis Data Analytics.

Impossible de détecter ou de découvrir mon schéma

Dans certains cas, Kinesis Data Analytics ne peut pas détecter ou découvrir un schéma. Dans nombre de ces cas, vous pouvez tout de même utiliser Kinesis Data Analytics.

Supposons que vous disposez de données codées en UTF-8 qui n'utilisent pas de séparateur ou de données qui utilisent un format autre que le format CSV (valeurs séparées par des virgules), ou que l'API de découverte ne découvre pas votre schéma. Vous pouvez alors définir un schéma manuellement ou utiliser des fonctions de manipulation des chaînes pour structurer vos données.

Pour découvrir le schéma de votre flux, Kinesis Data Analytics échantillonne de façon aléatoire les données les plus récentes de votre flux. Si vous n'envoyez pas constamment des données à votre flux, il est possible que Kinesis Data Analytics ne puisse pas extraire d'échantillon et détecter de schéma. Pour de plus amples informations, veuillez consulter Utilisation de la fonction de découverte de schéma sur des données de diffusion.

Les données de référence sont obsolètes

Les données de référence sont chargées à partir de l'objet Amazon Simple Storage Service (Amazon S3) dans l'application lorsque celle-ci est lancée ou mise à jour, ou lors d'interruptions de l'application qui sont provoquées par des problèmes de service.

Les données de référence ne sont pas chargées dans l'application lorsque des mises à jour sont apportées à l'objet Amazon S3 sous-jacent.

Si les données de référence de l'application ne sont pas à jour, vous pouvez les recharger en suivant les étapes ci-dessous :

1. Dans la console Kinesis Data Analytics, choisissez le nom de l'application dans la liste, puis sélectionnez Détails de l'application.
2. Choisissez Go to SQL editor (Accéder à l'éditeur SQL) pour ouvrir la page Real-time analytics (Analyse en temps réel) pour l'application.
3. Dans la vue Source Data (Source de données), choisissez le nom de la table de données de référence.
4. Choisissez Actions, Synchronize reference data table (Synchroniser la table de données de référence).

L'application n'écrit pas vers la destination

Si les données ne sont pas écrites vers la destination, vérifiez les points suivants :

- Vérifiez que le rôle de l'application dispose de suffisamment d'autorisations pour accéder à la destination. Pour plus d'informations, consultez Stratégie d'autorisations pour l'écriture dans un flux Kinesis ou Stratégie d'autorisations pour l'écriture dans un flux de diffusion Firehose.
- Vérifiez que la destination de l'application est correctement configurée et que l'application utilise le bon nom de flux de sortie.

- Vérifiez les métriques Amazon CloudWatch pour votre flux de sortie pour voir si les données sont en cours d'écriture. Pour obtenir plus d'informations sur l'utilisation des métriques CloudWatch, consultez Surveillance avec Amazon CloudWatch.
- Ajoutez un flux de journaux CloudWatch à l'aide de the section called "AddApplicationCloudWatchLoggingOption". Votre application écrit les erreurs de configuration dans le flux de journaux.

Si le rôle et la configuration de la destination semblent corrects, essayez de redémarrer l'application en spécifiant `LAST_STOPPED_POINT` pour InputStartingPositionConfiguration.

Importants paramètres de l'état des applications à surveiller

Pour vous assurer que votre application s'exécute correctement, nous vous recommandons de surveiller certains paramètres importants.

Le paramètre le plus important à surveiller est la métrique Amazon CloudWatch `MillisBehindLatest`. Cette métrique représente votre retard dans la lecture du flux par rapport à l'heure actuelle. Elle vous aide à déterminer si vous traitez les enregistrements du flux source suffisamment vite.

En règle générale, vous devez configurer une alarme CloudWatch qui se déclenche si votre retard dépasse une heure. Toutefois, la durée dépend de votre cas d'utilisation. Vous pouvez l'ajuster selon vos besoins.

Pour de plus amples informations, veuillez consulter Bonnes pratiques.

Erreur de code non valide lors de l'exécution d'une application

Lorsque vous ne pouvez pas enregistrer et exécuter le code SQL de votre application Amazon Kinesis Data Analytics, les causes les plus courantes sont les suivantes :

- Le flux a été redéfini dans votre code SQL : Après avoir créé un flux et la pompe associée au flux, vous ne pouvez pas redéfinir le même flux dans votre code. Pour plus d'informations sur la création d'un flux, consultez CRÉER UN FLUX dans le manuel Référence SQL Amazon Kinesis Data Analytics. Pour plus d'informations sur la création d'une pompe, consultez CREATE PUMP.
- Une clause GROUP BY utilise plusieurs colonnes ROWTIME : Vous pouvez spécifier une seule colonne ROWTIME dans la clause GROUP BY. Pour plus d'informations, consultez GROUP BY et ROWTIME dans le manuel Référence SQL Amazon Kinesis Data Analytics.

- Un ou plusieurs types de données ont une conversion non valide : Dans ce cas, votre code a une conversion implicite non valide. Par exemple, vous pouvez convertir un type `timestamp` en type `bigint` dans votre code.
- Un flux a le même nom que le flux réservé d'un service : Un flux ne peut pas avoir le même nom que le flux réservé d'un service `error_stream`.

L'application écrit des erreurs dans le flux d'erreurs

Si votre application écrit des erreurs dans le flux d'erreurs intégré à l'application, vous pouvez décoder la valeur dans le champ `DATA_ROW` à l'aide de bibliothèques standard. Pour plus d'informations sur le flux d'erreur, consultez Gestion des erreurs.

Débit insuffisant ou valeur de MillisBehindLatest élevée

Si la métrique MillisBehindLatest de votre application augmente sans cesse ou est constamment supérieure à 1000 (une seconde), cela peut être due aux raisons suivantes :

- Vérifiez la métrique CloudWatch InputBytes de votre application. Si vous ingérez plus de 4 Mo/s, cela peut entraîner une augmentation de `MillisBehindLatest`. Pour améliorer le débit de votre application, augmentez la valeur du paramètre `InputParallelism`. Pour de plus amples informations, veuillez consulter Mise en parallèle des flux d'entrée pour un débit accru.
- Vérifiez la métrique de réussite de diffusion de sortie de votre application pour détecter les échecs de diffusion vers votre destination. Vérifiez que vous avez correctement configuré la sortie et que votre flux de sortie a une capacité suffisante.
- Si votre application utilise une fonction AWS Lambda pour le prétraitement ou en tant que sortie, vérifiez la métrique CloudWatch InputProcessing.Duration ou LambdaDelivery.Duration de l'application. Si la durée d'invocation de la fonction Lambda est supérieure à 5 secondes, vous pouvez envisager les opérations suivantes :
  - Augmentez l'allocation de Mémoire de la fonction Lambda. Vous pouvez effectuer cette opération dans la console AWS Lambda, sur la page Configuration, sous Basic settings (Paramètres de base). Pour plus d'informations, consultez Configuration des fonctions Lambda dans le Manuel du développeur AWS Lambda.
  - Augmentez le nombre de partitions dans votre flux d'entrée de l'application. Cela augmente le nombre de fonctions parallèles appelées par l'application, ce qui peut augmenter le débit.

  - Vérifiez que la fonction n'effectue pas des appels bloquants qui ont un impact sur les performances, comme des demandes synchrones de ressources externes.
  - Examinez votre fonction AWS Lambda à la recherche d'autres domaines dans lesquels vous pouvez améliorer les performances. Vérifiez les journaux CloudWatch de la fonction Lambda de l'application. Pour plus d'informations, consultez Accès aux métriques Amazon CloudWatch pour dans le Manuel du développeur AWS Lambda.
- Vérifiez que votre application n'atteint pas la limite par défaut d'unités de traitement Kinesis (KPU). Si votre application atteint cette limite, vous pouvez demander une augmentation de limite. Pour de plus amples informations, veuillez consulter Dimensionnement automatique des applications pour augmenter le débit.
- Si votre application rencontre toujours des problèmes après l'augmentation de votre limite de KPU, vérifiez que le débit d'entrée de votre application ne dépasse pas 100 Mo/sec. S'il dépasse 100 Mo/sec, nous recommandons de mettre en œuvre des modifications visant à réduire le débit global afin de stabiliser l'application, par exemple en réduisant la quantité de données envoyées à la source de données depuis laquelle l'application Kinesis Data Analytics Sql lit. Nous recommandons également d'autres approches, notamment l'augmentation du parallélisme de l'application, la réduction de la durée des calculs, le remplacement des types de données en colonnes de VARCHAR par des types de données de plus petite taille (par exemple, INTEGER, LONG, etc.) et la réduction des données traitées par échantillonnage ou filtrage.

> **Note**
>
> Nous vous conseillons de revoir régulièrement la métrique `InputProcessing.OkBytes` de votre application afin de pouvoir planifier à l'avance l'utilisation de plusieurs applications SQL ou de migrer vers managed-flink/latest/java/ si le débit d'entrée prévu de votre application dépasse 100 Mo/sec.

Référence SQL Kinesis Data Analytics

Pour plus d'informations sur les éléments du langage SQL pris en charge par Kinesis Data Analytics, consultez le manuel Référence SQL Kinesis Data Analytics.

Référence d'API

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation API Service géré Amazon pour Apache Flink V2.

Vous pouvez utiliser le AWS CLI pour explorer l'API Amazon Kinesis Data Analytics. Ce guide fournit des exercices Mise en route avec les applications Amazon Kinesis Data Analytics pour SQL qui utilisent l' AWS CLI.

Rubriques

Actions

Les actions suivantes sont prises en charge :

- DescribeApplication
- DiscoverInputSchema
- ListApplications
- ListTagsForResource
- StartApplication
- StopApplication
- TagResource
- UntagResource
- UpdateApplication

AddApplicationCloudWatchLoggingOption

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Ajoute un flux de CloudWatch journal pour surveiller les erreurs de configuration des applications. Pour plus d'informations sur l'utilisation des flux de CloudWatch journaux avec les applications Amazon Kinesis Analytics, consultez la section Travailler avec CloudWatch Amazon Logs.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "string",
      "RoleARN": "string"
   },
   "CurrentApplicationVersionId": number
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application Kinesis Analytics.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CloudWatchLoggingOption

Fournit le nom de ressource Amazon (ARN) du flux de CloudWatch journal et l'ARN du rôle IAM. Remarque : Pour écrire des messages d'application dans le rôle IAM utilisé CloudWatch, l'action de `PutLogEvents` stratégie doit être activée.

Type : objet CloudWatchLoggingOption

Obligatoire : oui

CurrentApplicationVersionId

ID de version de l’application Kinesis Analytics.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

Éléments de réponse

Si l’action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d’une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d’état HTTP : 400

InvalidArgumentException

La valeur du paramètre d’entrée spécifiée n’est pas valide.

Code d’état HTTP : 400

ResourceInUseException

L’application n’est pas disponible pour cette opération.

Code d’état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

AddApplicationInput

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Ajoute une source de streaming à votre application Amazon Kinesis. Pour obtenir des informations conceptuelles, veuillez consulter Configuration de l'entrée de l'application.

Vous pouvez ajouter une source de streaming lorsque vous créez une application ou utiliser cette opération pour ajouter une source de streaming après avoir créé une application. Pour plus d'informations, consultez CreateApplication.

Toutes les mises à jour de la configuration, y compris l'ajout d'une source de streaming à l'aide de cette opération, débouchent sur une nouvelle version de l'application. Vous pouvez utiliser cette DescribeApplicationopération pour trouver la version actuelle de l'application.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:AddApplicationInput`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "Input": {
      "InputParallelism": {
         "Count": number
      },
      "InputProcessingConfiguration": {
         "InputLambdaProcessor": {
            "ResourceARN": "string",
            "RoleARN": "string"
         }
      },
      "InputSchema": {
```

```
            "RecordColumns": [
               {
                  "Mapping": "string",
                  "Name": "string",
                  "SqlType": "string"
               }
            ],
            "RecordEncoding": "string",
            "RecordFormat": {
               "MappingParameters": {
                  "CSVMappingParameters": {
                     "RecordColumnDelimiter": "string",
                     "RecordRowDelimiter": "string"
                  },
                  "JSONMappingParameters": {
                     "RecordRowPath": "string"
                  }
               },
               "RecordFormatType": "string"
            }
         },
         "KinesisFirehoseInput": {
            "ResourceARN": "string",
            "RoleARN": "string"
         },
         "KinesisStreamsInput": {
            "ResourceARN": "string",
            "RoleARN": "string"
         },
         "NamePrefix": "string"
      }
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de votre application Amazon Kinesis Analytics existante à laquelle vous souhaitez ajouter la source de streaming.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

Version actuelle de votre application Amazon Kinesis Analytics. Vous pouvez utiliser cette DescribeApplicationopération pour trouver la version actuelle de l'application.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

Input

L'entrée à ajouter.

Type : objet Input

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

CodeValidationException

Le code d'application (requête) fourni par l'utilisateur n'est pas valide. Il peut s'agir d'une simple erreur de syntaxe.

Code d'état HTTP : 400

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

AddApplicationInputProcessingConfiguration

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Ajoute un InputProcessingConfigurationà une application. Un processeur d'entrée prétraite les enregistrements sur le flux d'entrée avant l'exécution du code SQL de l'application. Actuellement, le seul processeur d'entrée disponible est AWS Lambda.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   }
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application à laquelle vous souhaitez ajouter la configuration de traitement des entrées.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

Version de l'application à laquelle vous souhaitez ajouter la configuration de traitement des entrées. Vous pouvez utiliser cette DescribeApplicationopération pour obtenir la version actuelle de l'application. Si la version spécifiée n'est pas la version actuelle, `ConcurrentModificationException` est renvoyé.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

InputId

ID de la configuration d'entrée à laquelle ajouter la configuration de traitement d'entrée. Vous pouvez obtenir une liste des identifiants d'entrée pour une application à l'aide de cette DescribeApplicationopération.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

InputProcessingConfiguration

InputProcessingConfigurationÀ ajouter à l'application.

Type : objet InputProcessingConfiguration

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2

- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

AddApplicationOutput

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Ajoute une destination externe à votre application Amazon Kinesis Analytics.

Si vous souhaitez qu'Amazon Kinesis Analytics diffuse les données d'un flux intégré à l'application vers une destination externe (comme un flux Amazon Kinesis, un flux de diffusion Amazon Kinesis Firehose ou une fonction AWS Lambda), vous devez ajouter la configuration appropriée à votre application à l'aide de cette opération. Vous pouvez configurer une ou plusieurs sorties pour votre application. Chaque configuration de sortie mappe un flux intégré à l'application et une destination externe.

Vous pouvez utiliser l'une des configurations de sortie pour diffuser des données depuis votre flux d'erreurs intégré à l'application vers une destination externe, ce qui vous permet d'analyser les erreurs. Pour de plus amples informations, veuillez consulter Présentation de la sortie d'application (Destination).

Toutes les mises à jour de la configuration, y compris l'ajout d'une source de streaming à l'aide de cette opération, débouchent sur une nouvelle version de l'application. Vous pouvez utiliser cette DescribeApplicationopération pour trouver la version actuelle de l'application.

Pour connaître les restrictions sur le nombre d'entrées et de sorties d'applications que vous pouvez configurer, veuillez consulter Restrictions.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:AddApplicationOutput`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
```

```
    "Output": {
       "DestinationSchema": {
          "RecordFormatType": "string"
       },
       "KinesisFirehoseOutput": {
          "ResourceARN": "string",
          "RoleARN": "string"
       },
       "KinesisStreamsOutput": {
          "ResourceARN": "string",
          "RoleARN": "string"
       },
       "LambdaOutput": {
          "ResourceARN": "string",
          "RoleARN": "string"
       },
       "Name": "string"
    }
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application à laquelle vous souhaitez ajouter la configuration de sortie.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

Version de l'application à laquelle vous souhaitez ajouter la configuration de sortie. Vous pouvez utiliser cette DescribeApplicationopération pour obtenir la version actuelle de l'application. Si la version spécifiée n'est pas la version actuelle, `ConcurrentModificationException` est renvoyé.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

Output

Tableau d'objets, chacun décrivant une configuration de sortie. Dans la configuration de sortie, vous spécifiez le nom d'un flux intégré à l'application, d'une destination (c'est-à-dire un flux Amazon Kinesis, un flux de diffusion Amazon Kinesis Firehose ou AWS une fonction Lambda), et vous enregistrez la formation à utiliser lors de l'écriture vers la destination.

Type : objet Output

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

AddApplicationReferenceDataSource

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Ajoute une source de données de référence à une application existante.

Amazon Kinesis Analytics lit les données de référence (un objet Amazon S3) et crée une table intégrée à l'application dans votre application. Dans la demande, vous fournissez la source (le nom du compartiment S3 et le nom de la clé d'objet), le nom de la table intégrée à l'application à créer et les informations de mappage nécessaires qui décrivent la façon dont les données d'objet Amazon S3 sont mappées aux colonnes de la table obtenue intégrée à l'application.

Pour obtenir des informations conceptuelles, veuillez consulter Configuration de l'entrée de l'application. Pour connaître les restrictions sur les sources de données que vous pouvez configurer, veuillez consulter Restrictions.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:AddApplicationOutput`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceDataSource": {
      "ReferenceSchema": {
         "RecordColumns": [
            {
               "Mapping": "string",
               "Name": "string",
               "SqlType": "string"
            }
         ],
         "RecordEncoding": "string",
```

```
         "RecordFormat": {
            "MappingParameters": {
               "CSVMappingParameters": {
                  "RecordColumnDelimiter": "string",
                  "RecordRowDelimiter": "string"
               },
               "JSONMappingParameters": {
                  "RecordRowPath": "string"
               }
            },
            "RecordFormatType": "string"
         }
      },
      "S3ReferenceDataSource": {
         "BucketARN": "string",
         "FileKey": "string",
         "ReferenceRoleARN": "string"
      },
      "TableName": "string"
   }
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom d'une application existante.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

Version de l’application pour laquelle vous ajoutez la source de données de référence. Vous pouvez utiliser cette DescribeApplicationopération pour obtenir la version actuelle de l'application. Si la version spécifiée n’est pas la version actuelle, `ConcurrentModificationException` est renvoyé.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

ReferenceDataSource

Source des données de référence pouvant être un objet dans votre compartiment Amazon S3. Amazon Kinesis Analytics lit l'objet et copie les données dans la table intégrée à l'application qui est créée. Vous fournissez un compartiment S3, une clé d'objet et le nom de la table obtenue intégrée à l'application qui est créée. Vous devez également fournir un rôle IAM disposant des autorisations requises qu'Amazon Kinesis Analytics peut endosser pour lire l'objet de votre compartiment S3 en votre nom.

Type : objet ReferenceDataSource

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

CreateApplication

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Crée une application Amazon Kinesis Analytics. Vous pouvez configurer chaque application avec une source de streaming en entrée, un code d'application pour traiter l'entrée et jusqu'à trois destinations dans lesquelles vous souhaitez qu'Amazon Kinesis Analytics écrive les données de sortie de votre application. Pour un aperçu, consultez la section Fonctionnement.

Dans la configuration d'entrée, vous mappez la source de streaming à un flux intégré à l'application, que vous pouvez considérer comme une table constamment mise à jour. Dans le mappage, vous devez fournir un schéma du flux intégré à l'application et mapper chaque colonne de données du flux intégré à un élément de données de la source de streaming.

Le code de votre application correspond à une ou plusieurs instructions SQL qui lisent les données d'entrée, les transforment et génèrent une sortie. Le code de votre application peut créer un ou plusieurs artefacts SQL tels que des flux ou des pompes SQL.

Dans la configuration de sortie, vous pouvez configurer l'application pour écrire des données à partir de flux intégrés à l'application créés dans vos applications vers un maximum de trois destinations.

Pour lire les données de votre flux source ou écrire des données dans des flux de destination, Amazon Kinesis Analytics a besoin de vos autorisations. Vous pouvez accorder ces autorisations en créant des rôles IAM. Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:CreateApplication`.

Pour obtenir des exercices d'introduction à la création d'une application Amazon Kinesis Analytics, consultez Mise en route.

Syntaxe de la requête

```
{
   "ApplicationCode": "string",
   "ApplicationDescription": "string",
```

```
    "ApplicationName": "string",
    "CloudWatchLoggingOptions": [
       {
          "LogStreamARN": "string",
          "RoleARN": "string"
       }
    ],
    "Inputs": [
       {
          "InputParallelism": {
             "Count": number
          },
          "InputProcessingConfiguration": {
             "InputLambdaProcessor": {
                "ResourceARN": "string",
                "RoleARN": "string"
             }
          },
          "InputSchema": {
             "RecordColumns": [
                {
                   "Mapping": "string",
                   "Name": "string",
                   "SqlType": "string"
                }
             ],
             "RecordEncoding": "string",
             "RecordFormat": {
                "MappingParameters": {
                   "CSVMappingParameters": {
                      "RecordColumnDelimiter": "string",
                      "RecordRowDelimiter": "string"
                   },
                   "JSONMappingParameters": {
                      "RecordRowPath": "string"
                   }
                },
                "RecordFormatType": "string"
             }
          },
          "KinesisFirehoseInput": {
             "ResourceARN": "string",
             "RoleARN": "string"
          },
```

```
            "KinesisStreamsInput": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "NamePrefix": "string"
         }
      ],
      "Outputs": [
         {
            "DestinationSchema": {
               "RecordFormatType": "string"
            },
            "KinesisFirehoseOutput": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "KinesisStreamsOutput": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "LambdaOutput": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "Name": "string"
         }
      ],
      "Tags": [
         {
            "Key": "string",
            "Value": "string"
         }
      ]
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationCode

Une ou plusieurs instructions SQL qui lisent les données d'entrée, les transforment et génèrent une sortie. Par exemple, vous pouvez écrire une instruction SQL qui lit les données d'un flux

intégré à l'application, génère une exécution moyenne du nombre de clics publicitaires par fournisseur et insère les lignes obtenues dans un autre flux intégré à l'application à l'aide de pompes. Pour plus d'informations sur le modèle classique, consultez Code d'application.

Vous pouvez fournir cette série d'instructions SQL où la sortie d'une instruction peut être utilisée comme l'entrée de la prochaine instruction. Vous stockez les résultats intermédiaires en créant des flux intégrés à l'application et des pompes.

Notez que le code d'application doit créer les flux avec des noms spécifiés dans les `Outputs`. Par exemple, si votre flux `Outputs` définit les flux de sortie nommés `ExampleOutputStream1` et `ExampleOutputStream2`, alors votre code d'application doit créer ces flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 0. Longueur maximale de 102400.

Obligatoire : non

ApplicationDescription

Description résumée de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 0. Longueur maximale de 1024.

Obligatoire : non

ApplicationName

Nom de votre application Amazon Kinesis Analytics (par exemple, `sample-app`).

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CloudWatchLoggingOptions

Utilisez ce paramètre pour configurer un flux de CloudWatch journal afin de surveiller les erreurs de configuration des applications. Pour plus d'informations, consultez la section Travailler avec Amazon CloudWatch Logs.

Type : tableau d’objets CloudWatchLoggingOption

Obligatoire : non

Inputs

Utilisez ce paramètre pour configurer l'entrée de l'application.

Vous pouvez configurer votre application pour qu'elle reçoive une entrée provenant d'une source de streaming. Dans cette configuration, vous mappez cette source de streaming au flux intégré à l'application qui est créé. Votre code d'application peut interroger le flux intégré à l'application comme une table (vous pouvez la considérer comme une table mise à jour en permanence).

Pour la source de streaming, vous fournissez son ARN (Amazon Resource Name) et le format des données sur le flux (par exemple, JSON, CSV, etc.). Vous devez également fournir un rôle IAM qu'Amazon Kinesis Analytics peut endosser pour lire ce flux en votre nom.

Pour créer le flux intégré à l'application, vous devez spécifier un schéma pour transformer vos données en une version schématisée utilisée dans SQL. Dans le schéma, vous indiquez le mappage nécessaires des éléments de données dans la source de streaming, pour enregistrer des colonnes dans le flux intégré à l'application.

Type : tableau d’objets Input

Obligatoire : non

Outputs

Vous pouvez configurer la sortie de l’application pour écrire des données à partir de n’importe quel flux intégré à l’application vers un maximum de trois destinations.

Ces destinations peuvent être des flux Amazon Kinesis, des flux de diffusion Amazon Kinesis Firehose, des destinations Lambda AWS ou une combinaison des trois.

Dans la configuration, vous spécifiez le nom du flux intégré à l’application, le flux de destination ou l’Amazon Resource Name (ARN) de la fonction Lambda, ainsi que le format à utiliser lors de l’écriture de données. Vous devez également fournir un rôle IAM qu’Amazon Kinesis Analytics peut endosser pour écrire dans le flux de destination ou la fonction Lambda en votre nom.

Dans la configuration de sortie, vous fournissez également le flux de sortie ou l’ARN de la fonction Lambda. Pour les destinations de flux, vous indiquez le format des données dans le flux (par

exemple, JSON, CSV). Vous devez également fournir un rôle IAM qu'Amazon Kinesis Analytics peut endosser pour écrire dans le flux ou la fonction Lambda en votre nom.

Type : tableau d'objets Output

Obligatoire : non

Tags

Liste d'une ou plusieurs balises à affecter à l'application. Une balise est une paire clé-valeur qui identifie une application. Notez que le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50. Pour plus d'informations, consultez la section Utilisation du balisage.

Type : tableau d'objets Tag

Membres du tableau : Nombre minimum de 1 élément. Nombre maximum de 200 éléments.

Obligatoire : non

Syntaxe de la réponse

```
{
   "ApplicationSummary": {
      "ApplicationARN": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string"
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

ApplicationSummary

En réponse à votre demande `CreateApplication`, Amazon Kinesis Analytics renvoie une réponse contenant un résumé de l'application créée, y compris l'Amazon Resource Name (ARN), le nom et le statut de l'application.

Type : objet ApplicationSummary

Erreurs

CodeValidationException

Le code d'application (requête) fourni par l'utilisateur n'est pas valide. Il peut s'agir d'une simple erreur de syntaxe.

Code d'état HTTP : 400

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

LimitExceededException

Le nombre d'applications autorisées a été dépassé.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

TooManyTagsException

Application créée avec trop de balises ou trop de balises ajoutées à une application. Notez que le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DeleteApplication

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Supprime l'application spécifiée. Amazon Kinesis Analytics arrête l'exécution de l'application et supprime l'application, y compris tous les artefacts d'application (tels que les flux intégrés à l'application, la table de référence et le code d'application).

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:DeleteApplication`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CreateTimestamp": number
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application Amazon Kinesis Analytics à supprimer.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CreateTimestamp

Vous pouvez utiliser l'opération `DescribeApplication` pour obtenir cette valeur.

Type : Timestamp

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DeleteApplicationCloudWatchLoggingOption

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Supprime un flux de CloudWatch journal d'une application. Pour plus d'informations sur l'utilisation des flux de CloudWatch journaux avec les applications Amazon Kinesis Analytics, consultez la section Travailler avec CloudWatch Amazon Logs.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOptionId": "string",
   "CurrentApplicationVersionId": number
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application Kinesis Analytics.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CloudWatchLoggingOptionId

L'option `CloudWatchLoggingOptionId` de CloudWatch journalisation à supprimer. Vous pouvez l'obtenir `CloudWatchLoggingOptionId` en utilisant l'DescribeApplicationopération.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

ID de version de l'application Kinesis Analytics.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DeleteApplicationInputProcessingConfiguration

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Supprime un code InputProcessingConfigurationd'une entrée.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string"
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application Kinesis Analytics.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

ID de version de l'application Kinesis Analytics.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

InputId

ID de la configuration d'entrée à partir de laquelle supprimer la configuration de traitement d'entrée. Vous pouvez obtenir une liste des identifiants d'entrée pour une application en utilisant l'DescribeApplicationopération.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DeleteApplicationOutput

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Supprime la configuration de destination de sortie de la configuration de votre application. Amazon Kinesis Analytics n'écrit plus de données depuis le flux intégré à l'application correspondant vers la destination de sortie externe.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:DeleteApplicationOutput`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "OutputId": "string"
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application Amazon Kinesis Analytics.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

Version de l'application Amazon Kinesis Analytics. Vous pouvez utiliser cette DescribeApplicationopération pour obtenir la version actuelle de l'application. Si la version spécifiée n'est pas la version actuelle, `ConcurrentModificationException` est renvoyé.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

OutputId

L'ID de la configuration à supprimer. Chaque configuration de sortie ajoutée à l'application, que ce soit lors de la création de l'application ou ultérieurement à l'aide de l'AddApplicationOutputopération, possède un identifiant unique. Vous devez fournir l'ID pour identifier de manière unique la configuration de sortie que vous souhaitez supprimer de la configuration de l'application. Vous pouvez utiliser l'DescribeApplicationopération pour obtenir le détail`OutputId`.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DeleteApplicationReferenceDataSource

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Supprime une configuration de source de données de référence de la configuration d'application spécifiée.

Si l'application est en cours d'exécution, Amazon Kinesis supprime immédiatement la table intégrée à l'application que vous avez créée à l'aide AddApplicationReferenceDataSourcede l'opération.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics.DeleteApplicationReferenceDataSource`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceId": "string"
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom d'une application existante.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

CurrentApplicationVersionId

Version de l'application. Vous pouvez utiliser cette DescribeApplicationopération pour obtenir la version actuelle de l'application. Si la version spécifiée n'est pas la version actuelle, `ConcurrentModificationException` est renvoyé.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

ReferenceId

ID de la source de données de référence. Lorsque vous ajoutez une source de données de référence à votre application à l'aide du AddApplicationReferenceDataSource, Amazon Kinesis Analytics attribue un identifiant. Vous pouvez utiliser cette DescribeApplicationopération pour obtenir l'ID de référence.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DescribeApplication

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Renvoie des informations sur une application Amazon Kinesis Analytics spécifique.

Si vous souhaitez récupérer la liste de toutes les applications de votre compte, utilisez l'ListApplicationsopération.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:DescribeApplication`. Vous pouvez l'utiliser `DescribeApplication` pour obtenir l'ID de version actuel de l'application, dont vous avez besoin pour appeler d'autres opérations telles que `Update`.

Syntaxe de la requête

```
{
   "ApplicationName": "string"
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Syntaxe de la réponse

```
{
   "ApplicationDetail": {
      "ApplicationARN": "string",
      "ApplicationCode": "string",
      "ApplicationDescription": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string",
      "ApplicationVersionId": number,
      "CloudWatchLoggingOptionDescriptions": [
         {
            "CloudWatchLoggingOptionId": "string",
            "LogStreamARN": "string",
            "RoleARN": "string"
         }
      ],
      "CreateTimestamp": number,
      "InputDescriptions": [
         {
            "InAppStreamNames": [ "string" ],
            "InputId": "string",
            "InputParallelism": {
               "Count": number
            },
            "InputProcessingConfigurationDescription": {
               "InputLambdaProcessorDescription": {
                  "ResourceARN": "string",
                  "RoleARN": "string"
               }
            },
            "InputSchema": {
               "RecordColumns": [
                  {
                     "Mapping": "string",
                     "Name": "string",
                     "SqlType": "string"
                  }
               ],
               "RecordEncoding": "string",
               "RecordFormat": {
                  "MappingParameters": {
                     "CSVMappingParameters": {
```

```
                        "RecordColumnDelimiter": "string",
                        "RecordRowDelimiter": "string"
                     },
                     "JSONMappingParameters": {
                        "RecordRowPath": "string"
                     }
                  },
                  "RecordFormatType": "string"
               }
            },
            "InputStartingPositionConfiguration": {
               "InputStartingPosition": "string"
            },
            "KinesisFirehoseInputDescription": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "KinesisStreamsInputDescription": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "NamePrefix": "string"
         }
      ],
      "LastUpdateTimestamp": number,
      "OutputDescriptions": [
         {
            "DestinationSchema": {
               "RecordFormatType": "string"
            },
            "KinesisFirehoseOutputDescription": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "KinesisStreamsOutputDescription": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "LambdaOutputDescription": {
               "ResourceARN": "string",
               "RoleARN": "string"
            },
            "Name": "string",
            "OutputId": "string"
```

```
         }
      ],
      "ReferenceDataSourceDescriptions": [
         {
            "ReferenceId": "string",
            "ReferenceSchema": {
               "RecordColumns": [
                  {
                     "Mapping": "string",
                     "Name": "string",
                     "SqlType": "string"
                  }
               ],
               "RecordEncoding": "string",
               "RecordFormat": {
                  "MappingParameters": {
                     "CSVMappingParameters": {
                        "RecordColumnDelimiter": "string",
                        "RecordRowDelimiter": "string"
                     },
                     "JSONMappingParameters": {
                        "RecordRowPath": "string"
                     }
                  },
                  "RecordFormatType": "string"
               }
            },
            "S3ReferenceDataSourceDescription": {
               "BucketARN": "string",
               "FileKey": "string",
               "ReferenceRoleARN": "string"
            },
            "TableName": "string"
         }
      ]
   }
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

ApplicationDetail

Fournit une description de l'application, comme l'Amazon Resource Name (ARN), le statut, la dernière version et les détails de configuration d'entrée et de sortie de l'application.

Type : objet ApplicationDetail

Erreurs

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

DiscoverInputSchema

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Déduit un schéma en évaluant des exemples d'enregistrements sur la source de streaming spécifiée (flux Amazon Kinesis ou flux de diffusion Amazon Kinesis Firehose) ou sur l'objet S3 spécifié. Dans la réponse, l'opération renvoie le schéma déduit ainsi que les exemples d'enregistrements utilisés par l'opération pour déduire le schéma.

Vous pouvez utiliser le schéma déduit lors de la configuration d'une source de streaming pour votre application. Pour obtenir des informations conceptuelles, veuillez consulter Configuration de l'entrée de l'application. Notez que lorsque vous créez une application à l'aide de la console Amazon Kinesis Analytics, celle-ci utilise cette opération pour déduire un schéma et l'afficher dans l'interface utilisateur de la console.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:DiscoverInputSchema`.

Syntaxe de la requête

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "string"
   },
   "ResourceARN": "string",
   "RoleARN": "string",
   "S3Configuration": {
      "BucketARN": "string",
```

```
        "FileKey": "string",
        "RoleARN": "string"
    }
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

InputProcessingConfiguration

InputProcessingConfigurationÀ utiliser pour prétraiter les enregistrements avant de découvrir le schéma des enregistrements.

Type : objet InputProcessingConfiguration

Obligatoire : non

InputStartingPositionConfiguration

Point à partir duquel vous souhaitez qu'Amazon Kinesis Analytics commence à lire les enregistrements correspondant aux objectifs de découverte de sources de streaming spécifiés.

Type : objet InputStartingPositionConfiguration

Obligatoire : non

ResourceARN

L'Amazon Resource Name (ARN) de la source de streaming.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

S3Configuration

Spécifiez ce paramètre pour découvrir un schéma à partir des données d'un objet Amazon S3.

Type : objet S3Configuration

Obligatoire : non

Syntaxe de la réponse

```
{
   "InputSchema": {
      "RecordColumns": [
         {
            "Mapping": "string",
            "Name": "string",
            "SqlType": "string"
         }
      ],
      "RecordEncoding": "string",
      "RecordFormat": {
         "MappingParameters": {
            "CSVMappingParameters": {
               "RecordColumnDelimiter": "string",
               "RecordRowDelimiter": "string"
            },
            "JSONMappingParameters": {
               "RecordRowPath": "string"
            }
         },
         "RecordFormatType": "string"
      }
   },
   "ParsedInputRecords": [
      [ "string" ]
   ],
   "ProcessedInputRecords": [ "string" ],
```

```
    "RawInputRecords": [ "string" ]
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

InputSchema

Schéma déduit de la source de streaming. Il identifie le format des données de la source de streaming et la façon dont chaque élément de données est mappé aux colonnes correspondantes que vous pouvez créer dans le flux intégré à l'application.

Type : objet SourceSchema

ParsedInputRecords

Tableau d'éléments, où chaque élément correspond à une ligne d'un enregistrement de flux (un enregistrement de flux peut comporter plusieurs lignes).

Type : Tableau de tableaux de chaînes

ProcessedInputRecords

Données de flux modifiées par le processeur spécifié dans le paramètre `InputProcessingConfiguration`.

Type : tableau de chaînes

RawInputRecords

Données de flux brutes qui ont été échantillonnées pour déduire le schéma.

Type : tableau de chaînes

Erreurs

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceProvisionedThroughputExceededException

Discovery n'a pas réussi à obtenir d'enregistrement depuis la source de streaming à cause d'Amazon Kinesis ProvisionedThroughputExceededException Streams. Pour plus d'informations, consultez GetRecordsle manuel Amazon Kinesis Streams API Reference.

Code d'état HTTP : 400

ServiceUnavailableException

Le service n'est pas disponible. Faites un retour en arrière et réessayez l'opération.

Code d'état HTTP : 500

UnableToDetectSchemaException

Le format de date n'est pas valide. Amazon Kinesis Analytics n'est pas en mesure de détecter le schéma de la source de streaming donnée.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python

- AWS SDK pour Ruby V3

ListApplications

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Renvoie la liste des applications Amazon Kinesis Analytics associées à votre compte. Pour chaque application, la réponse inclut le nom, l'Amazon Resource Name (ARN) et son statut de l'application. Si la réponse renvoie la valeur `HasMoreApplications` comme vraie, vous pouvez envoyer une autre requête en ajoutant le code `ExclusiveStartApplicationName` dans le corps de la requête et en définissant la valeur sur le nom de la dernière application indiqué dans la réponse précédente.

Si vous souhaitez obtenir des informations détaillées sur une application spécifique, utilisez DescribeApplication.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:ListApplications`.

Syntaxe de la requête

```
{
   "ExclusiveStartApplicationName": "string",
   "Limit": number
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ExclusiveStartApplicationName

Nom de l'application avec laquelle commencer la liste. Lorsque vous utilisez la pagination pour récupérer la liste, vous n'avez pas besoin de spécifier ce paramètre dans la première requête. Toutefois, dans les requêtes suivantes, vous ajoutez le nom de la dernière application figurant dans la réponse précédente pour obtenir la page suivante des applications.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : non

Limit

Nombre maximum d'applications à répertorier.

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale de 50.

Obligatoire : non

Syntaxe de la réponse

```
{
   "ApplicationSummaries": [
      {
         "ApplicationARN": "string",
         "ApplicationName": "string",
         "ApplicationStatus": "string"
      }
   ],
   "HasMoreApplications": boolean
}
```

Eléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200.

Les données suivantes sont renvoyées au format JSON par le service.

ApplicationSummaries

Liste d'objets `ApplicationSummary`.

Type : tableau d'objets ApplicationSummary

HasMoreApplications

Renvoie la valeur vraie si d'autres applications sont à récupérer.

Type : booléen

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

ListTagsForResource

Récupère la liste des balises clé-valeur attribuées à l'application. Pour plus d'informations, consultez la section Utilisation du balisage.

Syntaxe de la requête

```
{
   "ResourceARN": "string"
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ResourceARN

L'ARN de l'application pour laquelle des balises sont à récupérer.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

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 au format JSON par le service.

Tags

Les balises clé-valeur attribuées à l'application.

Type : tableau d'objets Tag

Membres du tableau : Nombre minimum de 1 élément. Nombre maximum de 200 éléments.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2

- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

StartApplication

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Démarre l'application Amazon Kinesis Analytics spécifiée. Après avoir créé une application, vous devez exclusivement appeler cette opération pour démarrer votre application.

Une fois que l'application démarre, elle commence à consommer les données d'entrée, à les traiter et à écrire la sortie sur la destination configurée.

Le statut de l'application doit être `READY` pour que vous puissiez la démarrer. Vous pouvez obtenir l'état de l'application dans la console ou à l'aide de l'DescribeApplicationopération.

Après avoir démarré l'application, vous pouvez l'empêcher de traiter l'entrée en appelant l'StopApplicationopération.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:StartApplication`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "InputConfigurations": [
      {
         "Id": "string",
         "InputStartingPositionConfiguration": {
            "InputStartingPosition": "string"
         }
      }
   ]
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

InputConfigurations

Identifie l'entrée spécifique, par ID, que l'application commence à consommer. Amazon Kinesis Analytics commence à lire la source de streaming associée à l'entrée. Vous pouvez également spécifier l'endroit de la source de streaming à partir duquel Amazon Kinesis Analytics doit commence à lire.

Type : tableau d'objets InputConfiguration

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

InvalidApplicationConfigurationException

La configuration de l'application fournie par l'utilisateur n'est pas valide.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

StopApplication

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Empêche l'application de traiter les données d'entrée. Vous ne pouvez arrêter une application que si elle est en cours d'exécution. Vous pouvez utiliser cette DescribeApplicationopération pour connaître l'état de l'application. Une fois l'application arrêtée, Amazon Kinesis Analytics arrête de lire les données en entrée, l'application arrête de traiter les données et aucune sortie n'est écrite vers la destination.

Cette opération exige des autorisations pour exécuter l'action `kinesisanalytics:StopApplication`.

Syntaxe de la requête

```
{
   "ApplicationName": "string"
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application en cours d'exécution à arrêter.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

TagResource

Ajoute une ou plusieurs balises clé-valeur à une application Kinesis Analytics. Notez que le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50. Pour plus d'informations, consultez la section Utilisation du balisage.

Syntaxe de la requête

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ResourceARN

L'ARN de l'application auquel attribuer les balises.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

Tags

Les balises clé-valeur à attribuer à l'application.

Type : tableau d'objets Tag

Membres du tableau : Nombre minimum de 1 élément. Nombre maximum de 200 éléments.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

TooManyTagsException

Application créée avec trop de balises ou trop de balises ajoutées à une application. Notez que le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS

- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

UntagResource

Supprime une ou plusieurs balises d'une application Kinesis Analytics. Pour plus d'informations, consultez la section Utilisation du balisage.

Syntaxe de la requête

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ResourceARN

L'ARN de l'application Kinesis Analytics à partir duquel les balises doivent être supprimées.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

TagKeys

Liste des clés de balises à supprimer de l'application spécifiée.

Type : tableau de chaînes

Membres du tableau : Nombre minimum de 1 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 corps HTTP vide.

Erreurs

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

TooManyTagsException

Application créée avec trop de balises ou trop de balises ajoutées à une application. Notez que le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++

- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

UpdateApplication

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Met à jour une application Amazon Kinesis Analytics existante. À l'aide de cette API, vous pouvez mettre à jour le code de l'application, la configuration d'entrée et la configuration de sortie.

Notez qu'Amazon Kinesis Analytics met à jour le code `CurrentApplicationVersionId` chaque fois que vous mettez à jour votre application.

Cette opération nécessite une autorisation pour l'action `kinesisanalytics:UpdateApplication`.

Syntaxe de la requête

```
{
   "ApplicationName": "string",
   "ApplicationUpdate": {
      "ApplicationCodeUpdate": "string",
      "CloudWatchLoggingOptionUpdates": [
         {
            "CloudWatchLoggingOptionId": "string",
            "LogStreamARNUpdate": "string",
            "RoleARNUpdate": "string"
         }
      ],
      "InputUpdates": [
         {
            "InputId": "string",
            "InputParallelismUpdate": {
               "CountUpdate": number
            },
            "InputProcessingConfigurationUpdate": {
               "InputLambdaProcessorUpdate": {
                  "ResourceARNUpdate": "string",
                  "RoleARNUpdate": "string"
```

```
            }
         },
         "InputSchemaUpdate": {
            "RecordColumnUpdates": [
               {
                  "Mapping": "string",
                  "Name": "string",
                  "SqlType": "string"
               }
            ],
            "RecordEncodingUpdate": "string",
            "RecordFormatUpdate": {
               "MappingParameters": {
                  "CSVMappingParameters": {
                     "RecordColumnDelimiter": "string",
                     "RecordRowDelimiter": "string"
                  },
                  "JSONMappingParameters": {
                     "RecordRowPath": "string"
                  }
               },
               "RecordFormatType": "string"
            }
         },
         "KinesisFirehoseInputUpdate": {
            "ResourceARNUpdate": "string",
            "RoleARNUpdate": "string"
         },
         "KinesisStreamsInputUpdate": {
            "ResourceARNUpdate": "string",
            "RoleARNUpdate": "string"
         },
         "NamePrefixUpdate": "string"
      }
   ],
   "OutputUpdates": [
      {
         "DestinationSchemaUpdate": {
            "RecordFormatType": "string"
         },
         "KinesisFirehoseOutputUpdate": {
            "ResourceARNUpdate": "string",
            "RoleARNUpdate": "string"
         },
```

```
            "KinesisStreamsOutputUpdate": {
               "ResourceARNUpdate": "string",
               "RoleARNUpdate": "string"
            },
            "LambdaOutputUpdate": {
               "ResourceARNUpdate": "string",
               "RoleARNUpdate": "string"
            },
            "NameUpdate": "string",
            "OutputId": "string"
         }
      ],
      "ReferenceDataSourceUpdates": [
         {
            "ReferenceId": "string",
            "ReferenceSchemaUpdate": {
               "RecordColumns": [
                  {
                     "Mapping": "string",
                     "Name": "string",
                     "SqlType": "string"
                  }
               ],
               "RecordEncoding": "string",
               "RecordFormat": {
                  "MappingParameters": {
                     "CSVMappingParameters": {
                        "RecordColumnDelimiter": "string",
                        "RecordRowDelimiter": "string"
                     },
                     "JSONMappingParameters": {
                        "RecordRowPath": "string"
                     }
                  },
                  "RecordFormatType": "string"
               }
            },
            "S3ReferenceDataSourceUpdate": {
               "BucketARNUpdate": "string",
               "FileKeyUpdate": "string",
               "ReferenceRoleARNUpdate": "string"
            },
            "TableNameUpdate": "string"
         }
```

```
        ]
    },
    "CurrentApplicationVersionId": number
}
```

Paramètres de demande

Cette demande accepte les données suivantes au format JSON.

ApplicationName

Nom de l'application Amazon Kinesis Analytics à mettre à jour.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

ApplicationUpdate

Décrit les mises à jour de l'application.

Type : objet ApplicationUpdate

Obligatoire : oui

CurrentApplicationVersionId

ID de la version d'application actuelle. Vous pouvez utiliser l'DescribeApplicationopération pour obtenir cette valeur.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

Éléments de réponse

Si l'action aboutit, le service renvoie une réponse HTTP 200 avec un corps HTTP vide.

Erreurs

CodeValidationException

Le code d'application (requête) fourni par l'utilisateur n'est pas valide. Il peut s'agir d'une simple erreur de syntaxe.

Code d'état HTTP : 400

ConcurrentModificationException

Exception envoyée suite à la modification simultanée d'une application. Par exemple, deux personnes tentent de modifier la même application en même temps.

Code d'état HTTP : 400

InvalidArgumentException

La valeur du paramètre d'entrée spécifiée n'est pas valide.

Code d'état HTTP : 400

ResourceInUseException

L'application n'est pas disponible pour cette opération.

Code d'état HTTP : 400

ResourceNotFoundException

L'application spécifiée est introuvable.

Code d'état HTTP : 400

UnsupportedOperationException

La requête a été rejetée car un paramètre spécifié n'est pas pris en charge ou parce qu'une ressource spécifiée n'est pas valide pour cette opération.

Code d'état HTTP : 400

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- Interface de ligne de commande AWS
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go v2
- AWS SDK pour Java V2
- AWS SDK pour V3 JavaScript
- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

Types de données

Les types de données suivants sont pris en charge :

- ApplicationDetail
- ApplicationSummary
- ApplicationUpdate
- CloudWatchLoggingOption
- CloudWatchLoggingOptionDescription
- CloudWatchLoggingOptionUpdate
- CSVMappingParameters
- DestinationSchema
- Input
- InputConfiguration
- InputDescription
- InputLambdaProcessor
- InputLambdaProcessorDescription
- InputLambdaProcessorUpdate
- InputParallelism
- InputParallelismUpdate
- InputProcessingConfiguration

- InputProcessingConfigurationDescription
- InputProcessingConfigurationUpdate
- InputSchemaUpdate
- InputStartingPositionConfiguration
- InputUpdate
- JSONMappingParameters
- KinesisFirehoseInput
- KinesisFirehoseInputDescription
- KinesisFirehoseInputUpdate
- KinesisFirehoseOutput
- KinesisFirehoseOutputDescription
- KinesisFirehoseOutputUpdate
- KinesisStreamsInput
- KinesisStreamsInputDescription
- KinesisStreamsInputUpdate
- KinesisStreamsOutput
- KinesisStreamsOutputDescription
- KinesisStreamsOutputUpdate
- LambdaOutput
- LambdaOutputDescription
- LambdaOutputUpdate
- MappingParameters
- Output
- OutputDescription
- OutputUpdate
- RecordColumn
- RecordFormat
- ReferenceDataSource
- ReferenceDataSourceDescription
- ReferenceDataSourceUpdate

- S3Configuration
- S3ReferenceDataSource
- S3ReferenceDataSourceDescription
- S3ReferenceDataSourceUpdate
- SourceSchema
- Tag

ApplicationDetail

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Fournit une description de l'application, y compris l'Amazon Resource Name (ARN), le statut, la dernière version et la configuration d'entrée et de sortie de l'application.

Table des matières

ApplicationARN

ARN de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

ApplicationName

Nom de l'application.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

ApplicationStatus

Statut de l'application.

Type : chaîne

Valeurs valides : `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

Obligatoire : oui

ApplicationVersionId

Fournit la version d'application actuelle.

Type : long

Plage valide : valeur minimum de 1. Valeur maximale de 999999999.

Obligatoire : oui

ApplicationCode

Renvoie le code d'application que vous avez fourni pour effectuer une analyse des données sur l'un des flux intégrés à l'application dans votre application.

Type : chaîne

Contraintes de longueur : longueur minimum de 0. Longueur maximale de 102400.

Obligatoire : non

ApplicationDescription

Description de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 0. Longueur maximale de 1024.

Obligatoire : non

CloudWatchLoggingOptionDescriptions

Décrit les flux de CloudWatch journaux configurés pour recevoir des messages d'application. Pour plus d'informations sur l'utilisation des flux de CloudWatch journaux avec les applications Amazon Kinesis Analytics, consultez la section Travailler avec CloudWatch Amazon Logs.

Type : tableau d'objets CloudWatchLoggingOptionDescription

Obligatoire : non

CreateTimestamp

Heure à laquelle la version de l'application a été créée.

Type : Timestamp

Obligatoire : non

InputDescriptions

Décrit la configuration d'entrée de l'application. Pour plus d'informations, consultez Configuration de l'entrée de l'application.

Type : tableau d'objets InputDescription

Obligatoire : non

LastUpdateTimestamp

Horodatage de la dernière mise à jour de l'application.

Type : Timestamp

Obligatoire : non

OutputDescriptions

Décrit la configuration de sortie de l'application. Pour plus d'informations, consultez Configuration de la sortie d'application.

Type : tableau d'objets OutputDescription

Obligatoire : non

ReferenceDataSourceDescriptions

Décrit les sources de données de référence configurées pour l'application. Pour plus d'informations, consultez Configuration de l'entrée de l'application.

Type : tableau d'objets ReferenceDataSourceDescription

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ApplicationSummary

> **Note**
>
> Cette documentation est destinée à la version 1 de l'API Amazon Kinesis Data Analytics, qui est compatible uniquement avec les applications SQL. La version 2 de l'API est compatible avec les applications SQL et Java. Pour plus d'informations sur la version 2, consultez la documentation de l'API Amazon Kinesis Data Analytics V2.

Fournit des informations récapitulatives de l'application, notamment l'Amazon Resource Name (ARN), son nom et son statut.

Table des matières

ApplicationARN

ARN de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

ApplicationName

Nom de l'application.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

ApplicationStatus

Statut de l'application.

Type : chaîne

Valeurs valides : `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ApplicationUpdate

Décrit les mises à jour à appliquer à une application Amazon Kinesis Analytics existante.

Table des matières

ApplicationCodeUpdate

Décrit les mises à jour du code d'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 0. Longueur maximale de 102400.

Obligatoire : non

CloudWatchLoggingOptionUpdates

Décrit les mises à jour des options de CloudWatch journalisation des applications

Type : tableau d'objets CloudWatchLoggingOptionUpdate

Obligatoire : non

InputUpdates

Décrit les mises à jour de la configuration d'entrée d'application.

Type : tableau d'objets InputUpdate

Obligatoire : non

OutputUpdates

Décrit les mises à jour de la configuration de sortie d'application.

Type : tableau d'objets OutputUpdate

Obligatoire : non

ReferenceDataSourceUpdates

Décrit les mises à jour des sources de données de référence de l'application.

Type : tableau d'objets ReferenceDataSourceUpdate

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

CloudWatchLoggingOption

Fournit une description des options de CloudWatch journalisation, notamment le nom de ressource Amazon (ARN) du flux de journal et l'ARN du rôle.

Table des matières

LogStreamARN

ARN du CloudWatch journal pour recevoir les messages de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN IAM du rôle à utiliser pour envoyer les messages de l’application. Remarque : Pour écrire des messages d'application dans le rôle IAM utilisé CloudWatch, l'action de `PutLogEvents` stratégie doit être activée.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

CloudWatchLoggingOptionDescription

Description de l'option de CloudWatch journalisation.

Table des matières

LogStreamARN

ARN du CloudWatch journal pour recevoir les messages de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN IAM du rôle à utiliser pour envoyer les messages de l’application. Remarque : Pour écrire des messages d'application CloudWatch, l'action de `PutLogEvents` stratégie doit être activée pour le rôle IAM utilisé.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

CloudWatchLoggingOptionId

ID de la description de l'option de CloudWatch journalisation.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

CloudWatchLoggingOptionUpdate

Décrit les mises à jour des options d' CloudWatch enregistrement.

Table des matières

CloudWatchLoggingOptionId

ID de l'option de CloudWatch journalisation à mettre à jour

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

LogStreamARNUpdate

ARN du CloudWatch journal pour recevoir les messages de l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

ARN IAM du rôle à utiliser pour envoyer les messages de l’application. Remarque : Pour écrire des messages d'application CloudWatch, l'action de `PutLogEvents` stratégie doit être activée pour le rôle IAM utilisé.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

CSVMappingParameters

Fournit des informations de mappage supplémentaires lorsque le format d'enregistrement utilise des délimiteurs, tels que CSV. Par exemple, les enregistrements suivants utilisent le format CSV lorsque les enregistrements utilisent « \n » comme délimiteur de ligne et une virgule (« , ») comme délimiteur de colonne :

```
"name1", "address1"

"name2", "address2"
```

Table des matières

RecordColumnDelimiter

Délimiteur de colonne. Par exemple, dans un format CSV, la virgule (« , ») est le délimiteur de colonne classique.

Type : chaîne

Contraintes de longueur : longueur minimum de 1.

Obligatoire : oui

RecordRowDelimiter

Délimiteur de ligne. Par exemple, dans un format CSV, « \n » est le délimiteur de ligne classique.

Type : chaîne

Contraintes de longueur : longueur minimum de 1.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2

- AWS SDK pour Ruby V3

DestinationSchema

Décrit le format de données utilisé pour écrire les enregistrements dans la destination. Pour plus d'informations, consultez Configuration de la sortie d'application.

Table des matières

RecordFormatType

Spécifie le format des enregistrements présents dans le flux de sortie.

Type : chaîne

Valeurs valides : `JSON | CSV`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Input

Lorsque vous configurez l'entrée de l'application, vous spécifiez la source de diffusion, le nom du flux intégré à l'application qui est créée, et le mappage entre les deux. Pour plus d'informations, consultez Configuration de l'entrée de l'application.

Table des matières

InputSchema

Décrit le format des données de la source de diffusion, et la manière dont chaque élément de données est mappé aux colonnes correspondantes dans le flux intégré à l'application qui est en cours de création.

Egalement utilisé pour décrire le format de la source de données de référence.

Type : objet SourceSchema

Obligatoire : oui

NamePrefix

Préfixe de nom à utiliser lors de la création d'un flux intégré à l'application. Supposons que vous spécifiez un préfixe « »MyInApplicationStream. Amazon Kinesis Analytics crée ensuite un ou plusieurs flux intégrés à l'application (selon `InputParallelism` le nombre que vous avez spécifié) portant les noms MyInApplicationStream « _001 », MyInApplicationStream « _002 », etc.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : oui

InputParallelism

Décrit le nombre de flux intégrés à l'application à créer.

Les données de votre source sont acheminées vers ces flux d'entrée intégrés à l'application.

(consultez Configuration de l'entrée de l'application).

Type : objet InputParallelism

Obligatoire : non

InputProcessingConfiguration

Le InputProcessingConfigurationpour la saisie. Un processeur d'entrée transforme les enregistrements au fur et à mesure qu'ils sont reçus depuis le flux, avant l'exécution de l'application du code SQL. Actuellement, la seule configuration de traitement d'entrée disponible est InputLambdaProcessor.

Type : objet InputProcessingConfiguration

Obligatoire : non

KinesisFirehoseInput

Si la source de streaming est un flux de diffusion Amazon Kinesis Firehose, identifie l'ARN (Amazon Resource Name) du flux de diffusion et un rôle IAM qui permet à Amazon Kinesis Analytics d'accéder au flux en votre nom.

Remarque : vous devez indiquer l'élément `KinesisStreamsInput` ou `KinesisFirehoseInput`.

Type : objet KinesisFirehoseInput

Obligatoire : non

KinesisStreamsInput

Si la source de streaming est un flux Amazon Kinesis, identifie l'Amazon Resource Name (ARN) du flux et un rôle IAM qui permet à Amazon Kinesis Analytics d'accéder au flux en votre nom.

Remarque : vous devez indiquer l'élément `KinesisStreamsInput` ou `KinesisFirehoseInput`.

Type : objet KinesisStreamsInput

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++

- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputConfiguration

Lorsque vous démarrez votre application, vous fournissez cette configuration, qui identifie la source d'entrée et le point de la source d'entrée à partir duquel vous souhaitez que l'application commence à traiter les enregistrements.

Table des matières

Id

ID de la source d'entrée. Vous pouvez obtenir cet identifiant en appelant l'DescribeApplicationopération.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

InputStartingPositionConfiguration

Point à partir duquel vous souhaitez que l'application commence à traiter les enregistrements issus de la source de streaming.

Type : objet InputStartingPositionConfiguration

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputDescription

Décrit la configuration d'entrée de l'application. Pour plus d'informations, consultez Configuration de l'entrée de l'application.

Table des matières

InAppStreamNames

Renvoie les noms de flux intégrés à l'application qui sont mappés à la source du flux.

Type : tableau de chaînes

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : non

InputId

ID d'entrée associé à l'entrée de l'application. Il s'agit de l'identifiant qu'Amazon Kinesis Analytics attribue à chaque configuration d'entrée que vous ajoutez à votre application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : non

InputParallelism

Décrit le parallélisme configuré (nombre de flux intégrés à l'application mappés à la source du flux).

Type : objet InputParallelism

Obligatoire : non

InputProcessingConfigurationDescription

Description du préprocesseur qui s'exécute sur les enregistrements de cette entrée avant que le code de l'application ne soit exécuté.

Type : objet InputProcessingConfigurationDescription

Obligatoire : non

InputSchema

Décrit le format des données de la source de diffusion, et la manière dont chaque élément de données est mappé aux colonnes correspondantes dans le flux intégré à l'application qui est en cours de création.

Type : objet SourceSchema

Obligatoire : non

InputStartingPositionConfiguration

Point à partir duquel l'application est configurée pour lire à partir du flux d'entrée.

Type : objet InputStartingPositionConfiguration

Obligatoire : non

KinesisFirehoseInputDescription

Si un flux de diffusion Amazon Kinesis Firehose est configuré en tant que source de streaming, fournit l'ARN du flux de diffusion et un rôle IAM qui permet à Amazon Kinesis Analytics d'accéder au flux en votre nom.

Type : objet KinesisFirehoseInputDescription

Obligatoire : non

KinesisStreamsInputDescription

Si un flux Amazon Kinesis est configuré en tant que source de streaming, fournit l'Amazon Resource Name (ARN) du flux Amazon Kinesis et un rôle IAM qui permet à Amazon Kinesis Analytics d'accéder au flux en votre nom.

Type : objet KinesisStreamsInputDescription

Obligatoire : non

NamePrefix

Préfixe du nom intégré à l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputLambdaProcessor

Objet contenant l'Amazon Resource Name (ARN) de la fonction AWS Lambda utilisée pour prétraiter les enregistrements du flux, et l'ARN du rôle IAM utilisé pour accéder à la fonction Lambda. AWS

Table des matières

ResourceARN

L'ARN de la fonction AWS Lambda qui s'exécute sur les enregistrements du flux.

> **Note**
>
> Pour spécifier une version antérieure à la dernière version de la fonction Lambda, incluez la version de la fonction dans l'ARN de la fonction Lambda. Pour plus d'informations sur les ARN Lambda, voir Exemples d'ARN : Lambda AWS

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

L'ARN du rôle IAM utilisé pour accéder à la fonction AWS Lambda.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputLambdaProcessorDescription

Objet contenant l'Amazon Resource Name (ARN) de la fonction AWS Lambda utilisée pour prétraiter les enregistrements du flux, ainsi que l'ARN du rôle IAM utilisé pour accéder à l'expression Lambda. AWS

Table des matières

ResourceARN

L'ARN de la fonction Lambda AWS qui est utilisée pour prétraiter les enregistrements du flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

L'ARN du rôle IAM utilisé pour accéder à la fonction AWS Lambda.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputLambdaProcessorUpdate

Représente une mise à jour du InputLambdaProcessorqui est utilisé pour prétraiter les enregistrements du flux.

Table des matières

ResourceARNUpdate

L'Amazon Resource Name (ARN) de la nouvelle fonction Lambda AWS qui est utilisée pour prétraiter les enregistrements du flux.

> **Note**
>
> Pour spécifier une version antérieure à la dernière version de la fonction Lambda, incluez la version de la fonction dans l'ARN de la fonction Lambda. Pour plus d'informations sur les ARN Lambda, voir Exemples d'ARN : Lambda AWS

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

L'ARN du nouveau rôle IAM utilisé pour accéder à la fonction AWS Lambda.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputParallelism

Décrit le nombre de flux intégrés à l'application à créer pour une source de diffusion donnée. Pour plus d'informations sur le parallélisme, consultez Configuration de l'entrée de l'application.

Table des matières

Count

Nombre de flux intégrés à l'application à créer. Pour plus d’informations, consultez Limites .

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale de 64.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputParallelismUpdate

Fournit des mises à jour du nombre de parallélismes.

Table des matières

CountUpdate

Nombre de flux intégrés à l'application à créer pour la source de streaming donnée.

Type : entier

Plage valide : valeur minimum de 1. Valeur maximale de 64.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputProcessingConfiguration

Fournit une description d'un processeur utilisé pour prétraiter les enregistrements dans le flux avant qu'ils ne soient traitées par le code de votre application. Actuellement, le seul processeur d'entrée disponible est AWS Lambda.

Table des matières

InputLambdaProcessor

Le InputLambdaProcessorqui est utilisé pour prétraiter les enregistrements du flux avant d'être traités par le code de votre application.

Type : objet InputLambdaProcessor

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputProcessingConfigurationDescription

Fournit les informations de configuration relatives à un processeur d'entrée. Actuellement, le seul processeur d'entrée disponible est AWS Lambda.

Table des matières

InputLambdaProcessorDescription

Fournit des informations de configuration sur les éléments associés InputLambdaProcessorDescription.

Type : objet InputLambdaProcessorDescription

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputProcessingConfigurationUpdate

Décrit les mises à jour d'un InputProcessingConfiguration.

Table des matières

InputLambdaProcessorUpdate

Fournit des informations de mise à jour pour un InputLambdaProcessor.

Type : objet InputLambdaProcessorUpdate

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputSchemaUpdate

Décrit les mises à jour du schéma d'entrée de l'application.

Table des matières

RecordColumnUpdates

Liste d'objets `RecordColumn`. Chaque objet décrit le mappage de l'élément de la source de streaming à la colonne correspondante du flux intégré à l'application.

Type : tableau d'objets RecordColumn

Membres du tableau : Nombre minimum de 1 élément. Nombre maximum de 1 000 éléments.

Obligatoire : non

RecordEncodingUpdate

Indique l'encodage des enregistrements dans la source de diffusion. Par exemple, UTF-8.

Type : chaîne

Modèle : `UTF-8`

Obligatoire : non

RecordFormatUpdate

Spécifie le format des enregistrements présents dans la source de diffusion.

Type : objet RecordFormat

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputStartingPositionConfiguration

Décrit le point à partir duquel l'application lit dans la source de streaming.

Table des matières

InputStartingPosition

Position de début dans le flux.

- `NOW` : commencer à lire juste après l'enregistrement le plus récent du flux, en commençant à l'horodatage de la requête émise par le client.
- `TRIM_HORIZON` : commencer à lire au dernier enregistrement non découpé du flux, qui correspond au plus ancien enregistrement disponible du flux. Cette option n'est pas disponible pour un flux de diffusion Amazon Kinesis Firehose.
- `LAST_STOPPED_POINT` : reprendre la lecture depuis l'endroit où l'application a arrêté la lecture pour la dernière fois.

Type : chaîne

Valeurs valides : `NOW | TRIM_HORIZON | LAST_STOPPED_POINT`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

InputUpdate

Décrit les mises à jour apportées à une configuration d'entrée spécifique (identifiée par l'`InputId` d'une application).

Table des matières

InputId

ID d'entrée de l'entrée de l'application à mettre à jour.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

InputParallelismUpdate

Décrit les mises à jour du parallélisme (le nombre de flux intégrés à l'application créés par Amazon Kinesis Analytics pour une source de streaming spécifique).

Type : objet InputParallelismUpdate

Obligatoire : non

InputProcessingConfigurationUpdate

Décrit les mises à jour d'une configuration de traitement des entrées.

Type : objet InputProcessingConfigurationUpdate

Obligatoire : non

InputSchemaUpdate

Décrit le format des données de la source de streaming, et la manière dont les éléments d'enregistrements de la source de streaming sont mappés aux colonnes du flux intégré à l'application qui est en cours de création.

Type : objet InputSchemaUpdate

Obligatoire : non

KinesisFirehoseInputUpdate

Si un flux de diffusion Amazon Kinesis Firehose est la source de streaming à mettre à jour, fournit un ARN de flux et un ARN de rôle IAM mis à jour.

Type : objet KinesisFirehoseInputUpdate

Obligatoire : non

KinesisStreamsInputUpdate

Si un flux Amazon Kinesis est la source de streaming à mettre à jour, fournit un Amazon Resource Name (ARN) de flux et un ARN de rôle IAM mis à jour.

Type : objet KinesisStreamsInputUpdate

Obligatoire : non

NamePrefixUpdate

Préfixe de nom pour les flux intégrés à l'application créés par Amazon Kinesis Analytics pour une source de streaming spécifique.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

JSONMappingParameters

Fournit des informations de mappage supplémentaires lorsque JSON est le format d'enregistrement utilisé sur la source de diffusion.

Table des matières

RecordRowPath

Chemin d'accès au parent de premier niveau qui contient les enregistrements.

Type : chaîne

Contraintes de longueur : longueur minimum de 1.

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisFirehoseInput

Identifie un flux de diffusion Amazon Kinesis Firehose comme source de diffusion. Vous fournissez l'Amazon Resource Name (ARN) du flux de diffusion et l'ARN d'un rôle IAM qui permet à Amazon Kinesis Analytics d'accéder au flux en votre nom.

Table des matières

ResourceARN

ARN du flux de diffusion d'entrée.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom. Vous devez vous assurer que le rôle dispose des autorisations nécessaires pour accéder au flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisFirehoseInputDescription

Décrit le flux de diffusion Amazon Kinesis Firehose configuré en tant que source de streaming dans la configuration d'entrée de l'application.

Table des matières

ResourceARN

Amazon Resource Name (ARN) du flux de diffusion Amazon Kinesis Firehose.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

ARN du rôle IAM qu'Amazon Kinesis Analytics endosse pour accéder au flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisFirehoseInputUpdate

Lors de la mise à jour d'une configuration d'entrée d'application, fournit des informations sur un flux de diffusion Amazon Kinesis Firehose utilisé en tant que source de streaming.

Table des matières

ResourceARNUpdate

Amazon Resource Name (ARN) du flux de diffusion Amazon Kinesis Firehose d'entrée à lire.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisFirehoseOutput

Lors de la configuration d'une sortie d'application, identifie un flux de diffusion Amazon Kinesis Firehose en tant que destination. Vous fournissez l'Amazon Resource Name (ARN) du flux et un rôle IAM qui permet à Amazon Kinesis Analytics d'écrire dans le flux en votre nom.

Table des matières

ResourceARN

ARN du flux de diffusion Amazon Kinesis Firehose de destination dans lequel écrire.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour écrire dans le flux de destination en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisFirehoseOutputDescription

Pour une sortie d'application, décrit le flux de diffusion Amazon Kinesis Firehose configuré en tant que destination.

Table des matières

ResourceARN

Amazon Resource Name (ARN) du flux de diffusion Amazon Kinesis Firehose.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisFirehoseOutputUpdate

Lorsque vous mettez à jour une configuration de sortie à l'aide de cette UpdateApplicationopération, fournit des informations sur un flux de diffusion Amazon Kinesis Firehose configuré comme destination.

Table des matières

ResourceARNUpdate

Amazon Resource Name (ARN) du flux de diffusion Amazon Kinesis Firehose dans lequel écrire.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisStreamsInput

Identifie un flux Amazon Kinesis en tant que source de diffusion. Vous fournissez l'Amazon Resource Name (ARN) du flux et l'ARN d'un rôle IAM qui permet à Amazon Kinesis Analytics d'accéder au flux en votre nom.

Table des matières

ResourceARN

ARN du flux Amazon Kinesis d'entrée à lire.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisStreamsInputDescription

Décrit le flux Amazon Kinesis configuré en tant que source de streaming dans la configuration d'entrée de l'application.

Table des matières

ResourceARN

Amazon Resource Name (ARN) du flux Amazon Kinesis.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisStreamsInputUpdate

Lors de la mise à jour d'une configuration d'entrée d'application, fournit des informations sur un flux Amazon Kinesis utilisé en tant que source de streaming.

Table des matières

ResourceARNUpdate

Amazon Resource Name (ARN) du flux Amazon Kinesis d'entrée à lire.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisStreamsOutput

Lors de la configuration d'une sortie d'application, identifie un flux Amazon Kinesis en tant que destination. Vous fournissez l'Amazon Resource Name (ARN) du flux et l'ARN d'un rôle IAM qu'Amazon Kinesis Analytics peut utiliser pour écrire dans le flux en votre nom.

Table des matières

ResourceARN

ARN du flux Amazon Kinesis de destination dans lequel écrire.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour écrire dans le flux de destination en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisStreamsOutputDescription

Pour une sortie d'application, décrit le flux Amazon Kinesis configuré en tant que destination.

Table des matières

ResourceARN

Amazon Resource Name (ARN) du flux Amazon Kinesis.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

KinesisStreamsOutputUpdate

Lorsque vous mettez à jour une configuration de sortie à l'aide de cette UpdateApplicationopération, fournit des informations sur un flux Amazon Kinesis configuré comme destination.

Table des matières

ResourceARNUpdate

Amazon Resource Name (ARN) du flux Amazon Kinesis dans lequel vous souhaitez écrire la sortie.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour accéder au flux en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

LambdaOutput

Lors de la configuration de la sortie de l'application, identifie une fonction AWS Lambda comme destination. Vous fournissez l'Amazon Resource Name (ARN) de la fonction et l'ARN d'un rôle IAM qu'Amazon Kinesis Analytics peut utiliser pour écrire dans la fonction en votre nom.

Table des matières

ResourceARN

Amazon Resource Name (ARN) de la fonction Lambda de destination dans laquelle écrire.

> **Note**
>
> Pour spécifier une version antérieure à la dernière version de la fonction Lambda, incluez la version de la fonction dans l'ARN de la fonction Lambda. Pour plus d'informations sur les ARN Lambda, voir Exemples d'ARN : Lambda AWS

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour écrire dans la fonction de destination en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

LambdaOutputDescription

Pour une sortie d'application, décrit la fonction AWS Lambda configurée comme destination.

Table des matières

ResourceARN

Amazon Resource Name (ARN) de la fonction Lambda de destination.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour écrire dans la fonction de destination.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

LambdaOutputUpdate

Lorsque vous mettez à jour une configuration de sortie à l'aide de l'UpdateApplicationopération, fournit des informations sur une fonction AWS Lambda configurée comme destination.

Table des matières

ResourceARNUpdate

Amazon Resource Name (ARN) de la fonction Lambda de destination.

> **Note**
>
> Pour spécifier une version antérieure à la dernière version de la fonction Lambda, incluez la version de la fonction dans l'ARN de la fonction Lambda. Pour plus d'informations sur les ARN Lambda, voir Exemples d'ARN : Lambda AWS

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

RoleARNUpdate

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour écrire dans la fonction de destination en votre nom. Vous devez accorder les autorisations nécessaires à ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

MappingParameters

Lors de la création ou de la mise à jour d'une application, lorsque l'entrée de l'application est configurée, fournit des informations de mappage supplémentaires propres au format d'enregistrement (par exemple JSON, CSV ou des champs d'enregistrement délimités par un délimiteur) sur la source de diffusion.

Table des matières

CSVMappingParameters

Fournit des informations de mappage supplémentaires lorsque le format d'enregistrement utilise des délimiteurs (par exemple, CSV).

Type : objet CSVMappingParameters

Obligatoire : non

JSONMappingParameters

Fournit des informations de mappage supplémentaires lorsque JSON est le format d'enregistrement utilisé sur la source de diffusion.

Type : objet JSONMappingParameters

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Output

Décrit la configuration de sortie d'application dans laquelle vous identifiez un flux intégré à l'application et une destination dans laquelle vous souhaitez que les données du flux intégré à l'application soient écrites. La destination peut être un flux Amazon Kinesis ou un flux de diffusion Amazon Kinesis Firehose.

Pour plus d'informations sur le nombre de destinations sur lesquelles une application peut écrire et sur les autres limites, consultez Limites.

Table des matières

DestinationSchema

Décrit le format de données utilisé pour écrire les enregistrements dans la destination. Pour plus d'informations, consultez Configuration de la sortie d'application.

Type : objet DestinationSchema

Obligatoire : oui

Name

Nom du flux intégré à l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : oui

KinesisFirehoseOutput

Identifie un flux de diffusion Amazon Kinesis Firehose en tant que destination.

Type : objet KinesisFirehoseOutput

Obligatoire : non

KinesisStreamsOutput

Identifie un flux Amazon Kinesis en tant que destination.

Type : objet KinesisStreamsOutput

Obligatoire : non

LambdaOutput

Identifie une fonction AWS Lambda comme destination.

Type : objet LambdaOutput

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

OutputDescription

Décrit la configuration de sortie de l'application, qui inclut le nom du flux intégré à l'application et la destination dans laquelle les données du flux sont écrites. La destination peut être un flux Amazon Kinesis ou un flux de diffusion Amazon Kinesis Firehose.

Table des matières

DestinationSchema

Format de données utilisé pour écrire des données dans la destination.

Type : objet DestinationSchema

Obligatoire : non

KinesisFirehoseOutputDescription

Décrit le flux de diffusion Amazon Kinesis Firehose configuré en tant que destination d'écriture d'une sortie.

Type : objet KinesisFirehoseOutputDescription

Obligatoire : non

KinesisStreamsOutputDescription

Décrit le flux Amazon Kinesis configuré en tant que destination d'écriture d'une sortie.

Type : objet KinesisStreamsOutputDescription

Obligatoire : non

LambdaOutputDescription

Décrit la fonction AWS Lambda configurée comme destination où la sortie est écrite.

Type : objet LambdaOutputDescription

Obligatoire : non

Name

Nom du flux intégré à l'application configuré en sortie.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : non

OutputId

Identifiant unique pour la configuration de sortie.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

OutputUpdate

Décrit les mises à jour de la configuration de sortie identifiée par l'`OutputId`.

Table des matières

OutputId

Identifie la configuration de sortie spécifique que vous souhaitez mettre à jour.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

DestinationSchemaUpdate

Décrit le format de données utilisé pour écrire les enregistrements dans la destination. Pour plus d'informations, consultez Configuration de la sortie d'application.

Type : objet DestinationSchema

Obligatoire : non

KinesisFirehoseOutputUpdate

Décrit un flux de diffusion Amazon Kinesis Firehose en tant que destination de la sortie.

Type : objet KinesisFirehoseOutputUpdate

Obligatoire : non

KinesisStreamsOutputUpdate

Décrit un flux Amazon Kinesis en tant que destination de la sortie.

Type : objet KinesisStreamsOutputUpdate

Obligatoire : non

LambdaOutputUpdate

Décrit une fonction AWS Lambda en tant que destination de la sortie.

Type : objet LambdaOutputUpdate

Obligatoire : non

NameUpdate

Si vous souhaitez spécifier un flux intégré à l'application différent pour cette configuration de sortie, utilisez ce champ pour spécifier le nouveau nom du flux intégré à l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

RecordColumn

Décrit le mappage de chaque élément de données de la source de streaming à la colonne correspondante du flux intégré à l'application.

Egalement utilisé pour décrire le format de la source de données de référence.

Table des matières

Name

Nom de la colonne créée dans le flux d'entrée intégré à l'application ou la table de référence.

Type : chaîne

Obligatoire : oui

SqlType

Type de colonne créé dans le flux d'entrée intégré à l'application ou la table de référence.

Type : chaîne

Contraintes de longueur : longueur minimum de 1.

Obligatoire : oui

Mapping

Référence à l'élément de données dans l'entrée de diffusion ou la source de données de référence. Cet élément est obligatoire si c'RecordFormatTypeest le cas`JSON`.

Type : chaîne

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2

- AWS SDK pour Ruby V3

RecordFormat

Décrit le format d'enregistrement et les informations relatives au mappage qui doivent être appliquées pour schématiser les enregistrements présents dans le flux.

Table des matières

RecordFormatType

Type de format d'enregistrement.

Type : chaîne

Valeurs valides : `JSON | CSV`

Obligatoire : oui

MappingParameters

Lors de la création ou de la mise à jour d'une application, lorsque l'entrée de l'application est configurée, fournit des informations de mappage supplémentaires propres au format d'enregistrement (par exemple JSON, CSV ou des champs d'enregistrement délimités par un délimiteur) sur la source de diffusion.

Type : objet MappingParameters

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ReferenceDataSource

Décrit la source des données de référence en fournissant les informations de cette source (nom de compartiment S3 et nom de la clé d'objet), le nom de la table intégrée à l'application qui est créée et le schéma nécessaire pour mapper les éléments de données de l'objet Amazon S3 à la table intégrée à l'application.

Table des matières

ReferenceSchema

Décrit le format des données de la source de diffusion et la manière dont chaque élément de données est mappé aux colonnes correspondantes qui sont créées dans le flux intégré à l'application.

Type : objet SourceSchema

Obligatoire : oui

TableName

Nom de la table intégrée à l'application à créer.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : oui

S3ReferenceDataSource

Identifie le compartiment et l'objet S3 qui contient les données de référence. Identifie également le rôle IAM que peut endosser Amazon Kinesis Analytics pour lire cet objet en votre nom. Une application Amazon Kinesis Analytics ne charge les données de référence qu'une seule fois. Si les données sont modifiées, vous appelez l'opération `UpdateApplication` pour déclencher le rechargement des données dans votre application.

Type : objet S3ReferenceDataSource

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ReferenceDataSourceDescription

Décrit la source de données de référence configurée pour une application.

Table des matières

ReferenceId

ID de la source de données de référence. Il s'agit de l'ID attribué par Amazon Kinesis Analytics lorsque vous ajoutez la source de données de référence à votre application à l'aide AddApplicationReferenceDataSourcede l'opération.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

S3ReferenceDataSourceDescription

Fournit le nom du compartiment S3, le nom de clé d'objet qui contient les données de référence. Il fournit également l'Amazon Resource Name (ARN) du rôle IAM qu'Amazon Kinesis Analytics peut assumer pour lire l'objet Amazon S3 et remplir le tableau de référence intégré à l'application.

Type : objet S3ReferenceDataSourceDescription

Obligatoire : oui

TableName

Nom du tableau intégré à l'application créé par la configuration de source de données de référence spécifique.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : oui

ReferenceSchema

Décrit le format des données de la source de diffusion et la manière dont chaque élément de données est mappé aux colonnes correspondantes qui sont créées dans le flux intégré à l'application.

Type : objet SourceSchema

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

ReferenceDataSourceUpdate

Lorsque vous mettez à jour la configuration d'une source de données de référence pour une application, cet objet fournit toutes les valeurs mises à jour (telles que le nom de compartiment source et le nom de clé d'objet), le nom du tableau intégré à l'application qui est créé et les informations de mappage mises à jour qui mappent les données de l'objet Amazon S3 au tableau de référence intégré à l'application qui est créé.

Table des matières

ReferenceId

ID de la source de données de référence en cours de mise à jour. Vous pouvez utiliser l'DescribeApplicationopération pour obtenir cette valeur.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 50.

Modèle : `[a-zA-Z0-9_.-]+`

Obligatoire : oui

ReferenceSchemaUpdate

Décrit le format des données de la source de diffusion et la manière dont chaque élément de données est mappé aux colonnes correspondantes qui sont créées dans le flux intégré à l'application.

Type : objet SourceSchema

Obligatoire : non

S3ReferenceDataSourceUpdate

Décrit le nom du compartiment S3, le nom de clé d'objet et le rôle IAM qu'Amazon Kinesis Analytics peut assumer pour lire l'objet Amazon S3 en votre nom et remplir le tableau de référence intégré à l'application.

Type : objet S3ReferenceDataSourceUpdate

Obligatoire : non

TableNameUpdate

Nom de la table intégrée à l'application créée par cette mise à jour.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 32.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

S3Configuration

Fournit une description d'une source de données Amazon S3, y compris l'Amazon Resource Name (ARN) du compartiment S3, l'ARN du rôle IAM qui est utilisé pour accéder au compartiment et le nom de l'objet Amazon S3 qui contient les données.

Table des matières

BucketARN

ARN du compartiment S3 qui contient les données.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

FileKey

Nom de l'objet qui contient les données.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 1024.

Obligatoire : oui

RoleARN

ARN IAM du rôle utilisé pour accéder aux données.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

S3ReferenceDataSource

Identifie le compartiment et l'objet S3 qui contient les données de référence. Identifie également le rôle IAM que peut endosser Amazon Kinesis Analytics pour lire cet objet en votre nom.

Une application Amazon Kinesis Analytics ne charge les données de référence qu'une seule fois. Si les données changent, vous appelez l'UpdateApplicationopération pour déclencher le rechargement des données dans votre application.

Table des matières

BucketARN

Amazon Resource Name (ARN) du compartiment S3.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

FileKey

Nom de la clé d'objet contenant les données de référence.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 1024.

Obligatoire : oui

ReferenceRoleARN

ARN du rôle IAM que le service peut endosser pour lire les données en votre nom. Ce rôle doit avoir l'autorisation d'effectuer l'action `s3:GetObject` sur l'objet et la politique d'approbation qui permet au principal du service Amazon Kinesis Analytics d'assumer ce rôle.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

S3ReferenceDataSourceDescription

Fournit le nom de compartiment et le nom de clé d'objet qui stockent les données de référence.

Table des matières

BucketARN

Amazon Resource Name (ARN) du compartiment S3.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

FileKey

Nom de clé d'objet Amazon S3.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 1024.

Obligatoire : oui

ReferenceRoleARN

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour lire l'objet Amazon S3 en votre nom afin de remplir le tableau de référence intégré à l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : oui

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

S3ReferenceDataSourceUpdate

Décrit le nom du compartiment S3, le nom de clé d'objet et le rôle IAM qu'Amazon Kinesis Analytics peut assumer pour lire l'objet Amazon S3 en votre nom et remplir le tableau de référence intégré à l'application.

Table des matières

BucketARNUpdate

Amazon Resource Name (ARN) du compartiment S3.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

FileKeyUpdate

Noms de clé d'objet.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 1024.

Obligatoire : non

ReferenceRoleARNUpdate

ARN du rôle IAM que peut endosser Amazon Kinesis Analytics pour lire l'objet Amazon S3 et remplir le tableau de référence intégré dans l'application.

Type : chaîne

Contraintes de longueur : longueur minimum de 1. Longueur maximale de 2048.

Modèle : `arn:.*`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

SourceSchema

Décrit le format des données de la source de diffusion et la manière dont chaque élément de données est mappé aux colonnes correspondantes qui sont créées dans le flux intégré à l'application.

Table des matières

RecordColumns

Liste d'objets `RecordColumn`.

Type : tableau d'objets RecordColumn

Membres du tableau : Nombre minimum de 1 élément. Nombre maximum de 1 000 éléments.

Obligatoire : oui

RecordFormat

Spécifie le format des enregistrements présents dans la source de diffusion.

Type : objet RecordFormat

Obligatoire : oui

RecordEncoding

Indique l'encodage des enregistrements dans la source de diffusion. Par exemple, UTF-8.

Type : chaîne

Modèle : `UTF-8`

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Tag

Une paire clé-valeur (la valeur est facultative) que vous pouvez définir et affecter aux AWS ressources. Si vous spécifiez une balise qui existe déjà, la valeur de la balise est remplacée par la valeur que vous spécifiez dans la requête. Notez que le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50. Pour plus d'informations, consultez la section Utilisation du balisage.

Table des matières

Key

Clé de la balise clé-valeur.

Type : chaîne

Contraintes de longueur : Longueur minimum de 1. Longueur maximale de 128.

Obligatoire : oui

Value

Valeur de la balise clé-valeur. La valeur est facultative.

Type : chaîne

Contraintes de longueur : Longueur minimum de 0. Longueur maximale de 256.

Obligatoire : non

consultez aussi

Pour plus d'informations sur l'utilisation de cette API dans l'un des AWS SDK spécifiques au langage, consultez les pages suivantes :

- AWS SDK pour C++
- AWS SDK pour Java V2
- AWS SDK pour Ruby V3

Historique du document pour Amazon Kinesis Data Analytics

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version d'Amazon Kinesis Data Analytics.

- Version de l'API : 2015-08-14
- Dernière mise à jour de la documentation : 8 mai 2019

Modification	Description	Date
Balisage des applications Kinesis Data Analytics	Utilisez le balisage d'application pour déterminer les coûts par application, pour contrôler les accès, ou à des fins définis par l'utilisateur. Pour de plus amples informations, veuillez consulter Utilisation du balisage.	8 mai 2019
Journalisation des appels d'API Kinesis Data Analytics avec AWS CloudTrail	Amazon Kinesis Data Analytics est intégrée avec AWS CloudTrail, un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un service AWS dans Kinesis Data Analytics. Pour de plus amples informations, veuillez consulter Utiliser AWS CloudTrail.	22 mars 2019
Kinesis Data Analytics est disponible dans la région de Francfort	Kinesis Analytics est désormais disponible dans la région Europe (Francfort). Pour plus d'informations, consultez et points de	18 juillet 2018

Modification	Description	Date
	terminaison : Kinesis Data Analytics.	
Utilisation des données de référence de la console	Vous pouvez dès à présent travailler avec les données de référence de l'application de la console. Pour plus d'informations, consultez Exemple : ajout de données de référence à une application Kinesis Data Analytics.	13 juillet 2018
Exemples de requêtes à fenêtres	Exemples d'applications pour fenêtres et regroupement. Pour plus d'informations, consultez Exemples : Fenêtres et Regroupement.	9 juillet 2018
Test des applications	Conseils pour tester les modifications apportées au schéma et au code d'application. Pour plus d'informations, consultez Test des applications.	3 juillet 2018
Exemples d'applications pour le prétraitement des données	Autres exemples de code pour REGEX_LOG_PARSE, REGEX_REPLACE et les opérateurs DateTime. Pour plus d'informations, consultez Exemples : Transformation de données.	18 mai 2018

Modification	Description	Date
Augmentation de la taille des lignes renvoyées et du code SQL	La taille maximale d'une ligne renvoyée passe à 512 Ko, et la taille maximale du code SQL dans une application passe à 100 Ko. Pour de plus amples informations, veuillez consulter Limites.	2 mai 2018
Exemples de fonctions AWS Lambda dans Java et .NET.	Exemples de code pour la création de fonctions Lambda pour le prétraite ment d'enregistrements et les destinations d'applica tion. Pour plus d'informa tions, consultez Création de fonctions Lambda pour le prétraitement et Création de fonctions Lambda pour des destinations d'application.	22 mars 2018
Nouvelle fonction `HOTSPOTS`	Recherchez et renvoyez des informations sur les régions relativement denses de vos données. Pour de plus amples informations, veuillez consulter Exemple : Détection des points chauds sur un flux (fonction HOTSPOTS).	19 mars 2018

Modification	Description	Date
Fonction Lambda en tant que destination	Envoyer des résultats d'analyse à une fonction Lambda en tant que destination. Pour de plus amples informations, veuillez consulter Utilisation d'une fonction Lambda en tant que sortie.	20 décembre 2017
Nouvelle fonction `RANDOM_CUT_FOREST_WITH_EXPLANATION`	Identifier les champs qui contribuent à un score d'anomalie score dans un flux de données. Pour de plus amples informations, veuillez consulter Exemple : détection d'anomalies de données et message d'explication (fonction RANDOM_CUT_FOREST_WITH_EXPLANATION).	2 novembre 2017
Découverte de schéma sur des données statiques	Exécuter une découverte de schéma sur des données statiques stockées dans un compartiment Amazon S3. Pour de plus amples informations, veuillez consulter Utilisation de la fonction de découverte de schéma sur des données statiques.	6 octobre 2017

Modification	Description	Date
Fonction de prétraitement Lambda	Prétraiter des enregistrements dans un flux d'entrée avec AWS Lambda avant de procéder à une analyse. Pour de plus amples informations, veuillez consulter Prétraitement des données à l'aide d'une fonction Lambda.	6 octobre 2017
Applications Auto Scaling	Augmenter automatiquement le débit de données de votre application à l'aide de la fonction de scalabilité automatique. Pour de plus amples informations, veuillez consulter Dimensionnement automatique des applications pour augmenter le débit.	13 septembre 2017
Plusieurs flux d'entrée intégrés à l'application	Augmenter le débit de l'application avec plusieurs flux de données intégrés à l'application. Pour de plus amples informations, veuillez consulter Mise en parallèle des flux d'entrée pour un débit accru.	29 juin 2017

Modification	Description	Date
Guide d'utilisation d'AWS Management Console pour Kinesis Data Analytics	Modifier un schéma déduit et un code SQL à l'aide de l'éditeur de schémas et de l'éditeur SQL dans la console Kinesis Data Analytics. Pour de plus amples informations, veuillez consulter Étape 4 (facultatif) Modification du schéma et du code SQL à l'aide de la console.	7 avril 2017
Publication	Version publique du Guide du développeur Amazon Kinesis Data Analytics.	11 août 2016
Version préliminaire	Version préliminaire du Guide du développeur Amazon Kinesis Data Analytics.	29 janvier 2016

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le Glossaire AWS dans la Référence Glossaire AWS.