aws

SQL Entwicklerhandbuch

Amazon-Kinesis-Data-Analytics für SQL-Anwendungen

Amazon-Kinesis-Data-Analytics für SQL-Anwendungen: SQL Entwicklerhandbuch

Table of Contents

Nach reiflicher Überlegung haben wir uns entschieden, Amazon Kinesis Data Analytics für SQL-Anwendungen einzustellen:

1. Ab dem 1. September 2025 werden wir keine Bugfixes für Amazon Kinesis Data Analytics for SQL-Anwendungen bereitstellen, da wir aufgrund der bevorstehenden Einstellung nur eingeschränkten Support dafür haben werden.

2. Ab dem 15. Oktober 2025 können Sie keine neuen Kinesis Data Analytics for SQL-Anwendungen mehr erstellen.

3. Wir werden Ihre Anwendungen ab dem 27. Januar 2026 löschen. Sie können Ihre Amazon Kinesis Data Analytics for SQL-Anwendungen nicht starten oder betreiben. Ab diesem Zeitpunkt ist kein Support mehr für Amazon Kinesis Data Analytics for SQL verfügbar. Weitere Informationen finden Sie unter Einstellung von Amazon Kinesis Data Analytics für SQL-Anwendungen.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Einstellung von Amazon Kinesis Data Analytics für SQL-Anwendungen

Plan zur Einstellung

Nach reiflicher Überlegung haben wir die Entscheidung getroffen, Amazon Kinesis Data Analytics für SQL-Anwendungen einzustellen. Um Ihnen bei der Planung und Migration weg von Amazon Kinesis Data Analytics for SQL-Anwendungen zu helfen, werden wir das Angebot über einen Zeitraum von 15 Monaten schrittweise einstellen. Dies sind wichtige Daten, die es zu beachten gilt: 1. September 2025, 15. Oktober 2025 und 27. Januar 2026.

1. Am 15. Oktober 2025 werden wir Ihre Bewerbungen einstellen und sie einem `READY` Status zuordnen. Sie können Ihre Anwendungen zu diesem Zeitpunkt neu starten und Ihre Anwendungen weiterhin wie gewohnt verwenden, vorbehaltlich der Servicebeschränkungen.
2. Ab dem 15. Oktober 2025 können Sie keine neuen Amazon Kinesis Data Analytics for SQL-Anwendungen mehr erstellen. Sie können alle vorhandenen Anwendungen wie gewohnt ausführen, vorbehaltlich der Servicebeschränkungen.
3. Wir werden Ihre Bewerbungen ab dem 27. Januar 2026 löschen. Sie können Ihre Amazon Kinesis Data Analytics for SQL-Anwendungen nicht starten oder betreiben. Ab diesem Zeitpunkt ist kein Support mehr für Amazon Kinesis Data Analytics for SQL-Anwendungen verfügbar.

Wir empfehlen Ihnen, Ihre Anwendungen vor dem 15. Oktober 2025 auf Amazon Managed Service für Apache Flink oder Amazon Managed Service für Apache Flink Studio zu migrieren. Ressourcen, die Sie bei Ihrer Migration unterstützen, finden Sie unter. Beispiel für die Migration zu einem Managed Service für Apache Flink Studio Weitere Informationen über Amazon Managed Service für Apache Flink oder Amazon Managed Service für Apache Flink Studio finden Sie im Amazon Managed Service for Apache Flink Developer Guide.

> **Note**
>
> Um eine vollständige Liste Ihrer Amazon Kinesis Data Analytics for SQL-Anwendungen anzuzeigen, können Sie die `ListApplications` API aufrufen. Weitere Informationen finden Sie unter ListApplications.

Migration zu Amazon Managed Service für Apache Flink Studio

Weitere Informationen zur Migration Ihrer Anwendungen sowie Code- und Architekturbeispiele finden Sie unter. Beispiel für die Migration zu einem Managed Service für Apache Flink Studio

Häufig gestellte Fragen

Warum bieten Sie Amazon Kinesis Data Analytics für SQL-Anwendungen nicht mehr an?

Wir haben festgestellt, dass Kunden Amazon Managed Service für Apache Flink-Angebote für Workloads zur Verarbeitung von Datenströmen in Echtzeit bevorzugen. Amazon Managed Service für Apache Flink ist ein serverloser, hoch skalierbarer und verfügbarer Echtzeit-Streamverarbeitungsservice mit Apache Flink, einer Open-Source-Engine für die Verarbeitung von Datenströmen. Amazon Managed Service für Apache Flink bietet Funktionen wie native Skalierung, Semantik zur Exact-Once-Verarbeitung, Unterstützung mehrerer Sprachen (einschließlich SQL), über 40 Quell- und Ziel-Konnektoren, dauerhaften Anwendungsstatus und mehr. Diese Funktionen helfen Kunden dabei, durchgängige Streaming-Pipelines aufzubauen und die Genauigkeit und Aktualität der Daten sicherzustellen.

Welche Optionen stehen den Kunden jetzt zur Verfügung?

Wir empfehlen Kunden, ihre bestehenden Amazon Kinesis Data Analytics for SQL-Anwendungen entweder auf Amazon Managed Service für Apache Flink Studio oder Amazon Managed Service für Apache Flink aufzurüsten. In Amazon Managed Service für Apache Flink Studio erstellen Kunden Abfragen mit SQL, Python oder Scala mithilfe interaktiver Notebooks. Für lang andauernde Anwendungen in Amazon Kinesis Data Analytics for SQL empfehlen wir Amazon Managed Service für Apache Flink, wo Kunden Anwendungen mit Java, Python, Scala und Embedded SQL mit allen Apache Flinks APIs, Konnektoren und mehr erstellen können.

Wie viele Kunden verwenden Amazon Kinesis Data Analytics für SQL-Anwendungen?

Wir können keine Kundeninformationen weitergeben.

Wie führen Kunden ein Upgrade von Amazon Kinesis Data Analytics für SQL-Anwendungen auf ein Amazon Managed Service für Apache Flink-Angebot durch?

Um ein Upgrade auf Amazon Managed Service für Apache Flink oder Amazon Managed Service für Apache Flink Studio durchzuführen, müssen Kunden ihre Anwendung neu erstellen. Um Ihnen zu helfen, haben wir eine Bibliothek mit gängigen SQL-Abfragen bereitgestellt und erfahren, wie Sie

diese in Amazon Managed Service für Apache Flink Studio neu schreiben können. Siehe Replizieren von Kinesis Data Analytics for SQL-Abfragen in Managed Service für Apache Flink Studio. Wir haben auch gängige Musterarchitekturen bereitgestellt, denen Kunden folgen können, wenn sie Anwendungen mit langer Laufzeit entwickeln oder maschinelles Lernen in Amazon Managed Service für Apache Flink verwenden. Siehe Kinesis Data Firehose als Quelle durch Kinesis Data Streams ersetzen

Weitere Informationen über Amazon Managed Service für Apache Flink finden Sie unter Amazon Managed Service für Apache Flink.

Anleitungen zur Migration finden Sie unter. Beispiel für die Migration zu einem Managed Service für Apache Flink Studio

Unterstützt Amazon Managed Service für Apache Flink die bestehenden Anwendungsfunktionen von Amazon Kinesis Data Analytics for SQL?

Amazon Managed Service für Apache Flink unterstützt viele der in Amazon Kinesis Data Analytics for SQL-Anwendungen verfügbaren Konzepte wie Konnektoren und Windowing sowie Funktionen, die in Amazon Kinesis Data Analytics für SQL-Anwendungen nicht verfügbar waren, wie native Skalierung, Semantik für die Exactly-Once-Verarbeitung, Unterstützung mehrerer Sprachen (einschließlich SQL), über 40 Quell- und Ziel-Connectors, dauerhaften Anwendungsstatus und mehr.

Beispiel für die Migration zu einem Managed Service für Apache Flink Studio

Nach reiflicher Überlegung haben wir die Entscheidung getroffen, Amazon Kinesis Data Analytics für SQL-Anwendungen einzustellen. Um Ihnen bei der Planung und Migration weg von Amazon Kinesis Data Analytics for SQL-Anwendungen zu helfen, werden wir das Angebot über einen Zeitraum von 15 Monaten schrittweise einstellen. Dies sind wichtige Daten, die es zu beachten gilt: 1. September 2025, 15. Oktober 2025 und 27. Januar 2026.

1. Ab dem 1. September 2025 werden wir keine Bugfixes für Amazon Kinesis Data Analytics for SQL-Anwendungen bereitstellen, da wir aufgrund der bevorstehenden Einstellung nur eingeschränkten Support dafür haben werden.
2. Ab dem 15. Oktober 2025 können Sie keine neuen Amazon Kinesis Data Analytics for SQL-Anwendungen mehr erstellen.
3. Wir werden Ihre Anwendungen ab dem 27. Januar 2026 löschen. Sie können Ihre Amazon Kinesis Data Analytics for SQL-Anwendungen nicht starten oder betreiben. Ab diesem Zeitpunkt ist kein Support mehr für Amazon Kinesis Data Analytics for SQL-Anwendungen verfügbar. Weitere Informationen hierzu finden Sie unter Einstellung von Amazon Kinesis Data Analytics für SQL-Anwendungen.

Wir empfehlen Ihnen, Amazon Managed Service für Apache Flink zu verwenden. Es kombiniert Benutzerfreundlichkeit mit fortschrittlichen Analysefunktionen, sodass Sie Anwendungen zur Stream-Verarbeitung in wenigen Minuten erstellen können.

Dieser Abschnitt enthält Code- und Architekturbeispiele, die Ihnen helfen, Ihre Amazon Kinesis Data Analytics for SQL-Anwendungs-Workloads auf Managed Service for Apache Flink zu migrieren.

Weitere Informationen finden Sie auch in diesem AWS Blogbeitrag: Migration von Amazon Kinesis Data Analytics for SQL Applications zu Managed Service for Apache Flink Studio.

Replizieren von Kinesis Data Analytics for SQL-Abfragen in Managed Service für Apache Flink Studio

Für die Migration Ihrer Workloads auf Managed Service für Apache Flink Studio oder Managed Service für Apache Flink finden Sie in diesem Abschnitt Abfrageübersetzungen, die Sie für allgemeine Anwendungsfälle verwenden können.

Bevor Sie sich mit diesen Beispielen befassen, empfehlen wir Ihnen, zunächst den Artikel Verwenden eines Studio-Notebooks mit einem Managed Service für Apache Flink zu lesen.

Neuerstellung von Kinesis Data Analytics für SQL-Abfragen in Managed Service für Apache Flink Studio

Die folgenden Optionen bieten Übersetzungen gängiger SQL-basierter Kinesis Data Analytics Analytics-Anwendungsabfragen an Managed Service for Apache Flink Studio.

Mehrschrittige Anwendung

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;
```

DateTime Werte transformieren

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;
```

Einfache Warnungen

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;
```

Gedrosselte Warnungen

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;
```

Aggregieren von Teilergebnissen aus einer Abfrage

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;
```

Umwandeln von Zeichenfolgewerten

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;
```

Ersetzen einer Teilzeichenfolge mit 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;
```

Regex-Protokollanalyse

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;
```

DateTime Werte transformieren

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;
```

Fenster und Aggregation

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;
```

Rollierendes Fenster mit 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;
```

Die am häufigsten vorkommenden Werte werden abgerufen (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;
```

Ungefähre Top-K-Artikel

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
```

Analysieren von Web-Protokollen (Funktion 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;
```

Aufteilen von Zeichenfolgen auf mehrere Felder (Funktion 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;
```

Joins

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;
```

Fehler

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 von Random Cut Forest-Workloads

Wenn Sie Workloads, die Random Cut Forest verwenden, von Kinesis Analytics für SQL zu Managed Service für Apache Flink verschieben möchten, zeigt dieser AWS -Blogbeitrag, wie Sie Managed Service für Apache Flink verwenden, um einen Online-RCF-Algorithmus zur Erkennung von Anomalien auszuführen.

Kinesis Data Firehose als Quelle durch Kinesis Data Streams ersetzen

Eine vollständige Anleitung finden Sie unter Converting-KDAsql-/. KDAStudio

In der folgenden Übung ändern Sie Ihren Datenfluss, um Amazon-Managed-Service für Apache Flink Studio zu verwenden. Dies bedeutet auch, von Amazon-Kinesis-Data-Firehose zu Amazon-Kinesis-Data-Streams zu wechseln.

Zunächst stellen wir eine typische KDA-SQL-Architektur vor, bevor wir zeigen, wie Sie diese mithilfe von Amazon-Managed-Service für Apache Flink Studio und Amazon-Kinesis-Data-Streams ersetzen können. Alternativ können Sie die Vorlage auch hier starten: CloudFormation

Amazon-Kinesis-Data-Analytics-SQL und Amazon-Kinesis-Data-Firehose

Hier ist der SQL-Architekturfluss von Amazon-Kinesis-Data-Analytics:

Wir untersuchen zunächst die Einrichtung von Amazon-Kinesis-Data-Analytics-SQL und Amazon-Kinesis-Data-Firehose. Der Anwendungsfall ist ein Handelsmarkt, auf dem Handelsdaten,

einschließlich Börsenticker und Preise, aus externen Quellen in Amazon-Kinesis-Systeme gestreamt werden. Amazon Kinesis Data Analytics for SQL verwendet den Input-Stream, um Fensterabfragen wie Tumbling Window auszuführen, um das Handelsvolumen und den`min`,`max`, und `average` Handelspreis über ein einminütiges Fenster für jeden Börsenticker zu ermitteln.

Amazon-Kinesis-Data-Analytics-SQL ist so eingerichtet, dass es Daten aus der Amazon-Kinesis-Data-Firehose-API aufnimmt. Nach der Verarbeitung sendet Amazon-Kinesis-Data-Analytics-SQL die verarbeiteten Daten an eine andere Amazon-Kinesis-Data-Firehose, die dann die Ausgabe in einem Amazon-S3-Bucket speichert.

In diesem Fall verwenden Sie den Amazon-Kinesis-Datengenerator. Mit dem Amazon-Kinesis-Datengenerator können Sie Testdaten an Ihre Amazon-Kinesis-Data-Streams oder Amazon-Kinesis-Data-Firehose-Bereitstellungsstreams senden. Um zu beginnen, folgen Sie den Anweisungen hier. Verwenden Sie hier die CloudFormation Vorlage anstelle der in der Anleitung angegebenen Vorlage:.

Sobald Sie die CloudFormation Vorlage ausgeführt haben, enthält der Ausgabebereich die Amazon Kinesis Data Generator-URL. Melden Sie sich mit der Cognito-Benutzer-ID und dem Passwort, die Sie hier eingerichtet haben, beim Portal an. Wählen Sie die Region und den Namen des Zielstreams aus. Wählen Sie für den aktuellen Status die Bereitstellungsstreams von Amazon-Kinesis-Data-Firehose. Wählen Sie für den neuen Status den Namen des Amazon-Kinesis-Data-Firehose Streams. Sie können je nach Ihren Anforderungen mehrere Vorlagen erstellen und die Vorlage mithilfe der Schaltfläche Vorlage testen ausprobieren, bevor Sie sie an den Ziel-Stream senden.

Im Folgenden finden Sie ein Beispiel für eine Nutzlast mit Amazon-Kinesis-Datengenerator. Der Datengenerator zielt darauf ab, die eingegebenen Amazon-Kinesis-Firehose-Streams kontinuierlich zu streamen. Der Amazon-Kinesis-SDK-Client kann auch Daten von anderen Produzenten senden.

```
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
```

Der folgende JSON-Code wird verwendet, um eine zufällige Reihe von Handelszeiten und -daten, Börsentickerdaten und Aktienkursen zu generieren:

```
date.now(YYYY-MM-DD HH:mm:ss.SSS),
"random.arrayElement(["AAPL","AMZN","MSFT","META","GOOGL"])",
random.number(2000)
```

Sobald Sie Daten senden auswählen, beginnt der Generator mit dem Senden von Mock-Daten.

Externe Systeme streamen die Daten an Amazon-Kinesis-Data-Firehose. Mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen können Sie Streaming-Daten mithilfe von Standard-SQL analysieren. Der Service ermöglicht die Erstellung und Ausführung von SQL-Code für Streaming-Quellen zum Durchführen von Zeitreihenanalysen, Füllen von Echtzeit-Dashboards und Erstellen von Echtzeitmetriken. Amazon-Kinesis-Data-Analytics for SQL-Anwendungen könnte einen Ziel-Stream aus SQL-Abfragen im Eingabe-Stream erstellen und den Ziel-Stream an eine andere Amazon-Kinesis-Data-Firehose senden. Das Ziel Amazon-Kinesis-Data-Firehose könnte die Analysedaten als Endzustand an Amazon-S3 senden.

Der Legacy-Code von Amazon-Kinesis-Data-Analytics-SQL basiert auf einer Erweiterung des SQL-Standards.

Sie verwenden die folgende Abfrage in Amazon-Kinesis-Data-Analytics-SQL. Sie erstellen zunächst einen Ziel-Stream für die Abfrageausgabe. Dann verwenden Sie `PUMP`, ein Amazon-Kinesis-Data-Analytics-Repository-Objekt (eine Erweiterung des SQL-Standards), das eine kontinuierlich ablaufende `INSERT INTO stream SELECT ... FROM`-Abfragefunktion bietet und so die kontinuierliche Eingabe der Ergebnisse einer Abfrage in einen benannten Stream ermöglicht.

```
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);
```

Das vorangegangene SQL verwendet zwei Zeitfenster: `tradeTimestamp` das Zeitfenster stammt aus der Payload des eingehenden Streams und `ROWTIME.tradeTimestamp` wird auch oder genannt`Event Time`. `client-side time` Häufig ist es wünschenswert, diese Zeit in Analysen zu verwenden, da dies die Zeit ist, zu der ein Ereignis aufgetreten ist. Zahlreiche Ereignisquellen, wie Smartphones und Web-Clients, besitzen jedoch keine zuverlässigen Uhren, was zu ungenauen Zeiten führen kann. Zusätzlich können Konnektivitätsprobleme dazu führen, dass Datensätze in einem Stream nicht in der gleichen Reihenfolge angezeigt werden, in der sie aufgetreten sind.

In-App-Streams enthalten außerdem eine spezielle Spalte namens `ROWTIME`. In dieser wird ein Zeitstempel gespeichert, wenn Amazon-Kinesis-Data-Analytics eine Zeile in den ersten In-Application-Stream einfügt. `ROWTIME` spiegelt den Zeitstempel wider, zu dem Amazon-Kinesis-Data-Analytics nach dem Lesen aus der Streaming-Quelle einen Datensatz in den ersten In-Application-Stream eingefügt hat. Dieser `ROWTIME`-Wert wird anschließend in der gesamten Anwendung beibehalten.

Das SQL bestimmt die Anzahl der Ticker als`volume`, `minmax`, und den `average` Preis über ein 60-Sekunden-Intervall.

Die Verwendung dieser Zeiten in Abfragen mit Fenstern auf Zeitbasis hat Vor- und Nachteile. Wählen Sie eine oder mehrere dieser Zeiten und entwickeln Sie eine Strategie für den Umgang mit den relevanten Nachteilen, abhängig von Ihrem Anwendungsfall.

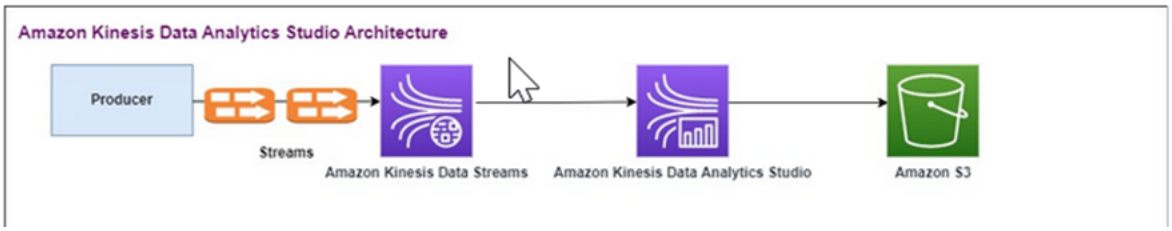
Eine Zwei-Fenster-Strategie mit zwei zeitbasierten Fenster verwendet sowohl `ROWTIME` als auch eine der beiden anderen Zeiten, beispielsweise die Ereigniszeit.

- Sie sollten `ROWTIME` als erstes Fenster verwenden, das die Häufigkeit steuert, mit der die Abfrage die Ergebnisse ausgibt, wie im folgenden Beispiel gezeigt. Sie wird nicht als logische Zeit verwendet.
- Sie sollten eine der beiden anderen Zeiten als logische Zeit verwenden, um sie mit Ihren Analysen zu verknüpfen. Diese Zeit stellt den Zeitpunkt dar, zu dem das Ereignis aufgetreten ist. Im folgenden Beispiel besteht das Ziel der Analyse darin, die Datensätze zu gruppieren und eine Zahl nach Ticker zurückzugeben.

Amazon-Managed-Service für Apache Flink

In der aktualisierten Architektur ersetzen Sie Amazon-Kinesis-Data-Firehose durch Amazon-Kinesis-Data-Streams . Amazon-Kinesis-Data-Analytics for SQL-Anwendungen wird durch Amazon-Managed-Service für Apache Flink Studio ersetzt. Apache Flink-Code wird interaktiv in einem Apache Zeppelin-Notebook ausgeführt. Amazon-Managed-Service für Apache Flink Studio sendet die aggregierten Handelsdaten an einen Amazon-S3-Bucket, um sie zu speichern. Die Schritte werden im Folgenden dargestellt:

Dies ist der Architekturfluss von Amazon-Managed-Service für Apache Flink Studio:



Erstellen Sie einen Kinesis Data Stream

So erstellen Sie einen Datenstrom mit der Konsole

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Erweitern Sie in der Navigationsleiste die Regionsauswahl und wählen Sie eine Region aus.
3. Klicken Sie auf Create data stream (Daten-Stream erstellen).
4. Geben Sie auf der Seite Kinesis-Stream erstellen einen Namen für Ihren Datenstrom ein und wählen Sie dann den standardmäßigen Kapazitätsmodus On-Demand.

   Im Modus On-Demand können Sie dann Kinesis-Stream erstellen wählen, um Ihren Datenstrom zu erstellen.

   Auf der Seite Kinesis streams (Kinesis-Streams) wird für den Wert Status des Streams Creating (Erstellen) angezeigt, während der Stream erstellt wird. Sobald der Stream verwendet werden kann, ändert sich der Wert von Status in Active (Aktiv).
5. Wählen Sie den Namen des Streams aus. Auf der Seite Stream Details (Stream-Details) wird eine Zusammenfassung der Stream-Konfiguration zusammen mit Überwachungsinformationen angezeigt.

6. Ändern Sie im Amazon Kinesis Data Generator den Stream/delivery Stream in den neuen Amazon Kinesis Data Streams: TRADE_SOURCE_STREAM.

   JSON und Nutzlast entsprechen denen, die Sie für Amazon-Kinesis-Data-Analytics-SQL verwendet haben. Verwenden Sie den Amazon-Kinesis-Datengenerator, um einige Beispiele für Handelsnutzdaten zu erstellen und verwenden Sie den TRADE_SOURCE_STREAM-Datenstrom als Ziel für diese Übung:

   ```
   {{date.now(YYYY-MM-DD HH:mm:ss.SSS)}},
   "{{random.arrayElement(["AAPL","AMZN","MSFT","META","GOOGL"])}}",
   {{random.number(2000)}}
   ```

7. AWS-Managementkonsole Gehen Sie zu Managed Service for Apache Flink und wählen Sie dann Create application.
8. Wählen Sie im Navigationsbereich links Studio-Notebooks aus und wählen Sie dann Studio-Notebook erstellen.
9. Geben Sie einen Namen für das Studio-Notebook ein.
10. Geben Sie unter AWS -Glue-Datenbank eine bestehende AWS Glue -Datenbank an, die die Metadaten für Ihre Quellen und Ziele definiert. Wenn Sie keine AWS Glue Datenbank haben, wählen Sie Create und gehen Sie wie folgt vor:

    a. Wählen Sie in der AWS Glue-Konsole im Menü auf der linken Seite unter Datenkatalog die Option Datenbanken aus.
    b. Wählen Sie Datenbank erstellen aus
    c. Geben Sie auf der Seite Datenbank erstellen einen Namen für die Datenbank ein. Wählen Sie im Abschnitt Standort – optional Amazon-S3 durchsuchen und dann den Amazon-S3-Bucket aus. Wenn noch keinen Amazon-S3-Bucket eingerichtet haben, können Sie diesen Schritt überspringen und später dazu zurückkehren.
    d. (Optional). Geben Sie eine Beschreibung für die Datenbank ein.
    e. Wählen Sie Datenbank erstellen aus.
11. Wählen Sie Notebook erstellen aus.
12. Sobald Ihr Notebook erstellt ist, wählen Sie Ausführen.
13. Sobald das Notebook erfolgreich gestartet wurde, starten Sie ein Zeppelin-Notebook, indem Sie In Apache Zeppelin öffnen wählen.
14. Wählen Sie auf der Seite Zeppelin-Notizbuch die Option Neue Notiz erstellen und geben Sie ihr einen Namen. MarketDataFeed

Der Flink-SQL-Code wird im Folgenden erklärt, aber zuerst einmal sieht ein Zeppelin-Notebook-Bildschirm so aus. Jedes Fenster im Notebook ist ein separater Codeblock und diese können einzeln ausgeführt werden.

Code bei Amazon-Managed-Service für Apache Flink

Amazon-Managed-Service für Apache Flink Studio verwendet Zeppelin Notebooks, um den Code auszuführen. In diesem Beispiel erfolgt die Zuordnung zum SSQL-Code, der auf Apache Flink 1.13 basiert. Der Code im Zeppelin-Notizbuch wird im Folgenden, Block für Block, angezeigt.

Bevor Sie Code in Ihrem Zeppelin Notebook ausführen, müssen die Flink-Konfigurationsbefehle ausgeführt werden. Wenn Sie nach dem Ausführen von Code (ssql, Python oder Scala) eine Konfigurationseinstellung ändern müssen, müssen Sie Ihr Notebook beenden und neu starten. In diesem Beispiel müssen Sie Checkpointing einrichten. Checkpointing ist erforderlich, damit Sie in Amazon-S3 Daten in eine Datei streamen können. Dadurch können Daten, die zu Amazon-S3 gestreamt werden, in eine Datei geleitet werden. Die folgende Anweisung legt das Intervall auf 5000 Millisekunden fest.

```
%flink.conf
execution.checkpointing.interval 5000
```

`%flink.conf` gibt an, dass es sich bei diesem Block um Konfigurationsanweisungen handelt. Weitere Informationen zur Flink-Konfiguration einschließlich Checkpointing finden Sie unter Apache Flink Checkpointing.

Die Eingabetabelle für die Quelle Amazon Kinesis Data Streams wird mit dem folgenden Flink-SSQL-Code erstellt. Beachten Sie, dass das `TRADE_TIME` Feld die vom Datengenerator date/time erstellten Daten speichert.

```
%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');
```

Sie können den Eingabestream mit dieser Anweisung anzeigen:

```
%flink.ssql(type=update)-- testing the source stream

select * from TRADE_SOURCE_STREAM;
```

Bevor Sie die aggregierten Daten an Amazon-S3 senden, können Sie sie direkt in Amazon-Managed-Service für Apache Flink Studio mit einer Auswahlabfrage im rollierenden Fenster anzeigen. Dadurch werden die Handelsdaten in einem Zeitfenster von einer Minute zusammengefasst. Beachten Sie, dass die %flink.ssql-Anweisung eine Bezeichnung (type=update) haben muss:

```
%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;
```

Sie können dann in Amazon-S3 eine Tabelle für das Ziel erstellen. Sie müssen ein Wasserzeichen verwenden. Ein Wasserzeichen ist eine Fortschrittsmetrik, die einen Zeitpunkt angibt, zu dem Sie sicher sind, dass keine verzögerten Ereignisse mehr eintreten werden. Das Wasserzeichen wird benötigt, damit verspätete Ankünfte berücksichtigt werden. Das Intervall von `‘5’ Second` ermöglicht es Handelsaktionen mit 5-sekündiger Verspätung in den Amazon-Kinesis Data Stream einzutreten und trotzdem aufgenommen zu werden, wenn sie einen Zeitstempel haben, der innerhalb des Fensters liegt. Weitere Informationen finden Sie unter Wasserzeichen generieren.

```
%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');
```

Diese Anweisung fügt die Daten in die TRADE_DESTINATION_S3 ein. TUMPLE_ROWTIME ist der Zeitstempel der inklusiven Obergrenze des rollierenden Fensters.

```
%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;
```

Lassen Sie Ihre Anweisung 10 bis 20 Minuten lang laufen, um einige Daten in Amazon-S3 zu sammeln. Brechen Sie dann Ihre Anweisung ab.

Dadurch wird die Datei in Amazon-S3 geschlossen, sodass sie angesehen werden kann.

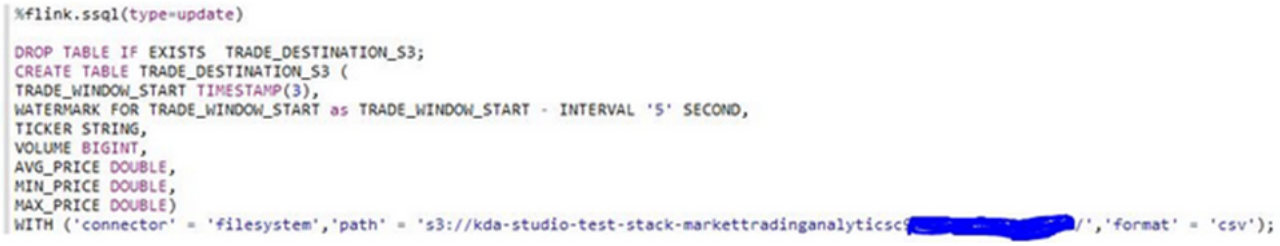
So sieht der Inhalt aus:

Sie können die CloudFormation -Vorlage verwenden, um die Infrastruktur zu erstellen.

CloudFormation erstellt die folgenden Ressourcen in Ihrem AWS Konto:

- Amazon-Kinesis-Data-Streams
- Amazon-Managed-Service für Apache Flink
- AWS Glue Datenbank
- Amazon-S3-Bucket
- IAM-Rollen und Richtlinien für den Zugriff auf geeignete Ressourcen durch Amazon-Managed-Service für Apache Flink Studio

Importieren Sie das Notizbuch und ändern Sie den Namen des Amazon S3 S3-Buckets durch den neuen Amazon S3 S3-Bucket, der von erstellt wurde CloudFormation.



Weitere Informationen

Im Folgenden finden Sie einige zusätzliche Ressourcen, mit denen Sie mehr über die Verwendung von Managed Service für Apache Flink Studio erfahren können:

- Entwicklerhandbuch für Managed Service für Apache Flink Studio-Notebooks
- Dokumentation für Apache Flink 1.13
- Workshop zu Managed Service für Apache Flink Studio
- Apache Flink Windowing
- Entwicklerhandbuch für Amazon-Kinesis-Data-Analytics – Aus einem Kinesis Data Analytics Stream in einen S3 Bucket schreiben

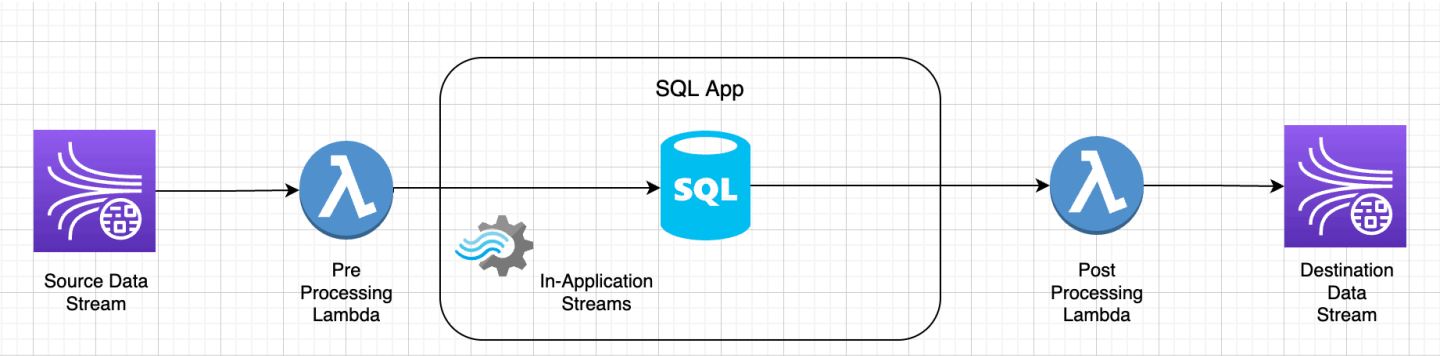
Nutzung benutzerdefinierter Funktionen () UDFs

Der Zweck des Musters besteht darin, zu demonstrieren, wie Zeppelin-Notebooks UDFs in Kinesis Data Analytics-Studio für die Verarbeitung von Daten im Kinesis-Stream genutzt werden können. Managed Service for Apache Flink Studio verwendet Apache Flink, um erweiterte Analysefunktionen bereitzustellen, darunter Semantik zur Exact-Once-Verarbeitung, Ereigniszeitfenster, Erweiterbarkeit durch benutzerdefinierte Funktionen und Kundenintegrationen, Unterstützung für wichtige Sprachen, dauerhaften Anwendungsstatus, horizontale Skalierung, Unterstützung mehrerer Datenquellen, erweiterbare Integrationen und mehr. Diese sind entscheidend für die Sicherstellung der Genauigkeit, Vollständigkeit, Konsistenz und Zuverlässigkeit der Verarbeitung von Datenströmen und sie sind in Amazon-Kinesis-Data-Analytics for SQL nicht verfügbar.

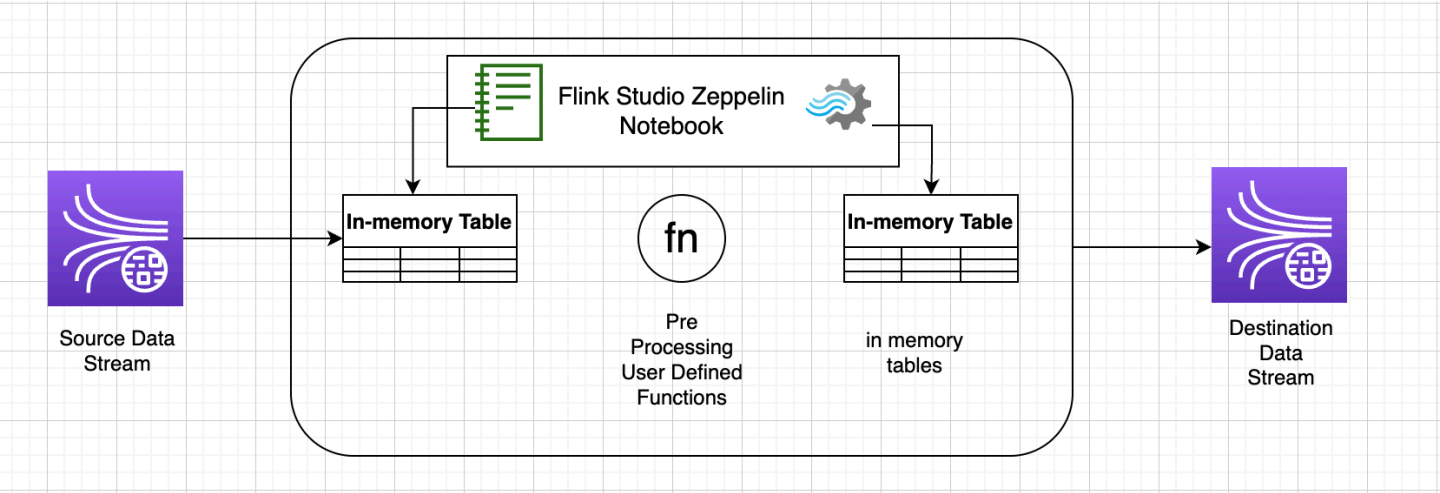
In dieser Beispielanwendung zeigen wir, wie Sie das Zeppelin-Notebook von KDA-Studio für die Verarbeitung von Daten im Kinesis-Stream nutzen UDFs können. Mit Studio-Notebooks für Kinesis Data Analytics können Sie Datenströme interaktiv in Echtzeit abfragen und auf einfache Weise Streamverarbeitungsanwendungen mit Standard-SQL, Python und Scala erstellen und ausführen. Mit ein paar Klicks können Sie ein serverloses Notebook starten AWS-Managementkonsole, um

Datenströme abzufragen und innerhalb von Sekunden Ergebnisse zu erhalten. Weitere Informationen finden Sie unter Verwenden eines Studio-Notebooks mit Kinesis Data Analytics for Apache Flink.

Lambda-Funktionen, die für die pre/post Verarbeitung von Daten in KDA-SQL-Anwendungen verwendet werden:



Benutzerdefinierte Funktionen für die pre/post Verarbeitung von Daten mit KDA-Studio Zeppelin-Notebooks



Benutzerdefinierte Funktionen () UDFs

Um gängige Geschäftslogik in einem Operator wiederzuverwenden, kann es nützlich sein, auf eine benutzerdefinierte Funktion zu verweisen, um Ihren Datenstrom zu transformieren. Dies kann entweder innerhalb des Managed Service für Apache Flink Studio-Notebooks oder als extern referenzierte Anwendungs-JAR-Datei erfolgen. Die Verwendung benutzerdefinierter Funktionen kann die Transformationen oder Datenanreicherungen vereinfachen, die Sie möglicherweise bei Streaming-Daten durchführen.

In Ihrem Notebook verweisen Sie auf eine einfache Java-Anwendungsdatei, die Funktionen zur Anonymisierung privater Telefonnummern bietet. Sie können auch Python oder Scala UDFs zur

Verwendung im Notizbuch schreiben. Wir haben uns für ein Anwendungs-Jar in Java entschieden, um die Funktionalität des Imports einer Anwendungs-Jar in ein Pyflink-Notebook hervorzuheben.

Einrichtung der Umgebung

Um dieser Anleitung zu folgen und mit Ihren Streaming-Daten zu interagieren, verwenden Sie ein AWS CloudFormation Skript, um die folgenden Ressourcen zu starten:

- Kinesis Data Streams als Quelle und Ziel
- Glue-Datenbank
- IAM-Rolle
- Managed Service für Apache Flink-Anwendung
- Lambda-Funktion zum Starten der Managed Service für Apache Flink Studio-Anwendung
- Lambda-Rolle zur Ausführung der vorherigen Lambda-Funktion
- Benutzerdefinierte Ressource zum Aufrufen der Lambda-Funktion

Laden Sie die CloudFormation Vorlage hier herunter.

Erstellen Sie den CloudFormation Stapel

1. Gehen Sie zu AWS-Managementkonsole und wählen Sie CloudFormationunter der Liste der Dienste aus.
2. Wählen Sie auf der CloudFormationSeite Stacks und dann Create Stack with new resources (Standard) aus.
3. Wählen Sie auf der Seite Stack erstellen die Option Eine Vorlagendatei hochladen und dann die Datei `kda-flink-udf.yml` aus, die Sie zuvor heruntergeladen haben. Laden Sie die Datei hoch und wählen Sie Weiter.
4. Geben Sie der Vorlage einen Namen wie zum Beispiel `kinesis-UDF`, damit Sie sich diesen leicht merken können, und aktualisieren Sie Eingabeparameter wie den Eingabe-Stream, falls Sie einen anderen Namen wünschen. Wählen Sie Weiter aus.
5. Fügen Sie auf der Seite „Stack-Optionen konfigurieren“ bei Bedarf Tags hinzu und wählen Sie dann Weiter.
6. Markieren Sie auf der Seite Überprüfen die Kästchen, die die Erstellung von IAM-Ressourcen ermöglichen, und wählen Sie dann Absenden aus.

Der Start des CloudFormation Stacks kann je nach Region, in der Sie starten, 10 bis 15 Minuten dauern. Sobald Sie den `CREATE_COMPLETE`-Status für den gesamten Stack sehen, können Sie fortfahren.

Arbeiten mit einem Managed Service für Apache Flink Studio Notebook

Studio-Notebooks für Kinesis Data Analytics ermöglichen Ihnen die interaktive Abfrage von Datenströmen in Echtzeit und die einfache Erstellung und Ausführung von Stream-Verarbeitungsanwendungen mit Standard-SQL, Python und Scala. Mit ein paar Klicks können Sie ein serverloses Notebook starten AWS-Managementkonsole, um Datenströme abzufragen und innerhalb von Sekunden Ergebnisse zu erhalten.

Ein Notebook ist eine webbasierte Entwicklungsumgebung. Notebooks bieten ein einfaches interaktives Entwicklungserlebnis in Kombination mit den fortschrittlichen Datenstromverarbeitungsfunktionen von Apache Flink. Studio-Notebooks verwenden Notebooks, die mit Apache Zeppelin betrieben werden, und verwenden Apache Flink als Stream-Verarbeitungs-Engine. Studio-Notebooks kombinieren diese Technologien nahtlos, um Entwicklern aller Qualifikationsstufen erweiterte Analysen von Datenströmen zugänglich zu machen.

Apache Zeppelin bietet für Ihre Studio-Notebooks eine komplette Suite von Analysetools, darunter die folgenden:

- Datenvisualisierung
- Exportieren der Daten in Dateien
- Kontrolle über das Ausgabeformat zur Erleichterung von Analysen

Verwendung des Notebooks

1. Gehen Sie zu AWS-Managementkonsole und wählen Sie Amazon Kinesis unter der Liste der Dienste aus.
2. Wählen Sie auf der linken Navigationsseite Analytics-Anwendungen und dann Studio-Notebooks aus.
3. Stellen Sie sicher, dass das KinesisDataAnalyticsStudioNotebook läuft.
4. Wählen Sie das Notizbuch und dann In Apache Zeppelin öffnen aus.
5. Laden Sie die Datei Datenproduzent Zeppelin-Notebook herunter, mit der Sie Daten lesen und in den Kinesis Stream laden werden.

6. Importieren Sie das Zeppelin-Notebook namens `Data Producer`. Achten Sie darauf, die Eingabe-`STREAM_NAME` und -`REGION` im Code des Notebooks zu ändern. Der Name des Eingabestreams ist in der CloudFormation -Stack-Ausgabe zu finden.
7. Führen Sie das Datenproduzenten-Notebook aus, indem Sie auf die Schaltfläche Diesen Absatz ausführen klicken, um Beispieldaten in die Eingabe des Kinesis Data Streams einzufügen.
8. Laden Sie beim Laden der Beispieldaten MaskPhoneNumber-Interactive notebook herunter. Dieses Programm liest Eingabedaten, anonymisiert Telefonnummern aus dem Eingabestream und speichert anonymisierte Daten im Ausgabestrom.
9. Importieren Sie das `MaskPhoneNumber-interactive`-Zeppelin-Notizbuch.
10. Führen Sie jeden Absatz im Notebook aus.

    a. In Absatz 1 importieren Sie eine benutzerdefinierte Funktion zur Anonymisierung von Telefonnummern.

    ```
    %flink(parallelism=1)
    import com.mycompany.app.MaskPhoneNumber
    stenv.registerFunction("MaskPhoneNumber", new MaskPhoneNumber())
    ```

    b. Im nächsten Absatz erstellen Sie eine speicherinterne Tabelle zum Lesen von Eingabestreamdaten. Stellen Sie sicher, dass der Streamname und die AWS Region korrekt sind.

    ```
    %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. Überprüfen Sie, ob Daten in die speicherinterne Tabelle geladen werden.

```
%flink.ssql(type=update)
select * from customer_reviews
```

d. Rufen Sie die benutzerdefinierte Funktion auf, um die Telefonnummer zu anonymisieren.

```
%flink.ssql(type=update)
select customer_id, product, review, MaskPhoneNumber('mask_phone', phone) as
 phoneNumber from customer_reviews
```

e. Nachdem die Telefonnummern maskiert sind, erstellen Sie eine Ansicht mit einer maskierten Nummer.

```
%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. Überprüfen Sie die Daten.

```
%flink.ssql(type=update)
select * from sentiments_view
```

g. Erstellen Sie eine speicherinterne Tabelle für die Kinesis-Stream-Ausgabe. Stellen Sie sicher, dass Streamname und AWS Region korrekt sind.

```
%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. Fügen Sie aktualisierte Datensätze in den Ziel-Kinesis Stream ein.

```
%flink.ssql(type=update)
INSERT INTO customer_reviews_stream_table
SELECT customer_id, product, review, phoneNumber
FROM sentiments_view
```

i. Sichten und überprüfen Sie Daten aus dem Ziel-Kinesis Stream.

```
%flink.ssql(type=update)
select * from customer_reviews_stream_table
```

Werbung für ein Notebook als Anwendung

Nachdem Sie jetzt Ihren Notebookcode interaktiv getestet haben, stellen Sie ihn als Streaming-Anwendung mit dauerhaftem Zustand bereit. Sie müssen zuerst die Anwendungskonfiguration ändern, um einen Speicherort für Ihren Code in Amazon-S3 anzugeben.

1. Wählen Sie auf dem AWS-Managementkonsole Ihr Notebook aus und wählen Sie unter Als Anwendungskonfiguration bereitstellen — optional die Option Bearbeiten aus.
2. Wählen Sie unter Ziel für Code in Amazon-S3 den Amazon-S3-Bucket aus, der durch die CloudFormation -Skripte erstellt wurde. Der Vorgang kann einige Minuten dauern.
3. Sie können die Notiz in ihrer vorliegenden Form nicht bewerben. Wenn Sie dies versuchen, erhalten Sie eine Fehlermeldung, da `Select`-Anweisungen nicht unterstützt werden. Um dieses Problem zu vermeiden, laden Sie das MaskPhoneNumber-Streaming Zeppelin Notebook herunter.
4. Importieren Sie das `MaskPhoneNumber-streaming`-Zeppelin-Notizbuch.
5. Öffnen Sie die Notiz und wählen Sie Aktionen für. KinesisDataAnalyticsStudio
6. Wähle Build MaskPhoneNumber -Streaming und exportiere nach S3. Achten Sie darauf, den Anwendungsnamen umzubenennen und keine Sonderzeichen zu verwenden.

7. Wählen Sie Erstellen und Exportieren. Die Einrichtung der Streaming-Anwendung dauert einige Minuten.
8. Sobald der Build abgeschlossen ist, wählen Sie Bereitstellen mit der AWS -Konsole.
9. Überprüfen Sie auf der nächsten Seite die Einstellungen und stellen Sie sicher, dass Sie die richtige IAM-Rolle auswählen. Wählen Sie als Nächstes Streaming-Anwendung erstellen.
10. Nach einigen Minuten wird die Meldung angezeigt, dass die Streaming-Anwendung erfolgreich erstellt wurde.

Weitere Informationen zur Bereitstellung von Anwendungen mit dauerhaftem Zustand und Grenzwerten finden Sie unter Bereitstellen als Anwendung mit dauerhaftem Zustand.

Bereinigen

Optional können Sie jetzt den CloudFormation -Stack deinstallieren. Dadurch werden alle Dienste entfernt, die Sie zuvor eingerichtet haben.

Was ist Amazon-Kinesis-Data-Analytics for SQL-Anwendungen?

Mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen können Sie Streaming-Daten mit Standard-SQL verarbeiten und analysieren. Der Service ermöglicht die schnelle Erstellung und Ausführung von leistungsstarkem SQL-Code für Streaming-Quellen zum Durchführen von Zeitreihenanalysen, Füllen von Echtzeit-Dashboards und Erstellen von Echtzeitmetriken.

Zur Vorbereitung der Nutzung von Kinesis Data Analytics erstellen Sie eine Kinesis-Data-Analytics-Anwendung, die kontinuierlich Streaming-Daten liest und verarbeitet. Der Service unterstützt die Aufnahme von Daten aus Amazon Kinesis Data Streams- und Amazon Data Firehose-Streaming-Quellen. Anschließend erstellen Sie mit dem interaktiven Editor Ihren SQL-Code und testen diesen mit Streaming-Livedaten. Sie können auch Ziele konfigurieren, an die Kinesis Data Analytics die Ergebnisse senden soll.

Kinesis Data Analytics unterstützt Amazon Data Firehose (Amazon S3, Amazon Redshift, Amazon OpenSearch Service und Splunk) und Amazon Kinesis Data AWS Lambda Streams als Ziele.

Wann sollte ich Amazon-Kinesis-Data-Analytics verwenden?

Amazon-Kinesis-Data-Analytics ermöglicht die schnelle Erstellung von SQL-Code, der kontinuierlich Daten nahezu in Echtzeit liest, verarbeitet und speichert. Mithilfe von Standard-SQL-Abfragen für Streaming-Daten können Sie Anwendungen erstellen, mit denen sich Einblicke in Ihre Daten gewinnen lassen. Im Folgenden finden Sie einige Beispielszenarien für die Verwendung von Kinesis Data Analytics:

- Generieren von Zeitreihenanalysen – Sie können Metriken über Zeitfenster berechnen und dann über einen Kinesis Daten-Bereitstellungs-Stream Werte an Amazon-S3 oder Amazon-Redshift streamen.
- Füllen von Echtzeit-Dashboards – Sie können aggregierte und verarbeitete Ergebnisse von Streaming-Daten downstream senden und verarbeiten, um Echtzeit-Dashboards zu füllen.
- Erstellen von Echtzeitmetriken – Sie können benutzerdefinierte Metriken und Auslöser für den Einsatz in der Echtzeit-Überwachung, in Benachrichtigungen und Alarmen erstellen.

Weitere Informationen zu den SQL-Sprachelementen, die von Kinesis Data Analytics unterstützt werden, finden Sie in der SQL-Referenz zu Amazon-Kinesis-Data-Analytics.

Verwenden Sie Amazon-Kinesis-Data-Analytics zum ersten Mal?

Wenn Sie Amazon-Kinesis-Data-Analytics zum ersten Mal verwenden, empfehlen wir Ihnen, nacheinander die folgenden Abschnitte zu lesen:

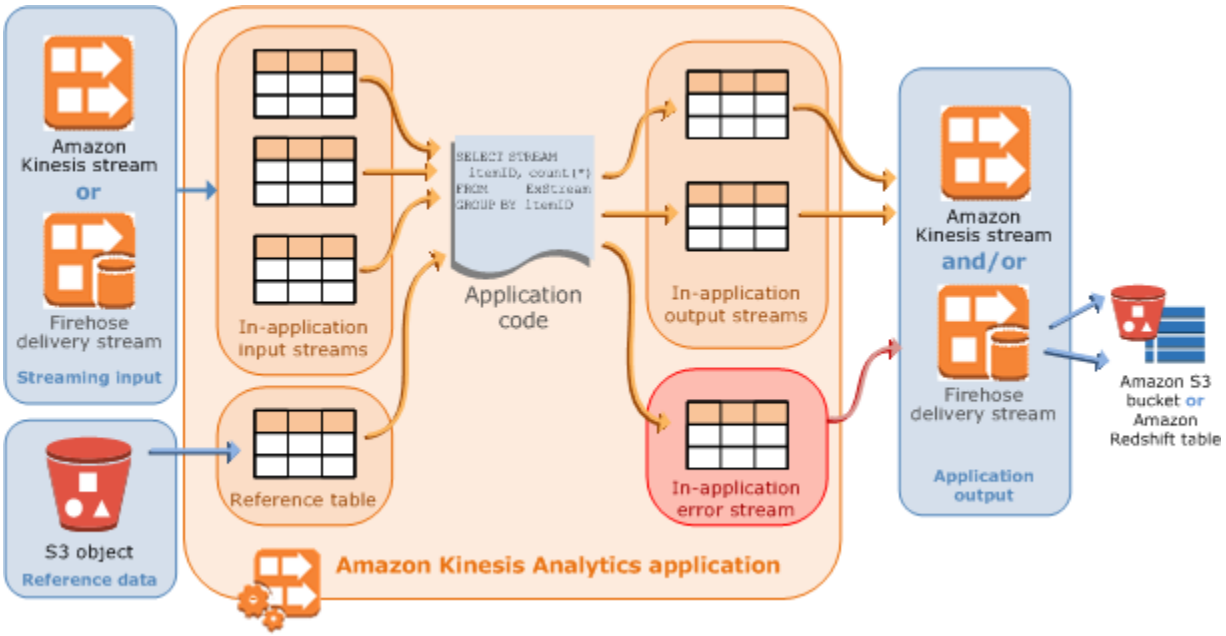
1. Lesen Sie den Abschnitt zur Funktionsweise in diesem Dokument. In diesem Abschnitt werden verschiedene Kinesis Data Analytics Analytics-Komponenten vorgestellt, mit denen Sie ein end-to-end Erlebnis erstellen können. Weitere Informationen finden Sie unter Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's.
2. Absolvieren Sie die Einstiegsübungen. Weitere Informationen finden Sie unter Erste Schritte mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen.
3. Machen Sie sich mit den SQL-Streaming-Konzepten vertraut. Weitere Informationen finden Sie unter SQL-Streaming-Konzepte.
4. Versuchen Sie es mit weiteren Beispielen. Weitere Informationen finden Sie unter Kinesis Data Analytics for SQL.

Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

Eine Anwendung ist die primäre Ressource in Amazon-Kinesis-Data-Analytics, die Sie in Ihrem Konto erstellen können. Sie können Anwendungen mithilfe der AWS-Managementkonsole oder der Kinesis Data Analytics Analytics-API erstellen und verwalten. Kinesis Data Analytics bietet API-Operationen zur Verwaltung von Anwendungen. Eine Liste der verfügbaren API-Operationen finden Sie unter Aktionen.

Kinesis Data Analytics-Anwendungen lesen und verarbeiten kontinuierlich Streaming-Daten in Echtzeit. Sie schreiben Anwendungscode mithilfe von SQL, um die eingehenden Streaming-Daten zu verarbeiten und eine Ausgabe zu erzeugen. Anschließend schreibt Kinesis Data Analytics die Ausgabe an ein konfiguriertes Ziel. Das folgende Diagramm zeigt eine typische Anwendungsarchitektur.



Jede Anwendung hat einen Namen, eine Beschreibung, eine Versions-ID und einen Status. Amazon-Kinesis-Data-Analytics weist eine Versions-ID zu, sobald Sie eine Anwendung erstellen. Diese Versionskennung wird aktualisiert, wenn Sie Teile der Anwendungskonfiguration aktualisieren. Wenn Sie beispielsweise eine Eingabekonfiguration hinzufügen oder löschen, eine Referenzdatenquelle hinzufügen, eine Ausgabekonfiguration hinzufügen oder löschen oder den Anwendungscode aktualisieren, aktualisiert Kinesis Data Analytics die aktuelle Anwendungsversions-ID. Außerdem verwaltet Kinesis Data Analytics die Zeitstempel der Anwendungserstellung und des letzten Änderungsdatums.

Zusätzlich zu diesen grundlegenden Eigenschaften enthalten Anwendungen Folgendes:

- Eingabedaten: – die Streaming-Quelle für Ihre Anwendung. Sie können entweder einen Kinesis-Datenstream oder einen Firehose-Datenbereitstellungsstream als Streaming-Quelle auswählen. In der Eingabekonfiguration ordnen Sie die Streaming-Quelle einem In-Application-Eingabe-Stream zu. Der In-Application-Stream entspricht einer kontinuierlich aktualisierten Tabelle, auf die Sie die Operationen `SELECT` und `INSERT` SQL anwenden können. In Ihrem Anwendungscode können Sie zusätzliche In-Application-Streams erstellen, um Zwischenergebnisse aus Abfragen zu speichern.

  Sie können optional eine einzelne Streaming-Quelle in mehrere In-Application-Eingabe-Streams aufteilen, um den Durchsatz zu verbessern. Weitere Informationen erhalten Sie unter Einschränkungen und Konfigurieren der Anwendungseingabe.

  Amazon-Kinesis-Data-Analytics stellt für jeden In-Application-Stream eine Zeitstempelspalte mit dem Namen Zeitstempel und die ROWTIME-Spalte bereit. Sie verwenden diese Spalte in Abfragen mit Zeitfenstern. Weitere Informationen finden Sie unter Abfragen mit Fenstern.

  Sie können optional eine Referenzdatenquelle zur Erweiterung Ihres Eingabedaten-Streams innerhalb der Anwendung konfigurieren. Sie erhalten dann eine In-Application-Referenztabelle. Sie müssen Ihre Referenzdaten als Objekt in Ihrem S3-Bucket speichern. Wenn die Anwendung gestartet wird, liest Amazon-Kinesis-Data-Analytics das Amazon-S3-Objekt und erstellt eine In-Application-Tabelle. Weitere Informationen finden Sie unter Konfigurieren der Anwendungseingabe.

- Anwendungscode – Eine Reihe von SQL-Anweisungen, die Eingabedaten verarbeiten und Ausgabedaten erzeugen. Sie können SQL-Anweisungen für In-Application-Streams und Referenztabellen schreiben. Sie können auch JOIN-Abfragen erstellen, um Daten aus diesen beiden Quellen zu kombinieren.

  Weitere Informationen zu den SQL-Sprachelementen, die von Kinesis Data Analytics unterstützt werden, finden Sie in der SQL-Referenz zu Amazon-Kinesis-Data-Analytics.

  In seiner einfachsten Form kann der Anwendungscode aus einer einzelnen SQL-Anweisung bestehen, die aus Streaming-Eingabedaten auswählt und die Ergebnisse in Streaming-Ausgabedaten einfügt. Es kann sich auch um eine Reihe von SQL-Anweisungen handeln, wobei die Ausgabedaten einer SQL-Anweisung als Eingabedaten für die nächste SQL-Anweisung verwendet werden können. Außerdem können Sie Anwendungscode erstellen, um einen Eingabe-Stream in mehrere Streams aufzuteilen. Sie können dann zusätzliche Abfragen anwenden, um diese Streams zu verarbeiten. Weitere Informationen finden Sie unter Anwendungscode.

- Output – Im Anwendungscode werden Abfrageergebnisse in In-Application-Streams eingespeist. Sie können in Ihrem Anwendungscode weitere In-Application-Streams erstellen, um Zwischenergebnisse zu speichern. Anschließend können Sie optional die Anwendungsausgabe so konfigurieren, dass Daten jener In-Application-Streams, die die Anwendungsausgabe enthalten (diese Streams werden auch als In-Application-Ausgabe-Streams bezeichnet), in externen Zielen dauerhaft gespeichert werden. Externe Ziele können ein Firehose-Lieferstream oder ein Kinesis-Datenstrom sein. Beachten Sie die folgenden Hinweise zu diesen Zielen:
  - Sie können einen Firehose-Lieferstream so konfigurieren, dass Ergebnisse in Amazon S3, Amazon Redshift oder Amazon OpenSearch Service (OpenSearch Service) geschrieben werden.

  - Sie können die Anwendungsausgabe auch in ein benutzerdefiniertes Ziel schreiben (anstatt in Amazon-S3 oder Amazon-Redshift). Legen Sie dazu in Ihrer Ausgabekonfiguration einen Kinesis-Datenstrom als Ziel fest. Anschließend konfigurieren Sie so, AWS Lambda dass der Stream abgefragt und Ihre Lambda-Funktion aufgerufen wird. Ihr Lambda-Funktionscode erhält Stream-Daten als Eingabe. Sie können die eingehenden Daten in Ihrem Lambda-Funktionscode

an das gewünschte benutzerdefinierte Ziel schreiben. Weitere Informationen finden Sie unter Verwenden AWS Lambda mit Amazon Kinesis Data Analytics.

Weitere Informationen finden Sie unter Konfigurieren der Anwendungsausgabe.

Beachten Sie außerdem Folgendes:

- Amazon-Kinesis-Data-Analytics benötigt Berechtigungen zum Lesen von Datensätzen aus einer Streaming-Quelle und zum Schreiben der Ausgabe von Anwendungen an externe Ziele. Sie verwenden IAM-Rollen, um diese Berechtigungen zu erteilen.

- Kinesis Data Analytics stellt automatisch für jede Anwendung einen In-Application-Fehler-Stream bereit. Wenn Ihre Anwendung Probleme bei der Verarbeitung bestimmter Datensätze hat (z. B. wegen eines Typenkonflikts oder später Verfügbarkeit) wird dieser Datensatz in den Fehler-Stream geschrieben. Sie können die Ausgabe von Anwendungen so konfigurieren, dass Kinesis Data Analytics angewiesen wird, die Daten aus dem Fehler-Stream zur weiteren Auswertung an einem externen Ziel dauerhaft zu speichern. Weitere Informationen finden Sie unter Fehlerbehandlung.

- Amazon-Kinesis-Data-Analytics stellt sicher, dass die Ausgabedatensätze Ihrer Anwendung in das konfigurierte Ziel geschrieben werden. Es wird ein „Mindestens einmal“-Verarbeitungs- und Bereitstellungsmodell verwendet – auch dann, wenn eine Anwendung unterbrochen wird. Weitere Informationen finden Sie unter Bereitstellungsmodell für die Weiterleitung der Anwendungsausgabe an ein externes Ziel.

Themen

Konfigurieren der Anwendungseingabe

Ihre Amazon-Kinesis-Data-Analytics-Anwendung kann Eingaben aus einer einzelnen Streaming-Quelle erhalten und optional eine einzelne Referenzdatenquelle verwenden. Weitere Informationen finden Sie unter Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's. Die Abschnitte in diesem Thema beschreiben die Quellen von Anwendungseingaben.

Themen

Konfigurieren einer Streaming-Quelle

Wenn Sie eine Anwendung erstellen, geben Sie eine Streaming-Quelle an. Sie können auch eine Eingabe ändern, nachdem Sie die Anwendung erstellt haben. Amazon-Kinesis-Data-Analytics unterstützt die folgenden Streaming-Quellen für Ihre Anwendung:

- Einen Kinesis-Datenstrom
- Ein Firehose-Lieferstream

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Bestandskunden, die Kinesis-Data-Analytics-for-SQL-Anwendungen mit `KinesisFirehoseInput` verwenden, können weiterhin Kinesis Data Analytics einsetzen, um Anwendungen mit `KinesisFirehoseInput` innerhalb eines bestehenden Kontos hinzuzufügen. Wenn Sie bereits Kunde sind und mit Kinesis-Data-Analytics-for-SQL-Anwendungen ein neues Konto mit `KinesisFirehoseInput` erstellen möchten, können

Sie über das Formular zur Erhöhung des Service-Limits einen Fall erstellen. Weitere Informationen erhalten Sie im AWS Support -Sicherheitszentrum. Wir empfehlen, neue Anwendungen immer zu testen, bevor Sie sie in die Produktionsumgebung überführen.

> **Note**
>
> Wenn der Kinesis-Datenstrom verschlüsselt ist, greift Kinesis Data Analytics nahtlos auf die Daten im verschlüsselten Stream zu. Weitere Konfigurationsschritte sind nicht erforderlich. Kinesis Data Analytics speichert keine unverschlüsselten Daten, die aus Kinesis-Datenströmen gelesen wurden. Weitere Informationen finden Sie unter Was bedeutet eine serverseitige Verschlüsselung in Kinesis-Daten-Streams?

Kinesis Data Analytics fragt die Streaming-Quelle kontinuierlich nach neuen Daten ab und übernimmt diese in In-Application-Streams, je nach Eingabekonfiguration.

> **Note**
>
> Das Hinzufügen eines Kinesis-Streams als Eingabe Ihrer Anwendung wirkt sich nicht auf die Daten im Stream aus. Wenn eine andere Ressource, z. B. ein Firehose-Lieferstream, ebenfalls auf denselben Kinesis-Stream zugreifen würde, würden sowohl Firehose Firehose-Lieferstream als auch die Kinesis Data Analytics-Anwendung dieselben Daten erhalten. Der Durchsatz und die Drosselung könnten jedoch beeinträchtigt sein.

Ihr Anwendungscode kann den In-Application-Stream abfragen. Im Rahmen der Eingabekonfiguration stellen Sie Folgendes bereit:

- Streaming-Quelle – Sie stellen den Amazon-Ressourcennamen (ARN) des Streams und eine IAM-Rolle bereit, die Kinesis Data Analytics annehmen kann, um in Ihrem Namen auf den Stream zuzugreifen.
- Namenspräfix des In-Application-Streams – Wenn Sie die Anwendung starten, erstellt Kinesis Data Analytics den angegebenen In-Application-Stream. In Ihrem Anwendungscode greifen Sie mittels dieses Namens auf den In-Application-Stream zu.

  Sie können eine Streaming-Quelle optional mehreren In-Application-Streams zuordnen. Weitere Informationen finden Sie unter Einschränkungen. In diesem Fall erstellt Amazon Kinesis Data

Analytics die angegebene Anzahl von In-Application-Streams mit folgenden Namen: *prefix* _001 *prefix*_002, und. *prefix* _003 Standardmäßig ordnet Kinesis Data Analytics die Streaming-Quelle einem anwendungsinternen Stream mit dem Namen zu. *prefix* _001

Es gibt eine Einschränkung in Bezug auf die Rate, mit der Sie Zeilen in einen In-Application-Stream einfügen können. Daher unterstützt Kinesis Data Analytics mehrere solcher In-Application-Streams, damit Sie wesentlich schneller Datensätze in Ihre Anwendung importieren können. Wenn Ihre Anwendung mit den Daten in der Streaming-Quelle nicht Schritt halten kann, können Sie Parallelitätseinheiten hinzufügen, um die Leistung zu verbessern.

- Zuweisungsschema – Sie beschreiben das Datensatzformat (JSON, CSV) für die Streaming-Quelle. Außerdem beschreiben Sie, wie jeder Datensatz im Stream den Spalten im erstellten In-Application-Stream zugeordnet wird. An dieser Stelle stellen Sie Spaltennamen und Datentypen bereit.

> **Note**
>
> Kinesis Data Analytics fügt den Kennungen (Stream-Name und Spaltennamen) bei der Erstellung des Eingabe-In-Application-Streams Anführungszeichen hinzu. Bei der Abfrage dieses Streams und der Spalten müssen Sie diese Bezeichner in Anführungszeichen mit übereinstimmender Groß- und Kleinschreibung (exakt übereinstimmende Groß- und Kleinbuchstaben) angeben. Weitere Informationen zu Kennungen finden Sie unter Kennungen in der Amazon-Managed-Service für Apache Flink-SQL-Referenz.

Sie können in der Amazon-Kinesis-Data-Analytics-Konsole eine Anwendung erstellen und Eingaben konfigurieren. Die Konsole führt anschließend die erforderlichen API-Aufrufe aus. Sie können Anwendungseingaben konfigurieren, wenn Sie eine neue Anwendungs-API erstellen oder einer vorhandenen Anwendung eine Eingabekonfiguration hinzufügen. Weitere Informationen erhalten Sie unter CreateApplication und AddApplicationInput. Im Folgenden sehen Sie den Eingabekonfigurationsteil des `Createapplication`-API-Anforderungstexts:

```
"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"
    }
]
```

Konfigurieren einer Referenzquelle

Optional können Sie einer vorhandenen Anwendung auch eine Referenzdatenquelle hinzufügen, um die Daten aus Streaming-Quellen anzureichern. Sie müssen Referenzdaten als Objekt in Ihrem Amazon-S3-Bucket speichern. Wenn die Anwendung gestartet wird, liest Amazon-Kinesis-Data-Analytics das Amazon-S3-Objekt und erstellt eine In-Application-Referenztabelle. Ihr Anwendungscode kann die Tabelle dann mit einem In-Application-Stream verknüpfen.

Sie speichern Referenzdaten im Amazon-S3-Objekt unter Verwendung der unterstützten Formate (CSV, JSON). Angenommen, Ihre Anwendung führt Analysen von Börsenbestellungen aus. Nehmen Sie das folgende Datensatzformat für die Streaming-Quelle an:

```
Ticker, SalePrice, OrderId
```

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

In diesem Fall könnten Sie eine Referenzdatenquelle in Betracht ziehen, um Details zu den einzelnen Börsentickern bereitzustellen, wie den Firmennamen.

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

Sie können eine Referenzdatenquelle der Anwendung entweder mit der API oder mit der Konsole hinzufügen. Amazon-Kinesis-Data-Analytics stellt die folgenden API-Aktionen zur Verwaltung von Referenz-Datenquellen bereit:

- AddApplicationReferenceDataSource
- UpdateApplication

Informationen zum Hinzufügen von Referenzdaten mithilfe der Konsole finden Sie unter Beispiel: Hinzufügen von Referenzdaten zu einer Kinesis Data Analytics-Anwendung.

Beachten Sie Folgendes:

- Wenn die Anwendung ausgeführt wird, erstellt Kinesis Data Analytics eine In-Application-Referenztabelle und lädt anschließend sofort die Referenzdaten.
- Wenn die Anwendung nicht ausgeführt wird (wenn sie z. B. im Bereitschaftsstatus ist), speichert Kinesis Data Analytics nur die aktualisierte Eingabekonfiguration. Wenn die Ausführung der Anwendung gestartet wird, lädt Kinesis Data Analytics die Referenzdaten als Tabelle in Ihrer Anwendung.

Gehen wir davon aus, dass Sie die Daten aktualisieren möchten, nachdem Kinesis Data Analytics die In-Application-Referenztabelle erstellt hat. Vielleicht haben Sie das Amazon-S3-Objekt aktualisiert oder Sie möchten ein anderes Amazon-S3-Objekt verwenden. In diesem Fall können Sie entweder UpdateApplication explizit aufrufen oder in der Konsole Aktionen, Referenzdatentabelle synchronisieren wählen. Kinesis Data Analytics aktualisiert die In-Application-Referenztabelle nicht automatisch.

Es gibt eine Einschränkung in Bezug auf die Größe des Amazon-S3-Objekts, das Sie als Referenzdatenquelle erstellen können. Weitere Informationen finden Sie unter Einschränkungen. Wenn die Größe des Objekts den Grenzwert überschreitet, kann Kinesis Data Analytics die Daten nicht laden. Der Anwendungsstatus wird als „ausgeführt“ angezeigt, die Daten werden jedoch nicht gelesen.

Wenn Sie eine Referenzdatenquelle hinzufügen, müssen Sie folgende Informationen bereitstellen:

- Name von S3-Bucket und Objektschlüssel – Zusätzlich zum Bucket-Namen und Objektschlüssel stellen Sie auch eine IAM-Rolle bereit, die Kinesis Data Analytics annehmen kann, um das Objekt in Ihrem Namen zu lesen.
- Name der In-Application-Referenztabelle – Kinesis Data Analytics erstellt diese In-Application-Tabelle und füllt sie durch Lesen des Amazon-S3-Objekts aus. Dies ist der Tabellenname, den Sie in Ihrem Anwendungscode angeben.
- Zuweisungsschema – Sie beschreiben das Datensatzformat (JSON, CSV) und die Kodierung der im Amazon-S3-Objekt gespeicherten Daten. Sie beschreiben außerdem die Zuordnung der einzelnen Datenelemente zu den Spalten in der In-Application-Referenztabelle.

Im Folgenden wird der Anforderungstext in der `AddApplicationReferenceDataSource`-API-Anforderung gezeigt.

```
{
    "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"
    }
}
```

Arbeiten mit JSONPath

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

JSONPath ist eine standardisierte Methode, um Elemente eines JSON-Objekts abzufragen. JSONPath verwendet Pfadausdrücke, um durch Elemente, verschachtelte Elemente und Arrays in einem JSON-Dokument zu navigieren. Weitere Informationen über JSON finden Sie auf der Website Introducing JSON.

Amazon Kinesis Data Analytics verwendet JSONPath Ausdrücke im Quellschema der Anwendung, um Datenelemente in einer Streaming-Quelle zu identifizieren, die Daten im JSON-Format enthält.

Weitere Informationen dazu, wie Sie Streaming-Daten dem Eingabestream Ihrer Anwendung zuordnen, finden Sie unter the section called “Zuweisung von Streaming-Quellenelementen zu SQL-Eingabespalten”.

Zugreifen auf JSON-Elemente mit JSONPath

Im Folgenden erfahren Sie, wie Sie mithilfe von JSONPath Ausdrücken auf verschiedene Elemente in JSON-formatierten Daten zugreifen können. Die Beispiele in diesem Abschnitt gehen davon aus, dass der Quell-Stream den folgenden JSON-Datensatz enthält:

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

Zugriff auf JSON-Elemente

Verwenden Sie die folgende Syntax, um ein Element in JSON-Daten mit JSONPath abzufragen. Hier stellt $ das Stammverzeichnis der Datenhierarchie dar und `elementName` ist der Name des Elementknoten, der abgefragt werden soll.

```
$.elementName
```

Der folgende Ausdruck fragt das `customerName`-Element aus dem vorherigen JSON-Beispiel ab.

```
$.customerName
```

Der vorherige Ausdruck gibt Folgendes aus dem vorherigen JSON-Datensatz zurück.

```
John Doe
```

> **Note**
>
> Pfadausdrücke beachten die Groß- und Kleinschreibung. Der Ausdruck `$.customername` gibt `null` aus dem vorherigen JSON-Beispiel zurück.

> **Note**
>
> Wenn an der Stelle, die der Pfadausdruck angibt, kein Element angezeigt wird, gibt der Ausdruck `null` zurück. Der folgende Ausdruck gibt `null` aus dem vorherigen Beispiel zurück, weil es kein übereinstimmendes JSON-Element gibt.
>
> ```
> $.customerId
> ```

Zugriff auf verschachtelte JSON-Elemente

Um ein verschachteltes JSON-Element abzufragen, verwenden Sie die folgende Syntax.

```
$.parentElement.element
```

Der folgende Ausdruck fragt das `city`-Element aus dem vorherigen JSON-Beispiel ab.

```
$.address.city
```

Der vorherige Ausdruck gibt Folgendes aus dem vorherigen JSON-Datensatz zurück.

```
Anytown
```

Sie können mithilfe der folgenden Syntax weitere Ebenen mit Unterelementen abfragen.

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

Der folgende Ausdruck fragt das `street`-Element aus dem vorherigen JSON-Beispiel ab.

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

Der vorherige Ausdruck gibt Folgendes aus dem vorherigen JSON-Datensatz zurück.

```
AnyStreet
```

Zugriff auf Arrays

Sie können auf die Daten in einem JSON-Array wie folgt zugreifen:

- Abrufen aller Elemente im Array als einzelne Zeile.
- Abrufen eines jeden Elements im Array als eigene Zeile.

Abrufen aller Elemente in einem Array in einer einzelnen Zeile

Um den gesamten Inhalt eines Arrays als einzelne Zeile abzufragen, verwenden Sie die folgende Syntax.

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

Der folgende Ausdruck fragt den gesamten Inhalt des `orders`-Elements im vorherigen JSON-Beispiel ab, das in diesem Abschnitt verwendet wird. Es gibt den Array-Inhalt in einer einzelnen Spalte in einer einzigen Zeile zurück.

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

Der vorherige Ausdruck gibt Folgendes aus dem JSON-Beispiel-Datensatz zurück, der in diesem Abschnitt verwendet wird.

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

Abrufen aller Elemente in einem Array in separaten Zeilen

Um die einzelnen Elemente in einem Array als eigene Zeilen abzufragen, verwenden Sie die folgende Syntax.

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

Der folgende Ausdruck fragt die `orderId`-Elemente im vorhergehenden JSON-Beispiel ab und gibt jedes Arrayelement als eigene Zeile zurück.

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

Der vorherige Ausdruck gibt Folgendes aus dem vorherigen JSON-Datensatz zurück, wobei jedes Datenelement als eigene Zeile zurückgegeben wird.

23284
63122
77284

Note

Wenn Ausdrücke, die andere Elemente als Array-Elemente abfragen, in einem Schema enthalten sind, das einzelne Array-Elemente abfragt, werden die Elemente, die keine Array-Elemente sind, für jedes Element im Array wiederholt. Angenommen, ein Schema für das vorherige JSON-Beispiel enthält die folgenden Ausdrücke:

- $.customerName
- $.orders[0:].orderId

In diesem Fall sehen die zurückgegebenen Daten aus dem Beispiel-Eingabe-Stream-Element wie folgt aus, wobei das `name`-Element für jedes `orderId`-Element wiederholt wird.

Hans Muster	23284
Hans Muster	63122
Hans Muster	77284

Note

Für Array-Ausdrücke in Amazon-Kinesis-Data-Analytics gelten folgende Einschränkungen:

- In einem Array-Ausdruck wird nur eine Dereferenzierungsebene unterstützt. Das folgende Ausdrucksformat wird nicht unterstützt.

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

- In einem Schema kann nur ein Array auf eine Ebene gebracht werden. Es können mehrere Arrays referenziert werden – sie werden als einzelne Zeile zurückgegeben, die alle Elemente im Array enthält. Es können jedoch nur für ein Array alle Elemente als einzelne Zeilen zurückgegeben werden.

  Ein Schema, das Elemente im folgenden Format enthält, ist gültig. Dieses Format gibt den Inhalt des zweiten Arrays als einzelne Spalte wider, was für jedes Element im ersten Array wiederholt wird.

```
$.arrayObjectOne[0:].element
$.arrayObjectTwo[0:]
```

  Ein Schema, das Elemente im folgenden Format enthält, ist nicht gültig.

```
$.arrayObjectOne[0:].element
$.arrayObjectTwo[0:].element
```

Weitere Überlegungen

Bei der Arbeit mit sollten JSONPath Sie zusätzlich Folgendes beachten:

- Wenn ein einzelnes Element in den JSONPath Ausdrücken im Anwendungsschema auf keine Arrays zugreift, wird für jeden verarbeiteten JSON-Datensatz eine einzelne Zeile im Eingabestream der Anwendung erstellt.
- Wenn ein Array auf eine Ebene gebracht wird (d. h. seine Elemente werden als einzelne Zeilen zurückgegeben), führen fehlende Elemente dazu, dass im In-Application-Stream ein Nullwert erstellt wird.

- Ein Array wird stets in Form mindestens einer Zeile auf eine Ebene gebracht. Wenn keine Werte zurückgegeben würden (d. h., wenn das Array leer ist oder keines seiner Elemente abgefragt wird), wird eine einzelne Zeile mit ausschließlich Nullwerten zurückgegeben.

  Der folgende Ausdruck gibt aus dem vorherigen JSON-Beispiel Datensätze mit Nullwerten zurück, da es im angegebenen Pfad kein übereinstimmendes Element gibt.

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

  Der vorherige Ausdruck gibt Folgendes aus dem vorherigen JSON-Beispieldatensatz zurück.

Null
Null
Null

Verwandte Themen

- Einführung von JSON

Zuweisung von Streaming-Quellenelementen zu SQL-Eingabespalten

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

Mit Amazon-Kinesis-Data-Analytics können Sie mithilfe von Standard-SQL Streaming-Daten im JSON- oder CSV-Format verarbeiten und analysieren.

- Um Streaming-CSV-Daten zu verarbeiten und zu analysieren, weisen Sie den Spalten im Eingabe-Stream Spaltennamen und Datentypen zu. Ihre Anwendung importiert pro Spaltendefinition der Reihe nach eine einzelne Spalte aus dem Eingabe-Stream.

Sie müssen nicht alle Spalten im Anwendungs-Eingabe-Stream einschließen. Sie können jedoch keine Spalten aus dem Quell-Stream überspringen. Sie können beispielsweise die ersten drei Spaltenaus einem Eingabe-Stream mit fünf Elemente importieren. Sie können jedoch nicht nur die Spalten 1, 2 und 4 importieren.

- Um Streaming-JSON-Daten zu verarbeiten und zu analysieren, verwenden Sie JSONPath Ausdrücke, um JSON-Elemente aus einer Streaming-Quelle SQL-Spalten in einem Eingabestream zuzuordnen. Weitere Informationen zur Verwendung JSONPath mit Amazon Kinesis Data Analytics finden Sie unterArbeiten mit JSONPath. Die Spalten in der SQL-Tabelle haben Datentypen, die von JSON-Typen zugeordnet wurden. Weitere Informationen zu unterstützten Datentypen finden Sie unter Datentypen. Weitere Informationen zum Umwandeln von JSON-Daten in SQL-Daten finden Sie unter Zuweisung von JSON-Datentypen zu SQL-Datentypen.

Weitere Informationen zum Konfigurieren von Eingabe-Streams finden Sie unter Konfigurieren der Anwendungseingabe.

Zuweisung von JSON-Daten zu SQL-Spalten

Sie können JSON-Elemente mithilfe der AWS-Managementkonsole oder der Kinesis Data Analytics Analytics-API den Eingabespalten zuordnen.

- Informationen zum Zuordnen von Elementen zu Spalten mithilfe der Konsole finden Sie unter Arbeiten mit dem Schema-Editor.
- Informationen zum Zuordnen von Elementen zu Spalten mithilfe der Kinesis Data Analytics-API finden Sie im folgenden Abschnitt.

Um JSON-Elemente zu Spalten im In-Application-Eingabe-Stream zuzuordnen, benötigen Sie ein Schema die folgenden Informationen für jede Spalte:

- Quellausdruck: Der JSONPath Ausdruck, der die Position der Daten für die Spalte identifiziert.
- Spaltenname: Der Name, den Ihre SQL-Abfragen verwenden, um die Daten zu referenzieren.
- Datentyp: Der SQL-Datentyp der Spalte.

Verwenden der API

Um Elemente aus einer Streaming-Quelle zu Eingabespalten zuzuordnen, können Sie die Kinesis Data Analytics-API-Aktion CreateApplication verwenden. Um den In-Application-Stream zu erstellen,

geben Sie ein Schema für die Umwandlung Ihrer Daten in eine schematisierte Version an, die in SQL verwendet wird. Die Aktion CreateApplication konfiguriert Ihre Anwendung für den Empfang von Eingaben aus einer einzelnen Streaming-Quelle. Um JSON-Elemente oder CSV-Spalten zu SQL-Spalten zuzuordnen, erstellen Sie ein RecordColumn-Objekt im SourceSchema `RecordColumns`-Array. Das RecordColumn-Objekt hat das folgende Schema:

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

Die Felder im RecordColumn-Objekt haben die folgenden Werte:

- `Mapping`: Der JSONPath Ausdruck, der die Position der Daten im Eingabestream-Datensatz identifiziert. Dieser Wert ist für ein Eingabeschema für einen Quell-Stream im CSV-Format nicht vorhanden.
- `Name`: der Spaltenname im In-Application-SQL-Daten-Stream.
- `SqlType`: der Datentyp der Daten im In-Application-SQL-Daten-Stream.

Beispiel für ein JSON-Eingabeschema

Das folgende Beispiel zeigt das Format des `InputSchema`-Werts für ein JSON-Schema.

```
"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"
}
```

Beispiel für ein CSV-Eingabeschema

Das folgende Beispiel zeigt das Format des `InputSchema`-Werts für ein Schema im CSV-Format (durch Komma getrennte Werte).

```
"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"
}
```

Zuweisung von JSON-Datentypen zu SQL-Datentypen

JSON-Datentypen werden entsprechend dem Eingabeschema der Anwendung in die entsprechenden SQL-Datentypen umgewandelt. Weitere Informationen zu unterstützten SQL-Datentypen finden Sie unter Datentypen. Amazon-Kinesis-Data-Analytics wandelt JSON-Datentypen entsprechend den folgenden Regeln in SQL-Datentypen um.

Null-Literal

Ein Null-Literal im JSON-Eingabe-Stream (d. h. `"City":null`) wird unabhängig vom Zieldatentyp in einen SQL-Nullwert umgewandelt.

Boolesches Literal

Ein boolesches Literal im JSON-Eingabe-Stream (d. h. `"Contacted":true`) wird wie folgt in SQL-Daten umgewandelt:

- Numerisch (DECIMAL, INT usw.): `true` wird in 1, `false` wird in 0 umgewandelt.
- Binär (BINARY oder VARBINARY):
  - `true`: Ergebnis hat den niedrigsten Bitsatz und die verbleibenden Bits sind gelöscht.
  - `false`: Alle Bits im Ergebnis sind gelöscht.

  Die Umwandlung in VARBINARY führt zum Längenwert 1 Byte.
- BOOLEAN: wird in den entsprechenden SQL BOOLEAN-Wert umgewandelt.
- Zeichen (CHAR oder VARCHAR): wird in den entsprechenden Zeichenfolgenwert (`true` oder `false`) umgewandelt. Der Wert wird abgeschnitten, um an die Feldlänge angepasst zu werden.
- Datum/Uhrzeit (DATE, TIME oder TIMESTAMP): Die Umwandlung schlägt fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.

Zahl

Ein numerisches Literal im JSON-Eingabe-Stream (d. h. `"CustomerId":67321`) wird wie folgt in SQL-Daten umgewandelt:

- Numerisch (DECIMAL, INT usw.): wird direkt umgewandelt. Wenn der umgewandelte Wert die Größe oder Genauigkeit des Zieldatentyps überschreitet (d. h. `123.4` wird in INT umgewandelt), schlägt die Umwandlung fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.
- Binär (BINARY oder VARBINARY): Die Umwandlung schlägt fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.
- BOOLESCH:
  - `0`: wird in `false` umgewandelt.
  - Alle anderen Zahlen: werden in `true` umgewandelt.
- Zeichen (CHAR oder VARCHAR): wird in eine Zeichenfolgendarstellung der Zahl umgewandelt.
- Datum/Uhrzeit (DATE, TIME oder TIMESTAMP): Die Umwandlung schlägt fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.

Zeichenfolge

Ein Zeichenfolgenwert im JSON-Eingabe-Stream (d. h. `"CustomerName":"John Doe"`) wird wie folgt in SQL-Daten umgewandelt:

- Numerisch (DECIMAL, INT usw.): Amazon-Kinesis-Data-Analytics versucht, den Wert in den Zieldatentyp umzuwandeln. Wenn der Wert nicht umgewandelt werden kann, schlägt die Umwandlung fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.
- Binär (BINARY oder VARBINARY): Wenn die Quellzeichenfolge ein gültiges binäres Literal ist (d. h. `X'3F67A23A'` mit einer geraden Anzahl von f), wird der Wert in den Zieldatentyp umgewandelt. Andernfalls schlägt die Umwandlung fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.
- BOOLEAN: Wenn die Quellzeichenfolge `"true"` ist, wird sie in `true` umgewandelt. Bei diesem Vergleich wird die Groß-/Kleinschreibung nicht beachtet. Andernfalls wird sie in `false` umgewandelt.
- Zeichen (CHAR oder VARCHAR): wird in den Zeichenfolgenwert in der Eingabe umgewandelt. Wenn der Wert länger als der Zieldatentyp ist, wird er abgeschnitten und es wird kein Fehler in den Fehler-Stream geschrieben.

- Datum/Uhrzeit (DATE, TIME oder TIMESTAMP): Wenn die Quellzeichenfolge in einem Format erstellt wurde, das in den Zielwert umgewandelt werden kann, wird der Wert umgewandelt. Andernfalls schlägt die Umwandlung fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.

  Gültige datetime-Formate sind:

  - „1992-02-14“
  - „1992-02-14 18:35:44.0“

Array oder Object

Ein Array oder Objekt im JSON-Eingabe-Stream wird wie folgt in SQL-Daten umgewandelt:

- Zeichen (CHAR oder VARCHAR): wandelt den Quelltext des Arrays oder Objekts um. Siehe Zugriff auf Arrays.
- Alle anderen Datentypen: Die Umwandlung schlägt fehl und es wird ein Erzwingungsfehler in den Fehler-Stream geschrieben.

Ein Beispiel für ein JSON-Array finden Sie unter Arbeiten mit JSONPath.

Verwandte Themen

- Konfigurieren der Anwendungseingabe
- Datentypen
- Arbeiten mit dem Schema-Editor
- CreateApplication
- RecordColumn
- SourceSchema

Verwenden der Funktion der Schemaerkennung für Streaming-Daten

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

Die Bereitstellung eines Eingabeschemas, das beschreibt, wie Datensätze im Streaming-Eingabestream einem anwendungsinternen Stream zugeordnet werden, kann umständlich und fehleranfällig sein. Sie können die DiscoverInputSchema-API (Erkennungs-API genannt), um ein Schema abzuleiten. Aus Zufallsstichproben von Datensätzen in der Streaming-Quelle kann die API ein Schema ableiten (d. h. Spaltennamen, Datentypen und Position des Datenelements in den eingehenden Daten).

> **Note**
>
> Informationen zum Verwenden der Discovery-API zum Erstellen eines Schemas aus einer in Amazon-S3 gespeicherten Datei finden Sie unter Verwenden der Funktion der Schemaerkennung für statische Daten.

Die Konsole verwendet die Discovery-API, um ein Schema für eine angegebene Streaming-Quelle zu generieren. Mithilfe der Konsole können Sie das Schema auch aktualisieren, z. B. Spalten hinzufügen oder entfernen, Spaltennamen oder Datentypen ändern usw. Gehen Sie bei den Änderungen jedoch sorgfältig vor, um sicherzustellen, dass Sie kein ungültiges Schema erstellen.

Nachdem Sie ein Schema für Ihren In-Application-Stream fertiggestellt haben, gibt es Funktionen, die Sie zur Bearbeitung von Zeichenfolgen- und Datums-/Uhrzeitwerten verwenden können. Sie können diese Funktionen in Ihrem Anwendungscode verwenden, wenn Sie im resultierenden In-Application-Stream mit Zeilen arbeiten. Weitere Informationen finden Sie unter Beispiel: Werte transformieren DateTime.

Benennung von Spalten während der Schemaerkennung

Während der Schemaerkennung versucht Amazon-Kinesis-Data-Analytics einen möglichst großen Teil des ursprünglichen Spaltennamens aus der Streaming-Eingabequelle beizubehalten, außer in folgenden Fällen:

- Der Spaltenname der Quelle ist ein reserviertes SQL-Schlüsselwort wie `TIMESTAMP`, `USER`, `VALUES` oder `YEAR`.
- Der Spaltenname des Quell-Streams enthält nicht unterstützte Zeichen. Es werden nur Buchstaben, Ziffern und Unterstriche (_) unterstützt.
- Die Spaltenname des Quell-Streams beginnt mit einer Zahl.
- Der Spaltenname des Quell-Streams enthält mehr als 100 Zeichen.

Wenn eine Spalte umbenannt wird, beginnt der umbenannte Spaltenname mit `COL_`. In einigen Fällen können keine der ursprünglichen Spaltennamen beibehalten werden, beispielsweise wenn der gesamte Name aus nicht unterstützten Zeichen besteht. In einem solchen Fall wird die Spalte mit `COL_#` benannt, wobei # eine Zahl ist, die den Platz der Spalte in der Spaltenreihenfolge angibt.

Nach dem Abschluss der Erkennung können Sie das Schema über die Konsole aktualisieren, um Spalten hinzuzufügen oder zu entfernen oder um Spaltennamen, Datentypen oder Datengröße zu ändern.

Beispiele für während der Erkennung vorgeschlagene Spaltennamen

Spaltenname im Quell-Stream	Während der Erkennung vorgeschlagener Spaltenname
USER	COL_USER
USER@DOMAIN	COL_USERDOMAIN
@@	COL_0

Probleme bei der Schemaerkennung

Was geschieht, wenn Kinesis Data Analytics kein Schema für eine bestimmte Streaming-Quelle ableitet?

Kinesis Data Analytics leitet Ihr Schema für gängige Formate wie CSV und JSON ab, die UTF-8 kodiert sind. Kinesis Data Analytics unterstützt alle mit UTF-8 kodierten Datensätze (einschließlich Rohtexte wie Anwendungsprotokolle und Datensätze) mit einem benutzerdefinierten Spalten- und Zeilentrennzeichen. Wenn Kinesis Data Analytics kein Schema ableitet, können Sie ein Schema manuell mittels des Schema-Editors in der Konsole (oder über die API) definieren.

Wenn Ihre Daten keinem Muster folgen (das Sie mittels des Schema-Editors angeben können), können Sie ein Schema als einzelne Spalte vom Typ VARCHAR(N) definieren, wobei N die größte Anzahl von Zeichen ist, die Ihr Datensatz voraussichtlich enthalten wird. Anschließend können Sie die Zeichenfolgen- und Datum-/Uhrzeitbearbeitung verwenden, um Ihre Daten zu strukturieren, wenn sie sich in einem In-Application-Stream befinden. Beispiele finden Sie unter Beispiel: Werte transformieren DateTime.

Verwenden der Funktion der Schemaerkennung für statische Daten

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

Mit der Funktion der Schemaerkennung können Sie ein Schema aus den Daten in einem Stream oder den Daten in einer statischen Datei erstellen, die in einem Amazon-S3-Bucket gespeichert wird. Gehen wir davon aus, dass Sie ein Schema für eine Kinesis Data Analytics-Anwendung, als Referenz oder bei Nichtverfügbarkeit von Live-Streaming-Daten, erstellen möchten. Sie können die Funktion Schema-Erkennung bei einer statischen Datei verwenden, die ein Beispiel der Daten im erwarteten Format Ihrer Streaming- oder Referenzdaten enthält. Kinesis Data Analytics kann die Schemaerkennung für Beispieldaten aus einer JSON- oder CSV-Datei ausführen, die in einem Amazon-S3-Bucket gespeichert ist. Wenn die Schemaerkennung auf eine Datendatei angewandt wird, wird dazu entweder die Konsole oder die DiscoverInputSchema-API mit dem festgelegten Parameter `S3Configuration` verwendet.

Ausführen der Schemaerkennung mithilfe der Konsole

Um die Erkennung für eine statische Datei mithilfe der Konsole auszuführen, gehen Sie wie folgt vor:

1. Fügen Sie ein Referenzdatenobjekt zu einem S3-Bucket hinzu.
2. Wählen Sie auf der Hauptseite der Anwendung in der Kinesis Data Analytics-Konsole Referenzdaten verbinden aus.
3. Geben Sie den Bucket, den Pfad und die IAM-Rolle für den Zugriff auf das Amazon-S3-Objekt mit den Referenzdaten ein.
4. Klicken Sie auf Discover schema (Schema erkennen).

Weitere Informationen zum Hinzufügen von Referenzdaten und zum Erkennen des Schemas in der Konsole finden Sie unter Beispiel: Hinzufügen von Referenzdaten zu einer Kinesis Data Analytics-Anwendung.

Ausführen der Schemaerkennung mithilfe der API

Um die Erkennung für eine statische Datei mithilfe der API auszuführen, geben Sie die API mit einer `S3Configuration`-Struktur mit den folgenden Informationen an:

- `BucketARN`: der Amazon-Ressourcenname (ARN) des Amazon-S3-Buckets, der die Datei enthält. Das Format eines Amazon S3-Bucket-ARN finden Sie unter Amazon Resource Names (ARNs) und Amazon Service Namespaces: Amazon Simple Storage Service (Amazon S3).
- `RoleARN`: der ARN einer IAM-Rolle mit der `AmazonS3ReadOnlyAccess`-Richtlinie. Informationen zum Hinzufügen einer Richtlinie zu einer Rolle finden Sie unter Ändern einer Rolle.
- `FileKey`: der Dateiname des Objekts.

Um ein Schema aus einem Amazon-S3-Objekt mithilfe der **`DiscoverInputSchema`**-API zu generieren

1. Stellen Sie sicher, dass Sie die AWS CLI Einrichtung haben. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS Command Line Interface ()AWS CLI ein im Abschnitt „Erste Schritte“.
2. Erstellen Sie eine Datei `data.csv` mit dem folgenden Inhalt:

   ```
   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. Melden Sie sich bei der Amazon S3 S3-Konsole unter an https://console.aws.amazon.com/s3/.
4. Erstellen Sie einen Amazon-S3-Bucket und laden Sie die von Ihnen erstellte `data.csv`-Datei hoch. Merken Sie sich den ARN des erstellten Buckets. Informationen zum Erstellen eines Amazon-S3-Buckets und zum Hochladen einer Datei finden Sie unter Erste Schritte mit Amazon-Simple-Storage-Service.
5. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole. Erstellen Sie eine Rolle mit der `AmazonS3ReadOnlyAccess`-Richtlinie. Merken Sie sich den ARN der neuen Rolle.

Informationen zum Erstellen einer Rolle finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen Amazon-Service. Informationen zum Hinzufügen einer Richtlinie zu einer Rolle finden Sie unter Ändern einer Rolle.

6. Führen Sie den folgenden `DiscoverInputSchema` Befehl in der aus AWS CLI und ersetzen Sie dabei Ihren Amazon S3 S3-Bucket und Ihre IAM-Rolle: ARNs

```
$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. Die Antwort sieht in etwa so aus:

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

Vorverarbeitung von Daten mithilfe einer Lambda-Funktion

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

Wenn die Daten in Ihrem Stream eine Formatkonvertierung, Transformation, Anreicherung oder Filterung benötigen, können Sie die Daten mithilfe einer Funktion vorverarbeiten. AWS Lambda Sie können dies vor der Ausführung von SQL-Code in der Anwendung tun, oder bevor die Anwendung ein Schema aus Ihrem Datenstrom erstellt.

Die Verwendung einer Lambda-Funktion für die Vorverarbeitung von Datensätzen ist in den folgenden Fällen nützlich:

- Umwandeln von Datensätzen aus anderen Formaten (z. B. KPL oder GZIP) in Formate, die Kinesis Data Analytics analysieren kann. Kinesis Data Analytics unterstützt derzeit JSON- oder CSV-Datenformate.
- Erweitern von Daten in ein Format, das für Operationen wie beispielsweise Aggregation oder Entdeckung von Anomalien leichter zugänglich ist. Wenn z. B. mehrere Datenwerte zusammen in einer Zeichenfolge gespeichert werden, können Sie die Daten in separate Spalten erweitern.
- Die Datenanreicherung mit anderen Amazon-Services, wie z. B. Extrapolation oder Fehlerkorrektur.
- Anwenden einer komplexen Zeichenfolgentransformation auf Datensatzfelder.
- Datenfilterung für die Bereinigung der Daten.

Verwenden einer Lambda-Funktion für die Vorverarbeitung von Datensätzen

Wenn Sie Ihre Kinesis Data Analytics-Anwendung erstellen, aktivieren Sie die Lambda-Vorverarbeitung auf der Seite Mit einer Quelle verbinden.

Verwendung einer Lambda-Funktion für die Vorverarbeitung von Datensätzen in einer Kinesis Data Analytics-Anwendung

1. Melden Sie sich bei der Managed Service for Apache Flink-Konsole unter /kinesisanalytics an AWS-Managementkonsole und öffnen Sie sie. https://console.aws.amazon.com

2. Wählen Sie auf der Seite Mit einer Quelle verbinden für Ihre Anwendung die Option Aktiviert im Bereich Datensatzvorverarbeitung mit AWS Lambda.
3. Um eine bereits von Ihnen erstellte Lambda-Funktion zu verwenden, wählen Sie die Funktion aus der Dropdown-Liste Lambda-Funktion aus.
4. Um eine neue Lambda-Funktion aus einer der Lambda-Vorverarbeitungsvorlagen zu erstellen, wählen Sie die Vorlage aus der Dropdown-Liste aus. Klicken Sie dann auf View <template name> in Lambda (<Vorlagenname> in Lambda anzeigen), um die Funktion zu bearbeiten.
5. Um eine neue Lambda-Funktion zu erstellen, wählen Sie Neu erstellen. Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Erstellen einer HelloWorld Lambda-Funktion und Erkunden Sie die Konsole im AWS Lambda Entwicklerhandbuch.
6. Wählen Sie die Version der zu verwendenden Lambda-Funktion aus, die genutzt werden soll. Um die neueste Version zu verwenden, wählen Sie $LATEST.

Wenn Sie eine Lambda-Funktion für die Datensatz-Vorverarbeitung auswählen oder erstellen, werden die Datensätze vorverarbeitet, bevor der SQL-Code Ihrer Anwendung ausgeführt wird oder Ihre Anwendung ein Schema aus den Datensätzen erstellt.

Lambda-Vorverarbeitungsberechtigungen

Zur Verwendung der Lambda-Vorverarbeitung benötigt die IAM-Rolle der Anwendung die folgende Berechtigungsrichtlinie:

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

Lambda-Vorverarbeitungsmetriken

Sie können Amazon verwenden CloudWatch , um die Anzahl der Lambda-Aufrufe, die Anzahl der verarbeiteten Byte, Erfolge und Misserfolge usw. zu überwachen. Informationen zu CloudWatch Metriken, die von der Lambda-Vorverarbeitung von Kinesis Data Analytics ausgegeben werden, finden Sie unter Amazon Kinesis Analytics Analytics-Metriken.

Verwendung AWS Lambda mit der Kinesis Producer Library

Die Kinesis Producer Library (KPL) aggregiert kleine vom Benutzer formatierte Datensätze in größere Datensätze von bis zu 1 MB, um den Durchsatz von Amazon-Kinesis-Data-Streams zu optimieren. Die Kinesis Client Library (KCL) für Java unterstützt eine Disaggregation dieser Datensätze. Sie müssen jedoch ein spezielles Modul verwenden, um die Datensätze zu deaggregieren, wenn Sie sie AWS Lambda als Verbraucher Ihrer Streams verwenden.

Den erforderlichen Projektcode und Anweisungen finden Sie in den Deaggregationsmodulen der Kinesis Producer Library. AWS Lambda GitHub Sie können die Komponenten in diesem Projekt verwenden, um KPL-serialisierte Daten AWS Lambda in Java, Node.js und Python zu verarbeiten. Die Komponenten können auch als Teil einer mehrsprachigen KCL-Anwendung verwendet werden.

Datenvorverarbeitung, Ereigniseingabe, Datenantwortmodell Model/Record

Zur Vorverarbeitung von Datensätzen muss Ihre Lambda-Funktion mit den benötigten Ereignis-Eingabedaten und Datensatz-Antwortmodellen konform sein.

Ereignis-Eingabedatenmodell

Kinesis Data Analytics liest kontinuierlich Daten aus Ihrem Kinesis-Datenstream oder Firehose-Lieferstream. Für jeden abgerufenen Stapel von Datensätzen verwaltet der Service, wie jeder Stapel an Ihre Lambda-Funktion übergeben wird. Die Funktion empfängt eine Liste der Datensätze als Eingabe. Innerhalb Ihrer Funktion durchlaufen Sie die Liste und wenden Ihre Geschäftslogik an, um Ihre Vorverarbeitungsanforderungen (wie z. B. Datenformatkonvertierung oder Anreicherung) zu erfüllen.

Das Eingabemodell für Ihre Vorverarbeitungsfunktion variiert geringfügig, je nachdem, ob die Daten von einem Kinesis-Datenstream oder einem Firehose-Lieferstream empfangen wurden.

Wenn es sich bei der Quelle um einen Firehose-Lieferstream handelt, sieht das Eingabedatenmodell für Ereignisse wie folgt aus:

Kinesis Data Firehose-Anforderungsdatenmodell

Feld	Description
`invocationId`	Die Lambda-Aufrufs-ID (zufällige GUID).

Feld	Description
`applicationArn`	Der Amazon-Ressourcenname (ARN) der Kinesis Data Analytics -Anwendung
`streamArn`	ARN des Bereitstellungs-Streams

Datensätze

Feld	Description
`recordId`	Datensatz-ID (zufällige GUID)
`kinesisFirehoseRecordMetadata`	Feld: `approximateArrivalTimestamp` — Description: Ungefähre Ankunftszeit des Datensatzes des Bereitstellungs-Streams
`data`	Base64-kodierte Quell-Datensatz-Nutzlast

Das folgende Beispiel zeigt die Eingabe aus einem Firehose-Bereitstellungs-Stream:

```
{
   "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":"49572672223665514422805246926656954630972486059535892482",
         "data":"aGVsbG8gd29ybGQ=",
         "kinesisFirehoseRecordMetadata":{
            "approximateArrivalTimestamp":1520280173
         }
      }
```

```
    ]
}
```

Wenn die Quelle ein Kinesis-Datenstrom ist, ist dies das Ereignis-Eingabemodell:

Kinesis-Streams-Anforderungsdatenmodell

Feld	Description
`invocationId`	Die Lambda-Aufrufs-ID (zufällige GUID).
`applicationArn`	ARN der Kinesis-Data-Analytics-Anwendung
`streamArn`	ARN des Bereitstellungs-Streams
Datensätze	

Feld	Description	
`recordId`	Datensatz-ID basierend auf Kinesis-Datensatz-Sequenznummer	
`kinesisStreamRecordMetadata`		

Feld	Description
`sequenceNumber`	Sequenznummern aus dem Kinesis-Stream-Datensatz
`partitionKey`	Partitionsschlüssel aus dem Kinesis-Stream-Datensatz
`shardId`	`ShardId` aus dem Kinesis-Stream-Datensatz
`approximateArrival`	Ungefähre Ankunftszeit des Datensatzes des

Feld	Description
Feld	Description
	Feld: Description Timestan: Bereitstellungs-Streams
data	Base64-kodierte Quell-Datensatz-Nutzlast

Das folgende Beispiel zeigt die Eingabe aus einem Kinesis-Daten-Stream:

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

Datensatz-Antwortmodell

Alle von Ihrer Lambda-Vorverarbeitungsfunktion (mit Datensatz IDs) zurückgegebenen Datensätze, die an die Lambda-Funktion gesendet werden, müssen zurückgegeben werden. Sie müssen die folgenden Parameter enthalten. Andernfalls weist Kinesis Data Analytics sie zurück und behandelt

sie wie Vorverarbeitungsfehler. Die Datennutzlast des Datensatzes kann entsprechend den Vorverarbeitungsanforderungen umgewandelt werden.

Antwortdatenmodell

Datensätze

Feld	Description
`recordId`	Die Datensatz-ID wird während des Aufrufs von Kinesis Data Analytics an Lambda übertragen. Der transformierte Datensatz muss dieselbe Datensatz-ID enthalten. Jede fehlende Übereinstimmung zwischen der ID des ursprüngl ichen Datensatzes und der ID des transformierten Datensatz es wird als Datenvorverarbeitungsfehler behandelt.
`result`	Der Status der Datentransformation des Datensatzes. Die möglichen Werte sind: • `Ok`: Der Datensatz wurde erfolgreich umgewandelt. Kinesis Data Analytics nimmt den Datensatz für die SQL-Verar beitung auf. • `Dropped`: Der Datensatz wurde absichtlich von Ihrer Verarbeitungs-Logik herausgenommen. Kinesis Data Analytics nimmt den Datensatz aus der SQL-Verarbeitung. Das Feld der Datennutzlast ist für einen `Dropped`-Datensat z optional. • `ProcessingFailed` : Der Datensatz konnte nicht umgewandelt werden. Kinesis Data Analytics sieht die Verarbeitung durch Ihre Lambda-Funktion als nicht erfolgrei ch an und schreibt einen Fehler in den Fehler-Stream. Weitere Informationen zum Fehler-Stream finden Sie unter Fehlerbehandlung. Das Feld der Datennutzlast ist für einen `ProcessingFailed` -Datensatz optional.
`data`	Die transformierte Datennutzlast nach der base64-Kodierung. Jede Datennutzlast kann mehrere JSON-Dokumente enthalten

Feld	Description
	, wenn die Anwendung das JSON-Upload-Datenformat verwendet. Oder es können mehrere CSV-Zeilen (mit einem Zeilentrennzeichen in jede Zeile) enthalten sein, wenn die Anwendung das CSV-Upload-Datenformat verwendet. Der Kinesis Data Analytics-Service analysiert und verarbeitet Daten, die entweder aus mehreren JSON-Dokumenten bestehen oder mehrere CSV-Zeilen enthalten, innerhalb derselben Datennutzlast erfolgreich.

Das folgende Beispiel zeigt die Ausgabe einer Lambda-Funktion:

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

Geläufige Datenverarbeitungsfehler

Dies sind die häufigsten Gründe, weshalb eine Vorverarbeitung fehlschlagen kann.

- Nicht alle Datensätze (mit Datensatz IDs) in einem Batch, die an die Lambda-Funktion gesendet werden, werden an den Kinesis Data Analytics Analytics-Dienst zurückgegeben.
- In der Antwort fehlt entweder die Datensatz-ID, der Status oder das Feld der Datennutzlast. Das Feld der Datennutzlast ist für einen `Dropped`- oder `ProcessingFailed`-Datensatz optional.
- Die Zeitüberschreitungen der Lambda-Funktion reichen nicht zur Vorverarbeitung der Daten aus.
- Die Antwort der Lambda-Funktion überschreitet die durch den AWS Lambda -Service auferlegten Antwort-Limits.

Im Falle von Fehlern bei der Datenvorverarbeitung führt Kinesis Data Analytics Lambda-Aufrufe auf derselben Gruppe von Datensätzen erneut durch, bis der Aufruf erfolgreich ist. Sie können die folgenden CloudWatch Metriken überwachen, um Einblicke in Fehler zu erhalten.

- Kinesis Data Analytics-Anwendung `MillisBehindLatest`: Gibt an, mit welcher zeitlichen Differenz eine Anwendung aus der Streaming-Quelle liest.
- Kinesis Data Analytics `InputPreprocessing` CloudWatch Analytics-Anwendungsmetriken: Zeigt unter anderem die Anzahl der Erfolge und Misserfolge an. Weitere Informationen finden Sie unter Amazon-Kinesis-Analytics-Metriken.
- AWS Lambda CloudWatch Funktionsmetriken und Protokolle.

Erstellen von Lambda-Funktionen für die Vorverarbeitung

Ihre Amazon-Kinesis-Data-Analytics-Anwendung kann Lambda-Funktionen zur Vorverarbeitung von Datensätzen bei der Aufnahme in die Anwendung verwenden. Kinesis Data Analytics bietet in der Konsole die folgenden Vorlagen als Startpunkt für die Vorverarbeitung Ihrer Daten an.

Themen

- Erstellen einer Lambda-Funktion zur Vorverarbeitung in Node.js
- Erstellen einer Lambda-Funktion zur Vorverarbeitung in Python
- Erstellen einer Lambda-Funktion zur Vorverarbeitung in Java
- Erstellen einer Lambda-Funktion zur Vorverarbeitung in .NET

Erstellen einer Lambda-Funktion zur Vorverarbeitung in Node.js

Die folgenden Vorlagen zum Erstellen einer Lambda-Funktion zur Vorverarbeitung in Node.js finden Sie in der Kinesis Data Analytics-Konsole:

Lambda-Vorlage	Sprache und Version	Description
Allgemeine Kinesis Analytics-Eingabeverarbeitung	Node.js 6.10	Ein Kinesis Data Analytics-Datensatzvorprozessor, der JSON- oder CSV-Datensätze als Eingabe empfängt und diese dann mit einem Verarbeitungsstatus zurückgibt. Verwenden Sie diesen Prozessor als Startpunkt für benutzerdefinierte Transformationslogik.

Lambda-Vorlage	Sprache und Version	Description
Komprimierte Eingabeverarbeitung	Node.js 6.10	Ein Kinesis Data Analytics-Datensatzprozessor der komprimierte JSON- oder CSV-Datensätze (GZIP- oder Deflate-komprimiert) als Eingabe empfängt und dekomprimierte Datensätze mit einem Verarbeitungsstatus zurückgibt.

Erstellen einer Lambda-Funktion zur Vorverarbeitung in Python

Die folgenden Vorlagen zum Erstellen einer Lambda-Funktion zur Vorverarbeitung in Python finden Sie in der Konsole:

Lambda-Vorlage	Sprache und Version	Description
Allgemeine Kinesis Analytics-Eingabeverarbeitung	Python 2.7	Ein Kinesis Data Analytics-Datensatzvorprozessor, der JSON- oder CSV-Datensätze als Eingabe empfängt und diese dann mit einem Verarbeitungsstatus zurückgibt. Verwenden Sie diesen Prozessor als Startpunkt für benutzerdefinierte Transformationslogik.
KPL-Eingabeverarbeitung	Python 2.7	Ein Kinesis Data Analytics-Datensatzprozessor der Kinesis Producer Library-(KPL)-Aggregate von JSON- oder CSV-Datensätzen als Eingabe empfängt und disaggregierte Datensätze mit einem Verarbeitungsstatus zurückgibt.

Erstellen einer Lambda-Funktion zur Vorverarbeitung in Java

Zum Erstellen einer Lambda-Funktion zur Vorverarbeitung von Datensätzen in Java verwenden Sie die Java-Events-Klassen.

Der folgende Code zeigt das Beispiel einer Lambda-Funktion zur Vorverarbeitung mit 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;
    }

}
```

Erstellen einer Lambda-Funktion zur Vorverarbeitung in .NET

Zum Erstellen einer Lambda-Funktion zur Vorverarbeitung in .NET verwenden Sie die .NET-Events-Klassen.

Der folgende Code zeigt das Beispiel einer Lambda-Funktion zur Vorverarbeitung von Datensätzen mit 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;
        }
    }
```

Weitere Informationen zum Erstellen von Lambda-Funktionen für die Vorverarbeitung und Ziele in .NET finden Sie unter [Amazon.Lambda.KinesisAnalyticsEvents].

Parallelisieren von Eingabe-Streams zur Steigerung des Durchsatzes

> **Note**
>
> Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie nicht bereits Kinesis Data Analytics for SQL. Weitere Informationen finden Sie unter Limits.

Amazon-Kinesis-Data-Analytics-Anwendungen können mehrere In-Application-Streams unterstützen, um eine Anwendung über den Durchsatz eines einzelnen In-Application-Eingabe-Streams hinaus zu skalieren. Weitere Informationen zu In-Application-Eingabe-Streams finden Sie unter Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's.

In fast allen Fällen skaliert Amazon Kinesis Data Analytics Ihre Anwendung, um die Kapazität der Kinesis-Streams oder Firehose-Quellstreams zu bewältigen, die in Ihre Anwendung eingespeist werden. Wenn der Durchsatz Ihres Quell-Streams jedoch den Durchsatz eines einzelnen In-Application-Eingabe-Streams überschreitet, können Sie die Zahl der von Ihrer Anwendung verwendeten In-Application-Eingabe-Streams explizit erhöhen. Sie führen dies mithilfe des Parameters `InputParallelism` aus.

Wenn der Parameter `InputParallelism` größer als eins ist, teilt Amazon-Kinesis-Data-Analytics die Partitionen Ihres Quell-Streams gleichmäßig auf die In-Application-Streams auf. Wenn Ihr Quell-Stream beispielsweise über 50 Shards verfügt und Sie `InputParallelism` auf 2 festgelegt haben, erhält jeder In-Application-Eingabe-Stream die Eingabe von 25 Quell-Stream-Shards.

Wenn Sie die Zahl der In-Application-Streams erhöhen, muss Ihre Anwendung auf die Daten in jedem Stream explizit zugreifen. Weitere Informationen zum Zugriff auf mehrere In-Application-Streams in Ihrem Code finden Sie unter Zugriff auf getrennte In-Application-Streams in Ihrer Amazon-Kinesis-Data-Analytics-Anwendung.

Kinesis Data Streams und Firehose-Stream-Shards sind zwar beide auf dieselbe Weise auf anwendungsinterne Streams aufgeteilt, unterscheiden sich jedoch darin, wie sie Ihrer Anwendung angezeigt werden:

- Die Datensätze aus einem Kinesis-Datenstrom enthalten ein `shard_id`-Feld, über das der Quell-Shard des Datensatzes identifiziert werden kann.
- Die Datensätze aus einem Firehose-Lieferstream enthalten kein Feld, das den Quell-Shard oder die Quellpartition des Datensatzes identifiziert. Dies liegt daran, dass Firehose diese Informationen aus Ihrer Anwendung abstrahiert.

Testen, ob die Zahl der In-Application-Eingabe-Streams erhöht werden sollte

In der Mehrzahl der Fälle kann ein einzelner In-Application-Eingabe-Stream den Durchsatz eines einzelnen Quell-Streams verarbeiten, abhängig von Komplexität und Größe der Eingabe-Streams. Um festzustellen, ob Sie die Anzahl der In-Application-Input-Streams erhöhen müssen, können Sie die `InputBytes` und `MillisBehindLatest` -Metriken in Amazon CloudWatch überwachen.

Wenn die `InputBytes` Kennzahl größer als 100 ist MB/sec (oder Sie erwarten, dass sie diese Rate übersteigen wird), kann dies zu einer Zunahme `MillisBehindLatest` und Erhöhung der Auswirkungen von Anwendungsproblemen führen. Um dies zu Adressieren, empfehlen wir die folgenden Sprachauswahl für Ihre Anwendung:

- Verwenden Sie mehrere Streams und Kinesis-Data-Analytics-for-SQL-Anwendungen, wenn die Skalierungsanforderungen Ihrer Anwendung 100 MB/Sekunde überschreiten.
- Verwenden Sie Kinesis Data Analytics for Java-Anwendungen, wenn Sie einen einzelnen Stream und eine einzelne Anwendung verwenden möchten.

Wenn die `MillisBehindLatest`-Metrik eines der folgenden Merkmale aufweist, sollten Sie die `InputParallelism`-Einstellung Ihrer Anwendung erhöhen:

- Die `MillisBehindLatest`-Metrik steigt schrittweise an. Das bedeutet, dass Ihre Anwendung hinter die neuesten Daten im Stream zurückfällt.
- Die `MillisBehindLatest`-Metrik liegt konsistent über 1000 (eine Sekunde).

Sie müssen die `InputParallelism`-Einstellung Ihrer Anwendung nicht erhöhen, wenn Folgendes zutrifft:

- Die `MillisBehindLatest`-Metrik nimmt schrittweise ab. Das bedeutet, dass Ihre Anwendung zu den neuesten Daten im Stream aufholt.
- Die `MillisBehindLatest`-Metrik liegt unter 1000 (eine Sekunde).

Weitere Informationen zur Verwendung CloudWatch finden Sie im CloudWatch Benutzerhandbuch.

Implementieren mehrerer In-Application-Eingabe-Streams

Sie können beim Erstellen einer Anwendung die Anzahl der In-Application-Eingabe-Streams mittels CreateApplication festlegen. Nach der Erstellung einer Anwendung legen Sie diese Zahl mittels UpdateApplication fest.

> **Note**
>
> Sie können die `InputParallelism`-Einstellung nur mithilfe der Amazon-Kinesis-Data-Analytics-API oder der AWS CLI festlegen. Sie können diese Einstellung nicht mit dem

festlegen AWS-Managementkonsole. Informationen zur Einrichtung von finden Sie AWS CLI unterSchritt 2: Richten Sie das AWS Command Line Interface ()AWS CLI ein.

Festlegen der Zahl der Eingabe-Streams für eine neue Anwendung

Im folgenden Beispiel wird gezeigt, wie Sie mit der API-Aktion `CreateApplication` die Zahl der Eingabe-Streams einer neuen Anwendung auf 2 festzulegen.

Mehr über `CreateApplication` erfahren Sie unter 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": [ ... ],
 }]
}
```

Festlegen der Zahl der Eingabe-Streams für eine vorhandene Anwendung

Im folgenden Beispiel wird gezeigt, wie Sie mit der API-Aktion `UpdateApplication` die Zahl der Eingabe-Streams einer vorhandenen Anwendung auf 2 festzulegen.

Mehr über `Update_Application` erfahren Sie unter UpdateApplication.

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

```
}
```

Zugriff auf getrennte In-Application-Streams in Ihrer Amazon-Kinesis-Data-Analytics-Anwendung

Um mehrere In-Application-Eingabe-Streams in Ihrer Anwendung zu verwenden, müssen Sie diese explizit aus den verschiedenen Streams auswählen. Das folgende Codebeispiel zeigt, wie Sie mehrere Eingabe-Streams in der Anwendung abfragen, die Sie in der Einführungsübung erstellt haben.

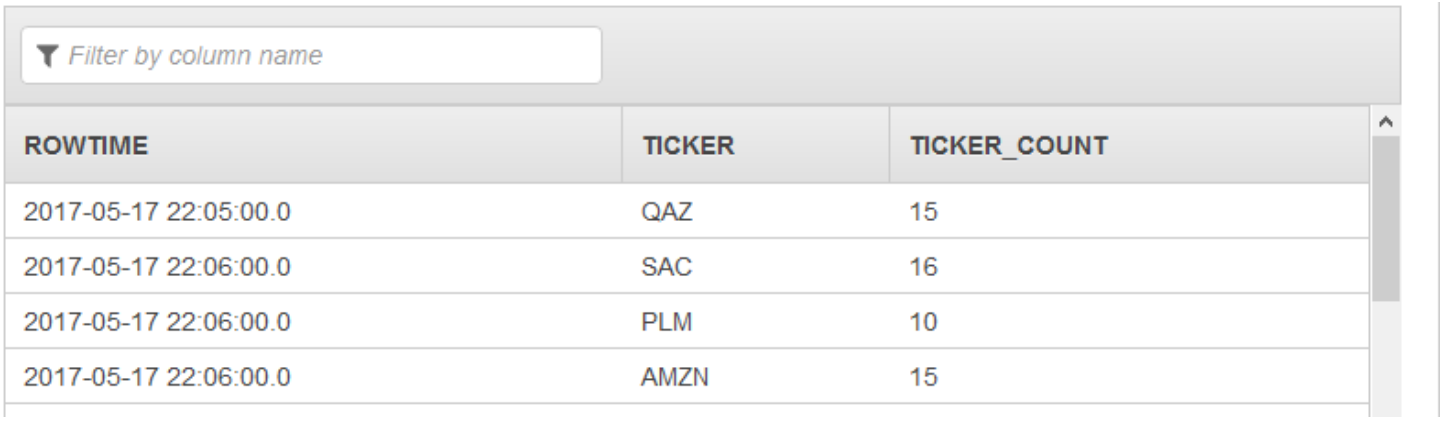
Im folgenden Beispiel werden die einzelnen Quell-Streams zunächst mittels COUNT zusammengefasst, bevor sie zu einem einzelnen In-Application-Stream mit dem Namen `in_application_stream001` kombiniert werden. Die Zusammenfassung der Quell-Streams im Voraus hilft, sicherzustellen, dass der kombinierte In-Application-Stream den Datenverkehr aus mehreren Streams verarbeiten kann, ohne überlastet zu werden.

> **Note**
>
> Um dieses Beispiel auszuführen und Ergebnisse aus beiden In-Application-Eingabe-Streams zu erhalten, müssen Sie die Anzahl der Shards in Ihrem Quell-Stream und den Parameter `InputParallelism` in Ihrer Anwendung aktualisieren.

```
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;
```

Das vorherige Codebeispiel produziert im `in_application_stream001` eine Ausgabe ähnlich der folgenden:



Weitere Überlegungen

Beachten Sie Folgendes, wenn Sie mehrere Eingabe-Streams verwenden:

- Die maximale Anzahl der In-Application-Streams ist 64.
- Die In-Application-Eingabe-Streams werden gleichmäßig auf die Shards des Eingabe-Streams der Anwendung aufgeteilt.
- Die Leistung wird durch das Hinzufügen von In-Application-Streams nicht linear skaliert. Das bedeutet, dass eine Verdopplung der Anzahl der In-Application-Streams nicht den Durchsatz verdoppelt. Bei einer typischen Zeilengröße kann jeder In-Application-Stream einen Durchsatz von ungefähr 5.000 bis 15.000 Zeilen pro Sekunde erreichen. Durch die Erhöhung der Zahl der In-Application-Streams auf 10 können Sie einen Durchsatz von 20.000 bis 30.000 Zeilen pro Sekunde erreichen. Die Durchsatzgeschwindigkeit ist von der Zahl, den Datentypen und der Datengröße der Felder im Eingabe-Stream abhängig.
- Einige Zusammenfassungsfunktionen (wie AVG) können zu unerwarteten Ergebnissen führen, wenn sie auf in verschiedene Shards aufgeteilte Eingabe-Streams angewendet werden. Da Sie die Zusammenfassungsoperation vor der Zusammenfassung zu einem einzelnen Stream auf einzelnen Shards ausführen müssen, werden die Ergebnisse möglicherweise in Richtung auf den Stream gewichtet, der eine größere Zahl von Datensätzen enthält.
- Wenn Ihre Anwendung weiterhin eine schlechte Leistung aufweist (was sich in einer hohen `MillisBehindLatest` Metrik widerspiegelt), nachdem Sie die Anzahl der Eingabestreams erhöht haben, haben Sie möglicherweise Ihr Limit an Kinesis Processing Units (KPUs) erreicht. Weitere

Informationen finden Sie unter Automatisches Skalieren von Anwendungen zur Erhöhung des Durchsatzes.

Anwendungscode

Der Anwendungscode besteht aus einer Reihe von SQL-Anweisungen, die Eingabedaten verarbeiten und Ausgabedaten erzeugen. Diese SQL-Anweisungen verarbeiten In-Application-Streams und Referenztabellen. Weitere Informationen finden Sie unter Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's.

Weitere Informationen zu den SQL-Sprachelementen, die von Kinesis Data Analytics unterstützt werden, finden Sie in der Amazon-Kinesis-Data-Analytics-SQL-Referenz.

In relationalen Datenbanken arbeiten Sie mit Tabellen und verwenden dabei INSERT-Anweisungen, um Datensätze und die SELECT-Anweisung zur Abfrage der Daten hinzuzufügen. In Amazon-Kinesis-Data-Analytics arbeiten Sie mit Streams. Sie können eine SQL-Anweisung zur Abfrage dieser Streams schreiben. Die Ergebnisse der Abfrage eines In-Application-Streams werden stets an einen anderen In-Application-Stream gesendet. Bei der Ausführung komplexer Analysen können Sie mehrere In-Application-Streams erstellen, die die Ergebnisse zwischengeschalteter Analysen enthalten. Und schließlich können Sie die Anwendungsausgabe so konfigurieren, dass die Ergebnisse der letzten Analyse (aus einem oder mehreren In-Application-Streams) an externe Ziele weitergeleitet werden. Zusammenfassend kann gesagt werden, dass das folgende Muster für das Schreiben von Anwendungs-Code typisch ist:

- Die SELECT-Anweisung wird stets im Zusammenhang mit einer INSERT-Anweisung verwendet. Das bedeutet, dass Sie Ergebnisse in einen anderen In-Application-Stream einfügen, wenn Sie Zeilen auswählen.
- Die INSERT-Anweisung wird stets im Zusammenhang mit einer Pump verwendet. Das bedeutet, dass Sie Pumps verwenden, um zu einem In-Application-Stream zu schreiben.

Das folgende Beispiel für einen Anwendungs-Code liest Datensätze aus einem In-Application-Stream (SOURCE_SQL_STREAM_001) und schreibt diese zu einem anderen In-Application-Stream (DESTINATION_SQL_STREAM). Sie können Datensätze mittels Pumps in In-Application-Streams einfügen, wie im Folgenden gezeigt:

```
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**
>
> Die Kennungen, die Sie für Stream- und Spaltennamen angeben, folgen SQL-Standardkonventionen. Wenn Sie beispielsweise eine Kennung in Anführungszeichen setzen, wird bei der Kennung zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie keine Anführungszeichen setzen, werden für die Kennung standardmäßig Großbuchstaben verwendet. Weitere Informationen zu Kennungen finden Sie unter Kennungen in der Amazon-Managed-Service für Apache Flink-SQL-Referenz.

Ihr Anwendungs-Code kann aus zahlreichen SQL-Anweisungen bestehen. Zum Beispiel:

- Sie können SQL-Abfragen sequenziell schreiben, sodass das Ergebnis einer SQL-Anweisung für die nächste SQL-Anweisung verwendet wird.
- Sie können auch SQL-Abfragen schreiben, die unabhängig voneinander ausgeführt werden. Sie können beispielsweise zwei SQL-Anweisungen schreiben, die denselben In-Application-Stream abfragen, die Ausgabe jedoch an verschiedene In-Application-Streams senden. Anschließend können Sie die neu erstellten In-Application-Streams unabhängig voneinander abfragen.

Sie können In-Application-Streams erstellen, um Zwischenergebnisse zu speichern. Sie fügen Daten mittels Pumpen in In-Application-Streams ein. Weitere Informationen finden Sie unter In-Application-Streams und Pumps.

Wenn Sie eine In-Application-Referenztabelle hinzufügen, können Sie SQL schreiben, um Daten in In-Application-Streams und Referenztabellen zusammenzuführen. Weitere Informationen finden Sie unter Beispiel: Hinzufügen von Referenzdaten zu einer Kinesis Data Analytics-Anwendung.

Amazon-Kinesis-Data-Analytics schreibt Daten entsprechend der Ausgabekonfiguration der Anwendung aus bestimmten In-Application-Streams zum externen Ziel. Stellen Sie sicher, dass Ihr Anwendungs-Code zu den In-Application-Streams schreibt, die in der Ausgabekonfiguration angegeben sind.

Weitere Informationen finden Sie unter den folgenden Themen:

- SQL-Streaming-Konzepte
- Amazon-Kinesis-Data-Analytics SQL-Referenz

Konfigurieren der Anwendungsausgabe

In Ihrem Anwendungscode schreiben Sie die Ausgabe von SQL-Anweisungen in einen oder mehrere In-Application-Streams. Sie können Ihrer Anwendung optional eine Ausgabekonfiguration hinzufügen, um alles, was in einen anwendungsinternen Stream geschrieben wurde, an einem externen Ziel wie einem Amazon Kinesis Kinesis-Datenstream, einem Firehose-Lieferstream oder einer Funktion zu persistieren. AWS Lambda

Es gibt eine Begrenzung für die Anzahl der externen Ziele, an die Sie eine Anwendungsausgabe weiterleiten können. Weitere Informationen finden Sie unter Einschränkungen.

> **Note**
>
> Wir empfehlen die Verwendung eines externen Ziels, an das die Daten aus dem In-Application-Fehler-Stream weitergeleitet werden, damit Sie die Fehler untersuchen können.

In jeder dieser Ausgabekonfigurationen stellen Sie Folgendes bereit:

- Name des In-Application-Streams – Der Stream, den Sie an ein externes Ziel weiterleiten möchten.

  Kinesis Data Analytics sucht nach dem In-Application-Stream, den Sie in der Ausgabekonfiguration angegeben haben. (Beim Streamnamen wird zwischen Groß- und Kleinschreibung unterschieden und er muss exakt übereinstimmen.) Stellen Sie sicher, dass Ihr Anwendungscode diesen anwendungsinternen Stream erstellt.
- Externes Ziel — Sie können Daten in einem Kinesis-Datenstream, einem Firehose-Lieferstream oder einer Lambda-Funktion persistieren. Sie geben den Amazon-Ressourcennamen (ARN) des Streams oder der Funktion an. Sie stellen außerdem eine IAM-Rolle bereit, die Kinesis Data Analytics übernehmen kann, um in Ihrem Namen in den Stream oder schreiben oder Funktionen auszuführen. Sie beschreiben das Datensatzformat (JSON, CSV), das Kinesis Data Analytics beim Schreiben in das externe Ziel verwenden soll.

Wenn Kinesis Data Analytics nicht in das Streaming- oder Lambda-Ziel schreiben kann, führt der Service für unbegrenzte Zeit Versuche aus. Dies führt zu einer hohen Auslastung und Ihre Anwendung fällt zurück. Wenn dieses Problem nicht behoben wird, verarbeitet Ihre Anwendung ab einem gewissen Zeitpunkt keine neuen Daten mehr. Sie können Amazon-Kinesis-Analytics-Metriken überwachen und Alarmen für Ausfälle festlegen. Weitere Informationen zu Metriken und Alarmen finden Sie unter Amazon CloudWatch Metrics verwenden und CloudWatchAmazon-Alarme erstellen.

Sie können die Anwendungsausgabe mittels der AWS-Managementkonsole konfigurieren. Die Konsole führt den API-Aufruf aus, um die Konfiguration zu speichern.

Erstellen einer Ausgabe mit dem AWS CLI

In diesem Abschnitt wird beschrieben, wie der Abschnitt `Outputs` des Anforderungstextes für eine `CreateApplication`- oder `AddApplicationOutput`-Operation erstellt wird.

Erstellen einer Kinesis Stream-Ausgabe

Das folgende JSON-Fragment zeigt den Abschnitt `Outputs` im `CreateApplication`-Anforderungstext zum Erstellen eines Ziels für den Amazon-Kinesis-Datenstrom.

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

]
```

Erstellen einer Firehose Delivery Stream-Ausgabe

Das folgende JSON-Fragment zeigt den `Outputs` Abschnitt im `CreateApplication` Anforderungstext für die Erstellung eines Amazon Data Firehose-Lieferstream-Ziels.

```
"Outputs": [
   {
```

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

Erstellen einer Lambda-Funktion

Das folgende JSON-Fragment zeigt den `Outputs` Abschnitt im Hauptteil der `CreateApplication` Anfrage zum Erstellen eines AWS Lambda Funktionsziels.

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

Verwenden einer Lambda-Funktion als Ausgabe

Wenn Sie es AWS Lambda als Ziel verwenden, können Sie Ihre SQL-Ergebnisse einfacher nachbearbeiten, bevor Sie sie an ein endgültiges Ziel senden. Zu gängigen Aufgaben der Nachverarbeitung gehören:

- Aggregieren von mehreren Zeilen in einem einzigen Datensatz
- Kombinieren aktueller Ergebnisse mit vorherigen Ergebnissen für die Einordnung spät eintreffender Daten
- Bereitstellung auf verschiedene Ziele abhängig von der Art von Informationen
- Aufzeichnung von Formatumsetzungen (z. B. die Umsetzung in Protobuf)

- Bearbeitung oder Transformation von Zeichenfolgen
- Datenanreicherung nach der analytischen Verarbeitung
- Benutzerdefinierte Verarbeitung für koordinatenbasierte Anwendungsfälle
- Datenverschlüsselung

Lambda-Funktionen können analytische Informationen für eine Vielzahl von AWS Diensten und anderen Zielen bereitstellen, darunter die folgenden:

- Amazon Simple Storage Service (Amazon-S3)
- Benutzerdefiniert APIs
- Amazon-DynamoDB
- Amazon Aurora
- Amazon-Redshift
- Amazon-Simple-Notification-Service (Amazon-SNS)
- Amazon-Simple-Queue-Service (Amazon SQS)
- Amazon CloudWatch

Weitere Informationen zum Erstellen von Lambda-Anwendungen finden Sie unter Erste Schritte mit AWS Lambda.

Themen

- Berechtigungen für Lambda als Ausgabe
- Lambda als Ausgabemetriken
- Ereignis-Eingabedatenmodell und Datensatz-Antwortmodell von Lambda als Ausgabe
- Häufigkeit der Lambda-Ausgabeaufrufe
- Hinzufügen einer Lambda-Funktion zur Verwendung als Ausgabe
- Häufige Lambda-Ausgabefehler
- Erstellen von Lambda-Funktionen für Anwendungsziele

Berechtigungen für Lambda als Ausgabe

Um Lambda als Ausgabe verwenden zu können, benötigt die IAM-Rolle für die Lambda-Ausgabe der Anwendung die folgende Berechtigungsrichtlinie:

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

Lambda als Ausgabemetriken

Sie verwenden Amazon CloudWatch , um die Anzahl der gesendeten Byte, Erfolge und Misserfolge usw. zu überwachen. Informationen zu CloudWatch Metriken, die von Kinesis Data Analytics mithilfe von Lambda als Ausgabe ausgegeben werden, finden Sie unter Amazon Kinesis Analytics Analytics-Metriken.

Ereignis-Eingabedatenmodell und Datensatz-Antwortmodell von Lambda als Ausgabe

Um Kinesis Data Analytics-Ausgabedatensätze senden zu können, muss Ihre Lambda-Funktion mit den erforderlichen Ereignis-Eingabedaten- und Datensatz-Antwortmodellen konform sein.

Ereignis-Eingabedatenmodell

Kinesis Data Analytics sendet die Ausgabedatensätze mit dem folgenden Anforderungsmodell fortlaufend von der Anwendung als Ausgabefunktion an die Lambda-Funktion. Innerhalb Ihrer Funktion durchlaufen Sie die Liste und wenden Ihre Geschäftslogik an, um Ihre Ausgabeanforderungen (z. B. Datentransformation vor dem senden an ein endgültiges Ziel) zu erfüllen.

Feld	Description
`invocationId`	Die Lambda -Aufrufs-ID (zufällige GUID).
`applicationArn`	Der Amazon-Ressourcenname (ARN) der Kinesis Data Analytics -Anwendung.
Datensätze	

Feld	Description
Feld	Description
`recordId`	Datensatz-ID (zufällige GUID)
`lambdaDeliveryRecordMetadata`	Feld: `retryHint` — Description: Anzahl der wiederholten Nachrichtenzustellungen
data	Base64-kodierte Augabedatensatz-Nutzlast

> **Note**
>
> `retryHint` ist ein Wert, der sich bei jedem Zustellungsfehler erhöht. Dieser Wert wird nicht dauerhaft gespeichert und wird zurückgesetzt werden, wenn die Anwendung unterbrochen wird.

Datensatz-Antwortmodell

Jeder Datensatz, der als Ausgabefunktion (mit Datensatz IDs) an Ihr Lambda gesendet wird`DeliveryFailed`, muss entweder mit `0k` oder bestätigt werden und er muss die folgenden Parameter enthalten. Andernfalls wird er von Kinesis Data Analytics als unzustellbar behandelt.

Datensätze

Feld	Description
`recordId`	Die Datensatz-ID wird während des Aufrufs von Kinesis Data Analytics an Lambda übertragen. Jede fehlende Übereinst immung zwischen der ID des ursprünglichen Datensatzes und

Feld	Description
	der ID des bestätigten Datensatzes wird als Zustellungsfehler behandelt.
`result`	Der Status der Datensatzzustellung. Folgende Werte sind möglich: • `Ok`: Der Datensatz wurde erfolgreich umgewandelt und an das endgültige Ziel gesendet. Kinesis Data Analytics nimmt den Datensatz für die SQL-Verarbeitung auf. • `DeliveryFailed` : Der Datensatz wurde dem endgültig en Ziel durch Lambda als Ausgabefunktion nicht erfolgrei ch zugestellt. Kinesis Data Analytics versucht fortgesetzt, die Datensätze mit Zustellungsfehlern erneut an Lambda als Ausgabefunktion zu senden.

Häufigkeit der Lambda-Ausgabeaufrufe

Eine Kinesis Data Analytics-Anwendung puffert die Ausgabedatensätze und ruft häufig die AWS Lambda -Zielfunktion auf.

- Wenn Datensätze als Taumelfenster an den anwendungsinternen Zielstream innerhalb der Datenanalyseanwendung ausgegeben werden, wird die AWS Lambda Zielfunktion pro Taumelfenster-Trigger aufgerufen. Wenn z. B. ein rollierendes Fenster von 60 Sekunden verwendet wird, um die Datensätze an den In-Application-Stream des Ziels auszugeben, wird die Lambda-Funktion einmal alle 60 Sekunden aufgerufen.
- Wenn Datensätze innerhalb der Anwendung als fortlaufende Abfrage oder gleitendes Fenster zum In-Application-Stream des Ziels gesendet werden, wird die Lambda-Zielfunktion ca. einmal pro Sekunde aufgerufen.

> **Note**
>
> Pro Aufruf der Lambda-Funktion gelten Nutzlastgrenzwerte. Beim Überschreiten dieser Limits erhalten Sie Ausgabedatensätze, die über mehrere Lambda-Funktionsaufrufe aufgeteilt und gesendet werden.

Hinzufügen einer Lambda-Funktion zur Verwendung als Ausgabe

Das folgende Verfahren veranschaulicht, wie Sie die Lambda-Funktion als Ausgabe für eine Kinesis Data Analytics-Anwendung hinzufügen.

1. Melden Sie sich bei der Managed Service for Apache Flink-Konsole unter /kinesisanalytics an AWS-Managementkonsole und öffnen Sie sie. https://console.aws.amazon.com
2. Wählen Sie die Anwendung in der Liste aus und klicken Sie dann auf Application details.
3. Klicken Sie im Bereich Destination auf Connect new destination.
4. Wählen Sie für das Element Destination (Ziel) die Option AWS Lambda function (-Funktion).
5. Wählen Sie im Bereich Datensätze an AWS Lambda liefern entweder eine vorhandene Lambda-Funktion aus oder klicken Sie auf Neu erstellen.
6. Wenn Sie eine neue Lambda-Funktion erstellen, verfahren Sie wie folgt:

    a. Wählen Sie eine der bereitgestellten Vorlagen. Weitere Informationen finden Sie unter Erstellen von Lambda-Funktionen für Anwendungsziele.

    b. Die Seite Create Function (Funktion erstellen) wird in einer neuen Webbrowser-Registerkarte geöffnet. Geben Sie im Feld Name der Funktion einen sinnvollen Namen (z. B. **`myLambdaFunction`**).

    c. Aktualisieren Sie die Vorlage mit Nachverarbeitungsfunktionalität für Ihre Anwendung. Weitere Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Erste Schritte im AWS Lambda -Entwicklerhandbuch.

    d. Wählen Sie in der Kinesis Data Analytics-Konsole aus der Liste Lambda-Funktionen die gerade erstellte Lambda-Funktion aus. Wählen Sie $LATEST für die Lambda-Funktionsversion.
7. Wählen Sie im Bereich In-application stream die Option Choose an existing in-application stream aus. Wählen Sie für In-application stream name den Ausgabe-Stream Ihrer Anwendung aus. Die Ergebnisse aus dem ausgewählten Ausgabe-Stream werden zur Lambda-Ausgabefunktion gesendet.
8. Belassen Sie im Formular die übrigen Standardwerte und wählen Sie Save and continue.

Ihre Anwendung sendet nun Datensätze aus dem In-Application-Stream zu Ihrer Lambda-Funktion. Sie können die Ergebnisse der Standardvorlage in der CloudWatch Amazon-Konsole sehen. Überwachen Sie die Metrik `AWS/KinesisAnalytics/LambdaDelivery.OkRecords`, um die Anzahl der Datensätze zu sehen, die an die Lambda-Funktion übermittelt werden.

Häufige Lambda-Ausgabefehler

Nachfolgend werden die häufigsten Gründe aufgeführt, aus denen die Übermittlung an die Lambda-Funktion fehlschlagen kann.

- Nicht alle Datensätze (mit Datensatz IDs) in einem Batch, die an die Lambda-Funktion gesendet werden, werden an den Kinesis Data Analytics Analytics-Dienst zurückgegeben.
- In der Antwort fehlt entweder die Datensatz-ID oder das Statusfeld.
- Die Zeitüberschreitungen der Lambda-Funktion sind für den Abschluss der Geschäftslogik innerhalb der Lambda-Funktion nicht ausreichend.
- Die Geschäftslogik innerhalb der Lambda-Funktion erfasst nicht alle Fehler. Dies führt aufgrund unbehandelter Ausnahmen zu einer Zeitüberschreitung und einer zu hohen Auslastung. Diese werden oft als Poison-Pill-Nachrichten bezeichnet.

Bei Datenübermittlungsfehlern führt Kinesis Data Analytics Lambda-Aufrufe für dieselbe Gruppe von Datensätzen erneut durch, bis der Aufruf erfolgreich ist. Um Einblick in Fehler zu erhalten, können Sie die folgenden CloudWatch Metriken überwachen:

- Kinesis Data Analytics Analytics-Anwendung Lambda as CloudWatch Output-Metriken: Gibt unter anderem die Anzahl der Erfolge und Misserfolge an. Weitere Informationen finden Sie unter Amazon-Kinesis-Analytics-Metriken.
- AWS Lambda CloudWatch Funktionsmetriken und Protokolle.

Erstellen von Lambda-Funktionen für Anwendungsziele

Ihre Kinesis Data Analytics Analytics-Anwendung kann AWS Lambda Funktionen als Ausgabe verwenden. Kinesis Data Analytics stellt Vorlagen zum Erstellen von Lambda-Funktionen zur Verfügung, die als Ziel Ihrer Anwendung verwendet werden. Diese Vorlagen dienen als Ausgangspunkt für die Nachverarbeitung der Ausgabe Ihrer Anwendung.

Themen

- Erstellen eines Lambda-Funktionsziels in Node.js
- Erstellen eines Lambda-Funktionsziels in Python
- Erstellen eines Lambda-Funktionsziels in Java
- Erstellen eines Lambda-Funktionsziels in .NET

Erstellen eines Lambda-Funktionsziels in Node.js

Die folgende Vorlage zum Erstellen eines Lambda-Funktionsziels in Node.js finden Sie in der Konsole:

Lambda als Ausgabe-Blueprint	Sprache und Version	Description
`kinesis-analytics-output`	Node.js 12.x	Übermitteln von Ausgabedatensätzen von einer Kinesis Data Analytics-Anwendung an ein benutzerdefiniertes Ziel.

Erstellen eines Lambda-Funktionsziels in Python

Die folgenden Vorlagen zum Erstellen eines Lambda-Funktionsziels in Python finden Sie in der Konsole:

Lambda als Ausgabe-Blueprint	Sprache und Version	Description
`kinesis-analytics-output-sns`	Python 2.7	Stellen Sie Ausgabedatensätze aus einer Kinesis Data Analytics-Anwendung an Amazon-SNS bereit.
`kinesis-analytics-output-ddb`	Python 2.7	Stellen Sie Ausgabedatensätze aus einer Kinesis Data Analytics-Anwendung an Amazon-DynamoDB bereit.

Erstellen eines Lambda-Funktionsziels in Java

Zum Erstellen eines Lambda-Funktionsziels in Java verwenden Sie die Java Events-Klassen.

Der folgende Code zeigt das Beispiel eines Lambda-Funktionsziels mit 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;
    }

}
```

Erstellen eines Lambda-Funktionsziels in .NET

Zum Erstellen eines Lambda-Funktionsziels in .NET verwenden Sie die .NET-Events-Klassen.

Der folgende Code zeigt das Beispiel eines Lambda-Funktionsziels mit 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;
        }
    }
```

Weitere Informationen zum Erstellen von Lambda-Funktionen für die Vorverarbeitung und Ziele in .NET finden Sie unter [Amazon.Lambda.KinesisAnalyticsEvents].

Bereitstellungsmodell für die Weiterleitung der Anwendungsausgabe an ein externes Ziel

Amazon-Kinesis-Data-Analytics verwendet für die Anwendungsausgabe an die konfigurierten Ziele ein Bereitstellungsmodell nach dem Verfahren „mindestens einmal“. Wenn die Anwendung ausgeführt wird, erstellt Kinesis Data Analytics interne Checkpoints. Diese Checkpoints sind Zeitpunkte, an denen die Ausgabedaten ohne Datenverluste an die Ziele geliefert wurden. Der Service verwendet die Checkpoints nach Bedarf, um sicherzustellen, dass die Ausgabe Ihrer Anwendungen mindestens einmal an die konfigurierten Ziele geliefert wird.

In einer normalen Situation verarbeitet Ihre Anwendung eingehende Daten kontinuierlich. Kinesis Data Analytics schreibt die Ausgabe an die konfigurierten Ziele, z. B. einen Kinesis-Datenstream oder einen Firehose-Lieferstream. Ihre Anwendung kann jedoch gelegentlich unterbrochen werden, z. B.:

- Sie möchten möglicherweise die Anwendung anhalten und später erneut starten.

- Sie löschen die IAM-Rolle, die Kinesis Data Analytics benötigt, um Ihre Anwendungsausgabe zum konfigurierten Ziel zu schreiben. Ohne die IAM-Rolle besitzt Kinesis Data Analytics keine Berechtigungen, um in Ihrem Namen in das externe Ziel zu schreiben.
- Ein Netzwerkausfall oder der Ausfall eines anderen internen Service führt dazu, dass Ihre Anwendung vorübergehend angehalten wird.

Wenn Ihre Anwendung neu gestartet wird, stellt Kinesis Data Analytics sicher, dass Ausgabedaten weiterhin ab einem Punkt verarbeitet und geschrieben werden, der vor oder an dem Punkt liegt, an dem der Fehler aufgetreten ist. Dadurch wird sichergestellt, dass alle Anwendungsausgaben an die konfigurierten Ziele geliefert werden.

Gehen wir davon aus, dass Sie mehrere Ziele für den selben In-Application-Stream konfiguriert haben. Nachdem die Anwendung nach dem Fehler wiederhergestellt wurde, setzt Kinesis Data Analytics die Ausgabe an die konfigurierten Ziele ab dem letzten Datensatz fort, der an das langsamste Ziel geliefert wurde. Dies kann dazu führen, dass derselbe Ausgabedatensatz mehr als einmal für andere Ziele bereitgestellt wird. In diesem Fall müssen Sie potenzielle Doppelerfassungen im Ziel extern behandeln.

Fehlerbehandlung

Amazon-Kinesis-Data-Analytics gibt API- oder SQL-Fehler direkt an Sie zurück. Weitere Informationen über API-Operationen finden Sie im Abschnitt Aktionen. Weitere Informationen zur Behandlung von SQL-Fehlern finden Sie in der Amazon-Kinesis-Data-Analytics -SQL-Referenz.

Amazon-Kinesis-Data-Analytics meldet Laufzeitfehler über einen In-Application-Stream mit dem Namen `error_stream`.

Melden von Fehlern über einen In-Application-Stream

Amazon-Kinesis-Data-Analytics meldet Laufzeitfehler an den In-Application-Stream mit dem Namen `error_stream`. Im Folgenden finden Sie Beispiele für Fehler, die auftreten könnten:

- Ein aus der Streaming-Quelle gelesener Datensatz entspricht nicht dem Eingabeschema.
- Ihr Anwendungscode legt eine Division durch Null fest.
- Die Zeilen werden nicht in der richtigen Reihenfolge angezeigt (z. B. wird im Stream ein Datensatz mit einem durch einen Benutzer geänderten `ROWTIME`-Wert angezeigt, der bewirkt, dass ein Datensatz nicht in der richtigen Reihenfolge angezeigt wird).

- Die Daten im Quell-Stream können nicht in den im Schema festgelegten Datentyp umgewandelt werden (Erzwingungsfehler). Informationen darüber, welche Datentypen konvertiert werden können, finden Sie unter Zuweisung von JSON-Datentypen zu SQL-Datentypen.

Wir empfehlen, diese Fehler entweder programmgesteuert in Ihrem SQL-Code zu behandeln oder die Daten im Fehler-Stream dauerhaft an einem externen Ziel zu speichern. In diesem Fall müssen Sie Ihrer Anwendung Ausgabekonfiguration hinzufügen (siehe Konfigurieren der Anwendungsausgabe). Ein Beispiel für die Funktionsweise des In-Application-Streams finden Sie unter Beispiel: Erkunden des In-Application-Fehler-Streams.

> **Note**
>
> Ihre Kinesis Data Analytics-Datenanalyseanwendung kann nicht programmgesteuert auf den Fehler-Stream zugreifen oder ihn ändern, da der Fehler-Stream über das Systemkonto erstellt wurde. Sie müssen anhand der Fehlerausgabe bestimmen, welche Fehler in Ihrer Anwendung auftreten können. Anschließend schreiben Sie den SQL-Code Ihrer Anwendung so, dass erwartete Fehlerbedingungen verarbeitet werden.

Fehler-Stream-Schema

Der Fehler-Stream weist das folgende Schema auf:

Feld	Datentyp	Hinweise
ERROR_TIME	TIMESTAMP (ZEITSTEMPEL)	Der Zeitpunkt, an dem der Fehler auftrat
ERROR_LEVEL	VARCHAR(10)	
ERROR_NAME	VARCHAR (32)	
MESSAGE	VARCHAR (4096)	
DATA_ROWTIME	TIMESTAMP (ZEITSTEMPEL)	Die Zeilenzeit des eingehend en Datensatzes
DATA_ROW	VARCHAR (49152)	Die in Hexadezimalzeichen kodierten Daten in der

		ursprünglichen Zeile. Sie können diesen Hexwert mithilfe von Standardbibliotheken dekodieren oder dazu Webressourcen wie diesen Hex-in-Zeichenfolgen-Konverter verwenden.
PUMP_NAME	VARCHAR (128)	Die ursprüngliche Pumpe, wie mit `CREATE PUMP` definiert.

Automatisches Skalieren von Anwendungen zur Erhöhung des Durchsatzes

Amazon-Kinesis-Data-Analytics sorgt für die flexible Skalierung Ihrer Anwendung, um den Datendurchsatz Ihres Quell-Streams und Ihre Abfragekomplexität in den meisten Szenarien zu bewältigen. Amazon-Kinesis-Data-Analytics stellt Kapazität in Form von Kinesis Processing Units (KPU) zur Verfügung. Eine einzelne KPU bietet Ihnen den Arbeitsspeicher (4 GB) und zugehörige Datenverarbeitung und Netzwerke.

Das Standardlimit KPUs für Ihre Anwendung ist 64. Anweisungen zum Anfordern einer Erhöhung dieses Limits finden Sie unter So fordern Sie eine Erhöhung Ihres Limits an unter Amazon-Service-Limits.

Verwenden von Tagging

In diesem Abschnitt wird beschrieben, wie Sie Schlüssel-Wert-Metadaten-Tags zu Kinesis Data Analytics-Anwendungen hinzufügen. Diese Tags können für die folgenden Zwecke verwendet werden:

- Festlegung der Abrechnung für einzelne Kinesis-Data-Analytics-Anwendungen. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags im AWS Benutzerhandbuch Fakturierungs- und Kostenverwaltung.
- Steuern des Zugriffs auf Anwendungsressourcen basierend auf Tags. Weitere Informationen finden Sie unter Zugriffssteuerung mit Tags im Benutzerhandbuch.

- Benutzerdefinierte Zwecke. Sie können die Anwendungsfunktionalität basierend auf dem Vorhandensein von Benutzer-Tags definieren.

Bitte beachten Sie die folgenden Informationen über Tagging:

- Die maximale Anzahl an Anwendungs-Tags enthält System-Tags. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50.
- Wenn eine Aktion eine Tag-Liste beinhaltet, die doppelte `Key`-Werte enthält, löst der Service eine `InvalidArgumentException` aus.

Dieses Thema enthält die folgenden Abschnitte:

Hinzufügen von Tags, wenn eine Anwendung erstellt wird

Sie fügen Tags hinzu, wenn Sie eine Anwendung mithilfe des `tags` CreateApplicationAktionsparameters erstellen.

Das folgende Beispiel zeigt den `Tags`-Knoten für eine `CreateApplication`-Anforderung:

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

Hinzufügen oder Aktualisieren von Tags für eine vorhandene Anwendung

Mithilfe der TagResourceAktion fügen Sie einer Anwendung Tags hinzu. Mithilfe der UpdateApplicationAktion können Sie einer Anwendung keine Tags hinzufügen.

Fügen Sie zum Aktualisieren eines vorhandenen Tags ein Tag mit demselben Schlüssel wie das vorhandene Tag hinzu.

In der folgenden Beispiel-Anfrage für die Aktion`TagResource` werden neue Tags hinzugefügt oder vorhandene Tags aktualisiert:

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

Auflisten von Tags für eine Anwendung

Um vorhandene Tags aufzulisten, verwenden Sie die ListTagsForResourceAktion.

In der folgenden Beispiel-Anfrage für die Aktion `ListTagsForResource` werden Tags für eine Anwendung aufgelistet:

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

Entfernen von Tags aus einer Anwendung

Um Tags aus einer Anwendung zu entfernen, verwenden Sie die UntagResourceAktion.

Die folgende Beispiel-Anfrage für die Aktion `UntagResource` entfernt Tags aus einer Anwendung:

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

Erste Schritte mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen

Im Folgenden finden Sie Themen, die Ihnen beim Einstieg mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen helfen. Wenn Sie noch keine Erfahrung mit Kinesis-Data-Analytics-for-SQL-Anwendungen haben, empfehlen wir, sich mit den Konzepten und der Terminologie im Abschnitt Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's vertraut zu machen, bevor Sie die Schritte im Abschnitt „Erste Schritte“ ausführen.

Themen

Melde dich an für ein AWS-Konto

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

Um sich für eine anzumelden AWS-Konto

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

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

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

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

Erstellen eines Benutzers mit Administratorzugriff

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

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

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

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung -Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

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

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center -Benutzerhandbuch.
2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

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

Anmelden als Administratorbenutzer

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

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

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center - Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

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

Schritt 1: Einrichten eines -Kontos und Erstellen eines Administratorbenutzers

Bevor Sie Amazon-Kinesis-Data-Analytics zum ersten Mal verwenden, führen Sie die folgenden Aufgaben aus:

1. Melden Sie sich an für AWS
2. Erstellen eines IAM-Benutzers

Melden Sie sich an für AWS

Wenn Sie sich für Amazon Web Services registrieren, AWS-Konto ist Ihr Konto automatisch für alle Services angemeldet AWS, einschließlich Amazon Kinesis Data Analytics. Berechnet werden Ihnen aber nur die Services, die Sie nutzen.

Mit Kinesis Data Analytics zahlen Sie nur für die Ressourcen, die Sie wirklich nutzen. Wenn Sie ein neuer AWS Kunde sind, können Sie kostenlos mit Kinesis Data Analytics beginnen. Weitere Informationen finden Sie unter AWS – kostenloses Nutzungskontingent.

Wenn Sie bereits eine haben AWS-Konto, fahren Sie mit der nächsten Aufgabe fort. Wenn Sie noch keine haben AWS-Konto, führen Sie die Schritte im folgenden Verfahren aus, um eine zu erstellen.

Um ein zu erstellen AWS-Konto

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

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

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

Notieren Sie sich Ihre AWS-Konto ID, da Sie sie für die nächste Aufgabe benötigen.

Erstellen eines IAM-Benutzers

Dienste in AWS, wie Amazon Kinesis Data Analytics, erfordern, dass Sie beim Zugriff auf sie Anmeldeinformationen angeben, damit der Service feststellen kann, ob Sie über Berechtigungen für den Zugriff auf die Ressourcen verfügen, die diesem Service gehören. Für die Konsole müssen Sie Ihr Passwort eingeben. Sie können Zugriffsschlüssel erstellen, mit denen Sie AWS-Konto auf die AWS CLI OR-API zugreifen können. Wir empfehlen jedoch nicht, dass Sie AWS mit den Anmeldeinformationen für Ihren Zugriff darauf zugreifen AWS-Konto. Stattdessen empfehlen wir, AWS Identity and Access Management (IAM) zu verwenden. Erstellen Sie einen IAM-Benutzer und fügen Sie den Benutzer zu einer IAM-Gruppe mit Administrator-Berechtigungen hinzu. Anschließend gewähren Sie dem von Ihnen erstellten IAM-Benutzer administrative Berechtigungen. Sie können dann AWS mit einer speziellen URL und den Anmeldeinformationen dieses IAM-Benutzers darauf zugreifen.

Wenn Sie sich für registriert haben AWS, aber noch keinen IAM-Benutzer für sich selbst erstellt haben, können Sie mit der IAM-Konsole einen erstellen.

Für die Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie einen Benutzer namens (`adminuser`) mit Administratorrechten haben. Befolgen Sie die Schritte zum Einrichten des `adminuser` in Ihrem Konto.

Erstellen eines Administrator-Benutzers und Anmelden in der Konsole

1. Erstellen Sie einen Administratorbenutzer namens `adminuser` in Ihrem AWS-Konto. Weitere Anweisungen finden Sie unter Creating Your First IAM User and Administrators Group (Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe) im IAM User Guide (IAM-Benutzerhandbuch).
2. Ein Benutzer kann sich AWS-Managementkonsole mit einer speziellen URL bei der anmelden. Weitere Informationen finden Sie unter Wie sich Benutzer in Ihrem Konto anmelden im IAM-Benutzerhandbuch.

Weitere Informationen zu IAM finden Sie unter:

- AWS Identity and Access Management (ICH BIN)
- Erste Schritte mit IAM
- IAM Benutzerhandbuch

Nächster Schritt

Schritt 2: Richten Sie das AWS Command Line Interface ()AWS CLI ein

Melden Sie sich an für ein AWS-Konto

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

Um sich für eine anzumelden AWS-Konto

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

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

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

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

Erstellen eines Benutzers mit Administratorzugriff

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

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

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

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung -Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

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

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center -Benutzerhandbuch.
2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

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

Anmelden als Administratorbenutzer

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

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

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center - Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

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

Schritt 2: Richten Sie das AWS Command Line Interface ()AWS CLI ein

Folgen Sie den Schritten zum Herunterladen und Konfigurieren von AWS Command Line Interface (AWS CLI).

Important

Sie benötigen das nicht AWS CLI , um die Schritte in der Übung Erste Schritte auszuführen. Für einige der Übungen in diesem Handbuch wird die AWS CLI jedoch benötigt. Sie können

diesen Schritt überspringen und zu Schritt 3: Erstellen Ihrer Amazon-Kinesis-Data-Analytics-Startanwendung einem AWS CLI späteren Zeitpunkt weitergehen und ihn dann einrichten, wenn Sie ihn benötigen.

Um das einzurichten AWS CLI

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface -Benutzerhandbuch:

   - Erste Schritte mit dem AWS Command Line Interface
   - Konfiguration des AWS Command Line Interface

2. Fügen Sie der AWS CLI Konfigurationsdatei ein benanntes Profil für den Administratorbenutzer hinzu. Sie verwenden dieses Profil, wenn Sie die AWS CLI Befehle ausführen. Weitere Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

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

   Eine Liste der verfügbaren AWS-Regionen Regionen und Endpunkte finden Sie unter Regionen und Endpunkte in der Allgemeine Amazon Web Services-Referenz.

3. Überprüfen Sie die Einrichtung, indem Sie die folgenden Hilfebefehle in die Befehlszeile eingeben:

   ```
   aws help
   ```

Nächster Schritt

Schritt 3: Erstellen Ihrer Amazon-Kinesis-Data-Analytics-Startanwendung

Schritt 3: Erstellen Ihrer Amazon-Kinesis-Data-Analytics-Startanwendung

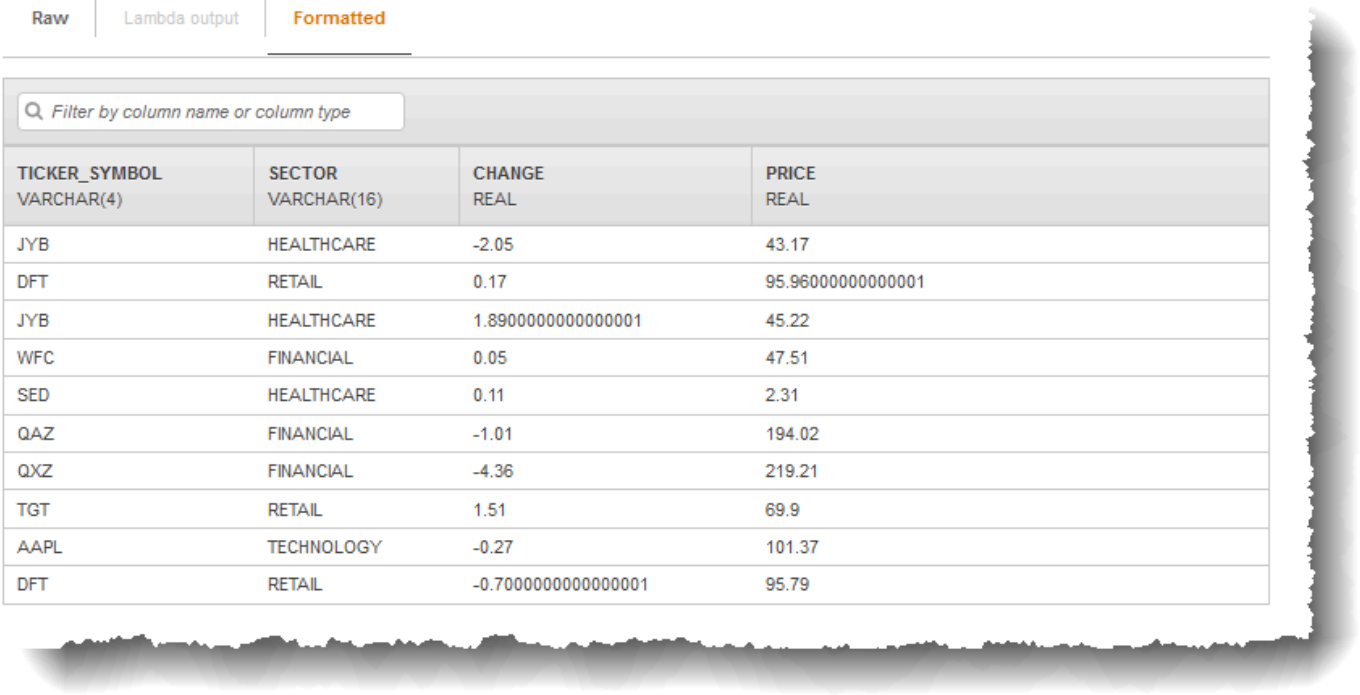
Wenn Sie die Schritte in diesem Abschnitt durchführen, können Sie Ihre erste Kinesis Data Analytics-Anwendung mithilfe der Konsole erstellen.

> **Note**
>
> Wir empfehlen, den Abschnitt Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's noch einmal zu lesen, bevor Sie die Einstiegsübung absolvieren.

Für diese Einstiegsübung können Sie die Konsole verwenden, um entweder mit dem Demo-Stream oder Vorlagen mit Anwendungs-Code zu arbeiten.

- Wenn Sie den Demo-Stream verwenden, erstellt die Konsole einen Kinesis-Datenstrom mit der Bezeichnung `kinesis-analytics-demo-stream` in Ihrem Konto.

  Eine Kinesis Data Analytics-Anwendung erfordert eine Streaming-Quelle. Für diese Quelle verwenden mehrere SQL-Beispiele in diesem Handbuch verwenden den Demo-Stream `kinesis-analytics-demo-stream`. Die Konsole führt darüber hinaus ein Skript aus, das dem Stream kontinuierlich Beispieldaten (simulierte Aktienhandelsdatensätze) hinzufügt, wie im Folgenden gezeigt.



In dieser Übung können Sie `kinesis-analytics-demo-stream` als Streaming-Quelle für Ihre Anwendung verwenden.

> **Note**
>
> Der Demo-Stream verbleibt in Ihrem Konto. Sie können ihn verwenden, um andere Beispiele in diesem Handbuch zu testen. Wenn Sie die Konsole jedoch verlassen, beendet das von der Konsole verwendete Skript das Füllen mit Daten. Die Konsole bietet die Option, bei Bedarf erneut mit dem Füllen des Streams zu beginnen.

- Wenn Sie die Vorlagen mit Beispiel-Anwendungscode verwenden, benutzen Sie den von der Konsole bereitgestellten Vorlagencode, um einfache Analysen auf dem Demo-Stream durchzuführen.

Sie können diese Funktionen verwenden, um Ihre erste Anwendung schnell folgendermaßen einzurichten:

1. Erstellen Sie eine Anwendung – Sie müssen nur einen Namen vergeben. Die Konsole erstellt die Anwendung und der Service setzt den Anwendungsstatus auf `READY`.

2. Konfigurieren von Eingaben – Zuerst fügen Sie eine Streaming-Quelle, den Demo-Stream, hinzu. Sie müssen einen Demo-Stream in der Konsole erstellen, bevor Sie diesen verwenden können. Die Konsole leitet dann anhand einer Stichprobe von Datensätzen im Demo-Stream ein Schema für den In-Application-Eingabe-Stream ab, der erstellt wird. Die Konsole benennt den In-Application-Stream mit `SOURCE_SQL_STREAM_001`.

   Die Konsole verwendet zum Ableiten des Schemas die Erkennungs-API. Falls erforderlich können Sie das abgeleitete Schema bearbeiten. Weitere Informationen finden Sie unter DiscoverInputSchema. Kinesis Data Analytics verwendet dieses Schema zum Erstellen eines In-Application-Streams.

   Wenn Sie die Anwendung starten, liest Kinesis Data Analytics den Demo-Stream kontinuierlich in Ihrem Auftrag und fügt Zeilen im In-Application-Eingabe-Stream `SOURCE_SQL_STREAM_001` ein.

3. Angeben des Anwendungscodes – Sie verwenden eine Vorlage (mit der Bezeichnung Continuous filter), die den folgenden Code bereitstellt:

   ```
   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%';
   ```

   Der Anwendungscode fragt den In-Application-Stream `SOURCE_SQL_STREAM_001` ab. Der Code fügt dann mittels Pumps die resultierenden Zeilen in einen anderen In-Application-Stream `DESTINATION_SQL_STREAM` ein. Weitere Informationen zu diesem Codierungsmuster finden Sie unter Anwendungscode.

   Weitere Informationen zu den SQL-Sprachelementen, die von Kinesis Data Analytics unterstützt werden, finden Sie in der SQL-Referenz zu Amazon-Kinesis-Data-Analytics.

4. Konfigurieren der Ausgabe – In dieser Übung konfigurieren Sie keine Ausgabe. Das bedeutet, dass Sie keine Daten im In-Application-Stream ablegen, die Ihre Anwendung für ein externes Ziel erstellt. Stattdessen überprüfen Sie Abfrageergebnisse in der Konsole. Weitere Beispiele in diesem Handbuch zeigen, wie die Ausgabe konfiguriert wird. Ein Beispiel finden Sie unter Beispiel: Erstellen einfacher Warnungen.

> **Important**
>
> In dieser Übung wird die Region USA Ost (Nord-Virginia) (us-east-1) zum Einrichten der Anwendung verwendet. Sie können jedes der unterstützten AWS-Regionen verwenden.

Nächster Schritt

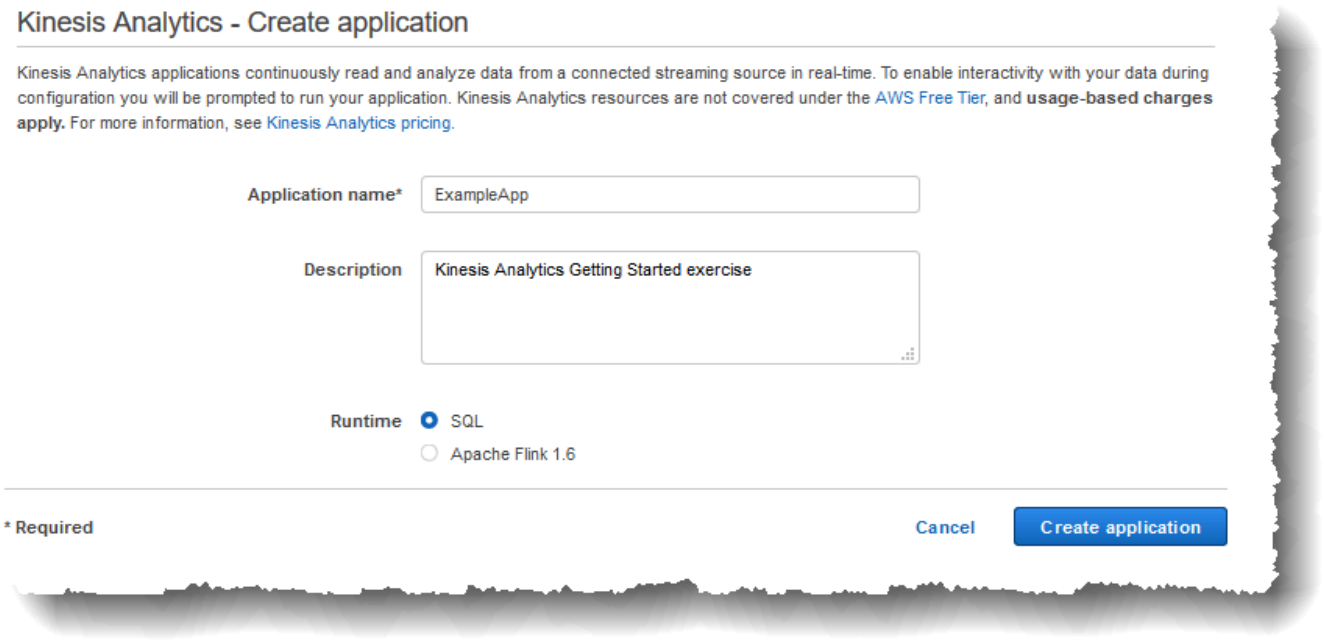
Schritt 3.1: Eine Anwendung erstellen

Schritt 3.1: Eine Anwendung erstellen

In diesem Abschnitt erstellen Sie eine Amazon-Kinesis-Data-Analytics-Anwendung. Sie konfigurieren die Anwendungseingabe im nächsten Schritt.

So erstellen Sie eine Datenanalyseanwendung

1. Melden Sie sich bei der Managed Service for Apache Flink-Konsole unter https://console.aws.amazon.com/kinesisanalytics an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie Create application aus.
3. Geben Sie auf der Seite Anwendung erstellen einen Anwendungsnamen und eine Beschreibung ein, wählen Sie SQL für die Laufzeit-Einstellung der Anwendung und klicken Sie dann auf Anwendung erstellen.



Dadurch wird eine Kinesis Data Analytics-Anwendung mit dem Status BEREIT erstellt. Die Konsole zeigt das Anwendungs-Hub, über das Sie Eingabe und Ausgabe konfigurieren können.

> **Note**
>
> Zum Erstellen einer Anwendung benötigt die CreateApplication-Operation nur den Namen der Anwendung. Sie können Eingabe- und Ausgabekonfiguration nach dem Erstellen einer Anwendung in der Konsole hinzufügen.

Im nächsten Schritt konfigurieren Sie die Eingabe für die Anwendung. In der Eingabekonfiguration fügen Sie eine Streaming-Datenquelle zur Anwendung hinzu und finden ein Schema für einen In-Application-Eingabe-Stream, indem Sie Daten auf der Streaming-Quelle abfragen.

Nächster Schritt

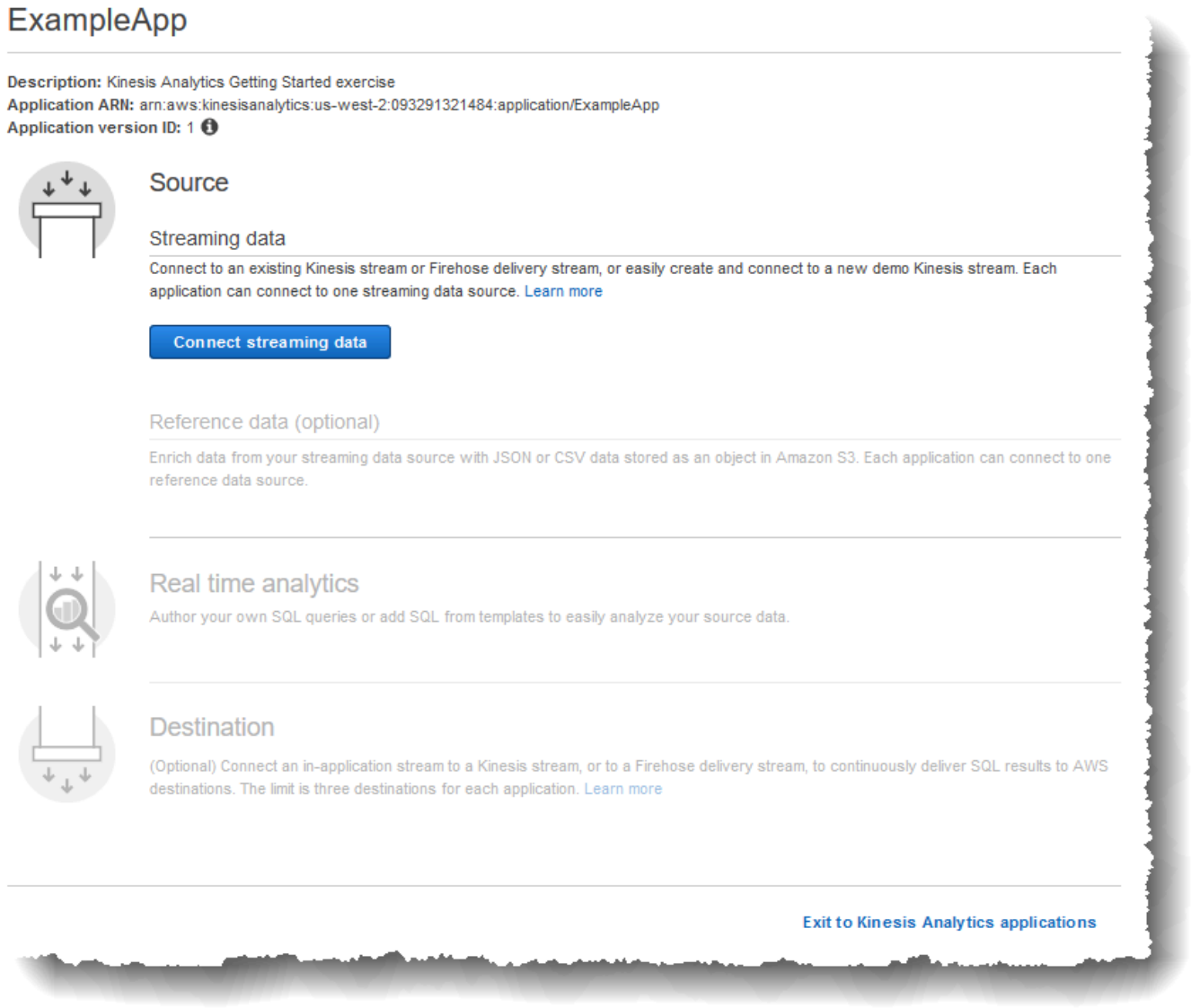
Schritt 3.2: Konfigurieren der Eingabe

Schritt 3.2: Konfigurieren der Eingabe

Ihre Anwendung benötigt eine Streaming-Quelle. Um Ihnen den Einstieg zu erleichtern, kann die Konsole einen Demo-Stream erstellen (als `kinesis-analytics-demo-stream` bezeichnet). Darüber hinaus führt die Konsole ein Skript aus, durch das der Stream mit Datensätzen gefüllt wird.

So fügen Sie eine Streaming-Quelle zu Ihrer Anwendung hinzu

1. Wählen Sie auf der Anwendungs-Hub-Seite in der Konsole Connect streaming data (Streaming-Daten verbinden) aus.



2. Lesen Sie auf der angezeigten Seite Folgendes:

- Abschnitt Source, in dem Sie eine Streaming-Quelle für Ihre Anwendung festlegen können. Sie können eine vorhandene Streaming-Quelle auswählen oder eine neue erstellen. In dieser Übung erstellen Sie einen neuen Stream, den Demo-Stream.

  Standardmäßig benennt die Konsole den erstellten In-Application-Eingabe-Stream mit `INPUT_SQL_STREAM_001`. Lassen Sie den Namen für diese Übung so, wie er angegeben wird.

  - Stream-Referenzname – Diese Option zeigt den Namen des In-Application-Eingabe-Streams an, der erstellt wird, `SOURCE_SQL_STREAM_001`. Sie können den Namen ändern. Behalten Sie für diese Übung jedoch diesen Namen bei.

    In der Eingabekonfiguration ordnen Sie den Demo-Stream einem In-Application-Eingabe-Stream zu, der erstellt wird. Wenn Sie die Anwendung starten, liest Amazon-Kinesis-Data-Analytics den Demo-Stream kontinuierlich und fügt Zeilen im In-Application-Eingabe-Stream ein. Sie fragen diesen In-Application-Eingabe-Stream in Ihrem Anwendungscode ab.

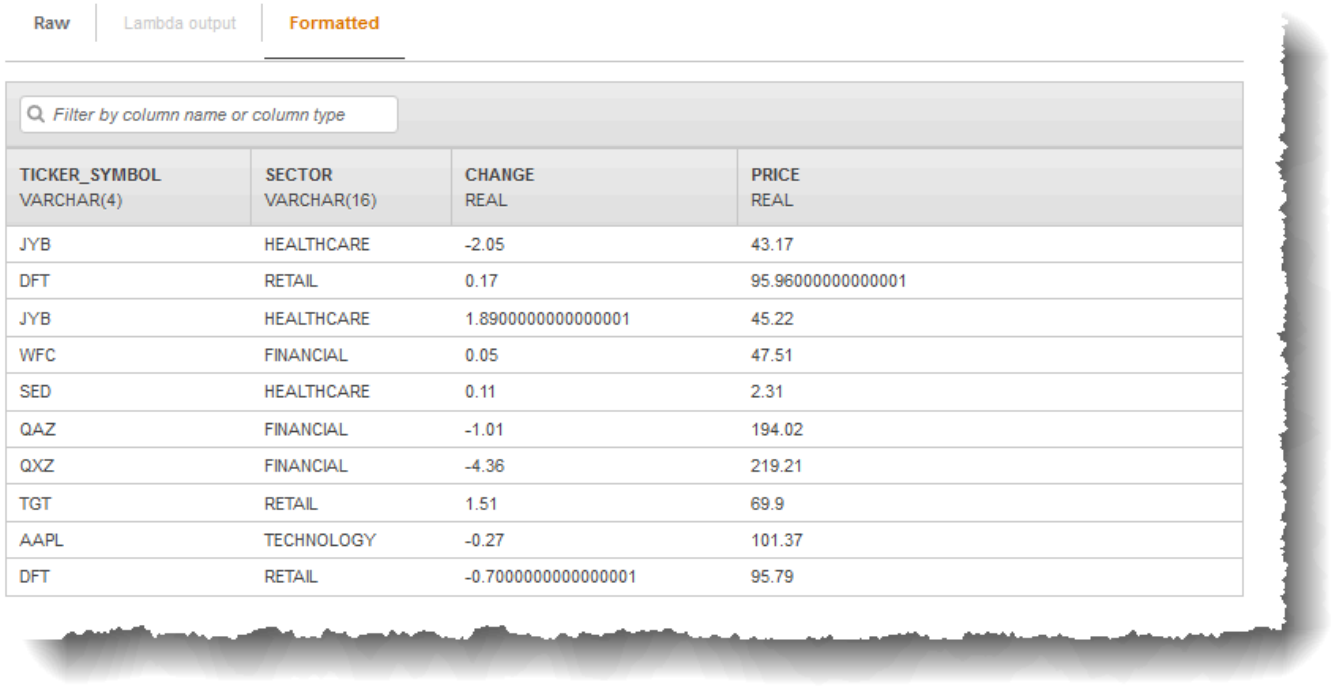
  - Vorverarbeitung von Datensätzen mit AWS Lambda: Mit dieser Option geben Sie einen AWS Lambda Ausdruck an, der die Datensätze im Eingabestream ändert, bevor Ihr Anwendungscode ausgeführt wird. In dieser Übung lassen Sie die Option Disabled ausgewählt. Weitere Informationen zur Lambda-Vorverarbeitung finden Sie unter Vorverarbeitung von Daten mithilfe einer Lambda-Funktion.

  Nachdem Sie alle Informationen auf dieser Seite bereitgestellt haben, sendet die Konsole eine Aktualisierungsanforderung (siehe UpdateApplication), um die Eingabekonfiguration zur Anwendung hinzuzufügen.

3. Wählen Sie auf der Seite Source die Option Configure a new stream aus.
4. Wählen Sie Create demo stream aus. In der Konsole werden die folgenden Schritte zum Konfigurieren der Anwendungseingabe ausgeführt:

- Die Konsole erstellt einen Kinesis-Datenstrom mit dem Namen `kinesis-analytics-demo-stream`.
- Die Konsole füllt den Stream mit Börsenticker-Beispieldaten.
- Durch die Eingabe DiscoverInputSchema leitet die Konsole ein Schema ab, indem Beispieldatensätze im Stream gelesen werden. Das abgeleitete Schema ist das Schema für den In-Application-Eingabe-Stream, der erstellt wird. Weitere Informationen finden Sie unter Konfigurieren der Anwendungseingabe.
- Die Konsole zeigt das abgeleitete Schema und die aus der Streaming-Quelle zum Ableiten des Schemas gelesenen Beispieldaten an.

Die Konsole zeigt die Beispieldatensätze auf der Streaming-Quelle an.



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

Folgendes wird auf der Konsolenseite Stream sample angezeigt:

- Die Registerkarte Raw stream sample zeigt die Stream-Rohdatensätze, die von der API-Aktion DiscoverInputSchema zum Ableiten des Schemas abgerufen wurden.
- Die Registerkarte Formatted stream sample zeigt die tabellarische Version der Daten in der Registerkarte Raw stream sample.

- Wenn Sie Edit schema auswählen, können Sie das abgeleitete Schema bearbeiten. Ändern Sie das abgeleitete Schema für diese Übung nicht. Weitere Informationen zum Bearbeiten eines Schemas finden Sie unter Arbeiten mit dem Schema-Editor.

  Wenn Sie Rediscover schema auswählen, führt die Konsole den Befehl DiscoverInputSchema erneut aus und leitet das Schema ab.

5. Wählen Sie Save and continue aus.

   Sie haben jetzt eine Anwendung, der eine Eingabekonfiguration hinzugefügt wurde. Im nächsten Schritt fügen Sie SQL-Code hinzu, um Analysen am In-Application-Eingabe-Stream der Daten durchzuführen.

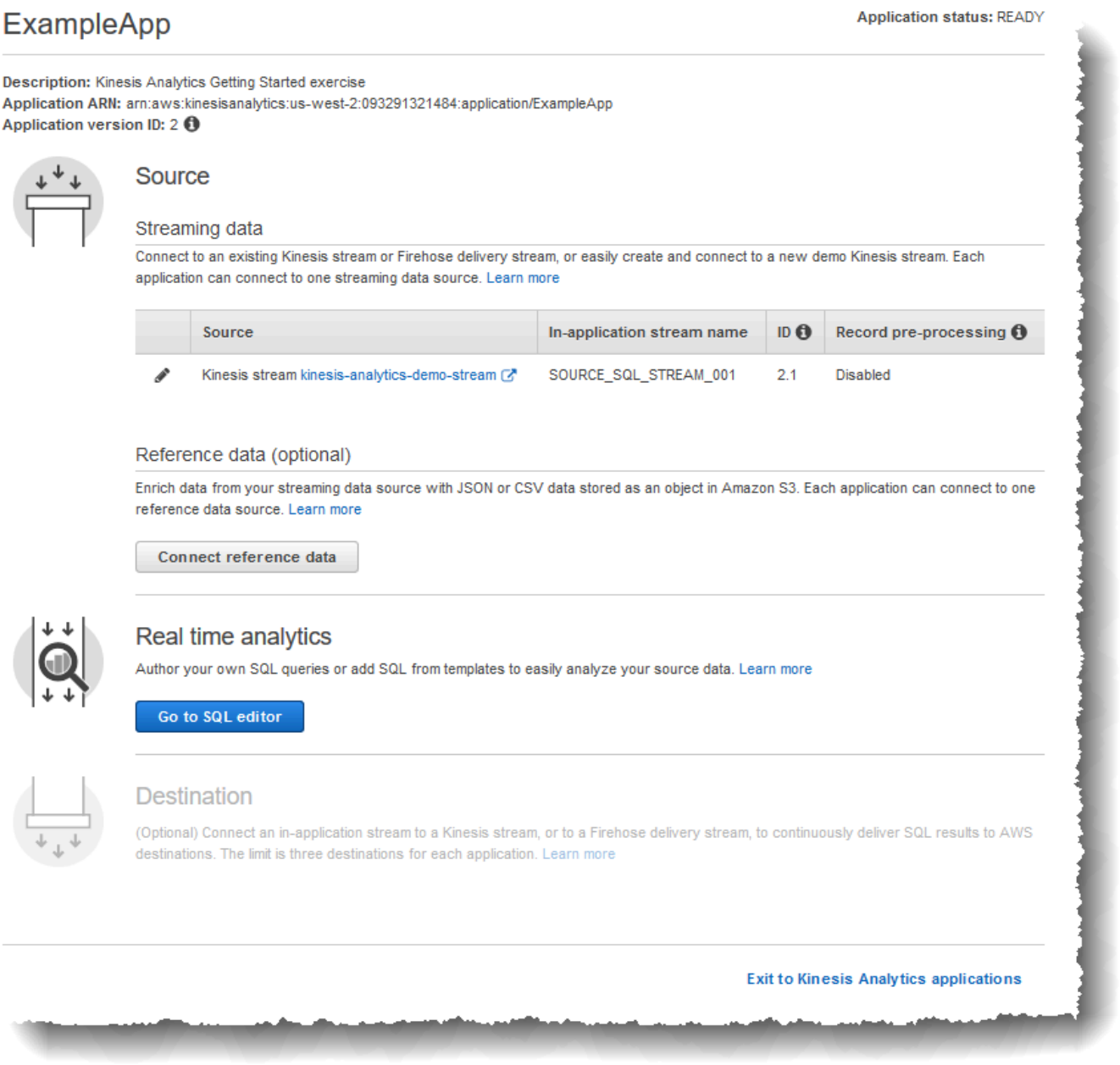
Nächster Schritt

Schritt 3.3: Hinzufügen von Echtzeit-Analysen (Hinzufügen von Anwendungscode)

Schritt 3.3: Hinzufügen von Echtzeit-Analysen (Hinzufügen von Anwendungscode)

Sie können Ihre eigenen SQL-Abfragen für den In-Application-Stream schreiben. Für den folgenden Schritt verwenden Sie jedoch eine der Vorlagen, die Beispielcode bereitstellt.

1. Wählen Sie auf der Anwendungs-Hub-Seite Go to SQL editor aus.



2. Im Feld Möchten Sie mit dem Ausführen von "" beginnen? ExampleApp Wählen Sie im Dialogfeld Ja, Anwendung starten aus.

   Die Konsole sendet eine Anforderung zum Starten der Anwendung (siehe StartApplication) und anschließend wird die SQL-Editor-Seite angezeigt.

3. Die Konsole öffnet die SQL-Editor-Seite. Überprüfen Sie die Seite einschließlich der Schaltflächen (Add SQL from templates, Save and run SQL) und verschiedenen Registerkarten.
4. Wählen Sie im SQL-Editor Add SQL from templates aus.

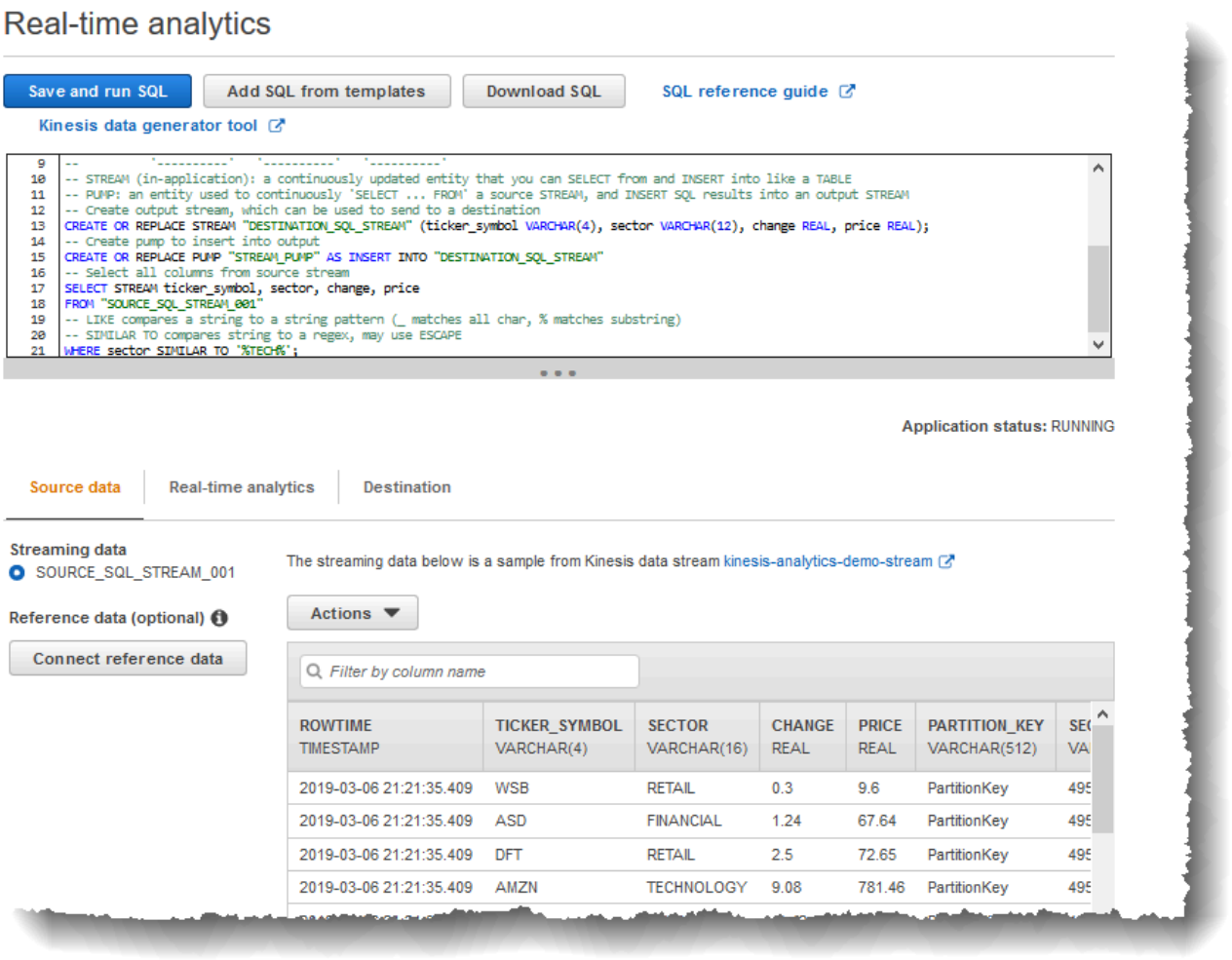
5. Wählen Sie in der Liste der verfügbaren Vorlagen Continuous filter aus. Der Beispiel-Code liest Daten aus einem In-Application-Stream (über die `WHERE`-Klausel werden die Zeilen gefiltert) und fügt sie wie folgt in einen anderen In-Application-Stream ein:

   - Der In-Application-Stream `DESTINATION_SQL_STREAM` wird erstellt.
   - Es wird das Pump `STREAM_PUMP`erstellt, diese wird zur Auswahl von Zeilen aus `SOURCE_SQL_STREAM_001` verwendet, die dann in `DESTINATION_SQL_STREAM` eingefügt werden.

6. Wählen Sie Add this SQL to editor aus.
7. Testen Sie den Anwendungscode wie folgt:

   Denken Sie daran, dass Sie die Anwendung bereits gestartet haben (Status RUNNING). Daher ist Amazon-Kinesis-Data-Analytics bereits kontinuierlich dabei, Daten aus der Streaming-Quelle zu lesen und Zeilen zum In-Application-Stream `SOURCE_SQL_STREAM_001` hinzuzufügen.

   a. Klicken Sie im SQL-Editor auf Save and run SQL. Die Konsole sendet zunächst eine Aktualisierungsanforderung, um den Anwendungscode zu speichern. Anschließend wird der Code kontinuierlich ausgeführt.
   b. Die Ergebnisse können Sie auf der Registerkarte Real-time analytics sehen.



Der SQL-Editor umfasst folgende Registerkarten:

- Auf der Registerkarte Source data wird ein In-Application-Eingabe-Stream angezeigt, welcher der Streaming-Quelle zugeordnet ist. Wenn Sie den In-Application-Stream auswählen, können Sie sehen, dass Daten ankommen. Beachten Sie die zusätzlichen Spalten im In-Application-Eingabe-Stream, die nicht in den Eingabekonfiguration angegeben wurden. Hierzu gehören die folgenden Zeitstempelspalten:

  - ROWTIME – In jeder Zeile in einem In-Application-Stream gibt es eine spezielle Spalte mit der Bezeichnung `ROWTIME`. Diese Spalte ist der Zeitstempel, also der Zeitpunkt, an dem Amazon-Kinesis-Data-Analytics die Zeile in den ersten In-Application-Stream (den In-Application-Eingabe-Stream, welcher der Streaming-Quelle zugeordnet ist) eingefügt hat.

- Approximate_Arrival_Time – Jeder Kinesis Data Analytics-Datensatz enthält einen Wert mit der Bezeichnung `Approximate_Arrival_Time`. Dieser wird ist der Zeitstempel für das ungefähre Eintreffen, der gesetzt wird, wenn die Streaming-Quelle den entsprechenden Datensatz erfolgreich erhält und speichert. Wenn Kinesis Data Analytics Datensätze aus einer Streaming-Quelle liest, wird diese Spalte in den In-Application-Eingabe-Stream abgerufen.

  Diese Zeitstempelwerte sind in Abfragen mit Zeitfenstern hilfreich. Weitere Informationen finden Sie unter Abfragen mit Fenstern.

- Auf der Registerkarte Real-time analytics werden alle anderen In-Application-Streams angezeigt, die von Ihrem Anwendungscode erstellt wurden. Auch der Fehler-Stream ist hier enthalten. Kinesis Data Analytics sendet alle Zeilen, die nicht verarbeitet werden können, in den Fehler-Stream. Weitere Informationen finden Sie unter Fehlerbehandlung.

  Wählen Sie `DESTINATION_SQL_STREAM` aus, um die von Ihrem Anwendungscode eingefügten Zeilen anzuzeigen. Beachten Sie auch hier die zusätzlichen Spalten, die nicht von Ihrem Anwendungscode erstellt wurden. Diese Spalten beinhalten die Zeitstempelspalte `ROWTIME`. Kinesis Data Analytics kopiert diese Werte einfach aus der Quelle (`SOURCE_SQL_STREAM_001`).

- Auf der Registerkarte Ziel wird das externe Ziel angezeigt, in das Kinesis Data Analytics die Abfrageergebnisse schreibt. Sie haben noch kein externes Ziel für Ihre Anwendungsausgabe konfiguriert.

Nächster Schritt

Schritt 3.4: (Optional) Aktualisieren des Anwendungscodes

Schritt 3.4: (Optional) Aktualisieren des Anwendungscodes

In diesem Schritt erfahren Sie, wie Sie den Anwendungscode aktualisieren.

So aktualisieren Sie den Anwendungscode

1. Erstellen Sie wie folgt einen anderen In-Application-Stream:

   - Erstellen Sie einen anderen In-Application-Stream mit der Bezeichnung `DESTINATION_SQL_STREAM_2`.
   - Erstellen Sie eine Pump und verwenden Sie diese anschließend, um durch Auswahl von Zeilen aus `DESTINATION_SQL_STREAM` Zeilen in den neu erstellten Stream einzufügen.

   Fügen Sie im SQL-Editor folgenden Code an den vorhandenen Anwendungscode an:

   ```
   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";
   ```

   Speichern Sie den Code und führen Sie ihn aus. Zusätzliche In-Application-Streams werden auf der Registerkarte Real-time analytics angezeigt.

2. Erstellen Sie zwei In-Application-Streams. Filtern Sie Zeilen in `SOURCE_SQL_STREAM_001` basierend auf dem Börsenticker und fügen Sie sie in diese separaten Streams ein.

   Fügen Sie Ihrem Anwendungscode die folgenden SQL-Anweisungen an:

   ```
   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%';
```

Speichern Sie den Code und führen Sie ihn aus. Beachten Sie die zusätzlichen In-Application-Streams auf der Registerkarte Real-time analytics.

Sie haben nun Ihre erste funktionierende Amazon-Kinesis-Data-Analytics-Anwendung. In dieser Übung haben Sie folgende Aufgaben ausgeführt:

- Sie haben Ihre erste Kinesis Data Analytics-Anwendung erstellt.

- Sie haben die Anwendungseingabe konfiguriert, die den Demo-Stream als Streaming-Quelle identifiziert und einem erstellten In-Application-Stream (`SOURCE_SQL_STREAM_001`) zugewiesen hat. Kinesis Data Analytics liest den Demo-Stream kontinuierlich und fügt Datensätze in den In-Application-Stream ein.

- Ihr Anwendungscode hat `SOURCE_SQL_STREAM_001` abgefragt und die Ausgabe in einen anderen In-Application-Stream mit der Bezeichnung `DESTINATION_SQL_STREAM` geschrieben.

Jetzt können Sie optional die Anwendungsausgabe konfigurieren, um die Ausgabe in ein externes Ziel zu schreiben. Sie können also die Anwendungsausgabe so konfigurieren, dass Datensätze in `DESTINATION_SQL_STREAM` in ein externes Ziel geschrieben werden. In dieser Übung ist dies ein optionaler Schritt. Um zu erfahren, wie Sie das Ziel konfigurieren, gehen Sie weiter zum nächsten Schritt.

Nächster Schritt

Schritt 4 (optional): Bearbeiten des Schema- und SQL-Codes mithilfe der Konsole.

Schritt 4 (optional): Bearbeiten des Schema- und SQL-Codes mithilfe der Konsole

Im Folgenden finden Sie Informationen dazu, wie Sie ein abgeleitetes Schema und SQL-Code für Amazon-Kinesis-Data-Analytics. bearbeiten. Arbeiten Sie zu diesem Zweck mit dem Schema-Editor und dem SQL-Editor, die Bestandteil der Kinesis Data Analytics-Konsole sind.

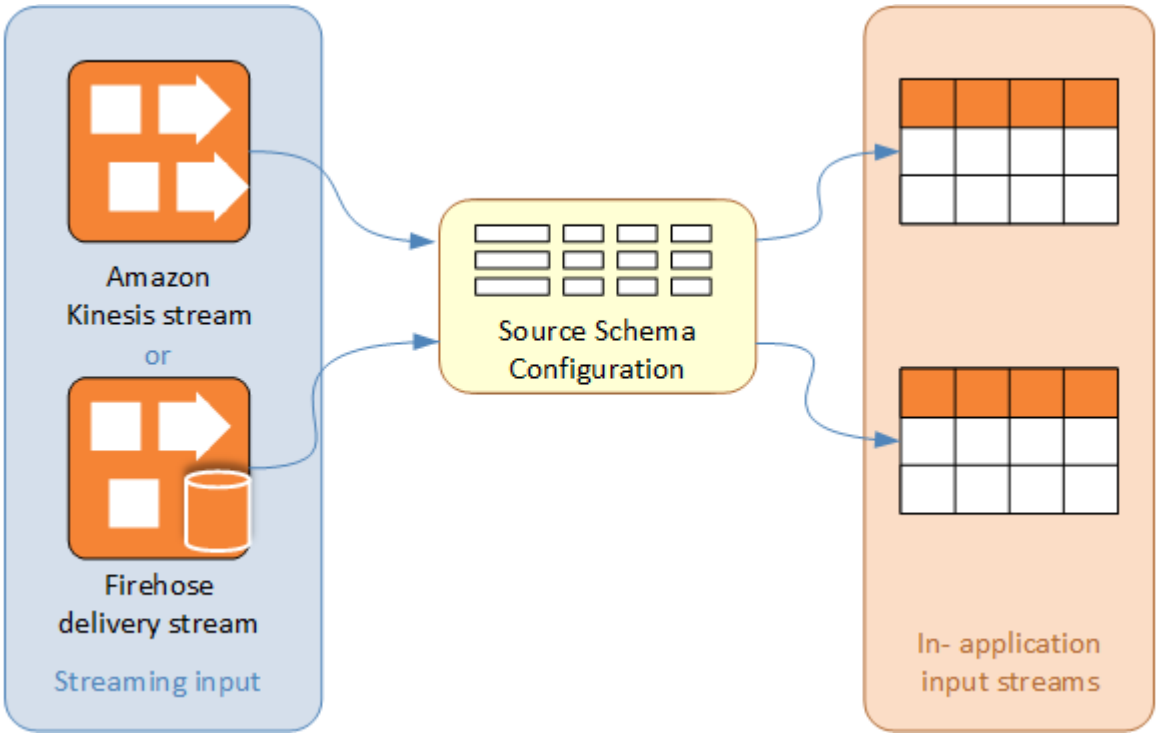
> **Note**
>
> Für den Zugriff auf und das Sampling von Daten in der Konsole muss die Anmelderolle des Benutzers über die Berechtigung `kinesisanalytics:GetApplicationState` verfügen. Weitere Informationen zu Kinesis Data Analytics-Anwendungsberechtigungen finden Sie unter Übersicht über die Verwaltung von Zugriffsberechtigungen.

Themen

- Arbeiten mit dem Schema-Editor
- Arbeiten mit dem SQL-Editor

Arbeiten mit dem Schema-Editor

Das Schema für den Eingabe-Stream einer Amazon-Kinesis-Data-Analytics-Anwendung definiert, wie Daten aus dem Stream für SQL-Abfragen in der Anwendung zur Verfügung gestellt werden.



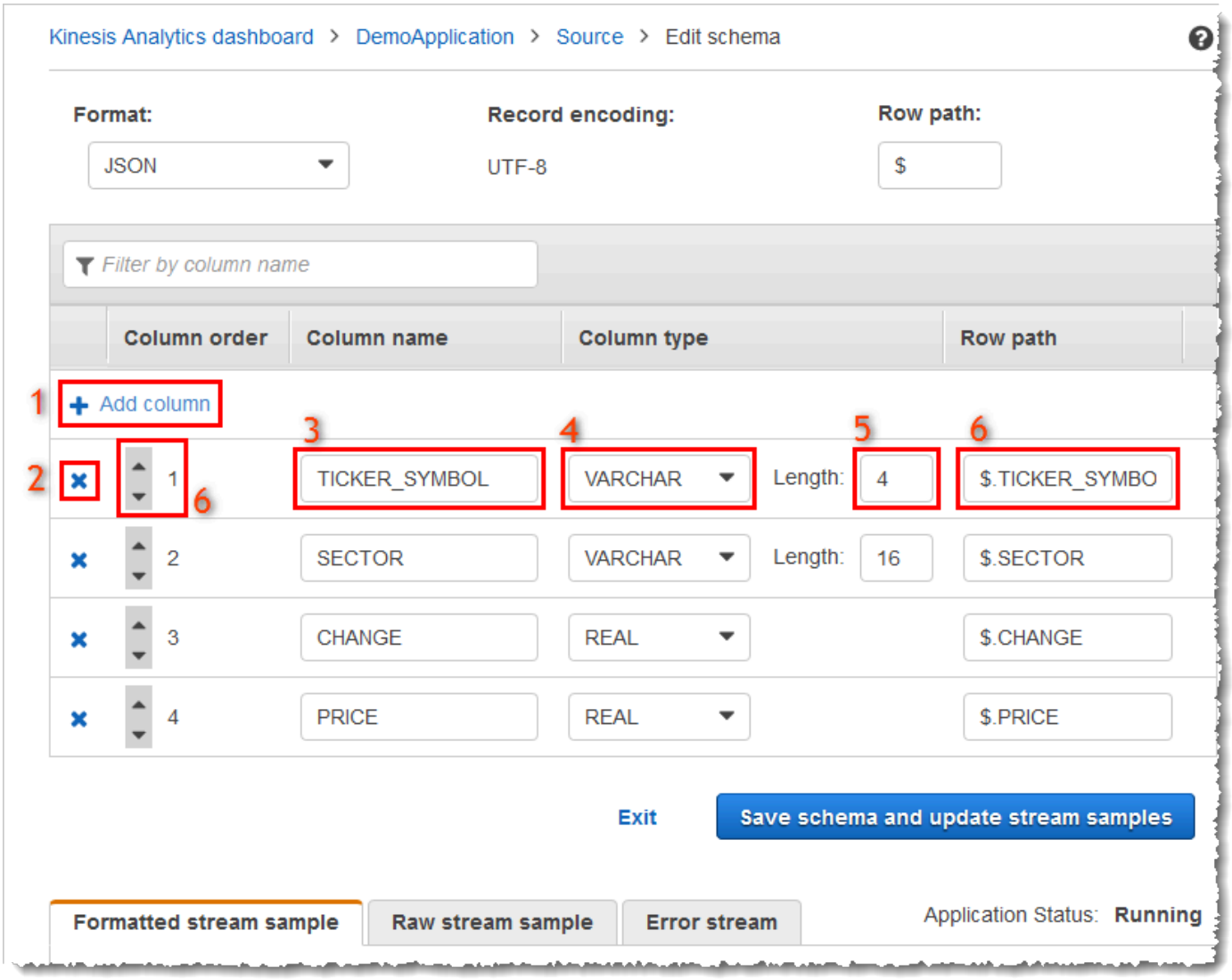
Das Schema enthält Auswahlkriterien, mit denen Sie festlegen können, welcher Teil der Streaming-Eingabe im In-Application-Eingabe-Stream in eine Datenspalte umgewandelt wird. Bei dieser Eingabe kann es sich Folgendes handeln:

- Ein JSONPath Ausdruck für JSON-Eingabestreams. JSONPath ist ein Tool zum Abfragen von JSON-Daten.
- Eine Spaltennummer für Eingabe-Streams im CSV-Format (durch Komma getrennte Werte).
- Ein Spaltenname und ein SQL-Datentyp für die Darstellung der Daten im In-Application-Daten-Stream. Der Datentyp enthält auch eine Länge für Zeichen oder Binärdaten.

Die Konsole versucht, das Schema mittels DiscoverInputSchema zu generieren. Wenn eine Schemaerkennung fehlschlägt oder ein falsches oder unvollständiges Schema zurückgegeben wird, müssen Sie das Schema mit dem Schema-Editor manuell bearbeiten.

Hauptbildschirm des Schema-Editors

Der folgende Screenshot zeigt den Hauptbildschirm des Schema-Editors.



Sie können die folgenden Bearbeitungen auf das Schema anwenden:

- Hinzufügen einer Spalte (1): Möglicherweise müssen Sie eine Datenspalte hinzufügen, wenn ein Datenelement nicht automatisch erkannt wird.
- Löschen einer Spalte (2): Sie können Daten aus dem Quell-Stream ausschließen, wenn Ihre Anwendung diese nicht benötigt. Dieser Ausschluss wirkt sich nicht auf die Daten im Quell-Stream aus. Wenn Daten ausgeschlossen werden, werden sie der Anwendung einfach nicht zur Verfügung gestellt.
- Umbenennen einer Spalte (3): Ein Spaltenname darf nicht leer sein, muss mehr als ein einzelnes Zeichen enthalten und darf keine reservierten SQL-Schlüsselwörter enthalten. Der Name muss

darüber hinaus Namenskriterien für gewöhnliche SQL-Kennungen entsprechen: Er muss mit einem Buchstaben beginnen und darf nur Buchstaben, Unterstriche und Ziffern enthalten.

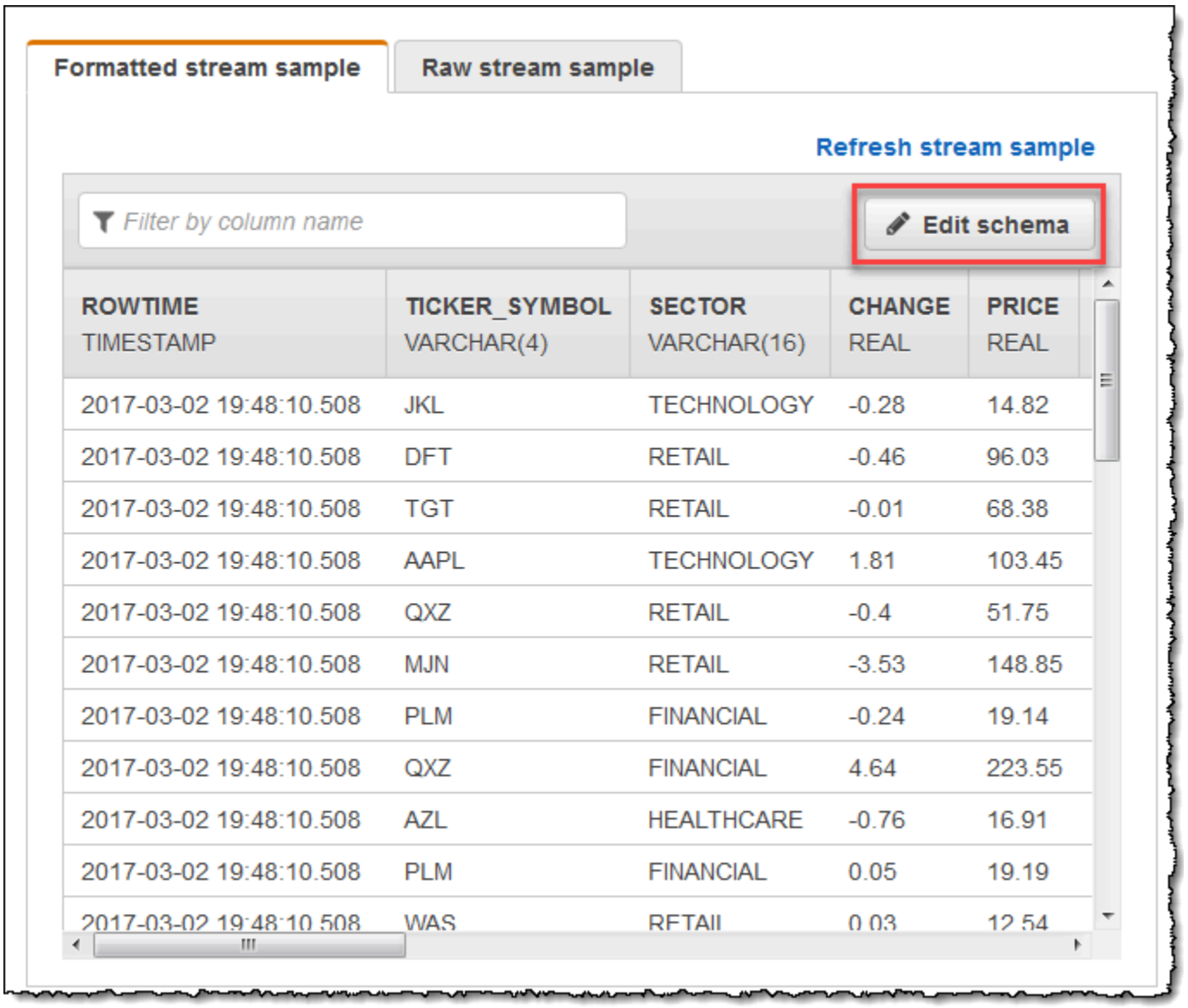
- Ändern des Datentyps (4) oder der Länge (5) einer Spalte: Sie können einen kompatiblen Datentyp für eine Spalte angeben. Wenn Sie einen nicht kompatiblen Datentyp angeben, wird die Spalte entweder mit NULL aufgefüllt oder der In-Application-Stream wird überhaupt nicht aufgefüllt. In letzteren Fall werden Fehler in den Fehler-Stream geschrieben. Wenn Sie eine Länge für eine Spalte angeben, die zu klein ist, werden die eingehenden Daten abgeschnitten.
- Ändern Sie die Auswahlkriterien einer Spalte (6): Sie können den JSONPath Ausdruck oder die CSV-Spaltenreihenfolge bearbeiten, anhand derer die Quelle der Daten in einer Spalte bestimmt wird. Um die Auswahlkriterien für ein JSON-Schema zu ändern, geben Sie einen neuen Wert für den Zeilenpfadausdruck ein. Ein CSV-Schema verwendet die Spaltenreihenfolge als Auswahlkriterium. Um die Auswahlkriterien für ein CSV-Schema zu ändern, ändern Sie die Reihenfolge der Spalten.

Bearbeiten des Schemas für eine Streaming-Quelle

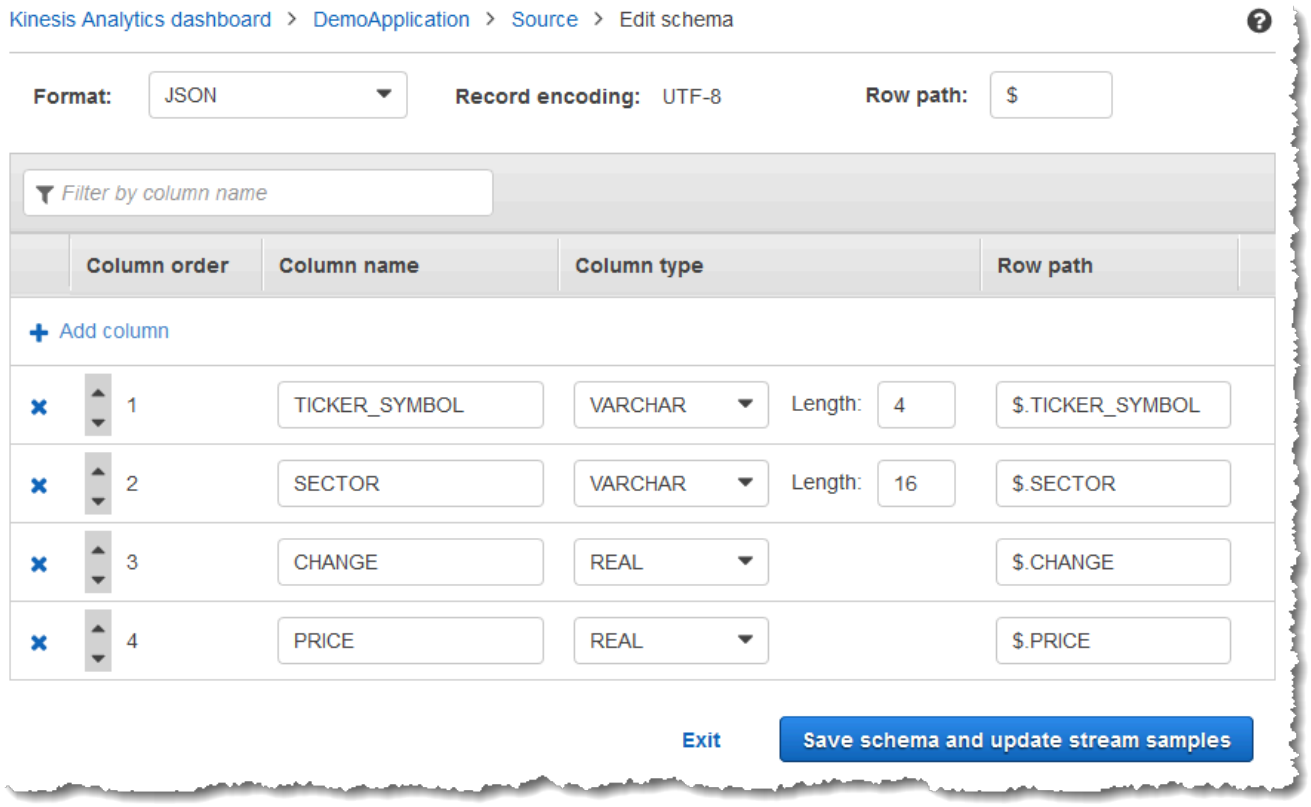
Wenn Sie ein Schema für eine Streaming-Quelle bearbeiten müssen, führen Sie die folgenden Schritte aus.

So bearbeiten Sie das Schema für eine Streaming-Quelle

1. Wählen Sie auf der Seite Source die Option Edit schema.



2. Bearbeiten Sie auf der Seite Edit schema das Quell-Schema.



3. Wählen Sie für Format JSON oder CSV. Für das JSON- oder CSV-Format wird die Kodierung ISO-8859-1 unterstützt.

Weitere Informationen zum Bearbeiten des Schemas für das JSON- oder CSV-Format finden Sie in den Verfahren in den nächsten Abschnitten.

Bearbeiten eines JSON-Schemas

Sie können JSON-Schemata bearbeiten, indem Sie die folgenden Schritte ausführen.

So bearbeiten Sie ein JSON-Schema

1. Wählen Sie im Schema-Editor Add column, um eine Spalte hinzuzufügen.

   In der ersten Spaltenposition wird eine neue Spalte angezeigt. Zum Ändern der Spaltenreihenfolge wählen Sie die nach oben und unten zeigenden Pfeile neben dem Spaltennamen.

   Geben Sie für neue Spalte folgende Informationen an:

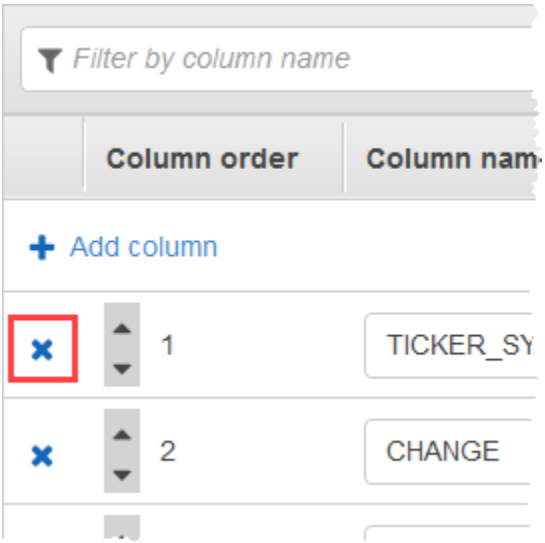
- Geben Sie in Column name einen Namen ein.

  Ein Spaltenname darf nicht leer sein, muss mehr als ein einzelnes Zeichen enthalten und darf keine reservierten SQL-Schlüsselwörter enthalten. Dieser muss ebenfalls den Namenskriterien für gewöhnliche SQL-Kennungen entsprechen: Er muss mit einem Buchstaben beginnen und darf nur Buchstaben, Unterstriche und Ziffern enthalten.
- Geben Sie für Column Type einen SQL-Datentyp ein.

  Jeder unterstützte SQL-Datentyp kann ein Spaltentyp sein. Wenn es sich beim neuen Datentyp um CHAR, VARBINARY oder VARCHAR handelt, geben Sie für Length eine Datenlänge an. Weitere Informationen finden Sie unter Data Types.
- Geben Sie für Row path einen Zeilenpfad an. Ein Zeilenpfad ist ein gültiger JSONPath Ausdruck, der einem JSON-Element zugeordnet ist.

  > **Note**
  >
  > Der Basiswert für Row path ist der Pfad zum übergeordneten Element auf höchster Ebene, das die zu importierenden Daten enthält. Dieser Wert ist standardmäßig $. Weitere Informationen finden Sie unter `RecordRowPath` in `JSONMappingParameters`.

2. Um eine Spalte zu löschen, wählen Sie das Symbol x neben der Spaltennummer.

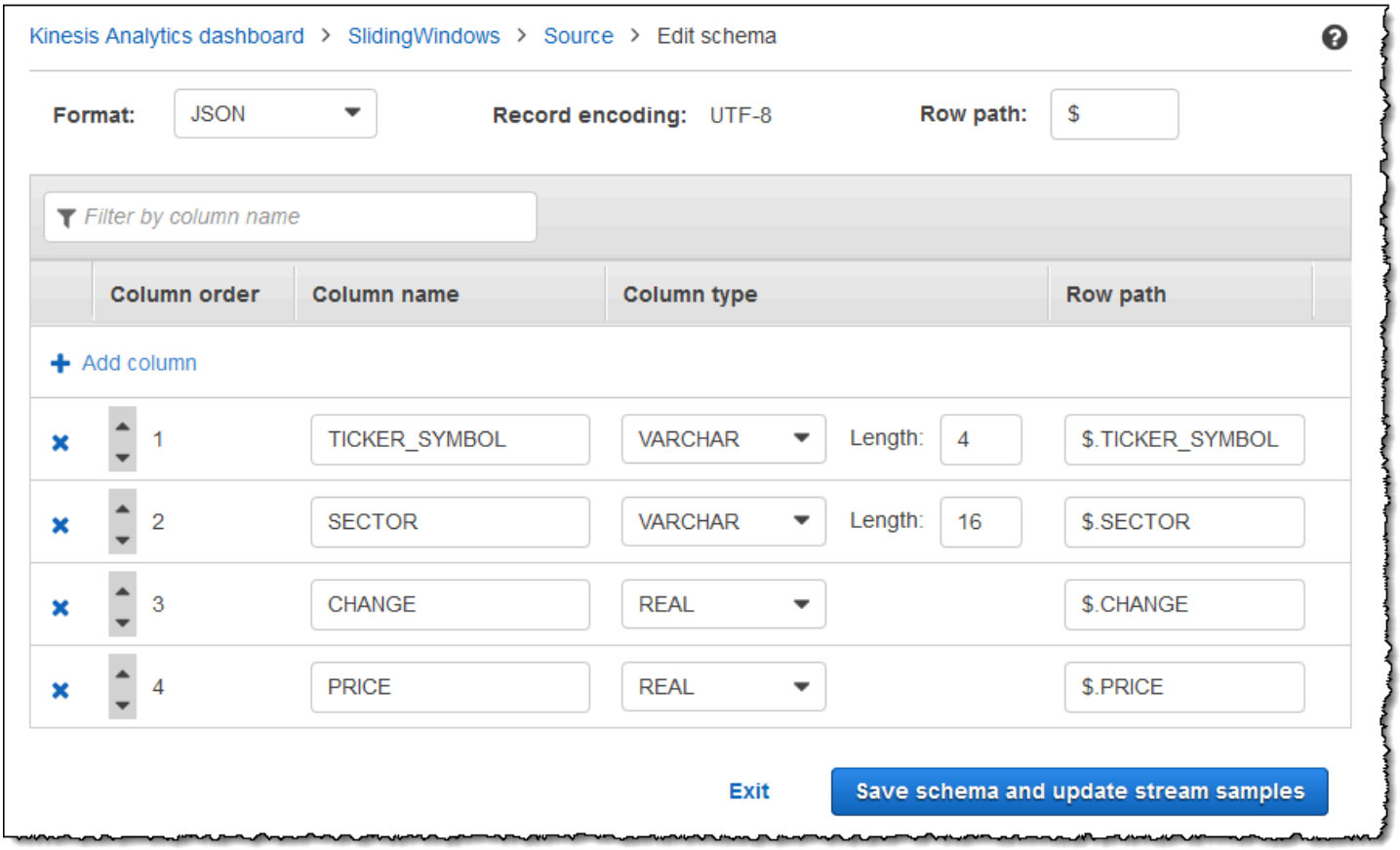


3. Geben Sie unter Column name (Spaltenname) einen neuen Namen für eine Spalte ein, um diese umzubenennen. Der Name der neuen Spalte darf nicht leer sein, muss mehr als ein einzelnes Zeichen enthalten und darf keine reservierten SQL-Schlüsselwörter enthalten. Dieser muss

ebenfalls den Namenskriterien für gewöhnliche SQL-Kennungen entsprechen: Er muss mit einem Buchstaben beginnen und darf nur Buchstaben, Unterstriche und Ziffern enthalten.

4. Um den Datentyp einer Spalte zu ändern, wählen Sie für Column type einen neuen Datentyp aus. Handelt es sich beim neuen Datentyp um `CHAR`, `VARBINARY` oder `VARCHAR`, geben Sie unter Length (Länge) eine Datenlänge ein. Weitere Informationen finden Sie unter Data Types.
5. Wählen Sie Save schema and update stream, um Ihre Änderungen zu speichern.

Das geänderte Schema wird im Editor angezeigt und sieht ähnlich dem folgenden Schema aus.



Wenn Ihr Schema zahlreiche Zeilen besitzt, können Sie die Zeilen mittels Filter by column name filtern. Um beispielsweise Spaltennamen zu bearbeiten, die mit `P` beginnen, wie eine `Price`-Spalte, geben Sie `P` im Feld Nach Spaltennamen filtern ein.

Bearbeiten eines CSV-Schemas

Sie können CSV-Schemata bearbeiten, indem Sie die folgenden Schritte ausführen.

So bearbeiten Sie ein CSV-Schema

1. Wählen Sie im Schema-Editor für Row delimiter das Trennzeichen aus, das im eingehenden Daten-Stream verwendet wird. Dies ist das Trennzeichen zwischen Datensätzen von Daten in Ihrem Stream, z. B. ein Zeilenumbruchzeichen.
2. Wählen Sie für Column delimiter das Trennzeichen aus, das im eingehenden Daten-Stream verwendet wird. Dies ist das Trennzeichen zwischen Feldern von Daten in Ihrem Stream, z. B. ein Komma.
3. Um eine Spalte hinzuzufügen, wählen Sie Add column.

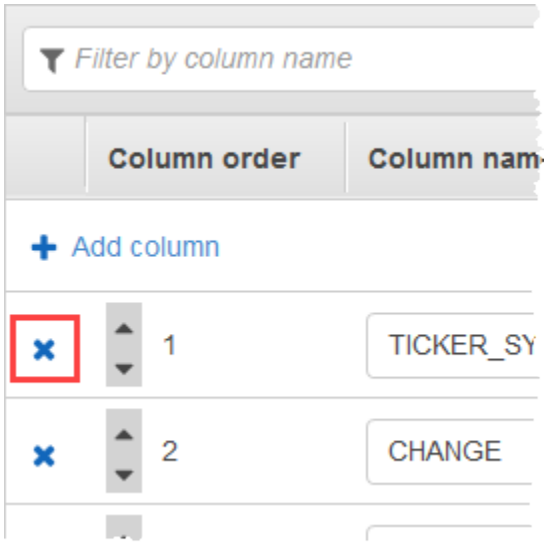
   In der ersten Spaltenposition wird eine neue Spalte angezeigt. Zum Ändern der Spaltenreihenfolge wählen Sie die nach oben und unten zeigenden Pfeile neben dem Spaltennamen.

   Geben Sie für neue Spalte folgende Informationen an:

   - Geben Sie unter Column name (Spaltenname) einen Namen ein.

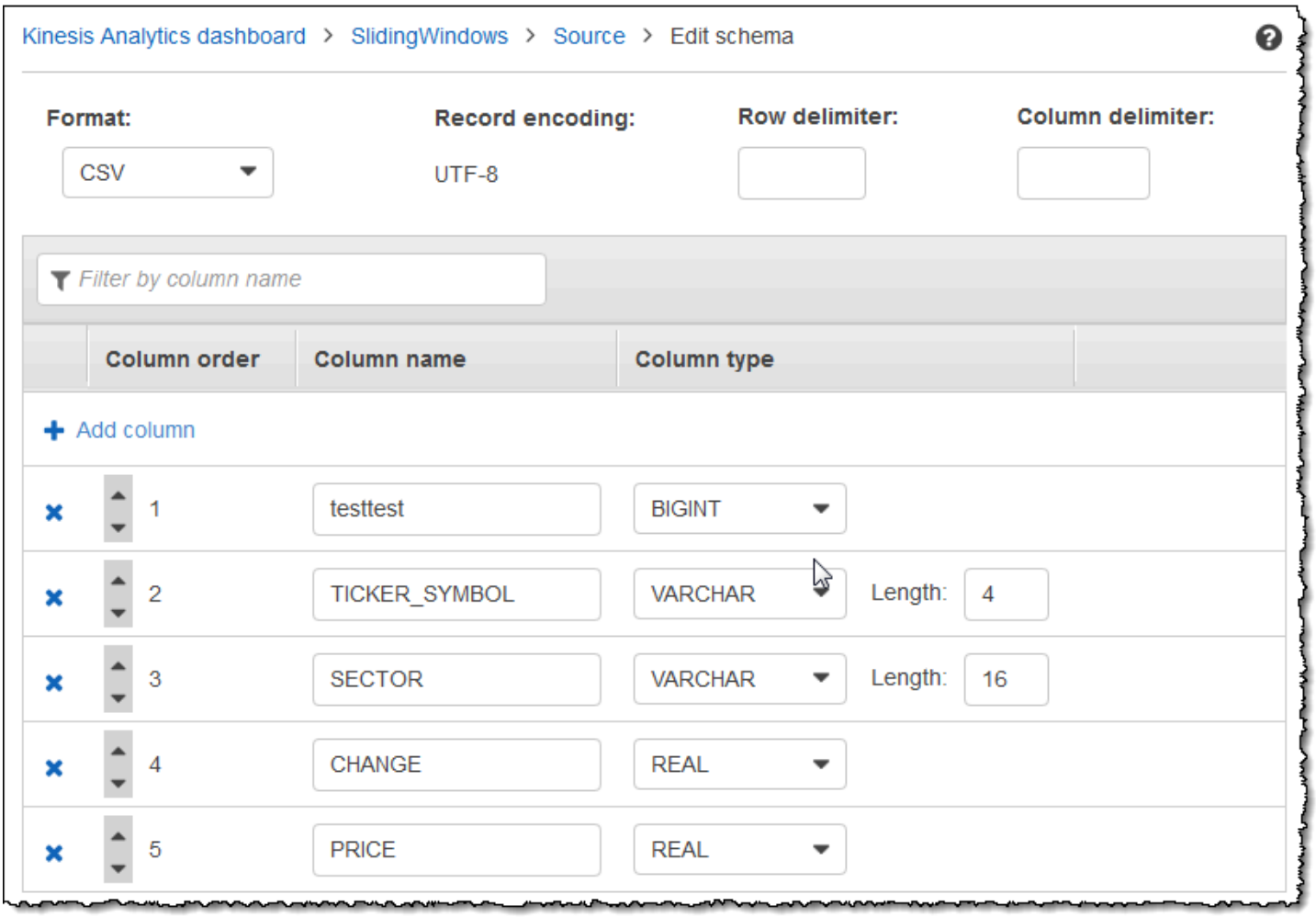
     Ein Spaltenname darf nicht leer sein, muss mehr als ein einzelnes Zeichen enthalten und darf keine reservierten SQL-Schlüsselwörter enthalten. Dieser muss ebenfalls den Namenskriterien für gewöhnliche SQL-Kennungen entsprechen: Er muss mit einem Buchstaben beginnen und darf nur Buchstaben, Unterstriche und Ziffern enthalten.
   - Geben Sie unter Column Type (Spaltentyp) einen SQL-Datentyp ein.

     Jeder unterstützte SQL-Datentyp kann ein Spaltentyp sein. Wenn es sich beim neuen Datentyp um CHAR, VARBINARY oder VARCHAR handelt, geben Sie für Length eine Datenlänge an. Weitere Informationen finden Sie unter Data Types.
4. Um eine Spalte zu löschen, wählen Sie das Symbol x neben der Spaltennummer.



5. Geben Sie unter Column name (Spaltenname) einen neuen Namen für eine Spalte ein, um diese umzubenennen. Der Name der neuen Spalte darf nicht leer sein, muss mehr als ein einzelnes Zeichen enthalten und darf keine reservierten SQL-Schlüsselwörter enthalten. Dieser muss ebenfalls den Namenskriterien für gewöhnliche SQL-Kennungen entsprechen: Er muss mit einem Buchstaben beginnen und darf nur Buchstaben, Unterstriche und Ziffern enthalten.
6. Um den Datentyp einer Spalte zu ändern, wählen Sie für Column type einen neuen Datentyp aus. Wenn es sich beim neuen Datentyp um CHAR, VARBINARY oder VARCHAR handelt, geben Sie für Length eine Datenlänge an. Weitere Informationen finden Sie unter Data Types.
7. Wählen Sie Save schema and update stream, um Ihre Änderungen zu speichern.

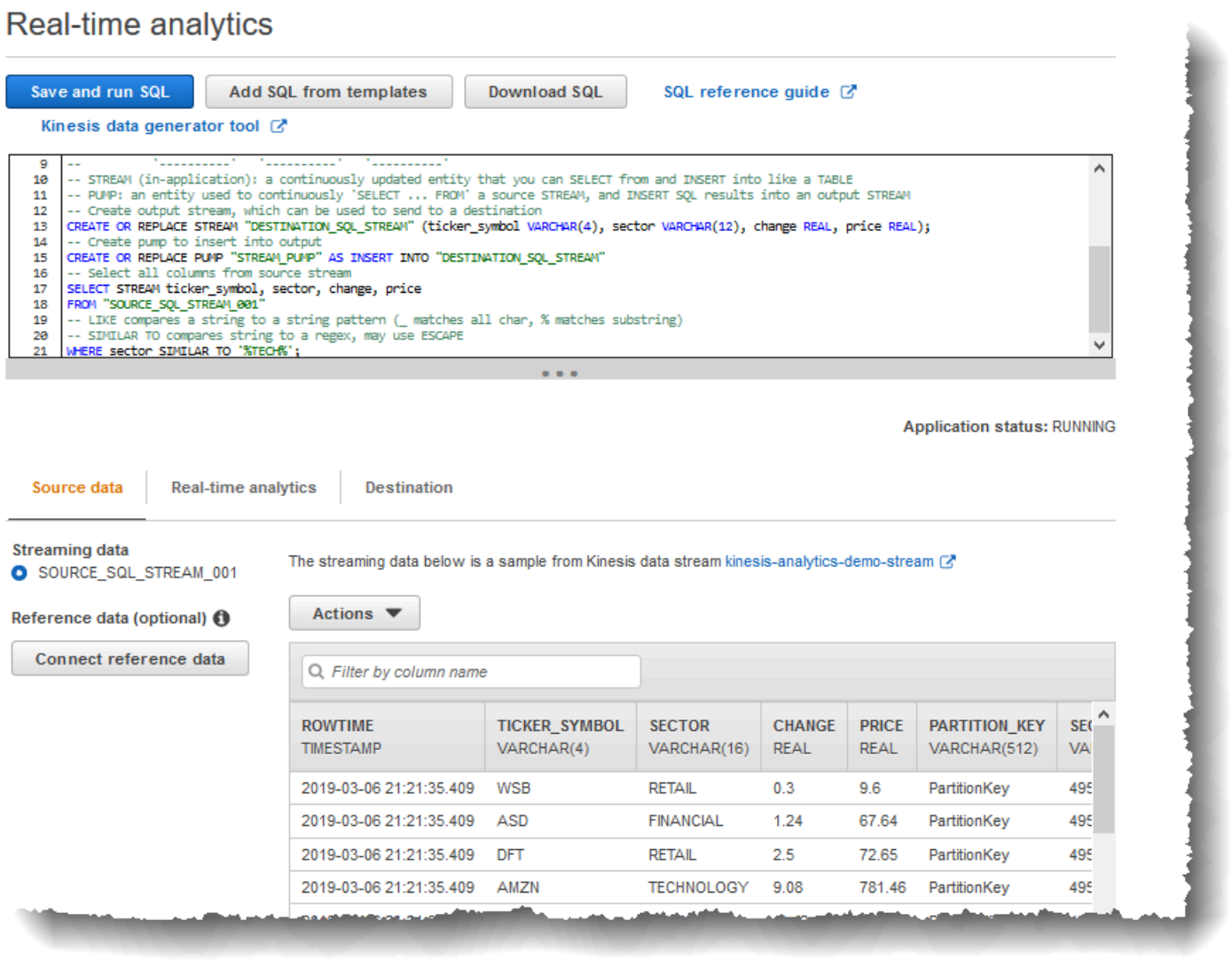
Das geänderte Schema wird im Editor angezeigt und sieht ähnlich dem folgenden Schema aus.



Wenn Ihr Schema zahlreiche Zeilen besitzt, können Sie die Zeilen mittels Filter by column name filtern. Um beispielsweise Spaltennamen zu bearbeiten, die mit `P` beginnen, wie eine `Price`-Spalte, geben Sie `P` im Feld Nach Spaltennamen filtern ein.

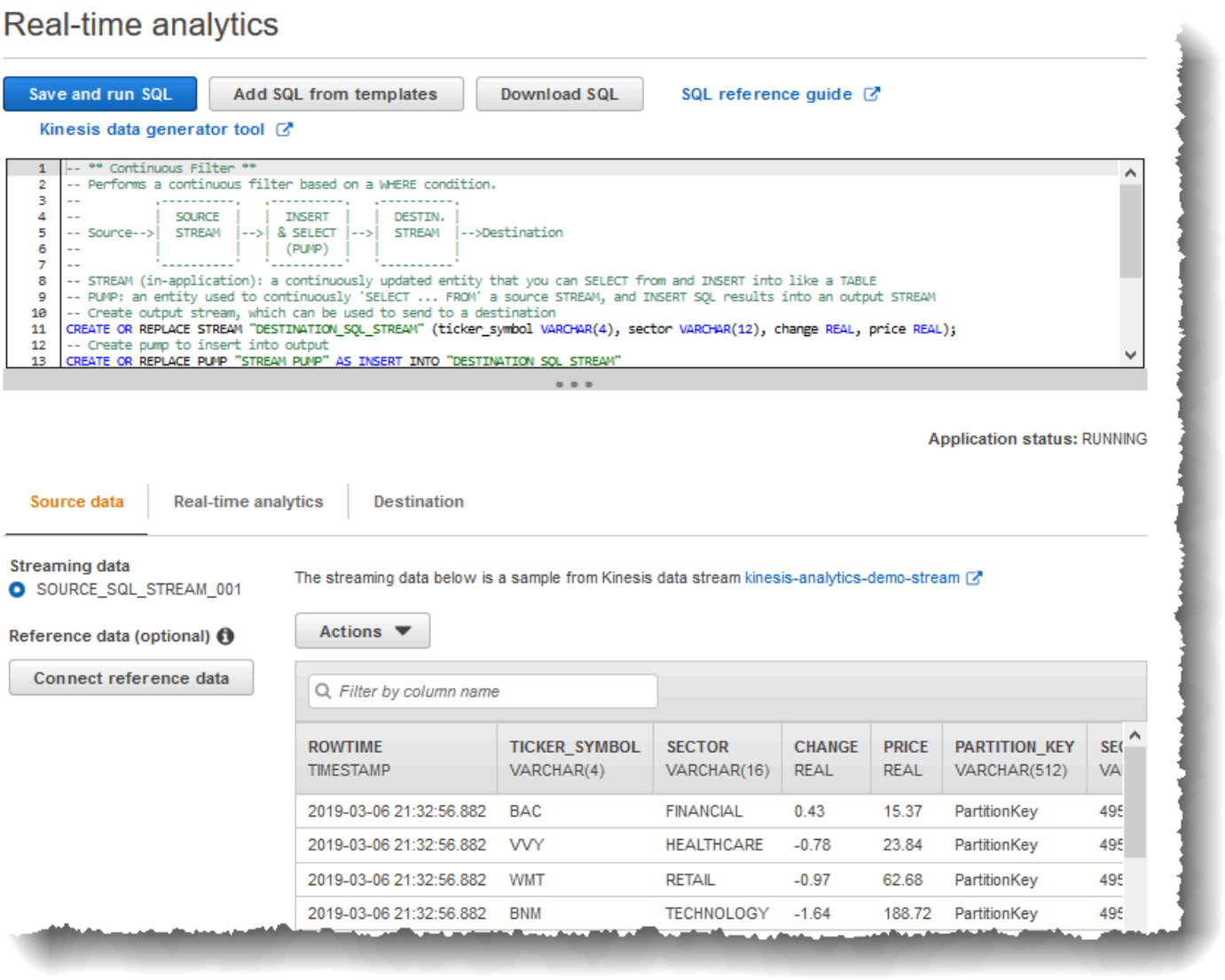
Arbeiten mit dem SQL-Editor

Im Folgenden finden Sie Informationen zu Abschnitten im SQL-Editor und dazu, wie diese jeweils funktionieren. Sie können im SQL-Editor entweder eigenen Code erstellen oder Add SQL from templates wählen. Eine SQL-Vorlage stellt Ihnen SQL-Beispiel-Code bereit, der Sie beim Schreiben gängiger Amazon-Kinesis-Data-Analytics-Anwendungen unterstützen kann. Die Beispielanwendungen in diesem Handbuch verwenden einige dieser Vorlagen. Weitere Informationen finden Sie unter Kinesis Data Analytics for SQL.



Registerkarte „Source Data“

Die Registerkarte Source data identifiziert eine Streaming-Quelle. Sie identifiziert auch den In-Application-Eingabe-Stream, dem diese Quelle zugeordnet ist und der die Eingabekonfiguration der Anwendung bereitstellt.



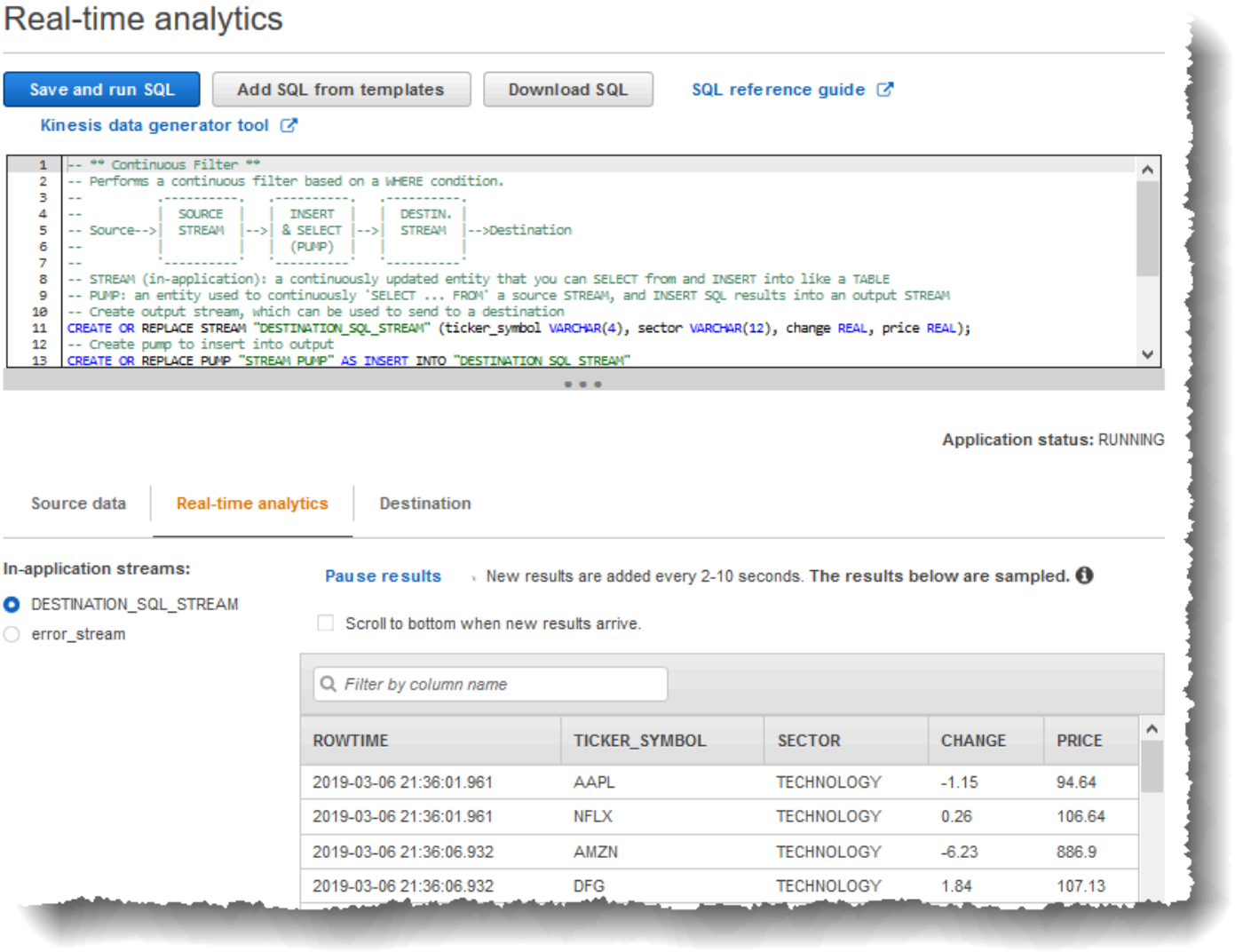
Amazon-Kinesis-Data-Analytics stellt die folgenden Zeitstempelspalten bereit, sodass Sie in Ihrer Eingabekonfiguration keine explizite Zuweisung bereitstellen müssen:

- ROWTIME – In jeder Zeile in einem In-Application-Stream gibt es eine spezielle Spalte mit der Bezeichnung `ROWTIME`. Diese Spalte ist der Zeitstempel für den Zeitpunkt, zu dem Kinesis Data Analytics die Zeile in den ersten In-Application-Stream eingefügt hat.
- Approximate_Arrival_Time – Datensätze in Ihrer Streaming-Quelle enthalten die Spalte `Approximate_Arrival_Timestamp`. Dies ist der Zeitstempel für die ungefähre Ankunftszeit, der gesetzt wird, wenn die Streaming-Quelle den entsprechenden Datensatz erfolgreich erhält und speichert. Kinesis Data Analytics ruft diese Spalte in den In-Application-Eingabe-Stream als `Approximate_Arrival_Time` ab. Amazon-Kinesis-Data-Analytics stellt diese Spalte nur in dem In-Application-Eingabe-Stream bereit, der der Streaming-Quelle zugeordnet ist.

Diese Zeitstempelwerte sind in Abfragen mit Zeitfenstern hilfreich. Weitere Informationen finden Sie unter Abfragen mit Fenstern.

Registerkarte „Real-Time Analytics“

Auf der Registerkarte Real-time analytics werden alle In-Application-Streams angezeigt, die von Ihrem Anwendungs-Code erstellt werden. Diese Gruppe von Streams enthält den Fehler-Stream (`error_stream`), den Amazon-Kinesis-Data-Analytics für alle Anwendungen bereitstellt.



Registerkarte „Destination“

Über die Registerkarte Zieladresse können Sie die Anwendungsausgabe so konfigurieren, dass sie In-Application-Streams an externe Ziele weiterleitet. Sie können die Ausgabe für die Weiterleitung

von Daten aus allen In-Application-Streams an externe Ziele konfigurieren. Weitere Informationen finden Sie unter Konfigurieren der Anwendungsausgabe.

SQL-Streaming-Konzepte

Amazon-Kinesis-Data-Analytics implementiert den ANSI 2008 SQL-Standard mit Erweiterungen. Diese Erweiterungen ermöglichen Ihnen die Verarbeitung von Streaming-Daten. Die folgenden Themen behandeln wichtige SQL-Streaming-Konzepte.

Themen

In-Application-Streams und Pumps

Wenn Sie die Anwendungseingabe konfigurieren, wird eine Streaming-Quelle einem erstellten In-Application-Stream zugeordnet. Die Daten fließen kontinuierlich aus der Streaming-Quelle in den In-Application-Stream. Ein In-Application-Stream funktioniert wie eine Tabelle, die Sie mit SQL-Anweisungen abfragen können, wird jedoch als Stream bezeichnet, da die Daten kontinuierlich fließen.

> **Note**
>
> Verwechseln Sie In-Application-Streams nicht mit Amazon Kinesis Kinesis-Datenströmen und Firehose-Lieferströmen. In-Application-Streams sind nur im Kontext einer Amazon-Kinesis-Data-Analytics-Anwendung vorhanden. Kinesis-Datenströme und Firehose-Lieferströme existieren unabhängig von Ihrer Anwendung. Sie können sie in der Konfiguration Ihrer Anwendungseingabe als Streaming-Quelle oder in der Ausgabekonfiguration als Ziel konfigurieren.

Sie können außerdem bei Bedarf weitere In-Application-Streams erstellen, um Zwischenergebnisse aus Abfragen zu speichern. Das Erstellen eines In-Application-Streams ist ein zweistufiger Prozess. Zuerst erstellen Sie einen In-Application-Stream. Anschließend pumpen Sie Daten in

diesen. Angenommen, die Eingabekonfiguration Ihrer Anwendung erstellt einen In-Application-Stream mit dem Namen INPUTSTREAM. Im folgenden Beispiel erstellen Sie einen anderen Stream (TEMPSTREAM) und pumpen anschließend Daten aus INPUTSTREAM in diesen.

1. Erstellen Sie einen In-Application-Stream (TEMPSTREAM) mit drei Spalten, wie im Folgenden gezeigt:

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

   Die Spaltennamen werden in Anführungszeichen angegeben, wodurch zwischen Groß- und Kleinschreibung unterschieden wird. Weitere Informationen finden Sie im Abschnitt Kennungen in der SQL-Referenz zu Amazon-Kinesis-Data-Analytics.

2. Fügen Sie mittels eines Pumps Daten in den Stream ein. Eine Pump ist eine kontinuierlich ausgeführte Einfüge-Abfrage, die Daten aus einem In-Application-Stream in einen anderen In-Application-Stream einfügt. Die folgende Anweisung erstellt eine Pump (SAMPLEPUMP) und fügt Daten in den TEMPSTREAM ein, indem sie Datensätze aus einem anderen Stream (INPUTSTREAM) auswählt.

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

Sie können in einen In-Application-Stream aus mehreren Quellen schreiben und es können mehrere Datensätze aus einem Stream ausgewählt werden. Stellen Sie sich einen In-Application-Stream als Implementierung eines publish/subscribe Messaging-Paradigmas vor. In diesem Paradigma kann eine Datenzeile, einschließlich Erstellungs- und Empfangszeitpunkt, durch eine Kaskade von SQL-Streaming-Anweisungen verarbeitet, interpretiert und weitergeleitet werden, ohne in einem herkömmlichen RDBMS gespeichert werden zu müssen.

Nachdem ein In-Application-Stream erstellt wurde, können Sie normale SQL-Abfragen ausführen.

> **Note**
>
> Bei der Abfrage von Streams müssen die meisten SQL-Anweisungen ein zeilen- oder zeitbasiertes Fenster verwenden. Weitere Informationen finden Sie unter Abfragen mit Fenstern.

Sie können Streams auch zusammenfügen. Beispiele für das Zusammenfügen von Streams finden Sie unter Operationen für Streaming-Daten: Zusammenführen von Streams.

Zeitstempel und die ROWTIME-Spalte

In-Application-Streams enthalten eine spezielle Spalte namens `ROWTIME`. In dieser wird ein Zeitstempel gespeichert, wenn Amazon-Kinesis-Data-Analytics eine Zeile in den ersten In-Application-Stream einfügt. `ROWTIME` spiegelt den Zeitstempel wider, zu dem Amazon-Kinesis-Data-Analytics nach dem Lesen aus der Streaming-Quelle einen Datensatz in den ersten In-Application-Stream eingefügt hat. Dieser `ROWTIME`-Wert wird anschließend in der gesamten Anwendung beibehalten.

> **Note**
>
> Wenn Sie Datensätze aus einem In-Application-Stream in einen anderen pumpen, müssen Sie die Spalte `ROWTIME` nicht explizit kopieren. Amazon-Kinesis-Data-Analytics kopiert diese Spalte für Sie.

Amazon-Kinesis-Data-Analytics stellt sicher, dass die `ROWTIME`-Werte gleichmäßig erhöht werden. Sie verwenden diesen Zeitstempel in Abfragen mit Fenstern auf Zeitbasis. Weitere Informationen finden Sie unter Abfragen mit Fenstern.

Sie können die Spalte ROWTIME in Ihrer `SELECT`-Anweisung wie jede andere Spalte in Ihrem In-Application-Stream aufrufen. Beispiel:

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

Verstehen der verschiedenen Zeiten in Streaming-Analysen

Zusätzlich zu `ROWTIME` gibt es weitere Arten von Zeiten in Echtzeit-Streaming-Anwendungen. Dies sind:

- Ereigniszeit – Der Zeitstempel für den Zeitpunkt, an dem das Ereignis aufgetreten ist. Dies wird manchmal auch die clientseitige Zeit genannt. Häufig ist es wünschenswert, diese Zeit in Analysen zu verwenden, da dies die Zeit ist, zu der ein Ereignis aufgetreten ist. Zahlreiche Ereignisquellen, wie Smartphones und Web-Clients, besitzen jedoch keine zuverlässigen Uhren, was zu ungenauen Zeiten führen kann. Zusätzlich können Konnektivitätsprobleme dazu führen, dass Datensätze in einem Stream nicht in der gleichen Reihenfolge angezeigt werden, in der sie aufgetreten sind.

- Aufnahmezeit – Der Zeitstempel für den Zeitpunkt, an dem ein Datensatz der Streaming-Quelle hinzugefügt wurde. Amazon-Kinesis-Data-Streams enthalten in jedem Datensatz ein Feld namens `APPROXIMATE_ARRIVAL_TIME`, das diesen Zeitstempel bereitstellt. Dies wird manchmal auch als serverseitige Zeit bezeichnet. Diese Aufnahmezeit ist häufig eine nahe Annäherung an die Ereigniszeit. Wenn es eine Verzögerung bei der Aufnahme des Datensatzes in den Stream gibt, kann dies zu Ungenauigkeiten führen, was in der Regel selten ist. Die Aufnahmezeit ist selten nicht in der richtigen Reihenfolge, auch wenn dies aufgrund der verteilten Natur von Streaming-Daten vorkommen kann. Daher reflektiert die Aufnahmezeit die Ereigniszeit meistens genau und in der richtigen Reihenfolge.

- Verarbeitungszeit – Der Zeitstempel für den Zeitpunkt, an dem Amazon-Kinesis-Data-Analytics eine Zeile in den ersten In-Application-Stream einfügt. Amazon-Kinesis-Data-Analytics stellt diesen Zeitstempel in der Spalte `ROWTIME` bereit, die in jedem In-Application-Stream vorhanden ist. Die Verarbeitungszeit wird stets gleichmäßig erhöht. Sie ist jedoch nicht genau, wenn Ihre Anwendung hinter das Schreiben von Daten zurückfällt. (Wenn eine Anwendung zurückfällt, gibt die Verarbeitungszeit die Ereigniszeit nicht genau wieder.) Diese `ROWTIME` ist in Bezug auf die Uhr zwar sehr genau, stellt möglicherweise jedoch nicht die Zeit dar, zu der das Ereignis tatsächlich aufgetreten ist.

Die Verwendung dieser Zeiten in Abfragen mit Fenstern auf Zeitbasis hat Vor- und Nachteile. Wir empfehlen, mindestens eine dieser Zeiten zu wählen und je nach Anwendungsfall eine Strategie für den Umgang mit den relevanten Nachteilen zu entwickeln.

> **Note**
>
> Wenn Sie Fenster auf Zeilenbasis verwenden, stellt Zeit kein Problem dar und Sie können diesen Abschnitt ignorieren.

Wir empfehlen, eine Zwei-Fenster-Strategie, die zwei Fenster auf Zeitbasis verwendet, sowohl `ROWTIME` und eine der beiden anderen Zeiten (Aufnahme- oder Ereigniszeit).

- Sie sollten `ROWTIME` als erstes Fenster verwenden, das die Häufigkeit steuert, mit der die Abfrage die Ergebnisse ausgibt, wie im folgenden Beispiel gezeigt. Sie wird nicht als logische Zeit verwendet.
- Sie sollten eine der beiden anderen Zeiten als logische Zeit verwenden, um sie mit Ihren Analysen zu verknüpfen. Diese Zeit stellt den Zeitpunkt dar, zu dem das Ereignis aufgetreten ist. Im folgenden Beispiel besteht das Ziel der Analyse darin, die Datensätze zu gruppieren und eine Zahl nach Ticker zurückzugeben.

Der Vorteil dieser Strategie besteht darin, dass eine Zeit verwendet werden kann, die den Zeitpunkt angibt, zu dem das Ereignis aufgetreten ist. Auch treten mit dieser Strategie keinerlei Probleme auf, wenn Ihre Anwendung zurückfällt oder Ereignisse nicht in der richtigen Reihenfolge eintreffen. Wenn die Anwendung zurückfällt, wenn Datensätze in den In-Application-Stream eingefügt werden, werden sie dennoch von der logischen Zeit im zweiten Fenster gruppiert. Die Abfrage verwendet `ROWTIME`, um die Reihenfolge der Verarbeitung zu gewährleisten. Datensätze, die verspätet eintreffen (der Aufnahmezeitstempel zeigt im Vergleich zum `ROWTIME`-Wert einen früheren Wert an), werden ebenfalls erfolgreich verarbeitet.

Betrachten Sie die folgende Abfrage für den in der Erste Schritte-Übung verwendeten Demo-Stream. Die Abfrage verwendet die `GROUP  BY`-Klausel und gibt die Tickerzahl innerhalb eines rollierenden Zeitfensters von einer Minute an.

```
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);
```

In GROUP BY gruppieren Sie die Datensätze zunächst anhand der ROWTIME in einem Zeitfenster von einer Minute und anschließend nach APPROXIMATE_ARRIVAL_TIME.

Die Zeitstempelwerte im Ergebnis werden auf das nächste 60-Sekunden-Intervall gerundet. Die erste von der Abfrage ausgegebene Ergebnisgruppe zeigt Datensätze in der ersten Minute an. Die zweite Gruppe von ausgegebenen Ergebnissen zeigt Datensätze in den nächsten Minuten an, basierend auf ROWTIME. Der letzte Datensatz gibt an, dass die Anwendung den Datensatz verspätet in den In-Application-Stream aufgenommen hat(sie zeigt im Vergleich zum Aufnahmezeitstempel einen verspäteten ROWTIME-Wert an).

```
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.
```

Sie können die Ergebnisse kombinieren, um eine endgültige genaue Zahl pro Minute zu erhalten, indem die Ergebnisse zu einer Downstream-Datenbank verschoben werden. Sie können die Anwendungsausgabe beispielsweise so konfigurieren, dass die Ergebnisse in einem Firehose-Lieferdatenstrom gespeichert werden, der in eine Amazon Redshift Redshift-Tabelle schreiben

kann. Wenn sich die Ergebnisse in einer Amazon-Redshift-Tabelle befinden, können Sie eine Tabellenabfrage durchführen, um die gesamte Zahl für die Gruppe nach `Ticker_Symbol` zu berechnen. Im Fall von XYZ ist die Gesamtzahl genau (6+1), obwohl ein Datensatz spät eingetroffen ist.

Kontinuierliche Abfragen

Eine Abfrage über einen Stream wird kontinuierlich über Streaming-Daten ausgeführt. Diese kontinuierliche Ausführung ermöglicht Szenarien wie die Fähigkeit von Anwendungen, einen Stream kontinuierlich abzufragen und Warnungen zu generieren.

In der Einführungsübung gibt es einen In-Application-Stream namens `SOURCE_SQL_STREAM_001`. Dieser erhält kontinuierlich Aktienpreise aus einem Demo-Stream (einem Kinesis-Datenstrom). Das Schema ist wie folgt:

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

Angenommen, Sie sind an Aktienpreisänderungen von mehr als 15 % interessiert. Sie können in Ihrem Anwendungscode die folgende Abfrage verwenden. Diese Abfrage wird kontinuierlich ausgeführt und gibt Datensätze aus, wenn eine Aktienpreisänderung von mehr als 15 % entdeckt wird.

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

Verwenden Sie das folgende Verfahren, um eine Amazon-Kinesis-Data-Analytics-Anwendung einzurichten und diese Abfrage zu testen.

So testen Sie die Abfrage

1. Erstellen Sie eine Anwendung mit der Erste Schritte-Übung.
2. Ersetzen Sie die `SELECT`-Anweisung im Anwendungscode durch die vorherige `SELECT`-Abfrage. Der resultierende Anwendungscode wird im Folgenden gezeigt:

   ```
   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;
```

Abfragen mit Fenstern

SQL-Abfragen in Ihrem Anwendungscode werden kontinuierlich über In-Application-Streams ausgeführt. Ein In-Application-Stream entspricht einer unbegrenzten Menge von Daten, die kontinuierlich durch Ihre Anwendung fließen. Um daher aus dieser kontinuierlich aktualisierten Eingabe Ergebnissätze zu erhalten, werden Abfragen häufig durch ein Fenster begrenzt, das auf Zeit- oder Zeilenbasis definiert wird. Diese Fenster werden auch als Fenster-SQL bezeichnet.

Für eine Abfrage mit einem Fenster auf Zeitbasis geben Sie die Fenstergröße in Bezug auf die Zeit an (beispielsweise ein Fenster von einer Minute). Dies erfordert in Ihrem In-Application-Stream eine Zeitstempelspalte, die gleichmäßig erhöht wird. (Der Zeitstempel für eine neue Zeile ist größer oder gleich groß wie der der vorherigen Zeile.) Amazon-Kinesis-Data-Analytics stellt für jeden In-Application-Stream eine solche Zeitstempelspalte bereit; diese hat den Namen `ROWTIME`. Sie können diese Spalte verwenden, wenn Sie Abfragen auf Zeitbasis angeben. Für Ihre Anwendung wählen Sie möglicherweise eine andere Zeitstempeloption. Weitere Informationen finden Sie unter Zeitstempel und die ROWTIME-Spalte.

Bei einer Abfrage mit einem Fenster auf Zeilenbasis geben Sie die Größe des Fensters in Bezug auf die Anzahl der Zeilen an.

Sie können eine Abfrage so angeben, dass Datensätze in einem rollierenden, gleitenden oder versetzten Fenster verarbeitet werden, abhängig von den Anforderungen Ihrer Anwendung. Kinesis Data Analytics unterstützt die folgenden Fenstertypen:

- Versetzte Fenster: Eine Abfrage, die Daten unter Verwendung zeitbasierter Fenster mit Schlüsseln aggregiert, die bei Erhalt der Daten geöffnet werden. Die Schlüssel ermöglichen die Nutzung mehrerer überlappender Fenster. Dies ist die empfohlene Methode, um Daten mithilfe zeitbasierter Fenster zu aggregieren, da Stagger-Fenster im Vergleich zu Tumbling-Fenstern verspätete out-of-order Daten reduzieren.

- Rollierende Fenster: Eine Abfrage, die Daten unter Verwendung unterschiedlicher zeitbasierter Fenster aggregiert, die in regelmäßigen Abständen geöffnet und geschlossen werden.
- Gleitende Fenster: Eine Abfrage, die Daten kontinuierlich unter Verwendung eines festen Zeit- oder eines rowcount-Intervalls aggregiert.

Versetzte Fenster

Das Verwenden von Versetzte Fenster ist eine Methode zur Anordnung von Fenstern für die Analyse von Datengruppen, die ungleichmäßig eingehen. Diese Methode eignet sich für beliebige Zeitreihenanalysen, z. B. für eine Gruppe zusammengehöriger Verkaufs- oder Protokolldatensätze.

Beispiel: VPC Flow Logs haben einen Erfassungszeitraum von ca. 10 Minuten. Sie können jedoch einen Erfassungszeitraum von bis zu 15 Minuten aufweisen, wenn das Aggregieren von Daten auf dem Client erfolgt. Versetzte Fenster eignen sich ideal, um diese Protokolle zu Analysezwecken zu aggregieren.

Versetzte Fenster lösen das Problem, das auftritt, wenn zusammengehörige Datensätze nicht in dasselbe zeitbeschränkte Fenster fallen (beispielsweise bei Verwendung rollierender Fenster).

Teilergebnisse mit rollierenden Fenstern

Es gibt bestimmte Einschränkungen bei der Verwendung Rollierende Fenster für die Aggregation von verspäteten out-of-order Daten.

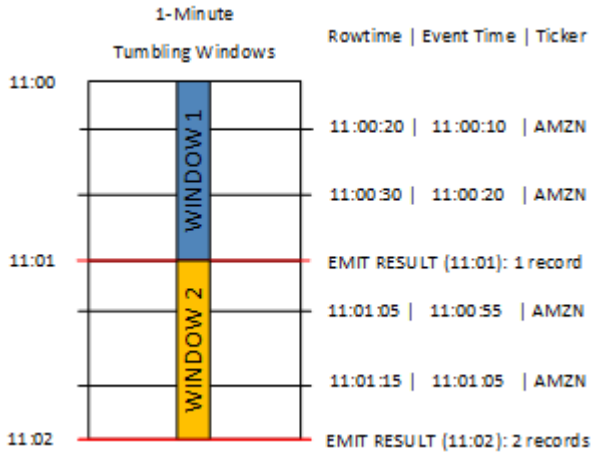
Wenn rollierende Fenster zum Analysieren von Gruppen zeitversetzter Daten verwendet werden, fallen die einzelnen Datensätze möglicherweise in unterschiedliche Fenster. Daher müssen die Teilergebnisse jedes Fensters zu einem späteren Zeitpunkt kombiniert werden, um vollständige Ergebnisse für jede Gruppe von Datensätzen anzuzeigen.

In der folgenden Abfrage für rollierende Fenster werden Datensätze nach Zeilenzeit, Ereigniszeit und Tickersymbol in Fenstern gruppiert:

```
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);
```

In der folgenden Abbildung zählt eine Anwendung die empfangenen Handelstransaktionen basierend auf dem Zeitpunkt der Transaktionen (Ereigniszeit) minutenweise. Die Anwendung kann ein rollierendes Fenster zum Gruppieren der Daten basierend auf Zeilenzeit und Ereigniszeit verwenden. Die Anwendung empfängt vier Datensätze, die alle innerhalb einer Minute eintreffen. Die Anwendung gruppiert die Datensätze nach Zeilenzeit, Ereigniszeit und Tickersymbol. Da einige der Datensätze nach dem Ende des ersten rollierenden Fensters ankommen, fallen nicht alle Datensätze in dasselbe rollierende Ein-Minuten-Fenster.



Das vorhergehende Diagramm enthält die folgenden Ereignisse:

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

Die Ergebnismenge der Anwendung mit rollierendem Fenster ähnelt der folgenden.

ROWTIME	EVENT_TIME	TICKER_SYMBOL	COUNT
11:01:00	11:00:00	AMZN	2
11:02:00	11:00:00	AMZN	1
11:02:00	11:01:00	AMZN	1

In der Ergebnismenge werden drei Ergebnisse zurückgegeben:

- Ein Datensatz mit einem `ROWTIME`-Wert von 11:01:00, der die ersten zwei Datensätze aggregiert.
- Ein Datensatz um 11:02:00, der nur den dritten Datensatz aggregiert. Dieser Datensatz weist einen `ROWTIME`-Wert im zweiten Fenster, aber einen `EVENT_TIME`-Wert im ersten Fenster auf.
- Ein Datensatz um 11:02:00, der nur den vierten Datensatz aggregiert.

Zum Analysieren der vollständigen Ergebnismenge müssen die Datensätze im persistenten Speicher aggregiert werden. Dadurch wachsen Komplexität und Verarbeitungsanforderungen der Anwendung.

Vollständige Ergebnisse mit versetzten Fenstern

Zur Verbesserung der Genauigkeit bei der Analyse zeitbezogener Datensätze stellt Kinesis Data Analytics einen neuen Fenstertyp mit der Bezeichnung Versetzte Fenster bereit. Fenster dieses Typs werden geöffnet, wenn das erste Ereignis mit übereinstimmendem Partitionsschlüssel ankommt, nicht für einen festen Zeitraum. Die Fenster werden basierend auf dem angegebenen Alter geschlossen, das ab dem Zeitpunkt gemessen wird, zu dem das Fenster geöffnet wird.

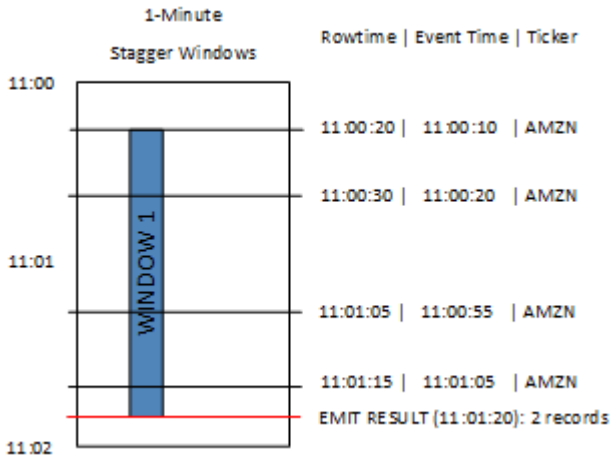
Ein versetztes Fenster ist ein separates zeitbeschränktes Fenster für jede Schlüsselgruppierung in einer WINDOW-Klausel. Die Anwendung aggregiert jedes Ergebnis der WINDOW-Klausel im eigenen Zeitfenster, statt ein einzelnes Fenster für alle Ergebnisse zu verwenden.

In der folgenden Abfrage für versetzte Fenster werden Datensätze nach Ereigniszeit und Tickersymbol in Fenstern gruppiert:

```
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);
```

Im folgenden Diagramm werden Ereignisse nach Ereigniszeit und Tickersymbol in versetzten Fenstern gruppiert.



Das vorstehende Diagramm umfasst die folgenden Ereignisse, die denen entsprechen, die mit der Anwendung mit rollierenden Fenstern analysiert wurden:

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

Die Ergebnismenge der Anwendung für versetzte Fenster ähnelt der folgenden.

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

Der zurückgegebene Datensatz aggregiert die ersten drei Eingabedatensätze. Die Datensätze werden nach versetzten Eine-Minute-Fenstern gruppiert. Das versetzte Fenster startet, wenn die Anwendung den ersten AMZN-Datensatz (mit einem `ROWTIME`-Wert von 11:00:20) empfängt. Wenn das versetzte Eine-Minute-Fenster abläuft (um 11:01:20), wird ein Datensatz mit den Ergebnissen, die in das versetzte Fenster fallen (basierend auf `ROWTIME` und `EVENT_TIME`), in den Ausgabe-Stream geschrieben. Bei Verwendung eines versetzten Fensters werden alle Datensätze mit `ROWTIME`- und `EVENT_TIME`-Werten innerhalb eines Eine-Minute-Fensters in einem einzelnen Ergebnis ausgegeben.

Der letzte Datensatz (mit einer `EVENT_TIME` außerhalb der einminütigen Aggregation) wird separat aggregiert. Dies liegt daran, dass `EVENT_TIME` einen der Partitionsschlüssel darstellt, mit dem die Datensätze in Ergebnissätze eingeteilt werden, und der Partitionsschlüssel für die `EVENT_TIME` des ersten Fensters `11:00` ist.

Die Syntax für versetzte Fenster wird in einer speziellen `WINDOWED BY`-Klausel definiert. Diese Klausel wird anstelle der `GROUP BY`-Klausel für Streaming-Aggregationen verwendet. Die Klausel folgt unmittelbar auf die optionale `WHERE`-Klausel und steht vor der `HAVING`-Klausel.

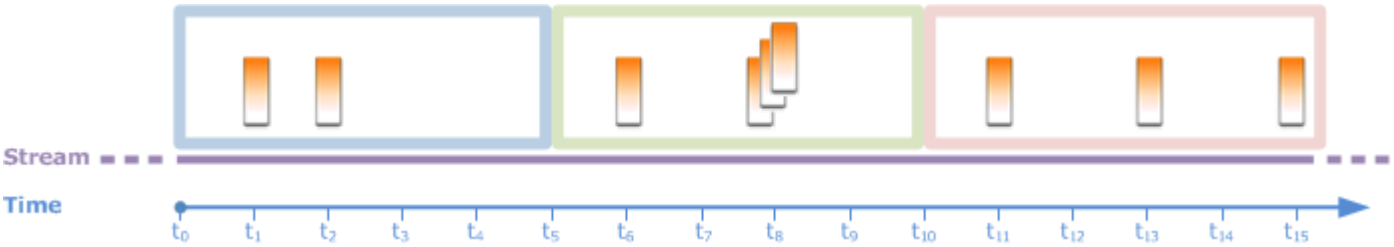
Das versetzte Fenster wird in der `WINDOWED BY`-Klausel definiert und nimmt zwei Parameter an: Partitionsschlüssel und Fensterdauer. Der Partitionsschlüssel partitioniert den eingehenden Daten-Stream und definiert, wann das Fenster geöffnet wird. Ein versetztes Fenster wird geöffnet, sobald das erste Ereignis mit eindeutigem Partitionsschlüssel im Stream auftritt. Das versetzte Fenster schließt nach einem festen Zeitraum, der als Fensterdauer definiert ist. Die Syntax wird im folgenden Codebeispiel veranschaulicht:

```
...
FROM <stream-name>
WHERE <... optional statements...>
WINDOWED BY STAGGER(
 PARTITION BY <partition key(s)>
```

```
 RANGE INTERVAL <window length, interval>
);
```

Rollierende Fenster (Zusammenfassungen mit GROUP BY)

Wenn eine Abfrage mit Fenster jedes Fenster so verarbeitet, dass diese sich nicht überschneiden, wird das Fenster als rollierendes Fenster bezeichnet. In diesem Fall gehört jeder Datensatz in einem In-Application-Stream zu einem bestimmten Fenster. Er wird nur einmal verarbeitet (wenn die Abfrage das Fenster verarbeitet, zu dem der Datensatz gehört).



Beispielsweise verarbeitet eine Zusammenfassungsabfrage, die eine `GROUP BY`-Klausel verwendet, Zeilen in einem rollierenden Fenster. Der Demo-Stream in der Erste Schritte-Übung empfängt Aktienpreisdaten, die dem In-Application-Stream `SOURCE_SQL_STREAM_001` in Ihrer Anwendung zugeordnet sind. Dieser Stream weist das folgende Schema auf.

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

Angenommen, Sie möchten in Ihrem Anwendungscode zusammengefasste Mindest-/Höchstpreise für jeden Ticker im Zeitraum von einer Minute finden. Sie können die folgende Abfrage verwenden.

```
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);
```

Das vorherige Beispiel ist eine Abfrage mit Fenster auf Zeitbasis. Die Abfrage gruppiert Datensätze nach ROWTIME-Werten. Für Berichte auf Minutenbasis rundet die Funktion STEP die ROWTIME-Werte auf die nächste Minute ab.

> **Note**
>
> Sie können auch die FLOOR-Funktion verwenden, um Datensätze in Fenstern zu gruppieren. Die Funktion FLOOR kann Zeitwerte jedoch nur auf eine gesamte Zeiteinheit (Stunde, Minute, Sekunde usw.) abrunden. STEP wird für die Gruppierung von Datensätzen in rollierende Zeitfenster empfohlen, da hierdurch Werte auf ein beliebiges Intervall, z. B. 30 Sekunden, abgerundet werden können.

Diese Abfrage ist ein Beispiel für ein nicht überschneidendes (rollierendes) Fenster. Die GROUP BY-Klausel gruppiert Datensätze in einem Fenster von einer Minute. Jeder Datensatz gehört zu einem bestimmten Fenster (keine Überschneidungen). Die Abfrage gibt einen Ausgabedatensatz pro Minute aus, der den in der jeweiligen Minute aufgezeichneten min/max Tickerpreis angibt. Diese Art von Abfrage ist nützlich, um aus dem Eingabedaten-Stream regelmäßige Berichte zu generieren. In diesem Beispiel werden jede Minute Berichte generiert.

So testen Sie die Abfrage

1. Richten Sie anhand der Erste Schritte-Übung eine Anwendung ein.
2. Ersetzen Sie die SELECT-Anweisung im Anwendungscode durch die vorherige SELECT-Abfrage. Der resultierende Anwendungscode wird im Folgenden gezeigt:

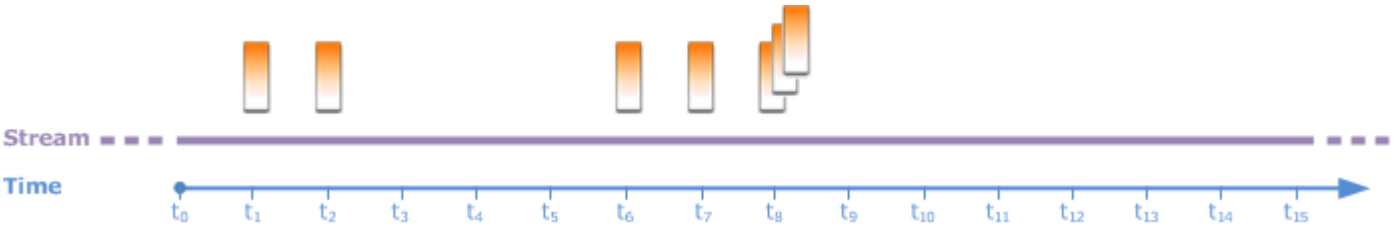
```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                                    ticker_symbol VARCHAR(4),
                                    Min_Price     DOUBLE,
                                    Max_Price     DOUBLE);
-- CREATE OR REPLACE PUMP to insert into output
CREATE OR REPLACE 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);
```

Gleitende Fenster

Statt Datensätze mittels `GROUP BY` zu gruppieren, können Sie ein Fenster auf Zeit- oder Zeilenbasis definieren. Sie führen dies aus, indem Sie eine explizite `WINDOW`-Klausel hinzufügen.

In diesem Fall gibt Amazon-Kinesis-Data-Analytics eine Ausgabe aus, während das Fenster über die Zeit gleitet und neue Datensätze im Stream angezeigt werden. Kinesis Data Analytics gibt diese Ausgabe aus, indem Zeilen im Fenster verarbeitet werden. Bei der Art der Verarbeitung können sich Fenster überschneiden; ein Datensatz kann Teil mehrerer Fenster sein und wird mit jeden Fenster verarbeitet. Das folgende Beispiel zeigt ein gleitendes Fenster.

Betrachten Sie eine einfache Abfrage, die Datensätze im Stream zählt. Dieses Beispiel setzt ein 5-Sekunden-Fenster voraus. Im folgenden Beispiel-Stream treffen neue Datensätze zu den Zeitpunkten t_1, t_2, t_6 und t_7 ein. Drei Datensätze treffen zum Zeitpunkt t_8 Sekunden ein.



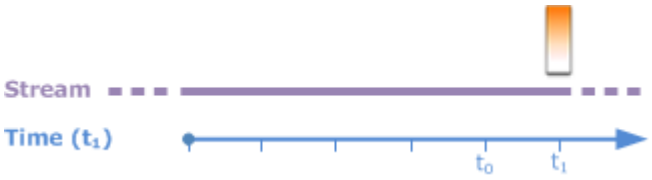
Beachten Sie Folgendes:

- Das Beispiel setzt ein 5-Sekunden-Fenster voraus. Das Fenster von 5 Sekunden gleitet kontinuierlich über die Zeit.
- Für jede Zeile, die in ein Fenster eintritt, wird vom gleitenden Fenster eine Ausgabezeile ausgegeben. Direkt nach dem Start der Anwendung wird Ihnen die Abfrageausgabe für jeden neuen Datensatz angezeigt, der im Stream angezeigt wird, auch wenn ein bestimmtes 5-Sekunden-Fenster noch nicht verstrichen ist. Die Abfrage gibt beispielsweise eine Ausgabe aus, wenn ein Datensatz in der ersten und zweiten Sekunde eintrifft. Später verarbeitet die Abfrage Datensätze im 5-Sekunden-Fenster.
- Die Fenster gleiten mit der Zeit. Wenn ein alter Datensatz im Stream aus dem Fenster herausfällt, gibt die Abfrage keine Ausgabe aus, es sei denn, es gibt auch einen neuen Datensatz im Stream, der in dieses 5-Sekunden-Fenster fällt.

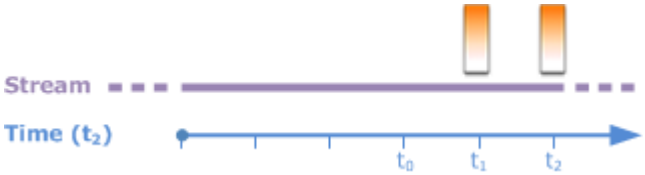
Angenommen, die Abfrage wird zum Zeitpunkt t_0 gestartet. Dann geschieht Folgendes:

1. Zum Zeitpunkt t_0 startet die Abfrage. Die Abfrage gibt keine Ausgabe (Zählwert) aus, da es zu diesem Zeitpunkt keine Datensätze gibt.

2. Zum Zeitpunkt t_1 trifft ein neuer Datensatz im Stream ein und die Abfrage gibt den Zählwert 1 aus.

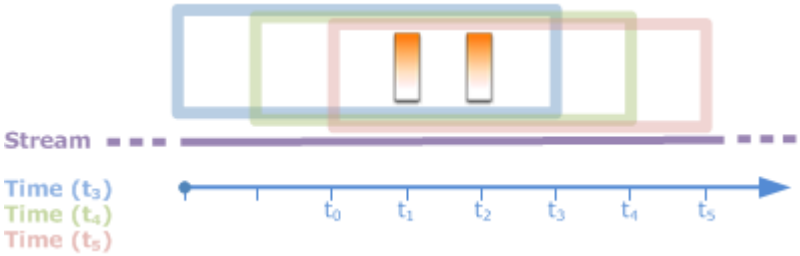


3. Zum Zeitpunkt t_2 trifft ein weiterer Datensatz ein und die Abfrage gibt den Zählwert 2 aus.

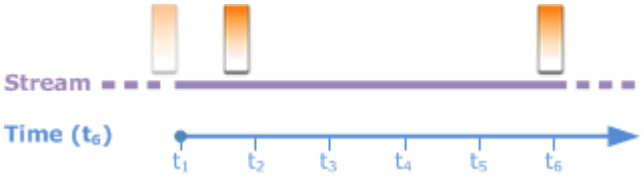


4. Das 5-Sekunden-Fenster gleitet mit der Zeit:
   - Zum Zeitpunkt t_3, das gleitende Fenster von t_3 zu t_0
   - Zum Zeitpunkt t4 (das gleitende Fenster von t_4 zu t_0)
   - Zum Zeitpunkt t_5, das gleitetende Fenster von t_5 zu t_0.

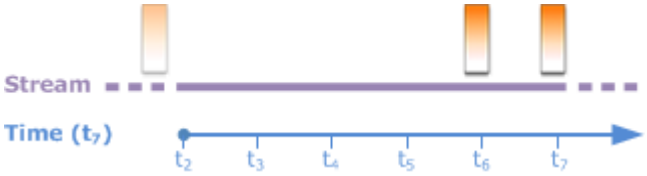
   Zu all diesen Zeitpunkten besitzt das 5-Sekunden-Fenster dieselben Datensätze – es gibt keine neuen Datensätze. Daher gibt die Abfrage keine Ausgabe aus.



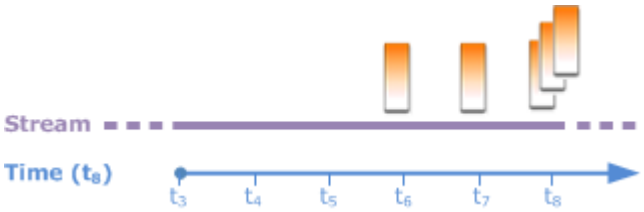
5. Zum Zeitpunkt t_6 liegt das 5-Sekunden-Fenster zwischen t_6 und t_1. Die Abfrage erkennt einen einzelnen neuen Datensatz bei t_6 und gibt daher als Ausgabe 2 aus. Der Datensatz bei t_1 liegt nicht mehr im Fenster und wird nicht gezählt.



6. Zum Zeitpunkt t_7 liegt das 5-Sekunden-Fenster zwischen t_7 und t_2. Die Abfrage erkennt einen einzelnen neuen Datensatz bei t_7 und gibt daher als Ausgabe 2 aus. Der Datensatz liegt zum Zeitpunkt t_2 nicht mehr im 5-Sekunden-Fenster und wird daher nicht gezählt.



7. Zum Zeitpunkt t_8 liegt das 5-Sekunden-Fenster zwischen t_8 und t_3. Die Abfrage erkennt drei neue Datensätze und gibt daher die Datensatzzahl 5 aus.



Zusammenfassend kann gesagt werden, dass das Fenster eine feste Größe hat und mit der Zeit gleitet. Die Abfrage gibt eine Ausgabe aus, wenn neue Datensätze angezeigt werden.

> **Note**
>
> Wir empfehlen, keine gleitenden Fenster zu verwenden, die länger als eine Stunde sind. Wenn Sie ein längeres Fenster verwenden, nimmt der Neustart der Anwendung nach einer regulären Systemwartung mehr Zeit in Anspruch. Dies liegt daran, dass die Quelldaten erneut aus dem Stream gelesen werden müssen.

In den folgenden Beispielabfragen wird die `WINDOW`-Klausel verwendet, um Fenster zu definieren und Zusammenfassungen auszuführen. Da die Abfragen `GROUP BY` nicht angeben, verwendet die Abfrage für die Verarbeitung von Datensätzen im Stream gleitende Fenster.

Beispiel 1: Verarbeiten eines Streams mit einem gleitenden 1-Minuten-Fenster

Betrachten Sie den Demo-Stream in der Einführungsübung, der den In-Application-Stream `SOURCE_SQL_STREAM_001` auffüllt. Im Folgenden wird das Schema gezeigt.

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

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

Angenommen, Ihre Anwendung soll Zusammenfassungen mithilfe eines gleitenden 1-Minuten-Fensters verarbeiten. Das bedeutet, dass Ihre Anwendung für jeden neuen, im Stream angezeigten Datensatz eine Ausgabe ausgeben soll, indem Datensätze im vorhergehenden 1-Minuten-Fenster zusammengefasst werden.

Sie können die folgende Abfrage mit Fenster auf Zeitbasis verwenden. Die Abfrage verwendet die `WINDOW`-Klausel, um das 1-Minuten-Bereichsintervall zu definieren. Die Angabe `PARTITION BY` in der `WINDOW`-Klausel gruppiert Datensätze nach Tickerwerten innerhalb des gleitenden Fensters.

```
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);
```

So testen Sie die Abfrage

1. Richten Sie anhand der Erste Schritte-Übung eine Anwendung ein.
2. Ersetzen Sie die `SELECT`-Anweisung im Anwendungscode durch die vorherige `SELECT`-Abfrage. Der resultierende Anwendungscode wird im Folgenden gezeigt.

   ```
   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);
```

Beispiel 2: Abfrage, die Zusammenfassungen auf ein gleitendes Fenster anwendet

Die folgende Abfrage für den Demo-Stream gibt die durchschnittliche prozentuale Änderung in Bezug auf den Preis der einzelnen Ticker in einem 10-Sekunden-Fenster zurück.

```
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);
```

So testen Sie die Abfrage

1. Richten Sie anhand der Erste Schritte-Übung eine Anwendung ein.
2. Ersetzen Sie die SELECT-Anweisung im Anwendungscode durch die vorherige SELECT-Abfrage. Der resultierende Anwendungscode wird im Folgenden gezeigt.

```
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);
```

Beispiel 3: Abfragen von Daten aus mehreren gleitenden Fenstern für denselben Stream

Sie können Abfragen schreiben, um Ausgaben auszugeben, in denen jeder Spaltenwert mittels unterschiedlicher gleitender Fenster berechnet wird, die für denselben Stream definiert sind.

Im folgenden Beispiel gibt die Abfrage den Ausgabe Ticker, den Preis, a2 und a10 aus. Sie gibt eine Ausgabe für Ticker-Symbole aus, deren gleitender Zwei-Zeilen-Durchschnitt den gleitenden Zehn-Zeilen-Durchschnitt überschreitet. Die Spaltenwerte `a2` und `a10` werden aus Fenstern abgeleitet, die über zwei Zeilen und zehn Zeilen gleiten.

```
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);
```

Um diese Abfrage anhand des Demo-Streams zu testen, befolgen Sie das in Beispiel 1 beschriebene Testverfahren.

Operationen für Streaming-Daten: Zusammenführen von Streams

In Ihrer Anwendung kann es mehrere In-Application-Streams geben. Sie können `JOIN`-Abfragen schreiben, um Daten zu korrelieren, die über diese Streams eintreffen. Angenommen, es liegen die folgenden In-Application-Streams vor:

- OrderStream— Empfängt aufgegebene Lagerbestellungen.

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

- TradeStream— Erhält die daraus resultierenden Aktiengeschäfte für diese Bestellungen.

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

Im Folgenden finden Sie Beispiele für `JOIN`-Abfragen, die Daten aus diesen Streams korrelieren.

Beispiel 1: Bericht zu Bestellungen, bei denen es innerhalb einer Minute nach Platzierung der Bestellung eine Handelsaktion gibt

In diesem Beispiel führt die Abfrage `OrderStream` und `TradeStream` zusammen. Da jedoch nur Handelsaktionen ausgegeben werden sollen, die innerhalb einer Minute nach der Bestellung ausgeführt werden, definiert die Abfrage das 1-Minuten-Fenster für den `TradeStream`. Informationen zu Abfragen mit Fenstern finden Sie unter Gleitende Fenster.

```
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;
```

Sie können die Fenster mittels der `WINDOW`-Klausel explizit definieren und die vorhergehende Abfrage wie folgt schreiben:

```
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)
```

Wenn Sie diese Abfrage in Ihren Anwendungscode einschließen, wird der Anwendungscode kontinuierlich ausgeführt. Für jeden im `OrderStream` eintreffenden Datensatz gibt die Anwendung eine Ausgabe aus, wenn es innerhalb des 1-Minuten-Fensters nach Platzierung der Bestellung eine Handelsaktion gibt.

Die Zusammenführung in der vorhergehenden Abfrage ist eine interne Zusammenführung, bei der die Abfrage Datensätze im `OrderStream` ausgibt, für die es einen übereinstimmenden Datensatz in `TradeStream` gibt (und umgekehrt). Mithilfe einer externen Zusammenführung können Sie ein weiteres interessantes Szenario erstellen. Stellen Sie sich folgendes Szenario vor: Es sollen

Börsenaufträge erteilt werden, für die es innerhalb einer Minute nach Platzierung des Auftrags keine Handelsaktionen gibt. Ebenso gibt es Handelsaktionen innerhalb desselben Fensters, die jedoch für andere Börsenaufträge gelten. In diesem Beispiel wird eine externe Zusammenführung gezeigt.

```
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;
```

Kinesis Data Analytics for SQL

In diesem Abschnitt finden Sie Beispiele für das Erstellen und Arbeiten mit Anwendungen in Amazon-Kinesis-Data-Analytics. Sie enthalten Beispielcode und step-by-step Anweisungen, die Ihnen helfen, Kinesis Data Analytics Analytics-Anwendungen zu erstellen und Ihre Ergebnisse zu testen.

Bevor Sie sich diesen Beispielen zuwenden, sollten Sie zunächst Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's und Erste Schritte mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen lesen.

Themen

Beispiele: Umwandeln von Daten

Es kann vorkommen, dass Ihr Anwendungscode eingehende Datensätze vorverarbeiten muss, bevor in Amazon-Kinesis-Data-Analytics Analysen ausgeführt werden können. Dies kann aus verschiedenen Gründen der Fall sein. Beispielsweise entsprechen die Datensätze möglicherweise nicht den unterstützten Datensatzformaten, was zu nicht normalisierten Spalten in In-Application-Eingabe-Streams führen kann.

Dieser Abschnitt zeigt Beispiele für die Verwendung der verfügbaren Zeichenfolgenfunktionen zur Normalisierung von Daten und für die Extrahierung von Informationen, die Sie aus Zeichenfolgenspalten benötigen, usw. Der Abschnitt stellt darüber hinaus Datum-/Uhrzeitfunktionen bereit, die möglicherweise nützlich für Sie sind.

Vorverarbeiten von Streams mit Lambda

Hinweise zur Vorverarbeitung von Streams mit finden Sie AWS Lambda unterVorverarbeitung von Daten mithilfe einer Lambda-Funktion.

Themen

Beispiele: Umwandeln von Zeichenfolgewerten

Amazon-Kinesis-Data-Analytics unterstützt Formate wie JSON und CSV für Datensätze in einer Streaming-Quelle. Details hierzu finden Sie unter RecordFormat. Diese Datensätze werden anschließend gemäß der Eingabekonfiguration Zeilen in einem In-Application-Stream zugeordnet. Details hierzu finden Sie unter Konfigurieren der Anwendungseingabe. Die Eingabekonfiguration gibt an, wie Datensatzfelder in der Streaming-Quelle Spalten in einem In-Application-Stream zugeordnet werden.

Diese Zuordnung funktioniert, wenn die Datensätze in der Streaming-Quelle die unterstützten Formate berücksichtigen, wodurch es zu einem In-Application-Stream mit normalisierten Daten kommt. Was passiert jedoch, wenn die Daten in Ihrer Streaming-Quelle nicht den unterstützten Standards entsprechen? Was geschieht beispielsweise, wenn Ihre Streaming-Quelle Daten wie z. B. Clickstream-Daten, IoT-Sensoren und Anwendungsprotokolle umfasst?

Berücksichtigen Sie die folgenden Beispiele:

- Streaming-Quelle enthält Anwendungsprotokolle– Die Anwendungsprotokolle befolgen das standardmäßige Apache-Protokollformat und werden im JSON-Format in den Stream geschrieben.

  ```
  {
     "Log":"192.168.254.30 - John [24/May/2004:22:01:02 -0700] "GET /icons/
  apache_pb.gif HTTP/1.1" 304 0"
  }
  ```

  Weitere Informationen zum standardmäßigen Apache-Protokollformat finden Sie unter Log Files auf der Apache-Website.

- Streaming-Quelle enthält teilweise strukturierte Daten – Im folgenden Beispiel sind zwei Datensätze zu sehen. Der Wert des Feldes `Col_E_Unstructured` besteht aus einer Reihe von durch Kommas getrennten Werten. Es gibt fünf Spalten: die ersten vier haben Werte vom Typ "Zeichenfolge" und die letzte Spalte enthält durch Kommas getrennte Werte.

```
{ "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"}
```

- Die Aufzeichnungen in Ihrer Streaming-Quelle enthalten URLs, und Sie benötigen einen Teil des URL-Domainnamens für Analysen.

```
{ "referrer" : "http://www.amazon.com"}
{ "referrer" : "http://www.stackoverflow.com" }
```

In diesen Fällen lassen sich mit dem folgenden zwei Schritte umfassenden Prozess in der Regel In-Application-Streams mit normalisierten Daten erstellen:

1. Konfigurieren Sie die Anwendungseingabe, um das unstrukturierte Feld einer Spalte des Typs `VARCHAR(N)` in dem erstellten In-Application-Eingabe-Stream zuzuordnen.
2. Verwenden Sie in Ihrem Anwendungscode Zeichenfolgefunktionen, um diese einzelne Spalte in mehrere Spalten aufzuteilen, und speichern Sie die Zeilen anschließend in einem anderen In-Application-Stream. Dieser über Ihren Anwendungscode erstellte In-Application-Stream enthält normalisierte Daten. Sie können dann an diesem In-Application-Stream Analysen durchführen.

Amazon-Kinesis-Data-Analytics bietet die folgenden Zeichenfolgeoperationen, SQL-Standardfunktionen und Erweiterungen des SQL-Standards für die Arbeit mit Zeichenfolgespalten:

- Zeichenfolgeoperatoren – Operatoren wie `LIKE` und `SIMILAR` sind hilfreich für den Vergleich von Zeichenfolgen. Weitere Informationen finden Sie unter String-Operatoren in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.
- SQL-Funktionen – Die folgenden Funktionen sind bei der Bearbeitung einzelner Zeichenfolgen hilfreich. Weitere Informationen finden Sie unter Zeichenketten- und Suchfunktionen in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

- `CHAR_LENGTH` – Gibt die Länge einer Zeichenfolge an.
- `INITCAP` – Gibt eine konvertierte Version der Eingabezeichenfolge zurück: jeder Anfangsbuchstabe eines Worts – Leerzeichen als Trennzeichen – ist ein Großbuchstabe, die restlichen Buchstaben sind Kleinbuchstaben.
- `LOWER/UPPER` – Konvertiert eine Zeichenfolge in Klein- oder Großbuchstaben.
- `OVERLAY` – Ersetzt einen Teil der ersten Zeichenfolgenarguments (die ursprüngliche Zeichenfolge) mit dem zweiten Zeichenfolgenargument (die Ersetzungszeichenfolge).
- `POSITION` – Sucht nach einer Zeichenfolge in einer anderen Zeichenfolge.
- `REGEX_REPLACE` – Ersetzt eine Teilzeichenfolge durch eine andere Teilzeichenfolge.
- `SUBSTRING` – Extrahiert einen Teil einer Quell-Zeichenfolge ab einer bestimmten Position.
- `TRIM` – Entfernt Instances des angegebenen Zeichens ab dem Anfang oder Ende der Quellzeichenfolge.

- SQL-Erweiterungen — Diese sind nützlich für die Arbeit mit unstrukturierten Zeichenketten wie Logs und URIs. Weitere Informationen finden Sie unter Funktionen zum Analysieren von Protokollen in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.
  - `FAST_REGEX_LOG_PARSER` – Arbeitet ähnlich wie der regex-Parser, nimmt jedoch einige „Abkürzungen“, um schneller Ergebnisse bereitzustellen. Beispielsweise stoppt der schnelle Parser für reguläre Ausdrücke bei der ersten ermittelten Übereinstimmung (auch als träge Semantik bekannt).
  - `FIXED_COLUMN_LOG_PARSE` – Analysiert Felder mit fester Breite und konvertiert diese automatisch in die gegebenen SQL-Typen.
  - `REGEX_LOG_PARSE` – Analysiert eine Zeichenfolge basierend auf regulären Java-Standardausdrücken.
  - `SYS_LOG_PARSE`— Analysiert Einträge, die häufig in UNIX/Linux Systemprotokollen zu finden sind.
  - `VARIABLE_COLUMN_LOG_PARSE` – Teilt eine Eingabezeichenfolge in durch ein Trennzeichen oder eine Trennzeichenfolge getrennte Felder.
  - `W3C_LOG_PARSE` – Kann zum schnellen Formatieren von Apache-Protokollen verwendet werden.

Beispiele für die Verwendung dieser Funktionen finden Sie in den folgenden Themen:

Themen

- Beispiel: Extrahieren eines Teils einer Zeichenfolge (Funktion SUBSTRING)
- Beispiel: Ersetzen einer Teilzeichenfolge mit Regex (Funktion REGEX_REPLACE)
- Beispiel: Analysieren von Protokollzeichenfolgen basierend auf regulären Ausdrücken (Funktion REGEX_LOG_PARSE)
- Beispiel: Analysieren von Web-Protokolle (Funktion W3C_LOG_PARSE)
- Beispiel: Aufteilen von Zeichenfolgen auf mehrerer Felder (Funktion VARIABLE_COLUMN_LOG_PARSE)

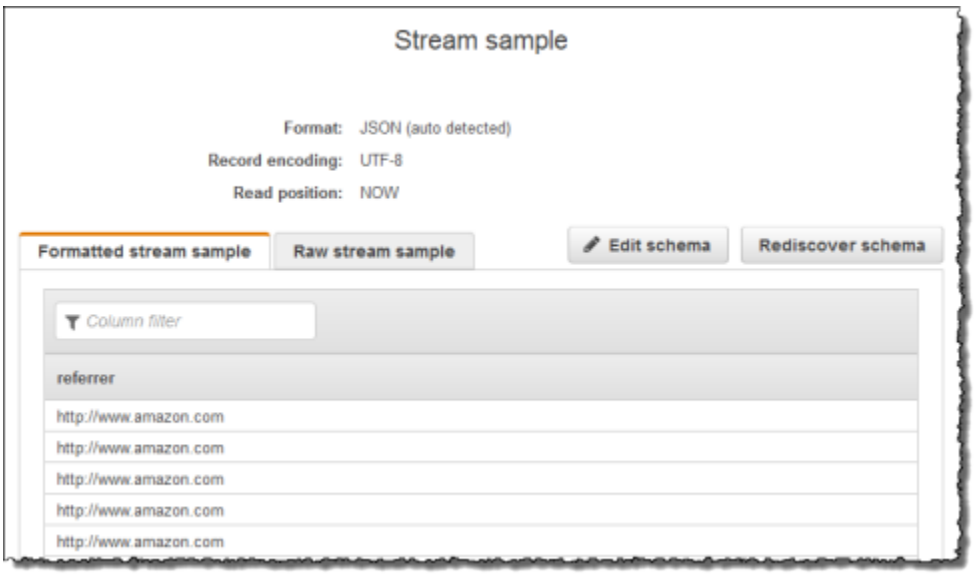
Beispiel: Extrahieren eines Teils einer Zeichenfolge (Funktion SUBSTRING)

In diesem Beispiel wird die Funktion `SUBSTRING` zum Umwandeln einer Zeichenfolge in Amazon-Kinesis-Data-Analytics verwendet. Die Funktion `SUBSTRING` extrahiert einen Teil einer Quell-Zeichenfolge ab einer bestimmten Position. Weitere Informationen finden Sie unter SUBSTRING in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

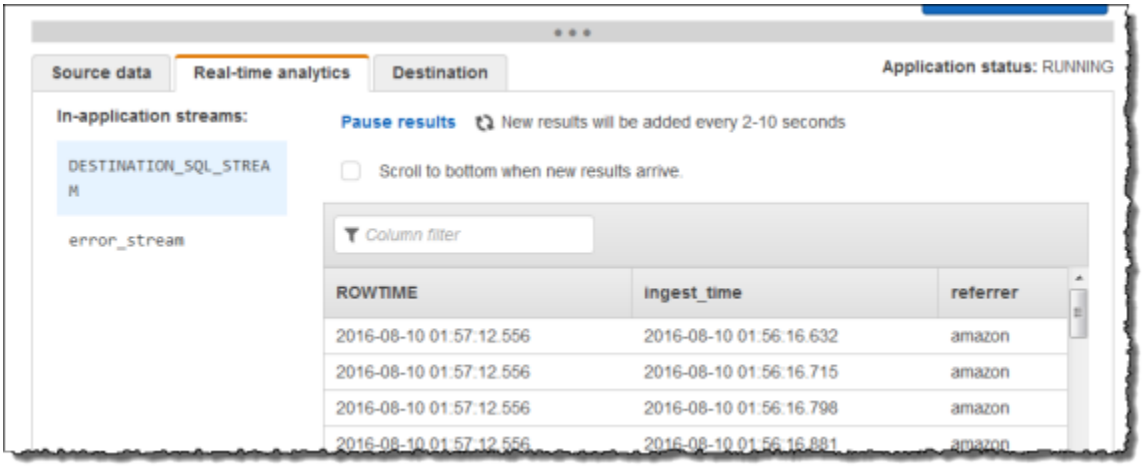
In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Amazon-Kinesis-Datenstrom.

```
{ "REFERRER" : "http://www.amazon.com" }
{ "REFERRER" : "http://www.amazon.com"}
{ "REFERRER" : "http://www.amazon.com"}
...
```

Anschließend erstellen Sie eine Kinesis Data Analytics-Anwendung in der Konsole mit dem Kinesis-Datenstrom als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze auf der Streaming-Quelle und erschließt ein In-Application-Schema mit einer Spalte (`REFERRER`), wie hier gezeigt.



Anschließend verwenden Sie die Anwendungscode mit der Funktion `SUBSTRING` zum Analysieren der URL-Zeichenfolge, um den Namen des Unternehmens abzurufen. Fügen Sie die resultierenden Daten dann wie nachstehend dargestellt in einen anderen In-Application-Stream ein:



Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Protokolldatensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Führen Sie den folgenden Python-Code aus, um Beispiel-Protokolldatensätze zu füllen. Dieser einfache Code schreibt kontinuierlich denselben Protokolldatensatz in den Stream.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie als Nächstes wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com

2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden).
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.

   b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.

   c. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema weist nur eine Spalte auf.

   d. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
-- 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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

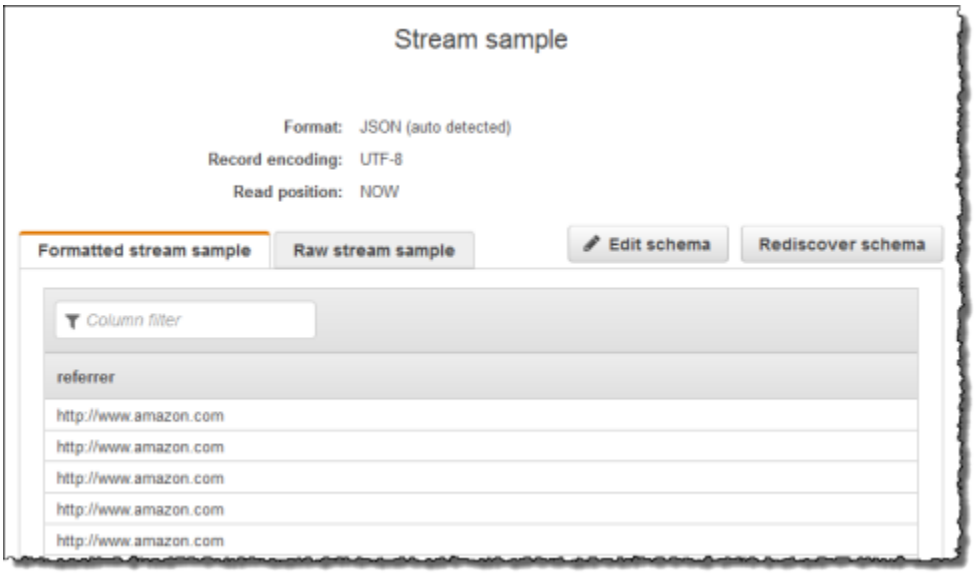
Beispiel: Ersetzen einer Teilzeichenfolge mit Regex (Funktion REGEX_REPLACE)

In diesem Beispiel wird die Funktion `REGEX_REPLACE` zum Umwandeln einer Zeichenfolge in Amazon-Kinesis-Data-Analytics verwendet. `REGEX_REPLACE` ersetzt eine Teilzeichenfolge durch eine andere Teilzeichenfolge. Weitere Informationen finden Sie unter REGEX_REPLACE in der SQL-Referenz Amazon-Managed-Service für Apache Flink.

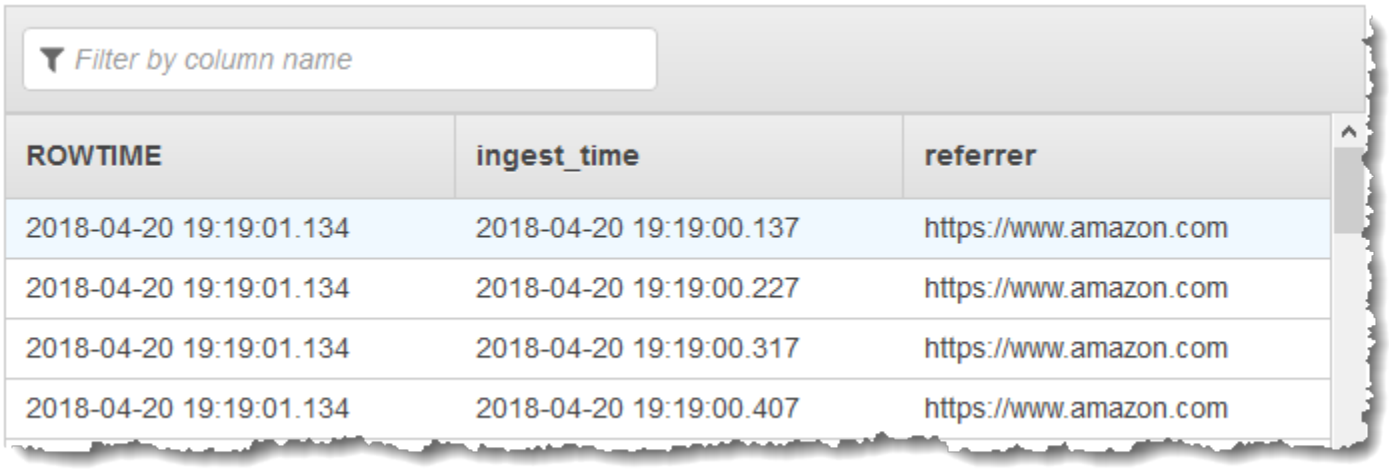
In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Amazon-Kinesis-Datenstrom:

```
{ "REFERRER" : "http://www.amazon.com" }
{ "REFERRER" : "http://www.amazon.com"}
{ "REFERRER" : "http://www.amazon.com"}
...
```

Anschließend erstellen Sie eine Kinesis Data Analytics-Anwendung in der Konsole mit dem Kinesis-Datenstrom als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze auf der Streaming-Quelle und erschließt ein In-Application-Schema mit einer Spalte (REFERRER), wie hier gezeigt.



Anschließend verwenden Sie den Anwendungscode mit der Funktion `REGEX_REPLACE`, um die URL so umzuwandeln, dass sie `https://` anstelle von `http://` verwendet. Sie fügen Sie die resultierenden Daten dann wie nachstehend dargestellt in einen anderen In-Application-Stream ein:



Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Protokolldatensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Führen Sie den folgenden Python-Code aus, um die Beispiel-Protokolldatensätze zu füllen. Dieser einfache Code schreibt kontinuierlich denselben Protokolldatensatz in den Stream.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie als Nächstes wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden).
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:
    a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.
    b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.
    c. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema weist nur eine Spalte auf.
    d. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

    a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein:

    ```
    -- 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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

Beispiel: Analysieren von Protokollzeichenfolgen basierend auf regulären Ausdrücken (Funktion REGEX_LOG_PARSE)

In diesem Beispiel wird die Funktion `REGEX_LOG_PARSE` zum Umwandeln einer Zeichenfolge in Amazon-Kinesis-Data-Analytics verwendet. `REGEX_LOG_PARSE` analysiert eine Zeichenfolge basierend auf regulären Java-Standardausdrücken. Weitere Informationen finden Sie unter REGEX_LOG_PARSE in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Amazon-Kinesis-Stream:

```
{"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\""}
...
```

Anschließend erstellen Sie eine Kinesis Data Analytics-Anwendung in der Konsole mit dem Kinesis-Datenstrom als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze auf der Streaming-Quelle und erschließt ein In-Application-Schema mit einer Spalte (LOGENTRY), wie im Folgenden gezeigt:

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"

Anschließend verwenden Sie den Anwendungscode mit der Funktion `REGEX_LOG_PARSE` zum Analysieren der Protokollzeichenfolge, um die Datenelemente abzurufen. Fügen Sie die resultierenden Daten wie im folgenden Screenshot dargestellt in einen anderen In-Application-Stream ein:

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 [

Themen

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Protokolldatensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Führen Sie den folgenden Python-Code aus, um Beispiel-Protokolldatensätze zu füllen. Dieser einfache Code schreibt kontinuierlich denselben Protokolldatensatz in den Stream.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie als Nächstes wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Wählen Sie Create application aus und geben Sie einen Anwendungsnamen an.
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden).
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:
   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.
   b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.
   c. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema weist nur eine Spalte auf.
   d. Wählen Sie Save and continue aus.
5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:
   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

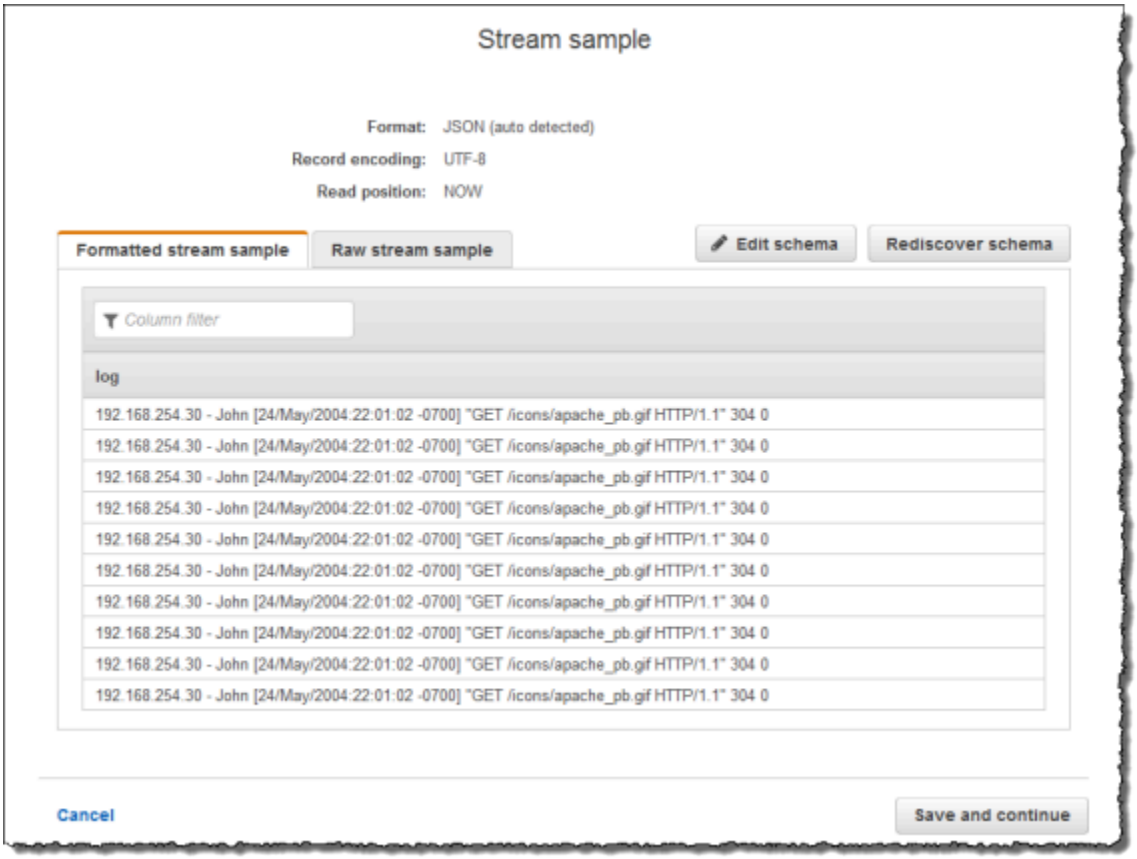
Beispiel: Analysieren von Web-Protokolle (Funktion W3C_LOG_PARSE)

In diesem Beispiel wird die Funktion `W3C_LOG_PARSE` zum Umwandeln einer Zeichenfolge in Amazon-Kinesis-Data-Analytics verwendet. Sie können `W3C_LOG_PARSE` verwenden, um Apache-Protokolle schnell zu formatieren. Weitere Informationen finden Sie unter W3C_LOG_PARSE in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

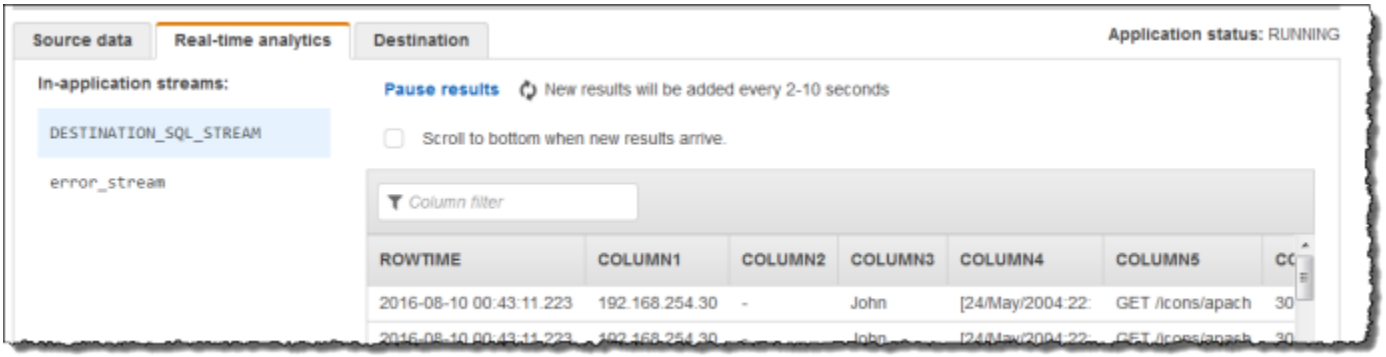
In diesem Beispiel schreiben Sie Protokolldatensätze in einen Amazon-Kinesis-Datenstrom. Beispielprotokolle werden nachfolgend angezeigt:

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

Anschließend erstellen Sie eine Kinesis Data Analytics-Anwendung in der Konsole mit dem Kinesis-Datenstrom als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze auf der Streaming-Quelle und erschließt ein In-Application-Schema mit einer Spalte (Protokoll), wie im Folgenden gezeigt:



Anschließend verwenden Sie den Anwendungscode mit der Funktion `W3C_LOG_PARSE` zum Analysieren des Protokolls und erstellen einen anderen In-Application-Stream mit verschiedenen Protokollfeldern in separaten Spalten, wie im Folgenden gezeigt:



Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Protokolldatensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Führen Sie den folgenden Python-Code aus, um die Beispiel-Protokolldatensätze zu füllen. Dieser einfache Code schreibt kontinuierlich denselben Protokolldatensatz in den Stream.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden).
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:
   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.
   b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.
   c. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema weist nur eine Spalte auf.
   d. Wählen Sie Save and continue aus.
5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:
   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

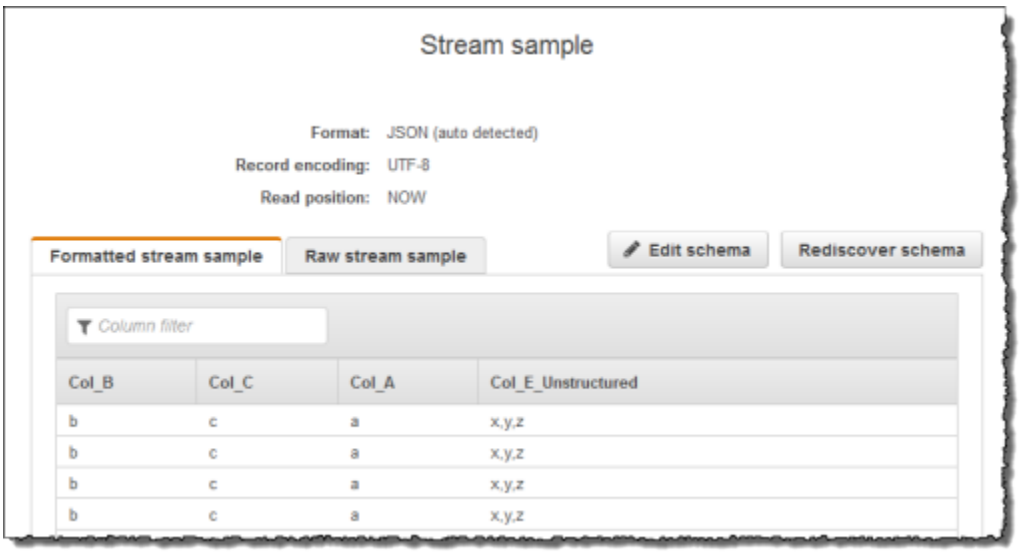
Beispiel: Aufteilen von Zeichenfolgen auf mehrerer Felder (Funktion VARIABLE_COLUMN_LOG_PARSE)

Dieses Beispiel verwendet die Funktion `VARIABLE_COLUMN_LOG_PARSE` zum Bearbeiten von Zeichenfolgen in Kinesis Data Analytics. `VARIABLE_COLUMN_LOG_PARSE` teilt eine Eingabezeichenfolge in durch ein Trennzeichen oder eine Trennzeichenfolge getrennte Felder. Weitere Informationen finden Sie unter VARIABLE_COLUMN_LOG_PARSE in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

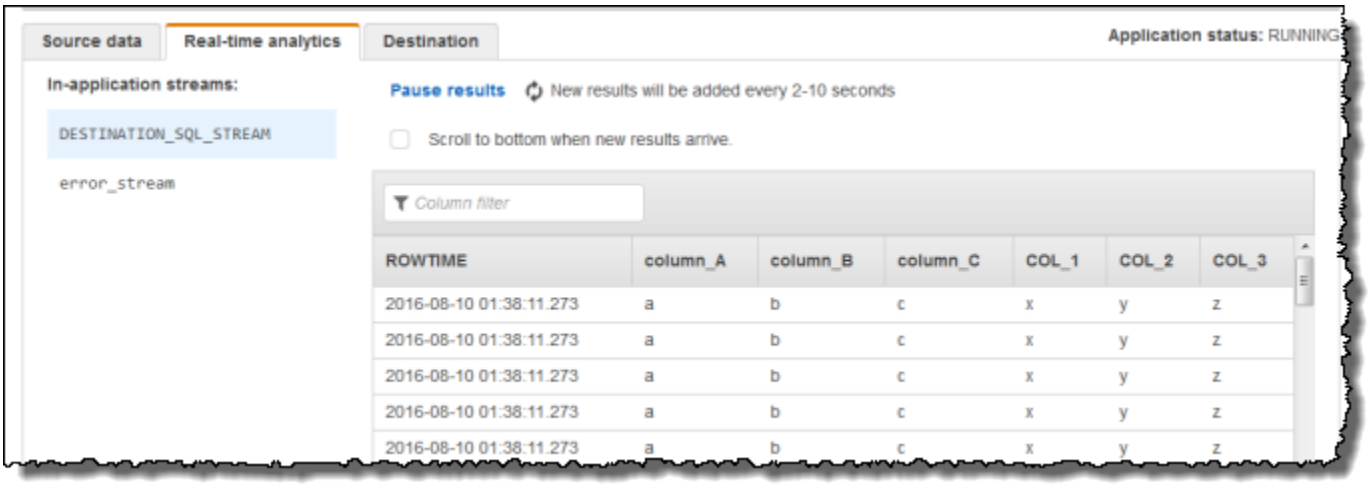
In diesem Beispiel schreiben Sie semistrukturierte Datensätze in einen Amazon-Kinesis-Datenstrom. Die Beispieldatensätze sind wie folgt:

```
{ "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"}
```

Anschließend erstellen Sie eine Kinesis Data Analytics-Anwendung in der Konsole mit dem Kinesis Stream als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze auf der Streaming-Quelle und erschließt ein In-Application-Schema mit vier Spalten, wie im Folgenden gezeigt:



Anschließend verwenden Sie den Anwendungscode mit der Funktion `VARIABLE_COLUMN_LOG_PARSE` zum Analysieren der durch Komma getrennten Werte und fügen normalisierte Zeilen in einen anderen In-Application-Stream ein, wie im Folgenden gezeigt:



Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Protokolldatensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Führen Sie den folgenden Python-Code aus, um die Beispiel-Protokolldatensätze zu füllen. Dieser einfache Code schreibt kontinuierlich denselben Protokolldatensatz in den Stream.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden).
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.

   b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.

   c. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Beachten Sie, dass das abgeleitete Schema nur eine Spalte aufweist.

   d. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie den Anwendungscode im SQL-Editor und überprüfen Sie die Ergebnisse wie folgt:

   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein:

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

Beispiel: Werte transformieren DateTime

Amazon-Kinesis-Data-Analytics unterstützt die Konvertierung von Spalten in Zeitstempel. Beispielsweise möchten Sie vielleicht Ihren eigenen Zeitstempel als Teil einer `GROUP BY`-Klausel in Form eines anderen zeitbasierten Fensters zusätzlich zur Spalte `ROWTIME` verwenden. Kinesis Data Analytics stellt Vorgänge und SQL-Funktionen für die Arbeit mit Datums- und Uhrzeitfeldern bereit.

- Operatoren für Datum und Uhrzeit – Sie können Rechenoperationen mit Datums-, Uhrzeit- und Intervall-Datentypen ausführen. Weitere Informationen finden Sie unter Datums-, Zeitstempel- und Intervalloperatoren in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

- SQL-Funktionen – Diese umfassen u. a. das Folgende. Weitere Informationen finden Sie unter Datums- und Uhrzeitfunktionen in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.
  - `EXTRACT()` – Extrahiert ein Feld von einem Datums-, Uhrzeit-, Zeitstempel- oder Intervall Ausdruck.
  - `CURRENT_TIME` – Gibt die Zeit an, zu der die Abfrage ausgeführt wird (UTC).
  - `CURRENT_DATE` – Gibt das Datum an, an dem die Abfrage ausgeführt wird (UTC).
  - `CURRENT_TIMESTAMP` – Gibt den Zeitstempel an, an dem die Abfrage ausgeführt wird (UTC).
  - `LOCALTIME` – Gibt die aktuelle Uhrzeit an, wenn die Abfrage ausgeführt wird, entsprechend der Definition der Umgebung, in der Kinesis Data Analytics ausgeführt wird (UTC).
  - `LOCALTIMESTAMP` – Gibt den aktuellen Zeitstempel an, entsprechend der Definition der Umgebung, in der Kinesis Data Analytics ausgeführt wird (UTC).

- SQL-Erweiterungen – Diese umfassen u. a. die Folgenden. Weitere Informationen finden Sie unter Datums- und Zeitfunktionen und Konvertierungsfunktionen für Datum/Uhrzeit in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.
  - `CURRENT_ROW_TIMESTAMP` – Gibt einen neuen Zeitstempel für alle Zeilen im Stream an.
  - `TSDIFF` – Gibt die Differenz zwischen zwei Zeitstempeln in Millisekunden an.
  - `CHAR_TO_DATE` – Wandelt eine Zeichenfolge in ein Datum um.
  - `CHAR_TO_TIME` – Wandelt eine Zeichenfolge in eine Uhrzeit um.
  - `CHAR_TO_TIMESTAMP` – Wandelt eine Zeichenfolge in einen Zeitstempel um.
  - `DATE_TO_CHAR` – Wandelt ein Datum in eine Zeichenfolge um.
  - `TIME_TO_CHAR` – Wandelt eine Uhrzeit in eine Zeichenfolge um.
  - `TIMESTAMP_TO_CHAR` – Wandelt eine Zeitstempel in einen Zeichenfolge um.

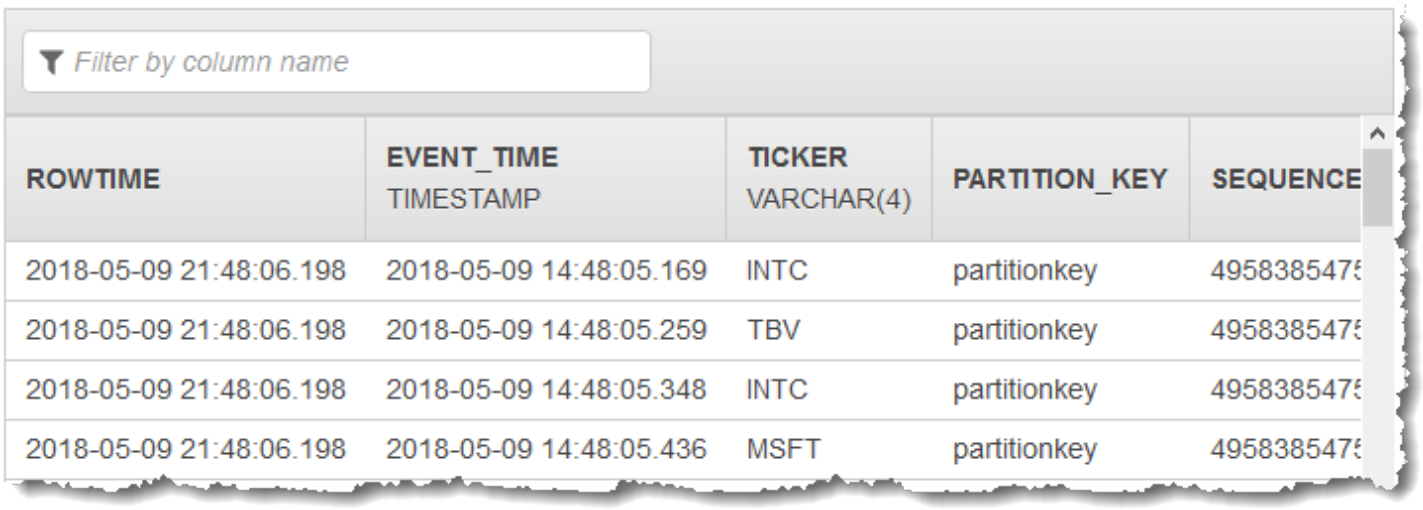
Die meisten der oben genannten SQL-Funktionen verwenden ein Format zum Umwandeln der Spalten. Das Format ist flexibel. Sie können beispielsweise das Format `yyyy-MM-dd hh:mm:ss` festlegen, damit die Eingabezeichenfolge `2009-09-16 03:15:24` in einen Zeitstempel umgewandelt wird. Weitere Informationen finden Sie unter Char To Timestamp (Sys) in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

Beispiele: Umwandeln von Datumsangaben

In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Amazon-Kinesis-Datenstrom.

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

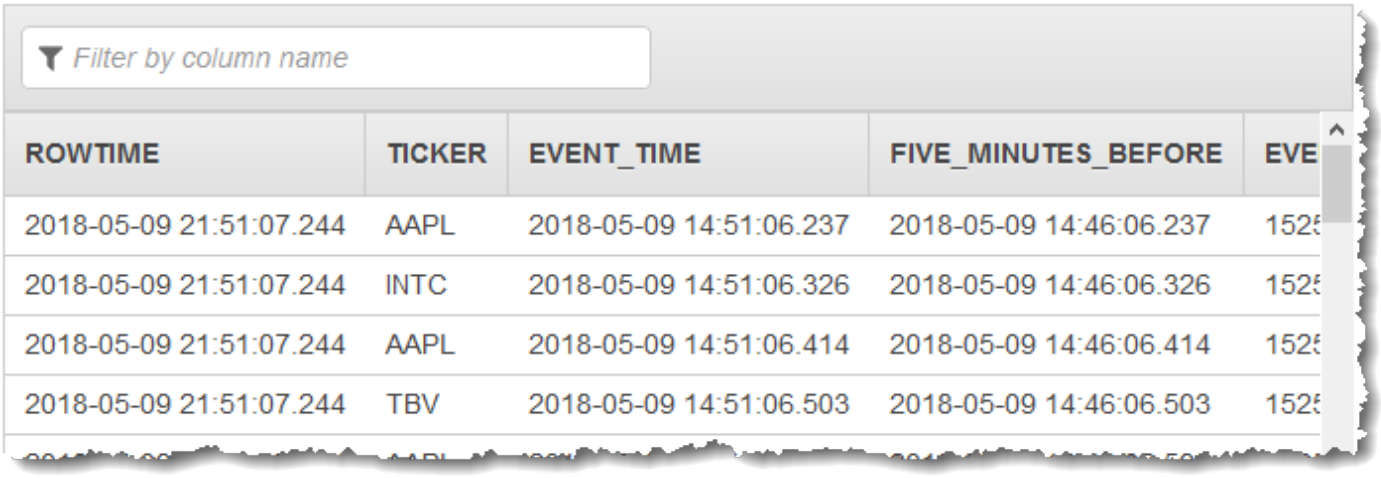
Anschließend erstellen Sie eine Kinesis Data Analytics-Anwendung in der Konsole mit dem Kinesis-Stream als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze auf der Streaming-Quelle und erschließt ein In-Application-Schema mit zwei Spalten (`EVENT_TIME` und `TICKER`), wie hier gezeigt.



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

Anschließend verwenden Sie den Anwendungscode mit SQL-Funktionen, um das `EVENT_TIME`-Zeitstempelfeld auf verschiedene Weise zu konvertieren. Fügen Sie die resultierenden Daten dann wie im folgenden Screenshot dargestellt in einen anderen In-Application-Stream ein:



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

Schritt 1: Erstellen Sie einen Kinesis-Datenstrom

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie ihn folgendermaßen mit Ereigniszeit- und Ticker-Datensätzen:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).

3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard.
4. Führen Sie den folgenden Python-Code aus, um den Stream mit Beispieldaten zu füllen. Dieser einfache Code schreibt kontinuierlich einen Datensatz mit einem zufälligen Tickersymbol und dem aktuellen Zeitstempel in den Stream.

```
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"))
```

Schritt 4: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie eine Anwendung wie folgt:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden), um eine Verbindung mit der Quelle herzustellen.
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.

   b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.

   c. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema verfügt über zwei Spalten.

   d. Klicken Sie auf Edit schema (Schema bearbeiten). Ändern Sie den Wert für Column type (Spaltentyp) der Spalte EVENT_TIME in `TIMESTAMP`.

   e. Wählen Sie Save schema and update stream samples (Schema speichern und Stream-Beispiel aktualisieren). Nachdem die Konsole das Schema gespeichert hat, klicken Sie auf Exit (Beenden).

   f. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

Beispiel: Umwandeln von mehreren Datentypen

Eine gängige Anforderung in ETL-Anwendungen (Extrahieren, Transformieren und Laden) ist die Verarbeitung von mehreren Datensatztypen für eine Streaming-Quelle. Sie können Kinesis Data Analytics-Anwendungen erstellen, um diese Arten von Streaming-Quellen zu verarbeiten. Der Prozess läuft folgendermaßen ab:

1. Zunächst ordnen Sie die Streaming-Quelle einem In-Application-Eingabe-Stream zu, ähnlich wie bei allen anderen Kinesis Data Analytics-Anwendungen.
2. Anschließend schreiben Sie in Ihrem Anwendungscode SQL-Anweisungen, um Zeilen bestimmter Typen aus dem In-Application-Eingabe-Stream abzurufen. Sie fügen diese dann in separate In-Application-Streams ein. (Sie können zusätzliche In-Application-Streams in Ihrem Anwendungscode erstellen.)

In dieser Übung verfügen Sie über eine Streaming-Quelle, die Datensätze von zwei Typen (`Order` und `Trade`) empfängt. Dies sind Börsenbestellungen und die entsprechenden Abschlüsse. Bei jeder Bestellung kann es null oder mehr Abschlüsse geben. Beispieldatensätze für jeden Typ werden im Folgenden gezeigt:

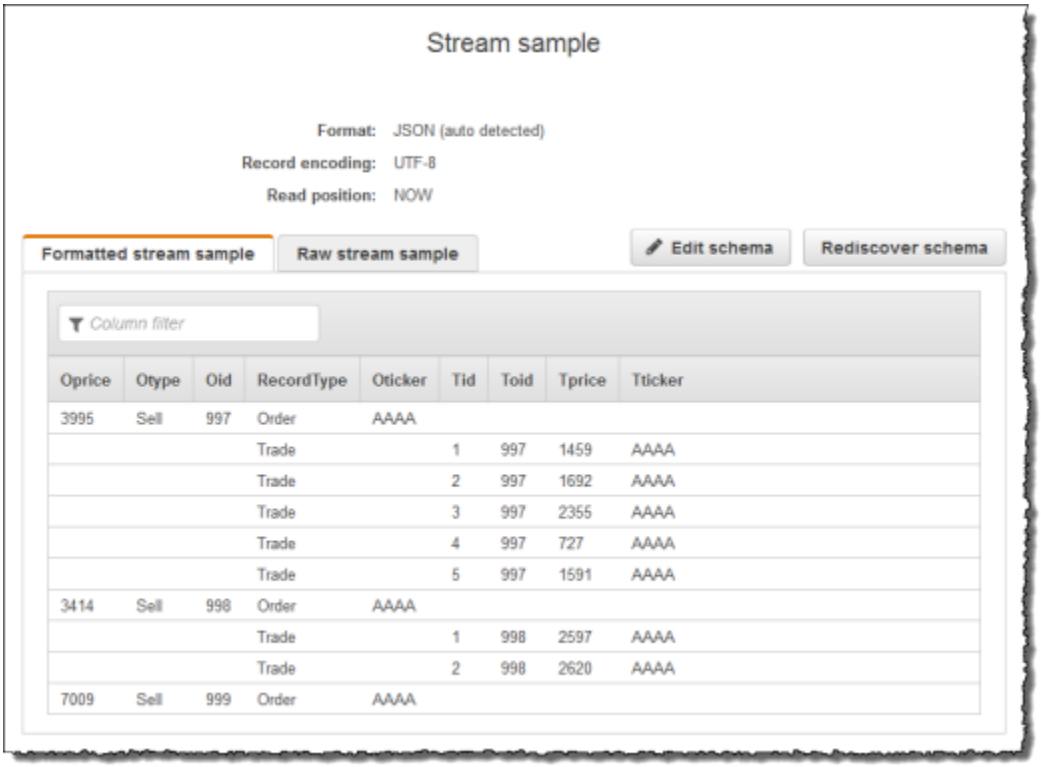
Order record

```
{"RecordType": "Order", "Oprice": 9047, "Otype": "Sell", "Oid": 3811, "Oticker":
 "AAAA"}
```

Trade record

```
{"RecordType": "Trade", "Tid": 1, "Toid": 3812, "Tprice": 2089, "Tticker": "BBBB"}
```

Wenn Sie eine Anwendung mit dem erstellen AWS-Managementkonsole, zeigt die Konsole das folgende abgeleitete Schema für den erstellten anwendungsinternen Eingabestream an. Standardmäßig benennt die Konsole diesen In-Application-Stream mit `SOURCE_SQL_STREAM_001`.



Beim Speichern der Konfiguration liest Amazon-Kinesis-Data-Analytics kontinuierlich Daten aus der Streaming-Quelle und fügt Zeilen in den In-Application-Stream ein. Sie können jetzt Analysen von Daten im In-Application-Stream durchführen.

Im Beispielcode in diesem Beispiel erstellen Sie zuerst zwei zusätzliche In-Application-Streams, `Order_Stream` und `Trade_Stream`. Anschließend filtern Sie basierend auf dem Datensatztyp Zeilen aus dem `SOURCE_SQL_STREAM_001`-Stream und fügen diese mithilfe von Pumps in die neu erstellten Streams ein. Weitere Informationen zu diesem Codierungsmuster finden Sie unter Anwendungscode.

1. Filtern der Zeilen zu Bestellung und Abschluss in separate In-Application-Streams:

   a. Filtern Sie die Datensätze zu Bestellung in `SOURCE_SQL_STREAM_001` und speichern Sie die Bestellungen in `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. Filtern Sie die Datensätze zu Abschlüssen in `SOURCE_SQL_STREAM_001` und speichern Sie die Bestellungen in `Trade_Stream`.

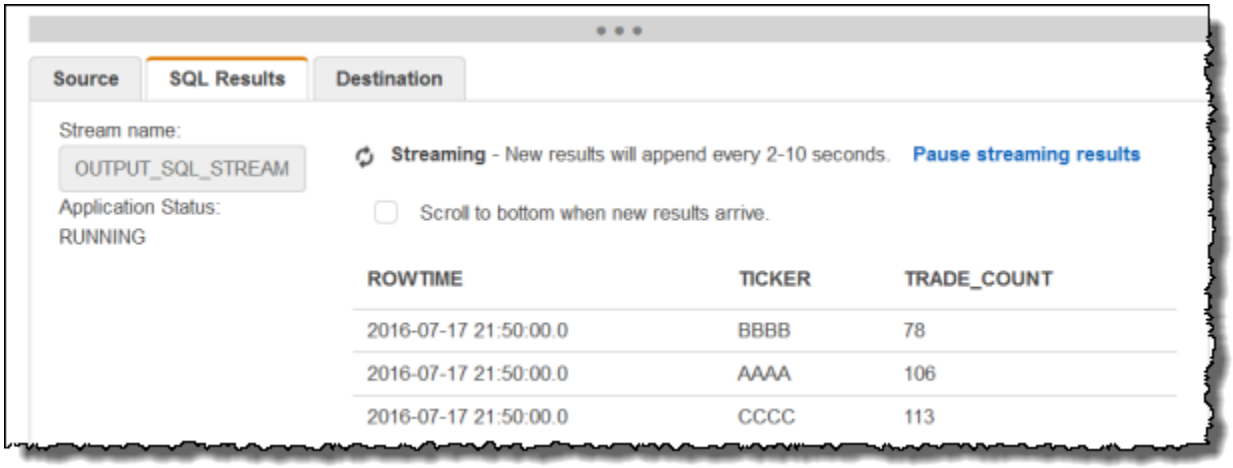
```
--Create Trade_Stream.
CREATE OR REPLACE STREAM "Trade_Stream"
           (trade_id     integer,
            order_id     integer,
            trade_price  DOUBLE,
            ticker       varchar(4),
            record_type  varchar(10)
            );

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. Jetzt können Sie zusätzliche Analysen zu diesen Streams durchführen. In diesem Beispiel wird die Anzahl der Abschlüsse nach Ticker in einem rollierenden Fenster von einer Minute gezählt. Die Ergebnisse werden in einem anderen Stream, `DESTINATION_SQL_STREAM`, gespeichert.

```
--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);
```

Das Ergebnis sieht folgendermaßen aus:



Themen

- Schritt 1: Vorbereitung der Daten
- Schritt 2: Erstellen einer -Anwendung

Nächster Schritt

Schritt 1: Vorbereitung der Daten

Schritt 1: Vorbereitung der Daten

In diesem Abschnitt erstellen Sie einen Kinesis-Datenstrom, und füllen die Datensätze zu Bestellungen und Käufen im Stream. Dies ist die Streaming-Quelle für die Anwendung, die Sie im nächsten Schritt erstellen.

Themen

Schritt 1.1: Erstellen einer Streaming-Quelle

Sie können einen Kinesis-Datenstrom mithilfe der Konsole oder mit AWS CLI erstellen. In diesem Beispiel wird `OrdersAndTradesStream` als Stream-Name angenommen.

- Konsole verwenden — Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis. Wählen Sie Data Streams aus und erstellen Sie anschließend einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
- Verwenden von AWS CLI — Verwenden Sie den folgenden `create-stream` AWS CLI Kinesis-Befehl, um den Stream zu erstellen:

```
$ aws kinesis create-stream \
--stream-name OrdersAndTradesStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

Schritt 1.2: Ausfüllen der Streaming-Quelle

Führen Sie das folgenden Python-Skript aus, um Beispiel-Protokolldatensätze im `OrdersAndTradesStream` zu füllen. Wenn Sie den Stream mit einem anderen Namen erstellt haben, aktualisieren Sie den Python-Code dementsprechend.

1. Installieren Sie Python und `pip`.

   Informationen zur Installation von Python finden Sie auf der Website für Python.

   Sie können mithilfe von pip Abhängigkeiten installieren. Informationen zur Installation von pip finden Sie unter Installation auf der Website für pip.
2. Führen Sie den folgenden Python-Code aus. Der Befehl `put-record` im Code schreibt die JSON-Datensätze zum Stream.

```
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"))
```

Nächster Schritt

Schritt 2: Erstellen einer -Anwendung

Schritt 2: Erstellen einer -Anwendung

In diesem Abschnitt erstellen Sie eine Kinesis Data Analytics-Anwendung. Anschließend aktualisieren Sie die Anwendung, indem Sie Eingabekonfiguration hinzufügen, die die im vorigen Abschnitt erstellte Streaming-Quelle einem In-Application-Eingabe-Stream zuordnet.

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter https://console.aws.amazon.com /kinesisanalytics.
2. Wählen Sie Create application aus. Dieses Beispiel verwendet als Anwendungsnamen **`ProcessMultipleRecordTypes`**.
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden), um eine Verbindung mit der Quelle herzustellen.
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:
   a. Wählen Sie den Stream aus, den Sie in Schritt 1: Vorbereitung der Daten erstellt haben.
   b. Wählen Sie die Option zum Erstellen einer IAM-Rolle.
   c. Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden.
   d. Wählen Sie Save and continue aus.
5. Wählen Sie im Anwendungs-Hub Go to SQL editor aus. Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie den Anwendungscode im SQL-Editor und überprüfen Sie die Ergebnisse wie folgt:
   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
--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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen). Wählen Sie die Registerkarte Real-time analytics (Echtzeitanalyse), um alle In-Application-Streams anzuzeigen, die von der Anwendung erstellt wurden, und die Daten zu überprüfen.

Nächster Schritt

Sie können die Anwendungsausgabe so konfigurieren, dass Ergebnisse an einem externen Ziel, z. B. einem anderen Kinesis-Stream oder einem Firehose-Datenlieferstrom, gespeichert werden.

Beispiele: Fenster und Aggregation

In diesem Abschnitt finden Sie Beispiele für Amazon-Kinesis-Data-Analytics-Anwendungen mit Fenster- und Aggregationsabfragen. (Weitere Informationen finden Sie unter Abfragen mit Fenstern.) Jedes Beispiel enthält step-by-step Anweisungen und Beispielcode für die Einrichtung der Kinesis Data Analytics Analytics-Anwendung.

Themen

- Beispiel: Versetztes Fenster
- Beispiel: Rollierendes Fenster mit ROWTIME
- Beispiel: Rollierendes Fenster mit einem Ereignis-Zeitstempel
- Beispiel: Abrufen der am häufigsten auftretenden Werte (TOP_K_ITEMS_TUMBLING)
- Beispiel: Aggregieren von Teilergebnissen aus einer Abfrage

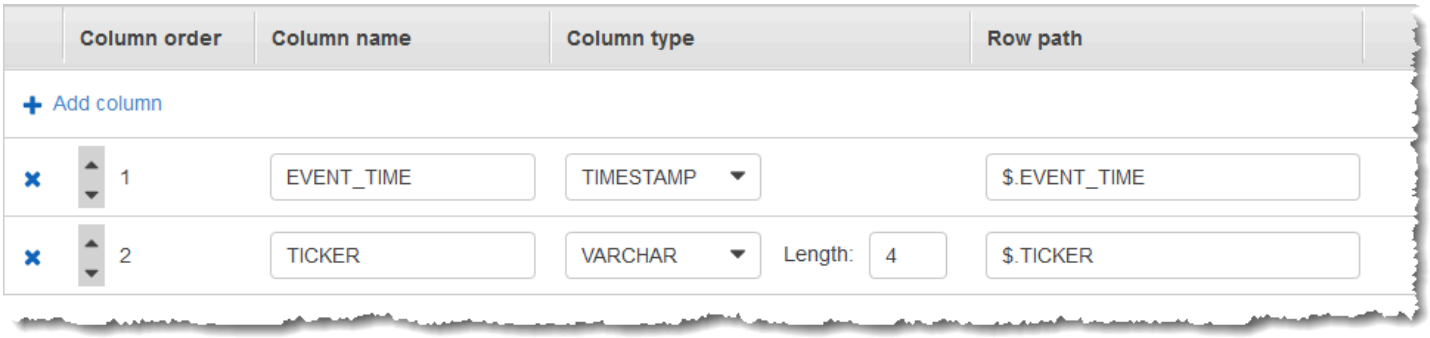
Beispiel: Versetztes Fenster

Wenn eine fensterorientierte Abfrage separate Fenster für jeden eindeutigen Partitionsschlüssel verarbeitet und damit beginnt, wenn Daten mit übereinstimmendem Schlüssel ankommen, wird das Fenster als versetztes Fenster bezeichnet. Details hierzu finden Sie unter Versetzte Fenster. Dieses Amazon-Kinesis-Data-Analytics-Beispiel verwendet die Spalten EVENT_TIME und TICKER, um versetzte Fenster zu erstellen. Der Quell-Stream enthält Gruppen von sechs Datensätzen mit identischen EVENT_TIME- und TICKER-Werten, die innerhalb eines einminütigen Zeitraums ankommen, aber nicht denselben Minutenwert (z. B. `18:41:xx`) aufweisen müssen.

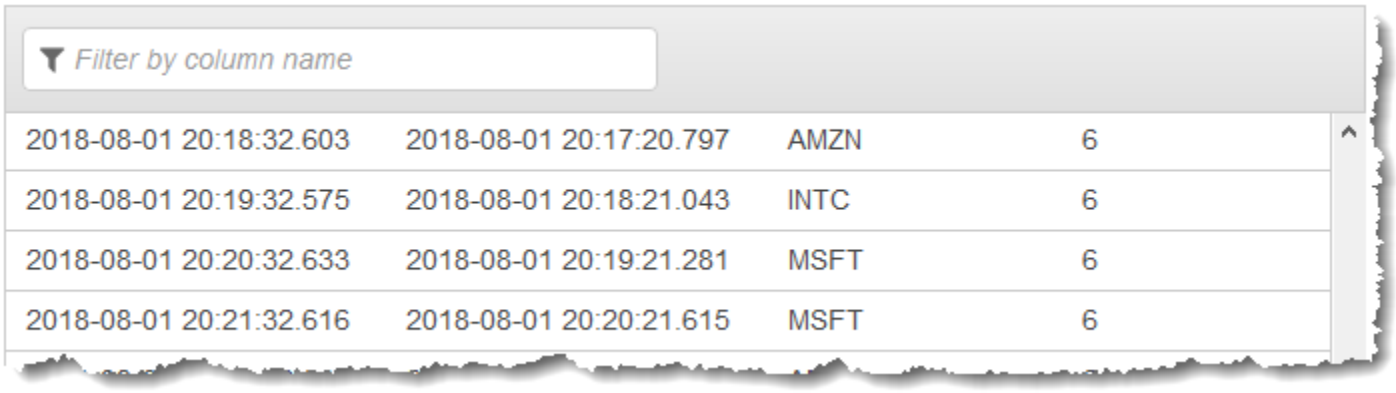
In diesem Beispiel schreiben Sie die folgenden Datensätze zu den folgenden Zeiten in einen Kinesis-Datenstrom. Das Skript schreibt die Zeiten nicht in den Stream. Allerdings wird der Zeitpunkt, zu dem der Datensatz von der Anwendung übernommen wird, in das `ROWTIME`-Feld geschrieben:

```
{"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
...
```

Anschließend erstellen Sie eine Kinesis Data Analytics Analytics-Anwendung in der AWS-Managementkonsole, mit dem Kinesis-Datenstream als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze aus der Streaming-Quelle und leitet wie nachstehend veranschaulicht ein In-Application-Schema mit zwei Spalten (`EVENT_TIME` und `TICKER`) ab.



Sie verwenden den Anwendungscode mit der Funktion `COUNT`, um eine fensterbasierte Aggregation der Daten zu erstellen. Danach fügen Sie die resultierenden Daten wie im folgenden Screenshot dargestellt in einen anderen In-Application-Stream ein:



Im folgenden Verfahren erstellen Sie eine Kinesis Data Analytics-Anwendung, die Werte im Eingabe-Stream in einem versetzten Fenster basierend auf EVENT_TIME und TICKER aggregiert.

Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Datensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie dann einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Um Datensätze in einer Produktionsumgebung in einen Kinesis-Daten-Stream zu schreiben, empfiehlt sich die Verwendung der Kinesis Producer Library oder der API der Kinesis-Daten-Streams. Der Einfachheit halber werden in diesem Beispiel mit dem Python-Skript Datensätze generiert. Führen Sie den Code aus, um die Beispiel-Ticker-Datensätze zu füllen. Dieser einfache Code schreibt während einer Minute kontinuierlich eine Gruppe von sechs Datensätzen mit derselben zufälligen `EVENT_TIME` und demselben Tickersymbol in den Stream. Führen Sie das Skript weiter aus, damit Sie das Anwendungsschema später erstellen können.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).

3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden), um eine Verbindung mit der Quelle herzustellen.
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.
   b. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema verfügt über zwei Spalten.
   c. Klicken Sie auf Edit schema (Schema bearbeiten). Ändern Sie den Wert für Column type (Spaltentyp) der Spalte EVENT_TIME in `TIMESTAMP`.
   d. Wählen Sie Save schema and update stream samples (Schema speichern und Stream-Beispiel aktualisieren). Nachdem die Konsole das Schema gespeichert hat, klicken Sie auf Exit (Beenden).
   e. Wählen Sie Save and continue aus.
5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.
6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).

Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

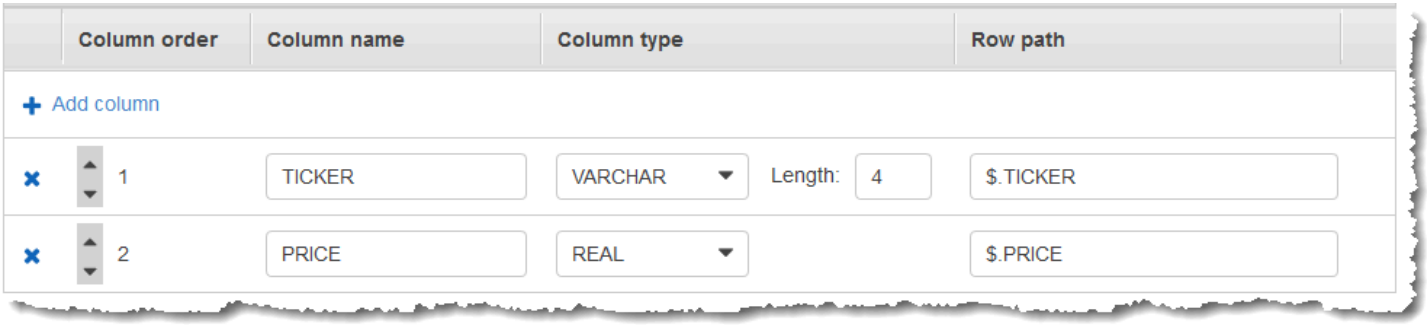
Beispiel: Rollierendes Fenster mit ROWTIME

Wenn eine Abfrage mit Fenster jedes Fenster so verarbeitet, dass diese sich nicht überschneiden, wird das Fenster als rollierendes Fenster bezeichnet. Details hierzu finden Sie unter Rollierende Fenster (Zusammenfassungen mit GROUP BY). Dieses Amazon-Kinesis-Data-Analytics-Beispiel erstellt anhand der `ROWTIME`-Spalte rollierende Fenster. Die `ROWTIME`-Spalte zeigt den Zeitpunkt an, zu dem der Datensatz von der Anwendung gelesen wurde.

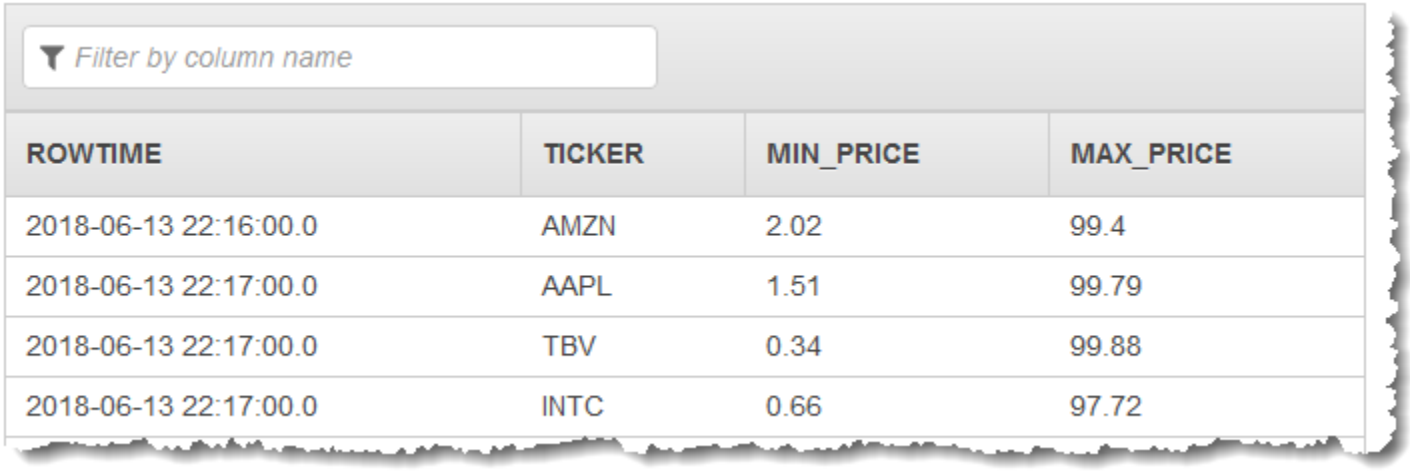
In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Kinesis-Datenstrom.

```
{"TICKER": "TBV", "PRICE": 33.11}
{"TICKER": "INTC", "PRICE": 62.04}
{"TICKER": "MSFT", "PRICE": 40.97}
{"TICKER": "AMZN", "PRICE": 27.9}
...
```

Anschließend erstellen Sie eine Kinesis Data Analytics Analytics-Anwendung in der AWS-Managementkonsole, mit dem Kinesis-Datenstream als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze aus der Streaming-Quelle und leitet wie nachstehend veranschaulicht ein In-Application-Schema mit zwei Spalten (`TICKER` und `PRICE`) ab.



Sie verwenden den Anwendungscode mit den Funktionen `MIN` und `MAX`, um eine fensterbasierte Aggregation der Daten zu erstellen. Danach fügen Sie die resultierenden Daten wie im folgenden Screenshot dargestellt in einen anderen In-Application-Stream ein:



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

Im folgenden Verfahren erstellen Sie eine Kinesis Data Analytics-Anwendung, die Werte im Eingabe-Stream in einem rollierenden Fenster basierend auf ROWTIME aggregiert.

Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Datensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie dann einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Um Datensätze an einen Kinesis-Datenstream in einer Produktionsumgebung zu schreiben, empfiehlt sich die Verwendung der Kinesis Client Library oder der API der Kinesis-Daten-Streams. Der Einfachheit halber werden in diesem Beispiel mit dem Python-Skript Datensätze generiert. Führen Sie den Code aus, um die Beispiel-Ticker-Datensätze zu füllen. Dieser einfache Code schreibt kontinuierlich einen zufälligen Ticker-Datensatz an den Stream. Führen Sie das Skript weiter aus, damit Sie das Anwendungsschema später erstellen können.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Wählen Sie Create application (Anwendung erstellen) aus, geben Sie einen Anwendungsnamen ein und wählen Sie dann Create application (Anwendung erstellen) aus.
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden), um eine Verbindung mit der Quelle herzustellen.
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.

b. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema verfügt über zwei Spalten.

c. Wählen Sie Save schema and update stream samples (Schema speichern und Stream-Beispiel aktualisieren). Nachdem die Konsole das Schema gespeichert hat, klicken Sie auf Exit (Beenden).

d. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.

6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).

Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

Beispiel: Rollierendes Fenster mit einem Ereignis-Zeitstempel

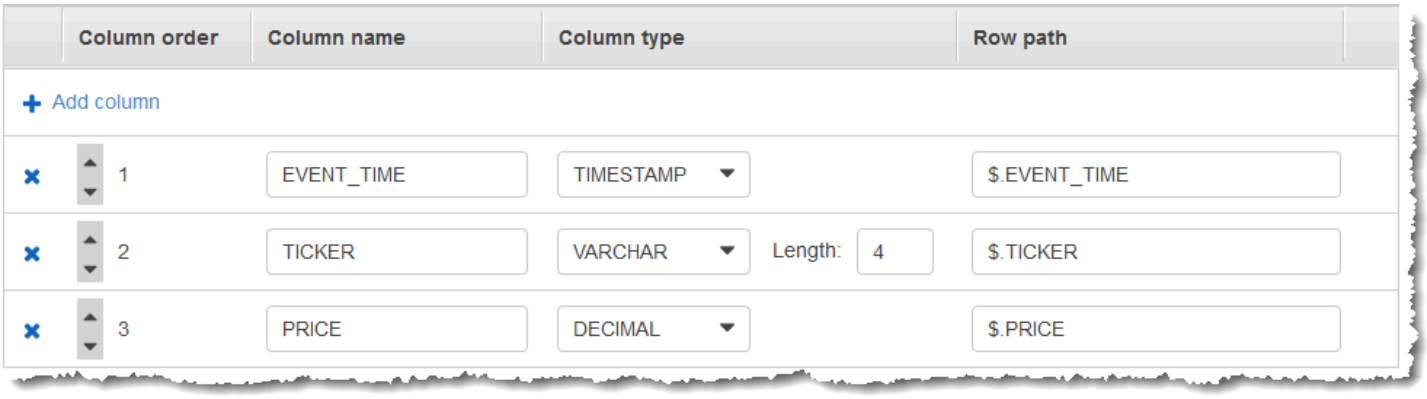
Wenn eine Abfrage mit Fenster jedes Fenster so verarbeitet, dass diese sich nicht überschneiden, wird das Fenster als rollierendes Fenster bezeichnet. Details hierzu finden Sie unter Rollierende Fenster (Zusammenfassungen mit GROUP BY). Dieses Amazon-Kinesis-Data-Analytics-Beispiel veranschaulicht ein rollierendes Fenster, das einen Ereignis-Zeitstempel verwendet. Dabei handelt es sich um einen vom Benutzer erstellten Zeitstempel, der in den Streaming-Daten enthalten ist. In

diesem Beispiel wurde dieser Ansatz gewählt, anstatt nur ROWTIME zu verwenden. Dabei handelt es sich um einen Zeitstempel, den Kinesis Data Analytics erstellt, wenn die Anwendung den Datensatz empfängt. Sie würden für die Streaming-Daten einen Ereignis-Zeitstempel verwenden, wenn die von Ihnen erstellte Aggregation auf dem Zeitpunkt des Auftretens Ereignisses basieren soll und nicht auf dem Zeitpunkt des Empfangs durch die Anwendung. In diesem Beispiel löst der `ROWTIME`-Wert die Aggregation einmal pro Minute aus und die Datensätze werden sowohl durch `ROWTIME` als auch durch die eingeschlossene Ereigniszeit aggregiert.

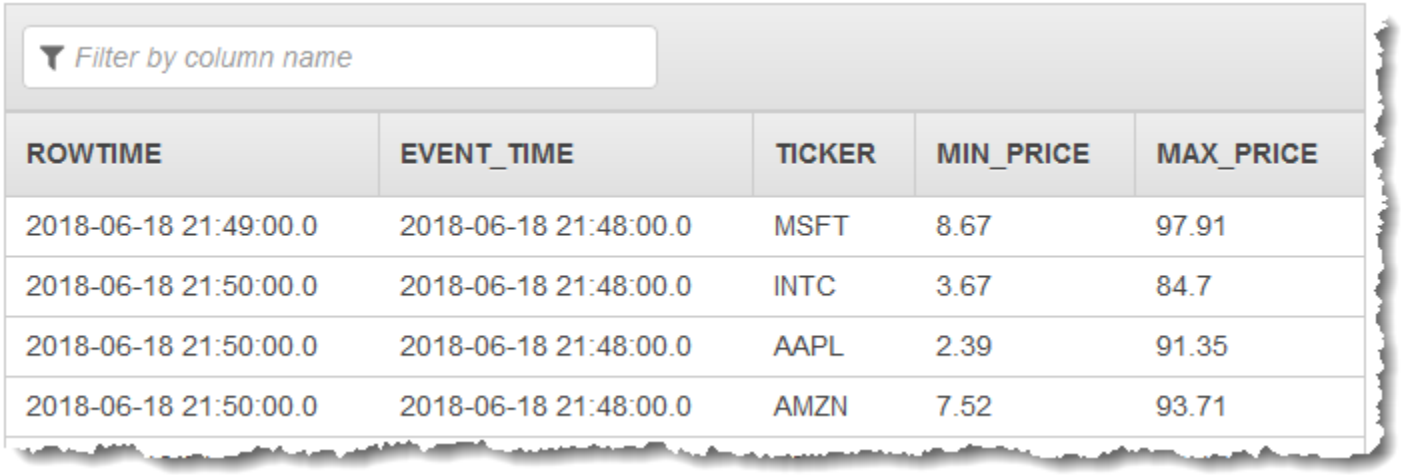
In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Amazon-Kinesis-Stream. Der Wert `EVENT_TIME` ist auf 5 Sekunden in der Vergangenheit eingestellt, um eine Verarbeitungs- und Übertragungsverzögerung zu simulieren, die möglicherweise zu einer Verzögerung zwischen dem Auftreten des Ereignisses und dem Zeitpunkt, an dem der Datensatz in Kinesis Data Analytics aufgenommen wird, führt.

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

Anschließend erstellen Sie eine Kinesis Data Analytics Analytics-Anwendung in der AWS-Managementkonsole, mit dem Kinesis-Datenstream als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze aus der Streaming-Quelle und leitet wie nachstehend veranschaulicht ein In-Application-Schema mit drei Spalten (`EVENT_TIME`, `TICKER` und `PRICE`) ab.



Sie verwenden den Anwendungscode mit den Funktionen `MIN` und `MAX`, um eine fensterbasierte Aggregation der Daten zu erstellen. Danach fügen Sie die resultierenden Daten wie im folgenden Screenshot dargestellt in einen anderen In-Application-Stream ein:


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

Im folgenden Verfahren erstellen Sie eine Kinesis Data Analytics-Anwendung, die Werte im Eingabe-Stream in einem rollierenden Fenster basierend auf einer Ereigniszeit aggregiert.

Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Datensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie dann einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Um Datensätze an einen Kinesis-Datenstream in einer Produktionsumgebung zu schreiben, empfiehlt sich die Verwendung der Kinesis Client Library oder der API der Kinesis-Daten-Streams. Der Einfachheit halber werden in diesem Beispiel mit dem Python-Skript Datensätze generiert. Führen Sie den Code aus, um die Beispiel-Ticker-Datensätze zu füllen. Dieser einfache Code schreibt kontinuierlich einen zufälligen Ticker-Datensatz an den Stream. Führen Sie das Skript weiter aus, damit Sie das Anwendungsschema später erstellen können.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Wählen Sie Create application (Anwendung erstellen) aus, geben Sie einen Anwendungsnamen ein und wählen Sie dann Create application (Anwendung erstellen) aus.
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden), um eine Verbindung mit der Quelle herzustellen.
4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.

b. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema verfügt über drei Spalten.

c. Klicken Sie auf Edit schema (Schema bearbeiten). Ändern Sie den Wert für Column type (Spaltentyp) der Spalte EVENT_TIME in `TIMESTAMP`.

d. Wählen Sie Save schema and update stream samples (Schema speichern und Stream-Beispiel aktualisieren). Nachdem die Konsole das Schema gespeichert hat, klicken Sie auf Exit (Beenden).

e. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.

6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein.

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).

Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

Beispiel: Abrufen der am häufigsten auftretenden Werte (TOP_K_ITEMS_TUMBLING)

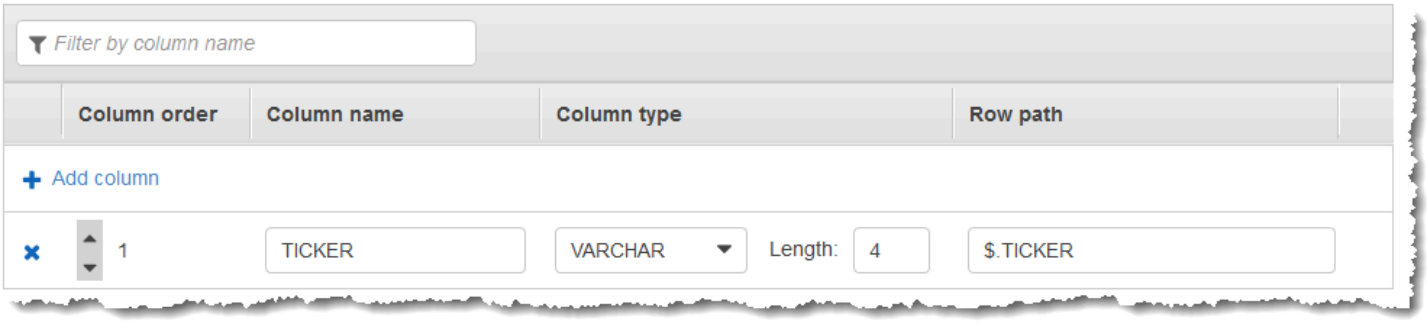
Dieses Amazon-Kinesis-Data-Analytics-Beispiel veranschaulicht, wie mit der Funktion `TOP_K_ITEMS_TUMBLING` die am häufigsten auftretenden Werte in einem rollierenden Fenster abgerufen werden. Weitere Informationen finden Sie in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink unter `TOP_K_ITEMS_TUMBLING`-Funktion.

Die Funktion `TOP_K_ITEMS_TUMBLING` ist beim Aggregieren von mehr als Zehn- oder Hunderttausenden von Schlüsseln hilfreich, und wenn Sie Ihre Ressourcennutzung reduzieren möchten. Die Funktion führt zu dem gleichen Ergebnis wie die Aggregation mit den Klauseln `GROUP BY` und `ORDER BY`.

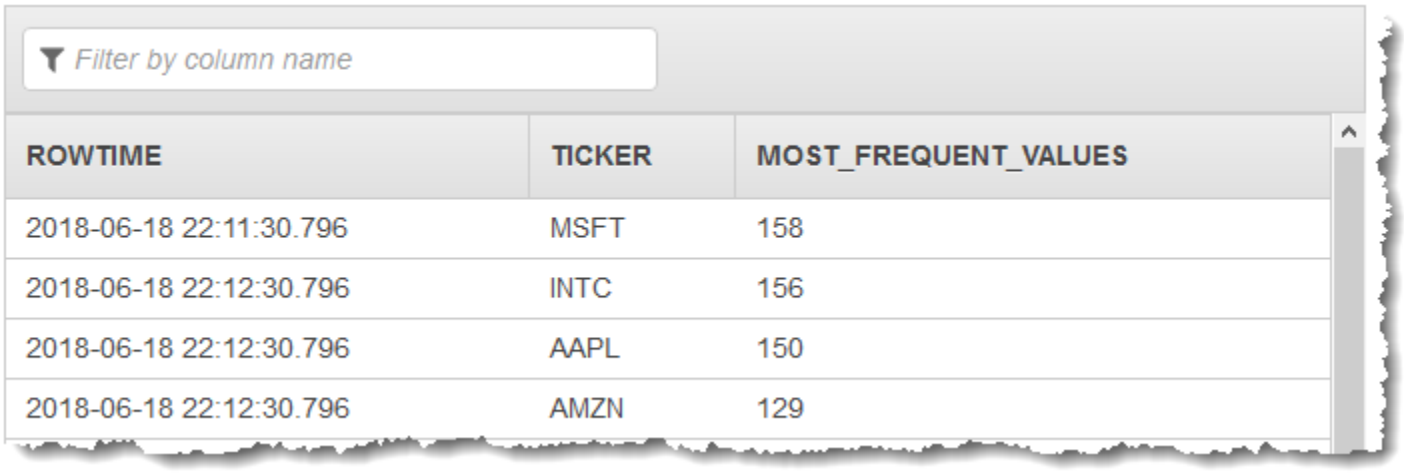
In diesem Beispiel schreiben Sie die folgenden Datensätze in einen Amazon-Kinesis-Datenstrom:

```
{"TICKER": "TBV"}
{"TICKER": "INTC"}
{"TICKER": "MSFT"}
{"TICKER": "AMZN"}
...
```

Anschließend erstellen Sie eine Kinesis Data Analytics Analytics-Anwendung in der AWS-Managementkonsole, mit dem Kinesis-Datenstream als Streaming-Quelle. Der Erkennungsvorgang liest Beispieldatensätze aus der Streaming-Quelle und leitet wie nachstehend veranschaulicht ein In-Application-Schema mit einer Spalte (`TICKER`) ab.



Sie verwenden den Anwendungscode mit der Funktion `TOP_K_VALUES_TUMBLING`, um eine fensterbasierte Aggregation der Daten zu erstellen. Danach fügen Sie die resultierenden Daten wie im folgenden Screenshot dargestellt in einen anderen In-Application-Stream ein:



Im folgenden Verfahren erstellen Sie eine Kinesis Data Analytics-Anwendung, die die am häufigsten auftretenden Werte im Eingabe-Stream abruft.

Themen

- Schritt 1: Erstellen eines Kinesis-Datenstroms
- Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 1: Erstellen eines Kinesis-Datenstroms

Erstellen Sie einen Amazon-Kinesis-Datenstrom und füllen Sie die Datensätze wie folgt aus:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie dann einen Stream mit einer Shard. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
4. Um Datensätze an einen Kinesis-Datenstream in einer Produktionsumgebung zu schreiben, empfiehlt sich die Verwendung der Kinesis Client Library oder der API der Kinesis-Daten-Streams. Der Einfachheit halber werden in diesem Beispiel mit dem Python-Skript Datensätze generiert. Führen Sie den Code aus, um die Beispiel-Ticker-Datensätze zu füllen. Dieser einfache Code schreibt kontinuierlich einen zufälligen Ticker-Datensatz an den Stream. Lassen Sie das Skript laufen, sodass Sie in einem späteren Schritt das Anwendungsschema erstellen können.

```
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"))
```

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

1. Öffnen Sie die Managed Service for Apache Flink-Konsole unter /kinesisanalytics. https://console.aws.amazon.com
2. Klicken Sie auf Create application (Anwendung erstellen), geben Sie einen Anwendungsnamen ein und klicken Sie erneut auf Create application (Anwendung erstellen).
3. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden), um eine Verbindung mit der Quelle herzustellen.

4. Gehen Sie auf der Seite Connect to source (Mit Quelle verbinden) wie folgt vor:

   a. Wählen Sie den Stream aus, den Sie im vorherigen Abschnitt erstellt haben.

   b. Klicken Sie auf Discover schema (Schema erkennen). Warten Sie, bis die Konsole das abgeleitete Schema und die Beispieldatensätze anzeigt, die zum Ableiten des Schemas für den erstellten In-Application-Stream verwendet werden. Das abgeleitete Schema weist eine Spalte auf.

   c. Wählen Sie Save schema and update stream samples (Schema speichern und Stream-Beispiel aktualisieren). Nachdem die Konsole das Schema gespeichert hat, klicken Sie auf Exit (Beenden).

   d. Wählen Sie Save and continue aus.

5. Klicken Sie auf der Detailseite der Anwendung auf Go to SQL editor (Gehe zu SQL-Editor). Um die Anwendung zu starten, wählen Sie im angezeigten Dialogfeld Yes, start application (Ja, Anwendung starten) aus.

6. Schreiben Sie im SQL-Editor den Anwendungscode und überprüfen Sie die Ergebnisse wie folgt:

   a. Kopieren Sie den folgenden Anwendungscode und fügen Sie diesen in den Editor ein:

```
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. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).

   Auf der Registerkarte Real-time analytics (Echtzeitanalyse) können Sie alle In-Application-Streams sehen, die von der Anwendung erstellt wurden, und die Daten überprüfen.

Beispiel: Aggregieren von Teilergebnissen aus einer Abfrage

Wenn ein Amazon-Kinesis-Datenstrom Datensätze mit einer Ereigniszeit enthält, die nicht genau dem Zeitpunkt der Datenübernahme entspricht, enthalten bestimmte Ergebnisse in einem rollierenden Fenster Datensätze, die innerhalb des Fensters eingetroffen sind, aber nicht unbedingt eingetreten sein müssen. In diesem Fall enthält das rollierende Fenster nur eine Teilmenge der gewünschten Ergebnisse. Es gibt mehrere Möglichkeiten, wie Sie dieses Problem beheben können:

- Verwenden Sie nur ein rollierendes Fenster und aggregieren Sie Teilergebnisse in der Nachverarbeitung über eine Datenbank oder ein Data Warehouse mithilfe von upsert-Operationen. Dieser Ansatz ist bei der Verarbeitung einer Anwendung sehr effizient. Verspätete Daten für Aggregat-Operatoren (`sum`, `min`, `max` usw.) werden ohne jegliche Einschränkungen verarbeitet. Der Nachteil dieser Methode besteht darin, dass Sie eine zusätzliche Anwendungslogik im Datenbank-Layer entwickeln und pflegen müssen.
- Verwenden Sie ein rollierendes und ein gleitendes Fenster, das früh zu Teilergebnissen führt, aber innerhalb des Zeitraums des gleitenden Fensters zudem vollständige Ergebnisse erzielen wird. Dieser Ansatz verarbeitet verspätete Daten durch Überschreiben anstatt mit einer upsert-Operation. Daher muss keine zusätzliche Anwendungslogik im Datenbank-Layer hinzugefügt werden. Der Nachteil dieses Ansatzes besteht darin, dass mehr Kinesis-Verarbeitungseinheiten (KPUs) verwendet werden und dennoch zwei Ergebnisse erzielt werden, die in einigen Anwendungsfällen möglicherweise nicht funktionieren.

Weitere Informationen zu rollierenden und gleitenden Fenstern finden Sie unter Abfragen mit Fenstern.

Im folgenden Verfahren erzielt die Aggregation über ein rollierendes Fenster zwei (an den `CALC_COUNT_SQL_STREAM`-In-Application-Stream übermittelte) Teilergebnisse, die zusammen das Endergebnis ergeben. Die Anwendung erstellt dann eine zweite (an den `DESTINATION_SQL_STREAM`-In-Application-Stream übermittelte) Aggregation, um die beiden Teilergebnisse zu vereinen.

So erstellen Sie eine Anwendung, die Teilergebnisse mittels einer Ereigniszeit aggregiert

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.

2. Klicken Sie im Navigationsbereich auf Data Analytics (Datenanalyse). Erstellen Sie eine Kinesis Data Analytics-Anwendung gemäß der Beschreibung im Erste Schritte mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen-Tutorial.

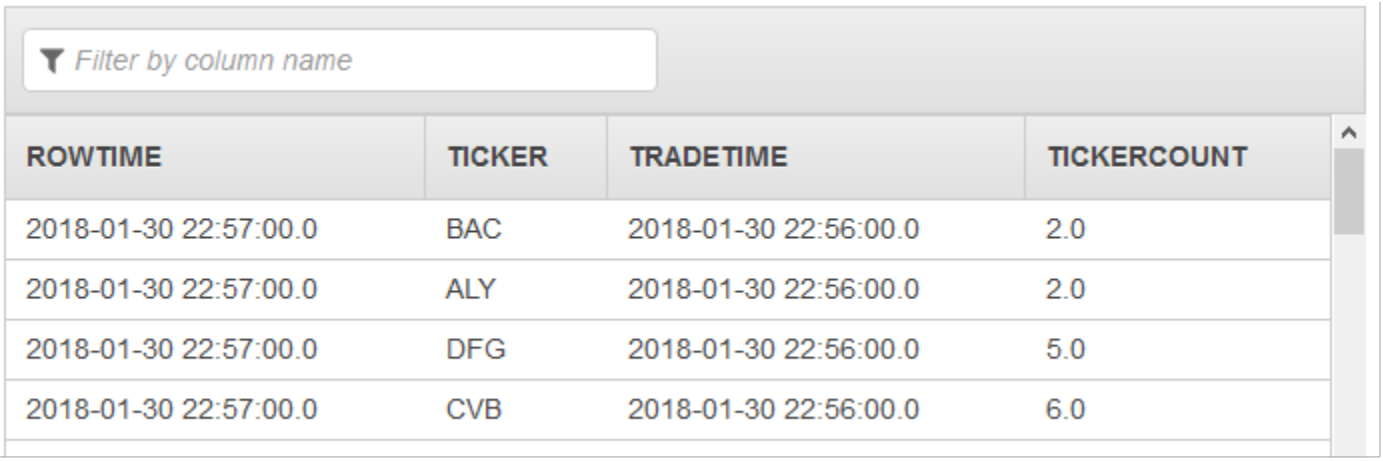
3. Ersetzen Sie im SQL-Editor den Anwendungscode durch Folgendes:

```
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);
```

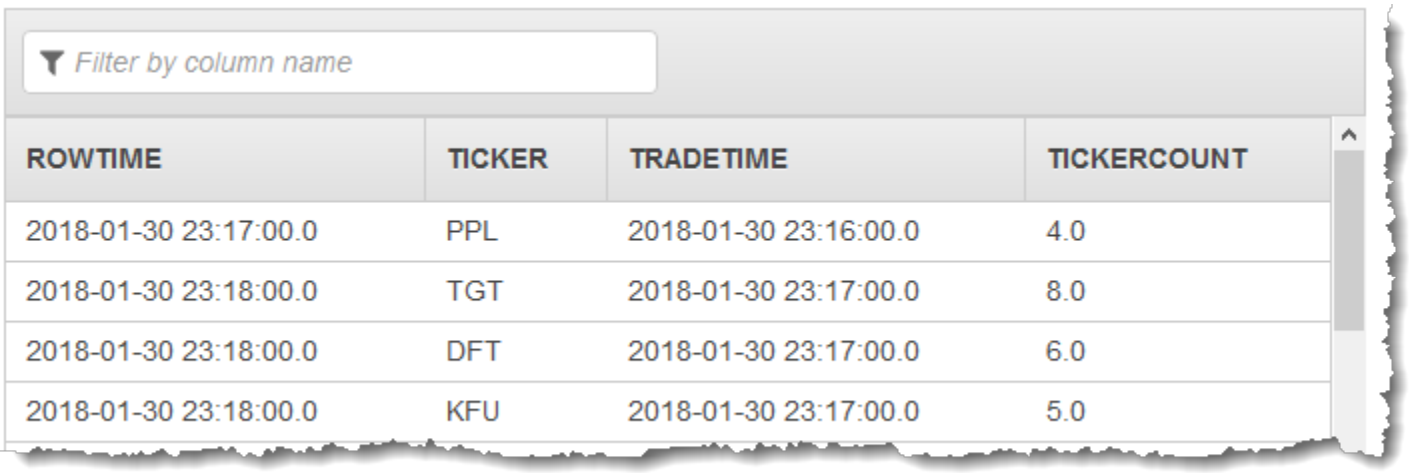
Die SELECT-Anweisung im Anwendungscode filtert Zeilen im SOURCE_SQL_STREAM_001 nach Aktien mit Preisänderungen von mehr als 1 % und fügt diese Zeilen mittels eines Pump in einen anderen CHANGE_STREAM-In-Application-Stream ein.

4. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).

Die erste Pumpe gibt einen Stream ähnlich dem Folgenden an `CALC_COUNT_SQL_STREAM` aus. Beachten Sie, dass der Ergebnissatz unvollständig ist:



Die zweite Pumpe gibt dann einen Stream mit dem vollständigen Ergebnissatz an `DESTINATION_SQL_STREAM` aus:



Beispiele: Joins

In diesem Abschnitt finden Sie Beispiele für Kinesis Data Analytics-Anwendungen mit Join-Abfragen. Jedes Beispiel enthält step-by-step Anweisungen zum Einrichten und Testen Ihrer Kinesis Data Analytics Analytics-Anwendung.

Themen

- Beispiel: Hinzufügen von Referenzdaten zu einer Kinesis Data Analytics-Anwendung

Beispiel: Hinzufügen von Referenzdaten zu einer Kinesis Data Analytics-Anwendung

In dieser Übung fügen Sie Referenzdaten zu einer vorhandenen Kinesis Data Analytics-Anwendung hinzu. Weitere Informationen zu Referenzdaten finden Sie in den folgenden Themen:

- Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's
- Konfigurieren der Anwendungseingabe

In diesem Beispiel fügen Sie Referenzdaten zu der Anwendung hinzu, die Sie in der Erste Schritte-Übung zu Kinesis Data Analytics erstellt haben. Die Referenzdaten stellen den Firmennamen für die einzelnen Tickersymbole bereit. Beispiel:

```
Ticker, Company
AMZN,Amazon
ASD, SomeCompanyA
MMB, SomeCompanyB
WAS,  SomeCompanyC
```

Führen Sie zunächst die Schritte in der Übung Erste Schritte aus, um eine Starter-Anwendung zu erstellen. Anschließend befolgen Sie diese Schritte, um Referenzdaten einzurichten und diese Ihrer Anwendung hinzuzufügen:

1. Vorbereitung der Daten
   - Speichern Sie die vorherigen Referenzdaten als Objekt in Amazon-Simple-Storage-Service (Amazon-S3).
   - Erstellen Sie eine IAM-Rolle, die Kinesis Data Analytics annehmen kann, um das S3-Objekt in Ihrem Namen zu lesen.
2. Fügen Sie Ihrer Anwendung die Referenzdatenquelle hinzu.

   Kinesis Data Analytics liest das Amazon-S3-Objekt und erstellt eine In-Application-Referenztabelle, die Sie in Ihrem Anwendungs-Code abfragen können.
3. Testen Sie den Code.

   In Ihrem Anwendungscode schreiben Sie eine Join-Abfrage, um den In-Application-Stream mit der In-Application-Referenztabelle zusammenzuführen, um Firmennamen für die einzelnen Tickersymbol zu erhalten.

Themen

Schritt 1: Vorbereiten

In diesem Abschnitt speichern Sie Beispielreferenzdaten als Objekt in einem Amazon-S3-Bucket. Sie erstellen darüber hinaus eine IAM-Rolle, die Kinesis Data Analytics annehmen kann, um das Objekt in Ihrem Namen zu lesen.

Speichern von Referenzdaten als Amazon-S3-Objekt

In diesem Schritt speichern Sie die Beispielreferenzdaten als Amazon-S3-Objekt.

1. Öffnen Sie einen Text-Editor, fügen Sie die folgenden Daten hinzu und speichern Sie die Datei als `TickerReference.csv`.

   ```
   Ticker, Company
   AMZN,Amazon
   ASD, SomeCompanyA
   MMB, SomeCompanyB
   WAS,  SomeCompanyC
   ```

2. Laden Sie die Datei `TickerReference.csv` in Ihren S3-Bucket hoch. Weitere Anleitungen finden Sie unter Upload eines Objekts zu Amazon-S3 im Benutzerhandbuch für Amazon-Simple-Storage-Service.

Erstellen einer IAM-Rolle

Als Nächstes erstellen Sie eine IAM-Rolle, die Kinesis Data Analytics annehmen kann und lesen das Amazon-S3-Objekt ein.

1. Erstellen Sie in AWS Identity and Access Management (IAM) eine IAM-Rolle mit dem Namen. **`KinesisAnalytics-ReadS3Object`** Folgen Sie beim Erstellen einer Rolle den Anweisungen unter Erstellen einer Rolle für einen Amazon-Service (AWS-Managementkonsole) im IAM-Benutzerhandbuch.

Geben Sie in der IAM-Konsole Folgendes an:

- Wählen Sie unter Rollentyp auswählen die Option AWS Lambda. Nachdem Sie die Rolle erstellt haben, ändern Sie die Vertrauensrichtlinie, sodass Kinesis Data Analytics (nicht AWS Lambda) die Rolle übernehmen kann.
- Fügen Sie auf der Seite Attach Policy keine Richtlinie hinzu.

2. Aktualisieren Sie die IAM-Rollenrichtlinien:

   a. Wählen Sie in der IAM-Konsole die von Ihnen erstellte Rolle aus.

   b. Aktualisieren Sie auf der Registerkarte Vertrauensstellungen die Vertrauensrichtlinie, um Kinesis Data Analytics die Berechtigung für die Annahme der Rolle zu gewähren. Die Vertrauensrichtlinie wird im Folgenden angezeigt:

      JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   c. Fügen Sie auf der Registerkarte Berechtigungen eine von Amazon verwaltete Richtlinie mit dem Namen AmazonS3 hinzu. ReadOnlyAccess Damit gewähren Sie der Rolle Berechtigungen für das Lesen von Amazon-S3-Objekten. Diese Richtlinie wird im Folgenden angezeigt:

      JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

Schritt 2: Hinzufügen der Referenzdatenquelle zur Anwendungskonfiguration

Bei diesem Schritt fügen Sie Ihrer Anwendungskonfiguration eine Referenzdatenquelle hinzu. Um beginnen zu können, benötigen Sie die folgenden Informationen:

- Den Namen Ihres S3-Buckets und den Namen des Objektschlüssels
- Der Amazon-Ressourcenname (ARN) der IAM-Rolle

1. Wählen Sie auf der Hauptseite der Anwendung Connect reference data (Referenzdaten verbinden) aus.
2. Wählen Sie auf der Seite Referenzdatenquelle verbinden den Amazon-S3-Bucket mit Ihrem Referenzdatenobjekt aus. Geben Sie dann den Schlüsselnamen des Objekts ein.
3. Geben Sie **`CompanyName`** als Namen der In-Application-Referenztabelle ein.
4. Wählen Sie im Abschnitt Zugriff auf ausgewählte Ressourcen die Option Aus IAM-Rollen auswählen, die Kinesis Analytics annehmen kann, und wählen Sie die IAM-Rolle KinesisAnalytics -reads3Object aus, die Sie im vorherigen Abschnitt erstellt haben.
5. Klicken Sie auf Discover schema (Schema erkennen). Die Konsole erkennt in den Referenzdaten zwei Spalten.
6. Klicken Sie auf Save and close.

Schritt 3: Test: Abfragen der In-Application-Referenztabelle

Sie können nun die In-Application-Referenztabelle `CompanyName` abfragen. Sie können die Referenzdaten verwenden, um Ihre Anwendung zu erweitern, indem Sie die Tickerpreisdaten mit der Referenztabelle zusammenführen. Das Ergebnis zeigt den Firmennamen an.

1. Ersetzen Sie den Anwendungscode durch den folgenden Code. Die Abfrage führt den In-Application-Eingabe-Stream mit der In-Application-Referenztabelle zusammen. Der Anwendungscode schreibt die Ergebnisse in einen anderen In-Application-Stream, `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. Stellen Sie sicher, dass die Anwendungsausgabe auf der Registerkarte angezeigt wird. SQLResults Stellen Sie sicher, dass einige Zeilen Firmennamen anzeigen (nicht alle Beispielreferenzdaten besitzen Firmennamen).

Beispiele: Machine Learning

In diesem Abschnitt finden Sie Beispiele für Amazon-Kinesis-Data-Analytics-Anwendungen die Machine Learning-Abfragen nutzen. Abfragen des maschinellen Lernens führen eine komplexere Datenanalyse durch und decken anhand der Geschichte der Daten im Stream ungewöhnliche Muster auf. Die Beispiele enthalten step-by-step Anweisungen zum Einrichten und Testen Ihrer Kinesis Data Analytics Analytics-Anwendung.

Themen

- Beispiel: Erkennen von Datenanomalien in einem Stream (Funktion RANDOM_CUT_FOREST)
- Beispiel: Erkennen von Datenanomalien und Suchen nach einer Erklärung (Funktion RANDOM_CUT_FOREST_WITH_EXPLANATION)
- Beispiel: Erkennen von Hotspots in einem Stream (HOTSPOTS-Funktion)

Beispiel: Erkennen von Datenanomalien in einem Stream (Funktion RANDOM_CUT_FOREST)

Amazon-Kinesis-Data-Analytics stellt eine Funktion (`RANDOM_CUT_FOREST`) bereit, die jedem Datensatz basierend auf Werten in den numerischen Spalten eine Anomaliebewertung zuweisen kann. Weitere Informationen finden Sie unter `RANDOM_CUT_FOREST`-Funktion in der SQL-Referenz zu Amazon-Managed-Service für Apache Flink.

In dieser Übung schreiben Sie Anwendungscode, um Datensätzen in der Streaming-Quelle Ihrer Anwendung eine Anomaliebewertung zuzuweisen. Um die Anwendung einzurichten, führen Sie Folgendes aus:

1. Einrichten einer Streaming-Quelle – Sie richten einen Kinesis-Datenstrom ein und schreiben `heartRate`-Beispieldaten wie folgt:

   ```
   {"heartRate": 60, "rateType":"NORMAL"}
   ...
   {"heartRate": 180, "rateType":"HIGH"}
   ```

   Das Verfahren stellt ein Python-Skript für die Auffüllung des Streams bereit. Die `heartRate`-Werte werden nach dem Zufallsprinzip generiert, wobei 99 % der Datensätze `heartRate`-Werte zwischen 60 und 100 aufweisen und nur 1 % der `heartRate`-Werte zwischen 150 und 200 liegen. Daher stellen die Datensätze mit `heartRate`-Werten zwischen 150 und 200 Anomalien dar.
2. Konfigurieren der Eingabe – Sie erstellen mittels der Konsole eine Kinesis Data Analytics-Anwendung und konfigurieren die Anwendungseingabe durch Zuweisung der Streaming-Quelle zu einem In-Application-Stream (`SOURCE_SQL_STREAM_001`). Wenn die Anwendung gestartet wird, liest Kinesis Data Analytics kontinuierlich die Streaming-Quelle und fügt Datensätze in den In-Application-Stream ein.
3. Angeben des Anwendungscodes – Das Bespiel verwendet den folgenden Anwendungscode:

   ```
   --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;
```

Der Code liest die Zeilen im `SOURCE_SQL_STREAM_001`, weist eine Anomaliebewertung zu und schreibt die resultierenden Zeilen in einen anderen In-Application-Stream (`TEMP_STREAM`). Der Anwendungscode sortiert anschließend die Datensätze im `TEMP_STREAM` und speichert die Ergebnisse zu einem anderen In-Application-Stream (`DESTINATION_SQL_STREAM`). Sie verwenden Pumps, um Zeilen in In-Application-Streams einzufügen. Weitere Informationen finden Sie unter In-Application-Streams und Pumps.

4. Konfigurieren der Ausgabe – Sie konfigurieren die Anwendungsausgabe für die Weiterleitung von Daten im `DESTINATION_SQL_STREAM` an ein externes Ziel, bei dem es sich um einen anderen Kinesis-Datenstrom handelt. Die Prüfung der den einzelnen Datensätzen zugewiesenen Anomaliebewertungen und die Festlegung, welche Bewertung eine Anomalie anzeigt (und eine Warnung auslösen muss), werden außerhalb der Anwendung ausgeführt. Sie können eine AWS Lambda Funktion verwenden, um diese Anomaliewerte zu verarbeiten und Warnmeldungen zu konfigurieren.

Die Übung verwendet die Region USA Ost (Nord-Virginia) (`us-east-1`), um diese Streams und Ihre Anwendung zu erstellen. Wenn Sie eine andere Region verwenden, müssen Sie den Code entsprechend aktualisieren.

Themen

- Schritt 1: Vorbereiten

- Schritt 2: Eine Anwendung erstellen
- Schritt 3: Konfigurieren Sie die Anwendungsausgabe.
- Schritt 4: Überprüfen der Ausgabe

Nächster Schritt

Schritt 1: Vorbereiten

Schritt 1: Vorbereiten

Bevor Sie eine Amazon-Kinesis-Data-Analytics-Anwendung für diese Übung erstellen, müssen Sie zwei Kinesis-Datenströme erstellen. Konfigurieren Sie einen der Streams als Streaming-Quelle für Ihre Anwendung und den anderen Stream als das Ziel, an das Kinesis Data Analytics die Ausgabe Ihrer Anwendung weiterleitet.

Themen

- Schritt 1.1: Erstellen der Eingabe- und Ausgabe-Daten-Streams
- Schritt 1.2: Schreiben Sie Beispieldatensätze in den Eingabe-Stream

Schritt 1.1: Erstellen der Eingabe- und Ausgabe-Daten-Streams

In diesem Abschnitt erstellen Sie zwei Kinesis-Streams: `ExampleInputStream` und `ExampleOutputStream`. Sie können diese Streams mithilfe der AWS-Managementkonsole oder der AWS CLI erstellen.

- So verwenden Sie die -Konsole:

  1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
  2. Klicken Sie auf Create data stream (Daten-Stream erstellen). Erstellen Sie einen Stream mit einer Shard namens `ExampleInputStream`. Weitere Informationen finden Sie unter Einen Stream erstellen im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.
  3. Wiederholen Sie den vorherigen Schritt und erstellen Sie einen Stream mit einer Shard namens `ExampleOutputStream`.

- Um das zu verwenden AWS CLI

  1. Verwenden Sie den folgenden `create-stream` AWS CLI Kinesis-Befehl, um den ersten Stream zu erstellen (`ExampleInputStream`).

     ```
     $ aws kinesis create-stream \
     --stream-name ExampleInputStream \
     --shard-count 1 \
     --region us-east-1 \
     --profile adminuser
     ```

  2. Führen Sie den gleichen Befehl aus und ändern Sie den Namen des Streams in `ExampleOutputStream`. Über diesen Befehl wird der zweite Stream erstellt, der von der Anwendung zum Schreiben der Ausgabe verwendet wird.

Schritt 1.2: Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Schritt führen Sie Python-Code aus, um kontinuierlich Beispieldatensätze zu generieren und diese Datensätze in den `ExampleInputStream`-Stream zu schreiben.

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

1. Installieren Sie Python und `pip`.

   Informationen zur Installation von Python finden Sie auf der Website für Python.

   Sie können mithilfe von pip Abhängigkeiten installieren. Informationen zur Installation von pip finden Sie unter Installation auf der Website für pip.

2. Führen Sie den folgenden Python-Code aus. Der Befehl `put-record` im Code schreibt die JSON-Datensätze zum Stream.

   ```
   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"))
```

Nächster Schritt

Schritt 2: Eine Anwendung erstellen

Schritt 2: Eine Anwendung erstellen

In diesem Abschnitt erstellen Sie folgendermaßen eine Amazon-Kinesis-Data-Analytics-Anwendung:

- Konfigurieren Sie die Anwendungseingabe für die Verwendung des Kinesis-Datenstroms, den Sie in the section called “Schritt 1: Vorbereiten” als Streaming-Quelle erstellt haben.
- Verwenden Sie die Vorlage Anomaly Detection (Anomalieerkennung) in der Konsole.

So erstellen Sie eine Anwendung

1. Folgen Sie den Schritten 1, 2 und 3 der Übung Erste Schritte zu Kinesis Data Analytics (sieheSchritt 3.1: Eine Anwendung erstellen).

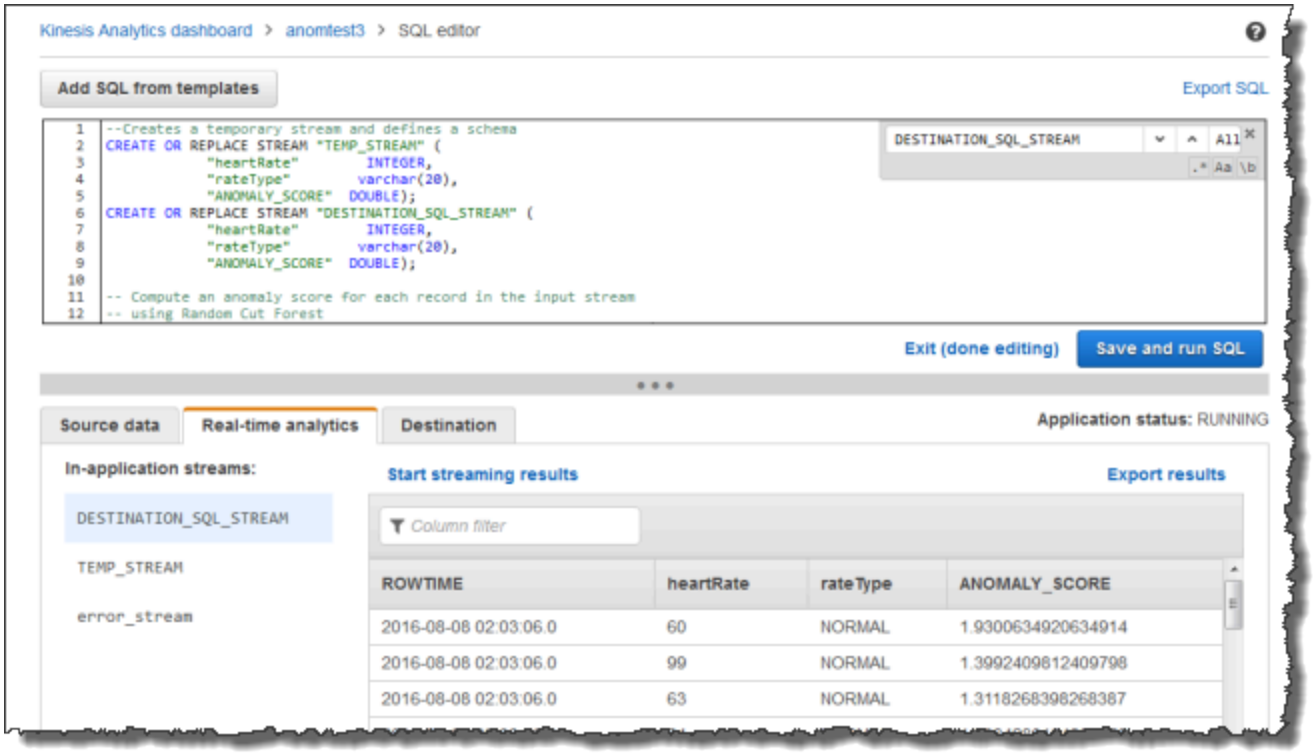
   - Führen Sie in der Quellkonfiguration Folgendes aus:
     - Geben Sie die Streaming-Quelle an, die Sie im vorherigen Abschnitt erstellt haben.
     - Nachdem die Konsole das Schema abgeleitet hat, bearbeiten Sie das Schema und legen den `heartRate`-Spaltentyp als `INTEGER` fest.

       Die Mehrzahl der Herzfrequenz Werte ist normal und der Erkennungsprozess wird dieser Spalte sehr wahrscheinlich den Typ `TINYINT` zuweisen. Ein sehr kleiner Prozentsatz der Werte zeigt jedoch eine sehr hohe Herzfrequenz an. Wenn diese hohen Werte nicht in den `TINYINT`-Typ passen, sendet Kinesis Data Analytics diese Zeilen an einen Fehler-Stream. Aktualisieren Sie den Datentyp auf `INTEGER`, sodass er alle generierten Herzfrequenzdaten aufnehmen kann.
   - Verwenden Sie die Vorlage Anomaly Detection (Anomalieerkennung) in der Konsole. Anschließend aktualisieren Sie den Vorlagencode, um den geeigneten Spaltennamen bereitzustellen.
2. Aktualisieren Sie den Anwendungscode durch Bereitstellen von Spaltennamen. Der resultierende Anwendungscode wird im Folgenden angezeigt (fügen Sie diesen Code in den SQL-Editor ein):

```
--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. Führen Sie den SQL-Code aus und überprüfen Sie die Ergebnisse in der Kinesis Data Analytics-Konsole:



Nächster Schritt

[Schritt 3: Konfigurieren Sie die Anwendungsausgabe.]

Schritt 3: Konfigurieren Sie die Anwendungsausgabe.

Nachdem the section called “Schritt 2: Eine Anwendung erstellen” ausgeführt wurde, verfügen Sie über Anwendungscode, der Herzfrequenzdaten aus einer Streaming-Quelle liest und den einzelnen Daten Anomaliebewertungen zuweist.

Sie können nun die Anwendungsergebnisse aus dem In-Application-Stream an ein externes Ziel senden, das ein anderer Kinesis-Datenstrom (`OutputStreamTestingAnomalyScores`) ist. Anschließend können Sie die Anomaliebewertungen analysieren und ermitteln, welche Herzfrequenz anormal ist. Sie können diese Anwendung dann zusätzlich erweitern, um Warnungen zu generieren.

Gehen Sie wie folgt vor, um die Anwendungsausgabe zu konfigurieren:

1. Öffnen Sie die Amazon-Kinesis-Data-Analytics-Konsole. Wählen Sie im SQL-Editor Destination oder Add a destination im Anwendungs-Dashboard aus.
2. Wählen Sie auf der Seite Connect to destination (Mit Ziel verbinden) den von Ihnen im vorhergehenden Abschnitt erstellten `OutputStreamTestingAnomalyScores`-Stream aus.

   Sie verfügen nun über ein externes Ziel, an das Amazon-Kinesis-Data-Analytics alle Datensätze weiterleitet, die Ihre Anwendung in den In-Application-Stream `DESTINATION_SQL_STREAM` schreibt.
3. Sie können optional konfigurieren AWS Lambda , dass der `OutputStreamTestingAnomalyScores` Stream überwacht und Ihnen Warnmeldungen gesendet werden. Detaillierte Anweisungen finden Sie unter Vorverarbeitung von Daten mithilfe einer Lambda-Funktion. Wenn Sie keine Warnungen festlegen, können Sie die Datensätze überprüfen, die Kinesis Data Analytics in das externe Ziel schreibt. Hierbei handelt es sich, wie in Schritt 4: Überprüfen der Ausgabe beschrieben, um den Kinesis-Datenstrom `OutputStreamTestingAnomalyScores`.

Nächster Schritt

Schritt 4: Überprüfen der Ausgabe

Schritt 4: Überprüfen der Ausgabe

Nachdem Sie die Anwendungsausgabe in the section called “Schritt 3: Konfigurieren Sie die Anwendungsausgabe.” konfiguriert haben, verwenden Sie die folgenden AWS CLI -Befehle, um Datensätze im Ziel-Stream zu lesen, der von der Anwendung geschrieben wird:

1. Führen Sie den Befehl `get-shard-iterator` aus, um einen Zeiger auf Daten im Ausgabe-Stream abzurufen.

```
aws kinesis get-shard-iterator \
--shard-id shardId-000000000000 \
--shard-iterator-type TRIM_HORIZON \
--stream-name OutputStreamTestingAnomalyScores \
--region us-east-1 \
--profile adminuser
```

   Sie erhalten eine Antwort mit einem Shard-Iteratorwert wie in der folgenden Beispielantwort gezeigt:

```
  {
      "ShardIterator":
      "shard-iterator-value"   }
```

   Kopieren Sie den Shard-Iteratorwert.

2. Führen Sie den Befehl AWS CLI `get-records` aus.

```
aws kinesis get-records \
--shard-iterator shared-iterator-value \
--region us-east-1 \
--profile adminuser
```

   Der Befehl gibt eine Seite mit Datensätzen und einen anderen Shard-Iterator zurück, den Sie im folgenden `get-records` Befehl verwenden können, um den nächsten Satz von Datensätzen abzurufen.

Beispiel: Erkennen von Datenanomalien und Suchen nach einer Erklärung (Funktion RANDOM_CUT_FOREST_WITH_EXPLANATION)

Amazon-Kinesis-Data-Analytics stellt die `RANDOM_CUT_FOREST_WITH_EXPLANATION`-Funktion bereit, die jedem Datensatz basierend auf Werten in den numerischen Spalten eine Anomaliebewertung zuweist. Die Funktion liefert auch eine Erläuterung für die Anomalie. Weitere Informationen finden Sie unter RANDOM_CUT_FOREST_WITH_EXPLANATION in der SQL-Referenz für Amazon-Managed-Service für Apache Flink.

In dieser Übung schreiben Sie Anwendungscode, um Anomaliebewertungen für Datensätzen in der Streaming-Quelle Ihrer Anwendung anzufordern. Sie können auch eine Erklärung für jede Anomalie anfordern.

Themen

Erster Schritt

Schritt 1: Vorbereitung der Daten

Schritt 1: Vorbereitung der Daten

Bevor Sie für dieses Beispiel eine Amazon-Kinesis-Data-Analytics-Anwendung erstellen, erstellen Sie einen Kinesis-Datenstrom zur Verwendung als Streaming-Quelle für Ihre Anwendung. Sie führen auch Python-Code aus, um simulierte Blutdruckdaten in den Stream zu schreiben.

Themen

Schritt 1.1: Erstellen eines Kinesis-Datenstrom

In diesem Abschnitt erstellen Sie einen Kinesis-Datenstrom mit dem Namen `ExampleInputStream`. Sie können diesen Datenstream mit dem AWS-Managementkonsole oder dem erstellen AWS CLI.

- So verwenden Sie die Konsole:

  1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
  2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams). Wählen Sie dann Create Kinesis-Stream (Kinesis-Stream erstellen).
  3. Geben Sie als Name **`ExampleInputStream`** ein. Geben Sie für die Anzahl von Shards **`1`** ein.

- Um den Datenstream AWS CLI zu erstellen, führen Sie alternativ den folgenden Befehl aus:

```
$ aws kinesis create-stream --stream-name ExampleInputStream --shard-count 1
```

Schritt 1.2: Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Schritt führen Sie Python-Code aus, um kontinuierlich Beispieldatensätze zu generieren und in den von Ihnen erstellten Daten-Stream zu schreiben.

1. Installieren Sie Python und pip.

   Informationen zum Installieren von Python finden Sie unter Python.

   Sie können mithilfe von pip Abhängigkeiten installieren. Informationen zum Installieren von pip finden Sie unter Installation in der Dokumentation zu pip.

2. Führen Sie den folgenden Python-Code aus. Sie können zu der Region wechseln, die Sie in diesem Beispiel verwenden möchten. Der Befehl `put-record` im Code schreibt die JSON-Datensätze zum Stream.

```
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"))
```

Nächster Schritt

Schritt 2: Erstellen einer Analyseanwendung

Schritt 2: Erstellen einer Analyseanwendung

In diesem Abschnitt erstellen Sie eine Amazon-Kinesis-Data-Analytics-Anwendung und konfigurieren sie für die Verwendung des Kinesis-Datenstroms, den Sie in the section called “Schritt 1: Vorbereitung der Daten” als Streaming-Quelle erstellt haben. Anschließend führen Sie Anwendungscode aus, von dem die Funktion `RANDOM_CUT_FOREST_WITH_EXPLANATION` genutzt wird.

So erstellen Sie eine Anwendung

1. Öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Navigationsbereich Data Analytics (Datenanalyse) und dann Create application (Anwendung erstellen).
3. Geben Sie einen Anwendungsnamen und (optional) eine Beschreibung ein und wählen Sie Create application.
4. Wählen Sie Streaming-Daten Connect und wählen Sie dann eine Option ExampleInputStreamaus der Liste aus.
5. Wählen Sie Discover schema und vergewissern Sie sich, dass als Spalten `Systolic` und `Diastolic` appear as `INTEGER` angezeigt werden. Wenn Sie über einen weiteren Typ verfügen, wählen Sie Edit schema und weisen Sie beiden den Typ `INTEGER` zu.
6. Wählen Sie unter Real time analytics die Option Go to SQL editor. Wenn Sie dazu aufgefordert werden, bestimmen Sie, dass Ihre Anwendung ausgeführt wird.
7. Fügen Sie den folgenden Code in den SQL-Editor ein und wählen Sie dann 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;
```

Nächster Schritt

Schritt 3: Untersuchen der Ergebnisse

Schritt 3: Untersuchen der Ergebnisse

Wenn Sie den SQL-Code für dieses Beispiel ausführen, werden zuerst Zeilen mit einer Anomaliebewertung gleich null angezeigt. Dies geschieht während des anfänglichen Lernens. Danach sollten die Ergebnisse wie folgt aussehen:

```
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"}}
```

- Der Algorithmus in der Funktion `RANDOM_CUT_FOREST_WITH_EXPLANATION` erkennt, dass die Spalten `Systolic` und `Diastolic` numerisch sind und nutzt sie als Eingabe.

- Da sich in der Spalte `BloodPressureLevel` Text befindet, wird sie vom Algorithmus nicht berücksichtigt. Diese Spalte ist dient nur als visuelle Hilfe, sodass sie normalen, hohen und niedrigen Blutdruckwerte in diesem Beispiel schnell zu erkennen sind.
- In der Spalte `ANOMALY_SCORE` sind Datensätze mit höherer Bewertung anormal. Der zweite Datensatz in diesem Beispiel-Ergebnissatz ist mit einer Anomaliebewertung von 3,855851061 am meisten anormal.
- Um zu verstehen, wie stark jede der numerischen Spalten, die vom Algorithmus berücksichtigt wird, zur Anomaliebewertung beiträgt, nehmen Sie auf das JSON-Feld mit dem Namen `ATTRIBUTION_SCORE` in der Spalte `ANOMALY_SCORE` Bezug. Im Falle der zweiten Zeile in dieser Gruppe von Beispielergebnissen tragen die Spalten `Systolic` und `Diastolic` im Verhältnis von 1,7447:2,1111 zur Anomalie bei. Anders ausgedrückt lässt sich die Begründung der Anomaliebewertung zu 45 Prozent auf den systolischen Wert zurückführen. Der verbleibende Beitrag ist im diastolischen Wert begründet.
- Anhand des JSON-Feldes mit dem Namen `DIRECTION` können Sie bestimmen, in welcher Richtung der durch die zweite Zeile in diesem Beispiel dargestellte Punkt anormal ist. In diesem Fall sind sowohl der diastolische als auch der systolische Wert beide als `HIGH` gekennzeichnet. Dem JSON-Feld mit dem Namen `STRENGTH` können Sie entnehmen, mit welcher Wahrscheinlichkeit diese Richtungen richtig sind. In diesem Beispiel findet der Algorithmus es wahrscheinlicher, dass der diastolische Wert hoch ist. Tatsächlich liegt der normale Wert der diastolischen Messung gewöhnlich bei 60-80 und ein Wert von 123 ist viel höher als erwartet.

Beispiel: Erkennen von Hotspots in einem Stream (HOTSPOTS-Funktion)

Amazon-Kinesis-Data-Analytics bietet die `HOTSPOTS`-Funktion, die Informationen über relativ dichte Bereiche in Ihren Daten finden und ausgeben kann. Weitere Informationen finden Sie unter HOTSPOTS in der SQL-Referenz für Amazon-Managed-Service für Apache Flink.

In dieser Übung schreiben Sie Anwendungscode zum Suchen nach Hotspots in der Streaming-Quelle Ihrer Anwendung. Um die Anwendung einzurichten, führen Sie die folgenden Schritte aus:

1. Einrichten einer Streaming-Quelle – Sie richten einen Kinesis-Stream ein und schreiben wie folgt Beispielkoordinatendaten:

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626528026, "y": 4.648868803193405, "is_hot": "Y"}
```

Das Beispiel stellt ein Python-Skript bereit, mit dem Sie den Stream befüllen können. Die Werte `x` und `y` werden zufällig generiert, dabei sind einige Datensätze um bestimmte Stellen herum gruppiert.

Das `is_hot`-Feld wird als Indikator bereitgestellt, wenn das Skript den Wert absichtlich als Teil eines Hotspots generiert hat. So können Sie ermitteln, ob die Hotspot-Erkennungsfunktion ordnungsgemäß funktioniert.

2. Erstellen der Anwendung – Mithilfe der AWS-Managementkonsole erstellen Sie dann eine Kinesis Data Analytics-Anwendung. Konfigurieren Sie die Anwendungseingabe, indem Sie die Streaming-Quelle einem In-Application-Stream zuweisen (`SOURCE_SQL_STREAM_001`). Wenn die Anwendung gestartet wird, liest Kinesis Data Analytics kontinuierlich die Streaming-Quelle und fügt Datensätze in den In-Application-Stream ein.

   In dieser Übung verwenden Sie den folgenden Code für die Anwendung:

   ```
   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)
       );
   ```

   Der Code liest Zeilen im `SOURCE_SQL_STREAM_001`, analysiert ihn auf wichtige Hotspots hin und schreibt die resultierenden Daten in einen anderen In-Application-Stream (`DESTINATION_SQL_STREAM`). Sie verwenden Pumps, um Zeilen in In-Application-Streams einzufügen. Weitere Informationen finden Sie unter In-Application-Streams und Pumps.

3. Konfigurieren der Ausgabe – Sie konfigurieren die Anwendungsausgabe für das Senden von Daten von der Anwendung an ein externes Ziel, bei dem es sich um einen anderen Kinesis-

Datenstrom handelt. Überprüfen Sie die Hotspot-Ergebnisse und bestimmen Sie, welche Ergebnisse darauf hinweisen, dass ein Hotspot aufgetreten ist (und dass Sie darüber informiert sein müssen). Sie können eine AWS Lambda Funktion verwenden, um Hotspot-Informationen weiter zu verarbeiten und Benachrichtigungen zu konfigurieren.

4. Überprüfen Sie die Ausgabe — Das Beispiel beinhaltet eine JavaScript Anwendung, die Daten aus dem Ausgabestrom liest und sie grafisch anzeigt, sodass Sie die Hotspots, die die Anwendung generiert, in Echtzeit anzeigen können.

Die Übung verwendet die Region USA West (Oregon) (`us-west-2`), um diese Streams und Ihre Anwendung zu erstellen. Wenn Sie eine andere Region verwenden, aktualisieren Sie den Code dementsprechend.

Themen

Schritt 1: Erstellen der Eingabe- und Ausgabe-Streams

Bevor Sie eine Amazon-Kinesis-Data-Analytics-Anwendung für das Hotspots-Beispiel erstellen, erstellen Sie zwei Kinesis-Datenströme. Konfigurieren Sie einen der Streams als Streaming-Quelle für Ihre Anwendung und den anderen Stream als das Ziel, an das Kinesis Data Analytics die Ausgabe Ihrer Anwendung weiterleitet.

Themen

Schritt 1.1: Erstellen der Kinesis-Datenströme

In diesem Abschnitt erstellen Sie zwei Kinesis-Datenströme: `ExampleInputStream` und `ExampleOutputStream`.

Erstellen Sie diese Daten-Streams mithilfe der Konsole oder der AWS CLI.

- So erstellen Sie die Daten-Streams mithilfe der Konsole:

  1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
  2. Klicken Sie im Navigationsbereich auf Data Streams (Daten-Streams).
  3. Klicken Sie auf Create Kinesis stream (Kinesis-Stream erstellen) und erstellen Sie einen Stream mit einer Shard namens `ExampleInputStream`.
  4. Wiederholen Sie den vorherigen Schritt und erstellen Sie einen Stream mit einer Shard namens `ExampleOutputStream`.

- So erstellen Sie einen Daten-Stream mithilfe der AWS CLI:

  - Erstellen Sie Streams (`ExampleInputStream`und`ExampleOutputStream`) mit dem folgenden `create-stream` AWS CLI Kinesis-Befehl. Zum Erstellen des zweiten Streams, den die Anwendung zum Schreiben der Ausgabe verwenden wird, führen Sie den gleichen Befehl aus und ändern Sie den Namen des Streams in `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
    ```

Schritt 1.2: Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Schritt führen Sie Python-Code aus, um kontinuierlich Beispieldatensätze zu generieren und zum `ExampleInputStream`-Stream zu schreiben.

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626580026, "y": 4.648868803193405, "is_hot": "Y"}
```

1. Installieren Sie Python und `pip`.

Informationen zur Installation von Python finden Sie auf der Website für Python.

Sie können mithilfe von pip Abhängigkeiten installieren. Informationen zur Installation von pip finden Sie unter Installation auf der Website für pip.

2. Führen Sie den folgenden Python-Code aus. Dieser Code führt Folgendes aus:

   - Generiert einen potenziellen Hotspot an einer Stelle in der XY-Ebene.
   - Generiert eine Gruppe von 1000 Punkten für jeden Hotspot. Von diesen Punkten werden 20 Prozent rund um den Hotspot gruppiert. Der Rest wird nach dem Zufallsprinzip innerhalb des gesamten Raums generiert.
   - Der `put-record`-Befehl schreibt die JSON-Datensätze in den Stream.

   > **Important**
   >
   > Laden Sie diese Datei nicht auf einen Webserver hoch, da sie Ihre AWS -Anmeldeinformationen enthält.

```
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"),
    )
```

Nächster Schritt

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

Schritt 2: Erstellen Sie die Amazon-Kinesis-Data-Analytics-Anwendung

In diesem Abschnitt des Hotspots-Beispiels erstellen Sie wie folgt eine Kinesis Data Analytics-Anwendung:

- Konfigurieren Sie die Anwendungseingabe für die Verwendung des Kinesis-Datenstroms, den Sie in Schritt 1 als Streaming-Quelle erstellt haben.
- Verwenden Sie den bereitgestellten Anwendungscode in der AWS-Managementkonsole.

So erstellen Sie eine Anwendung

1. Erstellen Sie eine Kinesis Data Analytics-Anwendung-Datenanalyseanwendung, indem Sie die Schritte 1, 2, und 3 der Übung Erste Schritte (siehe Schritt 3.1: Eine Anwendung erstellen) ausführen.

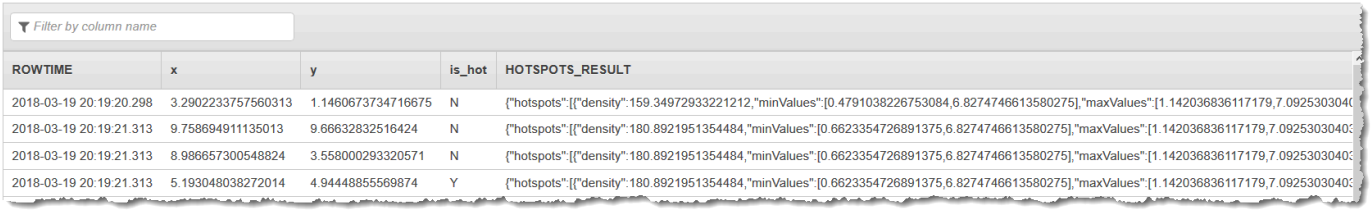
   Führen Sie in der Quellkonfiguration Folgendes aus:

   - Geben Sie die Streaming-Quelle an, die Sie in the section called “Schritt 1: Erstellen von Streams” erstellt haben.
   - Nachdem die Konsole das Schema abgeleitet hat, bearbeiten Sie das Schema. Stellen Sie sicher, dass die Spaltentypen `x` und `y` auf `DOUBLE` festgelegt sind und dass der Spaltentyp `IS_HOT` auf `VARCHAR` festgelegt ist.
2. Verwenden Sie den folgenden Anwendungscode (Sie können diesen Code in den SQL-Editor einfügen):

```
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. Führen Sie den SQL-Code aus und überprüfen Sie die Ergebnisse.



Nächster Schritt

Schritt 3: Konfigurieren der Anwendungsausgabe

Schritt 3: Konfigurieren der Anwendungsausgabe

An diesem Punkt im Hotspots-Beispiel ermittelt der Amazon-Kinesis-Data-Analytics-Anwendungscode bedeutende Hotspots aus einer Streaming-Quelle und weist jedem ein Heat-Ergebnis zu.

Sie können nun das Anwendungsergebnis aus dem In-Application-Stream an ein externes Ziel senden, das ein anderer Kinesis-Datenstrom (`ExampleOutputStream`) ist. Sie können dann die Hotspot-Ergebnisse analysieren und einen entsprechenden Schwellenwert für Hotspot-Heat bestimmen. Sie können diese Anwendung zusätzlich erweitern, um Warnungen zu generieren.

So konfigurieren Sie die Anwendungsausgabe

1. Öffnen Sie die Kinesis Data Analytics Analytics-Konsole unter https://console.aws.amazon.com/kinesisanalytics.
2. Wählen Sie im SQL-Editor Destination oder Add a destination im Anwendungs-Dashboard aus.
3. Wählen Sie auf der Seite Add a destination (Ziel hinzufügen) die Option Select from your streams (Aus Streams auswählen) aus. Wählen Sie dann den `ExampleOutputStream`-Stream, den Sie im vorherigen Abschnitt erstellt haben.

Sie verfügen nun über ein externes Ziel, an das Amazon-Kinesis-Data-Analytics alle Datensätze weiterleitet, die Ihre Anwendung in den In-Application-Stream `DESTINATION_SQL_STREAM` schreibt.

4. Sie können optional konfigurieren, dass der `ExampleOutputStream` Stream überwacht AWS Lambda und Ihnen Warnmeldungen gesendet werden. Weitere Informationen finden Sie unter Verwenden einer Lambda-Funktion als Ausgabe. Sie können auch die Datensätze überprüfen, die Kinesis Data Analytics in das externe Ziel schreibt. Hierbei handelt es sich um den Kinesis-Stream `ExampleOutputStream`, wie unter Schritt 4: Überprüfen der Anwendungsausgabe beschrieben.

Nächster Schritt

Schritt 4: Überprüfen der Anwendungsausgabe

Schritt 4: Überprüfen der Anwendungsausgabe

In diesem Abschnitt des Hotspot-Beispiels richten Sie eine Webanwendung ein, die Hotspot-Informationen in einem Scalable Vector Graphics (SVG)-Steuerelement anzeigt.

1. Erstellen Sie eine Datei `index.html` mit dem folgenden Inhalt:

```
<!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. Erstellen Sie in demselben Verzeichnis eine Datei namens `hotspots_viewer.js` mit folgendem Inhalt. Geben Sie Ihre Anmeldeinformationen und den Namen des Ausgabe-Streams in den bereitgestellten Variablen an.

```
// 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
```

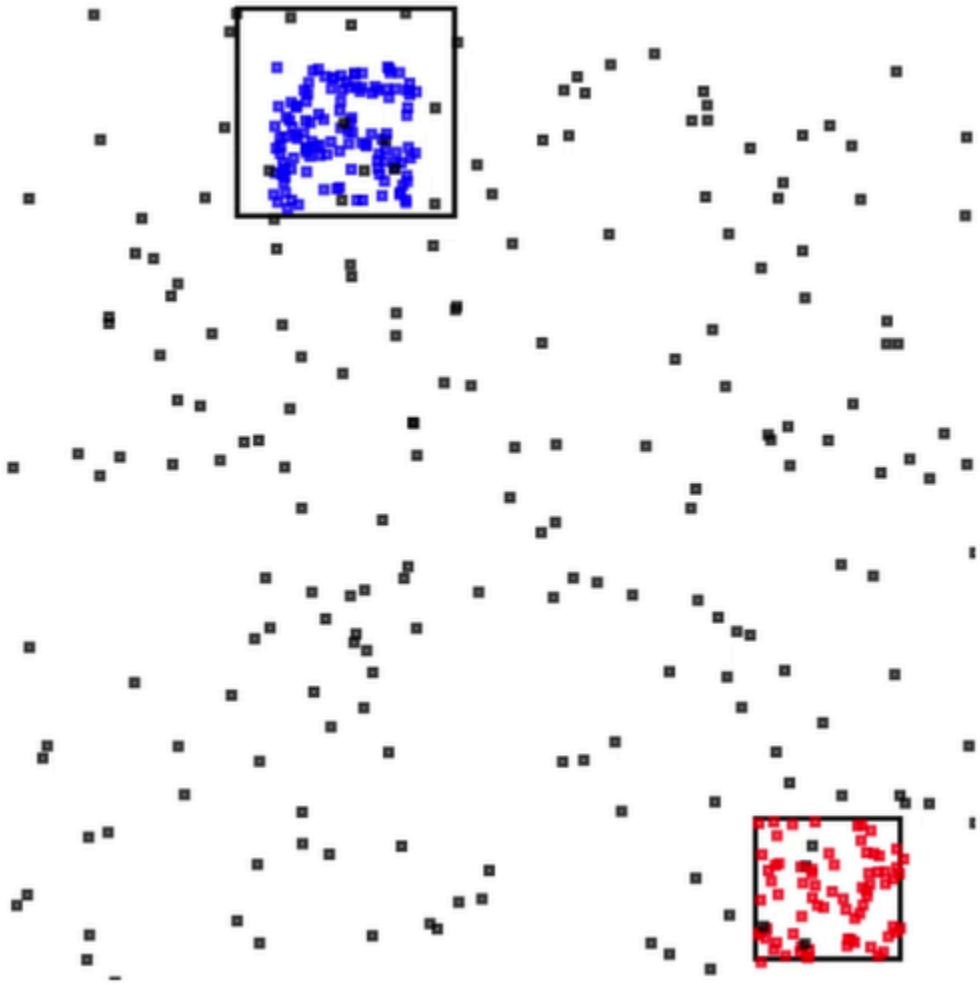
```
    xRange = [0, 10],  // The range of values to display on the x-axis
    yRange = [0, 10];  // The range of values to display on the y-axis

////////////////////////////////////////////////////////////////////////////////
// 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. Öffnen Sie mit dem Python-Code aus dem ersten Abschnitt `index.html` in einem Webbrowser. Wie im Folgenden verdeutlicht wird, erscheinen die Hotspot-Informationen auf der Seite.



Beispiele: Warnungen und Fehler

In diesem Abschnitt finden Sie Beispiele für Kinesis Data Analytics-Anwendungen die Warnungen und Fehlermeldungen nutzen. Jedes Beispiel enthält step-by-step Anweisungen und Code, mit denen Sie Ihre Kinesis Data Analytics Analytics-Anwendung einrichten und testen können.

Themen

- Beispiel: Erstellen einfacher Warnungen
- Beispiel: Erstellen gedrosselter Warnungen
- Beispiel: Erkunden des In-Application-Fehler-Streams

Beispiel: Erstellen einfacher Warnungen

In dieser Kinesis Data Analytics-Anwendung wird die Abfrage kontinuierlich im In-Application-Stream ausgeführt, der über dem Demo-Stream erstellt wird. Weitere Informationen finden Sie unter Kontinuierliche Abfragen.

Wenn Zeilen eine Änderung des Aktienpreises um mehr als 1 % zeigen, werden diese Zeilen in einen anderen In-Application-Stream eingefügt. In der Übung können Sie die Anwendungsausgabe für die Weiterleitung der Ergebnisse an ein externes Ziel konfigurieren. Sie können die Ergebnisse anschließend weiter untersuchen. Sie können beispielsweise eine AWS Lambda Funktion verwenden, um Datensätze zu verarbeiten und Ihnen Benachrichtigungen zu senden.

So erstellen Sie eine Anwendung für einfache Warnungen

1. Erstellen Sie die Analyseanwendung wie in der Übung unter Erste Schritte für Kinesis Data Analytics beschrieben.
2. Ersetzen Sie im SQL-Editor in Kinesis Data Analytics den Anwendungscode durch Folgendes:

```
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;
```

   Die `SELECT`-Anweisung im Anwendungscode filtert Zeilen im `SOURCE_SQL_STREAM_001` nach Aktienpreisänderungen von mehr als 1 %. Sie fügt diese Zeilen dann mittels eines Pump in einen anderen `DESTINATION_SQL_STREAM`-In-Application-Stream ein. Weitere Informationen zum Kodierungsmuster, das die Verwendung von Pumps zum Einfügen von Zeilen in In-Application-Streams beschreibt, finden Sie unter Anwendungscode.
3. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).
4. Fügen Sie ein Ziel hinzu. Zu diesem Zweck wählen Sie entweder die Registerkarte Destination (Ziel) im SQL-Editor oder Add a destination (Ziel hinzufügen) auf der Detailseite der Anwendung.

a. Wählen Sie im SQL-Editor die Registerkarte Destination (Ziel) und danach Connect to a destination (Mit einem Ziel verbinden).

   Klicken Sie auf der Seite Connect to destination (Mit Ziel verbinden) auf Create New (Neu erstellen).

b. Wählen Sie Go to Kinesis Streams.

c. Erstellen Sie in der Amazon-Kinesis-Data-Streams -Konsole einen neuen Kinesis Stream (z. B. `gs-destination`) mit einer Shard. Warten Sie, bis der Status des Streams ACTIVE ist.

d. Kehren Sie zur Kinesis Data Analytics-Konsole zurück. Wählen Sie auf der Seite Connect to destination (Mit Ziel verbinden) den von Ihnen erstellten Stream aus.

   Wenn der Stream nicht angezeigt wird, aktualisieren Sie die Seite.

e. Wählen Sie Save and continue aus.

Sie besitzen nun ein externes Ziel, einen Kinesis-Datenstrom, an dem Kinesis Data Analytics die Ausgabe Ihrer Anwendung im In-App-`DESTINATION_SQL_STREAM`-Stream beibehält.

5. Konfigurieren Sie die Konfiguration AWS Lambda , um den von Ihnen erstellten Kinesis-Stream zu überwachen und eine Lambda-Funktion aufzurufen.

   Detaillierte Anweisungen finden Sie unter Vorverarbeitung von Daten mithilfe einer Lambda-Funktion.

Beispiel: Erstellen gedrosselter Warnungen

In dieser Kinesis Data Analytics-Datenanalyseanwendung wird die Abfrage kontinuierlich im In-App-Stream ausgeführt, der über dem Demo-Stream erstellt wird. Weitere Informationen finden Sie unter Kontinuierliche Abfragen. Wenn Zeilen eine Änderung des Aktienpreises um mehr als 1 % zeigen, werden diese Zeilen in einen anderen In-Application-Stream eingefügt. Die Anwendung drosselt die Warnungen so, dass sofort eine Warnung gesendet wird, wenn sich der Aktienkurs ändert. Allerdings wird nicht mehr als eine Warnung pro Minute pro Aktionssymbol an den In-Application-Stream übermittelt.

So erstellen Sie eine Anwendung für gedrosselte Warnungen

1. Erstellen Sie eine Kinesis Data Analytics-Anwendung wie in der Übung unter Erste Schritte für Kinesis Data Analytics beschrieben.
2. Ersetzen Sie im SQL-Editor in Kinesis Data Analytics den Anwendungscode durch Folgendes:

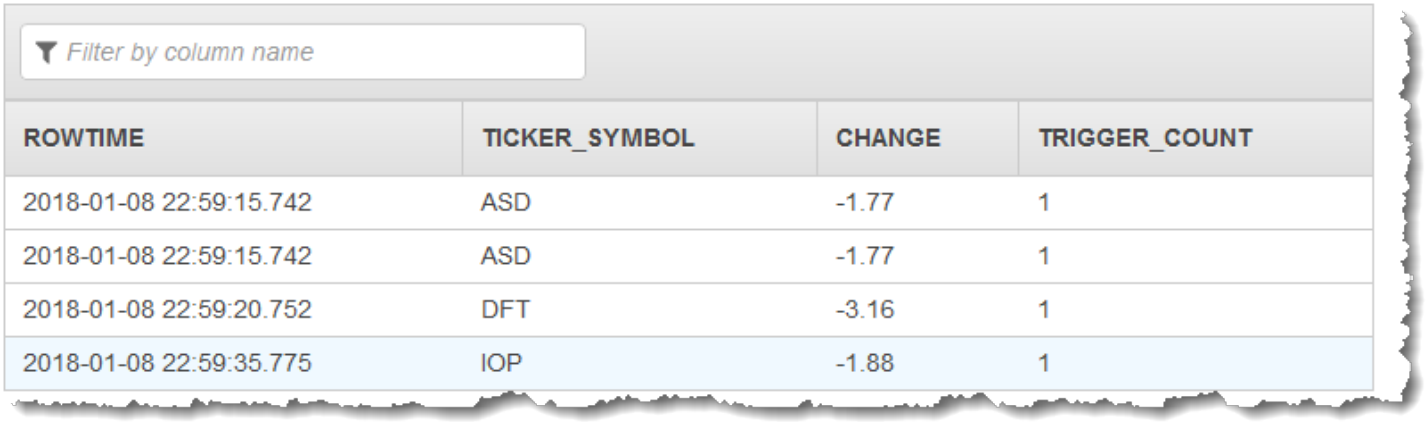
```
CREATE OR REPLACE STREAM "CHANGE_STREAM"
           (ticker_symbol VARCHAR(4),
            sector        VARCHAR(12),
            change        DOUBLE,
            price         DOUBLE);

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;
```

Die `SELECT`-Anweisung im Anwendungscode filtert Zeilen im `SOURCE_SQL_STREAM_001` nach Aktien mit Preisänderungen von mehr als 1 % und fügt diese Zeilen mittels eines Pump in einen anderen `CHANGE_STREAM`-In-Application-Stream ein.

Die Anwendung erstellt dann einen zweiten Stream mit dem Namen `TRIGGER_COUNT_STREAM` für die gedrosselten Warnungen. Eine zweite Abfrage wählt Datensätze aus einem Fenster aus, das jedes Mal in den Vordergrund kommt, wenn darin ein Datensatz aufgenommen wird, sodass nur ein Datensatz pro Börsenticker pro Minute in den Stream geschrieben wird.

3. Klicken Sie auf Save and run SQL (SQL speichern und ausführen).

Das Beispiel gibt einen Stream ähnlich dem folgenden an `TRIGGER_COUNT_STREAM` aus:



ROWTIME	TICKER_SYMBOL	CHANGE	TRIGGER_COUNT
2018-01-08 22:59:15.742	ASD	-1.77	1
2018-01-08 22:59:15.742	ASD	-1.77	1
2018-01-08 22:59:20.752	DFT	-3.16	1
2018-01-08 22:59:35.775	IOP	-1.88	1

Beispiel: Erkunden des In-Application-Fehler-Streams

Amazon-Kinesis-Data-Analytics stellt für jede von Ihnen erstellte Anwendung einen In-App-Fehler-Stream bereit. Alle Zeilen, die Ihre Anwendung nicht verarbeiten kann, werden an diesen Fehler-Stream gesendet. Es besteht die Möglichkeit, die Daten aus dem Fehler-Stream zur näheren Untersuchung dauerhaft an einem externen Ziel zu speichern.

Sie führen die folgenden Übungen in der Konsole aus. In diesen Beispielen erzeugen Sie Fehler in der Eingangskonfiguration, indem Sie das bei dem Erkennungsprozess abgeleitete Schema bearbeiten. Dann prüfen Sie die an den Fehler-Stream gesendeten Zeilen.

Themen

- Erzeugen eines Analysefehlers
- Erzeugen eines "Division durch Null"-Fehlers

Erzeugen eines Analysefehlers

In dieser Übung erzeugen Sie einen Analysefehler.

1. Erstellen Sie eine Kinesis Data Analytics-Anwendung wie in der Übung unter Erste Schritte für Kinesis Data Analytics beschrieben.
2. Wählen Sie auf der Detailseite der Anwendung Connect streaming data (Streaming-Daten verbinden).
3. Wenn Sie die Einstiegsübung absolviert haben, ist ein Demo-Stream (`kinesis-analytics-demo-stream`) in Ihrem Konto vorhanden. Wählen Sie auf der Seite Connect to source (Mit Quelle verbinden) diesen Demo-Stream aus.
4. Kinesis Data Analytics leitet anhand von Beispieldaten aus dem Demo-Stream ein Schema für den zu erstellenden In-App-Eingabe-Stream ab. Die Konsole zeigt das abgeleitete Schema und Beispieldaten auf der Registerkarte Formatted stream sample an.
5. Als Nächstes bearbeiten Sie das Schema und ändern den Spaltentyp, um den Analysefehler zu erzeugen. Wählen Sie Edit schema (Schema bearbeiten).
6. Ändern Sie die Spalte `TICKER_SYMBOL` von `VARCHAR(4)` zu `INTEGER`.

   Da der Spaltentyp des zu erstellenden anwendungsinternen Schemas nun ungültig ist, kann Kinesis Data Analytics die Daten nicht an den In-App-Stream übermitteln. Die Zeilen werden stattdessen zum Fehler-Stream gesendet.
7. Wählen Sie Save schema.
8. Wählen Sie Refresh schema samples.

   Beachten Sie, dass es im Beispiel Formatted stream keine Zeilen gibt. Die Registerkarte Error stream zeigt jedoch Daten mit einer Fehlermeldung an. Die Registerkarte Error Stream zeigt die an den In-Application-Fehler-Stream gesendeten Daten an.

   Da Sie den Datentyp der Spalte geändert haben, konnte Kinesis Data Analytics die Daten nicht an den In-App-Eingabe-Stream senden. Stattdessen wurden die Daten an den Fehler-Stream gesendet.

Erzeugen eines "Division durch Null"-Fehlers

In dieser Übung aktualisieren Sie den Anwendungscode, um einen Laufzeitfehler (Division durch Null) zu verursachen. Beachten Sie, dass Amazon-Kinesis-Data-Analytics die resultierenden Zeilen zum

anwendungsinternen Fehler-Stream und nicht zum In-App-`DESTINATION_SQL_STREAM`-Stream sendet, in den die Ergebnisse geschrieben werden sollen.

1. Erstellen Sie eine Kinesis Data Analytics-Anwendung wie in der Übung unter Erste Schritte für Kinesis Data Analytics beschrieben.

   Überprüfen Sie die Ergebnisse auf der Registerkarte Real-time analytics wie folgt:

   Sour

2. Aktualisieren Sie die Anweisung `SELECT` im Anwendungscode, um einen "Division durch Null"-Fehler zu erzeugen, z. B.:

   ```
   SELECT STREAM ticker_symbol, sector, change, (price / 0) as ProblemColumn
   FROM "SOURCE_SQL_STREAM_001"
   WHERE sector SIMILAR TO '%TECH%';
   ```

3. Führen Sie die Anwendung aus.

   Aufgrund von Laufzeitfehlern durch die Division durch Null sendet Kinesis Data Analytics Zeilen zum In-App-Fehler-Stream, anstatt die Ergebnisse zum `DESTINATION_SQL_STREAM` zu schreiben. Wählen Sie auf der Registerkarte Real-time analytics (Echtzeitanalyse) den Fehler-Stream aus. Anschließend werden die Zeilen im In-Application-Fehler-Stream angezeigt.

Beispiele: Lösungsvorlagen

Auf der AWS Solutions Site stehen AWS CloudFormation Vorlagen zur Verfügung, mit denen Sie schnell komplette Streaming-Datenlösungen erstellen können.

Die folgenden Vorlagen sind verfügbar:

Einblicke in AWS-Konto Aktivitäten in Echtzeit

Diese Lösung erfasst und visualisiert Kennzahlen zum Ressourcenzugriff und zur Nutzung AWS-Konto Ihrer Ressourcen in Echtzeit. Weitere Informationen finden Sie unter Echtzeit-Einblicke in AWS-Konto Aktivitäten.

AWS IoT Geräteüberwachung in Echtzeit mit Kinesis Data Analytics

Diese Lösung sammelt, verarbeitet, analysiert und visualisiert Daten zur IoT-Gerätekonnektivität und -Aktivität in Echtzeit. Weitere Informationen finden Sie unter AWS IoT Geräteüberwachung in Echtzeit mit Kinesis Data Analytics.

Echtzeit-Web Analytics mit Kinesis Data Analytics

Diese Lösung sammelt, verarbeitet, analysiert und visualisiert Website-Clickstream-Daten in Echtzeit. Weitere Informationen finden Sie unter Echtzeit-Web-Analysen mit Kinesis Data Analytics.

Amazon-Connected-Vehicle-Solution

Diese Lösung sammelt, verarbeitet, analysiert und visualisiert IoT-Daten von Fahrzeugen in Echtzeit. Weitere Informationen finden Sie unter Amazon-Connected-Vehicle-Solution.

Sicherheit in Amazon Kinesis Data Analytics

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die auf die Anforderungen der sicherheitssensibelsten Unternehmen zugeschnitten sind.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS -Compliance-Programms getestet und überprüft. Informationen zu den Compliance-Programmen, die für Kinesis Data Analytics gelten, finden Sie unter AWS Services in Scope by Compliance Program.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Kinesis Data Analytics anwenden können. In den folgenden Themen erfahren Sie, wie Sie Kinesis Data Analytics konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere Amazon-Services nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer Kinesis Data Analytics Analytics-Ressourcen helfen können.

Topics

Datenschutz in Amazon-Kinesis-Data-Analytics for SQL-Anwendungen

Sie können Ihre Daten mithilfe von Tools schützen, die von AWS bereitgestellt werden. Kinesis Data Analytics kann mit Diensten zusammenarbeiten, die die Verschlüsselung von Daten unterstützen, darunter Kinesis Data Streams, Firehose und Amazon S3.

Datenverschlüsselung in Kinesis Data Analytics

Verschlüsselung im Ruhezustand

Beachten Sie die folgenden Informationen zur Verschlüsselung von Daten im Ruhezustand mit Kinesis Data Analytics:

- Sie können Daten im eingehenden Kinesis-Datenstrom mit verschlüsseln. StartStreamEncryption Weitere Informationen finden Sie unter Was bedeutet eine serverseitige Verschlüsselung in Kinesis-Daten-Streams?.
- Ausgabedaten können im Ruhezustand mit Firehose verschlüsselt werden, um Daten in einem verschlüsselten Amazon S3 S3-Bucket zu speichern. Sie können den Schlüssel angeben, der von Ihrem Amazon-S3-Bucket zur Verschlüsselung verwendet wird. Weitere Informationen hierzu finden Sie unter Daten schützen durch serverseitige Verschlüsselung mit von KMS verwalteten Schlüsseln (SSE-KMS).
- Der Code Ihrer Anwendung wird im Ruhezustand verschlüsselt.
- Die Referenzdaten Ihrer Anwendung werden im Ruhezustand verschlüsselt.

Verschlüsselung während der Übertragung

Kinesis Data Analytics verschlüsselt alle Daten während der Übertragung. Die Verschlüsselung während der Übertragung ist für alle Kinesis Data Analytics-Anwendungen aktiviert und kann nicht deaktiviert werden.

Kinesis Data Analytics verschlüsselt Daten während der Übertragung bei den folgenden Szenarien:

- Daten während der Übertragung von Kinesis Data Streams an Kinesis Data Analytics.
- Daten während der Übertragung zwischen internen Komponenten innerhalb von Kinesis Data Analytics.

- Daten, die zwischen Kinesis Data Analytics und Firehose übertragen werden.

Schlüsselverwaltung

Bei der Datenverschlüsselung in Kinesis Data Analytics werden vom Service verwaltete Schlüssel verwendet. Vom Kunden verwaltete Schlüssel werden nicht unterstützt.

Identitäts- und Zugriffsverwaltung in Kinesis Data Analytics

Amazon-Kinesis-Data-Analytics benötigt zum Lesen von Datensätzen aus einer Streaming-Quelle die Berechtigungen, die Sie in der Konfiguration Ihrer Anwendungseingabe angeben. Darüber hinaus benötigt Amazon-Kinesis-Data-Analytics auch Berechtigungen, um Ihre Anwendungsausgabe in Streams zu schreiben, die Sie in der Konfiguration Ihrer Anwendungsausgabe angeben.

Sie können diese Berechtigungen durch Erstellen einer IAM-Rolle erteilen, die Amazon-Kinesis-Data-Analytics übernehmen kann. Die Berechtigungen, die Sie dieser Rolle erteilen, legen fest, welche Maßnahmen Amazon-Kinesis-Data-Analytics durchführen kann, wenn der Service die Rolle übernimmt.

> **Note**
>
> Die Informationen in diesem Abschnitt sind hilfreich, wenn Sie selbst eine IAM-Rolle erstellen möchten. Wenn Sie in der Amazon-Kinesis-Data-Analytics-Konsole eine Anwendung erstellen, kann die Konsole zu diesem Zeitpunkt eine IAM-Rolle für Sie erstellen. Die Konsole verwendet die folgenden Namenskonvention für IAM-Rollen, die von ihr erstellt werden:
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> Nachdem die Rolle erstellt wurde, können Sie die Rolle und die angefügten Richtlinien in der IAM-Konsole überprüfen.

Jeder IAM-Rolle sind zwei Richtlinien angefügt. In der Vertrauensrichtlinie geben Sie an, wer die Rolle annehmen kann. In der Berechtigungsrichtlinie (es kann mehrere geben) geben Sie die Berechtigungen an, die Sie der betreffenden Rolle erteilen möchten. In den folgenden Abschnitten werden diese Richtlinien beschrieben. Sie können diese bei der Erstellung von IAM-Rollen verwenden.

Vertrauensrichtlinie

Um Amazon-Kinesis-Data-Analytics Berechtigungen für die Annahme einer Rolle für den Zugriff auf eine Streaming- oder Referenzquelle zu gewähren, können Sie die folgende Vertrauensrichtlinie an eine IAM-Rolle anfügen:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Berechtigungsrichtlinie

Wenn Sie eine IAM-Rolle erstellen, um Amazon-Kinesis-Data-Analytics das Lesen aus der Streaming-Quelle einer Anwendung zu ermöglichen, müssen Sie relevanten Leseaktionen Berechtigungen gewähren. Abhängig von Ihrer Quelle (z. B. einem Kinesis-Stream, einem Firehose-Lieferstream oder einer Referenzquelle in einem Amazon S3 S3-Bucket) können Sie die folgende Berechtigungsrichtlinie anhängen.

Berechtigungsrichtlinie für das Lesen eines Kinesis-Streams

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "ReadInputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:123456789012:stream/inputStreamName"
            ]
        }
    ]
}
```

Berechtigungsrichtlinie für das Lesen eines Firehose-Lieferstreams

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:Get*"
            ],
            "Resource": [
                "arn:aws:firehose:us-
east-1:123456789012:deliverystream/inputFirehoseName"
            ]
        }
    ]
}
```

> **Note**
>
> Die `firehose:Get*`-Berechtigung bezieht sich auf eine interne Zugriffsmethode, die Kinesis Data Analytics für den Zugriff auf den Stream verwendet. Es gibt keinen öffentlichen Zugang für einen Firehose-Lieferstream.

Wenn Sie Amazon-Kinesis-Data-Analytics in der Ausgabekonfiguration Ihrer Anwendung anweisen, Ausgaben zu externen Zielen zu schreiben, müssen Sie der IAM-Rolle die folgende Berechtigung gewähren.

Berechtigungsrichtlinie für das Schreiben an einen Kinesis-Stream

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:123456789012:stream/output-stream-
name"
            ]
        }
    ]
}
```

Berechtigungsrichtlinie für das Schreiben zu einem Firehose-Bereitstellungs-Stream

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/output-
firehose-name"
            ]
        }
    ]
}
```

Berechtigungsrichtlinie für das Lesen einer Referenzdatenquelle aus einem Amazon-S3-Bucket

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

Serviceübergreifende Confused-Deputy-Prävention

In kann es zu einem dienstübergreifenden Identitätswechsel kommen AWS, wenn ein Dienst (der anrufende Dienst) einen anderen Dienst (den aufgerufenen Dienst) aufruft. Der aufrufende Service kann so manipuliert werden, dass er auf die Ressourcen eines anderen Kunden reagiert, obwohl er nicht über die entsprechenden Berechtigungen verfügen sollte, was zu einem Verwirrter-Stellvertreter-Problem führt.

Um zu verhindern, dass Abgeordnete verwirrt werden, AWS bietet dieses Tool Tools, mit denen Sie Ihre Daten für alle Dienste schützen können. Dabei werden Dienstprinzipale verwendet, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde. Dieser Abschnitt konzentriert sich speziell bezüglich Kinesis Data Analytics auf die dienstübergreifende Vermeidung verwirrter Stellvertreter. Weitere Informationen zu diesem Thema finden Sie im IAM-Benutzerhandbuch im Abschnitt Das Verwirrte-Stellvertreter-Problem.

Im Kontext von Kinesis Data Analytics for SQL empfehlen wir, die SourceAccount globalen Bedingungsschlüssel aws: SourceArn und aws: in Ihrer Rollenvertrauensrichtlinie zu verwenden, um den Zugriff auf die Rolle nur auf die Anfragen zu beschränken, die von den erwarteten Ressourcen generiert werden.

Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der Wert von `aws:SourceArn` muss der ARN der von Kinesis Data Analytics verwendeten Ressource sein, für den das folgende Format festgelegt ist: `arn:aws:kinesisanalytics:region:account:resource`.

Der effektivste Weg, um sich vor dem Verwirrten-Stellvertreter-Problem zu schützen, ist die Verwendung des globalen Bedingungskontextschlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource.

Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen `aws:SourceArn`-Schlüssel mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel: `arn:aws:kinesisanalytics::111122223333:*`.

Die meisten Aktionen in der Kinesis Data Analytics for SQL API CreateApplication, z. B. AddApplicationInputund DeleteApplicationwerden im Kontext bestimmter Anwendungen ausgeführt, die DiscoverInputSchemaAktion wird jedoch nicht im Kontext einer Anwendung ausgeführt. Das

bedeutet, dass es der in dieser Aktion verwendeten Rolle nicht erlaubt ist, eine Ressource im `SourceArn`-Bedingungsschlüssel vollständig zu spezifizieren. Es folgt ein Beispiel, das einen Platzhalter-ARN verwendet:

```
{
   ...
   "ArnLike":{
      "aws:SourceArn":"arn:aws:kinesisanalytics:us-east-1:123456789012:*"
   }
   ...
}
```

Die von Kinesis Data Analytics for SQL generierte Standardrolle verwendet diesen Platzhalter. Dadurch wird sichergestellt, dass die Erkennung des Eingabeschemas in der Konsolenumgebung reibungslos funktioniert. Wir empfehlen jedoch, die Vertrauensrichtlinie so zu bearbeiten, dass ein vollständiger ARN verwendet wird, nachdem das Schema erkannt wurde, um die vollständige Abwehr von verwirrten Stellvertretern zu implementieren.

Richtlinien für Rollen, die Sie Kinesis Data Analytics zur Verfügung stellen, sowie Vertrauensrichtlinien für Rollen, die für Sie generiert wurden, können die SourceAccount Bedingungsschlüssel aws: SourceArn und aws: verwenden.

Um sich vor dem Confused-Deputy-Problem zu schützen, führen Sie die folgenden Schritte durch:

Schutz vor dem Verwirrter-Stellvertreter-Problem

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie Rollen und dann die Rolle aus, die Sie ändern möchten.
3. Wählen Sie Vertrauensrichtlinie bearbeiten aus.
4. Ersetzen Sie auf der Seite Vertrauensrichtlinie bearbeiten die Standard-JSON-Richtlinie durch eine Richtlinie, die einen oder beide der globalen Bedingungskontextschlüssel `aws:SourceArn` und die `aws:SourceAccount` verwendet. Im Folgenden ist eine Beispielrichtlinie aufgeführt:
5. Wählen Sie Richtlinie aktualisieren.

   JSON

```
{
```

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

Authentifizierung und Zugriffskontrolle für

Für den Zugriff auf sind Anmeldeinformationen erforderlich. Diese Anmeldeinformationen müssen über Berechtigungen für den Zugriff auf AWS Ressourcen wie eine Anwendung oder eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance verfügen. In den folgenden Abschnitten wird beschrieben, wie Sie mithilfe von AWS Identity and Access Management (IAM) und dauerhaft Zugriff auf Ihre Ressourcen erhalten können.

Zugriffskontrolle

Auch wenn Sie über gültige Anmeldeinformationen zur Authentifizierung Ihrer Anfragen verfügen, können Sie die -Ressourcen nur mit entsprechenden Berechtigungen erstellen oder darauf zugreifen. So benötigen Sie beispielsweise Berechtigungen zum Erstellen einer -Anwendung.

In den folgenden Abschnitten wird die Verwaltung von Berechtigungen für beschrieben. Wir empfehlen Ihnen, zunächst die Übersicht zu lesen.

- Übersicht über die Verwaltung von Zugriffsberechtigungen für Ihre -Ressourcen
- Verwenden von identitätsbasierten Richtlinien (IAM-Richtlinien) für AWS Organizations

- API-Berechtigungen: Aktionen, Berechtigungen und Ressourcenreferenz

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln oder indem Sie eine AWS Oder-API-Operation AWS CLI aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Übersicht über die Verwaltung von Zugriffsberechtigungen für Ihre -Ressourcen

> **Warning**
>
> Für neue Projekte empfehlen wir, den neuen Managed Service für Apache Flink Studio anstelle von SQL-Anwendungen zu verwenden. Der Managed Service für Apache Flink Studio kombiniert Benutzerfreundlichkeit mit fortschrittlichen Analysefunktionen, sodass Sie in wenigen Minuten anspruchsvolle Anwendungen zur Stream-Verarbeitung erstellen können.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

> **Note**
>
> Ein Kontoadministrator (oder Administratorbenutzer) ist ein Benutzer mit Administratorrechten. Weitere Informationen finden Sie unter Bewährte Methoden für IAM im IAM-Benutzerhandbuch.

Themen

Ressourcen und Operationen

Die primäre Ressource ist eine Anwendung. In einer Richtlinie identifizieren Sie die Ressource, für welche die Richtlinie gilt, mithilfe eines Amazon-Ressourcennamens (ARN).

Diesen Ressourcen sind eindeutige Amazon-Ressourcennamen (ARNs) zugeordnet, wie in der folgenden Tabelle dargestellt.

Ressourcentyp	ARN-Format
Anwendung	`arn:aws:kinesisanalytics:` *`region`*`:`*`account-id`*`:application/` *`application-name`*

bietet eine Reihe von Operationen für die Arbeit mit Ressourcen. Eine Liste der verfügbaren Operationen finden Sie unter Aktionen.

Grundlegendes zum Eigentum an Ressourcen

Der AWS-Konto besitzt die Ressourcen, die im Konto erstellt wurden, unabhängig davon, wer die Ressourcen erstellt hat. Insbesondere ist der Ressourcenbesitzer derjenige AWS-Konto der Prinzipalentität (d. h. das Root-Konto, ein Benutzer oder eine IAM-Rolle), die die Anfrage zur Ressourcenerstellung authentifiziert. Die Funktionsweise wird anhand der folgenden Beispiele deutlich:

- Wenn Sie Ihre Root-Kontoanmeldedaten verwenden, AWS-Konto um eine Anwendung zu erstellen, AWS-Konto sind Sie der Eigentümer der Ressource. (In stellt die Ressource eine Anwendung dar.)
- Wenn Sie in Ihrem einen Benutzer erstellen AWS-Konto und diesem Benutzer die Berechtigung zum Erstellen einer Anwendung erteilen, kann der Benutzer eine Anwendung erstellen. Ihre AWS-Konto, zu der der Benutzer gehört, besitzt jedoch die Anwendungsressource. Es wird dringend empfohlen, Rollen und nicht Benutzern Berechtigungen zu erteilen.
- Wenn Sie in Ihrem System eine IAM-Rolle AWS-Konto mit den Berechtigungen zum Erstellen einer Anwendung erstellen, kann jeder, der die Rolle übernehmen kann, eine Anwendung erstellen. Ihre AWS-Konto, zu der der Benutzer gehört, besitzt die Anwendungsressource.

Verwalten des Zugriffs auf Ressourcen

Eine Berechtigungsrichtlinie beschreibt, wer Zugriff auf welche Objekte hat. Im folgenden Abschnitt werden die verfügbaren Optionen zum Erstellen von Berechtigungsrichtlinien erläutert.

> **Note**
>
> Dieser Abschnitt behandelt die Verwendung von IAM um Zusammenhang mit . Er enthält keine detaillierten Informationen über den IAM-Service. Eine umfassende IAM-Dokumentation finden Sie unter Was ist IAM? im IAM-Benutzerhandbuch. Informationen zur IAM-

Richtliniensyntax und Beschreibungen finden Sie in der IAM-JSON-Richtlinienreferenz im IAM-Benutzerhandbuch.

An eine IAM-Identität angefügte Richtlinien werden als identitätsbasierte Richtlinien (IAM-Richtlinien) bezeichnet. An Ressourcen angefügte Berechtigungsrichtlinien werden als ressourcenbasierte Richtlinien bezeichnet. Unterstützt nur identitätsbasierte Richtlinien (IAM-Richtlinien).

Themen

- Identitätsbasierte Richtlinien (IAM-Richtlinien)
- Ressourcenbasierte Richtlinien

Identitätsbasierte Richtlinien (IAM-Richtlinien)

Richtlinien können IAM-Identitäten angefügt werden. Sie können z. B. Folgendes tun:

- Eine Berechtigungsrichtlinie einem Benutzer oder einer Gruppe in Ihrem Konto anfügen – Um einem Benutzer die Berechtigung zum Erstellen einer -Ressource wie z. B. einer Anwendung zu erteilen, können Sie einem Benutzer oder einer Gruppe, der der Benutzer angehört, eine Berechtigungsrichtlinie anfügen.
- Einer Rolle eine Berechtigungsrichtlinie zuweisen (kontoübergreifende Berechtigungen gewähren) – Sie können einer IAM-Rolle eine identitätsbasierte Berechtigungsrichtlinie zuweisen, um kontoübergreifende Berechtigungen zu erteilen. Der Administrator in Konto A kann beispielsweise wie folgt eine Rolle erstellen, um einem anderen AWS-Konto (z. B. Konto B) oder einem Amazon-Service kontoübergreifende Berechtigungen zu gewähren:
  1. Der Administrator von Konto A erstellt eine IAM-Rolle und fügt ihr eine Berechtigungsrichtlinie an, die Berechtigungen für Ressourcen in Konto A erteilt.
  2. Der Administrator von Konto A weist der Rolle eine Vertrauensrichtlinie zu, die Konto B als den Prinzipal identifiziert, der die Rolle übernehmen kann.
  3. Der Administrator von Konto B kann nun Berechtigungen zur Übernahme der Rolle an alle Benutzer in Konto B delegieren. Daraufhin können die Benutzer in Konto B auf Ressourcen in Konto A oder diese erstellen. Der Prinzipal in der Vertrauensrichtlinie kann auch ein Amazon-Service-Prinzipal sein. Somit können Sie auch einem Amazon-Service die Berechtigungen zur Übernahme der Rolle erteilen.

Weitere Informationen zum Delegieren von Berechtigungen mithilfe von IAM finden Sie unter Zugriffsverwaltung im IAM-Benutzerhandbuch.

Es folgt ein Beispiel für eine Richtlinie, die der `kinesisanalytics:CreateApplication` -Aktion die Berechtigung erteilt, die zum Erstellen einer Anwendung benötigt wird.

> **Note**
>
> Die Richtlinie in diesem Beispiel dient nur der Veranschaulichung. Wenn Sie die Richtlinie an den Benutzer anhängen, kann der Benutzer mithilfe des AWS SDK AWS CLI oder eine Anwendung erstellen. Aber der Benutzer benötigt weitere Berechtigungen, um Ein- und Ausgabe zu konfigurieren. Außerdem benötigt der Benutzer weitere Berechtigungen zum Arbeiten mit der Konsole. Weitere Informationen finden Sie in anderen Abschnitten weiter unten.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Weitere Informationen zur Verwendung von identitätsbasierten Richtlinien mit kontenübergreifendem Zugang finden Sie unter Verwenden von identitätsbasierten Richtlinien (IAM-Richtlinien) für AWS

Organizations. Weitere Informationen zu Benutzern, Gruppen, Rollen und Berechtigungen finden Sie im Thema Identitäten (Benutzer, Gruppen und Rollen) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Andere Services, z. B. Amazon-S3, unterstützen auch ressourcenbasierte Berechtigungsrichtlinien. Beispielsweise können Sie einem S3-Bucket eine ressourcenbasierte Richtlinie zuweisen, um die Zugriffsberechtigungen für diesen Bucket zu verwalten. bietet keine Unterstützung für ressourcenbasierte Richtlinien.

Festlegen der Richtlinienelemente: Aktionen, Effekte und Prinzipale

Für jede -Ressource definiert der Service eine Reihe von API-Operationen. Zur Erteilung von Berechtigungen für diese API-Operationen definiert -Aktionen, die Sie in einer Richtlinie angeben können. Einige API-Operationen erfordern möglicherweise Berechtigungen für mehr als eine Aktion, um die API-Operation auszuführen. Weitere Informationen zu Ressourcen und API-Operationen finden Sie unter Ressourcen und Operationen und Aktionen.

Grundlegende Richtlinienelemente:

- Ressource – Sie verwenden einen Amazon-Ressourcennamen (ARN), um die Ressource, für die die Richtlinie gilt, zu identifizieren. Weitere Informationen finden Sie unter Ressourcen und Operationen.
- Aktion – Mit Aktionsschlüsselwörtern geben Sie die Ressourcenoperationen an, die Sie zulassen oder verweigern möchten. Sie können beispielsweise `create` verwenden, um Benutzern zu erlauben, eine Anwendung zu erstellen.
- Effekt – Die von Ihnen festgelegte Auswirkung (entweder Zugriffserlaubnis oder Zugriffsverweigerung), wenn ein Benutzer die jeweilige Aktion anfordert. Wenn Sie den Zugriff auf eine Ressource nicht ausdrücklich gestatten („Allow“), wird er automatisch verweigert. Sie können den Zugriff auf eine Ressource auch explizit verweigern. So können Sie sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.
- Prinzipal – In identitätsbasierten Richtlinien (IAM-Richtlinien) ist der Benutzer, dem die Richtlinie zugewiesen ist, automatisch der Prinzipal. In ressourcenbasierten Richtlinien müssen Sie den Benutzer, das Konto, den Service oder die sonstige Entität angeben, die die Berechtigungen erhalten soll (gilt nur für ressourcenbasierte Richtlinien). bietet keine Unterstützung für ressourcenbasierte Richtlinien.

Weitere Informationen zur Syntax sowie Beschreibungen von IAM-Richtlinien finden Sie in der -IAM-JSON-Richtlinienreferenz im IAM-Benutzerhandbuch.

Eine mit einer Liste von allen API-Operationen und den Ressourcen, für welche diese gelten, finden Sie unter API-Berechtigungen: Aktionen, Berechtigungen und Ressourcenreferenz.

Angeben von Bedingungen in einer Richtlinie

Beim Erteilen von Berechtigungen können Sie mithilfe der Sprache der Zugriffsrichtlinie die Bedingungen angeben, wann die Richtlinie wirksam werden soll. Beispielsweise kann festgelegt werden, dass eine Richtlinie erst ab einem bestimmten Datum gilt. Weitere Informationen zum Angeben von Bedingungen in einer Richtliniensyntax finden Sie im Thema Bedingung im IAM Benutzerhandbuch.

Bedingungen werden mithilfe vordefinierter Bedingungsschlüssel formuliert. Für gibt es keine speziellen Bedingungsschlüssel. Es gibt jedoch Bedingungsschlüssel für AWS alle Bereiche, die Sie je nach Bedarf verwenden können. Eine vollständige Liste der AWS-weiten Schlüssel finden Sie unter Verfügbare Schlüssel für Bedingungen im IAM-Benutzerhandbuch.

Verwenden von identitätsbasierten Richtlinien (IAM-Richtlinien) für AWS Organizations

Die folgenden Beispiele zu identitätsbasierten Richtlinien verdeutlichen, wie ein Kontoadministrator IAM-Identitäten (d. h. Benutzern, Gruppen und Rollen) Berechtigungsrichtlinien anfügen und somit Berechtigungen zum Durchführen von Operationen mit Ressourcen erteilen kann.

> **Important**
>
> Wir empfehlen Ihnen, zunächst die einführenden Themen zu lesen, in denen die Grundkonzepte und verfügbaren Optionen zum Verwalten des Zugriffs auf Ihre -Ressourcen erläutert werden. Weitere Informationen finden Sie unter Übersicht über die Verwaltung von Zugriffsberechtigungen für Ihre -Ressourcen.

Themen

- Erforderliche Berechtigungen für die Verwendung der Konsole
- Von Amazon verwaltete (vordefinierte) Richtlinien für Auftragsfunktionen
- Beispiele für vom Kunden verwaltete Richtlinien

Dies ist ein Beispiel für eine Berechtigungsrichtlinie.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Die Richtlinie enthält eine Anweisung:

- Die erste Anweisung erteilt Berechtigungen für eine Aktion (`kinesisanalytics:CreateApplication`) für eine Ressource unter Verwendung des Amazon-Ressourcennamens (ARN) der Anwendung. Der ARN gibt in diesem Fall einen Platzhalter (*) an. Dies bedeutet, dass die Berechtigung für jede beliebige Ressource erteilt wird.

Eine Tabelle mit allen API-Operationen und den Ressourcen, für die diese gelten, finden Sie unter API-Berechtigungen: Aktionen, Berechtigungen und Ressourcenreferenz.

Erforderliche Berechtigungen für die Verwendung der Konsole

Sie müssen einem Benutzer die erforderlichen Berechtigungen erteilen, damit dieser mit der Konsole arbeiten kann. Wenn ein Benutzer beispielsweise die Berechtigung zum Erstellen einer Anwendung haben soll, müssen Sie die entsprechenden Berechtigungen erteilen, damit dem Benutzer die Streaming-Quellen im Konto angezeigt werden und der Benutzer in der Konsole die Ein- und Ausgabe konfigurieren kann.

Wir empfehlen Folgendes:

- Verwenden Sie zum Erteilen von Benutzerberechtigungen die von Amazon verwalteten Richtlinien. Eine Aufstellung der verfügbaren Richtlinien finden Sie unter Von Amazon verwaltete (vordefinierte) Richtlinien für Auftragsfunktionen.
- Erstellen Sie benutzerdefinierte Richtlinien. In diesem Fall empfehlen wir, sich das Beispiel in diesem Abschnitt anzuschauen. Weitere Informationen finden Sie unter Beispiele für vom Kunden verwaltete Richtlinien.

Von Amazon verwaltete (vordefinierte) Richtlinien für Auftragsfunktionen

AWS adressiert viele gängige Anwendungsfälle durch die Bereitstellung eigenständiger IAM-Richtlinien, die von erstellt und verwaltet werden. AWS Diese von Amazon verwalteten Richtlinien erteilen die erforderlichen Berechtigungen für viele häufige Anwendungsfälle, sodass Sie nicht mühsam ermitteln müssen, welche Berechtigungen erforderlich sind. Weitere Informationen finden Sie unter Von Amazon verwaltete Richtlinien im IAM-Benutzerhandbuch.

Die folgenden von Amazon verwalteten Richtlinien, die Sie Benutzern in Ihrem Konto zuweisen können, sind spezifisch für:

- **`AmazonKinesisAnalyticsReadOnly`**— Erteilt Berechtigungen für Aktionen, die es einem Benutzer ermöglichen, Anwendungen aufzulisten und die input/output Konfiguration zu überprüfen. Es gewährt auch Berechtigungen, die es einem Benutzer ermöglichen, eine Liste von Kinesis-Streams und Firehose-Lieferstreams anzuzeigen. Während die Anwendung ausgeführt wird, kann der Benutzer in der Konsole Quelldaten und Ergebnisse von Echtzeitanalysen anzeigen.

- **`AmazonKinesisAnalyticsFullAccess`** – Erteilt Berechtigungen für alle Aktionen sowie alle anderen Berechtigungen, mit denen ein Benutzer Anwendungen erstellen und verwalten kann. Beachten Sie jedoch Folgendes:

  - Diese Berechtigungen sind nicht ausreichend, wenn der Benutzer in der Konsole eine neue IAM-Rolle erstellen möchte (mit diesen Berechtigungen kann der Benutzer eine bestehende Rolle auswählen). Wenn Sie möchten, dass der Benutzer in der Konsole eine IAM-Rolle erstellen kann, fügen Sie die von Amazon verwaltete Richtlinie `IAMFullAccess` hinzu.

- Ein Benutzer muss über die Berechtigung für die `iam:PassRole`-Aktion verfügen, um bei der Konfiguration einer Anwendung eine IAM-Rolle festzulegen. Diese von Amazon verwaltete Richtlinie erteilt dem Benutzer die Berechtigung für die `iam:PassRole`-Aktion nur für die IAM-Rollen, die mit dem Präfix `service-role/kinesis-analytics` beginnen.

  Wenn der Benutzer die Anwendung mit einer Rolle konfigurieren möchte, die nicht mit diesem Präfix ausgestattet ist, müssen Sie dem Benutzer zunächst die Benutzerberechtigung zum Ausführen der `iam:PassRole`-Aktion für diese spezifische Rolle gewähren.

Sie können auch Ihre eigenen, benutzerdefinierten IAM-Richtlinien erstellen, um Berechtigungen für -Aktionen und -Ressourcen zu gewähren. Die benutzerdefinierten Richtlinien können Sie dann den -Benutzern oder -Gruppen zuweisen, die diese Berechtigungen benötigen.

Beispiele für vom Kunden verwaltete Richtlinien

In den Beispielen in diesem Abschnitt finden Sie eine Gruppe von Beispielrichtlinien, die Sie Benutzern zuweisen können. Wenn Sie mit dem Erstellen von Richtlinien noch nicht vertraut sind, sollten Sie zunächst einen Benutzer in Ihrem Konto erstellen. Fügen Sie dem Benutzer dann die Richtlinien nacheinander an. Orientieren Sie sich hierbei an den in diesem Abschnitt beschriebenen Schritten. Während Sie dem Benutzer die Richtlinien zuweisen, können Sie dann die Konsole verwenden, um die Auswirkungen der einzelnen Richtlinien zu überprüfen.

Zunächst verfügt der Benutzer über keine Berechtigungen und kann in der Konsole keine Aktionen ausführen. Während Sie dem Benutzer Richtlinien anfügen, können Sie überprüfen, ob der Benutzer die verschiedenen Aktionen in der Konsole ausführen kann.

Wir empfehlen die Verwendung von zwei Browserfenstern. In einem Fenster erstellen Sie den Benutzer und erteilen Berechtigungen. Melden Sie sich im anderen Fall AWS-Managementkonsole mit den Anmeldeinformationen des Benutzers an und überprüfen Sie die Berechtigungen, wenn Sie ihnen gewähren.

Beispiele zur Erstellung einer IAM-Rolle, die Sie als Ausführungsrolle für Ihre Anwendung verwenden können, finden Sie unter Erstellen von IAM-Rollen im IAM-Benutzerhandbuch.

Beispielhafte Vorgehensweise

- Schritt 1: Erstellen eines IAM-Benutzers
- Schritt 2: Gewähren Sie dem Benutzer Berechtigungen für Aktionen, die nicht spezifisch für ihn sind

- Schritt 3: Gewähren der Berechtigung zum Anzeigen einer Liste der Anwendungen und der zugehörigen Details
- Schritt 4: Erteilen einer Berechtigung zum Starten einer bestimmten Anwendung
- Schritt 5: Erteilen der Berechtigung zum Erstellen einer -Anwendung
- Schritt 6: Berechtigen der Anwendung zur Verwendung der Lambda-Vorverarbeitung

Schritt 1: Erstellen eines IAM-Benutzers

Erstellen Sie zunächst einen Benutzer und fügen Sie den Benutzer einer IAM-Gruppe mit Administrator-Berechtigungen hinzu. Anschließend gewähren Sie dem von Ihnen erstellten IAM-Benutzer Administrator-Berechtigungen. Sie können dann AWS mit einer speziellen URL und den Anmeldeinformationen dieses Benutzers darauf zugreifen.

Weitere Anweisungen finden Sie unter Creating Your First IAM User and Administrators Group (Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe) im IAM User Guide (IAM-Benutzerhandbuch).

Schritt 2: Gewähren Sie dem Benutzer Berechtigungen für Aktionen, die nicht spezifisch für ihn sind

Gewähren Sie zunächst einem Benutzer die Berechtigung für alle Aktionen, die nicht spezifisch für ihn sind, die der Benutzer jedoch bei der Arbeit mit Anwendungen benötigt. Dazu gehören Berechtigungen für die Arbeit mit Streams (Amazon Kinesis Data Streams Streams-Aktionen, Amazon Data Firehose-Aktionen) und Berechtigungen für CloudWatch Aktionen. Weisen Sie die dem Benutzer die folgenden Richtlinien zu.

Sie müssen die Richtlinie aktualisieren, indem Sie eine IAM-Rolle angeben, für die Sie die `iam:PassRole`-Berechtigung gewähren möchten, oder mit einem Platzhalterzeichen (*) angeben, dass die Berechtigung allen IAM-Rollen erteilt werden soll. Dies ist keine sichere Methode, Sie haben jedoch möglicherweise während des Testings keine spezifische IAM-Rolle zur Verfügung.

JSON

```
{
    "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"
        }
    ]
```

```
}
```

Schritt 3: Gewähren der Berechtigung zum Anzeigen einer Liste der Anwendungen und der zugehörigen Details

Mit der folgenden Richtlinie werden einen Benutzer die folgenden Berechtigungen erteilt:

- Die Berechtigung für die `kinesisanalytics:ListApplications`-Aktion, mit der der Benutzer eine Liste von Anwendungen anzeigen kann. Hierbei handelt es sich um einen API-Aufruf auf der Service-Ebene, sodass Sie als `Resource`-Wert „*“ angeben müssen.
- Die Berechtigung für die `kinesisanalytics:DescribeApplication`-Aktion, mit der Informationen zu beliebigen Anwendungen abgerufen werden können.

Fügen Sie dem Benutzer diese Richtlinie hinzu.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:ListApplications"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:DescribeApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/*"
        }
    ]
}
```

Überprüfen Sie diese Berechtigungen, indem Sie sich unter Verwendung von Anmeldeinformationen des IAM-Benutzers an der Konsole anmelden.

Schritt 4: Erteilen einer Berechtigung zum Starten einer bestimmten Anwendung

Wenn der Benutzer eine der vorhandenen -Anwendungen starten soll, fügen Sie dem Benutzer die folgende Richtlinie an. Die Richtlinie stellt die Berechtigung zum Ausführen der `kinesisanalytics:StartApplication`-Aktion bereit. Sie müssen die Richtlinie aktualisieren, indem Sie Ihre Konto-ID, AWS Region und Anwendungsnamen angeben.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/application-name"
        }
    ]
}
```

Schritt 5: Erteilen der Berechtigung zum Erstellen einer -Anwendung

Wenn der Benutzer eine -Anwendung erstellen soll, können Sie ihm die folgende Richtlinie anfügen. Sie müssen die Richtlinie aktualisieren und eine AWS Region, Ihre Konto-ID und entweder einen bestimmten Anwendungsnamen, den der Benutzer erstellen soll, oder ein „*“ angeben, damit der Benutzer einen beliebigen Anwendungsnamen angeben kann (und somit mehrere Anwendungen erstellen).

JSON

```
{
    "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:us-
east-1:123456789012:application/application-name"
        }
    ]
}
```

Schritt 6: Berechtigen der Anwendung zur Verwendung der Lambda-Vorverarbeitung

Wenn die Anwendung die Lambda-Vorverarbeitung verwenden können soll, fügen Sie der Rolle die folgende Richtlinie an.

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

API-Berechtigungen: Aktionen, Berechtigungen und Ressourcenreferenz

Wenn Sie die Zugriffskontrolle einrichten und eine Berechtigungsrichtlinie für eine IAM-Identität (identitätsbasierte Richtlinie) verfassen, können Sie die folgende Liste als Referenz verwenden. In der die einzelnen API-Operationen, die entsprechenden Aktionen, für die Sie Berechtigungen zur Ausführung der Aktion erteilen können, und die AWS Ressource, für die Sie die Berechtigungen erteilen können, aufgeführt. Die Aktionen geben Sie im Feld `Action` und den Wert für die Ressource im Feld `Resource` der Richtlinie an.

Sie können in Ihren Richtlinien AWS allgemeine Bedingungsschlüssel verwenden, um Bedingungen auszudrücken. Eine vollständige Liste der AWS-weiten Schlüssel finden Sie unter Verfügbare Schlüssel im IAM-Benutzerhandbuch.

> **Note**
>
> Um eine Aktion anzugeben, verwenden Sie das Präfix `kinesisanalytics` gefolgt vom Namen der API-Operation (z. B. `kinesisanalytics:AddApplicationInput`).

API und erforderliche Berechtigungen für Aktionen

API-Operation:

Erforderliche Berechtigungen (API-Aktion):

Ressourcen:

API und erforderliche Berechtigungen für Aktionen

Amazon-RDS-API und erforderliche Berechtigungen für Aktionen

API-Operation:AddApplicationInput

Aktion: `kinesisanalytics:AddApplicationInput`

Ressourcen:

```
arn:aws:kinesisanalytics: region:accountId:application/application-
name
```

GetApplicationState

Die Konsole verwendet eine als `GetApplicationState` bezeichnete interne Methode zum Erfassen einer Stichprobe von oder zum Zugriff auf Anwendungsdaten. Ihre Serviceanwendung benötigt Berechtigungen für die interne `kinesisanalytics:GetApplicationState`-API, um über die AWS-Managementkonsole eine Stichprobe der Anwendungsdaten erfassen oder auf sie zugreifen zu können.

Überwachung von Amazon Kinesis Data Analytics

Kinesis Data Analytics bietet Überwachungsfunktionen für Ihre Anwendungen. Weitere Informationen finden Sie unter Überwachen.

Konformitätsprüfung für Amazon-Kinesis-Data-Analytics for SQL-Anwendungen

Externe Prüfer bewerten die Sicherheit und Konformität von Amazon Kinesis Data Analytics im Rahmen mehrerer AWS Compliance-Programme. Zu diesen Programmen gehören SOC, PCI, HIPAA und andere.

Eine Liste der AWS Services, die in den Geltungsbereich bestimmter Compliance-Programme fallen, finden Sie unter Amazon Services in Umfang nach Compliance-Programmen. Allgemeine Informationen finden Sie unter AWS -Compliance-Programme.

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen in AWS Artifact.

Ihre Compliance-Verantwortung bei Verwendung von Kinesis Data Analytics hängt von der Vertraulichkeit der Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Wenn Ihre Nutzung von Kinesis Data Analytics Gegenstand der Einhaltung von Richtlinien wie HIPAA oder PCI ist, stellt AWS Ressourcen zur Unterstützung bereit:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Umgebungen

beschrieben, auf denen auf Sicherheit und Compliance ausgerichtete Basisumgebungen eingerichtet werden. AWS

- Whitepaper „Architecting for HIPAA Security and Compliance“ — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-konforme Anwendungen entwickeln können. AWS
- AWS Ressourcen zur Einhaltung von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- AWS Config— Mit diesem AWS Service wird bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub CSPM— Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus und hilft Ihnen AWS , die Einhaltung der Sicherheitsstandards und bewährten Verfahren der Sicherheitsbranche zu überprüfen.

Resilienz in Amazon-Kinesis-Data-Analytics

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur bietet Kinesis Data Analytics mehrere Funktionen, die Sie bei der Unterstützung Ihrer Datenausfallsicherheit und Ihrer Backup-Anforderungen unterstützen.

Notfallwiederherstellung

Kinesis Data Analytics wird in einem Serverless-Modus ausgeführt und bietet dank der automatischen Migration Unterstützung bei Host-Leistungsverschlechterungen, Availability-Zone-Verfügbarkeit und anderen infrastrukturbezogenen Problemen. Wenn dies auftritt, stellt Kinesis Data Analytics sicher, dass die Anwendung ohne Datenverlust verarbeitet wird. Weitere Informationen finden Sie unter Bereitstellungsmodell für die Weiterleitung der Anwendungsausgabe an ein externes Ziel.

Infrastruktursicherheit in Kinesis-Data-Analytics-for-SQL-Anwendungen

Als verwalteter Service ist Amazon Kinesis Data Analytics durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben werden.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Kinesis Data Analytics zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS-Security-Token-Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Bewährte Methoden für die Sicherheit für Kinesis Data Analytics

Amazon-Kinesis-Data-Analytics enthält eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Verwenden Sie IAM-Rollen zum Zugriff auf andere Amazon-Services

Ihre Kinesis Data Analytics Analytics-Anwendung muss über gültige Anmeldeinformationen verfügen, um auf Ressourcen in anderen Services wie Kinesis-Datenströmen, Firehose-Lieferströmen oder Amazon S3 S3-Buckets zugreifen zu können. Sie sollten AWS Anmeldeinformationen nicht direkt in der Anwendung oder in einem Amazon S3 S3-Bucket speichern. Dabei handelt es sich um langfristige Anmeldeinformationen, die nicht automatisch rotiert werden und bedeutende geschäftliche Auswirkungen haben könnten, wenn sie kompromittiert werden.

Stattdessen sollten Sie mithilfe einer IAM-Rolle temporäre Anmeldeinformationen für Ihre Anwendung für den Zugriff auf andere Ressourcen verwalten. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen für den Zugriff auf andere Ressourcen verwenden.

Weitere Informationen finden Sie unter folgenden Themen im IAM-Benutzerhandbuch:

- IAM-Rollen
- Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services

Implementieren einer serverseitigen Verschlüsselung in abhängigen Ressourcen

Daten im Ruhezustand und Daten während der Übertragung werden in Kinesis Data Analytics verschlüsselt, und diese Verschlüsselung kann nicht deaktiviert werden. Sie sollten serverseitige Verschlüsselung in Ihren abhängigen Ressourcen wie Kinesis-Datenströmen, Firehose-Lieferströmen und Amazon S3 S3-Buckets implementieren. Weitere Informationen zum Implementieren der serverseitigen Verschlüsselung bei abhängigen Ressourcen finden Sie unter Datenschutz.

Wird zur Überwachung von API-Aufrufen verwendet CloudTrail

Kinesis Data Analytics ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem Amazon-Service in Kinesis Data Analytics ausgeführt wurden.

Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Kinesis Data Analytics gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen finden Sie unter the section called “Benutzen AWS CloudTrail”.

Überwachung von für SQL-Anwendungen

Die Überwachung ist wichtig, um Zuverlässigkeit, Verfügbarkeit und Leistung von und Ihrer - Anwendung aufrechtzuerhalten. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Mehrpunktfehler leichter debuggen können. Aber bevor Sie mit der Überwachung von beginnen, sollten Sie einen Überwachungsplan mit Antworten auf die folgenden Fragen erstellen:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Im nächsten Schritt legen Sie einen Ausgangswert für normale Performance in Ihrer Umgebung fest, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Sie können bei der Überwachung von historische Überwachungsdaten speichern. Diese gespeicherten Daten bieten eine Basis für den Vergleich mit aktuellen Leistungsdaten, zur Identifikation normaler Leistungsmuster und von Leistungsanomalien sowie zur Entwicklung von Verfahren für den Umgang mit Problemen.

Mit überwachen Sie die Anwendung. Die Anwendung verarbeitet Datenströme (Eingabe oder Ausgabe), die beide Identifikatoren enthalten, mit denen Sie Ihre Suche auf Protokolle eingrenzen können. CloudWatch Weitere Informationen dazu, wie Prozesse Datenströme verarbeiten, finden Sie unter Amazon-Kinesis-Data-Analytics for SQL-Anwendungen: So funktioniert's.

Die wichtigste Metrik ist `millisBehindLatest`, was angibt, mit welcher zeitlichen Differenz zur aktuellen Zeit eine Anwendung von der Streaming-Quelle liest. In einem typischen Fall sollte der Rückstand in Millisekunden bei Null liegen oder gegen Null gehen. Kurze Spitzenwerte sind normal, sie manifestieren sich als erhöhter `millisBehindLatest`-Wert.

Wir empfehlen Ihnen, einen CloudWatch Alarm einzurichten, der ausgelöst wird, wenn die Anwendung beim Lesen der Streaming-Quelle mehr als eine Stunde hinterherhinkt. Bei bestimmten Anwendungsfällen, bei denen eine Verarbeitung nahezu in Echtzeit erforderlich ist, wie bei der

Ausgabe der verarbeiteten Daten an eine Live-Anwendung, können Sie für den Alarm einen niedrigeren Wert, z. B. fünf Minuten, festlegen.

Themen

Überwachungstools

AWS stellt verschiedene Tools bereit, die Sie zur Überwachung verwenden können. Sie können einige dieser Tools so konfigurieren, dass diese die Überwachung für Sie übernehmen, während bei anderen Tools ein manuelles Eingreifen nötig ist. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Automatisierte Überwachungstools

Sie können die folgenden automatisierten Tools zur Überwachung von verwenden und möglicherweise auftretende Probleme melden:

- Amazon CloudWatch Alarms — Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (Amazon SNS) - Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie gesendet wird. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Status befinden. Der Status muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein. Weitere Informationen finden Sie unter Überwachung mit Amazon CloudWatch.
- Amazon CloudWatch Logs — Überwachen, speichern und greifen Sie auf Ihre Protokolldateien aus AWS CloudTrail oder anderen Quellen zu. Weitere Informationen finden Sie unter Überwachung von Protokolldateien im CloudWatch Amazon-Benutzerhandbuch.
- Amazon CloudWatch Events — Ordnen Sie Ereignisse zu und leiten Sie sie an eine oder mehrere Zielfunktionen oder Streams weiter, um Änderungen vorzunehmen, Statusinformationen zu erfassen und Korrekturmaßnahmen zu ergreifen. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch Events im CloudWatch Amazon-Benutzerhandbuch.

- AWS CloudTrail Protokollüberwachung — Teilen Sie Protokolldateien zwischen Konten, überwachen CloudTrail Sie Protokolldateien in Echtzeit, indem Sie sie an CloudWatch Logs senden, schreiben Sie Protokollverarbeitungsanwendungen in Java und überprüfen Sie, ob sich Ihre Protokolldateien nach der Lieferung von nicht geändert haben CloudTrail. Weitere Informationen finden Sie unter Arbeiten mit CloudTrail Protokolldateien im AWS CloudTrail Benutzerhandbuch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Überwachung umfasst die manuelle Überwachung der Elemente, die von den CloudWatch Alarmen nicht abgedeckt werden. Die Dashboards CloudWatch Trusted Advisor,, und andere AWS-Managementkonsole Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer AWS Umgebung.

- Auf der CloudWatch Startseite wird Folgendes angezeigt:
  - Aktuelle Alarme und Status
  - Diagramme mit Alarmen und Ressourcen
  - Servicestatus

  Darüber hinaus können CloudWatch Sie Folgendes verwenden:
  - Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
  - Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
  - Durchsuchen Sie alle Ihre Metriken und gehen Sie sie durch
  - Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden
- AWS Trusted Advisor kann Ihnen bei der Überwachung Ihrer Leistung, Zuverlässigkeit, Sicherheit und Kosteneffektivität helfen. Vier Trusted Advisor -Prüfungen sind für alle Benutzer verfügbar. Mehr als 50 Prüfungen sind für Benutzer mit einem Business- oder Enterprise-Support-Plan verfügbar. Weitere Informationen finden Sie unter AWS Trusted Advisor.

Überwachung mit Amazon CloudWatch

Sie können Anwendungen mit Amazon überwachen CloudWatch. CloudWatch sammelt und verarbeitet Rohdaten in lesbare Messwerte, die nahezu in Echtzeit verfügbar sind. Diese Statistiken werden für einen Zeitraum von zwei Wochen gespeichert. Sie können auf die Verlaufsdaten zugreifen und sich einen besseren Eindruck von der Leistung Ihrer Webanwendung oder Ihres

Service verschaffen. Standardmäßig werden Metrikdaten automatisch an gesendet CloudWatch. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch? im CloudWatch Amazon-Benutzerhandbuch.

Themen

Metriken und Dimensionen

Der `AWS/KinesisAnalytics`-Namespace enthält die folgenden Metriken.

Metrik	Description
`Bytes`	Anzahl der gelesenen (pro Eingabe-Stream) oder geschriebenen (pro Ausgabe-Stream) Byte. Ebenen: Pro Eingabe-Stream und pro Ausgabe-S tream
`KPUs`	Die Anzahl der Kinesis Processing Units, die zum Ausführen der Anwendung für die Stream-Ve rarbeitung verwendet werden. Die durchschnittliche Anzahl der KPUs Nutzer pro Stunde bestimmt die Abrechnung für Ihre Anwendung. Ebenen: Anwendungsebene
`MillisBehindLatest`	Gibt an, mit welcher zeitlichen Differenz zur aktuellen Zeit eine Anwendung von der Streaming-Quelle liest. Ebenen: Anwendungsebene
`Records`	Anzahl der gelesenen (pro Eingabe-Stream) oder geschriebenen (pro Ausgabe-Stream) Datensätze.

Metrik	Description
	Ebenen: Pro Eingabe-Stream und pro Ausgabe-S tream
`Success`	1 für jeden erfolgreichen Zustellungsversuch an das für die Anwendung konfigurierte Ziel, 0 für jeden fehlgeschlagenen Zustellungsversuch. Der durchschnittliche Wert dieser Metrik zeigt an, wie viele erfolgreiche Zustellungen erfolgt sind. Ebenen: pro Ziel
`InputProcessing.Duration`	Die Zeit, die für jeden AWS Lambda Funktionsaufruf benötigt wurde, der von ausgeführt wurde. Ebenen: Pro Eingabe-Stream
`InputProcessing.OkRecords`	Die Anzahl der von einer Lambda-Funktion zurückgegebenen Datensätze mit dem Status `Ok`. Ebenen: Pro Eingabe-Stream
`InputProcessing.OkBytes`	Die Byte-Summe der mit dem Status `Ok` markierte n Datensätze, die von einer Lambda-Funktion zurückgegeben wurden. Ebenen: Pro Eingabe-Stream
`InputProcessing.DroppedReco rds`	Die Anzahl der von einer Lambda-Funktion zurückgegebenen Datensätze mit dem Status `Dropped`. Ebenen: Pro Eingabe-Stream
`InputProcessing.ProcessingF ailedRecords`	Die Anzahl der von einer Lambda-Funktion zurückgegebenen Datensätze mit dem Status `ProcessingFailed` . Ebenen: Pro Eingabe-Stream

Metrik	Description
`InputProcessing.Success`	Die Anzahl der durch Lambda erfolgreich ausgeführten Aufrufe pro Ebenen: Pro Eingabe-Stream
`LambdaDelivery.OkRecords`	Die Anzahl der von einer Lambda-Funktion zurückgegebenen Datensätze mit dem Status `Ok`. Stufen: Pro Lambda-Ziel
`LambdaDelivery.DeliveryFailedRecords`	Die Anzahl der von einer Lambda-Funktion zurückgegebenen Datensätze mit dem Status `DeliveryFailed` . Stufen: Pro Lambda-Ziel
`LambdaDelivery.Duration`	Die Zeit, die für den Aufruf einer Lambda-Funktion benötigt wird pro Stufen: Pro Lambda-Ziel

Stellt Metriken für die folgenden Dimensionen bereit.

Dimension	Description
`Flow`	Pro Eingabe-Stream: Eingabe Pro Ausgabe-Stream: Ausgabe
`Id`	Pro Eingabe-Stream: Eingabe-ID Pro Ausgabe-Stream: Ausgabe-ID

Anzeigen von -Metriken und -Dimensionen

Wenn Ihre Anwendung Datenströme verarbeitet, sendet sie die folgenden Metriken und Dimensionen an CloudWatch. Sie können die folgenden Vorgehensweisen nutzen, um die Metriken für anzuzeigen.

Metriken sind in der Konsole zuerst nach dem Service-Namespace und dann nach verschiedenen Dimensionskombinationen innerhalb eines Namespace gruppiert.

So zeigen Sie Metriken mit der CloudWatch Konsole an

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Metrics (Metriken) aus.
3. Wählen Sie im Bereich CloudWatch Metriken nach Kategorie für eine Metrikkategorie aus.
4. Führen Sie im oberen Bereich einen Bildlauf durch, um die vollständige Liste der Metriken anzuzeigen.

Um Metriken anzuzeigen, verwenden Sie AWS CLI

- Geben Sie als Eingabeaufforderung den folgenden Befehl ein.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Metriken werden auf den folgenden Ebenen gesammelt:

- Anwendung
- Eingabe-Stream
- Ausgabe-Stream

CloudWatch Alarme zur Überwachung erstellen

Sie können einen CloudWatch Amazon-Alarm erstellen, der eine Amazon SNS-Nachricht sendet, wenn sich der Status des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen festgelegten Zeitraum. Der Alarm führt eine oder mehrere Aktionen durch, basierend auf dem Wert der Metrik im Vergleich zu einem bestimmten Schwellenwert in einer Reihe von Zeiträumen. Die Aktion ist eine Benachrichtigung, die an ein Amazon-SNS-Thema oder eine Auto Scaling-Richtlinie gesendet wird.

Alarme rufen nur Aktionen für nachhaltige Statusänderungen auf. Damit ein CloudWatch Alarm eine Aktion auslöst, muss sich der Status geändert haben und für einen bestimmten Zeitraum beibehalten worden sein.

Sie können Alarme mithilfe der CloudWatch API AWS-Managementkonsole CloudWatch AWS CLI, oder einrichten, wie im Folgenden beschrieben.

Um mit der CloudWatch Konsole einen Alarm einzustellen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Alarm erstellen aus. Der Create Alarm Wizard (Assistent zur Alarmerstellung) wird gestartet.
3. Wählen Sie Kinesis Analytics Metrics (Amazon-Kinesis-Analytics-Metriken) aus. Scrollen Sie dann durch die -Metriken, um die Metrik zu finden, für die Sie einen Alarm setzen möchten.

   Um in diesem Dialogfeld nur die -Metriken anzuzeigen, suchen Sie nach der Dateisystem-ID Ihres Dateisystems. Wählen Sie die Metrik, für die ein Alarm erstellt werden soll, und dann Weiter aus.
4. Geben Sie unter Name, Description (Beschreibung) und Whenever (Wenn) die Werte für die Metrik ein.
5. Wenn Sie Ihnen eine E-Mail senden CloudWatch möchten, wenn der Alarmstatus erreicht ist, wählen Sie im Feld Wann immer dieser Alarm: die Option Status ist ALARM. Wählen Sie im Feld Send notification to: (Benachrichtigung senden an:) ein SNS-Thema aus. Wenn Sie Create topic auswählen, können Sie den Namen und die E-Mail Adressen für eine neue E-Mail-Abonnementliste einrichten. Diese Liste wird gespeichert und erscheint für künftige Alarme in der Liste.

   > **Note**
   >
   > Wenn Sie Create topic (Thema erstellen) verwenden, um ein neues Amazon-SNS-Thema einzurichten, müssen die E-Mail Adressen überprüft werden, bevor die Empfänger Benachrichtigungen erhalten. E-Mail Nachrichten werden nur gesendet, wenn der Alarm in einen Alarmzustand wechselt. Wenn es zu dieser Änderung des Alarmzustands kommt, bevor die E-Mail Adressen überprüft wurden, erhalten die Empfänger keine Benachrichtigung.
6. Überprüfen Sie im Alarm Preview den Alarm, den Sie gerade erstellen.
7. Wählen Sie Create Alarm, um den Alarm zu erstellen.

So richten Sie einen Alarm mit der CloudWatch CLI ein

- Rufen Sie die Seite `mon-put-metric-alarm` auf. Weitere Informationen finden Sie in der Amazon CloudWatch CLI Reference.

So richten Sie mithilfe der CloudWatch API einen Alarm ein

- Rufen Sie die Seite `PutMetricAlarm` auf. Weitere Informationen finden Sie in der Amazon CloudWatch API-Referenz.

Mit Amazon CloudWatch Logs arbeiten

Wenn eine -Anwendung falsch konfiguriert ist, kann sie beim Start in den Status „Running“ (Wird ausgeführt) wechseln. Auch ist es möglich, dass die Anwendung Daten aus dem In-Application-Eingabe-Stream aktualisieren, jedoch nicht verarbeiten kann. Indem Sie der Anwendung eine CloudWatch Protokolloption hinzufügen, können Sie nach Problemen mit der Anwendungskonfiguration suchen.

Kann unter folgenden Bedingungen Konfigurationsfehler generieren:

- Der Kinesis-Datenstrom, der für die Eingabe genutzt wird, ist nicht vorhanden.
- Der Amazon Data Firehose-Lieferstream, der für die Eingabe verwendet wurde, ist nicht vorhanden.
- Der als Referenzdatenquelle verwendete Amazon-S3-Bucket ist nicht vorhanden.
- Die in der Referenzdatenquelle in dem S3-Bucket angegebene Datei ist nicht vorhanden.
- Die richtige Ressource ist in der Rolle AWS Identity and Access Management (IAM), die die entsprechenden Berechtigungen verwaltet, nicht definiert.
- Die richtige Berechtigung ist in der IAM-Rolle, die die damit verbundenen Berechtigungen verwaltet, nicht definiert.
- ist nicht berechtigt, die IAM-Rolle anzunehmen, die die damit verbundenen Berechtigungen verwaltet.

Weitere Informationen zu Amazon CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Die PutLogEvents Richtlinienaktion wird hinzugefügt

benötigt Berechtigungen zum Schreiben von Fehlern bei Fehlkonfigurationen. CloudWatch Sie können diese Berechtigungen für die IAM-Rolle, die annehmen soll, wie folgt hinzufügen. Weitere Informationen zur Verwendung der IAM-Rolle finden Sie unter Identitäts- und Zugriffsverwaltung in Kinesis Data Analytics.

Vertrauensrichtlinie

Zum Erteilen von -Berechtigungen, um eine IAM-Rolle anzunehmen, können Sie an die Rolle die folgenden Vertrauensrichtlinien anhängen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Berechtigungsrichtlinie

Um einer Anwendung Berechtigungen zum Schreiben von Protokollereignissen CloudWatch aus einer Ressource zu erteilen, können Sie die folgende IAM-Berechtigungsrichtlinie verwenden.

JSON

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

Hinzufügen einer Überwachung auf Konfigurationsfehler

Verwenden Sie die folgenden API-Aktionen, um einer neuen oder vorhandenen Anwendung eine CloudWatch Protokolloption hinzuzufügen oder eine Protokolloption für eine bestehende Anwendung zu ändern.

> **Note**
>
> Derzeit können Sie einer Anwendung nur mithilfe von API-Aktionen eine CloudWatch Protokolloption hinzufügen. Sie können CloudWatch Protokolloptionen nicht über die Konsole hinzufügen.

Hinzufügen einer CloudWatch Protokolloption beim Erstellen einer Anwendung

Das folgende Codebeispiel zeigt, wie Sie die `CreateApplication` Aktion verwenden, um beim Erstellen einer Anwendung eine CloudWatch Protokolloption hinzuzufügen. Weitere Informationen zu `Create_Application` finden Sie unter 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>"
 }]
}
```

Hinzufügen einer CloudWatch Protokolloption zu einer vorhandenen Anwendung

Das folgende Codebeispiel zeigt, wie Sie die `AddApplicationCloudWatchLoggingOption`-Aktion nutzen können, um einer bestehenden Anwendung eine CloudWatch-Protokollierungsoption hinzuzufügen. Mehr über `AddApplicationCloudWatchLoggingOption` erfahren Sie unter 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>
}
```

Aktualisierung einer vorhandenen CloudWatch Protokolloption

Das folgende Codebeispiel zeigt, wie die `UpdateApplication` Aktion verwendet wird, um eine vorhandene CloudWatch Protokolloption zu ändern. Mehr über `UpdateApplication` erfahren Sie unter 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>
}
```

Löschen einer CloudWatch Protokolloption aus einer Anwendung

Das folgende Codebeispiel zeigt, wie die `DeleteApplicationCloudWatchLoggingOption` Aktion zum Löschen einer vorhandenen CloudWatch Protokolloption verwendet wird. Mehr über `DeleteApplicationCloudWatchLoggingOption` erfahren Sie unter 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>
}
```

Konfigurationsfehler

Die folgenden Abschnitte enthalten Informationen zu Fehlern, die in Amazon CloudWatch Logs aufgrund einer falsch konfigurierten Anwendung auftreten können.

Format der Fehlermeldungen

Die Fehlermeldungen, die aufgrund von Fehlkonfigurationen von Anwendungen generiert werden, haben das folgende Format.

```
{
"applicationARN": "string",
"applicationVersionId": integer,
"messageType": "ERROR",
"message": "string",
"inputId": "string",
"referenceId": "string",
"errorCode": "string"
"messageSchemaVersion": "integer",
}
```

Die Felder in Fehlermeldungen können die folgenden Informationen enthalten:

- `applicationARN`: der Amazon-Ressourcenname (ARN) der Anwendung, die den Fehler erzeugt hat. Beispiel: `arn:aws:kinesisanalytics:us-east-1:112233445566:application/sampleApp`
- `applicationVersionId`: die Version der Anwendung zu dem Zeitpunkt, zu dem der Fehler aufgetreten ist. Weitere Informationen finden Sie unter ApplicationDetail.
- `messageType`: der Typ der Mitteilung. Derzeit kann dieser Typ nur `ERROR` sein.
- `message`: die Fehlerdetails. Beispiel:

```
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`: die der Anwendungseingabe zugeordnete ID. Dieser Wert ist nur vorhanden, wenn diese Eingabe die Ursache des Fehlers ist. Dieser Wert ist nicht vorhanden, wenn `referenceId` verfügbar ist. Weitere Informationen finden Sie unter DescribeApplication.
- `referenceId`: die der Referenzdatenquelle der Anwendung zugeordnete ID. Dieser Wert ist nur vorhanden, wenn diese Quelle die Ursache des Fehlers ist. Dieser Wert ist nicht vorhanden, wenn `inputId` verfügbar ist. Weitere Informationen finden Sie unter DescribeApplication.
- `errorCode`: die Kennung des Fehlers. Diese ID ist entweder `InputError` oder `ReferenceDataError`.
- `messageSchemaVersion`: ein Wert, der die aktuelle Version des Meldungsschemas angibt, zurzeit 1. Sie können diesen Wert überprüfen, um festzustellen, ob das Fehlermeldungsschema aktualisiert wurde.

Fehler

Zu den Fehlern, die möglicherweise in CloudWatch Logs für angezeigt werden, gehören die folgenden.

Ressource ist nicht vorhanden

Wenn ein ARN für einen nicht vorhandenen Kinesis-Eingabe-Stream angegeben wird, der ARN aber syntaktisch korrekt ist, wird eine Fehlermeldung der folgenden Art angezeigt.

```
{
"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"
}
```

Wenn ein falscher Amazon-S3-Dateischlüssel für Referenzdaten verwendet wird, wird ein Fehler wie der folgende generiert:

```
{
 "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"
}
```

Rolle ist nicht vorhanden

Wenn ein ARN wird für eine IAM-Eingaberolle angegeben wird, die nicht vorhanden ist, aber die ARN syntaktisch korrekt ist, wird ein Fehler wie folgt generiert.

```
{
 "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"
```

```
}
```

Rolle hat keine Berechtigung zum Zugriff auf die Ressource

Wenn eine Eingaberolle verwendet wird, die nicht über die erforderlichen Berechtigungen für den Zugriff auf Eingaberessourcen wie etwa einen Kinesis-Quell-Stream verfügt, wird eine Fehlermeldung der folgenden Art angezeigt.

```
{
 "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"
}
```

Protokollieren von AWS CloudTrail-API-Aufrufen mit

ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden. CloudTrail erfasst alle API-Aufrufe als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der -Konsole und Code-Aufrufe der -API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Informationen.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn Aktivitäten in auftreten, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail

Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle - Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle Aktionen werden von der API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe von `CreateApplication` und `UpdateApplication` Aktionen Einträge in den CloudTrail Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root-Benutzer des AWS-Kontos oder mit Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Grundlagen von -Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen

oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die DescribeApplicationAktionen AddApplicationCloudWatchLoggingOptionund demonstriert.


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

Einschränkungen

Termine der Einstellung

Nach reiflicher Überlegung haben wir die Entscheidung getroffen, Amazon Kinesis Data Analytics für SQL-Anwendungen einzustellen. Um Ihnen bei der Planung und Migration weg von Amazon Kinesis Data Analytics for SQL-Anwendungen zu helfen, werden wir das Angebot über einen Zeitraum von 15 Monaten schrittweise einstellen. Es gibt zwei wichtige Daten zu beachten: den 15. Oktober 2025 und den 27. Januar 2026.

1. Am 15. Oktober 2025 werden wir Ihre Bewerbungen einstellen und sie einem `READY` Status zuordnen. Sie können Ihre Anwendungen zu diesem Zeitpunkt neu starten und Ihre Anwendungen weiterhin wie gewohnt verwenden, vorbehaltlich der Servicebeschränkungen.
2. Ab dem 15. Oktober 2025 können Sie keine neuen Amazon Kinesis Data Analytics for SQL-Anwendungen mehr erstellen. Sie können alle vorhandenen Anwendungen wie gewohnt ausführen, vorbehaltlich der Servicebeschränkungen.
3. Wir werden Ihre Bewerbungen ab dem 27. Januar 2026 löschen. Sie können Ihre Amazon Kinesis Data Analytics for SQL-Anwendungen nicht starten oder betreiben. Ab diesem Zeitpunkt ist kein Support mehr für Amazon Kinesis Data Analytics for SQL-Anwendungen verfügbar.

Wir empfehlen Ihnen, Ihre Anwendungen vor dem 15. Oktober 2025 auf Amazon Managed Service für Apache Flink oder Amazon Managed Service für Apache Flink Studio zu migrieren. Ressourcen, die Sie bei Ihrer Migration unterstützen, finden Sie unter. Beispiel für die Migration zu einem Managed Service für Apache Flink Studio Weitere Informationen über Amazon Managed Service für Apache Flink oder Amazon Managed Service für Apache Flink Studio finden Sie im Amazon Managed Service for Apache Flink Developer Guide.

Einschränkungen

Beachten Sie beim Arbeiten mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen die folgenden Einschränkungen:

- Kinesis Data Analytics for SQL ist in den folgenden AWS Regionen verfügbar: USA Ost (Ohio), USA Ost (Nord-Virginia), USA West (Oregon), Kanada (Zentral), Europa (Paris), Europa (Irland), Europa (Frankfurt), Europa (London), Asien-Pazifik (Hongkong), Asien-Pazifik (Mumbai), Asien-

Pazifik (Sydney), Asien-Pazifik (Singapur), Asien-Pazifik (Seoul), Asien-Pazifik (Tokio), Süd Amerika (Sao Paulo), AWS GovCloud (US-Ost), AWS GovCloud (US-West). Wir haben nicht vor, Kinesis Data Analytics for SQL in weiteren AWS Regionen einzuführen.

- Nach dem 28. Juni 2023 können Sie keine neuen Kinesis Data Analytics for SQL-Anwendungen mit der AWS Management-Konsole erstellen, wenn Sie Kinesis Data Analytics for SQL noch nicht verwenden. Informationen zu den Abkündigungsdaten von Kinesis Data Analytics for SQL finden Sie unter. Termine der Einstellung Wenn Sie vor dem 28. Juni 2023 eine Kinesis Data Analytics for SQL-Anwendung erstellt haben, ändert sich nichts daran, wie Sie heute Anwendungen in einer AWS Region erstellen und ausführen, in der Sie Kinesis Data Analytics for SQL bereits verwenden. In einer Region, in der Sie Kinesis Data Analytics for SQL nicht verwenden, können Sie jedoch keine neuen Anwendungen mehr mit der AWS Konsole erstellen.
- Nach dem 12. September 2023 können Sie keine neuen Anwendungen mit Kinesis Data Firehose als Quelle erstellen, wenn Sie Kinesis Data Analytics for SQL nicht bereits verwenden. Informationen zu den Abkündigungsdaten von Kinesis Data Analytics for SQL finden Sie unter. Termine der Einstellung Bestandskunden, die Kinesis-Data-Analytics-for-SQL-Anwendungen mit `KinesisFirehoseInput` verwenden, können weiterhin Kinesis Data Analytics einsetzen, um Anwendungen mit `KinesisFirehoseInput` innerhalb eines bestehenden Kontos hinzuzufügen. Wenn Sie bereits Kunde sind und ein neues Konto bei Kinesis-Data-Analytics-for-SQL-Anwendungen mit `KinesisFirehoseInput` erstellen möchten, können Sie einen Support-Fall eröffnen. Weitere Informationen erhalten Sie im AWS Support Center.
- Die Größe einer Zeile in einem In-Application-Stream ist auf 512 KB begrenzt. Kinesis Data Analytics verwendet bis zu 1 KB zum Speichern von Metadaten. Diese Metadaten zählen zum Zeilenlimit. Wenn die Datensatzgröße in der Streaming-Quelle mehr als 50 KB beträgt, können Sie den Datensatz in Ihrem Anwendungsstream in mehrere Zeilen aufteilen, indem Sie in der Eingabekonfiguration ein entsprechendes Schema angeben, indem Sie ein Zeilentrennzeichen angeben.
- Der SQL-Code in einer Anwendung ist auf 100 KB begrenzt.
- Das längste Fenster, das wir für eine Abfrage mit Fenstern empfehlen, ist eine Stunde. In-Application-Streams werden im flüchtigen Speicher gespeichert, und unerwartete Anwendungsunterbrechungen führen dazu, dass die Anwendung den Stream aus den Quelldaten im flüchtigen Speicher neu erstellt.
- Der größte Durchsatz, den wir für einen einzelnen In-Application-Stream empfehlen, liegt je nach Komplexität der Abfrage der Anwendung zwischen 2 und 20 MB/s.
- Sie können in Ihrem Konto bis zu 50 Kinesis Data Analytics Analytics-Anwendungen pro AWS Region erstellen. Sie können einen Fall erstellen, um weitere Anwendungen über das Formular zur

Erhöhung Ihres Service-Limits anzufordern. Weitere Informationen erhalten Sie im AWS Support - Sicherheitszentrum.

- Der maximale Streaming-Durchsatz, den eine einzelne Kinesis Data Analytics for SQL-Anwendung verarbeiten kann, beträgt ungefähr 100 MB/sec. This assumes that you have increased the number of in-application streams to the maximum value of 64, and you have increased your KPU limit beyond 8 (see the following limit for details). If your application needs to process more than 100 MB/sec der Eingaben. Führen Sie einen der folgenden Schritte aus:
  - Verwenden Sie mehrere Kinesis-Data-Analytics-for-SQL-Anwendungen für die Verarbeitung der Eingabe
  - Verwenden Sie Managed Service für Apache Flink für Java-Anwendungen, wenn Sie weiterhin mit einem einzelnen Stream und einer einzelnen Anwendung arbeiten möchten.

  > **Note**
  >
  > Wir empfehlen, die `InputProcessing.0kBytes` Metrik Ihrer Anwendung regelmäßig zu überprüfen, damit Sie im Voraus planen können, mehrere SQL-Anwendungen zu verwenden oder zu Amazon Managed Service for Apache Flink for Java Applications zu migrieren, wenn der prognostizierte Eingabedurchsatz Ihrer Anwendung 100 MB/s überschreitet. Wir empfehlen außerdem, einen CloudWatch Alarm zu aktivieren, `InputProcessing.0kBytes` sodass Sie benachrichtigt werden, wenn sich Ihre Anwendung dem Grenzwert für den Eingabedurchsatz nähert. Dies kann nützlich sein, da Sie Ihre Anwendungsabfrage aktualisieren können, um einen höheren Durchsatz zu erzielen und so Gegendruck und Verzögerungen bei der Analyse zu vermeiden. Weitere Informationen finden Sie unter Fehlerbehebung. Eine Alarmierung kann auch nützlich sein, wenn Sie über einen Mechanismus verfügen, mit dem der Durchsatz im Upstream-Bereich reduziert werden kann.

- Die Anzahl der Kinesis Processing Units (KPU) ist auf acht begrenzt. Anweisungen zum Anfordern einer Erhöhung dieses Limits finden Sie unter So fordern Sie eine Erhöhung Ihres Limits an unter Amazon-Service-Limits.

  Mit Kinesis Data Analytics zahlen Sie nur für das, was Sie tatsächlich nutzen. Ihnen wird ein Stundensatz berechnet, der auf der durchschnittlichen Anzahl von Daten basiert, die für KPUs die Ausführung Ihrer Streamverarbeitungsanwendung verwendet werden. Eine einzelne KPU bietet Ihnen 1 vCPU und 4 GB Arbeitsspeicher.

- Jede Anwendung kann nur über eine Streaming-Quelle und bis zu eine Referenzdatenquelle verfügen.
- Sie können bis zu drei Ziele für Ihre Kinesis Data Analytics-Anwendung konfigurieren. Wir empfehlen die Verwendung eines dieser Ziele, um Daten des In-Application-Fehler-Streams dauerhaft zu speichern.
- Das Amazon-S3-Objekt, in dem die Referenzdaten gespeichert werden, kann bis zu 1 GB groß sein.
- Wenn Sie die Referenzdaten, die im S3-Bucket gespeichert sind, ändern, nachdem Sie Referenzdaten in eine anwendungsinterne Tabelle hochgeladen haben, müssen Sie den UpdateApplication Vorgang (mithilfe der API oder AWS CLI) verwenden, um die Daten in der anwendungsinternen Tabelle zu aktualisieren. Derzeit unterstützt das das Aktualisieren von Referenzdaten in Ihrer Anwendung AWS-Managementkonsole nicht.
- Kinesis Data Analytics unterstützt derzeit keine von der Amazon-Kinesis-Producer-Library (KPL) generierten Daten.
- Sie können bis zu 50 Tags pro Anwendung zuweisen.

Bewährte Methoden

In diesem Abschnitt werden bewährte Methoden bei der Arbeit mit Amazon-Kinesis-Data-Analytics-Anwendungen beschrieben.

Themen

Verwalten von Anwendungen

Halten Sie bei der Verwaltung von Amazon-Kinesis-Data-Analytics-Anwendungen die folgenden bewährten Methoden ein:

- CloudWatch Amazon-Alarme einrichten — Sie können die von Kinesis Data Analytics bereitgestellten CloudWatch Metriken verwenden, um Folgendes zu überwachen:
  - Eingabebytes und Eingabedatensätze (Anzahl der Bytes und Datensätze in der Anwendung)
  - Ausgabebytes und Ausgabedatensätze
  - `MillisBehindLatest` (Gibt an, mit welcher zeitlichen Differenz zur aktuellen Zeit eine Anwendung von der Streaming-Quelle liest.)

  Wir empfehlen, dass Sie mindestens zwei CloudWatch Alarme für die folgenden Metriken für Ihre in Produktion befindlichen Anwendungen einrichten:
  - `MillisBehindLatest` – Bei den meisten Fällen empfehlen wir, dass Sie den Alarm so einrichten, dass er ausgelöst wird, wenn die Anwendung mit den neuesten Daten eine Stunde lang durchschnittlich für eine Minute hinterher liegt. Für Anwendungen mit geringeren end-to-end Verarbeitungsanforderungen können Sie dies auf eine geringere Toleranz einstellen. Mit diesem Alarm können Sie sicherstellen, dass Ihre Anwendung die neuesten Daten liest.

- Beschränken Sie die Anzahl der Produktionsanwendungen, die denselben Kinesis-Datenstrom lesen, auf zwei Anwendungen, um zu verhindern, dass die `ReadProvisionedThroughputException`-Ausnahmebedingung ausgelöst wird.

  > **Note**
  >
  > In diesem Fall bezeichnet Anwendung eine beliebigen Anwendung, die von der Streamingquelle lesen kann. Nur eine Kinesis Data Analytics Analytics-Anwendung kann aus einem Firehose-Lieferstream lesen. Viele Anwendungen können jedoch aus einem Kinesis-Datenstream lesen, z. B. eine Kinesis Data Analytics-Anwendung oder. AWS Lambda Der empfohlene Anwendungsgrenzwert gilt für alle Anwendungen, die Sie zum Lesen von Daten aus einer Streaming-Quelle konfigurieren.

  Amazon-Kinesis-Data-Analytics liest für jede Anwendung eine Streaming-Quelle ca. einmal pro Sekunde aus. Anwendungen, die mit Ihrer Leseleistung zurückfallen, können Daten möglicherweise mit einer schnelleren Rate einlesen, um aufzuholen. Um Anwendungen einen genügend großen Durchsatz bereitzustellen, damit sie aufholen können, beschränken Sie die Anzahl der Anwendungen, die von derselben Datenquelle lesen.

- Beschränken Sie die Anzahl der Produktionsanwendungen, die aus demselben Firehose-Lieferstream lesen, auf eine Anwendung.

  Ein Firehose-Lieferstream kann in Ziele wie Amazon S3 und Amazon Redshift schreiben. Er kann auch als Streaming-Quelle für Ihre Kinesis Data Analytics-Anwendung dienen. Daher empfehlen wir, nicht mehr als eine Kinesis Data Analytics Analytics-Anwendung pro Firehose-Lieferstream zu konfigurieren. So stellen Sie sicher, dass der Bereitstellungs-Stream auch für andere Ziele bereitstellen kann.

Skalierungsanwendungen

Richten Sie Ihre Anwendung für Ihre zukünftigen Skalierungsanforderungen ein, indem Sie den Standardwert (1) für die Anzahl von In-Application-Eingabe-Streams erhöhen. Wir empfehlen die folgende Sprachauswahl basierend auf dem Durchsatz Ihrer Anwendung:

- Verwenden Sie mehrere Streams und Kinesis-Data-Analytics-for-SQL-Anwendungen, wenn die Skalierungsanforderungen Ihrer Anwendung 100 MB/Sekunde überschreiten.
- Verwenden Sie Managed Service für Apache Flink-Anwendungen, wenn Sie einen einzelnen Stream und eine einzelne Anwendung verwenden möchten.

> **Note**
>
> Wir empfehlen, die `InputProcessing.0kBytes` Metrik Ihrer Anwendung regelmäßig zu überprüfen, damit Sie im Voraus planen können, mehrere SQL-Anwendungen zu verwenden oder zu Managed- zu migrieren. flink/latest/java/ if your application's projected input throughput exceeds 100 MB/sec

Überwachen von Anwendungen

Wir empfehlen, einen CloudWatch Alarm zu aktivieren, `InputProcessing.0kBytes` sodass Sie benachrichtigt werden, wenn sich Ihre Anwendung dem Grenzwert für den Eingabedurchsatz nähert. Dies kann nützlich sein, da Sie Ihre Anwendungsabfrage aktualisieren können, um einen höheren Durchsatz zu erzielen und so Gegendruck und Verzögerungen bei der Analyse zu vermeiden. Weitere Informationen finden Sie unter Fehlerbehebung. Dies kann auch nützlich sein, wenn Sie über einen Mechanismus zur Reduzierung des Durchsatzes im Upstream-Bereich verfügen.

- Der größte Durchsatz, den wir für einen einzelnen In-Application-Stream empfehlen, liegt je nach Komplexität der Abfrage der Anwendung zwischen 2 und 20 MB/s.
- Der maximale Streaming-Durchsatz, den eine einzelne Kinesis Data Analytics for SQL-Anwendung verarbeiten kann, beträgt ungefähr 100 MB/s. Dies setzt voraus, dass Sie die Anzahl der In-App-Streams auf den Höchstwert von 64 erhöht haben und dass Sie Ihr KPU-Limit auf über 8 erhöht haben. Weitere Informationen finden Sie unter Limits.

> **Note**
>
> Wir empfehlen, die `InputProcessing.0kBytes` Metrik Ihrer Anwendung regelmäßig zu überprüfen, damit Sie im Voraus planen können, mehrere SQL-Anwendungen zu verwenden oder zu Managed- flink/latest/java/ if your application's projected input throughput exceeds 100 MB/sec zu migrieren.

Definieren des Eingabeschemas

Bei der Konfiguration von Anwendungseingabe in der Konsole müssen Sie zunächst eine Streaming-Quelle angeben. Die Konsole verwendet die Erkennungs-API (siehe DiscoverInputSchema), um ein Schema abzuleiten, indem Datensätze von der Streaming-Quelle gesampelt werden. Das Schema definiert u. a. Namen und Datentypen der Spalten in dem resultierenden In-Application-Stream. Die Konsole zeigt das Schema an. Wir empfehlen, mit diesem abgeleiteten Schema wie folgt vorzugehen:

- Testen Sie das abgeleitete Schema gründlich. Bei dem Erkennungsprozess werden nur Proben der Datensätze aus der Streaming-Quelle verwendet, um daraus ein Schema abzuleiten. Wenn Ihre Streaming-Quelle über zahlreiche Datensatztypen verfügt, wurden möglicherweise nicht alle Datensatztypen von der Erkennungs-API erfasst. Dadurch kann ein Schema entstehen, dass die Daten in der Streaming-Quelle nicht genau widerspiegelt.

  Wenn Ihre Anwendung gestartet wird, können diese nicht berücksichtigten Datensatztypen zu Fehlern bei der Strukturanalyse führen. Amazon-Kinesis-Data-Analytics sendet diese Datensätze an den In-Application-Fehler-Stream. Um Fehler bei der Strukturanalyse zu reduzieren, empfehlen wir, das abgeleitete Schema interaktiv in der Konsole zu testen und den In-Application-Stream auf ausgelassene Datensätze hin zu überwachen.

- Die Kinesis Data Analytics API unterstützt für Spalten in der Eingabekonfiguration nicht die Angabe einer `NOT NULL`-Einschränkung. Wenn Sie in Ihrem In-Application-Stream für bestimmte Spalten `NOT NULL`-Einschränkungen festlegen möchten, erstellen Sie diese In-Application-Streams in Ihrem Anwendungscode. Anschließend können Sie Daten von einem In-Application-Stream in einen anderen kopieren und dort die Beschränkung durchsetzen.

  Bei jedem Versuch, Zeilen mit `NULL`-Werten einzufügen, wenn ein Wert erforderlich ist, wird ein Fehler ausgegeben. Kinesis Data Analytics sendet diese Fehler an den In-App-Fehler-Stream.

- Verallgemeinern Sie Datentypen, die bei dem Erkennungsprozess abgeleitet wurden. Der Erkennungsvorgang empfiehlt Spalten und Datentypen basierend auf einer Zufallsauswahl von Datensätzen aus der Streaming-Quelle. Wir empfehlen, dass Sie die Datensätze sorgfältig überprüfen und sie für alle möglichen Fälle von Datensätzen in der Eingabe generalisieren. Auf diese Weise wird sichergestellt, dass bei der Strukturanalyse während der Ausführung der Anwendung weniger Fehler auftreten. Wenn beispielsweise ein Schema mit einem Spaltentyp `SMALLINT` abgeleitet wurde, sollten Sie ihn ggf. in einen `INTEGER` ändern.

- Verwenden Sie SQL-Funktionen in Ihrem Anwendungscode, um unstrukturierte Daten oder Spalten zu verarbeiten. Möglicherweise haben Sie unstrukturierte Daten oder Spalten in Ihrer Eingabe, z. B. Protokolldaten. Beispiele finden Sie unter Beispiel: Werte transformieren DateTime. Ein Ansatz zur Verarbeitung dieser Art von Daten ist, das Schema mit nur einem Spaltentyp `VARCHAR(N)` zu definieren, wobei `N` die längstmögliche Zeile ist, die Sie in Ihrem Stream erwarten. Anschließend können Sie dann die eingehenden Datensätze mit Ihrer Anwendung einlesen und mit den `String`- und `Date Time`-Funktionen analysieren und so ein Schema für die Rohdaten gewinnen.

- Stellen Sie sicher, dass Sie bei der Verarbeitung Streaming-Quelldaten, die tiefer als zwei Ebenen verschachtelt sind, vollständig abarbeiten. JSON-Quelldaten können verschachtelt sein. Die Erkennungs-API leitet ein Schema ab, das die Eingabe auf eine Verschachtelungsebene reduziert. Die Erkennungs-API versucht auch bei zwei Schachtelungsebenen, diese auf eine Ebene zu reduzieren. Bei mehr als zwei Schachtelungsebenen kann die Strukturtiefe nur bedingt reduziert werden. Um verschachtelte Strukturen vollständig zu verarbeiten, müssen Sie das abgeleitete Schema manuell an Ihre Anforderungen anpassen. Verwenden Sie dazu eine der beiden folgenden Strategien:

  - Verwenden Sie den JSON-Zeilenpfad, um genau die benötigten Schlüssel-Wert-Paare für Ihre Anwendung auszulesen. Der JSON-Zeilenpfad gibt einen Zeiger auf das jeweilige Schlüssel-Wert-Paar in Ihrer Anwendung zurück. Diese Strategie funktioniert bei beliebiger Verschachtelungstiefe.
  - Verwenden Sie den JSON-Zeilenpfad, um komplexe JSON-Objekte und herausziehen. Verwenden Sie anschließend in Ihrem Anwendungscode die Funktionen zur String-Manipulation, um bestimmte Daten, die Sie benötigen, zu exzerpieren.

Herstellen einer Verbindung mit Ausgaben

Wir empfehlen, dass jede Anwendung über mindestens zwei Ausgaben verfügen sollte:

- Verwenden Sie das erste Ziel, um die Ergebnisse Ihrer SQL-Abfragen auszugeben.
- Verwenden Sie das zweite Ziel, um den gesamten Fehlerstream einzufügen und ihn über einen Firehose-Lieferstream an einen S3-Bucket zu senden.

Anwendungscode erstellen

Wir empfehlen Folgendes:

- Geben Sie aus den folgenden Gründen in der SQL-Anweisung keine Zeitfenster an, die größer als eine Stunde sind:
  - Anwendungen müssen gelegentlich neu gestartet werden, entweder aufgrund einer Aktualisierung oder aus internen Kinesis Data Analytics-spezifischen Gründen. Bei einem Neustart müssen alle in dem Zeitfenster enthaltenen Daten erneut aus der Streaming-Datenquelle eingelesen werden. Dies dauert eine Weile, bevor Kinesis Data Analytics Ausgaben für dieses Fenster produzieren kann.
  - Kinesis Data Analytics muss für diese Dauer alles, was im Zusammenhang mit der Anwendung steht, inklusive der relevanten Daten, bereithalten. Dies verbraucht eine erhebliche Menge an Kinesis Data Analytics-Verarbeitungseinheiten.
- Halten Sie während der Entwicklung die Fenstergröße in Ihren SQL-Anweisungen klein, um schneller Ergebnisse zu erhalten. Wenn Sie die Anwendung in der Produktionsumgebung bereitstellen, können Sie die Fenstergröße geeignet anpassen.
- Verwenden Sie anstelle einer einzigen komplexen SQL-Anweisung mehrere Anweisungen, die in jedem Schritt Zwischenergebnisse in In-Application-Streams speichern. Dies vereinfacht auch das Debugging.
- Bei der Verwendung von rollierenden Zeitfenstern empfehlen wir, dass Sie zwei Fenster verwenden: eines für die Verarbeitungszeit und eines für die logische Zeit (Zeitpunkt der Erfassung oder des Ereignisses). Weitere Informationen finden Sie unter Zeitstempel und die ROWTIME-Spalte.

Testen von Anwendungen

Wenn Sie das Schema oder den Anwendungscode für Ihre Kinesis Data Analytics-Anwendung ändern, empfehlen wir Ihnen, die Änderungen mit einer Testanwendung zu überprüfen, bevor Sie diese für die Produktion bereitstellen.

Einrichten einer Testanwendung

Sie können eine Testanwendung entweder über die Konsole oder mithilfe einer CloudFormation -Vorlage einrichten. Mithilfe einer CloudFormation Vorlage können Sie sicherstellen, dass die

Codeänderungen, die Sie an der Testanwendung und Ihrer Live-Anwendung vornehmen, konsistent sind.

Beim Einrichten einer Testanwendung können Sie die Anwendung entweder mit Ihren Live-Daten verbinden oder einen Stream zu Testzwecken mit Mock-Daten befüllen. Wir empfehlen für das Befüllen eines Streams mit Mock-Daten zwei Methoden:

- Verwenden Sie den Kinesis Data Generator (KDG). Der KDG verwendet eine Datenvorlage, mit der zufällige Daten an einen Kinesis-Stream gesendet werden. Er ist einfach zu verwenden, eignet sich jedoch nicht zum Testen von komplexen Beziehungen zwischen Datenelementen wie beispielsweise in Anwendungen, die Hotspots oder Anomalien erkennen.
- Verwenden Sie eine benutzerdefinierte Python-Anwendung, um komplexere Daten an einen Kinesis-Datenstrom zu senden. Mithilfe einer Python-Anwendung können Sie komplexe Beziehungen zwischen Datenelementen wie Hotspots oder Anomalien erzeugen. Eine Python-Beispielanwendung, die Daten in Clusterform an einen Daten-Hotspot sendet, finden Sie unter Beispiel: Erkennen von Hotspots in einem Stream (HOTSPOTS-Funktion).

Wenn Sie Ihre Testanwendung ausführen, zeigen Sie Ihre Ergebnisse mit einem Ziel an (z. B. einem Firehose-Lieferstream zu einer Amazon Redshift Redshift-Datenbank), anstatt Ihren In-Application-Stream auf der Konsole anzusehen. Die in der Konsole angezeigten Daten sind ein Auszug aus dem Stream und enthalten nicht alle Datensätze.

Testen von Schemaänderungen

Wenn Sie Änderungen am Eingabe-Stream-Schema einer Anwendung vornehmen, überprüfen Sie mit der Testanwendung Folgendes:

- Die Daten aus Ihrem Stream werden in den richtigen Datentyp umgewandelt. Stellen Sie beispielsweise sicher, dass Datum-/Uhrzeit-Angaben nicht als Zeichenfolge in die Anwendung aufgenommen werden.
- Die Daten werden in dem gewünschten Datentyp analysiert und umgewandelt. Wenn bei der Analyse oder Umwandlung Fehler auftreten, können Sie diese in der Konsole ausgeben oder dem Fehler-Stream ein Ziel zuweisen und die Fehler im Zielspeicher betrachten.
- Die Datenfelder für Zeichendaten sind ausreichend lang und die Zeichendaten werden in der Anwendung nicht abgeschnitten. Sie können die Datensätze in Ihrem Zielspeicher überprüfen, um sicherzustellen, dass die Anwendungsdaten nicht abgeschnitten werden.

Testen von Codeänderungen

Um Änderungen an Ihrem SQL-Code zu testen, müssen Sie die Domain Ihrer Anwendung kennen. Sie müssen feststellen können, welche Ausgabe geprüft werden und wie eine korrekte Ausgabe aussehen muss. Informationen zu potenziellen Problembereichen beim Prüfen von Änderungen am SQL-Anwendungscode finden Sie unter Fehlerbehebung bei Amazon-Kinesis-Data-Analytics for SQL-Anwendungen.

Fehlerbehebung bei Amazon-Kinesis-Data-Analytics for SQL-Anwendungen

Die folgenden Informationen können zum Beheben von möglichen Problemen mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen nützlich sein.

Themen

Gestoppte Anwendungen

- Was ist eine gestoppte Anwendung von Kinesis Data Analytics for SQL?

  Eine gestoppte Anwendung ist eine Anwendung, bei der wir beobachtet haben, dass sie mindestens drei Monate lang keine Datensätze verarbeitet hat. Das bedeutet, dass Kunden für Kinesis-Data-Analytics-for-SQL-Ressourcen zahlen, die sie nicht nutzen.

- Wann wird AWS mit dem Stoppen inaktiver Anwendungen begonnen?

  AWS wird am 14. November 2023 mit dem Stoppen inaktiver Anwendungen beginnen und bis zum 21. November 2023 abgeschlossen sein. Wir werden inaktive Anwendungen während der Bürozeiten in der Zeitzone der jeweiligen Region stoppen.

- Können gestoppte Kinesis-Data-Analytics-for-SQL-Anwendungen neu gestartet werden?

  Ja. Wenn Sie Ihre Anwendung neu starten müssen, können Sie dies wie gewohnt tun. Es ist nicht erforderlich, ein Support-Ticket zu erstellen.

- Werden meine Abfrageergebnisse auch gelöscht, wenn eine Anwendung im Leerlauf AWS beendet wird?

  Nein. Erstens verarbeitet Ihre Anwendung keine Abfragen, da sie inaktiv ist. Zweitens werden Ihre Abfrageergebnisse nicht in Kinesis Data Analytics for SQL gespeichert. Sie konfigurieren Ihre Kinesis-Data-Analytics-for-SQL-Anwendung mit einem Weiterleitungsziel, an das die Ergebnisse der Berechnungen gesendet werden (z. B. in Amazon-S3 oder in einem anderen Datenstrom). Somit behalten Sie die vollen Eigentumsrechte an Ihren Daten und sie bleiben gemäß den Bedingungen dieses Speicherservices abrufbar.

- Was mache ich, wenn ich nicht möchte, dass meine Anwendung gestoppt wird?

  Sie können dem Serviceteam (kda-sql-questions@amazon .com) eine E-Mail mit der Bitte senden, dass Anwendungen nicht vor dem 10. November 2023 eingestellt werden. Die E-Mail sollte Ihre Konto-ID und Ihren Anwendungs-ARN enthalten.

SQL-Code kann nicht ausgeführt werden

Wenn Sie herausfinden müssen, wie Sie die einwandfreie Funktionsweise einer bestimmten SQL-Anweisung sicherstellen können, stehen Ihnen dafür verschiedene Ressourcen bei der Verwendung von Kinesis Data Analytics zur Verfügung:

- Weitere Informationen zu SQL-Anweisungen finden Sie unterKinesis Data Analytics for SQL. Dieser Abschnitt enthält eine Reihe von SQL-Beispielen, die Sie verwenden können.
- Die Amazon-Kinesis-Data-Analytics-SQL-Referenz stellt detaillierte Anleitungen zum Erstellen von SQL-Streaming-Anweisungen bereit.
- Falls Sie immer noch Probleme haben, empfehlen wir, dass Sie eine entsprechende Frage in den Kinesis Data Analytics-Foren stellen.

Mein Schema wird nicht erkannt oder gefunden

In einigen Fällen kann Kinesis Data Analytics kein Schema erkennen oder finden. In vielen dieser Fälle lässt sich Kinesis Data Analytics trotzdem verwenden.

Nehmen wir an, dass Sie gemäß UTF-8 kodierte Daten haben, die kein Trennzeichen verwenden, oder Daten mit einem anderen Format als das CSV-Dateiformat (durch Kommas getrennte Werte) oder dass die Erkennungs-API Ihr Schema nicht gefunden hat. In diesen Fällen können Sie ein

Schema manuell definieren oder die Funktionen zur Zeichenkettenmanipulation verwenden, um Ihre Daten zu strukturieren.

Um das Schema für Ihren Stream zu finden, nimmt Kinesis Data Analytics zufällige Stichproben der aktuellen Daten in Ihrem Stream. Wenn Sie nicht konsistent Daten an Ihren Stream senden, kann Kinesis Data Analytics möglicherweise keine Stichprobe abrufen, um ein Schema zu erkennen. Weitere Informationen finden Sie unter Verwenden der Funktion der Schemaerkennung für Streaming-Daten.

Referenzdaten sind nicht mehr auf dem neuesten Stand

Referenzdaten werden aus dem Amazon-Simple-Storage-Service (Amazon-S3)-Objekt in die Anwendung geladen, wenn die Anwendung gestartet oder aktualisiert wird, oder während Anwendungsunterbrechungen aufgrund von Service-Problemen.

Referenzdaten werden nicht in die Anwendung geladen, wenn das zugrunde liegende Amazon-S3-Objekt aktualisiert wird.

Wenn die Referenzdaten in der Anwendung nicht auf dem neuesten Stand sind, können Sie die Daten neu laden, indem Sie die folgenden Schritte ausführen:

1. Wählen Sie in der Kinesis Data Analytics-Konsole den Namen der Anwendung in der Liste aus und klicken Sie dann auf Anwendungsdetails.
2. Klicken Sie auf Go to SQL-Editor (Gehe zu SQL-Editor), um die Seite Real-time analytics (Echtzeitanalyse) für die Anwendung zu öffnen.
3. Wählen Sie in der Ansicht Source Data (Quelldaten) den Namen Ihrer Referenzdatentabelle aus.
4. Wählen Sie Actions (Aktionen), Synchronize reference data table (Referenzdatentabelle synchronisieren).

Anwendung wird nicht ans Ziel geschrieben

Wenn keine Daten ans Ziel geschrieben werden, überprüfen Sie Folgendes:

- Stellen Sie sicher, dass die Rolle der Anwendung über ausreichende Berechtigungen für den Zugriff auf das Ziel verfügt. Weitere Informationen finden Sie unter Berechtigungsrichtlinie für das Schreiben an einen Kinesis-Stream oder Berechtigungsrichtlinie für das Schreiben zu einem Firehose-Bereitstellungs-Stream.

- Stellen Sie sicher, dass das Anwendungsziel richtig konfiguriert ist und dass die Anwendung den richtigen Namen für den Ausgabe-Stream verwendet.
- Überprüfen Sie die CloudWatch Amazon-Metriken für Ihren Output-Stream, um zu sehen, ob Daten geschrieben werden. Informationen zur Verwendung von CloudWatch Metriken finden Sie unterÜberwachung mit Amazon CloudWatch.
- Fügen Sie einen CloudWatch Log-Stream hinzu mitthe section called "AddApplicationCloudWatchLoggingOption". Die Anwendung schreibt Konfigurationsfehler in den Protokoll-Stream.

Wenn Rolle und Ziel korrekt konfiguriert sind, starten Sie die Anwendung neu. Geben Sie dabei `LAST_STOPPED_POINT` für die InputStartingPositionConfiguration ein.

Wichtige zu überwachende Parameter zum Anwendungsstatus

Um sicherzustellen, dass Ihre Anwendung einwandfrei ausgeführt wird, empfehlen wir, dass Sie bestimmte wichtige Parameter überwachen.

Der wichtigste zu überwachende Parameter ist die CloudWatch Amazon-Metrik`MillisBehindLatest`. Diese Metrik stellt dar, mit welcher zeitlichen Differenz zur aktuellen Zeit Sie aus dem Stream lesen. Mit dieser Metrik können Sie ermitteln, ob Sie Datensätze aus dem Quell-Stream schnell genug verarbeiten.

In der Regel sollten Sie einen CloudWatch Alarm einrichten, der ausgelöst wird, wenn Sie länger als eine Stunde in Rückstand geraten. Die Zeitdauer hängt allerdings von Ihrem Anwendungsfall ab. Sie können sie nach Bedarf anpassen.

Weitere Informationen finden Sie unter Bewährte Methoden.

Fehler „Ungültiger Code" beim Ausführen einer Anwendung

Zu häufigen Ursachen dafür, dass sich der SQL-Code für Ihre Amazon-Kinesis-Data-Analytics-Anwendung nicht speichern und ausführen lässt, zählen unter anderem diese:

- Der Stream wurde in Ihrem SQL-Code umdefiniert – Nachdem Sie einen Stream und die zugeordnete Pumpe erstellt haben, können Sie diesen Stream nicht in Ihrem Code umdefinieren. Weitere Informationen zum Erstellen eines Streams finden Sie unter STREAM ERSTELLEN in der Amazon-Kinesis-Data-Analytics-SQL-Referenz. Weitere Informationen zum Erstellen eines Pumps finden Sie unter CREATE PUMP.

- Eine GROUP BY-Klausel verwendet mehrere ROWTIME-Spalten – Sie können nur eine ROWTIME-Spalte in der GROUP BY-Klausel festlegen. Weitere Informationen finden Sie unter GROUP BY und ROWTIME in der Amazon-Kinesis-Data-Analytics-SQL-Referenz.
- Ein oder mehrere Datentypen weisen eine ungültige Umwandlung auf – In diesem Fall weist Ihr Code eine ungültige implizite Umwandlung auf. Beispielsweise wandeln Sie in Ihrem Code möglicherweise einen `timestamp` in ein `bigint` um.
- Ein Stream hat denselben Namen wie ein für einen Service reservierter Stream – Ein Stream kann nicht denselben Namen haben wie der für einen Service reservierte Stream`error_stream`.

Anwendung schreibt Fehler an den Fehler-Stream

Wenn Ihre Anwendung Fehler an den In-Application-Fehler-Stream schreibt, können Sie den Wert im Feld `DATA_ROW` mittels Standard-Bibliotheken decodieren. Weitere Informationen zum Fehler-Stream finden Sie unter Fehlerbehandlung.

Ungenügender Durchsatz oder hoch MillisBehindLatest

Wenn die MillisBehindLatestKennzahl Ihrer Anwendung stetig ansteigt oder konstant über 1000 (eine Sekunde) liegt, kann dies folgende Gründe haben:

- Überprüfen Sie die InputBytes CloudWatch Metrik Ihrer Anwendung. Wenn die Verarbeitung mehr als 4 MB/s benötigt, kann dies zu einer Erhöhung von `MillisBehindLatest` führen. Um den Durchsatz Ihrer Anwendung zu verbessern, erhöhen Sie den Wert des `InputParallelism`-Parameters. Weitere Informationen finden Sie unter Parallelisieren von Eingabe-Streams zur Steigerung des Durchsatzes.
- Überprüfen Sie die Erfolg-Metrik Ihrer Anwendung für die Anwendungsausgabe nach Fehlern bei der Bereitstellung am Ziel. Vergewissern Sie sich, dass Sie die Ausgabe korrekt konfiguriert haben, und dass der Ausgabe-Stream über ausreichend Kapazität verfügt.
- Wenn Ihre Anwendung eine AWS Lambda Funktion zur Vorverarbeitung oder als Ausgabe verwendet, überprüfen Sie die Metrik .Duration oder InputProcessingLambdaDelivery CloudWatch .Duration der Anwendung. Wenn der Aufruf der Lambda-Funktion länger als 5 Sekunden dauert, ziehen Sie Folgendes in Betracht:
  - Erhöhen Sie die Speicher-Zuweisung der Lambda-Funktion. Sie können dies auf der AWS Lambda -Konsole auf der Seite Configuration (Konfiguration) unter Basic settings (Grundlegende Einstellungen) tun. Weitere Informationen finden Sie unter Konfigurieren von Lambda-Funktionen im AWS Lambda -Entwicklerhandbuch.

  - Erhöhen Sie die Anzahl der Shards im Eingabe-Stream der Anwendung. Dadurch wird die Anzahl der Aufrufe von parallelen Funktionen der Anwendung erhöht, die den Durchsatz erhöhen können.
  - Vergewissern Sie sich, dass die Funktion keine Blockierungsaufrufe macht, die die Leistung beeinträchtigen, wie z. B. synchrone Anfragen für externe Ressourcen.
  - Untersuchen Sie Ihre AWS Lambda Funktion, um festzustellen, ob es andere Bereiche gibt, in denen Sie die Leistung verbessern können. Überprüfen Sie die CloudWatch Protokolle der Lambda-Funktion der Anwendung. Weitere Informationen finden Sie unter Accessing Amazon CloudWatch Metrics for im AWS Lambda Developer Guide.
- Vergewissern Sie sich, dass Ihre Anwendung nicht die Standardbegrenzung für Kinesis Processing Units (KPU) erreicht. Wenn Ihre Anwendung diese Grenze erreicht, können Sie eine Erweiterung der Begrenzung anfordern. Weitere Informationen finden Sie unter Automatisches Skalieren von Anwendungen zur Erhöhung des Durchsatzes.
- Wenn Ihre Anwendung nach der Erhöhung Ihres KPU-Limits immer noch Probleme hat, überprüfen Sie, ob der Eingabedurchsatz Ihrer Anwendung 100 nicht überschreitet. Wir empfehlenMB/sec. If it exceeds 100MB/sec, Änderungen vorzunehmen, um den Gesamtdurchsatz zu reduzieren und die Anwendung zu stabilisieren, indem Sie beispielsweise die Datenmenge reduzieren, die an die Datenquelle gesendet wird, aus der die Kinesis Data Analytics SQL-Anwendung liest. Wir empfehlen auch andere Ansätze, darunter die Erhöhung der Parallelität der Anwendung, die Verkürzung der Berechnungsdauer, die Umstellung von spaltenförmigen Datentypen von VARCHAR auf Datentypen mit kleineren Größen (z. B. INTEGER, LONG usw.) und die Reduzierung der durch Sampling oder Filterung verarbeiteten Daten.

> **Note**
>
> Wir empfehlen, die `InputProcessing.0kBytes` Metrik Ihrer Anwendung regelmäßig zu überprüfen, damit Sie im Voraus planen können, mehrere SQL-Anwendungen zu verwenden oder zu Managed- zu migrieren. flink/latest/java/ if your application's projected input throughput will exceed 100 MB/sec

Kinesis Data Analytics-SQL-Referenz

Weitere Informationen zu den SQL-Sprachelementen, die von Kinesis Data Analytics unterstützt werden, finden Sie in der Kinesis Data Analytics-SQL-Referenz.

API-Referenz

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Dokumentation zu Amazon-Managed-Service für Apache Flink API V2.

Sie können das verwenden AWS CLI , um die Amazon Kinesis Data Analytics Analytics-API zu erkunden. Dieses Handbuch stellt Erste Schritte mit Amazon-Kinesis-Data-Analytics for SQL-Anwendungen-Übungen bereit, welche die AWS CLI verwenden.

Topics

- Aktionen
- Datentypen

Aktionen

Folgende Aktionen werden unterstützt:

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- CreateApplication
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource

- DescribeApplication
- DiscoverInputSchema
- ListApplications
- ListTagsForResource
- StartApplication
- StopApplication
- TagResource
- UntagResource
- UpdateApplication

AddApplicationCloudWatchLoggingOption

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Fügt einen CloudWatch Protokollstream hinzu, um Fehler bei der Anwendungskonfiguration zu überwachen. Weitere Informationen zur Verwendung von CloudWatch Protokollstreams mit Amazon Kinesis Analytics Analytics-Anwendungen finden Sie unter Arbeiten mit Amazon CloudWatch Logs.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "string",
      "RoleARN": "string"
   },
   "CurrentApplicationVersionId": number
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Der Name der Kinesis Analytics-Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CloudWatchLoggingOption

Stellt den Amazon Resource Name (ARN) für den CloudWatch Logstream und den ARN für die IAM-Rolle bereit. Hinweis: Um Anwendungsnachrichten schreiben zu können CloudWatch, muss für die verwendete IAM-Rolle die `PutLogEvents` Richtlinienaktion aktiviert sein.

Typ: CloudWatchLoggingOption Objekt

Erforderlich: Ja

CurrentApplicationVersionId

Die Versions-ID der Kinesis Analytics-Anwendung.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

AddApplicationInput

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Fügt Ihrer Amazon-Kinesis-Anwendung eine Streaming-Quelle hinzu. Konzeptuelle Informationen finden Sie unter Konfiguration des Anwendungseingangs.

Sie können entweder direkt bei der Erstellung einer Anwendung eine Streaming-Quelle hinzufügen oder Sie können hierzu nach Erstellung einer Anwendung diesen Vorgang verwenden. Weitere Informationen finden Sie unter CreateApplication.

Jedes Konfigurationsupdate, einschließlich dem Hinzufügen einer Streaming-Quelle mit dieser Operation, führt zu einer neuen Version der Anwendung. Mit der DescribeApplication-Operation können Sie die aktuelle Anwendungsversion ermitteln.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:AddApplicationInput`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "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"
   }
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name Ihrer vorhandenen Amazon-Kinesis-Analytics-Anwendung, zu der Sie die Streaming-Quelle hinzufügen möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Aktuelle Version Ihrer Amazon-Kinesis-Analytics-Anwendung. Mit der DescribeApplication-Operation können Sie die aktuelle Anwendungsversion ermitteln.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

Input

Die Eingabe, die hinzugefügt werden soll.

Typ: Input Objekt

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

CodeValidationException

Der vom Benutzer bereitgestellte Anwendungscode (Abfrage) ist ungültig. Dies kann ein einfacher Syntaxfehler sein.

message

Test

HTTP-Statuscode: 400

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

AddApplicationInputProcessingConfiguration

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Fügt einer InputProcessingConfigurationAnwendung eine hinzu. Ein Eingabeprozessor führt die Vorverarbeitung von Datensätzen aus, bevor der SQL-Code der Anwendung ausgeführt wird. Derzeit ist der einzige verfügbare Input-Prozessor AWS Lambda.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   }
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der Anwendung, der Sie die Ausgabeverarbeitungskonfiguration hinzufügen möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Version der Anwendung, der Sie die Ausgabeverarbeitungskonfiguration hinzufügen möchten. Sie können den DescribeApplicationVorgang verwenden, um die aktuelle Anwendungsversion abzurufen. Wenn die angegebene Version nicht die aktuelle Version ist, wird `ConcurrentModificationException` zurückgegeben.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

InputId

Die ID der Eingabekonfiguration, zu der die Eingabeverarbeitungskonfiguration hinzugefügt werden soll. Mithilfe der DescribeApplicationOperation können Sie eine Liste der Eingaben IDs für eine Anwendung abrufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

InputProcessingConfiguration

Die InputProcessingConfiguration, die der Anwendung hinzugefügt werden sollen.

Typ: InputProcessingConfiguration Objekt

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

AddApplicationOutput

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Fügt Ihrer Amazon-Kinesis-Analytics-Anwendung ein externes Ziel hinzu.

Wenn Sie möchten, dass Amazon Kinesis Analytics Daten aus einem anwendungsinternen Stream innerhalb Ihrer Anwendung an ein externes Ziel übermittelt (z. B. einen Amazon Kinesis-Stream, einen Amazon Kinesis Firehose-Lieferstream oder eine AWS Lambda-Funktion), fügen Sie Ihrer Anwendung mithilfe dieses Vorgangs die entsprechende Konfiguration hinzu. Sie können einen oder mehrere Ausgaben für Ihre Anwendung konfigurieren. Jede Ausgabekonfiguration bildet einen In-Application-Stream und ein externes Ziel ab.

Sie können eine der Ausgabekonfigurationen verwenden, um Daten aus Ihrem In-Application-Fehler-Stream in der Anwendung an ein externes Ziel zu liefern, damit Sie die Fehler analysieren können. Weitere Informationen finden Sie unter Informationen zur Anwendungsausgabe (Ziel).

Jedes Konfigurationsupdate, einschließlich dem Hinzufügen einer Streaming-Quelle mit dieser Operation, führt zu einer neuen Version der Anwendung. Mit der DescribeApplication-Operation können Sie die aktuelle Anwendungsversion ermitteln.

Die Begrenzungen für die Anzahl der konfigurierbaren Anwendungsein- und -ausgänge finden Sie unter Begrenzungen.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:AddApplicationOutput`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "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"
      }
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der Anwendung, der Sie die Ausgabekonfiguration hinzufügen möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Version der Anwendung, der Sie die Ausgabekonfiguration hinzufügen möchten. Sie können den DescribeApplicationVorgang verwenden, um die aktuelle Anwendungsversion abzurufen. Wenn die angegebene Version nicht die aktuelle Version ist, wird `ConcurrentModificationException` zurückgegeben.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

Output

Ein Array von Objekten, die jeweils eine Ausgabekonfiguration beschreiben. In der Ausgabekonfiguration geben Sie den Namen eines In-Application-Streams, eines Ziels (d. h. eines Amazon Kinesis-Streams, eines Amazon Kinesis Firehose-Lieferdatenstroms oder einer AWS Lambda-Funktion) an und zeichnen die Formation auf, die beim Schreiben in das Ziel verwendet werden soll.

Typ: Output Objekt

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

AddApplicationReferenceDataSource

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Fügt einer bestehenden Anwendung eine Referenzdatenquelle hinzu.

Amazon-Kinesis-Analytics liest Referenzdaten (d. h. ein Amazon-S3-Objekt) und erstellt eine In-Application-Tabelle innerhalb Ihrer Anwendung. In der Anforderung geben Sie die Quelle (S3-Bucket-Name und Objektschlüsselname), den Namen der zu erstellenden In-Application-Tabelle und die notwendigen Zuordnungsinformationen an. Diese beschreiben, wie Daten im Amazon-S3-Objekt auf Spalten in der resultierenden In-Application-Tabelle abgebildet werden.

Konzeptuelle Informationen finden Sie unter Konfiguration des Anwendungseingangs. Die Begrenzungen der Datenquellen, die Sie Ihrer Anwendung hinzufügen können, finden Sie unter Begrenzungen.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:AddApplicationOutput`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "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"
  }
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name einer bestehenden Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Version der Anwendung, für die Sie die Referenzdatenquelle hinzufügen. Sie können den DescribeApplicationVorgang verwenden, um die aktuelle Anwendungsversion

abzurufen. Wenn die angegebene Version nicht die aktuelle Version ist, wird `ConcurrentModificationException` zurückgegeben.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

ReferenceDataSource

Die Referenzdatenquelle kann ein Objekt in Ihrem Amazon-S3-Bucket sein. Amazon-Kinesis-Analytics liest das Objekt und kopiert die Daten in die erstellte In-Application-Tabelle. Sie stellen einen S3-Bucket, den Namen des Objektschlüssels und die daraus resultierende In-Applikation-Tabelle bereit. Sie müssen einer IAM-Rolle außerdem Berechtigungen geben, die Amazon-Kinesis-Analytics annehmen kann, um das Objekt in Ihrem S3-Bucket in Ihrem Namen zu lesen.

Typ: ReferenceDataSource Objekt

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

CreateApplication

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Erstellt eine Amazon-Kinesis-Data-Analytics-Anwendung. Sie können jede Anwendung mit einer Streaming-Quelle als Eingabe, Anwendungscode zur Verarbeitung der Eingabe und bis zu drei Zielen konfigurieren, an die Amazon-Kinesis-Analytics die Ausgabedaten aus Ihrer Anwendung schreiben soll. Einen Überblick finden Sie unter Funktionsweise.

In der Eingabekonfiguration ordnen Sie die Streaming-Quelle einem In-Application-Stream zu, den Sie sich als Tabelle vorstellen können, die sich ständig aktualisiert. Bei der Zuordnung müssen Sie ein Schema für den In-Application-Stream angeben und jede Datenspalte im In-Application-Stream einem Datenelement in der Streaming-Quelle zuordnen.

Der Code Ihrer Anwendung besteht aus einer oder mehreren SQL-Anweisungen, die Eingabedaten lesen, umwandeln und generieren. Ihr Anwendungscode kann ein oder mehrere SQL-Artefakte wie SQL-Streams oder Pumpen erstellen.

In der Ausgabekonfiguration können Sie die Anwendung so konfigurieren, dass Daten aus In-Application-Streams, die in Ihren Anwendungen erstellt wurden, an bis zu drei Ziele geschrieben werden.

Um Daten aus Ihrem Quell-Stream zu lesen oder Daten in Ziel-Streams zu schreiben, benötigt Amazon-Kinesis-Analytics Ihre Berechtigungen. Sie können diese Berechtigungen erteilen, indem Sie eine IAM-Rolle erstellen. Diese Operation erfordert zur Ausführung der `kinesisanalytics:CreateApplication`-Aktion Berechtigungen.

Einführende Übungen zur Erstellung einer Amazon-Kinesis-Analytics-Anwendung finden Sie unter Erste Schritte.

Anforderungssyntax

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

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationCode

Ein oder mehrere SQL-Anweisungen, die Eingabedaten lesen, umwandeln und generieren. Sie können beispielsweise eine SQL-Anweisung schreiben, die Daten aus einem In-Application-Stream liest, einen laufenden Durchschnitt der Anzahl der Werbeklicks pro Anbieter erzeugt und die resultierenden Zeilen in einen weiteren In-Application-Stream leitet. Weitere Informationen über das typische Muster finden Sie unter Anwendungscode.

Sie können eine solche Reihe von SQL-Anweisungen bereitstellen, bei denen die Ausgabe einer Anweisung als Eingabe für die nächste Anweisung verwendet werden kann. Sie speichern Zwischenergebnisse, indem Sie In-Application-Streams und Weiterleitungen anlegen.

Beachten Sie, dass der Anwendungscode die Streams mit den in der `Outputs` angegebenen Namen erstellen muss. Wenn Ihre `Outputs` beispielsweise Ausgabe-Streams mit den Namen `ExampleOutputStream1` und `ExampleOutputStream2` definiert, muss Ihr Anwendungscode diese Streams erstellen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge von 102400.

Erforderlich: Nein

ApplicationDescription

Zusammenfassende Beschreibung der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 1024 Zeichen.

Erforderlich: Nein

ApplicationName

Name Ihrer Amazon-Kinesis-Analytics-Anwendung (z. B. `sample-app`).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CloudWatchLoggingOptions

Verwenden Sie diesen Parameter, um einen CloudWatch Protokollstream zur Überwachung von Anwendungskonfigurationsfehlern zu konfigurieren. Weitere Informationen finden Sie unter Arbeiten mit Amazon CloudWatch Logs.

Typ: Array von CloudWatchLoggingOption-Objekten

Erforderlich: Nein

Inputs

Verwenden Sie diesen Parameter zum Konfigurieren der Anwendungseingabe.

Sie können Ihre Anwendung so konfigurieren, dass sie Eingaben von einer einzigen Streaming-Quelle erhält. In dieser Konfiguration ordnen Sie diese Streaming-Quelle einem erzeugten In-Application-Stream zu. Ihr Anwendungscode kann dann den In-Application-Stream wie eine Tabelle abfragen (Sie können sich dies als eine ständig aktualisierte Tabelle vorstellen).

Für die Streaming-Quelle, geben Sie den Amazon-Ressourcennamen (ARN) und das Format der Daten im Stream (z. B. JSON, CSV, usw.) an. Sie müssen außerdem eine IAM-Rolle angeben, die Amazon-Kinesis-Analytics übernehmen kann, um diesen Stream in Ihrem Namen zu lesen.

Um den In-Application-Stream zu erstellen, müssen Sie ein Schema für die Umwandlung Ihrer Daten in eine schematisierte Version angeben, die in SQL verwendet wird. Im Schema stellen Sie die notwendige Zuordnung der Datenelemente in der Streaming-Quelle bereit, um Spalten im In-App-Stream aufzuzeichnen.

Typ: Array von Input-Objekten

Erforderlich: Nein

Outputs

Sie können die Anwendungsausgabe so konfigurieren, dass Daten aus jedem der In-Application-Streams an bis zu drei Ziele geschrieben werden.

Bei diesen Zielen kann es sich um Amazon-Kinesis-Streams, Amazon-Kinesis Firehose-Bereitstellungsstreams, AWS -Lambda-Ziele oder eine beliebige Kombination der drei handeln.

In der Konfiguration geben Sie den Namen des In-Application-Streams, den Zielstream oder den Amazon-Ressourcennamen (ARN) der Lambda-Funktion und das Format an, das beim Schreiben

von Daten verwendet werden soll. Sie stellen außerdem eine IAM-Rolle bereit, die Amazon-Kinesis-Analytics übernehmen kann, um in Ihrem Namen in den Stream oder an die Lambda-Funktion zu schreiben.

In der Ausgabekonfiguration geben Sie auch den Ausgangsstream oder den ARN der Lambda-Funktion an. Für Stream-Ziele, geben Sie das Format der Daten im Stream (z. B. JSON, CSV) an. Sie müssen außerdem eine IAM-Rolle bereitstellen, die Amazon-Kinesis-Analytics übernehmen kann, um in Ihrem Namen in den Stream oder an die Lambda-Funktion zu schreiben.

Typ: Array von Output-Objekten

Erforderlich: Nein

Tags

Eine Liste mit einem oder mehreren Tags, die der Anwendung zugewiesen werden können. Ein Tag ist ein Schlüssel/Wert-Paar, das eine Anwendung identifiziert. Beachten Sie, dass die maximale Anzahl von Anwendungs-Tags System-Tags einschließt. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50. Weitere Informationen über das Markieren mit Tags finden Sie unter Nutzung von Tags.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

Antwortsyntax

```
{
   "ApplicationSummary": {
      "ApplicationARN": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

ApplicationSummary

Als Antwort auf Ihre `CreateApplication`-Anfrage sendet Amazon-Kinesis-Analytics eine Antwort mit einer Zusammenfassung der erstellten Anwendung, einschließlich des Amazon-Ressourcennamens (ARN), des Namens und des Status der Anwendung.

Typ: ApplicationSummary Objekt

Fehler

CodeValidationException

Der vom Benutzer bereitgestellte Anwendungscode (Abfrage) ist ungültig. Dies kann ein einfacher Syntaxfehler sein.

message

Test

HTTP-Statuscode: 400

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

LimitExceededException

Die Anzahl der zulässigen Anwendungen wurde überschritten.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

TooManyTagsException

Anwendung, die mit zu vielen Tags erstellt wurde, oder einer Anwendung wurden zu viele Tags hinzugefügt. Beachten Sie, dass die maximale Anzahl von Anwendungs-Tags System-Tags einschließt. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50.

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3

- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteApplication

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Löscht die angegebene Anwendung. Amazon-Kinesis-Analytics stoppt die Anwendungsausführung und löscht die Anwendung, einschließlich aller Anwendungsartefakte (z. B. In-App-Streams, Referenztabelle und Anwendungscode).

Diese Operation erfordert zur Ausführung der `kinesisanalytics:DeleteApplication`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CreateTimestamp": number
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der Amazon-Kinesis-Analytics-Anwendung, die gelöscht werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CreateTimestamp

Sie können den Vorgang `DescribeApplication` verwenden, um diesen Wert abzurufen.

Typ: Zeitstempel

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteApplicationCloudWatchLoggingOption

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Löscht einen CloudWatch Protokollstream aus einer Anwendung. Weitere Informationen zur Verwendung von CloudWatch Log-Streams mit Amazon Kinesis Analytics Analytics-Anwendungen finden Sie unter Arbeiten mit Amazon CloudWatch Logs.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOptionId": "string",
   "CurrentApplicationVersionId": number
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Der Name der Kinesis Analytics-Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CloudWatchLoggingOptionId

Die `CloudWatchLoggingOptionId` CloudWatch Logging-Option, die gelöscht werden soll. Sie können das `CloudWatchLoggingOptionId` mithilfe der DescribeApplicationOperation abrufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Die Versions-ID der Kinesis Analytics-Anwendung.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteApplicationInputProcessingConfiguration

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Löscht eine InputProcessingConfigurationaus einer Eingabe.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string"
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Der Name der Kinesis Analytics-Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Die Versions-ID der Kinesis Analytics-Anwendung.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

InputId

Die ID der Eingabekonfiguration, aus der die Eingabeverarbeitungskonfiguration gelöscht werden soll. Mithilfe der DescribeApplicationOperation können Sie eine Liste der Eingaben IDs für eine Anwendung abrufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteApplicationOutput

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Löscht die Ausgabezielkonfiguration aus Ihrer Anwendungskonfiguration. Amazon-Kinesis-Analytics schreibt keine Daten mehr aus dem entsprechenden In-Application-Stream in das externe Ausgabeziel.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:DeleteApplicationOutput`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "OutputId": "string"
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Amazon-Kinesis-Analytics-Anwendungsname.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Amazon-Kinesis-Analytics-Anwendungsversion. Sie können den DescribeApplicationVorgang verwenden, um die aktuelle Anwendungsversion abzurufen. Wenn die angegebene Version nicht die aktuelle Version ist, wird `ConcurrentModificationException` zurückgegeben.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

OutputId

Die ID der Konfiguration, die gelöscht werden soll. Jede Ausgabekonfiguration, die der Anwendung hinzugefügt wird, entweder bei der Erstellung der Anwendung oder später mithilfe des AddApplicationOutputVorgangs, hat eine eindeutige ID. Sie müssen die ID angeben, um die Ausgabekonfiguration, die Sie aus der Anwendungskonfiguration löschen möchten, eindeutig zu identifizieren. Sie können die DescribeApplicationOperation verwenden, um die spezifische Information abzurufen`OutputId`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteApplicationReferenceDataSource

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Löscht eine Referenzdatenquellenkonfiguration aus der angegebenen Anwendungskonfiguration.

Wenn die Anwendung ausgeführt wird, entfernt Amazon Kinesis Analytics sofort die anwendungsinterne Tabelle, die Sie mit dem AddApplicationReferenceDataSourceVorgang erstellt haben.

Diese Operation erfordert zur Ausführung der `kinesisanalytics.DeleteApplicationReferenceDataSource`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceId": "string"
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name einer bestehenden Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

CurrentApplicationVersionId

Die Version der Anwendung. Sie können den DescribeApplicationVorgang verwenden, um die aktuelle Anwendungsversion abzurufen. Wenn die angegebene Version nicht die aktuelle Version ist, wird `ConcurrentModificationException` zurückgegeben.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

ReferenceId

Die ID der Referenzdatenquelle. Wenn Sie Ihrer Anwendung mithilfe von eine Referenzdatenquelle hinzufügen AddApplicationReferenceDataSource, weist Amazon Kinesis Analytics eine ID zu. Sie können den DescribeApplicationVorgang verwenden, um die Referenz-ID abzurufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DescribeApplication

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Gibt Informationen hinsichtlich einer bestimmten Amazon-Kinesis-Analytics-Anwendung zurück.

Wenn Sie eine Liste aller Anwendungen in Ihrem Konto abrufen möchten, verwenden Sie den ListApplicationsVorgang.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:DescribeApplication`-Aktion Berechtigungen. Sie können `DescribeApplication` verwenden, um die aktuelle versionId der Anwendung abzurufen, die Sie benötigen, um andere Operationen wie `Update` aufzurufen.

Anforderungssyntax

```
{
   "ApplicationName": "string"
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Antwortsyntax

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

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

ApplicationDetail

Stellt eine Beschreibung der Anwendung bereit, z. B. den Amazon-Ressourcennamen (ARN) der Anwendung, den Status, die neueste Version sowie Informationen zur Eingabe- und Ausgabekonfiguration.

Typ: ApplicationDetail Objekt

Fehler

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3

- AWS SDK für Python
- AWS SDK for Ruby V3

DiscoverInputSchema

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Leitet ein Schema ab, indem Beispieldatensätze für die angegebene Streaming-Quelle (Amazon-Kinesis-Stream oder Amazon-Kinesis Firehose-Bereitstellungsstream) oder das S3-Objekt ausgewertet werden. In der Antwort gibt die Operation das abgeleitete Schema und auch die Beispieldatensätze zurück, die bei der Operation zur Ableitung des Schemas verwendet wurden.

Sie können das abgeleitete Schema bei der Konfiguration einer Streaming-Quelle für Ihre Anwendung verwenden. Konzeptuelle Informationen finden Sie unter Konfiguration des Anwendungseingangs. Beachten Sie, dass, wenn Sie eine Anwendung mit der Amazon-Kinesis-Analytics-Konsole erstellen, die Konsole diesen Vorgang verwendet, um ein Schema abzuleiten und es in der Benutzeroberfläche der Konsole anzuzeigen.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:DiscoverInputSchema`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "string"
   },
   "ResourceARN": "string",
   "RoleARN": "string",
   "S3Configuration": {
      "BucketARN": "string",
```

```
        "FileKey": "string",
        "RoleARN": "string"
    }
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

InputProcessingConfiguration

Der InputProcessingConfiguration, der verwendet werden soll, um die Datensätze vorzuverarbeiten, bevor das Schema der Datensätze ermittelt wird.

Typ: InputProcessingConfiguration Objekt

Erforderlich: Nein

InputStartingPositionConfiguration

Punkt, an dem Amazon-Kinesis-Analytics mit dem Lesen von Datensätzen für den angegebenen Zweck der Streaming-Quellenerkennung beginnen soll.

Typ: InputStartingPositionConfiguration Objekt

Erforderlich: Nein

ResourceARN

Der Amazon-Ressourcenname (ARN) der Streaming-Quelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

S3Configuration

Geben Sie diesen Parameter an, um ein Schema anhand von Daten in einem Amazon-S3-Objekt zu ermitteln.

Typ: S3Configuration Objekt

Erforderlich: Nein

Antwortsyntax

```
{
   "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" ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

InputSchema

Aus der Streaming-Quelle abgeleitetes Schema. Erkennt das Format der Daten in der Streaming-Quelle, und wie jedes Datenelement auf die entsprechenden Spalten im In-Application-Stream abgebildet wird, die Sie erzeugen können.

Typ: SourceSchema Objekt

ParsedInputRecords

Ein Array von Elementen, wobei jedes Element einer Zeile in einem Stream-Datensatz entspricht (ein Stream-Datensatz kann mehr als eine Zeile enthalten).

Typ: Array von Arrays von Strings

ProcessedInputRecords

Streamen Sie Daten, die von dem im `InputProcessingConfiguration`-Parameter angegebenen Prozessor geändert wurden.

Typ: Zeichenfolgen-Array

RawInputRecords

Stream-Rohdaten, aus denen Stichproben entnommen wurden, um das Schema abzuleiten.

Typ: Zeichenfolgen-Array

Fehler

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceProvisionedThroughputExceededException

Discovery konnte aufgrund der Amazon Kinesis Streams ProvisionedThroughputExceededException keinen Datensatz von der Streaming-Quelle abrufen. Weitere Informationen finden Sie GetRecordsin der Amazon Kinesis Streams API-Referenz.

HTTP-Statuscode: 400

ServiceUnavailableException

Der Service ist nicht verfügbar. Gehen Sie zurück und versuchen Sie den Vorgang erneut.

HTTP Status Code: 500

UnableToDetectSchemaException

Das Datumsformat ist nicht gültig. Amazon-Kinesis-Analytics kann für die angegebene Streaming-Quelle kein Schema erkennen.

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

ListApplications

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Gibt eine Liste der Amazon-Kinesis-Analytics-Anwendungen in Ihrem Konto zurück. Die Antwort enthält den Anwendungsnamen, Amazon-Ressourcennamen (ARN) und Status für jede Anwendung. Wenn die Antwort den Wert „true“ für `HasMoreApplications` zurückgibt, können Sie eine weitere Anfrage senden, indem Sie `ExclusiveStartApplicationName` in den Anfragetext einfügen und den letzten Anwendungsnamen aus der vorherigen Antwort als Wert dafür einsetzen.

Wenn Sie detaillierte Informationen zu einer bestimmten Anwendung wünschen, verwenden Sie DescribeApplication.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:ListApplications`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "ExclusiveStartApplicationName": "string",
   "Limit": number
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ExclusiveStartApplicationName

Name der Anwendung, mit der die Liste beginnen soll. Wenn Sie eine Paginierung verwenden, um die Liste abzurufen, müssen Sie diesen Parameter nicht in der ersten Anfrage angeben. In nachfolgenden Anfragen fügen Sie jedoch den letzten Anwendungsnamen aus der vorherigen Antwort hinzu, um die nächste Seite mit Anwendungen zu erhalten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Nein

Limit

Maximale Anzahl von Anwendungen, für die Auflistung.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

Erforderlich: Nein

Antwortsyntax

```
{
   "ApplicationSummaries": [
      {
         "ApplicationARN": "string",
         "ApplicationName": "string",
         "ApplicationStatus": "string"
      }
   ],
   "HasMoreApplications": boolean
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

ApplicationSummaries

Liste von `ApplicationSummary`-Objekten.

Typ: Array von ApplicationSummary-Objekten

HasMoreApplications

Gibt „true“ zurück, wenn mehr Anwendungen abgerufen werden müssen.

Typ: Boolescher Wert

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

ListTagsForResource

Ruft die Liste der Schlüssel-Wert-Tags ab, die der Anwendung zugewiesen sind. Weitere Informationen finden Sie unter Nutzung von Tags.

Anforderungssyntax

```
{
   "ResourceARN": "string"
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ResourceARN

Der ARN der Anwendung, für welche Tags abgerufen werden sollen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Antwortsyntax

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Tags

Die der Anwendung zugewiesenen Schlüssel-Wert-Tags.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 200 Elemente.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

StartApplication

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Startet die angegebene Amazon-Kinesis-Analytics-Anwendung. Nachdem Sie eine Anwendung erstellt haben, müssen Sie diesen Vorgang exklusiv aufrufen, um Ihre Anwendung zu starten.

Nach dem Start der Anwendung beginnt sie mit der Aufnahme der Eingabedaten, verarbeitet sie und schreibt die Ausgabe in das konfigurierte Ziel.

Der Anwendungsstatus muss `READY` sein, damit Sie eine Anwendung starten können. Sie können den Anwendungsstatus in der Konsole oder mithilfe des DescribeApplicationVorgangs abrufen.

Nachdem Sie die Anwendung gestartet haben, können Sie verhindern, dass die Anwendung die Eingabe verarbeitet, indem Sie den StopApplicationVorgang aufrufen.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:StartApplication`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "ApplicationName": "string",
   "InputConfigurations": [
      {
         "Id": "string",
         "InputStartingPositionConfiguration": {
            "InputStartingPosition": "string"
         }
      }
   ]
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

InputConfigurations

Identifiziert anhand der ID die spezifische Eingabe, mit deren Aufnahme die Anwendung beginnt. Amazon-Kinesis-Analytics beginnt mit dem Lesen der Streaming-Quelle, die der Eingabe zugeordnet ist. Sie können auch angeben, wo in der Streaming-Quelle Amazon-Kinesis-Analytics mit dem Lesen beginnen soll.

Typ: Array von InputConfiguration-Objekten

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

InvalidApplicationConfigurationException

Die vom Nutzer angegebene Anwendungskonfiguration ist ungültig.

message

Test

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

StopApplication

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Stoppt die Verarbeitung von Eingabedaten durch die Anwendung. Sie können eine Anwendung nur dann beenden, wenn sie sich im laufenden Zustand befindet. Sie können den DescribeApplicationVorgang verwenden, um den Status der Anwendung zu ermitteln. Nachdem die Anwendung gestoppt wurde, beendet Amazon-Kinesis-Analytics das Lesen von Daten aus der Eingabe, die Anwendung beendet die Verarbeitung von Daten und es wird keine Ausgabe in das Ziel geschrieben.

Diese Operation erfordert zur Ausführung der `kinesisanalytics:StopApplication`-Aktion Berechtigungen.

Anforderungssyntax

```
{
   "ApplicationName": "string"
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der laufenden Anwendung, die beendet werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

TagResource

Fügt einer Kinesis Analytics-Anwendung ein oder mehrere Schlüssel-Wert-Tags hinzu. Beachten Sie, dass die maximale Anzahl von Anwendungs-Tags System-Tags einschließt. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50. Weitere Informationen über das Markieren mit Tags finden Sie unter Nutzung von Tags.

Anforderungssyntax

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

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ResourceARN

Der ARN der Anwendung, der die Tags zugewiesen werden sollen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Tags

Die Schlüssel-Wert-Tags, die der Anwendung zugewiesen werden sollen.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

TooManyTagsException

Anwendung, die mit zu vielen Tags erstellt wurde, oder einer Anwendung wurden zu viele Tags hinzugefügt. Beachten Sie, dass die maximale Anzahl von Anwendungs-Tags System-Tags einschließt. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

UntagResource

Entfernt ein oder mehrere Tags aus einer Kinesis Analytics-Anwendung. Weitere Informationen über das Markieren mit Tags finden Sie unter Nutzung von Tags.

Anforderungssyntax

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ResourceARN

Der ARN der Kinesis Analytics-Anwendung, aus der die Tags entfernt werden sollen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

TagKeys

Eine Liste von Schlüsseln für Tags, die aus der angegebenen Anwendung entfernt werden sollen.

Typ: Zeichenfolgen-Array

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 200 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

TooManyTagsException

Anwendung, die mit zu vielen Tags erstellt wurde, oder einer Anwendung wurden zu viele Tags hinzugefügt. Beachten Sie, dass die maximale Anzahl von Anwendungs-Tags System-Tags einschließt. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

UpdateApplication

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Aktualisiert eine bestehende Amazon-Kinesis-Analytics-Anwendung. Mit dieser API können Sie den Anwendungscode, die Eingabekonfiguration und die Ausgabekonfiguration aktualisieren.

Beachten Sie, dass Amazon-Kinesis-Analytics die `CurrentApplicationVersionId` jedes Mal aktualisiert, wenn Sie Ihre Anwendung aktualisieren.

Diese Operation setzt eine Berechtigung für die `kinesisanalytics:UpdateApplication`-Aktion voraus.

Anforderungssyntax

```
{
   "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
}
```

Anforderungsparameter

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ApplicationName

Name der zu aktualisierenden Amazon-Kinesis-Analytics-Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

ApplicationUpdate

Beschreibt Anwendungsupdates.

Typ: ApplicationUpdate Objekt

Erforderlich: Ja

CurrentApplicationVersionId

Die aktuelle Anwendungsversions-ID. Sie können den Vorgang DescribeApplication verwenden, um diesen Wert abzurufen.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

CodeValidationException

Der vom Benutzer bereitgestellte Anwendungscode (Abfrage) ist ungültig. Dies kann ein einfacher Syntaxfehler sein.

message

Test

HTTP-Statuscode: 400

ConcurrentModificationException

Ausnahme, die als Ergebnis einer gleichzeitigen Änderung an einer Anwendung ausgelöst wurde. Beispiel: Zwei Personen versuchen, dieselbe Anwendung gleichzeitig zu bearbeiten.

message

HTTP-Statuscode: 400

InvalidArgumentException

Der angegebene Eingabeparameterwert ist ungültig.

message

HTTP-Statuscode: 400

ResourceInUseException

Die Anwendung ist für diesen Vorgang nicht verfügbar.

message

HTTP-Statuscode: 400

ResourceNotFoundException

Die angegebene Anwendung kann nicht gefunden werden.

message

HTTP-Statuscode: 400

UnsupportedOperationException

Die Anfrage wurde abgelehnt, weil ein bestimmter Parameter nicht unterstützt wird oder eine angegebene Ressource für diesen Vorgang nicht gültig ist.

HTTP-Statuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS Befehlszeilenschnittstelle V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK für Kotlin
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

Datentypen

Die folgenden Datentypen werden unterstützt:

- 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**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Stellt eine Beschreibung der Anwendung bereit, einschließlich des Amazon-Ressourcennamens (ARN) der Anwendung, des Status, der neuesten Version sowie der Eingabe- und Ausgabekonfiguration.

Inhalt

ApplicationARN

ARN der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

ApplicationName

Name der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

ApplicationStatus

Status der Anwendung.

Typ: Zeichenfolge

Zulässige Werte: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

Erforderlich: Ja

ApplicationVersionId

Gibt die aktuelle Anwendungsversion an.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 999999999.

Erforderlich: Ja

ApplicationCode

Gibt den Anwendungscode zurück, den Sie zur Durchführung der Datenanalyse für einen der In-Application-Streams in Ihrer Anwendung angegeben haben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge von 102400.

Erforderlich: Nein

ApplicationDescription

Beschreibung der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 1024 Zeichen.

Erforderlich: Nein

CloudWatchLoggingOptionDescriptions

Beschreibt die CloudWatch Protokollstreams, die für den Empfang von Anwendungsnachrichten konfiguriert sind. Weitere Informationen zur Verwendung von CloudWatch Log-Streams mit

Amazon Kinesis Analytics Analytics-Anwendungen finden Sie unter Arbeiten mit Amazon CloudWatch Logs.

Typ: Array von CloudWatchLoggingOptionDescription-Objekten

Erforderlich: Nein

CreateTimestamp

Zeitstempel der Erstellung der Anwendung.

Typ: Zeitstempel

Erforderlich: Nein

InputDescriptions

Beschreibt die Konfiguration der Anwendungseingabe. Weitere Informationen finden Sie unter Konfigurieren der Anwendungseingabe.

Typ: Array von InputDescription-Objekten

Erforderlich: Nein

LastUpdateTimestamp

Zeitstempel der letzten Aktualisierung der Anwendung.

Typ: Zeitstempel

Erforderlich: Nein

OutputDescriptions

Beschreibt die Ausgabekonfiguration der Anwendung. Weitere Informationen finden Sie unter Konfigurieren der Anwendungsausgabe.

Typ: Array von OutputDescription-Objekten

Erforderlich: Nein

ReferenceDataSourceDescriptions

Beschreibt Referenzdatenquellen, die für die Anwendung konfiguriert sind. Weitere Informationen finden Sie unter Konfigurieren der Anwendungseingabe.

Typ: Array von ReferenceDataSourceDescription-Objekten

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ApplicationSummary

> **Note**
>
> Diese Dokumentation bezieht sich auf Version 1 der Amazon-Kinesis-Data-Analytics-API, die nur SQL-Anwendungen unterstützt. Version 2 der API unterstützt SQL- und Java-Anwendungen. Weitere Informationen zu Version 2 finden Sie in der Amazon-Kinesis-Data-Analytics-API-V2-Dokumentation.

Stellt Informationen bereit, darunter den Amazon-Ressourcennamen (ARN), Namen und Status der Anwendung.

Inhalt

ApplicationARN

ARN der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

ApplicationName

Name der Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

ApplicationStatus

Status der Anwendung.

Typ: Zeichenfolge

Zulässige Werte: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ApplicationUpdate

Beschreibt Updates, die auf eine bestehende Amazon Kinesis Analytics-Anwendung angewendet werden sollen.

Inhalt

ApplicationCodeUpdate

Beschreibt Aktualisierungen des Anwendungscodes.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge von 102400.

Erforderlich: Nein

CloudWatchLoggingOptionUpdates

Beschreibt Aktualisierungen der Optionen für die CloudWatch Anwendungsprotokollierung.

Typ: Array von CloudWatchLoggingOptionUpdate-Objekten

Erforderlich: Nein

InputUpdates

Beschreibt Aktualisierungen der Konfiguration der Anwendungseingabe.

Typ: Array von InputUpdate-Objekten

Erforderlich: Nein

OutputUpdates

Beschreibt Aktualisierungen der Konfiguration der Anwendungsausgabe.

Typ: Array von OutputUpdate-Objekten

Erforderlich: Nein

ReferenceDataSourceUpdates

Beschreibt Aktualisierungen der Anwendungsreferenzdatenquellen.

Typ: Array von ReferenceDataSourceUpdate-Objekten

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOption

Enthält eine Beschreibung der CloudWatch Protokollierungsoptionen, einschließlich des Amazon Resource Name (ARN) des Log-Streams und der Rolle ARN.

Inhalt

LogStreamARN

ARN des CloudWatch Protokolls zum Empfangen von Anwendungsnachrichten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

IAM-ARN der Rolle, die zum Senden von Anwendungsnachrichten verwendet werden soll. Hinweis: Um Anwendungsnachrichten schreiben zu können CloudWatch, muss für die verwendete IAM-Rolle die `PutLogEvents` Richtlinienaktion aktiviert sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

CloudWatchLoggingOptionDescription

Beschreibung der CloudWatch Protokollierungsoption.

Inhalt

LogStreamARN

ARN des CloudWatch Protokolls zum Empfangen von Anwendungsnachrichten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

IAM-ARN der Rolle, die zum Senden von Anwendungsnachrichten verwendet werden soll. Hinweis: Um Anwendungsnachrichten schreiben zu können CloudWatch, muss für die verwendete IAM-Rolle die `PutLogEvents` Richtlinienaktion aktiviert sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

CloudWatchLoggingOptionId

ID der Beschreibung der CloudWatch Protokollierungsoption.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOptionUpdate

Beschreibt die CloudWatch Protokollierung von Optionsaktualisierungen.

Inhalt

CloudWatchLoggingOptionId

ID der zu aktualisierenden CloudWatch Protokollierungsoption

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

LogStreamARNUpdate

ARN des CloudWatch Protokolls zum Empfangen von Anwendungsnachrichten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

IAM-ARN der Rolle, die zum Senden von Anwendungsnachrichten verwendet werden soll. Hinweis: Um Anwendungsnachrichten schreiben zu können CloudWatch, muss für die verwendete IAM-Rolle die `PutLogEvents` Richtlinienaktion aktiviert sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CSVMappingParameters

Bietet zusätzliche Zuordnungsinformationen, wenn das Datensatzformat Trennzeichen verwendet, beispielsweise CSV. Beispiel: Die folgenden Beispieldatensätze verwenden das CSV-Format, wobei die Datensätze das "\n" als Zeilentrennzeichen und ein Komma (",") als Spaltentrennzeichen verwenden:

```
"name1", "address1"

"name2", "address2"
```

Inhalt

RecordColumnDelimiter

Das Spaltentrennzeichen. Bei einem CSV-Format beispielsweise ist ein Komma (",") das typische Spaltentrennzeichen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Ja

RecordRowDelimiter

Das Zeilentrennzeichen. Bei einem CSV-Format beispielsweise ist "\n" das typische Zeilentrennzeichen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

DestinationSchema

Beschreibt das Datenformat, wenn Datensätze in das Ziel geschrieben werden. Weitere Informationen finden Sie unter Konfigurieren der Anwendungsausgabe.

Inhalt

RecordFormatType

Gibt das Format der Datensätze im Ausgabe-Stream an.

Typ: Zeichenfolge

Zulässige Werte: `JSON | CSV`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Input

Wenn Sie die Anwendungseingabe konfigurieren, geben Sie die Streaming-Quelle an, den Namen des In-Application-Streams, der erstellt wird, sowie die Zuordnung zwischen den beiden. Weitere Informationen finden Sie unter Konfigurieren der Anwendungseingabe.

Inhalt

InputSchema

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden Spalten im erzeugten In-Application-Stream zugeordnet wird.

Wird auch zum Beschreiben des Formats der Referenzdatenquelle verwendet.

Typ: SourceSchema Objekt

Erforderlich: Ja

NamePrefix

Das Namenspräfix zum Erstellen eines In-Application-Streams. Angenommen, Sie geben ein Präfix "“ anMyInApplicationStream. Amazon Kinesis Analytics erstellt dann einen oder mehrere (gemäß der von Ihnen angegebenen `InputParallelism` Anzahl) anwendungsinterne Streams mit den Namen "MyInApplicationStream_001“, "MyInApplicationStream_002“ usw.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Ja

InputParallelism

Gibt die Anzahl der zu erstellenden In-Application-Streams an.

Die Daten aus Ihrer Quelle werden in diese In-Application-Eingabe-Streams geleitet.

(siehe Konfigurieren der Anwendungseingabe.

Typ: InputParallelism Objekt

Erforderlich: Nein

InputProcessingConfiguration

Der für die Eingabe InputProcessingConfiguration. Ein Eingabeprozessor wandelt Datensätze um, sobald sie vom Stream empfangen werden, bevor der SQL-Code der Anwendung ausgeführt wird. Derzeit ist als einzige Eingabeverarbeitungskonfiguration InputLambdaProcessor verfügbar.

Typ: InputProcessingConfiguration Objekt

Erforderlich: Nein

KinesisFirehoseInput

Identifiziert, wenn die Streaming-Quelle ein Amazon Kinesis Firehose-Delivery-Stream ist, den ARN des Delivery-Streams und eine IAM-Rolle, die Amazon Kinesis Analytics ermöglicht, für Sie auf den Stream zuzugreifen.

Hinweis: Entweder `KinesisStreamsInput` oder `KinesisFirehoseInput` ist erforderlich.

Typ: KinesisFirehoseInput Objekt

Erforderlich: Nein

KinesisStreamsInput

Identifiziert, wenn die Streaming-Quelle ein Amazon Kinesis-Stream ist, den Amazon Resource Name (ARN) des Streams und eine IAM-Rolle, die Amazon Kinesis Analytics ermöglicht, für Sie auf den Stream zuzugreifen.

Hinweis: Entweder `KinesisStreamsInput` oder `KinesisFirehoseInput` ist erforderlich.

Typ: KinesisStreamsInput Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputConfiguration

Wenn Sie Ihre Anwendung starten, geben Sie diese Konfiguration an, die die Eingabequelle und den Punkt in der Eingabequelle identifiziert, an dem die Anwendung mit der Verarbeitung von Datensätzen beginnen soll.

Inhalt

Id

ID der Eingabequelle. Sie können diese ID erhalten, indem Sie die DescribeApplicationOperation aufrufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

InputStartingPositionConfiguration

Punkt, an dem die Anwendung mit der Verarbeitung von Datensätzen aus der Streaming-Quelle beginnen soll.

Typ: InputStartingPositionConfiguration Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputDescription

Beschreibt die Konfiguration der Anwendungseingabe. Weitere Informationen finden Sie unter Konfigurieren der Anwendungseingabe.

Inhalt

InAppStreamNames

Gibt die Namen der In-App-Streams zurück, die der Stream-Quelle zugeordnet sind.

Typ: Zeichenfolgen-Array

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Nein

InputId

Eingabe-ID, der die Eingabe der Anwendung zugeordnet ist. Dies ist die ID, die Amazon-Kinesis-Analytics jeder Eingabekonfiguration zuweist, die Sie Ihrer Anwendung hinzufügen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Nein

InputParallelism

Gibt die konfigurierte Parallelität an (Anzahl der der Streaming-Quelle zugewiesenen In-Application-Streams).

Typ: InputParallelism Objekt

Erforderlich: Nein

InputProcessingConfigurationDescription

Die Beschreibung des Präprozessors, der für Datensätze in dieser Eingabe ausgeführt wird, bevor der Code der Anwendung ausgeführt wird.

Typ: InputProcessingConfigurationDescription Objekt

Erforderlich: Nein

InputSchema

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden Spalten im erzeugten In-Application-Stream zugeordnet wird.

Typ: SourceSchema Objekt

Erforderlich: Nein

InputStartingPositionConfiguration

Punkt, an dem die Anwendung, gemäß ihrer Konfiguration, aus dem Eingabestream liest.

Typ: InputStartingPositionConfiguration Objekt

Erforderlich: Nein

KinesisFirehoseInputDescription

Wenn ein Amazon-Kinesis-Firehose-Bereitstellungs-Stream als Streaming-Quelle konfiguriert ist, stellt er den ARN des Bereitstellungs-Streams und eine IAM-Rolle bereit, die es Amazon-Kinesis-Analytics ermöglicht, für Sie auf den Stream zuzugreifen.

Typ: KinesisFirehoseInputDescription Objekt

Erforderlich: Nein

KinesisStreamsInputDescription

Wenn ein Amazon-Kinesis Stream als Streaming-Quelle konfiguriert ist, stellt er den Amazon-Ressourcennamen (ARN) des Amazon-Kinesis Streams und eine IAM-Rolle bereit, die es Amazon-Kinesis-Analytics ermöglicht, für Sie auf den Stream zuzugreifen.

Typ: KinesisStreamsInputDescription Objekt

Erforderlich: Nein

NamePrefix

In-App-Namenspräfix.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessor

Ein Objekt, das den Amazon-Ressourcennamen (ARN) der AWS Lambda-Funktion enthält, die zur Vorverarbeitung von Datensätzen im Stream verwendet wird, und den ARN der IAM-Rolle, die für den Zugriff auf die AWS Lambda-Funktion verwendet wird.

Inhalt

ResourceARN

Der ARN der AWS -Lambda-Funktion, die mit den Datensätzen im Stream arbeitet.

> **Note**
>
> Zum Angeben einer früheren Version der Lambda-Funktion als die neueste, fügen Sie die Version der Lambda-Funktion im ARN der Lambda-Funktion ein. Weitere Informationen zu Lambda finden Sie unter ARNs Beispiel ARNs: AWS Lambda

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

Der ARN der IAM-Rolle, die für den Zugriff auf die AWS Lambda-Funktion verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessorDescription

Ein Objekt, das den Amazon-Ressourcennamen (ARN) der AWS Lambda-Funktion enthält, die zur Vorverarbeitung von Datensätzen im Stream verwendet wird, und den ARN der IAM-Rolle, die für den Zugriff auf den AWS Lambda-Ausdruck verwendet wird.

Inhalt

ResourceARN

Der ARN der AWS -Lambda-Funktion, die zur Vorverarbeitung der Datensätze im Stream verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, die für den Zugriff auf die AWS Lambda-Funktion verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

InputLambdaProcessorUpdate

Stellt eine Aktualisierung von dar InputLambdaProcessor, die zur Vorverarbeitung der Datensätze im Stream verwendet wird.

Inhalt

ResourceARNUpdate

Der Amazon-Ressourcenname (ARN) der neuen AWS -Lambda-Funktion mit dem die Datensätze im Stream vorverarbeitet werden.

> **Note**
>
> Zum Angeben einer früheren Version der Lambda-Funktion als die neueste, fügen Sie die Version der Lambda-Funktion im ARN der Lambda-Funktion ein. Weitere Informationen zu Lambda finden Sie unter ARNs Beispiel ARNs: AWS Lambda

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

Der ARN der neuen IAM-Rolle, die für den Zugriff auf die AWS Lambda-Funktion verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputParallelism

Gibt die Anzahl der In-Application-Streams an, die für eine bestimmte Streaming-Quelle erstellt werden sollen. Weitere Informationen zur Parallelität finden Sie unter Konfigurieren der Anwendungseingabe.

Inhalt

Count

Die Anzahl der zu erstellenden In-Application-Streams. Weitere Informationen finden Sie unter Limits.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 64.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputParallelismUpdate

Stellt Aktualisierungen für den Parallelitätenzähler bereit.

Inhalt

CountUpdate

Gibt die Anzahl der In-Application-Streams an, die für die angegebene Streaming-Quelle erstellt werden sollen.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 64.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfiguration

Stellt eine Beschreibung eines Prozessors bereit, der für die Vorverarbeitung der Datensätze im Stream verwendet wird, bevor sie von Ihrem Anwendungscode verarbeitet werden. Derzeit ist der einzige verfügbare Input-Prozessor AWS Lambda.

Inhalt

InputLambdaProcessor

InputLambdaProcessorDas wird verwendet, um die Datensätze im Stream vorzuverarbeiten, bevor sie von Ihrem Anwendungscode verarbeitet werden.

Typ: InputLambdaProcessor Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfigurationDescription

Stellt Konfigurationsinformationen zu einem Eingabeprozessor bereit. Derzeit ist der einzige verfügbare Input-Prozessor AWS Lambda.

Inhalt

InputLambdaProcessorDescription

Stellt Konfigurationsinformationen zu den zugehörigen bereit InputLambdaProcessorDescription.

Typ: InputLambdaProcessorDescription Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfigurationUpdate

Beschreibt Updates für eine InputProcessingConfiguration.

Inhalt

InputLambdaProcessorUpdate

Stellt Aktualisierungsinformationen für eine bereit InputLambdaProcessor.

Typ: InputLambdaProcessorUpdate Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputSchemaUpdate

Beschreibt Aktualisierungen für das Eingabeschema der Anwendung.

Inhalt

RecordColumnUpdates

Eine Liste von `RecordColumn`-Objekten. Jedes Objekt beschreibt die Zuordnung des Streaming-Quell-Elements zu der entsprechenden Spalte im In-Application-Stream.

Typ: Array von RecordColumn-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 1000 Elemente.

Erforderlich: Nein

RecordEncodingUpdate

Gibt die Codierung der Datensätze in der Streaming-Quelle an. Zum Beispiel UTF-8.

Typ: Zeichenfolge

Pattern: `UTF-8`

Erforderlich: Nein

RecordFormatUpdate

Gibt das Format der Datensätze in der Streaming-Quelle an.

Typ: RecordFormat Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

InputStartingPositionConfiguration

Beschreibt den Punkt, an dem die Anwendung aus der Streaming-Quelle liest.

Inhalt

InputStartingPosition

Die Startposition im Stream.

- `NOW` - Beginnt unmittelbar nach dem letzten Datensatz im Stream mit dem Anforderungszeitstempel, den der Kunde ausgegeben hat, zu lesen.
- `TRIM_HORIZON`- Beginnt am letzten nicht getrimmten Datensatz im Stream, der der älteste im Stream verfügbare Datensatz ist, zu lesen. Diese Option ist für einen Amazon-Kinesis-Firehose-Bereitstellungs-Stream nicht verfügbar.
- `LAST_STOPPED_POINT` - Setzt den Lesevorgang an der Stelle fort, an der die Anwendung zuletzt aufgehört hat.

Typ: Zeichenfolge

Zulässige Werte: `NOW | TRIM_HORIZON | LAST_STOPPED_POINT`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputUpdate

Beschreibt Aktualisierungen einer bestimmten Eingabekonfiguration (identifiziert durch die `InputId` einer Anwendung).

Inhalt

InputId

Eingabe-ID der Anwendungseingabe, die aktualisiert werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

InputParallelismUpdate

Beschreibt die Parallelitätsaktualisierungen (die Anzahl an In-App-Streams, die Amazon-Kinesis-Analytics für die spezifische Streaming-Quelle erstellt).

Typ: InputParallelismUpdate Objekt

Erforderlich: Nein

InputProcessingConfigurationUpdate

Beschreibt Updates für eine Konfiguration der Eingabeverarbeitung.

Typ: InputProcessingConfigurationUpdate Objekt

Erforderlich: Nein

InputSchemaUpdate

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden Spalten im erzeugten In-Application-Stream zugeordnet wird.

Typ: InputSchemaUpdate Objekt

Erforderlich: Nein

KinesisFirehoseInputUpdate

Wenn ein Amazon-Kinesis-Firehose-Bereitstellungs-Stream die zu aktualisierende Streaming-Quelle ist, stellt er einen aktualisierten Stream-ARN und einen aktualisierten IAM-Rollen-ARN bereit.

Typ: KinesisFirehoseInputUpdate Objekt

Erforderlich: Nein

KinesisStreamsInputUpdate

Wenn ein Amazon-Kinesis Stream die zu aktualisierende Streaming-Quelle ist, stellt er einen aktualisierten Amazon-Ressourcennamen (ARN) und IAM-Rollen-ARN des Streams bereit.

Typ: KinesisStreamsInputUpdate Objekt

Erforderlich: Nein

NamePrefixUpdate

Namenspräfix für In-App-Streams, die Amazon-Kinesis-Analytics für die spezifische Streaming-Quelle erstellt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

JSONMappingParameters

Gibt zusätzliche Zuordnungsinformationen an, wenn das Aufzeichnungsformat JSON für die Streaming-Quelle verwendet wird.

Inhalt

RecordRowPath

Pfad zur obersten übergeordneten Ebene, die die Datensätze enthält.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInput

Identifiziert einen Amazon Kinesis Firehose-Delivery-Stream als Streaming-Quelle. Sie stellen den Amazon Ressourcennamen (ARN) des Delivery-Streams und einen IAM-Rollen-ARN bereit, mit dem Amazon Kinesis Analytics in Ihrem Namen auf den Stream zugreifen kann.

Inhalt

ResourceARN

Der ARN des Eingabe-Delivery-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen. Sie müssen sicherstellen, dass die Rolle über die erforderlichen Berechtigungen für den Zugriff auf den Stream verfügt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInputDescription

Beschreibt den Amazon-Kinesis-Firehose-Bereitstellungs-Stream, der in der Anwendungseingabekonfiguration als Streaming-Quelle konfiguriert ist.

Inhalt

ResourceARN

Amazon-Ressourcennamen (ARN) des Amazon-Kinesis-Firehose-Bereitstellungs-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics annimmt, um auf den Stream zuzugreifen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInputUpdate

Stellt beim Aktualisieren der Anwendungseingabekonfiguration Informationen über einen Amazon-Kinesis-Firehose-Bereitstellungs-Stream als Streaming-Quelle bereit.

Inhalt

ResourceARNUpdate

Amazon-Ressourcennamen (ARN) des Amazon-Kinesis-Firehose-Bereitstellungs-Streams, der gelesen werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisFirehoseOutput

Identifiziert beim Konfigurieren der Anwendungsausgabe einen Amazon Kinesis Firehose-Delivery-Stream als Ziel. Sie stellen den Amazon Ressourcennamen (ARN) des Streams und eine IAM-Rolle bereit, mit der Amazon Kinesis Analytics in Ihrem Namen in den Stream schreiben kann.

Inhalt

ResourceARN

Der ARN des Amazon Kinesis Firehose-Ziel-Delivery-Streams, in den geschrieben werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

Der ARN der IAM-Rolle, den Amazon Kinesis Analytics verwenden kann, um für Sie in den Ziel-Stream zu schreiben. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisFirehoseOutputDescription

Beschreibt für eine Anwendungsausgabe einen Amazon-Kinesis-Firehose-Bereitstellungs-Stream, der als ihr Ziel konfiguriert ist.

Inhalt

ResourceARN

Amazon-Ressourcennamen (ARN) des Amazon-Kinesis-Firehose-Bereitstellungs-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics annehmen kann, um für Sie auf den Stream zuzugreifen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisFirehoseOutputUpdate

Stellt beim Aktualisieren einer Ausgabekonfiguration mithilfe des UpdateApplicationVorgangs Informationen über einen Amazon Kinesis Firehose-Lieferstream bereit, der als Ziel konfiguriert ist.

Inhalt

ResourceARNUpdate

Der Amazon-Ressourcennamen (ARN) des Amazon-Kinesis–Bereitstellungs-Streams, in den geschrieben werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisStreamsInput

Identifiziert einen Amazon Kinesis-Stream als Streaming-Quelle. Sie stellen den Amazon Ressourcennamen (ARN) des Streams und einen IAM-Rollen-ARN bereit, mit dem Amazon Kinesis Analytics in Ihrem Namen auf den Stream zugreifen kann.

Inhalt

ResourceARN

Der ARN des zu lesenden Amazon Kinesis-Eingabe-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisStreamsInputDescription

Beschreibt den Amazon-Kinesis-Stream, der in der Anwendungseingabekonfiguration als Streaming-Quelle konfiguriert ist.

Inhalt

ResourceARN

Der Amazon-Ressourcenname (ARN) des Amazon-Kinesis-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics annehmen kann, um für Sie auf den Stream zuzugreifen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisStreamsInputUpdate

Stellt beim Aktualisieren der Anwendungseingabekonfiguration Informationen über einen Amazon-Kinesis-Stream als Streaming-Quelle bereit.

Inhalt

ResourceARNUpdate

Der Amazon-Ressourcenname (ARN) des zu lesenden Amazon-Kinesis-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisStreamsOutput

Identifiziert beim Konfigurieren der Anwendungsausgabe einen Amazon Kinesis-Stream als Ziel. Sie stellen den Amazon Ressourcennamen (ARN) des Streams sowie einen IAM-Rollen-ARN bereit, den Amazon Kinesis Analytics verwendet kann, um in Ihrem Namen in den Stream zu schreiben.

Inhalt

ResourceARN

Der ARN des Amazon Kinesis-Ziel-Streams, in den geschrieben werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

Der ARN der IAM-Rolle, den Amazon Kinesis Analytics verwenden kann, um für Sie in den Ziel-Stream zu schreiben. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisStreamsOutputDescription

Beschreibt für eine Anwendungsausgabe den Amazon-Kinesis-Stream, der als Ziel konfiguriert ist.

Inhalt

ResourceARN

Der Amazon-Ressourcenname (ARN) des Amazon-Kinesis-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics annehmen kann, um für Sie auf den Stream zuzugreifen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutputUpdate

Stellt beim Aktualisieren einer Ausgabekonfiguration mithilfe des UpdateApplicationVorgangs Informationen über einen Amazon Kinesis Kinesis-Stream bereit, der als Ziel konfiguriert ist.

Inhalt

ResourceARNUpdate

Der Amazon-Ressourcenname (ARN) des Amazon-Kinesis-Streams, in den geschrieben werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie auf den Stream zuzugreifen. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

LambdaOutput

Identifiziert bei der Konfiguration der Anwendungsausgabe eine AWS Lambda-Funktion als Ziel. Sie stellen den Amazon Ressourcennamen (ARN) der Funktion sowie einen IAM-Rollen-ARN bereit, den Amazon Kinesis Analytics verwendet kann, um in Ihrem Namen in die Funktion zu schreiben.

Inhalt

ResourceARN

Der Amazon-Ressourcenname (ARN) der Lambda-Zielfunktion, in die geschrieben werden soll.

> **Note**
>
> Zum Angeben einer früheren Version der Lambda-Funktion als die neueste, fügen Sie die Version der Lambda-Funktion im ARN der Lambda-Funktion ein. Weitere Informationen zu Lambda finden Sie unter ARNs Beispiel ARNs: AWS Lambda

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie in die Zielfunktion zu schreiben. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutputDescription

Beschreibt für eine Anwendungsausgabe die AWS Lambda-Funktion, die als Ziel konfiguriert ist.

Inhalt

ResourceARN

Der Amazon-Ressourcenname (ARN) der Lambda-Zielfunktion.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie in die Zielfunktion zu schreiben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutputUpdate

Stellt beim Aktualisieren einer Ausgabekonfiguration mithilfe des UpdateApplicationVorgangs Informationen zu einer AWS Lambda-Funktion bereit, die als Ziel konfiguriert ist.

Inhalt

ResourceARNUpdate

Der Amazon-Ressourcenname (ARN) der Lambda-Zielfunktion.

> **Note**
>
> Zum Angeben einer früheren Version der Lambda-Funktion als die neueste, fügen Sie die Version der Lambda-Funktion im ARN der Lambda-Funktion ein. Weitere Informationen zu Lambda finden Sie unter ARNs Beispiel ARNs: AWS Lambda

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

RoleARNUpdate

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics verwenden kann, um für Sie in die Zielfunktion zu schreiben. Sie müssen die erforderlichen Berechtigungen für diese Rolle erteilen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MappingParameters

Wenn Sie die Eingaben für die Anwendungen beim Erstellen oder Aktualisieren einer Anwendung konfigurieren, werden auf der Streaming-Quelle zusätzliche Zuordnungsinformationen bereitgestellt, die spezifisch für das Datensatz-Format sind (z. B. JSON, CSV, oder durch ein Trennzeichen abgetrennte Datensatzfelder).

Inhalt

CSVMappingParameters

Bietet zusätzliche Zuordnungsinformationen, wenn das Datensatzformat Trennzeichen verwendet (z. B. CSV).

Typ: CSVMappingParameters Objekt

Erforderlich: Nein

JSONMappingParameters

Gibt zusätzliche Zuordnungsinformationen an, wenn das Aufzeichnungsformat JSON für die Streaming-Quelle verwendet wird.

Typ: JSONMappingParameters Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Output

Beschreibt die Anwendungsausgabekonfiguration, in der Sie einen In-Application-Stream und ein Ziel identifizieren, in das die In-Application-Stream-Daten geschrieben werden sollen. Das Ziel kann ein Amazon-Kinesis-Stream oder ein Amazon-Kinesis-Firehose-Delivery-Stream sein.

Informationen zu Limits für die Anzahl der Ziele, die eine Anwendung schreiben kann, und andere Einschränkungen finden Sie unter Limits.

Inhalt

DestinationSchema

Beschreibt das Datenformat, wenn Datensätze in das Ziel geschrieben werden. Weitere Informationen finden Sie unter Konfigurieren der Anwendungsausgabe.

Typ: DestinationSchema Objekt

Erforderlich: Ja

Name

Der Name des In-Application-Streams.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Ja

KinesisFirehoseOutput

Identifiziert einen Amazon Kinesis Firehose-Delivery-Stream als Ziel.

Typ: KinesisFirehoseOutput Objekt

Erforderlich: Nein

KinesisStreamsOutput

Identifiziert einen Amazon Kinesis-Stream als Ziel.

Typ: KinesisStreamsOutput Objekt

Erforderlich: Nein

LambdaOutput

Identifiziert eine AWS Lambda-Funktion als Ziel.

Typ: LambdaOutput Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputDescription

Beschreibt die Ausgabekonfiguration der Anwendung, die den Namen des In-App-Streams und das Ziel umfasst, in das die Stream-Daten geschrieben werden. Das Ziel kann ein Amazon-Kinesis-Stream oder ein Amazon-Kinesis-Firehose-Delivery-Stream sein.

Inhalt

DestinationSchema

Datenformat, das zum Schreiben von Daten in das Ziel verwendet wird.

Typ: DestinationSchema Objekt

Erforderlich: Nein

KinesisFirehoseOutputDescription

Beschreibt den Amazon-Kinesis-Firehose-Bereitstellungs-Stream, der als Ziel konfiguriert ist, in das die Ausgabe geschrieben wird.

Typ: KinesisFirehoseOutputDescription Objekt

Erforderlich: Nein

KinesisStreamsOutputDescription

Beschreibt den Amazon-Kinesis-Stream, der als Ziel konfiguriert ist, in das die Ausgabe geschrieben wird.

Typ: KinesisStreamsOutputDescription Objekt

Erforderlich: Nein

LambdaOutputDescription

Beschreibt die AWS Lambda-Funktion, die als Ziel konfiguriert ist, in das die Ausgabe geschrieben wird.

Typ: LambdaOutputDescription Objekt

Erforderlich: Nein

Name

Name des In-Application-Streams, der als Ausgabe konfiguriert ist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Nein

OutputId

Eindeutige Kennung für eine Ausgabekonfiguration.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputUpdate

Beschreibt Aktualisierungen der Ausgabekonfiguration, identifiziert durch die `OutputId`.

Inhalt

OutputId

Identifiziert die spezifische Ausgabekonfiguration, die Sie aktualisieren möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

DestinationSchemaUpdate

Beschreibt das Datenformat, wenn Datensätze in das Ziel geschrieben werden. Weitere Informationen finden Sie unter Konfigurieren der Anwendungsausgabe.

Typ: DestinationSchema Objekt

Erforderlich: Nein

KinesisFirehoseOutputUpdate

Beschreibt einen Amazon-Kinesis-Firehose-Bereitstellungs-Stream als Ziel für die Ausgabe.

Typ: KinesisFirehoseOutputUpdate Objekt

Erforderlich: Nein

KinesisStreamsOutputUpdate

Beschreibt einen Amazon-Kinesis-Stream als Ziel für die Ausgabe.

Typ: KinesisStreamsOutputUpdate Objekt

Erforderlich: Nein

LambdaOutputUpdate

Beschreibt eine AWS Lambda-Funktion als Ziel für die Ausgabe.

Typ: LambdaOutputUpdate Objekt

Erforderlich: Nein

NameUpdate

Wenn Sie für diese Ausgabekonfiguration einen anderen In-Application-Stream angeben möchten, verwenden Sie dieses Feld, um den Namen des neuen In-Application-Streams anzugeben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecordColumn

Beschreibt die Zuordnung jedes Datenelements in der Streaming-Quelle zur entsprechenden Spalte im In-Application-Stream.

Wird auch zum Beschreiben des Formats der Referenzdatenquelle verwendet.

Inhalt

Name

Der Name der Spalte, die im Eingabe-Stream der In-Application oder in der Referenztabelle erstellt wird.

Typ: Zeichenfolge

Erforderlich: Ja

SqlType

Der Typ der Spalte, die im Eingabe-Stream der In-Application oder in der Referenztabelle erstellt wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Ja

Mapping

Referenz zum Datenelement in der Streaming-Eingabe oder der Referenzdatenquelle. Dieses Element ist erforderlich, wenn dies der RecordFormatTypeFall ist`JSON`.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecordFormat

Beschreibt das Datensatzformat und die relevanten Zuordnungsinformationen, die angewendet werden sollten, um die Datensätze im Stream schematisch darzustellen.

Inhalt

RecordFormatType

Der Typ des Datensatzformats.

Typ: Zeichenfolge

Zulässige Werte: `JSON | CSV`

Erforderlich: Ja

MappingParameters

Wenn Sie die Eingaben für die Anwendungen beim Erstellen oder Aktualisieren einer Anwendung konfigurieren, werden auf der Streaming-Quelle zusätzliche Zuordnungsinformationen bereitgestellt, die spezifisch für das Datensatz-Format sind (z. B. JSON, CSV, oder durch ein Trennzeichen abgetrennte Datensatzfelder).

Typ: MappingParameters Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSource

Beschreibt die Referenzdatenquelle, indem die Quellinformation (S3-Bucket-Name und Objektschlüsselname), der resultierende Name der erstellten In-Application-Tabelle und das erforderliche Schema zum Zuordnen der Datenelemente im Amazon S3-Objekt zur In-Application-Tabelle bereitgestellt werden.

Inhalt

ReferenceSchema

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden im In-Application-Stream erzeugten Spalten zugeordnet wird.

Typ: SourceSchema Objekt

Erforderlich: Ja

TableName

Der Name der zu erstellenden In-Application-Tabelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Ja

S3ReferenceDataSource

Identifiziert den S3-Bucket und das Objekt mit den Referenzdaten. Identifiziert außerdem die IAM-Rolle, die Amazon Kinesis Analytics annehmen kann, um dieses Objekt für Sie zu lesen. Eine Amazon Kinesis Analytics-Anwendung lädt Referenzdaten nur einmal. Wenn sich die Daten ändern, rufen Sie die `UpdateApplication`-Operation auf, um das erneute Laden von Daten in Ihre Anwendung auszulösen.

Typ: S3ReferenceDataSource Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSourceDescription

Beschreibt die für eine Anwendung konfigurierte Referenzdatenquelle.

Inhalt

ReferenceId

Die ID der Referenzdatenquelle. Dies ist die ID, die Amazon Kinesis Analytics zuweist, wenn Sie Ihrer Anwendung mithilfe des Vorgangs die Referenzdatenquelle hinzufügen. AddApplicationReferenceDataSource

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

S3ReferenceDataSourceDescription

Stellt den S3-Bucket-Namen und den Objektschlüsselnamen bereit, die die Referenzdaten enthalten. Es stellt auch den Amazon-Ressourcennamen (ARN) der IAM-Rolle bereit, die Amazon-Kinesis-Analytics annehmen kann, um das Amazon-S3-Objekt zu lesen und die Referenztabelle in der Anwendung zu füllen.

Typ: S3ReferenceDataSourceDescription Objekt

Erforderlich: Ja

TableName

Der Name der In-App-Tabelle, die durch die spezifische Konfiguration der Referenzdatenquelle erstellt wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Ja

ReferenceSchema

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden im In-Application-Stream erzeugten Spalten zugeordnet wird.

Typ: SourceSchema Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSourceUpdate

Wenn Sie eine Referenzdatenquellenkonfiguration für eine Anwendung aktualisieren, stellt dieses Objekt alle aktualisierten Werte (wie den Quell-Bucket-Namen und Objektschlüsselnamen), den Namen der erstellten In-Application-Tabelle und aktualisierte Zuordnungsinformation zum Zuordnen der Daten im Amazon-S3-Objekt zur In-Application-Tabelle bereit, die erstellt wird.

Inhalt

ReferenceId

ID der Referenzdatenquelle, die aktualisiert wird. Sie können den Vorgang DescribeApplication verwenden, um diesen Wert abzurufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge = 50 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

ReferenceSchemaUpdate

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden im In-Application-Stream erzeugten Spalten zugeordnet wird.

Typ: SourceSchema Objekt

Erforderlich: Nein

S3ReferenceDataSourceUpdate

Beschreibt den S3-Bucket-Namen, den Objektschlüsselnamen und die IAM-Rolle, die Amazon-Kinesis-Analytics annehmen kann, um das Amazon-S3-Objekt in Ihrem Namen zu lesen und die In-App-Referenztabelle zu füllen.

Typ: S3ReferenceDataSourceUpdate Objekt

Erforderlich: Nein

TableNameUpdate

Name der In-App-Tabelle, die durch dieses Update erstellt wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3Configuration

Bietet eine Beschreibung einer Amazon-S3-Datenquelle, einschließlich des Amazon-Ressourcennamen (ARN) des S3-Buckets, des ARN der IAM-Rolle, die für den Zugriff auf den Bucket verwendet wird, und des Namens des Amazon-S3-Objekts, das die Daten enthält.

Inhalt

BucketARN

ARN des S3-Buckets, der die Daten enthält.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

FileKey

Der Name des Objekts, das die Daten enthält.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge von 1 024.

Erforderlich: Ja

RoleARN

IAM-ARN der Rolle, die für den Zugriff auf die Daten verwendet wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSource

Identifiziert den S3-Bucket und das Objekt mit den Referenzdaten. Identifiziert außerdem die IAM-Rolle, die Amazon Kinesis Analytics annehmen kann, um dieses Objekt für Sie zu lesen.

Eine Amazon Kinesis Analytics-Anwendung lädt Referenzdaten nur einmal. Wenn sich die Daten ändern, rufen Sie die UpdateApplication-Operation auf, um das erneute Laden von Daten in Ihre Anwendung auszulösen.

Inhalt

BucketARN

Der Amazon-Ressourcenname (ARN) des S3-Buckets.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

FileKey

Der Objektschlüsselname, der Referenzdaten enthält.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge von 1 024.

Erforderlich: Ja

ReferenceRoleARN

Der ARN der IAM-Rolle, die der Service verwenden kann, um Daten für Sie zu lesen. Diese Rolle muss über die Berechtigung für die `s3:GetObject`-Aktion für das Objekt und die Vertrauensrichtlinie verfügen, mit der Amazon Kinesis Analytics-Service-Prinzipal diese Rolle übernehmen kann.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSourceDescription

Stellt den Bucket-Namen und Objektschlüsselnamen bereit, der die Referenzdaten speichert.

Inhalt

BucketARN

Der Amazon-Ressourcenname (ARN) des S3-Buckets.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

FileKey

Schlüsselname des Amazon-S3-Objekts.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge von 1 024.

Erforderlich: Ja

ReferenceRoleARN

Der ARN der IAM-Rolle, den Amazon-Kinesis-Analytics annehmen kann, um das Amazon-S3-Objekt in Ihrem Namen zu lesen und die In-App-Referenztabelle zu füllen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSourceUpdate

Beschreibt den S3-Bucket-Namen, den Objektschlüsselnamen und die IAM-Rolle, die Amazon-Kinesis-Analytics annehmen kann, um das Amazon-S3-Objekt in Ihrem Namen zu lesen und die In-App-Referenztabelle zu füllen.

Inhalt

BucketARNUpdate

Der Amazon-Ressourcenname (ARN) des S3-Buckets.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

FileKeyUpdate

Objektschlüsselname.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 1024 Zeichen.

Erforderlich: Nein

ReferenceRoleARNUpdate

Der ARN der IAM-Rolle, die Amazon-Kinesis-Analytics annehmen kann, um das Amazon-S3-Objekt zu lesen und die In-App-Referenztabelle zu füllen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:.*`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SourceSchema

Beschreibt das Format der Daten in der Streaming-Quelle und wie jedes Datenelement den entsprechenden im In-Application-Stream erzeugten Spalten zugeordnet wird.

Inhalt

RecordColumns

Eine Liste von `RecordColumn`-Objekten.

Typ: Array von RecordColumn-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 1000 Elemente.

Erforderlich: Ja

RecordFormat

Gibt das Format der Datensätze in der Streaming-Quelle an.

Typ: RecordFormat Objekt

Erforderlich: Ja

RecordEncoding

Gibt die Codierung der Datensätze in der Streaming-Quelle an. Zum Beispiel UTF-8.

Typ: Zeichenfolge

Pattern: `UTF-8`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

Tag

Ein Schlüssel-Wert-Paar (der Wert ist optional), das Sie definieren und Ressourcen zuweisen AWS können. Wenn Sie ein Tag spezifizieren, das bereits existiert, wird der Tag-Wert durch den Wert ersetzt, den Sie in der Anfrage angeben. Beachten Sie, dass die maximale Anzahl von Anwendungs-Tags System-Tags einschließt. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50. Weitere Informationen über das Markieren mit Tags finden Sie unter Einsatz von Tags.

Inhalt

Key

Der Schlüssel des Schlüssel-Wert-Tags.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Value

Der Wert des Schlüssel-Wert-Tags. Der -Wert ist optional.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Dokumentenverlauf für Amazon-Kinesis-Data-Analytics

Die folgende Tabelle beschreibt die wichtigsten Änderungen in der Dokumentation seit der letzten Version von Amazon-Kinesis-Data-Analytics.

- API-Version: 2015-08-14
- Letzte Aktualisierung der Dokumentation: 8. Mai 2019

Änderung	Beschreibung	Datum
Kinesis Data Analytics-Anwendungen mit Tags versehen	Verwenden Sie Anwendungs-Tagging zum Bestimmen der Kosten pro Anwendung, zur Kontrolle des Zugriffs oder für benutzerdefinierte Zwecke. Weitere Informationen finden Sie unter Verwenden von Tagging.	8. Mai 2019
Protokollieren von Kinesis Data Analytics Analytics-API-Aufrufen mit AWS CloudTrail	Amazon Kinesis Data Analytics ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Kinesis Data Analytics ausgeführt wurden. Weitere Informationen finden Sie unter Benutzen AWS CloudTrail.	22. März 2019
Kinesis Data Analytics ist in der Region Frankfurt verfügbar	Kinesis Analytics ist jetzt in der Region Europa (Frankfurt) verfügbar. Weitere Informationen finden Sie unter Kontingente und Endpunkte	18. Juli 2018

Änderung	Beschreibung	Datum
	von AWS: Kinesis Data Analytics.	
Verwenden von Referenzdaten in der Konsole	Sie können nun mit Anwendungsreferenzdaten in der Konsole arbeiten. Weitere Informationen finden Sie unter Beispiel: Hinzufügen von Referenzdaten zu einer Kinesis Data Analytics-Anwendung.	13. Juli 2018
Beispiele für fensterbasierte Abfragen	Beispielanwendungen für Fenster und Aggregation. Weitere Informationen finden Sie unter Beispiele: Fenster und Aggregation.	9. Juli 2018
Testen von Anwendungen	Anleitung zum Testen von Änderungen an Anwendungsschemata und Code. Weitere Informationen finden Sie unter Testen von Anwendungen.	3. Juli 2018
Beispielanwendungen für die Vorverarbeitung von Daten	Zusätzliche Codebeispiele für REGEX_LOG_PARSE, REGEX_REPLACE und Operatoren. DateTime Weitere Informationen finden Sie unter Beispiele: Umwandeln von Daten.	18. Mai 2018

Änderung	Beschreibung	Datum
Heraufsetzen der Größe für die zurückgegebenen Zeilen und den SQL-Code	Die Obergrenze für die Größe einer zurückgegebenen Zeile wird auf 512 KB, die Obergrenze für die Größe des SQL-Codes in einer Anwendung auf 100 KB erhöht. Weitere Informationen finden Sie unter Einschränkungen.	2. Mai 2018
AWS Lambda Funktionsbeispiele in Java und .NET	Codebeispiele für die Erstellung von Lambda-Funktionen für die Vorverarbeitung von Datensätzen und für Anwendungsziele. Weitere Informationen erhalten Sie unter Erstellen von Lambda-Funktionen für die Vorverarbeitung und Erstellen von Lambda-Funktionen für Anwendungsziele.	22. März 2018
Neue HOTSPOTS-Funktion	Suchen und Zurückgeben von Informationen über relativ dichte Bereiche in Ihren Daten. Weitere Informationen finden Sie unter Beispiel: Erkennen von Hotspots in einem Stream (HOTSPOTS-Funktion).	19. März 2018

Änderung	Beschreibung	Datum
Lambda-Funktion als ein Ziel	Senden von Analyseer gebnissen zu einer Lambda-Funktion als Ziel. Weitere Informationen finden Sie unter Verwenden einer Lambda-Fu nktion als Ausgabe.	20. Dezember 2017
Neue `RANDOM_CU T_FOREST_WITH_EXPL ANATION` -Funktion	Anfordern einer Erläuteru ng dazu, welche Felder zu einer Anomaliebewertung in einem Daten-Stream beitragen . Weitere Informationen finden Sie unter Beispiel: Erkennen von Datenanomalien und Suchen nach einer Erklärung (Funktion RANDOM_CU T_FOREST_WITH_EXPL ANATION).	2. November 2017
Schema-Erkennung für statische Daten	Ausführen der Schema-Er kennung für statische Daten in einem Amazon-S3-Bucket. Weitere Informationen finden Sie unter Verwenden der Funktion der Schemaerk ennung für statische Daten.	6. Oktober 2017
Lambda-Vorverarbeitungsfunk tion	Datensätze in einem Eingabestream mit AWS Lambda vor der Analyse vorverarbeiten. Weitere Informationen finden Sie unter Vorverarbeitung von Daten mithilfe einer Lambda-Fu nktion.	6. Oktober 2017

Änderung	Beschreibung	Datum
Auto Scaling von Anwendungen	Automatisches Erhöhen des Datendurchsatzes Ihrer Anwendung mit automatischer Skalierung. Weitere Informationen finden Sie unter Automatisches Skalieren von Anwendungen zur Erhöhung des Durchsatzes.	13. September 2017
Mehrere In-Application-Eingabe-Streams	Erhöhen des Durchsatzes mit mehreren In-Applications-Streams. Weitere Informationen finden Sie unter Parallelisieren von Eingabe-Streams zur Steigerung des Durchsatzes.	29. Juni 2017
Leitfaden zur Verwendung von AWS-Managementkonsole for Kinesis Data Analytics	Bearbeiten eines abgeleiteten Schemas und von SQL-Code mithilfe des Schema-Editors und des SQL-Editors in der Kinesis Data Analytics -Konsole. Weitere Informationen finden Sie unter Schritt 4 (optional): Bearbeiten des Schema- und SQL-Codes mithilfe der Konsole.	7. April 2017
Öffentliche Freigabe	Veröffentlichung des Amazon-Kinesis-Data-Analytics-Entwicklerhandbuches.	11. August 2016
Vorversion	Vorabversion des Amazon-Kinesis-Data-Analytics-Entwicklerhandbuches.	29. Januar 2016

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.