Panduan Developer SQL

Panduan Developer Amazon Kinesis Data Analytics untuk Aplikasi SQL

Panduan Developer Amazon Kinesis Data Analytics untuk Aplikasi SQL: Panduan Developer SQL

Table of Contents

..... x
Apa Itu Amazon Kinesis Data Analytics untuk Aplikasi SQL? 1
Kapan Saya Harus Menggunakan Amazon Kinesis Data Analytics? 1
Apakah Anda Pengguna Baru Amazon Kinesis Data Analytics? 1
Cara Kerjanya 3
Input 6
Mengonfigurasi Sumber Streaming 7
Mengonfigurasi Sumber Referensi 10
Bekerja dengan JSONPath 13
Memetakan Elemen Sumber Streaming ke Kolom Input SQL 18
Menggunakan Fitur Penemuan Skema pada Data Streaming 24
Menggunakan Fitur Penemuan Skema pada Data Statis 26
Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda 31
Menyejajarkan Aliran Input untuk Peningkatan Throughput 42
Kode Aplikasi 47
Output 49
Membuat Output Menggunakan AWS CLI 50
Menggunakan Fungsi Lambda sebagai Output 52
Model Pengiriman Output Aplikasi 61
Penanganan Kesalahan 61
Melaporkan Kesalahan Menggunakan Aliran Kesalahan dalam Aplikasi 62
Menskalakan Otomatis Aplikasi 63
Penandaan 63
Menambahkan Tanda saat Aplikasi dibuat 64
Menambahkan atau Memperbarui Tanda untuk Aplikasi yang Ada 65
Mencantumkan Tanda untuk Aplikasi 65
Menghapus Tanda dari Aplikasi 66
Memulai 67
Mendaftar untuk Akun AWS 67
Buat pengguna dengan akses administratif 68
Langkah 1: Siapkan akun 69
Mendaftar untuk AWS 69
Membuat Pengguna IAM 70
Langkah Selanjutnya 71

Mendaftar untuk Akun AWS 67
Buat pengguna dengan akses administratif 68
Langkah 2: Mengatur AWS CLI 73
Langkah Selanjutnya 74
Langkah 3: Buat Aplikasi Analitik Starter Anda 74
Langkah 3.1: Buat Aplikasi 77
Langkah 3.2: Konfigurasikan Input 79
Langkah 3.3: Tambahkan Analitik Waktu Nyata (Tambahkan Kode Aplikasi) 82
Langkah 3.4: (Opsional) Perbarui Kode Aplikasi 86
Langkah 4 (Opsional) Edit Skema dan Kode SQL Menggunakan Konsol 88
Menggunakan Editor Skema 89
Menggunakan Editor SQL 98
Konsep SQL Streaming 102
Aliran dan Pompa dalam Aplikasi 102
Stempel waktu dan Kolom ROWTIME 104
Memahami Berbagai Waktu dalam Analitik Streaming 104
Kueri Berkelanjutan 107
Kueri Jendela 108
Jendela Stagger 109
Jendela Tumbling 114
Jendela Geser 116
Gabungan Streaming 121
Contoh 1: Laporkan Pesanan Jika Ada Perdagangan dalam Satu Menit Setelah Pesanan Ditempatkan 122
Migrasi ke Managed Service untuk Apache Flink 124
Mereplikasi Kinesis Data Analytics untuk Kueri SQL di Managed Service untuk Apache Flink Studio 124
Membuat ulang Kinesis Data Analytics untuk kueri SQL di Managed Service untuk Apache Flink Studio 125
Migrasi beban kerja Random Cut Forest 154
Mengganti Kinesis Data Firehose sebagai sumber dengan Kinesis Data Streams 155
Analisis Data Amazon Kinesis - SQL dan Amazon Kinesis Data Firehose 155
Layanan Terkelola Amazon untuk Apache Flink Studio 158
Memanfaatkan fungsi yang ditentukan pengguna (UDF) 164
Fungsi yang ditentukan pengguna (UDF) 165
Pengaturan lingkungan 166

Bekerja dengan Layanan Terkelola untuk notebook Apache Flink Studio 167
Mempromosikan notebook sebagai Aplikasi 170
Pembersihan 171
Kinesis Data Analytics untuk contoh SQL 172
Mengubah Data 172
Memproses Aliran Terlebih Dulu dengan Lambda 172
Mengubah Nilai String 173
Mengubah Nilai DateTime 194
Mengubah Beberapa Tipe Data 198
Jendela dan Agregasi 207
Jendela Stagger 207
Jendela Tumbling Menggunakan ROWTIME 211
Jendela Tumbling Menggunakan Stempel Waktu Peristiwa 215
Nilai yang Paling Sering Terjadi (TOP_K_ITEMS_TUMBLING) 219
Menggabungkan Hasil Parsial 222
Bergabung 225
Contoh: Tambahkan Sumber Data Referensi 226
Machine Learning 230
Mendeteksi Anomali 230
Contoh: Deteksi Anomali dan Dapatkan Penjelasan 239
Contoh: Deteksi Hotspot 245
Peringatan dan Kesalahan 258
Peringatan Sederhana 259
Peringatan Terbatas 260
Aliran Kesalahan dalam Aplikasi 262
Akselerator Solusi 264
Wawasan waktu nyata tentang aktivitas Akun AWS 264
Pemantauan perangkat AWS IoT secara langsung dengan Kinesis Data Analytics 264
Analitik web waktu nyata dengan Kinesis Data Analytics 264
Solusi Kendaraan Terhubung Amazon 265
Keamanan 266
Perlindungan Data 267
Enkripsi data 267
Identity and Access Management 268
Kebijakan Kepercayaan 268
Kebijakan Izin 269

Pencegahan confused deputy lintas layanan 272
Autentikasi dan Kontrol Akses 274
Kontrol Akses 274
Mengautentikasi dengan identitas 275
Gambaran Umum Pengelolaan Akses 278
Menggunakan Kebijakan Berbasis Identitas (Kebijakan IAM) 284
Referensi Izin API 292
Pemantauan 293
Validasi Kepatuhan 293
Ketangguhan 294
Pemulihan Bencana 294
Keamanan Infrastruktur 294
Praktik Terbaik Keamanan 295
Gunakan IAM role untuk mengakses layanan Amazon lainnya 295
Terapkan Enkripsi Sisi Server di Sumber Daya Dependen 296
Gunakan CloudTrail untuk Memantau Panggilan API 296
Pemantauan 297
Alat Pemantauan 298
Alat Otomatis 298
Alat Manual 299
Pemantauan CloudWatch dengan Amazon 299
Metrik dan Dimensi 300
Melihat Metrik dan Dimensi 302
Alarm 303
Log 304
Menggunakan AWS CloudTrail 311
Informasi di CloudTrail 311
Memahami Entri File Berkas Log 312
Batas 315
Praktik Terbaik 318
Mengelola Aplikasi 318
Menskalakan Aplikasi 319
Aplikasi Pemantauan 320
Mendefinisikan Skema Input 321
Menyambung ke Output 322
Menulis Kode Aplikasi 322

Menguji Aplikasi ... 323
Menyiapkan Aplikasi Uji ... 323
Menguji Perubahan Skema ... 324
Menguji Perubahan Kode ... 324
Pemecahan Masalah ... 325
Aplikasi yang dihentikan ... 325
Tidak Dapat Menjalankan Kode SQL ... 326
Tidak Dapat Mendeteksi atau Menemukan Skema Saya ... 326
Data Referensi Kedaluwarsa ... 327
Aplikasi Tidak Menulis ke Tujuan ... 327
Parameter Kondisi Aplikasi Penting untuk Dipantau ... 328
Kesalahan Kode Tidak Valid Saat Menjalankan Aplikasi ... 328
Aplikasi Menulis Kesalahan ke Aliran Kesalahan ... 329
Throughput Tidak Cukup atau Tinggi MillisBehindLatest ... 329
Referensi SQL ... 331
Referensi API ... 332
Tindakan ... 332
AddApplicationCloudWatchLoggingOption ... 334
AddApplicationInput ... 337
AddApplicationInputProcessingConfiguration ... 341
AddApplicationOutput ... 345
AddApplicationReferenceDataSource ... 349
CreateApplication ... 353
DeleteApplication ... 361
DeleteApplicationCloudWatchLoggingOption ... 364
DeleteApplicationInputProcessingConfiguration ... 367
DeleteApplicationOutput ... 370
DeleteApplicationReferenceDataSource ... 373
DescribeApplication ... 376
DiscoverInputSchema ... 381
ListApplications ... 387
ListTagsForResource ... 390
StartApplication ... 393
StopApplication ... 396
TagResource ... 398
UntagResource ... 401

UpdateApplication 404
Tipe Data 409
ApplicationDetail 412
ApplicationSummary 416
ApplicationUpdate 418
CloudWatchLoggingOption 420
CloudWatchLoggingOptionDescription 422
CloudWatchLoggingOptionUpdate 424
CSVMappingParameters 426
DestinationSchema 427
Input 428
InputConfiguration 431
InputDescription 432
InputLambdaProcessor 435
InputLambdaProcessorDescription 437
InputLambdaProcessorUpdate 438
InputParallelism 440
InputParallelismUpdate 441
InputProcessingConfiguration 442
InputProcessingConfigurationDescription 443
InputProcessingConfigurationUpdate 444
InputSchemaUpdate 445
InputStartingPositionConfiguration 447
InputUpdate 448
JSONMappingParameters 450
KinesisFirehoseInput 451
KinesisFirehoseInputDescription 453
KinesisFirehoseInputUpdate 454
KinesisFirehoseOutput 455
KinesisFirehoseOutputDescription 457
KinesisFirehoseOutputUpdate 458
KinesisStreamsInput 459
KinesisStreamsInputDescription 461
KinesisStreamsInputUpdate 462
KinesisStreamsOutput 463
KinesisStreamsOutputDescription 465

KinesisStreamsOutputUpdate 466
LambdaOutput 467
LambdaOutputDescription 469
LambdaOutputUpdate 470
MappingParameters 472
Output 473
OutputDescription 475
OutputUpdate 477
RecordColumn 479
RecordFormat 481
ReferenceDataSource 482
ReferenceDataSourceDescription 484
ReferenceDataSourceUpdate 486
S3Configuration 488
S3ReferenceDataSource 490
S3ReferenceDataSourceDescription 492
S3ReferenceDataSourceUpdate 494
SourceSchema 496
Tag 498
Riwayat Dokumen 499
AWSGlosarium 504

Untuk proyek baru, kami menyarankan Anda menggunakan Managed Service baru untuk Apache Flink Studio melalui Kinesis Data Analytics untuk Aplikasi SQL. Layanan Terkelola untuk Apache Flink Studio menggabungkan kemudahan penggunaan dengan kemampuan analitis tingkat lanjut, memungkinkan Anda membangun aplikasi pemrosesan aliran yang canggih dalam hitungan menit.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Apa Itu Amazon Kinesis Data Analytics untuk Aplikasi SQL?

Dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL, Anda dapat memproses dan menganalisis data streaming menggunakan SQL standar. Layanan ini memungkinkan Anda menulis dan menjalankan kode SQL yang kuat dengan cepat pada sumber streaming untuk melakukan analitik deret waktu, mengumpan dasbor waktu nyata, dan membuat metrik waktu nyata.

Untuk memulai Kinesis Data Analytics, Anda membuat aplikasi Kinesis Data Analytics yang terus membaca dan memproses data streaming. Layanan ini mendukung pengambilan data dari Amazon Kinesis Data Streams dan sumber streaming Amazon Data Firehose. Kemudian, Anda menulis kode SQL Anda menggunakan editor interaktif dan mengujinya dengan data streaming langsung. Anda juga dapat mengonfigurasi tujuan tempat Anda ingin Kinesis Data Analytics mengirimkan hasilnya.

Kinesis Data Analytics mendukung Amazon Data Firehose (Amazon S3, Amazon Redshift, Amazon Service, dan Splunk), dan OpenSearch AWS LambdaAmazon Kinesis Data Streams sebagai tujuan.

Kapan Saya Harus Menggunakan Amazon Kinesis Data Analytics?

Amazon Kinesis Data Analytics memungkinkan Anda menulis kode SQL yang terus membaca, memproses, dan menyimpan data dalam waktu dekat secara langsung dengan cepat. Menggunakan kueri SQL standar pada data streaming, Anda dapat membangun aplikasi yang mengubah dan memberikan wawasan ke data Anda. Berikut adalah beberapa contoh skenario untuk menggunakan Kinesis Data Analytics:

- Buat analitik deret waktu – Anda dapat menghitung metrik dari jendela waktu, lalu mengalirkan nilai ke Amazon S3 atau Amazon Redshift melalui aliran pengiriman data Kinesis.
- Umpan dasbor waktu nyata – Anda dapat mengirim hasil data streaming yang dikumpulkan dan diproses ke hilir untuk mengumpan dasbor waktu nyata.
- Buat metrik waktu nyata – Anda dapat membuat metrik dan pemicu khusus untuk digunakan dalam pemantauan, notifikasi, dan alarm waktu nyata.

Untuk informasi tentang elemen bahasa SQL yang didukung oleh Kinesis Data Analytics, lihat Refererensi SQL Amazon Kinesis Data Analytics.

Apakah Anda Pengguna Baru Amazon Kinesis Data Analytics?

Jika Anda baru pertama kali menggunakan Amazon Kinesis Data Analytics, sebaiknya baca bagian-bagian berikut secara berurutan:

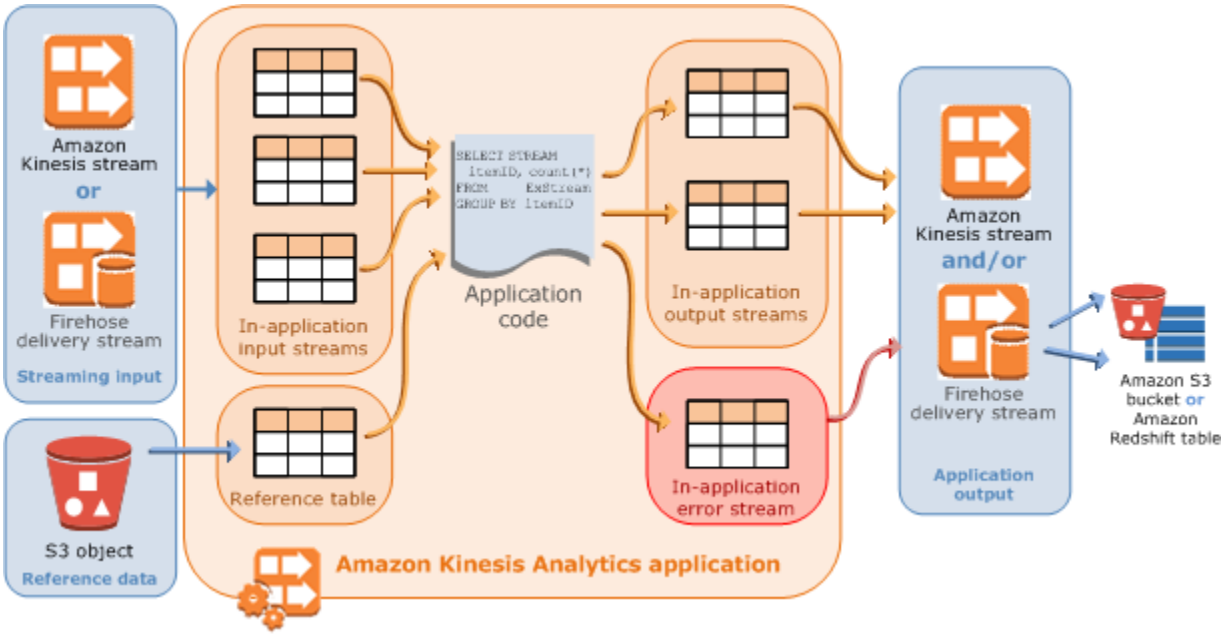
1. Baca bagian Cara Kerjanya dari panduan ini. Bagian ini memperkenalkan berbagai komponen Kinesis Data Analytics yang Anda gunakan untuk menciptakan end-to-end pengalaman. Untuk informasi selengkapnya, lihat Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya.
2. Cobalah latihan Memulai. Untuk informasi selengkapnya, lihat Memulai dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL.
3. Jelajahi konsep SQL streaming. Untuk informasi selengkapnya, lihat Konsep SQL Streaming.
4. Coba contoh tambahan. Lihat informasi yang lebih lengkap di Kinesis Data Analytics untuk contoh SQL.

Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

Aplikasi adalah sumber daya utama di Amazon Kinesis Data Analytics yang dapat Anda buat di akun Anda. Anda dapat membuat dan mengelola aplikasi menggunakan AWS Management Console atau Kinesis Data Analytics API. Kinesis Data Analytics menyediakan operasi API untuk mengelola aplikasi. Untuk daftar operasi API, lihat Tindakan.

Aplikasi Kinesis Data Analytics terus membaca dan memproses data streaming secara langsung. Anda menulis kode aplikasi menggunakan SQL untuk memproses data streaming yang masuk dan menghasilkan output. Kemudian, Kinesis Data Analytics menulis output ke tujuan yang dikonfigurasi. Diagram berikut menggambarkan arsitektur aplikasi yang khas.



Setiap aplikasi memiliki nama, deskripsi, ID versi, dan status. Amazon Kinesis Data Analytics menetapkan ID versi saat Anda pertama kali membuat aplikasi. ID versi ini diperbarui saat Anda memperbarui konfigurasi aplikasi. Misalnya, jika Anda menambahkan konfigurasi input, menambah

atau menghapus sumber data referensi, menambah atau menghapus konfigurasi output, atau memperbarui kode aplikasi, Kinesis Data Analytics akan memperbarui ID versi aplikasi saat ini. Kinesis Data Analytics juga mempertahankan stempel waktu saat aplikasi dibuat dan terakhir diperbarui.

Selain sifat dasar ini, setiap aplikasi terdiri dari berikut ini:

- Input – Sumber streaming untuk aplikasi Anda. Anda dapat memilih aliran data Kinesis atau aliran pengiriman data Firehose sebagai sumber streaming. Dalam konfigurasi input, Anda memetakan sumber streaming ke aliran input dalam aplikasi. Aliran dalam aplikasi seperti tabel yang terus diperbarui tempat Anda dapat melakukan operasi `SELECT` dan `INSERT SQL`. Dalam kode aplikasi Anda, Anda dapat membuat aliran dalam aplikasi tambahan untuk menyimpan hasil kueri menengah.

  Anda dapat secara opsional membuat partisi satu sumber streaming di beberapa aliran input dalam aplikasi untuk meningkatkan throughput. Lihat informasi yang lebih lengkap di Batas dan Mengonfigurasi Input Aplikasi.

  Amazon Kinesis Data Analytics menyediakan kolom stempel waktu di setiap aliran aplikasi yang disebut Stempel waktu dan Kolom ROWTIME. Anda dapat menggunakan kolom ini di kueri jendela berbasis waktu. Untuk informasi selengkapnya, lihat Kueri Jendela.

  Anda dapat secara opsional mengonfigurasi sumber data referensi untuk memperkaya aliran data input Anda dalam aplikasi. Ini menghasilkan tabel referensi dalam aplikasi. Anda harus menyimpan data referensi sebagai objek dalam bucket S3 Anda. Ketika aplikasi dimulai, Amazon Kinesis Data Analytics membaca objek Amazon S3 dan membuat tabel dalam aplikasi. Untuk informasi selengkapnya, lihat Mengonfigurasi Input Aplikasi.

- Kode aplikasi – Serangkaian pernyataan SQL yang memproses input dan menghasilkan output. Anda dapat menulis pernyataan SQL di aliran dalam aplikasi dan tabel referensi. Anda juga dapat menulis kueri JOIN untuk menggabungkan data dari kedua sumber ini.

Untuk informasi tentang elemen bahasa SQL yang didukung oleh Kinesis Data Analytics, lihat Refererensi SQL Amazon Kinesis Data Analytics.

Dalam bentuk yang paling sederhana, kode aplikasi dapat menjadi satu pernyataan SQL yang memilih dari input streaming dan memasukkan hasil ke output streaming. Ini juga bisa berupa serangkaian pernyataan SQL tempat output dari satu umpan ke input dari pernyataan SQL berikutnya. Selanjutnya, Anda dapat menulis kode aplikasi untuk membagi aliran input menjadi beberapa aliran. Anda selanjutnya dapat menerapkan kueri tambahan untuk memproses aliran ini. Untuk informasi selengkapnya, lihat Kode Aplikasi.

- Output – Dalam kode aplikasi, hasil kueri masuk ke aliran dalam aplikasi. Dalam kode aplikasi Anda, Anda dapat membuat satu atau beberapa aliran dalam aplikasi untuk menyimpan hasil menengah. Anda selanjutnya dapat secara opsional mengonfigurasi output aplikasi untuk menyimpan data di aliran dalam aplikasi yang menahan output aplikasi Anda (juga disebut sebagai aliran output dalam aplikasi) ke tujuan eksternal. Tujuan eksternal dapat berupa aliran pengiriman Firehose atau aliran data Kinesis. Perhatikan hal berikut mengenai tujuan ini:
  - Anda dapat mengonfigurasi aliran pengiriman Firehose untuk menulis hasil ke Amazon S3, Amazon Redshift, atau Amazon OpenSearch Service (Service). OpenSearch
  - Anda juga dapat menulis output aplikasi ke tujuan kustom bukan ke Amazon S3 atau Amazon Redshift. Untuk melakukannya, Anda menentukan Kinesis data stream sebagai tujuan dalam konfigurasi output Anda. Kemudian, Anda mengonfigurasi AWS Lambda untuk melakukan polling aliran dan menjalankan fungsi Lambda Anda. Kode fungsi Lambda Anda menerima data aliran sebagai input. Dalam kode fungsi Lambda Anda, Anda dapat menulis data yang masuk ke tujuan kustom Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Lambda dengan Amazon Kinesis Data Analytics.

  Untuk informasi selengkapnya, lihat Mengonfigurasi Output Aplikasi.

Selain itu, perhatikan hal berikut:

- Amazon Kinesis Data Analytics memerlukan izin untuk membaca catatan dari sumber streaming dan menulis output aplikasi ke tujuan eksternal. Anda menggunakan IAM role untuk memberikan izin ini.

- Kinesis Data Analytics secara otomatis menyediakan aliran kesalahan dalam aplikasi untuk setiap aplikasi. Jika aplikasi Anda memiliki masalah saat memproses catatan tertentu (misalnya, karena ketidakcocokan tipe atau terlambat datang), catatan tersebut ditulis ke aliran kesalahan. Anda dapat mengonfigurasi output aplikasi untuk mengarahkan Kinesis Data Analytics untuk menyimpan data aliran kesalahan ke tujuan eksternal untuk evaluasi lebih lanjut. Untuk informasi selengkapnya, lihat Penanganan Kesalahan.

- Amazon Kinesis Data Analytics memastikan catatan output aplikasi Anda ditulis ke tujuan yang dikonfigurasi. Ini menggunakan mode pemrosesan dan pengiriman "setidaknya sekali", meskipun jika Anda mengalami gangguan aplikasi. Lihat informasi yang lebih lengkap di Model Pengiriman untuk Menyimpan Output Aplikasi untuk Tujuan Eksternal.

Topik

- Mengonfigurasi Input Aplikasi
- Kode Aplikasi
- Mengonfigurasi Output Aplikasi
- Penanganan Kesalahan
- Secara Otomatis Menskalakan Aplikasi untuk Meningkatkan Throughput
- Menggunakan Penandaan

Mengonfigurasi Input Aplikasi

Aplikasi Amazon Kinesis Data Analytics Anda dapat menerima input dari satu sumber streaming dan, secara opsional, menggunakan satu sumber data referensi. Untuk informasi selengkapnya, lihat Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya. Bagian dalam topik ini menjelaskan sumber input aplikasi.

Topik

- Mengonfigurasi Sumber Streaming

- Mengonfigurasi Sumber Referensi
- Bekerja dengan JSONPath
- Memetakan Elemen Sumber Streaming ke Kolom Input SQL
- Menggunakan Fitur Penemuan Skema pada Data Streaming
- Menggunakan Fitur Penemuan Skema pada Data Statis
- Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda
- Menyejajarkan Aliran Input untuk Peningkatan Throughput

Mengonfigurasi Sumber Streaming

Ketika Anda membuat aplikasi, Anda menentukan sumber streaming. Anda juga dapat memodifikasi input setelah Anda membuat aplikasi. Amazon Kinesis Data Analytics mendukung sumber streaming berikut untuk aplikasi Anda:

- Kinesis data stream
- Aliran pengiriman Firehose

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Pelanggan lama yang menggunakan Kinesis Data Analytics untuk aplikasi SQL `KinesisFirehoseInput` dengan dapat terus menambahkan aplikasi `KinesisFirehoseInput` dengan dalam akun yang ada menggunakan Kinesis Data Analytics. Jika Anda adalah pelanggan lama dan ingin membuat akun baru dengan Kinesis Data Analytics untuk aplikasi `KinesisFirehoseInput` SQL, Anda dapat membuat kasus melalui formulir peningkatan batas layanan. Untuk informasi selengkapnya, lihat PusatAWS Support. Kami menyarankan untuk selalu menguji aplikasi baru sebelum mempromosikan ke produksi.

> **Note**
>
> Jika Kinesis data stream dienkripsi, Kinesis Data Analytics mengakses data dalam aliran terenkripsi tanpa perlu konfigurasi lebih lanjut. Kinesis Data Analytics tidak menyimpan data

tidak terenkripsi yang dibaca dari Kinesis Data Streams. Untuk informasi selengkapnya, lihat Apa Itu Enkripsi Sisi Server untuk Kinesis Data Streams?.

Kinesis Data Analytics terus melakukan polling terhadap sumber streaming untuk data baru dan menyerapnya di aliran dalam aplikasi sesuai dengan konfigurasi input.

> **Note**
>
> Menambahkan Aliran Kinesis sebagai input aplikasi Anda tidak memengaruhi data dalam aliran. Jika sumber daya lain seperti aliran pengiriman Firehose juga mengakses aliran Kinesis yang sama, aliran pengiriman Firehose dan aplikasi Kinesis Data Analytics akan menerima data yang sama. Namun, throughput dan throttling mungkin akan terpengaruh.

Kode aplikasi Anda dapat mengkueri aliran dalam aplikasi. Sebagai bagian dari konfigurasi input, Anda memberikan yang berikut ini:

- Sumber streaming – Anda memberikan Amazon Resource Name (ARN) aliran dan IAM role yang dapat diasumsikan Kinesis Data Analytics untuk mengakses streaming atas nama Anda.
- Prefiks nama aliran dalam aplikasi – Saat Anda memulai aplikasi, Kinesis Data Analytics membuat aliran dalam aplikasi yang ditentukan. Dalam kode aplikasi Anda, Anda mengakses aliran dalam aplikasi menggunakan nama ini.

  Anda dapat secara opsional memetakan sumber streaming ke beberapa aliran dalam aplikasi. Untuk informasi selengkapnya, lihat Batas. Dalam hal ini, Amazon Kinesis Data Analytics membuat sejumlah aliran dalam aplikasi yang ditentukan dengan nama sebagai berikut: *prefiks*_001,*prefiks*_002, dan *prefiks*_003. Secara default, Kinesis Data Analytics memetakan sumber streaming ke satu aliran dalam aplikasi bernama *prefiks*_001.

  Ada batas kecepatan Anda dapat memasukkan baris di aliran dalam aplikasi. Oleh karena itu, Kinesis Data Analytics mendukung beberapa aliran dalam aplikasi sehingga Anda dapat membawa catatan ke aplikasi Anda dengan kecepatan yang jauh lebih cepat. Jika Anda merasa aplikasi Anda tidak mengikuti data dalam sumber streaming, Anda dapat menambahkan unit paralelisme untuk meningkatkan performa.
- Skema pemetaan – Anda menjelaskan format catatan (JSON, CSV) pada sumber streaming. Anda juga menjelaskan bagaimana setiap catatan di aliran memetakan ke kolom di aliran dalam aplikasi yang dibuat. Di sini Anda memberikan nama kolom dan tipe data.

> **Note**
>
> Kinesis Data Analytics menambahkan tanda kutip di sekitar pengidentifikasi (nama aliran dan nama kolom) saat membuat aliran input dalam aplikasi. Ketika mengkueri aliran ini dan kolom, Anda harus menentukan keduanya dalam tanda kutip menggunakan casing yang sama (mencocokkan huruf kecil dan huruf besar dengan tepat). Untuk informasi selengkapnya tentang pengidentifikasi, lihat Pengidentifikasi di Amazon Managed Service for Apache Flink SQL Reference.

Anda dapat membuat aplikasi dan mengonfigurasi input di konsol Amazon Kinesis Data Analytics. Konsol kemudian membuat panggilan API yang diperlukan. Anda dapat mengonfigurasi input aplikasi ketika Anda membuat API aplikasi baru atau menambahkan konfigurasi input ke aplikasi yang ada. Lihat informasi yang lebih lengkap di CreateApplication dan AddApplicationInput. Berikut adalah bagian konfigurasi input dari isi permintaan API `Createapplication`:

```
"Inputs": [
       {
           "InputSchema": {
               "RecordColumns": [
                   {
                       "Mapping": "string",
                       "Name": "string",
                       "SqlType": "string"
                   }
               ],
               "RecordEncoding": "string",
               "RecordFormat": {
                   "MappingParameters": {
                       "CSVMappingParameters": {
                           "RecordColumnDelimiter": "string",
                           "RecordRowDelimiter": "string"
                       },
                       "JSONMappingParameters": {
                           "RecordRowPath": "string"
                       }
                   },
                   "RecordFormatType": "string"
               }
           },
           "KinesisFirehoseInput": {
```

```
                "ResourceARN": "string",
                "RoleARN": "string"
            },
            "KinesisStreamsInput": {
                "ResourceARN": "string",
                "RoleARN": "string"
            },
            "Name": "string"
        }
    ]
```

Mengonfigurasi Sumber Referensi

Anda juga dapat menambahkan sumber data referensi ke aplikasi yang ada untuk memperkaya data yang masuk dari sumber streaming. Anda harus menyimpan data referensi sebagai objek di bucket Amazon S3. Ketika aplikasi dimulai, Amazon Kinesis Data Analytics membaca objek Amazon S3 dan membuat tabel referensi dalam aplikasi. Kode aplikasi Anda selanjutnya dapat menggabungkannya dengan aliran dalam aplikasi.

Anda menyimpan data referensi di objek Amazon S3 menggunakan format yang didukung (CSV, JSON). Misalnya, aplikasi Anda melakukan analitik pada pesanan stok. Asumsikan format catatan berikut pada sumber streaming:

```
Ticker, SalePrice, OrderId

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

Dalam hal ini, Anda selanjutnya dapat mempertimbangkan untuk mempertahankan sumber data referensi untuk memberikan detail pada setiap ticker saham, seperti nama perusahaan.

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

Anda dapat menambahkan sumber data referensi aplikasi baik dengan API maupun dengan konsol. Amazon Kinesis Data Analytics menyediakan tindakan API berikut untuk mengelola sumber data referensi:

- AddApplicationReferenceDataSource
- UpdateApplication

Untuk informasi selengkapnya tentang menambahkan data referensi menggunakan konsol, lihat Contoh: Menambahkan Data Referensi ke Aplikasi Kinesis Data Analytics.

Perhatikan hal berikut:

- Jika aplikasi berjalan, Kinesis Data Analytics membuat tabel referensi dalam aplikasi, lalu segera memuat data referensi.
- Jika aplikasi tidak berjalan (misalnya, dalam status siap), Kinesis Data Analytics hanya menyimpan konfigurasi input yang diperbarui. Saat aplikasi mulai berjalan, Kinesis Data Analytics memuat data referensi dalam aplikasi Anda sebagai tabel.

Misalkan Anda ingin me-refresh data setelah Kinesis Data Analytics membuat tabel referensi dalam aplikasi. Mungkin Anda memperbarui objek Amazon S3, atau Anda ingin menggunakan objek Amazon S3 yang berbeda. Dalam hal ini, Anda dapat secara eksplisit memanggil UpdateApplication, atau pilih Action (Tindakan), Synchronize reference data table (Sinkronkan tabel data referensi) di konsol. Kinesis Data Analytics tidak me-refresh tabel referensi dalam aplikasi secara otomatis.

Ada batas pada ukuran objek Amazon S3 yang dapat Anda buat sebagai sumber data referensi. Untuk informasi selengkapnya, lihat Batas. Jika ukuran objek melebihi batas, Kinesis Data Analytics tidak dapat memuat data. Status aplikasi muncul sebagai berjalan, tetapi data tidak sedang dibaca.

Ketika Anda menambahkan sumber data referensi, Anda memberikan informasi berikut:

- Bucket S3 dan nama kunci objek – Selain nama bucket dan kunci objek, Anda juga memberikan IAM role yang dapat diasumsikan oleh Kinesis Data Analytics untuk membaca objek atas nama Anda.
- Nama tabel referensi dalam aplikasi – Kinesis Data Analytics membuat tabel dalam aplikasi ini dan mengisinya dengan membaca objek Amazon S3. Ini adalah nama tabel yang Anda tentukan dalam kode aplikasi Anda.
- Skema pemetaan – Anda mendeskripsikan format catatan (JSON, CSV), encoding data yang tersimpan dalam objek Amazon S3. Anda juga menjelaskan bagaimana setiap elemen data dipetakan ke kolom dalam tabel referensi dalam aplikasi.

Berikut ini menunjukkan isi permintaan dalam permintaan API `AddApplicationReferenceDataSource`.

```
{
    "applicationName": "string",
    "CurrentapplicationVersionId": number,
    "ReferenceDataSource": {
        "ReferenceSchema": {
            "RecordColumns": [
                {
                    "IsDropped": boolean,
                    "Mapping": "string",
                    "Name": "string",
                    "SqlType": "string"
                }
            ],
            "RecordEncoding": "string",
            "RecordFormat": {
                "MappingParameters": {
                    "CSVMappingParameters": {
                        "RecordColumnDelimiter": "string",
                        "RecordRowDelimiter": "string"
                    },
                    "JSONMappingParameters": {
                        "RecordRowPath": "string"
                    }
                },
                "RecordFormatType": "string"
            }
        },
        "S3ReferenceDataSource": {
            "BucketARN": "string",
            "FileKey": "string",
            "ReferenceRoleARN": "string"
        },
        "TableName": "string"
    }
}
```

Bekerja dengan JSONPath

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

JSONPath adalah cara standar untuk mengkueri elemen dari objek JSON. JSONPath menggunakan ekspresi jalur untuk menavigasi elemen, elemen bersarang, dan array dalam dokumen JSON. Untuk informasi selengkapnya tentang JSON, lihat Memperkenalkan JSON.

Amazon Kinesis Data Analytics menggunakan ekspresi JSONPath dalam skema sumber aplikasi untuk mengidentifikasi elemen data dalam sumber streaming yang berisi data berformat JSON.

Untuk informasi selengkapnya tentang cara memetakan data streaming ke aliran input aplikasi Anda, lihat the section called “Memetakan Elemen Sumber Streaming ke Kolom Input SQL”.

Mengakses Elemen JSON dengan JSONPath

Berikut ini, Anda dapat menemukan cara menggunakan ekspresi JSONPath untuk mengakses berbagai elemen dalam data berformat JSON. Untuk contoh di bagian ini, anggap aliran sumber berisi catatan JSON berikut:

```
{
  "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" }
```

```
  ]
}
```

Mengakses Elemen JSON

Untuk mengkueri elemen dalam data JSON menggunakan JSONPath, gunakan sintaksis berikut. Di sini, $ mewakili akar hierarki data dan `elementName` adalah nama node elemen yang dikueri.

```
$.elementName
```

Ekspresi berikut mengkueri elemen `customerName` dalam contoh JSON sebelumnya.

```
$.customerName
```

Ekspresi sebelumnya mengembalikan hal berikut dari catatan JSON sebelumnya.

```
John Doe
```

> **Note**
>
> Ekspresi jalur peka huruf besar/kecil. Ekspresi `$.customername` mengembalikan `null` dari contoh JSON sebelumnya.

> **Note**
>
> Jika tidak ada elemen yang muncul di lokasi yang ditentukan oleh ekspresi jalur, ekspresi akan mengembalikan `null`. Ekspresi berikut mengembalikan `null` dari contoh JSON sebelumnya, karena tidak ada elemen yang cocok.
>
> ```
> $.customerId
> ```

Mengakses Elemen JSON Bersarang

Untuk mengkueri elemen JSON bersarang, gunakan sintaksis berikut.

```
$.parentElement.element
```

Ekspresi berikut mengkueri elemen `city` dalam contoh JSON sebelumnya.

```
$.address.city
```

Ekspresi sebelumnya mengembalikan hal berikut dari catatan JSON sebelumnya.

```
Anytown
```

Anda dapat mengkueri tingkat subelemen selanjutnya menggunakan sintaksis berikut.

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

Ekspresi berikut mengkueri elemen `street` dalam contoh JSON sebelumnya.

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

Ekspresi sebelumnya mengembalikan hal berikut dari catatan JSON sebelumnya.

```
AnyStreet
```

Mengakses Array

Anda dapat mengakses data dalam array JSON dengan cara berikut:

- Mengambil semua elemen dalam array sebagai satu baris.
- Mengambil setiap elemen dalam array sebagai baris terpisah.

Mengambil Semua Elemen dalam Array dalam Satu Baris

Untuk mengkueri seluruh isi array sebagai satu baris, gunakan sintaksis berikut.

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

Ekspresi berikut mengkueri seluruh isi elemen `orders` dalam contoh JSON sebelumnya yang digunakan di bagian ini. Ini mengembalikan isi array dalam satu kolom dalam satu baris.

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

Ekspresi sebelumnya mengembalikan hal berikut dari catatan JSON contoh yang digunakan di bagian ini.

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

Mengambil Semua Elemen dalam Array dalam Baris Terpisah

Untuk mengkueri elemen individual dalam array sebagai baris terpisah, gunakan sintaksis berikut.

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

Ekspresi berikut mengkueri elemen `orderId` dalam contoh JSON sebelumnya, dan mengembalikan setiap elemen array sebagai baris terpisah.

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

Ekspresi sebelumnya mengembalikan hal berikut dari catatan JSON sebelumnya, dengan setiap item data yang dikembalikan sebagai baris terpisah.

23284
63122
77284

> **Note**
>
> Jika ekspresi yang mengkueri elemen non-array disertakan dalam skema yang mengkueri elemen array individual, elemen non-array diulang untuk setiap elemen dalam array. Sebagai contoh, misalkan skema untuk contoh JSON sebelumnya termasuk ekspresi berikut:
>
> - $.customerName
> - $.orders[0:].orderId

Dalam kasus ini, baris data yang dikembalikan dari elemen aliran input sampel menyerupai hal berikut, dengan elemen `name` yang diulang untuk setiap elemen `orderId`.

John Doe	23284
John Doe	63122
John Doe	77284

Note

Batasan berikut berlaku untuk ekspresi array di Amazon Kinesis Data Analytics:

- Hanya satu tingkat dereferensi yang didukung dalam ekspresi array. Format ekspresi berikut tidak didukung.

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

- Hanya satu array yang dapat diratakan dalam skema. Beberapa array dapat direferensikan—dikembalikan sebagai satu baris yang berisi semua elemen dalam array. Namun, hanya satu array yang dapat memiliki masing-masing elemen yang dikembalikan sebagai baris individual.

  Skema yang berisi elemen dalam format berikut adalah valid. Format ini mengembalikan isi dari array kedua sebagai satu kolom, diulang untuk setiap elemen dalam array pertama.

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

  Skema yang berisi elemen dalam format berikut tidak valid.

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

Pertimbangan Lainnya

Pertimbangan tambahan untuk bekerja dengan JSONPath adalah sebagai berikut:

- Jika tidak ada array yang diakses oleh elemen individual dalam ekspresi JSONPath dalam skema aplikasi, satu baris dibuat dalam aliran input aplikasi untuk setiap catatan JSON yang diproses.
- Ketika array diratakan (yaitu, elemennya dikembalikan sebagai baris individual), setiap elemen yang hilang menghasilkan nilai null yang dibuat di aliran dalam aplikasi.
- Array selalu diratakan untuk setidaknya satu baris. Jika tidak ada nilai yang akan dikembalikan (yaitu, array kosong atau tidak ada elemen yang dikueri), satu baris dengan semua nilai null dikembalikan.

  Ekspresi berikut mengembalikan catatan dengan nilai null dari contoh JSON sebelumnya, karena tidak ada elemen yang cocok di jalur yang ditentukan.

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

  Ekspresi sebelumnya mengembalikan hal berikut dari catatan contoh JSON sebelumnya.

  ```
  null
  null
  nol
  ```

Topik-Topik Terkait

- Memperkenalkan JSON

Memetakan Elemen Sumber Streaming ke Kolom Input SQL

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

Dengan Amazon Kinesis Data Analytics, Anda dapat memproses dan menganalisis data streaming baik dalam format JSON maupun CSV menggunakan SQL standar.

- Untuk memproses dan menganalisis data CSV streaming, Anda menetapkan nama kolom dan tipe data untuk kolom aliran input. Aplikasi Anda mengimpor satu kolom dari aliran input per definisi kolom, secara berurutan.

  Anda tidak harus menyertakan semua kolom dalam aliran input aplikasi, tetapi Anda tidak dapat melewatkan kolom dari aliran sumber. Misalnya, Anda dapat mengimpor tiga kolom pertama dari aliran input yang berisi lima elemen, tetapi Anda tidak dapat mengimpor kolom 1, 2, dan 4 saja.
- Untuk memproses dan menganalisis data JSON streaming, Anda menggunakan ekspresi JSONPath untuk memetakan elemen JSON dari sumber streaming ke kolom SQL di aliran input. Untuk informasi selengkapnya tentang menggunakan JSONPath dengan Amazon Kinesis Data Analytics, lihat Bekerja dengan JSONPath. Kolom dalam tabel SQL memiliki tipe data yang dipetakan dari tipe JSON. Untuk tipe data yang didukung, lihat Tipe Data. Untuk detail tentang mengonversi data JSON ke data SQL, lihat Memetakan Tipe Data JSON ke Tipe Data SQL.

Untuk informasi selengkapnya tentang cara mengonfigurasi aliran input, lihat Mengonfigurasi Input Aplikasi.

Memetakan Data JSON ke Kolom SQL

Anda dapat memetakan elemen JSON ke kolom input menggunakan AWS Management Console atau Kinesis Data Analytics API.

- Untuk memetakan elemen ke kolom menggunakan konsol, lihat Menggunakan Editor Skema.
- Untuk memetakan elemen ke kolom menggunakan API Kinesis Data Analytics, lihat bagian berikut.

Untuk memetakan elemen JSON ke kolom dalam aliran input dalam aplikasi, Anda memerlukan skema dengan informasi berikut untuk setiap kolom:

- Ekspresi sumber: Ekspresi JSONPath yang mengidentifikasi lokasi data untuk kolom.
- Nama Kolom: Nama yang kueri SQL Anda gunakan untuk mereferensi data.
- Tipe Data: Tipe data SQL untuk kolom.

Menggunakan API

Untuk memetakan elemen dari sumber streaming ke kolom input, Anda dapat menggunakan tindakan CreateApplication API Kinesis Data Analytics. Untuk membuat aliran dalam aplikasi, tentukan skema untuk mengubah data Anda menjadi versi skematis yang digunakan di SQL. Tindakan CreateApplication mengonfigurasi aplikasi Anda untuk menerima input dari satu sumber streaming. Untuk memetakan elemen JSON atau kolom CSV ke kolom SQL, Anda membuat objek RecordColumn di array SourceSchema `RecordColumns`. Objek RecordColumn memiliki skema-skema berikut:

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

Bidang dalam objek RecordColumn memiliki nilai berikut:

- `Mapping`: Ekspresi JSONPath yang mengidentifikasi lokasi data dalam catatan aliran input. Nilai ini tidak tersedia untuk skema input untuk aliran sumber dalam format CSV.
- `Name`: Nama kolom dalam aliran data SQL dalam aplikasi.
- `SqlType`: Tipe data dari data dalam aliran data SQL dalam aplikasi.

Contoh Skema Input JSON

Contoh berikut menunjukkan format nilai `InputSchema` untuk skema JSON.

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

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

Contoh Skema Input CSV

Contoh berikut menunjukkan format nilai `InputSchema` untuk skema dalam format nilai yang dipisahkan koma (CSV).

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

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

Memetakan Tipe Data JSON ke Tipe Data SQL

Tipe data JSON dikonversi menjadi tipe data SQL yang sesuai menurut skema input aplikasi. Untuk informasi tentang tipe data SQL yang didukung, lihat Tipe Data. Amazon Kinesis Data Analytics mengonversi tipe data JSON menjadi tipe data SQL sesuai dengan aturan berikut.

Null Literal

Literal null dalam aliran input JSON (yaitu, `"City":null`) mengonversi ke null SQL terlepas dari tipe data tujuan.

Boolean Literal

Boolean literal dalam aliran input JSON (yaitu,`"Contacted":true`) mengonversi ke data SQL sebagai berikut:

- Numerik (DECIMAL, INT, dan sebagainya): `true` mengonversi menjadi 1; `false` mengonversi menjadi 0.
- Biner (BINARY atau VARBINARY):
  - `true`: Hasil memiliki set bit terendah dan sisa bit yang dihapus.
  - `false`: Hasil memiliki semua bit yang dihapus.

  Konversi ke VARBINARY menghasilkan nilai dengan panjang 1 byte.
- BOOLEAN: Mengonversi menjadi nilai SQL BOOLEAN yang sesuai.
- Karakter (CHAR atau VARCHAR): Mengonversi ke nilai string yang sesuai (`true` atau `false`). Nilai dipotong agar sesuai dengan panjang bidang.

- Datetime (DATE, TIME, atau TIMESTAMP): Konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.

Angka

Angka literal dalam aliran input JSON (yaitu,`"CustomerId":67321`) mengonversi ke data SQL sebagai berikut:

- Numerik (DECIMAL, INT, dan sebagainya): Mengonversi langsung. Jika nilai yang dikonversi melebihi ukuran atau presisi dari tipe data target (yaitu, mengonversi `123.4` ke INT), konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.
- Biner (BINARY atau VARBINARY): Konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.
- BOOLEAN:
  - `0`: Mengonversi ke `false`.
  - Semua angka lainnya: Mengkonversi ke`true`.
- Karakter (CHAR atau VARCHAR): Mengonversi ke representasi string dari nomor tersebut.
- Datetime (DATE, TIME, atau TIMESTAMP): Konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.

String

Nilai string dalam aliran input JSON (yaitu, `"CustomerName":"John Doe"`) mengonversi ke data SQL sebagai berikut:

- Numerik (DECIMAL, INT, dan sebagainya): Amazon Kinesis Data Analytics mencoba mengonversi nilai ke tipe data target. Jika nilai tidak dapat dikonversi, konversi gagal dan kesalahan paksaan ditulis ke pengalliran kesalahan.
- Biner (BINARY atau VARBINARY): Jika string sumber adalah biner literal yang valid (yaitu `X'3F67A23A'`, dengan bilangan genap f), nilai dikonversi ke tipe data target. Jika tidak, konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.
- BOOLEAN: Jika string sumber adalah `"true"`, mengonversi ke `true`. Perbandingan ini peka terhadap huruf besar-kecil. Jika tidak, mengonversi ke `false`.
- Karakter (CHAR atau VARCHAR): Mengonversi ke nilai string dalam input. Jika nilai lebih panjang dari tipe data target, nilai dipotong dan tidak ada kesalahan yang ditulis ke aliran kesalahan.

- Datetime (DATE, WAKTU, atau TIMESTAMP): Jika string sumber dalam format yang dapat dikonversi ke nilai target, nilai akan dikonversi. Jika tidak, konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.

  Format datetime yang valid meliputi:

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

Array atau Objek

Array atau objek dalam aliran input JSON mengonversi ke data SQL sebagai berikut:

- Karakter (CHAR atau VARCHAR): Mengonversi ke teks sumber dari array atau objek. Lihat Mengakses Array.
- Semua tipe data lainnya: Konversi gagal dan kesalahan paksaan ditulis ke aliran kesalahan.

Untuk contoh array JSON, lihat Bekerja dengan JSONPath.

Topik-Topik Terkait

- Mengonfigurasi Input Aplikasi
- Tipe Data
- Menggunakan Editor Skema
- CreateApplication
- RecordColumn
- SourceSchema

Menggunakan Fitur Penemuan Skema pada Data Streaming

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

Menyediakan skema input yang menjelaskan bagaimana catatan pada peta input streaming untuk aliran dalam aplikasi bisa merepotkan dan rawan kesalahan. Anda dapat menggunakan API DiscoverInputSchema (disebut API penemuan) untuk menyimpulkan skema. Menggunakan sampel catatan acak pada sumber streaming, API dapat menyimpulkan skema (yaitu, nama kolom, tipe data, dan posisi elemen data dalam data masuk).

> **Note**
>
> Untuk menggunakan API Penemuan untuk menghasilkan skema dari file yang disimpan di Amazon S3, lihat Menggunakan Fitur Penemuan Skema pada Data Statis.

Konsol menggunakan API Penemuan untuk menghasilkan skema untuk sumber streaming tertentu. Dengan menggunakan konsol, Anda juga dapat memperbarui skema, termasuk menambahkan atau menghapus kolom, mengubah nama kolom atau tipe data, dan sebagainya. Namun, buat perubahan dengan hati-hati untuk memastikan Anda tidak membuat skema yang tidak valid.

Setelah Anda menyelesaikan skema untuk aliran dalam aplikasi Anda, ada fungsi yang dapat Anda gunakan untuk memanipulasi nilai string dan datetime. Anda dapat menggunakan fungsi-fungsi ini dalam kode aplikasi Anda ketika bekerja dengan baris dalam aliran dalam aplikasi yang dihasilkan. Untuk informasi selengkapnya, lihat Contoh: Mengubah Nilai DateTime .

Penamaan Kolom Selama Penemuan Skema

Selama penemuan skema, Amazon Kinesis Data Analytics mencoba mempertahankan sebanyak mungkin nama kolom asli dari sumber input streaming, kecuali dalam kasus berikut:

- Nama kolom aliran sumber adalah kata kunci SQL tersimpan, seperti `TIMESTAMP`, `USER`, `VALUES`, atau `YEAR`.
- Nama kolom aliran sumber berisi karakter yang tidak didukung. Hanya huruf, angka, dan karakter garis bawah (_) yang didukung.
- Nama kolom aliran sumber dimulai dengan angka.
- Nama kolom aliran sumber lebih panjang dari 100 karakter.

Jika nama kolom diganti, nama kolom skema diganti dimulai dengan `COL_`. Dalam beberapa kasus, tidak ada nama kolom asli yang dapat dipertahankan—misalnya, jika seluruh nama adalah karakter yang tidak didukung. Dalam kasus seperti itu, kolom diberi nama `COL_#`, dengan # menjadi angka yang menunjukkan tempat kolom dalam urutan kolom.

Setelah penemuan selesai, Anda dapat memperbarui skema menggunakan konsol untuk menambah atau menghapus kolom, atau mengubah nama kolom, tipe data, atau ukuran data.

Contoh Nama Kolom yang Disarankan Penemuan

Nama Kolom Aliran Sumber	Nama Kolom yang Disarankan Penemuan
USER	COL_USER
USER@DOMAIN	COL_USERDOMAIN
@@	COL_0

Masalah Penemuan Skema

Apa yang terjadi jika Kinesis Data Analytics tidak menyimpulkan skema untuk sumber streaming tertentu?

Kinesis Data Analytics menyimpulkan skema Anda untuk format umum, seperti CSV dan JSON, yang dikodekan UTF-8. Kinesis Data Analytics mendukung catatan apa pun yang dikodekan UTF-8 (termasuk teks mentah seperti log dan catatan aplikasi) dengan kolom khusus dan pembatas baris. Jika Kinesis Data Analytics tidak menyimpulkan skema, Anda dapat menentukan skema secara manual menggunakan editor skema di konsol (atau menggunakan API).

Jika data Anda tidak mengikuti pola (yang dapat Anda tentukan menggunakan editor skema), Anda dapat menentukan skema sebagai satu kolom tipe VARCHAR(N), dengan N adalah jumlah karakter terbesar yang Anda harapkan akan disertakan dalam catatan Anda. Dari sana, Anda dapat menggunakan string dan manipulasi date-time untuk menyusun data Anda setelah berada di aliran dalam aplikasi. Sebagai contoh, lihat Contoh: Mengubah Nilai DateTime .

Menggunakan Fitur Penemuan Skema pada Data Statis

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

Fitur penemuan skema dapat menghasilkan skema baik dari data dalam aliran maupun data dalam file statis yang disimpan dalam bucket Amazon S3. Misalkan Anda ingin membuat skema untuk aplikasi Kinesis Data Analytics untuk tujuan referensi atau saat data streaming langsung tidak tersedia. Anda dapat menggunakan fitur penemuan skema pada file statis yang berisi sampel data dalam format yang diharapkan dari data streaming atau referensi Anda. Kinesis Data Analytics dapat menjalankan penemuan skema pada data sampel dari file JSON atau CSV yang disimpan di bucket Amazon S3. Menggunakan penemuan skema pada file data baik menggunakan konsol, maupun API DiscoverInputSchema dengan parameter `S3Configuration` yang ditentukan.

Menjalankan Penemuan Skema Menggunakan Konsol

Untuk menjalankan penemuan pada file statis menggunakan konsol, lakukan hal berikut:

1. Tambahkan objek data referensi ke bucket S3.
2. Pilih Connect reference data (Sambungkan data referensi) di halaman utama aplikasi di konsol Kinesis Data Analytics.
3. Sediakan bucket, data jalur dan IAM role untuk mengakses objek Amazon S3 yang berisi data referensi.
4. Pilih Discover schema (Temukan skema).

Untuk informasi selengkapnya tentang cara menambahkan data referensi dan menemukan skema di konsol, lihat Contoh: Menambahkan Data Referensi ke Aplikasi Kinesis Data Analytics.

Menjalankan Penemuan Skema Menggunakan API

Untuk menjalankan penemuan pada file statis menggunakan API, Anda menyediakan API dengan struktur `S3Configuration` menggunakan informasi berikut:

- `BucketARN`: Amazon Resource Name (ARN) dari bucket Amazon S3 yang berisi file. Untuk format ARN bucket Amazon S3, lihat Amazon Resource Names (ARN) dan Amazon Service Namespaces: Amazon Simple Storage Service (Amazon S3).
- `RoleARN`: ARN IAM role dengan kebijakan `AmazonS3ReadOnlyAccess`. Untuk informasi tentang cara menambahkan kebijakan ke peran, lihat Mengubah Peran.
- `FileKey`: Nama file objek.

Untuk menghasilkan skema dari objek Amazon S3 menggunakan API **`DiscoverInputSchema`**

1. Pastikan Anda memiliki AWS CLI pengaturan. Untuk informasi selengkapnya, lihat Langkah 2: Mengatur AWS Command Line Interface (AWS CLI) dalam bagian Memulai.
2. Buat file bernama `data.csv` dengan konten berikut:

```
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. Masuk ke konsol Amazon S3 di https://console.aws.amazon.com/s3/.
4. Buat bucket Amazon S3 dan unggah file `data.csv` yang Anda buat. Catat ARN dari bucket yang dibuat. Untuk informasi tentang membuat bucket Amazon S3 dan mengunggah file, lihat Memulai dengan Amazon Simple Storage Service.
5. Buka konsol IAM di https://console.aws.amazon.com/iam/. Buat peran dengan kebijakan `AmazonS3ReadOnlyAccess`. Catat ARN peran baru. Untuk informasi selengkapnya tentang membuat peran, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan Amazon. Untuk informasi tentang cara menambahkan kebijakan ke peran, lihat Mengubah Peran.
6. Jalankan `DiscoverInputSchema` perintah berikut di AWS CLI, ganti ARN untuk bucket Amazon S3 dan peran IAM Anda:

```
$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. Responsnya terlihat seperti berikut ini:

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

Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

Jika data dalam aliran Anda memerlukan konversi format, transformasi, pengayaan, atau pemfilteran, Anda dapat memproses data menggunakan fungsi terlebih dahulu. AWS Lambda Anda dapat melakukan ini sebelum kode SQL aplikasi Anda mengeksekusi atau sebelum aplikasi Anda membuat skema dari aliran data Anda.

Menggunakan fungsi Lambda untuk memproses catatan terlebih dulu berguna dalam skenario berikut:

- Mengubah catatan dari format lain (seperti KPL atau GZIP) ke dalam format yang dapat dianalisis oleh Kinesis Data Analytics. Kinesis Data Analytics saat ini mendukung format data JSON atau CSV.
- Memperluas data ke dalam format yang lebih mudah diakses untuk operasi seperti agregasi atau deteksi anomali. Misalnya, jika beberapa nilai data disimpan bersama-sama dalam string, Anda dapat memperluas data ke dalam kolom terpisah.
- Pengayaan data dengan layanan Amazon lainnya, seperti ekstrapolasi atau koreksi kesalahan.
- Menerapkan transformasi string yang kompleks untuk mencatat bidang.
- Penyaringan data untuk membersihkan data.

Menggunakan Fungsi Lambda untuk Memproses Catatan Terlebih Dulu

Saat membuat aplikasi Kinesis Data Analytics, Anda mengaktifkan prapemrosesan Lambda di halaman Connect to a Source (Sambungkan ke Sumber).

Untuk menggunakan fungsi Lambda untuk memproses catatan terlebih dulu di aplikasi Kinesis Data Analytics

1. Masuk ke AWS Management Console dan buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Di halaman Connect to a Source (Sambungkan ke Sumber) untuk aplikasi Anda, pilih Enabled (Aktifkan) di bagian Catan prapemrosesan dengan AWS Lambda.
3. Untuk menggunakan fungsi Lambda yang sudah Anda buat, pilih fungsi di daftar menurun Fungsi Lambda.
4. Untuk membuat fungsi Lambda baru dari salah satu templat prapemrosesan Lambda, pilih template dari daftar menurun. Selanjutnya pilih View <template name> in Lambda (Lihat <template name> di Lambda) untuk mengedit fungsi.
5. Untuk membuat fungsi Lambda baru, pilih Create new (Buat baru). Untuk informasi tentang membuat fungsi Lambda, lihat Membuat Fungsi HelloWorld Lambda dan Menjelajahi Konsol di Panduan Pengembang.AWS Lambda
6. Pilih versi fungsi Lambda yang akan digunakan. Untuk menggunakan versi terbaru, pilih $LATEST.

Saat Anda memilih atau membuat fungsi Lambda untuk prapemrosesan catatan, catatan diproses terlebih dulu sebelum kode SQL aplikasi Anda mengeksekusi atau aplikasi Anda menghasilkan skema dari catatan.

Izin Prapemrosesan Lambda

Untuk menggunakan prapemrosesan Lambda, IAM role aplikasi memerlukan kebijakan izin berikut:

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

```
        "Resource": "<FunctionARN>"
    }
```

Metrik Prapemrosesan Lambda

Anda dapat menggunakan Amazon CloudWatch untuk memantau jumlah pemanggilan Lambda, byte yang diproses, keberhasilan dan kegagalan, dan sebagainya. Untuk informasi tentang CloudWatch metrik yang dipancarkan oleh preprocessing Kinesis Data Analytics Lambda, lihat Metrik Amazon Kinesis Analytics.

Menggunakan AWS Lambda dengan Perpustakaan Produser Kinesis

Kinesis Producer Library (KPL) menggabungkan catatan kecil yang diformat pengguna ke dalam catatan yang lebih besar hingga 1 MB untuk memanfaatkan throughput Amazon Kinesis Data Streams dengan lebih baik. Kinesis Client Library (KCL) untuk Java mendukung pemisahkan catatan-catatan ini. Namun, Anda harus menggunakan modul khusus untuk mendeagregasi catatan saat Anda menggunakan AWS Lambda sebagai konsumen aliran Anda.

Untuk mendapatkan kode dan instruksi proyek yang diperlukan, lihat Modul Deagregasi Perpustakaan Produser Kinesis untuk selanjutnya. AWS LambdaGitHub Anda dapat menggunakan komponen dalam proyek ini untuk memproses data serial KPL AWS Lambda dalam Java, Node.js, dan Python. Anda juga dapat menggunakan komponen ini sebagai bagian dari aplikasi KCL multibahasa.

Model Data Input Peristiwa Prapemrosesan Data/Model Respons Catatan

Untuk memproses catatan terlebih dulu, fungsi Lambda Anda harus sesuai dengan data input peristiwa yang diperlukan dan model respon catatan.

Model Data Input Peristiwa

Kinesis Data Analytics terus membaca data dari aliran data Kinesis atau aliran pengiriman Firehose Anda. Untuk setiap batch catatan yang diambil, layanan mengelola bagaimana setiap batch akan diteruskan ke fungsi Lambda Anda. Fungsi Anda menerima daftar catatan sebagai input. Dalam fungsi Anda, Anda mengulangi melalui daftar dan menerapkan logika bisnis Anda untuk memenuhi persyaratan prapemrosesan Anda (seperti konversi format data atau pengayaan).

Model input ke fungsi preprocessing Anda sedikit berbeda, tergantung pada apakah data diterima dari aliran data Kinesis atau aliran pengiriman Firehose.

Jika sumbernya adalah aliran pengiriman Firehose, model data input peristiwa adalah sebagai berikut:

Model Data Permintaan Selang Api Data Kinesis

Bidang	Deskripsi
`invocationId`	Id invokasi Lambda (GUID acak).
`applicationArn`	Amazon Resource Name (ARN) aplikasi Kinesis Data Analytics
`streamArn`	ARN aliran pengiriman
catatan	

Bidang	Deskripsi	
`recordId`	ID catatan (GUID acak)	
`kinesisFirehoseRecordMetadata`		

Bidang	Deskripsi
`approximateArrivalTimestamp`	Perkiraan waktu kedatangan catatan aliran pengiriman

`data`	Muatan catatan sumber berkode Base64	

Contoh berikut menunjukkan input dari aliran pengiriman Firehose:

```
{
   "invocationId":"00540a87-5050-496a-84e4-e7d92bbaf5e2",
   "applicationArn":"arn:aws:kinesisanalytics:us-east-1:12345678911:application/lambda-
test",
   "streamArn":"arn:aws:firehose:us-east-1:AAAAAAAAAAAA:deliverystream/lambda-test",
   "records":[
      {
         "recordId":"49572672223665514422805246926656954630972486059535892482",
```

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

Jika sumbernya adalah Kinesis data stream, model data input peristiwa adalah sebagai berikut:

Model Data Permintaan Aliran Kinesis

Bidang	Deskripsi
`invocationId`	Id invokasi Lambda (GUID acak).
`applicationArn`	ARN aplikasi Kinesis Data Analytics
`streamArn`	ARN aliran pengiriman
catatan	

Bidang	Deskripsi
`recordId`	ID catatan berdasarkan nomor urut catatan Kinesis
`kinesisStreamRecordMetadata`	

Bidang	Deskripsi
`sequenceNumber`	Nomor urut dari catatan aliran Kinesis
`partitionKey`	Kunci partisi dari catatan aliran Kinesis

Bidang	Deskripsi
Bidang	Deskripsi
Bidang	Deskripsi
shardId	ShardId dari catatan aliran Kinesis
approximateArrivalTimestamp	Perkiraan waktu kedatangan catatan aliran pengiriman
data	Muatan catatan sumber berkode Base64

Contoh berikut menunjukkan input dari Kinesis data 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
         }
    }
  ]
}
```

Model Respons Catatan

Semua catatan yang dikembalikan dari fungsi prapemrosesan Lambda Anda (dengan ID catatan) yang dikirim ke fungsi Lambda harus dikembalikan. Catatan tersebut harus berisi parameter berikut, atau Kinesis Data Analytics menolaknya dan menganggapnya sebagai kegagalan prapemrosesan data. Bagian muatan data dari catatan dapat diubah untuk mencapai persyaratan prapemrosesan.

Model Data Respons

catatan

Bidang	Deskripsi
`recordId`	ID catatan diteruskan dari Kinesis Data Analytics ke Lambda selama invokasi. Catatan yang diubah harus berisi ID catatan yang sama. Ketidakcocokan apa pun antara ID dari catatan asli dan ID dari catatan yang diubah dianggap sebagai kegagalan prapemrosesan data.
`result`	Status transformasi data dari catatan. Nilai yang mungkin adalah: • `Ok`: Catatan berhasil diubah. Kinesis Data Analytics menyerap catatan untuk pemrosesan SQL. • `Dropped`: Catatan sengaja dijatuhkan oleh logika pemrosesan Anda. Kinesis Data Analytics menjatuhkan catatan dari pemrosesan SQL. Bidang muatan data bersifat opsional untuk catatan `Dropped`. • `ProcessingFailed` : Catatan tidak dapat diubah. Kinesis Data Analytics menganggapnya tidak berhasil diproses oleh fungsi Lambda Anda dan menulis kesalahan pada aliran kesalahan. Untuk informasi selengkapnya tentang aliran kesalahan, lihat Penanganan Kesalahan. Bidang muatan data bersifat opsional untuk catatan `ProcessingFailed` .
`data`	Muatan data yang diubah, setelah enkode base64. Setiap muatan data dapat berisi beberapa dokumen JSON jika format data penyerapan aplikasi adalah JSON. Atau masing-masing

Bidang	Deskripsi
	dapat berisi beberapa baris CSV (dengan pembatas baris yang ditentukan di setiap baris) jika format data penyerapan aplikasi adalah CSV. Layanan Kinesis Data Analytics berhasil mengurai dan memproses data dengan beberapa dokumen JSON atau baris CSV dalam muatan data yang sama.

Contoh berikut menunjukkan output dari fungsi Lambda:

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

Kegagalan Prapemrosesan Data Umum

Berikut adalah alasan umum prapemrosesan bisa gagal.

- Tidak semua catatan (dengan ID catatan) dalam batch yang dikirim ke fungsi Lambda dikembalikan ke layanan Kinesis Data Analytics.
- Respons kehilangan ID catatan, status, atau bidang muatan data. Bidang muatan data bersifat opsional untuk catatan `Dropped` atau `ProcessingFailed`.
- Waktu habis fungsi Lambda tidak cukup untuk memproses data terlebih dulu.
- Respons fungsi Lambda melebihi batas respons yang dikenakan oleh layanan AWS Lambda .

Untuk kegagalan prapemrosesan data, Kinesis Data Analytics terus mencoba kembali invokasi Lambda pada kumpulan catatan yang sama hingga berhasil. Anda dapat memantau CloudWatch metrik berikut untuk mendapatkan wawasan tentang kegagalan.

- Aplikasi Kinesis Data Analytics `MillisBehindLatest`: Menunjukkan seberapa jauh aplikasi membaca dari sumber streaming.

- Metrik `InputPreprocessing` CloudWatch aplikasi Kinesis Data Analytics: Menunjukkan jumlah keberhasilan dan kegagalan, di antara statistik lainnya. Untuk informasi selengkapnya, lihat Metrik Amazon Kinesis Analytics.
- AWS Lambda CloudWatch metrik fungsi dan log.

Membuat Fungsi Lambda untuk Prapemrosesan

Aplikasi Amazon Kinesis Data Analytics Anda dapat menggunakan fungsi Lambda untuk memproses catatan terlebih dulu saat dimasukkan ke dalam aplikasi. Kinesis Data Analytics menyediakan templat berikut di konsol untuk digunakan sebagai titik awal prapemrosesan data Anda.

Topik

- Membuat Fungsi Lambda Prapemrosesan di Node.js
- Membuat Fungsi Lambda Prapemrosesan di Phyton
- Membuat Fungsi Lambda Prapemrosesan di Java
- Membuat Fungsi Lambda Prapemrosesan di .NET

Membuat Fungsi Lambda Prapemrosesan di Node.js

Templat berikut untuk membuat fungsi Lambda prapemrosesan di Node.js tersedia di konsol Kinesis Data Analytics:

Cetak Biru Lambda	Bahasa dan versi	Deskripsi
Pemrosesan Input Kinesis Data Analytics Umum	Node.js 6.10	Praprosesor catatan Kinesis Data Analytics yang menerima catatan JSON atau CSV sebagai input, lalu mengembalikannya dengan status pemrosesan. Gunakan prosesor ini sebagai titik awal untuk logika transformasi kustom.
Proses Input Terkompresi	Node.js 6.10	Prosesor catatan Kinesis Data Analytics yang menerima catatan JSON atau CSV yang dikompresi (GZIP atau Deflate yang dikompresi) sebagai input dan mengembalikan catatan yang didekompresi dengan status pemrosesan.

Membuat Fungsi Lambda Prapemrosesan di Phyton

Templat berikut untuk membuat fungsi Lambda prapemrosesan di Phyton tersedia di konsol:

Cetak Biru Lambda	Bahasa dan versi	Deskripsi
Pemrosesan Input Kinesis Analytics Umum	Python 2.7	Praprosesor catatan Kinesis Data Analytics yang menerima catatan JSON atau CSV sebagai input, lalu mengembalikannya dengan status pemrosesan. Gunakan prosesor ini sebagai titik awal untuk logika transformasi kustom.
Pemrosesan Input KPL	Python 2.7	Prosesor catatan Kinesis Data Analytics yang menerima agregat catatan JSON atau CSV Kinesis Producer Library (KPL) sebagai input dan mengembalikan catatan yang dipisah dengan status pemrosesan.

Membuat Fungsi Lambda Prapemrosesan di Java

Untuk membuat fungsi Lambda di Java untuk prapemrosesan catatan, gunakan kelas peristiwa Java.

Kode berikut menunjukkan fungsi Lambda sampel yang memproses catatan terlebih dahulu menggunakan 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;
    }

}
```

Membuat Fungsi Lambda Prapemrosesan di .NET

Untuk membuat fungsi Lambda di .NET untuk prapemrosesan catatan, gunakan kelas peristiwa .NET.

Kode berikut menunjukkan fungsi Lambda sampel yang memproses catatan terlebih dahulu menggunakan 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;
        }
    }
```

Untuk informasi selengkapnya tentang membuat fungsi Lambda untuk prapemrosesan dan tujuan di .NET, lihat `Amazon.Lambda.KinesisAnalyticsEvents`.

Menyejajarkan Aliran Input untuk Peningkatan Throughput

> **Note**
>
> Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Untuk informasi selengkapnya, lihat Batasan-batasan.

Aplikasi Amazon Kinesis Data Analytics dapat mendukung beberapa aliran input dalam aplikasi, untuk menskalakan aplikasi di luar throughput satu aliran input dalam aplikasi. Untuk informasi selengkapnya tentang aliran input dalam aplikasi, lihat Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya.

Dalam hampir semua kasus, Amazon Kinesis Data Analytics menskalakan aplikasi Anda untuk menangani kapasitas aliran Kinesis atau aliran sumber Firehose yang dimasukkan ke dalam aplikasi

Anda. Namun, jika throughput aliran sumber Anda melebihi throughput satu aliran input dalam aplikasi, Anda dapat secara eksplisit meningkatkan jumlah aliran input dalam aplikasi yang digunakan aplikasi Anda. Anda melakukannya dengan parameter `InputParallelism`.

Saat parameter `InputParallelism` lebih besar dari satu, Amazon Kinesis Data Analytics membagi partisi aliran sumber Anda secara merata di antara aliran dalam aplikasi. Misalnya, jika aliran sumber Anda memiliki 50 serpihan, dan Anda mengatur `InputParallelism` ke 2, setiap aliran input dalam aplikasi menerima input dari 25 serpihan aliran sumber.

Ketika Anda meningkatkan jumlah aliran dalam aplikasi, aplikasi Anda harus mengakses data di setiap aliran secara eksplisit. Untuk informasi tentang mengakses beberapa aliran dalam aplikasi di kode Anda, lihat Mengakses Aliran dalam Aplikasi Terpisah di Aplikasi Amazon Kinesis Data Analytics Anda.

Meskipun Kinesis Data Streams dan Firehose stream shards keduanya dibagi di antara aliran dalam aplikasi dengan cara yang sama, keduanya berbeda dalam cara tampilannya ke aplikasi Anda:

- Catatan dari Kinesis data stream mencakup bidang `shard_id` yang dapat digunakan untuk mengidentifikasi serpihan sumber untuk catatan.
- Catatan dari aliran pengiriman Firehose tidak menyertakan bidang yang mengidentifikasi pecahan atau partisi sumber rekaman. Ini karena Firehose mengabstraksi informasi ini dari aplikasi Anda.

Mengevaluasi Apakah Akan Meningkatkan Jumlah Aliran Input dalam Aplikasi Anda

Dalam kebanyakan kasus, satu aliran input dalam aplikasi dapat menangani throughput dari satu aliran sumber, tergantung pada kompleksitas dan ukuran data dari input aliran. Untuk menentukan apakah Anda perlu menambah jumlah aliran input dalam aplikasi, Anda dapat memantau `InputBytes` dan `MillisBehindLatest` metrik di Amazon. CloudWatch

Jika metrik `InputBytes` lebih besar dari 100 MB/detik (atau Anda mengantisipasi bahwa metrik akan lebih besar dari tingkat ini), ini dapat menyebabkan peningkatan `MillisBehindLatest` dan meningkatkan dampak dari masalah aplikasi. Untuk mengatasi hal ini, sebaiknya buat pilihan bahasa berikut untuk aplikasi Anda:

- Gunakan beberapa aliran dan Kinesis Data Analytics untuk aplikasi SQL jika aplikasi Anda harus menskalakan lebih dari 100 MB/detik.
- Gunakan Kinesis Data Analytics untuk Aplikasi Java jika Anda ingin menggunakan satu aliran dan aplikasi.

Jika metrik `MillisBehindLatest` memiliki salah satu karakteristik berikut, Anda harus meningkatkan pengaturan `InputParallelism` aplikasi Anda:

- Metrik `MillisBehindLatest` meningkat secara bertahap, menunjukkan aplikasi Anda tertinggal dari data terbaru dalam aliran.
- Metrik `MillisBehindLatest` secara konsisten di atas 1000 (satu detik).

Anda tidak perlu meningkatkan pengaturan `InputParallelism` aplikasi Anda jika berikut ini benar:

- Metrik `MillisBehindLatest` berkurang secara bertahap, menunjukkan aplikasi Anda mengikuti data terbaru dalam aliran.
- Metrik `MillisBehindLatest` di bawah 1000 (satu detik).

Untuk informasi selengkapnya tentang penggunaan CloudWatch, lihat Panduan CloudWatch Pengguna.

Menerapkan Beberapa Aliran Input dalam Aplikasi

Anda dapat mengatur jumlah aliran input dalam aplikasi ketika aplikasi dibuat menggunakan CreateApplication. Anda menetapkan jumlah ini setelah aplikasi dibuat menggunakan UpdateApplication.

> **Note**
>
> Anda hanya dapat menetapkan pengaturan `InputParallelism` menggunakan API Amazon Kinesis Data Analytics atau AWS CLI. Anda tidak dapat mengatur pengaturan ini menggunakan AWS Management Console. Untuk informasi tentang pengaturan AWS CLI, lihatLangkah 2: Mengatur AWS Command Line Interface (AWS CLI).

Mengatur Jumlah Aliran Input Aplikasi Baru

Contoh berikut menunjukkan cara menggunakan tindakan API `CreateApplication` untuk mengatur jumlah aliran input aplikasi baru ke 2.

Untuk informasi selengkapnya tentang `CreateApplication`, lihat 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": [ ... ],
 }]
}
```

Mengatur Jumlah Aliran Input Aplikasi yang Ada

Contoh berikut menunjukkan cara menggunakan tindakan API `UpdateApplication` untuk mengatur jumlah aliran input aplikasi yang ada ke 2.

Untuk informasi selengkapnya tentang `Update_Application`, lihat UpdateApplication.

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

Mengakses Aliran dalam Aplikasi Terpisah di Aplikasi Amazon Kinesis Data Analytics Anda

Untuk menggunakan beberapa aliran input dalam aplikasi Anda, Anda harus secara eksplisit memilih dari aliran yang berbeda. Contoh kode berikut menunjukkan bagaimana query beberapa input stream dalam aplikasi yang dibuat dalam tutorial Memulai.

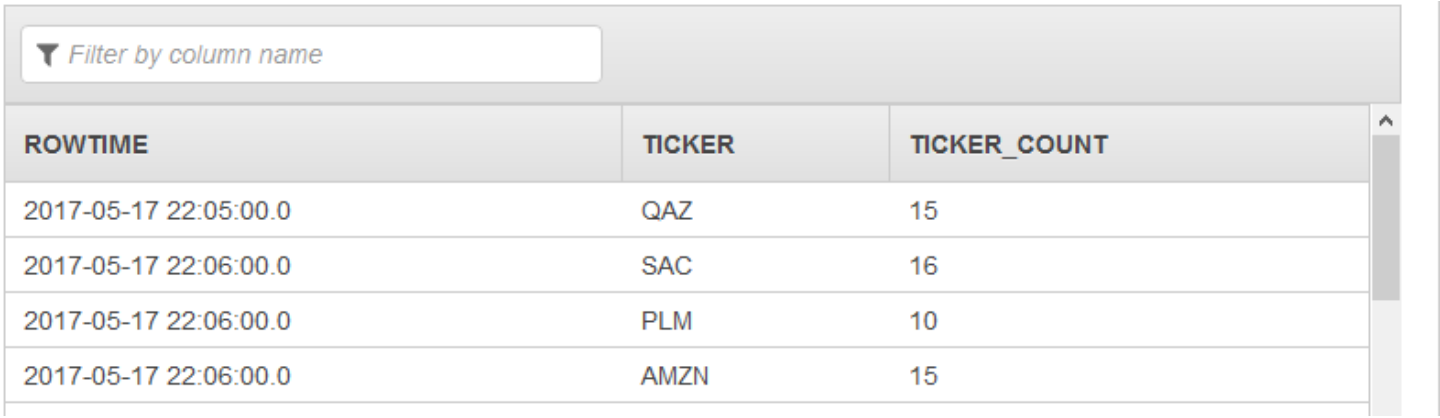
Dalam contoh berikut, setiap pengairan sumber dikumpulkan terlebih dulu menggunakan COUNT sebelum digabungkan ke dalam satu aliran dalam aplikasi yang disebut

`in_application_stream001`. Menggabungkan aliran sumber sebelumnya membantu memastikan aliran dalam aplikasi gabungan dapat menangani lalu lintas dari beberapa aliran tanpa kelebihan muatan.

> **Note**
>
> Untuk menjalankan contoh ini dan mendapatkan hasil dari kedua aliran input dalam aplikasi, perbarui kedua jumlah serpihan dalam aliran sumber Anda dan parameter `InputParallelism` dalam aplikasi Anda.

```
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;
```

Contoh kode sebelumnya menghasilkan output dalam `in_application_stream001` yang mirip dengan yang berikut:



Pertimbangan Tambahan

Saat menggunakan beberapa aliran input, perhatikan hal berikut:

- Jumlah maksimum aliran input dalam aplikasi adalah 64.
- Aliran input dalam aplikasi didistribusikan secara merata di antara serpihan aliran input aplikasi.
- Keuntungan performa dari menambahkan aliran dalam aplikasi tidak diskalakan secara linier. Artinya, menggandakan jumlah aliran dalam aplikasi tidak menggandakan throughput. Dengan ukuran baris yang khas, setiap aliran dalam aplikasi dapat mencapai throughput sekitar 5.000 hingga 15.000 baris per detik. Dengan meningkatkan jumlah aliran dalam aplikasi hingga 10, Anda dapat mencapai throughput 20.000 hingga 30.000 baris per detik. Kecepatan throughput tergantung pada jumlah, tipe data, dan ukuran data dari bidang dalam aliran input.
- Beberapa fungsi agregat (seperti AVG) dapat membuat hasil yang tidak terduga ketika diterapkan ke aliran input yang dipartisi menjadi serpihan yang berbeda. Karena Anda perlu menjalankan operasi agregat pada serpihan individu sebelum menggabungkannya ke dalam aliran agregat, hasilnya mungkin ditimbang ke aliran mana pun yang berisi lebih banyak catatan.
- Jika aplikasi Anda terus mengalami performa yang buruk (direfleksikan dengan metrik `MillisBehindLatest` tinggi) setelah meningkatkan jumlah aliran input, Anda mungkin sudah mencapai batas Unit Pemrosesan Kinesis (KPU). Lihat informasi yang lebih lengkap di Secara Otomatis Menskalakan Aplikasi untuk Meningkatkan Throughput.

Kode Aplikasi

Kode aplikasi adalah serangkaian pernyataan SQL yang memproses input dan menghasilkan output. Pernyataan SQL ini beroperasi pada aliran dalam aplikasi dan tabel referensi. Untuk informasi selengkapnya, lihat Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya.

Untuk informasi tentang elemen bahasa SQL yang didukung oleh Kinesis Data Analytics, lihat Refererensi SQL Amazon Kinesis Data Analytics.

Dalam basis data relasional, Anda bekerja dengan tabel, menggunakan pernyataan INSERT untuk menambahkan catatan dan pernyataan SELECT untuk mengkueri data. Di Amazon Kinesis Data Analytics, Anda bekerja dengan aliran. Anda dapat menulis pernyataan SQL untuk mengkueri aliran ini. Hasil mengkueri satu aliran dalam aplikasi selalu dikirim ke aliran dalam aplikasi lainnya. Saat melakukan analitik yang kompleks, Anda dapat membuat beberapa aliran dalam aplikasi untuk menyimpan hasil analitik perantara. Akhirnya, Anda mengonfigurasi output aplikasi untuk meneruskan hasil analisis akhir (dari satu atau beberapa aliran dalam aplikasi) ke tujuan eksternal. Singkatnya, berikut adalah pola khas untuk menulis kode aplikasi:

- Pernyataan SELECT selalu digunakan dalam konteks pernyataan INSERT. Artinya, ketika Anda memilih baris, Anda menyisipkan hasil ke aliran dalam aplikasi lainnya.
- Pernyataan INSERT selalu digunakan dalam konteks pompa. Artinya, Anda menggunakan pompa untuk menulis ke aliran dalam aplikasi.

Kode aplikasi contoh berikut membaca catatan dari satu aliran (SOURCE_SQL_STREAM_001) dalam aplikasi dan menulis ke aliran dalam aplikasi lain (DESTINATION_SQL_STREAM). Anda dapat menyisipkan catatan ke aliran dalam aplikasi menggunakan pompa, seperti yang ditunjukkan di bawah ini:

```
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**
>
> Pengidentifikasi yang Anda tentukan untuk nama aliran dan nama kolom mengikuti konvensi SQL standar. Misalnya, jika Anda menempatkan tanda kutip di sekitar pengindentifikasi, hal itu menjadikan pengidentifikasi peka huruf besar/kecil. Jika tidak, pengindentifikasi default ke

huruf besar. Untuk informasi selengkapnya tentang pengidentifikasi, lihat Pengidentifikasi di Amazon Managed Service for Apache Flink SQL Reference.

Kode aplikasi Anda dapat terdiri dari banyak pernyataan SQL. Misalnya:

- Anda dapat menulis kueri SQL secara berurutan dengan hasil dari satu pernyataan SQL mengumpan ke pernyataan SQL berikutnya.
- Anda juga dapat menulis kueri SQL yang berjalan independen satu sama lain. Misalnya, Anda dapat menulis dua pernyataan SQL yang mengkueri aliran dalam aplikasi yang sama, tetapi mengirim output ke aliran dalam aplikasi yang berbeda. Anda selanjutnya dapat mengkueri aliran dalam aplikasi yang baru dibuat secara independen.

Anda dapat membuat aliran dalam aplikasi untuk menyimpan hasil perantara. Anda memasukkan data di aliran dalam aplikasi menggunakan pompa. Untuk informasi selengkapnya, lihat Aliran dan Pompa dalam Aplikasi.

Jika Anda menambahkan tabel referensi dalam aplikasi, Anda dapat menulis SQL untuk menggabungkan data di aliran dalam aplikasi dan tabel referensi. Untuk informasi selengkapnya, lihat Contoh: Menambahkan Data Referensi ke Aplikasi Kinesis Data Analytics.

Berdasarkan konfigurasi output aplikasi, Amazon Kinesis Data Analytics menulis data dari aliran dalam aplikasi tertentu ke tujuan eksternal sesuai dengan konfigurasi output aplikasi. Pastikan kode aplikasi Anda menulis ke aliran dalam aplikasi yang ditentukan dalam konfigurasi output.

Untuk informasi lain, lihat topik berikut:

- Konsep SQL Streaming
- Referensi SQL Amazon Kinesis Data Analytics

Mengonfigurasi Output Aplikasi

Dalam kode aplikasi Anda, Anda menulis output pernyataan SQL untuk satu atau lebih aliran dalam aplikasi. Anda dapat menambahkan konfigurasi output ke aplikasi Anda. untuk menyimpan semua yang ditulis ke aliran dalam aplikasi ke tujuan eksternal seperti aliran data Amazon Kinesis, aliran pengiriman Firehose, atau fungsi. AWS Lambda

Ada batas pada jumlah tujuan eksternal yang dapat Anda gunakan untuk menyimpan output aplikasi. Untuk informasi selengkapnya, lihat Batas.

> **Note**
>
> Sebaiknya gunakan satu tujuan eksternal untuk menyimpan data aliran kesalahan dalam aplikasi agar Anda dapat menyelidiki kesalahan.

Dalam setiap konfigurasi output ini, Anda memberikan hal berikut:

- Nama aliran dalam aplikasi – Aliran yang ingin Anda simpan ke tujuan eksternal.

  Kinesis Data Analytics mencari aliran dalam aplikasi yang Anda tentukan dalam konfigurasi output. (Nama aliran adalah peka huruf besar/kecil dan harus sama persis.) Pastikan kode aplikasi Anda membuat aliran dalam aplikasi ini.
- Tujuan eksternal — Anda dapat menyimpan data ke aliran data Kinesis, aliran pengiriman Firehose, atau fungsi Lambda. Anda menyediakan Amazon Resource Name (ARN) aliran atau fungsi. Anda juga menyediakan IAM role yang dapat diambil Kinesis Data Analytics untuk menulis ke aliran atau fungsi atas nama Anda. Anda menjelaskan format catatan (JSON, CSV) ke Kinesis Data Analytics yang digunakan saat menulis ke tujuan eksternal.

Jika Kinesis Data Analytics tidak dapat menulis ke tujuan streaming atau Lambda, layanan terus mencoba tanpa batas. Ini membuat tekanan balik, yang menyebabkan aplikasi Anda tertinggal. Jika masalah ini tidak teratasi, aplikasi Anda akhirnya berhenti memproses data baru. Anda dapat memantau Metrik Kinesis Data Analytics dan mengatur alarm untuk kegagalan. Untuk informasi selengkapnya tentang metrik dan alarm, lihat Menggunakan CloudWatchMetrik Amazon dan Membuat Alarm Amazon. CloudWatch

Anda dapat mengonfigurasi output aplikasi menggunakan AWS Management Console. Konsol membuat panggilan API untuk menyimpan konfigurasi.

Membuat Output Menggunakan AWS CLI

Bagian ini menjelaskan cara membuat bagian `Outputs` dari isi permintaan untuk operasi `CreateApplication` atau `AddApplicationOutput`.

Membuat Output Aliran Kinesis

Fragmen JSON berikut menunjukkan bagian `Outputs` di isi permintaan `CreateApplication` untuk membuat tujuan Amazon Kinesis data stream.

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

]
```

Membuat Output Aliran Pengiriman Firehose

Fragmen JSON berikut menunjukkan `Outputs` bagian di badan `CreateApplication` permintaan untuk membuat tujuan aliran pengiriman Amazon Data Firehose.

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

Membuat Output Fungsi Lambda

Fragmen JSON berikut menunjukkan `Outputs` bagian dalam badan `CreateApplication` permintaan untuk membuat tujuan AWS Lambda fungsi.

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

Menggunakan Fungsi Lambda sebagai Output

Menggunakan AWS Lambda sebagai tujuan memungkinkan Anda untuk lebih mudah melakukan pasca-pemrosesan hasil SQL Anda sebelum mengirimnya ke tujuan akhir. Tugas pasca-pemrosesan umum meliputi berikut ini:

- Menggabungkan beberapa baris ke dalam satu catatan
- Menggabungkan hasil saat ini dengan hasil sebelumnya untuk mengatasi data yang datang terlambat
- Mengirimkan ke berbagai tujuan berdasarkan tipe informasi
- Terjemahan format catatan (seperti menerjemahkan ke Protobuf)
- Manipulasi atau transformasi string
- Pengayaan data setelah pemrosesan analitik
- Pemrosesan khusus untuk kasus penggunaan geospasial
- Enkripsi data

Fungsi Lambda dapat memberikan informasi analitik ke berbagai AWS layanan dan tujuan lainnya, termasuk yang berikut:

- Amazon Simple Storage Service (Amazon S3)
- API Khusus
- Amazon DynamoDB
- Aurora Apache

- Amazon Redshift
- Layanan Pemberitahuan Sederhana Amazon (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon CloudWatch

Untuk informasi selengkapnya tentang membuat aplikasi Lambda, lihat Memulai dengan AWS Lambda.

Topik

- Lambda sebagai Izin Output
- Lambda sebagai Metrik Output
- Lambda sebagai Model Data Input Peristiwa Output dan Model Respons Catatan
- Frekuensi Invokasi Output Lambda
- Menambahkan Fungsi Lambda untuk Digunakan sebagai Output
- Lambda umum sebagai Kegagalan Output
- Membuat Fungsi Lambda untuk Tujuan Aplikasi

Lambda sebagai Izin Output

Untuk menggunakan Lambda sebagai output, IAM role output Lambda aplikasi memerlukan kebijakan izin berikut:

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

Lambda sebagai Metrik Output

Anda menggunakan Amazon CloudWatch untuk memantau jumlah byte yang dikirim, keberhasilan dan kegagalan, dan sebagainya. Untuk informasi tentang CloudWatch metrik yang dipancarkan oleh Kinesis Data Analytics menggunakan Lambda sebagai output, lihat Metrik Amazon Kinesis Analytics.

Lambda sebagai Model Data Input Peristiwa Output dan Model Respons Catatan

Untuk mengirim catatan output Kinesis Data Analytics, fungsi Lambda Anda harus sesuai dengan data input peristiwa yang diperlukan dan model respons catatan.

Model Data Input Peristiwa

Kinesis Data Analytics terus mengirimkan catatan output dari aplikasi ke Lambda sebagai fungsi output dengan model permintaan berikut. Dalam fungsi Anda, Anda mengulangi melalui daftar dan menerapkan logika bisnis Anda untuk memenuhi persyaratan output Anda (seperti transformasi data sebelum mengirim ke tujuan akhir).

Bidang	Deskripsi
`invocationId`	ID invokasi Lambda (GUID acak).
`applicationArn`	Amazon Resource Name (ARN) aplikasi Kinesis Data Analytics.
catatan	

Bidang	Deskripsi
`recordId`	ID catatan (GUID acak)
`lambdaDeliveryRecordMetadata`	

Bidang	Deskripsi
`retryHir`	Jumlah coba lagi pengiriman

Bidang	Deskripsi
data	Muatan catatan output terenkode Base64

> **Note**
>
> `retryHint` adalah nilai yang meningkat untuk setiap kegagalan pengiriman. Nilai ini tidak bertahan lama, dan menyetel ulang jika aplikasi terganggu.

Model Respons Catatan

Setiap catatan yang dikirim ke Lambda Anda sebagai fungsi output (dengan ID catatan) harus dibenarkan dengan `Ok` atau `DeliveryFailed`, dan harus berisi parameter berikut. Jika tidak, Kinesis Data Analytics menganggapnya sebagai kegagalan pengiriman.

catatan

Bidang	Deskripsi
`recordId`	ID catatan diteruskan dari Kinesis Data Analytics ke Lambda selama invokasi. Ketidakcocokan apa pun antara ID catatan asli dan ID catatan yang dibenarkan dianggap sebagai kegagalan pengiriman.
`result`	Status pengiriman catatan. Berikut adalah nilai yang mungkin: • `Ok`: Catatan berhasil diubah dan dikirim ke tujuan akhir. Kinesis Data Analytics menyerap catatan untuk pemrosesan SQL. • `DeliveryFailed` : Catatan gagal dikirim ke tujuan akhir oleh Lambda sebagai fungsi output. Kinesis Data Analytics terus mencoba mengirimkan catatan pengiriman yang gagal ke Lambda sebagai fungsi output.

Frekuensi Invokasi Output Lambda

Aplikasi Kinesis Data Analytics menyangga catatan output dan sering memanggil fungsi tujuan AWS Lambda .

- Jika catatan dipancarkan ke aliran dalam aplikasi tujuan dalam aplikasi analitik data sebagai jendela jatuh, fungsi AWS Lambda tujuan dipanggil per pemicu jendela jatuh. Misalnya, jika jendela tumbling senilai 60 detik digunakan untuk memancarkan catatan ke aliran dalam aplikasi tujuan, fungsi Lambda dipanggil sekali setiap 60 detik.
- Jika catatan dipancarkan ke aliran dalam aplikasi tujuan di aplikasi sebagai kueri berkelanjutan atau jendela geser, fungsi tujuan Lambda dipanggil sekitar sekali per detik.

> **Note**
>
> Fungsi per Lambda memanggil batas ukuran muatan permintaan berlaku. Melebihi batas tersebut menghasilkan catatan output yang dibagi dan dikirim di beberapa panggilan fungsi Lambda.

Menambahkan Fungsi Lambda untuk Digunakan sebagai Output

Prosedur berikut menunjukkan cara menambahkan fungsi Lambda sebagai output untuk aplikasi Kinesis Data Analytics.

1. Masuk ke AWS Management Console dan buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih aplikasi dalam daftar, lalu pilih Application details (Detail aplikasi).
3. Di bagian Destination (Tujuan), pilih Connect new destination (Sambungkan tujuan baru).
4. Untuk item Destination (Tujuan), pilih fungsiAWS Lambda .
5. Di bagian Kirim catatan ke AWS Lambda, pilih fungsi atau versi Lambda yang sudah ada, atau pilih Create new (Buat baru).
6. Jika Anda membuat fungsi Lambda baru, lakukan hal berikut:

    a. Pilih salah satu templat yang disediakan. Untuk informasi selengkapnya, Membuat Fungsi Lambda untuk Tujuan Aplikasi.

    b. Halaman Buat Fungsi terbuka di tab peramban baru. Di kotak Name (Nama), beri nama yang bermakna untuk fungsi (misalnya, **`myLambdaFunction`**).

    c. Perbarui templat dengan fungsi pasca-pemrosesan untuk aplikasi Anda. Untuk informasi tentang cara membuat fungsi Lambda, lihat Memulai dalam Panduan DeveloperAWS Lambda .

d. Di konsol Kinesis Data Analytics, dalam daftar Fungsi Lambda, pilih fungsi Lambda yang baru saja Anda buat. Pilih $LATEST untuk versi fungsi Lambda.

7. Di bagian In-application stream (Aliran dalam aplikasi), pilih Choose an existing in-application stream (Pilih aliran dalam aplikasi yang sudah ada). Untuk Nama aliran dalam aplikasi, pilih aliran output aplikasi Anda. Hasil dari aliran output yang dipilih dikirim ke fungsi output Lambda.
8. Biarkan sisa formulir dengan nilai default, dan pilih Save and continue (Simpan dan lanjutkan).

Aplikasi Anda sekarang mengirimkan catatan dari aliran dalam aplikasi ke fungsi Lambda Anda. Anda dapat melihat hasil template default di CloudWatch konsol Amazon. Pantau metrik `AWS/KinesisAnalytics/LambdaDelivery.OkRecords` untuk melihat jumlah catatan yang dikirim ke fungsi Lambda.

Lambda umum sebagai Kegagalan Output

Berikut adalah alasan umum pengiriman ke fungsi Lambda dapat gagal.

- Tidak semua catatan (dengan ID catatan) dalam batch yang dikirim ke fungsi Lambda dikembalikan ke layanan Kinesis Data Analytics.
- Respons kehilangan ID catatan atau bidang status.
- Waktu habis fungsi Lambda tidak cukup untuk mencapai logika bisnis dalam fungsi Lambda.
- Logika bisnis dalam fungsi Lambda tidak menangkap semua kesalahan, yang mengakibatkan waktu habis dan tekanan balik karena pengecualian yang tidak ditangani. Ini sering disebut sebagai pesan “pil beracun”.

Untuk kegagalan pengiriman data, Kinesis Data Analytics terus mencoba kembali invokasi Lambda pada kumpulan catatan yang sama hingga berhasil. Untuk mendapatkan wawasan tentang kegagalan, Anda dapat memantau CloudWatch metrik berikut:

- Aplikasi Kinesis Data Analytics Lambda sebagai metrik CloudWatch Output: Menunjukkan jumlah keberhasilan dan kegagalan, di antara statistik lainnya. Untuk informasi selengkapnya, lihat Metrik Amazon Kinesis Analytics.
- AWS Lambda CloudWatch metrik fungsi dan log.

Membuat Fungsi Lambda untuk Tujuan Aplikasi

Aplikasi Kinesis Data Analytics Anda dapat AWS Lambda menggunakan fungsi sebagai output. Kinesis Data Analytics menyediakan templat untuk membuat fungsi Lambda yang digunakan sebagai tujuan aplikasi Anda. Gunakan templat ini sebagai titik awal untuk pasca-pemrosesan output dari aplikasi Anda.

Topik

- Membuat Tujuan Fungsi Lambda di Node.js
- Membuat Tujuan Fungsi Lambda di Python
- Membuat Tujuan Fungsi Lambda di Java
- Membuat Tujuan Fungsi Lambda di .NET

Membuat Tujuan Fungsi Lambda di Node.js

Templat berikut untuk membuat fungsi Lambda tujuan di Node.js tersedia di konsol:

Lambda sebagai Cetak Biru Output	Bahasa dan Versi	Deskripsi
`kinesis-analytics-output`	Node.js 12.x	Kirimkan catatan output dari aplikasi Kinesis Data Analytics ke tujuan khusus.

Membuat Tujuan Fungsi Lambda di Python

Templat berikut untuk membuat fungsi Lambda tujuan di Phyton tersedia di konsol:

Lambda sebagai Cetak Biru Output	Bahasa dan Versi	Deskripsi
`kinesis-analytics-output-sns`	Python 2.7	Kirimkan catatan output dari aplikasi Kinesis Data Analytics ke Amazon SNS.

Lambda sebagai Cetak Biru Output	Bahasa dan Versi	Deskripsi
`kinesis-analytics-output-ddb`	Python 2.7	Kirimkan catatan output dari aplikasi Kinesis Data Analytics ke Amazon DynamoDB.

Membuat Tujuan Fungsi Lambda di Java

Untuk membuat fungsi Lambda tujuan di Java, gunakan kelas peristiwa Java.

Kode berikut menunjukkan fungsi Lambda tujuan sampel menggunakan 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;
    }
```

```
}
```

Membuat Tujuan Fungsi Lambda di .NET

Untuk membuat fungsi Lambda tujuan di .NET, gunakan kelas peristiwa .NET.

Kode berikut menunjukkan fungsi Lambda tujuan sampel menggunakan 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;
        }
    }
```

Untuk informasi selengkapnya tentang membuat fungsi Lambda untuk pra-pemrosesan dan tujuan di .NET, lihat `Amazon.Lambda.KinesisAnalyticsEvents`.

Model Pengiriman untuk Menyimpan Output Aplikasi untuk Tujuan Eksternal

Amazon Kinesis Data Analytics menggunakan model pengiriman "setidaknya sekali" untuk output aplikasi ke tujuan yang dikonfigurasi. Saat aplikasi berjalan, Kinesis Data Analytics mengambil titik pemeriksaan internal. Titik pemeriksaan ini adalah titik dalam waktu ketika catatan output dikirim ke tujuan tanpa kehilangan data. Layanan menggunakan titik pemeriksaan yang diperlukan untuk memastikan output aplikasi Anda dikirim setidaknya sekali ke tujuan yang dikonfigurasi.

Dalam situasi normal, aplikasi Anda memproses data yang masuk terus-menerus. Kinesis Data Analytics menulis output ke tujuan yang dikonfigurasi, seperti aliran data Kinesis atau aliran pengiriman Firehose. Namun, aplikasi Anda sesekali dapat terganggu, misalnya:

- Anda memilih untuk menghentikan aplikasi Anda dan memulai ulang nanti.
- Anda menghapus IAM role yang diperlukan oleh Kinesis Data Analytics untuk menulis output aplikasi Anda ke tujuan yang dikonfigurasi. Tanpa IAM role, Kinesis Data Analytics tidak memiliki izin untuk menulis ke tujuan eksternal atas nama Anda.
- Penghentian jaringan atau kegagalan layanan internal lainnya menyebabkan aplikasi Anda berhenti berjalan sejenak.

Saat aplikasi dimulai ulang, Kinesis Data Analytics memastikan aplikasi terus memproses dan menulis output dari titik sebelum atau sama dengan saat terjadinya kegagalan. Ini membantu memastikan Kinesis Data Analytics tidak melewatkan pengiriman output aplikasi apa pun ke tujuan yang dikonfigurasi.

Misalkan Anda mengonfigurasi beberapa tujuan dari aliran dalam aplikasi yang sama. Setelah aplikasi pulih dari kegagalan, Kinesis Data Analytics melanjutkan penyimpanan output ke tujuan yang dikonfigurasi dari catatan terakhir yang dikirim ke tujuan paling lambat. Ini dapat mengakibatkan catatan output yang sama dikirimkan lebih dari sekali ke tujuan lain. Dalam hal ini, Anda harus menangani duplikasi potensial di tujuan secara eksternal.

Penanganan Kesalahan

Amazon Kinesis Data Analytics mengembalikan kesalahan API atau SQL langsung kepada Anda. Untuk informasi selengkapnya tentang operasi API, lihatTindakan. Untuk informasi selengkapnya tentang menangani kesalahan SQL, lihat Referensi SQL Amazon Kinesis Data Analytics.

Amazon Kinesis Data Analytics melaporkan kesalahan runtime menggunakan aliran kesalahan dalam aplikasi yang disebut `error_stream`.

Melaporkan Kesalahan Menggunakan Aliran Kesalahan dalam Aplikasi

Amazon Kinesis Data Analytics melaporkan kesalahan runtime ke aliran kesalahan dalam aplikasi yang disebut `error_stream`. Berikut adalah contoh kesalahan yang mungkin terjadi:

- Catatan yang dibaca dari sumber streaming tidak sesuai dengan skema input.
- Kode aplikasi Anda menentukan pembagian dengan nol.
- Baris rusak (misalnya, catatan muncul di aliran dengan nilai `ROWTIME` yang diubah pengguna yang menyebabkan catatan keluar dari urutan).
- Data dalam aliran sumber tidak dapat dikonversi ke tipe data yang ditentukan dalam skema (Kesalahan paksaan). Untuk informasi tentang tipe data yang dapat dikonversi, lihat Memetakan Tipe Data JSON ke Tipe Data SQL.

Sebaiknya tangani kesalahan ini secara terprogram dalam kode SQL Anda atau simpan data pada aliran kesalahan ke tujuan eksternal. Anda diharuskan menambahkan konfigurasi output (lihat Mengonfigurasi Output Aplikasi) ke aplikasi Anda. Untuk contoh cara kerja aliran kesalahan dalam aplikasi, lihat Contoh: Menjelajahi Aliran Kesalahan dalam Aplikasi.

> **Note**
>
> Aplikasi Kinesis Data Analytics Anda tidak dapat mengakses atau memodifikasi aliran kesalahan secara terprogram karena aliran kesalahan dibuat menggunakan akun sistem. Anda harus menggunakan output kesalahan untuk menentukan jenis kesalahan yang mungkin dihadapi aplikasi Anda. Anda selanjutnya menulis kode SQL aplikasi Anda untuk menangani kondisi kesalahan yang diantisipasi.

Skema

Aliran kesalahan ini memiliki skema berikut:

Bidang	Tipe Data	Catatan
ERROR_TIME	TIMESTAMP	Waktu terjadinya kesalahan

ERROR_LEVEL	VARCHAR(10)	
ERROR_NAME	VARCHAR(32)	
MESSAGE	VARCHAR(4096)	
DATA_ROWTIME	TIMESTAMP	Waktu baris catatan yang masuk
DATA_ROW	VARCHAR(49152)	Data yang dikodekan hex di baris asli. Anda dapat menggunakan pustaka standar untuk melakukan dekode hex nilai ini, atau menggunakan sumber daya web seperti Konverter Hex untuk String.
PUMP_NAME	VARCHAR(128)	Pompa asal, seperti yang ditentukan dengan `CREATE PUMP`

Secara Otomatis Menskalakan Aplikasi untuk Meningkatkan Throughput

Amazon Kinesis Data Analytics secara elastis menskalakan aplikasi Anda untuk mengakomodasi throughput data dari aliran sumber dan kompleksitas kueri Anda untuk sebagian besar skenario. Kinesis Data Analytics menyediakan kapasitas berupa Unit Pemrosesan Kinesis (KPU). Satu KPU memberi Anda memori (4 GB) dan komputasi serta jaringan yang sesuai.

Batas default untuk KPU untuk aplikasi Anda adalah 64. Untuk petunjuk tentang cara meminta peningkatan batas ini, lihat Untuk meminta peningkatan batas di Amazon Service Limits.

Menggunakan Penandaan

Bagian ini menjelaskan cara menambahkan tanda metadata nilai kunci ke aplikasi Kinesis Data Analytics. Tanda ini dapat digunakan untuk tujuan berikut:

- Menentukan penagihan untuk aplikasi Kinesis Data Analytics individual. Untuk informasi selengkapnya, lihat Menggunakan Tanda Alokasi Biaya dalam Panduan Manajemen Penagihan dan Biaya AWS.
- Mengontrol akses ke sumber daya aplikasi berdasarkan tanda. Untuk informasi selengkapnya, lihat Mengontrol Akses Menggunakan Tanda di Panduan Pengguna .
- Tujuan yang ditentukan pengguna. Anda dapat menentukan fungsi aplikasi berdasarkan adanya tanda pengguna.

Catat informasi berikut tentang penandaan:

- Jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50.
- Jika tindakan berisi daftar tanda yang memiliki nilai `Key` duplikat, layanan melempar `InvalidArgumentException`.

Topik ini berisi bagian-bagian berikut:

- Menambahkan Tanda saat Aplikasi dibuat
- Menambahkan atau Memperbarui Tanda untuk Aplikasi yang Ada
- Mencantumkan Tanda untuk Aplikasi
- Menghapus Tanda dari Aplikasi

Menambahkan Tanda saat Aplikasi dibuat

Anda menambahkan tanda saat membuat aplikasi menggunakan`tags` parameter CreateApplicationtindakan.

Contoh permintaan berikut menunjukkan simpul `Tags` untuk permintaan `CreateApplication`:

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

```
    }
]
```

Menambahkan atau Memperbarui Tanda untuk Aplikasi yang Ada

Anda menambahkan tag ke aplikasi menggunakan TagResourcetindakan. Anda tidak dapat menambahkan tanda ke aplikasi menggunakan UpdateApplicationtindakan.

Untuk memperbarui tanda yang ada, tambahkan tanda dengan kunci yang sama dengan tanda yang ada.

Contoh permintaan berikut untuk tindakan `TagResource` menambahkan tanda baru atau memperbarui tanda yang ada:

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

Mencantumkan Tanda untuk Aplikasi

Untuk mencantumkan tanda yang ada, Anda menggunakan ListTagsForResourcetindakan.

Contoh permintaan berikut untuk tindakan `ListTagsForResource` mencantumkan tanda untuk aplikasi:

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

Menghapus Tanda dari Aplikasi

Untuk menghapus tanda dari aplikasi, Anda menggunakan UntagResourcetindakan.

Contoh permintaan berikut untuk tindakan `UntagResource` menghapus tanda dari aplikasi:

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

Memulai dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL

Setelah itu, Anda dapat menemukan topik untuk membantu Anda mulai menggunakan Amazon Kinesis Data Analytics untuk Aplikasi SQL. Jika Anda baru mengenal Kinesis Data Analytics untuk Aplikasi SQL, sebaiknya tinjau konsep dan terminologi yang disajikan di Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya sebelum melakukan langkah-langkah di bagian Memulai.

Topik

- Mendaftar untuk Akun AWS
- Buat pengguna dengan akses administratif
- Langkah 1: Siapkan Akun dan Buat Pengguna Administrator
- Mendaftar untuk Akun AWS
- Buat pengguna dengan akses administratif
- Langkah 2: Mengatur AWS Command Line Interface (AWS CLI)
- Langkah 3: Buat Aplikasi Amazon Kinesis Data Analytics Starter Anda
- Langkah 4 (Opsional) Edit Skema dan Kode SQL Menggunakan Konsol

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua AWS layanan dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Langkah 1: Siapkan Akun dan Buat Pengguna Administrator

Sebelum Anda menggunakan Amazon Kinesis Data Analytics untuk pertama kali, selesaikan tugas berikut:

1. Mendaftar untuk AWS
2. Membuat Pengguna IAM

Mendaftar untuk AWS

Ketika Anda mendaftar ke Amazon Web Services, Anda Akun AWS secara otomatis mendaftar untuk semua layanan AWS, termasuk Amazon Kinesis Data Analytics. Anda hanya membayar biaya layanan yang Anda gunakan.

Dengan Kinesis Data Analytics, Anda hanya membayar untuk sumber daya yang Anda gunakan. Jika Anda adalah AWS pelanggan baru, Anda dapat memulai dengan Kinesis Data Analytics secara gratis. Untuk informasi selengkapnya, lihat Tingkat Penggunaan Gratis AWS.

Jika Anda sudah memiliki Akun AWS, lompat ke tugas berikutnya. Jika Anda tidak memiliki Akun AWS, lakukan langkah-langkah dalam prosedur berikut untuk membuatnya.

Untuk membuat Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.

2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua AWS layanan dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Catat Akun AWS ID Anda karena Anda akan membutuhkannya untuk tugas berikutnya.

Membuat Pengguna IAM

Layanan di AWS, seperti Amazon Kinesis Data Analytics, mengharuskan Anda memberikan kredensi saat Anda mengaksesnya sehingga layanan dapat menentukan apakah Anda memiliki izin untuk mengakses sumber daya yang dimiliki oleh layanan tersebut. Konsol tersebut memerlukan kata sandi. Anda dapat membuat kunci akses bagi Anda Akun AWS untuk mengakses AWS CLI atau API. Namun, kami tidak menyarankan Anda mengakses AWS menggunakan kredensil untuk Anda. Akun AWS Sebagai gantinya, sebaiknya gunakan (IAM) AWS Identity and Access Management . Buat pengguna IAM, tambahkan pengguna tersebut ke grup IAM dengan izin administratif, lalu berikan izin administratif ke pengguna IAM yang sudah dibuat. Anda kemudian dapat mengakses AWS menggunakan URL khusus dan kredenal pengguna IAM itu.

Jika Anda mendaftar AWS, tetapi Anda belum membuat pengguna IAM untuk diri sendiri, Anda dapat membuatnya menggunakan konsol IAM.

Latihan Memulai dalam panduan ini mengasumsikan Anda memiliki pengguna (`adminuser`) dengan hak istimewa administrator. Ikuti prosedur untuk membuat `adminuser` di akun Anda.

Untuk membuat pengguna administrator dan masuk ke konsol tersebut

1. Buat pengguna administrator yang disebut `adminuser` di Akun AWS Anda. Untuk melihat instruksi, buka Membuat Grup Pengguna dan Administrator IAM Pertama Anda di Panduan Pengguna IAM.
2. Seorang pengguna dapat masuk ke AWS Management Console menggunakan URL khusus. Untuk informasi selengkapnya, Cara Pengguna Masuk ke Akun Anda dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang IAM, lihat berikut ini:

- AWS Identity and Access Management (IAM)
- Memulai
- Panduan Pengguna IAM

Langkah Selanjutnya

Langkah 2: Mengatur AWS Command Line Interface (AWS CLI)

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua AWS layanan dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Langkah 2: Mengatur AWS Command Line Interface (AWS CLI)

Ikuti langkah-langkah untuk mengunduh dan mengkonfigurasi AWS Command Line Interface (AWS CLI).

> **Important**
>
> Anda tidak AWS CLI perlu melakukan langkah-langkah dalam latihan Memulai. Namun, beberapa latihan dalam panduan ini menggunakan AWS CLI. Anda dapat melewati langkah ini dan pergi ke Langkah 3: Buat Aplikasi Amazon Kinesis Data Analytics Starter Anda, dan kemudian mengatur AWS CLI nanti ketika Anda membutuhkannya.

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk melakukannya, lihat topik berikut di Panduan Pengguna AWS Command Line Interface :

   - Mendapatkan Set Up dengan AWS Command Line Interface
   - Mengkonfigurasi AWS Command Line Interface

2. Tambahkan profil bernama untuk pengguna administrator di file AWS CLI konfigurasi. Anda menggunakan profil ini saat menjalankan AWS CLI perintah. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface .

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

   Untuk daftar yang tersedia Wilayah AWS, lihat Wilayah dan Titik Akhir di. Referensi Umum Amazon Web Services

3. Verifikasikan penyiapan dengan memasukkan perintah bantuan berikut pada prompt perintah.

```
aws help
```

Langkah Selanjutnya

Langkah 3: Buat Aplikasi Amazon Kinesis Data Analytics Starter Anda

Langkah 3: Buat Aplikasi Amazon Kinesis Data Analytics Starter Anda

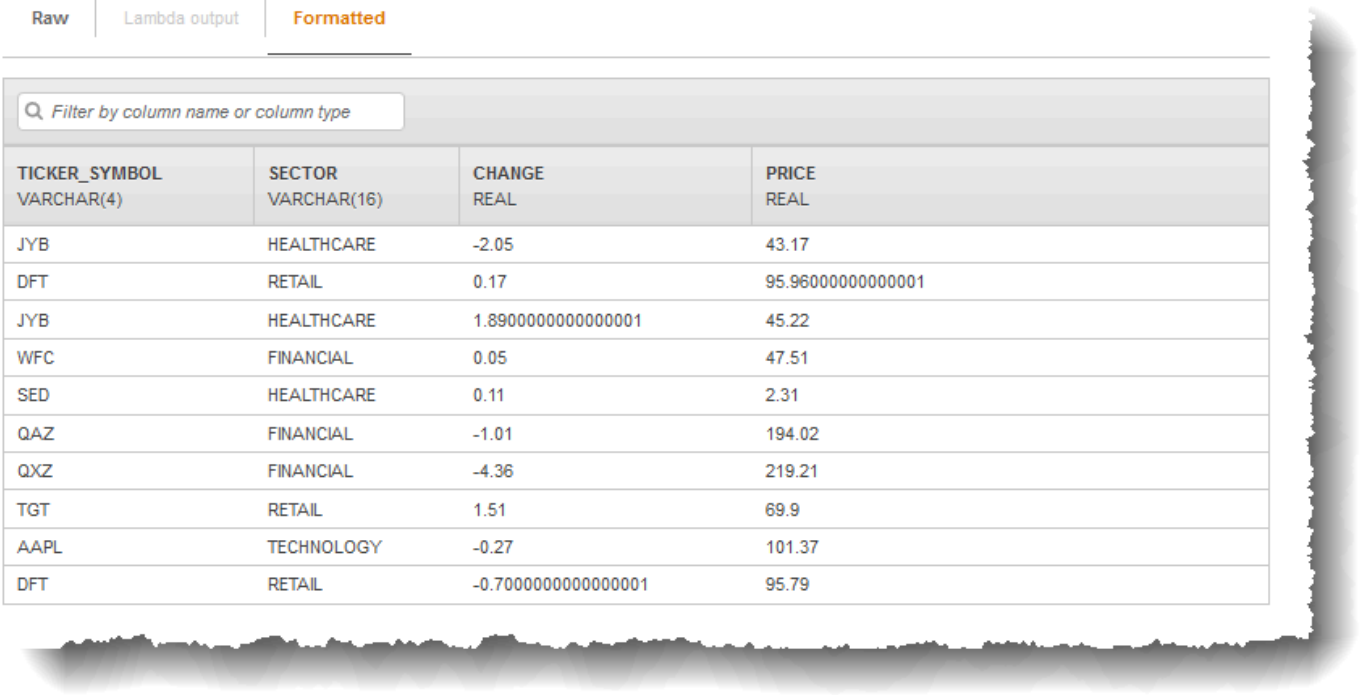
Dengan mengikuti langkah-langkah di bagian ini, Anda dapat membuat aplikasi Kinesis Data Analytics pertama menggunakan konsol.

> **Note**
>
> Sebaiknya tinjau Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya sebelum mencoba latihan Memulai.

Untuk latihan Memulai ini, Anda dapat menggunakan konsol untuk bekerja dengan aliran demo atau templat dengan kode aplikasi.

- Jika Anda memilih untuk menggunakan aliran demo, konsol membuat Kinesis data stream di akun Anda yang disebut `kinesis-analytics-demo-stream`.

  Aplikasi Kinesis Data Analytics membutuhkan sumber streaming. Untuk sumber ini, beberapa contoh SQL dalam panduan ini menggunakan aliran demo`kinesis-analytics-demo-stream`. Konsol juga menjalankan skrip yang terus menambahkan data sampel (simulasi catatan perdagangan saham) ke aliran ini, seperti yang ditunjukkan berikut ini.



Anda dapat menggunakan `kinesis-analytics-demo-stream` sebagai sumber streaming untuk aplikasi Anda dalam latihan ini.

> **Note**
>
> Aliran demo tetap ada di akun Anda. Anda dapat menggunakannya untuk menguji contoh lain dalam panduan ini. Namun, ketika Anda meninggalkan konsol, skrip yang digunakan konsol berhenti mengisi data. Jika diperlukan, konsol menyediakan opsi untuk mulai mengisi kembali aliran.

- Jika Anda memilih untuk menggunakan templat dengan contoh kode aplikasi, Anda menggunakan kode templat yang disediakan konsol untuk melakukan analitik sederhana pada aliran demo.

Anda menggunakan fitur ini untuk menyiapkan aplikasi pertama Anda dengan cepat sebagai berikut:

1. Create an application (Buat aplikasi) – Anda hanya perlu memberikan nama. Konsol membuat aplikasi dan layanan yang menetapkan status aplikasi ke `READY`.

2. Configure input (Konfigurasikan Input) – Pertama-tama, Anda menambahkan sumber streaming, aliran demo. Anda harus membuat aliran demo di konsol sebelum Anda dapat menggunakannya. Kemudian, konsol mengambil sampel acak catatan pada aliran demo dan menyimpulkan skema untuk aliran input dalam aplikasi yang dibuat. Konsol menamai aliran dalam aplikasi ini `SOURCE_SQL_STREAM_001`.

   Konsol menggunakan API penemuan untuk menyimpulkan skema. Jika perlu, Anda dapat mengedit skema yang disimpulkan. Untuk informasi selengkapnya, lihat DiscoverInputSchema. Kinesis Data Analytics menggunakan skema ini untuk membuat aliran dalam aplikasi.

   Saat Anda memulai aplikasi, Kinesis Data Analytics terus membaca aliran demo atas nama Anda dan memasukkan baris di aliran input dalam aplikasi `SOURCE_SQL_STREAM_001`.

3. Specify application code (Tentukan kode aplikasi) – Anda menggunakan templat (disebut Filter berkelanjutan) yang menyediakan kode berikut:

   ```
   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%';
   ```

   Kode aplikasi dapat mengkueri aliran dalam aplikasi `SOURCE_SQL_STREAM_001`. Kode kemudian memasukkan baris yang dihasilkan di aliran dalam aplikasi lainnya `DESTINATION_SQL_STREAM`, menggunakan pompa. Untuk informasi selengkapnya tentang pola pengkodean ini, lihat Kode Aplikasi.

   Untuk informasi tentang elemen bahasa SQL yang didukung oleh Kinesis Data Analytics, lihat Refererensi SQL Amazon Kinesis Data Analytics.

4. Mengonfigurasi output— Dalam latihan ini, Anda tidak mengonfigurasi output apa pun. Artinya, Anda tidak menyimpan data di aliran dalam aplikasi yang aplikasi Anda buat untuk tujuan eksternal apa pun. Sebaliknya, Anda memverifikasi hasil kueri di konsol. Contoh tambahan dalam panduan ini menunjukkan cara mengonfigurasi output. Sebagai contoh, lihat Contoh: Membuat Peringatan Sederhana.

> **Important**
>
> Latihan ini menggunakan Wilayah US East (N. Virginia) (us-east-1) untuk menyiapkan aplikasi. Anda dapat menggunakan salah satu yang didukung Wilayah AWS.

Langkah Selanjutnya

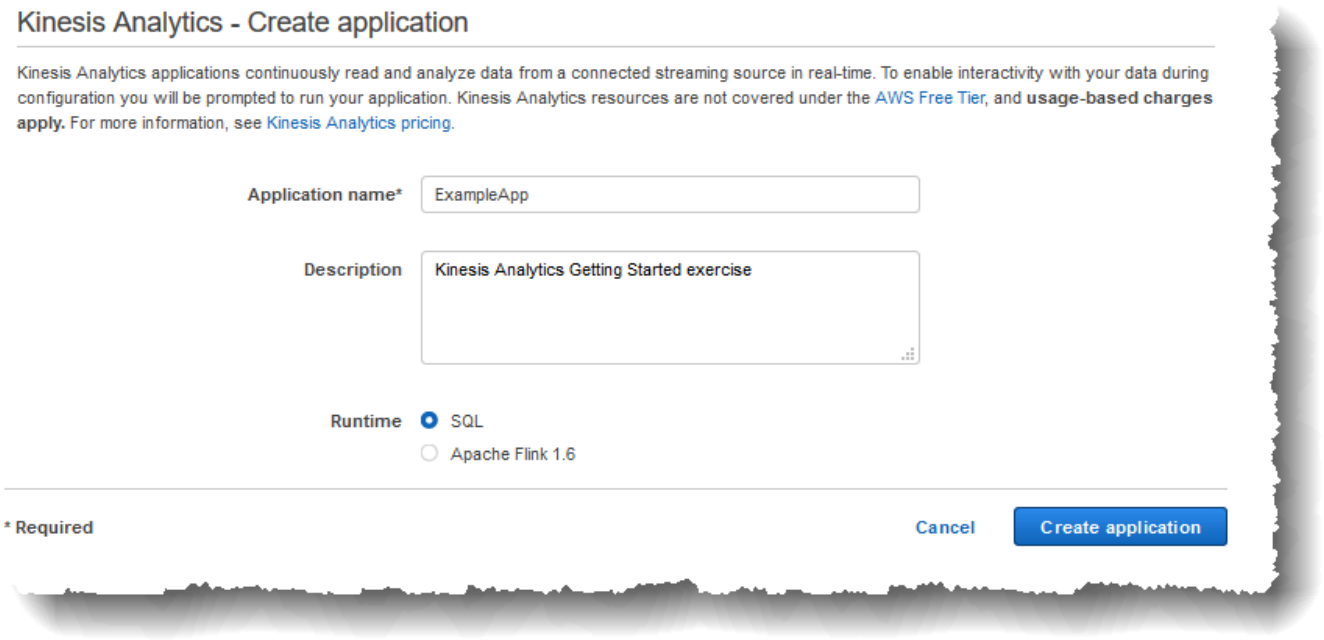
Langkah 3.1: Buat Aplikasi

Langkah 3.1: Buat Aplikasi

Di bagian ini, Anda membuat aplikasi Amazon Kinesis Data Analytics. Anda mengonfigurasi input aplikasi di langkah berikutnya.

Untuk membuat aplikasi analitik data

1. Masuk ke AWS Management Console dan buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi).
3. Di halaman Buat aplikasi, masukkan nama aplikasi, masukkan deskripsi, pilih SQL untuk pengaturan Runtime aplikasi, lalu pilih Create application (Buat aplikasi).



Melakukan hal ini akan membuat aplikasi Kinesis Data Analytics dengan status READY. Konsol menunjukkan hub aplikasi tempat Anda dapat mengonfigurasi input dan output.

> **Note**
>
> Untuk membuat aplikasi, operasi CreateApplication hanya memerlukan nama aplikasi. Anda dapat menambahkan konfigurasi input dan output setelah membuat aplikasi di konsol.

Di langkah berikutnya, Anda mengonfigurasi input untuk aplikasi. Dalam konfigurasi input, Anda menambahkan sumber data streaming ke aplikasi dan menemukan skema untuk aliran input dalam aplikasi dengan mengambil sampel data pada sumber streaming.

Langkah Selanjutnya

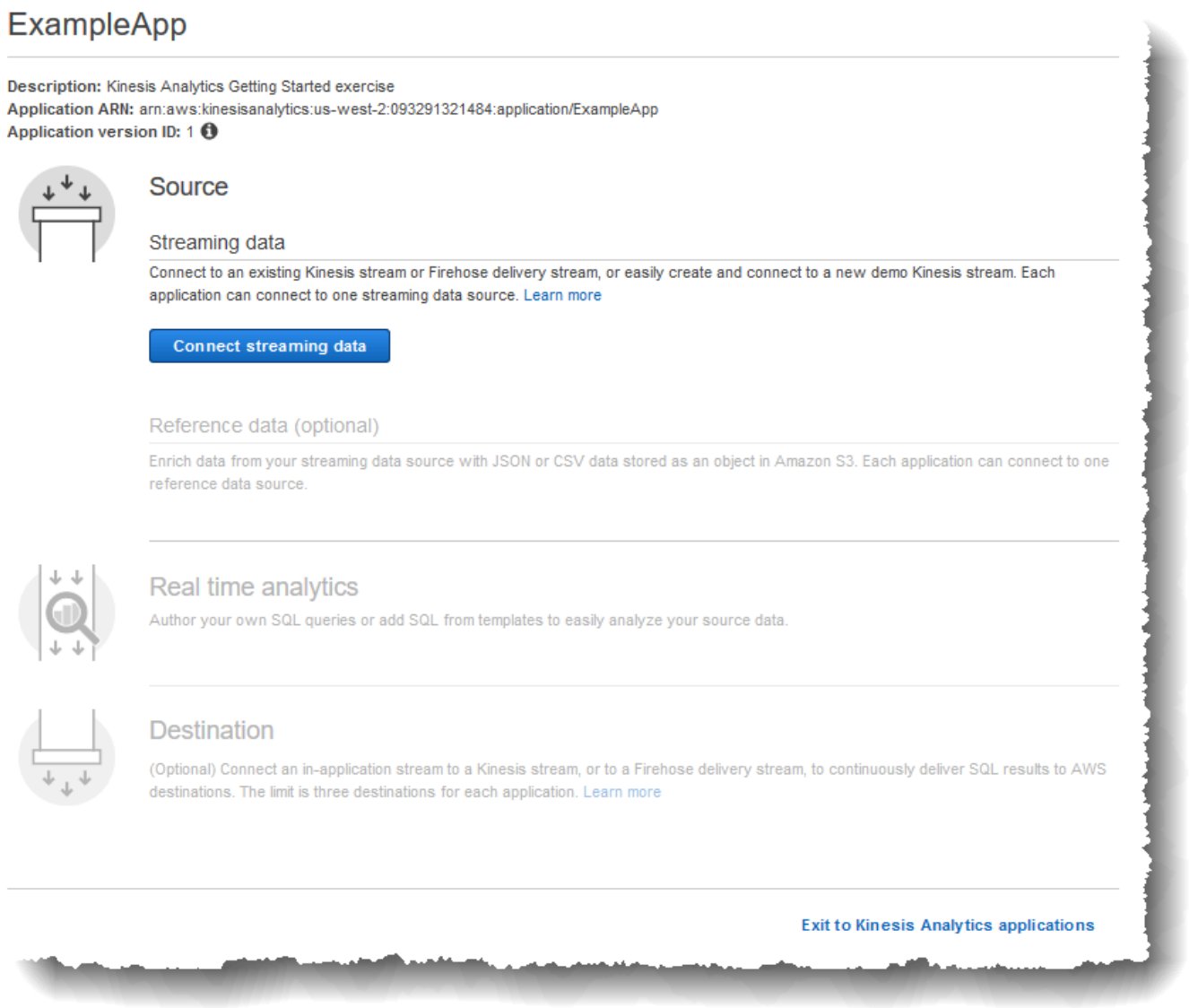
Langkah 3.2: Konfigurasikan Input

Langkah 3.2: Konfigurasikan Input

Aplikasi Anda membutuhkan sumber streaming. Untuk membantu Anda memulai, konsol dapat membuat aliran demo (disebut `kinesis-analytics-demo-stream`). Konsol juga menjalankan skrip yang mengisi catatan di aliran.

Untuk menambahkan sumber streaming ke aplikasi Anda

1. Di halaman hub aplikasi di konsol, pilih Connect data streaming (Sambungkan data streaming).



2. Di halaman yang muncul, tinjau hal berikut:

- Bagian sumber, tempat Anda menentukan sumber streaming untuk aplikasi Anda. Anda dapat memilih sumber aliran yang ada atau membuat sumber aliran. Dalam latihan ini, Anda membuat aliran baru, aliran demo.

  Secara default, konsol menamai aliran input dalam aplikasi yang dibuat sebagai `INPUT_SQL_STREAM_001`. Untuk latihan ini, simpan nama ini seperti yang terlihat.

  - Nama referensi aliran – Opsi ini menunjukkan nama aliran input dalam aplikasi yang dibuat, `SOURCE_SQL_STREAM_001`. Anda bisa mengganti namanya, tetapi untuk latihan ini, simpan nama ini.

    Dalam konfigurasi input, Anda memetakan aliran demo ke aliran input dalam aplikasi yang dibuat. Saat Anda memulai aplikasi, Amazon Kinesis Data Analytics terus membaca aliran demo dan memasukkan baris di aliran input dalam aplikasi. Anda mengkueri aliran input dalam aplikasi di kode aplikasi Anda.

  - Rekam pra-pemrosesan dengan AWS Lambda: Opsi ini adalah tempat Anda menentukan AWS Lambda ekspresi yang memodifikasi catatan dalam aliran input sebelum kode aplikasi Anda dijalankan. Dalam latihan ini, biarkan opsi Disabled (Nonaktif) dipilih. Untuk informasi selengkapnya tentang prapemrosesan Lambda, lihat Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda.

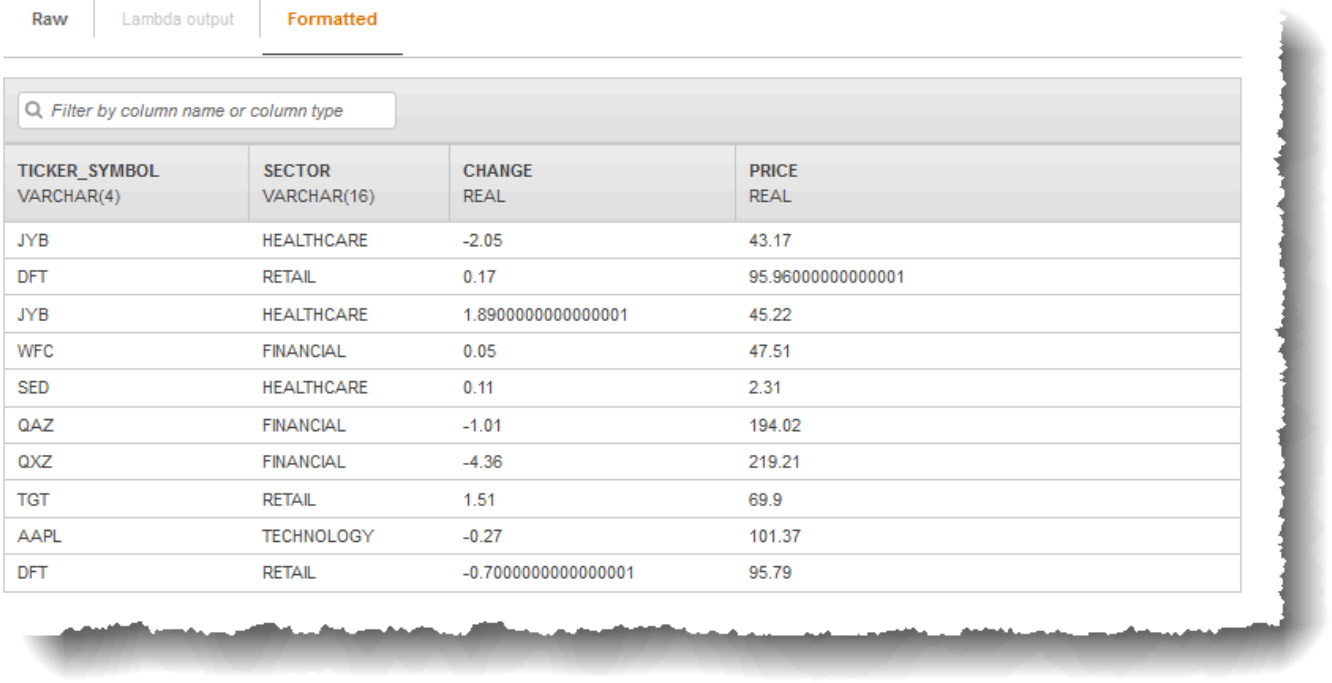
  Setelah Anda memberikan semua informasi di halaman ini, konsol akan mengirimkan permintaan pembaruan (lihat UpdateApplication) untuk menambahkan konfigurasi input aplikasi.

3. Di halaman Sumber, pilih Configure a new stream (Konfigurasikan aliran baru).
4. Pilih Create demo stream (Buat aliran demo). Konsol mengonfigurasi input aplikasi dengan melakukan hal berikut:

   - Konsol ini membuat Kinesis data stream yang disebut `kinesis-analytics-demo-stream`.
   - Konsol mengisi aliran dengan data ticker stok sampel.

- Dengan menggunakan tindakan input DiscoverInputSchema, konsol menyimpulkan skema dengan membaca catatan sampel di aliran. Skema yang disimpulkan adalah skema untuk aliran input dalam aplikasi yang dibuat. Untuk informasi selengkapnya, lihat Mengonfigurasi Input Aplikasi.
- Konsol menunjukkan skema yang disimpulkan dan data sampel yang dibaca dari sumber streaming untuk menyimpulkan skema.

Konsol menampilkan catatan sampel pada sumber streaming.



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

Beriku ini muncul di halaman konsol Sampel aliran:

- Tab Raw stream sample (Sampel aliran mentah) menunjukkan catatan aliran mentah yang sampelnya diambil oleh tindakan API DiscoverInputSchema untuk menyimpulkan skema.
- Tab Formatted stream sample (Sampel aliran yang diformat) menunjukkan versi tabel dari data di tab Raw stream sample (Sampel aliran mentah).
- Jika Anda memilih Edit schema (Edir skema), Anda dapat mengedit skema yang disimpulkan. Untuk latihan ini, jangan ubah skema yang disimpulkan. Untuk informasi selengkapnya tentang mengedit skema, lihat Menggunakan Editor Skema.

Jika Anda memilih Rediscover schema(Temukan kembali skema), Anda dapat meminta konsol untuk menjalankan DiscoverInputSchema kembali dan menyimpulkan skema.

5. Jangan pilih Save and continue (Simpan dan lanjutkan).

   Anda sekarang memiliki aplikasi dengan konfigurasi input yang ditambahkan ke dalamnya. Di langkah berikutnya, Anda menambahkan kode SQL untuk melakukan beberapa analitik pada data aliran input dalam aplikasi.

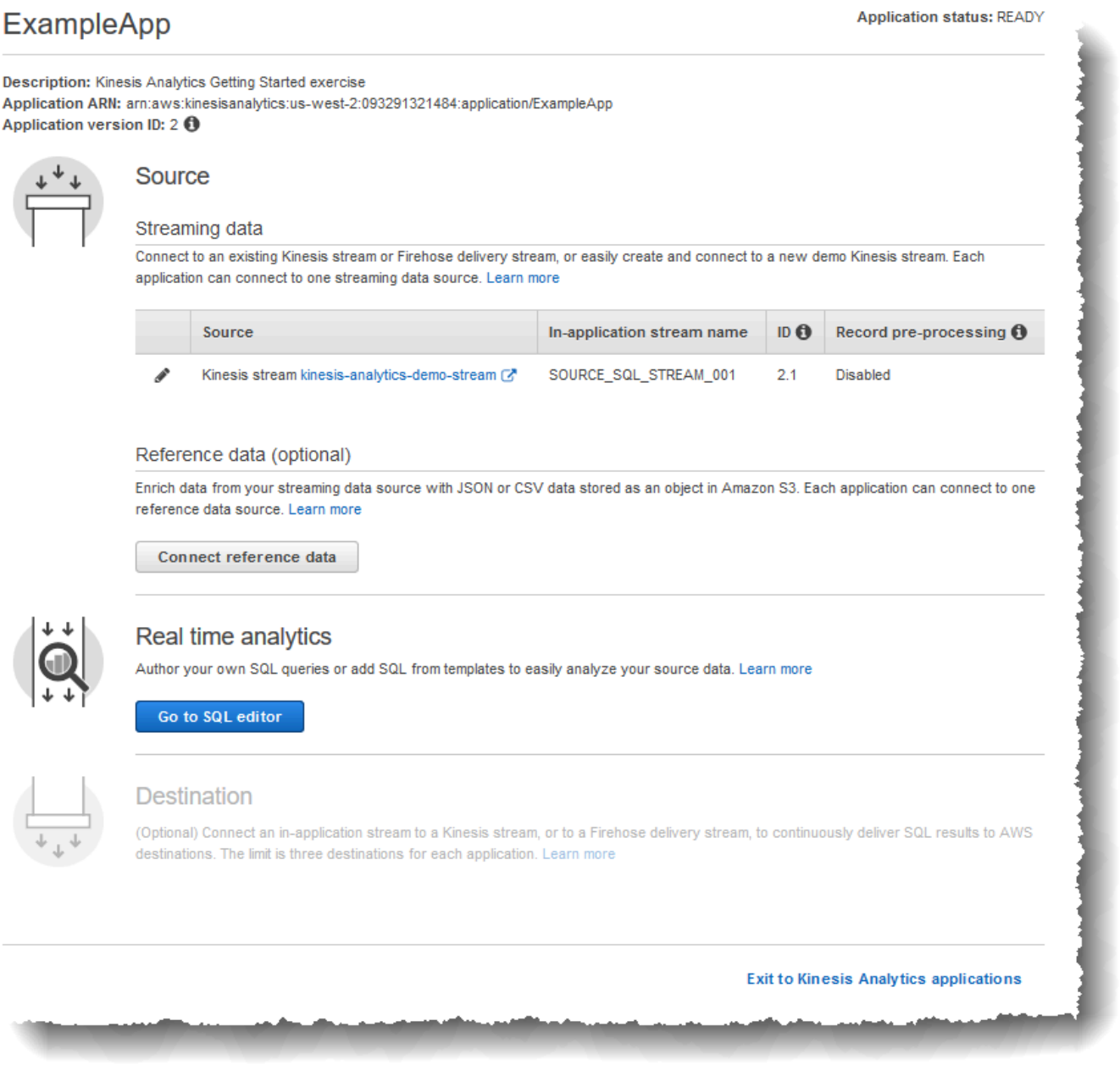
Langkah Selanjutnya

Langkah 3.3: Tambahkan Analitik Waktu Nyata (Tambahkan Kode Aplikasi)

Langkah 3.3: Tambahkan Analitik Waktu Nyata (Tambahkan Kode Aplikasi)

Anda dapat menulis kueri SQL Anda sendiri di aliran dalam aplikasi, tetapi untuk langkah berikut, Anda menggunakan salah satu templat yang menyediakan kode sampel.

1. Di halaman hub aplikasi, pilih Go to SQL editor (Buka editor SQL).



2. Dalam Apakah Anda ingin mulai menjalankan "ExampleApp“? kotak dialog, pilih Ya, mulai aplikasi.

   Konsol mengirimkan permintaan untuk memulai aplikasi (lihat StartApplication), lalu halaman editor SQL muncul.

3. Konsol membuka halaman editor SQL. Tinjau halaman, termasuk tombol (Add SQL from templates (Tambahkan SQL dari templat), Save and run SQL (Simpan dan jalankan SQL)) dan berbagai tab.

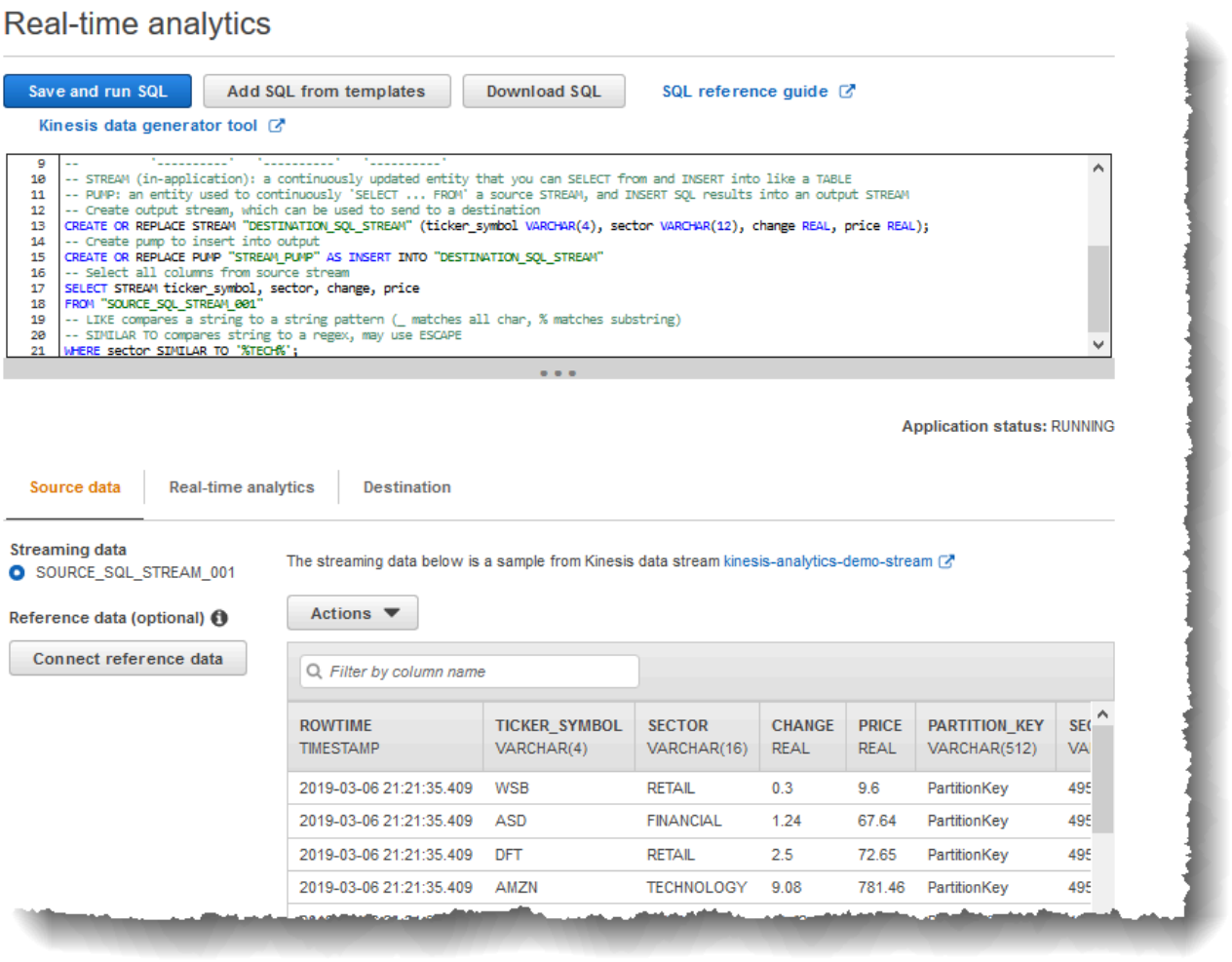
4. Di editor SQL, pilih Add SQL from templates (Tambahkan SQL dari templat).
5. Dari daftar templat yang tersedia, pilih Continuous filter (Filter berkelanjutan). Kode sampel membaca data dari satu aliran dalam aplikasi (klausa `WHERE` memfilter baris) dan memasukkannya di aliran dalam aplikasi lainnya sebagai berikut:

   - Ini membuat aliran dalam aplikasi `DESTINATION_SQL_STREAM`.
   - Ini membuat pompa `STREAM_PUMP`, dan menggunakannya untuk memilih baris dari `SOURCE_SQL_STREAM_001` dan memasukkannya dalam `DESTINATION_SQL_STREAM`.

6. Pilih Add this SQL to editor (Tambahkan SQL ini ke editor).
7. Uji kode aplikasi sebagai berikut:

   Ingat, Anda sudah memulai aplikasi (statusnya adalah RUNNING). Oleh karena itu, Amazon Kinesis Data Analytics terus membaca data dari sumber streaming dan menambahkan baris ke aliran dalam aplikasi `SOURCE_SQL_STREAM_001`.

   a. Di Editor SQL, pilih Save and run SQL (Simpan dan jalankan SQL). Konsol pertama-tama mengirimkan permintaan pembaruan untuk menyimpan kode aplikasi. Kemudian, kode terus berjalan.
   b. Anda dapat melihat hasilnya di tab Real-time analytics (Analitik waktu nyata).



Editor SQL memiliki tab berikut:

- Tab Source data (Data sumber) menampilkan aliran input dalam aplikasi yang dipetakan ke sumber streaming. Pilih aliran dalam aplikasi, dan Anda dapat melihat data yang masuk. Perhatikan kolom tambahan di aliran input dalam aplikasi yang tidak ditentukan dalam konfigurasi input. Ini termasuk kolom stempel waktu berikut:

  - ROWTIME – Setiap baris di aliran dalam aplikasi memiliki kolom khusus yang disebut `ROWTIME`. Kolom ini adalah stempel waktu ketika Amazon Kinesis Data Analytics memasukkan baris di aliran dalam aplikasi pertama (aliran input dalam aplikasi yang dipetakan ke sumber streaming).

- Approximate_Arrival_Time – Setiap catatan Kinesis Data Analytics menyertakan nilai yang disebut `Approximate_Arrival_Time`. Nilai ini adalah perkiraan kedatangan stempel waktu yang ditetapkan ketika sumber streaming berhasil menerima dan menyimpan catatan. Ketika Kinesis Data Analytics membaca catatan dari sumber streaming, kolom ini akan dimasukkan ke aliran input dalam aplikasi.

Nilai stempel waktu ini berguna dalam kueri jendela yang berbasis waktu. Untuk informasi selengkapnya, lihat Kueri Jendela.

- Tab Real-time analytics (Analitik waktu nyata) menunjukkan semua aliran dalam aplikasi lainnya yang dibuat dengan kode aplikasi Anda. Tab ini juga mencakup aliran kesalahan. Kinesis Data Analytics mengirimkan baris apa pun yang tidak dapat diproses ke aliran kesalahan. Untuk informasi selengkapnya, lihat Penanganan Kesalahan.

  Pilih `DESTINATION_SQL_STREAM` untuk melihat baris yang dimasukkan kode aplikasi Anda. Perhatikan kolom tambahan yang tidak dibuat oleh kode aplikasi Anda. Kolom ini termasuk kolom stempel waktu `ROWTIME`. Kinesis Data Analytics hanya menyalin nilai ini dari sumbernya (`SOURCE_SQL_STREAM_001`).

- Tab Destination (Tujuan) menunjukkan tujuan eksternal tempat Kinesis Data Analytics menulis hasil kueri. Anda belum mengonfigurasi tujuan eksternal apa pun untuk output aplikasi Anda.

Langkah Selanjutnya

Langkah 3.4: (Opsional) Perbarui Kode Aplikasi

Langkah 3.4: (Opsional) Perbarui Kode Aplikasi

Di langkah ini, Anda menjelajahi cara memperbarui kode aplikasi.

Untuk memperbarui kode aplikasi

1. Buat aliran dalam aplikasi lainnya seperti berikut:

- Buat aliran dalam aplikasi lainnya yang disebut DESTINATION_SQL_STREAM_2.
- Buat pompa, lalu gunakan untuk memasukkan baris dalam aliran yang baru dibuat dengan memilih baris dari DESTINATION_SQL_STREAM.

Di editor SQL, tambahkan kode berikut untuk kode aplikasi yang ada:

```
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";
```

Simpan dan jalankan kode. Aliran dalam aplikasi tambahan muncul di tab Real-time analytics (Analitik waktu nyata).

2. Buat dua aliran dalam aplikasi. Filter baris di SOURCE_SQL_STREAM_001 berdasarkan ticker saham, lalu masukkan ke dalam aliran terpisah ini.

   Tambahkan pernyataan SQL berikut ke kode aplikasi Anda:

```
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%';
```

Simpan dan jalankan kode. Perhatikan aliran dalam aplikasi tambahan di tab Real-time analytics (Analitik waktu nyata).

Anda sekarang memiliki aplikasi Amazon Kinesis Data Analytics pertama yang berfungsi. Dalam latihan ini, Anda melakukan hal berikut:

- Membuat aplikasi Kinesis Data Analytics pertama Anda.

- Mengonfigurasi input aplikasi yang mengidentifikasi aliran demo sebagai sumber streaming dan memetakannya ke aliran dalam aplikasi (`SOURCE_SQL_STREAM_001`) yang dibuat. Kinesis Data Analytics terus membaca aliran demo dan memasukkan catatan di aliran dalam aplikasi.

- Kode aplikasi Anda mengkueri `SOURCE_SQL_STREAM_001` dan menulis output ke aliran dalam aplikasi lainnya yang disebut `DESTINATION_SQL_STREAM`.

Sekarang Anda dapat secara opsional mengonfigurasi output aplikasi untuk menulis output aplikasi ke tujuan eksternal. Artinya, Anda dapat mengonfigurasi output aplikasi untuk menulis catatan di `DESTINATION_SQL_STREAM` ke tujuan eksternal. Untuk latihan ini, ini adalah langkah opsional. Untuk mempelajari cara mengonfigurasi tujuan, lanjutkan ke langkah berikutnya.

Langkah Selanjutnya

Langkah 4 (Opsional) Edit Skema dan Kode SQL Menggunakan Konsol.

Langkah 4 (Opsional) Edit Skema dan Kode SQL Menggunakan Konsol

Setelah itu, Anda dapat menemukan informasi tentang cara mengedit skema yang disimpulkan dan cara mengedit kode SQL untuk Amazon Kinesis Data Analytics. Anda melakukannya dengan

menggunakan editor skema dan editor SQL yang merupakan bagian dari konsol Kinesis Data Analytics.

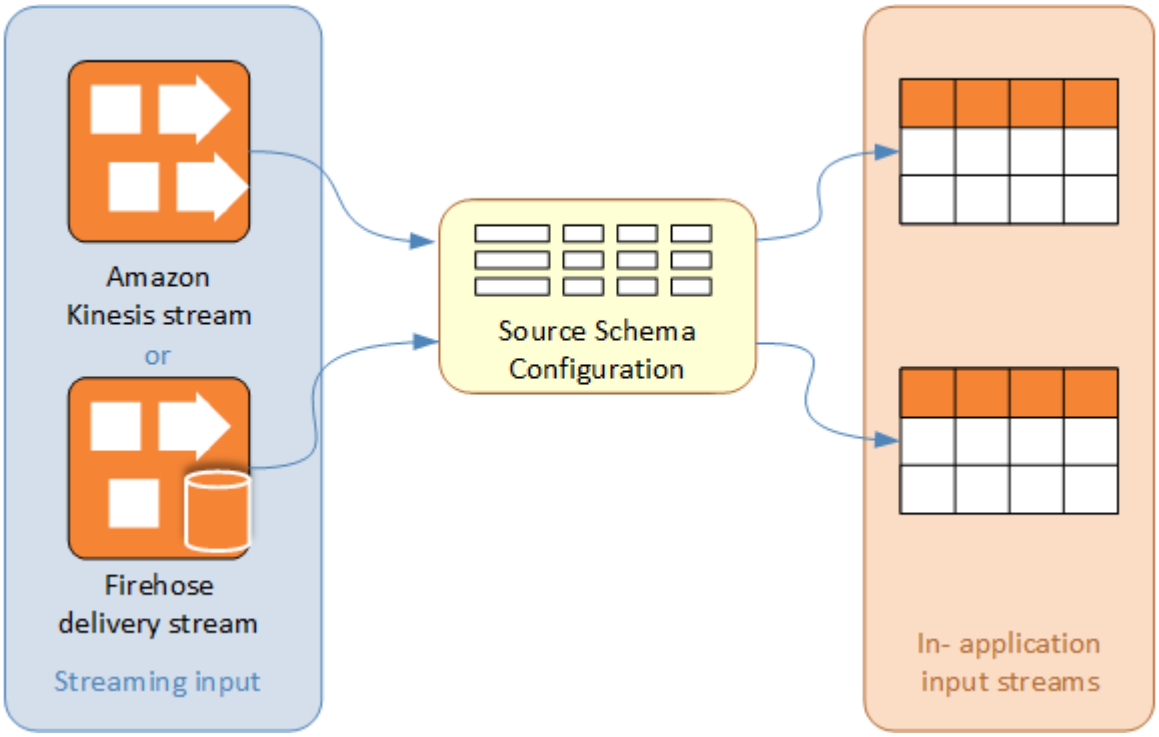
> **Note**
>
> Untuk mengakses atau mengambil sampe data di konsol, peran pengguna login Anda harus memiliki izin `kinesisanalytics:GetApplicationState`. Untuk informasi selengkapnya tentang izin aplikasi Kinesis Data Analytics, lihat Gambaran Umum Pengelolaan Akses.

Topik

- Menggunakan Editor Skema
- Menggunakan Editor SQL

Menggunakan Editor Skema

Skema untuk aliran input aplikasi Amazon Kinesis Data Analytics mendefinisikan bagaimana data dari aliran disediakan untuk kueri SQL dalam aplikasi.



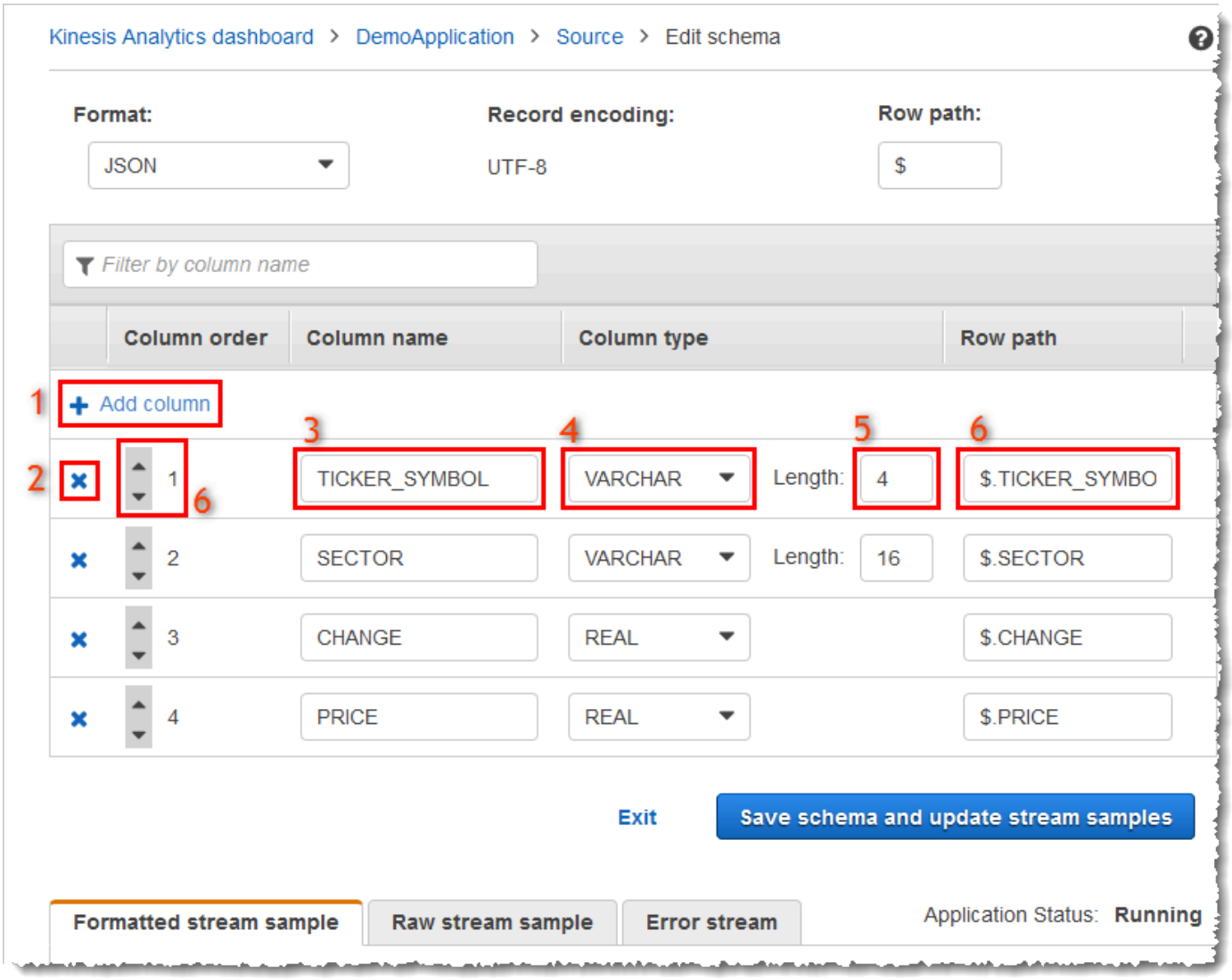
Skema berisi kriteria pilihan untuk menentukan bagian dari input streaming yang diubah menjadi kolom data di aliran input dalam aplikasi. Ini dapat menjadi salah satu dari yang berikut:

- Ekspresi JSONPath untuk aliran input JSON. JSONPath adalah alat untuk mengkueri data JSON.
- Jumlah kolom untuk aliran input dalam format nilai yang dipisahkan koma (CSV).
- Nama kolom dan tipe data SQL untuk menyajikan data di aliran data dalam aplikasi. Tipe data juga berisi panjang untuk karakter atau data biner.

Konsol mencoba membuat skema menggunakan DiscoverInputSchema. Jika penemuan skema gagal atau mengembalikan skema yang salah atau tidak lengkap, Anda harus mengedit skema secara manual menggunakan editor skema.

Layar Utama Editor Skema

Tangkapan layar berikut menunjukkan layar utama untuk Editor Skema.



Anda dapat menerapkan pengeditan berikut ke skema:

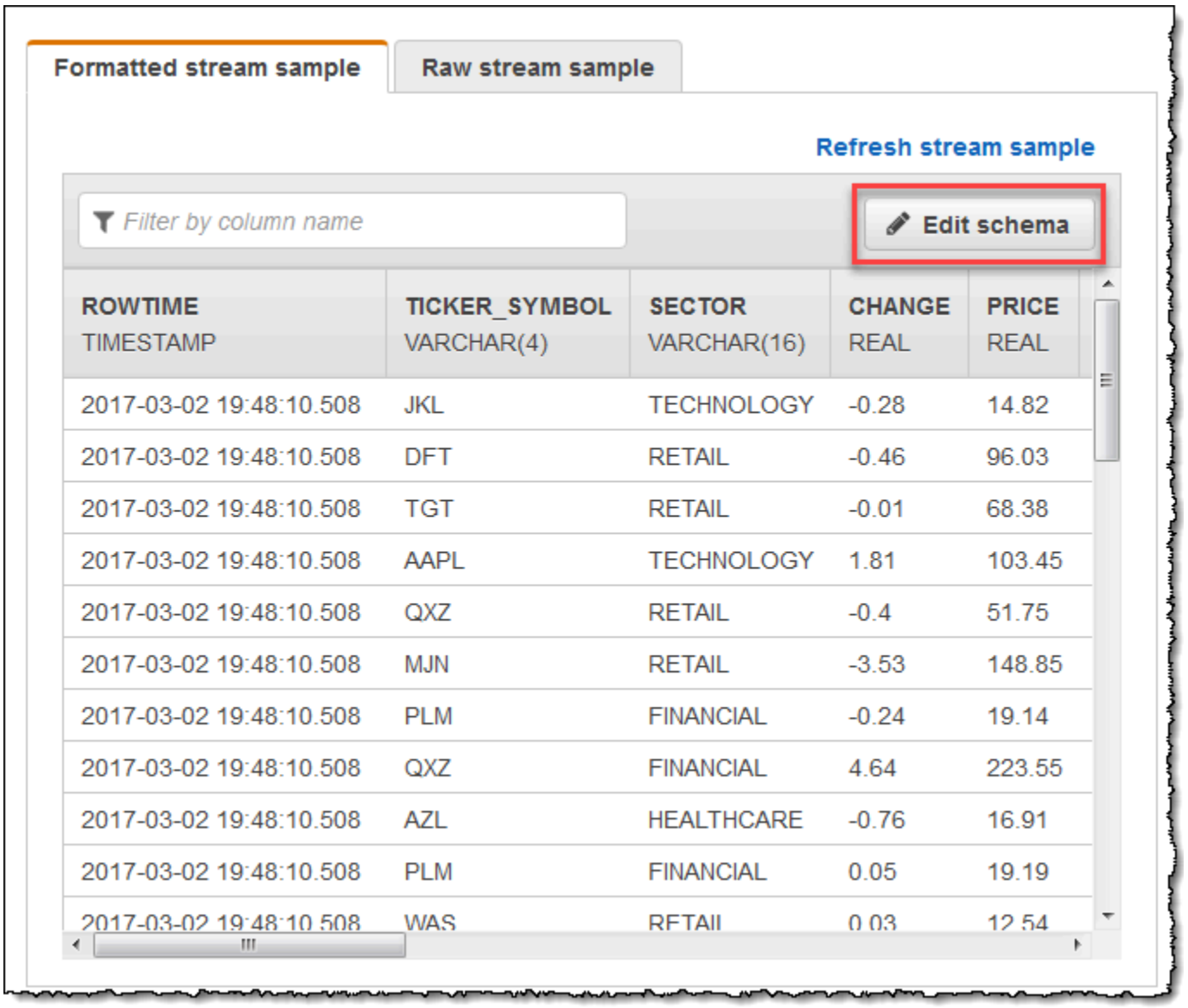
- Tambahkan kolom (1): Anda mungkin perlu menambahkan kolom data jika item data tidak terdeteksi secara otomatis.
- Hapus kolom (2): Anda dapat mengecualikan data dari aliran sumber jika aplikasi Anda tidak memerlukannya. Pengecualian ini tidak memengaruhi data dalam aliran sumber. Jika data dikecualikan, data tersebut tidak disediakan untuk aplikasi.
- Ubah nama kolom (3). Nama kolom tidak boleh kosong, harus lebih panjang dari satu karakter, dan tidak boleh berisi kata kunci SQL tersimpan. Nama juga harus memenuhi kriteria penamaan untuk pengidentifikasi biasa SQL: Nama harus dimulai dengan huruf dan hanya berisi huruf, karakter garis bawah, dan angka.
- Ubah tipe data (4) atau panjang (5) kolom: Anda dapat menentukan tipe data yang kompatibel untuk kolom. Jika Anda menentukan tipe data yang tidak kompatibel, kolom diisi dengan NULL atau aliran dalam aplikasi tidak diisi sama sekali. Dalam kasus terakhir, kesalahan ditulis ke aliran kesalahan. Jika Anda menentukan panjang untuk kolom yang terlalu kecil, data yang masuk akan dipotong.
- Ubah kriteria pemilihan kolom (6): Anda dapat mengedit ekspresi JSONPath atau urutan kolom CSV yang digunakan untuk menentukan sumber data dalam kolom. Untuk mengubah kriteria pemilihan untuk skema JSON, masukkan nilai baru untuk ekspresi jalur baris. Skema CSV menggunakan urutan kolom sebagai kriteria pilihan. Untuk mengubah kriteria pemilihan skema CSV, ubah urutan kolom.

Mengedit Skema untuk Sumber Streaming

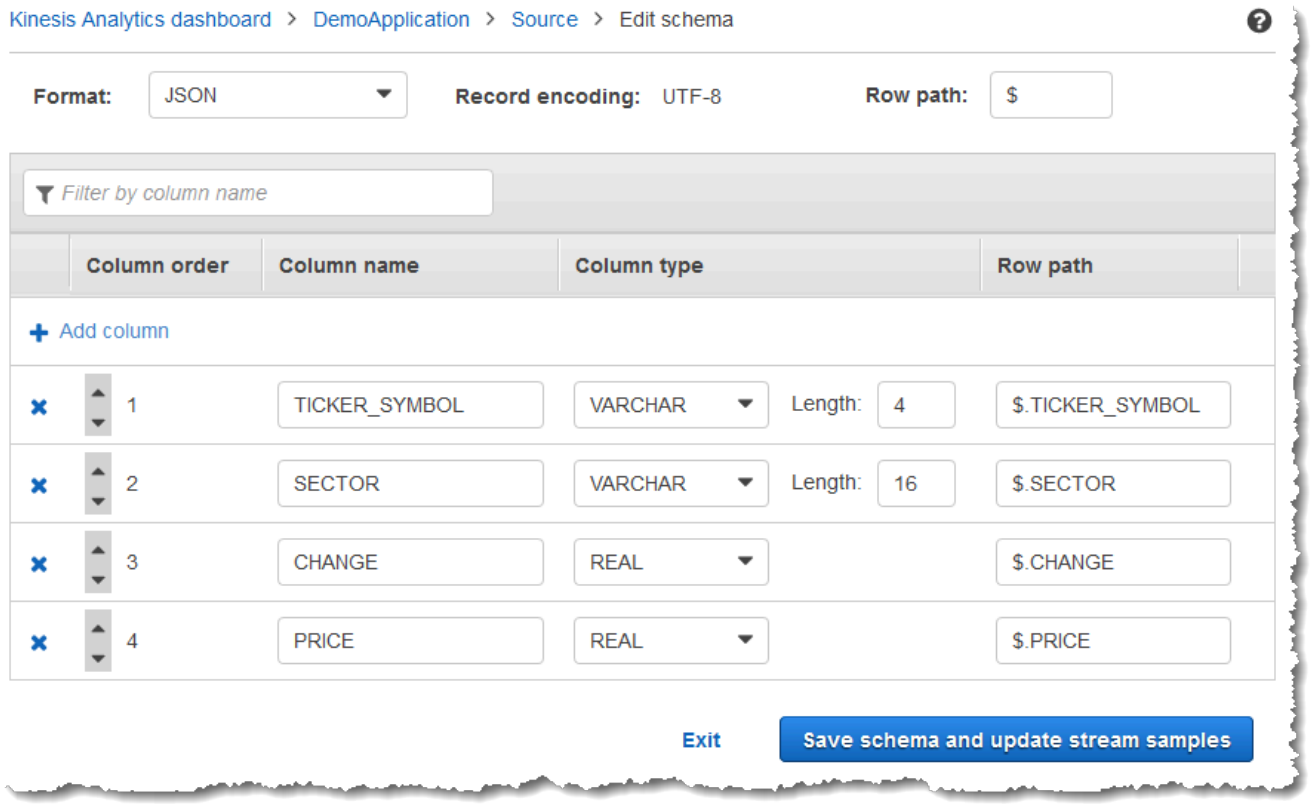
Jika Anda perlu mengedit skema untuk sumber streaming, ikuti langkah-langkah berikut.

Untuk mengedit skema sumber streaming

1. Di halaman Sumber, pilih Edit schema (Edit skema).



2. Di halaman Edit skema, edit skema sumber.



3. Untuk Format, pilih JSON atau CSV. Untuk format JSON atau CSV, pengkodean yang didukung adalah ISO 8859-1.

Untuk informasi selengkapnya tentang mengedit skema untuk format JSON atau CSV, lihat prosedur di bagian berikutnya.

Mengedit Skema JSON

Anda dapat mengedit skema JSON menggunakan langkah-langkah berikut.

Untuk mengedit skema JSON

1. Di editor skema, pilih Add column (Tambahkan kolom) untuk menambahkan kolom.

   Kolom baru muncul di posisi kolom pertama. Untuk mengubah urutan kolom, pilih panah atas dan bawah di samping nama kolom.

   Untuk kolom baru, masukkan informasi berikut:

   - Untuk Column name (Nama kolom), masukkan nama.

Nama kolom tidak boleh kosong, harus lebih panjang dari satu karakter, dan tidak boleh berisi kata kunci SQL tersimpan. Nama juga harus memenuhi kriteria penamaan untuk pengidentifikasi biasa SQL: Nama harus dimulai dengan huruf dan hanya berisi huruf, karakter garis bawah, serta angka.

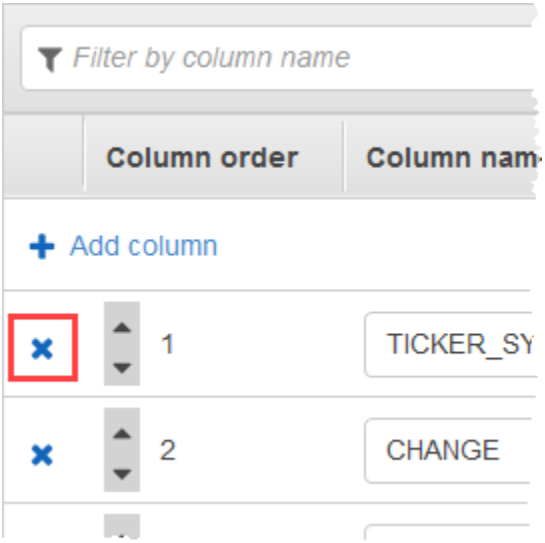
- Untuk Column type (Tipe kolom), masukkan tipe data SQL.

  Tipe kolom dapat berupa tipe data SQL yang didukung. Jika tipe data baru adalah CHAR, VARBINARY, atau VARCHAR, tentukan panjang data untuk Length (Panjang). Untuk informasi selengkapnya, lihat Tipe Data.

- Untuk Row path (Jalur baris), masukkan jalur baris. Jalur baris adalah ekspresi JSONPath valid yang memetakan ke elemen JSON.

> **Note**
>
> Nilai Row path (Jalur baris) dasar adalah jalur ke induk tingkat atas yang berisi data yang akan diimpor. Nilainya adalah $ secara default. Untuk informasi selengkapnya, lihat `RecordRowPath` di `JSONMappingParameters`.

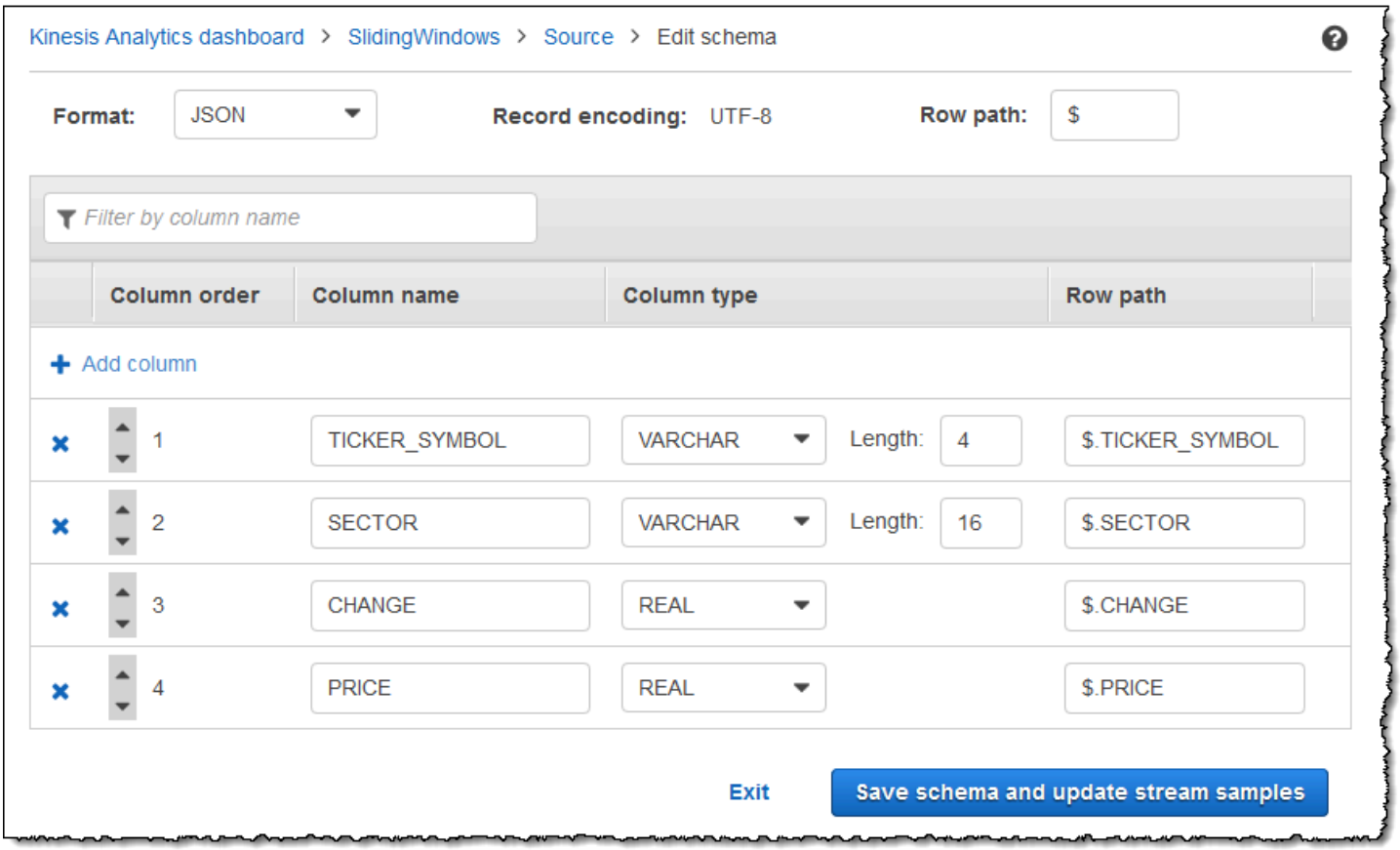
2. Untuk menghapus kolom, pilih ikon x di sebelah nomor kolom.



3. Untuk mengubah nama kolom, masukkan nama baru untuk Column name (Nama kolom). Nama kolom baru tidak boleh kosong, harus lebih panjang dari satu karakter, dan tidak boleh berisi kata kunci SQL tersimpan. Nama juga harus memenuhi kriteria penamaan untuk pengidentifikasi biasa SQL: Nama harus dimulai dengan huruf dan hanya berisi huruf, karakter garis bawah, serta angka.

4. Untuk mengubah tipe data kolom, pilih tipe data baru untuk Column type (Tipe kolom). Jika tipe data baru adalah `CHAR`, `VARBINARY`, atau `VARCHAR`, tentukan panjang data untuk Length (Panjang). Untuk informasi selengkapnya, lihat Tipe Data.
5. Pilih Save schema and update stream (Simpan skema dan perbarui aliran) untuk menyimpan perubahan Anda.

Skema yang dimodifikasi muncul di editor dan terlihat seperti berikut ini.



Jika skema Anda memiliki banyak baris, Anda dapat memfilter baris menggunakan Filter by column name (Filter berdasarkan nama kolom). Misalnya, untuk mengedit nama kolom yang dimulai dengan `P`, seperti kolom `Price`, masukkan `P` di kotak Filter by column name (Filter berdasarkan nama kolom).

Mengedit Skema CSV

Anda dapat mengedit skema CSV menggunakan langkah-langkah berikut.

Untuk mengedit skema CSV

1. Di editor skema, untuk Row delimiter (Pembatas baris), pilih pembatas yang digunakan oleh aliran data masuk Anda. Ini adalah pembatas antara catatan data dalam aliran Anda, seperti karakter baris baru.
2. Untuk Column delimiter (Pembatas kolom), pilih pembatas yang digunakan oleh aliran data masuk Anda. Ini adalah pembatas di antara bidang data dalam aliran Anda, seperti koma.
3. Untuk menambahkan kolom, pilih Add column (Tambahkan kolom).

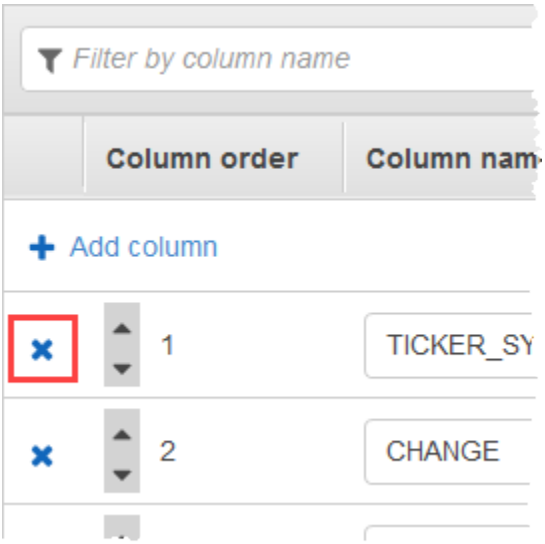
   Kolom baru muncul di posisi kolom pertama. Untuk mengubah urutan kolom, pilih panah atas dan bawah di samping nama kolom.

   Untuk kolom baru, masukkan informasi berikut:

   - Untuk Column name (Nama kolom), masukkan nama.

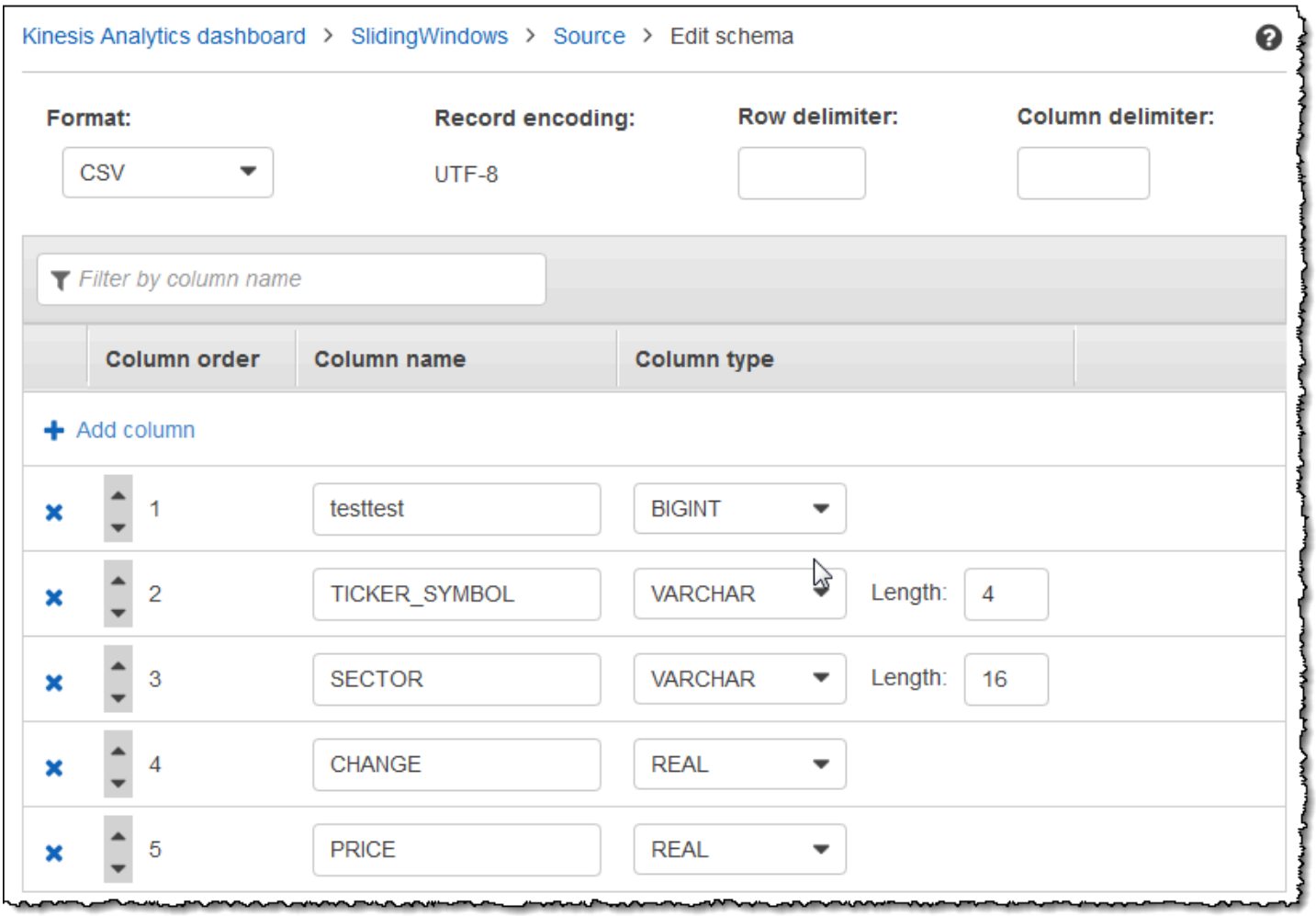
     Nama kolom tidak boleh kosong, harus lebih panjang dari satu karakter, dan tidak boleh berisi kata kunci SQL tersimpan. Nama juga harus memenuhi kriteria penamaan untuk pengidentifikasi biasa SQL: Nama harus dimulai dengan huruf dan hanya berisi huruf, karakter garis bawah, serta angka.
   - Untuk Column type (Tipe kolom), masukkan tipe data SQL.

     Tipe kolom dapat berupa tipe data SQL yang didukung. Jika tipe data baru adalah CHAR, VARBINARY, atau VARCHAR, tentukan panjang data untuk Length (Panjang). Untuk informasi selengkapnya, lihat Tipe Data.
4. Untuk menghapus kolom, pilih ikon x di sebelah nomor kolom.



5. Untuk mengubah nama kolom, masukkan nama baru di Column name (Nama kolom). Nama kolom baru tidak boleh kosong, harus lebih panjang dari satu karakter, dan tidak boleh berisi kata kunci SQL tersimpan. Nama juga harus memenuhi kriteria penamaan untuk pengidentifikasi biasa SQL: Nama harus dimulai dengan huruf dan hanya berisi huruf, karakter garis bawah, serta angka.
6. Untuk mengubah tipe data kolom, pilih tipe data baru untuk Column type (Tipe kolom). Jika tipe data baru adalah CHAR, VARBINARY, atau VARCHAR, tentukan panjang data untuk Length (Panjang). Untuk informasi selengkapnya, lihat Tipe Data.
7. Pilih Save schema and update stream (Simpan skema dan perbarui aliran) untuk menyimpan perubahan Anda.

Skema yang dimodifikasi muncul di editor dan terlihat seperti berikut ini.

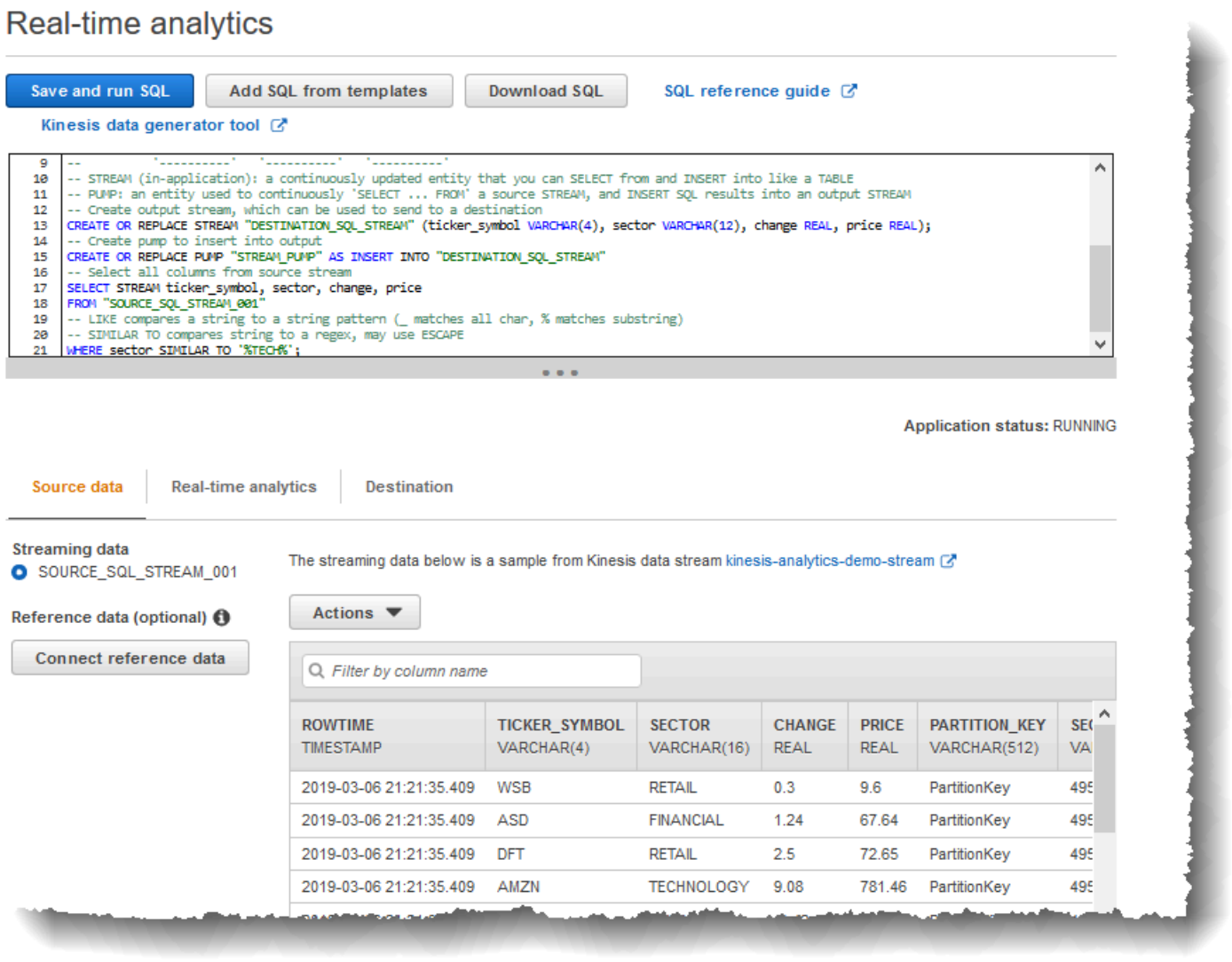


Jika skema Anda memiliki banyak baris, Anda dapat memfilter baris menggunakan Filter by column name (Filter berdasarkan nama kolom). Misalnya, untuk mengedit nama kolom yang dimulai dengan

P, seperti kolom `Price`, masukkan P di kotak Filter by column name (Filter berdasarkan nama kolom).

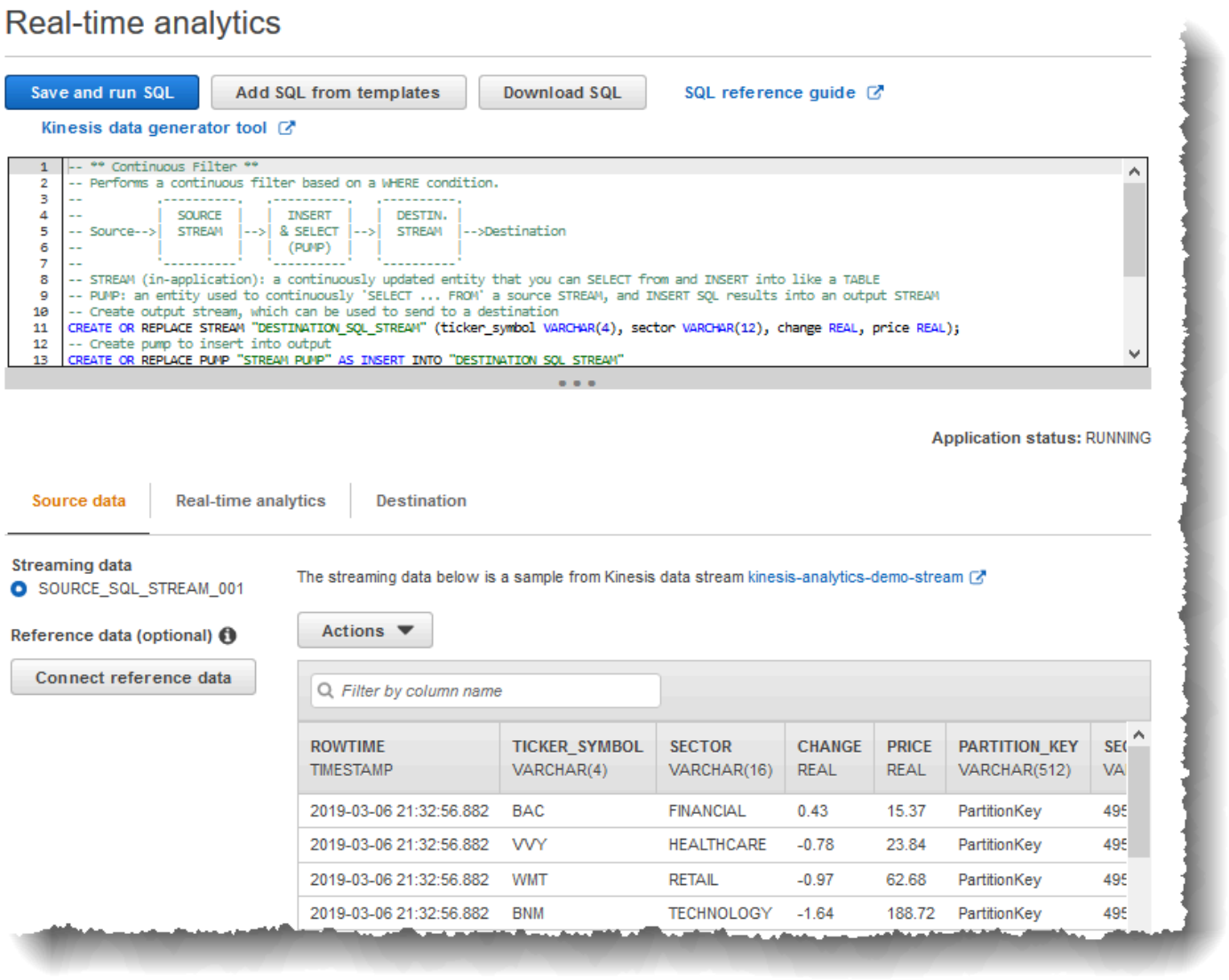
Menggunakan Editor SQL

Selanjutnya, Anda dapat menemukan informasi tentang bagian editor SQL dan cara kerja masing-masing. Dalam SQL editor, Anda dapat menulis kode Anda sendiri atau memilih Add SQL from templates (Tambahkan SQL dari templat). Templat SQL memberikan contoh kode SQL yang dapat membantu Anda menulis aplikasi Amazon Kinesis Data Analytics umum. Contoh aplikasi dalam panduan ini menggunakan beberapa templat ini. Untuk informasi selengkapnya, lihat Kinesis Data Analytics untuk contoh SQL.



Tab Data Sumber

Tab Source data (Data sumber) mengidentifikasi sumber streaming. Tab ini juga mengidentifikasi aliran input dalam aplikasi yang dipetakan sumber ini dan yang menyediakan konfigurasi input aplikasi.



Amazon Kinesis Data Analytics menyediakan kolom stempel waktu berikut, agar Anda tidak perlu menyediakan pemetaan eksplisit dalam konfigurasi input Anda:

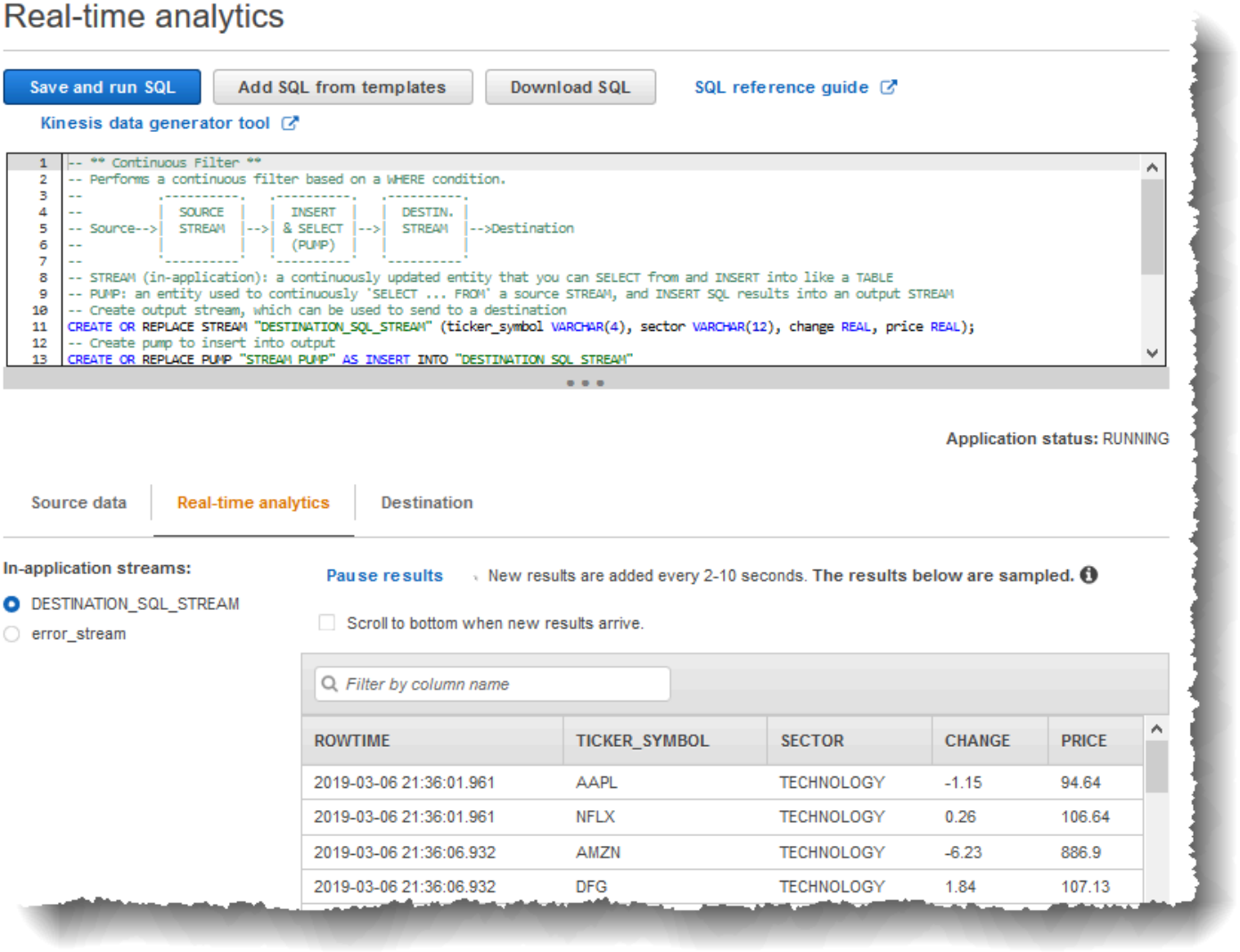
- ROWTIME – Setiap baris di aliran dalam aplikasi memiliki kolom khusus yang disebut `ROWTIME`. Kolom ini adalah stempel waktu untuk titik ketika Kinesis Data Analytics memasukkan baris di aliran dalam aplikasi pertama.

- Approximate_Arrival_Time – Catatan di sumber streaming menyertakan kolom `Approximate_Arrival_Timestamp`. Ini adalah perkiraan waktu stempel kedatangan yang ditetapkan ketika sumber streaming berhasil menerima dan menyimpan catatan terkait. Kinesis Data Analytics mengambil kolom ini ke aliran input dalam aplikasi sebagai`Approximate_Arrival_Time`. Amazon Kinesis Data Analytics hanya menyediakan kolom ini di aliran input dalam aplikasi yang dipetakan ke sumber streaming.

Nilai stempel waktu ini berguna dalam kueri jendela yang berbasis waktu. Untuk informasi selengkapnya, lihat Kueri Jendela.

Tab Analitik Waktu Nyata

Tab Real-time analytics (Analitik waktu nyata) menunjukkan semua aliran dalam aplikasi yang dibuat kode aplikasi Anda. Grup aliran ini mencakup aliran kesalahan (`error_stream`) yang disediakan Amazon Kinesis Data Analytics untuk semua aplikasi.



Tab Tujuan

Tab Destination (Tujuan) memungkinkan Anda mengonfigurasi output aplikasi untuk menyimpan aliran dalam aplikasi ke tujuan eksternal. Anda dapat mengonfigurasi output untuk menyimpan data di salah satu aliran dalam aplikasi ke tujuan eksternal. Untuk informasi selengkapnya, lihat Mengonfigurasi Output Aplikasi.

Konsep SQL Streaming

Amazon Kinesis Data Analytics menerapkan standar SQL ANSI 2008 dengan ekstensi. Ekstensi ini memungkinkan Anda memproses data streaming. Topik berikut mencakup konsep-konsep SQL streaming kunci.

Topik

- Aliran dan Pompa dalam Aplikasi
- Stempel waktu dan Kolom ROWTIME
- Kueri Berkelanjutan
- Kueri Jendela
- Operasi Data Streaming: Gabungan Streaming

Aliran dan Pompa dalam Aplikasi

Saat Anda mengonfigurasi input aplikasi, Anda memetakan sumber streaming ke aliran dalam aplikasi yang dibuat. Data terus mengalir dari sumber streaming ke aliran dalam aplikasi. Aliran dalam aplikasi bekerja seperti tabel yang dapat Anda kueri menggunakan pernyataan SQL, tetapi ini disebut aliran karena merupakan aliran data berkelanjutan.

> **Note**
>
> Jangan bingung antara aliran dalam aplikasi dengan aliran data Amazon Kinesis dan aliran pengiriman Firehose. Aliran dalam aplikasi hanya ada dalam konteks aplikasi Amazon Kinesis Data Analytics. Aliran data Kinesis dan aliran pengiriman Firehose ada secara independen dari aplikasi Anda. Anda dapat mengonfigurasi keduanya sebagai sumber streaming dalam konfigurasi input aplikasi Anda atau sebagai tujuan dalam konfigurasi output.

Anda juga dapat membuat lebih banyak aliran dalam aplikasi jika perlu untuk menyimpan hasil kueri menengah. Membuat aliran dalam aplikasi adalah proses dua langkah. Pertama, Anda membuat aliran dalam aplikasi, lalu Anda memompa data ke dalamnya. Misalnya, konfigurasi input aplikasi

Anda membuat aliran dalam aplikasi Anda bernama INPUTSTREAM. Dalam contoh berikut, Anda membuat aliran lain (TEMPSTREAM), lalu Anda memompa data dari INPUTSTREAM ke dalamnya.

1. Buat aliran dalam aplikasi (TEMPSTREAM) dengan tiga kolom, seperti yang ditunjukkan berikut:

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

   Nama-nama kolom ditentukan dalam tanda kutip, menjadikannya peka huruf besar/kecil. Untuk informasi selengkapnya, lihat Pengidentifikasi di Referensi SQL Amazon Kinesis Data Analytics.

2. Masukkan data ke dalam aliran menggunakan pompa. Pompa adalah kueri masuk berkelanjutan yang berjalan yang memasukkan data dari satu aliran dalam aplikasi ke aliran dalam aplikasi lainnya. Pernyataan berikut membuat pompa (SAMPLEPUMP) dan memasukkan data ke dalam TEMPSTREAM dengan memilih catatan dari aliran lainnya (INPUTSTREAM).

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

Anda dapat memasukkan beberapa penulis ke aliran dalam aplikasi, dan mungkin ada beberapa pembaca yang dipilih dari aliran. Pikirkan aliran dalam aplikasi sebagai penerapan paradigma pesan terbitkan/berlangganan. Dalam paradigma ini, baris data, termasuk waktu pembuatan dan waktu penerimaan, dapat diproses, diinterpretasikan, dan diteruskan oleh serangkaian pernyataan SQL streaming, tanpa harus disimpan dalam RDBMS tradisional.

Setelah aliran dalam aplikasi dibuat, Anda dapat melakukan kueri SQL normal.

> **Note**
>
> Ketika Anda mengkueri aliran, sebagian besar pernyataan SQL diikat menggunakan jendela berbasis baris atau berbasis waktu. Untuk informasi selengkapnya, lihat Kueri Jendela.

Anda juga dapat bergabung dengan aliran. Untuk contoh bergabung dengan aliran, lihat Operasi Data Streaming: Gabungan Streaming.

Stempel waktu dan Kolom ROWTIME

Aliran dalam aplikasi mencakup kolom khusus yang disebut `ROWTIME`. Kolom ini menyimpan stempel waktu ketika Amazon Kinesis Data Analytics memasukkan baris di aliran dalam aplikasi pertama. `ROWTIME` mencerminkan stempel waktu tempat Amazon Kinesis Data Analytics memasukkan catatan ke aliran dalam aplikasi pertama setelah membaca dari sumber streaming. Nilai `ROWTIME` ini selanjutnya dipertahankan di seluruh aplikasi Anda.

> **Note**
>
> Ketika Anda memompa catatan dari satu aliran dalam aplikasi ke aliran lainnya, Anda tidak perlu secara eksplisit menyalin kolom `ROWTIME`, Amazon Kinesis Data Analytics akan menyalin kolom ini untuk Anda.

Amazon Kinesis Data Analytics menjamin nilai-nilai `ROWTIME` ditingkatkan secara monoton. Anda menggunakan stempel waktu ini di kueri jendela berbasis waktu. Untuk informasi selengkapnya, lihat Kueri Jendela.

Anda dapat mengakses kolom ROWTIME di pernyataan `SELECT` seperti kolom lain di aliran dalam aplikasi Anda. Sebagai contoh:

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

Memahami Berbagai Waktu dalam Analitik Streaming

Selain `ROWTIME`, ada tipe waktu lain dalam aplikasi streaming waktu nyata. Ini adalah:

- Event time (Waktu peristiwa) – Stempel waktu ketika peristiwa terjadi. Ini kadang-kadang juga disebut waktu sisi klien. Waktu ini sering kali berguna ketika digunakan dalam analitik karena merupakan waktu ketika peristiwa terjadi. Namun, banyak sumber peristiwa, seperti ponsel dan klien web, tidak memiliki jam yang dapat diandalkan, yang dapat menyebabkan waktu yang tidak

akurat. Selain itu, masalah konektivitas dapat menyebabkan catatan yang muncul di aliran tidak dalam urutan yang sama dengan peristiwa yang terjadi.

- Ingest time (Waktu penyerapan) – Stempel waktu ketika catatan ditambahkan ke sumber streaming. Amazon Kinesis Data Streams mencakup bidang yang disebut `APPROXIMATE_ARRIVAL_TIME` di setiap catatan yang menyediakan stempel waktu ini. Ini kadang-kadang juga disebut sebagai waktu sisi server. Waktu penyerapan ini sering kali mendekati waktu kejadian. Jika ada jenis keterlambatan dalam penyerapan catatan ke aliran, ini dapat menyebabkan ketidakakuratan, yang biasanya jarang terjadi. Selain itu, waktu penyerapan jarang salah, tetapi bisa terjadi karena sifat data streaming yang terdistribusi. Oleh karena itu, waktu penyerapan sebagian besar merupakan cerminan waktu peristiwa yang akurat dan berurutan.

- Processing time (Waktu pemrosesan) – Stempel waktu ketika Amazon Kinesis Data Analytics memasukkan baris di aliran dalam aplikasi pertama. Amazon Kinesis Data Analytics menyediakan stempel waktu ini di kolom `ROWTIME` yang ada di setiap aliran dalam aplikasi. Waktu pemrosesan selalu meningkat secara monoton. Namun ini tidak akan akurat jika aplikasi Anda tertinggal. (Jika aplikasi tertinggal, waktu pemrosesan tidak mencerminkan waktu peristiwa secara akurat.) `ROWTIME` ini akurat dalam kaitannya dengan jam dinding, tetapi mungkin bukan waktu ketika peristiwa benar-benar terjadi.

Menggunakan setiap waktu ini dalam kueri jendela yang berbasis waktu memiliki kelebihan dan kekurangan. Sebaiknya pilih satu atau beberapa waktu ini, dan strategi untuk menangani kerugian yang relevan berdasarkan skenario kasus penggunaan Anda.

> **Note**
>
> Jika Anda menggunakan jendela berbasis baris, waktu bukan merupakan masalah dan Anda dapat mengabaikan bagian ini.

Kami merekomendasikan strategi dua jendela yang menggunakan dua berbasis waktu, `ROWTIME` dan salah satu waktu lainnya (waktu penyerapan atau peristiwa).

- Gunakan ROWTIME sebagai jendela pertama, yang mengontrol seberapa sering kueri memancarkan hasil, seperti yang ditunjukkan dalam contoh berikut. Ini tidak digunakan sebagai waktu logis.
- Gunakan salah satu waktu lain yang merupakan waktu logis yang ingin Anda kaitkan dengan analitik Anda. Waktu ini mewakili kapan peristiwa terjadi. Pada contoh berikut, tujuan analitik adalah mengelompokkan catatan dan dan kembali menghitung dengan ticker

Keuntungan strategi ini adalah ini dapat menggunakan waktu yang mewakili kapan peristiwa terjadi. Ini dapat menangani dengan mudah ketika aplikasi Anda tertinggal atau ketika peristiwa tiba tidak sesuai urutan. Jika aplikasi tertinggal ketika membawa catatan ke aliran dalam aplikasi, catatan masih dikelompokkan berdasarkan waktu logis di jendela kedua. Kueri menggunakan ROWTIME untuk menjamin urutan pemrosesan. Catatan yang terlambat (stempel waktu penyerapan menunjukkan nilai sebelumnya dibandingkan dengan nilai ROWTIME) juga berhasil diproses.

Pertimbangkan kueri berikut terhadap aliran demo yang digunakan dalam Latihan Memulai. Kueri menggunakan klausul GROUP BY dan memancarkan hitungan ticker dalam jendela tumbling satu menit.

```
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);
```

Di GROUP BY, Anda pertama-tama mengelompokkan catatan berdasarkan ROWTIME di jendela satu menit, lalu dengan APPROXIMATE_ARRIVAL_TIME.

Nilai stempel waktu dalam hasil dibulatkan ke interval 60 detik terdekat. Hasil grup pertama yang dipancarkan oleh kueri menunjukkan catatan di menit pertama. Grup kedua hasil yang dipancarkan menunjukkan catatan di menit berikutnya berdasarkan `ROWTIME`. Catatan terakhir menunjukkan aplikasi terlambat dalam membawa catatan di aliran dalam aplikasi (ini menunjukkan nilai `ROWTIME` yang terlambat dibandingkan dengan stempel waktu penyerapan).

```
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.
```

Anda dapat menggabungkan hasil untuk hitungan akurat akhir per menit dengan mendorong hasil ke basis data hilir. Misalnya, Anda dapat mengonfigurasi output aplikasi untuk mempertahankan hasil ke aliran pengiriman Firehose yang dapat menulis ke tabel Amazon Redshift. Setelah hasil berada di tabel Amazon Redshift, Anda dapat mengkueri tabel untuk mengomputasi total jumlah grup dengan `Ticker_Symbol`. Dalam hal `XYZ`, totalnya akurat (6+1) meskipun catatan tiba terlambat.

Kueri Berkelanjutan

Kueri melalui aliran terus mengeksekusi data streaming. Eksekusi berkelanjutan ini memungkinkan skenario, seperti kemampuan bagi aplikasi untuk terus mengkueri aliran dan menghasilkan pemberitahuan.

Dalam latihan Memulai, Anda memiliki aliran dalam aplikasi bernama `SOURCE_SQL_STREAM_001`. Aliran ini terus menerima harga saham dari aliran demo (Kinesis data stream). Skemanya adalah sebagai berikut:

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

```
  PRICE REAL)
```

Misalkan Anda tertarik dengan perubahan harga saham yang lebih besar dari 15 persen. Anda dapat menggunakan kueri berikut dalam kode aplikasi Anda. Kueri ini terus berjalan dan memancarkan catatan ketika perubahan harga saham yang lebih besar dari 15 persen terdeteksi.

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

Gunakan prosedur berikut untuk menyiapkan aplikasi Amazon Kinesis Data Analytics dan menguji kueri ini.

Untuk menguji kueri

1. Buat aplikasi mengikuti Latihan Memulai.
2. Ubah pernyataan SELECT dalam kode aplikasi dengan kueri SELECT sebelumnya. Kode aplikasi yang dihasilkan ditampilkan sebagai berikut:

   ```
   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;
   ```

Kueri Jendela

Kueri SQL dalam kode aplikasi Anda dijalankan secara terus-menerus melalui aliran dalam aplikasi. Aliran dalam aplikasi mewakili data tidak terbatas yang mengalir terus-menerus melalui aplikasi Anda. Oleh karena itu, untuk mendapatkan serangkaian hasil dari input yang terus diperbarui ini, Anda sering kali mengikat kueri menggunakan jendela yang didefinisikan dalam hal waktu atau baris. Ini juga disebut SQL jendela.

Untuk kueri jendela berbasis waktu, Anda menentukan ukuran jendela dalam hal waktu (misalnya, jendela satu menit). Ini memerlukan kolom stempel waktu di aliran dalam aplikasi Anda yang

meningkat secara monoton. (Stempel waktu untuk baris baru lebih besar dari atau sama dengan baris sebelumnya.) Amazon Kinesis Data Analytics menyediakan kolom stempel waktu yang disebut `ROWTIME` untuk setiap aliran aplikasi. Anda dapat menggunakan kolom ini saat menentukan kueri berbasis waktu. Untuk aplikasi Anda, Anda dapat memilih beberapa opsi stempel waktu lainnya. Untuk informasi selengkapnya, lihat Stempel waktu dan Kolom ROWTIME.

Untuk kueri jendela berbasis baris, Anda menentukan ukuran jendela dalam hal jumlah baris.

Anda dapat menentukan kueri untuk memproses catatan di jendela tumbling, jendela geser, atau jendela stagger, bergantung pada kebutuhan aplikasi Anda. Kinesis Data Analytics mendukung tipe jendela berikut:

- Jendela Stagger: Kueri yang mengumpulkan data menggunakan jendela berbasis waktu kunci yang terbuka saat data tiba. Kunci memungkinkan beberapa jendela yang tumpang tindih. Ini adalah cara yang disarankan untuk mengumpulkan data menggunakan jendela berbasis waktu, karena Windows Stagger mengurangi keterlambatan atau out-of-order data dibandingkan dengan jendela Tumbling.
- Jendela Tumbling: Kueri yang menggabungkan data menggunakan jendela berbasis waktu berbeda yang membuka dan menutup secara berkala.
- Jendela Geser: Kueri yang terus menggabungkan data, menggunakan waktu tetap atau interval rowcount.

Jendela Stagger

Menggunakan jendela stagger adalah metode jendela yang cocok untuk menganalisis grup data yang tiba pada waktu yang tidak konsisten. Ini sangat cocok untuk kasus penggunaan analitik deret waktu apa pun, seperti serangkaian penjualan terkait atau catatan log.

Misalnya, Log Alur VPC memiliki jendela tangkapan sekitar 10 menit. Namun, log alur tersebut dapat memiliki jendela tangkapan hingga 15 menit jika Anda menggabungkan data di klien. Jendela stagger cocok untuk menggabungkan log ini untuk analisis.

Jendela stagger mengatasi masalah catatan terkait yang tidak masuk ke jendela yang dibatasi waktu yang sama, seperti ketika jendela tumbling digunakan.

Hasil Parsial dengan Jendela Tumbling

Ada batasan tertentu dengan menggunakanJendela Tumbling untuk menggabungkan terlambat atau terlambat atau tidak sesuai out-of-order urutan.

Jika jendela tumbling digunakan untuk menganalisis grup data terkait waktu, catatan individu mungkin masuk ke jendela terpisah. Selanjutnya hasil parsial dari setiap jendela harus digabungkan nanti untuk menghasilkan hasil yang lengkap untuk setiap grup catatan.

Dalam kueri jendela tumbling berikut, catatan dikelompokkan ke dalam jendela berdasarkan waktu baris, waktu peristiwa, dan simbol ticker:

```
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);
```

Dalam diagram berikut, aplikasi menghitung jumlah perdagangan yang diterimanya, berdasarkan waktu perdagangan terjadi (waktu peristiwa) dengan satu menit granularitas. Aplikasi ini dapat menggunakan jendela tumbling untuk mengelompokkan data berdasarkan waktu baris dan waktu peristiwa. Aplikasi menerima empat catatan yang semuanya tiba dalam satu menit. Ini mengelompokkan catatan berdasarkan waktu baris, waktu peristiwa, dan simbol ticker. Karena beberapa catatan tiba setelah jendela tumbling pertama berakhir, semua catatan tidak masuk dalam jendela tumbling satu menit yang sama.

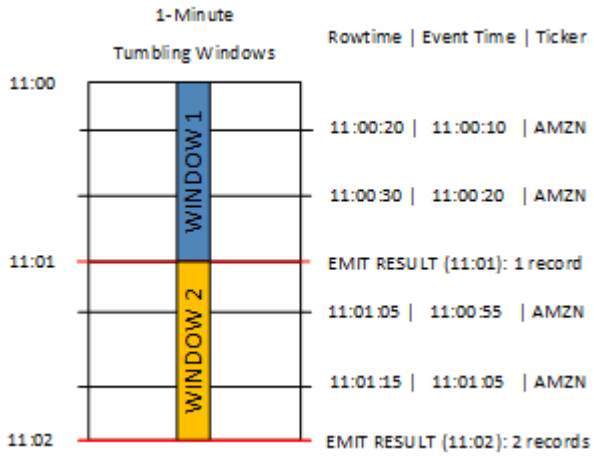


Diagram sebelumnya memiliki peristiwa berikut.

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

Hasil yang ditetapkan dari aplikasi jendela tumbling terlihat sama dengan yang berikut ini.

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

Dalam serangkaian hasil sebelumnya, tiga hasil dikembalikan:

- Catatan dengan `ROWTIME` dari 11:01:00 yang menggabungkan dua catatan pertama.
- Catatan pada 11:02:00 yang hanya menggabungkan catatan ketiga. Catatan ini memiliki `ROWTIME` dalam jendela kedua, tetapi `EVENT_TIME` dalam jendela pertama.
- Catatan pada 11:02:00 yang hanya menggabungkan catatan keempat.

Untuk menganalisis serangkaian hasil yang lengkap, catatan harus digabungkan di penyimpanan yang persisten. Hal ini menambah kompleksitas dan persyaratan pemrosesan untuk aplikasi.

Lengkapi Hasil dengan Jendela Stagger

Untuk meningkatkan akurasi analisis catatan data terkait waktu, Kinesis Data Analytics menawarkan tipe jendela baru yang disebut jendela stagger. Dalam tipe jendela ini, jendela terbuka ketika

peristiwa pertama yang cocok dengan kunci partisi tiba, dan bukan pada interval waktu yang tetap. Jendela ditutup berdasarkan masa yang ditentukan, yang diukur dari waktu ketika jendela dibuka.

Jendela stagger adalah jendela terbatas waktu terpisah untuk setiap pengelompokan kunci dalam klausa jendela. Aplikasi menggabungkan setiap hasil dari klausa jendela dalam jendela waktunya sendiri, daripada menggunakan satu jendela untuk semua hasil.

Dalam kueri jendela stagger berikut, catatan dikelompokkan ke dalam jendela berdasarkan waktu peristiwa dan simbol ticker:

```
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);
```

Dalam diagram berikut, peristiwa dikumpulkan berdasarkan waktu peristiwa dan simbol ticker ke jendela stagger.

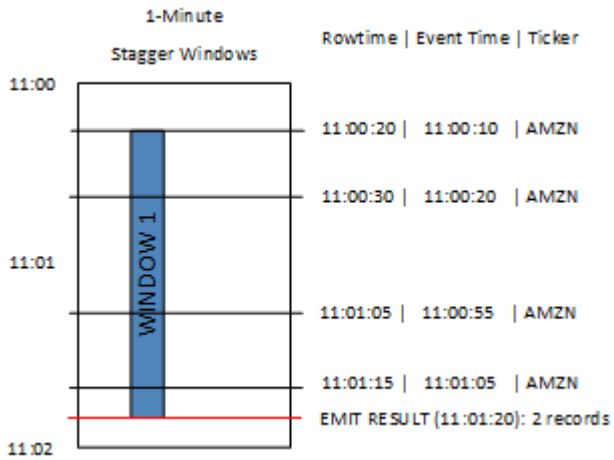


Diagram sebelumnya memiliki peristiwa berikut, yang merupakan peristiwa yang sama seperti aplikasi jendela tumbling yang dianalisis:

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

Hasil yang ditetapkan dari aplikasi jendela stagger terlihat sama dengan yang berikut ini.

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

Catatan yang dikembalikan menggabungkan tiga catatan input pertama. Catatan dikelompokkan oleh jendela stagger satu menit. Jendela stagger dimulai ketika aplikasi menerima catatan AMZN pertama (dengan `ROWTIME` dari 11:00:20). Ketika jendela stagger 1 menit berakhir (pada 11:01:20), catatan dengan hasil yang masuk dalam jendela stagger (berdasarkan `ROWTIME` dan `EVENT_TIME`) ditulis ke aliran output. Menggunakan jendela stagger, semua catatan dengan `ROWTIME` dan `EVENT_TIME` dalam jendela satu menit dipancarkan dalam satu hasil.

Catatan terakhir (dengan `EVENT_TIME` di luar agregasi satu menit) digabungkan secara terpisah. Ini karena `EVENT_TIME` adalah salah satu kunci partisi yang digunakan untuk memisahkan catatan ke serangkaian hasil, dan kunci partisi `EVENT_TIME` untuk jendela pertama adalah `11:00`.

Sintaksis untuk jendela stagger didefinisikan dalam klausa khusus, `WINDOWED BY`. Klausa ini digunakan sebagai pengganti klausa `GROUP BY` untuk agregasi streaming. Klausa muncul segera setelah klausa `WHERE` opsional dan sebelum klausa `HAVING`.

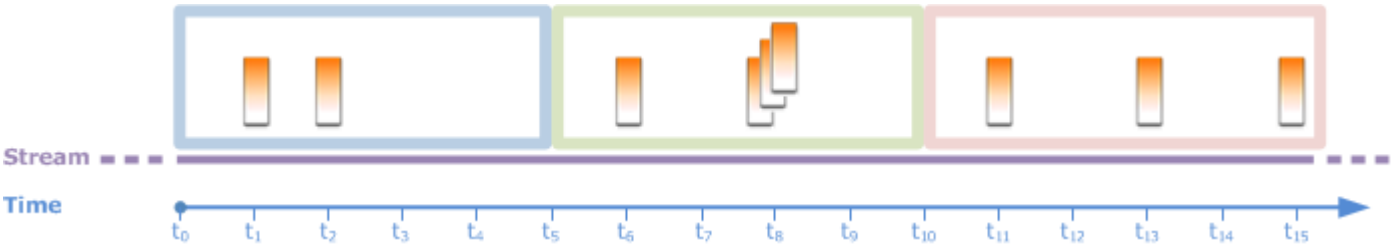
Jendela stagger didefinisikan dalam klausa `WINDOWED BY` dan mengambil dua parameter: kunci partisi dan panjang jendela. Kunci partisi membuat partisi aliran data yang masuk dan menentukan

kapan jendela terbuka. Jendela stagger terbuka saat peristiwa pertama dengan kunci partisi unik muncul di aliran. Jendela stagger tertutup setelah periode waktu yang ditetapkan oleh panjang jendela. Sintaksis ditampilkan dalam contoh kode berikut:

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

Jendela Tumbling (Agregasi Menggunakan GROUP BY)

Ketika kueri jendela memproses setiap jendela dengan cara yang tidak tumpang tindih, jendela disebut sebagai jendela tumbling. Dalam hal ini, setiap catatan di aliran dalam aplikasi termasuk dalam jendela tertentu. Catatan hanya diproses sekali (ketika kueri memproses jendela tempat catatan berada).



Misalnya, kueri agregasi yang menggunakan klausa `GROUP BY` memproses baris dalam jendela tumbling. Aliran demo dalam latihan memulai menerima data harga saham yang dipetakan ke aliran dalam aplikasi `SOURCE_SQL_STREAM_001` dalam aplikasi Anda. Aliran ini memiliki skema berikut.

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

Dalam kode aplikasi Anda, misalkan Anda ingin menemukan harga agregat (min, max) untuk setiap ticker selama jendela satu menit. Anda dapat menggunakan kueri berikut.

```
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);
```

Sebelumnya adalah contoh kueri jendela yang berbasis waktu. Kueri mengelompokkan catatan berdasarkan nilai `ROWTIME`. Untuk pelaporan per menit, fungsi `STEP` membulatkan nilai `ROWTIME` ke menit terdekat.

> **Note**
>
> Anda juga dapat menggunakan fungsi `FLOOR` untuk mengelompokkan catatan ke jendela. Namun, `FLOOR` hanya dapat membulatkan nilai waktu ke unit waktu keseluruhan (jam, menit, detik, dan sebagainya). `STEP` direkomendasikan untuk mengelompokkan catatan ke jendela tumbling karena dapat membulatkan nilai ke interval arbitrer, misalnya, 30 detik.

Kueri ini adalah contoh jendela non-tumpang tindih (tumbling). Klausa `GROUP BY` mengelompokkan catatan dalam jendela satu menit, dan setiap catatan termasuk ke jendela tertentu (tidak tumpang tindih). Kueri memancarkan satu catatan output per menit, memberikan harga ticker min/maks. yang tercatat pada menit tertentu. Tipe kueri berguna untuk membuat laporan berkala dari aliran data input. Dalam contoh ini, laporan dihasilkan setiap menit.

Untuk menguji kueri

1. Siapkan aplikasi dengan mengikuti latihan memulai.
2. Ubah pernyataan `SELECT` dalam kode aplikasi menggunakan kueri `SELECT` sebelumnya. Kode aplikasi yang dihasilkan ditampilkan sebagai berikut:

   ```
   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,
   ```

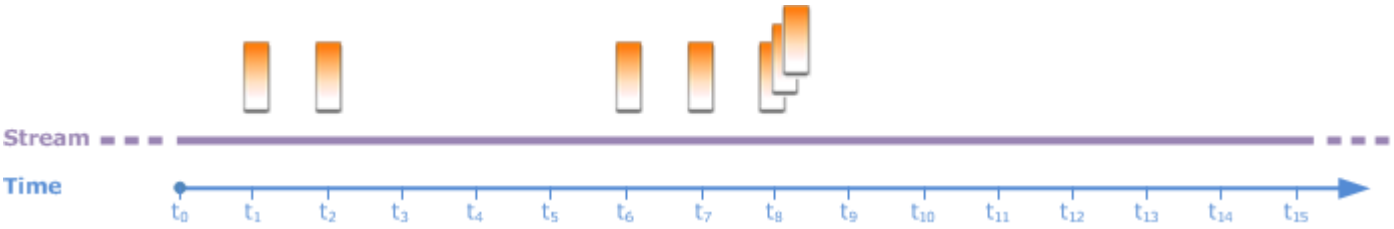
```
                MIN(Price) AS Min_Price,
                MAX(Price) AS Max_Price
FROM    "SOURCE_SQL_STREAM_001"
GROUP BY Ticker_Symbol,
         STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

Jendela Geser

Alih-alih mengelompokkan catatan menggunakan `GROUP BY`, Anda dapat menentukan jendela berbasis waktu atau berbasis baris. Anda melakukan ini dengan menambahkan klausul `WINDOW` eksplisit.

Dalam kasus ini, saat jendela meluncur seiring waktu, Amazon Kinesis Data Analytics memancarkan output saat catatan baru muncul di aliran. Kinesis Data Analytics memancarkan output ini dengan memproses baris di jendela. Windows dapat tumpang tindih dalam tipe pemrosesan ini, dan catatan dapat menjadi bagian dari beberapa jendela dan diproses dengan setiap jendela. Contoh berikut menggambarkan jendela geser.

Pertimbangkan kueri sederhana yang menghitung catatan di aliran. Contoh ini mengasumsikan jendela 5 detik. Dalam contoh aliran berikut, catatan baru tiba pada waktu t_1, t_2, t_6, dan t_7, serta tiga catatan tiba pada waktu t_8 detik.



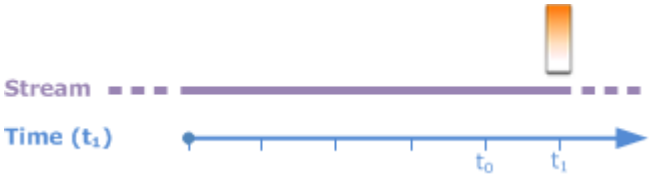
Ingatlah hal-hal berikut ini:

- Contoh mengasumsikan jendela 5 detik. Jendela 5 detik terus meluncur seiring waktu.
- Untuk setiap baris yang memasuki jendela, baris output dipancarkan oleh jendela geser. Segera setelah aplikasi dimulai, Anda melihat kueri memancarkan output untuk setiap catatan baru yang muncul di aliran, meskipun jendela 5 detik belum diteruskan. Sebagai contoh, kueri memancarkan output ketika catatan muncul di detik pertama dan detik kedua. Selanjutnya, kueri memproses catatan di jendela 5 detik.
- Jendela meluncur seiring waktu. Jika catatan lama di aliran masuk ke jendela, kueri tidak memancarkan output kecuali juga ada catatan baru di aliran yang masuk dalam jendela 5 detik.

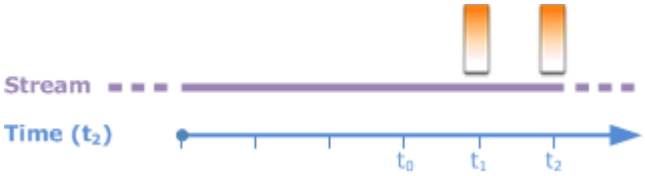
Misalkan kueri mulai mengeksekusi di t_0. Kemudian hal berikut terjadi:

1. Pada waktu t_0, kueri dimulai. Kueri tidak memancarkan output (nilai jumlah) karena saat ini tidak ada catatan.

2. Pada waktu t_1, catatan baru muncul di aliran, dan kueri memancarkan nilai hitungan 1.

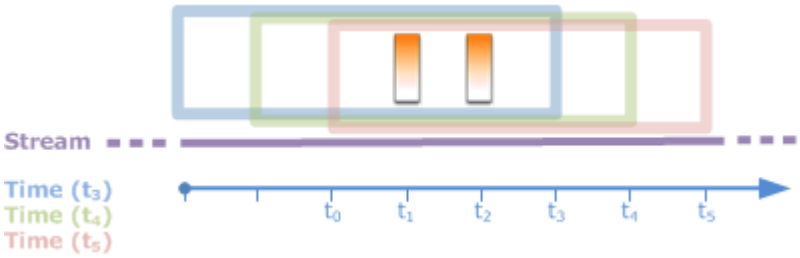


3. Pada waktu t_2, catatan baru muncul di aliran, dan kueri memancarkan hitungan 2.

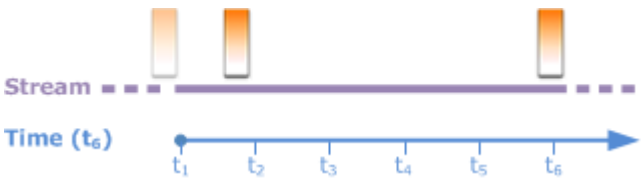


4. Jendela 5 detik meluncur seiring waktu:
   - Di t_3, jendela geser t_3 ke t_0
   - Di t4 (jendela geser t_4 ke t_0)
   - Di t_5, jendela geser t_5–t_0

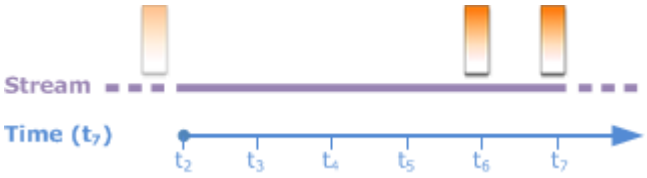
   Pada semua waktu ini, jendela 5 detik memiliki catatan yang sama—tidak ada catatan baru. Oleh karena itu, kueri tidak memancarkan output apa pun.



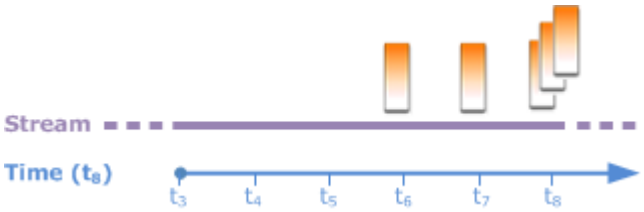
5. Pada waktu t_6, jendela 5 detik adalah (t_6 ke t_1). Kueri mendeteksi satu catatan baru di t_6 sehingga memancarkan output 2. Catatan di t_1 tidak lagi berada di jendela dan tidak masuk hitungan.



6. Pada waktu t_7, jendela 5 detik adalah t_7 ke t_2. Kueri mendeteksi satu catatan baru di t_7 sehingga memancarkan output 2. Catatan di t_2 tidak lagi di jendela 5 detik, dan oleh karena itu tidak dihitung.



7. Pada waktu t_8, jendela 5 detik adalah t_8 ke t_3. Kueri mendeteksi tiga catatan baru, dan oleh karena itu memancarkan catatan hitungan 5.



Singkatnya, jendela adalah ukuran tetap dan meluncur seiring waktu. Kueri memancarkan output ketika catatan baru muncul.

> **Note**
>
> Sebaiknya gunakan jendela geser tidak lebih dari satu jam. Jika Anda menggunakan jendela yang lebih lama, aplikasi memerlukan waktu lebih lama untuk memulai ulang setelah pemeliharaan sistem reguler. Hal ini karena sumber data harus dibaca kembali dari aliran.

Contoh kueri berikut menggunakan klausa `WINDOW` untuk menentukan jendela dan melakukan agregat. Karena kueri tidak menentukan `GROUP BY`, kueri menggunakan pendekatan jendela geser untuk memproses catatan di aliran.

Contoh 1: Proses Streaming menggunakan Jendela Geser 1 Menit

Pertimbangkan aliran demo dalam latihan Memulai yang mengisi aliran dalam aplikasi, `SOURCE_SQL_STREAM_001`. Berikut ini adalah skemanya.

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

Misalkan Anda ingin aplikasi Anda menghitung agregat menggunakan jendela geser 1 menit. Yaitu, untuk setiap catatan baru yang muncul di aliran, Anda ingin aplikasi memancarkan output dengan menerapkan agregat pada catatan di jendela 1 menit sebelumnya.

Anda dapat menggunakan kueri jendela berbasis waktu berikut. Kueri menggunakan klausul `WINDOW` untuk menentukan interval rentang 1 menit. `PARTITION BY` di klausa `WINDOW` mengelompokkan catatan berdasarkan nilai ticker dalam jendela geser.

```
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);
```

Untuk menguji kueri

1. Siapkan aplikasi dengan mengikuti Latihan Memulai.
2. Ubah pernyataan `SELECT` dalam kode aplikasi dengan kueri `SELECT` sebelumnya. Kode aplikasi yang dihasilkan adalah sebagai berikut.

```
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);
```

Contoh 2: Kueri yang Menerapkan Agregat di Jendela Geser

Kueri pada aliran demo berikut mengembalikan rata-rata perubahan persen dalam harga setiap ticker dalam jendela 10 detik.

```
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);
```

Untuk menguji kueri

1. Siapkan aplikasi dengan mengikuti Latihan Memulai.
2. Ubah pernyataan SELECT dalam kode aplikasi dengan kueri SELECT sebelumnya. Kode aplikasi yang dihasilkan adalah sebagai berikut.

```
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);
```

Contoh 3: Kueri Data dari Beberapa Jendela Geser di Aliran yang Sama

Anda dapat menulis kueri untuk memancarkan output tempat setiap nilai kolom dihitung menggunakan jendela geser yang berbeda yang didefinisikan melalui aliran yang sama.

Pada contoh berikut, kueri memancarkan ticker output, harga, a2, dan a10. Ini memancarkan output untuk simbol ticker yang rata-rata pergerakan dua barisnya melintasi rata-rata pergerakan sepuluh baris. Nilai kolom a2 dan a10 diturunkan dari jendela geser dua baris dan sepuluh baris.

```
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);
```

Untuk menguji kueri ini terhadap aliran demo, ikuti prosedur pengujian yang dijelaskan dalam Contoh 1.

Operasi Data Streaming: Gabungan Streaming

Anda dapat memiliki beberapa aliran dalam aplikasi di aplikasi Anda. Anda dapat menulis kueri `JOIN` untuk menghubungkan data yang tiba di aliran ini. Misalnya, anggap Anda memiliki aliran dalam aplikasi berikut:

- OrderStream— Menerima pesanan stok yang ditempatkan.

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

- TradeStream— Menerima perdagangan saham yang dihasilkan untuk pesanan tersebut.

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

Berikut ini adalah contoh kueri `JOIN` yang menghubungkan data di aliran ini.

Contoh 1: Laporkan Pesanan Jika Ada Perdagangan dalam Satu Menit Setelah Pesanan Ditempatkan

Dalam contoh ini, kueri Anda bergabung dengan `OrderStream` dan `TradeStream`. Namun, karena kita hanya menginginkan perdagangan yang ditempatkan satu menit setelah perintah, kueri mendefinisikan jendela 1 menit melalui `TradeStream`. Untuk informasi tentang kueri jendela, lihat Jendela Geser.

```
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;
```

Anda dapat menentukan jendela secara eksplisit menggunakan klausa `WINDOW` dan menulis kueri sebelumnya sebagai berikut:

```
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)
```

Ketika Anda menyertakan kueri ini dalam kode aplikasi Anda, kode aplikasi berjalan terus menerus. Untuk setiap catatan yang tiba di `OrderStream`, aplikasi memancarkan output jika ada perdagangan dalam jendela 1 menit setelah pesanan ditempatkan.

Gabung dengan kueri sebelumnya adalah gabungan dalam tempat kueri memancarkan catatan di `OrderStream` ketika ada catatan yang cocok di `TradeStream` (dan sebaliknya). Dengan menggunakan gabuangan luar, Anda dapat membuat skenario menarik lainnya. Misalkan Anda menginginkan pesanan saham yang tidak ada perdagangan selama satu menit setelah pesanan saham ditempatkan, dan perdagangan yang dilaporkan dalam jendela yang sama, tetapi untuk beberapa pesanan lainnya. Ini adalah contoh gabungan luar.

```
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;
```

Migrasi ke Layanan Terkelola untuk Contoh Apache Flink Studio

Contoh berikut menunjukkan cara memigrasikan Kinesis Data Analytics untuk aplikasi SQL ke Managed Service untuk Apache Flink Studio.

Mereplikasi Kinesis Data Analytics untuk Kueri SQL di Managed Service untuk Apache Flink Studio

> **Warning**
>
> Untuk proyek baru, kami menyarankan Anda menggunakan Managed Service untuk Apache Flink Studio melalui Kinesis Data Analytics untuk Aplikasi SQL. Layanan Terkelola untuk Apache Flink Studio menggabungkan kemudahan penggunaan dengan kemampuan analitis tingkat lanjut, memungkinkan Anda membangun aplikasi pemrosesan aliran yang canggih dalam hitungan menit.

Untuk memigrasikan beban kerja Anda ke Managed Service for Apache Flink Studio atau Managed Service for Apache Flink, bagian ini menyediakan terjemahan kueri yang dapat Anda gunakan untuk kasus penggunaan umum.

> **Note**
>
> Layanan Terkelola untuk Apache Flink dan Managed Service untuk Apache Flink Studio menawarkan fitur pemrosesan aliran data lanjutan yang tidak tersedia dalam aplikasi Kinesis Data Analytics berbasis SQL. Ini termasuk semantik pemrosesan tepat sekali, jendela waktu acara, ekstensibilitas menggunakan fungsi yang ditentukan pengguna dan integrasi khusus, dukungan bahasa imperatif, status aplikasi tahan lama, penskalaan horizontal, dukungan untuk berbagai sumber data, integrasi yang dapat diperluas, dan banyak lagi. Ini sangat penting untuk memastikan akurasi, kelengkapan, konsistensi, dan keandalan pemrosesan aliran data.

Sebelum Anda menjelajahi contoh-contoh ini, kami sarankan Anda meninjau terlebih dahulu Menggunakan notebook Studio dengan Managed Service for Apache Flink.

Topik

- Membuat ulang Kinesis Data Analytics untuk kueri SQL di Managed Service untuk Apache Flink Studio

Membuat ulang Kinesis Data Analytics untuk kueri SQL di Managed Service untuk Apache Flink Studio

Tabel berikut menyediakan terjemahan kueri aplikasi Kinesis Data Analytics berbasis SQL yang umum ke Managed Service for Apache Flink Studio.

Aplikasi Multi-Langkah

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;
```

Mengubah nilai DateTime

SQL-based Kinesis Data Analytics application

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

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

Managed Service for Apache Flink Studio

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

```
   EVENT_SECOND INT)

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

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

Peringatan sederhana

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;
```

Peringatan terhambat

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;
```

Mengagregasikan Hasil Sebagian dari Kueri

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;
```

Mengubah nilai string

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;
```

Mengganti substring menggunakan 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;
```

Penguraian log regex

SQL-based Kinesis Data Analytics application

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

Managed Service for Apache Flink Studio

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

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

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

Mengubah nilai DateTime

SQL-based Kinesis Data Analytics application

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

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

Managed Service for Apache Flink Studio

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

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

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

Windows dan agregasi

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;
```

Jendela Tumbling menggunakan 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;
```

Mengambil nilai yang paling sering terjadi (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;
```

Perkiraan item Top-K

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
```

Mengurai Log Web (Fungsi 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;
```

Pisahkan String menjadi Beberapa Bidang (Fungsi 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;
```

Bergabung

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;
```

Kesalahan

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;
```

Migrasi beban kerja Random Cut Forest

Jika Anda ingin memindahkan beban kerja yang menggunakan Random Cut Forest dari Kinesis Analytics untuk SQL ke Managed Service untuk Apache Flink, posting blog AWS ini menunjukkan

cara menggunakan Managed Service untuk Apache Flink untuk menjalankan algoritma RCF online untuk deteksi anomali.

Mengganti Kinesis Data Firehose sebagai sumber dengan Kinesis Data Streams

Lihat Converting-kdasql-kdastudio/ untuk tutorial lengkap.

Dalam latihan berikut, Anda akan mengubah aliran data Anda untuk menggunakan Amazon Managed Service untuk Apache Flink Studio. Ini juga berarti beralih dari Amazon Kinesis Data Firehose ke Amazon Kinesis Data Streams.

Pertama, kami membagikan arsitektur KDA-SQL yang khas, sebelum menunjukkan bagaimana Anda dapat mengganti ini menggunakan Amazon Managed Service untuk Apache Flink Studio dan Amazon Kinesis Data Streams. Atau Anda dapat meluncurkan AWS CloudFormation template di sini:

Analisis Data Amazon Kinesis - SQL dan Amazon Kinesis Data Firehose

Berikut adalah alur arsitektur SQL Amazon Kinesis Data Analytics:

Kami pertama-tama memeriksa penyiapan Amazon Kinesis Data Analytics-SQL dan Amazon Kinesis Data Firehose yang lama. Kasus penggunaan adalah pasar perdagangan di mana data perdagangan, termasuk ticker saham dan harga, mengalir dari sumber eksternal ke sistem Amazon Kinesis. Amazon Kinesis Data Analytics untuk SQL menggunakan aliran input untuk mengeksekusi kueri Windowed seperti jendela Tumbling untuk menentukan volume perdagangan dan`max`, `average` dan `min` harga perdagangan selama satu menit untuk setiap ticker saham.

Amazon Kinesis Data Analytics-SQL diatur untuk menelan data dari Amazon Kinesis Data Firehose API. Setelah diproses, Amazon Kinesis Data Analytics-SQL mengirimkan data yang diproses ke Amazon Kinesis Data Firehose lainnya, yang kemudian menyimpan output dalam bucket Amazon S3.

Dalam hal ini, Anda menggunakan Amazon Kinesis Data Generator. Amazon Kinesis Data Generator memungkinkan Anda mengirim data pengujian ke Amazon Kinesis Data Streams atau aliran

pengiriman Amazon Kinesis Data Firehose. Untuk memulai, silakan ikuti instruksi di sini. Gunakan AWS CloudFormation templat di sini sebagai pengganti yang disediakan dalam instruksi:.

Setelah Anda menjalankan AWS CloudFormation template, bagian output akan memberikan url Amazon Kinesis Data Generator. Masuk ke portal menggunakan id pengguna Cognito dan kata sandi yang Anda atur di sini. Pilih Wilayah dan nama aliran target. Untuk status saat ini, pilih aliran Pengiriman Amazon Kinesis Data Firehose. Untuk status baru, pilih nama Amazon Kinesis Data Firehose Streams. Anda dapat membuat beberapa template, tergantung pada kebutuhan Anda, dan menguji template menggunakan tombol Uji template sebelum mengirimnya ke aliran target.

Berikut ini adalah contoh payload menggunakan Amazon Kinesis Data Generator. Generator data menargetkan input Amazon Kinesis Firehose Streams untuk mengalirkan data secara terus menerus. Klien Amazon Kinesis SDK dapat mengirim data dari produsen lain juga.

```
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
```

JSON berikut digunakan untuk menghasilkan serangkaian waktu dan tanggal perdagangan acak, ticker saham, dan harga saham:

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

Setelah Anda memilih Kirim data, generator akan mulai mengirim data tiruan.

Sistem eksternal mengalirkan data ke Amazon Kinesis Data Firehose. Menggunakan Amazon Kinesis Data Analytics untuk Aplikasi SQL, Anda dapat menganalisis data streaming menggunakan SQL standar. Layanan ini memungkinkan Anda untuk membuat dan menjalankan kode SQL terhadap sumber streaming untuk melakukan analitik deret waktu, memberi umpan dasbor waktu nyata, dan membuat metrik waktu nyata. Amazon Kinesis Data Analytics untuk Aplikasi SQL dapat membuat aliran tujuan dari kueri SQL pada aliran input dan mengirim aliran tujuan ke Amazon Kinesis Data Firehose lainnya. Tujuan Amazon Kinesis Data Firehose dapat mengirim data analitik ke Amazon S3 sebagai status akhir.

Kode warisan Amazon Kinesis Data Analytics-SQL didasarkan pada perpanjangan Standar SQL.

Anda menggunakan kueri berikut di Amazon Kinesis Data Analytics-SQL. Anda pertama kali membuat aliran tujuan untuk output kueri. Kemudian, Anda akan menggunakanPUMP, yang merupakan Objek Repositori Amazon Kinesis Data Analytics (perpanjangan dari Standar SQL) yang menyediakan fungsionalitas kueri yang terus `INSERT INTO stream SELECT ... FROM` berjalan, sehingga memungkinkan hasil kueri untuk terus dimasukkan ke dalam aliran bernama.

```
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);
```

SQL di atas menggunakan dua jendela waktu — `tradeTimestamp` yang berasal dari payload aliran masuk dan juga `ROWTIME.tradeTimestamp` disebut atau. `Event Time` `client-side time` Waktu ini sering kali berguna ketika digunakan dalam analitik karena merupakan waktu ketika peristiwa terjadi. Namun, banyak sumber peristiwa, seperti ponsel dan klien web, tidak memiliki jam yang dapat diandalkan, yang dapat menyebabkan waktu yang tidak akurat. Selain itu, masalah konektivitas dapat menyebabkan catatan yang muncul di aliran tidak dalam urutan yang sama dengan peristiwa yang terjadi.

Aliran dalam aplikasi juga menyertakan kolom khusus yang disebut. `ROWTIME` Kolom ini menyimpan stempel waktu ketika Amazon Kinesis Data Analytics memasukkan baris di aliran dalam aplikasi pertama. `ROWTIME` mencerminkan stempel waktu tempat Amazon Kinesis Data Analytics memasukkan catatan ke aliran dalam aplikasi pertama setelah membaca dari sumber streaming. Nilai `ROWTIME` ini selanjutnya dipertahankan di seluruh aplikasi Anda.

SQL menentukan jumlah ticker sebagai`volume`,`min`, `max` dan `average` harga selama interval 60 detik.

Menggunakan setiap waktu ini dalam kueri jendela yang berbasis waktu memiliki kelebihan dan kekurangan. Pilih satu atau lebih dari waktu-waktu ini, dan strategi untuk menangani kerugian yang relevan berdasarkan skenario kasus penggunaan Anda.

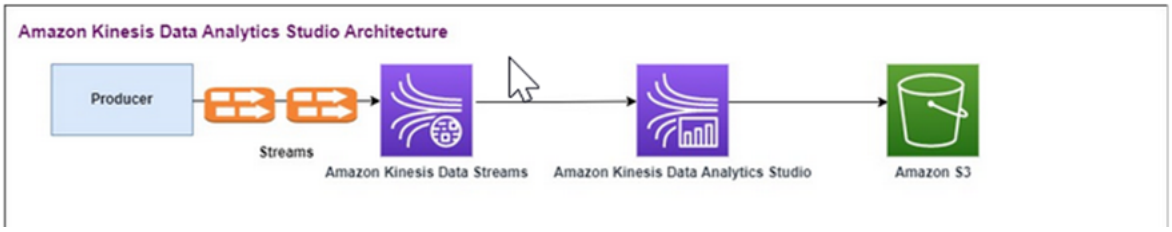
Strategi dua jendela menggunakan dua waktu berbasis, keduanya `ROWTIME` dan salah satu waktu lainnya seperti waktu acara.

- Gunakan `ROWTIME` sebagai jendela pertama, yang mengontrol seberapa sering kueri memancarkan hasil, seperti yang ditunjukkan dalam contoh berikut. Ini tidak digunakan sebagai waktu logis.
- Gunakan salah satu waktu lain yang merupakan waktu logis yang ingin Anda kaitkan dengan analitik Anda. Waktu ini mewakili kapan peristiwa terjadi. Pada contoh berikut, tujuan analitik adalah mengelompokkan catatan dan dan kembali menghitung dengan ticker

Layanan Terkelola Amazon untuk Apache Flink Studio

Dalam arsitektur yang diperbarui, Anda mengganti Amazon Kinesis Data Firehose dengan Amazon Kinesis Data Streams. Amazon Kinesis Data Analytics untuk Aplikasi SQL digantikan oleh Amazon Managed Service untuk Apache Flink Studio. Kode Apache Flink dijalankan secara interaktif dalam Notebook Apache Zeppelin. Amazon Managed Service untuk Apache Flink Studio mengirimkan data perdagangan agregat ke bucket Amazon S3 untuk penyimpanan. Langkah-langkahnya ditunjukkan sebagai berikut:

Berikut adalah Amazon Managed Service untuk alur arsitektur Apache Flink Studio:



Buat Aliran Data Kinesis

Untuk membuat aliran data menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di bilah navigasi, perluas pemilih Wilayah dan pilih Wilayah.
3. Pilih Create data stream (Buat aliran data).
4. Pada halaman Create Kinesis stream, masukkan nama untuk aliran data Anda dan terima mode kapasitas On-Demand default.

   Dengan mode On-Demand, Anda kemudian dapat memilih Create Kinesis stream untuk membuat aliran data Anda.

   Pada halaman Kinesis streams, Status streaming Anda adalah Membuat saat aliran sedang dibuat. Saat aliran siap digunakan, Status berubah menjadi Aktif.
5. Pilih nama streaming Anda. Halaman Detail Stream menampilkan ringkasan konfigurasi aliran Anda, bersama dengan informasi pemantauan.
6. Di Amazon Kinesis Data Generator, ubah Stream/Delivery stream ke Amazon Kinesis Data Streams baru: TRADE_SOURCE_STREAM.

   JSON dan Payload akan sama seperti yang Anda gunakan untuk Amazon Kinesis Data Analytics-SQL. Gunakan Amazon Kinesis Data Generator untuk menghasilkan beberapa contoh data muatan perdagangan dan menargetkan Aliran Data TRADE_SOURCE_STREAM untuk latihan ini:

   ```
   {{date.now(YYYY-MM-DD HH:mm:ss.SSS)}},
   "{{random.arrayElement(["AAPL","AMZN","MSFT","META","GOOGL"])}}",
   {{random.number(2000)}}
   ```
7. Saat AWS Management Console pergi ke Managed Service for Apache Flink dan kemudian pilih Create Application.
8. Di panel navigasi kiri, pilih buku catatan Studio lalu pilih Buat buku catatan studio.
9. Masukkan nama untuk notebook studio.
10. Di bawah database AWS Glue, sediakan AWS Glue database yang ada yang akan menentukan metadata untuk sumber dan tujuan Anda. Jika Anda tidak memiliki AWS Glue database, pilih Buat dan lakukan hal berikut:

a. Di konsol AWS Glue, pilih Databases di bawah katalog Data dari menu sebelah kiri.
b. Pilih Buat database
c. Di halaman Buat database, masukkan nama untuk database. Di bagian Lokasi - opsional, pilih Jelajahi Amazon S3 dan pilih bucket Amazon S3. Jika Anda belum memiliki bucket Amazon S3 yang sudah disiapkan, Anda dapat melewati langkah ini dan kembali lagi nanti.
d. (Opsional). Masukkan deskripsi untuk database.
e. Pilih Buat basis data.

11. Pilih Buat buku catatan
12. Setelah buku catatan Anda dibuat, pilih Jalankan.
13. Setelah notebook berhasil di-staeted, luncurkan notebook Zeppelin dengan memilih Open in Apache Zeppelin.
14. Pada halaman Notebook Zeppelin, pilih Buat catatan baru dan beri nama. MarketDataFeed

Kode SQL Flink dijelaskan berikut, tetapi pertama-tama inilah tampilan layar notebook Zeppelin. Setiap jendela di dalam notebook adalah blok kode terpisah, dan mereka dapat dijalankan satu per satu.

Layanan Terkelola Amazon untuk Kode Apache Flink Studio

Amazon Managed Service untuk Apache Flink Studio menggunakan Notebook Zeppelin untuk menjalankan kode. Pemetaan dilakukan untuk contoh ini ke kode ssql berdasarkan Apache Flink 1.13. Kode di Notebook Zeppelin ditampilkan di bawah satu blok pada satu waktu.

Sebelum menjalankan kode apa pun di Notebook Zeppelin Anda, perintah konfigurasi Flink harus dijalankan. Jika Anda perlu mengubah pengaturan konfigurasi apa pun setelah menjalankan kode (ssql, Python, atau Scala), Anda harus berhenti dan memulai ulang notebook Anda. Dalam contoh ini, Anda perlu mengatur checkpointing. Checkpointing diperlukan agar Anda dapat mengalirkan data ke file di Amazon S3. Ini memungkinkan streaming data ke Amazon S3 untuk dibuang ke file. Pernyataan di bawah ini menetapkan interval menjadi 5000 milidetik.

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

`%flink.conf`menunjukkan bahwa blok ini adalah pernyataan konfigurasi. Untuk informasi selengkapnya tentang konfigurasi Flink termasuk checkpointingg, lihat Apache Flink Checkpointing.

Tabel input untuk sumber Amazon Kinesis Data Streams dibuat dengan kode ssql Flink di bawah ini. Perhatikan bahwa TRADE_TIME bidang menyimpan tanggal/waktu yang dibuat oleh generator data.

```
%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');
```

Anda dapat melihat aliran masukan dengan pernyataan ini:

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

select * from TRADE_SOURCE_STREAM;
```

Sebelum mengirim data agregat ke Amazon S3, Anda dapat melihatnya langsung di Amazon Managed Service untuk Apache Flink Studio dengan kueri pilih jendela tumbling. Ini mengumpulkan data perdagangan dalam jendela waktu satu menit. Perhatikan bahwa pernyataan %flink.ssql harus memiliki penunjukan (type=update):

```
%flink.ssql(type=update)

select TUMBLE_ROWTIME(TRADE_TIME,
INTERVAL '1' MINUTE) as TRADE_WINDOW,
TICKER, COUNT(*) as VOLUME,
AVG(PRICE) as AVG_PRICE,
MIN(PRICE) as MIN_PRICE,
MAX(PRICE) as MAX_PRICE FROM TRADE_SOURCE_STREAMGROUP BY TUMBLE(TRADE_TIME, INTERVAL
 '1' MINUTE), TICKER;
```

Anda kemudian dapat membuat tabel untuk tujuan di Amazon S3. Anda perlu menggunakan watermark. Tanda air adalah metrik kemajuan yang menunjukkan titik waktu ketika Anda yakin bahwa tidak ada lagi peristiwa yang tertunda yang akan tiba. Alasan tanda air adalah untuk memperhitungkan kedatangan yang terlambat. Interval ‘5’ Second ini memungkinkan perdagangan untuk memasuki Amazon Kinesis Data Stream terlambat 5 detik dan masih

disertakan jika mereka memiliki stempel waktu di dalam jendela. Untuk informasi selengkapnya lihat Menghasilkan Tanda Air.

```
%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');
```

Pernyataan ini menyisipkan data ke dalam. `TRADE_DESTINATION_S3` `TUMPLE_ROWTIME`adalah stempel waktu dari batas atas inklusif dari jendela yang jatuh.

```
%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;
```

Biarkan pernyataan Anda berjalan selama 10 hingga 20 menit untuk mengumpulkan beberapa data di Amazon S3. Kemudian batalkan pernyataan Anda.

Ini menutup file di Amazon S3 sehingga dapat dilihat.

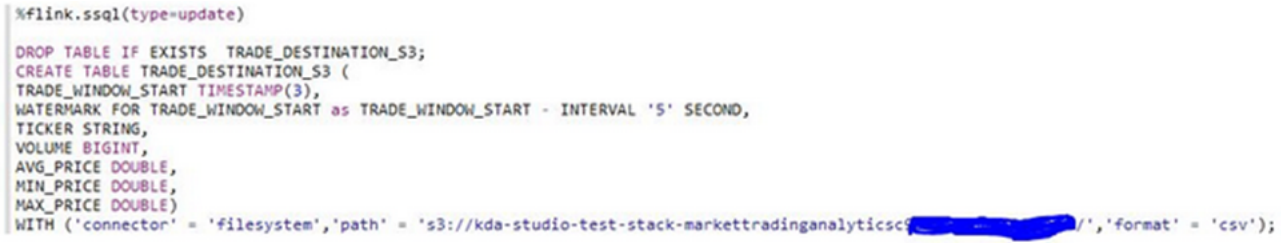
Berikut adalah apa isinya terlihat seperti:

Anda dapat menggunakan AWS CloudFormationtemplate untuk membuat infrastruktur.

AWS CloudFormationakan membuat sumber daya berikut di AWS akun Anda:

- Amazon Kinesis Data Streams
- Layanan Terkelola Amazon untuk Apache Flink Studio
- Basis data Amazon Glue
- Bucket Amazon S3
- Peran dan kebijakan IAM untuk Amazon Managed Service untuk Apache Flink Studio untuk mengakses sumber daya yang sesuai

Impor notebook dan ubah nama bucket Amazon S3 dengan bucket Amazon S3 baru yang dibuat oleh. AWS CloudFormation



Lihat lebih

Berikut adalah beberapa sumber daya tambahan yang dapat Anda gunakan untuk mempelajari lebih lanjut tentang penggunaan Layanan Terkelola untuk Apache Flink Studio:

- Layanan Terkelola untuk Panduan Pengembang Notebook Apache Flink Studio
- Apache Flink 1.13 Dokumentasi
- Layanan Terkelola untuk Workshop Apache Flink Studio
- Jendela Apache Flink
- Panduan Pengembang Amazon Kinesis Data Analytics — Menulis dari Stream Kinesis Data Analytics ke Bucket S3

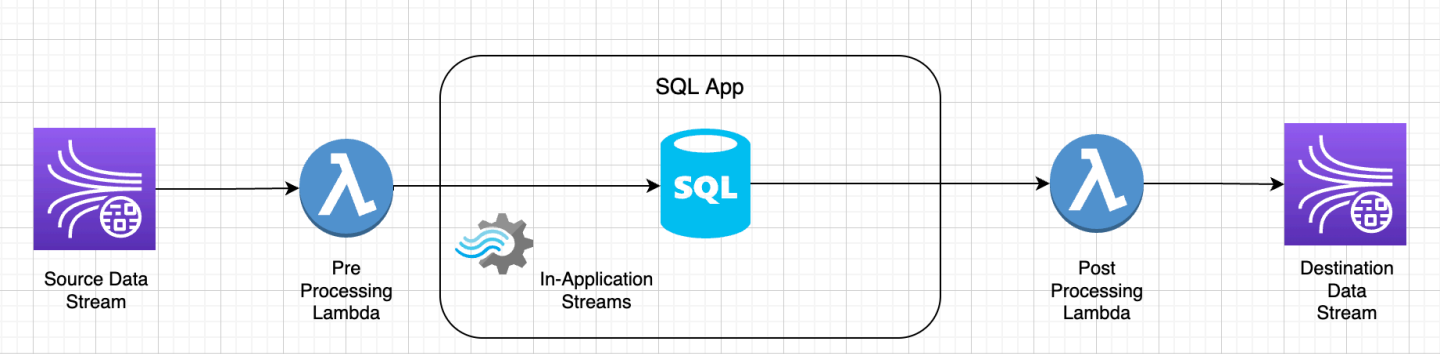
Memanfaatkan fungsi yang ditentukan pengguna (UDF)

Tujuan dari pola ini adalah untuk menunjukkan bagaimana memanfaatkan UDF dalam buku catatan Kinesis Data Analytics-Studio Zeppelin untuk memproses data dalam aliran Kinesis. Layanan Terkelola untuk Apache Flink Studio menggunakan Apache Flink untuk menyediakan kemampuan analitis tingkat lanjut, termasuk semantik pemrosesan yang tepat sekali, jendela waktu acara, ekstensibilitas menggunakan fungsi yang ditentukan pengguna dan integrasi pelanggan, dukungan bahasa imperatif, status aplikasi yang tahan lama, penskalaan horizontal, dukungan untuk beberapa sumber data, integrasi yang dapat diperluas, dan banyak lagi. Ini sangat penting untuk memastikan akurasi, kelengkapan, konsistensi, dan keandalan pemrosesan aliran data dan tidak tersedia dengan Amazon Kinesis Data Analytics untuk SQL.

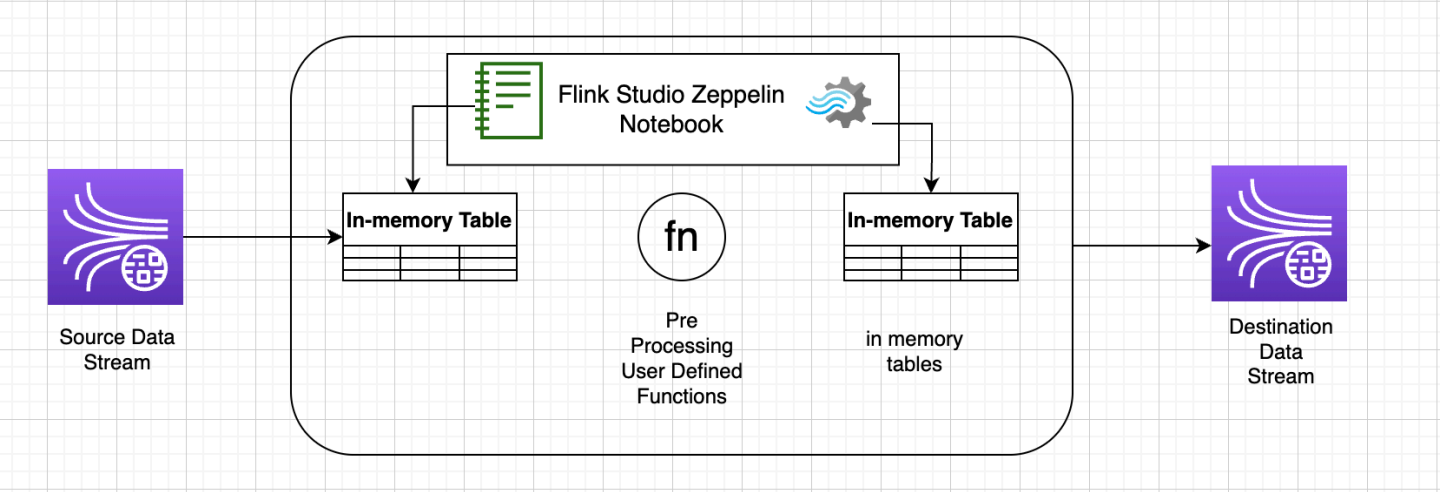
Dalam contoh aplikasi ini, kami akan mendemonstrasikan cara memanfaatkan UDF di notebook KDA-Studio Zeppelin untuk memproses data dalam aliran Kinesis. Notebook Studio untuk Kinesis Data Analytics memungkinkan Anda secara interaktif mengkueri aliran data secara langsung, dan dengan mudah membangun serta menjalankan aplikasi pemrosesan aliran menggunakan SQL, Python, dan Scala standar. Dengan beberapa klikAWS Management Console, Anda dapat meluncurkan notebook tanpa server untuk menanyakan aliran data dan mendapatkan hasil dalam hitungan detik. Untuk

informasi selengkapnya, lihat Menggunakan notebook Studio dengan Kinesis Data Analytics untuk Apache Flink.

Fungsi Lambda yang digunakan untuk pemrosesan data pra/pasca dalam aplikasi KDA-SQL:



Fungsi yang ditentukan pengguna untuk pemrosesan data pra/pasca menggunakan notebook KDA-Studio Zeppelin



Fungsi yang ditentukan pengguna (UDF)

Untuk menggunakan kembali logika bisnis umum menjadi operator, akan berguna untuk mereferensikan fungsi yang ditentukan pengguna untuk mengubah aliran data Anda. Ini dapat dilakukan baik dalam Managed Service for Apache Flink Studio notebook, atau sebagai file jar aplikasi yang direferensikan secara eksternal. Memanfaatkan fungsi yang ditentukan pengguna dapat menyederhanakan transformasi atau pengayaan data yang mungkin Anda lakukan melalui streaming data.

Di notebook Anda, Anda akan mereferensikan jar aplikasi Java sederhana yang memiliki fungsi untuk menganonimkan nomor telepon pribadi. Anda juga dapat menulis Python atau Scala UDF untuk

digunakan dalam notebook. Kami memilih jar aplikasi Java untuk menyoroti fungsionalitas mengimpor jar aplikasi ke notebook Pyflink.

Pengaturan lingkungan

Untuk mengikuti panduan ini dan berinteraksi dengan data streaming Anda, Anda akan menggunakan AWS CloudFormation skrip untuk meluncurkan sumber daya berikut:

- Sumber dan target Kinesis Data Streams
- Database Glue
- IAM role
- Layanan Terkelola untuk Aplikasi Apache Flink Studio
- Fungsi Lambda untuk memulai Managed Service untuk Apache Flink Studio Application
- Peran Lambda untuk menjalankan fungsi Lambda di atas
- Sumber daya khusus untuk menjalankan fungsi Lambda

Unduh AWS CloudFormation template di sini.

Buat AWS CloudFormation tumpukan

1. Pergi ke AWS Management Console dan pilih CloudFormationdi bawah daftar layanan.
2. Pada CloudFormationhalaman, pilih Stacks dan kemudian pilih Create Stack dengan sumber daya baru (standar).
3. Pada halaman Buat tumpukan, pilih Unggah File Template, lalu pilih `kda-flink-udf.yml` yang Anda unduh sebelumnya. Unggah file dan kemudian pilih Berikutnya.
4. Beri nama template, seperti `kinesis-UDF` agar mudah diingat, dan perbarui Parameter input seperti input-stream jika Anda menginginkan nama yang berbeda. Pilih Selanjutnya.
5. Pada halaman Configure stack options, tambahkan Tag jika Anda mau, lalu pilih Next.
6. Pada halaman Tinjauan, centang kotak yang memungkinkan pembuatan sumber daya IAM dan kemudian pilih Kirim.

AWS CloudFormationTumpukan mungkin membutuhkan waktu 10 hingga 15 menit untuk diluncurkan tergantung pada Wilayah yang Anda luncurkan. Setelah Anda melihat `CREATE_COMPLETE` status untuk seluruh tumpukan, Anda siap untuk melanjutkan.

Bekerja dengan Layanan Terkelola untuk notebook Apache Flink Studio

Notebook studio untuk Kinesis Data Analytics memungkinkan Anda melakukan kueri aliran data secara interaktif secara real time, dan dengan mudah membangun dan menjalankan aplikasi pemrosesan aliran menggunakan SQL, Python, dan Scala standar. Dengan beberapa klikAWS Management Console, Anda dapat meluncurkan notebook tanpa server untuk menanyakan aliran data dan mendapatkan hasil dalam hitungan detik.

Notebook adalah lingkungan pengembangan berbasis web. Dengan notebook, Anda mendapatkan pengalaman pengembangan interaktif sederhana yang dikombinasikan dengan kemampuan pemrosesan aliran data canggih yang disediakan oleh Apache Flink. Notebook studio menggunakan notebook yang didukung Apache Zeppelin, dan menggunakan Apache Flink sebagai mesin pemrosesan aliran. Notebook studio menggabungkan teknologi ini dengan mulus untuk membuat analitik lanjutan pada aliran data dapat diakses oleh pengembang dari semua keahlian.

Apache Zeppelin memberi notebook Studio Anda dengan rangkaian alat analitik lengkap, termasuk yang berikut:

- Visualisasi Data
- Mengekspor data ke file
- Mengontrol format output untuk analisis yang lebih mudah

Menggunakan notebook

1. Pergi ke AWS Management Console dan pilih Amazon Kinesis di bawah daftar layanan.
2. Di halaman navigasi sebelah kiri, pilih aplikasi Analytics, lalu pilih buku catatan Studio.
3. Verifikasi bahwa KinesisDataAnalyticsStudionotebook sedang berjalan.
4. Pilih notebook dan kemudian pilih Buka di Apache Zeppelin.
5. Unduh file Data Producer Zeppelin Notebook yang akan Anda gunakan untuk membaca dan memuat data ke dalam Kinesis Stream.
6. Impor `Data Producer` Notebook Zeppelin. Pastikan untuk memodifikasi input `STREAM_NAME` dan `REGION` dalam kode notebook. Nama aliran input dapat ditemukan di output AWS CloudFormation tumpukan.
7. Jalankan notebook Data Producer dengan memilih tombol Jalankan paragraf ini untuk menyisipkan data sampel ke dalam input Kinesis Data Stream.

8. Saat data sampel dimuat, unduh MaskPhoneNumber-Interactive notebook, yang akan membaca data input, menganonimkan nomor telepon dari aliran input dan menyimpan data anonim ke dalam aliran output.
9. Impor notebook `MaskPhoneNumber-interactive` Zeppelin.
10. Jalankan setiap paragraf di buku catatan.

    a. Dalam paragraf 1, Anda mengimpor Fungsi yang Ditetapkan Pengguna untuk menganonimkan nomor telepon.

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

    b. Pada paragraf berikutnya, Anda membuat tabel dalam memori untuk membaca data aliran input. Pastikan nama dan AWS wilayah streaming sudah benar.

    ```
    %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. Periksa apakah data dimuat ke dalam tabel dalam memori.

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

    d. Memanggil fungsi yang ditentukan pengguna untuk menganonimkan nomor telepon.

    ```
    %flink.ssql(type=update)
    ```

```
select customer_id, product, review, MaskPhoneNumber('mask_phone', phone) as
 phoneNumber from customer_reviews
```

e. Sekarang setelah nomor telepon ditutup, buat tampilan dengan nomor bertopeng.

```
%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. Verifikasi datanya.

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

g. Buat tabel dalam memori untuk Output Kinesis Stream. Pastikan nama stream dan AWS Region sudah benar.

```
%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. Masukkan catatan yang diperbarui di Aliran Kinesis target.

```
%flink.ssql(type=update)
```

```
INSERT INTO customer_reviews_stream_table
SELECT customer_id, product, review, phoneNumber
FROM sentiments_view
```

i. Lihat dan verifikasi data dari Aliran Kinesis target.

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

Mempromosikan notebook sebagai aplikasi

Sekarang setelah Anda menguji kode notebook Anda secara interaktif, Anda akan menyebarkan kode sebagai aplikasi streaming dengan status tahan lama. Anda harus terlebih dahulu memodifikasi konfigurasi Aplikasi untuk menentukan lokasi kode Anda di Amazon S3.

1. PadaAWS Management Console, pilih buku catatan Anda dan di Deploy sebagai konfigurasi aplikasi - opsional, pilih Edit.
2. Di bawah Tujuan untuk kode di Amazon S3, pilih bucket Amazon S3 yang dibuat oleh skrip. AWS CloudFormation Prosesnya mungkin memakan waktu beberapa menit.
3. Anda tidak akan dapat mempromosikan catatan apa adanya. Jika Anda mencoba, Anda akan mengalami kesalahan karena `Select` pernyataan tidak didukung. Untuk mencegah masalah ini, unduh Notebook MaskPhoneNumber-Streaming Zeppelin.
4. Impor `MaskPhoneNumber-streaming` Notebook Zeppelin.
5. Buka catatan dan pilih Tindakan untuk KinesisDataAnalyticsStudio.
6. Pilih Build MaskPhoneNumber -Streaming dan ekspor ke S3. Pastikan untuk mengganti nama Nama Aplikasi dan tidak menyertakan karakter khusus.
7. Pilih Bangun dan Ekspor. Ini akan memakan waktu beberapa menit untuk mengatur Aplikasi Streaming.
8. Setelah build selesai, pilih Deploy using AWS console.
9. Pada halaman berikutnya, tinjau pengaturan dan pastikan untuk memilih peran IAM yang benar. Selanjutnya, pilih Buat aplikasi streaming.
10. Setelah beberapa menit, Anda akan melihat pesan bahwa aplikasi streaming berhasil dibuat.

Untuk informasi selengkapnya tentang penerapan aplikasi dengan status dan batas tahan lama, lihat Menerapkan sebagai aplikasi dengan status tahan lama.

Pembersihan

Secara opsional, Anda sekarang dapat menghapus tumpukan. AWS CloudFormation Ini akan menghapus semua layanan yang Anda atur sebelumnya.

Kinesis Data Analytics untuk contoh SQL

Bagian ini memberikan contoh membuat dan bekerja dengan aplikasi di Amazon Kinesis Data Analytics. Mereka menyertakan contoh kode dan step-by-step instruksi untuk membantu Anda membuat aplikasi Kinesis Data Analytics dan menguji hasil Anda.

Sebelum menjelajahi contoh-contoh ini, sebaiknya tinjau Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya dan Memulai dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL terlebih dulu.

Topik

- Contoh: Mengubah Data
- Contoh: Jendela dan Agregasi
- Contoh: Bergabung
- Contoh: Machine Learning
- Contoh: Peringatan dan Kesalahan
- Contoh: Akselerator Solusi

Contoh: Mengubah Data

Ada kalanya kode aplikasi Anda harus memproses catatan masuk terlebih dulu sebelum melakukan analitik apa pun di Amazon Kinesis Data Analytics. Hal ini dapat terjadi karena berbagai alasan, seperti saat catatan tidak sesuai dengan format catatan yang didukung, sehingga mengakibatkan kolom yang tidak dinormalkan di aliran input dalam aplikasi.

Bagian ini memberikan contoh cara menggunakan fungsi string yang tersedia untuk menormalkan data, cara mengekstrak informasi yang Anda butuhkan dari kolom string, dan seterusnya. Bagian ini juga menunjuk ke fungsi waktu tanggal yang mungkin Anda anggap berguna.

Memproses Aliran Terlebih Dulu dengan Lambda

Untuk informasi tentang praprosesing stream dengan AWS Lambda, lihat. Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda

Topik

- Contoh: Mengubah Nilai String

- Contoh: Mengubah Nilai DateTime
- Contoh: Mengubah Beberapa Tipe Data

Contoh: Mengubah Nilai String

Amazon Kinesis Data Analytics mendukung format seperti JSON dan CSV untuk catatan pada sumber streaming. Untuk detailnya, lihat RecordFormat. Catatan ini kemudian dipetakan ke baris di aliran dalam aplikasi sesuai dengan konfigurasi input. Untuk detailnya, lihat Mengonfigurasi Input Aplikasi. Konfigurasi input menentukan bagaimana bidang catatan di sumber streaming dipetakan ke kolom di aliran dalam aplikasi.

Pemetaan ini berfungsi ketika catatan pada sumber streaming mengikuti format yang didukung, yang menghasilkan aliran dalam aplikasi dengan data yang dinormalkan. Namun bagaimana jika data pada sumber streaming Anda tidak sesuai dengan standar yang didukung? Misalnya, bagaimana jika sumber streaming Anda berisi data seperti data clickstream, sensor IoT, dan log aplikasi?

Pertimbangkan contoh berikut:

- Sumber streaming berisi log aplikasi – Log aplikasi mengikuti format log Apache standar, dan ditulis ke aliran menggunakan format JSON.

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

  Untuk informasi selengkapnya tentang format log Apache standar, lihat File Log di situs web Apache.

- Sumber streaming berisi data semi-terstruktur – Contoh berikut menunjukkan dua catatan. Nilai bidang `Col_E_Unstructured` adalah serangkaian nilai yang dipisahkan koma. Ada lima kolom: empat pertama memiliki nilai tipe string, dan kolom terakhir berisi nilai yang dipisahkan koma.

  ```
  { "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"}
```

- Catatan di sumber streaming berisi URL, dan Anda memerlukan sebagian nama domain URL untuk analitik.

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

Dalam kasus tersebut, proses dua langkah berikut umumnya berfungsi untuk membuat aliran dalam aplikasi yang berisi data yang dinormalkan:

1. Konfigurasikan input aplikasi untuk memetakan bidang tidak terstruktur ke kolom dari tipe `VARCHAR(N)` di aliran input dalam aplikasi yang dibuat.
2. Dalam kode aplikasi Anda, gunakan fungsi string untuk membagi kolom tunggal ini menjadi beberapa kolom, lalu menyimpan baris di aliran dalam aplikasi lain. Aliran dalam aplikasi yang dibuat kode Anda ini akan memiliki data yang dinormalkan. Anda kemudian dapat melakukan analisis pada aliran dalam aplikasi ini.

Amazon Kinesis Data Analytics menyediakan operasi string berikut, fungsi SQL standar, dan ekstensi ke standar SQL untuk bekerja dengan kolom string:

- Operator string – Operator seperti `LIKE` dan `SIMILAR` berguna dalam membandingkan string. Untuk informasi selengkapnya, lihat String Operator di Amazon Managed Service for Apache Flink SQL Reference.
- Fungsi SQL – Fungsi berikut berguna ketika memanipulasi string individual. Untuk informasi selengkapnya, lihat Fungsi String dan Pencarian di Amazon Managed Service for Apache Flink SQL Reference.
  - `CHAR_LENGTH` – Menyediakan panjang string.
  - `INITCAP` – Mengembalikan versi yang dikonversi dari string input sehingga karakter pertama dari setiap kata yang dipisahkan spasi adalah huruf besar, dan semua karakter lainnya adalah huruf kecil.
  - `LOWER/UPPER` – Mengonversi string ke huruf kecil atau huruf besar.

  - `OVERLAY` – Mengganti sebagian argumen string pertama (string asli) dengan argumen string kedua (string pengganti).
  - `POSITION` – Mencari string dalam string lain.
  - `REGEX_REPLACE` – Mengganti substring dengan substring alternatif.
  - `SUBSTRING` – Mengekstrak sebagian string sumber yang dimulai dari posisi tertentu.
  - `TRIM` – Menghapus instans karakter tertentu dari awal atau akhir string sumber.
- Ekstensi SQL – Ini berguna untuk bekerja dengan string tidak terstruktur seperti log dan URI. Untuk informasi selengkapnya, lihat Log Parsing Functions di Amazon Managed Service for Apache Flink SQL Reference.
  - `FAST_REGEX_LOG_PARSER` – Berfungsi mirip dengan parser regex, tetapi membutuhkan beberapa pintasan untuk memastikan hasil yang lebih cepat. Sebagai contoh, parser regex cepat berhenti pada pencocokan pertama yang ditemukannya (dikenal sebagai semantik nonaktif).
  - `FIXED_COLUMN_LOG_PARSE` – Menguraikan bidang dengan lebar tetap dan secara otomatis mengonversinya ke tipe SQL yang diberikan.
  - `REGEX_LOG_PARSE` – Menguraikan string berdasarkan pola ekspresi reguler Java default.
  - `SYS_LOG_PARSE` – Menguraikan entri yang umum ditemukan di log sistem UNIX/Linux.
  - `VARIABLE_COLUMN_LOG_PARSE` – Membagi string input ke dalam bidang yang dipisahkan oleh karakter pembatas atau string pembatas.
  - `W3C_LOG_PARSE` – Dapat digunakan untuk memformat log Apache dengan cepat.

Untuk contoh menggunakan fungsi ini, lihat topik berikut:

Topik

- Contoh: Mengekstrak Bagian String (Fungsi SUBSTRING)
- Contoh: Mengganti Substring menggunakan Regex (Fungsi REGEX_REPLACE)
- Contoh: Menguraikan String Log Berdasarkan Ekspresi Reguler (Fungsi REGEX_LOG_PARSE)
- Contoh: Mengurai Log Web (Fungsi W3C_LOG_PARSE)
- Contoh: Bagi String menjadi Beberapa Bidang (Fungsi VARIABLE_COLUMN_LOG_PARSE)

Contoh: Mengekstrak Bagian String (Fungsi SUBSTRING)

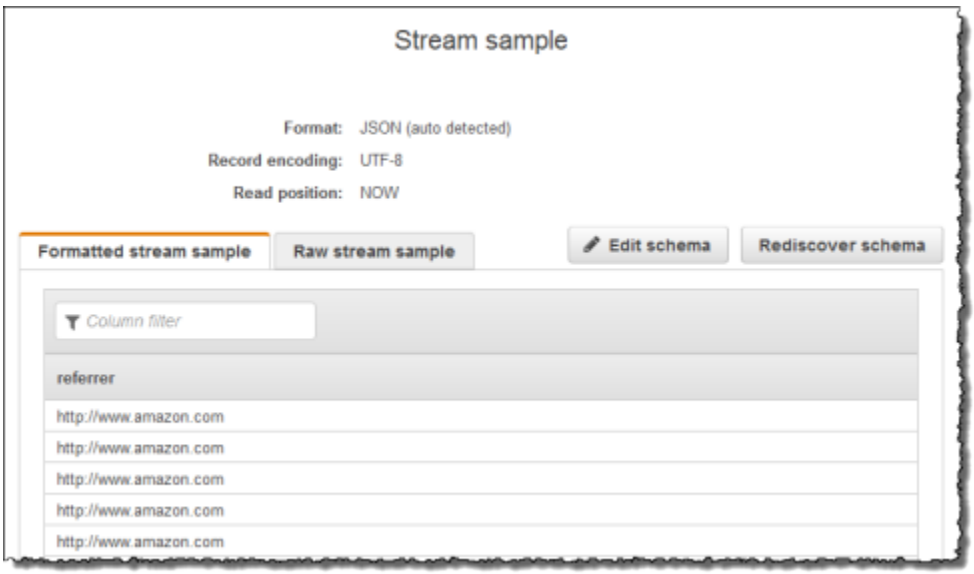
Contoh menggunakan fungsi `SUBSTRING` untuk mengubah string di Amazon Kinesis Data Analytics. Fungsi `SUBSTRING` mengekstrak sebagian string sumber yang dimulai dari posisi tertentu. Untuk

informasi selengkapnya, lihat SUBSTRING di Amazon Managed Service for Apache Flink SQL Reference.

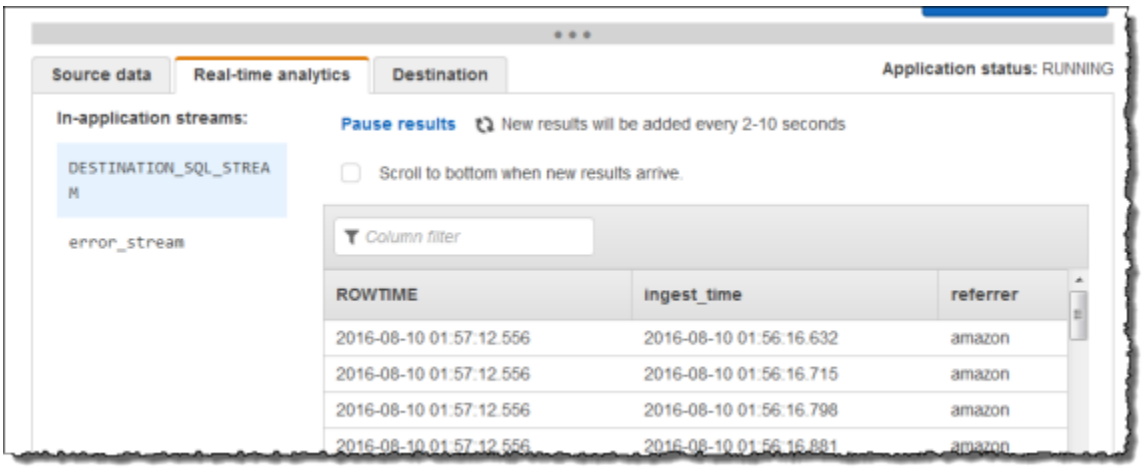
Dalam contoh ini, Anda menulis catatan berikut ke Amazon Kinesis data stream.

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

Anda kemudian membuat aplikasi Kinesis Data Analytics di konsol, menggunakan aliran data Kinesis sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan satu kolom (REFERRER), seperti yang ditunjukkan.



Kemudian, Anda menggunakan kode aplikasi dengan fungsi SUBSTRING untuk mengurai string URL untuk mengambil nama perusahaan. Kemudian Anda memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan di bawah ini:



Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan log sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Jalankan kode Python berikut untuk mengisi catatan log sampel. Kode sederhana ini terus menulis catatan log yang sama ke aliran.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Selanjutnya, buat aplikasi Kinesis Data Analytics sebagai berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming).
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:
   a. Pilih aliran yang Anda buat di bagian sebelumnya.
   b. Pilih opsi untuk membuat IAM role.
   c. Pilih Discover schema (Temukan skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan hanya memiliki satu kolom.
   d. Jangan pilih Save and continue (Simpan dan lanjutkan).
5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:
   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
-- 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. Pilih Save and run SQL (Simpan dan jalankan SQL). Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

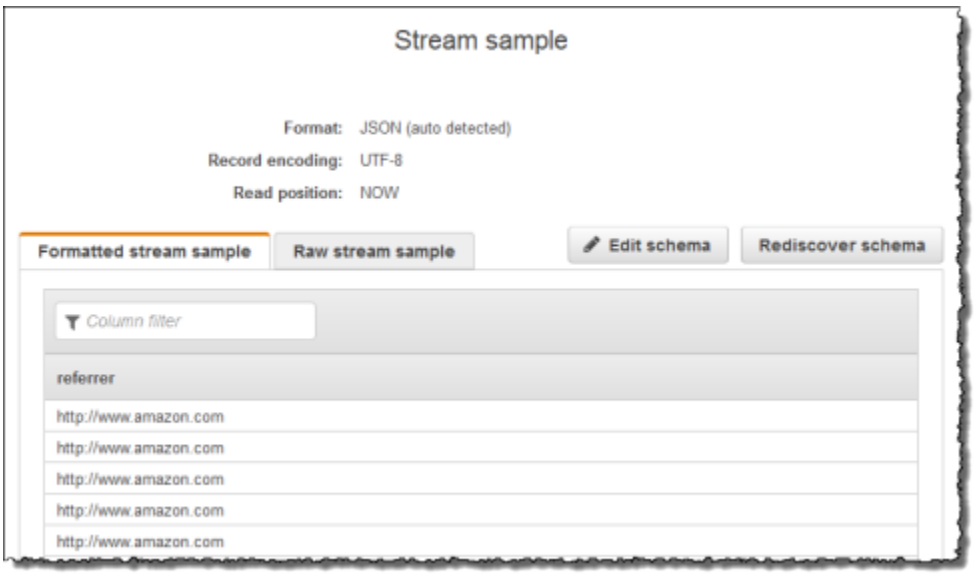
Contoh: Mengganti Substring menggunakan Regex (Fungsi REGEX_REPLACE)

Contoh ini menggunakan fungsi REGEX_REPLACE untuk mengubah string di Amazon Kinesis Data Analytics. REGEX_REPLACE mengganti substring dengan substring alternatif. Untuk informasi selengkapnya, lihat REGEX_REPLACE di Amazon Managed Service for Apache Flink SQL Reference.

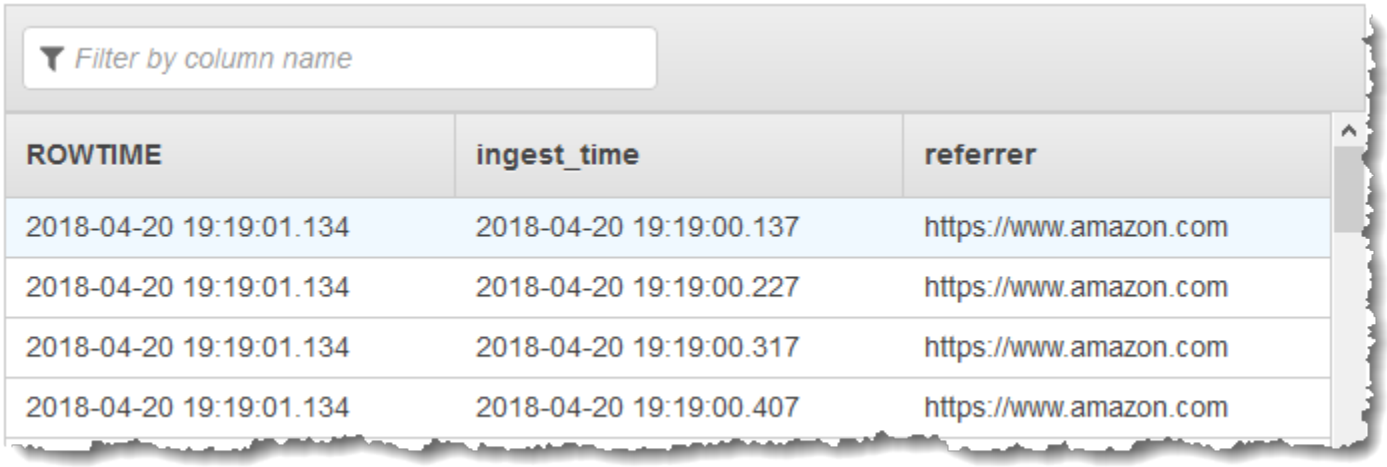
Dalam contoh ini, Anda menulis catatan berikut ke Amazon Kinesis data stream:

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

Anda kemudian membuat aplikasi Kinesis Data Analytics di konsol, dengan aliran data Kinesis sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan satu kolom (REFERRER) seperti yang ditunjukkan.



Kemudian, Anda menggunakan kode aplikasi dengan fungsi `REGEX_REPLACE` untuk mengonversi URL untuk menggunakan `https://` sebagai ganti `http://`. Anda memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan di bawah ini:



ROWTIME	ingest_time	referrer
2018-04-20 19:19:01.134	2018-04-20 19:19:00.137	https://www.amazon.com
2018-04-20 19:19:01.134	2018-04-20 19:19:00.227	https://www.amazon.com
2018-04-20 19:19:01.134	2018-04-20 19:19:00.317	https://www.amazon.com
2018-04-20 19:19:01.134	2018-04-20 19:19:00.407	https://www.amazon.com

Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan log sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Jalankan kode Python berikut untuk mengisi catatan log sampel. Kode sederhana ini terus menulis catatan log yang sama ke aliran.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Selanjutnya, buat aplikasi Kinesis Data Analytics sebagai berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.

2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming).
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:

   a. Pilih aliran yang Anda buat di bagian sebelumnya.

   b. Pilih opsi untuk membuat IAM role.

   c. Pilih Discover schema (Temukan skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan hanya memiliki satu kolom.

   d. Jangan pilih Save and continue (Simpan dan lanjutkan).

5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi dan verifikasi hasilnya sebagai berikut:

   a. Salin kode aplikasi berikut, dan tempelkan ke editor:

```
-- 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. Pilih Save and run SQL (Simpan dan jalankan SQL). Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

Contoh: Menguraikan String Log Berdasarkan Ekspresi Reguler (Fungsi REGEX_LOG_PARSE)

Contoh ini menggunakan fungsi `REGEX_LOG_PARSE` untuk mengubah string di Amazon Kinesis Data Analytics. `REGEX_LOG_PARSE` menguraikan string berdasarkan pola ekspresi reguler Java default. Untuk informasi selengkapnya, lihat [REGEX_LOG_PARSE](#) di Amazon Managed Service for Apache Flink SQL Reference.

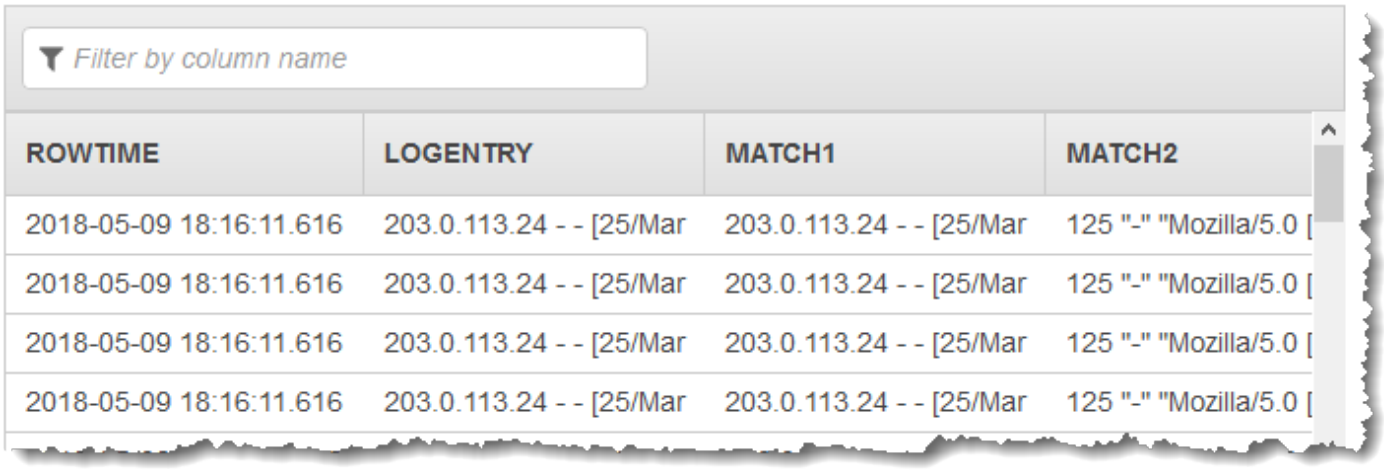
Dalam contoh ini, Anda menulis catatan berikut ke Amazon Kinesis stream:

```
{"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\""}
...
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di konsol, dengan aliran data Kinesis sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan satu kolom (LOGENTRY), seperti yang ditunjukkan berikut.

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"

Kemudian, Anda menggunakan kode aplikasi dengan fungsi `REGEX_LOG_PARSE` untuk mengurai string log untuk mengambil elemen data. Anda memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan dalam tangkapan layar bawah ini:



Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan log sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Jalankan kode Python berikut untuk mengisi catatan log sampel. Kode sederhana ini terus menulis catatan log yang sama ke aliran.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Selanjutnya, buat aplikasi Kinesis Data Analytics sebagai berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), dan tentukan nama aplikasi.
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming).
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:
    a. Pilih aliran yang Anda buat di bagian sebelumnya.
    b. Pilih opsi untuk membuat IAM role.
    c. Pilih Discover schema (Temukan skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan hanya memiliki satu kolom.
    d. Jangan pilih Save and continue (Simpan dan lanjutkan).
5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:

a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL). Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

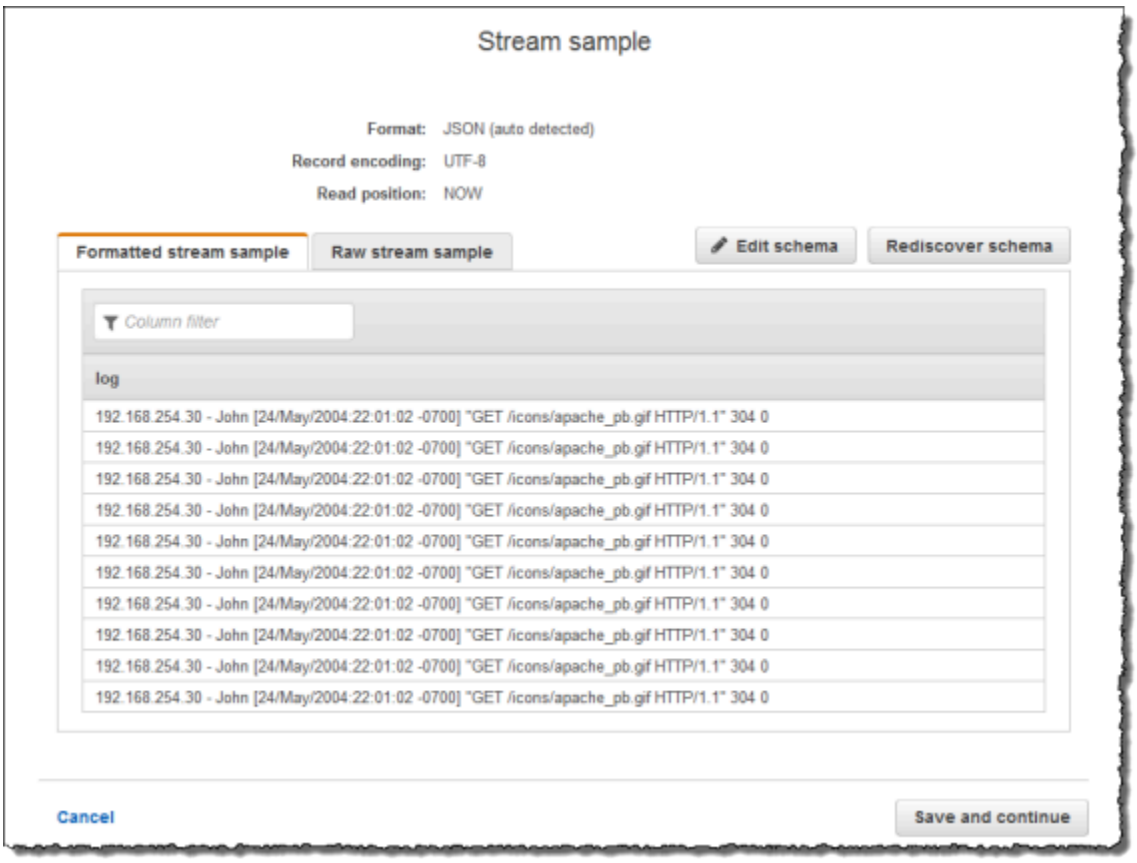
Contoh: Mengurai Log Web (Fungsi W3C_LOG_PARSE)

Contoh menggunakan fungsi `W3C_LOG_PARSE` untuk mengubah string di Amazon Kinesis Data Analytics. Anda dapat menggunakan `W3C_LOG_PARSE` untuk memformat log Apache dengan cepat. Untuk informasi selengkapnya, lihat W3C_LOG_PARSE di Amazon Managed Service for Apache Flink SQL Reference.

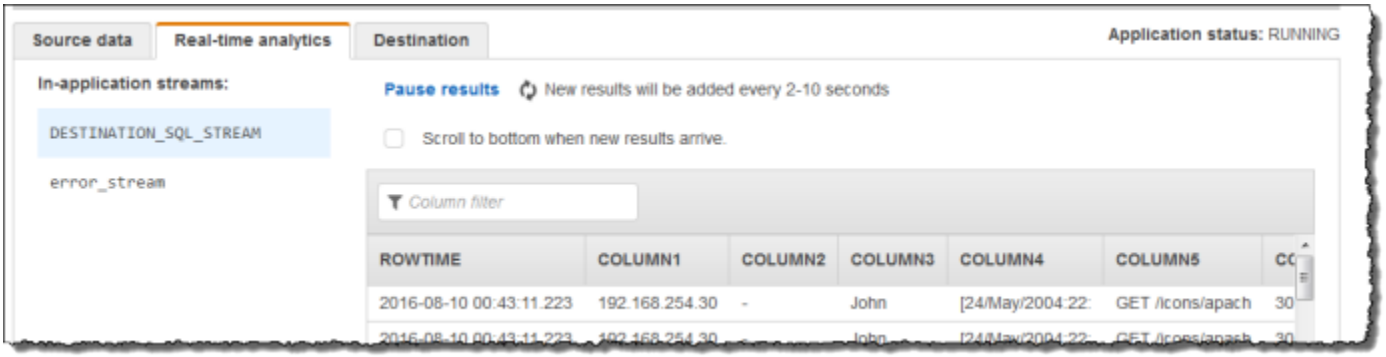
Dalam contoh ini, Anda menulis catatan log ke Amazon Kinesis data stream. Contoh log ditampilkan sebagai berikut:

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

Anda kemudian membuat aplikasi Kinesis Data Analytics di konsol, dengan aliran data Kinesis sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan satu kolom (log), seperti yang ditunjukkan di bawah ini:



Kemudian, Anda menggunakan kode aplikasi dengan fungsi `W3C_LOG_PARSE` untuk mengurai log, dan membuat aliran dalam aplikasi lainnya dengan berbagai bidang log di kolom terpisah, seperti yang ditunjukkan di bawah ini:



Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream, dan isi catatan log sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Jalankan kode Python berikut untuk mengisi catatan log sampel. Kode sederhana ini terus menulis catatan log yang sama ke aliran.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Buat aplikasi Kinesis Data Analytics sebagai berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming).
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:
   a. Pilih aliran yang Anda buat di bagian sebelumnya.
   b. Pilih opsi untuk membuat IAM role.
   c. Pilih Discover schema (Temukan skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan hanya memiliki satu kolom.
   d. Jangan pilih Save and continue (Simpan dan lanjutkan).
5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:
   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL). Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

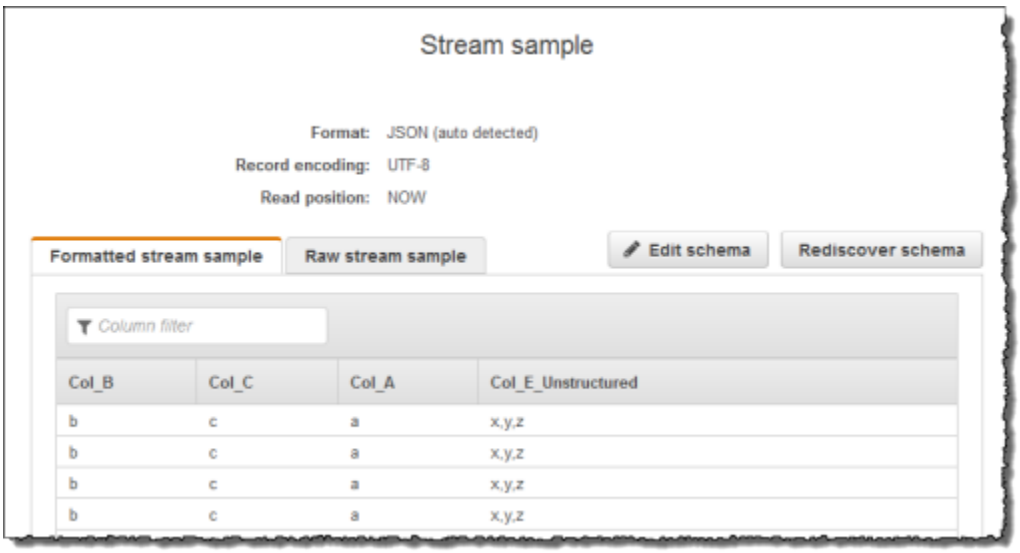
Contoh: Bagi String menjadi Beberapa Bidang (Fungsi VARIABLE_COLUMN_LOG_PARSE)

Contoh ini menggunakan fungsi `VARIABLE_COLUMN_LOG_PARSE` untuk memanipulasi string di Kinesis Data Analytics. `VARIABLE_COLUMN_LOG_PARSE` membagi string input ke dalam bidang yang dipisahkan oleh karakter pembatas atau string pembatas. Untuk informasi selengkapnya, lihat VARIABLE_COLUMN_LOG_PARSE di Amazon Managed Service for Apache Flink SQL Reference.

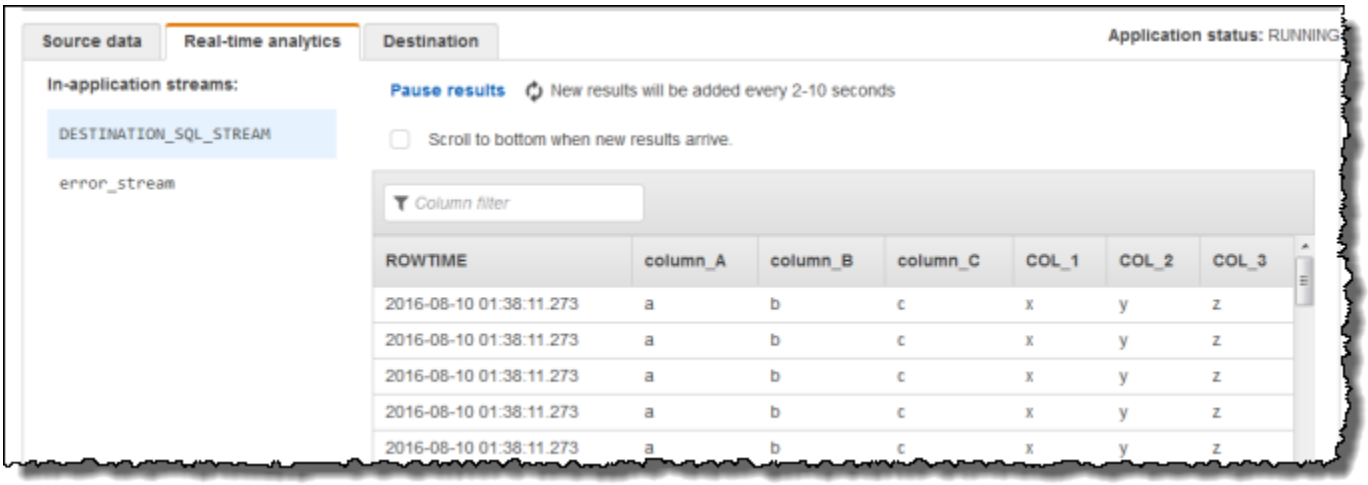
Dalam contoh ini, Anda menulis catatan semi-terstruktur ke Amazon Kinesis data stream. Contoh catatan adalah sebagai berikut:

```
{ "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"}
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di konsol, menggunakan aliran Kinesis sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan empat kolom, seperti yang ditunjukkan di bawah ini:



Kemudian, Anda menggunakan kode aplikasi dengan fungsi `VARIABLE_COLUMN_LOG_PARSE` untuk mengurai nilai yang dipisahkan koma, dan memasukkan baris yang dinormalkan di aliran dalam aplikasi lainnya, seperti yang ditampilkan di bawah ini:



Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan log sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Jalankan kode Python berikut untuk mengisi catatan log sampel. Kode sederhana ini terus menulis catatan log yang sama ke aliran.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Buat aplikasi Kinesis Data Analytics sebagai berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.

2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming).
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:

   a. Pilih aliran yang Anda buat di bagian sebelumnya.

   b. Pilih opsi untuk membuat IAM role.

   c. Pilih Discover schema (Temukan skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Perhatikan bahwa skema yang disimpulkan hanya memiliki satu kolom.

   d. Jangan pilih Save and continue (Simpan dan lanjutkan).

5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya:

   a. Salin kode aplikasi berikut dan tempelkan ke editor:

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL). Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

Contoh: Mengubah Nilai DateTime

Amazon Kinesis Data Analytics mendukung konversi kolom menjadi stempel waktu. Misalnya, Anda mungkin ingin menggunakan stempel waktu Anda sendiri sebagai bagian dari klausa `GROUP BY` sebagai jendela berbasis waktu lainnya, selain kolom `ROWTIME`. Kinesis Data Analytics menyediakan operasi dan fungsi SQL untuk bekerja dengan bidang tanggal dan waktu.

- Operator tanggal dan waktu – Anda dapat melakukan operasi aritmetika pada tanggal, waktu, dan tipe data interval. Untuk informasi selengkapnya, lihat Operator Tanggal, Stempel Waktu, dan Interval di Amazon Managed Service for Apache Flink SQL Reference.

- Fungsi SQL – Ini termasuk hal berikut. Untuk informasi selengkapnya, lihat Fungsi Tanggal dan Waktu di Amazon Managed Service for Apache Flink SQL Reference.
  - `EXTRACT()` – Mengekstrak satu bidang dari ekspresi date, time, timestamp, atau interval.
  - `CURRENT_TIME` – Mengembalikan waktu ketika kueri dijalankan (UTC).
  - `CURRENT_DATE` – Mengembalikan tanggal ketika kueri dijalankan (UTC).
  - `CURRENT_TIMESTAMP` – Mengembalikan stempel waktu ketika kueri dijalankan (UTC).
  - `LOCALTIME` – Mengembalikan waktu saat ini ketika kueri dijalankan sebagaimana ditetapkan oleh lingkungan tempat Amazon Kinesis Data Analytics berjalan (UTC).
  - `LOCALTIMESTAMP` – Mengembalikan stempel waktu saat ini sebagaimana ditetapkan oleh lingkungan tempat Amazon Kinesis Data Analytics berjalan (UTC).

- Ekstensi SQL – Ini termasuk hal berikut. Untuk informasi selengkapnya, lihat Fungsi Tanggal dan Waktu serta Fungsi Konversi Datetime di Amazon Managed Service for Apache Flink SQL Reference.
  - `CURRENT_ROW_TIMESTAMP` – Mengembalikan stempel waktu baru untuk setiap baris di aliran.
  - `TSDIFF` – Mengembalikan perbedaan dua stempel waktu dalam milidetik.
  - `CHAR_TO_DATE` – Mengonversi string ke tanggal.
  - `CHAR_TO_TIME` – Mengonversi string ke waktu.

- `CHAR_TO_TIMESTAMP` – Mengonversi string ke stempel waktu.
- `DATE_TO_CHAR` – Mengonversi tanggal ke string.
- `TIME_TO_CHAR` – Mengonversi waktu ke string.
- `TIMESTAMP_TO_CHAR` – Mengonversi stempel waktu ke string.

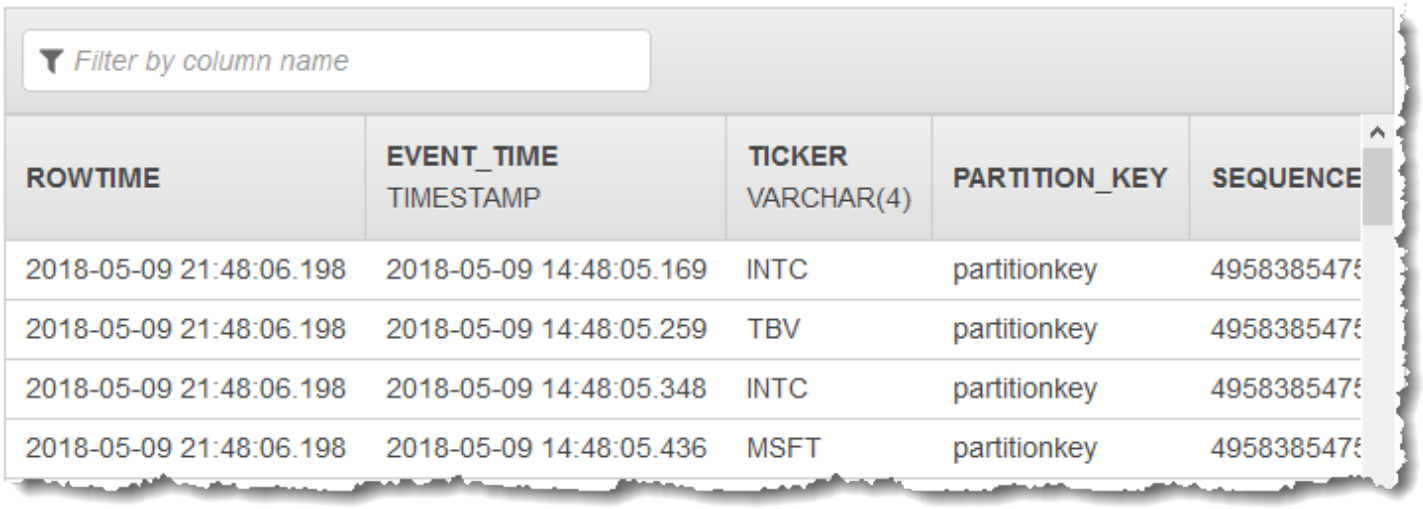
Sebagian besar fungsi SQL sebelumnya menggunakan format untuk mengonversi kolom. Formatnya fleksibel. Misalnya, Anda dapat menentukan format `yyyy-MM-dd hh:mm:ss` untuk mengonversi string input `2009-09-16 03:15:24` menjadi stempel waktu. Untuk informasi selengkapnya, Char To Timestamp (Sys) di Amazon Managed Service for Apache Flink SQL Reference.

Contoh: Mengubah Tanggal

Dalam contoh ini, Anda menulis catatan berikut ke Amazon Kinesis data stream.

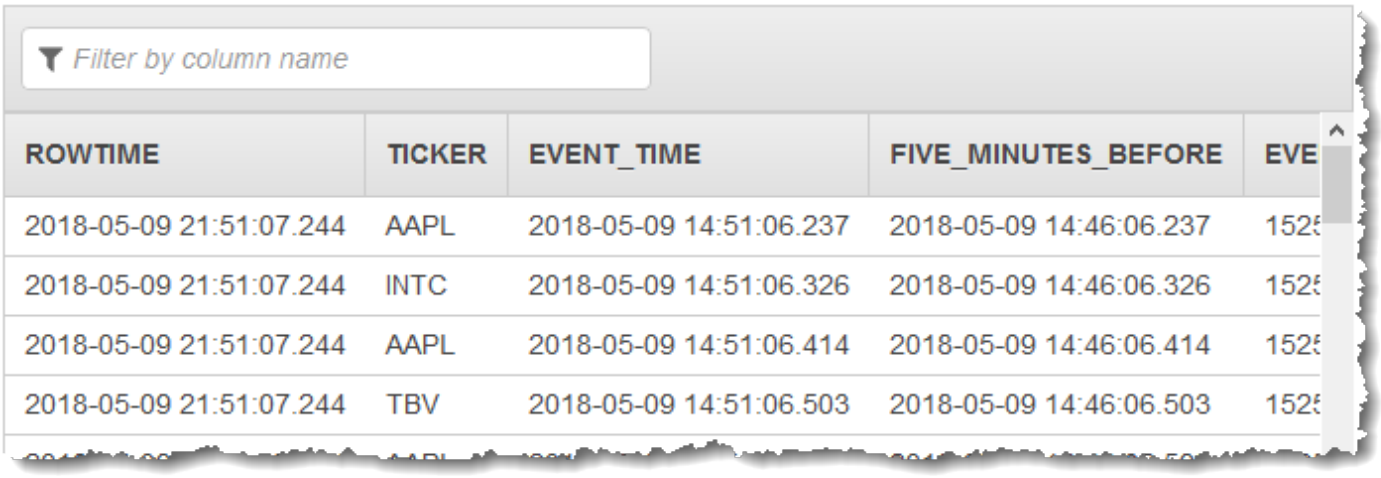
```
{"EVENT_TIME": "2018-05-09T12:50:41.337510", "TICKER": "AAPL"}
{"EVENT_TIME": "2018-05-09T12:50:41.427227", "TICKER": "MSFT"}
{"EVENT_TIME": "2018-05-09T12:50:41.520549", "TICKER": "INTC"}
{"EVENT_TIME": "2018-05-09T12:50:41.610145", "TICKER": "MSFT"}
{"EVENT_TIME": "2018-05-09T12:50:41.704395", "TICKER": "AAPL"}
...
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di konsol, dengan aliran Kinesis sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan dua kolom (`EVENT_TIME` dan `TICKER`) seperti yang ditunjukkan.


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	495838547
2018-05-09 21:48:06.198	2018-05-09 14:48:05.259	TBV	partitionkey	495838547
2018-05-09 21:48:06.198	2018-05-09 14:48:05.348	INTC	partitionkey	495838547
2018-05-09 21:48:06.198	2018-05-09 14:48:05.436	MSFT	partitionkey	495838547

Kemudian, Anda menggunakan kode aplikasi dengan fungsi SQL untuk mengonversi bidang stempel waktu `EVENT_TIME` dalam berbagai cara. Anda selanjutnya memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan dalam tangkapan layar bawah ini:



Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi dengan waktu peristiwa dan catatan ticker sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan.
4. Jalankan kode Python berikut untuk mengisi aliran dengan data sampel. Kode sederhana ini terus menulis catatan dengan simbol ticker acak dan stempel waktu saat ini ke aliran.

```
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"))
```

Langkah 2: Buat Aplikasi Amazon Kinesis Data Analytics

Buat aplikasi sebagai berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming) untuk menyambungkan ke sumber.
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:
    a. Pilih aliran yang Anda buat di bagian sebelumnya.
    b. Pilih untuk membuat IAM role.
    c. Pilih Discover schema (Temukan skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan memiliki dua kolom.
    d. Pilih Edit Schema (Edit Skema). Ubah Column type (Tipe kolom) dari kolom EVENT_TIME ke `TIMESTAMP`.

e. Pilih Save schema and update stream samples (Simpan skema dan perbarui sampel aliran). Setelah konsol menyimpan skema, pilih Exit (Keluar).

f. Jangan pilih Save and continue (Simpan dan lanjutkan).

5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi dan verifikasi hasilnya sebagai berikut:

   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL). Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

Contoh: Mengubah Beberapa Tipe Data

Persyaratan umum dalam aplikasi extract, transform, and load (ETL) adalah memproses beberapa tipe catatan pada sumber streaming. Anda dapat membuat aplikasi Kinesis Data Analytics untuk memproses jenis sumber streaming ini. Prosesnya adalah sebagai berikut:

1. Pertama, Anda memetakan sumber streaming ke aliran input dalam aplikasi, mirip dengan semua aplikasi Kinesis Data Analytics lainnya.
2. Kemudian, dalam kode aplikasi Anda, Anda menulis pernyataan SQL untuk mengambil baris tipe tertentu dari aliran input dalam aplikasi. Anda kemudian memasukkannya ke aliran dalam aplikasi terpisah. (Anda dapat membuat aliran dalam aplikasi tambahan di kode aplikasi Anda.)

Dalam latihan ini, Anda memiliki sumber streaming yang menerima catatan dari dua tipe (`Order` dan `Trade`). Ini adalah pesanan stok dan perdagangan yang sesuai. Untuk setiap order, bisa ada nol atau beberapa perdagangan. Contoh catatan dari setiap tipe ditunjukkan sebagai berikut:

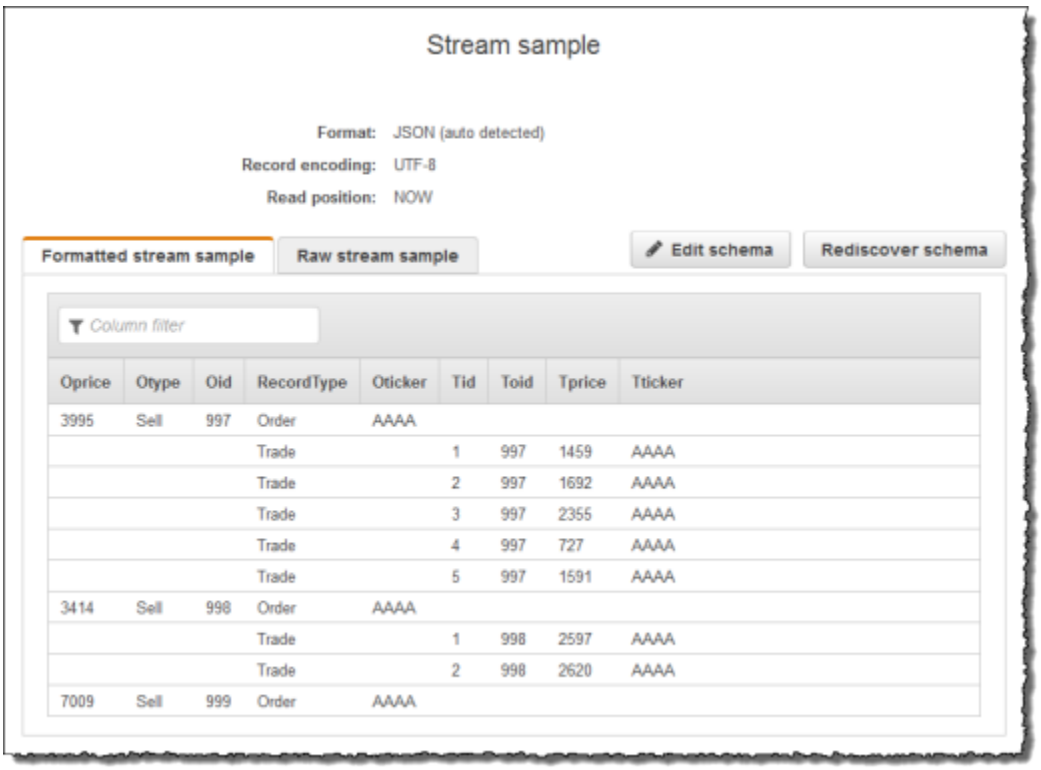
Catatan pesanan

```
{"RecordType": "Order", "Oprice": 9047, "Otype": "Sell", "Oid": 3811, "Oticker":
 "AAAA"}
```

Catatan perdagangan

```
{"RecordType": "Trade", "Tid": 1, "Toid": 3812, "Tprice": 2089, "Tticker": "BBBB"}
```

Saat Anda membuat aplikasi menggunakan AWS Management Console, konsol akan menampilkan skema yang disimpulkan berikut untuk aliran input dalam aplikasi yang dibuat. Secara default, konsol menamai aliran dalam aplikasi ini `SOURCE_SQL_STREAM_001`.



Saat Anda menyimpan konfigurasi, Amazon Kinesis Data Analytics terus membaca data dari sumber streaming dan memasukkan baris di aliran dalam aplikasi. Anda sekarang dapat melakukan analitik pada data di aliran dalam aplikasi.

Di kode aplikasi dalam contoh ini, Anda pertama-tama membuat dua aliran dalam aplikasi tambahan, `Order_Stream` dan `Trade_Stream`. Anda kemudian memfilter baris dari aliran `SOURCE_SQL_STREAM_001` berdasarkan tipe catatan dan memasukkannya dalam aliran yang baru dibuat menggunakan pompa. Untuk informasi tentang pola pengkodean ini, lihat Kode Aplikasi.

1. Filter baris pesanan dan perdagangan menjadi aliran dalam aplikasi terpisah:

   a. Filter catatan urutan di `SOURCE_SQL_STREAM_001`, dan simpan pesanan di `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. Filter catatan perdagangan di `SOURCE_SQL_STREAM_001`, dan simpan pesanan di `Trade_Stream`.

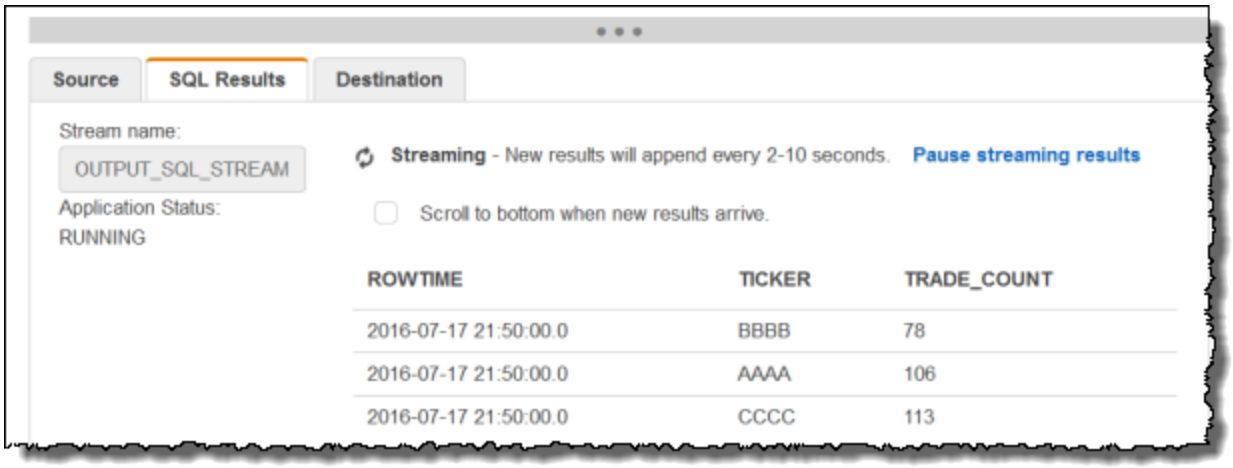
```
--Create Trade_Stream.
CREATE OR REPLACE STREAM "Trade_Stream"
           (trade_id     integer,
            order_id     integer,
            trade_price  DOUBLE,
            ticker       varchar(4),
            record_type  varchar(10)
            );

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. Sekarang Anda dapat melakukan analitik tambahan pada aliran ini. Dalam contoh ini, Anda menghitung jumlah perdagangan dengan ticker dalam satu menit jendela tumbling dan menyimpan hasilnya ke aliran lain, `DESTINATION_SQL_STREAM`.

```
--do some analytics on the Trade_Stream and Order_Stream.
-- To see results in console you must write to OPUT_SQL_STREAM.

CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
            ticker  varchar(4),
            trade_count   integer
            );

CREATE OR REPLACE PUMP "Output_Pump" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM ticker, count(*) as trade_count
      FROM   "Trade_Stream"
      GROUP BY ticker,
```

```
                FLOOR("Trade_Stream".ROWTIME TO MINUTE);
```

Anda melihat hasilnya, seperti yang ditunjukkan berikut:



Topik

- Langkah 1: Siapkan Data
- Langkah 2: Buat Aplikasi

Langkah Selanjutnya

Langkah 1: Siapkan Data

Langkah 1: Siapkan Data

Di bagian ini, Anda membuat Kinesis data stream, lalu mengisi catatan pesanan dan perdagangan di aliran. Ini adalah sumber streaming Anda untuk aplikasi yang Anda buat di langkah berikutnya.

Topik

- Langkah 1.1: Buat Sumber Streaming
- Langkah 1.2: Isi Sumber Streaming

Langkah 1.1: Buat Sumber Streaming

Anda dapat membuat Kinesis data stream menggunakan konsol atau AWS CLI. Contoh tersebut mengasumsikan `OrdersAndTradesStream` sebagai nama aliran.

- Menggunakan konsol — Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis. Pilih Data Streams (Aliran Data), lalu buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
- Menggunakan AWS CLI - Gunakan `create-stream` AWS CLI perintah Kinesis berikut untuk membuat aliran:

```
$ aws kinesis create-stream \
--stream-name OrdersAndTradesStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

Langkah 1.2: Isi Sumber Streaming

Jalankan skrip Python berikut untuk mengisi catatan sampel di `OrdersAndTradesStream`. Jika Anda membuat aliran dengan nama yang berbeda, perbarui kode Python dengan tepat.

1. Instal Python dan `pip`.

   Untuk informasi tentang menginstal Python, lihat situs web Python.

   Anda dapat menginstal dependensi menggunakan pip. Untuk informasi tentang menginstal pip, lihat Penginstalan di situs web pip.
2. Jalankan kode Phyton berikut. Perintah `put-record` dalam kode menulis catatan JSON ke aliran.

```
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"))
```

Langkah Selanjutnya

Langkah 2: Buat Aplikasi

Langkah 2: Buat Aplikasi

Di bagian ini, Anda membuat aplikasi Kinesis Data Analytics. Anda kemudian memperbarui aplikasi dengan menambahkan konfigurasi input yang memetakan sumber streaming yang Anda buat di bagian sebelumnya ke aliran input dalam aplikasi.

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi). Contoh ini menggunakan nama aplikasi **`ProcessMultipleRecordTypes`**.
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming) untuk menyambungkan ke sumber.
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:
   a. Pilih aliran yang Anda buat di Langkah 1: Siapkan Data.
   b. Pilih untuk membuat IAM role.
   c. Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat.
   d. Jangan pilih Save and continue (Simpan dan lanjutkan).
5. Di hub aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi dan verifikasi hasilnya:
   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
--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. Pilih Save and run SQL (Simpan dan jalankan SQL). Pilih tab Real-time analytics (Analitik waktu nyata) untuk melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

Langkah Selanjutnya

Anda dapat mengonfigurasi keluaran aplikasi untuk mempertahankan hasil ke tujuan eksternal, seperti aliran Kinesis lain atau aliran pengiriman data Firehose.

Contoh: Jendela dan Agregasi

Bagian ini memberikan contoh aplikasi Amazon Kinesis Data Analytics yang menggunakan kueri jendela dan agregat. (Untuk informasi selengkapnya, lihat Kueri Jendela.) Setiap contoh memberikan step-by-step instruksi dan kode contoh untuk menyiapkan aplikasi Kinesis Data Analytics.

Topik

- Contoh: Jendela Stagger
- Contoh: Jendela Tumbling Menggunakan ROWTIME
- Contoh: Jendela Tumbling Menggunakan Stempel Waktu Peristiwa
- Contoh: Mengambil Nilai yang Paling Sering Terjadi (TOP_K_ITEMS_TUMBLING)
- Contoh: Menggabungkan Hasil Parsial dari Kueri

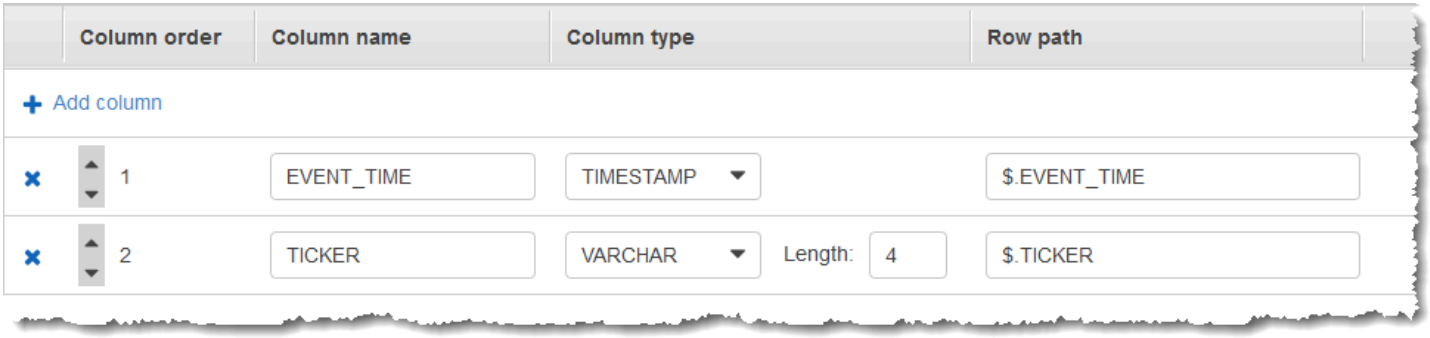
Contoh: Jendela Stagger

Ketika kueri jendela memproses jendela terpisah untuk setiap kunci partisi unik, yang dimulai ketika data dengan kunci yang cocok tiba, jendela dirujuk sebagai jendela stagger. Untuk detailnya, lihat Jendela Stagger. Contoh Amazon Kinesis Data Analytics menggunakan kolom EVENT_TIME dan TICKER untuk membuat jendela stagger. Aliran sumber berisi grup enam catatan dengan nilai EVENT_TIME dan TICKER identik yang tiba dalam periode satu menit, tetapi tidak harus dengan nilai menit yang sama (misalnya, `18:41:xx`).

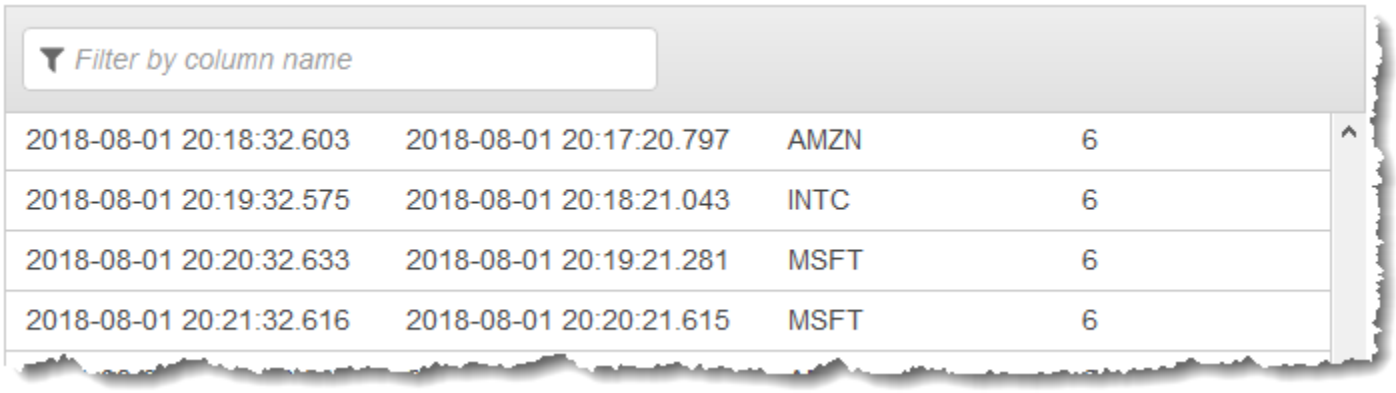
Dalam contoh ini, Anda menulis catatan berikut ke Kinesis data stream pada waktu berikut. Skrip tidak menulis waktu ke aliran, tetapi waktu catatan diserap oleh aplikasi ditulis ke bidang `ROWTIME`:

```
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:17:30
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:17:40
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:17:50
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:18:00
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:18:10
{"EVENT_TIME": "2018-08-01T20:17:20.797945", "TICKER": "AMZN"}   20:18:21
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:18:31
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:18:41
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:18:51
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:19:01
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:19:11
{"EVENT_TIME": "2018-08-01T20:18:21.043084", "TICKER": "INTC"}   20:19:21
...
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di AWS Management Console, dengan Kinesis data stream sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan dua kolom (`EVENT_TIME` dan `TICKER`) seperti yang ditunjukkan berikut ini.



Anda menggunakan kode aplikasi dengan fungsi `COUNT` untuk membuat jendela agregasi data. Anda selanjutnya memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan dalam tangkapan layar bawah ini:



Dalam prosedur berikut, Anda membuat aplikasi Kinesis Data Analytics yang menggabungkan nilai dalam aliran input di jendela stagger berdasarkan EVENT_TIME dan TICKER.

Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), lalu buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Untuk menulis catatan ke Kinesis data stream di lingkungan produksi, sebaiknya gunakan Kinesis Producer Library atau API Kinesis Data Streams. Untuk kemudahan, contoh ini menggunakan skrip Python berikut untuk menghasilkan catatan. Jalankan kode untuk mengisi catatan ticker sampel. Kode sederhana ini terus menulis grup enam catatan dengan `EVENT_TIME` acak yang sama dan simbol ticker ke aliran, selama satu menit. Jaga skrip agar tetap berjalan agar Anda dapat menghasilkan skema aplikasi di langkah berikutnya.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Buat aplikasi Amazon Kinesis Data Analytics seperti berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming) untuk menyambungkan ke sumber.
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:

   a. Pilih aliran yang Anda buat di bagian sebelumnya.
   b. Pilih Discover Schema (Temukan Skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan memiliki dua kolom.
   c. Pilih Edit Schema (Edit Skema). Ubah Column type (Tipe kolom) dari kolom EVENT_TIME ke `TIMESTAMP`.
   d. Pilih Save schema and update stream samples (Simpan skema dan perbarui sampel aliran). Setelah konsol menyimpan skema, pilih Exit (Keluar).
   e. Jangan pilih Save and continue (Simpan dan lanjutkan).
5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:

   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL).

   Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

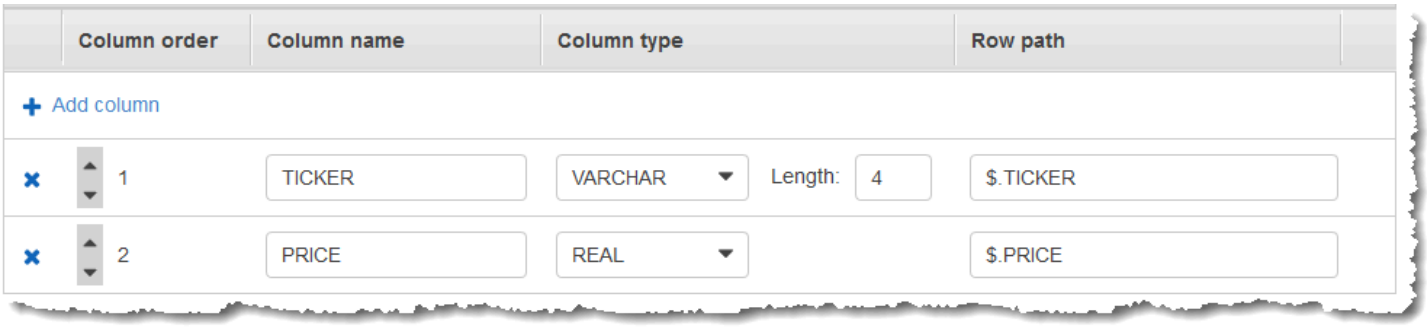
Contoh: Jendela Tumbling Menggunakan ROWTIME

Ketika kueri jendela memproses setiap jendela dengan cara yang tidak tumpang tindih, jendela disebut sebagai jendela tumbling. Untuk detailnya, lihat Jendela Tumbling (Agregasi Menggunakan GROUP BY). Contoh Amazon Kinesis Data Analytics ini menggunakan kolom `ROWTIME` untuk membuat jendela tumbling. Kolom `ROWTIME` mewakili waktu catatan dibaca oleh aplikasi.

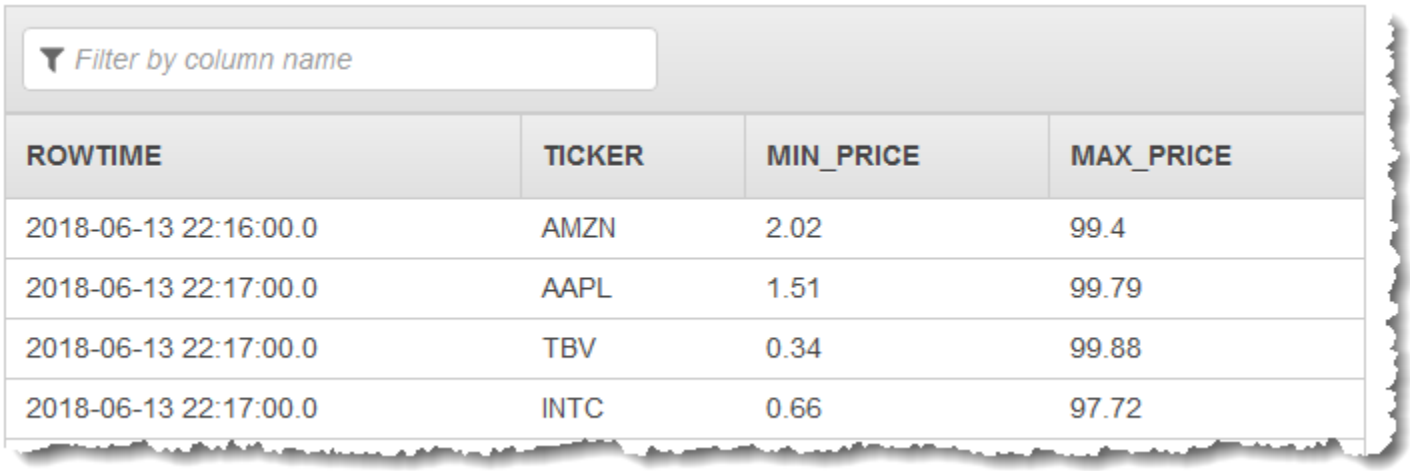
Dalam contoh ini, Anda menulis catatan berikut ke Kinesis data stream.

```
{"TICKER": "TBV", "PRICE": 33.11}
{"TICKER": "INTC", "PRICE": 62.04}
{"TICKER": "MSFT", "PRICE": 40.97}
{"TICKER": "AMZN", "PRICE": 27.9}
...
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di AWS Management Console, dengan Kinesis data stream sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan dua kolom (`TICKER` dan `PRICE`) seperti yang ditunjukkan berikut ini.



Anda menggunakan kode aplikasi dengan fungsi `MIN` dan `MAX` untuk membuat jendela agregasi data. Anda selanjutnya memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan dalam tangkapan layar bawah ini:



Dalam prosedur berikut, Anda membuat aplikasi Kinesis Data Analytics yang menggabungkan nilai dalam aliran input di jendela tumbling berdasarkan ROWTIME.

Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.

3. Pilih Create Kinesis stream (Buat Aliran Kinesis), lalu buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Untuk menulis catatan ke Kinesis data stream di lingkungan produksi, sebaiknya gunakan Kinesis Client Library atau API Kinesis Data Streams. Untuk kemudahan, contoh ini menggunakan skrip Python berikut untuk menghasilkan catatan. Jalankan kode untuk mengisi catatan ticker sampel. Kode sederhana ini terus menulis catatan ticker acak ke aliran. Jaga skrip agar tetap berjalan agar Anda dapat menghasilkan skema aplikasi di langkah berikutnya.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Buat aplikasi Amazon Kinesis Data Analytics seperti berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming) untuk menyambungkan ke sumber.
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:

   a. Pilih aliran yang Anda buat di bagian sebelumnya.

   b. Pilih Discover Schema (Temukan Skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan memiliki dua kolom.

   c. Pilih Save schema and update stream samples (Simpan skema dan perbarui sampel aliran). Setelah konsol menyimpan skema, pilih Exit (Keluar).

   d. Jangan pilih Save and continue (Simpan dan lanjutkan).
5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.
6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:

   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL).

   Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

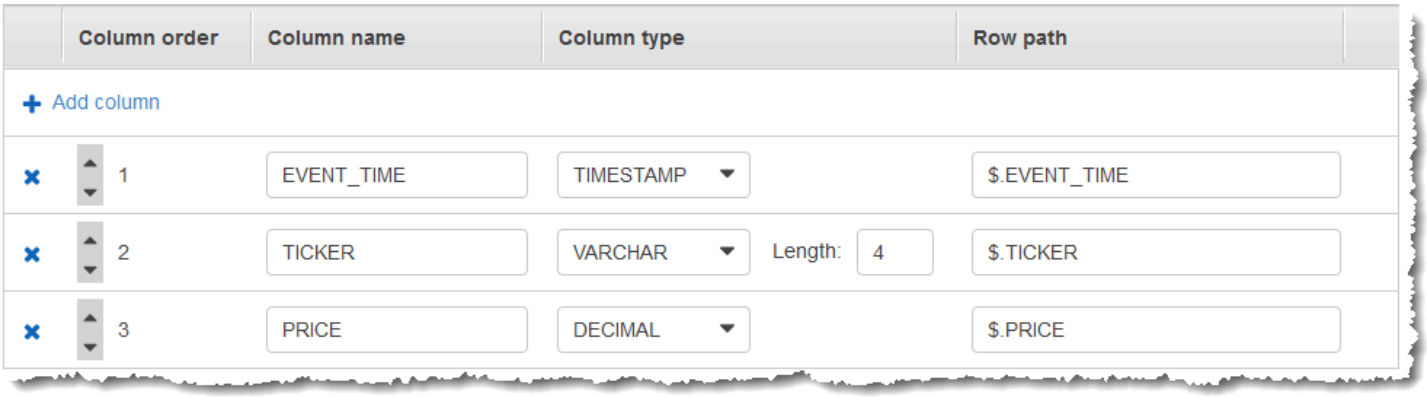
Contoh: Jendela Tumbling Menggunakan Stempel Waktu Peristiwa

Ketika kueri jendela memproses setiap jendela dengan cara yang tidak tumpang tindih, jendela disebut sebagai jendela tumbling. Untuk detailnya, lihat Jendela Tumbling (Agregasi Menggunakan GROUP BY). Contoh Amazon Kinesis Data Analytics menunjukkan jendela tumbling yang menggunakan stempel waktu peristiwa, yang merupakan stempel waktu yang dibuat pengguna yang disertakan dalam data streaming. Contoh tersebut menggunakan pendekatan ini daripada hanya menggunakan ROWTIME, yang merupakan stempel waktu yang dibuat Kinesis Data Analytics saat aplikasi menerima catatan. Anda akan menggunakan stempel waktu peristiwa dalam data streaming jika Anda ingin membuat agregasi berdasarkan kapan peristiwa terjadi, bukan ketika peristiwa diterima oleh aplikasi. Dalam contoh ini, nilai `ROWTIME` memicu agregasi setiap menit, dan catatan dikumpulkan oleh kedua `ROWTIME` dan waktu peristiwa yang disertakan.

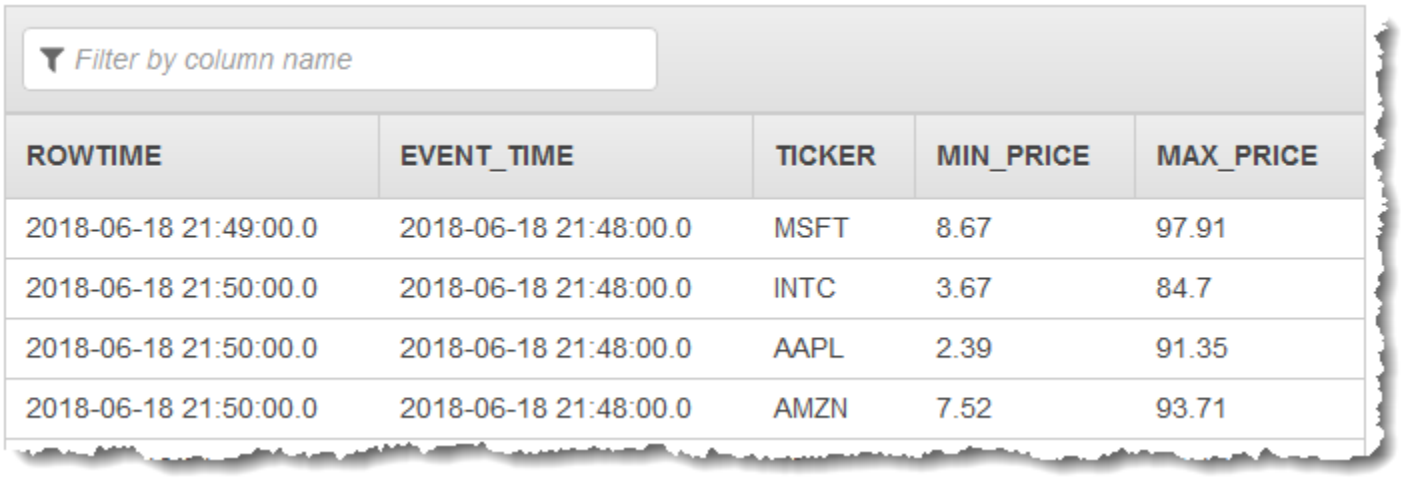
Dalam contoh ini, Anda menulis catatan berikut ke Amazon Kinesis stream. Nilai `EVENT_TIME` sebelumnya diatur ke 5 detik, untuk menyimulasikan keterlambatan pemrosesan dan transmisi yang mungkin membuat penundaan sejak peristiwa terjadi, hingga saat catatan tersebut diserap ke Kinesis Data Analytics.

```
{"EVENT_TIME": "2018-06-13T14:11:05.766191", "TICKER": "TBV", "PRICE": 43.65}
{"EVENT_TIME": "2018-06-13T14:11:05.848967", "TICKER": "AMZN", "PRICE": 35.61}
{"EVENT_TIME": "2018-06-13T14:11:05.931871", "TICKER": "MSFT", "PRICE": 73.48}
{"EVENT_TIME": "2018-06-13T14:11:06.014845", "TICKER": "AMZN", "PRICE": 18.64}
...
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di AWS Management Console, dengan Kinesis data stream sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan tiga kolom (`EVENT_TIME`, `TICKER`, dan `PRICE`) seperti yang ditunjukkan berikut ini.



Anda menggunakan kode aplikasi dengan fungsi `MIN` dan `MAX` untuk membuat jendela agregasi data. Anda selanjutnya memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan dalam tangkapan layar bawah ini:



Dalam prosedur berikut, Anda membuat aplikasi Kinesis Data Analytics yang menggabungkan nilai dalam aliran input di jendela tumbling berdasarkan waktu peristiwa.

Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), lalu buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Untuk menulis catatan ke Kinesis data stream di lingkungan produksi, sebaiknya gunakan Kinesis Client Library atau API Kinesis Data Streams. Untuk kemudahan, contoh ini menggunakan skrip Python berikut untuk menghasilkan catatan. Jalankan kode untuk mengisi catatan ticker sampel. Kode sederhana ini terus menulis catatan ticker acak ke aliran. Jaga skrip agar tetap berjalan agar Anda dapat menghasilkan skema aplikasi di langkah berikutnya.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Buat aplikasi Amazon Kinesis Data Analytics seperti berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming) untuk menyambungkan ke sumber.

4. Di halaman Sambungkan ke sumber, lakukan hal berikut:

   a. Pilih aliran yang Anda buat di bagian sebelumnya.

   b. Pilih Discover Schema (Temukan Skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan memiliki tiga kolom.

   c. Pilih Edit Schema (Edit Skema). Ubah Column type (Tipe kolom) dari kolom EVENT_TIME ke `TIMESTAMP`.

   d. Pilih Save schema and update stream samples (Simpan skema dan perbarui sampel aliran). Setelah konsol menyimpan skema, pilih Exit (Keluar).

   e. Jangan pilih Save and continue (Simpan dan lanjutkan).

5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.

6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:

   a. Salin kode aplikasi berikut dan tempelkan ke editor.

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL).

      Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

Contoh: Mengambil Nilai yang Paling Sering Terjadi (TOP_K_ITEMS_TUMBLING)

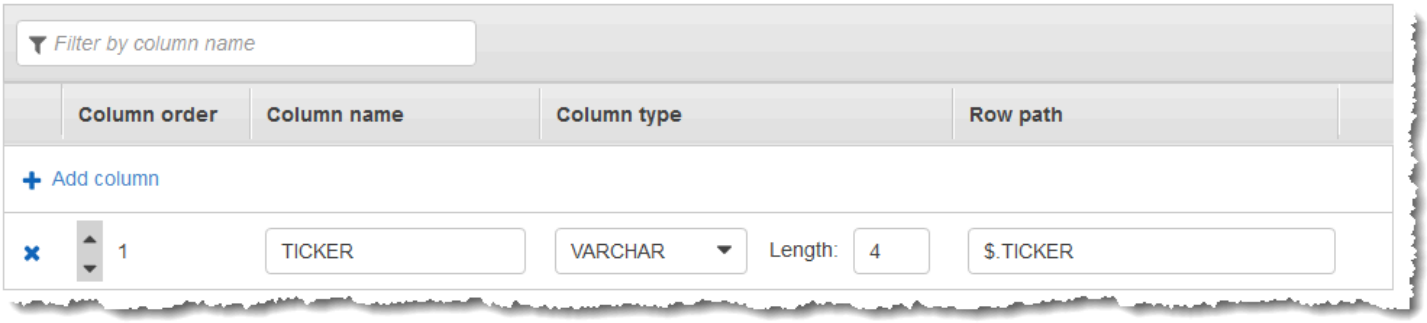
Contoh Amazon Kinesis Data Analytics menunjukkan cara menggunakan fungsi `TOP_K_ITEMS_TUMBLING` untuk mengambil nilai yang paling sering terjadi di jendela tumbling. Untuk informasi selengkapnya, lihat `TOP_K_ITEMS_TUMBLING`fungsi di Amazon Managed Service for Apache Flink SQL Reference.

Fungsi `TOP_K_ITEMS_TUMBLING` berguna ketika menggabungkan lebih dari puluhan atau ratusan ribu kunci, dan Anda ingin mengurangi penggunaan sumber daya Anda. Fungsi ini menghasilkan hasil yang sama seperti menggabungkan dengan klausa `GROUP BY` dan `ORDER BY`.

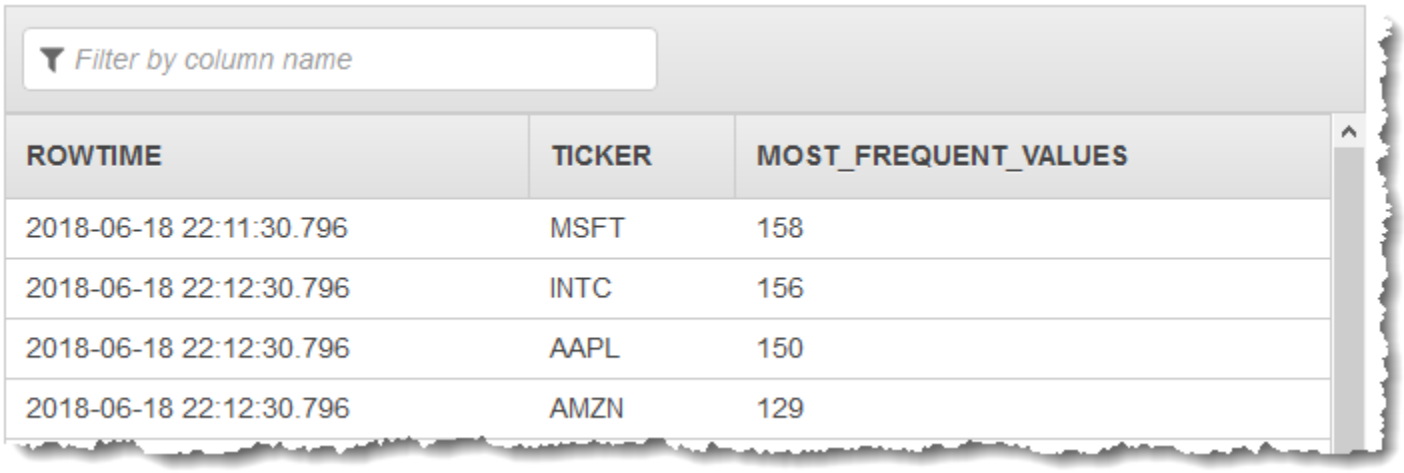
Dalam contoh ini, Anda menulis catatan berikut ke Amazon Kinesis data stream:

```
{"TICKER": "TBV"}
{"TICKER": "INTC"}
{"TICKER": "MSFT"}
{"TICKER": "AMZN"}
...
```

Anda kemudian membuat aplikasi Kinesis Data Analytics di AWS Management Console, dengan Kinesis data stream sebagai sumber streaming. Proses penemuan membaca catatan sampel pada sumber streaming dan menyimpulkan skema dalam aplikasi dengan satu kolom (`TICKER`) seperti yang ditunjukkan di bawah ini.



Anda menggunakan kode aplikasi dengan fungsi `TOP_K_VALUES_TUMBLING` untuk membuat jendela agregasi data. Anda selanjutnya memasukkan data yang dihasilkan ke aliran dalam aplikasi lainnya, seperti yang ditunjukkan dalam tangkapan layar bawah ini:



Dalam prosedur berikut, Anda membuat aplikasi Kinesis Data Analytics yang mengambil nilai yang paling sering terjadi di aliran input.

Topik

- Langkah 1: Buat Kinesis Data Stream
- Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 1: Buat Kinesis Data Stream

Buat Amazon Kinesis data stream dan isi catatan sebagai berikut:

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Streams (Aliran Data) di panel navigasi.
3. Pilih Create Kinesis stream (Buat Aliran Kinesis), lalu buat aliran dengan satu serpihan. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
4. Untuk menulis catatan ke Kinesis data stream di lingkungan produksi, sebaiknya gunakan Kinesis Client Library atau API Kinesis Data Streams. Untuk kemudahan, contoh ini menggunakan skrip Python berikut untuk menghasilkan catatan. Jalankan kode untuk mengisi catatan ticker sampel. Kode sederhana ini terus menulis catatan ticker acak ke aliran. Biarkan skrip tetap berjalan agar Anda dapat menghasilkan skema aplikasi di langkah berikutnya.

```
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"))
```

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Buat aplikasi Amazon Kinesis Data Analytics seperti berikut:

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/kinesisanalytics.
2. Pilih Create application (Buat aplikasi), masukkan nama aplikasi, dan pilih Create application (Buat aplikasi).
3. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming) untuk menyambungkan ke sumber.
4. Di halaman Sambungkan ke sumber, lakukan hal berikut:

    a. Pilih aliran yang Anda buat di bagian sebelumnya.

b. Pilih Discover Schema (Temukan Skema). Tunggu hingga konsol menampilkan skema yang disimpulkan dan catatan sampel yang digunakan untuk menyimpulkan skema untuk aliran dalam aplikasi yang dibuat. Skema yang disimpulkan memiliki satu kolom.

c. Pilih Save schema and update stream samples (Simpan skema dan perbarui sampel aliran). Setelah konsol menyimpan skema, pilih Exit (Keluar).

d. Jangan pilih Save and continue (Simpan dan lanjutkan).

5. Di halaman detail aplikasi, pilih Go to SQL editor (Buka editor SQL). Untuk memulai aplikasi, pilih Yes, start application (Ya, mulai aplikasi) di kotak dialog yang muncul.

6. Di editor SQL, tulis kode aplikasi, dan verifikasi hasilnya sebagai berikut:

a. Salin kode aplikasi berikut dan tempelkan ke editor:

```
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. Pilih Save and run SQL (Simpan dan jalankan SQL).

Di tab Real-time analytics (Analitik waktu nyata), Anda dapat melihat semua aliran dalam aplikasi yang dibuat aplikasi dan memverifikasi data.

Contoh: Menggabungkan Hasil Parsial dari Kueri

Jika Amazon Kinesis data stream berisi catatan yang memiliki waktu peristiwa yang tidak sama persis dengan waktu penyerapan, pilihan hasil di jendela tumbling berisi catatan yang tiba, tetapi tidak selalu

terjadi, di dalam jendela. Dalam kasus ini, jendela tumbling hanya berisi sebagian hasil yang Anda inginkan. Ada beberapa pendekatan yang dapat Anda gunakan untuk memperbaiki masalah ini:

- Cukup gunakan jendela tumbling, dan gabungkan hasil parsial dalam pascapemrosesan melalui basis data atau gudang data menggunakan upserts. Pendekatan ini efisien dalam memproses aplikasi. Ini menangani data yang terlambat tanpa batas untuk operator agregat (`sum`, `min`, `max`, dan sebagainya). Kelemahan dari pendekatan ini adalah Anda harus mengembangkan dan memelihara logika aplikasi tambahan dalam lapisan basis data.
- Gunakan jendela tumbling dan geser, yang menghasilkan hasil parsial awal, tetapi juga terus menghasilkan hasil yang lengkap selama periode jendela geser. Pendekatan ini menangani data yang terlambat dengan menimpa bukan upsert sehingga tidak ada logika aplikasi tambahan yang perlu ditambahkan dalam lapisan basis data. Kelemahan dari pendekatan ini adalah menggunakan lebih banyak unit pemrosesan Kinesis (KPU) dan masih menghasilkan dua hasil, yang mungkin tidak berfungsi untuk beberapa kasus penggunaan.

Untuk informasi selengkapnya tentang jendela tumbling dan geser, lihat Kueri Jendela.

Dalam prosedur berikut, agregasi jendela tumbling menghasilkan dua hasil parsial (dikirim ke aliran dalam aplikasi `CALC_COUNT_SQL_STREAM`) yang harus dikombinasikan untuk menghasilkan hasil akhir. Aplikasi kemudian menghasilkan agregasi kedua (dikirim ke aliran dalam aplikasi `DESTINATION_SQL_STREAM`) yang menggabungkan dua hasil parsial.

Untuk membuat aplikasi yang mengumpulkan hasil parsial menggunakan waktu peristiwa

1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Analytics (Analitik Data) di panel navigasi. Buat aplikasi Kinesis Data Analytics seperti yang dijelaskan dalam tutorial Memulai dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL.
3. Di editor SQL, ganti kode aplikasi dengan berikut ini:

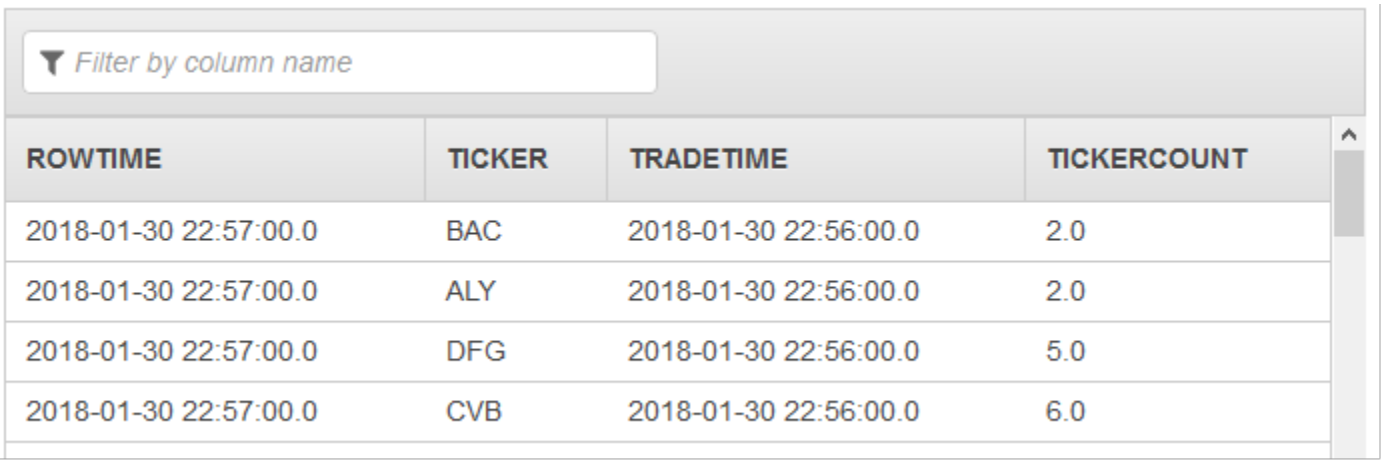
```
CREATE OR REPLACE STREAM "CALC_COUNT_SQL_STREAM"
    (TICKER      VARCHAR(4),
    TRADETIME   TIMESTAMP,
    TICKERCOUNT       DOUBLE);

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);
```

Pernyataan `SELECT` dalam kode aplikasi memfilter baris dalam `SOURCE_SQL_STREAM_001` untuk perubahan harga stok yang lebih besar dari 1 persen dan memasukkan baris ke aliran dalam aplikasi lainnya `CHANGE_STREAM` menggunakan pompa.

4. Pilih Save and run SQL (Simpan dan jalankan SQL).

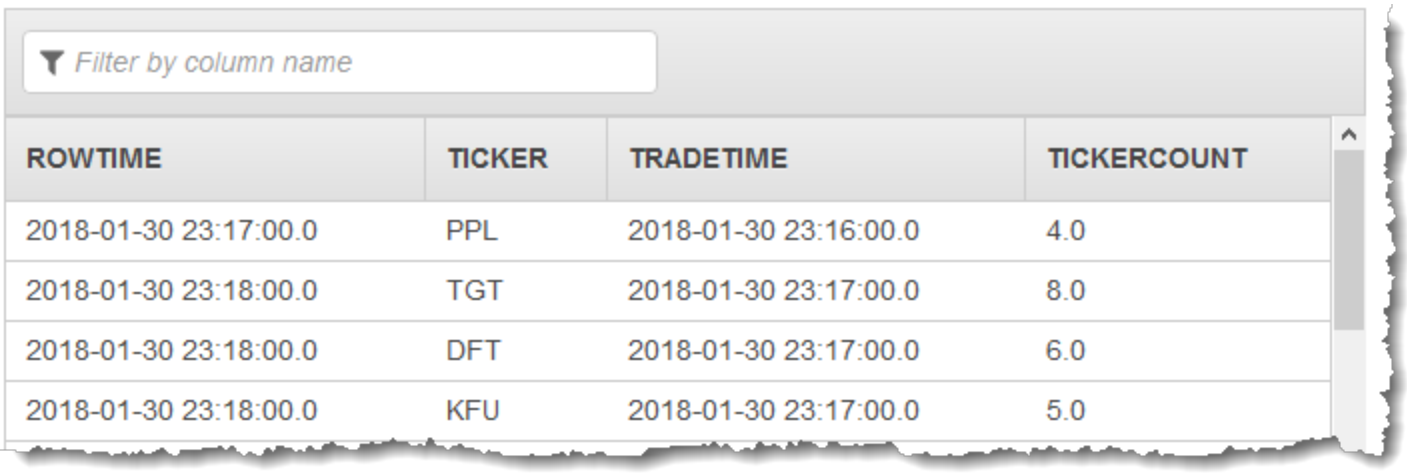
Pompa pertama mengeluarkan aliran ke `CALC_COUNT_SQL_STREAM` yang mirip dengan berikut ini. Perhatikan bahwa hasil set tidak lengkap:



Filter by column name

ROWTIME	TICKER	TRADETIME	TICKERCOUNT
2018-01-30 22:57:00.0	BAC	2018-01-30 22:56:00.0	2.0
2018-01-30 22:57:00.0	ALY	2018-01-30 22:56:00.0	2.0
2018-01-30 22:57:00.0	DFG	2018-01-30 22:56:00.0	5.0
2018-01-30 22:57:00.0	CVB	2018-01-30 22:56:00.0	6.0

Pompa kedua kemudian menghasilkan aliran ke `DESTINATION_SQL_STREAM` yang berisi sekumpulan hasil yang lengkap:



Filter by column name

ROWTIME	TICKER	TRADETIME	TICKERCOUNT
2018-01-30 23:17:00.0	PPL	2018-01-30 23:16:00.0	4.0
2018-01-30 23:18:00.0	TGT	2018-01-30 23:17:00.0	8.0
2018-01-30 23:18:00.0	DFT	2018-01-30 23:17:00.0	6.0
2018-01-30 23:18:00.0	KFU	2018-01-30 23:17:00.0	5.0

Contoh: Bergabung

Bagian ini memberikan contoh aplikasi Kinesis Data Analytics yang menggunakan kueri gabungan. Setiap contoh memberikan step-by-step instruksi untuk menyiapkan dan menguji aplikasi Kinesis Data Analytics Anda.

Topik

- Contoh: Menambahkan Data Referensi ke Aplikasi Kinesis Data Analytics

Contoh: Menambahkan Data Referensi ke Aplikasi Kinesis Data Analytics

Dalam latihan ini, Anda menambahkan data referensi ke aplikasi Kinesis Data Analytics yang ada. Untuk informasi tentang data referensi, lihat topik berikut:

- Amazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya
- Mengonfigurasi Input Aplikasi

Dalam latihan ini, Anda menambahkan data referensi ke aplikasi yang Anda buat di latihan Memulai Kinesis Data Analytics. Data referensi memberi perusahaan nama untuk setiap simbol ticker; sebagai contoh:

```
Ticker, Company
AMZN,Amazon
ASD, SomeCompanyA
MMB, SomeCompanyB
WAS,  SomeCompanyC
```

Pertama, selesaikan langkah-langkah di latihan Memulai untuk membuat aplikasi starter. Kemudian ikuti langkah ini untuk menyiapkan dan menambahkan data referensi ke aplikasi Anda:

1. Siapkan datanya
   - Simpan data referensi sebelumnya sebagai objek di Amazon Simple Storage Service (Amazon S3).
   - Buat IAM role yang dapat diambil oleh Kinesis Data Analytics untuk membaca objek Amazon S3 atas nama Anda.
2. Tambahkan sumber data referensi ke aplikasi Anda.

   Kinesis Data Analytics membaca objek Amazon S3 dan membuat tabel referensi dalam aplikasi yang dapat Anda kueri dalam kode aplikasi Anda.
3. Uji kodenya.

   Dalam kode aplikasi Anda, Anda menulis kueri bergabung untuk menggabungkan aliran dalam aplikasi dengan tabel referensi dalam aplikasi, untuk mendapatkan nama perusahaan untuk setiap simbol ticker.

Topik

- Langkah 1: Siapkan
- Langkah 2: Tambahkan Sumber Data Referensi ke Konfigurasi Aplikasi
- Langkah 3: Uji: Kuerikan Tabel Referensi dalam Aplikasi

Langkah 1: Siapkan

Di bagian ini, Anda menyimpan data referensi sampel sebagai objek di bucket Amazon S3. Anda juga membuat IAM role yang dapat diambil oleh Kinesis Data Analytics untuk membaca objek atas nama Anda.

Simpan Data Referensi sebagai Objek Amazon S3

Di langkah ini, Anda menyimpan data referensi sampel sebagai objek Amazon S3.

1. Buka editor teks, tambahkan data berikut, dan simpan file sebagai `TickerReference.csv`.

   ```
   Ticker, Company
   AMZN,Amazon
   ASD, SomeCompanyA
   MMB, SomeCompanyB
   WAS,  SomeCompanyC
   ```

2. Unggah file `TickerReference.csv` ke bucket S3 Anda. Untuk petunjuk, lihat Mengunggah Objek ke Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Buat IAM Role

Selanjutnya, buat IAM role yang dapat diambil oleh Kinesis Data Analytics dan membaca objek Amazon S3 atas nama Anda.

1. Di AWS Identity and Access Management (IAM), buat IAM role bernama **`KinesisAnalytics-ReadS3Object`**. Untuk membuat peran, ikuti petunjuk di Membuat Peran untuk Layanan Amazon (AWS Management Console) dalam Panduan Pengguna IAM.

   Di konsol IAM, tentukan hal berikut:

   - Untuk Select Role Type (Pilih Tipe Peran), pilih AWS Lambda. Setelah membuat peran, Anda akan mengubah kebijakan kepercayaan untuk mengizinkan Kinesis Data Analytics (bukan AWS Lambda) mengambil peran.

- Jangan lampirkan kebijakan apa pun di halaman Lampirkan Kebijakan.

2. Perbarui kebijakan IAM role:

   a. Di konsol IAM, pilih peran yang Anda buat.

   b. Di tab Trust Relationships (Hubungan Kepercayaan), perbarui kebijakan kepercayaan untuk memberi Kinesis Data Analytics izin mengambil peran. Kebijakan kepercayaan ditampilkan sebagai berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   c. Pada tab Izin, lampirkan kebijakan yang dikelola Amazon yang disebut AmazonS3. ReadOnlyAccess Kebijakan ini peran izin untuk membaca objek Amazon S3. Kebijakan ini ditampilkan sebagai berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

Langkah 2: Tambahkan Sumber Data Referensi ke Konfigurasi Aplikasi

Di langkah ini, Anda akan menambahkan sumber data referensi ke konfigurasi aplikasi Anda. Untuk memulai, Anda memerlukan informasi berikut:

- Nama bucket S3 dan nama kunci objek Anda
- Amazon Resource Name (ARN) IAM role

1. Di halaman utama aplikasi, pilih Connect reference data (Sambungkan data referensi).
2. Di halaman Sambungkan sumber data referensi, pilih bucket Amazon S3 yang berisi objek data referensi Anda, dan masukkan nama kunci objek.
3. Masukkan **`CompanyName`** untuk Nama tabel referensi dalam aplikasi.
4. Di bagian Akses ke sumber daya yang dipilih, pilih Pilih dari peran IAM yang dapat diasumsikan oleh Kinesis Analytics, dan pilih peran IAM -reads3Object yang Anda buat KinesisAnalyticsdi bagian sebelumnya.
5. Pilih Discover schema (Temukan skema). Konsol mendeteksi dua kolom dalam data referensi.
6. Pilih Save and close (Simpan dan pilih).

Langkah 3: Uji: Kuerikan Tabel Referensi dalam Aplikasi

Anda sekarang dapat mengkueri tabel referensi dalam aplikasi, `CompanyName`. Anda dapat menggunakan informasi referensi untuk memperkaya aplikasi Anda dengan menggabungkan data harga ticker dengan tabel referensi. Hasilnya menunjukkan nama perusahaan.

1. Ganti kode aplikasi Anda dengan hal berikut. Kueri menggabungkan aliran input dalam aplikasi dengan tabel referensi dalam aplikasi. Kode aplikasi menulis hasil ke aliran dalam aplikasi lainnya, `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. Verifikasi bahwa output aplikasi muncul di tab SQLResults. Pastikan beberapa baris menunjukkan nama perusahaan (data referensi sampel Anda tidak memiliki semua nama perusahaan).

Contoh: Machine Learning

Bagian ini memberikan contoh aplikasi Amazon Kinesis Data Analytics yang menggunakan kueri machine learning. Kueri machine learning melakukan analisis kompleks pada data, mengandalkan riwayat data dalam aliran untuk menemukan pola yang tidak biasa. Contoh memberikan step-by-step petunjuk untuk menyiapkan dan menguji aplikasi Kinesis Data Analytics Anda.

Topik

- Contoh: Mendeteksi Anomali Data pada Aliran (Fungsi RANDOM_CUT_FORECT)
- Contoh: Mendeteksi Anomali Data dan Mendapatkan Penjelasan (Fungsi RANDOM_CUT_FOREST_WITH_EXPLANATION)
- Contoh: Mendeteksi Hotspot di Aliran (Fungsi HOTSPOTS)

Contoh: Mendeteksi Anomali Data pada Aliran (Fungsi RANDOM_CUT_FORECT)

Amazon Kinesis Data Analytics menyediakan fungsi (`RANDOM_CUT_FOREST`) yang dapat menetapkan skor anomali ke setiap catatan berdasarkan nilai-nilai dalam kolom numerik. Untuk informasi selengkapnya, lihat `RANDOM_CUT_FOREST`Fungsi di Amazon Managed Service for Apache Flink SQL Reference.

Dalam latihan ini, Anda menulis kode aplikasi untuk menetapkan skor anomali ke catatan pada sumber streaming aplikasi Anda. Untuk menyiapkan aplikasi, Anda melakukan hal berikut:

1. Siapkan sumber streaming – Anda menyiapkan aliran data Kinesis dan menulis sampel data `heartRate`, seperti yang ditunjukkan berikut:

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

Prosedur ini memberi Anda skrip Python untuk mengisi aliran. Nilai `heartRate` dibuat secara acak, dengan 99 persen catatan memiliki nilai `heartRate` antara 60 dan 100, dan hanya 1 persen dari nilai `heartRate` antara 150 dan 200. Dengan demikian, catatan yang memiliki nilai `heartRate` antara 150 dan 200 adalah anomali.

2. Konfigurasi input – Dengan menggunakan konsol, Anda membuat aplikasi Kinesis Data Analytics dan mengonfigurasi input aplikasi dengan memetakan sumber streaming ke aliran dalam aplikasi (`SOURCE_SQL_STREAM_001`). Saat aplikasi dimulai, Kinesis Data Analytics terus membaca sumber streaming dan memasukkan catatan ke dalam aliran dalam aplikasi.

3. Tentukan kode aplikasi – Contoh ini menggunakan kode aplikasi berikut:

```
--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;
```

Kode membaca baris dalam `SOURCE_SQL_STREAM_001`, menetapkan skor anomali, dan menulis baris yang dihasilkan ke aliran dalam aplikasi lainnya (`TEMP_STREAM`). Kode aplikasi kemudian mengurutkan catatan di `TEMP_STREAM` dan menyimpan hasilnya ke aliran dalam

aplikasi lainnya (`DESTINATION_SQL_STREAM`). Anda menggunakan pompa untuk memasukkan baris di aliran dalam aplikasi. Untuk informasi selengkapnya, lihat Aliran dan Pompa dalam Aplikasi.

4. Konfigurasi output – Anda mengonfigurasi output aplikasi untuk menyimpan data dalam `DESTINATION_SQL_STREAM` ke tujuan eksternal, yang merupakan Kinesis data stream lainnya. Meninjau skor anomali yang ditetapkan untuk setiap catatan dan dan menentukan skor yang menunjukkan anomali (dan Anda perlu diberi tahu) adalah hal di luar aplikasi. Anda dapat menggunakan fungsi AWS Lambda untuk memproses skor anomali ini dan mengonfigurasi pemberitahuan.

Latihan ini menggunakan US East (N. Virginia) (`us-east-1`) untuk membuat aliran ini dan aplikasi Anda. Jika Anda menggunakan Wilayah lain, Anda harus memperbarui kode tersebut dengan sesuai.

Topik

- Langkah 1: Siapkan
- Langkah 2: Buat Aplikasi
- Langkah 3: Konfigurasi Output Aplikasi
- Langkah 4: Verifikasi Output

Langkah Selanjutnya

Langkah 1: Siapkan

Langkah 1: Siapkan

Sebelum membuat aplikasi Amazon Kinesis Data Analytics untuk latihan ini, Anda harus membuat dua Kinesis data streams. Konfigurasikan salah satu aliran sebagai sumber streaming untuk aplikasi Anda, dan aliran lainnya sebagai tujuan tempat Kinesis Data Analytics menyimpan output aplikasi Anda.

Topik

- Langkah 1.1: Buat Aliran Data Input dan Output
- Langkah 1.2: Menulis Catatan Sampel ke Aliran Input

Langkah 1.1: Buat Aliran Data Input dan Output

Di bagian ini, Anda akan membuat dua aliran Kinesis: `ExampleInputStream` dan `ExampleOutputStream`. Anda dapat membuat aliran ini menggunakan AWS Management Console atau AWS CLI.

- Untuk menggunakan konsol

  1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
  2. Pilih Create data stream (Buat aliran data). Buat aliran dengan satu serpihan bernama `ExampleInputStream`. Untuk informasi selengkapnya, lihat Buat Aliran di Panduan Developer Amazon Kinesis Data Streams.
  3. Ulangi langkah sebelumnya, yang membuat aliran dengan satu serpihan bernama `ExampleOutputStream`.

- Untuk menggunakan AWS CLI

  1. Gunakan perintah `create-stream` AWS CLI Kinesis berikut untuk membuat aliran pertama (`ExampleInputStream`).

     ```
     $ aws kinesis create-stream \
     --stream-name ExampleInputStream \
     --shard-count 1 \
     --region us-east-1 \
     --profile adminuser
     ```

  2. Jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`. Perintah ini membuat aliran kedua yang digunakan aplikasi untuk menulis output.

Langkah 1.2: Menulis Catatan Sampel ke Aliran Input

Dalam langkah ini, Anda menjalankan kode Python untuk terus menghasilkan catatan sampel dan menulis catatan ini ke aliran `ExampleInputStream`.

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

1. Instal Python dan `pip`.

   Untuk informasi tentang menginstal Python, lihat situs web Python.

   Anda dapat menginstal dependensi menggunakan pip. Untuk informasi tentang menginstal pip, lihat Penginstalan di situs web pip.

2. Jalankan kode Phyton berikut. Perintah `put-record` dalam kode menulis catatan JSON ke aliran.

```
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"))
```

Langkah Selanjutnya

Langkah 2: Buat Aplikasi

Langkah 2: Buat Aplikasi

Di bagian ini, Anda membuat aplikasi Amazon Kinesis Data Analytics sebagai berikut:

- Konfigurasikan input aplikasi untuk menggunakan Kinesis data stream yang Anda buat di the section called “Langkah 1: Siapkan” sebagai sumber streaming.
- Gunakan templat Deteksi Anomali di konsol.

Untuk membuat aplikasi

1. Ikuti langkah 1, 2, dan 3 dalam latihan Memulai Kinesis Data Analytics (lihat Langkah 3.1: Buat Aplikasi).

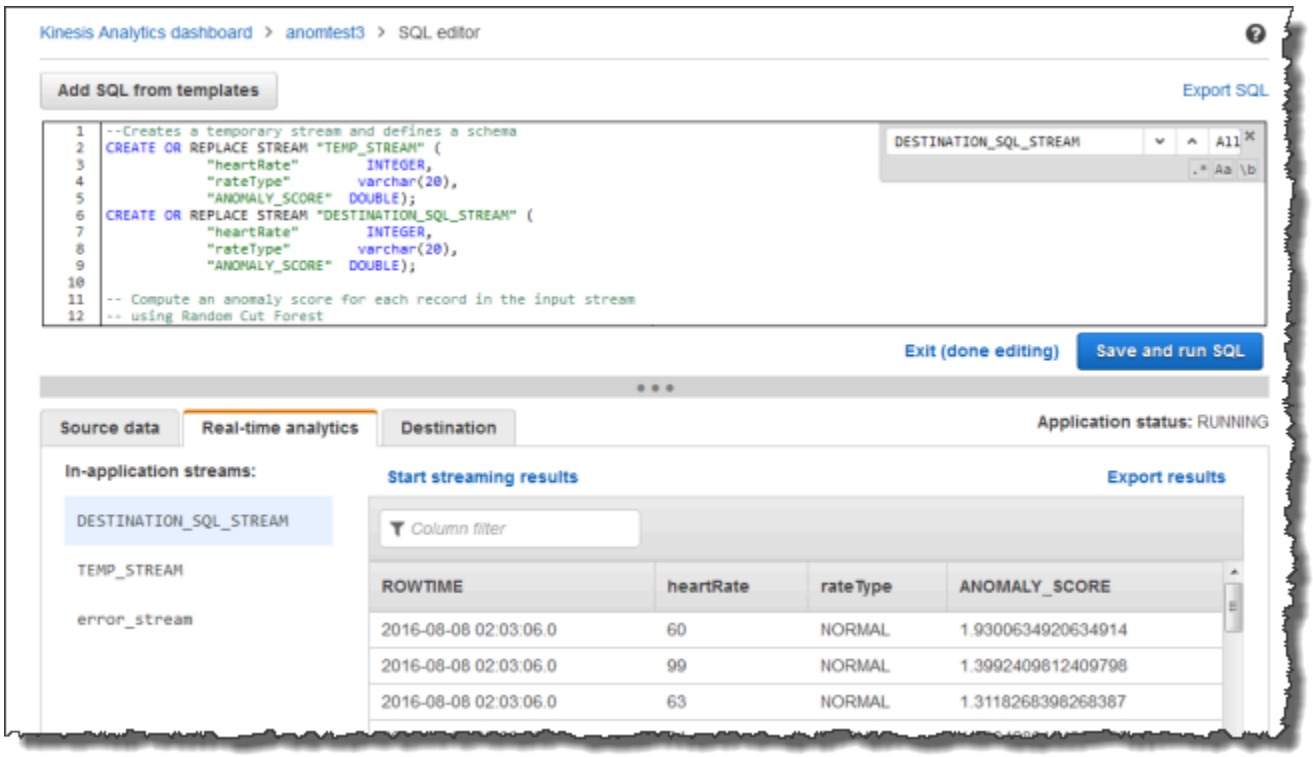
   - Dalam konfigurasi sumber, lakukan hal berikut:
     - Tentukan sumber streaming yang Anda buat di bagian sebelumnya.
     - Setelah konsol menyimpulkan skema, edit skema, dan tetapkan tipe kolom `heartRate` ke `INTEGER`.

       Sebagian besar nilai detak jantung normal, dan proses penemuan kemungkinan besar akan menetapkan tipe `TINYINT` ke kolom ini. Namun sebagian kecil nilai menunjukkan detak jantung yang tinggi. Jika nilai tinggi ini tidak cocok dengan tipe `TINYINT`, Kinesis Data Analytics mengirimkan baris ini ke aliran kesalahan. Perbarui tipe data ke `INTEGER` agar dapat menampung semua data detak jantung yang dihasilkan.
   - Gunakan templat Deteksi Anomali di konsol. Anda kemudian memperbarui kode templat untuk memberikan nama kolom yang sesuai.

2. Perbarui kode aplikasi dengan menyediakan nama kolom. Kode aplikasi yang dihasilkan ditampilkan di bawah ini (tempel kode ini ke editor SQL):

```
--Creates a temporary stream.
CREATE OR REPLACE STREAM "TEMP_STREAM" (
         "heartRate"        INTEGER,
         "rateType"         varchar(20),
         "ANOMALY_SCORE"    DOUBLE);

--Creates another stream for application output.
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
         "heartRate"        INTEGER,
         "rateType"         varchar(20),
         "ANOMALY_SCORE"    DOUBLE);

-- Compute an anomaly score for each record in the input stream
-- using Random Cut Forest
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "TEMP_STREAM"
      SELECT STREAM "heartRate", "rateType", ANOMALY_SCORE
      FROM TABLE(RANDOM_CUT_FOREST(
              CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001")));

-- Sort records by descending anomaly score, insert into output stream
CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM * FROM "TEMP_STREAM"
      ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE DESC;
```

3. Jalankan kode SQL dan tinjau hasilnya di konsol Kinesis Data Analytics:



Langkah Selanjutnya

[Langkah 3: Konfigurasi Output Aplikasi](#)

Langkah 3: Konfigurasi Output Aplikasi

Setelah menyelesaikan the section called “Langkah 2: Buat Aplikasi”, Anda memiliki kode aplikasi yang membaca data detak jantung dari sumber streaming dan menetapkan skor anomali untuk masing-masing.

Sekarang Anda dapat mengirim hasil aplikasi dari aliran dalam aplikasi ke tujuan eksternal, yang merupakan aliran data Kinesis lainnya (`OutputStreamTestingAnomalyScores`). Anda dapat menganalisis skor anomali dan menentukan detak jantung mana yang anomali. Anda kemudian dapat memperluas aplikasi ini lebih lanjut untuk menghasilkan pemberitahuan.

Ikuti langkah berikut untuk mengonfigurasi output aplikasi:

1. Buka konsol Amazon Kinesis Data Analytics. Di editor SQL, pilih Destination (Tujuan) atau Add a destination (Tambahkan tujuan) di dasbor aplikasi.

2. Di halaman Sambungkan ke tujuan, pilih aliran `OutputStreamTestingAnomalyScores` yang Anda buat di bagian sebelumnya.

   Sekarang Anda memiliki tujuan eksternal, tempat Amazon Kinesis Data Analytics menyimpan catatan yang ditulis aplikasi Anda ke aliran dalam aplikasi `DESTINATION_SQL_STREAM`.

3. Anda dapat mengonfigurasi AWS Lambda secara opsional untuk memantau aliran `OutputStreamTestingAnomalyScores` dan mengirimkan pemberitahuan. Untuk petunjuk, lihat Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda. Jika tidak menetapkan pemberitahuan, Anda dapat meninjau catatan yang ditulis oleh Kinesis Data Analytics ke tujuan eksternal, yaitu Kinesis data stream `OutputStreamTestingAnomalyScores`, seperti yang dijelaskan dalam Langkah 4: Verifikasi Output.

Langkah Selanjutnya

Langkah 4: Verifikasi Output

Langkah 4: Verifikasi Output

Setelah mengonfigurasi output aplikasi di the section called “Langkah 3: Konfigurasi Output Aplikasi”, gunakan perintah AWS CLI berikut untuk membaca catatan dalam aliran tujuan yang ditulis oleh aplikasi:

1. Jalankan perintah `get-shard-iterator` untuk mendapatkan pointer ke data pada aliran output.

   ```
   aws kinesis get-shard-iterator \
   --shard-id shardId-000000000000 \
   --shard-iterator-type TRIM_HORIZON \
   --stream-name OutputStreamTestingAnomalyScores \
   --region us-east-1 \
   --profile adminuser
   ```

   Anda mendapatkan respons dengan nilai iterator serpihan, seperti yang ditunjukkan dalam contoh respons berikut:

   ```
   {
       "ShardIterator":
       "shard-iterator-value"   }
   ```

Salin nilai iterator serpihan.

2. Jalankan perintah AWS CLI `get-records`.

```
aws kinesis get-records \
--shard-iterator shared-iterator-value \
--region us-east-1 \
--profile adminuser
```

Perintah mengembalikan halaman catatan dan iterator serpihan lain yang dapat Anda gunakan dalam perintah `get-records` berikutnya untuk mengambil serangkaian berikutnya catatan.

Contoh: Mendeteksi Anomali Data dan Mendapatkan Penjelasan (Fungsi RANDOM_CUT_FOREST_WITH_EXPLANATION)

Amazon Kinesis Data Analytics menyediakan fungsi `RANDOM_CUT_FOREST_WITH_EXPLANATION`, yang menetapkan skor anomali ke setiap catatan berdasarkan nilai-nilai dalam kolom numerik. Fungsi ini juga memberikan penjelasan tentang anomali. Untuk informasi selengkapnya, lihat RANDOM_CUT_FOREST_WITH_EXPLANATION di Amazon Managed Service for Apache Flink SQL Reference.

Dalam latihan ini, Anda menulis kode aplikasi untuk mendapatkan skor anomali untuk catatan di sumber streaming aplikasi Anda. Anda juga mendapatkan penjelasan untuk setiap anomali.

Topik

- Langkah 1: Siapkan Data
- Langkah 2: Buat Aplikasi Analitik
- Langkah 3: Periksa Hasilnya

Langkah Pertama

Langkah 1: Siapkan Data

Langkah 1: Siapkan Data

Sebelum Anda membuat aplikasi Amazon Kinesis Data Analytics untuk contoh, Anda membuat Kinesis data stream untuk digunakan sebagai sumber streaming untuk aplikasi Anda. Anda juga menjalankan kode Python untuk menulis simulasi data tekanan darah ke aliran.

Topik

- Langkah 1.1: Buat Kinesis Data Stream
- Langkah 1.2: Tulis Catatan Sampel ke Aliran Input

Langkah 1.1: Buat Kinesis Data Stream

Dalam bagian ini, Anda membuat Kinesis data stream bernama `ExampleInputStream`. Anda dapat membuat aliran data ini menggunakan AWS Management Console atau AWS CLI.

- Untuk menggunakan konsol:

  1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
  2. Pilih Data Streams (Aliran Data) di panel navigasi. Selanjutnya pilih Create Kinesis stream (Buat Kinesis stream).
  3. Untuk nama, masukkan **`ExampleInputStream`**. Untuk jumlah serpihan, masukkan **`1`**.
- Atau, untuk menggunakan AWS CLI guna membuat aliran data, jalankan perintah berikut:

```
$ aws kinesis create-stream --stream-name ExampleInputStream --shard-count 1
```

Langkah 1.2: Tulis Catatan Sampel ke Aliran Input

Dalam langkah ini, Anda menjalankan kode Python untuk terus membuat catatan sampel dan menulisnya ke aliran data yang Anda buat.

1. Instal Python dan pip.

   Untuk informasi tentang menginstal Python, lihat Python.

   Anda dapat menginstal dependensi menggunakan pip. Untuk informasi tentang menginstal pip, lihat Penginstalan di dokumentasi pip.
2. Jalankan kode Phyton berikut. Anda dapat mengubah Wilayah ke salah satu yang ingin Anda gunakan untuk contoh ini. Perintah `put-record` dalam kode menulis catatan JSON ke aliran.

```
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"))
```

Langkah Selanjutnya

[Langkah 2: Buat Aplikasi Analitik]

Langkah 2: Buat Aplikasi Analitik

Dalam bagian ini, Anda membuat aplikasi Amazon Kinesis Data Analytics dan mengonfigurasinya untuk menggunakan Kinesis data stream yang Anda buat sebagai sumber streaming di the section called "Langkah 1: Siapkan Data". Anda kemudian menjalankan kode aplikasi yang menggunakan fungsi `RANDOM_CUT_FOREST_WITH_EXPLANATION`.

Untuk membuat aplikasi

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Pilih Data Analytics (Analitik Data) di panel navigasi, lalu pilih Create application (Buat aplikasi).
3. Beri nama dan deskripsi aplikasi (opsional), dan pilih Create application (Buat aplikasi).
4. Pilih Connect streaming data, lalu pilih ExampleInputStreamdari daftar.
5. Pilih Discover schema (Temukan skema), dan pastikan `Systolic` dan `Diastolic` muncul sebagai kolom `INTEGER`. Jika memiliki tipe lain, pilih Edit schema (Edit skema), dan tetapkan tipe `INTEGER` ke keduanya.
6. Di bawah Real time analytics (Analitik waktu nyata), pilih Go to SQL editor (Buka editor SQL). Saat diminta, pilih untuk menjalankan aplikasi Anda.
7. Tempel mode berikut ke editor SQL, lalu pilih Save and run SQL (Simpan dan jalankan 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;
```

Langkah Selanjutnya

Langkah 3: Periksa Hasilnya

Langkah 3: Periksa Hasilnya

Ketika Anda menjalankan kode SQL untuk contoh ini, Anda pertama-tama melihat baris dengan skor anomali sama dengan nol. Hal ini terjadi selama tahap pembelajaran awal. Anda akan melihat hasil yang serupa dengan yang berikut:

```
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"}}
```

- Algoritme dalam fungsi `RANDOM_CUT_FOREST_WITH_EXPLANATION` melihat kolom `Systolic` dan `Diastolic` adalah numerik, dan menggunakannya sebagai input.
- Kolom `BloodPressureLevel` memiliki data teks, dan oleh karena itu tidak diperhitungkan oleh algoritme. Kolom ini hanya bantuan visual untuk membantu Anda melihat tingkat tekanan darah normal, tinggi, dan rendah dalam contoh ini dengan cepat.
- Di kolom `ANOMALY_SCORE`, catatan dengan skor yang lebih tinggi lebih anomali. Catatan kedua dalam kumpulan sampel hasil ini adalah yang paling anomali, dengan skor anomali 3,855851061.
- Untuk memahami sejauh mana masing-masing kolom numerik yang diperhitungkan oleh algoritme berkontribusi pada skor anomali, lihat bidang JSON bernama `ATTRIBUTION_SCORE` di kolom `ANOMALY_SCORE`. Dalam kasus baris kedua dalam kumpulan hasil sampel ini, kolom `Systolic` dan `Diastolic` berkontribusi terhadap anomali dalam rasio 1,7447:2,1111. Dengan kata lain, 45 persen penjelasan untuk skor anomali disebabkan oleh nilai sistolik, dan atribusi yang tersisa disebabkan oleh nilai diastolik.
- Untuk menentukan arah tempat titik yang diwakili oleh baris kedua dalam sampel ini adalah anomali, lihat bidang JSON bernama `DIRECTION`. Kedua nilai diastolik dan sistolik ditandai sebagai `HIGH` dalam kasus ini. Untuk menentukan keyakinan arah mana yang benar, lihat bidang JSON bernama `STRENGTH`. Dalam contoh ini, algoritme lebih yakin bahwa nilai diastolik tinggi. Memang, nilai normal untuk pembacaan diastolik biasanya 60-80, dan 123 jauh lebih tinggi dari yang diharapkan.

Contoh: Mendeteksi Hotspot di Aliran (Fungsi HOTSPOTS)

Amazon Kinesis Data Analytics menyediakan fungsi `HOTSPOTS`, yang dapat menemukan dan mengembalikan informasi tentang wilayah yang relatif padat dalam data Anda. Untuk informasi selengkapnya, lihat HOTSPOT di Amazon Managed Service for Apache Flink SQL Reference.

Dalam latihan ini, Anda menulis kode aplikasi untuk menemukan hotspot pada sumber streaming aplikasi Anda. Untuk menyiapkan aplikasi, Anda melakukan langkah-langkah berikut:

1. Siapkan sumber streaming – Anda menyiapkan aliran Kinesis dan menulis data koordinat sampel, seperti yang ditunjukkan berikut:

   ```
   {"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
   {"x": 0.722248626528026, "y": 4.648868803193405, "is_hot": "Y"}
   ```

   Contoh ini memberi Anda skrip Python untuk mengisi aliran. Nilai `x` dan `y` dibuat secara acak, dengan beberapa catatan yang diklasterkan di sekitar lokasi tertentu.

   Bidang `is_hot` disediakan sebagai indikator jika skrip sengaja membuat nilai sebagai bagian dari hotspot. Ini dapat membantu Anda mengevaluasi apakah fungsi deteksi hotspot bekerja dengan benar.
2. Buat aplikasi — MenggunakanAWS Management Console, Anda kemudian membuat aplikasi Kinesis Data Analytics. Konfigurasikan input aplikasi dengan memetakan sumber streaming ke aliran dalam aplikasi (`SOURCE_SQL_STREAM_001`). Saat aplikasi dimulai, Kinesis Data Analytics terus membaca sumber streaming dan memasukkan catatan ke dalam aliran dalam aplikasi.

   Dalam latihan ini, Anda menggunakan kode berikut untuk aplikasi:

   ```
   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)
    );
```

Kode membaca baris di `SOURCE_SQL_STREAM_001`, menganalisisnya untuk hotspot yang signifikan, dan menulis data yang dihasilkan ke aliran dalam aplikasi lainnya (`DESTINATION_SQL_STREAM`). Anda menggunakan pompa untuk memasukkan baris di aliran dalam aplikasi. Untuk informasi selengkapnya, lihat Aliran dan Pompa dalam Aplikasi.

3. Konfigurasikan output – Anda mengonfigurasi output aplikasi untuk mengirim data dari aplikasi ke tujuan eksternal, yang merupakan Kinesis data stream lainnya. Tinjau skor hotspot dan tentukan nilai apa yang menunjukkan hotspot terjadi (dan Anda perlu diberi tahu). Anda dapat menggunakan fungsi AWS Lambda untuk memproses informasi hotspot lebih lanjut dan mengonfigurasi pemberitahuan.
4. Verifikasi output - Contoh termasuk JavaScript aplikasi yang membaca data dari aliran output dan menampilkannya secara grafis, sehingga Anda dapat melihat hotspot yang dihasilkan aplikasi secara real time.

Latihan ini menggunakan US West (Oregon) (`us-west-2`) untuk membuat aliran ini dan aplikasi Anda. Jika Anda menggunakan Wilayah lain, perbarui kode tersebut dengan sesuai.

Topik

- Langkah 1 Buat Aliran Input dan Output
- Langkah 2: Buat Aplikasi Kinesis Data Analytics
- Langkah 3: Konfigurasikan Output Aplikasi
- Langkah 4: Verifikasi Output Aplikasi

Langkah 1 Buat Aliran Input dan Output

Sebelum membuat aplikasi Amazon Kinesis Data Analytics untuk Latihan hotspots, Anda membuat dua Kinesis data streams. Konfigurasikan salah satu aliran sebagai sumber streaming untuk aplikasi Anda, dan aliran lainnya sebagai tujuan tempat Kinesis Data Analytics menyimpan output aplikasi Anda.

Topik

- Langkah 1.1: Buat Kinesis Data Streams
- Langkah 1.2: Tulis Catatan Sampel ke Aliran Input

Langkah 1.1: Buat Kinesis Data Streams

Di bagian ini, Anda membuat dua Kinesis data streams: `ExampleInputStream` dan `ExampleOutputStream`.

Buat aliran data ini menggunakan konsol atau AWS CLI.

- Buat aliran data ini menggunakan konsol:

  1. Masuk ke AWS Management Console dan buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
  2. Pilih Data Streams (Aliran Data) di panel navigasi.
  3. Pilih Create Kinesis stream (Buat Aliran Kinesis), dan buat aliran dengan satu serpihan bernama `ExampleInputStream`.
  4. Ulangi langkah sebelumnya, yang membuat aliran dengan satu serpihan bernama `ExampleOutputStream`.

- Buat aliran data ini menggunakan AWS CLI:

  - Buat aliran (`ExampleInputStream` dan `ExampleOutputStream`) menggunakan perintah `create-stream` AWS CLI Kinesis berikut. Untuk membuat aliran kedua, aplikasi mana yang akan digunakan untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `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
```

Langkah 1.2: Tulis Catatan Sampel ke Aliran Input

Dalam langkah ini, Anda menjalankan kode Python untuk terus membuat catatan sampel dan menulis ke aliran `ExampleInputStream`.

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626580026, "y": 4.648868803193405, "is_hot": "Y"}
```

1. Instal Python dan `pip`.

   Untuk informasi tentang menginstal Python, lihat situs web Python.

   Anda dapat menginstal dependensi menggunakan pip. Untuk informasi tentang menginstal pip, lihat Penginstalan di situs web pip.

2. Jalankan kode Phyton berikut. Kode ini melakukan hal berikut:

   - Membuat potensi hotspot di suatu tempat di bidang (X, Y).
   - Membuat satu kumpulan 1.000 poin untuk setiap hotspot. Dari titik-titik ini, 20 persen diklasterkan di sekitar hotspot. Sisanya dihasilkan secara acak di seluruh ruang.
   - Perintah `put-record` menulis catatan JSON ke aliran.

   > **Important**
   >
   > Jangan unggah file ini ke server web karena berisi kredensial AWS Anda.

```
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"),
    )
```

Langkah Selanjutnya

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Langkah 2: Buat Aplikasi Kinesis Data Analytics

Di bagian contoh Hotspot ini, Anda membuat aplikasi Kinesis Data Analytics sebagai berikut:

- Konfigurasikan input aplikasi untuk menggunakan Kinesis data stream yang Anda buat sebagai sumber streaming di Langkah 1.
- Gunakan kode aplikasi yang disediakan di AWS Management Console.

Untuk membuat aplikasi

1. Buat aplikasi Kinesis Data Analytics dengan mengikuti langkah 1, 2, dan 3 dalam latihan Memulai (Langkah 3.1: Buat Aplikasilihat).

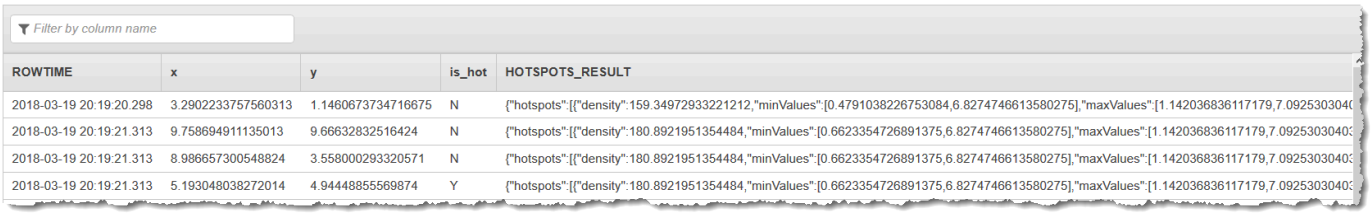
   Dalam konfigurasi sumber, lakukan hal berikut:

   - Tentukan sumber streaming yang Anda buat di the section called “Langkah 1: Buat Aliran”.
   - Setelah konsol menyimpulkan skema, edit skema. Pastikan tipe kolom `x` dan `y` ditetapkan ke `DOUBLE` dan tipe kolom `IS_HOT` ditetapkan ke `VARCHAR`.
2. Gunakan kode aplikasi berikut (Anda dapat menempelkan kode ini ke editor SQL):

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
    "x" DOUBLE,
    "y" DOUBLE,
    "is_hot" VARCHAR(4),
    HOTSPOTS_RESULT VARCHAR(10000)
);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
    INSERT INTO "DESTINATION_SQL_STREAM"
```

```
SELECT "x", "y", "is_hot", "HOTSPOTS_RESULT"
FROM TABLE (
    HOTSPOTS(
        CURSOR(SELECT STREAM "x", "y", "is_hot" FROM "SOURCE_SQL_STREAM_001"),
        1000,
        0.2,
        17)
);
```

3. Jalankan kode SQL dan tinjau hasilnya.



Langkah Selanjutnya

[Langkah 3: Konfigurasikan Output Aplikasi]

Langkah 3: Konfigurasikan Output Aplikasi

Pada titik ini di Contoh hotspot, Anda memiliki kode aplikasi Amazon Kinesis Data Analytics yang menemukan hotspot yang signifikan dari sumber streaming dan menetapkan skor heat untuk masing-masing.

Sekarang Anda dapat mengirim hasil aplikasi dari aliran dalam aplikasi ke tujuan eksternal, yang merupakan Kinesis data stream lainnya (`ExampleOutputStream`). Anda kemudian dapat menganalisis skor hotspot dan menentukan ambang yang tepat untuk hotspot heat. Anda dapat memperluas aplikasi ini lebih lanjut untuk menghasilkan pemberitahuan.

Untuk mengonfigurasi output aplikasi

1. Buka konsol Kinesis Data Analytics di https://console.aws.amazon.com/kinesisanalytics.
2. Di editor SQL, pilih Destination (Tujuan) atau Add a destination (Tambahkan tujuan) di dasbor aplikasi.
3. Di halaman Tambahkan tujuan, pilih Select from your streams (Pilih dari aliran Anda). Kemudian pilih aliran `ExampleOutputStream` yang Anda buat di bagian sebelumnya.

Sekarang Anda memiliki tujuan eksternal, tempat Amazon Kinesis Data Analytics menyimpan catatan yang ditulis aplikasi Anda ke aliran dalam aplikasi `DESTINATION_SQL_STREAM`.

4. Anda dapat mengonfigurasi AWS Lambda secara opsional untuk memantau aliran `ExampleOutputStream` dan mengirimkan pemberitahuan. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda sebagai Output. Anda juga dapat meninjau catatan yang ditulis oleh Kinesis Data Analytics ke tujuan eksternal, yaitu aliran Kinesis `ExampleOutputStream`, seperti yang dijelaskan dalam Langkah 4: Verifikasi Output Aplikasi.

Langkah Selanjutnya

Langkah 4: Verifikasi Output Aplikasi

Langkah 4: Verifikasi Output Aplikasi

Dalam bagian Contoh hotspot ini , Anda mengatur aplikasi web yang menampilkan informasi hotspot di kontrol Scalable Vector Graphics (SVG).

1. Buat file bernama `index.html` dengan konten berikut:

```
<!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. Buat file teks di direktori yang sama bernama `hotspots_viewer.js` dengan konten berikut. Berikan Anda, kredensial, dan nama aliran output dalam variabel yang disediakan.

```
// 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
```

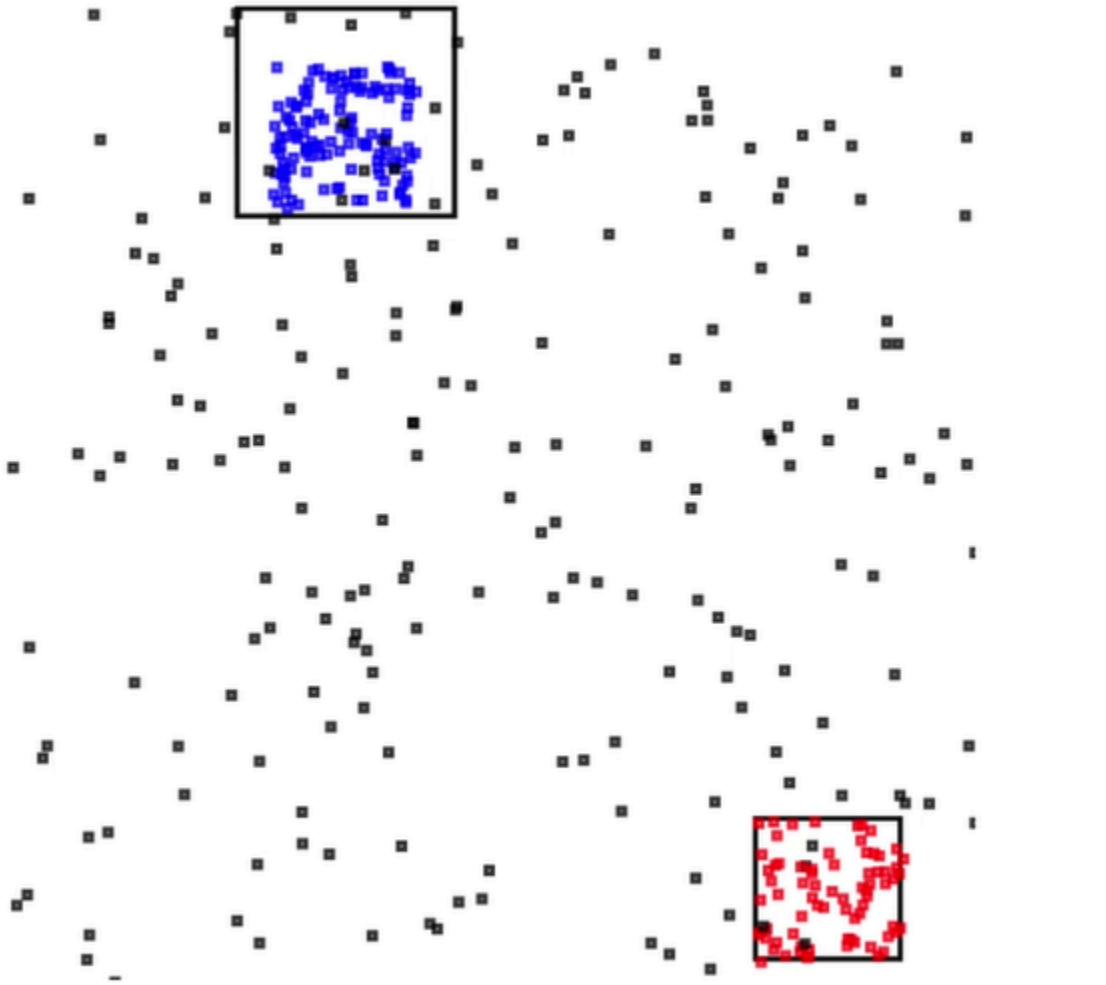
```
////////////////////////////////////////////////////////////////////////////////////////////
// D3 setup
////////////////////////////////////////////////////////////////////////////////////////////

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. Dengan kode Python dari bagian pertama yang berjalan, buka `index.html` di peramban web. Informasi hotspot muncul di halaman, seperti yang ditunjukkan berikut ini.



Contoh: Peringatan dan Kesalahan

Bagian ini memberikan contoh aplikasi Kinesis Data Analytics yang menggunakan peringatan dan kesalahan. Setiap contoh memberikan step-by-step instruksi dan kode untuk membantu Anda menyiapkan dan menguji aplikasi Kinesis Data Analytics Anda.

Topik

- Contoh: Membuat Peringatan Sederhana
- Contoh: Membuat Peringatan Terbatas
- Contoh: Menjelajahi Aliran Kesalahan dalam Aplikasi

Contoh: Membuat Peringatan Sederhana

Dalam aplikasi Kinesis Data Analytics ini, kueri berjalan terus menerus pada aliran dalam aplikasi yang dibuat melalui aliran demo. Untuk informasi selengkapnya, lihat Kueri Berkelanjutan.

Jika setiap baris menunjukkan perubahan harga saham yang lebih besar dari 1 persen, baris tersebut dimasukkan ke aliran dalam aplikasi lain. Dalam latihan, Anda dapat mengonfigurasi output aplikasi untuk menyimpan hasil ke tujuan eksternal. Anda selanjutnya dapat menyelidiki hasilnya lebih lanjut. Misalnya, Anda dapat menggunakan fungsi AWS Lambda untuk memproses catatan dan mengirimkan peringatan.

Untuk membuat aplikasi peringatan sederhana

1. Buat aplikasi analitik seperti yang dijelaskan dalam latihan Memulai Kinesis Data Analytics.
2. Di editor SQL dalam Kinesis Data Analytics, ganti kode aplikasi dengan berikut ini:

```
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;
```

   Pernyataan `SELECT` dalam kode aplikasi memfilter baris dalam `SOURCE_SQL_STREAM_001` untuk perubahan harga saham yang lebih besar dari 1 persen. Ini selanjutnya memasukkan baris ke aliran lain dalam aplikasi `DESTINATION_SQL_STREAM` menggunakan pompa. Untuk informasi selengkapnya tentang pola pengkodean yang menjelaskan penggunaan pompa untuk memasukkan baris ke aliran dalam aplikasi, lihat Kode Aplikasi.
3. Pilih Save and run SQL (Simpan dan jalankan SQL).
4. Tambahkan tujuan. Untuk melakukan ini, pilih tab Destination (Tujuan) di editor SQL atau pilih Add a destination (Tambahkan tujuan) pada halaman detail aplikasi.

   a. Di editor SQL, pilih tab Destination (Tujuan), lalu pilih Connect to a destination (Sambungkan ke tujuan).

Di halaman Sambungkan ke tujuan, pilih Create New (Buat Baru).

b. Pilih Go to Kinesis Streams (Buka Aliran Kinesis).

c. Pada konsol Amazon Kinesis Data Streams, buat aliran Kinesis baru (misalnya, `gs-destination`) dengan satu serpihan. Tunggu hingga status aliran ACTIVE (AKTIF).

d. Kembali ke konsol Kinesis Data Analytics. Di halaman Sambungkan ke tujuan, pilih aliran yang Anda buat.

Jika aliran tidak muncul, refresh halaman.

e. Jangan pilih Save and continue (Simpan dan lanjutkan).

Sekarang Anda memiliki tujuan eksternal, Kinesis Data Stream, tempat Kinesis Data Analytics menyimpan output aplikasi Anda di aliran dalam aplikasi `DESTINATION_SQL_STREAM`.

5. Konfigurasi AWS Lambda untuk memantau aliran Kinesis yang Anda buat dan memanggil fungsi Lambda.

Untuk petunjuk, lihat Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda.

Contoh: Membuat Peringatan Terbatas

Dalam aplikasi Kinesis Data Analytics ini, kueri berjalan terus menerus pada aliran dalam aplikasi yang dibuat melalui aliran demo. Untuk informasi selengkapnya, lihat Kueri Berkelanjutan. Jika setiap baris yang menunjukkan perubahan harga saham lebih besar dari 1 persen, baris tersebut dimasukkan ke aliran dalam aplikasi lain. Aplikasi membatasi peringatan sehingga peringatan segera dikirim ketika harga saham berubah. Namun, tidak lebih dari satu peringatan per menit per simbol saham dikirim ke aliran dalam aplikasi.

Untuk membuat aplikasi peringatan terbatas

1. Buat aplikasi Kinesis Data Analytics seperti yang dijelaskan dalam latihan Kinesis Data Analytics Getting Started.
2. Di editor SQL dalam Kinesis Data Analytics, ganti kode aplikasi dengan berikut ini:

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

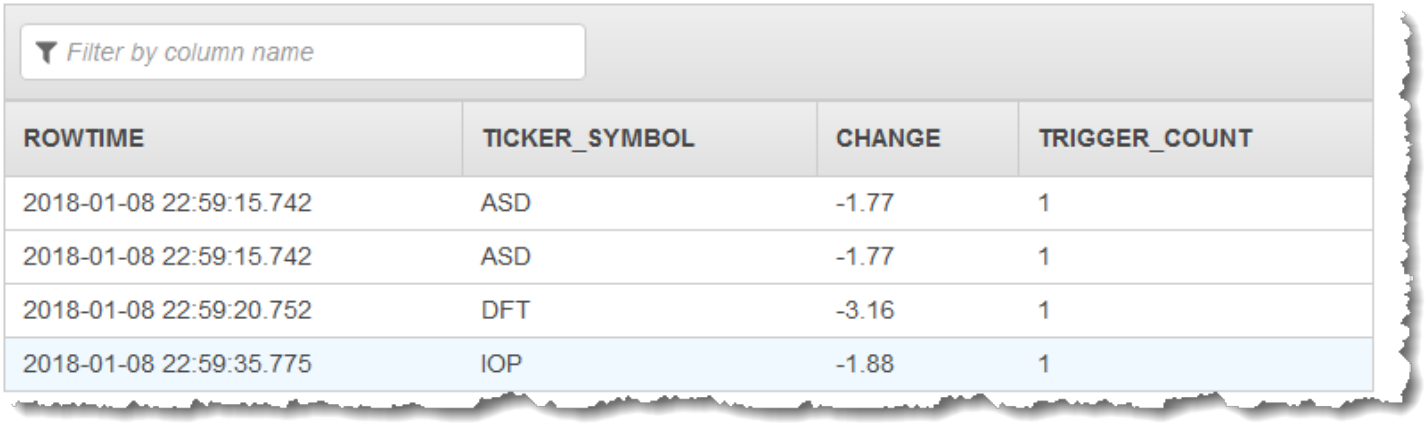
```
            price         DOUBLE);

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;
```

Pernyataan SELECT dalam kode aplikasi memfilter baris dalam SOURCE_SQL_STREAM_001 untuk perubahan harga stok yang lebih besar dari 1 persen dan memasukkan baris ke aliran dalam aplikasi lainnya CHANGE_STREAM menggunakan pompa.

Aplikasi ini selanjutnya membuat aliran kedua bernama TRIGGER_COUNT_STREAM untuk peringatan terbatas. Kueri kedua memilih catatan dari jendela yang melompat ke depan setiap kali catatan dimasukkan dalamnya, sehingga hanya satu catatan per ticker saham per menit yang ditulis ke aliran.

3. Pilih Save and run SQL (Simpan dan jalankan SQL).

Contoh menghasilkan aliran ke `TRIGGER_COUNT_STREAM` yang terlihat serupa seperti yang berikut ini:


Filter by column name

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

Contoh: Menjelajahi Aliran Kesalahan dalam Aplikasi

Amazon Kinesis Data Analytics menyediakan aliran kesalahan dalam aplikasi untuk setiap aplikasi yang Anda buat. Baris apa pun yang tidak dapat diproses aplikasi Anda dikirim ke aliran kesalahan ini. Anda mungkin mempertimbangkan untuk menyimpan data aliran kesalahan ke tujuan eksternal agar Anda dapat menyelidiki.

Anda dapat melakukan tindakan berikut di konsol. Dalam contoh ini, Anda memperkenalkan kesalahan dalam konfigurasi input dengan mengedit skema yang disimpulkan oleh proses penemuan, lalu Anda memverifikasi baris yang dikirim ke aliran kesalahan.

Topik

- Memperkenalkan Kesalahan Penguraian
- Memperkenalkan Kesalahan Membagi dengan Nol

Memperkenalkan Kesalahan Penguraian

Dalam latihan ini, Anda memperkenalkan kesalahan penguraian.

1. Buat aplikasi Kinesis Data Analytics seperti yang dijelaskan dalam latihan Kinesis Data Analytics Getting Started.
2. Pada halaman detail aplikasi, pilih Connect data streaming (Sambungkan data streaming).
3. Jika Anda mengikuti latihan Memulai, Anda memiliki aliran demo (`kinesis-analytics-demo-stream`) di akun Anda. Pada halaman Sambungkan ke sumber, pilih aliran demo ini.

4. Kinesis Data Analytics mengambil sampel dari aliran demo untuk menyimpulkan skema aliran input dalam aplikasi yang dibuat. Konsol menunjukkan skema yang disimpulkan dan data sampel di tab Formatted stream sample (Sampel aliran yang diformat).
5. Selanjutnya, edit skema dan ubah tipe kolom untuk memperkenalkan kesalahan penguraian. Pilih Edit schema (Edit skema).
6. Ubah tipe kolom `TICKER_SYMBOL` dari `VARCHAR(4)` ke `INTEGER`.

   Setelah tipe kolom skema dalam aplikasi yang dibuat tidak valid, Kinesis Data Analytics tidak dapat membawa data ke aliran dalam aplikasi. Sebaliknya, Kinesis Data Analytics akan mengirimkan baris ke aliran kesalahan.
7. Pilih Save schema (Simpan skema).
8. Pilih Refresh schema samples (Refresh sampel skema).

   Perhatikan bahwa tidak ada baris dalam sampel Aliran yang diformat. Namun, tab Error stream (Aliran kesalahan) menunjukkan data dengan pesan kesalahan. Tab Error stream (Aliran kesalahan) menampilkan data yang dikirim ke aliran kesalahan dalam aplikasi.

   Karena Anda mengubah tipe data kolom, Kinesis Data Analytics tidak dapat membawa data di aliran input dalam aplikasi. Sebaliknya, Kinesis Data Analytics mengirimkan data ke aliran kesalahan.

Memperkenalkan Kesalahan Membagi dengan Nol

Dalam latihan ini, Anda memperbarui kode aplikasi untuk memperkenalkan kesalahan runtime (pembagian dengan nol). Perhatikan bahwa Amazon Kinesis Data Analytics mengirimkan baris yang dihasilkan ke aliran kesalahan dalam aplikasi, bukan ke aliran dalam aplikasi `DESTINATION_SQL_STREAM` tempat hasilnya seharusnya ditulis.

1. Buat aplikasi Kinesis Data Analytics seperti yang dijelaskan dalam latihan Kinesis Data Analytics Getting Started.

   Verifikasi hasil di tab Real-time analytics (Analitik waktu nyata) sebagai berikut:

   Sour
2. Perbarui pernyataan `SELECT` dalam kode aplikasi untuk memperkenalkan membagi dengan nol; misalnya:

```
SELECT STREAM ticker_symbol, sector, change, (price / 0) as ProblemColumn
FROM "SOURCE_SQL_STREAM_001"
WHERE sector SIMILAR TO '%TECH%';
```

3. Jalankan aplikasi.

   Karena terjadi kesalahan runtime pembagian dengan nol, sebagai ganti menulis hasil ke `DESTINATION_SQL_STREAM`, Kinesis Data Analytics mengirimkan baris ke aliran kesalahan dalam aplikasi. Di tab Real-time analytics (Analitik waktu nyata), pilih aliran kesalahan, lalu Anda dapat melihat baris di aliran kesalahan dalam aplikasi.

Contoh: Akselerator Solusi

Situs Solusi AWS memiliki templat AWS CloudFormation yang tersedia yang dapat Anda gunakan untuk membuat solusi data streaming lengkap dengan cepat.

Templat berikut ini tersedia:

Wawasan waktu nyata tentang aktivitas Akun AWS

Solusi ini mencatat dan memvisualisasikan metrik akses dan penggunaan sumber daya Akun AWS secara langsung. Untuk informasi selengkapnya, lihat Wawasan waktu nyata tentang aktivitas Akun AWS.

Pemantauan perangkat AWS IoT secara langsung dengan Kinesis Data Analytics

Solusi ini mengumpulkan, memproses, menganalisis, dan memvisualisasikan konektivitas perangkat IoT dan data aktivitas secara langsung. Untuk informasi selengkapnya, lihat Pemantauan perangkat AWS IoT secara langsung dengan Kinesis Data Analytics.

Analitik web waktu nyata dengan Kinesis Data Analytics

Solusi ini mengumpulkan, memproses, menganalisis, dan memvisualisasikan clickstream data situs web secara langsung. Untuk informasi selengkapnya, lihat Analitik web secara langsung dengan Kinesis Data Analytics.

Solusi Kendaraan Terhubung Amazon

Solusi ini mengumpulkan, memproses, menganalisis, dan memvisualisasikan data IoT dari kendaraan secara langsung. Untuk informasi selengkapnya, lihat Solusi Kendaraan Terhubung Amazon.

Keamanan di

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda akan mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama ketika menggunakan . Topik berikut menunjukkan cara mengonfigurasi untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan belajar cara menggunakan layanan Amazon lain yang dapat membantu Anda memantau dan mengamankan sumber daya Anda.

Topik

- Perlindungan Data di Amazon Kinesis Data Analytics untuk Aplikasi SQL
- Identity and Access Management di Kinesis Data Analytics
- Otentikasi dan Kontrol Akses untuk
- Pemantauan
- Validasi Kepatuhan untuk Amazon Kinesis Data Analytics untuk Aplikasi SQL
- Ketahanan di Amazon Kinesis Data Analytics
- Keamanan Infrastruktur di Kinesis Data Analytics untuk Aplikasi SQL
- Praktik Terbaik Keamanan untuk Kinesis Data Analytics

Perlindungan Data di Amazon Kinesis Data Analytics untuk Aplikasi SQL

Anda dapat melindungi data Anda menggunakan alat yang disediakan oleh AWS. Kinesis Data Analytics dapat bekerja dengan layanan yang mendukung enkripsi data, termasuk Kinesis Data Streams, Firehose, dan Amazon S3.

Enkripsi Data di Kinesis Data Analytics

Enkripsi at Rest

Perhatikan hal berikut tentang mengenkripsi data at rest dengan Kinesis Data Analytics:

- Anda dapat mengenkripsi data pada aliran data Kinesis yang masuk menggunakan. StartStreamEncryption Untuk informasi selengkapnya, lihat Apa Itu Enkripsi Sisi Server untuk Kinesis Data Streams?.
- Data keluaran dapat dienkripsi saat istirahat menggunakan Firehose untuk menyimpan data dalam bucket Amazon S3 terenkripsi. Anda dapat menentukan kunci enkripsi yang digunakan bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci yang Dikelola KMS (SSE-KMS).
- Kode aplikasi Anda dienkripsi saat istirahat.
- Data referensi aplikasi Anda dienkripsi saat istirahat.

Enkripsi dalam Transit

Kinesis Data Analytics mengenkripsi semua data dalam transit. Enkripsi dalam transit diaktifkan untuk semua aplikasi Kinesis Data Analytics dan tidak dapat dinonaktifkan.

Kinesis Data Analytics mengenkripsi data dalam transit di skenario berikut:

- Data dalam transit dari Kinesis Data Streams ke Kinesis Data Analytics.
- Data dalam transit di antara komponen internal dalam Kinesis Data Analytics.
- Data dalam transit antara Kinesis Data Analytics dan Firehose.

Manajemen kunci

Enkripsi data di Kinesis Data Analytics menggunakan kunci yang dikelola layanan. Kunci yang dikelola pelanggan tidak didukung.

Identity and Access Management di Kinesis Data Analytics

Amazon Kinesis Data Analytics memerlukan izin untuk membaca catatan dari sumber streaming yang Anda tentukan dalam konfigurasi input aplikasi Anda. Amazon Kinesis Data Analytics juga memerlukan izin untuk menulis output aplikasi Anda ke aliran yang Anda tentukan dalam konfigurasi output aplikasi Anda.

Anda dapat memberikan izin ini dengan membuat IAM role yang dapat diasumsikan oleh Amazon Kinesis Data Analytics. Izin yang Anda berikan ke peran ini menentukan apa yang dapat dilakukan Amazon Kinesis Data Analytics saat layanan mengambil peran.

> **Note**
>
> Informasi di bagian ini berguna jika Anda ingin membuat IAM role sendiri. Saat Anda membuat aplikasi di konsol Amazon Kinesis Data Analytics, konsol dapat membuat IAM role untuk Anda saat itu. Konsol menggunakan konvensi penamaan berikut untuk IAM role yang dibuat:
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> Setelah peran dibuat, Anda dapat meninjau peran dan kebijakan terlampir di konsol IAM.

Setiap IAM role memiliki dua kebijakan yang dilampirkan padanya. Dalam kebijakan kepercayaan, Anda menentukan siapa yang dapat mengasumsikan peran tersebut. Dalam kebijakan izin (dapat ada satu atau lebih), Anda menentukan izin yang ingin Anda berikan ke peran ini. Bagian berikut menjelaskan kebijakan ini, yang dapat Anda gunakan saat Anda membuat IAM role.

Kebijakan Kepercayaan

Untuk memberi Amazon Kinesis Data Analytics izin mengambil peran untuk mengakses sumber streaming atau referensi, Anda dapat melampirkan kebijakan kepercayaan berikut ke IAM role tersebut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Kebijakan Izin

Jika Anda membuat IAM role untuk mengizinkan Amazon Kinesis Data Analytics membaca dari sumber streaming aplikasi, Anda harus memberikan izin untuk tindakan baca yang relevan. Bergantung pada sumber Anda (misalnya, aliran Kinesis, aliran pengiriman Firehose, atau sumber referensi di bucket Amazon S3), Anda dapat melampirkan kebijakan izin berikut.

Kebijakan Izin untuk Membaca Aliran Kinesis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/inputStreamName"
            ]
        }
    ]
}
```

Kebijakan Izin untuk Membaca Aliran Pengiriman Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:Get*"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-
id:deliverystream/inputFirehoseName"
            ]
        }
    ]
}
```

> **Note**
>
> Izin `firehose:Get*` mengacu pada penilai internal yang digunakan oleh Kinesis Data Analytics untuk mengakses aliran. Tidak ada akses publik untuk aliran pengiriman Firehose.

Jika Anda mengarahkan Amazon Kinesis Data Analytics untuk menulis output ke tujuan eksternal dalam konfigurasi output aplikasi, Anda harus memberikan izin berikut ke IAM role.

Kebijakan Izin untuk Menulis ke Aliran Kinesis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
```

```
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/output-stream-name"
            ]
        }
    ]
}
```

Kebijakan Izin untuk Menulis ke Aliran Pengiriman Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-id:deliverystream/output-
firehose-name"
            ]
        }
    ]
}
```

Kebijakan Izin untuk Membaca Sumber Data Referensi dari Bucket Amazon S3

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

Pencegahan confused deputy lintas layanan

Dalam AWS, peniruan lintas layanan dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (layanan yang disebut). Layanan panggilan dapat dimanipulasi untuk bertindak atas sumber daya pelanggan lain meskipun seharusnya tidak memiliki izin yang tepat, yang mengakibatkan masalah wakil yang membingungkan.

Untuk mencegah kebingungan deputi, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan menggunakan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda. Bagian ini berfokus pada pencegahan wakil kebingungan lintas layanan khusus untuk Kinesis Data Analytics. Namun, Anda dapat mempelajari lebih lanjut tentang topik ini di Bagian Deputi Masalah yang membingungkan dari Panduan Pengguna IAM.

Dalam konteks Kinesis Data Analytics untuk SQL, sebaiknya gunakan kunci konteks kondisi global aws SourceArn: dan aws SourceAccount: dalam kebijakan kepercayaan peran Anda untuk membatasi akses ke peran hanya pada permintaan yang dihasilkan oleh sumber daya yang diharapkan.

Gunakan `aws:SourceArn` jika Anda hanya ingin satu sumber daya dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Nilai `aws:SourceArn` harus berupa ARN dari sumber daya yang digunakan oleh Kinesis Data Analytics, yang ditentukan dengan format berikut:. `arn:aws:kinesisanalytics:region:account:resource`

Pendekatan yang direkomendasikan untuk masalah wakil yang membingungkan adalah dengan menggunakan kunci konteks kondisi `aws:SourceArn` global dengan ARN sumber daya penuh.

Jika Anda tidak mengetahui ARN lengkap dari sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan `aws:SourceArn` kunci dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya: `arn:aws:kinesisanalytics::111122223333:*`.

Sementara sebagian besar tindakan dalam Kinesis Data Analytics untuk SQL API CreateApplicationseperti AddApplicationInput, DeleteApplicationdan dibuat dalam konteks aplikasi

tertentu, DiscoverInputSchematindakan tidak dijalankan dalam konteks aplikasi apa pun. Itu berarti peran yang digunakan dalam tindakan ini tidak harus sepenuhnya menentukan sumber daya dalam kunci `SourceArn` kondisi. Berikut ini adalah contoh yang menggunakan ARN wildcard:

```
{
   ...
   "ArnLike":{
      "aws:SourceArn":"arn:aws:kinesisanalytics:us-east-1:123456789012:*"
   }
   ...
}
```

Peran default yang dihasilkan oleh Kinesis Data Analytics untuk SQL menggunakan wildcard ini. Ini memastikan menemukan skema input bekerja dengan mulus dalam pengalaman konsol. Namun, kami merekomendasikan untuk mengedit Kebijakan Kepercayaan untuk menggunakan ARN penuh setelah menemukan skema untuk menerapkan mitigasi wakil yang membingungkan sepenuhnya.

Kebijakan peran yang Anda berikan kepada Kinesis Data Analytics serta kebijakan kepercayaan peran yang dihasilkan untuk Anda dapat menggunakan kunci kondisi aws SourceArn: dan aws SourceAccount:.

Untuk melindungi dari masalah wakil yang membingungkan, lakukan langkah-langkah berikut:

Untuk melindungi dari masalah wakil yang membingungkan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Peran dan kemudian pilih peran yang ingin Anda ubah.
3. Pilih Edit kebijakan kepercayaan.
4. Pada halaman Edit kebijakan kepercayaan, ganti kebijakan JSON default dengan kebijakan yang menggunakan salah satu atau kedua kunci konteks kondisi `aws:SourceAccount` global. `aws:SourceArn` Lihat contoh kebijakan berikut:
5. Pilih Perbarui kebijakan.

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

Otentikasi dan Kontrol Akses untuk

Akses ke memerlukan kredensial. Kredensi tersebut harus memiliki izin untuk mengakses AWS sumber daya, seperti aplikasi atau instans Amazon Elastic Compute Cloud (Amazon EC2). Bagian berikut memberikan rincian tentang bagaimana Anda dapat menggunakan AWS Identity and Access Management (IAM) dan untuk membantu mengamankan akses ke sumber daya Anda.

Kontrol Akses

Anda dapat memiliki kredensial yang valid untuk melakukan autentikasi permintaan, tetapi kecuali jika Anda memiliki izin, Anda tidak dapat membuat atau mengakses sumber daya . Misalnya, Anda harus memiliki izin untuk membuat aplikasi.

Bagian berikut menggambarkan cara mengelola izin untuk . Sebaiknya Anda membaca gambaran umumnya terlebih dahulu.

- Ikhtisar Pengelolaan Izin Akses untuk Sumber Daya
- Menggunakan Kebijakan Berbasis Identitas (Kebijakan IAM) untuk
- Izin API: Tindakan, Izin, dan Referensi Sumber Daya

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) dalam AWS dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua AWS layanan dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas

yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS layanan dengan menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses AWS layanan dengan menggunakan kredensil yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apakah itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna

memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa AWS layanan, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa AWS layanan menggunakan fitur lain AWS layanan. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama AWS layanan, dikombinasikan dengan permintaan AWS layanan untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain AWS layanan atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
- Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke AWS layanan dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. AWS layanan Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Ikhtisar Pengelolaan Izin Akses untuk Sumber Daya

> **Warning**
>
> Untuk proyek baru, kami sarankan Anda menggunakan Managed Service baru untuk Apache Flink Studio over for SQL Applications. Layanan Terkelola untuk Apache Flink

Studio menggabungkan kemudahan penggunaan dengan kemampuan analitis tingkat lanjut, memungkinkan Anda membangun aplikasi pemrosesan aliran yang canggih dalam hitungan menit.

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin di Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

> **Note**
>
> Administrator akun (atau pengguna administrator) adalah pengguna dengan hak akses administrator. Untuk informasi selengkapnya, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Topik

- Sumber Daya dan Operasi
- Memahami Kepemilikan Sumber Daya
- Mengelola Akses ke Sumber Daya
- Menentukan Elemen Kebijakan: Tindakan, Pengaruh, dan Prinsipal
- Menentukan Syarat dalam Kebijakan

Sumber Daya dan Operasi

Di, sumber daya utama adalah aplikasi. Dalam kebijakan, Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang mengikuti kebijakan tersebut.

Sumber daya ini memiliki Amazon Resource Name (ARN) unik yang terkait dengannya, seperti yang ditunjukkan di tabel berikut.

Tipe Sumber Daya	Format ARN
Aplikasi	`arn:aws:kinesisanalytics:` *`region`*`:`*`account-id`*`:application/` *`application-name`*

menyediakan satu set operasi untuk bekerja dengan sumber daya. Untuk daftar operasi yang tersedia, lihat Tindakan.

Memahami Kepemilikan Sumber Daya

Akun AWS Memiliki sumber daya yang dibuat di akun, terlepas dari siapa yang menciptakan sumber daya. Secara khusus, pemilik sumber daya adalah entitas utama (yaitu, akun root, pengguna, atau peran IAM) yang mengautentikasi permintaan pembuatan sumber daya. Akun AWS Contoh berikut menggambarkan cara kerjanya:

- Jika Anda menggunakan kredensi akun root Anda Akun AWS untuk membuat aplikasi, Anda Akun AWS adalah pemilik sumber daya. (Dalam, sumber daya adalah aplikasi.)
- Jika Anda membuat pengguna di dalam Akun AWS dan memberikan izin untuk membuat aplikasi kepada pengguna tersebut, pengguna dapat membuat aplikasi. Namun, milik Anda Akun AWS, yang menjadi milik pengguna, memiliki sumber daya aplikasi. Kami sangat menyarankan Anda memberikan izin untuk peran dan bukan pengguna.
- Jika Anda membuat peran IAM Akun AWS dengan izin untuk membuat aplikasi, siapa pun yang dapat mengambil peran tersebut dapat membuat aplikasi. Anda Akun AWS, tempat pengguna berada, memiliki sumber daya aplikasi.

Mengelola Akses ke Sumber Daya

Kebijakan izin menjelaskan siapa yang memiliki akses ke suatu objek. Bagian berikut menjelaskan pilihan yang tersedia untuk membuat kebijakan izin.

> **Note**
>
> Bagian ini membahas penggunaan IAM dalam konteks . Bagian ini tidak memberikan informasi terperinci tentang layanan IAM. Untuk dokumentasi IAM lengkap, lihat Apa yang Dimaksud dengan IAM? dalam Panduan Pengguna IAM. Untuk informasi tentang sintaksis dan penjelasan kebijakan IAM, lihat Referensi Kebijakan IAM JSON dalam Panduan Pengguna IAM.

Kebijakan yang terlampir pada identitas IAM disebut sebagai kebijakan berbasis identitas (kebijakan IAM). Kebijakan yang dilampirkan pada sumber daya disebut sebagai kebijakan berbasis sumber daya. hanya mendukung kebijakan berbasis identitas (kebijakan IAM).

Topik

- Kebijakan Berbasis Identitas (Kebijakan IAM)
- Kebijakan Berbasis Sumber Daya

Kebijakan Berbasis Identitas (Kebijakan IAM)

Anda dapat melampirkan kebijakan ke identitas IAM Anda. Misalnya, Anda dapat melakukan hal berikut:

- Lampirkan kebijakan izin ke pengguna atau grup di akun Anda — Untuk memberikan izin pengguna untuk membuat sumber daya, seperti aplikasi, Anda dapat melampirkan kebijakan izin ke pengguna atau grup tempat pengguna tersebut berada.
- Melampirkan kebijakan izin pada peran (memberikan izin lintas akun) – Anda dapat melampirkan kebijakan izin berbasis identitas ke peran IAM untuk memberikan izin lintas akun. Misalnya, administrator di akun A dapat membuat peran untuk memberikan izin lintas akun ke akun lain Akun AWS (misalnya, akun B) atau layanan Amazon sebagai berikut:
  1. Administrator akun A membuat IAM role dan melampirkan kebijakan izin untuk peran yang memberikan izin pada sumber daya di akun A.
  2. Administrator akun A melampirkan kebijakan kepercayaan pada akun identifikasi peran B sebagai penanggung jawab yang dapat menjalankan peran tersebut.
  3. Administrator akun B kemudian dapat mendelegasikan izin untuk mengasumsikan peran ke pengguna dalam akun B. Dengan melakukannya, pengguna dalam akun B dapat membuat atau mengakses sumber daya di akun A. Prinsip dalam kebijakan kepercayaan juga dapat menjadi

prinsip layanan Amazon jika Anda ingin memberikan izin layanan Amazon untuk menjalankan peran tersebut.

Untuk informasi selengkapnya tentang penggunaan IAM untuk mendelegasikan izin, lihat Manajemen Akses dalam Panduan Pengguna IAM.

Berikut ini adalah contoh kebijakan yang memberikan izin untuk `kinesisanalytics:CreateApplication` tindakan, yang diperlukan untuk membuat aplikasi.

> **Note**
>
> Ini adalah contoh kebijakan pengantar. Saat Anda melampirkan kebijakan ke pengguna, pengguna akan dapat membuat aplikasi menggunakan AWS CLI atau AWS SDK. Namun, pengguna akan membutuhkan lebih banyak izin untuk mengonfigurasi input dan output. Selain itu, pengguna akan membutuhkan lebih banyak izin saat menggunakan konsol. Bagian selanjutnya memberikan informasi selengkapnya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang menggunakan kebijakan berbasis identitas dengan, lihat. Menggunakan Kebijakan Berbasis Identitas (Kebijakan IAM) untuk Untuk informasi lebih lanjut tentang pengguna, grup, peran, dan izin, lihat Identitas (Pengguna, Grup, dan Peran) dalam Panduan Pengguna IAM.

Kebijakan Berbasis Sumber Daya

Layanan lain, seperti Amazon S3, juga mendukung kebijakan izin berbasis sumber daya. Misalnya, Anda dapat melampirkan kebijakan ke bucket S3 untuk mengelola izin akses ke bucket tersebut. tidak mendukung kebijakan berbasis sumber daya.

Menentukan Elemen Kebijakan: Tindakan, Pengaruh, dan Prinsipal

Untuk setiap sumber daya , layanan menentukan serangkaian operasi API. Untuk memberikan izin bagi operasi API ini, menentukan serangkaian tindakan yang dapat Anda tentukan dalam kebijakan. Beberapa operasi API dapat memerlukan izin untuk lebih dari satu tindakan untuk melakukan operasi API. Untuk informasi selengkapnya tentang sumber daya dan operasi API, lihat Sumber Daya dan Operasi dan Tindakan.

Berikut adalah elemen-elemen kebijakan yang paling dasar:

- Sumber Daya – Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang diberlakukan oleh kebijakan tersebut. Untuk informasi selengkapnya, lihat Sumber Daya dan Operasi.
- Tindakan – Anda menggunakan kata kunci tindakan untuk mengidentifikasi operasi sumber daya yang ingin Anda izinkan atau tolak. Misalnya, Anda dapat menggunakan `create` untuk mengizinkan pengguna membuat aplikasi.
- Pengaruh – Anda menetapkan pengaruh, baik memperbolehkan atau menolak, ketika pengguna meminta tindakan tertentu. Jika Anda tidak secara eksplisit memberikan akses ke (mengizinkan) sumber daya, akses akan ditolak secara implisit. Anda juga dapat secara eksplisit menolak akses ke sumber daya, yang mungkin Anda lakukan untuk memastikan bahwa pengguna tidak dapat mengaksesnya, meskipun kebijakan yang berbeda memberikan akses.
- Prinsipal – Dalam kebijakan berbasis identitas (kebijakan IAM), pengguna yang kebijakannya terlampir adalah prinsipal implisit. Untuk kebijakan berbasis sumber daya, Anda menentukan pengguna, akun, layanan, atau entitas lain yang ingin Anda terima izinnya (berlaku hanya untuk kebijakan berbasis-sumber daya) tidak mendukung kebijakan berbasis-sumber daya.

Untuk mem-pelajari selengkapnya tentang sintaksis dan penjelasan kebijakan IAM, lihat Referensi Kebijakan JSON IAM dalam Panduan Pengguna IAM.

Untuk daftar yang menampilkan semua operasi API dan sumber daya yang diterapkan, lihat Izin API: Tindakan, Izin, dan Referensi Sumber Daya.

Menentukan Syarat dalam Kebijakan

Ketika Anda memberikan izin, Anda dapat menggunakan bahasa kebijakan akses untuk menentukan syarat kapan kebijakan akan berlaku. Misalnya, Anda mungkin ingin kebijakan diterapkan hanya setelah tanggal tertentu. Untuk informasi selengkapnya tentang menentukan kondisi dalam bahasa kebijakan, lihat Kondisi dalam Panduan Pengguna IAM.

Untuk menyatakan syarat, Anda menggunakan kunci kondisi yang telah ditentukan sebelumnya. Tidak ada kunci syarat khusus untuk . Namun, ada tombol kondisi AWS-wide yang dapat Anda gunakan sesuai kebutuhan. Untuk daftar lengkap tombol AWS-wide, lihat Kunci yang Tersedia untuk Ketentuan di Panduan Pengguna IAM.

Menggunakan Kebijakan Berbasis Identitas (Kebijakan IAM) untuk

Berikut ini adalah contoh kebijakan berbasis identitas yang menunjukkan bagaimana administrator akun dapat melampirkan kebijakan izin ke identitas IAM (yaitu, pengguna, grup, dan peran) dan memberikan izin untuk melakukan operasi pada sumber daya.

> **Important**
>
> Kami menyarankan Anda untuk terlebih dahulu meninjau topik pendahuluan yang menjelaskan konsep dasar dan pilihan yang tersedia untuk mengelola akses ke sumber daya. Untuk informasi selengkapnya, lihat Ikhtisar Pengelolaan Izin Akses untuk Sumber Daya.

Topik

- Izin yang Diperlukan untuk Menggunakan Konsol
- Kebijakan Amazon-Managed (Predefined) untuk
- Contoh Kebijakan yang Dikelola Pelanggan

Berikut adalah contoh kebijakan izin.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
```

```
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Kebijakan ini memiliki satu pernyataan:

- Pernyataan pertama memberikan izin untuk satu tindakan (`kinesisanalytics:CreateApplication`) pada sumber daya menggunakan Amazon Resource Name (ARN) untuk aplikasi. ARN dalam hal ini menentukan karakter wildcard (*) untuk menunjukkan izin diberikan untuk sumber daya apa pun.

Untuk tabel yang menunjukkan semua operasi API dan sumber daya yang diterapkan, lihat Izin API: Tindakan, Izin, dan Referensi Sumber Daya.

Izin yang Diperlukan untuk Menggunakan Konsol

Agar pengguna dapat bekerja dengan konsol, Anda harus memberikan izin yang diperlukan. Sebagai contoh, jika Anda ingin pengguna memiliki izin untuk membuat aplikasi, berikan izin yang menampilkan mereka sumber streaming di akun sehingga pengguna dapat mengonfigurasi input dan output pada konsol.

Sebaiknya lakukan hal berikut:

- Gunakan kebijakan yang dikelola Amazon untuk memberikan izin pengguna. Untuk kebijakan yang tersedia, lihat Kebijakan Amazon-Managed (Predefined) untuk .
- Buat kebijakan khusus. Dalam kasus ini, sebaiknya tinjau contoh yang diberikan dalam bagian ini. Untuk informasi selengkapnya, lihat Contoh Kebijakan yang Dikelola Pelanggan.

Kebijakan Amazon-Managed (Predefined) untuk

AWS mengatasi banyak kasus penggunaan umum dengan menyediakan kebijakan IAM mandiri yang dibuat dan dikelola oleh. AWS Kebijakan yang dikelola Amazon ini memberikan izin yang diperlukan

untuk kasus penggunaan umum sehingga Anda dapat menghindari keharusan untuk menyelidiki izin apa yang diperlukan. Untuk informasi selengkapnya, lihat Kebijakan yang Dikelola Amazon dalam Panduan Pengguna IAM.

Kebijakan yang dikelola Amazon berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, khusus untuk:

- **`AmazonKinesisAnalyticsReadOnly`**— Memberikan izin untuk tindakan yang memungkinkan pengguna untuk membuat daftar aplikasi dan meninjau konfigurasi input/output. Ini juga memberikan izin yang memungkinkan pengguna untuk melihat daftar aliran Kinesis dan aliran pengiriman Firehose. Saat aplikasi berjalan, pengguna dapat melihat data sumber dan hasil analisis waktu nyata di konsol.

- **`AmazonKinesisAnalyticsFullAccess`**— Memberikan izin untuk semua tindakan dan semua izin lain yang memungkinkan pengguna untuk membuat dan mengelola aplikasi. Namun, perhatikan hal berikut:

  - Izin ini tidak cukup jika pengguna ingin membuat IAM role baru di konsol (izin ini memungkinkan pengguna memilih peran yang ada). Jika Anda ingin pengguna dapat membuat IAM role di konsol, tambahkan kebijakan yang dikelola Amazon `IAMFullAccess`.

  - Seorang pengguna harus memiliki izin untuk `iam:PassRole` tindakan untuk menentukan peran IAM saat mengkonfigurasi aplikasi. Kebijakan yang dikelola Amazon ini memberikan izin untuk tindakan `iam:PassRole` kepada pengguna hanya pada IAM role yang dimulai dengan prefiks `service-role/kinesis-analytics`.

    Jika pengguna ingin mengonfigurasi aplikasi dengan peran yang tidak memiliki awalan ini, pertama-tama Anda harus secara eksplisit memberikan izin kepada pengguna untuk `iam:PassRole` tindakan pada peran tertentu.

Anda juga dapat membuat kebijakan IAM khusus Anda sendiri untuk memberikan izin untuk tindakan dan sumber daya. Anda dapat menyematkan kebijakan khusus ini untuk pengguna atau grup yang memerlukan izin tersebut.

Contoh Kebijakan yang Dikelola Pelanggan

Contoh dalam bagian ini menyediakan sekelompok contoh kebijakan yang dapat Anda lampirkan ke pengguna. Jika Anda baru membuat kebijakan, kami sarankan Anda terlebih dahulu membuat pengguna di akun Anda. Kemudian lampirkan kebijakan ke pengguna secara berurutan, seperti yang diuraikan dalam langkah-langkah di bagian ini. Anda selanjutnya dapat menggunakan konsol untuk memverifikasi efek setiap kebijakan saat Anda melampirkan kebijakan kepada pengguna.

Awalnya, pengguna tidak memiliki izin dan tidak dapat melakukan apa pun di konsol. Ketika Anda melampirkan kebijakan ke pengguna, Anda dapat memverifikasi pengguna dapat melakukan berbagai tindakan pada konsol.

Sebaiknya gunakan dua jendela browser. Di satu jendela, buat pengguna dan berikan izin. Di sisi lain, masuk ke AWS Management Console menggunakan kredensil pengguna dan verifikasi izin saat Anda memberikannya.

Untuk contoh yang menunjukkan cara membuat peran IAM yang dapat Anda gunakan sebagai peran eksekusi untuk aplikasi Anda, lihat Membuat Peran IAM di Panduan Pengguna IAM.

Contoh langkah-langkah

- Langkah 1: Buat Pengguna IAM
- Langkah 2: Izinkan Izin Pengguna untuk Tindakan yang Tidak Spesifik
- Langkah 3: Izinkan Pengguna Melihat Daftar Aplikasi dan Melihat Detail
- Langkah 4: Izinkan Pengguna Memulai Aplikasi Tertentu
- Langkah 5: Izinkan Pengguna Membuat Aplikasi
- Langkah 6: Izinkan Aplikasi Menggunakan Prapemrosesan Lambda

Langkah 1: Buat Pengguna IAM

Pertama, Anda perlu membuat pengguna, menambahkan pengguna ke grup IAM dengan izin administratif, dan kemudian memberikan izin administratif kepada pengguna yang Anda buat. Anda kemudian dapat mengakses AWS menggunakan URL khusus dan kredensil pengguna tersebut.

Untuk melihat instruksi, buka Membuat Grup Pengguna dan Administrator IAM Pertama Anda di Panduan Pengguna IAM.

Langkah 2: Izinkan Izin Pengguna untuk Tindakan yang Tidak Spesifik

Pertama, berikan izin pengguna untuk semua tindakan yang tidak spesifik untuk yang dibutuhkan pengguna saat bekerja dengan aplikasi. Ini termasuk izin untuk bekerja dengan aliran (tindakan Amazon Kinesis Data Streams, tindakan Amazon Data Firehose), dan izin untuk tindakan. CloudWatch Lampirkan kebijakan berikut ke pengguna.

Anda harus memperbarui kebijakan dengan memberi nama IAM role yang Anda ingin beri izin `iam:PassRole`, atau tentukan karakter wildcard (*) yang menunjukkan semua IAM role. Ini bukan praktik yang aman; namun, Anda mungkin tidak memiliki IAM role tertentu yang dibuat selama pengujian ini.

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

Langkah 3: Izinkan Pengguna Melihat Daftar Aplikasi dan Melihat Detail

Kebijakan berikut memberi pengguna izin berikut:

- Izin untuk tindakan `kinesisanalytics:ListApplications` sehingga pengguna dapat melihat daftar aplikasi. Ini adalah panggilan API tingkat layanan, dan oleh karena itu, Anda menentukan "*" sebagai nilai `Resource`.
- Izin untuk tindakan `kinesisanalytics:DescribeApplication` sehingga Anda bisa mendapatkan informasi tentang salah satu aplikasi.

Tambahkan kebijakan ini ke pengguna.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:ListApplications"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:DescribeApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/*"
        }
    ]
}
```

Verifikasi izin ini dengan masuk ke konsol menggunakan kredensil pengguna.

Langkah 4: Izinkan Pengguna Memulai Aplikasi Tertentu

Jika Anda ingin pengguna dapat memulai salah satu aplikasi yang ada, lampirkan kebijakan berikut kepada pengguna. Kebijakan ini menyediakan izin untuk tindakan `kinesisanalytics:StartApplication`. Anda harus memperbarui kebijakan dengan memberikan ID akun, AWS Wilayah, dan nama aplikasi Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
                {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

Langkah 5: Izinkan Pengguna Membuat Aplikasi

Jika Anda ingin pengguna membuat aplikasi, Anda kemudian dapat melampirkan kebijakan berikut kepada pengguna. Anda harus memperbarui kebijakan dan memberikan AWS Wilayah, ID akun Anda, dan nama aplikasi tertentu yang Anda ingin pengguna buat, atau “*” sehingga pengguna dapat menentukan nama aplikasi apa pun (dan dengan demikian membuat beberapa aplikasi).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication",
                "kinesisanalytics:UpdateApplication",
                "kinesisanalytics:AddApplicationInput",
                "kinesisanalytics:AddApplicationOutput"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

Langkah 6: Izinkan Aplikasi Menggunakan Prapemrosesan Lambda

Jika Anda ingin aplikasi dapat menggunakan prapemrosesan Lambda, lampirkan kebijakan berikut ke peran tersebut.

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

Izin API: Tindakan, Izin, dan Referensi Sumber Daya

Saat Anda mengatur Kontrol Akses dan menulis kebijakan izin yang dapat Anda lampirkan ke identitas IAM (kebijakan berbasis identitas), Anda dapat menggunakan daftar sebagai referensi. Daftar mencakup setiap operasi API, tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan, dan AWS sumber daya yang dapat Anda berikan izin. Anda menentukan tindakan dalam bidang `Action` kebijakan, dan Anda menentukan nilai sumber daya pada bidang `Resource` kebijakan.

Anda dapat menggunakan kunci kondisi AWS-wide dalam kebijakan Anda untuk menyatakan kondisi. Untuk daftar lengkap tombol AWS-wide, lihat Kunci yang Tersedia di Panduan Pengguna IAM.

> **Note**
>
> Untuk menentukan tindakan, gunakan awalan `kinesisanalytics` diikuti dengan nama operasi API (misalnya, `kinesisanalytics:AddApplicationInput`).

API dan Izin yang Diperlukan untuk Tindakan

Operasi API:

Izin yang Diperlukan (Tindakan API):

Sumber Daya:

API dan Izin yang Diperlukan untuk Tindakan

Amazon RDS API dan Izin yang Diperlukan untuk Tindakan

Operasi API:AddApplicationInput

Tindakan: `kinesisanalytics:AddApplicationInput`

Sumber Daya:

`arn:aws:kinesisanalytics:` *`region`*`:`*`accountId`*`:application/`*`application-name`*

GetApplicationState

Konsol menggunakan metode internal yang disebut `GetApplicationState` untuk membuat sampel atau mengakses data aplikasi. Aplikasi layanan Anda harus memiliki izin untuk `kinesisanalytics:GetApplicationState` API internal untuk mengambil sampel atau mengakses data aplikasi melalui. AWS Management Console

Pemantauan

menyediakan fungsionalitas pemantauan untuk aplikasi Anda. Untuk informasi selengkapnya, lihat Pemantauan.

Validasi Kepatuhan untuk Amazon Kinesis Data Analytics untuk Aplikasi SQL

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Kinesis Data Analytics sebagai bagian dari AWS beberapa program kepatuhan. Hal ini mencakup SOC, PCI, HIPAA, dan lainnya.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat Layanan Amazon dalam Lingkup menurut Program Kepatuhan. Untuk informasi umum, lihat Program Kepatuhan AWS.

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Kinesis Data Analytics ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta undang-undang dan peraturan yang berlaku. Jika penggunaan Kinesis Data Analytics Anda tunduk pada kepatuhan standar seperti HIPAA atau PCI, AWS menyediakan sumber daya untuk membantu:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- Arsitektur untuk Whitepaper Keamanan dan Kepatuhan HIPAA — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi yang sesuai dengan HIPAA. AWS
- AWS Sumber Daya Kepatuhan — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.

- AWS Config AWS Layanan ini menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di Amazon Kinesis Data Analytics

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Selain infrastruktur AWS global, Kinesis Data Analytics menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

Pemulihan Bencana

Kinesis Data Analytics berjalan dalam mode nirserver, dan menangani degradasi host, ketersediaan Availability Zone, dan masalah terkait infrastruktur lainnya dengan melakukan migrasi otomatis. Jika hal ini terjadi, Kinesis Data Analytics memastikan aplikasi diproses tanpa kehilangan data apa pun. Untuk informasi selengkapnya, lihat Model Pengiriman untuk Menyimpan Output Aplikasi untuk Tujuan Eksternal.

Keamanan Infrastruktur di Kinesis Data Analytics untuk Aplikasi SQL

Sebagai layanan terkelola, Amazon Kinesis Data Analytics dilindungi oleh AWS prosedur keamanan jaringan global yang dijelaskan dalam whitepaper Amazon Web Services: Tinjauan Proses Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Kinesis Data Analytics melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Praktik Terbaik Keamanan untuk Kinesis Data Analytics

Amazon Kinesis Data Analytics menyediakan sejumlah fitur keamanan untuk dipertimbangkan ketika Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Gunakan IAM role untuk mengakses layanan Amazon lainnya

Aplikasi Kinesis Data Analytics Anda harus memiliki kredenal yang valid untuk mengakses sumber daya di layanan lain, seperti aliran data Kinesis, aliran pengiriman Firehose, atau bucket Amazon S3. Anda tidak boleh menyimpan AWS kredensil secara langsung di aplikasi atau di ember Amazon S3. Ini adalah kredensial jangka panjang yang tidak dirotasi secara otomatis dan dapat menimbulkan dampak bisnis yang signifikan jika dibobol.

Sebaliknya, Anda harus menggunakan IAM role untuk mengelola kredensial sementara untuk aplikasi guna mengakses sumber daya lainnya. Ketika Anda menggunakan peran, Anda tidak perlu menggunakan kredensi jangka panjang untuk mengakses sumber daya lain.

Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna IAM:

- Peran IAM
- Skenario Umum untuk Peran: Pengguna, Aplikasi, dan Layanan

Terapkan Enkripsi Sisi Server di Sumber Daya Dependen

Data at rest dan data dalam transit dienkripsi di Kinesis Data Analytics, dan enkripsi ini tidak dapat dinonaktifkan. Anda harus menerapkan enkripsi sisi server di sumber daya dependen Anda, seperti aliran data Kinesis, aliran pengiriman Firehose, dan bucket Amazon S3. Untuk informasi selengkapnya tentang menerapkan enkripsi sisi server dalam sumber daya dependen, lihat Perlindungan Data.

Gunakan CloudTrail untuk Memantau Panggilan API

Kinesis Data Analytics terintegrasi AWS CloudTrail dengan, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan Amazon di Kinesis Data Analytics.

Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Kinesis Data Analytics, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk informasi selengkapnya, lihat the section called “Menggunakan AWS CloudTrail”.

Monitoring untuk Aplikasi SQL

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja dan aplikasi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multipoint jika terjadi. Namun, sebelum Anda mulai memantau, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah berikutnya adalah menetapkan dasar untuk performa normal di lingkungan Anda, dengan mengukur kinerja di berbagai waktu dan dengan kondisi beban yang berbeda. Saat Anda memantau, Anda dapat menyimpan data pemantauan historis. Dengan cara ini, Anda dapat membandingkannya dengan data performa saat ini, mengidentifikasi pola performa normal dan anomali performa, serta merancang metode untuk mengatasi masalah.

Dengan, Anda memantau aplikasi. Aplikasi memproses aliran data (input atau output), keduanya termasuk pengidentifikasi yang dapat Anda gunakan untuk mempersempit pencarian Anda pada CloudWatch log. Untuk informasi tentang cara memproses aliran data, lihatAmazon Kinesis Data Analytics untuk Aplikasi SQL: Cara Kerjanya.

Metrik yang paling penting adalah `millisBehindLatest`, yang menunjukkan seberapa jauh aplikasi membaca dari sumber streaming. Dalam kasus tertentu, milidetik di belakang harus di atau mendekati nol. Hal ini umum terjadi pada lonjakan singkat, yang muncul sebagai peningkatan `millisBehindLatest`.

Kami menyarankan Anda mengatur CloudWatch alarm yang memicu ketika aplikasi tertinggal lebih dari satu jam membaca sumber streaming. Untuk beberapa kasus penggunaan yang memerlukan pemrosesan waktu nyata yang sangat dekat, seperti memancarkan data yang diproses ke aplikasi langsung, Anda dapat memilih untuk mengatur alarm dengan nilai yang lebih rendah, misalnya lima menit.

Topik

- Alat Pemantauan
- Pemantauan CloudWatch dengan Amazon
- Pencatatan Panggilan API dengan AWS CloudTrail

Alat Pemantauan

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Alat Pemantauan Otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk melihat dan melaporkan saat terjadi kesalahan:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pemantauan CloudWatch dengan Amazon.
- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Untuk informasi selengkapnya, lihat Memantau File Log di Panduan CloudWatch Pengguna Amazon.
- CloudWatch Acara Amazon — Cocokkan acara dan arahkan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi selengkapnya, lihat Apa itu CloudWatch Acara Amazon di Panduan CloudWatch Pengguna Amazon.
- AWS CloudTrail Pemantauan Log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat Bekerja dengan File CloudTrail Log di PanduanAWS CloudTrail Pengguna.

Alat Pemantauan Manual

Bagian penting lainnya dari pemantauan melibatkan pemantauan secara manual item-item yang tidak tercakup oleh CloudWatch alarm. AWS Management Console Dasbor CloudWatch Trusted Advisor,,, dan lainnya memberikan at-a-glance pandangan tentang keadaan AWS lingkungan Anda.

- CloudWatch Halaman beranda menunjukkan yang berikut:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Membuat dasbor yang disesuaikan untuk memantau layanan yang penting bagi Anda
  - Data metrik grafik untuk memecahkan masalah dan mengungkap tren
  - Cari dan telusuri semua metrik Anda
  - Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah
- AWS Trusted Advisor dapat membantu Anda memantau untuk meningkatkan kinerja, keandalan, keamanan, dan efektivitas biaya. Empat pemeriksaan Trusted Advisor tersedia untuk semua pengguna. Lebih dari 50 pemeriksaan tersedia untuk pengguna dengan paket support Business atau Korporasi. Untuk informasi selengkapnya, lihat AWS Trusted Advisor.

Pemantauan CloudWatch dengan Amazon

Anda dapat memantau aplikasi menggunakan Amazon CloudWatch. CloudWatch mengumpulkan dan memproses data mentah dari metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini disimpan selama dua minggu. Artinya Anda dapat mengakses informasi historis untuk mendapatkan perspektif yang lebih baik tentang performa aplikasi web atau layanan Anda. Secara default, data metrik secara otomatis dikirim ke CloudWatch. Untuk informasi lebih lanjut, lihat Apa itu Amazon CloudWatch? di Panduan CloudWatch Pengguna Amazon.

Topik

- Metrik dan Dimensi
- Melihat Metrik dan Dimensi
- Membuat CloudWatch Alarm untuk Memantau
- Bekerja dengan Amazon CloudWatch Logs

Metrik dan Dimensi

Namespace `AWS/KinesisAnalytics` mencakup metrik berikut.

Metrik	Deskripsi
`Bytes`	Jumlah byte yang dibaca (per aliran input) atau yang ditulis (per aliran output). Tingkat: Per aliran input dan per aliran output
`KPUs`	Jumlah Unit Pemrosesan Kinesis yang digunakan untuk menjalankan aplikasi pemrosesan aliran Anda. Jumlah rata-rata KPU yang digunakan setiap jam menentukan penagihan untuk aplikasi Anda. Tingkat: Tingkat aplikasi
`MillisBehindLatest`	Menunjukkan seberapa tertinggal dari waktu saat ini aplikasi membaca dari sumber streaming. Tingkat: Tingkat aplikasi
`Records`	Jumlah catatan yang dibaca (per aliran input) atau yang ditulis (per aliran output). Tingkat: Per aliran input dan per aliran output
`Success`	1 untuk setiap upaya pengiriman yang berhasil ke tujuan yang dikonfigurasi untuk aplikasi Anda; 0 untuk setiap upaya pengiriman yang gagal. Nilai rata-rata metrik ini menunjukkan berapa banyak pengiriman yang berhasil dilakukan. Tingkat: Per tujuan.
`InputProcessing.Duration`	Waktu yang dibutuhkan untuk setiap pemanggilan AWS Lambda fungsi dilakukan oleh. Tingkat: Per aliran input

Metrik	Deskripsi
`InputProcessing.OkRecords`	Jumlah catatan yang dikembalikan oleh fungsi Lambda yang ditandai dengan status `Ok`. Tingkat: Per aliran input
`InputProcessing.OkBytes`	Jumlah byte catatan yang dikembalikan oleh fungsi Lambda yang ditandai dengan status `Ok`. Tingkat: Per aliran input
`InputProcessing.DroppedRecords`	Jumlah catatan yang dikembalikan oleh fungsi Lambda yang ditandai dengan status `Dropped`. Tingkat: Per aliran input
`InputProcessing.ProcessingFailedRecords`	Jumlah catatan yang dikembalikan oleh fungsi Lambda yang ditandai dengan status `ProcessingFailed` . Tingkat: Per aliran input
`InputProcessing.Success`	Jumlah doa Lambda yang berhasil oleh. Tingkat: Per aliran input
`LambdaDelivery.OkRecords`	Jumlah catatan yang dikembalikan oleh fungsi Lambda yang ditandai dengan status `Ok`. Tingkat: Per tujuan Lambda
`LambdaDelivery.DeliveryFailedRecords`	Jumlah catatan yang dikembalikan oleh fungsi Lambda yang ditandai dengan status `DeliveryFailed` . Tingkat: Per tujuan Lambda

Metrik	Deskripsi
`LambdaDelivery.Duration`	Waktu yang dibutuhkan untuk setiap pemanggilan fungsi Lambda dilakukan oleh. Tingkat: Per tujuan Lambda

menyediakan metrik untuk dimensi berikut.

Dimensi	Deskripsi
`Flow`	Per aliran input: Input Per aliran output: Output
`Id`	Per aliran input: Id Input Per aliran output: Id Output

Melihat Metrik dan Dimensi

Saat aplikasi Anda memproses aliran data, kirimkan metrik dan dimensi berikut ke file. CloudWatch Anda dapat menggunakan prosedur berikut untuk melihat metrik.

Di konsol, metrik dikelompokkan terlebih dahulu berdasarkan namespace layanan, lalu berdasarkan kombinasi dimensi dalam setiap namespace.

Untuk melihat metrik menggunakan konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Metrik.
3. Di panel CloudWatch Metrik menurut Kategori untuk, pilih kategori metrik.
4. Di panel atas, gulir untuk melihat daftar lengkap metrik.

Untuk melihat metrik menggunakan AWS CLI

- Pada prompt perintah, gunakan perintah berikut.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

metrik dikumpulkan pada tingkat berikut:

- Aplikasi
- Aliran input
- Aliran output

Membuat CloudWatch Alarm untuk Memantau

Anda dapat membuat CloudWatch alarm Amazon yang mengirimkan pesan Amazon SNS saat alarm berubah status. Alarm mengawasi satu metrik selama suatu periode waktu yang Anda tentukan. Alarm tersebut melakukan satu atau beberapa tindakan berdasarkan nilai metrik yang relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakan ini adalah notifikasi yang dikirim ke topik Amazon SNS atau kebijakan Penskalaan Otomatis.

Alarm memicu tindakan hanya untuk status dengan perubahan berkelanjutan saja. Agar CloudWatch alarm dapat memanggil suatu tindakan, status harus telah berubah dan dipertahankan untuk jangka waktu tertentu.

Anda dapat mengatur alarm menggunakan AWS Management Console,, atau CloudWatch API CloudWatch AWS CLI, seperti yang dijelaskan berikut.

Untuk mengatur alarm menggunakan CloudWatch konsol

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Buat Alarm. Wizard Buat Alarm dimulai.
3. Pilih Kinesis Analytics Metrics (Metrik Analitik Kinesis). Kemudian gulir metrik untuk menemukan metrik yang ingin Anda pasang alarm.

   Untuk menampilkan metrik saja, cari ID sistem file dari sistem file Anda. Pilih metrik untuk membuat alarm, lalu pilih Next (Selanjutnya).
4. Masukkan Name (Nama), Description (Deskripsi), dan Whenever (Kapan Pun) untuk metrik.

5. Jika Anda CloudWatch ingin mengirimi Anda email saat status alarm tercapai, di bidang Setiap kali alarm ini:, pilih Status adalah ALARM. Di bidang Send notification to: (Kirim notifikasi ke:), pilih topik SNS yang ada. Jika Anda memilih Create topic (Buat topik), Anda dapat mengatur nama dan alamat email untuk daftar langganan email baru. Daftar ini disimpan dan muncul dalam bidang isian untuk alarm selanjutnya.

   > **Note**
   >
   > Jika Anda menggunakan Buat topik untuk membuat topik Amazon SNS baru, alamat email harus diverifikasi sebelum menerima pemberitahuan. Email hanya dikirimkan saat alarm berada dalam status alarm. Jika perubahan status alarm ini terjadi sebelum alamat email diverifikasi, alamat tidak menerima notifikasi.

6. Di bagian Alarm Preview (Pratinjau Alarm), lakukan pratinjau alarm yang akan Anda buat.
7. Pilih Create Alarm (Buat Alarm) untuk membuat alarm.

Untuk mengatur alarm menggunakan CloudWatch CLI

- Panggil `mon-put-metric-alarm`. Untuk informasi selengkapnya, lihat Referensi Amazon CloudWatch CLI.

Untuk mengatur alarm menggunakan CloudWatch API

- Panggil `PutMetricAlarm`. Untuk informasi selengkapnya, lihat Referensi Amazon CloudWatch API.

Bekerja dengan Amazon CloudWatch Logs

Jika aplikasi salah dikonfigurasi, aplikasi dapat beralih ke keadaan berjalan selama aplikasi mulai. Atau aplikasi dapat memperbarui, tetapi tidak memproses data apa pun ke aliran input dalam aplikasi. Dengan menambahkan opsi CloudWatch log ke aplikasi, Anda dapat memantau masalah konfigurasi aplikasi.

dapat menghasilkan kesalahan konfigurasi dalam kondisi berikut:

- Kinesis data stream yang digunakan untuk input tidak ada.
- Aliran pengiriman Amazon Data Firehose yang digunakan untuk input tidak ada.

- Bucket Amazon S3 yang digunakan sebagai sumber data referensi tidak ada.
- File yang ditentukan dalam sumber data referensi dalam bucket S3 tidak ada.
- Sumber daya yang benar tidak ditentukan dalam peran AWS Identity and Access Management (IAM) yang mengelola izin terkait.
- Izin yang benar tidak didefinisikan dalam IAM role yang mengelola izin terkait.
- tidak memiliki izin untuk mengambil peran IAM yang mengelola izin terkait.

Untuk informasi selengkapnya tentang Amazon CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

Menambahkan Tindakan PutLogEvents Kebijakan

membutuhkan izin untuk menulis kesalahan konfigurasi ke. CloudWatch Anda dapat menambahkan izin ini ke peran IAM yang mengasumsikan, seperti yang dijelaskan berikut. Untuk informasi selengkapnya tentang penggunaan peran IAM, lihatIdentity and Access Management di Kinesis Data Analytics.

Kebijakan Kepercayaan

Untuk memberikan izin untuk mengambil peran IAM, Anda dapat melampirkan kebijakan kepercayaan berikut ke peran tersebut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Kebijakan Izin

Untuk memberikan izin aplikasi untuk menulis peristiwa log CloudWatch dari sumber daya, Anda dapat menggunakan kebijakan izin IAM berikut.

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

Menambahkan Pemantauan Kesalahan Konfigurasi

Gunakan tindakan API berikut untuk menambahkan opsi CloudWatch log ke aplikasi baru atau yang sudah ada atau ubah opsi log untuk aplikasi yang sudah ada.

> **Note**
>
> Saat ini Anda hanya dapat menambahkan opsi CloudWatch log ke aplikasi dengan menggunakan tindakan API. Anda tidak dapat menambahkan opsi CloudWatch log dengan menggunakan konsol.

Menambahkan Opsi CloudWatch Log Saat Membuat Aplikasi

Contoh kode berikut menunjukkan cara menggunakan `CreateApplication` tindakan untuk menambahkan opsi CloudWatch log saat Anda membuat aplikasi. Untuk informasi selengkapnya tentang `Create_Application`, lihat 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>"
 }]
}
```

Menambahkan Opsi CloudWatch Log ke Aplikasi yang Ada

Contoh kode berikut menunjukkan cara menggunakan `AddApplicationCloudWatchLoggingOption` tindakan untuk menambahkan opsi CloudWatch log ke aplikasi yang ada. Untuk informasi selengkapnya tentang `AddApplicationCloudWatchLoggingOption`, lihat 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>
}
```

Memperbarui Opsi CloudWatch Log yang Ada

Contoh kode berikut menunjukkan bagaimana menggunakan `UpdateApplication` tindakan untuk memodifikasi opsi CloudWatch log yang ada. Untuk informasi selengkapnya tentang `UpdateApplication`, lihat 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>
}
```

Menghapus Opsi CloudWatch Log dari Aplikasi

Contoh kode berikut menunjukkan cara menggunakan `DeleteApplicationCloudWatchLoggingOption` tindakan untuk menghapus opsi CloudWatch log yang ada. Untuk informasi selengkapnya tentang `DeleteApplicationCloudWatchLoggingOption`, lihat 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>
}
```

Kesalahan Konfigurasi

Bagian berikut berisi detail tentang kesalahan yang mungkin Anda lihat di CloudWatch Log Amazon dari aplikasi yang salah konfigurasi.

Format Pesan Kesalahan

Pesan kesalahan yang dibuat oleh kesalahan konfigurasi aplikasi muncul dalam format berikut ini.

```
{
"applicationARN": "string",
"applicationVersionId": integer,
"messageType": "ERROR",
"message": "string",
"inputId": "string",
"referenceId": "string",
"errorCode": "string"
"messageSchemaVersion": "integer",
```

```
}
```

Bidang dalam pesan kesalahan berisi informasi berikut:

- `applicationARN`: Amazon Resource Name (ARN) aplikasi yang dibuat, misalnya: `arn:aws:kinesisanalytics:us-east-1:112233445566:application/sampleApp`
- `applicationVersionId`: Versi aplikasi saat kesalahan ditemukan. Untuk informasi selengkapnya, lihat ApplicationDetail.
- `messageType`: Tipe pesan. Saat ini, tipe ini hanya bisa `ERROR`.
- `message`: Detail kesalahan, misalnya:

  ```
  There is a problem related to the configuration of your input. Please check that the
   resource exists, the role has the correct permissions to access the resource and
   that Kinesis Analytics can assume the role provided.
  ```

- `inputId`: ID yang terkait dengan input aplikasi. Nilai ini hanya tersedia jika input ini adalah penyebab kesalahan. Nilai ini tidak tersedia jika `referenceId` ada. Untuk informasi selengkapnya, lihat DescribeApplication.
- `referenceId`: ID yang terkait dengan sumber data referensi aplikasi. Nilai ini hanya tersedia jika sumber ini adalah penyebab kesalahan. Nilai ini tidak tersedia jika `inputId` ada. Untuk informasi selengkapnya, lihat DescribeApplication.
- `errorCode`: Pengidentifikasi untuk kesalahan. ID ini adalah `InputError` atau `ReferenceDataError`.
- `messageSchemaVersion`: Nilai yang menentukan versi skema pesan saat ini, 1 saat ini. Anda dapat memeriksa nilai ini untuk melihat apakah skema pesan kesalahan sudah diperbarui.

Kesalahan

Kesalahan yang mungkin muncul di CloudWatch Log untuk menyertakan yang berikut ini.

Sumber Daya Tidak Ada

Jika ARN ditentukan untuk aliran input Kinesis yang tidak ada, tetapi ARN secara sintaksis benar, kesalahan seperti berikut akan dihasilkan.

```
{
"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"
}
```

Jika kunci file Amazon S3 salah digunakan untuk data referensi, kesalahan seperti berikut akan dihasilkan.

```
{
 "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"
}
```

Peran Tidak Ada

Jika ARN ditentukan untuk aliran input IAM yang tidak ada, tetapi ARN secara sintaksis benar, kesalahan seperti berikut akan dihasilkan.

```
{
 "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"
```

```
}
```

Peran Tidak Memiliki Izin untuk Mengakses Sumber Daya

Jika peran input digunakan yang tidak memiliki izin untuk mengakses sumber daya input, seperti aliran sumber Kinesis, kesalahan seperti berikut akan dihasilkan.

```
{
 "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"
}
```

Pencatatan Panggilan API dengan AWS CloudTrail

terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di. CloudTrail menangkap semua panggilan API untuk sebagai peristiwa. Panggilan yang direkam mencakup panggilan dari konsol dan panggilan kode ke operasi API ini. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Informasi di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua . Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Semua tindakan dicatat oleh CloudTrail dan didokumentasikan dalam referensi API. Misalnya, panggilan ke `CreateApplication` dan `UpdateApplication` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan Pengguna root akun AWS atau kredensi pengguna.
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Bahwa permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami Entri File Berkas Log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan AddApplicationCloudWatchLoggingOptiondan DescribeApplicationtindakan.


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

Batas

Saat bekerja dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL, perhatikan batas berikut ini:

- Kinesis Data Analytics untuk SQL tersedia di Wilayah AWS berikut: AS Timur (Ohio), AS Timur (Virginia N.), AS Barat (Oregon), Kanada (Tengah), Eropa (Paris), Eropa (Irlandia), Eropa (Frankfurt), Eropa (London), Asia Pasifik (Hong Kong), Asia Pasifik (Mumbai), Asia Pasifik (Sydney), Asia Pasifik (Singapura), Asia Pasifik (Singapura), Asia Pasifik (Seoul), Asia Pasifik (Tokyo), Amerika Selatan (Sao Paulo), (AS-Timur), (AS-Barat). AWS GovCloud AWS GovCloud Kami tidak memiliki rencana untuk meluncurkan Kinesis Data Analytics untuk SQL ke AWS Wilayah tambahan.
- Setelah 28 Juni 2023, Anda tidak akan dapat membuat Kinesis Data Analytics baru untuk aplikasi SQL menggunakan konsol manajemen jika Anda belum menggunakan AWS Kinesis Data Analytics untuk SQL. Jika Anda membuat Kinesis Data Analytics untuk aplikasi SQL sebelum 28 Juni 2023, tidak ada perubahan pada cara Anda membuat dan menjalankan aplikasi hari ini di Wilayah di mana Anda sudah AWS menggunakan Kinesis Data Analytics untuk SQL. Namun, Anda tidak lagi dapat membuat aplikasi baru menggunakan AWS Konsol di Wilayah di mana Anda tidak menggunakan Kinesis Data Analytics untuk SQL.
- Setelah 12 September 2023, Anda tidak akan dapat membuat aplikasi baru menggunakan Kinesis Data Firehose sebagai sumber jika Anda belum menggunakan Kinesis Data Analytics untuk SQL. Pelanggan lama yang menggunakan Kinesis Data Analytics untuk aplikasi SQL `KinesisFirehoseInput` dengan dapat terus menambahkan aplikasi `KinesisFirehoseInput` dengan dalam akun yang ada menggunakan Kinesis Data Analytics. Jika Anda adalah pelanggan lama dan ingin membuat akun baru dengan Kinesis Data Analytics untuk aplikasi SQL `KinesisFirehoseInput` dengan Anda dapat membuka kasus dukungan. Untuk informasi selengkapnya, lihat Pusat AWS Support.
- Ukuran baris di aliran dalam aplikasi dibatasi hingga 512 KB. Kinesis Data Analytics menggunakan hingga 1 KB untuk menyimpan metadata. Metadata ini menghitung batas baris.
- Kode SQL dalam aplikasi dibatasi hingga 100 KB.
- Jendela terpanjang yang kami sarankan untuk kueri jendela adalah satu jam. Aliran dalam aplikasi disimpan dalam penyimpanan yang mudah berubah, dan gangguan aplikasi yang tidak terduga

akan menyebabkan aplikasi membangun kembali aliran dari sumber data dalam penyimpanan yang mudah berubah.

- Throughput yang paling kami rekomendasikan untuk satu aliran dalam aplikasi adalah antara 2 dan 20 MB/detik, bergantung pada kompleksitas kueri aplikasi.

- Anda dapat membuat hingga 50 aplikasi Kinesis Data Analytics AWS per Wilayah di akun Anda. Anda dapat membuat kasus untuk meminta aplikasi tambahan melalui bentuk peningkatan batas layanan. Untuk informasi selengkapnya, lihat Pusat AWS Support.

- Throughput streaming maksimum yang dapat diproses oleh satu Kinesis Data Analytics untuk aplikasi SQL adalah sekitar 100 MB/detik. Ini mengasumsikan Anda sudah meningkatkan jumlah aliran dalam aplikasi ke nilai maksimum 64, dan Anda telah meningkatkan batas KPU melebihi 8 (lihat batas berikut untuk detailnya). Jika aplikasi Anda perlu memproses lebih dari 100 MB/detik input, lakukan salah satu dari berikut ini:
  - Gunakan beberapa Kinesis Data Analytics untuk aplikasi SQL untuk memproses input
  - Gunakan Managed Service untuk Apache Flink untuk Aplikasi Java jika Anda ingin terus menggunakan satu aliran dan aplikasi.

  > **Note**
  >
  > Kami menyarankan untuk meninjau `InputProcessing.0kBytes` metrik aplikasi Anda secara berkala sehingga Anda dapat merencanakan ke depan untuk menggunakan beberapa aplikasi SQL atau bermigrasi ke Managed Service untuk Apache Flink untuk Aplikasi Java jika throughput input yang diproyeksikan aplikasi Anda melebihi 100 MB/detik. Kami juga menyarankan untuk membuat CloudWatch alarm `InputProcessing.0kBytes` agar Anda diberi tahu saat aplikasi Anda mendekati batas throughput input. Ini dapat berguna karena Anda dapat memperbarui kueri aplikasi Anda ke tradeoff untuk throughput yang lebih tinggi, sehingga menghindari tekanan balik dan penundaan dalam analitik. Untuk informasi selengkapnya, lihat Pemecahan Masalah. Mengkhawatirkan juga dapat berguna jika Anda memiliki mekanisme untuk mengurangi throughput di hulu.

- Jumlah unit pemrosesan Kinesis (KPU) dibatasi hingga delapan. Untuk petunjuk tentang cara meminta peningkatan batas ini, lihat Untuk meminta peningkatan batas di Amazon Service Limits.

  Dengan Kinesis Data Analytics, Anda hanya membayar apa yang Anda gunakan. Anda akan ditagih tarif per jam berdasarkan jumlah rata-rata KPU yang digunakan untuk menjalankan aplikasi pemrosesan aliran Anda. Satu KPU memberi Anda 1 vCPU dan memori 4 GB.

- Setiap aplikasi dapat memiliki satu sumber streaming dan hingga satu sumber data referensi.

- Anda dapat mengonfigurasi hingga tiga tujuan untuk aplikasi Kinesis Data Analytics. Sebaiknya gunakan salah satu dari tujuan ini untuk menyimpan aliran data kesalahan dalam aplikasi.

- Ukuran objek Amazon S3 yang menyimpan data referensi bisa sampai 1 GB.

- Jika Anda mengubah data referensi yang disimpan dalam bucket S3 setelah Anda mengunggah data referensi ke tabel dalam aplikasi, Anda perlu menggunakan operasi UpdateApplication (menggunakan API atau AWS CLI) untuk me-refresh data di tabel dalam aplikasi. Saat ini, AWS Management Console tidak mendukung data referensi yang di-refresh di aplikasi Anda.

- Saat ini, Kinesis Data Analytics tidak mendukung data yang dihasilkan oleh Amazon Kinesis Producer Library (KPL).

- Anda dapat menetapkan hingga 50 tanda per aplikasi.

Praktik Terbaik

Bagian ini menjelaskan praktik terbaik saat bekerja dengan aplikasi Amazon Kinesis Data Analytics.

Topik

- Mengelola Aplikasi
- Menskalakan Aplikasi
- Aplikasi Pemantauan
- Mendefinisikan Skema Input
- Menyambung ke Output
- Menulis Kode Aplikasi
- Menguji Aplikasi

Mengelola Aplikasi

Ketika mengelola aplikasi Amazon Kinesis Data Analytics, ikuti praktik terbaik berikut:

- Mengatur CloudWatch alarm Amazon — Anda dapat menggunakan CloudWatch metrik yang disediakan Kinesis Data Analytics untuk memantau hal-hal berikut:
  - Byte input dan catatan input (jumlah byte dan catatan yang masuk ke aplikasi)
  - Byte output dan catatan output
  - `MillisBehindLatest` (seberapa jauh di belakang aplikasi dalam membaca dari sumber streaming)

  Kami menyarankan Anda menyiapkan setidaknya dua CloudWatch alarm pada metrik berikut untuk aplikasi dalam produksi Anda:
  - `MillisBehindLatest` – Untuk sebagian besar kasus, sebaiknya atur alarm ini agar dipicu saat aplikasi Anda berada 1 jam di belakang data terbaru, rata-rata 1 menit. Untuk aplikasi dengan kebutuhan end-to-end pemrosesan yang lebih rendah, Anda dapat menyetelnya ke toleransi yang lebih rendah. Alarm ini dapat membantu memastikan aplikasi Anda membaca data terbaru.

- Untuk menghindari pengecualian `ReadProvisionedThroughputException`, batasi jumlah aplikasi produksi yang membaca dari Kinesis data stream yang sama ke dua aplikasi.

> **Note**
>
> Dalam kasus ini, aplikasi mengacu pada aplikasi apa pun yang dapat membaca dari sumber streaming. Hanya aplikasi Kinesis Data Analytics yang dapat membaca dari aliran pengiriman Firehose. Namun, banyak aplikasi dapat membaca dari aliran data Kinesis, seperti aplikasi Kinesis Data Analytics atau. AWS Lambda Batas aplikasi yang disarankan merujuk pada semua aplikasi yang Anda konfigurasikan untuk membaca dari sumber streaming.

Amazon Kinesis Data Analytics membaca sumber streaming kira-kira sekali per detik per aplikasi. Namun, aplikasi yang tertinggal mungkin membaca data pada tingkat yang lebih cepat untuk mengejar ketinggalan. Untuk memungkinkan throughput yang memadai bagi aplikasi untuk mengejar ketinggalan, batasi jumlah aplikasi yang membaca sumber data yang sama.

- Batasi jumlah pembacaan aplikasi produksi dari aliran pengiriman Firehose yang sama ke satu aplikasi.

  Aliran pengiriman Firehose dapat menulis ke tujuan seperti Amazon S3 dan Amazon Redshift. Ini juga bisa menjadi sumber streaming untuk aplikasi Kinesis Data Analytics Anda. Oleh karena itu, kami menyarankan agar Anda tidak mengonfigurasi lebih dari satu aplikasi Kinesis Data Analytics per aliran pengiriman Firehose. Hal ini membantu memastikan aliran pengiriman juga dapat dikirim ke tujuan lain.

Menskalakan Aplikasi

Siapkan aplikasi Anda untuk kebutuhan penskalaan Anda di masa mendatang dengan secara proaktif meningkatkan jumlah aliran dalam aplikasi input dari default (satu). Kami merekomendasikan pilihan bahasa berikut berdasarkan throughput aplikasi Anda:

- Gunakan beberapa aliran dan Kinesis Data Analytics untuk aplikasi SQL jika aplikasi Anda harus menskalakan lebih dari 100 MB/detik.

- Gunakan Managed Service untuk Apache Flink Applications jika Anda ingin menggunakan satu aliran dan aplikasi.

> **Note**
>
> Kami menyarankan untuk meninjau `InputProcessing.0kBytes` metrik aplikasi Anda secara berkala sehingga Anda dapat merencanakan ke depan untuk menggunakan beberapa aplikasi SQL atau bermigrasi ke managed-flink/latest/java/ jika throughput input yang diproyeksikan aplikasi Anda melebihi 100 MB/detik.

Aplikasi Pemantauan

Kami menyarankan untuk membuat CloudWatch alarm `InputProcessing.0kBytes` agar Anda diberi tahu saat aplikasi Anda mendekati batas throughput input. Ini dapat berguna karena Anda dapat memperbarui kueri aplikasi Anda ke tradeoff untuk throughput yang lebih tinggi, sehingga menghindari tekanan balik dan penundaan dalam analitik. Untuk informasi selengkapnya, lihat Pemecahan Masalah. Ini juga dapat berguna jika Anda memiliki mekanisme untuk mengurangi throughput di hulu.

- Throughput yang paling kami rekomendasikan untuk satu aliran dalam aplikasi adalah antara 2 dan 20 MB/detik, bergantung pada kompleksitas kueri aplikasi.
- Throughput streaming maksimum yang dapat diproses oleh satu Kinesis Data Analytics untuk aplikasi SQL adalah sekitar 100 MB/detik. Ini mengasumsikan bahwa Anda telah meningkatkan jumlah aliran dalam aplikasi ke nilai maksimum 64, dan Anda telah meningkatkan batas KPU Anda melebihi 8. Untuk informasi selengkapnya, lihat Batasan-batasan.

> **Note**
>
> Kami menyarankan untuk meninjau `InputProcessing.0kBytes` metrik aplikasi Anda secara berkala sehingga Anda dapat merencanakan ke depan untuk menggunakan beberapa aplikasi SQL atau bermigrasi ke managed-flink/latest/java/ jika throughput input yang diproyeksikan aplikasi Anda melebihi 100 MB/detik.

Mendefinisikan Skema Input

Saat mengonfigurasi input aplikasi di konsol, Anda terlebih dahulu menentukan sumber streaming. Konsol selanjutnya menggunakan API penemuan (lihat DiscoverInputSchema) untuk menyimpulkan skema dengan mengambil sampel catatan pada sumber streaming. Skema, di antaranya, mendefinisikan nama dan tipe data kolom di aliran dalam aplikasi yang dihasilkan. Konsol menampilkan skema. Sebaiknya lakukan hal berikut dengan skema yang disimpulkan ini:

- Cukup uji skema yang disimpulkan. Proses penemuan hanya menggunakan contoh catatan pada sumber streaming untuk menyimpulkan skema. Jika sumber streaming Anda memiliki banyak tipe catatan, API penemuan mungkin melewatkan pengambilan sampel satu atau lebih tipe catatan. Situasi ini dapat mengakibatkan skema yang tidak secara akurat mencerminkan data pada sumber streaming.

  Ketika aplikasi Anda dimulai, tipe catatan yang terlewat ini dapat mengakibatkan kesalahan penguraian. Amazon Kinesis Data Analytics mengirimkan catatan ini ke aliran kesalahan dalam aplikasi. Untuk mengurangi kesalahan penguraian ini, sebaiknya uji skema yang disimpulkan secara interaktif di konsol dan pantau aliran dalam aplikasi untuk catatan yang terlewat.

- API Kinesis Data Analytics tidak mendukung batasan `NOT NULL` pada kolom dalam konfigurasi input. Jika Anda menginginkan batasan `NOT NULL` pada kolom di aliran dalam aplikasi Anda, buat aliran dalam aplikasi menggunakan kode aplikasi Anda. Anda selanjutnya dapat menyalin data dari satu aliran dalam aplikasi ke aliran lainnya, lalu batasan diterapkan.

  Upaya apa pun untuk menyisipkan baris dengan nilai `NULL` ketika nilai diperlukan menghasilkan kesalahan. Kinesis Data Analytics mengirimkan kesalahan ini ke aliran kesalahan dalam aplikasi.

- Longgarkan tipe data yang disimpulkan oleh proses penemuan. Proses penemuan merekomendasikan kolom dan tipe data berdasarkan pengambilan sampel acak catatan pada sumber streaming. Sebaiknya tinjau ini dengan hati-hati dan pertimbangkan untuk melonggarkan tipe data ini untuk mencakup semua kemungkinan kasus catatan dalam input Anda. Hal ini memastikan kesalahan penguraian lebih sedikit di seluruh aplikasi saat sedang berjalan. Sebagai contoh, jika skema yang disimpulkan memiliki `SMALLINT` sebagai tipe kolom, pertimbangkan untuk mengubahnya menjadi `INTEGER`.

- Gunakan fungsi SQL dalam kode aplikasi Anda untuk menangani data atau kolom tidak terstruktur apa pun. Anda mungkin memiliki data atau kolom tidak terstruktur, seperti data log, di input Anda. Sebagai contoh, lihat Contoh: Mengubah Nilai DateTime . Salah satu pendekatan untuk menangani tipe data ini adalah menentukan skema dengan hanya satu kolom tipe `VARCHAR(N)`, dengan `N` adalah baris terbesar yang mungkin Anda harapkan untuk dilihat di aliran Anda. Dalam kode aplikasi Anda, Anda selanjutnya dapat membaca catatan masuk dan menggunakan fungsi `String` dan `Date Time` untuk mengurai dan membuat skema data mentah.

- Pastikan Anda benar-benar menangani data sumber streaming yang berisi nest lebih dari dua tingkat. Ketika sumber data adalah JSON, Anda dapat memiliki nest. API penemuan menyimpulkan skema yang meratakan satu tingkat nest. Untuk dua tingkat nest, API penemuan juga mencoba untuk meratakan ini. Di luar dua tingkat nest, ada dukungan terbatas untuk perataan. Untuk menangani nest sepenuhnya, Anda harus secara manual memodifikasi skema yang disimpulkan agar sesuai dengan kebutuhan Anda. Gunakan salah satu strategi berikut untuk melakukannya:

  - Gunakan jalur baris JSON untuk secara selektif menarik keluar hanya pasangan nilai kunci yang diperlukan untuk aplikasi Anda. Jalur baris JSON menyediakan pointer ke pasangan nilai kunci tertentu yang ingin Anda bawa dalam aplikasi Anda. Anda dapat melakukan ini untuk setiap tingkat nest.
  - Gunakan jalur baris JSON untuk secara selektif menarik keluar objek JSON kompleks, lalu menggunakan fungsi manipulasi string dalam kode aplikasi Anda untuk menarik data tertentu yang Anda butuhkan.

Menyambung ke Output

Sebaiknya setiap aplikasi memiliki setidaknya dua output:

- Gunakan tujuan pertama untuk memasukkan hasil kueri SQL Anda.
- Gunakan tujuan kedua untuk memasukkan seluruh aliran kesalahan dan mengirimkannya ke bucket S3 melalui aliran pengiriman Firehose.

Menulis Kode Aplikasi

Sebaiknya lakukan hal berikut:

- Dalam pernyataan SQL Anda, jangan tentukan jendela berbasis waktu yang lebih dari satu jam untuk alasan berikut:
  - Terkadang aplikasi harus dimulai ulang, baik karena Anda memperbarui aplikasi maupun alasan internal Kinesis Data Analytics. Saat dimulai ulang, semua data yang disertakan dalam jendela harus dibaca lagi dari sumber data streaming. Ini memerlukan waktu sebelum Kinesis Data Analytics dapat memancarkan output untuk jendela tersebut.
  - Kinesis Data Analytics harus mempertahankan segala sesuatu yang terkait dengan status aplikasi, termasuk data yang relevan, selama durasi. Ini menggunakan unit pemrosesan Kinesis Data Analytics yang signifikan.
- Selama pengembangan, jaga agar ukuran jendela tetap kecil dalam pernyataan SQL Anda agar Anda dapat melihat hasil lebih cepat. Ketika Anda men-deploy aplikasi untuk lingkungan produksi Anda, Anda dapat mengatur ukuran jendela yang sesuai.
- Sebagai ganti satu pernyataan SQL kompleks, pertimbangkan untuk membaginya menjadi beberapa pernyataan, di setiap langkah yang menyimpan hasil di aliran dalam aplikasi menengah. Hal ini dapat membantu Anda melakukan debug lebih cepat.
- Saat Anda menggunakan jendela tumbling, sebaiknya gunakan dua jendela, satu untuk waktu pemrosesan dan satu untuk waktu logis Anda (waktu menyerap atau waktu peristiwa). Untuk informasi selengkapnya, lihat Stempel waktu dan Kolom ROWTIME.

Menguji Aplikasi

Saat Anda mengubah skema atau kode aplikasi untuk aplikasi Kinesis Data Analytics, sebaiknya gunakan aplikasi uji untuk memverifikasi perubahan sebelum di-deploy ke produksi.

Menyiapkan Aplikasi Uji

Anda dapat menyiapkan aplikasi uji baik melalui konsol maupun menggunakan templat AWS CloudFormation . Menggunakan AWS CloudFormation templat membantu memastikan bahwa perubahan kode yang Anda buat pada aplikasi pengujian dan aplikasi langsung Anda konsisten.

Saat menyiapkan aplikasi uji, Anda dapat menghubungkan aplikasi ke data langsung Anda, atau Anda dapat mengisi aliran dengan data tiruan untuk menguji. Kami merekomendasikan dua metode untuk mengisi aliran dengan data tiruan:

- Gunakan Kinesis Data Generator (KDG). KDG menggunakan templat data untuk mengirim data acak ke aliran Kinesis. KDG mudah digunakan, tetapi tidak sesuai untuk menguji hubungan kompleks di antara item data, seperti untuk aplikasi yang mendeteksi hotspot atau anomali data.
- Gunakan aplikasi Python kustom untuk mengirim data yang lebih kompleks ke Kinesis data stream. Aplikasi Python dapat menghasilkan hubungan yang kompleks di antara item data, seperti hotspot atau anomali. Untuk contoh aplikasi Python yang mengirimkan data yang diklasterkan ke dalam hotspot data, lihat Contoh: Mendeteksi Hotspot di Aliran (Fungsi HOTSPOTS).

Saat menjalankan aplikasi pengujian, lihat hasil Anda menggunakan tujuan (seperti aliran pengiriman Firehose ke database Amazon Redshift) alih-alih melihat aliran dalam aplikasi di konsol. Data yang ditampilkan pada konsol adalah pengambilan sampel dari aliran dan tidak berisi semua catatan.

Menguji Perubahan Skema

Ketika mengubah skema aliran input aplikasi, gunakan aplikasi uji Anda untuk memverifikasi berikut ini benar:

- Data dari aliran Anda sedang dipaksa menjadi tipe data yang benar. Sebagai contoh, memastikan data datetime tidak sedang diserap ke dalam aplikasi sebagai string.
- Data yang diurai dan dipaksa menjadi tipe data yang Anda inginkan. Jika terjadi kesalahan penguraian atau paksaan, Anda dapat melihatnya di konsol, atau menetapkan tujuan untuk aliran kesalahan dan melihat kesalahan di penyimpanan tujuan.
- Bidang data untuk data karakter cukup panjang, dan aplikasi tidak memotong data karakter. Anda dapat memeriksa catatan data di penyimpanan tujuan Anda untuk memverifikasi data aplikasi Anda tidak terpotong.

Menguji Perubahan Kode

Pengujian perubahan kode SQL Anda memerlukan beberapa pengetahuan domain aplikasi Anda. Anda harus dapat menentukan output yang perlu diuji dan output benar yang seharusnya. Untuk area masalah potensial, untuk memverifikasi ketika memodifikasi kode SQL aplikasi Anda, lihat Pemecahan Masalah Amazon Kinesis Data Analytics untuk Aplikasi SQL.

Pemecahan Masalah Amazon Kinesis Data Analytics untuk Aplikasi SQL

Berikut ini dapat membantu Anda memecahkan masalah yang mungkin terjadi dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL.

Topik

- Aplikasi yang dihentikan
- Tidak Dapat Menjalankan Kode SQL
- Tidak Dapat Mendeteksi atau Menemukan Skema Saya
- Data Referensi Kedaluwarsa
- Aplikasi Tidak Menulis ke Tujuan
- Parameter Kondisi Aplikasi Penting untuk Dipantau
- Kesalahan Kode Tidak Valid Saat Menjalankan Aplikasi
- Aplikasi Menulis Kesalahan ke Aliran Kesalahan
- Throughput Tidak Cukup atau Tinggi MillisBehindLatest

Aplikasi yang dihentikan

- Apa yang dimaksud dengan Kinesis Data Analytics yang dihentikan untuk aplikasi SQL?

  Aplikasi yang dihentikan adalah aplikasi yang kami amati tidak memproses catatan apa pun selama minimal tiga bulan. Ini berarti pelanggan membayar Kinesis Data Analytics untuk sumber daya SQL yang tidak mereka gunakan.
- Kapan akan AWS mulai menghentikan aplikasi idle?

  AWSakan mulai menghentikan aplikasi idle pada 14 November 2023 dan selesai pada 21 November 2023. Kami akan menghentikan aplikasi idle di zona waktu jam kantor Wilayah itu.
- Dapatkah Kinesis Data Analytics yang dihentikan untuk aplikasi SQL dimulai kembali?

  Ya. Jika Anda perlu memulai kembali aplikasi Anda, Anda dapat melakukannya seperti biasa. Tidak perlu memotong tiket dukungan.
- Ketika AWS berhenti aplikasi idle, apakah hasil kueri saya juga akan dihapus?

Tidak. Pertama, karena aplikasi Anda menganggur, itu tidak memproses kueri. Kedua, hasil kueri Anda tidak disimpan dalam Kinesis Data Analytics untuk SQL. Anda mengonfigurasi Kinesis Data Analytics untuk aplikasi SQL dengan tujuan sink tempat hasil perhitungannya dikirim (misalnya, di Amazon S3 atau aliran data lainnya). Dengan demikian, Anda mempertahankan kepemilikan penuh atas data Anda dan itu akan tetap dapat diambil berdasarkan ketentuan layanan penyimpanan tersebut.

- Apa yang harus saya lakukan jika saya tidak ingin aplikasi saya dihentikan?

  Anda dapat mengirim email ke tim layanan (kda-sql-questions@amazon .com) yang meminta aplikasi tidak dihentikan kapan saja sebelum 10 November 2023. Email harus menyertakan ID akun dan ARN aplikasi Anda.

Tidak Dapat Menjalankan Kode SQL

Jika Anda perlu mencari tahu cara membuat pernyataan SQL tertentu agar berfungsi dengan benar, Anda memiliki beberapa sumber daya berbeda saat menggunakan Kinesis Data Analytics:

- Untuk informasi selengkapnya tentang pernyataan SQL, lihat Kinesis Data Analytics untuk contoh SQL. Bagian ini menyediakan sejumlah contoh SQL yang dapat Anda gunakan.
- Referensi SQL Amazon Kinesis Data Analytics menyediakan panduan terperinci untuk menulis pernyataan SQL streaming.
- Jika Anda masih mengalami masalah, sebaiknya ajukan pertanyaan di Forum Kinesis Data Analytics.

Tidak Dapat Mendeteksi atau Menemukan Skema Saya

Dalam beberapa kasus, Kinesis Data Analytics tidak dapat mendeteksi atau menemukan skema. Dalam banyak kasus ini, Anda masih dapat menggunakan Kinesis Data Analytics.

Misalkan Anda memiliki data yang dikodekan UTF-8 yang tidak menggunakan pembatas, atau data yang menggunakan format selain nilai yang dipisahkan koma (CSV), atau API penemuan tidak menemukan skema Anda. Dalam kasus ini, Anda dapat menentukan skema secara manual atau menggunakan fungsi manipulasi string untuk menyusun data Anda.

Untuk menemukan skema aliran Anda, Kinesis Data Analytics secara acak mengambil sampel data terbaru dalam aliran Anda. Jika Anda tidak mengirimkan data ke aliran secara konsisten, Kinesis

Data Analytics mungkin tidak dapat mengambil sampel dan mendeteksi skema. Untuk informasi selengkapnya, lihat Menggunakan Fitur Penemuan Skema pada Data Streaming.

Data Referensi Kedaluwarsa

Data referensi dimuat dari objek Amazon Simple Storage Service (Amazon S3) ke dalam aplikasi ketika aplikasi dimulai atau diperbarui, atau selama gangguan aplikasi yang disebabkan oleh masalah layanan.

Data referensi tidak dimuat ke dalam aplikasi ketika pembaruan dibuat untuk objek Amazon S3 yang mendasari.

Jika data referensi dalam aplikasi bukan yang terbaru, Anda dapat memuat ulang data dengan mengikuti langkah-langkah berikut:

1. Di konsol Kinesis Data Analytics, pilih nama aplikasi dalam daftar, lalu pilih Application details (Detail aplikasi).
2. Pilih Go to SQL editor (Buka editor SQL) untuk membuka halaman Analitik waktu nyata untuk aplikasi.
3. Di tampilan Data Sumber, pilih nama tabel data referensi Anda.
4. Pilih Actions (Tindakan), Synchronize reference data table (Sinkronkan tabel data referensi).

Aplikasi Tidak Menulis ke Tujuan

Jika data tidak ditulis ke tujuan, periksa berikut ini:

- Pastikan peran aplikasi memiliki izin yang memadai untuk mengakses tujuan. Untuk informasi lebih lanjut, lihat Kebijakan Izin untuk Menulis ke Aliran Kinesis atau Kebijakan Izin untuk Menulis ke Aliran Pengiriman Firehose.
- Pastikan tujuan aplikasi dikonfigurasi dengan benar dan aplikasi menggunakan nama yang benar untuk aliran output.
- Periksa CloudWatch metrik Amazon untuk aliran keluaran Anda untuk melihat apakah data sedang ditulis. Untuk informasi tentang menggunakan CloudWatch metrik, lihatPemantauan CloudWatch dengan Amazon.
- Tambahkan aliran CloudWatch log menggunakanthe section called “AddApplicationCloudWatchLoggingOption”. Aplikasi Anda akan menulis kesalahan konfigurasi ke aliran log.

Jika konfigurasi peran dan tujuan terlihat benar, coba mulai ulang aplikasi, yang menentukan `LAST_STOPPED_POINT` untuk InputStartingPositionConfiguration.

Parameter Kondisi Aplikasi Penting untuk Dipantau

Untuk memastikan aplikasi Anda berjalan dengan benar, sebaiknya pantau parameter penting tertentu.

Parameter yang paling penting untuk dipantau adalah CloudWatch metrik Amazon`MillisBehindLatest`. Metrik ini menunjukkan seberapa jauh waktu saat ini yang Anda baca dari aliran. Metrik ini membantu Anda menentukan apakah Anda sedang memproses catatan dari aliran sumber yang cukup cepat.

Sebagai aturan umum, Anda harus mengatur CloudWatch alarm untuk memicu jika Anda tertinggal lebih dari satu jam. Namun, jumlah waktu bergantung pada kasus penggunaan Anda. Anda dapat menyesuaikannya sesuai kebutuhan.

Untuk informasi selengkapnya, lihat Praktik Terbaik.

Kesalahan Kode Tidak Valid Saat Menjalankan Aplikasi

Jika Anda tidak dapat menyimpan dan menjalankan kode SQL untuk aplikasi Amazon Kinesis Data Analytics, berikut adalah penyebab umumnya:

- Aliran didefinisikan ulang dalam kode SQL – Setelah Anda membuat aliran dan pompa yang terkait dengan aliran, Anda tidak dapat mendefinisikan ulang aliran yang sama dalam kode Anda. Untuk informasi selengkapnya tentang membuat aliran, lihat BUAT ALIRAN di Referensi SQL Amazon Kinesis Data Analytics. Untuk informasi selengkapnya tentang membuat pompa, lihat BUAT POMPA.
- Klausa GROUP BY menggunakan beberapa kolom ROWTIME – Anda dapat menentukan hanya satu kolom ROWTIME dalam klausa GROUP BY. Untuk informasi selengkapnya, lihat GROUP BY dan ROWTIME di Referensi SQL Amazon Kinesis Data Analytics.
- Satu atau beberapa tipe data memiliki transmisi yang tidak valid – Dalam kasus ini, kode Anda memiliki transmisi implisit yang tidak valid. Misalnya, Anda mungkin mentransmisikan `timestamp` ke `bigint` dalam kode Anda.
- Aliran memiliki nama yang sama dengan nama aliran yang disimpan layanan – Aliran tidak boleh memiliki nama yang sama dengan aliran yang disimpan layanan `error_stream`.

Aplikasi Menulis Kesalahan ke Aliran Kesalahan

Jika aplikasi Anda menulis kesalahan ke aliran kesalahan dalam aplikasi, Anda dapat mendekode nilai dalam bidang `DATA_ROW` menggunakan pustaka standar. Untuk informasi selengkapnya tentang aliran kesalahan, lihat Penanganan Kesalahan.

Throughput Tidak Cukup atau Tinggi MillisBehindLatest

Jika MillisBehindLatestmetrik aplikasi Anda terus meningkat atau secara konsisten di atas 1000 (satu detik), itu bisa disebabkan oleh alasan berikut:

- Periksa InputBytes CloudWatch metrik aplikasi Anda. Jika Anda menelan lebih dari 4 MB/detik, ini dapat menyebabkan peningkatan dalam `MillisBehindLatest`. Untuk meningkatkan throughput aplikasi Anda, tingkatkan nilai parameter `InputParallelism`. Untuk informasi selengkapnya, lihat Menyejajarkan Aliran Input untuk Peningkatan Throughput.
- Periksa metrik Success pengiriman output aplikasi Anda untuk kegagalan dalam pengiriman ke tujuan Anda. Pastikan Anda sudah mengonfigurasi output dengan benar, dan aliran output Anda memiliki kapasitas yang memadai.
- Jika aplikasi Anda menggunakan AWS Lambda fungsi untuk pra-pemrosesan atau sebagai output, periksa metrik .Duration atau InputProcessingLambdaDelivery CloudWatch .Duration aplikasi. Jika durasi invokasi fungsi Lambda lebih lama dari 5 detik, pertimbangkan untuk melakukan hal berikut:
  - Tingkatkan alokasi Memori fungsi Lambda. Anda dapat melakukannya di konsol AWS Lambda, pada halaman Konfigurasi, di bawah Pengaturan Dasar. Untuk informasi selengkapnya, lihat Mengonfigurasi Fungsi Lambda dalam Panduan Developer AWS Lambda.
  - Tingkatkan jumlah serpihan dalam aliran input aplikasi Anda. Ini meningkatkan jumlah fungsi paralel yang akan dipanggil aplikasi, yang mungkin meningkatkan throughput.
  - Pastikan fungsi tidak memblokir panggilan yang memengaruhi performa, seperti permintaan sinkron untuk sumber daya eksternal.
  - Periksa fungsi AWS Lambda untuk melihat apakah ada area lain tempat Anda dapat meningkatkan performa. Periksa CloudWatch Log fungsi Lambda aplikasi. Untuk informasi selengkapnya, lihat Mengakses CloudWatch Metrik Amazon di Panduan AWS LambdaPengembang.
- Pastikan aplikasi Anda tidak mencapai batas default untuk Unit Pemrosesan Kinesis (KPU). Jika aplikasi Anda mencapai batas ini, Anda dapat meminta peningkatan batas. Untuk informasi selengkapnya, lihat Secara Otomatis Menskalakan Aplikasi untuk Meningkatkan Throughput.

- Jika aplikasi Anda masih mengalami masalah setelah batas KPU Anda meningkat, periksa apakah input throughput aplikasi Anda tidak melebihi 100MB/detik. Jika melebihi 100MB/detik, kami sarankan menerapkan perubahan untuk mengurangi throughput keseluruhan untuk menstabilkan aplikasi, misalnya dengan mengurangi jumlah data yang dikirim ke sumber data yang dibaca aplikasi Kinesis Data Analytics Sql. Kami juga merekomendasikan pendekatan lain, termasuk meningkatkan paralelisme aplikasi, mengurangi periode waktu perhitungan, mengubah tipe data kolumnar dari VARCHAR ke tipe data dengan ukuran yang lebih kecil (misalnya, INTEGER, LONG, dll), dan mengurangi data yang diproses dengan pengambilan sampel atau penyaringan.

> **Note**
>
> Kami menyarankan untuk meninjau `InputProcessing.OkBytes` metrik aplikasi Anda secara berkala sehingga Anda dapat merencanakan ke depan untuk menggunakan beberapa aplikasi SQL atau bermigrasi ke managed-flink/latest/java/ jika throughput input yang diproyeksikan aplikasi Anda akan melebihi 100 MB/detik.

Referensi SQL Kinesis Data Analytics

Untuk informasi tentang elemen bahasa SQL yang didukung oleh Kinesis Data Analytics, lihat Referensi SQL Kinesis Data Analytics.

Referensi API

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Layanan Terkelola Amazon untuk Dokumentasi Apache Flink API V2.

Anda dapat menggunakannya AWS CLI untuk menjelajahi API Amazon Kinesis Data Analytics. Panduan ini menyediakan latihan Memulai dengan Amazon Kinesis Data Analytics untuk Aplikasi SQL yang menggunakan AWS CLI.

Topik

- Tindakan
- Tipe Data

Tindakan

Tindakan berikut didukung:

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- CreateApplication
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource

- DescribeApplication
- DiscoverInputSchema
- ListApplications
- ListTagsForResource
- StartApplication
- StopApplication
- TagResource
- UntagResource
- UpdateApplication

AddApplicationCloudWatchLoggingOption

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menambahkan aliran CloudWatch log untuk memantau kesalahan konfigurasi aplikasi. Untuk informasi selengkapnya tentang penggunaan aliran CloudWatch log dengan aplikasi Amazon Kinesis Analytics, lihat Bekerja dengan Log Amazon. CloudWatch

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "string",
      "RoleARN": "string"
   },
   "CurrentApplicationVersionId": number
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Kinesis Analytics.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CloudWatchLoggingOption

Menyediakan CloudWatch log stream Amazon Resource Name (ARN) dan peran IAM ARN. Catatan: Untuk menulis pesan aplikasi CloudWatch, peran IAM yang digunakan harus mengaktifkan tindakan `PutLogEvents` kebijakan.

Tipe: Objek CloudWatchLoggingOption

Wajib: Ya

CurrentApplicationVersionId

ID versi aplikasi Kinesis Analytics.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

AddApplicationInput

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menambahkan sumber streaming ke aplikasi Amazon Kinesis Anda. Untuk informasi konseptual, lihat Mengonfigurasi Input Aplikasi.

Anda dapat menambahkan sumber streaming ketika Anda membuat aplikasi atau Anda dapat menggunakan operasi ini untuk menambahkan sumber streaming setelah Anda membuat aplikasi. Untuk informasi lebih lanjut, lihat CreateApplication.

Pembaruan konfigurasi apa pun, termasuk menambahkan sumber streaming menggunakan operasi ini, menghasilkan versi baru aplikasi. Anda dapat menggunakan DescribeApplicationoperasi untuk menemukan versi aplikasi saat ini.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:AddApplicationInput`.

Sintaksis Permintaan

```
{
   "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"
    }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Amazon Kinesis Analytics yang ada yang ingin Anda tambahkan sumber streaming-nya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

Versi aplikasi Amazon Kinesis Analytics Anda saat ini. Anda dapat menggunakan DescribeApplicationoperasi untuk menemukan versi aplikasi saat ini.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

Input

Input yang ditambahkan.

Tipe: Objek Input

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

CodeValidationException

Kode aplikasi yang disediakan pengguna (kueri) tidak valid. Ini bisa menjadi kesalahan sintaksis sederhana.

Kode Status HTTP: 400

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

AddApplicationInputProcessingConfiguration

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menambahkan InputProcessingConfigurationke aplikasi. Prosesor input memproses catatan terlebih dulu pada aliran input sebelum kode SQL aplikasi dijalankan. Saat ini, satu-satunya prosesor input yang tersedia adalah AWS Lambda.

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi tempat Anda ingin menambahkan konfigurasi pemrosesan input.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

Versi aplikasi tempat Anda ingin menambahkan konfigurasi pemrosesan input. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan versi aplikasi saat ini. Jika versi yang ditentukan bukan versi saat ini, `ConcurrentModificationException` dikembalikan.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

InputId

ID konfigurasi input untuk menambahkan konfigurasi pemrosesan input. Anda bisa mendapatkan daftar ID input untuk aplikasi menggunakan DescribeApplicationoperasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

InputProcessingConfiguration

InputProcessingConfigurationUntuk ditambahkan ke aplikasi.

Tipe: Objek InputProcessingConfiguration

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

AddApplicationOutput

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menambahkan tujuan eksternal ke aplikasi Amazon Kinesis Analytics Anda.

Jika Anda ingin Amazon Kinesis Analytics mengirimkan data dari aliran dalam aplikasi dalam aplikasi ke tujuan eksternal (seperti aliran Amazon Kinesis, aliran pengiriman Amazon Kinesis Firehose, atau fungsi AWS Lambda), Anda menambahkan konfigurasi yang relevan ke aplikasi menggunakan operasi ini. Anda dapat mengonfigurasi satu atau beberapa output untuk aplikasi Anda. Setiap konfigurasi output memetakan pengaliran dalam aplikasi dan tujuan eksternal.

Anda dapat menggunakan salah satu konfigurasi output untuk mengirimkan data dari pengaliran kesalahan dalam aplikasi Anda ke tujuan eksternal agar Anda dapat menganalisis kesalahan. Untuk informasi selengkapnya, lihat Memahami Output Aplikasi (Tujuan).

Pembaruan konfigurasi apa pun, termasuk menambahkan sumber streaming menggunakan operasi ini, menghasilkan versi baru aplikasi. Anda dapat menggunakan DescribeApplicationoperasi untuk menemukan versi aplikasi saat ini.

Untuk batas jumlah input dan output aplikasi yang dapat Anda konfigurasi, lihat Batas.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:AddApplicationOutput`.

Sintaksis Permintaan

```
{
   "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"
   }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi tempat Anda ingin menambahkan konfigurasi output.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

Versi aplikasi tempat Anda ingin menambahkan konfigurasi output. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan versi aplikasi saat ini. Jika versi yang ditentukan bukan versi saat ini, `ConcurrentModificationException` dikembalikan.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

Output

Susunan objek, masing-masing menjelaskan satu konfigurasi output. Dalam konfigurasi keluaran, Anda menentukan nama aliran dalam aplikasi, tujuan (yaitu aliran Amazon Kinesis, aliran pengiriman Amazon Kinesis Firehose, atau fungsi Lambda), dan merekam formasi AWS yang akan digunakan saat menulis ke tujuan.

Tipe: Objek Output

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

AddApplicationReferenceDataSource

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menambahkan sumber data referensi ke aplikasi yang ada.

Amazon Kinesis Analytics membaca data referensi (yaitu, objek Amazon S3) dan membuat tabel dalam aplikasi di dalam aplikasi Anda. Dalam permintaan, Anda menyediakan sumber (nama bucket S3 dan nama kunci objek), nama tabel dalam aplikasi yang dibuat, dan informasi pemetaan yang diperlukan yang menjelaskan cara memetakan data di objek Amazon S3 ke kolom di tabel dalam aplikasi yang dihasilkan.

Untuk informasi konseptual, lihat Mengonfigurasi Input Aplikasi. Untuk batas sumber data yang dapat ditambahkan ke aplikasi, lihat Batas.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:AddApplicationOutput`.

Sintaksis Permintaan

```
{
   "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"
  }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi yang ada.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

Versi aplikasi tempat Anda menambahkan sumber data referensi. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan versi aplikasi saat ini. Jika versi yang ditentukan bukan versi saat ini, `ConcurrentModificationException` dikembalikan.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

ReferenceDataSource

Sumber data referensi dapat berupa objek dalam bucket Amazon S3 Anda. Amazon Kinesis Analytics membaca objek dan menyalin data ke tabel dalam aplikasi yang dibuat. Anda menyediakan bucket S3, objek kunci nama, dan tabel dalam aplikasi yang dihasilkan yang dibuat. Anda juga harus menyediakan IAM role dengan izin yang diperlukan yang dapat diambil Amazon Kinesis Analytics untuk membaca objek dari bucket S3 atas nama Anda.

Tipe: Objek ReferenceDataSource

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

CreateApplication

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Membuat aplikasi Amazon Kinesis Analytics. Anda dapat mengonfigurasi setiap aplikasi dengan satu sumber streaming sebagai input, kode aplikasi untuk memproses input, dan hingga tiga tujuan tempat Anda ingin Amazon Kinesis Analytics menulis data output dari aplikasi Anda. Untuk gambaran umum, lihat Cara Kerjanya.

Dalam konfigurasi input, Anda memetakan sumber streaming ke aliran dalam aplikasi, yang dapat Anda anggap sebagai tabel yang terus diperbarui. Dalam pemetaan, Anda harus menyediakan skema untuk aliran dalam aplikasi dan memetakan setiap kolom data di aliran dalam aplikasi untuk elemen data dalam sumber streaming.

Kode aplikasi Anda adalah satu atau beberapa pernyataan SQL yang membaca data input, mengubahnya, dan menghasilkan output. Kode aplikasi Anda dapat membuat satu atau beberapa artefak SQL seperti aliran atau pompa SQL.

Dalam konfigurasi output, Anda dapat mengonfigurasi aplikasi untuk menulis data dari aliran dalam aplikasi yang dibuat dalam aplikasi Anda hingga tiga tujuan.

Untuk membaca data dari aliran sumber Anda atau menulis data ke aliran tujuan, Amazon Kinesis Analytics memerlukan izin Anda. Anda memberikan izin ini dengan membuat IAM role. Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:CreateApplication`.

Untuk latihan pendahuluan untuk membuat aplikasi Amazon Kinesis Analytics, lihat Memulai.

Sintaksis Permintaan

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

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationCode

Satu atau beberapa pernyataan SQL yang membaca data input, mengubahnya, dan menghasilkan output. Misalnya, Anda dapat menulis pernyataan SQL yang membaca data dari satu pengaliran dalam aplikasi, menghasilkan rata-rata jumlah klik iklan oleh vendor yang berjalan, dan menyisipkan baris yang dihasilkan di pengaliran dalam aplikasi lainnya menggunakan pompa. Untuk informasi lebih lanjut tentang pola umum, lihat Kode Aplikasi.

Anda dapat menyediakan serangkaian pernyataan SQL tersebut, di mana output dari satu pernyataan dapat digunakan sebagai input untuk pernyataan berikutnya. Anda menyimpan hasil antara dengan membuat pengaliran dalam aplikasi dan pompa.

Perhatikan bahwa kode aplikasi harus membuat pengaliran dengan nama yang ditentukan di `Outputs`. Misalnya, jika `Outputs` menentukan aliran output bernama `ExampleOutputStream1` dan `ExampleOutputStream2`, kode aplikasi Anda harus membuat aliran ini.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 102400.

Wajib: Tidak

ApplicationDescription

Deskripsi ringkasan dari aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 1024.

Wajib: Tidak

ApplicationName

Nama dari aplikasi Amazon Kinesis Analytics Anda (misalnya, `sample-app`).

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CloudWatchLoggingOptions

Gunakan parameter ini untuk mengonfigurasi aliran CloudWatch log untuk memantau kesalahan konfigurasi aplikasi. Untuk informasi selengkapnya, lihat Bekerja dengan CloudWatch Log Amazon.

Tipe: Array objek CloudWatchLoggingOption

Wajib: Tidak

Inputs

Gunakan parameter ini untuk mengonfigurasi input aplikasi.

Anda dapat mengonfigurasi aplikasi Anda untuk menerima input dari sumber streaming tunggal. Dalam konfigurasi ini, Anda memetakan sumber streaming ini ke pengaliran dalam aplikasi yang dibuat. Kode aplikasi Anda kemudian dapat mengkueri pengaliran dalam aplikasi seperti tabel (Anda dapat menganggapnya sebagai tabel yang terus memperbarui).

Untuk sumber streaming, Anda menyediakan Amazon Resource Name (ARN) dan format data pada pengaliran (misalnya, JSON, CSV, dll.). Anda juga harus menyediakan IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk membaca pengaliran ini atas nama Anda.

Untuk membuat pengaliran dalam aplikasi, Anda perlu menentukan skema untuk mengubah data Anda menjadi versi skematis yang digunakan di SQL. Dalam skema tersebut, Anda menyediakan pemetaan dari elemen data yang diperlukan dalam sumber streaming untuk mencatat kolom di aliran dalam aplikasi.

Tipe: Array objek Input

Wajib: Tidak

Outputs

Anda dapat mengonfigurasi output aplikasi untuk menulis data dari salah satu aliran dalam aplikasi hingga tiga tujuan.

Tujuan ini bisa berupa Amazon Kinesis streams, aliran pengiriman Amazon Kinesis Firehose, tujuan Lambda AWS , atau kombinasi ketiganya.

Dalam konfigurasi, Anda menentukan nama aliran dalam aplikasi, aliran tujuan, atau fungsi Lambda Amazon Resource Name (ARN), dan format yang akan digunakan saat menulis data. Anda juga harus menyediakan IAM role yang dapat diambil Amazon Kinesis Analytics untuk menulis ke aliran tujuan atau fungsi Lambda atas nama Anda.

Dalam konfigurasi output, Anda juga menyediakan aliran output atau fungsi Lambda ARN. Untuk tujuan aliran, Anda menyediakan format data dalam aliran (misalnya, JSON, CSV). Anda juga harus menyediakan IAM role yang dapat diambil Amazon Kinesis Analytics untuk menulis ke aliran atau fungsi Lambda atas nama Anda.

Tipe: Array objek Output

Wajib: Tidak

Tags

Daftar satu atau beberapa tanda yang ditetapkan ke aplikasi. Tanda adalah pasangan nilai kunci yang mengidentifikasi aplikasi. Perhatikan bahwa jumlah maksimum tanda aplikasi mencakup tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 200 item.

Wajib: Tidak

Sintaksis Respons

```
{
   "ApplicationSummary": {
      "ApplicationARN": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string"
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

ApplicationSummary

Untuk merespons permintaan `CreateApplication` Anda, Amazon Kinesis Analytics mengembalikan respons dengan ringkasan aplikasi yang dibuatnya, termasuk aplikasi Amazon Resource Name (ARN), nama, dan status.

Tipe: Objek ApplicationSummary

Kesalahan

CodeValidationException

Kode aplikasi yang disediakan pengguna (kueri) tidak valid. Ini bisa menjadi kesalahan sintaksis sederhana.

Kode Status HTTP: 400

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

LimitExceededException

Melebihi jumlah aplikasi yang diizinkan.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

TooManyTagsException

Aplikasi yang dibuat dengan terlalu banyak tanda, atau terlalu banyak tanda yang ditambahkan ke aplikasi. Perhatikan bahwa jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteApplication

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menghapus aplikasi yang ditentukan. Amazon Kinesis Analytics menghentikan eksekusi aplikasi dan menghapus aplikasi, termasuk artefak aplikasi apa pun (seperti aliran dalam aplikasi, tabel referensi, dan kode aplikasi).

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:DeleteApplication`.

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CreateTimestamp": number
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Amazon Kinesis Analytics yang dihapus.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CreateTimestamp

Anda dapat menggunakan operasi `DescribeApplication` untuk mendapatkan nilai ini.

Tipe: Timestamp

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteApplicationCloudWatchLoggingOption

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menghapus aliran CloudWatch log dari aplikasi. Untuk informasi selengkapnya tentang penggunaan aliran CloudWatch log dengan aplikasi Amazon Kinesis Analytics, lihat Bekerja dengan Log Amazon. CloudWatch

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOptionId": "string",
   "CurrentApplicationVersionId": number
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Kinesis Analytics.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CloudWatchLoggingOptionId

Opsi CloudWatch logging untuk dihapus. `CloudWatchLoggingOptionId` Anda bisa mendapatkan `CloudWatchLoggingOptionId` dengan menggunakan DescribeApplicationoperasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

ID versi aplikasi Kinesis Analytics.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteApplicationInputProcessingConfiguration

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menghapus InputProcessingConfigurationdari input.

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Kinesis Analytics.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

ID versi aplikasi Kinesis Analytics.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

InputId

ID konfigurasi input untuk menghapus konfigurasi pemrosesan input. Anda bisa mendapatkan daftar ID input untuk aplikasi dengan menggunakan DescribeApplicationoperasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteApplicationOutput

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menghapus konfigurasi tujuan output dari konfigurasi aplikasi Anda. Amazon Kinesis Analytics tidak akan lagi menulis data dari aliran dalam aplikasi yang sesuai ke tujuan output eksternal.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:DeleteApplicationOutput`.

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "OutputId": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Amazon Kinesis Analytics.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

Versi aplikasi Amazon Kinesis Analytics. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan versi aplikasi saat ini. Jika versi yang ditentukan bukan versi saat ini, `ConcurrentModificationException` dikembalikan.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

OutputId

ID konfigurasi yang akan dihapus. Setiap konfigurasi output yang ditambahkan ke aplikasi, baik ketika aplikasi dibuat atau nanti menggunakan AddApplicationOutputoperasi, memiliki ID unik. Anda perlu memberikan ID untuk secara unik mengidentifikasi konfigurasi output yang ingin Anda hapus dari konfigurasi aplikasi. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan yang spesifik`OutputId`.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteApplicationReferenceDataSource

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menghapus konfigurasi sumber data referensi dari konfigurasi aplikasi yang ditentukan.

Jika aplikasi berjalan, Amazon Kinesis Analytics segera menghapus tabel dalam aplikasi yang Anda buat menggunakan AddApplicationReferenceDataSourceoperasi.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics.DeleteApplicationReferenceDataSource`.

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceId": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi yang ada.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

CurrentApplicationVersionId

Versi aplikasi. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan versi aplikasi saat ini. Jika versi yang ditentukan bukan versi saat ini, `ConcurrentModificationException` dikembalikan.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

ReferenceId

ID dari sumber data referensi. Saat Anda menambahkan sumber data referensi ke aplikasi Anda menggunakan AddApplicationReferenceDataSource, Amazon Kinesis Analytics menetapkan ID. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan ID referensi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DescribeApplication

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Mengembalikan informasi tentang aplikasi Amazon Kinesis Analytics tertentu.

Jika Anda ingin mengambil daftar semua aplikasi di akun Anda, gunakan ListApplicationsoperasi.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:DescribeApplication`. Anda dapat menggunakan `DescribeApplication` untuk mendapatkan versionId aplikasi saat ini, yang Anda perlukan untuk memanggil operasi lain seperti `Update`.

Sintaksis Permintaan

```
{
   "ApplicationName": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Ya

Sintaksis Respons

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

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

ApplicationDetail

Menyediakan deskripsi aplikasi, seperti aplikasi Amazon Resource Name (ARN), status, versi terbaru, dan detail konfigurasi input serta output.

Tipe: Objek ApplicationDetail

Kesalahan

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DiscoverInputSchema

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menyimpulkan skema dengan mengevaluasi catatan sampel pada sumber streaming yang ditentukan (Amazon Kinesis stream atau aliran pengiriman Amazon Kinesis Firehose) atau objek S3. Dalam respons, operasi mengembalikan skema disimpulkan dan juga catatan sampel yang digunakan operasi untuk menyimpulkan skema.

Anda dapat menggunakan skema yang disimpulkan ketika mengonfigurasi sumber streaming untuk aplikasi Anda. Untuk informasi konseptual, lihat Mengonfigurasi Input Aplikasi. Perhatikan bahwa ketika Anda membuat aplikasi menggunakan konsol Amazon Kinesis Analytics, konsol menggunakan operasi ini untuk menyimpulkan skema dan menunjukkannya di antarmuka pengguna konsol.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:DiscoverInputSchema`.

Sintaksis Permintaan

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "string"
   },
   "ResourceARN": "string",
   "RoleARN": "string",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string",
      "RoleARN": "string"
```

```
    }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

[InputProcessingConfiguration](#)

Yang digunakan [InputProcessingConfiguration](#)untuk memproses catatan sebelum menemukan skema catatan.

Tipe: Objek [InputProcessingConfiguration](#)

Wajib: Tidak

[InputStartingPositionConfiguration](#)

Tunjuk tempat Anda ingin Amazon Kinesis Analytics mulai membaca catatan dari tujuan penemuan sumber streaming yang ditentukan.

Tipe: Objek [InputStartingPositionConfiguration](#)

Wajib: Tidak

[ResourceARN](#)

Amazon Resource Name (ARN) sumber streaming.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

[RoleARN](#)

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

S3Configuration

Tentukan parameter ini untuk menemukan skema dari data dalam objek Amazon S3.

Tipe: Objek S3Configuration

Wajib: Tidak

Sintaksis Respons

```
{
   "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" ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

InputSchema

Skema yang disimpulkan dari sumber streaming. Ini mengidentifikasi format data dalam sumber streaming, dan cara setiap elemen data dipetakan ke kolom yang sesuai di aliran dalam aplikasi yang dapat Anda buat.

Tipe: Objek SourceSchema

ParsedInputRecords

Array elemen, tempat setiap elemen sesuai dengan baris dalam catatan aliran (catatan aliran dapat memiliki lebih dari satu baris).

Tipe: Array dari array string.

ProcessedInputRecords

Data aliran yang dimodifikasi oleh prosesor yang ditentukan dalam parameter `InputProcessingConfiguration`.

Tipe: Array string

RawInputRecords

Data aliran mentah yang diambil sampel untuk menyimpulkan skema.

Tipe: Array string

Kesalahan

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceProvisionedThroughputExceededException

Discovery gagal mendapatkan rekaman dari sumber streaming karena Amazon ProvisionedThroughputExceededException Kinesis Streams. Untuk informasi selengkapnya, lihat GetRecordsdi Referensi API Amazon Kinesis Streams.

Kode Status HTTP: 400

ServiceUnavailableException

Layanan tidak tersedia. Kembali dan coba lagi operasi.

Kode Status HTTP: 500

UnableToDetectSchemaException

Format data tidak valid. Amazon Kinesis Analytics tidak dapat mendeteksi skema untuk sumber streaming yang diberikan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

ListApplications

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menampilkan daftar aplikasi Amazon Kinesis Analytics di akun Anda. Untuk setiap aplikasi, respons termasuk nama aplikasi, Amazon Resource Name (ARN), dan status. Jika respons mengembalikan nilai `HasMoreApplications` sebagai benar, Anda dapat mengirim permintaan lain dengan menambahkan `ExclusiveStartApplicationName` di isi permintaan, dan menetapkan nilai ini ke nama aplikasi terakhir dari respons sebelumnya.

Jika Anda ingin informasi terperinci tentang aplikasi tertentu, gunakan DescribeApplication.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:ListApplications`.

Sintaksis Permintaan

```
{
   "ExclusiveStartApplicationName": "string",
   "Limit": number
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ExclusiveStartApplicationName

Nama aplikasi untuk memulai daftar. Ketika menggunakan pemberian nomor halaman untuk mendapatkan daftar, Anda tidak perlu menentukan parameter ini dalam permintaan pertama. Namun, dalam permintaan berikutnya, Anda menambahkan nama aplikasi terakhir dari respons sebelumnya untuk mendapatkan halaman aplikasi berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Tidak

Limit

Jumlah maksimum aplikasi yang dicantumkan.

Tipe: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

Wajib: Tidak

Sintaksis Respons

```
{
   "ApplicationSummaries": [
      {
         "ApplicationARN": "string",
         "ApplicationName": "string",
         "ApplicationStatus": "string"
      }
   ],
   "HasMoreApplications": boolean
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

ApplicationSummaries

Daftar objek `ApplicationSummary`.

Tipe: Array objek ApplicationSummary

HasMoreApplications

Mengembalikan benar jika ada lebih banyak aplikasi yang diambil.

Tipe: Boolean

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

ListTagsForResource

Mengambil daftar tanda nilai kunci yang ditetapkan ke aplikasi. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.

Sintaksis Permintaan

```
{
   "ResourceARN": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ResourceARN

ARN aplikasi tempat mengambil tanda.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Sintaksis Respons

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Tags

Tanda nilai kunci yang ditetapkan ke aplikasi.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 200 item.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

StartApplication

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Memulai aplikasi Amazon Kinesis Analytics yang ditentukan. Setelah membuat aplikasi, Anda harus secara eksklusif memanggil operasi ini untuk memulai aplikasi Anda.

Setelah dimulai, aplikasi mulai menggunakan data input, memprosesnya, dan menulis output ke tujuan yang dikonfigurasi.

Status aplikasi harus `READY` bagi Anda untuk memulai aplikasi. Anda bisa mendapatkan status aplikasi di konsol atau menggunakan DescribeApplicationoperasi.

Setelah Anda memulai aplikasi, Anda dapat menghentikan aplikasi dari memproses input dengan memanggil StopApplicationoperasi.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:StartApplication`.

Sintaksis Permintaan

```
{
   "ApplicationName": "string",
   "InputConfigurations": [
      {
         "Id": "string",
         "InputStartingPositionConfiguration": {
            "InputStartingPosition": "string"
         }
      }
   ]
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

InputConfigurations

Mengidentifikasi input tertentu, dengan ID, yang mulai digunakan aplikasi. Amazon Kinesis Analytics mulai membaca sumber streaming yang terkait dengan input. Anda juga dapat menentukan tempat di dalam sumber streaming yang Anda inginkan untuk Amazon Kinesis Analytics mulai membaca.

Tipe: Array objek InputConfiguration

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

InvalidApplicationConfigurationException

Konfigurasi aplikasi yang disediakan pengguna tidak valid.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

StopApplication

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menghentikan aplikasi dari memproses data input. Anda dapat menghentikan aplikasi hanya jika sedang berjalan. Anda dapat menggunakan DescribeApplicationoperasi untuk menemukan status aplikasi. Setelah aplikasi dihentikan, Amazon Kinesis Analytics berhenti membaca data dari input, aplikasi berhenti memproses data, dan tidak ada output yang ditulis ke tujuan.

Operasi ini memerlukan izin untuk menjalankan tindakan `kinesisanalytics:StopApplication`.

Sintaksis Permintaan

```
{
   "ApplicationName": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi yang sedang berjalan yang dihentikan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

TagResource

Menambahkan satu atau beberapa tanda nilai kunci ke aplikasi Kinesis Analytics. Perhatikan bahwa jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.

Sintaksis Permintaan

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

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ResourceARN

ARN aplikasi untuk menetapkan tanda.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

Tags

Tanda nilai kunci yang ditetapkan ke aplikasi.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 200 item.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

TooManyTagsException

Aplikasi yang dibuat dengan terlalu banyak tanda, atau terlalu banyak tanda yang ditambahkan ke aplikasi. Perhatikan bahwa jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

UntagResource

Menghapus satu atau beberapa tanda dari aplikasi Kinesis Analytics. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.

Sintaksis Permintaan

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ResourceARN

ARN aplikasi Kinesis Analytics untuk menghapus tanda.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

TagKeys

Daftar kunci tanda yang dihapus dari aplikasi yang ditentukan.

Tipe: Array string

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 200 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

TooManyTagsException

Aplikasi yang dibuat dengan terlalu banyak tanda, atau terlalu banyak tanda yang ditambahkan ke aplikasi. Perhatikan bahwa jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

UpdateApplication

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Memperbarui aplikasi Amazon Kinesis Analytics yang sudah ada. Dengan menggunakan API ini, Anda dapat memperbarui kode aplikasi, konfigurasi input, dan konfigurasi output.

Perhatikan bahwa Amazon Kinesis Analytics memperbarui `CurrentApplicationVersionId` setiap kali Anda memperbarui aplikasi Anda.

Operasi ini memerlukan izin untuk tindakan `kinesisanalytics:UpdateApplication`.

Sintaksis Permintaan

```
{
   "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
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ApplicationName

Nama aplikasi Amazon Kinesis Analytics yang diperbarui.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

ApplicationUpdate

Menjelaskan pembaruan aplikasi.

Tipe: Objek ApplicationUpdate

Wajib: Ya

CurrentApplicationVersionId

ID versi aplikasi saat ini. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan nilai ini.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

CodeValidationException

Kode aplikasi yang disediakan pengguna (kueri) tidak valid. Ini bisa menjadi kesalahan sintaksis sederhana.

Kode Status HTTP: 400

ConcurrentModificationException

Pengecualian yang dibuat sebagai hasil modifikasi konkuren ke aplikasi. Sebagai contoh, dua individu yang mencoba untuk mengedit aplikasi yang sama pada waktu yang sama.

Kode Status HTTP: 400

InvalidArgumentException

Nilai parameter input yang ditentukan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Aplikasi ini tidak tersedia untuk operasi ini.

Kode Status HTTP: 400

ResourceNotFoundException

Aplikasi yang ditentukan tidak dapat ditemukan.

Kode Status HTTP: 400

UnsupportedOperationException

Permintaan ditolak karena parameter tertentu tidak didukung atau sumber daya tertentu tidak valid untuk operasi ini.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

Tipe Data

tipe data berikut didukung:

- 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**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menyediakan deskripsi aplikasi, termasuk Amazon Resource Name (ARN) aplikasi, status, versi terbaru, dan detail konfigurasi input serta output.

Daftar Isi

ApplicationARN

ARN aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

ApplicationName

Nama aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

ApplicationStatus

Status aplikasi.

Jenis: String

Nilai yang Valid: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

Wajib: Ya

ApplicationVersionId

Menyediakan versi aplikasi saat ini.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 999999999.

Wajib: Ya

ApplicationCode

Mengembalikan kode aplikasi yang Anda berikan untuk melakukan analisis data pada salah satu aliran dalam aplikasi di aplikasi Anda.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 102400.

Wajib: Tidak

ApplicationDescription

Deskripsi aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 1024.

Wajib: Tidak

CloudWatchLoggingOptionDescriptions

Menjelaskan aliran CloudWatch log yang dikonfigurasi untuk menerima pesan aplikasi. Untuk informasi selengkapnya tentang penggunaan aliran CloudWatch log dengan aplikasi Amazon Kinesis Analytics, lihat Bekerja dengan Log Amazon. CloudWatch

Tipe: Array objek CloudWatchLoggingOptionDescription

Wajib: Tidak

CreateTimestamp

Stempel waktu saat versi aplikasi dibuat.

Tipe: Timestamp

Wajib: Tidak

InputDescriptions

Menjelaskan konfigurasi input aplikasi. Untuk informasi selengkapnya, lihat Mengonfigurasi Input Aplikasi.

Tipe: Array objek InputDescription

Wajib: Tidak

LastUpdateTimestamp

Stempel waktu saat aplikasi terakhir diperbarui.

Tipe: Timestamp

Wajib: Tidak

OutputDescriptions

Menjelaskan konfigurasi output aplikasi. Untuk informasi lebih lanjut, lihat Mengonfigurasi Output Aplikasi.

Tipe: Array objek OutputDescription

Wajib: Tidak

ReferenceDataSourceDescriptions

Menjelaskan sumber data referensi yang dikonfigurasi untuk aplikasi. Untuk informasi selengkapnya, lihat Mengonfigurasi Input Aplikasi.

Tipe: Array objek ReferenceDataSourceDescription

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ApplicationSummary

> **Note**
>
> Dokumentasi ini ditujukan untuk API Amazon Kinesis Data Analytics versi 1, yang hanya mendukung aplikasi SQL. API versi 2 mendukung aplikasi SQL dan Java. Untuk informasi selengkapnya tentang versi 2, lihat Dokumentasi V2 API Amazon Kinesis Data Analytics.

Menyediakan informasi ringkasan aplikasi, termasuk Amazon Resource Name (ARN) aplikasi, nama, dan status.

Daftar Isi

ApplicationARN

ARN aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

ApplicationName

Nama aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

ApplicationStatus

Status aplikasi.

Jenis: String

Nilai yang Valid: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ApplicationUpdate

Menjelaskan pembaruan yang diterapkan pada aplikasi Amazon Kinesis Analytics yang sudah ada.

Daftar Isi

ApplicationCodeUpdate

Menjelaskan pembaruan kode aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 102400.

Wajib: Tidak

CloudWatchLoggingOptionUpdates

Menjelaskan pembaruan opsi CloudWatch pencatatan aplikasi.

Tipe: Array objek CloudWatchLoggingOptionUpdate

Wajib: Tidak

InputUpdates

Menjelaskan pembaruan konfigurasi input aplikasi.

Tipe: Array objek InputUpdate

Wajib: Tidak

OutputUpdates

Menjelaskan pembaruan konfigurasi output aplikasi.

Tipe: Array objek OutputUpdate

Wajib: Tidak

ReferenceDataSourceUpdates

Menjelaskan pembaruan sumber data referensi aplikasi.

Tipe: Array objek ReferenceDataSourceUpdate

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOption

Memberikan deskripsi opsi CloudWatch logging, termasuk aliran log Amazon Resource Name (ARN) dan peran ARN.

Daftar Isi

LogStreamARN

ARN dari CloudWatch log untuk menerima pesan aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM peran yang digunakan untuk mengirim pesan aplikasi. Catatan: Untuk menulis pesan aplikasi CloudWatch, peran IAM yang digunakan harus mengaktifkan tindakan `PutLogEvents` kebijakan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOptionDescription

Deskripsi opsi CloudWatch logging.

Daftar Isi

LogStreamARN

ARN dari CloudWatch log untuk menerima pesan aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM peran yang digunakan untuk mengirim pesan aplikasi. Catatan: Untuk menulis pesan aplikasi CloudWatch, peran IAM yang digunakan harus mengaktifkan tindakan `PutLogEvents` kebijakan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

CloudWatchLoggingOptionId

ID deskripsi opsi CloudWatch logging.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOptionUpdate

Menjelaskan pembaruan opsi CloudWatch logging.

Daftar Isi

CloudWatchLoggingOptionId

ID opsi CloudWatch logging untuk memperbarui

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

LogStreamARNUpdate

ARN dari CloudWatch log untuk menerima pesan aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN IAM peran yang digunakan untuk mengirim pesan aplikasi. Catatan: Untuk menulis pesan aplikasi CloudWatch, peran IAM yang digunakan harus mengaktifkan tindakan `PutLogEvents` kebijakan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CSVMappingParameters

Menyediakan informasi pemetaan tambahan saat format catatan menggunakan pembatas, seperti CSV. Misalnya, catatan sampel berikut menggunakan format CSV, di mana catatan menggunakan '\n' sebagai pembatas baris dan koma (“,”) sebagai pembatas kolom:

```
"name1", "address1"

"name2", "address2"
```

Daftar Isi

RecordColumnDelimiter

Pembatas kolom. Misalnya, dalam format CSV, koma (",") adalah pembatas kolom yang khas.

Jenis: String

Batasan Panjang: Panjang minimum 1.

Wajib: Ya

RecordRowDelimiter

Pembatas baris. Misalnya, dalam format CSV, '\n' adalah pembatas baris yang khas.

Jenis: String

Batasan Panjang: Panjang minimum 1.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DestinationSchema

Menjelaskan format data saat catatan ditulis ke tujuan. Untuk informasi lebih lanjut, lihat Mengonfigurasi Output Aplikasi.

Daftar Isi

RecordFormatType

Menentukan format catatan pada aliran output.

Jenis: String

Nilai yang Valid: `JSON | CSV`

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Input

Ketika Anda mengonfigurasi input aplikasi, Anda menentukan sumber streaming, nama aliran dalam aplikasi yang dibuat, dan pemetaan di antara keduanya. Untuk informasi lebih lanjut, lihat Pengonfigurasian Input Aplikasi.

Daftar Isi

InputSchema

Menjelaskan format data dalam sumber streaming, dan cara setiap elemen data memetakan ke kolom yang sesuai di pengaliran dalam aplikasi yang dibuat.

Juga digunakan untuk menjelaskan format sumber data referensi.

Tipe: Objek SourceSchema

Wajib: Ya

NamePrefix

Prefiks nama untuk digunakan saat membuat pengaliran dalam aplikasi. Misalkan Anda menentukan awalan "MyInApplicationStream.” Amazon Kinesis Analytics kemudian membuat satu atau lebih (sesuai jumlah `InputParallelism` yang Anda tentukan) aliran dalam aplikasi dengan nama MyInApplicationStream "_001,” “_002,” MyInApplicationStream dan seterusnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Ya

InputParallelism

Menjelaskan jumlah aliran dalam aplikasi yang dibuat.

Data dari sumber Anda dirutekan ke aliran input dalam aplikasi ini.

(Lihat Mengonfigurasi Input Aplikasi.

Tipe: Objek InputParallelism

Wajib: Tidak

InputProcessingConfiguration

InputProcessingConfigurationUntuk input. Prosesor input mengubah catatan seperti yang diterima dari pengaliran, sebelum dieksekusi oleh kode SQL aplikasi. Saat ini, satu-satunya konfigurasi pemrosesan input yang tersedia adalah InputLambdaProcessor.

Tipe: Objek InputProcessingConfiguration

Wajib: Tidak

KinesisFirehoseInput

Jika sumber streaming adalah aliran pengiriman Amazon Kinesis Firehose, ia mengidentifikasi ARN aliran pengiriman dan IAM role yang memungkinkan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda.

Catatan: `KinesisStreamsInput` atau `KinesisFirehoseInput` diperlukan.

Tipe: Objek KinesisFirehoseInput

Wajib: Tidak

KinesisStreamsInput

Jika sumber streaming adalah pengaliran Amazon Kinesis, ia mengidentifikasi Amazon Resource Name (ARN) pengaliran dan IAM role yang memungkinkan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda.

Catatan: `KinesisStreamsInput` atau `KinesisFirehoseInput` diperlukan.

Tipe: Objek KinesisStreamsInput

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputConfiguration

Ketika Anda memulai aplikasi Anda, Anda menyediakan konfigurasi ini, yang mengidentifikasi sumber input dan titik dalam sumber input tempat Anda ingin aplikasi mulai memproses catatan.

Daftar Isi

Id

ID sumber input. Anda bisa mendapatkan ID ini dengan memanggil DescribeApplicationoperasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

InputStartingPositionConfiguration

Titik tempat Anda ingin aplikasi mulai memproses catatan dari sumber streaming.

Tipe: Objek InputStartingPositionConfiguration

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputDescription

Menjelaskan konfigurasi input aplikasi. Untuk informasi selengkapnya, lihat Mengonfigurasi Input Aplikasi.

Daftar Isi

InAppStreamNames

Mengembalikan nama aliran dalam aplikasi yang dipetakan ke sumber aliran.

Tipe: Array string

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Tidak

InputId

ID input yang terkait dengan input aplikasi. Ini adalah ID yang diberikan Amazon Kinesis Analytics ke setiap konfigurasi input yang Anda tambahkan ke aplikasi Anda.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Tidak

InputParallelism

Menjelaskan paralelisme yang dikonfigurasi (jumlah aliran dalam aplikasi yang dipetakan ke sumber streaming).

Tipe: Objek InputParallelism

Wajib: Tidak

InputProcessingConfigurationDescription

Deskripsi prapemrosesan yang menjalankan catatan dalam input ini sebelum kode aplikasi dijalankan.

Tipe: Objek InputProcessingConfigurationDescription

Wajib: Tidak

InputSchema

Menjelaskan format data dalam sumber streaming, dan cara setiap elemen data memetakan ke kolom yang sesuai di pengaliran dalam aplikasi yang dibuat.

Tipe: Objek SourceSchema

Wajib: Tidak

InputStartingPositionConfiguration

Titik tempat aplikasi dikonfigurasi untuk membaca dari input aliran.

Tipe: Objek InputStartingPositionConfiguration

Wajib: Tidak

KinesisFirehoseInputDescription

Jika aliran pengiriman Amazon Kinesis Firehose dikonfigurasi sebagai sumber streaming, menyediakan ARN aliran pengiriman dan IAM role yang memungkinkan Amazon Kinesis Analytics mengakses aliran atas nama Anda.

Tipe: Objek KinesisFirehoseInputDescription

Wajib: Tidak

KinesisStreamsInputDescription

Jika Amazon Kinesis stream dikonfigurasi sebagai sumber streaming, menyediakan Amazon Resource Name (ARN) Amazon Kinesis stream dan IAM role yang memungkinkan Amazon Kinesis Analytics mengakses aliran atas nama Anda.

Tipe: Objek KinesisStreamsInputDescription

Wajib: Tidak

NamePrefix

Prefiks nama dalam aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessor

Objek yang berisi Nama Sumber Daya Amazon (ARN) dari fungsi AWS Lambda yang digunakan untuk memproses catatan dalam aliran, dan ARN peran IAM yang digunakan untuk mengakses fungsi Lambda. AWS

Daftar Isi

ResourceARN

ARN dari fungsi AWS Lambda yang beroperasi pada catatan di aliran.

> **Note**
>
> Untuk menentukan versi sebelumnya dari fungsi Lambda yang terbaru, sertakan versi fungsi Lambda dalam ARN fungsi Lambda. Untuk informasi selengkapnya tentang ARN Lambda, lihat Contoh ARN: Lambda AWS

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN dari peran IAM yang digunakan untuk mengakses fungsi Lambda AWS .

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessorDescription

Objek yang berisi Nama Sumber Daya Amazon (ARN) dari fungsi AWS Lambda yang digunakan untuk memproses catatan dalam aliran, dan ARN peran IAM yang digunakan untuk mengakses ekspresi Lambda. AWS

Daftar Isi

ResourceARN

ARN dari fungsi AWS Lambda yang digunakan untuk memproses catatan dalam aliran.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARN

ARN dari peran IAM yang digunakan untuk mengakses fungsi Lambda AWS .

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessorUpdate

Merupakan pembaruan untuk InputLambdaProcessoryang digunakan untuk memproses catatan dalam aliran.

Daftar Isi

ResourceARNUpdate

Nama Sumber Daya Amazon (ARN) dari fungsi AWS Lambda baru yang digunakan untuk memproses catatan dalam aliran.

> **Note**
>
> Untuk menentukan versi fungsi Lambda sebelumnya dari yang terbaru, sertakan versi fungsi Lambda dalam ARN fungsi Lambda. Untuk informasi selengkapnya tentang ARN Lambda, lihat Contoh ARN: Lambda AWS

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN dari peran IAM baru yang digunakan untuk mengakses fungsi Lambda AWS .

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputParallelism

Menjelaskan jumlah aliran dalam aplikasi yang harus diberikan untuk sumber streaming tertentu. Untuk informasi tentang paralelisme, lihat Mengonfigurasi Input Aplikasi.

Daftar Isi

Count

Jumlah aliran dalam aplikasi yang harus dibuat. Untuk informasi selengkapnya, lihat Batasan-batasan.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 64.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputParallelismUpdate

Menyediakan pembaruan untuk jumlah paralelisme.

Daftar Isi

CountUpdate

Jumlah aliran dalam aplikasi yang dibuat untuk sumber streaming tertentu.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 64.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfiguration

Memberikan deskripsi prosesor yang digunakan untuk pra-pemrosesan catatan di aliran sebelum diproses oleh kode aplikasi Anda. Saat ini, satu-satunya prosesor input yang tersedia adalah AWS Lambda.

Daftar Isi

InputLambdaProcessor

InputLambdaProcessorYang digunakan untuk memproses catatan dalam aliran sebelum diproses oleh kode aplikasi Anda.

Tipe: Objek InputLambdaProcessor

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfigurationDescription

Menyediakan informasi konfigurasi tentang prosesor input. Saat ini, satu-satunya prosesor input yang tersedia adalah AWS Lambda.

Daftar Isi

InputLambdaProcessorDescription

Memberikan informasi konfigurasi tentang yang terkait InputLambdaProcessorDescription.

Tipe: Objek InputLambdaProcessorDescription

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfigurationUpdate

Menjelaskan pembaruan ke sebuah InputProcessingConfiguration.

Daftar Isi

InputLambdaProcessorUpdate

Memberikan informasi pembaruan untuk sebuah InputLambdaProcessor.

Tipe: Objek InputLambdaProcessorUpdate

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputSchemaUpdate

Menjelaskan pembaruan untuk skema input aplikasi.

Daftar Isi

RecordColumnUpdates

Daftar objek `RecordColumn`. Setiap objek menjelaskan pemetaan elemen sumber streaming ke kolom yang sesuai di aliran dalam aplikasi.

Tipe: Array objek RecordColumn

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 1000 item.

Wajib: Tidak

RecordEncodingUpdate

Menentukan pengkodean catatan dalam sumber streaming. Misalnya, UTF-8.

Jenis: String

Pola: `UTF-8`

Wajib: Tidak

RecordFormatUpdate

Menentukan format catatan pada sumber streaming.

Tipe: Objek RecordFormat

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputStartingPositionConfiguration

Menjelaskan titik tempat aplikasi membaca dari sumber streaming.

Daftar Isi

InputStartingPosition

Posisi awal di aliran.

- `NOW` - Mulai membaca setelah catatan terbaru di aliran, memulai di stempel waktu permintaan yang dikeluarkan pelanggan.
- `TRIM_HORIZON` - Mulai membaca di catatan yang tidak dipangkas terakhir di aliran, yang merupakan catatan terlama yang tersedia di aliran. Opsi ini tidak tersedia untuk aliran pengiriman Amazon Kinesis Firehose.
- `LAST_STOPPED_POINT` - Melanjutkan membaca dari tempat aplikasi terakhir berhenti membaca.

Jenis: String

Nilai yang Valid: `NOW | TRIM_HORIZON | LAST_STOPPED_POINT`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputUpdate

Menjelaskan pembaruan pada konfigurasi input tertentu (diidentifikasi oleh `InputId` aplikasi).

Daftar Isi

InputId

ID input dari input aplikasi yang akan diperbarui.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

InputParallelismUpdate

Menjelaskan pembaruan paralelisme (jumlah aliran dalam aplikasi yang dibuat Amazon Kinesis Analytics untuk sumber streaming tertentu).

Tipe: Objek InputParallelismUpdate

Wajib: Tidak

InputProcessingConfigurationUpdate

Menjelaskan pembaruan untuk konfigurasi pemrosesan input.

Tipe: Objek InputProcessingConfigurationUpdate

Wajib: Tidak

InputSchemaUpdate

Menjelaskan format data pada sumber streaming, dan cara elemen catatan di sumber streaming dipetakan ke kolom aliran dalam aplikasi yang dibuat.

Tipe: Objek InputSchemaUpdate

Wajib: Tidak

KinesisFirehoseInputUpdate

Jika aliran pengiriman Amazon Kinesis Firehose adalah sumber streaming yang akan diperbarui, sediakan ARN aliran yang diperbarui dan ARN IAM role.

Tipe: Objek KinesisFirehoseInputUpdate

Wajib: Tidak

KinesisStreamsInputUpdate

Jika Amazon Kinesis stream adalah sumber streaming yang akan diperbarui, sediakan Amazon Resource Name (ARN) aliran yang diperbarui dengan ARN IAM role.

Tipe: Objek KinesisStreamsInputUpdate

Wajib: Tidak

NamePrefixUpdate

Prefiks nama untuk aliran dalam aplikasi yang dibuat Amazon Kinesis Analytics untuk sumber streaming tertentu.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

JSONMappingParameters

Memberikan informasi pemetaan tambahan ketika JSON adalah format rekaman pada sumber streaming.

Daftar Isi

RecordRowPath

Jalur ke induk tingkat teratas yang berisi catatan.

Jenis: String

Batasan Panjang: Panjang minimum 1.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInput

Mengidentifikasi pengaliran pengiriman Amazon Kinesis Firehose sebagai sumber streaming. Anda menyediakan Amazon Resource Name (ARN) pengaliran pengiriman dan ARN IAM role yang mengizinkan Amazon Kinesis Analytics mengakses streaming atas nama Anda.

Daftar Isi

ResourceARN

ARN aliran pengiriman input.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda. Anda harus memastikan peran memiliki izin yang diperlukan untuk mengakses aliran.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

KinesisFirehoseInputDescription

Menjelaskan aliran pengiriman Amazon Kinesis Firehose yang dikonfigurasi sebagai sumber streaming dalam konfigurasi input aplikasi.

Daftar Isi

ResourceARN

Amazon Resource Name (ARN) aliran pengiriman Amazon Kinesis Firehose.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARN

ARN IAM role yang diambil Amazon Kinesis Analytics untuk mengakses aliran.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInputUpdate

Saat memperbarui konfigurasi input aplikasi, berikan informasi tentang aliran pengiriman Amazon Kinesis Firehose sebagai sumber streaming.

Daftar Isi

ResourceARNUpdate

Amazon Resource Name (ARN) aliran pengiriman Amazon Kinesis Firehose input yang dibaca.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseOutput

Saat mengonfigurasi output aplikasi, mengidentifikasi pengaliran pengiriman Amazon Kinesis Firehose sebagai tujuan. Anda menyediakan pengaliran Amazon Resource Name (ARN) dan IAM role yang memungkinkan Amazon Kinesis Analytics menulis ke pengaliran atas nama Anda.

Daftar Isi

ResourceARN

ARN aliran pengiriman Amazon Kinesis Firehose tujuan yang ditulis.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk menulis ke pengaliran tujuan atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseOutputDescription

Untuk output aplikasi, jelaskan aliran pengiriman Amazon Kinesis Firehose yang dikonfigurasi sebagai tujuannya.

Daftar Isi

ResourceARN

Amazon Resource Name (ARN) aliran pengiriman Amazon Kinesis Firehose.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARN

ARN IAM role yang dapat diambil Amazon Kinesis Analytics untuk mengakses aliran.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseOutputUpdate

Saat memperbarui konfigurasi keluaran menggunakan UpdateApplicationoperasi, berikan informasi tentang aliran pengiriman Amazon Kinesis Firehose yang dikonfigurasi sebagai tujuan.

Daftar Isi

ResourceARNUpdate

Amazon Resource Name (ARN) aliran pengiriman Amazon Kinesis Firehose yang ditulis.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsInput

Mengidentifikasi pengaliran Amazon Kinesis sebagai sumber streaming. Anda memberikan Amazon Resource Name (ARN) dan ARN IAM role yang mengizinkan Amazon Kinesis Analytics mengakses pengaliran atas nama Anda.

Daftar Isi

ResourceARN

ARN aliran Amazon Kinesis input yang dibaca.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsInputDescription

Menjelaskan Amazon Kinesis stream yang dikonfigurasi sebagai sumber streaming dalam konfigurasi input aplikasi.

Daftar Isi

ResourceARN

Amazon Resource Name (ARN) Amazon Kinesis stream.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARN

ARN IAM role yang dapat diambil Amazon Kinesis Analytics untuk mengakses aliran.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsInputUpdate

Saat memperbarui konfigurasi input aplikasi, berikan informasi tentang Amazon Kinesis stream sebagai sumber streaming.

Daftar Isi

ResourceARNUpdate

Amazon Resource Name (ARN) Amazon Kinesis stream input yang dibaca.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutput

Saat mengonfigurasi output aplikasi, mengidentifikasi pengaliran Amazon Kinesis sebagai tujuan. Anda memberikan pengaliran Amazon Resource Name (ARN) dan juga ARN IAM role yang dapat digunakan Amazon Kinesis Analytics untuk menulis ke pengaliran atas nama Anda.

Daftar Isi

ResourceARN

ARN Amazon Kinesis stream tujuan yang ditulis.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk menulis ke pengaliran tujuan atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutputDescription

Untuk output aplikasi, jelaskan Amazon Kinesis stream yang dikonfigurasi sebagai tujuannya.

Daftar Isi

ResourceARN

Amazon Resource Name (ARN) Amazon Kinesis stream.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARN

ARN IAM role yang dapat diambil Amazon Kinesis Analytics untuk mengakses aliran.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutputUpdate

Saat memperbarui konfigurasi output menggunakan UpdateApplicationoperasi, berikan informasi tentang aliran Amazon Kinesis yang dikonfigurasi sebagai tujuan.

Daftar Isi

ResourceARNUpdate

Amazon Resource Name (ARN) Amazon Kinesis stream tempat Anda ingin menulis output.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk mengakses pengaliran atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutput

Saat mengonfigurasi output aplikasi, mengidentifikasi fungsi AWS Lambda sebagai tujuan. Anda menyediakan fungsi Amazon Resource Name (ARN) dan juga IAM role yang dapat digunakann Amazon Kinesis Analytics untuk menulis ke fungsi atas nama Anda.

Daftar Isi

ResourceARN

Amazon Resource Name (ARN) fungsi Lambda tujuan yang akan ditulis.

> **Note**
>
> Untuk menentukan versi sebelumnya dari fungsi Lambda yang terbaru, sertakan versi fungsi Lambda dalam ARN fungsi Lambda. Untuk informasi selengkapnya tentang ARN Lambda, lihat Contoh ARN: Lambda AWS

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

RoleARN

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk menulis ke fungsi tujuan atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutputDescription

Untuk keluaran aplikasi, jelaskan fungsi AWS Lambda yang dikonfigurasi sebagai tujuannya.

Daftar Isi

ResourceARN

Amazon Resource Name (ARN) fungsi Lambda tujuan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARN

ARN IAM role yang dapat diambil Amazon Kinesis Analytics untuk ditulis ke fungsi tujuan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutputUpdate

Saat memperbarui konfigurasi output menggunakan UpdateApplicationoperasi, berikan informasi tentang fungsi AWS Lambda yang dikonfigurasi sebagai tujuan.

Daftar Isi

ResourceARNUpdate

Amazon Resource Name (ARN) fungsi Lambda tujuan.

> **Note**
>
> Untuk menentukan versi sebelumnya dari fungsi Lambda yang terbaru, sertakan versi fungsi Lambda dalam ARN fungsi Lambda. Untuk informasi selengkapnya tentang ARN Lambda, lihat Contoh ARN: Lambda AWS

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

RoleARNUpdate

ARN IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk menulis ke fungsi tujuan atas nama Anda. Anda perlu memberikan izin yang diperlukan untuk peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MappingParameters

Ketika mengonfigurasi input aplikasi pada saat membuat atau memperbarui aplikasi, memberikan informasi pemetaan tambahan tertentu untuk format catatan (seperti JSON, CSV, atau bidang catatan yang dibatasi oleh beberapa pembatas) pada sumber streaming.

Daftar Isi

CSVMappingParameters

Menyediakan informasi pemetaan tambahan saat format catatan menggunakan pembatas (misalnya, CSV).

Tipe: Objek CSVMappingParameters

Wajib: Tidak

JSONMappingParameters

Memberikan informasi pemetaan tambahan ketika JSON adalah format rekaman pada sumber streaming.

Tipe: Objek JSONMappingParameters

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Output

Menjelaskan konfigurasi output aplikasi di mana Anda mengidentifikasi aliran dalam aplikasi dan tujuan di mana Anda akan menulis data aliran dalam aplikasi. Tujuan dapat berupa aliran Amazon Kinesis atau aliran pengiriman Amazon Kinesis Firehose.

Untuk batas berapa banyak tujuan yang dapat ditulis aplikasi dan batasan lainnya, lihat Batasan.

Daftar Isi

DestinationSchema

Menjelaskan format data saat catatan ditulis ke tujuan. Untuk informasi lebih lanjut, lihat Mengonfigurasi Output Aplikasi.

Tipe: Objek DestinationSchema

Wajib: Ya

Name

Namai aliran dalam aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Ya

KinesisFirehoseOutput

Mengidentifikasi aliran pengiriman Amazon Kinesis Firehose sebagai tujuan.

Tipe: Objek KinesisFirehoseOutput

Wajib: Tidak

KinesisStreamsOutput

Mengidentifikasi aliran Amazon Kinesis sebagai tujuan.

Tipe: Objek KinesisStreamsOutput

Wajib: Tidak

LambdaOutput

Mengidentifikasi fungsi AWS Lambda sebagai tujuan.

Tipe: Objek LambdaOutput

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputDescription

Menjelaskan konfigurasi output aplikasi, yang meliputi nama aliran dalam aplikasi dan tujuan tempat data aliran ditulis. Tujuan dapat berupa aliran Amazon Kinesis atau aliran pengiriman Amazon Kinesis Firehose.

Daftar Isi

DestinationSchema

Format data yang digunakan untuk menulis data ke tujuan.

Tipe: Objek DestinationSchema

Wajib: Tidak

KinesisFirehoseOutputDescription

Menjelaskan aliran pengiriman Amazon Kinesis Firehose yang dikonfigurasi sebagai tujuan tempat output ditulis.

Tipe: Objek KinesisFirehoseOutputDescription

Wajib: Tidak

KinesisStreamsOutputDescription

Menjelaskan aliran Amazon Kinesis yang dikonfigurasi sebagai tujuan tempat output ditulis.

Tipe: Objek KinesisStreamsOutputDescription

Wajib: Tidak

LambdaOutputDescription

Menjelaskan fungsi AWS Lambda yang dikonfigurasi sebagai tujuan di mana output ditulis.

Tipe: Objek LambdaOutputDescription

Wajib: Tidak

Name

Nama aliran dalam aplikasi yang dikonfigurasi sebagai output.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Tidak

OutputId

Pengidentifikasi unik untuk konfigurasi output.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputUpdate

Menjelaskan pembaruan ke konfigurasi output yang diidentifikasi oleh `OutputId`.

Daftar Isi

OutputId

Mengidentifikasi konfigurasi output tertentu yang ingin Anda perbarui.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

DestinationSchemaUpdate

Menjelaskan format data saat catatan ditulis ke tujuan. Untuk informasi lebih lanjut, lihat Mengonfigurasi Output Aplikasi.

Tipe: Objek DestinationSchema

Wajib: Tidak

KinesisFirehoseOutputUpdate

Menjelaskan aliran pengiriman Amazon Kinesis Firehose sebagai tujuan untuk output.

Tipe: Objek KinesisFirehoseOutputUpdate

Wajib: Tidak

KinesisStreamsOutputUpdate

Menjelaskan aliran Amazon Kinesis sebagai tujuan untuk output.

Tipe: Objek KinesisStreamsOutputUpdate

Wajib: Tidak

LambdaOutputUpdate

Menjelaskan fungsi AWS Lambda sebagai tujuan untuk output.

Tipe: Objek LambdaOutputUpdate

Wajib: Tidak

NameUpdate

Jika Anda ingin menentukan aliran dalam aplikasi yang berbeda untuk konfigurasi output ini, gunakan bidang ini untuk menentukan nama aliran dalam aplikasi baru.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecordColumn

Menjelaskan pemetaan setiap elemen data di sumber streaming ke kolom yang sesuai di aliran dalam aplikasi.

Juga digunakan untuk menjelaskan format sumber data referensi.

Daftar Isi

Name

Nama kolom yang dibuat di aliran input dalam aplikasi atau tabel referensi.

Tipe: String

Diperlukan: Ya

SqlType

Tipe kolom yang dibuat di aliran input dalam aplikasi atau tabel referensi.

Jenis: String

Batasan Panjang: Panjang minimum 1.

Wajib: Ya

Mapping

Referensi ke elemen data dalam input streaming atau sumber data referensi. Elemen ini diperlukan jika RecordFormatTypeada`JSON`.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

RecordFormat

Menjelaskan format catatan dan informasi pemetaan yang relevan yang harus diterapkan untuk skematisasi catatan pada aliran.

Daftar Isi

RecordFormatType

Tipe format catatan.

Jenis: String

Nilai yang Valid: `JSON | CSV`

Wajib: Ya

MappingParameters

Ketika mengonfigurasi input aplikasi pada saat membuat atau memperbarui aplikasi, memberikan informasi pemetaan tambahan tertentu untuk format catatan (seperti JSON, CSV, atau bidang catatan yang dibatasi oleh beberapa pembatas) pada sumber streaming.

Tipe: Objek MappingParameters

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSource

Menjelaskan sumber data referensi dengan menyediakan informasi sumber (nama bucket S3 dan nama kunci objek), nama tabel dalam aplikasi yang dihasilkan, dan skema yang diperlukan untuk memetakan elemen data di objek Amazon S3 ke tabel dalam aplikasi.

Daftar Isi

ReferenceSchema

Menjelaskan format data dalam sumber streaming, dan cara setiap elemen data memetakan ke kolom yang sesuai yang dibuat di aliran dalam aplikasi.

Tipe: Objek SourceSchema

Wajib: Ya

TableName

Nama tabel dalam aplikasi yang dibuat.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Ya

S3ReferenceDataSource

Mengidentifikasi bucket S3 dan objek yang berisi data referensi. Juga mengidentifikasi IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk membaca objek ini atas nama Anda. Aplikasi Amazon Kinesis Analytics memuat data referensi satu kali saja. Jika data berubah, Anda memanggil operasi `UpdateApplication` untuk memicu pemuatan ulang data ke dalam aplikasi Anda.

Tipe: Objek S3ReferenceDataSource

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSourceDescription

Menjelaskan sumber data referensi yang dikonfigurasi untuk aplikasi.

Daftar Isi

ReferenceId

ID dari sumber data referensi. Ini adalah ID yang diberikan Amazon Kinesis Analytics saat Anda menambahkan sumber data referensi ke aplikasi menggunakan AddApplicationReferenceDataSourceoperasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

S3ReferenceDataSourceDescription

Menyediakan nama bucket S3, nama kunci objek yang berisi data referensi. Ini juga menyediakan Amazon Resource Name (ARN) IAM role yang dapat diambil Amazon Kinesis Analytics untuk membaca objek Amazon S3 dan mengisi tabel referensi dalam aplikasi.

Tipe: Objek S3ReferenceDataSourceDescription

Wajib: Ya

TableName

Nama tabel dalam aplikasi yang dibuat dengan konfigurasi sumber data referensi tertentu.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Ya

ReferenceSchema

Menjelaskan format data dalam sumber streaming, dan cara setiap elemen data memetakan ke kolom yang sesuai yang dibuat di aliran dalam aplikasi.

Tipe: Objek SourceSchema

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSourceUpdate

Ketika Anda memperbarui konfigurasi sumber data referensi untuk aplikasi, objek ini menyediakan semua nilai yang diperbarui (seperti nama bucket sumber dan nama objek kunci), nama tabel dalam aplikasi yang dibuat, dan informasi pemetaan yang diperbarui yang memetakan data dalam objek Amazon S3 ke ke tabel referensi dalam aplikasi yang dibuat.

Daftar Isi

ReferenceId

ID sumber data referensi yang diperbarui. Anda dapat menggunakan DescribeApplicationoperasi untuk mendapatkan nilai ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

ReferenceSchemaUpdate

Menjelaskan format data dalam sumber streaming, dan cara setiap elemen data memetakan ke kolom yang sesuai yang dibuat di aliran dalam aplikasi.

Tipe: Objek SourceSchema

Wajib: Tidak

S3ReferenceDataSourceUpdate

Menjelaskan nama bucket S3, nama kunci objek, dan IAM role yang dapat diambil Amazon Kinesis Analytics untuk membaca objek Amazon S3 atas nama Anda dan mengisi tabel referensi dalam aplikasi.

Tipe: Objek S3ReferenceDataSourceUpdate

Wajib: Tidak

TableNameUpdate

Nama tabel dalam aplikasi yang dibuat dengan pembaruan ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3Configuration

Menyediakan deskripsi sumber data Amazon S3, termasuk Amazon Resource Name (ARN) bucket S3, ARN IAM role yang digunakan untuk mengakses bucket, dan nama objek Amazon S3 yang berisi data.

Daftar Isi

BucketARN

ARN bucket S3 yang berisi data.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

FileKey

Nama objek yang berisi data.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Wajib: Ya

RoleARN

ARN IAM peran yang digunakan untuk mengakses data.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSource

Mengidentifikasi bucket S3 dan objek yang berisi data referensi. Juga mengidentifikasi IAM role yang dapat diasumsikan Amazon Kinesis Analytics untuk membaca objek ini atas nama Anda.

Aplikasi Amazon Kinesis Analytics memuat data referensi satu kali saja. Jika data berubah, Anda memanggil UpdateApplicationoperasi untuk memicu pemuatan ulang data ke dalam aplikasi Anda.

Daftar Isi

BucketARN

Amazon Resource Name (ARN) dari bucket S3.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

FileKey

Nama kunci objek yang berisi data referensi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Wajib: Ya

ReferenceRoleARN

ARN IAM role yang dapat diasumsikan layanan untuk membaca data atas nama Anda. Peran ini harus memiliki izin untuk tindakan `s3:GetObject` pada objek dan kebijakan kepercayaan yang memungkinkan prinsipal layanan Amazon Kinesis Analytics untuk mengasumsikan peran ini.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSourceDescription

Menyediakan nama bucket dan nama kunci objek yang menyimpan data referensi.

Daftar Isi

BucketARN

Amazon Resource Name (ARN) dari bucket S3.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Ya

FileKey

Nama kunci objek Amazon S3.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Wajib: Ya

ReferenceRoleARN

ARN IAM role yang dapat diambil Amazon Kinesis Analytics untuk membaca objek Amazon S3 atas nama Anda untuk mengisi tabel referensi dalam aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSourceUpdate

Menjelaskan nama bucket S3, nama kunci objek, dan IAM role yang dapat diambil Amazon Kinesis Analytics untuk membaca objek Amazon S3 atas nama Anda dan mengisi tabel referensi dalam aplikasi.

Daftar Isi

BucketARNUpdate

Amazon Resource Name (ARN) dari bucket S3.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Wajib: Tidak

FileKeyUpdate

Nama kunci objek.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Wajib: Tidak

ReferenceRoleARNUpdate

ARN IAM role yang dapat diambil Amazon Kinesis Analytics untuk membaca objek Amazon S3 dan mengisi dalam aplikasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `arn:.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SourceSchema

Menjelaskan format data dalam sumber streaming, dan cara setiap elemen data memetakan ke kolom yang sesuai yang dibuat di pengaliran dalam aplikasi.

Daftar Isi

RecordColumns

Daftar objek `RecordColumn`.

Tipe: Array objek RecordColumn

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 1000 item.

Wajib: Ya

RecordFormat

Menentukan format catatan pada sumber streaming.

Tipe: Objek RecordFormat

Wajib: Ya

RecordEncoding

Menentukan pengkodean catatan dalam sumber streaming. Misalnya, UTF-8.

Jenis: String

Pola: `UTF-8`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Tag

Pasangan kunci-nilai (nilainya opsional) yang dapat Anda tentukan dan tetapkan ke sumber daya. AWS Jika Anda menentukan tanda yang sudah ada, nilai tanda diganti dengan nilai yang Anda tentukan dalam permintaan. Perhatikan bahwa jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.

Daftar Isi

Key

Nilai dari tanda nilai kunci.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

Value

Nilai dari tanda nilai kunci. Nilai ini bersifat opsional.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Riwayat Dokumen untuk Amazon Kinesis Data Analytics

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir Amazon Kinesis Data Analytics.

- Versi API: 2015-08-14
- Pembaruan dokumentasi terbaru: 8 Mei 2019

Perubahan	Deskripsi	Tanggal
Menandai Aplikasi Kinesis Data Analytics	Gunakan penandaan aplikasi untuk menentukan biaya per aplikasi, kontrol akses, atau untuk tujuan yang ditetapkan pengguna. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.	8 Mei 2019
Mencatat Panggilan API Kinesis Data Analytics dengan AWS CloudTrail	Amazon Kinesis Data Analytics terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di Kinesis Data Analytics. Untuk informasi selengkapnya, lihat Menggunakan AWS CloudTrail.	22 Maret 2019
Kinesis Data Analytics tersedia di wilayah Frankfurt	Kinesis Analytics kini tersedia di wilayah Europe (Frankfurt) Region. Untuk informasi selengkapnya, lihatdan Endpoint: Kinesis Data Analytics.	18 Juli 2018

Perubahan	Deskripsi	Tanggal
Gunakan data referensi di konsol	Anda sekarang dapat bekerja dengan data referensi aplikasi di konsol. Untuk informasi selengkapnya, lihat Contoh: Menambahkan Data Referensi ke Aplikasi Kinesis Data Analytics .	13 Juli 2018
Contoh kueri jendela	Contoh aplikasi untuk windows dan agregasi. Untuk informasi selengkapnya, lihat Contoh: Jendela dan Agregasi.	9 Juli 2018
Menguji aplikasi	Panduan pengujian perubahan skema dan kode aplikasi. Untuk informasi selengkapnya, lihat Menguji Aplikasi.	3 Juli 2018
Contoh aplikasi untuk prapemrosesan data	Sampel kode tambahan untuk operator REGEX_LOG _PARSE, REGEX_REP LACE, dan DateTime. Untuk informasi selengkapnya, lihat Contoh: Mengubah Data.	18 Mei 2018
Meningkatkan ukuran baris yang ditampilkan dan kode SQL	Batas ukuran untuk baris yang ditampilkan meningkat menjadi 512 KB, dan batas untuk ukuran kode SQL dalam aplikasi meningkat menjadi 100 KB. Untuk informasi selengkapnya, lihat Batas.	2 Mei 2018

Perubahan	Deskripsi	Tanggal
Contoh fungsi AWS Lambda di Java dan .NET	Sampel kode untuk membuat fungsi Lambda untuk prapemrosesan catatan dan untuk tujuan aplikasi. Untuk informasi selengkapnya, lihat Membuat Fungsi Lambda untuk Prapemrosesan dan Membuat Fungsi Lambda untuk Tujuan Aplikasi.	22 Maret 2018
Fungsi `HOTSPOTS` baru	Temukan dan tampilkan informasi tentang wilayah yang relatif padat di data Anda. Untuk informasi selengkapnya, lihat Contoh: Mendeteksi Hotspot di Aliran (Fungsi HOTSPOTS).	19 Maret 2018
Fungsi Lambda sebagai tujuan	Kirim hasil analisis ke fungsi Lambda sebagai tujuan. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda sebagai Output.	20 Desember 2017
Fungsi `RANDOM_CUT_FOREST_WITH_EXPLANATION` baru	Dapatkan penjelasan tentang bidang apa yang berkontribusi pada skor anomali dalam aliran data. Untuk informasi selengkapnya, lihat Contoh: Mendeteksi Anomali Data dan Mendapatkan Penjelasan (Fungsi RANDOM_CUT_FOREST_WITH_EXPLANATION).	2 November 2017

Perubahan	Deskripsi	Tanggal
Penemuan skema pada data statis	Jalankan penemuan skema pada data statis yang disimpan dalam bucket Amazon S3. Untuk informasi selengkapnya, lihat Menggunakan Fitur Penemuan Skema pada Data Statis.	6 Oktober 2017
Fitur prapemrosesan Lambda	Proses catatan terlebih dulu dalam aliran input dengan AWS Lambda sebelum analisis. Untuk informasi selengkapnya, lihat Memproses Data Terlebih Dulu Menggunakan Fungsi Lambda.	6 Oktober 2017
Menskalakan otomatis aplikasi	Secara otomatis tingkatkan throughput data aplikasi Anda dengan penskalaan otomatis. Untuk informasi selengkap nya, lihat Secara Otomatis Menskalakan Aplikasi untuk Meningkatkan Throughput.	13 September 2017
Beberapa aliran input dalam aplikasi	Tingkatkan throughput aplikasi dengan beberapa aliran dalam aplikasi. Untuk informasi selengkapnya, lihat Menyejajarkan Aliran Input untuk Peningkatan Throughpu t.	29 Juni 2017

Perubahan	Deskripsi	Tanggal
Panduan menggunakan AWS Management Console untuk Kinesis Data Analytics	Edit skema dan kode SQL yang disimpulkan menggunakan editor skema dan editor SQL di konsol Kinesis Data Analytics. Untuk informasi selengkapnya, lihat Langkah 4 (Opsional) Edit Skema dan Kode SQL Menggunakan Konsol.	7 April 2017
Rilis publik	Rilis publik Panduan Developer Amazon Kinesis Data Analytics.	11 Agustus, 2016
Rilis pratinjau	Rilis pratinjau Panduan Developer Amazon Kinesis Data Analytics.	29 Januari 2016

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS