SQL 개발자 안내서

Amazon Kinesis Data Analytics for SQL 애플리케이션 개발자 안내서

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Kinesis Data Analytics for SQL 애플리케이션 개발자 안내서: SQL 개발자 안내서

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

..... x
Amazon Kinesis Data Analytics for SQL 애플리케이션 단종 1
단종 계획 1
Amazon Managed Service for Apache Flink Studio로 마이그레이션 1
FAQ 2
Managed Service for Apache Flink로 마이그레이션 4
Amazon Kinesis Data Analytics for SQL 애플리케이션이란 무엇인가? 50
Amazon Kinesis Data Analytics를 언제 사용해야 하는가? 50
Amazon Kinesis Data Analytics를 처음 사용하십니까? 51
작동 방식 52
입력 55
스트리밍 소스 구성 56
참조 소스 구성 59
JSONPath로 작업하기 61
스트리밍 소스 요소를 SQL 입력 열에 매핑하기 67
스트리밍 데이터에 대해 스키마 검색 기능 사용 73
정적 데이터에 대해 스키마 검색 기능 사용 75
Lambda 함수를 사용하여 데이터 사전 처리 79
입력 스트림 병렬화를 통한 처리량 증대 89
애플리케이션 코드 94
출력 96
를 사용하여 출력 생성 AWS CLI 97
Lambda 함수를 출력으로 사용 98
애플리케이션 출력 전송 모델 107
오류 처리 107
애플리케이션 내 오류 스트림을 사용한 오류 보고 108
Auto Scaling 애플리케이션 109
태그 지정 109
애플리케이션이 생성될 때 태그 추가 110
기존 애플리케이션에 대한 태그 추가 또는 업데이트 110
애플리케이션에 대한 태그 나열 111
애플리케이션에서 태그 제거 111
시작하기 112
에 가입 AWS 계정 112

관리자 액세스 권한이 있는 사용자 생성 113
1단계: 계정 설정 114
에 가입 AWS 114
IAM 사용자 생성 115
다음 단계 115
에 가입 AWS 계정 112
관리자 액세스 권한이 있는 사용자 생성 113
2단계: 설정 AWS CLI 117
다음 단계 118
3단계: 스타터 Analytics 애플리케이션 생성 118
단계 3.1: 애플리케이션 만들기 121
3.2단계: 입력 구성 123
3.3단계: 실시간 분석 추가(애플리케이션 코드 추가) 126
3.4단계: 애플리케이션 코드 업데이트(선택 사항) 129
4단계 (선택 사항) 콘솔을 이용한 스키마와 SQL 코드 편집 131
스키마 편집기로 작업 132
SQL 편집기 작업 140
스트리밍 SQL 개념 144
애플리케이션 내 스트림과 펌프 144
타임스탬프와 ROWTIME 열 146
스트리밍 분석에서 사용되는 다양한 시간의 이해 146
연속 쿼리 149
윈도우 모드 쿼리 150
스태거 윈도우 151
텀블링 윈도우 155
슬라이딩 윈도우 157
스트림 조인 162
예 1: 주문 시점으로부터 1분 내에 거래가 이루어진 주문 보고 163
Kinesis Data Analytics for SQL 예 165
데이터 변환 165
Lambda를 통한 스트림 전처리 165
문자열 값 변환 166
DateTime 값 변환 186
복수의 데이터 유형 변환 190
윈도우 및 집계 199
스태거 윈도우 199

ROWTIME을 사용하는 텀블링 윈도우 203
이벤트 타임스탬프를 사용하는 텀블링 윈도우 207
가장 자주 발생하는 값(TOP_K_ITEMS_TUMBLING) 211
쿼리에서 부분적 결과 집계 214
조인 217
예: 참조 데이터 소스 추가 217
기계 학습 221
변칙 감지 222
예: 변칙 감지 및 설명 230
예: 핫스팟 감지 235
알림 및 오류 249
간단한 알림 249
조정된 알림 250
애플리케이션 내 오류 스트림 252
솔루션 액셀러레이터 254
AWS 계정 활동에 대한 실시간 인사이트 254
Kinesis Data Analytics를 사용한 실시간 AWS IoT 디바이스 모니터링 254
Kinesis Data Analytics를 사용한 실시간 웹 Analytics 254
Amazon Connected Vehicle 솔루션 254
보안 255
데이터 보호 256
데이터 암호화 256
ID 및 액세스 관리 257
신뢰 정책 257
권한 정책 258
교차 서비스 혼동된 대리자 방지 261
인증 및 액세스 통제 263
액세스 통제 263
ID를 통한 인증 264
액세스 관리 개요 267
정체 기반 정책(IAM 정책) 사용 272
API 권한 준거 279
모니터링 281
규정 준수 검증 281
복원성 281
재해 복구 282

인프라 보안 ... 282
보안 모범 사례 ... 282
IAM 역할을 사용하여 다른 Amazon 서비스에 액세스 ... 282
종속 리소스에서 서버 측 암호화 구현 ... 283
CloudTrail을 사용하여 API 직접 호출 모니터링 ... 283
모니터링 ... 284
모니터링 도구 ... 285
자동 도구 ... 285
수동 도구 ... 285
Amazon CloudWatch를 사용한 모니터링 ... 286
지표 및 차원 ... 286
지표 및 차원 보기 ... 289
경보 ... 289
로그 ... 291
사용 AWS CloudTrail ... 297
CloudTrail의 정보 ... 298
로그 파일 항목 이해 ... 299
한도 ... 301
단종 날짜 ... 301
Limits ... 301
모범 사례 ... 304
애플리케이션 관리 ... 304
Scaling 애플리케이션 ... 305
애플리케이션 모니터링 ... 306
입력 스키마 정의 ... 306
출력 연결 ... 308
애플리케이션 코드 작성 ... 308
애플리케이션 테스트 ... 309
테스트 애플리케이션 설정 ... 309
스키마 변경 사항 테스트 ... 309
코드 변경 사항 테스트 ... 310
문제 해결 ... 311
중지된 애플리케이션 ... 311
SQL 코드를 실행할 수 없음 ... 312
내 스키마를 탐지 또는 검색할 수 없음 ... 312
참조 데이터가 만료된 경우 ... 313

애플리케이션이 대상에 쓰지 않음 ... 313
모니터링해야 할 중요한 애플리케이션 상태 확인 파라미터 ... 314
애플리케이션 실행 시 잘못된 코드 오류 ... 314
애플리케이션이 오류 스트림에 오류라고 기록함 ... 314
불충분한 처리량 또는 높은 MillisBehindLatest ... 315
SQL 참조 ... 317
API 참조 ... 318
작업 ... 318
AddApplicationCloudWatchLoggingOption ... 320
AddApplicationInput ... 323
AddApplicationInputProcessingConfiguration ... 327
AddApplicationOutput ... 331
AddApplicationReferenceDataSource ... 335
CreateApplication ... 339
DeleteApplication ... 347
DeleteApplicationCloudWatchLoggingOption ... 350
DeleteApplicationInputProcessingConfiguration ... 353
DeleteApplicationOutput ... 356
DeleteApplicationReferenceDataSource ... 359
DescribeApplication ... 362
DiscoverInputSchema ... 367
ListApplications ... 372
ListTagsForResource ... 375
StartApplication ... 378
StopApplication ... 381
TagResource ... 383
UntagResource ... 386
UpdateApplication ... 389
데이터 타입 ... 394
ApplicationDetail ... 397
ApplicationSummary ... 401
ApplicationUpdate ... 403
CloudWatchLoggingOption ... 405
CloudWatchLoggingOptionDescription ... 406
CloudWatchLoggingOptionUpdate ... 408
CSVMappingParameters ... 410

DestinationSchema 411
Input 412
InputConfiguration 414
InputDescription 415
InputLambdaProcessor 418
InputLambdaProcessorDescription 420
InputLambdaProcessorUpdate 421
InputParallelism 423
InputParallelismUpdate 424
InputProcessingConfiguration 425
InputProcessingConfigurationDescription 426
InputProcessingConfigurationUpdate 427
InputSchemaUpdate 428
InputStartingPositionConfiguration 430
InputUpdate 431
JSONMappingParameters 433
KinesisFirehoseInput 434
KinesisFirehoseInputDescription 435
KinesisFirehoseInputUpdate 436
KinesisFirehoseOutput 437
KinesisFirehoseOutputDescription 438
KinesisFirehoseOutputUpdate 439
KinesisStreamsInput 440
KinesisStreamsInputDescription 441
KinesisStreamsInputUpdate 442
KinesisStreamsOutput 443
KinesisStreamsOutputDescription 444
KinesisStreamsOutputUpdate 445
LambdaOutput 446
LambdaOutputDescription 448
LambdaOutputUpdate 449
MappingParameters 451
Output 452
OutputDescription 454
OutputUpdate 456
RecordColumn 458

RecordFormat 460
ReferenceDataSource 461
ReferenceDataSourceDescription 463
ReferenceDataSourceUpdate 465
S3Configuration 467
S3ReferenceDataSource 469
S3ReferenceDataSourceDescription 471
S3ReferenceDataSourceUpdate 473
SourceSchema 475
Tag 477
설명서 이력 478
AWS 용어집 482

신중한 고려 끝에 두 단계로 Amazon Kinesis Data Analytics for SQL 애플리케이션을 단종하기로 결정했습니다.

1. 2025년 10월 15일부터 새 Kinesis Data Analytics for SQL 애플리케이션을 생성할 수 없습니다.

2. 2026년 1월 27일부터 애플리케이션이 삭제됩니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 시작하거나 작동할 수 없게 됩니다. 그 시점부터 Amazon Kinesis Data Analytics for SQL에 대한 지원을 더 이상 이용할 수 없습니다. 자세한 내용은 Amazon Kinesis Data Analytics for SQL 애플리케이션 단종 단원을 참조하십시오.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.

Amazon Kinesis Data Analytics for SQL 애플리케이션 단종

단종 계획

신중한 고려 끝에 Amazon Kinesis Data Analytics for SQL 애플리케이션을 단종하기로 결정했습니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 계획하고 마이그레이션하는 데 도움이 되도록 15개월에 걸쳐 오퍼링을 점진적으로 단종할 예정입니다. 알아두어야 할 중요한 두 날짜는 2025년 10월 15일과 2026년 1월 27일입니다.

1. 2025년 10월 15일에 고객의 애플리케이션이 중지되고 `READY` 상태로 전환됩니다. 이때 애플리케이션을 다시 시작하여 애플리케이션을 정상적으로 계속 사용할 수 있지만 서비스 제한이 적용됩니다.
2. 2025년 10월 15일부터는 새 Amazon Kinesis Data Analytics for SQL 애플리케이션을 생성할 수 없습니다. 기존 애플리케이션은 정상적으로 실행할 수 있지만 서비스 제한이 적용됩니다.
3. 2026년 1월 27일부터 애플리케이션이 삭제됩니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 시작하거나 작동할 수 없게 됩니다. 그 시점부터 Amazon Kinesis Data Analytics for SQL 애플리케이션에 대한 지원을 더 이상 이용할 수 없습니다.

2025년 10월 15일 이전에 애플리케이션을 Amazon Managed Service for Apache Flink 또는 Amazon Managed Service for Apache Flink Studio로 마이그레이션하는 것이 좋습니다. 마이그레이션에 도움이 되는 리소스는 Managed Service for Apache Flink Studio로 마이그레이션하는 예 섹션을 참조하세요. Amazon Managed Service for Apache Flink 또는 Amazon Managed Service for Apache Flink Studio에 대한 자세한 내용은 Amazon Managed Service for Apache Flink 개발자 안내서를 참조하세요.

> **Note**
>
> Amazon Kinesis Data Analytics for SQL 애플리케이션의 전체 목록을 보려는 경우 `ListApplications` API를 직접적으로 호출할 수 있습니다. 자세한 내용은 ListApplications를 참조하세요.

Amazon Managed Service for Apache Flink Studio로 마이그레이션

애플리케이션 마이그레이션에 대해 자세히 알아보고 코드 및 아키텍처 예를 보려면 Managed Service for Apache Flink Studio로 마이그레이션하는 예 섹션을 참조하세요.

FAQ

Amazon Kinesis Data Analytics for SQL 애플리케이션을 더 이상 제공하지 않게 되는 이유는 무엇인가요?

고객은 실시간 데이터 스트림 처리 워크로드에 Amazon Managed Service for Apache Flink 오퍼링을 선호하는 것으로 나타났습니다. Amazon Managed Service for Apache Flink는 데이터 스트림을 처리하는 오픈 소스 엔진인 Apache Flink를 사용하며 지연 시간이 짧고 확장성 및 가용성이 뛰어난 서버리스 실시간 스트림 처리 서비스입니다. Amazon Managed Service for Apache Flink는 기본 규모 조정, 정확히 한 번 처리하는 의미 체계, 다국어 지원(SQL 포함), 40여 개 소스 및 대상 커넥터, 안정적인 애플리케이션 상태 등과 같은 기능을 제공합니다. 이러한 기능을 통해 고객은 엔드 투 엔드 스트리밍 파이프라인을 구축하고 데이터의 정확성과 적시성을 보장할 수 있습니다.

현재 고객이 선택할 수 있는 방안은 무엇인가요?

고객은 기존 Amazon Kinesis Data Analytics for SQL 애플리케이션을 Amazon Managed Service for Apache Flink Studio 또는 Amazon Managed Service for Apache Flink로 업그레이드하는 것이 좋습니다. Amazon Managed Service for Apache Flink Studio에서는 대화형 노트북을 통해 SQL, Python 또는 Scala를 사용하여 쿼리를 생성할 수 있습니다. Amazon Kinesis Data Analytics for SQL에서 장기간 실행 중인 애플리케이션의 경우 Amazon Managed Service for Apache Flink를 사용하는 것이 좋습니다. 여기서는 Apache Flink의 모든 API, 커넥터 등을 통해 Java, Python, Scala, 임베디드 SQL을 사용하여 애플리케이션을 생성할 수 있습니다.

Amazon Kinesis Data Analytics for SQL 애플리케이션을 사용 중인 고객은 얼마나 되나요?

고객 세부 정보는 공개할 수 없습니다.

고객은 Amazon Kinesis Data Analytics for SQL 애플리케이션에서 Amazon Managed Service for Apache Flink 오퍼링으로 어떻게 업그레이드하나요?

Amazon Managed Service for Apache Flink 또는 Amazon Managed Service for Apache Flink Studio로 업그레이드하려면 고객이 애플리케이션을 다시 생성해야 합니다. 이를 돕기 위해 일반적인 SQL 쿼리 라이브러리와 Amazon Managed Service for Apache Flink Studio에서 이러한 쿼리를 다시 작성하는 방법을 제공합니다. Managed Service for Apache Flink Studio에서 Kinesis Data Analytics for SQL 쿼리를 복제하는 방법을 참조하세요. 또한 Amazon Managed Service for Apache Flink에서 장기 실행 애플리케이션을 빌드하거나 기계 학습을 사용하는 경우 고객이 따를 수 있는 일반적인 패턴 아키텍처를 제공합니다. 소스인 Kinesis Data Firehose를 Kinesis Data Streams로 대체 부분 참조

Amazon Managed Service for Apache Flink에 대한 자세한 내용은 Amazon Managed Service for Apache Flink를 참조하세요.

마이그레이션 가이드는 Managed Service for Apache Flink Studio로 마이그레이션하는 예 섹션을 참조하세요.

Amazon Managed Service for Apache Flink는 기존 Amazon Kinesis Data Analytics for SQL 애플리케이션 기능을 지원하나요?

Amazon Managed Service for Apache Flink는 커넥터 및 윈도우와 같은 Amazon Kinesis Data Analytics for SQL 애플리케이션에 있는 많은 개념은 물론이고, 기본 규모 조정, 정확히 한 번만 처리하는 의미 체계, 다국어 지원(SQL 포함), 40여 개의 소스 및 대상 커넥터, 안정적인 애플리케이션 상태 등과 같은 Amazon Kinesis Data Analytics for SQL 애플리케이션에 없는 여러 가지 기능을 지원합니다.

Managed Service for Apache Flink Studio로 마이그레이션하는 예

신중한 고려 끝에 Amazon Kinesis Data Analytics for SQL 애플리케이션을 단종하기로 결정했습니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 계획하고 마이그레이션하는 데 도움이 되도록 15개월에 걸쳐 오퍼링을 점진적으로 단종할 예정입니다. 알아두어야 할 중요한 두 날짜는 2025년 10월 15일과 2026년 1월 27일입니다.

1. 2025년 10월 15일부터는 새 Amazon Kinesis Data Analytics for SQL 애플리케이션을 생성할 수 없습니다.
2. 2026년 1월 27일부터 애플리케이션이 삭제됩니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 시작하거나 작동할 수 없게 됩니다. 그 시점부터 Amazon Kinesis Data Analytics for SQL 애플리케이션에 대한 지원을 더 이상 이용할 수 없습니다. 자세한 내용은 Amazon Kinesis Data Analytics for SQL 애플리케이션 단종을 참조하십시오.

Amazon Managed Service for Apache Flink를 사용하는 것이 좋습니다. 사용 편의성과 고급 분석 기능을 결합하여 몇 분 만에 스트림 처리 애플리케이션을 빌드할 수 있습니다.

이 섹션에서는 Amazon Kinesis Data Analytics for SQL 애플리케이션 워크로드를 Managed Service for Apache Flink로 이동하는 데 도움이 되는 코드 및 아키텍처 예를 제공합니다.

자세한 내용은 AWS 블로그 게시물: Migrate from Amazon Kinesis Data Analytics for SQL Applications to Managed Service for Apache Flink Studio를 참조하세요.

Managed Service for Apache Flink Studio에서 Kinesis Data Analytics for SQL 쿼리를 복제하는 방법

워크로드를 Managed Service for Apache Flink Studio 또는 Managed Service for Apache Flink로 마이그레이션하기 위해 이 섹션에서는 일반적인 사용 사례에 사용할 수 있는 쿼리 번역을 제공합니다.

이러한 예를 살펴보기 전에 먼저 Managed Service for Apache Flink가 포함된 Studio 노트북 사용을 검토하는 것이 좋습니다.

Managed Service for Apache Flink Studio에서 Kinesis Data Analytics for SQL 쿼리를 다시 만들기

아래 옵션에서는 일반적인 SQL 기반 Kinesis Data Analytics 애플리케이션 쿼리를 Managed Service for Apache Flink Studio로 변환하는 것을 보여줍니다.

다단계 애플리케이션

SQL-based Kinesis Data Analytics application

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

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ingest_time TIMESTAMP,
   ticker_symbol VARCHAR(4),
   sector VARCHAR(16),
   price REAL,
   change REAL);
CREATE
OR REPLACE PUMP "STREAM_PUMP_03" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM ingest_time,
      ticker_symbol,
      sector,
      price,
      change FROM "IN_APP_STREAM_02";
```

Managed Service for Apache Flink Studio

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

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

FROM
   'timestamp' VIRTUAL,
   WATERMARK FOR APPROXIMATE_ARRIVAL_TIME AS APPROXIMATE_ARRIVAL_TIME - INTERVAL '1'
 SECOND )
   PARTITIONED BY (TICKER_SYMBOL) WITH (
      'connector' = 'kinesis',
      'stream' = 'kinesis-analytics-demo-stream',
      'aws.region' = 'us-east-1',
      'scan.stream.initpos' = 'LATEST',
      'format' = 'json',
      'json.timestamp-format.standard' = 'ISO-8601');
DROP TABLE IF EXISTS IN_APP_STREAM_001;

CREATE TABLE IN_APP_STREAM_001 (
   INGEST_TIME TIMESTAMP,
   TICKER_SYMBOL VARCHAR(4),
```

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

DROP TABLE IF EXISTS IN_APP_STREAM_02;

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

DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;

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


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

```
   INSERT INTO
      IN_APP_STREAM_001
      SELECT
         APPROXIMATE_ARRIVAL_TIME AS INGEST_TIME,
         TICKER_SYMBOL,
         SECTOR,
         PRICE,
         CHANGE
      FROM
         SOURCE_SQL_STREAM_001;


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


Query 4 - % flink.ssql(type =
update
)
   INSERT INTO
      DESTINATION_SQL_STREAM
      SELECT
         INGEST_TIME,
         TICKER_SYMBOL,
         SECTOR,
         PRICE,
         CHANGE
      FROM
         IN_APP_STREAM_02;
```

DateTime 값 변환

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;
```

간단한 알림

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;
```

조정된 알림

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;
```

쿼리에서 부분적 결과 집계

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;
```

문자열 값 변환

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;
```

Regex를 사용하여 하위 문자열 대체

SQL-based Kinesis Data Analytics application

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

Managed Service for Apache Flink Studio

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

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

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

Regex 로그 파싱

SQL-based Kinesis Data Analytics application

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

Managed Service for Apache Flink Studio

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

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

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

DateTime 값 변환

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;
```

창 및 집계

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;
```

ROWTIME을 사용하는 텀블링 창

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;
```

가장 자주 발생하는 값 검색(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;
```

대략적인 Top-K 항목

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
```

웹 로그 구문 파싱(W3C_LOG_PARSE 함수)

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;
```

문자열을 복수의 필드로 분할(VARIABLE_COLUMN_LOG_PARSE 함수)

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;
```

조인

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;
```

오류

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;
```

Random Cut Forest 워크로드 마이그레이션

Random Cut Forest를 사용하는 워크로드를 Kinesis Analytics for SQL에서 Managed Service for Apache Flink로 이동하려는 경우, 이 AWS 블로그 게시물에서는 Managed Service for Apache Flink를 사용하여 이상 항목 탐지를 위한 온라인 RCF 알고리즘을 실행하는 방법을 보여줍니다.

소스인 Kinesis Data Firehose를 Kinesis Data Streams로 대체

전체 튜토리얼은 Converting-KDASQL-KDAStudio/를 참조하십시오.

다음 연습에서는 Amazon Managed Service for Apache Flink Studio를 사용하도록 데이터 흐름을 변경해 보겠습니다. Amazon Kinesis Data Firehose에서 Amazon Kinesis Data Streams로 전환하는 것도 의미합니다.

먼저 일반적인 KDA-SQL 아키텍처를 공유하고, Amazon Managed Service for Apache Flink Studio 및 Amazon Kinesis Data Streams를 사용하여 이를 대체할 수 있는 방법을 보여줍니다. 또는 여기에서 AWS CloudFormation 템플릿을 시작할 수 있습니다.

Amazon Kinesis Data Analytics-SQL 및 Amazon Kinesis Data Firehose

Amazon Kinesis Data Analytics SQL 아키텍처 흐름은 다음과 같습니다:

먼저 기존 Amazon Kinesis Data Analytics-SQL과 Amazon Kinesis Data Firehose의 설정을 살펴봅니다. 사용 사례는 주식 시세 및 가격을 포함한 거래 데이터가 외부 소스에서 Amazon Kinesis 시스템으로 스트리밍되는 거래 시장입니다. Amazon Kinesis Data Analytics for SQL은 입력 스트림을 사용하여 텀블링 윈도우와 같은 윈도우 쿼리를 실행하여 각 주식 시세 표시기에 대해 1분 기간의 거래량 및 `min`, `max`, `average` 거래 가격을 결정합니다.

Amazon Kinesis Data Analytics-SQL은 Amazon Kinesis Data Firehose API에서 데이터를 수집하도록 설정되어 있습니다. 처리가 끝나면 Amazon Kinesis Data Analytics-SQL은 처리된 데이터를 다른 Amazon Kinesis Data Firehose로 보내고, 이 서비스는 출력을 Amazon S3 버킷에 저장합니다.

이 경우에는 Amazon Kinesis 데이터 생성기를 사용합니다. Amazon Kinesis 데이터 생성기를 사용하면 Amazon Kinesis Data Streams 또는 Amazon Kinesis Data Firehose 전송 스트림으로 테스트 데이

터를 전송할 수 있습니다. 시작하려면 여기의 지침을 따르세요. 지침에 제공된 템플릿 대신 여기에 있는 AWS CloudFormation 템플릿을 사용합니다.

AWS CloudFormation 템플릿을 실행하면 출력 섹션에 Amazon Kinesis Data Generator URL이 제공됩니다. 여기에서 설정한 Cognito 사용자 ID와 비밀번호를 사용하여 포털에 로그인합니다. 리전 및 대상 스트림 명칭을 선택합니다. 현재 상태에서는 Amazon Kinesis Data Firehose 전송 스트림을 선택합니다. 새 상태에서는 Amazon Kinesis Data Firehose 스트림 명칭을 선택합니다. 요건에 따라 여러 템플릿을 생성하고 대상 스트림으로 보내기 전에 템플릿 테스트 버튼을 사용하여 템플릿을 테스트할 수 있습니다.

다음은 Amazon Kinesis 데이터 생성기를 사용하는 샘플 페이로드입니다. 데이터 생성기는 입력 Amazon Kinesis Firehose 스트림을 대상으로 하여 데이터를 지속적으로 스트리밍합니다. Amazon Kinesis SDK 클라이언트는 다른 생산자로부터도 데이터를 전송할 수 있습니다.

```
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
```

다음 JSON은 일련의 거래 시간 및 날짜, 주식 시세 표시기, 주가를 무작위로 생성하는 데 사용됩니다.

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

데이터 보내기를 선택하면 생성기가 모의 데이터 전송을 시작합니다.

외부 시스템은 Amazon Kinesis Data Firehose로 데이터를 스트리밍합니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 사용하면 표준 SQL을 사용하여 스트리밍 데이터를 분석할 수 있습니다. 이 서비스를 사용하면 스트리밍 소스에 대해 SQL 코드를 작성하고 실행하여 시계열 분석을 수행하고, 실시간 대시보드를 공급하고, 실시간 지표를 생성할 수 있습니다. Amazon Kinesis Data Analytics for SQL 애플리케이션은 입력 스트림의 SQL 쿼리에서 대상 스트림을 생성하고 대상 스트림을 다른 Amazon Kinesis Data Firehose로 보낼 수 있습니다. 대상 Amazon Kinesis Data Firehose는 분석 데이터를 Amazon S3에 최종 상태로 보낼 수 있습니다.

Amazon Kinesis Data Analytics-SQL 레거시 코드는 SQL 표준의 확장을 기반으로 합니다.

Amazon Kinesis Data Analytics-SQL에서 다음 쿼리를 사용합니다. 먼저 쿼리 출력을 위한 대상 스트림을 생성합니다. 그런 다음 연속적 `INSERT INTO stream SELECT ... FROM` 쿼리 기능 실행을 제

공하는 Amazon Kinesis Data Analytics 리포지토리 객체(SQL 표준의 확장)인 PUMP를 사용하여 쿼리 결과를 명명된 스트림에 지속적으로 입력할 수 있습니다.

```
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);
```

앞의 SQL은 두 개의 시간 윈도우를 사용합니다. 하나는 수신 스트림 페이로드에서 오는 `tradeTimestamp`이고, 다른 하나는 `Event Time` 또는 `client-side time`이라고도 하는 `ROWTIME.tradeTimestamp`입니다. 이벤트가 발생한 시간이기 때문에 분석에서 이 시간을 사용하는 것이 바람직한 경우가 많습니다. 그러나 스마트폰 및 웹 클라이언트와 같은 많은 이벤트 소스가 신뢰할 만한 시계를 갖추고 있지 않아 부정확한 시간을 야기할 수 있습니다. 또한 연결 문제로 인해 레코드가 이벤트가 발생한 순서대로 스트림에 나타나지 않을 수 있습니다.

애플리케이션 내 스트림에는 `ROWTIME`이라고 하는 특수 열이 포함됩니다. Amazon Kinesis Data Analytics가 첫 번째 애플리케이션 내 스트림에 행을 삽입하면 타임스탬프를 저장합니다. `ROWTIME`은 Amazon Kinesis Data Analytics가 스트리밍 소스에서 읽은 후 첫 번째 애플리케이션 내 스트림을 레코

드에 삽입한 타임스탬프를 나타냅니다. 그런 다음 이 ROWTIME 값은 애플리케이션 전체에 걸쳐 유지됩니다.

SQL은 티커 수를 60초 간격 내의 volume, min, max, average 가격으로 결정합니다.

시간 기반 창 모드 쿼리에서 이들 시간 각각을 사용하는 것은 장점과 단점이 있습니다. 이들 시간 중 하나 이상을 선택하고 사용 시나리오를 바탕으로 관련 단점을 처리할 전략을 선택하십시오.

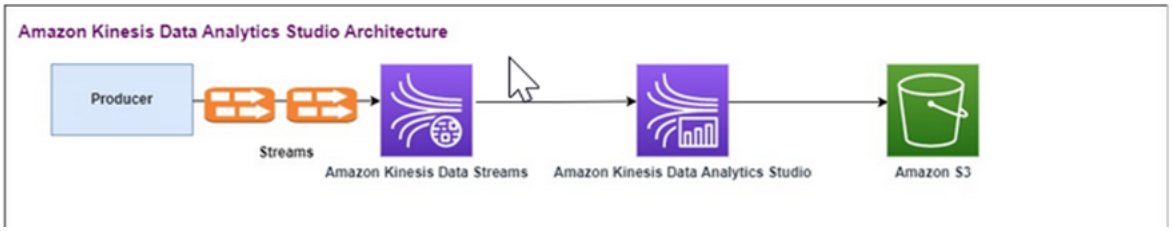
두 개의 시간 기반 창, 즉 ROWTIME과 이벤트 시간 같은 다른 시간인 두 가지의 시간 기반을 사용하는 전략

- 다음 예에서와 같이 쿼리가 결과를 방출하는 빈도를 제어하는 첫 번째 창으로 ROWTIME을 사용합니다. 논리 시간으로는 사용되지 않습니다.
- 분석과 연관시키고자 하는 논리 시간인 다른 시간 중 하나를 사용하십시오. 이 시간은 이벤트가 발생한 시간을 나타냅니다. 다음 예에서 분석 목적은 레코드를 그룹화하고 티커별 카운트를 반환하는 것입니다.

Amazon Managed Service for Apache Flink Studio

업데이트된 아키텍처에서는 Amazon Kinesis Data Firehose를 Amazon Kinesis Data Streams로 교체합니다. Amazon Kinesis Data Analytics for SQL 애플리케이션은 Amazon Managed Service for Apache Flink Studio로 대체되었습니다. Apache Flink 코드는 Apache Zeppelin Notebook 내에서 대화형 방식으로 실행됩니다. Amazon Managed Service for Apache Flink Studio에서 집계된 거래 데이터를 Amazon S3 버킷으로 전송하여 저장합니다. 단계는 다음과 같습니다:

Amazon Managed Service for Apache Flink Studio 아키텍처 흐름은 다음과 같습니다:



Kinesis Data Stream 생성

콘솔을 사용하여 데이터 스트림을 만들려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.

2. 탐색 모음에서 리전 선택기를 확장하고 리전을 선택합니다.
3. 데이터 스트림 생성을 선택합니다.
4. Kinesis 스트림 생성 페이지에서 데이터 스트림의 명칭을 입력하고 기본 온디맨드 용량 모드를 수락합니다.

   온디맨드 모드를 사용하면 Kinesis 스트림 생성을 선택하여 데이터 스트림을 생성할 수 있습니다.

   스트림이 생성 중인 동안 Kinesis streams(Kinesis 스트림) 페이지에서 스트림의 Status(상태)는 Creating(생성 중)입니다. 스트림을 사용할 준비가 되면 Status(상태)가 Active(활성)로 변경됩니다.
5. 스트림 명칭을 선택합니다. 스트림 세부 정보 페이지에 모니터링 정보와 함께 스트림 구성 요약이 표시됩니다.
6. Amazon Kinesis 데이터 생성기에서 스트림/전송 스트림을 새 Amazon Kinesis Data Streams 으로 변경합니다: TRADE_SOURCE_STREAM.

   JSON과 페이로드는 Amazon Kinesis Data Analytics-SQL에서 사용한 것과 동일합니다. Amazon Kinesis 데이터 생성기를 사용하여 몇 가지 샘플 거래 페이로드 데이터를 생성하고 TRADE_SOURCE_STREAM 데이터 스트림을 이 연습의 대상으로 삼으십시오.

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

7. 에서 Managed Service for Apache Flink로 AWS Management Console 이동한 다음 애플리케이션 생성을 선택합니다.
8. 왼쪽 탐색 창에서 Studio 노트북을 선택한 다음 Studio 노트북 생성을 선택합니다.
9. 스튜디오 노트북의 명칭을 입력합니다.
10. AWS Glue 데이터베이스에서 소스 및 대상의 메타데이터를 정의할 기존 AWS Glue 데이터베이스를 입력합니다. AWS Glue 데이터베이스가 없는 경우 생성을 선택하고 다음을 수행합니다.

    a. AWS Glue 콘솔의 왼쪽 메뉴에 있는 데이터 카탈로그에서 데이터베이스를 선택합니다.

    b. 데이터베이스 생성을 선택합니다.

    c. 데이터베이스 생성 페이지에서 데이터베이스의 명칭을 입력합니다. 위치 - 옵션 섹션에서 Browse Amazon S3 검색을 선택하고 Amazon S3 버킷을 선택합니다. Amazon S3 버킷이 아직 설정되지 않은 경우 이 단계를 건너뛰고 나중에 다시 돌아올 수 있습니다.

    d. (선택 사항). 데이터베이스에 대한 설명을 입력합니다.

   e. 데이터베이스 생성를 선택합니다.
11. 노트북 생성을 선택합니다.
12. 노트북이 생성되면 실행을 선택합니다.
13. 노트북이 성공적으로 시작되면 Apache Zeppelin에서 열기를 선택하여 Zeppelin 노트북을 실행합니다.
14. Zeppelin 노트북 페이지에서 새 노트 만들기를 선택하고 명칭을 MarketDataFeed로 지정합니다.

Flink SQL 코드에 대한 설명은 다음과 같지만, 먼저 Zeppelin 노트북 화면의 모양은 다음과 같습니다. 노트북 내의 각 창은 별도의 코드 블록이며 한 번에 하나씩 실행할 수 있습니다.

Amazon Managed Service for Apache Flink Studio 코드

Amazon Managed Service for Apache Flink Studio는 Zeppelin 노트북을 사용하여 코드를 실행합니다. 이 예에서는 Apache Flink 1.13에 근거한 ssql 코드에 매핑했습니다. Zeppelin 노트북의 코드는 한 번에 한 블록씩 아래에 나와 있습니다.

Zeppelin 노트북에서 코드를 실행하기 전에 Flink 구성 명령을 실행해야 합니다. 코드(ssql, Python 또는 Scala)를 실행한 후 구성 설정을 변경해야 하는 경우 노트북을 중지하고 다시 시작해야 합니다. 이 예에서는 체크포인트를 설정해야 합니다. Amazon S3의 파일로 데이터를 스트리밍하려면 체크포인트가 요구됩니다. 이렇게 하면 Amazon S3로 스트리밍되는 데이터를 파일로 플러싱할 수 있습니다. 다음 명령문은 간격을 5,000밀리초로 설정합니다.

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

`%flink.conf`은 이 블록이 구성 명령문임을 나타냅니다. 체크포인트를 포함한 Flink 구성에 대한 자세한 내용은 Apache Flink 체크포인트를 참조하세요.

소스 Amazon Kinesis Data Streams의 입력 테이블은 다음 Flink ssql 코드를 사용하여 생성됩니다. 참고로 `TRADE_TIME` 필드에는 데이터 생성기가 생성한 날짜/시간이 저장됩니다.

```
%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');
```

다음 명령문을 사용하여 입력 스트림을 볼 수 있습니다:

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

select * from TRADE_SOURCE_STREAM;
```

Amazon S3로 집계 데이터를 전송하기 전에 Amazon Managed Service for Apache Flink Studio에서 텀블링 창 선택 쿼리를 사용하여 직접 데이터를 볼 수 있습니다. 이렇게 하면 거래 데이터가 1분 단위로 집계됩니다. 참고로 %flink.ssql 문에는 (유형=업데이트) 라는 대상이 있어야 합니다:

```
%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;
```

그런 다음 Amazon S3에서 대상 주소에 대한 표를 생성할 수 있습니다. 워터마크를 사용해야 합니다. 워터마크는 지연된 이벤트가 더 이상 발생하지 않을 것으로 확신하는 시점을 나타내는 진행률 지표입니다. 워터마크를 사용하는 이유는 늦게 도착하는 사람들을 고려하기 위함입니다. 간격 '5' Second을 통해 거래가 5초 늦게 Amazon Kinesis Data Stream에 입력될 수 있으며, 기간 내에 타임스탬프가 있는 거래는 여전히 포함됩니다. 자세한 내용은 워터마크 생성을 참조하세요.

```
%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');
```

이 명령문은 데이터를 TRADE_DESTINATION_S3에 삽입합니다. TUMPLE_ROWTIME는 텀블링 창의 포함 상한선의 타임스탬프입니다.

```
%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;
```

명령문을 10~20분 동안 실행하여 Amazon S3에 일부 데이터를 누적하십시오. 그런 다음 명령문을 중단하십시오.

그러면 Amazon S3의 파일이 닫히고 해당 파일을 볼 수 있게 됩니다.

내용은 다음과 같습니다:

part-2d9eca09-1d57-450f-b583-11b33b089518-0-3

```
"2023-03-01 17:23:59.999",AMZN,16,1014.75,224.0,1855.0
"2023-03-01 17:23:59.999",AAPL,20,935.45,123.0,1972.0
"2023-03-01 17:23:59.999",MSFT,20,847.55,15.0,1963.0
"2023-03-01 17:23:59.999",META,23,968.5217391304348,125.0,1995.0
"2023-03-01 17:23:59.999",GOOGL,21,949.0,43.0,1996.0
"2023-03-01 17:26:59.999",GOOGL,19,957.578947368421,180.0,1968.0
"2023-03-01 17:26:59.999",AAPL,25,969.72,24.0,1939.0
"2023-03-01 17:26:59.999",AMZN,24,1021.875,101.0,1995.0
"2023-03-01 17:26:59.999",META,14,715.1428571428571,16.0,1504.0
"2023-03-01 17:26:59.999",MSFT,18,1050.4444444444443,123.0,1977.0
"2023-03-01 17:30:59.999",AAPL,20,860.3,94.0,1776.0
"2023-03-01 17:30:59.999",GOOGL,19,1078.4736842105262,9.0,1732.0
"2023-03-01 17:30:59.999",MSFT,20,1180.35,181.0,1981.0
"2023-03-01 17:30:59.999",AMZN,20,974.3,50.0,1791.0
"2023-03-01 17:30:59.999",META,21,902.2857142857143,6.0,1896.0
"2023-03-01 17:33:59.999",AAPL,21,1029.8095238095239,121.0,1657.0
"2023-03-01 17:33:59.999",GOOGL,21,1008.6190476190476,62.0,1979.0
"2023-03-01 17:33:59.999",META,21,1119.142857142857,126.0,1916.0
"2023-03-01 17:33:59.999",AMZN,27,1073.2592592592594,179.0,1849.0
"2023-03-01 17:33:59.999",MSFT,10,1353.4,584.0,1996.0
"2023-03-01 17:36:59.999",GOOGL,18,1262.5,15.0,1997.0
"2023-03-01 17:36:59.999",MSFT,19,780.8947368421053,30.0,1558.0
"2023-03-01 17:36:59.999",AAPL,15,1263.1333333333334,185.0,1928.0
"2023-03-01 17:36:59.999",AMZN,20,882.85,153.0,1764.0
"2023-03-01 17:36:59.999",META,28,858.5,14.0,1917.0
"2023-03-01 17:40:59.999",AMZN,22,976.9545454545455,98.0,1878.0
"2023-03-01 17:40:59.999",GOOGL,25,830.32,39.0,1991.0
"2023-03-01 17:40:59.999",MSFT,18,1325.0,28.0,1966.0
"2023-03-01 17:40:59.999",META,19,855.578947368421,96.0,1725.0
"2023-03-01 17:40:59.999",AAPL,16,1134.25,1.0,1939.0
```

AWS CloudFormation 템플릿을 사용하여 인프라를 생성할 수 있습니다.

AWS CloudFormation 는 AWS 계정에 다음 리소스를 생성합니다.

- Amazon Kinesis Data Streams
- Amazon Managed Service for Apache Flink Studio
- AWS Glue 데이터베이스
- Amazon S3 버킷
- Amazon Managed Service for Apache Flink Studio에서 적절한 리소스에 액세스하기 위한 IAM 역할 및 정책

노트북을 가져오고에서 생성한 새 Amazon S3 버킷으로 Amazon S3 버킷 이름을 변경합니다 AWS CloudFormation.

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

DROP TABLE IF EXISTS  TRADE_DESTINATION_S3;
CREATE TABLE TRADE_DESTINATION_S3 (
TRADE_WINDOW_START TIMESTAMP(3),
WATERMARK FOR TRADE_WINDOW_START as TRADE_WINDOW_START - INTERVAL '5' SECOND,
TICKER STRING,
VOLUME BIGINT,
AVG_PRICE DOUBLE,
MIN_PRICE DOUBLE,
MAX_PRICE DOUBLE)
WITH ('connector' = 'filesystem','path' = 's3://kda-studio-test-stack-markettradinganalyticsc[redacted]/','format' = 'csv');
```

자세한 내용을 참조하십시오.

다음은 Managed Service for Apache Flink Studio를 사용하는 방법에 대해 자세히 알아볼 수 있는 몇 가지 추가 리소스입니다.

- Managed Service for Apache Flink Studio Notebooks 개발자 가이드
- 아파치 플링크 1.13 설명서
- Managed Service for Apache Flink Studio 워크샵
- Apache Flink 창
- Amazon Kinesis Data Analytics 개발자 가이드 - Kinesis Data Analytics 스트림에서 S3 버킷으로 작성

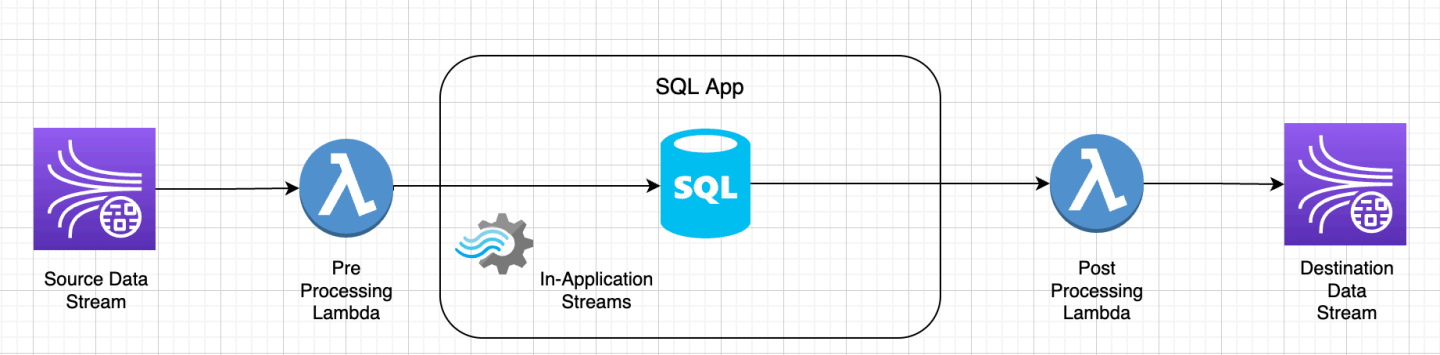
사용자 정의 함수 (UDF) 활용

이 패턴의 목적은 Kinesis Data Analytics-Studio Zeppelin 노트북에서 UDF를 활용하여 Kinesis 스트림의 데이터를 처리하는 방법을 시연하는 것입니다. Managed Service for Apache Flink Studio는 Apache Flink를 사용하여 정확히 한 번 처리되는 의미 체계를 비롯한 고급 분석 기능, 이벤트 시간 윈도우, 사용자 정의 함수 및 고객 통합을 사용한 확장성, 명령형 언어 지원, 안정적인 애플리케이션 상태, 수평적 스

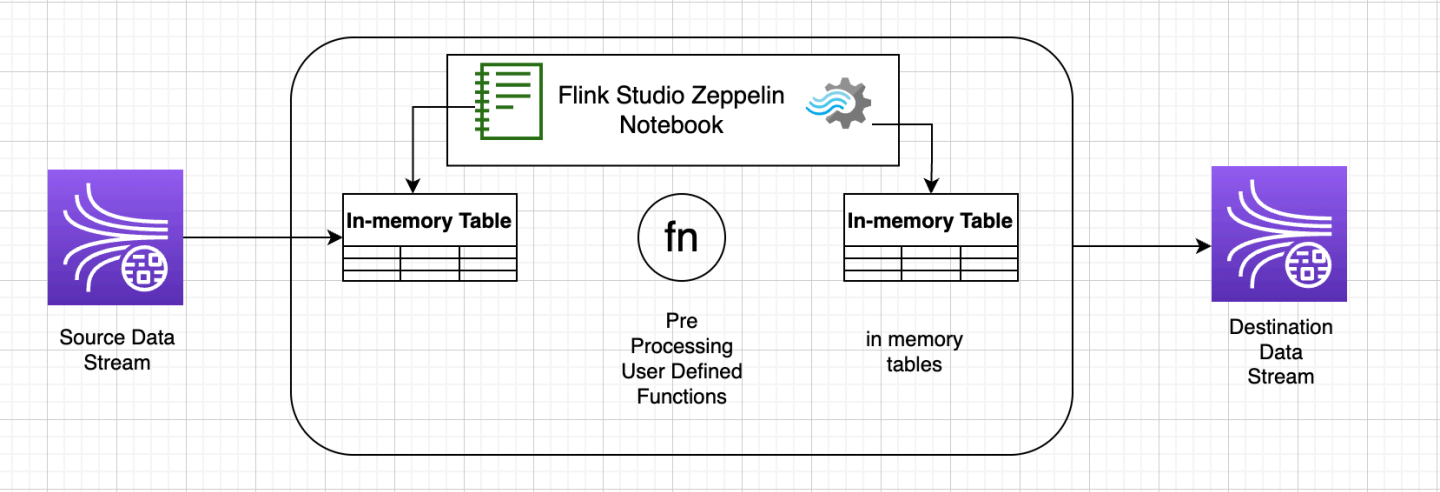
케일링, 다중 데이터 소스 지원, 확장 가능한 통합 기능 등을 제공합니다. 이는 데이터 스트림 처리의 정확성, 완전성, 일관성 및 안정성을 보장하는 데 중요하며 Amazon Kinesis Data Analytics for SQL에서는 사용할 수 없습니다.

이 샘플 애플리케이션에서는 KDA-Studio Zeppelin 노트북의 UDF를 활용하여 Kinesis 스트림의 데이터를 처리하는 방법을 보여줍니다. Kinesis Data Analytics용 Studio 노트북을 사용하면 실시간으로 대화형 방식으로 데이터 스트림을 쿼리하고 표준 SQL, Python 및 Scala를 사용하여 스트림 처리 애플리케이션을 쉽게 구축하고 실행할 수 있습니다. 에서 몇 번의 클릭만으로 서버리스 노트북을 시작하여 데이터 스트림을 쿼리하고 몇 초 만에 결과를 얻을 AWS Management Console수 있습니다. 자세한 설명은 Kinesis Data Analytics for Apache Flink와 함께 Studio 노트북 사용하기를 참조하십시오.

KDA-SQL 애플리케이션에서 데이터의 사전/사후 처리에 사용되는 Lambda 함수:



KDA-Studio Zeppelin 노트북을 사용하여 데이터를 사전/사후 처리하기 위한 사용자 정의 함수



사용자 정의 함수(UDF)

일반적인 비즈니스 로직을 연산자로 재사용하려면 사용자 정의 함수를 참조하여 데이터 스트림을 변환하는 것이 유용할 수 있습니다. 이 작업은 Managed Service for Apache Flink Studio 노트북 내에서

수행하거나 외부에서 참조되는 애플리케이션 jar 파일로 수행할 수 있습니다. 사용자 정의 함수를 활용하면 스트리밍 데이터를 통해 수행할 수 있는 변환 또는 데이터 강화를 단순화할 수 있습니다.

노트북에서는 개인 전화번호를 익명화하는 기능이 있는 간단한 Java 애플리케이션 항아리를 참조하게 될 것입니다. 노트북 내에서 사용할 Python 또는 Scala UDF를 작성할 수도 있습니다. 애플리케이션 jar를 Pyflink 노트북으로 가져오는 기능을 강조하기 위해 Java 애플리케이션 jar를 선택했습니다.

환경 설정

이 가이드를 따르고 스트리밍 데이터와 상호 작용하려면 AWS CloudFormation 스크립트를 사용하여 다음 리소스를 시작해야 합니다.

- 소스 및 대상 Kinesis 데이터 스트림
- Glue 데이터베이스
- IAM 역할
- Managed Service for Apache Flink 애플리케이션
- Managed Service for Apache Flink 애플리케이션을 시작하기 위한
- 위의 Lambda 함수를 실행하기 위한 Lambda 역할
- Lambda 함수를 간접적으로 호출할 사용자 지정 리소스

여기에서 AWS CloudFormation 템플릿을 다운로드합니다.

AWS CloudFormation 스택 생성

1. 로 이동하여 서비스 목록에서 CloudFormation을 AWS Management Console 선택합니다.
2. CloudFormation 페이지에서 스택을 선택하고 새 리소스로 스택 생성(표준)을 선택합니다.
3. 스택 생성 페이지에서 템플릿 파일 업로드를 선택한 다음 이전에 다운로드한 `kda-flink-udf.yml`을 선택합니다. 파일을 업로드하고 다음을 선택합니다.
4. 기억하기 쉽도록 템플릿에 `kinesis-UDF` 같은 명칭을 지정하고 다른 명칭을 원하면 input-stream과 같은 입력 파라미터를 업데이트하십시오. Next(다음)를 선택합니다.
5. 스택 옵션 구성 페이지에서 원하는 경우 태그를 추가하고 다음을 선택합니다.
6. 검토 페이지에서 IAM 리소스 생성을 허용하는 확인란을 선택한 다음 제출을 선택합니다.

시작하려는 리전에 따라 AWS CloudFormation 스택을 시작하는 데 10~15분이 걸릴 수 있습니다. 전체 스택의 `CREATE_COMPLETE` 상태가 확인되면 계속할 준비가 된 것입니다.

Managed Service for Apache Flink Studio 노트북을 사용하여 작업하기

Kinesis Data Analytics용 Studio 노트북을 사용하면 실시간으로 대화형 방식으로 데이터 스트림을 쿼리하고 표준 SQL, Python 및 Scala를 사용하여 스트림 처리 애플리케이션을 쉽게 구축하고 실행할 수 있습니다. 에서 몇 번의 클릭만으로 서버리스 노트북을 시작하여 데이터 스트림을 쿼리하고 몇 초 만에 결과를 얻을 AWS Management Console수 있습니다.

노트북은 웹 기반 개발 환경입니다. 노트북을 사용하면 Apache Flink에서 제공하는 고급 데이터 스트림 처리 기능과 결합된 간단한 대화형 개발 환경을 얻을 수 있습니다. Studio 노트북은 Apache Zeppelin 기반 노트북을 사용하며 Apache Flink를 스트림 처리 엔진으로 사용합니다. Studio 노트북은 이러한 기술을 원활하게 결합하여 모든 기술을 갖춘 개발자가 데이터 스트림에 대한 고급 분석을 이용할 수 있도록 합니다.

Apache Zeppelin은 Studio 노트북에 다음을 포함한 완벽한 분석 도구 제품군을 제공합니다:

- 데이터 시각화
- 데이터를 파일로 내보내기
- 보다 쉬운 분석을 위한 출력 형식 제어

노트북 사용하기

1. 로 이동하여 서비스 목록에서 Amazon Kinesis를 AWS Management Console 선택합니다.
2. 왼쪽 탐색 페이지에서 Analytics 애플리케이션을 선택한 다음 Studio 노트북을 선택합니다.
3. KinesisDataAnalyticsStudio 노트북이 실행 중인지 확인하십시오.
4. 노트북을 선택한 다음 Apache Zeppelin에서 열기를 선택합니다.
5. 데이터를 읽고 Kinesis 스트림으로 로드하는 데 사용할 Data Producer Zeppelin Notebook 파일을 다운로드하십시오.
6. `Data Producer` Zeppelin Notebook을 가져오십시오. 노트북 코드의 입력 `STREAM_NAME` 및 `REGION`을 수정해야 합니다. 입력 스트림 명칭은 AWS CloudFormation 스택 출력에서 찾을 수 있습니다.
7. 입력 Kinesis Data Stream에 샘플 데이터를 삽입하려면 이 단락 실행 버튼을 선택하여 Data Producer 노트북을 실행합니다.
8. 샘플 데이터가 로드되는 동안 MaskPhoneNumber-Interactive 노트북을 다운로드하면 입력 데이터를 읽고, 입력 스트림에서 전화번호를 익명화하고, 익명화된 데이터를 출력 스트림에 저장할 수 있습니다.

9. `MaskPhoneNumber-interactive` Zeppelin 노트북을 가져오십시오.
10. 노트북의 각 단락을 실행하십시오.

    a. 단락 1에서는 사용자 정의 함수를 가져와서 전화번호를 익명화합니다.

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

    b. 다음 단락에서는 입력 스트림 데이터를 읽을 수 있는 메모리 내 표를 생성합니다. 스트림 이름과 AWS 리전이 올바른지 확인합니다.

    ```
    %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. 데이터가 메모리 내 표에 로드되었는지 확인하십시오.

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

    d. 사용자 정의 함수를 호출하여 전화번호를 익명화합니다.

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

    e. 이제 전화번호가 마스킹되었으니 마스킹된 번호로 뷰를 만드십시오.

```
%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. 데이터를 확인합니다.

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

g. 출력 Kinesis 스트림에 대한 메모리 내 표를 생성합니다. 스트림 이름과 AWS 리전이 올바른지 확인합니다.

```
%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. 대상 Kinesis 스트림에 업데이트된 레코드를 삽입합니다.

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

i. 대상 Kinesis 스트림의 데이터를 보고 확인합니다.

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

노트북을 애플리케이션으로 승격

노트북 코드를 대화형 방식으로 테스트했으니 이제 코드를 지속 가능한 상태의 스트리밍 애플리케이션으로 배포해 보겠습니다. Amazon S3에서 코드를 저장할 위치를 지정하려면 먼저 애플리케이션 구성을 수정해야 합니다.

1. 에서 노트북을 AWS Management Console선택하고 애플리케이션 구성으로 배포 - 선택 사항에서 편집을 선택합니다.
2. Amazon S3의 코드 대상에서AWS CloudFormation 스크립트로 생성된 Amazon S3 버킷을 선택합니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.
3. 현재 상태로는 메모를 홍보할 수 없습니다. 내보내기를 시도하면 `Select` 명령문이 지원되지 않으므로 오류가 발생합니다. 이 문제를 방지하려면 MaskPhoneNumber-Streaming Zeppelin Notebook을 다운로드하십시오.
4. `MaskPhoneNumber-streaming` Zeppelin Notebook을 가져오십시오.
5. 노트를 열고 KinesisDataAnalyticsStudio를 위한 조치를 선택합니다.
6. MaskPhoneNumber-Streaming 구축을 선택하고 S3로 내보내기를 선택합니다. 애플리케이션 명칭을 바꾸고 특수 문자를 포함하지 않도록 하십시오.
7. 구축 및 내보내기를 선택합니다. 스트리밍 애플리케이션을 설정하는 데 몇 분 정도 걸립니다.
8. 빌드가 완료되면 AWS 콘솔을 사용하여 배포를 선택합니다.
9. 다음 페이지에서 설정을 검토하고 올바른 IAM 역할을 선택했는지 확인하십시오. 다음으로 스트리밍 애플리케이션 생성을 선택합니다.
10. 몇 분 후 스트리밍 애플리케이션이 성공적으로 생성되었다는 메시지가 표시됩니다.

지속 가능한 상태 및 제한이 있는 애플리케이션을 배포하는 방법에 대한 자세한 설명은 지속 가능한 상태의 애플리케이션으로 배포를 참조하십시오.

정리

원하는 경우 이제 AWS CloudFormation 스택을 제거할 수 있습니다. 이렇게 하면 이전에 설정한 모든 서비스가 제거됩니다.

Amazon Kinesis Data Analytics for SQL 애플리케이션이란 무엇인가?

Amazon Kinesis Data Analytics for SQL 애플리케이션을 사용하면 표준 SQL을 이용하여 스트리밍 데이터를 처리하고 분석할 수 있습니다. 이 서비스를 사용하면 스트리밍 소스에 대해 강력한 SQL 코드를 작성하고 실행하여 시계열 분석을 수행하고, 실시간 대시보드를 제공하고, 실시간 지표를 생성할 수 있습니다.

Kinesis Data Analytics를 시작하려면 스트리밍 데이터를 지속적으로 읽고 처리하는 Kinesis Data Analytics 애플리케이션을 생성합니다. 이 서비스는 Amazon Kinesis Data Streams 및 Amazon Data Firehose 스트리밍 소스의 데이터 수집을 지원합니다. 그런 다음 대화형 편집기를 사용하여 SQL 코드를 작성하고 라이브 스트리밍 데이터로 테스트합니다. Kinesis Data Analytics가 결과를 전송하려는 목적지를 구성할 수도 있습니다.

Kinesis Data Analytics는 Amazon Data Firehose(Amazon S3, Amazon Redshift, Amazon OpenSearch Service 및 Splunk) 및 AWS Lambda Amazon Kinesis Data Streams를 대상으로 지원합니다.

Amazon Kinesis Data Analytics를 언제 사용해야 하는가?

Amazon Kinesis Data Analytics를 사용하면 거의 실시간으로 데이터를 지속적으로 읽고 처리하고 저장하는 SQL 코드를 신속하게 작성할 수 있습니다. 스트리밍 데이터에 대해 표준 SQL 쿼리를 사용하면 데이터를 변환하고 쉽게 파악할 수 있는 애플리케이션을 구성할 수 있습니다. 다음은 Kinesis Data Analytics 사용에 대한 몇 가지 시나리오 예입니다:

- 시계열 분석 생성 – 여러 시간 창에 걸쳐 지표를 계산한 다음 Kinesis 데이터 전송 스트림을 통해 스트림 값을 Amazon S3 또는 Amazon Redshift로 전송할 수 있습니다.
- 실시간 대시보드 입력 – 집계 및 처리된 스트리밍 데이터 결과를 다운 스트림하여 실시간 대시보드에 제공할 수 있습니다.
- 실시간 지표 생성 – 실시간 모니터링, 알림 및 경보에 사용할 맞춤 지표와 트리거를 생성할 수 있습니다.

Kinesis Data Analytics에서 지원되는 SQL 언어 요소에 대한 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조를 참조하십시오.

Amazon Kinesis Data Analytics를 처음 사용하십니까?

Amazon Kinesis Data Analytics를 처음 사용한다면, 다음 섹션을 순서대로 읽어보기를 권장합니다:

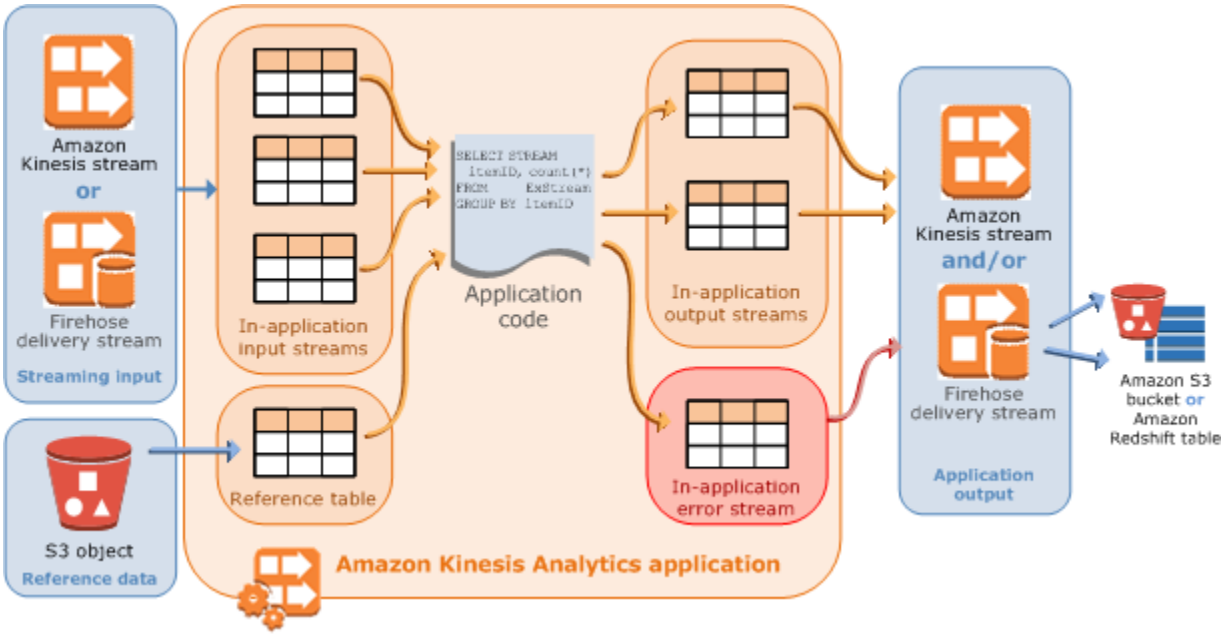
1. 이 가이드의 동작 원리 섹션을 읽으십시오. 이 섹션에서는 엔드 투 엔드 경험을 만드는 데에 함께 사용하는 다양한 Kinesis Data Analytics 구성 요소를 소개합니다. 자세한 설명은 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 섹션을 참조하십시오.
2. 시작 연습을 수행해 보십시오. 자세한 설명은 Amazon Kinesis Data Analytics for SQL 애플리케이션 시작하기 섹션을 참조하십시오.
3. 스트리밍 SQL 개념을 읽어 보십시오. 자세한 설명은 스트리밍 SQL 개념 섹션을 참조하십시오.
4. 추가 예를 시도해 보십시오. 자세한 설명은 Kinesis Data Analytics for SQL 예 섹션을 참조하십시오.

Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 한도를 참조하십시오.

Amazon Kinesis Data Analytics에서 애플리케이션은 사용자 계정에서 생성할 수 있는 기본 리소스입니다. AWS Management Console 또는 Kinesis Data Analytics API를 사용하여 애플리케이션을 생성하고 관리할 수 있습니다. Kinesis Data Analytics는 애플리케이션 관리용 API 작업을 제공합니다. API 작업 목록은 작업 섹션을 참조하십시오.

Kinesis Data Analytics 애플리케이션은 실시간으로 스트리밍 데이터를 연속적으로 읽고 처리합니다. 수신 스트리밍 데이터를 처리하고 출력을 생성하도록 사용자가 SQL을 사용하여 애플리케이션 코드를 작성합니다. 그러면, Kinesis Data Analytics가 출력을 구성 목적지에 작성합니다. 다음 다이어그램은 일반적인 애플리케이션 아키텍처를 보여 줍니다.



각 애플리케이션은 명칭, 설명, 버전 ID 및 상태를 갖습니다. 애플리케이션을 처음 생성하면 Amazon Kinesis Data Analytics가 버전 ID를 할당합니다. 임의의 애플리케이션 구성을 업데이트하면 이 버전 ID

도 업데이트됩니다. 예를 들어 입력 구성을 추가하거나, 참조 데이터 소스를 추가 또는 삭제하거나, 출력 구성을 추가 또는 삭제하거나, 애플리케이션 코드를 업데이트하면 Kinesis Data Analytics 가 현재 애플리케이션 버전 ID를 업데이트합니다. 또한 Kinesis Data Analytics는 애플리케이션이 생성되고 마지막으로 업데이트된 시기에 대한 타임스탬프를 유지합니다.

이러한 기본 속성에 더해, 각 애플리케이션은 다음으로 구성됩니다:

- 입력 – 애플리케이션의 스트리밍 소스입니다. Kinesis 데이터 스트림 또는 Firehose 데이터 전송 스트림을 스트리밍 소스로 선택할 수 있습니다. 입력 구성에서 스트리밍 소스를 애플리케이션 내 입력 스트림에 매핑합니다. 애플리케이션 내 스트림은 지속적으로 업데이트되는 표와 같습니다. 이 표에서 `SELECT` 및 `INSERT SQL` 작업을 수행할 수 있습니다. 중간 쿼리 결과를 저장하도록 애플리케이션 코드에서 애플리케이션 내 스트림을 추가로 생성할 수 있습니다.

  선택적으로 스트리밍 소스를 복수의 애플리케이션 내 스트림에 분할하여 처리량을 향상시킬 수 있습니다. 자세한 정보는 한도 및 애플리케이션 입력 구성 섹션을 참조하십시오.

  Amazon Kinesis Data Analytics는 각 애플리케이션 스트림 내에 타임스탬프와 ROWTIME 열이라는 타임스탬프 열을 제공합니다. 시간 기반 윈도우 쿼리에서 이 열을 사용할 수 있습니다. 자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.

  선택적으로 참조 데이터 소스를 구성하여 애플리케이션 내에서의 입력 데이터 스트림을 보강할 수 있습니다. 그 결과물이 애플리케이션 내 참조 표입니다. 참조 데이터는 S3 버킷에 객체로 저장해야 합니다. 애플리케이션이 시작하면 Amazon Kinesis Data Analytics이 Amazon S3 객체를 읽고 애플리케이션 내 표를 생성합니다. 자세한 설명은 애플리케이션 입력 구성 섹션을 참조하십시오.

- 애플리케이션 코드 – 입력을 처리하고 출력을 생산하는 일련의 SQL 문입니다. 애플리케이션 내 스트림과 참조 표에 대해 SQL 문을 작성할 수 있습니다. JOIN 쿼리를 작성하여 이 두 소스의 데이터를 결합할 수도 있습니다.

  Kinesis Data Analytics에서 지원되는 SQL 언어 요소에 대한 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조를 참고하십시오.

가장 간단한 형태의 애플리케이션 코드는 스트리밍 입력에서 선택하여 결과를 스트리밍 출력에 삽입하는 단일 SQL 문이 될 수 있습니다. 또한, 하나의 출력이 다음 SQL 문의 입력으로 공급되는 일련의 SQL 문이 될 수도 있습니다. 뿐만 아니라 입력 스트림을 여러 스트림으로 분할하는 애플리케이션 코드를 작성할 수 있습니다. 그런 다음 추가 쿼리를 적용하여 이러한 스트림을 처리할 수 있습니다. 자세한 설명은 애플리케이션 코드 섹션을 참조하십시오.

- 출력 – 애플리케이션 내 코드에서 쿼리 결과가 애플리케이션 내 스트림으로 향합니다. 중간 결과를 보관하도록 애플리케이션 코드에서 하나 이상의 애플리케이션 내 스트림을 추가로 생성할 수 있습니다. 그런 다음 애플리케이션 출력(애플리케이션 내 출력 스트림이라고도 함)을 외부 대상에 보관하는 애플리케이션 내 스트림에 데이터를 유지하도록 애플리케이션 출력을 선택적으로 구성할 수 있습니다. 외부 대상은 Kinesis 데이터 스트림 또는 Firehose 전송 스트림이 될 수 있습니다. 이러한 목적지에 대해 다음을 참조하십시오:
  - Amazon S3, Amazon Redshift 또는 Amazon OpenSearch Service (OpenSearch Service)에 결과를 작성하도록 Firehose 전송 스트림을 구성할 수 있습니다.
  - 애플리케이션 출력을 Amazon S3 또는 Amazon Redshift로 맞춤 목적지로 할 수도 있습니다. 이렇게 하려면 출력 구성에서 Kinesis 데이터 스트림을 목적지로 지정해야 합니다. 그런 다음 스트림을 폴링하고 Lambda 함수를 호출 AWS Lambda 하도록를 구성합니다. 사용자의 Lambda 함수 코드는 스트림 데이터를 입력으로 수신합니다. Lambda 함수 코드에서 수신 데이터를 맞춤 대상에 작성할 수 있습니다. 자세한 내용은 Amazon Kinesis Data Analytics AWS Lambda 에서 사용을 참조하세요.

  자세한 내용은 애플리케이션 출력 구성 단원을 참조하십시오.

또한 다음 사항에 유의하십시오:

- Amazon Kinesis Data Analytics는 스트리밍 소스로부터 오는 레코드를 읽고 애플리케이션 출력을 외부 대상에 작성할 수 있는 권한이 필요합니다. IAM 역할을 사용하여 그와 같은 권한을 부여합니다.
- Kinesis Data Analytics는 각 애플리케이션의애플리케이션 내 오류 스트림을 자동으로 제공합니다. 애플리케이션이 특정 레코드를 처리하는 동안 문제가 발생할 경우(예: 유형 불일치 또는 지연 도착) 해당 레코드는 오류 스트림에 작성됩니다. 추가 평가를 위해 오류 스트림 데이터가 외부 목적지 향하

도록 애플리케이션 출력을 Kinesis Data Analytics로 가도록 구성할 수 있습니다. 자세한 설명은 오류 처리 섹션을 참조하십시오.

- Amazon Kinesis Data Analytics는 애플리케이션 출력 레코드가 구성 목적지에 작성되도록 합니다. 애플리케이션 중단을 경험하더라도 '최소 1회' 처리 및 전송 모델을 사용합니다. 자세한 설명은 애플리케이션 출력을 외부 대상에 유지하기 위한 전송 모델 섹션을 참조하십시오.

주제

- 애플리케이션 입력 구성
- 애플리케이션 코드
- 애플리케이션 출력 구성
- 오류 처리
- 처리량 증가를 위해 애플리케이션 용량을 자동으로 확장 또는 축소
- 태그 지정 사용

애플리케이션 입력 구성

Amazon Kinesis Data Analytics 애플리케이션이 단일 스트리밍 소스로부터 입력을 수신할 수도 있고, 선택적으로 하나의 참조 데이터 소스를 사용할 수도 있습니다. 자세한 설명은 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 섹션을 참조하십시오. 이 주제 섹션에서는 애플리케이션 입력 소스에 대해 설명합니다.

주제

- 스트리밍 소스 구성
- 참조 소스 구성
- JSONPath로 작업하기
- 스트리밍 소스 요소를 SQL 입력 열에 매핑하기
- 스트리밍 데이터에 대해 스키마 검색 기능 사용
- 정적 데이터에 대해 스키마 검색 기능 사용
- Lambda 함수를 사용하여 데이터 사전 처리
- 입력 스트림 병렬화를 통한 처리량 증대

스트리밍 소스 구성

애플리케이션을 생성하는 시점에 스트리밍 소스를 지정합니다. 애플리케이션을 만든 후 입력을 수정할 수도 있습니다. Amazon Kinesis Data Analytics는 애플리케이션에 다음과 같은 스트리밍 소스를 지원합니다.

- Kinesis 데이터 스트림
- Firehose 전송 스트림

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. `KinesisFirehoseInput`과 함께 Kinesis Data Analytics for SQL 애플리케이션을 사용하는 기존 고객은 Kinesis Data Analytics를 사용하여 기존 계정 내에서 `KinesisFirehoseInput`와 함께 애플리케이션을 계속 추가할 수 있습니다. 기존 고객이 Kinesis Data Analytics for SQL 애플리케이션을 `KinesisFirehoseInput`과 함께 사용하여 새 계정을 생성하려는 경우 서비스 한도 증가 양식을 통해 사례를 생성할 수 있습니다. 자세한 정보는 AWS Support Center 섹션을 참조하십시오. 프로덕션으로 승격하기 전에 항상 새 애플리케이션을 테스트하는 것이 좋습니다.

> **Note**
>
> Kinesis 데이터 스트림이 암호화되었다면 Kinesis Data Analytics는 추가 구성 없이도 암호화된 시스템 내의 데이터에 액세스할 수 있습니다. Kinesis Data Analytics는 Kinesis Data Streams에서 읽은 암호화되지 않은 데이터는 저장하지 않습니다. 자세한 설명은 Kinesis Data Streams을 위한 서버측 암호화란 무엇인가?를 참조하십시오.

Kinesis Data Analytics는 새로운 데이터에 대해 스트리밍 소스를 지속적으로 폴링하여 입력 구성에 따라 애플리케이션 내 스트림에서 수집합니다.

> **Note**
>
> 애플리케이션의 입력으로 Kinesis 스트림을 추가해도 스트림의 데이터에는 영향을 미치지 않습니다. Firehose 전송 스트림을 비롯한 다른 리소스도 동일한 Kinesis 스트림에 액세스하면

Firehose 전송 스트림과 Kinesis Data Analytics 애플리케이션 모두 동일한 데이터를 수신합니다. 그러나 처리량 및 조절이 영향을 받을 수 있습니다.

그러면 애플리케이션 코드가 이를 애플리케이션 내 스트림에서 쿼리할 수 있습니다. 입력 구성의 일부로 다음을 제공합니다:

- 스트리밍 소스 – 스트림의 Amazon 리소스 이름(ARN)과 Kinesis Data Analytics가 사용자를 대신하여 스트림에 액세스할 수 있는 권한을 주는 IAM 역할을 제공합니다.
- 애플리케이션 내 스트림 명칭의 접두사 – 애플리케이션을 시작할 때 Kinesis Data Analytics가 지정 애플리케이션 내 스트림을 생성합니다. 그러면 애플리케이션 코드에서 이 명칭을 사용하여 애플리케이션 내 스트림에 액세스합니다.

  선택적으로 스트리밍 소스를 복수의 애플리케이션 내 스트림에 매핑할 수 있습니다. 자세한 설명은 한도 섹션을 참조하십시오. 이 경우 Amazon Kinesis Data Analytics가 다음과 같은 명칭을 지니는 지정된 수의 애플리케이션 내 스트림을 생성합니다: `###_001`, `###_002`와 `###_003`. 기본 설정으로 Kinesis Data Analytics는 스트리밍 소스를 `###_001`라고 하는 하나의 애플리케이션 내 스트림에 매핑합니다.

  애플리케이션 내 스트림에 행을 삽입할 수 있는 속도에는 한도가 있습니다. 따라서 Kinesis Data Analytics는 그와 같은 애플리케이션 내 스트림을 여러 개 지원하므로 레코드를 훨씬 더 빠른 속도로 애플리케이션에 가져올 수 있습니다. 애플리케이션이 스트리밍 소스에서 데이터를 따라 잡지 못할 경우 병렬 처리 단위를 추가하여 성능을 개선할 수 있습니다.
- 스키마 매핑 – 스트리밍 소스의 레코드 형식(JSON, CSV)에 대해 명시합니다. 또한 스트림 내 각 레코드를 생성되는 애플리케이션 내 스트림 내 열로 매핑하는 방식에 대해서도 설명합니다. 여기에 열 명칭과 데이터 유형을 입력합니다.

> **Note**
>
> Kinesis Data Analytics는 입력 애플리케이션 내 스트림을 생성할 때 식별자(스트림 명칭 및 열 명칭) 주위에 물음표를 추가합니다. 이 스트림 및 열을 쿼리할 때는 대소문자를 똑같이 구분하여 따옴표로 지정해야 합니다. 식별자에 대한 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 식별자를 참조하십시오.

Amazon Kinesis Data Analytics 콘솔에서 애플리케이션을 생성하고 입력을 구성할 수 있습니다. 그런 다음에는 콘솔이 필요한 API 호출을 수행합니다. 새 애플리케이션 API를 생성하거나 기존

애플리케이션에 입력 구성을 추가할 때 애플리케이션 입력을 구성할 수 있습니다. 자세한 정보는 CreateApplication 및 AddApplicationInput 섹션을 참조하십시오. 다음은 `Createapplication` API 요청 본문의 입력 구성 부분입니다.

```
"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"
       }
   ]
```

참조 소스 구성

또한 선택적으로 참조 데이터 소스를 기존 애플리케이션에 추가하여 스트리밍 소스에서 들어오는 데이터를 보강할 수 있습니다. 참조 데이터는 Amazon S3 버킷에 객체로 저장해야 합니다. 애플리케이션이 시작되면 Amazon Kinesis Data Analytics이 Amazon S3 객체를 읽고 애플리케이션 내 참조 표를 생성합니다. 그러면 애플리케이션 코드에서 이를 애플리케이션 내 스트림에 조인할 수 있습니다.

지원되는 형식(CSV, JSON)을 사용하여 참조 데이터를 Amazon S3 객체에 저장합니다. 예를 들어, 애플리케이션이 주식 주문에 대한 분석을 수행한다고 가정해 보겠습니다. 스트리밍 소스에 대해 다음이 레코드 형식을 취합니다.

```
Ticker, SalePrice, OrderId

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

이 경우, 참조 데이터 소스를 유지하여 회사 명칭과 같은 각 주식 티커에 관한 상세 정보를 제공하는 것을 고려할 수 있을 것입니다.

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

API 또는 콘솔을 사용하여 애플리케이션 참조 데이터 소스를 추가할 수 있습니다. Amazon Kinesis Data Analytics는 참조 데이터 소스를 관리하기 위해 다음과 같은 API 작업을 제공합니다.

- AddApplicationReferenceDataSource
- UpdateApplication

콘솔을 사용하여 참조 데이터를 추가하는 방법에 대한 자세한 설명은 예: 참조 데이터를 Kinesis Data Analytics 애플리케이션에 추가 섹션을 참조하십시오.

다음 사항에 유의하세요.

- 애플리케이션이 실행 중이면 Kinesis Data Analytics는 애플리케이션 내 참조 표를 생성한 다음 바로 참조 표를 로드합니다.

- 애플리케이션이 실행 중이 아니면 (예: 대기 모드에 있을 때) Kinesis Data Analytics는 업데이트된 입력 구성만 저장합니다. 애플리케이션 실행이 시작되면 Kinesis Data Analytics가 참조 표를 애플리케이션에 표로 로드합니다.

Kinesis Data Analytics가 애플리케이션 내 참조 표를 생성한 후 데이터를 새로 고치려고 한다고 가정해 보겠습니다. 아마도 Amazon S3 객체를 업데이트했거나 다른 Amazon S3 객체를 사용하려 하려고 할 것입니다. 이 경우, UpdateApplication을 명시적으로 호출하거나, 콘솔에서 작업, 참조 데이터 표 동기화를 선택할 수 있습니다. Kinesis Data Analytics는 애플리케이션 내 참조 표를 자동으로 새로 고치지 않습니다.

참조 데이터 소스로 생성할 수 있는 Amazon S3 객체의 크기에는 한도가 있습니다. 자세한 설명은 한도 섹션을 참조하십시오. 객체 크기가 한도를 초과하는 경우 Kinesis Data Analytics는 데이터를 로드할 수 없습니다. 실행 중에 애플리케이션 상태는 표시되지만, 데이터는 읽혀지지 않습니다.

참조 데이터 소스를 추가할 때 다음의 정보를 제공합니다.

- S3 버킷 및 객체 키 명칭 – 버킷 명칭 및 객체 키 외에도 Kinesis Data Analytics가 사용자를 대신하여 객체를 읽을 권한을 줄 IAM 역할도 제공합니다.
- 애플리케이션 내 참조 표 명칭 – Kinesis Data Analytics가 이 애플리케이션 내 표를 생성하고 Amazon S3 객체를 읽어서 표를 채웁니다. 이는 애플리케이션 코드에서 지정한 표 명칭입니다.
- 스키마 매핑 – 레코드 형식(JSON, CSV), Amazon S3 객체에 저장된 데이터의 인코딩을 기술합니다. 각 데이터 요소가 애플리케이션 내 참조 표에 있는 열에 어떻게 매핑되는지도 기술합니다.

다음은 `AddApplicationReferenceDataSource` API 요청의 요청 본문입니다.

```
{
    "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"
    }
}
```

JSONPath로 작업하기

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 제한을 참조하십시오.

JSONPath는 JSON 객체의 요소를 쿼리하는 표준화된 방법입니다. JSONPath는 경로 표현식을 사용하여 JSON 문서에서 요소, 중첩된 요소 및 배열을 탐색합니다. JSON에 대한 자세한 설명은 JSON 소개를 참조하십시오.

Amazon Kinesis Data Analytics는 애플리케이션의 소스 스키마에 JSONPath 표현식을 사용하여 스트리밍 소스에서 JSON 형식의 데이터가 포함된 데이터 요소를 식별합니다.

스트리밍 데이터를 애플리케이션의 입력 스트림에 매핑하는 방법에 대한 자세한 설명은 the section called “스트리밍 소스 요소를 SQL 입력 열에 매핑하기” 섹션을 참조하십시오.

JSONPath로 JSON 요소에 액세스하기

JSONPath 표현식을 사용하여 JSON 형식 데이터의 다양한 요소에 액세스하는 방법을 알아볼 수 있습니다. 이 섹션의 예에서는 소스 스트림에 다음 JSON 레코드가 포함된 것으로 가정합니다.

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

JSON 요소에 액세스

JSONPath를 사용하여 JSON 데이터에 있는 요소를 쿼리하려면 다음의 구문을 사용합니다. 여기에서, `$`는 데이터 계층의 루트를 나타내고 `elementName`은 쿼리에 대한 요소 노드의 명칭입니다.

```
$.elementName
```

다음 표현식은 앞의 JSON 예제에 있는 `customerName` 요소를 쿼리하는 것이다.

```
$.customerName
```

앞의 표현식은 앞선 JSON 레코드로부터 다음을 반환합니다.

```
John Doe
```

> **Note**
>
> 경로 표현식은 대/소문자를 구분합니다. 표현식 `$.customername`은 앞선 JSON 예로부터 `null`을 반환합니다.

> **Note**
>
> 경로 표현식에서 지정한 위치에 요소가 나타나지 않을 경우, 표현식은 `null`을 반환합니다. 다음 표현식은 앞선 JSON 예로부터 `null`을 반환하는데, 일치하는 요소가 없기 때문입니다.
>
> ```
> $.customerId
> ```

중첩된 JSON 요소에 대한 액세스

중첩된 JSON 요소를 쿼리하려면 다음 문구를 사용합니다.

```
$.parentElement.element
```

다음 표현식은 앞의 JSON 예제에 있는 `city` 요소를 쿼리하는 것이다.

```
$.address.city
```

앞의 표현식은 앞선 JSON 레코드로부터 다음을 반환합니다.

```
Anytown
```

다음 문구를 사용하여 하위 요소를 쿼리할 수 있습니다.

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

다음 표현식은 앞의 JSON 예제에 있는 `street` 요소를 쿼리하는 것이다.

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

앞의 표현식은 앞선 JSON 레코드로부터 다음을 반환합니다.

```
AnyStreet
```

배열 액세스

다음과 같은 방법으로 JSON 배열의 데이터에 액세스할 수 있습니다.

- 배열의 모든 요소를 단일 행으로 검색합니다.
- 배열의 각 요소를 별도의 행으로 검색합니다.

배열의 모든 요소를 단일 행으로 검색

배열의 전체 콘텐츠를 단일 행으로 쿼리하려면 다음의 구문을 사용합니다.

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

다음 표현식은 이 섹션에 사용된 위의 JSON 예에 있는 `orders` 요소의 전체 콘텐츠를 쿼리합니다. 단일 열 단일 행에 있는 배열 콘텐츠를 반환합니다.

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

앞의 표현식은 이 섹션에 사용된 예 JSON 레코드로부터 다음을 반환합니다.

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

배열의 모든 요소를 별도의 행으로 검색

배열의 개별 요소를 별도의 행으로 쿼리하려면 다음의 구문을 사용합니다.

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

다음 표현식은 위의 JSON 예에 있는 `orderId` 요소를 쿼리하는 것으로 각 배열 요소를 별도의 행으로 반환합니다.

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

위의 표현식은 앞에 나온 JSON 레코드로부터 다음을 반환합니다. 이때 각 데이터 항목은 별도의 행으로 반환됩니다.

23284
63122
77284

> **Note**
>
> 비배열 요소를 쿼리하는 표현식이 개별 배열 요소를 쿼리하는 스키마에 포함되어 있는 경우, 비배열 요소는 해당 배열에 있는 각 요소에 대해 반복됩니다. 예를 들어, 위의 JSON 예에 대한 스키마가 다음의 표현식을 포함하고 있다고 가정해 보겠습니다.
>
> - $.customerName
> - $.orders[0:].orderID
>
> 이 경우, 예 입력 스트림 요소로부터 반환된 데이터 행은 다음과 비슷할 것입니다. 이때 모든 `orderId` 요소에 대해 `name` 요소가 반복됩니다.
>
> | | |
> |---|---|
> | John Doe | 23284 |
> | John Doe | 63122 |
> | John Doe | 77284 |

> **Note**
>
> Amazon Kinesis Data Analytics에서 배열 표현식에는 다음 제한이 적용됩니다:
>
> - 배열 표현식에서는 한 가지 레벨의 역참조만 지원됩니다. 다음의 표현식 형식은 지원되지 않습니다.
>
> ```
> $.arrayObject[0:].element[0:].subElement
> ```

- 한 가지 배열만 스키마에서 평면화할 수 있습니다. 복수의 어레이가 참조되어 – 어레이에 모든 요소가 포함된 하나의 행으로 반환될 수 있습니다. 그러나 하나의 어레이만 개별 행으로 반환되는 요소의 각각을 가질 수 있습니다.

  다음 형식의 요소를 포함하는 스키마는 유효합니다. 이 형식은 두 번째 어레이의 콘텐츠를 단일 열로 반환합니다. 첫 번째 어레이에 있는 모든 요소에 대해 반복됩니다.

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

  다음 형식의 요소를 포함하는 스키마는 유효하지 않습니다.

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

기타 고려 사항

JSONPath 작업 시 추가적인 고려 사항은 다음과 같습니다.

- 애플리케이션 스키마의 JSONPath 표현식에 있는 개별 요소가 배열에 액세스하지 않는 경우, 처리된 각 JSON 레코드에 대해 애플리케이션의 입력 스트림에 단일 행이 생성됩니다.
- 배열을 평면화할 때(즉, 해당 요소가 개별 행으로 반환됨) 누락된 요소가 있을 경우 애플리케이션 내 스트림에 null 값이 생성됩니다.
- 배열은 항상 하나 이상의 행에 대해 평면화됩니다. 어떤 값도 반환되지 않는 경우(즉, 어레이가 비어 있거나 요소에 대한 쿼리가 없는 경우) 모든 null 값인 단일 행이 반환됩니다.

  다음 표현식은 앞선 JSON 예로부터 null 값을 지닌 레코드를 반환하는데, 지정된 경로에 일치하는 요소가 없기 때문입니다.

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

  앞의 표현식은 앞선 JSON 예 레코드로부터 다음을 반환합니다.

null
null

```
null
```

관련 항목

- JSON 소개

스트리밍 소스 요소를 SQL 입력 열에 매핑하기

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 한도를 참조하십시오.

Amazon Kinesis Data Analytics에서는 표준 SQL을 사용하여 JSON 또는 CSV 형식의 스트리밍 데이터를 처리하고 분석할 수 있습니다.

- 스트리밍 CSV 데이터를 처리하고 분석하려면 입력 스트림의 열에 대해 열 명칭과 데이터 유형을 할당합니다. 애플리케이션이 입력 스트림에서 열 정의에 따라 순서대로 열을 하나씩 가져옵니다.

  애플리케이션 입력 스트림에 있는 열을 모두 포함할 필요는 없지만 소스 스트림에서 열을 건너뛸 수는 없습니다. 예를 들어, 5개 요소를 포함하는 입력 스트림으로부터 첫 3개 열을 가져올 수 있으나 1번, 2번 및 4번 열만을 가져올 수는 없습니다.
- 스트리밍 JSON 데이터를 처리하고 분석하려면 JSONPath 표현식을 사용하여 JSON 요소를 스트리밍 소스로부터 입력 스트림의 SQL 열로 매핑합니다. Amazon Kinesis Data Analytics에서 JSONPath을 사용하는 방법은 JSONPath로 작업하기을 참조하십시오. SQL 표에 있는 열은 JSON 유형으로부터 매핑된 데이터 유형을 가집니다. 지원되는 데이터 유형은 데이터 유형 섹션을 참조하십시오. JSON 데이터의 SQL 데이터 변환에 관한 상세 내용은 JSON 데이터 유형을 SQL 데이터 유형으로 매핑하기 섹션을 참조하십시오.

입력 스트림 구성 방법에 관한 자세한 정보는 애플리케이션 입력 구성 섹션을 참조하십시오.

JSON 데이터를 SQL 열에 매핑하기

AWS Management Console 또는 Kinesis Data Analytics API를 사용하여 JSON 요소를 입력 열에 매핑할 수 있습니다.

- 콘솔을 사용하여 요소를 열에 매핑하는 방법은 스키마 편집기로 작업을 참조하십시오.
- Kinesis Data Analytics API를 사용하여 요소를 열에 매핑하는 방법은 다음 섹션을 참조하십시오.

JSON 요소를 애플리케이션 내 입력 스트림에 있는 열에 매핑하려면 각 열에 대해 다음의 정보가 스키마에 필요합니다.

- 소스 표현식: 열의 데이터 위치를 식별하는 JSONPath 식입니다.
- 열 명칭: SQL 쿼리에서 데이터를 참조하는 데 사용하는 명칭입니다.
- 데이터 형식: 열의 SQL 데이터 형식입니다.

API 사용

스트리밍 소스의 요소를 입력 열로 매핑할 때 Kinesis Data Analytics API CreateApplication 작업을 사용할 수 있습니다. 애플리케이션 내 스트림을 생성하기 위해서는 SQL에 사용되는 스키마화된 버전으로 데이터를 변환할 스키마를 지정합니다. CreateApplication 작업은 단일 스트리밍 소스로부터 입력을 수신하도록 애플리케이션을 구성합니다. JSON 요소 또는 CSV 열을 SQL 열로 매핑하려면 SourceSchema `RecordColumns` 어레이에서 RecordColumn을 생성합니다. RecordColumn 객체에는 다음의 스키마가 있습니다.

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

RecordColumn 객체에 있는 필드는 다음의 값을 가집니다.

- `Mapping`: 입력 스트림에서 데이터의 위치를 식별하는 JSONPath 표현식입니다. CSV 형식의 소스 스트림에 대한 입력 스키마의 경우는 이 값이 존재하지 않습니다.
- `Name`: 애플리케이션 내 SQL 데이터 스트림에 있는 열의 명칭입니다.
- `SqlType`: 애플리케이션 내 SQL 데이터 스트림에 있는 데이터의 유형입니다.

JSON 입력 스키마 예

다음 예는 JSON 스키마에 대한 `InputSchema` 값의 형식을 보여 줍니다.

```
"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"
}
```

CSV 입력 스키마 예

다음 예는 CSV(쉼표로 분리된 값) 형식 스키마에 대한 `InputSchema` 값의 형식을 보여 줍니다.

```
"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"
}
```

JSON 데이터 유형을 SQL 데이터 유형으로 매핑하기

JSON 데이터 유형은 애플리케이션의 입력 스키마에 따라 상응하는 SQL 데이터 유형으로 변환됩니다. 지원되는 SQL 데이터 형식에 관한 자세한 설명은 데이터 유형을 참조하십시오. Amazon Kinesis Data Analytics는 다음 규칙에 따라 JSON 데이터 유형을 SQL 데이터 유형으로 변환합니다.

Null 리터럴

JSON 입력 스트림에 있는 null 리터럴(`"City":null`)은 대상의 데이터 유형에 상관없이 SQL null로 변환됩니다.

부울 리터럴

JSON 입력 스트림에 있는 부울 리터럴(`"Contacted":true`)은 다음과 같이 SQL 데이터로 변환됩니다.

- 숫자(DECIMAL, INT 등): `true`는 1로, `false`는 0으로 변환.
- 이진수(BINARY 또는 VARBINARY):
  - `true`: 결과는 가장 낮은 비트 세트를 가지며 남은 비트는 정리됩니다.
  - `false`: 결과의 모든 비트가 정리됩니다.

  VARBINARY로 변환하면 길이가 1바이트인 값이 도출됩니다.
- 부울: 상응하는 SQL 부울 값으로 변환됩니다.
- 문자(CHAR 또는 VARCHAR): 상응하는 문자열 값으로 변환됩니다(`true` 또는 `false`). 값은 필드의 길이에 맞게 잘립니다.
- 날짜/시간(DATE, TIME 또는 TIMESTAMP): 변환이 실패하면 강제 변환 오류가 오류 스트림에 기록됩니다.

숫자

JSON 입력 스트림에 있는 숫자 리터럴(`"CustomerId":67321`)은 다음과 같이 SQL 데이터로 변환됩니다.

- 숫자(DECIMAL, INT 등): 직접 변환됩니다. 변환된 값이 대상 데이터 유형의 크기 또는 정밀도를 초과하는 경우(즉, `123.4`를 INT로 변환하는 경우), 변환이 실패하고 강제 변환 오류가 오류 스트림에 기록됩니다.
- 이진수(BINARY 또는 VARBINARY): 변환이 실패하고 강제 변환 오류가 오류 스트림에 기록됩니다.
- BOOLEAN:
  - `0`: `false`로 변환됩니다.
  - 기타 모든 수: `true`로 변환됩니다.
- 문자(CHAR 또는 VARCHAR): 숫자의 문자열 표현으로 변환됩니다.
- 날짜/시간(DATE, TIME 또는 TIMESTAMP): 변환이 실패하면 강제 변환 오류가 오류 스트림에 기록됩니다.

String

JSON 입력 스트림에 있는 문자열 값(`"CustomerName":"John Doe"`)은 다음과 같이 SQL 데이터로 변환됩니다.

- 수치(DECIMAL, INT 등): Amazon Kinesis Data Analytics는 값을 대상 데이터 유형으로 변환합니다. 값을 변환할 수 없는 경우, 변환이 실패하고 강제 변환 오류가 오류 스트림에 기록됩니다.
- 이진수(BINARY 또는 VARBINARY): 소스 스트링이 유효한 이진 리터럴(즉, 짝수 f를 지닌 `X'3F67A23A'`)인 경우 값은 대상 데이터 유형으로 변환됩니다. 그렇지 않은 경우 변환이 실패하고 강제 변환 오류가 오류 스트림에 기록됩니다.
- 부울: 소스 스트링이 `"true"`인 경우, `true`로 변환됩니다. 이 비교는 대/소문자를 구분하지 않습니다. 그렇지 않은 경우, `false`로 변환됩니다.
- 문자(CHAR 또는 VARCHAR): 입력에서 문자열 값으로 변환됩니다. 값이 대상 데이터 유형보다 더 긴 경우, 잘려지고 오류 스트림에 오류가 기록되지 않습니다.
- 날짜/시간(DATE, TIME 또는 TIMESTAMP): 소스 스트링이 대상 값으로 변환할 수 있는 형식인 경우 값은 변환됩니다. 그렇지 않은 경우 변환이 실패하고 강제 변환 오류가 오류 스트림에 기록됩니다.

  유효한 날짜/시간 형식은 다음과 같습니다.

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

어레이 또는 객체

JSON 입력 스트림에 있는 어레이 또는 객체가 다음과 같이 SQL 데이터로 변환됩니다.

- 문자(CHAR 또는 VARCHAR): 어레이 또는 객체의 소스 텍스트로 변환됩니다. 배열 액세스 섹션을 참조하십시오.
- 기타 모든 데이터 유형: 변환이 실패하고 강제 변환 오류가 오류 스트림에 기록됩니다.

JSON 어레이의 예는 JSONPath로 작업하기를 참조하십시오.

관련 항목

- 애플리케이션 입력 구성
- 데이터 형식
- 스키마 편집기로 작업

- CreateApplication
- RecordColumn
- SourceSchema

스트리밍 데이터에 대해 스키마 검색 기능 사용

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 제한을 참조하십시오.

스트리밍 입력 상의 레코드가 어떻게 애플리케이션 내 스트림으로 매핑되는지 설명하는 입력 스키마를 제공하는 것은 복잡하고 오류가 발생하기 쉽습니다. DiscoverInputSchemaAPI(검색 API)를 사용하여 스키마를 유추할 수 있습니다. API는 스트리밍 소스 상의 랜덤 레코드 샘플을 사용하여 스키마를 유추할 수 있습니다(즉, 열 명칭, 데이터 유형, 수신 데이터에서의 데이터 요소 위치).

> **Note**
>
> 검색 API를 사용하여 Amazon S3에 저장된 파일에서 스키마를 생성하려면 정적 데이터에 대해 스키마 검색 기능 사용을 참조하십시오.

콘솔은 검색 API를 사용하여 지정된 스트리밍 소스에 대한 스키마를 생성합니다. 콘솔을 사용하면 열 추가 또는 삭제, 열 명칭 또는 데이터 유형 변경 등을 포함하여 스키마를 업데이트할 수 있습니다. 그러나 유효하지 않은 스키마를 생성하지 않도록 주의하여 변경합니다.

애플리케이션 내 스트림에 대한 스키마를 완료한 후에 문자열 및 날짜/시간 값을 조작하는 데 사용할 수 있는 함수가 있습니다. 결과로 얻은 애플리케이션 내 스트림에서 열 작업을 수행할 때 이들 함수를 애플리케이션 코드에서 사용할 수 있습니다. 자세한 설명은 예: DateTime 값 변환 섹션을 참조하십시오.

스키마 검색 시 열 명칭 지정

스키마 검색 시에 Amazon Kinesis Data Analytics는 다음의 경우를 제외하고 스트리밍 입력 소스로부터 원래 열 명칭을 최대한 많이 보존하려 시도합니다:

- 소스 스트림 열 명칭은 예약된 SQL 키워드(TIMESTAMP, USER, VALUES 또는 YEAR)입니다.
- 소스 스트림 열 명칭은 지원되지 않는 문자를 포함합니다. 글자, 숫자 및 밑줄 문자(_)만 지원됩니다.
- 소스 스트림 열 명칭은 숫자로 시작합니다.
- 소스 스트림 열 명칭은 100자보다 더 깁니다.

열의 명칭이 재지정되는 경우 재지정된 스키마 열 명칭은 COL_로 시작합니다. 일부 경우, 예를 들어 명칭 전체가 지원되지 않는 문자인 경우 원래 열 명칭 중 어느 것도 유지되지 않을 수 있습니다. 그와 같은 경우 열의 명칭이 COL_#로 지정되는데, 여기서 #는 열 순서에서 열의 자리를 나타내는 숫자입니다.

검색이 완료된 후 콘솔을 사용하여 열 추가 또는 삭제, 열 명칭, 데이터 유형 또는 데이터 크기 변경 등을 포함하여 스키마를 업데이트할 수 있습니다.

검색의 예-열 명칭 제안

소스 스트림 열 명칭	검색-열 명칭 제안
USER	COL_USER
USER@DOMAIN	COL_USERDOMAIN
@@	COL_0

스키마 검색 문제

Kinesis Data Analytics가 특정 스트리밍 소스에 대한 스키마를 유추할 수 없다면 어떻게 되는가?

Kinesis Data Analytics는 UTF-8 인코딩된 CSV 및 JSON과 같은 일반적인 형식에 대해 스키마를 유추합니다. Kinesis Data Analytics는 맞춤 열 및 행 구분자를 사용하여 모든 UTF-8 인코딩 레코드 (애플리케이션 로그 및 레코드와 같은 원시 텍스트 포함) 를 지원합니다. Kinesis Data Analytics가 스키마를 유추하지 못한다면 콘솔에서 스키마 편집기를 사용하거나 API를 사용하여 스키마를 수동으로 정의할 수 있습니다.

(데이터가 스키마 편집기를 사용하여 지정할 수 있는) 패턴에 따르지 않는 경우, 스키마를 단일 유형 열 VARCHAR(N)로 정의할 수 있습니다. 여기서 N은 레코드가 포함할 것으로 예상하는 최대 문자 수입니다. 거기에서 문자열 및 날짜 시간 조작을 통해 데이터가 애플리케이션 내 스트림 안에 포함된 후에 데이터를 구성할 수 있습니다. 예를 보려면 예: DateTime 값 변환 섹션을 참조하십시오.

정적 데이터에 대해 스키마 검색 기능 사용

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 제한을 참조하십시오.

스키마 검색 기능은 스트림의 데이터나 Amazon S3 버킷에 저장된 정적 파일의 데이터로부터 스키마를 생성할 수 있습니다. 참조 목적으로 Kinesis Data Analytics 애플리케이션용 스키마를 생성하는 경우나 실시간 스트리밍 데이터를 사용할 수 없는 경우를 가정해 보겠습니다. 참조 데이터 또는 스트리밍의 필요 형식으로 된 데이터 샘플을 포함하는 정적 파일에 대해 스키마 검색 기능을 사용할 수 있습니다. Kinesis Data Analytics는 Amazon S3 버킷에 저장된 JSON 또는 CSV 파일의 샘플 데이터에 대해 스키마 검색을 실행할 수 있습니다. 데이터 파일에 대해 스키마 검색을 사용할 때는 콘솔을 사용하거나 DiscoverInputSchema API를 지정된 `S3Configuration` 파라미터와 함께 사용합니다.

콘솔을 사용하여 스키마 검색 실행

콘솔을 사용하여 정적 파일에 대해 검색을 실행하려면 다음을 수행합니다.

1. 참조 데이터 객체를 S3 버킷에 추가합니다.
2. Kinesis Data Analytics 콘솔의 애플리케이션 기본 페이지에서 참조 데이터 연결을 선택합니다.
3. 참조 데이터가 포함된 Amazon S3 객체에 액세스하기 위한 버킷, 경로 및 IAM 역할 데이터를 제공합니다.
4. Discover schema(스키마 발견)를 선택합니다.

콘솔에서 참조 데이터를 추가하고 스키마를 검색하는 방법에 대한 자세한 설명은 예: 참조 데이터를 Kinesis Data Analytics 애플리케이션에 추가 섹션을 참조하십시오.

API를 사용하여 스키마 검색 실행

API를 사용하여 정적 파일에 대해 검색을 실행하려면 `S3Configuration` 구조를 포함하는 API에 다음 정보를 제공합니다.

- `BucketARN`: 파일을 포함하는 Amazon S3 버킷의 Amazon 리소스 이름(ARN). Amazon S3 버킷 ARN의 형식은 Amazon 리소스 이름(ARN) 및 Amazon Service Namespaces: Amazon Simple Storage Service(Amazon S3)를 참조하십시오.

- `RoleARN`: `AmazonS3ReadOnlyAccess` 정책이 적용되는 IAM 역할의 ARN. 역할에 정책을 추가하는 방법에 대한 자세한 설명은 역할 수정을 참조하십시오.
- `FileKey`: 객체의 파일 명칭입니다.

`DiscoverInputSchema` API를 사용하여 Amazon S3 객체에서 스키마를 생성하려면

1. 를 AWS CLI 설정했는지 확인합니다. 자세한 설명은 시작하기 섹션의 2단계: AWS Command Line Interface (AWS CLI) 설정 항목을 참조하십시오.
2. 다음 콘텐츠를 가진 `data.csv`이라는 파일을 생성합니다:

```
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. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔에 로그인하십시오.
4. Amazon S3 버킷을 생성하고 생성된 파일을 업로드합니다. 생성된 버킷의 ARN을 메모해 둡니다. Amazon S3 버킷 생성 및 파일 업로드 방법에 대한 자세한 설명은 Amazon Simple Storage 서비스 시작하기를 참조하십시오.
5. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다. `AmazonS3ReadOnlyAccess` 정책을 사용하여 역할을 생성합니다. 새 역할의 ARN을 기록합니다. 역할 생성에 대한 자세한 설명은 Amazon 서비스에 권한 위임 역할 생성을 참조하십시오. 역할에 정책을 추가하는 방법에 대한 자세한 설명은 역할 수정을 참조하십시오.
6. 에서 다음 `DiscoverInputSchema` 명령을 실행하여 Amazon S3 버킷 및 IAM 역할의 ARN을 AWS CLI대체합니다. ARNs

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

7. 응답은 다음과 유사해 보입니다.

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

Lambda 함수를 사용하여 데이터 사전 처리

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 제한을 참조하십시오.

스트림의 데이터에 형식 변환, 변환, 보강 또는 필터링이 필요한 경우 AWS Lambda 함수를 사용하여 데이터를 사전 처리할 수 있습니다. 애플리케이션 SQL 코드가 실행되기 전에 또는 애플리케이션이 데이터 스트림에서 스키마를 생성하기 전에 이 작업을 수행할 수 있습니다.

Lambda 함수를 사용하여 레코드를 사전 처리하면 다음과 같은 경우에 유용합니다:

- 다른 형식 (예: KPL 또는 GZIP)의 레코드를 Kinesis Data Analytics가 분석할 수 있는 형식으로 변환. Kinesis Data Analytics는 현재 JSON 또는 CSV 데이터 형식을 지원합니다.
- 집계, 변칙 검색 등과 같은 작업을 위해 액세스하기 쉬운 형식으로 데이터 확장. 예를 들어, 여러 데이터 값을 하나의 문자열에 함께 저장할 경우 데이터를 별도 열로 확장할 수 있습니다.
- 다른 Amazon 서비스 (예: 외삽, 오류 수정 등)의 데이터 강화.
- 레코드 필드에 복잡한 문자열 변환 적용
- 데이터 정리를 위해 데이터 필터링

Lambda 함수를 사용하여 레코드 사전 처리

Kinesis Data Analytics 애플리케이션을 생성할 때 소스에 연결 페이지에서 Lambda 사전 처리를 활성화합니다.

Kinesis Data Analytics 애플리케이션에서 Lambda 함수를 사용하여 레코드를 사전 처리하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesisanalytics Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션의 소스에 연결 페이지의 사전 처리 기록 AWS Lambda 항목 섹션에서 활성화를 선택합니다.
3. 이미 생성된 Lambda 함수를 사용하려면 Lambda 함수 드롭다운 목록에서 함수를 선택합니다.
4. Lambda 사전 처리 템플릿 중 하나에서 새 Lambda 함수를 생성하려면 드롭다운 목록에서 해당 템플릿을 선택합니다. 그런 다음 View <template name> in Lambda(Lambda에서 <템플릿 명칭> 보기)를 선택하여 함수를 편집합니다.
5. 새 Lambda 함수를 생성하려면 새로운 생성을 선택합니다. Lambda 함수 생성에 관한 정보는 AWS Lambda 개발자 가이드의 HelloWorld Lambda 함수 생성 및 콘솔 탐색을 참조하십시오.
6. 사용할 Lambda 함수의 버전을 선택합니다. 최신 버전을 사용하려면 [$LATEST]를 선택합니다.

레코드 사전 처리를 위해 Lambda 함수를 선택하거나 생성할 경우 애플리케이션 SQL 코드가 실행되거나 애플리케이션에서 레코드로부터 스키마를 생성하기 이전에 레코드가 사전 처리됩니다.

Lambda 사전 처리 권한

Lambda 사전 처리를 사용하려면 애플리케이션의 IAM 역할에 다음과 같은 권한 정책이 필요합니다.

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

Lambda 사전 처리 지표

Amazon CloudWatch를 사용하여 Lambda 간접 호출 수, 처리된 바이트 수, 성공 및 실패 등을 모니터링할 수 있습니다. Kinesis Data Analytics Lambda 사전 처리에서 발생되는 CloudWatch 지표에 대한 자세한 설명은 Amazon Kinesis Analytics 지표를 참조하십시오.

Kinesis 생산자 라이브러리와 AWS Lambda 함께 사용

Kinesis Producer Library(KPL)는 사용자 형식의 작은 레코드를 최대 1MB의 큰 레코드로 집계하여 Amazon Kinesis Data Streams 처리량을 효율적으로 활용할 수 있게 합니다. Java용 Kinesis Client Library(KCL)은 이러한 레코드의 분해를 지원합니다. 그러나를 스트림의 소비자 AWS Lambda 로 사용할 때는 특수 모듈을 사용하여 레코드를 분해해야 합니다.

필요한 프로젝트 코드와 지침을 얻으려면 GitHub에서 AWS Lambda를 위한 Kinesis Producer Library Deaggregation 모듈을 참조하십시오. 이 프로젝트의 구성 요소를 사용하여 Java, Node.js 및 Python AWS Lambda 에서 내에서 KPL 직렬화된 데이터를 처리할 수 있습니다. 복수의 언어 KCL 애플리케이션의 일부로 이러한 구성 요소를 사용할 수 있습니다.

데이터 사전 처리 이벤트 입력 데이터 모델/레코드 응답 모델

레코드를 사전 처리하려면 Lambda 함수가 필수 이벤트 입력 데이터 및 레코드 응답 모델을 준수해야 합니다.

이벤트 입력 데이터 모델

Kinesis Data Analytics는 Kinesis 데이터 스트림 또는 Firehose 전송 스트림에서 데이터를 연속적으로 읽습니다. 추출하는 각각의 배치 레코드의 경우 서비스가 Lambda 함수에 전달되는 방법을 관리합니다. 함수에서는 레코드 목록을 입력으로 수신합니다. 함수 내에서 목록을 반복하고 비즈니스 로직을 적용하여 사전 처리 요구 사항(예: 데이터 포맷 변환 또는 강화)을 충족합니다.

사전 처리 함수에 대한 입력 모델은 데이터가 Kinesis 데이터 스트림에서 수신되었는지 Firehose 전송 스트림에서 수신되었는지에 따라 조금씩 달라집니다.

소스가 Firehose 전송 스트림인 경우 이벤트 입력 데이터 모델은 다음과 같습니다.

Kinesis Data Firehose 요청 데이터 모델

필드	설명
`invocationId`	Lambda 간접 호출 ID(임의의 GUID)

필드	설명
applicationArn	Kinesis Data Analytics 애플리케이션 Amazon 리소스 이름(ARN)
streamArn	전송 스트림 ARN
레코드	

필드	설명
recordId	레코드 ID(임의 GUID)
kinesisFirehoseRecordMetadata	

필드	설명
approximateArrivalTimestamp	전송 스트림 레코드의 대략적인 도착 시간

data	Base64 인코딩된 소스 레코드 페이로드

다음 예는 Firehose 전송 스트림의 입력을 보여 줍니다.

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

소스가 Kinesis 데이터 스트림인 경우 이벤트 입력 데이터 모델은 다음과 같습니다:

Kinesis 스트림 요청 데이터 모델

필드	설명
`invocationId`	Lambda 간접 호출 ID(임의의 GUID)
`applicationArn`	Kinesis Data Analytics 애플리케이션 ARN
`streamArn`	전송 스트림 ARN
레코드	

필드	설명
`recordId`	Kinesis 레코드 시퀀스 번호를 기반으로 하는 레코드 ID
`kinesisStreamRecordMetadata`	

필드	설명
`sequenceNumber`	Kinesis 스트림 레코드의 시퀀스 번호
`partitionKey`	Kinesis 스트림 레코드의 파티션 키
`shardId`	`ShardId` Kinesis 스트림 레코드의
`approximateArrivalTimestamp`	전송 스트림 레코드의 대략적인 도착 시간

data	Base64 인코딩된 소스 레코드 페이로드

다음 예는 Kinesis 데이터 스트림의 입력을 보여 줍니다.

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

레코드 응답 모델

Lambda 함수에 전송되는 Lambda 사전 처리 함수(레코드 ID 포함)에서 반환된 모든 레코드가 반환되어야 합니다. 이러한 레코드는 다음 파라미터가 포함되어 있어야 하며 그렇지 않으면 Kinesis Data Analytics가 이를 거부하고 데이터 사전 처리 실패로 간주합니다. 레코드의 데이터 페이로드 부분을 사전 처리 요구 사항에 맞게 변환할 수 있습니다.

응답 데이터 모델

레코드

필드	설명
`recordId`	레코드 ID는 간접 호출 중에 Kinesis Data Analytics에서 Lambda로 전달됩니다. 변환된 레코드에는 동일한 레코드 ID가 포함되어야 합니다. 원래 레코드의 ID와 변환된 레코드의 ID 간 불일치는 데이터 사전 처리 실패로 간주됩니다.

필드	설명
`result`	레코드의 데이터 변환 상태입니다. 가능한 값은 다음과 같습니다. • `Ok`: 레코드가 성공적으로 변환되었습니다. Kinesis Data Analytics는 SQL 처리를 위해 레코드를 수집합니다. • `Dropped`: 처리 로직에 의해 레코드가 의도적으로 삭제되었습니다. Kinesis Data Analytics는 SQL 처리에서 레코드를 삭제합니다. `Dropped` 레코드에 대해 데이터 페이로드 필드는 선택 사항입니다. • `ProcessingFailed` : 레코드를 변환할 수 없습니다. Kinesis Data Analytics는 Lambda 함수가 이를 성공적으로 처리하지 못한 것으로 간주하고 오류 스트림에 오류를 기록합니다. 오류 스트림에 대한 자세한 설명은 오류 처리 섹션을 참조하십시오. `ProcessingFailed` 레코드에 대해 데이터 페이로드 필드는 선택 사항입니다.
`data`	base64 인코딩 후 변환된 데이터 페이로드입니다. 각 데이터 페이로드는 애플리케이션 수집 데이터 형식이 JSON인 경우 여러 JSON 문서를 포함할 수 있습니다. 또는 애플리케이션 수집 데이터 형식이 CSV인 경우 여러 CSV 행(각 행에서 지정된 행 구분 기호 사용)을 포함할 수 있습니다. Kinesis Data Analytics 서비스가 여러 JSON 문서 또는 동일한 데이터 페이로드 내 CSV 행에서 데이터를 구문 분석하고 처리합니다.

다음 예는 Lambda 함수의 출력을 보여 줍니다.

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

일반 데이터 사전 처리 기능

다음은 사전 처리에 실패할 수 있는 일반적인 이유입니다.

- Lambda 함수에 전송되는 배치 레코드 (레코드 ID 포함) 전체가 서비스에 반환 되지는 않습니다.
- 응답에 레코드 ID, 상태 또는 데이터 페이로드 필드가 없습니다. `Dropped` 또는 `ProcessingFailed` 레코드에 대해 데이터 페이로드 필드는 선택 사항입니다.
- Lambda 함수 제한 시간이 데이터를 사전 처리하는 데 충분하지 않습니다.
- Lambda 함수 응답이 AWS Lambda 서비스에서 부여한 응답 한도를 초과합니다.

데이터 사전 처리에 실패한 경우 Kinesis Data Analytics는 동일한 레코드 세트에 대해 호출을 성공할 때까지 계속 재시도합니다. 다음 CloudWatch 지표를 모니터링하면 실패의 원인을 알 수 있습니다.

- Kinesis Data Analytics 애플리케이션 `MillisBehindLatest`: 애플리케이션이 스트리밍 소스에서 읽어오는 시간이 얼마나 뒤처져 있는 지를 나타냅니다.
- Kinesis Data Analytics 애플리케이션 `InputPreprocessing` CloudWatch 지표: 통계 중에서 성공 횟수와 실패 횟수를 나타냅니다. 자세한 설명은 Amazon Kinesis Analytics 지표를 참조하십시오.
- AWS Lambda CloudWatch 지표 및 로그 함수.

사전 처리용 Lambda 함수 생성

Amazon Kinesis Data Analytics 애플리케이션은 애플리케이션으로 수집되는 레코드를 사전 처리하는 데 Lambda 함수를 사용할 수 있습니다. Kinesis Data Analytics는 콘솔에서 데이터 사전 처리를 위한 출발점으로 사용할 수 있는 다음과 같은 템플릿을 제공합니다.

주제

- Node.js에서 Lambda 사전 처리 함수 생성
- Python에서 사전 처리 함수 생성
- Java에서 사전 처리 함수 생성
- .NET에서 Lambda 사전 처리 함수 생성

Node.js에서 Lambda 사전 처리 함수 생성

Node.js에서 Lambda 사전 처리 함수 생성에 사용할 아래의 템플릿은 Kinesis Data Analytics 콘솔에서 사용할 수 있습니다.

Lambda 창사진	언어 및 버전	설명
일반 Kinesis Data Analytics 입력 처리	Node.js 6.10	Kinesis Data Analytics는 JSON 또는 CSV 레코드를 입력으로 수신한 다음 사전 처리 상태로 반환하는 사전 처리 프로세서를 기록합니다. 이 프로세서를 맞춤 변환 로직에 대한 시작점으로 사용합니다.
압축된 입력 처리	Node.js 6.10	Kinesis Data Analytics는 압축된(GZIP 또는 Deflate 압축) JSON 또는 CSV 레코드를 입력으로 수신하고 압축 해제된 레코드를 처리 상태로 반환하는 사전 처리 프로세서를 기록합니다.

Python에서 사전 처리 함수 생성

Python에서 Lambda 사전 처리 함수를 생성하기 위한 아래의 템플릿은 콘솔에서 사용할 수 있습니다:

Lambda 청사진	언어 및 버전	설명
일반 Kinesis Analytics 입력 처리	Python 2.7	Kinesis Data Analytics는 JSON 또는 CSV 레코드를 입력으로 수신한 다음 사전 처리 상태로 반환하는 사전 처리 프로세서를 기록합니다. 이 프로세서를 맞춤 변환 로직에 대한 시작점으로 사용합니다.
KPL 입력 처리	Python 2.7	Kinesis Data Analytics는 JSON의 Kinesis Producer Library (KPL) 집합 또는 CSV 레코드를 입력으로 수신한 다음 처리 상태와 함께 분리된 기록 반환을 기록합니다.

Java에서 사전 처리 함수 생성

Java에서 레코드 사전 처리용 Lambda 함수를 생성하려면 Java 이벤트 클래스를 사용하십시오.

다음 코드는 Java를 사용하여 레코드를 사전 처리하는 Lambda 함수 샘플입니다:

```
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;
    }

}
```

.NET에서 Lambda 사전 처리 함수 생성

.NET에서 레코드 사전 처리용 Lambda 함수를 생성하려면 .NET 이벤트 클래스를 사용합니다.

다음 코드는 C#을 사용하여 레코드를 사전 처리하는 Lambda 함수 예입니다:

```
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;
        }
    }
```

.NET에서 사전 처리 및 목적지용 Lambda 함수 생성에 대한 자세한 설명은 Amazon.Lambda.KinesisAnalyticsEvents를 참조하십시오.

입력 스트림 병렬화를 통한 처리량 증대

> **Note**
>
> 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우, Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. 자세한 설명은 한도를 참조하십시오.

Amazon Kinesis Data Analytics 애플리케이션은 복수의 애플리케이션 내 입력 스트림을 지원하여 애플리케이션의 용량을 단일 애플리케이션 내 입력 스트림의 처리량 이상으로 확장할 수 있습니다. 애플리케이션 내 입력 스트림에 대한 자세한 설명은 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 섹션을 참조하십시오.

대부분의 경우 Amazon Kinesis Data Analytics는 애플리케이션에 피드되는 Kinesis 스트림 또는 Firehose 소스 스트림의 용량을 처리하도록 애플리케이션의 규모를 조정합니다. 그러나 소스 스트림의 처리량이 단일 애플리케이션 내 입력 스트림의 처리량을 초과하는 경우 애플리케이션이 사용하는 애플리케이션 내 입력 스트림의 수를 명시적으로 늘릴 수 있습니다. `InputParallelism` 파라미터를 사용하여 그렇게 합니다.

`InputParallelism` 파라미터가 1보다 클 경우 Amazon Kinesis Data Analytics는 소스 스트림의 파티션을 애플리케이션 내 스트림 간에 균등하게 분할합니다. 예를 들어, 소스 스트림의 샤드 수가 50개이고 `InputParallelism`을 2로 설정한 경우, 각각의 애플리케이션 내 입력 스트림은 25개의 소스 스트림 샤드로부터 입력을 수신합니다.

애플리케이션 내 수를 늘릴 경우 애플리케이션이 각 스트림에서 데이터에 명시적으로 액세스해야 합니다. 코드에서 복수의 애플리케이션 내 스트림에 액세스하는 것에 대한 정보는 Amazon Kinesis Data Analytics 애플리케이션에서 별도의 애플리케이션 내 스트림에 액세스하는 방법 을 참조하십시오.

Kinesis 데이터 스트림과 Firehose 샤드가 둘 다 동일한 방식으로 애플리케이션 내 스트림 간에 분할되지만, 애플리케이션에 표시되는 방식은 다릅니다.

- Kinesis 데이터 스트림의 레코드에는 레코드의 소스 샤드를 식별하는 데 사용 가능한 `shard_id` 필드가 포함됩니다.
- Firehose 전송 스트림의 레코드는 레코드의 소스 샤드 또는 파티션을 식별하는 필드를 포함하지 않습니다. Firehose가 이 정보를 애플리케이션에서 추상화하기 때문입니다.

애플리케이션 내 입력 스트림 수를 증대해야 할지 여부에 대한 평가

대부분의 경우 단일 애플리케이션 내 입력 스트림은 입력 스트림의 복잡성 및 데이터 크기에 따라 단일 소스 스트림의 처리량을 처리할 수 있습니다. Amazon CloudWatch에서 `InputBytes` 및 `MillisBehindLatest`의 지표를 모니터링하여 애플리케이션 내 입력 스트림의 증가 여부를 판단할 수 있습니다.

`InputBytes` 지표가 100MB/초보다 큰 경우 (또는 커질 것으로 예상하는 경우) `MillisBehindLatest`가 증가하고 애플리케이션 문제의 영향이 증가할 수 있습니다. 이를 해결하기 위해 애플리케이션에 대한 다음의 언어 선택을 권장합니다.

- 100MB/초 이상으로 애플리케이션 규모 조정이 필요한 경우 복수의 스트림과 Kinesis Data Analytics for SQL 애플리케이션를 사용하십시오.
- 단일 스트림과 애플리케이션을 사용하려는 경우 Java 애플리케이션용 Kinesis Data Analytics를 사용하십시오.

`MillisBehindLatest` 지표가 다음 특성 중 하나라도 가지고 있을 경우 애플리케이션의 `InputParallelism` 설정을 늘려야 합니다.

- `MillisBehindLatest` 지표가 점진적으로 증가하여 애플리케이션이 스트림에서의 최신 데이터보다 뒤쳐지고 있음을 나타냅니다.
- `MillisBehindLatest` 지표가 지속적으로 1000(1초)을 초과합니다.

다음에 해당하는 경우 애플리케이션의 `InputParallelism` 설정을 늘릴 필요가 없습니다.

- `MillisBehindLatest` 지표가 점진적으로 감소하여 애플리케이션이 스트림에서의 최신 데이터를 따라잡고 있음을 나타냅니다.
- `MillisBehindLatest` 지표가 1000(1초) 아래입니다.

CloudWatch에 대한 자세한 정보는 Amazon CloudWatch 사용자 가이드를 참조하십시오.

복수의 애플리케이션 내 입력 스트림 구현

애플리케이션을 생성할 때 CreateApplication을 사용하여 애플리케이션 내 입력 스트림의 수를 설정할 수 있습니다. 애플리케이션을 생성한 후에 UpdateApplication을 사용하여 이 수를 설정합니다.

> **Note**
>
> Amazon Kinesis Data Analytics API 또는 AWS CLI를 사용해서만 `InputParallelism` 설정값을 설정할 수 있습니다. 를 사용하여이 설정을 설정할 수 없습니다 AWS Management Console. 설정에 대한 자세한 내용은 섹션을 AWS CLI참조하세요2단계: AWS Command Line Interface (AWS CLI) 설정.

새 애플리케이션의 입력 스트림 수 설정

다음 예는 `CreateApplication` API 작업을 사용하여 새 애플리케이션의 입력 스트림 수를 2로 설정하는 방법을 보여줍니다.

`CreateApplication`에 대한 자세한 정보는 CreateApplication을 참조하세요.

```
{
   "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": [ ... ],
 }]
}
```

기존 애플리케이션의 입력 스트림 수 설정

다음 예는 `UpdateApplication` API 작업을 사용하여 기존 애플리케이션의 입력 스트림 수를 2로 설정하는 방법을 보여 줍니다.

`Update_Application`에 대한 자세한 정보는 UpdateApplication을 참조하세요.

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

Amazon Kinesis Data Analytics 애플리케이션에서 별도의 애플리케이션 내 스트림에 액세스하는 방법

애플리케이션에서 복수의 애플리케이션 내 입력 스트림을 사용하려면 각기 다른 스트림에서 명시적으로 선택해야 합니다. 다음의 코드 예는 시작하기 자습서에서 생성한 애플리케이션에서 복수의 입력 스트림을 쿼리하는 방법을 보여 줍니다.

다음 예에서는 각 소스 스트림은 수를 사용하여 우선 집계된 후 `in_application_stream001`라는 단일 애플리케이션 내 스트림으로 결합됩니다. 소스 스트림을 미리 집계하면 결합된 애플리케이션 내 스트림이 과부하 없이 복수의 스트림에서 발생하는 트래픽을 처리하는 데 도움이 됩니다.

> **Note**
>
> 이 예를 실행하여 두 애플리케이션 내 입력 스트림으로부터 결과를 얻으려면 소스 스트림에 있는 샤드의 수와 애플리케이션의 `InputParallelism` 파라미터를 둘 다 업데이트합니다.

```
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;
```

위의 코드 예는 다음과 비슷한 결과를 `in_application_stream001`에 출력합니다:

Filter by column name

ROWTIME	TICKER	TICKER_COUNT
2017-05-17 22:05:00.0	QAZ	15
2017-05-17 22:06:00.0	SAC	16
2017-05-17 22:06:00.0	PLM	10
2017-05-17 22:06:00.0	AMZN	15

추가 고려 사항

복수의 입력 스트림을 사용하는 경우 다음에 유의해야 합니다.

- 애플리케이션 내 입력 스트림의 최대 수는 64입니다.
- 애플리케이션 내 입력 스트림은 애플리케이션 입력 스트림의 샤드 간에 균등하게 분배됩니다.
- 애플리케이션 내 스트림 추가에 따른 성능상 이득은 선형적으로 증가하지 않습니다. 즉, 애플리케이션 내 스트림을 두 배로 늘린다고 처리량이 두 배가 되지는 않습니다. 행 크기가 일반적인 경우, 각 애플리케이션 내 스트림은 초당 약 5,000~15,000개의 행을 처리할 수 있습니다. 애플리케이션 내 스트림 수를 10으로 늘리면 초당 20,000~30,000 정도의 처리량을 확보할 수 있습니다. 처리 속도는 입력 스트림 필드의 수, 데이터 유형 및 데이터 크기에 좌우됩니다.
- 일부 집계 함수(예: AVG)는 여러 샤드에 분할된 입력 스트림에 적용할 경우 예상치 못한 결과를 산출할 수 있습니다. 샤드를 집계 스트림에 결합하기 전에 개별 샤드에 대한 집계 작업을 실행해야 하기 때문에 결과를 레코드가 더 많이 포함된 것에 대해 가감해야 할 수 있습니다.
- 입력 스트림 수를 늘린 후에도 계속해서 성능 저하가 일어나면 (`MillisBehindLatest`의 지표가 높게) Kinesis 처리 단위(KPU)의 한도에 이른 것일 수도 있습니다. 자세한 설명은 처리량 증가를 위해 애플리케이션 용량을 자동으로 확장 또는 축소 섹션을 참조하십시오.

애플리케이션 코드

애플리케이션 코드는 입력을 처리하고 출력을 생성하는 일련의 SQL 문입니다. 이 SQL 문은 애플리케이션 내 스트림과 참조 표로 운영됩니다. 자세한 설명은 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 섹션을 참조하십시오.

Kinesis Data Analytics에서 지원되는 SQL 언어 요소에 대한 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조를 참조하십시오.

관계형 데이터베이스에서 INSERT 문을 통해 레코드를 추가하고 SELECT 문을 통해 데이터를 쿼리하는 표 작업을 수행합니다. Amazon Kinesis Data Analytics에서는 스트림을 사용합니다. SQL 문을 작성하여 이들 스트림을 쿼리할 수 있습니다. 하나의 애플리케이션 내 스트림을 쿼리한 결과는 항상 또 다른 애플리케이션 내 스트림으로 전송됩니다. 복잡한 분석을 수행할 때는 여러 개의 애플리케이션 내 스트림을 생성하여 중간 분석 결과를 유지할 수 있습니다. 그리고 끝으로 최종 분석 결과를(하나 이상의 애플리케이션 내 스트림으로부터) 외부 대상에 유지하도록 애플리케이션 출력을 구성합니다. 요약하자면, 애플리케이션 코드 작성의 일반적인 패턴은 다음과 같습니다.

- INSERT 문 맥락에서는 항상 SELECT 문이 사용됩니다. 즉, 행을 선택할 때 결과를 다른 애플리케이션 내 스트림에 삽입합니다.

- 펌프 맥락에서는 항상 INSERT 문이 사용됩니다. 즉, 펌프를 사용하여 애플리케이션 내 스트림에 작성합니다.

다음 예 애플리케이션 코드는 하나의 애플리케이션 내 스트림(SOURCE_SQL_STREAM_001)으로부터 레코드를 읽고 이를 또 다른 애플리케이션 내 스트림(DESTINATION_SQL_STREAM)에 기록합니다. 다음과 같이 펌프를 사용하여 레코드를 애플리케이션 내 스트림에 삽입할 수 있습니다.

```
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**
>
> 스트림 명칭 및 열 명칭에 대해 지정하는 식별자는 표준 SQL 규칙에 따릅니다. 예를 들어 식별자 앞뒤로 따옴표를 사용하는 경우 식별자는 대소문자를 구분합니다. 따옴표를 사용하지 않는 경우에는 기본적으로 식별자는 대문자입니다. 식별자에 대한 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 식별자를 참조하십시오.

애플리케이션 코드는 복수의 SQL 문으로 구성될 수 있습니다. 예시:

- 한 SQL 문의 결과가 다음 SQL 문으로 전달되는 경우 SQL 쿼리를 순차적으로 작성할 수 있습니다.
- 서로 독립적으로 실행하는 SQL 문도 작성할 수 있습니다. 예를 들어, 동일한 애플리케이션 내 스트림을 쿼리하지만 다른 애플리케이션 내 스트림으로 출력을 전송하는 두 개의 SQL 문을 작성할 수 있습니다. 그런 다음 새로 생성된 애플리케이션 내 스트림을 독립적으로 쿼리할 수 있습니다.

애플리케이션 내 스트림을 생성하여 중간 결과를 저장할 수 있습니다. 펌프를 사용하여 데이터를 애플리케이션 내 스트림에 삽입합니다. 자세한 설명은 애플리케이션 내 스트림과 펌프 섹션을 참조하십시오.

애플리케이션 내 참조 표를 추가하는 경우 SQL을 작성하여 애플리케이션 내 스트림 및 참조 표에 있는 데이터를 조인할 수 있습니다. 자세한 설명은 예: 참조 데이터를 Kinesis Data Analytics 애플리케이션에 추가 섹션을 참조하십시오.

애플리케이션의 출력 구성에 따라 Amazon Kinesis Data Analytics는 특정 애플리케이션 내 스트림으로부터 외부 목적지로 데이터를 기록합니다. 애플리케이션 코드가 출력 구성에 지정된 애플리케이션 내 스트림으로 기록하는지 확인합니다.

자세한 정보는 다음 주제를 참조하십시오:

- 스트리밍 SQL 개념
- Amazon Kinesis Data Analytics SQL 참조

애플리케이션 출력 구성

애플리케이션 코드에서 SQL 문의 출력을 하나 이상의 애플리케이션 스트림에 작성합니다. 필요에 따라 애플리케이션에 출력 구성을 추가할 수 있습니다.는 인애플리케이션 스트림에 기록된 모든 내용을 Amazon Kinesis 데이터 스트림, Firehose 전송 스트림 또는 AWS Lambda 함수와 같은 외부 대상으로 유지할 수 있습니다.

애플리케이션 출력을 유지하는 데 사용할 수 있는 외부 대상의 수에는 제한이 있습니다. 자세한 설명은 한도 섹션을 참조하십시오.

> **Note**
>
> 오류를 조사할 수 있도록 하나의 외부 대상을 사용하여 애플리케이션 내 오류 스트림 데이터를 유지하는 것이 좋습니다.

이들 출력 구성 각각에 대해 다음을 제공합니다:

- 애플리케이션 내 스트림 명칭 – 외부 목적지에 유지하고자 하는 스트림.

  Kinesis Data Analytics는 출력 구성에 지정된 애플리케이션 내 스트림을 탐색합니다. (이 스트림 명칭은 대/소문자를 구분하며 정확히 일치해야 합니다.) 애플리케이션 코드가 이 애플리케이션 내 스트림을 생성하는지 확인하십시오.
- 외부 대상 – Kinesis 데이터 스트림, Firehose 전송 스트림 또는 Lambda 함수에 데이터를 유지할 수 있습니다. 스트림 또는 함수의 Amazon 리소스 이름(ARN)을 제공합니다. 또한 Kinesis Data

Analytics가 귀하를 대신하여 스트림 또는 기능을 가질 수 있도록 수 IAM 역할을 줄 수 있습니다 외부 목적지에 작성할 때 사용할 레코드 형식(JSON, CSV)을 Kinesis Data Analytics에 기술합니다.

Kinesis Data Analytics가 스트리밍 또는 Lambda 목적지에 작성할 수 없을 경우에는 계속하여 무한정 시도합니다. 이로 인해 백 프레셔(back pressure)가 발생하여 애플리케이션이 뒤쳐지게 됩니다. 이 문제가 해결되지 않을 경우 결국 애플리케이션이 신규 데이터 처리를 중단합니다. Amazon Kinesis Data Analytics 지표를 모니터링하고 장애에 대한 경보를 설정할 수 있습니다. 지표 및 경보에 대한 자세한 설명은 Using Amazon CloudWatch 지표 사용과 Creating Amazon CloudWatch 경보 생성을 참조하십시오.

AWS Management Console을(를) 사용하여 애플리케이션 출력을 구성할 수 있습니다. 콘솔은 API 호출을 통해 구성을 저장합니다.

를 사용하여 출력 생성 AWS CLI

이 섹션에서는 `CreateApplication` 또는 `AddApplicationOutput` 작업에 대한 요청 본문의 `Outputs` 섹션을 생성하는 방법을 설명합니다.

Kinesis 스트림 출력 생성

다음 JSON 조각은 Amazon Kinesis 데이터 스트림 목적지를 생성하기 위한 `CreateApplication` 요청 본문의 `Outputs` 섹션을 보여줍니다.

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

]
```

Firehose 전송 스트림 출력 생성

다음 JSON 조각은 Amazon Data Firehose 전송 스트림 대상을 생성하기 위한 `CreateApplication` 요청 본문의 `Outputs` 섹션을 보여줍니다.

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

Lambda 함수 출력 생성

다음 JSON 조각은 AWS Lambda 함수 대상을 생성하기 위한 `CreateApplication` 요청 본문의 `Outputs` 섹션을 보여줍니다.

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

Lambda 함수를 출력으로 사용

AWS Lambda 를 대상으로 사용하면 SQL 결과를 최종 대상으로 보내기 전에 사후 처리를 보다 쉽게 수행할 수 있습니다. 일반적인 사후 처리 작업에는 다음이 포함됩니다.

- 단일 레코드에 여러 행 집계

- 현재 결과와 과거 결과를 결합하여 늦게 도착하는 데이터 처리
- 정보의 유형에 따라 서로 다른 대상으로 전송
- 레코드 형식 변환(예: Protobuf로 변환)
- 문자열 조작 또는 변환
- 분석 처리 후 데이터 강화
- 지리 공간 사용 사례에 대한 맞춤 처리
- 데이터 암호화

Lambda 함수는 다음을 포함하여 다양한 AWS 서비스 및 기타 대상에 분석 정보를 제공할 수 있습니다.

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

Lambda 애플리케이션 생성에 대한 자세한 설명은 AWS Lambda로 시작하기 섹션을 참조하십시오.

주제

- 출력 권한으로서 Lambda
- 출력 지표로서의 Lambda
- 출력 이벤트 입력 데이터 모델 및 레코드 응답 모델로서 Lambda
- Lambda 출력 호출 빈도
- 출력으로 사용할 Lambda 함수 추가
- 일반적인 Lambda 출력 실패
- 애플리케이션 목적지용 Lambda 함수 생성

출력 권한으로서 Lambda

Lambda를 출력으로 사용하려면 애플리케이션의 Lambda 출력 IAM 역할에 다음과 같은 권한 정책이 필요합니다:

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

출력 지표로서의 Lambda

Amazon CloudWatch를 사용하여 전송된 바이트 수, 성공 및 실패 등을 모니터링 할 수 있습니다. Kinesis Data Analytics에서 Lambda를 출력으로 사용하여 방출되는 CloudWatch 지표에 대한 자세한 설명은 Amazon Kinesis Analytics 지표를 참조하십시오.

출력 이벤트 입력 데이터 모델 및 레코드 응답 모델로서 Lambda

Kinesis Data Analytics 출력 레코드를 전송하려면 Lambda 함수가 필수 이벤트 입력 데이터 및 레코드 응답 모델을 준수해야 합니다.

이벤트 입력 데이터 모델

Kinesis Data Analytics는 다음 요청 모델을 사용하여 애플리케이션의 출력 레코드를 출력 함수인 Lambda로 계속 전송합니다. 함수 내에서 목록을 반복하고 비즈니스 로직을 적용하여 출력 요구 사항(예: 최종 대상으로 전송하기 전에 데이터 변환)을 충족합니다.

필드	설명
`invocationId`	Lambda 간접 호출 ID(임의의 GUID)
`applicationArn`	Kinesis Data Analytics 애플리케이션 Amazon 리소스 이름(ARN).
레코드	

필드	설명
<table><tr><th>필드</th><th>설명</th></tr><tr><td>recordId</td><td>레코드 ID(임의 GUID)</td></tr><tr><td>lambdaDeliveryRecordMetadata</td><td><table><tr><th>필드</th><th>설명</th></tr><tr><td>retryHint</td><td>전송 재시도 횟수</td></tr></table></td></tr><tr><td>data</td><td>Base64 인코딩된 출력 레코드 페이로드</td></tr></table>	

> **Note**
>
> `retryHint`는 전송이 실패할 때마다 증가하는 값입니다. 이 값은 영구적으로 유지되지 않으며 애플리케이션이 중단되면 재설정됩니다.

레코드 응답 모델

출력 함수인 Lambda로 전송된 각 레코드(레코드 ID 포함)는 `Ok` 또는 `DeliveryFailed`로 승인되어야 하며 다음 파라미터를 포함해야 합니다. 그렇지 않으면 Kinesis Data Analytics가 이를 전송 실패로 처리합니다.

레코드

필드	설명
`recordId`	레코드 ID는 간접 호출 중에 Kinesis Data Analytics에서 Lambda로 전달됩니다. 원래 레코드의 ID와 승인된 레코드의 ID 간 불일치는 전송 실패로 처리됩니다.
`result`	레코드의 전송 상태입니다. 유효한 값은 다음과 같습니다.

필드	설명
	• `Ok`: 레코드가 성공적으로 변환되고 최종 목적지로 전송되었습니다. Kinesis Data Analytics는 SQL 처리를 위해 레코드를 수집합니다. • `DeliveryFailed` : Lambda가 출력 함수로 레코드를 최종 목적지로 성공적으로 전달하지 못했습니다. Kinesis Data Analytics는 전송 실패 레코드를 Lambda에 출력 함수로 전송하는 작업을 계속 재시도합니다.

Lambda 출력 호출 빈도

Kinesis Data Analytics 애플리케이션은 출력 레코드를 버퍼링하고 AWS Lambda 목적지 함수를 빈번하게 간접 호출합니다.

- 레코드가 데이터 분석 애플리케이션 내의 대상 인애플리케이션 스트림으로 텀블링 윈도우로 내보내지면 텀블링 윈도우 트리거당 AWS Lambda 대상 함수가 호출됩니다. 예를 들어 60초 텀블링 윈도우가 레코드를 목적지 애플리케이션 내 스트림으로 방출하는 데 사용되면 Lambda 함수가 60초에 한 번씩 간접 호출됩니다.
- 레코드가 애플리케이션에서 연속 쿼리 또는 슬라이딩 윈도우로서 대상 애플리케이션 내 스트림으로 방출된 경우 약 1초에 한 번씩 Lambda 목적지 함수가 간접 호출됩니다.

> **Note**
>
> Lambda 함수 당 간접 호출 요청 페이로드 크기 한도가 적용됩니다. 이러한 제한을 초과하면 출력 레코드가 분할되어 여러 Lambda 함수 호출로 전송됩니다.

출력으로 사용할 Lambda 함수 추가

다음 절차는 Lambda 함수를 Kinesis Data Analytics 애플리케이션의 출력으로 추가하는 방법을 보여줍니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesisanalytics Managed Service for Apache Flink 콘솔을 엽니다.
2. 목록에서 애플리케이션을 선택한 다음 [Application details]를 선택합니다.

3. [Destination] 섹션에서 [Connect new destination]을 선택합니다.
4. [Destination] 항목에서 [AWS Lambda function]을 선택합니다.
5. 레코드를 AWS Lambda에 전송 섹션에서 기존 Lambda 함수 및 버전을 선택하거나, 또는 새로 만들기를 선택합니다.
6. Lambda 함수를 새로 생성하는 경우에는 다음을 수행합니다:
   a. 제공된 템플릿 중 하나를 선택합니다. 자세한 설명은 애플리케이션 목적지용 Lambda 함수 생성 섹션을 참조하십시오.
   b. 새 브라우저 탭에서 함수 생성 페이지가 열립니다. 명칭 상자에서 함수에 의미 있는 명칭(예: **`myLambdaFunction`**)을 지정합니다.
   c. 애플리케이션의 사후 처리 기능으로 템플릿을 업데이트합니다. Lambda 함수 생성에 대한 자세한 설명은 AWS Lambda 개발자 가이드의 시작하기를 참조하십시오.
   d. Kinesis Data Analytics 콘솔의 Lambda 함수 목록에서, 방금 생성한 Lambda 함수를 선택합니다. Lambda 함수 버전의 경우 $LATEST를 선택합니다.
7. [In-application stream] 섹션에서 [Choose an existing in-application stream]을 선택합니다. [In-application stream name]에서 애플리케이션의 출력 스트림을 선택합니다. 선택한 출력 스트림의 결과는 Lambda 출력 함수로 전송됩니다.
8. 양식의 나머지 부분은 기본값으로 두고 [Save and continue]를 선택합니다.

이제 애플리케이션에서 애플리케이션 내 스트림의 레코드를 Lambda 함수로 전송합니다. Amazon CloudWatch 콘솔에서 기본 템플릿의 결과를 볼 수 있습니다. Lambda 함수에 전송되는 레코드 수를 보려면 `AWS/KinesisAnalytics/LambdaDelivery.OkRecords` 지표를 모니터링하십시오.

일반적인 Lambda 출력 실패

다음은 Lambda 함수로 전송할 수 없는 일반적인 이유입니다.

- Lambda 함수에 전송되는 배치 내의 일부 레코드(레코드 ID 포함)가 Kinesis Data Analytics 서비스에 반환 되지는 않습니다.
- 응답에 레코드 ID 또는 상태 필드가 없습니다.
- Lambda 함수 제한 시간이 Lambda 함수 내의 비즈니스 로직을 수행하는 데 충분하지 않습니다.
- Lambda 함수 내의 비즈니스 로직은 모든 오류를 잡아내지 못하므로 처리되지 않은 예외로 인해 시간 초과 및 백 프레셔(back pressure)가 발생합니다. 이를 "독약" 메시지라고 하기도 합니다.

데이터 전송에 실패한 경우 Kinesis Data Analytics는 동일한 레코드 세트에 대해 Lambda 간접 호출을 성공할 때까지 계속 재시도합니다. 다음 CloudWatch 지표를 모니터링하여 실패의 원인을 알 수 있습니다:

- CloudWatch 출력 지표로서의 Kinesis Data Analytics 애플리케이션 Lambda: 다른 통계 중에서 성공 횟수와 실패 횟수를 나타냅니다. 자세한 설명은 Amazon Kinesis Analytics 지표를 참조하십시오.
- AWS Lambda CloudWatch 지표 및 로그 함수.

애플리케이션 목적지용 Lambda 함수 생성

Kinesis Data Analytics 애플리케이션은 AWS Lambda 함수를 출력으로 사용할 수 있습니다. Kinesis Data Analytics는 애플리케이션의 대상으로 사용할 Lambda 함수를 생성하기 위한 템플릿을 제공합니다. 이러한 템플릿은 애플리케이션에서 사후 처리 출력을 위한 시작점으로 사용합니다.

주제

- Node.js에서 Lambda 함수 목적지 생성
- Python에서 Lambda 함수 목적지 생성
- Java에서 Lambda 함수 목적지 생성
- .NET에서 Lambda 함수 목적지 생성

Node.js에서 Lambda 함수 목적지 생성

Node.js에서 Lambda 함수를 생성하는 다음 템플릿은 콘솔에서 사용할 수 있습니다:

출력 청사진으로서의 Lambda	언어 및 버전	설명
`kinesis-analytics-output`	Node.js 12.x	Kinesis Data Analytics 애플리케이션의 출력 레코드를 맞춤 목적지로 전송합니다.

Python에서 Lambda 함수 목적지 생성

Python에서 목적지 Lambda 함수를 생성할 때 다음 템플릿은 콘솔에서 사용할 수 있습니다:

출력 청사진으로서의 Lambda	언어 및 버전	설명
`kinesis-analytics-output-sns`	Python 2.7	Kinesis Data Analytics 애플리케이션의 출력 기록을 Amazon SNS로 전송하십시오.
`kinesis-analytics-output-ddb`	Python 2.7	Kinesis Data Analytics 애플리케이션의 출력 기록을 Amazon DynamoDB로 전송하십시오.

Java에서 Lambda 함수 목적지 생성

Java에서 목적지 Lambda 함수를 생성하려면 Java 이벤트 클래스를 사용합니다.

다음 코드는 Java를 사용한 목적지 Lambda 함수 예입니다:

```
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;
    }

}
```

.NET에서 Lambda 함수 목적지 생성

.NET에서 목적지 Lambda 함수를 생성하려면 .NET 이벤트 클래스를 사용합니다.

다음 코드는 C#을 사용한 목적지 Lambda 함수 예입니다:

```
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;
        }
```

```
    }
```

.NET에서 사전 처리 및 목적지 Lambda 함수 생성에 대한 자세한 설명은 `Amazon.Lambda.KinesisAnalyticsEvents`을 참조하십시오.

애플리케이션 출력을 외부 대상에 유지하기 위한 전송 모델

Amazon Kinesis Data Analytics에서는 구성할 애플리케이션 출력에 대해 "최소 1회" 전송 모델을 적용합니다. 애플리케이션을 실행 중일 때 Kinesis Data Analytics에서는 내부 체크포인트를 사용합니다. 이러한 체크포인트는 출력 레코드가 데이터 손실 없이 대상으로 전송된 시점입니다. Kinesis Analytics에서는 필요에 따라 체크포인트를 사용하여 애플리케이션 출력이 구성된 대상으로 최소한 한번은 전달되도록 합니다.

정상적인 상황에서 애플리케이션은 들어오는 데이터를 지속적으로 처리합니다. Kinesis Data Analytics는 Kinesis 데이터 스트림 또는 Firehose 전송 스트림 같은 구성된 대상에 출력을 작성합니다. 하지만 애플리케이션은 경우에 따라 중단될 수 있습니다. 예:

- 애플리케이션을 중단하고 나중에 재시작하도록 선택하는 경우.
- Kinesis Data Analytics이 애플리케이션 출력을 구성된 목적지에 작성할 필요가 있는 IAM 역할을 삭제하는 경우. IAM 역할이 없다면 Kinesis Data Analytics는 귀하를 대신하여 외부 대상에 작성할 수 있는 권한이 없습니다.
- 네트워크 중단 또는 기타 내부 서비스 결함으로 인해 애플리케이션의 실행이 일시적으로 중지됩니다.

애플리케이션이 다시 시작될 경우 Kinesis Data Analytics는 실패가 발생할 때 또는 그 이전 시점부터 출력을 계속 처리 및 기록합니다. 따라서 애플리케이션 출력이 구성된 대상으로 확실하게 전송됩니다.

동일한 애플리케이션 내 스트림에서 여러 대상을 구성했다고 가정해 보겠습니다. 애플리케이션이 실패에서 복구되면 Kinesis Data Analytics는 가장 느린 목적지로 전송된 마지막 레코드부터 구성된 목적지로 출력을 계속 유지합니다. 이렇게 할 경우 동일한 출력 레코드가 다른 목적지로 두 번 이상 전송될 수 있습니다. 이 경우 대상에서의 잠재적 중복을 외부적으로 처리해야 합니다.

오류 처리

Amazon Kinesis Data Analytics는 API 또는 SQL 오류를 사용자에게 직접 반환합니다. API 작업에 대한 자세한 설명은 작업 섹션을 참조하십시오. SQL 오류 처리에 대한 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조를 참고하십시오.

Amazon Kinesis Data Analytics는 `error_stream`이라고 하는 애플리케이션 내 오류 스트림을 사용하여 런타임 오류를 보고합니다.

애플리케이션 내 오류 스트림을 사용한 오류 보고

Amazon Kinesis Data Analytics는 `error_stream`이라고 하는 애플리케이션 내 오류 스트림에 런타임 오류를 보고합니다. 다음은 발생할 수 있는 오류의 예입니다:

- 스트리밍 소스로부터 읽은 레코드가 입력 스키마에 부합하지 않습니다.
- 애플리케이션 코드가 0으로 나누기를 지정했습니다.
- 행이 순서에 벗어난 상태입니다(예: 레코드의 순서 이탈을 야기하는 사용자가 수정한 `ROWTIME` 값을 지닌 스트림 상에 레코드가 나타나는 경우).
- 소스 스트림에 있는 데이터가 스키마에 지정된 데이터 유형으로 변환되지 않습니다(강제 변환 오류). 변환 가능한 데이터 유형에 관한 정보는 JSON 데이터 유형을 SQL 데이터 유형으로 매핑하기 섹션을 참조하십시오.

SQL 코드에서 이러한 오류를 프로그래밍 방식으로 처리하거나 오류 스트림 상의 데이터를 외부 대상에 유지하는 것이 좋습니다. 이를 위해서는 애플리케이션에 출력 구성을 추가해야 합니다(애플리케이션 출력 구성 참조). 애플리케이션 내 오류 스트림 작동 방식의 예는 예: 애플리케이션 내 오류 스트림 탐색 섹션을 참조하십시오.

> **Note**
>
> 시스템 계정을 사용하여 오류 스트림이 생성되었기 때문에 Kinesis Data Analytics 애플리케이션은 오류 스트림을 프로그래밍 방식으로 액세스하거나 수정할 수 없습니다. 오류 출력을 사용하여 애플리케이션에서 발생할 수 있는 에러가 무엇인지 판단해야 합니다. 그런 다음 애플리케이션의 SQL 코드를 작성하여 예상되는 오류 상태를 처리합니다.

오류 스트림 스키마

오류 스트림의 스키마는 다음과 같습니다.

필드	데이터 형식	참고
ERROR_TIME	TIMESTAMP	오류 발생 시간

ERROR_LEVEL	VARCHAR(10)	
ERROR_NAME	VARCHAR(32)	
MESSAGE	VARCHAR(4096)	
DATA_ROWTIME	TIMESTAMP	수신 레코드의 ROW TIME
DATA_ROW	VARCHAR(49152)	기존 행에 있는 16진수 인코딩 데이터. 표준 라이브러리를 사용하여 이 값을 16진수로 디코딩하거나 이 16진수 - 문자열 변환기 같은 웹 리소스를 사용할 수 있습니다.
PUMP_NAME	VARCHAR(128)	`CREATE PUMP`로 정의된 원본 펌프

처리량 증가를 위해 애플리케이션 용량을 자동으로 확장 또는 축소

Amazon Kinesis Data Analytics는 대부분의 시나리오에서 소스 스트림의 데이터 처리량과 쿼리 복잡성을 수용할 수 있도록 애플리케이션을 탄력적으로 확장합니다. Kinesis Data Analytics는 용량을 Kinesis 처리 단위(KPU) 형식으로 제공합니다. 단일 KPU는 메모리(4GB)와 해당 컴퓨팅 및 네트워킹을 제공합니다.

귀하의 애플리케이션을 위한 KPU의 기본값 한도는 64입니다. 이 한도 증가 요청 지침은 Amazon 서비스 한도에서 한도 상향 요청하기를 참조하십시오.

태그 지정 사용

이 섹션에서는 Kinesis Data Analytics 애플리케이션에 키-값 메타데이터를 추가하는 방법을 설명합니다. 이러한 태그는 다음과 같은 용도로 사용할 수 있습니다:

- 개별 Kinesis Data Analytics 애플리케이션에 대한 청구 결정 자세한 내용을 알아보려면 AWS 청구 및 비용 관리 가이드의 비용 할당 태그 사용을 참조하십시오.
- 태그를 기반으로 애플리케이션 리소스에 대한 액세스 통제. 자세한 설명은 사용자 가이드의 태그를 사용한 액세스 통제를 참조하십시오.

- 사용자 정의 용도. 사용자 태그의 존재 유무를 기반으로 애플리케이션 기능을 정의할 수 있습니다.

태그 지정에 대한 다음 정보를 참조하십시오.

- 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다.
- 작업에 중복된 `Key` 값이 있는 태그 목록이 포함된 경우 서비스에서 `InvalidArgumentException`가 발생합니다.

이 주제는 다음 섹션을 포함하고 있습니다.

- 애플리케이션이 생성될 때 태그 추가
- 기존 애플리케이션에 대한 태그 추가 또는 업데이트
- 애플리케이션에 대한 태그 나열
- 애플리케이션에서 태그 제거

애플리케이션이 생성될 때 태그 추가

CreateApplication 작업의 `tags` 파라미터를 사용하여 애플리케이션을 만들 때 태그를 추가합니다.

다음 예 요청은 `CreateApplication` 요청에 대한 `Tags` 노드를 보여줍니다.

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

기존 애플리케이션에 대한 태그 추가 또는 업데이트

TagResource 작업을 사용하여 애플리케이션에 태그를 추가합니다. UpdateApplication 작업을 사용해서는 애플리케이션에 태그를 추가할 수 없습니다.

기존 태그를 업데이트하려면, 기존 태그의 동일한 키로 태그를 추가합니다.

`TagResource` 작업을 위한 다음 예 요청은 새 태그를 추가하거나 기존 태그를 업데이트합니다.

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

애플리케이션에 대한 태그 나열

기존 태그를 나열하려면 ListTagsForResource 작업을 사용합니다.

`ListTagsForResource` 작업을 위한 다음 예 요청은 애플리케이션에 대한 태그를 나열합니다.

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

애플리케이션에서 태그 제거

애플리케이션에서 태그를 제거하려면 UntagResource 작업을 사용합니다.

`UntagResource` 작업을 위한 다음 요청 예는 애플리케이션에서 태그를 제거합니다.

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

Amazon Kinesis Data Analytics for SQL 애플리케이션 시작하기

아래에서 Amazon Kinesis Data Analytics for SQL 애플리케이션 사용을 시작하는 데 도움이 되는 주제를 다룹니다. Kinesis Data Analytics for SQL 애플리케이션을 처음 사용한다면 시작 섹션의 단계를 수행하기 전에 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식에 나와 있는 개념과 용어를 검토하는 것이 좋습니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 1단계: 계정 설정 및 관리자 사용자 생성
- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 2단계: AWS Command Line Interface (AWS CLI) 설정
- 3단계: 스타터 Amazon Kinesis Data Analytics 애플리케이션 생성
- 4단계 (선택 사항) 콘솔을 이용한 스키마와 SQL 코드 편집

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   가입 절차의 일부로는 전화 또는 문자 메시지를 수신하고 전화 키패드에 확인 코드를 입력하는 것이 포함됩니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

1단계: 계정 설정 및 관리자 사용자 생성

Amazon Kinesis Data Analytics를 처음 사용한다면 먼저 다음 작업을 완료해야 합니다:

1. 에 가입 AWS
2. IAM 사용자 생성

에 가입 AWS

Amazon Web Services에 가입하면 Amazon Kinesis Data Analytics를 AWS포함한의 모든 서비스에가 자동으로 등록 AWS 계정 됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다.

Kinesis Data Analytics에서는 사용한 리소스에 대해서만 비용을 지불합니다. 신규 AWS 고객인 경우 Kinesis Data Analytics를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하세요.

이미이 있는 경우 다음 작업으로 AWS 계정건너뜁니다. 이 없는 경우 다음 절차의 단계를 AWS 계정수행하여 생성합니다.

를 생성하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   가입 절차의 일부로는 전화 또는 문자 메시지를 수신하고 전화 키패드에 확인 코드를 입력하는 것이 포함됩니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

다음 작업에 필요하므로 AWS 계정 ID를 기록해 둡니다.

IAM 사용자 생성

Amazon Kinesis Data Analytics AWS와 같은의 서비스는 서비스에 액세스할 때 자격 증명을 제공해야 해당 서비스가 소유한 리소스에 액세스할 수 있는 권한이 있는지 확인할 수 있습니다. 콘솔은 암호를 요구합니다. AWS CLI 또는 API에 액세스 AWS 계정 하기 위해에 대한 액세스 키를 생성할 수 있습니다. 그러나에 대한 자격 증명을 AWS 사용하여에 액세스하는 것은 권장하지 않습니다 AWS 계정. 대신 AWS Identity and Access Management (IAM) 사용을 권장합니다. IAM 사용자를 생성하여 관리자 권한과 함께 IAM 그룹에 추가한 다음, 생성한 IAM 사용자에게 관리자 권한을 부여하십시오. 그런 다음 특수 URL과 해당 IAM 사용자의 자격 증명을 AWS 사용하여에 액세스할 수 있습니다.

에 가입 AWS했지만 IAM 사용자를 직접 생성하지 않은 경우 IAM 콘솔을 사용하여 사용자를 생성할 수 있습니다.

이 가이드의 시작하기 연습에서는 관리자 권한이 있는 사용자(`adminuser`)가 있다고 가정합니다. 절차에 따라 계정에서 `adminuser`를 만듭니다.

관리자 사용자를 만들고 콘솔에 로그인하려면

1. 귀하의 AWS 계정계정에서 `adminuser`라는 관리자 사용자를 만듭니다. 관련 지침은 IAM 사용자 가이드의 첫 번째 IAM 사용자 및 관리자 그룹 생성 섹션을 참조하십시오.
2. 사용자는 특수 URL을 AWS Management Console 사용하여에 로그인할 수 있습니다. 자세한 설명은 IAM 사용자 가이드의 사용자 계정에 로그인하는 방법을 참조하세요.

IAM에 대한 자세한 설명은 다음을 참조하세요.

- AWS Identity and Access Management (IAM)
- 시작하기
- IAM 사용자 가이드

다음 단계

2단계: AWS Command Line Interface (AWS CLI) 설정

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   가입 절차의 일부로는 전화 또는 문자 메시지를 수신하고 전화 키패드에 확인 코드를 입력하는 것이 포함됩니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

2단계: AWS Command Line Interface (AWS CLI) 설정

단계에 따라 AWS Command Line Interface ()를 다운로드하고 구성합니다AWS CLI.

> **Important**
>
> 시작하기 연습의 단계를 수행하는 AWS CLI 데가 필요하지 않습니다. 이 안내서의 일부 연습에서는 AWS CLI를 사용합니다. 이 단계를 건너뛰고 로 이동 3단계: 스타터 Amazon Kinesis Data Analytics 애플리케이션 생성한 다음 AWS CLI 나중에 필요할 때를 설정할 수 있습니다.

를 설정하려면 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 주제를 참조하세요.

   - 를 사용하여 설정 AWS Command Line Interface
   - 구성 AWS Command Line Interface

2. 구성 파일에 관리자 사용자의 AWS CLI 명명된 프로필을 추가합니다. AWS CLI 명령을 실행할 때 이 프로파일을 사용합니다. 프로파일 명명에 대한 자세한 설명은AWS Command Line Interface 사용자 가이드의 프로파일 명명을 참조하십시오.

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

   사용 가능한 목록은의 리전 및 엔드포인트를 AWS 리전참조하세요Amazon Web Services 일반 참조.

3. 명령 프롬프트에서 다음 help 명령을 입력하여 설정을 확인하십시오:

   ```
   aws help
   ```

다음 단계

3단계: 스타터 Amazon Kinesis Data Analytics 애플리케이션 생성

3단계: 스타터 Amazon Kinesis Data Analytics 애플리케이션 생성

이 섹션에 제시된 단계에 따라 먼저 콘솔을 사용하여 귀하의 첫 번째 Kinesis Data Analytics 애플리케이션을 생성할 수 있습니다.

> **Note**
> 시작하기 연습을 수행하기 전에 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 섹션을 복습하는 것이 좋습니다.

이 시작하기 연습을 위해 콘솔을 이용하여 애플리케이션 코드가 있는 템플릿이나 데모 스트림으로 작업할 수 있습니다.

- 데모 스트림을 사용하기로 선택하는 경우, 콘솔이 `kinesis-analytics-demo-stream`이라는 Kinesis 데이터 스트림을 귀하의 계정에 생성합니다.

  Kinesis Data Analytics 애플리케이션에는 스트리밍 소스가 필요합니다. 이 소스의 경우, 이 설명서의 몇몇 SQL 예가 데모 스트림 `kinesis-analytics-demo-stream`을 사용합니다. 콘솔은 다음과 같이 이 스트림에 샘플 데이터(시뮬레이션된 주식 거래 레코드)를 지속적으로 추가하는 스크립트를 실행합니다.

  Raw | Lambda output | Formatted

  Filter by column name or column type

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

  이 연습에서는 `kinesis-analytics-demo-stream`을 애플리케이션의 스트리밍 소스로 사용할 수 있습니다.

  > **Note**
  >
  > 데모 스트림은 계정에 남아 있습니다. 이 설명서에서 다른 예를 테스트하는 데 사용할 수 있습니다. 그러나 콘솔을 벗어나면 콘솔이 사용하는 스크립트가 데이터 채우기를 중단합니다. 필요한 경우, 콘솔이 스트림 채우기 재시작 옵션을 제공합니다.

- 예 애플리케이션 코드가 있는 템플릿을 사용하기로 선택하는 경우, 데모 스트림에 대한 간단한 분석을 수행할 수 있도록 콘솔이 제공하는 템플릿 코드를 활용합니다.

이 기능들을 사용하여 다음과 같인 첫 번째 애플리케이션을 신속하게 설정합니다.

1. 애플리케이션 생성 – 명칭만 제공하면 됩니다. 콘솔이 애플리케이션을 생성하고 서비스가 애플리케이션 상태를 READY로 설정합니다.

2. 입력 구성 – 먼저 스트리밍 소스인 데모 스트림을 추가합니다. 사용하기 전에 콘솔에서 데모 스트림을 생성해야 합니다. 그런 다음, 콘솔이 데모 스트림에서 레코드 샘플을 무작위로 취하고 생성된 애플리케이션 내 입력 스트림에 대한 스키마를 유추합니다. 콘솔 명칭은 애플리케이션 내 스트림 SOURCE_SQL_STREAM_001입니다.

   콘솔은 검색 API를 사용하여 스키마를 유추합니다. 필요할 경우 유추된 스키마를 직접 편집할 수 있습니다. 자세한 설명은 DiscoverInputSchema 섹션을 참조하십시오. Kinesis Data Analytics는 이 스키마를 사용하여 애플리케이션 내 스트림을 생성합니다.

   애플리케이션을 시작하면 Kinesis Data Analytics이 여러분을 대신하여 지속적으로 데모 스트림을 읽고 SOURCE_SQL_STREAM_001 애플리케이션 내 입력 스트림에 행을 삽입합니다.

3. 애플리케이션 코드 지정 – 다음 코드를 제공하는 템플릿(연속 필터라고 함)을 사용합니다:

```
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%';
```

   애플리케이션 코드가 애플리케이션 내 스트림 SOURCE_SQL_STREAM_001을 쿼리합니다. 그러면 코드가 펌프를 사용하여 다른 애플리케이션 내 스트림 DESTINATION_SQL_STREAM에 결과 행을 삽입합니다. 이 코딩 패턴에 대한 자세한 설명은 애플리케이션 코드 섹션을 참조하십시오.

   Kinesis Data Analytics에서 지원되는 SQL 언어 요소에 대한 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조를 참조하십시오.

4. 출력 구성 – 이 실습에서는 출력을 구성하지 않습니다. 즉, 애플리케이션이 임의의 외부 대상에 생성하는 애플리케이션 내 스트림에 데이터를 유지하지 않는다는 의미입니다. 대신 콘솔에서 쿼리 결과를 확인합니다. 이 설명의 추가 예는 출력을 구성하는 방법을 보여줍니다. 그 중 한 예를 보여주는 예: 간단한 알림 생성 섹션을 참조하십시오.

> **Important**
>
> 이 연습에서는 미국 동부 (버지니아 북부) 리전(us-east-1)을 사용하여 애플리케이션을 설정합니다. 지원되는 중 하나를 사용할 수 있습니다 AWS 리전.

다음 단계

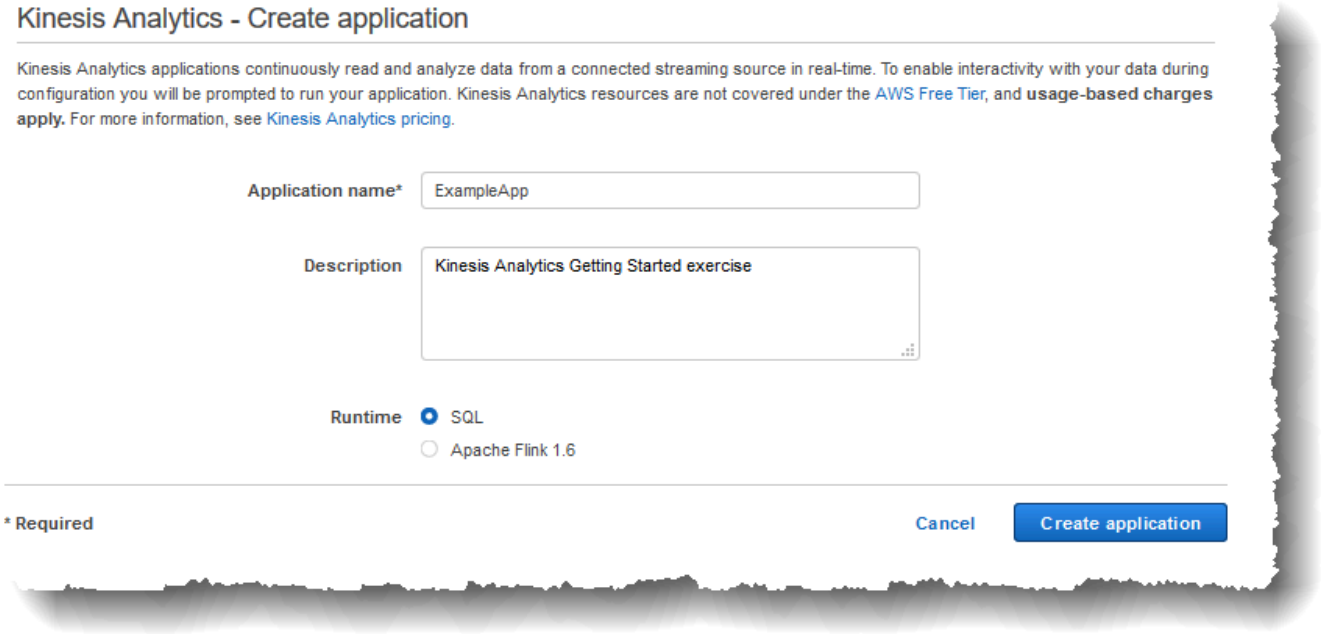
단계 3.1: 애플리케이션 만들기

단계 3.1: 애플리케이션 만들기

이 섹션에서는 Amazon Kinesis Data Analytics 애플리케이션을 생성합니다. 다음 단계에서는 애플리케이션 입력을 구성합니다.

데이터 분석 애플리케이션을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesisanalytics Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택합니다.
3. 애플리케이션 생성 페이지에 애플리케이션 명칭과 설명을 입력하고 애플리케이션의 런타임 설정용 SQL을 선택한 후, 애플리케이션 생성을 선택합니다.



그러면 Kinesis Data Analytics 애플리케이션이 준비 상태로 생성됩니다. 압력 및 출력을 구성할 수 있는 애플리케이션 허브가 콘솔에 표시됩니다.

> **Note**
>
> 애플리케이션 생성을 위한 CreateApplication 작업에는 애플리케이션 명칭만 있으면 됩니다. 콘솔에서 애플리케이션을 생성한 후에 입력 및 출력 구성을 추가할 수 있습니다.

다음 단계에서는 애플리케이션에 대한 입력을 구성합니다. 입력 구성에서 스트리밍 데이터 소스를 애플리케이션에 추가하고 스트리밍 소스에서 데이터를 샘플링하여 애플리케이션 내 입력 스트림에 대한 스키마를 검색합니다.

다음 단계

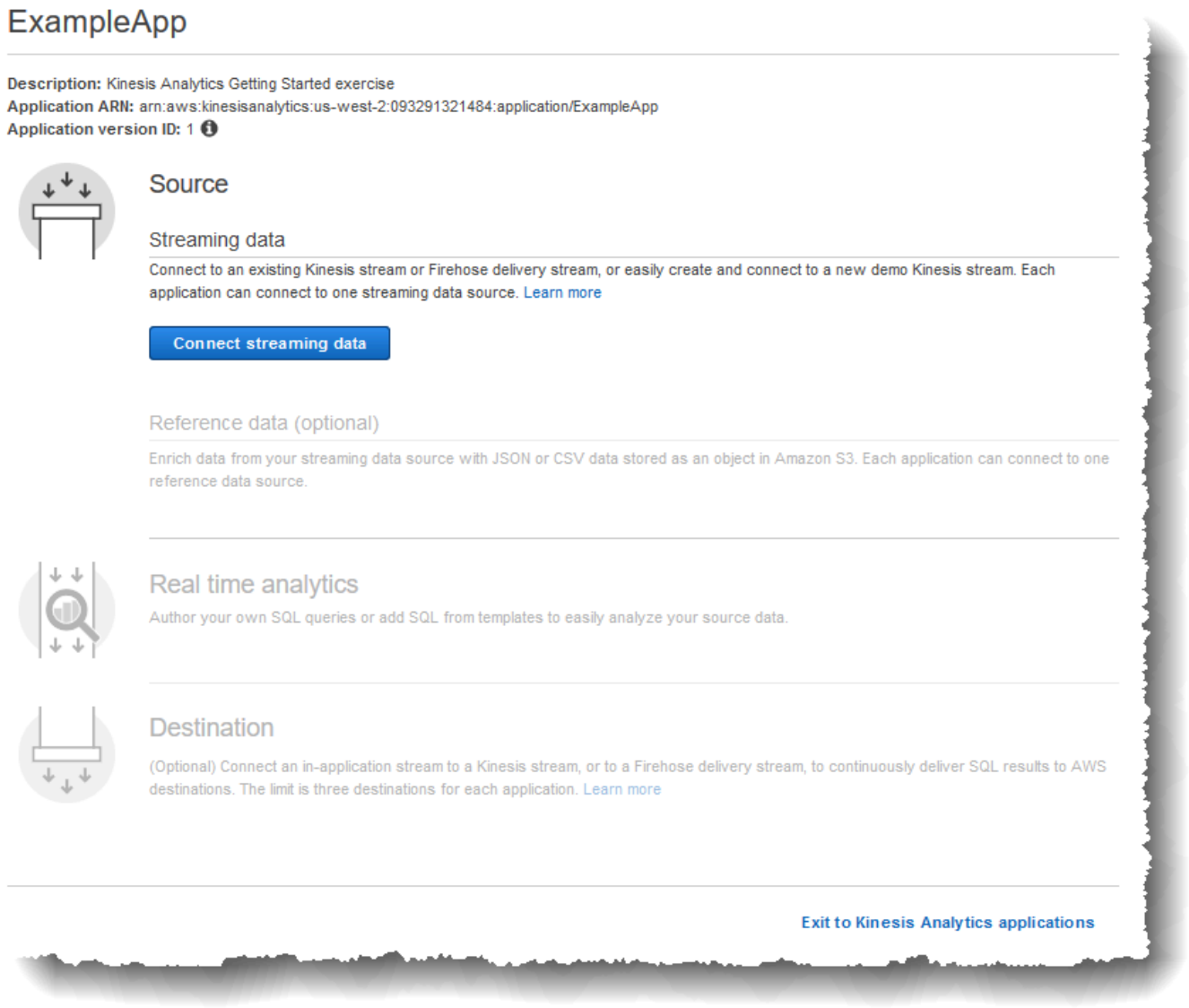
3.2단계: 입력 구성

3.2단계: 입력 구성

애플리케이션은 스트리밍 소스가 필요합니다. 용이한 시작을 위해 콘솔이 데모 스트림(`kinesis-analytics-demo-stream`)을 생성할 수 있습니다. 또한 콘솔은 레코드를 스트림에 채우는 스크립트를 실행합니다.

스트리밍 소스를 애플리케이션에 추가하려면

1. 콘솔의 애플리케이션 허브에서 Connect streaming data(스트리밍 데이터 연결)를 선택합니다.



2. 이어 나타나는 페이지에서 다음을 검토합니다.

   - 애플리케이션의 스트리밍 소스를 지정하는 [Source] 항목입니다. 기존 스트림 소스를 선택하거나 새로 만들 수 있습니다. 이 실습에서는 새 스트림(데모 스트림)을 생성합니다.

기본 설정으로 콘솔은 생성된 애플리케이션 내 입력 스트림을 `INPUT_SQL_STREAM_001`로 명명합니다. 이 실습에서는 처음부터 이 명칭을 유지합니다.

- 스트림 참조 명칭 – 이 옵션은 생성된 애플리케이션 내 입력 스트림(`SOURCE_SQL_STREAM_001`)의 명칭을 표시합니다. 이 명칭을 변경할 수 있지만 이 실습에서는 이 명칭을 유지합니다.

입력 구성에서 데모 스트림을 생성된 애플리케이션 내 입력 스트림에 매핑합니다. 애플리케이션을 시작하면 Amazon Kinesis Data Analytics가 연속적으로 데모 스트림을 읽고 애플리케이션 내 입력 스트림에 행을 삽입합니다. 이 애플리케이션 내 입력을 애플리케이션 코드에서 쿼리합니다.

- 를 사용한 레코드 사전 처리 AWS Lambda:이 옵션은 애플리케이션 코드가 실행되기 전에 입력 스트림의 레코드를 수정하는 AWS Lambda 표현식을 지정하는 곳입니다. 이 연습에서는 [Disabled] 옵션을 선택한 상태로 둡니다. Lambda 사전 처리에 대한 자세한 정보는 Lambda 함수를 사용하여 데이터 사전 처리을 참조하십시오.

이 페이지에서 모든 정보를 제공하고 나면 콘솔이 업데이트 요청을 전송하여(UpdateApplication 참조) 애플리케이션에 입력 구성을 추가합니다.

3. [Source] 페이지에서 [Configure a new stream]을 선택합니다.
4. [Create demo stream]을 선택합니다. 콘솔은 다음을 수행하여 애플리케이션 입력을 구성합니다.
   - 이 콘솔은 `kinesis-analytics-demo-stream`이라는 Kinesis 데이터 스트림을 생성합니다.
   - 콘솔은 주식 티커 데이터 샘플을 스트림에 채웁니다.
   - DiscoverInputSchema 입력 작업을 통해 콘솔은 스트림에서 샘플 레코드를 읽어 스키마를 유추합니다. 유추되는 스키마는 생성된 애플리케이션 내 입력 스트림에 대한 스키마입니다. 자세한 설명은 애플리케이션 입력 구성 섹션을 참조하십시오.
   - 콘솔은 유추된 스키마와 스트리밍 소스로부터 읽어 스키마를 유추한 샘플 데이터를 보여줍니다.

콘솔은 스트리밍 소스에 대한 샘플 레코드를 표시합니다.

Raw | Lambda output | Formatted

Filter by column name or column type

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

[Stream sample] 콘솔 페이지에 다음 항목이 표시됩니다.

- [Raw stream sample] 탭은 스키마 유추를 위해 DiscoverInputSchema API 작업을 통해 샘플링한 원시 스트림 레코드를 보여줍니다.
- [Formatted stream sample] 탭은 [Raw stream sample] 탭에 있는 데이터의 표 버전을 보여 줍니다.
- [Edit schema]를 선택하면 유추된 스키마를 편집할 수 있습니다. 이 실습에서는 유추된 스키마를 변경하지 않습니다. 스키마 편집에 대한 자세한 설명은 스키마 편집기로 작업 섹션을 참조하십시오.

  [Rediscover schema]를 선택하면 콘솔이 DiscoverInputSchema를 다시 실행하고 스키마를 유추하도록 요청할 수 있습니다.

5. [Save and continue]를 선택합니다.

   이제 애플리케이션에 입력 구성을 추가했습니다. 다음 단계에서는 SQL 코드를 추가하여 데이터 애플리케이션 내 입력 스트림에 대한 분석을 수행합니다.

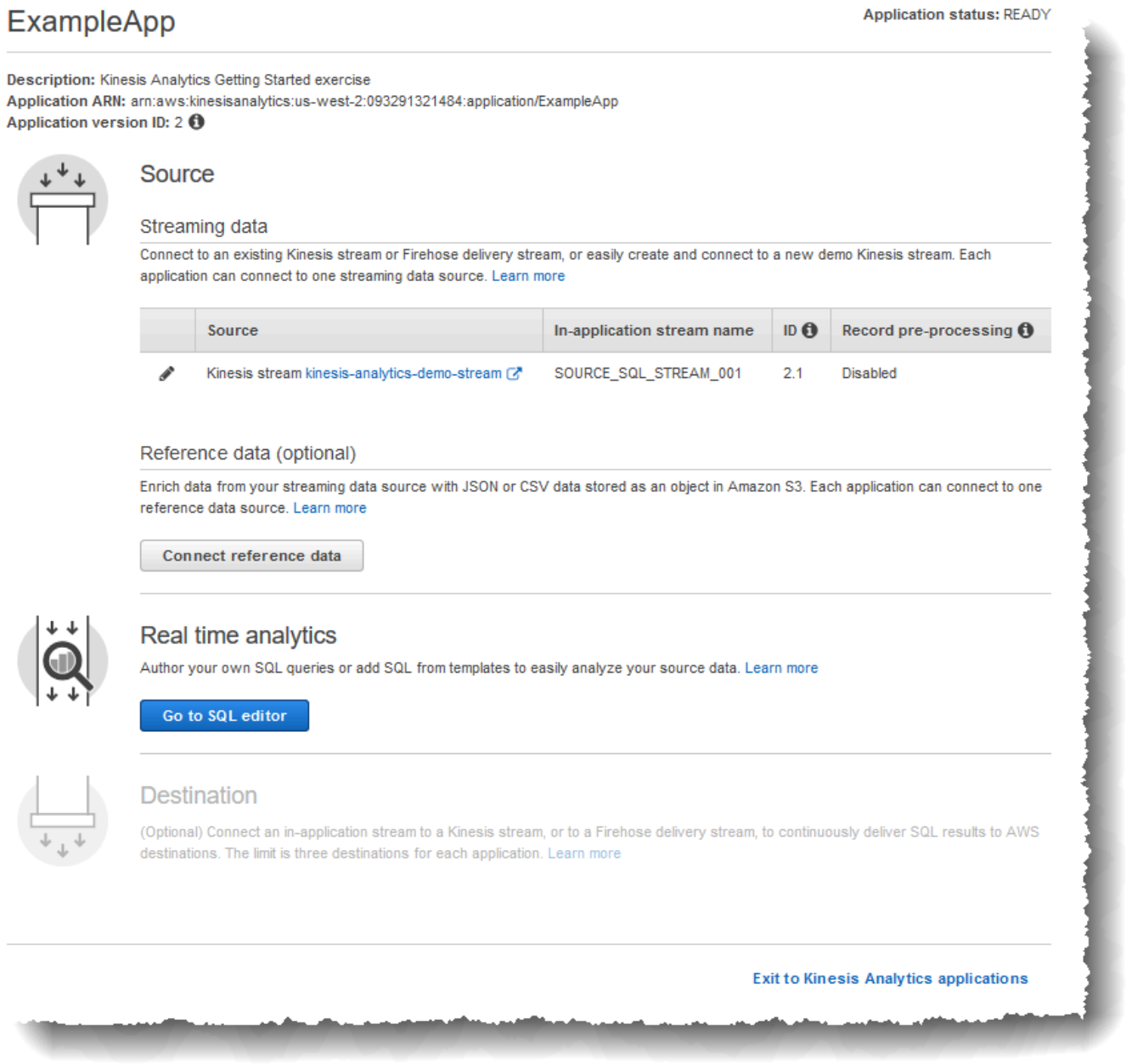
다음 단계

[3.3단계: 실시간 분석 추가(애플리케이션 코드 추가)]

3.3단계: 실시간 분석 추가(애플리케이션 코드 추가)

애플리케이션 내 스트림에 대해 자체 SQL 쿼리를 작성할 수 있지만, 다음 단계에는 샘플 코드를 제공하는 템플릿 중 하나를 사용합니다.

1. 애플리케이션 허브 페이지에서 [Go to SQL editor]를 선택합니다.



2. Would you like to start running "ExampleApp"?("ExampleApp"을 실행하시겠습니까?) 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.

   콘솔이 애플리케이션 시작 요청을 전송하고(StartApplication 참조), SQL 편집기 페이지가 나타납니다.

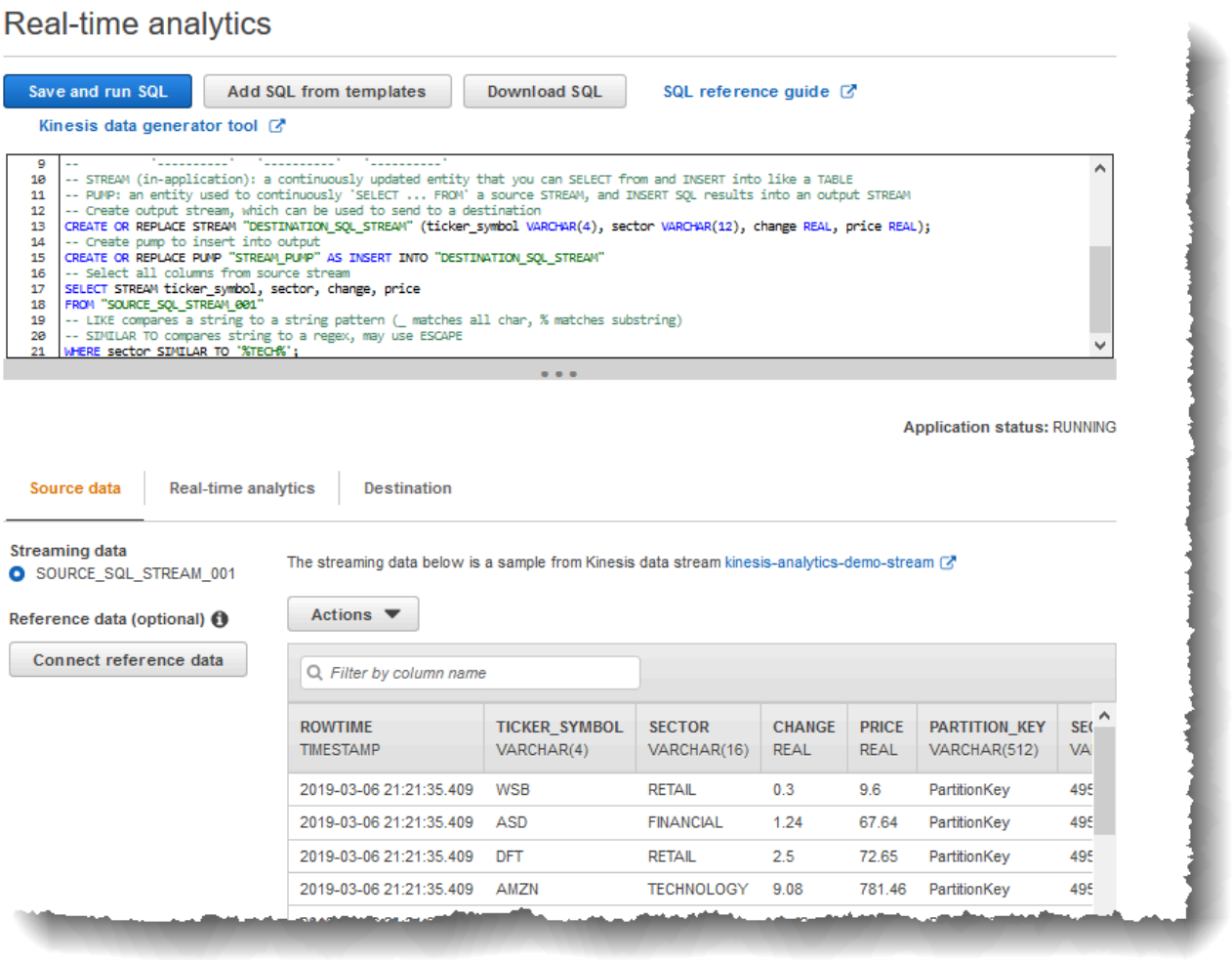
3. 콘솔이 SQL 편집기 페이지를 엽니다. 버튼([Add SQL from templates], [Save and run SQL]) 및 다양한 탭을 포함하여 페이지를 검토합니다.
4. SQL 편집기에서 [Add SQL from templates]를 선택합니다.
5. 가용한 템플릿 목록에서 [Continuous filter]를 선택합니다. 샘플 코드는 하나의 애플리케이션 내 스트림으로부터 오는 데이터를 읽고(`WHERE` 절이 행을 필터링) 다음과 같이 그것을 다른 애플리케이션 내 스트림에 삽입합니다.

   - 애플리케이션 내 스트림 `DESTINATION_SQL_STREAM`을 생성합니다.
   - 펌프 `STREAM_PUMP`를 생성하고, 생성한 펌프를 사용하여 `SOURCE_SQL_STREAM_001`에서 행을 선택한 다음 `DESTINATION_SQL_STREAM`에 삽입합니다.

6. [Add this SQL to editor]를 선택합니다.
7. 다음과 같이 애플리케이션 코드를 시험합니다.

   애플리케이션을 이미 시작했다는 점을 명심하십시오(상태는 RUNNING), 그러므로 Amazon Kinesis Data Analytics는 이미 스트리밍 소스로부터 지속적으로 읽고 행을 애플리케이션 내 스트림 `SOURCE_SQL_STREAM_001`에 추가하고 있습니다.

   a. SQL 편집기에서 [Save and run SQL]을 선택합니다. 우선 콘솔이 업데이트 요청을 전송하여 애플리케이션 코드를 저장합니다. 그런 다음, 코드가 연속적으로 실행됩니다.
   b. [Real-time analytics] 탭에서 결과를 확인할 수 있습니다.



SQL 편집기에는 다음과 같은 탭이 있습니다.

- [Source data] 탭에서는 스트리밍 소스로 매핑되는 애플리케이션 내 입력 스트림을 확인할 수 있습니다. 애플리케이션 내 스트림을 선택하면 수신되는 데이터를 확인할 수 있습니다. 입력 구성에서 지정되지 않은 애플리케이션 내 입력 스트림에서의 추가 열에 주목합니다. 여기에는 다음과 같은 타임스탬프 열이 포함됩니다:

  - ROWTIME – 애플리케이션 내 스트림에 있는 각 열에는 `ROWTIME`라고 하는 특수 열이 있습니다. 이 열은 Amazon Kinesis Data Analytics가 첫 번째 애플리케이션 내 스트림 (스트리밍 소스에 매핑되는 애플리케이션 내 입력 스트림)에 행을 삽입할 때의 타임스탬프입니다.

- Approximate_Arrival_Time – 각 Kinesis Data Analytics 레코드에는 `Approximate_Arrival_Time`라는 값이 포함됩니다. 이 값은 스트리밍 소스가 레코드를 성공적으로 수신하여 저장하는 시점에 정해지는 대략적인 도착 타임스탬프입니다. Kinesis Data Analytics가 스트리밍 소스로부터 레코드를 읽을 때 이 열을 애플리케이션 내 입력 스트림으로 가져옵니다.

이들 타임스탬프 값은 시간 기반 윈도우 모드 쿼리에 유용합니다. 자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.

- [Real-time analytics] 탭에서는 애플리케이션 코드에서 생성한 다른 모든 애플리케이션 내 스트림을 확인할 수 있습니다. 여기에는 오류 스트림도 포함됩니다. Kinesis Data Analytics는 처리할 수 없는 모든 행을 오류 스트림으로 보냅니다. 자세한 설명은 오류 처리 섹션을 참조하십시오.

`DESTINATION_SQL_STREAM`을 선택하여 애플리케이션 코드가 삽입한 행을 확인합니다. 애플리케이션이 생성하지 않은 추가 열에 주목합니다. 이러한 열에는 `ROWTIME` 타임스탬프 열이 포함됩니다. Kinesis Data Analytics는 단순히 소스 (`SOURCE_SQL_STREAM_001`)에서 이러한 값을 복사합니다.

- 목적지 탭에는 Kinesis Data Analytics가 쿼리 결과를 작성하는 외부 목적지가 나타납니다. 아직 애플리케이션 출력의 외부 대상을 구성하지 않았습니다.

다음 단계

3.4단계: 애플리케이션 코드 업데이트(선택 사항)

3.4단계: 애플리케이션 코드 업데이트(선택 사항)

이 단계에서는 애플리케이션 코드 업데이트 방법을 알아봅니다.

애플리케이션 코드를 업데이트하려면

1. 다음과 같이 또 다른 애플리케이션 내 스트림을 생성합니다.

- DESTINATION_SQL_STREAM_2라고 하는 또 다른 애플리케이션 내 스트림을 생성합니다.
- 펌프를 생성한 다음 생성된 펌프로 DESTINATION_SQL_STREAM에서 행을 선택하여 새로 생성된 스트림에 행을 삽입합니다.

SQL 편집기에서 다음 코드를 기존 애플리케이션 코드에 추가합니다.

```
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";
```

코드를 저장하고 실행합니다. 추가 애플리케이션 내 스트림이 [Real-time analytics] 탭에 표시됩니다.

2. 두 개의 애플리케이션 내 스트림을 생성합니다. 주식 티커를 바탕으로 SOURCE_SQL_STREAM_001에 있는 행을 필터링한 다음 생성된 별도의 스트림에 삽입합니다.

   다음의 SQL 문을 애플리케이션 코드에 추가합니다.

```
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%';
```

코드를 저장하고 실행합니다. 추가 애플리케이션 내 스트림이 [Real-time analytics] 탭에 표시되는지 확인합니다.

이제 Amazon Kinesis Data Analytics 애플리케이션을 처음으로 사용할 수 있습니다. 이 실습에서 다음 작업을 수행했습니다:

- 최초의 Kinesis Data Analytics 애플리케이션을 생성했습니다.
- 데모 스트림을 스트리밍 소스로 식별하여 생성된 애플리케이션 내 스트림 (`SOURCE_SQL_STREAM_001`)에 매핑하도록 애플리케이션을 구성했습니다. Kinesis Data Analytics가 지속적으로 데모 스트림을 읽고 애플리케이션 내 스트림에 레코드를 삽입합니다.
- 애플리케이션 코드가 `SOURCE_SQL_STREAM_001`을 쿼리하고 `DESTINATION_SQL_STREAM`라고 하는 또 다른 애플리케이션 내 스트림에 출력을 작성했습니다.

이제 선택적으로 애플리케이션 출력을 외부 대상에 작성하도록 애플리케이션 출력을 구성할 수 있습니다. 즉, `DESTINATION_SQL_STREAM`에 있는 레코드를 외부 대상에 작성하도록 애플리케이션 출력을 구성할 수 있습니다. 이 연습에서는 선택 사항 단계입니다. 대상을 구성하는 방법을 알아보려면 다음 단계로 이동합니다.

다음 단계

4단계 (선택 사항) 콘솔을 이용한 스키마와 SQL 코드 편집.

4단계 (선택 사항) 콘솔을 이용한 스키마와 SQL 코드 편집

아래에서 유추된 스키마를 편집하는 방법과 Amazon Kinesis Data Analytics에 사용할 SQL 코드를 편집하는 방법을 확인할 수 있습니다. Kinesis Data Analytics 콘솔에 속하는 SQL 편집기와 스키마 편집기로 작업하면 됩니다.

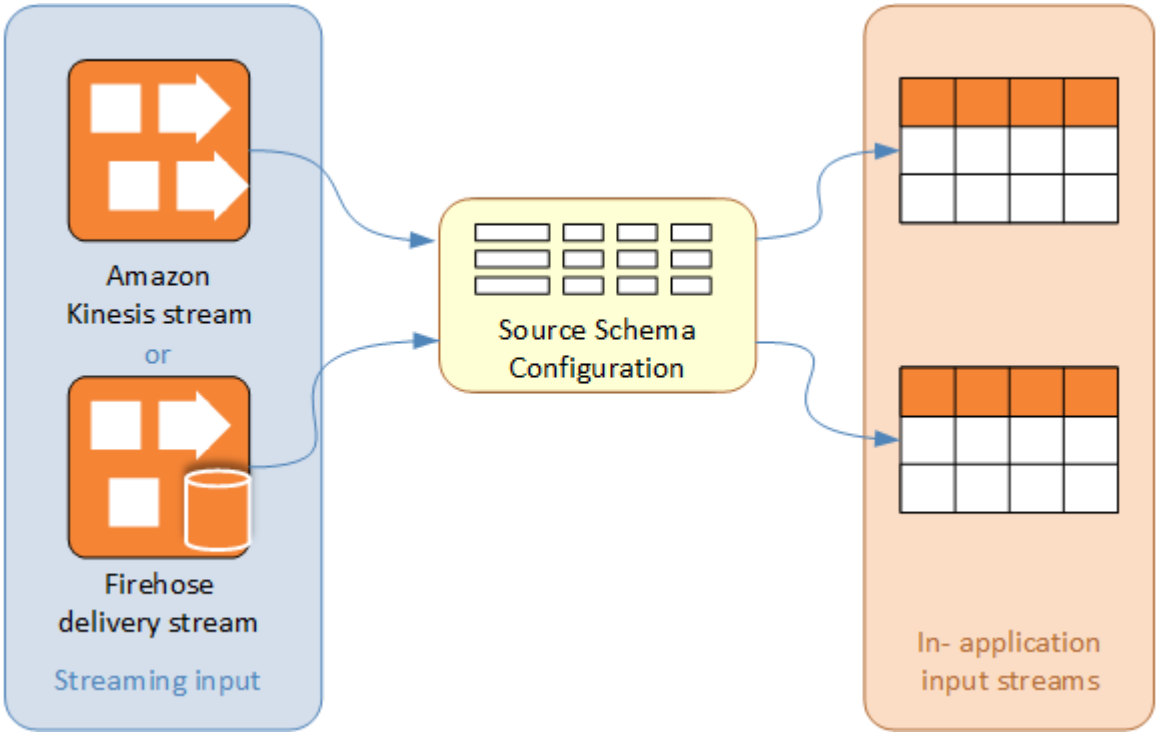
> **Note**
>
> 콘솔에서 데이터에 액세스하거나 샘플링하려면 로그인 사용자의 역할에 `kinesisanalytics:GetApplicationState` 권한이 있어야 합니다. Kinesis Data Analytics 애플리케이션 권한에 대한 자세한 설명은 액세스 관리 개요을 참조하십시오.

주제

- 스키마 편집기로 작업
- SQL 편집기 작업

스키마 편집기로 작업

Amazon Kinesis Data Analytics 애플리케이션 입력 스트림의 스키마는 스트림에서 나오는 데이터를 애플리케이션에서 SQL 쿼리가 사용할 수 있는 방식을 정의합니다.



스키마에는 스트리밍 입력 중 어느 부분을 애플리케이션 입력에 있는 데이터 열로 변환할지 결정하는 선택 기준이 포함되어 있습니다. 이 입력은 다음 중 하나일 수 있습니다.

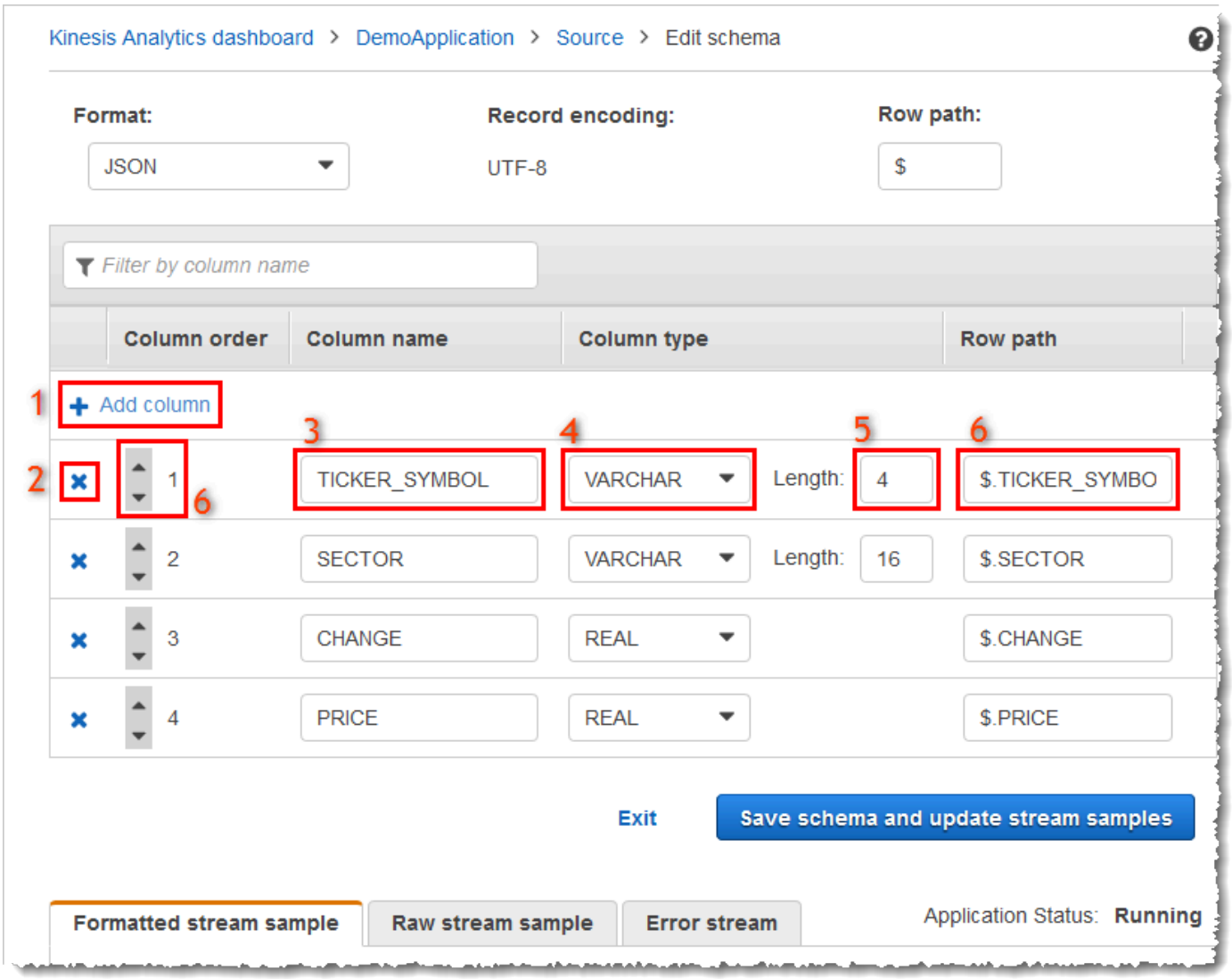
- JSON 입력 스트림에 대한 JSONPath 표현식입니다. JSONPath는 JSON 데이터 쿼리를 위한 도구입니다.

- CSV(쉼표로 분리된 값) 형식의 입력 스트림에 대한 열 번호입니다.
- 애플리케이션 내 데이터 스트림에 데이터를 표현하기 위한 열 명칭과 SQL 데이터 유형입니다. 데이터 유형에는 문자 또는 이진 데이터의 길이 또한 포함됩니다.

콘솔은 DiscoverInputSchema를 사용해서 스키마 생성을 시도합니다. 스키마 검색이 실패하거나 부정확한 또는 불완전한 스키마를 반환하는 경우, 스키마 편집기를 이용해 스키마를 수동으로 편집해야 합니다.

스키마 편집기 기본 화면

다음 스크린샷은 스키마 편집기의 기본 화면을 보여줍니다.



다음과 같은 편집 작업을 스키마에 적용할 수 있습니다.

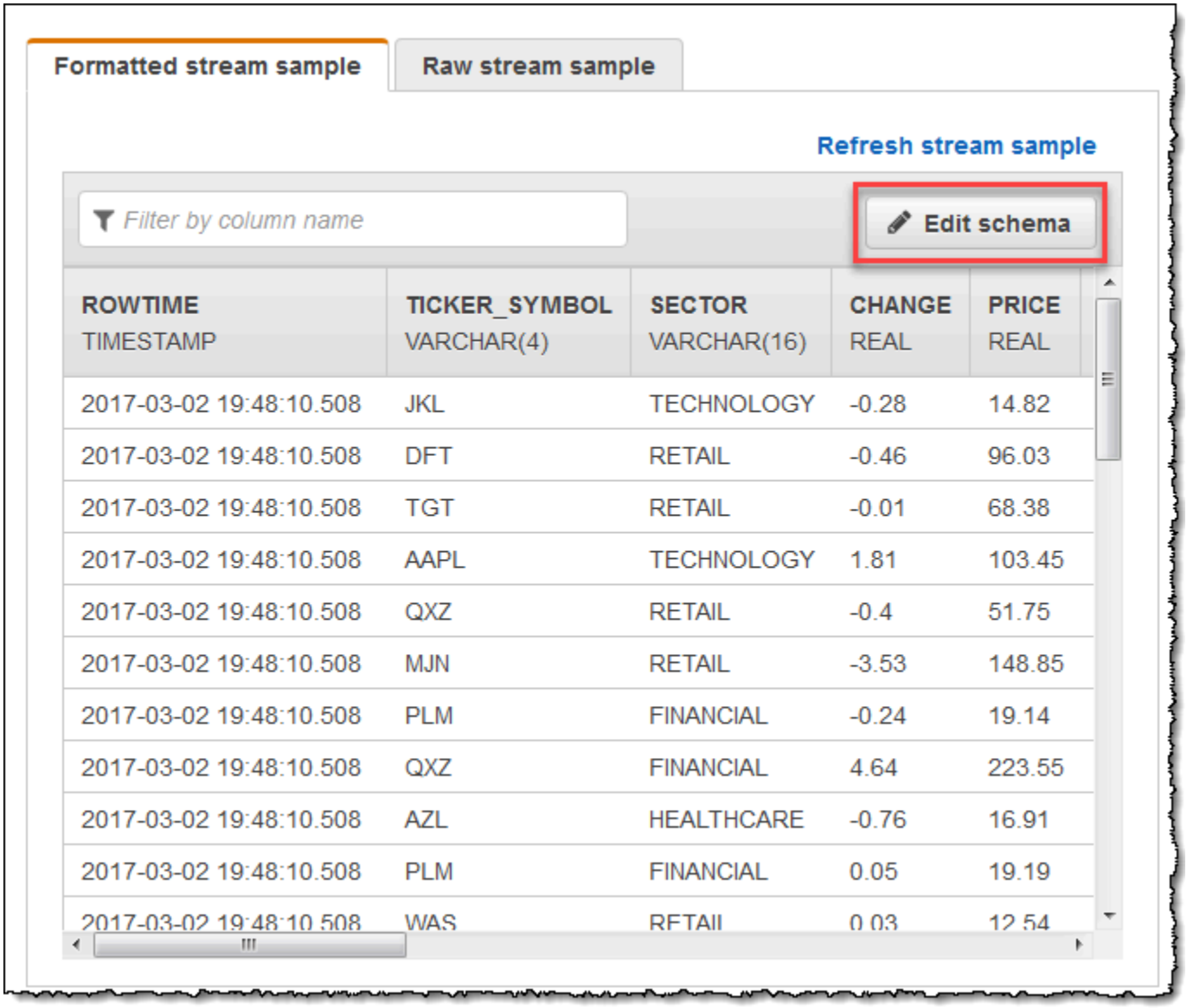
- 열 추가(1): 데이터 항목이 자동으로 감지되지 않는 경우 열을 추가해야 할 수 있습니다.
- 열 삭제(2): 애플리케이션이 필요로 하지 않는 경우 데이터를 소스 스트림으로부터 제외할 수 있습니다. 이렇게 제외된 경우에도 소스 스트림에 있는 데이터에 영향을 주지 않습니다. 데이터가 제외되는 경우 해당 데이터를 애플리케이션에서 사용할 수 없습니다.
- 열 명칭 바꾸기 (3): 열 명칭은 공백일 수 없고, 길이가 한 문자 이상이어야 하며, 예약된 SQL 키워드가 포함되어서는 안 됩니다. 명칭은 SQL 일반 식별자의 명명 기준에 부합해야 합니다. 즉, 문자로 시작해야 하고 문자, 밑줄 및 숫자만 포함해야 합니다.
- 열의 데이터 유형(4) 또는 길이(5) 변경: 열에 대해 호환 가능 데이터 유형을 지정할 수 있습니다. 비호환 데이터 유형을 지정하는 경우 열이 NULL로 채워지거나 애플리케이션 내 스트림이 채워지지 않습니다. 후자의 경우 오류가 오류 스트림에 작성됩니다. 열의 길이를 너무 작게 지정하는 경우 수신 데이터가 잘립니다.
- 열 선택 기준 변경(6): 열의 데이터 소스를 결정하는 데 사용되는 JSONPath 표현식이나 CSV 열 순서를 편집할 수 있습니다. JSON 스키마의 선택 기준을 변경하려면 행 경로 표현식에 대한 새 값을 입력합니다. CSV 스키마는 열 순서를 선택 기준으로 사용합니다. CSV 스키마에 대한 선택 기준을 변경하려면 열 순서를 변경합니다.

스트리밍 소스에 대한 스키마 편집

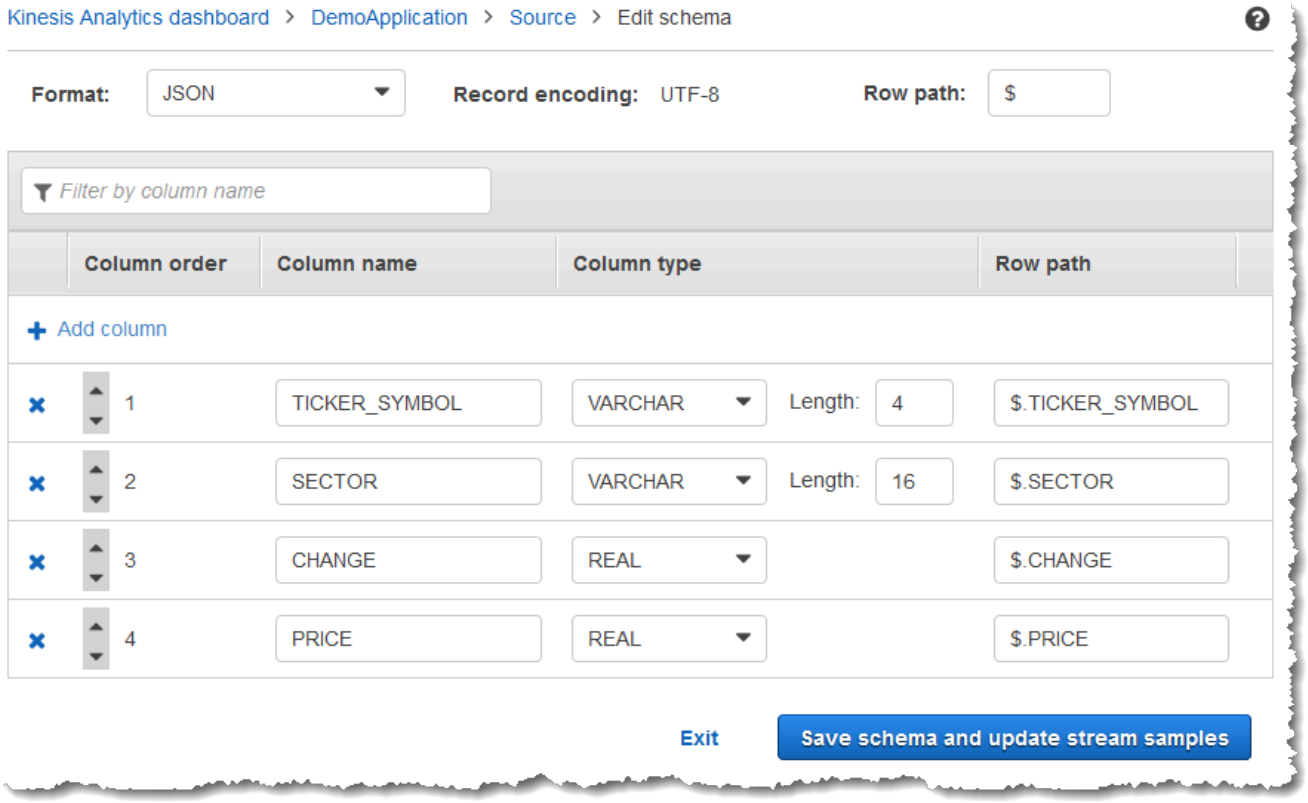
스트리밍 소스에 대한 스키마를 편집해야 하는 경우 다음 단계를 따릅니다.

스트리밍 소스에 대한 스키마 편집 방법

1. 소스 페이지에서 Edit schema(스키마 편집)를 선택합니다.



2. [Edit schema] 페이지에서 소스 스키마를 편집합니다.



3. 형식(Format)에서 JSON 또는 CSV를 선택합니다. JSON 또는 CSV 형식의 경우, 지원되는 인코딩은 ISO 8859-1입니다.

JSON 또는 CSV 형식의 스키마 편집에 대한 자세한 정보는 다음 섹션에 수록된 절차를 참조하십시오.

JSON 스키마 편집

다음 단계에 따라 JSON 스키마를 편집할 수 있습니다.

JSON 스키마를 편집하려면

1. 스키마 편집기에서 [Add column]을 선택하여 열을 추가합니다.

   첫 번째 열 위치에서 새 열이 나타납니다. 열 순서를 변경하려면 열 명칭 옆에 있는 위/아래 화살표를 선택합니다.

   새 열에 대해 다음의 정보를 제공합니다.

   - [Column name]에서 명칭을 입력합니다.

열 명칭은 공백일 수 없고, 길이가 한 문자 이상이어야 하며, 예약된 SQL 키워드가 포함되어서는 안 됩니다. 명칭은 SQL 일반 식별자의 명명 기준에 부합해야 합니다. 즉, 문자로 시작해야 하고 문자, 밑줄 및 숫자만 포함해야 합니다.

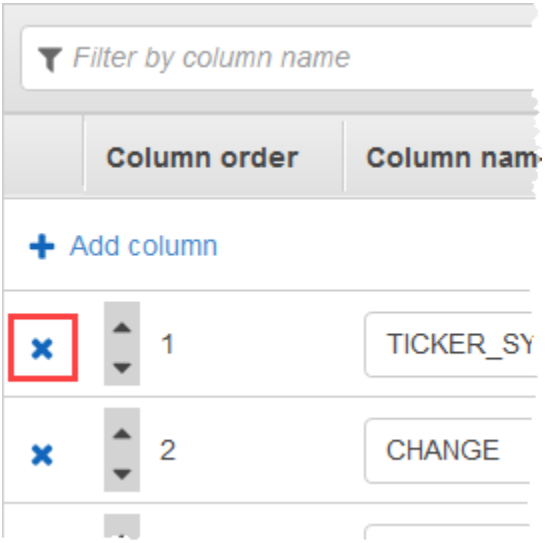
- [Column type]에서 SQL 데이터 유형을 입력합니다.

  열 유형은 지원되는 SQL 데이터 유형이면 됩니다. 새 데이터 유형이 CHAR, VARBINARY 또는 VARCHAR인 경우 [Length]에서 데이터 길이를 지정합니다. 자세한 정보는 데이터 형식을 참조하십시오.

- [Row path]에서 행 경로를 입력합니다. 행 경로는 JSON 요소로 매핑된 유효한 JSONPath 표현식입니다.

> **Note**
>
> 기본 [Row path] 값은 가져올 데이터를 포함하는 최상위 구성 요소로 향하는 경로입니다. 기본적으로 이 값은 $입니다. 자세한 설명은 `JSONMappingParameters`에서 `RecordRowPath` 섹션을 참조하십시오.

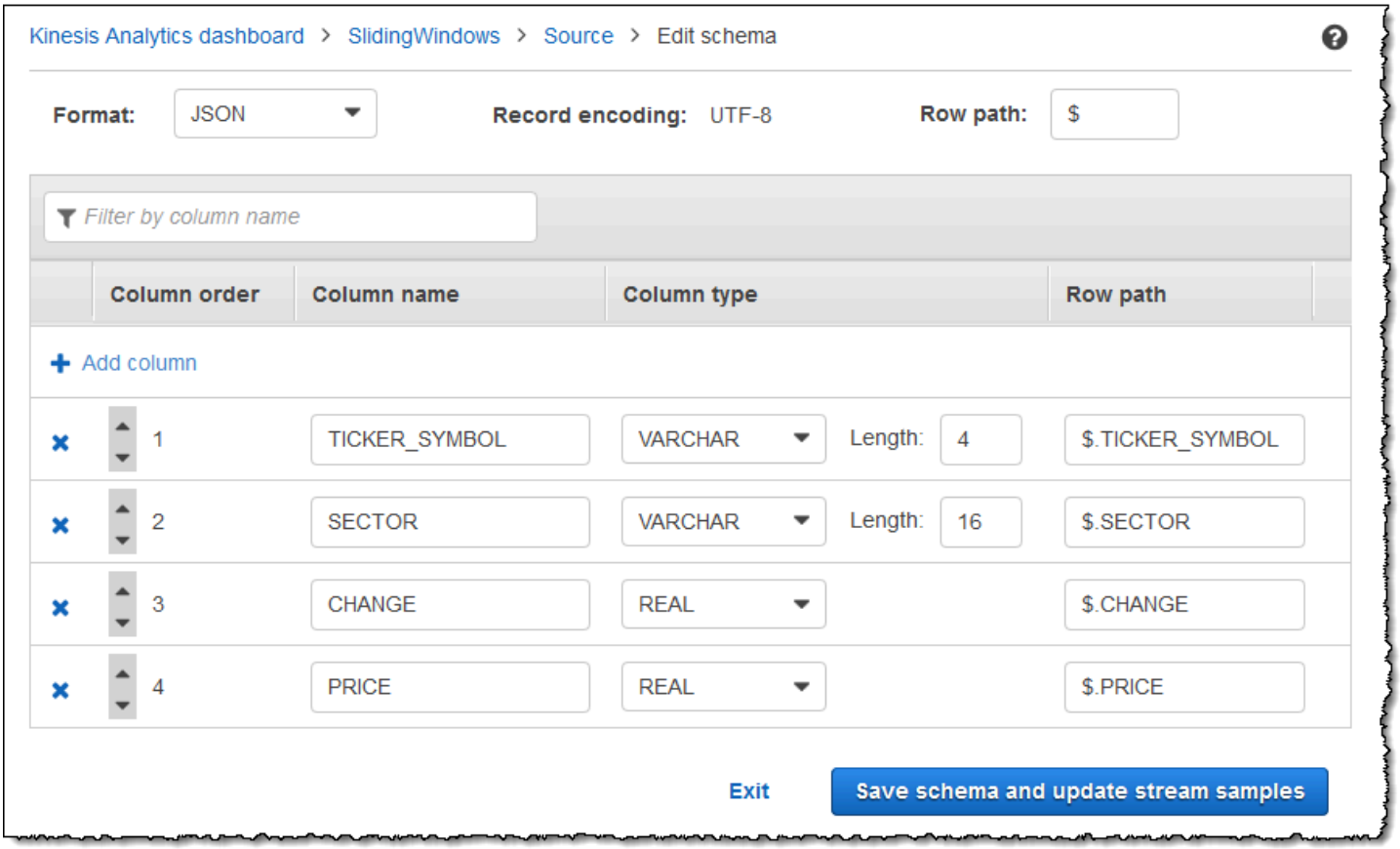
2. 열을 삭제하려면 열 번호 옆에 있는 x 아이콘을 선택합니다.



3. 열 명칭을 다시 지정하려면 열 명칭에서 새 명칭을 입력합니다. 새 열 명칭은 공백일 수 없고, 길이가 한 문자 이상이어야 하며, 예약된 SQL 키워드가 포함되어서는 안 됩니다. 명칭은 SQL 일반 식별자의 명명 기준에 부합해야 합니다. 즉, 문자로 시작해야 하고 문자, 밑줄 및 숫자만 포함해야 합니다.

4. 열의 데이터 유형을 변경하려면 [Column type]에서 새 데이터 유형을 선택합니다. 새 데이터 유형이 `CHAR`, `VARBINARY` 또는 `VARCHAR`인 경우 Length(길이)에 데이터 길이를 지정합니다. 자세한 정보는 데이터 형식을 참조하십시오.
5. [Save schema and update stream]을 선택하여 변경 사항을 저장합니다.

수정된 스키마가 편집기에 표시되는데, 다음과 비슷할 것입니다.



스키마에 행이 많은 경우, [Filter by column name]을 사용하여 행을 필터링할 수 있습니다. 예를 들어, `Price` 열 같이 P로 시작하는 열 명칭을 편집하려면, 열 명칭별 필터링 상자에 P을 입력합니다.

CSV 스키마 편집

다음 단계에 따라 CSV 스키마를 편집할 수 있습니다.

CSV 스키마를 편집하려면

1. 스키마 편집기의 [Row delimiter]에서 수신 데이터 스트림이 사용할 구분 기호를 선택합니다. 이는 스트림에 있는 데이터 레코드 간의 구분 기호입니다(예: 줄 바꿈 문자).

2. [Column delimiter]에서 수신 데이터 스트림이 사용할 구분 기호를 선택합니다. 이는 스트림에 있는 데이터 필드 간의 구분 기호입니다(예: 쉼표).
3. 열을 추가하려면 [Add column]을 선택합니다.

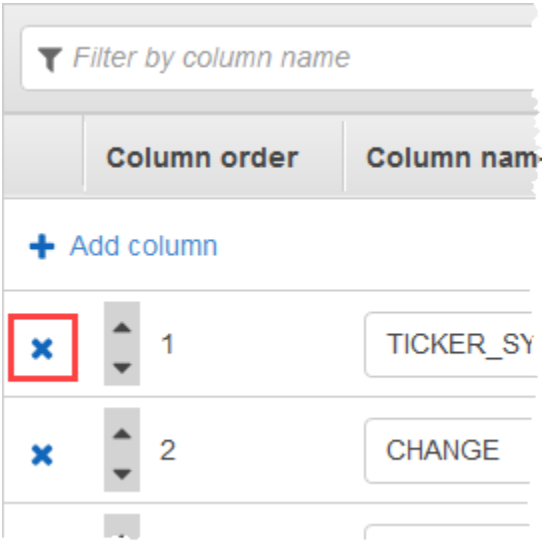
   첫 번째 열 위치에서 새 열이 나타납니다. 열 순서를 변경하려면 열 명칭 옆에 있는 위/아래 화살표를 선택합니다.

   새 열에 대해 다음의 정보를 제공합니다.

   - 열 명칭에 명칭을 입력합니다.

     열 명칭은 공백일 수 없고, 길이가 한 문자 이상이어야 하며, 예약된 SQL 키워드가 포함되어서는 안 됩니다. 명칭은 SQL 일반 식별자의 명명 기준에 부합해야 합니다. 즉, 문자로 시작해야 하고 문자, 밑줄 및 숫자만 포함해야 합니다.
   - 열 유형에서 SQL 데이터 유형을 입력합니다.

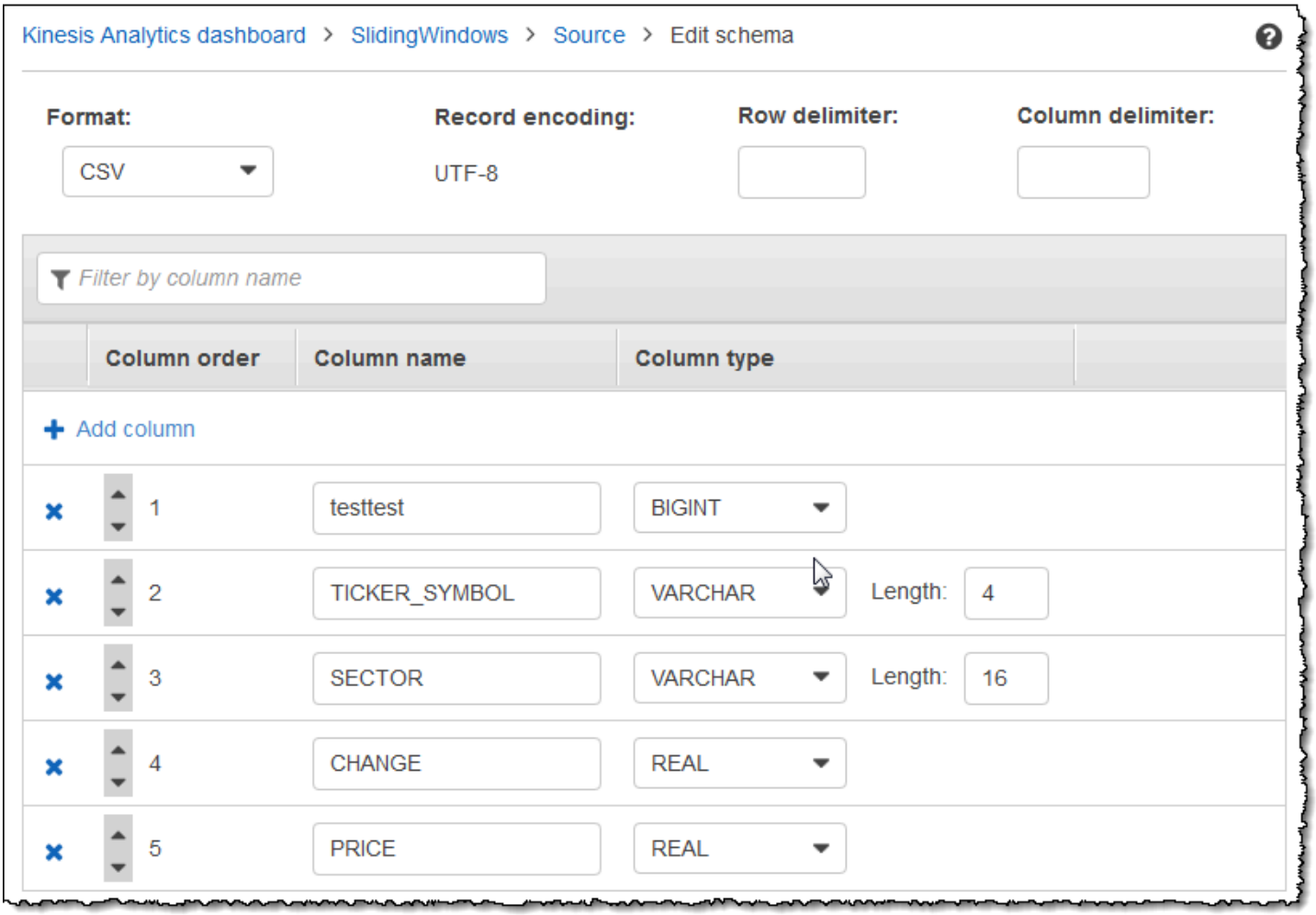
     열 유형은 지원되는 SQL 데이터 유형이면 됩니다. 새 데이터 유형이 CHAR, VARBINARY 또는 VARCHAR인 경우 [Length]에서 데이터 길이를 지정합니다. 자세한 정보는 데이터 형식을 참조하십시오.
4. 열을 삭제하려면 열 번호 옆에 있는 x 아이콘을 선택합니다.



5. 열 명칭을 다시 지정하려면 열 명칭에서 새 명칭을 입력합니다. 새 열 명칭은 공백일 수 없고, 길이가 한 문자 이상이어야 하며, 예약된 SQL 키워드가 포함되어서는 안 됩니다. 명칭은 SQL 일반 식별자의 명명 기준에 부합해야 합니다. 즉, 문자로 시작해야 하고 문자, 밑줄 및 숫자만 포함해야 합니다.

6. 열의 데이터 유형을 변경하려면 [Column type]에서 새 데이터 유형을 선택합니다. 새 데이터 유형이 CHAR, VARBINARY 또는 VARCHAR인 경우 [Length]에서 데이터 길이를 지정합니다. 자세한 정보는 데이터 형식을 참조하십시오.
7. [Save schema and update stream]을 선택하여 변경 사항을 저장합니다.

수정된 스키마가 편집기에 표시되는데, 다음과 비슷할 것입니다.

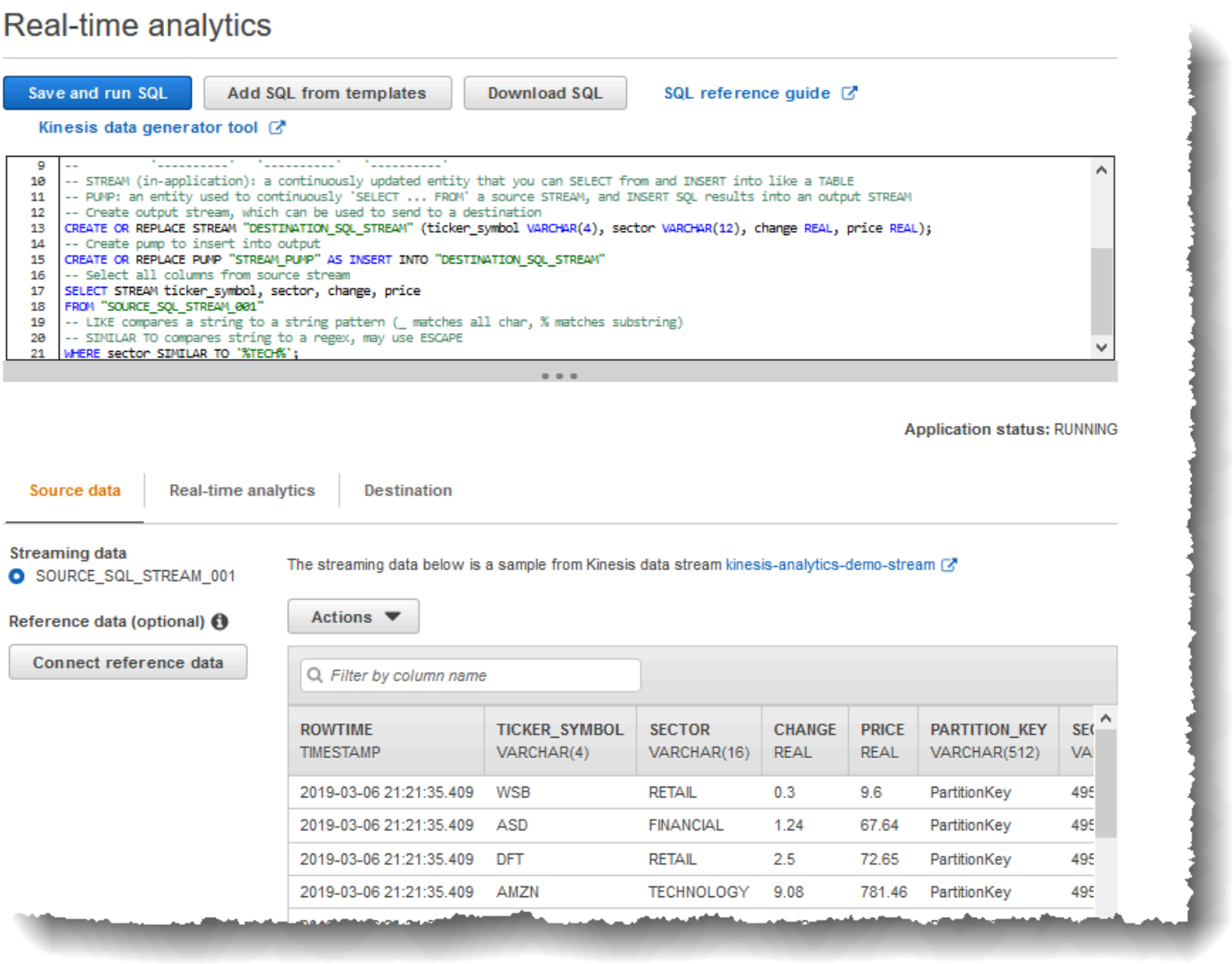


스키마에 행이 많은 경우, [Filter by column name]을 사용하여 행을 필터링할 수 있습니다. 예를 들어, `Price` 열 같이 P로 시작하는 열 명칭을 편집하려면, 열 명칭별 필터링 상자에 P을 입력합니다.

SQL 편집기 작업

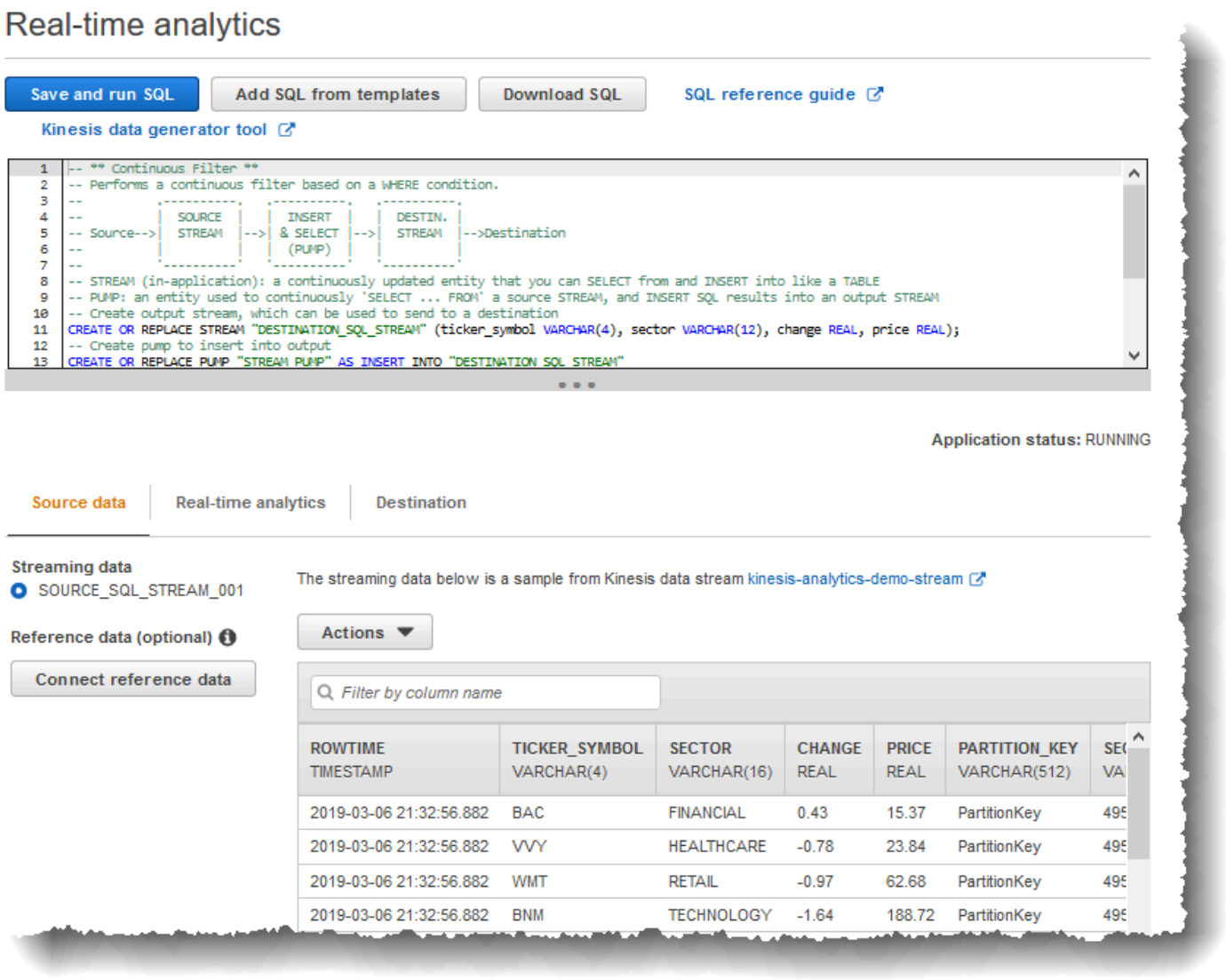
다음에서 SQL 편집기의 각 섹션과 사용 방법에 관한 정보를 확인할 수 있습니다. SQL 편집기에서 코드를 직접 작성하거나 [Add SQL from templates]을 선택할 수 있습니다. 보편적인 Amazon Kinesis Data Analytics 애플리케이션을 작성하는 데 도움이 될 수 있는 예 SQL 코드가 SQL 템플릿에 제시되

어 있습니다. 이 설명서에 있는 예 애플리케이션은 다음 템플릿 중 일부를 사용합니다. 자세한 설명은 Kinesis Data Analytics for SQL 예 섹션을 참조하십시오.



소스 데이터 탭

[Source data] 탭에서 스트리밍 소스를 식별합니다. 또한 이 소스가 매핑되고 애플리케이션 입력 구성을 제공하는 애플리케이션 내 입력 스트림을 식별합니다.



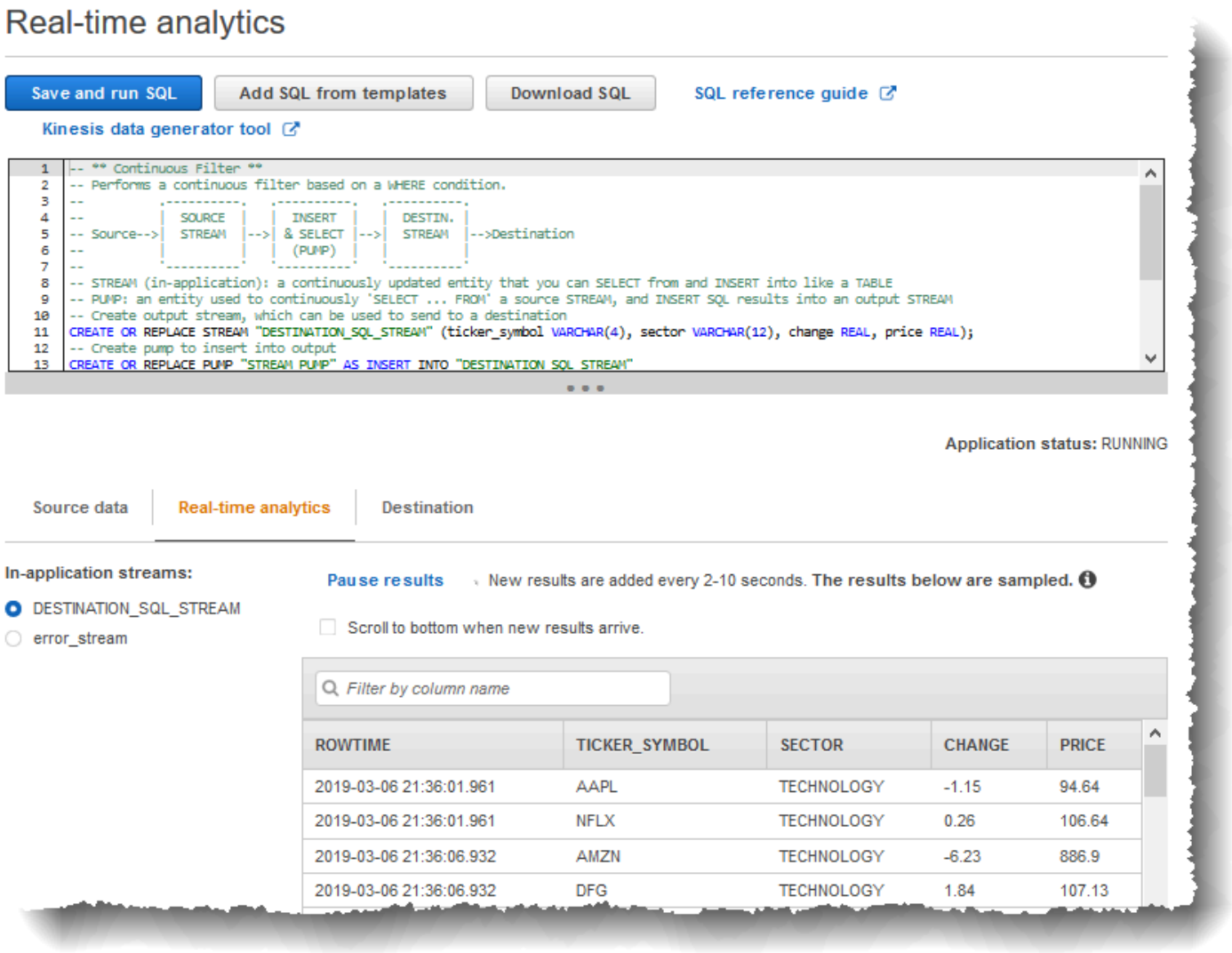
Amazon Kinesis Data Analytics는 입력 구성에 명시적인 매핑을 제공할 필요가 없도록 다음의 타임스탬프 열을 제공합니다:

- ROWTIME – 애플리케이션 내 스트림에 있는 각 열에는 `ROWTIME`라고 하는 특수 열이 있습니다. 이 열은 Kinesis Data Analytics가 첫 번째 애플리케이션 내 스트림에 행을 삽입한 시점의 타임스탬프입니다.
- Approximate_Arrival_Time – 스트리밍 소스에 대한 레코드는 `Approximate_Arrival_Timestamp` 열을 포함합니다. 이것은 스트리밍 소스가 관련 레코드를 성공적으로 수신하여 저장하는 시점을 정하는 대략적인 도착 시간 타임스탬프입니다. Kinesis Data Analytics는 이 열을 `Approximate_Arrival_Time`로서 애플리케이션 내 입력 스트림으로 가져옵니다. Amazon Kinesis Data Analytics는 스트리밍 소스에 매핑된 애플리케이션 내 입력 스트림에서만 이 열을 제공합니다.

이들 타임스탬프 값은 시간 기반 윈도우 모드 쿼리에 유용합니다. 자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.

실시간 분석 탭

[Real-time analytics] 탭에서는 애플리케이션 코드에서 생성한 모든 애플리케이션 내 스트림을 확인할 수 있습니다. 이 스트림 그룹은 모든 애플리케이션에 대해 Amazon Kinesis Data Analytics가 제공하는 오류 스트림(`error_stream`)을 포함합니다.



대상 탭

대상 탭을 사용하면 애플리케이션 내 스트림을 외부 대상에 유지하도록 애플리케이션을 구성할 수 있습니다. 임의의 애플리케이션 내 스트림에 있는 데이터를 외부 대상에 유지하도록 출력을 구성할 수 있습니다. 자세한 설명은 애플리케이션 출력 구성 섹션을 참조하십시오.

스트리밍 SQL 개념

Amazon Kinesis Data Analytics는 확장 기능을 통해 ANSI 2008 SQL 표준을 구현합니다. 확장을 통해 스트리밍 데이터를 처리할 수 있습니다. 다음 주제에서는 핵심적인 스트리밍 SQL 개념을 다룹니다.

주제

- 애플리케이션 내 스트림과 펌프
- 타임스탬프와 ROWTIME 열
- 연속 쿼리
- 윈도우 모드 쿼리
- 스트리밍 데이터 작업: 스트림 조인

애플리케이션 내 스트림과 펌프

애플리케이션 입력을 구성할 때 스트리밍 소스를 생성된 애플리케이션 내 스트림에 매핑합니다. 데이터는 스트리밍 소스에서 애플리케이션 내 스트림으로 연속적으로 흐릅니다. 애플리케이션 내 스트림은 SQL 문을 사용하여 쿼리할 수 있는 표와 같이 작동하지만 연속적인 데이터 흐름을 나타내기 때문에 스트림이라고 합니다.

> **Note**
>
> 애플리케이션 내 스트림을 Amazon Kinesis Data Streams 및 Firehose 전송 스트림과 혼동하지 마세요. 애플리케이션 내 스트림은 Amazon Kinesis Data Analytics 애플리케이션의 컨텍스트에서만 존재합니다. Kinesis 데이터 스트림과 Firehose 전송 스트림은 애플리케이션과 독립적으로 존재합니다. 애플리케이션 입력 구성에서 스트리밍 소스로 구성하거나 출력 구성에서 대상 주소로 구성할 수 있습니다.

필요에 따라 애플리케이션 내 스트림을 추가로 생성하여 중간 쿼리 결과를 저장할 수도 있습니다. 애플리케이션 내 스트림 생성 프로세스는 2개 단계로 구성됩니다. 먼저 애플리케이션 내 스트림을 생성한 다음 펌프를 사용하여 데이터를 스트림에 삽입합니다. 예를 들어, 명칭이 `INPUTSTREAM`인 애플리케이션 내 스트림을 생성하도록 애플리케이션 입력을 구성한다고 가정해 보겠습니다. 다음 예에서 또 다른 스트림(`TEMPSTREAM`)을 생성한 다음 펌프를 사용하여 `INPUTSTREAM`로부터 데이터를 생성된 스트림에 삽입합니다.

1. 다음과 같이 세 열을 지닌 애플리케이션 내 스트림(TEMPSTREAM)을 생성합니다.

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

   열 명칭을 따옴표로 지정하여 대소문자를 구분하도록 합니다. 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조에서 식별자를 참조하십시오.

2. 펌프를 사용하여 데이터를 스트림에 삽입합니다. 펌프는 한 애플리케이션 내 스트림에서 다른 애플리케이션 내 스트림에 데이터를 삽입하는 연속적인 삽입 쿼리입니다. 다음 명령문을 통해 펌프(SAMPLEPUMP)를 생성하고 또 다른 스트림(INPUTSTREAM)에서 레코드를 선택함으로써 데이터를 TEMPSTREAM에 삽입합니다.

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

복수의 작성자가 애플리케이션 내 스트림에 삽입하도록 할 수 있으며, 해당 스트림에서 복수의 구독자가 선택되도록 할 수 있습니다. 애플리케이션 내 스트림을 게시/구독 메시징 패러다임을 구현하는 것으로 생각하십시오. 이 패러다임에서 생성 시간 및 수신 시간을 포함한 데이터 행은 전통적인 RDBMS에 저장할 필요 없이 스트리밍 SQL 문의 캐스케이드에 의해 생성 시간 및 수신 시간을 포함하여 데이터 행이 처리, 해석 및 전달될 수 있는 PUB/SUB 메시징 패러다임의 구현으로 애플리케이션 내 스트림을 생각할 수 있습니다.

애플리케이션 내 스트림이 생성된 후에 정상적인 SQL 쿼리를 수행할 수 있습니다.

> **Note**
>
> 스트림을 쿼리할 때 대부분의 SQL 문은 행 기반 또는 시간 기반 윈도우를 사용하여 구속됩니다. 자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.

스트림을 조인할 수도 있습니다. 스트림 조인에 대한 예는 스트리밍 데이터 작업: 스트림 조인 섹션을 참조하십시오.

타임스탬프와 ROWTIME 열

애플리케이션 내 스트림에는 ROWTIME이라고 하는 특수 열이 있습니다. Amazon Kinesis Data Analytics가 첫 번째 애플리케이션 내 스트림에 행을 삽입하면 타임스탬프를 저장합니다. ROWTIME은 Amazon Kinesis Data Analytics이 스트리밍 소스에서 읽은 후 첫 번째 애플리케이션 내 스트림을 레코드에 삽입한 타임스탬프를 나타냅니다. 그런 다음 이 ROWTIME 값은 애플리케이션 전체에 걸쳐 유지됩니다.

> **Note**
> 한 애플리케이션 내 스트림에서 다른 스트림으로 레코드를 펌핑할 경우, 열을 명시적으로 복사할 필요가 없습니다. Amazon Kinesis Data Analytics가 대신 이 열을 복사합니다.

Amazon Kinesis Data Analytics는 ROWTIME 값의 단조 증가를 보증합니다. 시간 기반 윈도우 모드 쿼리에서 이 타임스탬프를 사용합니다. 자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.

애플리케이션 내 스트림에 있는 임의의 다른 열과 같이 SELECT 문에서 ROWTIME 열에 액세스할 수 있습니다. 예시:

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

스트리밍 분석에서 사용되는 다양한 시간의 이해

ROWTIME 이외에도 실시간 스트리밍 애플리케이션에는 다른 유형의 시간이 사용됩니다. 예를 들면 다음과 같습니다.

- 이벤트 타임 – 이벤트가 발생한 시점의 타임스탬프입니다. 때때로 클라이언트 측 시간이라고도 합니다. 이벤트가 발생한 시간이기 때문에 분석에서 이 시간을 사용하는 것이 바람직한 경우가 많습니다. 그러나 스마트폰 및 웹 클라이언트와 같은 많은 이벤트 소스가 신뢰할 만한 시계를 갖추고 있지 않아 부정확한 시간을 야기할 수 있습니다. 또한 연결 문제로 인해 레코드가 이벤트가 발생한 순서대로 스트림에 나타나지 않을 수 있습니다.

- 인제스트 타임 – 레코드가 스트리밍 소스에 추가된 시점의 타임스탬프입니다. Amazon Kinesis Data Streams에는 이 타임스탬프를 보여주는 `APPROXIMATE_ARRIVAL_TIME`로 불리는 필드가 모든 레코드에 포함되어 있습니다. 이는 서버 측 시간이라고도 합니다. 수집 시간은 이벤트 시간의 근사치인 경구가 많습니다. 레코드가 스트림으로 수집되는 시간이 지체되는 경우 부정확성을 야기할 수 있는데, 이런 경우는 일반적으로 드뭅니다. 수집 시간이 순서에서 벗어나는 경우는 드물지만 스트리밍 데이터의 분산성 때문에 그런 일이 발생할 수 있습니다. 따라서 수집 시간은 이벤트 시간을 반영함에 있어 대부분 정확하고 순서대로입니다.

- 처리 시간 – Amazon Kinesis Data Analytics가 첫 번째 애플리케이션 내 스트림에 행을 삽입할 때의 타임스탬프입니다. Amazon Kinesis Data Analytics는 애플리케이션 내 스트림에 있는 `ROWTIME` 열에 이 타임스탬프를 제공합니다. 처리 시간은 항상 단조 증가합니다. 그러나 애플리케이션이 뒤쳐진다면 정확하지 않을 것입니다. (애플리케이션이 뒤쳐지면 처리 시간이 이벤트 시간을 정확하게 반영하지 못합니다.) 이 `ROWTIME`은 일반 시계에 비해 정확하지만 이벤트가 실제 발생한 시간이 아닐 수 있습니다.

시간 기반 창 모드 쿼리에서 이들 시간 각각을 사용하는 것은 장점과 단점이 있습니다. 이들 시간 중 하나 이상을 선택하고 사용 시나리오를 바탕으로 관련 전략을 채택하는 것이 좋습니다.

> **Note**
>
> 행 기반 윈도우를 사용하는 경우 시간은 문제가 아니며 이 섹션은 무시해도 됩니다.

두 개의 시간 기반 윈도우, 즉 `ROWTIME`과 다른 시간(수집 또는 이벤트 시간) 중 하나를 사용하는 2개 윈도우 전략을 권장합니다.

- 다음 예에서와 같이 쿼리가 결과를 방출하는 빈도를 제어하는 첫 번째 윈도우로 `ROWTIME`을 사용합니다. 논리 시간으로는 사용되지 않습니다.
- 분석과 연관시키고자 하는 논리 시간인 다른 시간 중 하나를 사용하십시오. 이 시간은 이벤트가 발생한 시간을 나타냅니다. 다음 예에서 분석 목적은 레코드를 그룹화하고 티커별 수를 반환하는 것입니다.

이 전략의 장점은 이벤트가 언제 발생했는지 나타내는 시간을 사용할 수 있다는 것입니다. 애플리케이션이 뒤쳐지거나 이벤트가 부적절하게 도착하면 정상적으로 처리할 수 있습니다. 애플리케이션이 애

플리케이션 내 스트림으로 레코드를 가져오는 시점이 뒤쳐지는 경우에도 두 번째 윈도우에서 논리 시간 별로 그룹화할 수 있습니다. 쿼리는 ROWTIME을 사용하여 처리 순서를 보장합니다. 지연되는 레코드도(수집 타임스탬프가 ROWTIME 값에 비해 앞서는 경우) 성공적으로 처리됩니다.

다음 쿼리를 시작하기 실습에서 사용한 데모 스트림과 비교하여 검토합니다. 쿼리는 GROUP BY 절을 사용하여 1분 텀블링 윈도우 내에서 티커 수를 방출합니다.

```
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);
```

GROUP BY에서 먼저 ROWTIME을 바탕으로 1분 윈도우 내에서 레코드를 그룹화한 다음 APPROXIMATE_ARRIVAL_TIME을 기준으로 그룹화합니다.

결과에서의 타임스탬프 값은 가장 가까운 60초 간격으로 내림됩니다. 쿼리에 의해 방출되는 첫 번째 그룹 결과는 첫 1분 간의 레코드를 보여 줍니다. 방출된 결과의 두 번째 그룹은 ROWTIME을 기준으로 그 다음 분 동안의 레코드를 보여 줍니다. 마지막 레코드는 애플리케이션이 애플리케이션 내 스트림에 레코드를 늦게 가져왔음을 나타냅니다(수집 타임스탬프에 비해 ROWTIME 값이 늦는 경우).

```
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.
```

결과를 다운스트림 데이터베이스로 푸시함으로써 최종적으로 정확한 분당 수에 대해 결과를 결합할 수 있습니다. 예를 들어, 결과를 Firehose 전송 스트림에 유지하여 Amazon Redshift 테이블에 작성할 수 있도록 애플리케이션 출력을 구성할 수 있습니다. 결과가 Amazon Redshift 표에 기록되고 나면 이 표를 쿼리하여 `Ticker_Symbol`별로 총 수 그룹을 계산할 수 있습니다. `XYZ`의 경우 레코드가 늦게 도착하더라도 총계는 정확합니다(6+1).

연속 쿼리

스트림에 대한 쿼리는 스트리밍 데이터에 대해 연속적으로 실행됩니다. 이러한 연속 실행은 애플리케이션이 스트림을 연속적으로 쿼리하고 알림을 생성하는 시나리오를 가능하게 합니다.

시작하기 연습에서는 명칭이 `SOURCE_SQL_STREAM_001`인 애플리케이션 내 스트림이 있습니다. 그것은 데모 스트림(Kinesis 데이터 스트림)에서 주식 가격을 지속적으로 받습니다. 스키마는 다음과 같습니다:

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

15%가 넘는 주가 변동에 관심이 있다고 가정해 보겠습니다. 애플리케이션 코드에서 다음 쿼리를 사용할 수 있습니다. 이 쿼리는 연속적으로 실행되며 15%를 초과하는 주가 변동이 감지되는 경우 레코드를 방출합니다.

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

다음 절차를 사용하여 Amazon Kinesis Data Analytics 애플리케이션을 설정하고 이 쿼리를 시험합니다.

쿼리 테스트 방법

1. 시작하기 연습에 따라 애플리케이션을 생성합니다.
2. 애플리케이션 코드에서 SELECT 문을 앞의 SELECT 쿼리로 바꿉니다. 그러면 애플리케이션 코드가 다음과 같을 것입니다.

```
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;
```

윈도우 모드 쿼리

애플리케이션 코드에서 SQL 쿼리는 애플리케이션 내 스트림에 대해 연속적으로 실행됩니다. 애플리케이션 내 스트림은 애플리케이션을 통해 연속적으로 흐르는 무한 데이터를 나타냅니다. 따라서 연속적으로 업데이트되는 이 입력으로부터 결과 세트를 얻으려면 시간 또는 행을 기반으로 정의된 윈도우를 사용하여 쿼리를 자주 바인딩해야 합니다. 이를 윈도우 모드 SQL이라고도 합니다.

시간 기반 윈도우 모드 쿼리의 경우 시간 기반 윈도우 크기를 지정합니다(예: 1분 윈도우). 이를 위해서는 단순 증가하는 타임스탬프 열이 애플리케이션 내 스트림에 있어야 합니다. (새 행의 타임스탬프가 이전 행보다 크거나 같은 경우.) Amazon Kinesis Data Analytics는 각 애플리케이션 내 스트림에 대해 ROWTIME라는 그러한 타임스탬프 열을 제공합니다. 시간 기반 쿼리를 지정할 때 이 열을 사용할 수 있습니다. 애플리케이션에 대해 몇몇 다른 타임스탬프 옵션을 선택할 수 있습니다. 자세한 설명은 타임스탬프와 ROWTIME 열 섹션을 참조하십시오.

행 기반 윈도우 모드 쿼리의 경우 행 수 기준 윈도우 크기를 지정합니다.

애플리케이션의 필요에 따라 텀블링 윈도우, 슬라이딩 윈도우 또는 스태거 윈도우 방식으로 레코드를 처리하도록 쿼리를 지정할 수 있습니다. Kinesis Data Analytics는 다음 창 유형을 지원합니다:

- 스태거 윈도우: 데이터가 도착하면 열리는 키 참조 시간 기반 윈도우를 이용해 데이터를 집계하는 쿼리. 복수의 중복 윈도우를 허용하는 키. 스태거 윈도우는 텀블링 윈도우에 비해 지연되거나 순서가 맞지 않는 데이터가 적으며, 따라서 시간 기반 윈도우를 이용해 데이터를 집계할 때 권장되는 방식입니다.

- 텀블링 윈도우: 정기적으로 열리고 닫히는 개별 시간 기반 윈도우를 이용해 데이터를 집계하는 쿼리.
- 슬라이딩 윈도우: 고정 시간이나 행 개수 간격을 이용해 데이터를 지속적으로 집계하는 쿼리.

스태거 윈도우

스태거 윈도우를 사용하는 것은 일관성 없는 시간에 도착하는 데이터 그룹을 분석하는 데 적합한 윈도우 배치 방법입니다. 이 방법은 관련 판매나 로그 기록 모음 같은 시계열 분석 사용 사례에 적합합니다.

예를 들어, VPC 플로우 로그에는 약 10분의 캡처 윈도우가 있습니다. 그러나 클라이언트에서 데이터를 수집하는 경우 최대 15분의 캡처 윈도우가 있습니다. 스태거 윈도우는 분석을 위해 이러한 로그를 집계할 때 아주 유용한 방법입니다.

스태거 윈도우를 사용하면 텀블링 윈도우를 사용할 때처럼 관련 데이터가 같은 시간 제한 윈도우에 속하지 않는 문제가 발생하지 않습니다.

텀블링 윈도우를 이용할 때의 부분적 결과

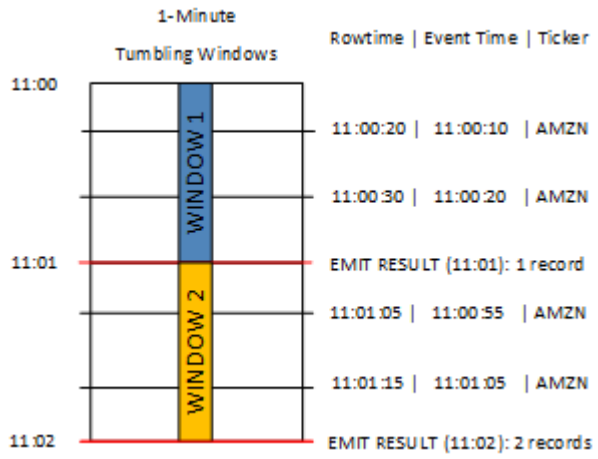
지연되거나 순서가 맞지 않는 데이터에 대해 텀블링 윈도우을(를) 이용할 때는 몇 가지 제한 사항이 있습니다.

텀블링 윈도우를 사용하여 시간 관련 데이터 그룹을 분석하면 개별 레코드가 별도의 윈도우로 나뉠 수 있습니다. 따라서 각 윈도우에서의 부분 결과는 나중에 결합되어 각 레코드 그룹에 대한 완전한 결과를 산출해야 합니다.

다음 텀블링 윈도우 쿼리에서 레코드는 행 시간, 이벤트 시간 및 티커 기호를 기준으로 윈도우로 그룹화됩니다.

```
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);
```

다음 그림에서 애플리케이션은 1분 단위의 거래 발생 시간(이벤트 시간)을 기준으로, 자신이 수신한 거래의 수를 계산합니다. 애플리케이션은 텀블링 윈도우를 이용해 행 시간과 이벤트 시간을 기준으로 데이터를 그룹화할 수 있습니다. 애플리케이션은 레코드 4개를 수신하며 각 레코드는 이전 레코드 수신으로부터 1분 안에 도착합니다. 행 시간, 이벤트 시간 및 티커 기호를 기준으로 레코드를 그룹화합니다. 일부 레코드는 첫 번째 텀블링 윈도우가 끝난 후 도착하며, 따라서 모든 레코드가 같은 1분 텀블링 윈도우에 속하지는 않습니다.



앞의 그림에는 다음과 같은 이벤트가 존재합니다.

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

텀블링 윈도우 애플리케이션의 결과 모음은 다음처럼 표시됩니다.

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

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

앞의 결과 세트에서는 3가지 결과가 반환됩니다.

- 최초 레코드 2개를 집계하는 `ROWTIME`이 11:01:00인 레코드.
- 3번째 레코드만 집계하는 11:02:00 시점의 레코드. 이 레코드는 두 번째 윈도우에는 `ROWTIME`이 있지만 첫 번째 윈도우에는 `EVENT_TIME`이 있습니다.
- 4번째 레코드만 집계하는 11:02:00 시점의 레코드.

전체 결과 세트를 분석하려면, 레코드를 지속성 스토어에서 집계해야 합니다. 이렇게 되면 애플리케이션의 복잡성과 처리 요구사항이 증가합니다.

스태거 윈도우를 이용할 때의 전체 결과

시간 관련 데이터 레코드의 정확성 개선을 위해, Kinesis Data Analytics는 스태거 윈도우라는 새로운 윈도우 유형을 제공합니다. 이 유형의 윈도우에서는 파티션 키와 일치하는 최초 이벤트가 고정된 시간 간격과 무관하게 도착했을 때 윈도우가 열립니다. 윈도우는 윈도우가 열린 시간을 기준으로 측정하는, 지정된 기간을 기준으로 닫힙니다.

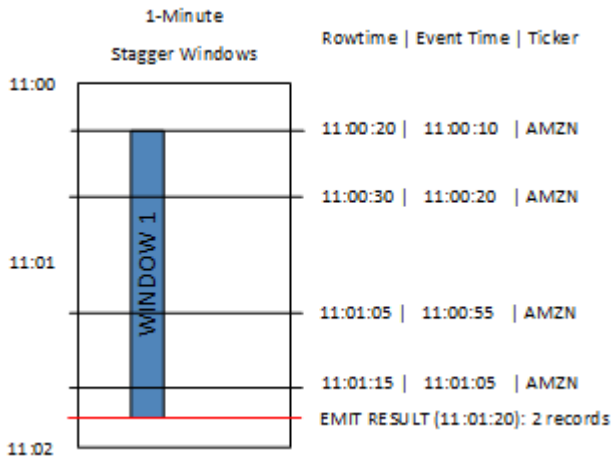
스태거 윈도우는 윈도우 절의 각 키 그룹화에 대한 별도의 시간 제한 윈도우입니다. 애플리케이션은 모든 결과에 단일 윈도우를 이용하는 대신, 자체 시간 윈도우 내 각 윈도우 절의 결과를 집계합니다.

다음 스태거 윈도우 쿼리에서 레코드는 이벤트 시간 및 티커 기호를 기준으로 윈도우로 그룹화됩니다.

```
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);
```

다음 그림에서 이벤트는 이벤트 시간 및 티커 기호를 기준으로 스태거 창으로 집계됩니다.



앞의 그림에는 텀블링 윈도우 애플리케이션이 분석한 것과 같은 이벤트인 다음 이벤트가 존재합니다.

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

스태거 윈도우 애플리케이션의 결과 모음은 다음처럼 표시됩니다.

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

반환된 레코드는 최초 입력 레코드 3개를 집계합니다. 레코드는 1분 스태거 윈도우로 그룹화됩니다. 스태거 윈도우는 애플리케이션이 (ROWTIME이 11:00:20인) 첫 번째 AMZN 레코드를 수신할 때 시작됩니다. 1분 스태거 윈도우가(11:01:20에) 만료되면, (ROWTIME 및 EVENT_TIME을 기준으로 할 때) 결과가 스태거 윈도우에 속하는 레코드가 출력 스트림에 작성됩니다. 스태거 윈도우를 이용하면, ROWTIME과 EVENT_TIME이 1분 윈도우 이내인 모든 레코드가 단일 결과로 출력됩니다.

마지막 레코드(1 분 집합 외부의 EVENT_TIME 포함)는 별도로 집계됩니다. 이는 EVENT_TIME이 레코드를 결과 세트로 분리하는 데 사용되는 파티션 키 중 하나이고 첫 번째 윈도우에 대한 EVENT_TIME의 파티션 키가 11:00이기 때문입니다.

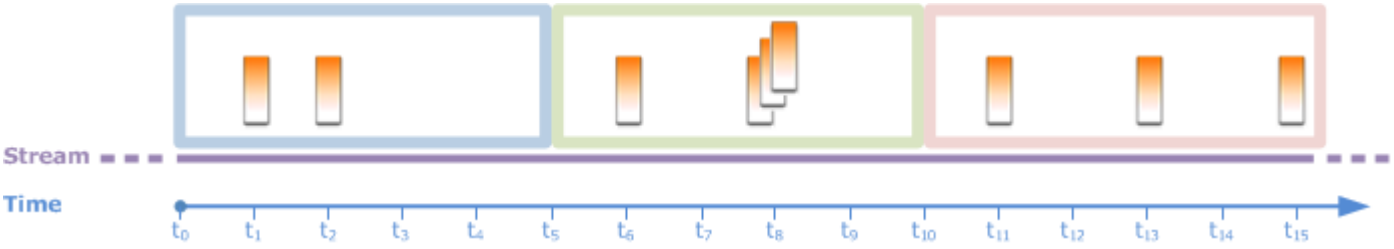
스태거 윈도우의 구문은 특수 절인 WINDOWED BY에서 정의됩니다. 이 절은 스트리밍 집계에서 GROUP BY 대신 사용합니다. 이 절은 선택 사항인 WHERE 절 바로 뒤, HAVING 절 앞에 표시됩니다.

스태거 윈도우는 WINDOWED BY 절에서 정의되며 파티션 키와 윈도우 길이라는 두 가지 파라미터를 취합니다. 파티션 키는 들어오는 데이터 스트림을 분할하며 윈도우가 열리는 시간을 정의합니다. 스태거 윈도우는 고유 파티션 키가 있는 첫 번째 윈도우가 스트림에 표시될 때 열립니다. 스태거 윈도우는 윈도우 길이로 정의하는 고정된 기간이 지나면 닫힙니다. 구문은 다음 코드 예에서 확인할 수 있습니다:

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

텀블링 윈도우(그룹별 집계)

윈도우 모드 쿼리가 비중첩 방식으로 각 윈도우를 처리하는 경우 이 윈도우를 텀블링 윈도우라고 합니다. 이 경우 애플리케이션 내 스트림의 각 레코드는 특정 윈도우에 속합니다. 쿼리가 레코드가 속한 윈도우를 처리할 때 한 번만 처리됩니다.



예를 들어, GROUP BY 절을 사용한 집계 쿼리는 텀블링 윈도우 내에 있는 행을 처리합니다. 시작하기 실습의 데모 스트림은 애플리케이션의 애플리케이션 내 스트림(SOURCE_SQL_STREAM_001)으로 매핑되는 주가 데이터를 수신합니다. 이 스트림의 스키마는 다음과 같습니다.

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

애플리케이션 코드에서 1분 윈도우에 대해 티커 별로 집계(최소, 최대) 가격을 찾고자 한다고 가정해 보겠습니다. 다음 쿼리를 사용할 수 있습니다.

```
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);
```

앞의 예는 시간 기반 윈도우 모드 쿼리의 예입니다. 이 쿼리에서는 ROWTIME 값을 기준으로 레코드를 그룹화합니다. 분 기준 보고의 경우 STEP 함수는 ROWTIME 값을 가장 가까운 분으로 내림합니다.

> **Note**
>
> FLOOR 함수를 사용하여 레코드를 윈도우로 그룹화할 수도 있습니다. 그러나 함수를 사용하여 레코드를 윈도우로 그룹화할 수도 있지만, FLOOR는 시간 값을 전체 시간 단위(시, 분, 초 등)로만 내림할 수 있습니다. STEP은 값을 임의의 간격(예: 30초)으로 내림할 수 있으므로 레코드를 텀블링 윈도우로 그룹화하려는 경우에 권장됩니다.

이 쿼리는 비중첩(텀블링) 윈도우의 예입니다. 1분 윈도우에 있는 GROUP BY 절 그룹 레코드와 각 레코드는 특정 윈도우(비중첩)에 속합니다. 쿼리는 분당 하나의 출력 레코드를 방출하여 특정 분에서 기록되는 최소/최대 티커 가격을 제공합니다. 이런 유형의 쿼리는 입력 데이터 스트림으로부터 주기적 보고서를 생성하는 데 유용합니다. 이 예에서는 1분마다 보고서가 생성됩니다.

쿼리 테스트 방법

1. 시작하기 실습에 따라 애플리케이션을 설정합니다.

2. 애플리케이션 코드에서 SELECT 문을 앞의 SELECT 쿼리로 바꿉니다. 그러면 애플리케이션 코드가 다음과 같을 것입니다.

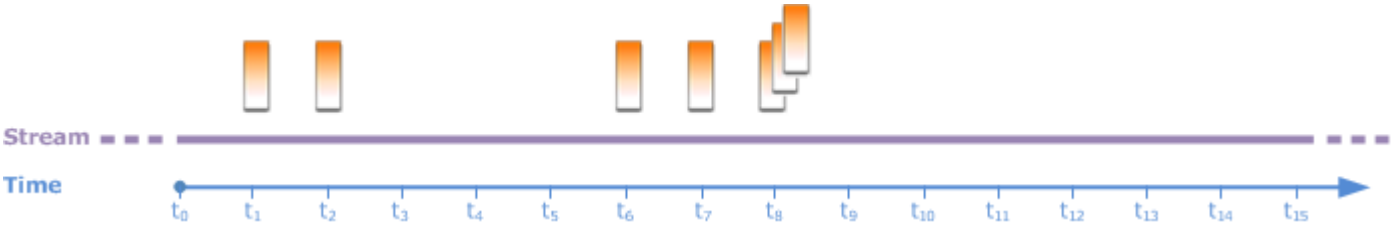
```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                                  ticker_symbol VARCHAR(4),
                                  Min_Price     DOUBLE,
                                  Max_Price     DOUBLE);
-- CREATE OR REPLACE PUMP to insert into output
CREATE OR REPLACE PUMP "STREAM_PUMP"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

슬라이딩 윈도우

GROUP BY를 사용하여 레코드를 그룹화하는 대신 시간 기반 또는 행 기반 윈도우를 정의할 수 있습니다. 명시적 WINDOW 절을 추가함으로써 이를 수행합니다.

이 경우 시간이 지남에 따라 창이 미끄러지면서 스트림에 새 레코드가 표시되면 Amazon Kinesis Data Analytics에서 출력을 내보냅니다. Kinesis Data Analytics는 윈도우 속의 행을 처리하여 이 출력을 내보냅니다. 윈도우는 이러한 유형의 처리에서 중첩될 수 있으며, 레코드는 복수의 윈도우의 일부일 수 있고 각 윈도우에서 처리될 수 있습니다. 다음 예는 슬라이딩 윈도우에 대한 설명입니다.

스트림 상에서 레코드를 계수하는 간단한 쿼리를 생각해 보겠습니다. 이 예에서는 5초 윈도우를 가정합니다. 다음 예 스트림에서 새 레코드가 t_1, t_2, t_6, 및 t_7에 도착하고 세 개의 레코드가 t_8 초에 도착합니다.



다음 사항에 유의하십시오:

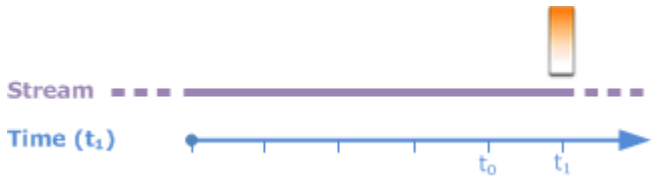
- 이 예에서는 5초 윈도우를 가정합니다. 5초 윈도우가 시간 경과에 따라 연속적으로 이동합니다.

- 윈도우로 들어오는 모든 행에 대해 출력 행이 슬라이딩 윈도우에 의해 방출됩니다. 애플리케이션이 시작되면 바로 5초 윈도우가 지나가지 않은 상태에서도 스트림 상에 나타나는 모든 새 레코드에 대해 쿼리가 출력을 방출합니다. 예를 들어, 레코드가 처음 1초와 그 다음 1초에 나타날 때 쿼리가 출력을 방출합니다. 나중에 쿼리는 5초 윈도우에서 레코드를 처리합니다.
- 윈도우는 시간 경과에 따라 이동합니다. 스트림 상의 오래된 레코드가 윈도우를 벗어나는 경우, 5초 윈도우에 속하는 새 레코드가 스트림에 나타나지 않는 한 쿼리가 출력을 방출하지 않습니다.

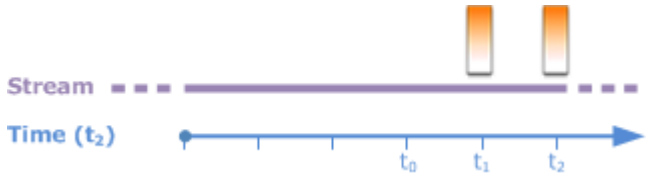
쿼리가 t_0에 실행을 시작한다고 가정해 보겠습니다. 그러면 다음이 발생합니다:

1. 시간 t_0에 쿼리가 시작됩니다. 이 시점에는 레코드가 없기 때문에 쿼리가 출력(개수 값)을 방출하지 않습니다.

2. t_1 시점에 새 레코드가 스트림 상에 나타나고, 쿼리가 수 값 1을 방출합니다.

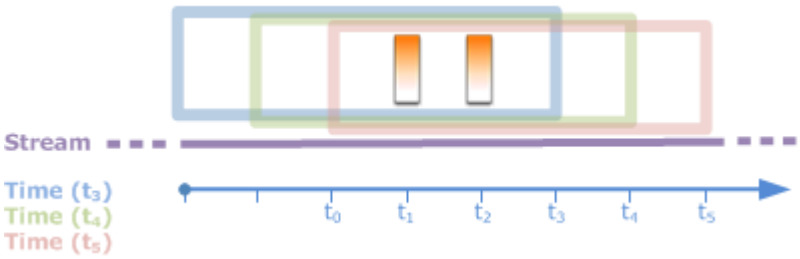


3. t_2 시점에 또 다른 레코드가 나타나고, 쿼리가 수 값 2를 방출합니다.

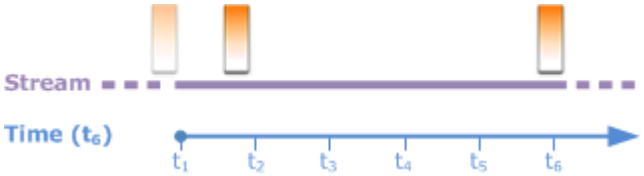


4. 5초 윈도우가 시간 경과에 따라 이동합니다.
   - t_3에서 슬라이딩 윈도우가 t_3에서 t_0으로 이동
   - t4에서 (슬라이딩 윈도우가 t_4에서 t_0으로 이동)
   - t_5에서 슬라이딩 윈도우가 t_5에서 t_0으로 이동

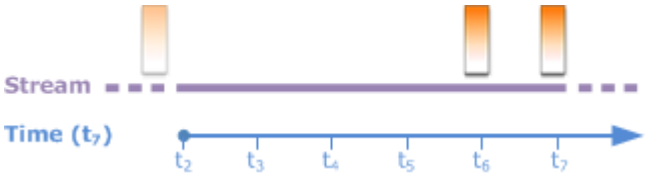
   이 모든 시간에서 5초 윈도우는 동일한 레코드를 지닙니다-새 레코드는 없습니다. 따라서 쿼리는 출력을 방출하지 않습니다.



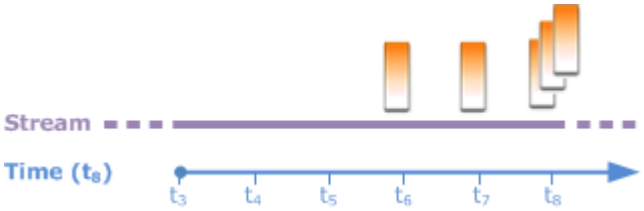
5. 시간 t_6에서 5초 윈도우는 (t_6에서 t_1). 쿼리가 t_6에서 새 레코드 하나를 감지하고 출력 2를 방출합니다. t_1에서 레코드는 더 이상 윈도우에 없으므로 계수되지 않습니다.



6. t_7에서 5초 윈도우는 t_7에서 t_2로 이동합니다. 쿼리가 t_7에서 새 레코드 하나를 감지하고 출력 2를 방출합니다. t_2에서 레코드는 더 이상 5초 윈도우에 없으므로 계수되지 않습니다.



7. t_8에서 5초 윈도우는 t_8에서 t_3로 이동합니다. 쿼리가 새 레코드 3개를 감지하므로 레코드 개수 5를 방출합니다.



요약하자면 윈도우는 크기가 고정이며 시간 경과에 따라 이동합니다. 쿼리는 새 레코드가 나타날 때 출력을 방출합니다.

> **Note**
>
> 1시간 이내의 슬라이딩 윈도우를 사용하는 것이 좋습니다. 더 긴 윈도우를 사용하는 경우 애플리케이션을 정기적인 시스템 유지 관리 이후 다시 시작하는 데 시간이 더 걸립니다. 원본 데이터를 스트림에서 다시 읽어야 하기 때문입니다.

다음은 WINDOW 절을 사용하여 윈도우를 정의하고 집계를 수행하는 예 쿼리입니다. 쿼리가 GROUP BY를 지정하지 않기 때문에 쿼리는 슬라이딩 윈도우 접근 방식을 사용하여 스트림 상에서 레코드를 처리합니다.

예 1: 1분 슬라이딩 윈도우를 사용하여 스트림 처리하기

애플리케이션 내 스트림(SOURCE_SQL_STREAM_001)을 채우는 시작하기 실습에서의 데모 스트림을 검토합니다. 스키마는 다음과 같습니다.

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

애플리케이션이 1분 슬라이딩 윈도우를 사용하여 집계를 연산하도록 하는 것을 가정해 보겠습니다. 즉, 스트림 상에 나타나는 새 레코드 각각에 대해 애플리케이션이 앞선 1분 윈도우의 레코드에 집계를 적용함으로써 출력을 방출하도록 하는 것입니다.

다음의 시간 기반 윈도우 형식 쿼리를 사용할 수 있습니다. 쿼리는 WINDOW 절을 사용하여 1분 범위 간격을 정의합니다. WINDOW의 PARTITION BY 절이 슬라이딩 윈도우 내에 있는 티커 값을 기준으로 레코드를 그룹화합니다.

```
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);
```

쿼리 테스트 방법

1. 시작하기 실습에 따라 애플리케이션을 설정합니다.
2. 애플리케이션 코드에서 SELECT 문을 앞의 SELECT 쿼리로 바꿉니다. 그러면 애플리케이션 코드가 다음과 같을 것입니다.

```
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);
```

예 2: 슬라이딩 윈도에 집계를 적용하는 쿼리

데모 스트림에 대한 다음 쿼리는 10초 윈도우에서의 각 티커의 가격 변동률 평균을 반환합니다.

```
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);
```

쿼리 테스트 방법

1. 시작하기 실습에 따라 애플리케이션을 설정합니다.
2. 애플리케이션 코드에서 SELECT 문을 앞의 SELECT 쿼리로 바꿉니다. 그러면 애플리케이션 코드가 다음과 같을 것입니다.

```
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);
```

예 3: 동일한 스트림 상에서 복수의 슬라이딩 윈도우에서 나오는 데이터의 쿼리

각 열의 값이 동일한 스트림에 대해 정의된 여러 슬라이딩 윈도를 사용하여 계산되는 출력을 방출하도록 쿼리를 작성할 수 있습니다.

다음 예에서는 쿼리가 출력 티커, 가격, a2 및 a10을 방출합니다. 2행 이동 평균이 10행 이동 평균을 교차하는 티커 기호에 대해 출력을 방출합니다. a2 및 a10 열 값은 2행 및 10행 슬라이딩 윈도우로부터 추출됩니다.

```
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);
```

데모 스트림에 대해 이 쿼리를 시험하려면 예시 1에 설명된 테스트 절차에 따릅니다.

스트리밍 데이터 작업: 스트림 조인

애플리케이션에 복수의 애플리케이션 내 스트림을 가질 수 있습니다. JOIN 쿼리를 작성하여 이들 스트림에 도착하는 데이터를 연결할 수 있습니다. 예를 들어 다음과 같은 애플리케이션 내 스트림이 있다고 가정하겠습니다.

- OrderStream – 주식 주문을 수신합니다.

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

- TradeStream – 주문에 따라 주식 거래 결과를 수신합니다.

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

다음은 이들 스트림의 데이터를 연결하는 JOIN 쿼리 예입니다.

예 1: 주문 시점으로부터 1분 내에 거래가 이루어진 주문 보고

이 예에서 쿼리를 통해 OrderStream과 TradeStream을 조인합니다. 그러나 주문 시점에서 1분 이내에 체결된 매매한 원하는 것이기 때문에 쿼리는 TradeStream에 대해 1분 윈도우를 정의합니다. 윈도우 형식 쿼리에 대한 자세한 설명은 슬라이딩 윈도우 섹션을 참조하십시오.

```
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;
```

WINDOW 절을 명시적으로 사용하고 다음과 같이 앞선 쿼리를 작성하는 윈도우를 정의할 수 있습니다.

```
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)
```

이 쿼리를 애플리케이션 코드에 포함시키는 경우 애플리케이션 코드는 연속적으로 실행됩니다. OrderStream에 도착한 모든 레코드에 대해, 주문이 이루어 진 후 1분 윈도우 내에 거래가 있으면 애플리케이션은 출력을 방출합니다.

앞선 쿼리에서의 조인은 TradeStream에 일치하는 레코드가 있는 경우에만 OrderStream에 있는 레코드를 방출하는(또는 그 반대) 내부 조인입니다. 외부 조인을 사용하면 또 다른 흥미로운 시나리오를 만들 수 있습니다. 주식 주문이 이루어진 후 동일한 윈도우 내에 체결된 매매가 없는 주식 주문과 동일

한 윈도우 내에 있지만 다른 주문에 대해 체결된 매매가 보고되도록 원한다고 가정해 보겠습니다. 다음은 외부 조인의 예입니다.

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

Kinesis Data Analytics for SQL 예

이 섹션은 Amazon Kinesis Data Analytics에서 애플리케이션 생성 및 작동 예를 제공합니다. 여기에는 Kinesis Data Analytics 애플리케이션을 생성하고 그 결과를 테스트하는 데 도움이 되는 예 코드와 단계별 지침이 포함되어 있습니다.

이 예를 살펴보기 전에 먼저 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 및 Amazon Kinesis Data Analytics for SQL 애플리케이션 시작하기 섹션을 검토하는 것이 좋습니다.

주제

- 예: 데이터 변환
- 예: 윈도우 및 집계
- 예: 조인
- 예: 기계 학습
- 예: 알림 및 오류
- 예: 솔루션 액셀러레이터

예: 데이터 변환

애플리케이션 코드가 Amazon Kinesis Data Analytics에서 분석을 수행하기 전에 수신 레코드를 재처리해야 하는 경우가 있습니다. 이러한 상황은 다양한 이유로 발생할 수 있는데, 레코드가 지원 레코드 형식에 부합하지 않아 애플리케이션 내 입력 스트림에서 열이 비정규화되는 경우가 한 예입니다.

이 섹션에서는 가용한 문자열 함수를 사용하여 데이터를 정규화하는 방법, 문자열에서 필요한 정보를 추출하는 방법 등에 관한 예를 제공합니다. 유용한 날짜 시간 함수에 대해서도 이 섹션에서 소개합니다.

Lambda를 통한 스트림 전처리

를 사용하여 스트림을 사전 처리하는 방법에 대한 자세한 내용은 섹션을 AWS Lambda참조하세요Lambda 함수를 사용하여 데이터 사전 처리.

주제

- 예: 문자열 값 변환
- 예: DateTime 값 변환

- 예: 복수의 데이터 유형 변환

예: 문자열 값 변환

Amazon Kinesis Data Analytics는 스트리밍 소스 상의 레코드에 대해 JSON 및 CSV와 같은 형식을 지원합니다. 자세한 설명은 RecordFormat 섹션을 참조하십시오. 그런 다음 이들 레코드를 입력 구성에 따라 애플리케이션 내 스트림에 있는 행에 매핑합니다. 자세한 설명은 애플리케이션 입력 구성 섹션을 참조하십시오. 입력 구성에서는 스트리밍 소스에 있는 레코드 필드를 애플리케이션 내 스트림에 있는 열로 매핑하는 방식을 지정합니다.

이 매핑은 스트리밍 소스 상의 레코드가 지원 형식에 따를 때 작동하며, 애플리케이션 내 스트림이 정규화된 데이터로 채워집니다. 그러나 스트리밍 소스상의 데이터가 지원 표준에 부합하지 않는 경우에는 어떻게 되겠습니까? 예를 들어, 스트리밍 소스에 클릭스트림 데이터, IoT 센서 및 애플리케이션 로그와 같은 데이터가 포함되어 있다면 어떻게 되겠습니까?

아래 예를 참조하십시오:

- 스트리밍 소스에 애플리케이션 로그가 포함된 경우 – 애플리케이션 로그는 표준 Apache 로그 형식에 따르고 JSON 형식을 사용하여 스트림에 기록됩니다.

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

  표준 Apache 로그 형식에 관한 자세한 정보는 Apache 웹 사이트의 Log Files를 참조하십시오.

- 스트리밍 소스에 반정형 데이터가 포함된 경우 – 다음 예에 두 가지 레코드가 소개되어 있습니다. `Col_E_Unstructured` 필드 값은 일련의 CSV(쉼표로 분리된 값)입니다. 5개 열 가운데 처음 4개는 문자열 유형의 값이고 마지막 열에는 CSV가 포함됩니다.

  ```
  { "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"}
```

- 스트리밍 소스의 레코드에는 URL이 포함되는데 분석을 위해서는 URL 도메인 명칭의 일부가 필요합니다.

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

그와 같은 경우, 정규화된 데이터를 포함하는 애플리케이션 내 스트림 생성을 위해서는 일반적으로 다음과 같은 2단계의 프로세스가 요구됩니다.

1. 비정형 필드를 생성되는 애플리케이션 내 입력 스트림의 `VARCHAR(N)` 유형 열에 매핑하도록 애플리케이션을 구성합니다.
2. 애플리케이션 코드에서 문자열 함수를 사용하여 이 단일 열을 복수 열로 분할한 다음 행을 또 다른 애플리케이션 내 스트림에 저장합니다. 애플리케이션 코드가 생성하는 이 애플리케이션 내 스트림은 정규화된 데이터를 가지게 됩니다. 그런 다음 애플리케이션 내 스트림에 있는 데이터에 대해 분석을 수행할 수 있습니다.

Amazon Kinesis Data Analytics는 문자열 열 작업을 수행할 수 있도록 다음과 같은 문자열 연산, 표준 SQL 함수, SQL 표준 확장을 제공합니다:

- 문자열 연산자 – `LIKE`와`SIMILAR` 같은 연산자는 문자열 비교에 유용합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 문자열 연산자를 참조하십시오.
- SQL 함수 – 다음 함수는 개별 문자열 조작에 유용합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 문자열 및 검색 함수를 참조하십시오.
  - `CHAR_LENGTH` – 문자열의 길이를 정합니다.
  - `INITCAP` – 입력 문자열에서 공백으로 구분된 각 단어의 첫 번째 문자가 대문자로 변환되고 나머지 문자는 모두 소문자로 변환된 버전을 반환합니다.
  - `LOWER/UPPER` – 문자열을 대문자 또는 소문자로 변환합니다.
  - `OVERLAY` – 첫 번째 문자열(원본 문자열) 인수 중 일부를 두 번째 문자열(대체 문자열) 인수로 대체합니다.
  - `POSITION` – 다른 문자열 내에서 문자열을 검색합니다.
  - `REGEX_REPLACE` – 하위 문자열을 다른 하위 문자열로 대체합니다.

  - SUBSTRING – 특정 위치에서 시작하는 소스 문자열의 일부를 추출합니다.
  - TRIM – 소스 문자열의 시작 또는 끝부분에서 지정된 문자의 인스턴스를 제거합니다.
- SQL 확장 – 로그 및 URI와 같은 비정형 문자열 작업 수행에 유용합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 로그 파싱 함수를 참조하십시오.
  - FAST_REGEX_LOG_PARSER – regex parser와 비슷하게 작동하지만, 보다 빠른 결과를 얻기 위해 몇 가지 경로를 단축합니다. 예를 들어, 빠른 regex parser는 첫 번째 일치에서 중지합니다(지연 시맨틱).
  - FIXED_COLUMN_LOG_PARSE – 고정 너비 필드를 구문 분석하고 지정된 SQL 유형으로 자동으로 변환합니다.
  - REGEX_LOG_PARSE – 기본적인 Java 정규식 패턴을 기준으로 문자열 구문 분석을 수행합니다.
  - SYS_LOG_PARSE – UNIX/Linux 시스템 로그에서 흔히 발견되는 항목을 처리합니다.
  - VARIABLE_COLUMN_LOG_PARSE – 입력 문자열을 구분 기호 문자 또는 문자열에 의해 분리되는 필드로 분할합니다.
  - W3C_LOG_PARSE – Apache 로그의 신속한 포맷을 위해 사용할 수 있습니다.

이들 함수의 예는 다음 주제를 참조하십시오:

주제

- 예: 문자열의 일부분 추출(SUBSTRING 함수)
- 예: Regex(REGEX_REPLACE 함수)를 사용하여 하위 문자열 대체
- 예: 정규식(REGEX_LOG_PARSE 함수) 기반 로그 문자열 구문 분석
- 예: 웹 로그 구문 분석(W3C_LOG_PARSE 함수)
- 예: 문자열을 복수의 필드로 분할(VARIABLE_COLUMN_LOG_PARSE 함수)

예: 문자열의 일부분 추출(SUBSTRING 함수)

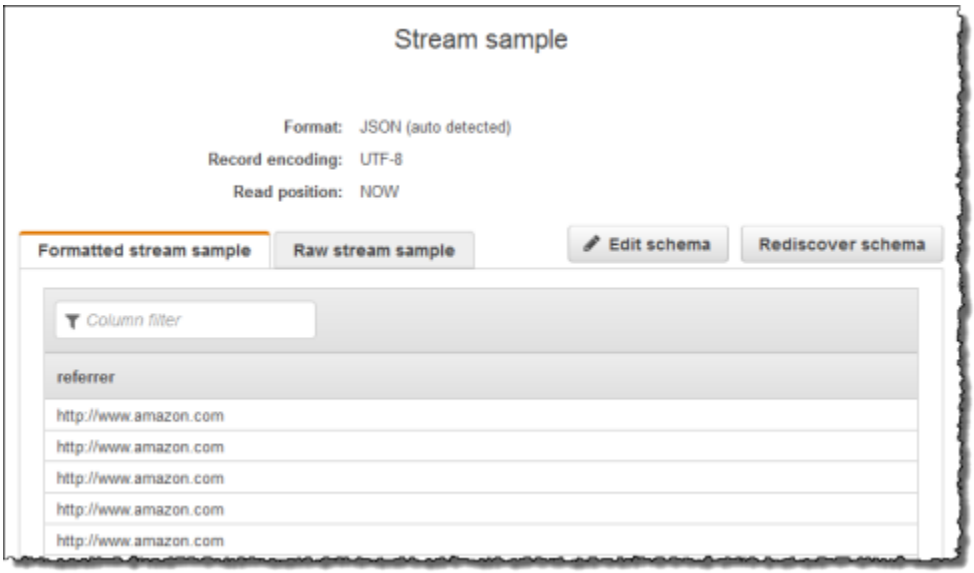
이 예는 SUBSTRING 함수를 사용하여 Amazon Kinesis Data Analytics의 문자열을 변환합니다. SUBSTRING 함수는 특정 위치에서 시작하는 소스 문자열의 일부를 추출합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 SUBSTRING을 참조하십시오.

이 예에서는 다음 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다.

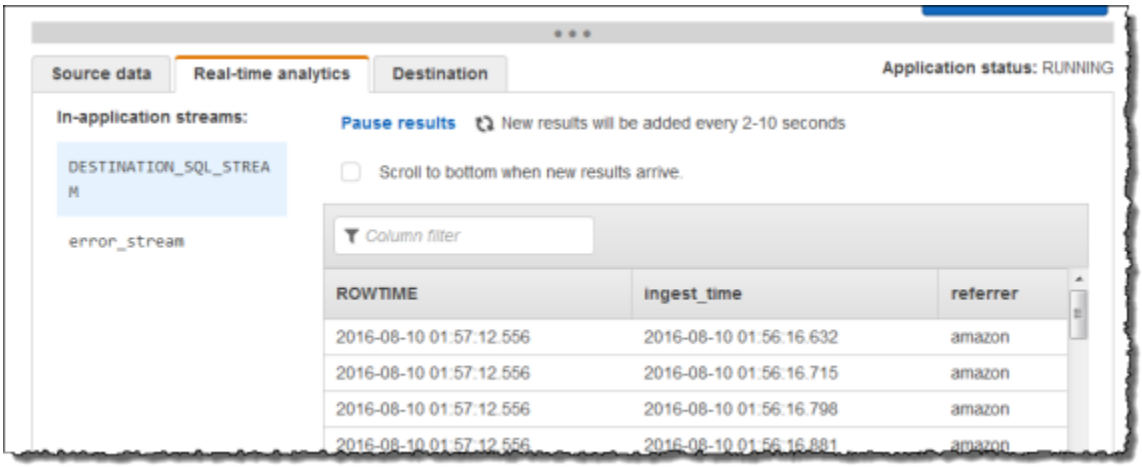
```
{ "REFERRER" : "http://www.amazon.com" }
{ "REFERRER" : "http://www.amazon.com"}
{ "REFERRER" : "http://www.amazon.com"}
```

```
...
```

그런 다음 콘솔에서 Kinesis Data Analytics 애플리케이션을 생성하고 Kinesis 데이터 스트림을 스트리밍 소스로 사용합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 한 열(REFERRER)로 애플리케이션 내 스키마를 유추합니다.



그런 다음 SUBSTRING 함수를 지닌 애플리케이션 코드를 사용하여 URL 문자열 구문 분석을 수행하고 회사 명칭을 검색합니다. 그러면 다음과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.



주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 로그 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 다음의 Python 코드를 실행하여 샘플 로그 레코드를 채웁니다. 이 단순한 코드는 동일한 로그 레코드를 스트림에 연속적으로 씁니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

이후 다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다.

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 스트리밍 데이터 연결을 선택합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 이전 섹션에서 생성한 스트림을 선택합니다.
   b. IAM 역할 생성 옵션을 선택합니다.
   c. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마는 한 열만 지닙니다.
   d. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.
   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
-- 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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

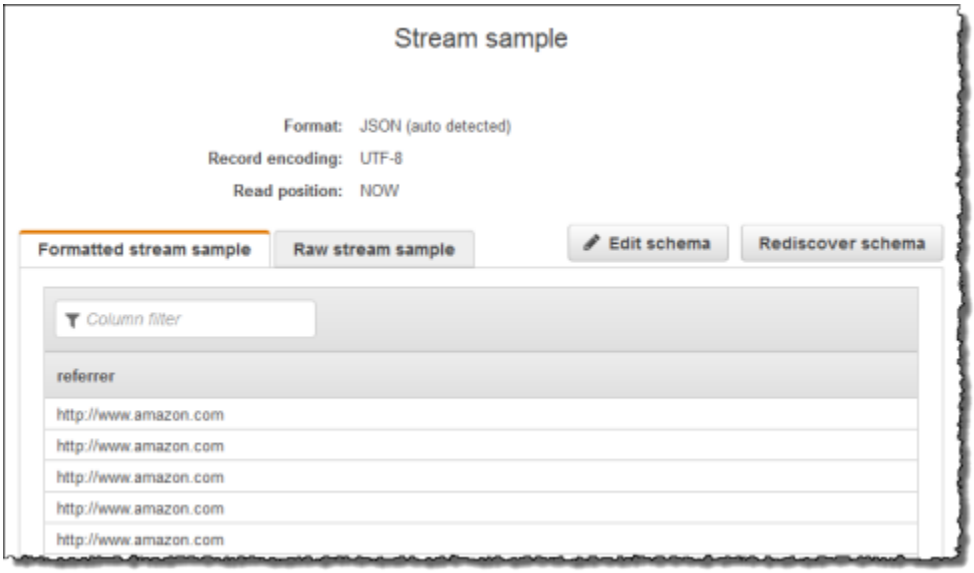
예: Regex(REGEX_REPLACE 함수)를 사용하여 하위 문자열 대체

이 예는 REGEX_REPLACE 함수를 사용하여 Amazon Kinesis Data Analytics의 문자열을 변환합니다. REGEX_REPLACE는 하위 문자열을 대체 하위 문자열로 교체합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 REGEX_REPLACE를 참조하십시오.

이 예에서는 다음 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다:

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

그런 다음 콘솔에서 Kinesis Data Analytics 애플리케이션을 생성하고 Kinesis 데이터 스트림을 스트리밍 소스로 취합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 한 열(REFERRER)로 애플리케이션 내 스키마를 유추합니다.



그런 다음 REGEX_REPLACE 함수를 지닌 애플리케이션 코드를 사용하여 URL을 변환하고 http://대신 https://를 사용합니다. 다음과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.

Filter by column name

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

주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 로그 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 다음의 Python 코드를 실행하여 샘플 로그 레코드를 채웁니다. 이 단순한 코드는 동일한 로그 레코드를 스트림에 연속적으로 씁니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

이후 다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다.

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 스트리밍 데이터 연결을 선택합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 이전 섹션에서 생성한 스트림을 선택합니다.
   b. IAM 역할 생성 옵션을 선택합니다.
   c. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마는 한 열만 지닙니다.
   d. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.
   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
-- 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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: 정규식(REGEX_LOG_PARSE 함수) 기반 로그 문자열 구문 분석

이 예는 REGEX_LOG_PARSE 함수를 사용하여 Amazon Kinesis Data Analytics의 문자열을 변환합니다. REGEX_LOG_PARSE는 기본 Java 정규식 패턴을 기준으로 문자열을 파싱합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 REGEX_LOG_PARSE를 참조하십시오.

이 예에서는 다음 레코드를Amazon Kinesis 스트림에 기록합니다:

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

그런 다음 콘솔에서 Kinesis Data Analytics 애플리케이션을 생성하고 Kinesis 데이터 스트림을 스트리밍 소스로 취합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 한 열(LOGENTRY)로 애플리케이션 내 스키마를 유추합니다.

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"

그런 다음 REGEX_LOG_PARSE 함수를 지닌 애플리케이션 코드를 사용하여 로그 문자열 구문 분석을 수행하고 데이터 요소를 검색합니다. 스크린샷과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.

Filter by column name

ROWTIME	LOGENTRY	MATCH1	MATCH2
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [
2018-05-09 18:16:11.616	203.0.113.24 - - [25/Mar	203.0.113.24 - - [25/Mar	125 "-" "Mozilla/5.0 [

주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 로그 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.

3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 다음의 Python 코드를 실행하여 샘플 로그 레코드를 채웁니다. 이 단순한 코드는 동일한 로그 레코드를 스트림에 연속적으로 씁니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

이후 다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다.

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. [Create application]을 선택하고 애플리케이션 명칭을 지정합니다.

3. 애플리케이션 세부 정보 페이지에서 스트리밍 데이터 연결을 선택합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 이전 섹션에서 생성한 스트림을 선택합니다.
   b. IAM 역할 생성 옵션을 선택합니다.
   c. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마는 한 열만 지닙니다.
   d. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.
   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

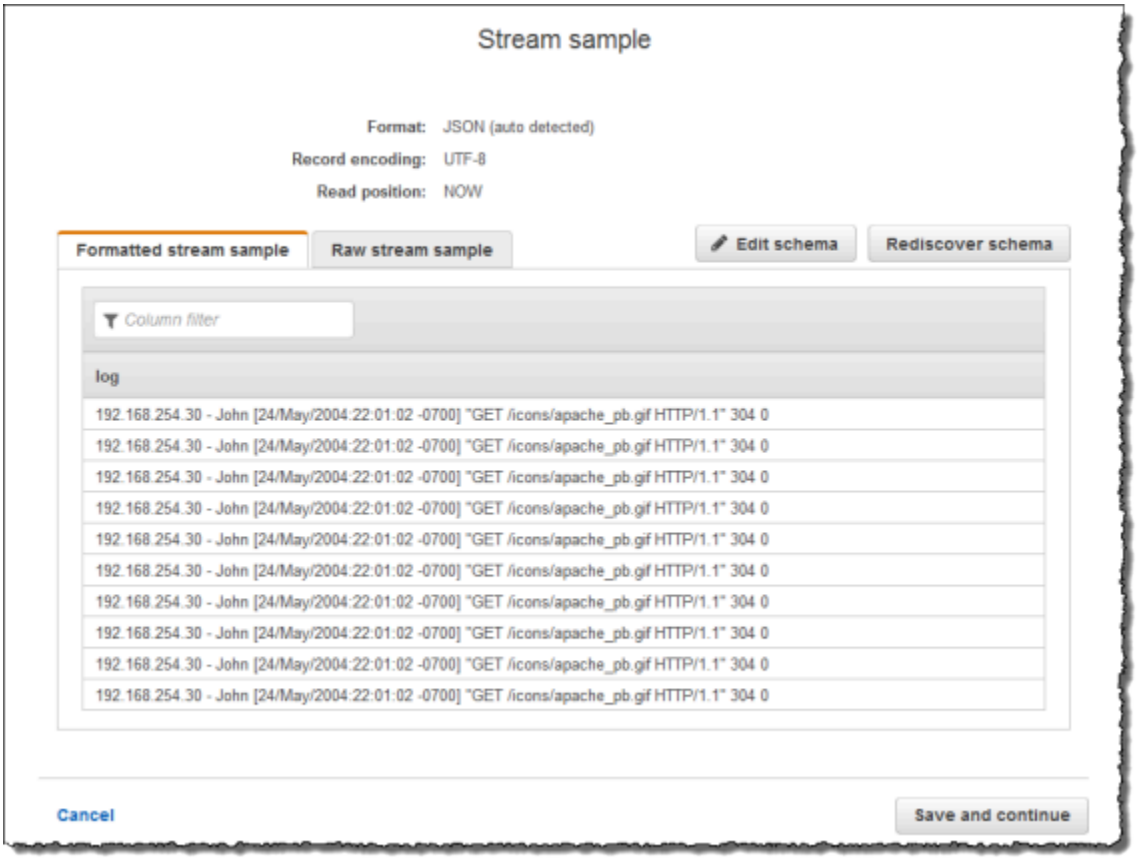
예: 웹 로그 구문 분석(W3C_LOG_PARSE 함수)

이 예는 `W3C_LOG_PARSE` 함수를 사용하여 Amazon Kinesis Data Analytics의 문자열을 변환합니다. `W3C_LOG_PARSE`를 사용하여 Apache 로그의 형식을 빠르게 지정할 수 있습니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 W3C_LOG_PARSE를 참조하십시오.

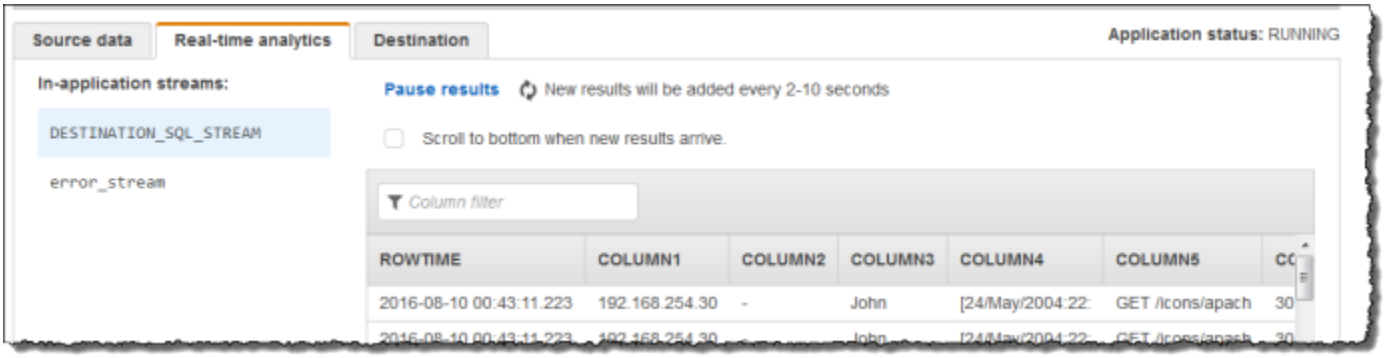
이 예에서는 로그 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다. 다음은 로그의 예입니다:

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

그런 다음 콘솔에서 Kinesis Data Analytics 애플리케이션을 생성하고 Kinesis 데이터 스트림을 스트리밍 소스로 취합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 한 열(로그)로 애플리케이션 내 스키마를 유추합니다.



그런 다음 `W3C_LOG_PARSE` 함수를 지닌 애플리케이션 코드를 사용하여 로그 구문 분석을 수행하고, 다음과 같이 별도의 열에 다양한 로그 필드로 또 다른 애플리케이션 내 스트림을 생성합니다.



주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 로그 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 다음의 Python 코드를 실행하여 샘플 로그 레코드를 채웁니다. 이 단순한 코드는 동일한 로그 레코드를 스트림에 연속적으로 씁니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 스트리밍 데이터 연결을 선택합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 이전 섹션에서 생성한 스트림을 선택합니다.
   b. IAM 역할 생성 옵션을 선택합니다.
   c. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마는 한 열만 지닙니다.
   d. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.
   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: 문자열을 복수의 필드로 분할(VARIABLE_COLUMN_LOG_PARSE 함수)

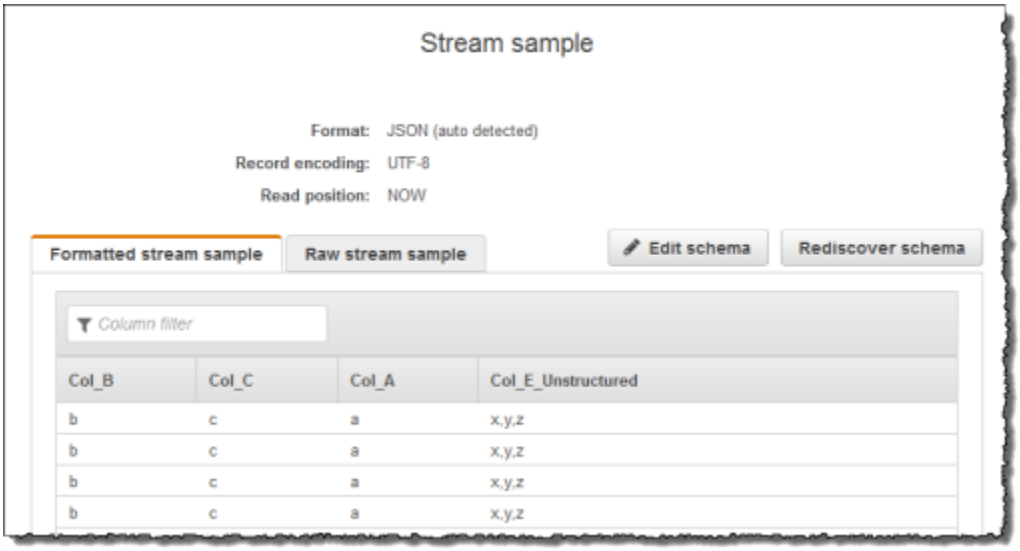
이 예에서는 `VARIABLE_COLUMN_LOG_PARSE` 함수를 이용해 Kinesis Data Analytics의 문자열을 조작합니다. `VARIABLE_COLUMN_LOG_PARSE`는 입력 문자열을 구분 기호 문자 또는 문자열에 의해 분리되는 필드로 분할합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 VARIABLE_COLUMN_LOG_PARSE를 참조하십시오.

이 예에서는 반정형 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다. 예 레코드는 다음과 같습니다:

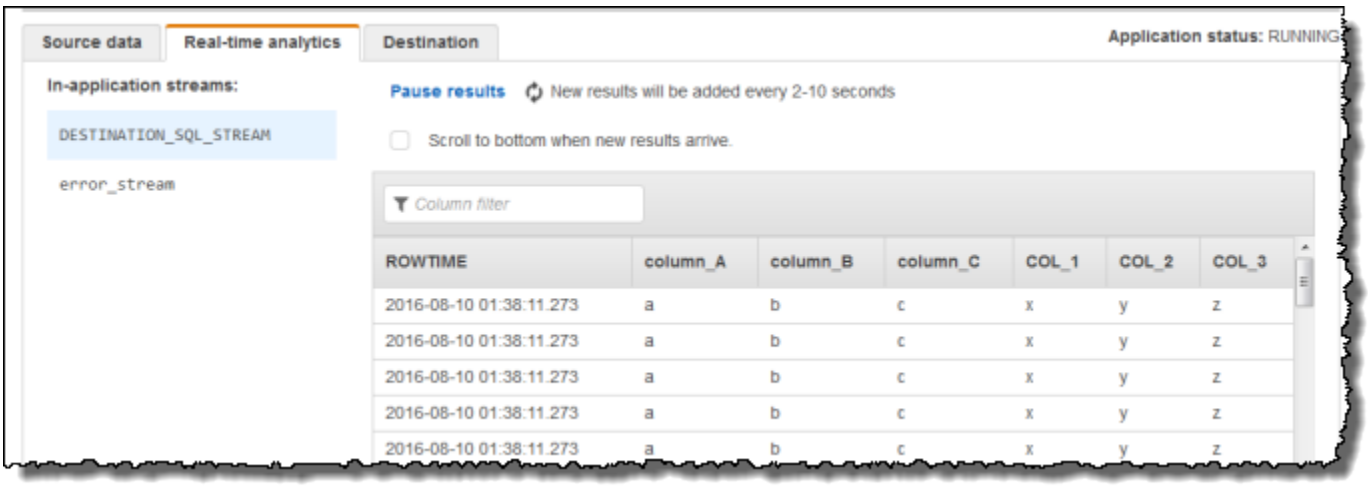
```
{ "Col_A" : "string",
  "Col_B" : "string",
  "Col_C" : "string",
  "Col_D_Unstructured" : "value,value,value,value"}
{ "Col_A" : "string",
  "Col_B" : "string",
```

```
  "Col_C" : "string",
  "Col_D_Unstructured" : "value,value,value,value"}
```

그런 다음 콘솔에서 Kinesis Data Analytics 애플리케이션을 생성하고 Kinesis 스트림을 스트리밍 소스로 사용합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 네 개의 열로 애플리케이션 내 스키마를 유추합니다.



그런 다음 `VARIABLE_COLUMN_LOG_PARSE` 함수를 지닌 애플리케이션 코드를 사용하여 CSV 구문 분석을 수행하고, 다음과 같이 또 다른 애플리케이션 내 스트림에 정규화된 행을 삽입합니다.



주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 로그 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 다음의 Python 코드를 실행하여 샘플 로그 레코드를 채웁니다. 이 단순한 코드는 동일한 로그 레코드를 스트림에 연속적으로 씁니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 스트리밍 데이터 연결을 선택합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 이전 섹션에서 생성한 스트림을 선택합니다.
   b. IAM 역할 생성 옵션을 선택합니다.
   c. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 참고로 유추된 스키마는 한 열만 지닙니다.
   d. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 결과를 확인합니다.
   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다:

```
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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: DateTime 값 변환

Amazon Kinesis Data Analytics는 열을 타임스탬프로 변환하는 것을 지원합니다. 예를 들어 `GROUP BY` 절의 일부인 자체 타임스탬프를 `ROWTIME` 열에 더하여 또 다른 시간 기반 윈도우로 사용할 수 있습니다. Kinesis Data Analytics는 날짜 및 시간 필드 작업을 위한 작업 및 SQL 함수를 제공합니다.

- 날짜 및 시간 연산자 – 날짜, 시간 및 간격 데이터 유형에 대한 산술 연산을 수행할 수 있습니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조의 날짜, 타임스탬프 및 간격 연산자를 참조하십시오.

- SQL 함수 – 여기에는 다음이 포함됩니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조의 날짜 및 시간 함수를 참조하십시오.
  - `EXTRACT()` – 날짜, 시간, 타임스탬프 또는 간격 표현식에서 필드 하나를 추출합니다.
  - `CURRENT_TIME` – 쿼리가 실행될 때의 시간을 반환합니다(UTC).
  - `CURRENT_DATE` – 쿼리가 실행될 때의 날짜를 반환합니다(UTC).
  - `CURRENT_TIMESTAMP` – 쿼리가 실행될 때의 타임스탬프를 반환합니다(UTC).
  - `LOCALTIME` – Kinesis Data Analytics가 실행 중인 환경에서 정의된 대로 쿼리가 실행될 때의 현재 시간(UTC)을 반환합니다.
  - `LOCALTIMESTAMP` – Kinesis Data Analytics가 실행 중인 환경에 의해 정의되는 대로 현재 타임스탬프를 반환합니다(UTC).

- SQL 확장 – 여기에는 다음이 포함됩니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 날짜 및 시간 함수와 날짜 시간 변환 함수를 참조하십시오.
  - `CURRENT_ROW_TIMESTAMP` – 스트림 상의 각 행에 대해 새 타임스탬프를 반환합니다.
  - `TSDIFF` – 두 타임스탬프 간의 차이를 밀리초 단위로 반환합니다.
  - `CHAR_TO_DATE` – 문자열을 날짜로 변환합니다.
  - `CHAR_TO_TIME` – 문자열을 시간으로 변환합니다.

- `CHAR_TO_TIMESTAMP` – 문자열을 타임스탬프로 변환합니다.
- `DATE_TO_CHAR` – 날짜를 문자열로 변환합니다.
- `TIME_TO_CHAR` – 시간을 문자열로 변환합니다.
- `TIMESTAMP_TO_CHAR` – 타임스탬프를 문자열로 변환합니다.

위의 SQL 함수 중 대부분은 형식을 사용하여 열을 변환합니다. 형식은 유연합니다. 예를 들어, 형식 `yyyy-MM-dd hh:mm:ss`를 지정하여 입력 문자열 `2009-09-16 03:15:24`를 타임스탬프로 변환할 수 있습니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조의 Char To Timestamp(Sys)를 참조하십시오.

예: 날짜 변환

이 예에서는 다음 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다.

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

그런 다음 콘솔에서 Kinesis Data Analytics 애플리케이션을 생성하고 Kinesis 스트림을 스트리밍 소스로 취합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 두 개의 열(`EVENT_TIME` 및 `TICKER`)로 애플리케이션 내 스키마를 유추합니다.

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

그런 다음 SQL 함수를 지닌 애플리케이션 코드를 사용하여 EVENT_TIME 타임스탬프 필드를 다양한 방법으로 변환합니다. 그러면 다음 스크린샷과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.

Filter by column name

ROWTIME	TICKER	EVENT_TIME	FIVE_MINUTES_BEFORE	EVE
2018-05-09 21:51:07.244	AAPL	2018-05-09 14:51:06.237	2018-05-09 14:46:06.237	1525
2018-05-09 21:51:07.244	INTC	2018-05-09 14:51:06.326	2018-05-09 14:46:06.326	1525
2018-05-09 21:51:07.244	AAPL	2018-05-09 14:51:06.414	2018-05-09 14:46:06.414	1525
2018-05-09 21:51:07.244	TBV	2018-05-09 14:51:06.503	2018-05-09 14:46:06.503	1525

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 이벤트 시간 및 티커 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림을 생성합니다.
4. 다음의 Python 코드를 실행하여 스트림을 샘플 데이터로 채웁니다. 이 간단한 코드는 지속적으로 임의 티커 기호 및 현재 타임스탬프가 포함된 레코드를 스트림에 씁니다.

```
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"))
```

2단계: Amazon Kinesis Data Analytics 애플리케이션 생성

다음과 같이 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 Connect streaming data(스트리밍 데이터 연결)를 선택하여 소스에 연결합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 이전 섹션에서 생성한 스트림을 선택합니다.
   b. IAM 역할 생성을 선택합니다.
   c. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마에는 두 개의 열이 있습니다.
   d. Edit Schema(스키마 편집)를 선택합니다. EVENT_TIME 열의 열 유형을 `TIMESTAMP`으로 변경합니다.

e. [Save schema and update stream samples]를 선택합니다. 콘솔에서 스키마를 저장한 이후 종료를 선택합니다.

f. [Save and continue]를 선택합니다.

5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.

6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.

   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: 복수의 데이터 유형 변환

추출, 변환 및 로드(ETL) 애플리케이션의 공통적인 요건은 복수의 레코드 유형을 한 스트리밍 소스에서 처리하는 것입니다. Kinesis Data Analytics 애플리케이션을 생성하여 이러한 유형의 스트리밍 소스를 처리할 수 있습니다. 프로세스는 다음과 같습니다.

1. 먼저 스트리밍 소스를 다른 모든 Kinesis Data Analytics 애플리케이션과 유사한 애플리케이션 내 입력 스트림에 매핑합니다.
2. 그런 다음 애플리케이션 코드에서 SQL 문을 작성하여 애플리케이션 내 입력 스트림에서 특정 유형의 열을 가져옵니다. 그런 다음 이를 별도의 애플리케이션 내 스트림에 삽입합니다. (애플리케이션 코드에서 추가 애플리케이션 내 스트림을 생성할 수 있습니다.)

이 실습에서는 두 가지 유형의 레코드(`Order` 및 `Trade` 유형)를 수신하는 스트리밍 소스를 가집니다. 두 가지 유형은 주식 주문과 해당 거래입니다. 각 주문에 대해 거래는 0 이상이 될 수 있습니다. 각 유형의 예 레코드가 다음에 나와 있습니다.

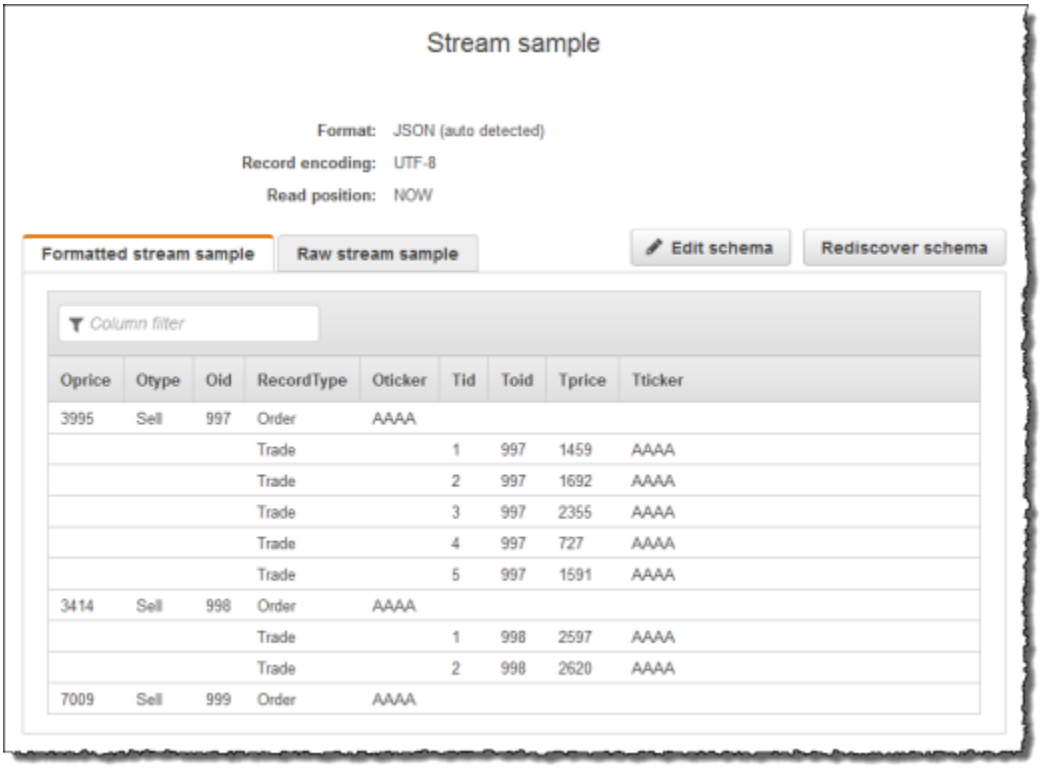
[Order record]

```
{"RecordType": "Order", "Oprice": 9047, "Otype": "Sell", "Oid": 3811, "Oticker":
 "AAAA"}
```

[Trade record]

```
{"RecordType": "Trade", "Tid": 1, "Toid": 3812, "Tprice": 2089, "Tticker": "BBBB"}
```

를 사용하여 애플리케이션을 생성하면 콘솔 AWS Management Console에 생성된 애플리케이션 내 입력 스트림에 대해 다음과 같은 추론된 스키마가 표시됩니다. 기본적으로 콘솔은 이 애플리케이션 내 스트림을 `SOURCE_SQL_STREAM_001`로 명명합니다.



구성을 저장하면 Amazon Kinesis Data Analytics가 스트리밍 소스로부터 데이터를 연속적으로 읽고 애플리케이션 내 스트림에 행을 삽입합니다. 이제 애플리케이션 내 스트림에 있는 데이터에 대해 분석을 수행할 수 있습니다.

이 예의 애플리케이션 코드에서 먼저 두 가지 추가 애플리케이션 내 스트림(`Order_Stream` 및 `Trade_Stream`)을 생성합니다. 그런 다음 레코드 유형에 따라 `SOURCE_SQL_STREAM_001` 스트림으로부터 행을 필터링하고 펌프를 사용하여 새로 생성된 스트림에 삽입합니다. 이 코딩 패턴에 대한 정보는 애플리케이션 코드 섹션을 참조하십시오.

1. 주문 및 거래 행을 별도의 애플리케이션 내 스트림에 필터링합니다.

   a. `SOURCE_SQL_STREAM_001`에 있는 주문 레코드를 필터링하고 주문을 `Order_Stream`에 저장합니다.

```
--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. SOURCE_SQL_STREAM_001에 있는 거래 레코드를 필터링하고 주문을 Trade_Stream에 저장합니다.

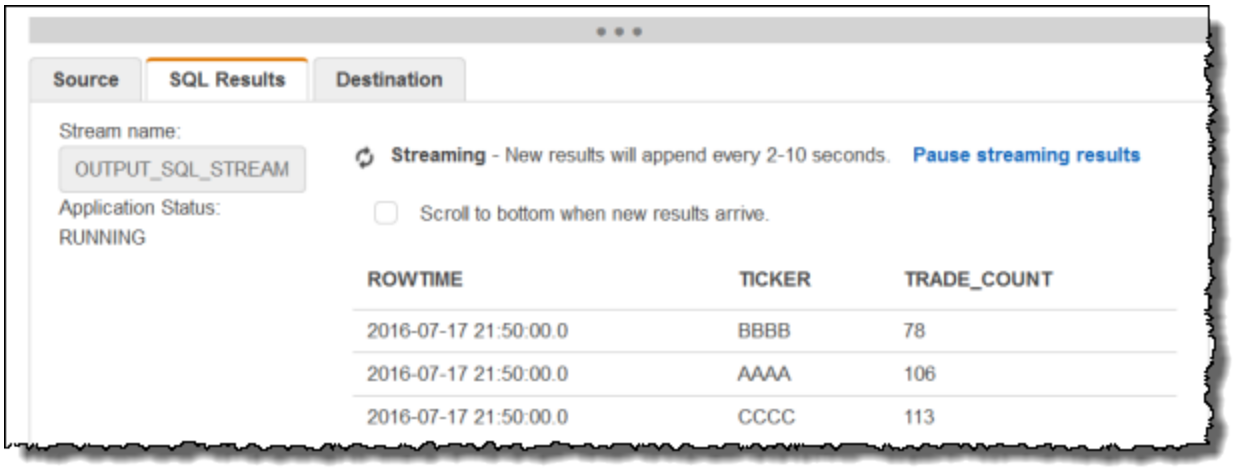
```
--Create Trade_Stream.
CREATE OR REPLACE STREAM "Trade_Stream"
           (trade_id     integer,
            order_id     integer,
            trade_price  DOUBLE,
            ticker       varchar(4),
            record_type  varchar(10)
            );

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. 이제 이들 스트림에 대한 추가 분석을 수행할 수 있습니다. 이 예에서는 1분 시간 범위(텀블링 윈도우)에 대해 티커별 거래 수를 집계하여 결과를 또 다른 스트림(DESTINATION_SQL_STREAM)에 저장합니다.

```
--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);
```

결과는 다음과 같을 것입니다.



주제

- 1단계: 데이터 준비
- 2단계: 애플리케이션 만들기

다음 단계

1단계: 데이터 준비

1단계: 데이터 준비

이 섹션에서는 Kinesis 데이터 스트림을 생성한 다음 주문 및 거래 레코드를 스트림에 채워 넣습니다. 이는 다음 단계에서 생성할 애플리케이션의 스트리밍 소스입니다.

주제

- 1.1단계: 스트리밍 소스 생성
- 1.2단계: 스트리밍 소스 채우기

1.1단계: 스트리밍 소스 생성

콘솔 또는 AWS CLI을 사용하여 Kinesis 데이터 스트림을 생성할 수 있습니다. 이 예에서는 `OrdersAndTradesStream`을 스트림 명칭으로 가정합니다.

- 콘솔 사용 -에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다. [Data Streams]를 선택한 다음 샤드가 하나인 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
- 사용 AWS CLI - 다음 Kinesis `create-stream` AWS CLI 명령을 사용하여 스트림을 생성합니다.

```
$ aws kinesis create-stream \
--stream-name OrdersAndTradesStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

1.2단계: 스트리밍 소스 채우기

다음의 Python 스크립트를 실행하여 샘플 레코드를 `OrdersAndTradesStream`에 채웁니다. 다른 명칭의 스트림을 생성한 경우 Python 코드를 적절히 업데이트합니다.

1. Python 및 `pip`를 설치합니다.

   Python 설치에 관한 정보는 Python 웹사이트를 참조하십시오.

   pip를 사용하여 종속 프로그램을 설치할 수 있습니다. pip 설치에 관한 정보는 pip 웹 사이트에 있는 Installation을 참조하십시오.

2. 다음 Python 코드를 실행합니다. 코드에 있는 `put-record` 명령이 JSON 레코드를 스트림에 작성합니다.

```
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"))
```

다음 단계

[2단계: 애플리케이션 만들기]

2단계: 애플리케이션 만들기

이 섹션에서는 Kinesis Data Analytics 애플리케이션을 생성합니다. 그런 다음 이전 섹션에서 생성한 스트리밍 소스를 애플리케이션 내 입력 스트림에 매핑하는 입력 구성을 추가함으로써 애플리케이션을 업데이트합니다.

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택합니다. 이 예에서는 애플리케이션 명칭 **ProcessMultipleRecordTypes**을(를) 사용합니다.
3. 애플리케이션 세부 정보 페이지에서 Connect streaming data(스트리밍 데이터 연결)를 선택하여 소스에 연결합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.
   a. 1단계: 데이터 준비 단계에서 만든 스트림을 선택합니다.
   b. IAM 역할 생성을 선택합니다.
   c. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다.
   d. [Save and continue]를 선택합니다.
5. 애플리케이션 허브에서 [Go to SQL editor]를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 결과를 확인합니다.
   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
--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. [Save and run SQL]을 선택합니다. Real-time analytics(실시간 분석) 탭을 선택하여 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증합니다.

다음 단계

또 다른 Kinesis 스트림이나 Firehose 데이터 전송 스트림과 같은 외부 대상에 결과를 유지하도록 애플리케이션 출력을 구성할 수 있습니다.

예: 윈도우 및 집계

이 섹션에서는 윈도우 형식 및 집계 쿼리를 사용하는 Amazon Kinesis Data Analytics 애플리케이션 예를 소개합니다. (자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.) 각 예에서는 Kinesis Data Analytics 애플리케이션을 설정하기 위한 단계별 지침과 예 코드를 제공합니다.

주제

- 예: 스태거 윈도우
- 예: ROWTIME을 사용하는 텀블링 윈도우
- 예: 이벤트 타임스탬프를 사용하는 텀블링 윈도우
- 예: 가장 자주 발생하는 값 검색(TOP_K_ITEMS_TUMBLING)
- 예: 쿼리에서 부분적 결과 집계

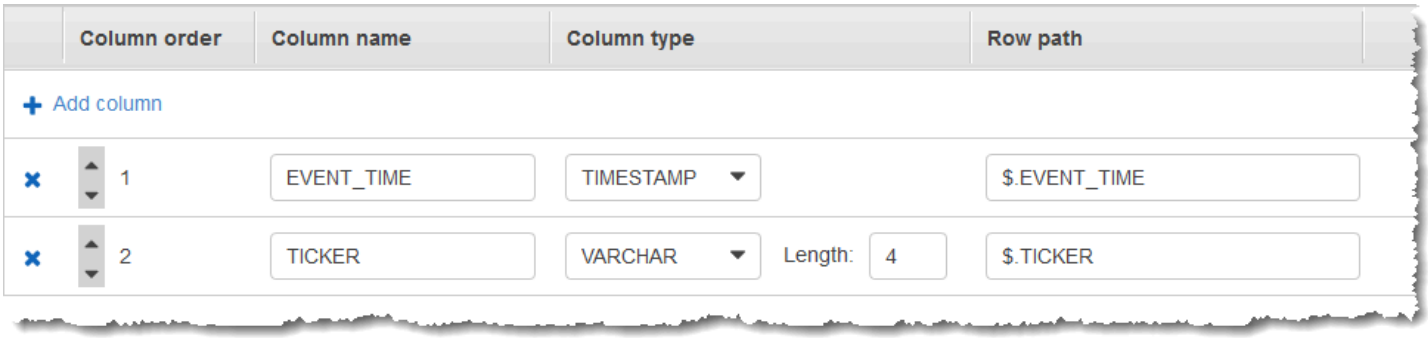
예: 스태거 윈도우

키와 일치하는 데이터가 도착할 때 윈도우 모드 쿼리가 고유 파티션 키에 대한 개별 윈도우를 처리하기 시작한다면, 이 윈도우는 스태거 윈도우라고 부릅니다. 자세한 설명은 스태거 윈도우 섹션을 참조하십시오. 이 Amazon Kinesis Data Analytics 예는 EVENT_TIME 열과 TICKER 열을 사용하여 스태거 윈도우를 생성합니다. 스소 스트림은 1분 이내에 도착하지만 분 값이 다를 수도 있는(예: `18:41:xx`) 같은 EVENT_TIME 및 TICKER 값을 가진 레코드 6개 그룹으로 구성됩니다.

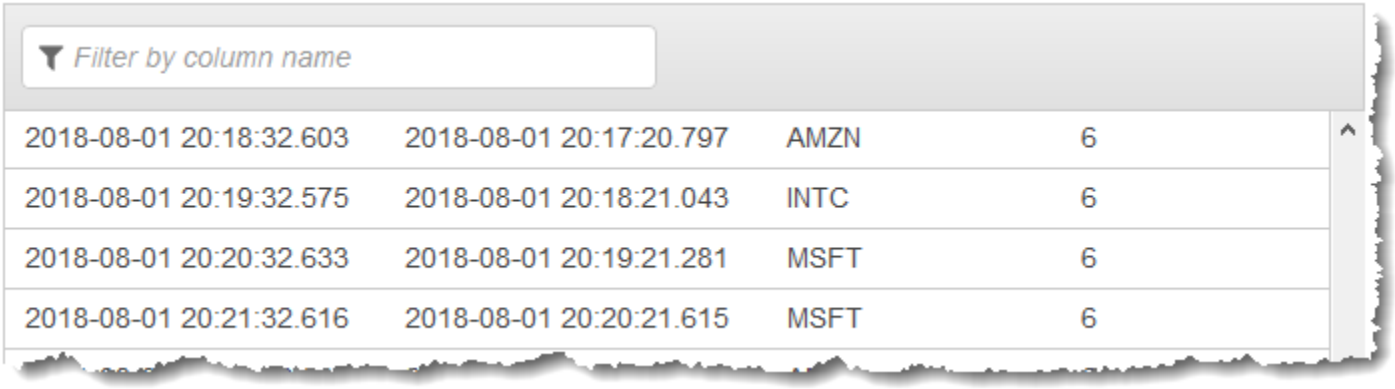
이 예에서는 다음 레코드를 다음 시간에 Kinesis 데이터 스트림에 기록합니다. 이 스크립트는 시간을 스트림에 쓰지 않지만 애플리케이션에서 레코드를 수집하는 시간은 `ROWTIME` 필드에 기록됩니다.

```
{"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
...
```

그런 다음 Kinesis 데이터 스트림을 스트리밍 소스로 AWS Management Console사용하여에서 Kinesis Data Analytics 애플리케이션을 생성합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 두 개의 열(`EVENT_TIME` 및 `TICKER`)로 애플리케이션 내 스키마를 유추합니다.



`COUNT` 함수가 포함된 애플리케이션 코드를 사용하여 데이터의 윈도우 모드 집계를 생성합니다. 그런 다음 아래 스크린샷과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.



다음 절차에서는 EVENT_TIME 및 TICKER를 기반으로 스태거 윈도우의 입력 스트림에서 값을 집계하는 Kinesis Data Analytics 애플리케이션을 생성합니다.

주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.

2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Create Kinesis stream(Kinesis 스트림 생성)을 선택한 다음 샤드가 하나 있는 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 프로덕션 환경에서 Kinesis 데이터 스트림에 레코드를 기록하려면 Kinesis Producer Library 또는 Kinesis Data Streams API를 사용하는 것이 좋습니다. 이 예에서는 간단한 설명을 위해 다음 Python 스크립트를 사용하여 레코드를 생성합니다. 코드를 실행하여 샘플 티커 레코드를 채웁니다. 이 단순 코드는 같은 무작위 `EVENT_TIME`과 티커 기호를 이용하는 레코드 6개 집단을 스트림에 1분 이상 계속 작성합니다. 이후 단계에서 애플리케이션 스키마를 생성할 수 있도록 스크립트를 실행 중 상태로 유지합니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 Connect streaming data(스트리밍 데이터 연결)를 선택하여 소스에 연결합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.

   a. 이전 섹션에서 생성한 스트림을 선택합니다.

   b. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마에는 두 개의 열이 있습니다.

   c. Edit Schema(스키마 편집)를 선택합니다. EVENT_TIME 열의 열 유형을 `TIMESTAMP`으로 변경합니다.

   d. [Save schema and update stream samples]를 선택합니다. 콘솔에서 스키마를 저장한 이후 종료를 선택합니다.

   e. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.

   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
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. [Save and run SQL]을 선택합니다.

   Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: ROWTIME을 사용하는 텀블링 윈도우

윈도우 모드 쿼리가 비중첩 방식으로 각 윈도우를 처리하는 경우 이 윈도우를 텀블링 윈도우라고 합니다. 자세한 설명은 텀블링 윈도우(그룹별 집계) 섹션을 참조하십시오. 이 Amazon Kinesis Data Analytics 예는 `ROWTIME` 열을 사용하여 텀플링 윈도우를 생성합니다. `ROWTIME` 열은 시간을 나타내며 애플리케이션이 레코드를 읽었습니다.

이 예에서는 다음 레코드를 Kinesis 데이터 스트림에 기록합니다.

```
{"TICKER": "TBV", "PRICE": 33.11}
{"TICKER": "INTC", "PRICE": 62.04}
{"TICKER": "MSFT", "PRICE": 40.97}
{"TICKER": "AMZN", "PRICE": 27.9}
...
```

그런 다음 Kinesis 데이터 스트림을 스트리밍 소스로 AWS Management Console사용하여에서 Kinesis Data Analytics 애플리케이션을 생성합니다. 검색 프로세스는 스트리밍 소스 상의 샘플 레코드를 읽고 다음과 같이 두 개의 열(`TICKER` 및 `PRICE`)로 애플리케이션 내 스키마를 유추합니다.

	Column order	Column name	Column type		Row path
+ Add column					
✖	1	TICKER	VARCHAR	Length: 4	$.TICKER
✖	2	PRICE	REAL		$.PRICE

MIN 및 MAX 함수가 포함된 애플리케이션 코드를 사용하여 데이터의 윈도우 모드 집계를 생성합니다. 그런 다음 아래 스크린샷과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.

Filter by column name

ROWTIME	TICKER	MIN_PRICE	MAX_PRICE
2018-06-13 22:16:00.0	AMZN	2.02	99.4
2018-06-13 22:17:00.0	AAPL	1.51	99.79
2018-06-13 22:17:00.0	TBV	0.34	99.88
2018-06-13 22:17:00.0	INTC	0.66	97.72

다음 절차에서는 ROWTIME을 기반으로 텀플링 윈도우의 입력 스트림에서 값을 집계하는 Kinesis Data Analytics 애플리케이션을 생성합니다.

주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.

3. Create Kinesis stream(Kinesis 스트림 생성)을 선택한 다음 샤드가 하나 있는 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 프로덕션 환경에서 Kinesis 데이터 스트림에 레코드를 기록하려면 Kinesis Client Library 또는 Kinesis Data Streams API를 사용하는 것이 좋습니다. 이 예에서는 간단한 설명을 위해 다음 Python 스크립트를 사용하여 레코드를 생성합니다. 코드를 실행하여 샘플 티커 레코드를 채웁니다. 이 단순한 코드는 임의의 티커 레코드를 스트림에 연속적으로 씁니다. 이후 단계에서 애플리케이션 스키마를 생성할 수 있도록 스크립트를 실행 중 상태로 유지합니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 Connect streaming data(스트리밍 데이터 연결)를 선택하여 소스에 연결합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.

   a. 이전 섹션에서 생성한 스트림을 선택합니다.

   b. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마에는 두 개의 열이 있습니다.

   c. [Save schema and update stream samples]를 선택합니다. 콘솔에서 스키마를 저장한 이후 종료를 선택합니다.

   d. [Save and continue]를 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.
6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.

   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
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. [Save and run SQL]을 선택합니다.

      Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

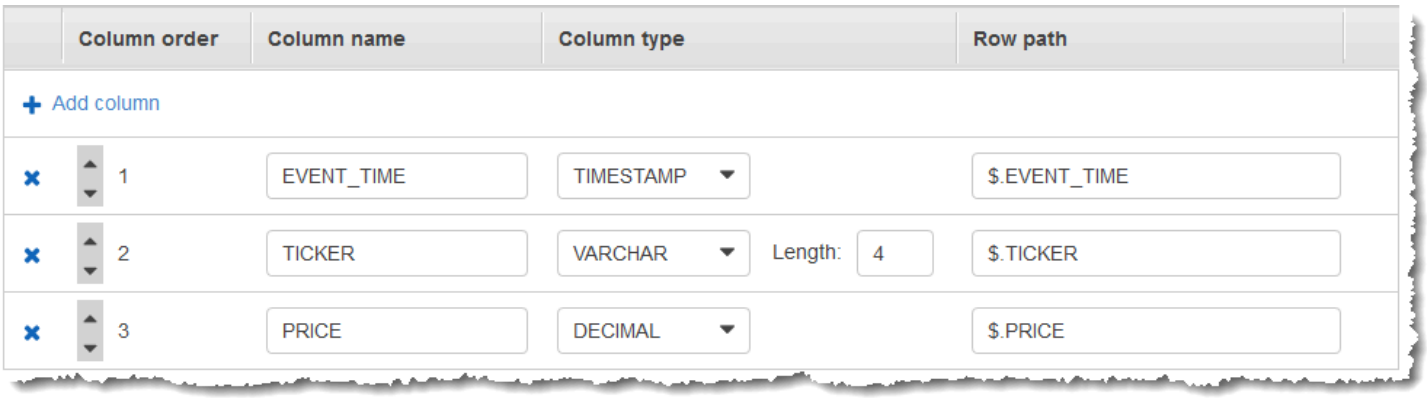
예: 이벤트 타임스탬프를 사용하는 텀블링 윈도우

윈도우 모드 쿼리가 비중첩 방식으로 각 윈도우를 처리하는 경우 이 윈도우를 텀블링 윈도우라고 합니다. 자세한 설명은 텀블링 윈도우(그룹별 집계) 섹션을 참조하십시오. 이 Amazon Kinesis Data Analytics 예에서는 스트리밍 데이터에 포함된 사용자 생성 타임스탬프인 이벤트 타임스탬프를 사용하는 텀블링 윈도우를 보여줍니다. 여기서는 애플리케이션이 레코드를 수신할 때 Kinesis Data Analytics가 생성하는 타임스탬프인 ROWTIME를 사용하는 대신에 이 접근 방식을 사용합니다. 이벤트가 애플리케이션에 수신된 시간 대신에 이벤트가 발생한 시간을 기반으로 집계를 생성하려는 경우 스트리밍 데이터에서 이벤트 타임스탬프를 사용합니다. 이 예에서는 `ROWTIME` 값이 1분마다 집계를 트리거하며 레코드가 `ROWTIME` 및 포함된 이벤트 시간별로 모두 집계됩니다.

이 예에서는 다음 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다. 이벤트가 발생한 시간부터 레코드가 Kinesis Data Analytics에 수집된 시간까지 지연을 생성할 수 있는 처리 및 전송 지연을 시뮬레이션하기 위해 이전에 `EVENT_TIME` 값이 5초로 설정되어 있습니다.

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

그런 다음 Kinesis 데이터 스트림을 스트리밍 소스로 AWS Management Console사용하여에서 Kinesis Data Analytics 애플리케이션을 생성합니다. 검색 프로세스는 스트리밍 소스에서 샘플 레코드를 읽고 다음과 같이 세 개의 열(`EVENT_TIME`, `TICKER` 및 `PRICE`)로 애플리케이션 내 스키마를 유추합니다.



`MIN` 및 `MAX` 함수가 포함된 애플리케이션 코드를 사용하여 데이터의 윈도우 모드 집계를 생성합니다. 그런 다음 아래 스크린샷과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.

Filter by column name

ROWTIME	EVENT_TIME	TICKER	MIN_PRICE	MAX_PRICE
2018-06-18 21:49:00.0	2018-06-18 21:48:00.0	MSFT	8.67	97.91
2018-06-18 21:50:00.0	2018-06-18 21:48:00.0	INTC	3.67	84.7
2018-06-18 21:50:00.0	2018-06-18 21:48:00.0	AAPL	2.39	91.35
2018-06-18 21:50:00.0	2018-06-18 21:48:00.0	AMZN	7.52	93.71

다음 절차에서는 이벤트 시간을 기반으로 텀플링 윈도우의 입력 스트림에서 값을 집계하는 Kinesis Data Analytics 애플리케이션을 생성합니다.

주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Create Kinesis stream(Kinesis 스트림 생성)을 선택한 다음 샤드가 하나 있는 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 프로덕션 환경에서 Kinesis 데이터 스트림에 레코드를 기록하려면 Kinesis Client Library 또는 Kinesis Data Streams API를 사용하는 것이 좋습니다. 이 예에서는 간단한 설명을 위해 다음 Python 스크립트를 사용하여 레코드를 생성합니다. 코드를 실행하여 샘플 티커 레코드를 채웁니다. 이 단순한 코드는 임의의 티커 레코드를 스트림에 연속적으로 씁니다. 이후 단계에서 애플리케이션 스키마를 생성할 수 있도록 스크립트를 실행 중 상태로 유지합니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 Connect streaming data(스트리밍 데이터 연결)를 선택하여 소스에 연결합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.

   a. 이전 섹션에서 생성한 스트림을 선택합니다.

b. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마에는 세 개의 열이 있습니다.

c. Edit Schema(스키마 편집)를 선택합니다. EVENT_TIME 열의 열 유형을 `TIMESTAMP`으로 변경합니다.

d. [Save schema and update stream samples]를 선택합니다. 콘솔에서 스키마를 저장한 이후 종료를 선택합니다.

e. [Save and continue]를 선택합니다.

5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.

6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.

   a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다.

```
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. [Save and run SQL]을 선택합니다.

   Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: 가장 자주 발생하는 값 검색(TOP_K_ITEMS_TUMBLING)

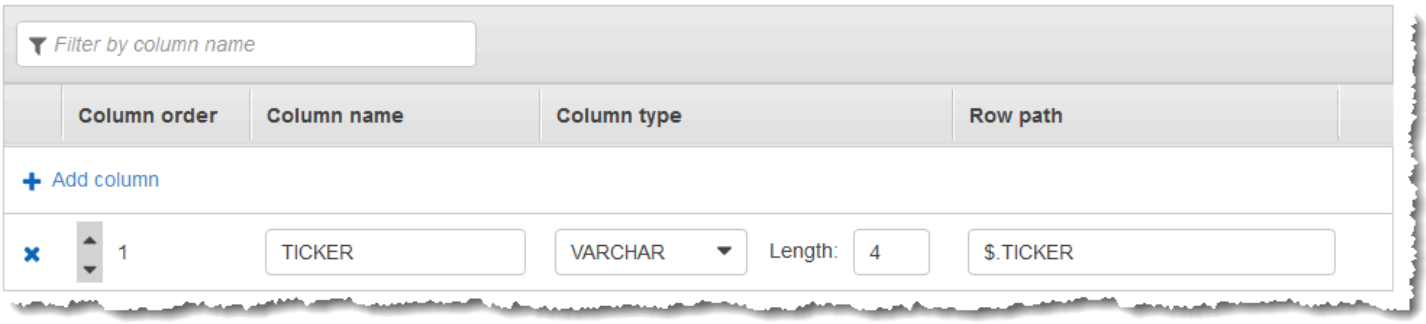
이 Amazon Kinesis Data Analytics 예에서는 `TOP_K_ITEMS_TUMBLING` 함수를 사용하여 텀블링 윈도우에서 가장 자주 발생하는 값을 검색하는 방법을 보여줍니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 `TOP_K_ITEMS_TUMBLING` 함수를 참조하십시오.

`TOP_K_ITEMS_TUMBLING` 함수는 수만 또는 수십만 개 이상의 키를 집계할 때 리소스 사용을 줄이려는 경우에 유용합니다. 이 함수는 `GROUP BY` 및 `ORDER BY` 절을 사용하여 집계하는 것과 동일한 결과를 생성합니다.

이 예에서는 다음 레코드를 Amazon Kinesis 데이터 스트림에 기록합니다:

```
{"TICKER": "TBV"}
{"TICKER": "INTC"}
{"TICKER": "MSFT"}
{"TICKER": "AMZN"}
...
```

그런 다음 Kinesis 데이터 스트림을 스트리밍 소스로 AWS Management Console사용하여에서 Kinesis Data Analytics 애플리케이션을 생성합니다. 검색 프로세스는 스트리밍 소스에서 샘플 레코드를 읽고 다음과 같이 하나의 열(`TICKER`)을 사용하여 애플리케이션 내 스키마를 유추합니다.



`TOP_K_VALUES_TUMBLING` 함수가 포함된 애플리케이션 코드를 사용하여 데이터의 윈도우 모드 집계를 생성합니다. 그런 다음 아래 스크린샷과 같이 다른 애플리케이션 내 스트림에 결과 데이터를 삽입합니다.

Filter by column name

ROWTIME	TICKER	MOST_FREQUENT_VALUES
2018-06-18 22:11:30.796	MSFT	158
2018-06-18 22:12:30.796	INTC	156
2018-06-18 22:12:30.796	AAPL	150
2018-06-18 22:12:30.796	AMZN	129

다음 절차에서는 입력 스트림에서 가장 자주 발생하는 값을 검색하는 Kinesis Data Analytics 애플리케이션을 생성합니다.

주제

- 1단계: Kinesis 데이터 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성

1단계: Kinesis 데이터 스트림 생성

Amazon Kinesis 데이터 스트림을 생성하고 다음과 같이 레코드를 채웁니다:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.
3. Create Kinesis stream(Kinesis 스트림 생성)을 선택한 다음 샤드가 하나 있는 스트림을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
4. 프로덕션 환경에서 Kinesis 데이터 스트림에 레코드를 기록하려면 Kinesis Client Library 또는 Kinesis Data Streams API를 사용하는 것이 좋습니다. 이 예에서는 간단한 설명을 위해 다음 Python 스크립트를 사용하여 레코드를 생성합니다. 코드를 실행하여 샘플 티커 레코드를 채웁니다. 이 단순한 코드는 임의의 티커 레코드를 스트림에 연속적으로 씁니다. 이후 단계에서 애플리케이션 스키마를 생성할 수 있도록 스크립트를 실행 중 상태로 둡니다.

```
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"))
```

2단계: Kinesis Data Analytics 애플리케이션 생성

다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

1. https://console.aws.amazon.com/kinesisanalytics에서 Managed Service for Apache Flink 콘솔을 엽니다.
2. 애플리케이션 생성을 선택하고 애플리케이션 명칭을 입력한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 세부 정보 페이지에서 Connect streaming data(스트리밍 데이터 연결)를 선택하여 소스에 연결합니다.
4. Connect to source(소스에 연결) 페이지에서 다음을 수행합니다.

   a. 이전 섹션에서 생성한 스트림을 선택합니다.

b. Discover schema(스키마 발견)를 선택합니다. 유추된 스키마와, 생성된 애플리케이션 내 스트림에 대한 스키마를 유추하는 데 사용된 샘플 레코드를 콘솔이 표시할 때까지 기다립니다. 유추된 스키마에는 열이 한 개 있습니다.

c. [Save schema and update stream samples]를 선택합니다. 콘솔에서 스키마를 저장한 이후 종료를 선택합니다.

d. [Save and continue]를 선택합니다.

5. 애플리케이션 세부 정보 페이지에서 Go to SQL editor(SQL 편집기로 이동)를 선택합니다. 애플리케이션을 시작하려면 나타나는 대화 상자에서 Yes, start application(예, 애플리케이션 시작)을 선택합니다.

6. SQL 편집기에서 애플리케이션 코드를 작성하고 다음과 같이 결과를 확인합니다.

a. 다음 애플리케이션 코드를 복사하여 편집기에 붙여넣습니다:

```
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. [Save and run SQL]을 선택합니다.

Real-time analytics(실시간 분석) 탭에서 애플리케이션이 생성한 모든 애플리케이션 내 스트림을 확인하고 데이터를 검증할 수 있습니다.

예: 쿼리에서 부분적 결과 집계

Amazon Kinesis 데이터 스트림에 처리 시간이 정확히 일치하지 않는 이벤트 시간이 있는 레코드가 포함되어 있는 경우, 텀블링 윈도의 결과 선택에는 창에 도착했다고 되어 잇지만 실제로는 그렇지 아니한

레코드가 포함됩니다. 이 경우 텀블링 윈도우에는 원하는 결과의 일부분만 포함됩니다. 이 문제를 해결하려면 다음과 같은 여러 가지 접근 방식을 사용할 수 있습니다.

- 텀블링 윈도우만 사용하고, upsert를 사용하여 데이터베이스 또는 데이터 웨어하우스를 통한 사후 처리에서 부분적 결과를 집계합니다. 이 접근 방법은 애플리케이션을 처리하는 데 효율적입니다. 집계 연산자(`sum`, `min`, `max` 등)에 대해 지연 데이터를 무기한 처리합니다. 이 접근 방식의 단점은 데이터베이스 계층에서 추가 애플리케이션 로직을 개발하고 유지 관리해야 한다는 것입니다.
- 부분적 결과를 조기에 생성하지만 슬라이딩 윈도우 기간 동안 전체 결과를 계속 생성하는 텀블링 및 슬라이딩 윈도우를 사용합니다. 이 접근 방식은 데이터베이스 계층에서 추가 애플리케이션 로직을 추가할 필요가 없도록 upsert 대신 덮어쓰기를 사용하여 만기 데이터를 처리합니다. 이 접근 방식의 단점은 이것이 더 많은 Kinesis 처리 단위(KPU)를 사용하지만 여전히 두 개의 결과를 생성하므로 일부 사용 사례에는 효과적이지 않을 수 있다는 것입니다.

텀블링 및 슬라이딩 윈도우에 대한 자세한 설명은 윈도우 모드 쿼리 섹션을 참조하십시오.

다음 절차에서는 텀블링 윈도우 집계가 최종 결과를 생성하기 위해 결합해야 하는 두 개의 부분적 결과(`CALC_COUNT_SQL_STREAM` 애플리케이션 내 스트림으로 전송됨)를 생성합니다. 그런 다음 애플리케이션은 두 개의 부분적 결과를 결합하는 두 번째 집계(`DESTINATION_SQL_STREAM` 애플리케이션 내 스트림으로 전송됨)를 생성합니다.

이벤트 시간을 사용하여 부분적 결과를 집계하는 애플리케이션을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Analytics(데이터 분석)를 선택합니다. 교재 Amazon Kinesis Data Analytics for SQL 애플리케이션 시작하기의 설명에 따라 Kinesis Data Analytics 애플리케이션을 생성합니다.
3. SQL 편집기에서 애플리케이션 코드를 다음으로 바꿉니다.

```
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);
```

애플리케이션 코드의 SELECT문이 SOURCE_SQL_STREAM_001에서 주가 변동이 1%보다 큰 행을 필터링하여 펌프를 사용하여 또 다른 애플리케이션 내 스트림 CHANGE_STREAM에 삽입합니다.

4. [Save and run SQL]을 선택합니다.

첫 번째 펌프는 다음과 비슷한 CALC_COUNT_SQL_STREAM에 스트림을 출력합니다. 결과 집합은 불완전합니다.

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

그런 다음 두 번째 펌프는 전체 결과 집합이 포함된 `DESTINATION_SQL_STREAM`에 스트림을 출력합니다.

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

예: 조인

이 섹션에서는 조인 쿼리를 사용하는 Kinesis Data Analytics 애플리케이션 예를 보여드립니다. 각 예는 Kinesis Data Analytics 애플리케이션의 설정 및 테스트를 위한 단계별 지침과 코드를 제공합니다.

주제

- 예: 참조 데이터를 Kinesis Data Analytics 애플리케이션에 추가

예: 참조 데이터를 Kinesis Data Analytics 애플리케이션에 추가

이 연습에서는 참조 데이터를 기존 Kinesis Data Analytics 애플리케이션에 추가합니다. 참조 데이터에 대한 정보는 다음 주제를 참조하십시오:

- Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식
- 애플리케이션 입력 구성

이 연습에서는 Kinesis Data Analytics 시작하기 연습에서 생성한 애플리케이션에 참조 데이터를 추가합니다. 참조 데이터는 각 티커 기호에 회사 명칭을 부여합니다; 예:

```
Ticker, Company
AMZN,Amazon
ASD, SomeCompanyA
MMB, SomeCompanyB
```

```
WAS,  SomeCompanyC
```

우선 시작하기 연습의 단계를 완료하여 스타터 애플리케이션을 생성합니다. 그런 다음 이러한 단계를 따라 참조 데이터를 설정하고 애플리케이션에 추가합니다.

1. 데이터 준비
   - Amazon Simple Storage Service (Amazon S3) 의 객체로 저장합니다.
   - Kinesis Data Analytics가 사용자를 대신하여 Amazon S3 객체를 읽을 수 있도록 IAM 역할을 생성합니다.
2. 참조 데이터 소스를 애플리케이션에 추가합니다.

   Kinesis Data Analytics는 Amazon S3 객체를 읽고 애플리케이션 코드에서 쿼리할 수 있는 애플리케이션 내 참조 표를 생성합니다.
3. 코드를 테스트합니다.

   애플리케이션 코드에서 조인 쿼리를 작성하여 애플리케이션 내 스트림을 애플리케이션 내 참조 표와 조인시켜 회사 명칭에 각 티커 기호를 부여합니다.

주제

- 1단계: 준비
- 2단계: 참조 데이터 원본을 애플리케이션 구성에 추가
- 3단계: 애플리케이션 내 참조 표 쿼리 테스트

1단계: 준비

이 섹션에서는 샘플 참조 데이터를 Amazon S3 버킷에 객체로 저장합니다. Kinesis Data Analytics가 사용자를 대신하여 객체를 읽을 수 있도록 IAM 역할을 생성합니다.

참조 데이터를 Amazon S3 객체로 저장

이 단계에서는 샘플 참조 데이터를 Amazon S3 객체로 저장합니다.

1. 텍스트 편집기를 열고 다음 데이터를 추가한 다음 파일을 `TickerReference.csv`로 저장합니다.

   ```
   Ticker, Company
   AMZN,Amazon
   ASD, SomeCompanyA
   ```

```
MMB, SomeCompanyB
WAS,  SomeCompanyC
```

2. `TickerReference.csv` 파일을 S3 버킷에 업로드합니다. Amazon Simple Storage Service 사용자 가이드에서 Amazon S3로 객체 업로드를 참조하십시오.

IAM 역할 생성

다음으로, Kinesis Data Analytics가 사용자를 대신하여 Amazon S3 객체를 읽을 수 있도록 IAM 역할을 생성합니다.

1. AWS Identity and Access Management (IAM)에서 라는 IAM 역할을 생성합니다**`KinesisAnalytics-ReadS3Object`**. 이 역할을 생성하려면 IAM 사용자 가이드에서 Amazon 서비스를 위한 역할 생성 (AWS Management Console)의 지침을 따르십시오.

   IAM 콘솔에서 다음을 지정합니다:

   - 역할 유형 선택에서 AWS Lambda을 선택합니다. 역할을 생성한 후 신뢰 정책을 변경하여 Kinesis Data Analytics(그렇지 않음 AWS Lambda)가 역할을 수임하도록 허용합니다.
   - [Attach Policy] 페이지에서 정책을 연결하지 않습니다.
2. IAM 역할 정책 업데이트:

   a. IAM 콘솔에서 생성한 역할을 선택합니다.

   b. 신뢰 관계 탭에서 신뢰 정책을 업데이트하여 Kinesis Data Analytics에게 역할을 맡을 권한을 부여합니다. 신뢰 정책은 다음과 같습니다:

      JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
```

c. 허락 탭에서 AmazonS3ReadOnlyAccess라고 하는 Amazon 관리형 정책을 연결합니다. 이렇게 하면 Amazon S3 객체를 읽을 수 있는 역할 권한이 부여됩니다. 이 정책은 다음과 같습니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

2단계: 참조 데이터 원본을 애플리케이션 구성에 추가

이 단계에서는 애플리케이션 구성에 참조 데이터 원본을 추가합니다. 시작하려면 다음 정보가 필요합니다.

- S3 버킷 명칭 및 객체 키 명칭
- IAM 역할 Amazon 리소스 이름(ARN)

1. 애플리케이션의 기본 페이지에서 Connect reference data(참조 데이터 연결)를 선택합니다.
2. 참조 데이터 리소스 연결 페이지에서 참조 데이터 객체가 포함된 Amazon S3 버킷을 선택하고 객체의 키 명칭을 입력합니다.
3. 애플리케이션 내 참조 표 명칭에 **`CompanyName`**을 입력합니다.

4. Access to chosen resources(선택한 리소스에 액세스) 섹션에서 Choose from IAM roles that Kinesis Analytics can assume(Kinesis Analytics가 수임할 수 있는 IAM 역할 중에 선택)을 선택하고, 이전 섹션에서 생성한 KinesisAnalytics-ReadS3Object IAM 역할을 선택합니다.
5. Discover schema(스키마 발견)를 선택합니다. 콘솔에서 참조 데이터에 있는 두 개의 열이 감지됩니다.
6. [Save and close]를 선택합니다.

3단계: 애플리케이션 내 참조 표 쿼리 테스트

이제 애플리케이션 내 참조 표 `CompanyName`을 쿼리할 수 있습니다. 참조 정보를 활용하여 티커 가격 데이터를 참조 표에 조인하면 애플리케이션을 보강할 수 있습니다. 결과는 회사 명칭을 표시합니다.

1. 애플리케이션 코드를 다음으로 대체합니다. 쿼리가 애플리케이션 내 입력 스트림을 애플리케이션 내 참조 표와 조인시킵니다. 애플리케이션 코드가 결과를 또 다른 애플리케이션 내 스트림 `DESTINATION_SQL_STREAM`에 작성합니다.

```
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. 애플리케이션 출력이 [SQLResults] 탭에 표시되는지 확인합니다. 일부 행이 회사 명칭을 표시하는지 확인합니다(샘플 참조 데이터가 모든 회사 명칭을 가지고 있지는 않습니다).

예: 기계 학습

이 섹션에서는 기계 학습 쿼리를 사용하는 Amazon Kinesis Data Analytics 애플리케이션 예를 소개합니다. 기계 학습 쿼리는 스트림의 데이터 기록에 의존하여 데이터의 복잡한 분석을 수행하고 비정상적인 패턴을 찾습니다. 이 예에서는 Kinesis Data Analytics 애플리케이션 설정 및 테스트 방법을 단계적으로 설명합니다.

주제

- 예: 스트림에서 데이터 변칙을 감지하는 방법(RANDOM_CUT_FOREST 함수)
- 예: 데이터 변칙 감지 및 설명(RANDOM_CUT_FOREST_WITH_EXPLANATION 함수)

- 예: 스트림에서 핫스팟 감지(HOTSPOTS 함수)

예: 스트림에서 데이터 변칙을 감지하는 방법(RANDOM_CUT_FOREST 함수)

Amazon Kinesis Data Analytics는 수 열에 있는 값을 바탕으로 각 레코드의 이상 점수를 할당할 수 있는 함수(RANDOM_CUT_FOREST)를 제공합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 RANDOM_CUT_FOREST 함수를 참조하십시오.

이 실습에서는 애플리케이션 코드를 작성해 변칙 점수를 애플리케이션의 스트리밍 소스에 있는 레코드에 할당합니다. 애플리케이션을 설정하려면 다음을 수행합니다:

1. 스트리밍 소스 설정 – 다음과 같이 샘플 heartRate 데이터 스트림을 설정하고 샘플 데이터를 작성합니다:

   ```
   {"heartRate": 60, "rateType":"NORMAL"}
   ...
   {"heartRate": 180, "rateType":"HIGH"}
   ```

   이 절차에서는 Python 스크립트를 통해 스트림을 채우는 방법을 알아봅니다. heartRate 값은 무작위로 생성되는데, heartRate 값이 60과 100 사이인 레코드가 99%이고 heartRate 값이 150과 200 사이인 레코드는 1%밖에 되지 않습니다. 따라서 heartRate 값이 150과 200 사이인 레코드는 변칙입니다.

2. 입력 구성 – 콘솔을 사용하여 Kinesis Data Analytics 애플리케이션을 생성하고, 스트리밍 소스를 애플리케이션 내 스트림(SOURCE_SQL_STREAM_001)에 매핑함으로써 애플리케이션 입력을 구성합니다. 애플리케이션을 시작하면 Kinesis Data Analytics이 지속적으로 스트리밍 소스를 읽어서 애플리케이션 내 스트림에 레코드를 삽입합니다.

3. 애플리케이션 코드 지정 – 이 예는 다음 애플리케이션 코드를 사용합니다:

   ```
   --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;
```

코드가 SOURCE_SQL_STREAM_001에 있는 행을 읽고, 변칙 점수를 할당하고, 결과 행을 또 다른 애플리케이션 내 스트림(TEMP_STREAM)에 작성합니다. 그런 다음 애플리케이션 코드가 TEMP_STREAM에 있는 레코드를 정렬하고 결과를 또 다른 애플리케이션 내 스트림(DESTINATION_SQL_STREAM)에 저장합니다. 펌프를 사용하여 행을 애플리케이션 내 스트림에 삽입합니다. 자세한 설명은 애플리케이션 내 스트림과 펌프 섹션을 참조하십시오.

4. 출력 구성 – DESTINATION_SQL_STREAM에 있는 데이터를 또 다른 Kinesis 데이터 스트림인 외부 대상에 유지하도록 애플리케이션 출력을 구성합니다. 각 레코드에 할당된 변칙 점수를 검토하고 어떤 점수가 변칙을 나타내는지 판단하는 것은 애플리케이션 외부에서 이루어집니다. AWS Lambda 함수를 사용하여 이러한 이상 점수를 처리하고 알림을 구성할 수 있습니다.

이 연습에서는 미국 동부(버지니아 북부)us-east-1)를 사용하여 이러한 스트림과 애플리케이션을 생성합니다. 다른 리전을 사용하는 경우 그에 따라 코드를 업데이트해야 합니다.

주제

- 1단계: 준비
- 단계 2: 애플리케이션 만들기
- 3단계: 애플리케이션 출력 구성
- 4단계: 출력 확인

다음 단계

1단계: 준비

1단계: 준비

이 연습에 사용할 Amazon Kinesis Data Analytics 애플리케이션을 생성하기 전에 Kinesis 데이터 스트림 두 개를 생성해야 합니다. 스트림 중 하나를 애플리케이션의 스트리밍 소스로 구성하고 또 다른 스트림을 Kinesis Data Analytics가 애플리케이션 출력을 유지하는 목적지로 구성합니다.

주제

- 1.1단계: 입력 및 출력 데이터 스트림 생성
- 1.2단계: 샘플 레코드를 입력 스트림에 작성

1.1단계: 입력 및 출력 데이터 스트림 생성

이 섹션에서는 2개의 Kinesis 스트림을 생성합니다: `ExampleInputStream` 및 `ExampleOutputStream`. AWS Management Console 또는 AWS CLI을(를) 사용하여 이러한 스트림을 만들 수 있습니다.

- 콘솔을 사용하려면
  1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
  2. 데이터 스트림 생성을 선택합니다. 샤드가 하나인 스트림(`ExampleInputStream`이라고 함)을 생성합니다. 자세한 설명은 Amazon Kinesis Data Streams 개발자 가이드의 스트림 생성을 참조하십시오.
  3. 이전 단계를 반복하여 샤드가 하나인 스트림(`ExampleOutputStream`이라고 함)을 생성합니다.
- 를 사용하려면 AWS CLI
  1. 다음 Kinesis `create-stream` AWS CLI 명령을 사용하여 첫 번째 스트림()을 생성합니다`ExampleInputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

2. 동일한 명령을 실행하여 스트림 명칭을 `ExampleOutputStream`으로 변경합니다. 이 명령은 두 번째 스트림을 생성하고 애플리케이션은 이를 사용하여 출력을 작성합니다.

1.2단계: 샘플 레코드를 입력 스트림에 작성

이 단계에서는 Python 코드를 실행하여 샘플 레코드를 연속적으로 생성하고 이러한 레코드를 `ExampleInputStream` 스트림에 작성합니다.

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

1. Python 및 `pip`를 설치합니다.

   Python 설치에 관한 정보는 Python 웹사이트를 참조하십시오.

   pip를 사용하여 종속 프로그램을 설치할 수 있습니다. pip 설치에 관한 정보는 pip 웹 사이트에 있는 Installation을 참조하십시오.
2. 다음 Python 코드를 실행합니다. 코드에 있는 `put-record` 명령이 JSON 레코드를 스트림에 작성합니다.

```
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"))
```

다음 단계

단계 2: 애플리케이션 만들기

단계 2: 애플리케이션 만들기

이 섹션에서는 다음과 같이 Amazon Kinesis Data Analytics 애플리케이션을 생성합니다:

- the section called “1단계: 준비”에서 스트리밍 소스로 생성한 Kinesis 데이터 스트림을 사용하도록 애플리케이션 입력을 구성합니다.
- 콘솔에서 변칙 감지 템플릿을 사용합니다.

애플리케이션 생성

1. Kinesis Data Analytics 시작하기 연습에서의 1, 2 및 3단계를 따르십시오 (단계 3.1: 애플리케이션 만들기 참조).
    - 소스 구성에서 다음을 수행합니다:
        - 이전 섹션에서 생성한 스트리밍 소스를 지정합니다.

- 콘솔이 스키마를 유추한 후에 스키마를 편집하고 `heartRate` 열 유형을 `INTEGER`로 설정합니다.

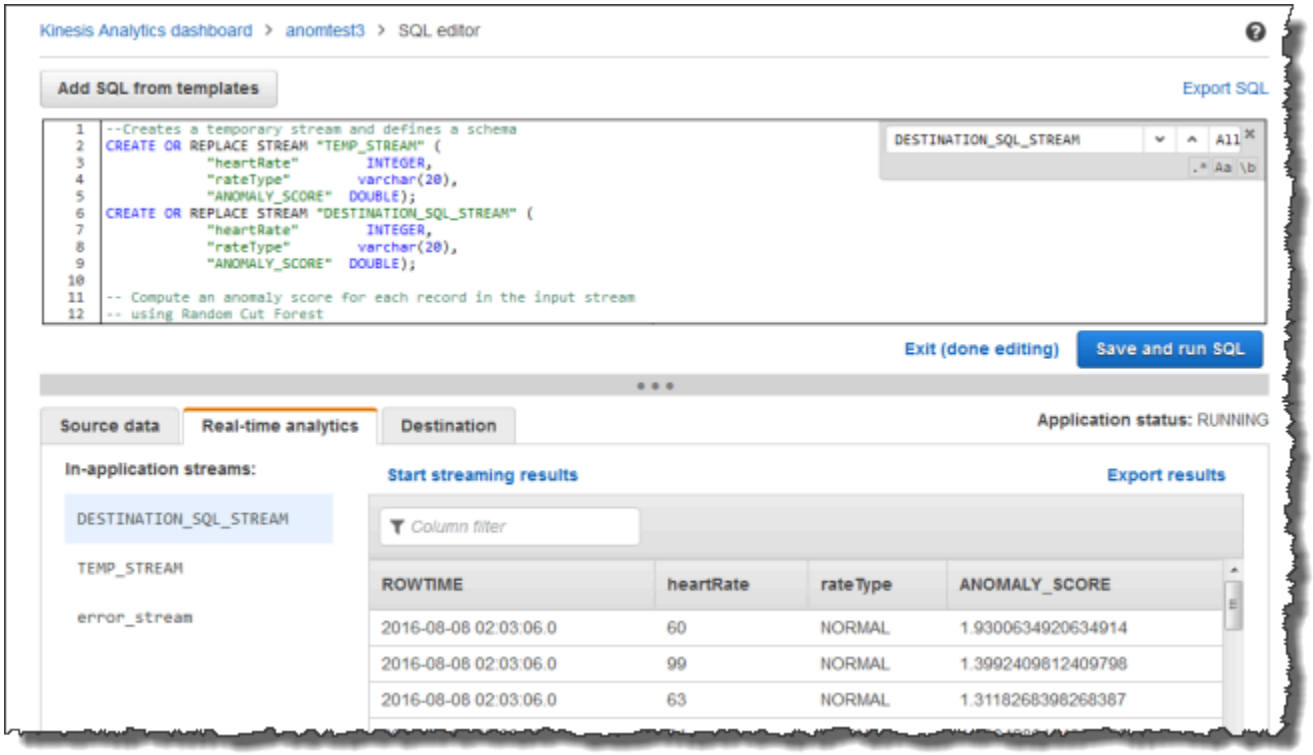
  대부분의 심박수 값은 정상이며 검색 프로세스는 이 열에 `TINYINT` 유형을 할당할 가능성이 높습니다. 그러나 높은 심박수를 나타내는 값은 백분율이 매우 낮습니다. 이러한 높은 값이 `TINYINT` 유형에 부합하지 않을 경우 Kinesis Data Analytics는 해당 행을 오류 스트림으로 전송합니다. 생성된 모든 심박수 데이터를 수용할 수 있도록 데이터 유형을 `INTEGER`로 업데이트합니다.

- 콘솔에서 변칙 감지 템플릿을 사용합니다. 그런 다음 템플릿 코드를 업데이트하여 적절한 열 명칭을 부여합니다.

2. 열 명칭을 부여하여 애플리케이션 코드를 업데이트합니다. 결과로 얻은 애플리케이션 코드는 다음과 같이 표시됩니다(이 코드를 복사하여 SQL 편집기에 붙여넣습니다).

```
--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. Kinesis Data Analytics 콘솔에서 SQL 코드를 실행하고 결과를 검토합니다:



다음 단계

[3단계: 애플리케이션 출력 구성]

3단계: 애플리케이션 출력 구성

the section called "단계 2: 애플리케이션 만들기" 완료 이후 스트리밍 소스로부터 심박수 데이터를 읽고 변칙 점수를 각각에 할당하는 애플리케이션 코드를 가지고 있습니다.

이제 애플리케이션 내 스트림에서 외부 목적지인 또 다른 Kinesis 데이터 스트림 (`OutputStreamTestingAnomalyScores`)으로 애플리케이션 결과를 전송할 수 있습니다. 변칙 점수를 분석하여 어떤 심박수가 변칙인지 판단할 수 있습니다. 이 애플리케이션을 더욱 확장하여 알림을 만들 수 있습니다.

다음 단계에 따라 애플리케이션 출력을 구성합니다.

1. Amazon Kinesis Data Analytics 콘솔을 엽니다. SQL 편집기에서 [Destination]을 선택하거나 애플리케이션 대시보드에서 [Add a destination]을 선택합니다.

2. Connect to destination(대상 주소에 연결) 페이지에서 이전 섹션을 통해 생성한 `OutputStreamTestingAnomalyScores` 스트림을 선택합니다.

   이제 애플리케이션 내 스트림 `DESTINATION_SQL_STREAM`에 애플리케이션이 작성하는 모든 레코드를 Amazon Kinesis Data Analytics가 유지하는 외부 대상이 생성되었습니다.

3. 선택적으로 `OutputStreamTestingAnomalyScores` 스트림을 모니터링하고 알림을 보내 AWS Lambda 도록를 구성할 수 있습니다. 지침은 Lambda 함수를 사용하여 데이터 사전 처리 단원을 참조하십시오. 알림을 설정하지 않으면 4단계: 출력 확인에 설명된 대로, Kinesis Data Analytics이 외부 목적지인 Kinesis 데이터 스트림 `OutputStreamTestingAnomalyScores`으로 기록하는 레코드를 검토할 수 있습니다.

다음 단계

4단계: 출력 확인

4단계: 출력 확인

the section called “3단계: 애플리케이션 출력 구성”에서 애플리케이션 출력을 구성한 이후 다음 AWS CLI 명령을 사용하여 애플리케이션이 작성한 대상 스트림 내의 레코드를 읽습니다.

1. `get-shard-iterator` 명령을 실행하여 출력 스트림 상의 데이터에 대한 포인터를 확보합니다.

```
aws kinesis get-shard-iterator \
--shard-id shardId-000000000000 \
--shard-iterator-type TRIM_HORIZON \
--stream-name OutputStreamTestingAnomalyScores \
--region us-east-1 \
--profile adminuser
```

   다음 예 응답에서와 같이 샤드 반복자 값을 지닌 응답을 얻습니다:

```
  {
      "ShardIterator":
      "shard-iterator-value"   }
```

   샤드 반복자 값을 복사합니다.

2. AWS CLI `get-records` 명령을 실행합니다.

```
aws kinesis get-records \
```

```
--shard-iterator shared-iterator-value \
--region us-east-1 \
--profile adminuser
```

이 명령은 다음 레코드 세트를 가져오는 후속 `get-records` 명령에서 사용할 수 있는 레코드 페이지와 또 다른 샤드 반복자를 반환합니다.

예: 데이터 변칙 감지 및 설명 (RANDOM_CUT_FOREST_WITH_EXPLANATION 함수)

Amazon Kinesis Data Analytics은 수치 열에 있는 값을 바탕으로 각 레코드에 이상 점수를 할당하는`RANDOM_CUT_FOREST_WITH_EXPLANATION` 함수를 제공합니다. 또한 함수는 변칙에 대한 설명을 제공합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조의 RANDOM_CUT_FOREST_WITH_EXPONSION을 참조하십시오.

이 연습에서는 애플리케이션 코드를 작성하여 애플리케이션의 스트리밍 소스에 있는 레코드의 변칙 점수를 가져옵니다. 또한 각 변칙에 대한 설명을 얻습니다.

주제

- 1단계: 데이터 준비
- 2단계: 분석 애플리케이션 생성
- 3단계: 결과 검사

첫 단계

1단계: 데이터 준비

1단계: 데이터 준비

이 예의 경우 Amazon Kinesis Data Analytics 애플리케이션을 생성하기 전에 애플리케이션의 스트리밍 소스로 사용할 Kinesis 데이터 스트림을 생성합니다. 또한 Python 코드를 실행하여 시뮬레이션된 혈압 데이터를 스트림에 작성합니다.

주제

- 1.1단계: Kinesis 데이터 스트림 생성
- 1.2단계: 샘플 레코드를 입력 스트림에 작성

1.1단계: Kinesis 데이터 스트림 생성

이 섹션에서는 `ExampleInputStream`이라는 Kinesis 데이터 스트림을 생성합니다. AWS Management Console 또는를 사용하여이 데이터 스트림을 생성할 수 있습니다 AWS CLI.

- 콘솔을 사용하려면:
  1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
  2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다. 그런 다음 Kinesis 스트림 생성을 선택합니다.
  3. 명칭에는 **`ExampleInputStream`**을(를) 입력합니다. 샤드 수에는 **`1`**을(를) 입력합니다.
- 또는 AWS CLI 를 사용하여 데이터 스트림을 생성하려면 다음 명령을 실행합니다.

```
$ aws kinesis create-stream --stream-name ExampleInputStream --shard-count 1
```

1.2단계: 샘플 레코드를 입력 스트림에 작성

이 단계에서는 Python 코드를 실행하여 샘플 레코드를 연속적으로 생성하고 생성한 데이터 스트림에 작성합니다.

1. Python 및 pip를 설치합니다.

   Python 설치에 관한 자세한 설명은 Python을 참조하십시오.

   pip를 사용하여 종속 프로그램을 설치할 수 있습니다. pip 설치에 관한 자세한 설명은 pip 설명서의 설치를 참조하십시오.
2. 다음 Python 코드를 실행합니다. 리전은 이 예에서 사용할 리전으로 변경할 수 있습니다. 코드에 있는 `put-record` 명령이 JSON 레코드를 스트림에 작성합니다.

```
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"))
```

다음 단계

2단계: 분석 애플리케이션 생성

2단계: 분석 애플리케이션 생성

이 섹션에서는 Amazon Kinesis Data Analytics 애플리케이션을 생성하고 the section called "1단계: 데이터 준비"에서 스트리밍 소스로 만든 Kinesis 데이터 스트림을 사용하도록 구성합니다. 그런 다음 `RANDOM_CUT_FOREST_WITH_EXPLANATION` 함수를 사용하는 애플리케이션 코드를 실행합니다.

애플리케이션 생성

1. https://console.aws.amazon.com/kinesis에서 Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Data Analytics(데이터 분석)를 선택한 다음 애플리케이션 생성을 선택합니다.
3. 애플리케이션 명칭 및 설명(선택 사항)을 입력하고 [Create application]을 선택합니다.
4. 스트리밍 데이터 연결을 선택한 다음 목록에서 ExampleInputStream을 선택합니다.
5. [Discover schema]를 선택하고 `Systolic` 및 `Diastolic`이 `INTEGER` 열로 나타나는지 확인하십시오. 다른 유형이 있으면 [Edit schema]를 선택하고 유형 `INTEGER`를 두 유형 모두에 할당합니다.
6. [Real time analytics]에서 [Go to SQL editor]를 선택합니다. 메시지가 표시되면 애플리케이션을 실행하도록 선택합니다.
7. 다음 코드를 SQL 편집기에 붙여넣은 다음 [Save and run SQL]을 선택합니다.

```
--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;
```

다음 단계

3단계: 결과 검사

3단계: 결과 검사

이 예에 대한 SQL 코드를 실행하면 먼저 변칙 점수가 0인 행이 표시됩니다. 이는 초기 학습 단계 중에 발생합니다. 그런 다음 다음과 비슷한 결과를 얻게 됩니다:

```
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"}}
```

- `RANDOM_CUT_FOREST_WITH_EXPLANATION` 함수의 알고리즘에서는 `Systolic` 및 `Diastolic` 열이 숫자이며 이들 열을 입력으로 사용합니다.
- `BloodPressureLevel` 열은 텍스트 데이터를 포함하므로 알고리즘에 의해 고려되지 않습니다. 이 열은 시각적으로 보조하기 위한 것으로, 이 예에서 정상, 최고 및 최저 혈압 레벨을 신속하게 파악하는 데 도움이 됩니다.
- `ANOMALY_SCORE` 열에서 점수가 더 높은 레코드는 보다 더 변칙적입니다. 이 샘플 결과 세트의 두 번째 레코드는 변칙 점수가 3.855851061로, 가장 변칙적입니다.
- 알고리즘에 의해 고려된 각 숫자 열이 변칙 점수에 영향을 미치는 정도를 이해하려면 `ANOMALY_SCORE` 열에서 `ATTRIBUTION_SCORE`라는 JSON 필드를 참조하십시오. 이 샘플 결과 세트의 두 번째 행의 경우 `Systolic` 및 `Diastolic` 행은 1.7447:2.1111 비율로 변칙에 영향을 미칩니다. 즉 변칙 점수에 대한 설명의 45%는 수축 값에 의한 것이고 나머지 속성은 이완 값에 의한 것입니다.
- 이 샘플의 두 번째 행에 표시된 점의 변칙성을 확인하려면 `DIRECTION`이라는 JSON 필드를 참조하십시오. 이 경우 이완 및 수축 값 모두 `HIGH`로 표시됩니다. 이러한 방향의 정확성에 대한 신뢰도를 확인하려면 `STRENGTH`라는 JSON 필드를 참조하십시오. 이 예에서 알고리즘은 이완 값이 높다는 것을 보다 더 확신합니다. 실제로 정상적인 확장 판독 값은 보통 60에서 80 사이이며 123은 예상보다 훨씬 높습니다.

예: 스트림에서 핫스팟 감지(HOTSPOTS 함수)

Amazon Kinesis Data Analytics는 데이터에서 상대적으로 밀도가 높은 리전에 대한 정보를 찾아 반환할 수 있는 `HOTSPOTS` 함수를 제공합니다. 자세한 설명은 Amazon Managed Service for Apache Flink SQL 참조에서 핫스팟을 참조하십시오.

이 실습에서는 애플리케이션 코드를 작성해 애플리케이션의 스트리밍 소스에서 핫스팟을 찾습니다. 애플리케이션을 설정하려면 다음 단계를 수행합니다.

1. 스트리밍 소스 설정 – 다음과 같이 Kinesis 스트림을 설정하고 샘플 좌표 데이터를 작성합니다:

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626528026, "y": 4.648868803193405, "is_hot": "Y"}
```

이 예에서는 Python 스크립트를 통해 스트림을 채우는 방법을 알아봅니다. x 및 y 값은 무작위로 생성되며, 특정 위치 주변에서 일부 기록이 클러스터링됩니다.

`is_hot` 필드가 제공되어 스크립트가 의도적으로 핫스팟의 일부로서 값을 생성했는지 여부를 나타냅니다. 이는 핫스팟 감지 함수가 제대로 작동하고 있는지 평가하는 데 도움이 될 수 있습니다.

2. 애플리케이션 생성 – AWS Management Console를 사용하여 Kinesis Data Analytics 애플리케이션을 생성합니다. 스트리밍 소스를 애플리케이션 내 스트림(`SOURCE_SQL_STREAM_001`)으로 매핑하여 애플리케이션 입력을 구성합니다. 애플리케이션을 시작하면 Kinesis Data Analytics가 지속적으로 스트리밍 소스를 읽고 애플리케이션 내 스트림에 레코드를 삽입합니다.

   이 연습에서는 애플리케이션에 대해 다음 코드를 사용합니다.

```
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)
    );
```

   코드가 `SOURCE_SQL_STREAM_001`에 있는 행을 읽고, 중요 핫스팟에 대해 이를 분석하고, 결과 데이터를 또 다른 애플리케이션 내 스트림(`DESTINATION_SQL_STREAM`)에 작성합니다. 펌프를 사용하여 행을 애플리케이션 내 스트림에 삽입합니다. 자세한 설명은 애플리케이션 내 스트림과 펌프 섹션을 참조하십시오.

3. 출력 구성 – 애플리케이션이 외부 목적지인 또 다른 Kinesis 데이터 스트림으로 데이터를 보내도록 애플리케이션 출력을 구성합니다. 핫스팟 점수를 검토하고 어떤 점수가 핫스팟 발생(알림이 발생해야 함)을 나타내는지 판단합니다. AWS Lambda 함수를 사용하여 핫스팟 정보를 추가로 처리하고 알림을 구성할 수 있습니다.

4. 출력 확인 – 이 예에는 출력 스트림으로부터 데이터를 읽고 이를 그래픽으로 표현하는 JavaScript 애플리케이션이 포함되어 있습니다. 따라서 애플리케이션이 실시간으로 생성하는 핫스팟을 볼 수 있습니다.

이 연습에서는 미국 서부(오레곤)(`us-west-2`)을 사용하여 이러한 스트림과 애플리케이션을 생성합니다. 다른 리전을 사용하는 경우 그에 따라 코드를 업데이트합니다.

주제

- 1단계: 입력 및 출력 스트림 생성
- 2단계: Kinesis Data Analytics 애플리케이션 생성
- 3단계: 애플리케이션 출력 구성
- 4단계: 애플리케이션 출력 확인

1단계: 입력 및 출력 스트림 생성

핫스팟 예를 위해 Amazon Kinesis Data Analytics 애플리케이션을 생성하기 전에 먼저 Kinesis 데이터 스트림 2개를 생성합니다. 스트림 중 하나를 애플리케이션의 스트리밍 소스로 구성하고 또 다른 스트림을 Kinesis Data Analytics가 애플리케이션 출력을 유지하는 목적지로 구성합니다.

주제

- 1.1단계: Kinesis 데이터 스트림 생성
- 1.2단계: 샘플 레코드를 입력 스트림에 작성

1.1단계: Kinesis 데이터 스트림 생성

이 섹션에서는 다음 2개의 Kinesis 데이터 스트림을 생성합니다: `ExampleInputStream` 및 `ExampleOutputStream`.

콘솔 또는 AWS CLI을(를) 사용하여 데이터 스트림을 생성합니다.

- 콘솔을 사용하여 데이터 스트림을 생성:

  1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
  2. 탐색 창에서 Data Streams(데이터 스트림)를 선택합니다.

3. Kinesis 스트림 생성을 선택한 다음 샤드가 하나인 스트림(`ExampleInputStream`이라고 함)을 생성합니다.
4. 이전 단계를 반복하여 샤드가 하나인 스트림(`ExampleOutputStream`이라고 함)을 생성합니다.

- AWS CLI을(를) 사용하여 데이터 스트림 생성:
  - 다음 Kinesis `create-stream` AWS CLI 명령을 사용하여 스트림(`ExampleInputStream` 및 `ExampleOutputStream`)을 생성합니다. 애플리케이션이 출력을 작성하기 위해 사용할 두 번째 스트림을 생성하려면 동일한 명령을 실행하여 스트림 명칭을 `ExampleOutputStream`으로 변경합니다.

```
$ 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
```

1.2단계: 샘플 레코드를 입력 스트림에 작성

이 단계에서는 Python 코드를 실행하여 샘플 레코드를 연속적으로 생성하고 `ExampleInputStream` 스트림에 작성합니다.

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626580026, "y": 4.648868803193405, "is_hot": "Y"}
```

1. Python 및 `pip`를 설치합니다.

   Python 설치에 관한 정보는 Python 웹사이트를 참조하십시오.

   pip를 사용하여 종속 프로그램을 설치할 수 있습니다. pip 설치에 관한 정보는 pip 웹 사이트에 있는 Installation을 참조하십시오.
2. 다음 Python 코드를 실행합니다. 이 코드는 다음을 수행합니다.

- (X, Y) 평면 어딘가에 잠재적 핫스팟을 생성합니다.
- 각 핫스팟마다 1,000포인트 세트를 생성합니다. 이 포인트에서 20%가 핫스팟 주변에 클러스터링됩니다. 나머지는 전체 공간 내에 무작위로 생성됩니다.
- `put-record` 명령은 JSON 레코드를 스트림에 작성합니다.

> **Important**
>
> 이 파일에는 귀하의 AWS 자격 증명이 포함되어 있으므로 이 파일을 웹 서버에 업로드하지 마십시오.

```
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"),
    )
```

다음 단계

[2단계: Kinesis Data Analytics 애플리케이션 생성](#)

2단계: Kinesis Data Analytics 애플리케이션 생성

핫스팟 예 섹션에서 다음과 같이 Kinesis Data Analytics 애플리케이션을 생성합니다:

- 1 단계에서 스트리밍 소스로 생성한 Kinesis 데이터 스트림을 사용하도록 애플리케이션 입력을 구성합니다.
- AWS Management Console에서 제공된 애플리케이션 코드를 사용합니다.

애플리케이션 생성

1. 시작하기 연습 (단계 3.1: 애플리케이션 만들기 참조) 의 1, 2, 3단계에 따라 Kinesis Data Analytics 애플리케이션을 생성합니다.

   소스 구성에서 다음을 수행합니다:

   - the section called “1단계: 스트림 생성” 섹션에서 생성한 스트리밍 소스를 지정합니다.
   - 콘솔에서 스키마를 유추한 이후 스키마를 편집합니다. x 및 y열 유형이 DOUBLE로 설정되고 IS_HOT 열 유형이 VARCHAR로 설정되어야 합니다.
2. 다음 애플리케이션 코드를 사용합니다(이 코드를 복사하여 SQL 편집기에 붙여넣을 수 있습니다).

```
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. SQL 코드를 실행하고 결과를 검토합니다.

Filter by column name

ROWTIME	x	y	is_hot	HOTSPOTS_RESULT
2018-03-19 20:19:20.298	3.2902233757560313	1.1460673734716675	N	{"hotspots":[{"density":159.34972933221212,"minValues":[0.4791038226753084,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040
2018-03-19 20:19:21.313	9.758694911135013	9.66632832516424	N	{"hotspots":[{"density":180.8921951354484,"minValues":[0.6623354726891375,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040
2018-03-19 20:19:21.313	8.986657300548824	3.558000293320571	N	{"hotspots":[{"density":180.8921951354484,"minValues":[0.6623354726891375,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040
2018-03-19 20:19:21.313	5.193048038272014	4.94448855569874	Y	{"hotspots":[{"density":180.8921951354484,"minValues":[0.6623354726891375,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040

다음 단계

[3단계: 애플리케이션 출력 구성]

3단계: 애플리케이션 출력 구성

이 시점에 핫스팟 예에서 Amazon Kinesis Data Analytics 애플리케이션 코드를 보유하여 스트리밍 소스로부터 중요 핫스팟을 발견하고 히트 점수를 각각에게 할당합니다.

이제 애플리케이션 내 스트림에서 외부 목적지인 또 다른 Kinesis 데이터 스트림 (`ExampleOutputStream`)으로 에플리케이션 결과를 보낼 수 있습니다. 그런 다음 핫스팟 점수를 분석하고 핫스팟 히트에 대한 적절한 임계점을 판단할 수 있습니다. 이 애플리케이션을 더욱 확장하여 알림을 만들 수 있습니다.

애플리케이션 출력을 구성하려면

1. https://console.aws.amazon.com/kinesisanalytics에서 Kinesis Data Analytics 콘솔을 엽니다.
2. SQL 편집기에서 [Destination]을 선택하거나 애플리케이션 대시보드에서 [Add a destination]을 선택합니다.
3. Add a destination(대상 주소 추가) 페이지에서 Select from your streams(나의 스트림에서 선택)을 선택합니다. 그런 다음 이전 섹션에서 생성한 `ExampleOutputStream` 스트림을 선택합니다.

   이제 애플리케이션 내 스트림 `DESTINATION_SQL_STREAM`에 애플리케이션이 작성하는 모든 레코드를 Amazon Kinesis Data Analytics가 유지하는 외부 대상이 생성되었습니다.
4. 선택적으로 `ExampleOutputStream` 스트림을 모니터링하고 알림을 보내 AWS Lambda 도록를 구성할 수 있습니다. 자세한 내용은 Lambda 함수를 출력으로 사용 단원을 참조하십시오. 4단계: 애플리케이션 출력 확인에서 설명한 대로 Kinesis Data Analytics가 외부 목적지인 Kinesis 스트림 `ExampleOutputStream`에 기록하는 레코드를 검토할 수 있습니다.

다음 단계

4단계: 애플리케이션 출력 확인

4단계: 애플리케이션 출력 확인

핫스팟 예의 이번 섹션에서 Scalable Vector Graphics(SVG) 제어에서 핫스팟 정보를 표시하는 웹 애플리케이션을 설정합니다.

1. 다음 콘텐츠를 가진 `index.html`이라는 파일을 생성합니다:

```
<!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. 명칭이 `hotspots_viewer.js`인 동일한 디렉터리에 다음 내용을 포함한 파일을 생성합니다. 제공된 변수에 자격 증명 및 출력 스트림 명칭을 입력합니다.

```
// 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) {
```

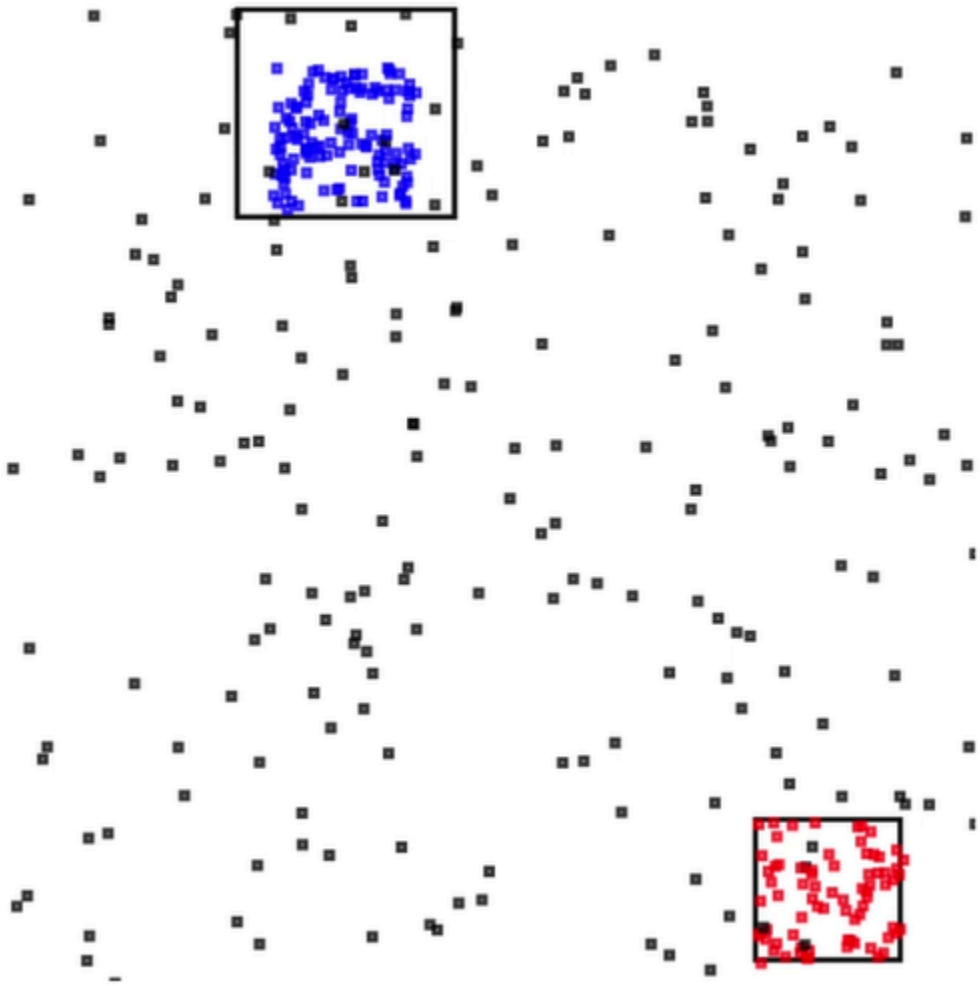
```
        return c + m * (x - a);
    };
}

// 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. 실행 중인 첫 번째 섹션의 Python 코드를 통해 웹 브라우저에서 `index.html`을 엽니다. 다음과 같이 핫스팟 정보가 페이지에 표시됩니다.



예: 알림 및 오류

이 섹션에서는 알림 및 오류를 사용하는 Kinesis Data Analytics 애플리케이션 예를 제공합니다. 각 예에는 Kinesis Data Analytics 애플리케이션을 설정 및 테스트에 도움이 되는 단계별 지침과 코드를 제공합니다.

주제

- 예: 간단한 알림 생성
- 예: 조정된 알림 생성
- 예: 애플리케이션 내 오류 스트림 탐색

예: 간단한 알림 생성

이 Kinesis Data Analytics 애플리케이션에서는 데모 스트림에 대해 생성된 애플리케이션 내 스트림 상에서 쿼리가 연속적으로 실행됩니다. 자세한 설명은 연속 쿼리 섹션을 참조하십시오.

임의의 행이 1%보다 큰 주가 변동을 보이는 경우, 해당 행은 또 다른 애플리케이션 내 스트림에 삽입됩니다. 실습에서 결과를 외부 대상에 유지하도록 애플리케이션 출력을 구성할 수 있습니다. 그런 다음 결과를 추가로 조사할 수 있습니다. 예를 들어 AWS Lambda 함수를 사용하여 레코드를 처리하고 알림을 보낼 수 있습니다.

간단한 알림 애플리케이션을 만드는 방법

1. Kinesis Data Analytics 시작하기에 설명된 대로 분석 애플리케이션을 생성하십시오.
2. Kinesis Data Analytics의 SQL 편집기에서 애플리케이션 코드를 다음으로 바꿉니다:

```
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;
```

애플리케이션 코드의 `SELECT` 문은 1%보다 큰 주가 변동에 대해 `SOURCE_SQL_STREAM_001`의 행을 필터링합니다. 그런 다음 펌프를 사용하여 다른 애플리케이션 내 스트림 `DESTINATION_SQL_STREAM`에 이러한 행을 삽입합니다. 펌프를 사용하여 행을 애플리케이션 내 스트림에 삽입하는 방법을 설명하는 코딩 패턴에 관한 자세한 설명은 애플리케이션 코드 섹션을 참조하십시오.

3. [Save and run SQL]을 선택합니다.
4. 대상을 추가합니다. 이렇게 하려면 SQL 편집기에서 대상 주소 탭을 선택하거나 애플리케이션 세부 정보 페이지에서 Add a destination(대상 추가)을 선택합니다.

   a. SQL 편집기에서 대상 주소 탭을 선택한 다음 Connect to a destination(대상에 연결)을 선택합니다.

      Connect to destination(대상에 연결) 페이지에서 Create New(새로 생성)를 선택합니다.

   b. [Go to Kinesis Streams]를 선택합니다.

   c. Amazon Kinesis Data Streams 콘솔에서 샤드가 하나인 새로운 Kinesis 스트림을 생성합니다(예: `gs-destination`). 스트림 상태가 [ACTIVE]가 될 때까지 기다립니다.

   d. Kinesis Data Analytics 콘솔로 돌아가십시오. Connect to destination(대상에 연결) 페이지에서 앞서 생성한 스트림을 선택합니다.

      스트림이 표시되지 않으면 페이지를 새로 고칩니다.

   e. [Save and continue]를 선택합니다.

   이제 외부 목적지인 Kinesis 데이터 스트림이 생겼으며 Kinesis Data Analytics이 애플리케이션 내 스트림에 애플리케이션 출력을 지속합니다.

5. 생성한 Kinesis 스트림을 모니터링하고 Lambda 함수를 호출 AWS Lambda 하도록를 구성합니다.

   지침은 Lambda 함수를 사용하여 데이터 사전 처리 단원을 참조하십시오.

예: 조정된 알림 생성

이 Kinesis Data Analytics 애플리케이션에서는 데모 스트림에 대해 생성된 애플리케이션 내 스트림 상에서 쿼리가 연속적으로 실행됩니다. 자세한 설명은 연속 쿼리 섹션을 참조하십시오. 임의의 행이 1%보다 큰 주가 변동을 보이는 경우, 해당 행은 또 다른 애플리케이션 내 스트림에 삽입됩니다. 애플리케이션은 알림을 조정하고, 주가가 변동하는 즉시 알림이 전송됩니다. 하지만 주식 기호당 1분에 1개 이상의 알림이 애플리케이션 내 스트림으로 전송되지 않습니다.

조정된 알림 애플리케이션을 생성하려면

1. Kinesis Data Analytics 시작하기 연습에 설명된 대로 Kinesis Data Analytics 애플리케이션을 생성합니다.
2. Kinesis Data Analytics의 SQL 편집기에서 애플리케이션 코드를 다음으로 바꿉니다:

```
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;
```

애플리케이션 코드의 SELECT문이 SOURCE_SQL_STREAM_001에서 주가 변동이 1%보다 큰 행을 필터링하여 펌프를 사용하여 또 다른 애플리케이션 내 스트림 CHANGE_STREAM에 삽입합니다.

그런 다음 애플리케이션은 조정된 알림에 대해 TRIGGER_COUNT_STREAM이라는 두 번째 스트림을 생성합니다. 두 번째 쿼리는 레코드가 허용될 때마다 앞으로 건너뛰는 윈도우에서 레코드를 선택하여 1분에 주식 티커당 하나의 레코드만 스트림에 작성되도록 합니다.

3. [Save and run SQL]을 선택합니다.

위 예는 다음과 비슷한 스트림을 TRIGGER_COUNT_STREAM에 출력합니다.

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

예: 애플리케이션 내 오류 스트림 탐색

Amazon Kinesis Data Analytics은 생성된 각 애플리케이션에 대해 애플리케이션 내 오류 스트림을 제공합니다. 애플리케이션이 처리할 수 없는 행은 모두 이 오류 스트림으로 전송됩니다. 조사를 위해 오류 스트림 데이터를 외부 대상에 유지하는 것을 고려할 수도 있습니다.

콘솔에서 다음 실습을 수행합니다. 이 예에서 검색 프로세스에서 유추된 스키마를 편집하여 오류를 입력 구성에 추가하고 오류 스트림에 전송된 행을 확인합니다.

주제

- 구문 분석 오류 추가
- 0으로 나누기 오류 추가

구문 분석 오류 추가

이 실습에서는 구문 분석 오류를 추가합니다.

1. Kinesis Data Analytics 시작하기 연습에 설명된 대로 Kinesis Data Analytics 애플리케이션을 생성합니다.
2. 애플리케이션 세부 정보 페이지에서 스트리밍 데이터 연결을 선택합니다.

3. 시작하기 실습을 수행했으면 계정에 데모 스트림(`kinesis-analytics-demo-stream`)이 있을 것입니다. Connect to source(소스에 연결) 페이지에서 이 데모 스트림을 선택합니다.
4. Kinesis Data Analytics이 데모 스트림에서 샘플을 취하여 생성된 애플리케이션 내 입력 스트림에 대한 스키마를 유추합니다. 콘솔의 [Formatted stream sample] 탭에서 유추된 스키마와 샘플 데이터를 확인할 수 있습니다.
5. 다음으로 스키마를 편집하고 열 유형을 수정하여 구문 분석 오류를 추가합니다. Edit schema(스키마 편집)를 선택합니다.
6. `TICKER_SYMBOL` 열 유형을 `VARCHAR(4)`에서 `INTEGER`로 변경합니다.

   생성된 애플리케이션 내 스키마의 열 유형이 유효하지 않다면, Kinesis Data Analytics이 애플리케이션 내 스트림으로 데이터를 가져올 수 없습니다. 대신 행을 오류 스트림에 전송합니다.
7. [Save schema]를 선택합니다.
8. [Refresh schema samples]를 선택합니다.

   [Formatted stream] 샘플에 행이 없다는 점에 유의하십시오. 그러나 [Error stream] 탭에서는 오류 메시지와 함께 데이터를 보여 줍니다. [Error stream] 탭에서는 애플리케이션 내 오류 스트림으로 전송된 데이터를 보여 줍니다.

   열 데이터 유형을 변경했기 때문에 Kinesis Data Analytics가 애플리케이션 내 입력 스트림에서 데이터를 가져올 수 없었습니다. 대신 데이터를 오류 스트림으로 전송했습니다.

0으로 나누기 오류 추가

이 연습에서는 애플리케이션 코드를 업데이트하여 런타임 오류(0으로 나누기)를 추가합니다. Amazon Kinesis Data Analytics는 당초 결과를 쓰여 질 애플리케이션 내 스트림 대신`DESTINATION_SQL_STREAM` 애플리케이션 내 오류 스트림으로 결과 행을 전송합니다.

1. Kinesis Data Analytics 시작하기 연습에 설명된 대로 Kinesis Data Analytics 애플리케이션을 생성합니다.

   다음과 같이 [Real-time analytics] 탭에서 결과를 확인합니다.

   Sour
2. 애플리케이션 코드에서 `SELECT` 문을 업데이트하여 0으로 나누기를 추가합니다. 예:

```
SELECT STREAM ticker_symbol, sector, change, (price / 0) as ProblemColumn
FROM "SOURCE_SQL_STREAM_001"
```

```
WHERE sector SIMILAR TO '%TECH%';
```

3. 애플리케이션을 실행합니다.

   0으로 나누기 런타임 오류가 발생하기 때문에 결과를 DESTINATION_SQL_STREAM에 작성하지 않고 Kinesis Data Analytics는 행을 애플리케이션 내 오류 스트림으로 전송합니다. Real-time analytics(실시간 분석) 탭에서 오류 스트림을 선택하면 애플리케이션 내 오류 스트림에서 행을 볼 수 있습니다.

예: 솔루션 액셀러레이터

AWS 솔루션 사이트에는 전체 스트리밍 데이터 솔루션을 빠르게 생성하는 데 사용할 수 있는 AWS CloudFormation 템플릿이 있습니다.

다음의 템플릿을 사용할 수 있습니다:

AWS 계정 활동에 대한 실시간 인사이트

이 솔루션은 AWS 계정()에 대한 리소스 액세스 및 사용량 지표를 실시간으로 기록하고 시각화합니다. 자세한 내용은 AWS 계정 활동에 대한 실시간 인사이트를 참조하세요.

Kinesis Data Analytics를 사용한 실시간 AWS IoT 디바이스 모니터링

이 솔루션은 IoT 기기 연결 및 활동 데이터를 실시간으로 수집, 처리, 분석 및 가시화합니다. 자세한 내용은 Kinesis Data Analytics를 사용한 실시간 AWS IoT 디바이스 모니터링을 참조하세요.

Kinesis Data Analytics를 사용한 실시간 웹 Analytics

이 솔루션은 웹 사이트 클릭스트림 데이터를 실시간으로 수집, 처리, 분석 및 가시화합니다. 자세한 설명은 Kinesis Data Analytics을 통한 실시간 웹 분석을 참조하십시오.

Amazon Connected Vehicle 솔루션

이 솔루션은 차량의 IoT 데이터를 실시간으로 수집, 처리, 분석 및 가시화합니다. 자세한 설명은 Amazon Connected Vehicle 솔루션 섹션을 참조하십시오.

Amazon Kinesis Data Analytics의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Kinesis Data Analytics 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내의AWS 서비스를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 데이터의 민감도, 조직의 요건 및 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Kinesis Data Analytics 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 충족하도록 Kinesis Data Analytics를 구성하는 방법을 보여줍니다. 또한 Kinesis Data Analytics 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 Amazon 서비스의 사용 방법을 배우게 됩니다.

주제

- Amazon Kinesis Data Analytics for SQL 애플리케이션에서의 데이터 보호
- Kinesis Data Analytics의 ID 및 액세스 관리
- 에 대한 인증 및 액세스 통제
- Amazon Kinesis Data Analytics 모니터링
- Amazon Kinesis Data Analytics for SQL 애플리케이션을 위한 규정 준수 검증
- Amazon Kinesis Data Analytics에서의 복원성
- SQL 애플리케이션을 위한 Kinesis Data Analytics의 인프라 보안
- Kinesis Data Analytics를 위한 보안 모범 사례

Amazon Kinesis Data Analytics for SQL 애플리케이션에서의 데이터 보호

에서 제공하는 도구를 사용하여 데이터를 보호할 수 있습니다 AWS. Kinesis Data Analytics는 Kinesis Data Streams, Firehose, Amazon S3를 비롯하여 데이터 암호화를 지원하는 서비스와 함께 사용할 수 있습니다.

Kinesis Data Analytics에서의 데이터 암호화

유휴 상태에서의 암호화

다음은 Kinesis Data Analytics를 사용한 유휴 상태에서의 데이터를 암호화하는 데 대한 설명입니다:

- StartStreamEncryption을 사용하여 수신 Kinesis Data Streams에서 데이터를 암호화할 수 있습니다. 자세한 설명은 Kinesis 데이터 스트림용 서버 측 암호화란 무엇인가?를 참조하십시오.
- 출력 데이터는 Firehose를 사용하여 유휴 상태에서 암호화하여 암호화된 Amazon S3 버킷에 데이터를 저장할 수 있습니다. Amazon S3 버킷이 사용하는 암호화 키를 지정할 수 있습니다. 자세한 설명은 KMS 관리형 키(SSE-KMS)와 서버 측 암호화를 사용한 데이터 보호를 참조하십시오.
- 애플리케이션의 코드는 암호화된 상태로 저장됩니다.
- 애플리케이션의 준거 데이터는 암호화된 상태로 저장됩니다.

전송 중 데이터 암호화

Kinesis Data Analytics는 전송 중 모든 데이터를 암호화합니다. 모든 Kinesis Data Analytics 애플리케이션은 전송 중 데이터 암호화가 활성화되고 이를 비활성화할 수 없습니다.

Kinesis Data Analytics는 다음 시나리오에서 전송 중인 데이터를 암호화합니다:

- Kinesis Data Streams에서 Kinesis Data Analytics로 전송 중인 데이터
- Kinesis Data Analytics 내의 내부 구성 요소 간 전송 중 데이터
- Kinesis Data Analytics와 Firehose 간에 전송 중인 데이터입니다.

키 관리

Kinesis Data Analytics의 데이터 암호화는 서비스 관리형 키를 사용합니다. 고객 관리형 키는 지원되지 않습니다.

Kinesis Data Analytics의 ID 및 액세스 관리

Amazon Kinesis Data Analytics에는 애플리케이션 입력 구성에서 지정한 스트리밍 소스에서 레코드를 읽을 수 있는 권한이 필요합니다. Amazon Kinesis Data Analytics에는 애플리케이션 출력 구성에서 지정한 스트림에 애플리케이션 출력을 기록할 권한도 필요합니다.

Amazon Kinesis Data Analytics가 취할 수 있는 IAM 역할을 생성함으로써 이러한 권한을 부여할 수 있습니다. 이 역할에 부여한 권한은 서비스가 권한을 받았을 때 Amazon Kinesis Data Analytics가 수행할 수 있는 작업을 결정합니다.

> **Note**
>
> IAM 역할을 생성하고자 하는 경우 이 섹션의 정보가 유용합니다. Amazon Kinesis Data Analytics 콘솔에서 애플리케이션을 생성하는 경우 콘솔이 여러분을 대신할 IAM 역할을 생성할 수 있습니다. 콘솔은 자신이 생성하는 IAM 역할에 대해 다음의 명명 규칙을 사용합니다:
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> 역할이 생성되면 IAM 콘솔에서 역할을 검토하고 정책을 연결할 수 있습니다.

각 IAM 역할에는 두 가지 정책이 연계되어 있습니다. 신뢰 정책에서는 역할을 위임할 사용자를 지정합니다. 권한 정책에서(하나 이상 존재할 수 있음) 이 역할을 부여하고자 하는 권한을 지정합니다. 다음 섹션은 정책에 대한 설명으로 IAM 역할 생성 시 사용할 수 있습니다.

신뢰 정책

Amazon Kinesis Data Analytics이 스트리밍 또는 참조 소스에 액세스할 수 있도록 권한을 부여하기 위해 IAM 역할에 다음 신뢰 정책을 연계할 수 있습니다:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

권한 정책

IAM 역할을 생성하여 Amazon Kinesis Data Analytics이 애플리케이션의 스트리밍 소스를 읽도록 하게 하려면, 해당 읽기 작업에 대한 권한을 부여해야 합니다. 소스(예: Kinesis 스트림 또는 Firehose 전송 스트림 또는 Amazon S3 버킷의 참조 소스)에 따라 다음과 같은 권한 정책을 연결할 수 있습니다.

Kinesis 스트림을 읽을 수 있는 권한 정책

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:123456789012:stream/inputStreamName"
            ]
        }
    ]
}
```

Firehose 전송 스트림을 읽을 수 있는 권한 정책

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:Get*"
            ],
            "Resource": [
                "arn:aws:firehose:us-
east-1:123456789012:deliverystream/inputFirehoseName"
            ]
        }
    ]
}
```

> **Note**
>
> `firehose:Get*` 권한은 Kinesis Data Analytics가 스트림에 액세스하는 데 사용하는 내부 접근자를 나타냅니다. Firehose 전송 스트림에는 퍼블릭 접근자가 없습니다.

애플리케이션 출력 구성에서 Amazon Kinesis Data Analytics가 외부 목적지에 출력을 작성하도록 지정하는 경우, 다음의 권한을 IAM 역할에 부여해야 합니다.

Kinesis 스트림의 쓰기 권한 정책

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "WriteOutputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:123456789012:stream/output-stream-
name"
            ]
        }
    ]
}
```

Firehose 전송 시스템에 대한 쓰기 권한 정책

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/output-
firehose-name"
            ]
        }
    ]
}
```

Amazon S3 버킷에서 준거 데이터 소스를 읽을 수 있는 권한 정책

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

교차 서비스 혼동된 대리자 방지

에서 AWS교차 서비스 가장은 한 서비스(호출 서비스)가 다른 서비스(호출 서비스)를 호출할 때 발생할 수 있습니다. 호출 서비스는 다른 고객의 리소스에 대해 적절한 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있으며, 이로 인해 대리인 문제가 발생할 수 있습니다.

혼동된 대리자를 방지하기 위해는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 AWS 제공합니다. 이 섹션에서는 Kinesis Data Analytics와 관련된 서비스 간 혼동 방지 기능에 중점을 두고 있지만, IAM 사용자 가이드의 혼동된 대리인 문제 섹션에서 이 주제에 대해 자세히 알아볼 수 있습니다.

Kinesis Data Analytics for SQL의 경우, 역할 신뢰 정책의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 역할에 대한 액세스를 예상 리소스에 의해 생성된 요청으로만 제한하는 것이 좋습니다.

하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 `aws:SourceArn`을(를) 사용하십시오. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`을(를) 사용하십시오.

`aws:SourceArn`의 값은 Kinesis Data Analytics에서 사용하는 리소스의 ARN이어야 하며, 그것은 다음 형식으로 지정됩니다: `arn:aws:kinesisanalytics:region:account:resource`

혼동된 대리인 문제에 대한 권장 접근 방식은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용하는 것입니다.

리소스의 전체 ARN을 모르거나 여러 리소스를 지정한 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자 (*)를 포함한 `aws:SourceArn` 키를 사용하십시오. 예를 들어 `arn:aws:kinesisanalytics::111122223333:*`입니다.

SQL API용 Kinesis Data Analytics에서 CreateApplication, AddApplicationInput 및 DeleteApplication 같은 대부분의 작업은 특정 애플리케이션의 컨텍스트에서 이루어지지만, DiscoverInputSchema 작업은 여하한 애플리케이션 문맥으로도 실행되지 않습니다. 즉, 이 작업에 사용되는 역할이 `SourceArn` 조건 키에 리소스를 완전히 지정해서는 안 됩니다. 다음은 와일드카드 ARN을 사용하는 예입니다:

```
{
   ...
   "ArnLike":{
      "aws:SourceArn":"arn:aws:kinesisanalytics:us-east-1:123456789012:*"
   }
   ...
}
```

Kinesis Data Analytics for SQL에서 생성되는 기본 역할은 이 와일드카드를 사용합니다. 이렇게 하면 콘솔 환경에서 입력 스키마 검색이 원활하게 작동할 수 있습니다. 하지만 스키마를 발견한 후에는 전체 ARN을 사용하도록 신뢰 정책을 편집하여 혼동 부정 행위를 완전히 완화하는 것이 좋습니다.

Kinesis Data Analytics에 제공하는 역할 정책과 사용자에 대해 생성된 역할의 신뢰 정책은 aws:SourceArn 및 aws:SourceAccount 조건 키를 사용할 수 있습니다.

혼동된 대리인 문제를 방지하려면 다음 단계를 수행하십시오:

혼동된 대리인 문제를 방지하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 역할을 선택한 다음 수정하려는 역할을 선택합니다.
3. 신뢰 정책 편집을 선택합니다.
4. 신뢰 정책 편집 페이지에서 기본 JSON 정책을 `aws:SourceArn` 및 `aws:SourceAccount` 글로벌 조건 컨텍스트 키 중 하나 또는 둘 다를 사용하는 정책으로 대체합니다. 다음 정책 예를 참조하십시오:
5. 정책 업데이트를 선택합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"kinesisanalytics.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"Account ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:kinesisanalytics:us-
east-1:123456789012:application/my-app"
            }
         }
      }
   ]
}
```

에 대한 인증 및 액세스 통제

액세스는 자격 증명을 요구합니다. 이러한 자격 증명에는 애플리케이션 또는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스와 같은 AWS 리소스에 액세스할 수 있는 권한이 있어야 합니다. 다음 섹션에서는 AWS Identity and Access Management (IAM)을 사용하여 리소스에 대한 액세스를 보호할 수 있는 방법에 대한 세부 정보를 제공합니다.

액세스 통제

요청을 인증하는 데 유효한 자격 증명이 있더라도 권한이 없다면 리소스를 생성하거나 액세스할 수 없습니다. 예컨대, 애플리케이션을 생성하려면 권한이 있어야 합니다.

다음 섹션에서는 에 대한 권한을 관리하는 방법을 설명합니다. 먼저 개요를 읽어 보면 도움이 됩니다.

- 리소스에 대한 액세스 권한 관리 개요

- 을 위한 정체 기반 정책(IAM 정책) 사용
- API 권한: 작업, 권한 및 리소스 준거

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. , AWS 계정 루트 사용자 IAM 사용자 또는 IAM 역할을 수임하여 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것이 AWS 좋습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명 액세스 시 역할을 AWS 계정수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 및 애플리케이션에서 사용할 수 있도록 자체 ID 소스의 사용자 AWS 계정 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS

CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다(역할을 프록시로 사용하는 대신). 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에

나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

리소스에 대한 액세스 권한 관리 개요

> **Warning**
>
> 새 프로젝트의 경우, SQL 애플리케이션용보다 새로운 Managed Service for Apache Flink Studio를 사용하는 것이 좋습니다. Managed Service for Apache Flink Studio는 사용 편의성과 고급 분석 기능을 결합하여 정교한 스트림 처리 애플리케이션을 몇 분 만에 구축할 수 있도록 합니다.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하십시오:

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따릅니다.

- 보안 인증 공급자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용자 가이드에서 사용자(콘솔)에 권한 추가의 지침을 따르십시오.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하십시오.

주제

- 리소스 및 작업
- 리소스 소유권 이해
- 리소스에 대한 액세스 관리
- 정책 요소 지정: 작업, 효과, 보안 주체
- 정책에서 조건 지정

리소스 및 작업

에서는 기본 리소스가 애플리케이션입니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다.

다음 표에서처럼 이러한 리소스에는 고유한 Amazon 리소스 이름(ARN)이 연계됩니다.

리소스 유형	ARN 형식
Application	`arn:aws:kinesisanalytics:` *`region`*`:`*`account-id`*`:application/`*`application-name`

은 리소스를 처리하기 위한 일련의 작업을 제공합니다. 사용 가능한 작업 목록은 작업 섹션을 참조하십시오.

리소스 소유권 이해

는 리소스를 생성한 사용자에 관계없이 계정에서 생성된 리소스를 AWS 계정 소유합니다. 특히 리소스 소유자는 리소스 생성 요청을 인증하는 AWS 계정 보안 주체 엔터티(즉, 루트 계정, 사용자 또는 IAM 역할)의 입니다. 다음의 예제에서는 이러한 작동 방법을 설명합니다.

- 의 루트 계정 자격 증명을 사용하여 애플리케이션을 AWS 계정 생성하는 경우 AWS 계정 는 리소스의 소유자입니다. (에서는 리소스가 애플리케이션입니다.)

- 에서 사용자를 생성하고 해당 사용자에게 애플리케이션을 생성할 수 있는 권한을 AWS 계정 부여하면 사용자가 애플리케이션을 생성할 수 있습니다. 하지만 사용자가 속한 AWS 계정이 애플리케이션 리소스를 소유합니다. 사용자가 아닌 역할에 권한을 부여하는 것이 좋습니다.
- 에서 애플리케이션을 생성할 권한이 AWS 계정 있는 IAM 역할을 생성하는 경우 해당 역할을 수임할 수 있는 사람은 누구나 애플리케이션을 생성할 수 있습니다. AWS 계정사용자가 속한이 애플리케이션 리소스를 소유합니다.

리소스에 대한 액세스 관리

권한 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 다음 섹션에서는 권한 정책을 만드는 데 사용 가능한 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. 전체 IAM 설명은 IAM 사용자 가이드에서 IAM이란 무엇인가?를 참조하십시오. IAM 정책 구문과 설명에 대한 자세한 설명은 IAM 사용자 가이드의 IAM JSON 정책 준거를 참조하십시오.

IAM ID에 연계된 정책을 정체 기반 정책(IAM 정책)이라고 합니다. 리소스에 연계된 정책은 리소스 기반 정책이라고 합니다. 정체 기반 정책(IAM 정책)만 지원합니다.

주제

- 정체 기반 정책(IAM 정책)
- 리소스 기반 정책

정체 기반 정책(IAM 정책)

정책을 IAM 보안 인증에 연결할 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다:

- 계정 내 사용자 또는 그룹에 권한 정책 연계 – 사용자에게 애플리케이션과 같은 리소스 생성 권한을 부여하려면 사용자 혹은 해당 사용자가 속한 그룹에 허용 정책을 연계할 수 있습니다
- 역할에 권한 정책 연계(교차 계정 권한 부여) – 정체 기반 권한 정책을 IAM 역할에 연계하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어 계정 A의 관리자는 다음과 같이 다른 계정 AWS 계정 (예: 계정 B) 또는 Amazon 서비스에 교차 계정 권한을 부여하는 역할을 생성할 수 있습니다.

1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 권한 정책을 역할에 연결합니다.
2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
3. 계정 B 관리자는 계정 B의 아무 사용자나 역할을 수임할 권한을 위임할 수 있습니다. 그러면 계정 B의 사용자는 계정 A에서 리소스를 생성하거나 액세스할 수 있습니다. Amazon 서비스에 역할 수임 권한을 부여할 경우 신뢰 정책의 보안 주체가 Amazon 서비스 주체가 될 수도 있습니다.

IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 설명은 IAM 사용자 가이드의 액세스 관리 섹션을 참조하십시오.

다음은 애플리케이션 생성에 요구되는 `kinesisanalytics:CreateApplication` 작업에 대해 권한을 부여하는 정책 예입니다.

> **Note**
>
> 다음은 설명을 위한 정책 예입니다. 정책을 사용자에게 연결하면 사용자는 AWS CLI 또는 AWS SDK를 사용하여 애플리케이션을 생성할 수 있습니다. 하지만 입력 및 출력을 구성하기 위해서는 더 많은 권한이 필요합니다. 콘솔을 사용할 때도 더 많은 권한이 필요합니다. 자세한 정보는 이어지는 섹션에서 설명됩니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

정체 기반 정책 사용 방법에 대한 자세한 설명은 을 위한 정체 기반 정책(IAM 정책) 사용 을 참조하십시오. 사용자, 그룹, 역할 및 권한에 대한 자세한 설명은 IAM 사용자 가이드에서 정체(사용자, 그룹 및 역할)를 참조하십시오.

리소스 기반 정책

Amazon S3과 같은 다른 서비스도 리소스 기반 권한 정책을 지원합니다. 예를 들어, 정책을 S3 버킷에 연결하여 해당 버킷에 대한 액세스 권한을 관리할 수 있습니다. 는 리소스 기반 정책을 지원하지 않습니다.

정책 요소 지정: 작업, 효과, 보안 주체

각 리소스에 대해 서비스는 API 작업 세트를 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 에서는 정책에서 지정할 수 있는 작업을 정의합니다. 일부 API 작업에서는 API 작업을 수행하기 위해 복수의 작업에 대한 권한이 필요할 수 있습니다. 리소스 및 API 작업에 대한 자세한 설명은 리소스 및 작업 및 작업 섹션을 참조하십시오.

다음은 가장 기본적인 정책 요소입니다:

- 리소스 – Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 설명은 리소스 및 작업 섹션을 참조하십시오.
- 조치 – 조치 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어, `create`를 사용하여 사용자가 애플리케이션을 생성하도록 권한을 부여할 수 있습니다.
- Effect - 사용자가 특정 작업을 요청할 때 허용할지 아니면 거부할지 그 결과를 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 다른 정책에서 액세스 권한을 부여하는 경우라도 사용자가 해당 리소스에 액세스할 수 없도록 하기 위해 리소스에 대한 권한을 명시적으로 거부할 수도 있습니다.
- 보안 주체 – 정체 기반 정책(IAM 정책)에서 정책이 연계된 사용자는 암묵적인 보안 주체입니다. 리소스 기반 정책의 경우 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당). 의 경우 리소스 기반 정책을 지원하지 않습니다.

IAM 정책 구문 및 설명에 대해 자세히 알아보려면 IAM 사용자 가이드의 IAM JSON 정책 참조 섹션을 참조하십시오.

모든 API 작업과 해당 작업이 적용되는 리소스를 보여주는 목록은 API 권한: 작업, 권한 및 리소스 준거을 참조하십시오.

정책에서 조건 지정

권한을 부여할 때 액세스 정책 언어를 사용하여 조건이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건을 참조하십시오.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. 에만 해당되는 특정한 조건 키는 없습니다. 그러나 필요에 따라 사용할 수 있는 AWS전체 조건 키가 있습니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 조건에 사용 가능한 키를 참조하세요.

을 위한 정체 기반 정책(IAM 정책) 사용

다음 정체 기반 정책의 예는 계정 관리자가 권한 정책을 IAM 정체(즉, 사용자, 그룹 및 역할)에 연계하고 이 과정을 통해 리소스에서 작업을 수행할 권한을 부여할 수 있는 방법을 보여줍니다.

> **Important**
>
> 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명에 대한 소개 주제 부분을 먼저 읽어 보십시오. 자세한 설명은 리소스에 대한 액세스 권한 관리 개요을 참조하십시오.

주제

- 콘솔 사용에 요구되는 권한
- Amazon 관리형 (사전 정의) 정책
- 고객 관리형 정책 예

다음은 권한 정책의 예입니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
```

```
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

이 정책에는 하나의 문이 있습니다.

- 첫 번째 문은 애플리케이션의 Amazon 리소스 이름(ARN)을 사용하여 하나의 작업 (`kinesisanalytics:CreateApplication`)을 리소스에 사용할 수 있는 권한을 부여합니다. 이 경우 ARN은 와일드카드 문자(*)를 사용하여 임의의 리소스에 대해 권한이 부여되었음을 나타냅니다.

모든 API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 API 권한: 작업, 권한 및 리소스 준거을 참조하십시오.

콘솔 사용에 요구되는 권한

이 콘솔로 작업하는 사용자에는 필요한 권한을 부여해야 합니다. 예를 들어, 애플리케이션을 생성할 수 있는 권한을 사용자에게 부여하려면 해당 계정에서 스트리밍 소스를 사용자에게 보여주어 사용자가 콘솔에서 입력 및 출력을 구성할 수 있는 권한을 부여해야 합니다.

다음과 같이 하는 것이 좋습니다:

- Amazon 관리형 정책을 사용한 사용자 권한 부여. 가용 정책에 대해서는 Amazon 관리형 (사전 정의) 정책 을 참조하십시오.
- 맞춤 정책을 생성하십시오. 이 경우, 이 섹션에서 제공한 예를 검토할 것을 권장합니다. 자세한 설명은 고객 관리형 정책 예 섹션을 참조하십시오.

Amazon 관리형 (사전 정의) 정책

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 이러한 Amazon 관리형 정책은 사용자가 필요한 권한을 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 설명은 IAM 사용자 가이드의 Amazon 관리형 정책을 참조하십시오.

계정의 사용자에게 연계할 수 있는 다음의 Amazon 관리형 정책은 다음의 경우에 한합니다:

- **`AmazonKinesisAnalyticsReadOnly`** – 사용자가 애플리케이션을 나열하고 입력/출력 구성을 검토할 수 있는 작업 권한을 부여합니다. 사용자가 Kinesis 스트림 및 Firehose 전송 스트림 목록을 열람할 권한도 부여합니다. 애플리케이션이 실행되면 사용자는 콘솔에서 소스 데이터와 실시간 분석 결과를 볼 수 있습니다.

- **`AmazonKinesisAnalyticsFullAccess`** – 사용자가 애플리케이션을 생성 및 관리하도록 모든 작업 및 기타 모든 권한을 부여합니다. 단, 다음을 참고하십시오:

  - 사용자가 콘솔에서 새로운 IAM 역할을 생성하고자 하는 경우에는 이들 권한만으로 충분하지 않습니다(해당 권한은 사용자가 기존 역할을 선택할 수 있도록 하는 것입니다). 사용자가 콘솔에서 IAM 역할을 생성할 수 있도록 하려면 `IAMFullAccess` Amazon 관리형 정책을 추가합니다.

  - 사용자는 애플리케이션을 구성할 때 IAM 역할을 지정하기 위한 `iam:PassRole` 작업에 대한 권한을 보유해야 합니다. 이 Amazon 관리형 정책은 접두사 `service-role/kinesis-analytics`로 시작하는 IAM 역할에 대해서만 `iam:PassRole` 작업 권한을 사용자에게 부여합니다.

    사용자가 이 접두사가 없는 역할을 가지고 애플리케이션을 구성하는 경우에는 먼저 구체적인 역할에 대한 `iam:PassRole` 작업 권한을 사용자에게 명시적으로 부여해야 합니다.

작업 및 리소스에 대한 권한을 허용하는 고유의 맞춤 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 사용자 또는 그룹에 이러한 맞춤 정책을 연결할 수 있습니다.

고객 관리형 정책 예

이 섹션의 예에서는 사용자에게 연결할 수 있는 샘플 정책 그룹을 제공합니다. 정책을 처음 생성하는 경우 먼저 계정에서 사용자를 생성하는 것이 좋습니다. 그런 다음 이 섹션의 단계에 설명된 대로 사용자에게 정책을 순서대로 연결하십시오. 그런 다음 콘솔을 사용하여 정책을 사용자에게 연결할 때 각 정책의 효과를 확인할 수 있습니다.

처음에 사용자는 권한을 가지고 있지 않으며 콘솔에서 어떠한 작업도 수행할 수 없습니다. 정책을 사용자에게 연결하면 사용자가 콘솔에서 다양한 작업을 수행할 수 있는지 확인할 수 있습니다.

대신 브라우저 창 2개를 사용하는 것이 좋습니다. 하나의 창에서 사용자를 생성하고 권한을 부여하십시오. 다른 쪽에서는 사용자의 자격 증명을 AWS Management Console 사용하여에 로그인하고 권한을 부여할 때 권한을 확인합니다.

애플리케이션을 위한 집행 역할로 사용할 수 있는 IAM 역할을 생성하는 방법을 보여 주는 예는 IAM 사용자 가이드에서 IAM 역할 생성을 참조하십시오.

단계 예

- 단계 1: IAM 사용자 생성
- 단계 2: 한정되지 않은 작업에 대한 사용자 권한 부여
- 단계 3: 애플리케이션 목록 열람 및 세부 정보 열람 권한을 사용자에게 부여
- 단계 4: 특정 애플리케이션을 시작할 수 있는 권한을 사용자에게 부여
- 단계 5: 애플리케이션 생성 권한을 사용자에게 부여
- 단계 6: Lambda 사전 처리 사용 권한을 애플리케이션에 부여

단계 1: IAM 사용자 생성

먼저 사용자를 생성하고 사용자를 관리 권한이 있는 IAM 그룹에 추가한 후 생성한 사용자에게 관리자 권한을 부여해야 합니다. 그런 다음 특수 URL과 해당 사용자의 자격 증명을 AWS 사용하여에 액세스할 수 있습니다.

관련 지침은 IAM 사용자 가이드의 첫 번째 IAM 사용자 및 관리자 그룹 생성을 참조하십시오.

단계 2: 한정되지 않은 작업에 대한 사용자 권한 부여

먼저 사용자가 애플리케이션으로 작업할 때 필요로 하는 에 지정되지 않은 모든 작업에 대한 사용자 권한을 부여합니다. 여기에는 스트림(Amazon Kinesis Data Streams 작업, Amazon Data Firehose 작업)을 사용한 작업 권한과 CloudWatch 작업에 대한 권한이 포함됩니다. 다음 정책을 사용자에게 연계합니다.

`iam:PassRole` 권한을 부여하고자 하는 IAM 역할의 명칭을 제공하여 정책을 업데이트하거나 모든 IAM 역할을 나타내는 와일드카드 문자(*)를 지정해야 합니다. 이것이 확실한 방법은 아니지만, 이 테스트에서 생성된 특정 IAM 역할이 없을 수 있습니다.

JSON

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:CreateStream",
                "kinesis:DeleteStream",
                "kinesis:DescribeStream",
                "kinesis:ListStreams",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:GetLogEvents",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListPolicyVersions",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/service-role/role-name"
        }
    ]
}
```

단계 3: 애플리케이션 목록 열람 및 세부 정보 열람 권한을 사용자에게 부여

다음 정책은 사용자에게 다음의 권한을 부여합니다.

- 사용자가 애플리케이션 목록을 열람할 수 있도록 하는 `kinesisanalytics:ListApplications` 작업에 대한 권한. 이는 서비스 수준 API 호출이기 때문에 `Resource` 값으로 "*"를 지정합니다.
- 임의의 애플리케이션에 대한 정보를 확보할 수 있도록 하는 `kinesisanalytics:DescribeApplication` 작업에 대한 권한.

이 정책을 사용자에게 추가합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:ListApplications"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:DescribeApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/*"
        }
    ]
}
```

사용자 자격 증명으로 콘솔에 로그인하여 이러한 권한을 확인합니다.

단계 4: 특정 애플리케이션을 시작할 수 있는 권한을 사용자에게 부여

사용자가 기존 애플리케이션 중 하나를 시작할 수 있도록 하려면 다음 정책을 사용자에게 연결합니다. 이 정책은 `kinesisanalytics:StartApplication` 작업에 대한 권한을 제공합니다. 계정 ID, AWS 리전 및 애플리케이션 이름을 제공하여 정책을 업데이트해야 합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/application-name"
        }
    ]
}
```

단계 5: 애플리케이션 생성 권한을 사용자에게 부여

사용자가 애플리케이션을 생성하도록 하려면 다음 정책을 사용자에게 연결할 수 있습니다. 사용자가 애플리케이션 이름을 지정할 수 있도록(따라서 여러 애플리케이션을 생성할 수 있도록) 정책을 업데이트하고 AWS 리전, 계정 ID 및 사용자가 생성할 특정 애플리케이션 이름 또는 "*"를 제공해야 합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication",
                "kinesisanalytics:UpdateApplication",
                "kinesisanalytics:AddApplicationInput",
                "kinesisanalytics:AddApplicationOutput"
            ],
            "Resource": "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/application-name"
        }
    ]
}
```

단계 6: Lambda 사전 처리 사용 권한을 애플리케이션에 부여

애플리케이션에서 Lambda 사전 처리를 사용하도록 허용하려면 다음 정책을 역할에 연결합니다.

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

API 권한: 작업, 권한 및 리소스 준거

액세스 통제을 설정하고 IAM ID에 연결할 수 있는 권한 정책(정체 기반 정책)을 작성할 때 다음 목록을 준거로 사용할 수 있습니다. 는 각 API 작업, 작업을 수행할 수 있는 권한을 부여할 수 있는 해당 작업, 권한을 부여할 수 있는 리소스가 포함되어 있습니다. AWS 정책의 `Action`필드에서 작업을 지정하고, 정책의 `Resource`필드에서 리소스 값을 지정합니다.

정책에서 AWS전체 조건 키를 사용하여 조건을 표시할 수 있습니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 키를 참조하세요.

> **Note**
>
> 작업을 지정하려면 `kinesisanalytics` 접두사 다음에 API 작업 명칭을 사용합니다(예: `kinesisanalytics:AddApplicationInput`).

API 및 작업에 대한 필수 권한

API 작업:

필요한 권한(API 작업):

리소스:

API 및 작업에 대한 필수 권한

Amazon RDS API 및 작업에 대한 필수 권한

API 작업:AddApplicationInput

작업: `kinesisanalytics:AddApplicationInput`

리소스:

`arn:aws:kinesisanalytics: region:accountId:application/application-name`

GetApplicationState

콘솔은 `GetApplicationState`라는 내부 메서드를 사용하여 애플리케이션 데이터를 샘플링 또는 액세스합니다. 귀하의 서비스 애플리케이션은 AWS Management Console를 통해 애플리케이션 데이터를 샘플링 또는 액세스하려는 내부 `kinesisanalytics:GetApplicationState` API에 권한을 부여해야 합니다.

Amazon Kinesis Data Analytics 모니터링

Kinesis Data Analytics는 애플리케이션에 대한 모니터링 기능을 제공합니다. 자세한 내용은 모니터링 단원을 참조하십시오.

Amazon Kinesis Data Analytics for SQL 애플리케이션을 위한 규정 준수 검증

타사 감사자는 여러 규정 준수 프로그램의 일환으로 Amazon Kinesis Data Analytics의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위의 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 Amazon 서비스를 참조하세요. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 에서 보고서 다운로드를 참조하세요 AWS Artifact.

Kinesis Data Analytics 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다. Kinesis Data Analytics 사용이 HIPAA 또는 PCI와 같은 표준을 준수해야 하는 경우 다음과 같이 AWS 에서 제공하는 도움말 리소스를 활용하십시오:

- 보안 및 규정 준수 빠른 시작 가이드 -이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공합니다 AWS.
- HIPAA 보안 및 규정 준수를 위한 설계 백서 -이 백서에서는 기업이 AWS 를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS Config -이 AWS 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

Amazon Kinesis Data Analytics에서의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가

이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 외에도 Kinesis Data Analytics는 데이터 복원성과 백업 요구 사항을 지원하는 몇 가지 기능을 제공합니다.

재해 복구

Kinesis Data Analytics는 서버리스 모드에서 실행되며, 자동 마이그레이션을 수행하여 호스트 저하, 가용 영역 가용성 및 기Kinesis Data Analytics타 인프라 관련 문제를 처리합니다. 이 경우 Kinesis Data Analytics는 데이터 손실 없이 애플리케이션이 처리되도록 보장합니다. 자세한 설명은 애플리케이션 출력을 외부 대상에 유지하기 위한 전송 모델 섹션을 참조하십시오.

SQL 애플리케이션을 위한 Kinesis Data Analytics의 인프라 보안

관리형 서비스인 Amazon Kinesis Data Analytics는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 Kinesis Data Analytics에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

Kinesis Data Analytics를 위한 보안 모범 사례

Amazon Kinesis Data Analytics는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주십시오.

IAM 역할을 사용하여 다른 Amazon 서비스에 액세스

Kinesis Data Analytics 애플리케이션에는 Kinesis 데이터 스트림, Firehose 전송 스트림 또는 Amazon S3 버킷 등 다른 서비스의 리소스에 액세스할 수 있는 유효한 자격 증명이 있어야 합니다. 자격 AWS

증명을 애플리케이션 또는 Amazon S3 버킷에 직접 저장해서는 안 됩니다. 이러한 보안 인증은 자동으로 교체되지 않으며 손상된 경우 비즈니스에 큰 영향을 줄 수 있는 장기 보안 인증입니다.

대신 IAM 역할을 사용하여 애플리케이션이 다른 리소스에 액세스 할 수 있도록 임시 자격 증명을 관리해야 합니다. 역할을 사용하면 다른 리소스에 액세스하기 위해 장기 자격 증명을 사용할 필요가 없습니다.

자세한 설명은 IAM 사용자 가이드에서 다음 주제를 참조하십시오:

- IAM 역할
- 역할에 대한 일반적인 시나리오: 사용자, 애플리케이션 및 서비스

종속 리소스에서 서버 측 암호화 구현

저장 데이터와 전송 중 데이터는 Kinesis Data Analytics에서 암호화되며, 이 암호화는 비활성화할 수 없습니다. Kinesis Data Streams, Firehose 전송 시스템, Amazon S3 버킷 등 종속 리소스에서 서버 측 암호화를 구현해야 합니다. 종속 리소스에서 서버 측 암호화 구현에 대한 자세한 설명은 데이터 보호 섹션을 참조하십시오.

CloudTrail을 사용하여 API 직접 호출 모니터링

Kinesis Data Analytics는 Kinesis Data Analytics에서 사용자 AWS CloudTrail, 역할 또는 Amazon 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스와 통합됩니다.

CloudTrail에서 수집한 정보를 사용하여 Kinesis Data Analytics에 온 요청, 요청을 한 IP 주소, 요청을 한 사람, 요청 시간 및 추가 세부 정보를 확인할 수 있습니다.

자세한 내용은 the section called “사용 AWS CloudTrail” 단원을 참조하십시오.

SQL 애플리케이션용 모니터링

모니터링은 와(과) 사용자 애플리케이션의 안정성, 가용성 및 성능을 유지하는 중요한 역할을 합니다. 다중 지점 실패가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분에서 모니터링 데이터를 수집해야 합니다. 하지만 모니터링을 시작하기 전에 다음 질문에 대한 답변을 포함하는 모니터링 계획을 작성해야 합니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

다음 단계에서는 다양한 시간과 다양한 부하 조건에서 성능을 측정하여 환경에서 일반 성능의 기준선을 설정합니다. 을(를) 모니터링하면서 모니터링 데이터를 저장할 수 있습니다. 데이터를 저장하는 경우 현재 성능 데이터와 비교하고, 일반적인 성능 패턴과 성능 이상을 식별하고, 문제를 해결할 방법을 강구할 수 있습니다.

을(를) 사용하여 애플리케이션을 모니터링합니다. 애플리케이션은 데이터 스트림 (입력 또는 출력)을 처리하며, 둘 다 CloudWatch 로그에서 검색 범위를 좁히는 데 사용할 수 있는 ID를 포함합니다. 데이터 스트림 처리 방법에 관한 내용은 Amazon Kinesis Data Analytics for SQL 애플리케이션: 작동 방식 섹션을 참조하십시오.

가장 중요한 지표는 애플리케이션이 스트리밍 소스에서 읽어오는 현재 시간에서 뒤처진 정도를 나타내는 `millisBehindLatest`입니다. 일반적인 경우에 지연 시간이 밀리초 또는 거의 0에 가까운 레벨이어야 합니다. `millisBehindLatest`이 증가한 것처럼 보이는 일시적인 증가는 보편적 현상입니다.

애플리케이션의 스트리밍 소스를 읽는 시점이 한 시간 넘게 지연되는 경우, CloudWatch 경보가 발동되도록 설정하는 것이 좋습니다. 라이브 애플리케이션으로 처리된 데이터를 전송하는 것과 같은 준실시간 처리의 경우 경보를 최저 레벨(예: 5분)으로 설정해야 할 수 있습니다.

주제

- 모니터링 도구
- Amazon CloudWatch를 사용한 모니터링
- 을 사용하여 AWS CloudTrail API 호출 로깅

모니터링 도구

AWS 는를 모니터링하는 데 사용할 수 있는 다양한 도구를 제공합니다. 이들 도구 중에는 모니터링을 자동으로 수행하도록 구성할 수 있는 도구도 있지만, 수동 작업이 필요한 도구도 있습니다. 모니터링 작업은 최대한 자동화하는 것이 좋습니다.

자동 모니터링 도구

다음과 같은 자동 모니터링 도구를 사용하여 를 관찰하고 문제 발생 시 보고할 수 있습니다.

- Amazon CloudWatch 경보 – 지정한 기간 동안 단일 지표를 감시하고, 여러 기간에 대해 지정된 임계값과 관련하여 지표 값을 기준으로 하나 이상의 작업을 수행합니다. 이 작업은 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책에 전송되는 알림입니다. CloudWatch 경보는 특정 상태에 있다는 이유만으로는 작업을 호출하지 않습니다. 상태가 변경되고 지정한 기간 동안 유지되어야 합니다. 자세한 내용은 Amazon CloudWatch를 사용한 모니터링을 참조하십시오.
- Amazon CloudWatch Logs – AWS CloudTrail 또는 기타 소스의 로그 파일을 모니터링, 저장 및 액세스합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 로그 파일 모니터링을 참조하세요.
- Amazon CloudWatch Events – 이벤트를 일치시키고 하나 이상의 대상 함수 또는 스트림으로 라우팅하여 값을 변경하거나 상태 정보를 캡처하거나 수정 작업을 수행합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch Events란 무엇인가요를 참조하세요.
- AWS CloudTrail 로그 모니터링 - 계정 간에 로그 파일을 공유하고, CloudTrail 로그 파일을 CloudWatch Logs로 전송하여 실시간으로 모니터링하고, Java로 로그 처리 애플리케이션을 작성하고, CloudTrail에서 전송한 후 로그 파일이 변경되지 않았는지 확인합니다. 자세한 내용은AWS CloudTrail 사용 설명서의 CloudTrail 로그 파일 작업을 참조하세요.

수동 모니터링 도구

모니터링의 또 한 가지 중요한 부분은 CloudWatch 경보에 포함되지 않는 항목을 수동으로 모니터링해야 한다는 점입니다. , CloudWatch Trusted Advisor및 기타 AWS Management Console 대시보드는 AWS 환경 상태를 at-a-glance 볼 수 있습니다.

- CloudWatch 홈 페이지에 다음이 표시됩니다:
  - 현재 경보 및 상태
  - 경보 및 리소스 그래프
  - 서비스 상태

또한 CloudWatch를 사용하여 다음을 수행할 수 있습니다.

- 맞춤 대시보드를 생성하여 관심 있는 서비스 모니터링
- 지표 데이터를 그래프로 작성하여 문제를 해결하고 추세 파악
- 모든 지표 검색 및 찾아보기
- 문제에 대해 알려주는 경보 생성 및 편집

- AWS Trusted Advisor 를 모니터링하여 성능, 신뢰성, 보안 및 비용 효율성을 개선할 수 있습니다. 모든 사용자에 대해 4가지 Trusted Advisor 검사를 수행할 수 있습니다. Business 또는 Enterprise Support 플랜을 보유한 고객은 사용자에 대해 50개 이상의 검사를 수행할 수 있습니다. 자세한 내용은 AWS Trusted Advisor 단원을 참조하십시오.

Amazon CloudWatch를 사용한 모니터링

Amazon CloudWatch를 사용하여 애플리케이션을 모니터링할 수 있습니다. CloudWatch는 원시 데이터를 수집하여 읽을 수 있고 실시간에 가까운 지표로 처리합니다. 이 통계는 2주 동안 보관됩니다. 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 기본적으로 지표 데이터는 CloudWatch로 자동 전송됩니다. 자세한 설명은 Amazon CloudWatch 사용자 가이드의 Amazon CloudWatch란 무엇인가?를 참조하십시오.

주제

- 지표 및 차원
- 지표 및 차원 보기
- 모니터링용 CloudWatch 경보 생성
- Amazon CloudWatch Logs로 작업하기

지표 및 차원

`AWS/KinesisAnalytics` 네임스페이스에는 다음과 같은 지표가 포함됩니다.

지표	설명
`Bytes`	읽거나(입력 스트림 1개당) 쓴(출력 스트림 1개당) 바이트 수 레벨: 입력 스트림 1개당, 출력 스트림 1개당

지표	설명
KPUs	스트림 처리 애플리케이션 실행에 사용되는 Kinesis 처리 단위(KPU)의 개수. 각 시간 당 사용된 KPU 평균 개수는 애플리케이션 설계를 결정합니다. 레벨: 애플리케이션 레벨
MillisBehindLatest	애플리케이션이 스트리밍 소스에서 읽어오는 현재 시간에서 뒤처진 정도를 나타냅니다. 레벨: 애플리케이션 레벨
Records	읽거나(입력 스트림 1개당) 쓴(출력 스트림 1개당) 레코드 수 레벨: 입력 스트림 1개당, 출력 스트림 1개당
Success	애플리케이션에 대해 구성된 대상으로의 각 전송 성공 시도마다 1이, 모든 실패 전송 시도마다 0입니다. 이 지표의 평균 값은 수행되는 성공 전송 수를 나타냅니다. 레벨: 대상마다.
InputProcessing.Duration	에서 수행한 각 AWS Lambda 함수 호출에 소요된 시간입니다. 레벨: 입력 스트림 1개당
InputProcessing.OkRecords	Ok의 상태로 표시된 Lambda 함수가 반환한 레코드의 수. 레벨: 입력 스트림 1개당
InputProcessing.OkBytes	Ok의 상태로 표시된 Lambda 함수가 반환한 레코드의 바이트 합계. 레벨: 입력 스트림 1개당

지표	설명
`InputProcessing.DroppedRecords`	`Dropped`의 상태로 표시된 Lambda 함수가 반환한 레코드의 수. 레벨: 입력 스트림 1개당
`InputProcessing.ProcessingFailedRecords`	`ProcessingFailed` 의 상태로 표시된 Lambda 함수가 반환한 레코드의 수. 레벨: 입력 스트림 1개당
`InputProcessing.Success`	성공적인 Lambda 간접 호출의 수 레벨: 입력 스트림 1개당
`LambdaDelivery.OkRecords`	`Ok`의 상태로 표시된 Lambda 함수가 반환한 레코드의 수. 레벨: Lambda 목적지별
`LambdaDelivery.DeliveryFailedRecords`	`DeliveryFailed` 의 상태로 표시된 Lambda 함수가 반환한 레코드의 수. 레벨: Lambda 목적지별
`LambdaDelivery.Duration`	Lambda 함수 간접 호출에 걸린 시간. 레벨: Lambda 목적지별

는 다음 차원의 지표를 제공합니다.

차원	설명
`Flow`	입력 스트림 1개당: 입력 출력 스트림 1개당: 출력
`Id`	입력 스트림 1개당: 입력 ID

차원	설명
	출력 스트림 1개당: 출력 ID

지표 및 차원 보기

애플리케이션이 데이터 스트림을 처리할 때 다음 지표와 차원을 CloudWatch로 전송합니다. 다음 절차에 따라 에 대한 지표를 볼 수 있습니다.

콘솔에서 지표는 먼저 서비스 네임스페이스 별로 그룹화된 다음, 각 네임스페이스 내에서 차원 조합 별로 그룹화됩니다.

CloudWatch 콘솔을 사용하여 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 카테고리별 CloudWatch 지표 창에서 지표 카테고리를 선택합니다.
4. 위쪽 창에서 지표의 전체 목록이 보일 때까지 아래로 스크롤합니다.

를 사용하여 지표를 보려면 AWS CLI

- 명령 프롬프트에서 다음 명령을 사용합니다.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

지표는 다음 레벨별로 수집됩니다:

- Application
- 입력 스트림
- 출력 스트림

모니터링용 CloudWatch 경보 생성

경보가 상태를 변경하면 Amazon SNS 메시지를 전송하는 Amazon CloudWatch 경보를 생성할 수 있습니다. 경보는 지정한 기간 동안 단일 지표를 감시합니다. 기간 수에 대한 주어진 임계값과 지표 값을

비교하여 하나 이상의 작업을 수행합니다. 이 작업은 Amazon SNS 주제 또는 Auto Scaling 정책으로 전송되는 알림입니다.

경보는 지속적인 상태 변경에 대해서만 작업을 호출합니다. 작업을 간접 호출하는 CloudWatch 경보의 경우 특정 시간 동안 상태 변경 및 유지가 지속되어야 합니다.

다음 설명과 같이 AWS Management Console, CloudWatch AWS CLI또는 CloudWatch API를 사용하여 경보를 설정할 수 있습니다.

CloudWatch 콘솔을 이용한 경보 설정 방법

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 경보 생성을 선택합니다. Create Alarm Wizard(경보 생성 마법사)가 시작합니다.
3. Kinesis Analytics Metrics(Kinesis 분석 지표)를 선택합니다. 그런 다음 지표를 통해 스크롤하여 경보를 설정하고자 하는 지표를 찾습니다.

   지표를 표시하려면 파일 시스템의 파일 시스템 ID를 검색합니다. 지표를 선택하여 경보를 생성한 후 다음을 선택합니다.
4. 지표에 대한 명칭, 설명 및 다음 경우 항상 값을 입력합니다.
5. 경보 상태 도달 시 CloudWatch에서 이메일을 보내게 하려면, 이 경우 언제든 경보: 필드에서 경보 상태를 선택합니다. 알림 보내기: 필드에서 기존 SNS 주제를 선택합니다. 주제 생성을 선택한 경우 새 이메일 구독 목록에 대한 명칭 및 이메일 주소를 설정할 수 있습니다. 이 목록은 향후 경보를 위해 필드에 저장되고 표시됩니다.

   > **Note**
   >
   > 새 Amazon SNS 주제를 생성하기 위해 주제 생성을 사용할 경우 이메일 주소는 알림을 받기 전에 검증되어야 합니다. 이메일은 경보가 경보 상태에 입력될 때만 전송됩니다. 이러한 경보 상태 변경이 이메일이 검증되기 전에 발생할 경우에는 알림을 받지 못합니다.
6. [Alarm Preview] 영역에서 생성할 경보를 미리 확인할 수 있습니다.
7. [Create Alarm]을 선택하여 경보를 생성합니다.

CloudWatch CLI를 사용하여 경보를 설정하려면

- `mon-put-metric-alarm`을 호출합니다. 자세한 설명은 Amazon CloudWatch CLI 참조를 확인하십시오.

CloudWatch API를 사용하여 경보 설정

- `PutMetricAlarm`을 호출합니다. 자세한 설명은 Amazon CloudWatch API 참조를 참조하십시오.

Amazon CloudWatch Logs로 작업하기

애플리케이션이 잘못 구성되면 애플리케이션이 시작하는 동안 실행 상태로 전환될 수 있습니다. 또는 애플리케이션 내 입력 스트림에 데이터를 업데이트할 수 있으나 처리할 수는 없습니다. CloudWatch 로그 옵션을 애플리케이션 추가하면 애플리케이션 구성 문제를 모니터링 할 수 있습니다.

다음 조건에서 구성 오류를 생성할 수 있습니다:

- 입력에 사용되는 Kinesis 데이터 스트림이 존재하지 않습니다.
- 입력에 사용되는 Amazon Data Firehose 전송 스트림이 존재하지 않습니다.
- 참조 데이터 소스로 사용되는 Amazon S3 버킷이 존재하지 않습니다.
- S3 버킷의 참조 데이터 소스에 지정된 파일이 존재하지 않습니다.
- 관련 권한을 관리하는 AWS Identity and Access Management (IAM) 역할에 올바른 리소스가 정의되어 있지 않습니다.
- 관련 권한을 관리하는 IAM 역할에 정확한 권한이 정의되어 있지 않습니다.
- 관련 권한을 관리하는 IAM 역할을 수행할 권한이 없습니다.

Amazon CloudWatch에 대한 자세한 설명은 Amazon CloudWatch 사용자 가이드를 참조하십시오.

PutLogEvents 정책 작업 추가

CloudWatch에 잘못된 구성 오류를 쓸 권한이 필요합니다. 다음 설명과 같이 IAM 역할의 권한을 추가할 수 있습니다. IAM 역할에 대한 자세한 설명은 Kinesis Data Analytics의 ID 및 액세스 관리 섹션을 참조하십시오.

신뢰 정책

IAM 역할에 대한 권한을 부여하려면 다음의 신뢰 정책을 해당 역할에 부가합니다.

JSON

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

권한 정책

리소스에서 CloudWatch에 로그 이벤트 작성 권한을 애플리케이션에 부여하려면 다음 IAM 권한 정책을 사용합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt0123456789000",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:log-
stream:my-log-stream*"
            ]
        }
    ]
}
```

구성 오류 모니터링 추가

다음 API 작업을 통해 CloudWatch 로그 옵션을 신규 또는 기존 애플리케이션에 추가하거나 기존 애플리케이션에 대한 로그 옵션을 변경합니다.

> **Note**
>
> 현재, API 작업을 사용하면 한 개의 CloudWatch 로그 옵션을 애플리케이션에 추가할 수 있습니다. 콘솔을 통해서는 CloudWatch 로그 옵션을 추가할 수 없습니다.

애플리케이션 생성 시 CloudWatch 로그 옵션 추가

다음 코드 예는 애플리케이션을 생성 시 `CreateApplication` 작업을 사용하여 CloudWatch 로그 옵션을 추가하는 방법을 보여 줍니다. `Create_Application`에 대한 자세한 설명은 CreateApplication 섹션을 참조하십시오.

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

기존 애플리케이션에 CloudWatch 로그 옵션 추가

다음 코드 예는 `AddApplicationCloudWatchLoggingOption` 작업을 사용하여 기존 애플리케이션에 CloudWatch 로그 옵션을 추가하는 방법을 보여 줍니다. `AddApplicationCloudWatchLoggingOption`에 대한 자세한 정보는 AddApplicationCloudWatchLoggingOption 섹션을 참조하세요.

```
{
   "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>
}
```

기존 CloudWatch 로그 옵션 업데이트

다음 코드 예는 `UpdateApplication` 작업을 사용하여 기존 CloudWatch 로그 옵션을 수정하는 방법을 보여 줍니다. `UpdateApplication`에 대한 자세한 정보는 UpdateApplication 섹션을 참조하세요.

```
{
   "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>
}
```

애플리케이션에서 CloudWatch 로그 옵션 삭제

다음 코드 예는 `DeleteApplicationCloudWatchLoggingOption` 작업을 사용하여 기존 CloudWatch 로그 옵션을 삭제하는 방법을 보여 줍니다. `DeleteApplicationCloudWatchLoggingOption`에 대한 자세한 정보는 DeleteApplicationCloudWatchLoggingOption 섹션을 참조하세요.

```
{
   "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>
}
```

구성 오류

다음 섹션에서는 잘못 구성된 애플리케이션으로 생기는 오류를 Amazon CloudWatch Logs에서 볼 수 있는 방법을 자세히 알아보겠습니다.

오류 메시지 형식

애플리케이션 구성 오류에 의해 생성되는 오류 메시지는 다음과 같은 형식으로 표시됩니다.

```
{
"applicationARN": "string",
"applicationVersionId": integer,
"messageType": "ERROR",
"message": "string",
"inputId": "string",
"referenceId": "string",
"errorCode": "string"
"messageSchemaVersion": "integer",
}
```

오류 메시지의 필드에는 다음 정보가 포함됩니다.

- `applicationARN`: 애플리케이션 생성을 위한 Amazon 리소스 이름(ARN)(예: `arn:aws:kinesisanalytics:us-east-1:112233445566:application/sampleApp`)
- `applicationVersionId`: 오류 발생 시점의 애플리케이션 버전 자세한 설명은 ApplicationDetail 섹션을 참조하십시오.
- `messageType`: 메시지 유형. 현재 메시지 유형은 `ERROR`가 유일합니다.
- `message`: 오류의 상세 정보. 예를 들면 다음과 같습니다.

  ```
  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`: 애플리케이션 입력과 관련된 ID. 이 값은 해당 입력이 오류의 원인인 경우에만 나타납니다. `referenceId`가 존재하는 경우에는 이 값이 나타나지 않습니다. 자세한 설명은 DescribeApplication 섹션을 참조하십시오.
- `referenceId`: 애플리케이션 참조 데이터 소스와 관련된 ID. 이 값은 해당 소스가 오류의 원인인 경우에만 나타납니다. `inputId`가 존재하는 경우에는 이 값이 나타나지 않습니다. 자세한 설명은 DescribeApplication 섹션을 참조하십시오.

- `errorCode`: 오류의 식별자. 이 ID는 `InputError` 와 `ReferenceDataError` 중에 하나입니다.
- `messageSchemaVersion`: 현재 스키마 버전을 지정하는 값(현재는 1). 이 값을 통해 오류 메시지 스키마가 업데이트되었는지 확인할 수 있습니다.

오류

다음을 포함해 CloudWatch 로그에 나타날 수도 있는 오류.

리소스가 존재하지 않음

존재하지 않는 Kinesis 입력 스트림에 대해 ARN이 지정되어 있지만 ARN이 구문상 정확한 경우 다음과 같은 오류가 생성됩니다.

```
{
"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"
}
```

참조 데이터에 부정확한 Amazon S3 파일 키가 사용된 경우 다음과 같은 오류가 생성됩니다.

```
{
 "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"
}
```

역할이 존재하지 않음

존재하지 않는 IAM 입력 역할에 대해 ARN이 지정되어 있지만 ARN이 구문상 정확한 경우 다음과 같은 오류가 생성됩니다.

```
{
 "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"
}
```

역할이 리소스에 액세스할 수 있는 권한을 보유하지 않음

Kinesis 소스 스트림과 같은 입력 리소스에 액세스할 권한을 가지고 있지 않은 입력 역할을 사용하는 경우 다음과 같은 오류가 생성됩니다.

```
{
 "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"
}
```

을 사용하여 AWS CloudTrail API 호출 로깅

는에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 콘솔로부터의 직접 호출과 API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 이벤트를 포

함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 설명은 AWS CloudTrail 사용자 가이드를 참조하십시오.

CloudTrail의 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. 에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 작업은 CloudTrail에서 로깅되고 API 참조에 기록됩니다. 예컨대, `CreateApplication` 및 `UpdateApplication` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 AWS 계정 루트 사용자 또는 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하십시오.

로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 AddApplicationCloudWatchLoggingOption 작업과 DescribeApplication 작업을 보여 주는 CloudTrail 로그 항목의 예입니다.

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

한도

단종 날짜

신중한 고려 끝에 Amazon Kinesis Data Analytics for SQL 애플리케이션을 단종하기로 결정했습니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 계획하고 마이그레이션하는 데 도움이 되도록 15개월에 걸쳐 오퍼링을 점진적으로 단종할 예정입니다. 알아두어야 할 중요한 두 날짜는 2025년 10월 15일과 2026년 1월 27일입니다.

1. 2025년 10월 15일에 고객의 애플리케이션이 중지되고 `READY` 상태로 전환됩니다. 이때 애플리케이션을 다시 시작하여 애플리케이션을 정상적으로 계속 사용할 수 있지만 서비스 제한이 적용됩니다.
2. 2025년 10월 15일부터는 새 Amazon Kinesis Data Analytics for SQL 애플리케이션을 생성할 수 없습니다. 기존 애플리케이션은 정상적으로 실행할 수 있지만 서비스 제한이 적용됩니다.
3. 2026년 1월 27일부터 애플리케이션이 삭제됩니다. Amazon Kinesis Data Analytics for SQL 애플리케이션을 시작하거나 작동할 수 없게 됩니다. 그 시점부터 Amazon Kinesis Data Analytics for SQL 애플리케이션에 대한 지원을 더 이상 이용할 수 없습니다.

2025년 10월 15일 이전에 애플리케이션을 Amazon Managed Service for Apache Flink 또는 Amazon Managed Service for Apache Flink Studio로 마이그레이션하는 것이 좋습니다. 마이그레이션에 도움이 되는 리소스는 Managed Service for Apache Flink Studio로 마이그레이션하는 예 섹션을 참조하세요. Amazon Managed Service for Apache Flink 또는 Amazon Managed Service for Apache Flink Studio에 대한 자세한 내용은 Amazon Managed Service for Apache Flink 개발자 안내서를 참조하세요.

Limits

Amazon Kinesis Data Analytics for SQL 애플리케이션 작업 시에 다음 한도에 유의하십시오:

- SQL용 Kinesis Data Analytics는 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 캐나다(중부), 유럽(파리), 유럽(아일랜드), 유럽(프랑크푸르트), 유럽(런던), 아시아 태평양(홍콩), 아시아 태평양(뭄바이), 아시아 태평양(시드니), 아시아 태평양(싱가포르), 아시아 태평양(서울), 아시아 태평양(도쿄), 남아메리카(상파울루), AWS GovCloud(미국 동부), AWS GovCloud우드(미국 서부) AWS 리전에서 사용할 수 있습니다. SQL용 Kinesis Data Analytics를 추가 AWS 리전으로 시작할 계획은 없습니다.

- 2023년 6월 28일 이후에는 아직 Kinesis Data Analytics for SQL을 사용하지 않는 경우 AWS 관리 콘솔을 사용하여 새 Kinesis Data Analytics for SQL 애플리케이션을 생성할 수 없습니다. Kinesis Data Analytics for SQL의 단종 날짜에 대한 자세한 내용은 단종 날짜 섹션을 참조하세요. 2023년 6월 28일 이전에 Kinesis Data Analytics for SQL 애플리케이션을 생성한 경우 Kinesis Data Analytics for SQL을 이미 사용하고 있는 AWS 리전에서 애플리케이션을 생성하고 실행하는 방법에는 변경 사항이 없습니다. 그러나 Kinesis Data Analytics for SQL을 사용하지 않는 리전에서는 AWS 콘솔을 사용하여 새 애플리케이션을 더 이상 생성할 수 없습니다.
- 2023년 9월 12일 이후에는 Kinesis Data Analytics for SQL의 기존 사용자가 아닌 경우 Kinesis Data Firehose를 소스로 사용하여 새 애플리케이션을 생성할 수 없습니다. Kinesis Data Analytics for SQL의 단종 날짜에 대한 자세한 내용은 단종 날짜 섹션을 참조하세요. `KinesisFirehoseInput`과 함께 Kinesis Data Analytics for SQL 애플리케이션을 사용하는 기존 고객은 Kinesis Data Analytics를 사용하여 기존 계정 내에서 `KinesisFirehoseInput`와 함께 애플리케이션을 계속 추가할 수 있습니다. 기존 고객이 Kinesis Data Analytics for SQL 애플리케이션과 `KinesisFirehoseInput`을 사용하여 새 계정을 생성하려는 경우 지원 사례를 개설할 수 있습니다. 자세한 정보는 AWS Support Center 섹션을 참조하십시오.
- 애플리케이션 내 스트림에서 행의 크기는 512KB로 제한됩니다. Kinesis Data Analytics는 메타데이터 저장에 최대 1KB를 사용합니다. 이 메타데이터는 행 한도에 계수됩니다. 스트리밍 소스의 레코드 크기가 50KB보다 큰 경우 행 구분 기호를 통해 입력 구성에 적절한 스키마를 제공하여 레코드를 애플리케이션 스트림의 여러 행으로 분할할 수 있습니다.
- 애플리케이션의 SQL 코드는 크기가 100KB로 제한됩니다.
- 윈도우 모드 쿼리에서는 1시간 이내의 윈도우를 사용하는 것이 좋습니다. 애플리케이션 내 스트림은 휘발성 스토리지에 저장되며, 예기치 않은 애플리케이션 중단 시 휘발성 스토리지의 소스 데이터에서 스트림이 다시 빌드됩니다.
- 단일 애플리케이션 내 스트림에 권장되는 최대 처리량은 애플리케이션 쿼리의 복잡성에 따라 2~20MB/초입니다.
- 계정에서 AWS 리전당 최대 50개의 Kinesis Data Analytics 애플리케이션을 생성할 수 있습니다. 서비스 한도 상향 양식을 통해 추가 애플리케이션 요청을 할 수 있습니다. 자세한 정보는 AWS Support Center 섹션을 참조하십시오.
- 단일 Kinesis Data Analytics for SQL 애플리케이션으로 처리할 수 있는 최대 스트리밍 처리량은 약 100MB/초입니다. 이는 애플리케이션 내의 수를 최대 64로 올리고 KPU 한도를 8 이상으로 올렸다는 가정에 근거합니다(자세한 설명은 다음의 한도 참조). 애플리케이션이 100MB/초를 초과하여 입력을 처리해야 하는 경우 다음 중 하나를 수행하십시오:
  - 복수의 Kinesis Data Analytics for SQL 애플리케이션을 사용하여 입력 처리
  - 단일 스트림 및 애플리케이션을 계속하여 사용하려면 Java 애플리케이션용 Managed Service for Apache Flink를 사용하십시오.

> **Note**
>
> 애플리케이션의 예상 입력 처리량이 초당 100MB를 초과하는 경우, 여러 SQL 애플리케이션을 사용할 계획을 미리 세우거나 Amazon Managed Service for Apache Flink for Java 애플리케이션으로 마이그레이션할 수 있도록 애플리케이션의 `InputProcessing.OkBytes` 지표를 정기적으로 검토하세요. 또한 애플리케이션이 입력 처리량 `InputProcessing.OkBytes` 한도에 가까워지면 알림을 받을 수 있도록 CloudWatch 경보를 생성하는 것이 좋습니다. 이는 애플리케이션 쿼리를 업데이트하여 처리량을 늘려주므로 분석의 역압과 지연을 방지할 수 있으므로 유용할 수 있습니다. 자세한 설명은 문제 해결을 참조하십시오. 업스트림의 처리량을 줄이는 메커니즘이 있는 경우에도 경보가 유용할 수 있습니다.

- Kinesis 처리 단위(KPU) 수는 8로 제한됩니다. 이 한도 증가 요청 지침은 Amazon 서비스 한도에서 한도 상향 요청하기를 참조하십시오.

  Kinesis Data Analytics에서는 사용한 만큼만 지불하면 됩니다. 스트림 처리 애플리케이션을 실행하는 데 사용되는 KPU의 평균 개수에 따라 시간당 비용이 청구됩니다. 단일 KPU는 1개의 vCPU 및 4GB의 메모리를 제공합니다.
- 각 애플리케이션은 스트리밍 소스 하나와 참조 데이터 소스 하나를 가질 수 있습니다.
- Kinesis Data Analytics 애플리케이션에 대해 최대 3개의 목적지를 구성할 수 있습니다. 이들 대상 중 하나를 사용하여 애플리케이션 내 오류 스트림 데이터를 유지할 것을 권장합니다.
- 참조 데이터를 저장하는 Amazon S3 객체의 크기는 최대 1GB가 될 수 있습니다.
- 참조 데이터를 애플리케이션 내 테이블에 업로드한 후 S3 버킷에 저장된 참조 데이터를 변경하는 경우 작업을 사용하여UpdateApplication(API 또는 사용 AWS CLI) 애플리케이션 내 테이블의 데이터를 새로 고쳐야 합니다. 현재 AWS Management Console 는 애플리케이션에서 참조 데이터 새로 고침을 지원하지 않습니다.
- 현재, Kinesis Data Analytics는 Amazon Kinesis Producer Library(KPL)에 의해 생성된 데이터를 지원하지 않습니다.
- 애플리케이션당 최대 50개의 태그를 지정할 수 있습니다.

모범 사례

이 섹션에서는 Amazon Kinesis Data Analytics 애플리케이션 작업의 모범 사례를 설명합니다.

주제

- 애플리케이션 관리
- Scaling 애플리케이션
- 애플리케이션 모니터링
- 입력 스키마 정의
- 출력 연결
- 애플리케이션 코드 작성
- 애플리케이션 테스트

애플리케이션 관리

Amazon Kinesis Data Analytics 애플리케이션을 관리할 때 다음 모범 사례를 따르십시오:

- Amazon CloudWatch 경보 설정 – Kinesis Data Analytics에서 제공하는 CloudWatch 지표를 사용하여 다음을 모니터링할 수 있습니다:
  - 입력 바이트 및 입력 레코드(애플리케이션으로 들어오는 바이트 및 레코드의 수)
  - 입력 바이트 및 출력 레코드
  - `MillisBehindLatest` (애플리케이션이 스트리밍 소스에서 읽어오는 시간이 뒤처진 정도를 나타냅니다)

  프로덕션 내 애플리케이션 용 지표에 대해 둘 이상의 경보를 설정하는 것이 좋습니다:

  - `MillisBehindLatest` – 대부분의 경우, 애플리케이션이 최신 데이터보다 한 시간 뒤쳐진 경우 평균 1분 간격으로 경보가 발동되도록 설정하는 것이 좋습니다. 종단 간 처리 필요성이 낮은 애플리케이션의 경우, 허용치를 더 적게 조정할 수 있습니다. 이 경보는 애플리케이션이 최신 데이터를 읽고 있는지 확인하는 데 도움이 됩니다.

- `ReadProvisionedThroughputException` 예외를 피하기 위해 동일한 Kinesis 데이터 스트림을 읽는 프로덕션 애플리케이션의 수를 2개로 제한하십시오.

> **Note**
>
> 이 경우, 애플리케이션은 스트리밍 소스로부터 읽을 수 있는 모든 애플리케이션을 가리킵니다. Kinesis Data Firehose 전송 스트림은 Firehose 전송 스트림에서 읽을 수 있습니다. 그러나 많은 애플리케이션이 Kinesis Data Analytics 애플리케이션 또는 같은 Kinesis 데이터 스트림에서 읽을 수 있습니다 AWS Lambda. 권장 애플리케이션 한도는 스트리밍 소스로부터 읽도록 구성하는 모든 애플리케이션을 참고합니다.

Amazon Kinesis Data Analytics가 스트리밍 소스로부터 읽는 속도는 애플리케이션당 약 1회/초입니다. 그러나 뒤쳐진 애플리케이션의 경우 따라잡기 위해 더 빨리 읽을 수 있습니다. 애플리케이션이 따라잡을 수 있는 적절한 처리량을 허용하기 위해 동일한 데이터 소스를 읽는 애플리케이션 수를 제한하십시오.

- 동일한 Firehose 전송 스트림으로부터 읽는 프로덕션 애플리케이션의 수를 하나로 제한하세요.

  Firehose 전송 스트림은 Amazon S3 및 Amazon Redshift와 같은 대상에 작성할 수 있습니다. 또한 Kinesis Data Analytics 애플리케이션의 스트리밍 소스가 될 수 있습니다. 따라서 Firehose 전송 스트림당 Kinesis Data Analytics 애플리케이션 수를 하나 이하로 구성할 것을 권장합니다. 이렇게 하면 전송 스트림을 다른 대상으로도 전송할 수 있습니다.

Scaling 애플리케이션

입력 애플리케이션 내 스트림 수를 사전에 기본값(1)에서 증가시켜 향후 조정 니즈에 맞게 애플리케이션을 설정하십시오. 애플리케이션의 처리량에 따라 다음의 언어 선택을 권장합니다:

- 애플리케이션이 100MB/초 이상으로 조정해야 할 경우 복수의 스트림과 SQL 애플리케이션용 Kinesis Data Analytics를 사용하십시오.
- 단일 스트림과 애플리케이션을 사용하려는 경우 Managed Service for Apache Flink Applications를 사용하십시오.

> **Note**
> 애플리케이션의 예상 입력 처리량이 100MB/초를 초과하는 경우 여러 SQL 애플리케이션을 사용할 계획을 미리 계획하거나 Managed-flink/latest/java/로 마이그레이션할 수 있도록 애플리케이션의 `InputProcessing.OkBytes` 메트릭을 정기적으로 검토하는 것이 좋습니다.

애플리케이션 모니터링

애플리케이션이 입력 처리량 `InputProcessing.OkBytes` 한도에 가까워지면 알림을 받을 수 있도록 CloudWatch 경보를 생성하는 것이 좋습니다. 이는 애플리케이션 쿼리를 업데이트하여 처리량을 늘려주므로 분석의 역압과 지연을 방지할 수 있으므로 유용할 수 있습니다. 자세한 설명은 문제 해결을 참조하십시오. 업스트림에서 처리량을 줄이는 메커니즘이 있는 경우에도 유용할 수 있습니다.

- 단일 애플리케이션 내 스트림에 권장되는 최대 처리량은 애플리케이션 쿼리의 복잡성에 따라 2~20MB/초입니다.
- 단일 Kinesis Data Analytics for SQL 애플리케이션으로 처리할 수 있는 최대 스트리밍 처리량은 약 100MB/초입니다. 이는 애플리케이션 내 스트림 수를 최대값인 64개로 늘리고 KPU 한도를 8개 이상으로 늘렸다고 가정합니다. 자세한 설명은 한도를 참조하십시오.

> **Note**
> 애플리케이션의 예상 입력 처리량이 100MB/초를 초과하는 경우 여러 SQL 애플리케이션을 사용할 계획을 미리 계획하거나 Managed-flink/latest/java/로 마이그레이션할 수 있도록 애플리케이션의 `InputProcessing.OkBytes` 메트릭을 정기적으로 검토하는 것이 좋습니다.

입력 스키마 정의

콘솔에서 애플리케이션 입력을 구성할 때 먼저 스트리밍 소스를 지정합니다. 그런 다음 콘솔이 검색 API(DiscoverInputSchema 참조)를 통해 스트리밍 소스에서 레코드를 샘플링하여 스키마를 유추합니다. 특히 스키마는 산출된 애플리케이션 내 스트림에 있는 열의 데이터 유형을 정의합니다. 콘솔이 스키마를 표시합니다. 유추된 스키마로 다음 작업을 수행하는 것이 좋습니다.

- 유추된 스키마를 적절히 테스트합니다. 검색 프로세스는 스트리밍 소스의 레코드 샘플만 사용하여 스키마를 유추합니다. 스트리밍 소스의 레코드 유형이 다수인 경우, 검색 API가 하나 이상의 레코드

유형의 샘플링을 놓칠 가능성이 있습니다. 이러한 상황은 스키마가 스트리밍 소스상 데이터를 정확히 반영하지 못하는 결과를 초래할 수 있습니다.

애플리케이션이 시작될 때 이러한 레코드 유형이 누락되면 구문 분석 오류가 발생할 수 있습니다. Amazon Kinesis Data Analytics는 이러한 레코드를 애플리케이션 내 오류 스트림으로 보냅니다. 이러한 파싱 오류를 줄이기 위해 유추된 스키마를 콘솔에서 양방향으로 테스트하고 누락된 레코드에 대해 애플리케이션 내 스트림을 모니터링할 것을 권장합니다.

- Kinesis Data Analytics API는 입력 구성에서 열에 대한 `NOT NULL` 제약 지정을 지원하지 않습니다. 애플리케이션 내 스트림에서 열에 대한 `NOT NULL` 제약을 원하는 경우 애플리케이션 코드를 사용하여 이를 애플리케이션 내 스트림에 생성해야 합니다. 그런 다음 하나의 애플리케이션 내 스트림에서 다른 애플리케이션 내 스트림으로 데이터를 복사하면 제약이 적용됩니다.

  값이 필요할 때 `NULL` 값이 있는 행을 삽입하려고 하면 오류가 발생합니다. Kinesis Data Analytics는 이러한 오류를 애플리케이션 내 오류 스트림으로 보냅니다.

- 검색 프로세스에 의해 유추된 데이터 유형을 완화합니다. 검색 프로세스는 스트리밍 소스에서의 무작위 레코드 샘플링을 바탕으로 열 및 데이터 유형을 추천합니다. 이를 신중히 검토하여 입력에 있는 레코드의 가능한 모든 경우를 망라할 수 있도록 이들 데이터 유형의 완화를 고려할 것을 권장합니다. 이렇게 하면 애플리케이션 전반에 걸쳐 실행 중에 파싱 오류의 발생을 줄일 수 있습니다. 예를 들어, 유추된 스키마의 열 유형이 `SMALLINT` 인 경우 `INTEGER`로 변경하는 것을 고려할 수 있습니다.

- 애플리케이션 코드에서 SQL 함수를 사용하여 구성되지 않은 임의의 데이터 도는 열을 처리합니다. 입력에 로그 데이터와 같이 구성되지 않은 데이터 또는 열이 있을 수 있습니다. 예를 보려면 예: DateTime 값 변환 섹션을 참조하십시오. 이러한 유형의 데이터를 처리하는 접근 방식 중 하나는 `VARCHAR(N)` 한 가지 유형의 열로만 스키마를 정의하는 것입니다. 여기에서 `N`는 스트림에서 나타날 것으로 예상되는 행 중 가장 큰 것입니다. 그런 다음 애플리케이션 코드에서 수신 레코드를 읽습니다. `String` 그리고 `Date Time` 함수를 사용하여 원시 데이터를 구문 분석하고 스키마로 변환합니다.

- 두 개 수준을 초과하는 중첩을 포함하는 스트리밍 소스 데이터를 완전히 처리하고 있는지 확인합니다. 소스 데이터가 JSON인 경우 중첩을 가질 수 있습니다. 검색 API가 하나의 수준 중첩을 평면화하는 스키마를 유추합니다. 두 개 수준의 중첩의 경우에도 검색 API가 평면화를 시도합니다. 두 개 수준을 초과하는 중첩의 경우 평면화 지원에 제한이 있습니다. 중첩을 온전히 처리하기 위해서는 유추

된 스키마를 필요에 맞게 직접 수정해야 합니다. 다음 전략 중 하나를 사용하여 이를 수행하면 됩니다.

- JSON 행 경로를 사용하여 애플리케이션에 요구되는 키 값만 선택적으로 가져옵니다. A JSON 행 경로는 애플리케이션으로 가져오고자 하는 특정 키 값 페어에 대한 포인터를 제공합니다. 모든 수준의 중첩에 대해 이를 수행할 수 있습니다.
- JSON 행 경로를 사용하여 복잡한 JSON 객체를 선택적으로 가져온 다음 애플리케이션 코드에서 문자열 조작 함수를 사용하여 필요로 하는 특정 데이터를 가져옵니다.

출력 연결

모든 애플리케이션이 둘 이상의 출력을 가지는 것이 좋습니다:

- 첫 번째 대상을 사용하여 SQL 쿼리의 결과를 삽입합니다.
- 두 번째 대상을 사용하여 전체 오류 스트림을 삽입하고 Firehose 전송 스트림을 통해 S3 버킷으로 전송합니다.

애플리케이션 코드 작성

다음과 같이 하는 것이 좋습니다:

- SQL 문에서 시간 기반 창을 1시간 이상으로 지정하지 마십시오. 그 이유는 다음과 같습니다.
  - 때때로 애플리케이션 업데이트 또는 Kinesis Data Analytics 내부 이유로 애플리케이션을 다시 시작해야 합니다. 애플리케이션을 다시 시작하는 경우 해당 창에 포함된 모든 데이터를 스트리밍 데이터 소스로부터 다시 읽어야 합니다. 이 때문에 Kinesis Data Analytics가 해당 창에 출력하는 데 시간이 소요됩니다.
  - Kinesis Data Analytics는 해당 기간 동안의 애플리케이션 상태에 관한 모든 요소(관련 데이터 포함)를 유지해야 합니다. 이 경우 상당한 Kinesis Data Analytics 처리 단위가 소모됩니다.
- 개발 시에 결과를 보다 빨리 확인할 수 있도록 SQL 문에서 창 크기를 작게 유지하십시오. 애플리케이션을 프로덕션 환경에 배포할 때 창 크기를 적절히 설정할 수 있습니다.
- 복잡한 단일 SQL 문을 복수의 문으로 나누고 각 단계에서 결과를 중간 애플리케이션 내 스트림에 저장하는 것을 고려할 수 있습니다. 이렇게 하면 디버깅을 신속하게 수행할 수 있습니다.

- 텀블링 창을 사용할 때 두 개의 창을 사용할 것을 권장합니다. 하나는 처리 시간, 다른 하나는 논리 시간 (수집 시간 또는 이벤트 시간) 창입니다. 자세한 설명은 타임스탬프와 ROWTIME 열 섹션을 참조하십시오.

애플리케이션 테스트

Kinesis Data Analytics 애플리케이션의 스키마 또는 애플리케이션 코드를 변경할 경우, 변경 사항을 프로덕션에 배포하기 전에 테스트 애플리케이션을 사용하여 변경 사항을 확인하는 것이 좋습니다.

테스트 애플리케이션 설정

콘솔을 통해 또는 AWS CloudFormation 템플릿을 사용하여 테스트 애플리케이션을 설정할 수 있습니다. AWS CloudFormation 템플릿을 사용하면 테스트 애플리케이션과 라이브 애플리케이션에 대한 코드 변경 사항이 일관성을 유지하는 데 도움이 됩니다.

테스트 애플리케이션을 설정할 때 애플리케이션을 라이브 데이터에 연결하거나 테스트 대상이 될 모의 데이터로 스트림을 채울 수 있습니다. 다음 두 가지 방법으로 모의 데이터로 스트림을 채울 수 있습니다.

- Kinesis 데이터 제너레이터(KDG)를 사용합니다. KDG는 데이터 템플릿을 사용하여 임의 데이터를 Kinesis 스트림에 보냅니다. KDG는 사용이 간편하지만, 데이터 핫스팟이나 이상을 탐지하는 애플리케이션의 경우와 같이 데이터 항목 사이의 복잡한 관계를 테스트하는 데 적합하지 않습니다.
- 맞춤 Python 애플리케이션을 사용하여 더 복잡한 데이터를 데이터 스트림으로 보냅니다. Python 애플리케이션은 핫스팟이나 이상과 같은 데이터 항목들 사이의 복잡한 관계를 생성할 수 있습니다. 클러스터링된 데이터를 데이터 핫스팟으로 보내는 Python 애플리케이션의 예는 예: 스트림에서 핫스팟 감지(HOTSPOTS 함수)를 참조하십시오.

테스트 애플리케이션을 실행할 때 콘솔의 애플리케이션 내 스트림을 확인하는 대신 대상(Amazon Redshift 데이터베이스로 향하는 Firehose 전송 스트림)을 사용하여 결과를 모니터링합니다. 콘솔에 표시되는 데이터는 스트림의 샘플링이며 레코드를 모두 포함하지는 않습니다.

스키마 변경 사항 테스트

애플리케이션의 입력 스트림 스키마를 변경할 때 테스트 애플리케이션을 사용하여 다음 사항이 사실인지 확인합니다.

- 스트림의 데이터가 올바른 데이터 유형으로 강제 변환되고 있습니다. 예를 들어, 날짜/시간 데이터가 문자열로 애플리케이션에 수집되지 않도록 해야 합니다.
- 데이터는 원하는 데이터 유형으로 구문 분석 및 수집되고 있습니다. 분석 또는 강제 변환 오류가 발생할 경우, 콘솔에서 이러한 오류를 확인하거나 대상을 오류 스트림에 할당하고 대상 스토어의 오류를 확인할 수 있습니다.
- 문자 데이터의 데이터 필드는 길이가 충분하며, 애플리케이션은 문자 데이터를 잘라내지 않습니다. 대상 스토어의 데이터 레코드를 점검하여 애플리케이션 데이터가 잘리지 않음을 점검할 수 있습니다.

코드 변경 사항 테스트

SQL 코드의 변경 사항 테스트에는 애플리케이션의 도메인 지식이 필요합니다. 어떤 출력을 테스트해야 하는지와 어떤 것이 올바른 출력인지 판단할 수 있어야 합니다. 애플리케이션의 SQL 코드 수정 시 확인해야 할 문제 가능성이 있는 영역에 대해서는 Amazon Kinesis Data Analytics for SQL 애플리케이션 문제 해결를 참조하십시오.

Amazon Kinesis Data Analytics for SQL 애플리케이션 문제 해결

다음은 Amazon Kinesis Data Analytics for SQL 애플리케이션 사용 시 발생하는 문제를 해결하는 데 도움이 됩니다.

주제

- 중지된 애플리케이션
- SQL 코드를 실행할 수 없음
- 내 스키마를 탐지 또는 검색할 수 없음
- 참조 데이터가 만료된 경우
- 애플리케이션이 대상에 쓰지 않음
- 모니터링해야 할 중요한 애플리케이션 상태 확인 파라미터
- 애플리케이션 실행 시 잘못된 코드 오류
- 애플리케이션이 오류 스트림에 오류라고 기록함
- 불충분한 처리량 또는 높은 MillisBehindLatest

중지된 애플리케이션

- 중지된 SQL용 애플리케이션용 Kinesis Data Analytics란 무엇인가?

  중지된 애플리케이션이란 최소 3개월 동안 어떠한 기록도 처리하지 않은 것으로 관찰된 애플리케이션을 말합니다. 이는 고객이 사용하지 않는 SQL 리소스용 Kinesis Data Analytics에 대한 비용을 지불한다는 것입니다

- 유휴 애플리케이션 중지는 언제 AWS 시작하나요?

  AWS 는 2023년 11월 14일에 유휴 애플리케이션을 중지하기 시작하여 2023년 11월 21일까지 완료될 예정입니다. 해당 리전의 근무 시간 시간 대에 유휴 애플리케이션을 중단할 것입니다.

- 중지된 Kinesis Data Analytics for SQL 애플리케이션을 다시 사용할 수 있는가?

  예. 애플리케이션을 다시 시작해야 하는 경우 평소대로 다시 시작할 수 있습니다. 지원 티켓을 사용하실 필요는 없습니다.

- 가 유휴 애플리케이션을 AWS 중지하면 내 쿼리 결과도 삭제되나요?

아니요. 첫째, 애플리케이션이 유휴 상태이기 때문에 쿼리 작업을 하지 않습니다. 둘째, 쿼리 결과는 SQL용 Kinesis Data Analytics에 저장되지 않습니다. 계산 결과가 전송되는 목적지 싱크 (즉, Amazon S3 또는 다른 데이터 스트림 내부)와 Kinesis Data Analytics for SQL 애플리케이션을 구성하십시오. 그렇게 하면 데이터에 대한 전체 소유권은 사용자에게 있으며 해당 스토리지 서비스 약관에 따라 데이터를 검색할 수 있습니다.

- 내 애플리케이션이 중지되지 않게 하려면 어떻게 해야 하는가?

  서비스 팀 (kda-sql-questions@amazon.com) 에 이메일을 보내 2023년 11월 10일 이전에 애플리케이션이 중지되지 않도록 요청하면 됩니다. 이메일에는 계정 ID와 애플리케이션 ARN이 포함되어야 합니다.

SQL 코드를 실행할 수 없음

Kinesis Data Analytics 사용 시, 특정 SQL 문이 올바르게 실행되기를 원하시면 여러가지 리소스를 사용할 수 있습니다:

- SQL 문에 대한 자세한 설명은 Kinesis Data Analytics for SQL 예 섹션을 참조하십시오. 이 섹션에서는 활용 가능한 복수의 SQL 예를 제공합니다.
- Amazon Kinesis Data Analytics SQL 참조에는 상세한 스트리밍 SQL 문 작성 지침이 들어 있습니다.
- 여전히 문제가 발생하면 Kinesis Data Analytics 포럼에 질문을 남겨 주십시오.

내 스키마를 탐지 또는 검색할 수 없음

가끔 Kinesis Data Analytics가 스키마를 감지하지 못하거나 검색하지 못하는 경우가 있습니다. 대부분의 경우, 그래도 Kinesis Data Analytics를 사용할 수 있습니다.

구분 기호를 사용하지 않는 UTF-8 인코딩 데이터나 CSV(쉼표로 분리된 값) 이외의 형식을 사용하는 데이터가 있거나, 검색 API가 스키마를 검색하지 못하는 경우를 가정해 보겠습니다. 이 경우 수작업을 통해 또는 문자열 조작 함수를 사용하여 데이터를 구성함으로써 스키마를 정의할 수 있습니다.

스트림에 대한 스키마를 검색하기 위해 Kinesis Data Analytics가 스트림에서 최신 데이터를 무작위로 샘플링합니다. 스트림에 데이터를 지속적으로 전송하지 않으면 Kinesis Data Analytics가 샘플을 검색하지 못하거나 스키마를 감지하지 못할 수 있습니다. 자세한 설명은 스트리밍 데이터에 대해 스키마 검색 기능 사용 섹션을 참조하십시오.

참조 데이터가 만료된 경우

애플리케이션이 시작 또는 업데이트 시 또는 서비스 문제로 애플리케이션 중단 중에 Amazon Simple Storage Service (Amazon S3) 객체의 참조 데이터를 애플리케이션으로 로드합니다.

기본 Amazon S3 객체에 업데이트가 이루어지면 애플리케이션에 참조 데이터가 로드되지 않습니다.

애플리케이션의 참조 데이터가 최신 상태가 아닌 경우 다음 단계를 수행하여 데이터를 다시 로드할 수 있습니다.

1. Kinesis Data Analytics console 콘솔에서 목록에 있는 애플리케이션 명칭을 선택한 다음 애플리케이션 세부 정보를 선택합니다.
2. Go to SQL editor(SQL 편집기로 이동)를 선택하여 애플리케이션에 대한 Real-time analytics(실시간 분석) 페이지를 엽니다.
3. Source Data(소스 데이터) 보기에서 참조 데이터 표 명칭을 선택합니다.
4. 작업, Synchronize reference data table(참조 데이터 표 동기화)을 선택합니다.

애플리케이션이 대상에 쓰지 않음

데이터가 대상에 기록되지 않는 경우 다음 사항을 확인하십시오,

- 애플리케이션의 역할이 대상에 액세스할 수 있는 충분한 권한을 가지고 있는지 확인합니다. 자세한 설명은 Kinesis 스트림의 쓰기 권한 정책 또는 Firehose 전송 시스템에 대한 쓰기 권한 정책 섹션을 참조하십시오.
- 애플리케이션 대상이 올바르게 구성되고 애플리케이션이 출력 스트림의 올바른 명칭을 사용하고 있는지 확인합니다.
- 출력 스트림의 Amazon CloudWatch 지표를 점검하여 데이터가 기록되고 있는지 확인합니다. CloudWatch 지표 사용 방법에 관한 자세한 설명은 Amazon CloudWatch를 사용한 모니터링을 참조하십시오.
- the section called “AddApplicationCloudWatchLoggingOption”를 사용하여 CloudWatch 로그 스트림을 추가합니다. 애플리케이션은 구성 오류를 로그 스트림에 작성합니다.

역할 및 대상 구성이 올바르게 보이는 경우 `LAST_STOPPED_POINT`에 InputStartingPositionConfiguration를 지정하여 애플리케이션을 다시 시작해 봅니다.

모니터링해야 할 중요한 애플리케이션 상태 확인 파라미터

애플리케이션이 올바르게 실행됨을 확인하기 위해 특정한 중요 파라미터를 모니터링할 것을 권장합니다.

모니터링할 가장 중요한 파라미터는 Amazon CloudWatch 지표 `MillisBehindLatest`입니다. 이 지표는 스트림으로부터 읽는 현재 시점이 얼마나 뒤쳐져 있는지를 나타냅니다. 이 지표는 소스 스트림으로부터 오는 레코드를 충분히 빨리 처리하고 있는지 여부를 판단하는 데 도움이 됩니다.

일반적 규칙은 한 시간 이상 뒤쳐지면 CloudWatch 알람이 생기도록 설정해야 합니다. 단, 시간의 양은 용례에 의존합니다. 필요에 따라 조정할 수 있습니다.

자세한 설명은 모범 사례 섹션을 참조하십시오.

애플리케이션 실행 시 잘못된 코드 오류

Amazon Kinesis Data Analytics 애플리케이션 SQL 코드 저장 및 실행이 되지 않는다면 통상 다음과 같은 이유로 인한 것입니다:

- SQL 코드의 스트림이 재정의된 경우 – 스트림 및 스트림과 관련된 펌프를 생성하면 코드에서 동일한 스트림을 재정의할 수 없습니다. 스트림 생성에 대한 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조에 있는 스트림 생성을 참조하십시오. 펌프 생성에 대한 자세한 설명은 펌프 생성을 참조하십시오.
- GROUP BY 절에서 복수의 ROWTIME 열을 사용하는 경우 – GROUP BY 절에 하나의 ROWTIME 열만 지정할 수 있습니다. 자세한 설명은 Amazon Kinesis Data Analytics SQL 참조의 GROUP BY 및 ROWTIME을 참조하십시오.
- 하나 이상의 데이터 유형이 잘못된 캐스팅을 가지고 있는 경우 – 이 경우는 코드에 잘못된 암묵적 캐스팅이 들어있는 것입니다. 예를 들어, 코드에서 `timestamp`를 `bigint`로 변환하고 있을 수 있습니다.
- 스트림이 서비스가 예약한 스트림 명칭과 동일한 명칭을 가진 경우 – 스트림은 서비스가 예약한 스트림 `error_stream` 같은 명칭을 가질 수 없습니다.

애플리케이션이 오류 스트림에 오류라고 기록함

애플리케이션이 오류를 애플리케이션 내 오류 스트림에 기록하는 경우 표준 라이브러리를 사용하여 `DATA_ROW` 필드의 값을 디코딩할 수 있습니다. 오류 스트림에 대한 자세한 설명은 오류 처리 섹션을 참조하십시오.

불충분한 처리량 또는 높은 MillisBehindLatest

애플리케이션의 MillisBehindLatest 지표가 꾸준히 증가하거나 지속적으로 1000(1초)보다 높으면 다음과 같은 이유 때문일 수 있습니다.

- 애플리케이션의 InputBytes CloudWatch 지표를 확인하십시오. 초당 4MB 이상으로 수집 중이면 `MillisBehindLatest`가 증가할 수 있습니다. 애플리케이션의 처리량을 개선하려면 `InputParallelism` 파라미터의 값을 높이십시오. 자세한 설명은 입력 스트림 병렬화를 통한 처리량 증대 섹션을 참조하십시오.
- 애플리케이션의 출력 전송 성공 지표에서 대상으로 전송 실패 여부를 확인합니다. 출력을 올바르게 구성했고 출력 스트림에 충분한 용량이 있는지 확인합니다.
- 애플리케이션이 사전 처리를 위해 또는 출력으로 AWS Lambda 함수를 사용하는 경우 애플리케이션의 InputProcessing.Duration 또는 LambdaDelivery.Duration CloudWatch 지표를 확인합니다. Lambda 함수 간접 호출 시간이 5초 이상 걸리면 다음 작업을 고려해 보십시오:
  - Lambda 함수의 메모리 할당을 높입니다. 이는 AWS Lambda 콘솔의 구성 페이지에 있는 기본 설정에서 수행할 수 있습니다. 자세한 설명은 AWS Lambda 개발자 가이드의 Lambda 함수 구성을 참조하십시오.
  - 애플리케이션의 입력 스트림에서 샤드 수를 늘립니다. 이렇게 하면 애플리케이션에서 호출될 병렬 함수 수가 증가되어 처리량이 증가될 수 있습니다.
  - 함수가 외부 리소스에 대한 동기 요청 등 성능에 영향을 미치는 호출을 차단하고 있지 않은지 확인합니다.
  - AWS Lambda 함수를 검사하여 성능을 개선할 수 있는 다른 영역이 있는지 확인합니다. 애플리케이션 Lambda 함수의 CloudWatch 로그를 확인하십시오. 자세한 설명은AWS Lambda 개발자 가이드의 Amazon CloudWatch 지표 액세스를 참조하십시오.
- 애플리케이션이 Kinesis Processing Units (KPU)의 기본 한도를 넘어섰는지 확인합니다. 애플리케이션이 이 한도에 도달하면 한도 증가를 요청할 수 있습니다. 자세한 설명은 처리량 증가를 위해 애플리케이션 용량을 자동으로 확장 또는 축소 섹션을 참조하십시오.
- KPU 한도를 늘린 후에도 애플리케이션에 여전히 문제가 있는 경우 애플리케이션의 입력 처리량이 초당 100MB를 초과하지 않는지 확인하십시오. 100MB/초를 초과하는 경우 Kinesis Data Analytics Sql 애플리케이션이 읽는 데이터 소스로 전송되는 데이터의 양을 줄이는 등 애플리케이션 안정화를 위해 전체 처리량을 줄이는 변경을 하는 것이 좋습니다. 또한 애플리케이션의 병렬성을 높이고, 계산 시간을 단축하고, 열 기반 데이터 유형을 VARCHAR에서 크기가 더 작은 데이터 유형 (예: INTEGER, LONG 등) 으로 변경하고, 샘플링 또는 필터링으로 처리되는 데이터를 줄이는 등의 다른 접근 방식을 권장합니다.

> **Note**
>
> 애플리케이션의 예상 입력 처리량이 100MB/초를 초과하는 경우, 여러 SQL 애플리케이션을 사용할 계획을 미리 계획하거나 Managed-flink/latest/java/로 마이그레이션할 수 있도록 애플리케이션의 `InputProcessing.OkBytes` 메트릭을 정기적으로 검토하는 것이 좋습니다.

Kinesis Data Analytics SQL 참조

Kinesis Data Analytics에서 지원되는 SQL 언어 요소에 대한 자세한 설명은 Kinesis Data Analytics SQL 참조를 참고하십시오.

API 참조

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Managed Service for Apache Flink API V2 설명서를 참조하십시오.

를 사용하여 Amazon Kinesis Data Analytics API를 탐색 AWS CLI 할 수 있습니다. 이 가이드는 AWS CLI을(를) 사용하는 Amazon Kinesis Data Analytics for SQL 애플리케이션 시작하기 연습을 제공합니다.

주제

- 작업
- 데이터 타입

작업

다음 작업이 지원됩니다.

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- CreateApplication
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource
- DescribeApplication

- DiscoverInputSchema
- ListApplications
- ListTagsForResource
- StartApplication
- StopApplication
- TagResource
- UntagResource
- UpdateApplication

AddApplicationCloudWatchLoggingOption

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

CloudWatch 로그 스트림을 추가하여 애플리케이션 구성 오류를 모니터링 합니다. Amazon Kinesis Analytics 애플리케이션에서 CloudWatch 로그 스트림을 사용하는 방법에 대한 자세한 설명은 Amazon CloudWatch Logs 사용을 참조하십시오.

구문 요청

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "string",
      "RoleARN": "string"
   },
   "CurrentApplicationVersionId": number
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

Kinesis Analytics 애플리케이션 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CloudWatchLoggingOption

CloudWatch 로그 스트림 Amazon 리소스 이름(ARN) 과 IAM 역할 ARN을 제공합니다. 참고: 애플리케이션 메시지를 CloudWatch에 쓰려면 사용하는 IAM 역할에는 `PutLogEvents` 정책 작업이 활성화되어 있어야 합니다.

타입: CloudWatchLoggingOption객체

필수 여부: 예

CurrentApplicationVersionId

Kinesis Analytics 애플리케이션의 버전 ID.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

AddApplicationInput

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

Amazon Kinesis 애플리케이션에 스트리밍 소스를 추가합니다. 개념 정보는 애플리케이션 입력 구성을 참조하십시오.

애플리케이션을 생성할 때 스트리밍 소스를 추가하거나 애플리케이션을 생성한 후 이 작업을 사용하여 스트리밍 소스를 추가할 수 있습니다. 자세한 설명은 CreateApplication을 참조하십시오.

이 작업을 사용한 스트리밍 소스 추가를 포함하여 모든 구성 업데이트 시 새로운 버전의 애플리케이션이 생성됩니다. DescribeApplication 작업을 사용하여 현재 애플리케이션 버전을 찾을 수 있습니다.

이 작업에는 `kinesisanalytics:AddApplicationInput` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "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"
   }
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

스트리밍 소스를 추가하려는 기존 Amazon Kinesis Analytics 애플리케이션의 명칭입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

Amazon Kinesis Analytics 애플리케이션의 최신 버전입니다. DescribeApplication 작업을 사용하여 현재 애플리케이션 버전을 찾을 수 있습니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

Input

추가할 입력.

타입: Input객체

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

CodeValidationException

사용자가 제공한 애플리케이션 코드 (쿼리) 가 유효하지 않습니다. 이는 단순한 구문 오류일 수 있습니다.

HTTP 상태 코드: 400

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

AddApplicationInputProcessingConfiguration

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

애플리케이션에 InputProcessingConfiguration을 추가합니다. 입력 프로세서는 애플리케이션의 SQL 코드가 실행되기 전에 입력 스트림에서 레코드를 전처리합니다. 현재 사용할 수 있는 유일한 입력 프로세서는 AWS Lambda입니다.

구문 요청

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   }
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

입력 구성을 추가할 애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

입력 구성을 추가할 애플리케이션의 버전. DescribeApplication 작업을 사용하여 최신 애플리케이션 버전을 찾을 수 있습니다. 지정된 버전이 최신 버전이 아닌 경우 `ConcurrentModificationException`가 반환됩니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

InputId

입력 처리 구성을 추가할 입력 구성의 ID. DescribeApplication 작업을 사용하여 애플리케이션의 입력 ID 목록을 가져올 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

InputProcessingConfiguration

애플리케이션에 추가할 InputProcessingConfiguration.

타입: InputProcessingConfiguration객체

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

AddApplicationOutput

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

Amazon Kinesis Analytics 애플리케이션에 외부 대상을 추가합니다.

Amazon Kinesis Analytics가 애플리케이션 내의 인애플리케이션 스트림에서 외부 대상(예: Amazon Kinesis 스트림, Amazon Kinesis Firehose 전송 스트림 또는 AWS Lambda 함수)으로 데이터를 전송하도록 하려면이 작업을 사용하여 애플리케이션에 관련 구성을 추가합니다. 애플리케이션에 대해 출력을 하나 이상 구성할 수 있습니다. 각 출력 구성은 애플리케이션 내 스트림과 외부 대상을 매핑합니다.

오류를 분석할 수 있도록 출력 구성 중 하나를 사용하여 애플리케이션 내 오류 스트림에서 외부 대상으로 데이터를 전송할 수 있습니다. 자세한 설명은 애플리케이션 출력(목적지) 이해 섹션을 참조하십시오.

이 작업을 사용한 스트리밍 소스 추가를 포함하여 모든 구성 업데이트 시 새로운 버전의 애플리케이션이 생성됩니다. DescribeApplication 작업을 사용하여 현재 애플리케이션 버전을 찾을 수 있습니다.

구성할 수 있는 애플리케이션 입력 및 출력 수 제한은 제한을 참조하십시오.

이 작업에는 `kinesisanalytics:AddApplicationOutput` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "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"
    }
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

출력 구성을 추가할 애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

출력 구성을 추가할 애플리케이션의 버전. DescribeApplication 작업을 사용하여 최신 애플리케이션 버전을 찾을 수 있습니다. 지정된 버전이 최신 버전이 아닌 경우 `ConcurrentModificationException`가 반환됩니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

Output

각각 출력 구성 하나를 설명하는 객체의 배열입니다. 출력 구성에서 인애플리케이션 스트림의 이름, 대상(즉, Amazon Kinesis 스트림, Amazon Kinesis Firehose 전송 스트림 또는 AWS Lambda 함수)을 지정하고 대상에 쓸 때 사용할 형식을 기록합니다.

타입: Output객체

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

AddApplicationReferenceDataSource

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

기존 애플리케이션에 준거 데이터 소스를 추가합니다.

Amazon Kinesis Analytics는 준거 데이터(즉, Amazon S3 객체)를 읽고 애플리케이션 내에 애플리케이션 내 표를 만듭니다. 요청에서 소스(S3 버킷 명칭과 객체 키 명칭), 생성할 애플리케이션 내 표의 명칭, Amazon S3 객체의 데이터가 결과 애플리케이션 내 표의 열에 매핑하는 방법을 설명하는 필요한 매핑 정보를 제공합니다.

개념 정보는 애플리케이션 입력 구성을 참조하십시오. 애플리케이션에 추가할 수 있는 데이터 소스에 대한 한도는 한도를 참조하십시오.

이 작업에는 `kinesisanalytics:AddApplicationOutput` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "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"
    }
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

기존 애플리케이션의 명칭입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

준거 데이터 소스가 추가되는 애플리케이션의 버전입니다. DescribeApplication 작업을 사용하여 최신 애플리케이션 버전을 받을 수 있습니다. 지정된 버전이 최신 버전이 아닌 경우 `ConcurrentModificationException`가 반환됩니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

ReferenceDataSource

준거 데이터 소스는 Amazon S3 버킷의 객체일 수 있습니다. Amazon Kinesis Analytics는 객체를 읽고 생성된 애플리케이션 내 표에 데이터를 복사합니다. S3 버킷, 객체 키 명칭, 생성된 애플리케이션 내 표를 제공합니다. Amazon Kinesis Analytics가 사용자를 대신해 S3 버킷에서 객체를 읽기 위해 맡을 수 있는 IAM 역할도 제공해야 합니다.

타입: ReferenceDataSource객체

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateApplication

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

Amazon Kinesis Analytics 애플리케이션을 생성합니다. 스트리밍 소스 하나를 입력으로 사용하고, 애플리케이션 코드를 입력으로 사용하고, Amazon Kinesis Analytics에서 애플리케이션의 출력 데이터를 기록할 대상을 최대 3개까지 설정하여 각 애플리케이션을 구성할 수 있습니다. 개관을 위해 작동 방식을 참조하십시오.

입력 구성에서 스트리밍 소스를 애플리케이션 내 스트림에 매핑합니다. 그것은 지속적으로 업데이트되는 표로 간주할 수 있습니다. 매핑에서 애플리케이션 내 스트림에 대한 스키마를 제공하고 애플리케이션 내 스트림에 있는 각 데이터 열을 스트리밍 소스의 데이터 요소에 매핑합니다.

애플리케이션 코드는 입력 데이터를 읽고, 해당 데이터를 변환하고, 출력을 생성하는 하나 이상의 SQL 문입니다. 애플리케이션 코드는 SQL 스트림 또는 펌프와 같은 하나 이상의 SQL 아티팩트를 생성할 수 있습니다.

출력 구성에서는 애플리케이션에서 만든 애플리케이션 내 스트림의 데이터를 최대 세 개의 대상에 쓰도록 애플리케이션을 구성할 수 있습니다.

귀하의 소스 스트림으로부터 데이터를 읽거나 대상 스트림에 데이터를 쓰려면 Amazon Kinesis Analytics는 귀하의 권한을 필요로 합니다. 귀하는 IAM 역할을 생성함으로써 그러한 권한을 부여할 수 있습니다. 이 작업에는 `kinesisanalytics:CreateApplication` 조치를 수행할 권한이 요구됩니다.

Amazon Kinesis Analytics 애플리케이션을 생성하기 위한 입문 연습은 시작하기를 참조하십시오.

구문 요청

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

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationCode

입력 데이터를 읽고, 해당 데이터를 변환하고, 출력을 생성하는 하나 이상의 SQL 문입니다. 예컨대, 한 애플리케이션 내 스트림에서 데이터를 읽고 공급업체의 광고 클릭 수의 이동 평균을 생성하는 SQL 문을 쓰고, 펌프를 사용하여 다른 애플리케이션 내 스트림에 결과 행을 삽입할 수 있습니다. 일반적인 패턴에 대한 자세한 설명은 애플리케이션 코드를 참조하십시오.

한 문의 출력을 다음 문의 입력으로 사용할 수 있는 일련의 SQL 문을 제공할 수 있습니다. 애플리케이션 내 스트림과 펌프를 만들어 중간 결과를 저장합니다.

애플리케이션 코드는 `Outputs`에 지정된 명칭으로 스트림을 만들어야 합니다. 예컨대, `Outputs`이 `ExampleOutputStream1` 및 `ExampleOutputStream2`라는 출력 스트림을 정의하면 애플리케이션 코드는 이러한 스트림을 만들어야 합니다.

유형: 문자열

길이 제한: 최소 길이는 0. 최대 길이 102400.

필수 여부: 아니요

ApplicationDescription

애플리케이션의 요약 설명.

유형: 문자열

길이 제약 조건: 최소 길이는 0입니다. 최대 길이 1,024.

필수 여부: 아니요

ApplicationName

Amazon Kinesis Analytics 애플리케이션의 명칭 (예: `sample-app`).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CloudWatchLoggingOptions

이 파라미터를 사용하여 CloudWatch 로그 스트림을 애플리케이션 구성 오류를 모니터링합니다. 자세한 설명은 Amazon CloudWatch Logs 사용을 참조하십시오.

타입: CloudWatchLoggingOption객체 배열

필수: 아니요

Inputs

이 파라미터를 사용하여 애플리케이션 입력을 구성합니다.

단일 스트리밍 소스로부터 입력을 수신하도록 애플리케이션을 구성할 수 있습니다. 이 구성에서는 이 스트리밍 소스를 생성된 애플리케이션 내 스트림에 매핑합니다. 그러면 애플리케이션 코드가 애플리케이션 내 스트림을 표처럼 쿼리할 수 있습니다(지속적으로 업데이트되는 표로 간주할 수 있음).

스트리밍 소스의 경우 Amazon 리소스 이름(ARN)과 스트림의 데이터 형식(예: JSON, CSV 등)을 제공합니다. Amazon Kinesis Analytics가 사용자를 대신해 이 스트림을 읽기 위해 맡을 수 있는 IAM 역할도 제공해야 합니다.

애플리케이션 내 스트림을 생성하기 위해서는 SQL에 사용되는 스키마화된 버전으로 데이터를 변환할 스키마를 지정해야 합니다. 스키마에서, 스트리밍 소스의 데이터 요소를 인앱 스트림의 레코드 열에 매핑합니다.

타입: Input객체 배열

필수: 아니요

Outputs

애플리케이션 내 스트림의 데이터를 최대 세 개의 대상에 쓰도록 애플리케이션 출력을 구성할 수 있습니다.

이러한 대상은 Amazon Kinesis 스트림, Amazon Kinesis Firehose 전송 AWS 스트림, Lambda 목적지 또는 이 세 가지의 조합일 수 있습니다.

구성에서 애플리케이션 내 스트림 명칭, 대상 스트림 또는 Lambda 함수 Amazon Resource Name (ARN), 데이터를 쓸 때 사용할 형식을 지정합니다. 또한 Amazon Kinesis Analytics가 귀하를 대신하여 대상 스트림 또는 Lambda 함수에 작성하기 위해 취할 수 있는 IAM 역할도 제공해야 합니다.

출력 구성에서는 출력 스트림 또는 Lambda 함수 ARN도 제공합니다. 스트림 대상의 경우 스트림의 데이터 형식 (예: JSON, CSV) 을 제공합니다. 또한 Amazon Kinesis Analytics가 귀하를 대신하여 스트림 또는 Lambda 함수에 작성하기 위해 취할 수 있는 IAM 역할도 제공해야 합니다.

타입: Output객체 배열

필수: 아니요

Tags

애플리케이션에 할당할 하나 이상의 태그 목록입니다. 태그는 애플리케이션을 식별하는 키값 페어입니다. 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다. 자세한 설명은 태그 사용하기를 참조하십시오

타입: Tag객체 배열

어레이 멤버: 최소 항목 수 1개. 최대 항목 수 200.

필수 여부: 아니요

응답 구문

```
{
   "ApplicationSummary": {
      "ApplicationARN": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string"
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

ApplicationSummary

귀하의 `CreateApplication` 요청에 대응하여, Amazon Kinesis Analytics는 애플리케이션 Amazon 리소스 이름 (ARN), 명칭 및 상태를 포함하여 생성한 애플리케이션의 요약이 포함된 응답을 반환합니다.

타입: ApplicationSummary 객체

오류

CodeValidationException

사용자가 제공한 애플리케이션 코드 (쿼리) 가 유효하지 않습니다. 이는 단순한 구문 오류일 수 있습니다.

HTTP 상태 코드: 400

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

LimitExceededException

허용된 애플리케이션 수를 초과했습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

TooManyTagsException

애플리케이션에 너무 많은 태그 또는 너무 많은 태그가 추가된 상태로 애플리케이션이 생성되었습니다. 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteApplication

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

지정된 애플리케이션을 삭제합니다. Amazon Kinesis Analytics는 애플리케이션 실행을 중단하고 애플리케이션 아티팩트 (예: 애플리케이션 내 스트림, 참조 애플리케이션 내, 애플리케이션 코드) 를 비롯한 애플리케이션을 삭제합니다.

이 작업에는 `kinesisanalytics:DeleteApplication` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "ApplicationName": "string",
   "CreateTimestamp": number
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

삭제할 Amazon Kinesis Analytics 애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CreateTimestamp

`DescribeApplication` 작업으로 이 값을 얻을 수 있습니다.

유형: 타임스탬프

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteApplicationCloudWatchLoggingOption

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

애플리케이션에서 CloudWatch 로그 스트림을 삭제합니다. Amazon Kinesis Analytics 애플리케이션에서 CloudWatch 로그 스트림을 사용하는 방법에 대한 자세한 설명은 Amazon CloudWatch Logs 사용을 참조하십시오.

구문 요청

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOptionId": "string",
   "CurrentApplicationVersionId": number
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

Kinesis Analytics 애플리케이션 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CloudWatchLoggingOptionId

삭제할 CloudWatch 로깅 옵션 `CloudWatchLoggingOptionId` 중 하나입니다. DescribeApplication 작업을 `CloudWatchLoggingOptionId` 사용하여 가져올 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

Kinesis Analytics 애플리케이션의 버전 ID.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteApplicationInputProcessingConfiguration

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

입력에서 InputProcessingConfiguration을 삭제합니다.

구문 요청

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string"
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

Kinesis Analytics 애플리케이션 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

Kinesis Analytics 애플리케이션의 버전 ID.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

InputId

입력 처리 구성을 삭제할 입력 구성의 ID입니다. DescribeApplication 작업을 사용하여 애플리케이션의 입력 ID 목록을 가져올 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteApplicationOutput

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

애플리케이션 구성에서 출력 대상 구성을 삭제합니다. Amazon Kinesis Analytics는 더 이상 해당 애플리케이션 내 스트림의 데이터를 외부 출력 대상에 쓰지 않습니다.

이 작업에는 `kinesisanalytics:DeleteApplicationOutput` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "OutputId": "string"
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

Amazon Kinesis Analytics 애플리케이션 명칭

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

Amazon Kinesis Analytics 애플리케이션 버전 DescribeApplication 작업을 사용하여 최신 애플리케이션 버전을 찾을 수 있습니다. 지정된 버전이 최신 버전이 아닌 경우 `ConcurrentModificationException`가 반환됩니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

OutputId

삭제할 구성의 ID. 애플리케이션을 만들 때 또는 나중에 AddApplicationOutput 작업을 사용하여 애플리케이션에 추가되는 각 출력 구성에는 고유한 ID가 있습니다. 애플리케이션 구성에서 삭제하려는 출력 구성을 고유하게 식별하려면 ID를 제공해야 합니다. DescribeApplication 작업을 통하여 특정 `OutputId`를 가져올 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteApplicationReferenceDataSource

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

지정된 애플리케이션 구성에서 참조 데이터 소스 구성을 삭제합니다.

애플리케이션이 실행 중인 경우, Amazon Kinesis Analytics는 AddApplicationReferenceDataSource 작업을 사용하여 생성한 애플리케이션 내 표를 즉시 제거합니다.

이 작업에는 `kinesisanalytics.DeleteApplicationReferenceDataSource` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceId": "string"
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

기존 애플리케이션의 명칭입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

CurrentApplicationVersionId

애플리케이션의 버전입니다. DescribeApplication 작업을 사용하여 최신 애플리케이션 버전을 찾을 수 있습니다. 지정된 버전이 최신 버전이 아닌 경우 `ConcurrentModificationException`가 반환됩니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

ReferenceId

준거 데이터 소스의 ID. 이 ID는 AddApplicationReferenceDataSource를 사용하여 준거 데이터 소스를 애플리케이션에 추가할 때 Amazon Kinesis Analytics에서 할당하는 ID입니다. DescribeApplication 작업을 사용하여 참조 ID를 가져올 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeApplication

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

Amazon Kinesis Analytics 애플리케이션에 대한 정보를 반환합니다.

계정의 모든 애플리케이션 목록을 검색하려면 ListApplications 작업을 사용하십시오.

이 작업에는 `kinesisanalytics:DescribeApplication` 조치를 수행할 권한이 요구됩니다. `Update`와 같은 다른 작업을 호출하는 데 필요한 현재 애플리케이션 버전 ID를 가져오는 데 `DescribeApplication`을 사용할 수 있습니다.

구문 요청

```
{
   "ApplicationName": "string"
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

응답 구문

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

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

ApplicationDetail

애플리케이션 Amazon 리소스 이름(ARN), 상태, 최신 버전, 입력 및 출력 구성 세부 정보와 같은 애플리케이션에 대한 설명을 제공합니다.

타입: ApplicationDetail 객체

오류

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DiscoverInputSchema

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

지정된 스트리밍 소스 (Amazon Kinesis 스트림 또는 Amazon Kinesis Firehose 전송 스트림) 또는 S3 객체의 샘플 레코드를 평가하여 스키마를 유추합니다. 응답에서 작업은 추론된 스키마와 작업에서 스키마를 추론하는 데 사용한 샘플 레코드를 반환합니다.

애플리케이션의 스트리밍 소스를 구성할 때 유추된 스키마를 사용할 수 있습니다. 개념 정보는 애플리케이션 입력 구성을 참조하십시오. Amazon Kinesis Analytics 콘솔을 사용하여 애플리케이션을 생성하면 콘솔은 이 작업을 사용하여 스키마를 유추하고 콘솔 사용자 인터페이스에 표시한다는 점에 유의하십시오.

이 작업에는 `kinesisanalytics:DiscoverInputSchema` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "string"
   },
   "ResourceARN": "string",
   "RoleARN": "string",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string",
      "RoleARN": "string"
   }
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

InputProcessingConfiguration

레코드의 스키마를 발견하기 전에 레코드를 사전 처리하는 데 사용할 InputProcessingConfiguration.

유형: InputProcessingConfiguration객체

필수 여부: 아니요

InputStartingPositionConfiguration

Amazon Kinesis Analytics가 지정된 스트리밍 소스 검색 목적의 레코드 읽기를 시작하도록 하려는 시점입니다.

유형: InputStartingPositionConfiguration객체

필수 여부: 아니요

ResourceARN

스트리밍 소스의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

S3Configuration

Amazon S3 객체의 데이터에서 스키마를 검색하려면 이 파라미터를 지정합니다.

유형: S3Configuration객체

필수 항목 여부: 아니요

응답 구문

```
{
   "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" ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

InputSchema

스트리밍 소스에서 추론된 스키마. 스트리밍 소스의 데이터의 형식을 식별하고 각 데이터 요소가 애플리케이션 내 스트림에서 생성할 수 있는 해당 열에 매핑하는 방법을 설명합니다.

유형: SourceSchema객체

ParsedInputRecords

각 요소가 스트림 레코드의 행에 해당하는 요소 배열 (스트림 레코드는 행을 두 개 이상 포함할 수 있음).

유형: 문자열 배열들의 배열

ProcessedInputRecords

`InputProcessingConfiguration` 파라미터에 지정된 프로세서가 수정한 스트림 데이터.

유형: 문자열 배열

RawInputRecords

스키마를 유추하기 위해 샘플링된 원시 스트림 데이터입니다.

유형: 문자열 배열

오류

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceProvisionedThroughputExceededException

Amazon Kinesis Streams의 프로비저닝된 처리량 ProvisionedExceedExceedeException 때문에 디스커버리가 스트리밍 소스에서 레코드를 가져오지 못했습니다. 자세한 내용을 알아보려면 Amazon Kinesis 스트림 API 참조의 GetRecords를 참조하십시오.

HTTP 상태 코드: 400

ServiceUnavailableException

이 서비스를 사용할 수 없습니다. 작업을 다시 시도하십시오.

HTTP 상태 코드: 500

UnableToDetectSchemaException

유효하지 않는 날짜 형식입니다. Amazon Kinesis Analytics는 지정된 스트리밍 소스에 대한 스키마를 탐지할 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListApplications

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

계정에 있는 Amazon Kinesis Analytics 애플리케이션 목록을 반환합니다. 각 애플리케이션에 대한 응답에는 애플리케이션 명칭, Amazon 리소스 이름 (ARN) 및 상태가 포함됩니다. 응답에서 `HasMoreApplications` 값이 참으로 반환되는 경우 요청 본문에 `ExclusiveStartApplicationName`를 추가하여 다른 요청을 보내고 이 값을 이전 응답의 마지막 애플리케이션 명칭으로 설정할 수 있습니다.

특정 애플리케이션에 대한 자세한 정보가 필요하면 DescribeApplication을 사용하십시오.

이 작업에는 `kinesisanalytics:ListApplications` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "ExclusiveStartApplicationName": "string",
   "Limit": number
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ExclusiveStartApplicationName

목록을 시작할 때 사용할 애플리케이션의 명칭. 페이지 매김을 사용하여 목록을 검색할 경우 첫 번째 요청에서 이 파라미터를 지정하지 않아도 됩니다. 하지만 후속 요청에서는 이전 응답의 마지막 애플리케이션 명칭을 추가하여 애플리케이션의 다음 페이지를 가져옵니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 아니요

Limit

나열할 애플리케이션의 최대수.

타입: 정수

유효 범위: 최소값 1. 최대값 50.

필수 항목 여부: 아니요

응답 구문

```
{
   "ApplicationSummaries": [
      {
         "ApplicationARN": "string",
         "ApplicationName": "string",
         "ApplicationStatus": "string"
      }
   ],
   "HasMoreApplications": boolean
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

ApplicationSummaries

`ApplicationSummary` 객체의 목록.

타입: ApplicationSummary객체 배열

HasMoreApplications

검색할 애플리케이션이 더 있는 경우 참을 반환합니다.

유형: 부울

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListTagsForResource

애플리케이션에 할당된 키값 태그의 목록을 검색합니다. 자세한 설명은 태그 사용을 참조하십시오.

구문 요청

```
{
   "ResourceARN": "string"
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ResourceARN

태그를 검색하기 위한 애플리케이션의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 예

응답 구문

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

Tags

애플리케이션에 할당된 키값 태그.

타입: Tag객체 배열

어레이 멤버: 최소 항목 수 1개. 최대 항목 수 200.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3

- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

StartApplication

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

지정된 Amazon Kinesis Analytics 애플리케이션을 시작합니다. 애플리케이션을 생성한 후에는 이 작업을 독점적으로 호출하여 애플리케이션을 시작해야 합니다.

애플리케이션이 시작되면 입력 데이터를 소비하기 시작하고 처리한 다음 구성된 대상에 출력을 씁니다.

애플리케이션을 시작하려면 애플리케이션 상태가 해당 애플리케이션 상태여야 `READY` 합니다. 콘솔에서 또는 DescribeCustomKeyStores 작업을 사용하여 애플리케이션 상태를 볼 수 있습니다.

애플리케이션을 시작한 후 StopApplication 작업을 호출하여 애플리케이션이 입력을 처리하지 못하도록 할 수 있습니다.

이 작업에는 `kinesisanalytics:StartApplication` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "ApplicationName": "string",
   "InputConfigurations": [
      {
         "Id": "string",
         "InputStartingPositionConfiguration": {
            "InputStartingPosition": "string"
         }
      }
   ]
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

InputConfigurations

애플리케이션이 사용하기 시작하는 특정 입력을 ID로 식별합니다. Amazon Kinesis Analytics는 입력과 관련된 스트리밍 소스를 읽기 시작합니다. 또한 귀하는 스트리밍 소스의 어디에서 Amazon Kinesis Analytics가 읽기를 시작할지를 지정할 수 있습니다.

타입: InputConfiguration객체 배열

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

InvalidApplicationConfigurationException

사용자가 제공한 애플리케이션 구성이 유효하지 않습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

StopApplication

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

애플리케이션이 입력 데이터를 처리하지 못하도록 중지합니다. 실행 상태인 경우에만 애플리케이션을 중지할 수 있습니다. DescribeApplication 작업을 사용하여 애플리케이션 상태를 찾을 수 있습니다. 애플리케이션이 중지되면 Amazon Kinesis Analytics는 입력에서 데이터 읽기를 중지하고, 애플리케이션은 데이터 처리를 중지하며, 대상에 출력은 기록되지 않습니다.

이 작업에는 `kinesisanalytics:StopApplication` 조치를 수행할 권한이 요구됩니다.

구문 요청

```
{
   "ApplicationName": "string"
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

중지하려는 실행 중 애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

TagResource

Kinesis Analytics 애플리케이션에 하나 이상의 키값 태그를 추가합니다. 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다. 자세한 설명은 태그 사용하기를 참조하십시오.

구문 요청

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

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ResourceARN

태그를 지정하기 위한 애플리케이션의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 예

Tags

애플리케이션에 할당된 키값 태그.

타입: Tag객체 배열

어레이 멤버: 최소 항목 수 1개. 최대 항목 수 200.

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

TooManyTagsException

애플리케이션에 너무 많은 태그 또는 너무 많은 태그가 추가된 상태로 애플리케이션이 생성되었습니다. 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

UntagResource

Kinesis Analytics 애플리케이션에서 하나 이상의 태그를 제거합니다. 자세한 설명은 태그 사용하기를 참조하십시오

구문 요청

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ResourceARN

태그를 제거할 Kinesis Analytics 애플리케이션의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 예

TagKeys

지정된 애플리케이션에서 제거할 태그의 키 목록입니다.

타입: 문자열 배열

배열 구성원: 최소수는 1개입니다. 최대 항목 수 200.

길이 제약: 최소 길이 1. 최대 길이는 128.

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

TooManyTagsException

애플리케이션에 너무 많은 태그 또는 너무 많은 태그가 추가된 상태로 애플리케이션이 생성되었습니다. 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK

- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

UpdateApplication

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

기존 Amazon Kinesis Analytics 애플리케이션을 업데이트합니다. 이 API를 사용하여 애플리케이션 코드, 입력 구성 및 출력 구성을 업데이트할 수 있습니다.

Amazon Kinesis Analytics는 애플리케이션을 업데이트할 때마다 `CurrentApplicationVersionId`을 업데이트 한다는 것을 유의하십시오.

이 작업에는 `kinesisanalytics:UpdateApplication` 작업 권한이 필요합니다.

구문 요청

```
{
   "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
}
```

요청 파라미터

요청은 JSON 형식으로 다음 데이터를 받습니다.

ApplicationName

업데이트할 Amazon Kinesis Analytics 애플리케이션의 명칭입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 예

ApplicationUpdate

애플리케이션 업데이트에 대해 설명합니다.

타입: ApplicationUpdate객체

필수 여부: 예

CurrentApplicationVersionId

최신 애플리케이션 버전 ID. DescribeApplication 작업을 사용하여 이 값을 얻을 수 있습니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

응답 요소

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

CodeValidationException

사용자가 제공한 애플리케이션 코드 (쿼리) 가 유효하지 않습니다. 이는 단순한 구문 오류일 수 있습니다.

HTTP 상태 코드: 400

ConcurrentModificationException

애플리케이션을 동시에 수정한 결과 예외가 발생했습니다. 예컨대, 두 사람이 동시에 같은 애플리케이션을 편집하려고 하는 경우를 예로 들 수 있습니다.

HTTP 상태 코드: 400

InvalidArgumentException

지정한 입력 파라미터 값이 유효하지 않습니다.

HTTP 상태 코드: 400

ResourceInUseException

이 작업을 위한 애플리케이션을 얻을 수 없습니다.

HTTP 상태 코드: 400

ResourceNotFoundException

지정된 애플리케이션을 찾을 수 없습니다.

HTTP 상태 코드: 400

UnsupportedOperationException

지정된 파라미터가 지원되지 않거나 지정된 리소스가 이 작업에 유효하지 않아 요청이 거부되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

데이터 타입

다음 데이터 타입이 지원됩니다.

- 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**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

애플리케이션 Amazon 리소스 이름(ARN), 상태, 최신 버전, 입력 및 출력 구성을 포함하여 애플리케이션에 대한 설명을 제공합니다.

내용

ApplicationARN

애플리케이션의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

ApplicationName

애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

ApplicationStatus

애플리케이션 상태.

타입: 문자열

유효 값: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

필수 여부: 예

ApplicationVersionId

최신 애플리케이션 버전을 제공합니다.

타입: Long

유효 범위: 최소값 1. 최대값 999999999.

필수 여부: 예

ApplicationCode

애플리케이션의 모든 애플리케이션 내 스트림에 대한 데이터 분석을 수행하기 위해 제공한 애플리케이션 코드를 반환합니다.

유형: 문자열

길이 제한: 최소 길이는 0. 최대 길이 102400.

필수 여부: 아니요

ApplicationDescription

애플리케이션의 설명.

유형: 문자열

길이 제약 조건: 최소 길이는 0입니다. 최대 길이 1,024.

필수 여부: 아니요

CloudWatchLoggingOptionDescriptions

애플리케이션 메시지를 수신하도록 구성된 CloudWatch 로그 스트림을 설명합니다. Amazon Kinesis Analytics 애플리케이션에서 CloudWatch 로그 스트림을 사용하는 방법에 대한 자세한 설명은 Amazon CloudWatch Logs 사용을 참조하십시오.

유형: CloudWatchLoggingOptionDescription 객체 배열

필수 여부: 아니요

CreateTimestamp

애플리케이션 버전이 생성된 타임스탬프.

유형: 타임스탬프

필수 여부: 아니요

InputDescriptions

애플리케이션 입력 구성을 설명합니다. 자세한 설명은 애플리케이션 입력 구성을 참조하십시오.

유형: InputDescription 객체 배열

필수 여부: 아니요

LastUpdateTimestamp

애플리케이션이 마지막으로 업데이트된 시간의 타임스탬프입니다.

유형: 타임스탬프

필수 여부: 아니요

OutputDescriptions

애플리케이션 출력 구성을 설명합니다. 자세한 설명은 애플리케이션 출력 구성을 참조하십시오.

유형: OutputDescription 객체 배열

필수 여부: 아니요

ReferenceDataSourceDescriptions

애플리케이션을 위해 구성된 참조 데이터 소스를 설명합니다. 자세한 설명은 애플리케이션 입력 구성을 참조하십시오.

유형: ReferenceDataSourceDescription 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

ApplicationSummary

> **Note**
>
> 이 설명서는 Amazon Kinesis Data Analytics API 버전 1용이며, SQL 애플리케이션만 지원합니다. API 버전 2에서 SQL 및 Java 애플리케이션을 지원합니다. 버전 2에 대한 자세한 설명은 Amazon Kinesis Data Analytics API V2 설명서를 참조하십시오.

애플리케이션의 Amazon 리소스 이름(ARN), 명칭 및 상태를 비롯한 애플리케이션 요약 정보를 제공합니다.

내용

ApplicationARN

애플리케이션의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

ApplicationName

애플리케이션의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이 128.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

ApplicationStatus

애플리케이션 상태.

타입: 문자열

유효 값: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ApplicationUpdate

기존 Amazon Kinesis Analytics 애플리케이션에 적용할 업데이트 설명.

내용

ApplicationCodeUpdate

애플리케이션 코드 업데이트 설명.

유형: 문자열

길이 제한: 최소 길이는 0. 최대 길이 102400.

필수 여부: 아니요

CloudWatchLoggingOptionUpdates

애플리케이션 CloudWatch 로깅 옵션 업데이트 설명.

유형: CloudWatchLoggingOptionUpdate 객체 배열

필수 여부: 아니요

InputUpdates

애플리케이션 입력 구성 업데이트 설명.

유형: InputUpdate 객체 배열

필수 여부: 아니요

OutputUpdates

애플리케이션 출력 구성 업데이트 설명.

유형: OutputUpdate 객체 배열

필수 여부: 아니요

ReferenceDataSourceUpdates

애플리케이션 참조 데이터 소스 업데이트 설명.

유형: ReferenceDataSourceUpdate 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOption

Amazon 리소스 이름(ARN) 로그 스트림, ARN 역할을 포함하여 Amazon CloudWatch 로깅 옵션에 대한 설명을 제공합니다.

내용

LogStreamARN

애플리케이션 메시지를 수신하기 위한 CloudWatch 로그의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

애플리케이션 메시지를 보내는 데 사용할 역할의 IAM ARN. 참고: 애플리케이션 메시지를 CloudWatch에 쓰려면 사용하는 IAM 역할에는 `PutLogEvents` 정책 작업이 활성화되어 있어야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOptionDescription

CloudWatch 로깅 옵션에 대한 설명.

내용

LogStreamARN

애플리케이션 메시지를 수신하기 위한 CloudWatch 로그의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

애플리케이션 메시지를 보내는 데 사용할 역할의 IAM ARN. 참고: 애플리케이션 메시지를 CloudWatch에 쓰려면 사용된 IAM 역할에 `PutLogEvents` 정책 작업이 활성화되어 있어야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

CloudWatchLoggingOptionId

CloudWatch 로깅 옵션 설명의 ID입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudWatchLoggingOptionUpdate

CloudWatch 로깅 옵션 업데이트 설명.

내용

CloudWatchLoggingOptionId

업데이트할 CloudWatch 로깅 옵션의 ID

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

LogStreamARNUpdate

애플리케이션 메시지를 수신하기 위한 CloudWatch 로그의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARNUpdate

애플리케이션 메시지를 보내는 데 사용할 역할의 IAM ARN. 참고: 애플리케이션 메시지를 CloudWatch에 쓰려면 사용된 IAM 역할에 `PutLogEvents` 정책 작업이 활성화되어 있어야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CSVMappingParameters

레코드 형식이 구분 기호를 사용(예: CSV)할 때 추가 매핑 정보를 제공합니다. 예를 들어 다음 샘플 레코드는 CSV 형식을 사용하며, 레코드에는 행 구분 기호로 '\n'가 사용되고, 열 구분 기호로 쉼표(",")가 사용됩니다.

```
"name1", "address1"

"name2", "address2"
```

내용

RecordColumnDelimiter

열 구분 기호입니다. 예를 들어, CSV 형식에서 쉼표(",")는 일반적인 열 구분 기호입니다.

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 예

RecordRowDelimiter

행 구분 기호입니다. 예를 들어, CSV 형식에서 '\n'은 일반적인 행 구분 기호입니다.

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DestinationSchema

대상에 레코드를 쓸 때 데이터 형식을 기술합니다. 자세한 설명은 애플리케이션 출력 구성을 참조하십시오.

내용

RecordFormatType

출력 스트림의 레코드 형식을 지정합니다.

타입: 문자열

유효 값: `JSON | CSV`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Input

애플리케이션 입력을 구성할 때 스트리밍 소스, 생성된 애플리케이션 내 스트림 명칭 및 그 둘 사이의 매핑을 지정합니다. 자세한 설명은 애플리케이션 입력 구성을 참조하십시오.

내용

InputSchema

스트리밍 소스에서 데이터의 형식 및 각 데이터 요소가 애플리케이션 내 스트림에서 생성되는 해당 열에 매핑되는 방법을 설명합니다.

준거 데이터 소스의 형식을 기술하는 데에도 사용됩니다.

유형: SourceSchema 객체

필수 여부: 예

NamePrefix

애플리케이션 내 스트림을 생성할 때 사용할 명칭 접두사입니다. "MyInApplicationStream" 접두사를 지정하는 경우를 예로 들어 보겠습니다. Amazon Kinesis Analytics는 한 개 이상(지정한 `InputParallelism` 수 기준) 애플리케이션 내 스트림을 "MyInApplicationStream_001," "MyInApplicationStream_002" 등의 명칭으로 생성합니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 예

InputParallelism

생성할 애플리케이션 내 스트림 수를 설명합니다.

소스의 데이터는 이러한 애플리케이션 내 입력 스트림을 통해 라우팅됩니다.

(애플리케이션 입력 구성을 참조하십시오.)

유형: InputParallelism 객체

필수 여부: 아니요

InputProcessingConfiguration

입력을 위한 InputProcessingConfiguration입니다. 입력 프로세서는 애플리케이션의 SQL 코드가 실행되기 전에 스트림에서 받은 레코드를 변환합니다. 현재 사용 가능한 입력 처리 구성은 InputLambdaProcessor뿐입니다.

유형: InputProcessingConfiguration 객체

필수 여부: 아니요

KinesisFirehoseInput

스트리밍 소스가 Amazon Kinesis Firehose 전송 스트림인 경우, Amazon Kinesis Analytics가 스트림에 액세스할 수 있도록 허용하는 IAM 역할과 전송 스트림의 ARN을 식별합니다.

참고: `KinesisStreamsInput` 또는 `KinesisFirehoseInput`은 필수입니다.

유형: KinesisFirehoseInput 객체

필수 여부: 아니요

KinesisStreamsInput

스트리밍 소스가 Amazon Kinesis 스트림인 경우, Amazon Kinesis Analytics가 스트림에 액세스할 수 있도록 허용하는 IAM 역할과 스트림의 Amazon 리소스 이름(ARN)을 식별합니다.

참고: `KinesisStreamsInput` 또는 `KinesisFirehoseInput`은 필수입니다.

유형: KinesisStreamsInput 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputConfiguration

애플리케이션을 시작할 때 이 구성을 제공합니다. 이 구성은 입력 소스와 애플리케이션이 레코드 처리를 시작할 입력 소스 내 지점을 식별합니다.

내용

Id

입력 소스 ID. 이 ID는 DescribeApplication 작업을 호출함으로써 가져올 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

InputStartingPositionConfiguration

애플리케이션이 스트리밍 소스의 레코드 처리를 시작하기를 원하는 시점입니다.

유형: InputStartingPositionConfiguration 객체

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputDescription

애플리케이션 입력 구성을 설명합니다. 자세한 설명은 애플리케이션 입력 구성을 참조하십시오.

내용

InAppStreamNames

스트림 소스에 매핑된 애플리케이션 내 스트림 명칭을 반환합니다.

유형: 문자열 배열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 아니요

InputId

애플리케이션 입력과 연계된 입력 ID. 이는 Amazon Kinesis Analytics가 애플리케이션에 추가하는 각 입력 구성에 할당하는 ID입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

Required: No

InputParallelism

구성된 병렬 처리 수 (스트리밍 소스에 매핑된 애플리케이션 내 스트림 수) 를 설명합니다.

유형: InputParallelism 객체

필수 여부: 아니요

InputProcessingConfigurationDescription

애플리케이션 코드가 실행되기 전에 이 입력의 레코드에서 실행되는 전처리기에 대한 설명입니다.

유형: InputProcessingConfigurationDescription 객체

필수 여부: 아니요

InputSchema

스트리밍 소스에서 데이터의 형식 및 각 데이터 요소가 애플리케이션 내 스트림에서 생성되는 해당 열에 매핑되는 방법을 설명합니다.

유형: SourceSchema 객체

필수 여부: 아니요

InputStartingPositionConfiguration

애플리케이션이 입력 스트림에서 읽도록 구성된 시점입니다.

유형: InputStartingPositionConfiguration 객체

필수 여부: 아니요

KinesisFirehoseInputDescription

Amazon Kinesis Firehose 전송 스트림이 스트리밍 소스로 구성된 경우, Amazon Kinesis Analytics가 스트림에 액세스할 수 있도록 허용하는 IAM 역할과 전송 스트림의 ARN을 제공합니다.

유형: KinesisFirehoseInputDescription 객체

필수 여부: 아니요

KinesisStreamsInputDescription

Amazon Kinesis 스트림이 스트리밍 소스로 구성된 경우, Amazon Kinesis Analytics가 스트림에 액세스할 수 있도록 허용하는 IAM 역할과 Amazon Kinesis 스트림의 Amazon 리소스 이름(ARN)을 제공합니다.

유형: KinesisStreamsInputDescription 객체

필수 여부: 아니요

NamePrefix

애플리케이션 내 명칭 접두사.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessor

스트림의 레코드를 사전 처리하는 데 사용되는 AWS Lambda 함수의 Amazon 리소스 이름(ARN)과 AWS Lambda 함수에 액세스하는 데 사용되는 IAM 역할의 ARN이 포함된 객체입니다.

내용

ResourceARN

스트림의 레코드에서 작동하는 AWS Lambda 함수의 ARN입니다.

> **Note**
>
> Lambda 함수의 최신 버전보다 이전 버전을 지정하려면 Lambda 함수 ARN에 Lambda 함수 버전을 포함시키십시오. Lambda ARNs 대한 자세한 내용은 예제 ARNs AWS.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

AWS Lambda 함수에 액세스하는 데 사용되는 IAM 역할의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessorDescription

스트림의 레코드를 사전 처리하는 데 사용되는 AWS Lambda 함수의 Amazon 리소스 이름(ARN)과 AWS Lambda 표현식에 액세스하는 데 사용되는 IAM 역할의 ARN이 포함된 객체입니다.

내용

ResourceARN

스트림의 레코드에서 전처리하는 데 사용되는 AWS Lambda 함수의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

Required: No

RoleARN

AWS Lambda 함수에 액세스하는 데 사용되는 IAM 역할의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputLambdaProcessorUpdate

스트림의 레코드를 전처리하는 데 사용되는 InputLambdaProcessor에 대한 업데이트를 나타냅니다.

내용

ResourceARNUpdate

스트림의 레코드를 사전 처리하는 데 사용되는 새 AWS Lambda 함수의 Amazon 리소스 이름 (ARN) 입니다.

> **Note**
>
> Lambda 함수의 최신 버전보다 이전 버전을 지정하려면 Lambda 함수 ARN에 Lambda 함수 버전을 포함시키십시오. Lambda ARNs 대한 자세한 내용은 예제 ARNs AWS.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

Required: No

RoleARNUpdate

AWS Lambda 함수에 액세스하는 데 사용되는 새 IAM 역할의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputParallelism

특정 스트리밍 소스에 대해 생성할 애플리케이션 내 스트림의 수를 기술합니다. 병렬 처리에 대한 자세한 설명은 애플리케이션 입력 구성을 참조하십시오.

내용

Count

생성할 애플리케이션 내 스트림의 수. 자세한 설명은 제한을 참조하십시오.

타입: 정수

유효 범위: 최소값 1. 최대값은 64입니다.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputParallelismUpdate

병렬 처리 수를 업데이트합니다.

내용

CountUpdate

특정 스트리밍 소스를 위해 생성할 애플리케이션 내 스트림 수.

타입: 정수

유효 범위: 최소값 1. 최대값은 64입니다.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfiguration

애플리케이션 코드에서 처리하기 전에 스트림에서 레코드를 사전 처리하는 데 사용되는 프로세서에 대한 설명을 제공합니다. 현재 사용할 수 있는 유일한 입력 프로세서는 AWS Lambda입니다.

내용

InputLambdaProcessor

애플리케이션 코드가 처리를 하기 전에 스트림의 레코드를 사전 처리할 때 사용하는 InputLambdaProcessor.

유형: InputLambdaProcessor 객체

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfigurationDescription

입력 프로세서에 대한 구성 정보를 제공합니다. 현재 사용할 수 있는 유일한 입력 프로세서는 AWS Lambda입니다.

내용

InputLambdaProcessorDescription

관련 InputLambdaProcessorDescription에 대한 구성 정보를 제공합니다.

유형: InputLambdaProcessorDescription 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputProcessingConfigurationUpdate

InputProcessingConfiguration 업데이트에 대해 설명합니다.

내용

InputLambdaProcessorUpdate

InputLambdaProcessor에 대한 업데이트 정보를 제공합니다.

유형: InputLambdaProcessorUpdate 객체

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputSchemaUpdate

애플리케이션의 입력 스키마 업데이트에 대해 설명합니다.

내용

RecordColumnUpdates

`RecordColumn` 객체의 목록. 각 객체는 스트리밍 소스 요소가 애플리케이션 내 스트림의 해당 열에 매핑되는 방법을 기술합니다.

유형: RecordColumn객체 어레이

배열 멤버: 최소 항목 수는 1입니다. 최대 항목 수 1,000.

필수 여부: 아니요

RecordEncodingUpdate

스트리밍 소스에서 레코드의 인코딩을 지정합니다. 예: UTF-8.

유형: String

패턴: `UTF-8`

Required: No

RecordFormatUpdate

스트리밍 소스에서 레코드의 형식을 지정합니다.

유형: RecordFormat 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputStartingPositionConfiguration

애플리케이션이 스트리밍 소스에서 읽는 지점을 설명합니다.

내용

InputStartingPosition

스트림 상의 시작 위치.

- `NOW` - 스트림의 가장 최근 레코드가 나온 직후에 읽기를 시작하고, 고객이 발행한 요청 타임스탬프에서 시작합니다.
- `TRIM_HORIZON` - 스트림에서 트리밍되지 않은 마지막 레코드, 즉 스트림에서 가장 오래된 레코드에서 읽기를 시작합니다. Amazon Kinesis Firehose 전송 스트림에는 이 옵션을 사용할 수 없습니다.
- `LAST_STOPPED_POINT` - 애플리케이션이 마지막으로 읽기를 중단한 시점부터 읽기를 재개합니다.

타입: 문자열

유효 값: `NOW | TRIM_HORIZON | LAST_STOPPED_POINT`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputUpdate

특정 입력 구성(애플리케이션의 `InputId`으로 식별)에 대한 업데이트를 설명합니다.

내용

InputId

업데이트할 애플리케이션 입력의 입력 ID.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

InputParallelismUpdate

병렬 처리 업데이트에 대해 설명합니다(Amazon Kinesis Analytics가 특정 스트리밍 소스에 대해 생성하는 애플리케이션 내 스트림 수).

유형: InputParallelismUpdate 객체

필수 여부: 아니요

InputProcessingConfigurationUpdate

입력 처리 구성의 업데이트를 설명합니다.

유형: InputProcessingConfigurationUpdate 객체

필수 여부: 아니요

InputSchemaUpdate

스트리밍 소스에서의 데이터 형식 및 스트리밍 소스의 레코드 요소가 생성된 애플리케이션 내 스트림의 열에 어떻게 매핑되는지를 설명합니다.

유형: InputSchemaUpdate 객체

필수 여부: 아니요

KinesisFirehoseInputUpdate

Amazon Kinesis Firehose 전송 스트림이 업데이트 대상 스트리밍 소스인 경우, 업데이트된 스트림 ARN 및 IAM 역할 ARN을 제공합니다.

유형: KinesisFirehoseInputUpdate 객체

필수 여부: 아니요

KinesisStreamsInputUpdate

Amazon Kinesis 스트림이 업데이트할 스트리밍 소스인 경우, 업데이트된 스트림의 Amazon 리소스 이름 (ARN) 과 IAM 역할 ARN을 제공합니다.

유형: KinesisStreamsInputUpdate 객체

필수 여부: 아니요

NamePrefixUpdate

Amazon Kinesis Analytics가 특정 스트리밍 소스에 대해 생성하는 애플리케이션 내 스트림의 명칭 접두사입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

JSONMappingParameters

스트리밍 소스의 레코드 형식이 JSON일 경우 추가 매핑 정보를 제공합니다.

내용

RecordRowPath

레코드를 포함하는 상위 수준에 대한 경로입니다.

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInput

스트리밍 소스로 Amazon Kinesis Firehose 전송 스트림을 지정합니다. 전송 스트림의 Amazon 리소스 이름(ARN)과, Amazon Kinesis Analytics가 사용자를 대신하여 스트림에 액세스할 수 있도록 하는 IAM 역할 ARN을 제공하십시오.

내용

ResourceARN

입력 전송 스트림의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다. 역할이 스트림에 액세스하는 데 필요한 권한을 갖고 있는지 확인해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInputDescription

애플리케이션 입력 구성에서 스트리밍 소스로 구성된 Amazon Kinesis Firehose 전송 스트림을 설명합니다.

내용

ResourceARN

Amazon Kinesis Firehose 전송 시스템의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스하기 위해 취하는 IAM 역할의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseInputUpdate

애플리케이션 입력 구성을 업데이트할 때, 스트리밍 소스로 Amazon Kinesis Firehose 전송 스트림을 지정합니다.

내용

ResourceARNUpdate

읽기 위한 입력 Amazon Kinesis Firehose 전송 스트림의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARNUpdate

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseOutput

애플리케이션 출력을 구성할 때 대상으로 Amazon Kinesis Firehose 전송 스트림을 지정합니다. 스트림의 Amazon 리소스 이름(ARN)과, Amazon Kinesis Analytics가 사용자를 대신하여 스트림에 쓸 수 있도록 허용하는 IAM 역할을 제공합니다.

내용

ResourceARN

데이터를 쓸 대상 Amazon Kinesis Firehose 전송 스트림의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

Amazon Kinesis Analytics가 사용자를 대신하여 대상 스트림에 쓸 수 있는 권한을 가진 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseOutputDescription

애플리케이션 출력을 위해, 그 목적지로 구성된 Amazon Kinesis Firehose 전송 스트림을 설명합니다.

내용

ResourceARN

Amazon Kinesis Firehose 전송 시스템의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스하기 위해 취할 수 있는 IAM 역할의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisFirehoseOutputUpdate

UpdateApplication 작업을 사용하여 출력 구성을 업데이트할 때 목적지로 구성된 Amazon Kinesis Firehose 전송 스트림에 대한 정보를 제공합니다.

내용

ResourceARNUpdate

데이터를 쓸 대상 Amazon Kinesis Firehose 전송 스트림의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARNUpdate

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsInput

스트리밍 소스로 Amazon Kinesis 스트림을 지정합니다. 스트림의 Amazon 리소스 이름(ARN)과, Amazon Kinesis Analytics가 사용자를 대신하여 스트림에 액세스할 수 있도록 허용하는 IAM 역할 ARN을 제공합니다.

내용

ResourceARN

읽을 입력 Amazon Kinesis 스트림의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsInputDescription

애플리케이션 입력 구성에서 스트리밍 소스로 구성된 Amazon Kinesis 스트림을 설명합니다.

내용

ResourceARN

Amazon Kinesis 스트림의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스하기 위해 취할 수 있는 IAM 역할의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsInputUpdate

애플리케이션 입력 구성을 업데이트할 때 스트리밍 소스로 Amazon Kinesis 스트림 정보를 입력합니다.

내용

ResourceARNUpdate

읽기 위한 입력 Amazon Kinesis 스트림의 Amazon 리소스 이름(ARN)

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARNUpdate

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutput

애플리케이션 출력을 구성할 때 대상으로 Amazon Kinesis 스트림을 지정합니다. 스트림의 Amazon 리소스 이름(ARN)과, Amazon Kinesis Analytics가 사용자를 대신하여 스트림에 쓸 수 있는 권한을 가진 IAM 역할 ARN을 제공합니다.

내용

ResourceARN

데이터를 쓸 대상 Amazon Kinesis 스트림의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

Amazon Kinesis Analytics가 사용자를 대신하여 대상 스트림에 쓸 수 있는 권한을 가진 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutputDescription

애플리케이션 출력의 경우 Amazon Kinesis를 목적지로 구성한 것으로 설명합니다.

내용

ResourceARN

Amazon Kinesis 스트림의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARN

Amazon Kinesis Analytics가 스트림에 액세스하기 위해 취할 수 있는 IAM 역할의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KinesisStreamsOutputUpdate

UpdateApplication 작업을 사용하여 출력 구성을 업데이트할 때 목적지로 구성된 Amazon Kinesis 스트림에 대한 정보를 제공합니다.

내용

ResourceARNUpdate

출력을 기록할 Amazon Kinesis 스트림의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARNUpdate

Amazon Kinesis Analytics가 스트림에 액세스할 수 있는 권한을 주는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutput

애플리케이션 출력을 구성할 때는 AWS Lambda 함수를 대상으로 식별합니다. 함수의 Amazon 리소스 이름(ARN)과, Amazon Kinesis Analytics가 사용자를 대신하여 함수에 쓸 수 있는 권한을 가진 IAM 역할 ARN을 제공합니다.

내용

ResourceARN

데이터를 쓸 Lambda 함수의 Amazon 리소스 이름(ARN)입니다.

> **Note**
>
> Lambda 함수의 최신 버전보다 이전 버전을 지정하려면 Lambda 함수 ARN에 Lambda 함수 버전을 포함시키십시오. Lambda ARNs 대한 자세한 내용은 예제 ARNs AWS.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

RoleARN

Amazon Kinesis Analytics가 사용자를 대신하여 대상 함수에 대한 쓰기 권한을 가질 수 있는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutputDescription

애플리케이션 출력의 경우는 대상으로 구성된 AWS Lambda 함수를 설명합니다.

내용

ResourceARN

목적지 Lambda 함수의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARN

Amazon Kinesis Analytics가 목적지 함수에 쓰기 위해 취할 수 있는 IAM 역할의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LambdaOutputUpdate

UpdateApplication 작업을 사용하여 출력 구성을 업데이트할 때는 대상으로 구성된 AWS Lambda 함수에 대한 정보를 제공합니다.

내용

ResourceARNUpdate

목적지 Lambda 함수의 Amazon 리소스 이름(ARN).

> **Note**
> Lambda 함수의 최신 버전보다 이전 버전을 지정하려면 Lambda 함수 ARN에 Lambda 함수 버전을 포함시키십시오. Lambda ARNs 대한 자세한 내용은 예제 ARNs AWS.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

RoleARNUpdate

Amazon Kinesis Analytics가 사용자를 대신하여 대상 함수에 대한 쓰기 권한을 가질 수 있는 IAM 역할의 ARN입니다. 이 역할에 다음과 같은 권한을 부여해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MappingParameters

애플리케이션을 생성하거나 업데이트할 때 애플리케이션 입력을 구성할 경우 스트리밍 소스의 레코드 형식(예: JSON, CSV 또는 몇 개의 구분 기호로 구분된 레코드 필드)과 관련된 추가 매핑 정보를 제공합니다.

내용

CSVMappingParameters

레코드 형식이 구분 기호를 사용(예: CSV)할 때 추가 매핑 정보를 제공합니다.

유형: CSVMappingParameters 객체

필수 여부: 아니요

JSONMappingParameters

스트리밍 소스의 레코드 형식이 JSON일 경우 추가 매핑 정보를 제공합니다.

유형: JSONMappingParameters 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Output

인애플리케이션 스트림 데이터를 쓸 대상과, 인애플리케이션 스트림을 식별하는 애플리케이션 출력 구성을 기술합니다. 대상은 Amazon Kinesis 스트림 도는 Amazon Kinesis Firehose 전송 시스템이 될 수 있습니다.

애플리케이션이 쓸 수 있는 대상 수의 제한과 기타 제한을 보려면 제한을 참조하십시오.

내용

DestinationSchema

대상에 레코드를 쓸 때 데이터 형식을 기술합니다. 자세한 설명은 애플리케이션 출력 구성을 참조하십시오.

유형: DestinationSchema 객체

필수 여부: 예

Name

인애플리케이션 스트림의 이름입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 예

KinesisFirehoseOutput

대상으로 Amazon Kinesis Firehose 전송 스트림을 지정합니다.

유형: KinesisFirehoseOutput 객체

필수 여부: 아니요

KinesisStreamsOutput

대상으로 Amazon Kinesis 스트림을 지정합니다.

유형: KinesisStreamsOutput 객체

필수 여부: 아니요

LambdaOutput

AWS Lambda 함수를 대상으로 식별합니다.

유형: LambdaOutput 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputDescription

애플리케이션 출력 구성을 설명합니다. 여기에는 애플리케이션 내 스트림 명칭과 스트림 데이터가 기록되는 대상이 포함됩니다. 대상은 Amazon Kinesis 스트림 도는 Amazon Kinesis Firehose 전송 시스템이 될 수 있습니다.

내용

DestinationSchema

대상에 데이터를 쓰는 데 사용되는 데이터 형식입니다.

유형: DestinationSchema 객체

필수 여부: 아니요

KinesisFirehoseOutputDescription

출력이 기록되는 대상으로 구성된 Amazon Kinesis Firehose 전송 스트림을 지정합니다.

유형: KinesisFirehoseOutputDescription 객체

필수 여부: 아니요

KinesisStreamsOutputDescription

출력이 기록되는 대상으로 구성된 Amazon Kinesis 스트림을 설명합니다.

유형: KinesisStreamsOutputDescription 객체

필수 여부: 아니요

LambdaOutputDescription

출력이 기록되는 대상으로 구성된 AWS Lambda 함수를 설명합니다.

유형: LambdaOutputDescription 객체

필수 여부: 아니요

Name

출력으로 구성된 애플리케이션 내 스트림의 명칭.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 아니요

OutputId

출력 구성을 위한 고유 식별자.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

패턴: `[a-zA-Z0-9_.-]+`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputUpdate

로 식별되는 출력 구성의 업데이트에 대해 설명합니다. `OutputId`

내용

OutputId

업데이트하려는 특정 출력 구성을 지정합니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

DestinationSchemaUpdate

대상에 레코드를 쓸 때 데이터 형식을 기술합니다. 자세한 설명은 애플리케이션 출력 구성을 참조하십시오.

유형: DestinationSchema 객체

필수 여부: 아니요

KinesisFirehoseOutputUpdate

출력을 위한 대상으로 Amazon Kinesis Firehose 전송 스트림을 지정합니다.

유형: KinesisFirehoseOutputUpdate 객체

필수 여부: 아니요

KinesisStreamsOutputUpdate

출력을 위한 대상으로 Amazon Kinesis 스트림을 지정합니다.

유형: KinesisStreamsOutputUpdate 객체

필수 여부: 아니요

LambdaOutputUpdate

AWS Lambda 함수를 출력의 대상으로 설명합니다.

유형: LambdaOutputUpdate 객체

필수 여부: 아니요

NameUpdate

이 출력 구성에 대해 다른 애플리케이션 내 스트림을 지정하려면 이 필드를 사용하여 새 애플리케이션 내 스트림 명칭을 지정하십시오.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecordColumn

스트리밍 소스의 각 데이터 요소가 인애플리케이션 스트림에 있는 해당 열에 매핑되는 방법을 기술합니다.

준거 데이터 소스의 형식을 기술하는 데에도 사용됩니다.

내용

Name

인애플리케이션 입력 스트림 또는 참조 테이블에서 생성된 열의 이름입니다.

유형: 문자열

필수 항목 여부: 예

SqlType

인애플리케이션 입력 스트림 또는 참조 테이블에서 생성된 열의 형식입니다.

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 예

Mapping

스트리밍 입력 또는 참조 데이터 소스에 있는 데이터 요소에 대한 참조입니다. RecordFormatType이 `JSON`인 경우 이 요소를 반드시 지정해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecordFormat

스트림의 레코드를 스키마로 변환하기 위해 적용해야 하는 레코드 형식 및 관련 매핑 정보를 기술합니다.

내용

RecordFormatType

레코드 형식의 유형입니다.

타입: 문자열

유효 값: `JSON | CSV`

필수 사항 여부: 예

MappingParameters

애플리케이션을 생성하거나 업데이트할 때 애플리케이션 입력을 구성할 경우 스트리밍 소스의 레코드 형식(예: JSON, CSV 또는 몇 개의 구분 기호로 구분된 레코드 필드)과 관련된 추가 매핑 정보를 제공합니다.

유형: MappingParameters 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSource

소스 정보(S3 버킷 이름 및 객체 키 이름), 생성된 인애플리케이션 테이블 이름 및 Amazon S3 객체의 데이터 요소를 인애플리케이션 테이블에 매핑하는 데 필요한 스키마를 제공하여 참조 데이터 소스를 지정합니다.

내용

ReferenceSchema

스트리밍 소스에서 데이터의 형식 및 각 데이터 요소가 애플리케이션 내 스트림에서 생성된 해당 열에 매핑되는 방법을 설명합니다.

유형: SourceSchema 객체

필수 여부: 예

TableName

생성할 인애플리케이션 테이블 이름입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 예

S3ReferenceDataSource

참조 데이터를 포함하는 S3 버킷과 객체를 식별합니다. 또한 Amazon Kinesis Analytics가 이 객체를 읽기 위해 사용할 수 있는 IAM 역할을 식별합니다. Amazon Kinesis Analytics 애플리케이션 로드는 데이터를 한 번만 참조합니다. 데이터가 변경될 경우 데이터를 애플리케이션으로 다시 로드하도록 트리거하는 `UpdateApplication` 작업을 호출합니다.

유형: S3ReferenceDataSource 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSourceDescription

애플리케이션에 구성된 준거 데이터 소스를 설명합니다.

내용

ReferenceId

준거 데이터 소스의 ID. 이 ID는 AddApplicationReferenceDataSource 작업을 사용하여 준거 데이터 소스를 애플리케이션에 추가할 때 Amazon Kinesis Analytics에서 할당하는 ID입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

S3ReferenceDataSourceDescription

S3 버킷 명칭과 준거 데이터를 포함하는 객체 키 명칭을 제공합니다. 또한 Amazon Kinesis Analytics가 Amazon S3 객체를 읽고 애플리케이션 내 참조 표를 채우는 데 맡길 수 있는 IAM 역할의 Amazon 리소스 이름(ARN) 도 제공합니다.

유형: S3ReferenceDataSourceDescription 객체

필수 여부: 예

TableName

특정 준거 데이터 소스 구성으로 생성된 애플리케이션 내 표 명칭.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 예

ReferenceSchema

스트리밍 소스에서 데이터의 형식 및 각 데이터 요소가 애플리케이션 내 스트림에서 생성된 해당 열에 매핑되는 방법을 설명합니다.

유형: SourceSchema 객체

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReferenceDataSourceUpdate

애플리케이션의 준거 데이터 소스 구성을 업데이트하는 경우 이 객체는 업데이트된 모든 값 (예: 원본 버킷 명칭 및 객체 키 명칭), 생성된 애플리케이션 내 표 명칭 및 Amazon S3 객체의 데이터를 생성된 애플리케이션 내 참조 표에 매핑하는 업데이트된 매핑 정보를 제공합니다.

내용

ReferenceId

업데이트 중인 준거 데이터 소스의 ID. DescribeApplication 작업을 사용하여 이 값을 얻을 수 있습니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 50.

Pattern: `[a-zA-Z0-9_.-]+`

필수 여부: 예

ReferenceSchemaUpdate

스트리밍 소스에서 데이터의 형식 및 각 데이터 요소가 애플리케이션 내 스트림에서 생성된 해당 열에 매핑되는 방법을 설명합니다.

유형: SourceSchema 객체

필수 여부: 아니요

S3ReferenceDataSourceUpdate

Amazon Kinesis Analytics가 사용자를 대신하여 Amazon S3 객체를 읽고 애플리케이션 내 참조 표를 채우는 데 맡길 수 있는 S3 버킷 명칭, 객체 키 명칭 및 IAM 역할을 설명합니다.

유형: S3ReferenceDataSourceUpdate 객체

필수 여부: 아니요

TableNameUpdate

이 업데이트로 생성되는 애플리케이션 내 표 명칭.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 32.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3Configuration

S3 버킷의 Amazon 리소스 이름(ARN), 버킷에 액세스하는 데 사용되는 IAM 역할의 ARN, 데이터가 포함된 Amazon S3 객체의 명칭 등 Amazon S3 데이터 소스에 대한 설명을 제공합니다.

내용

BucketARN

데이터가 들어있는 S3 버킷의 명칭.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

FileKey

데이터가 들어있는 객체의 명칭.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1,024.

필수 여부: 예

RoleARN

데이터 액세스에 사용되는 역할의 IAM ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSource

참조 데이터를 포함하는 S3 버킷과 객체를 식별합니다. 또한 Amazon Kinesis Analytics가 이 객체를 읽기 위해 사용할 수 있는 IAM 역할을 식별합니다.

Amazon Kinesis Analytics 애플리케이션 로드는 데이터를 한 번만 참조합니다. 데이터가 변경될 경우 데이터를 애플리케이션으로 다시 로드하도록 트리거하는 UpdateApplication 작업을 호출합니다.

내용

BucketARN

S3 버킷의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

FileKey

참조 데이터를 포함하는 객체 키 이름입니다.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1,024.

필수 여부: 예

ReferenceRoleARN

서비스에서 자동으로 데이터를 읽기 위해 수임할 수 있는 IAM 역할의 ARN입니다. 이 역할은 객체에 대해 `s3:GetObject` 작업을 수행할 수 있는 권한과, Amazon Kinesis Analytics 서비스 보안 주체가 이 역할을 수임하도록 허용하는 신뢰 정책이 있어야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSourceDescription

참조 데이터를 저장하는 버킷 명칭과 객체 키 명칭을 제공합니다.

내용

BucketARN

S3 버킷의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

FileKey

Amazon S3 객체 키 명칭.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1,024.

필수 여부: 예

ReferenceRoleARN

Amazon Kinesis Analytics가 사용자를 대신하여 Amazon S3 객체를 읽고 애플리케이션 내 참조 표를 채울 수 있는 IAM 역할의 ARN입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

Pattern: `arn:.*`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

S3ReferenceDataSourceUpdate

Amazon Kinesis Analytics가 사용자를 대신하여 Amazon S3 객체를 읽고 애플리케이션 내 참조 표를 채우는 데 맡길 수 있는 S3 버킷 명칭, 객체 키 명칭 및 IAM 역할을 설명합니다.

내용

BucketARNUpdate

S3 버킷의 Amazon 리소스 이름(ARN).

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

Required: No

FileKeyUpdate

객체 키 명칭.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 1024.

필수 여부: 아니요

ReferenceRoleARNUpdate

Amazon S3 객체를 읽고 애플리케이션 내 참조 표를 채우기 위해 Amazon Kinesis Analytics가 취할 수 있는 IAM 역할의 ARN.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 2,048.

패턴: `arn:.*`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SourceSchema

스트리밍 소스에서 데이터의 형식 및 각 데이터 요소가 애플리케이션 내 스트림에서 생성된 해당 열에 매핑되는 방법을 설명합니다.

내용

RecordColumns

`RecordColumn` 객체의 목록.

유형: RecordColumn객체 어레이

배열 멤버: 최소 항목 수는 1입니다. 최대 항목 수는 1000입니다.

필수 여부: 예

RecordFormat

스트리밍 소스에서 레코드의 형식을 지정합니다.

유형: RecordFormat 객체

필수 여부: 예

RecordEncoding

스트리밍 소스에서 레코드의 인코딩을 지정합니다. 예: UTF-8.

유형: String

패턴: `UTF-8`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Tag

정의하여 AWS 리소스에 할당할 수 있는 키-값 페어(값은 선택 사항)입니다. 이미 있는 태그를 지정하는 경우, 태그 값은 요청에서 지정한 값으로 대체됩니다. 애플리케이션 태그의 최대 수는 시스템 태그를 포함합니다. 사용자 정의 애플리케이션 태그의 최대 수는 50입니다. 자세한 설명은 태그 사용하기를 참조하십시오.

내용

Key

키값 태그의 키.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 128.

필수 여부: 예

Value

키값 태그의 값. 값은 옵션입니다.

유형: 문자열

길이 제약: 최소 길이는 0. 최대 길이 256.

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon Kinesis Data Analytics에 대한 설명서 이력

다음 표는 Amazon Kinesis Data Analytics의 최신 릴리스 이후 변경된 중요 사항에 대한 설명입니다.

- API 버전: 2015-08-14
- 최종 설명서 업데이트: 2019년 5월 8일

변경 사항	설명	날짜
Kinesis Data Analytics 애플리케이션에 태그 지정하기	애플리케이션 태그 지정을 사용하여 애플리케이션별 비용, 액세스 통제 또는 사용자 정의 용도를 결정합니다. 자세한 설명은 태그 지정 사용 섹션을 참조하십시오.	2019년 5월 8일
를 사용하여 Kinesis Data Analytics API 호출 로깅 AWS CloudTrail	Amazon Kinesis Data Analytics는 Kinesis Data Analytics에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스인와 통합됩니다. 자세한 내용은 사용 AWS CloudTrail 단원을 참조하십시오.	2019년 3월 22일
Kinesis Data Analytics는 프랑크푸르트 리전에서 사용할 수 있습니다.	Kinesis Analytics는 유럽(프랑크푸르트) 리전에서 사용할 수 있습니다. 자세한 설명은 및 엔드포인트: Kinesis Data Analytics를 참조하십시오.	2018년 7월 18일
콘솔에서 준거 데이터 사용	이제 콘솔에서 애플리케이션 준거 데이터로 작업할 수 있습니다. 자세한 정보는 예: 참조 데이터를 Kinesis Data	2018년 7월 13일

변경 사항	설명	날짜
	Analytics 애플리케이션에 추가을 참조하십시오.	
창 모드 쿼리 예	창 및 집계를 위한 애플리케이션 예입니다. 자세한 정보는 예: 윈도우 및 집계을 참조하십시오.	2018년 7월 9일
애플리케이션 테스트	애플리케이션 스키마 및 코드의 변경 테스트에 대한 지침입니다. 자세한 정보는 애플리케이션 테스트을 참조하십시오.	2018년 7월 3일
데이터를 사전 처리하기 위한 애플리케이션 예	REGEX_LOG_PARSE, REGEX_REPLACE 및 DateTime 연산자에 대한 추가 코드 예. 자세한 정보는 예: 데이터 변환을 참조하십시오.	2018년 5월 18일
반환 행 및 SQL 코드 크기의 증가	반환 행의 크기 제한이 512KB로 증가하고 애플리케이션의 SQL 코드의 크기 제한은 100KB로 증가했습니다. 자세한 설명은 한도 섹션을 참조하십시오.	2018년 5월 2일
AWS Lambda Java 및 .NET의 함수 예제	레코드 사전 처리 및 애플리케이션 목적지용 Lambda 함수 생성 코드 샘플입니다. 자세한 정보는 사전 처리용 Lambda 함수 생성 및 애플리케이션 목적지용 Lambda 함수 생성 섹션을 참조하십시오.	2018년 3월 22일

변경 사항	설명	날짜
새 HOTSPOTS 함수	데이터에서 상대적으로 밀도가 높은 리전에 대한 정보를 찾아 반환합니다. 자세한 설명은 예: 스트림에서 핫스팟 감지(HOTSPOTS 함수) 섹션을 참조하십시오.	2018년 3월 19일
대상으로서 Lambda 함수	분석 결과를 대상인 Lambda 함수로 전송합니다. 자세한 설명은 Lambda 함수를 출력으로 사용 섹션을 참조하십시오.	2017년 12월 20일
새 RANDOM_CUT_FOREST_WITH_EXPLANATION 함수	데이터 스트림의 이례적인 점수에 영향을 미치는 필드에 대해 설명합니다. 자세한 설명은 예: 데이터 변칙 감지 및 설명(RANDOM_CUT_FOREST_WITH_EXPLANATION 함수) 섹션을 참조하십시오.	2017년 11월 2일
정적 데이터에서 스키마 검색	Amazon S3 버킷에 저장된 정적 데이터에서 스키마 검색을 실행합니다. 자세한 설명은 정적 데이터에 대해 스키마 검색 기능 사용 섹션을 참조하십시오.	2017년 10월 6일
Lambda 사전 처리 기능	분석 AWS Lambda 전를 사용하여 입력 스트림의 레코드를 사전 처리합니다. 자세한 내용은 Lambda 함수를 사용하여 데이터 사전 처리 단원을 참조하십시오.	2017년 10월 6일

변경 사항	설명	날짜
Auto Scaling 애플리케이션	Auto Scaling을 사용하여 애플리케이션의 데이터 처리량을 자동으로 늘립니다. 자세한 설명은 처리량 증가를 위해 애플리케이션 용량을 자동으로 확장 또는 축소 섹션을 참조하십시오.	2017년 9월 13일
여러 애플리케이션 내 입력 스트림	여러 애플리케이션 내 스트림을 사용하여 애플리케이션 처리량을 늘립니다. 자세한 설명은 입력 스트림 병렬화를 통한 처리량 증대 섹션을 참조하십시오.	2017년 6월 29일
Kinesis Data Analytics AWS Management Console 용 사용 설명서	Kinesis Data Analytics 콘솔의 스키마 편집기 및 SQL 편집기를 사용하여 유추된 스키마 및 SQL 코드를 편집합니다. 자세한 설명은 4단계 (선택 사항) 콘솔을 이용한 스키마와 SQL 코드 편집 섹션을 참조하십시오.	2017년 7월 4일
공개 릴리스	Amazon Kinesis Data Analytics 개발자 가이드가 공개되었습니다.	2016년 8월 11일
프리뷰 릴리스	Amazon Kinesis Data Analytics 개발자 가이드의 프리뷰 릴리스	2016년 1월 29일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.