Amazon Kinesis Data Analytics 开发者指南

SQL 开发人员指南

亚马逊云科技

Amazon Kinesis Data Analytics 开发者指南: SQL 开发人员指南

Table of Contents

对于新项目，我们建议您使用新的 Kinesis Data Analytics 工作室，而不是 SQL 应用程序的 Kinesis Data Analytics。Kinesis Data Analytics Studio 将易用性与高级分析功能相结合，使您能够在几分钟内构建复杂的流处理应用程序。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

什么是 SQL 应用程序的Amazon Kinesis Data Analytics？

借端体验的Amazon Kinesis 使您能够使用标准 SQL 来处理和分析流数据。您可以使用该服务针对流式传输源快速编写和运行强大的 SQL 代码，以执行时间序列分析，为实时控制面板提供信息以及创建实时指标。

要开始使用 Kinesis Data Analytics，您需要创建一个持续读取和处理流数据的 Kinesis 数据分析应用程序。该服务支持从亚马逊 Kinesis Data Streams 和亚马逊 Kinesis Data Firehose 流媒体源提取数据。然后，您可以使用交互式编辑器编写 SQL 代码，并使用实时流数据测试它。您还可以配置希望 Kinesis Data Analytics 将结果发送到的目的地。

Kinesis Data Analytics 支持Amazon Kinesis Data Firehose（Amazon S3、Amazon Redshift ft、亚马逊 OpenSearch 服务和 Splunk）和Amazon Kinesis Data Streams 作为目的地。Amazon Lambda

我应该何时使用Amazon Kinesis Data Analytics 体验的各种

Amazon Kinesis Data Analytics 使您能够快速编写 SQL 代码，以近乎实时的方式持续读取、处理和存储数据。通过对流数据采用标准 SQL 查询，您可以构建转换数据并深入了解这些数据的应用程序。以下是使用 Kinesis Data Analytics 的一些示例场景：

- 生成时间序列分析 — 您可以计算时间窗内的指标，然后通过 Kinesis 数据传输流将值传输到 Amazon S3 或 Amazon Redshift。
- 馈送实时仪表板-您可以向下游发送汇总和处理过的流数据结果，以馈送实时仪表板。
- 创建实时指标-您可以创建自定义指标和触发器，用于实时监控、通知和警报。

有关 Kinesis Data Analytics 支持的 SQL 语言元素的信息，请参阅 Amazon Kinesis Data Analytics SQL 参考资料。

您是Amazon Kinesis Data Analytics 的首次用户吗？

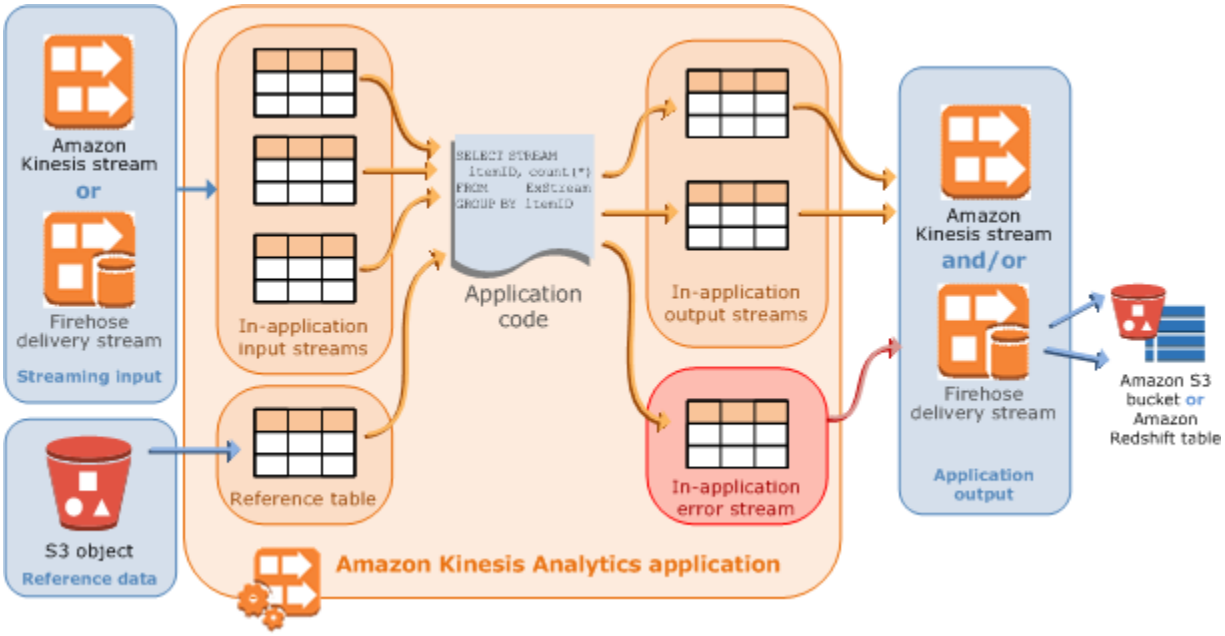
如果您是首次接触 Amazon Kinesis Data Analytics 端体验的用户，我们建议您按顺序阅读以下内容：

1. 阅读本指南的“工作原理”部分。本部分Kinesis Data Analytics 打造端 end-to-end 体验的各种组件。有关更多信息，请参阅适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)：
2. 尝试入门练习。有关更多信息，请参阅Amazon Kinesis Data Analytics 入门 (p. 40)：
3. 了解流式 SQL 概念。有关更多信息，请参阅流式 SQL 概念 (p. 64)：
4. 尝试其他示例。有关更多信息，请参阅 示例应用程序 (p. 78)。

适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理

应用程序是Amazon Kinesis Data Analytics 中的主要资源，您可以在自己的账户中创建该资源。您可以使用 Amazon Web Services Management Console或 Kinesis Data Analytics API 创建和管理应用程序。Kinesis Data Analytics 提供 API 操作来管理应用程序。有关 API 操作的列表，请参阅操作 (p. 199)。

Kinesis Data Analytics 应用程序持续读取和处理流式数据。您可以使用 SQL 编写应用程序代码来处理传入的流数据并生成输出。然后，Kinesis Data Analytics 将输出写入配置的目的地。以下示意图说明典型的应用程序架构。



每个应用程序都有名称、描述、版本 ID 和状态。Amazon Kinesis Data Analytics 会在您首次创建应用程序时分配版本 ID。在您更新任何应用程序配置时，此版本 ID 会更新。例如，如果您添加输入配置、添加或删除参考数据源、添加或删除输出配置或更新应用程序代码，Kinesis Data Analytics 会更新当前应用程序版本 ID。Kinesis Data Analytics 还保留应用程序创建和上次更新的时间戳。

除了这些基本属性外，每个应用程序还包括：

- 输入-应用程序的流媒体源。您可以选择 Kinesis 数据流或 Kinesis Data Firehose 数据传输流。在输入配置中，您可以将流式传输源映射到应用程序内部输入流。应用程序内部流类似于可以对其执行 SELECT 和 INSERT SQL 操作的连续更新表。在您的应用程序代码中，可以创建其他应用程序内部流来存储中间查询结果。

  您可以选择将单个流式传输源分区到多个应用程序内部输入流以提高吞吐量。有关更多信息，请参阅 Limits (p. 189) 和 配置应用程序输入 (p. 4)。

  Amazon Kinesis Data Analytics 在每个应用程序流中提供一个时间戳列，名为时间戳和 ROWTIME 列 (p. 65)。您可以在基于时间的窗口式查询中使用该列。有关更多信息，请参阅窗口式查询 (p. 67)：

您可以选择配置引用数据源，以便丰富您在应用程序中的输入数据流。这会生成应用程序内部引用表。您必须将引用数据存储为 S3 存储桶中的对象。应用程序启动时，Amazon Kinesis Data Analytics 读取 Amazon S3 对象并创建应用程序内部表。有关更多信息，请参阅配置应用程序输入 (p. 4)：

- 应用程序代码-处理输入和生成输出的一系列 SQL 语句。您可以针对应用程序内部流和引用表编写 SQL 语句。您还可以编写 JOIN 查询以组合来自这两个源的数据。

  有关 Kinesis Data Analytics 支持的 SQL 语言元素的信息，请参阅 Amazon Kinesis Data Analytics SQL 参考资料。

  在采用最简单的形式时，应用程序代码可以是单个 SQL 语句，它从流式输入中选择并将结果插入到流式输出中。它还可以是一系列 SQL 语句，将一个 SQL 语句的输出馈送到下一个 SQL 语句的输入。此外，您可以编写应用程序代码以将输入流拆分为多个流。然后，您可以应用其他查询来处理这些流。有关更多信息，请参阅应用程序代码 (p. 28)：

- 输出-在应用程序代码中，查询结果转到应用程序内部流。在您的应用程序代码中，您可以创建一个或多个应用程序内部流保存中间结果。然后，您可以选择配置应用程序输出以在应用程序内部流中永久保存数据，而这些应用程序内部流将应用程序输出（也称为应用程序内部输出流）保存到外部目标中。外部目标可以是 Kinesis Data Firehose 传输流。请注意有关这两种目标的以下信息：
  - 你可以配置 Kinesis Data Firehose 传输流，将结果写入Amazon S3、Amazon Redshift ft 或亚马逊 OpenSearch OpenSearch 服务（服务）。

  - 您也可以将应用程序输出写入自定义目的地，而不是 Amazon S3 或 Amazon Redshift。为此，在输出配置中将 Kinesis 数据流标识为目标。然后，您配置Amazon Lambda为轮询直播并调用您的 Lambda 函数。您的 Lambda 函数代码将流式数据作为输入接收流式数据。在您的 Lambda 函数代码中，您可以将传入的数据写入您的自定义目的地。有关更多信息，请参阅 Amazon Kinesis Data Analytics。Amazon Lambda

  有关更多信息，请参阅配置应用程序输出 (p. 29)：

此外，请注意以下情况：

- Amazon Kinesis Data Analytics 需要权限才能从流媒体源读取记录并将应用程序输出写入外部目的地。您可以使用 IAM 角色授予这些权限。

- Kinesis Data Analytics 自动为每个应用程序提供应用程序内错误流。如果您的应用程序在处理某些记录时出现问题（例如由于类型不匹配或者到达延迟），记录将写入错误流。您可以将应用程序输出配置为指示 Kinesis Data Analytics 将错误流数据保存到外部目标以进行进一步评估。有关更多信息，请参阅错误处理 (p. 36)：

- Amazon Kinesis Data Analytics 可确保将您的应用程序输出记录写入配置的目的地。它“至少一次”使用处理和传输模型，即使在您遇到应用程序中断的情况下也是如此。有关更多信息，请参阅 将应用程序输出永久保存到外部目标的传输模型 (p. 36)。

主题

配置应用程序输入

您的 Amazon Kinesis Data Analytics 应用程序可以接收来自单个流媒体源的输入，也可以选择使用一个参考数据源。有关更多信息，请参阅适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)：本主题的此部分介绍了应用程序输入源。

主题

配置流式传输源

当您创建应用程序时，可以指定流式传输源。您也可以在创建应用程序后修改输入。Amazon Kinesis Data Analytics 支持您的应用程序的以下流媒体源：

- Kinesis 数据流
- Kinesis Data Firehose 传输流

> **Note**
>
> 如果 Kinesis 数据流已加密，Kinesis Data Analytics 无需进一步配置即可无缝访问加密流中的数据。Kinesis Data Analytics 不存储从 Kinesis Data Streams 读取的未加密数据。有关更多信息，请参阅什么是 Kinesis 数据流的服务器端加密？。

Kinesis Data Analytics 持续轮询流媒体源以获取新数据，并根据输入配置将其提取到应用程序内流中。

> **Note**
>
> 添加 Kinesis Stream 作为应用程序的输入不会影响流中的数据。如果其他资源，例如 Kinesis Data Firehose 传输流，也访问了相同的 Kinesis 流，则 Kinesis Data Firehose 传输流和 Kinesis Data Analytics 应用程序都将收到相同的数据。但是，吞吐量和限制可能会受到影响。

您的应用程序代码可以查询应用程序内部流。作为输入配置的一部分，您需要提供以下内容：

- 流的-您提供该传输流的 Amazon 资源名称（ARN）以及 Kinesis Data Analytics 可以代您访问该流的 IAM 角色。
- 应用程序内流名称前缀 — 启动应用程序时，Kinesis Data Analytics 会创建指定的应用程序内流。在您的应用程序代码中，可以使用此名称访问应用程序内部流。

  您可以选择将一个流式传输源映射到多个应用程序内部流。有关更多信息，请参阅Limits (p. 189)：在这种情况下，Amazon Kinesis Data Analytics 会创建指定数量的应用程序内流，其名称如下：*##_001_002###*和*##*_003。默认情况下，Kinesis Data Analytics 将流媒体源映射到一个名为 *p* refix 的应用程序内流_001。

  在应用程序内部流中插入行时有速度限制。因此，Kinesis Data Analytics 支持多个这样的应用程序内流，因此您可以以更快的速度将记录导入应用程序。如果发现应用程序无法及时处理流式传输源中的数据，您可以添加并行度单元以提高性能。
- 映射架构 — 您可以描述流媒体源上的记录格式（JSON、CSV）。您还描述流上的每个记录如何映射到创建的应用程序内部流中的列。您可以在此处提供列名和数据类型。

  Note

  在创建应用程序内输入流时，Kinesis Data Analytics 在标识符（流名称和列名）两侧添加引号。在查询该流和列时，您必须在引号内使用相同的大小写指定它们 (小写和大写字母完全匹配)。有关标识符的更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的标识符。

您可以在 Amazon Kinesis Data Analytics 控制台中创建应用程序和配置输入。然后，控制台执行必需的 API 调用。创建新应用程序 API 或者将输入配置添加到现有应用程序时，可以配置应用程序输入。有关更多信息，请参阅 CreateApplication (p. 214) 和 AddApplicationInput (p. 202)。下面是 `Createapplication` API 请求正文的输入配置部分：

```
"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"
       }
```

```
    ]
```

配置引用源

您还可以选择将引用数据源添加到现有应用程序中，以便扩充来自流式传输源的数据。您必须将引用数据作为对象存储在 Amazon S3 存储桶。应用程序启动时，Amazon Kinesis Data Analytics 读取 Amazon S3 对象并创建应用程序内部参考表。然后，您的应用程序代码可以将其与应用程序内部流联接。

您使用支持的格式（CSV、JSON）将引用数据存储在 Amazon S3 对象中。例如，假设您的应用程序对股票订单执行分析。对流式传输源采用以下记录格式：

```
Ticker, SalePrice, OrderId

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

在这种情况下，您可能考虑维护引用数据源，以提供有关每个股票行情机的详细信息，如公司名称。

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

您可以使用 API 或控制台添加应用程序参考数据源。Amazon Kinesis Data Analytics 提供以下 API 操作来管理参考数据源：

- AddApplicationReferenceDataSource (p. 211)
- UpdateApplication (p. 249)

有关使用控制台添加引用数据的信息，请参阅示例：向 Kinesis Data Analytics 应用程序添加参考数据 (p. 133)。

请注意以下几点：

- 如果应用程序正在运行，Kinesis Data Analytics 会创建应用程序内参考表，然后立即加载参考数据。
- 如果应用程序未运行（例如，它处于就绪状态），Kinesis Data Analytics 仅保存更新的输入配置。当应用程序开始运行时，Kinesis Data Analytics 会以表的形式加载应用程序中的参考数据。

假设你想在 Kinesis Data Analytics 创建应用程序内参考表后刷新数据。也许你更新了 Amazon S3 对象，或者你想使用不同的 Amazon S3 对象。在这种情况下，可以在控制台中显式调用UpdateApplication (p. 249)，也可以选择 “操作”、“同步参考数据表”。Kinesis Data Analytics 不会自动刷新应用程序内的参考表。

您可以创建作为参考数据源的 Amazon S3 对象的大小有限制。有关更多信息，请参阅Limits (p. 189)：如果对象大小超过限制，Kinesis Data Analytics 将无法加载数据。应用程序状态显示为正在运行，但是不读取数据。

当添加引用数据源时，您需要提供以下信息：

- S3 存储段和对象密钥名称 — 除了存储段名称和对象密钥外，您还提供一个 IAM 角色，Kinesis Data Analytics 可以假定该角色代表您读取对象。
- 应用程序内参考表名称 — Kinesis Data Analytics 创建此应用程序内表并通过读取 Amazon S3 对象来填充该表。这是您在应用程序代码中指定的表名称。

- 映射架构 — 您可以描述存储在 Amazon S3 对象中的数据的记录格式（JSON、CSV）和编码。您还可以描述每个对象元素如何映射到应用程序内部引用表中的列。

下面显示 AddApplicationReferenceDataSource API 请求中的请求正文。

```
{
    "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"
    }
}
```

使用 JSONPath

JSONPath 是查询 JSON 对象的元素的标准化方式。JSONPath 使用表达式在 JSON 文档中的元素、嵌套元素和数组之间导航。有关 JSON 的更多信息，请参阅 JSON 简介。

Amazon Kinesis Data Analytics 在应用程序的源架构中使用 JSONPath 表达式来识别包含 JSON 格式数据的流媒体源中的数据元素。

有关如何将流式数据映射到应用程序输入流的更多信息，请参阅the section called “将流式传输源元素映射到 SQL 输入列” (p. 11)。

使用 JSONPath 访问 JSON 元素

接下来，您将了解如何使用 JSONPath 表达式访问 JSON 格式的数据中的各种元素。对于此部分中的示例，请假设源流包含以下 JSON 记录：

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

访问 JSON 元素

要使用 JSONPath 查询 JSON 数据中的元素，请使用以下语法。在此处，$ 表示数据层次结构的根，elementName 是要查询的元素节点的名称。

```
$.elementName
```

以下表达式将查询前面的 JSON 示例中的 customerName 元素。

```
$.customerName
```

前面的表达式将从前面的 JSON 记录返回以下内容。

```
John Doe
```

> **Note**
>
> 路径表达式区分大小写。表达式 $.customername 将从前面的 JSON 示例返回 null。

> **Note**
>
> 如果路径表达式指定的位置没有出现任何元素，则表达式返回 null。以下表达式将从前面的 JSON 示例返回 null，因为没有匹配元素。

```
$.customerId
```

访问嵌套 JSON 元素

要查询嵌套 JSON 元素，请使用以下语法。

```
$.parentElement.element
```

以下表达式将查询前面的 JSON 示例中的 city 元素。

```
$.address.city
```

前面的表达式将从前面的 JSON 记录返回以下内容。

```
Anytown
```

您可以使用以下语句进一步查询子元素。

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

以下表达式将查询前面的 JSON 示例中的 `street` 元素。

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

前面的表达式将从前面的 JSON 记录返回以下内容。

```
AnyStreet
```

访问数组

您可以通过以下方式访问 JSON 数组中的数据：

- 将数组中的所有元素作为单一的行进行检索。
- 将数组中的每个元素作为单独的行进行检索。

以单个行检索数组中的所有元素

要以单个行查询一个数组的全部内容，请使用以下语法。

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

以下表达式将查询本部分前面所用 JSON 示例中的 `orders` 元素的全部内容。它将返回单个行的单个列中的数组内容。

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

上述表达式从本部分所用的示例 JSON 记录返回如下内容。

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

以单独的行分别检索数组中的所有元素

要以单独的行分别查询一个数组中的各个元素，请使用以下语法。

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

以下表达式将查询前面的 JSON 示例中的 `orderId` 元素，并将每个数组元素作为一个单独的行返回。

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

前面的表达式将从前面的 JSON 记录返回以下内容，其中每个数据项都将作为一个单独的行返回。

23284
63122
77284

Note

如果查询非数组元素的表达式包含在查询各个数组元素的架构中，则非数组元素将针对数组中的每个元素重复。例如，假设前面的 JSON 示例的架构包含以下表达式：

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

在这种情况下，从示例输入流元素返回的数据行将类似于以下内容 (其中的 `name` 元素将针对每个 `orderId` 元素重复)。

John Doe	23284
John Doe	63122
John Doe	77284

Note

以下限制适用于Amazon Kinesis Data Analytics 中的数组表达式：

- 数组表达式中仅支持一个级别的解除引用。不支持以下表达式格式。

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

- 一个架构中仅可平展一个数组。可以引用多个数组——以包含数组中所有元素的一行形式返回。但是，只有一个数组可以让其每个元素都作为单个行返回。

  包含以下格式的元素的架构有效。此格式会将第二个数组的内容作为单个列返回，该内容针对第一个数组中的每一个元素重复。

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

  包含以下格式的元素的架构无效。

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

其他考虑因素

使用 JSONPath 的其他注意事项如下所示：

- 如果应用程序架构中的 JSONPath 表达式中的单个元素没有访问任何数组，则会在应用程序的输入流中为处理的每个 JSON 记录创建一行。
- 当数组被平展后（即，它的元素以单独的行返回），任何缺失的元素都会导致在应用程序内部流中创建一个 null 值。
- 一个数组将始终被平展为至少一行。如果不会返回任何值 (即，数组为空或没有查询数组的任何元素)，则会返回包含所有 null 值的单个行。

  以下表达式将从前面的 JSON 示例返回包含 null 值的记录，因为在指定的路径没有匹配的元素。

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

  前面的表达式将从前面的 JSON 示例记录返回以下内容。

null
null
null

相关主题

- 介绍 JSON

将流式传输源元素映射到 SQL 输入列

借助 Amazon Kinesis Data Analytics，您可以使用标准 SQL 处理和分析 JSON 或 CSV 格式的流媒体数据。

- 要处理和分析流 CSV 数据，您可以为输入流的列分配列名称和数据类型。您的应用程序将按顺序从每个列定义的输入流中导入一个列。

  您不需要在应用程序输入流中包含所有列，但您不能跳过源流中的列。例如，您可以从包含五个列的一个输入流中导入前三个列，但不能进导入列 1、2 和 4。
- 要处理和分析流 JSON 数据，您可以使用 JSONPath 表达式将流式传输源中的 JSON 元素映射到输入流中的 SQL 列。有关将 JSonPath 与 Amazon Kinesis Data Analytics 一起使用的更多信息，请参阅使用 JSONPath (p. 7)。SQL 表中的列具有从 JSON 类型中映射的数据类型。有关支持的数据类型，请参阅数据类型。有关将 JSON 数据转换成 SQL 数据的详细信息，请参阅将 JSON 数据类型映射到 SQL 数据类型 (p. 13)。

有关如何配置输入流的详细信息，请参阅配置应用程序输入 (p. 4)。

将 JSON 数据映射到 SQL 列

您可以使用Amazon Web Services Management Console或 Kinesis Data Analytics API 将 JSON 元素映射到输入列。

- 要使用控制台将元素映射到列，请参阅使用架构编辑器 (p. 54)。
- 要使用 Kinesis Data Analytics API 将元素映射到列，请参阅以下部分。

要将 JSON 元素映射到应用程序内部输入流中的列，您需要使用在每个列中包含以下信息的架构：

- 源表达式：指定列的数据位置的 JSONPath 表达式。

- 列名称：您的 SQL 查询用于引用数据的名称。
- 数据类型：列的 SQL 数据类型。

使用 API

要将流媒体源中的元素映射到输入列，可以使用 Kinesis Data Analytics APICreateApplication (p. 214) 操作。要创建应用程序内部流，请指定一个架构以将您的数据转换为 SQL 中使用的架构化版本。CreateApplication (p. 214) 操作会将您的应用程序配置为接收来自单个流式传输源的输入。要将 JSON 元素或 CSV 列映射到 SQL 列，您应在 SourceSchema (p. 313) `RecordColumns` 数组中创建一个 RecordColumn (p. 304) 对象。RecordColumn (p. 304) 对象具有以下架构：

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

RecordColumn (p. 304) 对象中的字段具有以下值：

- `Mapping`：用于识别列输入源记录中数据的位置的 JSONPath 表达式。采用 CSV 格式的源流的输入架构不存在此值。
- `Name`：应用程序内 SQL 数据流中的列名称。
- `SqlType`：应用程序内 SQL 数据流中的数据的数据类型。

JSON 输入架构示例

以下示例展示了 JSON 架构的 `InputSchema` 值的格式。

```
"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 输入架构示例

以下示例展示了采用逗号分隔值 (CSV) 格式的架构的 `InputSchema` 值的格式。

```
"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"
}
```

将 JSON 数据类型映射到 SQL 数据类型

JSON 数据类型将根据应用程序的输入架构转换为相应的 SQL 输入类型。有关支持的 SQL 数据类型的信息，请参阅数据类型。Amazon Kinesis Data Analytics 根据以下规则将 JSON 数据类型转换为 SQL 数据类型。

Null 文本

无论目标数据类型如何，JSON 输入流中的 null 文本 (即，`"City":null`) 都将转换为 SQL null。

布尔文本

JSON 输入流中的布尔文本 (即 `"Contacted":true`) 将按以下形式转换为 SQL 数据：

- 数值 (DECIMAL、INT 等)：`true` 转换为 1；`false` 转换为 0.
- 二进制 (BINARY 或 VARBINARY)：
  - `true`：结果已设置最低位并清除剩余位。
  - `false`：结果已清除所有位。

  转换为 VARBINARY 将产生长度为 1 个字节的值。

- 布尔值：转换为相应的 SQL 布尔值。
- 字符 (CHAR 或 VARCHAR)：转换为相应的字符串值 (`true` 或 `false`)。值将被截断以适应字段的长度。
- 日期时间 (DATE、TIME 或 TIMESTAMP)：转换将失败，并且一个强制转换错误将写入到错误流。

数字

JSON 输入流中的数字文本 (即 `"CustomerId":67321`) 将按以下形式转换为 SQL 数据：

- 数值 (DECIMAL、INT 等)：直接转换。如果转换后的值超过目标数据类型 (即，将 `123.4` 转换为 INT) 的大小或精度，转换将失败，并且一个强制转换错误将写入到错误流。
- 二进制 (BINARY 或 VARBINARY)：转换将失败，并且一个强制转换错误将写入到错误流。
- BOOLEAN:
  - `0`：转换为 `false`。
  - 所有其他数字：转换为 `true`。
- 字符 (CHAR 或 VARCHAR)：转换为数字的字符串表示形式。
- 日期时间 (DATE、TIME 或 TIMESTAMP)：转换将失败，并且一个强制转换错误将写入到错误流。

字符串

JSON 输入流中的字符串值 (即 `"CustomerName":"John Doe"`) 将按以下形式转换为 SQL 数据：

- 数字 (十进制、整数等) ：Amazon Kinesis Data Analytics 尝试将值转换为目标数据类型。如果该值无法转换，则转换将失败，并且一个强制转换错误将写入到错误流。
- 二进制 (BINARY 或 VARBINARY)：如果源字符串是有效的二进制文本 (即包含偶数数量的 f 的 `X'3F67A23A'`)，则该值将转换为目标数据类型。否则，转换将失败，并且一个强制转换错误将写入到错误流。
- 布尔值：如果源字符串为 `"true"`，则转换为 `true`。此比较不区分大小写。否则，转换为 `false`。
- 字符 (CHAR 或 VARCHAR)：转换为输入中的字符串值。如果该值长于目标数据类型，那么它将被截断，并且任何错误都不会将写入到错误流。
- 日期时间 (DATE、TIME 或 TIMESTAMP)：如果源字符串采用了可转换为目标值的格式，则转换该值。否则，转换将失败，并且一个强制转换错误将写入到错误流。

  有效的日期时间格式包括：
  - “1992-02-14”
  - “1992-02-14 18:35:44.0”

数组或对象

JSON 输入流中的数组或对象将按以下形式转换为 SQL 数据：

- 字符 (CHAR 或 VARCHAR)：转换为数组或对象的源文本。请参阅访问数组 (p. 9)。
- 所有其他数据类型：转换将失败，并且一个强制转换错误将写入到错误流。

有关 JSON 数组的示例，请参阅使用 JSONPath (p. 7)。

相关主题

- 配置应用程序输入 (p. 4)
- 数据类型

- 使用架构编辑器 (p. 54)
- CreateApplication (p. 214)
- RecordColumn (p. 304)
- SourceSchema (p. 313)

针对流数据使用架构发现功能

提供输入架构以描述流输入中的记录映射到应用程序内部流可能非常麻烦，并且容易出错。可以使用 DiscoverInputSchema (p. 233) API（称为发现 API）来推断架构。API 通过使用有关流式传输源的记录的随机示例，可以推断架构（即列名、数据类型和传入数据中数据元素的位置）。

> Note
>
> 要使用 Discovice API 从存储在 Amazon S3 中的某个文件生成架构，请参阅针对静态数据使用架构发现功能 (p. 16)。

控制台使用发现 API 来生成指定流式传输源的架构。利用控制台，您还可以更新架构，包括添加或删除列、更改列名称或数据类型等。不过，请仔细进行更改以确保不会创建无效的架构。

在为应用程序内部流完成架构后，您可以使用一些函数来处理字符串和日期时间值。在生成的应用程序内部流中使用行时，您可以在应用程序代码中利用这些函数。有关更多信息，请参阅示例：转换 DateTime 值 (p. 112)：

架构发现期间的列命名

在架构发现期间，Amazon Kinesis Data Analytics 尝试尽可能多地保留流式传输输入源的原始列名，以下情况除外：

- 源流列名称是预留 SQL 关键字，例如 `TIMESTAMP`、`USER`、`VALUES` 或 `YEAR`。
- 源流列名称包含不受支持的字符。只有字母、数字和下划线字符（_）受支持。
- 源流列名称以数字开头。
- 源流列名称的长度超过 100 个字符。

如果重命名了某个列，则重命名的架构列名称将以 `COL_` 开头。在某些情况下，不能保留任何原始列名，例如，如果整个名称都是不支持的字符。在这种情况下，该列将被命名为 `COL_#`，其中 # 是一个数字，表示该列在列顺序中的位置。

发现完成后，您可以使用控制台更新架构以添加或删除列，也可以更改列名称、数据类型或数据大小。

发现建议的列名称的示例

源流列名称	发现建议的列名称
USER	COL_USER
USER@DOMAIN	COL_USERDOMAIN
@@	COL_0

架构发现问题

如果 Kinesis Data Analytics 没有推断出给定流媒体源的架构，会发生什么？

Kinesis Data Analytics 会推断出常见格式的架构，例如 CSV 和 JSON，它们采用 UTF-8 编码。Kinesis Data Analytics 支持任何带有自定义列和行分隔符的 UTF-8 编码记录（包括应用程序日志和记录等原始文本）。如果 Kinesis Data Analytics 无法推断出架构，则可以使用控制台上的架构编辑器（或使用 API）手动定义架构。

如果您的数据没有遵循模式 (可使用架构编辑器指定)，您可以将架构定义为单个 VARCHAR(N) 类型的列，其中 N 是您希望记录包含的最大字符数。在此处，当数据位于应用程序内部流中后，您可以使用字符串和日期-时间操作来构造数据。有关示例，请参阅 示例：转换 DateTime 值 (p. 112)。

针对静态数据使用架构发现功能

架构发现功能可以根据流中的数据或存储在 Amazon S3 存储桶中的静态文件中的数据生成架构。假设您想要为 Kinesis Data Analytics 应用程序生成架构以供参考或在直播数据不可用时使用。您可以对包含流式数据或参考数据预期格式的数据样本的静态文件使用架构发现功能。Kinesis Data Analytics 可以对存储在 Amazon S3 存储桶中的 JSON 或 CSV 文件中的示例数据运行架构发现。要针对数据文件使用架构发现功能，需使用控制台或指定了 DiscoverInputSchema (p. 233) 参数的 `S3Configuration` API。

使用控制台运行架构发现

要使用控制台对静态文件运行发现功能，请执行以下操作：

1. 向 S3 存储桶添加引用数据对象。
2. 在 Kinesis Data Analytics 控制台的应用程序主页中选择 “Connect 参考数据”。
3. 提供存储桶、路径和 IAM 角色数据，用于访问包含参考数据的 Amazon S3 对象。
4. 选择 Discover schema (发现架构)。

有关如何在控制台中添加引用数据和发现架构的更多信息，请参阅示例：向 Kinesis Data Analytics 应用程序添加参考数据 (p. 133)。

使用 API 运行架构发现

要使用 API 针对静态文件运行发现，您需要为 API 提供具有以下信息的 `S3Configuration` 结构：

- `BucketARN`：包含该文件的 Amazon S3 存储桶的 Amazon 资源名称（ARN）。有关 Amazon S3 存储桶 ARN 的格式，请参阅 Amazon 资源名称（ARN）和Amazon Service Simple Storage Service (Amazon S3)。
- `RoleARN`：包含`AmazonS3ReadOnlyAccess`策略的 IAM 角色的 ARN。有关如何将策略添加到角色的信息，请参阅修改角色。
- `FileKey`：对象的文件名称。

使用**`DiscoverInputSchema`** API 从 Amazon S3 对象生成架构

1. 确保您设置了 Amazon CLI。有关更多信息，请参阅“入门”部分中的步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 42)。
2. 使用以下内容创建名为 `data.csv` 的文件：

```
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. 通过 https://console.aws.amazon.com/s3/ 登录到 Amazon S3 控制台。

4. 创建 Amazon S3 存储桶并上载您创建的`data.csv`文件。请记下所创建存储桶的 ARN。有关创建 Amazon S3 存储桶并上载文件的信息，请参阅 Ama zon Simple Storage Service 入门。
5. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。创建具有 `AmazonS3ReadOnlyAccess` 策略的角色。记下新角色的 ARN。有关创建 Amazon Service（Amazon）创建向 Amazon Service 委派权限的角色。有关如何将策略添加到角色的信息，请参阅修改角色。
6. 在中运行以下`DiscoverInputSchema`命令Amazon CLI，将 ARN 替换为您的 Amazon S3 存储桶和 IAM 角色：

```
$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. 该响应应该类似于下列内容：

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

使用 Lambda 函数预处理数据

如果流中的数据需要转变格式、转换、扩充或筛选，您可以使用 Amazon Lambda 函数预处理数据。您可以在执行应用程序 SQL 代码或应用程序通过数据流创建架构之前执行此操作。

使用 Lambda 函数预处理记录在以下场景中很有用：

- 将记录从其他格式（例如 KPL 或 GZIP）转换为 Kinesis Data Analytics 可以分析的格式。Kinesis Data Analytics 目前支持 JSON 或 CSV 数据格式。
- 将数据扩展为聚合或异常检测等操作更易访问的格式。例如，如果多个数据值存储在同一字符串中，您可以将数据展开为多个分开的列。
- 使用其他亚马逊服务（例如外推或错误更正）丰富数据。
- 将复杂的字符串转换应用于记录字段。
- 用于整理数据的数据筛选。

使用 Lambda 函数预处理记录

创建 Kinesis Data Analytics 应用程序时，您可以在 “Connect 源” 页面中启用 Lambda 预处理。

使用 Lambda 函数预处理 Kinesis Data Analytics 应用程序中的记录

1. 登录Amazon Web Services Management Console并打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 在应用程序的 “Connect 源” 页面上，在 “记录预处理方式”Amazon Lambda 部分中选择 “启用”。
3. 要使用您已经创建的 Lambda 函数，请在 Lambda 函数下拉列表中选择该函数。
4. 要从其中一个 Lambda 预处理模板创建新的 Lambda 函数，请从下拉列表中选择该模板。然后，选择 View <template name> in Lambda (在 Lambda 中查看 <模板名称>) 以编辑该函数。
5. 要创建新的 Lambda 函数，请选择新建。有关创建 Lambda 函数的信息，请参阅Amazon Lambda开发人员指南中的创建 HelloWorld Lambda 函数和浏览控制台。
6. 选择要使用的 Lambda 函数的版本。要使用最新版本，请选择 $LATEST。

当您选择或创建 Lambda 函数进行记录预处理时，将在执行应用程序 SQL 代码或应用程序根据记录生成架构之前对记录进行预处理。

Lambda 预处理权限

要使用 Lambda 预处理，应用程序的 IAM 角色需要以下权限策略：

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

Lambda 预处理指标

您可以使用 Amazon CloudWatch 监控 Lambda 调用的数量、处理的字节数、成功和失败等。有关 Kinesis Data Analytics Lambda 预处理发布的 CloudWatch 指标的信息，请参阅 Amazon Kinesis Analytics 指标。

Amazon Lambda与 Kinesis 制作人库一起使用

Kinesis Producer Lib rary (KPL) 将用户格式化的小型记录聚合为最大 1 MB 的较大记录，以更好地利用 Amazon Kinesis Data Streams 的吞吐量。适用于 Java 的 Kinesis Client Library (KCL) 支持分解这些记录。但在将 Amazon Lambda 作为流使用者时，必须使用特殊模块取消聚合记录。

要获取必要的项目代码和说明，请参阅Amazon Lambda上的 Kinesis Producer 库分解模块 GitHub。您可以使用此项目中的组件在 Amazon Lambda 中通过 Java、Node.js 和 Python 处理 KPL 序列化数据。您也可以将这些组件用在多语言 KCL 应用程序中。

数据预处理事件输入数据模型/记录响应模型

要预处理记录，您的 Lambda 函数必须符合所需的事件输入数据和记录响应模型。

事件输入数据模型

Kinesis Data Analytics 持续读取来自 Kinesis 数据流或 Kinesis Data Firehose 传输流的数据。对于检索的每批记录，该服务管理如何将每批记录传递给您的 Lambda 函数。您的函数将接收到的记录列表作为输入。在您的函数中，您对列表进行迭代，并应用业务逻辑来完成您的预处理要求 (如数据格式转换或扩充)。

预处理函数的输入模型略有不同，具体取决于数据是从 Kinesis 数据流还是 Kinesis Data Firehose 传输流接收的。

如果源是一个 Kinesis Data Firehose 传输流，则事件输入数据模型如下所示：

Kinesis Data Firehose 请求数据模型

字段	描述
`invocationId`	Lambda 调用 ID（随机 GUID）。
`applicationArn`	Kinesis Data Analytics 应用程序Amazon 资源名称（ARN
`streamArn`	传输流 ARN
记录	

字段	描述	
`recordId`	记录 ID (随机 GUID)	
`kinesisFirehoseRecordMetadata`		

字段	描述	
`approximateArrivalTimestamp`	传输流记录大致到达时间	

字段	描述	
`data`	Base64 编码的源记录负载	

以下示例显示来自 Firehose 传输流的输入：

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

如果源是 Kinesis 数据流，则事件输入数据模型如下所示：

Kinesis 流请求数据模型

字段	描述
invocationId	Lambda 调用 ID（随机 GUID）。
applicationArn	Kinesis Data Analytics 应用程序 ARN
streamArn	传输流 ARN
记录	

字段	描述	
recordId	基于 Kinesis 记录序列号的记录 ID	
kinesisStreamRecordMetadata		

字段	描述	
sequenceNumber	从 Kinesis 流记录中得到的序列号	
partitionKey	从 Kinesis 流记录中得到的分区键	
shardId	从 Kinesis 流记录中得到的 ShardId	
approximateArrivalTimestamp	传输流记录大致到达时间	

data	Base64 编码的源记录负载	

以下示例显示来自 Kinesis 数据流的输入：

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

记录响应模型

必须返回发送到 Lambda 函数，并从您的 Lambda 预处理函数返回的所有记录 (带记录 ID)。它们必须包含以下参数，否则 Kinesis Data Analytics 会拒绝这些参数并将其视为数据预处理故障。可对记录的数据负载部分进行转换，以满足预处理要求。

响应数据模型

记录

字段	描述
`recordId`	在调用期间，记录 ID 从 Kinesis Data Analytics 传递到 Lambda。转换后的记录必须包含相同记录 ID。原始记录的 ID 和转换记录的 ID 之间如果有不匹配，将被视为数据预处理失败。
`result`	记录的数据转换的状态。可能的值包括： • `Ok`：记录已成功转换。Kinesis Data Analytics 提取记录进行 SQL 处理。 • `Dropped`：该记录是您的处理逻辑故意删除的。Kinesis Data Analytics 从 SQL 处理中删除记录。对于 `Dropped` 记录，数据负载字段是可选的。 • `ProcessingFailed`：记录无法转换。Kinesis Data Analytics 认为您的 Lambda 函数未成功对其进行处理，并将错误写入错误流。有关错误流的更多信息，请参阅错误处理 (p. 36)。对于 `ProcessingFailed` 记录，数据负载字段是可选的。
`data`	转换后的数据负载 (使用 base64 编码之后)。如果应用程序提取数据格式为 JSON，则每个数据负载可能包含多个 JSON 文档。或者，如果应用程序提取数据格式为 CSV，则每个数据负载可能包含多个 CSV 行 (在每一行中指定行分隔符)。Kinesis Data Analytics 服务成功解析和处理同一数据负载中包含多个 JSON 文档或 CSV 行的数据。

以下示例显示来自 Lambda 函数的输出：

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

```
}
```

常见的数据预处理失败情况

以下是预处理失败的常见原因。

- 并非批量中发送到 Lambda 函数的所有记录（带有记录 ID）都会返回到 Kinesis Data Analytics 服务。
- 响应中缺少记录 ID、状态或数据负载字段。对于 `Dropped` 或 `ProcessingFailed` 记录，数据负载字段是可选的。
- Lambda 函数超时不足以预处理数据。
- Lambda 函数的响应超过了Amazon Lambda服务施加的响应限制。

对于数据预处理失败，Kinesis Data Analytics 会继续对同一组记录重试 Lambda 调用，直到成功为止。您可以监控以下 CloudWatch 指标，以深入了解故障。

- Kinesis Data Analytics 应用程序`MillisBehindLatest`：表示应用程序在读取流媒体源时落后多远。
- Kinesis Data Analytics 应用程序`InputPreprocessing` CloudWatch 指标：指明成功和失败的数量以及其他统计数据。有关更多信息，请参阅 Amazon Kinesis Analytics 指标。
- Amazon Lambda函数 CloudWatch 指标和日志。

创建 Lambda 函数以进行预处理

您的 Amazon Kinesis Data Analytics 应用程序可以在将记录摄入应用程序时使用 Lambda 函数对记录进行预处理。Kinesis Data Analytics 在控制台上提供以下模板，用作预处理数据的起点。

主题

在 Node.js 中创建预处理 Lambda 函数

Kinesis Data Analytics 控制台上提供了以下用于在 Node.js 中创建预处理 Lambda 函数的模板：

Lambda 蓝图	语言和版本	描述
通用 Kinesis Data Analytics 输入处理	Node.js 6.10	一种 Kinesis Data Analytics 记录预处理器，它接收 JSON 或 CSV 记录作为输入，然后将其与处理状态一起返回。使用此处理器作为自定义转换逻辑的起点。
压缩输入处理	Node.js 6.10	一种 Kinesis Data Analytics 记录处理器，它接收压缩（GZIP 或 Deflate 压缩）JSON 或 CSV 记录作为输入，并返回具有处理状态的解压缩记录。

在 Python 中创建预处理 Lambda 函数

控制台上提供了以下用于在 Python 中创建预处理 Lambda 函数的模板：

Lambda 蓝图	语言和版本	描述
通用 Kinesis Analytics 输入处理	Python 2.7	一种 Kinesis Data Analytics 记录预处理器，它接收 JSON 或 CSV 记录作为输入，然后将其与处理状态一起返回。使用此处理器作为自定义转换逻辑的起点。
KPL 输入处理	Python 2.7	一个 Kinesis Data Analytics 记录处理器，它接收 Kinesis Producer Library (KPL) 将 JSON 或 CSV 记录聚合作为输入，并返回具有处理状态的分解记录。

在 Java 中创建预处理 Lambda 函数

要在 Java 中创建用于预处理记录的 Lambda 函数，请使用 Java 事件类。

以下代码演示了使用 Java 预处理记录的 Lambda 函数示例：

```
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;
    }

}
```

在.NET 中创建预处理 Lambda 函数

要在.NET 中创建 Lambda 函数来预处理记录，请使用 .NET 事件类。

以下代码演示了使用 C# 预处理记录的 Lambda 函数示例：

```
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;
        }
    }
```

有关创建 Lambda 函数以用于预处理并在 .NET 中的目标的更多信息，请参阅Amazon.Lambda.KinesisAnalyticsEvents。

并行处理输入流以增加吞吐量

Amazon Kinesis Data Analytics 应用程序可以支持多个应用程序内输入流，从而将应用程序扩展到单个应用程序内输入流的吞吐量之外。有关应用程序内部输入流的更多信息，请参阅 适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)。

在几乎所有情况下，Amazon Kinesis Data Analytics 都会扩展您的应用程序以处理馈入应用程序的 Kinesis 流或 Kinesis Data Firehose 源流的容量。但是，如果您的源流的吞吐量超出单个应用程序内部输入流的吞吐量，您可显式增加您的应用程序使用的应用程序内部输入流的数量。您需使用 `InputParallelism` 参数执行此操作。

当`InputParallelism`参数大于一时，Amazon Kinesis Data Analytics 会在应用程序内流之间平均分配源流的分区。例如，如果您的源流具有 50 个分片，并且您已将 `InputParallelism` 设置为 2，则每个应用程序内部输入流都将收到来自 25 个源流分片的输入。

当您增加应用程序内部流的数量后，您的应用程序必须显式访问每个流中的数据。有关通过代码访问多个应用程序内部流的信息，请参阅在您的 Amazon Kinesis Data Analytics 应用程序中访问单独的应用程序内流 (p. 27)。

尽管 Kinesis Data Streams 和 Kinesis Data Firehose 流分片在应用程序内流之间划分的方式相同，但它们在应用程序中的显示方式有所不同：

- 来自 Kinesis 数据流的记录包括一个`shard_id`字段，该字段可用于识别记录的源分片。

- 来自 Kinesis Data Firehose 传输流的记录不包含标识记录源分片或分区的字段。这是因为 Kinesis Data Firehose 会将这些信息从您的应用程序中抽象出来。

评估是否增加您的应用程序内部输入流的数量

在大多数情况下，单个应用程序内部输入流可处理单个源流的吞吐量，具体取决于输入流的复杂度和数据大小。要确定是否需要增加应用程序内输入流的数量，您可以监控亚马逊中的`InputBytes`和`MillisBehindLatest`指标 CloudWatch。

如果 `InputBytes` 指标大于 100 MB/秒（或您预期该指标将大于此速率），这可能会使 `MillisBehindLatest` 增大并导致应用程序问题的影响扩大。为解决此问题，我们建议您为应用程序选择以下语言：

- 如果您的应用程序的扩展需求超过 100 MB/秒，请对 SQL 应用程序使用多个流和 Kinesis Data Analytics。
- 如果您想使用单个数据流和应用程序，请使用适用于 Java 应用程序的 Kinesis Data Analytics。

如果 `MillisBehindLatest` 指标具有以下任一特征，则应调高您的应用程序的 `InputParallelism` 设置：

- `MillisBehindLatest` 指标逐渐增大，指示您的应用程序落后于流中的最新数据。
- `MillisBehindLatest` 指标始终高于 1000 (1 秒)。

如果满足以下条件，您无需调高您的应用程序的 `InputParallelism` 设置：

- `MillisBehindLatest` 指标逐渐减小，指示您的应用程序跟上了流中的最新数据。
- `MillisBehindLatest` 指标小于 1000 (1 秒)。

有关使用的更多信息 CloudWatch，请参阅CloudWatch 用户指南。

实施多个应用程序内部输入流

如果应用程序是使用 CreateApplication (p. 214) 创建的，您可以设置应用程序内部输入流的数量。您应在使用UpdateApplication (p. 249)创建应用程序之后设置此数量。

> Note
>
> 您只能使用Amazon Kinesis Data Analytics API 或Amazon CLI.`InputParallelism` 您无法使用 Amazon Web Services Management Console设置该设置。有关设置 Amazon CLI 的信息，请参阅步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 42)。

设置新应用程序的输入流计数

以下示例说明了如何使用 `CreateApplication` API 操作将新应用程序的输入流计数设置为 2。

有关 `CreateApplication` 的更多信息，请参阅 CreateApplication (p. 214)。

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

设置现有应用程序的输入流计数

以下示例说明了如何使用 UpdateApplication API 操作将现有应用程序的输入流计数设置为 2。

有关 Update_Application 的更多信息，请参阅 UpdateApplication (p. 249)。

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

在您的 Amazon Kinesis Data Analytics 应用程序中访问单独的应用程序内流

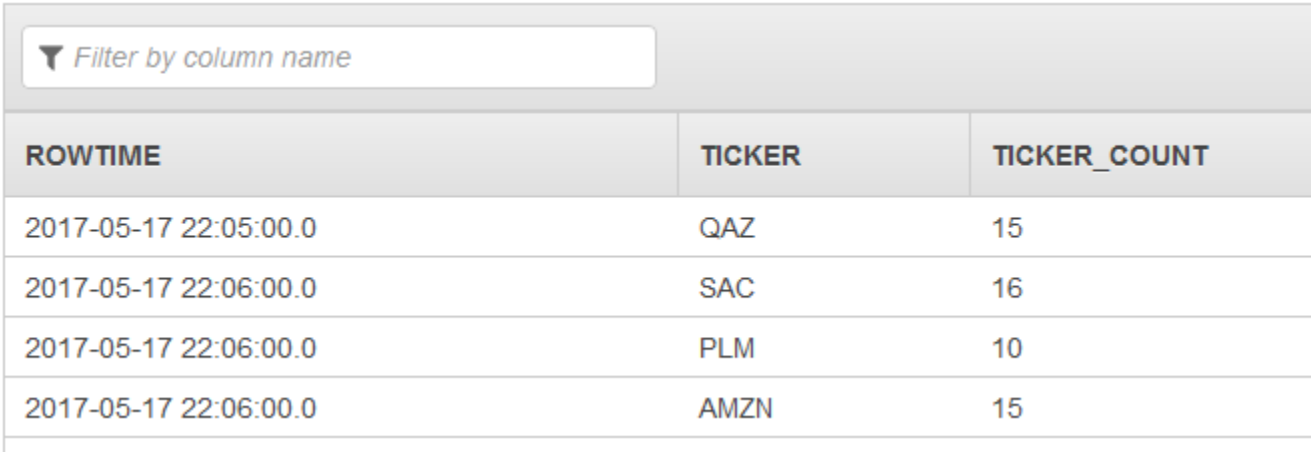
要使用您的应用程序中的多个应用程序内部输入流，您必须从不同的流中显式选择。以下代码示例说明了如何在创建于“入门”教程中的应用程序中查询多个输入流。

在以下示例中，首先使用 CO UNT 聚合每个源流，然后合并为一个名为的应用程序内流in_application_stream001。预先聚合源流有助于确保组合的应用程序内部流可处理来自多个流的流量而不会过载。

> **Note**
>
> 要运行此示例并获得来自应用程序内部输入流的结果，请更新源流中的分片数和应用程序中的 InputParallelism 参数。

```
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;
```

前面的代码示例将在 in_application_stream001 中生成类似于下面的输出：



其他注意事项

在使用多个输入流时，请注意以下事项：

- 应用程序内部输入流的最大数量为 64。
- 应用程序内部输入流在应用程序的输入流的分片之间均匀分配。
- 通过添加应用程序内部流获得的性能改进无法线性扩展。也就是说，使应用程序内部流的数量加倍不会使吞吐量加倍。对于典型行大小，每个应用程序内部流可实现每秒大约 5,000 到 15,000 行的吞吐量。通过将应用程序内部流计数增加到 10，您可以实现每秒 20,000 到 30,000 行的吞吐量。吞吐量流速取决于输入流中的字段的计数、数据类型和数据大小。
- 某些聚合函数（如 AVG）在应用于分区到不同分片中的输入流时可能生成意外结果。由于您需要在将各个分片组合到聚合流中之前对它们运行聚合操作，因此结果可能向包含更多记录的流加权。
- 如果在增加输入流数量后，您的应用程序继续遇到性能不佳（反映在高`MillisBehindLatest`指标上），则可能已经达到了 Kinesis 处理单元 (KPU) 的极限。有关更多信息，请参阅 自动扩展应用程序以提高吞吐量 (p. 37)。

应用程序代码

应用程序代码是处理输入和生产输出的一系列 SQL 语句。这些 SQL 语句运行于应用程序内部流和引用表中。有关更多信息，请参阅适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)：

有关 Kinesis Data Analytics 支持的 SQL 语言元素的信息，请参阅 Amazon Kinesis Data Analytics SQL 参考资料。

在关系数据库中，可使用 INSERT 语句添加记录和使用 SELECT 语句查询数据来处理表。在Amazon Kinesis Data Analytics 中，您可以使用直播。可以编写 SQL 语句来查询这些流。查询一个应用程序内部流所获得的结果始终将发送到另一个应用程序内部流. 在执行复杂分析时，您可以创建多个应用程序内部流来保存中间分析的结果。最后，将应用程序输出配置为将最终分析的结果（来自一个或多个应用程序内部流）保存到外部目标。概括来说，以下是用于编写应用程序代码的典型模式：

- SELECT 语句始终用于 INSERT 语句的上下文中。即，在选择行时，您将结果插入另一个应用程序内部流中。
- INSERT 语句始终用于数据泵的上下文中。即，您使用数据泵对应用程序内部流进行写入。

以下示例应用程序代码读取一个应用程序内部流 (SOURCE_SQL_STREAM_001) 中的记录，并将其写入另一个应用程序内部流 (DESTINATION_SQL_STREAM)。您可以使用数据泵将记录插入应用程序内部流中，如下所示：

```
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

为流名称和列名称指定的标识符需遵循标准 SQL 约定。例如，如果您为标识符加上引号，则可使标识符区分大小写。如果没有这样做，则标识符将默认为大写。有关标识符的更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的标识符。

您的应用程序代码可包含多个 SQL 语句。例如：

- 您可以按顺序编写 SQL 查询，其中一个 SQL 语句的结果将馈送到下一个 SQL 语句。
- 您也可以编写彼此单独运行的 SQL 查询。例如，您可以编写两个 SQL 语句，它们查询同一应用程序内部流，但将输出发送到不同的应用程序内部流。随后，您可以单独查询新创建的应用程序内部流。

您可以创建应用程序内部流来保存中间结果。可使用数据泵将数据插入应用程序内部流。有关更多信息，请参阅应用程序内部流和数据泵 (p. 64)：

如果您添加一个应用程序内部引用表，则可编写 SQL 以联接应用程序内部流和引用表中的数据。有关更多信息，请参阅示例：向 Kinesis Data Analytics 应用程序添加参考数据 (p. 133)：

根据应用程序的输出配置，Amazon Kinesis Data Analytics 根据应用程序的输出配置将数据从特定的应用程序内流写入到外部目的地。确保您的应用程序代码写入输出配置中指定的应用程序内部流。

有关更多信息，请参阅以下主题：

- 流式 SQL 概念 (p. 64)
- Amazon Kinesis Data Analy

配置应用程序输出

在您的应用程序代码中，将 SQL 语句的输出写入一个或多个应用程序内部流。您可以选择将输出配置添加到您的应用程序。将写入以将所有写入到应用程序内部流的 Amazon Kinesis 数据流、Kinesis Data Firehose 传输流或Amazon Lambda函数。

可以用来永久保存应用程序输出的外部目标的数量有限制。有关更多信息，请参阅Limits (p. 189)：

Note

建议用一个外部目标来永久保存应用程序内部错误流数据，以方便调查错误。

在所有这些输出配置中，可以提供以下内容：

- 应用程序内部流名称 - 要永久保存到外部目标的流。

  Kinesis Data Analytics 会查找您在输出配置中指定的应用程序内流。（流名称区分大小写，并且必须完全匹配）。确保应用程序代码创建了这一应用程序内部流。

- 外部目标 — 您可以将数据保存到 Kinesis 数据流、Kinesis Data Firehose 传输流或 Lambda 函数。您需要提供流或函数的 Amazon 资源名称 (ARN)。您还提供可由 Kinesis Data Analytics 代入可由您写入可由。您可以向 Kinesis Data Analytics 描述写入外部目标时要使用的记录格式（JSON、CSV）。

如果 Kinesis Data Analytics 无法写入流媒体或 Lambda 目标，则该服务将继续无限期地尝试。这将产生反向压力，导致您的应用程序滞后。如果不能解决这一问题，您的应用程序最终将停止处理新数据。您可以监控 Kinesis Data Analytics 指标并设置故障警报。有关指标和警报的更多信息，请参阅使用亚马逊 CloudWatch 指标和创建亚马逊 CloudWatch警报。

您可以使用 Amazon Web Services Management Console配置应用程序输出。控制台将调用 API 以保存配置。

使用 Amazon CLI 创建输出

此部分介绍如何为 `CreateApplication` 或 `AddApplicationOutput` 操作创建请求正文的 `Outputs` 部分。

创建 Kinesis 流的 AM

以下 JSON 片段显示了`CreateApplication`请求正`Outputs`文中用于创建 Amazon Kinesis 数据流目标的部分。

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

]
```

创建 Kinesis Data Firehose 传输流的

以下 JSON 片段显示了`CreateApplication`请求正`Outputs`文中用于创建 Amazon Kinesis Data Firehose 传输流目的地的部分。

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

创建 Lambda 函数输出

以下 JSON 代码段显示用于创建 Amazon Lambda 函数目标的 `CreateApplication` 请求正文中的 `Outputs` 部分。

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

使用 Lambda 函数作为输出

通过将 Amazon Lambda 作为目标，您可以更轻松地执行 SQL 结果后处理，然后再将其发送到最终目标。常见的后处理任务包括：

- 将多行聚合为一条记录
- 将当前结果与过去的结果相结合以解决迟到数据的问题
- 根据信息类型传输到不同的目标
- 记录格式转换 (如转换为 Protobuf)
- 字符串操作或转换
- 分析处理后的数据扩充
- 地理空间使用案例的自定义处理
- 数据加密

Lambda 函数可以向各种Amazon服务和其他目的地提供分析信息，包括：

- Amazon Simple Storage Service (Amazon S3)
- 自定义 API
- Amazon DynamoDB
- Apache Aurora
- Amazon Redshift
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- 亚马逊 CloudWatch

有关创建 Lambda 应用程序的更多信息，请参阅入门Amazon Lambda。

主题

Lambda 作为输出权限

要使用 Lambda 作为输出，应用程序的 Lambda 输出 IAM 角色需要以下权限策略：

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

Lambda 作为输出指标

您可以使用 Amazon CloudWatch 监控发送的字节数、成功和失败等。有关 Kinesis Data Analytics 使用 Lambda 作为输出发布的 CloudWatch 指标的信息，请参阅 Amazon Kinesis Analytics 指标。

Lambda 作为输出事件输入数据模型和记录响应模型

要发送 Kinesis Data Analytics 输出记录，您的 Lambda 函数必须符合所需的事件输入数据和记录响应模型。

事件输入数据模型

Kinesis Data Analytics 使用以下请求模型将应用程序的输出记录作为输出函数持续发送到 Lambda。在您的函数中，遍历列表并应用业务逻辑来完成输出要求 (例如，在将数据发送到最终目标之前先进行数据转换)。

字段	描述
invocationId	Lambda 调用 ID（随机 GUID）。
applicationArn	Kinesis Data Analytics 应用程序ARN Amazon 资源名称。
记录	

字段	描述	
recordId	记录 ID (随机 GUID)	
lambdaDeliveryRecordMetadata		

字段	描述	
retryHint	传输重试次数	

data	Base64 编码的输出记录负载	

> **Note**
>
> retryHint 是一个每次传输失败时都会增加的值。该值不会持久不变，并在应用程序中断时重置。

记录响应模型

作为输出函数（带有记录 ID）发送到您的 Lambda 的每条记录都必须使用0k或进行确认DeliveryFailed，并且必须包含以下参数。否则，Kinesis Data Analytics 会将其视为交付失败。

记录

字段	描述
`recordId`	在调用期间，记录 ID 从 Kinesis Data Analytics 传递到 Lambda。如果原始记录的 ID 和确认记录的 ID 之间不匹配，就会被视为传输失败。
`result`	记录的传输状态。以下是可能的值： • `Ok`：记录已成功转换并发送到最终目的地。Kinesis Data Analytics 提取记录进行 SQL 处理。 • `DeliveryFailed`：Lambda 未将记录作为输出函数成功传送到最终目的地。Kinesis Data Analytics 不断重试将交付失败的记录作为输出函数发送到 Lambda。

Lambda 输出调用频率

Kinesis Data Analytics 应用程序会缓冲输出记录并频繁调用Amazon Lambda目标函数。

- 如果在数据分析应用程序中使用滚动窗口将记录发送到目标应用程序内部流，每次触发滚动窗口时，都会调用 Amazon Lambda 目标函数。例如，如果使用 60 秒的滚动窗口将记录发送到目标应用程序内流，则每 60 秒调用一次 Lambda 函数。
- 如果记录作为连续查询或滑动窗口发送到应用程序内的目标应用程序流，则 Lambda 目标函数大约每秒调用一次。

> Note
>
> 每个 Lambda 函数调用请求的有效负载大小限制适用。超过这些限制会导致输出记录被拆分并在多个 Lambda 函数调用之间发送。

添加一个 Lambda 函数以用作输出

以下过程演示如何添加 Lambda 函数作为 Kinesis Data Analytics 应用程序的输出。

1. 登录Amazon Web Services Management Console并打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择列表中的应用程序，然后选择 Application details。
3. 在 Destination 部分，选择 Connect new destination。
4. 对于 Destination 项，选择 Amazon Lambda function。
5. 在将记录传送到Amazon Lambda部分中，选择现有的 Lambda 函数和版本，或者选择新建。
6. 如果您要创建新的 Lambda 函数，请记下：
   a. 选择提供的模板之一。有关更多信息，请参阅 为应用程序目标创建 Lambda 函数 (p. 34)。
   b. 将在新的浏览器选项卡中打开 Create Function (创建函数) 页。在 Name (名称) 框中，为函数指定一个有意义的名称（例如，**`myLambdaFunction`**）。
   c. 针对您的应用程序用后处理功能更新模板。有关创建 Lambda 函数的信息，请参阅Amazon Lambda开发人员指南中的入门。
   d. 在 Kinesis Data Analytics 控制台的 Lambda 函数列表中，选择您刚刚创建的 Lambda 函数。选择 $LATES T 获取 Lambda 函数版本。

7. 在 In-application stream 部分，选择 Choose an existing in-application stream。对于 In-application stream name，选择应用程序的输出流。所选输出流的结果将发送到 Lambda 输出函数。
8. 保持表单其余部分为默认值，然后选择 Save and continue。

现在，您的应用程序将记录从应用程序内流发送到您的 Lambda 函数。您可以在亚马逊 CloudWatch 控制台中查看默认模板的结果。监控`AWS/KinesisAnalytics/LambdaDelivery.OkRecords`指标以查看传输到 Lambda 函数的记录数量。

常见的 Lambda 作为输出故障

以下是向 Lambda 函数交付失败的常见原因。

- 并非批处理中发送到 Lambda 函数的所有记录（带有记录 ID）都会返回到 Kinesis Data Analytics 服务。
- 响应中缺少记录 ID 或状态字段。
- Lambda 函数超时不足以完成 Lambda 函数中的业务逻辑。
- Lambda 函数中的业务逻辑不会捕获所有错误，导致因未处理的异常而产生超时和反向压力。这些消息通常称为“毒丸”消息。

对于数据传输失败，Kinesis Data Analytics 会继续对同一组记录重试 Lambda 调用，直到成功为止。要深入了解故障，您可以监控以下 CloudWatch 指标：

- Kinesis Data Analytics 应用程序 Lambda as 输出 CloudWatch 指标：指明成功和失败的数量以及其他统计数据。有关更多信息，请参阅 Amazon Kinesis Analytics 指标。
- Amazon Lambda函数 CloudWatch 指标和日志。

为应用程序目标创建 Lambda 函数

您的 Kinesis Data Analytics 应用程序可以使用Amazon Lambda函数作为输出。Kinesis Data Analytics 提供了用于创建 Lambda 函数的模板，这些函数用作应用程序的目的地。可以将这些模板作为应用程序输出后处理的起点。

主题

在 Node.js 中创建 Lambda 函数目标

控制台上提供了以下用于在 Node.js 中创建目标 Lambda 函数的模板：

Lambda 作为输出蓝图	语言和版本	描述
`kinesis-analytics-output`	Node.js 12.x	将输出记录从 Kinesis Data Analytics 应用程序传输到自定义目的地。

在 Python 中创建 Lambda 函数目标

控制台上提供了以下用于在 Python 中创建目标 Lambda 函数的模板：

Lambda 作为输出蓝图	语言和版本	描述
kinesis-analytics-output-sns	Python 2.7	将输出记录从 Kinesis Data Analytics 应用程序传递到Amazon SNS。
kinesis-analytics-output-ddb	Python 2.7	将输出记录从 Kinesis Data Analytics 应用程序传递到Amazon DynamoDB。

在 Java 中创建 Lambda 函数目标

要在 Java 中创建目标 Lambda 函数，请使用 Java 事件类。

以下代码演示了使用 Java 的目标 Lambda 函数示例：

```
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;
    }

}
```

在.NET 中创建 Lambda 函数目标

要在.NET 中创建目标 Lambda 函数，请使用 .NET 事件类。

以下代码演示了使用 C# 的目标 Lambda 函数示例：

```
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;
        }
    }
```

有关在.NET 中创建用于预处理的 Lambda 函数和目标的更多信息，请参阅Amazon.Lambda.KinesisAnalyticsEvents。

将应用程序输出永久保存到外部目标的传输模型

Amazon Kinesis Data Analytics 使用 “至少一次” 交付模型将应用程序输出到配置的目的地。当应用程序运行时，Kinesis Data Analytics 会使用内部检查点。这些检查点是指将输出记录传输到目标并且未丢失数据的时间点。该服务根据需要使用检查点以确保至少向配置的目标传输一次应用程序输出。

在正常情况下，您的应用程序会持续处理传入的数据。Kinesis Data Analytics 将输出写入配置的目标位置，例如 Kinesis 数据流或 Kinesis Data Firehose 传输流。不过，您的应用程序偶尔可能会中断，例如：

- 您选择停止应用程序并稍后重新启动。
- 您删除了 Kinesis Data Analytics 将应用程序输出写入配置目标所需的 IAM 角色。如果没有 IAM 角色，Kinesis Data Analytics 就没有任何权限代表你写入外部目标。
- 网络中断或其他内部服务故障导致应用程序暂时停止运行。

当您的应用程序重新启动时，Kinesis Data Analytics 可确保它继续处理和写入故障发生之前或等于故障发生时点的输出。这有助于确保它将所有应用程序输出传输到配置的目标，而不会遗漏任何内容。

假设您从同一应用程序内部流中配置多个目标。应用程序从故障中恢复后，Kinesis Data Analytics 将从传送到最慢目的地的最后一条记录恢复到配置目标的持久输出。这可能会导致相同的输出记录多次传送到其他目的地。在这种情况下，您必须在外部处理目标中的潜在重复项。

错误处理

Amazon Kinesis Data Analytics 会直接向您返回 API 或 SQL 错误。有关 API 操作的更多信息，请参阅操作 (p. 199)。有关处理 SQL 错误的更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考。

Amazon Kinesis Data Analytics 使用名为的应用程序内错误流报告运行时错误`error_stream`。

使用应用程序内部错误流报告错误

Amazon Kinesis Data Analytics 向名为的应用程序内错误流报告运行时错误`error_stream`。下面是可能出现的错误的示例：

- 从流式传输源读取的一个记录不符合输入架构。
- 您的应用程序代码指定被零除。
- 行顺序错误（例如，流中出现的一个记录具有 ROWTIME 值，一个用户修改该值后导致记录顺序错误）。
- 源流中的数据无法转换为架构中指定的数据类型 (强制转换错误)。有关可转换的数据类型的信息，请参阅将 JSON 数据类型映射到 SQL 数据类型 (p. 13)。

建议以编程方式在 SQL 代码中处理这些错误或将错误流上的数据永久保存到外部目标。这需要将输出配置添加到您的应用程序 (请参阅配置应用程序输出 (p. 29))。要查看应用程序内错误流工作方式的示例，请参阅示例：探索应用程序内部错误流 (p. 157)。

> Note
>
> 您的 Kinesis 数据分析应用程序无法以编程方式访问或修改错误流，因为错误流是使用系统帐户创建的。必须使用错误输出来确定应用程序可能遇到的错误。然后，编写应用程序的 SQL 代码来处理预期的错误情况。

错误流架构

错误流具有以下架构：

字段	数据类型	备注
ERROR_TIME	TIMESTAMP	错误出现的时间
ERROR_LEVEL	VARCHAR(10)	
ERROR_NAME	VARCHAR(32)	
MESSAGE	VARCHAR(4096)	
DATA_ROWTIME	TIMESTAMP	传入记录的行时间
DATA_ROW	VARCHAR(49152)	原始行中的十六进制编码数据. 您可以使用标准库对此值进行十六进制解码，也可以使用 Web 资源，如 Hex to String Converter。
PUMP_NAME	VARCHAR(128)	利用 CREATE PUMP 定义的原始泵

自动扩展应用程序以提高吞吐量

Amazon Kinesis Data Analytics 可弹性扩展您的应用程序，以适应源流的数据吞吐量和大多数场景下的查询复杂性。Kinesis Data Analytics 以 Kinesis 处理单元 (KPU) 的形式提供容量。一个 KPU 可为您提供内存 (4 GB) 和相应的计算和联网能力。

应用程序的默认 KPU 限制为 32 个。有关如何申请提高此限额的说明，请参阅请求提高亚马逊服务限额。

使用标记

本节介绍如何向 Kinesis Data Analytics 应用程序添加键值元数据标签。这些标签可用于以下目的：

- 确定各个 Kinesis Data Analytics 应用程序的账单。有关更多信息，请参阅 Billing and Cost Man age Amazonment 指南中的使用成本分配标签。
- 根据标签控制对应用程序资源的访问。有关更多信息，请参阅用户指南中的使用标签控制访问。
- 用户定义的目的。您可以根据用户标签定义应用程序的功能。

请注意与标记相关的以下信息：

- 应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。
- 如果某项操作包含的标签列表存在重复的 Key 值，服务将提示 InvalidArgumentException。

本主题包含下列部分：

创建应用程序时添加标签

在使用CreateApplication操作tags参数创建应用程序时，您可以添加标签。

以下示例请求显示了 CreateApplication 请求的 Tags 节点：

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

为现有应用程序添加或更新标签

您可以使用TagResource操作向应用程序添加标签。您无法使用UpdateApplication操作向应用程序添加标签。

要更新现有标签，可添加一个与现有标签的键相同的标签。

针对 TagResource 操作的以下示例请求可添加新标签或更新现有标签：

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

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

列出应用程序的标签

要列出现有标签，请使用ListTagsForResource操作。

针对 ListTagsForResource 操作的以下示例请求可列出应用程序的标签：

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

从应用程序删除标签

要从应用程序中删除标签，请使用UntagResource操作。

以下UntagResource操作请求示例从应用程序中删除标签：

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

Amazon Kinesis Data Analytics 入门

接下来，您可以找到一些主题来帮助您开始使用适用于 SQL 应用程序的 Amazon Kinesis Data Analytics。如果您是 Kinesis Data Analytics for SQL 应用程序的新手，我们建议您在执行 “入门” 部分中的步骤适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)之前，先阅读中介绍的概念和术语。

主题

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户后，启用多重身份验证 (MFA) 来保护您的管理员用户。有关说明，请参阅 IAM 用户指南中的为 IAM 用户启用虚拟 MFA 设备（控制台）。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA，并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户中创建 IAM 用户
- 适用于 Amazon 资源的访问管理

- 基于 IAM 身份的策略示例

步骤 1：设置 账户并创建管理员用户

首次使用 Amazon Kinesis Data Analytics 前，请完成以下任务：

1. 注册 Amazon (p. 41)
2. 创建 IAM 用户 (p. 41)

注册 Amazon

当您注册 Amazon Web Services 时，您Amazon Web Services 账户会自动注册中的Amazon所有服务，包括 Amazon Kinesis Data Analytics。您只需为使用的服务付费。

借助 Kinesis Data Analytics，您仅需为实际使用的资源付费。如果您是新Amazon客户，还可以免费试用 Kinesis Data Analytics。有关更多信息，请参阅 Amazon 免费使用套餐。

如果您已有Amazon Web Services 账户，请跳到下一个任务。如果您还没有Amazon Web Services 账户，请执行以下步骤创建。

创建 Amazon Web Services 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

请记下您的Amazon Web Services 账户 ID，因为在下一个任务中您会用到它。

创建 IAM 用户

中的Amazon服务 (例如 Amazon Kinesis Data Analytics) 要求您在访问时提供凭证，以便服务可以确定您是否有权限访问服务拥有的资源。控制台要求您的密码。您可以为您的 Amazon Web Services 账户 创建访问密钥以访问 Amazon CLI 或 API。但是，我们不建议使用您的 Amazon Web Services 账户 的凭证访问 Amazon。相反，我们建议您使用 Amazon Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。您随后便可以使用一个特殊的 URL 和该 IAM 用户的凭证访问 Amazon。

如果您已注册但尚未为Amazon自己创建一个 IAM 用户，则可以使用 IAM 控制台自行创建。

本指南中的入门练习假定您拥有具有管理权限的用户 (`adminuser`)。请按照以下过程在您的账户中创建 `adminuser`。

创建管理员用户和登录控制台

1. 创建一个名为您的`adminuser`的管理员用户Amazon Web Services 账户。有关说明，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。
2. 用户可使用特殊 URL 登录 Amazon Web Services Management Console。有关更多信息，请参阅《IAM 用户指南》中的用户如何登录您的账户。

有关 IAM 的更多信息，请参阅以下文档：

- Amazon Identity and Access Management (IAM)
- 入门
- IAM 用户指南

下一个步骤

步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 42)

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户后，启用多重身份验证 (MFA) 来保护您的管理员用户。有关说明，请参阅 IAM 用户指南中的为 IAM 用户启用虚拟 MFA 设备（控制台）。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA，并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- 基于 IAM 身份的策略示例

步骤 2：设置 Amazon Command Line Interface (Amazon CLI)

按照以下步骤下载和配置 Amazon Command Line Interface (Amazon CLI)。

Important

您不需要使用 Amazon CLI 以执行入门练习中的步骤。但是，本指南中的某些练习会用到 Amazon CLI。您可以跳过此步骤转至 步骤 3：创建 Amazon Kinesis Data Analytics 应用程序 (p. 43)，并在稍后需要时设置 Amazon CLI。

设置 Amazon CLI

1. 下载并配置 Amazon CLI。有关说明，请参阅《Amazon Command Line Interface 用户指南》中的以下主题：
   - 开始设置 Amazon Command Line Interface
   - 配置 Amazon Command Line Interface
2. 在 Amazon CLI 配置文件中为管理员用户添加一个命名配置文件。在执行 Amazon CLI 命令时，您将使用此配置文件。有关命名配置文件的更多信息，请参阅Amazon Command Line Interface用户指南中的命名配置文件。

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

   有关可用Amazon Web Services 区域列表，请参阅《Amazon Web Services 一般参考》中的区域和终端节点。
3. 在命令提示符处输入以下帮助命令来验证设置：

   ```
   aws help
   ```

下一个步骤

步骤 3：创建 Amazon Kinesis Data Analytics 应用程序 (p. 43)

步骤 3：创建 Amazon Kinesis Data Analytics 应用程序

按照本节中的步骤操作，您可以使用控制台创建您的第一个 Kinesis Data Analytics 应用程序。

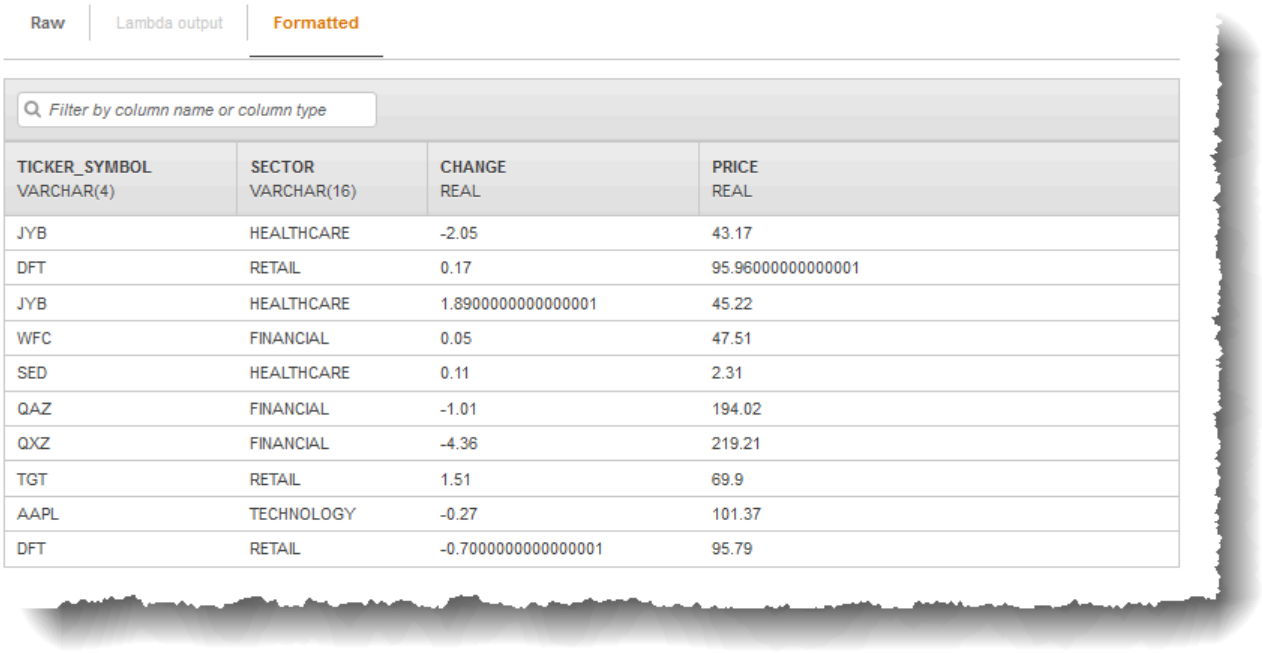
Note

在尝试入门练习之前，我们建议您先查看 适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)。

在本入门练习中，您可以通过控制台来使用演示流或包含应用程序代码的模板。

- 如果您选择使用演示流，则控制台会在您的账户中创建名为的 Kinesis 数据流`kinesis-analytics-demo-stream`。

  Kinesis 数据分析应用程序需要流媒体源。对于此源，本指南中的几个 SQL 示例使用演示流 `kinesis-analytics-demo-stream`。控制台还会运行持续向此流添加示例数据 (模拟的股票交易记录) 的脚本，如下所示。


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

在本练习中，您可以使用 kinesis-analytics-demo-stream 作为应用程序的流式传输源。

Note

演示流会保留在账户中。您可以使用它测试本指南中的其他示例。但是，如果您退出控制台，控制台使用的脚本会停止填充数据。控制台可在需要时提供重新开始填充流的选项。

- 如果您选择使用包含示例应用程序代码的模板，请使用控制台提供的模板代码对演示流执行简单分析。

您可以使用如下这些功能快速设置您的首个应用程序：

1. 创建应用程序-您只需要提供一个名称。控制台创建应用程序，服务将应用程序状态设置为 READY。

2. 配置输入-首先，添加一个流媒体源，即演示流。您必须先在控制台中创建演示流才能使用它。然后，控制台对演示流中的记录进行随机采样，并推断创建的应用程序内部输入流的架构。控制台将应用程序内部流命名为 SOURCE_SQL_STREAM_001。

   控制台使用发现 API 来推断架构。如果需要，可以编辑推断的架构。有关更多信息，请参阅DiscoverInputSchema (p. 233)：Kinesis Data Analytics 使用此架构创建应用程序内流。

   当您启动应用程序时，Kinesis Data Analytics 会代表您持续读取演示流，并在SOURCE_SQL_STREAM_001应用程序内输入流中插入行。

3. 指定应用程序代码 - 您可以使用提供以下代码的模板（称为连续式筛选器）：

```
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%';
```

应用程序代码将查询应用程序内部流 SOURCE_SQL_STREAM_001。之后，该代码将使用泵将生成的行插入到另一个应用程序内部流 DESTINATION_SQL_STREAM。有关此编码模式的更多信息，请参阅应用程序代码 (p. 28)。

有关 Kinesis Data Analytics 支持的 SQL 语言元素的信息，请参阅 Amazon Kinesis Data Analytics SQL 参考资料。

4. 配置输出-在本练习中，您不配置任何输出。也就是说，您不会将应用程序创建的应用程序内部流中的数据永久保存到任何外部目标，而应在控制台中验证查询结果。本指南中的其他示例演示了如何配置输出。要查看其中一个示例，请参阅示例：创建简单警报 (p. 155)。

   Important

   演习使用美国东部（弗吉尼亚北部）区域 (us-east-1) 来设置应用程序。您可以使用任何受支持的 Amazon Web Services 区域。

下一个步骤

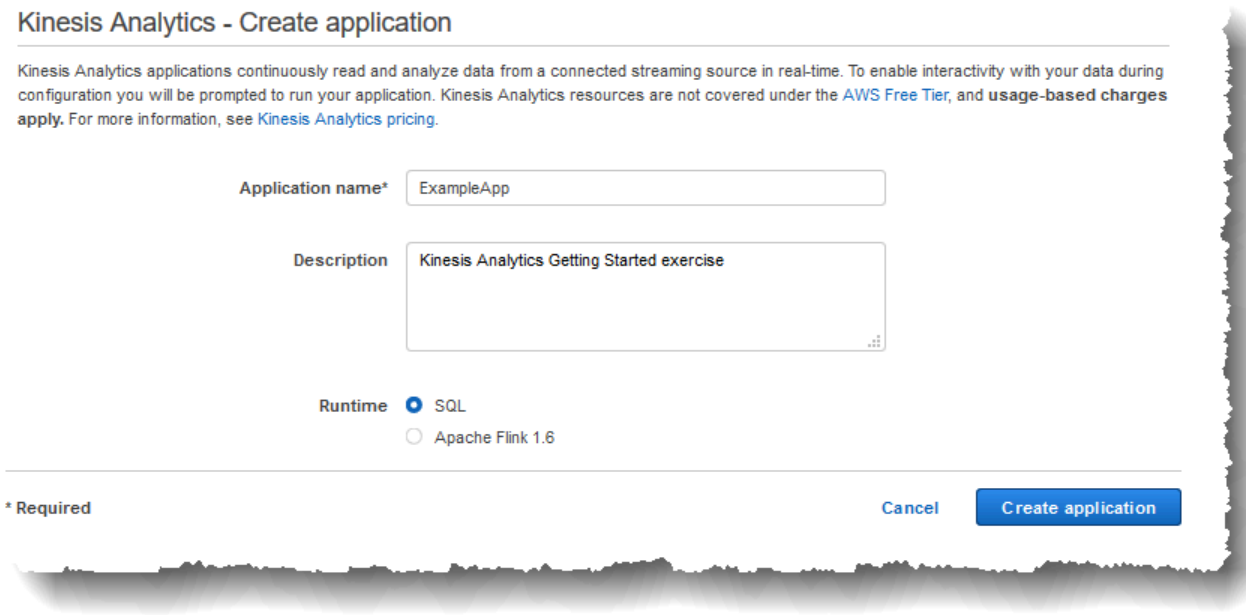
步骤 3.1：创建应用程序 (p. 45)

步骤 3.1：创建应用程序

在本节中，创建 Amazon Kinesis Data Analytics 应用程序。您将在下一步骤中配置应用程序输入。

创建数据分析应用程序

1. 登录Amazon Web Services Management Console并打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application（创建应用程序）。
3. 在创建应用程序页面上，键入应用程序名称，键入描述，为应用程序的运行时设置选择 SQL，然后选择创建应用程序。



这样做会创建一个状态为 READY 的 Kinesis 数据分析应用程序。控制台显示您可在其中配置输入和输出的应用程序中心。

> **Note**
>
> 要创建应用程序，CreateApplication (p. 214) 操作只需要应用程序名称。您可以在控制台中创建应用程序后添加输入和输出配置。

您可以在下一步中为应用程序配置输入。在输入配置中，您将为应用程序添加一个流式数据源，并通过流式传输源数据采样来发现应用程序内部输入流的架构。

下一个步骤

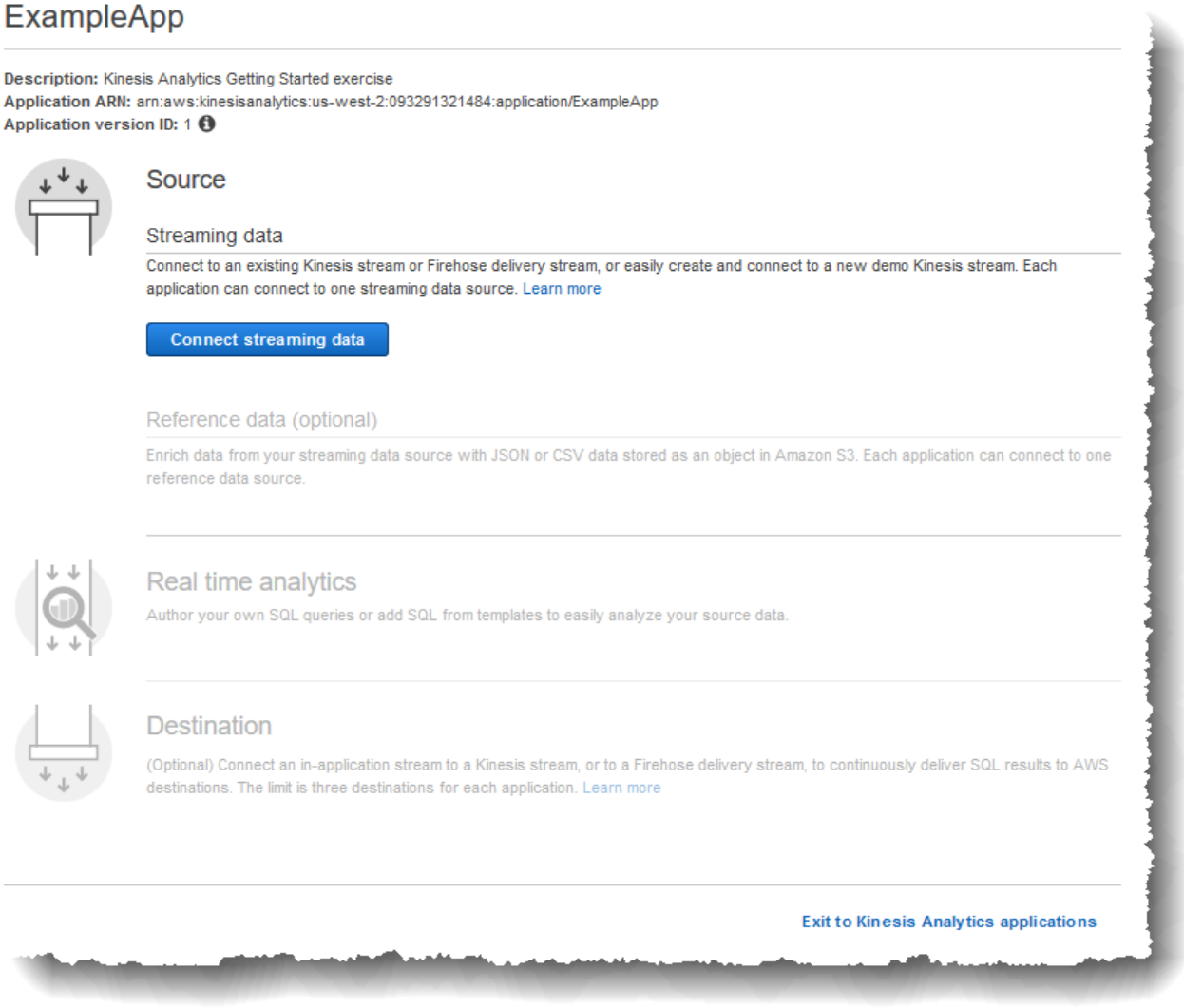
步骤 3.2：配置输入 (p. 46)

步骤 3.2：配置输入

您的应用程序需要流式传输源。控制台创建了演示流（名为 `kinesis-analytics-demo-stream`）以帮助您入门。控制台还会运行在流中填充记录的脚本。

为应用程序添加流式传输源

1. 在控制台中的应用程序中心页面上，选择 Connect streaming data (连接流数据)。



2. 在显示的页面上查看以下内容：

   - Source 部分，您可在其中指定应用程序的流式传输源。您可以选择现有流式传输源或创建流式传输源。在本练习中，您将新建一个流，即演示流。

     默认情况下，控制台将创建的应用程序内部输入流命名为 INPUT_SQL_STREAM_001。在本练习中，请按原样保留该名称。

   - 流引用名称-此选项显示创建的应用程序内部输入流的名称SOURCE_SQL_STREAM_001。您可以更改此名称，但本练习将保留此名称。

     在输入配置中，您可以将演示流映射到创建的应用程序内部输入流。当您启动应用程序时，Amazon Kinesis Data Analytics 会持续读取演示流并在应用程序内输入流中插入行。您可以在应用程序代码中查询此应用程序内部输入流。

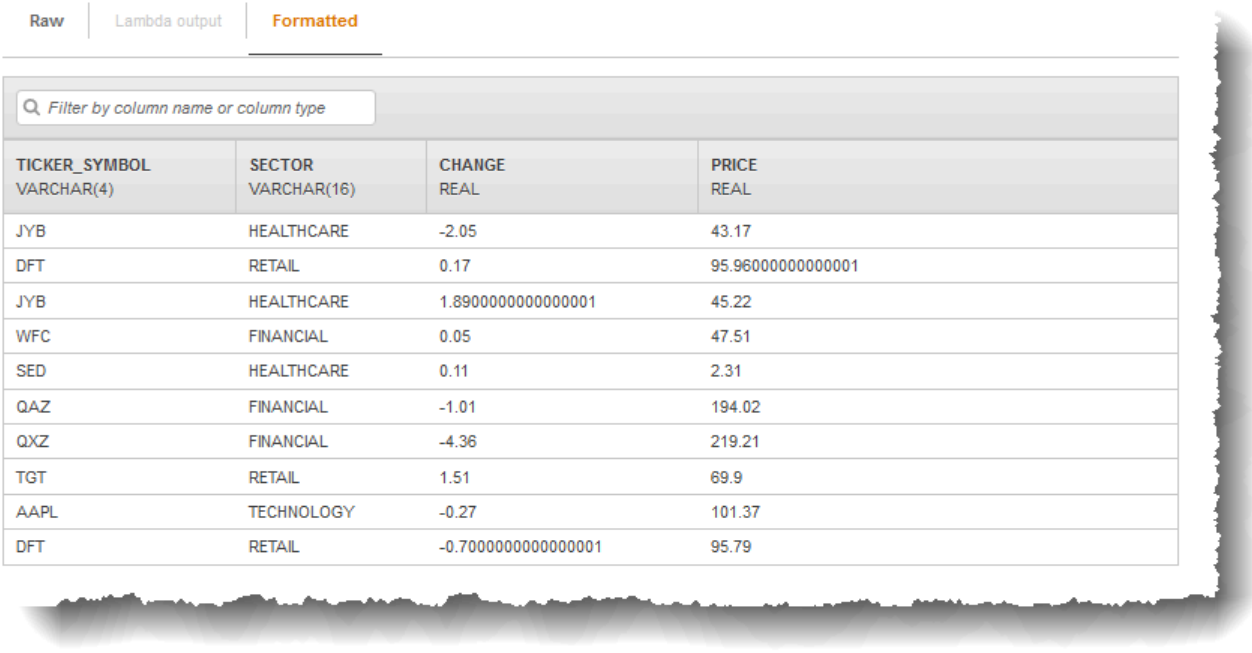
- 使用以下@@ 方法进行记录预处理Amazon Lambda：在此选项中，您可以指定一个Amazon Lambda表达式，该表达式在应用程序代码执行之前修改输入流中的记录。在此练习中，选中 Disabled 选项。有关 Lambda 预处理的更多信息，请参阅使用 Lambda 函数预处理数据 (p. 19)。

在本页上提供所有信息后，控制台会发送更新请求（请参阅UpdateApplication (p. 249)）以便将输入配置添加到应用程序。

3. 在 Source 页面上选择 Configure a new stream、
4. 选择 Create demo stream。控制台通过执行以下操作来配置应用程序输入：

   - 控制台会创建名为 Kinesis Data Streamskinesis-analytics-demo-stream。
   - 控制台将使用示例股票行情机数据填充流。
   - 利用 DiscoverInputSchema (p. 233) 输入操作，控制台将通过读取流上的示例记录来推断架构。推断出的架构是适用于创建的应用程序内部输入流的架构。有关更多信息，请参阅配置应用程序输入 (p. 4)：
   - 控制台将显示推断的架构和从流式传输源读取的用来推断架构的示例数据。

   控制台显示流式传输源中的示例记录。


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

以下内容将显示在 Stream sample 控制台页面上：

- Raw stream sample 选项卡显示 DiscoverInputSchema (p. 233) API 操作为推断架构而采用的抽样原始流记录。
- Formatted stream sample 选项卡显示 Raw stream sample 选项卡中的数据的表格式版本。
- 如果您选择 Edit schema，则可以编辑推断的架构。在本练习中，请不要更改推断的架构。有关编辑架构的更多信息，请参阅使用架构编辑器 (p. 54)。

  如果您选择 Rediscover schema，则可以请求控制台再次运行 DiscoverInputSchema (p. 233) 并推断架构。

5. 选择 Save and continue。

   现在，您已具有一个已添加输入配置的应用程序。在下一步中，您将添加 SQL 代码以便对应用程序内部输入流中的数据执行一些分析。

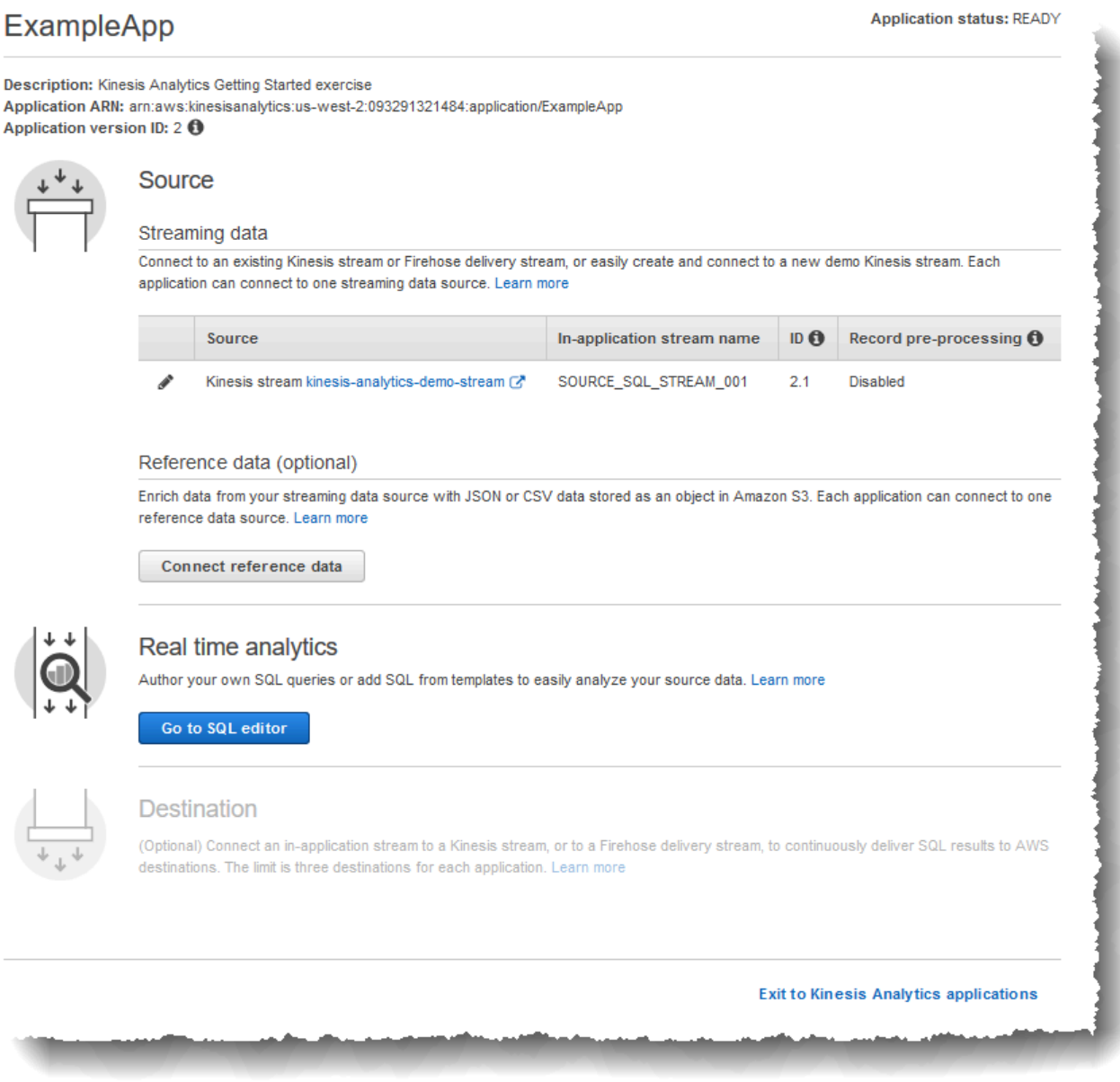
下一个步骤

步骤 3.3：添加实时分析（添加应用程序代码）(p. 49)

步骤 3.3：添加实时分析（添加应用程序代码）

您可以针对应用程序内部流编写您自己的 SQL 查询，但在下一步中，您将使用一个提供示例代码的模板。

1. 在应用程序中心页面上，选择 Go to SQL editor。



2. 在 “你想开始运行”ExampleApp “吗？ 对话框中，选择是，启动应用程序。

   控制台会发送启动应用程序（请参阅StartApplication (p. 241)）的请求，然后会显示 SQL 编辑器页面。

3. 控制台将打开 SQL 编辑器页面。查看该页面，其中包含多个按钮（Add SQL from templates、Save and run SQL）和不同选项卡。
4. 在 SQL 编辑器中，选择 Add SQL from templates。
5. 从可用模板列表中，选择 Continuous filter。示例代码读取来自一个应用程序内部流的数据 (WHERE 子句将筛选行)，并将数据插入到另一个应用程序内部流，如下所示：

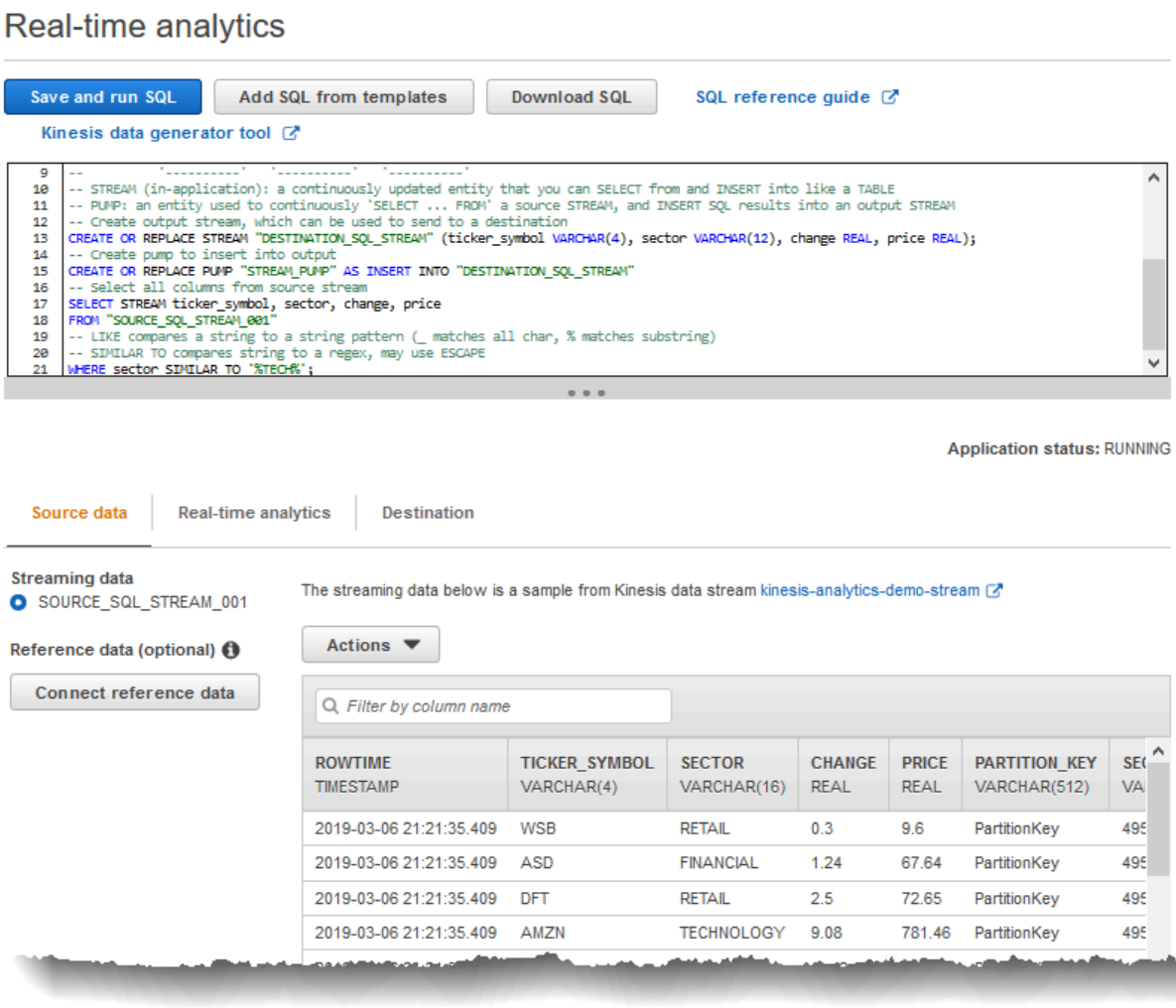
   - 它将创建应用程序内部流 DESTINATION_SQL_STREAM。
   - 它将创建泵 STREAM_PUMP，并使用此泵从 SOURCE_SQL_STREAM_001 中选择行，并将这些行插入到 DESTINATION_SQL_STREAM。

6. 选择 Add this SQL to editor。
7. 按如下方式测试应用程序代码：

   请记住，您已启动应用程序（状态为 RUNNING）。因此，Amazon Kinesis Data Analytics 已经在持续从流媒体源读取数据，并将行添加到应用程序内流中SOURCE_SQL_STREAM_001。

   a. 在 SQL 编辑器中，选择 Save and run SQL。控制台首先会发送保存应用程序代码的更新请求。然后，代码会持续执行。

   b. 您可以在 Real-time analytics 选项卡中查看结果。



SQL 编辑器包含以下选项卡：

- Source data 选项卡显示映射到流式传输源的应用程序内部输入流。选择应用程序内部流，您可以看到数据不断传入。请注意应用程序内部输入流中未在输入配置中指定的其他列。其中包括以下时间戳列：

- ROWTIM E — 应用程序内流中的每一行都有一个名为的特殊列ROWTIME。此列是 Amazon Kinesis Data Analytics 在第一个应用程序内流（映射到流源的应用程序内输入流）中插入该行时的时间戳。

- Ap@@ rocisate_Arrival_Tim e — 每条 Kinesis Data Analytics 记录都包含一个名为的值Approximate_Arrival_Time。此值是流式源成功接收和存储该记录时设置的大致到达时间戳。当 Kinesis Data Analytics 从流媒体源读取记录时，它会将此列提取到应用程序内的输入流中。

这些时间戳值在基于时间的窗口式查询中非常有用。有关更多信息，请参阅窗口式查询 (p. 67)：

- Real-time analytics 选项卡显示应用程序代码创建的所有其他应用程序内部流。它还包括错误流。Kinesis Data Analytics 会将其无法处理的任何行发送到错误流。有关更多信息，请参阅错误处理 (p. 36)：

选择 DESTINATION_SQL_STREAM 可查看应用程序代码插入的行。请注意您的应用程序代码未创建的其他列。这些列包括ROWTIME时间戳列。Kinesis Data Analytics 只是从源 (SOURCE_SQL_STREAM_001) 复制这些值。

- 目的地选项卡显示了 Kinesis Data Analytics 写入查询结果的外部目的地。您尚未为应用程序输出配置任何外部目标。

下一个步骤

步骤 3.4：(可选) 更新应用程序代码 (p. 52)

步骤 3.4：(可选) 更新应用程序代码

在此步骤中，您将研究如何更新应用程序代码。

更新应用程序代码

1. 按如下方式创建另一个应用程序内部流：

   - 创建名为 DESTINATION_SQL_STREAM_2 的另一个应用程序内部流。
   - 创建数据泵，然后从 DESTINATION_SQL_STREAM 中选择行，以便使用数据泵在新创建的流中插入这些行。

   在 SQL 编辑器中，将以下代码附加到现有应用程序代码中：

```
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";
```

保存并运行代码。Real-time analytics 选项卡上将显示其他应用程序内部流。

2. 创建两个应用程序内部流。根据股票代码筛选 SOURCE_SQL_STREAM_001 中的行，然后将这些行插入到这些单独的流中。

   将以下 SQL 语句附加到您的应用程序代码：

```
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%';
```

   保存并运行代码。请注意 Real-time analytics 选项卡上的其他应用程序内部流。

现在，您已有 Amazon Kinesis Data Analytics 应用程序。在本练习中，您已完成以下操作：

- 创建了您的第一个 Kinesis 数据分析应用程序。
- 配置了应用程序输入，将演示流标识为流媒体源并将其映射到已创建的应用程序内流 (SOURCE_SQL_STREAM_001)。Kinesis Data Analytics 持续读取演示流并在应用程序内流中插入记录。
- 应用程序代码已查询 SOURCE_SQL_STREAM_001，并将输出写入到名为 DESTINATION_SQL_STREAM 的另一个应用程序内部流。

现在，您可以选择性地配置应用程序输出，以便将应用程序输出写入到外部目标。也就是说，您可配置应用程序输出以便将 DESTINATION_SQL_STREAM 中的记录写入到外部目标。在本练习中，此步骤为可选步骤。要了解如何配置目标，请转到下一步。

下一个步骤

步骤 4：(可选) 使用控制台编辑架构和 SQL 代码 (p. 53).

步骤 4：(可选) 使用控制台编辑架构和 SQL 代码

接下来，您将了解有关如何编辑推断的架构以及如何为 Amazon Kinesis Data Analytics 编辑SQL 代码的信息。您可以使用作为 Kinesis Data Analytics 控制台一部分的架构编辑器和 SQL 编辑器来实现此目的。

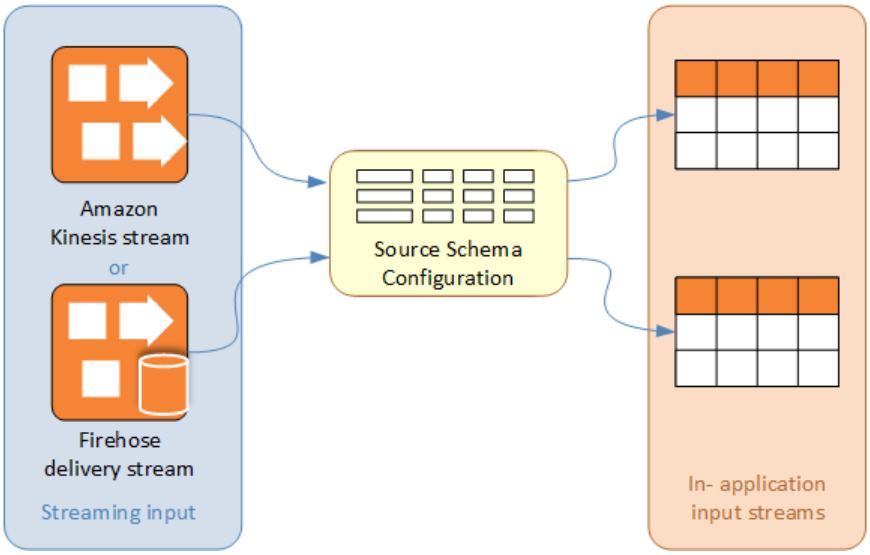
Note

要在控制台中访问示例数据，您的登录用户角色必须具有 `kinesisanalytics:GetApplicationState` 权限。有关 Kinesis Data Analytics 应用程序权限的更多信息，请参阅访问管理概述 (p. 166)。

主题

使用架构编辑器

AAmazon Kinesis Data Analytics 应用程序输入流的架构定义了如何将流中的数据提供给应用程序中的 SQL 查询。



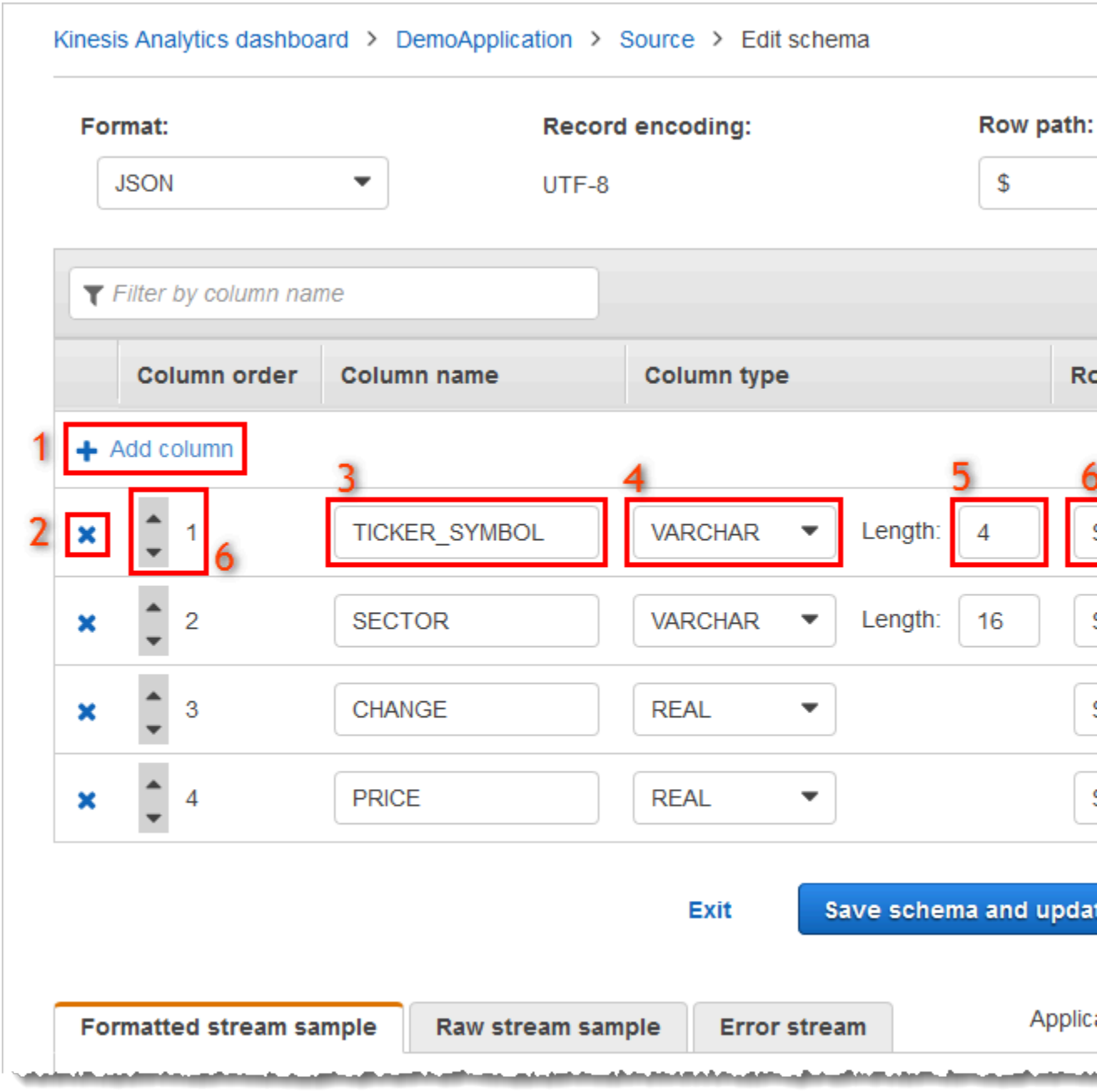
架构包含选择条件，用于确定哪部分流输入将转换为应用程序内部输入流中的数据列。此输入可以是以下项之一：

- JSON 输入流的 JSONPath 表达式。JSONPath 是用于查询 JSON 数据的工具。
- 输入流的列编号 (逗号分隔值 (CSV) 格式)。
- 列名称和用于在应用程序内数据流中呈现数据的 SQL 数据类型。该数据类型还包含字符或二进制数据的长度。

控制台将尝试使用 DiscoverInputSchema (p. 233) 生成架构。如果架构发现失败或返回了不正确或不完整的架构，您必须使用架构编辑器手动编辑架构。

架构编辑器主屏幕

以下屏幕截图显示了架构编辑器的主屏幕。



您可以将下列编辑应用于架构：

- 添加列 (1)：如果未自动检测到数据项，您可能需要添加数据列。
- 删除列 (2)：如果您的应用程序不需要源流中的数据，您可以排除该数据。此排除不会影响源流中的数据。排除数据后，数据将不可供应用程序使用。
- 重命名列 (3)。列名称不能为空，长度必须超过一个字符，并且不得包含保留的 SQL 关键字。该名称还必须符合 SQL 普通标识符的命名标准：名称必须以字母开头，并且只包含字母、下划线字符和数字。

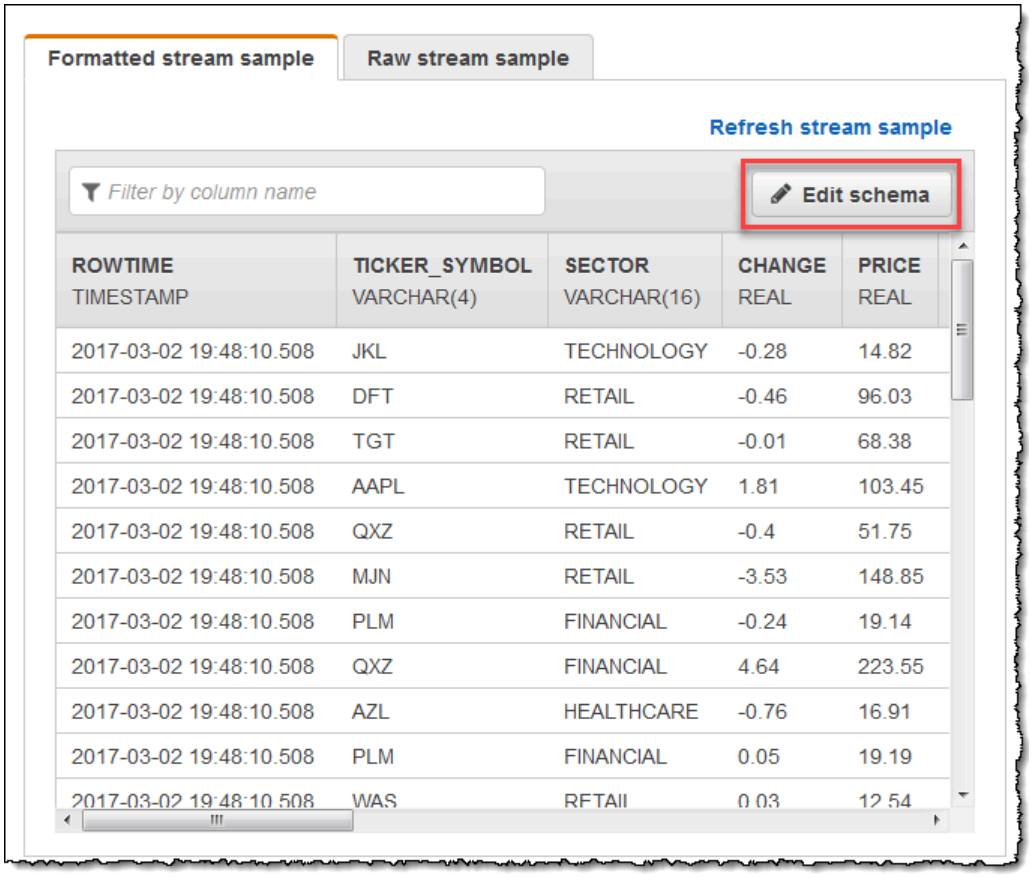
- 更改列的数据类型 (4) 或长度 (5)：您可以为列指定兼容的数据类型。如果您指定了不兼容的数据类型，则将使用 NULL 填充列或完全不填充应用程序内部流。在后一种情况下，错误将写入到错误流。如果您为列指定的长度太小，传入数据将被截断。
- 更改列的选择条件 (6)：您可以编辑用于确定列中数据源的 JSONPath 表达式或 CSV 列顺序。要更改 JSON 架构的选择条件，请为行路径表达式输入新值。CSV 架构将使用列顺序作为选择条件。要更改 CSV 架构的选择条件，请更改列的顺序。

编辑流式传输源的架构

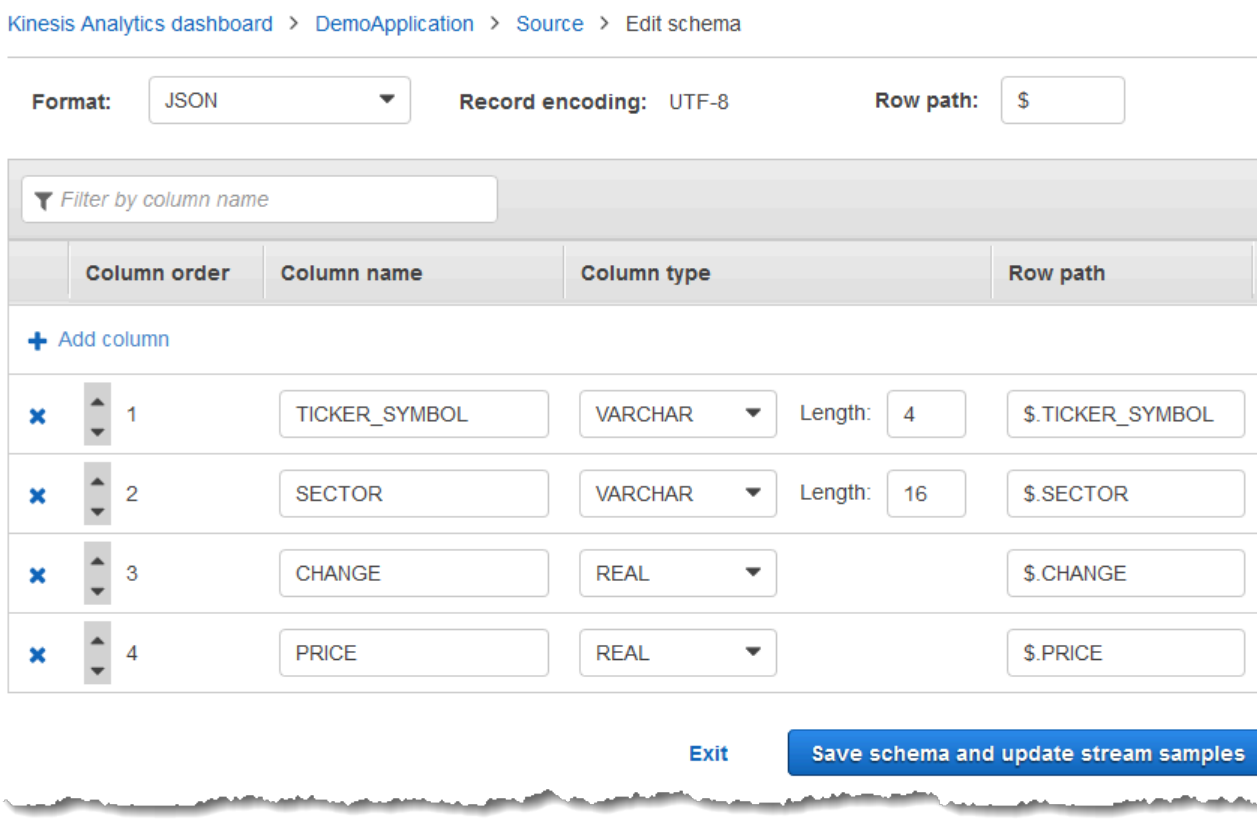
如果您需要编辑流式传输源的架构，请按照以下步骤操作。

编辑流式传输源的架构

1. 在 Source 页上，选择 Edit schema。



2. 在 Edit schema 页上，编辑源架构。



3. 对于 Format，选择 JSON 或 CSV。对于 JSON 或 CSV 格式，支持的编码为 ISO 8859-1。

有关编辑 JSON 或 CSV 格式的架构的更多信息，请参阅后续章节中的过程。

编辑 JSON 架构

您可以通过以下步骤编辑 JSON 架构。

编辑 JSON 架构

1. 在架构编辑器，选择 Add column 以添加列。

   新列将显示在第一个列位置。要更改列顺序，请选择列名称旁边的向上和向下箭头。

   对于新列，请提供以下信息：

   - 对于 Column name，键入一个名称。

     列名称不能为空，长度必须超过一个字符，并且不得包含保留的 SQL 关键字。它还必须符合 SQL 普通标识符的命名标准：必须以字母开头，并且只包含字母、下划线字符和数字。

   - 对于 Column type，键入一个 SQL 数据类型。

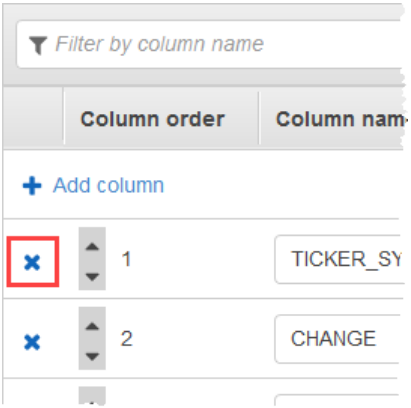
     列类型可以是任何受支持的 SQL 数据类型。如果新数据类型为 CHAR、VARBINARY 或 VARCHAR，请在 Length (长度) 中指定数据长度。有关更多信息，请参阅数据类型。

   - 对于 Row path，提供一个行路径。行路径是映射到 JSON 元素的有效 JSONPath 表达式。

Note

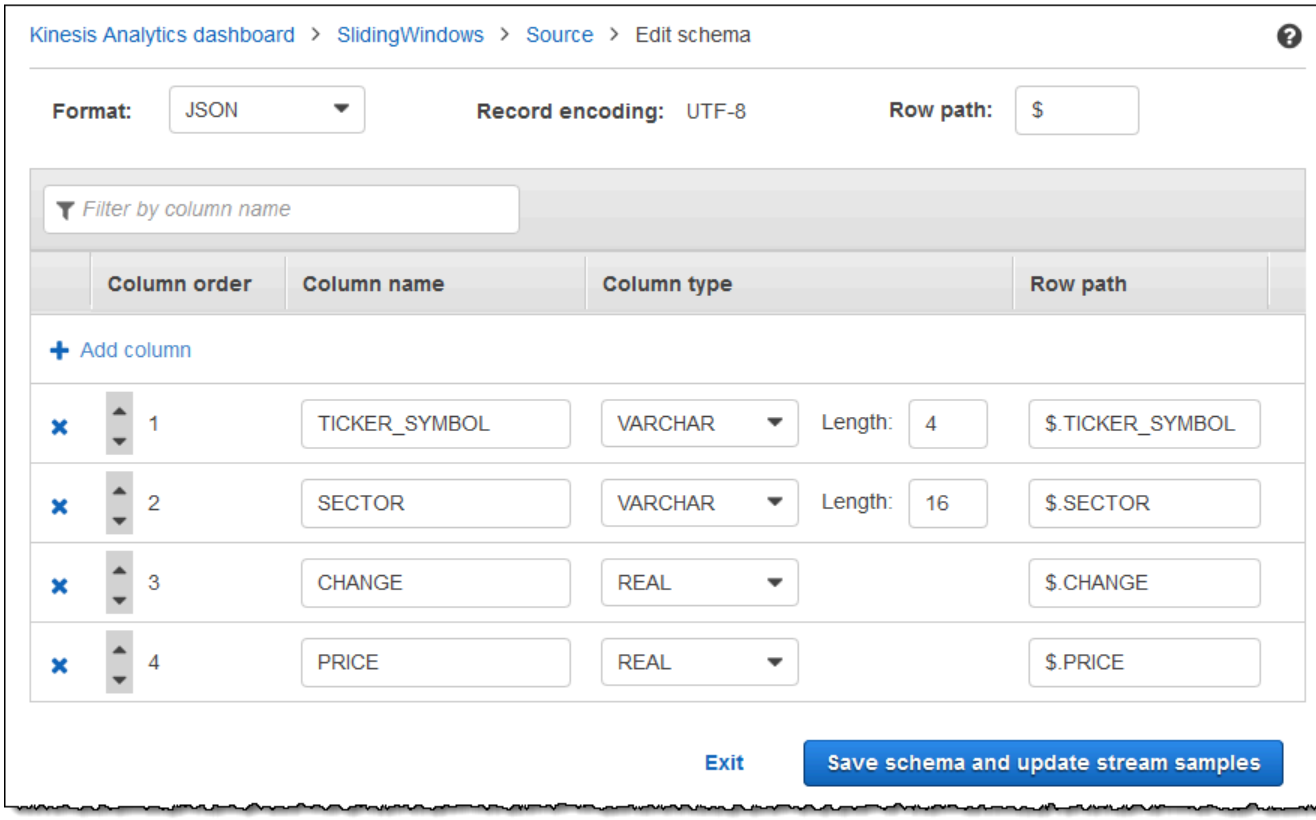
基础 Row path 值是指向包含要将数据导入到的顶级父项的路径。默认情况下，此值为 $。有关更多信息，请参阅 JSONMappingParameters 中的 RecordRowPath。

2. 要删除列，请选择列编号旁的 x 图标。



3. 要重命名列，请在 Column name (列名) 中输入新名称。新列名称不能为空，长度必须超过一个字符，并且不得包含保留的 SQL 关键字。它还必须符合 SQL 普通标识符的命名标准：必须以字母开头，并且只包含字母、下划线字符和数字。
4. 要更改列的数据类型，请在 Column type 中选择新数据类型。如果新数据类型为 CHAR、VARBINARY 或 VARCHAR，请在 Length (长度) 中指定数据长度。有关更多信息，请参阅数据类型。
5. 选择 Save schema and update stream 以保存您的更改。

修改后的架构将显示在编辑器中，类似于以下内容。



如果您的架构具有许多行，您可使用 Filter by column name 来筛选行。例如，要编辑以开头的列名P，例如Price列，请在按列名筛选框P中输入。

编辑 CSV 架构

您可通过以下步骤编辑 CSV 架构。

编辑 CSV 架构

1. 在架构编辑器中，对于 Row delimiter，选择您的传入数据流使用的分隔符。这是您的流中数据记录之间的分隔符 (如换行符)。
2. 对于 Column delimiter，选择您的传入数据流使用的分隔符。这是您的流中数据字段之间的分隔符 (如逗号)。
3. 要添加列，请选择 Add column。

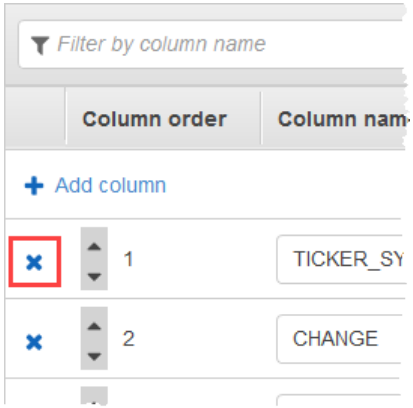
   新列将显示在第一个列位置。要更改列顺序，请选择列名称旁边的向上和向下箭头。

   对于新列，请提供以下信息：

   - 对于 Column name (列名)，输入一个名称。

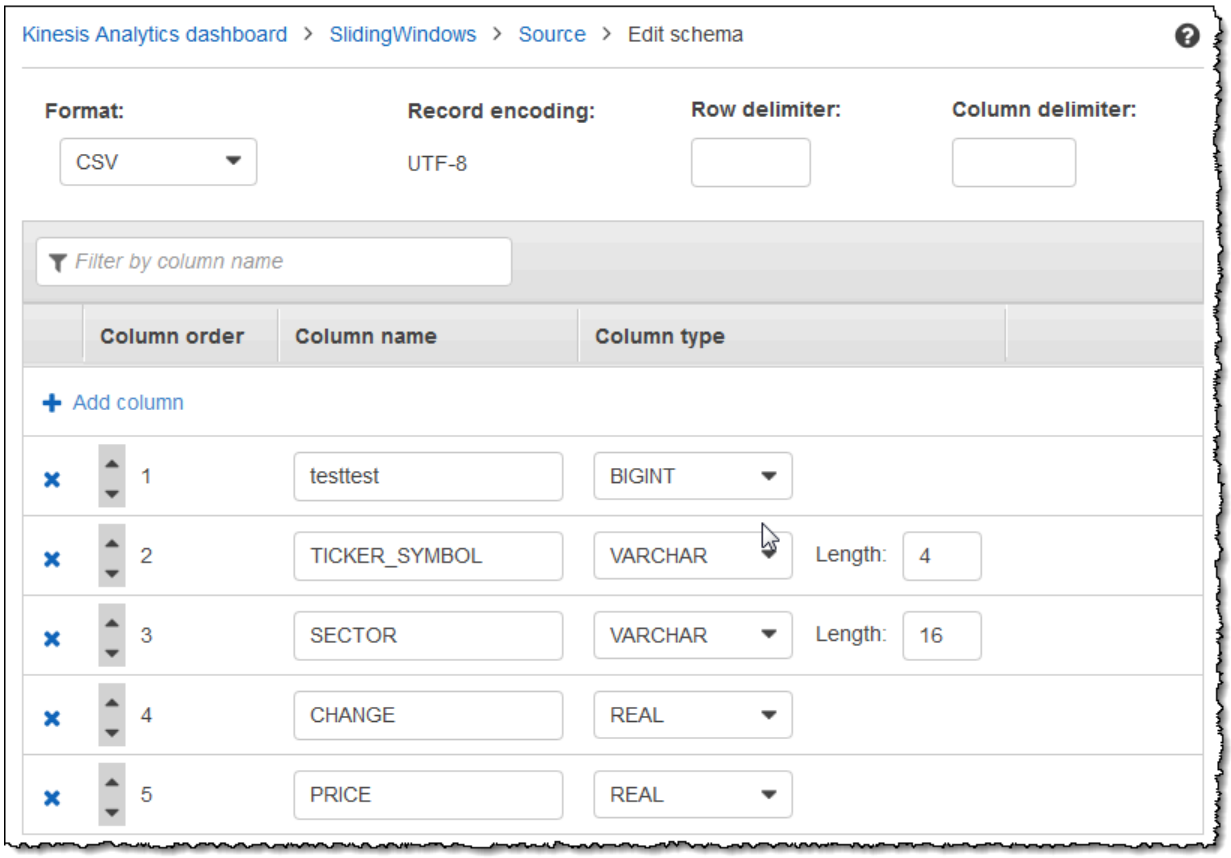
     列名称不能为空，长度必须超过一个字符，并且不得包含保留的 SQL 关键字。它还必须符合 SQL 普通标识符的命名标准：必须以字母开头，并且只包含字母、下划线字符和数字。
   - 对于 Column type (列类型)，输入一个 SQL 数据类型。

     列类型可以是任何受支持的 SQL 数据类型。如果新数据类型为 CHAR、VARBINARY 或 VARCHAR，请在 Length (长度) 中指定数据长度。有关更多信息，请参阅数据类型。
4. 要删除列，请选择列编号旁的 x 图标。



5. 要重命名列，请在 Column name (列名) 中输入新名称。新列名称不能为空，长度必须超过一个字符，并且不得包含保留的 SQL 关键字。它还必须符合 SQL 普通标识符的命名标准：必须以字母开头，并且只包含字母、下划线字符和数字。
6. 要更改列的数据类型，请在 Column type 中选择新数据类型。如果新数据类型为 CHAR、VARBINARY 或 VARCHAR，请在 Length (长度) 中指定数据长度。有关更多信息，请参阅数据类型。
7. 选择 Save schema and update stream 以保存您的更改。

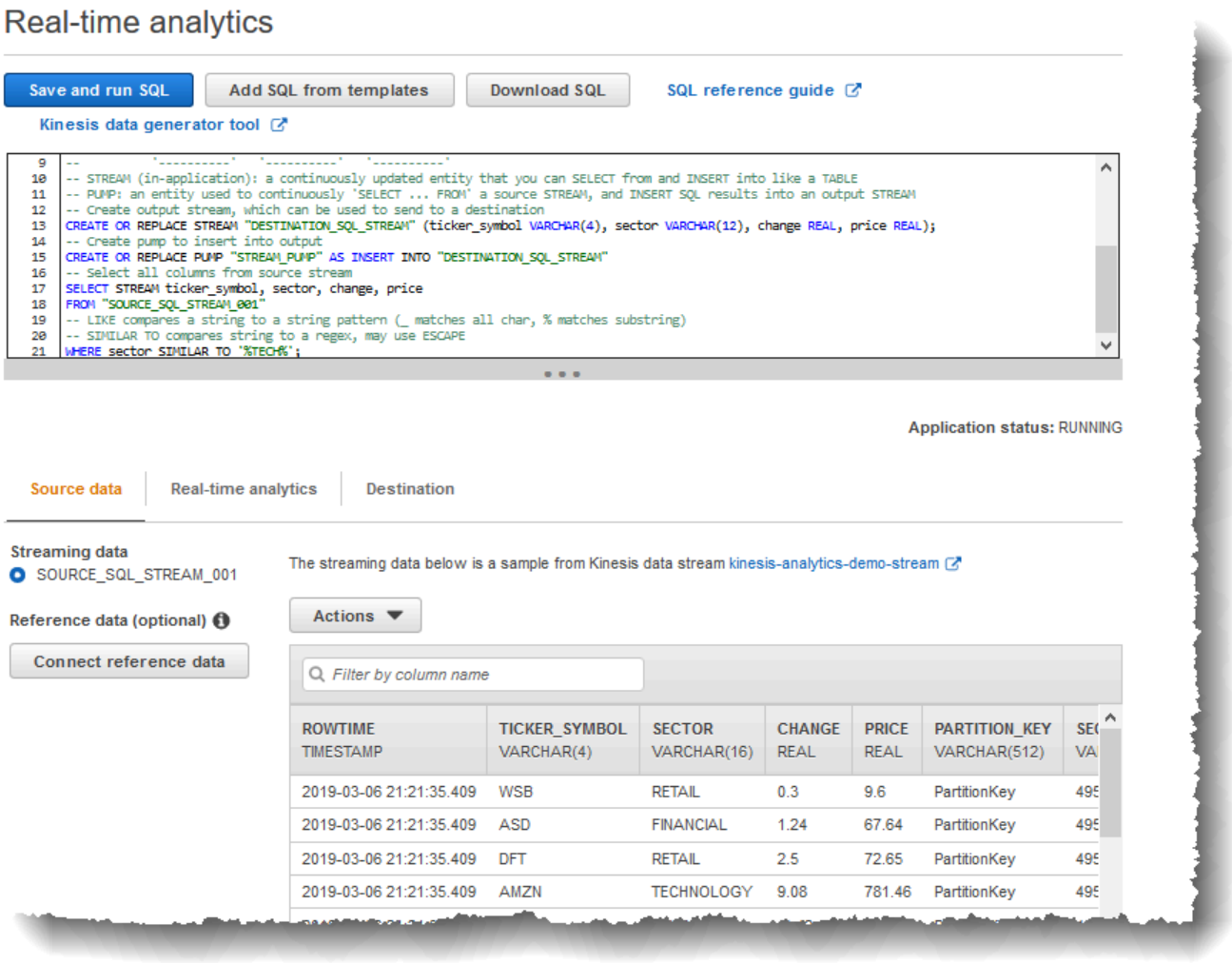
修改后的架构将显示在编辑器中，类似于以下内容。



如果您的架构具有许多行，您可使用 Filter by column name 来筛选行。例如，要编辑以开头的列名P，例如`Price`列，请在按列名筛选框P中输入。

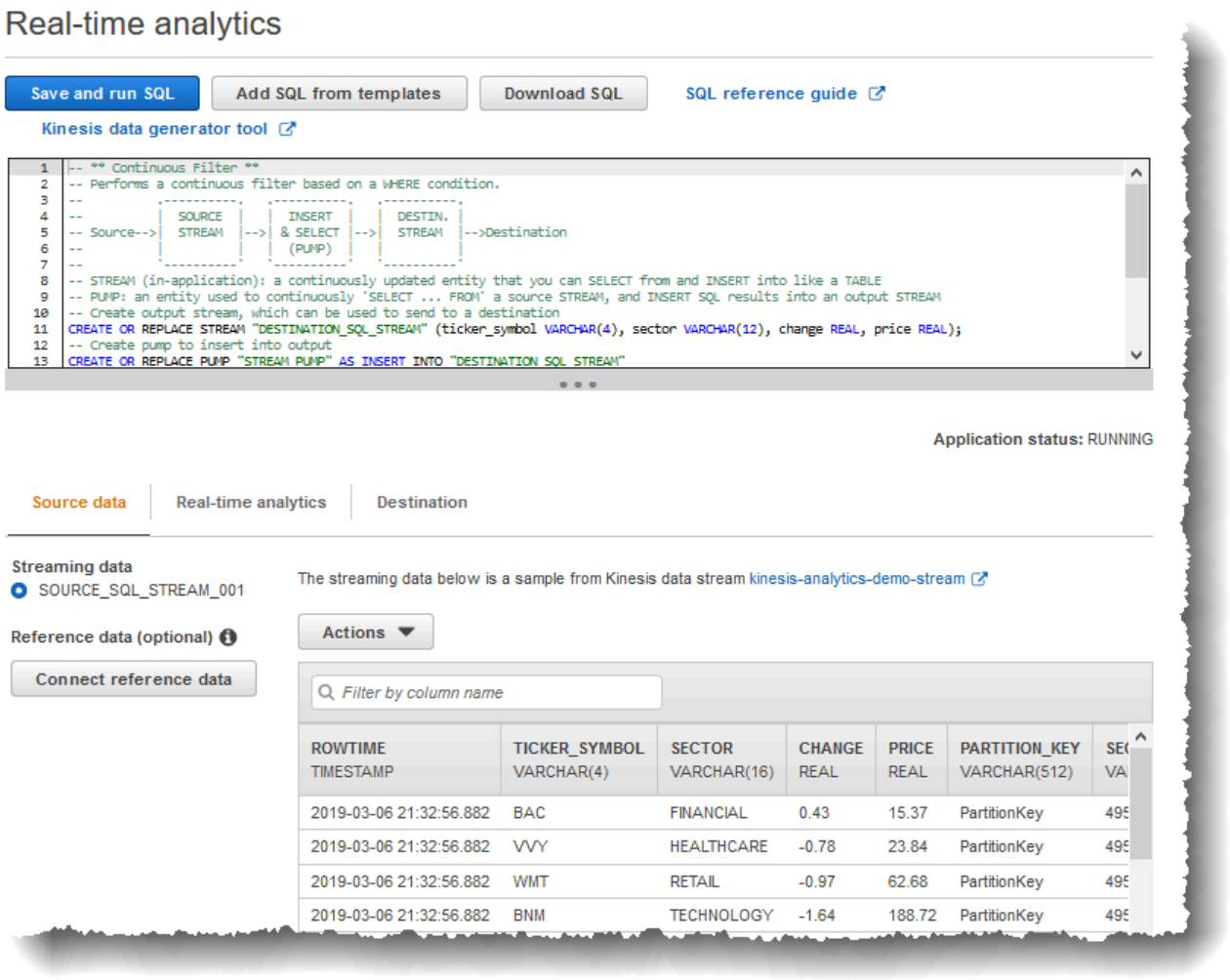
使用 SQL 编辑器

在下文中，您可以找到有关 SQL 编辑器的各个部分及其工作方式的信息。在 SQL 编辑器中，您可以自行编写代码，也可以选择 Add SQL from templates。SQL 模板为您提供示例 SQL 代码，可以帮助您编写常见的 Amazon Kinesis Data Analytics 应用程序。本指南中的示例应用程序使用了其中的一些模板。有关更多信息，请参阅示例应用程序 (p. 78)：



“Source Data”选项卡

Source data 选项卡可标识流式传输源。它还可标识作为此源的映射目标并提供了应用程序输入配置的应用程序内部输入流。



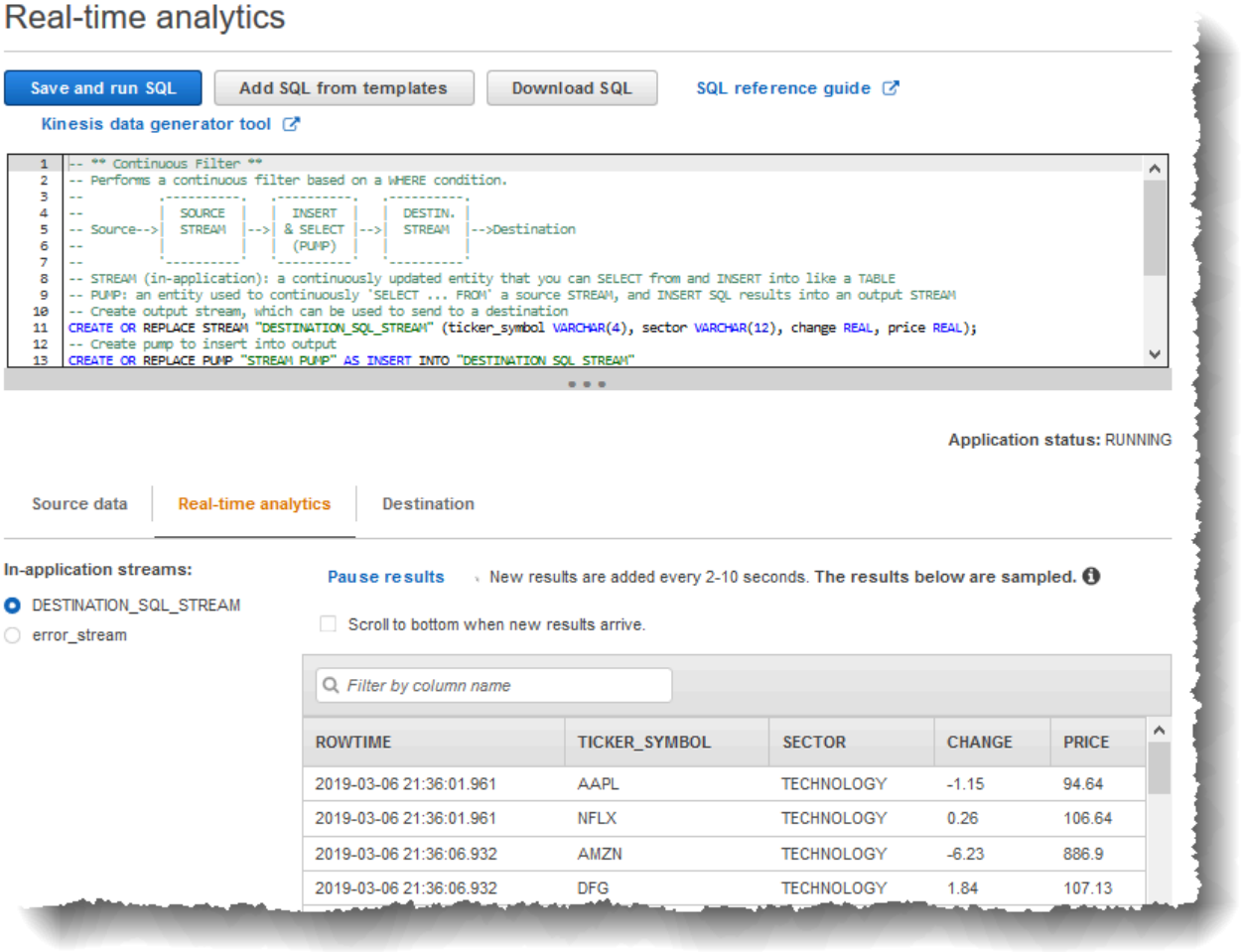
Amazon Kinesis Data Analytics 提供以下时间戳列，因此您无需在输入配置中提供明确的映射：

- ROWTIM E — 应用程序内流中的每一行都有一个名为的特殊列`ROWTIME`。此列是 Kinesis Data Analytics 在第一个应用程序内流中插入该行时的时间戳。
- Aprocisate_Arrival_Tim e — 直播源上的记录包括该`Approximate_Arrival_Timestamp`列。这是流媒体源成功接收和存储相关记录时设置的近似到达时间戳。Kinesis Data Analytics 以如下方式将此列提取到应用程序内部输入流中`Approximate_Arrival_Time`。Amazon Kinesis Data Analytics 仅在映射到流媒体源的应用程序内输入流中提供此列。

这些时间戳值在基于时间的窗口式查询中非常有用。有关更多信息，请参阅窗口式查询 (p. 67)：

“Real-Time Analytics”选项卡

Real-time analytics (实时分析) 选项卡显示了应用程序代码创建的所有应用程序内部流。这组数据流包括 Amazon Kinesis Data Analytics 为所有应用程序提供的错误流 (`error_stream`)。



“Destination”选项卡

Destination (目标) 选项卡可让您配置应用程序输出，以便将应用程序内部流永久保存到外部目标。您可以配置输出，以便将任何应用程序内部流中的数据保存到外部目标。有关更多信息，请参阅 配置应用程序输出 (p. 29)。

流式 SQL 概念

Amazon Kinesis Data Analytics 通过扩展实现了 ANSI 2008 SQL 标准。这些扩展使您可以处理流数据。以下主题介绍了重要的流式 SQL 概念。

主题

应用程序内部流和数据泵

当您配置应用程序输入时，您需要将一个流式传输源映射到已创建的应用程序内部流。数据从流式传输源持续流动到应用程序内部流。应用程序内部流类似于可使用 SQL 语句进行查询的表，但是，它之所以称为流是因为它表示连续的数据流。

> **Note**
>
> 不要将应用程序内数据流与Amazon Kinesis 数据流和 Kinesis Data Firehose 传输流混淆。应用程序内数据流仅存在于Amazon Kinesis Data Analytics 应用程序中。Kinesis 数据流和 Kinesis Data Firehose 传输流独立于您的应用程序而存在。您可以将它们配置为应用程序输入配置中的流式源，或者配置为输出配置中的一个目标。

您还可以根据需要创建更多的应用程序内部流来存储中间查询结果。创建应用程序内部流是一个两步过程。首先，创建应用程序内部流，然后将数据泵送到其中。例如，假设您应用程序的输入配置创建了名为 INPUTSTREAM 的应用程序内部流。在以下示例中，您将创建另一个流 (TEMPSTREAM)，然后从 INPUTSTREAM 将数据泵送到其中。

1. 按如下所示使用三列创建应用程序内部流 (TEMPSTREAM)：

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

   在引号内指定列名，输入时区分大小写。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的标识符。

2. 使用数据泵将数据插入流。数据泵是连续运行的插入查询，它将数据从一个应用程序内部流插入另一个应用程序内部流。以下语句创建一个数据泵 (SAMPLEPUMP)，然后通过从另一个流 (INPUTSTREAM) 选择记录，将数据插入 TEMPSTREAM。

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

```
            inputcolumn3
FROM "INPUTSTREAM";
```

您可以使多个写入器插入一个应用程序内部流，可以从该流选择多个读取器。可以将应用程序内部流视为实施发布/订阅消息发送范式。在此范式中，数据行（包括创建时和接收时）可以通过一串关联的流式 SQL 语句进行处理、解释和转发，无需存储在传统的 RDBMS 中。

在创建应用程序内部流后，您可以执行普通 SQL 查询。

Note

查询流时，会使用基于行或基于时间的窗口绑定大多数 SQL 语句。有关更多信息，请参阅窗口式查询 (p. 67)：

您还可以联接流。有关联接流的示例，请参阅流数据操作：流联接 (p. 76)。

时间戳和 ROWTIME 列

应用程序内部流包含名为 ROWTIME 的特殊列。当 Amazon Kinesis Data Analytics 在第一个应用程序内流中插入一行时，它会存储时间戳。 ROWTIME反映了 Amazon Kinesis Data Analytics 在读取流媒体源后将记录插入到第一个应用程序内数据流的时间戳。之后，该 ROWTIME 值在您的整个应用程序中进行维护。

Note

当您将记录从一个应用程序内数据流传输到另一个应用程序内流时，无需明确复制该ROWTIME列，Amazon Kinesis Data Analytics 会为您复制此列。

Amazon Kinesis Data Analytics 保证ROWTIME数值是单调增加的。您可以在基于时间的窗口式查询中使用此时间戳。有关更多信息，请参阅窗口式查询 (p. 67)：

您可以访问 SELECT 语句中的 ROWTIME 列，就像应用程序内部流中的任何其他列一样。例如：

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

了解流式分析中的各种时间

除了 ROWTIME 之外，在实时流式应用程序中还存在其他类型的时间。这些时间是：

- 事件时间-事件发生时的时间戳。它有时也称为客户端时间。经常需要在分析中使用此时间，因为它是事件发生时的时间。但是，许多事件源（例如手机和 Web 客户端）没有可靠的时钟，这可能会导致时间不准确。此外，连接问题可能会导致记录没有按照事件发生顺序出现在流中。

- 采集时间 — 将记录添加到流媒体源的时间戳。Amazon Kinesis Data StreamsAPPROXIMATE_ARRIVAL_TIME 在提供此时间戳的每条记录中都包含一个名为的字段。有时这也称为服务器端时间。此接收时间通常非常接近事件时间。如果记录接收到流时存在任何种类的延迟，会导致不准确，这种情况通常很少见。此外，接收时间很少出现顺序问题，但由于流数据的分布特点，它也会出现。因此，接收时间通常准确地反映按顺序排列的事件时间。

- 处理时间 — Amazon Kinesis Data Analytics 在第一个应用程序内数据流中插入一行时的时间戳。Amazon Kinesis Data Analytics 在每个应用程序内流中存在的ROWTIME列中提供此时间戳。处理时间总是单调增

加。但如果应用程序滞后，则处理时间不准确。（如果应用程序滞后，则处理时间无法准确反映事件时间）。此 ROWTIME 相对于时钟来说很准确，但可能不是事件实际发生的时间。

在基于时间的窗口式查询中使用这些时间有优点也有缺点。我们建议您选择这些时间中的一个或多个，并根据您的使用案例场景选择一种策略来处理相关缺点。

Note

如果您使用的是基于行的窗口，则时间不是问题，您可以忽略本部分。

我们建议采用双窗口策略，这两个窗口基于不同的时间，即 ROWTIME 和其他时间（接收时间或事件时间）中的一个。

- 使用 ROWTIME 作为第一个窗口，控制查询发送结果的频率，如以下示例所示。它不用作逻辑时间。
- 使用其他时间中您希望与分析关联的逻辑时间。该时间表示事件的发生时间。在以下示例中，分析目标是按股票行情机对记录分组并返回计数。

此策略的优势是它可以使用事件发生时间。它可以轻松处理您的应用程序落后或事件无序到达的情况。当将记录放入应用程序内部流时，如果应用程序落后，记录仍然按第二个窗口中的逻辑时间分组。查询使用 ROWTIME 确保处理顺序。落后的任何记录（与 ROWTIME 值相比，接收时间戳显示的值较早）也会成功处理。

针对入门练习中使用的演示流，考虑以下查询。查询使用 GROUP BY 子句，并在一分钟滚动窗口中发送股票行情机计数。

```
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);
```

在 GROUP BY 中，您首先在一分钟窗口中基于 ROWTIME 对记录分组，然后基于 APPROXIMATE_ARRIVAL_TIME 分组。

结果中的时间戳值已向下舍入为最接近的 60 秒间隔。查询发送的第一组结果显示第一分钟的记录。发送的第二组结果基于 ROWTIME 显示后续分钟内的记录。最后一条记录指示应用程序在应用程序内部流中放入记录是最晚的（与接收时间戳相比，它显示最晚的 ROWTIME 值）。

```
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.
```

您可以将结果推送到下游数据库，以汇总结果来得到每分钟的最终准确计数。例如，您可以将应用程序输出配置为将结果保存到可以写入 Amazon Redshift 表的 Kinesis Data Firehose 传输流中。结果出现在 Amazon Redshift 表中后，您可以查询该表以计算分组总数Ticker_Symbol。对于 XYZ，总数是精确的 (6+1)，即使记录延迟到达也是如此。

连续查询

对流数据连续执行流查询。此连续执行使许多方案得以实现，例如应用程序连续查询流和生成警报的能力。

在入门练习中，您创建了一个名为 SOURCE_SQL_STREAM_001 的应用程序内部流。它持续接收来自演示流（Kinesis 数据流）的股票价格。架构如下：

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

假设您对超过 15% 的股票价格变化感兴趣。您可以在应用程序代码中使用以下查询。此查询连续运行，当检测到大于 15% 的股票价格变化时，此查询会发送记录。

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

使用以下程序设置 Amazon Kinesis Data Analytics 应用程序并测试此查询。

测试查询

1. 按照入门练习创建应用程序。
2. 使用先前的 SELECT 查询替换应用程序代码中的 SELECT 语句。下面显示得到的应用程序代码：

```
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;
```

窗口式查询

对应用程序内部流连续执行应用程序代码中的 SQL 查询。应用程序内部流表示通过您的应用程序持续流动的无边界数据。因此，要从该连续更新输入获取结果集，通常可以使用根据时间或行定义的窗口来限制查询。这些查询也称为窗口式 SQL。

对于基于时间的窗口式查询，需要以时间为单位指定窗口大小（例如，一分钟窗口）。这需要在应用程序内部流中有一个单调递增的时间戳列。（新行的时间戳大于或等于前一行。）Amazon Kinesis Data AnalyticsROWTIME 为每个应用程序内数据流提供了一个名为的时间戳列。在指定基于时间的查询时，可以使用该列。对于您的应用程序，可以选择其他某个时间戳选项。有关更多信息，请参阅时间戳和 ROWTIME 列 (p. 65)：

对于基于行的窗口式查询，可以使用行数为单位指定窗口大小。

您可以指定查询以滚动窗口、滑动窗口或错开窗口方式处理记录，具体取决于您的应用程序需求。Kinesis Data Analytics 支持以下窗口类型：

- 交错窗口 (p. 68)：一个使用基于时间的键控窗口聚合数据的查询，这种窗口在数据到达时打开。这些键允许多个重叠的窗口。这是使用基于时间的窗口聚合数据的推荐方法，因为与Tumbling窗口相比，Stagger Windows可以减少延迟或 out-of-order 数据。
- 滚动窗口 (p. 72)：一个使用基于时间的不同窗口聚合数据的查询，这些窗口以固定时间间隔打开和关闭。
- 滑动窗口 (p. 73)：一个使用固定时间或行计数间隔连续聚合数据的查询。

交错窗口

使用交错窗口 是一种窗口化方法，适用于分析到达时间不一致的数据组。这种方法非常适合任何时间序列分析使用案例，例如一组相关的销售或日志记录。

例如，VPC 流日志具有一个大约 10 分钟的捕获窗口。但是，如果您在客户端上聚合数据，则最多可以具有 15 分钟的捕获窗口。交错窗口非常适合用于聚合这些日志进行分析。

交错窗口解决了多条相关记录不属于同一时间限制窗口的问题，例如在使用了滚动窗口的情况下。

滚动窗口的部分结果

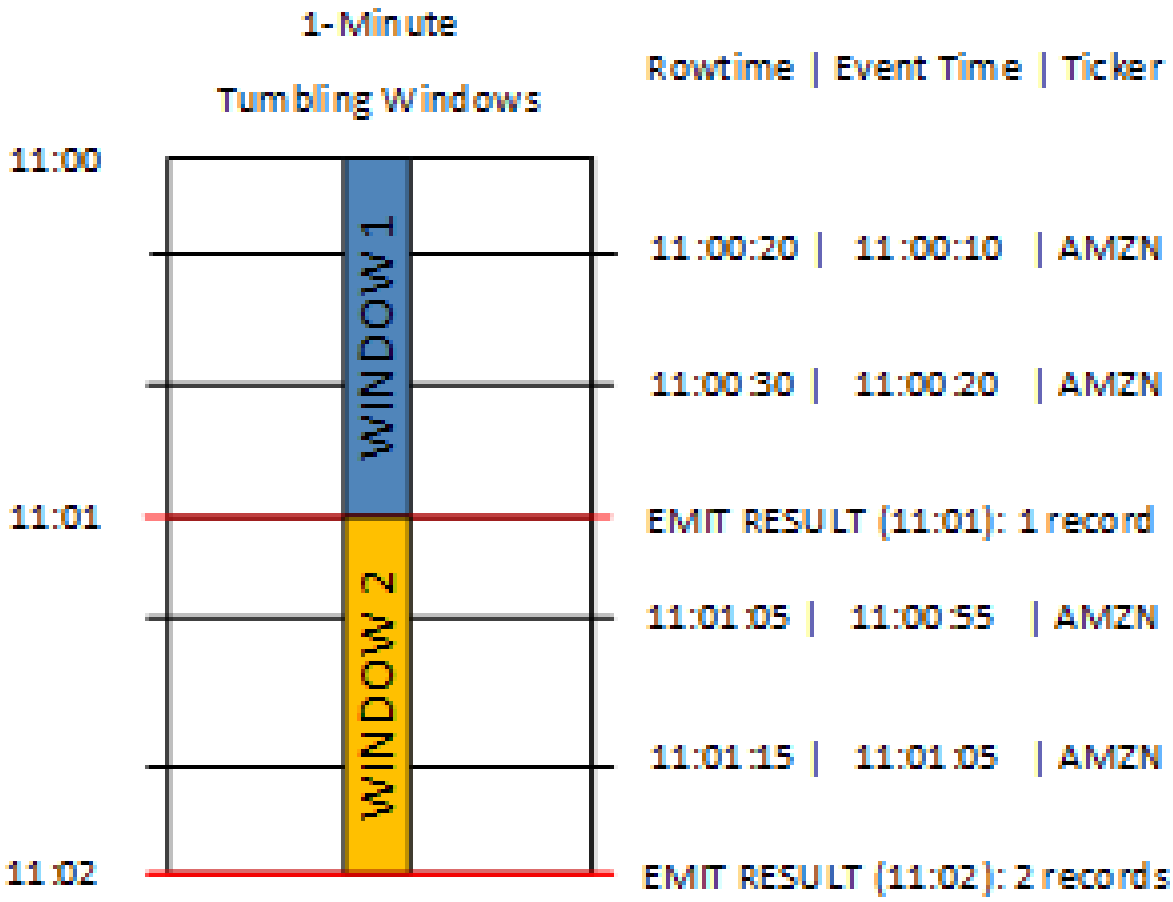
滚动窗口 (p. 72)用于聚合后期 out-of-order 数据存在某些限制。

如果使用滚动窗口来分析与时间相关的多组数据，则个别记录可能属于单独的窗口。因此，必须稍后组合每个窗口的部分结果，以便为每组记录生成完整的结果。

在以下滚动窗口查询中，记录按行时间、事件时间和股票代码分组为若干窗口：

```
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);
```

在下图中，应用程序根据交易发生的时间（事件时间）以一分钟的粒度计算它收到的交易数量。应用程序可以使用滚动窗口根据行时间和事件时间对数据进行分组。该应用程序接收四条记录，所有记录都在彼此的一分钟之内到达。它按行时间、事件时间和股票代码对记录进行分组。因为一些记录在第一个滚动窗口结束后到达，所以记录并非全部位于同一个一分钟的滚动窗口内。



上图包含以下事件。

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

滚动窗口应用程序的结果集类似于以下内容。

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

在前面的结果集中，返回了三个结果：

- ROWTIME 为 11:01:00 的记录，它聚合前两个记录。
- 11:02:00 的记录，它仅聚合第三条记录。此记录在第二个窗口中有一个 ROWTIME，但在第一个窗口中有一个 EVENT_TIME。
- 11:02:00 的记录，它仅聚合第四条记录。

要分析完整的结果集，必须在持久性存储中聚合记录。这给应用程序增加了复杂性和处理要求。

完整结果与交错窗口

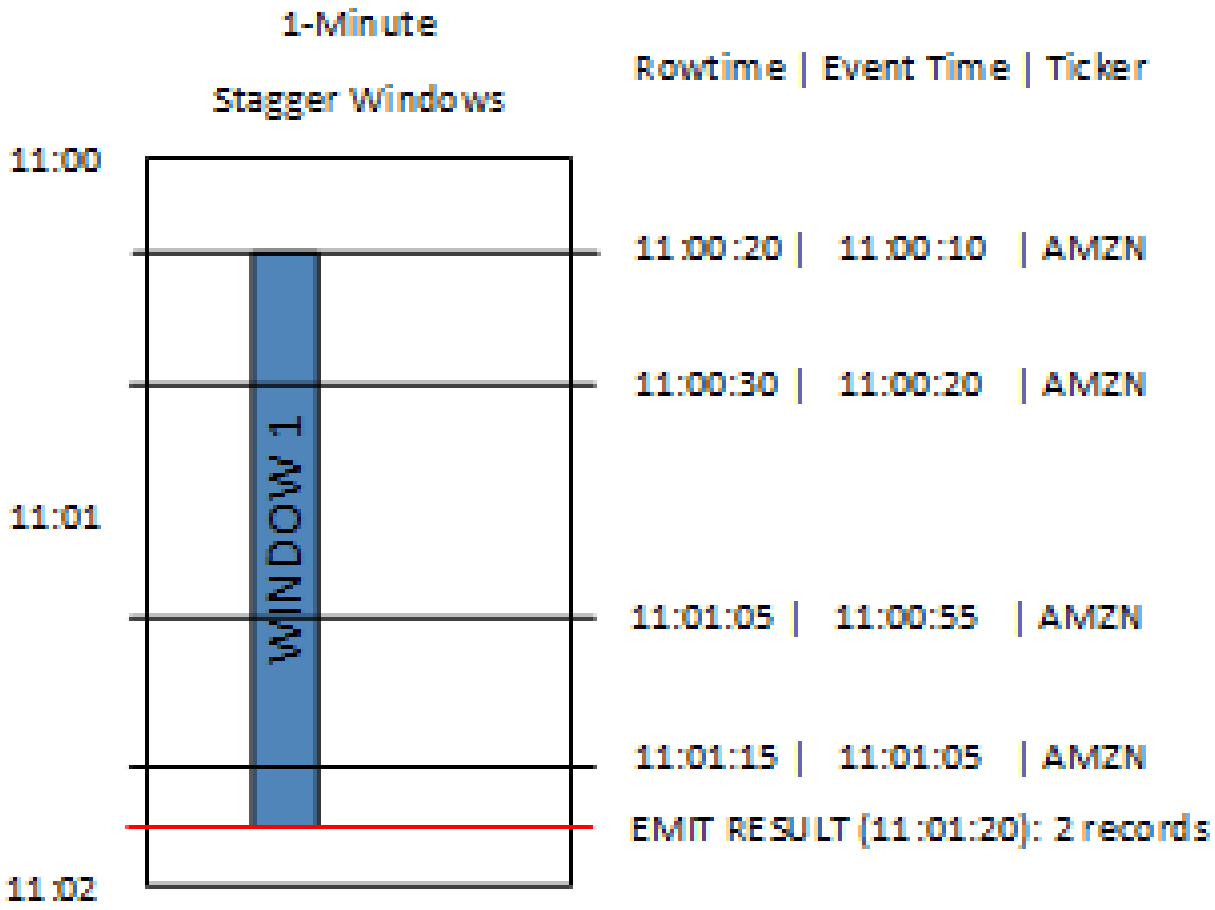
为了提高分析时间相关数据记录的准确性，Kinesis Data Analytics 提供了一种名为错开窗口的新窗口类型。在此窗口类型中，窗口在与分区键匹配的第一个事件到达时打开，而不是在固定的时间间隔打开。窗口根据指定的期限关闭，期限是从窗口打开的时间开始计算的。

交错窗口是窗口子句中每个键分组的单独时间限制窗口。应用程序将窗口子句的每个结果聚合在其自己的时间窗口内，而不是对所有结果使用单个窗口。

在以下交错窗口查询中，记录按事件时间和股票代码分组为若干窗口：

```
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);
```

在下图中，事件按事件时间和股票代码聚合到交错窗口中。



上图包含以下事件，这些事件与分析的滚动窗口应用程序相同：

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

交错窗口应用程序的结果集类似于以下内容。

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

返回的记录聚合了前三条输入记录。记录按一分钟的交错窗口分组。当应用程序收到第一条 AMZN 记录（ROWTIME 为 11:00:20）时，交错窗口开始。当 1 分钟交错窗口到期时 (11:01:20)，对于包含位于交错窗口

内的结果（基于 ROWTIME 和 EVENT_TIME）的记录，将被写入输出流。使用交错窗口，在一分钟窗口内具有 ROWTIME 和 EVENT_TIME 的所有记录都将在单个结果中发出。

最后一条记录（其 EVENT_TIME 位于一分钟聚合之外）是单独聚合的。这是因为 EVENT_TIME 是用于将记录分成多个结果集的分区键之一，而第一个窗口的 EVENT_TIME 的分区键是 11:00。

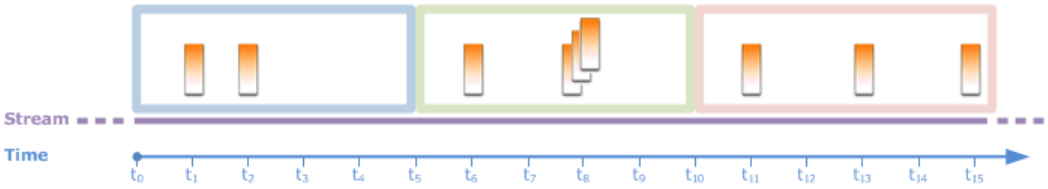
交错窗口的语法在特殊子句 WINDOWED BY 中定义。对于流式处理聚合，使用此子句代替 GROUP BY 子句。该子句紧跟在可选的 WHERE 子句之后和 HAVING 子句之前。

交错窗口在 WINDOWED BY 子句中定义，并带有两个参数：分区键和窗口长度。分区键对传入的数据流进行分区，并定义窗口何时打开。当流中出现具有唯一分区键的第一个事件时，将打开一个交错窗口。在经过由窗口长度定义的固定时间段之后，交错窗口关闭。以下代码示例说明了此语法：

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

滚动窗口（使用 GROUP BY 组的聚合）

当一个窗口式查询以非重叠方式处理每个窗口时，这样的窗口称为滚动窗口。在这种情况下，应用程序内部流上的每个记录属于特定窗口。它只处理一次（当查询处理记录所属的窗口时）。



例如，使用 GROUP BY 子句的聚合查询在一个滚动窗口中处理行。入门练习中的演示流接收股票价格数据，而这些数据映射到应用程序中的应用程序内部流 SOURCE_SQL_STREAM_001。这个流具有以下架构。

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

在您的应用程序代码中，假设您希望针对一分钟窗口找到每个股票行情机的聚合（最小、最大）价格。您可以使用以下查询。

```
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);
```

上述示例是一个基于时间的窗口式查询。该查询根据 ROWTIME 值将记录分组。对于每分钟进行的报告，STEP 函数将 ROWTIME 值向下舍入到最接近的分钟。

Note

您也可以使用 FLOOR 函数将记录分组为不同的窗口。但是，FLOOR 只能将时间值向下舍入到完整的时间单位（小时、分钟、秒等）。建议使用 STEP 以将记录分组为不同的滚动窗口，因为它可以将值向下舍入到任意间隔，例如，30 秒。

该查询是非重叠 (滚动) 窗口示例。GROUP BY 子句在一分钟窗口内对记录进行分组，每个记录属于一个特定窗口 (不重叠)。查询每分钟发送一个输出记录，在其中提供在特定分钟时记录的最小/最大股票行情机价格。在根据输入数据流生成周期性报告时，此类查询很有用。在本示例中，报告是每分钟生成的。

测试查询

1. 按照入门练习设置应用程序。
2. 使用先前的 SELECT 查询替换应用程序代码中的 SELECT 语句。下面显示得到的应用程序代码：

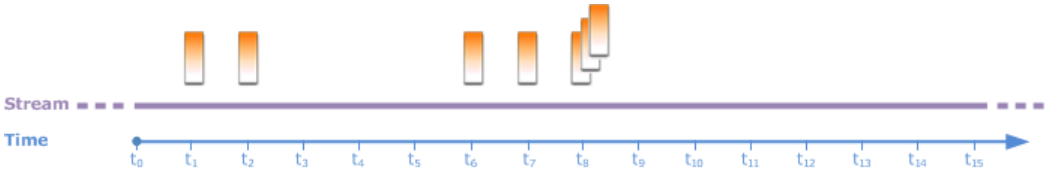
```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                                    ticker_symbol VARCHAR(4),
                                    Min_Price     DOUBLE,
                                    Max_Price     DOUBLE);
-- CREATE OR REPLACE PUMP to insert into output
CREATE OR REPLACE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

滑动窗口

您可以不使用 GROUP BY 对记录分组，而是定义基于时间或基于行的窗口。您应通过添加显式 WINDOW 子句执行此操作。

在这种情况下，当窗口随时间推移时，当直播中出现新记录时，Amazon Kinesis Data Analytics 会发出输出。Kinesis Data Analytics 通过处理窗口中的行来发出此输出。窗口在这种类型的处理中可以重叠，一个记录可以属于多个窗口并且可随各个窗口一起处理。以下示例说明了滑动的窗口。

考虑创建一个简单的查询对流中的记录进行计数。此示例假定有一个 5 秒的窗口。在以下示例流中，新记录在时间 t_1、t_2、t 和 t_6 到达$_7$，三条记录在时间 t_8 秒到达。



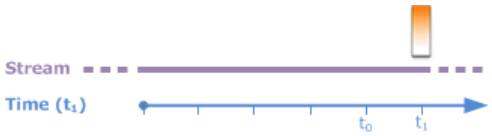
记住以下内容：

- 此示例假定有一个 5 秒的窗口。该 5 秒窗口持续随着时间滑动。
- 对于进入窗口的每一行，滑动窗口会发送输出行。应用程序启动后不久，您会看到查询针对出现在流中的每个新记录发送输出，即使尚未经过 5 秒窗口。例如，当记录出现在第一秒和第二秒时，查询会发送输出。稍后，查询会处理 5 秒窗口中的记录。
- 该窗口随着时间滑动。如果流中的旧记录落后于窗口，查询将不会发送任何输出，除非流中也有一个新记录落在该 5 秒窗口中。

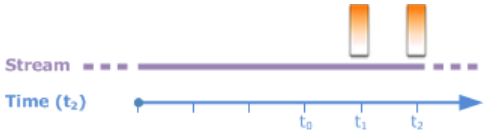
假设查询从 t 开始执行$_0$。那么，将出现以下情况：

1. 在时间 t_0，查询开始。查询不发送输出 (计数值)，因为此时没有记录。

2. 在时间 t 时$_1$，流上会出现一条新记录，查询发出 count 值 1。

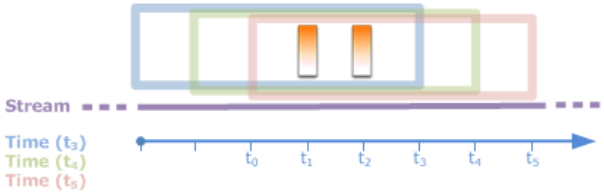


3. 在时间 t_2 处，出现另一条记录，查询发出计数 2。

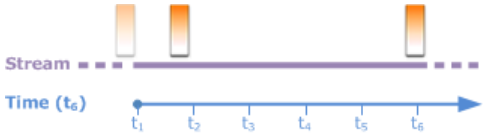


4. 此 5 秒窗口随着时间滑动：
   - 在 t 处$_3$，滑动窗口 t_3 到 t_0
   - 在 t4（将窗口 t 滑动$_4$到 t_0）
   - 在 t_5 处滑动窗口 t_5 —t_0

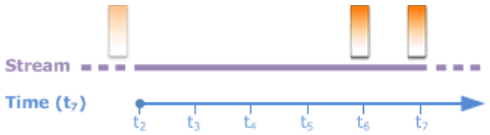
   在所有这些时候，5 秒窗口的记录都是一样的，没有新记录。因此，查询不会发送任何输出。



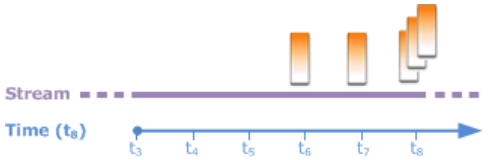
5. 在时间 t 时$_6$，5 秒的窗口为（t_6 到 t_1）。该查询在 t 处检测到一条新记录$_6$，因此它发出输出 2。t 处的记录$_1$已不在窗口中，也不算在内。



6. 在时间 t 时$_7$，5 秒的窗口是 t_7 到 t_2。该查询在 t 处检测到一条新记录$_7$，因此它发出输出 2。t_2 处的记录不再在 5 秒窗口中，因此不计算在内。



7. 在时间 t 时$_8$，5 秒的窗口是 t_8 到 t_3。查询检测到三个新记录，因此发送了记录计数 5。



总之，窗口是固定大小，并且随时间滑动。当出现新记录时，查询会发送输出。

Note

我们建议使用滑动窗口的时间不要超过 1 小时。如果您使用时间更长的窗口，应用程序在常规系统维护之后需要更长的时间才能重新启动。这是因为必须再次从流中读取源数据。

以下示例查询使用 WINDOW 子句定义窗口和执行聚合。由于查询不指定 GROUP BY，因此查询使用滑动窗口方法处理流中的记录。

示例 1：使用一个 1 分钟滑动窗口处理流

在填充应用程序内部流 SOURCE_SQL_STREAM_001 时，请考虑“入门”练习中的演示流。下面是架构。

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

假设您希望应用程序使用 1 分钟滑动窗口计算聚合。也就是说，对于出现在流中的每个新记录，您希望应用程序通过对前面的 1 分钟窗口中的记录应用聚合来发送输出。

您可以使用以下基于时间的窗口式查询。查询使用 WINDOW 子句定义 1 分钟范围间隔。WINDOW 子句中的 PARTITION BY 按照滑动窗口中的股票行情机值对记录进行分组。

```
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);
```

测试查询

1. 按照入门练习设置应用程序。
2. 使用先前的 SELECT 查询替换应用程序代码中的 SELECT 语句。生成的应用程序代码如下。

```
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);
```

示例 2：对滑动窗口应用聚合的查询

针对演示流的以下查询将返回一个 10 秒窗口中，每个股票行情机的价格的平均百分比变化。

```
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);
```

测试查询

1. 按照入门练习设置应用程序。
2. 使用先前的 SELECT 查询替换应用程序代码中的 SELECT 语句。生成的应用程序代码如下。

```
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);
```

示例 3：从同一流的多个滑动窗口查询数据

您可以编写查询以发送输出，其中的每个列值都是使用同一流上定义的不同滑动窗口计算的。

在以下示例中，查询将发送输出股票行情机、价格、a2 和 a10。查询将发送股票代码的输出，这些代码的两行移动平均值超过了 10 行移动平均值。a2 和 a10 列值派生自 2 行和 10 行滑动窗口。

```
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);
```

要针对演示流测试此查询，请按照示例 1 (p. 75)中介绍的测试过程操作。

流数据操作：流联接

您可以在应用程序中拥有多个应用程序内部流。您可以编写 JOIN 查询关联到达这些流的数据。例如，假设您拥有以下应用程序内部流：

- OrderStream— 接收正在下达的库存订单。

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

- TradeStream— 接收这些订单的股票交易结果。

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

以下 JOIN 查询示例与这些流上的数据相关联。

示例 1：报告所提交订单在 1 分钟内有成交记录的订单

在此示例中，查询联接了 OrderStream 和 TradeStream。但是，由于我们只需要在下订单后 1 分钟内产生的交易，因此查询针对 TradeStream 定义 1 分钟的窗口。有关窗口式查询的信息，请参阅滑动窗口 (p. 73)。

```
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;
```

您可以按如下所示使用 WINDOW 子句并编写前述查询来明确定义窗口：

```
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)
```

当您将此查询包含在您的应用程序代码中时，应用程序代码将连续运行。对于 OrderStream 上到达的各个记录，如果在下订单后的 1 分钟窗口内存在交易，则应用程序发送输出。

前述查询中的联接是内部联接，对于 TradeStream 中存在匹配记录的 OrderStream，该查询会在其中发出记录（反之亦然）。使用外部连接可以创建另一个有趣场景。假设您需要查询在提交股票订单的 1 分钟内没有交易的订单，以及在同一窗口内为其他一些订单报告交易。这是外部联接 示例。

```
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;
```

示例应用程序

本节提供了在Amazon Kinesis Data Analytics 中创建和使用应用程序的示例。它们包括示例代码和 step-by-step 说明，可帮助您创建 Kinesis 数据分析应用程序和测试结果。

在开始探索这些示例之前，建议您先查看适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)和Amazon Kinesis Data Analytics 入门 (p. 40)。

主题

迁移到 Kinesis Data Analytics for Apache Flink：示例

> Warning
>
> 对于新项目，我们建议您使用新的 Kinesis Data Analytics 工作室，而不是 SQL 应用程序的 Kinesis Data Analytics。Kinesis Data Analytics Studio 将易用性与高级分析功能相结合，使您能够在几分钟内构建复杂的流处理应用程序。

要将您的工作负载迁移到 Kinesis Data Analytics Studio 或 Apache Flink 版 Kinesis Data Analytics，本部分提供了可用于常见用例的查询翻译。

> Note
>
> 适用于 Apache Flink 的 Kinesis Data Analytics 和 Kinesis Data Analytics Studio 提供基于 SQL 的 Kinesis Data Analytics 应用程序所不具备的高级数据流处理功能。其中包括一次性处理语义、事件时间窗口、使用用户定义函数和自定义集成的可扩展性、命令式语言支持、持久的应用程序状态、水平扩展、对多个数据源的支持、可扩展的集成等。这些对于确保数据流处理的准确性、完整性、一致性和可靠性至关重要。

在探索这些示例之前，我们建议您先阅读使用带有 Apache Flink 托管服务的 Studio 笔记本。

主题

在 Kinesis Data Analytics 工作室中为 SQL 查询重新创建 Kinesis Data Analytics

下表提供了基于 SQL 的 Kinesis Data Analytics 应用程序对 Kinesis Data Analytics Studio 的常见查询的翻译。

多步骤应用程序

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";
```

Kinesis Data Analytics 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;
```

转变 DateTime 价值观

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"
```

Kinesis Data Analytics 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;
```

简单提醒

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
```

Kinesis Data Analytics 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;
```

限流提醒

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;
```

Kinesis Data Analytics 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;
```

聚合查询的部分结果

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
      )
;
```

Kinesis Data Analytics 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;
```

转换字符串值

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";
```

Kinesis Data Analytics 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;
```

使用正则表达式替换子字符串

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";
```

Kinesis Data Analytics 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;
```

正则表达式日志解析

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;
```

Kinesis Data Analytics 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;
```

转变 DateTime 价值观

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"
```

Kinesis Data Analytics Studio

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

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

```
         EVENT_TIME - INTERVAL '5' MINUTE AS FIVE_MINUTES_BEFORE,
         UNIX_TIMESTAMP() AS EVENT_UNIX_TIMESTAMP,
         DATE_FORMAT(EVENT_TIME, 'yyyy-MM-dd hh:mm:ss') AS EVENT_TIMESTAMP_AS_CHAR,
         EXTRACT(SECOND
      FROM
         EVENT_TIME) AS EVENT_SECOND
      FROM
         DESTINATION_SQL_STREAM;
```

Windows 和聚合

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);
```

Kinesis Data Analytics 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;
```

使用 Rowtime 翻滚窗口

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);
```

Kinesis Data Analytics 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;
```

检索最常出现的值 (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
   )
;
```

Kinesis Data Analytics 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;
```

大约 Top-K 项目

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));
```

Kinesis Data Analytics 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
```

解析 Web 日志（W3C_LOG_PARSE 函数）

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);
```

Kinesis Data Analytics 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;
```

将字符串拆分成多个字段（VARIABLE_COLUMN_LOG_PARSE 函数）

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;
```

Kinesis Data Analytics Studio

```
%flink.ssql(type=update)
DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001 CREATE TABLE SOURCE_SQL_STREAM_001 ( log
 VARCHAR(1024))
   WITH ( 'connector' = 'kinesis',
          'stream' = 'SOURCE_SQL_STREAM_001',
          'aws.region' = 'us-east-1',
          'scan.stream.initpos' = 'LATEST',
          'format' = 'json',
          'json.timestamp-format.standard' = 'ISO-8601');

% flink.ssql(type=update)
   select
      SPLIT_INDEX(log, ' ', 0),
      SPLIT_INDEX(log, ' ', 1),
      SPLIT_INDEX(log, ' ', 2),
      SPLIT_INDEX(log, ' ', 3),
      SPLIT_INDEX(log, ' ', 4),
      SPLIT_INDEX(log, ' ', 5)
```

```
)
from
   SOURCE_SQL_STREAM_001;
```

Joins

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   ticker_symbol VARCHAR(4),
   "Company" varchar(20),
   sector VARCHAR(12),
   change DOUBLE,
   price DOUBLE);
CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM ticker_symbol,
      "c"."Company",
      sector,
      change,
      priceFROM "SOURCE_SQL_STREAM_001"
      LEFT JOIN
         "CompanyName" as "c"
         ON "SOURCE_SQL_STREAM_001".ticker_symbol = "c"."Ticker";
```

Kinesis Data Analytics 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;
```

错误

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%';
```

Kinesis Data Analytics 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;
```

示例：转换数据

有时，您的应用程序代码必须预处理传入的记录，然后才能在 Amazon Kinesis Data Analytics 中执行任何分析。此情况是由多种原因导致的，例如，不符合支持的记录格式的记录会导致应用程序内部输入流中出现非规范化的列。

此部分提供了有关如何使用可用的字符串函数来规范化数据、如何从字符串列中提取所需信息等操作的示例，还指明了您可能发现很有用的日期时间函数。

使用 Lambda 预处理流

有关使用 Amazon Lambda 预处理流的信息，请参阅使用 Lambda 函数预处理数据 (p. 19)。

主题

示例：转换字符串值

Amazon Kinesis Data Analytics 支持流媒体源上记录的 JSON 和 CSV 等格式。有关详细信息，请参阅RecordFormat (p. 305)。然后，这些记录根据输入配置映射到应用程序内部流中的行。有关详细信息，请参阅配置应用程序输入 (p. 4)。输入配置指定流式传输源中的记录字段如何映射到应用程序内部流中的列。

当流式传输源中的记录遵循支持的格式时，此映射有效，这会导致应用程序内部流具有规范化数据。但是，如果流式传输源中的数据不符合支持的标准怎么办？ 例如，如果流式传输源包含点击流数据、IoT 传感器和应用程序日志等数据时该怎么办呢？

请考虑以下示例：

- 流式源包含应用程序日志 - 应用程序日志遵循标准 Apache 日志格式，并使用 JSON 格式写入到流。

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

  有关标准 Apache 日志格式的更多信息，请参阅 Apache 网站上的日志文件。

- 流媒体源包含半结构化数据-以下示例显示了两条记录。Col_E_Unstructured 字段值是一系列逗号分隔的值。这里总共有五列：前四列包含字符串类型的值，最后一列包含逗号分隔的值。

```
{ "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"}
```

- 流式传输源中的记录包含 URL，需要部分 URL 域名才能进行分析。

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

此种情况下，以下包含两个步骤的过程通常适用于创建包含规范化数据的应用程序内部流：

1. 配置应用程序输入以便将非结构化字段映射到创建的应用程序内部输入流中的 VARCHAR(N) 类型的列。
2. 在应用程序代码中，使用字符串功能将这一个列拆分为多个列，并将行保存到其他应用程序内部流。应用程序代码创建的该应用程序内部流将包含规范化数据。然后，您可以对该应用程序内部流进行分析。

Amazon Kinesis Data Analytics 提供以下字符串操作、标准 SQL 函数以及用于处理字符串列的 SQL 标准扩展：

- 字符串运算符-LIKE 和SIMILAR等运算符在比较字符串时很有用。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的字符串运算符。
- SQL 函数-以下函数在操作单个字符串时很有用。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的字符串和搜索函数。
  - CHAR_LENGTH – 提供字符串的长度。
  - INITCAP – 返回输入字符串的转换后版本，在这类字符串中，每个以空格分隔的单词的第一个字符都是大写的，所有其他字符都是小写的。
  - LOWER/UPPER – 将字符串转换为小写或大写。
  - OVERLAY – 使用第二个字符串参数 (替代字符串) 替换第一个字符串参数的一部分 (原始字符串)。
  - POSITION - 在某个字符串中搜索其他字符串。
  - REGEX_REPLACE – 将子字符串替换为备用子字符串。
  - SUBSTRING - 从特定位置开始提取部分源字符串。
  - TRIM – 从源字符串的开头或结尾删除指定字符的实例。
- SQL 扩展-这些扩展对于处理非结构化字符串（例如日志和 URI）很有用。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的日志解析函数。
  - FAST_REGEX_LOG_PARSER— 工作原理与正则表达式解析器类似，但它需要几个快捷方式才能确保更快的结果。例如，较快的正则表达式解析程序会在找到第一个匹配项时停止（称为延迟语义）。
  - FIXED_COLUMN_LOG_PARSE – 分析固定宽度的字段，自动将其转换为指定的 SQL 类型。
  - REGEX_LOG_PARSE – 根据默认的 Java 正则表达式模式分析字符串。
  - SYS_LOG_PARSE – 分析 UNIX/Linux 系统日志中的常见条目。
  - VARIABLE_COLUMN_LOG_PARSE – 将输入字符串拆分为多个由分隔符或分隔符字符串分隔的字段。
  - W3C_LOG_PARSE – 可用于快速格式化 Apache 日志。

有关使用这些函数的示例，请参阅以下主题：

主题

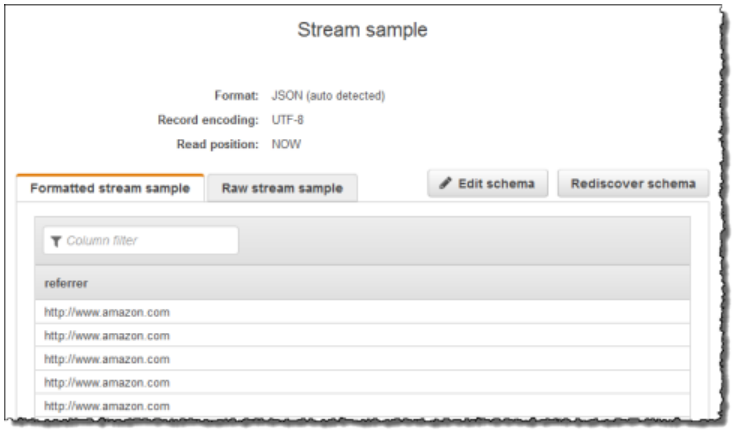
示例：提取部分字符串 (SUBSTRING 函数)

此示例使用SUBSTRING函数在 Amazon Kinesis Data Analytics 中转换字符串。SUBSTRING 函数从特定位置开始提取部分源字符串。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 S UBST RING。

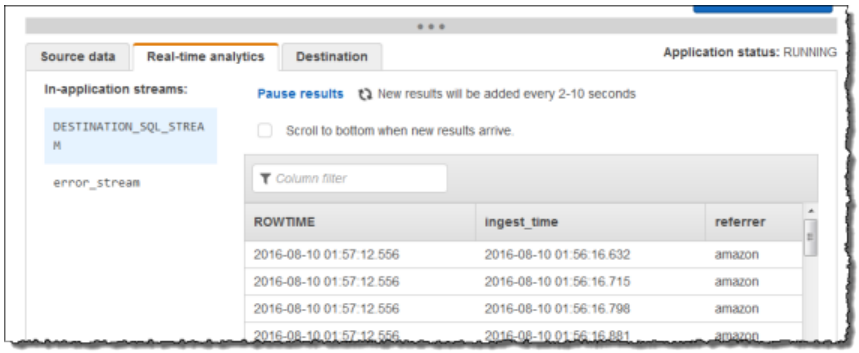
在本示例中，您将以下记录写入 Amazon Kinesis Data Streams。

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

然后，您使用 Kinesis 数据流作为流媒体源，在主机上创建 Amazon Kinesis 数据分析应用程序。发现过程读取有关流式传输源的示例记录，并推断出具有一列 (REFERRER) 的应用程序内部架构，如下所示。



然后，您可以将应用程序代码与 SUBSTRING 函数结合使用，来解析 URL 字符串以检索公司名称。随后将结果数据插入另一个应用程序内部流，如下所示：



主题

- 步骤 1：创建 Kinesis Data Streams (p. 101)
- 步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 102)

步骤 1：创建 Kinesis Data Streams

创建 Amazon Kinesis 数据流并按如下方式填充日志记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建带有一个分片的流。有关更多信息，请参阅 Amazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 运行以下 Python 代码以便填充示例日志记录。这段简单代码不断地将同一日志记录写入到流中。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

接下来，按如下方式创建 Amazon Kinesis 数据分析应用程序：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在上一部分中创建的流。
   b. 选择创建 IAM 角色的选项。
   c. 选择 Discover schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构仅包含一列。
   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
-- 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. 选择 Save and run SQL。在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

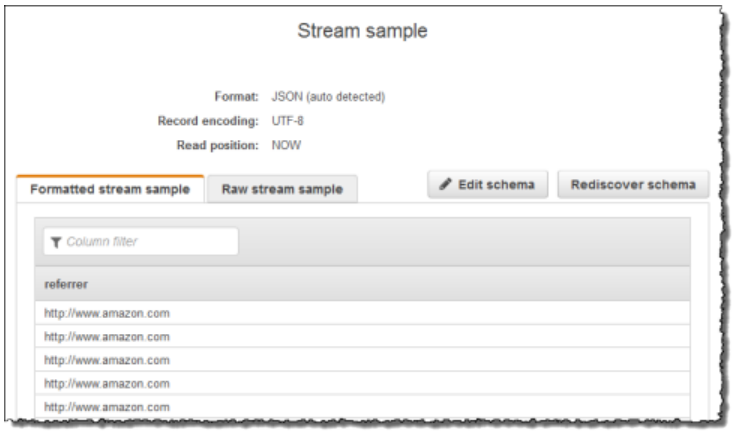
示例：使用正则表达式替换子字符串 (REGEX_REPLACE 函数)

此示例使用REGEX_REPLACE函数在 Amazon Kinesis Data Analytics 中转换字符串。 REGEX_REPLACE用备用子字符串替换子字符串。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 REGEX_REPLAC E。

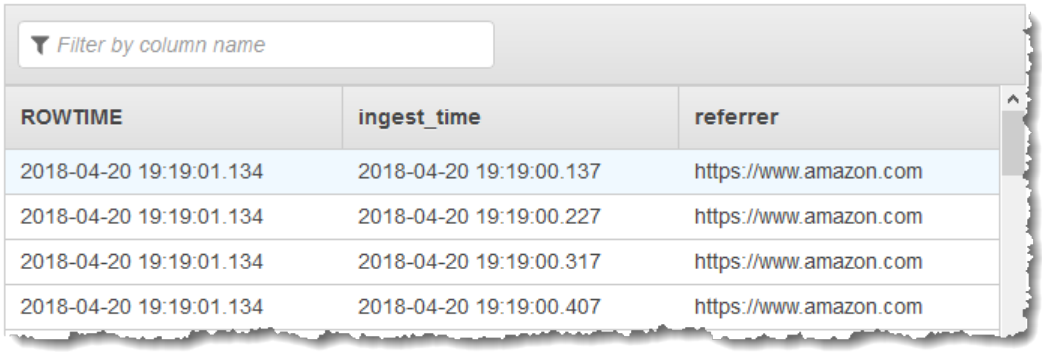
在此示例中，您将以下记录写入 Amazon Kinesis 数据流：

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

然后，您在主机上创建 Amazon Kinesis 数据分析应用程序，将 Kinesis 数据流作为流媒体源。发现过程读取有关流式传输源的示例记录，并推断出具有一列 (REFERRER) 的应用程序内部架构，如下所示。



然后，通过 REGEX_REPLACE 函数使用应用程序代码将 URL 转换为使用 `https://` 而不是 `http://`。随后将结果数据插入另一个应用程序内部流，如下所示：



主题

步骤 1：创建 Kinesis Data Streams

创建 Amazon Kinesis 数据流并按如下方式填充日志记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建带有一个分片的流。有关更多信息，请参阅 Amazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 运行以下 Python 代码以便填充示例日志记录。这段简单代码不断地将同一日志记录写入到流中。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

接下来，按如下方式创建 Amazon Kinesis 数据分析应用程序：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在上一部分中创建的流。
   b. 选择创建 IAM 角色的选项。
   c. 选择 Discover schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构仅包含一列。
   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果，如下所示：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中：

```
-- 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. 选择 Save and run SQL。在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

示例：根据正则表达式分析日志字符串 (REGEX_LOG_PARSE 函数)

此示例使用REGEX_LOG_PARSE函数在 Amazon Kinesis Data Analytics 中转换字符串。REGEX_LOG_PARSE根据默认 Java 正则表达式模式解析字符串。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 REGEX_LOG_PARSE。

在此示例中，您将以下记录写入Amazon Kinesis 直播：

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

然后，您在主机上创建 Amazon Kinesis 数据分析应用程序，将 Kinesis 数据流作为流媒体源。发现过程读取有关流式传输源的示例记录，并推断出具有一列 (LOGENTRY) 的应用程序内部架构，如下所示。

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"

然后，在 REGEX_LOG_PARSE 函数中使用应用程序代码分析日志字符串从而检索数据元素。随后将结果数据插入另一个应用程序内部流，如下面的屏幕截图所示：

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 [

主题

- 步骤 1：创建 Kinesis Data Streams (p. 106)
- 步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 107)

步骤 1：创建 Kinesis Data Streams

创建 Amazon Kinesis 数据流并按如下方式填充日志记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建带有一个分片的流。有关更多信息，请参阅 Amazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 运行以下 Python 代码以便填充示例日志记录。这段简单代码不断地将同一日志记录写入到流中。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

接下来，按如下方式创建 Amazon Kinesis 数据分析应用程序：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application，并指定应用程序名称。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在上一部分中创建的流。
   b. 选择创建 IAM 角色的选项。
   c. 选择 Discover schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构仅包含一列。
   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
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. 选择 Save and run SQL。在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

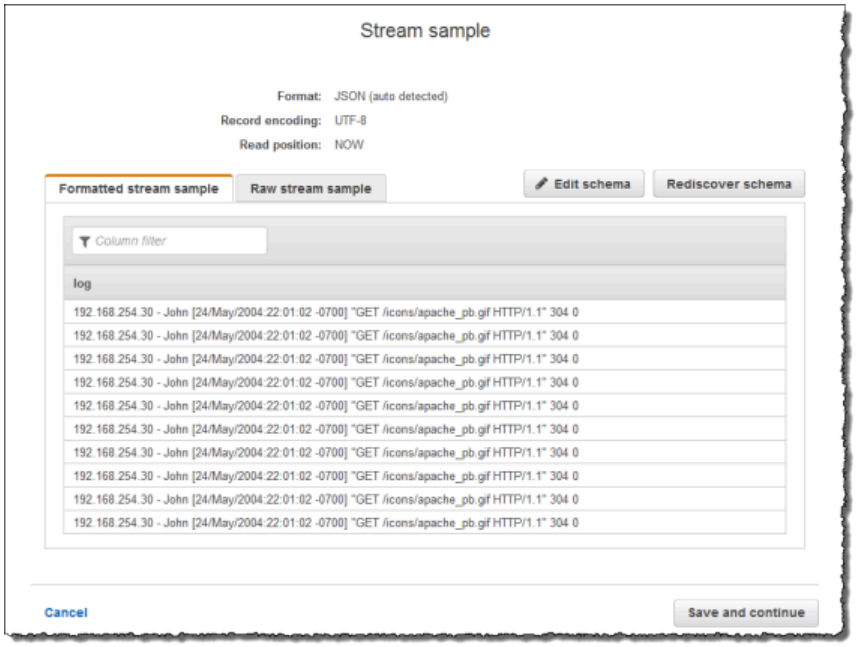
示例：分析 Web 日志 (W3C_LOG_PARSE 函数)

此示例使用W3C_LOG_PARSE函数在 Amazon Kinesis Data Analytics 中转换字符串。您可以使用 W3C_LOG_PARSE 快速格式化 Apache 日志。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 W3C_LOG_PARSE。

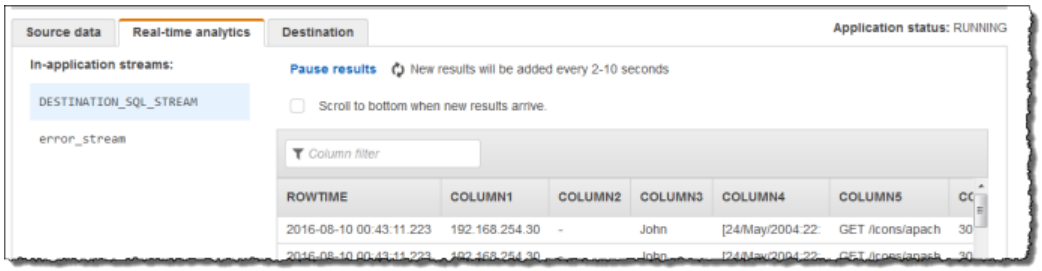
在本示例中，您将日志记录写入 Amazon Kinesis Data Streams。示例日志如下所示：

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

然后，您在主机上创建 Amazon Kinesis 数据分析应用程序，将 Kinesis 数据流作为流媒体源。发现过程读取流式传输源上的示例记录，并推断出具有一个列（日志）的应用程序内部架构，如下所示：



然后，您将使用应用程序代码和 W3C_LOG_PARSE 函数解析日志，创建另一应用程序内部流（将不同日志字段放置在单独的列中），如下所示：



主题

- 步骤 1：创建 Kinesis Data Streams (p. 108)
- 步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 109)

步骤 1：创建 Kinesis Data Streams

创建 Amazon Kinesis 数据流，并按如下方式填充日志记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建带有一个分片的流。有关更多信息，请参阅 Amazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 运行以下 Python 代码以便填充示例日志记录。这段简单代码不断地将同一日志记录写入到流中。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

按如下方式创建 Amazon Kinesis 数据分析应用程序：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在上一部分中创建的流。
   b. 选择创建 IAM 角色的选项。
   c. 选择 Discover schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构仅包含一列。
   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
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. 选择 Save and run SQL。在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

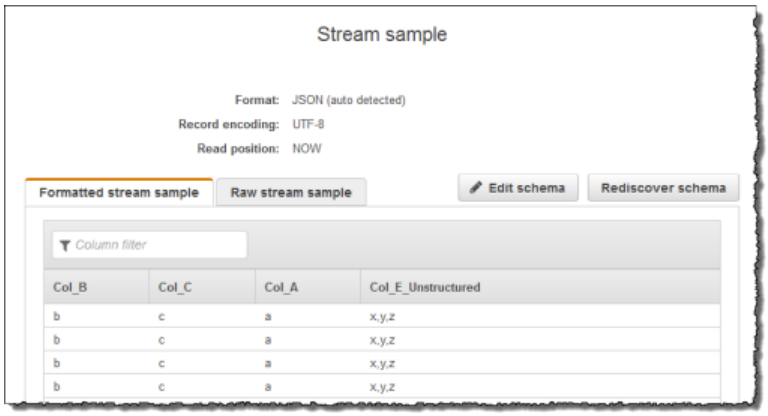
示例：将字符串拆分到多个字段 (VARIABLE_COLUMN_LOG_PARSE 函数)

此示例使用VARIABLE_COLUMN_LOG_PARSE函数在 Kinesis Data Analytics 中操作字符串。VARIABLE_COLUMN_LOG_PARSE将输入字符串拆分成由分隔符或分隔符字符串分隔的字段。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 VARI ABLE_COLUMN_LOG_PARSE。

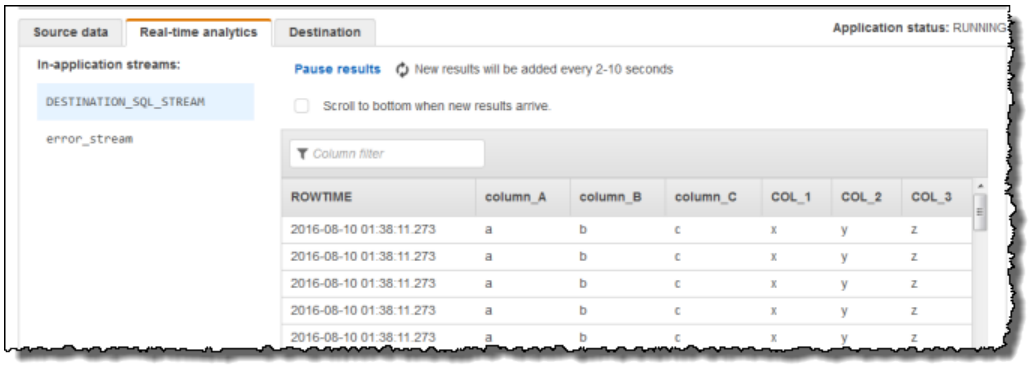
在此示例中，您将半结构化记录写入 Amazon Kinesis 数据流。示例记录如下所示：

```
{ "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"}
```

然后，使用 Kinesis 直播作为流媒体源，在主机上创建 Amazon Kinesis 数据分析应用程序。发现过程读取流式传输源上的示例记录，并推断出具有四个列的应用程序内部架构，如下所示：



然后，您将使用应用程序代码和 VARIABLE_COLUMN_LOG_PARSE 函数解析逗号分隔的值，将规范化的行插入到其他应用程序内部流，如下所示：



主题

- 步骤 1：创建 Kinesis Data Streams (p. 111)
- 步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 112)

步骤 1：创建 Kinesis Data Streams

创建 Amazon Kinesis 数据流并按如下方式填充日志记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建带有一个分片的流。有关更多信息，请参阅 Amazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 运行以下 Python 代码以便填充示例日志记录。这段简单代码不断地将同一日志记录写入到流中。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

按如下方式创建 Amazon Kinesis 数据分析应用程序：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在上一部分中创建的流。
   b. 选择创建 IAM 角色的选项。
   c. 选择 Discover schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。请注意推断的架构仅包含一列。
   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中，编写应用程序代码并确认结果：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中：

```
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. 选择 Save and run SQL。在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

示例：转换 DateTime 值

Amazon Kinesis Data Analytics 支持将列转换为时间戳。例如，除了ROWTIME列之外，您可能还想将自己的时间戳作为GROUP BY子句的一部分用作另一个基于时间的窗口。Kinesis Data Analytics 提供用于处理日期和时间字段的操作和 SQL 函数。

- 日期和时间运算符-您可以对日期、时间和间隔数据类型执行算术运算。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的日期、时间戳和间隔运算符。

- SQL 函数-包括以下内容。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的日期和时间函数。
  - EXTRACT() - 从日期、时间、时间戳或间隔表达式中提取一个字段。
  - CURRENT_TIME— 返回执行查询的时间 (UTC)。
  - CURRENT_DATE— 返回执行查询的日期 (UTC)。
  - CURRENT_TIMESTAMP - 返回执行查询的时间戳 (UTC)。
  - LOCALTIME— 返回查询的当前执行时间，由 Kinesis Data Analytics 运行的环境 (UTC) 定义。
  - LOCALTIMESTAMP— 返回由 Kinesis Data Analytics 运行的环境 (UTC) 定义的当前时间戳。

- SQL 扩展-包括以下内容。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的日期和时间函数以及日期时间转换函数。
  - CURRENT_ROW_TIMESTAMP - 为流中的每一行返回一个新的时间戳。
  - TSDIFF - 返回两个时间戳的差异 (以毫秒为单位)。
  - CHAR_TO_DATE— 将串串串串串串串串串串串串串串串串
  - CHAR_TO_TIME— 将串串串串串串串串串串串串串串串
  - CHAR_TO_TIMESTAMP - 将字符串转换为时间戳。
  - DATE_TO_CHAR— 将日期转换为字符串。
  - TIME_TO_CHAR— 将时间转换为字符串。
  - TIMESTAMP_TO_CHAR - 将时间戳转换为字符串。

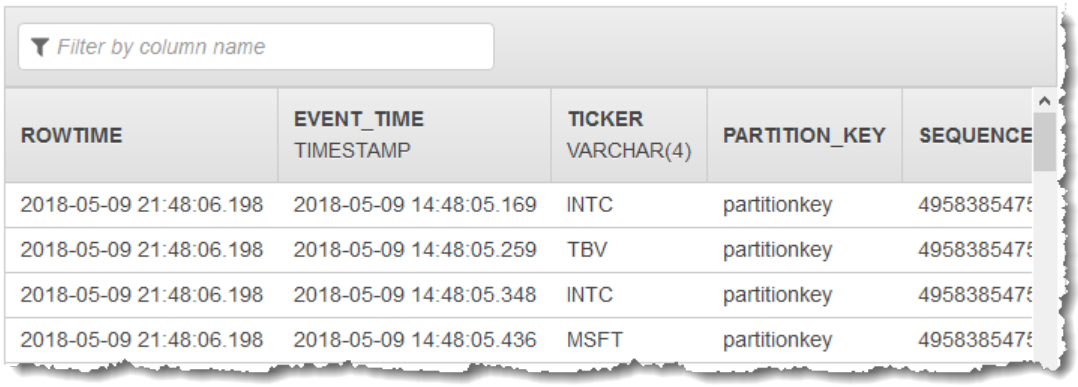
上面大多数 SQL 函数都采用一种格式来转换列。格式是灵活多变的。例如，您可以指定采用格式 yyyy-MM-dd hh:mm:ss 将输入字符串 2009-09-16 03:15:24 转换为时间戳。有关更多信息，请参阅 Amazon Kinesis 数据分析 SQL 参考中的 Char To Timestamp (Sys)。

示例：转换日期

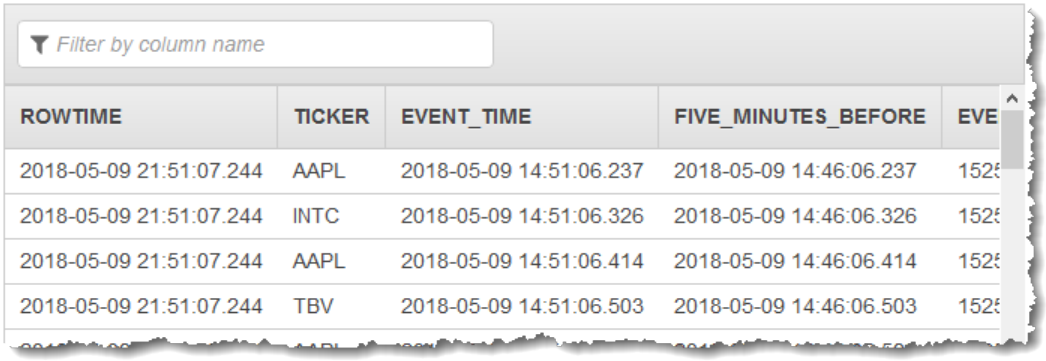
在此示例中，将以下串串串串串串串串串串串串串串串串串串串串串串串串串串串串串串

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

然后，你在主机上创建一个 Amazon Kinesis 数据分析应用程序，将 Kinesis 直播作为流媒体源。发现过程读取流式传输源中的示例记录，并推断出具有两个列 (EVENT_TIME 和 TICKER) 的如下应用程序内部架构。



然后，将该应用程序代码与 SQL 函数结合使用，以多种方式转换 EVENT_TIME 时间戳字段。随后将结果数据插入另一个应用程序内部流，如下面的屏幕截图所示：



步骤 1：创建 Kinesis Data Streams

创建 Amazon Kinesis 数据流并在其中填充事件时间和行情记录，如下所示：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建带有一个分片的流。
4. 运行以下 Python 代码以便用示例数据填充流。此简单代码会不断将具有随机股票代号和当前时间戳的记录写入流中。

```
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'))
```

步骤 2：创建Amazon Kinesis Data Analytics 应用程序

按如下方式创建应用程序：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)，以连接到源。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在上一部分中创建的流。
   b. 选择以创建 IAM 角色。
   c. 选择 Discover schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构有两列。
   d. 选择 Edit Schema (编辑架构)。将 EVENT_TIME 列的 Column type (列类型) 更改为 TIMESTAMP。
   e. 选择 Save schema and update stream samples。在控制台保存架构后，选择 Exit (退出)。
   f. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果，如下所示：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
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. 选择 Save and run SQL。在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

示例：转换多个数据类型

提取、转换和加载 (ETL) 应用程序的共同要求是处理流式传输源中的多种记录。您可以创建 Amazon Kinesis 数据分析应用程序来处理此类流媒体源。过程如下所述：

1. 首先，将流媒体源映射到应用程序内的输入流，与所有其他 Kinesis 数据分析应用程序类似。
2. 然后，在应用程序代码中编写 SQL 语句，从应用程序内部输入流中检索特定类型的行。然后，将这些行插入单独的应用程序内部流。(您可以在应用程序代码中创建其他应用程序内部流)。

在本练习中，您具有一个可接收两种类型 (`Order` 和 `Trade`) 记录的流式传输源。它们分别表示库存订单和相应交易。每批订单可以有零笔或多笔交易。下面显示了每个类型的示例记录：

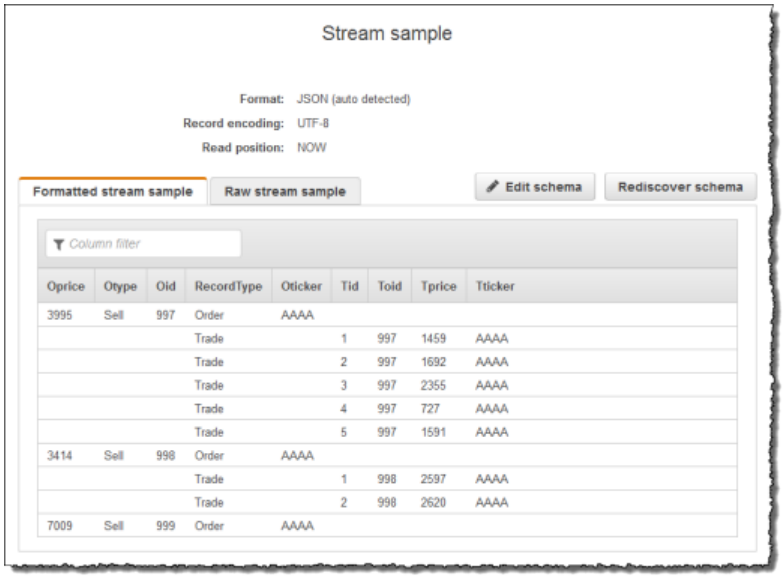
Order record

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

Trade record

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

在使用 Amazon Web Services Management Console创建应用程序时，控制台将为创建的应用程序内部输入流显示以下推断架构。默认情况下，控制台将该应用程序内部流命名为 `SOURCE_SQL_STREAM_001`。



当您保存配置时，Amazon Kinesis Data Analytics 会持续从流媒体源读取数据，并在应用程序内流中插入行。现在，您可以对应用程序内部流中的数据进行分析。

在本示例的应用程序代码中，首先创建两个额外的应用程序内部流：`Order_Stream` 和 `Trade_Stream`。然后，根据记录类型筛选 `SOURCE_SQL_STREAM_001` 流中的行，使用数据泵将这些行插入到新创建的流。有关此编码模式的信息，请参阅应用程序代码 (p. 28)。

1. 将订单和交易行筛选到单独的应用程序内部流:

a. 筛选 SOURCE_SQL_STREAM_001 中的订单记录，并将订单保存到 Order_Stream。

```
--Create Order_Stream.
CREATE OR REPLACE STREAM "Order_Stream"
           (
            order_id     integer,
            order_type   varchar(10),
            ticker       varchar(4),
            order_price  DOUBLE,
            record_type  varchar(10)
            );

CREATE OR REPLACE PUMP "Order_Pump" AS
   INSERT INTO "Order_Stream"
      SELECT STREAM oid, otype,oticker, oprice, recordtype
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  recordtype = 'Order';
```

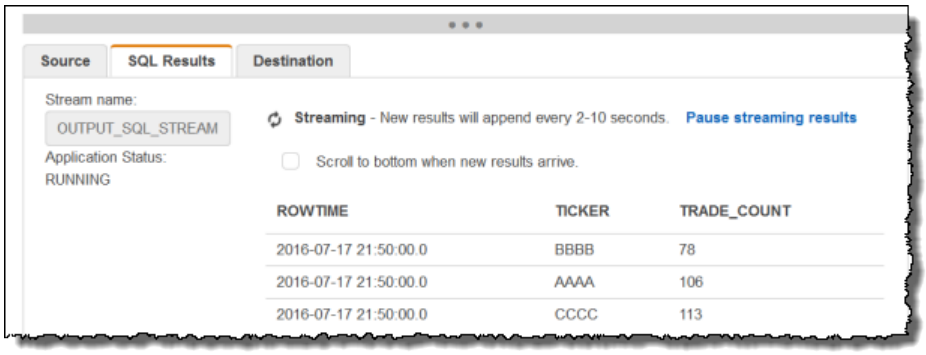
b. 筛选 SOURCE_SQL_STREAM_001 中的交易记录，并将订单保存到 Trade_Stream。

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

CREATE OR REPLACE PUMP "Trade_Pump" AS
   INSERT INTO "Trade_Stream"
      SELECT STREAM tid, toid, tprice, tticker, recordtype
      FROM   "SOURCE_SQL_STREAM_001"
      WHERE  recordtype = 'Trade';
```

2. 现在，您可以对这些流进行其他分析。在本示例中，您将按股票在时长一分钟的滚动窗口中计算交易数，并将结果保存到另一个流 DESTINATION_SQL_STREAM。

```
--do some analytics on the Trade_Stream and Order_Stream.
-- To see results in console you must write to OPUT_SQL_STREAM.

CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
            ticker  varchar(4),
            trade_count   integer
            );

CREATE OR REPLACE PUMP "Output_Pump" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM ticker, count(*) as trade_count
      FROM   "Trade_Stream"
      GROUP BY ticker,
                FLOOR("Trade_Stream".ROWTIME TO MINUTE);
```

此时将显示如下结果：



主题

- 步骤 1：准备数据 (p. 118)
- 步骤 2：创建 应用程序 (p. 120)

下一个步骤

步骤 1：准备数据 (p. 118)

步骤 1：准备数据

在本节中，您将创建 Kinesis 数据流，然后在该数据流上填充订单和交易记录。此式源将用于下一步创建的应用程序。

主题

- 步骤 1.1：创建流式传输源 (p. 118)
- 步骤 1.2：填充流式传输源 (p. 118)

步骤 1.1：创建流式传输源

您可以使用控制台或创建 Kinesis 数据流Amazon CLI。本示例采用 `OrdersAndTradesStream` 作为流名称。

- 使用控制台：https://console.aws.amazon.com/kinesis 登录Amazon Web Services Management Console，打开 Kinesis 控制台：。选择 Data Streams，然后创建带有一个分片的流。有关更多信息，请参阅 Amazon Kinesis Dat a Streams 开发人员指南中的创建流Configuram s。
- 使用Amazon CLI — 使用以下 Kinesis`create-stream`Amazon CLI 命令创建直播：

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

步骤 1.2：填充流式传输源

运行以下 Python 脚本以便在 `OrdersAndTradesStream` 中填充示例记录。如果您使用其他名称创建了流，请相应更新 Python 代码。

1. 安装 Python 和 `pip`。

有关安装 Python 的信息，请访问 Python 网站。

您可以使用 pip 安装依赖项。有关安装 pip 的信息，请参阅 pip 网站上的安装。

2. 运行以下 Python 代码。代码中的 put-record 命令将 JSON 记录写入到流。

```
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'))
```

下一个步骤

步骤 2：创建 应用程序 (p. 120)

步骤 2：创建 应用程序

在本部分中，创建 Amazon Kinesis Data Analytics Analyto 然后，通过添加将您在前一部分中创建的流式传输源映射到应用程序内部输入流的输入配置，您可以更新应用程序。

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application（创建应用程序）。此示例使用应用程序名称 **ProcessMultipleRecordTypes**。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)，以连接到源。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：
   a. 选择在步骤 1：准备数据 (p. 118)中创建的流。
   b. 选择以创建 IAM 角色。
   c. 等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。
   d. 选择 Save and continue。
5. 在应用程序中心上，选择 Go to SQL editor。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果：
   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
--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. 选择 Save and run SQL。选择 Real-time analytics (实时分析) 选项卡可查看应用程序已创建的所有应用程序内部流并验证数据。

下一个步骤

您可以将应用程序输出配置为将结果保存到外部目的地，例如另一个 Kinesis 数据流或 Kinesis Data Firehose 数据传输流。

示例：窗口和聚合

本节提供了使用窗口查询和聚合查询的 Amazon Kinesis Data Analytics 应用程序的示例。（有关更多信息，请参阅 窗口式查询 (p. 67)。）每个示例都提供了设置 Amazon Kinesis 数据分析应用程序的 step-by-step 说明和示例代码。

主题

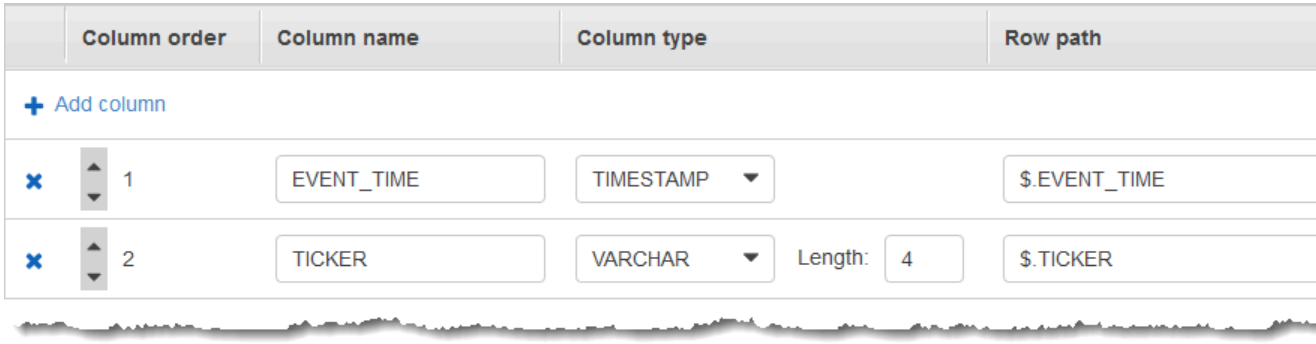
示例：交错窗口

当窗口化查询处理每个唯一分区键的单独窗口时，从具有匹配键的数据到达时开始，该窗口被称为交错窗口。有关详细信息，请参阅交错窗口 (p. 68)。这个 Amazon Kinesis Data Analytics 示例使用 EVENT_TIME 和 TICKER 列来创建交错窗口。源流包含具有相同 EVENT_TIME 和 TICKER 值的六个记录组成的组，这些值在一分钟时间内到达，但不一定具有相同的分钟值（例如 18:41:xx）。

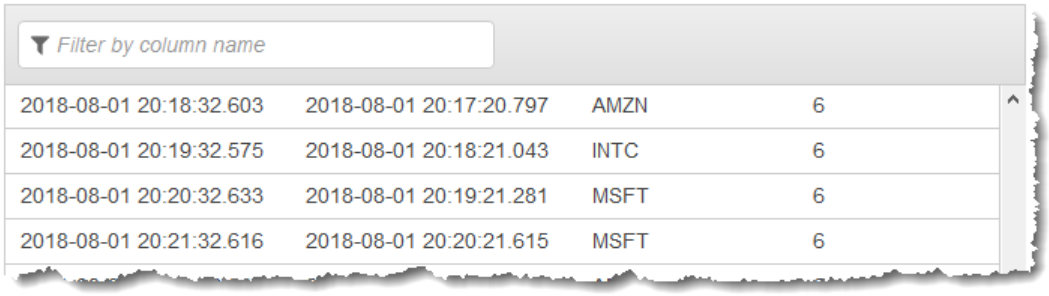
在本示例中，您在以下时间将以下记录写入 Kinesis 数据流。该脚本不会将时间写入流，但应用程序接收记录的时间将写入 ROWTIME 字段：

```
{"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
...
```

然后，您将 Kinesis Data Analytics 应用程序Amazon Web Services Management Console，将 Kinesis Data Analytics 应用程序。发现过程读取流式传输源中的示例记录，并推断出具有两个列（EVENT_TIME 和 TICKER）的如下所示的应用程序内部架构。



您使用应用程序代码以及 COUNT 函数以创建数据的窗口式聚合。然后，将结果数据插入另一个应用程序内部流，如下面的屏幕截图所示：



在以下步骤中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序基于 EVENT_TIME 和 TICKER 在交错窗口中聚合输入流中的值。

主题

- 步骤 1：创建 Kinesis Data 流 (p. 122)
- 步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 123)

步骤 1：创建 Kinesis Data 流

创建 Amazon Kinesis 数据流并按如下方式填充记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建具有一个分片的流。有关更多信息，请参阅 Amamamamazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 要在生产环境中将记录写入到 Kinesis 数据流，我们建议您使用 Kinesis 创建器库或 Kinesis 数据流 API。为简单起见，此示例使用以下 Python 脚本以便生成记录。运行此代码以填充示例股票代码记录。这段简单代码在一分钟时间中连续地将一组六个记录与相同的随机 EVENT_TIME 和股票代码符号一起写入流中。让脚本保持运行，以便可以在后面的步骤中生成应用程序架构。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

创建 Sis Data Analytics 应用程序，如下所示：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)，以连接到源。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：

   a. 选择在上一部分中创建的流。
   b. 选择 Discover Schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构有两列。
   c. 选择 Edit Schema (编辑架构)。将 EVENT_TIME 列的 Column type (列类型) 更改为 TIMESTAMP。
   d. 选择 Save schema and update stream samples。在控制台保存架构后，选择 Exit (退出)。
   e. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：

   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
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. 选择 Save and run SQL。

在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

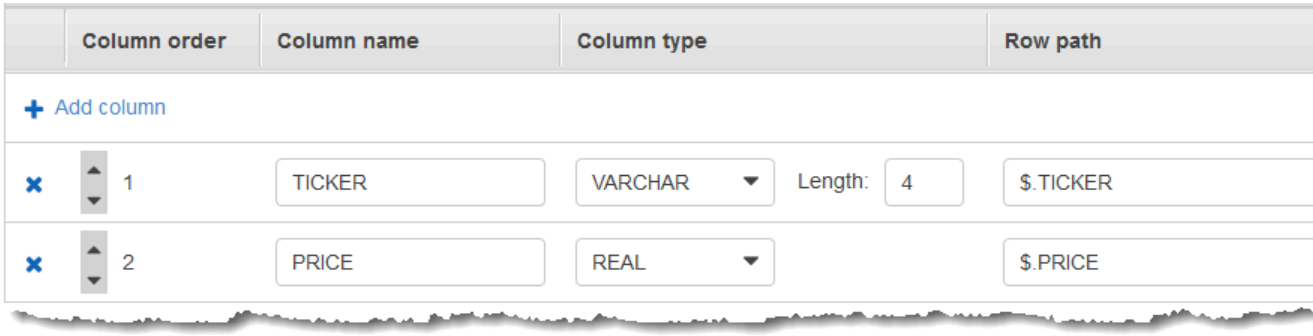
示例：使用 ROWTIME 的滚动窗口

当一个窗口式查询以非重叠方式处理每个窗口时，这样的窗口称为滚动窗口。有关详细信息，请参阅滚动窗口（使用 GROUP BY 组的聚合）(p. 72)。这个 Amazon Kinesis Data Analytics 示例使用该ROWTIME列来创建翻滚窗口。ROWTIME 列表示应用程序读取该记录的时间。

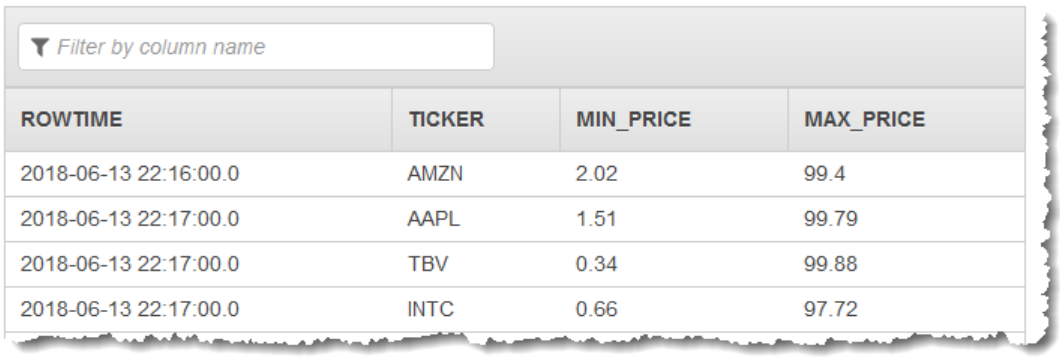
在此示例中，您将以下记录写入 Kinesis 数据流。

```
{"TICKER": "TBV", "PRICE": 33.11}
{"TICKER": "INTC", "PRICE": 62.04}
{"TICKER": "MSFT", "PRICE": 40.97}
{"TICKER": "AMZN", "PRICE": 27.9}
...
```

然后，您将 Kinesis Data Analytics 应用程序Amazon Web Services Management Console，将 Kinesis Data Analytics 应用程序。发现过程读取流式传输源中的示例记录，并推断出具有两个列（TICKER 和 PRICE）的如下所示的应用程序内部架构。



您使用应用程序代码以及 MIN 和 MAX 函数以创建数据的窗口式聚合。然后，将结果数据插入另一个应用程序内部流，如下面的屏幕截图所示：



在以下步骤中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序基于 ROWTIME 在滚动窗口中聚合输入流中的值。

主题

步骤 1：创建 Kinesis Data 流

创建 Amazon Kinesis 数据流并按如下方式填充记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建具有一个分片的流。有关更多信息，请参阅 Amamamamazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 要在生产环境中将记录写入到 Kinesis 数据流，我们建议您使用 Kinesis 客户端库或 Kinesis 数据流 API。为简单起见，此示例使用以下 Python 脚本以便生成记录。运行此代码以填充示例股票代码记录。这段简单代码不断地将随机的股票代码记录写入到流中。让脚本保持运行，以便可以在后面的步骤中生成应用程序架构。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

创建 Sis Data Analytics 应用程序，如下所示：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，输入应用程序名称，然后选择 Create application (创建应用程序)。

3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)，以连接到源。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：

   a. 选择在上一部分中创建的流。
   b. 选择 Discover Schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构有两列。
   c. 选择 Save schema and update stream samples。在控制台保存架构后，选择 Exit (退出)。
   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：

   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
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. 选择 Save and run SQL。

      在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

示例：使用事件时间戳的滚动窗口

当一个窗口式查询以非重叠方式处理每个窗口时，这样的窗口称为滚动窗口。有关详细信息，请参阅滚动窗口（使用 GROUP BY 组的聚合）(p. 72)。这个 Amazon Kinesis Data Analytics 示例演示了一个使用事件时间戳的翻滚窗口，事件时间戳是用户创建的时间戳，包含在流媒体数据中。它使用这种方法，而不仅仅是使用 ROWTIME，后者是 Kinesis Data Analytics 在应用程序收到记录时创建的时间戳。如果您想根据事件发生的时间而非应用程序收到此事件的时间创建一个聚合，则可以在流数据中使用事件时间戳。在本例中，ROWTIME 值每分钟触发一次此聚合，ROWTIME 和所包含事件时间都会聚合记录。

在此示例中，您将以下记录写入 Amazon Kinesis 流。该EVENT_TIME值设置为过去 5 秒，以模拟处理和传输延迟，这可能会造成从事件发生到将记录导入 Kinesis Data Analytics 的延迟。

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

然后，您将 Kinesis Data Analytics 应用程序Amazon Web Services Management Console，将 Kinesis Data Analytics 应用程序。发现过程读取流式传输源中的示例记录，并推断出具有三个列（EVENT_TIME、TICKER 和 PRICE）的如下所示的应用程序内部架构。

	Column order	Column name	Column type		Row path
+ Add column					
✖	1	EVENT_TIME	TIMESTAMP		$.EVENT_TIME
✖	2	TICKER	VARCHAR	Length: 4	$.TICKER
✖	3	PRICE	DECIMAL		$.PRICE

您使用应用程序代码以及 MIN 和 MAX 函数以创建数据的窗口式聚合。然后，将结果数据插入另一个应用程序内部流，如下面的屏幕截图所示：

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

在以下步骤中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序根据事件时间在滚动窗口中聚合输入流中的值。

主题

步骤 1：创建 Kinesis Data 流

创建 Amazon Kinesis 数据流并按如下方式填充记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建具有一个分片的流。有关更多信息，请参阅 Amamamamazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 要在生产环境中将记录写入到 Kinesis 数据流，我们建议您使用 Kinesis 客户端库或 Kinesis 数据流 API。为简单起见，此示例使用以下 Python 脚本以便生成记录。运行此代码以填充示例股票代码记录。这段简单代码不断地将随机的股票代码记录写入到流中。让脚本保持运行，以便可以在后面的步骤中生成应用程序架构。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

创建 Sis Data Analytics 应用程序，如下所示：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，输入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)，以连接到源。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：

   a. 选择在上一部分中创建的流。
   b. 选择 Discover Schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构有三列。
   c. 选择 Edit Schema (编辑架构)。将 EVENT_TIME 列的 Column type (列类型) 更改为 TIMESTAMP。
   d. 选择 Save schema and update stream samples。在控制台保存架构后，选择 Exit (退出)。
   e. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：

   a. 复制下面的应用程序代码并将其粘贴到编辑器中。

```
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. 选择 Save and run SQL。

在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

示例：检索最常出现的值 (TOP_K_ITEMS_TUMBLING)

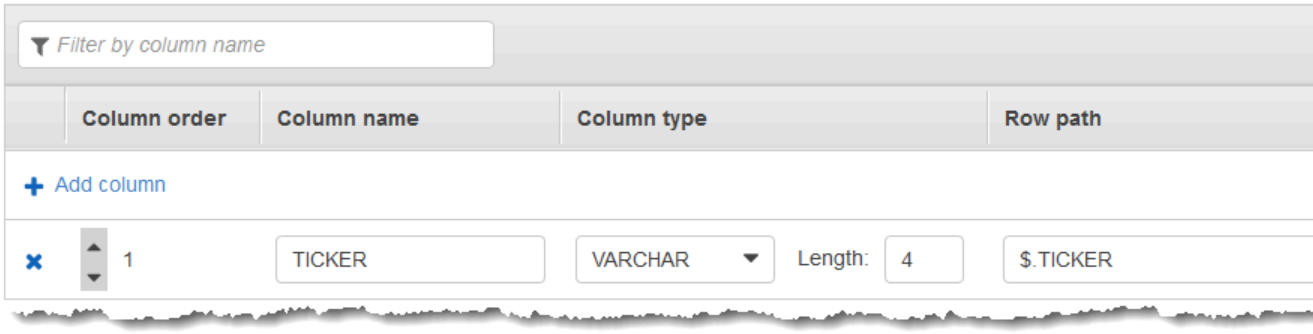
这个 Amazon Kinesis Data Analytics 示例演示了如何使用该TOP_K_ITEMS_TUMBLING函数在翻滚窗口中检索最常出现的值。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的TOP_K_ITEMS_TUMBLING函数。

在对数万或数十万个密钥进行聚合时，如果您希望降低资源占用，则 TOP_K_ITEMS_TUMBLING 函数很有用。该函数生成的结果与使用 GROUP BY 和 ORDER BY 子句进行聚合一样。

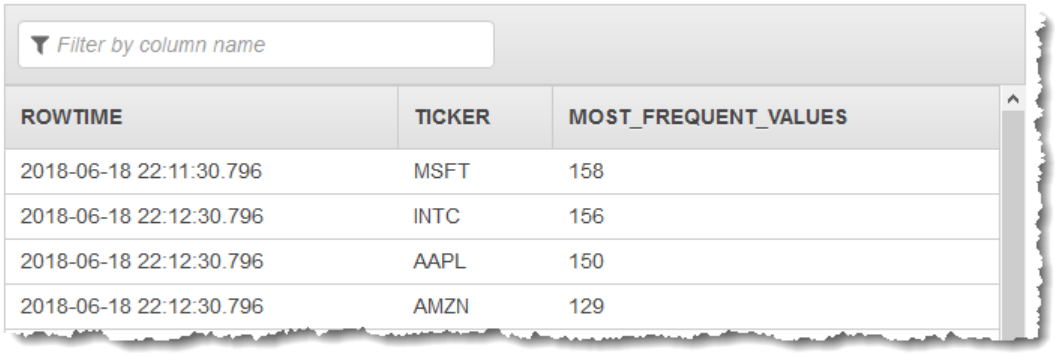
在此示例中，您将以下记录写入 Amazon Kinesis 数据流：

```
{"TICKER": "TBV"}
{"TICKER": "INTC"}
{"TICKER": "MSFT"}
{"TICKER": "AMZN"}
...
```

然后，您将 Kinesis Data Analytics 应用程序Amazon Web Services Management Console，将 Kinesis Data Analytics 应用程序。发现过程读取流式传输源上的示例记录，并推断出具有一个列 (TICKER) 的应用程序内部架构，如下所示：



您使用应用程序代码以及 TOP_K_VALUES_TUMBLING 函数以创建数据的窗口式聚合。然后，将结果数据插入另一个应用程序内部流，如下面的屏幕截图所示：



在以下步骤中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序检索输入流中最常出现的值。

主题

- 步骤 1：创建 Kinesis Data 流 (p. 130)
- 步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 131)

步骤 1：创建 Kinesis Data 流

创建 Amazon Kinesis 数据流并按如下方式填充记录：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Streams (数据流)。
3. 选择 Create Kinesis stream (创建 Kinesis 流)，然后创建具有一个分片的流。有关更多信息，请参阅 Amamamamazon Kinesis Dat a Streams 开发人员指南中的创建流。
4. 要在生产环境中将记录写入到 Kinesis 数据流，我们建议您使用 Kinesis 客户端库或 Kinesis 数据流 API。为简单起见，此示例使用以下 Python 脚本以便生成记录。运行此代码以填充示例股票代码记录。这段简单代码不断地将随机的股票代码记录写入到流中。让脚本保持运行，以便可以在后面的步骤中生成应用程序架构。

```
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'))
```

步骤 2：创建 Kinesis Data Analytics 应用程序

创建 Sis Data Analytics 应用程序，如下所示：

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择 Create application (创建应用程序)，键入应用程序名称，然后选择 Create application (创建应用程序)。
3. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)，以连接到源。
4. 在 Connect to source (连接到源) 页面上，执行以下操作：

   a. 选择在上一部分中创建的流。

   b. 选择 Discover Schema (发现架构)。等待控制台显示推断的架构和为创建的应用程序内部流推断架构所使用的示例记录。推断的架构包含一列。

   c. 选择 Save schema and update stream samples。在控制台保存架构后，选择 Exit (退出)。

   d. 选择 Save and continue。
5. 在应用程序详细信息页面上，选择 Go to SQL editor (转到 SQL编辑器)。要启动应用程序，请在显示的对话框中选择 Yes, start application (是，启动应用程序)。
6. 在 SQL 编辑器中编写应用程序代码并确认结果如下所示：

   a. 复制下面的应用程序代码并将其粘贴到编辑器中：

```
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. 选择 Save and run SQL。

      在 Real-time analytics (实时分析) 选项卡上，可以查看应用程序已创建的所有应用程序内部流并验证数据。

示例：从查询中聚合部分结果

如果 Amazon Kinesis 数据流包含的事件时间与摄取时间不完全匹配的记录，则滚动窗口中的精选结果包含在该窗口内到达但未必发生的记录。在这种情况下，滚动窗口只包含您需要的部分结果集。您可以通过多种方法来纠正这一问题：

- 仅使用滚动窗口，并使用 upsert 通过数据库或数据仓库在后处理中聚合部分结果。这种方法在处理应用程序时很有效。它为聚合运算符（sum、min、max 等）无限期地处理后期数据。这种方法的缺点是，您必须在数据库层开发和维护额外的应用程序逻辑。
- 使用滚动和滑动窗口，这会在早期生成部分结果，还会继续在滑动窗口期间生成完整的结果。此方法使用 overwrite 操作而非 upsert 操作来处理新近数据，这样就不需要在数据库层添加任何其他应用程序逻辑。这种方法的缺点是它使用更多的 Kinesis 处理单元 (KPU)，但仍然会产生两个结果，这可能不适用于某些用例。

有关滚动和滑动窗口的更多信息，请参阅窗口式查询 (p. 67)。

在以下过程中，滚动窗口聚合会生成两个部分结果（发送到 CALC_COUNT_SQL_STREAM 应用程序内部流），它们必须合并以生成最终结果。然后，应用程序生成第二个聚合（发送到 DESTINATION_SQL_STREAM 应用程序内部流），它合并这两个部分结果。

创建使用事件时间聚合部分结果的应用程序

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
2. 在导航窗格中，选择 Data Analytics (数据分析)。按照??? (p. 40)教程中所述创建 Kinesis Data Analytics 应用程序。
3. 在 SQL 编辑器中，将应用程序代码替换为以下内容：

```
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);
```

   应用程序代码中的 SELECT 语句将在 SOURCE_SQL_STREAM_001 中筛选出显示股票价格更改大于 1% 的行，并使用数据泵将这些行插入另一个应用程序内部流 CHANGE_STREAM。
4. 选择 Save and run SQL。

第一个数据泵将流输出到与以下内容类似的 CALC_COUNT_SQL_STREAM。请注意，结果集不完整：

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

然后，第二个泵将流输出到 DESTINATION_SQL_STREAM，其中包含完整的结果集：

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

示例：联接

本部分提供了使用联接查询的 Amazon Kinesis 数据分析应用程序的示例。每个示例都提供了设置和测试 Kinesis 数据分析应用程序的 step-by-step 说明。

主题

- 示例：向 Kinesis Data Analytics 应用程序添加参考数据 (p. 133)

示例：向 Kinesis Data Analytics 应用程序添加参考数据

在本练习中，您将参考数据添加到现有的 Amazon Kinesis 数据分析应用程序。有关引用数据的信息，请参阅以下主题：

- 适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)
- 配置应用程序输入 (p. 4)

在本练习中，您将参考数据添加到在 Kinesis Data Analytics 入门练习中创建的应用程序中。引用数据提供每个股票代号对应的公司名称；例如：

```
Ticker, Company
AMZN,Amazon
```

```
ASD, SomeCompanyA
MMB, SomeCompanyB
WAS,  SomeCompanyC
```

首先，请完成入门练习中的步骤，以创建一个启动应用程序。然后，执行以下步骤进行设置，并将引用数据添加到应用程序：

1. 准备数据
   - 将上述参考数据作为对象存储在 Amazon Simple Storage Service (Amazon S3) 中。
   - 创建一个让 Kinesis Data Analytics 可代入以代表您读取 Amazon S3 对象的 IAM 角色。
2. 将引用数据源添加到您的应用程序。

   Kinesis Data Analytics 读取 Amazon S3 对象并创建应用程序内部参考表，您可以在应用程序代码中查询该参考表。
3. 测试代码。

   在应用程序代码中，将编写一个联接查询来联接应用程序内部流和应用程序内部引用表，从而获取每个股票代码的对应公司名称。

主题

步骤 1：准备

在本节中，您将示例参考数据作为对象存储在 Amazon S3 存储桶中。您还创建可由 Kinesis Data Analytics 代入以代表您读取对象的 IAM 角色。

将参考数据存储为 Amazon S3 对象

在此步骤中，您将示例参考数据存储为 Amazon S3 对象。

1. 打开文本编辑器，添加以下数据，然后将文件另存为 `TickerReference.csv`。

   ```
   Ticker, Company
   AMZN,Amazon
   ASD, SomeCompanyA
   MMB, SomeCompanyB
   WAS,  SomeCompanyC
   ```

2. 将 `TickerReference.csv` 文件上传到 S3 存储桶。有关说明，请参阅 Amazon Simple Storage Service 用户指南中的将对象上传到 Amazon S3。

创建 IAM 角色

接下来，创建一个 Kinesis Data Analytics 可以代入的 IAM 角色并读取 Amazon S3 对象。

1. 在Amazon Identity and Access Management (IAM) 中，创建一个名为的 IAM 角色**`KinesisAnalytics-ReadS3Object`**。要创建角色，请按照 IAM 用户指南中为亚马逊服务创建角色 (Amazon Web Services Management Console) 中的说明进行操作。

   在 IAM 控制台上，指定以下内容：

- 对于 “选择角色类型”，选择Amazon Lambda。创建角色后，您将更改信任策略以允许 Kinesis Data Analytics（不是Amazon Lambda）担任该角色。
- 不要在 Attach Policy 页面上附加任何策略。

2. 更新 IAM 角色策略：

   a. 在 IAM 控制台上，选择您创建的角色。

   b. 在 “信任关系” 选项卡上，更新信任策略以授予 Kinesis Data Analytics 担任该角色的权限。下面显示了信任策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   c. 在 “权限” 选项卡上，附加名为 AmazonS3 的亚马逊托管政策ReadOnlyAccess。这向角色授予对 Amazon S3 对象的读取权限。下面显示了此策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

步骤 2：将引用数据源添加到应用程序配置中

在该步骤中，将引用数据源添加到应用程序配置中。要开始操作，需要以下信息：

- S3 存储桶名称和对象键名称
- IAM 角色的 Amazon Resource ARN ame (

1. 在应用程序主页面中，选择 Connect reference data (连接引用数据)。
2. 在 Connect 参考数据源页面中，选择包含您的参考数据对象的 Amazon S3 存储桶，然后输入该对象的密钥名称。
3. 输入**CompanyName**应用程序内参考表名称。
4. 在 “访问所选资源” 部分，选择 “从 Kinesis Analytics 可以担任的 IAM 角色中选择”，然后选择您在上一节中创建的 KinesisAnalytics-reads3Objec t IAM 角色。
5. 选择 Discover schema (发现架构)。控制台检测到引用数据中的两列。

6. 选择 Save and close。

步骤 3：测试：查询应用程序内部引用表

现在，您可以查询应用程序内部引用表 CompanyName。您可以联接股票价格数据和引用表以使用引用信息扩充应用程序。结果显示公司名称。

1. 用以下内容替换您的应用程序代码。查询会联接应用程序内部输入流和应用程序内部引用表。应用程序代码将结果写入到另一应用程序内部流 DESTINATION_SQL_STREAM。

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ticker_symbol VARCHAR(4), "Company"
 varchar(20), sector VARCHAR(12), change DOUBLE, price DOUBLE);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
   SELECT STREAM ticker_symbol, "c"."Company", sector, change, price
   FROM "SOURCE_SQL_STREAM_001" LEFT JOIN "CompanyName" as "c"
   ON "SOURCE_SQL_STREAM_001".ticker_symbol = "c"."Ticker";
```

2. 验证 SQLResults 选项卡中是否会显示应用程序输出。确保某些行显示公司名称 (示例引用数据不包含所有公司名称)。

示例：机器学习

本节提供了使用机器学习查询的 Amazon Kinesis Data Analytics 应用程序的示例。机器学习查询对数据执行复杂的分析，同时依靠流中数据的历史记录以查找异常模式。这些示例提供了设置和测试 Kinesis 数据分析应用程序的 step-by-step说明。

主题

- 示例：在流中检测数据异常情况 (RANDOM_CUT_FOREST 函数) (p. 136)
- 示例：检测数据异常和获取说明 (RANDOM_CUT_FOREST_WITH_EXPLANATION 函数) (p. 142)
- 示例：检测流上的热点 (HOTSPOTS 函数) (p. 145)

示例：在流中检测数据异常情况 (RANDOM_CUT_FOREST 函数)

Amazon Kinesis Data Analytics 提供了一个函数 (RANDOM_CUT_FOREST)，该函数可以根据数字列中的值为每条记录分配异常分数。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的RANDOM_CUT_FOREST函数。

在本练习中，您将编写应用程序代码以将异常分数分配给应用程序的流式传输源中的记录。要设置应用程序，请执行以下操作：

1. 设置流媒体源-设置 Kinesis 数据流并写入示例heartRate数据，如下所示：

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

此过程提供用于填充流的 Python 脚本。heartRate 值将随机生成，99% 的记录具有的 heartRate 值介于 60 和 100 之间，仅 1% 的记录具有的 heartRate 值介于 150 和 200 之间。因此，heartRate 值介于 150 和 200 之间的记录是异常情况。

2. 配置输入-使用控制台创建 Kinesis Data Analytics 应用程序，并通过将流媒体源映射到应用程序内部流来配置应用程序输入（SOURCE_SQL_STREAM_001）。应用程序启动时，Kinesis Data Analytics 会持续读取流媒体源并将记录插入应用程序内流。
3. 指定应用程序代码 - 示例使用以下应用程序代码：

```
--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;
```

   此代码读取 SOURCE_SQL_STREAM_001 中的行，分配异常分数，并将结果行写入另一个应用程序内部流 (TEMP_STREAM)。随后，应用程序代码将对 TEMP_STREAM 中的记录进行排序，并将结果保存到另一个应用程序内部流 (DESTINATION_SQL_STREAM)。您使用数据泵将流插入到应用程序内部流。有关更多信息，请参阅应用程序内部流和数据泵 (p. 64)：
4. 配置输出-将应用程序输出配置为将中的数据保存DESTINATION_SQL_STREAM到外部目的地，即另一个 Kinesis 数据流。查看分配给每条记录的异常分数并确定哪个分数指示应用程序外部的异常情况 (您需要收到这些异常情况的警报)。您可以使用 Amazon Lambda 函数处理这些异常分数并配置警报。

此练习使用美国东部（弗吉尼亚北部us-east-1）() 来创建这些直播和应用程序。如果您使用任何其他区域，则必须相应地更新代码。

主题

- 步骤 1：准备 (p. 137)
- 步骤 2：创建应用程序 (p. 139)
- 步骤 3：配置应用程序输出 (p. 140)
- 步骤 4：验证输出 (p. 141)

下一个步骤

步骤 1：准备 (p. 137)

步骤 1：准备

为本练习创建 Amazon Kinesis Data Analytics 应用程序，创建 Kinesis Data Analytics 应用程序。将其中一个流配置为应用程序的流媒体源，将另一个流配置为 Kinesis Data Analytics 保存应用程序输出的目的地。

主题

步骤 1.1：创建输入和输出数据流

在本节中，您将创建两个 Kinesis 直播：ExampleInputStream和ExampleOutputStream。您可以使用 Amazon Web Services Management Console或 Amazon CLI 创建这些流。

- 要使用 控制台

  1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
  2. 选择创建数据流。创建带有一个名为 ExampleInputStream 的分片的流。有关更多信息，请参阅 Amazon Kinesis D ata Stre ams 开发人员指南的。
  3. 重复上一步骤以创建带有一个名为 ExampleOutputStream 的分片的流。

- 使用 Amazon CLI

  1. 使用以下 Kinesiscreate-streamAmazon CLI 命令创建第一个数据流 (ExampleInputStream)。

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

  2. 运行同一命令，同时将流名称更改为 ExampleOutputStream。此命令创建应用程序用来写入输出的第二个流。

步骤 1.2：将示例记录写入输入流

在此步骤中，您运行 Python 代码以持续生成示例记录，并将这些记录写入 ExampleInputStream 流。

```
{"heartRate": 60, "rateType":"NORMAL"}
...
{"heartRate": 180, "rateType":"HIGH"}
```

1. 安装 Python 和 pip。

   有关安装 Python 的信息，请访问 Python 网站。

   您可以使用 pip 安装依赖项。有关安装 pip 的信息，请参阅 pip 网站上的安装。

2. 运行以下 Python 代码。代码中的 put-record 命令将 JSON 记录写入到流。

```
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'))
```

下一个步骤

步骤 2：创建应用程序 (p. 139)

步骤 2：创建应用程序

在本节中，创建 Amazon Kinesis Data Analytics 应用程序，创建，如下所示：

- 将应用程序输入配置为使用您在中创建的 Kinesis 数据流the section called “步骤 1：准备” (p. 137)作为流式传输源。
- 在控制台上使用 Anomaly Detection (异常检测) 模板。

创建应用程序

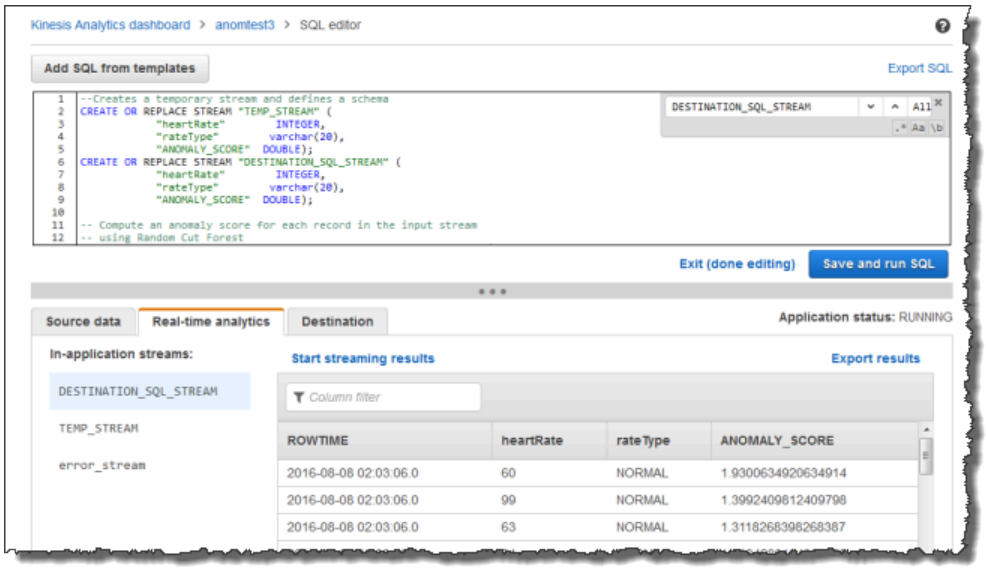
1. 按照 Kinesis Data Analytics 入门练习中的步骤 1、2 和 3 进行操作（参见步骤 3.1：创建应用程序 (p. 45)）。
   - 在源配置中，执行以下操作：
     - 指定您在上一部分中创建的流式传输源。
     - 在控制台推断架构后，编辑架构并将 `heartRate` 列类型设置为 `INTEGER`.

       大多数心率值是正常的，发现过程最有可能将 `TINYINT` 类型分配给此列。但有小部分值显示了高心率。如果这些高值不适合`TINYINT`类型，Kinesis Data Analytics 会将这些行发送到错误流。将数据类型更新为 `INTEGER`，以便能适合所有生成的心率数据。
   - 在控制台上使用 Anomaly Detection (异常检测) 模板。随后，您更新模板代码以提供适当的列名称。
2. 通过提供列名称来更新应用程序代码。下面显示了生成的应用程序代码 (将此代码粘贴到 SQL 编辑器中)：

```
--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. 运行 SQL 代码并在 Kinesis Data Analytics 控制台上查看结果：



下一个步骤

步骤 3：配置应用程序输出 (p. 140)

步骤 3：配置应用程序输出

完成the section called “步骤 2：创建应用程序” (p. 139)后，您已拥有用于从流式传输源读取心率数据并为每条记录分配一个异常分数的应用程序代码。

现在，您可以将应用程序内流中的应用程序结果发送到外部目的地，即另一个 Kinesis 数据流 (OutputStreamTestingAnomalyScores)。您可以分析异常分数并确定哪种心率是异常的。然后，您可以进一步扩展此应用程序以生成警报。

执行以下步骤可配置应用程序输出：

1. 打开 Amazon Kinesis Data Analytics 在 SQL 编辑器中，在应用程序控制面板中选择 Destination 或 Add a destination。
2. 在 Connect to destination (连接到目标) 页面中，选择您在上一部分中创建的 OutputStreamTestingAnomalyScores 流。

   现在，您有了外部目标，Amazon Kinesis Data Analytics 会将您的应用程序写入应用程序内流的所有记录保存到该目的地DESTINATION_SQL_STREAM。
3. 您可以选择配置 Amazon Lambda 以监控 OutputStreamTestingAnomalyScores 流并发送警报。有关说明，请参阅 使用 Lambda 函数预处理数据 (p. 19)。如果您未设置警报，则可以查看 Kinesis Data Analytics 写入外部目的地（即 Kinesis 数据流）的记录OutputStreamTestingAnomalyScores，如中所述步骤 4：验证输出 (p. 141)。

下一个步骤

步骤 4：验证输出 (p. 141)

步骤 4：验证输出

在the section called “步骤 3：配置应用程序输出” (p. 140)中配置应用程序输出后，使用以下 Amazon CLI 命令读取应用程序在目标流中写入的记录。

1. 运行 get-shard-iterator 命令以获取指向输出流中的数据的指针。

```
aws kinesis get-shard-iterator \
--shard-id shardId-000000000000 \
--shard-iterator-type TRIM_HORIZON \
--stream-name OutputStreamTestingAnomalyScores \
--region us-east-1 \
--profile adminuser
```

   您将获得带分片迭代器值的响应，如以下示例响应中所示：

```
  {
      "ShardIterator":
      "shard-iterator-value"   }
```

   复制该分片迭代器值。
2. 运行 get-records命令 Amazon CLI。

```
aws kinesis get-records \
--shard-iterator shared-iterator-value \
--region us-east-1 \
--profile adminuser
```

   此命令将返回一页记录和另一个分片迭代器，您可在后续 get-records 命令中使用该迭代器来提取下一组记录。

示例：检测数据异常和获取说明 (RANDOM_CUT_FOREST_WITH_EXPLANATION 函数)

Amazon Kinesis Data Analytics 提供了该函数，该RANDOM_CUT_FOREST_WITH_EXPLANATION函数根据数字列中的值为每条记录分配异常分数。该函数还能提供异常说明。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 RANDOM_CUT_FOREST_W IT H_EXPLANION。

在本练习中，您将编写应用程序代码，以获取应用程序流式传输源中记录的异常分数。以及获取每个异常的说明。

主题

第一步

步骤 1：准备数据 (p. 142)

步骤 1：准备数据

在为本示例 (p. 142)创建 AAmazon Kinesis Data Analytics 应用程序之前，您需要创建一个 Kinesis 数据流用作应用程序的流媒体源。另外，您还需运行 Python 代码来将模拟血压数据写入流中。

主题

步骤 1.1：创建 Kinesis Data 流

在本节中，您将创建名为的 Kinesis 数据流ExampleInputStream。您可以使用 Amazon Web Services Management Console或 Amazon CLI 创建该数据流。

- 使用控制台：

  1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
  2. 在导航窗格中，选择 Data Streams (数据流)。然后选择创建 Kinesis 流。
  3. 对于名称，请键入 **ExampleInputStream**。对于分片数，请键入 **1**。
- 或者，要使用 Amazon CLI 创建数据流，请运行以下命令：

```
$ aws kinesis create-stream --stream-name ExampleInputStream --shard-count 1
```

步骤 1.2：将示例记录写入输入流

在此步骤中，您运行 Python 代码以不断生成示例记录并将其写入您创建的数据流中。

1. 安装 Python 和 pip。

有关安装 Python 的信息，请参阅 Python。

您可以使用 pip 安装依赖项。有关安装 pip 的信息，请参阅 pip 文档中的安装。

2. 运行以下 Python 代码。可以将区域改为您要用于此示例的区域。代码中的 put-record 命令将 JSON 记录写入到流。

```
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'))
```

下一个步骤

步骤 2：创建分析应用程序 (p. 144)

步骤 2：创建分析应用程序

在本节中，您将创建 AAmazon Kinesis Data Analytics 应用程序并将其配置为使用您创建的 Kinesis 数据流作为中的流媒体源the section called “步骤 1：准备数据” (p. 142)。然后，运行使用 RANDOM_CUT_FOREST_WITH_EXPLANATION 函数的应用程序代码。

创建应用程序

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在导航窗格中选择 Data Analytics (数据分析)，然后选择创建应用程序。
3. 提供应用程序名称和描述 (可选)，并选择 Create application。
4. 选择 “Connect 流数据”，然后ExampleInputStream从列表中选择。
5. 选择 Discover schema，并确保 Systolic 和 Diastolic 显示为 INTEGER 列。如果二者为另一种类型，则选择 Edit schema，并将 INTEGER 类型分配给二者。
6. 在 Real time analytics 下，选择 Go to SQL editor。出现提示时，选择运行您的应用程序。
7. 将以下代码粘贴到 SQL 编辑器中，然后选择 Save and run SQL。

```
--Creates a temporary stream.
CREATE OR REPLACE STREAM "TEMP_STREAM" (
         "Systolic"                 INTEGER,
         "Diastolic"                INTEGER,
         "BloodPressureLevel"       varchar(20),
         "ANOMALY_SCORE"            DOUBLE,
         "ANOMALY_EXPLANATION"      varchar(512));

--Creates another stream for application output.
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
         "Systolic"                 INTEGER,
         "Diastolic"                INTEGER,
         "BloodPressureLevel"       varchar(20),
         "ANOMALY_SCORE"            DOUBLE,
         "ANOMALY_EXPLANATION"      varchar(512));

-- Compute an anomaly score with explanation for each record in the input stream
-- using RANDOM_CUT_FOREST_WITH_EXPLANATION
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "TEMP_STREAM"
      SELECT STREAM "Systolic", "Diastolic", "BloodPressureLevel", ANOMALY_SCORE,
 ANOMALY_EXPLANATION
      FROM TABLE(RANDOM_CUT_FOREST_WITH_EXPLANATION(
               CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001"), 100, 256, 100000,
 1, true));

-- Sort records by descending anomaly score, insert into output stream
CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
      SELECT STREAM * FROM "TEMP_STREAM"
      ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE DESC;
```

下一个步骤

步骤 3：检查结果 (p. 144)

步骤 3：检查结果

当您对此示例 (p. 142)运行 SQL 代码时，首先会看到异常分数等于零的行。这种情况发生在初始学习阶段。然后，您会得到类似如下的结果：

```
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 函数中的算法看到 Systolic (收缩压) 和 Diastolic (舒张压) 列为数字，于是将它们作为输入。
- BloodPressureLevel 列包含文本数据，因此不会被算法所考虑。该列只是一个可视化助手，用来帮助您快速发现本例中的正常、高、低血压水平。
- 在 ANOMALY_SCORE 列中，分数越高的记录越异常。此示例结果集中的第二个记录最异常，异常分数为 3.855851061。
- 要了解算法所考虑的每个数字列在多大程度上造成异常评分，请参阅 ATTRIBUTION_SCORE 列中名为 ANOMALY_SCORE 的 JSON 字段。对于该示例结果集中的第二行，Systolic 和 Diastolic 列造成异常的比例为 1.7447:2.1111。换句话说，异常分数原因的 45% 归咎于收缩压值，55% 归咎于舒张压值。
- 要确定此示例中第二行所代表的点的方向是否异常，请参阅名为 DIRECTION 的 JSON 字段。在本例中，舒张压和收缩压值均标记为 HIGH。要确定这些方向正确的置信度，请参阅名为 STRENGTH 的 JSON 字段。在此示例中，算法更加确信舒张值太高。事实上，舒张压读数的正常值通常为 60—80，而 123 远高于预期。

示例：检测流上的热点 (HOTSPOTS 函数)

Amazon Kinesis Data Analytics 提供了该HOTSPOTS功能，它可以定位和返回有关数据中相对密集区域的信息。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的热点。

在本练习中，您将编写应用程序代码以查找应用程序的流式传输源上的热点。要设置应用程序，请执行以下步骤：

1. 设置流媒体源-设置 Kinesis 直播并写入样本坐标数据，如下所示：

   ```
   {"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
   {"x": 0.722248626528026, "y": 4.648868803193405, "is_hot": "Y"}
   ```

   本示例提供了用于填充流的 Python 脚本。x 和 y 值是随机生成的，一些记录集中在特定位置周围。

   如果脚本有意生成值作为热点的一部分，is_hot 字段将作为指示器提供。这可以帮助您评估热点检测函数是否正常运行。
2. 创建应用程序-然后使用创建 Kinesis 数据分析应用程序。Amazon Web Services Management Console 通过将流式传输源映射到应用程序内部流 (SOURCE_SQL_STREAM_001) 来配置应用程序输入。应用程序启动时，Kinesis Data Analytics 会持续读取流媒体源并将记录插入应用程序内流。

在本练习中，您将为应用程序使用以下代码：

```
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)
    );
```

此代码读取 SOURCE_SQL_STREAM_001 中的行，分析它是否有大量热点，并将生成的数据写入到另一个应用程序内部流 (DESTINATION_SQL_STREAM)。您使用数据泵将流插入到应用程序内部流。有关更多信息，请参阅应用程序内部流和数据泵 (p. 64)：

3. 配置输出-将应用程序输出配置为将数据从应用程序发送到外部目的地，即另一个 Kinesis 数据流。查看热点分数并确定哪些分数表明出现了热点 (并且您需要收到警报)。您可以使用 Amazon Lambda 函数进一步处理热点信息并配置警报。
4. 验证输出-该示例包括一个从输出流读取数据并以图形方式显示的 JavaScript 应用程序，因此您可以实时查看该应用程序生成的热点。

该练习使用美国西部（俄勒冈）(us-west-2) 来创建这些直播和您的应用程序。如果您使用任何其他区域，请相应地更新代码。

主题

步骤 1：创建输入和输出流

为热点示例 (p. 145)创建 Amazon Kinesis Data Analytics 应用程序，创建 Kinesis Data Analytics 应用程序。将其中一个流配置为应用程序的流媒体源，将另一个流配置为 Kinesis Data Analytics 保存应用程序输出的目的地。

主题

步骤 1.1：创建 Kinesis Data Streams

在本节中，您将创建两个 Kinesis 数据流：ExampleInputStream和ExampleOutputStream。

使用控制台或 Amazon CLI 创建这些数据流。

- 使用控制台创建数据流：

  1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Kinesis 控制台：https://console.aws.amazon.com/kinesisvideo/home。
  2. 在导航窗格中，选择 Data Streams (数据流)。
  3. 选择创建 Kinesis 流，然后创建带有一个名为 ExampleInputStream 的分片的流。
  4. 重复上一步骤以创建带有一个名为 ExampleOutputStream 的分片的流。

- 要使用 Amazon CLI 创建数据流，请执行以下操作：

  - 使用以下 Kinesiscreate-streamAmazon CLI 命令创建直播（ExampleInputStream和ExampleOutputStream）。要创建另一个流 (应用程序将用于写入输出)，请运行同一命令以将流名称更改为 ExampleOutputStream。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser

$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

步骤 1.2：将示例记录写入输入流

在此步骤中，您运行 Python 代码以持续生成示例记录并将其写入 ExampleInputStream 流。

```
{"x": 7.921782426109737, "y": 8.746265312709893, "is_hot": "N"}
{"x": 0.722248626580026, "y": 4.648868803193405, "is_hot": "Y"}
```

1. 安装 Python 和 pip。

   有关安装 Python 的信息，请访问 Python 网站。

   您可以使用 pip 安装依赖项。有关安装 pip 的信息，请参阅 pip 网站上的安装。

2. 运行以下 Python 代码。此代码将执行以下操作：

   - 在 (X, Y) 平面上的某个位置生成潜在热点。
   - 为每个热点生成一系列点 (1000 个)。这些点中有 20% 集中在热点周围。其余的点在整个空间内随机生成。
   - put-record 命令将 JSON 记录写入到流。

     Important

     不要将此文件上传到 Web 服务器，因为它包含您的Amazon证书。

```
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'))
```

下一个步骤

步骤 2：创建 Kinesis Data Analytics 应用程序 (p. 148)

步骤 2：创建 Kinesis Data Analytics 应用程序

在热点示例 (p. 145)的这一部分中，您可以按如下方式创建 Amazon Kinesis 数据分析应用程序：

- 将应用程序输入配置为使用您在步骤 1 (p. 146) 中创建的 Kinesis 数据流作为流媒体源。

- 在 Amazon Web Services Management Console中使用提供的应用程序代码。

创建应用程序

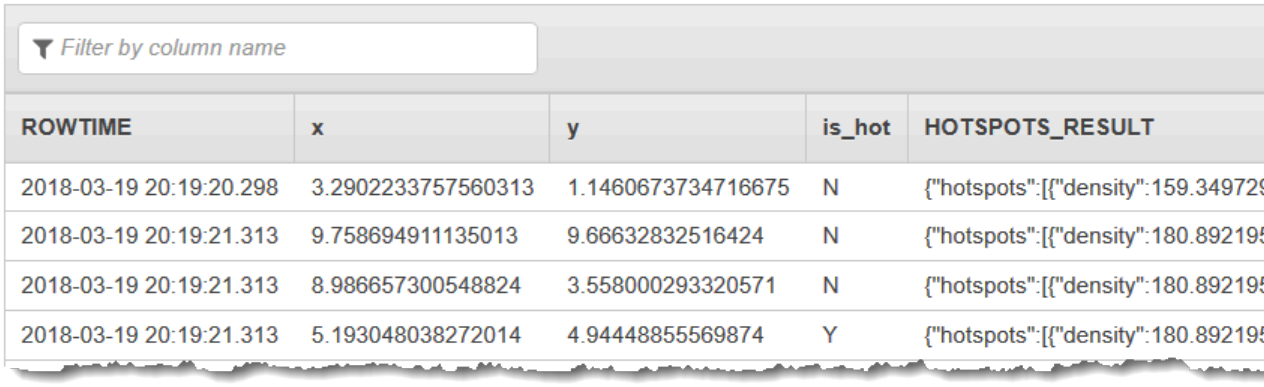
1. 按照入门练习中的步骤 1、2 和 3 创建 Kinesis 数据分析应用程序（参见步骤 3.1：创建应用程序 (p. 45)）。

   在源配置中，执行以下操作：

   - 指定您在the section called “步骤 1：创建流” (p. 146)中创建的流式传输源。
   - 在控制台推断架构后编辑架构。确保 x 和 yDOUBLE 列类型设置为 ，并确保 IS_HOT 列类型设置为 VARCHAR.
2. 使用以下应用程序代码 (您可以将此代码粘贴到 SQL 编辑器中)：

```
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. 运行 SQL 代码并审查结果。



下一个步骤

步骤 3：配置应用程序输出 (p. 149)

步骤 3：配置应用程序输出

此时，在热点示例 (p. 145)中，您有 Amazon Kinesis Data Analytics 应用程序代码从流媒体源中发现重要的热点，并为每个热点分配热点分数。

现在，您可以将应用程序内流中的应用程序结果发送到外部目的地，即另一个 Kinesis 数据流 (ExampleOutputStream)。然后，您可以分析热点分数并为热点热确定合适的阈值。您可以进一步扩展此应用程序以生成警报。

配置应用程序输出

1. 通过 https://console.aws.amazon.com/kinesisanalytics 打开 Kinesis Data Analytics 控制台。
2. 在 SQL 编辑器中，在应用程序控制面板中选择 Destination 或 Add a destination。
3. 在 Add a destination (添加目标) 页面上，选择 Select from your streams (从流中选择)。然后选择在上一部分中创建的 ExampleOutputStream 流。

   现在，您有了外部目标，Amazon Kinesis Data Analytics 可以在其中保存所有记录，您的应用程序会写入应用程序内流DESTINATION_SQL_STREAM。
4. 您可以选择配置 Amazon Lambda 以监控 ExampleOutputStream 流并发送警报。有关更多信息，请参阅使用 Lambda 函数作为输出 (p. 31)：您还可以查看 Kinesis Data Analytics 写入外部目的地（即 Kinesis 流）的记录ExampleOutputStream，如中所述步骤 4：验证应用程序输出 (p. 150)。

下一个步骤

步骤 4：验证应用程序输出 (p. 150)

步骤 4：验证应用程序输出

在热点示例 (p. 145)的此部分中，您将设置一个 Web 应用程序，该应用程序在可扩展矢量图形 (SVG) 控件中显示热点信息。

1. 使用以下内容创建名为 index.html 的文件：

```
<!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>
```

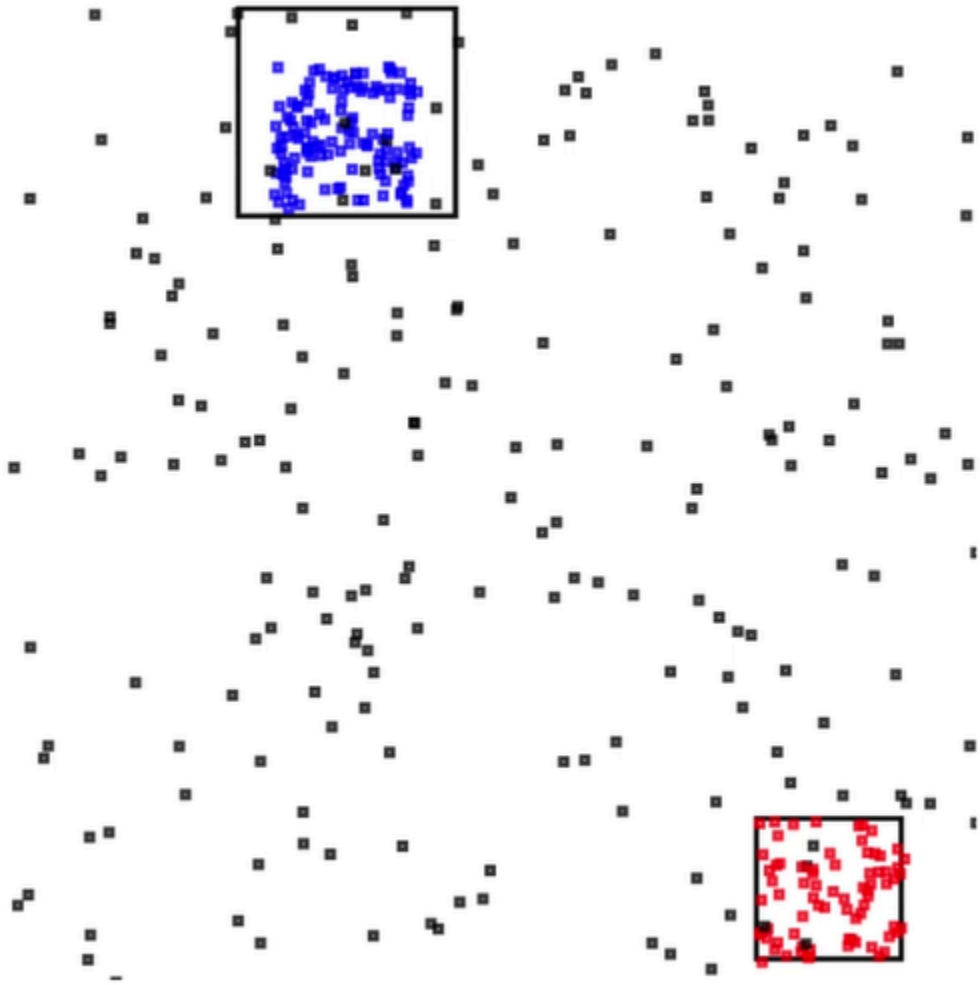
2. 在同一目录中创建一个名为 hotspots_viewer.js 的文件，该文件包含以下内容。在提供的变量中提供您的、证书和输出流名称。

```
// 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 Amazon 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 Amazon 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. 在第一个部分中 Python 代码运行时，在 Web 浏览器中打开 `index.html`。热点信息显示在页面上，如下所示。



示例：警报和错误

本部分提供了使用警报和错误的 Amazon Kinesis 数据分析应用程序的示例。每个示例均提供 step-by-step 说明和代码，以帮助您设置和测试 Kinesis 数据分析应用程序。

主题

示例：创建简单警报

在这个 Amazon Kinesis 数据分析应用程序中，查询在通过演示流创建的应用程序内流上持续运行。有关更多信息，请参阅连续查询 (p. 67)：

如果任何行显示股票价格变动大于 1%，这些行将被插入另一个应用程序内部流中。在本练习中，可以将应用程序输出配置为将结果保存到外部目标。随后，可以进一步调查结果。例如，您可以使用 Amazon Lambda 函数处理记录和发送警报。

创建简单的警报应用程序

1. 按照 Kinesis Data Analytics 入门练习中的说明创建分析应用程序。
2. 在 Kinesis Data Analytics 的 SQL 编辑器中，将应用程序代码替换为以下代码：

```
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;
```

   应用程序代码中的 SELECT 语句可在 SOURCE_SQL_STREAM_001 中筛选出股票价格变化大于 1% 的行。之后，该代码将使用数据泵将这些行插入到另一个应用程序内部流 DESTINATION_SQL_STREAM。有关说明使用数据泵将行插入应用程序内部流中的编码模式的更多信息，请参阅应用程序代码 (p. 28)。
3. 选择 Save and run SQL。
4. 添加一个目标。为此，请在 SQL 编辑器中选择 Destination (目标)，也可以在应用程序中心中选择 Add a destination (添加目标)。

   a. 在 SQL 编辑器中，选择 Destination (目标) 选项卡，然后选择 Connect to a destination (连接到目标)。

      在 Connect to destination (连接到目标) 页面中，选择 Create New (新建)。

   b. 选择 Go to Kinesis Streams。

   c. 在 Amazon Kinesis Data Streams 控制台上，用一个分片创建一个新的 Kinesis 直播（例如gs-destination）。请等待，直到流状态为 ACTIVE。

   d. 返回 Kinesis Data Analytics 控制台。在 Connect to destination (连接到目标) 页面上，选择您创建的流。

      如果流未显示，请刷新页面。

   e. 选择 Save and continue。

   现在，您有了外部目标，即 Kinesis 数据流，Kinesis Data Analytics 将应用程序输出保存在DESTINATION_SQL_STREAM应用程序内部流中。
5. 配置Amazon Lambda为监控您创建的 Kinesis 直播并调用 Lambda 函数。

   有关说明，请参阅 使用 Lambda 函数预处理数据 (p. 19)。

示例：创建受限警报

在这个 Amazon Kinesis 数据分析应用程序中，查询在通过演示流创建的应用程序内流上持续运行。有关更多信息，请参阅连续查询 (p. 67)：如果任何行显示股票价格变动大于 1%，这些行将被插入另一个应用程序内部流中。应用程序会限制警报，以便在股票价格变化时立即发送警报。但每个股票代号每分钟向应用程序内部流发送的警报不超过一个。

创建受限警报应用程序

1. 按照 Kinesis 数据分析入门练习中的说明创建 Kinesis Data Analytics 应用程序。
2. 在 Kinesis Data Analytics 的 SQL 编辑器中，将应用程序代码替换为以下代码：

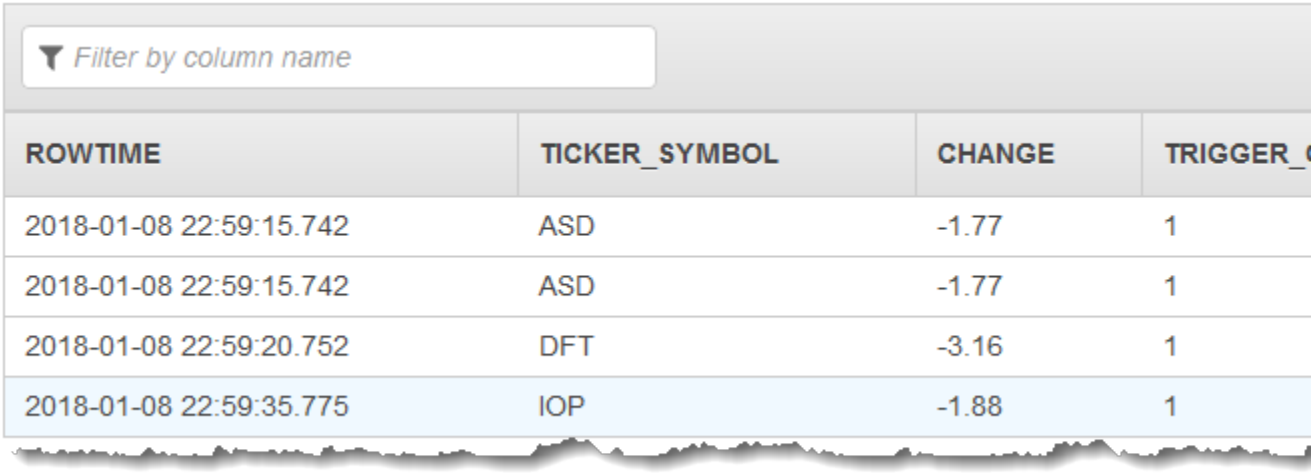
```
CREATE OR REPLACE STREAM "CHANGE_STREAM"
           (ticker_symbol VARCHAR(4),
            sector        VARCHAR(12),
            change        DOUBLE,
            price         DOUBLE);

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;
```

   应用程序代码中的 SELECT 语句将在 SOURCE_SQL_STREAM_001 中筛选出显示股票价格更改大于 1% 的行，并使用数据泵将这些行插入另一个应用程序内部流 CHANGE_STREAM。

   然后，应用程序为受限警报创建第二个名为 TRIGGER_COUNT_STREAM 的流。第二个查询从一个窗口中选择记录，每次记录被允许进入该窗口时，该窗口都向前跳，以便每个股票报价每分钟只有一个记录被写入到流中。

3. 选择 Save and run SQL。

该示例将流输出到与以下内容类似的 TRIGGER_COUNT_STREAM：



示例：探索应用程序内部错误流

Amazon Kinesis Data Analytics 为您创建的每个应用程序提供应用程序内错误流。应用程序无法处理的所有行将发送到此错误流。您可以考虑将错误流数据保存到外部目标以便于调查。

您将在控制台中执行以下练习。在这些示例中，您将通过编辑由发现过程推断的架构将错误引入输入配置中，然后验证已发送到错误流的行。

主题

- 引入分析错误 (p. 157)
- 引入被零除错误 (p. 158)

引入分析错误

在本练习中，您引入了分析错误。

1. 按照 Kinesis 数据分析入门练习中的说明创建 Kinesis Data Analytics 应用程序。
2. 在应用程序详细信息页面上，选择 Connect streaming data (连接流数据)。
3. 如果已完成入门练习，则账户中会有一个演示流 (`kinesis-analytics-demo-stream`)。在 Connect to source (连接到源) 页面上，选择此演示流。
4. Kinesis Data Analytics 从演示流中提取样本，以推断其创建的应用程序内输入流的架构。控制台在 Formatted stream sample 选项卡中显示推断的架构和示例数据。
5. 接下来，可以编辑架构并修改列类型以引入分析错误。选择 Edit schema。
6. 将 `TICKER_SYMBOL` 列类型从 `VARCHAR(4)` 更改为 `INTEGER`。

   现在创建的应用程序内架构的列类型无效，Kinesis Data Analytics 无法在应用程序内流中导入数据。而只能将行发送到错误流。
7. 选择 Save schema。
8. 选择 Refresh schema samples。

   请注意，Formatted stream 示例中没有行。不过，Error stream 选项卡将显示数据与错误消息。Error stream 选项卡将显示已发送到应用程序内部错误流的数据。

   由于您更改了列数据类型，Kinesis Data Analytics 无法将数据导入应用程序内输入流。而只能将数据发送到错误流。

引入被零除错误

在本练习中，您将更新应用程序代码以引入运行时错误 (被零除)。请注意，Amazon Kinesis Data Analytics 将生成的行发送到DESTINATION_SQL_STREAM应用程序内错误流，而不是发送到应该写入结果的应用程序内流。

1. 按照 Kinesis 数据分析入门练习中的说明创建 Kinesis Data Analytics 应用程序。

   验证 Real-time analytics 选项卡上的结果，如下所示：

   Sour
2. 在应用程序代码中更新 SELECT 语句以引入被零除；例如：

```
SELECT STREAM ticker_symbol, sector, change, (price / 0) as ProblemColumn
FROM "SOURCE_SQL_STREAM_001"
WHERE sector SIMILAR TO '%TECH%';
```

3. 运行应用程序。

   由于会出现除以零的运行时错误，因此 Kinesis Data Analytics 不会将结果写入到，而是将行发送到应用程序内错误流。DESTINATION_SQL_STREAM在 Real-time analytics (实时分析) 选项卡上，选择错误流，随后应用程序内部错误流中将显示行。

示例：解决方案加速器

Amazon解决方案网站有可用的Amazon CloudFormation模板，您可以使用这些模板来快速创建完整的流媒体数据解决方案。

可用模板如下：

对Amazon Web Services 账户活动的实时见解

该解决方案实时记录并可视化您的Amazon Web Services 账户（们）的资源访问和使用指标。有关Amazon Web Services 账户有关有关有关

使用 Kinesis Data Analytics 进行实时Amazon IoT设备监控

此解决方案实时收集、处理、分析和直观显示 IoT 设备连接和活动数据。有关更多信息，请参阅使用 Kinesis Data Analytics 进行实时Amazon IoT设备监控。

使用 Kinesis Data Analytics

此解决方案实时收集、处理、分析和直观显示网站点击流数据。有关更多信息，请参阅使用 Kinesis Data Analytics 进行实时 Web 分析。

亚马逊联网汽车解决方案

此解决方案实时收集、处理、分析和直观显示车辆中的 IoT 数据。有关

Amazon Kinesis Data Analytics

Amazon 十分重视云安全性。作为 Amazon 客户，您将会从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Kinesis Data Analytics 的合规性计划，请参阅合规性计划范围内的Amazon服务。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Kinesis Data Analytics 时应用责任共担模型。以下主题说明如何配置 Kinesis Data Analytics 以实现您的安全性和合规性目标。您还将了解如何使用其他服务来帮助您监控和保护您的 Kinesis Data Analytics 资源。

主题

Amazon Kinesis Data Analytics 中的数据保护

您可以使用提供的工具保护您的数据Amazon。Kinesis Data Analytics 可以与支持加密数据的服务一起使用，包括 Kinesis Data Streams、Kinesis Data Firehose 和 Amazon S3。

Kinesis Data Analytics

静态加密

请注意以下有关使用 Kinesis Data Analytics 加密静态数据的注意事项：

- 您可以使用对传入的 Kinesis 数据流上的数据进行加密StartStreamEncryption。有关更多信息，请参阅什么是 Kinesis 数据流的服务器端加密？。
- 可以使用 Kinesis Data Firehose 对输出数据进行静态加密，将数据存储在加密的 Amazon S3 存储桶中。您可以指定 Amazon S3 存储桶使用的加密密钥。有关更多信息，请参阅使用具有 KMS 托管密钥的服务器端加密 (SSE-KMS) 保护数据。
- 您的应用程序的代码是静态加密的。

- 您的应用程序的参考数据是静态加密的。

传输中加密

Kinesis Data Analytics 会加密传输中的所有数据。所有 Kinesis Data Analytics 应用程序均启用传输中加密，无法禁用。

Kinesis Data Analytics 会在以下情况下加密传输中的数据：

- 从 Kinesis Data Streamt 传输到 Kinesis Data Analytics
- 在 Kinesis Data Analytics 内部组件之间传输的数据。
- 在 Kinesis Data Analytics 和 Kinesis Data Firehose 之间传输的数据。

密钥管理

Kinesis Data Analytics 中的数据加密使用服务管理密钥。客户管理的密钥不受支持。

Kinesis Data Analytics 中的Identity and Access Management

Amazon Kinesis Data Analytics 需要权限才能从您在应用程序输入配置中指定的流媒体源读取记录。Amazon Kinesis Data Analytics 还需要权限才能将应用程序输出写入您在应用程序输出配置中指定的流。

您可以通过创建 AAmazon Kinesis Data Analytics 可以代入的 IAM 角色来授予这些权限。您授予此角色的权限决定了服务担任该角色时 Amazon Kinesis Data Analytics 可以做什么。

> **Note**
>
> 如果您要自己创建 IAM 角色，本节中的信息将很有用。当您在 Amazon Kinesis Data Analytics 控制台中创建应用程序时，控制台当时可以为您创建 IAM 角色。控制台对其创建的 IAM 角色使用以下命名约定：
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> 创建角色后，您可以在 IAM 控制台中查看该角色和附加的策略。

每个 IAM 角色都有两个附加的策略。在信任策略中，您可以指定谁可以代入该角色。在权限策略（可以有一个或多个）中，您应当指定要向此角色授予的权限。以下部分描述了这些策略，您可以在创建 IAM 角色时使用这些策略。

信任策略

要授予 Amazon Kinesis Data Analytics 代入角色访问流媒体或参考源的权限，您可以将以下信任策略附加到 IAM 角色：

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

权限策略

如果您正在创建 IAM 角色以允许 Amazon Kinesis Data Analytics 从应用程序的流媒体源读取数据，则必须授予相关读取操作的权限。根据您的来源（例如，Kinesis 直播、Kinesis Data Firehose 传输流或 Amazon S3 存储桶中的参考来源），您可以附加以下权限策略。

读取 Kinesis 直播的权限政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/inputStreamName"
            ]
        }
    ]
}
```

读取 Kinesis Data Firehose 传输流的权限策略

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:Get*"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-
id:deliverystream/inputFirehoseName"
            ]
        }
    ]
}
```

Note

权`firehose:Get*`限是指 Kinesis Data Analytics 用来访问直播的内部访问器。Kinesis Data Firehose 传输流没有公共访问器。

如果您指示 Amazon Kinesis Data Analytics 将输出写入应用程序输出配置中的外部目标，则需要向 IAM 角色授予以下权限。

写入 Kinesis 直播的权限政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/output-stream-name"
            ]
        }
    ]
}
```

写入到 Firehose 传输流的权限策略

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-id:deliverystream/output-firehose-
name"
            ]
        }
    ]
}
```

从 Amazon S3 存储桶读取参考数据源的权限策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

防止跨服务混淆代理

在中Amazon，一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。即使该服务不应有适当的权限，也可以操纵调用服务来对另一个客户的资源进行操作，从而导致混淆代理问题。

为防止混淆代理人，Amazon提供可帮助您保护所有服务的服务委托人数据的工具。本节重点介绍专门针对Kinesis Data Analytics的跨服务混乱副手预防措施，但是，您可以在 IAM 用户指南的 “困惑的副手问题” 部分了解有关此主题的更多信息。

在 Kinesis Data Analytics for SQL 的上下文中，我们建议在您的角色信任策略中使用 a ws:SourceArn 和 aws:SourceAccount 全局条件上下文密钥，以将对角色的访问限制为仅由预期资源生成的请求。

如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

的值`aws:SourceArn`必须是 Kinesis Data Analytics 使用的资源的 ARN，使用以下格式指`arn:aws:kinesisanalytics:region:account:resource`定：

解决混淆代理问题的推荐方法是使用具有完整资源 ARN 的`aws:SourceArn`全局条件上下文密钥。

如果您不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的`aws:SourceArn`密钥。例如：`arn:aws:kinesisanalytics::111122223333:*`。

虽然 Kinesis Data Analytics for SQL API 中的大多数操作（例如CreateApplicationAddApplicationInput和）DeleteApplication都是在特定应用程序的上下文中执行的，但该DiscoverInputSchema操作不会在任何应用程序的上下文中执行。这意味着此操作中使用的角色不得在`SourceArn`条件键中完全指定资源。以下是使用通配符 ARN 的示例：

```
{
   ...
   "ArnLike":{
      "aws:SourceArn":"arn:aws:kinesisanalytics:us-east-1:123456789012:*"
   }
   ...
}
```

适用于 SQL 的 Kinesis Data Analytics 生成的默认角色使用此通配符。这样可以确保在控制台体验中无缝发现输入架构。但是，我们建议在发现架构后编辑信任策略以使用完整的 ARN，以实现完全混乱的副手缓解措施。

您向 Kinesis Data Analytics 提供的角色策略以及为您生成的角色的信任策略可以使用 a ws:SourceArn 和 a ws:SourceAccount 条件密钥。

为了防止出现混乱的副手问题，请执行以下步骤：

防代理问题

1. 登录 Amazon 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择 Roles (角色)，然后选择您要修改的角色。

3. 选择 Edit trust policy（编辑信任策略）。
4. 在编辑信任策略页面上，将默认 JSON 策略替换为使用一个或两个`aws:SourceAccount`全局条件上下文键的策略。`aws:SourceArn`请参见以下策略示例：
5. 选择 Update policy（更新策略）。

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

Amazon Kinesis Data Analytics 的身份验证

要访问 Amazon Kinesis Data Analytics，这些凭证必须有权访问Amazon资源，例如 Amazon Kinesis 数据分析应用程序或 Amazon Elastic Compute Cloud (Amazon EC2) 实例。以下部分提供详细信息来说明如何使用 Amazon Identity and Access Management(IAM) 和 AAmazon Kinesis Data Analytics 来帮助保护对您的资源的访问。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 Amazon Kinesis Data Analytics 资源。例如，您必须拥有创建 Amazon Kinesis Data Analytics 应用程序的权限。

下面几节介绍如何管理 Amazon Kinesis Data Analytics 的权限。我们建议您先阅读概述。

- 管理 Amazon Kinesis Data Analytics 资源的访问权限的概览 (p. 166)
- 将基于身份的策略（IAM 策略）应用于 Amazon Kinesis Data Analytics (p. 169)
- Amazon Kinesis Data Analytics API 权限：操作、权限和资源参考 (p. 174)

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。您必须作为 Amazon Web Services 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 Amazon）。

如果您以编程方式访问 Amazon，则 Amazon 将提供软件开发工具包 (SDK) 和命令行界面（CLI），以便使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。有关使用推荐的方法自行对请求签名的更多信息，请参阅 Amazon 一般参考中的签名版本 4 签名流程。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 Amazon Account Management 参考指南 中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 Amazon Web Services。

联合身份是来自企业用户目录、Web 身份提供程序、Amazon Directory Service、的用户，或任何使用通过身份源提供的凭证来访问 Amazon Web Services 的用户。当联合身份访问 Amazon Web Services 账户时，他们代入角色，而角色提供临时凭证。

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以担任角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access（联合用户访问）– 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon Web Services，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon Web Services使用其它 Amazon Web Services中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage

Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。

- 主体权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅服务授权参考。
- 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。
- 服务相关角色 – 服务相关角色是与 Amazon Web Service关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

管理 Amazon Kinesis Data Analytics 资源的访问权限的概览

> Warning
>
> 对于新项目，我们建议您使用新的 Kinesis Data Analytics 工作室，而不是 SQL 应用程序的 Kinesis Data Analytics。Kinesis Data Analytics Studio 将易用性与高级分析功能相结合，使您能够在几分钟内构建复杂的流处理应用程序。

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

> Note
>
> 账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅 IAM 用户指南中的 IAM 最佳实践。

主题

Amazon Kinesis Data Analytics

在Amazon Kinesis Data Analytics 中，主要资源是应用程序。在策略中，您可以使用 Amazon Resource Name（ARN）标识策略应用到的资源。

这些资源具有关联的唯一 Amazon 资源名称 (ARN)，如下表所示。

资源类型	ARN 格式
应用程序	`arn:aws:kinesisanalytics:region:account-id:application/application-name`

Amazon Kinesis Data Analytics 提供了一组与Amazon Kinesis Data Analytics 资源配合使用的操作。有关可用操作的列表，请参阅 Amazon Kinesis Data Analytics操作 (p. 199)。

了解资源所有权

Amazon Web Services 账户对在该账户下创建的资源具有所有权，而无论创建资源的人员是谁。具体而言，资源所有者是对资源创建请求进行身份验证Amazon Web Services 账户的委托人实体（即根账户、用户或 IAM 角色）的主体。以下示例说明了它的工作原理：

- 如果您使用您的Amazon Web Services 账户根账户凭证创建应用程序，则您Amazon Web Services 账户即为该资源的所有者。（在Amazon Kinesis Data Analytics 中，资源是一个应用程序。）
- 如果您在您的中创建一个用户Amazon Web Services 账户并授予该用户创建应用程序的权限，则该用户便能创建应用程序。不过Amazon Web Services 账户，您的应用程序资源将归用户所属的所有。我们强烈建议您向角色而不是用户授予权限。
- 如果您在您的中创建的 IAM 角色Amazon Web Services 账户具有创建应用程序的权限，则能够代入该角色的任何人都可以创建应用程序。应用程序资源由用户所属的用户所有。Amazon Web Services 账户

管理对资源的访问

权限策略规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

> **Note**
>
> 本节讨论如何在 Amazon Kinesis Data Analytics 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅 IAM 用户指南中的什么是 IAM？。有关 IAM policy 语法和说明的信息，请参阅 IAM 用户指南中的 IAM JSON 策略参考。

附加到 IAM 身份的策略称作基于身份的策略 (IAM policy)。附加到资源的策略称作基于资源的 策略。Amazon Kinesis Data Analytics 只支持基于身份的策略（IAM 策略）。

主题

- 基于身份的策略（IAM 策略）(p. 167)
- 基于资源的策略 (p. 168)

基于身份的策略（IAM 策略）

您可以向 IAM 身份附加策略。例如，您可以执行以下操作：

- 将@@ 权限策略附加到账户中的用户或组-要向一个或一组用户授予创建 Amazon Kinesis Data Analytics 资源（如应用程序）的权限，您可以将权限策略附加到用户或用户所属的组。

- 向角色附加权限策略（授予跨账户权限）– 您可以向 IAM 角色附加基于身份的权限策略，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色，以向其他Amazon Web Services 账户（如账户 B）或某项 Amazon 服务授予跨账户权限，如下所述：
  1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以向角色挂载信任策略，将账户 B 标识为能够担任该角色的委托人。
  3. 之后，账户 B 管理员可以委派权限，指派账户 B 中的任何用户担任该角色。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源了。如果您需要授予 Amazon 服务权限来代入该角色，则信任策略中的主体也可以是 Amazon 服务委托人。

  有关使用 IAM 委托权限的更多信息，请参阅 IAM 用户指南中的访问权限管理。

以下是授予`kinesisanalytics:CreateApplication` 操作权限的示例策略，这是创建 Amazon Kinesis Data Analytics 应用程序所必需的。

> **Note**
>
> 这是介绍性示例策略。当您将策略附加到用户时，用户将能够使用Amazon CLI或Amazon SDK 创建应用程序。但用户需要更多权限才能配置输入和输出。此外，使用控制台时，用户需要更多权限。后续章节将提供更多相关信息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

有关对 Amazon Kinesis Data Analytics 使用策略的更多信息，请参阅将基于身份的策略（IAM 策略）应用于 Amazon Kinesis Data Analytics (p. 169)。有关用户、组、角色和权限的更多信息，请参阅 IAM 用户指南中的身份（用户、组和角色）。

基于资源的策略

其他服务 [如 Simple Storage Service（Amazon S3）] 还支持基于资源的权限策略。例如，您可以将策略附加到 S3 存储桶以管理对该存储桶的访问权限。Amazon Kinesis Data Analytics 不支持基于资源的策略。

指定策略元素：操作、效果和委托人

对于每种 Amazon Kinesis Data Analytics 资源，该服务都定义了一组 API 操作。为授予执行这些 API 操作的权限，Amazon Kinesis Data Analytics 定义了一组您可以在策略中指定的操作。某些 API 操作可能需要多个操作的权限才能执行 API 操作。有关资源和 API 操作的更多信息，请参阅Amazon Kinesis Data Analytics (p. 167)和 Amazon Kinesis Data Analytics操作 (p. 199)。

以下是最基本的策略元素：

- 资源 – 您使用 Amazon 资源名称（ARN）来标识策略应用到的资源。有关更多信息，请参阅 Amazon Kinesis Data Analytics (p. 167)。
- 操作 – 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，您可以使用 `create` 允许用户创建应用程序。

- 效果 – 用于指定用户请求特定操作时的效果（可以是允许或拒绝）。如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。
- 主体 – 在基于身份的策略（IAM policy）中，附加了策略的用户是隐式主体。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体（仅适用于基于资源的策略）。Amazon Kinesis Data Analytics 不支持基于资源的策略。

要了解 IAM policy 语法和说明的更多信息，请参阅 IAM 用户指南中的 IAM JSON 策略参考。

有关显示所有 Amazon Kinesis Data Analytics API 操作及其适用资源的列表，请参阅Amazon Kinesis Data Analytics API 权限：操作、权限和资源参考 (p. 174)。

在策略中指定条件

当您授予权限时，可使用访问策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅 IAM 用户指南中的条件。

要表示条件，您可以使用预定义的条件键。Amazon Kinesis Data Analytics 没有特定条件键。但有 Amazon 范围内的条件密钥，您可以根据需要使用。有关 Amazon 范围内的键的完整列表，请参阅《IAM 用户指南》中的条件的可用键。

将基于身份的策略（IAM 策略）应用于 Amazon Kinesis Data Analytics

以下是基于身份的策略的示例，这些示例展示了账户管理员如何将权限策略附加到 IAM 身份（即用户、组和角色）并授予对 Amazon Kinesis Data Analytics 资源执行操作的权限。

> **Important**
>
> 我们建议您首先阅读以下介绍性主题，这些主题说明了可用于管理 Amazon Kinesis Data Analytics 资源访问的基本概念和选项。有关更多信息，请参阅管理 Amazon Kinesis Data Analytics 资源的访问权限的概览 (p. 166)：

主题

- 使用 Amazon Kinesis Data Analytics 控制台所需的权限 (p. 170)
- Amazon Kinesis Data Analytics 的策略 (p. 170)
- 客户托管策略示例 (p. 171)

下面介绍权限策略示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

该策略包含一个语句：

- 第一条语句使用应用程序的亚马逊资源名称 (ARNkinesisanalytics:CreateApplication) 授予对资源执行一次 Kinesis Data Analytics 操作 () 的权限。此示例中的 ARN 指定通配符 (*)，表示为任何资源授予相应权限。

有关显示所有 Kinesis Data Analytics API 操作及其适用资源的表，请参阅Amazon Kinesis Data Analytics API 权限：操作、权限和资源参考 (p. 174)。

使用 Amazon Kinesis Data Analytics 控制台所需的权限

对于要使用 Kinesis Data Analytics 控制台的用户，您必须授予必要的权限。例如，如果希望用户拥有相应权限以创建应用程序，请授予显示账户中的流式传输源的权限，以便用户可以在控制台中配置输入和输出。

我们建议执行下列操作：

- 使用亚马逊管理的政策授予用户权限。有关可用的策略，请参阅Amazon Kinesis Data Analytics 的策略 (p. 170)。
- 创建自定义策略。在本示例中，我们建议您查看此部分中提供的示例。有关更多信息，请参阅客户托管策略示例 (p. 171)：

Amazon Kinesis Data Analytics 的策略

Amazon通过提供由创建和管理的独立 IAM policy来满足许多常用案例的要求。Amazon这些 Amazon 托管策略可针对常用案例授予必要的权限，使您免去调查所需权限的工作。有关更多信息，请参阅 IAM 用户指南中的A mazon 托管策略。

以下由亚马逊托管的策略（可附加到您账户中的用户）特定于 Amazon Kinesis Data Analytics：

- **AmazonKinesisAnalyticsReadOnly**— 授予Amazon Kinesis Data Analytics 操作的权限，使用户能够列出 Amazon Kinesis Data Analytics 应用程序并查看输入/输出配置。它还授予权限，允许用户查看 Kinesis 直播和 Kinesis Data Firehose 传输流列表。应用程序运行时，用户可以在控制台中查看源数据和实时分析结果。

- **AmazonKinesisAnalyticsFullAccess**— 授予所有 Amazon Kinesis Data Analytics 操作的权限以及允许用户创建和管理 Amazon Kinesis Data Analytics 应用程序的所有其他权限。但是，请注意以下事项：

  - 如果用户要在控制台中创建一个新的 IAM 角色，这些权限并不能满足需求（这些权限允许用户选择现有角色）。如果您希望用户能够在控制台中创建 IAM 角色，请添加 AIAMFullAccess mazon 管理的策略。

  - 在配置 AAmazon Kinesis Data Analytics 应用程序时，用户必须拥有iam:PassRole操作权限才能指定 IAM 角色。这项由亚马逊管理的政策仅向以前缀开头的 IAM 角色的用户授予iam:PassRole操作权限service-role/kinesis-analytics。

    如果用户想要使用没有此前缀的角色配置 Amazon Kinesis Data Analytics 应用程序，则必须首先明确授予用户对特定角色iam:PassRole执行操作的权限。

此外，您还可以创建您自己的自定义 IAM 策略，以授予对 AAmazon Kinesis Data Analytics 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的 用户或组。

客户托管策略示例

此部分中的示例提供了一组可附加到用户的示例策略。如果您是首次创建策略，建议您先在账户中创建用户。然后，按顺序将策略附加到用户，如此部分中的步骤所述。然后，在将每个策略附加到用户时，可使用控制台验证该策略的效果。

最初，用户没有权限并且无法在控制台中执行任何操作。在将策略附加到用户时，可以验证用户是否能在控制台中执行各种操作。

建议使用两种浏览器窗口。在一个窗口中，创建用户和授予权限。在另一个窗口中，使用用户的凭证登录 Amazon Web Services Management Console，并在授予权限时验证权限。

有关显示如何创建可用作 AAmazon Kinesis Data Analytics 应用程序执行角色的 IAM 角色的示例，请参阅 IAM 用户指南中的创建 IAM 角色。

示例步骤

步骤 1：创建一个 IAM 用户

首先，您需要创建一个用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的用户授予管理权限。您随后便Amazon可以使用一个特殊的 URL 和该用户的凭证访问。

有关说明，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。

第 2 步：允许用户执行非Amazon Kinesis Data Analytics 特有的操作

首先，授予用户在使用 Amazon Kinesis 数据分析应用程序时所需的所有非Amazon Kinesis Data Analytics 操作的权限。其中包括处理直播的权限（Amazon Kinesis Data Streams 操作、亚马逊 Kinesis Data Firehose CloudWatch 操作）和操作权限。将下面的策略附加到用户.

您需要通过提供要向其授予 `iam:PassRole` 权限的 IAM 角色名来更新策略，或者指定通配符 (*) 来表示所有 IAM 角色。这不是安全做法；但是，您可能未在此测试期间创建特定 IAM 角色。

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

步骤 3：允许用户查看应用程序列表和查看详细信息

以下策略为用户授予以下权限：

- kinesisanalytics:ListApplications 操作的权限，以便用户可以查看应用程序列表。这是服务级别的 API 调用，因此您应当指定“*”作为 Resource 值。
- kinesisanalytics:DescribeApplication 操作的权限，以便您可以获取任何应用程序的相关信息。

将此策略添加到用户。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:ListApplications"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:DescribeApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-id:application/*"
        }
    ]
```

```
}
```

使用用户凭证登录 Amazon Kinesis Data Analytics 控制台，验证这些权限。

步骤 4：允许用户启动特定应用程序

如果您希望用户能够启动现有的 Amazon Kinesis Data Analytics 应用程序，请将以下策略附加到该用户。该策略提供 `kinesisanalytics:StartApplication` 操作的权限：您必须通过提供账户 ID、Amazon区域和应用程序名称来更新策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
                {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

步骤 5：允许用户创建 Amazon Kinesis Data Analytics 应用程序

如果您希望用户创建 AAmazon Kinesis Data Analytics 应用程序，则可以将以下策略附加到该用户。您必须更新策略并提供Amazon区域、您的账户 ID 和您希望用户创建的特定应用程序名称，或者提供 “*”，以便用户可以指定任何应用程序名称（从而创建多个应用程序）。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication",
                "kinesisanalytics:UpdateApplication",
                "kinesisanalytics:AddApplicationInput",
                "kinesisanalytics:AddApplicationOutput"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

步骤 6：允许应用程序使用 Lambda 预处理

如果您希望应用程序能够使用 Lambda 预处理，请将以下策略附加到该角色。

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

Amazon Kinesis Data Analytics API 权限：操作、权限和资源参考

在设置 访问控制 (p. 164) 和编写可附加到 IAM 身份的权限策略 (基于身份的策略) 时，可使用下面的 列表作为参考。列表包含每个 Amazon Kinesis Data Analytics API 操作、您可授予执行权限的对应操作以及您可授予权限的Amazon资源。您可以在策略的 `Action` 字段中指定这些操作，并在策略的 `Resource` 字段中指定资源值。

您可以在 Amazon Kinesis Data Analytics 策略中使用Amazon范围的条件键来表示条件。有关 Amazon范围内的键的完整列表，请参阅《IAM 用户指南》https://docs.amazonaws.cn/IAM/latest/UserGuide/reference_policies_elements.html#AvailableKeys中的可用键。

> Note
>
> 要指定操作，请在 API 操作名称之前使用 `kinesisanalytics` 前缀 (例如，`kinesisanalytics:AddApplicationInput`)。

Amazon Kinesis Data Analytics API 和必需

API 操作：

所需权限（API 操作）：

资源:

Amazon Kinesis Data Analytics API 和必需

Amazon RDS API 和必需的操作权限

API 操作： AddApplicationInput (p. 202)

操作：`kinesisanalytics:AddApplicationInput`

资源：

`arn:aws:kinesisanalytics:` *`region`*`:`*`accountId`*`:application/`*`application-name`*

GetApplicationState

控制台使用名为 `GetApplicationState` 的内部方法对应用程序数据进行采样或访问。您的 Kinesis Data Analytics 服务应用程序需要拥有内部`kinesisanalytics:GetApplicationState` API 的权限，才能通过对应用程序数据进行采样或访问Amazon Web Services Management Console。

监控Amazon Kinesis Data Analytics

Kinesis Data Analytics 为您的应用程序提供监控功能。有关更多信息，请参阅监控 (p. 177)：

Amazon Kinesis Data Analytics 的合规性验

作为多项合规性计划的一部分，第三方审计员将评估 Amazon Kinesis Data Analytics 的安全Amazon性和合规性。其中包括 SOC、PCI、HIPAA 等。

有关特定合规性计划范围内的Amazon服务的列表，请参阅合规性计划范围内的亚马逊服务。有关一般信息，请参阅Amazon合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参见下载 Amazon Artifact 中的报告。

您在使用 Kinesis Data Analytics 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。如果您对 Kinesis Data Analytics 的使用需遵守 HIPAA 或 PCI 等标准，将Amazon提供以下有用资源：

- 《安全性与合规性快速入门指南》– 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署基于安全性和合规性的基准环境的步骤。
- 设计符合 HIPAA 安全性和合规性要求的架构白皮书 – 此白皮书介绍公司如何使用 Amazon 创建符合 HIPAA 标准的应用程序。
- Amazon 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config – 此Amazon服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub – 此 Amazon 服务提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

Amazon Kinesis Data Analytics

Amazon全球基础设施围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施。

除了Amazon全球基础设施之外，Kinesis Data Analytics 还提供多种功能，以帮助支持您的数据恢复能力和备份需求。

灾难恢复

Kinesis Data Analytics 在无服务器模式下运行，并通过执行自动迁移来处理主机降级、可用区可用性和其他与基础设施相关的问题。发生这种情况时，Kinesis Data Analytics 可确保在处理应用程序时不会丢失任何数据。有关更多信息，请参阅将应用程序输出永久保存到外部目标的传输模型 (p. 36)：

SQL 应用程序 Kinesis Data Analytics 中的基础设施安全

作为一项托管式服务，Amazon Kinesis Data Analytics 由《Amazon Web Services：安全流程概览》白皮书中所述的Amazon全球网络安全程序提供保护。

您可以使用Amazon发布的 API 调用通过网络访问 Kinesis Data Analytics。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service (Amazon STS) 生成临时安全凭证来对请求进行签名。

Kinesis Data Analytics 的安全最佳实践

Amazon Kinesis Data Analytics 提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的考虑因素而不是惯例。

使用 IAM 角色访问其他 Amazon 服务

您的 Kinesis Data Analytics 应用程序必须具有有效凭证才能访问其他服务中的资源，例如 Kinesis 数据流、Kinesis Data Firehose 交付流或 Amazon S3 存储桶。您不应直接在应用程序或 Amazon S3 存储桶中存储Amazon凭证。这些是不会自动轮换的长期凭证，如果它们受到损害，可能会对业务产生重大影响。

相反，您应该使用 IAM 角色来管理应用程序的临时凭证，以访问其他资源。在使用角色时，您不需要使用长期凭证即可访问其他资源。

有关更多信息，请参阅 IAM 用户指南中的以下主题：

- IAM 角色
- 针对角色的常见情形：用户、应用程序和服务

实施从属资源中的服务器端加密

静态数据和传输中的数据在 Kinesis Data Analytics 中加密，并且无法禁用此加密。您应该在依赖资源中实现服务器端加密，例如 Kinesis 数据流、Kinesis Data Firehose 传输流和 Amazon S3 存储桶。有关在从属资源中实施服务器端加密的更多信息，请参阅 数据保护 (p. 159)。

CloudTrail 用于监控 API 调用

Kinesis Data Analytics 已与Amazon CloudTrail集成，后者是一项提供 Kinesis Data Analytics 中由用户、角色或亚马逊服务所采取操作的记录。

使用收集的信息 CloudTrail，您可以确定向 Kinesis Data Analytics 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

有关更多信息，请参阅the section called “使用 Amazon CloudTrail” (p. 186)：

监控 SQL 应用程序的 Amazon Kinesis Data Analytics

监控是保持 Amazon Kinesis Data Analytics Amazon Kinesis Data Analytics 应用程序的可靠性、可用性和性能的重要方面。您应从 Amazon 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。不过，在开始监控 Amazon Kinesis Data Analytics a Analyta Analyta Analytics 之前，您应制定一个监控计划并在计划中回答下列问题

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

下一步，通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 Amazon Kinesis Data Analyta Analyta Analyta Analytics 性能的基准。在监控 Amazon Kinesis Data Analytics 时，您可以存储历史监控数据。如果您这样做，则可以将历史监控数据与当前性能数据进行比较，确定性能的正常模式和性能异常，并找出解决问题的方法。

使用Amazon Kinesis Data Analytics，您可以监控应用程序。应用程序处理数据流（输入或输出），两者都包含标识符，您可以使用这些标识符来缩小对 CloudWatch 日志的搜索范围。有关 Amazon Kinesis Data Analytics 如何处理数据流的信息，请参阅适用于 SQL 应用程序的Amazon Kinesis Data Analytics：工作原理 (p. 2)。

最重要的指标是 `millisBehindLatest`，表示应用程序读取流式传输源的滞后程度。通常情况下，滞后时间应当为零或接近零毫秒。通常会出现短暂峰值，`millisBehindLatest` 中会出现增长。

我们建议您设置 CloudWatch 警报，在应用程序读取流源超过一个小时时触发。对于某些几乎要求实时处理的使用情况（例如，将已处理的数据发送到实时应用程序），您可以选择将警报设置为更低值，如 5 分钟。

主题

监控工具

Amazon提供各种可以用来监控 Amazon Kinesis Data Analytics 的工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

自动监控工具

您可以使用以下自动化监控工具来监控 Amazon Kinesis Data Analyta Analyta Analytics，并在出现错误时进行报告：

- Amazon CloudWatch 警报 — 按您指定的时间段观察单个指标，并根据在若干时间段内观察单个指标，并根据在若干时间段内执行一项或多项操作。操作是一个发送到 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略的通知。 CloudWatch 警报将不会调用操作，因为

这些操作均处于特定状态，该状态必须改变并在指定数量的时间段内一直保持。有关更多信息，请参阅使用亚马逊进行监控 CloudWatch (p. 178)：

- Amazon CloudWatch Logs — 监控、存储和访问来自Amazon CloudTrail或其他来源的日志文件。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的监控日志文件。
- Amazon CloudWatch Events — 匹配事件并将事件传送到一个或多个目标函数或流来进行更改、捕获状态信息和采取纠正措施。有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的 “什么是亚马逊 CloudWatch 活动”。
- Amazon CloudTrail日志监控 — 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 LoCloudWatch gs 对它们进行实时监控，在 Java 中编写日志处理应用程序，以及验证您的日志文件在交付后未发生更改 CloudTrail。有关更多信息，请参阅《Amazon CloudTrail用户指南》中的 “使用CloudTrail日志文件”。

手动监控工具

监控 Amazon Kinesis Data Analytics a Analytics 的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项 AAmazon Kinesis Data AnalyticsTrusted Advisor、、和其他Amazon Web Services Management Console控制面板提供了您的Amazon环境状态 at-a-glance 视图。 CloudWatch

- CloudWatch 主页显示以下内容：
  - 当前告警和状态
  - 告警和资源图表
  - 服务运行状况

  此外，您还可以使用 CloudWatch 执行以下操作：
  - 创建自定义控制面板以监控您关心的服务
  - 绘制指标数据图，排除问题并发现趋势
  - 搜索并浏览您所有的指标
  - 创建和编辑告警以接收问题通知
- Amazon Trusted Advisor可以帮助您监控，以提高性能、可靠性、安全性和成本效益。所有用户可以使用 4 项 Trusted Advisor 检查。具有商业或企业支持计划的用户可以使用超过 50 个检查。有关更多信息，请参阅Amazon Trusted Advisor：

使用亚马逊进行监控 CloudWatch

您可以使用亚马逊监控Amazon Kinesis Data Analytics 应用程序 CloudWatch。 CloudWatch 从 Kinesis Data Anala Analyta Analytics 收集和处理原始数据，并将数据处理为易读 这些统计数据会保存两周。这样您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。默认情况下，Kinesis Data Analytics 指标数据会自动发送到 CloudWatch。有关更多信息，请参阅 Amazon CloudWatch 是什么？ 在亚马逊 CloudWatch 用户指南中。

主题

Kinesis Data Analytics

`AWS/KinesisAnalytics` 命名空间包括以下指标。

指标	描述
Bytes	读取（每个输入流）或写入（每个输出流）的字节数。 级别：每个输入流和每个输出流
KPUs	用于运行流处理应用程序的 Kinesis 处理单元的数量。每小时所用的 KPU 平均数量决定了您的应用程序所需的费用。 级别：应用程序级
MillisBehindLatest	表示应用程序读取流式源的时间相对于当前时间的延迟。 级别：应用程序级
Records	读取（每个输入流）或写入（每个输出流）的记录数。 级别：每个输入流和每个输出流
Success	1 表示到为您的应用程序配置的目的地的每个成功交付尝试；0 表示每个失败交付尝试。该指标的平均值表示执行了多少成功的交付。 级别：按目的地。
InputProcessing.Duration	Kinesis Data Analytics 执行的每次Amazon Lambda函数调用所花费的时间。 级别：每个输入流
InputProcessing.OkRecords	标有Ok状态的 Lambda 函数返回的记录数。 级别：每个输入流
InputProcessing.OkBytes	标有Ok状态的 Lambda 函数返回的记录的字节总和。 级别：每个输入流
InputProcessing.DroppedRecords	标有Dropped状态的 Lambda 函数返回的记录数。 级别：每个输入流
InputProcessing.ProcessingFailedRecords	标有ProcessingFailed状态的 Lambda 函数返回的记录数。 级别：每个输入流
InputProcessing.Success	Kinesis Data Analytics 成功调用 Lambda 的次数。 级别：每个输入流
LambdaDelivery.OkRecords	标有Ok状态的 Lambda 函数返回的记录数。 等级：每个 Lambda 目的地
LambdaDelivery.DeliveryFailedRecords	标有DeliveryFailed状态的 Lambda 函数返回的记录数。 等级：每个 Lambda 目的地

指标	描述
LambdaDelivery.Duration	Kinesis Data Analytics 执行的每次 Lambda 函数调用所花费的时间。 等级：每个 Lambda 目的地

Amazon Kinesis Data Analytics 提供以下维度的指标。

维度	描述
Flow	每个输入流：输入 每个输出流：输出
Id	每个输入流：输入 ID 每个输出流：输出 ID

查看Amazon Kinesis Data Analytics

当您的Amazon Kinesis Data Analytics 应用程序处理数据流时，Kinesis Data Analytics 会将以下指标和维度发送到 CloudWatch。您可以按照以下过程查看 Kinesis Data Analytics 的指标。

在控制台中，指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。

使用 CloudWatch 控制控制控制控制控制控制指标

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
2. 在导航窗格中，选择 Metrics（指标）。
3. 在 Kinesis Data Analytics 的 “按类别划分的CloudWatch 指标” 窗格中，选择一个指标类别。
4. 在上方窗格中，滚动以查看完整指标列表。

使用 Amazon CLI 查看指标

- 在命令提示符处，使用以下命令。

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Kinesis Data Analytics 指标按以下级别收集：

- 应用程序
- 输入流
- 输出流

创建 CloudWatch 警报以监控 Amazon Kinesis Data Analytics

您可以创建 Amazon CloudWatch 警报，在告警改变状态时发送 Amazon SNS 消息。警报会在您规定的时间范围内监控某一项指标。它在多个时间段内根据相对于给定阈值的指标值，执行一项或多项操作。操作是一个发送到 Amazon SNS 主题或 Auto Scaling 策略的通知。

告警仅为持续状态更改调用操作。要使 CloudWatch 警报调用操作，该状态必须改变并在指定的时间段内保持不变后才调用。

您可以使用Amazon Web Services Management Console、 CloudWatch Amazon CLI或 CloudWatch API 设置警报，如下所述。

使用 CloudWatch 控制控制警警警警报

1. 登录Amazon Web Services Management Console并打开 CloudWatch 主机，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 选择 Create Alarm（创建告警）。Create Alarm Wizard (创建警报向导) 启动。
3. 选择 Kinesis Analytics Metrics (Kinesis Analytics 指标)。然后滚动 Amazon Kinesis Data Analyta Analyta Analytics (Amazon Kinesis Analyt

   要仅显示 Amazon Kinesis Data Analyta Analyta Analytics 指标，请搜索文件系统的文件系统 ID。选择要为其创建警报的指标，然后选择 Next (下一步)。
4. 输入指标的 Name (名称)、Description (描述) 和 Whenever (每当) 值。
5. 如果您希望在达到警报状态时 CloudWatch 向您发送一封电子邮件，在 Whenever this alarm (每当此警报:) 字段中，选择状态为 “警报”。在 Send notification to (发送通知到) 字段中，选择一个现有 SNS 主题。如果您选择 Create topic (创建主题)，那么您就可以为新电子邮件订阅列表设置名称和电子邮件地址。此列表将保存下来并会在将来的警报字段中显示出来。

   Note

   如果您使用 Create topic (创建主题) 创建一个新 Amazon SNS 主题，那么电子邮件地址在接收通知之前必须通过验证。当警报进入警报状态时，才会发送电子邮件。如果在验证电子邮件地址之前警报状态发生了变化，那么它们不会接收到通知。
6. 在 Alarm Preview 部分中，预览您即将创建的警报。
7. 选择 Create Alarm 以创建警报。

使用 CloudWatch CLL设置警报

- 调用 `mon-put-metric-alarm`。有关更多信息，请参阅 Amazon CloudWatch CL设置警报。

使用 CloudWatch AP设置警报

- 调用 `PutMetricAlarm`。有关更多信息，请参阅 Amazon CloudWatch AP设置警报。

使用亚马逊 CloudWatch 日志

如果 AAmazon Kinesis Data Analytics 应用程序配置错误，它可能会在应用程序启动期间过渡到运行状态。或者它可以更新，但不会处理进入应用程序内部输入流的任何数据。通过向应用程序添加 CloudWatch 日志选项，您可以监视应用程序配置问题。

在以下情况下，Amazon Kinesis Data Analytics 可能会生成配置错误：

- 用于输入的 Kinesis 数据流不存在。
- 用于输入的 Amazon Kinesis Data Firehose 传输流不存在。
- 用作参考数据源的 Amazon S3 存储桶不存在。
- S3 存储桶的引用数据源中的指定文件不存在。
- 在管理相关权限的 Amazon Identity and Access Management (IAM) 角色中未定义正确的资源。
- 管理相关权限的 IAM 角色中未定义正确权限。
- Kinesis Data Analytics 无权担任管理相关权限的 IAM 角色。

有关亚马逊的更多信息 CloudWatch，请参阅亚马逊 CloudWatch 用户指南。

添加 PutLogEvents 策略操作

Amazon Kinesis Data Analytics 需要权限才能将错误配置错误写入 CloudWatch。您可以将这些权限添加到 AAmazon Kinesis Data Analytics 担任的 IAM 角色中，如下所述。有关使用 IAM 角色进行 AAmazon Kinesis Data Analytics 的更多信息，请参阅Kinesis Data Analytics 中的Identity and Access Management (p. 160)。

信任策略

要授予 Kinesis Data Analytics a Analyta Analytics 的权限以担任 IAM 角色，您可以将以下信任策略附加到该角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

权限策略

要向应用程序授予 CloudWatch 从 Kinesis Data Analytics 资源写入日志事件的权限，您可以使用以下 IAM 权限策略。

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

添加配置错误监控

使用以下 API 操作向新应用程序或现有应用程序添加日志选项，或更改现有应用程序的日志选项。CloudWatch

> **Note**
>
> 您目前只能使用 API 操作向应用程序添加 CloudWatch 日志选项。您无法使用控制控制台添加 CloudWatch 日志选项。

在创建应用程序时添加 CloudWatch 日志选项

以下代码示例演示了在创建应用程序时如何使用`CreateApplication`操作添加 CloudWatch 日志选项。有关 `Create_Application` 的更多信息，请参阅CreateApplication (p. 214)。

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

向现有应用程序添加 CloudWatch 日志选项

以下代码示例说明了如何使用 `AddApplicationCloudWatchLoggingOption` 操作将 CloudWatch 日志选项添加到现有应用程序中。有关 `AddApplicationCloudWatchLoggingOption` 的更多信息，请参阅 AddApplicationCloudWatchLoggingOption (p. 200)。

```
{
   "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>
}
```

更新现有 CloudWatch 日志选项

下面的代码示例演示了如何使用`UpdateApplication`操作修改现有 CloudWatch 日志选项。有关 `UpdateApplication` 的更多信息，请参阅 UpdateApplication (p. 249)。

```
{
   "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>
}
```

从应用程序中删除 CloudWatch 日志选项

下面的代码示例演示了如何使用DeleteApplicationCloudWatchLoggingOption操作来删除现有 CloudWatch 日志选项。有关 DeleteApplicationCloudWatchLoggingOption 的更多信息，请参阅 DeleteApplicationCloudWatchLoggingOption (p. 221)。

```
{
   "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>
}
```

配置错误

以下部分包含有关您可能在 Amazon CloudWatch Logs 中看到的来自错误配置的应用程序的错误的详细信息。

错误消息格式

由应用程序错误配置产生的错误消息将采用以下格式显示。

```
{
"applicationARN": "string",
"applicationVersionId": integer,
"messageType": "ERROR",
"message": "string",
"inputId": "string",
"referenceId": "string",
"errorCode": "string"
"messageSchemaVersion": "integer",
}
```

错误消息中的字段包含以下信息：

- applicationARN：生成应用程序的 Amazon 资源名称 (ARN)，例如：arn:aws:kinesisanalytics:us-east-1:112233445566:application/sampleApp
- applicationVersionId：遇到错误时的应用程序版本。有关更多信息，请参阅ApplicationDetail (p. 254)：
- messageType：消息类型。目前，此类型只能是 ERROR。
- message：错误的详细信息，例如：

```
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：与应用程序输入关联的 ID。仅当此输入为错误的原因时，此值才存在。如果 referenceId 存在，则此值不会存在。有关更多信息，请参阅DescribeApplication (p. 229)：
- referenceId：与应用程序引用数据源关联的 ID。仅当此源为错误的原因时，此值才会存在。如果 inputId 存在，则此值不会存在。有关更多信息，请参阅DescribeApplication (p. 229)：

- errorCode：错误的标识符。此 ID 为 InputError 或 ReferenceDataError。
- messageSchemaVersion：指定最新消息架构版本的值，当前为 1。您可以检查此值以了解错误消息架构是否已更新。

错误

Amazon Kinesis Data Analytics CloudWatch 日志中可能出现的错误包括以下内容。

资源不存在

如果为不存在的 Kinesis 输入流指定了 ARN，但是 ARN 在语法上是正确的，则会生成类似以下内容的错误。

```
{
"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"
}
```

如果使用错误的 Amazon S3 文件密钥作为参考数据，则会生成如下错误。

```
{
 "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"
}
```

角色不存在

如果为不存在的 IAM 输入角色指定了 ARN，但 ARN 在语法上正确，则会产生类似于下面的错误。

```
{
 "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"
}
```

角色无权访问资源

如果使用的输入角色无权访问输入资源，例如 Kinesis 源流，则会生成如下错误。

```
{
 "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"
}
```

通过使用记录 Kinesis Data AnalyticsAmazon CloudTrail

Amazon Kinesis Data Analytics 与集成，后者是一项服务，提供用户Amazon CloudTrail、角色或Amazon服务在 Kinesis Data Analytics (Kinesis Data Analytics) 的对象 CloudTrail 将 Kinesis Data Analytics 的API调用作为事件捕获。这些捕获包括来自 Kinesis Data Analytics 控制台的调用和对 Kinesis Data Analytics API 操作的代码调用。如果您创建跟踪，则可以使 Amazon S3 存储桶持续传送 CloudTrail 事件（包括使用 Kinesis Data Analytics）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history (事件历史记录) 中查看最新事件。通过使用 CloudTrail，您可以确定向 Kinesis Data Analytics 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解更多信息 CloudTrail，请参阅Amazon CloudTrail用户指南。

Kinesis Data Analytics CloudTrail

CloudTrail 在您创建Amazon账户时，即针对该账户启用了。当在 Kinesis Data Analytics 中发生活动时，该活动将记录在事件中，并与其他Amazon服务 CloudTrail 事件一同保存在 Event history (事件历史记录) 您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录Amazon账户中的事件（包括使用 Kinesis Data Analytics 的事件），请创建跟踪。通过跟踪 CloudTrail ，您可以将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）桶。此外，您可以配置其他Amazon服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 为配置 Amazon SNS 通知 CloudTrail
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Kinesis Data Analytics 操作均由 Kinesis Data Analytics API 参考资料记录 CloudTrail 并记录在这些操作中。例如，对`CreateApplication`和 `UpdateApplication`操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用Amazon Web Services 账户根用户还是用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Kinesis Data Analytics 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。 CloudTrail 日志文件不是公有 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了AddApplicationCloudWatchLoggingOption和DescribeApplication操作。

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

Limits

使用适用于 SQL 应用程序的 Amazon Kinesis Data Analytics 时，请注意以下限制：

- 应用程序内流中行的大小限制为 512 KB。Kinesis Data Analytics 最多使用 1 KB 来存储元数据。此元数据计入行限制。
- 应用程序中的 SQL 代码限制为 100 KB。
- 对于窗口化查询，我们建议的最长时间为 1 小时。应用程序内流存储在易失性存储中，在出现意外的应用程序中断时，会导致应用程序从易失性存储中的源数据重建流。
- 对于单个应用程序内流，我们建议的最大吞吐量为 2 到 20 MB/ 秒，具体取决于应用程序查询的复杂性。
- 该服务是专门提供的。有关更多信息，请参阅《Amazon一般参考》Amazon Kinesis Data Analytics。
- 您可以为每个账户创建 50 个 Kinesis Data Analytics 应用程序。可以创建一个案例，通过服务限制增加表来申请其他应用程序。有关更多信息，请参阅Amazon Web Services Support中心。
- 单个 Kinesis Data Analytics for SQL 应用程序可以处理的最大流传输吞吐量约为 100 MB/秒。这假设您已将应用程序内流的数量增加到最大值 64，并将 KPU 限制提高到超过 8（有关详细信息，请参阅以下限制）。如果您的应用程序需要处理超过 100 MB /秒的输入，请执行以下操作之一：
  - 使用适用于 SQL 应用程序的多个 Kinesis Data Analytics 来处理输入
  - 如果您想继续使用单个数据流和应用程序，请使用适用于 Java 应用程序的 Kinesis Data Analytics。
- Kinesis 处理单元 (KPU) 的数量限制为八个。有关如何申请提高此限额的说明，请参阅请求提高亚马逊服务限额。

  使用 Kinesis Data Analytics，您可以按实际用量付费。将根据运行流处理应用程序所使用的 KPU 平均数量来按小时费率计费。一个 KPU 可为您提供 1 个 vCPU 和 4 GB 内存。
- 每个应用程序可以具有一个流式传输源和最多一个引用数据源。
- 您最多可以为 Kinesis Data Analytics 应用程序配置三个目的地。建议您使用这些目标中的一个来永久保存应用程序内部错误流数据。
- 存储参考数据的 Amazon S3 对象的大小可达 1 GB。

- 如果在将 S3 存储桶中存储的引用数据上传到应用程序内部表后更改此数据，您需要使用 UpdateApplication (p. 249) 操作（使用 API 或 Amazon CLI）以在应用程序内部表中刷新数据。目前，Amazon Web Services Management Console在应用程序中不支持刷新引用数据。

- 目前，Kinesis Data Analytics 不支持 Amazon Kinesis 制作人库 (KPL) 生成的数据。

- 您可以为每个应用程序分配最多 50 个标签。

最佳实践

本节介绍使用Amazon Kinesis Data Analytics 应用程序时的最佳实践。

主题

管理应用程序

在管理 Amazon Kinesis Data Analytics 应用程序时，请遵循以下最佳实践：

- 设置亚马逊 CloudWatch 警报 — 您可以使用 Kinesis Data Analytics 提供的 CloudWatch 指标来监控以下内容：
  - 输入字节和输入记录（输入应用程序的字节和记录的数目）
  - 输出字节和输出记录
  - `MillisBehindLatest`（从流式传输源进行读取时的应用程序滞后）

  我们建议您针对生产中应用程序的以下指标设置至少两个 CloudWatch 警报：
  - `MillisBehindLatest`— 在大多数情况下，我们建议您将此警报设置为在应用程序比最新数据慢 1 小时（平均 1 分钟）时触发。对于 end-to-end 处理需求较低的应用程序，可以将其调整为较低的容差。此警报可帮助确保应用程序读取最新数据。

- 为避免出现`ReadProvisionedThroughputException`异常，请将从同一 Kinesis 数据流读取的生产应用程序数量限制为两个应用程序。

  Note

  在这种情况下，应用程序 指可从流式传输源读取的任何应用程序。只有 Kinesis Data Analytics 应用程序才能从 Kinesis Data Firehose 传输流中读取。但是，许多应用程序可以从 Kinesis 数据流中读取，例如 Kinesis Data Analytics 应用程序或Amazon Lambda。建议的应用程序限制指的是配置为从流式传输源读取的总应用程序数。

  Amazon Kinesis Data Analytics 大约每秒读取一个流媒体源一次。不过，滞后的应用程序可以更快的速率读取数据以便保持一致。要允许应用程序的足够吞吐量以便保持一致，请限制读取同一数据源的应用程序的数目。

- 将从同一 Kinesis Data Firehose 交付流读取的生产应用程序数量限制为一个应用程序。

  Kinesis Data Firehose 传输流可以写入Amazon S3 和Amazon Redshift ft 等目的地。它也可以作为 Kinesis Data Analytics 应用程序的流媒体源。因此，我们建议您不要为每个 Kinesis Data Firehose 交付流配置多个 Kinesis Data Analytics 应用程序。这有助于确保传输流还可以传输到其他目标。

扩展应用程序

通过从默认值（一）开始主动增加应用程序内输入流的数量，设置应用程序以满足您未来的扩展需求。我们建议根据应用程序的吞吐量选择以下语言：

- 如果您的应用程序的扩展需求超过 100 MB/秒，请对 SQL 应用程序使用多个流和 Kinesis Data Analytics。
- 如果您想使用单一数据流和应用程序，请使用适用于 Flink 应用程序的 Kinesis Data Analytics。

定义输入架构

在控制台中配置应用程序输入时，您首先指定流式传输源。随后，控制台将使用发现 API（请参阅 DiscoverInputSchema (p. 233)）对流式传输源上的记录采样来推断架构。此外，架构在产生的应用程序内部流中定义列的名称和数据类型。控制台将显示架构。建议您对此推断的架构执行以下操作：

- 充分测试推断的架构。发现过程仅使用流式传输源上的记录示例来推断架构。如果您的流式传输源有多种记录类型，则发现 API 可能错过了一种或多种记录类型的采样。这种情况会导致架构不能准确反映流式传输源上的数据。

  当您的应用程序启动时，这些丢失的记录类型可能会导致解析错误。Amazon Kinesis Data Analytics 将这些记录发送到应用程序内错误流。要减少这些解析错误，建议您在控制台中以交互方式测试推断的架构，并监控应用程序内部流是否缺少记录。

- Kinesis Data Analytics API 不支持在输入配置中指定列的`NOT NULL`约束。如果您希望在应用程序内部流中指定列的 `NOT NULL` 约束，请使用应用程序代码创建这些应用程序内部流。随后，您可以将数据从一个应用程序内部流复制到另一个应用程序内部流，之后将强制实施该约束。

  当需要`NULL`值时，任何尝试插入带有值的行都会导致错误。Kinesis Data Analytics 将这些错误发送到应用程序内错误流。

- 放宽发现过程所推断的数据类型。发现过程将根据流式传输源中记录的随机采样来建议列和数据类型。建议您仔细审查这些列和数据类型，并考虑放宽这些数据类型以涵盖输入中所有可能的记录情况。这可确保应用程序在运行时出现的分析错误更少。例如，如果推断的架构具有 `SMALLINT` 作为列类型，则可考虑将列类型更改为 `INTEGER`。

- 在应用程序代码中使用 SQL 函数来处理任何非结构化的数据或列。您的输入中可包含非结构化的数据或列（例如，日志数据）。有关示例，请参阅 示例：转换 DateTime 值 (p. 112)。处理此类数据的一种方法是，定义仅具有一个类型 `VARCHAR(N)` 的列的架构，其中 `N` 是您应在流中看到的可能最大的行。随后，可在您的应用程序代码中读取传入记录，并使用 `String` 和 `Date Time` 函数来解析和架构化原始数据。

- 确保您能够完全处理包含两个级别以上的嵌套的流式传输源数据。当源数据为 JSON 时，您可以使用嵌套。发现 API 会推断可展平一个嵌套层的架构。对于两层嵌套，发现 API 也将尝试展平嵌套。在超出两层嵌套的情况下，对展平的支持是有限的。要完全处理嵌套，您必须手动修改推断的架构来满足您的需求。可使用下列任一策略执行此操作：

  - 使用 JSON 行路径可选择性地只提取应用程序所需的键值对。JSON 行路径为要引入应用程序中的特定键值对提供了一个指针。您可为任何级别的嵌套执行此操作。

- 使用 JSON 行路径可选择性地提取复杂的 JSON 对象，然后在应用程序代码中使用字符串处理函数提取所需的特定数据。

连接到输出

建议每个应用程序具有至少两个输出：

- 使用第一个目标插入 SQL 查询的结果。
- 使用第二个目的地插入整个错误流，然后通过 Kinesis Data Firehose 传输流将其发送到 S3 存储桶。

创作应用程序代码

我们建议执行下列操作：

- 在 SQL 语句中，不要指定超过 1 个小时的基于时间的窗口，原因如下：
  - 有时需要重新启动应用程序，这要么是因为你更新了应用程序，要么是出于 Kinesis Data Analytics 内部原因。在重新启动时，将从流式数据源中重新读取窗口中包含的所有数据。这需要一段时间，Kinesis Data Analytics 才能为该窗口发出输出。
  - Kinesis Data Analytics 必须在此期间维护与应用程序状态相关的所有内容，包括相关数据。这会消耗大量的 Kinesis Data Analytics 处理单元。
- 在开发期间，在 SQL 语句中设置较小的窗口以便更快地查看结果。当您将应用程序部署到生产环境时，可以设置适当的窗口大小。
- 不要使用单个复杂的 SQL 语句，而是在将结果保存到中间应用程序内部流的每个步骤中将此语句分成多个语句。这可帮助您加快调试速度。
- 在您使用滚动窗口时，建议您使用两个窗口，一个用于处理时间，另一个用于逻辑时间（接收时间或事件时间）。有关更多信息，请参阅时间戳和 ROWTIME 列 (p. 65)：

测试应用程序

当您更改 Kinesis Data Analytics 应用程序的架构或应用程序代码时，我们建议在将更改部署到生产环境之前使用测试应用程序对其进行验证。

设置测试应用程序

您可以通过控制台或使用 Amazon CloudFormation 模板设置测试应用程序。使用 Amazon CloudFormation 模板有助于确保对测试应用程序和活动应用程序进行的代码更改保持一致。

设置测试应用程序时，您可以将应用程序连接到活动数据，或者使用模拟数据来填充流以进行测试。建议通过两种方法来使用模拟数据填充流：

- 使用 Kinesis Data Generator (KDG)。KDG 使用数据模板将随机数据发送到 Kinesis 流。KDG 易于使用，但不适合测试数据项之间的复杂关系，例如检测数据热点或异常的应用程序。
- 使用自定义 Python 应用程序将更复杂的数据发送到 Kinesis 数据流。Python 应用程序可以生成数据项之间的复杂关系，例如热点或异常。有关将数据集群化发送到数据热点的 Python 应用程序示例，请参阅示例：检测流上的热点 (HOTSPOTS 函数) (p. 145)。

运行测试应用程序时，使用目的地（例如 Kinesis Data Firehose 传输流到 Amazon Redshift 数据库）查看结果，而不是在控制台上查看应用程序内的数据流。控制台上显示的数据是流的采样，并未包含所有记录。

测试架构更改

更改应用程序的输入流架构时，使用测试应用程序来验证以下情况：

- 来自您的流中的数据强制转换为正确的数据类型。例如，确保日期时间数据未作为字符串接收到应用程序中。
- 进行解析的数据强制转换为您需要的数据类型。如果出现解析或强制转换错误，您可以在控制台上查看，或者为错误流分配目标并在目标存储中查看错误。
- 字符数据的数据字段具有足够的长度，并且应用程序未截断字符数据。您可以在目标存储中查看数据记录，验证未截断您的应用程序数据。

测试代码更改

测试对 SQL 代码的更改时，需要了解关于您应用程序的领域的一些知识。您必须确定需要能够测试哪些输出以及正确的输出应该是什么。有关在修改应用程序的 SQL 代码时可能需要验证的问题领域，请参阅Amazon Kinesis Data Analytics S (p. 195)。

Amazon Kinesis Data Analytics S

以下内容可帮助您排查在 Amazon Kinesis Data Analytics for SQL 应用程序中可能遇到的问题。

主题

无法运行 SQL 代码

如果你需要弄清楚如何让特定 SQL 语句正常工作，那么在使用 Kinesis Data Analytics 时，你有几种不同的资源：

- 有关 SQL 语句的更多信息，请参阅示例应用程序 (p. 78)。此部分提供了一些可使用的 SQL 示例。
- Amazon Kinesis Data Analytics SQL 参考提供了编写流式传输 SQL 语句的详细指南。
- 如果你仍然遇到问题，我们建议你在 Kinesis Data Analytics 论坛上提问。

无法检测到或发现我的架构

在某些情况下，Kinesis Data Analytics 无法检测或发现架构。在其中许多情况下，您仍然可以使用 Kinesis Data Analytics。

假设您具有不使用分隔符的 UTF-8 编码数据，或具有使用非逗号分隔值 (CSV) 格式的数据，或发现 API 未发现您的架构。在这些情况下，可以手动定义架构或使用字符串操作函数来构建数据。

为了发现直播架构，Kinesis Data Analytics 会随机采样直播中的最新数据。如果您没有持续向直播发送数据，Kinesis Data Analytics 可能无法检索样本和检测架构。有关更多信息，请参阅针对流数据使用架构发现功能 (p. 15)：

引用数据已过时

在 Amazon Simple Storage Service (Amazon S3) 对象启动或更新应用程序时，或者在服务问题导致的应用程序中断期间，参考数据会加载到应用程序中。

更新底层 Amazon S3 对象时，参考数据不会加载到应用程序中。

如果应用程序中的引用数据不是最新的，可以通过以下步骤重新加载这些数据：

1. 在 Kinesis Data Analytics 控制台上，在列表中选择应用程序名称，然后选择应用程序详细信息。
2. 选择 Go to SQL editor (转到 SQL 编辑器) 打开应用程序的 Real-time analytics (实时分析) 页面。

3. 在 Source Data (源数据) 视图中，选择引用数据表名称。
4. 依次选择 Actions (操作)、Synchronize reference data table (同步引用数据表)。

应用程序不写入到目标

如果数据未被写入到目标，请检查以下各项：

- 验证应用程序的角色具有足够权限来访问目标。有关更多信息，请参阅 写入 Kinesis 直播的权限政策 (p. 162) 或 写入到 Firehose 传输流的权限策略 (p. 162)。
- 验证是否已正确配置应用程序目标，并且应用程序正在使用正确的输出流名称。
- 查看您的输出流的亚马逊 CloudWatch 指标，看看是否正在写入数据。有关使用 CloudWatch 指标的信息，请参阅使用亚马逊进行监控 CloudWatch (p. 178)。
- 使用添加 CloudWatch 日志流the section called “AddApplicationCloudWatchLoggingOption” (p. 200)。您的应用程序将配置错误写入日志流。

如果该角色和目标配置看起来正确，请尝试重启应用程序，同时为 `LAST_STOPPED_POINT` 指定 InputStartingPositionConfiguration (p. 278)。

要监控的重要应用程序运行状况参数

要确保您的应用程序正常运行，我们建议您监控某些重要参数。

要监控的最重要的参数是亚马逊 CloudWatch 指标`MillisBehindLatest`。此指标表示落后于您从流中读取的当前时间多久。此指标可帮助确定您是否足够快速地处理源流中的记录。

通常，您应该设置 CloudWatch 警报，以便在落后超过一小时时触发。但是，时间量取决于您的使用案例。您可以按需进行调整。

有关更多信息，请参阅最佳实践 (p. 191)：

在运行应用程序时出现无效代码错误

当您无法为 Amazon Kinesis Data Analytics 应用程序保存和运行 SQL 代码时，常见原因如下：

- 在 SQL 代码中重新定义了该流 — 在创建了流和与该流关联的泵之后，您无法在代码中重新定义相同的流。有关创建直播的更多信息，请参阅《Amazon Kinesis Data Stream s SQL 参考》中的 CREATE STREAM。有关创建数据泵的更多信息，请参阅 CREATE PUMP。
- GROUP BY 子句使用多个 ROWTIME 列 — 在 GROUP BY 子句中只能指定一个 ROWTIME 列。有关更多信息，请参阅 Amazon Kinesis Data Analytics SQL 参考中的 G ROUP BY 和 ROW TIM E。
- 一种或多种数据类型的转换无效 — 在这种情况下，您的代码具有无效的隐式转换。例如，您可能在代码中将 `timestamp` 转换成 `bigint`。
- 流的名称与服务预留流名称相同 - 流的名称不能与服务预留流 `error_stream` 的名称相同。

应用程序正在将错误写入到错误流

如果您的应用程序正在将错误写入到应用程序内部错误流，则可以使用标准库解码 `DATA_ROW` 字段中的值。有关错误流的更多信息，请参阅错误处理 (p. 36)。

吞吐量不足或过高 MillisBehindLatest

如果您的应用程序的MillisBehindLatest指标稳步增加或持续超过 1000（一秒），则可能是由于以下原因造成的：

- 检查您的应用程序的InputBytes CloudWatch 指标。如果提取速度超过 4 MB/秒，这可能会导致 `MillisBehindLatest` 增加。要提高应用程序的吞吐量，请增加 `InputParallelism` 参数值。有关更多信息，请参阅并行处理输入流以增加吞吐量 (p. 25)：
- 检查应用程序的输出传输 Success 指标以确定传输到目标是否失败。确认您已正确配置输出，并且您的输出流具有足够的容量。
- 如果您的应用程序使用某个Amazon Lambda函数进行预处理或作为输出，请检查该应用程序的 InputProcessing.Duration 或LambdaDelivery. Durati CloudWatch on 指标。如果 Lambda 函数调用持续时间超过 5 秒，请考虑执行以下操作：
  - 增加 Lambda 函数的内存分配。您可以在 Amazon Lambda 控制台的 Configuration (配置) 页面上的 Basic settings (基本设置) 中执行此操作。有关更多信息，请参阅 https://docs.amazonaws.cn/lambda/latest/dg/resource-model.html 开发人员指南 中的Amazon Lambda配置 Lambda 函数。
  - 在应用程序的输入流中提高分片数。这会提高应用程序将调用的并行函数的数量，从而可能提高吞吐量。
  - 确认函数没有进行影响性能的阻止性调用，如同步的外部资源请求。
  - 检查 Amazon Lambda 函数以确定是否具有可提高性能的其他方面。检查应用程序 Lambda 函数 CloudWatch 的日志。有关更多信息，请参阅《Amazon Lambda开发者指南》中的 “访问亚马逊 CloudWatch 指标”。
- 验证您的应用程序未达到 Kinesis 处理单元 (KPU) 的默认限制。如果您的应用程序达到该限制，您可以请求提高限制。有关更多信息，请参阅 自动扩展应用程序以提高吞吐量 (p. 37)。

Kinesis Data Analytics

有关Amazon Kinesis Data Analytics 支持的 SQL 语言元素的信息，请参阅Ama zon Kinesis Data Analytics SQL 参考资料。

API 引用

> **Note**
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

您可以使用Amazon CLI来浏览Amazon Kinesis Data Analytics API。本指南提供了使用 Amazon CLI 的 Amazon Kinesis Data Analytics 入门 (p. 40) 练习。

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

主题

操作

支持以下操作：

AddApplicationCloudWatchLoggingOption

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

添加 CloudWatch 日志流来监控应用程序配置错误。有关在 Amazon Kinesis Analytics 应用程序中使用 CloudWatch 日志流的更多信息，请参阅使用亚马逊 CloudWatch 日志。

请求语法

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "string",
      "RoleARN": "string"
   },
   "CurrentApplicationVersionId": number
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 200)

Kinesis Analytics 应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

CloudWatchLoggingOption (p. 200)

提供 CloudWatch 日志流亚马逊资源名称（ARN）和 IAM 角色 ARN。注意：要向写入应用程序消息 CloudWatch，所使用的 IAM 角色必须启用`PutLogEvents`策略操作。

类型：CloudWatchLoggingOption (p. 259) 对象

必需：是

CurrentApplicationVersionId (p. 200)

Kinesis Analytics 应用程序的版本 ID。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

AddApplicationInput

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

为您的Amazon Kinesis 应用程序添加流媒体源。有关概念信息，请参阅配置应用程序输入。

可以在创建应用程序时添加流源，也可以在创建应用程序后使用此操作添加流源。有关更多信息，请参阅CreateApplication。

任何配置更新（包括使用此操作添加流式传输源）都会生成新版本的应用程序。您可以使用 DescribeApplication 操作来找到当前的应用程序版本。

此操作需要执行 `kinesisanalytics:AddApplicationInput` 操作的权限。

请求语法

```
{
   "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"
    }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 202)

您希望为其添加流媒体源的现有 Amazon Kinesis Analytics 应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：[a-zA-Z0-9_.-]+

必需：是

CurrentApplicationVersionId (p. 202)

您的Amazon Kinesis Analytics 应用程序的当前版本。您可以使用 DescribeApplication 操作来找到当前的应用程序版本。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

Input (p. 202)

要添加的输入。

类型：Input (p. 264) 对象

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

CodeValidationException

用户提供的应用程序代码（查询）无效。这可能是一个简单的语法错误。

HTTP 状态代码：400

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

AddApplicationInputProcessingConfiguration

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

InputProcessingConfiguration向应用程序中添加。在应用程序的 SQL 代码执行之前，输入处理器会在应用程序的 SQL 代码执行之前预处理输入流上的记录。目前，唯一可用的输入处理器为 Amazon Lambda。

请求语法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 205)

您要向其添加输入处理配置的应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：[a-zA-Z0-9_.-]+

必需：是

CurrentApplicationVersionId (p. 205)

您要向其添加输入处理配置的应用程序的版本。您可以使用DescribeApplication操作来获取当前应用程序版本。如果指定的版本不是当前版本，ConcurrentModificationException则返回。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

InputId (p. 205)

要添加输入处理配置的输入配置的 ID。您可以使用DescribeApplication操作获取应用程序的输入 ID 列表。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：[a-zA-Z0-9_.-]+

必需：是

InputProcessingConfiguration (p. 205)

InputProcessingConfiguration要添加到应用程序中的。

类型：InputProcessingConfiguration (p. 274) 对象

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK

- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

AddApplicationOutput

> **Note**
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

将外部目标添加到您的 Amazon Kinesis Analytics 应用程序。

如果您希望 Amazon Kinesis Analytics 将数据从应用程序中的应用程序内部流传递到外部目标（例如 Amazon Kinesis Firehose 传输流或Amazon Lambda 函数），则可以使用此将相关配置添加到您的应用程序操作。您可以为您的应用程序配置一个或多个输出。每个输出配置都映射一个应用程序内部流和一个外部目标。

您可以使用其中一个输出配置将数据从应用程序内的错误流传输到外部目标，以便您分析错误。有关更多信息，请参阅了解应用程序输出（目标）。

任何配置更新（包括使用此操作添加流式传输源）都会生成新版本的应用程序。您可以使用 DescribeApplication 操作来找到当前的应用程序版本。

有关可配置的应用程序输入和输出数量的限制，请参阅限制。

此操作需要执行 `kinesisanalytics:AddApplicationOutput` 操作的权限。

请求语法

```
{
   "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"
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 208)

要将输出配置添加到的应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：[a-zA-Z0-9_.-]+

必需：是

CurrentApplicationVersionId (p. 208)

您要向其添加输出配置的版本。您可以使用DescribeApplication操作来获得当前应用程序版本。如果指定的版本不是当前版本，ConcurrentModificationException则返回。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

Output (p. 208)

对象的数组，每个对象描述一项输出配置。在输出配置中，指定应用程序内部流的名称、目标（即 Amazon Kinesis 流、Amazon Kinesis Firehose 传输流或 Amazon Lambda 函数），并记录在写入该目标时要使用的格式。

类型：Output (p. 298) 对象

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

AddApplicationReferenceDataSource

> **Note**
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

将引用数据源添加到现有应用程序。

Amazon Kinesis Analytics 读取参考数据（即 Amazon S3 对象），并在应用程序中创建应用程序内部表。在请求中，您提供源（S3 存储桶名称和对象键名称），要创建的应用程序内部表的名称，以及描述 Amazon S3 对象中的数据如何映射到所生成应用程序内部表中的列的必要映射信息。

有关概念信息，请参阅配置应用程序输入。有关您可以添加到应用程序的数据源的限制，请参阅限制。

此操作需要执行 `kinesisanalytics:AddApplicationOutput` 操作的权限。

请求语法

```
{
   "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"
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 211)

现有应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

CurrentApplicationVersionId (p. 211)

您为其添加引用数据来源的应用程序版本。您可以使用DescribeApplication操作来获得当前应用程序版本。如果指定的版本不是当前版本，`ConcurrentModificationException`则返回。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

ReferenceDataSource (p. 211)

参考数据源可以是 Amazon S3 存储桶中的对象。Amazon Kinesis Analytics 读取对象，并将数据复制到创建的应用程序内部表。您需要提供 S3 存储桶、对象键名称和创建的结果应用程序内部表。您还必须提供具有必要权限的 IAM 角色，Amazon Kinesis Analytics 可以代入该角色代表您从 S3 存储桶中读取对象。

类型：ReferenceDataSource (p. 306) 对象

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

CreateApplication

> **Note**
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

创建Amazon Kinesis Analytics 应用程序。您可以将每个应用程序配置为一个流源作为输入，使用应用程序代码来处理输入，最多三个目的地，您希望 Amazon Kinesis Analytics 写入应用程序的输出数据。有关概述，请参阅其工作原理。

在输入配置中，您将流源映射到应用程序内流，可以将其视为不断更新的表。在映射中，您必须为应用程序内部流提供架构，并将应用程序内部流流中的每个数据列映射到流式源中的数据元素。

您的应用程序代码是一个或多个 SQL 语句，用于读取输入数据、转换输入数据并生成输出。您的应用程序代码可以创建一个或多个 SQL 工件，例如 SQL 流或泵。

在输出配置中，您可以将应用程序配置为将数据从应用程序中创建的应用程序内流写入到最多三个目的地。

要从源流中读取数据或将数据写入目标流，Amazon Kinesis Analytics 需要您的权限。您可以通过创建 IAM 角色来授予这些权限。此操作需要执行 `kinesisanalytics:CreateApplication` 操作的权限。

有关创建 Amazon Kinesis Analytics 应用程序的入门练习，请参阅入门。

请求语法

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

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationCode (p. 214)

一个或多个读取输入数据、转换数据并生成输出的 SQL 语句。例如，您可以编写一条 SQL 语句，该语句从一个应用程序内部流中读取数据，生成供应商广告点击次数的运行平均值，并使用泵将结果行插入另一个应用程序内部流中。有关典型模式的更多信息，请参阅应用程序代码。

您可以提供这样的一系列 SQL 语句，其中一条语句的输出可以用作下一条语句的输入。您可以通过创建应用程序内部流和泵来存储中间结果。

请注意，应用程序代码必须使用在 `Outputs` 中指定的名称创建流。例如，如果您的 `Outputs` 定义了名为 `ExampleOutputStream1` 和 `ExampleOutputStream2` 的输出流，则您的应用程序代码必须创建这两个流。

类型：字符串

长度约束：最小长度为 0。最大长度为 102400。

必需：否

ApplicationDescription (p. 214)

应用程序的摘要描述。

类型：字符串

长度约束：最小长度为 0。长度上限为 1024。

必需：否

ApplicationName (p. 214)

Amazon Kinesis Analytics 应用程序的名称（例如，`sample-app`）。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

CloudWatchLoggingOptions (p. 214)

使用此参数可配置 CloudWatch 日志流来监控应用程序配置错误。有关更多信息，请参阅使用 Amazon CloudWatch 日志。

类型：CloudWatchLoggingOption (p. 259) 对象数组

必需：否

Inputs (p. 214)

使用该参数可配置应用程序输入。

您可以将您的应用程序配置为接收来自单个流式传输源的输入。在此配置中，您会将此流式传输源映射到已创建的应用程序内部流。然后，您的应用程序代码可以像对表一样查询应用程序内部流（您可以将其视为连续更新的表）。

对于流式传输源，您可以在该流上提供其 Amazon 资源名称（ARN）和数据格式（例如，JSON、CSV 等）。您还必须提供可由 Amazon Kinesis Analytics 代入以代表您读取此流的 IAM 角色。

要创建应用程序内部流，您需要指定一个架构来将您的数据转换成 SQL 中使用的架构化版本。在架构中，您需要提供流式传输源中数据元素的必要映射，以记录应用程序内部流中的列。

类型：Input (p. 264) 对象数组

必需：否

Outputs (p. 214)

您可以配置应用程序输出，将来自任何应用程序内流的数据写入最多三个目的地。

这些目的地可以是Amazon Kinesis 直播、Amazon Kinesis Firehose 配送流、Amazon Lambda 目的地或三者的任意组合。

在配置中，您可以指定应用程序内流名称、目标流或 Lambda 函数 Amazon 资源名称 (ARN) 以及写入数据时使用的格式。您还必须提供 Amazon Kinesis Analytics 代入以代表您对目标流或 Lambda 函数进行写入的。

在输出配置中，您还提供输出流或 Lambda 函数 ARN。对于流式目的地，您可以在该流中提供数据的格式（例如，JSON、CSV）。您还必须提供 Amazon Kinesis Analytics 代入以代表您对流或 Lambda 函数进行写入的。

类型：Output (p. 298) 对象数组

必需：否

Tags (p. 214)

分配给应用程序的一个或多个标签的列表。标签是用于标识应用程序的键/值对。请注意，应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。有关更多信息，请参阅使用标签。

类型：Tag (p. 314) 对象数组

数组成员：最少 1 项。最多 20项。

必需：否

响应语法

```
{
   "ApplicationSummary": {
      "ApplicationARN": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

ApplicationSummary (p. 217)

针对您的`CreateApplication`请求，Amazon Kinesis Analytics 会返回回复，其中包含其创建的应用程序的摘要，包括应用程序亚马逊资源名称 (ARN)、名称和状态。

类型：ApplicationSummary (p. 257) 对象

错误

CodeValidationException

用户提供的应用程序代码（查询）无效。这可能是一个简单的语法错误。

HTTP 状态代码：400

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

LimitExceededException

超过了允许的应用程序数量。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

TooManyTagsException

创建的应用程序的标签过多，或者向应用程序添加的标签过多。请注意，应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DeleteApplication

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

删除指定的应用程序。Amazon Kinesis Analytics 会停止应用程序执行并删除应用程序，包括所有应用程序对象（例如应用程序内流、参考表和应用程序代码）。

此操作需要执行 `kinesisanalytics:DeleteApplication` 操作的权限。

请求语法

```
{
   "ApplicationName": "string",
   "CreateTimestamp": number
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 219)

要删除的Amazon Kinesis Analytics 应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

CreateTimestamp (p. 219)

您可以使用`DescribeApplication`操作来获取此值。

类型：Timestamp

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DeleteApplicationCloudWatchLoggingOption

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

从应用程序中删除 CloudWatch 日志流。有关在 Amazon Kinesis Analytics 应用程序中使用 CloudWatch 日志流的更多信息，请参阅使用亚马逊 CloudWatch 日志。

请求语法

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOptionId": "string",
   "CurrentApplicationVersionId": number
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 221)

Kinesis Analytics 应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：[a-zA-Z0-9_.-]+

必需：是

CloudWatchLoggingOptionId (p. 221)

要删除 CloudWatch 的日志记录选项的。CloudWatchLoggingOptionId您可以使用该CloudWatchLoggingOptionIdDescribeApplication操作获取。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：[a-zA-Z0-9_.-]+

必需：是

CurrentApplicationVersionId (p. 221)

Kinesis Analytics 应用程序的版本 ID。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DeleteApplicationInputProcessingConfiguration

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

InputProcessingConfiguration从输入中删除。

请求语法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 223)

Kinesis Analytics 应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：[a-zA-Z0-9_.-]+

必需：是

CurrentApplicationVersionId (p. 223)

Kinesis Analytics 应用程序的版本 ID。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

InputId (p. 223)

要从中删除输入处理配置的输入配置的 ID。您可以使用DescribeApplication操作获取应用程序的输入 ID 列表。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：[a-zA-Z0-9_.-]+

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DeleteApplicationOutput

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

从应用程序配置中删除输出目标配置。Amazon Kinesis Analytics 将不再将数据从相应的应用程序内流写入到外部输出目标。

此操作需要执行 `kinesisanalytics:DeleteApplicationOutput` 操作的权限。

请求语法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "OutputId": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 225)

Amazon Kinesis Analytics 应用程序

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

CurrentApplicationVersionId (p. 225)

Amazon Kinesis Analytics 应用程序 您可以使用该DescribeApplication操作来获取当前应用程序版本。如果指定的版本不是当前版本，`ConcurrentModificationException`则返回。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

OutputId (p. 225)

要删除的配置的 ID。无论是在创建应用程序时还是之后使用该AddApplicationOutput操作添加至应用程序的每个输出配置，都有一个唯一的 ID。您需要提供 ID 才能唯一标识要从应用程序配置中删除的输出配置。您可以使用该DescribeApplication操作来获取具体信息`OutputId`。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DeleteApplicationReferenceDataSource

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

从指定的应用程序配置中删除参考数据源配置。

如果应用程序正在运行，Amazon Kinesis Analytics 会立即删除您使用该AddApplicationReferenceDataSource操作创建的应用程序内表。

此操作需要执行 `kinesisanalytics.DeleteApplicationReferenceDataSource` 操作的权限。

请求语法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceId": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 227)

现有应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

CurrentApplicationVersionId (p. 227)

应用程序的版本。您可以使用DescribeApplication操作来获取当前应用程序版本。如果指定的版本不是当前版本，`ConcurrentModificationException`则返回。

类型：长整型

有效范围：最小值为 1。最大值为 99999999999。

必需：是

ReferenceId (p. 227)

引用数据源的 ID。当您使用向应用程序添加参考数据源时，Amazon Kinesis Analytics 会分配一个 ID。AddApplicationReferenceDataSource您可以使用该DescribeApplication操作来获取参考 ID。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DescribeApplication

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

返回有关有关特定Amazon Kinesis Analytics 应用程序的信息。

如果您想检索账户中所有应用程序的列表，请使用ListApplications操作。

此操作需要执行 `kinesisanalytics:DescribeApplication` 操作的权限。你可以用它`DescribeApplication`来获取当前的应用程序 versionId，你需要调用其他操作，例如`Update`。

请求语法

```
{
   "ApplicationName": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 229)

应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

响应语法

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

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

ApplicationDetail (p. 229)

提供应用程序的描述，例如应用程序 Amazon 资源名称 (ARN)、状态、最新版本以及输入和输出配置的详细信息。

类型：ApplicationDetail (p. 254) 对象

错误

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DiscoverInputSchema

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

通过评估指定流源（亚马逊 Kinesis 流或 Amazon Kinesis Amazon Kinesis Firehose e 传输流）或 S3 对象上的示例记录来推断架构。在响应中，该操作返回推断的架构以及该操作用于推断架构的示例记录。

在为应用程序配置流源时，您可以使用推断的架构。有关概念信息，请参阅配置应用程序输入。请注意，当您使用 Amazon Kinesis Analytics 控制台创建应用程序时，控制台使用此操作来推断架构并将其显示在控制台用户界面中。

此操作需要执行 `kinesisanalytics:DiscoverInputSchema` 操作的权限。

请求语法

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "string"
   },
   "ResourceARN": "string",
   "RoleARN": "string",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string",
      "RoleARN": "string"
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

InputProcessingConfiguration (p. 233)

InputProcessingConfiguration用于在发现记录架构之前对记录进行预处理。

类型：InputProcessingConfiguration (p. 274) 对象

必需：否

InputStartingPositionConfiguration (p. 233)

你想让 Amazon Kinesis Analytics 开始读取来自指定流媒体源发现目的的记录的时刻。

类型：InputStartingPositionConfiguration (p. 278) 对象

必需：否

ResourceARN (p. 233)

流式源的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN (p. 233)

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

S3Configuration (p. 233)

指定此参数可从 Amazon S3 对象中的数据中发现架构。

类型：S3Configuration (p. 309) 对象

必需：否

响应语法

```
{
   "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" ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

InputSchema (p. 234)

从流媒体源推断出来的架构。它可识别流式源中的数据的格式以及每个数据元素映射到您可以创建应用程序内部流中的相应列的方式。

类型：SourceSchema (p. 313) 对象

ParsedInputRecords (p. 234)

元素数组，其中每个元素对应流记录中的一行（一条流记录可以有多行）。

类型：字符串数组

ProcessedInputRecords (p. 234)

由`InputProcessingConfiguration`参数中指定的处理器修改的流数据。

类型：字符串数组

RawInputRecords (p. 234)

为推断架构而采样的原始流数据。

类型：字符串数组

错误

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceProvisionedThroughputExceededException

由于亚马逊 Kinesis Streams 的缘故，Discovery 未能从直播来源获得记录 ProvisionedThroughputExceededException。有关更多信息，请参阅 GetRecordsAmazon Kinesis Streams Streams Streams Streams

HTTP 状态代码：400

ServiceUnavailableException

该服务不可用。请退后并重试该操作。

HTTP 状态代码：500

UnableToDetectSchemaException

数据格式无效。Amazon Kinesis Analytics 无法检测给定流媒体源的架构。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ListApplications

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

返回您账户中的Amazon Kinesis Analytics 应用程序列表。对于每个应用程序，响应都包括应用程序名称、Amazon Resource ARN ame Resource Name Manager 的状态。如果响应返回的`HasMoreApplications`值为 true，则可以通过在请求正文`ExclusiveStartApplicationName`中添加来发送另一个请求，并将该值设置为先前响应中的最后一个应用程序名称。

如果您想了解有关特定应用程序的详细信息，请使用DescribeApplication。

此操作需要执行 `kinesisanalytics:ListApplications` 操作的权限。

请求语法

```
{
   "ExclusiveStartApplicationName": "string",
   "Limit": number
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ExclusiveStartApplicationName (p. 237)

列表开头的应用程序的名称。使用分页检索列表时，不需要在第一个请求中指定此参数。但是，在后续请求中，您可以添加先前响应中的最后一个应用程序名称以获取下一页应用程序。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：否

Limit (p. 237)

应用程序的最大数量。

类型：整数

有效范围：最小值为 1。最大值为 50。

必需：否

响应语法

```
{
   "ApplicationSummaries": [
      {
         "ApplicationARN": "string",
         "ApplicationName": "string",
```

```
        "ApplicationStatus": "string"
      }
   ],
   "HasMoreApplications": boolean
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

ApplicationSummaries (p. 237)

ApplicationSummary 对象的列表。

类型：ApplicationSummary (p. 257) 对象数组

HasMoreApplications (p. 237)

如果还有更多应用程序需要检索，则返回 true。

类型：布尔值

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ListTagsForResource

检索分配给应用程序的键值标签列表。有关更多信息，请参阅使用标签。

请求语法

```
{
   "ResourceARN": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ResourceARN (p. 239)

要检索其标签的应用程序的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：arn:.*

必需：是

响应语法

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

Tags (p. 239)

分配给应用程序的键值标签。

类型：Tag (p. 314) 对象数组

数组成员：最少 1 项。最多 200 项。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

StartApplication

> Note
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

启动指定的Amazon Kinesis Analytics 应用程序。创建应用程序后，必须专门调用此操作才能启动应用程序。

应用程序启动后，它开始使用输入数据，对其进行处理，并将输出写入配置的目标。

应用程序状态必须READY为，您才能启动应用程序。您可以在控制台中或使用DescribeApplication操作获取应用程序状态。

启动应用程序后，可以通过调用StopApplication操作来阻止应用程序处理输入。

此操作需要执行 `kinesisanalytics:StartApplication` 操作的权限。

请求语法

```
{
   "ApplicationName": "string",
   "InputConfigurations": [
      {
         "Id": "string",
         "InputStartingPositionConfiguration": {
            "InputStartingPosition": "string"
         }
      }
   ]
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 241)

应用程序名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

InputConfigurations (p. 241)

按 ID 标识应用程序开始使用的特定输入。Amazon Kinesis Analytics 开始读取与输入相关的流媒体源。您还可以指定希望 Amazon Kinesis Analytics 在流媒体源中开始阅读的位置。

类型：InputConfiguration (p. 266) 对象数组

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

InvalidApplicationConfigurationException

用户提供的应用程序的配置无效。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

StopApplication

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

停止应用程序处理输入数据。只有当应用程序处于运行状态时，才能将其停止。您可以使用DescribeApplication操作来找到应用程序状态。应用程序停止后，Amazon Kinesis Analytics 停止从输入中读取数据，应用程序停止处理数据，并且不会向目标写入任何输出。

此操作需要执行 `kinesisanalytics:StopApplication` 操作的权限。

请求语法

```
{
   "ApplicationName": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 243)

要停止的正在运行的应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

TagResource

向 Kinesis Analytics 应用程序添加一个或多个键值标签。请注意，应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。有关更多信息，请参阅使用标记。

请求语法

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

请求参数

请求接受采用 JSON 格式的以下数据。

ResourceARN (p. 245)

分配标签的应用程序的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

Tags (p. 245)

分配给应用程序的键值标签。

类型：Tag (p. 314) 对象数组

数组成员：最少 1 项。最多 200 项。

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

TooManyTagsException

创建的应用程序的标签过多，或者添加到应用程序的标签过多。请注意，应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

UntagResource

从 Kinesis Analytics 中删除一个或多个标签。有关更多信息，请参阅使用标签。

请求语法

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ResourceARN (p. 247)

要从中删除标签的 Kinesis Analytics 的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

TagKeys (p. 247)

要从指定应用程序中删除的标签键列表。

类型：字符串数组

数组成员：最少 1 项。最多 20项。

长度限制：最小长度为 1。最大长度为 128。

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

TooManyTagsException

创建的应用程序的标签过多，或者添加到应用程序的标签过多。请注意，应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

UpdateApplication

> **Note**
>
> 本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

更新现有的Amazon Kinesis Analytics 应用程序。使用此 API，您可以更新应用程序代码、输入配置和输出配置。

请注意，`CurrentApplicationVersionId`每次您更新应用程序时，Amazon Kinesis Analytics 都会更新。

此操作需要 `kinesisanalytics:UpdateApplication` 操作权限。

请求语法

```
{
   "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
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ApplicationName (p. 249)

要更新的Amazon Kinesis Analytics 应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：[a-zA-Z0-9_.-]+

必需：是

ApplicationUpdate (p. 249)

描述应用程序更新。

类型：ApplicationUpdate (p. 258) 对象

必需：是

CurrentApplicationVersionId (p. 249)

当前应用程序版本 ID。您可以使用DescribeApplication操作来获取此值。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

CodeValidationException

用户提供的应用程序代码（查询）无效。这可能是一个简单的语法错误。

HTTP 状态代码：400

ConcurrentModificationException

由于并行修改应用程序而引发异常。例如，两个人试图同时编辑同一个应用程序。

HTTP 状态代码：400

InvalidArgumentException

指定的输入参数值无效。

HTTP 状态代码：400

ResourceInUseException

应用程序不可用于此操作。

HTTP 状态代码：400

ResourceNotFoundException

找不到指定的应用程序。

HTTP 状态代码：400

UnsupportedOperationException

由于不支持指定的参数或指定的资源对此操作无效，请求被拒绝。

HTTP 状态代码：400

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- Amazon 命令行界面
- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- Amazon适用于 SDK JavaScript
- 适用于 PHP V3 的 Amazon SDK
- 适用于 Python 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

数据类型

支持以下数据类型：

ApplicationDetail

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

提供应用程序的描述，包括应用程序 Amazon 资源名称 (ARN)、状态、最新版本以及输入和输出配置。

目录

ApplicationARN

应用程序的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

ApplicationCode

返回您为对应用程序中的任何应用程序内流执行数据分析而提供的应用程序代码。

类型：字符串

长度约束：最小长度为 0。最大长度为 102400。

必需：否

ApplicationDescription

关于应用程序的描述。

类型：字符串

长度约束：最小长度为 0。长度上限为 1024。

必需：否

ApplicationName

应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

ApplicationStatus

应用程序的状态。

类型：字符串

有效值：`DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

必需：是

ApplicationVersionId

提供当前的应用程序版本。

类型：长整型

有效范围：最小值为 1。最大值为 999999999。

必需：是

CloudWatchLoggingOptionDescriptions

描述配置为接收应用程序消息 CloudWatch 的日志流。有关在 Amazon Kinesis Analytics 应用程序中使用 CloudWatch 日志流的更多信息，请参阅使用亚马逊 CloudWatch 日志。

类型：CloudWatchLoggingOptionDescription (p. 260) 对象数组

必需：否

CreateTimestamp

创建应用程序版本时的时间戳。

类型：Timestamp

必需：否

InputDescriptions

描述应用程序的输入配置。有关更多信息，请参阅配置应用程序输入。

类型：InputDescription (p. 267) 对象数组

必需：否

LastUpdateTimestamp

上次更新应用程序时的时间戳。

类型：Timestamp

必需：否

OutputDescriptions

描述应用程序输出配置。有关更多信息，请参阅配置应用程序输出。

类型：OutputDescription (p. 300) 对象数组

必需：否

ReferenceDataSourceDescriptions

描述为应用程序配置的参考数据源。有关更多信息，请参阅配置应用程序输入。

类型：ReferenceDataSourceDescription (p. 307) 对象数组

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK

- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ApplicationSummary

Note

本文档适用于 Amazon Kinesis Data Analytics API 版本 1，该版本仅支持 SQL 应用程序。版本 2 的 API 支持 SQL 和 Java 应用程序。有关版本 2 的更多信息，请参阅 Amazon Kinesis Data Analytics API V2 文档。

提供应用程序摘要信息，包括应用程序的 Amazon 资源名称（ARN）、名称和状态。

目录

ApplicationARN

应用程序的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

ApplicationName

应用程序的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

模式：`[a-zA-Z0-9_.-]+`

必需：是

ApplicationStatus

应用程序的状态。

类型：字符串

有效值：`DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ApplicationUpdate

描述要应用于现有Amazon Kinesis Analytics 应用程序的更新。

目录

ApplicationCodeUpdate

描述应用程序代码更新。

类型：字符串

长度约束：最小长度为 0。最大长度为 102400。

必需：否

CloudWatchLoggingOptionUpdates

描述应用程序 CloudWatch 记录选项更新。

类型：CloudWatchLoggingOptionUpdate (p. 261) 对象数组

必需：否

InputUpdates

描述应用程序输入配置更新。

类型：InputUpdate (p. 279) 对象数组

必需：否

OutputUpdates

描述应用程序输出配置更新。

类型：OutputUpdate (p. 302) 对象数组

必需：否

ReferenceDataSourceUpdates

描述应用程序参考数据源更新。

类型：ReferenceDataSourceUpdate (p. 308) 对象数组

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

CloudWatchLoggingOption

提供 CloudWatch 日志记录选项的描述，包括日志流 Amazon 资源名称（ARN）和角色 ARN。

目录

LogStreamARN

用于接收应用程序消息 CloudWatch 的日志 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

用于发送应用程序消息的角色的 IAM ARN。注意：要向写入应用程序消息 CloudWatch，所使用的 IAM 角色必须启用`PutLogEvents`策略操作。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

CloudWatchLoggingOptionDescription

CloudWatch 记录选项的描述。

目录

CloudWatchLoggingOptionId

CloudWatch 记录选项描述的 ID。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：否

LogStreamARN

用于接收应用程序消息 CloudWatch 的日志 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

用于发送应用程序消息的角色的 IAM ARN。注意：要向写入应用程序消息 CloudWatch，所使用的 IAM 角色必须启用`PutLogEvents`策略操作。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

CloudWatchLoggingOptionUpdate

描述 CloudWatch 记录选项更新。

目录

CloudWatchLoggingOptionId

要更新的 CloudWatch 日志记录选项的 ID

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

LogStreamARNUpdate

用于接收应用程序消息 CloudWatch 的日志 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

用于发送应用程序消息的角色的 IAM ARN。注意：要向写入应用程序消息 CloudWatch，所使用的 IAM 角色必须启用`PutLogEvents`策略操作。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

CSVMappingParameters

提供当记录格式使用带分隔符的格式（如 CSV）时的其他映射信息。例如，以下示例记录使用 CSV 格式，其中记录使用“\n”作为行分隔符，使用逗号（“,”）作为列分隔符：

```
"name1", "address1"

"name2", "address2"
```

目录

RecordColumnDelimiter

列分隔符。例如，在 CSV 格式中，逗号（“,”）是典型列分隔符。

类型：字符串

长度限制：最小长度为 1。

必需：是

RecordRowDelimiter

行分隔符。例如，在 CSV 格式中，“\n”是典型行分隔符。

类型：字符串

长度限制：最小长度为 1。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

DestinationSchema

描述将记录写入目标时的数据格式。有关更多信息，请参阅配置应用程序输出。

目录

RecordFormatType

指定输出流上的记录格式。

类型：字符串

有效值：`JSON | CSV`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

Input

配置应用程序输入时，请指定流式源、创建的应用程序内部流名称以及二者之间的映射。有关更多信息，请参阅配置应用程序输入。

目录

InputParallelism

描述要创建的应用程序内部流的数量。

您的源中的数据将路由到这些应用程序内部输入流。

（请参阅配置应用程序输入。）

类型：InputParallelism (p. 272) 对象

必需：否

InputProcessingConfiguration

InputProcessingConfiguration用于输入。在应用程序的 SQL 代码执行之前，输入处理器会在从流收到记录时转换记录。目前，唯一可用的输入处理配置为 InputLambdaProcessor。

类型：InputProcessingConfiguration (p. 274) 对象

必需：否

InputSchema

描述流式源中的数据的格式以及每个数据元素映射到所创建应用程序内部流中的相应列的方式。

还用于描述引用数据来源的格式。

类型：SourceSchema (p. 313) 对象

必需：是

KinesisFirehoseInput

如果流式源是一个 Amazon Kinesis Firehose 传输流，则标识该传输流的 ARN 和可让 Amazon Kinesis Analytics 代表您访问该流的 IAM 角色。

注意：需要 `KinesisStreamsInput` 或 `KinesisFirehoseInput` 之一。

类型：KinesisFirehoseInput (p. 282) 对象

必需：否

KinesisStreamsInput

如果流式源是一个 Amazon Kinesis 流，则标识该传输流的 Amazon 资源名称（ARN）和可让 Amazon Kinesis Analytics 代表您访问该流的 IAM 角色。

注意：需要 `KinesisStreamsInput` 或 `KinesisFirehoseInput` 之一。

类型：KinesisStreamsInput (p. 288) 对象

必需：否

NamePrefix

创建应用程序内部流时要使用的名称前缀。假设您指定了前缀“MyInApplicationStream”。Amazon Kinesis Analytics 随后创建一个或多个（根据您指定的`InputParallelism`计数）应用程序内部流，其

名称分别为 “MyInApplicationStream_001”、“Amazon Kinesis AnalyMyInApplicationStream tics”，以此类推。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputConfiguration

启动应用程序时，需要提供此配置，用于标识输入源和输入源中您希望应用程序开始处理记录的点。

目录

Id

输入源 ID。您可以通过调用该DescribeApplication操作来获取此 ID。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

InputStartingPositionConfiguration

您希望应用程序开始处理来自流媒体源的记录的时刻。

类型：InputStartingPositionConfiguration (p. 278) 对象

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputDescription

描述应用程序的输入配置。有关更多信息，请参阅配置应用程序输入。

目录

InAppStreamNames

返回映射到流源的应用程序内流名称。

类型：字符串数组

长度限制：最小长度为 1。最大长度为 32。

必需：否

InputId

与应用程序输入关联的输入 ID。这是Amazon Kinesis Analytics 为您添加到应用程序的每个输入配置分配的 ID。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：否

InputParallelism

描述配置的并行度（映射到流式源的应用程序内部流的数量）。

类型：InputParallelism (p. 272) 对象

必需：否

InputProcessingConfigurationDescription

在应用程序代码运行之前，在此输入中的记录上执行的预处理器的描述。

类型：InputProcessingConfigurationDescription (p. 275) 对象

必需：否

InputSchema

描述流式源中的数据的格式以及每个数据元素映射到所创建应用程序内部流中的相应列的方式。

类型：SourceSchema (p. 313) 对象

必需：否

InputStartingPositionConfiguration

将应用程序配置为从输入流中读取数据的点。

类型：InputStartingPositionConfiguration (p. 278) 对象

必需：否

KinesisFirehoseInputDescription

如果将 Amazon Kinesis Firehose 传输流配置为流式源，则提供传输流的 ARN 和可让 Amazon Kinesis Analytics 代表您访问该流的 IAM 角色。

类型：KinesisFirehoseInputDescription (p. 283) 对象

必需：否

KinesisStreamsInputDescription

如果将 Amazon Kinesis 流式源配置为流式源，则提供可让 Amazon Kinesis 流的 Amazon 资源名称（ARN）和可让 Amazon Kinesis Analytics 代表您访问该流的 IAM 角色。

类型：KinesisStreamsInputDescription (p. 289) 对象

必需：否

NamePrefix

应用程序内名称前缀。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputLambdaProcessor

一个对象，其中包含用于预处理流中记录的 Amazon Lambda 函数的 Amazon 资源名称（ARN），以及用于访问 Amazon Lambda 函数的 IAM 角色的 ARN。

目录

ResourceARN

用于处理流中记录的 Amazon Lambda 函数的 ARN。

Note

要指定相比最新版本较早的 Lambda 函数版本，请在 Lambda 函数 ARN 中包括 Lambda 函数版本。有关 Lambda ARN 的更多信息，请参阅示例 ARN：Amazon Lambda

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

用于访问 Amazon Lambda 函数的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputLambdaProcessorDescription

一个对象，其中包含用于预处理流中记录的 AmazonLambda 函数的 Amazon 资源名称（ARN），以及用于访问Amazon Lambda 表达式的 IAM 角色的 ARN。

目录

ResourceARN

用于预处理流中记录的 AmazonLambda 函数的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN

用于访问 Amazon Lambda 函数的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputLambdaProcessorUpdate

InputLambdaProcessor表示对用于预处理流中记录的更新。

目录

ResourceARNUpdate

用于预处理流中的记录的 AmazonLambda 函数的 Amazon 资源名称 (ARN)。

Note

要指定相比最新版本较早的 Lambda 函数版本，请在 Lambda 函数 ARN 中包括 Lambda 函数版本。有关 Lambda ARN 的更多信息，请参阅示例 ARN：Amazon Lambda

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

用于访问Amazon Lambda 函数的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputParallelism

描述要为指定流式源创建的应用程序内部流的数量。有关并行的信息，请参阅配置应用程序输入。

目录

Count

要创建的应用程序内部流的数量。有关更多信息，请参阅限制。

类型：整数

有效范围：最小值为 1。最大值为 64。

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputParallelismUpdate

提供并行度计数的更新。

目录

CountUpdate

要为指定流源创建的应用程序内部流的数量。

类型：整数

有效范围：最小值为 1。最大值为 64。

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputProcessingConfiguration

提供处理器的描述，该处理器用于在您的应用程序代码处理流中记录之前预处理这些记录。目前，唯一可用的输入处理器为 Amazon Lambda。

目录

InputLambdaProcessor

用于InputLambdaProcessor在您的应用程序代码处理流中记录之前预处理这些记录。

类型：InputLambdaProcessor (p. 269) 对象

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputProcessingConfigurationDescription

提供有关输入处理器的配置信息。目前，唯一可用的输入处理器为 Amazon Lambda。

目录

InputLambdaProcessorDescription

提供有关关联的配置信息InputLambdaProcessorDescription。

类型：InputLambdaProcessorDescription (p. 270) 对象

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputProcessingConfigurationUpdate

描述对的更新InputProcessingConfiguration。

目录

InputLambdaProcessorUpdate

提供更新信息InputLambdaProcessor。

类型：InputLambdaProcessorUpdate (p. 271) 对象

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputSchemaUpdate

描述应用程序输入架构的更新。

目录

RecordColumnUpdates

RecordColumn 对象的列表。每个对象描述流式源元素到应用程序内部流中相应列的映射。

类型：RecordColumn (p. 304) 对象数组

数组成员：最少 1 项。最多 1000 项。

必需：否

RecordEncodingUpdate

指定流式源中的记录的编码。例如，UTF-8。

类型：字符串

模式：UTF-8

必需：否

RecordFormatUpdate

指定流式源上的记录的格式。

类型：RecordFormat (p. 305) 对象

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputStartingPositionConfiguration

描述应用程序从流源读取数据的时刻。

目录

InputStartingPosition

直播中的起始位置。

- NOW-在流式传输中的最新记录之后立即开始阅读，从客户发布的请求时间戳开始。
- TRIM_HORIZON-从直播中最后一条未修剪的记录开始阅读，这是直播中最古老的记录。此选项不适用于Amazon Kinesis Firehose 配送流。
- LAST_STOPPED_POINT-从应用程序上次停止读取的位置恢复读取。

类型：字符串

有效值：NOW | TRIM_HORIZON | LAST_STOPPED_POINT

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

InputUpdate

描述对特定输入配置（由应用程序标识）InputId的更新。

目录

InputId

要更新的应用程序输入的输入 ID。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：[a-zA-Z0-9_.-]+

必需：是

InputParallelismUpdate

描述并行度更新（Amazon Kinesis Analytics 为特定流媒体源创建的应用程序内流的数量）。

类型：InputParallelismUpdate (p. 273) 对象

必需：否

InputProcessingConfigurationUpdate

描述输入处理配置的更新。

类型：InputProcessingConfigurationUpdate (p. 276) 对象

必需：否

InputSchemaUpdate

描述流式源上的数据格式，以及流式源中的记录元素如何映射到所创建应用程序内部流的列。

类型：InputSchemaUpdate (p. 277) 对象

必需：否

KinesisFirehoseInputUpdate

如果 Amazon Kinesis Firehose 交付流是要更新的流媒体源，则提供更新的直播 ARN 和 IAM 角色 ARN。

类型：KinesisFirehoseInputUpdate (p. 284) 对象

必需：否

KinesisStreamsInputUpdate

如果 Amazon Kinesis 流是要更新的流式源，则提供更新的流的 Amazon 资源名称（ARN）和 IAM 角色 ARN。

类型：KinesisStreamsInputUpdate (p. 290) 对象

必需：否

NamePrefixUpdate

Amazon Kinesis Analytics 为特定流媒体源创建的应用程序内直播的名称前缀。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

JSONMappingParameters

提供当 JSON 是流式源上的记录格式时的其他映射信息。

目录

RecordRowPath

指向包含记录的顶层父级的路径。

类型：字符串

长度限制：最小长度为 1。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisFirehoseInput

将 Amazon Kinesis Firehose 传输流标识为流式源。您提供传输流的 Amazon 资源名称（ARN）和可让 Amazon Kinesis Analytics 代表您访问该流的 IAM 角色 ARN。

目录

ResourceARN

输入传输流的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。您需要确保该角色有必要的权限以访问流。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisFirehoseInputDescription

描述在应用程序输入配置中配置为流媒体源的 Amazon Kinesis Firehose 传输流。

目录

ResourceARN

可由 Amazon Kinesis Firehose 传输流的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN

可由 Amazon Kinesis Analytics 代入以访问流的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisFirehoseInputUpdate

更新应用程序输入配置时，将 Amazon Kinesis Firehose 传输流式传输流标识为流式源。

目录

ResourceARNUpdate

传输流式传输流的 Amazon Kinesis Firehose 传输流的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisFirehoseOutput

配置应用程序输出时，将 Amazon Kinesis Firehose 传输流标识为目标。您提供流的 Amazon 资源名称（ARN）和可让 Amazon Kinesis Analytics 代表您写入流的 IAM 角色。

目录

ResourceARN

要写入的目标 Amazon Kinesis Firehose 传输流的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

可由 Amazon Kinesis Analytics 代入以代表您对目标流进行写入的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisFirehoseOutputDescription

对于应用程序输出，描述配置为目标的 Amazon Kinesis Firehose。

目录

ResourceARN

亚马逊 Kinesis Firehose 传输流的 Amazon Resource Name (ARN) 。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN

可由 Amazon Kinesis Analytics 代入以访问流的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisFirehoseOutputUpdate

使用UpdateApplication操作更新输出配置时，提供有关配置为目标的 Amazon Kinesis Firehose 传输流的信息。

目录

ResourceARNUpdate

要写入的 Amazon Kinesis Firehose 传输流的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisStreamsInput

将 Amazon Kinesis 流标识为流式源。您提供流的 Amazon 资源名称（ARN）和可让 Amazon Kinesis Analytics 代表您访问该流的 IAM 角色 ARN。

目录

ResourceARN

要读取的输入 Amazon Kinesis 流的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisStreamsInputDescription

描述在应用程序输入配置中配置为流媒体源的 Amazon Kinesis 流。

目录

ResourceARN

可由 Amazon Kinesis 流的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN

可由 Amazon Kinesis Analytics 代入以访问流的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisStreamsInputUpdate

更新应用程序输入配置时，提供流式源 Amazon Kinesis 流的 Amazon Kinesis 流式源。

目录

ResourceARNUpdate

要读取的输入 Amazon Kinesis 流的 Amazon Resource Name (ARN)

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisStreamsOutput

在配置应用程序输出时，将 Amazon Kinesis 流标识为目标。您提供流的 Amazon 资源名称（ARN），以及用于让 Amazon Kinesis Analytics 代表您写入流的 IAM 角色 ARN。

目录

ResourceARN

要写入的目标 Amazon Kinesis 流的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

可由 Amazon Kinesis Analytics 代入以代表您对目标流进行写入的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisStreamsOutputDescription

对于应用程序输出，描述了配置为目标的 Amazon Kinesis 流。

目录

ResourceARN

可由 Amazon Kinesis 流的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN

可由 Amazon Kinesis Analytics 代入以访问流的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

KinesisStreamsOutputUpdate

使用UpdateApplication操作更新输出配置时，提供有关配置为目标的 Amazon Kinesis 流的信息。

目录

ResourceARNUpdate

要使用其写入输出的 Amazon Kinesis 流的 Amazon Resource Name (ARN)。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

可由 Amazon Kinesis Analytics 代入以代表您访问流的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

LambdaOutput

在配置应用程序输出时，将 Amazon Lambda 函数标识为目标。您提供函数的 Amazon 资源名称（ARN），以及用于让 Amazon Kinesis Analytics 代表您写入函数的 IAM 角色 ARN。

目录

ResourceARN

要向其写入的目标 Lambda 函数的 Amazon 资源名称（ARN）。

Note

要指定相比最新版本较早的 Lambda 函数版本，请在 Lambda 函数 ARN 中包括 Lambda 函数版本。有关 Lambda ARN 的更多信息，请参阅示例 ARN：Amazon Lambda

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

RoleARN

可由 Amazon Kinesis Analytics 代入以代表您对目标函数进行写入的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

LambdaOutputDescription

对于应用程序输出，描述了配置为其目标的Amazon Lambda 函数。

目录

ResourceARN

目标 Lambda 函数的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARN

可由 Amazon Kinesis Analytics 代入以对目标函数进行写入的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

LambdaOutputUpdate

使用UpdateApplication操作更新输出配置时，提供有关配置为目标的Amazon Lambda 函数的信息。

目录

ResourceARNUpdate

目标 Lambda 函数的 Amazon 资源名称（ARN）。

Note

要指定相比最新版本较早的 Lambda 函数版本，请在 Lambda 函数 ARN 中包括 Lambda 函数版本。有关 Lambda ARN 的更多信息，请参阅示例 ARN：Amazon Lambda

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

RoleARNUpdate

可由 Amazon Kinesis Analytics 代入以代表您对目标函数进行写入的 IAM 角色的 ARN。您需要向此角色授予必需的权限。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

MappingParameters

如果在创建或更新应用程序时配置应用程序输入，请提供特定于流式源上的记录格式（如 JSON、CSV 或由某个分隔符分隔的记录字段）的其他映射信息。

目录

CSVMappingParameters

提供当记录格式使用带分隔符的格式（如 CSV）时的其他映射信息。

类型：CSVMappingParameters (p. 262) 对象

必需：否

JSONMappingParameters

提供当 JSON 是流式源上的记录格式时的其他映射信息。

类型：JSONMappingParameters (p. 281) 对象

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

Output

描述应用程序输出配置，在其中您标识应用程序内部流以及希望应用程序内部流数据写入到的目标。目标可以是 Amazon Kinesis 流或 Amazon Kinesis Firehose 传输流。

有关应用程序可以写入的目标数的限制以及其他限制，请参阅限制。

目录

DestinationSchema

描述将记录写入目标时的数据格式。有关更多信息，请参阅配置应用程序输出。

类型：DestinationSchema (p. 263) 对象

必需：是

KinesisFirehoseOutput

将 Amazon Kinesis Firehose 传输流标识为目标。

类型：KinesisFirehoseOutput (p. 285) 对象

必需：否

KinesisStreamsOutput

将 Amazon Kinesis 流标识为目标。

类型：KinesisStreamsOutput (p. 291) 对象

必需：否

LambdaOutput

将 Amazon Lambda 函数标识为目标。

类型：LambdaOutput (p. 294) 对象

必需：否

Name

应用程序内部流的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

OutputDescription

描述应用程序输出配置，其中包括应用程序内流名称和写入流数据的目的地。目标可以是 Amazon Kinesis 流或 Amazon Kinesis Firehose 传输流。

目录

DestinationSchema

用于将数据写入目标的数据格式。

类型：DestinationSchema (p. 263) 对象

必需：否

KinesisFirehoseOutputDescription

描述配置为写入输出目标的 Amazon Kinesis Firehose 传输流。

类型：KinesisFirehoseOutputDescription (p. 286) 对象

必需：否

KinesisStreamsOutputDescription

描述配置为写入输出目的地的 Amazon Kinesis 流。

类型：KinesisStreamsOutputDescription (p. 292) 对象

必需：否

LambdaOutputDescription

描述配置为写入输出目标的Amazon Lambda 函数。

类型：LambdaOutputDescription (p. 295) 对象

必需：否

Name

配置为输出的应用程序内部流的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：否

OutputId

输出配置的唯一标识符。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

OutputUpdate

描述对所标识的输出配置的更新`OutputId`。

目录

DestinationSchemaUpdate

描述将记录写入目标时的数据格式。有关更多信息，请参阅配置应用程序输出。

类型：DestinationSchema (p. 263) 对象

必需：否

KinesisFirehoseOutputUpdate

将 Amazon Kinesis Firehose 传输流标识为输出的对象。

类型：KinesisFirehoseOutputUpdate (p. 287) 对象

必需：否

KinesisStreamsOutputUpdate

将 Amazon Kinesis 流标识为输出的对象。

类型：KinesisStreamsOutputUpdate (p. 293) 对象

必需：否

LambdaOutputUpdate

将 LAmazon ambda 函数标识为输出对象。

类型：LambdaOutputUpdate (p. 296) 对象

必需：否

NameUpdate

如果要为此输出配置指定不同的应用程序内流，请使用此字段指定新的应用程序内流名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：否

OutputId

标识要更新的特定输出配置。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

RecordColumn

描述流式源中的每个数据元素到应用程序内部流中的相应列的映射。

还用于描述引用数据来源的格式。

目录

Mapping

对流输入中数据元素的引用，或者引用数据来源。如果是，则此元素为必填元素JSON。RecordFormatType

类型：字符串

必需：否

Name

在应用程序内部输入流或引用表中创建的列的名称。

类型：字符串

必需：是

SqlType

在应用程序内部输入流或引用表中创建的列的类型。

类型：字符串

长度限制：最小长度为 1。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

RecordFormat

描述应该应用的记录格式和相关映射信息，以便在流中架构化记录。

目录

MappingParameters

如果在创建或更新应用程序时配置应用程序输入，请提供特定于流式源上的记录格式（如 JSON、CSV 或由某个分隔符分隔的记录字段）的其他映射信息。

类型：MappingParameters (p. 297) 对象

必需：否

RecordFormatType

记录格式的类型。

类型：字符串

有效值：`JSON | CSV`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ReferenceDataSource

描述引用数据来源，方式是提供源信息（S3 桶名称和对象键名称）、创建的结果应用程序内部表名称以及要将 Amazon S3 对象中的对象素映射到应用程序内部表的所需架构。

目录

ReferenceSchema

描述流式源中的数据的格式，以及每个数据元素如何映射到在应用程序内部流中创建的相应列。

类型：SourceSchema (p. 313) 对象

必需：是

S3ReferenceDataSource

标识 S3 桶和包含引用数据的对象。此外，还标识可由 Amazon Kinesis Analytics 代入以代表您读取此对象的 IAM 角色。Amazon Kinesis Analytics 应用程序仅加载一次引用数据。如果数据更改，您可以调用 `UpdateApplication` 操作来触发将数据重新加载到应用程序。

类型：S3ReferenceDataSource (p. 310) 对象

必需：否

TableName

要创建的应用程序内部表的名称。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ReferenceDataSourceDescription

描述为应用程序配置的参考数据源。

目录

ReferenceId

引用数据源的 ID。这是 Amazon Kinesis Analytics 在您使用AddApplicationReferenceDataSource操作将参考数据源添加到应用程序时分配的 ID。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

ReferenceSchema

描述流式源中的数据的格式，以及每个数据元素如何映射到在应用程序内部流中创建的相应列。

类型：SourceSchema (p. 313) 对象

必需：否

S3ReferenceDataSourceDescription

提供 S3 桶名称，即包含引用数据的对象键名称。它还提供了 IAM 角色的 Amazon 资源名称（ARN），Amazon Kinesis Analytics 可以代为读取 Amazon S3 对象并填充应用程序引用表。

类型：S3ReferenceDataSourceDescription (p. 311) 对象

必需：是

TableName

由特定参考数据源配置创建的应用程序内表名。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

ReferenceDataSourceUpdate

更新应用程序引用数据来源配置时，此对象提供所有更新的值（例如源桶名称和对象键名称）、创建的应用程序内部表名称以及要将 Amazon S3 对象中的对象素映射到对象内部的更新映射信息创建的应用程序参考表。

目录

ReferenceId

正在更新的引用数据来源的 ID。您可以使用DescribeApplication操作来获取此值。

类型：字符串

长度限制：最小长度为 1。长度上限为 50。

模式：`[a-zA-Z0-9_.-]+`

必需：是

ReferenceSchemaUpdate

描述流式源中的数据的格式，以及每个数据元素如何映射到在应用程序内部流中创建的相应列。

类型：SourceSchema (p. 313) 对象

必需：否

S3ReferenceDataSourceUpdate

描述 Amazon Kinesis Analytics 可以代替您读取 Amazon S3 对象并填充应用程序内参考表的 S3 存储桶名称、对象密钥名称和 IAM 角色。

类型：S3ReferenceDataSourceUpdate (p. 312) 对象

必需：否

TableNameUpdate

此更新创建的应用程序内表名。

类型：字符串

长度限制：最小长度为 1。最大长度为 32。

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

S3Configuration

提供 Amazon S3 数据源的描述，包括 S3 桶的 Amazon 资源名称（ARN）、用于访问存储桶的 IAM 角色的 ARN 以及包含数据的 Amazon S3 对象的名称。

目录

BucketARN

包含数据的 S3 桶的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

FileKey

包含数据的对象的名称。

类型：字符串

长度限制：最小长度为 1。长度上限为 1024。

必需：是

RoleARN

用于访问数据的角色的 IAM ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

S3ReferenceDataSource

标识 S3 桶和包含引用数据的对象。此外，还标识可由 Amazon Kinesis Analytics 代入以代表您读取此对象的 IAM 角色。

Amazon Kinesis Analytics 应用程序仅加载一次引用数据。如果数据更改，您可以调用 UpdateApplication 操作来触发将数据重新加载到应用程序。

目录

BucketARN

S3 桶的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

FileKey

包含引用数据的对象键名称。

类型：字符串

长度限制：最小长度为 1。长度上限为 1024。

必需：是

ReferenceRoleARN

可由此服务代入以代表您读取数据的 IAM 角色的 ARN。此角色必须具有对象上的 `s3:GetObject` 操作权限以及允许 Amazon Kinesis Analytics 服务委托人代入此角色的信任策略。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

S3ReferenceDataSourceDescription

提供存储引用数据的存储桶名称和对象键名称。

目录

BucketARN

S3 桶的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

FileKey

Amazon S3 ervice S3 S3

类型：字符串

长度限制：最小长度为 1。长度上限为 1024。

必需：是

ReferenceRoleARN

可ARN Amazon Kinesis Analytics 代入以代表您读取 Simple S3 对象以填入应用程序内引用表的 IAM 角色的 IAM 角色的 IAM 角色的 IAM 角色的 IA

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

S3ReferenceDataSourceUpdate

描述 Amazon Kinesis Analytics 可以代替您读取 Amazon S3 对象并填充应用程序内参考表的 S3 存储桶名称、对象密钥名称和 IAM 角色。

目录

BucketARNUpdate

S3 桶的 Amazon 资源名称（ARN）。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

FileKeyUpdate

对象键名称。

类型：字符串

长度限制：最小长度为 1。长度上限为 1024。

必需：否

ReferenceRoleARNUpdate

可由 Amazon Kinesis Analytics 代入以读取 Amazon S3 对象并填充应用程序内部对象的 IAM 角色的 ARN。

类型：字符串

长度限制：最小长度为 1。最大长度为 2048。

模式：`arn:.*`

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

SourceSchema

描述流式源中的数据的格式，以及每个数据元素如何映射到在应用程序内部流中创建的相应列。

目录

RecordColumns

`RecordColumn` 对象的列表。

类型：RecordColumn (p. 304) 对象数组

数组成员：最少 1 项。最多 1000 项。

必需：是

RecordEncoding

指定流式源中的记录的编码。例如，UTF-8。

类型：字符串

模式：`UTF-8`

必需：否

RecordFormat

指定流式源上的记录的格式。

类型：RecordFormat (p. 305) 对象

必需：是

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

Tag

您可以定义并分配给Amazon资源的键值对（该值是可选的）。如果您指定的标签已经存在，则该标签值将替换为您在请求中指定的值。请注意，应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。有关更多信息，请参阅使用标签。

目录

Key

键值标签的密钥。

类型：字符串

长度限制：最小长度为 1。最大长度为 128。

必需：是

Value

键值标签的值。 值是可选的。

类型：字符串

长度约束：最小长度为 0。长度上限为 256。

必需：否

另请参阅

有关在特定语言的 Amazon 软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java V2 的 Amazon SDK
- 适用于 Ruby V3 的 Amazon SDK

Amazon Kinesis Data Analytics 的文档历史记录

下表描述了自 Amazon Kinesis Data Analytics 上一次发布以来对文档所做的重要更改。

- API 版本：2015-08-14
- 最近文档更新时间：2019 年 5 月 8 日

更改	描述	日期
标记 Kinesis Data Analytics 应用程序	使用应用程序标记来确定每个应用程序的成本，控制访问，或用于用户定义的目的。有关更多信息，请参阅 使用标记 (p. 38)。	2019 年 5 月 8 日
记录 Kinesis Data Analytics API 调用Amazon CloudTrail	Amazon Kinesis Data Analytics 已与Amazon CloudTrail，提供用户、角色或执行操作的记录的服务AmazonKinesis Data Analytics 中的服务。有关更多信息，请参阅 使用 Amazon CloudTrail (p. 186)。	2019 年 3 月 22 日
Kinesis Data Analytics 在法兰克福区域中	Kinesis Analytics 现已在欧洲（法兰克福）区域推出。有关更多信息，请参阅 。和：Endpoints Kinesis Data Analytics.	2018 年 7 月 18 日
在控制台中使用引用数据	现在，您可以在控制台中使用应用程序引用数据。有关更多信息，请参阅 示例：向 Kinesis Data Analytics 应用程序添加参考数据 (p. 133)。	2018 年 7 月 13 日
窗口式查询示例	适用于窗口和聚合的示例应用程序 有关更多信息，请参阅 示例：窗口和聚合 (p. 121)。	2018 年 7 月 9 日
测试应用程序	测试对应用程序架构和代码的更改的指导。有关更多信息，请参阅 测试应用程序 (p. 193)。	2018 年 7 月 3 日
预处理数据示例应用程序	REGEX_LOG_PARSE、REGEX_REPLACE 和 DateTime 运算符的其他代码示例。有关更多信息，请参阅 示例：转换数据 (p. 99)。	2018 年 5 月 18 日
返回的行和 SQL 代码的大小增加	返回的行的大小限制增加为 512 KB，应用程序中 SQL 代码的大小限制增加为 100 KB。有关更多信息，请参阅 Limits (p. 189)。	2018 年 5 月 2 日

更改	描述	日期
使用 Java 和 .NET 的 Amazon Lambda 函数示例	创建 Lambda 函数以预处理记录或作为应用程序目标的代码示例。有关更多信息，请参阅创建 Lambda 函数以进行预处理 (p. 23)和为应用程序目标创建 Lambda 函数 (p. 34)。	2018 年 3 月 22 日
新的 HOTSPOTS 函数	查找和返回有关数据中相对密集的区域的信息。有关更多信息，请参阅 示例：检测流上的热点 (HOTSPOTS 函数) (p. 145)。	2018 年 3 月 19 日
Lambda 函数作为目标	将分析结果发送到作为目标的 Lambda 函数。有关更多信息，请参阅 使用 Lambda 函数作为输出 (p. 31)。	2017 年 12 月 20 日
新的 RANDOM_CUT_FOREST_WITH_EXPLANATION 函数	了解在数据流中哪些字段会产生异常评分。有关更多信息，请参阅 示例：检测数据异常和获取说明 (RANDOM_CUT_FOREST_WITH_EXPLANATION 函数) (p. 142)。	2017 年 11 月 2 日
静态数据上的架构发现	在 Amazon S3 存储桶中存储的静态数据运行架构查找。有关更多信息，请参阅 针对静态数据使用架构发现功能 (p. 16)。	2017 年 10 月 6 日
Lambda 预处理功能	在分析之前，使用 Amazon Lambda 预处理输入流中的记录。有关更多信息，请参阅 使用 Lambda 函数预处理数据 (p. 19)。	2017 年 10 月 6 日
自动扩展应用程序	使用自动扩展来自动增加应用程序的数据吞吐量。有关更多信息，请参阅 自动扩展应用程序以提高吞吐量 (p. 37)。	2017 年 9 月 13 日
多个应用程序内部输入流	通过多个应用程序内部流提高应用程序吞吐量。有关更多信息，请参阅 并行处理输入流以增加吞吐量 (p. 25)。	2017 年 6 月 29 日
使用指南Amazon Web Services Management Console对于 Kinesis Data Analytics	在 Kinesis Data Analytics 控制台中使用架构编辑器和 SQL 编辑器编辑推断架构和 SQL 代码。有关更多信息，请参阅 步骤 4：(可选) 使用控制台编辑架构和 SQL 代码 (p. 53)。	2017 年 4 月 7 日
公开发行版	公开发行版Amazon Kinesis Data Analytics 开发者指南.	2016 年 8 月 11 日
预览版	的预览版Amazon Kinesis Data Analytics 开发者指南.	2016 年 1 月 29 日

Amazon词汇表

有关最新Amazon术语，请参阅《Amazon一般参考》中的Amazon术语表。