aws

SQL 開發人員指南

適用於 SQL 應用程式的 Amazon Kinesis Data Analytics 開發人員指南

適用於 SQL 應用程式的 Amazon Kinesis Data Analytics 開發人員指南: SQL 開發人員指南

Table of Contents

針對新專案，我們建議您優先選擇新的 Managed Service for Apache Flink Studio，而非 Kinesis Data Analytics for SQL 應用程式。Managed Service for Apache Flink Studio 易於使用且具備進階分析功能，可讓您在幾分鐘內建置複雜的串流處理應用程式。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

什麼是 Amazon Kinesis Data Analytics for SQL 應用程式？

搭配 Amazon Kinesis Data Analytics for SQL 應用程式，您可以使用標準 SQL 來處理和分析串流資料。此服務可讓您針對串流來源快速撰寫和執行 SQL 程式碼，以執行時間序列分析、饋送即時儀表板，以及建立即時指標。

若要開始使用 Kinesis Data Analytics，您需要建立一個 Kinesis Data Analytics 應用程式，以持續讀取和處理串流資料。此服務支援從 Amazon Kinesis 資料串流和亞馬遜資料 Firehose 串流來源擷取資料。然後，您可以使用互動式編輯器撰寫 SQL 程式碼，並使用即時串流資料進行測試。您也可以設定希望 Kinesis Data Analytics 傳送結果的目的地。

Kinesis Data Analytics 支援 Amazon 資料 Firehose (Amazon S3、Amazon 紅移、亞馬遜 OpenSearch 服務和潑濺)，以及作為目的地的 Amazon Kinesis Data Streams。 AWS Lambda

什麼時候應該使用 Amazon Kinesis Data Analytics？

Amazon Kinesis Data Analytics 可讓您快速撰寫 SQL 程式碼，以近乎即時的速度持續讀取、處理和存放資料。對串流資料使用標準 SQL 查詢，您可以建構應用程式來轉換資料並提供洞見。以下是使用 Kinesis Data Analytics 的一些範例情境：

- 產生時間序列分析：您可以隨時間窗口計算指標，然後透過 Kinesis 資料交付串流將值串流至 Amazon S3 或 Amazon Redshift。
- 饋送即時儀表板：您可以向下游傳送彙總和已處理的串流資料結果，以饋送即時儀表板。
- 創建即時指標：您可以創建自定義指標和觸發器，以用於即時監控、通知和警報。

如需 Kinesis Data Analytics 支援之 SQL 語言元素的相關資訊，請參閱 Amazon Kinesis Data Analytics SQL 參考資料。

您是第一次使用 Amazon Kinesis Data Analytics 嗎？

若是第一次使用 Amazon Kinesis Data Analytics，建議您依序閱讀以下區段：

1. 請閱讀本指南的運作方式章節。本節介紹您可以用來建立 end-to-end 體驗的各種 Kinesis Data Analytics 元件。如需詳細資訊，請參閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式。

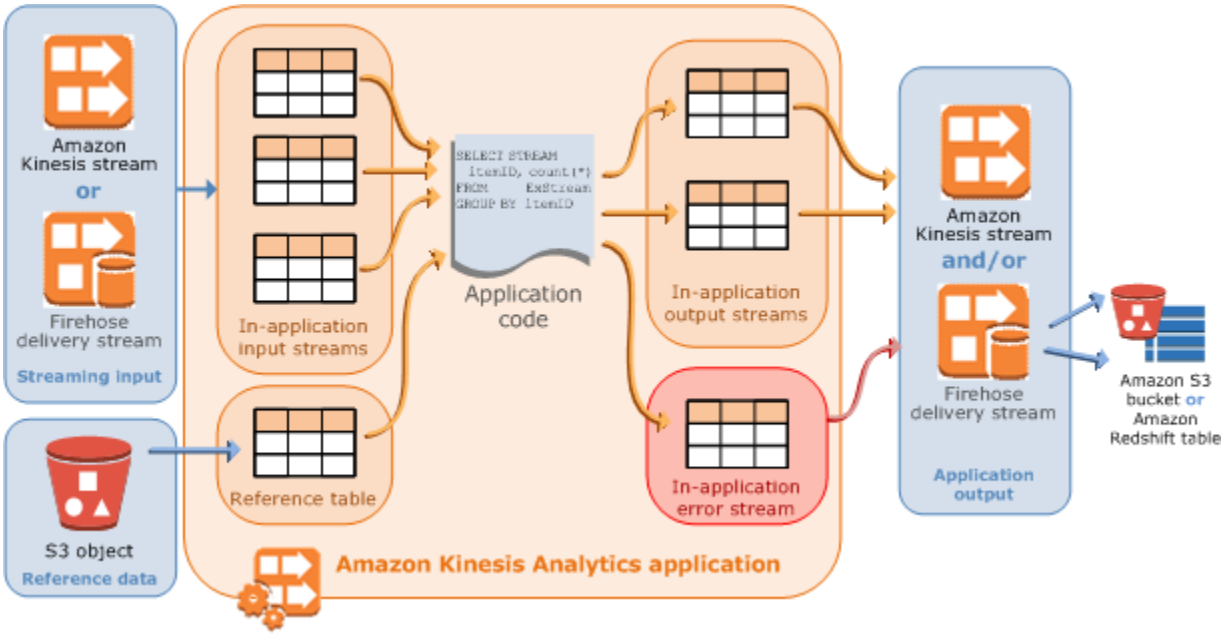
2. 嘗試入門練習。如需詳細資訊，請參閱 Amazon Kinesis Data Analytics for SQL 應用程式入門。
3. 探索串流 SQL 概念。如需詳細資訊，請參閱 串流 SQL 概念。
4. 嘗試其他範例 如需更多詳細資訊，請參閱 Kinesis Data Analytics for SQL 範例。

Amazon Kinesis Data Analytics for SQL 應用程式：運作方式

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

應用程式是 Amazon Kinesis Data Analytics 中的主要資源，您可以在帳戶中建立。您可以使用 AWS Management Console 或 Kinesis Data Analytics API 建立和管理應用程式。Kinesis Data Analytics 提供 API 操作來管理應用程式。如需 API 操作的清單，請參閱 動作。

Kinesis Data Analytics 應用程式會持續即時讀取和處理串流資料。您可以使用 SQL 撰寫應用程式程式碼，以處理傳入的串流資料並產生輸出。然後，Kinesis Data Analytics 會將輸出寫入設定的目的地。下圖說明典型的應用程式架構。



每個應用程式都有名稱、描述、版本 ID 和狀態。Amazon Kinesis Data Analytics 會在您第一次建立應用程式時指派一個版本 ID。此版本 ID 會在您更新任何應用程式組態時更新。舉例來說，如果您新增輸入組態、新增或刪除參考資料來源、新增或刪除輸出組態，或更新應用程式碼，Kinesis Data Analytics 會更新目前的應用程式版本 ID。Kinesis Data Analytics 也會維護應用程式建立和上次更新的時間戳記。

除了這些基本屬性之外，每個應用程式還包含以下內容：

- 輸入：應用程式的串流來源。您可以選取 Kinesis 資料串流或 Firehose 資料傳遞串流作為串流來源。在此輸入組態中，將串流來源映射到應用程式內輸入串流。應用程式內串流就像是持續更新的資料表，您可以在其上執行 SELECT 和 INSERT SQL 作業。在應用程式碼中，您可以建立其他應用程式內串流來儲存中繼查詢結果。

  在多個應用程式內輸入串流中，您可以選擇性地分割單一串流來源，以改善輸送量。如需詳細資訊，請參閱 限制 及 設定應用程式輸入。

  在每個應用程式串流中，Amazon Kinesis Data Analytics 會提供名為 時間戳記和 ROWTIME 欄 的時間戳記欄。您可以在基於時間的窗口查詢中使用此欄。如需詳細資訊，請參閱 窗口化查詢。

  您可以選擇性地設定參考資料來源，以豐富應用程式內的輸入資料串流。此舉會產生應用程式內參考資料表。您必須將參考資料作為物件存放在 S3 儲存貯體中。Amazon Kinesis Data Analytics 會在應用程式啟動時讀取 Amazon S3 物件，並建立應用程式內資料表。如需詳細資訊，請參閱 設定應用程式輸入。

- 應用程式碼：處理輸入並產生輸出的一系列 SQL 陳述式。您可以針對應用程式內串流和參考資料表撰寫 SQL 陳述式。您也可以撰寫 JOIN 查詢，以合併來自這兩個來源的資料。

  如需 Kinesis Data Analytics 支援之 SQL 語言元素的相關資訊，請參閱 Amazon Kinesis Data Analytics SQL 參考資料。

  以最簡單的形式來說，應用程式碼可以是單一 SQL 陳述式，可從串流輸入中選取，並將結果插入串流輸出。它也可以是一系列的 SQL 陳述式，其中一個的輸出會饋送至下一個 SQL 陳述式的輸入。此外，您可以撰寫應用程式碼，將輸入串流分割成多個串流。然後，您可以套用其他查詢來處理這些串流。如需詳細資訊，請參閱 應用程式碼。

- 輸出：在應用程式碼中，查詢結果會移至應用程式內串流。在應用程式碼中，您可以建立一或多個應用程式內串流來儲存中繼結果。然後，您可以選擇性地設定應用程式輸出，將資料保留在應用程式內串流，該串流會將應用程式的輸出 (也稱為應用程式內輸出串流) 保存在外部目的地。外部目標可以是 Firehose 傳送串流或 Kinesis 資料串流。請注意下列與這些目的地相關的資訊：
  - 您可以設定 Firehose 交付串流，將結果寫入 Amazon S3、Amazon Redshift 或 Amazon OpenSearch 服務 (服OpenSearch 務)。
  - 您也可以將應用程式輸出寫入自訂目的地，而不是 Amazon S3 或 Amazon Redshift。若要這樣做，您要在輸出組態中指定 Kinesis 資料串流作為目的地。然後，您可 AWS Lambda 以設定輪詢串流並叫用 Lambda 函數。Lambda 函數程式碼會接收串流資料作為輸入。在 Lambda 函數程式碼中，您可以將傳入資料寫入自訂目的地。如需詳細資訊，請參閱 AWS Lambda 搭配 Amazon Kinesis Data Analytics 使用。

  如需詳細資訊，請參閱 設定應用程式輸出。

此外，請注意下列事項：

- Amazon Kinesis Data Analytics 需要許可才能讀取串流來源的記錄，並將應用程式輸出寫入外部目的地。您可以透過 IAM 角色來授予這些許可。
- Amazon Kinesis Data Analytics 會為每個應用程式提供應用程式內錯誤串流。如果您的應用程式在處理特定記錄時發生問題 (例如，因為類型不符或延遲到達)，則該記錄會寫入錯誤串流。您可以設定應用程式輸出來引導 Kinesis Data Analytics，將錯誤串流資料保留到外部目的地，以供進一步評估。如需詳細資訊，請參閱 錯誤處理。
- Amazon Kinesis Data Analytics 可確保應用程式輸出記錄有寫入設定的目的地。即使您遇到應用程式中斷，它也會使用「至少一次」處理和交付模式。如需更多詳細資訊，請參閱 將應用程式輸出保存至外部目標的交付模型。

主題

- 設定應用程式輸入
- 應用程式碼
- 設定應用程式輸出

- 錯誤處理
- 自動擴展應用程式以增加輸送量
- 使用標記

設定應用程式輸入

您的 Amazon Kinesis Data Analytics 應用程式可以從單一串流來源接收輸入，並選擇性地使用一個參考資料來源。如需詳細資訊，請參閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式。本主題中的各節說明應用程式輸入來源。

主題

- 設定串流來源
- 配置參考來源
- 使用 JSONPath
- 將串流來源元素映射至 SQL 輸入資料欄
- 在串流資料上使用結構描述探索功能
- 在靜態資料上使用結構描述探索功能
- 使用 Lambda 函數預處理資料
- 平行化輸入串流以提高輸送量

設定串流來源

當您建立應用程式時，就要指定串流來源。您也可以在建立應用程式後修改輸入。Amazon Kinesis Data Analytics 支援您的應用程式採用下列串流來源：

- Kinesis 資料串流
- 一個 Firehose 投遞分流

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。搭配使用 Kinesis Data Analytics for SQL 應用程式與 `KinesisFirehoseInput` 的現有客戶，可以繼續使用

KinesisFirehoseInput，在使用 Kinesis Data Analytics 的現有帳戶中新增應用程式。如果您是現有客戶，並且希望使用 Kinesis Data Analytics for SQL 與 KinesisFirehoseInput 建立新帳戶，您可以使用增加服務限制額度表單來開立案例。如需詳細資訊，請參閱 AWS Support 中心。我們建議在推廣生產之前，務必測試任何新的應用程式。

> **Note**
>
> 如果 Kinesis 資料串流經過加密，Kinesis Data Analytics 就能順暢存取加密串流中的資料，無需進一步設定。Kinesis Data Analytics 不會儲存從 Kinesis 資料串流讀取的未加密資料。如需詳細資訊，請參閱什麼是 Kinesis Data Streams 伺服器端加密？。

Kinesis Data Analytics 會持續輪詢串流來源是否有新資料，並根據輸入組態將其導入應用程式內串流。

> **Note**
>
> 將 Kinesis 串流新增為應用程式的輸入不會影響串流中的資料。如果其他資源 (例如 Firehose 交付串流) 也存取了相同的 Kinesis 串流，則 Firehose 交付串流和 Kinesis Data Analytics 應用程式都會收到相同的資料。但是，輸送量和限流可能會受到影響。

應用程式碼會查詢應用程式內串流。作為輸入組態的一部分，您提供以下項目：

- 串流來源：您提供的串流 Amazon Resource Name (ARN) 和 IAM 角色，可讓 Kinesis Data Analytics 代您存取串流。
- 應用程式內串流名稱字首：當您啟動應用程式時，Kinesis Data Analytics 會建立指定的應用程式內串流。在應用程式碼中，您可以使用此名稱存取應用程式內串流。

  您可以選擇性地將串流來源映射至多個應用程式內串流。如需詳細資訊，請參閱 限制。*##### ##Amazon Kinesis Data Analytics ########################## _001### _002 ### _003*。根據預設，Kinesis Data Analytics 會將串流來源映射到一個名為*##* _001 的應用程式內串流。

  您在應用程式內串流中插入資料列的速率有其限制。因此，Kinesis Data Analytics 支援多個此類應用程式內串流，讓您更快速地將記錄匯入應用程式。如果發現應用程式未跟上串流來源中的資料，您可以新增平行處理單位以改善效能。

- 映射結構描述：描述串流來源上的記錄格式 (JSON、CSV)。同時也描述串流上的每條記錄，如何映射至所建立應用程式內串流的資料欄。這是您提供欄名稱和資料類型的地方。

> **Note**
>
> 建立輸入應用程式內串流時，Kinesis Data Analytics 會在識別符 (串流名稱和欄名稱) 周圍加上引號。查詢此串流和資料欄時，您必須使用相同的大小寫 (完全符合小寫和大寫字母) ，並以引號指定。如需識別符的詳細資訊，請參閱《Amazon Managed Service for Apache Flink SQL 參考資料》中的識別符。

您可以在 Amazon Kinesis Data Analytics 主控台中建立應用程式並設定輸入。然後，控制台進行必要的 API 呼叫。在建立新應用程式 API，或將輸入組態新增至現有應用程式時，您可以設定應用程式輸入。如需詳細資訊，請參閱 CreateApplication 及 AddApplicationInput。以下是 `Createapplication` API 請求主體的輸入配置部分：

```
"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"
        }
    ]
```

配置參考來源

您也可以選擇將參考資料來源新增至現有的應用程式，以豐富來自串流來源的資料。您必須將參考資料作為物件存放在 S3 儲存貯體中。Amazon Kinesis Data Analytics 會在應用程式啟動時讀取 Amazon S3 物件，並建立應用程式內參考資料表。接著，您的應用程式程式碼就可以將其與應用程式內串流聯結。

您可以使用支援的格式 (CSV、JSON) 將參考資料存放在 Amazon S3 物件中。舉例來說，假設您的應用程式對股票訂單執行分析。假設串流來源採用以下記錄格式：

```
Ticker, SalePrice, OrderId

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

在這種情況下，您可以考慮維護一個參考資料來源，以提供每個股票股票代號的詳細資料，例如公司名稱。

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

您可以使用 API 或主控台新增應用程式參考資料來源。Amazon Kinesis Data Analytics 提供下列 API 動作來管理參考資料來源：

- AddApplicationReferenceDataSource
- UpdateApplication

如需使用主控台新增任務的詳細資訊，請參閱 範例：將參考資料新增至 Kinesis Data Analytics 應用程式。

注意下列事項：

- 如果應用程式正在執行，Kinesis Data Analytics 會建立應用程式內參考資料表，然後立即載入參考資料。
- 如果應用程式未執行 (例如處於就緒狀態)，Kinesis Data Analytics 只會儲存更新的輸入組態。當應用程式開始執行時，Kinesis Data Analytics 會將參考資料以表格形式載入您的應用程式。

假設在 Kinesis Data Analytics 建立應用程式內參考資料表之後，您想要重新整理資料。也許你更新了 Amazon S3 物件，或者你想使用不同的 Amazon S3 物件。在此情況下，您可以明確呼叫UpdateApplication，或選擇主控台中的動作、同步化參考資料表。Kinesis Data Analytics 不會自動重新整理應用程式內參考資料表。

您可以建立為參考資料來源的 Amazon S3 物件大小有其限制。如需詳細資訊，請參閱 限制。如果物件大小超過限制，Kinesis Data Analytics 將無法載入資料。應用程式狀態會顯示為執行中，但不會讀取資料。

當新增參考資料來源時，您要提供以下資訊：

- S3 儲存貯體和物件金鑰名稱：除了儲存貯體名稱和物件金鑰之外，您還要提供一個 Kinesis Data Analytics 可以擔任的 IAM 角色，以代表您讀取物件。
- 應用程式內參考資料表名稱：Kinesis Data Analytics 會建立此應用程式內表格，並透過讀取 Amazon S3 物件來填入資料。這是您在應用程式碼中指定的資料表名稱。
- 映射結構描述：描述記錄格式 (JSON、CSV)，以及儲存在 Amazon S3 物件中的資料編碼。同時也說明每個資料元素如何映射，對應至應用程式內參考資料表中的資料欄。

以下顯示 `AddApplicationReferenceDataSource` API 請求中的請求內文。

```
{
    "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

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

JSONPath 是查詢 JSON 物件元素的標準化方式。JSONPath 使用路徑表達式導覽 JSON 文件中的元素、巢狀元素及陣列。如需 JSON 的詳細資訊，請參閱介紹 JSON。

Amazon Kinesis Data Analytics 會在應用程式的來源結構描述中使用 JsonPath 運算式，藉此在內含 JSON 格式資料的串流來源中識別資料元素。

如需將串流資料映射至應用程式輸入串流的詳細資訊，請參閱 the section called “將串流來源元素映射至 SQL 輸入資料欄”。

使用 JSONPath 存取 JSON 元素

接下來，你可以找到如何使用 JSONPath 表達式來存取 JSON 格式資料的各種元素。針對本節中的範例，假設來源串流包含下列 JSON 記錄：

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

存取 JSON 元素

若要用 JSONPath 查詢 JSON Data 中的元素，請使用以下語法。在此，$ 表示資料階層的根，elementName 是要查詢的元素節點名稱。

```
$.elementName
```

下列運算式會查詢上述 JSON 範例中的 customerName 元素。

```
$.customerName
```

上述運算式會從前面的 JSON 記錄傳回下列項目。

```
John Doe
```

> **Note**
>
> 路徑運算式區分大小寫。運算式 `$.customername` 會從前面的 JSON 範例傳回 `null`。

> **Note**
>
> 如果路徑表達式指定的位置沒有出現任何元素，則表達式會傳回 `null`。下列運算式會從前面的 JSON 範例傳回 `null`，因為沒有相符的元素。
>
> ```
> $.customerId
> ```

存取巢狀 JSON 元素

若要查詢巢狀 JSON 元素，請使用下列語法。

```
$.parentElement.element
```

下列運算式會查詢上述 JSON 範例中的 `city` 元素。

```
$.address.city
```

上述運算式會從前面的 JSON 記錄傳回下列項目。

```
Anytown
```

您可以使用以下語法查詢更多層次的子元素。

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

下列運算式會查詢上述 JSON 範例中的 `street` 元素。

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

上述運算式會從前面的 JSON 記錄傳回下列項目。

```
AnyStreet
```

存取陣列

您可以通過以下方式存取 JSON 陣列中的資料：

- 將陣列中的所有元素擷取為單一資料列。
- 將陣列中的每個元素擷取為單獨的列。

將陣列中的所有元素擷取為單一資料列。

若要將陣列的全部內容查詢為單一資料列，請使用下列語法。

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

下列運算式會查詢本節使用之上述 JSON 範例中的 orders 元素完整內容。它用單欄單列傳回陣列內容。

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

上述運算式會從本節使用之範例 JSON 記錄傳回下列項目。

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

將陣列中的所有元素擷取為單一資料列。

若要將陣列中的個別元素查詢為單獨的資料列，請使用下列語法。

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

下列運算式會查詢前述 JSON 範例中的 orderId 元素，並將每個陣列元素傳回為個別的資料列。

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

上述運算式會從前面的 JSON 記錄傳回下列項目，每個資料項目都會以個別的資料列傳回。

23284
63122
77284

Note

如果查詢非陣列元素的運算式包含在查詢個別陣列元素的結構描述中，則會針對陣列中的每個元素重複非陣列元素。舉例來說，假設上述 JSON 範例的結構描述包含下列運算式：

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

在這種情況下，來自範例輸入串流元素的傳回資料列類似於下列項目，每個 `orderId` 元素都會重複 `name` 元素。

John Doe	23284
John Doe	63122
John Doe	77284

Note

下列限制適用於 Amazon Kinesis Data Analytics 中的陣列運算式：

- 陣列運算式僅支援一個層級的解除參考。不支援下列運算式格式。

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

- 結構描述中只能展平一個陣列。可以參考多個陣列 — 傳回為包含陣列中所有元素的一列。但是，只有一個陣列可以將其中的每個元素作為單獨的列傳回。

  包含以下格式元素的結構描述是有效的。這種格式會傳回第二個陣列的內容作為單一欄，在第一個陣列中的每個元素重複。

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

包含以下格式元素的結構描述是無效的。

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

其他考量

使用 JSONPath 的其他注意事項如下：

- 如果在應用程式結構描述中，JSONPath 運算式的個別元素沒有存取任何陣列，則會針對處理的每個 JSON 記錄，在應用程式的輸入串流中建立單一資料列。
- 當陣列平面化 (即其元素會以個別資料列傳回) 時，任何遺失的元素都會導致應用程式內串流中出現 Null 值。
- 陣列永遠會平面化到至少一列。如果不會傳回任何值 (也就是陣列為空或未查詢任何元素)，則會傳回包含所有 Null 值的單一資料列。

  下列運算式會從前面的 JSON 範例傳回帶有 null 值的紀錄 ，因為指定的路徑沒有相符元素。

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

  上述運算式會從前面的 JSON 範例記錄傳回下列項目。

null
null
null

相關主題

- JSON 簡介

將串流來源元素映射至 SQL 輸入資料欄

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

透過 Amazon Kinesis Data Analytics，您可以使用標準 SQL 來處理和分析 JSON 或 CSV 格式的串流資料。

- 若要處理和分析串流 CSV 資料，請為輸入串流的欄指派欄名稱和資料類型。您的應用程式會依序從每個資料欄定義的輸入串流匯入一個資料欄。

  您不必在應用程式輸入串流中包含所有資料欄，但無法略過來源串流中的資料欄。例如，您可以從包含五個元素的輸入串流匯入前三個資料欄，但不能只匯入欄 1、2 和 4。
- 若要處理和分析串流 JSON 資料，您可以使用 JSONPath 運算式，將 JSON 元素從串流來源映射至輸入串流的 SQL 資料欄。如需搭配 Amazon Kinesis Data Analytics 使用 JSONPath 的詳細資訊，請參閱 使用 JSONPath。在 SQL 表中的欄具有從 JSON 類型映射的資料類型。關於支援的資料類型，請參閱資料類型。如需將 JSON 資料轉換為 SQL 資料的詳細資訊，請參閱 將 JSON 資料類型映射到 SQL 資料類型。

如需如何配置輸入串流的詳細資訊，請參閱 設定應用程式輸入。

將 JSON 資料映射至 SQL 資料欄

您可以使用 AWS Management Console 或 Kinesis 資料分析 API 將 JSON 元素對應至輸入資料欄。

- 若要使用主控台將元素映射至欄，請參閱 使用結構描述編輯器。
- 若要使用 Kinesis Data Analytics API 將元素映射至欄，請參閱下節。

若要將 JSON 元素映射至應用程式內輸入串流的資料欄，您需要每個資料欄的結構描述，且其中須包含下列資訊：

- 來源表達式：識別資料欄資料位置的 JSONPath 表達式。
- 資料欄名稱：SQL 查詢用來參考資料的名稱。
- 資料類型：資料欄的 SQL 資料類型。

使用 API

若要將串流來源中的元素映射至輸入資料欄，您可以使用 Kinesis Data Analytics API CreateApplication 動作。如要建立應用程式內串流，您必須指定一個結構描述，以將資料轉換為 SQL 中使用的架構化版本。CreateApplication 動作可以設定您的應用程式，以從單一串流來源接收輸入。若要將 JSON 元素或 CSV 資料欄映射至 SQL 資料欄，請在 SourceSchema `RecordColumns` 陣列中建立 RecordColumn 物件。RecordColumn 物件包含以下結構描述：

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

RecordColumn 物件中的欄位具有下列值：

- `Mapping`：識別輸入串流記錄資料位置的 JSONPath 運算式。CSV 格式的來源串流輸入結構描述不存在此值。
- `Name`：應用程式內 SQL 資料串流中的資料欄名稱。
- `SqlType`：應用程式內 SQL 資料串流中資料的類型。

JSON 輸入結構描述範例

下列範例會示範 JSON 結構描述 `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 輸入結構描述範例

下列範例會示範使用逗號分隔值 (CSV) 格式的結構描述 `InputSchema` 值格式。

```
"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"
}
```

將 JSON 資料類型映射到 SQL 資料類型

JSON 資料類型轉換為相應的 SQL 資料類型，根據的是應用程式之輸入結構描述。如需支援之 SQL 資料類型的詳細資訊，請參閱資料類型。Amazon Kinesis Data Analytics 會根據下列規則將 JSON 資料類型轉換為 SQL 資料類型。

Null 常值

JSON 輸入串流中的 null 常值 (也就是 `"City":null`) 會轉換為 SQL null，不論目的地資料類型為何。

布林常值。

JSON 輸入串流中的布林常值 (也就是 `"Contacted":true`) 會轉換為 SQL 資料，如下所示：

- 數字 (DECIMAL、INT 等)：`true` 轉換為 1；`false` 轉換為 0。
- 二進制 (BINARY 或 VARBINARY)：
  - `true`：結果具有最低位元組，並清除剩餘位元數。
  - `false`：結果已清除所有位元數。

  轉換為 VARBINARY 會產生 1 個位元的長度值。
- 布林值：轉換為相應的 SQL 布林值。
- 字元 (CHAR 或 VARCHAR)：轉換為對應的字串值 (`true` 或 `false`)。該值被截斷以適合欄位的長度。
- 日期時間 (DATE、TIME 或 TIMESTAMP)：轉換失敗，強制錯誤會寫入錯誤串流。

Number

JSON 輸入串流中的常值 (也就是 `"CustomerId":67321`) 會轉換為 SQL 資料，如下所示：

- 數字 (DECIMAL、INT 等)：直接轉換。如果轉換後的值超過目標資料類型 (也就是轉換 `123.4` 為 INT) 的大小或精確度，則轉換會失敗，並將強制錯誤寫入錯誤串流。
- 二進位 (BINARY 或 VARBINARY)：轉換失敗，強制錯誤會寫入錯誤串流。
- 布林值：
  - `0`：轉換為 `false`。
  - 所有其他數字：轉換為 `true`。
- 字符 (CHAR 或 VARCHAR) ：轉換為數字的字符串表示。
- 日期時間 (DATE、TIME 或 TIMESTAMP)：轉換失敗，強制錯誤會寫入錯誤串流。

字串

JSON 輸入串流中的字串值 (也就是 `"CustomerName":"John Doe"`) 會轉換為 SQL 資料，如下所示：

- 數字 (DECIMAL、INT 等)：Amazon Kinesis Data Analytics 會嘗試將值轉換為目標資料類型。如果無法轉換值，則轉換會失敗，並且會將強制錯誤寫入錯誤資料流。
- 二進位 (BINARY 或 VARBINARY)：如果來源字串是有效的二進位常值 (也就是 `X'3F67A23A'`，具有偶數 f)，該值會轉換為目標資料類型。否則，轉換會失敗，並將強制錯誤寫入錯誤串流。
- 布林值：如果來源字串為 `"true"`，則會轉換為 `true`。此比較不區分大小寫。否則，會轉換為 `false`。
- 字元 (CHAR 或 VARCHAR)：轉換為輸入的字串值。如果值長於目標資料類型，則會截斷該值，且不會將錯誤寫入錯誤串流。
- 日期時間 (DATE、TIME 或 TIMESTAMP)：如果來源字串的格式可轉換為目標值，則會轉換該值。否則，轉換會失敗，並將強制錯誤寫入錯誤串流。

  有效的日期時間格式包括：
  - "1992-02-14"
  - "1992-02-14 18:35:44.0"

物件的陣列

JSON 輸入流中的陣列或物件轉換為 SQL 資料，如下所示：

- 字元 (CHAR 或 VARCHAR)：轉換為陣列或物件的來源文字。請參閱 存取陣列。
- 其他所有資料類型：轉換失敗，並將強制錯誤寫入錯誤串流。

如需 JSON 陣列的範例，請參閱 使用 JSONPath。

相關主題

- 設定應用程式輸入
- 資料類型
- 使用結構描述編輯器
- CreateApplication
- RecordColumn
- SourceSchema

在串流資料上使用結構描述探索功能

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

如要提供輸入結構描述，以說明串流輸入中的記錄如何映射至應用程式內串流，可能會很麻煩且容易出錯。您可以使用 DiscoverInputSchema API (稱為探索 API) 來推斷結構描述。使用串流來源上的隨機範例記錄，API 可以推斷結構描述 (也就是資料欄名稱、資料類型以及傳入資料中資料元素的位置)。

> **Note**
>
> 若要使用探索 API 在 Amazon S3 存放的檔案產生結構描述，請參閱 在靜態資料上使用結構描述探索功能。

主控台使用探索 API 產生指定串流來源的結構描述。使用主控台，您也可以更新結構描述，包括新增或移除資料欄、變更資料欄名稱或資料類型等。但是，請仔細進行變更，以確保不會建立無效的結構描述。

完成應用程式內串流的結構描述之後，您可以使用一些函數來操作字串和日期時間值。在產生的應用程式內串流中使用資料列時，您可以在應用程式碼中使用這些函數。如需詳細資訊，請參閱 範例：轉換 DateTime 值。

探索結構描述期間的資料欄命名

在結構描述探索期間，Amazon Kinesis Data Analytics 會嘗試從串流輸入來源盡可能保留原始資料欄名稱，但下列情況除外：

- 來源串流資料欄名稱是保留的 SQL 關鍵字，例如 TIMESTAMP、USER、VALUES 或 YEAR。
- 來源串流資料欄名稱包含不支援的字元。僅允許使用字母、數字和底線 (_)。
- 來源串流資料欄名稱以數字開頭。
- 來源串流資料欄名稱長度超過 100 個字元。

如果重新命名資料欄，則重新命名的結構描述欄名稱會以 COL_ 開頭。在某些情況下，無法保留原始欄名稱，例如整個名稱是不支援的字元。在這種情況下，欄被命名為 COL_#，# 是指示欄在順序中的位置數字。

探索完成後，您可以使用主控台更新結構描述，以新增或移除資料欄，或變更資料欄名稱、資料類型或資料大小。

探索建議的資料欄名稱範例

來源串流資料欄名稱	探索建議的資料欄名稱範例
USER	COL_USER
USER@DOMAIN	COL_USERDOMAIN
@@	COL_0

結構描述探索問題

如果 Kinesis Data Analytics 未推斷指定串流來源的結構描述，會發生什麼情況？

Kinesis Data Analytics 會推斷您的結構描述使用常見的格式，例如 CSV 和 JSON，這些格式都是 UTF-8 編碼。Kinesis Data Analytics 使用自訂欄和資料列分隔符號支援任何 UTF-8 編碼記錄 (包括應用程式日誌和記錄等原始文字)。如果 Kinesis Data Analytics 未推斷結構描述，您可以使用主控台上的結構描述編輯器 (或使用 API) 手動定義結構描述。

如果您的資料不遵循模式 (您可以使用結構描述編輯器表明)，您可以將結構描述定義為 VARCHAR (N) 類型的單一資料欄，其中 N 是您預期記錄包含的最多字元數。從那裡，您可以使用字串和日期時間操作，在資料傳入應用程式內串流之後建構資料。如需範例，請參閱 範例：轉換 DateTime 值。

在靜態資料上使用結構描述探索功能

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

結構描述探索功能，可以從串流資料或儲存在 Amazon S3 儲存貯體的靜態檔案資料產生結構描述。假設您想要為 Kinesis Data Analytics 應用程式產生結構描述，以供參考或因應無法使用即時串流資料的時刻。在靜態檔案上使用結構描述探索功能，該檔案包含串流或參考資料中預期格式的資料樣本。Kinesis Data Analytics 可以針對儲存在 Amazon S3 儲存貯體的 JSON 或 CSV 檔案中的範例資料執行結構描述探索。在資料檔案上使用結構描述探索，該檔案使用主控台或指定了 `S3Configuration` 參數的 DiscoverInputSchema API。

使用主控台執行結構描述探索

若要使用主控台在靜態檔案上運行探索，請執行下列動作：

1. 將參考資料物件新增至 S3 儲存貯體。
2. 在 Kinesis Data Analytics 資料分析主控台的應用程式主頁中，選擇連接參考資料。
3. 提供儲存貯體、路徑和 IAM 角色資料，以存取包含參考資料的 Amazon S3 物件。
4. 選擇探索結構描述。

如需如何在主控台中新增參考資料和探索結構描述的詳細資訊，請參閱 範例：將參考資料新增至 Kinesis Data Analytics 應用程式。

使用主控台執行結構描述探索

若要使用 API 在靜態檔案上執行探索，請為 API 提供具有下列資訊的 `S3Configuration` 結構：

- `BucketARN`：內含檔案的 Amazon S3 儲存貯體之 Amazon Resource Name (ARN)。如需 Amazon S3儲存貯體 ARN 的格式，請參閱 Amazon Resource Names (ARNs) 和 Amazon Service 命名空間：Amazon Simple Storage Service (Amazon S3)。

- RoleARN：具有 AmazonS3ReadOnlyAccess 政策的 IAM 角色之 ARN。如需將政策新增至角色的詳細資訊，請參閱修改角色。
- FileKey：物件的檔案名稱。

使用 **DiscoverInputSchema** API 從 Amazon S3 物件產生結構描述

1. 請確定您已 AWS CLI 設定。如需詳細資訊，請參閱入門指南一節中的 步驟 2：設定 AWS Command Line Interface (AWS CLI)。
2. 使用下列內容建立名為 data.csv 的檔案：

```
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. 至 https://console.aws.amazon.com/s3/ 登入 Amazon S3 主控台。
4. 建立 Amazon S3 儲存貯體並上傳您建立的 data.csv 檔案 請記下您建立之儲存貯體的 ARN。如需有關建立 Amazon S3 儲存貯體及上傳檔案的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》。
5. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。使用 AmazonS3ReadOnlyAccess 政策建立角色。請記下新角色的 ARN。如需有關建立角色的詳細資訊，請參閱建立角色以委派許可給 Amazon 服務。如需將政策新增至角色的詳細資訊，請參閱修改角色。
6. 在中執行下列DiscoverInputSchema命令 AWS CLI，將 ARN 取代為您的 Amazon S3 儲存貯體和 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. 回應看起來類似以下的內容。

```
{
    "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 函數預處理資料

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

如果串流中的資料需要格式轉換、轉換、擴充或篩選，您可以使用函數預先處理資料。 AWS Lambda 您可以在應用程式 SQL 程式碼執行之前，或在應用程式從資料串流建立結構描述之前執行此動作。

在下列案例中，使用 Lambda 函數來預先處理記錄非常有用：

- 將記錄從其他格式 (例如 KPL 或 GZIP) 轉換為 Kinesis Data Analytics 分析可以分析的格式。Kinesis Data Analytics 目前支援 JSON 或 CSV 資料格式。
- 將資料擴展為彙總或異常偵測等作業更容易存取的格式。舉例來說，如果多個資料值一起存儲在一個字串中，則可以將資料擴展到單獨的欄中。
- 透過其他 Amazon 服務進行資料充實，例如外推法或錯誤修正。
- 將複雜的字符串轉換應用於記錄欄位。
- 用於清理資料的數據過濾。

使用 Lambda 函數預處理資料

建立 Kinesis Data Analytics 應用程式時，您可以在連接至來源頁面中啟用 Lambda 預先處理。

使用 Lambda 函數預先處理 Kinesis Data Analytics 應用程式中的記錄

1. 登入 AWS Management Console 並開啟適用於 Apache Flink 的受管理服務主控台，網址為 https://console.aws.amazon.com/kinesisanalytics。
2. 在應用程式的連接至來源頁面上，選擇記錄預處理方式] AWS Lambda區段中的啟用。
3. 若要使用已建立的 Lambda 函數，請在 Lambda 函數下拉式清單中選擇函數。
4. 若要從其中一個 Lambda 預處理範本建立新的 Lambda 函數，請從下拉式清單中選擇範本。然後選擇 觀看 Lambda 中的 <template name> 來編輯函數。
5. 選擇建立新的來建立新 Lambda 函數。如需建立 Lambda 函數的相關資訊，請參閱AWS Lambda 開發人員指南中的建立 HelloWorld Lambda 函數和探索主控台。
6. 選擇要使用的 Lambda 函數版本。若要使用最新版本，請選擇 $LATEST。

當您選擇或建立 Lambda 函數進行記錄預先處理時，系統會在執行應用程式 SQL 程式碼，或應用程式從記錄產生結構描述之前預先處理記錄。

Lambda 預處理許可

若要使用 Lambda 預處理，應用程式的 IAM 角色需要下列許可政策：

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

Lambda 預處理指標

您可以使 CloudWatch 用 Amazon 監控 Lambda 叫用的數量、處理的位元組數、成功與失敗等。如需 Kinesis 資料分析 Lambda 預先處理所發出 CloudWatch 指標的相關資訊，請參閱 Amazon Kin esis Analytics 指標。

搭 AWS Lambda 配 Kinesis 製作者程式庫使用

Kinesis Producer Library (KPL) 會將使用者格式化的小型記錄彙整成至多 1 MB 的較大型記錄，以便更妥善利用 Amazon Kinesis Data Streams 輸送量。適用於 Java 的 Kinesis Client Library (KCL) 支援取消彙整這類記錄。但是，當您用作流的消費者時，您必須使用 AWS Lambda 特殊模塊來分解記錄。

若要取得必要的專案程式碼和指示，請參閱〈Kinesis Productor 程式庫解彙總模組〉。 AWS LambdaGitHub您可以使用這個項目中的組件來處理 Java，Node.js 和 Python AWS Lambda 中的 KPL 序列化數據。上述元件也可用於建構多語言 KCL 應用程式。

資料預處理事件輸入資料模型/記錄響應模型

若要預處理記錄，您的 Lambda 函數必須符合所需的事件輸入資料和記錄回應模型。

事件輸入資料模型

Kinesis Data Analytics 會持續讀取 Kinesis 資料串流或 Firehose 交付串流中的資料。對於擷取的每批記錄，服務會管理每個批次傳遞至 Lambda 函數的方式。您的函數接收記錄列表作為輸入。在函數中，您可以迭代列表並應用業務邏輯以完成預處理需求（例如資料格式轉換或擴充）。

預先處理函數的輸入模型會略有不同，具體取決於資料是從 Kinesis 資料串流還是 Firehose 傳遞串流接收。

如果來源是 Firehose 傳遞串流，則事件輸入資料模型如下：

Kinesis Data Firehose 請求數據模型

欄位	描述
`invocationId`	Lambda 調用 ID (隨機 GUID)。
`applicationArn`	Kinesis Data Analytics 應用程式的 Amazon Resource Name (ARN)
`streamArn`	交付串流 ARN
紀錄	

欄位	描述	
`recordId`	記錄 ID (隨機 GUID)	

欄位	描述
欄位	描述
kinesisFirehoseRecordMetadata	欄位 描述 approximateArrivalTimestamp 交付串流記錄大約到達時間
data	Base64 編碼來源記錄承載

下列範例顯示來自 Firehose 交付串流的輸入：

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

如果來源是 Kinesis 資料串流，則事件輸入資料模型如下：

Kinesis 串流請求資料模型

欄位	描述
invocationId	Lambda 調用 ID (隨機 GUID)。

欄位	描述
applicationArn	Kinesis Data Analytics 應用程式 ARN
streamArn	交付串流 ARN

紀錄

欄位	描述	
recordId	以 Kinesis 記錄序號為基礎的記錄 ID	
kinesisStreamRecordMetadata	欄位 / 描述 sequenceNumber：來自 Kinesis 串流記錄的序號 partitionKey：Kinesis 串流記錄中的分割區索引鍵 shardId：Kinesis 串流記錄的 ShardId approximateArrivalTimestamp：交付串流記錄大約到達時間	
資料	Base64 編碼來源記錄承載	

下列範例顯示 Kinesis 資料串流的輸入：

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

紀錄回應模型

必須傳回送至 Lambda 函數的所有從 Lambda 預處理函數傳回的記錄 (含有記錄 ID)。它們必須包含以下參數，否則 Kinesis Data Analytics 會拒絕這類紀錄，並將其視為資料預處理失敗。資料有效承載部分可以轉換，以完成預處理要求。

回應資料模型

紀錄

欄位	描述
`recordId`	在調用期間，記錄 ID 會從 Kinesis Data Analytics 傳遞至 Lambda。轉換記錄必須包含相同的記錄 ID。原始記錄的 ID 與轉換記錄的 ID 若有任何不符，就會視為資料轉換失敗。
`result`	記錄的資料轉換狀態。可能值如下： • Ok：記錄已成功轉換。Kinesis Data Analytics 會擷取記錄讓 SQL 處理。 • `Dropped`：您的處理邏輯故意丟棄了記錄。Kinesis Data Analytics 捨棄經 SQL 處理的紀錄。資料承載欄位對於 `Dropped` 記錄而言是選擇性的。 • `ProcessingFailed` ：無法轉換記錄。Kinesis Data Analytics 認為 Lambda 函數未成功處理它，並將錯誤寫入錯

欄位	描述
	誤串流。關於錯誤串流的詳細資訊，請查看 錯誤處理。資料承載欄位對於 `ProcessingFailed` 記錄而言是選擇性的。
`data`	已轉換資料承載 (base64 編碼後)。如果應用程式擷取資料格式為 JSON，則每個資料承載都可以包含多個 JSON 文件。或者，如果應用程式擷取資料格式為 CSV，則每個列都可以包含多個 CSV 列 (在每一列中指定資料列分隔符號)。Kinesis Data Analytics 服務能夠在相同資料承載中成功剖析和處理包含多個 JSON 文件或 CSV 列的資料。

以下是 Lambda 函數輸出的範例：

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

常見的資料預處理失敗

以下是預處理失敗的常見原因。

- 並非所有傳送至 Lambda 函數的批次記錄 (具有記錄 ID) 都會傳回 Kinesis Data Analytics 服務。
- 回應遺失記錄 ID、狀態或資料承載欄位。資料承載欄位對於 `Dropped` 或 `ProcessingFailed` 記錄而言是選擇性的。
- Lambda 函數逾時不足以預處理資料。
- Lambda 函數回應超過 AWS Lambda 服務施加的回應限制。

對於資料預處理失敗，Kinesis Data Analytics 會繼續在同一組記錄上重試 Lambda 調用，直到成功為止。您可以監視下列 CloudWatch 指標，以深入瞭解失敗。

- Kinesis Data Analytics 應用程式 `MillisBehindLatest`：指出應用程式從串流來源讀取落後的程度。
- Kinesis Data Analytics 應用程式指`InputPreprocessing` CloudWatch 標：指出成功和失敗次數以及其他統計資料。如需詳細資訊，請參閱Amazon Kinesis Analytics 指標。
- AWS Lambda 功能 CloudWatch 指標和日誌。

建立 Lambda 函數以進行預處理

將記錄導入應用程式時，您的 Amazon Kinesis Data Analytics 應用程式可以使用 Lambda 函數來預處理記錄。Kinesis Data Analytics 會在主控台上提供下列範本，作為預處理資料的起點。

主題

- 在 Node.js 中建立預處理 Lambda 函數
- 在 Python 中建立預處理 Lambda 函數
- 在 Java 中建立預處理 Lambda 函數
- 在 .NET 中建立預處理 Lambda 函數

在 Node.js 中建立預處理 Lambda 函數

Kinesis Data Analytics 主控台提供了下列在 Node.js 中建立預處理 Lambda 函數的範本：

Lambda 藍圖	語言與版本	描述
一般 Kinesis Data Analytics 輸入處理	Node.js 6.10	Kinesis Data Analytics 記錄預處理器，可接收 JSON 或 CSV 記錄做為輸入，然後傳回處理狀態。使用此處理器作為自訂轉換邏輯的起點。
壓縮輸入處理	Node.js 6.10	Kinesis Data Analytics 記錄預處理器，可接收壓縮的 (GZIP 或 Deflate 壓縮) JSON 或 CSV 記錄做為輸入，然後傳回解壓縮的紀錄與處理狀態。

在 Python 中建立預處理 Lambda 函數

主控台提供在 Python 中建立預處理 Lambda 函數的下列範本：

Lambda 藍圖	語言與版本	描述
一般 Kinesis Analytics 輸入處理	Python 2.7	Kinesis Data Analytics 記錄預處理器，可接收 JSON 或 CSV 記錄做為輸入，然後傳回處理狀態。使用此處理器作為自訂轉換邏輯的起點。
KPL 輸入處理	Python 2.7	接收 Kinesis 生產者程式庫 (KPL) 的 Kinesis Data Analytics 記錄處理器，彙總 JSON 或 CSV 記錄作為輸入，並傳回具有處理狀態的分解記錄。

在 Java 中建立預處理 Lambda 函數

如要在 Java 中建立預處理紀錄的 Lambda 函數，請使用 Java 事件類別。

下列程式碼會示範使用 Java，且可預處理紀錄的範例 Lambda 函數：

```
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 中建立預處理 Lambda 函數

如要在 .NET 中建立預處理紀錄的 Lambda 函數，請使用 .NET 事件類別。

下列程式碼會示範使用 C#，且可預處理紀錄的範例 Lambda 函數：

```
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;
        }
    }
```

如需在 .NET 中建立用於預處理和目的地的 Lambda 函數詳細資訊，請參閱 Amazon.Lambda.KinesisAnalyticsEvents。

平行化輸入串流以提高輸送量

> **Note**
>
> 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。如需詳細資訊，請參閱限制。

Amazon Kinesis Data Analytics 應用程式可支援多個應用程式內輸入串流，將應用程式擴展到超越單一應用程式內輸入串流的輸送量。如需應用程式內輸入串流的詳細資訊，請參閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式。

在幾乎所有情況下，Amazon Kinesis Data Analytics 都會擴展您的應用程式，以處理饋入應用程式的 Kinesis 串流或 Firehose 來源串流的容量。不過，如果來源串流的輸送量超過單一應用程式內輸入串流的輸送量，您可以明確增加應用程式使用的應用程式內輸入串流數目。您可以使用 `InputParallelism` 參數來執行此操作。

當 `InputParallelism` 參數大於一時，Amazon Kinesis Data Analytics 會在應用程式內串流中平均分割來源串流的分割區。舉例來說，如果來源串流有 50 個碎片，且您設定 `InputParallelism` 為 2，則每個應用程式內輸入串流都會接收來自 25 個來源串流碎片的輸入。

增加應用程式內串流的數量時，您的應用程式必須明確存取每個串流中的資料。如需在程式碼中存取多個應用程式內串流的相關資訊，請參閱 在 Amazon Kinesis Data Analytics 應用程式中存取個別應用程式內串流。

雖然 Kinesis Data Streams 和 Firehose 串流碎片會以相同的方式在應用程式內串流分割，但它們在應用程式中的外觀有所不同：

- Kinesis 資料串流中的記錄包含 shard_id 欄位，可用來識別記錄來源碎片。
- Firehose 傳送串流中的記錄不包含識別記錄來源碎片或分割區的欄位。這是因為 Firehose 會將這些資訊從您的應用程式中抽象出來。

評估是否增加應用程式內輸入串流的數量

在大多數情況下，單一應用程式內輸入串流可以處理單一來源串流的輸送量，視輸入串流的複雜性和資料大小而定。若要判斷是否需要增加應用程式內輸入串流的數量，您可以在 Amazon CloudWatch 中監控InputBytes和MillisBehindLatest指標。

如果 InputBytes 指標大於每秒 100 MB (或者您預期它會大於此速率)，這可能會導致 MillisBehindLatest 提高，並增加應用程式問題的影響。為了解決這個問題，我們建議您為應用程式選擇下列語言：

- 如果應用程式的擴展需求超過每秒 100 MB，請針對 SQL 應用程式使用多個串流和 Kinesis Data Analytics。
- 如果您想要使用單一串流和應用程式，請使用Kinesis Data Analytics for Java 應用程式。

如果 MillisBehindLatest 指標具有下列任一特性，則應增加應用程式的 InputParallelism 設定：

- MillisBehindLatest 指標逐漸增加，表示您的應用程式逐漸落後串流中的最新資料。
- MillisBehindLatest 指標一直高於 1000 (一秒)。

如果符合下列條件，就不需要增加應用程式的 InputParallelism 設定：

- MillisBehindLatest 指標逐漸減少，表示您的應用程式逐漸趕上串流中的最新資料。
- MillisBehindLatest 指標低於 1000 (一秒)。

若要取得有關使用的更多資訊 CloudWatch，請參閱CloudWatch 使用指南。

實作多個應用程式內輸入串流

在使用 CreateApplication 建立應用程式時，您可以設定應用程式內輸入串流的數目。您可以使用 UpdateApplication 在建立應用程式後設定此數字。

> **Note**
>
> 您只能使用 Amazon Kinesis Data Analytics API 或 AWS CLI來做出 `InputParallelism` 設定。您無法使用設定此設定 AWS Management Console。如需有關設定的資訊 AWS CLI，請參閱步驟 2：設定 AWS Command Line Interface (AWS CLI)。

設定新應用程式的輸入串流計數

以下範例示範如何使用 `CreateApplication` API 動作，將新的應用程式輸入串流計數設定為 2。

如需 `CreateApplication` 的相關資訊，請參閱 CreateApplication。

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

設定現有應用程式的輸入串流計數

以下範例示範如何使用 `UpdateApplication` API 動作，將現有應用程式輸入串流計數設定為 2。

如需 `Update_Application` 的相關資訊，請參閱 UpdateApplication。

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

```
      }
   ],
}
```

在 Amazon Kinesis Data Analytics 應用程式中存取個別應用程式內串流

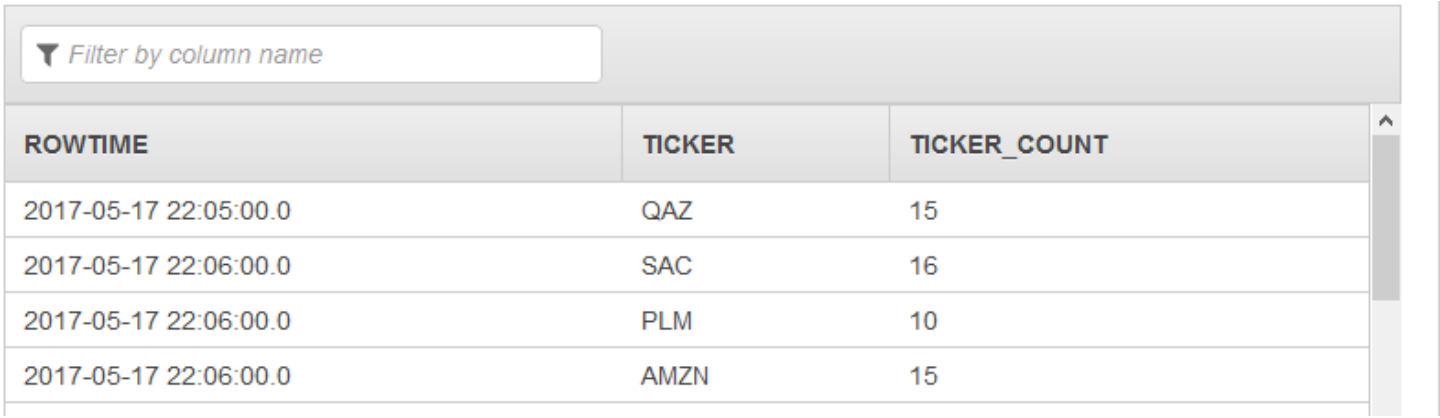
若要在應用程式中使用多個應用程式內輸入串流，您必須從不同的串流中明確選取。下列程式碼範例，示範了如何在入門教學課程建立的應用程式中查詢多個輸入串流。

在下列範例中，會先使用 COUNT 彙總每個來源資料流，然後再合併到名為 `in_application_stream001` 的單一應用程式內串流。事先彙總來源串流，有助於確保合併的應用程式內串流可以處理來自多個串流的流量，而不會超載。

> **Note**
>
> 若要執行此範例並從應用程式內輸入串流取得結果，請同時更新來源串流中的碎片數目，和應用程式中的 `InputParallelism` 參數。

```
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;
```

上述程式碼範例會在 `in_application_stream001` 產生輸出，類似下列所示：



其他考量

當您使用多個輸入串流時，請注意下列事項：

- 應用程式內輸入串流數量上限為 64 個。
- 應用程式內的輸入串流，會平均分佈在應用程式輸入串流的碎片之間。
- 新增應用程式內串流所帶來的效能不會線性擴展。也就是說，加倍應用程式內串流的數量並不會讓輸送量增加一倍。使用典型的資料列大小，每個應用程式內串流可達到每秒約 5,000 至 15,000 個資料列的輸送量。將應用程式內串流計數增加到 10，您可以達到每秒 20,000 到 30,000 個資料列的輸送量。輸送量速度取決於輸入串流中欄位的計數、資料類型和資料大小。
- 當套用至分割成不同碎片的輸入資料流時，某些彙總函式 (例如 AVG) 可能會產生非預期的結果。因為您必須先在個別碎片上執行彙總作業，然後再將它們合併到彙總串流中，因此結果可能會加權為包含更多記錄的任何資料流。
- 如果在增加輸入串流數量後，您的應用程式在持續效能不佳 (反映在高 `MillisBehindLatest` 指標)，您可能已達到 Kinesis 處理單元 (KPU) 的上限。如需更多詳細資訊，請參閱 自動擴展應用程式以增加輸送量。

應用程式碼

應用程式碼是處理輸入並產生輸出的一系列 SQL 陳述式。這些 SQL 陳述式會在應用程式內串流和參考資料表上運作。如需更多詳細資訊，請參閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式。

如需 Kinesis Data Analytics 支援之 SQL 語言元素的相關資訊，請參閱 Amazon Kinesis Data Analytics SQL 參考資料。

在關聯式資料庫中，您可以使用資料表，以 INSERT 陳述式來新增記錄，並使用 SELECT 陳述式來查詢資料。在 Amazon Kinesis Data Analytics 中，您可以使用串流。您可以撰寫 SQL 陳述式來查詢這些串流。查詢一個應用程式內串流的結果，一律會傳送到另一個應用程式內串流。執行複雜的分析時，您可以建立數個應用程式內串流來保存中繼分析的結果。最後設定應用程式輸出，將最終分析的結果 (從一或多個應用程式內串流) 保存到外部目的地。總而言之，以下是編寫應用程式碼的典型模式：

- SELECT 陳述式一律會在 INSERT 陳述式的環境中使用。也就是說，當您選取資料列時，會將結果插入另一個應用程式內串流。
- SELECT 陳述式一律會在幫浦的環境中使用。也就是說，您可以使用幫浦來寫入應用程式內串流。

下列範例應用程式碼會從一個應用程式內 (SOURCE_SQL_STREAM_001) 串流讀取記錄，然後將其寫入另一個應用程式內串流 (DESTINATION_SQL_STREAM)。您可以使用幫浦將記錄插入應用程式內串流，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**
>
> 您為串流和資料欄名稱指定的識別符遵循標準 SQL 慣例。舉例來說，如果您在識別符周圍加上引號，即代表標識符區分大小寫。如果不這麼做，識別符預設為大寫。如需識別符的詳細資訊，請參閱《Amazon Managed Service for Apache Flink SQL 參考資料》中的識別符。

您的應用程式碼可以由許多 SQL 陳述式組成。例如：

- 您可以依照順序來撰寫 SQL 查詢，即一個 SQL 陳述式的結果會饋送到下一個 SQL 陳述式。
- 您也可以撰寫獨立執行的 SQL 查詢。例如，您可以撰寫兩個 SQL 陳述式來查詢相同的應用程式內串流，但會將輸出傳送至不同的應用程式內串流。然後，您可以獨立查詢新建立的應用程式內串流。

您可以建立應用程式內串流以儲存中繼結果。您可以使用幫浦將資料列插入應用程式內串流。如需更多詳細資訊，請參閱 應用程式內串流與幫浦。

如果您新增了應用程式內參考資料表，可以撰寫 SQL 來聯結應用程式內串流和參考資料表中的資料。如需更多詳細資訊，請參閱 範例：將參考資料新增至 Kinesis Data Analytics 應用程式。

根據應用程式的輸出組態，Amazon Kinesis Data Analytics 會根據應用程式的輸出組態，將資料從特定應用程式內串流寫入外部目的地。請確定您的應用程式碼已寫入輸出組態中指定的應用程式內串流。

如需詳細資訊，請參閱下列主題：

- 串流 SQL 概念
- Amazon Kinesis Data Analytics SQL 參考資料

設定應用程式輸出

在應用程式碼中，您可以將 SQL 陳述式的輸出寫入一或多個應用程式內串流。您可以選擇性地將輸出組態新增至應用程式。將寫入應用程式內串流的所有內容保留到外部目的地，例如 Amazon Kinesis 資料串流、Firehose 交付串流或函數。 AWS Lambda

您可以用來保存應用程式輸出的外部目的地數量有限制。如需詳細資訊，請參閱 限制。

> **Note**
>
> 建議您使用其中一個目的地來保留應用程式內錯誤串流資料，藉此調查錯誤。

在每種輸出組態中，您提供下列項目：

- 應用程式內串流名稱：您要保留至外部目的地的串流。

  Kinesis Data Analytics 會尋找您在輸出組態中指定的應用程式內串流。(串流名稱區分大小寫，必須完全匹配。) 請確定您的應用程式碼會建立此應用程式內串流。
- 外部目標 — 您可以將資料保留至 Kinesis 資料串流、Firehose 交付串流或 Lambda 函數。您提供資料串流或函數的 Amazon Resource Name (ARN)。您也必須提供 Kinesis Data Analytics 可擔任的 IAM 角色，以代您讀取串流或函數。在寫入外部目的地時，您可以向 Kinesis Data Analytics 描述要使用的記錄格式 (JSON、CSV)。

如果 Kinesis Data Analytics 無法寫入串流或 Lambda 目的地，服務會繼續無限期地嘗試。這會產生背壓，導致您的應用程式落後。如果此問題未解決，您的應用程式最終會停止處理新資料。您可以監控 Kinesis Data Analytics 指標並設定故障警示。如需有關指標和警示的詳細資訊，請參閱使用 Amazon CloudWatch 指標和建立 Amazon CloudWatch 警示。

您可以使用 AWS Management Console配置應用程式輸出。控制台進行 API 呼叫以保存配置。

使用建立輸出 AWS CLI

本節說明如何建立 `CreateApplication` 或 `AddApplicationOutput` 作業的請求內文 `Outputs` 區段。

建立 Kinesis 串流輸出

下列 JSON 片段顯示 `CreateApplication` 要求內文中用來建立 Amazon Kinesis 資料串流目的地的 `Outputs` 區段。

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

]
```

建立 Firehose 傳送串流輸出

下列 JSON 片段顯示`CreateApplication`請求內文中用於建立 Amazon 資料 Firehose 交付串流目的地的`Outputs`區段。

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

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

建立 Lambda 函數輸出

以下 JSON 片段顯示`CreateApplication`請求主體中用於創建 AWS Lambda 函數目標的`Outputs`部分。

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

使用 Lambda 函數作為輸出

AWS Lambda 作為目的地使用可讓您在將 SQL 結果傳送至最終目的地之前，更輕鬆地執行後續處理。常見後續處理任務包括下列：

- 將多個列彙總到單個記錄中
- 結合目前結果與過去的結果，以處理延遲到達的資料
- 根據資訊類型交付到不同目標
- 記錄格式轉換 (如翻譯成 Protobuf)
- 字串操作或轉換
- 分析處理後的資料擴充
- 地理空間使用案例的自訂處理

- 資料加密

Lambda 函數可以將分析資訊傳遞至各種 AWS 服務和其他目的地，包括：

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

如需有關建立 Lambda 應用程式的詳細資訊，請參閱入門 AWS Lambda。

主題

- Lambda 作為輸出許可
- Lambda 作為輸出指標
- Lambda 作為輸出事件輸入資料模型和記錄回應模型
- Lambda 輸出調用頻率
- 新增用作輸出的 Lambda 函數
- Lambda 作為輸出之常見故障
- 為應用程式目的地建立 Lambda 函數

Lambda 作為輸出許可

若要使用 Lambda 做為輸出，應用程式的 Lambda 輸出 IAM 角色需要下列許可政策：

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

```
        "lambda:GetFunctionConfiguration"
    ],
    "Resource": "FunctionARN"
}
```

Lambda 作為輸出指標

您可以使 CloudWatch 用 Amazon 監視傳送的位元組數、成功與失敗等。如需 Kinesis 資料分析使用 Lambda 作為輸出所發出的 CloudWatch 指標的相關資訊，請參閱 Amazon Kinesis Analytics 指標。

Lambda 作為輸出事件輸入資料模型和記錄回應模型

若要傳送 Kinesis Data Analytics 輸出記錄，您的 Lambda 函數必須符合所需的事件輸入資料和記錄回應模型。

事件輸入資料模型

Kinesis Data Analytics 會持續將輸出記錄從應用程式傳送至 Lambda，這些輸出紀錄用作輸出函數，並具有下列要求模型。在函數中，迭代列表並應用業務邏輯來完成輸出需求（例如，在發送到最終目的地之前的資料轉換）。

欄位	描述
invocationId	Lambda 調用 ID (隨機 GUID)。
applicationArn	Kinesis Data Analytics 應用程式的 Amazon Resource Name (ARN)。

紀錄

欄位	描述
recordId	記錄 ID (隨機 GUID)
lambdaDeliveryRecordMetadata	欄位 描述 retryHir 交付重試次數

欄位	描述

欄位	描述
資料	Base64 編碼輸出紀錄承載

> **Note**
>
> `retryHint` 值在每次交付失敗都會增加。此值不會長期存在，如果應用程式中斷，則會重設。

紀錄回應模型

傳送至 Lambda 做為輸出函數 (含記錄 ID) 的每筆記錄，都必須使用 `Ok` 或 `DeliveryFailed` 來確認，且必須包含下列參數。否則，Kinesis Data Analytics 會將它們視為交付失敗。

紀錄

欄位	描述
`recordId`	在調用期間，記錄 ID 會從 Kinesis Data Analytics 傳遞至 Lambda。原始記錄與經確認記錄的 ID 若有任何不符，就會視為幾交付失敗。
`result`	記錄交付的狀態。以下是可能的值： • `Ok`：記錄已成功轉換並傳送至最終目的地。Kinesis Data Analytics 會擷取記錄讓 SQL 處理。 • `DeliveryFailed` ：Lambda 作為輸出函數，未成功將記錄交付至最終目的地。Kinesis Data Analytics 會持續重試，將交付失敗記錄傳送至 Lambda 作為輸出函數。

Lambda 輸出調用頻率

Kinesis Data Analytics 應用程式會緩衝輸出記錄，並經常調用 AWS Lambda 目標函數。

- 如果將記錄發送到資料分析應用程式中的目標應用程式內串流作為暫停視窗，則會在每個暫停視窗觸發程序叫用 AWS Lambda 目標函數。例如，如果使用 60 秒的輪轉窗口將記錄發送到目的地應用程式內串流，則每 60 秒會調用 Lambda 函數一次。
- 如果記錄以連續查詢或滑動窗口的形式，發送到應用程式的目的地應用程式內串流，則每秒大約會調用一次 Lambda 目的地函數。

> **Note**
>
> 適用每個 Lambda 函數調用要求承載大小限制。超過這些限制會導致輸出記錄分割，並跨越多個 Lambda 函數呼叫傳送。

新增用作輸出的 Lambda 函數

以下程序說明如何將 Lambda 函數新增為 Kinesis Data Analytics 應用程式的輸出。

1. 登入 AWS Management Console 並開啟適用於 Apache Flink 的受管理服務主控台，網址為 https://console.aws.amazon.com/kinesisanalytics。
2. 選擇清單中的應用程式，然後選擇應用程式詳細資訊。
3. 在目的地區段中，選擇連接新目的地。
4. 針對目的地項目，選擇 AWS Lambda 函數。
5. 在交付記錄至 AWS Lambda 區段中，選擇現有的 Lambda 函數和版本，或選擇建立新的。
6. 如果您要建立新 Lambda 函數，請執行下列動作：
   a. 選擇其中一個範本。如需詳細資訊，為應用程式目的地建立 Lambda 函數。
   b. 建立函數 頁面在 Web 瀏覽器的新瀏覽器標籤中開啟。在名稱方塊中，為函數指定一個有意義的名稱 (例如 **`myLambdaFunction`**)。
   c. 用後續處理功能為應用程式更新範本。如需建立 Lambda 函數的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 入門。
   d. 在 Kinesis Data Analytics 主控台的 Lambda 函數清單中，選擇您剛建立的 Lambda 函數。為 Lambda 函數版本選擇 $LATEST。
7. 在應用程式內串流區段，選擇選擇現有的應用程式內串流。針對應用程式內串流名稱，選擇應用程式的輸出串流。所選輸出串流的結果會傳送至 Lambda 輸出函數。
8. 將表格的其餘部分保留為預設值，然後選擇儲存並繼續。

您的應用程式現在會將記錄從應用程式內串流傳送到 Lambda 函數。您可以在 Amazon CloudWatch 主控台中查看預設範本的結果。監控 `AWS/KinesisAnalytics/LambdaDelivery.OkRecords` 指標，以查看交付至 Lambda 函數的記錄數。

Lambda 作為輸出之常見故障

以下是交付至 Lambda 函數可能會失敗的常見原因。

- 並非所有傳送至 Lambda 函數的批次記錄 (具有記錄 ID) 都會傳回 Kinesis Data Analytics 服務。
- 回應遺失記錄 ID 或狀態欄位。
- Lambda 函數逾時不足以完成 Lambda 函數中的業務邏輯。
- Lambda 函數中的業務邏輯不會擷取所有錯誤，導致未處理的例外狀況造成逾時和背壓。這些通常被稱為「毒丸」訊息。

對於資料傳遞失敗，Kinesis Data Analytics 會繼續在同一組記錄上重試 Lambda 調用，直到成功為止。若要深入瞭解失敗，您可以監視下列 CloudWatch 指標：

- Kinesis Data Analytics 應用程式 Lambda 作為輸出指 CloudWatch 標：指出成功和失敗的次數以及其他統計資料。如需詳細資訊，請參閱Amazon Kinesis Analytics 指標。
- AWS Lambda 功能 CloudWatch 指標和日誌。

為應用程式目的地建立 Lambda 函數

您的 Kinesis Data Analytics 應用程式可以使用 AWS Lambda 函數做為輸出。Kinesis Data Analytics 提供範本，可讓您建立 Lambda 函數作為應用程式的目的地使用。使用這些範本做為應用程式後續處理輸出的起點。

主題

- 在 Node.js 中建立一個 Lambda 函數目的地
- 在 Python 中建立一個 Lambda 函數目的地
- 在 Java 中建立一個 Lambda 函數目的地
- 在 .NET 中建立一個 Lambda 函數目的地

在 Node.js 中建立一個 Lambda 函數目的地

主控台提供在 Node.js 中建立目的地 Lambda 函數的下列範本：

Lambda 作為輸出藍圖	語言與版本	描述
`kinesis-analytics-output`	Node.js 12.x	將輸出記錄從 Kinesis Data Analytics 應用程式傳送至自訂目的地。

在 Python 中建立一個 Lambda 函數目的地

主控台提供在 Python 中建立目的地 Lambda 函數的下列範本：

Lambda 作為輸出藍圖	語言與版本	描述
`kinesis-analytics-output-sns`	Python 2.7	將輸出記錄從 Kinesis Data Analytics 應用程式傳送至 Amazon SNS。
`kinesis-analytics-output-ddb`	Python 2.7	將輸出記錄從 Kinesis Data Analytics 應用程式傳送至 Amazon DynamoDB。

在 Java 中建立一個 Lambda 函數目的地

如要在 Java 中建立目的地 Lambda 函數，請使用 Java 事件類別。

下列程式碼會示範使用 Java 的範例目的地 Lambda 函數：

```
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 中建立一個 Lambda 函數目的地

如要在 .NET 中建立目的地 Lambda 函數，請使用 .NET 事件類別。

下列程式碼會示範使用 C# 的範例目的地 Lambda 函數：

```
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;
        }
    }
```

如需在 .NET 中建立用於預先處理和目的地的 Lambda 函數詳細資訊，請參閱 Amazon.Lambda.KinesisAnalyticsEvents。

將應用程式輸出保存至外部目標的交付模型

Amazon Kinesis Data Analytics 使用「至少一次」交付模型，將應用程式輸出到設定的目的地。當應用程式執行時，Kinesis Data Analytics 會採用內部檢查點。這些檢查點是輸出記錄傳遞至目的地，且沒有資料遺失的時間點。服務會視需要使用檢查點，以確保應用程式輸出至少會傳送一次至設定的目的地。

在正常情況下，您的應用程式會持續處理傳入的資料。Kinesis Data Analytics 會將輸出寫入設定的目標，例如 Kinesis 資料串流或 Firehose 交付串流。不過，您的應用程式可能會偶爾中斷，例如：

- 您選擇停止應用程式並稍後重新啟動。
- 您刪除了 Kinesis Data Analytics 將應用程式輸出寫入設定目的地所需的 IAM 角色。如果沒有 IAM 角色，Kinesis Data Analytics 就沒有任何權限代您寫入外部目的地。
- 網路中斷或其他內部服務失敗，導致應用程式暫時停止執行。

當您的應用程式重新啟動時，Kinesis Data Analytics 可確保從失敗發生之前或當下時間點繼續處理和寫入輸出，。此舉能確保傳送到設定目的地的應用程式輸出不會有遺漏。

假設您從相同的應用程式內串流設定了多個目的地。應用程式從失敗復原後，Kinesis Data Analytics 會從上次傳遞至最慢目的地的記錄，繼續保留輸出至設定目的地。這可能會導致相同的輸出記錄多次傳送到其他目的地。在這種情況下，您必須在外部處理目的地的潛在重複。

錯誤處理

Amazon Kinesis Data Analytics 會直接傳回 API 或 SQL 錯誤給您。如需 API 操作的詳細資訊，請參閱動作。如需有關處理 SQL 錯誤的詳細資訊，請參閱 Amazon Kinesis Data Analytics SQL 參考資料。

Amazon Kinesis Data Analytics 會使用名為 `error_stream` 的應用程式內錯誤串流來報告執行期錯誤。

使用應用程式內錯誤串流回報錯誤

Amazon Kinesis Data Analytics 會使用名為 `error_stream` 的應用程式內錯誤串流來報告執行期錯誤。以下是可能發生的錯誤範例：

- 從串流來源讀取的記錄不符合輸入結構描述。
- 您的應用程式碼會指定除以零。
- 資料列失序 (例如，串流上出現一筆帶有 `ROWTIME` 值的記錄，使用者修改該值導致記錄失序)。
- 來源串流中的資料，無法轉換為結構描述中指定的資料類型 (強制錯誤)。若要取得可轉換哪些資料類型的資訊，請參閱 將 JSON 資料類型映射到 SQL 資料類型。

我們建議您以程式設計方式在 SQL 程式碼中處理這些錯誤，或將錯誤串流上的資料保存到外部目的地。此舉需要您將輸出配置 (請參閱 設定應用程式輸出) 添加到應用程式中。如需應用程式內錯誤串流運作方式的範例，請參閱 範例：探索應用程式內錯誤串流。

> **Note**
>
> Kinesis Data Analytics 應用程式無法以程式設計方式存取或修改錯誤串流，因為錯誤串流是使用系統帳戶建立的。您必須使用錯誤輸出來判斷應用程式可能會遇到的錯誤。然後，您可以撰寫應用程式的 SQL 程式碼來處理預期的錯誤狀況。

錯誤串流結構描述

錯誤串流具有以下結構描述：

欄位	資料類型	備註

ERROR_TIME	TIMESTAMP	發生錯誤的時間
ERROR_LEVEL	VARCHAR(10)	
ERROR_NAME	VARCHAR(32)	
MESSAGE	VARCHAR(4096)	
DATA_ROWTIME	TIMESTAMP	傳入記錄的列時間
DATA_ROW	VARCHAR(49152)	原始資料列中的十六進位編碼資料。您可以使用標準程式庫為此值進行十六進位解碼，或者使用 Web 資源，例如此十六進位到字串轉換器。
PUMP_NAME	VARCHAR(128)	原始幫浦，如 `CREATE PUMP` 之定義

自動擴展應用程式以增加輸送量

Amazon Kinesis Data Analytics 可彈性擴展您的應用程式，以容納來源串流資料的輸送量，以及大多數情況下的查詢複雜性。Kinesis Data Analytics 以 Kinesis 處理單元 (KPU) 的形式提供容量。單一 KPU 可提供您記憶體 (4 GB)，以及對應的運算和網路。

應用程式的 KPU 預設限制為 64。如需如何請求提高此限制的指示，請參閱 Amazon 服務配額中的請求提高限制。

使用標記

本節說明如何將索引鍵值中繼資料標籤新增至 Kinesis Data Analytics 應用程式。這些標籤可用於下列目的：

- 決定個別 Kinesis Data Analytics 應用程式的計費。如需詳細資訊，請參閱《AWS 帳單和成本管理使用者指南》中的使用成本分配標籤。
- 根據標籤控制對應用程式資源的存取。如需詳細資訊，請參閱《使用者指南》中的使用標籤控制存取權。
- 使用者定義的目的。您可以根據使用者標籤的存在來定義應用程式的功能。

注意有關標記的以下資訊：

- 應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。
- 如果動作包含具有重複 Key 值的標籤清單，則服務會擲出 InvalidArgumentException。

本主題包含下列章節：

建立應用程式時新增標籤

您可以在使用 CreateApplication 動作的 tags 參數建立應用程式時新增標籤。

以下範例請求顯示 CreateApplication 請求的 Tags 節點：

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

為現有應用程式新增或更新標籤

您可以使用 TagResource 動作將標籤新增至應用程式。您無法使用 UpdateApplication 動作為應用程式新增標籤。

若要更新現有的標籤，請使用與現有標籤相同的索引鍵新增標籤。

TagResource 動作的下列請求範例會新增標籤或更新現有標籤：

```
{
```

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

列出應用程式的標籤

若要列出現有標籤，請使用 ListTagsForResource 動作。

ListTagsForResource 動作的下列請求範例可列出應用程式的標籤：

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

從應用程式移除標籤

若要從應用程式移除標籤，請使用 UntagResource 動作。

UntagResource 動作的下列請求範例可移除應用程式的標籤：

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

Amazon Kinesis Data Analytics for SQL 應用程式入門

以下主題可協助您開始使用 Amazon Kinesis Data Analytics for SQL 應用程式。如果您是 Kinesis Data Analytics for SQL 應用程式的新手，建議您在執行入門章節中的步驟之前，先檢閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式 中介紹的概念和術語。

主題

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

步驟 1：設定 帳戶並建立管理員使用者

首次使用 Amazon Kinesis Data Analytics 之前，請先完成下列任務：

1. 註冊 AWS

2. 建立 IAM 使用者

註冊 AWS

註冊 Amazon Web Services 時，您的 AWS 帳戶 帳戶會自動註冊 AWS 的所有服務，包括 Amazon Kinesis Data Analytics。您只需支付實際使用服務的費用。

使用 Kinesis Data Analytics 時，您僅需按使用的資源量付費。如果您是 AWS 新客戶，可免費開始使用 Kinesis Data Analytics。如需更多詳細資訊，請參閱 AWS 免費用量方案。

如果您已擁有 AWS 帳戶，請跳至下一項任務。如果您沒有 AWS 帳戶，請執行下列程序中的步驟建立一個帳戶。

建立 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

請記下您的 AWS 帳戶 帳戶 ID，因為您下一個任務會需要此 ID。

建立 IAM 使用者

AWS 中的服務 (例如 Amazon Kinesis Data Analytics) 需要您在存取時提供登入資料，讓服務可以判斷您是否有權存取該服務所擁有的資源。主控台需要您的密碼。您可以建立 AWS 帳戶 的存取金鑰，用以存取 AWS CLI 或 API。不過，不建議您使用 AWS 帳戶 的登入資料來存取 AWS。建議您改用 AWS Identity and Access Management (IAM)。建立 IAM 使用者，並將使用者新增至擁有管理許可的 IAM 群組，然後將管理許可授予您建立的 IAM 使用者。您可以使用特殊 URL 與該 IAM 使用者的登入資料來存取 AWS。

如果您已註冊 AWS，但是尚未建立自己的 IAM 使用者，則可以使用 IAM 主控台建立使用者。

本指南中的「入門」練習假設您有具備管理員權限的使用者 (`adminuser`)。請遵循程序在您的帳戶中建立 `adminuser`。

建立管理員使用者並登入主控台

1. 在您的 AWS 帳戶 中建立一個名為 adminuser 的管理員使用者。如需說明，請前往《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。
2. 使用者可以使用特殊的 URL 登入 AWS Management Console。如需更多詳細資訊，請參閱《IAM 使用者指南》中的使用者如何登入您的帳戶。

如需 IAM 的詳細資訊，請參閱下列各項：

- AWS Identity and Access Management (IAM)
- 入門
- IAM 使用者指南

後續步驟

步驟 2：設定 AWS Command Line Interface (AWS CLI)

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

步驟 2：設定 AWS Command Line Interface (AWS CLI)

請遵循下列步驟來下載及設定 AWS Command Line Interface (AWS CLI)。

> **Important**
>
> 執行「入門」練習中的步驟不需用到 AWS CLI。不過，本指南中的某些練習會用到 AWS CLI。您可以跳過這個步驟並前往 步驟 3：建立 Amazon Kinesis Data Analytics 入門應用程式。，之後有需要時再設定 AWS CLI。

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

   - 設定 AWS Command Line Interface
   - 設定 AWS Command Line Interface

2. 在 AWS CLI 組態檔中，為管理員使用者新增命名描述檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用 AWS 區域 的清單，請參閱《Amazon Web Services 一般參考》中的區域與端點

3. 在命令提示字元中輸入下列 help 命令，以驗證設定：

   ```
   aws help
   ```

後續步驟

步驟 3：建立 Amazon Kinesis Data Analytics 入門應用程式。

步驟 3：建立 Amazon Kinesis Data Analytics 入門應用程式。

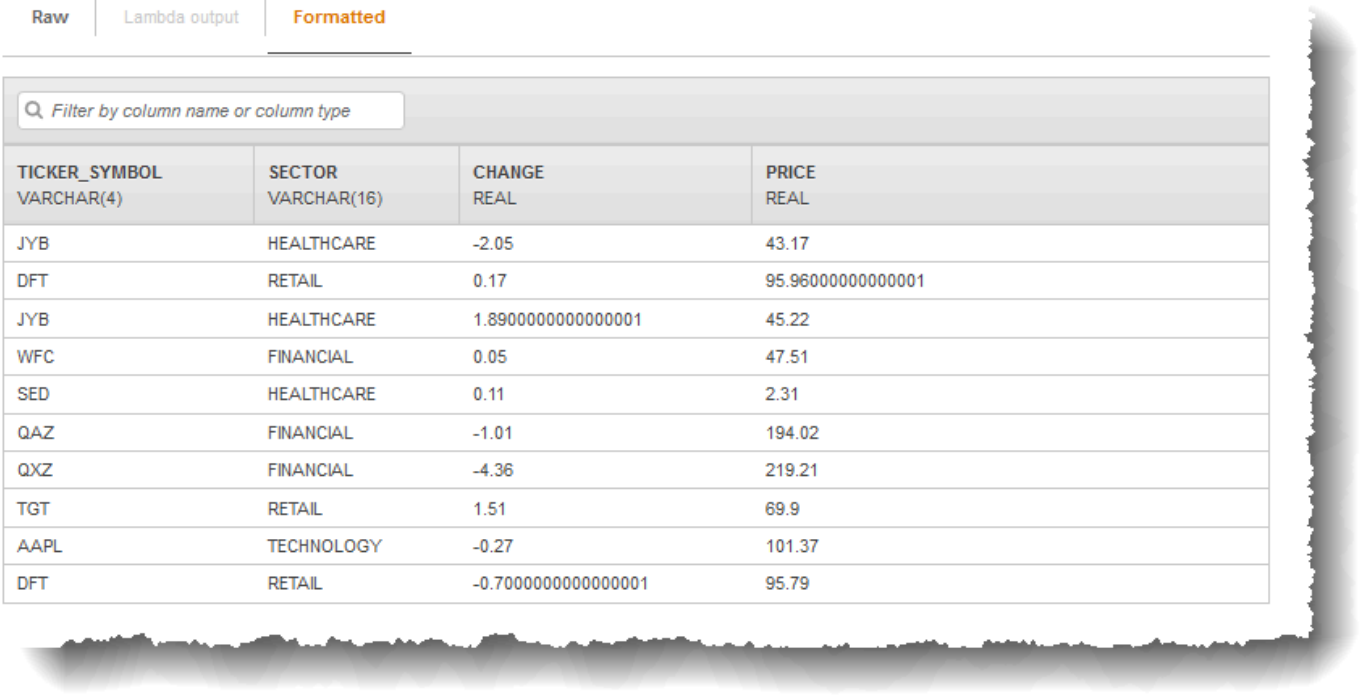
按照本節中的步驟操作，您可以使用主控台建立第一個 Kinesis Data Analytics 應用程式。

> **Note**
>
> 我們建議您在嘗試入門練習之前先檢閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式。

在這個入門練習中，您可以使用主控台來處理示範串流或包含應用程式碼的範本。

- 如果您選擇使用示範串流，主控台會在您的帳戶中建立名為 `kinesis-analytics-demo-stream` 的 Kinesis 資料串流。

  Kinesis Data Analytics 應用程式需要串流來源。針對此來源，本指南中的數個 SQL 範例會使用示範串流 `kinesis-analytics-demo-stream`。控制台還運行一個指令碼，將樣本資料（模擬股票交易記錄）持續添加到此串流中，如下所示。



  在本練習中，您可以把 `kinesis-analytics-demo-stream` 當作應用程式的串流來源。

> **Note**
>
> 示範串流會保留在您的帳戶中。您可以用其測試本指南中的其他範例。不過，當您離開主控台時，主控台使用的指令碼會停止填入資料。當需要時，控制台提供了開始再次填充串流的選項。

- 如果您選擇將範本與應用程式碼範例搭配使用，可使用主控台提供的範本程式碼，在示範串流上執行簡單的分析。

您可以使用這些功能來快速設定第一個應用程式，如下所示：

1. 建立應用程式：您只需要提供名稱即可。主控台會建立應用程式，且服務會將應用程式狀態設定為 READY。

2. 設定輸入：首先添加一個串流來源，即示範串流。您必須先在主控台中建立示範串流，才能使用它。然後，主控台會在示範串流上擷取隨機記錄樣本，並針對所建立的應用程式內輸入串流推斷結構描述。主控台會將應用程式內串流命名為 SOURCE_SQL_STREAM_001。

   主控台會使用探索 API 來推斷結構描述。如有必要，您可以編輯推斷的結構描述。如需詳細資訊，請參閱 DiscoverInputSchema。Kinesis Data Analytics 會使用此結構描述來建立應用程式內串流。

   當應用程式啟動時，Kinesis Data Analytics 會代表您持續讀取示範串流，並將資料列插入 SOURCE_SQL_STREAM_001 應用程式內串流。

3. 指定應用程式碼：使用提供下列程式碼的範本 (稱為持續篩選條件)：

```
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%';
```

應用程式碼會查詢應用程式內串流 SOURCE_SQL_STREAM_001。然後，程式碼會使用幫浦將產生的資料列插入另一個應用程式內串流 DESTINATION_SQL_STREAM 中。如需此編碼模式的更多資訊，請參閱 應用程式碼。

如需 Kinesis Data Analytics 支援之 SQL 語言元素的相關資訊，請參閱 Amazon Kinesis Data Analytics SQL 參考資料。

4. 設定輸出：在本練習中，不會設定任何輸出。也就是說，您不會將應用程式建立的內部串流資料保存到任何外部目的地。而是在主控台中驗證查詢結果。本指南中的其他範例說明如何設定輸出。如需範例，請參閱 範例：建立簡單提醒。

> **Important**
>
> 此練習使用美國東部 (維吉尼亞北部) 區域 (us-east-1) 來設定應用程式。您可以使用任何支援的 AWS 區域。

後續步驟

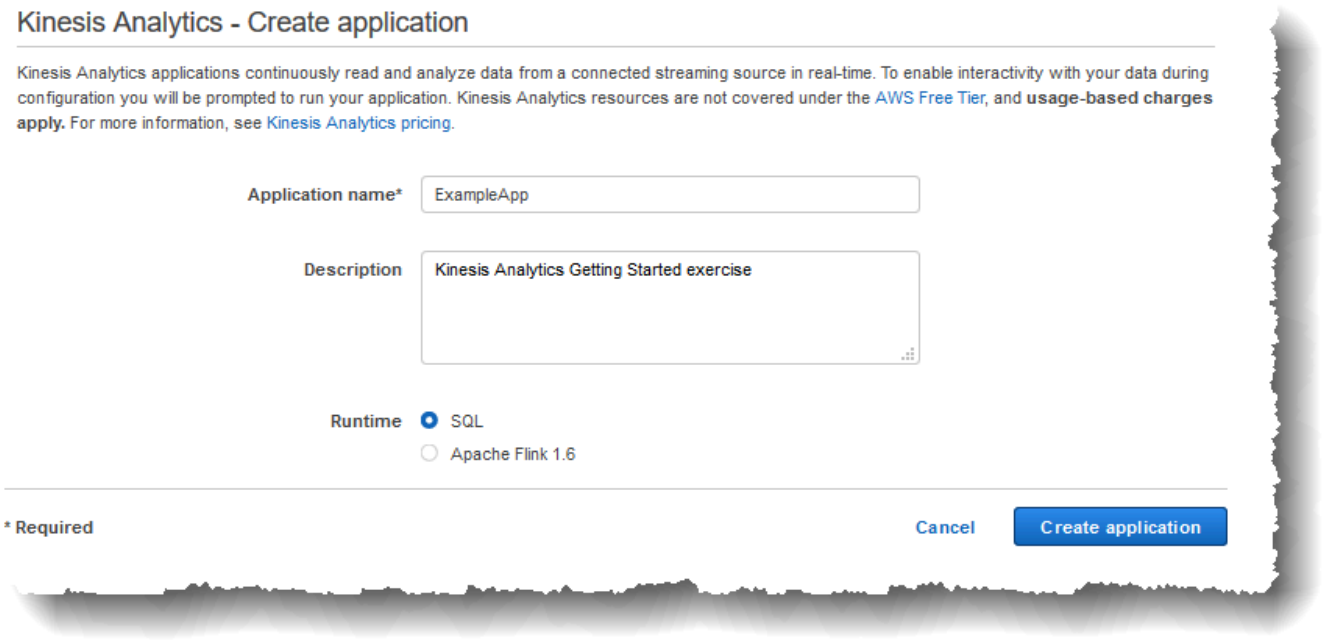
步驟 3.1：建立應用程式

步驟 3.1：建立應用程式

在本節建立 Amazon Kinesis Data Analytics 應用程式。在下一個步驟設定應用程式輸入。

若要建立資料分析應用程式

1. 前往https://console.aws.amazon.com/kinesisanalytics 登入 AWS Management Console，並開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式 。
3. 在建立應用程式頁面上，輸入應用程式名稱、輸入描述、在應用程式的執行期設定選擇 SQL，然後選擇建立應用程式。



此舉會建立狀態為 READY 的 Kinesis Data Analytics 應用程式。主控台顯示應用程式中樞，您可以在其中設定輸入和輸出。

> **Note**
>
> 若要建立應用程式，CreateApplication 作業只需要應用程式名稱。在主控台建立應用程式後，您可以新增輸入和輸出設定。

在下一個步驟設定應用程式輸入。在輸入組態中，您可以將串流資料來源新增至應用程式，並透過取樣串流來源的資料，來探索應用程式內輸入串流的結構描述。

後續步驟

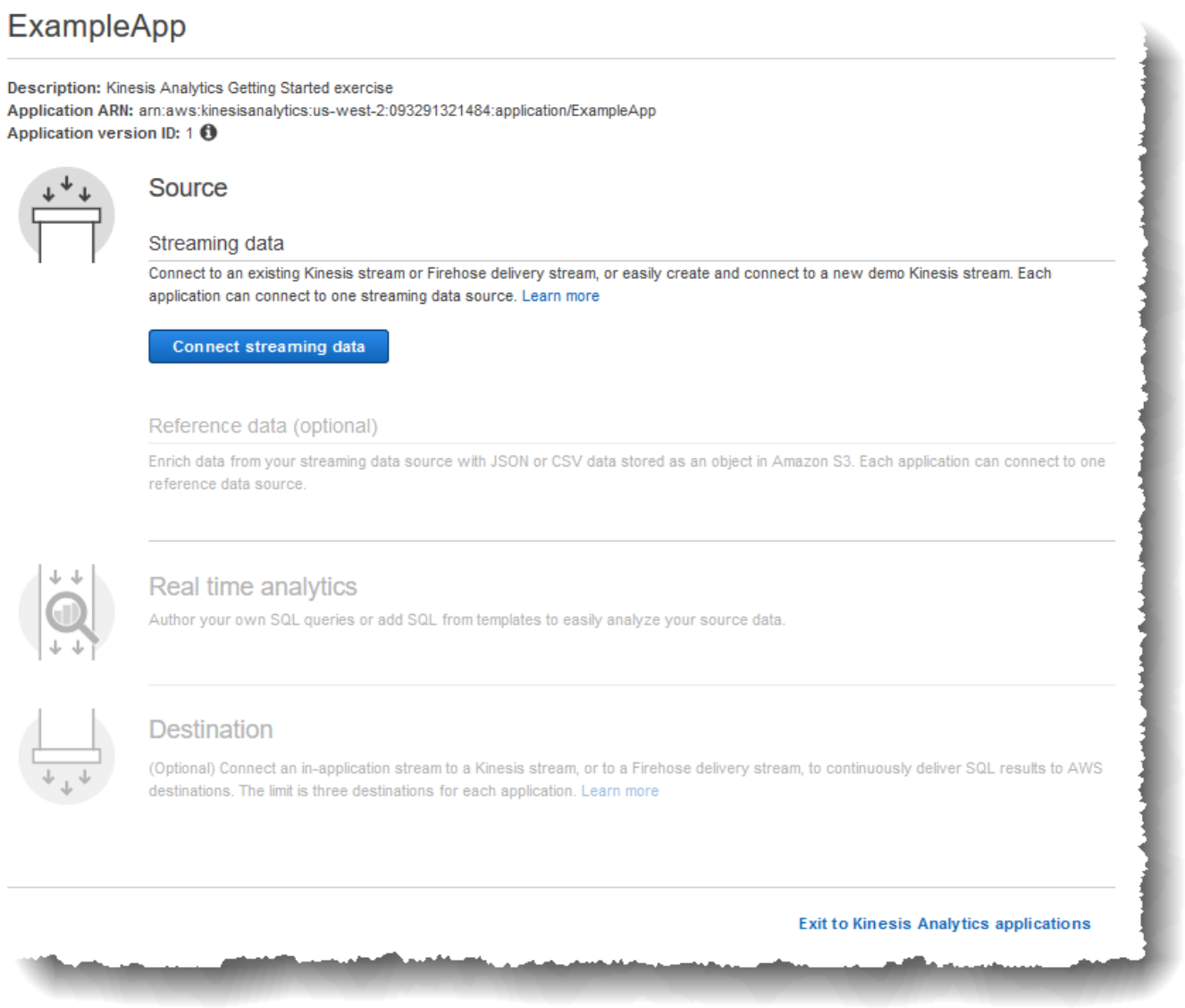
步驟 3.2：設定輸入

步驟 3.2：設定輸入

您的應用程式需要串流來源。為了協助您開始使用，主控台可以建立示範串流 (稱為 `kinesis-analytics-demo-stream`)。主制台還運行一個指令碼，用於填充串流中的記錄。

將串流來源新增至您的應用程式

1. 在應用程式中樞頁面，選擇連接串流資料來連接至來源。



2. 在顯示的頁面上，檢視下列項目：

   - 來源區段，您可以在此指定應用程式的串流來源。您可以選取現有的串流來源或建立一個。在本練習中建立新串流，即示範串流。

     依預設，主控台會將建立的應用程式內輸入串流命名為 INPUT_SQL_STREAM_001。針對本練習，請保留原來的名稱。

- 串流參考名稱：此選項會顯示建立的應用程式內輸入串流名稱 SOURCE_SQL_STREAM_001。您可以變更名稱，但在本練習中，請保留此名稱。

  在此輸入組態中，將此示範串流映射到建立的應用程式內串流。當應用程式啟動時，Amazon Kinesis Data Analytics 會持續讀取示範串流，並將資料列插入應用程式內輸入串流。您可以在應用程式碼中查詢此應用程式內輸入串流。

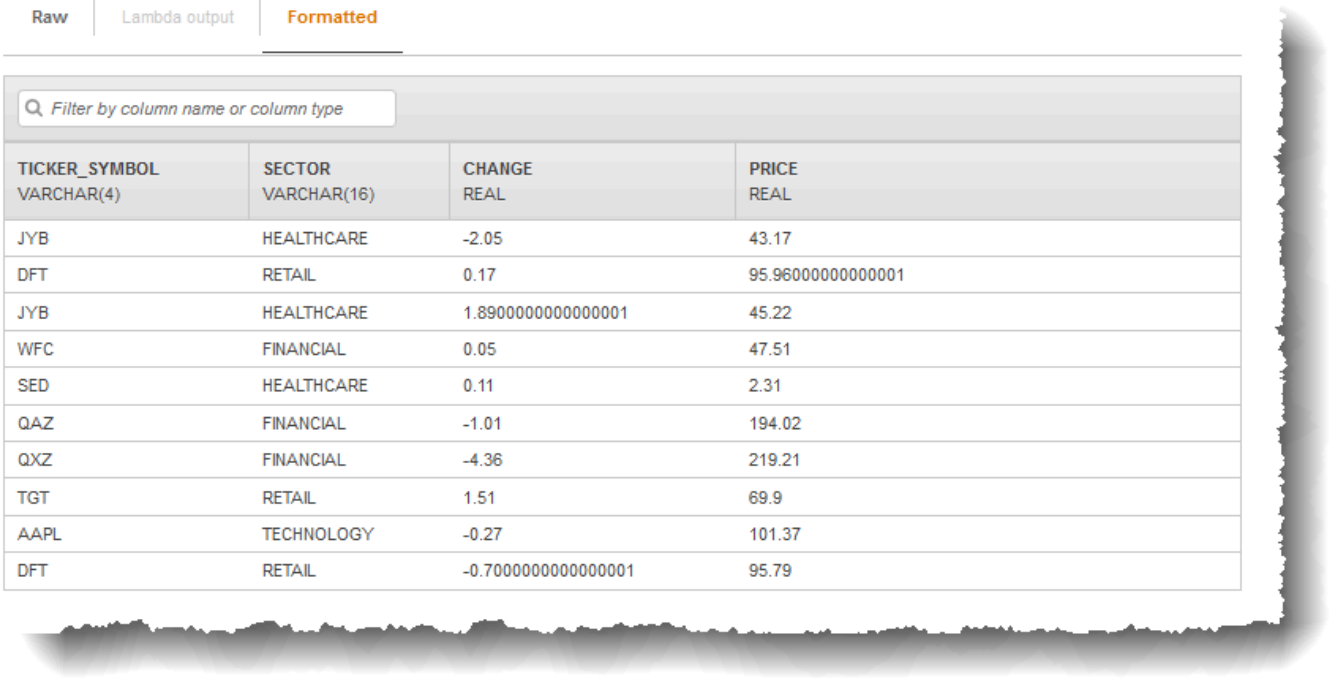
- 用 AWS Lambda 記錄預處理方式：這個選項讓您指定 AWS Lambda 運算式，該運算式可在應用程式碼執行之前修改輸入串流中的記錄。在本練習中，保持選取已停用選項。如需 Lambda 預處理的詳細資訊，請參閱 使用 Lambda 函數預處理資料。

在您提供此頁面上的所有資訊之後，主控台會傳送更新要求 (請參閱 UpdateApplication) 以新增應用程式的輸入組態。

3. 在來源頁面上，選擇設定新串流。
4. 選擇建立示範串流。主控台會執行下列操作以設定應用程式輸入：

   - 主控台會建立名為 kinesis-analytics-demo-stream 的 Kinesis 資料串流。
   - 控制台用股票代號資料範例填充串流。
   - 使用 DiscoverInputSchema 輸入動作，主控台會讀取串流上的範例記錄來推斷結構描述。所推斷的結構描述，即為建立的應用程式內輸入串流的結構描述。如需詳細資訊，請參閱 設定應用程式輸入。
   - 主控台會顯示推斷的結構描述，以及從串流來源讀取的範例資料，藉此推斷結構描述。

   主控台會顯示串流來源上的範例記錄。



以下內容會顯示在串流範例主控台頁面上：

- 原始串流範例標籤會顯示 DiscoverInputSchema API 動作取樣的原始串流記錄，以推斷結構描述。
- 格式化串流範例標籤會在原始串流範例標籤中顯示資料的表格式版本。
- 如果選擇編輯結構描述，則可以編輯推斷的結構描述。在本練習中，請勿變更推斷的結構描述。如需有關編輯結構描述的詳細資訊，請參閱 使用結構描述編輯器。

  如果您選擇重新探索結構描述，則可以要求主控台再次執行 DiscoverInputSchema 並推斷結構描述。

5. 選擇儲存並繼續。

   現在您得到一個加入了輸入組態的應用程式。在下一個步驟中新增 SQL 程式碼，以對應用程式內的資料輸入串流執行一些分析。

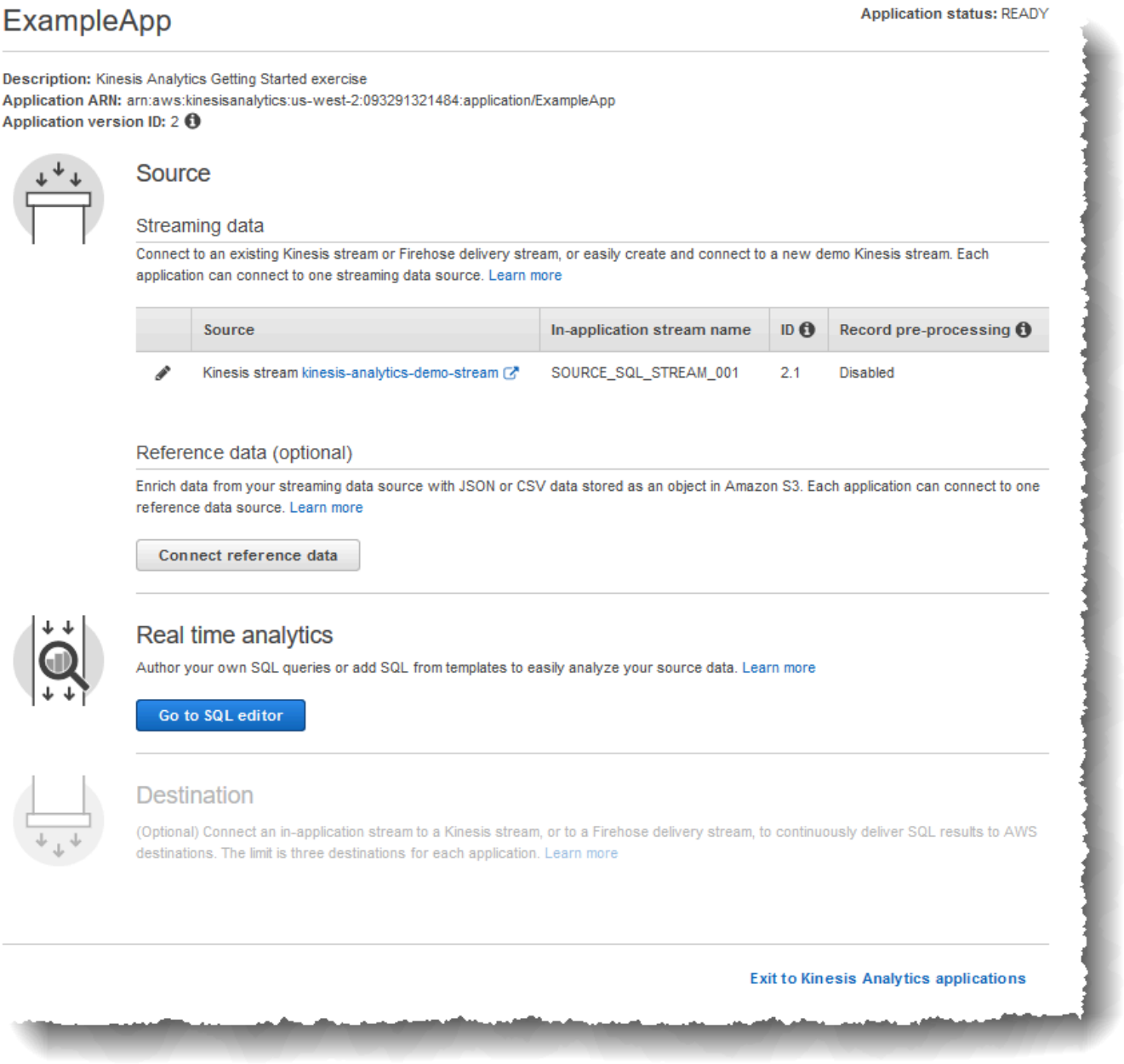
後續步驟

步驟 3.3：新增實時分析 (新應用程式碼)

步驟 3.3：新增實時分析 (新應用程式碼)

您可以針對應用程式內串流撰寫自己的 SQL 查詢，但在接下來的步驟中，您可以使用其中一個提供範例程式碼的範本。

1. 在應用程式中樞頁面，選擇至 SQL 編輯器。



2. 在 [您要開始執行 "ExampleApp「嗎？」對話方塊中，選擇是，啟動應用程式。

   主控台會傳送啟動應用程式的要求 (請參閱 StartApplication)，然後 SQL 編輯器頁面就會出現。

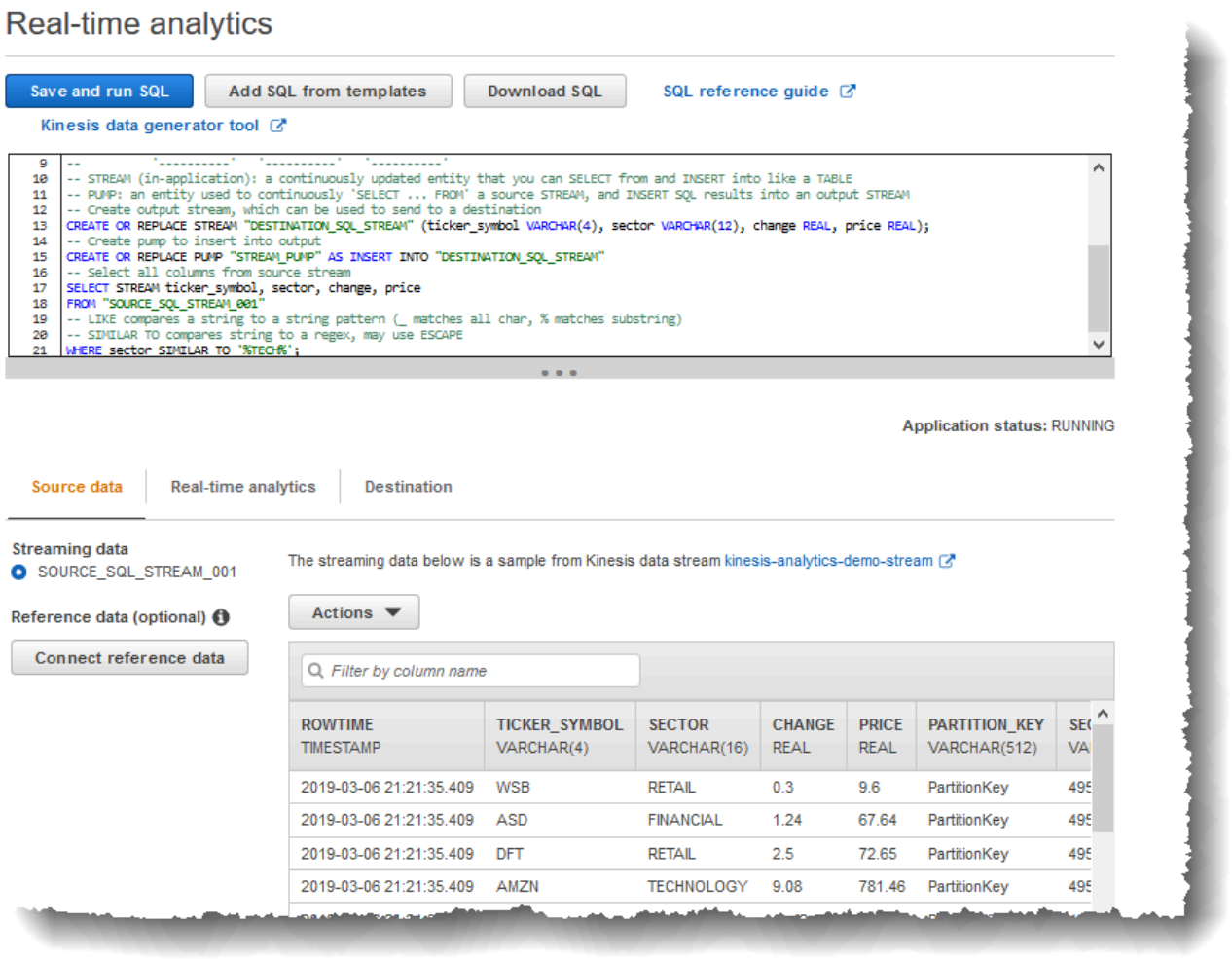
3. 主控台會開啟 SQL 編輯器頁面。檢閱頁面，包括按鈕 (從範本新增 SQL、儲存並執行 SQL) 和各種標籤。
4. 在 SQL 編輯器中，選擇從範本新增 SQL。
5. 從可用範本清單中，選擇連續篩選。範例程式碼會從一個應用程式內串流讀取資料 (WHERE 子句會篩選資料列)，並將其插入另一個應用程式內串流，如下所示：

   - 此舉會建立應用程式內串流 DESTINATION_SQL_STREAM。
   - 此舉會建立幫浦 STREAM_PUMP，並用它從 SOURCE_SQL_STREAM_001 中選取列並插入 DESTINATION_SQL_STREAM。

6. 選擇將此 SQL 新增至編輯器。
7. 依照下列方式來測試應用程式碼：

   請記住，您已經啟動了應用程式（狀態為 RUNNING）。儲存組態時，Amazon Kinesis Data Analytics 已持續從串流來源讀取資料，並將資料列加入應用程式內串流 SOURCE_SQL_STREAM_001。

   a. 在 SQL 編輯器中，選擇儲存並執行 SQL。控制台首先發送更新請求，以保存應用程式碼。然後，程式碼會持續執行。
   b. 您可以在即時分析標籤中查看結果。



SQL 編輯器包含下列標籤：

- 來源資料標籤會顯示映射至串流來源的應用程式內輸入串流。選擇應用程式內串流，您就可以看到傳入的資料。請注意應用程式內輸入串流中的其他欄位，輸入組態並未指定這些欄位。其中包括下列時間戳記欄：

  - ROWTIME：應用程式內串流中的每一列都有一個名為 ROWTIME 的特殊欄。此欄是 Amazon Kinesis Data Analytics 在第一個應用程式內串流 (對應至串流來源) 中插入資料列的時間戳記。

  - Approximate_Arrival_Time：每個 Kinesis Data Analytics 記錄都包含一個稱為 `Approximate_Arrival_Time` 的值。當串流來源成功接收並儲存記錄時，此值即為大

約的到達時間戳記。Kinesis Data Analytics 從串流來源讀取記錄時，會將此欄擷取到應用程式內輸入串流中。

這些時間戳記值在以時間為基礎的窗口化查詢中非常有用。如需詳細資訊，請參閱 窗口化查詢。

- 即時分析標籤會顯示應用程式碼建立的所有其他應用程式內串流。其中還包括錯誤串流。Kinesis Data Analytics 會將任何無法處理的資料列傳送至錯誤串流。如需詳細資訊，請參閱 錯誤處理。

  選擇 DESTINATION_SQL_STREAM 檢視應用程式碼插入的資料列。請注意應用程式碼未建立的其他資料欄。其中包括 ROWTIME 時間戳記欄：Kinesis Data Analytics 只會從來源 (SOURCE_SQL_STREAM_001) 複製這些值。

- 目的地標籤會顯示 Kinesis Data Analytics 寫入查詢結果的外部目標。您尚未為應用程式輸出設定任何外部目的地。

後續步驟

步驟 3.4：(選用) 更新應用程式碼

步驟 3.4：(選用) 更新應用程式碼

在此步驟中，探索如何更新應用程式碼。

如要更新應用程式碼

1. 建立另一個應用程式內串流，如下所示：

   - 建立另一個名為 DESTINATION_SQL_STREAM_2 的應用程式內串流。
   - 建立幫浦，然後透過從 DESTINATION_SQL_STREAM 中選取列，用幫浦在新建立的串流中插入列。

   在 SQL 編輯器中，將下列程式碼附加至現有的應用程式碼：

```
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";
```

儲存並執行程式碼。其他應用程式內串流會顯示在即時分析標籤上。

2. 建立兩個應用程式內串流。根據股票代號篩選 SOURCE_SQL_STREAM_001 中的列，然後將它們插入到這些單獨的串流中。

   將下列 SQL 陳述式附加至您的應用程式碼：

```
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%';
```

儲存並執行程式碼。請注意即時分析標籤上的其他應用程式內串流。

現在您即獲得第一個可運作的 Amazon Kinesis Data Analytics 應用程式。在本練習中，您進行了以下動作：

- 建立了您的第一個 Kinesis Data Analytics 應用程式。

- 設定分應用程式輸入，將示範串流識別為串流來源，並將其映射至建立的應用程式內串流 (SOURCE_SQL_STREAM_001)。Kinesis Data Analytics 會持續讀取示範串流，並將記錄插入應用程式內串流。

- 您的應用程式碼查詢了 SOURCE_SQL_STREAM_001，並將輸出寫入另一個名為 DESTINATION_SQL_STREAM 的應用程式內串流。

現在，您可以選擇性地設定應用程式輸出，以將其寫入外部目的地。換句話說，您可以設定應用程式輸出，將 DESTINATION_SQL_STREAM 的紀錄寫入外部目的地。針對本練習，此為選用步驟。若要瞭解如何設定目的地，請至下一個步驟。

後續步驟

步驟 4 (選用)：使用主控台編輯結構描述和 SQL 程式碼.

步驟 4 (選用)：使用主控台編輯結構描述和 SQL 程式碼

以下，您可以找到如何編輯推斷的架構描述，以及如何編輯 Amazon Kinesis Data Analytics 的 SQL 程式碼之相關資訊。您可以使用結構描述編輯器和 SQL 編輯器來達成此目的，後兩者屬於 Kinesis Data Analytics 主控台的一部分。

> **Note**
>
> 若要在主控台中存取或取樣資料，登入使用者的角色必須具有 kinesisanalytics:GetApplicationState 權限。如需 Kinesis Data Analytics 應用程式權限的詳細資訊，請參閱 管理存取概觀。

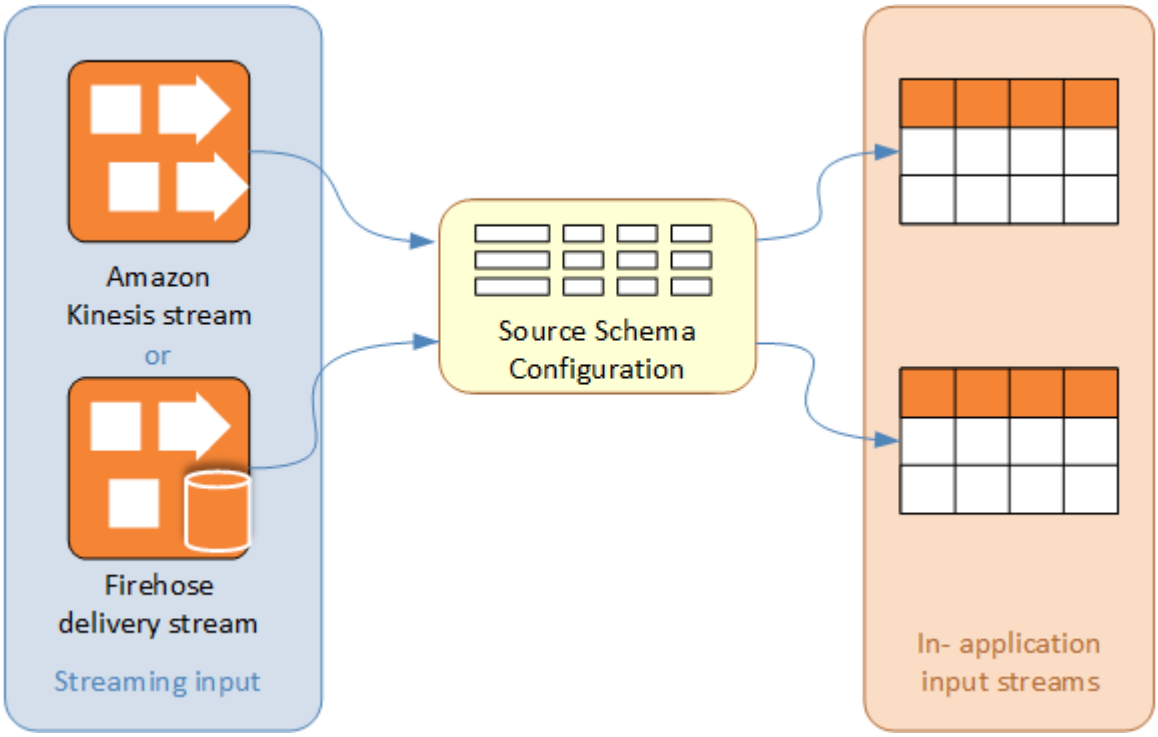
主題

- 使用結構描述編輯器

- 使用 SQL 編輯器

使用結構描述編輯器

Amazon Kinesis Data Analytics 應用程式輸入串流的結構描述，定義了串流中的資料如何開放給應用程式中的 SQL 查詢。



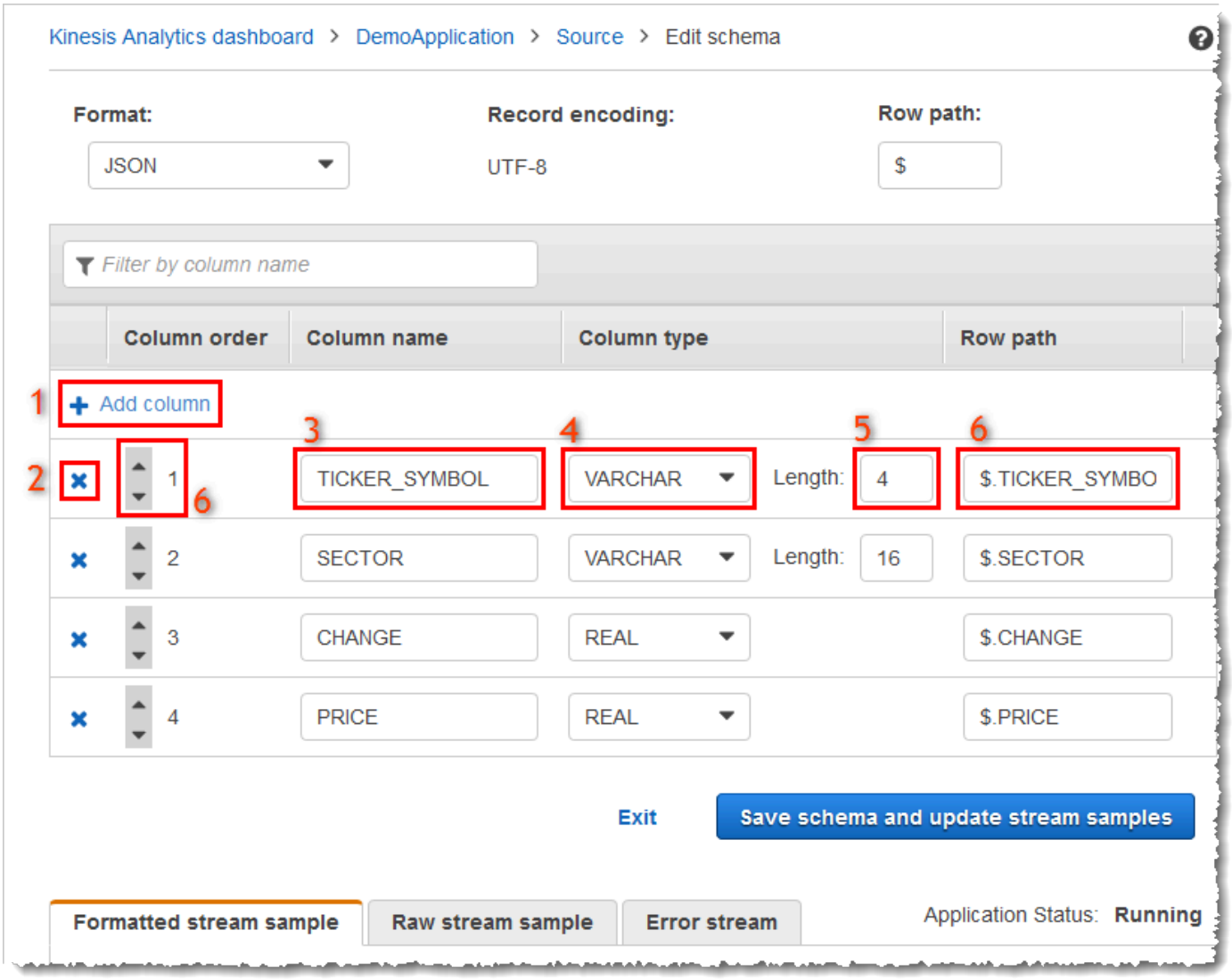
結構描述包含選取準則，可決定串流輸入的哪一部分要轉換為應用程式內輸入串流中的資料欄。此輸入可為下列之一：

- JSON 輸入資料流的 JSON 路徑運算式。JSONPath 是用於查詢 JSON 資料的工具。
- 逗點分隔值 (CSV) 格式的輸入串流之資料欄編號。
- 用於在應用程式內資料串流中提供資料的欄名稱和 SQL 資料類型。資料類型也包含字元或二進位資料的長度。

主控台嘗試使用 DiscoverInputSchema 產生結構描述。如果結構描述探索失敗，或傳回不正確或不完整的結構描述，您必須使用結構描述編輯器手動編輯結構描述。

結構描述編輯器主畫面

下列螢幕截圖顯示結構描述編輯器的主畫面。



您可以將下列編輯套用至結構描述：

- 新增欄 (1)：如果未自動偵測到資料項目，您可能需要新增資料欄。
- 刪除欄 (2)：如果應用程式不需要資料，您可以從來源串流中排除資料。此排除不會影響來源串流中的資料。如果排除資料，該資料就無法供應用程式使用。
- 重新命名欄位 (3)。資料欄名稱不能為空白、長度必須超過單一字元，且不得包含保留的 SQL 關鍵字。名稱也必須符合 SQL 一般識別符的命名條件：名稱必須以字母開頭，且只包含字母、底線字元和數字。
- 變更資料欄的資料類型 (4) 或長度 (5)：您可以為欄指定相容的資料類型。如果您指定不相容的資料類型，則資料欄會填入 NULL，或者完全不會填入應用程式內串流。在後者情況下，錯誤被寫入錯誤串流。如果您指定的欄長度太短，則會截斷傳入資料。

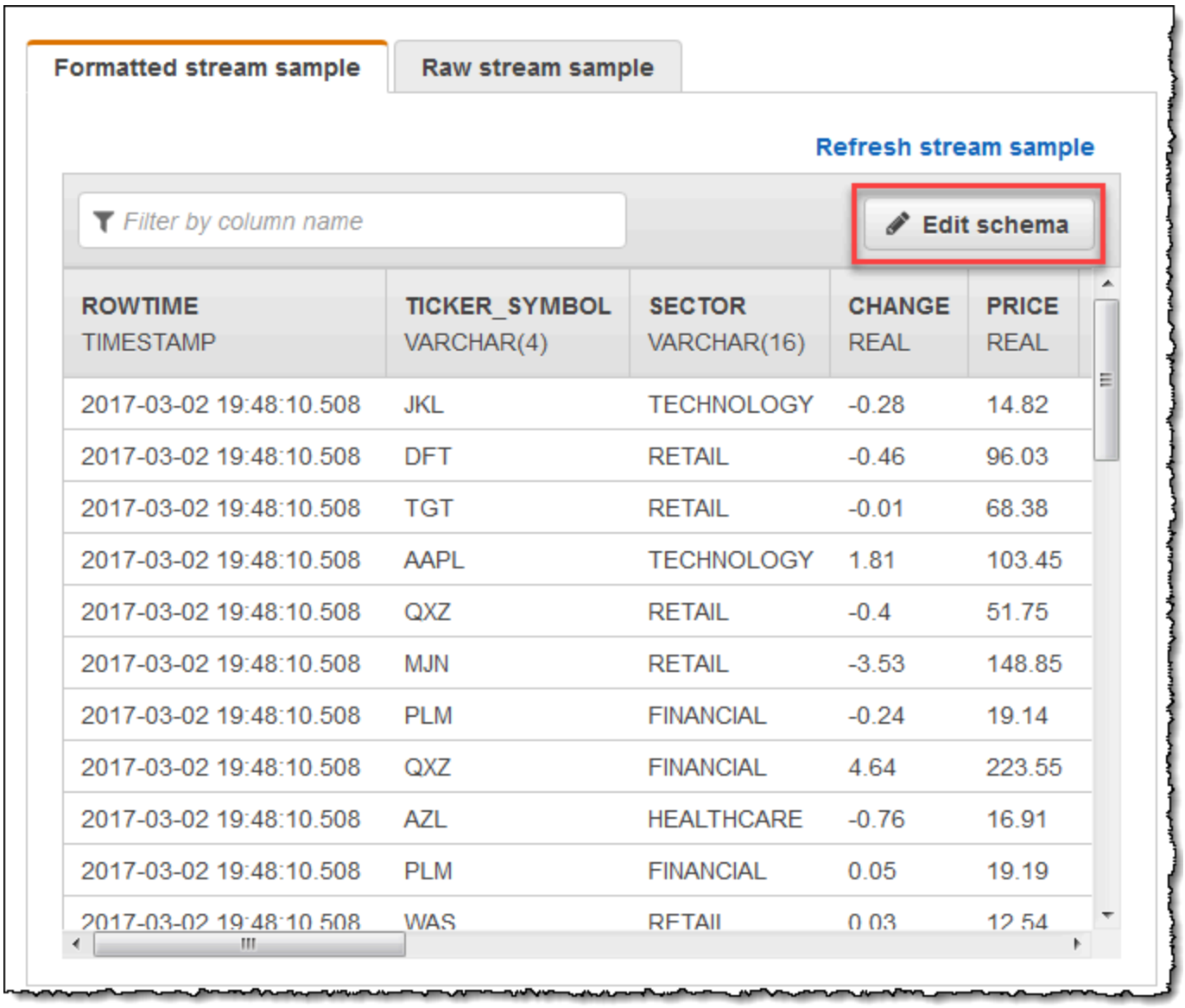
- 變更欄的選取條件 (6)：您可以編輯 JSONPath 運算式或 CSV 欄順序，藉此決定欄中資料來源。若要變更 JSON 結構描述的選取條件，請輸入列路徑運算式的新值。CSV 結構描述使用欄順序作為選取條件。若要變更 CSV 結構描述的選取條件，請變更欄的順序。

編輯串流來源的結構描述

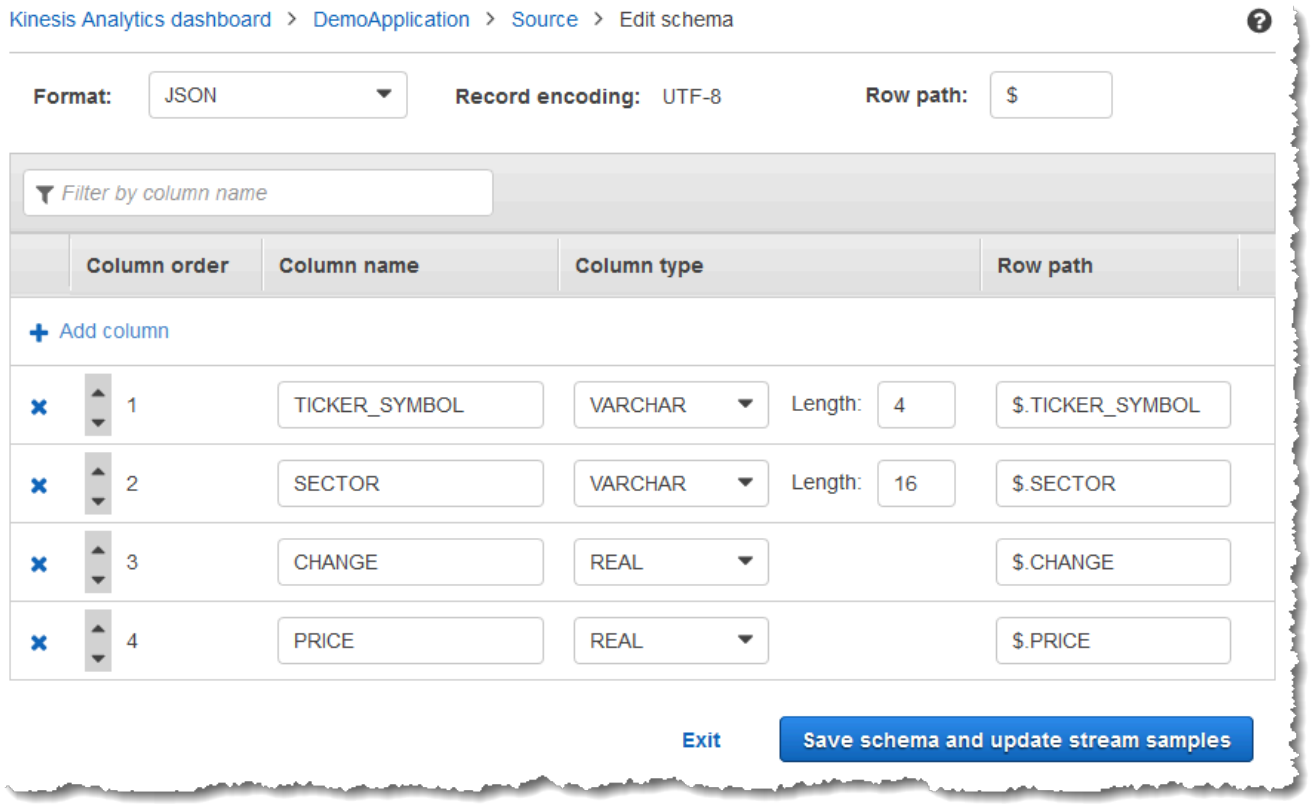
如果您需要編輯串流來源的結構描述，請依照下列步驟執行。

如要編輯串流來源的結構描述

1. 在來源頁面上，選擇編輯結構描述。



2. 在編輯結構描述頁面，編輯來源結構描述。



3. 在格式，選擇 JSON 或 CSV。針對 JSON 或 CSV 格式，支援的編碼是 ISO 8859-1。

如需有關編輯 JSON 或 CSV 格式結構描述的詳細資訊，請參閱下一節中的程序。

編輯 JSON 結構描述

您可以使用下列步驟編輯 JSON 結構描述。

若要編輯 JSON 結構描述

1. 在結構描述編輯器，選擇新增資料欄以新增資料欄。

   一個新的欄出現在第一欄位置。若要變更欄順序，請選擇欄名稱旁的向上和向下箭頭。

   針對每個新資料欄，請提供以下資訊：

   - 在資料欄名稱中，輸入名稱。

     資料欄名稱不能為空白、長度必須超過單一字元，且不得包含保留的 SQL 關鍵字。名稱也必須符合 SQL 一般識別符的命名條件：必須以字母開頭，且只包含字母、底線字元和數字。

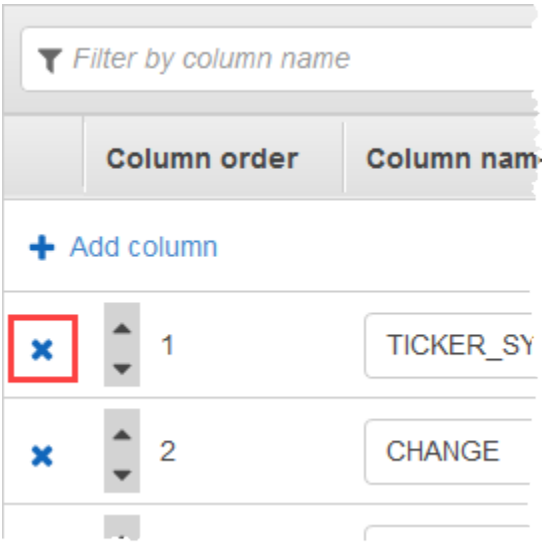
- 在資料欄類型中，輸入 SQL 資料類型。

  資料欄類型可以是任何支援的 SQL 資料類型。如果新資料類型為 CHAR、VARBINARY 或 VARCHAR，請指定長度的資料長度。如需詳細資訊，請參閱資料類型。

- 在資料列路徑，請提供資料列路徑。資料列路徑是映射至 JSON 元素的有效 JSONPath 運算式。

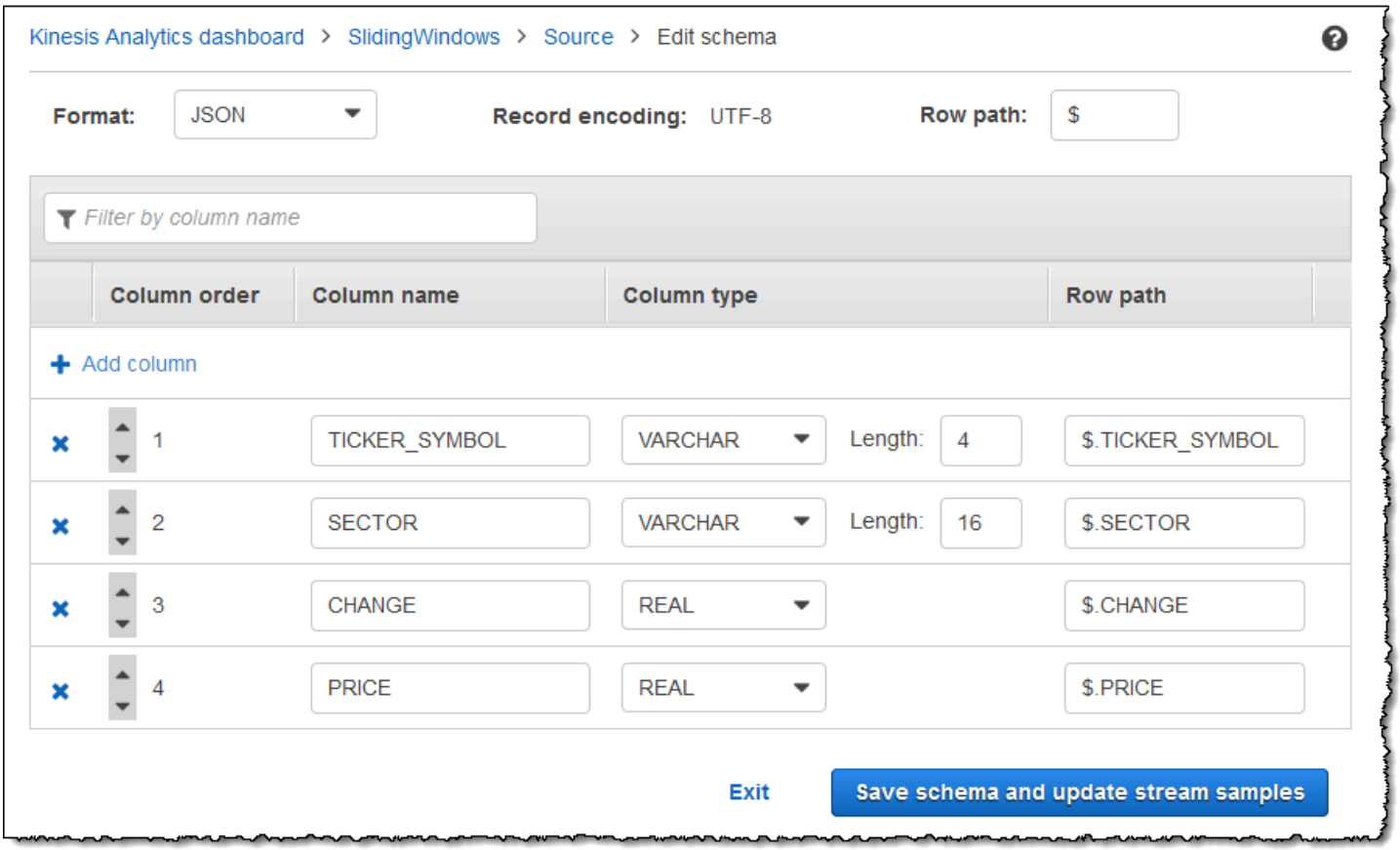
> **Note**
>
> 基本資料列路徑值，是包含要匯入之資料的最上層父系路徑。根據預設，此值為 $。如需詳細資訊，請參閱 `JSONMappingParameters` 中的 `RecordRowPath`。

2. 若要刪除欄，請選擇欄編號旁邊的 x 圖示。



3. 若要重新命名欄，請在欄名稱輸入新名稱。新資料欄名稱不能為空白、長度必須超過單一字元，且不得包含保留的 SQL 關鍵字。名稱也必須符合 SQL 一般識別符的命名條件：必須以字母開頭，且只包含字母、底線字元和數字。
4. 若要變更欄的資料類型，請為資歷欄類型選擇新的資料類型。如果新資料類型為 CHAR, VARBINARY or VARCHAR，請指定長度的資料長度。如需詳細資訊，請參閱資料類型。
5. 選擇儲存結構描述並更新串流，以儲存您的變更。

修改後的結構描述會顯示在編輯器中，類似如下。



如果您的結構描述有許多資料列，可以使用 依資料欄名稱篩選來篩選資料列。例如，若要編輯以 P 開頭的欄名稱 (例如 Price 欄)，請在依資料欄名稱篩選方塊中輸入 P。

編輯 CSV 結構描述

您可以使用下列步驟編輯 CSV 結構描述。

若要編輯 CSV 結構描述

1. 在結構描述編輯器中，針對資料列分隔符號，選擇傳入資料串流使用的分隔符號。這是串流中資料記錄 (例如換行字元) 之間的分隔符號。
2. 針對資料欄分隔符號，請選擇傳入資料串流使用的分隔符號。這是串流中資料欄位之間的分隔符號。
3. 若要新增資料欄，請選擇新增資料欄。

   一個新的欄出現在第一欄位置。若要變更欄順序，請選擇欄名稱旁的向上和向下箭頭。

   針對每個新資料欄，請提供以下資訊：

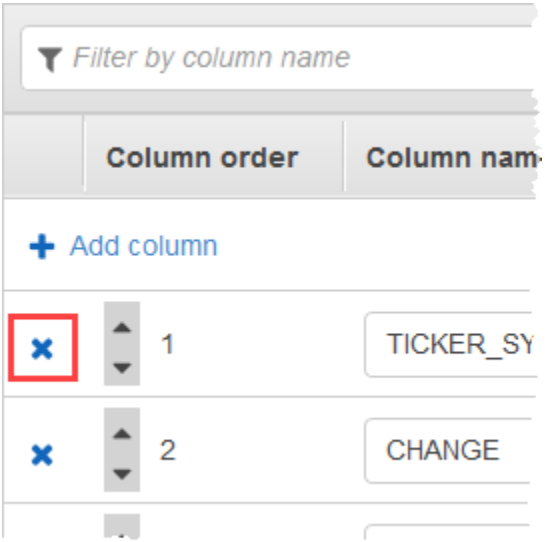
- 在資料欄名稱中輸入名稱。

  資料欄名稱不能為空白、長度必須超過單一字元，且不得包含保留的 SQL 關鍵字。名稱也必須符合 SQL 一般識別符的命名條件：必須以字母開頭，且只包含字母、底線字元和數字。

- 在資料欄類型中，輸入 SQL 資料類型。

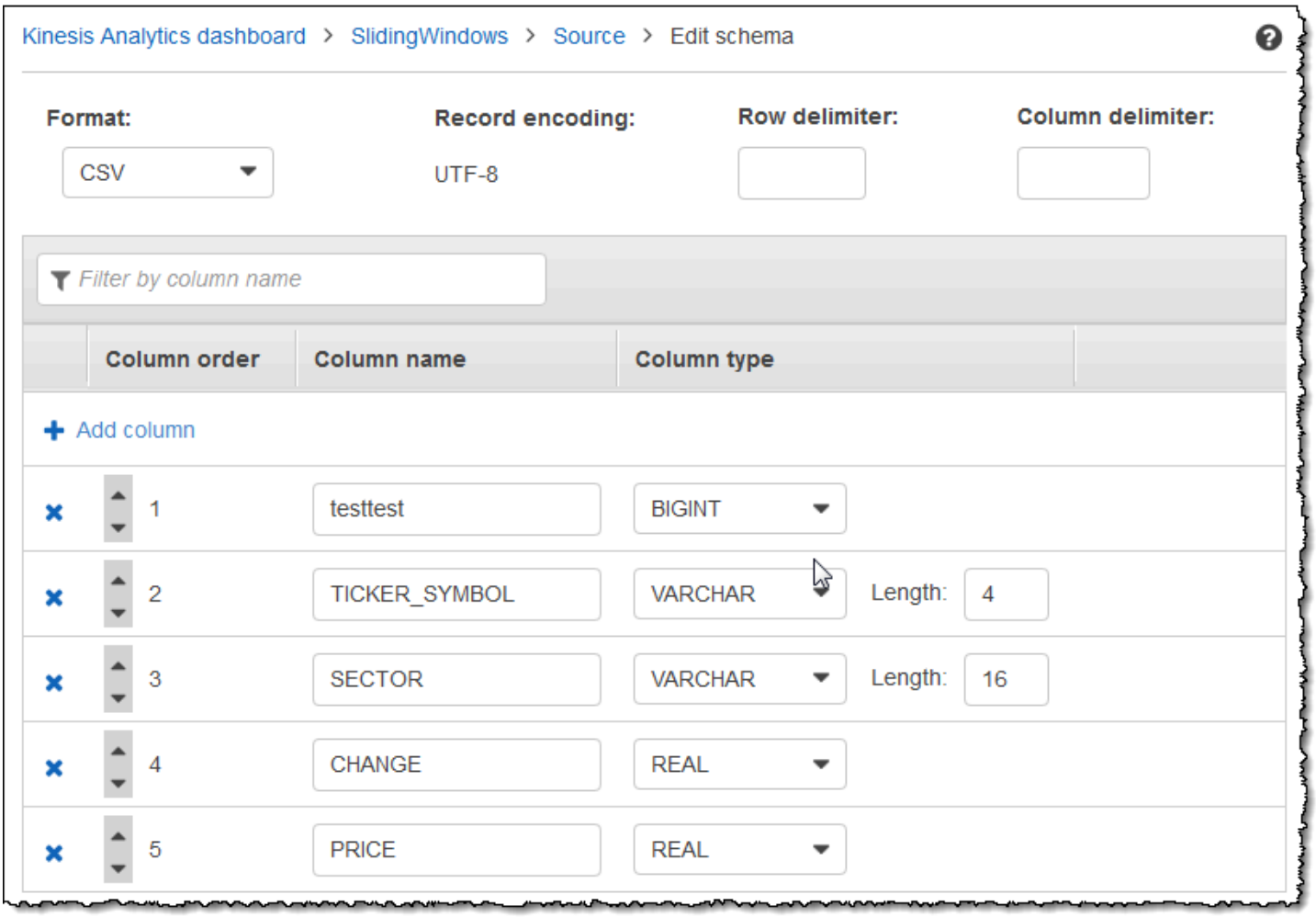
  資料欄類型可以是任何支援的 SQL 資料類型。如果新資料類型為 CHAR、VARBINARY 或 VARCHAR，請指定長度的資料長度。如需詳細資訊，請參閱資料類型。

4. 若要刪除欄，請選擇欄編號旁邊的 x 圖示。



5. 若要重新命名資料欄，請在資料欄名稱中輸入新名稱。新資料欄名稱不能為空白、長度必須超過單一字元，且不得包含保留的 SQL 關鍵字。名稱也必須符合 SQL 一般識別符的命名條件：必須以字母開頭，且只包含字母、底線字元和數字。
6. 若要變更欄的資料類型，請為資歷欄類型選擇新的資料類型。如果新資料類型為 CHAR、VARBINARY 或 VARCHAR，請指定長度的資料長度。如需詳細資訊，請參閱資料類型。
7. 選擇儲存結構描述並更新串流，以儲存您的變更。

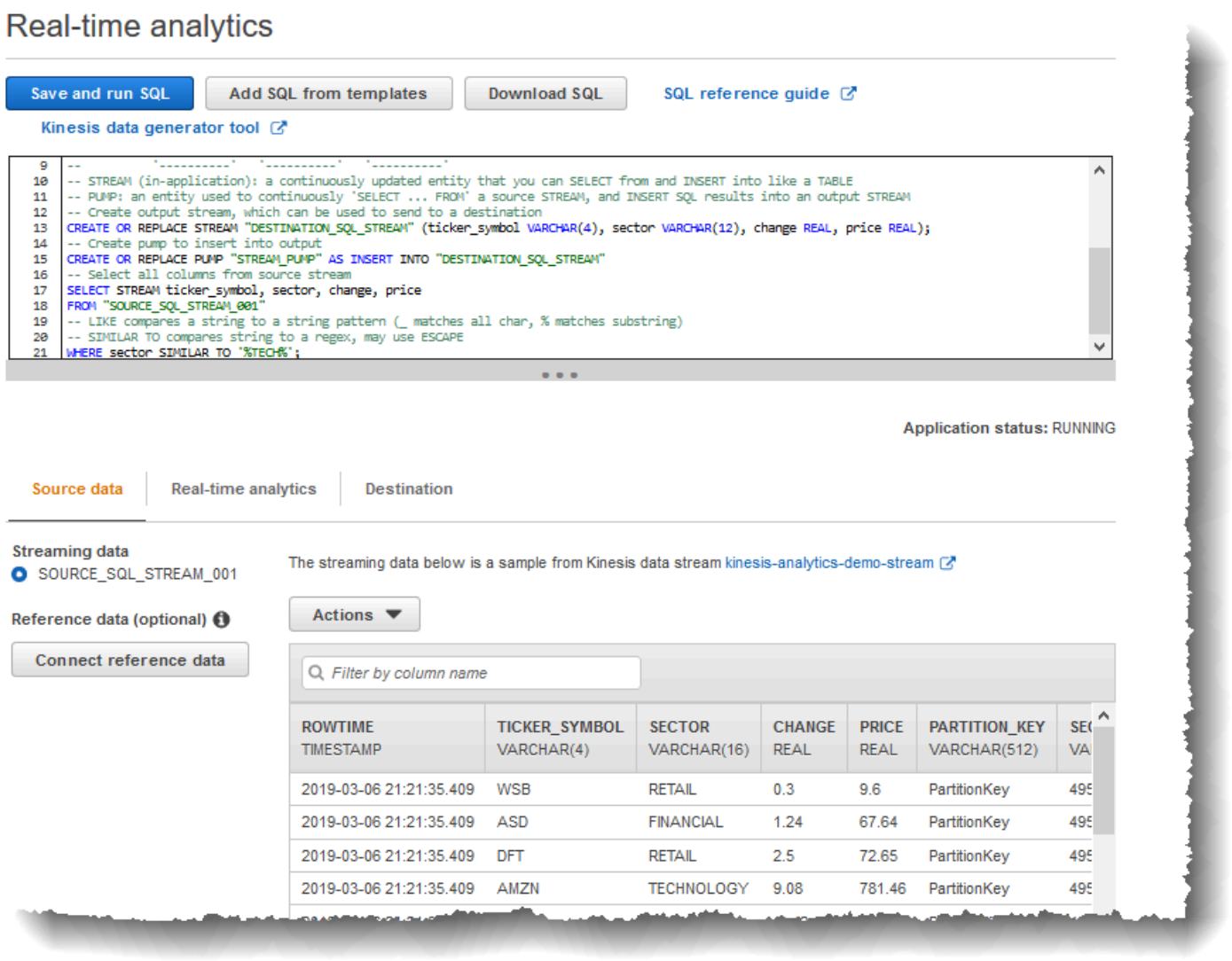
修改後的結構描述會顯示在編輯器中，類似如下。



如果您的結構描述有許多資料列，可以使用 依資料欄名稱篩選來篩選資料列。例如，若要編輯以 P 開頭的欄名稱 (例如 `Price` 欄)，請在依資料欄名稱篩選方塊中輸入 P。

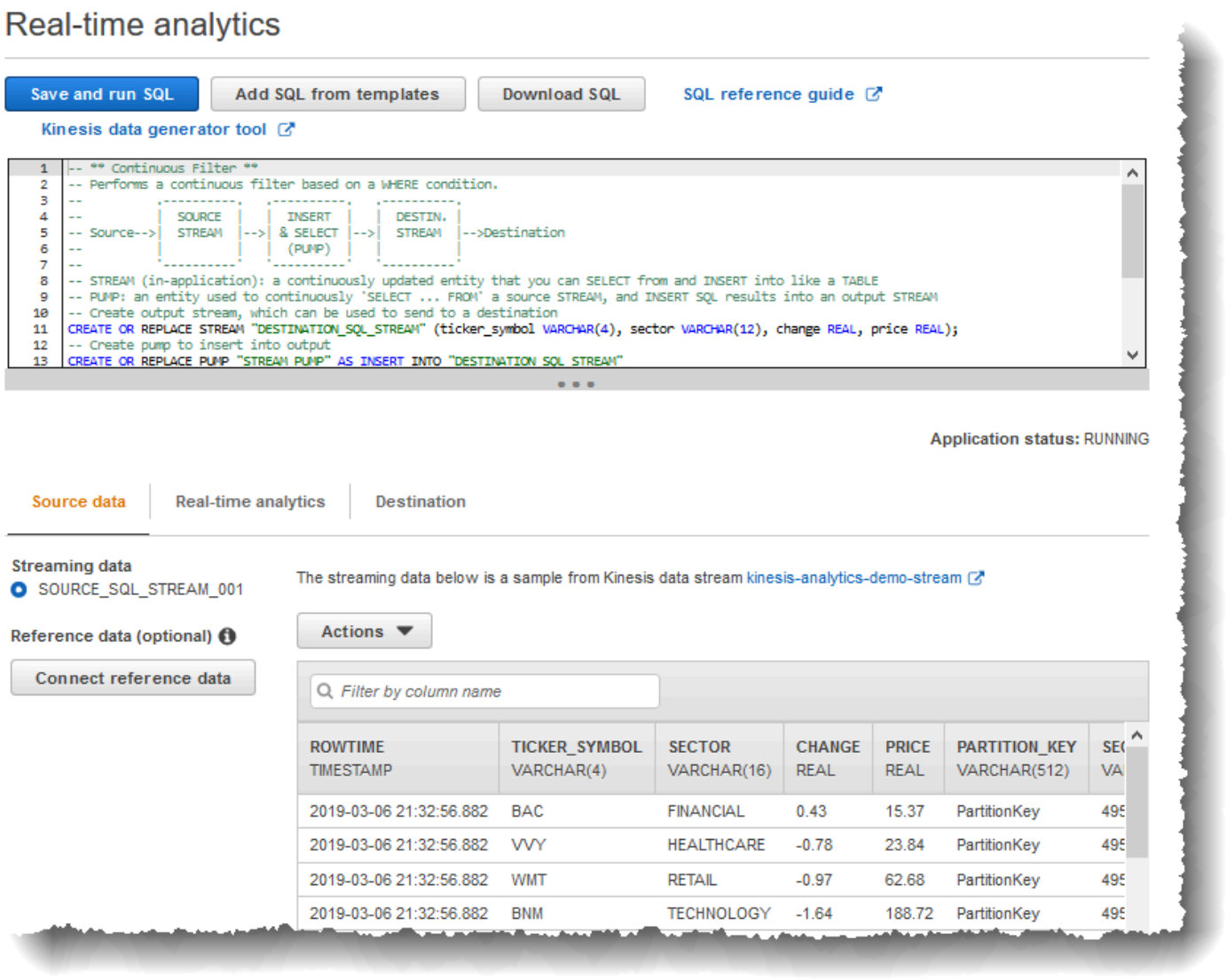
使用 SQL 編輯器

以下，您可以找到 SQL 編輯器區段以及各區段如何運作的相關資訊。在 SQL 編輯器中，您可以自行撰寫程式碼，或選擇從範本新增 SQL。SQL 範本提供範例 SQL 程式碼，可協助您撰寫常見的 Amazon Kinesis Data Analytics 應用程式。本指南中的範例應用程式使用其中一些範本。如需詳細資訊，請參閱 Kinesis Data Analytics for SQL 範例。



來源資料標籤

來源資料標籤可識別串流來源。它也會識別此來源映射的應用程式內輸入串流，並提供應用程式輸入組態。



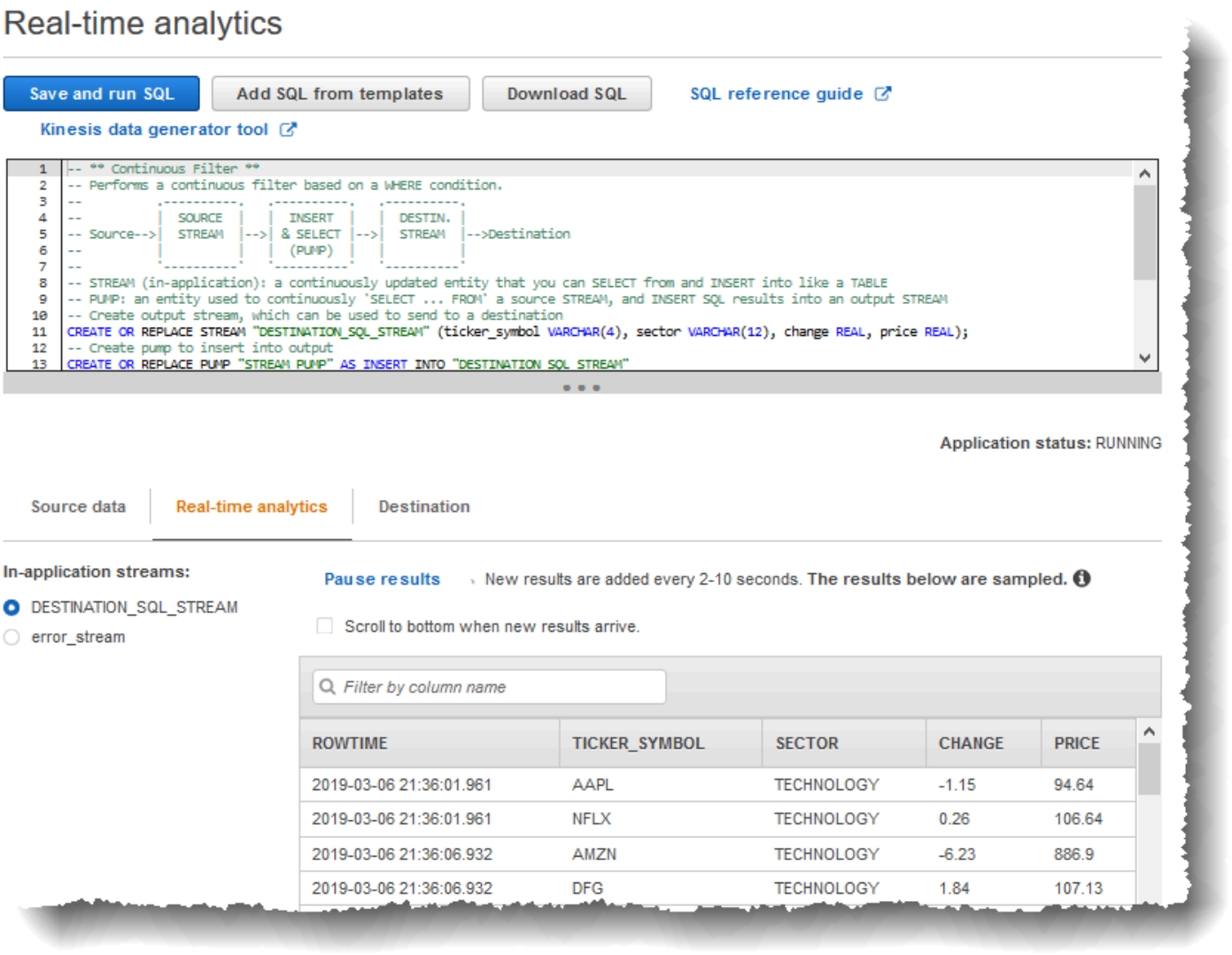
Amazon Kinesis Data Analytics 提供下列時間戳記欄，因此您不需要在輸入組態中提供明確的映射：

- ROWTIME：應用程式內串流中的每一列都有一個名為 ROWTIME 的特殊欄。此欄是 Kinesis 資料分析在第一個應用程式內串流中插入資料列時的時間戳記。
- Approximate_Arrival_Time：串流來源上的記錄包含 Approximate_Arrival_Timestamp 欄。當串流來源成功接收並儲存相關記錄時，此即為大約的到達時間戳記。Kinesis Data Analytics 會將此欄位擷取至應用程式內輸入串流作為 Approximate_Arrival_Time。Amazon Kinesis Data Analytics 僅會在對應至串流來源的應用程式內輸入串流中提供此欄。

這些時間戳記值在以時間為基礎的窗口化查詢中非常有用。如需詳細資訊，請參閱 窗口化查詢。

即時分析標籤

即時分析標籤會顯示應用程式碼建立的所有應用程式內串流。這組串流包括 Amazon Kinesis Data Analytics 提供給所有應用程式的錯誤串流 (`error_stream`)。



目的地標籤

目的地標籤可讓您設定應用程式輸出，以保留應用程式內串流至外部目的地。您可以設定輸出，將任何應用程式內串流的資料保存至外部目的地。如需更多詳細資訊，請參閱 設定應用程式輸出。

串流 SQL 概念

Amazon Kinesis Data Analytics 會使用延伸模組實作 ANSI 2008 SQL 標準。這些延伸模組可讓您處理串流資料。下列主題涵蓋重要的串流 SQL 概念。

主題

應用程式內串流與幫浦

在設定應用程式輸入時，您會將串流來源映射到建立的應用程式內串流。資料會持續從串流來源流入應用程式內串流。應用程式內串流的運作方式，與您可以使用 SQL 陳述式查詢的資料表類似，但這稱為串流，因為其代表連續的資料流程。

> **Note**
>
> 請勿將應用程式內串流與 Amazon Kinesis 資料串流和 Firehose 交付串流混淆。應用程式內串流僅存在 Amazon Kinesis Data Analytics 應用程式的環境中。Kinesis 資料串流和 Firehose 交付串流的存在與您的應用程式無關。您可以在應用程式輸入組態將它們設定為串流來源，或在輸出組態將其設定為目的地。

您也可以視需要建立更多應用程式內串流，以儲存中繼查詢結果。建立應用程式內串流分為兩個步驟。首先，建立應用程式內串流，然後將資料送入其中。例如，假設應用程式的輸入組態會建立名 INPUTSTREAM 為的應用程式內串流。在下列範例中，您會建立另一個串流 (TEMPSTREAM)，然後送進從 INPUTSTREAM 抽取的資料。

1. 建立具有三個資料欄的應用程式內串流 (TEMPSTREAM)，如下所示：

```
CREATE OR REPLACE STREAM "TEMPSTREAM" (
   "column1" BIGINT NOT NULL,
```

```
    "column2" INTEGER,
    "column3" VARCHAR(64));
```

欄名位於引號中，代表其區分大小寫。如需詳細資訊，請參閱《Amazon Kinesis Data Analytics SQL 參考》中的識別符。

2. 使用幫浦將數據插入流中。幫浦是執行中的連續插入查詢，可將資料從一個應用程式內串流插入另一個應用程式內串流。下列陳述式會建立幫浦 (SAMPLEPUMP)，並透過從另一個資料流 (INPUTSTREAM) 選取記錄，將資料插入 TEMPSTREAM 中。

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

您可以讓多個寫入器插入應用程式內串流，而且可以從串流中選取多個讀取器。將應用程式內串流視為實作發佈 / 訂閱訊息範例。在此範例中，資料列 (包括建立時間和接收時間) 可以透過串流 SQL 陳述式的串聯處理、解譯和轉送，而不必儲存在傳統的 RDBMS 中。

建立應用程式內串流之後，您可以執行一般 SQL 查詢。

> **Note**
>
> 當您查詢串流時，大多數 SQL 陳述式都會使用資料列或時間型窗口繫結。如需詳細資訊，請參閱 窗口化查詢。

您也可以加入串流。如需加入串流的範例，請參閱 串流資料作業：串流聯結。

時間戳記和 ROWTIME 欄

應用程式內串流包含一個名為 ROWTIME 的特殊欄。當 Amazon Kinesis Data Analytics 在第一個應用程式內串流中插入資料列時，它會儲存時間戳記。ROWTIME 反映 Amazon Kinesis Data Analytics 從串流來源讀取後，將記錄插入第一個應用程式內串流的時間戳記。然後此 ROWTIME 值會在整個應用程式中保留。

> **Note**
>
> 當您將記錄從一個應用程式內串流抽取到另一個應用程式內串流時，不需要明確複製該 ROWTIME 欄，Amazon Kinesis Data Analytics 會為您複製此欄。

Amazon Kinesis Data Analytics 保證這些 ROWTIME 值會單調增加。您可以在基於時間的窗口式查詢中使用此時間戳記。如需詳細資訊，請參閱 窗口化查詢。

您可以像在應用程式內串流中存取任何其他資料欄一樣，存取 SELECT 陳述式中的 ROWTIME 資料欄。例如：

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

了解串流分析中的不同時間

除了 ROWTIME 之外，實時串流應用程式中還有其他類型的時間。這些時間為：

- 事件時間：事件發生的時間戳記。這有時也稱為用戶端時間。在分析中通常偏好使用此時間，因其為事件發生的時間。不過，許多事件來源 (例如行動電話和 Web 用戶端) 沒有可靠的時鐘，這可能會導致不正確的時間。此外，連線問題可能會導致串流上的記錄顯示順序與事件發生的順序不相同。

- 擷取時間：記錄新增至串流來源的時間戳記。Amazon Kinesis Data Streams 在提供此時間戳記的每個記錄中，都包含一個名為 APPROXIMATE_ARRIVAL_TIME 的欄位。這有時也稱為伺服器端時間。這個擷取時間通常是接近事件時間的近似值。如果記錄擷取到串流時有任何延遲，則可能會導致錯誤，但通常很少見。此外，擷取時間很少出現故障，但由於串流資料的分散式性質，故障可能會發生。因此，擷取時間大多準確且符合順序地反映事件時間。

- 處理時間：Amazon Kinesis Data Analytics 在第一個應用程式內串流中插入資料列的時間戳記。在每個應用程式內串流中，Amazon Kinesis Data Analytics 會在 ROWTIME 欄提供時間戳記。處理時間總是會單調增加。但是，如果您的應用程式落後，此時間就不准確。(如果應用程式落後，處理時間就不能準確反映事件時間。) 此 ROWTIME 與掛鐘比對是準確的，但它可能不是事件實際發生的時間。

在基於時間的窗口查詢中，使用這些時間中的每一個時間都有優點和缺點。我們建議您選擇其中一個或多個時間，並擬好根據使用案例情境處理相關缺點的策略。

> **Note**
>
> 如果您使用以列為基礎的窗口，則時間不是問題，您可以忽略此節。

我們建議採用雙窗口策略，該策略使用兩個時間基礎，即 ROWTIME 與另一個其他時間 (擷取或事件時間) 兩者。

- 使用 ROWTIME 作為第一個窗口，此時段可控制查詢發出結果的頻率，如下列範例所示。這並非邏輯時間。
- 把其中一個其他時間當作邏輯時間，即您想要連結到分析的時間 此時間表示事件發生的時間。在下面的例子中，分析目標是按股票代號對記錄進行分組和返回計數。

此策略的優點，是可以使用代表事件發生的時間。當您的應用程式落後或事件出現故障時，它可以適當地處理。如果應用程式在將記錄引入應用程式內串流時落後，它們仍會依照第二個窗口中的邏輯時間進行分組。該查詢用 ROWTIME 來保證處理的順序。任何延遲的記錄 (與 ROWTIME 值相比，擷取時間戳記顯示較早的值) 也會成功處理。

請考慮下列針對入門練習中使用的示範串流的查詢。該查詢使用 GROUP BY 子句，並在一分鐘的輪轉窗口中發出股票代號計數。

```
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，您先依據 ROWTIME 將記錄分組在一分鐘的窗口中，然後再依據 APPROXIMATE_ARRIVAL_TIME。

結果中的時間戳記值會無條件捨去至最接近的 60 秒間隔。查詢發出的第一個群組結果，會顯示第一分鐘的記錄。發出的第二組結果，會根據 ROWTIME 顯示接下來幾分鐘內的記錄。最後一筆記錄指出應用程式導入紀錄到應用程式內串流的時間延遲 (與擷取時間戳記相比，顯示了延遲 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.
```

透過將結果推送至下游資料庫，您可以合併結果，以獲得最終精確的每分鐘計數。例如，您可以將應用程式輸出設定為將結果保留到可寫入 Amazon Redshift 資料表的 Firehose 交付串流。在 Amazon Redshift 資料表中顯示結果之後，您可以查詢資料表來計算依據 Ticker_Symbol 分組的總計數。在 XYZ 的情況，即使記錄遲到，總數也是準確的（6+1）。

持續查詢

串流上的查詢會透過串流資料持續執行。這種持續執行可啟用案例，例如應用程式能夠持續查詢串流並產生提醒。

在入門練習中，您有一個名為 SOURCE_SQL_STREAM_001 的應用程式內串流。它會持續從示範串流 (Kinesis 資料串流) 接收股票價格。結構描述如下：

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

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

假設您對超過 15% 的股票價格變動感興趣。您可以在應用程式碼中使用下列查詢。此查詢會持續執行，並在偵測到股票價格變動大於 15% 時發出記錄。

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

使用下列程序來設定 Amazon Kinesis Data Analytics 應用程式，並測試此查詢。

若要測試查詢

1. 按照入門練習建立應用程式。
2. 用上述 SELECT 查詢取代應用程式碼中的 SELECT 陳述式。產生的應用程式碼如下所示：

```
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;
```

窗口化查詢

應用程式碼中的 SQL 查詢會透過應用程式內串流持續執行。應用程式內串流，代表的是持續在應用程式中流動的無限制資料。因此，若要取得此持續更新輸入的結果集，經常會使用定義的窗口時段或資料列來限制查詢。這些也被稱為窗口式 SQL。

對於以時間為基礎的窗口化查詢，您可以根據時間來指定窗口大小 (例如，一分鐘的窗口)。這需要應用程式內串流中的時間戳記資料欄，該欄會單調增加。(新資料列的時間戳記大於或等於上一列。) 在每個應用程式內串流中，Amazon Kinesis Data Analytics 會提供這樣的時間戳記欄，名為 ROWTIME。您可以在指定基於時間的查詢中使用此欄。針對應用程式，您可以選擇其他時間戳記選項。如需詳細資訊，請參閱 時間戳記和 ROWTIME 欄。

針對以資料列為基礎的窗口化查詢，您可以根據資料列數來指定窗口大小。

您可以根據應用程式需求，指定查詢以輪轉窗口、滑動窗口或交錯窗口方式處理記錄。Kinesis Data Analytics 支援下列窗口類型：

- 交錯窗口：此查詢使用金鑰式時間窗口彙總資料，該窗口會在資料到達時打開。這些金鑰允許多個重疊的窗口。這是使用基於時間的窗口聚合數據的建議方式，因為 Stagger Windows 與「翻滾窗口」相比可以減少遲到或 out-of-order 數據。
- 輪轉窗口：此查詢使用定期開啟和關閉的時間窗口來彙總資料。
- 滑動視窗：此查詢使用固定時間或資料列計數間隔持續彙總資料。

交錯窗口

使用交錯窗口是一種窗口化方法，適用於分析不一致時間到達的資料群組。此方式非常適合任何時間序列分析使用案例，例如一組相關的銷售或日誌記錄。

例如，VPC 流程日誌的擷取窗口約為 10 分鐘。但如果你在客戶端上彙總資料，則可以有一個長達 15 分鐘的擷取窗口，。交錯視窗是彙總這些日誌以進行分析的理想選擇。

交錯窗口可解決相關記錄未落入相同時間限制窗口的問題，例如使用輪轉窗口時。

有輪轉窗口的部分結果

彙總遲到或順序錯誤的資料時，使用 輪轉窗口 有某些限制。

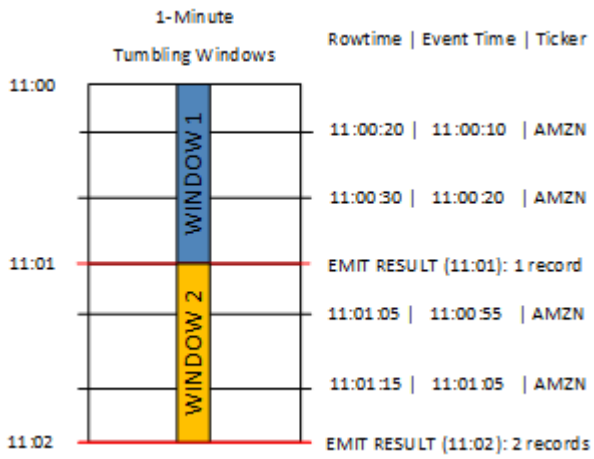
如果使用輪轉窗口來分析與時間相關的資料群組，則個別記錄可能會落入不同的窗口中。因此，每個窗口的部分結果必須在稍後合併，以產生每組記錄的完整結果。

在下面的輪轉窗口查詢中，記錄按列時間，事件時間和股票代號分組到窗口中：

```
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);
```

在下圖中，應用程式根據交易發生的時間（事件時間），以一分鐘的精細程度計算接收的交易數量。該應用程式可以使用輪轉窗口，根據列時間和事件時間幫資料分組。該應用程式收到四條記錄，這些記錄彼此都在一分鐘內到達。它按列時間，事件時間和股票符號幫記錄分組。因為有些記錄會在第一個輪轉窗口結束後到達，所以記錄不會全部落在相同的一分鐘輪轉窗口內。



前面的圖表具有以下事件。

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

來自輪轉窗口應用程式的結果集類似如下。

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

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

在之前的結果集中，傳回了三個結果：

- 記錄的 ROWTIME 為 11:01:00，彙總了前兩個記錄。
- 11：02：00 的記錄僅彙總了第三條記錄。此記錄在第二個窗口中有一個 ROWTIME，但在第一個窗口中有一個 EVENT_TIME。
- 11：02：00 的記錄僅彙總了第四條記錄。

若要分析完整的結果集，必須在持續性存放區彙總記錄。這會增加應用程式的複雜性和處理需求。

交錯窗口的完整結果

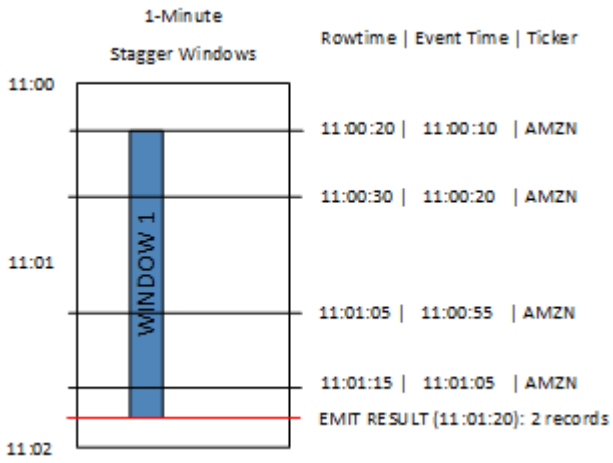
為了提高分析時間相關資料記錄的準確性，Kinesis Data Analytics 提供了一種稱為交錯窗口的新窗口類型。在此窗口類型中，窗口會在第一個與分割區索引鍵相符的事件到達時打開，而不是在固定的時間間隔。窗口會依指定的存留期關閉，測量由窗口開啟時起算。

在窗口子句中，交錯窗口是每個索引鍵群組的個別時間限制窗口。應用程式會在自己的時間窗口內彙總窗口子句的每個結果，而不是針對所有結果使用單一窗口。

在下面的交錯窗口查詢中，記錄按事件時間和股票代號分組到窗口中：

```
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);
```

在下圖中，事件按事件時間和股票代號彙總到交錯窗口中。



上圖有下列事件，這些事件輪轉窗口應用程式分析的相同：

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

來自交錯窗口應用程式的結果集類似如下。

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

傳回的記錄聚合前三個輸入記錄。記錄會依一分鐘的交錯窗口來分組。當應用程式收到第一筆 AMZN 記錄 (其中 ROWTIME 為 11:00:20) 時，就會啟動交錯窗口。當 1 分鐘交錯窗口到期時 (11:01:20)，會

將包含落在交錯窗口內 (以 ROWTIME 和 EVENT_TIME 為基礎) 的結果的記錄寫入輸出串流。如使用交錯窗口，所有在一分鐘視窗內有 ROWTIME 和 EVENT_TIME 的記錄都會在單一結果中發出。

最後一筆記錄 (在一分鐘彙總外有 EVENT_TIME) 會分別彙總。這是因為 EVENT_TIME 是用來將記錄分隔成結果集的分割區索引鍵之一，而第一個視窗的 EVENT_TIME 分割區索引鍵是 11:00。

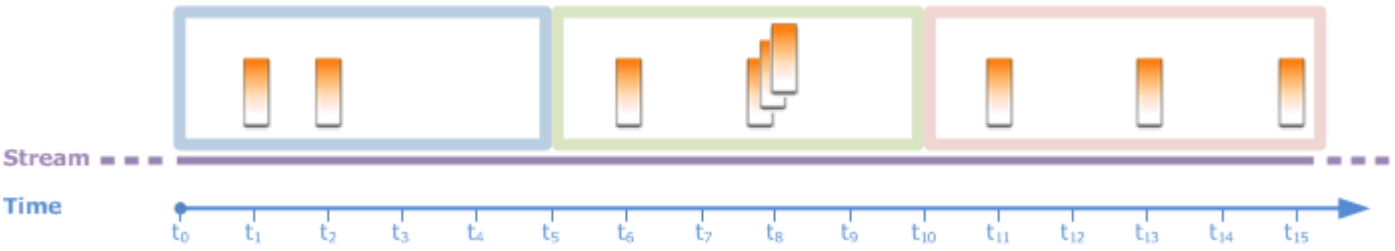
交錯窗口的語法是在特殊子句 WINDOWED BY 中定義的。使用此子句，而不是用於串流聚合的 GROUP BY 子句。子句會緊接顯示在選用 WHERE 子句之後，在 HAVING 子句之前。

交錯窗口在 WINDOWED BY 子句中定義，並採用兩個參數：分割區索引鍵和窗口長度。分割區索引鍵會分割傳入的資料串流，並定義窗口開啟的時間。當串流上出現具有唯一分割區索引鍵的第一個事件時，會開啟交錯窗口。交錯窗口會在窗口長度所定義的固定期間之後關閉。語法如下列程式碼範例所示：

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

輪轉窗口（使用 GROUP BY 彙總）

當窗口查詢以非重疊的方式處理每個窗口時，即稱作輪轉窗口。在此情況下，應用程式內串流上的每個記錄都屬於一個特定窗口。紀錄只會被處理一次（當查詢處理記錄所屬的窗口時）。



例如，使用 GROUP BY 子句的彙總查詢會處理輪轉窗口中的資料列。入門練習的示範串流，會接收您的應用程式內串流 SOURCE_SQL_STREAM_001 之股票價格資料。這個串流具有以下結構描述：

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

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

在應用程式碼中，假設您想要在一分鐘的窗口中尋找每個股票代號的彙總 (最低、最大值) 價格。您可以使用下列查詢：

```
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);
```

前面是以時間為基礎的窗口化查詢範例。此查詢按 ROWTIME 值為記錄分組。針對每分鐘進行報告，STEP 函數會將 ROWTIME 值無條件捨去至最接近的分鐘。

> **Note**
>
> 您也可以使用 FLOOR 函數來將記錄分組至窗口。但是，FLOOR 只能將時間值捨去到完整時間單位 (小時、分鐘、秒等)。建議使用 STEP 將記錄分組到輪轉窗口中，因為它可以將值捨去到任意間隔，例如 30 秒。

此查詢是非重疊 (輪轉) 窗口的例子。GROUP BY 子句會將記錄分組在一分鐘的窗口中，而且每個記錄都屬於特定的窗口 (不重疊)。該查詢每分鐘發出一個輸出記錄，提供在特定分鐘記錄的最小/最大股票價格。如要從輸入資料串流生成定期報告，這種類型的查詢就非常有用。在此範例中，每分鐘產生一次報告。

若要測試查詢

1. 按照入門練習設置應用程式。
2. 用上述 SELECT 查詢取代應用程式碼中的 SELECT 陳述式。產生的應用程式碼如下所示：

```
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
```

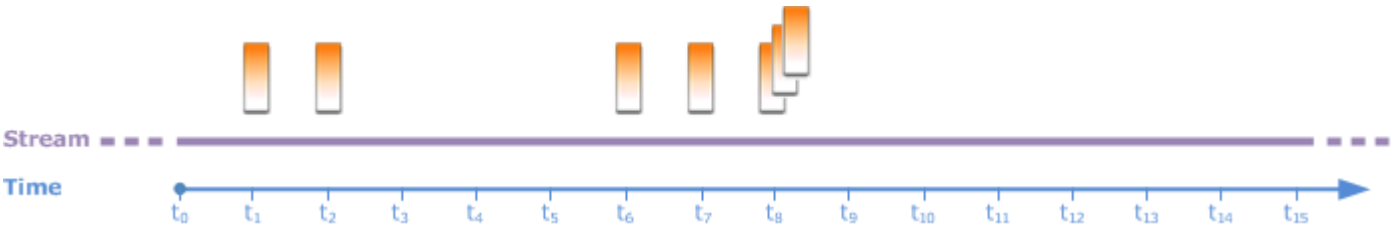
```
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM Ticker_Symbol,
                  MIN(Price) AS Min_Price,
                  MAX(Price) AS Max_Price
    FROM    "SOURCE_SQL_STREAM_001"
    GROUP BY Ticker_Symbol,
             STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

滑動視窗

您可以定義以時間或資料列為基礎的視窗，而不需使用 GROUP BY 將記錄分組。您可以透過添加明確的 WINDOW 子句來完成此操作。

在這種情況下，當窗口隨著時間的推移而滑動時，Amazon Kinesis Data Analytics 會在串流上出現新記錄時發出輸出。Kinesis Data Analytics 會透過處理窗口中的資料列來發出此輸出。窗口可以在這種類型的處理中重疊，且記錄可以是多個窗口的一部分，並用每個窗口進行處理。以下範例解釋了滑動窗口。

考慮用一個簡單的查詢來計算串流上的記錄。此範例假設了 5 秒的窗口。在下面的示範串流中，新的記錄到達時間為 t_1、t_2、t_6 和 t_7，三個記錄到達時間為 t_8 秒。



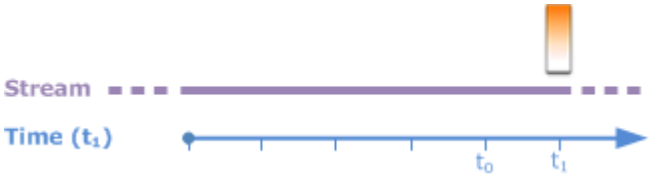
請謹記以下幾點：

- 此範例假設了 5 秒的窗口。5 秒的窗口會隨著時間連續滑動。
- 針對進入窗口的每一列，滑動窗口會發出一個輸出列。應用程式啟動後不久，即使 5 秒的窗口尚未通過，您也會看到查詢發出串流上每筆新記錄的輸出。例如，當記錄出現在第一秒和第二秒的查詢時，串流會發出輸出。稍後，查詢會在 5 秒的窗口中處理記錄。
- 窗口隨著時間滑動。如果串流上的舊記錄落在窗口外，則查詢不會發出輸出，除非串流上也有新記錄落在該 5 秒視窗內。

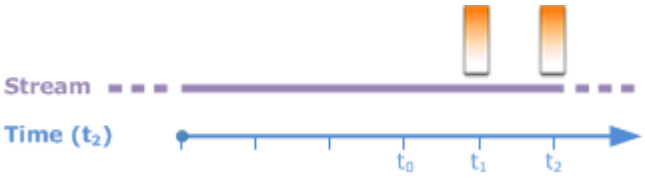
假設查詢在 t_0 開始執行。然後會發生以下情況：

1. t_0 時，查詢開始。查詢不會發出輸出（計數值），因為目前沒有記錄。

2. 在 t_1，一個新的記錄出現在串流上，查詢發出計數值 1。



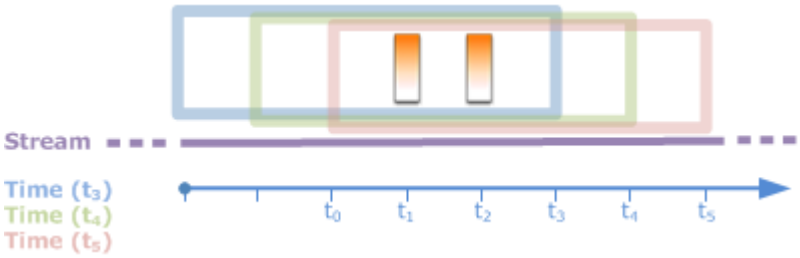
3. 在 t_2，另一個記錄出現，查詢發出計數值 2。



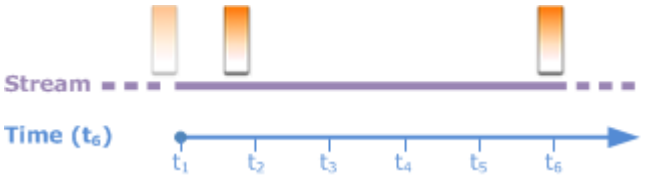
4. 5 秒的窗口會隨著時間滑動：

   - 在 t_3，滑動窗口 t_3 到 t_0
   - 在 t4 (滑動窗口 t_4 到 t_0)
   - 在 t_5 滑動窗口 t_5-t_0

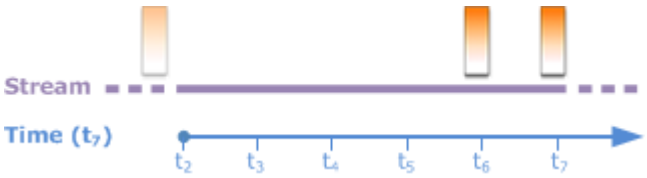
   在所有這些時刻，5 秒的窗口具有相同的記錄-沒有新記錄。因此，查詢沒有發出任何輸出。



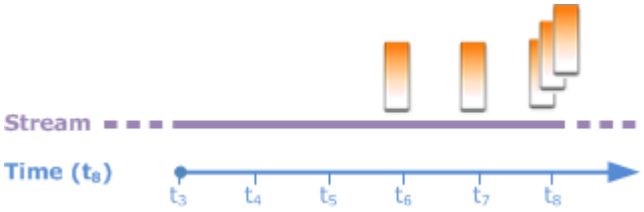
5. 在時間 t_6 時，5 秒的窗口為 (t_6 到 t_1)。查詢在 t_6 偵測到一個新的記錄，因此它發出輸出 2。t_1 處的記錄不再位於窗口中，並且不會計算在內。



6. 在時間 t_7 時，5 秒的窗口為 t_7 到 t_2。查詢在 t_7 偵測到一個新的記錄，因此它發出輸出 2。t_2 處的記錄不再位於 5 秒窗口中，因此不會計算在內。



7. 在時間 t_8 時，5 秒的窗口為 t_8 到 t_3。查詢偵測到三個新記錄，因此會發出記錄計數 5。



總之，窗口是一個固定的大小，並隨著時間的推移滑動。新的記錄出現時，查詢會發出輸出。

> **Note**
>
> 建議您使用不超過一個小時的滑動窗口。如果您使用較長的窗口，在定期系統維護後，應用程式需要更長的時間重新啟動。這是因為必須再次從串流中讀取來源資料。

下列範例查詢使用 WINDOW 子句來定義窗口和執行彙總。因為查詢未指定 GROUP BY，所以查詢會使用滑動窗口方法來處理串流上的記錄。

範例 1：使用 1 分鐘滑動窗口處理串流

請考慮在入門練習中填入應用程式內串流的示範串流 SOURCE_SQL_STREAM_001。以下為其結構描述。

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

假設您希望應用程式使用滑動 1 分鐘窗口來計算彙總。也就是說，對於出現在串流上的每個新記錄，您希望應用程式套用彙總到前 1 分鐘窗口的記錄，以發出輸出。

您可以使用下列時間窗口查詢。查詢會使用 WINDOW 子句來定義 1 分鐘的範圍間隔。在 WINDOW 子句中，PARTITION BY會按滑動窗口內的股票代號值對記錄進行分組。

```
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);
```

若要測試查詢

1. 按照入門練習設置應用程式。
2. 用上述 SELECT 查詢取代應用程式碼中的 SELECT 陳述式。產生的應用程式碼如下所示。

```
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);
```

範例 2：查詢在滑動視窗上套用彙總

示範串流上的下列查詢，會傳回 10 秒窗口中每個股票代號價格變動百分比的平均值。

```
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);
```

若要測試查詢

1. 按照入門練習設置應用程式。
2. 用上述 SELECT 查詢取代應用程式碼中的 SELECT 陳述式。產生的應用程式碼如下所示。

```
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);
```

範例 3：查詢相同串流上多個滑動視窗的資料

您可以撰寫查詢以發出輸出，其中每個資料欄值，是使用同一個資料流上定義的不同滑動窗口來計算。

在下面的例子中，查詢發出輸出股票代號，價格，a2 和 a10。它發出輸出給為兩列移動平均線穿過十列移動平均線的股票代碼。a2 和 a10 欄值衍生自兩列與十列滑動窗口。

```
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);
```

若要針對示範串流測試此查詢，請遵循 範例 1 中所述的測試程序。

串流資料作業：串流聯結

您可以在應用程式中擁有多個應用程式內串流。您可以撰寫 JOIN 查詢來關聯到達這些串流的資料。舉例來說，假設您有以下應用程式內串流。

- OrderStream— 接收正在下達的股票訂單。

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

- TradeStream— 接收這些訂單所產生的股票交易。

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

以下是將資料關聯到這些串流的 JOIN 查詢範例。

範例 1：報告下訂單後一分鐘內有交易的訂單

在此範例中，您的查詢會同時連接 OrderStream 和 TradeStream。但是，由於我們只想要訂單後一分鐘下達的交易，因此查詢定義了 1 分鐘的 TradeStream 窗口。如需有關窗口查詢的資訊，請參閱滑動視窗。

```
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;
```

您可以使用 WINDOW 子句明確定義窗口並編寫前面的查詢，如下所示：

```
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)
```

當您在應用程式碼中包含此查詢時，應用程式碼會持續執行。針對 OrderStream 上的每個到達記錄，如果在下訂單後的 1 分鐘窗口內有交易，則應用程序將發出輸出。

在前面的查詢中的連接是一個內部聯接，其中查詢在 OrderStream 發出記錄，TradeStream 中也有一個相符的記錄（反之亦然）。使用外部連接，您可以創建另一個有趣的情景。假設您想要下單後一分鐘內沒有交易的股票訂單，以及同一窗口其他訂單的交易。此即外部聯結的案例。

```
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;
```

遷移至 Managed Service for Apache Flink Studio 範例

下列範例示範如何將 Kinesis Data Analytics for SQL 移轉至 Managed Service for Apache Flink Studio。

在 Managed Service for Apache Flink Studio 中複寫 Kinesis Data Analytics for SQL 查詢

> **Warning**
>
> 針對新專案，我們建議您優先使用 Managed Service for Apache Flink Studio，而非 Kinesis Data Analytics for SQL 應用程式。Managed Service for Apache Flink Studio 易於使用且具備進階分析功能，讓您在幾分鐘內建置複雜的串流處理應用程式。

若要將您的工作負載遷移至 Managed Service for Apache Flink Studio 或 Managed Service for Apache Flink，本節提供可用於常見使用案例的查詢翻譯。

> **Note**
>
> Managed Service for Apache Flink 和 Managed Service for Apache Flink Studio 提供了 SQL 式 Kinesis Data Analytics 應用程式沒有的進階資料串流處理功能。其中包括僅一次處理語意、事件時間視窗、使用者定義函數和自訂整合的擴充性、命令式語言支援、持久的應用程式狀態、水平擴展、支援多個資料來源、可延伸整合等等。這些對於確保資料串流處理的準確性，完整性，一致性和可靠性至關重要。

在探索這些範例之前，我們建議您先檢閱將 Studio 筆記本用於 Managed Service for Apache Flink。

主題

- 在 Managed Service for Apache Flink Studio 中重建 Kinesis Data Analytics for SQL 查詢

在 Managed Service for Apache Flink Studio 中重建 Kinesis Data Analytics for SQL 查詢

下表提供一般 SQL 式 Kinesis Data Analytics 應用程式至 Managed Service for Apache Flink Studio 的查詢翻譯。

多步驟應用程式

SQL-based Kinesis Data Analytics application

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

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

Managed Service for Apache Flink Studio

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

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

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

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

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

DROP TABLE IF EXISTS IN_APP_STREAM_02;

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

DROP TABLE IF EXISTS DESTINATION_SQL_STREAM;

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


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

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


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


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

轉換 DateTime 值

SQL-based Kinesis Data Analytics application

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

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

Managed Service for Apache Flink Studio

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

PARTITIONED BY (TICKER) WITH (
   'connector' = 'kinesis', 'stream' = 'kinesis-analytics-demo-stream',
```

```
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601')

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

簡單提醒

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
```

Managed Service for Apache Flink Studio

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

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

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

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;
```

Managed Service for Apache Flink Studio

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

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

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

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

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

從查詢彙總部分結果

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
      )
;
```

Managed Service for Apache Flink Studio

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

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

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

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

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

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

```
         )
;

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

轉換字串值

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";
```

Managed Service for Apache Flink Studio

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

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

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

使用 Regex 替換子字串

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";
```

Managed Service for Apache Flink Studio

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

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

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

Regex 日誌剖析

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;
```

Managed Service for Apache Flink Studio

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

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

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

轉換 DateTime 值

SQL-based Kinesis Data Analytics application

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

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

Managed Service for Apache Flink Studio

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

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

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

視窗與彙總

SQL-based Kinesis Data Analytics application

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

Managed Service for Apache Flink Studio

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

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

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

使用列時間的輪轉視窗

SQL-based Kinesis Data Analytics application

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

Managed Service for Apache Flink Studio

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

Query 2 - % flink.ssql(type =
   update
)
      SELECT
         ticker,
         min(price) AS MIN_PRICE,
         max(price) AS MAX_PRICE
      FROM
         DESTINATION_SQL_STREAM
      GROUP BY
         TUMBLE(processing_time, INTERVAL '60' second),
         ticker;
```

擷取最常出現的值 (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
   )
;
```

Managed Service for Apache Flink Studio

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

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

Query 2 - % flink.ssql(type =
update
)
   SELECT
      *
   FROM
      (
         SELECT
            TICKER,
            COUNT(*) as MOST_FREQUENT_VALUES,
            ROW_NUMBER() OVER (PARTITION BY TICKER
         ORDER BY
            TICKER DESC) AS row_num
         FROM
            DESTINATION_SQL_STREAM
         GROUP BY
            TUMBLE(EVENT_TIME, INTERVAL '1' MINUTE),
            TICKER
      )
   WHERE
      row_num <= 5;
```

大約前 K 個項目

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));
```

Managed Service for Apache Flink Studio

```
%flinkssql
DROP TABLE IF EXISTS SOURCE_SQL_STREAM_001
CREATE TABLE SOURCE_SQL_STREAM_001 ( TS TIMESTAMP(3), WATERMARK FOR TS as TS -
 INTERVAL '5' SECOND, ITEM VARCHAR(1024),
PRICE DOUBLE)
   WITH ( 'connector' = 'kinesis', 'stream' = 'SOURCE_SQL_STREAM_001',
'aws.region' = 'us-east-1', 'scan.stream.initpos' = 'LATEST', 'format' = 'json',
'json.timestamp-format.standard' = 'ISO-8601');


%flink.ssql(type=update)
SELECT
   *
FROM
   (
      SELECT
         *,
         ROW_NUMBER() OVER (PARTITION BY AGG_WINDOW
      ORDER BY
         ITEM_COUNT DESC) as rownum
      FROM
         (
            select
               AGG_WINDOW,
               ITEM,
               ITEM_COUNT
            from
               (
                  select
                     TUMBLE_ROWTIME(TS, INTERVAL '60' SECONDS) as AGG_WINDOW,
                     ITEM,
                     count(*) as ITEM_COUNT
                  FROM
                     SOURCE_SQL_STREAM_001
                  GROUP BY
```

```
                    TUMBLE(TS, INTERVAL '60' SECONDS),
                    ITEM
                )
        )
    )
where
    rownum <= 3
```

剖析網頁日誌 (W3C_LOG_PARSE 函數)

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" ( column1 VARCHAR(16),
    column2 VARCHAR(16),
    column3 VARCHAR(16),
    column4 VARCHAR(16),
    column5 VARCHAR(16),
    column6 VARCHAR(16),
    column7 VARCHAR(16));
CREATE
OR REPLACE PUMP "myPUMP" ASINSERT INTO "DESTINATION_SQL_STREAM"
SELECT
    STREAM l.r.COLUMN1,
    l.r.COLUMN2,
    l.r.COLUMN3,
    l.r.COLUMN4,
    l.r.COLUMN5,
    l.r.COLUMN6,
    l.r.COLUMN7
FROM
    (
        SELECT
            STREAM W3C_LOG_PARSE("log", 'COMMON')
        FROM
            "SOURCE_SQL_STREAM_001"
    )
    AS l(r);
```

Managed Service for Apache Flink Studio

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

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

% flink.ssql(type=update)
   select
      SPLIT_INDEX(log, ' ', 0),
      SPLIT_INDEX(log, ' ', 1),
      SPLIT_INDEX(log, ' ', 2),
      SPLIT_INDEX(log, ' ', 3),
      SPLIT_INDEX(log, ' ', 4),
      SPLIT_INDEX(log, ' ', 5),
      SPLIT_INDEX(log, ' ', 6)
   from
      SOURCE_SQL_STREAM_001;
```

將字串拆分為多個欄位（VARIABLE_COLUMN_LOG_PARSE 函數）

SQL-based Kinesis Data Analytics application

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM"( "column_A" VARCHAR(16),
   "column_B" VARCHAR(16),
   "column_C" VARCHAR(16),
   "COL_1" VARCHAR(16),
   "COL_2" VARCHAR(16),
   "COL_3" VARCHAR(16));
CREATE
OR REPLACE PUMP "SECOND_STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT
   STREAM t."Col_A",
   t."Col_B",
   t."Col_C",
   t.r."COL_1",
   t.r."COL_2",
   t.r."COL_3"
FROM
```

```
   (
      SELECT
         STREAM "Col_A",
         "Col_B",
         "Col_C",
         VARIABLE_COLUMN_LOG_PARSE ("Col_E_Unstructured",
         'COL_1 TYPE VARCHAR(16),
         COL_2 TYPE VARCHAR(16),
         COL_3 TYPE VARCHAR(16)', ',') AS r
      FROM
         "SOURCE_SQL_STREAM_001"
   )
   as t;
```

Managed Service for Apache Flink Studio

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

% flink.ssql(type=update)
   select
      SPLIT_INDEX(log, ' ', 0),
      SPLIT_INDEX(log, ' ', 1),
      SPLIT_INDEX(log, ' ', 2),
      SPLIT_INDEX(log, ' ', 3),
      SPLIT_INDEX(log, ' ', 4),
      SPLIT_INDEX(log, ' ', 5)
)
from
   SOURCE_SQL_STREAM_001;
```

聯結

SQL-based Kinesis Data Analytics application

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

Managed Service for Apache Flink Studio

```
Query 1 - % flink.ssql(type =
update
) CREATE TABLE DESTINATION_SQL_STREAM (
   TICKER_SYMBOL VARCHAR(4),
   SECTOR VARCHAR(12),
   CHANGE INT,
   PRICE DOUBLE )
PARTITIONED BY (TICKER_SYMBOL) WITH (
   'connector' = 'kinesis',
   'stream' = 'kinesis-analytics-demo-stream',
   'aws.region' = 'us-east-1',
   'scan.stream.initpos' = 'LATEST',
   'format' = 'json',
   'json.timestamp-format.standard' = 'ISO-8601');

Query 2 - CREATE TABLE CompanyName (
```

```
   Ticker VARCHAR(4),
   Company VARCHAR(4)) WITH (
      'connector' = 'filesystem',
      'path' = 's3://kda-demo-sample/TickerReference.csv',
      'format' = 'csv' );

Query 3 - % flink.ssql(type =
update
)
   SELECT
      TICKER_SYMBOL,
      c.Company,
      SECTOR,
      CHANGE,
      PRICE
   FROM
      DESTINATION_SQL_STREAM
      LEFT JOIN
         CompanyName as c
         ON DESTINATION_SQL_STREAM.TICKER_SYMBOL = c.Ticker;
```

錯誤

SQL-based Kinesis Data Analytics application

```
SELECT
   STREAM ticker_symbol,
   sector,
   change,
   (
      price / 0
   )
   as ProblemColumnFROM "SOURCE_SQL_STREAM_001"
WHERE
   sector SIMILAR TO '%TECH%';
```

Managed Service for Apache Flink Studio

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

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

Query 2 - % flink.pyflink @udf(input_types = [DataTypes.BIGINT()],
   result_type = DataTypes.BIGINT()) def DivideByZero(price): try: price / 0
except
: return - 1 st_env.register_function("DivideByZero",
   DivideByZero)

   Query 3 - % flink.ssql(type =
update
)
   SELECT
      CURRENT_TIMESTAMP AS ERROR_TIME,
      *
   FROM
      (
         SELECT
            TICKER_SYMBOL,
            SECTOR,
            CHANGE,
            DivideByZero(PRICE) as ErrorColumn
         FROM
            DESTINATION_SQL_STREAM
         WHERE
            SECTOR SIMILAR TO '%TECH%'
      )
      AS ERROR_STREAM;
```

遷移隨機分割森林工作負載

如果您想要將使用隨機分割森林的工作負載從 Kinesis Analytics for SQL 移動到 Managed Service for Apache Flink，此AWS部落格文章示範如何使用 Managed Service for Apache Flink 來執行線上 RCF 演算法進行異常偵測。

將 Kinesis Data Firehose 取代為具有 Kinesis Data Streams 的來源

如需完整的教學課程，請參閱 Converting-KDASQL-KDAStudio/。

在下列練習中，您將變更資料流程來使用 Amazon Managed Service for Apache Flink Studio。這也意味著從 Amazon Kinesis Data Firehose 切換到 Amazon Kinesis Data Streams。

首先，我們分享一個典型的 KDA-SQL 架構，接著展示如何使用 Amazon Managed Service for Apache Flink Studio 和 Amazon Kinesis Data Streams.替換此架構。或您也可以在此處啟動 AWS CloudFormation 範本：

Amazon Kinesis Data Analytics-SQL 和 Amazon Kinesis Data Firehose

以下是 Amazon Kinesis Data Analytics SQL 架構流程：

我們首先檢查傳統 Amazon Kinesis Data Analytics-SQL 和 Amazon Kinesis Data Firehose 的設置。此使用案例是交易市場，其中包括股票代號和價格在內的交易資料會從外部來源串流至 Amazon Kinesis 系統。Amazon Kinesis Data Analytics for SQL 會使用輸入串流執行視窗查詢 (例如輪轉視窗)，以判斷每個股票代號在一分鐘視窗內的交易量和 `min`、`max` 與`average` 交易價格。

Amazon Kinesis Data Analytics-SQL 已準備好從 Amazon Kinesis Data Firehose API 擷取資料。處理完畢後，Amazon Kinesis Data Analytics-SQL 會將處理過的資料傳送到另一個 Amazon Kinesis Data Firehose，然後將輸出儲存在 Amazon S3 儲存貯體中。

在這種情況下，您可以使用 Amazon Kinesis 資料產生器。Amazon Kinesis 資料產生器可讓您將測試資料傳送到 Amazon Kinesis Data Streams 或 Amazon Kinesis Data Firehose 交付串流。如要開始使用，請按照此處的說明進行操作。使用此處的 AWS CloudFormation 範本代替說明中提供的範本：。

執行 AWS CloudFormation 範本後，輸出區段將提供網址給 Amazon Kinesis 資料產生器。使用您在此處設定的 Cognito 使用者 ID 和密碼登入入口網站。選取地區和目標串流名稱。針對目前的狀態，選擇 Amazon Kinesis Data Firehose 交付串流。針對新狀態，選擇 Amazon Kinesis Data Firehose 串流名稱。您可以根據需求建立多個範本，並在傳送範本至目標串流前使用測試範本按鈕來測試範本。

以下是使用 Amazon Kinesis 資料產生器的範例承載。資料產生器之目標為 Amazon Kinesis Firehose 的輸入串流，以持續串流資料。Amazon Kinesis SDK 用戶端也可以從其他生產者傳送資料。

```
2023-02-17 09:28:07.763,"AAPL",5032023-02-17 09:28:07.763,
"AMZN",3352023-02-17 09:28:07.763,
"GOOGL",1852023-02-17 09:28:07.763,
"AAPL",11162023-02-17 09:28:07.763,
"GOOGL",1582
```

以下 JSON 用於生成一系列隨機的交易時間和日期，股票代號和股票價格：

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

選擇傳送資料後，生成器將開始傳送模擬資料。

外部系統會將資料串流到 Amazon Kinesis Data Firehose。使用 Amazon Kinesis Data Analytics for SQL 應用程式，您可以用標準 SQL 來分析串流資料。此服務可讓您針對串流來源撰寫和執行 SQL 程式碼，以執行時間序列分析、饋送即時儀表板，以及建立即時指標。Amazon Kinesis Data Analytics for SQL 應用程式可以從輸入串流上的 SQL 查詢建立目標串流，然後將目標串流傳送到另一個 Amazon Kinesis Data Firehose。目的地 Amazon Kinesis Data Firehose 可以將分析資料傳送到 Amazon S3 做為最終狀態。

Amazon Kinesis Data Analytics-SQL 舊版程式碼的基礎，是 SQL 標準的延伸模組。

在 Amazon Kinesis Data Analytics-SQL 中使用以下查詢。首先建立查詢輸出的目標串流。然後，您可以使用 PUMP Amazon Kinesis Data Analytics 儲存庫物件 (SQL 標準的延伸模組)，提供持續執行的 `INSERT INTO stream SELECT ... FROM` 查詢功能，進而讓查詢結果持續輸入到具名串流中。

```
CREATE
OR REPLACE STREAM "DESTINATION_SQL_STREAM" (EVENT_TIME TIMESTAMP,
INGEST_TIME TIMESTAMP,
TICKER VARCHAR(16),
VOLUME BIGINT,
AVG_PRICE DOUBLE,
```

```
MIN_PRICE DOUBLE,
MAX_PRICE DOUBLE);

CREATE
OR REPLACE PUMP "STREAM_PUMP" AS
INSERT INTO
   "DESTINATION_SQL_STREAM"
   SELECT
      STREAM STEP("SOURCE_SQL_STREAM_001"."tradeTimestamp" BY INTERVAL '60' SECOND) AS
 EVENT_TIME,
      STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND) AS
 "STREAM_INGEST_TIME",
      "ticker",
       COUNT(*) AS VOLUME,
      AVG("tradePrice") AS AVG_PRICE,
      MIN("tradePrice") AS MIN_PRICE,
      MAX("tradePrice") AS MAX_PRICEFROM "SOURCE_SQL_STREAM_001"
   GROUP BY
      "ticker",
      STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND),
      STEP("SOURCE_SQL_STREAM_001"."tradeTimestamp" BY INTERVAL '60' SECOND);
```

上述 SQL 使用兩個時間視窗：tradeTimestamp 來自傳入串流有效負載，ROWTIME.tradeTimestamp 也稱為 Event Time 或 client-side time。需要在分析中偏好使用此時間，因其為事件發生的時間。不過，許多事件來源 (例如行動電話和 Web 用戶端) 沒有可靠的時鐘，這可能會導致不正確的時間。此外，連線問題可能會導致串流上的記錄顯示順序與事件發生的順序不相同。

應用程式內串流也包含一個名為 ROWTIME 的特殊資料行。當 Amazon Kinesis Data Analytics 在第一個應用程式內串流中插入資料列時，會儲存時間戳記。 ROWTIME 指的是 Amazon Kinesis Data Analytics 從串流來源讀取後，將記錄插入第一個應用程式內串流的時間戳記。接著整個應用程式中皆會保留此 ROWTIME 值。

SQL 確定股票代號的計數為 volumemin，max和 60 秒間隔內的average價格。

在時間類型的視窗查詢中，使用其中的任一個時間都有優點和缺點。選擇其中一個或多個時間，與根據使用案例情境來處理相關缺點的策略。

雙視窗策略使用兩個時間類型，包含 ROWTIME 與另一個其他時間，如事件時間。

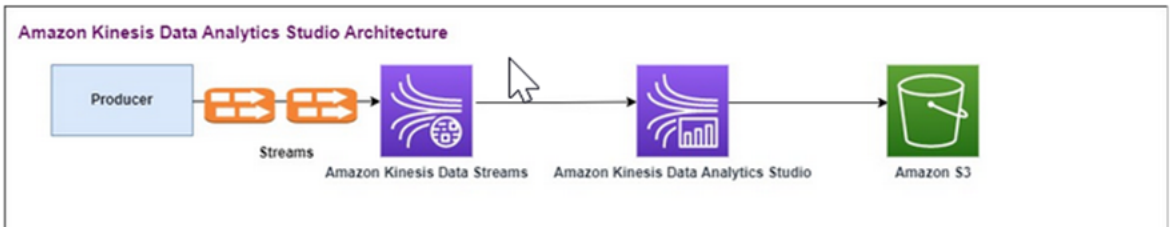
- 將 ROWTIME 當作第一個視窗，此視窗可控制查詢發出結果的頻率，如下列範例所示。這並非邏輯時間。

- 把其中一個其他時間當作邏輯時間，即您想要連結到分析的時間 此時間表示事件發生的時間。在下面的例子中，分析目標是按股票代號對記錄進行分組和返回計數。

Amazon Managed Service for Apache Flink Studio

在更新的架構中，您可以使用 Amazon Kinesis Data Streams 取代 Amazon Kinesis Data Firehose。Amazon Kinesis Data Analytics for SQL 應用程式已由 Amazon Managed Service for Apache Flink Studio 取代。Apache Flink 程式碼會在 Apache Zeppelin 筆記本中交互運行。Amazon Managed Service for Apache Flink Studio 會將彙總的交易資料傳送到 Amazon S3 儲存貯體來儲存。步驟如下所示：

此為 Amazon Managed Service for Apache Flink Studio 的架構流程：



建立 Kinesis Data Stream

使用主控台建立資料串流

1. 前往 https://console.aws.amazon.com/kinesis/ 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽列中，展開區域選擇工具，然後選擇一個區域。
3. 選擇 建立資料串流。
4. 在建立 Kinesis 串流頁面上，輸入資料串流的名稱，然後接受預設的隨需容量模式。

   在隨需模式下，您可以選擇建立 Kinesis 串流來建立資料串流。

   建立串流時，在 Kinesis 串流頁面上，串流的狀態會是正在建立。當串流就緒可供使用後，其狀態將變成作用中。
5. 選擇串流名稱。串流詳細資訊頁面會顯示串流組態的摘要以及監控資訊。
6. 在 Amazon Kinesis 資料產生器中，將串流/交付串流變更為新的 Amazon Kinesis Data Streams：TRADE_SOURCE_STREAM。

JSON 和承載會與您用於 Amazon Kinesis Data Analytics-SQL 的相同。使用 Amazon Kinesis 資料產生器產生一些交易承載資料範例，並針對本練習將 TRADE_SOURCE_STREAM 資料串流設為目標：

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

7. 在 AWS Management Console 上，至 Managed Service for Apache Flink 並選擇建立應用程式。
8. 在左邊的導覽窗格中，選擇 Studio 筆記本，然後選擇建立 Studio 筆記本。
9. 輸入 Studio 筆記本的名稱。
10. 在 AWSGlue 資料庫中，提供現有的資料 AWS Glue 資料庫，以定義您的來源和目的地之中繼資料。如果您沒有 AWS Glue 資料庫，請選擇建立，然後執行下列動作：
    a. 在 AWS Glue 主控台中，從左側選單選擇資料型錄中的資料庫。
    b. 選擇建立資料型錄。
    c. 在建立資料庫頁面中輸入資料庫的名稱。在位置 - 選用區段中，選擇瀏覽 Amazon S3並選取 Amazon S3 儲存貯體。如果您還沒有設定好 Amazon S3 儲存貯體，您可以跳過此步驟，稍後再回來。
    d. (選用)。輸入資料庫的說明。
    e. 選擇建立資料庫。
11. 選擇建立筆記本。
12. 建立您的筆記本後，選擇執行。
13. 筆記本成功啟用後，選擇在 Apache Zeppelin 中打開來啟動 Zeppelin 筆記本。
14. 在 Zeppelin 筆記本頁面上，選擇建立新的筆記並將其命名為 MarketDataFeed。

Flink SQL 程式碼解釋如下，但首先，這是 Zeppelin 筆記本的畫面。筆記本中的每個視窗都是單獨的程式碼區塊，一次可運行一個。

Amazon Managed Service for Apache Flink Studio 程式碼

Amazon Managed Service for Apache Flink Studio 使用 Zeppelin 筆記本來運行程式碼。此範例以 Apache Flink 1.13 為基礎映射到 ssql 程式碼。Zeppelin 筆記本中的程式碼一次顯示在一個區塊的下方。

在您的 Zeppelin 筆記本運行任何程式碼前，必須運行 Flink 組態命令。如果您需要在運行程式碼（ssql，Python 或 Scala）後更改任何組態設定，則需要停止並重新啟動筆記本。在此範例中，您需要設定檢查點。需要檢查點，才能將資料串流到 Amazon S3 中的檔案。這可將串流至 Amazon S3 的資料排清到檔案中。下面的陳述式將間隔設置為 5000 毫秒。

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

`%flink.conf` 表示此區塊為組態陳述式。如需有關 Flink 組態 (包括檢查點) 的詳細資訊，請參閱 Apache Flink 檢查點。

來源 Amazon Kinesis Data Streams 的輸入表是以下列 Flink ssql 程式碼建立的。請注意，TRADE_TIME 字段會儲存由資料生成器創建的日期/時間。

```
%flink.ssql

DROP TABLE IF EXISTS TRADE_SOURCE_STREAM;
CREATE TABLE TRADE_SOURCE_STREAM (--`arrival_time` TIMESTAMP(3) METADATA FROM
 'timestamp' VIRTUAL,
TRADE_TIME TIMESTAMP(3),
WATERMARK FOR TRADE_TIME as TRADE_TIME - INTERVAL '5' SECOND,TICKER STRING,PRICE
 DOUBLE,
STATUS STRING)WITH ('connector' = 'kinesis','stream' = 'TRADE_SOURCE_STREAM',
'aws.region' = 'us-east-1','scan.stream.initpos' = 'LATEST','format' = 'csv');
```

您可以使用以下陳述式查看輸入串流：

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

select * from TRADE_SOURCE_STREAM;
```

在彙總資料傳送到 Amazon S3 之前，您可以用翻轉視窗選擇查詢在 Amazon Managed Service for Apache Flink 中直接檢視該資料。此舉會在一分鐘的時間視窗中彙總交易數據。請注意，%flink.ssql 陳述式必須具有 (類型 = 更新) 指定：

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

select TUMBLE_ROWTIME(TRADE_TIME,
INTERVAL '1' MINUTE) as TRADE_WINDOW,
TICKER, COUNT(*) as VOLUME,
```

```
AVG(PRICE) as AVG_PRICE,
MIN(PRICE) as MIN_PRICE,
MAX(PRICE) as MAX_PRICE FROM TRADE_SOURCE_STREAMGROUP BY TUMBLE(TRADE_TIME, INTERVAL
 '1' MINUTE), TICKER;
```

然後，您可以在 Amazon S3 中建立目的地路由表。您需要使用浮水印。浮水印是一種進度指標，指出您確信不會再有延遲事件的時間點。浮水印是為了因應遲到的情形。間隔 '5' Second 允許交易延遲 5 秒進入 Amazon Kinesis Data Stream，如果在視窗內有時間戳記，則仍會包含在內。如需詳細資訊，請參閱產生浮水印。

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

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

此陳述式會將資料插入到 TRADE_DESTINATION_S3。TUMPLE_ROWTIME 是翻轉視窗包容性上界的時間戳記。

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

insert into TRADE_DESTINATION_S3
select TUMBLE_ROWTIME(TRADE_TIME,
INTERVAL '1' MINUTE),
TICKER, COUNT(*) as VOLUME,
AVG(PRICE) as AVG_PRICE,
MIN(PRICE) as MIN_PRICE,
MAX(PRICE) as MAX_PRICE FROM TRADE_SOURCE_STREAM
GROUP BY TUMBLE(TRADE_TIME, INTERVAL '1' MINUTE), TICKER;
```

讓陳述式執行 10 到 20 分鐘，以便在 Amazon S3 中累積一些資料。然後中止你的陳述式。

此舉會關閉 Amazon S3 中的檔案，讓其成為可檢視狀態。

以下是內容的樣子：

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

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

您可以使用AWS CloudFormation範本來建立基礎架構。

AWS CloudFormation 會在您的 AWS 帳戶中建立下列資源：

- Amazon Kinesis Data Streams
- Amazon Managed Service for Apache Flink Studio
- Amazon Glue 資料庫
- Amazon S3 儲存貯體
- 適用於 Amazon Managed Service for Apache Flink Studio，可存取適當資源的 IAM 角色和政策

匯入筆記本，並使用由 AWS CloudFormation 建立的新 Amazon S3 儲存貯體變更 Amazon S3 儲存貯體名稱。

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

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

查看更多

以下是一些額外的資源，您可以用其進一步了解如何使用 Managed Service for Apache Flink Studio：

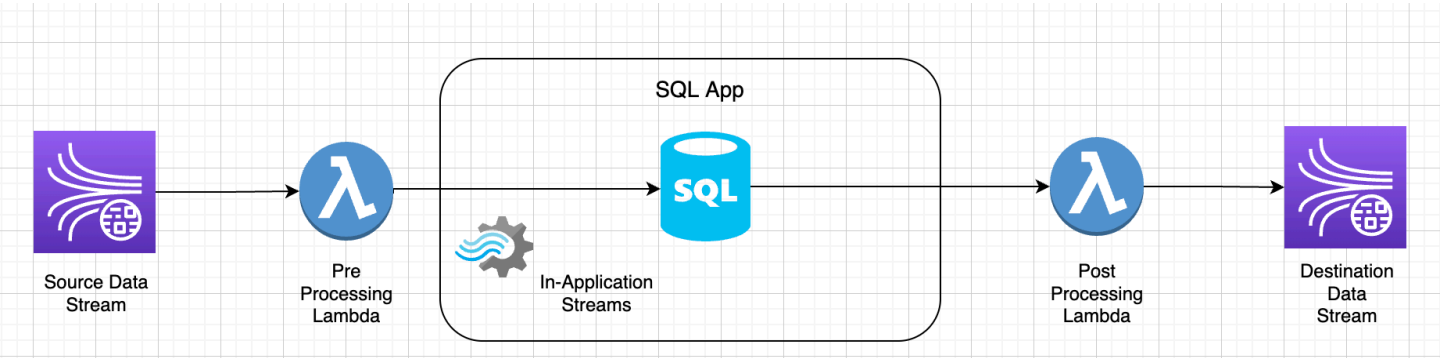
- Managed Service for Apache Flink 筆記本開發人員指南
- Apache Flink 1.13 文件
- Amazon Managed Service for Apache Flink Studio 研討會
- Apache Flink 視窗化
- Amazon Kinesis Data Analytics 開發人員指南 — 從 Kinesis Data Analytics 串流寫入 S3 儲存貯體

利用使用者定義函數 (UDF)

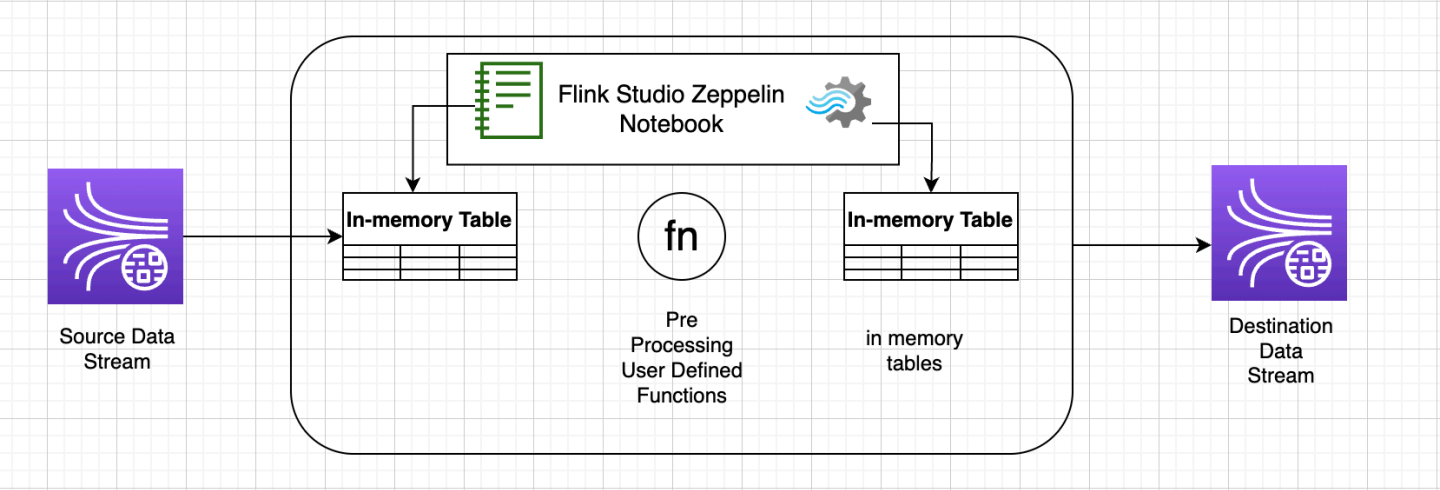
該模式的目的是示範如何在 Kinesis Data Analytics-Studio Zeppelin 筆記本中運用 UDF，以處理 Kinesis 串流中的資料。Managed Service for Apache Flink Studio 使用 Apache Flink 來提供進階分析能力，包括僅一次處理語意、事件時間視窗、使用者定義函數和客戶整合的擴充性、命令式語言支援、持久的應用程式狀態、水平擴展、支援多個資料來源、可延伸整合等等。這些對於確保資料串流處理的準確性，完整性，一致性和可靠性至關重要，且無法由 Amazon Kinesis Data Analytics for SQL 提供。

在此範例應用程式中，我們將示範如何利用 KDA-Studio Zeppelin 筆記本中的 UDF 來處理 Kinesis 串流中的資料。適用於 Kinesis Data Analytics 的 Studio 筆記本可讓您即時以互動方式查詢資料串流，並使用標準 SQL、Python 和 Scala 輕鬆建置和執行串流處理應用程式。只要在 AWS Management Console 按幾下滑鼠，即可啟動無伺服器筆記本，以查詢資料串流並在幾秒鐘內取得結果。如需詳細資訊，請參閱搭配 Kinesis Data Analytics for Apache Flink 使用 Studio 筆記本。

用於前/後處理 KDA-SQL 應用程式資料的 Lambda 函數：



使用 KDA-Studio Zeppelin 筆記本對資料進行前後處理的使用者定義函數



使用者定義的函數 (UDF)

若要將通用的商業邏輯重複使用到運算子中，不妨參考使用者定義函數來轉換資料串流。此舉可在 Managed Service for Apache Flink Studio 筆記本中完成，也可以將其當作外部引用的應用程式 JAR 文件。利用使用者定義的函數可以簡化轉換或資料擴充作業，這些作業可能會在串流資料上執行。

在筆記本中，您要引用一個簡單的 Java 應用程式 JAR，其具有匿名個人電話號碼的功能。您也可以編寫 Python 或 Scala UDF 以便在筆記本中使用。我們選擇了一個 Java 應用程式 JAR 來強調將應用程式 JAR 導入 Pyflink 筆記本的功能。

環境設定

若要遵循本指南並與您的串流資料互動，請使用 AWS CloudFormation 指令碼啟動下列資源：

- Kinesis Data Streams 做為來源與目標
- Glue 資料庫
- IAM 角色
- Managed Service for Apache Flink Studio 應用程式
- 啟動 Managed Service for Apache Flink Studio 應用程式的 Lambda 函數
- 要執行上述 Lambda 函數的 Lambda 角色
- 自訂資源，以調用 Lambda 函數

在此下載 AWS CloudFormation 範本。

建立 AWS CloudFormation 堆疊。

1. 至 AWS Management Console 並在服務列表中選擇 CloudFormation。
2. 在CloudFormation頁面上，選擇堆疊，並選擇用新資源建立堆疊 (標準)。
3. 在建立堆疊頁面上，選擇上傳範本檔案，然後選擇您先前下載的 `kda-flink-udf.yml` 檔案。選擇檔案，然後選擇下一步。
4. 給模板一個名稱便於記憶，如 `kinesis-UDF`。如想要不同名稱的話可更新輸入參數，如輸入串流。選擇下一步。
5. 在設定堆疊選項頁面，視需要新增標籤，然後選擇下一步。
6. 在檢閱頁面，勾選允許建立 IAM 資源的方塊，然後選擇提交。

AWS CloudFormation 堆疊可能需要 10 到 15 分鐘才能啟動，具體取決於您啟動的地區。一旦您看到整個堆疊的 `CREATE_COMPLETE` 狀態，就可以繼續。

使用 Managed Service for Apache Flink Studio 筆記本

適用於 Kinesis Data Analytics 的 Studio 筆記本可讓您即時以互動方式查詢資料串流，並使用標準 SQL、Python 和 Scala 輕鬆建置和執行串流處理應用程式。只要在 AWS Management Console 按幾下滑鼠，即可啟動無伺服器筆記本，以查詢資料串流並在幾秒鐘內取得結果。

筆記本是基於 Web 的開發環境。使用筆記本，您不僅能獲得簡單的互動式開發體驗，還能使用 Apache Flink 提供的進階資料串流處理功能。Studio 筆記本使用由 Apache Zeppelin 提供支援的筆記本，使用 Apache Flink 作為串流處理引擎。Studio 筆記本無縫結合了這些技術，讓所有技能背景的開發人員都能存取資料串流的進階分析。

Apache Zeppelin 為您的 Studio 筆記本提供了完整的分析工具套件，包括以下專案：

- 資料視覺化
- 將資料匯出到檔案
- 控制輸出格式以便於分析

使用筆記本

1. 至 AWS Management Console 並在服務列表中選擇 Amazon Kinesis。
2. 在左側導覽頁面上，選擇分析應用程式，然後選擇 Studio 筆記本。
3. 確認 KinesisDataAnalyticsStudio 筆記本正在執行。

4. 選擇筆記本，然後選擇在 Apache Zeppelin 中打開。
5. 下載資料生產者 Zeppelin 筆記本檔案，您可以使用該檔案讀取資料並將其載入 Kinesis 串流。
6. 匯入 Data ProducerZeppelin 筆記本。確保您有在筆記本程式碼中修改輸入 STREAM_NAME 和 REGION。輸入串流名稱可以在 AWS CloudFormation堆疊輸出中找到。
7. 選擇執行此段落按鈕，將樣本資料插入輸入 Kinesis 資料串流，以執行資料生產者筆記本。
8. 當樣本資料載入時，下載 MaskPhoneNumber-互動式筆記本，該筆記本會讀取輸入資料，從輸入串流中匿名化電話號碼，並將匿名數據儲存到輸出流中。
9. 匯入 MaskPhoneNumber-interactiveZeppelin 筆記本。
10. 執行筆記本中的每個段落。

    a. 在第 1 段，請匯入使用者定義函數以匿名化電話號碼。

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

    b. 在下一段，請建立記憶體內資料表來讀取輸入串流資料。請確認串流名稱和 AWS 區域正確無誤。

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

    DROP TABLE IF EXISTS customer_reviews;

    CREATE TABLE customer_reviews (
    customer_id VARCHAR,
    product VARCHAR,
    review VARCHAR,
    phone VARCHAR
    )
    WITH (
    'connector' = 'kinesis',
    'stream' = 'KinesisUDFSampleInputStream',
    'aws.region' = 'us-east-1',
    'scan.stream.initpos' = 'LATEST',
    'format' = 'json');
    ```

    c. 檢查資料是否已載入記憶體內資料表。

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

```
select * from customer_reviews
```

d. 調用用戶定義的功能以匿名化電話號碼。

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

e. 現在，電話號碼已被遮罩，請創建一個帶遮罩號碼的檢視。

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

DROP VIEW IF EXISTS sentiments_view;

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

f. 驗證資料。

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

g. 為輸出 Kinesis 串流建立記憶體內資料表。請確認串流名稱和 AWS 區域正確無誤。

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

DROP TABLE IF EXISTS customer_reviews_stream_table;

CREATE TABLE customer_reviews_stream_table (
customer_id VARCHAR,
product VARCHAR,
review VARCHAR,
phoneNumber varchar
)
WITH (
'connector' = 'kinesis',
'stream' = 'KinesisUDFSampleOutputStream',
'aws.region' = 'us-east-1',
'scan.stream.initpos' = 'TRIM_HORIZON',
```

```
'format' = 'json');
```

h. 在目標 Kinesis 串流中插入更新的記錄。

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

i. 檢視和驗證來自目標 Kinesis 串流的資料。

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

將筆記本提升為應用程式

現在您已經以互動方式測試筆記本程式碼，請將程式碼部署為具有持久狀態的串流應用程式。您必須先修改應用程式組態，以在 Amazon S3 中指定程式碼的位置。

1. 在 AWS Management Console，選擇您的筆記本，然後在部署為應用程式組態 - 選用中選擇編輯。
2. 在 Amazon S3 中的程式碼目的地中，選擇AWS CloudFormation指令碼建立的 Amazon S3 儲存貯體。該程序需要幾分鐘的時間。
3. 您無法按原樣提升筆記。嘗試的話會出錯，因為 `Select` 陳述式不受支援。若要避免這個問題，請下載MaskPhoneNumber - 串流 Zeppelin 筆記本。
4. 匯入 `MaskPhoneNumber-streaming` Zeppelin 筆記本。
5. 開啟筆記，然後選擇 KinesisDataAnalyticsStudio 的動作。
6. 選擇建立 MaskPhoneNumber - 串流並匯出至 S3。請務必重新命名應用程式名稱，且不要用特殊字元。
7. 選擇建置和匯出。需要幾分鐘的時間來設定串流應用程式。
8. 建置完成後，請選擇使用 AWS 主控台部署。
9. 在下一頁檢閱設定，並確保選擇正確的 IAM 角色。接下來，選擇建立串流應用程式。
10. 幾分鐘後，您會看到串流應用程式已成功建立的訊息。

如需部署具有持久狀態和限制之應用程式的詳細資訊，請參閱部署為具有持久狀態的應用程式。

清除

或者，您現在也可解除安裝 AWS CloudFormation 堆疊。此舉將刪除您之前設定的所有服務。

Kinesis Data Analytics for SQL 範例

本節提供在 Amazon Kinesis Data Analytics 中建立及使用應用程式的範例。其中包含範例程式碼和逐步指示，可協助您建立 Kinesis Data Analytics 應用程式並測試結果。

在探索這些範例之前，我們建議先檢閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式 與 Amazon Kinesis Data Analytics for SQL 應用程式入門：

主題

範例：轉換資料

有時候，您的應用程式碼必須預先處理傳入的記錄，然後才能在 Amazon Kinesis Data Analytics 中執行分析。發生這種情況的原因有許多種，例如記錄不符合支援的記錄格式，導致應用程式內輸入串流中產生非標準化的資料欄。

本節提供了如何使用可用的字串函數來標準化數據，與如何從字串資料欄提取所需資訊等範例。本節還指出了您可能會覺得有用的日期時間函數。

使用 Lambda 預處理串流

如需使用預先處理串流的詳細資訊 AWS Lambda，請參閱使用 Lambda 函數預處理資料。

主題

範例：轉換字串值

Amazon Kinesis Data Analytics 支援 JSON 和 CSV 等格式，用於串流來源上的記錄。如需詳細資訊，請參閱RecordFormat。然後，這些記錄會根據輸入組態映射到應用程式內串流的列。如需詳細資訊，請參閱設定應用程式輸入。輸入組態會指定串流來源中的記錄欄位的映射方式，將其對應至應用程式內串流中的資料欄。

當串流來源上的記錄遵循支援的格式時，此映射就會運作，產出內含標準化資料的應用程式內串流。但如果串流來源上的資料不符合支援的標準，該怎麼辦？ 舉例來說，如果您的串流來源包含點擊流資料、IoT 感應器和應用程式日誌等資料，該怎麼辦？

請考量以下範例：

- 串流來源包含應用程式日誌：應用程式日誌會遵循標準 Apache 日誌格式，並使用 JSON 格式寫入串流。

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

  如需有關標準 Apache 日誌格式的詳細資訊，請參閱 Apache 網站上的日誌檔案。

- 串流來源包含半結構化資料：下列範例顯示兩筆記錄。Col_E_Unstructured 欄位值是一系列逗號分隔值。共有五個資料欄：前四個資料行具有字串類型值，最後一欄包含逗號分隔值。

  ```
  { "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"}
  ```

- 串流來源上的記錄包含 URL，而您需要部分 URL 網域名稱以進行分析。

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

在這種情況下，下列兩步驟程序通常適用於建立內含標準化資料的應用程式內串流：

1. 設定應用程式輸入，將非結構化欄位映射至所建立的應用程式內輸入串流中 VARCHAR(N) 類型的資料欄。
2. 在應用程式碼中，使用字串函數將此單一欄分割成多個資料欄，然後將資料列儲存在另一個應用程式內串流中。您的應用程式碼建立的應用程式內串流將有標準化資料。接著您可以分析應用程式內串流。

Amazon Kinesis Data Analytics 提供下列字串操作、標準 SQL 函數，以及 SQL 標準的擴充功能，以便使用字串欄：

- 字串運算子：運算子 (例如 LIKE 和 SIMILAR) 在比較字串時很有用。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的字串運算子。
- SQL 函數：下列函數在操作個別字串時非常有用。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的字串與搜尋函數。
  - CHAR_LENGTH - 提供字串的長度。
  - INITCAP - 傳回輸入字串的轉換版本，使得每個空格分隔文字的第一個字元都是大寫字母，而其他所有字元都是小寫字母。
  - LOWER/UPPER - 將字串轉換為小寫或大寫。
  - OVERLAY - 以第二個字串引數 (取代字串) 取代第一個字串引數 (原始字串) 的一部分。
  - POSITION - 搜尋另一個字串中的字串。
  - REGEX_REPLACE - 以替代子字串取代子字串。
  - SUBSTRING - 從特定位置開始擷取來源字串的一部分。
  - TRIM - 從來源字串的開頭或結尾移除指定字元的執行個體。
- SQL 擴充功能:這些對於使用非結構化字串 (例如日誌和 URI) 非常有用。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的日誌剖析函數。
  - FAST_REGEX_LOG_PARSER - 工作方式類似於 regex 剖析器，但需要幾個快捷方式來確保更快的結果。舉例來說，快速 regex 剖析器找到第一個匹配 (稱為懶惰語義) 後就會停止。
  - FIXED_COLUMN_LOG_PARSE - 剖析固定寬度欄位，並自動將其轉換為指定的 SQL 類型。

- REGEX_LOG_PARSE - 根據預設 Java 規則運算式模式剖析字串。
- SYS_LOG_PARSE - 剖析 UNIX/Linux 系統日誌中常見的項目。
- VARIABLE_COLUMN_LOG_PARSE - 將輸入字串分割為以分隔符號字元或字串分隔的欄位。
- W3C_LOG_PARSE - 可用於快速格式化 Apache 日誌。

如需使用這些函式的範例，請參閱主題：

主題

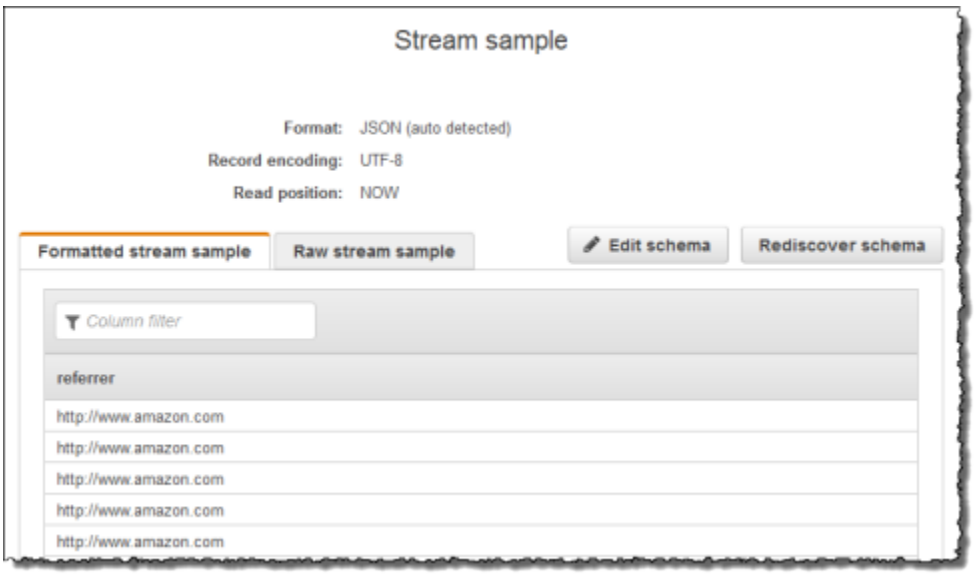
範例：擷取字串的一部分 (子字串函數)

此範例使用 SUBSTRING 函數來轉換 Amazon Kinesis Data Analytics 中的串流。SUBSTRING 函數從特定位置擷取來源字串的一部分。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的子字串。

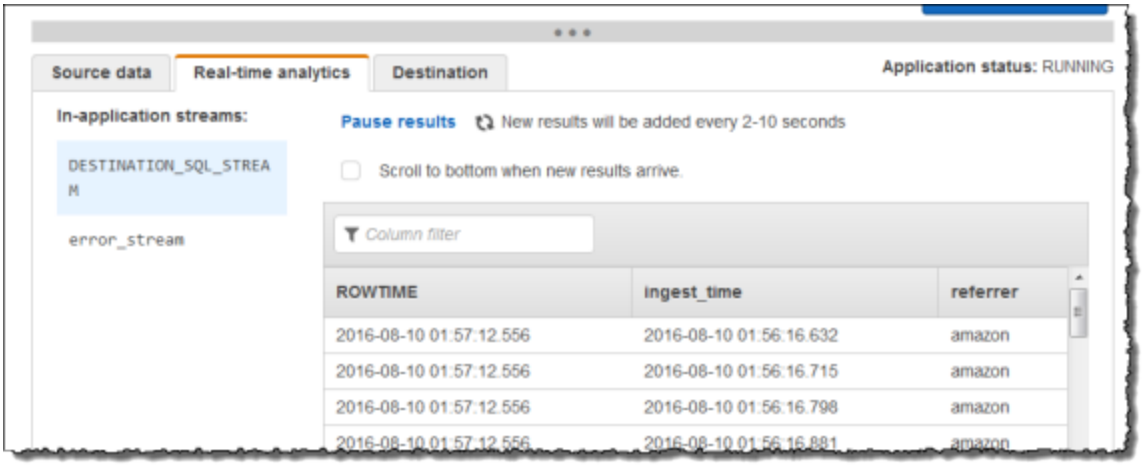
在此範例中，將下列記錄寫入 Amazon Kinesis 資料串流。

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

接著，您可以在主控台上建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流當作串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有一個資料欄 (REFERRER) 的應用程式內結構描述，如下所示。



然後，使用應用程式碼搭配 SUBSTRING 函數來剖析 URL 字串，以擷取公司名稱。接著將產生的資料插入另一個應用程式內串流，如下所示：



主題

- 步驟 1：建立 Kinesis 資料串流
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis 資料串流

建立 Amazon Kinesis 資料串流，並填入日誌紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。

2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 執行下列 Python 程式碼，填入範例日誌記錄。這個簡單的代碼會持續寫入相同的日誌記錄到串流。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

接下來，建立 Kinesis Data Analytics 應用程式，如下所示：

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料。
4. 在連接至來源頁面，執行下列動作：

a. 選擇您在上一節建立的串流。

b. 選擇建立 IAM 角色 選項。

c. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述只有一個資料欄。

d. 選擇儲存並繼續。

5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。

6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

a. 請複製以下應用程式碼，然後貼到編輯器中。

```
-- 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. 選擇儲存並執行 SQL。在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

範例：使用 Regex (REGEX_REPLACE 函數) 替換子字串

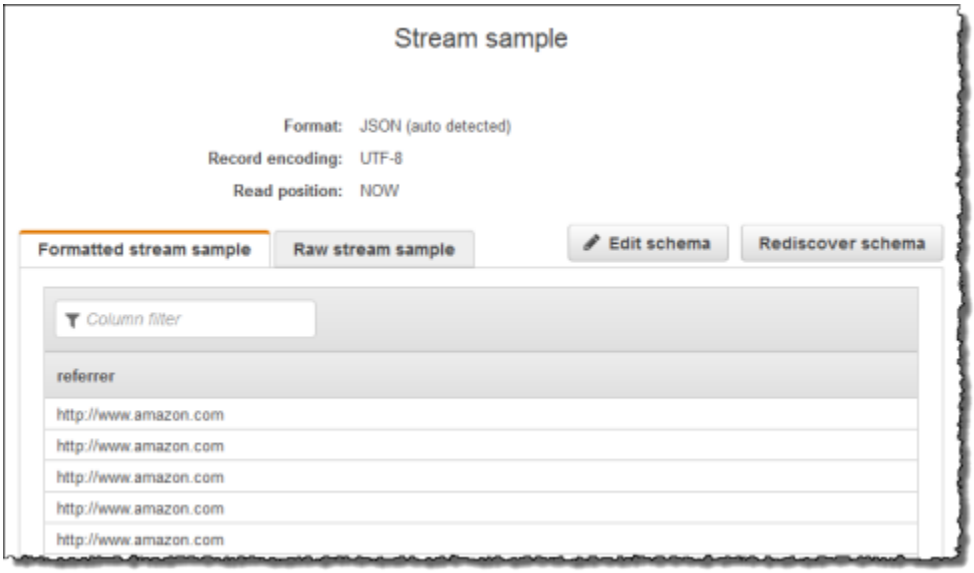
此範例使用 REGEX_REPLACE 函數來轉換 Amazon Kinesis Data Analytics 中的字串。REGEX_REPLACE 用替代子字串替換一個子字串。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的REGEX_REPLACE。

在此範例中，將下列記錄寫入 Amazon Kinesis data stream：

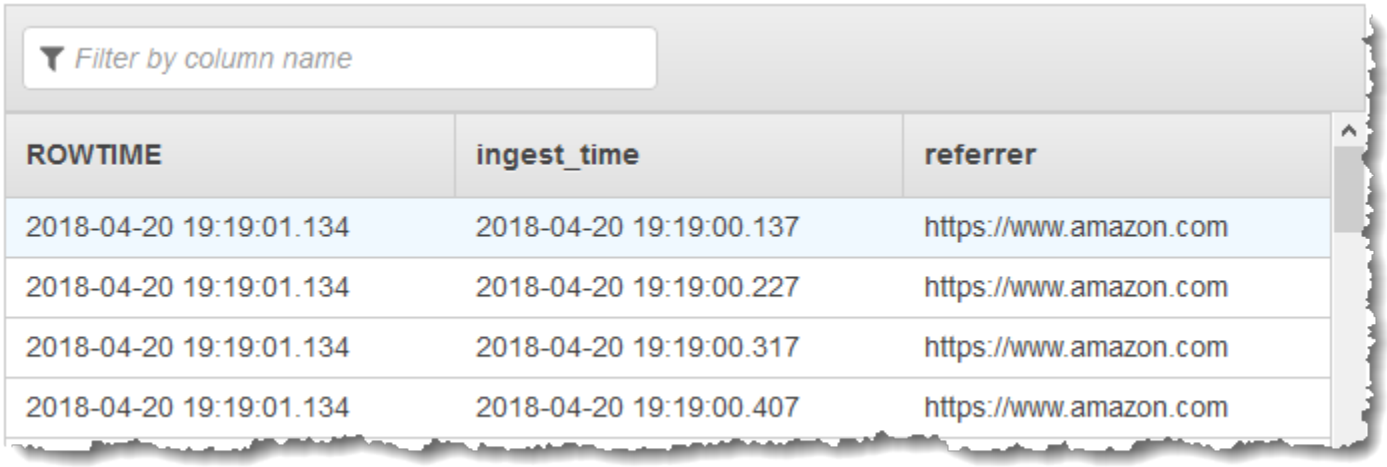
```
{ "REFERRER" : "http://www.amazon.com" }
{ "REFERRER" : "http://www.amazon.com"}
```

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

接著，您可以在主控台上建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有一個資料欄 (REFERRER) 的應用程式內結構描述，如下所示。



然後，將應用程式碼與 REGEX_REPLACE 函數搭配使用，以轉換 URL 使用 `https://` 而不是 `http://`。接著將產生的資料插入另一個應用程式內串流，如下所示：



主題

- 步驟 1：建立 Kinesis 資料串流
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis 資料串流

建立 Amazon Kinesis 資料串流，並填入日誌紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 執行下列 Python 程式碼，以填入範例日誌記錄。這個簡單的代碼會持續寫入相同的日誌記錄到串流。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

接下來，建立 Kinesis Data Analytics 應用程式，如下所示：

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料。
4. 在連接至來源頁面，執行下列動作：
   a. 選擇您在上一節建立的串流。
   b. 選擇建立 IAM 角色 選項。
   c. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述只有一個資料欄。
   d. 選擇儲存並繼續。
5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：
   a. 請複製以下應用程式碼，然後貼到編輯器中：

```
-- 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. 選擇儲存並執行 SQL。在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

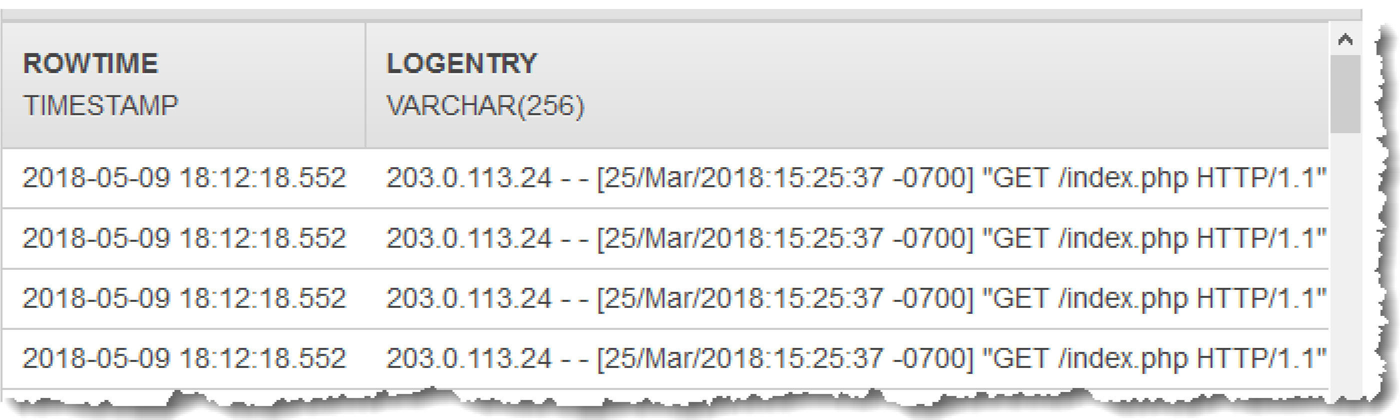
範例：根據規則運算式剖析日誌字串 (REGEX_LOG_PARSE 函數)

此範例使用 REGEX_LOG_PARSE 函數來轉換 Amazon Kinesis Data Analytics 中的字串。REGEX_LOG_PARSE 根據預設 Java 規則運算式模式剖析字串。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的[REGEX_LOG_PARSE](#)。

在此範例中，將下列記錄寫入 Amazon Kinesis 資料串流：

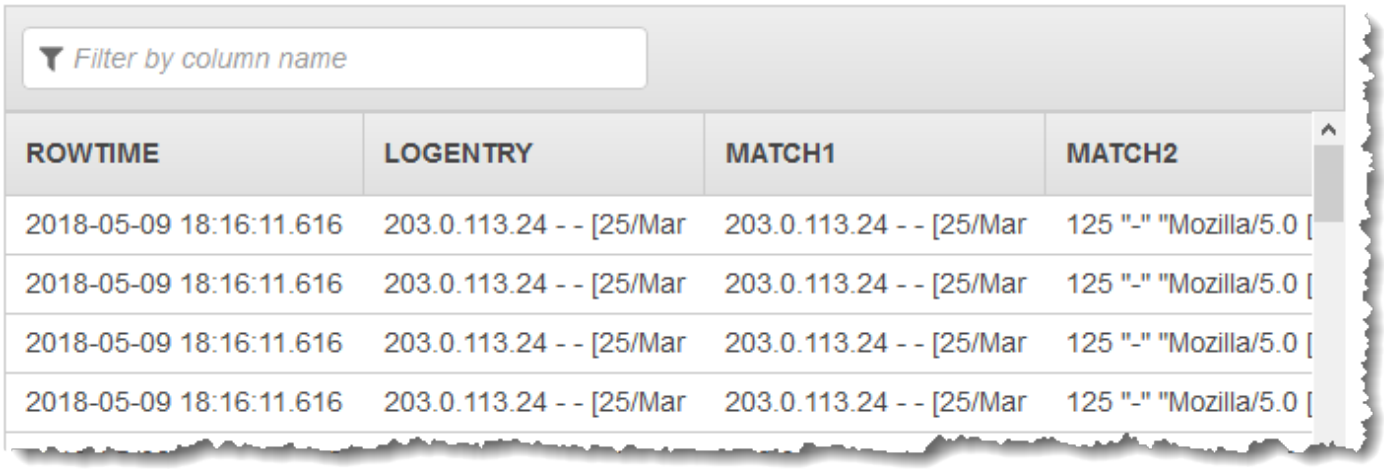
```
{"LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] \"GET /index.php HTTP/1.1\"
 200 125 \"-\" \"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0\""}
{"LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] \"GET /index.php HTTP/1.1\"
 200 125 \"-\" \"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0\""}
{"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\""}
...
```

接著，在主控台上建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有一個資料欄 (LOGENTRY) 的應用程式內結構描述，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"

然後，使用應用程式碼搭配 REGEX_LOG_PARSE 函數來剖析日誌字串，以擷取資料元素。接著將產生的資料插入另一個應用程式內串流，如下列螢幕擷取畫面所示：



主題

- 步驟 1：建立 Kinesis 資料串流
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis 資料串流

建立 Amazon Kinesis 資料串流，並填入日誌紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 執行下列 Python 程式碼，填入範例日誌記錄。這個簡單的代碼會持續寫入相同的日誌記錄到串流。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

接下來，建立 Kinesis Data Analytics 應用程式，如下所示：

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，然後指定應用程式名稱。
3. 在應用程式詳細資料頁面上，選擇連接串流資料。
4. 在連接至來源頁面，執行下列動作：
   a. 選擇您在上一節建立的串流。
   b. 選擇建立 IAM 角色 選項。
   c. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述只有一個資料欄。
   d. 選擇儲存並繼續。
5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

a. 請複製以下應用程式碼，然後貼到編輯器中。

```
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. 選擇儲存並執行 SQL。在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

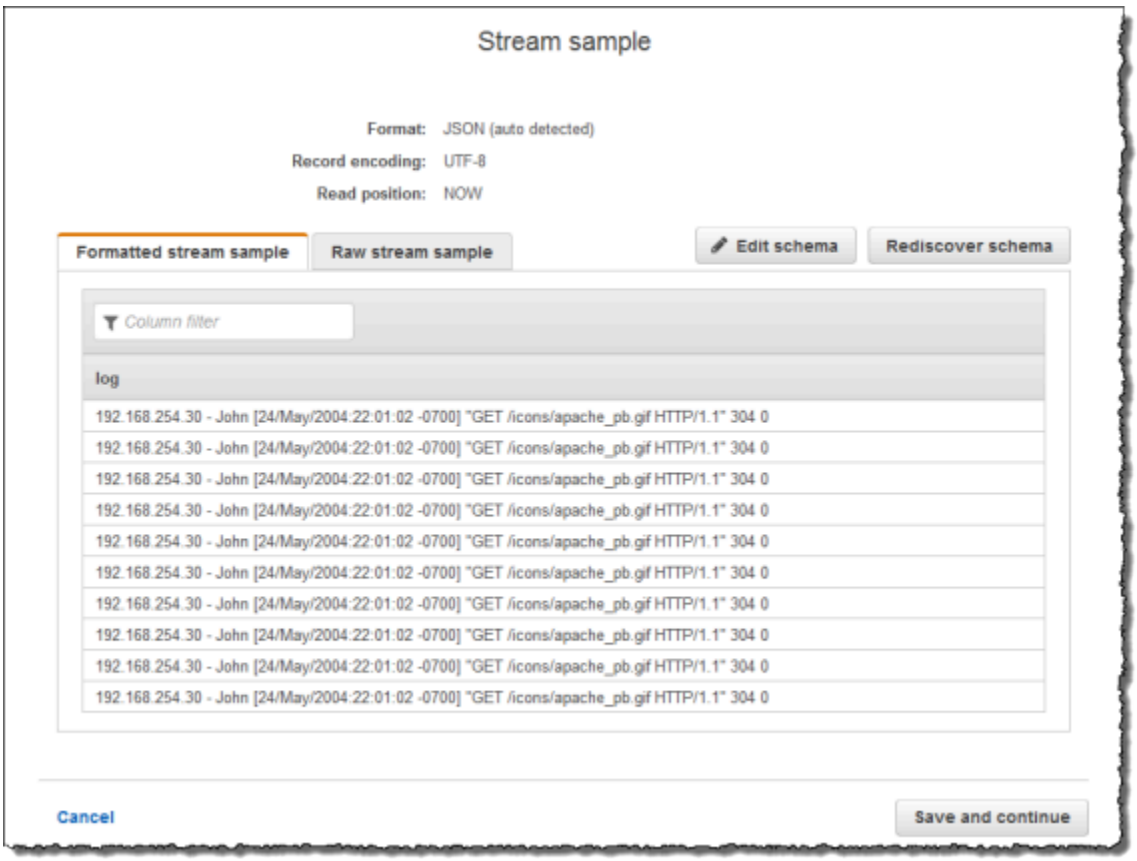
範例：剖析網頁日誌 (W3C_LOG_PARSE 函式)

此範例使用 W3C_LOG_PARSE 函數來轉換 Amazon Kinesis Data Analytics 中的串流。您可以使用 W3C_LOG_PARSE 來快速格式化 Apache 日誌。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的W3C_LOG_PARSE。

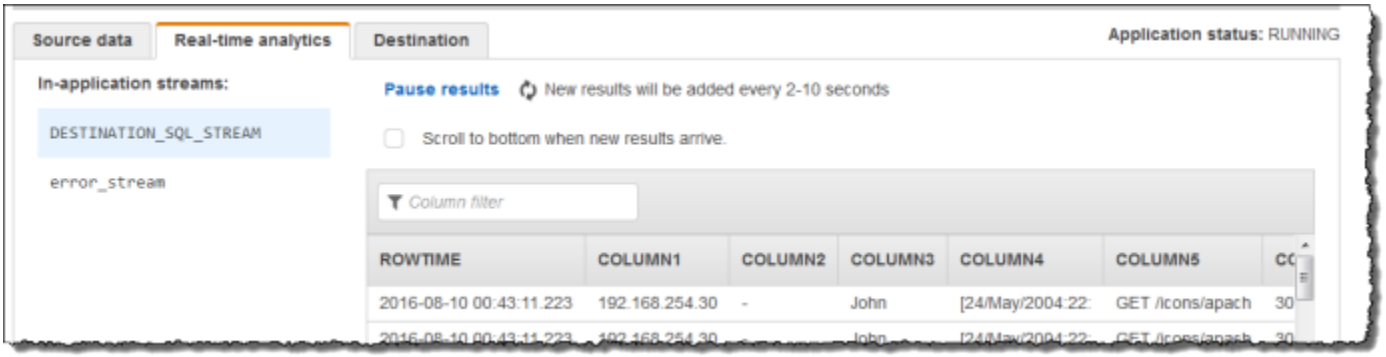
在此範例中，將日誌記錄寫入 Amazon Kinesis 資料串流。範例日誌如下所示：

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

接著，在主控台上建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並以一個資料欄 (日誌) 推斷應用程式內結構描述，如下所示：



然後，您可以將應用程式碼與 W3C_LOG_PARSE 函數搭配使用來剖析日誌，並建立另一個應用程式內串流，其中包含不同資料欄的各種日誌欄位，如下所示：



主題

- 步驟 1：建立 Kinesis 資料串流
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis 資料串流

建立 Amazon Kinesis 資料串流，並依照下列方式填入日誌記錄：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 執行下列 Python 程式碼，以填入範例日誌記錄。這個簡單的代碼會持續寫入相同的日誌記錄到串流。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

建立 Kinesis Data Analytics 應用程式，如下所示。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料。
4. 在連接至來源頁面，執行下列動作：
   a. 選擇您在上一節建立的串流。
   b. 選擇建立 IAM 角色 選項。
   c. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述只有一個資料欄。
   d. 選擇儲存並繼續。
5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：
   a. 請複製以下應用程式碼，然後貼到編輯器中。

```
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. 選擇儲存並執行 SQL。在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

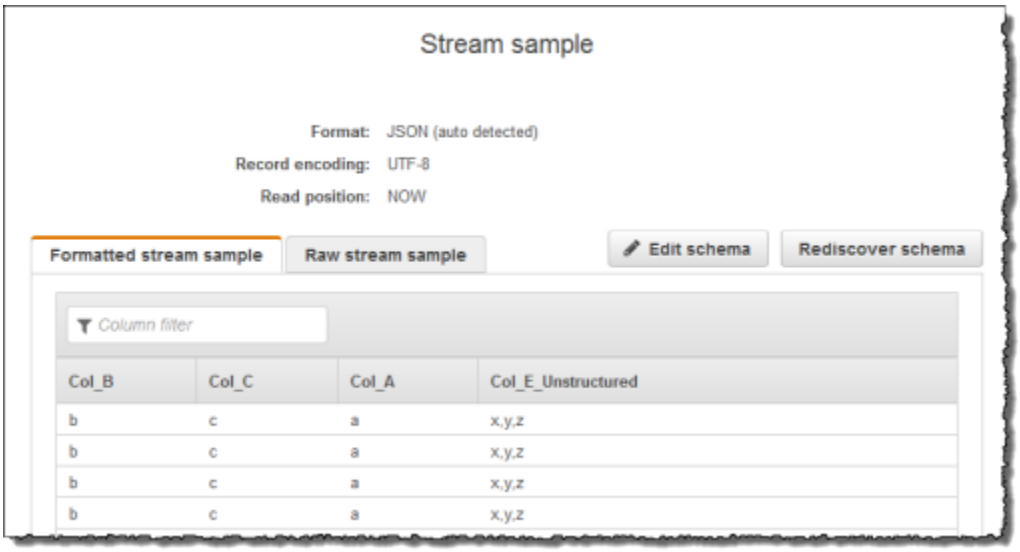
範例：將字串拆分為多個字段 (VARIABLE_COLUMN_LOG_PARSE 函數)

此範例使用 VARIABLE_COLUMN_LOG_PARSE 函數來操作 Kinesis Data Analytics 中的字串。VARIABLE_COLUMN_LOG_PARSE 將輸入字串分割，變成以分隔符號字元或字串分開的欄位。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的VARIABLE_COLUMN_LOG_PARSE。

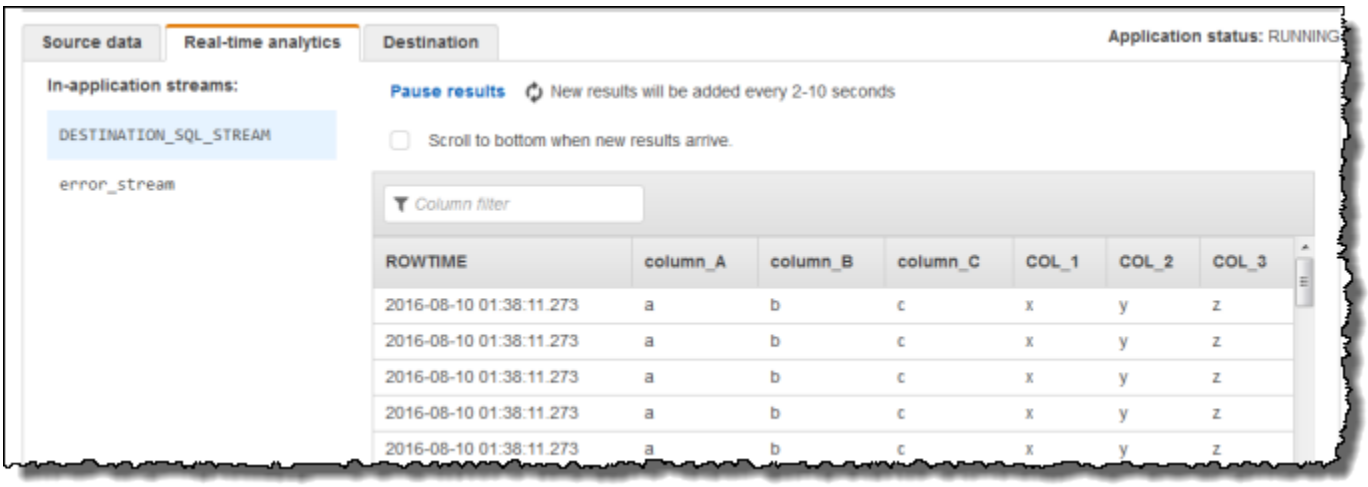
在此範例中，將半結構化記錄寫入 Amazon Kinesis 資料串流。範例記錄如下：

```
{ "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"}
```

接著，您可以在主控台上建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流當作串流來源。探索程序會讀取串流來源上的範例記錄，並以四個資料欄推斷應用程式內結構描述，如下所示：



然後，您可以將應用程式碼與 VARIABLE_COLUMN_LOG_PARSE 函數搭配使用來剖析逗號分隔的值，並將正規化的列插入另一個應用程式內串流，如下所示：



主題

- 步驟 1：建立 Kinesis 資料串流
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis 資料串流

建立 Amazon Kinesis 資料串流，並填入日誌紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 執行下列 Python 程式碼，以填入範例日誌記錄。這個簡單的代碼會持續寫入相同的日誌記錄到串流。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

建立 Kinesis Data Analytics 應用程式，如下所示。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料。
4. 在連接至來源頁面，執行下列動作：
   a. 選擇您在上一節建立的串流。
   b. 選擇建立 IAM 角色 選項。
   c. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。請注意，推斷的結構描述只有一個資料欄。
   d. 選擇儲存並繼續。
5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，撰寫應用程式碼，然後驗證結果：

a. 請複製以下應用程式碼，然後貼到編輯器中：

```
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. 選擇儲存並執行 SQL。在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

範例：轉換 DateTime 值

Amazon Kinesis Data Analytics 支援將資料欄轉換為時間戳記。舉例來說，除了 ROWTIME 欄之外，您可能想要使用自己的時間戳記做為 GROUP BY 子句的一部分，將其當作另一個以時間為基礎的視窗。Kinesis Data Analytics 提供操作和 SQL 函數，用於處理日期和時間欄位。

- 日期和時間運算子：您可以對日期、時間和間隔資料類型執行算術運算。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的日期、時間和間隔運算子。

- SQL 函數：包括以下項目。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的日期和時間函數。
  - EXTRACT( ) - 從日期、時間、時間戳記或間隔運算式中擷取一個欄位。
  - CURRENT_TIME - 傳回查詢執行的時間 (UTC)。

- CURRENT_DATE - 傳回查詢執行的日期 (UTC)。
- CURRENT_TIMESTAMP - 傳回查詢執行的時間戳記 (UTC)。
- LOCALTIME - 傳回由 Kinesis Data Analytics 運行環境定義，查詢執行時的目前時間 (UTC)。
- LOCALTIMESTAMP - 傳回由 Kinesis Data Analytics 運行環境定義的目前時間戳記 (UTC)。

- SQL 擴充功能：包括下列項目。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的日期和時間函數，以及日期時間轉換參考資料。
  - CURRENT_ROW_TIMESTAMP - 傳回串流中每一列的新時間戳記。
  - TSDIFF - 傳回兩個時間戳記的差值 (以毫秒為單位)。
  - CHAR_TO_DATE - 將字串轉換為日期。
  - CHAR_TO_TIME - 將字串轉換為時間。
  - CHAR_TO_TIMESTAMP - 將字串轉換為時間戳記。
  - DATE_TO_CHAR - 將日期轉換為字串。
  - TIME_TO_CHAR - 將時間轉換為字串。
  - TIMESTAMP_TO_CHAR - 將時間戳記轉換為字串。

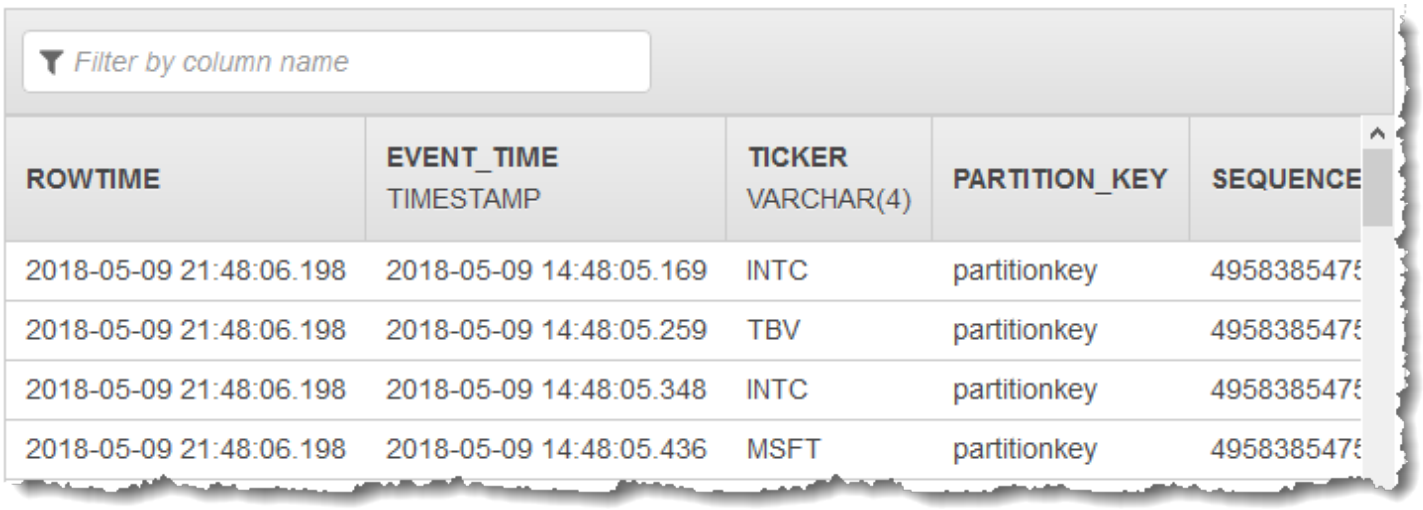
先前大多數 SQL 函數使用格式來轉換欄。此格式具靈活度。舉例來說，您可以指定格式 yyyy-MM-dd hh:mm:ss 將輸入字串 2009-09-16 03:15:24 轉換為時間戳記。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的文字至時間戳記 (Sys)。

範例：轉換日期

在此範例中，將下列記錄寫入 Amazon Kinesis 資料串流。

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

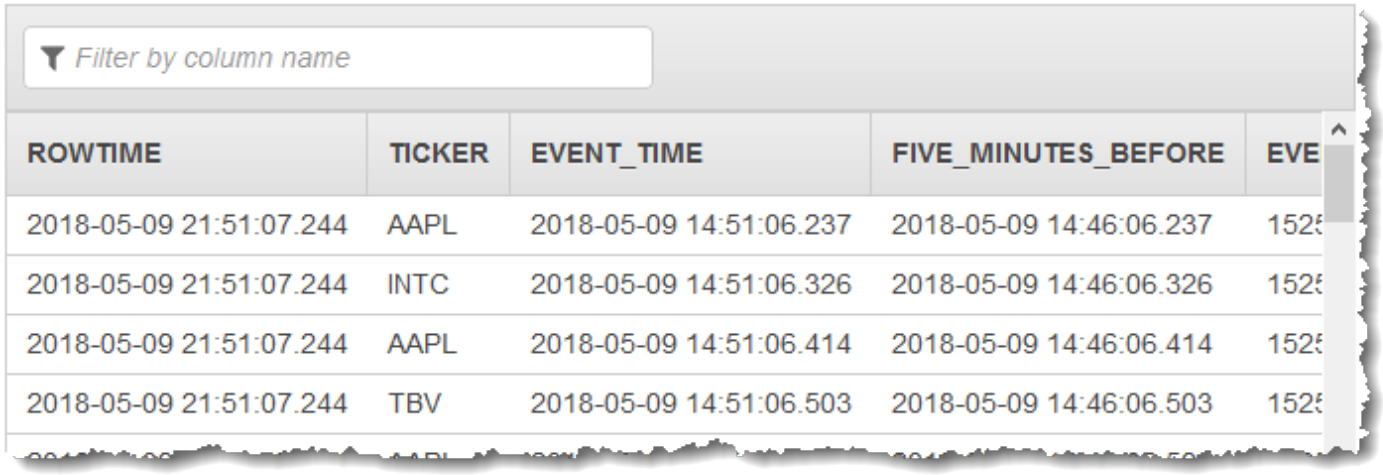
接著，您可以在主控台上建立 Kinesis Data Analytics 應用程式，並將 Kinesis 串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有兩個資料欄 (EVENT_TIME 和 TICKER) 的應用程式內結構描述，如下所示。



Filter by column name

ROWTIME	EVENT_TIME TIMESTAMP	TICKER VARCHAR(4)	PARTITION_KEY	SEQUENCE
2018-05-09 21:48:06.198	2018-05-09 14:48:05.169	INTC	partitionkey	4958385475
2018-05-09 21:48:06.198	2018-05-09 14:48:05.259	TBV	partitionkey	4958385475
2018-05-09 21:48:06.198	2018-05-09 14:48:05.348	INTC	partitionkey	4958385475
2018-05-09 21:48:06.198	2018-05-09 14:48:05.436	MSFT	partitionkey	4958385475

然後，將應用程式碼與 SQL 函數搭配使用，以各種方式轉換 EVENT_TIME 時間戳記欄位。接著將產生的資料插入另一個應用程式內串流，如下列螢幕擷取畫面所示：



Filter by column name

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

步驟 1：建立 Kinesis 資料串流

建立 Amazon Kinesis 資料串流，並將事件時間和股票記錄填入其中，如下所示：

1. 登入 AWS Management Console 並開啟運動主控台，網址為 https://console.aws.amazon.com/kinesis。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。
4. 執行下列 Python 程式碼，以使用範例資料填入串流。這個簡單的程式碼，會持續將帶有隨機股票代號和當前時間戳記的記錄寫入串流。

```
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"))
```

步驟 2：建立 Amazon Kinesis Data Analytics 應用程式

建立應用程式，如下所示：

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料來連接至來源。
4. 在連接至來源頁面，執行下列動作：

a. 選擇您在上一節建立的串流。

b. 選擇建立 IAM 角色。

c. 選擇探索結構描述。等待主控台顯示推斷的結構描述，以及用來推斷建立的應用程式內串流結構描述之範例記錄。推斷的結構描述有兩個資料欄。

d. 選擇編輯結構描述。將 EVENT_TIME 欄的欄類型變更為 TIMESTAMP。

e. 選擇 儲存結構描述並更新串流範例。主控台儲存結構描述後，選擇結束。

f. 選擇儲存並繼續。

5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。

6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

a. 請複製以下應用程式碼，然後貼到編輯器中。

```
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. 選擇儲存並執行 SQL。在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

範例：轉換多個資料類型

擷取、轉換和載入 (ETL) 應用程式的常見需求，是在串流來源上處理多個記錄類型。您可以建立 Kinesis Data Analytics 應用程式來處理這些類型的串流來源。程序如下：

1. 首先，將串流來源對應到應用程式內輸入串流，類似於所有其他 Kinesis Data Analytics 應用程式。
2. 然後，在應用程式碼中撰寫 SQL 陳述式，從應用程式內輸入串流擷取特定類型的資料列。接著將它們插入個別的應用程式內串流。(您可以在應用程式碼中建立其他應用程式內串流。)

在本練習中，您有一個接收兩種類型 (`Order` 和 `Trade`) 記錄的串流來源。這些是股票訂單和相應的交易。針對每個訂單，可能有零個或多個交易。每種類型的範例記錄如下所示：

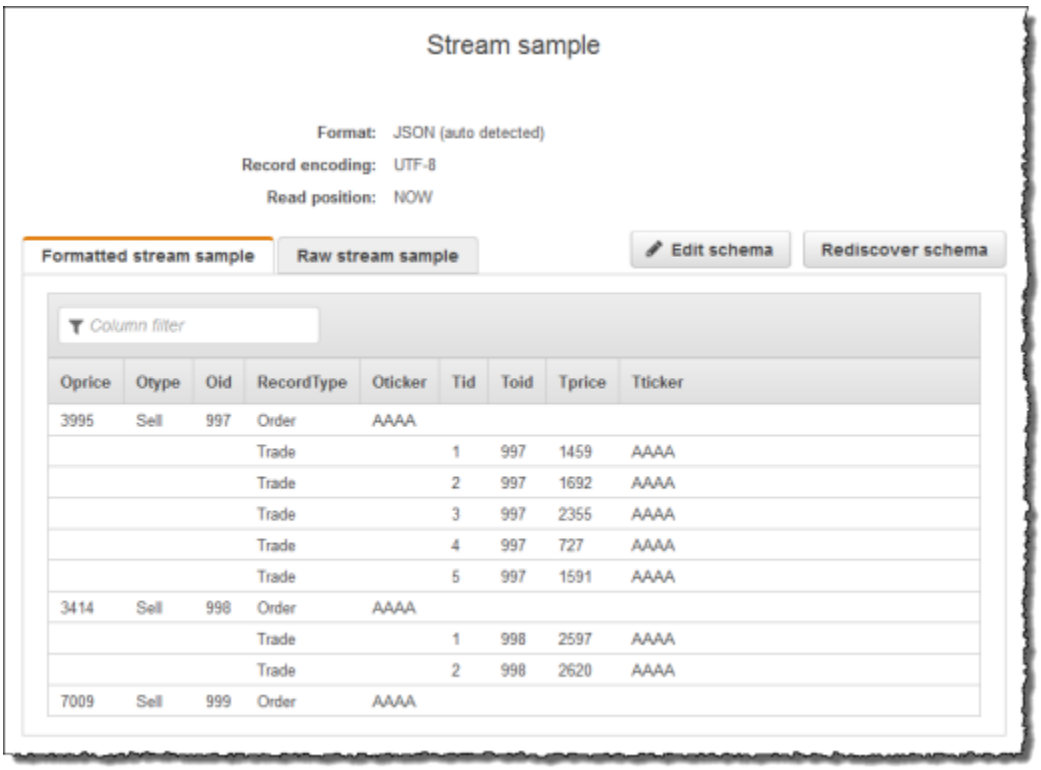
訂單記錄

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

交易記錄

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

當您使用建立應用程式時 AWS Management Console，主控台會針對建立的應用程式內輸入串流顯示下列推斷結構描述。根據預設，主控台會命名此應用程式內串流 `SOURCE_SQL_STREAM_001`。



儲存組態時，Amazon Kinesis Data Analytics 會持續從串流來源讀取資料，並在應用程式內串流中插入資料列。您現在可以對應用程式內串流中的資料執行分析。

在此範例的應用程式碼中，先建立兩個額外的應用程式內串流，Order_Stream 和 Trade_Stream。然後，您可以根據記錄類型從 SOURCE_SQL_STREAM_001 串流中篩選列，並使用幫浦將它們插入新建立的串流中。如需此編碼模式的相關資訊，請參閱 應用程式碼。

1. 將訂單和交易列過濾到個別的應用程式內串流：

   a. 篩選 SOURCE_SQL_STREAM_001 中的訂單記錄，並將訂單儲存在 Order_Stream 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';
```

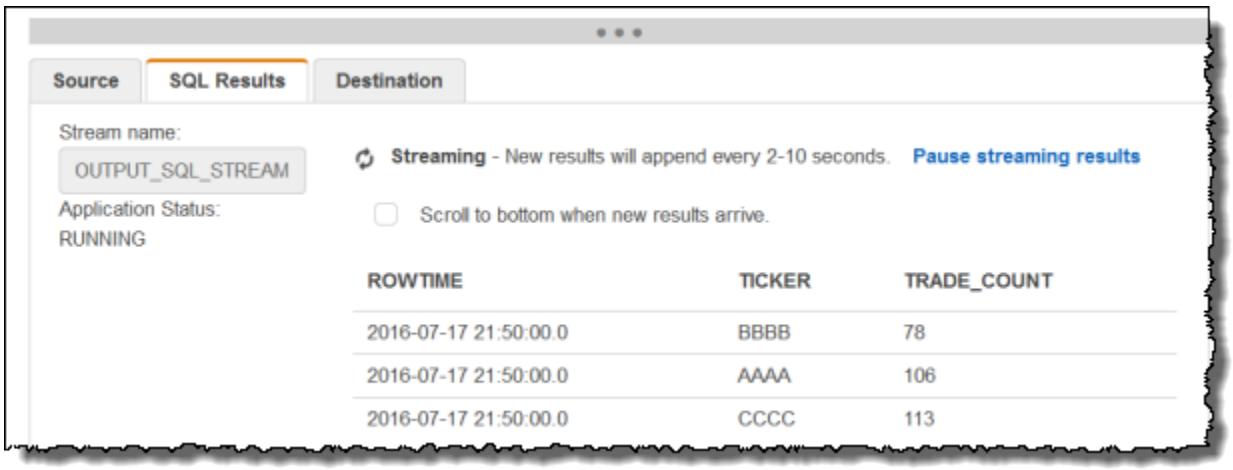
b. 篩選 SOURCE_SQL_STREAM_001 中的交易記錄，並將訂單儲存在 Trade_Stream 中。

```
--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. 現在，您可以對這些串流執行其他分析。在此範例，於一分鐘的翻轉視窗中計算股票代碼的交易數量，並將結果保存到另一個 DESTINATION_SQL_STREAM 串流中。

```
--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);
```

您會看到下列結果。



主題

- 步驟 1：準備資料
- 步驟 2：建立應用程式

後續步驟

步驟 1：準備資料

步驟 1：準備資料

在本節中，建立 Kinesis 資料串流，然後在串流中填入訂單和交易記錄。這是您在下一個步驟中建立之應用程式的串流來源。

主題

- 步驟 1.1：建立串流來源
- 步驟 1.2：填入串流來源

步驟 1.1：建立串流來源

您可以使用主控台或 AWS CLI建立 Kinesis 資料串流。此範例假設 `OrdersAndTradesStream` 為串流名稱。

- 使用主控台 — 登入 AWS Management Console 並開啟 Kinesis 主控台，網址為 https://console.aws.amazon.com/kinesis。選擇資料串流，然後建立具有一個碎片的串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。

- 使用 AWS CLI — 使用下列 Kinesis `create-stream` AWS CLI 指令建立串流：

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

步驟 1.2：填入串流來源

執行下列 Python 指令碼，將範例記錄填入 `OrdersAndTradesStream`。如果您使用不同的名稱建立串流，請更新相應的 Python 程式碼。

1. 安裝 Python 與 `pip`。

   如需安裝 Python 的相關資訊，請參閱 Python 網站。

   您可以使用 Pip 安裝相依性。如需安裝 Pip 的詳細資訊，請參閱 pip 網站的安裝。

2. 執行以下 Python 程式碼。程式碼中的 `put-record` 命令會將 JSON 記錄寫入串流。

```
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"))
```

後續步驟

[步驟 2：建立應用程式]

步驟 2：建立應用程式

在本節建立 Amazon Kinesis Data Analytics 應用程式。然後新增輸入組態，將上一節建立的串流來源對應至應用程式內輸入串流，藉此更新應用程式。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式 。此範例使用應用程式名稱 **ProcessMultipleRecordTypes**。
3. 在應用程式詳細資料頁面上，選擇連接串流資料來連接至來源。
4. 在連接至來源頁面，執行下列動作：
   a. 選擇您在 步驟 1：準備資料 中建立的串流。
   b. 選擇建立 IAM 角色。
   c. 等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。
   d. 選擇儲存並繼續。
5. 在應用程式中樞，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果：
   a. 請複製以下應用程式碼，然後貼到編輯器中。

```
--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. 選擇儲存並執行 SQL。選擇即時分析標籤，查看應用程式建立的所有應用程式內串流，並驗證資料。

後續步驟

您可以設定應用程式輸出，將結果保留至外部目的地，例如另一個 Kinesis 串流或 Firehose 資料傳遞串流。

範例：視窗與彙總

本節提供使用視窗語彙總查詢的 Amazon Kinesis Data Analytics 應用程式範例。(如需詳細資訊，請參閱 窗口化查詢。) 每個範例皆包含逐步指示與範例程式碼，協助您建立 Kinesis Data Analytics 應用程式並測試結果。

主題

- 範例：交錯視窗
- 範例：使用列時間的輪轉窗口

- 範例：使用事件時間戳記的輪轉窗口
- 範例：擷取最常發生的值 (TOP_K_ITEMS_TUMBLING)
- 範例：從查詢彙總部分結果

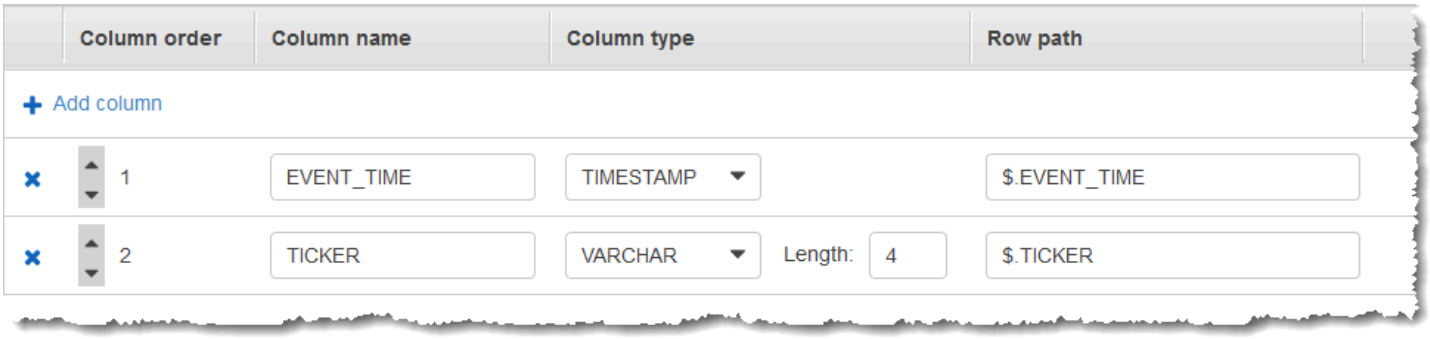
範例：交錯視窗

當視窗化查詢為每個不同的分割區索引鍵處理個別視窗時，從具有相符索引鍵的資料到達時開始，即為交錯視窗。如需詳細資訊，請參閱 交錯窗口。這個 Amazon Kinesis Data Analytics 範例使用 EVENT_TIME 和 TICKER 欄來建立交錯視窗。來源串流包含六個記錄的群組，其中有相同的 EVENT_TIME 和 TICKER 值，這些資料會在一分鐘內到達，但不一定具有相同的分鐘值 (例如 18:41:xx)。

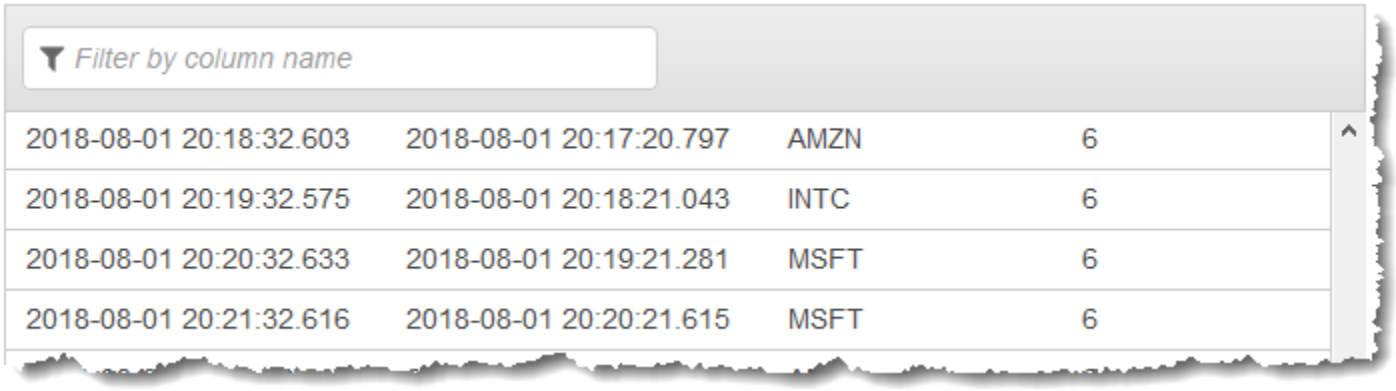
在此範例中，將下列記錄於指定時間寫入 Amazon Kinesis 資料串流。指令碼不會將時間寫入串流，但應用程式擷取記錄的時間會寫入 ROWTIME 欄位：

```
{"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
...
```

接著，在 AWS Management Console 建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有兩個資料欄 (EVENT_TIME 和 TICKER) 的應用程式內結構描述，如下所示。



您可以將應用程式碼與 COUNT 函數搭配使用，以建立資料的視窗化彙總。接著將產生的資料插入另一個應用程式內串流，如下列螢幕擷取畫面所示：



在下列程序中，您會建立 Kinesis Data Analytics 應用程式，根據 EVENT_TIME 和 TICKER，在交錯視窗彙總輸入串流中的值。

主題

- 步驟 1：建立 Kinesis Data Stream
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis Data Stream

建立 Amazon Kinesis 資料串流，並填入紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。

4. 若要在生產環境中將記錄寫入 Kinesis 資料串流，建議您使用 Kinesis Producer Library 或 Kinesis Data Streams API。為了簡單起見，這個例子使用下面的 Python 指令碼來生成記錄。執行程式碼以填入範例股票代號記錄。這個簡單的程式碼會在一分鐘的時間內，連續將具有相同隨機 EVENT_TIME 和股票代號的六筆記錄寫入串流。讓指令碼持續執行，以便在稍後的步驟產生應用程式結構描述。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

建立 Kinesis Data Analytics 應用程式，如下所示。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料來連接至來源。
4. 在連接至來源頁面，執行下列動作：

   a. 選擇您在上一節建立的串流。

   b. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述有兩個資料欄。

   c. 選擇編輯結構描述。將 EVENT_TIME 欄的欄類型變更為 TIMESTAMP。

   d. 選擇 儲存結構描述並更新串流範例。主控台儲存結構描述後，選擇結束。

   e. 選擇儲存並繼續。
5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

   a. 請複製以下應用程式碼，然後貼到編輯器中。

```
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. 選擇儲存並執行 SQL。

在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

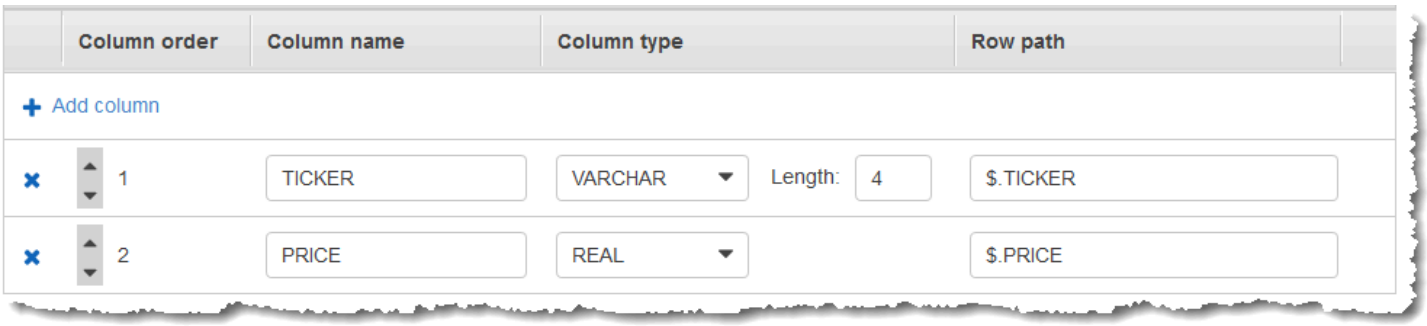
範例：使用列時間的輪轉窗口

當窗口查詢以非重疊的方式處理每個窗口時，即稱作輪轉窗口。如需詳細資訊，請參閱 輪轉窗口（使用 GROUP BY 彙總）。此 Amazon Kinesis Data Analytics 範例使用 ROWTIME 資料欄來建立輪轉窗口。該 ROWTIME 欄示應用程式讀取記錄的時間。

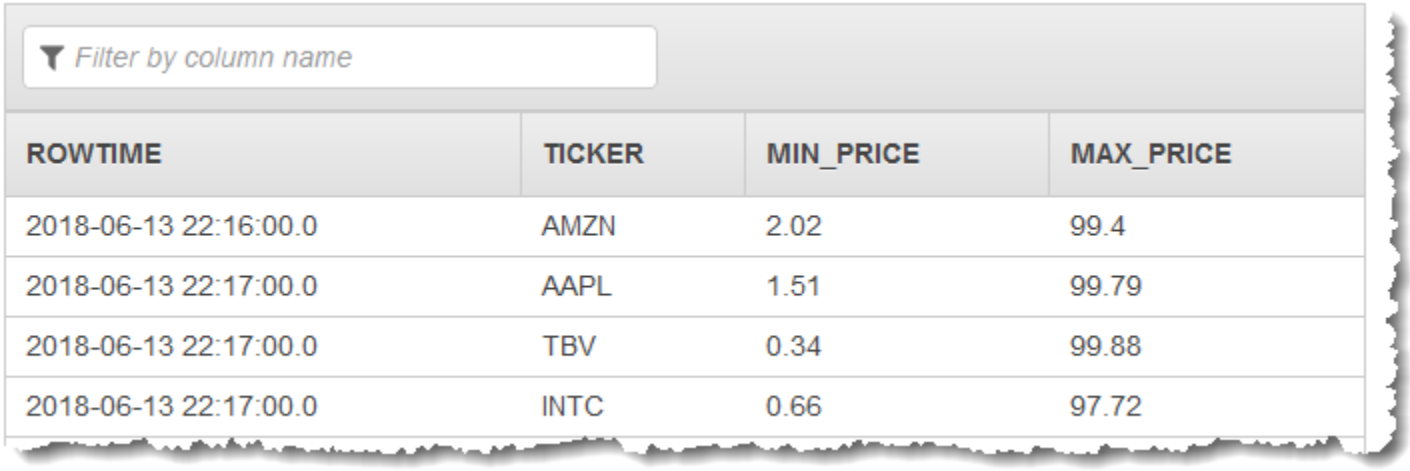
在此範例中，將下列記錄寫入 Amazon Kinesis 資料串流。

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

接著，在 AWS Management Console 建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有兩個資料欄 (TICKER 和 PRICE) 的應用程式內結構描述，如下所示。



您可以將應用程式碼與 MIN 和 MAX 函數搭配使用，以建立資料的窗口化彙總。接著將產生的資料插入另一個應用程式內串流，如下列螢幕擷取畫面所示：



Filter by column name

ROWTIME	TICKER	MIN_PRICE	MAX_PRICE
2018-06-13 22:16:00.0	AMZN	2.02	99.4
2018-06-13 22:17:00.0	AAPL	1.51	99.79
2018-06-13 22:17:00.0	TBV	0.34	99.88
2018-06-13 22:17:00.0	INTC	0.66	97.72

在下列程序中，建立 Kinesis Data Analytics 應用程式，以根據 ROWTIME 在輪轉窗口彙總輸入串流中的值。

主題

- 步驟 1：建立 Kinesis Data Stream
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis Data Stream

建立 Amazon Kinesis 資料串流，並填入紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 若要在生產環境中將記錄寫入 Kinesis 資料串流，建議您使用 Kinesis Client Library或 Kinesis Data Streams API。為了簡單起見，這個例子使用下面的 Python 指令碼來生成記錄。執行程式碼以填入範例股票代號記錄。這個簡單的程式碼會持續將隨機股票代號記錄寫入串流。讓指令碼持續執行，以便在稍後的步驟產生應用程式結構描述。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

建立 Kinesis Data Analytics 應用程式，如下所示。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料來連接至來源。
4. 在連接至來源頁面，執行下列動作：

    a. 選擇您在上一節建立的串流。

    b. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述有兩個資料欄。

    c. 選擇 儲存結構描述並更新串流範例。主控台儲存結構描述後，選擇結束。

d. 選擇儲存並繼續。

5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。

6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

a. 請複製以下應用程式碼，然後貼到編輯器中。

```
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. 選擇儲存並執行 SQL。

在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

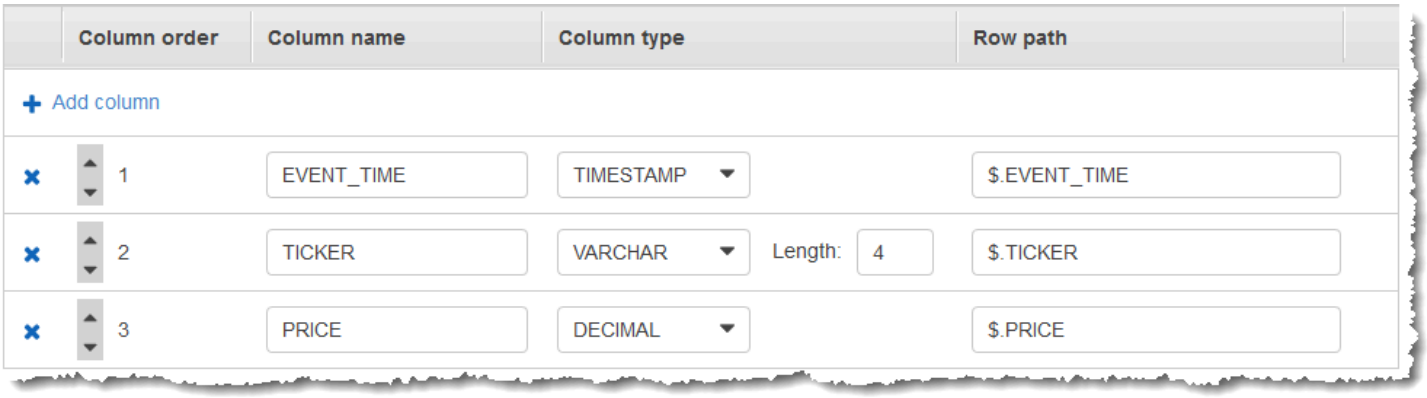
範例：使用事件時間戳記的輪轉窗口

當窗口查詢以非重疊的方式處理每個窗口時，即稱作輪轉窗口。如需詳細資訊，請參閱 輪轉窗口（使用 GROUP BY 彙總）。這個 Amazon Kinesis Data Analytics 範例示範使用事件時間戳記 (即使用者建立，包含在串流資料中的時間戳記) 的輪轉窗口。範例中使用這種方法，而非僅使用 ROWTIME，即 Kinesis Data Analytics 在應用程式收到記錄時所建立的時間戳記。如果您想要根據事件發生的時間 (而非應用程式收到事件的時間) 建立彙總，可在串流資料中使用事件時間戳記。在此範例中，ROWTIME 值會每分鐘觸發彙總，而記錄會依照 ROWTIME 和包含的事件時間兩者進行彙總。

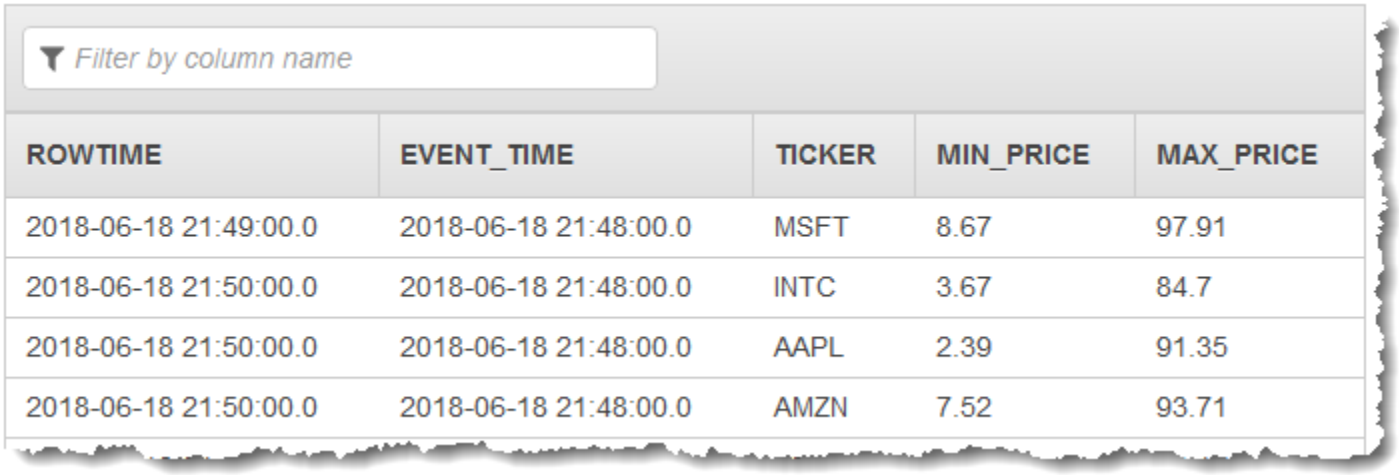
在此範例中，將下列記錄寫入 Amazon Kinesis 串流：此 EVENT_TIME 值設定為過去 5 秒，以模擬處理和傳輸延遲，這些延遲可能會造成事件發生與記錄導入 Kinesis Data Analytics 的時間差。

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

接著，在 AWS Management Console 建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並推斷含有三個資料欄 (EVENT_TIME、TICKER 和PRICE) 的應用程式內結構描述，如下所示。



您可以將應用程式碼與 MIN 和 MAX 函數搭配使用，以建立資料的窗口化彙總。接著將產生的資料插入另一個應用程式內串流，如下列螢幕擷取畫面所示：



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

在下列程序中，建立 Kinesis Data Analytics 應用程式，該應用程式會根據事件時間在輪轉窗口中彙總輸入串流的值。

主題

- 步驟 1：建立 Kinesis Data Stream
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis Data Stream

建立 Amazon Kinesis 資料串流，並填入紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 若要在生產環境中將記錄寫入 Kinesis 資料串流，建議您使用 Kinesis Client Library或 Kinesis Data Streams API。為了簡單起見，這個例子使用下面的 Python 指令碼來生成記錄。執行程式碼以填入範例股票代號記錄。這個簡單的程式碼會持續將隨機股票代號記錄寫入串流。讓指令碼持續執行，以便在稍後的步驟產生應用程式結構描述。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

建立 Kinesis Data Analytics 應用程式，如下所示。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料來連接至來源。
4. 在連接至來源頁面，執行下列動作：

   a. 選擇您在上一節建立的串流。

   b. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述有三個資料欄。

   c. 選擇編輯結構描述。將 EVENT_TIME 欄的欄類型變更為 TIMESTAMP。

   d. 選擇 儲存結構描述並更新串流範例。主控台儲存結構描述後，選擇結束。

   e. 選擇儲存並繼續。
5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

   a. 請複製以下應用程式碼，然後貼到編輯器中。

```
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. 選擇儲存並執行 SQL。

在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

範例：擷取最常發生的值 (TOP_K_ITEMS_TUMBLING)

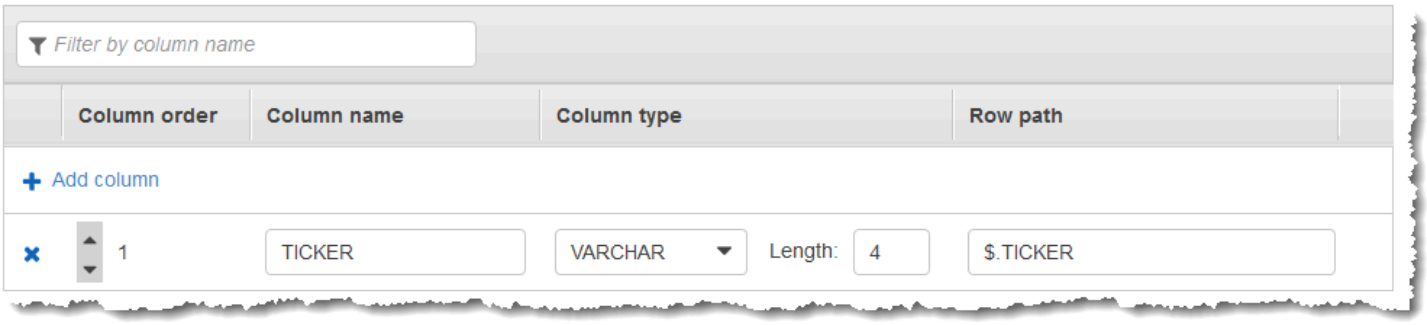
這個 Amazon Kinesis Data Analytics 範例示範如何使用 TOP_K_ITEMS_TUMBLING 函數擷取輪轉窗口中最常出現的值。如需詳細資訊，請參閱《Amazon Managed Service for Apache Flink SQL 參考資料》中的 TOP_K_ITEMS_TUMBLING 函數。

此 TOP_K_ITEMS_TUMBLING 功能在彙總超過數萬或數十萬個金鑰，且想要減少資源使用量時非常有用。該函數產生與 GROUP BY 和 ORDER BY 子句匯總相同的結果。

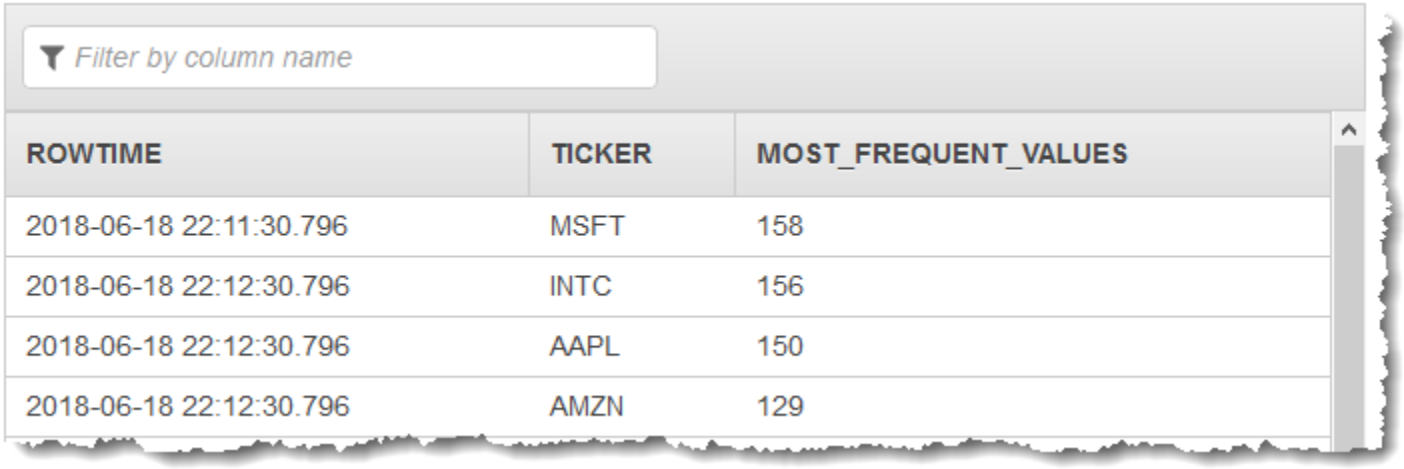
在此範例中，將下列記錄寫入 Amazon Kinesis 資料串流：

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

接著，在 AWS Management Console 建立 Kinesis Data Analytics 應用程式，並將 Kinesis 資料串流做為串流來源。探索程序會讀取串流來源上的範例記錄，並以一個資料欄 (TICKER) 推斷應用程式內結構描述，如下所示：



您可以將應用程式碼與 TOP_K_VALUES_TUMBLING 函數搭配使用，以建立資料的視窗化彙總。接著將產生的資料插入另一個應用程式內串流，如下列螢幕擷取畫面所示：



在下列程序中，建立 Kinesis Data Analytics 應用程式，以擷取輸入串流中最常出現的值。

主題

- 步驟 1：建立 Kinesis Data Stream
- 步驟 2：建立 Kinesis Data Analytics 應用程式

步驟 1：建立 Kinesis Data Stream

建立 Amazon Kinesis 資料串流，並填入紀錄，如下所示：

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中選擇資料串流。
3. 選擇建立 Kinesis 串流，然後建立內含一個碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
4. 若要在生產環境中將記錄寫入 Kinesis 資料串流，建議您使用 Kinesis Client Library或 Kinesis Data Streams API。為了簡單起見，這個例子使用下面的 Python 指令碼來生成記錄。執行程式碼以填入範例股票代號記錄。這個簡單的程式碼會持續將隨機股票代號記錄寫入串流。讓指令碼保持執行，以便在稍後的步驟中產生應用程式結構描述。

```
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"))
```

步驟 2：建立 Kinesis Data Analytics 應用程式

建立 Kinesis Data Analytics 應用程式，如下所示。

1. 前往 https://console.aws.amazon.com/kinesisanalytics 開啟 Managed Service for Apache Flink 主控台。
2. 選擇建立應用程式，輸入應用程式名稱，然後選擇建立應用程式。
3. 在應用程式詳細資料頁面上，選擇連接串流資料來連接至來源。
4. 在連接至來源頁面，執行下列動作：

   a. 選擇您在上一節建立的串流。

   b. 選擇探索結構描述。等待主控台顯示推斷的結構描述和範例記錄，這些記錄可用來推斷應用程式內串流所建立的結構描述。推斷的結構描述有一個資料欄。

   c. 選擇 儲存結構描述並更新串流範例。主控台儲存結構描述後，選擇結束。

   d. 選擇儲存並繼續。

5. 在應用程式詳細資訊頁面上，選擇至 SQL 編輯器。若要啟動應用程式，請在出現的對話方塊中選擇是，啟動應用程式。
6. 在 SQL 編輯器中，編寫應用程式碼並驗證結果，如下所示：

   a. 請複製以下應用程式碼，然後在編輯器中貼上。

```
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. 選擇儲存並執行 SQL。

   在即時分析標籤上，您可以查看應用程式建立的所有應用程式內串流，並驗證資料。

範例：從查詢彙總部分結果

如 Amazon Kinesis 資料串流中有事件時間與擷取時間不完全相符的記錄，輪轉窗口中一些結果抵達，但不一定在窗口中發生。在這種情況下，輪轉窗口只會包含您想要的部分結果集。您可以使用以下幾種方法來修正此問題：

- 僅使用輪轉窗口，並用 upserts 透過資料庫或資料倉儲彙總部分後處理結果。這種方法在處理應用程式時非常有效。它會無限期地處理彙總運算子 (sum、min、max 等等) 的延遲資料。這種方法的缺點是，您必須在資料庫層開發和維護其他應用程式邏輯。
- 使用輪轉和滑動窗口，可提早產生部分結果，但在滑動窗口期間也會繼續產生完整的結果。這種方法使用複寫，而非 upsert 處理延遲資料，這樣就不需要在資料庫層添加額外的應用程式邏輯。此方法的缺點是它使用了更多的 Kinesis 處理單元 (KPU)，且依舊產生兩個結果，這可能不適用於某些使用案例。

如需輪轉與滑動窗口的詳細資訊，請參閱 窗口化查詢。

在下列程序中，輪轉窗口彙總會產生兩個部分結果 (傳送至 CALC_COUNT_SQL_STREAM 應用程式內串流)，必須結合才能產生最終結果。接著，應用程式會產生第二個彙總 (傳送至 DESTINATION_SQL_STREAM 應用程式內串流)，結合兩個部分結果。

若要建立使用事件時間彙總部分結果的應用程式

1. 前往 https://console.aws.amazon.com/kinesis 登入 AWS Management Console 並開啟 Kinesis 主控台。
2. 在導覽窗格中，選擇資料分析。建立 Kinesis Data Analytics 應用程式，如 Amazon Kinesis Data Analytics for SQL 應用程式入門 教學課程所述。
3. 在 SQL 編輯器中，以下列項目取代應用程式碼：

```
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);
```

應用程式碼中的 SELECT 陳述式會篩選 SOURCE_SQL_STREAM_001 中的資料欄，找出變更大於 1% 的股票價格，並使用幫浦將這些資料列插入另一個應用程式內串流 CHANGE_STREAM。

4. 選擇儲存並執行 SQL。

第一個幫浦會將串流輸出至 CALC_COUNT_SQL_STREAM ，類似下列內容。請注意，結果集不完整：

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

然後，第二個幫浦輸出一個包含完整結果集的串流至 DESTINATION_SQL_STREAM：

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

範例：聯結

本節提供使用聯結查詢的 Kinesis Data Analytics 應用程式範例。每個範例皆包含逐步指示，協助您建立 Kinesis Data Analytics 應用程式並測試結果。

主題

範例：將參考資料新增至 Kinesis Data Analytics 應用程式

在本練習中，將參考資料新增至現有的 Kinesis Data Analytics 應用程式。如需參考資料的相關資訊，請參閱下列主題：

- Amazon Kinesis Data Analytics for SQL 應用程式：運作方式
- 設定應用程式輸入

在本練習中，將參考資料新增至您在 Kinesis Data Analytics 入門練習中建立的應用程式。參考資料提供每個股票代號的公司名稱，例如：

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

首先，完成入門練習中的步驟，以建立入門應用程式。接著請依照這些步驟設定參考資料，並將其新增至您的應用程式：

1. 準備資料
   - 將先前的參考資料做為物件存放在 Amazon Simple Storage Service (Amazon S3)。
   - 建立 Kinesis Data Analytics 可擔任的 IAM 角色，以代您讀取 Amazon S3 物件。
2. 將參考資料來源新增到應用程式。

   Kinesis Data Analytics 會讀取 Amazon S3 物件，並建立應用程式內參考資料表，讓您在應用程式碼中查詢。
3. 測試代碼。

在應用程式碼中，撰寫聯結查詢，將應用程式內串流與應用程式內參考資料表聯結，以取得每個股票代號的公司名稱。

主題

步驟 1：準備

在本節中，將範例參考資料做為物件存放在 Amazon S3 儲存貯體中。同時建立 Kinesis Data Analytics 可擔任的 IAM 角色，以代您讀取物件。

將參考資料存放為 Amazon S3 物件

在本節中，將範例參考資料做為 Amazon S3 物件儲存。

1. 開啟文字編輯器，加入以下資料，並將檔案儲存為 `TickerReference.csv`。

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

2. 將 `TickerReference.csv` 檔案上傳至 S3 儲存貯體。如需指示說明，請參閱 Amazon Simple Storage Service 使用者指南中的上傳物件至 Amazon S3。

建立 IAM 角色

接下來，建立 Kinesis Data Analytics 可擔任的 IAM 角色，並讀取 Amazon S3 物件。

1. 在 AWS Identity and Access Management (IAM) 中，建立名為 **`KinesisAnalytics-ReadS3Object`** 的 IAM 角色。若要建立角色，請按照 IAM 使用者指南中建立 Amazon 服務的角色 (AWS Management Console) 所述指示操作。

   在 IAM 主控台，指定下列項目：

- 在選取角色類型中，選擇 AWS Lambda。建立角色後，請變更信任政策以允許 Kinesis Data Analytics (而非 AWS Lambda) 擔任該角色。
- 請勿在附加政策頁面附加任何政策。

2. 更新 IAM 角色政策：

   a. 在 IAM 主控台中，選擇您剛建立的角色。

   b. 在信任關係標籤上，更新信任政策以授與 Kinesis Data Analytics 權限來擔任該角色。信任政策如下所示：

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

   c. 在許可標籤上，附加一個名為 AmazonS3ReadOnlyAccess 的 Amazon 受管政策。此舉會授予該角色讀取 Amazon S3 物件的許可。此政策如下所示。

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

步驟 2：將參考資料來源新增至應用程式組態

在此步驟中，將參考資料來源新增至您的應用程式組態。首先，您需要下列資訊：

- 您的 S3 儲存貯體名稱和物件金鑰名稱
- IAM 角色 Amazon Resource Name (ARN)。

1. 在應用程式的主頁面中，選擇連接參考資料。
2. 在連接參考資料來源頁面中，選擇包含參考資料物件的 Amazon S3 儲存貯體，然後輸入物件的金鑰名稱。
3. 輸入 **CompanyName** 作為應用程式內參考表名稱。
4. 在存取所選資源區段中，選擇從 Kinesis Analytics 可以擔任的 IAM 角色中選擇，然後選擇您在上一節建立的 KinesisAnalytics-ReadS3Object IAM 角色。
5. 選擇探索結構描述。控制台檢測到參考資料中的兩欄。
6. 選擇儲存與關閉。

步驟 3：測試：查詢應用程式內參考資料表

您現在可查詢應用程式內參考資料表 CompanyName。您可以使用參考資訊，將股票價格資料與參考資料表聯結在一起，以豐富您的應用程式。結果會顯示公司名稱。

1. 以下列代碼取代您的應用程式碼。查詢會將應用程式內輸入串流與應用程式內參考資料表聯結。應用程式碼會將結果寫入另一個應用程式內串流 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. 確認應用程式輸出是否顯示在 SQLResults 標籤中。請確定某些資料欄顯示公司名稱 (範例參考資料並未包含所有公司名稱)。

範例：機器學習

本節提供使用機器學習查詢的 Amazon Kinesis Data Analytics 應用程式範例。機器學習查詢會對資料執行複雜的分析，依賴串流中資料的歷史記錄來尋找不尋常的模式。每個範例皆包含逐步指示，協助您建立 Kinesis Data Analytics 應用程式並測試結果。

主題

範例：偵測串流上的資料異常 (RANDOM_CUT_FOREST 函數)

Amazon Kinesis Data Analytics 提供函數 (RANDOM_CUT_FOREST)，可根據數值欄中的值為每筆記錄指派異常分數。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的 RANDOM_CUT_FOREST 函數。

在本練習中，撰寫應用程式碼，將異常分數指派給應用程式串流來源上的記錄。若要設定應用程式，請執行下列動作：

1. 設定串流來源：設定 Kinesis 資料串流並寫入範例 heartRate 資料，如下所示：

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

   此程序會提供 Python 指令碼供您填入串流。這些 heartRate 值是隨機產生的，99% 記錄的 heartRate 值介於 60 到 100 之間，而只有 1% 的 heartRate 值介於 150 和 200 之間。因此，heartRate 值介於 150 和 200 之間的記錄為異常。
2. 設定輸入：使用主控台可建立 Kinesis Data Analytics 應用程式，並透過將串流來源對應至應用程式內串流 (SOURCE_SQL_STREAM_001)，以設定應用程式輸入。當應用程式啟動時，Kinesis Data Analytics 會持續讀取串流來源，並將記錄插入應用程式內串流。
3. 指定應用程式碼：此範例使用下列應用程式碼：

   ```
   --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;
```

程式碼會讀取 SOURCE_SQL_STREAM_001 中的資料列、指派異常分數，然後將產生的資料欄寫入另一個應用程式內串流 (TEMP_STREAM)。然後，應用程式碼會排序 TEMP_STREAM 中的記錄，並將結果儲存到另一個應用程式內串流 (DESTINATION_SQL_STREAM)。您可以使用幫浦將資料列插入應用程式內串流。如需更多詳細資訊，請參閱 應用程式內串流與幫浦。

4. 配置輸出：設定將應用程式輸出，將 DESTINATION_SQL_STREAM 中的資料保存在外部目的地，即另一個 Kinesis 資料串流。檢閱指派給每筆記錄的異常分數，並判斷哪些分數表示應用程式外部的異常 (且需要收到提醒)。您可以使用 AWS Lambda 函數來處理這些異常分數並設定提醒。

本練習會使用美國東部 (維吉尼亞北部) (us-east-1) 來建立這些串流，以及您的應用程式。如果您使用任何其他區域，則須更新相應程式碼。

主題

- 步驟 1：準備
- 步驟 2：建立應用程式
- 步驟 3：設定應用程式輸出
- 步驟 4：驗證輸出

後續步驟

步驟 1：準備

步驟 1：準備

為本練習建立 Amazon Kinesis Data Analytics 應用程式之前，您必須建立兩個 Kinesis 資料串流。將其中一個串流設定為應用程式的串流來源，另一個串流設定為 Kinesis Data Analytics 保留應用程式輸出的目的地。

主題

- 步驟 1.1：建立輸入和輸出資料串流
- 步驟 1.2：將範例記錄寫入輸入串流。

步驟 1.1：建立輸入和輸出資料串流

在本節中，您會建立兩個 Kinesis 串流：`ExampleInputStream` 和 `ExampleOutputStream`。您可以使用 AWS Management Console 或 AWS CLI 建立這些串流。

- 使用主控台

  1. 前往 https://console.aws.amazon.com/kinesis/ 登入 AWS Management Console 並開啟 Kinesis 主控台。
  2. 選擇 建立資料串流。建立具有 `ExampleInputStream` 碎片之串流。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的建立串流。
  3. 重複上一個步驟，以名為 `ExampleOutputStream` 的碎片建立串流。

- 使用 AWS CLI

  1. 若要建立第一個串流 (`ExampleInputStream`)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

  2. 運行相同的命令，將串流名稱更改為 `ExampleOutputStream`。這個命令會建立應用程式用來寫入輸出的第二個串流。

步驟 1.2：將範例記錄寫入輸入串流。

在此步驟中，執行 Python 程式碼以持續產生範例記錄，並將這些記錄寫入 ExampleInputStream 串流。

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

1. 安裝 Python 與 pip。

   如需安裝 Python 的相關資訊，請參閱 Python 網站。

   您可以使用 Pip 安裝相依性。如需安裝 Pip 的詳細資訊，請參閱 pip 網站的安裝。

2. 執行以下 Python 程式碼。程式碼中的 put-record 命令會將 JSON 記錄寫入串流。

```
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"))
```

後續步驟

步驟 2：建立應用程式

步驟 2：建立應用程式

在本節建立 Amazon Kinesis Data Analytics 應用程式，如下所示：

- 設定應用程式輸入，以使用您在 the section called “步驟 1：準備” 中建立的 Kinesis 資料串流作為串流來源。
- 使用主控台上的異常偵測範本。

建立應用程式

1. 按照 Kinesis Data Analytics 入門練習中的步驟 1、2 和 3 進行操作 (請參閱 步驟 3.1：建立應用程式)。

   - 在來源設定中，執行下列動作：
     - 指定您在前一節建立的串流來源。
     - 主控台推斷結構描述後，請編輯結構描述，並將 `heartRate` 欄類型設定為 `INTEGER`。

       大部分的心率值都是正常的，而且探索程序很可能會將 `TINYINT` 類型指派給此資料欄。但是，一小部分的值顯示出高心率。如果這些高數值不符合 `TINYINT` 類型，Kinesis Data

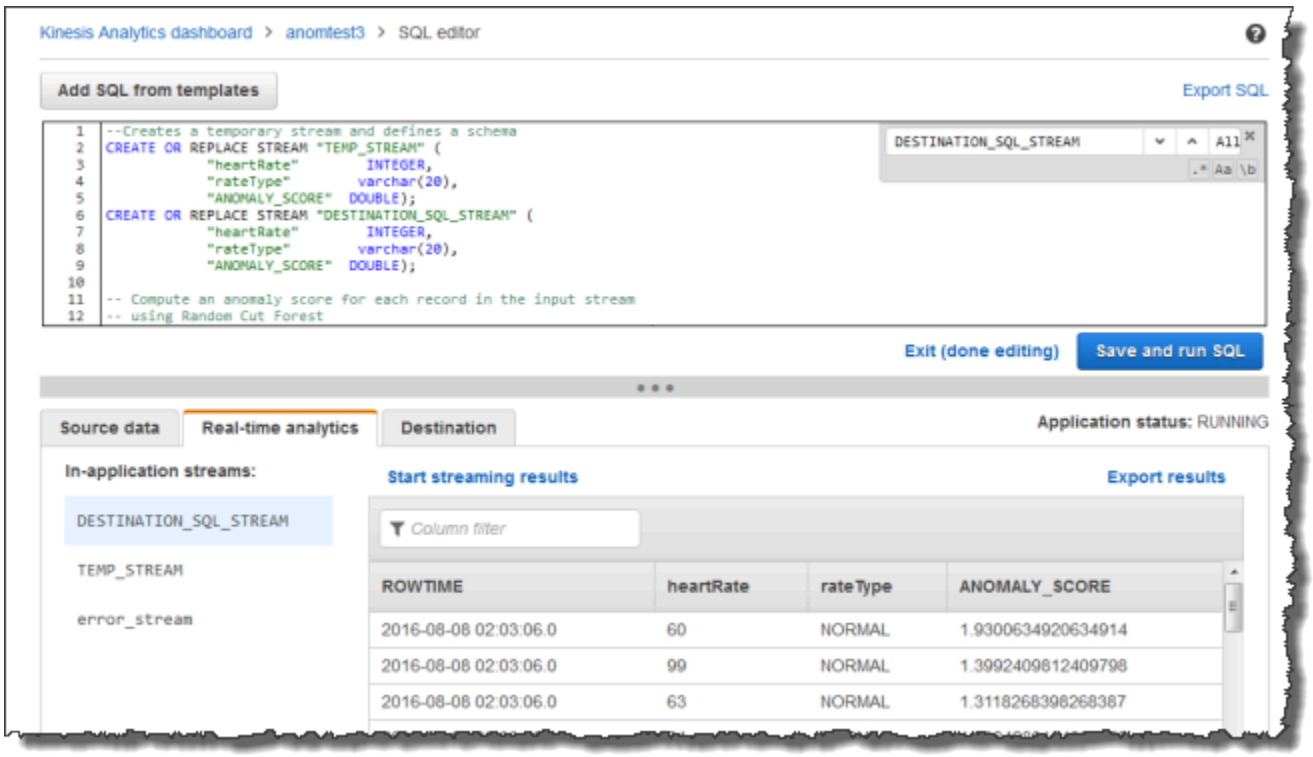
Analytics 會將這些資料列傳送至錯誤串流。將資料類型更新為 INTEGER，以便容納所有產生的心率資料。

- 使用主控台上的異常偵測範本。然後，您可以更新範本程式碼，以提供適當的資料欄名稱。

2. 提供資料欄名稱以更新應用程式碼。生成的應用程式碼如下所示 (將此代碼貼到 SQL 編輯器中)：

```
--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. 在 Kinesis Data Analytics 主控台上執行 SQL 程式碼並檢閱結果：



後續步驟

[步驟 3：設定應用程式輸出]

步驟 3：設定應用程式輸出

完成 the section called “步驟 2：建立應用程式” 後，將得到應用程式碼，該程式碼會從串流來源讀取心率資料，並為每個來源指派異常分數。

您現在可以將應用程式結果從應用程式內串流傳送到外部目的地，即另一個 Kinesis 資料串流 (`OutputStreamTestingAnomalyScores`)。您可以分析異常分數並確定哪些心率是異常的。然後，您可以進一步擴充此應用程式以產生提醒。

請依照下列步驟設定應用程式輸出：

1. 開啟 Amazon Kinesis Data Analytics 主控台。在 SQL 編輯器中，選擇應用程式儀表板中的目的地或新增目的地。
2. 在連線至目的地頁面，選擇您在前一節建立的 `OutputStreamTestingAnomalyScores` 串流。

現在您有一個外部目的地，可讓 Amazon Kinesis Data Analytics 將應用程式寫入的任何紀錄保留在應用程式內 DESTINATION_SQL_STREAM 串流中。

3. 您可以選擇性地設定 AWS Lambda 來監視 OutputStreamTestingAnomalyScores 串流，並傳送提醒給您。如需說明，請參閱 使用 Lambda 函數預處理資料。如果未設定提醒，您可以檢閱 Kinesis Data Analytics 寫入外部目的地 (Kinesis 資料串流) 的記錄 OutputStreamTestingAnomalyScores，如 步驟 4：驗證輸出 中所述。

後續步驟

步驟 4：驗證輸出

步驟 4：驗證輸出

在 the section called “步驟 3：設定應用程式輸出” 設定應用程式輸出後，請使用下列 AWS CLI 命令來讀取應用程式寫入目的地串流的記錄：

1. 運行 get-shard-iterator 命令以獲取指向輸出串流資料的指標。

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

您將得到具有碎片迭代器值的回應，如下列範例回應所示：

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

複製碎片迭代器值。

2. 執行 AWS CLI get-records 命令。

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

此命令會傳回記錄頁面，以及您可以在後續 `get-records` 命令中使用的另一個碎片迭代器來擷取下一組記錄。

範例：偵測資料異常並取得說明 (RANDOM_CUT_FOREST_WITH_EXPLANATION 函數)

Amazon Kinesis Data Analytics 提供 `RANDOM_CUT_FOREST_WITH_EXPLANATION` 函數，可根據數值欄中的值為每筆記錄指派異常分數。該函數還提供了異常的解釋。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的 RANDOM_CUT_FOREST_WITH_EXPLANATION。

在本練習中，撰寫應用程式碼，以取得應用程式串流來源上的紀錄異常分數。您還可以獲得每個異常的解釋。

主題

- 步驟 1：準備資料
- 步驟 2：建立 Analytics 應用程式
- 步驟 3：檢查結果

首要步驟

步驟 1：準備資料

步驟 1：準備資料

在為此範例建立 Amazon Kinesis Data Analytics 應用程式之前，您必須先建立 Kinesis 資料串流作為應用程式的串流來源。您也可以執行 Python 程式碼，將模擬的血壓資料寫入串流。

主題

- 步驟 1.1：建立 Kinesis 資料串流
- 步驟 1.2：將範例記錄寫入輸入串流

步驟 1.1：建立 Kinesis 資料串流

在本節中，建立名為 `ExampleInputStream` 的 Kinesis 資料串流。您可以使用 AWS Management Console 或 AWS CLI 建立這些資料串流。

- 使用主控台：

  1. 前往 https://console.aws.amazon.com/kinesis/ 登入 AWS Management Console 並開啟 Kinesis 主控台。
  2. 在導覽窗格中選擇 資料串流。選擇 建立 Kinesis 串流。
  3. 在角色名稱輸入 **ExampleInputStream**。在碎片數鍵入 **1**。
- 或者，若要使用 AWS CLI 建立資料串流，請執行下列命令：

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

步驟 1.2：將範例記錄寫入輸入串流

在此步驟中，執行 Python 程式碼以持續產生範例記錄，並將其寫入您建立的資料串流。

1. 安裝 Python 與 Pip。

   如需安裝 Python 的相關資訊，請參閱 Python 網站。

   您可以使用 Pip 安裝相依性。如需安裝 Pip 的詳細資訊，請參閱 Pip 文件中的安裝。
2. 執行以下 Python 程式碼。您可以將區域變更為要在本範例中使用的區域。程式碼中的 `put-record` 命令會將 JSON 記錄寫入串流。

```
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"))
```

後續步驟

[步驟 2：建立 Analytics 應用程式](#)

步驟 2：建立 Analytics 應用程式

在本節中，建立 Amazon Kinesis Data Analytics 應用程式，並將您在 the section called "步驟 1：準備資料" 中建立的 Kinesis 資料串流設定為串流來源 。然後執行使用該 RANDOM_CUT_FOREST_WITH_EXPLANATION 函數的應用程式碼。

建立應用程式

1. 在 https://console.aws.amazon.com/kinesis 上開啟 Kinesis 主控台。
2. 在導覽窗格中，選擇 Data Analytics，然後選擇建立應用程式。
3. 提供應用程式名稱和說明 (選用)，然後選擇建立應用程式。
4. 選擇連接串流資料，然後從清單中選擇 ExampleInputStream。
5. 選擇探索結構描述，並確定 Systolic 與 Diastolic 顯示為 INTEGER 資料欄。如果它們具備其他類型，請選擇編輯結構描述，然後將類型 INTEGER 指定給它們。
6. 在即時分析下方，選擇至 SQL 編輯器。出現提示時，選擇執行您的應用程式。
7. 將下列程式碼貼到 SQL 編輯器中，然後選擇儲存並執行 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;
```

後續步驟

步驟 3：檢查結果

步驟 3：檢查結果

執行此範例的 SQL 程式碼時，您會先看到異常分數等於零的資料列。這發生在初始學習階段。您會得到類似以下的結果：

```
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 函數中的演算法會確保 Systolic 和 Diastolic 資料欄為數值，並將當作它們輸入。

- 該 BloodPressureLevel 資料欄具有文本資料，因此演算法不會考慮。此資歷欄僅為視覺輔助，以幫助您快速發現範例中的正常、高血壓和低血壓水準。
- 在 ANOMALY_SCORE 欄中，分數較高的記錄更為異常。此樣本結果集內的第二個記錄最異常，異常分數為 3.855851061。
- 若要瞭解演算法考量的每個數值欄對異常分數的貢獻程度，請參閱 ANOMALY_SCORE 欄中名為 ATTRIBUTION_SCORE 的 JSON 欄位。在這組樣本結果第二列的情況下，Systolic 和 Diastolic 欄貢獻給異常的比例為 1. 7447:2.1111。換句話說，異常分數 45% 的解釋歸因於收縮值，其餘歸因於舒張值。
- 若要判斷此範例第二列所表示的點異常之方向，請參閱名為 DIRECTION 的 JSON 欄位。在此案例中，舒張值和收縮值都標記為 HIGH。若要判斷這些方向正確的信賴度，請參閱名為 STRENGTH 的 JSON 欄位。在此範例中，演算法更有信心舒張值高。事實上，舒張讀數的正常值通常為 60—80，123 比預期高得多。

範例：偵測串流上的熱點 (熱點功能)

Amazon Kinesis Data Analytics 提供的 HOTSPOTS 功能可以找出並傳回資料中相對密集區域的相關資訊。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink SQL 參考資料中的熱點。

在本練習中撰寫應用程式碼，以便在應用程式的串流來源上尋找熱點。若要設定應用程式，請執行下列動作：

1. 設定串流來源：設定 Kinesis 串流並寫入範例座標資料，如下所示：

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

   此範例提供 Python 指令碼供您填入串流。x 和 y 值是隨機產生的，有些記錄會叢集在某些位置。

   如果指令碼刻意生成數值做為熱點的一部分，則會提供 is_hot 欄位做為指標。這可協助您評估熱點偵測功能是否正常運作。

2. 建立應用程式：接著使用 AWS Management Console 建立 Kinesis Data Analytics 應用程式。將串流來源對應至應用程式內串流 (SOURCE_SQL_STREAM_001)，以設定應用程式輸入。當應用程式啟動時，Kinesis Data Analytics 會持續讀取串流來源，並將記錄插入應用程式內串流。

   在本練習中，針對應用程式使用下列程式碼：

   ```
   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)
    );
```

此程式碼會讀取 SOURCE_SQL_STREAM_001 中的資料列、分析找出顯著熱點，然後將產生的資料寫入另一個應用程式內串流 (DESTINATION_SQL_STREAM)。您可以使用幫浦將資料列插入應用程式內串流。如需更多詳細資訊，請參閱 應用程式內串流與幫浦。

3. 設定輸出：設定應用程式輸出，將資料從應用程式傳送至外部目的地，即另一個 Kinesis 資料串流。查看熱點分數，並判斷哪些分數表示出現了熱點（且需要收到提醒）。您可以使用 AWS Lambda 功能進一步處理熱點資訊和設定提醒。
4. 驗證輸出：此範例包含一個 JavaScript 應用程式，可從輸出串流讀取資料並以圖形方式顯示，以便您即時檢視應用程式產生的熱點。

本練習會使用美國西部 (奧勒岡) (us-west-2) 來建立這些串流及應用程式。如果您使用任何其他區域，請更新相應程式碼。

主題

- 步驟 1：建立輸入和輸出串流
- 步驟 2：建立 Amazon Kinesis Data Analytics 應用程式
- 步驟 3：設定應用程式輸出
- 步驟 4：驗證應用程式輸出

步驟 1：建立輸入和輸出串流

為熱點範例建立 Amazon Kinesis Data Analytics 應用程式之前，您必須建立兩個 Kinesis 資料串流。將其中一個串流設定為應用程式的串流來源，另一個串流設定為 Kinesis Data Analytics 保留應用程式輸出的目的地。

主題

步驟 1.1：建立 Kinesis 資料串流

在本節中，建立兩個 Kinesis 資料串流：ExampleInputStream 和 ExampleOutputStream。

使用主控台或 AWS CLI 建立資料串流。

- 如要使用主控台建立資料串流：
  1. 前往 https://console.aws.amazon.com/kinesis/ 登入 AWS Management Console 並開啟 Kinesis 主控台。
  2. 在導覽窗格中選擇 資料串流。
  3. 選擇建立 Kinesis 串流，然後建立內含一個名為 ExampleInputStream 的碎片之串流。
  4. 重複上一個步驟，以名為 ExampleOutputStream 的碎片建立串流。
- 使用 AWS CLI 建立資料串流：
  - 使用下列 Kinesis create-stream AWS CLI 命令建立串流 (ExampleInputStream 和 ExampleOutputStream)。若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，並將串流名稱變更為 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
```

步驟 1.2：將範例記錄寫入輸入串流

在此步驟中，執行 Python 程式碼以持續產生範例記錄，並將這些記錄寫入 ExampleInputStream 串流。

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

1. 安裝 Python 與 pip。

   如需安裝 Python 的相關資訊，請參閱 Python 網站。

   您可以使用 Pip 安裝相依性。如需安裝 Pip 的詳細資訊，請參閱 pip 網站的安裝。

2. 執行以下 Python 程式碼。該程式碼會執行下列作業：

   - 在 (X, Y) 平面中的某個位置生成潛在熱點。
   - 為每個熱點產生一組 1,000 個點。在這些點中，20% 會叢集在熱點周圍。其餘部分會在整個空間內隨機產生。
   - put-record 命令會將 JSON 記錄寫入串流。

   > **Important**
   >
   > 請勿將此檔案上傳到 Web 伺服器，因為它包含您的 AWS 憑證。

```
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"),
    )
```

後續步驟

步驟 2：建立 Amazon Kinesis Data Analytics 應用程式

步驟 2：建立 Amazon Kinesis Data Analytics 應用程式

在熱點範例的本節中，建立 Amazon Kinesis Data Analytics 應用程式，如下所示：

- 設定應用程式輸入，以使用您在步驟 1 中建立的 Kinesis 資料串流作為串流來源。
- 使用 AWS Management Console 中提供的應用程式碼。

建立應用程式

1. 按照入門練習中的步驟 1、2 和 3 建立 Kinesis Data Analytics 應用程式 (請參閱 步驟 3.1：建立應用程式)。

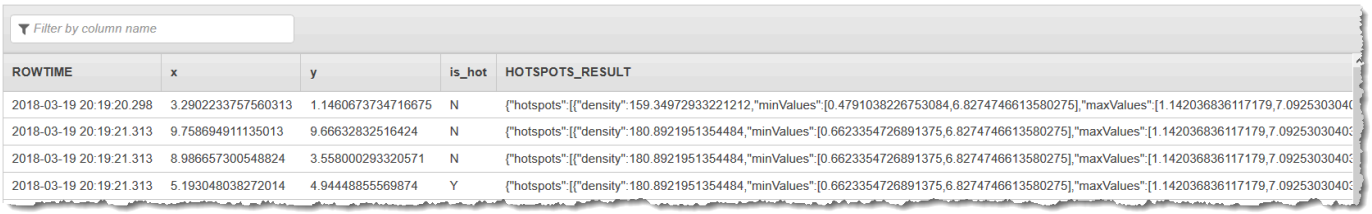
   在來源設定中，執行下列動作：

   - 指定您在 the section called “步驟 1：建立串流” 中建立的串流來源。
   - 主控台推斷結構描述後，請編輯結構描述。請確定 x 和 y 欄類型已設定為 DOUBLE，且 IS_HOT 欄類型設定為 VARCHAR。
2. 使用以下應用程式碼 (您可將此代碼貼到 SQL 編輯器中)：

```
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. 執行 SQL 程式碼並檢閱結果。



Filter by column name

ROWTIME	x	y	is_hot	HOTSPOTS_RESULT
2018-03-19 20:19:20.298	3.2902233757560313	1.1460673734716675	N	{"hotspots":[{"density":159.34972933221212,"minValues":[0.4791038226753084,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040
2018-03-19 20:19:21.313	9.758694911135013	9.66632832516424	N	{"hotspots":[{"density":180.8921951354484,"minValues":[0.6623354726891375,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040
2018-03-19 20:19:21.313	8.986657300548824	3.558000293320571	N	{"hotspots":[{"density":180.8921951354484,"minValues":[0.6623354726891375,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040
2018-03-19 20:19:21.313	5.193048038272014	4.94448855569874	Y	{"hotspots":[{"density":180.8921951354484,"minValues":[0.6623354726891375,6.8274746613580275],"maxValues":[1.142036836117179,7.0925303040

後續步驟

步驟 3：設定應用程式輸出

步驟 3：設定應用程式輸出

在熱點範例中，可用 Amazon Kinesis Data Analytics 應用程式碼探索串流來源的重要熱點，並為每個熱點指派熱度分數。

您現在可以將應用程式結果從應用程式內串流傳送到外部目的地，即另一個 Kinesis 資料串流 (ExampleOutputStream)。然後，您可以分析熱點分數，並確定熱點熱度的適當閾值。您可以進一步擴充此應用程式以產生提醒。

如要設定應用程式輸出

1. 在 https://console.aws.amazon.com/kinesis 上開啟 Kinesis Data Analytics 主控台。
2. 在 SQL 編輯器中，選擇應用程式儀表板中的目的地或新增目的地。
3. 在新增目的地頁面，選擇從串流中選取。然後，選擇您在上一節建立的 ExampleOutputStream 串流。

現在您有一個外部目的地，可讓 Amazon Kinesis Data Analytics 將應用程式寫入的任何紀錄保留在應用程式內 DESTINATION_SQL_STREAM 串流中。

4. 您可以選擇性地設定 AWS Lambda 來監視 ExampleOutputStream 串流，並傳送提醒給您。如需更多詳細資訊，請參閱 使用 Lambda 函數作為輸出。您也可以檢閱 Kinesis Data Analytics 寫入外部目的地 (Kinesis 串流 ExampleOutputStream) 的記錄，如 步驟 4：驗證應用程式輸出 中所述。

後續步驟

步驟 4：驗證應用程式輸出

步驟 4：驗證應用程式輸出

在熱點範例的這節中，設定 Web 應用程式，以可擴展向量圖形 (SVG) 控制顯示熱點資訊。

1. 使用下列內容建立名為 index.html 的檔案：

```
<!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. 在同一目錄中，建立名為 hotspots_viewer.js 的檔案，內含下列內容：在提供的變數中提供您的 、憑證和輸出串流名稱。

```
// Visualize example output from the Kinesis Analytics hotspot detection algorithm.
// This script assumes that the output stream has a single shard.

// Modify this section to reflect your AWS configuration
var awsRegion = "",        // The  where your Kinesis Analytics application is
 configured.
    accessKeyId = "",      // Your Access Key ID
    secretAccessKey = "",  // Your Secret Access Key
    outputStream = "";     // The name of the Kinesis Stream where the output from
 the HOTSPOTS function is being written

// The variables in this section should reflect way input data was generated and
 the parameters that the HOTSPOTS
// function was called with.
var windowSize = 1000, // The window size used for hotspot detection
    minimumDensity = 40,  // A filter applied to returned hotspots before
 visualization
    xRange = [0, 10],  // The range of values to display on the x-axis
    yRange = [0, 10];  // The range of values to display on the y-axis

////////////////////////////////////////////////////////////////////////////////
// D3 setup
////////////////////////////////////////////////////////////////////////////////
```

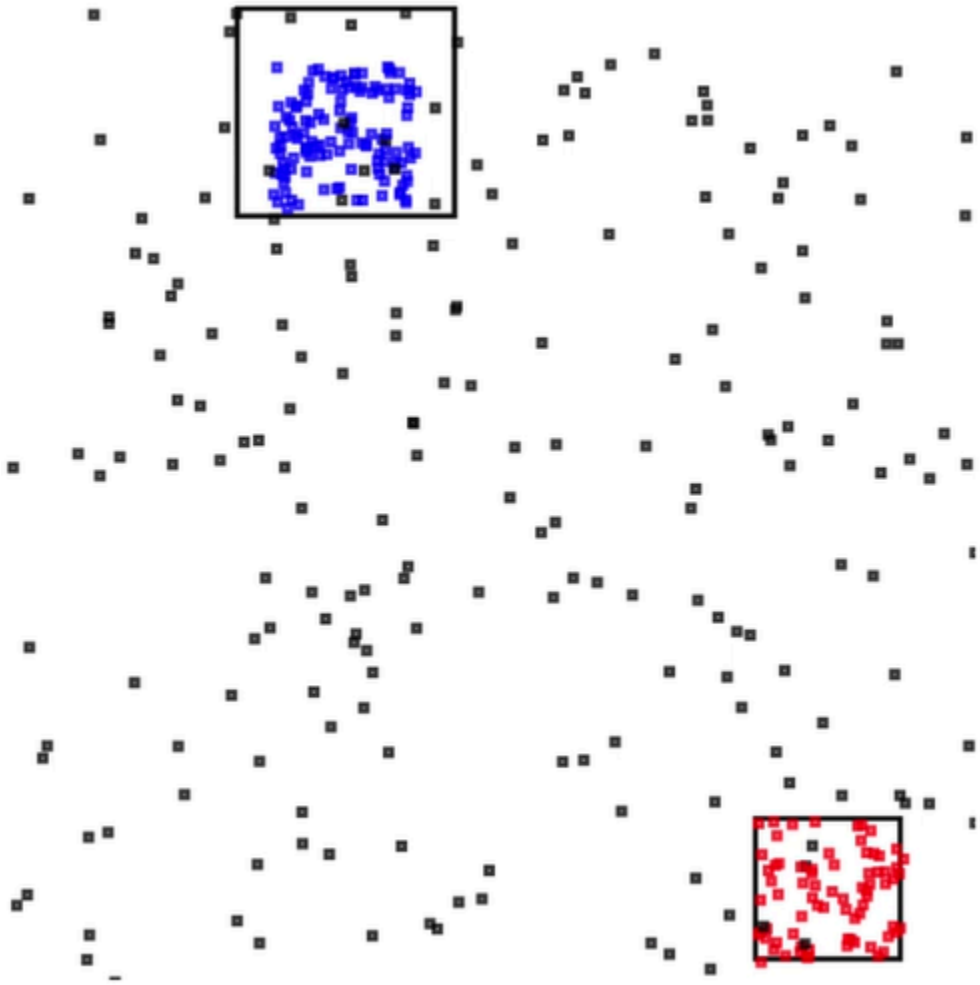
```
var svg = d3.select("svg"),
    margin = {"top": 20, "right": 20, "bottom": 20, "left": 20},
    graphWidth = +svg.attr("width") - margin.left - margin.right,
    graphHeight = +svg.attr("height") - margin.top - margin.bottom;

// Return the linear function that maps the segment [a, b] to the segment [c, d].
function linearScale(a, b, c, d) {
    var m = (d - c) / (b - a);
    return function(x) {
        return c + m * (x - a);
    };
}

// helper functions to extract the x-value from a stream record and scale it for
 output
var xValue = function(r) { return r.x; },
    xScale = linearScale(xRange[0], xRange[1], 0, graphWidth),
    xMap = function(r) { return xScale(xValue(r)); };

// helper functions to extract the y-value from a stream record and scale it for
 output
var yValue = function(r) { return r.y; },
    yScale = linearScale(yRange[0], yRange[1], 0, graphHeight),
    yMap = function(r) { return yScale(yValue(r)); };

// a helper function that assigns a CSS class to a point based on whether it was
 generated as part of a hotspot
var classMap = function(r) { return r.is_hot == "Y" ? "point hot" : "point
 cold"; };

var g = svg.append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

function update(records, hotspots) {

    var points = g.selectAll("circle")
        .data(records, function(r) { return r.dataIndex; });

    points.enter().append("circle")
        .attr("class", classMap)
        .attr("r", 3)
        .attr("cx", xMap)
        .attr("cy", yMap);
```

```
    points.exit().remove();

    if (hotspots) {
        var boxes = g.selectAll("rect").data(hotspots);

        boxes.enter().append("rect")
            .merge(boxes)
            .attr("class", "hotspot")
            .attr("x", function(h) { return xScale(h.minValues[0]); })
            .attr("y", function(h) { return yScale(h.minValues[1]); })
            .attr("width", function(h) { return xScale(h.maxValues[0]) -
 xScale(h.minValues[0]); })
            .attr("height", function(h) { return yScale(h.maxValues[1]) -
 yScale(h.minValues[1]); });

        boxes.exit().remove();
    }
}

////////////////////////////////////////////////////////////////////////////////////
// Use the AWS SDK to pull output records from Kinesis and update the visualization
////////////////////////////////////////////////////////////////////////////////////

var kinesis = new AWS.Kinesis({
    "region": awsRegion,
    "accessKeyId": accessKeyId,
    "secretAccessKey": secretAccessKey
});

var textDecoder = new TextDecoder("utf-8");

// Decode an output record into an object and assign it an index value
function decodeRecord(record, recordIndex) {
    var record = JSON.parse(textDecoder.decode(record.Data));
    var hotspots_result = JSON.parse(record.HOTSPOTS_RESULT);
    record.hotspots = hotspots_result.hotspots
        .filter(function(hotspot) { return hotspot.density >= minimumDensity});
    record.index = recordIndex
    return record;
}

// Fetch a new records from the shard iterator, append them to records, and update
 the visualization
```

```
function getRecordsAndUpdateVisualization(shardIterator, records, lastRecordIndex)
 {
    kinesis.getRecords({
        "ShardIterator": shardIterator
    }, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            return;
        }

        var newRecords = data.Records.map(function(raw) { return decodeRecord(raw,
 ++lastRecordIndex); });
        newRecords.forEach(function(record) { records.push(record); });

        var hotspots = null;
        if (newRecords.length > 0) {
            hotspots = newRecords[newRecords.length - 1].hotspots;
        }

        while (records.length > windowSize) {
            records.shift();
        }

        update(records, hotspots);

        getRecordsAndUpdateVisualization(data.NextShardIterator, records,
 lastRecordIndex);
    });
}

// Get a shard iterator for the output stream and begin updating the visualization.
 Note that this script will only
// read records from the first shard in the stream.
function init() {
    kinesis.describeStream({
        "StreamName": outputStream
    }, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            return;
        }

        var shardId = data.StreamDescription.Shards[0].ShardId;
```

```
        kinesis.getShardIterator({
            "StreamName": outputStream,
            "ShardId": shardId,
            "ShardIteratorType": "LATEST"
        }, function(err, data) {
            if (err) {
                console.log(err, err.stack);
                return;
            }
            getRecordsAndUpdateVisualization(data.ShardIterator, [], 0);
        })
    });
}

// Start the visualization
init();
```

3. 在執行第一部分的 Python 程式碼之後，在網頁瀏覽器中開啟 `index.html`。熱點資訊會顯示在頁面上，如下所示。



範例：提醒與錯誤

本節提供使用提醒與錯誤的 Kinesis Data Analytics 應用程式範例。每個範例皆包含逐步指示與程式碼，可協助您建立 Kinesis Data Analytics 應用程式並測試結果。

主題

- 範例：建立簡單提醒
- 範例：建立限流提醒
- 範例：探索應用程式內錯誤串流

範例：建立簡單提醒

在此 Kinesis Data Analytics 應用程式中，查詢會在透過示範串流建立的應用程式內串流上持續執行。如需更多詳細資訊，請參閱 持續查詢。

如果有任何資料列顯示大於 1% 的股票價格變更，則這些資料欄會插入另一個應用程式內串流。在練習中，您可以規劃應用程式輸出讓結果持續留在外部目的地。然後，您可以進一步調查結果。舉例來說，您可以使用 AWS Lambda 函數來處理記錄，並傳送提醒。

建立簡單提醒應用程式

1. 依照 Kinesis Data Analytics入門練習中所述建立分析應用程式。
2. 在 Kinesis Data Analytics 的 SQL 編輯器中，以下列項目取代應用程式碼：

```
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;
```

   應用程式碼中的 SELECT 陳述式會篩選 SOURCE_SQL_STREAM_001 中的資料欄，找出大於 1% 的股票價格變動。然後，它會使用幫浦將這些資料欄插入另一個應用程式內串流 DESTINATION_SQL_STREAM。如需使用幫浦將資料欄插入應用程式串流之編碼模式的相關詳細資訊，請參閱 應用程式碼。
3. 選擇 儲存並執行 SQL。
4. 新增目的地。如要新增，請在 SQL 編輯器中選擇目的地標籤，或在應用程式詳細資訊頁面上選擇新增目的地。

   a. 在 SQL 編輯器中，選擇目的地標籤，然後選擇連線至目的地。

      在連線至目的地頁面上，選擇新增。

   b. 選擇至 Kinesis 串流。

c. 在 Amazon Kinesis Data Streams 主控台上，建立具有一個碎片的新 Kinesis 串流 (例如 gs-destination)。等待直到流狀態為作用中。

d. 返回 Kinesis Data Analytics 主控台。在連線至目的地頁面，選擇您建立的串流。

如果串流未出現，請重新整理頁面。

e. 選擇儲存並繼續。

現在您有一個外部目的地，即 Kinesis 資料串流，Kinesis Data Analytics 會將您的應用程式輸出保留在 DESTINATION_SQL_STREAM 應用程式內串流中。

5. 設定 AWS Lambda 以監控您建立的 Kinesis 串流，並調用 Lambda 函數。

如需說明，請參閱 使用 Lambda 函數預處理資料。

範例：建立限流提醒

在此 Kinesis Data Analytics 應用程式中，查詢會在透過示範串流建立的應用程式內串流上持續執行。如需更多詳細資訊，請參閱 持續查詢。如果有任何資料欄顯示大於 1% 的股票價格變更，則這些資料列會插入另一個應用程式內串流。該應用程序會限流提醒，以便在股票價格變化時立即發送提醒。但是，每個股票代碼每分鐘不會發送超過一個提醒到應用程式內串流。

建立限流提醒應用程式

1. 依照 Kinesis Data Analytics入門練習中所述建立 Kinesis Data Analytics 應用程式。
2. 在 Kinesis Data Analytics 的 SQL 編輯器中，以下列項目取代應用程式碼：

```
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;
```

應用程式碼中的 SELECT 陳述式會篩選 SOURCE_SQL_STREAM_001 中的資料欄，找出變更大於 1% 的股票價格，並使用幫浦將這些資料列插入另一個應用程式內串流 CHANGE_STREAM。

接著，應用程式會建立第二個名為 TRIGGER_COUNT_STREAM 的串流，來進行限流提醒。第二個查詢會從視窗中選取記錄，該視窗在每次接受記錄時向前跳轉，因此每個股票代號每分鐘只有一筆記錄會寫入串流。

3. 選擇儲存並執行 SQL。

輸出至 TRIGGER_COUNT_STREAM 的串流類似於下列範例：

Filter by column name

ROWTIME	TICKER_SYMBOL	CHANGE	TRIGGER_COUNT
2018-01-08 22:59:15.742	ASD	-1.77	1
2018-01-08 22:59:15.742	ASD	-1.77	1
2018-01-08 22:59:20.752	DFT	-3.16	1
2018-01-08 22:59:35.775	IOP	-1.88	1

範例：探索應用程式內錯誤串流

Amazon Kinesis Data Analytics 會為您建立的每個應用程式提供應用程式內錯誤串流。應用程式無法處理的任何資料欄都會傳送至此錯誤串流。您可以考慮將錯誤串流資料保存到外部目的地，以便進行調查。

在主控台上執行以下練習。在這些範例中，編輯探索過程推斷的結構描述，然後驗證傳送至錯誤資料串流的資料欄，以便在輸入組態中引入錯誤。

主題

介紹剖析錯誤

在本練習中，您會引入剖析錯誤。

1. 依照 Kinesis Data Analytics 入門練習中所述建立 Kinesis Data Analytics 應用程式。
2. 在應用程式詳細資料頁面上，選擇連接串流資料。
3. 如果按照入門練習進行操作，您的帳戶中即會有一個示範串流 (`kinesis-analytics-demo-stream`)。在連接到來源頁面，選擇此示範串流。
4. Kinesis Data Analytics 會從示範串流取得範例，為其建立的應用程式內輸入串流推斷結構描述。主控台會在格式化串流範例標籤中顯示推斷的結構描述和範例資料。
5. 接下來，編輯結構描述並修改資料欄類型，以引入剖析錯誤。選擇編輯結構描述。
6. 將 `TICKER_SYMBOL` 資料欄類型從 `VARCHAR(4)` 變更為 `INTEGER`。

   既然建立的應用程式內結構描述之資料欄類型無效，Kinesis Data Analytics 就無法將資料引入應用程式內串流。相反地，它會將資料欄傳送至錯誤資料流。
7. 選擇儲存結構描述。
8. 選擇重新整理結構描述範例。

   請注意，格式化串流範例中沒有資料欄。但是，錯誤串流標籤顯示帶有錯誤訊息的資料。錯誤串流標籤會顯示傳送至應用程式內錯誤串流的資料。

   因為您已變更資料欄類型，Kinesis Data Analytics 無法將資料引入應用程式內的輸入串流。相反地，它會將資料欄傳送至錯誤資料流。

引入除以零錯誤

在本練習中，更新應用程式碼以引入執行期錯誤 (除以零)。請注意，Amazon Kinesis Data Analytics 會將產生的資料列傳送到應用程式內錯誤串流，而不是傳送到應該寫入結果的 DESTINATION_SQL_STREAM 應用程式內串流。

1. 依照 Kinesis Data Analytics 入門練習中所述建立 Kinesis Data Analytics 應用程式。

   在即時分析標籤上驗證結果，如下所示：

   酸

2. 更新應用程式碼中的 SELECT 陳述式，以引入除以零；例如：

```
SELECT STREAM ticker_symbol, sector, change, (price / 0) as ProblemColumn
FROM "SOURCE_SQL_STREAM_001"
WHERE sector SIMILAR TO '%TECH%';
```

3. 執行應用程式。

   由於發生除以零的執行期錯誤，Kinesis Data Analytics 會將資料欄傳送到應用程式內錯誤串流，而不是寫入 DESTINATION_SQL_STREAM。在即時分析標籤上，選擇錯誤串流，然後您就可以在應用程式內錯誤串流中看到這些資料欄。

範例：解決方案加速器

AWS解決方案網站提供AWS CloudFormation範本，您可以使用這些範本快速建立完整的串流資料解決方案。

可使用以下範本：

即時洞察 AWS 帳戶 活動

此解決方案可即時記錄和視覺化您的 AWS 帳戶 資源存取和使用量指標。如需詳細資訊，請參閱AWS帳戶活動的即時洞察。

使用 Kinesis Data Analytics 進行 AWS IoT 裝置的即時監控

此解決方案可即時收集、處理、分析和視覺化 IoT 裝置連線和活動資料。如需詳細資訊，請參閱使用 Kinesis Data Analytics 進行即時 AWS IoT 裝置監控。

使用 Kinesis Data Analytics 來進行即時 Web 分析

該解決方案實時收集，處理，分析和視覺化網站點擊流資料。如需詳細資訊，請參閱使用 Kinesis Data Analytics 進行即時 Web 分析。

Amazon Connected 車輛解決方案

該解決方案實時收集，處理，分析和視覺化來自車輛的 IoT 資料。如需詳細資訊，請參閱 Amazon Connected 車輛解決方案。

中的安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您將受益於資料中心和網路架構，專為滿足最敏感安全性組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同的責任模型 將此描述為雲端 本身 的安全和雲端內部的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。第三方稽核人員定期檢測及驗證安全的效率也是我們 AWS 合規計劃的一部分。若要進一步瞭解適用於 的合規計劃，請參閱 AWS 合規計劃範圍內的服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。

本文件有助於您了解如何在使用 時套用共同責任模型。下列主題說明如何將 設定為達到您的安全及法規遵循目標。您也將了解如何使用其他 Amazon 服務來監控和保護資源。

主題

- Amazon Kinesis Data Analytics for SQL 應用程式中的資料保護
- Kinesis Data Analytics 中的 Identity and Access Management
- 的身分驗證與存取控制
- 監控
- Amazon Kinesis Data Analytics for SQL 應用程式的合規驗證
- Amazon Kinesis Data Analytics 中的恢復能力
- Kinesis Data Analytics for SQL 應用程式中的基礎設施安全
- 適用於 Kinesis Data Analytics 的安全最佳實務

Amazon Kinesis Data Analytics for SQL 應用程式中的資料保護

您可以使用提供的工具保護您的數據 AWS。Kinesis Data Analytics 可以使用支援加密資料的服務，包括 Kinesis Data Streams、Firehose 和 Amazon S3。

Kinesis Data Analytics 中的資料加密

靜態加密

使用 Kinesis Data Analytics 來加密靜態資料時，請注意以下相關事項：

- 您可以使StartStreamEncryption用加密傳入 Kinesis 資料串流上的資料。如需詳細資訊，請參閱什麼是 Kinesis Data Streams 伺服器端加密？。
- 輸出資料可以使用 Firehose 進行靜態加密，將資料存放在加密的 Amazon S3 儲存貯體中。您可以指定 Amazon S3 儲存貯體使用的加密金鑰。如需詳細資訊，請參閱搭配使用伺服器端加密與 KMS 受管金鑰 (SSE-KMS) 來保護資料。
- 您的應用程式碼會進行靜態加密。
- 您應用程式的參考資料會進行靜態加密。

傳輸中加密

Kinesis Data Analytics 會對傳輸中的資料進行加密。所有 Kinesis Data Analytics 應用程式都會啟用傳輸中加密功能，且無法停用。

Kinesis Data Analytics 會在下列情況加密傳輸中的資料：

- 從 Kinesis Data Streams 傳輸到 Kinesis Data Analytics 的資料。
- 在 Kinesis Data Analytics 內部元件間傳輸的資料。
- Kinesis 資料分析與 Firehose 之間傳輸中的資料。

金鑰管理

Kinesis Data Analytics 中的資料加密使用服務管理金鑰。不支援客戶受管金鑰。

Kinesis Data Analytics 中的 Identity and Access Management

Amazon Kinesis Data Analytics 需要許可，才能從您在應用程式輸入組態中指定的串流來源讀取記錄。Amazon Kinesis Data Analytics 也需要許可，以將應用程式輸出寫入您在應用程式輸出組態中指定的串流。

您可以建立 Amazon Kinesis Data Analytics 可擔任的 IAM 角色來授與這些許可。您授與此角色的許可，決定了 Amazon Kinesis Data Analytics 擔任該角色時可進行的操作。

> **Note**
>
> 如果您想要自行建立 IAM 角色，本節資訊就非常有用。在 Amazon Kinesis Data Analytics 主控台中建立應用程式時，主控台可為您建立 IAM 角色。主控台會使用下列命名慣例來建立 IAM 角色：
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> 建立角色後，您可在 IAM 主控台中檢閱角色和附加政策。

各 IAM 角色都附加了兩項政策。在信任政策中，您可指定誰可以擔任角色。在許可政策 (可有一或多個) 中，您可指定要授與此角色的許可。以下各節說明了這些政策，以供您在建立 IAM 角色時使用。

信任政策

若要授與 Amazon Kinesis Data Analytics 許可來擔任角色，以存取串流或參考來源，您可將以下信任政策附加到 IAM 角色：

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

許可政策

如果您要建立 IAM 角色來允許 Amazon Kinesis Data Analytics 讀取應用程式的串流來源，就必須授予相關讀取動作的許可。根據您的來源 (例如 Kinesis 串流、Firehose 交付串流或 Amazon S3 儲存貯體中的參考來源)，您可以附加下列許可政策。

讀取 Kinesis Stream 的許可政策

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

讀取 Firehose 傳遞串流的權限原則

```
{
    "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**
>
> `firehose:Get*` 許可是指 Kinesis Data Analytics 用來存取串流的內部存取器。Firehose 交付流沒有公共訪問器。

如果您指示 Amazon Kinesis Data Analytics 將輸出寫入應用程式輸出組態中的外部目標，就需要將下列許可授與 IAM 角色。

寫入 Kinesis Stream 的許可政策

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

寫入 Firehose Delivery Stream 的許可政策

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

從 Amazon S3 儲存貯體讀取參考資料來源的許可政策

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

預防跨服務混淆代理人

在中 AWS，當某個服務 (呼叫服務) 呼叫另一個服務 (呼叫的服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來對其他客戶的資源採取動作，即使該服務不應有適當的許可，導致混淆代理人的問題。

為了防止代表混淆，請 AWS 提供工具，協助您使用已授予您帳戶資源存取權的服務主體來保護所有服務的資料。本節著重在如何預防 Kinesis Data Analytics 的跨服務混淆代理人，您也可以在 IAM 使用者指南的混淆代理人問題一節中深入了解此主題。

在適用於 SQL 的 Kinesis Data Analytics 的內容中，我們建議您在角色信任政策中使用 aws: SourceArn 和 aws: SourceAccount 全域條件內容金鑰，將角色的存取限制為只有預期資源產生的請求。

如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 aws:SourceArn。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 aws:SourceAccount。

aws:SourceArn 的值必須是 Kinesis Data Analytics 使用之資源的 ARN，並以下列格式指定：arn:aws:kinesisanalytics:region:account:resource

解決混淆代理人問題的建議方法，是在完整資源 ARN 使用 aws:SourceArn 全域條件內容索引鍵。

如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 aws:SourceArn 鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如：arn:aws:kinesisanalytics::111122223333:*。

雖然適用於 SQL API 的 Kinesis Data Analytics 中的大多數動作 (例如CreateApplicationAddApplicationInput和) DeleteApplication都是在特定應用程式的內容中執行，但DiscoverInputSchema動作不會在任何應用程式的內容中執行。這表示此動作中使用的角色不得在 SourceArn 條件索引鍵中完全指定資源。以下是使用萬用字元 ARN 的範例：

```
{
   ...
   "ArnLike":{
      "aws:SourceArn":"arn:aws:kinesisanalytics:us-east-1:123456789012:*"
   }
   ...
}
```

Kinesis Data Analytics for SQL 所產生的預設角色會使用此萬用字元。這可確保探索輸入結構描述在主控台體驗中順暢運作。不過，我們建議您編輯信任政策，以在探索結構描述之後使用完整 ARN 來全面緩解混淆代理人問題。

您提供給 Kinesis Data Analytics 的角色政策，以及為您產生的角色信任政策可以使用 aws: SourceArn 和 aws: SourceAccount 條件金鑰。

請執行下列步驟以防範混淆代理人問題：

防範混淆代理人問題

1. 登入 AWS 管理主控台，然後開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 請選擇角色，然後選擇您要修改的角色。
3. 選擇編輯信任政策。
4. 在編輯信任原則頁面上，使用一個或兩個 aws:SourceArn 和 aws:SourceAccount 全域條件內容索引鍵的政策取代預設 JSON 政策。請參閱以下政策範例：
5. 選擇 更新政策。

```
{
```

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

的身分驗證與存取控制

存取 需要憑證。這些登入資料必須具有存取 AWS 資源的許可，例如應用程式或 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。下列各節提供如何使用 AWS Identity and Access Management (IAM) 和 的詳細資訊，以協助您安全存取資源。

存取控制

您可以持有效登入資料為自己的請求進行身分驗證，但還須具備許可才能建立或存取 資源。例如，您必須具有許可才能建立應用程式。

以下章節說明如何管理 的許可。我們建議您先閱讀概觀。

- 管理您的 資源之存取許可的概觀
- 針對 使用以身分為基礎的政策 (IAM 政策)
- API 許可：動作、許可與資源參考

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程

式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱《AWS IAM Identity Center 使用者指南》中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰)的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需詳細資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色)的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色方法的相關資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並取得由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-idp.html中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人)存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。

- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色：服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者)的時機。

管理您的 資源之存取許可的概觀

> **Warning**
>
> 針對新專案，我們建議您優先選擇新的 Managed Service for Apache Flink Studio，而非 for SQL 應用程式。Managed Service for Apache Flink Studio 易於使用且具備進階分析功能，可讓您在幾分鐘內建置複雜的串流處理應用程式。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請遵循《AWS IAM Identity Center 使用者指南》的建立許可集合中的指示。

- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請遵循《IAM 使用者指南》的為第三方身分提供者 (聯合) 建立角色中的指示。

- IAM 使用者：
  - 建立您的使用者可擔任的角色。請遵循《IAM 使用者指南》的為 IAM 使用者建立角色中的指示。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南 的 新增權限至使用者 (主控台) 中的指示。

> **Note**
>
> 帳戶管理員 (或管理員使用者) 是具有管理員權限的使用者。如需詳細資訊，請參《IAM 使用者指南》中的 IAM 最佳實務。

主題

資源與操作

在 中，主要的資源是應用程式。在政策中，您使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。

這些資源都有與其相關的唯一 Amazon Resource Name (ARN)，如下表所示。

資源類型	ARN 格式
應用程式	`arn:aws:kinesisanalytics:` *`region`*`:`*`account-id`*`:application/` *`application-name`*

提供一組用於處理資源的操作。如需可用操作的清單，請參閱 動作。

了解資源所有權

無論是誰建立資源，都 AWS 帳戶 擁有在帳戶中建立的資源。具體來說，資源擁有者是驗證資源建立請求的主體實體 (即根帳戶、使用者或 IAM 角色) 的主體實體的。 AWS 帳戶 下列範例說明其如何運作：

- 如果您使用的 root 帳號憑證 AWS 帳戶 來建立應用程式，您 AWS 帳戶 就是資源的擁有者。(在中，資源即應用程式。)
- 如果您在您的中建立使用者， AWS 帳戶 並授與建立應用程式的權限給該使用者，則該使用者可以建立應用程式。不過 AWS 帳戶，使用者所屬的您擁有應用程式資源。我們強烈建議您將許可授與角色，而非使用者。
- 如果您在 AWS 帳戶 具有建立應用程式的許可中建立 IAM 角色，任何可以擔任該角色的人都可以建立應用程式。使用者所屬的 AWS 帳戶您擁有應用程式資源。

管理資源存取

許可政策描述誰可以存取哪些資源。下一節說明可用來建立許可政策的選項。

> **Note**
>
> 本節著重討論如何在 的環境中使用 IAM，它不提供 IAM 服務的詳細資訊。如需完整的 IAM 文件，請參閱 IAM 使用者指南中的什麼是 IAM？。如需 IAM 政策語法和說明的詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策參考。

連接至 IAM 身分的政策稱為身分識別型政策 (IAM 政策)。附加至資源的政策稱為以資源為基礎的政策。 僅支援以身分為基礎的政策 (IAM 政策)。

主題

- 身分類型政策 (IAM 政策)
- 資源型政策

身分類型政策 (IAM 政策)

您可以將政策連接到 IAM 身分。例如，您可以執行下列動作：

- 將許可政策連接至您帳戶中的使用者或群組：若要授予使用者建立資源 (例如資料表) 的許可，您可以將許可政策附加至使用者或其所屬的群組。
- 將許可政策連接至角色 (授予跨帳戶許可)：您可以將身分識別型許可政策連接至 IAM 角色，藉此授予跨帳戶許可。例如，帳戶 A 中的管理員可以建立角色，將跨帳戶許可授與另一個帳戶 AWS 帳戶 (例如，帳戶 B) 或 Amazon 服務，如下所示：
  1. 帳戶 A 管理員建立 IAM 角色，並將許可政策連接到可授與帳戶 A 中資源許可的角色。
  2. 帳戶 A 管理員會將信任政策連接至將帳戶 B 識別為可擔任角色之主體的角色。
  3. 帳戶 B 管理員即可將擔任該角色的許可委派給帳戶 B 中的任何使用者。這麼做可讓帳戶 B 的使用者建立或存取帳戶 A 的資源。如果您想要授予 Amazon 服務擔任該角色的許可，則信任政策中的主體也可以是 Amazon 服務主體。

  如需使用 IAM 來委派許可的相關資訊，請參閱《IAM 使用者指南》中的存取管理。

下列是授予許可給 `kinesisanalytics:CreateApplication` 動作的範例政策，此為建立應用程式的必要動作。

> **Note**
>
> 此為簡介範例政策。當您將原則附加至使用者時，使用者將能夠使用 AWS CLI 或 AWS SDK 建立應用程式。但是使用者需要更多許可來配置輸入和輸出。此外，使用者在使用主控台時需要更多許可。後面的章節提供了更多資訊。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
```

```
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

如需搭配 使用身分識別型政策的詳細資訊，請參閱 針對 使用以身分為基礎的政策 (IAM 政策) 。如需使用者、群組、角色和許可的詳細資訊，請參閱《IAM 使用者指南》中的身分 (使用者、群組和角色)。

資源型政策

其他服務 (例如 Amazon S3) 也支援以資源為基礎的許可政策。例如，您可以附加政策到 S3 儲存貯體，以管理存取許可到該儲存貯體。 不支援以資源為基礎的政策。

指定政策元素：動作、效果和委託人

對於每項資源，該服務都會定義一組 API 操作。定義一組您可在政策中指定的動作，以授予這些 API 操作的許可。為了執行 API 操作，某些 API 操作可能需要多個動作的許可。如需資源與 API 操作的詳細資訊，請參閱 資源與操作 與 動作。

以下是最基本的政策元素：

- 資源：您使用 Amazon Resource Name (ARN) 識別欲套用政策的資源。如需詳細資訊，請參閱 資源與操作。
- 動作：使用動作關鍵字識別您要允許或拒絕的資源操作。例如，您可以使用 `create` 來允許使用者建立應用程式。
- 效果：您可以指定使用者請求特定動作時會有什麼效果 (允許或拒絕)。如果您未明確授予存取 (允許) 資源，則隱含地拒絕存取。您也可以明確拒絕資源存取，這樣做可確保使用者無法存取資源，即使不同政策授予存取也是一樣。
- 委託人：在身分識別型政策 (IAM 政策) 中，政策所連接的使用者就是隱含委託人。對於以資源為基礎的政策，您可以指定想要收到許可的使用者、帳戶、服務或其他實體 (僅適用於以資源為基礎的政策)。 不支援以資源為基礎的政策。

如需進一步了解有關 IAM 政策語法和說明的詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策參考。

如需顯示所有 API 操作與其適用資源的清單，請參閱 API 許可：動作、許可與資源參考。

在政策中指定條件

當您授予許可時，可以使用存取政策語言來指定政策應該何時生效的條件。例如，建議只在特定日期之後套用政策。如需使用政策語言指定條件的詳細資訊，請參閱IAM 使用者指南中的條件。

欲表示條件，您可以使用預先定義的條件金鑰。沒有 特定的條件金鑰。但是，您可以根據需要使用AWS寬條件鍵。如需完整的 AWS全金鑰清單，請參閱《IAM 使用者指南》中的條件可用金鑰。

針對 使用以身分為基礎的政策 (IAM 政策)

以下是以身分為基礎的政策範例，此範例會示範帳戶管理員如何將許可政策附加至 IAM 身分 (即使用者、群組和角色)，並藉此授予許可，以對資源執行操作。

> **Important**
>
> 建議您先檢閱介紹主題，理解可用來管理 資源存取的基本概念和選項。如需詳細資訊，請參閱管理您的 資源之存取許可的概觀。

主題

- 使用主控台所需的許可
- 之 Amazon 受管 (預先定義) 政策
- 客戶受管政策範例

以下顯示許可政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1473028104000",
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:CreateApplication"
            ],
            "Resource": [
```

```
                "*"
            ]
        }
    ]
}
```

此政策具有一個陳述式：

- 第一個陳述式透過使用應用程式的 Amazon Resource Name (ARN)，來授與資源上的一個動作 (`kinesisanalytics:CreateApplication`) 許可。在這個案例中，ARN 指定萬用字元 (*) 來表示為任何資源授予許可。

如需顯示所有 API 操作與其適用資源的資料表，請參閱 API 許可：動作、許可與資源參考。

使用主控台所需的許可

如要讓使用者在主控台上作業，您必須授予必要的許可。舉例來說，如果您希望使用者擁有建立應用程式的許可，請授與許可向他們展示帳戶中的串流來源，以便使用者可以在主控台上設定輸入和輸出。

我們建議下列作法：

- 使用 Amazon 受管政策授予使用者許可。如需可用政策，請參閱 之 Amazon 受管 (預先定義) 政策。
- 建立自訂政策。在此情況下，我們建議您檢閱本節中提供的範例。如需詳細資訊，請參閱 客戶受管政策範例。

之 Amazon 受管 (預先定義) 政策

AWS 透過提供由建立和管理的獨立 IAM 政策來解決許多常見使用案例 AWS。這些 Amazon 受管政策會授予常用案例所需的許可，讓您不需調查需要哪些許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 Amazon 受管政策。

您可將下列 Amazon 受管政策附加到帳戶中的使用者，這些政策專屬於：

- **`AmazonKinesisAnalyticsReadOnly`**：授與許可給可讓使用者列出應用程式和檢閱輸入/輸出組態的動作。它也會授與允許使用者檢視 Kinesis 串流和 Firehose 傳遞串流清單的權限。應用程式正在執行時，使用者可以在主控台中檢視來源資料和即時分析結果。

- **AmazonKinesisAnalyticsFullAccess**：授與許可給所有可讓使用者建立和管理應用程式的動作和其他許可。但是，請注意以下內容：

  - 這些許可不足以讓使用者想要在主控台中建立新的 IAM 角色 (僅能允許使用者選取現有角色)。如果您希望讓使用者在主控台中建立 IAM 角色，請新增 IAMFullAccess Amazon 受管政策。

  - 設定應用程式時，使用者必須具有iam:PassRole 動作的許可，才能指定 IAM 角色。此 Amazon 受管政策只會針對字首為 service-role/kinesis-analytics 的 IAM 角色授予使用者 iam:PassRole 動作許可。

    如果使用者想用沒有此字首的角色來設定應用程式，您必須先明確授與使用者對該角色之 iam:PassRole 動作許可。

您也可以建立自訂 IAM 政策，以允許動作與資源的許可。您可以將這些自訂政策連接至需要這些許可的使用者或群組。

客戶受管政策範例

本節的範例提供一組範本政策可供您連接到使用者。如果您在建立政策方面是新手，我們建議您先在自己的帳戶中建立使用者。然後依序將策略附加到使用者，如本節所述步驟。當您將政策連接到使用者時，即可使用主控台來驗證每個政策的效果。

起初，使用者沒有許可且無法在主控台進行任何操作。隨著您將政策連接到使用者，便可以驗證使用者在主控台上能夠執行各種動作。

我們建議您使用兩個瀏覽器視窗。在一個視窗中，建立使用者並授予許可。另一方面， AWS Management Console 使用用戶的憑據登錄，並在授予他們時驗證權限。

如需相關範例，以了解如何建立可擔任應用程式之執行角色的 IAM 角色，請參閱《IAM 使用者指南》中的建立 IAM 角色。

範例步驟

- 步驟 1：建立 IAM 使用者
- 步驟 2：允許非特定動作的使用者許可
- 步驟 3：允許使用者查看應用程式清單與詳細資訊

- 步驟 4：允許使用者啟動特定應用程式
- 步驟 5：允許使用者建立應用程式
- 步驟 6：允許應用程式使用 Lambda 預處理

步驟 1：建立 IAM 使用者

首先，您需要建立 IAM 使用者，並將使用者新增至擁有管理許可的 IAM 群組，然後將管理許可授與您所建立的 IAM 使用者。然後，您可以 AWS 使用特殊的 URL 和該用戶的憑據進行訪問。

如需說明，請前往《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。

步驟 2：允許非特定動作的使用者許可

首先，針對使用者在使用應用程式時所需的所有非特定動作，授予使用者許可。其中包括使用串流的許可 (Amazon Kinesis Data Streams 動作、Amazon 資料 Firehose 動作)，以及動作的 CloudWatch 許可。將下列政策附加到使用者。

您必須提供要授與 `iam:PassRole` 許可的 IAM 角色名稱，或指定萬用字元 (*) 表示所有 IAM 角色來更新政策。這不是安全的做法；但您在此測試期間可能沒有建立特定的 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"
        }
    ]
}
```

步驟 3：允許使用者查看應用程式清單與詳細資訊

以下政策會授予使用者下列許可：

- kinesisanalytics:ListApplications 動作的許可，以便使用者檢視應用程式清單。此為服務層級的 API 呼叫，因此要指定 "*" 做為 Resource 值。
- kinesisanalytics:DescribeApplication動作的許可，以便您取得有關任何應用程式的資訊。

將此政策新增至使用者。

```
{
    "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/*"
        }
    ]
}
```

用使用者憑證登入主控台，以驗證這些許可。

步驟 4：允許使用者啟動特定應用程式

如果您希望使用者能夠啟動其中一個現有的應用程式，請將下列政策附加至使用者。此政策會提供 kinesisanalytics:StartApplication 動作的許可。您必須通過提供您的帳戶 ID， AWS 地區和應用程序名稱來更新保單。

```
{
    "Version": "2012-10-17",
    "Statement": [
                {
            "Effect": "Allow",
            "Action": [
                "kinesisanalytics:StartApplication"
            ],
            "Resource": "arn:aws:kinesisanalytics:aws-region:aws-account-
id:application/application-name"
        }
    ]
}
```

步驟 5：允許使用者建立應用程式

如果您希望使用者建立應用程式，可將下列政策附加至使用者。您必須更新政策，並提供 [AWS 地區]、您的帳戶 ID，以及您要讓使用者建立的特定應用程式名稱，或提供 [*]，讓使用者可以指定任何應用程式名稱 (從而建立多個應用程式)。

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

步驟 6：允許應用程式使用 Lambda 預處理

如果您希望應用程式能夠使用 Lambda 預先處理，請將下列政策附加至該角色。

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

```
            "lambda:GetFunctionConfiguration"
        ],
        "Resource": "<FunctionARN>"
    }
```

API 許可：動作、許可與資源參考

當您設定存取控制並撰寫可連接至 IAM 身分 (身分類型政策) 的許可政策時，可以參考下列清單。此單包括每個 API 作業、您可以授與執行動作權限的對應動作，以及您可以授與權限的 AWS 資源。您在政策的 `Action` 欄位中指定動作，然後在政策的 `Resource` 欄位中指定資源值。

您可以在原則中使用 AWS寬條件金鑰來表示條件。如需完 AWS整金鑰清單，請參閱 IAM 使用者指南中的可用金鑰。

> **Note**
>
> 若要指定動作，請使用後接 API 操作名稱的 `kinesisanalytics` 字首 (例如，`kinesisanalytics:AddApplicationInput`)。

API 和所需的動作許可

API 操作：

所需許可 (API 動作)：

資源：

API 和所需的動作許可

Amazon RDS API 和動作所需的許可

API 操作：AddApplicationInput

動作：`kinesisanalytics:AddApplicationInput`

資源：

`arn:aws:kinesisanalytics:` *`region`*`:`*`accountId`*`:application/`*`application-name`*

GetApplicationState

主控台使用名為 `GetApplicationState` 的內部方法來取樣或存取應用程式資料。您的服務應用程式需要具有內部 `kinesisanalytics:GetApplicationState` API 的許可，才能透過 AWS Management Console取樣或存取應用程式資料。

監控

提供監控功能給您的應用程式。如需詳細資訊，請參閱 監控。

Amazon Kinesis Data Analytics for SQL 應用程式的合規驗證

第三方稽核員會評估 Amazon Kinesis Data Analytics 的安全性和合規性，做為多個 AWS 合規計劃的一部分。這些包括 SOC、PCI、HIPAA 等。

如需特定合規計劃範圍內的 AWS 服務清單，請參閱合規計劃的 Amazon 服務範圍。如需一般資訊，請參閱 AWS 合規計劃。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告 AWS Artifact。

您使用 Kinesis Data Analytics 時的合規責任，取決於資料的機密性、您公司的合規目標及適用法律和法規。若您使用 Kinesis Data Analytics 時必須遵循特定標準，如 HIPAA 或 PCI， AWS 會提供資源予以協助：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。 AWS
- 建構 HIPAA 安全性與合規性白皮書 — 本白皮書說明公司如何使用建立符合 HIPAA 標準的應用 AWS 程式。
- AWS 合規資源 — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS Config— 此 AWS 服務評估您的資源配置是否符合內部實踐，行業準則和法規。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

Amazon Kinesis Data Analytics 中的恢復能力

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您所設計與操作的應用

程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎架構之外，Kinesis Data Analytics 還提供多種功能，協助支援您的資料恢復能力和備份需求。

災難復原

Kinesis Data Firehose 會在無伺服器模式中執行，並透過執行自動遷移，來處理主機降級、可用區域可用性和其他基礎設施相關的問題。發生這種情況時，Kinesis Data Analytics 可確保應用程式的處理過程沒有遺失任何資料。如需詳細資訊，請參閱 將應用程式輸出保存至外部目標的交付模型。

Kinesis Data Analytics for SQL 應用程式中的基礎設施安全

作為受管服務，Amazon Kinesis Data Analytics 受到 Amazon 網路服務：安 AWS 全程序概觀白皮書中所述的全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫透過網路存取 Kinesis Data Analytics。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

適用於 Kinesis Data Analytics 的安全最佳實務

在您開發和實作自己的安全政策時，可考慮使用 Amazon Kinesis Data Analytics 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

使用 IAM 角色存取其他 Amazon 服務

您的 Kinesis Data Analytics 應用程式必須具有有效的登入資料，才能存取其他服務中的資源，例如 Kinesis 資料串流、Firehose 交付串流或 Amazon S3 儲存貯體。您不應將 AWS 登入資料直接存放在應用程式或 Amazon S3 儲存貯體中。這些是不會自動輪換的長期憑證，如果遭到盜用，可能會對業務造成嚴重的影響。

反之，您應使用 IAM 角色為應用程式管理暫時性憑證，以存取其他資源。使用角色時，您不必使用長期憑證來存取其他資源。

如需詳細資訊，請參閱《IAM 使用者指南》中的以下主題：

- IAM 角色
- 常見的角色方案：使用者、應用程式和服務

在相依資源實作伺服器端加密

Kinesis Data Analytics 中的靜態資料和傳輸中的資料都會進行加密，而且無法停用此加密。您應該在相依資源中實作伺服器端加密，例如 Kinesis 資料串流、Firehose 交付串流和 Amazon S3 儲存貯體。如需關於在相依資源中實作伺服器端加密的詳細資訊，請參閱 資料保護。

用 CloudTrail 於監控 API 呼叫

Kinesis Data Analytics 與 Kinesis 資料分析中提供使用者 AWS CloudTrail、角色或 Amazon 服務所採取的動作記錄的服務整合在一起。

使用收集的資訊 CloudTrail，您可以判斷向 Kinesis Data Analytics 提出的請求、提出請求的來源 IP 位址、提出請求的人員、提出請求的時間以及其他詳細資訊。

如需詳細資訊，請參閱 the section called “使用 AWS CloudTrail”。

監控 SQL 應用程式

監控是維護應用程式可靠性、可用性與效能的重要環節。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。不過，開始監控 之前，您應該建立監控計劃，其中回答下列問題：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

下一步是在各個時間點和不同的負載條件下測量效能，以在您的環境中確立 正常效能的基準。在監控時，您可儲存歷史監控資料。藉此，您才能與目前的效能資料做比較、辨識正常效能模式和效能異常狀況、並規劃問題處理方式。

藉此，您可以監視應用程式。應用程式會處理資料串流 (輸入或輸出)，這兩者都包含識別碼，您可以使用這些識別碼縮小 CloudWatch 記錄的搜尋範圍。有關如何處理資料串流的詳細資訊，請參閱 Amazon Kinesis Data Analytics for SQL 應用程式：運作方式。

最重要的指標是 `millisBehindLatest`，表示應用程式從串流來源讀取的落後程度。在典型情況下，後面的毫秒應等於或接近零。出現短暫尖峰是正常的，此時 `millisBehindLatest` 會增加。

我們建議您設定 CloudWatch 警示，在應用程式落後超過一個小時讀取串流來源時觸發警示。針對需要非常接近即時處理的某些使用案例，例如將處理的資料傳送到即時應用程式，您可以選擇將提醒設定為較低的值，例如五分鐘。

主題

- 監控工具
- 使用 Amazon 監控 CloudWatch
- 使用 AWS CloudTrail 記錄 API 呼叫

監控工具

AWS 提供了可用於監視的各種工具。您可以設定其中一些工具來進行監控，但有些工具需要手動介入。建議您盡可能自動化監控任務。

自動化監控工具

您可以使用下列自動化監控工具來監看，並在發生錯誤時進行回報：

- Amazon A CloudWatch lar ps — 觀看您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。 CloudWatch 警示不會僅因為處於特定狀態而叫用動作；狀態必須已變更並維持指定數目的期間。如需詳細資訊，請參閱 使用 Amazon 監控 CloudWatch。
- Amazon CloudWatch 日誌 — 監控、存放和存取來自 AWS CloudTrail 或其他來源的日誌檔。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的監控日誌檔。
- Amazon E CloudWatch vents — 匹配事件並將其路由到一個或多個目標函數或串流，以進行變更、擷取狀態資訊並採取糾正措施。有關更多信息，請參閱 Amazon 用 CloudWatch 戶指南中的 Amazon CloudWatch 事件是什麼。
- AWS CloudTrail 防護記錄監控 — 在帳戶之間共用記 CloudTrail 錄檔、即時監控記錄檔案，方法是將CloudWatch 記錄檔傳送至記錄檔、以 Java 撰寫記錄處理應用程式，以及驗證記錄檔在傳送之後是否未變更 CloudTrail。若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 記錄檔〉。

手動監控工具

監視的另一個重要部分是手動監視 CloudWatch 警報未涵蓋的項目。、 CloudWatch Trusted Advisor、和其他 AWS Management Console 儀表板可提供您 AWS 環境狀態的 at-a-glance 檢視。

- CloudWatch 首頁會顯示下列內容：
  - 目前警示與狀態
  - 警示與資源的圖表
  - 服務運作狀態

  此外，您可以使用執行 CloudWatch 以下操作：
  - 建立自定儀表板來監控您注重的服務

  - 用於疑難排解問題以及探索驅勢的圖形指標資料。
  - 搜尋與瀏覽您所有的指標
  - 建立與編輯要通知發生問題的警示
- AWS Trusted Advisor 可以幫助您監控以提高性能，可靠性，安全性和成本效益。所有使用者均可使用四項 Trusted Advisor 檢查。具有商業或企業支援計畫的使用者可以使用超過 50 項以上的檢查。如需詳細資訊，請參閱 AWS Trusted Advisor。

使用 Amazon 監控 CloudWatch

您可以使用 Amazon 監控應用程序 CloudWatch。 CloudWatch 將原始資料收集並處理成可讀、近乎即時的指標。這些統計數字會保留兩週。您可以存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。根據預設，量度資料會自動傳送至 CloudWatch。有關更多信息，請參閱什麼是 Amazon CloudWatch？ 在 Amazon 用 CloudWatch 戶指南。

主題

- 指標與維度
- 檢視指標和維度
- 建立要監視的 CloudWatch 警示
- 使用 Amazon CloudWatch 日誌

指標與維度

AWS/KinesisAnalytics 命名空間包含下列指標。

指標	描述
Bytes	讀取（每輸入流）或寫入（每輸出流）的位元組數目。 級別：每輸入流和每輸出流
KPUs	用來執行串流處理應用程式的 Kinesis 處理單元數目。每小時使用的 KPU 平均數量會決定應用程式的計費方式。

指標	描述
	層次：應用程式層次
MillisBehindLatest	指出應用程式從串流來源讀取的時間落後目前時間之程度。 層次：應用程式層次
Records	讀取（每輸入流）或寫入（每輸出流）的紀錄數目。 級別：每輸入流和每輸出流
Success	在嘗試交付至您的應用程式設定之目的地時，每個成功交付為 1；每次失敗的交付嘗試為 0。此指標的平均值表示執行成功交付的次數。 層次：每目的地。
InputProcessing.Duration	執行的每個 AWS Lambda 函數調用所花費的時間。 級別：每輸入流
InputProcessing.OkRecords	Lambda 函數傳回，且標有 Ok 狀態的記錄數。 級別：每輸入流
InputProcessing.OkBytes	Lambda 函數傳回，且標有 Ok 狀態的記錄位元組總和。 級別：每輸入流
InputProcessing.DroppedRecords	Lambda 函數傳回，且標有 Dropped 狀態的記錄數。 級別：每輸入流
InputProcessing.ProcessingFailedRecords	Lambda 函數傳回，且標有 ProcessingFailed 狀態的記錄數。 級別：每輸入流

指標	描述
InputProcessing.Success	後者的 Lambda 調用成功次數。 級別：每輸入流
LambdaDelivery.OkRecords	Lambda 函數傳回，且標有 Ok 狀態的記錄數。 層級：每 Lambda 目的地
LambdaDelivery.DeliveryFailedRecords	Lambda 函數傳回，且標有 DeliveryFailed 狀態的記錄數。 層級：每 Lambda 目的地
LambdaDelivery.Duration	後者執行的每個 Lambda 函數調用所需的時間。 層級：每 Lambda 目的地

提供指標給下列維度。

維度	描述
Flow	每輸入流：輸入 每輸出流：輸出
Id	每輸入流：輸入 ID 每輸出流：輸出 ID

檢視指標和維度

當您的應用程式處理資料串流時，會將下列指標和維度傳送至 CloudWatch。您可以使用下列程序來檢視後者的指標。

在主控台中，指標會先依服務命名空間分組，再依各命名空間內不同的維度組合分類。

若要使用 CloudWatch 主控台檢視指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 在的「按類別分類的CloudWatch 測量結果」窗格中，選擇測量結果類別。
4. 在上方窗格中，向下捲動以檢視完整指標清單。

若要使用 AWS CLI

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

指標會在下列層級收集：

- 應用程式
- 輸入串流
- 輸出串流

建立要監視的 CloudWatch 警示

您可以建立 Amazon CloudWatch 警示，在警示狀態變更時傳送 Amazon SNS 訊息。警示會在您指定的期間，監看單一指標。警示會根據在數個期間與指定閾值相關的指標值，來執行一個或多個動作。此動作是傳送到 Amazon SNS 主題或 Auto Scaling 政策的通知。

警示僅會針對持續狀態變更調用動作。若要讓 CloudWatch 警示叫用動作，狀態必須已變更並維持一段指定的時間。

您可以使用 AWS Management ConsoleCloudWatch AWS CLI、或 CloudWatch API 設定警報，如下所述。

使用 CloudWatch 主控台設定鬧鐘

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 選擇建立警示。建立警示精靈隨即出現。

3. 選擇 Kinesis 分析指標。捲動指標，以找到您要設定警示的指標。

   若僅要顯示指標，請在您的檔案系統搜尋檔案系統 ID。選擇要建立警示的指標，然後選擇 下一步。

4. 輸入指標的名稱、描述和每當值。

5. 如果您想要 CloudWatch 在到達鬧鐘狀態時傳送電子郵件給您，請在「每當此警報：」欄位中選擇「狀態為鬧鐘」。在傳送通知至： 欄位中，選擇現有的 SNS 主題。如果您選取建立主題，即可為新電子郵件訂閱清單設定名稱和電子郵件地址。此清單會儲存並顯示在欄位中供未來警示使用。

   > **Note**
   >
   > 如果您使用建立主題來建立新的 Amazon SNS 主題，電子郵件地址必須先經過驗證才會接收通知。電子郵件只有在警示進入警示狀態時才會傳送。如果此警示狀態在驗證電子郵件地址之前發生變更，就不會收到通知。

6. 在警示預覽區段中，預覽您即將建立的警示。

7. 選擇建立警示以建立警示。

若要使用 CloudWatch CLI 設定警示

- 呼叫 `mon-put-metric-alarm`。如需詳細資訊，請參閱 Amazon CloudWatch CLI 參考資料。

若要使用 CloudWatch API 設定警示

- 呼叫 `PutMetricAlarm`。如需詳細資訊，請參閱 Amazon CloudWatch API 參考資料。

使用 Amazon CloudWatch 日誌

如果應用程式設定錯誤，則可能會在啟動期間轉換為執行中狀態。或也可能會更新，但不將任何資料處理到應用程序內輸入流中。藉由將 CloudWatch 記錄選項新增至應用程式，您可以監視應用程式設定問題。

在下列情況下會產生組態錯誤：

- 用於輸入的 Kinesis 資料串流不存在。
- 用於輸入的 Amazon 數據 Firehose 交付流不存在。
- 用作參考資料來源的 Amazon S3 儲存貯體不存在。

- S3 儲存貯體中參考資料來源指定的檔案不存在。
- 管理相關許可的 AWS Identity and Access Management (IAM) 角色中未定義正確的資源。
- 管理相關許可的 IAM 角色中未定義正確的資源。
- 沒有權限擔任管理相關許可的 IAM 角色。

有關 Amazon 的更多信息 CloudWatch，請參閱 Amazon CloudWatch 用戶指南。

新增原 PutLogEvents 則動作

需要權限才能將錯誤配置錯誤寫入。 CloudWatch您可以將這些許可新增至擔任的 IAM 角色，如下所述。如需使用 IAM 角色的詳細資訊，請參閱 Kinesis Data Analytics 中的 Identity and Access Management。

信任政策

若要授與許可來擔任 IAM 角色，您可以將以下信任政策附加到角色。

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

許可政策

若要授與應用程式許可以 CloudWatch 從資源寫入日誌事件，您可以使用下列 IAM 許可政策。

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

新增組態錯誤監視

使用下列 API 動作將 CloudWatch 記錄選項新增至新的或現有的應用程式，或變更現有應用程式的記錄選項。

> **Note**
>
> 您目前只能使用 API 動作將 CloudWatch 記錄選項新增至應用程式。您無法使用主控台新增 CloudWatch 記錄選項。

建立應用程式時新增 CloudWatch 記錄選項

下列程式碼範例示範如何在建立應用程式時使用`CreateApplication`動作來新增 CloudWatch log 選項。如需 `Create_Application` 的詳細資訊，請參閱CreateApplication。

```
{
   "ApplicationCode": "<The SQL code the new application will run on the input
 stream>",
   "ApplicationDescription": "<A friendly description for the new application>",
   "ApplicationName": "<The name for the new application>",
   "Inputs": [ ... ],
   "Outputs": [ ... ],
   "CloudWatchLoggingOptions": [{
      "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add
 to the new application>",
      "RoleARN": "<ARN of the role to use to access the log>"
 }]
}
```

將 CloudWatch 記錄選項新增至現有的應用程式

下列程式碼範例將示範如何使用AddApplicationCloudWatchLoggingOption動作，將 CloudWatch log 選項加入至現有的應用程式。如需 AddApplicationCloudWatchLoggingOption 的相關資訊，請參閱 AddApplicationCloudWatchLoggingOption。

```
{
   "ApplicationName": "<Name of the application to add the log option to>",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "<ARN of the log stream to add to the application>",
      "RoleARN": "<ARN of the role to use to access the log>"
   },
   "CurrentApplicationVersionId": <Version of the application to add the log to>
}
```

更新現有的 CloudWatch 記錄選項

下列程式碼範例將示範如何使用UpdateApplication動作來修改現有的 CloudWatch log 選項。如需 UpdateApplication 的相關資訊，請參閱 UpdateApplication。

```
{
   "ApplicationName": "<Name of the application to update the log option for>",
   "ApplicationUpdate": {
      "CloudWatchLoggingOptionUpdates": [
         {
            "CloudWatchLoggingOptionId": "<ID of the logging option to modify>",
            "LogStreamARNUpdate": "<ARN of the new log stream to use>",
            "RoleARNUpdate": "<ARN of the new role to use to access the log stream>"
         }
      ],
   },
   "CurrentApplicationVersionId": <ID of the application version to modify>
}
```

從應用程式刪除 CloudWatch 記錄選項

下列程式碼範例將示範如何使用DeleteApplicationCloudWatchLoggingOption動作刪除現有的 CloudWatch log 選項。如需 DeleteApplicationCloudWatchLoggingOption 的相關資訊，請參閱 DeleteApplicationCloudWatchLoggingOption。

```
{
   "ApplicationName": "<Name of application to delete log option from>",
   "CloudWatchLoggingOptionId": "<ID of the application log option to delete>",
   "CurrentApplicationVersionId": <Version of the application to delete the log option
 from>
}
```

組態錯誤

以下各節包含您可能在 Amazon CloudWatch Logs 中從設定錯誤的應用程式中看到的錯誤的詳細資訊。

錯誤訊息格式

應用程式設定錯誤所產生的錯誤訊息採用下列格式。

```
{
"applicationARN": "string",
"applicationVersionId": integer,
"messageType": "ERROR",
"message": "string",
"inputId": "string",
"referenceId": "string",
"errorCode": "string"
"messageSchemaVersion": "integer",
}
```

錯誤訊息中的欄位包含下列資訊：

- `applicationARN`：產生應用程式的 Amazon Resource Name (ARN)，例如：`arn:aws:kinesisanalytics:us-east-1:112233445566:application/sampleApp`
- `applicationVersionId`：遇到錯誤時的應用程式版本。如需詳細資訊，請參閱 ApplicationDetail。
- `messageType`：訊息類型。目前，這種類型只能為 `ERROR`。
- `message`：錯誤的詳細資訊，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`：與應用程式輸入相關聯的 ID。此值僅在此輸入是錯誤的原因時才會存在。如果有 `referenceId`，則此值不會存在。如需詳細資訊，請參閱 DescribeApplication。
- `referenceId`：與應用程式參考資料來源相關聯的 ID。此值僅在輸入是錯誤的原因時才會存在。如果有 `inputId`，則此值不會存在。如需詳細資訊，請參閱 DescribeApplication。
- `errorCode`：錯誤的識別符。此識別符為 `InputError` 或 `ReferenceDataError`。
- `messageSchemaVersion`：指定目前訊息結構描述版本的值，現為 1。您可以檢查此值，以查看錯誤訊息結構描述是否已更新。

錯誤

CloudWatch 記錄檔中可能出現的錯誤包括以下內容。

資源不存在

如果針對不存在的 Kinesis 輸入串流指定 ARN，但 ARN 在語法上正確，則會產生類似下列的錯誤。

```
{
"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"
}
```

如果參考資料使用不正確的 Amazon S3 檔案金鑰，就會產生類似下列的錯誤。

```
{
 "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

如果針對不存在的 IAM 輸入角色指定 ARN，但 ARN 在語法上正確，則會產生類似下列的錯誤。

```
{
 "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"
}
```

角色沒有存取資源的許可。

如果使用的輸入角色無權存取輸入資源 (例如 Kinesis 來源串流)，則會產生類似下列內容的錯誤。

```
{
 "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"
}
```

使用 AWS CloudTrail 記錄 API 呼叫

已與 AWS CloudTrail 整合，這項服務可提供由使用者、角色或 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 的所有 API 呼叫擷取為事件。擷取的呼叫包括從 主控台進行的呼叫，以及針對 API 操作的程式碼呼叫。如果您建立追蹤，就可以將 CloudTrail 事件持續交付到 Amazon S3 儲存貯體，包括 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新事件。您可以利用 CloudTrail 所收集的資訊來判斷向 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，請參閱AWS CloudTrail《使用者指南》。

CloudTrail 中的資訊

當您建立帳戶時，系統即會在 AWS 帳戶中啟用 CloudTrail。當 中發生活動時，該活動會記錄在 CloudTrail 事件中，其他 AWS 服務事件則記錄於事件歷史記錄中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

如需您 AWS 帳戶中正在進行事件的記錄 (包含 的事件)，請建立追蹤。追蹤能讓 CloudTrail 將日誌檔交付至 Amazon S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 會記錄所有動作，並記載於 API 參考中。例如，對 `CreateApplication` 以及 `UpdateApplication` 動作發出的呼叫會在 CloudTrail 日誌檔案中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用 AWS 帳戶根使用者 或使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity Element。

了解日誌檔項目

追蹤是一種組態，可讓事件以日誌檔案的形式交付至您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例為展示了 AddApplicationCloudWatchLoggingOption 和 DescribeApplication 的 CloudTrail 日誌項目。

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

限制

使用 Amazon Kinesis Data Analytics for SQL 應用程式時，請注意以下限制：

- Kinesis Data Analytics for SQL 支援下列 AWS 區域：美國東部 (俄亥俄)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、加拿大 (中部)、歐洲 (巴黎)、歐洲 (愛爾蘭)、歐洲 (法蘭克福)、歐洲 (倫敦)、亞太區域 (香港)、亞太區域 (孟買)、亞太區域 (雪梨)、亞太區域 (新加坡)、亞太區域 (首爾)、亞太區域 (東京)、南美洲 (聖保羅)、AWS GovCloud (美國東部) 和 AWS GovCloud (美國西部)。我們沒有計劃在其他 AWS 區域開放 Kinesis Data Analytics for SQL。
- 在 2023 年 6 月 28 日之後，如果您尚未使用 Kinesis Data Analytics for SQL，您將無法使用 AWS 管理主控台建立 Kinesis Data Analytics for SQL 應用程式。如果您在 2023 年 6 月 28 日之前建立了 Kinesis Data Analytics for SQL 應用程式，即代表您已在 AWS 使用 Kinesis Data Analytics for SQL，建立與執行應用程式的方式不會改變。不過，在您未使用 Kinesis Data Analytics for SQL 的區域，您無法再以 AWS 建立應用程式。
- 2023 年 9 月 12 日之後，如果尚未使用 Kinesis Data Analytics for SQL，您將無法使用 Kinesis Data Firehose 做為建立新應用程式的來源。搭配使用 Kinesis Data Analytics for SQL 應用程式與 `KinesisFirehoseInput` 的現有客戶，可以繼續使用 `KinesisFirehoseInput`，在使用 Kinesis Data Analytics 的現有帳戶中新增應用程式。如果您是現有客戶，並且希望使用 Kinesis Data Analytics for SQL 與 `KinesisFirehoseInput` 建立新帳戶，您可以開立支援案例。如需詳細資訊，請參閱 AWS Support 中心。
- 應用程式內串流中的資料列大小限制為 512 KB。Kinesis Data Analytics 最多使用 1 KB 來儲存中繼資料。此中繼資料會計入資料列限制。
- 應用程式中的 SQL 程式碼限制為 100 KB。
- 我們建議窗口查詢的最長窗口是一小時。應用程式內串流會儲存在揮發性儲存體中，意外的應用程式中斷會導致應用程式從揮發性儲存體中的來源資料重建串流。
- 針對單一應用程式內串流，我們建議的最大輸送量介於每秒 2 到 20 MB 之間，視應用程式查詢的複雜程度而定。

- 您可以在帳戶中為每個 AWS 區域建立最多 50 個 Kinesis Data Analytics 應用程式。您可以透過服務配額提高表單來建立案例，以請求額外的應用程式。如需詳細資訊，請參閱 AWS Support 中心。

- 單一 Kinesis Data Analytics for SQL 可處理的最大串流輸送量約為每秒 100 MB。這假設您已將應用程式內串流的數目增加到 64 個上限，而且您已將 KPU 限制增加到 8 以上 (如需詳細資訊，請參閱下列限制)。如果您的應用程式需要處理每秒 100 MB 以上的輸入，請執行下列其中一項操作：
  - 針對 SQL 應用程式使用多個 Kinesis Data Analytics 來處理輸入
  - 如果您想要繼續使用單一串流和應用程式，請使用 Managed Service for Apache Flink for Java 應用程式。

  > **Note**
  >
  > 我們建議您定期檢閱應用程式的 `InputProcessing.0kBytes` 指標，以便在應用程式的預計輸入輸送量超過每秒 100 MB時，事先規劃使用多個 SQL 應用程式，或移轉至 Managed Service for Apache Flink for Java 應用程式。我們也建議您建立 `InputProcessing.0kBytes` 的 CloudWatch 提醒，以便在應用程式接近輸入輸送量限制時收到通知。此舉很有用，因為您可以更新應用程式查詢來權衡取得更高的輸送量，從而避免了分析的背壓和延遲。如需詳細資訊，請參閱疑難排解。如果您有減少上游輸送量的機制，提醒也會很有用。

- Kinesis 處理單元 (KPU) 的數目限制為八。如需如何請求提高此限制的指示，請參閱 Amazon 服務配額中的請求提高限制。

  使用 Kinesis Data Analytics 時，您僅需按使用量付費。我們會根據您執行串流處理應用程式所使用的 KPU 數目平均，以小時費率計費。單一 KPU 可為您提供 1 個 vCPU 和 4 GB 的記憶體。

- 每個應用程式可以有一個串流來源和最多一個參考資料來源。

- 您最多可以為 Kinesis Data Analytics 應用程式設定三個目的地。建議您使用其中一個目的地來保留應用程式內錯誤串流資料。

- 存放參考資料的 Amazon S3 物件大小最多可達 1 GB。

- 如果在上傳參考資料到應用程式內表格後，您變更了存放在 S3 儲存貯體中的參考資料，則需要使用 UpdateApplication 作業 (使用 API 或 AWS CLI) 來重新整理應用程式內表格中的資料。目前，AWS Management Console 不支援在應用程式中重新整理參考資料。

- 目前 Kinesis Data Analytics 不支援 Amazon Kinesis Producer Library (KPL) 產生的資料。

- 您可以為每個應用程式指派最多 50 個標籤。

最佳實務

本節說明使用 Amazon Kinesis Data Analytics 應用程式時的最佳實務。

主題

管理應用程式

管理 Amazon Kinesis Data Analytics 應用程式時，請遵循下列最佳實務：

- 設定 Amazon CloudWatch 警示 — 您可以使用 Kinesis Data Analytics 提供的 CloudWatch 指標來監控下列項目：
  - 輸入位元組和輸入記錄（進入應用程式的位元組和記錄數）
  - 輸出位元組和輸出記錄
  - `MillisBehindLatest` (應用程式從串流來源讀取時的差距)

  我們建議您針對生產中的應用程式，針對下列指標設定至少兩個 CloudWatch 警示：
  - `MillisBehindLatest`：在大多數情況下，我們建議您將此警報的觸發條件設置為應用程式比最新資料晚 1 小時，平均為 1 分鐘。對於 end-to-end 處理需求較低的應用，您可以將其調整為較低的容差。此警報可確保您的應用程式讀取的是最新資料。

- 為避免發生 `ReadProvisionedThroughputException` 例外狀況，請將讀取同一 Kinesis 資料串流的生產應用程式數量限制為兩個。

> **Note**
>
> 在這種情況下，應用程式是指任何可以讀取串流來源的應用程式。只有 Kinesis Data Analytics 應用程式可以從 Firehose 交付串流讀取。不過，許多應用程式都可以從 Kinesis 資料串流讀取，例如 Kinesis Data Analytics 應用程式或. AWS Lambda建議的應用程式限制，指的是您設定來讀取一個串流來源的所有應用程式。

每個 Amazon Kinesis Data Analytics 應用程式每秒約讀取一次串流來源。但落後的應用程式可能會以更快的速度讀取資料來趕上進度。若要讓應用程式有足夠的輸送量來趕上進度，請限制讀取相同資料來源的應用程式數目。

- 將生產應用程式從同一個 Firehose 交付串流讀取的數量限制為一個應用程式。

  Firehose 交付串流可以寫入 Amazon S3 和亞馬 Amazon Redshift 等目的地。它也可以是 Kinesis Data Analytics 應用程式的串流來源。因此，我們建議您不要為每個 Firehose 交付串流設定一個以上的 Kinesis Data Analytics 應用程式。此舉能確保交付串流也可以傳遞至其他目的地。

擴展應用程式

透過主動增加輸入應用程式內串流預設數量 (一個)，以滿足應用程式未來的擴展需求。我們建議您根據應用程式的輸送量選擇下列語言：

- 如果您應用程式的擴展需求超過每秒 100 MB，請針使用多個串流和 Kinesis Data Analytics for SQL 應用程式。
- 如果您想要使用單一串流和應用程式，請選擇 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 我們建議您定期檢閱應用程式的 `InputProcessing.OkBytes` 指標，以便以事先規劃使用多個 SQL 應用程式，或者在應用程式預計輸入輸送量超過每秒 100 MB 時，移轉至 Managed Service for Apache Flink 應用程式。

監控應用程式

我們建議您在上建立 CloudWatch 警示，`InputProcessing.0kBytes`以便在應用程式接近輸入輸送量限制時收到通知。此舉很有用，因為您可以更新應用程式查詢來權衡取得更高的輸送量，從而避免了分析的背壓和延遲。如需詳細資訊，請參閱疑難排解。如果有減少上游輸送量的機制，此方式也很有用。

- 針對單一應用程式內的串流，我們建議的最大輸送量為每秒 2 到 20 MB，視應用程式查詢的複雜程度而定。
- 單一 Kinesis Data Analytics for SQL 應用程式可處理的最大串流輸送量約為每秒 100 MB。先決條件為您已將應用程式內串流的數目增加到上限 64，且 KPU 限制已調到 8 以上。如需詳細資訊，請參閱限制。

> **Note**
>
> 我們建議您定期檢閱應用程式的 `InputProcessing.0kBytes` 指標，以便以事先規劃使用多個 SQL 應用程式，或者在應用程式預計輸入輸送量超過每秒 100 MB 時，移轉至 Managed Service for Apache Flink 應用程式。

定義輸入結構描述

在主控台中設定應用程式輸入時，您必須先指定一個串流來源。然後，主控台會使用 Discovery API (請參閱DiscoverInputSchema)，透過在串流來源上取樣記錄來推斷結構描述。除此之外，在產生的應用程式內串流中，結構描述會定義資料行的名稱和資料類型。主控台會顯示結構描述。我們建議您用此推斷結構描述進行下列動作：

- 充分測試推斷的結構描述。探索程序只會使用串流來源上的取樣記錄來推斷結構描述。如果您的串流來源有 許多記錄類型，Discovery API 可能會錯過一個或多個記錄類型的取樣。這種情況可能會導致結構描述無法準確反映串流來源上的資料。

  當您的應用程式啟動時，這些遺漏的記錄類型可能會導致剖析錯誤。Amazon Kinesis Data Analytics 會將這些記錄傳送到應用程式內錯誤串流。若要減少這些剖析錯誤，建議您在主控台中以互動方式測試推斷的結構描述，並監視應用程式內串流是否有遺漏的記錄。

- Kinesis Data Analytics API 不支援對輸入組態中的資料行指定 NOT NULL 條件限制。如果您想要 NOT NULL 限制應用程式內串流中的資料行，請使用應用程式碼建立這些應用程式內串流。然後，您可以將資料從一個應用程式內串流複製到另一個，然後強制執行限制。

  當欄位需要值時，任何插入 NULL 值的嘗試都會導致錯誤。Kinesis Data Analytics 會將這些錯誤傳送至應用程式內錯誤串流。

- 放寬探索程序推斷的資料類型。探索程序會在串流來源上隨機抽樣記錄，並根據結果建議資料行和資料類型。我們建議您仔細檢閱這些資料，並考慮放寬這些資料類型，以涵蓋輸入中所有可能的記錄案例。這樣可減少應用程式執行時的剖析錯誤。舉例來說，如果推論的結構描述具有 SMALLINT 資料行類型，請考慮將其變更為 INTEGER。

- 在應用程式碼中使用 SQL 函數來處理任何非結構化資料或資料行。您的輸入中可能包含非結構化資料或資料行，例如日誌資料。如需範例，請參閱 範例：轉換 DateTime 值。處理此類資料的一種方法，是只用一個資料行的 VARCHAR(N) 類型定義結構描述，其中 N 是您預期在串流中看到的最大可能資料列。然後，您可以在應用程式碼讀取傳入的記錄，並使用 String 和 Date Time 函數剖析原始資料並進行結構描述。

- 請確定您已完全處理包含巢狀深度超過兩層的串流來源資料。當源數據是 JSON 時，即可能出現巢狀結構。Discovery API 會推斷結構描述，該結構描述會展平一層巢狀。針對兩個層級的巢狀，Discovery API 也會嘗試將這些層級平面化。如超過兩個巢狀層級，展平的支援就會受限。若要完全處理巢狀，您必須手動修改推斷的結構描述以符合需求。使用下列任一策略來達成此目標：

  - 使用 JSON 資料列路徑，選擇性地僅提取應用程式所需的金鑰值配對。JSON 資料列路徑會提供指標，指向您要引入應用程式的特定金鑰值配對。您可以對任何級別的巢狀執行此操作。
  - 使用 JSON 資料列路徑選擇性地提取複雜的 JSON 物件，然後在應用程式碼中，使用字串操作函式來提取所需的特定資料。

連接至輸出

我們建議每個應用程式至少有兩個輸出：

- 使用第一個目的地插入 SQL 查詢的結果。

- 使用第二個目的地插入整個錯誤串流，並透過 Firehose 交付串流將其傳送至 S3 儲存貯體。

撰寫應用程式碼

我們建議下列作法：

- 在 SQL 陳述式中，請勿指定超過一小時的時間型視窗，原因如下：
  - 有時候因為您更新了應用程式，或是基於 Kinesis Data Analytics 的內部原因，應用程式需要重新啟動。重新啟動時，必須從串流資料來源再次讀取視窗中包含的所有資料。Kinesis Data Analytics 需要一段時間才能為該視窗發出輸出。
  - 在這段時間內，Kinesis Data Analytics 必須維護與應用程式狀態相關的所有內容，包括相關資料。此舉會消耗大量的 Kinesis Data Analytics 處理單元。
- 在開發過程中，請在 SQL 陳述式中保持較小的視窗，以便更快地查看結果。將應用程式部署到生產環境時，可以視需要設定視窗大小。
- 請考慮將其分解為多個陳述式，而不是單一複雜的 SQL 陳述式，這些陳述式會在每個步驟中儲存中繼應用程式內串流。此舉可能會加快您的偵錯速度。
- 使用翻轉視窗時，我們建議您開啟兩個視窗，一個用於處理時間，另一個用於邏輯時間 (擷取時間或事件時間)。如需詳細資訊，請參閱 時間戳記和 ROWTIME 欄。

測試應用程式

在變更 Kinesis Data Analytics 應用程式的結構描述或應用程式碼時，我們建議您先使用測試應用程式來驗證變更，然後再將變更部署到生產環境。

設定測試應用程式

您可以透過主控台或使用 AWS CloudFormation 範本設定測試應用程式。使用 AWS CloudFormation 範本有助於確保您對測試應用程式和即時應用程式所做的程式碼變更一致。

設置測試應用程式時，您可以將其連接到即時資料，也可以使用要測試的模擬資料填入串流。我們建議使用兩種方法來將模擬數據填入串流：

- 使用 Kinesis 資料產生器 (KDG)。KDG 使用資料範本將隨機資料傳送至 Kinesis 串流。KDG 使用簡單，但不適合測試資料項目之間的複雜關係，例如偵測資料熱點或異常的應用程式。

- 使用自訂 Python 應用程式將更複雜的資料傳送至 Kinesis 資料串流。Python 應用程式可以產生資料項目之間的複雜關係，例如熱點或異常。如需將資料叢集傳送到資料熱點的 Python 應用程式範例，請參閱 範例：偵測串流上的熱點 (熱點功能)。

執行測試應用程式時，請使用目的地 (例如 Firehose 交付串流至 Amazon Redshift 資料庫) 來檢視結果，而不是在主控台上檢視應用程式內串流。控制台上顯示的數據是串流的取樣，並且不包含所有記錄。

測試結構描述變更

變更應用程式的輸入串流結構描述時，請使用測試應用程式來確認下列條件為真：

- 串流中的資料會強制轉換為正確的資料類型。舉例來說，請確定日期時間資料不會以字串形式擷取到應用程式中。
- 資料經過剖析並強制轉換為您想要的類型。如果發生剖析或強制錯誤，您可以在主控台上檢視錯誤，或將目的地指派給錯誤串流，並檢視目的地儲存區中的錯誤。
- 字元資料的欄位長度足夠，且應用程式並未截斷字元資料。您可以檢查目的地存放區中的資料記錄，以確認應用程式資料未遭截斷。

測試程式碼變更

您需要具備一些應用程式的領域知識，才能測試對 SQL 代碼的變更。您必須判斷需要測試哪些輸出，以及正確的輸出應該是什麼。關於修改應用程式 SQL 程式碼時要驗證的潛在問題區域，請參閱疑難排解 Amazon Kinesis Data Analytics for SQL 應用程式。

疑難排解 Amazon Kinesis Data Analytics for SQL 應用程式

以下內容可協助您對Amazon Kinesis Data Analytics for SQL 應用程式中可能遇到的問題進行疑難排解。

主題

停止的應用程式

- 什麼是停止的 Amazon Kinesis Data Analytics for SQL 應用程式？

  停止的應用程式，是我們觀察到至少三個月未處理任何記錄的應用程式。這表示客戶需要為未使用的 SQL 資源支付 Kinesis Data Analytics 費用。

- AWS 會從何時開始停止閒置的應用程式？

  AWS 將於 2023 年 11 月 14 日開始停止閒置的應用程式，並在 2023 年 11 月 21 日之前完成。我們將在該地區的辦公時間時區停止閒置的應用程式。

- 停止的 Kinesis Data Analytics for SQL 應用程式是否可重新啟動？

  是。如果需要重新啟動應用程式，可以如常操作。沒有必要提交支援票證。

- 當 AWS 停止閒置的應用程序時，我的查詢結果也會被刪除嗎？

  不會。首先，由於您的應用程式處於空閒狀態，因此不處理查詢。其次，您的查詢結果不會儲存在 Kinesis Data Analytics for SQL 中。您會為 Kinesis Data Analytics for SQL 應用程式設定接收目的地，計算的結果會傳送到該處 (如 Amazon S3 或其他資料串流)。因此，您保留資料的完整擁有權，並且在該儲存服務條款下，資料仍可擷取。

- 如果我不想停止應用程式，該怎麼辦？

  您可以寄送電子郵件給服務團隊 (kda-sql-questions@amazon.com)，要求不要在 2023 年 11 月 10 日之前停止應用程式。該電子郵件應包括您的帳戶 ID 和應用程式 ARN。

無法執行 SQL 程式碼

如果您需要弄清楚如何讓特定 SQL 陳述式正常運作，Kinesis Data Analytics 有數種可用的不同資源：

- 如需有關 SQL 陳述式的詳細資訊，請參閱 Kinesis Data Analytics for SQL 範例。本節提供您可以使用的許多 SQL 範例。
- 《Amazon Kinesis Data Analytics SQL 參考》提供編寫串流 SQL 陳述式的詳細指南。
- 如果您仍然遇到問題，我們建議您在 Kinesis Data Analytics 論壇上提出問題。

無法偵測或探索我的結構描述

在某些情況下，Kinesis Data Analytics 無法偵測或探索結構描述。在許多情況下，您仍然可以使用 Kinesis Data Analytics。

假設您的 UTF-8 編碼資料不使用分隔符號，或使用逗號分隔值 (CSV) 以外的格式的資料，或探索 API 未探索您的結構描述。在這些情況下，您可以手動定義結構描述，或使用字串操作函數來建構資料。

為了探索串流的結構描述，Kinesis Data Analytics 會隨機抽樣串流中的最新資料。如果您並未持續地將資料傳送至串流，Kinesis Data Analytics 可能無法擷取範例並偵測結構描述。如需更多詳細資訊，請參閱 在串流資料上使用結構描述探索功能。

參考資料已過期

啟動或更新應用程式時，或在服務問題造成的應用程式中斷期間，參考資料會從 Amazon Simple Storage Service (Amazon S3) 物件載入到應用程式。

對基礎 Amazon S3 物件進行更新時，不會將參考資料載入應用程式。

如果應用程式中的參考資料不是最新的，您可以按照下列步驟重新載入資料：

1. 前往 Kinesis Data Analytics 主控台，在清單中選擇應用程式名稱，然後選擇應用程式詳細資訊。
2. 選擇至 SQL 編輯器以開啟應用程式的即時分析頁面。

3. 在來源資料檢視中，選擇您的參考資料表名稱。
4. 選擇動作，同步化參考資料表。

應用程式未寫入目的地

如果未將資料寫入目的地，請檢查以下項目：

- 確認應用程式的角色具有足夠的許可來存取目的地。如需詳細資訊，請參閱 寫入 Kinesis Stream 的許可政策 或 寫入 Firehose Delivery Stream 的許可政策。
- 確認應用程式目的地設定正確，以及應用程式使用的輸出串流名稱正確。
- 檢查輸出串流的 Amazon CloudWatch 指標，看看是否正在寫入資料。如需使用 CloudWatch 指標的相關資訊，請參閱 使用 Amazon 監控 CloudWatch。
- 使用 the section called “AddApplicationCloudWatchLoggingOption” 新增 CloudWatch 日誌串流。您的應用程式會將設定錯誤寫入日誌串流。

如果角色和目的地組態看起來正確，請嘗試重新啟動應用程式，並在 InputStartingPositionConfiguration 指定 `LAST_STOPPED_POINT`。

要監控的重要應用程式運作狀態參數

為了確保您的應用程序正常運行，我們建議您監控某些重要參數。

要監控的最重要參數是 Amazon CloudWatch 指標 `MillisBehindLatest`。此指標表示您從串流讀取落後於目前的時間。此指標可協助您判斷處理來源串流記錄的速度是否不夠快。

一般而言，您應該設定 CloudWatch 警示，以便在落後超過一小時時觸發。但是，時間會依使用案例而定。您可以根據需要進行調整。

如需更多詳細資訊，請參閱 最佳實務。

運行應用程序時代碼錯誤無效

如您無法儲存並執行 Amazon Kinesis Data Analytics 應用程式的 SQL 程式碼，下列是常見的原因：

- 串流已在 SQL 程式碼中重新定義：建立串流與相關的幫浦後，您無法在程式碼中重新定義相同的串流。如需如何建立串流的資訊，請參閱《Amazon Kinesis Data Analytics SQL 參考》中的建立串流。如需建立幫浦的詳細資訊，請參閱建立幫浦。

- GROUP BY 子句使用多個 ROWTIME 資料欄：您只能在 GROUP BY 子句中指定一個 ROWTIME 資料行。如需詳細資訊，請參閱《Amazon Kinesis Data Analytics SQL 參考》中的 GROUP BY 和 ROWTIME。
- 一個或多個數據類型具有無效的轉換：在這種情況下，您的程式碼具有無效的隱含轉換。例如，您可能會在程式碼中將 `timestamp` 轉換為 `bigint`。
- 串流與服務保留串流的名稱相同：串流與服務保留串流 `error_stream` 不能有相同的名稱。

應用程式正在將錯誤寫入錯誤資料流

如果應用程式正在將錯誤寫入應用程式內錯誤串流，您可以使用標準程式庫在 `DATA_ROW` 欄位中解碼該值。關於錯誤串流的詳細資訊，請查看 錯誤處理。

輸送量不足或高 MillisBehindLatest

如果應用程式的 MillisBehindLatest 指標穩定增加或持續超過 1000 (一秒)，可能是下列原因所致：

- 檢查您應用程式的 InputBytes CloudWatch 指標。如果你攝入超過每秒 4 MB, 這可能會導致 `MillisBehindLatest` 增加。若要改善應用程式的輸送量，請增加 `InputParallelism` 參數的值。如需更多詳細資訊，請參閱 平行化輸入串流以提高輸送量。
- 檢查應用程式的輸出交付成功指標，以瞭解交付至目的地時是否失敗。請確認您已正確設定輸出，且輸出串流有足夠的容量。
- 如果您的應用程式使用 AWS Lambda 函數進行預處理或作為輸出，請檢查應用程式的InputProcessing.Duration 或 LambdaDelivery.Duration CloudWatch 指標。如果 Lambda 函數調用持續時間超過 5 秒，請考慮執行下列動作：
  - 增加 Lambda 函數的記憶體配置。您可以在 AWS Lambda 主控台組態頁面的基本設定中執行此動作。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的設定 Lambda 函數。
  - 增加應用程式輸入串流中的碎片數。此舉會增加應用程式將調用的平行函數數目，也可能會增加輸送量。
  - 確認函數沒有進行會影響效能的封鎖呼叫，例如對外部資源的同步請求。
  - 檢查您的 AWS Lambda 函數，看看是否有其他可以提高性能的地方。檢查應用程式 Lambda 函數的 CloudWatch Logs。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的存取 Amazon CloudWatch 指標。
- 確認您的應用程式未達到 Kinesis 處理單元 (KPU) 的預設限制。如果應用程式達到此限制，您可以要求提高限制。如需更多詳細資訊，請參閱 自動擴展應用程式以增加輸送量。

- 如果您的應用程式在 KPU 限制增加後仍有問題，請檢查應用程式的輸入輸送量是否不超過每秒 100MB。如果超過每秒 100 MB，我們建議您實施變更以減少整體輸送量來穩定應用程式，例如減少傳送至 Kinesis Data Analytics SQL 應用程式讀取資料來源的資料量。我們也建議使用其他方法，包括增加應用程式的平行處理能力、縮短運算的時間週期、將單欄資料類型從 VARCHAR 變更為較小的資料類型 (例如 INTEGER、LONG 等)、以及減少透過取樣或篩選處理的資料。

> **Note**
>
> 我們建議您定期檢閱應用程式的 `InputProcessing.OkBytes` 指標，以便在應用程式的預計輸入輸送量超過每秒 100 MB 時，事先規劃使用多個 SQL 應用程式，或移轉至 managed-flink/latest/java/。

Kinesis Data Analytics SQL 參考資料

如需 Kinesis Data Analytics 支援之 SQL 語言元素的相關資訊，請參閱 Kinesis Data Analytics SQL 參考資料。

API 參考

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需有關第 2 版的詳細資訊，請參閱 Amazon Managed Service for Apache Flink API V2 文件。

您可以使 AWS CLI 用探索 Amazon Kinesis Data Analytics API。本指南提供使用 AWS CLI的 Amazon Kinesis Data Analytics for SQL 應用程式入門 練習。

主題

動作

支援以下動作：

- ListApplications
- ListTagsForResource
- StartApplication
- StopApplication
- TagResource
- UntagResource
- UpdateApplication

AddApplicationCloudWatchLoggingOption

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

新增 CloudWatch 日誌串流來監控應用程式組態錯誤。如需將 CloudWatch 日誌串流與 Amazon Kinesis Analytics 應用程式搭配使用的詳細資訊，請參閱使用 Amazon CloudWatch Logs。

請求語法

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "string",
      "RoleARN": "string"
   },
   "CurrentApplicationVersionId": number
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

Kinesis Analytics 應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CloudWatchLoggingOption

提供 CloudWatch 日誌串流 Amazon Resource Name (ARN) 和 IAM 角色 ARN。備註：若要將應用程式訊息寫入 CloudWatch，所使用的 IAM 角色必須啟用 `PutLogEvents` 政策動作。

類型：CloudWatchLoggingOption 物件

必要：是

CurrentApplicationVersionId

Kinesis Analytic 應用程式的版本 ID。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

AddApplicationInput

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

將串流來源新增至您的 Amazon Kinesis 應用程式。如需概念資訊，請參閱設定應用程式輸入。

您可以在建立應用程式時新增串流來源，也可以在建立應用程式後使用此作業來新增串流來源。如需詳細資訊，請參閱 CreateApplication。

任何組態更新，包括使用此操作新增串流資源，都會產生應用程式的新版本。您可以使用 DescribeApplication 操作來尋找目前的應用程式版本。

這項操作需要許可來執行 `kinesisanalytics:AddApplicationInput` 動作。

請求語法

```
{
   "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"
      }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

要新增串流來源的現有 Amazon Kinesis Analytics 應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

您的 Amazon Kinesis Analytics 應用程式的最新版本。您可以使用 DescribeApplication 操作來尋找目前的應用程式版本。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

Input

要添加的輸入。

類型：Input 物件

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

CodeValidationException

使用者提供的應用程式碼 (查詢) 無效。這可能是一個簡單的語法錯誤。

HTTP 狀態碼：400

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

AddApplicationInputProcessingConfiguration

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

添加一個 InputProcessingConfiguration 到應用程式。輸入處理器會在輸入資料流上預先處理記錄，然後由應用程式的 SQL 程式碼執行。目前唯一可用的輸入處理器是 AWS Lambda。

請求語法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

您要將輸入處理組態新增到其中的應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

您要將輸入處理組態新增到其中的應用程式版本。您可以使用 DescribeApplication 操作來取得目前的應用程式版本。如果指定的版本不是目前版本，則會傳回 ConcurrentModificationException。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

InputId

要新增輸入處理組態的輸入組態 ID。您可以使用 DescribeApplication 作業取得應用程式的輸入 ID 清單。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：[a-zA-Z0-9_.-]+

必要：是

InputProcessingConfiguration

要新增到應用程式的 InputProcessingConfiguration。

類型：InputProcessingConfiguration 物件

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

AddApplicationOutput

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

將外部目標新增到您的 Amazon Kinesis Analytics 應用程式。

若您希望 Amazon Kinesis Analytics 將您應用程式內串流中的資料交付到外部目標 (例如 Amazon Kinesis 串流、Amazon Kinesis Firehose 交付串流，或是 AWS Lambda 函數)，您可以使用此操作將相關組態新增到您的應用程式。您可以為您的應用程式設定一或多個輸出。每個輸出組態都會映射一個應用程式內串流和外部目標。

您可以使用其中一個輸出組態，將資料從您的應用程式內錯誤串流交付到外部目標，讓您可以分析錯誤。如需詳細資訊，請參閱了解應用程式輸出 (目的地)。

任何組態更新，包括使用此操作新增串流資源，都會產生應用程式的新版本。您可以使用 DescribeApplication 操作來尋找目前的應用程式版本。

如需了解您可以設定的應用程式輸入數及輸出數限制，請參閱限制。

這項操作需要許可來執行 `kinesisanalytics:AddApplicationOutput` 動作。

請求語法

```
{
   "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"
    }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

您要將輸出組態新增到其中的應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

您要將輸出組態新增到其中的應用程式版本。您可以使用 DescribeApplication 操作來取得目前的應用程式版本。如果指定的版本不是目前版本，則會傳回 `ConcurrentModificationException`。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

Output

物件陣列，每個都會描述一個輸出組態。在輸出組態中，您可以指定應用程式內串流的名稱、目標 (Amazon Kinesis 串流、Amazon Kinesis Firehose 交付串流，或是 AWS Lambda 函數)，並記錄在寫入目標時要使用的格式。

類型：Output 物件

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面

- 適用於 .NET 的 AWS 軟體開發套件
- 適用於 C++ 的 AWS 軟體開發套件
- 適用於 Go 的 AWS 軟體開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 軟體開發套件第 3 版
- 適用於 Python 的 AWS 軟體開發套件
- 適用於 Ruby 的 AWS 軟體開發套件第 3 版

AddApplicationReferenceDataSource

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

將參考資料來源新增到現有應用程式。

Amazon Kinesis Analytics 會讀取參考資料 (即 Amazon S3 物件)，並在您的應用程式內建立應用程式內資料表。在請求中，您可以提供來源 (S3 儲存貯體名稱和物件鍵名稱)、要建立的應用程式內資料表名稱，以及描述 Amazon S3 物件中的資料如何映射到結果應用程式內資料表中資料行的必要映射資訊。

如需概念資訊，請參閱設定應用程式輸入。如需了解您可以新增到您應用程式的資料來源限制，請參閱限制。

這項操作需要許可來執行 `kinesisanalytics:AddApplicationOutput` 動作。

請求語法

```
{
   "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"
    }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

現有應用程式的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

您要為其新增參考資料來源的應用程式版本。您可以使用 DescribeApplication 操作來取得目前的應用程式版本。如果指定的版本不是目前版本，則會傳回 `ConcurrentModificationException`。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

ReferenceDataSource

參考資料來源可以是您 Amazon S3 儲存貯體中的物件。Amazon Kinesis Analytics 會讀取物件，並將資料複製到所建立的應用程式內資料表。您可以提供 S3 儲存貯體、物件鍵名稱，以及所建立的結果應用程式內資料表。您也必須提供具備必要許可的 IAM 角色，讓 Amazon Kinesis Analytics 能取得該角色來代您從 S3 儲存貯體讀取物件。

類型：ReferenceDataSource 物件

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

CreateApplication

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

建立 Amazon Kinesis Analytics 應用程式。您可以設定每個應用程式將一個串流來源作為輸入、處理輸入的應用程式碼，以及最多三個您希望 Amazon Kinesis Analytics 從應用程式寫入輸出資料的目的地。如需概觀，請參閱運作方式。

在輸入設定中，將串流來源映射至應用程式內串流，您可以將其視為不斷更新的資料表。在對映中，您必須提供應用程式內串流的結構描述，並將應用程式內串流的每個資料欄映射到串流來源的資料元素。

您的應用程式碼為一個或多個 SQL 陳述式，這些陳述式會讀取輸入資料、進行轉換，並產生輸出。您的應用程式碼可以建立一或多個 SQL 成品，例如 SQL 串流或幫浦。

在輸出組態中，您可以設定應用程式，將資料從應用程式中建立的應用程式內串流寫入最多三個目的地。

若要從來源串流讀取資料，或將資料寫入目的地串流，Amazon Kinesis Analytics 需要您的許可。您可以透過 IAM 角色來授予這些許可。這項操作需要許可來執行 `kinesisanalytics:CreateApplication` 動作。

如需建立 Amazon Kinesis Analytics 應用程式的入門練習，請參閱入門。

請求語法

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

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationCode

一個或多個 SQL 陳述式，該陳述式會讀取輸入資料、進行轉換，並產生輸出。例如，您可以撰寫 SQL 陳述式，從應用程式內串流讀取資料，產生執行中各廠商廣告的平均點擊數，然後使用幫浦將結果資料列插入另一個應用程式中串流。如需典型模式的詳細資訊，請參閱應用程式程式碼。

您可以提供這類一系列的 SQL 陳述式，在其中將一個陳述式的輸出用來做為下一個陳述式的輸入。您可以透過建立應用程式中串流和幫浦，來存放中繼結果。

啟注意，應用程式程式碼必須使用 `Outputs` 中指定的名稱建立串流。例如，若您的 `Outputs` 定義了名為 `ExampleOutputStream1` 及 `ExampleOutputStream2` 的輸出串流，則您的應用程式程式碼必須建立這些串流。

類型：字串

長度限制：長度下限為 0。長度上限為 102400。

必要：否

ApplicationDescription

應用程式的摘要描述。

類型：字串

長度限制：長度下限為 0。長度上限為 1024。

必要：否

ApplicationName

您 Amazon Kinesis Analytics 應用程式的名稱 (例如，`sample-app`)。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CloudWatchLoggingOptions

使用此參數可設定 CloudWatch 日誌串流來監控應用程式組態錯誤。如需詳細資訊，請參閱使用 Amazon CloudWatch Logs。

類型：CloudWatchLoggingOption 物件陣列

必要：否

Inputs

可藉由此參數來設定應用程式輸入。

您可以設定應用程式從單一串流來源接收輸入。在此組態中，您會將此串流來源映射到建立的應用程式內串流。您的應用程式程式碼接著會如同資料表般，查詢應用程式內串流 (您可以將它想成是持續更新的資料表)。

針對串流來源，您可以提供其 Amazon Resource Name (ARN) 及串流上的資料格式 (例如，JSON、CSV 等)。您也必須提供 Amazon Kinesis Analytics 可取得的 IAM 角色，代您讀取此串流。

要建立應用程式內部串流，您必須指定一個架構描述，以將您的資料轉換為 SQL 中使用的架構化版本。在結構描述中，您可以提供串流來源中資料元素的必要映射，來記錄應用程式內串流中的資料行。

類型：Input 物件陣列

必要：否

Outputs

您可以設定應用程式輸出，將資料從任何應用程式內串流寫入最多三個目的地。

這些目的地可以是 Amazon Kinesis 串流、Amazon Kinesis Firehose 交付串流、AWS Lambda 目的地或三者中的任意組合。

在組態中，指定應用程式內串流名稱、目標串流或 Lambda 函數 Amazon Resource Name (ARN)，以及寫入資料時使用的格式。您也必須提供 Amazon Kinesis Analytics 可擔任的 IAM 角色，以代您讀取目的地串流或 Lambda 函數。

在輸出配置中，您還可以提供輸出串流或 Lambda 函數 ARN。針對串流目的地，提供串流中的資料格式 (例如 JSON、CSV)。您也必須提供 Amazon Kinesis Analytics 可擔任的 IAM 角色，以代您讀取串流或 Lambda 函數。

類型：Output 物件陣列

必要：否

Tags

要指派給應用程式的一或多個標籤的清單。標籤是識別應用程式的鍵/值對。請注意，應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。如需詳細資訊，請參閱使用標記。

類型：Tag 物件陣列

陣列成員：項目數下限為 1。項目數上限為 200。

必要：否

回應語法

```
{
   "ApplicationSummary": {
      "ApplicationARN": "string",
      "ApplicationName": "string",
      "ApplicationStatus": "string"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

ApplicationSummary

Amazon Kinesis Analytics 根據您的 `CreateApplication` 要求傳回回應，其中包含建立的應用程式摘要，內含應用程式 Amazon Resource Name (ARN)、名稱和狀態。

類型：ApplicationSummary 物件

錯誤

CodeValidationException

使用者提供的應用程式碼 (查詢) 無效。這可能是一個簡單的語法錯誤。

HTTP 狀態碼：400

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

LimitExceededException

超過允許的應用程式數量。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

TooManyTagsException

使用太多標籤建立的應用程式，或在應用程式中加入太多標籤。請注意，應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

DeleteApplication

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

刪除指定的應用程式。Amazon Kinesis Analytics 會停止執行應用程式並刪除應用程式，包括任何應用程式成品 (例如應用程式內串流、參考資料表和應用程式碼)。

這項操作需要許可來執行 `kinesisanalytics:DeleteApplication` 動作。

請求語法

```
{
   "ApplicationName": "string",
   "CreateTimestamp": number
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

要刪除之 Amazon Kinesis Analytics 應用程式的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CreateTimestamp

若要取得此值，可以使用 `DescribeApplication` 操作。

類型：Timestamp

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版

- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

DeleteApplicationCloudWatchLoggingOption

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

從應用程式中刪除 CloudWatch 日誌串流。如需將 CloudWatch 日誌串流與 Amazon Kinesis Analytics 應用程式搭配使用的詳細資訊，請參閱使用 Amazon CloudWatch Logs。

請求語法

```
{
   "ApplicationName": "string",
   "CloudWatchLoggingOptionId": "string",
   "CurrentApplicationVersionId": number
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

Kinesis Analytics 應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CloudWatchLoggingOptionId

要刪除的 CloudWatch 日誌選項 `CloudWatchLoggingOptionId`。您可以使用 DescribeApplication作業取得 `CloudWatchLoggingOptionId`。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：[a-zA-Z0-9_.-]+

必要：是

CurrentApplicationVersionId

Kinesis Analytic 應用程式的版本 ID。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

DeleteApplicationInputProcessingConfiguration

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

從輸入中刪除 InputProcessingConfiguration。

請求語法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "InputId": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

Kinesis Analytics 應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

Kinesis Analytic 應用程式的版本 ID。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

InputId

要刪除輸入處理組態的輸入組態 ID。您可以使用 DescribeApplication 作業取得應用程式的輸入 ID 清單。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：[a-zA-Z0-9_.-]+

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

DeleteApplicationOutput

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

從應用程式組態中刪除輸出目的地組態。Amazon Kinesis Analytics 不會再將資料從對應的應用程式內串流寫入外部輸出目的地。

這項操作需要許可來執行 `kinesisanalytics:DeleteApplicationOutput` 動作。

請求語法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "OutputId": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

Amazon Kinesis Analytics 應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

Amazon Kinesis Analytics 應用程式版本。您可以使用 DescribeApplication 操作來取得目前的應用程式版本。如果指定的版本不是目前版本，則會傳回 `ConcurrentModificationException`。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

OutputId

要刪除組態的 ID。每個新增至應用程式的輸出組態都有唯一的 ID，無論是在建立應用程式時新增，或稍後使用 AddApplicationOutput 作業加入。您需要提供 ID，以唯一識別要從應用程式組態中刪除的輸出組態。您可以使用 DescribeApplication 作業取得特定的 `OutputId`。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

DeleteApplicationReferenceDataSource

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

從指定的應用程式組態中刪除參考資料來源組態。

如果應用程式正在執行，Amazon Kinesis Analytics 會立即移除您使用 AddApplicationReferenceDataSource 作業建立之應用程式內表格。

這項操作需要許可來執行 `kinesisanalytics.DeleteApplicationReferenceDataSource` 動作。

請求語法

```
{
   "ApplicationName": "string",
   "CurrentApplicationVersionId": number,
   "ReferenceId": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

現有應用程式的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

CurrentApplicationVersionId

應用程式的版本。您可以使用 DescribeApplication 操作來取得目前的應用程式版本。如果指定的版本不是目前版本，則會傳回 ConcurrentModificationException。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

ReferenceId

參考資料來源的 ID。當您使用 AddApplicationReferenceDataSource 將參考資料來源新增至應用程式時，Amazon Kinesis Analytics 會指派一個 ID。您可以使用 DescribeApplication 作業取得參考資料 ID。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：[a-zA-Z0-9_.-]+

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

DescribeApplication

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

傳回特定的 Amazon Kinesis Analytics 應用程式之相關資訊。

如果您想要擷取帳戶中所有應用程式的清單，請使用 ListApplications 作業。

這項操作需要許可來執行 `kinesisanalytics:DescribeApplication` 動作。您可以使用 `DescribeApplication` 來獲取當前應用程式 versionId，此舉需要呼叫其他操作，例如 `Update`。

請求語法

```
{
   "ApplicationName": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

回應語法

```
{
```

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

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

ApplicationDetail

提供應用程式的說明，例如應用程式 Amazon Resource Name (ARN)、狀態、最新版本，以及輸入和輸出組態詳細資訊。

類型：ApplicationDetail 物件

錯誤

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

DiscoverInputSchema

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

透過評估範例記錄來推斷結構描述，這些範例紀錄來自特定的串流來源 (Amazon Kinesis 串流或 Amazon Kinesis Firehose 交付串流) 或 S3 物件。在回應中，作業會傳回推斷的結構描述，以及作業用來推斷結構描述的範例記錄。

在為應用程式設定串流來源時，您可以使用推斷的結構描述。如需概念資訊，請參閱設定應用程式輸入。請注意，當您使用 Amazon Kinesis Analytics 主控台建立應用程式時，主控台會使用此操作推斷結構描述，並在主控台使用者介面中顯示結構描述。

這項操作需要許可來執行 `kinesisanalytics:DiscoverInputSchema` 動作。

請求語法

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "string",
         "RoleARN": "string"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "string"
   },
   "ResourceARN": "string",
   "RoleARN": "string",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string",
      "RoleARN": "string"
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

InputProcessingConfiguration

InputProcessingConfiguration 用於在探索記錄結構描述前預處理記錄。

類型：InputProcessingConfiguration 物件

必要：否

InputStartingPositionConfiguration

您希望 Amazon Kinesis Analytics 從特定串流來源探索目的開始讀取記錄的點。

類型：InputStartingPositionConfiguration 物件

必要：否

ResourceARN

串流來源的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

S3Configuration

指定此參數來探索 Amazon S3 物件中的資料結構描述。

類型：S3Configuration 物件

必要：否

回應語法

```
{
   "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" ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

InputSchema

從串流來源推斷出來的結構描述。辨別串流來源中的資料格式，以及每個資料元素如何映射至相應欄位，這些欄位位於建立的應用程式內串流。

類型：SourceSchema 物件

ParsedInputRecords

元素陣列，其中每個元素對應到串流記錄中的一列 (串流記錄可以有多個資料列)。

類型：字串陣列的陣列。

ProcessedInputRecords

在 `InputProcessingConfiguration` 參數中指定的處理器修改之串流資料。

類型：字串陣列

RawInputRecords

已取樣來推斷結構描述的原始串流資料。

類型：字串陣列

錯誤

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceProvisionedThroughputExceededException

探索無法從串流來源取得記錄，因為 Amazon Kinesis Streams ProvisionedThroughputExceededException。如需詳細資訊，請參閱 Amazon Kinesis Video 串流 API 參考資料中的 GetRecords。

HTTP 狀態碼：400

ServiceUnavailableException

服務無法使用。退回並重試操作。

HTTP 狀態碼：500

UnableToDetectSchemaException

資料格式無效。Amazon Kinesis Analytics 無法偵測指定串流來源的結構描述。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

ListApplications

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

傳回您帳戶中的 Amazon Kinesis Analytics 應用程式清單。在每個應用程式，回應包括應用程式名稱、Amazon Resource Name (ARN)，以及狀態。如果回應傳回 `HasMoreApplications` 值為 true，您可以傳送另一個要求，方法為在請求內文中加入 `ExclusiveStartApplicationName`，並將此值設定為先前回應的最後一個應用程式名稱。

如果您想要特定應用程式的詳細資訊，請使用 DescribeApplication。

這項操作需要許可來執行 `kinesisanalytics:ListApplications` 動作。

請求語法

```
{
   "ExclusiveStartApplicationName": "string",
   "Limit": number
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ExclusiveStartApplicationName

清單開頭的應用程式名稱。使用分頁來擷取列表時，您不必在第一個請求中指定此參數。不過，在後續請求中，您可以新增先前回應中的最後一個應用程式名稱，以取得應用程式的下一頁。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：否

Limit

要列出的應用程式數目上限。

類型：整數

有效範圍：最小值為 1。最大值為 50。

必要：否

回應語法

```
{
   "ApplicationSummaries": [
      {
         "ApplicationARN": "string",
         "ApplicationName": "string",
         "ApplicationStatus": "string"
      }
   ],
   "HasMoreApplications": boolean
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

ApplicationSummaries

ApplicationSummary 物件的清單。

類型：ApplicationSummary 物件陣列

HasMoreApplications

如有更多應用程式可擷取，則傳回 true。

類型：布林值

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

ListTagsForResource

擷取指派給應用程式的索引鍵值標籤清單。如需詳細資訊，請參閱使用標籤。

請求語法

```
{
   "ResourceARN": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ResourceARN

要從中擷取標籤的應用程式 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：arn:.*

必要：是

回應語法

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Tags

指派給應用程式的索引鍵值標籤。

類型：Tag 物件陣列

陣列成員：項目數下限為 1。項目數上限為 200。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版

- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

StartApplication

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

啟動指定的 Amazon Kinesis Analytics 應用程式。建立應用程式後，您必須專門呼叫此操作來啟動應用程序。

應用程式啟動後，即會開始使用輸入資料、處理輸入資料，然後將輸出寫入設定的目的地。

應用程式狀態必須是 READY 才能啟動應用程序。您可以使用 DescribeApplication 作業在主控台取得應用程式的狀態。

啟動應用程式之後，您可以呼叫 StopApplication 作業，來讓應用程式停止處理輸入。

這項操作需要許可來執行 `kinesisanalytics:StartApplication` 動作。

請求語法

```
{
   "ApplicationName": "string",
   "InputConfigurations": [
      {
         "Id": "string",
         "InputStartingPositionConfiguration": {
            "InputStartingPosition": "string"
         }
      }
   ]
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

InputConfigurations

依 ID 識別應用程式開始使用的特定輸入。Amazon Kinesis Analytics 會開始讀取與輸入相關聯的串流來源。您也可以指定 Amazon Kinesis Analytics 要在串流來源中的哪個位置開始讀取。

類型：InputConfiguration 物件陣列

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

InvalidApplicationConfigurationException

使用者提供的應用組態無效。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

StopApplication

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

讓應用程式停止處理輸入資料。只有在應用程式處於執行中狀態時，才能停止該應用程式。您可以使用 DescribeApplication 操作來查看應用程式狀態。應用程式停止後，Amazon Kinesis Analytics 會停止從輸入讀取資料、應用程式停止處理資料，且不會將輸出寫入到目的地。

這項操作需要許可來執行 `kinesisanalytics:StopApplication` 動作。

請求語法

```
{
   "ApplicationName": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

要停止的執行中應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

TagResource

將一或多個索引鍵值標籤新增至 Kinesis Analytics 應用程式。請注意，應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。如需詳細資訊，請參閱使用標籤。

請求語法

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

請求參數

請求接受採用 JSON 格式的下列資料。

ResourceARN

要指派標籤的應用程式 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

Tags

指派給應用程式的索引鍵值標籤。

類型：Tag 物件陣列

陣列成員：項目數下限為 1。項目數上限為 200。

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

TooManyTagsException

使用太多標籤建立的應用程式，或在應用程式中加入太多標籤。請注意，應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件

- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

UntagResource

從 Kinesis Analytics 應用程式移除一或多個標籤。如需詳細資訊，請參閱使用標籤。

請求語法

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ResourceARN

要從中移除標籤之 Kinesis Analytics 應用程式的 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

TagKeys

要從指定應用程式移除的標籤金鑰清單。

類型：字串陣列

陣列成員：項目數下限為 1。項目數上限為 200。

長度限制：長度下限為 1。長度上限為 128。

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

TooManyTagsException

使用太多標籤建立的應用程式，或在應用程式中加入太多標籤。請注意，應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版

- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

UpdateApplication

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

更新現有的 Amazon Kinesis Analytics 應用程式。使用此 API，您可以更新應用程式碼、輸入組態和輸出設定。

請注意，Amazon Kinesis Analytics 會在您每次更新應用程式時更新 `CurrentApplicationVersionId`。

這項操作需要 `kinesisanalytics:UpdateApplication` 動作的許可。

請求語法

```
{
   "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
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ApplicationName

要更新之 Amazon Kinesis Analytics 應用程式的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

ApplicationUpdate

說明應用程式更新。

類型：ApplicationUpdate 物件

必要：是

CurrentApplicationVersionId

目前的應用程式版本 ID。您可以使用 DescribeApplication 作業取得此值 。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

CodeValidationException

使用者提供的應用程式碼 (查詢) 無效。這可能是一個簡單的語法錯誤。

HTTP 狀態碼：400

ConcurrentModificationException

並行修改應用程式的結果拋出例外。例如，有兩個人嘗試同時編輯相同的應用程式。

HTTP 狀態碼：400

InvalidArgumentException

指定的輸入參數值無效。

HTTP 狀態碼：400

ResourceInUseException

應用程式不適用於此作業。

HTTP 狀態碼：400

ResourceNotFoundException

找不到指定的應用程式。

HTTP 狀態碼：400

UnsupportedOperationException

請求被拒絕，因為指定的參數不受支持，或指定的資源對此操作無效。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS 命令列介面
- 適用於 .NET 的 AWS 開發套件
- AWS SDK for C++
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 軟體開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件第 3 版
- 適用於 PHP 的 AWS 開發套件第 3 版

- AWS SDK for Python
- 適用於 Ruby V3 的 AWS 開發套件

資料類型

目前支援下列資料類型：

- ApplicationDetail
- ApplicationSummary
- ApplicationUpdate
- CloudWatchLoggingOption
- CloudWatchLoggingOptionDescription
- CloudWatchLoggingOptionUpdate
- CSVMappingParameters
- DestinationSchema
- Input
- InputConfiguration
- InputDescription
- InputLambdaProcessor
- InputLambdaProcessorDescription
- InputLambdaProcessorUpdate
- InputParallelism
- InputParallelismUpdate
- InputProcessingConfiguration
- InputProcessingConfigurationDescription
- InputProcessingConfigurationUpdate
- InputSchemaUpdate
- InputStartingPositionConfiguration
- InputUpdate
- JSONMappingParameters
- KinesisFirehoseInput

- KinesisFirehoseInputDescription
- KinesisFirehoseInputUpdate
- KinesisFirehoseOutput
- KinesisFirehoseOutputDescription
- KinesisFirehoseOutputUpdate
- KinesisStreamsInput
- KinesisStreamsInputDescription
- KinesisStreamsInputUpdate
- KinesisStreamsOutput
- KinesisStreamsOutputDescription
- KinesisStreamsOutputUpdate
- LambdaOutput
- LambdaOutputDescription
- LambdaOutputUpdate
- MappingParameters
- Output
- OutputDescription
- OutputUpdate
- RecordColumn
- RecordFormat
- ReferenceDataSource
- ReferenceDataSourceDescription
- ReferenceDataSourceUpdate
- S3Configuration
- S3ReferenceDataSource
- S3ReferenceDataSourceDescription
- S3ReferenceDataSourceUpdate
- SourceSchema
- Tag

ApplicationDetail

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

提供應用程式的說明，包含應用程式 Amazon Resource Name (ARN)、狀態、最新版本，以及輸入和輸出組態。

目錄

ApplicationARN

應用程式的 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

ApplicationName

應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

ApplicationStatus

應用程式的狀態。

類型：字串

有效值: DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING

必要：是

ApplicationVersionId

提供目前的應用程式版本。

類型：Long

有效範圍：最小值為 1。最大值為 999999999。

必要：是

ApplicationCode

傳回您提供的應用程式碼，以便對應用程式中的任何應用程式內串流執行資料分析。

類型：字串

長度限制：長度下限為 0。長度上限為 102400。

必要：否

ApplicationDescription

應用程式的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 1024。

必要：否

CloudWatchLoggingOptionDescriptions

描述設定為接收應用程式訊息的 CloudWatch 日誌串流。如需將 CloudWatch 日誌串流與 Amazon Kinesis Analytics 應用程式搭配使用的詳細資訊，請參閱使用 Amazon CloudWatch Logs。

類型：CloudWatchLoggingOptionDescription 物件陣列

必要：否

CreateTimestamp

建立應用程式的時間戳記。

類型：Timestamp

必要：否

InputDescriptions

描述應用程式輸入組態。如需詳細資訊，請參閱設定應用程式輸入。

類型：InputDescription 物件陣列

必要：否

LastUpdateTimestamp

上次更新應用程式時的時間戳記。

類型：Timestamp

必要：否

OutputDescriptions

描述應用程式輸出組態。如需詳細資訊，請參閱設定應用程式輸出。

類型：OutputDescription 物件陣列

必要：否

ReferenceDataSourceDescriptions

描述為應用程式設定的參考資料來源。如需詳細資訊，請參閱設定應用程式輸入。

類型：ReferenceDataSourceDescription 物件陣列

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

ApplicationSummary

> **Note**
>
> 此文件適用於 Amazon Kinesis Data Analytics API 第 1 版，僅支援 SQL 應用程式。第 2 版的 API 則支援 SQL 和 Java 應用程式。如需第 2 版的詳細資訊，請參閱 Amazon Kinesis Data Analytics API V2 文件。

提供應用程式摘要資訊，包括應用程式 Amazon Resource Name (ARN)、名稱和狀態。

目錄

ApplicationARN

應用程式的 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

ApplicationName

應用程式名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

模式：`[a-zA-Z0-9_.-]+`

必要：是

ApplicationStatus

應用程式的狀態。

類型：字串

有效值: `DELETING | STARTING | STOPPING | READY | RUNNING | UPDATING | AUTOSCALING`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

ApplicationUpdate

說明現有的 Amazon Kinesis Analytics 應用程式要套用之更新。

目錄

ApplicationCodeUpdate

說明應用程式碼更新。

類型：字串

長度限制：長度下限為 0。長度上限為 102400。

必要：否

CloudWatchLoggingOptionUpdates

說明應用程式 CloudWatch 記錄選項更新。

類型：CloudWatchLoggingOptionUpdate 物件陣列

必要：否

InputUpdates

描述應用程式輸入組態更新。

類型：InputUpdate 物件陣列

必要：否

OutputUpdates

描述應用程式輸出組態更新。

類型：OutputUpdate 物件陣列

必要：否

ReferenceDataSourceUpdates

描述應用程式參考資料來源更新。

類型：ReferenceDataSourceUpdate 物件陣列

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

CloudWatchLoggingOption

提供 CloudWatch 記錄選項的描述，包括日誌串流 Amazon Resource Name (ARN) 與角色 ARN。

目錄

LogStreamARN

接收應用程式訊息的 CloudWatch 日誌 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

用來傳送應用程式訊息之角色的 IAM ARN。備註：若要將應用程式訊息寫入 CloudWatch，所使用的 IAM 角色必須啟用 `PutLogEvents` 政策動作。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

CloudWatchLoggingOptionDescription

CloudWatch 記錄選項的說明。

目錄

LogStreamARN

接收應用程式訊息的 CloudWatch 日誌 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

用來傳送應用程式訊息之角色的 IAM ARN。備註：若要將應用程式訊息寫入 CloudWatch，所使用的 IAM 角色必須啟用 `PutLogEvents` 政策動作。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

CloudWatchLoggingOptionId

CloudWatch 記錄選項說明的 ID。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

CloudWatchLoggingOptionUpdate

說明 CloudWatch 記錄選項更新。

目錄

CloudWatchLoggingOptionId

要更新的 CloudWatch 記錄選項的 ID

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

LogStreamARNUpdate

接收應用程式訊息的 CloudWatch 日誌 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

用來傳送應用程式訊息之角色的 IAM ARN。備註：若要將應用程式訊息寫入 CloudWatch，所使用的 IAM 角色必須啟用 `PutLogEvents` 政策動作。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

CSVMappingParameters

當紀錄格式使用 CSV 等分隔符號時，提供其他的映射資訊。例如，以下的範例紀錄使用 CSV 格式，其紀錄使用 '\n' 做為資料列分隔符號，使用逗號 (",") 做為資料行分隔符號：

```
"name1", "address1"

"name2", "address2"
```

目錄

RecordColumnDelimiter

資料行分隔符號。例如，在 CSV 格式中，逗號 (",") 是典型的欄位分隔符號。

類型：字串

長度限制：長度下限為 1。

必要：是

RecordRowDelimiter

資料列分隔符號。例如，在 CSV 格式中，'\n' 是典型的資料列分隔符號。

類型：字串

長度限制：長度下限為 1。

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

DestinationSchema

描述紀錄寫入目標時所採用的資料格式。如需詳細資訊，請參閱設定應用程式輸出。

目錄

RecordFormatType

指定輸出串流中的記錄格式。

類型：字串

有效值: `JSON | CSV`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

Input

當您設定應用程式輸入時，您指定串流來源、建立的應用程式內串流名稱，以及兩者之間的映射。如需詳細資訊，請參閱設定應用程式輸入。

目錄

InputSchema

描述串流來源中的資料格式，以及每個資料元素如何映射至所建立應用程式內串流的對應欄位。

也用於描述參考資料來源的格式。

類型：SourceSchema 物件

必要：是

NamePrefix

建立應用程式內串流時要使用的名稱前綴。假設您指定前綴 "MyInApplicationStream"。則 Amazon Kinesis Analytics 會 (根據您指定的 `InputParallelism` 計數) 建立一或多個應用程式內串流，名為 "MyInApplicationStream_001"、"MyInApplicationStream_002" 等等。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：是

InputParallelism

描述要建立的應用程式內串流數量。

您來源的資料會路由到這些應用程式內輸入串流。

(請參閱設定應用程式輸入。

類型：InputParallelism 物件

必要：否

InputProcessingConfiguration

輸入的 InputProcessingConfiguration。輸入處理器在收到記錄時會轉換記錄，然後由應用程式的 SQL 程式碼執行。目前唯一可用的輸入處理組態是 InputLambdaProcessor。

類型：InputProcessingConfiguration 物件

必要：否

KinesisFirehoseInput

如果串流來源是 Amazon Kinesis Firehose 交付串流，請識別交付串流的 ARN 和能讓 Amazon Kinesis Analytics 代您存取串流的 IAM 角色。

注意：`KinesisStreamsInput` 或 `KinesisFirehoseInput` 為必要。

類型：KinesisFirehoseInput 物件

必要：否

KinesisStreamsInput

如果串流來源是 Amazon Kinesis 串流，請識別串流的 Amazon Resource Name (ARN) 和能代您存取串流的 IAM 角色。

注意：`KinesisStreamsInput` 或 `KinesisFirehoseInput` 為必要。

類型：KinesisStreamsInput 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputConfiguration

當啟動應用程式時，您要提供此組態以識別輸入來源，以及其中您希望應用程式開始處理記錄的點。

目錄

Id

輸入來源 ID。您可以呼叫 DescribeApplication 作業取得此 ID。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

InputStartingPositionConfiguration

您希望應用程式從串流來源開始處理記錄的點。

類型：InputStartingPositionConfiguration 物件

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputDescription

描述應用程式輸入組態。如需詳細資訊，請參閱設定應用程式輸入。

目錄

InAppStreamNames

傳回映射至串流來源的應用程式內串流名稱。

類型：字串陣列

長度限制：長度下限為 1。長度上限為 32。

必要：否

InputId

與應用程式輸入相關聯的輸入 ID。這是 Amazon Kinesis Analytics 指派給您新增至應用程式的每個輸入組態之 ID。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：否

InputParallelism

描述設定的平行處理 (映射至串流來源的應用程式內串流數量)。

類型：InputParallelism 物件

必要：否

InputProcessingConfigurationDescription

執行應用程式碼前，在此輸入的記錄上執行之預處理器描述。

類型：InputProcessingConfigurationDescription 物件

必要：否

InputSchema

描述串流來源中的資料格式，以及每個資料元素如何映射至所建立應用程式內串流的對應欄位。

類型：SourceSchema 物件

必要：否

InputStartingPositionConfiguration

應用程式設定從輸入串流讀取的點。

類型：InputStartingPositionConfiguration 物件

必要：否

KinesisFirehoseInputDescription

如果設定串流來源為 Amazon Kinesis Firehose 交付串流，會提供交付串流的 ARN，和能讓 Amazon Kinesis Analytics 代您存取串流的 IAM 角色。

類型：KinesisFirehoseInputDescription 物件

必要：否

KinesisStreamsInputDescription

如果串流來源是 Amazon Kinesis 串流，會提供 Amazon Kinesis 串流的 Amazon Resource Name (ARN)，和能代您允許 Amazon Kinesis Analytics 存取串流的 IAM 角色。

類型：KinesisStreamsInputDescription 物件

必要：否

NamePrefix

應用程式內名稱字首。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputLambdaProcessor

此為物件，包含用來預先處理串流記錄的 AWS Lambda 函數的 Amazon Resource Name (ARN)，以及用來存取 AWS Lambda 函數的 IAM 角色 ARN。

目錄

ResourceARN

在串流記錄中運作的 AWS Lambda 函數的 ARN。

> **Note**
>
> 若要指定非最新版的舊版 Lambda 函數，請在 Lambda 函數 ARN 中包含 Lambda 函數版本。如需 Lambda ARN 的詳細資訊，請參閱範例 ARN：AWS Lambda

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

用於存取 AWS Lambda 函數的 IAM 角色 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件

- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputLambdaProcessorDescription

此為物件，包含用來預先處理串流記錄的 AWS Lambda 函數的 Amazon Resource Name (ARN)，以及用來存取 AWS Lambda 運算式的 IAM 角色 ARN。

目錄

ResourceARN

在串流記錄中預處理 AWS Lambda 函數的 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

用於存取 AWS Lambda 函數的 IAM 角色 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputLambdaProcessorUpdate

表示InputLambdaProcessor的更新，該更新用於預處理串流中的記錄。

目錄

ResourceARNUpdate

在串流中預處理紀錄的新 AWS Lambda 函數之 Amazon Resource Name (ARN)。

> **Note**
>
> 若要指定非最新版的舊版 Lambda 函數，請在 Lambda 函數 ARN 中包含 Lambda 函數版本。如需 Lambda ARN 的詳細資訊，請參閱範例 ARN：AWS Lambda

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

用於存取 AWS Lambda 函數的新 IAM 角色 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件

- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputParallelism

描述針對指定串流來源所建立的應用程式內串流數目。如需平行處理的資訊，請參閱設定應用程式輸出。

目錄

Count

要建立的應用程式內串流數量。如需詳細資訊，請參閱限制。

類型：整數

有效範圍：最小值為 1。最大值為 64。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputParallelismUpdate

提供平行處理計數的更新。

目錄

CountUpdate

針對指定串流來源所建立的應用程式內串流數目。

類型：整數

有效範圍：最小值為 1。最大值為 64。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputProcessingConfiguration

提供處理器的描述，先用此處理器預先處理串流中的紀錄，再交由您的應用程式碼處理。目前唯一可用的輸入處理器是 AWS Lambda。

目錄

InputLambdaProcessor

InputLambdaProcessor，用來先預先處理串流中的記錄，再由您的應用程式碼處理它們。

類型：InputLambdaProcessor 物件

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputProcessingConfigurationDescription

提供有關輸入處理器的組態資訊。目前唯一可用的輸入處理器是 AWS Lambda。

目錄

InputLambdaProcessorDescription

提供與 InputLambdaProcessorDescription 相關的組態資訊說明。

類型：InputLambdaProcessorDescription 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputProcessingConfigurationUpdate

描述 InputProcessingConfiguration 的更新。

目錄

InputLambdaProcessorUpdate

提供 InputLambdaProcessor 的更新資訊。

類型：InputLambdaProcessorUpdate 物件

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputSchemaUpdate

說明應用程式輸入結構描述的更新。

目錄

RecordColumnUpdates

RecordColumn 物件的清單。每個物件皆描述串流來源元素與應用程式內串流中對應欄位的映射。

類型：RecordColumn 物件陣列

陣列成員：項目數下限為 1。項目數上限為 1000。

必要：否

RecordEncodingUpdate

指定串流來源中的記錄編碼。例如，UTF-8。

類型：字串

模式：UTF-8

必要：否

RecordFormatUpdate

指定串流來源中的記錄格式。

類型：RecordFormat 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputStartingPositionConfiguration

描述應用程式從串流來源讀取的點。

目錄

InputStartingPosition

串流上的開始位置。

- NOW：在串流中的最新記錄之後開始讀取，從客戶發出的請求時間戳記開始。
- TRIM_HORIZON：從串流中最後一個未修剪記錄開始讀取，這是串流中最舊的記錄。Amazon Kinesis Firehose 交付串流不可使用此選項。
- LAST_STOPPED_POINT：繼續閱讀應用程式上次停止讀取的位置。

類型：字串

有效值: NOW | TRIM_HORIZON | LAST_STOPPED_POINT

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

InputUpdate

描述特定輸入組態 (由應用程式的 `InputId` 識別) 之更新。

目錄

InputId

要更新的應用程式輸入之輸入 ID。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

InputParallelismUpdate

描述平行處理的更新 (Amazon Kinesis Analytics 為特定串流來源建立的應用程式內串流數目)。

類型：InputParallelismUpdate 物件

必要：否

InputProcessingConfigurationUpdate

描述輸入處理組態的更新。

類型：InputProcessingConfigurationUpdate 物件

必要：否

InputSchemaUpdate

描述串流來源中的資料格式，以及串流來源的紀錄元素如何映射至建立的應用程式內串流資料欄。

類型：InputSchemaUpdate 物件

必要：否

KinesisFirehoseInputUpdate

如果 Amazon Kinesis Firehose 交付串流是要更新的串流來源，則會提供更新的串流 ARN 和 IAM 角色 ARN。

類型：KinesisFirehoseInputUpdate 物件

必要：否

KinesisStreamsInputUpdate

如果 Amazon Kinesis 串流是要更新的串流來源，則會提供更新的串流 Amazon Resource Name (ARN) 和 IAM 角色 ARN。

類型：KinesisStreamsInputUpdate 物件

必要：否

NamePrefixUpdate

Amazon Kinesis Analytics 為特定串流來源建立的應用程式內串流之名稱前置詞。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

JSONMappingParameters

在 JSON 為串流來源的記錄格式時，提供額外的映射資訊。

目錄

RecordRowPath

包含紀錄的最上層父系路徑。

類型：字串

長度限制：長度下限為 1。

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisFirehoseInput

將 Amazon Kinesis Firehose 交付串流識別為串流來源。您提供的交付串流 Amazon Resource Name (ARN) 和 IAM 角色 ARN，可讓 Amazon Kinesis Analytics 代您存取串流。

目錄

ResourceARN

輸入交付串流的 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。您需要確保此角色具有存取串流的必要許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisFirehoseInputDescription

描述在應用程式輸入組態中設定為串流來源的 Amazon Kinesis Firehose 交付串流。

目錄

ResourceARN

Amazon Kinesis Firehose 交付串流的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

Amazon Kinesis Analytics 擔任的 IAM 角色 ARN，此角色可代您存取串流。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisFirehoseInputUpdate

在更新應用程式輸入組態時，提供 Amazon Kinesis Firehose 交付串流的資訊，以作為串流來源。

目錄

ResourceARNUpdate

要讀取的 Amazon Kinesis Firehose 交付串流的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisFirehoseOutput

在設定應用程式輸出時，將 Amazon Kinesis Firehose 交付串流識別為目標。您提供的串流 Amazon Resource Name (ARN) 和 IAM 角色，可讓 Amazon Kinesis Analytics 代您寫入串流。

目錄

ResourceARN

要寫入的目標 Amazon Kinesis Firehose 交付串流 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，代您寫入目標串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisFirehoseOutputDescription

針對應用程式輸出，描述設定為其目的地的 Amazon Kinesis Firehose 交付串流。

目錄

ResourceARN

Amazon Kinesis Firehose 交付串流的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可存取串流。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisFirehoseOutputUpdate

使用UpdateApplication作業更新輸出組態時，會提供設定為目的地之 Amazon Kinesis Firehose 交付串流的相關資訊。

目錄

ResourceARNUpdate

要寫入的 Amazon Kinesis Firehose 交付串流 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisStreamsInput

將 Amazon Kinesis 串流識別為串流來源。您提供的串流 Amazon Resource Name (ARN) 和 IAM 角色 ARN，可讓 Amazon Kinesis Analytics 代您存取串流。

目錄

ResourceARN

要讀取的輸入 Amazon Kinesis 串流 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisStreamsInputDescription

描述在應用程式輸入組態中設定為串流來源的 Amazon Kinesis 串流。

目錄

ResourceARN

Amazon Kinesis 串流的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可存取串流。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisStreamsInputUpdate

在更新應用程式輸入組態時，提供 Amazon Kinesis 串流的資訊，以作為串流來源。

目錄

ResourceARNUpdate

要讀取的輸入 Amazon Kinesis 串流之 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisStreamsOutput

在設定應用程式輸出時，將 Amazon Kinesis 串流識別為目標。您要提供串流 Amazon Resource Name (ARN) 以及可讓 Amazon Kinesis Analytics 使用的 IAM 角色 ARN，代您寫入串流。

目錄

ResourceARN

要寫入的目標 Amazon Kinesis 串流 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，代您寫入目標串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisStreamsOutputDescription

針對應用程式輸出，描述設定為其目的地的 Amazon Kinesis 串流。

目錄

ResourceARN

Amazon Kinesis 串流的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可存取串流。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

KinesisStreamsOutputUpdate

使用 UpdateApplication 作業更新輸出組態時，會提供設定為目的地之 Amazon Kinesis 串流的相關資訊。

目錄

ResourceARNUpdate

您想要寫入輸出的 Amazon Kinesis 串流之 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色可代您存取串流。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

LambdaOutput

設定應用程式輸出時，將 AWS Lambda 函數識別為目的地。您要提供函數 Amazon Resource Name (ARN) 以及可讓 Amazon Kinesis Analytics 使用的 IAM 角色，代您寫入函數。

目錄

ResourceARN

要寫入之目標 Lambda 函數的 Amazon Resource Name (ARN)。

> **Note**
>
> 若要指定非最新版的舊版 Lambda 函數，請在 Lambda 函數 ARN 中包含 Lambda 函數版本。如需 Lambda ARN 的詳細資訊，請參閱範例 ARN：AWS Lambda

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色代您寫入目標函數。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

LambdaOutputDescription

針對應用程式輸出，描述設定為其目的地的 AWS Lambda 函數。

目錄

ResourceARN

目的地 Lambda 函數的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色會寫入目標函數。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

LambdaOutputUpdate

使用 UpdateApplication 作業更新輸出組態時，會提供設定為目的地之AWS Lambda 函數的相關資訊。

目錄

ResourceARNUpdate

目的地 Lambda 函數的 Amazon Resource Name (ARN)。

> **Note**
>
> 若要指定非最新版的舊版 Lambda 函數，請在 Lambda 函數 ARN 中包含 Lambda 函數版本。如需 Lambda ARN 的詳細資訊，請參閱範例 ARN：AWS Lambda

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

RoleARNUpdate

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色代您寫入目標函數。您需要授予此角色必要的許可。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

MappingParameters

在建立或更新應用程式期間設定應用程式輸入時，請提供對串流來源記錄格式而言特定的額外映射資訊 (例如 JSON、CSV 或由一些分隔符號分隔的記錄欄位)。

目錄

CSVMappingParameters

當記錄格式使用 CSV 等分隔符號時，此屬性會提供其他的映射資訊。

類型：CSVMappingParameters 物件

必要：否

JSONMappingParameters

在 JSON 為串流來源的記錄格式時，提供額外的映射資訊。

類型：JSONMappingParameters 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

Output

描述應用程式輸出組態，您可在此識別應用程式內串流和應用程式內串流資料寫入的目標。目標可以是 Amazon Kinesis 串流或 Amazon Kinesis Firehose 交付串流。

如需應用程式可以寫入的目標數限制和其他限制，請參閱限制。

目錄

DestinationSchema

描述紀錄寫入目標時所採用的資料格式。如需詳細資訊，請參閱設定應用程式輸出。

類型：DestinationSchema 物件

必要：是

Name

應用程式內串流的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：是

KinesisFirehoseOutput

將 Amazon Kinesis Firehose 交付串流識別為目標。

類型：KinesisFirehoseOutput 物件

必要：否

KinesisStreamsOutput

將 Amazon Kinesis 串流識別為目標。

類型：KinesisStreamsOutput 物件

必要：否

LambdaOutput

將 AWS Lambda 函數識別為目的地。

類型：LambdaOutput 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

OutputDescription

描述應用程式輸出組態，其中包括應用程式內串流名稱和串流資料寫入的目的地。目標可以是 Amazon Kinesis 串流或 Amazon Kinesis Firehose 交付串流。

目錄

DestinationSchema

用於將資料寫入目的地的資料格式。

類型：DestinationSchema 物件

必要：否

KinesisFirehoseOutputDescription

描述設定為輸出寫入目的地的 Amazon Kinesis Firehose 交付串流。

類型：KinesisFirehoseOutputDescription 物件

必要：否

KinesisStreamsOutputDescription

描述設定為輸出寫入的目的地的 Amazon Kinesis 串流。

類型：KinesisStreamsOutputDescription 物件

必要：否

LambdaOutputDescription

描述設定為輸出寫入目的地的 AWS Lambda 函數。

類型：LambdaOutputDescription 物件

必要：否

Name

設定為輸出的應用程式內串流名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：否

OutputId

輸出組態的唯一識別符。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

OutputUpdate

描述由 `OutputId` 識別之輸出組態的更新。

目錄

OutputId

識別您想要更新的特定輸出組態。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

DestinationSchemaUpdate

描述紀錄寫入目標時所採用的資料格式。如需詳細資訊，請參閱設定應用程式輸出。

類型：DestinationSchema 物件

必要：否

KinesisFirehoseOutputUpdate

描述設定為輸出目的地的 Amazon Kinesis Firehose 交付串流。

類型：KinesisFirehoseOutputUpdate 物件

必要：否

KinesisStreamsOutputUpdate

將 Amazon Kinesis 串流描述為輸出目的地。

類型：KinesisStreamsOutputUpdate 物件

必要：否

LambdaOutputUpdate

將 AWS Lambda 函數描述為輸出的目的地。

類型：LambdaOutputUpdate 物件

必要：否

NameUpdate

如果您要為此輸出組態指定不同的應用程式內串流，請使用此欄位來指定新的應用程式內串流名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

RecordColumn

描述串流來源中每個資料元素與應用程式內串流中對應欄位的映射。

也用於描述參考資料來源的格式。

目錄

Name

在應用程式內輸入串流或參考表中建立的欄位名稱。

類型：字串

必要：是

SqlType

在應用程式內輸入串流或參考表中建立的欄位類型。

類型：字串

長度限制：長度下限為 1。

必要：是

Mapping

串流輸入資料元素或參考資料來源的參考。如果 RecordFormatType 是 JSON，則此為必要元素。

類型：字串

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

RecordFormat

描述紀錄格式和相關的映射資訊，它們應用來結構化串流內的記錄。

目錄

RecordFormatType

紀錄格式的類型。

類型：字串

有效值: `JSON | CSV`

必要：是

MappingParameters

在建立或更新應用程式期間設定應用程式輸入時，請提供對串流來源記錄格式而言特定的額外映射資訊 (例如 JSON、CSV 或由一些分隔符號分隔的記錄欄位)。

類型：MappingParameters 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

ReferenceDataSource

描述參考資料來源，方法是提供來源資訊 (S3 儲存貯體名稱和物件索引鍵名稱)、已建立的產生應用程式內資料表名稱，以及將 Amazon S3 物件中的資料元素映射至應用程式中資料表的必要結構描述。

目錄

ReferenceSchema

描述串流來源中的資料格式，以及每個資料元素如何映射至應用程式內串流中所建立的對應欄位。

類型：SourceSchema 物件

必要：是

TableName

要建立的應用程式內資料表名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：是

S3ReferenceDataSource

識別包含參考資料的 S3 儲存貯體和物件。也請識別 Amazon Kinesis Analytics 可擔任的 IAM 角色，代您讀取此物件。Amazon Kinesis Analytics 應用程式只會載入一次參考資料。如果資料變更，您可以呼叫 `UpdateApplication` 操作，觸發將資料重新載入到您的應用程式。

類型：S3ReferenceDataSource 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版

- 適用於 Ruby 的 AWS 開發套件第 3 版

ReferenceDataSourceDescription

描述設定給應用程式的參考資料來源。

目錄

ReferenceId

參考資料來源的 ID。當您使用 AddApplicationReferenceDataSource 作業將參考資料來源新增至應用程式時，Amazon Kinesis Analytics 會指派一個 ID。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

S3ReferenceDataSourceDescription

提供 S3 儲存貯體名稱，該物件金鑰包含參考資料。它還提供 Amazon Kinesis Analytics 可擔任的 IAM 角色之 Amazon Resource Name (ARN)，可讓 Amazon Kinesis Analytics 讀取 Amazon S3 物件並填入應用程式內參考資料表。

類型：S3ReferenceDataSourceDescription 物件

必要：是

TableName

由特定參考資料來源組態建立的應用程式內表格名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：是

ReferenceSchema

描述串流來源中的資料格式，以及每個資料元素如何映射至應用程式內串流中所建立的對應欄位。

類型：SourceSchema 物件

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

ReferenceDataSourceUpdate

當您更新應用程式的參考資料來源組態時，此物件會提供所有更新的值 (例如來源儲存貯體名稱和物件金鑰名稱)、建立的應用程式內表格名稱，以及更新的映射資訊，將 Amazon S3 物件中的資料映射至建立的應用程式內參考資料表。

目錄

ReferenceId

更新中參考資料來源的 ID。您可以使用 DescribeApplication 作業取得此值 。

類型：字串

長度限制：長度下限為 1。長度上限為 50。

模式：`[a-zA-Z0-9_.-]+`

必要：是

ReferenceSchemaUpdate

描述串流來源中的資料格式，以及每個資料元素如何映射至應用程式內串流中所建立的對應欄位。

類型：SourceSchema 物件

必要：否

S3ReferenceDataSourceUpdate

描述 Amazon Kinesis 分析可代表您讀取 Amazon S3 物件並填入應用程式內參考資料表的 S3 儲存貯體名稱、物件金鑰名稱和 IAM 角色。

類型：S3ReferenceDataSourceUpdate 物件

必要：否

TableNameUpdate

此更新所建立的應用程式內資料表名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

S3Configuration

提供 Amazon S3 資料來源的描述，包括 S3 儲存貯體的 Amazon Resource Name (ARN)、用於存取儲存貯體的 IAM 角色 ARN，以及包含資料的 Amazon S3 物件名稱。

目錄

BucketARN

包含該資料的 S3 儲存貯體 ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

FileKey

包含該資料的物件名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

必要：是

RoleARN

用來存取資料之角色的 IAM ARN。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

S3ReferenceDataSource

識別包含參考資料的 S3 儲存貯體和物件。也請識別 Amazon Kinesis Analytics 可擔任的 IAM 角色，代您讀取此物件。

Amazon Kinesis Analytics 應用程式只會載入一次參考資料。如果資料變更，您可以呼叫 UpdateApplication 操作，觸發將資料重新載入到您的應用程式。

目錄

BucketARN

S3 儲存貯體的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

FileKey

包含參考資料的物件索引鍵名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

必要：是

ReferenceRoleARN

服務可擔任代您讀取資料之 IAM 角色的 ARN。此角色必須有物件的 `s3:GetObject` 動作許可，以及讓 Amazon Kinesis Analytics 服務主體擔任此角色的信任政策。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

S3ReferenceDataSourceDescription

提供儲存參考資料的儲存貯體與物件金鑰名稱。

目錄

BucketARN

S3 儲存貯體的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

FileKey

Amazon S3 物件金鑰名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

必要：是

ReferenceRoleARN

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色代您讀取 Amazon S3 物件，以填入應用程式內參考資料表。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

S3ReferenceDataSourceUpdate

描述 Amazon Kinesis 分析可代表您讀取 Amazon S3 物件並填入應用程式內參考資料表的 S3 儲存貯體名稱、物件金鑰名稱和 IAM 角色。

目錄

BucketARNUpdate

S3 儲存貯體的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

FileKeyUpdate

物件金鑰名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

必要：否

ReferenceRoleARNUpdate

Amazon Kinesis Analytics 可擔任的 IAM 角色 ARN，此角色會讀取 Amazon S3 物件，以填入應用程式內。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`arn:.*`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

SourceSchema

描述串流來源中的資料格式，以及每個資料元素如何映射至應用程式內串流中所建立的對應欄位。

目錄

RecordColumns

`RecordColumn` 物件的清單。

類型：RecordColumn 物件陣列

陣列成員：項目數下限為 1。項目數上限為 1000。

必要：是

RecordFormat

指定串流來源中的記錄格式。

類型：RecordFormat 物件

必要：是

RecordEncoding

指定串流來源中的記錄編碼。例如，UTF-8。

類型：字串

模式：`UTF-8`

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

Tag

您可以定義並指派給 AWS 資源的鍵/值對 (值為選用)。如果您指定已存在的標籤，您在請求中指定的值會取代標籤值。請注意，應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。如需詳細資訊，請參閱使用標記。

目錄

Key

索引鍵值標籤的金鑰。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：是

Value

鍵值對標籤的值。此值是選用的。

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- 適用於 Ruby 的 AWS 開發套件第 3 版

Amazon Kinesis Data Analytics 的文件歷史紀錄

下表說明自上次發行 Amazon Kinesis Data Analytics 後，文件的重要變更。

- API 版本：2015-08-14
- 最新文件更新：2019 年 5 月 8 日

變更	描述	日期
標記 Kinesis Data Analytics 應用程式	使用應用程式標記來判定每個應用程式的成本、控制存取或用於使用者定義之目的。如需更多詳細資訊，請參閱 使用標記。	2019 年 5 月 8 日
用 AWS CloudTrail 記錄 Kinesis Data Analytics 的 API 呼叫	Amazon Kinesis Data Analytics 整合了 AWS CloudTrail，後者是一項服務，可提供由使用者、角色或 AWS 服務在 Kinesis Data Streams 中所採取動作的記錄。如需更多詳細資訊，請參閱 使用 AWS CloudTrail。	2019 年 3 月 22 日
Kinesis Data Analytics 服務於法蘭克福地區開放使用	Kinesis Analytics 現已於歐洲 (法蘭克福) 地區開放使用。如需詳細資訊，請參閱和端點：Kinesis Data Analytics。	2018 年 7 月 18 日
在主控台中使用參考資料	您現在可以在主控台中使用應用程式參考資料。如需詳細資訊，請參閱 範例：將參考資料新增至 Kinesis Data Analytics 應用程式。	2018 年 7 月 13 日

變更	描述	日期
視窗化查詢範例	視窗和彙總的範例應用程式。如需詳細資訊，請參閱 範例：視窗與彙總。	2018 年 7 月 9 日
測試應用程式	測試應用程式結構描述和程式碼變更的指引。如需詳細資訊，請參閱 測試應用程式。	2018 年 7 月 3 日
預處理資料的範例應用程式	REGEX_LOG_PARSE、REGEX_REPLACE 和 DateTime 運算子的其他程式碼範例。如需詳細資訊，請參閱 範例：轉換資料。	2018 年 5 月 18 日
傳回資料列和 SQL 程式碼的大小增加	傳回資料列的大小限制增加至 512 KB，而應用程式中 SQL 程式碼的大小限制增加至 100 KB。如需更多詳細資訊，請參閱 限制。	2018 年 5 月 2 日
在 Java 和 .NET 中的 AWS Lambda 函數範例	建立 Lambda 函數以預先處理記錄和應用程式目的地的程式碼範例。如需詳細資訊，請參閱建立 Lambda 函數以進行預處理及為應用程式目的地建立 Lambda 函數。	2018 年 3 月 22 日
新 HOTSPOTS 函數	尋找並傳回資料中相對密集區域的相關資訊。如需更多詳細資訊，請參閱 範例：偵測串流上的熱點 (熱點功能)。	2018 年 3 月 19 日
作為目的地的 Lambda 函數	將分析結果作為目的地傳送至 Lambda 函數。如需更多詳細資訊，請參閱 使用 Lambda 函數作為輸出。	2017 年 12 月 20 日

變更	描述	日期
新 RANDOM_CUT_FOREST_WITH_EXPLANATION 函數	說明哪些欄位造就了資料串流中的異常分數。如需更多詳細資訊，請參閱 範例：偵測資料異常並取得說明 (RANDOM_CUT_FOREST_WITH_EXPLANATION 函數)。	2017 年 11 月 2 日
靜態資料的結構描述探索	對 Amazon S3 儲存貯體中存放的靜態資料執行結構描述探索。如需更多詳細資訊，請參閱 在靜態資料上使用結構描述探索功能。	2017 年 10 月 6 日
Lambda 預處理功能	在分析前用 AWS Lambda 預處理輸入串流中的記錄。如需更多詳細資訊，請參閱 使用 Lambda 函數預處理資料。	2017 年 10 月 6 日
自動擴展應用程式	使用自動擴展功能自動增加應用程式的資料輸送量。如需更多詳細資訊，請參閱 自動擴展應用程式以增加輸送量。	2017 年 9 月 13 日
多個應用程式內輸入串流	利用多個應用程式內串流增加應用程式輸送量 如需更多詳細資訊，請參閱 平行化輸入串流以提高輸送量。	2017 年 6 月 29 日
Kinesis Data Analytics 的 AWS Management Console 使用指南	使用 Kinesis Data Analytics 主控台中的結構描述編輯器和 SQL 編輯器，編輯推斷結構描述和 SQL 程式碼。如需更多詳細資訊，請參閱 步驟 4 (選用)：使用主控台編輯結構描述和 SQL 程式碼。	2017 年 4 月 7 日

變更	描述	日期
公開發行	Amazon Kinesis Data Analytics 開發人員指南的公開發行版本。	2016 年 8 月 11 日
預覽版	Amazon Kinesis Data Analytics 開發人員指南的預覽版本。	2016 年 1 月 29 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。