

Azure Data Engineer Associate Certification Guide

A hands-on reference guide to developing your data engineering skills and preparing for the DP-203 exam

Newton Alex



Capítulo 10: Diseño y desarrollo de una solución de procesamiento stream	4
10.1. Requisitos técnicos.....	5
10.2. Diseño de una solución de procesamiento stream.....	5
10.2.1. Presentación de Azure Event Hubs	6
Particiones de Event Hubs.....	7
Grupos de consumidores de Event Hubs	7
10.2.2. Presentación de ASA	8
10.2.3. Presentación de Spark Streaming	8
10.3. Desarrollo de una solución de procesamiento de flujos utilizando ASA, Azure Databricks y Azure Event Hubs	9
10.3.1. Una solución de streaming utilizando Event Hubs y ASA.....	9
10.3.2. Una solución de streaming utilizando Event hubs y Spark Streaming.....	21
10.4. Procesamiento de datos con Spark Structured Streaming	23
10.5. Monitorización de rendimiento y regresiones funcionales	25
10.5.1. Monitorización en Event Hubs	25
10.5.2. Monitorización en ASA.....	26
Configuración de alertas en ASA	27
10.5.3. Monitorización en Spark Streaming.....	28
10.6. Procesamiento de datos de series temporales	30
10.6.1. Tipos de timestamps	30
10.6.2. Agregados con ventanas	30
10.6.3. Checkpointing o watermarking	30
10.6.4. Reproducir datos de un timestamp anterior	31
10.7. Diseñar y crear agregados con ventanas	32
10.7.1. Tumbling windows	32
10.7.2. Hopping windows.....	33
10.7.3. Sliding windows.....	34
10.7.4. Session windows	35
10.7.5. Snapshot windows	36
10.8. Configuración de los checkpoints/watermarking durante el procesamiento.....	37
10.8.1. Checkpointing en ASA	37
10.8.2. Checkpointing en Event Hubs	37
10.8.3. Checkpointing en Spark.....	38

10.9. Reproducir datos de stream archivados	39
10.10. Transformaciones usando streaming analytics.....	40
10.10.1. Las transformaciones COUNT y DISTINCT	40
10.10.2. Transformaciones CAST.....	40
10.10.3. Transformaciones LIKE	40
10.11. Manejo de los schema drifts	42
10.11.1. Manejando schema drifts usando Event Hubs.....	42
Registro de un esquema con schema registry	42
Recuperación de un esquema del Registro de Esquemas.....	43
10.11.2. Manejo de schema drifts en Spark.....	44
10.12. Procesamiento entre particiones	45
10.12.1. ¿Qué son las particiones?	45
10.12.2. Procesando datos a través de las particiones	45
10.13. Procesamiento dentro de una partición	47
10.14. Escalando recursos.....	48
10.14.1. Escalado en Event Hubs	48
¿Qué son las unidades de rendimiento (throughput units)?	48
10.14.2. Escalando en ASA	49
10.14.3. Escalado en Azure Databricks Spark Streaming	51
10.15. Manejo de las interrupciones	52
10.15.1. Manejo de interrupciones en Event Hubs.....	52
10.15.2. Manejo de interrupciones en ASA	54
10.16. Diseñando y configurando el manejo de excepciones.....	55
10.17. Upserting data.....	56
10.18. Diseño y creación de pruebas para data pipelines	58
10.19. Optimización de pipelines para fines analíticos o transaccionales.....	58
Resumen.....	59

Capítulo 10: Diseño y desarrollo de una solución de procesamiento stream

Bienvenido al siguiente capítulo de la serie de transformación de datos. Este capítulo trata de las soluciones de procesamiento stream, también conocidas como sistemas de procesamiento en tiempo real. Al igual que el procesamiento por lotes, el procesamiento stream es otro segmento importante de los data pipelines. Este es también un capítulo muy importante para su certificación.

Este capítulo se centrará en la introducción de los conceptos y tecnologías involucradas en la construcción de un sistema de procesamiento stream. Aprenderás sobre tecnologías como Azure Stream Analytics (ASA), Azure Event Hubs y Spark (desde la perspectiva del stream). Aprenderá a construir soluciones de stream de extremo a extremo utilizando estas tecnologías. Además, aprenderá conceptos importantes de streaming como el checkpointing, los agregados en ventana, la repetición de datos de stream antiguos, handling drift y conceptos de gestión de stream como la distribución de stream en particiones, el escalado de recursos, el manejo de errores y la inserción de datos.

Una vez que haya completado este tema, debería tener la suficiente confianza para construir un pipeline de stream de extremo a extremo utilizando las tecnologías que están disponibles en Azure.

Al igual que en los demás capítulos, me he tomado la libertad de reordenar los temas del programa de estudios para este capítulo agrupando los relacionados entre sí. Esto ayuda a crear un buen flujo para el capítulo. Vamos a cubrir los siguientes temas:

- Diseño de una solución de procesamiento stream
- Desarrollo de una solución de procesamiento stream usando ASA, Azure Databricks y Azure Event Hubs
- Procesamiento de datos usando Spark Structured Streaming
- Monitorización de rendimiento y regresiones funcionales
- Procesamiento de datos de series temporales
- Diseño y creación de agregados con ventanas
- Configuración de checkpoints/watermarking durante el procesamiento
- Reproducción de datos de stream archivados
- Gestión de las desviaciones del esquema
- Procesamiento entre particiones
- Procesamiento dentro de una partición
- Escalado de recursos
- Manejo de interrupciones
- Diseño y configuración de la gestión de excepciones
- Upserting data
- Diseño y creación de pruebas para data pipelines
- Optimización de pipelines para fines analíticos o transaccionales

10.1. Requisitos técnicos

Para este capítulo, necesitarás lo siguiente

Una cuenta de Azure (puede ser gratuita o de pago)

La capacidad de leer código Python básico (no te preocupes, es muy fácil)

¡Empecemos!

10.2. Diseño de una solución de procesamiento stream

Los sistemas de procesamiento stream o sistemas de procesamiento en tiempo real son sistemas que realizan el procesamiento de datos casi en tiempo real. Piense en las actualizaciones del mercado de valores, las actualizaciones del tráfico en tiempo real, la detección de fraudes con tarjetas de crédito en tiempo real, etc. Los datos entrantes se procesan en el momento en que llegan, con una latencia mínima, que suele oscilar entre milisegundos y segundos. En el capítulo 2, Diseño de una estructura de almacenamiento de datos, conocimos la arquitectura del data lake, donde vimos dos ramas de procesamiento: una para el stream y otra para el batch processing. En el capítulo anterior, el capítulo 9, Diseño y desarrollo de una solución de procesamiento por lotes, nos centramos en el pipeline de procesamiento por lotes. En este capítulo, nos centraremos en el procesamiento stream. Las cajas azules en el siguiente diagrama muestran el pipeline de stream:

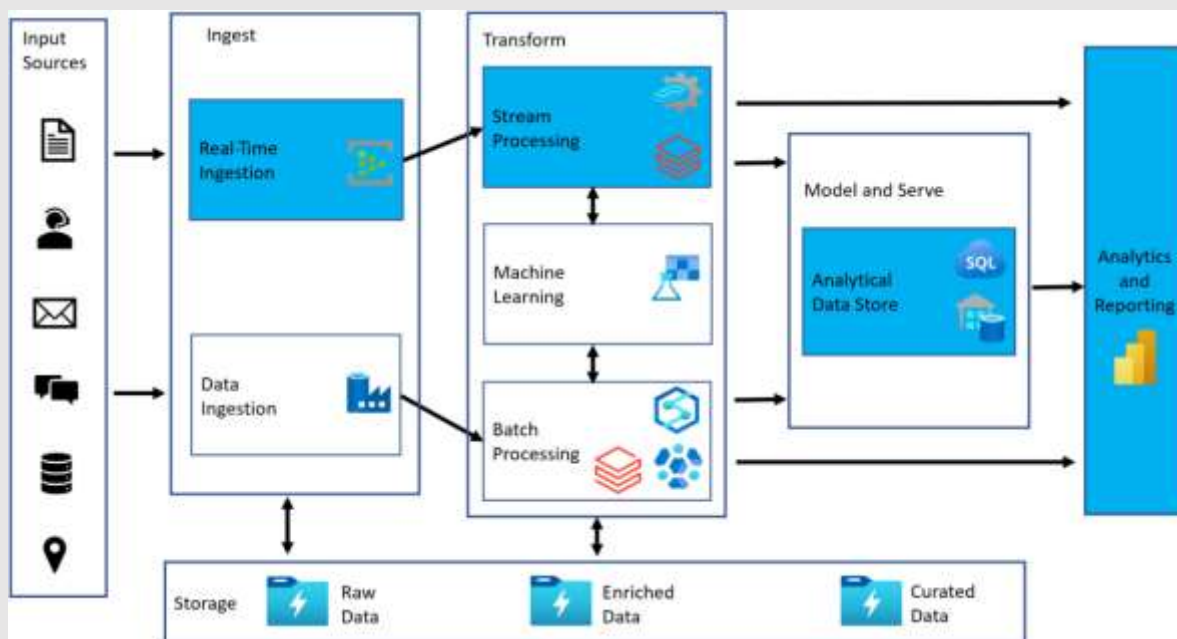


Figura 10.1 - La arquitectura de procesamiento stream

Los sistemas de procesamiento stream constan de cuatro componentes principales:

- Un **servicio de ingestión de eventos**, como **Azure Event Hubs**, **Azure IoT Hub**, **Apache Kafka**, etc., que ayuda a capturar, almacenar y transmitir eventos de múltiples fuentes de entrada a los servicios de procesamiento stream. Actúan como un amortiguador entre las fuentes de entrada y los servicios de procesamiento de datos. A veces, la tasa de entrada de eventos puede ser demasiado para que los sistemas de procesamiento stream puedan manejarla, y a veces, puede ser demasiado poco. Si la tasa de entrada de eventos es mayor, el servicio de ingestión almacenaría los eventos hasta que los servicios posteriores puedan procesarlos. Si la tasa de entrada de eventos es menor, el servicio de ingesta puede agruparlos por lotes para hacerlos más eficientes y enviarlos downstream.
- Los **sistemas de procesamiento stream**, como **ASA**, **Spark Streaming**, **Apache Storm**, **Apache Flink**, etc., ofrecen la posibilidad de filtrar, unir y agregar rápidamente los eventos entrantes para obtener información. Son muy similares a las transformaciones por lotes, pero trabajan con tamaños de datos mucho más pequeños y con mayor frecuencia que los sistemas por lotes.
- **Almacenes de datos analíticos** como **Synapse Dedicated SQL pool**, **CosmosDB**, **HBase**, etc., donde se suelen almacenar los datos procesados de los sistemas de procesamiento stream para la elaboración de informes de BI y otras consultas ad hoc. Algunos sistemas como ASA pueden enviar directamente los datos a los sistemas de informes como Power BI sin necesidad de un almacén de datos analíticos.
- **Sistemas de informes** como **Power BI**. Esta es la pieza final del pipeline donde se presentan los datos procesados en forma de informes para el consumo del negocio y la toma de decisiones.

Ahora, vamos a ver los servicios que están disponibles en Azure para construir sistemas de procesamiento stream. Empezaremos con Azure Event Hubs.

10.2.1. Presentación de Azure Event Hubs

Azure Event Hubs es un servicio de ingesta distribuido que puede ingerir, almacenar y transferir millones de eventos desde varias fuentes de entrada a múltiples consumidores. Actúa como un amortiguador entre los productores de eventos y los consumidores de eventos y desacopla los productores de eventos de los consumidores de eventos. Esto ayuda a los componentes de stream posteriores, como ASA o Spark Streaming, a procesar los datos de forma asíncrona. Azure Event Hubs es un servicio PaaS totalmente gestionado, por lo que no tenemos que preocuparnos del mantenimiento del servicio. Event Hubs puede autoinflarse para satisfacer los crecientes requisitos del sistema de streaming.

Este es un diagrama simplificado de la arquitectura de Azure Event Hubs:

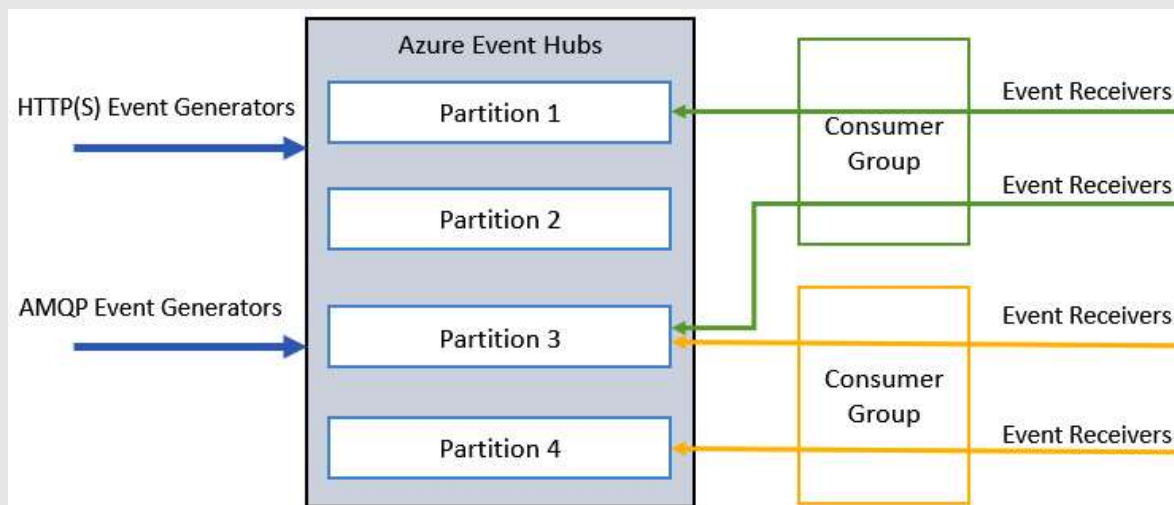


Figura 10.2 - La arquitectura de Azure Event Hubs

Como puedes ver en el diagrama anterior, los Event Hubs pueden recibir entradas a través de dos protocolos: el Protocolo de Transferencia de Hipertexto (HTTP) o el Protocolo Avanzado de Cola de Mensajes (AMQP). A continuación, distribuye los datos en particiones. Los receptores de eventos, que forman parte de grupos de consumidores, pueden suscribirse a las particiones y leer los eventos desde allí. Veamos qué significan estas terminologías, incluidas las particiones y los grupos de consumidores.

Particiones de Event Hubs

Event Hubs distribuye los datos entrantes en una o más particiones, como se muestra en la Figura 10.2. Las particiones ayudan a escalar horizontalmente, ya que permiten que varios consumidores lean los datos en paralelo.

Grupos de consumidores de Event Hubs

Un grupo de consumidores es una vista del hub de eventos. Puede haber muchos grupos de consumidores para cada hub de eventos, y cada grupo de consumidores puede tener su propia vista del hub de eventos. En otras palabras, tienen acceso a diferentes conjuntos de stream o particiones. Los consumidores (normalmente, las aplicaciones posteriores) acceden a las particiones a través de su propio grupo de consumidores. El grupo de consumidores mantiene los desplazamientos de estado en el stream, la información de checkpointing, etc. Las aplicaciones dentro de un grupo de consumidores pueden procesar los datos de forma independiente sin preocuparse de otros clientes.

Puedes saber más sobre los Event Hubs en <https://docs.microsoft.com/en-us/azure/event-hubs/>.

A continuación, vamos a ver ASA.

10.2.2. Presentación de ASA

ASA es el principal motor de procesamiento stream de Azure. Puede procesar grandes volúmenes de datos con latencias mínimas. Se puede utilizar para realizar funciones analíticas como filtrar, agregar, unir y más para obtener resultados rápidos del stream de datos entrante. ASA puede utilizarse para escenarios como el análisis de puntos de venta, la detección de fraudes con tarjetas de crédito, la detección de fallos en el IoT, el análisis de flujos de clics, etc. Normalmente, un job de ASA tiene tres etapas:

1. Lee desde un servicio de ingestión como Event Hubs, IoT Hub o Kafka.
2. Procesa los datos y genera información.
3. Por último, escribe los datos en un almacén analítico, como Azure Synapse Analytics, CosmosDB, o envía los resultados directamente a un servicio de informes de BI. ASA puede enviar directamente los resultados a Power BI.

Al igual que Event Hubs, ASA es también un servicio PaaS totalmente gestionado. Por lo tanto, no tenemos que preocuparnos de sus actualizaciones y parches.

Puede obtener más información sobre ASA en <https://docs.microsoft.com/en-us/azure/stream-analytics/>.

A continuación, veamos Spark Streaming.

10.2.3. Presentación de Spark Streaming

En los capítulos anteriores, hemos visto a Spark desde una perspectiva de procesamiento por lotes, pero lo que quizás no sepas es que puedes utilizar las mismas APIs del núcleo de Apache Spark para construir una solución de stream. Al igual que ASA, Spark también puede leer desde servicios de ingesta de datos como Azure Event Hubs, Kafka y otros, implementar las transformaciones de datos y, finalmente, escribir la salida en bases de datos analíticas o en cualquier otro almacén.

Spark Streaming divide internamente el stream entrante en micro-lotes y los procesa, por lo que la salida será un stream de micro-lotes.

Puede obtener más información sobre Spark Streaming en <https://spark.apache.org/docs/latest/streaming-programming-guide.html>.

A continuación, vamos a ver cómo construir una solución de procesamiento stream utilizando todas las tecnologías que hemos aprendido hasta ahora.

10.3. Desarrollo de una solución de procesamiento de flujos utilizando ASA, Azure Databricks y Azure Event Hubs

En esta sección, veremos dos ejemplos: uno con ASA como motor de streaming y otro con Spark como motor de streaming. Utilizaremos un generador de eventos ficticio para generar continuamente eventos de viaje. Configuraremos tanto ASA como Azure Databricks Spark para realizar el procesamiento en tiempo real y publicar los resultados. En primer lugar, vamos a empezar con ASA.

10.3.1. Una solución de streaming utilizando Event Hubs y ASA

En este ejemplo, vamos a crear un pipeline de streaming mediante la creación de una instancia de Event Hubs, una instancia de ASA, y la vinculación de los mismos. El pipeline leerá un stream de eventos de muestra, procesará los datos y mostrará el resultado en Power BI:

1. En primer lugar, vamos a crear una instancia de Event Hub. Desde el portal de Azure, busca Event Hubs y haz clic en el botón Create, como se muestra en la siguiente captura de pantalla:

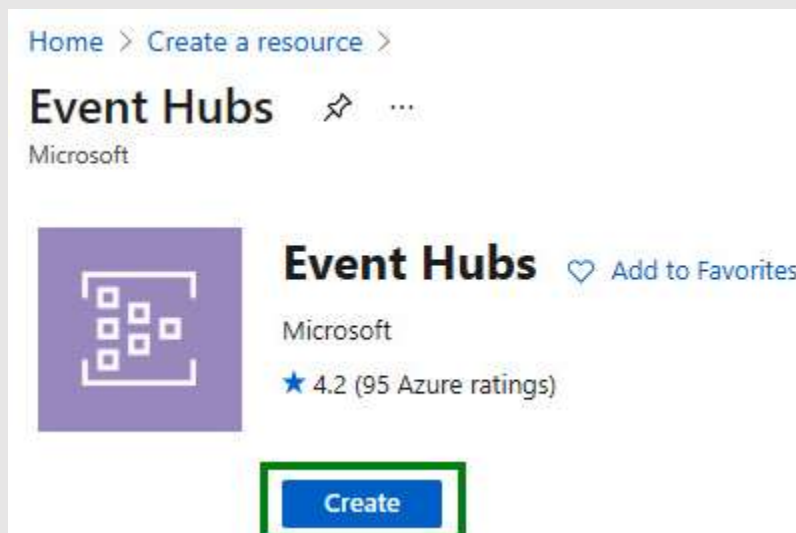


Figura 10.3 - Pantalla de creación de Event Hubs

2. Esto hará que aparezca la pantalla de Event Hubs Create Namespace, como se muestra en la siguiente captura de pantalla. Rellene los detalles, incluyendo el grupo de recursos, el nombre del Namespace, la ubicación, las unidades de rendimiento y cualquier otro campo necesario:

Home > Create a resource > Event Hubs >

Create Namespace

Event Hubs

Basics Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure subscription 1

Resource group * [Create new](#)

Instance Details

Enter required settings for this namespace, including a price tier and configuring the number of units (capacity).

Namespace name * DP203EHWS [.servicebus.windows.net](#)

Location * East US

The region selected supports Availability zones. Your namespace will have Availability Zones enabled. [Learn more.](#)

Pricing tier (View full pricing details) *

Throughput Units * 1

[Review + create](#) < Previous Next: Tags >

Figura 10.4 - Creación del Event Hubs namespace

- Introduzca los detalles y haga clic en el botón Revisar + crear para crear el área de trabajo Event Hubs. Una vez creado el área de trabajo, haga clic en el enlace + Event Hub (como se muestra en la siguiente captura de pantalla) para crear un nuevo event hub dentro del área de trabajo:

[+ Event Hub](#) [Delete](#) [Refresh](#)

Essentials

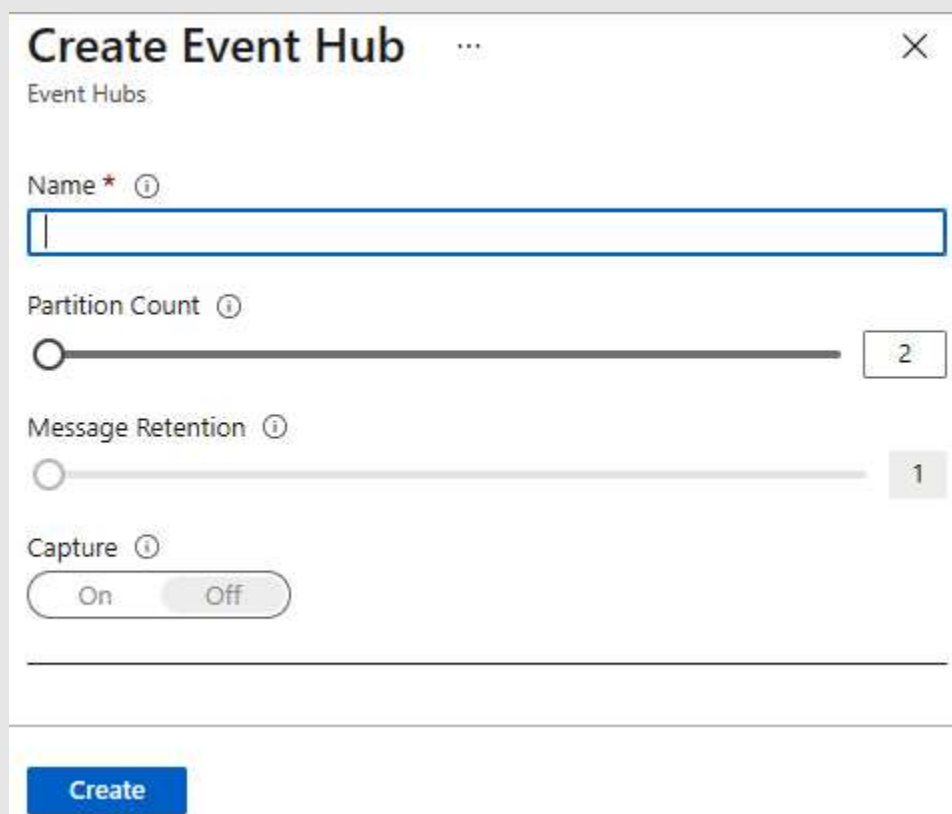
NAMESPACE CONTENTS 0 EVENT HUBS

KAFKA SURFACE NOT SUPPORTED

ZONE REDUNDANCY ENABLED

Figura 10.5 - Creación de un event hub desde el área de trabajo de Event Hubs

4. Esto hará que aparezca una pantalla, como se muestra en la siguiente captura de pantalla. Introduzca un Nombre en el campo dado, seleccione el valor de Recuento de Particiones (aprenderemos más sobre las particiones más adelante), y haga clic en Crear:



Create Event Hub ...

Event Hubs

Name * ⓘ

Partition Count ⓘ

Message Retention ⓘ

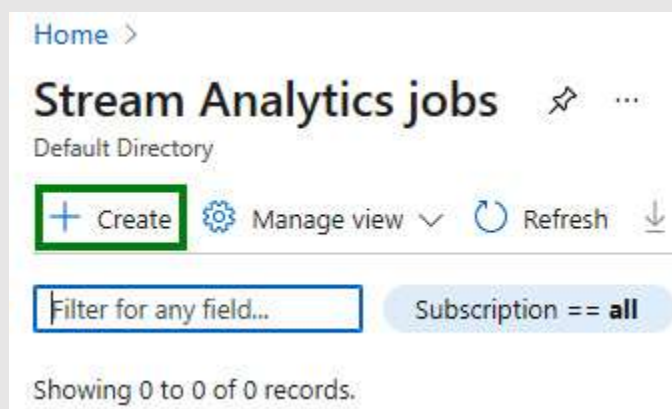
Capture ⓘ

On Off

Create

Figura 10.6 - Pantalla de creación de event hub

5. Ahora que tenemos un event hub, vamos a crear una instancia de ASA. Busca los trabajos de Stream Analytics en el portal de Azure y selecciona la opción. En la página de trabajos de Stream Analytics, haz clic en el enlace + Create:



Home >

Stream Analytics jobs ⭐ ...

Default Directory

+ Create ⚙️ Manage view ▾ ↻ Refresh ⬇

Filter for any field... Subscription == all

Showing 0 to 0 of 0 records.

Figura 10.7 - Creación de un job ASA

6. Esto hará que aparezca la pantalla de creación de jobs de análisis, como se muestra en la siguiente captura de pantalla. Rellene los detalles, incluyendo el nombre del job, la suscripción, el grupo de recursos, la ubicación, las unidades de streaming y cualquier otro campo necesario. A continuación, haga clic en el botón Crear para crear un nuevo job ASA:

Home > Stream Analytics jobs >

New Stream Analytics job

This will create a new Stream Analytics job. You will be charged according to Azure Stream Analytics billing model. [Learn more.](#)

Job name *

Subscription *

Resource group *

Create new

Location *

Hosting environment ⓘ

Cloud Edge

Streaming units (1 to 192) ⓘ

☐ Secure all private data assets needed by this job in my Storage account. ⓘ

Create

Figura 10.8 - Pantalla de creación de ASA

7. El siguiente paso es vincular la entrada a Event Hubs. Antes de este paso, necesitaremos obtener la cadena de conexión de Event Hubs. Ve a la página de Event Hubs, y haz clic en Shared access policies. Haga clic en el botón + Añadir, y añada una nueva política, como se muestra en la siguiente captura de pantalla:

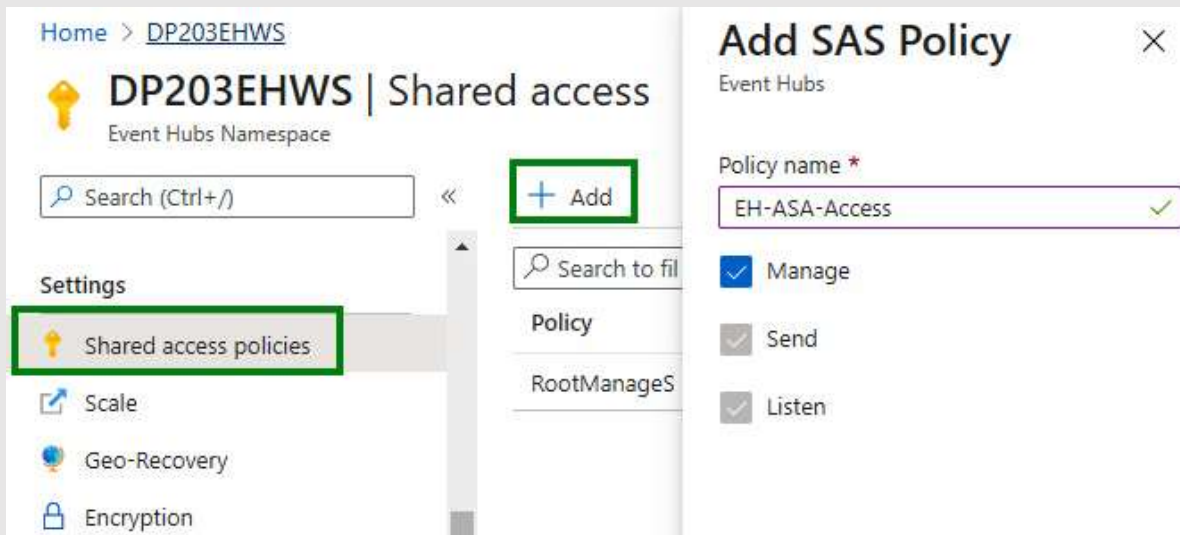


Figura 10.9 - Creación de políticas de acceso compartido en Event Hubs

- Una vez creada la política, basta con hacer clic en la política para recuperar el enlace de la cadena de conexión, como se muestra en la siguiente captura de pantalla:

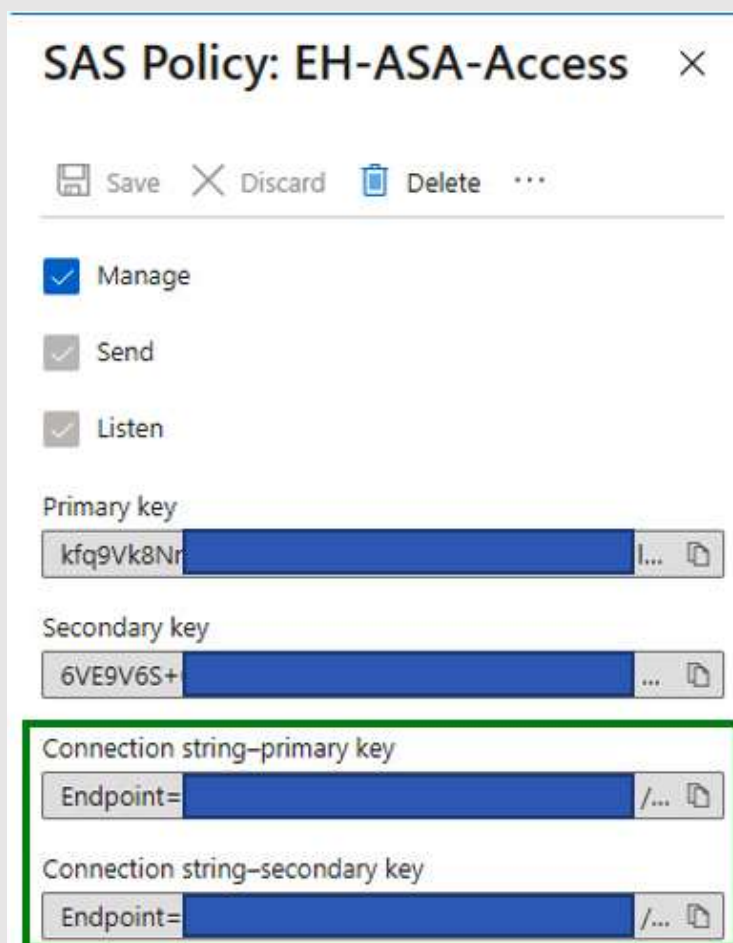


Figura 10.10 - Accediendo a la cadena de conexión para Event Hubs

9. Ahora, vuelva al portal de ASA y vincule la entrada (Event Hubs) y la salida (Power BI) a la instancia de ASA. Haga clic en la pestaña Entradas y seleccione Event Hub en la lista desplegable + Añadir entrada de stream. Esto hará que aparezca una pantalla en la que puede seleccionar la instancia de Event hub que creó anteriormente:

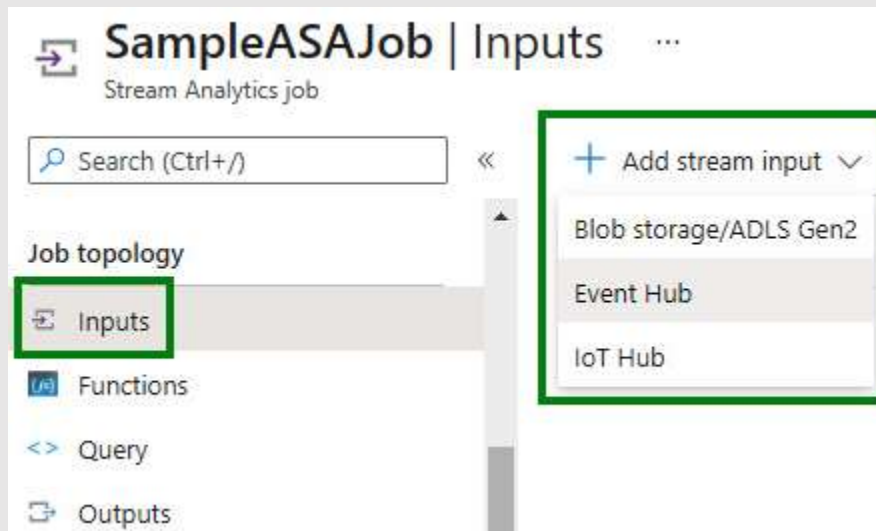


Figura 10.11 - Selección de un event hub como entrada para el job ASA

10. En la pantalla Event Hub New input, introduzca la cadena de conexión que ha copiado de la Figura 10.10:

The image shows the 'Event Hub New input' configuration form. It includes the following fields and options: 'input alias' (text field), 'Subscription' (dropdown menu showing 'Azure subscription 1'), 'Event Hub namespace' (dropdown menu showing 'DP203EHWS'), 'Event Hub name' (dropdown menu with 'Create new' and 'Use existing' radio buttons; 'Use existing' is selected, showing 'asaeh'), 'Event Hub consumer group' (dropdown menu with 'Create new' and 'Use existing' radio buttons; 'Use existing' is selected, showing '\$Default'), 'Authentication mode' (dropdown menu showing 'Connection string', highlighted with a green box), 'Event Hub policy name' (dropdown menu with 'Create new' and 'Use existing' radio buttons; 'Use existing' is selected, showing 'EH-ASA-Access'), 'Event Hub policy key' (text field with masked characters, highlighted with a green box), 'Partition key' (text field), and 'Event serialization format' (dropdown menu showing 'JSON'). A 'Save' button is at the bottom.

Figura 10.12 - Vinculación del event hub como entrada en ASA

11. Del mismo modo, haga clic en la pestaña Salidas y seleccione Power BI para la salida de ASA:

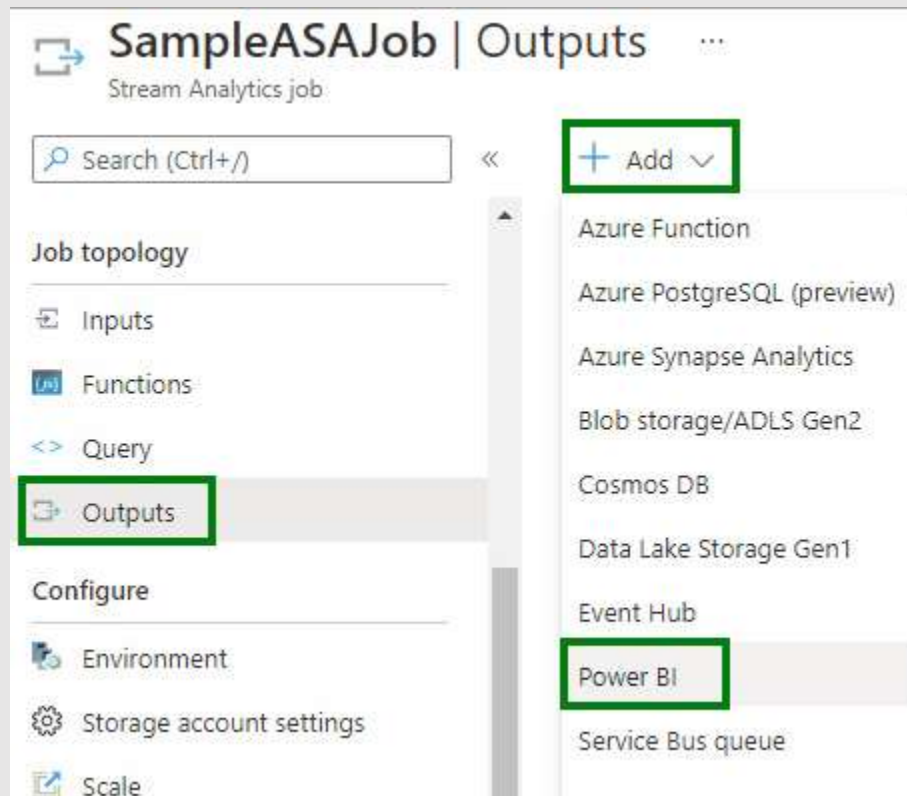


Figura 10.13 - Seleccionando Power BI como salida para el job de ASA

12. Esto hará que aparezca una pantalla como la siguiente. Aquí, tendrá que rellenar sus detalles de Power BI:

Figura 10.14 - Configuración de los detalles del sink de Power BI para el job ASA

13. Ahora que tenemos la entrada y la salida configuradas y listas, ejecute un generador de eventos de ejemplo para generar eventos de prueba. Aquí tenemos un sencillo módulo generador de eventos en Python. El código completo está disponible en el repositorio GitHub adjunto.

a) Importa las librerías Event hub necesarias:

```
from azure.eventhub.aio import EventHubProducerClient
from azure.eventhub import EventData
```

b) Instanciar un cliente productor:

```
producer = EventHubProducerClient.from_connection_string(
    conn_str=<SAS Access Connection String>,
    eventhub_name=<Event Hub Name>)
```

c) Crear una instancia de lote de eventos:

```
eventDataBatch = await producer.create_batch()
```

d) Crear un evento JSON:

```
cityList = ["San Francisco", "San José", "Los Ángeles",...]
tripDetail = {'tripId': str(uuid.uuid4()),
              'timestamp': str(datetime.datetime.utcnow()),
              'startLocation': random.choice(cityList),
              'endLocation': random.choice(cityList),
              'distancia': random.randint(10, 1000),
              'fare': random.randint(100, 1000) }
```

e) Añade los eventos al lote:

```
eventDataBatch.add(
    EventData(json.dumps(tripDetail)))
```

f) Envía el lote de eventos al event hub:

```
producer.send_batch(eventDataBatch)
```

Puedes repetir los pasos d, e y f en un bucle for durante el tiempo que necesites.

14. Cuando ejecutes el código anterior desde la línea de comandos, podrás ver los eventos que llegan al event hub en la página de resumen de Event Hubs, como se muestra a continuación:

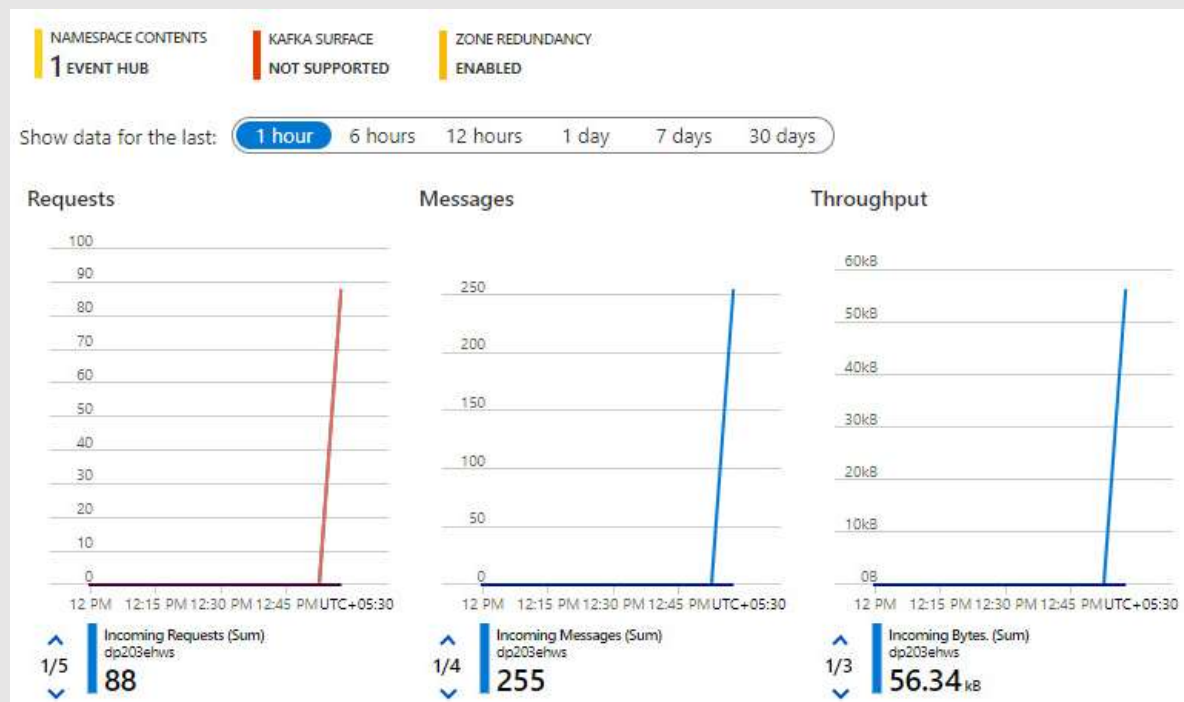


Figura 10.15 - Las páginas de resumen de los event hubs mostrando las métricas de los eventos

15. Ahora, vamos a leer estos datos y publicar el número de viajes por ubicación. En la página de resumen de ASA, puede introducir la consulta como se muestra en la siguiente captura de pantalla, y hacer clic en Iniciar:

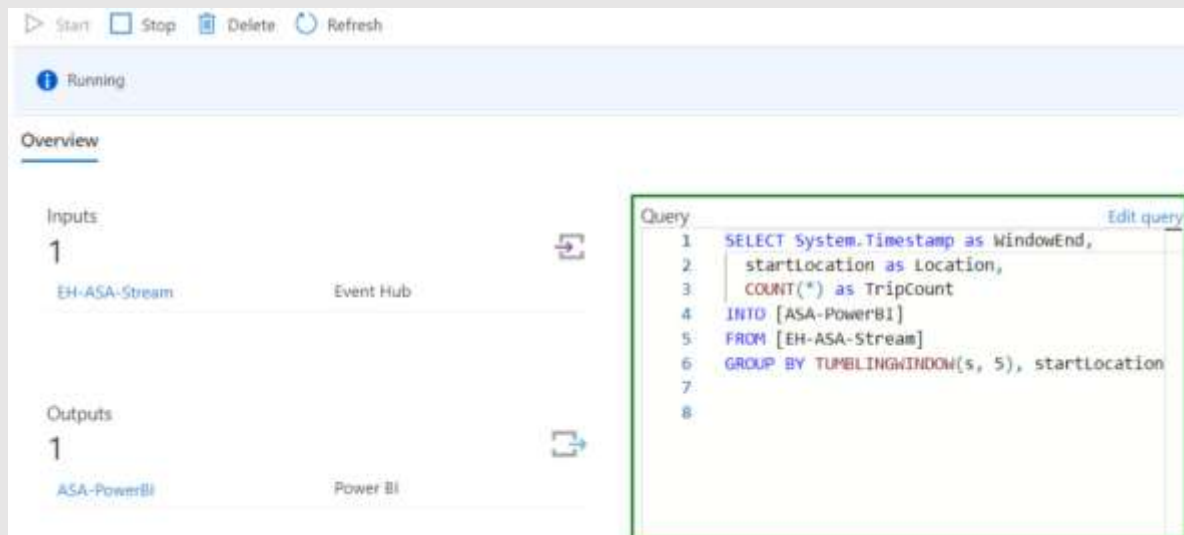


Figura 10.16 - El código de ejemplo de ASA

16. Ahora, el job ASA leerá continuamente el stream del event hub de entrada, procesará los datos y los publicará en Power BI. Si navega a Mi área de trabajo en su portal de Power BI, debería ver el dataset de ASA que fue configurado anteriormente en la pantalla de configuración de salida de Power BI (Figura 10.14):

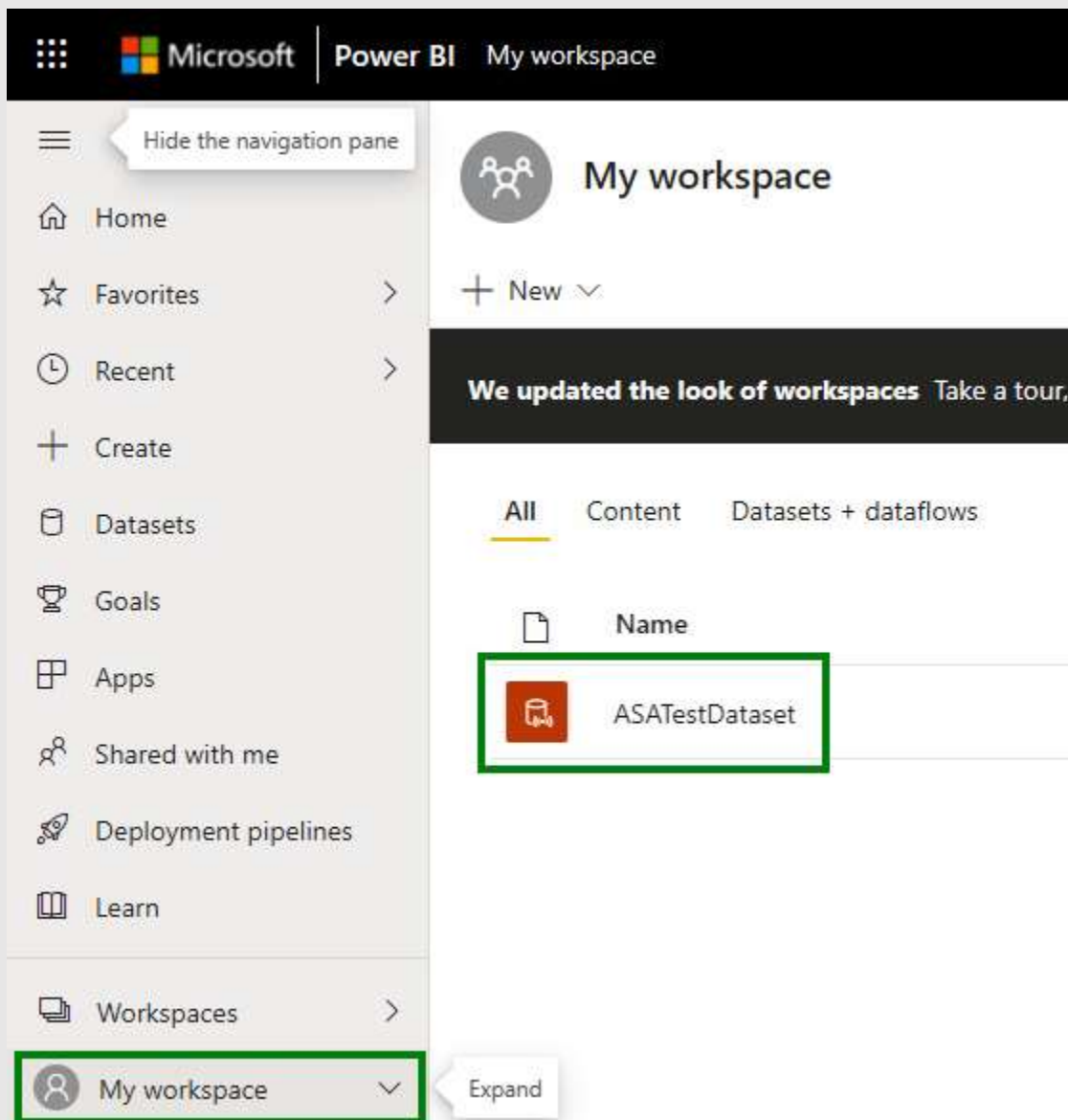


Figura 10.17 - Pantalla del dataset de Power BI

17. Puede hacer clic con el botón derecho en el dataset y crear un informe a partir de él, como se muestra en la siguiente captura de pantalla:

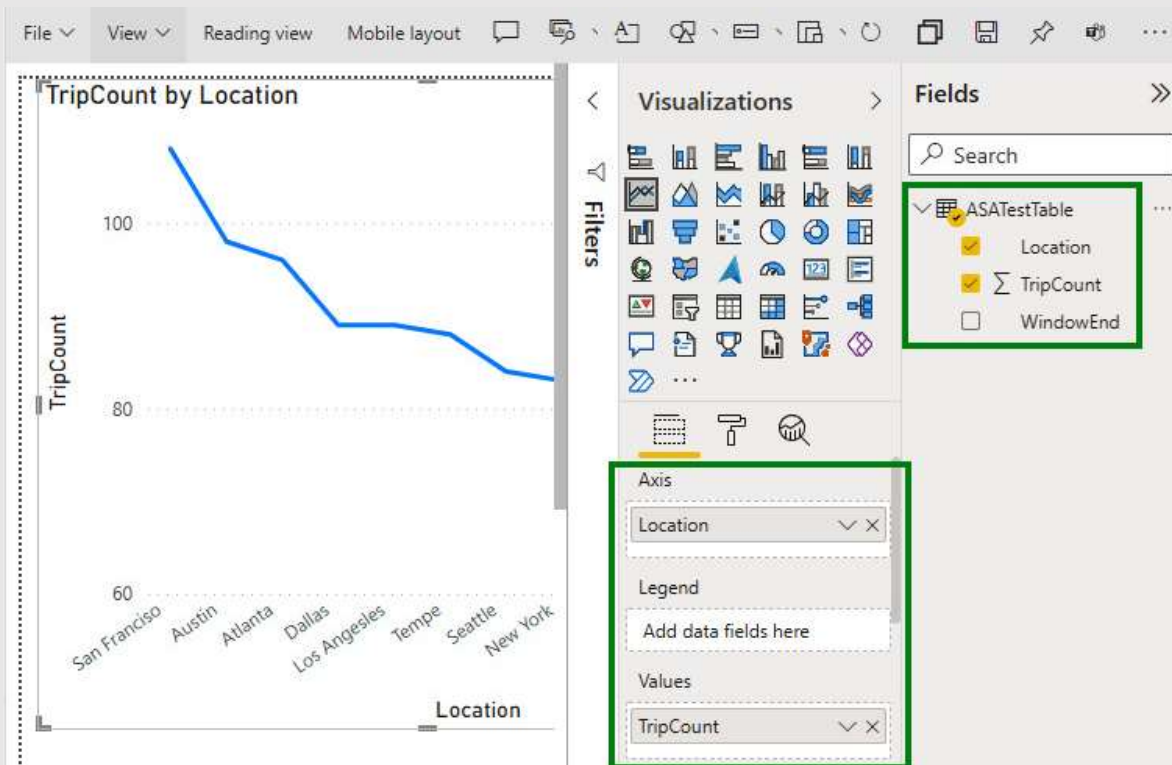


Figura 10.18 - Creación del dashboard de Power BI a partir de la salida de streaming de ASA

Ahora ya sabes cómo construir un pipeline de streaming usando Event hubs, ASA y Power BI.

ASA también puede trabajar con IoT Hub como servicio de ingesta. Con IoT Hub, ASA puede ser desplegado en dos modos diferentes:

- En el **modo Cloud**: Aquí, los dispositivos IoT envían los eventos a un trabajo de ASA en la nube de Azure, que es muy similar al modelo de Event Hubs.
- En **modo Edge**: Aquí, ASA puede ejecutarse directamente en los propios dispositivos IoT, realizar el procesamiento en tiempo real y enviar los eventos a IoT Hub.

Puede encontrar más detalles sobre el modo Edge de ASA en <https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-deploy-stream-analytics?view=iotedge-2020-11>.

A continuación, vamos a ver cómo construir una solución de streaming utilizando Azure Databricks Spark.

10.3.2. Una solución de streaming utilizando Event hubs y Spark Streaming

En este ejemplo, utilizaremos el mismo event hub y el mismo generador de datos que utilizamos para la opción ASA. El único cambio es que utilizaremos Azure Databricks Spark en lugar de ASA. Veamos cómo leer datos de Event hubs utilizando Databricks Spark.

Antes de poder conectarnos a Event Hubs y empezar a leer, tendremos que crear un cluster de Spark. Aprendimos a crear un clúster de Databricks Spark en el Capítulo 9, Diseño y desarrollo de una solución de procesamiento por lotes, en la sección Desarrollo de soluciones de procesamiento por lotes utilizando Data Factory, Data Lake, Spark, Azure Synapse Pipelines, PolyBase y Azure Databricks. Utilice las mismas instrucciones para crear un nuevo clúster de Spark:

1. Una vez que el clúster de Spark esté en funcionamiento, abra un Notebook de Spark e introduzca el siguiente código de ejemplo para procesar los datos de stream en el notebook.
2. Establezca la conexión a la instancia de Event Hubs de la misma manera que la generamos en la Figura 10.10, como se muestra en el siguiente bloque de código:

```
EHConnectionString = "<EVENT HUB CONNECTION STRING>"
EHConfig = {}
EHConfig['eventhubs.connectionString'] = sc._jvm.org.apache.spark.eventhubs.EventHubsUtils
.encrypt( EHConnectionString )
```

3. Conecta con el event hub:

```
EHStreamDF = spark.readStream.format("eventhubs") .options(**EHConfig).load()
```

4. A continuación, tenemos que definir el esquema de la entrada de streaming:

```
JsonSchema = StructType() \
.add("tripId", StringType()) \
.add("createdAt", TimestampType()) \
.add("startLocation", StringType()) \
.add("endLocation", StringType()) \
.add("distance", IntegerType()) \
.add("fare", IntegerType())
```

5. Ahora, vamos a definir el DataFrame que maneja el stream (EHStreamJsonDF) para extraer los valores clave de los eventos JSON entrantes:

```
stringDF = EHStreamDF.selectExpr("CAST(body AS STRING)")
jsonDF= stringDF.withColumn('tripjson', from_json(col('body'),schema=JsonSchema))
EHStreamJsonDF= jsonDF.select("tripjson.*")
```

6. Hasta este punto, nos hemos asegurado de que los eventos del event hub puedan ser procesados y actuados directamente usando DataFrames. El siguiente paso es definir la transformación que se va a aplicar y comenzar realmente el streaming:

```
EHStreamJsonDF.groupBy(window('createdAt','1
minutes'),'startLocation').count().orderBy('window')\
.writeStream.format("memory") \
.outputMode("complete") \
.option("truncate", "false") \
.option("checkpointLocation", "dbfs:/tripsCheckpointLocation/") \
.queryName("TripsTumblingQuery").start()
```

- En el paso anterior, estamos utilizando una ventana de 1 minuto y contando el número de viajes basados en startLocation. En las próximas secciones, aprenderemos acerca de las ventanas, los diferentes modos de salida, los puntos de control, y más.
- Una vez que ejecutamos el Paso 6, el streaming se inicia y busca continuamente eventos del event hub, los procesa, y envía la salida a una tabla llamada TripsTumblingQuery. Si se realiza SELECT * from the table, se pueden ver los recuentos de viajes:

Table		Data Profile	
	window	startLocation	count
1	{ "start": "2021-12-06T11:52:00.000+0000", "end": "2021-12-06T11:53:00.000+0000" }	San Francisco	4
2	{ "start": "2021-12-06T11:52:00.000+0000", "end": "2021-12-06T11:53:00.000+0000" }	Dallas	2
3	{ "start": "2021-12-06T11:52:00.000+0000", "end": "2021-12-06T11:53:00.000+0000" }	Atlanta	4
4	{ "start": "2021-12-06T11:52:00.000+0000", "end": "2021-12-06T11:53:00.000+0000" }	Tempe	8
5	{ "start": "2021-12-06T11:52:00.000+0000", "end": "2021-12-06T11:53:00.000+0000" }	San Jose	1
6	{ "start": "2021-12-06T11:52:00.000+0000", "end": "2021-12-06T11:53:00.000+0000" }	Denver	2

Showing all 14 rows

Figura 10.19 - Visualización de los resultados de la consulta de streaming

- Puede detener el streaming, como se muestra en el siguiente bloque de código:

```
for s in spark.streams.active:
    s.stop()
```

Así es como podemos conectar Azure Databricks a Event hubs y procesar datos en tiempo real.

Puedes obtener más información sobre Azure Databricks y Event hubs en <https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-stream-from-eventhubs>.

A continuación, vamos a ver Spark Structured Streaming.

10.4. Procesamiento de datos con Spark Structured Streaming

El streaming estructurado es una característica de Apache Spark en la que el stream entrante se trata como una tabla no limitada. Los datos de streaming entrantes se añaden continuamente a la tabla. Esta característica facilita la escritura de consultas de streaming, ya que ahora podemos escribir transformaciones de streaming de la misma manera que manejamos las transformaciones basadas en tablas. Por lo tanto, la misma sintaxis de procesamiento por lotes de Spark puede aplicarse aquí también. Spark trata las consultas de streaming estructurado como consultas incrementales sobre una tabla no limitada y las ejecuta a intervalos frecuentes para procesar continuamente los datos.

Spark admite tres modos de escritura para la salida de Structured Streaming:

- **Modo completo**: En este modo, toda la salida (también conocida como tabla de resultados) se escribe en el sink. El sink puede ser un blob store, un data warehouse o una herramienta de BI.
- **Modo de adición**: En este modo, sólo se escriben en el sink las nuevas filas de la última vez.
- **Modo de actualización**: En este modo, sólo se actualizan las filas que han cambiado; las demás filas no se actualizarán.

En la sección anterior, cuando utilizamos Azure Databricks Spark para procesar el stream, ya habíamos utilizado el concepto de Spark Structured Streaming. Cada vez que utilizamos los métodos `writestream` o `readstream`, Spark utiliza Structured Streaming. Veamos otro ejemplo en el que escribimos continuamente los datos de los viajes en streaming y los consultamos como una tabla normal:

```
EHStreamJsonDF.selectExpr(
    "tripId"\
    ,"timestamp"\
    ,"startLocation"\
    ,"endLocation"\
    ,"distance"\
    ,"fare")\
.writeStream.format("delta")\
.outputMode("append")\
.option("checkpointLocation", "dbfs:/TripsCheckpointLocation/")\
.start("dbfs:/TripsEventHubDelta")
```

En la consulta anterior, estamos especificando los nombres de las columnas a extraer como parte de `selectExpr`, el formato a escribir como `delta`, el modo de salida como `append`, la ubicación del punto de control como `dbfs:/TripsCheckpointLocation`, y finalmente, la ubicación del sink a escribir como `dbfs:/TripsEventHubDelta` dentro del método `start()`. Delta es una capa de almacenamiento de código abierto que puede ejecutarse sobre los data lakes. Mejora el data lake para soportar características como transacciones ACID, actualizaciones, borrados, sistemas unificados de lotes, interactivos y de streaming a través de Spark. Aprenderemos más sobre Delta en la sección

Compactación de archivos pequeños del capítulo 14, Optimización y resolución de problemas de almacenamiento y procesamiento de datos.

Ahora puedes consultar los datos como una tabla normal, de la siguiente manera:

```
%sql
CREATE TABLE IF NOT EXISTS TripsAggTumbling
  USING DELTA LOCATION "dbfs:/TripsEventHubDelta/"
SELECT * FROM TripsAggTumbling
```

Así es como podemos utilizar Spark Structured Streaming para manejar datos en streaming. Puedes aprender más sobre Structured Streaming en <https://docs.microsoft.com/en-us/azure/databricks/getting-started/spark/streaming>.

A continuación, vamos a ver cómo monitorizar el rendimiento del streaming.

10.5. Monitorización de rendimiento y regresiones funcionales

Exploremos las opciones de monitorización disponibles en Event hubs, ASA y Spark para escenarios de streaming.

10.5.1. Monitorización en Event Hubs

La pestaña de métricas de Event Hubs proporciona métricas que pueden utilizarse para la monitorización. Aquí hay una captura de pantalla de muestra de las opciones de métrica que están disponibles:

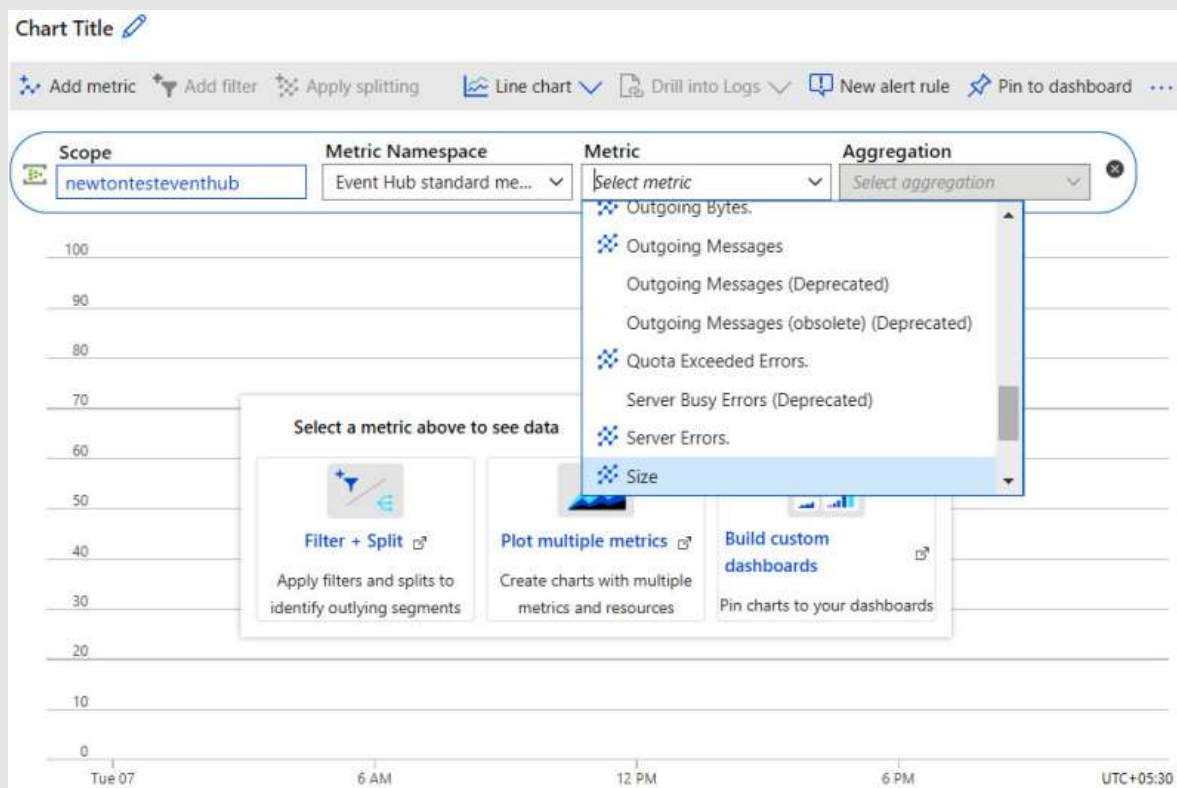


Figura 10.20 - La pantalla de métricas de Event Hubs

Podemos obtener métricas útiles como el número de Mensajes Entrantes, el número de Mensajes Salientes, Errores del Servidor, utilización de la CPU y la memoria, y más. Puedes utilizar todas estas métricas para trazar gráficos y dashboards según sea necesario.

A continuación, veamos las opciones de monitorización en ASA.

10.5.2. Monitorización en ASA

La página de visión general de ASA proporciona métricas de monitorización de alto nivel, como se muestra en la siguiente captura de pantalla:

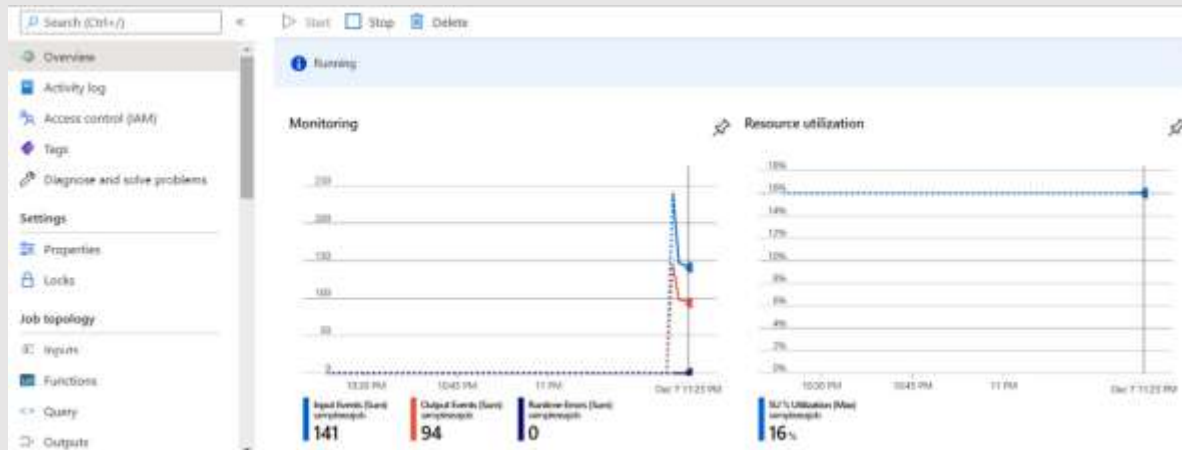


Figura 10.21 - La página de visión general de ASA con métricas

De forma similar a la página de métricas de Event Hubs, ASA también proporciona un rico conjunto de métricas que pueden utilizarse para la monitorización. Esta es una captura de pantalla de muestra de la pestaña de métricas de ASA:

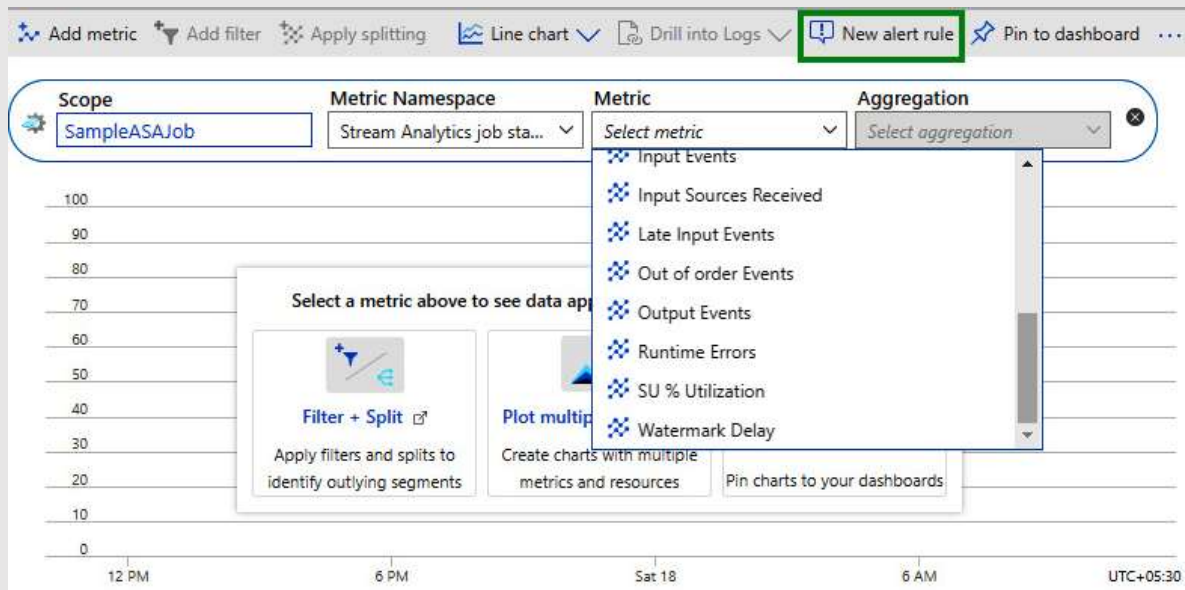


Figura 10.22 - La pestaña de métricas de ASA con métricas

Como puede ver en la captura de pantalla anterior, proporciona información útil como el % de utilización de la SU, los errores de tiempo de ejecución, el retraso de la marca de agua, y más. Incluso podemos configurar alertas sobre estas métricas. Veamos un ejemplo de cómo configurar una alerta para que se dispare si el SU% cruza el 80%.

Configuración de alertas en ASA

Examinemos cómo configurar una alerta para un uso elevado del SU%.

Seleccione el enlace Nueva regla de alerta de la Figura 10.22. Esto le llevará a una página que enumera todas las señales disponibles para construir la alerta. Seleccione la opción SU % Utilización, como se muestra en la siguiente captura de pantalla:

Select a signal [X]

Choose a signal below and configure the logic on the next screen to define the alert condition.

Signal type ⓘ Metrics Monitor service ⓘ All

Displaying 1 - 2 signals out of total 2 search results

Signal name	↑↓	Signal type	↑↓	Monitor service	↑↓
CPU % Utilization (Preview)		📈 Metric		Platform	
SU % Utilization		📈 Metric		Platform	

Done

Figura 10.23 - Selección del nombre de la señal para configurar la alerta

En la página que se abre, puede configurar la lógica de la alerta para establecer el umbral en el 80%. Esto se muestra en la siguiente captura de pantalla:

Alert logic ⓘ Monitoring 1 time series (\$0.1/time series)

Threshold ⓘ

Static Dynamic

Operator ⓘ Aggregation type * ⓘ

Greater than ▼ Maximum ▼

Threshold value * ⓘ

80 ✓

%

Condition preview

Whenever the maximum su % utilization is greater than 80%

Evaluated based on

Aggregation granularity (Period) * ⓘ Frequency of evaluation ⓘ

5 minutes ▼ Every 1 Minute ▼

Figura 10.24 - Configuración de la lógica de alerta de ASA

Una vez que hemos definido la lógica de alerta, tenemos que definir un grupo de acciones (que no se muestra en la captura de pantalla anterior) para determinar qué debe hacerse cuando la condición anterior coincide. Podríamos vincularlo a un libro de ejecución de automatización, activar una función de Azure para realizar una acción, trigger un Webhook o simplemente enviar notificaciones mediante correos electrónicos.

El proceso de creación de alertas es similar al de otras tecnologías como Event Hubs, Synapse SQL, etc.

A continuación, veamos las opciones de monitorización disponibles en Spark Streaming.

10.5.3. Monitorización en Spark Streaming

Los notebooks de Azure Databricks Spark proporcionan gráficos integrados que muestran continuamente métricas como la tasa de entrada, la tasa de procesamiento, la duración del lote y más. Aquí hay una captura de pantalla de ejemplo:



Figura 10.25 - Métricas de Spark Streaming desde un notebook de Spark

Estos gráficos pueden utilizarse para monitorizar el progreso de los jobs de Spark Streaming.

Así es como podemos monitorizar los jobs de streaming. Veamos a continuación cómo procesar datos de series temporales.

10.6. Procesamiento de datos de series temporales

Los datos de series temporales no son más que datos registrados de forma continua a lo largo del tiempo. Algunos ejemplos de datos de series temporales pueden ser las cotizaciones de las acciones registradas a lo largo del tiempo, los valores de los sensores de IoT, que muestran la salud de la maquinaria a lo largo del tiempo, y otros. Los datos de series temporales se utilizan sobre todo para analizar tendencias históricas e identificar cualquier anomalía en los datos, como el fraude con tarjetas de crédito, las alertas en tiempo real y las previsiones. Los datos de las series temporales siempre estarán muy apilados con actualizaciones muy raras.

Los datos de series temporales son un candidato perfecto para el procesamiento en tiempo real. Las soluciones de procesamiento de streaming que hemos discutido anteriormente en este capítulo, en la sección Desarrollo de una solución de procesamiento de streaming utilizando ASA, Azure Databricks y Azure Event Hubs, funcionarían perfectamente para los datos de series temporales. Veamos algunos de los conceptos importantes de los datos de series temporales.

10.6.1. Tipos de timestamps

El aspecto central de cualquier dato de series temporales es el atributo tiempo. Hay dos tipos de tiempo en los datos de las series temporales:

- **Tiempo del evento**: Indica el momento real en que se produjo el evento.
- **Tiempo de procesamiento**: Es el momento en que el evento es procesado por un sistema de procesamiento de datos.

Es importante tener en cuenta el tiempo del evento mientras se procesan los eventos en lugar del tiempo de procesamiento, ya que este último podría retrasarse debido a la velocidad de procesamiento, a los retrasos de la red y a otros problemas en el pipeline del procesamiento del stream.

10.6.2. Agregados con ventanas

Dado que los eventos de las series temporales son eventos no limitados, o en otras palabras, dado que no tienen un tiempo final bien definido, es necesario procesar los eventos en pequeños lotes (es decir, ventanas de tiempo). Existen diferentes mecanismos de ventanas, como las tumbling windows, hopping windows, sliding windows, etc. En la siguiente sección exploraremos en detalle estas técnicas de ventanas.

10.6.3. Checkpointing o watermarking

Checkpointing o watermarking se refiere al proceso de mantener un registro del último evento o timestamp que fue procesado por el sistema de procesamiento de streaming. Esto ayuda a asegurar que comencemos desde el lugar previamente detenido y no perdemos el procesamiento de ningún

evento después de interrupciones, actualizaciones del sistema, retrasos en el procesamiento, y más. Aprenderemos cómo podemos lograr esto en la sección Configuración de checkpoints/watermarking durante el procesamiento de este capítulo.

10.6.4. Reproducir datos de un timestamp anterior

Es posible que tengamos que volver a procesar eventos antiguos en caso de fallos de la máquina o errores en la lógica de procesamiento. En estos casos, herramientas como Event Hubs proporcionan funciones que permiten reproducir los eventos de nuevo desde una ubicación de desplazamiento anterior.

Puede obtener más información sobre los datos de series temporales en <https://docs.microsoft.com/en-us/azure/architecture/data-guide/scenarios/time-series>.

A continuación, vamos a echar un vistazo a cada uno de los conceptos, en detalle, comenzando con las diferentes opciones de agregados por ventanas.

10.7. Diseñar y crear agregados con ventanas

En esta sección, vamos a explorar los diferentes agregados con ventanas que están disponibles en ASA. ASA soporta los siguientes cinco tipos de ventanas:

- Tumbling windows
- Hopping windows
- Sliding windows
- Session windows
- Snapshot windows

Veamos cada una de ellas en detalle. Utilizaremos el siguiente esquema de eventos de muestra en nuestros ejemplos.

```
eventSchema = StructType()  
  .add("tripId", StringType())  
  .add("createdAt", TimestampType())  
  .add("startLocation", StringType())  
  .add("endLocation", StringType())  
  .add("distance", IntegerType())  
  .add("fare", IntegerType())
```

Empecemos con las Tumbling windows.

10.7.1. Tumbling windows

Las Tumbling windows son ventanas de tiempo no superpuestas. Todas las ventanas tienen el mismo tamaño. A continuación se muestra una representación de su aspecto:

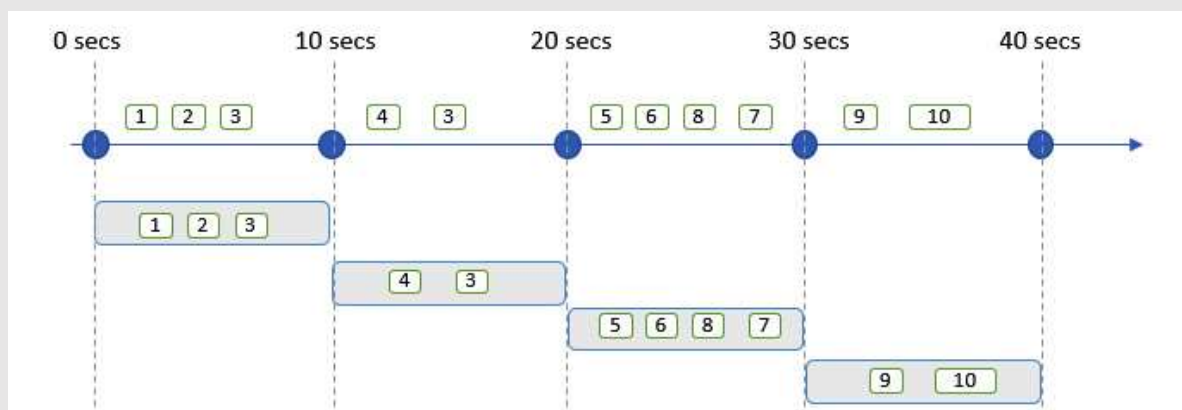


Figura 10.26 - Ejemplo de una ventana de volteo

Esta es la sintaxis de cómo usar una tumbling window:

```
{TUMBLINGWINDOW | TUMBLING} ( timeunit , windowsize, [offsetsize] )  
{TUMBLINGWINDOW | TUMBLING} ( Duration( timeunit , windowsize ), [Offset(timeunit ,  
offsetsize)] )
```

Este es un ejemplo de código para utilizar una tumbling window en ASA. Calcula el número de viajes agrupados por tripLocation, en tumbling windows de 10 segundos de duración:

```
SELECT System.Timestamp() AS WindowEnd, tripLocation, COUNT(*)  
INTO [Output]  
FROM [Input] TIMESTAMP BY createdAt  
GROUP BY tripLocation, TumblingWindow(Duration(second, 10), Offset(millisecond, -1))
```

A continuación, veamos las hopping windows.

10.7.2. Hopping windows

Las Hopping windows son simplemente ventanas de tumbling windows. Cada ventana tendrá un tamaño fijo de superposición con la ventana anterior. A continuación se muestra una representación de su aspecto:

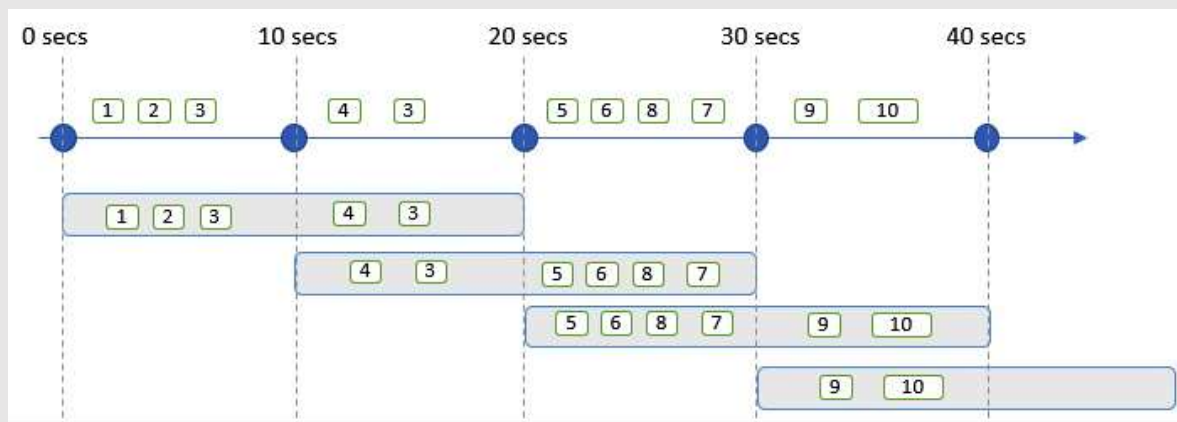


Figura 10.27 - Ejemplo de una ventana de salto

Esta es la sintaxis de una hopping window:

```
{HOPPINGWINDOW | HOPPING} ( timeunit , windowsize , hopsize, [offsetsize] )  
{HOPPINGWINDOW | HOPPING} ( Duration( timeunit , windowsize ), Hop (timeunit , windowsize  
, [Offset(timeunit , offsetsize)])
```

Si el windowsize y hopsize tienen la misma unidad de tiempo, puede utilizar la primera sintaxis.

Este es un ejemplo de hopping window. Cada 10 segundos, obtenga el recuento de viajes por tripLocation de los últimos 20 segundos. Aquí, el tamaño de la ventana es de 20 segundos, y el hop size es de 10 segundos:

```

SELECT System.Timestamp() AS WindowEnd, tripLocation, COUNT(*)
INTO [Output]
FROM [Input] TIMESTAMP BY createdAt
GROUP BY tripLocation, HoppingWindow(Duration(second, 20), Hop(second, 10),
Offset(millisecond, -1))

```

A continuación, veamos las ventanas deslizantes.

10.7.3. Sliding windows

Las Sliding windows tienen un tamaño fijo, pero la ventana sólo se mueve hacia adelante cuando se añade o se quita algún evento. En caso contrario, no emiten ningún resultado:

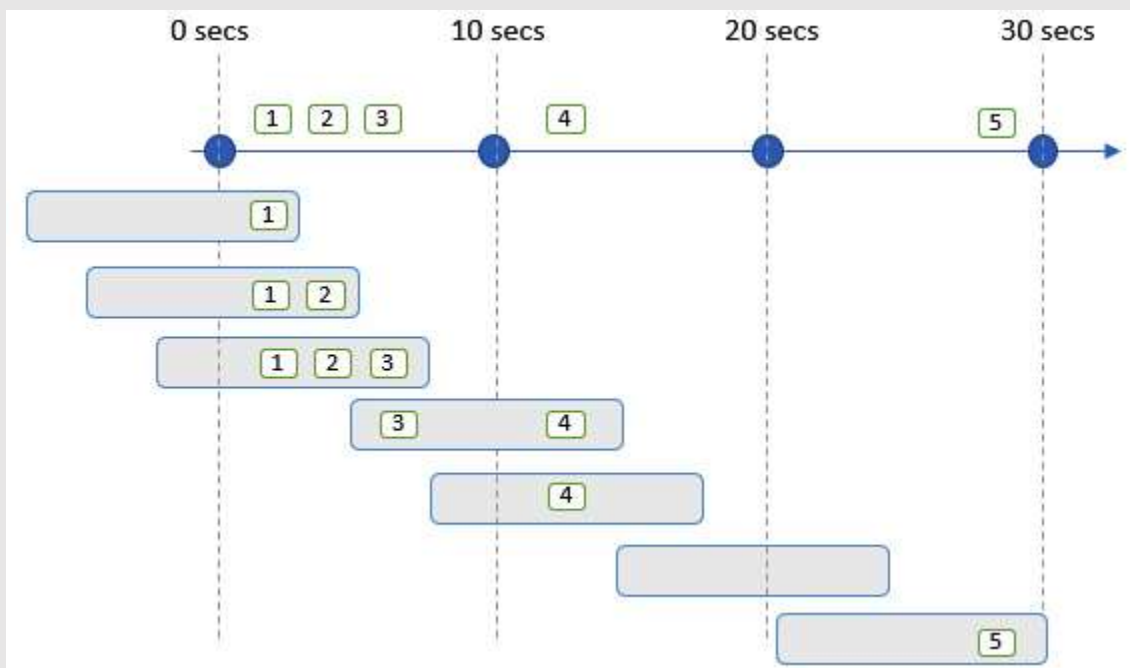


Figura 10.28 - Un ejemplo de ventana deslizante

Esta es la sintaxis de la ventana deslizante:

```

{SLIDINGWINDOW | SLIDING} ( timeunit, windowsize )
{SLIDINGWINDOW | SLIDING} ( Duration( timeunit, windowsize ) )

```

Este es un ejemplo de ventana deslizante. Cada 10 segundos, alerta si una ubicación aparece más de 5 veces:

```

SELECT System.Timestamp() AS WindowEnd, tripLocation, COUNT(*)
INTO [Output]
FROM [Input] TIMESTAMP BY createdAt
GROUP BY tripLocation, SlidingWindow(second, 10)
HAVING COUNT(*) > 5

```

A continuación, veamos las ventanas de sesión.

10.7.4. Session windows

Las Session windows no tienen un tamaño fijo. Tenemos que especificar un tamaño máximo de ventana y un tiempo de espera para las ventanas de sesión. La ventana de sesión trata de captar el mayor número de eventos posible dentro del tamaño máximo de la ventana. Por otro lado, si no hay eventos, espera el tiempo de espera y cierra la ventana. Esta es una representación de su aspecto:

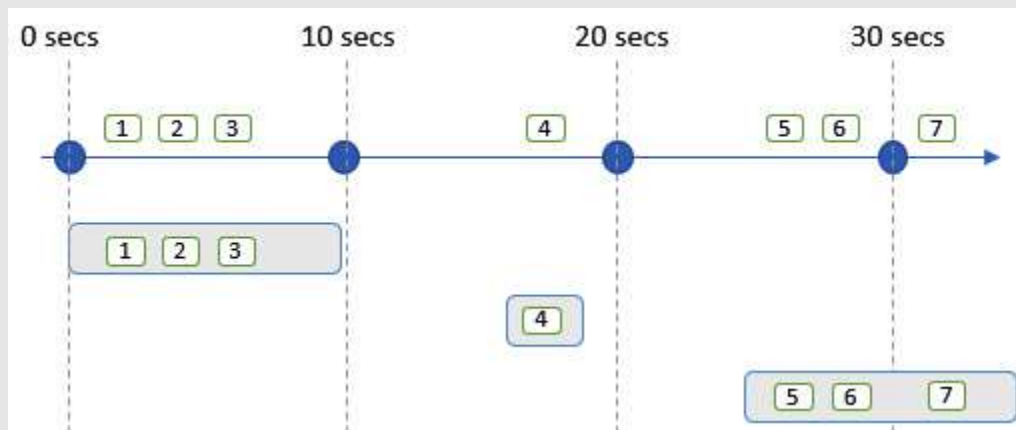


Figura 10.29 - Un ejemplo de ventana de sesión

Esta es la sintaxis de la session window:

```
{SESSIONWINDOW | SESSION} (timeunit, timeoutSize, maxDurationSize) [OVER (PARTITION BY  
partitionKey)]  
{SESSIONWINDOW | SESSION} (Timeout(timeunit , timeoutSize), MaxDuration(timeunit,  
maxDurationSize)) [OVER (PARTITION BY partitionKey)]
```

Este es un ejemplo de una session window. Encuentre el número de viajes que ocurren dentro de los 5 segundos de cada uno:

```
SELECT System.Timestamp() AS WindowEnd, tripId, COUNT(*)  
INTO [Output]  
FROM [Input] TIMESTAMP BY createdAt  
GROUP BY tripId, SessionWindow(second, 5, 10)
```

A continuación, echemos un vistazo a las snapshot windows.

10.7.5. Snapshot windows

Una snapshot window no es realmente una windowing technique. Simplemente se utiliza para obtener una snapshot de los eventos en un momento determinado:

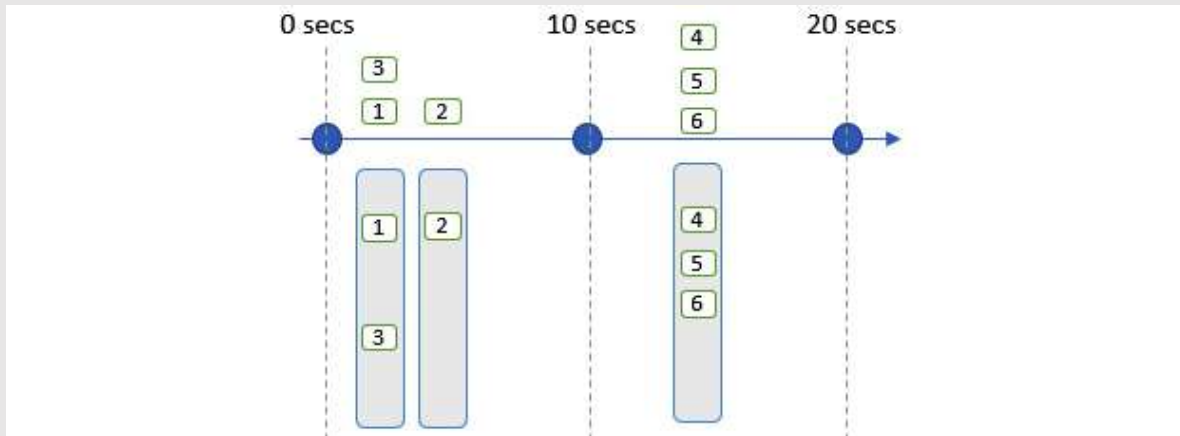


Figura 10.30 - Un ejemplo de snapshot window

A continuación se muestra cómo podemos utilizar una snapshot window:

```
SELECT tripId, COUNT(*)  
INTO [Output]  
FROM [Input] TIMESTAMP BY createdAt  
GROUP BY tripId, System.Timestamp()
```

Estas son todas las opciones de windowing que tenemos en ASA. Puede obtener más información sobre las funciones de windowing de ASA en <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-window-functions>.

El siguiente concepto importante para el manejo de datos de series temporales es el checkpointing o watermarking. Puede obtener más información sobre las funciones de ventana de ASA en <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-window-functions>.

El siguiente concepto importante para el manejo de datos de series temporales es el checkpointing o watermarking. Vamos a profundizar en este tema.

10.8. Configuración de los checkpoints/watermarking durante el procesamiento

Veamos las opciones de checkpointing disponibles en ASA, Event Hubs y Spark.

10.8.1. Checkpointing en ASA

ASA hace un checkpointing interno periódicamente. Los usuarios no necesitan hacer un checkpointing explícito. El proceso de checkpointing se utiliza para recuperar trabajos durante actualizaciones del sistema, reintentos de trabajos, fallos de nodos, etc.

Durante los fallos de nodo o las actualizaciones del sistema operativo, ASA restaura automáticamente el estado del nodo que ha fallado en un nuevo nodo y continúa el procesamiento.

NOTA

Durante las actualizaciones de servicio de ASA (no las actualizaciones del OS), los checkpoints no se mantienen, y es necesario volver a reproducir el flujo correspondiente al tiempo de inactividad.

A continuación, vamos a ver cómo hacer checkpoint en Event Hubs.

10.8.2. Checkpointing en Event Hubs

El checkpointing o watermarking en Event Hubs se refiere al proceso de marcar el offset dentro de un stream o partición para indicar el punto hasta el que se ha completado el procesamiento. El Checkpointing en los Event Hubs es responsabilidad del proceso consumidor de eventos. El Checkpointing es una operación relativamente cara, por lo que normalmente es mejor hacer un checkpoint después de un batch de procesamiento de eventos. La idea principal del checkpointing es tener un punto de reinicio en caso de que el hub de eventos falle o sufra actualizaciones de servicio.

Aquí hay un código de ejemplo de cómo realizar el checkpointing en Event Hubs usando el almacén BlobCheckpoint. Todas las entradas dentro de < > son valores proporcionados por el usuario:

```
Import asyncio
from azure.eventhub.aio import EventHubConsumerClient
from azure.eventhub.extensions.checkpointstoreblobaio import BlobCheckpointStore
async def on_event(partition_context, event):
    # Process the event
    # Checkpoint after processing
    await partition_context.update_checkpoint(event)
async def main():
    storeChkPoint = BlobCheckpointStore.from_connection_string(
        <STORE_CONN_STRING>,
        <STORE_CONTAINER_NAME>
```

```

)
ehClient = EventHubConsumerClient.from_connection_string(
    <EVENTHUB_CONN_STRING>,
    <CONSUMER_GROUP_NAME>,
    eventhub_name=<EVENTHUB_NAME>,
    checkpoint_store= storeChkPoint
)
async with ehClient:
    await ehClient.receive(on_event)
if __name__ == '__main__':
    from azure.eventhub.extensions.checkpointstoreblobaio import BlobCheckpointStore
    async def on_event(partition_context, event):

```

A continuación, vamos a ver el checkpointing en Spark.

10.8.3. Checkpointing en Spark

La función Structured Streaming de Spark ofrece una semántica de extremo a extremo de exactamente una vez. En otras palabras, asegura que un evento se entrega exactamente una vez. Para ello, Spark utiliza el checkpointing y los registros de escritura anticipada. En nuestro ejemplo de Structured Streaming anterior, aprendimos a configurar la ubicación del checkpointing en las consultas de Spark. Aquí tenemos otro ejemplo sencillo:

```

EHStreamJsonDF.writeStream.format("delta")\
    .outputMode("append")\
    .option("checkpointLocation", "dbfs:/TripsCheckpointLocation/")\
    .start("dbfs:/TripsEventHubDelta")

```

Spark Structured Streaming se encarga del checkpointing internamente. Esto es así para que el usuario no tenga que preocuparse de comprobar manualmente el stream de entrada.

Ahora que hemos aprendido cómo hacer un checkpoint de los datos, vamos a ver cómo reproducir los datos antiguos en caso de fallos, reinicios, actualizaciones, etc.

10.9. Reproducir datos de stream archivados

Event Hubs almacena hasta 7 días de datos, que pueden ser reproducidos utilizando las bibliotecas de cliente consumidor de EventHub. He aquí un sencillo ejemplo en Python:

```
Consumer_client = EventHubConsumerClient.from_connection_string(  
    conn_str=CONNECTION_STR,  
    consumer_group='$Default',  
    eventhub_name=EVENTHUB_NAME,  
)  
consumer_client.receive(  
    on_event=on_event,  
    partition_id="0",  
    starting_position="-1" # "-1" is the start of the partition.  
)
```

Puedes especificar offsets o timestamps para el valor de starting_position.

Puedes aprender más sobre las APIs de Python EventHub en [https://azuresdkdocs.blob.core.windows.net/\\$web/python/azure-eventhub/latest/azure.eventhub.html](https://azuresdkdocs.blob.core.windows.net/$web/python/azure-eventhub/latest/azure.eventhub.html).

Echemos un vistazo a algunas de las transformaciones de datos más comunes que son posibles utilizando el análisis de streaming.

10.10. Transformaciones usando streaming analytics

Uno de los temas comunes que puede notar en las consultas de streaming es que si hay algún tipo de transformación involucrada, siempre habrá una windowed aggregation que debe ser especificada. Tomemos el ejemplo de contar el número de entradas distintas en un marco temporal.

10.10.1. Las transformaciones COUNT y DISTINCT

Este tipo de transformación se puede utilizar para contar el número de eventos distintos que han ocurrido en una ventana de tiempo. He aquí un ejemplo para contar el número de trips únicos en los últimos 10 segundos:

```
SELECT
  COUNT(DISTINCT tripId) AS TripCount,
  System.TIMESTAMP() AS Time
INTO [Output]
FROM [Input] TIMESTAMP BY createdAt
GROUP BY TumblingWindow(second, 10)
```

A continuación, vamos a ver un ejemplo en el que podemos castear el tipo de entrada en un formato diferente.

10.10.2. Transformaciones CAST

La transformación CAST puede utilizarse para convertir el tipo de datos sobre la marcha. Este es un ejemplo para convertir la tarifa en un FLOAT y calcular la tarifa total cada 10 minutos:

```
SELECT tripId, SUM(CAST(fare AS FLOAT)) AS TenSecondFares
INTO [Output]
FROM [Input] TIMESTAMP BY createdAt
GROUP BY tripId, TumblingWindow(second, 10)
```

A continuación, vamos a ver cómo comparar entradas utilizando LIKE.

10.10.3. Transformaciones LIKE

Podemos utilizar esta transformación para realizar coincidencias de cadenas utilizando patrones. En el siguiente ejemplo, intentamos hacer coincidir todos los startLocation de San Francisco:

```
SELECT *
INTO [Output]
FROM [Input] TIMESTAMP BY timestamp
WHERE startLocation LIKE 'S%F'
```

Estos son sólo algunos ejemplos de transformaciones. Puede encontrar una lista más detallada en <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-stream-analytics-query-patterns>.

A continuación, vamos a ver cómo manejar schema drifts en una solución de streaming.

10.11. Manejo de los schema drifts

Un schema drift se refiere a los cambios en el esquema a lo largo del tiempo debido a los cambios que se producen en las fuentes de eventos. Esto puede deberse a que las columnas o los campos más nuevos envejecen, las columnas se eliminan, etc.

10.11.1. Manejando schema drifts usando Event Hubs

Si un publicador de eventos necesita compartir los detalles del esquema con el consumidor, tiene que serializar el esquema junto con los datos, utilizando formatos como Apache Avro, y enviarlo a través de Event Hubs. Aquí, el esquema tiene que ser enviado con cada evento, lo que no es un enfoque muy eficiente.

Si se trata de esquemas definidos estáticamente en el lado del consumidor, cualquier cambio de esquema en el lado del productor supondría un problema.

Event Hubs ofrece una función denominada Azure Schema Registry para gestionar la evolución y el cambio de esquemas. Proporciona un repositorio central para compartir los esquemas entre los editores y consumidores de eventos. Examinemos cómo crear y utilizar Azure Schema Registry.

Registro de un esquema con schema registry

A continuación se muestra un ejemplo de cómo registrar un esquema:

1. Importar las librerías necesarias:

```
from azure.schemaregistry import SchemaRegistryClient
from azure.identity import DefaultAzureCredential
```

2. Defina su esquema:

```
sampleSchema = """
{"namespace": "com.azure.sampleschema.avro",
"type": "record",
"name": "Trip",
"fields": [
  {"name": "tripId", "type": "string"},
  {"name": "startLocation", "type": "string"},
  {"name": "endLocation", "type": "string"}
]
}"""
```

3. Cree el Schema Registry client:

```
azureCredential = DefaultAzureCredential()  
schema_registry_client = SchemaRegistryClient(  
    fully_qualified_namespace=<SCHEMA-NAMESPACE>.servicebus.windows.net,  
    credential=azureCredential)
```

4. Registrar el esquema:

```
With schema_registry_client:  
    schema_properties = schema_registry_client.register_schema(  
        <SCHEMA_GROUPNAME>,  
        <SCHEMA_NAME>,  
        sampleSchema,  
        "Avro")
```

5. Obtenga el ID del esquema:

```
schema_id = schema_properties.id
```

A continuación, vamos a ver cómo recuperar el esquema.

Recuperación de un esquema del Registro de Esquemas

Estos son los pasos de alto nivel para recuperar el esquema:

1. Importar las librerías necesarias:

```
from azure.identity import DefaultAzureCredential  
from azure.schemaregistry import SchemaRegistryClient
```

2. Crear el cliente de Schema Registry:

```
azureCredential = DefaultAzureCredential()  
schema_registry_client = SchemaRegistryClient(  
    fully_qualified_namespace=<SCHEMA-NAMESPACE>.servicebus.windows.net,  
    credential=azureCredential)
```

3. Recupera el esquema:

```
With schema_registry_client:  
    schema = schema_registry_client.get_schema(schema_id)  
    definition = schema.definition  
    properties = schema.properties
```

Una vez que tengas el esquema, puedes definir tus eventos y empezar a leer los datos en el formato correcto.

Puedes aprender más sobre el Registro de Esquemas de Event Hubs en <https://docs.microsoft.com/en-us/azure/event-hubs/schema-registry-overview>.

A continuación, vamos a ver cómo manejar los schema drifts en Spark.

10.11.2. Manejo de schema drifts en Spark

Azure Databricks Delta Lake proporciona una función llamada Schema Evolution para ocuparse de los cambios de esquema a lo largo del tiempo. Se adapta automáticamente al nuevo esquema cuando se añaden nuevas columnas. La Evolución de Esquema se puede habilitar añadiendo la opción `.option('mergeSchema', 'true')` al comando de streaming `writeStream`.

He aquí un ejemplo sencillo:

```
StreamDF.writeStream.format("delta")\
.option("mergeSchema", "true") \
.outputMode("append")\
.option("checkpointLocation", "dbfs:/CheckpointLocation/")\
.start("dbfs:/StreamData")
```

Una vez especificada la opción `mergeSchema`, Spark se encarga de manejar las nuevas columnas en el stream automáticamente.

A continuación, vamos a ver la partición de eventos en Event Hubs.

10.12. Procesamiento entre particiones

Antes de ver cómo procesar los datos a través de las particiones, primero, vamos a entender las particiones.

10.12.1. ¿Qué son las particiones?

Los Event Hubs pueden distribuir los eventos entrantes en múltiples streams para que puedan ser accedidos, en paralelo, por los consumidores. Estos streams paralelos se llaman particiones. Cada partición almacena los datos reales del evento y los metadatos del evento, como su offset en la partición, su timestamp del lado del servidor cuando el evento fue aceptado, su número en la secuencia del stream, y más. El particionamiento ayuda a escalar el procesamiento en tiempo real, ya que aumenta el paralelismo al proporcionar múltiples stream de entrada para los motores de procesamiento posteriores. Además, mejora la disponibilidad al redirigir los eventos a otras particiones sanas si algunas de las particiones fallan.

Puedes aprender más sobre las particiones de Event Hubs en <https://docs.microsoft.com/en-in/azure/event-hubs/event-hubs-scalability>.

Ahora, veamos cómo enviar datos a través de las particiones y cómo procesar los datos a través de las particiones.

10.12.2. Procesando datos a través de las particiones

Las librerías cliente para acceder a los Event Hubs están disponibles en diferentes lenguajes como C#, Python, Java y otros. En esta sección, utilizaremos la biblioteca de Python para nuestros ejemplos.

Event Hubs proporciona la clase `EventHubConsumerClient` como parte de las librerías cliente de Python para consumir los datos del event hub. `EventHubConsumerClient` se puede utilizar para leer eventos de todas las particiones con balanceo de carga y checkpointing. Ya hemos utilizado esta clase en el ejemplo de checkpointing en la sección Checkpointing en Event Hubs. El mismo ejemplo funcionará para leer a través de las particiones. Aquí, proporcionaremos los pasos importantes de nuevo por conveniencia.

Tenga en cuenta que todas las entradas dentro de los corchetes angulares, `< >`, son valores proporcionados por el usuario:

1. Instanciar el checkpoint store:

```
storeChkPoint = BlobCheckpointStore.from_connection_string(  
    <STORE_CONN_STRING>,  
    <STORE_CONTAINER_NAME>  
)
```

2. Instanciar la clase EventHubConsumerClient:

```
ehClient = EventHubConsumerClient.from_connection_string(  
    <EVENTHUB_CONN_STRING>,  
    <CONSUMER_GROUP_NAME>,  
    eventhub_name=<EVENTHUB_NAME>,  
    checkpoint_store= storeChkPoint # Esto permite el equilibrio de carga entre particiones  
)
```

3. Definir un método on_event para procesar el evento cuando llegue:

```
Def on_event(partition_context, event):  
    # Process the event  
    partition_context.update_checkpoint(event)
```

4. Llama al método on_event cuando llega un evento:

```
With ehClient:  
    ehClient.receive(  
    on_event=on_event,  
        starting_position="-1", # Para empezar desde el principio de la partición.  
    )
```

Cuando no especificamos una partición concreta al instanciar la clase EventHubConsumerClient, ésta leerá automáticamente de todas las particiones del grupo de consumidores especificado. Puedes saber más sobre las librerías Python de Event Hubs en [https://azuresdkdocs.blob.core.windows.net/\\$web/python/azure-eventhub/latest/azure.eventhub.html](https://azuresdkdocs.blob.core.windows.net/$web/python/azure-eventhub/latest/azure.eventhub.html).

A continuación, vamos a ver los detalles de cómo procesar los datos dentro de una partición.

10.13. Procesamiento dentro de una partición

Al igual que en el ejemplo anterior, en el que aprendimos a procesar a través de las particiones, podemos utilizar la clase `EventHubConsumerClient` para procesar los datos dentro de las particiones individuales, también. Todo lo que tenemos que hacer es especificar el ID de la partición en la llamada `client.receive`, como se demuestra en el siguiente fragmento de código. El resto del código seguirá siendo el mismo que en el ejemplo anterior:

```
With client:
    client.receive(
        on_event=on_event,
        partition_id='0', # Para leer sólo la partición 0
    )
```

Así es como podemos procesar programáticamente los datos de particiones específicas de Event Hubs.

A continuación, vamos a ver cómo escalar los recursos para el procesamiento de stream.

10.14. Escalando recursos

Veamos cómo escalar recursos en Event Hubs, ASA y Azure Databricks Spark.

10.14.1. Escalado en Event Hubs

Hay dos formas en las que Event Hubs soporta el escalado:

- **Partición:** Ya hemos aprendido cómo el particionamiento puede ayudar a escalar nuestra instancia de Event Hubs aumentando el paralelismo con el que los consumidores de eventos pueden procesar los datos. El particionamiento ayuda a reducir la contención si hay demasiados productores y consumidores, lo que, a su vez, lo hace más eficiente.
- **Autoinflado:** Se trata de una función de escalado automático de los Event Hubs. A medida que aumenta el uso, EventHub añade más unidades de rendimiento a su instancia de Event Hubs, aumentando así su capacidad. Puedes habilitar esta función si ya has saturado tu cuota utilizando la técnica de partición que hemos explorado antes, en la sección Procesamiento a través de particiones.

A continuación, vamos a explorar el concepto de unidades de rendimiento.

¿Qué son las unidades de rendimiento (throughput units)?

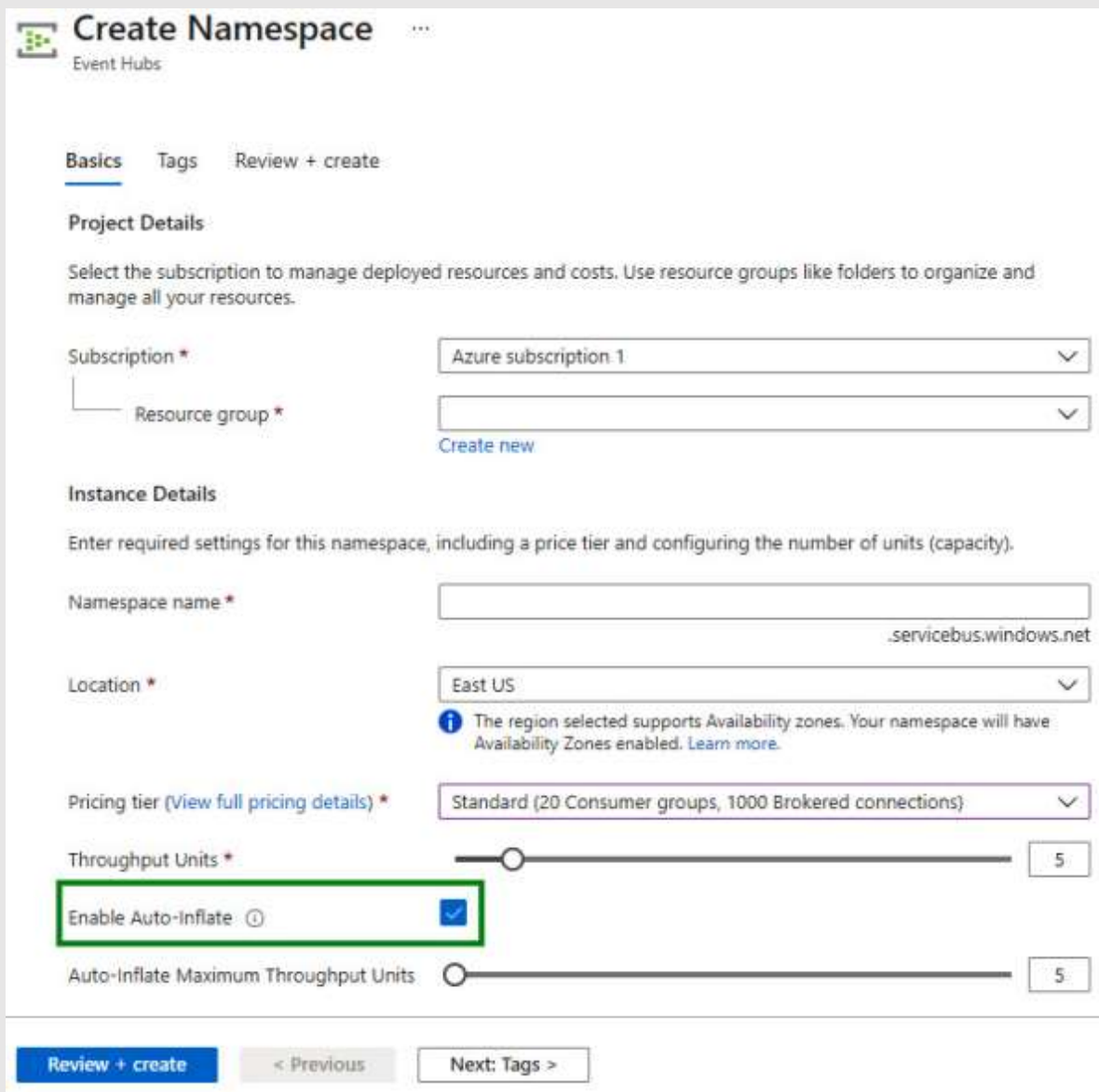
Las unidades de rendimiento (throughput units) son unidades de capacidad que se pueden comprar en Event Hubs. Una sola unidad de rendimiento permite lo siguiente

- Entrada de hasta 1 MB por segundo o 1.000 eventos por segundo
- Salida de hasta 2 MB por segundo o 4.096 eventos por segundo

NOTA

En el nivel Premium de Event Hubs, las throughput units se denominan unidades de procesamiento

Así es como puedes activar la función de autoinflado en Event Hubs:



Create Namespace ...
Event Hubs

Basics Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure subscription 1

Resource group *
Create new

Instance Details

Enter required settings for this namespace, including a price tier and configuring the number of units (capacity).

Namespace name *
 .servicebus.windows.net

Location * East US
 The region selected supports Availability zones. Your namespace will have Availability Zones enabled. [Learn more.](#)

Pricing tier (View full pricing details) * Standard (20 Consumer groups, 1000 Brokered connections)

Throughput Units * 5

Enable Auto-Inflate ⓘ ☒

Auto-Inflate Maximum Throughput Units 5

[Review + create](#) < Previous Next: Tags >

Figura 10.31 - Habilitar el auto-inflado en Event Hubs

Puedes aprender más sobre la escalabilidad de Event Hubs en <https://docs.microsoft.com/en-in/azure/event-hubs/event-hubs-scalability>.

A continuación, vamos a ver cómo escalar el motor de procesamiento, ASA.

10.14.2. Escalando en ASA

Puedes escalar tus clusters de ASA directamente desde el portal de Azure. Sólo tienes que hacer clic en la pestaña Escala de la página de inicio de ASA, y configurar el conmutador de unidades de Streaming, como se muestra en la siguiente captura de pantalla:

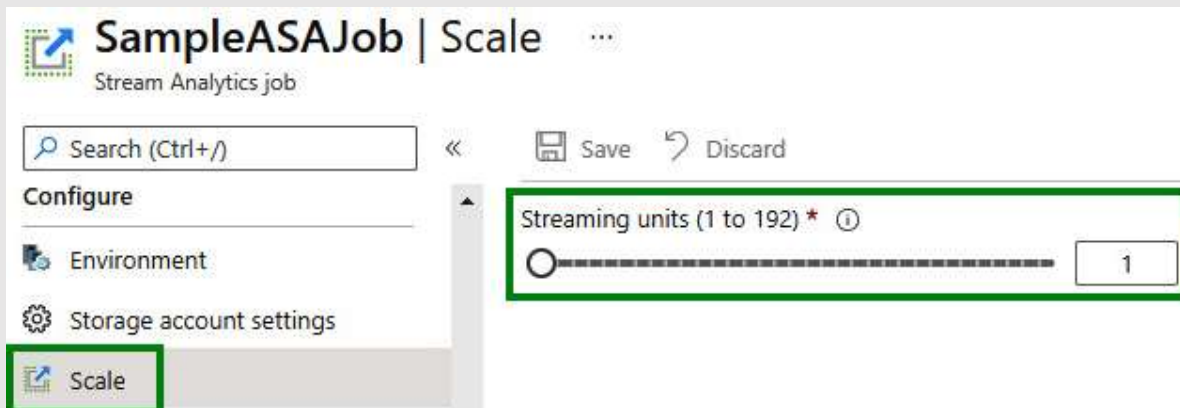


Figura 10.32 - Escalado de un job ASA

Las unidades de streaming son una medida de la capacidad de streaming. Las unidades de streaming no tienen una configuración de capacidad absoluta como en el caso de las unidades de rendimiento de los Event Hubs. En general, cuanto mayor sea el número de unidades de streaming, mayor será la capacidad. Tendrás que hacer algunas pruebas de ensayo y error para identificar el punto óptimo.

TIP

Para las consultas ASA que no utilizan la transformación PARTITION BY, la recomendación es comenzar con seis unidades de stream y luego modificar los números de manera iterativa ejecutando sus trabajos y monitoreando la métrica de utilización del SU%.

Puede aprender sobre la optimización de las unidades de streaming en <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-streaming-unit-consumption>.

Puede obtener más información sobre el escalado de ASA en <https://docs.microsoft.com/en-us/azure/stream-analytics/scale-cluster>.

ASA no tiene un mecanismo interno de autoescalado. Sin embargo, puede simular uno utilizando servicios como Azure Automation. Se trata de un servicio externo que puede utilizarse para supervisar las métricas de ASA y desencadenar un aumento o una reducción de escala de forma externa.

Si está interesado en saber más, puede consultar <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-autoscale>.

A continuación, vamos a ver cómo escalar Azure Databricks Spark.

10.14.3. Escalado en Azure Databricks Spark Streaming

Aprendimos a configurar el autoescalado para Azure Databricks en el Capítulo 9, Diseño y desarrollo de una solución de procesamiento por lotes, en la sección Recursos de escalado. Consulta esa sección para refrescar tu memoria sobre cómo configurar el autoescalado para Spark.

A continuación, veamos cómo manejar las interrupciones.

10.15. Manejo de las interrupciones

Las interrupciones en el procesamiento del stream pueden ocurrir debido a varias razones, como problemas de conectividad de la red, actualizaciones de servicios en segundo plano, errores intermitentes, etc. Los concentradores de eventos y ASA ofrecen opciones para gestionar dichas interrupciones de forma nativa utilizando el concepto de zonas de disponibilidad de Azure. Las zonas de disponibilidad son ubicaciones físicamente aisladas en Azure que ayudan a las aplicaciones a ser resistentes a los fallos e interrupciones locales. Azure enumera las regiones que se emparejan para formar zonas de disponibilidad.

Los servicios que admiten zonas de disponibilidad despliegan sus aplicaciones en todas las ubicaciones dentro de la zona de disponibilidad para mejorar la tolerancia a fallos. Además, se aseguran de que las actualizaciones del servicio se realicen siempre una tras otra para las ubicaciones de la zona de disponibilidad. Por lo tanto, garantizan que en ningún momento todas las ubicaciones sufrirán una interrupción debido a errores de actualización del servicio. Tanto Event Hubs como ASA son compatibles con las zonas de disponibilidad. Veamos cómo habilitar esta función tanto para Event Hubs como para ASA.

10.15.1. Manejo de interrupciones en Event Hubs

Cuando Event Hubs se despliega en regiones que forman parte de las zonas de disponibilidad, tanto los metadatos como los eventos se replican en todas las ubicaciones de esa zona de disponibilidad. Para utilizar las zonas de disponibilidad, todo lo que hay que hacer es seleccionar una región que soporte la zona de disponibilidad para el campo Ubicación, como se muestra en la siguiente captura de pantalla:

Create Namespace ...

Event Hubs

Basics Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure subscription 1

Resource group * [Create new](#)

Instance Details

Enter required settings for this namespace, including a price tier and configuring the number of units (capacity).

Namespace name * .servicebus.windows.net

Location * East US

i The region selected supports /availability zones. Your namespace will have Availability Zones enabled. [Learn more.](#)

Pricing tier (View full pricing details) *

Throughput Units * 1

[Review + create](#) [< Previous](#) [Next: Tags >](#)

Figura 10.33 - Elección de ubicaciones con zonas de disponibilidad en Event Hubs

Una vez que hayas elegido la región que soporta la zona de disponibilidad y hayas hecho clic en Revisar + crear, la instancia de Event Hubs se crea en todas las ubicaciones de la zona de disponibilidad.

Aquí hay algunos otros puntos para hacer que la instancia de Event Hubs sea más resistente:

Intenta construir una lógica de retroceso y reintento en tus aplicaciones. Esto es para que los errores transitorios puedan ser capturados, y el procesamiento del stream pueda ser reintentado. Si tu aplicación está construida usando los SDKs de Event Hubs, entonces la lógica de reintento ya está incorporada.

Si la aplicación no necesita mensajes estrictamente ordenados, puedes enviar los eventos sin especificar una partición. Esto asegurará que Event Hubs equilibre la carga de los eventos en todas las particiones. Si una partición falla, la instancia de Event Hubs distribuirá el evento a otras particiones, mejorando así la disponibilidad.

Puedes aprender más sobre las zonas de disponibilidad de Event Hubs en <https://docs.microsoft.com/en-in/azure/event-hubs/event-hubs-geo-dr>.

A continuación, vamos a ver cómo manejar las interrupciones en ASA.

10.15.2. Manejo de interrupciones en ASA

Al igual que los Event Hubs, ASA también se despliega en zonas de disponibilidad (o regiones emparejadas con Azure) por defecto. Además, ASA garantiza que las actualizaciones del servicio siempre se produzcan en lotes separados entre las ubicaciones de las zonas de disponibilidad. No se requiere ninguna configuración por parte de los usuarios.

NOTA

En el momento de redactar este documento, la región de India Central no tiene una región emparejada para ASA.

Puede obtener más información sobre las zonas de disponibilidad y la gestión de interrupciones en ASA en <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-job-reliability>.

A continuación, vamos a ver cómo diseñar y configurar la gestión de excepciones.

10.16. Diseñando y configurando el manejo de excepciones

Las excepciones de Event Hubs proporcionan información muy clara sobre el motivo de los errores. Todos los problemas de EventHub lanzan un objeto de excepción EventHubsException.

El objeto de excepción EventHubsException contiene la siguiente información

- **IsTransient**: Indica si las excepciones se pueden reintentar.
- **Reason**: Indica el motivo de la excepción: Indica el motivo real de la excepción. Algunos ejemplos de razones pueden ser los tiempos de espera, la superación de los límites de cuota, la superación del tamaño de los mensajes, la desconexión de la conexión del cliente, etc.

Este es un ejemplo sencillo de cómo capturar excepciones en .NET:

```
try
{
    // Process Events
}
catch (EventHubsException ex) where
(ex.Reason == EventHubsException.FailureReason.MessageSizeExceeded)
{
    // Take action for the oversize messages
}
```

Puedes aprender más sobre el manejo de excepciones en <https://docs.microsoft.com/en-us/azure/event-hubs/exceptions-dotnet>.

A continuación, vamos a ver la inserción de datos utilizando Synapse Analytics.

10.17. Upserting data

El upserting se refiere a la actividad INSERT o UPDATE en una base de datos o en cualquier almacén de datos analíticos que lo soporte. Ya hemos visto el UPSERT como parte de la actividad por lotes, en la sección Upserting data del capítulo 9, Diseño y desarrollo de una solución de procesamiento por lotes. ASA soporta UPSERT con CosmosDB. CosmosDB es una base de datos No-SQL totalmente gestionada y distribuida globalmente. Aprenderemos más sobre CosmosDB en el Capítulo 14, Optimización y resolución de problemas de almacenamiento y procesamiento de datos, en la sección Implementación de HTAP con Synapse Link y CosmosDB.

ASA tiene dos comportamientos diferentes basados en el nivel de compatibilidad que se establezca. ASA admite tres niveles de compatibilidad diferentes. Puedes pensar en los niveles de compatibilidad como versiones de la API. A medida que ASA fue evolucionando, los niveles de compatibilidad aumentaron. La versión 1.0 fue la primera versión de compatibilidad, y la 1.2 es la última versión de compatibilidad. El principal cambio en la versión 1.2 es la compatibilidad con el protocolo de mensajería AMQP.

Puede establecer el nivel de compatibilidad, como se muestra en la siguiente captura de pantalla:

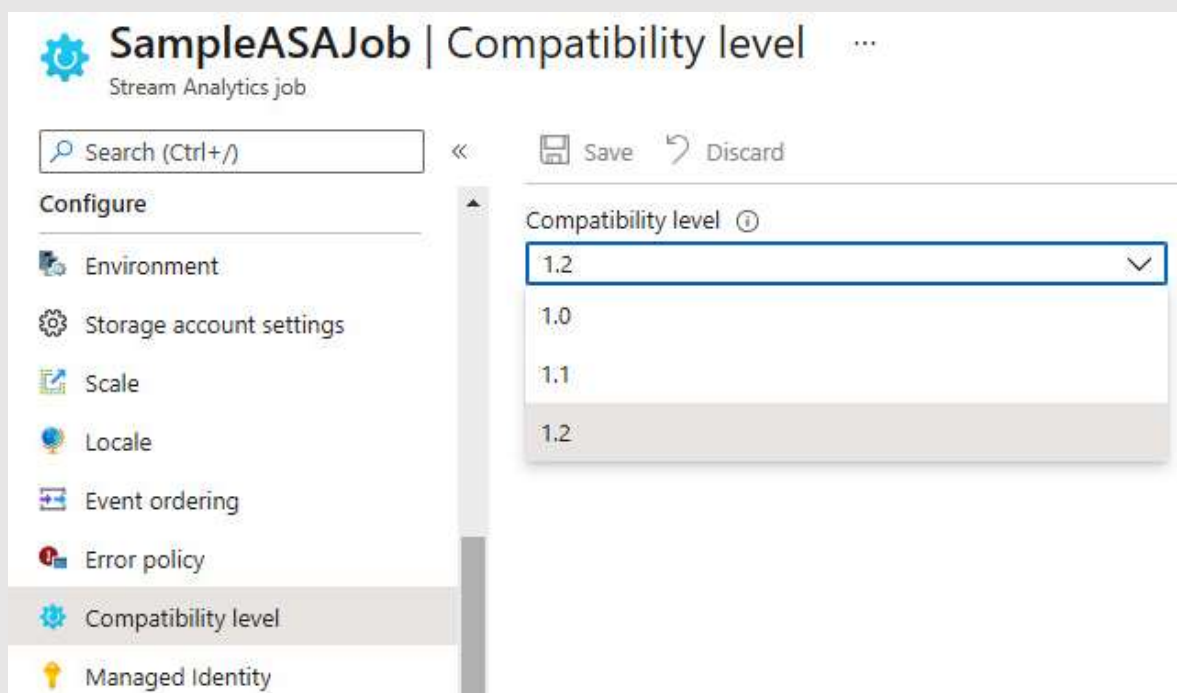


Figura 10.34 - Actualización del nivel de compatibilidad en ASA

Con los niveles de compatibilidad 1.0 y 1.1, ASA realiza una inserción o actualización a nivel de propiedad dentro del documento. Permite realizar actualizaciones parciales en el documento como una operación PATCH.

Con un nivel de compatibilidad de 1.2 en adelante, ASA realiza una operación de inserción o sustitución del documento. En primer lugar, ASA realiza una inserción. Si esto falla debido a un conflicto de ID de documento, entonces realiza una actualización.

NOTA:

Las upserts funcionan en CosmosDB cuando el ID del documento está configurado. Si no se ha establecido, los escenarios de actualización arrojarán un error.

Puedes obtener más información sobre ASA - CosmosDB Upserts en <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-documentdb-output#upserts-from-stream-analytics>.

A continuación, vamos a ver cómo crear pruebas para las canalizaciones de datos.

10.18. Diseño y creación de pruebas para data pipelines

Esta sección ya se ha tratado en el Capítulo 9, Diseño y desarrollo de una solución de procesamiento por lotes, en la sección Diseño y creación de pruebas para canalizaciones de datos. Por favor, consulte esa sección para más detalles.

10.19. Optimización de pipelines para fines analíticos o transaccionales

Cubriremos este tema en el Capítulo 14, Optimización y resolución de problemas de almacenamiento y procesamiento de datos, en la sección Optimización de canalizaciones con fines analíticos o transaccionales, ya que todo ese capítulo trata de las optimizaciones.

Resumen

Así llegamos al final de este capítulo. Este es uno de los capítulos más importantes tanto desde la perspectiva del plan de estudios como de la ingeniería de datos. Las soluciones batch y streaming son fundamentales para construir un buen sistema de procesamiento de big data.

Así que vamos a recapitular lo que hemos aprendido en este capítulo. Comenzamos con diseños para sistemas de streaming utilizando Event Hubs, ASA y Spark Streaming. Hemos aprendido a monitorizar dichos sistemas utilizando las opciones de monitorización disponibles en cada uno de esos servicios. A continuación, aprendimos sobre los datos de series temporales y conceptos importantes como los agregados en ventanas, el checkpointing, la reproducción de datos archivados, el manejo de las derivas del esquema, cómo escalar utilizando particiones y la adición de unidades de procesamiento. Además, exploramos la función upsert, y hacia el final, aprendimos sobre el manejo de errores y el manejo de interrupciones.

Ahora debería sentirse cómodo con la creación de soluciones de streaming en Azure. Como siempre, por favor revisa los enlaces de seguimiento que se han proporcionado para aprender más, y prueba los ejemplos por ti mismo para entender los detalles de estas tecnologías.

En el próximo capítulo, aprenderemos a gestionar batches y pipelines.