

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 9: Diseño y desarrollo de una solución de Batch Processing Solution (procesamiento por lotes)	4
9.1. Requisitos técnicos	5
9.2. Diseño de una solución de batch processing	5
9.3. Desarrollo de soluciones de procesamiento por lotes utilizando Data Factory, Data Lake, Spark, Azure Synapse Pipelines, PolyBase y Azure Databricks	7
9.3.1. Storage	8
9.3.2. Ingesta de datos	8
9.3.3. Preparación de datos/limpieza de datos	12
9.3.4. Transformación	12
9.3.5. Configuración de una actividad ADB notebook en ADF	18
9.3.6. Opciones de tecnología de procesamiento por lotes	20
9.3.7. Utilización de PolyBase para introducir los datos en el almacén de datos de Analytics	20
Opciones para cargar con PolyBase	22
9.3.8. Uso de Power BI para visualizar los datos	23
9.4. Creación de data pipelines	28
9.5. Integración de notebooks Jupyter/Python en un data pipelines	29
9.6. Diseño e implementación de cargas de datos incrementales	32
9.7. Diseñar y desarrollar dimensiones de cambio lento	32
9.8. Manejo de datos duplicados	32
9.9. Manejo de datos faltantes	32
9.10. Manejo de datos que llegan tarde	33
9.10.1. Tratamiento de los datos que llegan tarde en la fase de ingesta/transformación	33
9.10.2. Manejo de los datos que llegan tarde en la fase de servicio	33
9.11. Inserción de datos	35
9.12. Regresar a un estado anterior	36
9.13. Presentación de Azure Batch	38
9.13.1. Ejecución de un Azure Batch job de ejemplo	39
9.14. Configurar el batch size	41
9.15. Escalado de recursos	43
9.15.1. Azure Batch	43
9.15.2. Azure Databricks	44
9.15.3. Synapse Spark	45

9.15.4. Synapse SQL	46
9.16. Configuración de la batch retention	48
9.17. Diseño y configuración de la gestión de excepciones.....	49
9.17.1. Tipos de errores	49
9.17.2. Acciones correctivas.....	50
9.18. Gestión de los requisitos de seguridad y cumplimiento de la normativa.....	51
9.18.1. El Azure Security Benchmark	53
9.18.2. Mejores prácticas para Azure Batch	53
Resumen.....	55

Capítulo 9: Diseño y desarrollo de una solución de Batch Processing Solution (procesamiento por lotes)

Bienvenido al siguiente capítulo de la serie de transformación de datos. Si ha llegado hasta aquí, es que realmente se toma en serio la certificación. ¡Buen trabajo! Ya ha cruzado la mitad del camino, con sólo unos pocos capítulos más por delante.

En el capítulo anterior, aprendimos sobre un montón de tecnologías, como Spark, Azure Data Factory (ADF) y Synapse SQL. Aquí continuaremos la racha y aprenderemos sobre algunas tecnologías más relacionadas con el procesamiento por lotes. Aprenderemos cómo construir pipelines de lotes de extremo a extremo, cómo usar Spark Notebooks en data pipelines, cómo usar tecnologías como PolyBase para acelerar la copia de datos, y más. También aprenderemos técnicas para manejar datos que llegan tarde, escalar clusters, depurar problemas de pipelines, y manejar la seguridad y el cumplimiento de los pipelines. Después de completar este capítulo, debería ser capaz de diseñar e implementar batch pipelines de extremo a extremo basados en ADF utilizando tecnologías como Synapse SQL, Azure Databricks Spark, PolyBase y Azure Batch en los pipelines de ADF.

Este capítulo va a ser un poco más largo, ya que tenemos que cubrir muchos conceptos importantes. Me he tomado la libertad de reorganizar los temas del programa para este capítulo agrupando los que están relacionados. Esto ayudará a crear un mejor flujo para el capítulo. Vamos a cubrir los siguientes temas:

- Diseño de una solución de procesamiento por lotes
- Desarrollo de soluciones de procesamiento por lotes utilizando Data Factory, Data Lake, Spark, Azure Synapse Pipelines, PolyBase y Azure Databricks
- Creación de data pipelines
- Integración de notebooks Jupyter/Python en un data pipeline
- Diseño e implementación de cargas de datos incrementales
- Diseño y desarrollo de slowly changing dimensions
- Manejo de datos duplicados
- Manejo de datos faltantes
- Manejo de datos que llegan tarde
- Reintroducción de datos
- Regreso a un estado anterior
- Presentación de Azure Batch
- Configuración del tamaño del lote
- Escalado de recursos
- Configuración de la retención de lotes
- Diseño y configuración del manejo de excepciones
- Diseño y creación de pruebas para pipelines de datos
- Depuración de trabajos de Spark utilizando la interfaz de usuario de Spark
- Manejo de los requisitos de seguridad y cumplimiento

9.1. Requisitos técnicos

Para este capítulo, necesitará lo siguiente

Una cuenta de Azure (gratuita o de pago)
Un área de trabajo de Synapse activa
Un área de trabajo Azure Data Factory activa

¡Empecemos!

9.2. Diseño de una solución de batch processing

En el Capítulo 2, Diseño de una estructura de almacenamiento de datos, aprendimos sobre la arquitectura del data lake. He presentado el diagrama aquí de nuevo por conveniencia. En el siguiente diagrama, hay dos ramas, una para el procesamiento por lotes y otra para el procesamiento en tiempo real. La parte resaltada en verde es la solución de procesamiento por lotes para un data lake. El procesamiento por lotes suele tratar grandes cantidades de datos y tarda más tiempo en procesarse en comparación con el procesamiento en flujo.

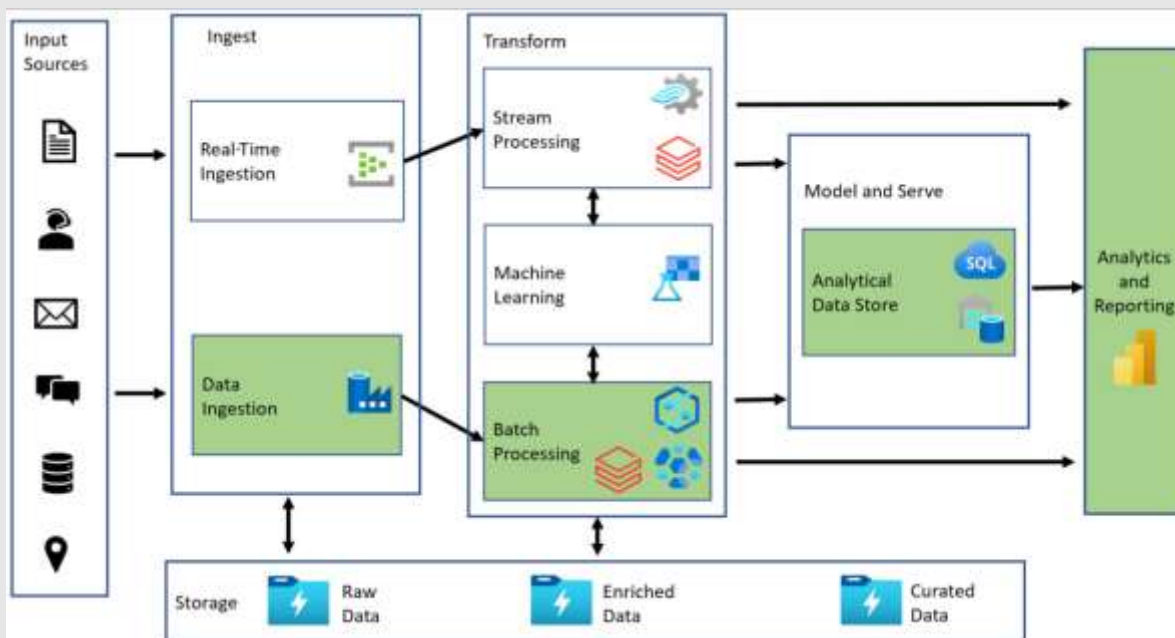


Figura 9.1 - Arquitectura de procesamiento por lotes

Una solución de procesamiento por lotes suele constar de cinco componentes principales:

- Sistemas de almacenamiento como Azure Blob storage, ADLS Gen2, HDFS o similares
- Sistemas de transformación/procesamiento por lotes como Spark, SQL o Hive (a través de Azure HDInsight)

- Analytical data stores como Synapse Dedicated SQL pool, Cosmos DB y HBase (a través de Azure HDInsight)
- Sistemas de orquestación como ADF y Oozie (a través de Azure HDInsight)
- Sistemas de informes de Business Intelligence (BI) como Power BI

Ya hemos explorado muchas de estas tecnologías de forma independiente en los capítulos anteriores. En las próximas secciones, veremos cómo vincularlas para crear un pipeline para un sistema de procesamiento por lotes.

Pero antes de entrar en detalles, quiero presentarte otra tecnología con el mismo nombre: Azure Batch. Azure Batch es un sistema de procesamiento por lotes de propósito general que se puede utilizar para ejecutar la computación de alto rendimiento (HPC). Azure Batch se encarga de ejecutar sus aplicaciones en paralelo en múltiples máquinas virtuales (VM) o contenedores. Azure Batch es un servicio independiente en Azure, mientras que el batch pipeline que estamos discutiendo es una colección de servicios puestos juntos como un pipeline para obtener información de los datos. Exploraremos más sobre Azure Batch más adelante en este capítulo.

Veamos ahora cómo construir un sistema de procesamiento por lotes utilizando ADF, Spark, Synapse, PolyBase y Azure Databricks.

9.3. Desarrollo de soluciones de procesamiento por lotes utilizando Data Factory, Data Lake, Spark, Azure Synapse Pipelines, PolyBase y Azure Databricks

Vamos a intentar construir un batch pipeline de extremo a extremo utilizando todas las tecnologías listadas en la cabecera del tema. Utilizaremos nuestro ejemplo de Imaginary Airport Cab (IAC) de los capítulos anteriores para crear un requisito de muestra para nuestro pipeline de procesamiento por lotes. Supongamos que estamos obteniendo continuamente datos de viajes de diferentes regiones (códigos postales), que se almacenan en Azure Blob storage, y las tarifas de los viajes se almacenan en un Azure SQL Server. Tenemos el requisito de combinar (merge) estos dos conjuntos de datos y generar informes de ingresos diarios para cada región.

Para atender este requisito, podemos construir un pipeline como se muestra en el siguiente diagrama:

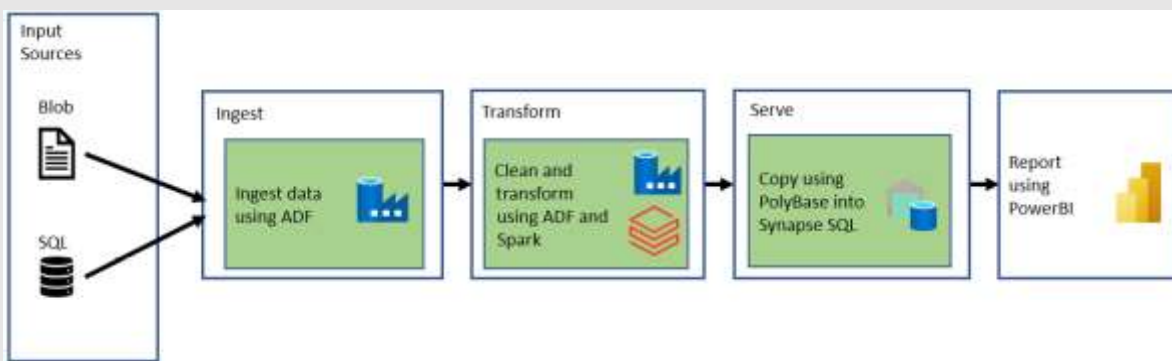


Figura 9.2 - Arquitectura de alto nivel del caso de uso de lotes

El pipeline anterior, traducido a un pipeline ADF, tendría el aspecto de la siguiente figura:

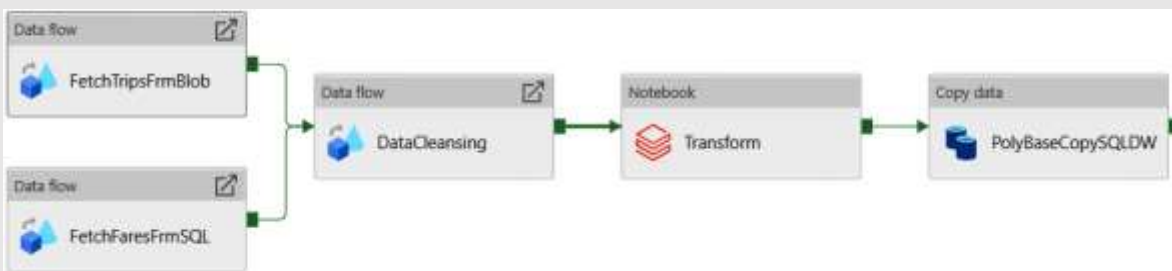


Figura 9.3 - Ejemplo de canalización de datos

Como se puede ver, el pipeline tiene cuatro etapas:

- **Ingesta de datos:** Las dos primeras etapas, FetchTripsFrmBlob y FetchFaresFrmSQL introducen los datos en el data lake.
- **Limpieza de datos:** La etapa DataCleansing del diagrama limpia los datos.
- **Transformación:** La etapa Spark Notebook Transform del diagrama transforma los datos.

- **Carga en una base de datos analítica**: La etapa PolyBaseCopySQLDW para copiar los datos en un Synapse SQL pool.

La última etapa serían las herramientas de BI que leen desde la base de datos analítica y generan informes (que no se muestra en el diagrama ya que no es una actividad de ADF).

Antes de empezar a ver cada una de estas etapas, vamos a definir cuál va a ser nuestro almacenamiento.

9.3.1. Storage

Consideremos ADLS Gen2 como nuestro almacenamiento de data lake. Podemos crear la siguiente estructura de carpetas para manejar nuestro batch pipeline:

Los datos en bruto de los trips pueden ser almacenados aquí:

```
iac/raw/trips/2022/01/01
```

Los datos depurados pueden copiarse en la carpeta transform/in:

```
iac/transform/in/2022/01/01
```

La salida de los datos transformados puede pasar a la carpeta transform/out:

```
iac/transform/out/2022/01/01
```

Por último, podemos importar los datos de transform/out a Synapse SQL Dedicated pool utilizando PolyBase.

Tenga en cuenta que herramientas como ADF y PolyBase también ofrecen la posibilidad de mover directamente los datos entre Spark y Synapse SQL Dedicated pool. Puede elegir este enfoque directo en lugar de almacenar los datos intermedios en el data lake si eso le resulta más conveniente en términos de rendimiento y coste. Pero en la mayoría de los data lakes, más de una herramienta puede acceder a los datos intermedios del data lake y será útil mantener conjuntos de datos históricos para futuros análisis. Por lo tanto, podría tener sentido mantener una copia en el data lake también.

Veamos ahora en detalle cada una de las etapas del batch pipeline.

9.3.2. Ingesta de datos

Este es el proceso de introducir todos los datos en bruto en el data lake. Los datos procedentes de diversas fuentes llegan a la zona sin procesar del data lake. En función de la procedencia de los

datos, como sistemas locales, otros sistemas en la nube, etc., podemos utilizar diferentes herramientas de ingestión. Veamos algunas de las opciones disponibles en Azure:

- **Azure Data Factory** - Ya estás familiarizado con esta tecnología. Proporciona soporte de ingestión de datos desde cientos de fuentes de datos, e incluso desde otras nubes como AWS, GCP, Oracle, etc. Volveremos a utilizarla para construir nuestro pipeline tal y como se recomienda en el temario.
- **Azure Copy (AzCopy)** - Se trata de una herramienta de línea de comandos que se puede utilizar para copiar datos a través de Internet y es ideal para tamaños de datos más pequeños (preferiblemente en el rango de 10-15 TB). Puede obtener más información sobre AzCopy aquí: <https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10>.
- **Azure ExpressRoute** - Si necesita una forma segura de transferir datos a Azure, utilice ExpressRoute. Dirige sus datos a través de conexiones privadas dedicadas a Azure en lugar de la Internet pública. Esta es también la opción preferida si desea tener un pipeline dedicado con una velocidad de transferencia de datos más rápida. Puede obtener más información sobre Azure ExpressRoute aquí: <https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction>.

Consideremos un ejemplo de ingesta utilizando ADF para leer desde el Blob store:

1. Para conectar primero con el Blob store, tendrás que crear un linked service en ADF. Puede crear uno desde la pestaña Manage de ADF como se muestra en la siguiente captura de pantalla.

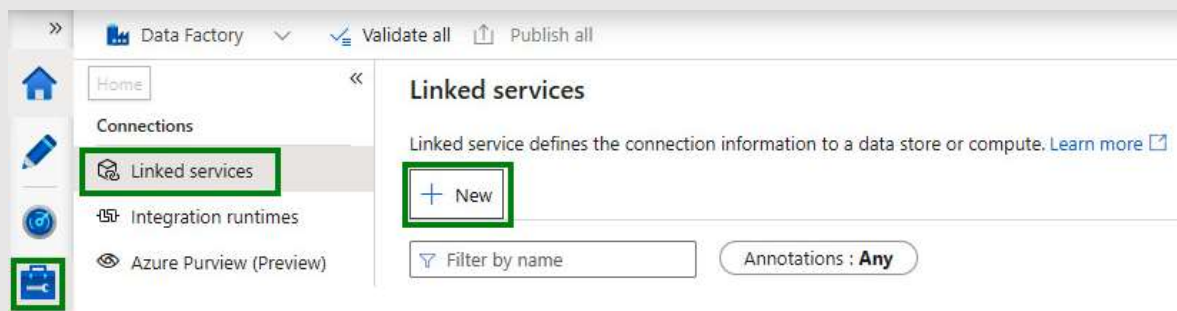


Figura 9.4 - Configuración de un servicio vinculado en ADF

Haz clic en el símbolo + Nuevo y elige Azure Blob storage de la lista. Obtendrás una página como la que se muestra en la siguiente captura de pantalla. Una vez que la rellenes y crees el servicio vinculado, podrás acceder a los datos de tu Blob storage directamente desde ADF.

New linked service (Azure Blob Storage)

Name *
AzureBlobStorage1

Description

Connect via integration runtime * ⓘ
AutoResolveIntegrationRuntime

Authentication method
Account key

Connection string Azure Key Vault

Account selection method ⓘ
☒ From Azure subscription ☐ Enter manually

Azure subscription ⓘ
Select all

Storage account name *
[Empty field]

Additional connection properties
+ New

Test connection ⓘ
☒ To linked service ☐ To file path

Create Back Test connection Cancel

Figura 9.5 - Creación de un linked service de Azure blob storage

El siguiente paso es crear un data flow en ADF.

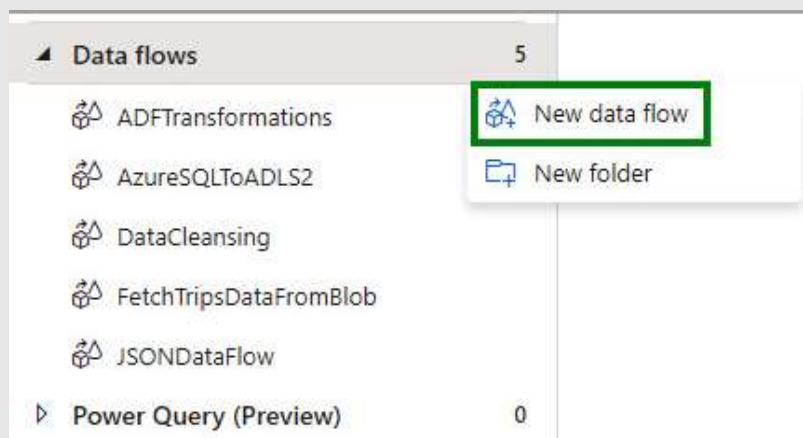


Figura 9.6 - Creación de un nuevo data flow en ADF

En el data flow, tendrás que especificar el origen y el destino. Para el tipo de origen, seleccione la opción Dataset y defina un nuevo dataset utilizando el servicio vinculado de Blob storage que creó anteriormente.

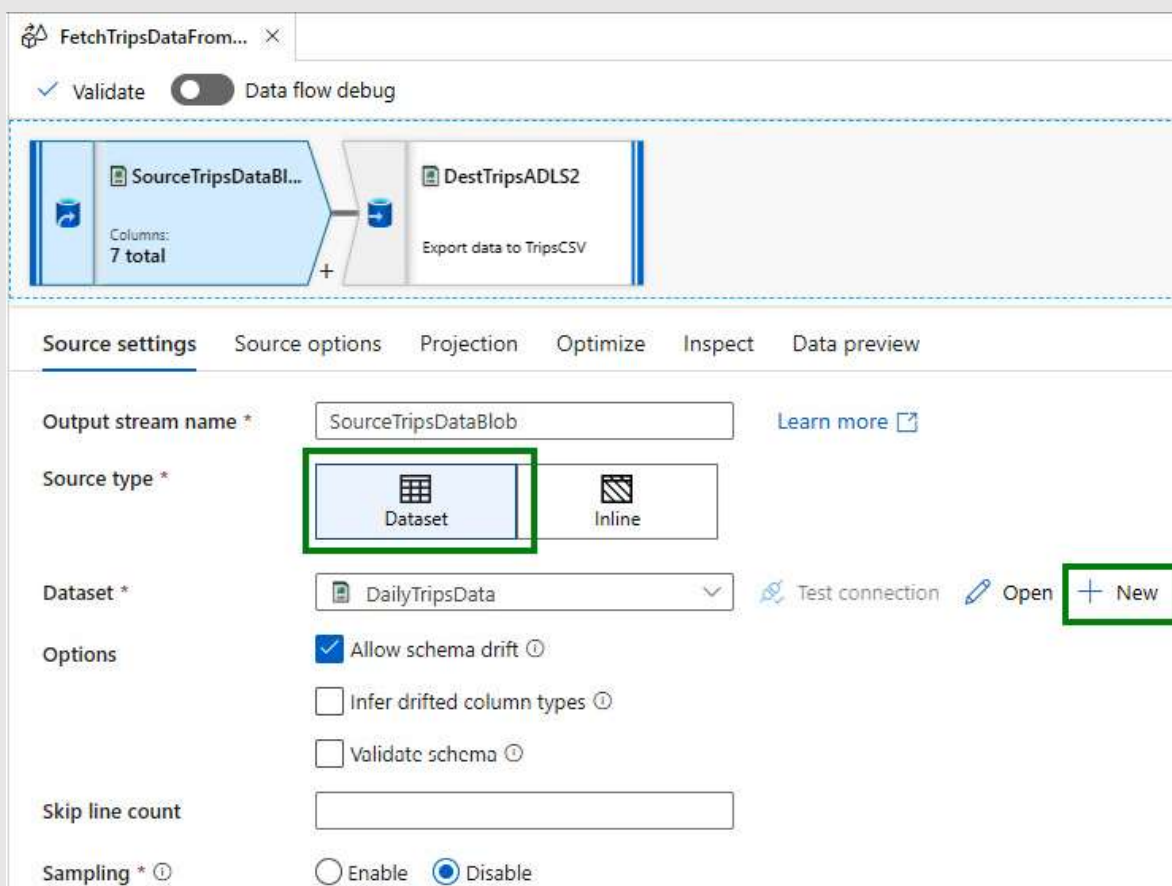


Figura 9.7 - Creación de datasets de origen y destino en un data flow

Este es el aspecto de la pantalla de creación de dataset cuando se hace clic en + Nuevo para la opción Dataset.

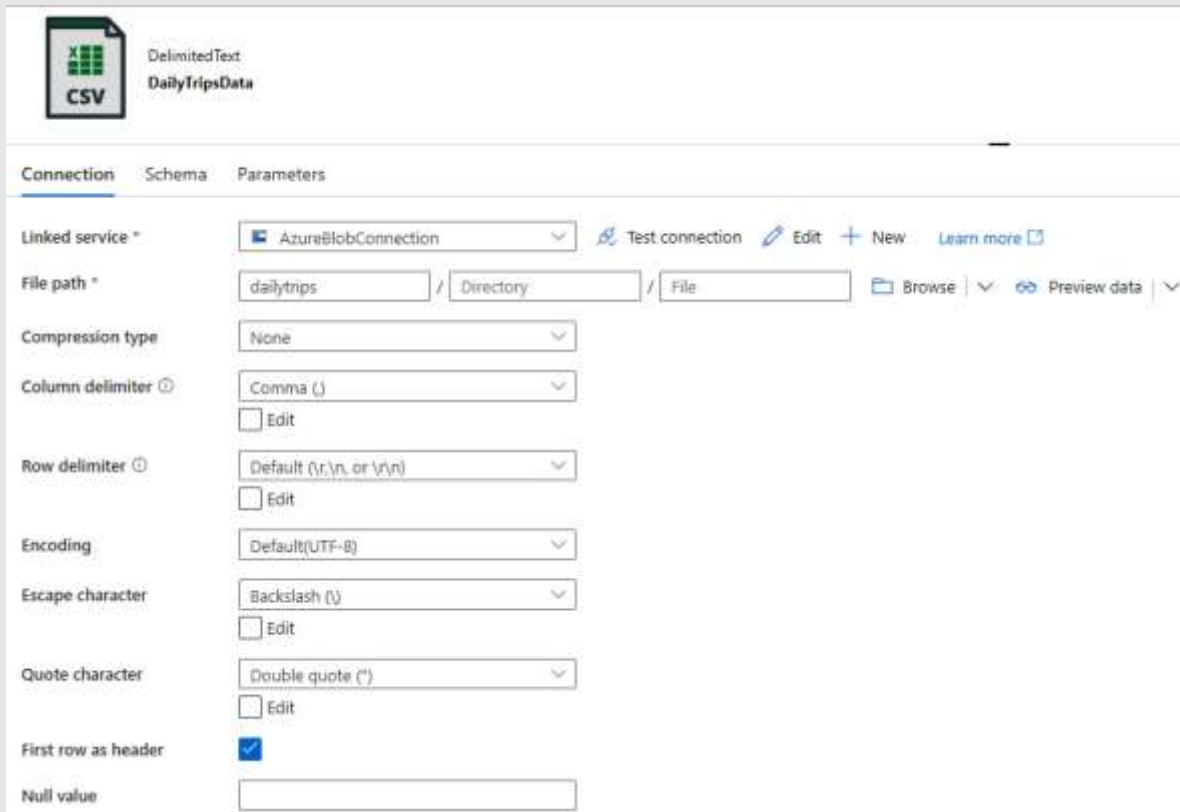


Figura 9.8 - Pantalla del dataset de ejemplo

Con el data flow creado, ya tenemos la parte de ingestión de datos resuelta. Bueno, en realidad el paso aún no está completo. Todavía tenemos que añadir este paso de copia de datos a nuestro pipeline y ejecutarlo para ver los resultados. Pero antes de eso, vamos a conocer todos los componentes restantes del procesamiento por lotes. Veamos ahora el componente de preparación/limpieza de datos.

9.3.3. Preparación de datos/limpieza de datos

En el Capítulo 8, Ingesta y transformación de datos, exploramos múltiples transformaciones de limpieza de datos, como el manejo de datos faltantes, la eliminación de datos atípicos, etc. Puede utilizar exactamente las mismas transformaciones para esta etapa. Como ya hemos explorado los detalles en el capítulo anterior, no los repetiremos de nuevo aquí. Pasemos a la etapa de transformación.

9.3.4. Transformación

Una vez que nuestros datos están limpios y preparados, podemos ejecutar nuestra lógica de transformación utilizando servicios como Spark, SQL, Hive, etc. En este ejemplo, según el programa de certificación, utilizaremos Azure Databricks Spark. Estos son los pasos:

1. Crear un área de trabajo ADB - seleccionar Azure Databricks en el portal de Azure y crear una nueva área de trabajo como se muestra en la siguiente captura de pantalla.

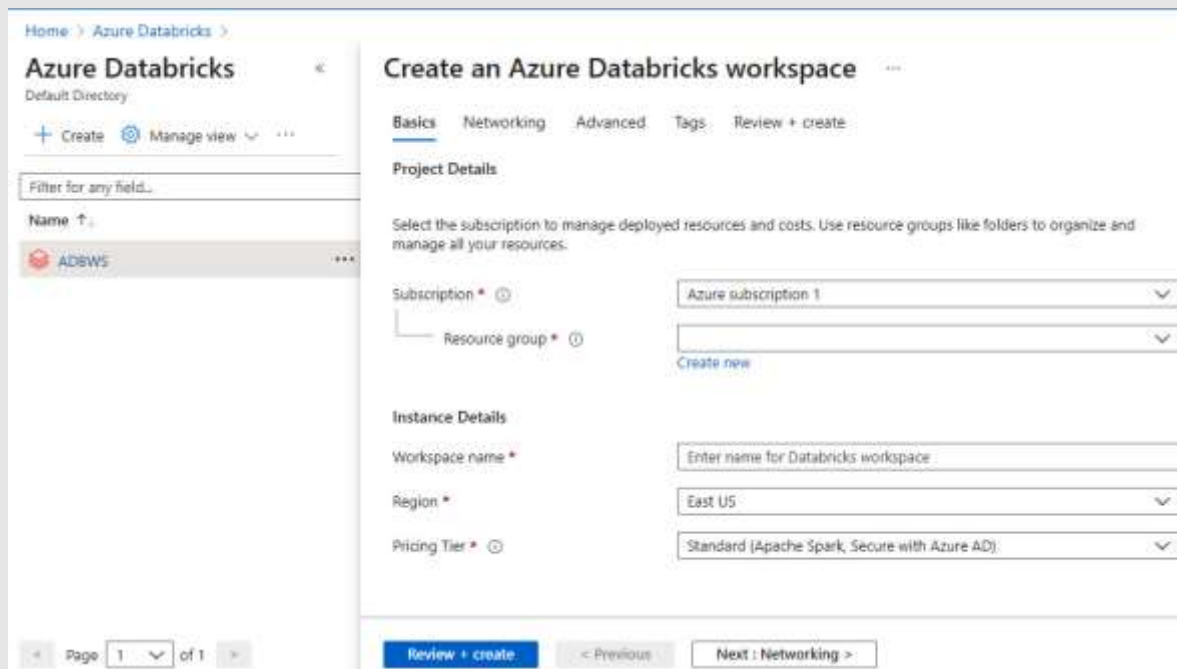


Figura 9.9 - Ejemplo de la página de creación del área de trabajo de Azure Databricks

2. Una vez creado el área de trabajo, haz clic en el botón Launch Workspace para lanzar el área de trabajo. Esto le llevará al portal de Azure Databricks.

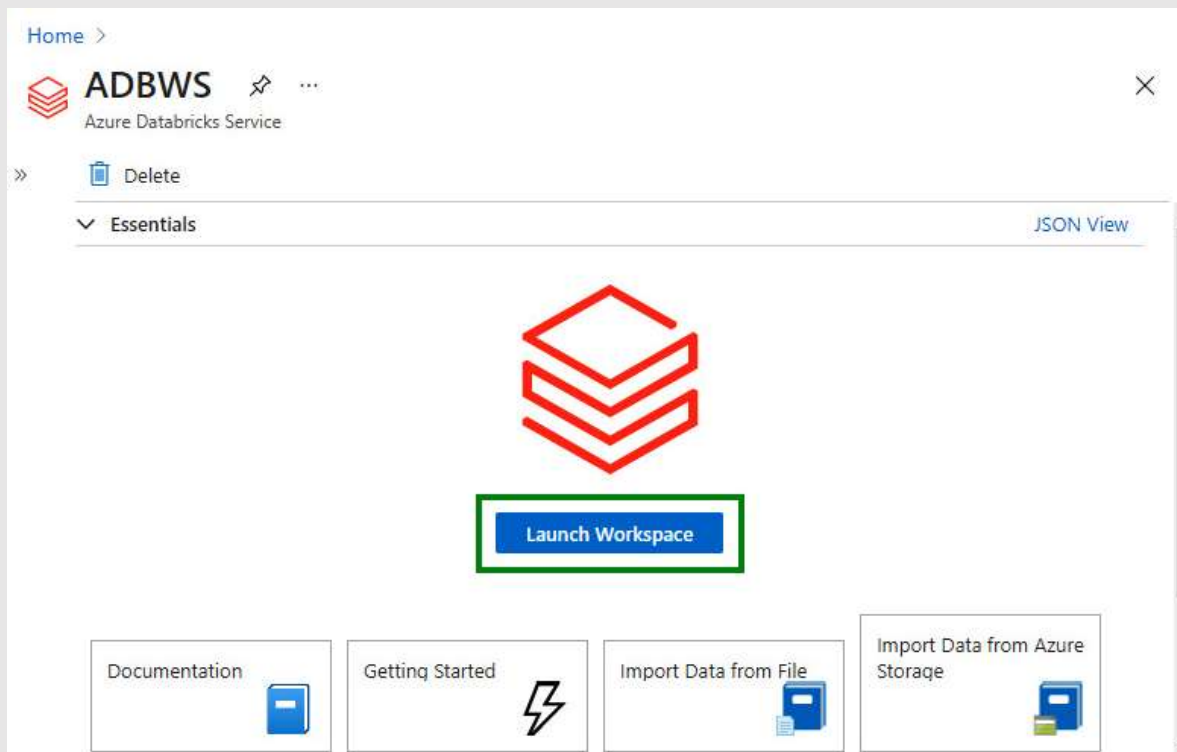


Figura 9.10 - Lanzamiento del área de trabajo de Azure Databricks

Desde esta pantalla, puedes crear clusters, notebooks y más. Esto nos lleva al siguiente paso.

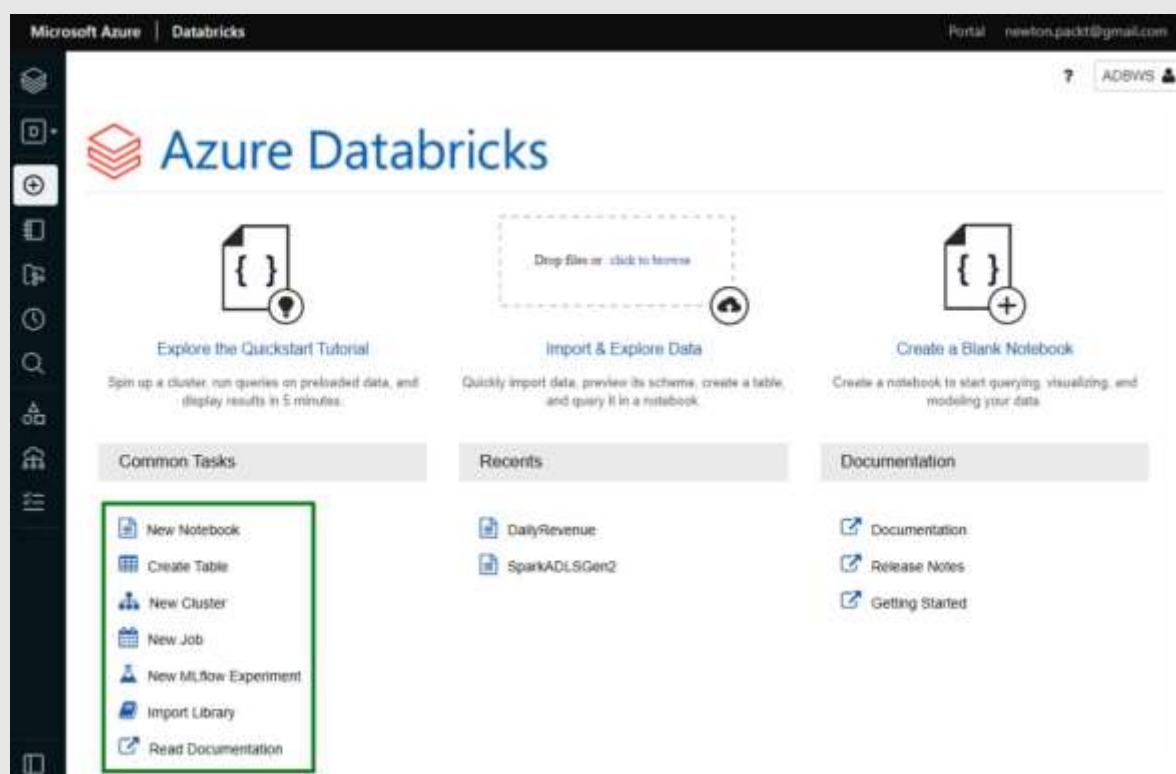


Figura 9.11 - Portal de Azure Databricks

3. Crea un cluster ADB - haz clic en el enlace Nuevo Cluster en el portal de Databricks e introduce los detalles.

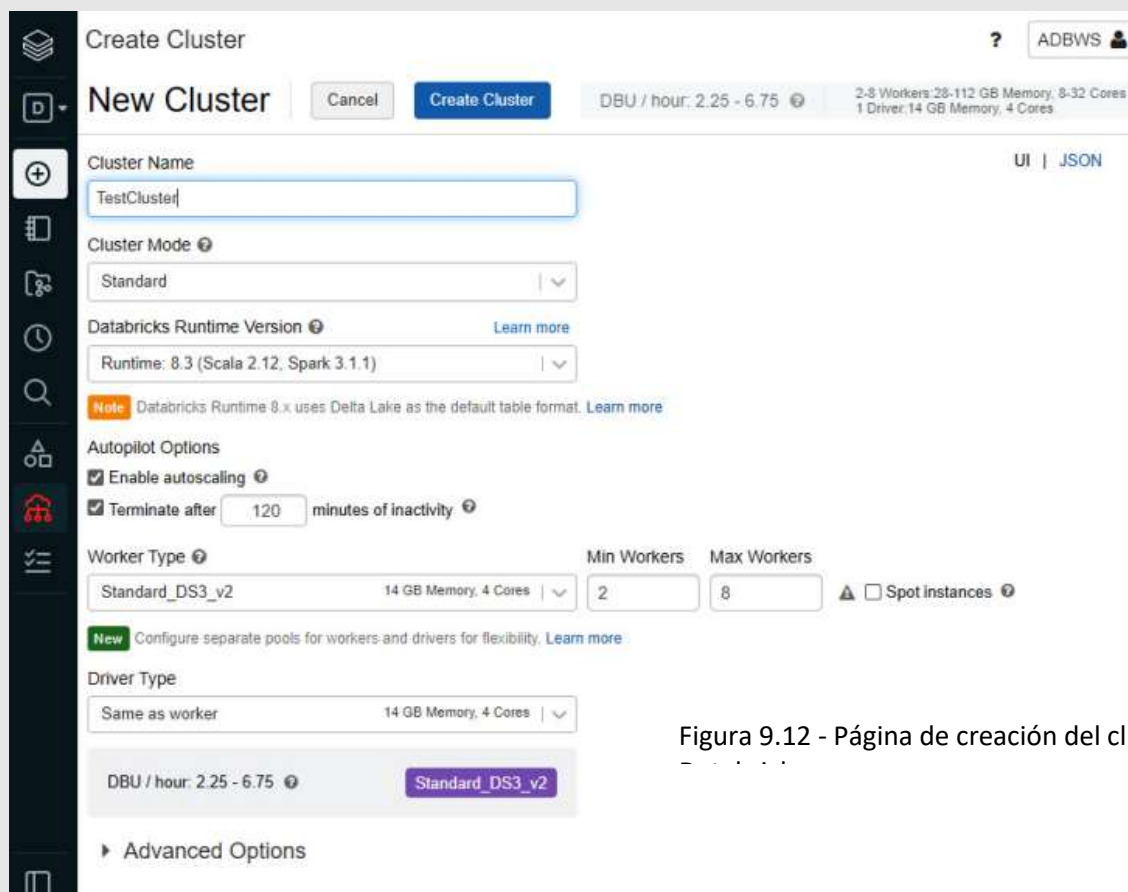


Figura 9.12 - Página de creación del cluster Azure

Esto creará el cluster que será necesario para ejecutar las transformaciones para el batch pipeline.

4. A continuación, haz clic en Create y luego en Notebook para construir tu lógica de transformación - desde el portal de Azure Databricks, puedes elegir New Notebooks, o hacerlo desde las pestañas laterales como se muestra:

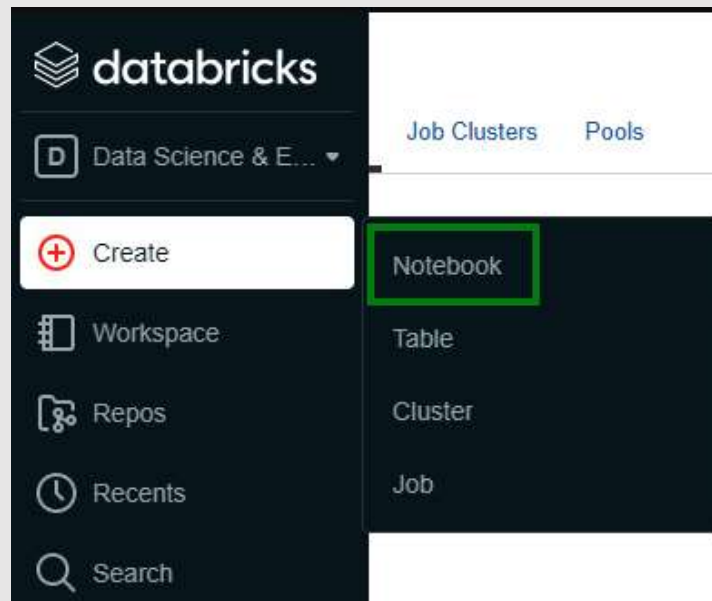


Figura 9.13 - Creación de nuevos notebooks en Azure Databricks

Una vez creado el nuevo Notebook, obtendrás un editor como el que se muestra en la siguiente captura de pantalla. Puedes escribir el código Spark dentro de los bloques Cmd. Azure Databricks soporta los lenguajes Scala, Python, SQL y R.

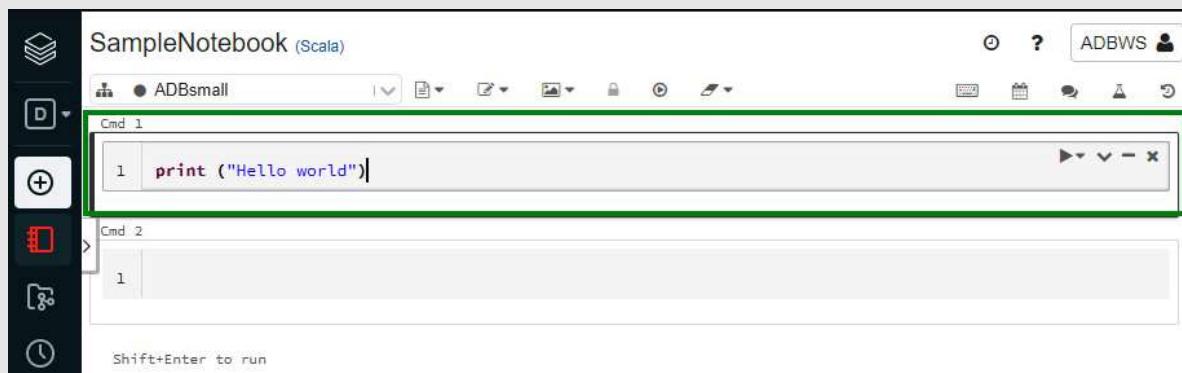


Figura 9.14 - Azure Databricks notebooks

Para generar un informe de viaje diario, podemos escribir un simple script de transformación Scala como se muestra en los siguientes pasos. Tendrá que escribirlos en las secciones Cmd que se muestran en la captura de pantalla anterior:

1. En primer lugar, tenemos que establecer la configuración inicial para que ADB Spark pueda hablar con ADLS Gen2:

```
spark.conf.set("fs.azure.account.auth.type." + storageAccountName + ".dfs.core.windows.net",
"OAuth")

spark.conf.set("fs.azure.account.oauth.provider.type." + storageAccountName +
".dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")

spark.conf.set("fs.azure.account.oauth2.client.id." + storageAccountName +
".dfs.core.windows.net", "" + appId + "")

spark.conf.set("fs.azure.account.oauth2.client.secret." + storageAccountName +
".dfs.core.windows.net", "" + secret + "")

spark.conf.set("fs.azure.account.oauth2.client.endpoint." + storageAccountName +
".dfs.core.windows.net", "https://login.microsoftonline.com/" + tenantID + "/oauth2/token")

spark.conf.set("fs.azure.createRemoteFileSystemDuringInitialization", "true")

spark.conf.set("fs.azure.createRemoteFileSystemDuringInitialization", "false")
```

2. Leer los datos del viaje (almacenados como un archivo CSV):

```
%scala
.add("tripId",IntegerType)
.add("driverId",IntegerType)
.add("customerId",IntegerType)
.add("cabId",IntegerType)
.add("tripDate",IntegerType)
.add("startLocation",StringType)
.add("endLocation",StringType)

val tripsCSV = spark.read.format("csv")
    .option("header", "true")
    .schema(tripsSchema)
    .load("abfss:/path/to/csv")
```

3. Leer los datos de las tarifas (almacenados como archivos Parquet):

```
%scala
val faresSchema = new StructType()
    .add("tripId",IntegerType)
    .add("fare",IntegerType)
    .add("currency",StringType)
```



```
val faresParquet = spark.read.format("parquet")  
    .schema(faresSchema)  
    .load("abfss:/path/to/parquet")
```

4. Unirlos con tripld y agrupar por startLocation:

```
val joinDF = tripsCSV.join(  
    faresParquet, tripsCSV("tripld") ===  
        faresParquet("tripld"), "inner")  
    .groupBy("startLocation")  
    .sum("fare");
```

5. Imprimir la tabla de salida con las columnas Ciudad y Tarifa (Fare):

```
import org.apache.spark.sql.functions.col;  
val outputDF = joinDF.select(col("startLocation").alias("City"), col("sum(fare)").alias("Fare"));
```

6. Por último, escriba la salida en ADLS Gen2 en la carpeta transform/fares/out:

```
outputDF.write.mode("overwrite").parquet("abfss:/path/to/output")
```

Intenta ejecutar el código y depurar cualquier problema dentro de Azure Databricks antes de conectarlo al pipeline de ADF.

TIP

Azure Databricks proporciona una biblioteca llamada dbutils que se puede utilizar para las operaciones del sistema de archivos, tales como la lista de archivos y la gestión de los secretos y las operaciones de datos, tales como el resumen de los conjuntos de datos y así sucesivamente. Aquí hay un ejemplo:

```
dbutils.fs.cp("/ruta/al/fuente.txt", "/ruta/al/destino.txt")
```

Puedes aprender más sobre dbutils aquí: <https://docs.microsoft.com/en-us/azure/databricks/dev-tools/databricks-utils>.

A continuación, vamos a ver cómo configurar ADF para llamar al notebook de Spark que acabamos de crear.

9.3.5. Configuración de una actividad ADB notebook en ADF

Estos son los pasos para añadir un notebooks ADB en un pipeline de ADF:

1. En la pestaña Actividades en Azure Data Factory, elige Notebook en Databricks y añádelo al pipeline arrastrando el icono al área de la hoja de trabajo como se muestra en la siguiente captura de pantalla.

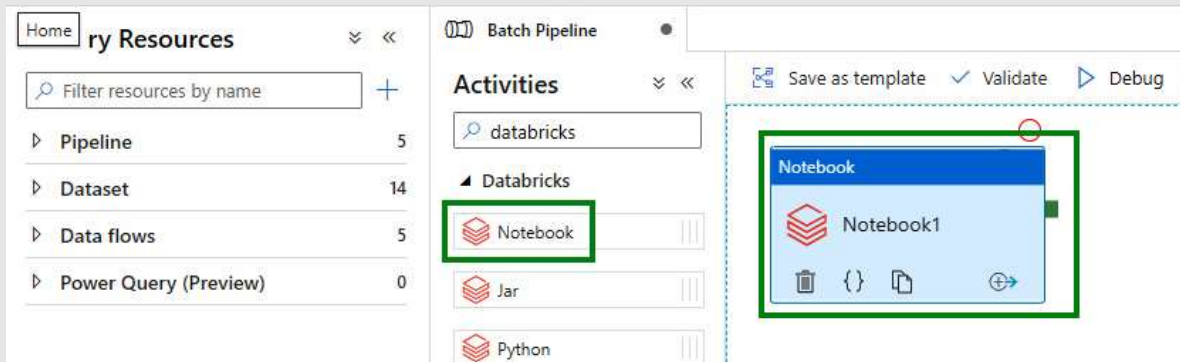


Figura 9.15 - Elección de una actividad Azure Databricks en ADF

2. Tienes que enlazar esta actividad Notebook con el notebook que has creado en el paso anterior. Para vincular el notebook, primero tendrás que obtener el token de acceso de Azure Databricks. Puedes generar el token de acceso desde el portal de Azure Databricks desde la pestaña de Configuración de Usuario como se muestra en la siguiente captura de pantalla. Haz clic en el botón Generar nuevo token para crear un nuevo token de acceso.

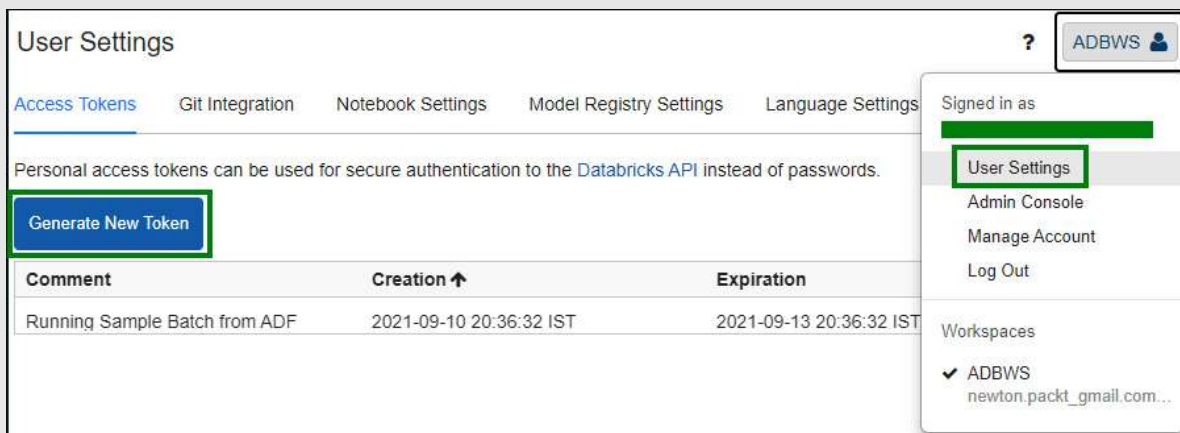


Figura 9.16 - Obtención del token de acceso desde Azure Databricks

3. Ahora, vincula el notebook ADB creado anteriormente utilizando un servicio vinculado. De forma similar a como creamos un servicio vinculado para Azure Blob storage, necesitamos crear uno para Azure Databricks Spark también. La siguiente pantalla muestra la configuración del servicio vinculado para Azure Databricks. Tendrás que rellenar la URL del área de trabajo de Databricks, el campo Access token - con el token de acceso que generaste en el paso anterior, y

seleccionar si quieres crear un Nuevo cluster de trabajo o apuntar a un cluster interactivo Existente, etc.

The screenshot shows the 'Edit linked service (Azure Databricks)' configuration window. The 'Access token' field is highlighted with a green box. The configuration includes the following details:

- Name:** ADBatchPool
- Description:** (Empty text box)
- Connect via integration runtime:** AutoResolveIntegrationRuntime
- Account selection method:** Enter manually
- Databricks Workspace URL:** https://adb-364728729096317.17.azuredatabricks.net
- Authentication type:** Access Token
- Access token:** (Redacted with asterisks)
- Select cluster:** New job cluster (selected), Existing interactive cluster, Existing instance pool
- Cluster version:** 8.0.x-scala2.12
- Cluster node type:** Standard_F4
- Python Version:** 2
- Worker options:** Fixed, Autoscaling (selected)
- Min Workers:** (Empty text box)

Figura 9.17 - Creación de un servicio vinculado de Azure Databricks

Una vez que hayas creado el servicio vinculado e introducido esos datos en la actividad del notebook de ADF, tu etapa de transformación de ejemplo estará completa. El último paso que queda pendiente es importar los datos de esta etapa de transformación a SQL Dedicated pool y servir los datos desde allí a Power BI. Pero antes de ir allí, veamos las opciones disponibles para el procesamiento por lotes en Azure.

9.3.6. Opciones de tecnología de procesamiento por lotes

Aquí hay una tabla útil reproducida de Azure que puede ayudarle a decidir sobre las tecnologías a utilizar para sus escenarios de lotes:

Capability	Azure Data Lake Analytics	Azure Synapse	HDInsight with Spark	HDInsight with Hive	HDInsight with Hive LLAP	Azure Databricks
Autoscaling	No	No	Yes	Yes	Yes	Yes
Scale-out granularity	Per job	Per cluster	Per cluster	Per cluster	Per cluster	Per cluster
In-memory caching of data	No	Yes	Yes	No	Yes	Yes
Query from external relational stores	Yes	No	Yes	No	No	Yes
Authentication	Azure AD	SQL / Azure AD	No	Azure AD ¹	Azure AD ¹	Azure AD
Auditing	Yes	Yes	No	Yes ¹	Yes ¹	Yes
Row-level security	No	Yes ²	No	Yes ¹	Yes ¹	No
Supports firewalls	Yes	Yes	Yes	Yes ³	Yes ³	No
Dynamic data masking	No	Yes	No	Yes ¹	Yes ¹	No

Figura 9.18 - Comparación de las tecnologías de procesamiento por lotes en Azure

Puede obtener más información sobre las opciones de procesamiento por lotes aquí: <https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/batch-processing>.

Veamos a continuación cómo copiar los datos generados desde la etapa de transformación en Synapse SQL utilizando PolyBase.

9.3.7. Utilización de PolyBase para introducir los datos en el almacén de datos de Analytics

PolyBase es una herramienta que permite a servicios como SQL Server y Synapse Dedicated SQL pool copiar y consultar datos directamente desde ubicaciones externas. Las fuentes externas podrían ser Azure Storage, Oracle, Teradata, Hadoop, MongoDB, etc. **PolyBase está integrado en T-SQL, por lo que cada vez que usamos un comando COPY INTO <table> FROM para leer datos de una ubicación de almacenamiento externa, PolyBase entra en acción.** PolyBase es una de las formas más rápidas y escalables de copiar datos.

Para el escenario del data lake, vamos a utilizar PolyBase para copiar los datos transformados de Azure Databricks en el Synapse Dedicated SQL pool utilizando un staging ADLS o Blob store. Los pasos para hacerlo son los siguientes:

1. Preparar los datos de origen en archivos de texto en ADLS o en el Blob store.
2. Defina una tabla externa con el esquema adecuado en la instancia del Dedicated SQL pool. El formato de los datos entrantes debe coincidir exactamente con el esquema de la tabla externa. Si no es así, las filas pueden ser eliminadas.
3. Si los datos provienen de una fuente no relacional, tendremos que transformarlos en un formato de filas y columnas que se ajuste correctamente al esquema de la tabla externa.
4. Ejecute el comando COPY INTO para cargar los datos en tablas externas dedicated SQL pool utilizando PolyBase.
5. A partir de aquí, puede servir los datos directamente o hacer más procesamiento utilizando un Dedicated SQL pool antes de servirlos a las herramientas de BI.

A continuación se muestra un ejemplo de cómo utilizar PolyBase:

```
CREATE EXTERNAL FILE FORMAT [Dp203ParquetFormat]
WITH ( FORMAT_TYPE = PARQUET)

CREATE EXTERNAL DATA SOURCE [Dp203DataSource]
WITH (
LOCATION = 'abfss://path/to/storage/location'
)

CREATE EXTERNAL TABLE TripExtTable
WITH (
LOCATION = '/path/to/data/*.parquet',
DATA_SOURCE = [Dp203DataSource],
FILE_FORMAT = [Dp203ParquetFormat]
) AS
SELECT
[tripId] INT,
[driverId] INT,
...
[endLocation] VARCHAR(50)
FROM
    OPENROWSET(BULK '/path/to/data/*.parquet', FORMAT='PARQUET')
```

Ahora copie los datos de la tabla externa en una tabla SQL real:

```
CREATE TABLE TripsProdTable
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = ROUND_ROBIN
)
AS
SELECT * FROM TripsExtTable
```

Con la última sentencia CREATE EXTERNAL TABLE AS SELECT (CETAS), PolyBase copia los datos en Synapse SQL Dedicated pool.

NOTA

PolyBase no admite formatos anidados como JSON, XML y WinZip al momento de escribir este libro. Para los archivos JSON, podría intentar aplanar los datos primero usando las técnicas que vimos en el Capítulo 8, Ingesta y Transformación de Datos, en la sección de Shredding de JSON, antes de usar PolyBase para cargarlos en Synapse SQL.

Opciones para cargar con PolyBase

PolyBase también está disponible como parte de otros servicios, como los siguientes

- **Azure Data Factory**: esta versión de PolyBase puede utilizarse como una actividad dentro de ADF. La actividad de copia de datos puede ser definida como un pipeline que puede ser programado regularmente. Puede leer más sobre ello aquí: <https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-sql-data-warehouse?tabs=data-factory#use-polybase-to-load-data-into-azure-synapse-analytics>.
- **Azure Databricks** - Esta versión de PolyBase puede utilizarse para transferir datos entre Azure Databricks y los pools de Synapse SQL. Puede obtener más información al respecto aquí: <https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-extract-load-sql-data-warehouse>.
- **SQL Server** - Esta versión se puede utilizar si el origen es un SQL Server. La plataforma de Servicios de Integración de SQL Server (SSIS) se puede utilizar para definir los mapeos de origen y destino y hacer la orquestación mientras se utiliza la versión SSIS de PolyBase. Puede obtener más información al respecto aquí: <https://docs.microsoft.com/en-us/sql/integration-services/sql-server-integration-services>.

Puede obtener más información sobre PolyBase aquí: <https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-versioned-feature-summary>.

Veamos ahora la última pieza de nuestro rompecabezas en la tubería de procesamiento por lotes, que es la visualización de los datos utilizando Power BI.

9.3.8. Uso de Power BI para visualizar los datos

Power BI es una herramienta de visualización de datos de Business Intelligence (BI) que está profundamente integrada con los servicios en la nube de Azure. Tiene conectividad incorporada a muchas nubes y tecnologías locales y ayuda a visualizar y compartir información de los datos.

Los pasos para visualizar los datos de Synapse SQL a través de Power BI son los siguientes:

1. Ir a powerbi.microsoft.com y crear una cuenta. Una vez que haya iniciado sesión, cree un nuevo espacio de trabajo haciendo clic en el botón Crear un área de trabajo de la pestaña Áreas de trabajo, como se muestra en la siguiente captura de pantalla.

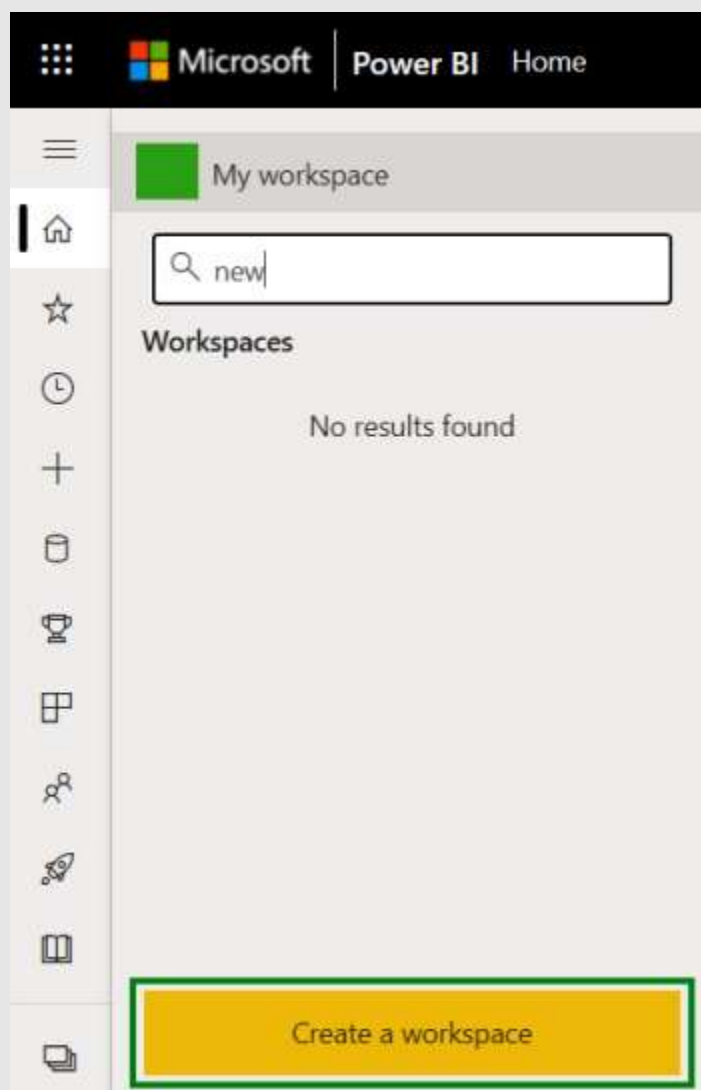


Figura 9.19 - Creación de un área de trabajo en Power BI

2. A continuación, vaya al área de trabajo de Synapse y cree un servicio vinculado a Power BI. Haga clic en la pestaña Gestionar del área de trabajo de Synapse y seleccione Servicios vinculados. A continuación, haga clic en + Nuevo y busque Power BI como se muestra en la siguiente captura de pantalla.

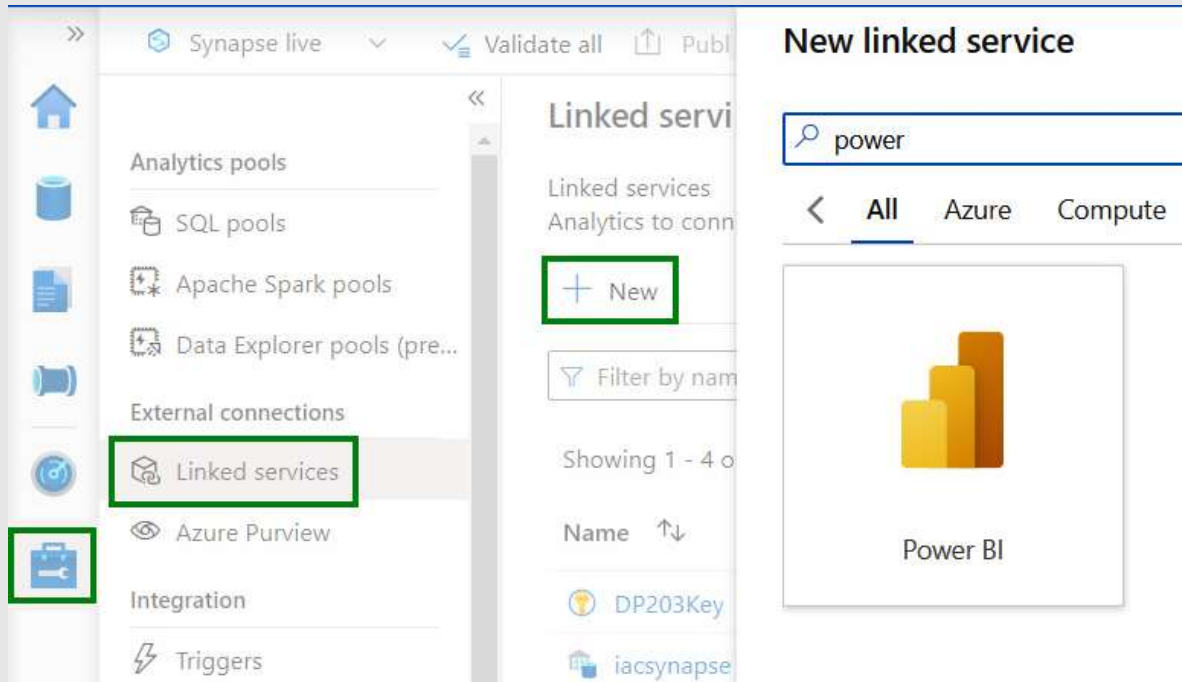


Figura 9.20 - Creación de un servicio vinculado a Power BI

3. En la pantalla de configuración del servicio vinculado, dé un nombre al servicio vinculado y seleccione el área de trabajo de Power BI previamente creada para el campo Nombre del área de trabajo.

New linked service (Power BI)

i Choose a name for your linked service. This name cannot be updated later.

Name *

PowerBIWorkspace

Description

Tenant

Microsoft ()

Workspace name *

PowerBITripWS ()

☐ Edit

Annotations

+ New

> Advanced ⓘ

Create Back Cancel

Figura 9.21 - Configuración del servicio vinculado de Power BI

- Una vez que rellene los campos y haga clic en Crear, se creará un nuevo servicio vinculado a Power BI.
- Ahora vaya a la pestaña Editor en el área de trabajo de Synapse. Debería ver una sección de Power BI como se muestra en la siguiente captura de pantalla. Haga clic en + Nuevo conjunto de datos de Power BI y elija la tabla SQL que desea incluir como su conjunto de datos de Power BI.

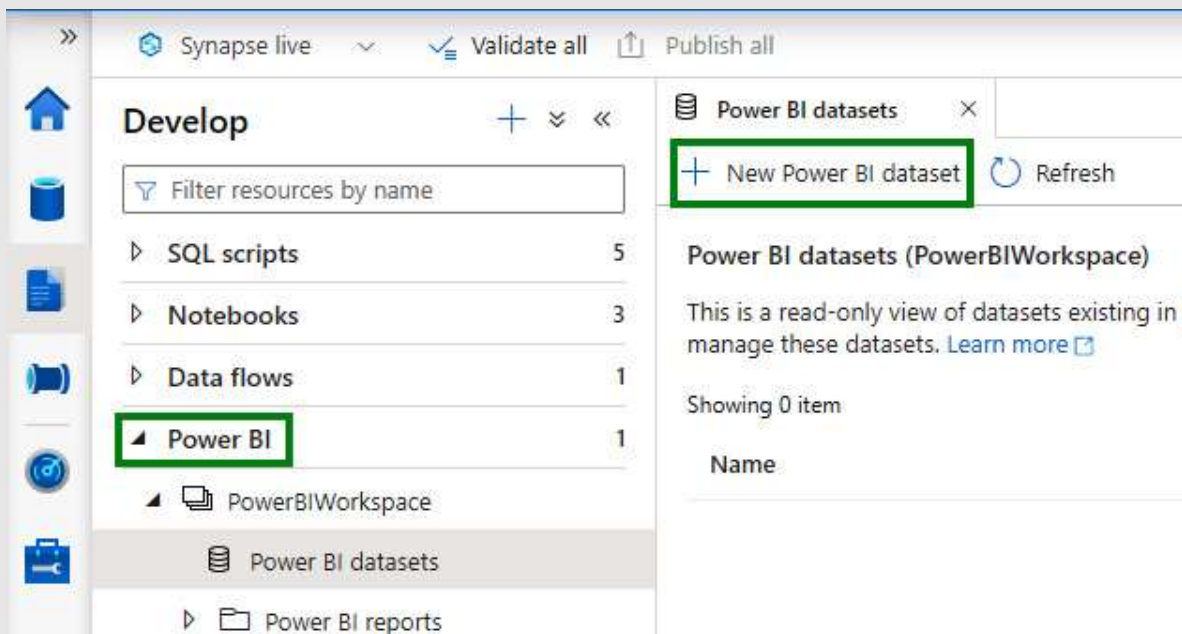


Figura 9.22 - Creación de un nuevo conjunto de datos de Power BI en Synapse

6. Aparecerá una pantalla de descarga como se muestra en la siguiente captura de pantalla. Simplemente descargue el archivo .pbids.

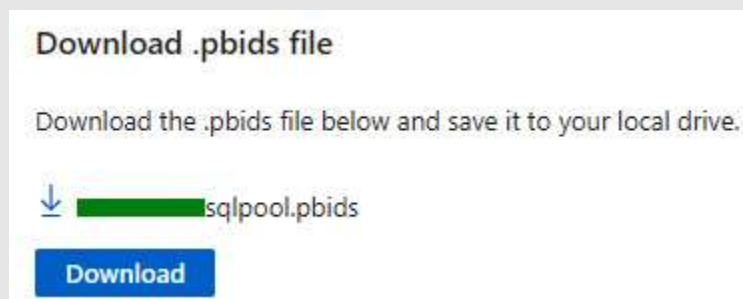


Figura 9.23 - Descarga del archivo .pbids para Power BI

7. A continuación, haga doble clic en el archivo .pbids y se abrirá Power BI Desktop. Si no tiene Power BI Desktop, puede instalarlo desde aquí: <https://PowerBI.microsoft.com/en-us/downloads/>.
8. Una vez que se abra Power BI Desktop, elija cualquiera de las Visualizaciones y arrastre y suelte los campos de la tabla desde la pestaña de Campos de la derecha. Power BI mostrará ahora los gráficos de sus datos.

9.4. Creación de data pipelines

Los data pipelines son una colección de varias actividades de procesamiento de datos organizadas en una secuencia particular para producir los conocimientos deseados a partir de los datos brutos. Ya hemos visto muchos ejemplos en Azure Data Factory donde encadenamos las actividades para producir un resultado final deseable. ADF no es la única tecnología disponible en Azure. Azure también admite pipelines de Synapse (que es una implementación de ADF dentro de Synapse) y tecnologías de código abierto como Oozie (disponible a través de Azure HDInsight), que puede ayudar a orquestar pipelines. Si su carga de trabajo sólo utiliza software de código abierto, entonces Oozie podría encajar. Pero si la canalización utiliza otros servicios de Azure o de terceros externos, entonces ADF podría ser una mejor opción, ya que ADF proporciona plugins de origen y de descarga fácilmente disponibles para una enorme lista de tecnologías.

Puedes crear un pipeline desde la pestaña Pipeline de Azure Data Factory. Todo lo que tienes que hacer es seleccionar las actividades para tu pipeline desde la pestaña de Actividades y hacer clic y arrastrar en el lienzo. Puede enlazar las actividades utilizando el cuadro verde (a la derecha de cada actividad) y encadenar los bloques de forma secuencial o paralela para obtener la salida requerida. La siguiente captura de pantalla muestra un ejemplo.

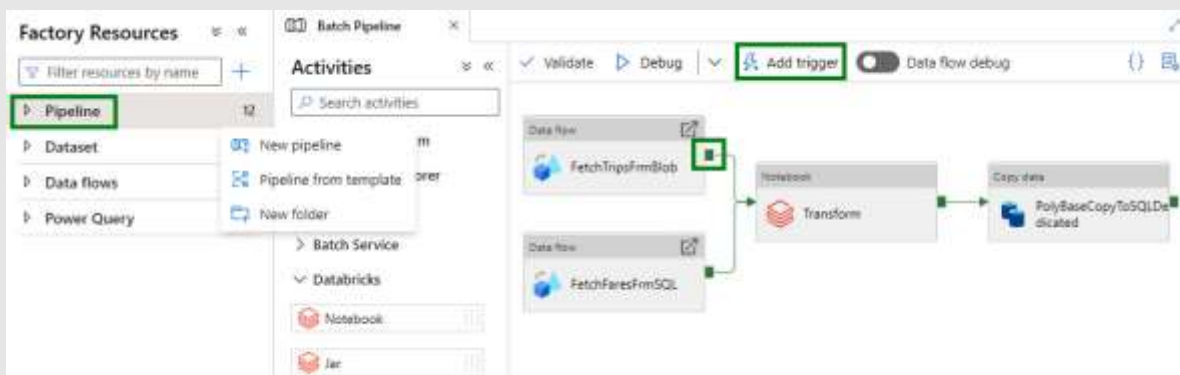


Figura 9.25 - Creación de nuevos pipelines en el ADF

Una vez que tenga el pipeline unido, puede activarlo utilizando el botón Add trigger. El disparador puede ser único, basado en eventos o recurrente.

Espero que ahora comprenda cómo crear y publicar un batch pipeline de extremo a extremo. Veamos ahora cómo integrar los notebooks de Jupyter y Python en un data pipeline.

9.5. Integración de notebooks Jupyter/Python en un data pipelines

La integración de los notebooks Jupyter/Python en nuestro data pipeline de ADF se puede realizar mediante la actividad Spark en ADF. Para este ejercicio necesitarás un clúster Azure HDInsight Spark.

El requisito previo para integrar los notebooks Jupyter es crear servicios vinculados a Azure Storage y HDInsight desde ADF y tener un clúster HDInsight Spark en funcionamiento.

Ya has visto cómo crear servicios vinculados, en la sección Desarrollo de soluciones de procesamiento por lotes mediante el uso de Data Factory, Data Lake, Spark, Azure Synapse Pipelines, PolyBase y Azure Databricks anteriormente en este capítulo, así que no repetiré los pasos aquí.

Seleccione la actividad Spark de ADF y especifique el servicio vinculado de HDInsight que creó en el campo Servicio vinculado de HDInsight en la pestaña Clúster de HDI, como se muestra en la siguiente captura de pantalla.

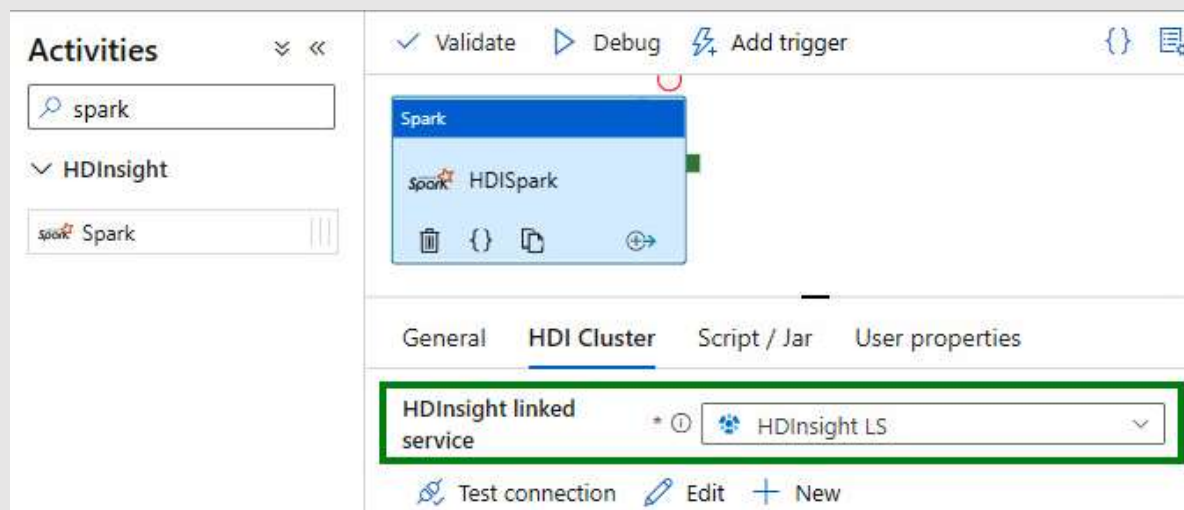


Figura 9.26 - Configuración de una actividad Spark en ADF

Ahora, inicie el notebook de Jupyter yendo a <https://<TUHDICLUSTER>.azurehdinsight.net/jupyter> o desde el dashboard de HDInsight como se muestra en la siguiente captura de pantalla.

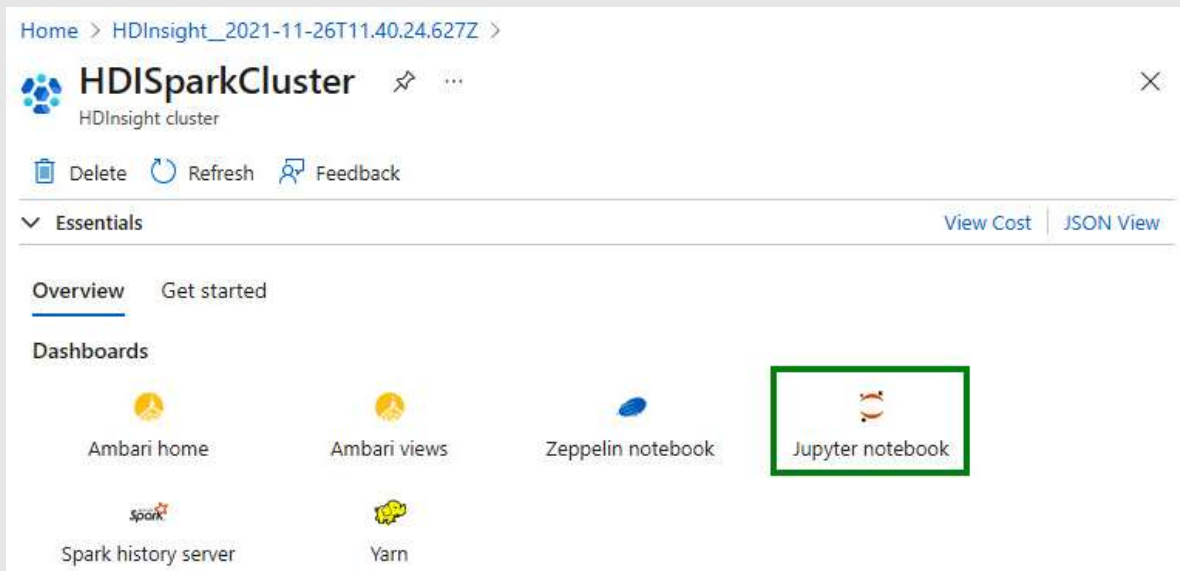


Figura 9.27 - Lanzamiento del notebook Jupyter desde HDInsight

Desde la página de lanzamiento de Jupyter, puedes seleccionar PySpark o PySpark3 para iniciar un notebook de Python.

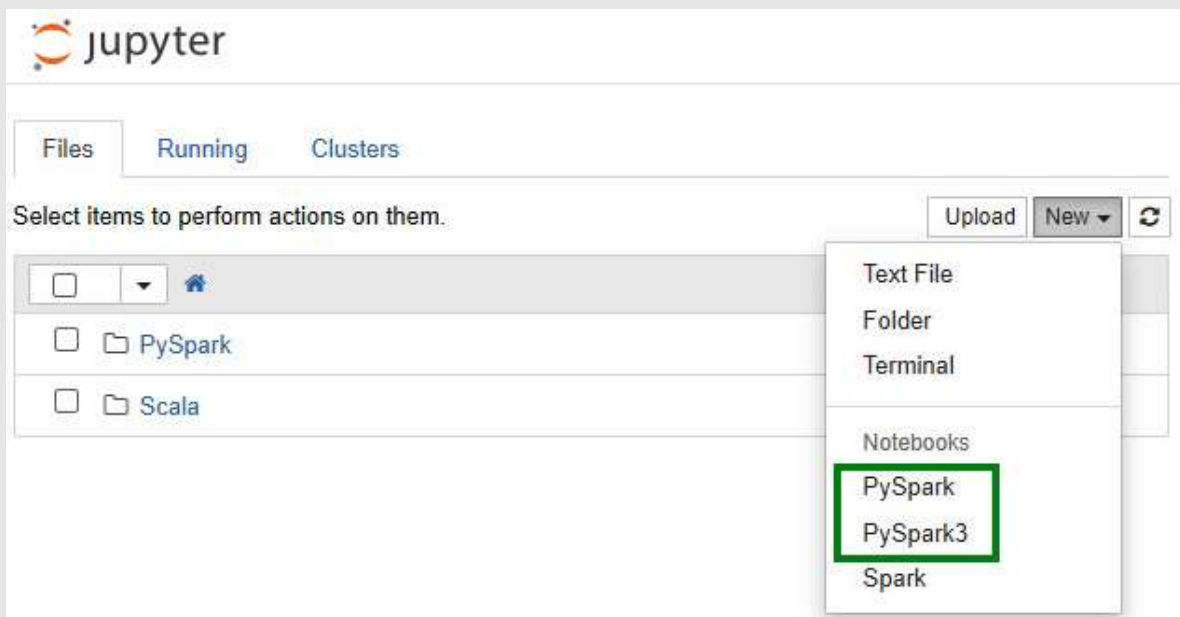


Figura 9.28 - Lanzamiento de un notebook Jupyter PySpark desde HDInsight

Puedes escribir tus transformaciones en el notebook Jupyter y ejecutarlo utilizando los pipelines del ADF como cualquier otra actividad del ADF. Ahora ya sabes cómo ejecutar notebooks Jupyter desde data pipelines.

Puedes aprender más sobre la ejecución de notebooks Spark desde ADF aquí: <https://docs.microsoft.com/en-us/azure/data-factory/v1/data-factory-spark>.

Las siguientes secciones se centrarán en algunas técnicas avanzadas para la carga de datos y la preparación de datos. Algunos de estos temas se repiten de los capítulos anteriores, por lo que los omitiremos proporcionando referencias a los capítulos anteriores y nos centraremos en los nuevos temas que aún no se han cubierto.

9.6. Diseño e implementación de cargas de datos incrementales

Cubrimos la carga incremental de datos en el Capítulo 4, Diseño de la capa de servicio. Consulta ese capítulo para refrescar tus conocimientos sobre las cargas de datos incrementales.

A continuación veremos cómo implementar las dimensiones que cambian lentamente.

9.7. Diseñar y desarrollar dimensiones de cambio lento

En el capítulo 4, "Diseño de la capa de servicio", también tratamos en detalle las dimensiones de cambio lento (SCD). Por favor, consulte ese capítulo para refrescar su conocimiento de los conceptos.

9.8. Manejo de datos duplicados

Ya tratamos este tema en el Capítulo 8, Ingesta y transformación de datos. Por favor, consulte ese capítulo para refrescar sus conocimientos sobre el manejo de datos duplicados.

Veamos ahora cómo manejar los datos que faltan.

9.9. Manejo de datos faltantes

Ya hemos analizado este tema en el Capítulo 8, Ingesta y transformación de datos. Por favor, consulte ese capítulo para refrescar sus conocimientos sobre el manejo de datos duplicados.

Veamos ahora cómo manejar los datos que llegan tarde.

9.10. Manejo de datos que llegan tarde

Todavía no hemos cubierto este escenario, así que vamos a profundizar en el manejo de los datos que llegan tarde.

Un escenario de datos que llegan tarde puede considerarse en tres etapas diferentes de un data pipeline: durante la fase de ingestión de datos, la fase de transformación y la fase de servicio.

9.10.1. Tratamiento de los datos que llegan tarde en la fase de ingesta/transformación

Durante las fases de ingesta y transformación, las actividades suelen incluir la copia de datos en el data lake y la realización de transformaciones de datos mediante motores como Spark, Hive, etc. En estos escenarios, se pueden utilizar los dos métodos siguientes:

- Drop the data, si su aplicación puede soportar cierta cantidad de pérdida de datos. Esta es la opción más fácil. Puedes mantener un registro del último timestamp que ha sido procesado. Y si los nuevos datos tienen un timestamp más antiguo, puedes simplemente ignorar ese mensaje y seguir adelante.
- Vuelva a ejecutar el pipeline desde la pestaña de Monitorización del ADF, si su aplicación no puede manejar la pérdida de datos.

A continuación, vamos a ver cómo manejar los datos que llegan tarde en la etapa de servicio.

9.10.2. Manejo de los datos que llegan tarde en la fase de servicio

En la fase de servicio, el manejo de los datos se realiza normalmente a través de un esquema de estrella o de copo de nieve para escenarios OLAP. En estos casos, puede haber situaciones en las que una dimensión llega tarde (o un hecho puede llegar temprano). Veamos algunos métodos comunes para manejar estos escenarios:

- **Drop the message:** Al igual que en la etapa de ingesta/transformación, esta es la opción más sencilla, especialmente si los datos antiguos no aportan mucho valor.
- **Almacenar el mensaje y volver a intentarlo al cabo de un tiempo:** En esta técnica, almacene las filas de hechos que llegaron antes en una tabla de preparación (staging table) e intente insertar este hecho cuando en la siguiente iteración, esperando que la dimensión haya llegado para entonces. Repita este proceso un número predeterminado de veces antes de declarar el fracaso.
- **Inserte un registro ficticio en la tabla de dimensiones:** En esta técnica, si el registro de dimensión correspondiente no existe, basta con introducir un registro ficticio en su lugar. Tendrá que volver a visitar todos los registros ficticios y actualizarlos con valores reales una vez que lleguen los valores de la dimensión.

- Si tiene suficientes detalles sobre la dimensión, puede deducir la fila de la dimensión e insertar la nueva fila de la dimensión derivada con una nueva clave sustituta.

Estas son algunas de las formas de manejar los datos que llegan tarde. Veamos ahora cómo insertar datos.

9.11. Inserción de datos

La reinserción se refiere a las transacciones UPDATE o INSERT en los data stores. Los data stores pueden ser relacionales, de valor-clave, o cualquier otro almacén que soporte el concepto de actualización de filas o blobs.

ADF admite operaciones de upsert si el sink es un almacén basado en SQL. El único requisito adicional es que la actividad del sink debe estar precedida por una operación de Alter Row. A continuación se muestra una captura de pantalla de ejemplo de un sink de ADF con Allow upsert habilitado.

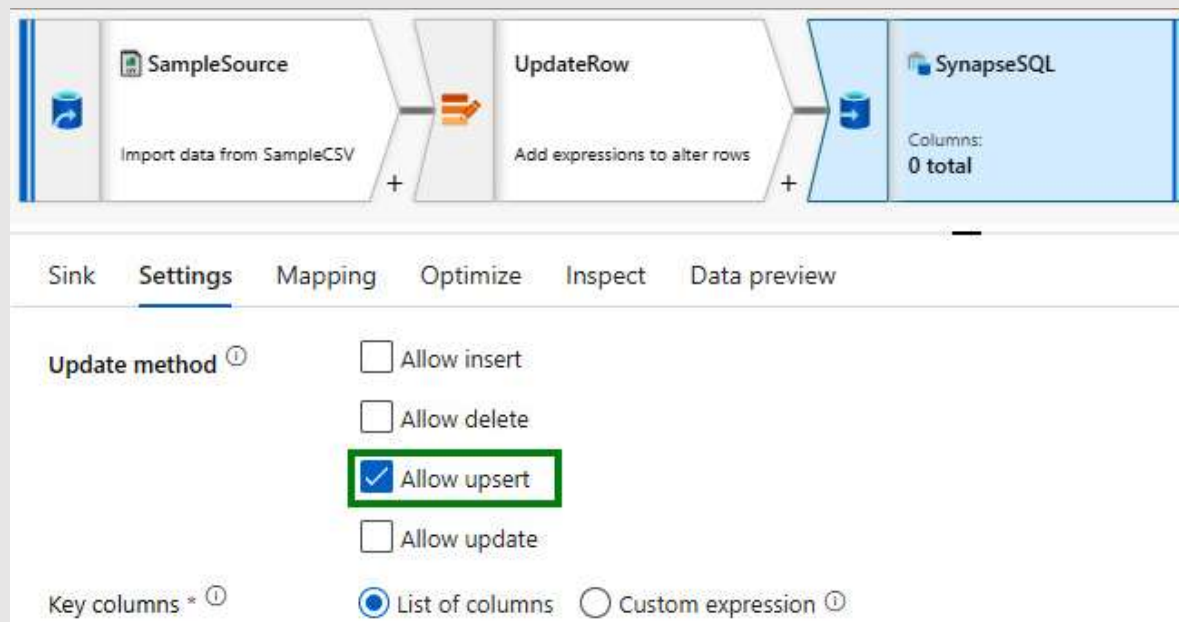


Figura 9.29 - Operación de upsert en el ADF

Una vez que haya guardado la configuración anterior, el ADF realizará automáticamente un upsert si ya existe una fila en el sumidero configurado. Veamos a continuación cómo retroceder a un estado anterior.

9.12. Regresar a un estado anterior

La regresión a un estado anterior o la vuelta a un estado estable es una técnica muy utilizada en bases de datos y escenarios OLTP. En los escenarios OLTP, las instrucciones de transformación se agrupan en una transacción y si alguna de las instrucciones falla o llega a un estado incoherente, toda la transacción retrocede. Aunque las bases de datos proporcionan esta funcionalidad, hoy en día no disponemos de este tipo de soporte en Azure Data Factory u Oozie (HDInsight). Tendremos que construir nuestras propias etapas de rollback dependiendo de la actividad. Veamos un ejemplo de cómo hacer un rollback de una actividad de copia de datos en ADF.

El ADF proporciona opciones para comprobar la consistencia y establecer límites para la tolerancia a fallos. Puede habilitarlas en las opciones de Configuración de una actividad de copia, como se muestra en la siguiente captura de pantalla.

The screenshot shows the 'Settings' tab of an Azure Data Factory copy activity. At the top, there's a warning: 'You will be charged # of used DIUs * copy duration * \$0.25/DIU-hour. Local currency and'. Below this, several settings are listed:

- Data integration unit**: Set to 'Auto' with an 'Edit' link.
- Degree of copy parallelism**: An empty input field with an 'Edit' link (checked).
- Data consistency verification**: A checkbox that is checked, highlighted with a green box.
- Fault tolerance**: A dropdown menu, also highlighted with a green box, showing a list of options: 'Select all', 'Skip incompatible rows', 'Skip missing files', 'Skip forbidden files', and 'Skip files with invalid names'. A '+ New' button is visible to the right of the dropdown.
- Enable logging**: A checkbox.
- Logging settings**: A section header with a dropdown arrow.
- Storage connection name**: A text input field with an asterisk and an info icon.
- Enable staging**: A checkbox.

Figura 9.30 - Habilitar la verificación de consistencia y la tolerancia a fallos en una actividad de copia del ADF

Si la actividad falla debido a las comprobaciones de consistencia o a la tolerancia a fallos más allá de un nivel, puede definir una actividad de seguimiento Eliminar en la ruta de fallo (enlace naranja) para limpiar completamente el directorio, como se muestra en la siguiente captura de pantalla.

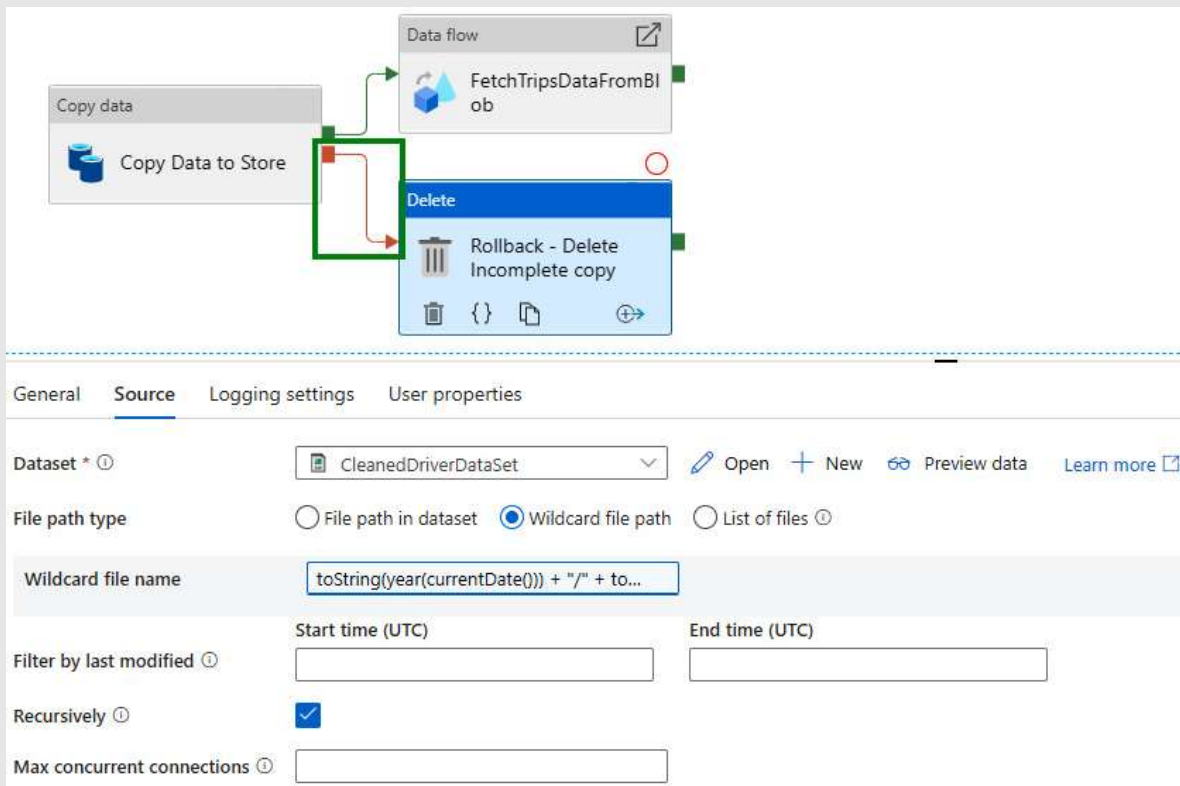


Figura 9.31 - Eliminación de una actividad de copia incompleta

NOTA

Puede seleccionar la casilla de verificación Recursivamente para procesar todos los archivos de la carpeta de entrada y sus subcarpetas recursivamente.

Tenga en cuenta que la técnica de eliminación de actividades puede ampliarse para otros escenarios como la inserción de datos en tablas, el reintento de transformaciones, etc. Las técnicas de reversión correspondientes en estos casos también serán diferentes, como el borrado de archivos, el borrado de filas en una tabla, etc.

Espero que te hayas hecho a la idea de cómo retroceder a una etapa anterior usando ADF. A continuación, nos centraremos en el servicio Azure Batch del que hemos hablado en la introducción de este capítulo.

9.13. Presentación de Azure Batch

Azure Batch es un servicio de Azure que se puede utilizar para realizar procesamiento paralelo por lotes a gran escala. Se suele utilizar para aplicaciones de computación de alto rendimiento como el análisis de imágenes, el renderizado 3D, la secuenciación del genoma, el reconocimiento óptico de caracteres, etc.

Azure Batch consta de tres componentes principales:

- **Gestión de recursos:** Se encarga de la gestión de los nodos (cosas como las VM y los contenedores Docker), el autoescalado, la gestión de las VM de baja prioridad y la gestión de las aplicaciones. Las aplicaciones son simplemente archivos ZIP de todos los ejecutables, bibliotecas y archivos de configuración necesarios para la ejecución del trabajo por lotes.
- **Gestión de procesos:** Se encarga de la programación de jobs y tareas, de reintentar los jobs fallidos, de imponer restricciones a los jobs, etc. Un job es una unidad lógica de trabajo. Un job se divide en tareas que pueden ejecutarse en paralelo en los nodos de la VM o del pool de contenedores.
- **Monitorización de recursos y procesos:** Se encarga de todos los aspectos de la monitorización. Hay varias opciones disponibles a través del portal de Azure, Application Insights y, por último, registros y métricas.

Como cualquier otro servicio de Azure, Batch puede ser completamente aprovisionado usando el portal de Azure. Aquí hay una captura de pantalla de ejemplo de una pantalla de Batch.

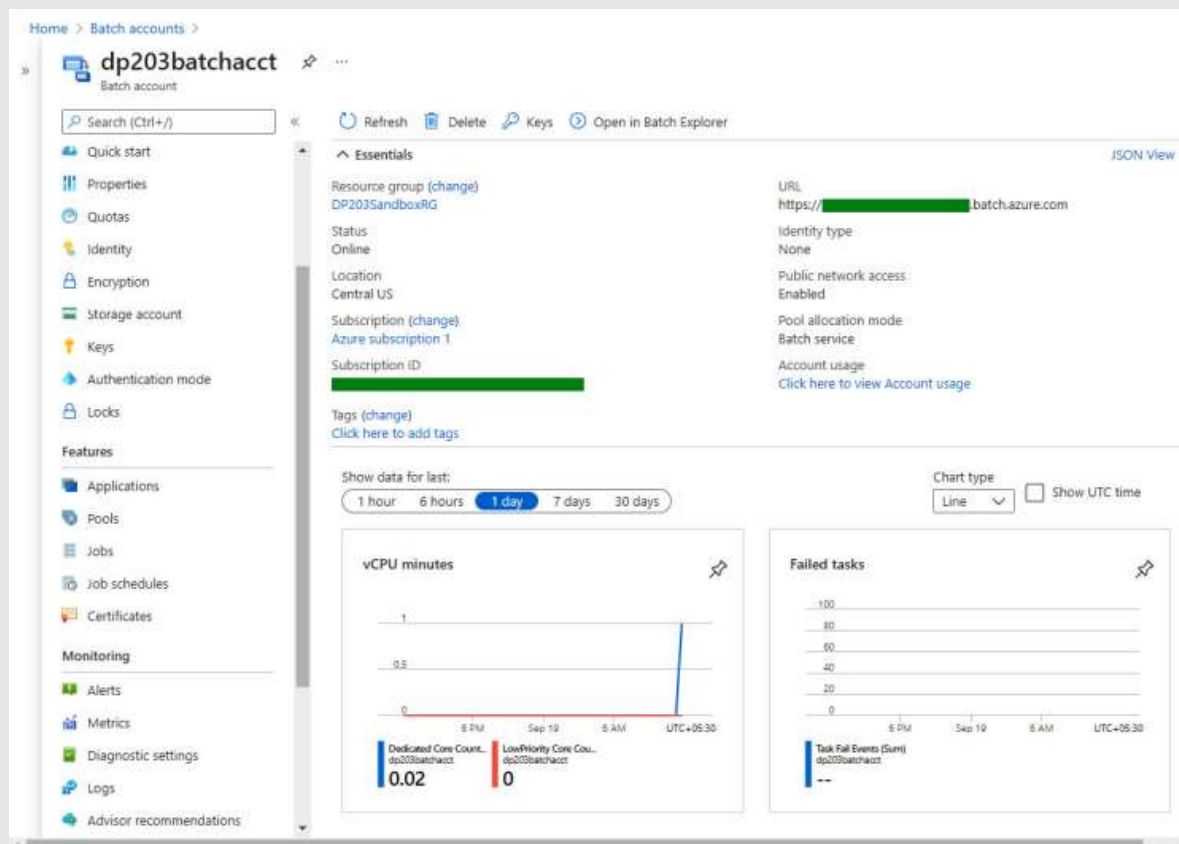


Figura 9.32 - Pantalla del portal de Azure Batch

Puede configurar y supervisar sus Batch jobs directamente desde el portal. Veamos a continuación cómo ejecutar un Batch job utilizando la CLI.

9.13.1. Ejecución de un Azure Batch job de ejemplo

Azure Batch proporciona múltiples enfoques para ejecutar jobs. Podemos utilizar el portal de Azure, Azure CLI, e incluso interfaces programáticas a través de .NET y PowerShell. Vamos a ver un ejemplo utilizando Azure CLI para este caso, ya que hemos explorado bastante el portal de Azure.

En este ejemplo, aprenderemos a crear una cuenta Azure Batch y a configurar un pool de VMs para ejecutar el job. A continuación, aprenderemos a ejecutar una aplicación en el pool y a descargar los resultados. Tendrás que reemplazar las opciones resaltadas dentro de <> con tus propias entradas. Puedes crear un ResourceGroup temporal desde el portal de Azure y utilizarlo en estos ejemplos. Para todas las demás entradas, como BatchAccountName, BatchPoolName, etc., puedes simplemente especificar tus propios nombres:

1. Cree una cuenta por lotes como se muestra:

```
az batch account create -g <ResourceGroup> -n <BatchAccountName> -l centralus
```

2. Cree una cuenta de almacenamiento como se muestra:

```
az storage account create -g <ResourceGroup> -n <BatchStoreAcct> -l centralus --sku Standard_LRS
```

3. Ahora, vincule la cuenta de almacenamiento a la cuenta Batch:

```
az batch account set -g <ResourceGroup> -n <BatchAccountName> --storage-account  
<BatchStoreAcct>
```

4. A continuación, creamos un pool con VMs de Ubuntu para ejecutar nuestra aplicación Batch. Esta operación tarda unos minutos:

```
az batch pool create \  
  --id <ResourceGroup> --vm-size Standard_A1_v2 \  
  --target-dedicated-nodes 2 \  
  --image canonical:ubuntuserver:16.04-LTS \  
  --node-agent-sku-id "batch.node.ubuntu 16.04"
```

5. Puede comprobar el estado de la creación del pool como se muestra:

```
az batch pool show --pool-id <BatchPoolName> \N -.  
  --query "allocationState"
```

6. A continuación, cree una aplicación que deba ser ejecutada por el Batch job:

```
az batch application create --resource-group <ResourceGroup> --name <BatchAccountName> --application-name sampleapp1
```

7. A continuación, cree un job:

```
az batch job create \
  --id <BatchJobName> \
  --pool-id <BatchPoolName>
```

8. Cree las tareas bajo el job. Las tareas comenzarán a ejecutarse tan pronto como las cree:

```
for i in {1..4}
do
  az batch task create \
    --task-id sampletask$i \
    --job-id <BatchJobName> \
    --command-line "/bin/bash -c 'printenv; sleep 30s'"
done
```

9. Supervise los jobs como se muestra:

```
az batch task show \
  --job-id <BatchJobName> \
  --task-id <BatchTaskName>
```

10. Descargue los resultados como se muestra:

```
az batch task file download \
  --job-id <BatchJobName> \
  --task-id <BatchTaskName> \
  --file-path stdout.txt \
  --destination ./stdout.txt
```

11. Finalmente, puede eliminar cada una de las entidades como se muestra:

```
az batch job delete --job-id <BatchJobName>
az batch task delete --job-id <BatchJobName> --task-id <BatchTaskName>
az batch pool delete --pool-id <BatchPoolName>
```

Este ejemplo debería haberle dado una imagen del ciclo de vida completo de un job de Azure Batch. Puedes aprender más sobre Azure Batch aquí: <https://docs.microsoft.com/en-us/azure/batch/batch-technical-overview>.

Ahora que conoces los fundamentos de Azure Batch, vamos a ver a continuación cómo configurar el tamaño de los lotes.

9.14. Configurar el batch size

Para configurar el tamaño del lote, exploraremos cómo determinar el tamaño del lote en Azure Batch. El tamaño del lote se refiere tanto al tamaño de los Batch pools como al tamaño de las VMs en esos pools. Las siguientes directrices son lo suficientemente genéricas como para poder aplicarlas también a otros servicios como Spark y Hive.

Estos son algunos de los puntos a tener en cuenta a la hora de decidir el tamaño del lote:

- **Requisitos de la aplicación:** En función de si la aplicación hace un uso intensivo de la CPU, de la memoria, del almacenamiento o de la red, tendrá que elegir los tipos de VMs y los tamaños adecuados. Puedes encontrar todos los tamaños de VM soportados usando el siguiente comando Azure CLI (aquí, centralus es un ejemplo):

```
az batch location list-skus -location centralus
```

- **Perfil de datos:** Si sabes cómo están repartidos tus datos de entrada, te ayudará a decidir los tamaños de VM que serán necesarios. Tendremos que planificar la mayor cantidad de datos que serán procesados por cada una de las VMs.
- **El número de tareas que se pueden ejecutar por nodo (VM):** Si las tareas no necesitan una VM entera, será beneficioso ejecutar múltiples tareas dentro de las mismas VMs.
- **Diferentes pools para diferentes batch loads:** Si tiene diferentes tipos de Batch jobs que necesitan diferentes capacidades de VM, será eficiente tener diferentes pools para los diferentes Batch jobs.
- **Disponibilidad de VM por región:** No todos los tipos de VM estarán disponibles en todas las regiones. Por lo tanto, debe tener en cuenta las geolocalizaciones al planificar la creación de su Batch pool.
- **Cuotas de VM por cuenta de Batch:** Aparte de la disponibilidad de VM por región, suele haber limitaciones de cuota en sus cuentas de Batch y suscripciones de Azure. Algunos de estos límites de cuota son límites blandos y pueden aumentarse elevando una solicitud de soporte. Pero con el tiempo, llegará a los límites duros. Por lo tanto, planifique el tamaño de su Batch pool en función de sus limitaciones de cuota. Puede verificar su cuota en la pestaña Cuotas de las cuentas de Batch. Aquí hay una captura de pantalla de ejemplo de esa página.

Home > dp203batchacct

dp203batchacct | Quotas ↗ ... ×

Batch account

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Settings
 - Quick start
 - Properties
 - Quotas**
 - Identity
 - Encryption
 - Storage account
 - Keys
 - Authentication mode
 - Locks
- Features
 - Applications
 - Pools
 - Jobs

Request quota increase

Active jobs and schedules ⓘ

100

Pools ⓘ

20

Batch accounts per region per subscription ⓘ

1

Low-priority vCPUs ⓘ

10

⚠ Both the total dedicated vCPU quota and the dedicated vCPU per VM Series quotas are enforced.

Total dedicated vCPUs ⓘ

10

Dedicated vCPUs per VM Series

VM Series ↑↓	Quota ↑↓
A Series	0
A Series - compute intensive	0
Av2 Series	10
Basic A Series	0
D Series	0

Figura 9.33 - Cuotas de VM por lotes

Puede obtener más información sobre las opciones de Batch VM aquí: <https://docs.microsoft.com/en-us/azure/batch/batch-pool-vm-sizes>.

Veamos a continuación cómo escalar recursos en Azure Batch y otros servicios de procesamiento por lotes como Spark y Synapse SQL.

9.15. Escalado de recursos

El escalado se refiere al proceso de aumentar o disminuir los recursos de cómputo, almacenamiento o red para mejorar el rendimiento de los jobs o reducir los gastos. Hay dos tipos de escalado: Manual y Automático. Como puede ser obvio, con el escalado manual, decidimos el tamaño de antemano. Con el escalado automático, el servicio decide dinámicamente el tamaño de los recursos en función de varios factores, como la carga del clúster, el coste de ejecución del clúster, las limitaciones de tiempo, etc.

Exploremos las opciones de escalado disponibles en Azure Batch y luego echemos un vistazo rápido a las opciones disponibles en Spark y SQL también.

9.15.1. Azure Batch

Azure Batch ofrece una de las opciones de autoescalado más flexibles. Te permite especificar tu propia fórmula de autoescalado. Azure Batch utilizará entonces su fórmula para decidir cuántos recursos aumentar o reducir.

Se puede escribir una fórmula de escalado basada en lo siguiente

- **Métricas de tiempo:** Utilizando las estadísticas de la aplicación recogidas en intervalos de 5 minutos
- **Métricas de recursos:** Utilizando el uso de la CPU, la memoria y el ancho de banda de la red
- **Métricas de tareas:** Utilizando el número de tareas en cola o completadas

La propia pantalla de autoescalado de Azure Batch proporciona fórmulas de autoescalado de ejemplo que puedes adaptar y mejorar según tus necesidades. A continuación se muestra una pantalla de autoescala de ejemplo con la fórmula:

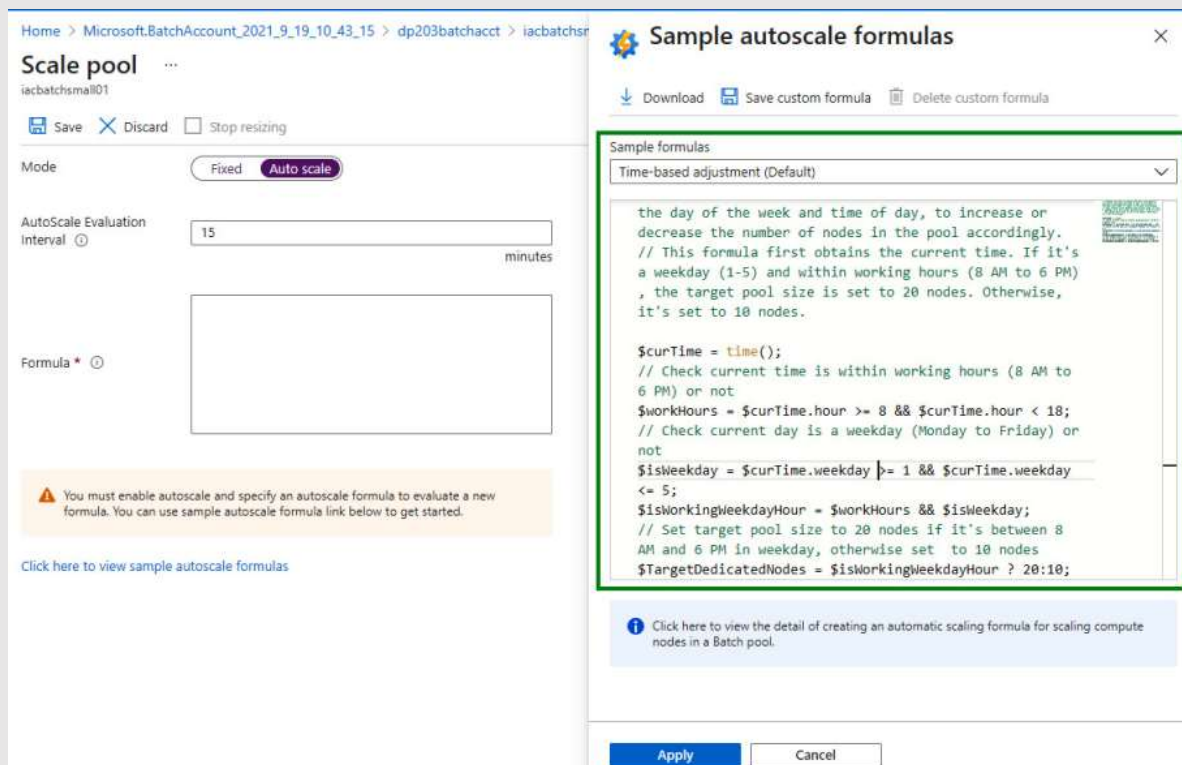


Figura 9.34 - Pantalla de fórmula de autoescalado por lotes

El ejemplo de la captura de pantalla anterior está estableciendo el número de nodos en función de la hora del día. Puedes encontrar más ejemplos de este tipo para el escalado de Azure Batch aquí: <https://docs.microsoft.com/en-us/azure/batch/batch-automatic-scaling>.

Veamos a continuación cómo podemos configurar las opciones de escalado en Spark y Synapse SQL Dedicated pools.

9.15.2. Azure Databricks

En Azure Databricks, mientras se crea el cluster, se puede seleccionar Enable Autoscaling y especificar el número de Min Workers y Max Workers. El clúster escalará automáticamente entre estos dos números en función de la carga. A diferencia de Azure Batch, aquí no tiene la flexibilidad de proporcionar su propia fórmula de escalado.

Create Cluster

New Cluster

Cancel Create Cluster

DBU / hour: 2.25 - 6.75

2-8 Workers: 28-112 GB Mem
1 Driver: 14 GB Memory, 4 Cores

Cluster Name: ADBSmallCluster

Cluster Mode: Standard

Databricks Runtime Version: Runtime: 8.3 (Scala 2.12, Spark 3.1.1)

Note: Databricks Runtime 8.x uses Delta Lake as the default table format. Learn more

Autopilot Options

☒ Enable autoscaling

☒ Terminate after 120 minutes of inactivity

Worker Type: Standard_DS3_v2 (14 GB Memory, 4 Cores)

Min Workers: 2 Max Workers: 8

☐ Spot instances

New: Configure separate pools for workers and drivers for flexibility. Learn more

Driver Type: Same as worker (14 GB Memory, 4 Cores)

DBU / hour: 2.25 - 6.75

Standard_DS3_v2

Advanced Options

Figura 9.35 - Opción de autoescalado de Azure Databricks Spark

Si quieres tener un cluster de tamaño fijo, sólo tienes que desmarcar la opción Enable autoscaling y proporcionar el número exacto de workers.

Puede ahorrar en el coste de sus clústeres utilizando instancias Spot. Las instancias Spot son VMs de Azure no utilizadas que son ofrecidas por Azure a un coste más barato, pero sin garantías de disponibilidad. Si Azure necesita recuperar la capacidad, puede retirar las Spot VM con 30 segundos de antelación. Puede utilizar esta opción si sus jobs pueden soportar interrupciones, como trabajos por lotes muy grandes, trabajos de desarrollo/prueba, etc.

9.15.3. Synapse Spark

Al igual que Azure Databricks, Synapse Spark también ofrece la opción de autoescalado en la pantalla de creación del clúster. La siguiente captura de pantalla muestra la pantalla con la opción Autoscale.

Microsoft Azure | Synapse Analytics | iacsynapsewvs

New Apache Spark pool

Basics * Additional settings * Tags Review + create

Create an Synapse Analytics Apache Spark pool with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize.

Apache Spark pool details

Name your Apache Spark pool and choose its initial settings.

Apache Spark pool name *

Isolated compute * ☐ Enabled ☒ Disabled

Node size family *

Node size *

Autoscale * ☒ Enabled ☐ Disabled

Number of nodes *

Estimated price
[Loading estimated cost information...](#)

[Review + create](#) [Next: Additional settings >](#) [Cancel](#)

Figura 9.36 - Opción de autoescalado de Synapse Spark

A continuación, veamos la opción disponible para los Synapse SQL Dedicated pools.

9.15.4. Synapse SQL

Synapse SQL no provee la opción de auto-escala, pero provee la opción de elegir el nivel de rendimiento del cluster mientras se crea el cluster como se muestra en la siguiente captura de pantalla. Cuanto mayor sea el número, mejor será el rendimiento.

Microsoft Azure | Synapse Analytics | iacsynapsewvs

New dedicated SQL pool

Basics * Additional settings * Tags Review + create

Create a dedicated SQL pool with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults. [Learn more](#)

Dedicated SQL pool details

Name your dedicated SQL pool and choose its initial settings.

Dedicated SQL pool name *

Performance level

Estimated price
326.37 INR
[View pricing details](#)

[Review + create](#) [Next: Additional settings >](#) [Cancel](#)

Figura 9.37 - Configuración del rendimiento de Synapse SQL durante la creación del cluster

Ahora ya sabes cómo configurar el escalado en Azure Batch, Azure Databricks Spark, Synapse Spark y Synapse SQL.

Veamos ahora cómo configurar la batch retention.

9.16. Configuración de la batch retention

El tiempo de retención por defecto de las tareas en Azure Batch es de 7 días a no ser que se elimine o pierda el nodo de cómputo. Sin embargo, podemos establecer el tiempo de retención requerido mientras añadimos un job.

Aquí hay un ejemplo usando las REST APIs. El retentionTime debe establecerse en el cuerpo de la solicitud como se muestra:

```
POST account.region.batch.azure.com/jobs/jobId/tasks?api-version=2021-06-01.14.0
```

Examine el siguiente cuerpo de solicitud:

```
{
  "id": "jobId",
  "prioridad": 100,
  "jobManagerTask": {
    "id": "taskId",
    "commandLine": "test.exe",
    "constraints": {
      "retentionTime": "PT1H"
    }
  }
}
```

PT1H especifica 1 hora y utiliza el formato ISO_8601. Puede obtener más información sobre el formato aquí: https://en.wikipedia.org/wiki/ISO_8601#Durations.

Veamos ahora cómo diseñar y configurar el manejo de excepciones.

9.17. Diseño y configuración de la gestión de excepciones

Azure Batch proporciona códigos de error, registros y eventos de monitorización para identificar y manejar los errores. Una vez identificados los errores, podemos gestionarlos mediante programación a través de las API y el código .NET.

A continuación se muestran algunos ejemplos de códigos de error devueltos por Batch:

Error Code	Category	User Message
AutoScalingFormulaSyntaxError	BadRequest	The specified autoscaling formula has a syntax error.
CommandLaunchFailed	UserError	Failed to launch the specified command line.
NodeBeingRebooted	Conflict	The specified node is being rebooted.
DiskFull	ServerError	There is not enough disk space on the node that was selected to run the task.

Figura 9.38 - Ejemplos de códigos de error de Batch

Puede obtener la lista completa de códigos de error aquí: <https://docs.microsoft.com/en-us/rest/api/batchservice/batch-status-and-error-codes>.

A continuación, veamos algunos tipos de error comunes en Azure Batch.

9.17.1. Tipos de errores

Hay cuatro grupos comunes de errores:

- **Errores de aplicación:** Para los errores de aplicación, Azure Batch escribe la salida estándar y el error estándar en los archivos stdout.txt y stderr.txt en el directorio de la tarea en el nodo de computación. Podemos analizar estos archivos para identificar el problema y tomar medidas correctivas.
- **Errores en las tareas:** Una tarea se considera fallida si devuelve un código de salida distinto de cero. El fallo puede deberse a múltiples razones, como errores de preprocesamiento, errores de carga de archivos o errores de línea de comandos. En todos estos casos, se establecerán los códigos de error correspondientes. Y podemos configurar programáticamente Batch para que reintente las tareas hasta un número determinado de veces. Para los errores de las tareas, debemos comprobar la propiedad executionInfo. Esta propiedad contiene detalles de los errores como el resultado, exitCode, failureInfo, etc., que pueden ayudar a identificar los errores.
- **Errores de job:** Un job es una colección de tareas. De forma similar a los errores de las tareas, necesitamos comprobar la propiedad executionInfo para determinar la causa del error de los jobs.

- **Errores de archivos de salida:** Durante la carga de archivos, Batch escribe los registros en dos archivos, a saber, fileuploadout.txt y fileuploaderr.txt. Estos dos archivos deben ser revisados en caso de errores de carga de archivos de salida.

Veamos a continuación algunas de las acciones correctivas más comunes.

9.17.2. Acciones correctivas

Aunque la acción correctiva para cada tipo de error será única, las siguientes genéricas ayudan si los jobs se atascan durante mucho tiempo o hay problemas con los nodos:

Reiniciar el nodo:

API: POST {batchUrl}/pools/{poolId}/nodos/{nodeId}/reboot?api-version=2021-06-01.14.0

Reimage el nodo:

API: POST {batchUrl}/pools/{poolId}/nodos/{nodeId}/reimage?api-version=2021-06-01.14.0

Eliminar el nodo del pool:

POST {batchUrl}/pools/{poolId}/removenodes?api-version=2021-06-01.14.0

Desactivar la programación de jobs en ese pool:

API: POST {batchUrl}/pools/{poolId}/nodos/{nodeId}/disablescheduling?api-version=2021-06-01.14.0

Veamos a continuación algunos de los requisitos de seguridad y cumplimiento para los data pipelines en general. Los conceptos e ideas que aquí se discuten también se aplican a otras tecnologías, como los pipelines basados en Spark y Hive.

9.18. Gestión de los requisitos de seguridad y cumplimiento de la normativa

La seguridad y el cumplimiento siempre serán uno de los requisitos principales de cualquier sistema basado en la nube. Azure proporciona un servicio llamado Azure Policy para habilitar y aplicar políticas de cumplimiento y seguridad en cualquiera de los servicios de Azure. En nuestro caso, podría ser Azure Synapse, Azure Batch, VMs, VNets, etc. Azure Policy ayuda a hacer cumplir las políticas y las acciones correctivas a escala.

Azure Policy contiene reglas de política predefinidas denominadas built-ins. Por ejemplo, una de las reglas podría ser Permitir que sólo se creen VMs de un tipo determinado en mi suscripción. Cuando se aplica esta política, si alguien intenta crear una VM de un SKU diferente, la política fallará la creación de la VM. Se mostrará un error diciendo No permitido por la política en la pantalla de validación para la creación de recursos.

Azure Policy tiene una enorme lista de políticas predeterminadas y acciones correctivas para diferentes casos de uso de conformidad. Puede elegir las políticas que son relevantes para su aplicación y aplicarlas. A continuación se muestra una pantalla de ejemplo de las políticas de Azure:

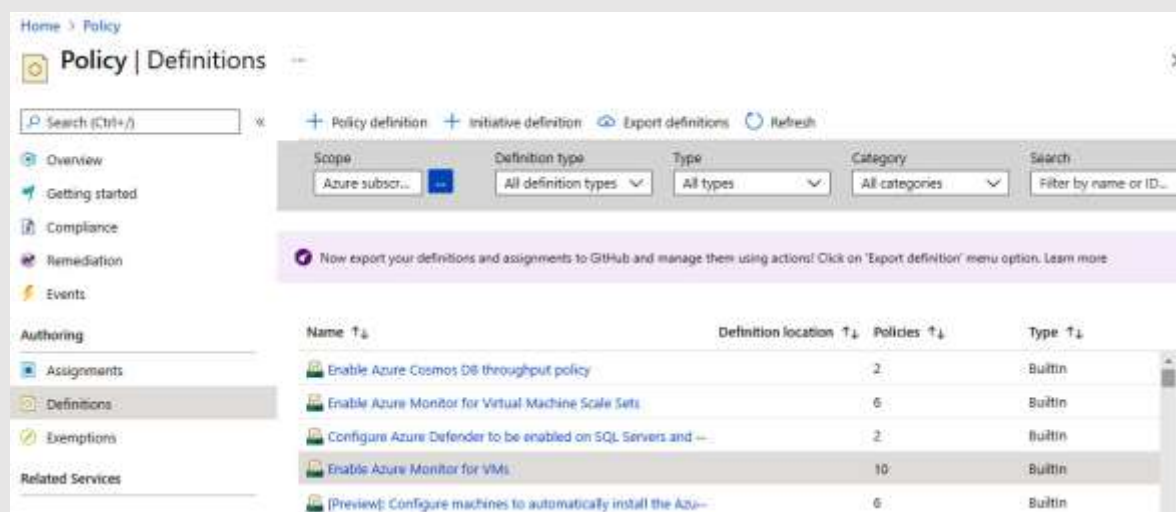


Figura 9.39 - Ejemplos de políticas de Azure

Esta es una captura de pantalla de la pantalla de políticas de Azure donde podemos aplicar las políticas de Azure y los métodos de remediación.

Home > Policy > Enable Azure Monitor for VMs >

Enable Azure Monitor for VMs

Assign initiative

Basics Parameters **Remediation** Non-compliance messages Review + create

Scope
Scope [Learn more about setting the scope](#) *

Exclusions
Optionally select resources to exclude from the policy assignment.

Basics
Initiative definition
Enable Azure Monitor for VMs

Assignment name * ⓘ
Enable Azure Monitor for VMs

Description

Policy enforcement ⓘ
Enabled Disabled

Review + create Cancel Previous Next

Figura 9.40 - Aplicar políticas de Azure

Por ejemplo, para cumplir con la ley HIPAA, uno de los requisitos es asegurar que el registro de auditoría esté habilitado.

Por lo tanto, si alguno de sus servicios de Azure no tiene activado el registro de auditoría, Azure Policy marcará un error de cumplimiento. Incluso puede configurar la herramienta para que desactive el servicio si no es conforme.

Puede obtener más información sobre el servicio Azure Policy aquí: <https://docs.microsoft.com/en-us/azure/governance/policy/overview>.

Veamos a continuación otra técnica recomendada para mejorar la seguridad de los sistemas de procesamiento por lotes, el Azure Security Benchmark.

9.18.1. El Azure Security Benchmark

Además de Azure Policy, Azure también proporciona un conjunto de recomendaciones para mejorar la seguridad de los datos, servicios y cargas de trabajo basados en la nube. Este conjunto de mejores prácticas se denomina Azure Security Benchmark. Este punto de referencia cubre una amplia gama de áreas, como la seguridad de la red, la gestión de identidades, el registro, la protección de datos, etc. Puede validar su servicio con este punto de referencia para asegurarse de que sigue el estándar del sector.

Los ejemplos de reglas de referencia de seguridad de red (NS) son los siguientes:

- NS-1: Desplegar pool(s) de Azure Batch dentro de una red virtual.
- NS-2: Conectar redes privadas entre sí.
- NS-3: Establecer el acceso de la red privada a los servicios de Azure.
- NS-4: Proteger las aplicaciones y los servicios de los ataques de la red externa.

Los ejemplos de reglas de referencia de gestión de identidades son los siguientes:

- IM-1: Estandarizar Azure AD como sistema central de identidad y autenticación.
- IM-2: Gestionar las identidades de las aplicaciones de forma segura y automática.
- IM-3: Utilizar el inicio de sesión único (SSO) de Azure AD para el acceso a las aplicaciones, etc.

Puede encontrar la lista completa aquí: <https://docs.microsoft.com/en-us/security/benchmark/azure/overview>.

Veamos a continuación algunas de las mejores prácticas para el servicio Azure Batch.

9.18.2. Mejores prácticas para Azure Batch

A continuación, un resumen de algunas de las mejores prácticas para asegurar los sistemas de lotes derivadas de la documentación de Azure:

- **Utilice endpoints privados:** Puedes utilizar el servicio Azure Private Link para restringir el acceso a los servicios de procesamiento por lotes desde redes externas.
- **Crear pools en redes virtuales:** En el servicio Azure Batch, puede crear batch pools dentro de una red virtual para que los nodos puedan comunicarse de forma segura entre sí.
- **Cree pools sin direcciones IP públicas:** Para reducir las posibilidades de ser descubierto desde redes públicas.
- Limitar el acceso remoto a los nodos del pool configurando cortafuegos.
- Cifrar los datos en tránsito utilizando https://.
- Encriptar los datos del Batch en reposo utilizando claves secretas.

- Aplique las políticas de cumplimiento y seguridad descritas en las secciones anteriores a su instancia de Azure Batch.

Puede encontrar una lista exhaustiva aquí: <https://docs.microsoft.com/en-us/azure/batch/security-best-practices>.

Resumen

Espero que ahora tengas una buena idea sobre los batch pipelines y el servicio Azure Batch. Hemos aprendido a crear pipelines batch de principio a fin, profundizando en cada una de las etapas, como la ingesta, las transformaciones, las integraciones de BI, etc. Luego conocimos un nuevo servicio llamado Azure Batch y aprendimos sobre la retención de lotes, el manejo de errores, el manejo de la autoescala, la construcción de data pipelines usando Batch, y más. También aprendimos sobre algunos de los aspectos críticos de seguridad y cumplimiento. Es mucha información para masticar. Intenta ojear el capítulo una vez más si tienes alguna duda.

A continuación nos centraremos en cómo diseñar y desarrollar una solución de procesamiento de streaming.

