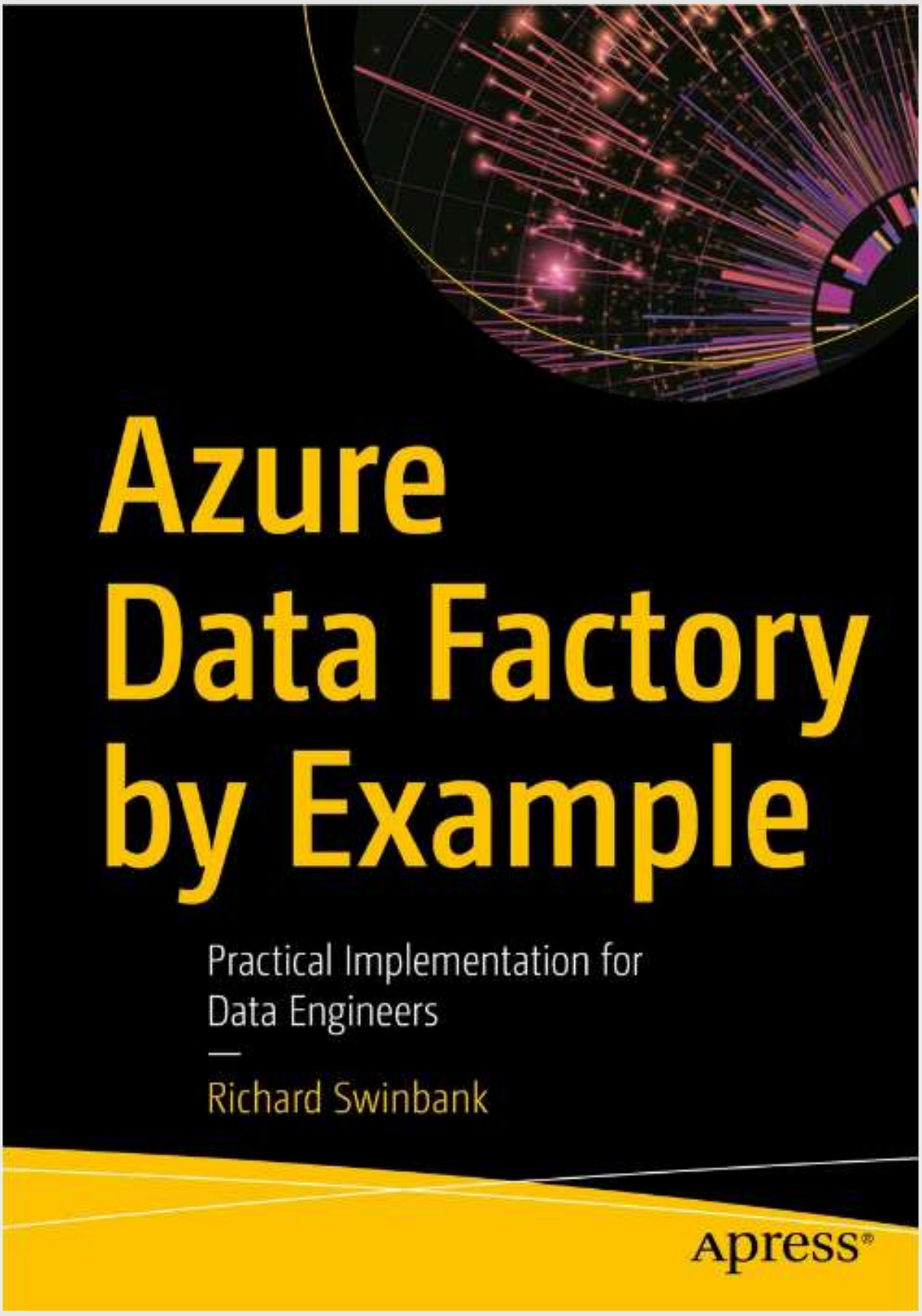

Azure
Data Factory
by Example
Practical Implementation for
Data Engineers
—
Richard Swinbank
Apress®

Contenido

6. Control del flujo 3

6.1. Crear un Per-File Pipeline 3

6.2. Utilizar condiciones de dependencia de la actividad 11

6.2.1. Explorar las interacciones de la condición de dependencia 15

6.2.2. Entender el resultado del pipeline 20

6.3. Levantamiento de errores 24

6.4. Utilizar actividades condicionales 25

6.4.1. Desviar filas de error 25

6.4.2. Cargas de filas de error 29

6.4.3. Comprender la actividad Switch 32

6.5. Utilizar actividades de iteración 34

6.5.1. Utilizar la actividad Get Metadata 34

6.5.2. Utilizar la actividad ForEach 35

6.5.3. Entender la actividad Until 39

Revisión del capítulo 41

Conceptos clave 41

6. Control del flujo

En el Capítulo 4, usted comenzó a construir pipelines de ADF que contenían más de una actividad, controlando su orden de ejecución mediante dependencias de actividad configuradas entre ellas. Las dependencias de actividad son una de las herramientas disponibles en ADF para controlar el flujo de ejecución de un pipeline.

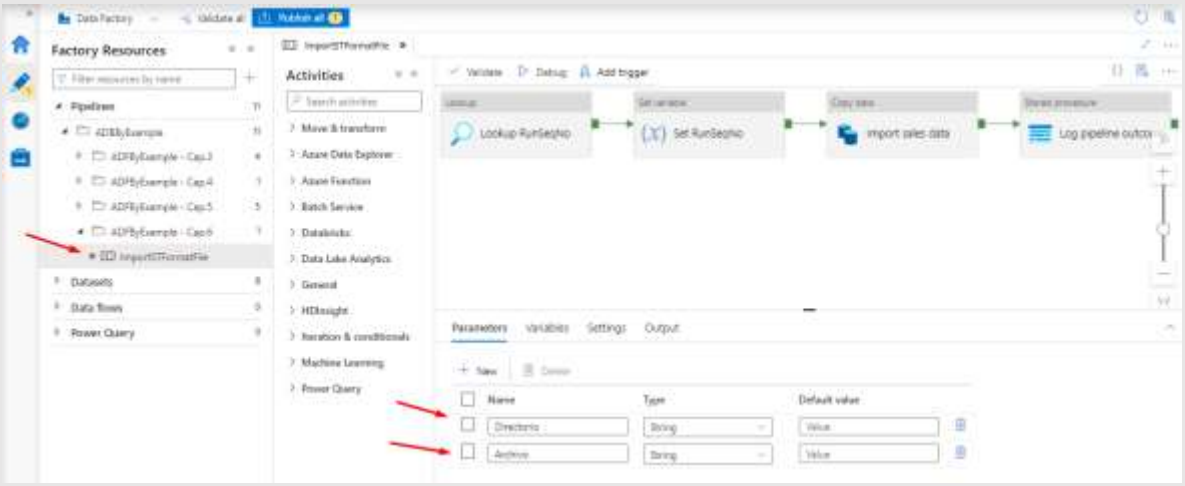
En este capítulo, desarrollará control flows más sofisticados utilizando dependencias de actividad con diferentes condiciones de dependencia. También encontrará otras actividades de ADF con el propósito específico de controlar el flujo. Para empezar, creará un nuevo pipeline para utilizarlo con algunas de estas actividades.

6.1. Crear un Per-File Pipeline

Los pipelines que construyó en los capítulos anteriores explotaron la capacidad de la actividad Copiar datos para describir fuentes de varios archivos utilizando comodines de directorio y nombre de archivo. Los pipelines de este capítulo introducirán controles de carga de archivos más finos, haciendo uso de un pipeline que carga un solo archivo especificado.

1. Cree un clon del pipeline "ImportSTFormatFolder" del capítulo 5 y muévalo a una nueva carpeta de conductos del "Capítulo 6". Nombra el nuevo pipeline "ImportSTFormatFile".

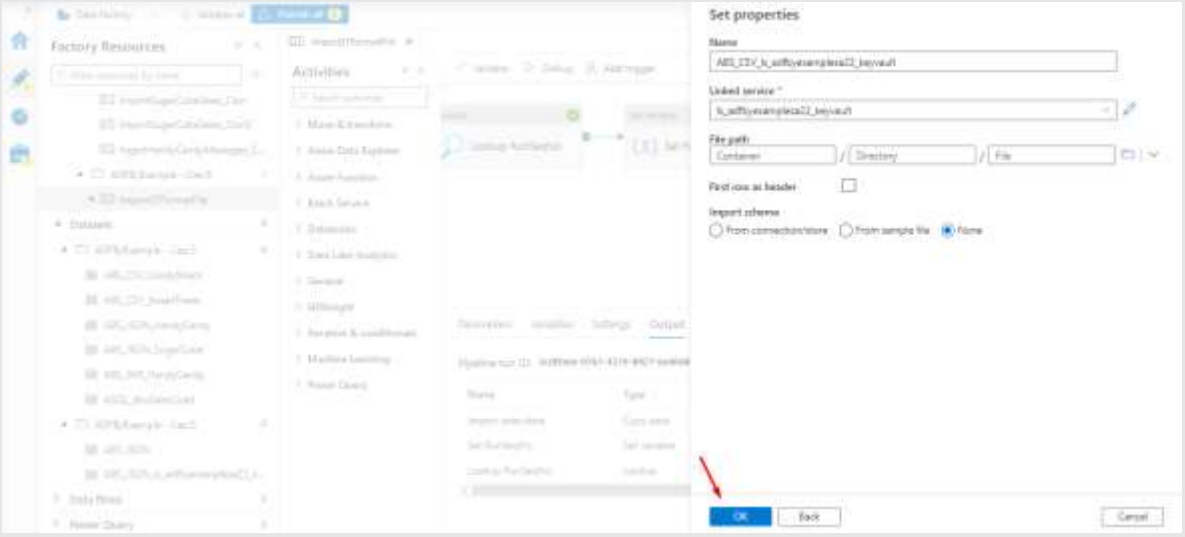
2. Abra la pestaña de configuración de parámetros del nuevo pipeline. Cambie el nombre de los dos parámetros existentes a "Directorio" y "Archivo" y elimine cualquier valor predeterminado.

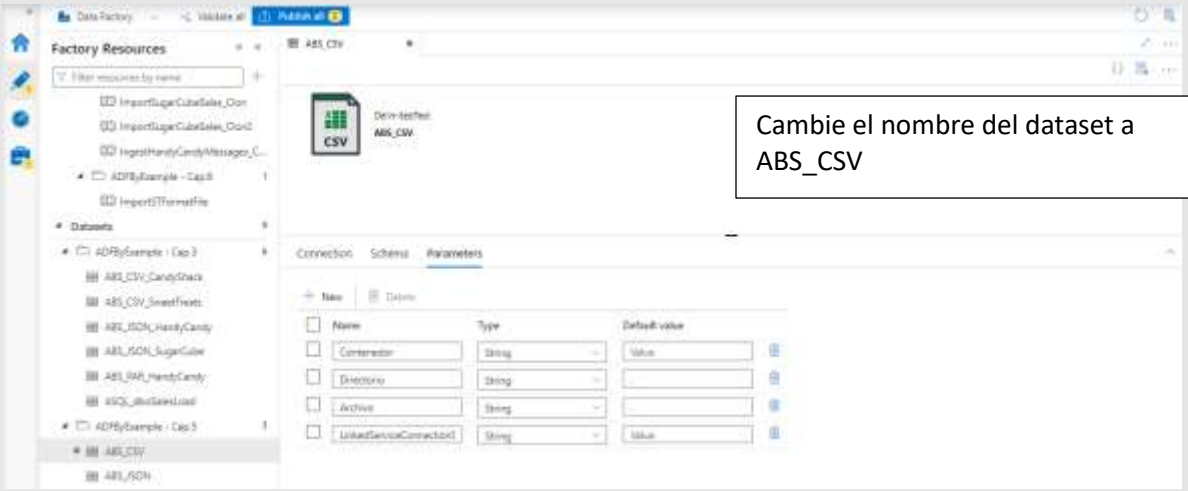


3. Seleccione la actividad de copia de datos "Import sales data", descarte cualquier solicitud de valores de parámetros y abra su pestaña Origen. Inmediatamente verá mensajes de error relacionados con los nombres de los parámetros originales (que ya no existen) - elimine las dos expresiones no válidas. Cambie el File path type a "File path in dataset".

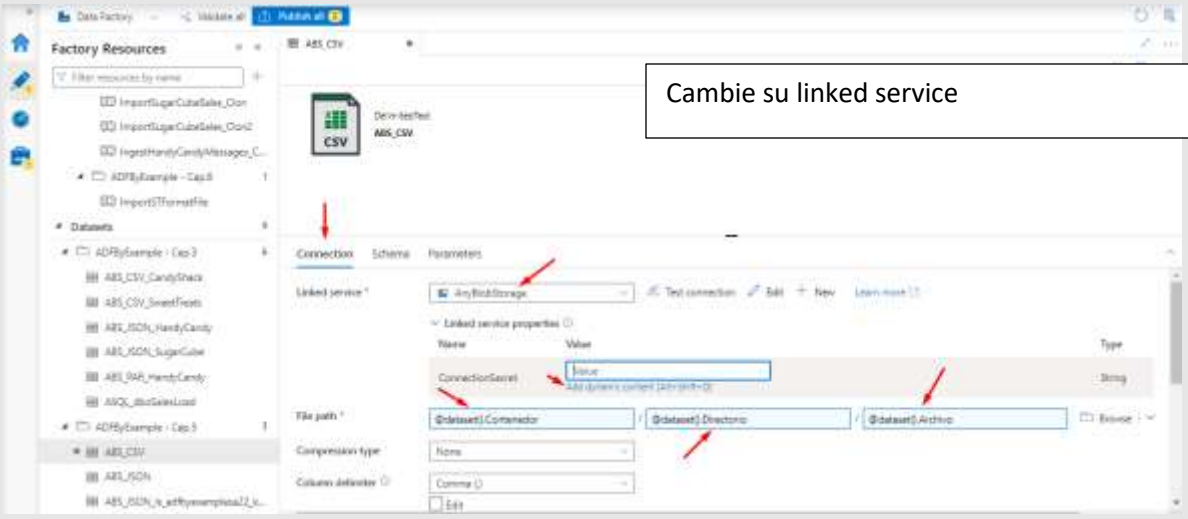
4. Cambie el dataset de origen de la actividad, seleccionando su dataset parametrizado "ABS_CSV" del capítulo 5. En la región de las propiedades del Dataset, asegúrese de que el valor del parámetro "Container" esté configurado como "sampledata", y luego actualice los valores de los parámetros del dataset "Directorio" y "Archivo" para utilizar los parámetros del pipeline con los mismos nombres. La Figura 6-1 muestra la configuración del Origen de la actividad Copiar datos.

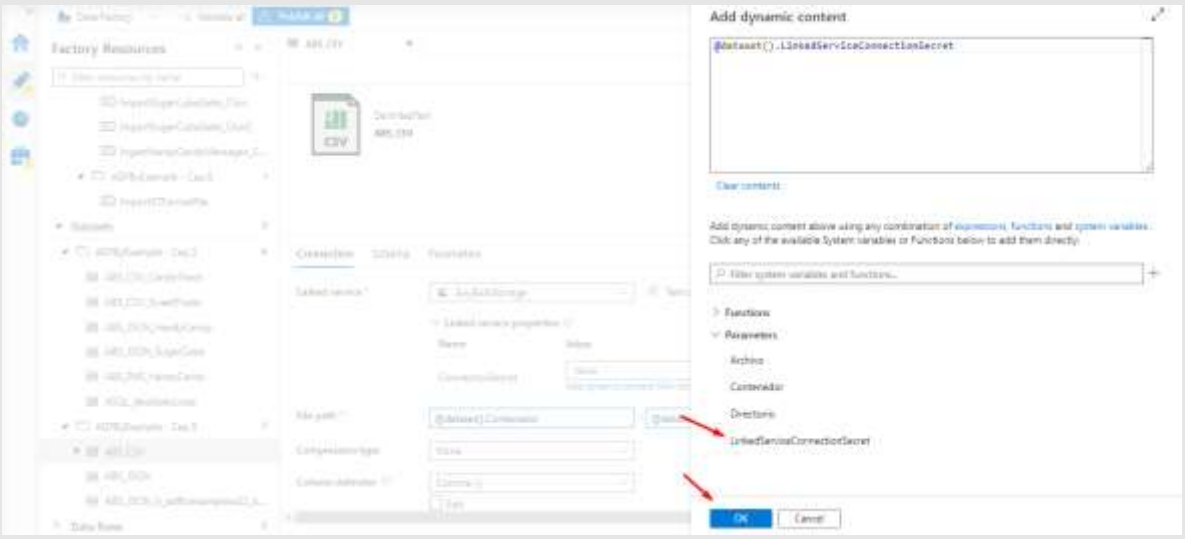
(Proceso creación del dataset ABS_CSV)

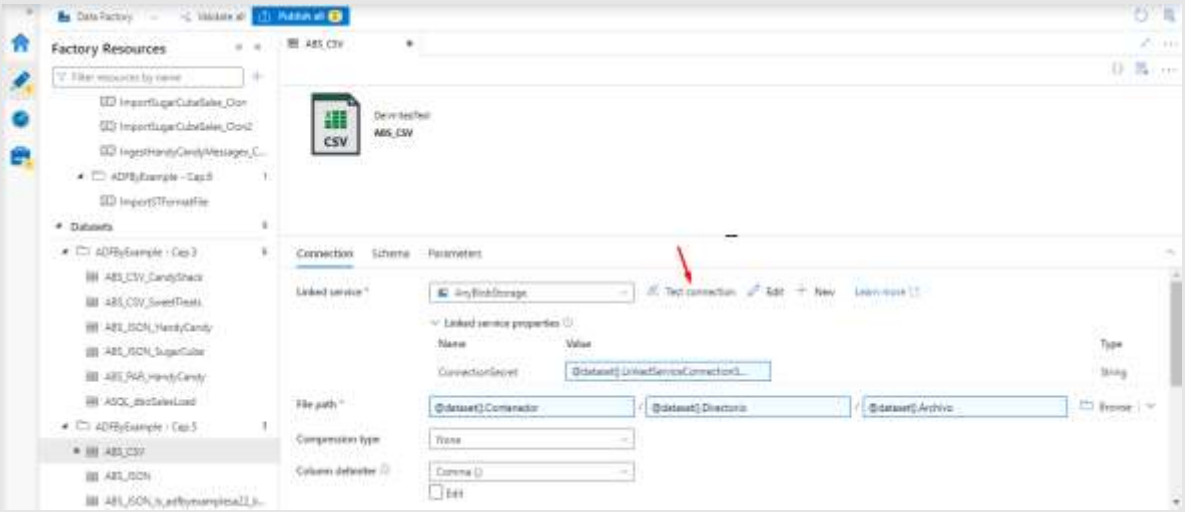


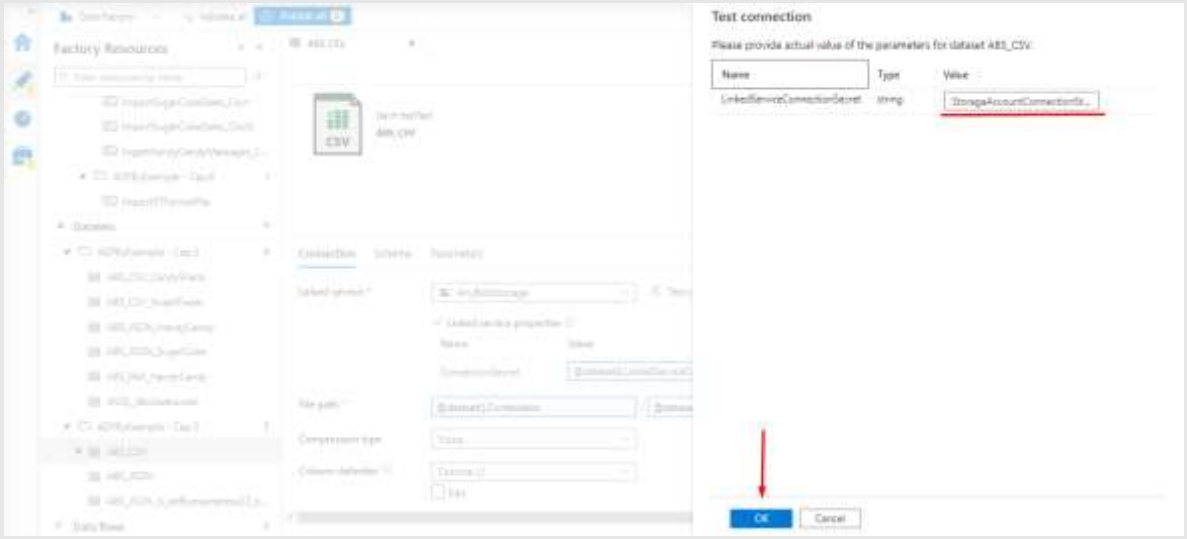


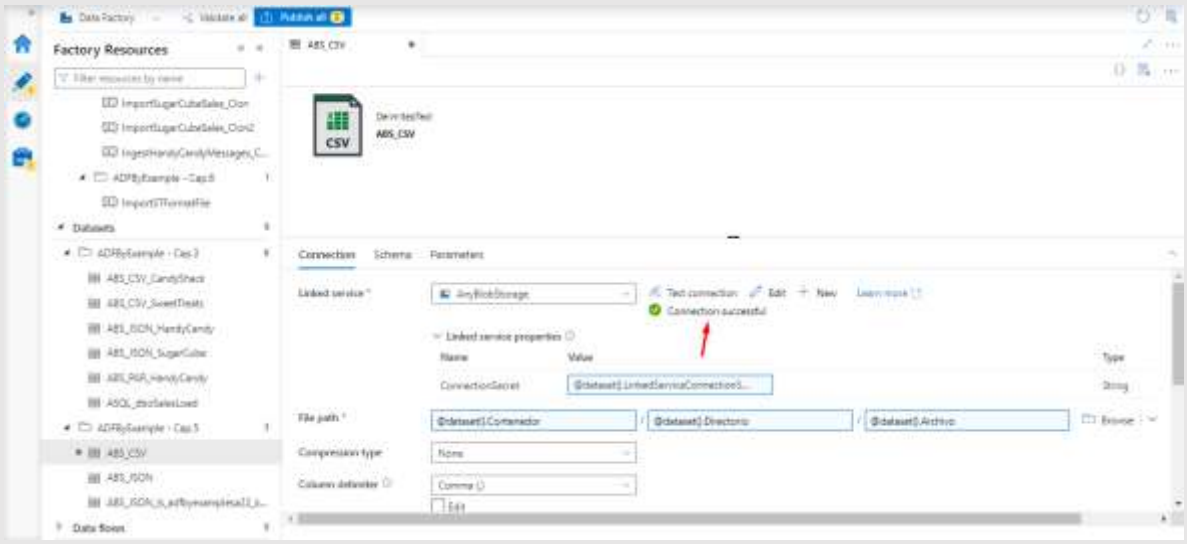
Cambie el nombre del dataset a ABS_CSV

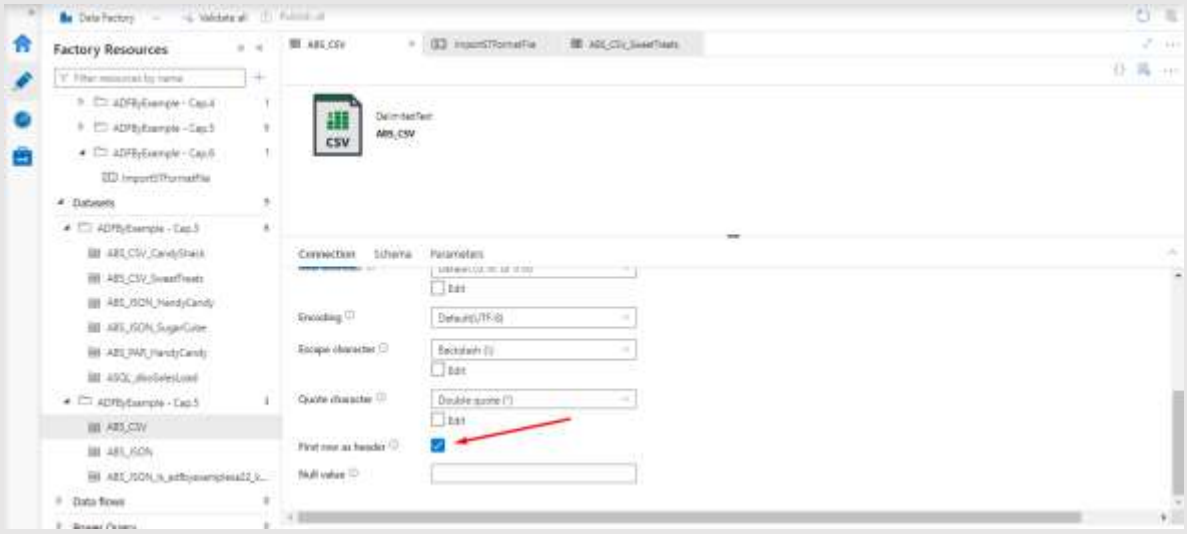

Cambie su linked service
Factory Resources
Connection
Schema
Parameters
Linked service *
Test connection
Edit
New
Learn more
Linked service properties
Name
Value
Type
ConnectionSecret
String
File path *
@dataset().Contenedor
@dataset().Directorio
@dataset().Archivo
Browse
Compression type
None
Column delimiter
Comma (,)
Edit

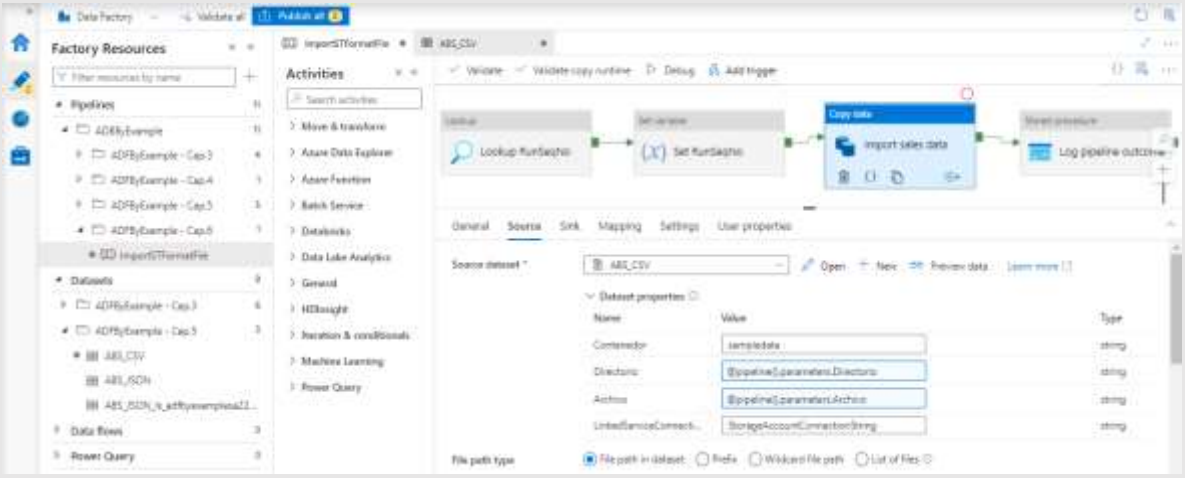

Add dynamic content
@dataset().LinkedServiceConnectionSecret
Clear contents
Add dynamic content above using any combination of expressions, functions and system variables.
Click any of the available System variables or Functions below to add them directly.
Filter system variables and functions...
Functions
Parameters
Archivo
Contenedor
Directorio
LinkedServiceConnectionSecret
OK
Cancel


Factory Resources
Connection
Schema
Parameters
Linked service *
Test connection
Edit
New
Learn more
Linked service properties
Name
Value
Type
ConnectionSecret
String
File path *
@dataset().Contenedor
@dataset().Directorio
@dataset().Archivo
Browse
Compression type
None
Column delimiter
Comma (,)
Edit

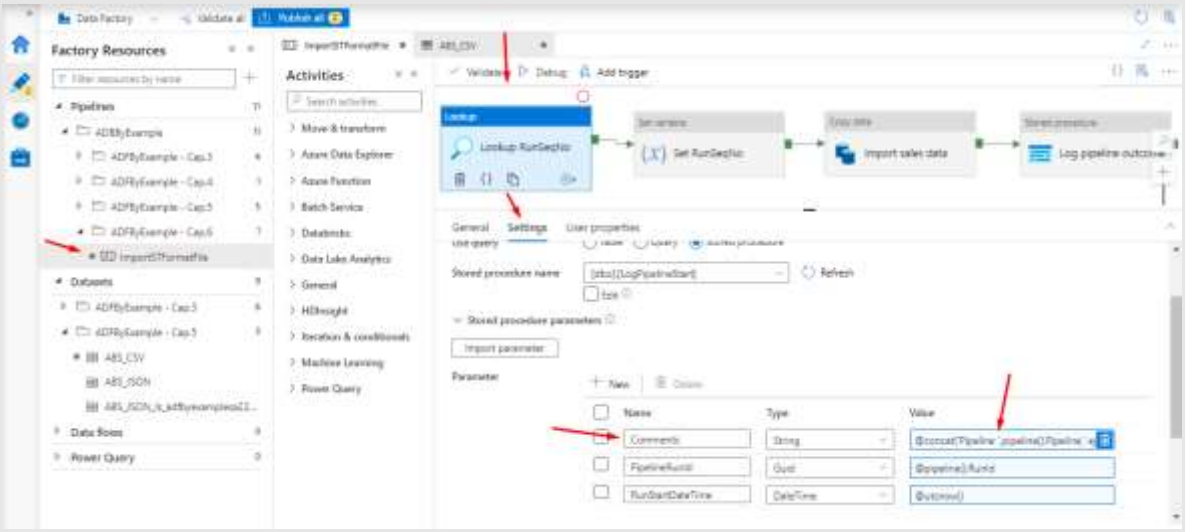

Test connection
Please provide actual value of the parameters for dataset ABS_CSV
Name
Type
Value
LinkedServiceConnectionSecret
String
OK
Cancel


Factory Resources
ABS_CSV
CSV
Connection
Schema
Parameters
Linked service *
Test connection
Edit
New
Connection successful
Linked service properties
Name
Value
Type
ConnectionSecret
String
File path *
Compression type
None
Column delimiter
Comma (,)
Edit


Factory Resources
ABS_CSV
CSV
Connection
Schema
Parameters
Encoding
Default(UTF-8)
Escape character
Backslash (\)
Quote character
Double quote (")
First row as header
Null value
Edit



5. En la pestaña Settings de la primera actividad del pipeline - la actividad Lookup denominada "Lookup RunSeqNo" - amplíe la expresión del parámetro "Comments" para incluir el archivo que se está cargando, por ejemplo, Pipeline @{pipeline().Pipeline} executed in @{pipeline().DataFactory} - loading file "@{pipeline().parameters.Directory}/@{pipeline().parameters.File}".



```
@concat('Pipeline ',pipeline().Pipeline,' ejecutado en ',pipeline().DataFactory,' cargado en ',pipeline().parameters.Directorio,'/',pipeline().parameters.Archivo)
```

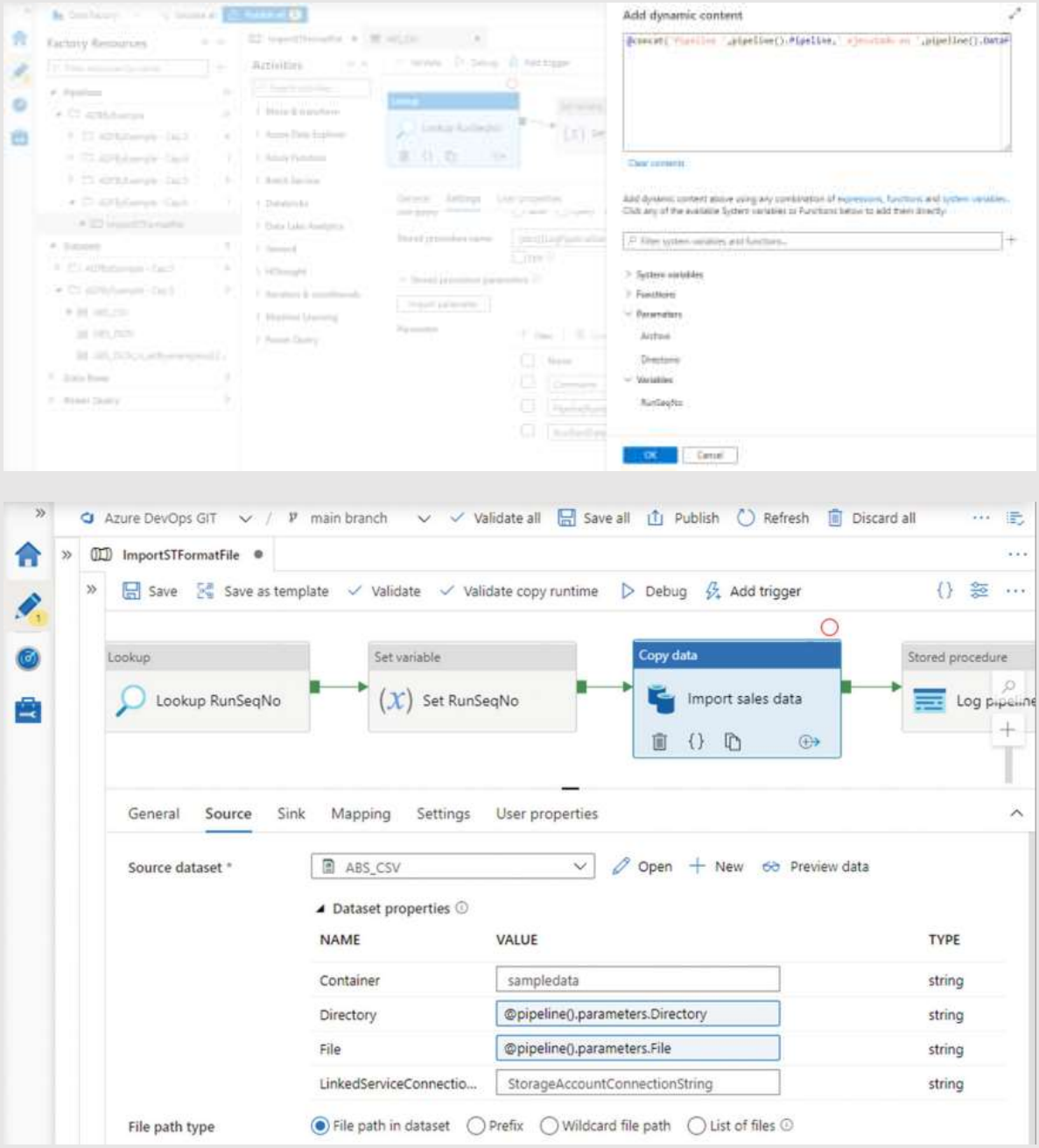
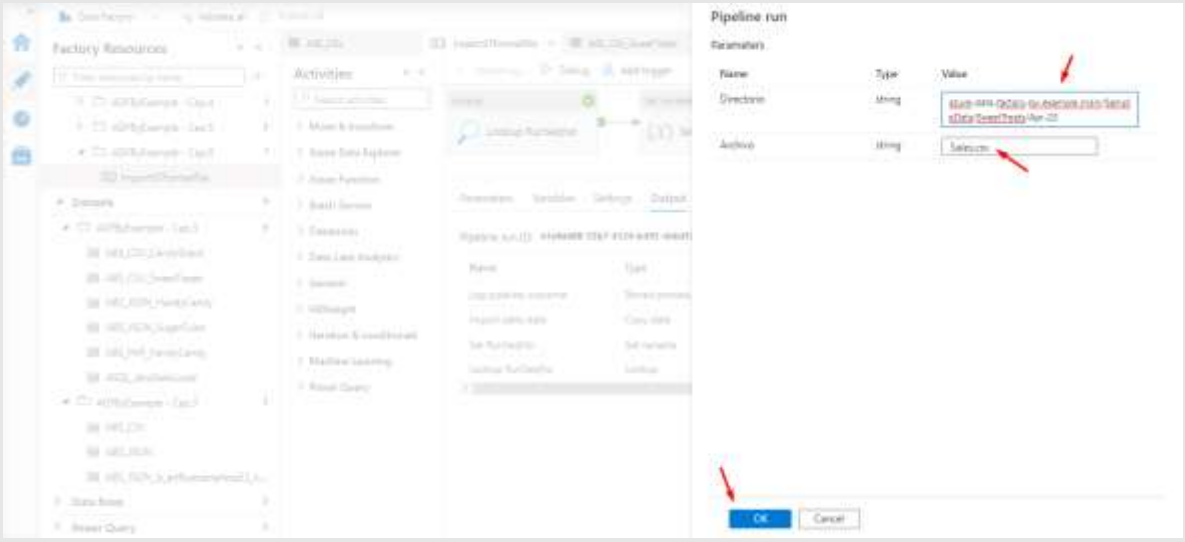
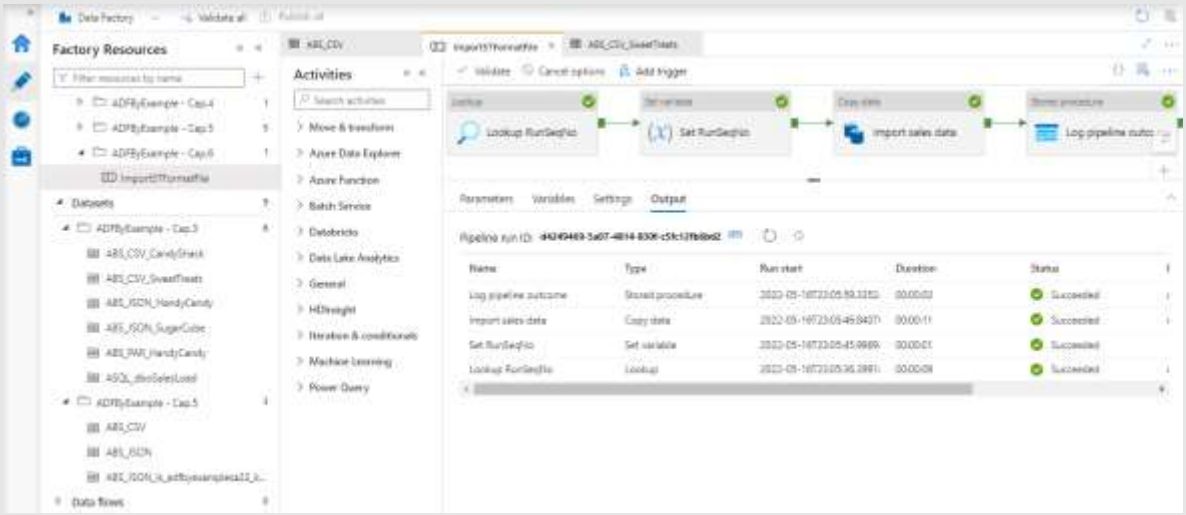
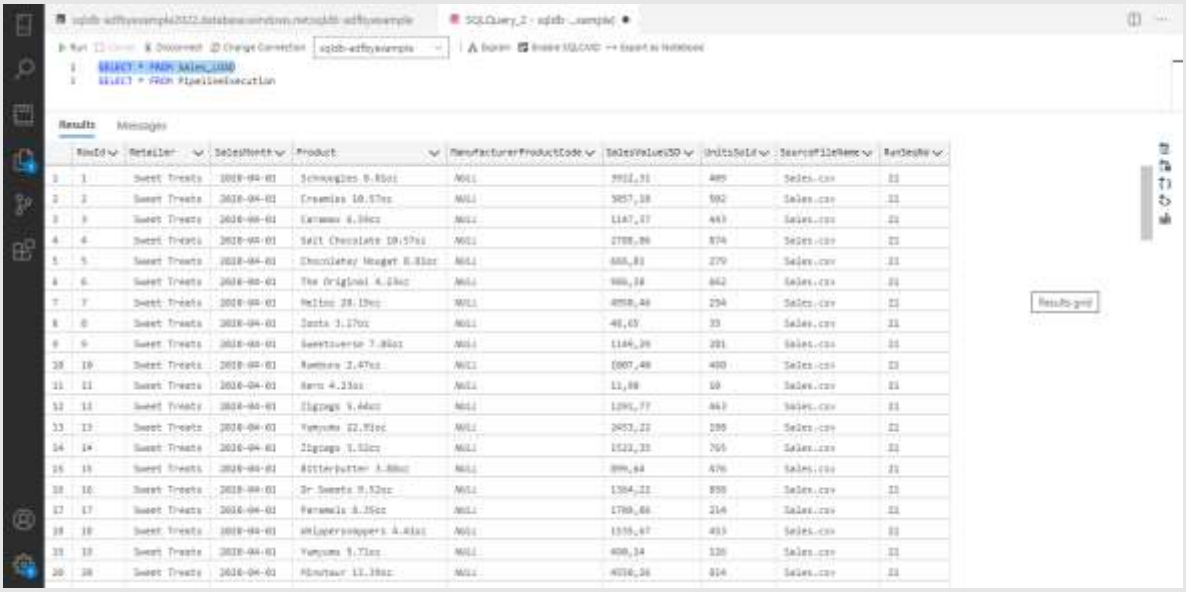


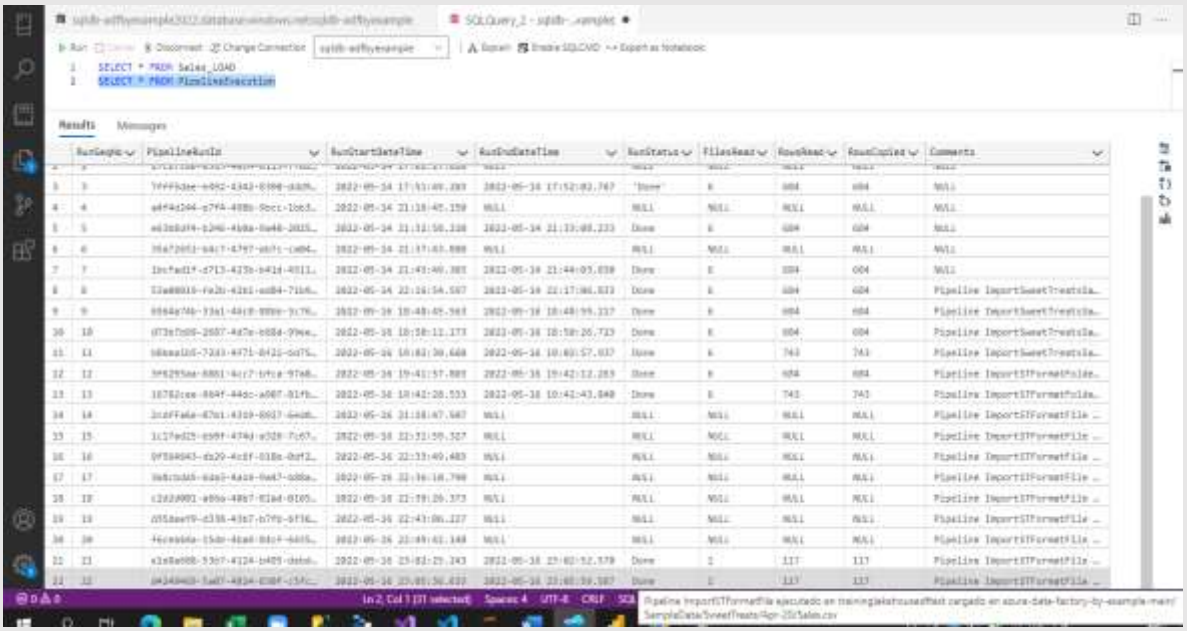
Figura 6-1 Actividad de copia de datos configurada para cargar un único archivo

Guarde y ejecute el proceso. Se le pedirá que proporcione valores para los parámetros "Directorio" y "Archivo" de la línea de producción: especifique la ruta completa y el nombre del archivo de datos de ventas de Sweet Treats o Desserts4All. Una vez finalizada la ejecución, verifique los resultados en las tablas de la base de datos [dbo].[PipelineExecution] y [dbo].[Sales_LOAD].


Pipeline run
Parameters
Name
Type
Value
Directorio
string
Archivo
string
Select...
OK
Cancel


Factory Resources
Activities
Lookup RunSeqNo
Set RunSeqNo
Import sales data
Log pipeline outco
Parameters
Variables
Settings
Output
Pipeline run ID:
Name
Type
Run start
Duration
Status
Log pipeline outcome
Stored procedure
Import sales data
Copy data
Set RunSeqNo
Set variable
Lookup RunSeqNo
Lookup
Succeeded


SELECT * FROM Sales_1000
SELECT * FROM PipelineExecution
Results
Messages
RowId
Retailer
SalesMonth
Product
ManufacturerProductCode
SalesValueUSD
UnitsSold
SourceFileName
RunSeqNo
Sweet Treats
Results grid


SELECT * FROM Sales_LOAD
SELECT * FROM PipelineExecution
Results
Messages

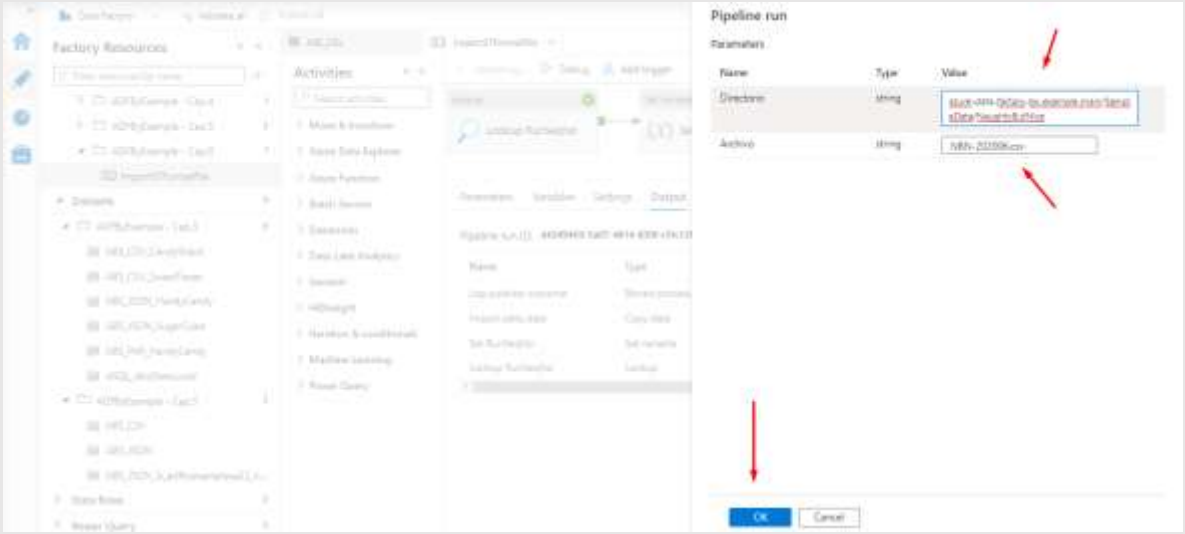
6.2. Utilizar condiciones de dependencia de la actividad

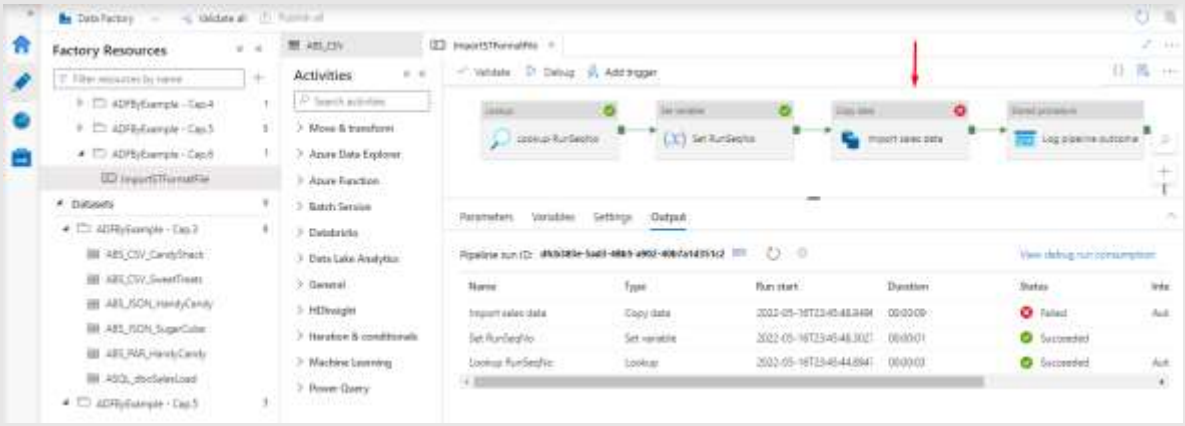
Un tercer minorista - "Naughty but Nice" - suministra datos de ventas en el formato Sweet Treats. La carpeta "NaughtyButNice" (en el mismo lugar que las carpetas "SweetTreats" y "Desserts4All") contiene los archivos de datos correspondientes. Vuelva a ejecutar el proceso "ImportSTFormatFile", esta vez con estos parámetros:

- En "Directorio", introduzca "azure-data-factory-by-example-main/SampleData/NaughtyButNice".

- Establezca "Archivo" como "NBN-202006.csv".

Consejo: Va a volver a ejecutar el proceso utilizando estos valores varias veces, por lo que es posible que desee establecerlos como valores predeterminados de los parámetros.





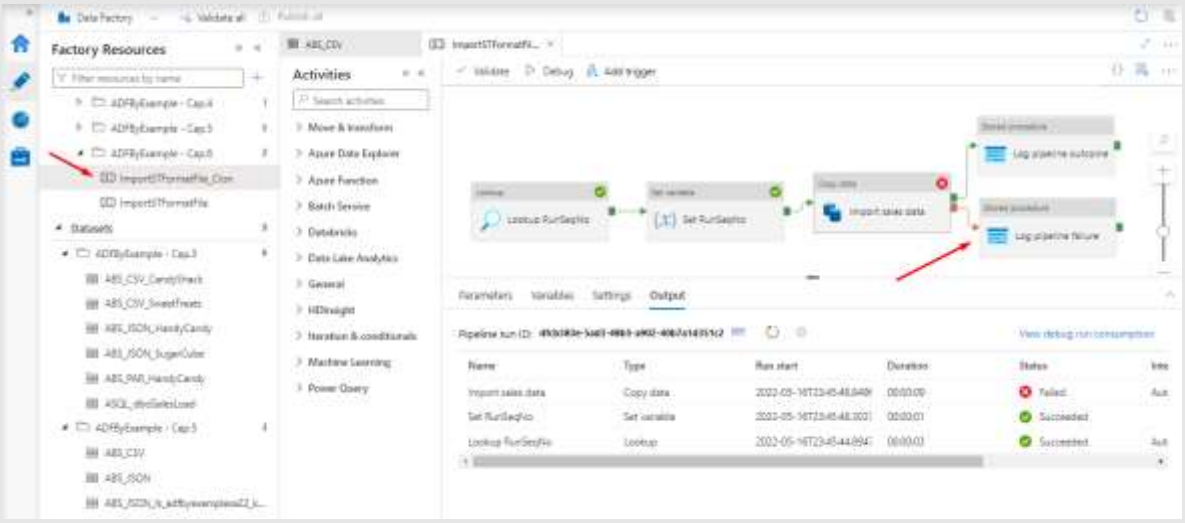
Cuando ejecute el proceso, éste fallará, informando de un error de conversión de tipo. Esto se debe a algunos datos mal formateados: falta una coma entre los dos primeros campos de algunos registros en la mitad del archivo.

El registro de ejecución en [dbo].[PipelineExecution] está incompleto para la ejecución del pipeline, porque no se alcanzó su actividad final de Stored procedure. Utilizando sólo la tabla de log, no es posible determinar si el pipeline ha fallado o simplemente sigue ejecutándose. Se puede mejorar esta situación registrando el fallo del pipeline.

1. Cree una copia de la actividad "Log pipeline outcome", la actividad final del Stored procedure de la pipeline. Para hacer esto, seleccione la actividad para mostrar sus controles de autoría, luego haga clic en el botón Clonar. Cambie el nombre de la nueva actividad por "Log pipeline failure".

2. La nueva actividad debe ejecutarse sólo si la actividad Copiar datos falla. Para configurar esto, seleccione la actividad Copiar datos. Desde sus controles de autoría, haz clic en el botón Add output (Añadir salida) y elige la condición de dependencia Failure (Fallo) en el menú emergente (mostrado en la Figura 6-2).

3. Aparece un asa roja en el borde derecho de la actividad Copiar datos. Haz clic y arrástralo a la actividad "Log pipeline failure".



4. La actividad "Log pipeline failure" ahora se ejecutará sólo si la actividad Copiar datos falla. Abra la pestaña de configuración de la actividad y desplácese hasta la sección Stored procedure parameters. Cambie el valor del parámetro "RunStatus" a "Failed".

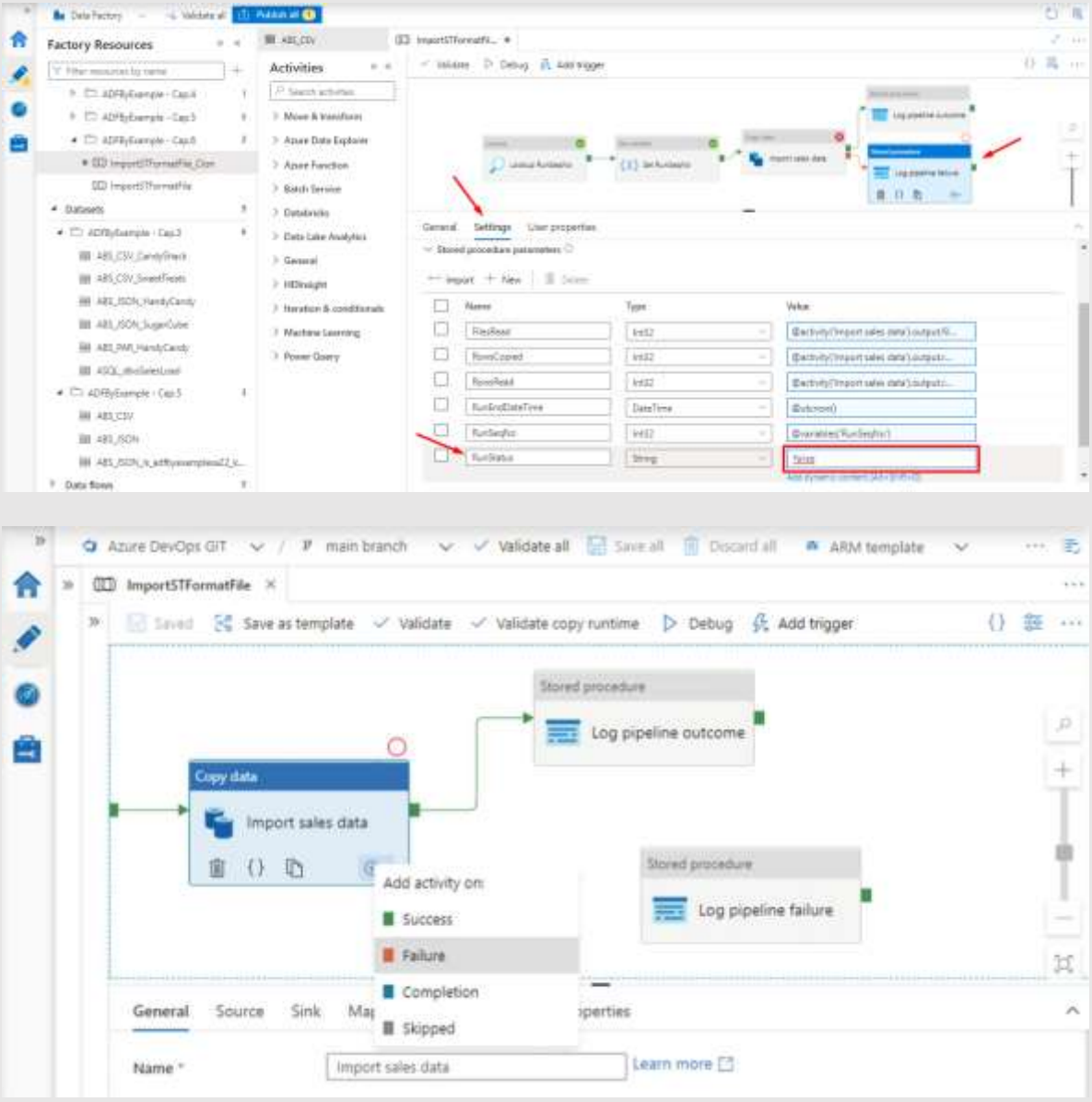
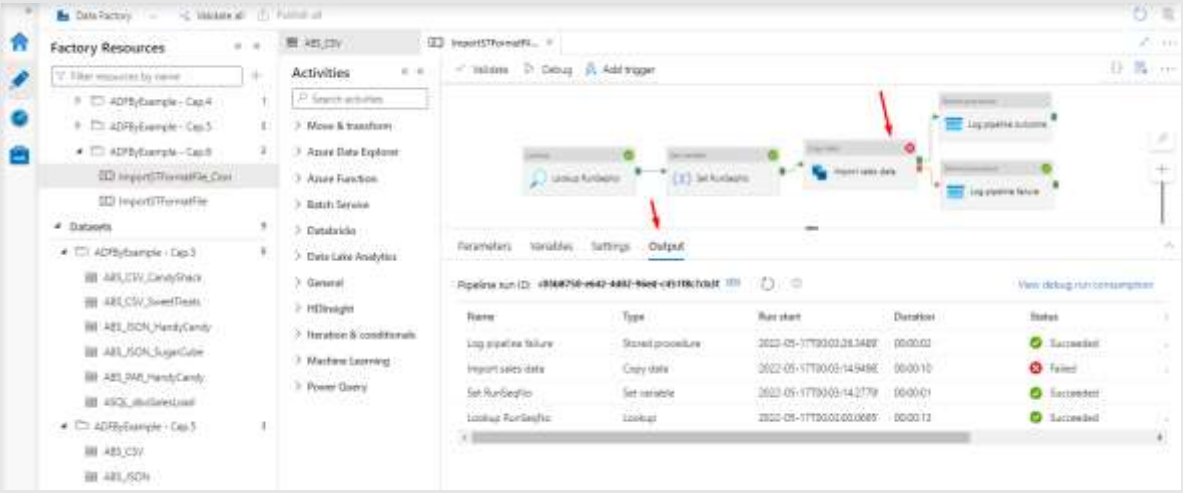
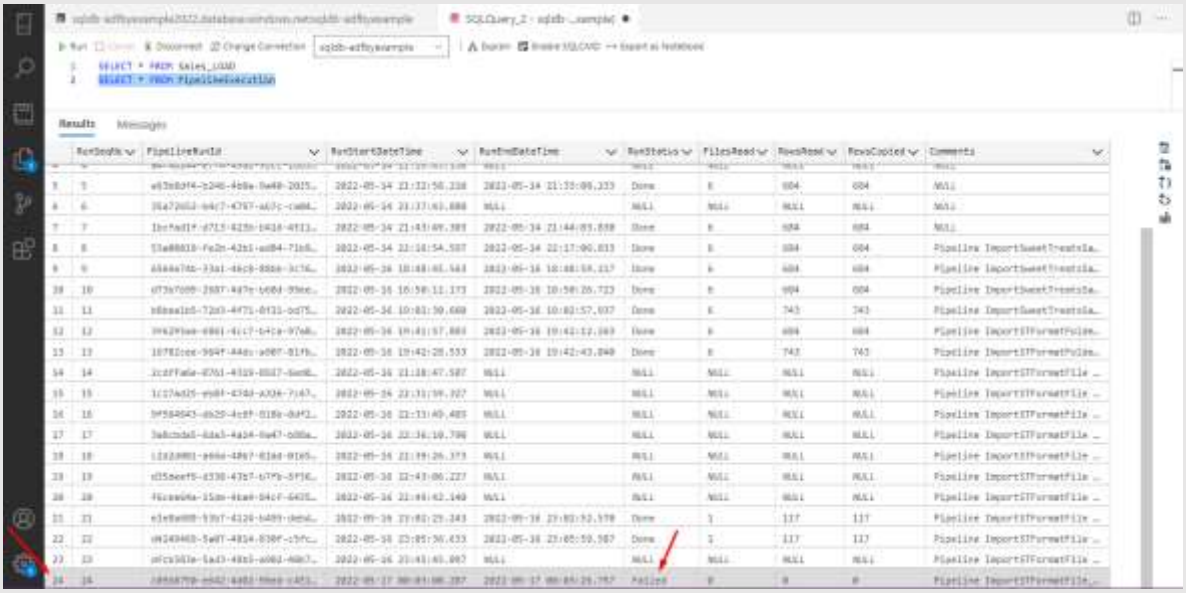


Figura 6-2 Añadir una condición de dependencia de fallo a la actividad Copiar datos

Guarde y ejecute el pipeline, utilizando los mismos valores de parámetros que antes. En la pestaña Output del panel de configuración del pipeline, observará que la actividad Copy data falla pero que la tarea de registro de fallos se ejecuta a continuación. El registro de la tabla de la base de datos [dbo].[PipelineExecution] muestra la ejecución del pipeline con un valor [RunStatus] de "Failed".



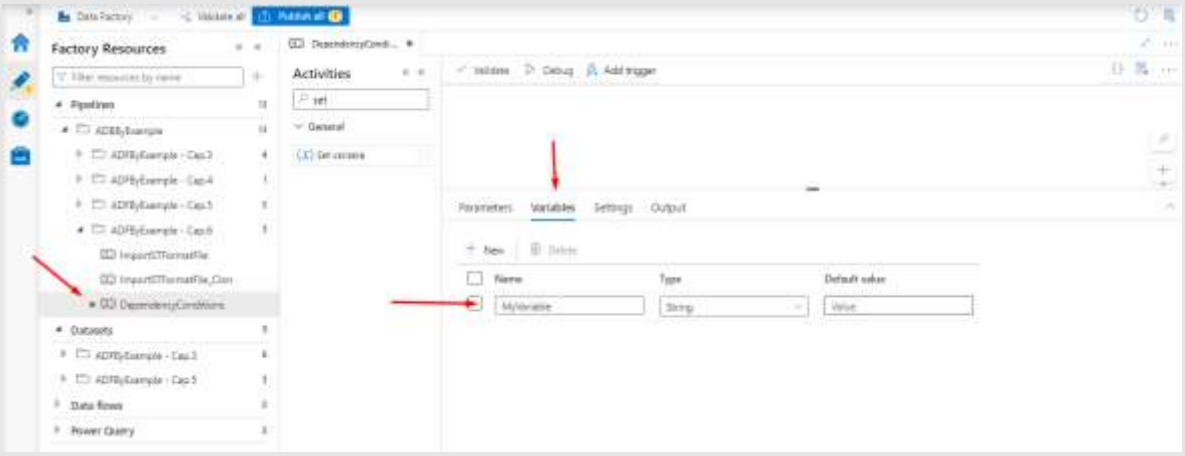


Sugerencia Mientras arrastra la ventana de visualización del lienzo para ver las diferentes actividades de un pipeline grande, es sorprendentemente fácil perder su lugar y terminar viendo una pantalla vacía. Para recuperarlo, haga clic con el botón derecho del ratón en algún lugar del espacio en blanco del lienzo y seleccione "Zoom para ajustar". Una vez que el lienzo se haya centrado de nuevo, seleccione Restablecer nivel de zoom en el menú emergente del botón derecho para restaurar las actividades a sus tamaños habituales.

6.2.1. Explorar las interacciones de la condición de dependencia

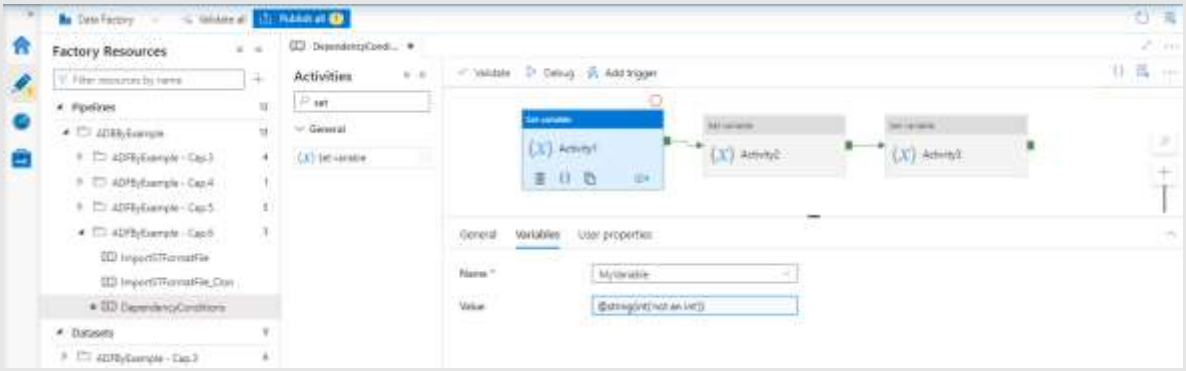
El menú emergente que se muestra en la Figura 6-2 contiene los cuatro valores posibles para una condición de dependencia de actividad: Éxito, Fallo, Finalización y Omisión. Combinando los cuatro valores de condición de diferentes maneras, puede lograr un sofisticado control de flujo en sus pipelines. En esta sección, usted creará un pipeline de prueba simple para permitirle explorar cómo diferentes combinaciones de condiciones de dependencia interactúan entre sí.

1. Cree un nuevo pipeline y defina una variable de pipeline de tipo String. Asigne un nombre a la variable.

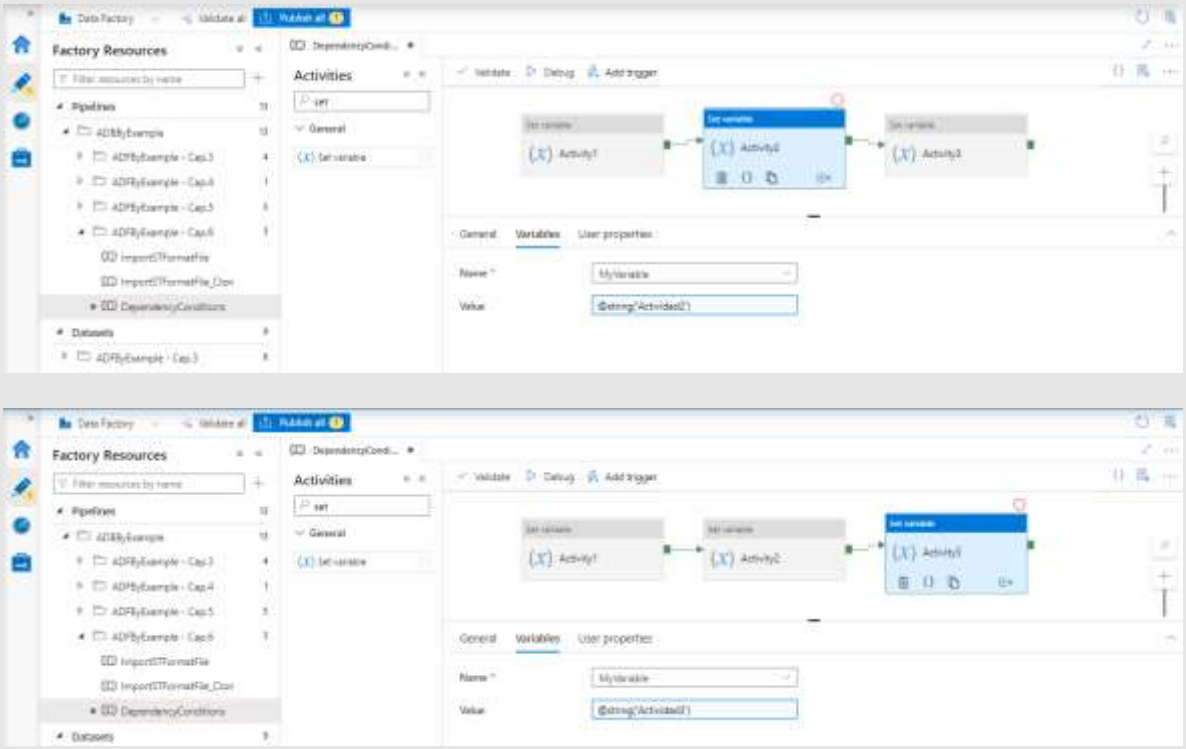


2. Arrastre tres actividades Set variable al lienzo de autoría y conéctelas en una cadena con las dependencias de la actividad Success.

3. En la pestaña Variables de la primera actividad, establezca el valor de su variable a la expresión @string(int('not an int')). Esta expresión intenta convertir el string "not an int" a un integer, lo cual fallará. (La conversión del string adjunto es necesaria para que coincida con el tipo de la variable).



4. Configure la segunda y tercera actividad para actualizar la misma variable, esta vez con valores de strings válidos de su elección.



La Figura 6-3 muestra la cadena de actividades Set variable junto con la pestaña de configuración Variables de la primera actividad.

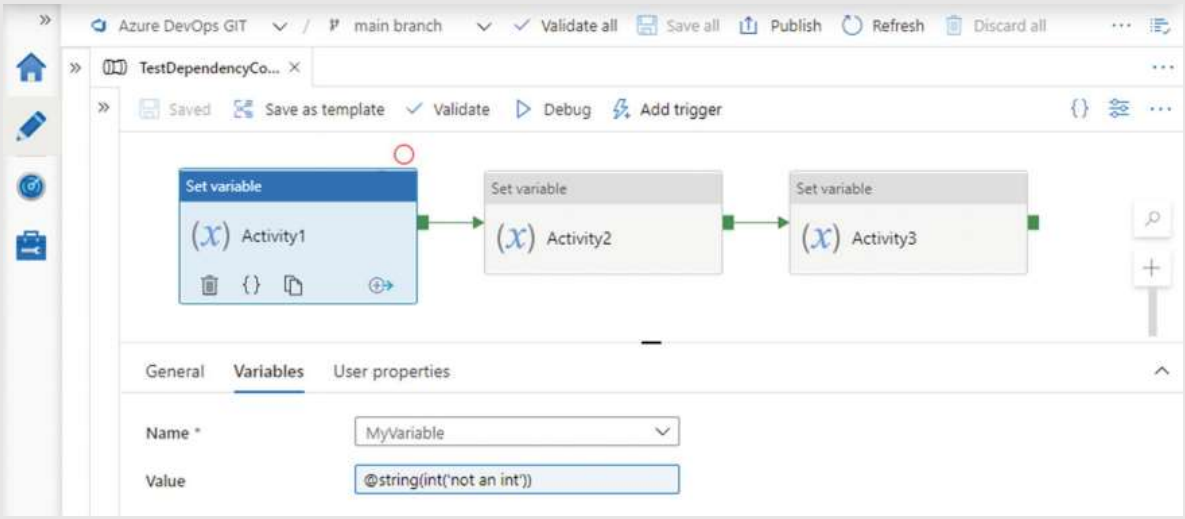
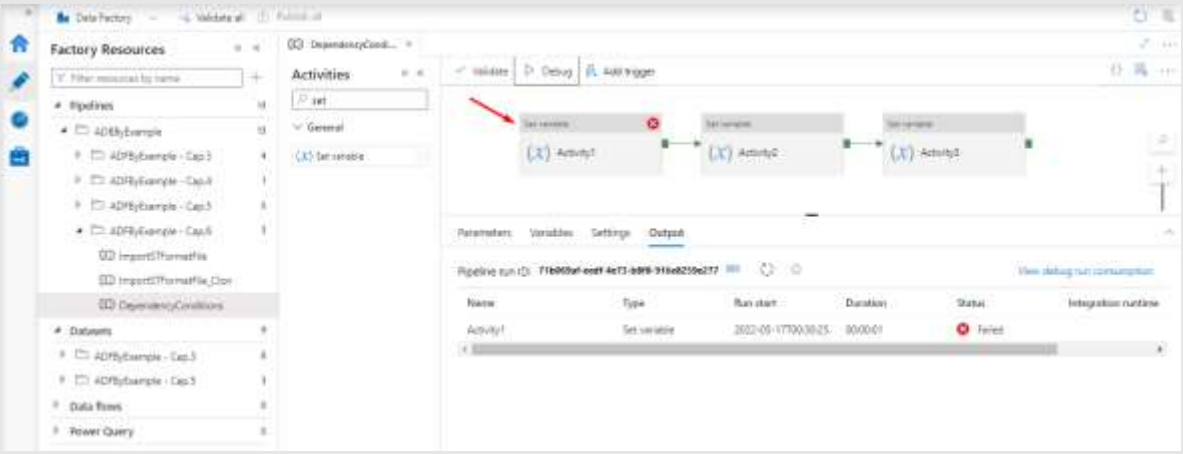


Figura 6-3 Actividad Set variable configurada para producir fallos

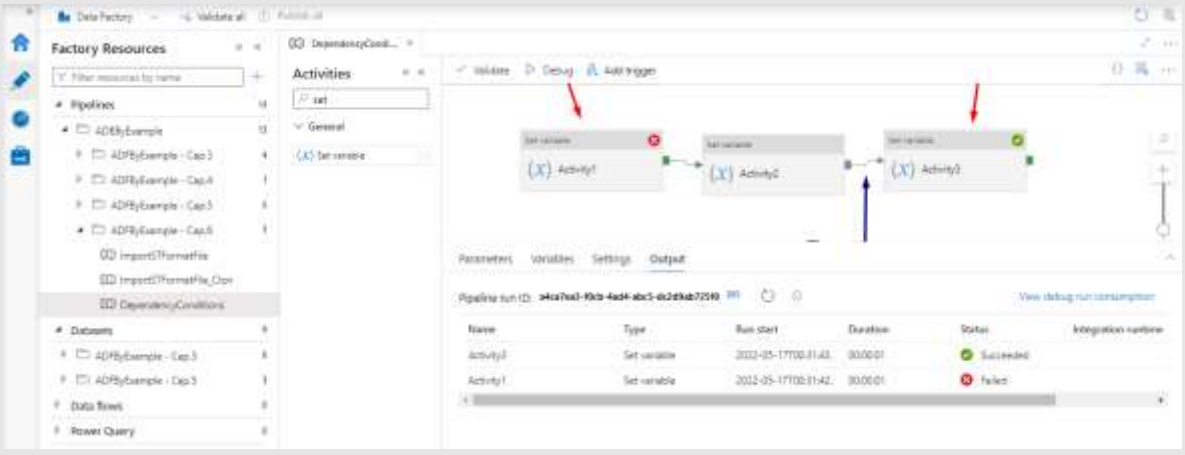
Ejecute el pipeline. Como es de esperar, la primera actividad falla y las dos siguientes no se ejecutan.



Comprender la condición de omisión (Skipped)

Haga clic con el botón derecho del ratón sobre la dependencia entre la segunda y la tercera actividad y cambie la condición de la dependencia a Skipped (Omisión). Vuelva a ejecutar la línea de producción. Esta vez

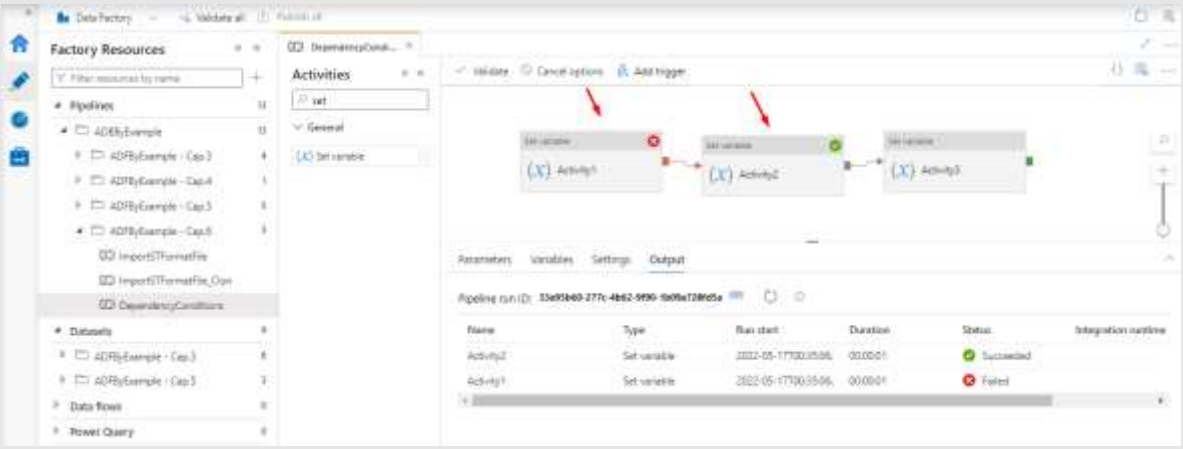
- La actividad2 no se ejecuta porque su condición de dependencia requiere que la actividad 1 tenga éxito.
- La Actividad3 se ejecuta porque su condición de dependencia requiere que la Actividad2 haya sido omitida.



Comprender la condición fallida (Failure)

Haga clic con el botón derecho del ratón sobre la dependencia entre la primera y la segunda actividad y cambie la condición de dependencia a Failure (Falla). Ejecute el pipeline, notando que

- La Actividad2 se ejecuta porque su condición de dependencia requiere que la Actividad1 falle.
- La Actividad3 ya no se ejecuta porque la Actividad2 no ha sido omitida.



Combinar las condiciones

Arrastre la manija verde de la segunda actividad y suéltela en la tercera, creando una condición de dependencia de Éxito, además de la condición de Omisión existente. La Figura 6-4 muestra las dos condiciones entre las actividades. Ejecute el proceso de nuevo para descubrir que las tres actividades se han ejecutado.

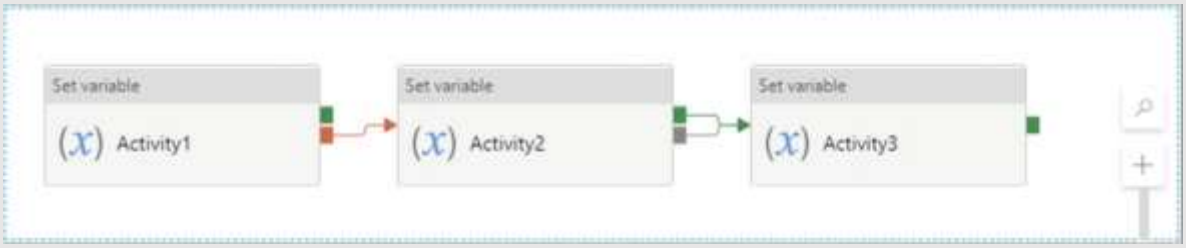


Figura 6-4 Dos condiciones de dependencia entre la Actividad2 y la Actividad3

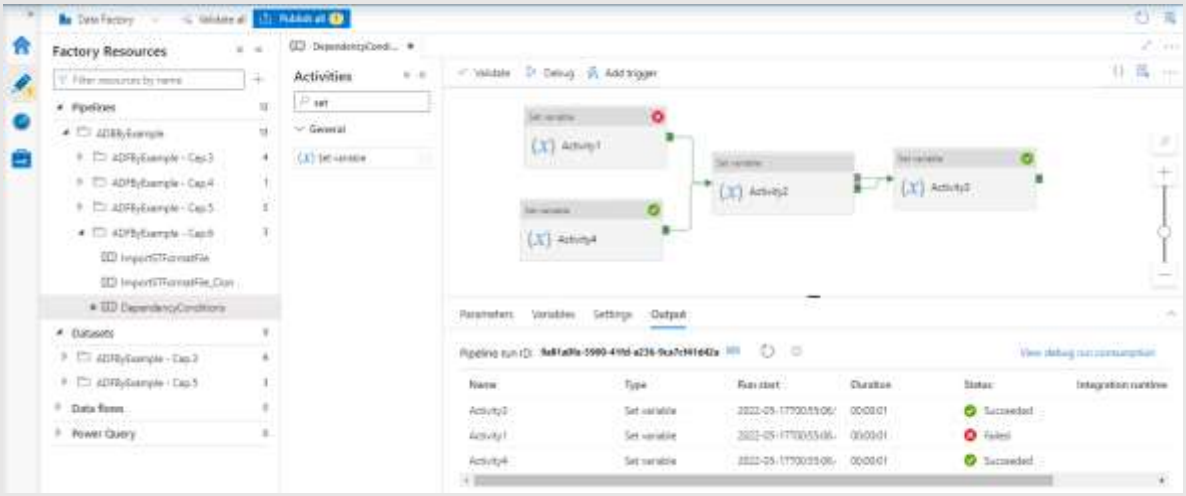
Cambie la condición entre las dos primeras actividades de nuevo a Success y ejecute el pipeline de nuevo. Esta vez, la Actividad2 no se ejecuta - la Actividad3 todavía lo hace, porque la condición Omitida se cumple.

Si se establecen múltiples condiciones de dependencia entre un par de actividades, la segunda actividad se ejecutará si se cumple alguna de las condiciones. En este ejemplo, la Actividad3 se ejecutará si la Actividad2 tuvo éxito o la Actividad2 fue omitida.

Crear Dependencias en Múltiples Actividades

Agregue una cuarta tarea Set variable a su pipeline y configúrela para asignar un valor de string válido a su variable. Cree una condición de dependencia de Success arrastrando el manejador de dependencia de la nueva actividad a Activity2, como se muestra en la Figura 6-5. Ejecute el pipeline. En esta ejecución

- La Actividad2 no se ejecuta porque la Actividad1 falla.
- La Actividad3 se ejecuta porque la Actividad2 se omite.



Cuando una actividad como la Actividad2 depende de más de una actividad ejecutada previamente, sólo se ejecuta si se cumplen las condiciones de dependencia de todas sus predecesoras. En este ejemplo, la Actividad2 se ejecutará sólo si la Actividad1 tuvo éxito y la Actividad4 tuvo éxito.

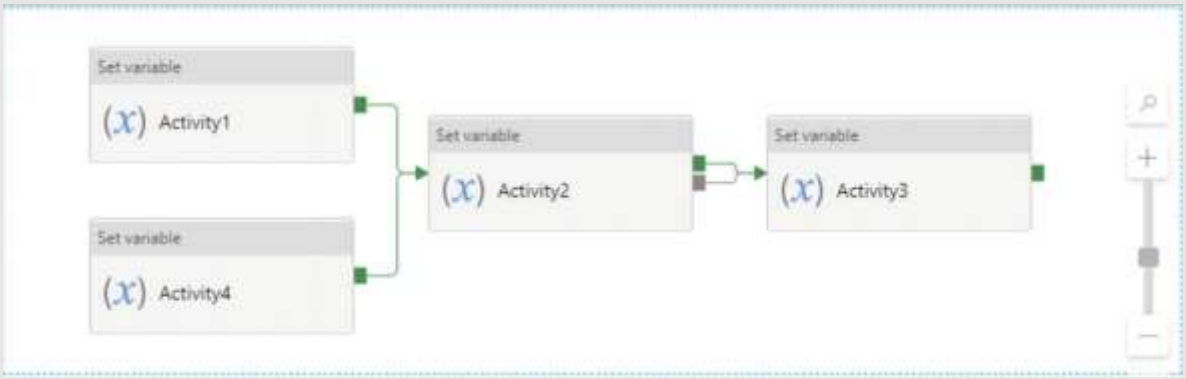
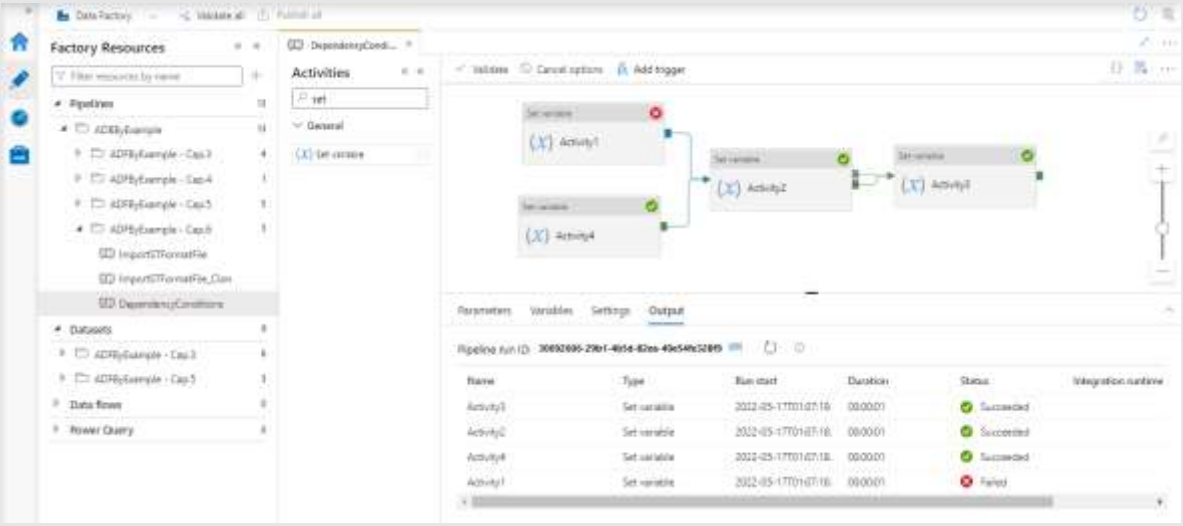


Figura 6-5 Actividad dependiente de múltiples predecesoras

Comprender la condición de finalización (Completion)

Cambie la condición de dependencia entre la actividad que falla y la segunda actividad a Finalización y ejecute el pipeline. Esta vez, las cuatro actividades se ejecutan.



Completion significa "either Success or Failure" - crear las condiciones de Success y Failure por separado tiene exactamente el mismo efecto. La Actividad2 se ejecuta en este caso porque la siguiente condición combinada es verdadera:

(Activity1 succeeded OR Activity1 failed) AND Activity4 succeeded

---

Para los desarrolladores de SSIS, las condiciones de dependencia de la actividad de ADF son comparables a las restricciones de precedencia de SSIS, con dos diferencias principales. En primer lugar, para que una actividad se ejecute, debe cumplirse una condición de dependencia para cada actividad de la que depende - las dependencias de actividad de ADF no tienen un equivalente a la propiedad LogicalAnd de las restricciones de precedencia de SSIS. En segundo lugar, SSIS soporta el uso de expresiones en las restricciones de precedencia, mientras que ADF no.

6.2.2. Entender el resultado del pipeline

Usted ha estado utilizando la pestaña Output en el panel de configuración del pipeline para inspeccionar los resultados de las actividades del pipeline. Además de los resultados de las actividades individuales, la ejecución de un pipeline tiene un resultado global propio, que se informa en la experiencia de monitorización de ADF UX. Para abrir la experiencia de monitorización, haga clic en el botón Monitor (icono de medidor) en la barra lateral de navegación.

La experiencia de monitorización se discute en profundidad en el Capítulo 12. Por el momento, seleccione la página de ejecuciones de pipeline, luego asegúrese de que la pestaña de depuración esté seleccionada - los controles relevantes se indican en la Figura 6-6.

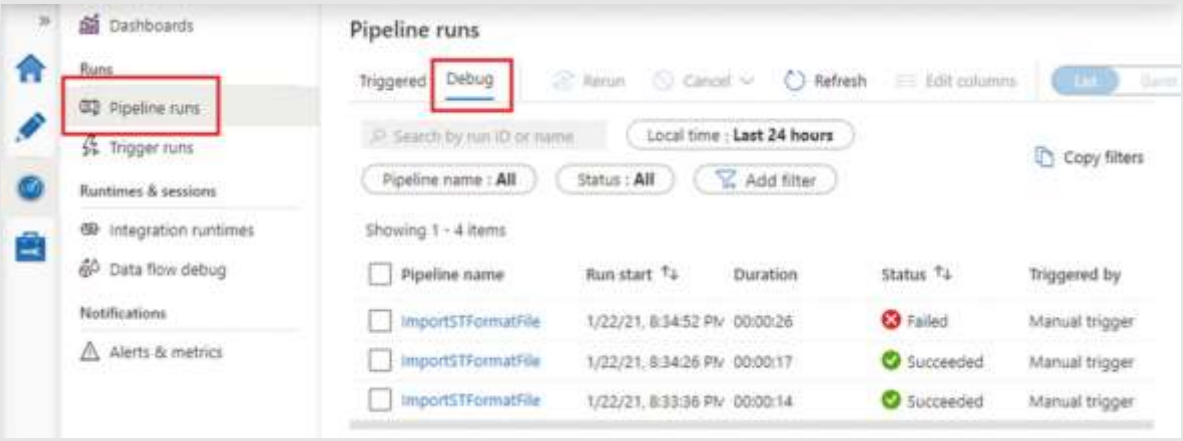


Figura 6-6 Ejecuciones de depuración de pipelines en la experiencia de monitorización de ADF UX

La Figura 6-6 muestra el resultado de la ejecución de la canalización "ImportSTFormatFile" tres veces, la tercera vez cargando el archivo Naughty but Nice "NBN-202006.csv", como se hizo anteriormente en el capítulo. Esta ejecución de la tubería se reporta como fallida, pero no simplemente porque la actividad de Copiar datos falló. Es útil poder entender la combinación de resultados de la actividad que produce este resultado.

El resultado de la ejecución de una tubería se determina de la siguiente manera:

- Se inspecciona el resultado de cada actividad "leaf". Una actividad "leaf" es una actividad que no tiene actividades sucesoras en el pipeline. Es importante destacar que si se salta una actividad de leaf, su predecesora se considera una actividad de leaf a efectos de evaluar el resultado del pipeline.

- Si alguna actividad leaf ha fallado, el resultado del pipeline también es fallido.

Las figuras 6-7 y 6-8 ilustran la diferencia entre estos escenarios. En la Figura 6-7, si la Actividad1 falla, entonces la Actividad2 se ejecuta y la Actividad3 se salta. Debido a que la Actividad3 es una leaf omitida, la Actividad1 se considera una leaf - y debido a que ha fallado, el pipeline mismo falla.

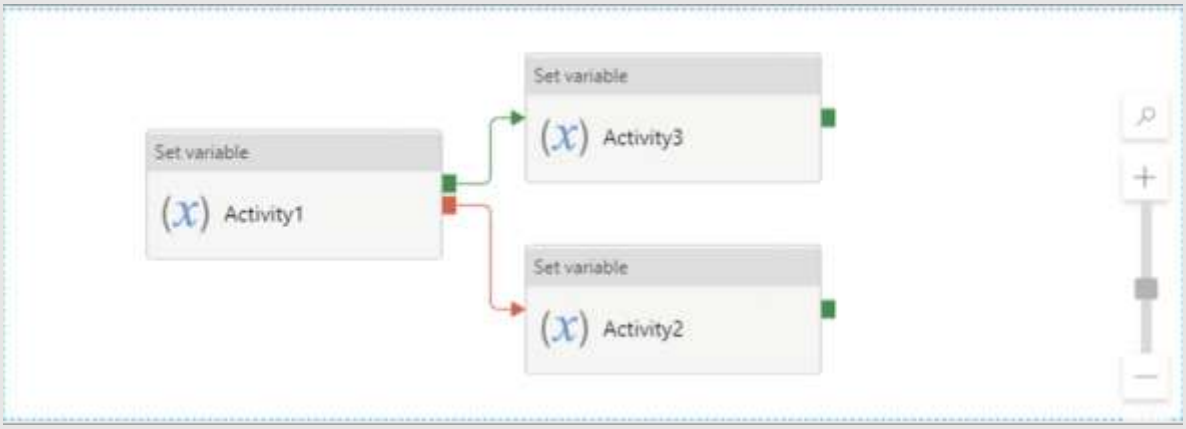


Figura 6-7 Cuando la Actividad1 falla, el pipeline falla

La Figura 6-8 muestra el mismo canal con la Actividad3 eliminada. En este caso, si la Actividad1 falla, la Actividad2 se ejecuta como antes, pero no se salta ninguna actividad. La Actividad3 es la única actividad leaf, por lo que, asumiendo que tiene éxito, el pipeline también tiene éxito.

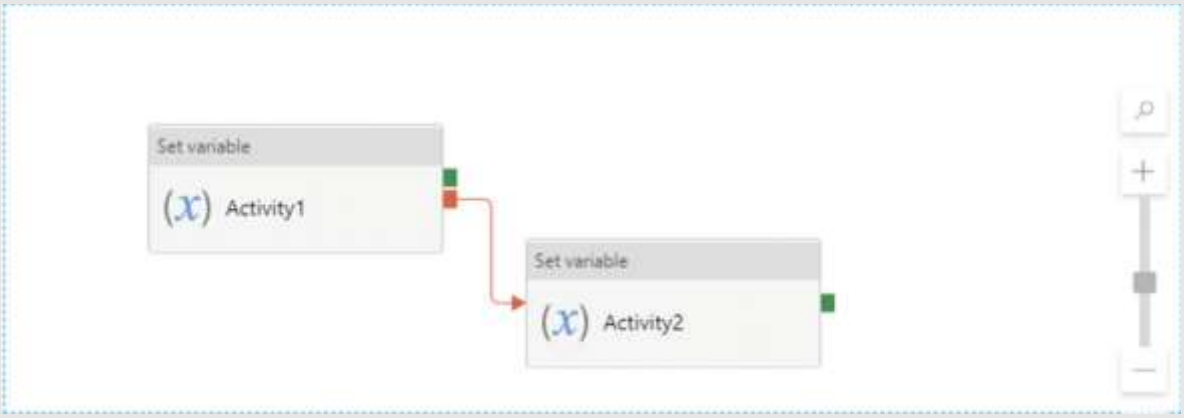


Figura 6-8 Cuando la Actividad1 falla, el proceso tiene éxito

En el caso del pipeline "ImportSTFormatFile", siempre se salta una actividad leaf - si la actividad Copy data tiene éxito, la actividad "Log pipeline failure" se salta, pero si la copia falla, "Log pipeline outcome" se salta en su lugar. Esto significa que la actividad Copiar datos del pipeline siempre se considera una actividad leaf (como predecesora de cualquier actividad de procedimiento almacenado que se haya omitido). Esto tiene el efecto - en este ejemplo - de que el resultado global del pipeline está determinado por el éxito o el fracaso de la actividad Copiar datos.

La Figura 6-9 muestra una versión alternativa de "ImportSTFormatFile", modificada para utilizar una única actividad de procedimiento almacenado con una condición de dependencia de finalización. Se utiliza una expresión ADF para calcular el valor del parámetro "RunStatus" del procedimiento almacenado (indicado en la figura). Esta versión del pipeline tiene sólo una actividad leaf - "Log pipeline outcome" - que se ejecuta tanto si la actividad Copy data tiene éxito como si no. Asumiendo que la actividad de registro tiene éxito, se informa que el pipeline tiene éxito incluso si la copia falla, porque todas sus actividades leaf han tenido éxito.

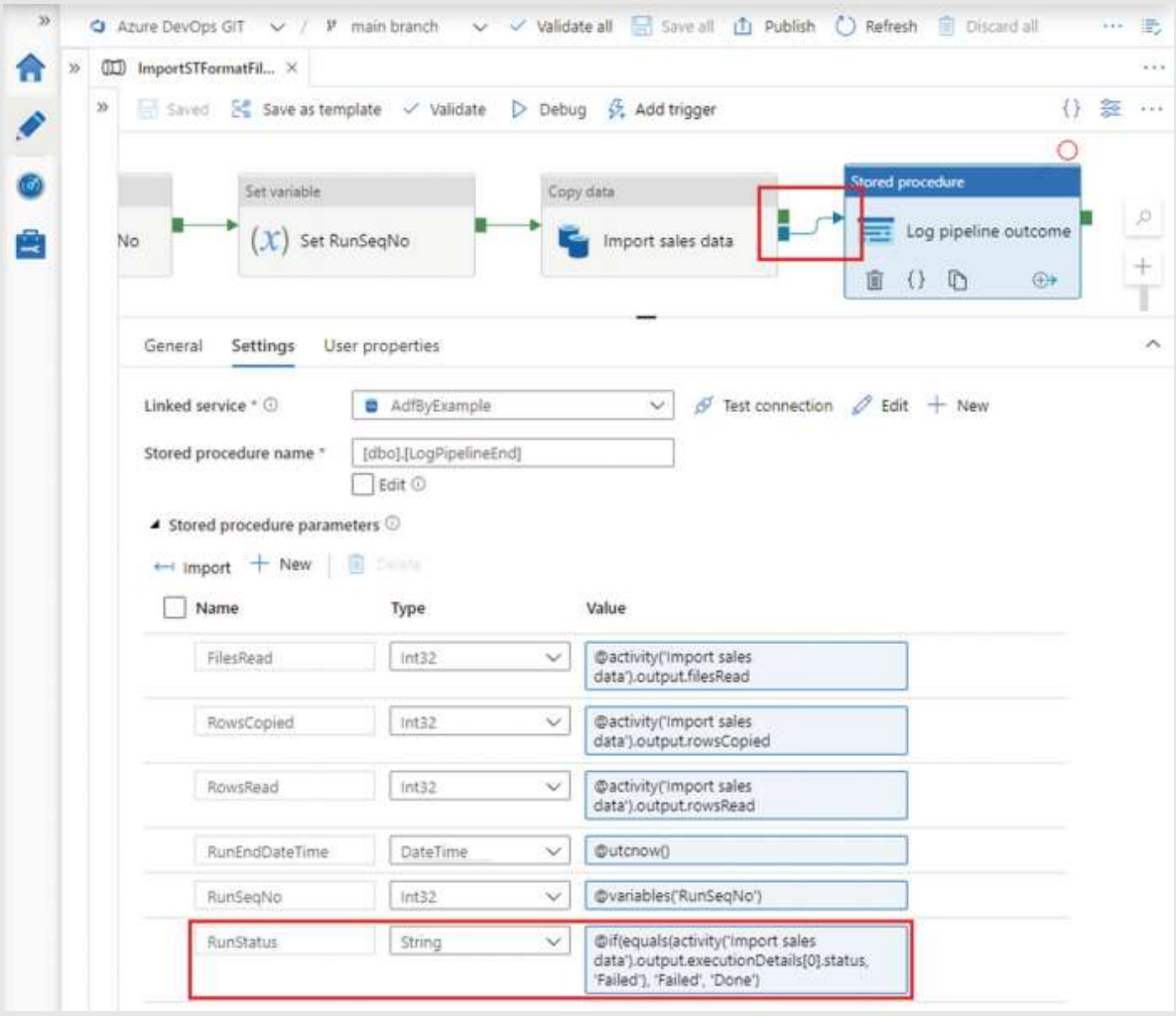


Figura 6-9 Una sola actividad de procedimiento almacenado reportando éxito o fracaso

Para los desarrolladores de SSIS Por defecto, cualquier error que se produzca durante la ejecución del paquete SSIS hace que el paquete falle, pero se puede modificar este comportamiento impidiendo que los errores se propaguen al paquete o proceso de llamada. La forma de cumplir este requisito en ADF es entender el comportamiento de las condiciones de dependencia de la actividad y organizarlas para conseguir el efecto deseado.

6.3. Levantamiento de errores

Entender por qué se informa de que un pipeline falla, le permite controlar la propagación de los errores planteados durante la ejecución de un pipeline. Por el contrario, a veces puede desear levantar errores propios, por ejemplo, si se detecta algún estado del sistema no válido.

En el momento de redactar este documento, ninguna actividad de Azure Data Factory permite generar errores directamente, pero un enfoque común es llamar a otra actividad de forma que se produzca un error. El fallo forzado de la actividad Set variable del ejemplo anterior (al intentar convertir un string no numérico en un entero) es un ejemplo de este patrón. Otro ejemplo es el uso de la sentencia RAISERROR o THROW de T-SQL para hacer fallar un procedimiento almacenado o una actividad Lookup.

En el Capítulo 3, te encontraste con un problema de deriva del esquema (schema drift), donde un campo renombrado en el esquema JSON de Sugar Cube causaba que los nombres de los productos desaparecieran de los datos cargados. El efecto de esto fue poblar el campo [Product] en [dbo].[Sales_LOAD] con NULLs en filas relacionadas con septiembre de 2020 - este es un ejemplo de estado inválido que podrías elegir para levantar como un error.

El listado 6-1 proporciona una versión revisada del procedimiento almacenado [dbo].[LogPipelineEnd]. El procedimiento inspecciona el campo [Product] de la tabla de la base de datos, utilizando RAISERROR para hacer que la línea de carga falle si se encuentran NULLs. Esto proporciona una comprobación básica de la calidad de los datos, convirtiendo las características indeseables de los datos en fallos detectables del pipeline. Si lo deseas, modifica tu procedimiento almacenado existente utilizando el script proporcionado - para probarlo, necesitarás crear un nuevo pipeline, combinando la carga de datos de Sugar Cube del Capítulo 3 con la funcionalidad de logging que desarrollaste después.

Listado 6-1 [dbo]. [LogPipelineEnd] genera un error si [Product] es NULL

```
ALTER PROCEDURE [dbo].[LogPipelineEnd] (
  @RunSeqNo INT
, @RunEndDateTime DATETIME
, @RunStatus VARCHAR(20)
, @FilesRead INT
, @RowsRead INT
, @RowsCopied INT
) AS
UPDATE dbo.PipelineExecution
SET RunEndDateTime = @RunEndDateTime
, RunStatus = @RunStatus
  , FilesRead = @FilesRead
  , RowsRead = @RowsRead
  RowsCopied = @RowsCopied
WHERE RunSeqNo = @RunSeqNo;
IF EXISTS (
  SELECT * FROM dbo.Sales_LOAD
  WHERE [Product] IS NULL
)
RAISERROR('Unexpected NULL in dbo.Sales_LOAD.[Product]', 11, 1);
```

6.4. Utilizar actividades condicionales

Existe una gran variedad de formas de manejar los errores de datos, dependiendo de factores que van desde los requerimientos de datos del negocio hasta el control de procesos técnicos. Hasta ahora, en este capítulo, has implementado pipelines que informan de fallos de carga, utilizando dependencias de actividades para controlar si la ejecución del pipeline tiene éxito o falla bajo esas condiciones.

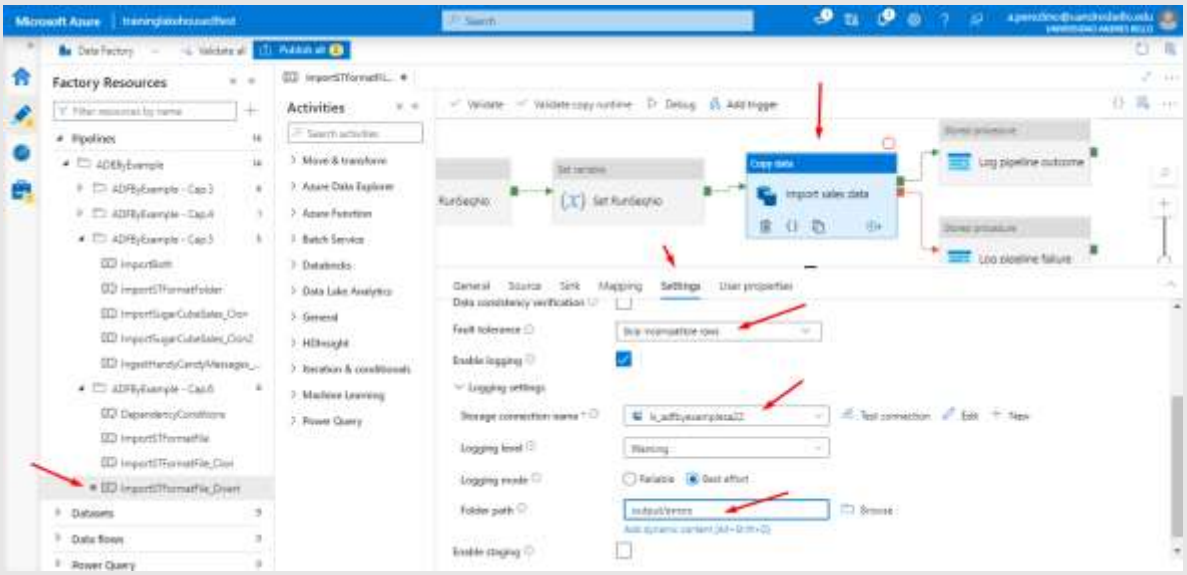
Tanto si el pipeline de carga falla como si no, los errores de formato en el archivo de ventas Naughty but Nice "NBN-202006.csv" hacen que se abandone la carga: cuando se encuentra un error, no se carga ninguna fila. Un enfoque alternativo es cargar todos los datos válidos que pueda, desviando otras filas a otro lugar para ser inspeccionadas y manejadas más tarde. En esta sección, utilizará la función de la actividad Copiar datos para desviar los errores, y luego utilizará una de las actividades condicionales de ADF para cargar los registros en una tabla de base de datos separada para su inspección.

6.4.1. Desviar filas de error

La actividad Copiar datos incluye una función que permite redirigir las filas con errores a un archivo log en el blob storage. Configure esa opción de la siguiente manera:

1. Clone el pipeline "ImportSTFormatFile" que se encuentra en el capítulo anterior. Nombre el clon "ImportSTFormatFile_Divert", luego abra la pestaña de configuración de la actividad Copiar datos.

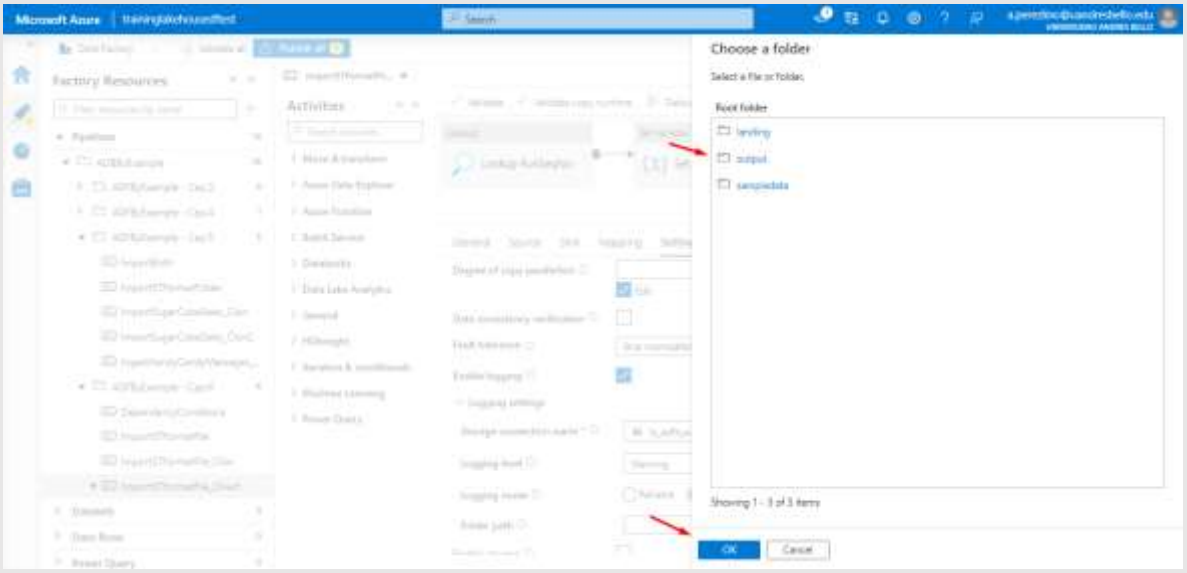
2. Desplácese hasta el desplegable de tolerancia a fallos (Fault tolerance) y seleccione "Omitir filas incompatibles"(Skip incompatible rows). Asegúrese de que la casilla de verificación "Enable logging" esté marcada, lo que hará que se muestren los Logging settings adicionales.

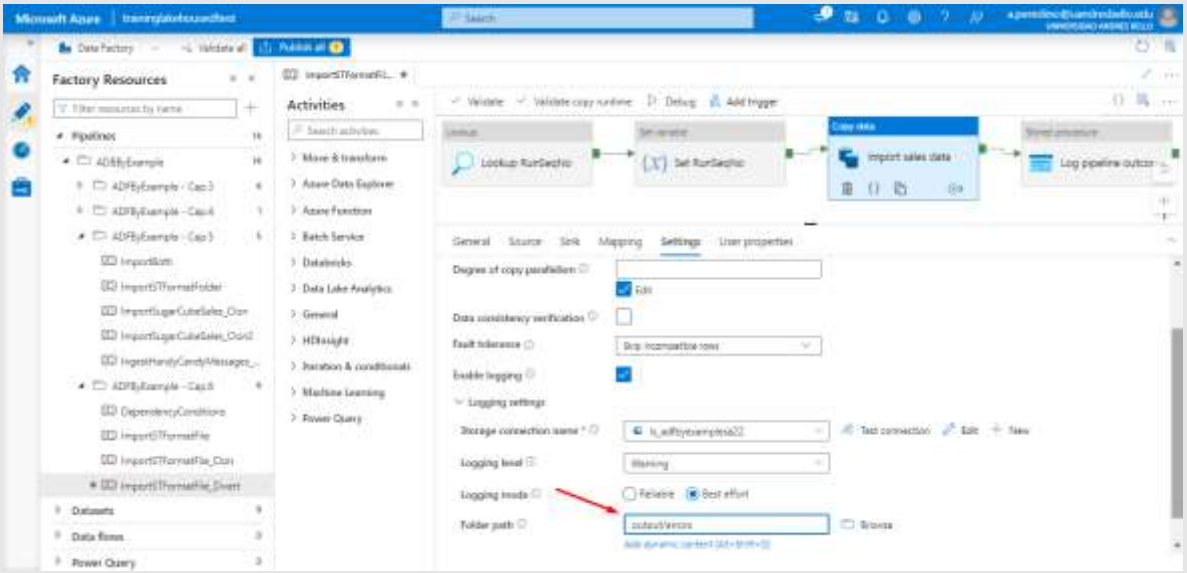


3. En Logging settings, elija su Azure blob storage linked service en el desplegable Storage connection name.

4. Se le pedirá un Folder path y, si ha elegido un servicio vinculado parametrizado, las propiedades requeridas del servicio vinculado. Acepte los valores predeterminados proporcionados para Logging level y Logging Mode.

5. Haga clic en el botón Browse (Examinar) situado a la derecha del campo Folder path (Ruta de la carpeta), elija el contenedor de "output" del blob storage y, a continuación, haga clic en OK (Aceptar). Edite el valor del Folder path, añadiendo "/errors".





La Figura 6-10 muestra los ajustes de fault tolerance configurados para la actividad de copia de datos.

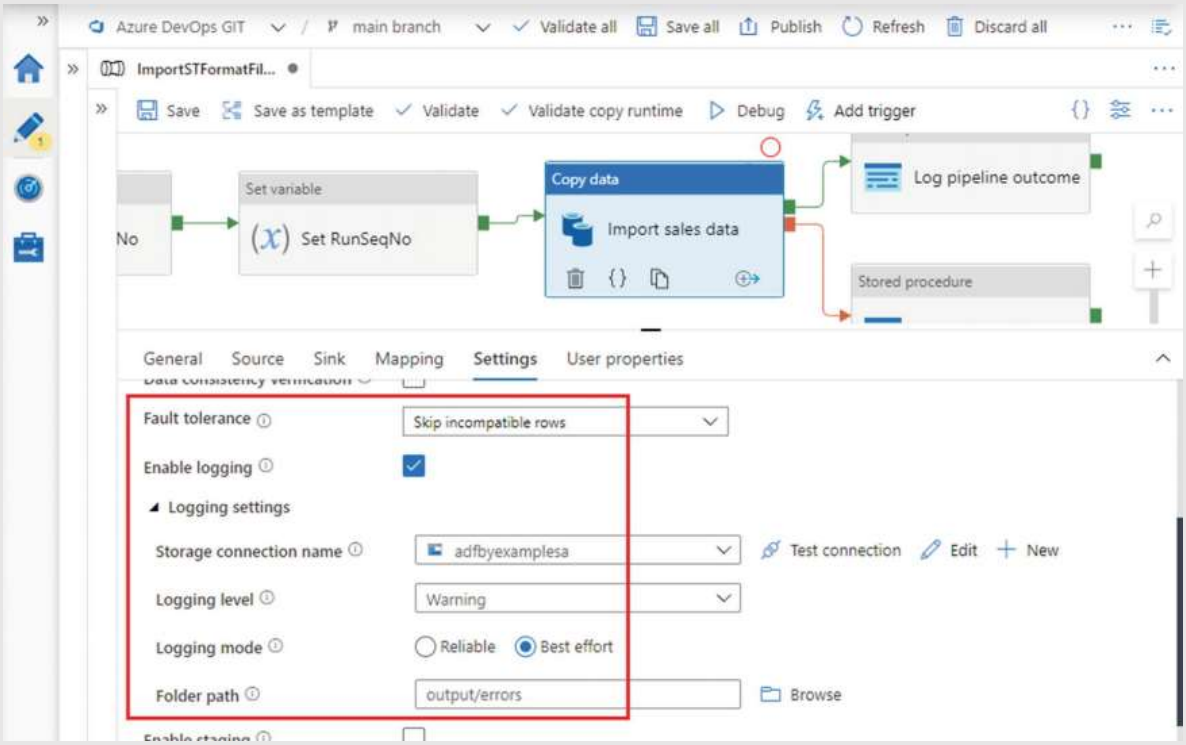


Figura 6-10 Configuración de fault tolerance de la actividad de copia de datos

---

Consejo Copy data activity logging puede activarse independientemente de la configuración de fault tolerance. Si se establece Logging level en "Info", se registrará la información sobre los archivos copiados correctamente, además de los archivos y filas omitidos.

Ejecute el pipeline modificado, cargando de nuevo el archivo de ventas "NBN-202006.csv" mal formateado. Esta vez, la nueva configuración de tolerancia a fallos (fault tolerance) de la actividad Copiar datos permitirá tener éxito. Cuando el pipeline se haya completado con éxito, inspeccione el objeto JSON de salida de la actividad Copiar datos (a través del enlace en la pestaña Output del panel de configuración del pipeline). En la Figura 6-11 se muestra un ejemplo que incluye la siguiente información

- **rowsRead**: El número de filas leídas del archivo de origen.
- **rowsCopied**: El número de filas copiadas a su Azure SQL DB. Es menor que el número de filas leídas, pero es mayor que cero porque se han cargado correctamente las filas formateadas en la tabla de la base de datos.
- **rowsSkipped**: El número de filas no copiadas. Son las filas mal formateadas en el archivo de origen.
- **logFilePath**: Las filas omitidas se han escrito en un archivo log específico para la ejecución de esta actividad. La ruta del archivo log termina en un nombre de carpeta GUID - este es el GUID

de la ejecución de la actividad que puede encontrar en el lado derecho de la pestaña Output en el panel de configuración del pipeline.

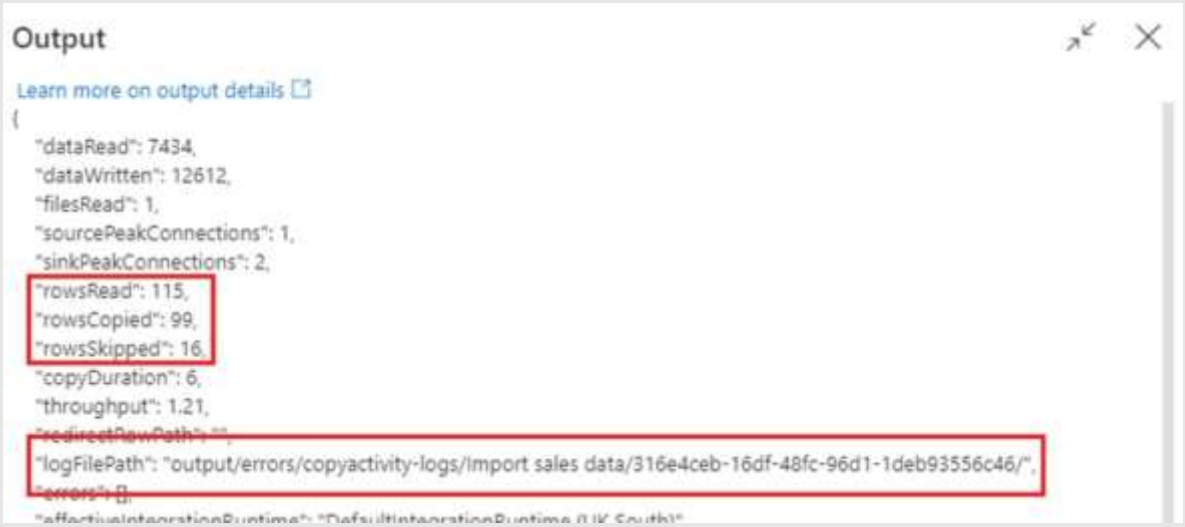


Figura 6-11 Información de tolerancia a fallos (Fault tolerance) en el JSON de salida de la actividad de copia de datos

Utilizando Azure Storage Explorer, busque la ubicación del archivo log. Descargue el archivo ".txt" que encontrará allí y ábralo en un editor de texto. La Figura 6-12 muestra un archivo de registro similar, también producido al cargar "NBN-202006.csv". El registro es un archivo CSV que contiene cinco columnas, la cuarta de las cuales - "OperationItem" - contiene toda la fila desviada.

Como se trata de un archivo CSV entre comillas, cada carácter de doble comilla en el registro original se ha escapado precediéndolo de otro. Aun así, se puede ver fácilmente que a los registros problemáticos les falta una coma entre los campos date y retailer (indicado en la Figura 6-12).

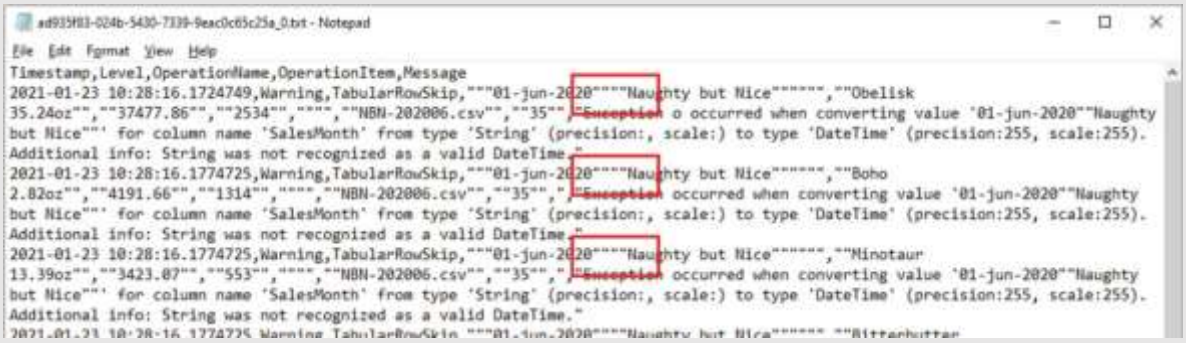


Figura 6-12 Registros de origen mal formateados en la actividad de copia de datos del log de fallos (Copy data activity fault log)

6.4.2. Cargas de filas de error

Una forma conveniente de inspeccionar los archivos log tabulares es recoger su contenido en una tabla de la base de datos. En esta sección, añadirá otra actividad Copiar datos para cargar el archivo log en una tabla de errores. Cree una tabla de log de errores en su Azure SQL Database utilizando el código del Listado 6-2.

```
CREATE TABLE dbo.LoadingError (
  [Timestamp] DATETIME2 NULL
, [Level] NVARCHAR(50) NULL
, [OperationName] NVARCHAR(50) NULL
, [OperationItem] NVARCHAR(4000) NULL
, [Message] NVARCHAR(4000) NULL
);
```

Listado 6-2Crear una tabla de errores de carga

Creación de un nuevo Sink Dataset

Su dataset existente de Azure SQL Database contiene un nombre de tabla codificado en su definición - [dbo].[Sales_LOAD]. Cree un nuevo dataset (y una carpeta "Chapter6") que le permita cargar log data en [dbo].[LoadingError], preferiblemente parametrizando el esquema y el nombre de la tabla en la definición del dataset. En el capítulo 7, podrá sacar más provecho de un dataset de tabla parametrizado.

Revisar el Source Dataset

Abra el conjunto de datos "ABS_CSV" que creó en el capítulo 5 e inspeccione las propiedades en su pestaña de configuración de la conexión. Hasta ahora, sólo ha realizado cambios en la configuración de la ruta del archivo, pero las propiedades del dataset pueden configurarse para adaptarse a muchas variaciones en el formato del archivo. Observe en particular que el carácter de escape por defecto es "Barra invertida (\)". Como ha visto en la Figura 6-12, el carácter de escape del archivo log es una comilla doble - cambie el valor seleccionado a "Double quote(")", y luego guarde el dataset revisado.

Al modificar recursos compartidos como los datasets, tenga cuidado de no introducir fallos en los pipelines u otros recursos que los utilizan. El cambio es seguro en esta situación, porque ninguno de los archivos CSV fuente que ha encontrado hasta ahora contiene caracteres de escape. Un enfoque alternativo es parametrizar el carácter de escape, dándole al parámetro del conjunto de datos un valor por defecto de "\" para garantizar la compatibilidad con versiones anteriores.

Consejo La hoja de Propiedades, disponible para los datasets y otros factory resources, incluye una pestaña Related, que indica los related factory resources. Al modificar un recurso existente, utilice la pestaña Related para identificar los recursos que pueden verse afectados por los cambios.

Utilizar la actividad de condición If

La nueva actividad Copiar datos utilizará la ruta del archivo log informada en la salida de la copia del archivo original. Un archivo log se encuentra en esta ubicación sólo cuando se han omitido uno o más registros; si no se omiten filas, no se crea ningún archivo. La ejecución de la actividad de copia de datos del archivo log cuando el archivo log no existe provocaría un error, por lo que sólo se puede permitir la ejecución de la actividad si se han omitido filas.

Puede controlar este comportamiento utilizando una actividad de condición If, que se encuentra en la sección Iteración y condicionales de la caja de herramientas de actividades.

1. Vuelva a la línea de producción "ImportSTFormatFile_Divert" y arrastre una actividad If Condition al lienzo de creación. Reorganice las dependencias de la actividad para colocarla entre las actividades Copy data de "Import sales data" y Stored procedure de "Log pipeline outcome" (log success), como se muestra en la Figura 6-13. Nombre la actividad "Si se omiten las filas (If rows skipped)".

2. Seleccione la pestaña de configuración de actividades de la condición If. Contiene un campo Expresión y dos casos: True y False. Los dos casos son contenedores de otras actividades - las actividades contenidas en el caso True se ejecutan cuando la expresión en el campo Expression se evalúa como true; las del caso False cuando no lo hace.

3. Haga clic en el campo Expresión para lanzar el constructor de expresiones. Introduzca una expresión que se evalúe como verdadera si la actividad anterior de Copiar datos ha omitido alguna fila. En la Figura 6-13 se muestra una posible opción.

4. Debajo del campo Expresión, haga clic en el icono del lápiz a la derecha del caso Verdadero - esto abre un nuevo lienzo que muestra las actividades del caso, actualmente ninguna. (Alternativamente, utilice el icono del lápiz que se muestra en el gráfico de actividades en el lienzo de autoría). El rastro de migas de pan en la parte superior izquierda del lienzo contiene el nombre del pipeline, el nombre de la actividad If Condition y las actividades True, indicando la parte del pipeline que está editando actualmente.

5. Arrastre una actividad de Copiar datos al lienzo de creación y abra su pestaña de configuración de Origen. Seleccione el conjunto de datos "ABS_CSV" e introduzca el valor "output" para su parámetro "Container". Establezca la opción de tipo de ruta de archivo como "Wildcard file path" y, a continuación, en Wildcard folder path, introduzca una expresión que devuelva la ruta del archivo log de la actividad de datos de copia anterior. Establezca el nombre de archivo comodín como "*".

Nota La ruta del archivo log informada por la actividad Copiar datos incluye el blob storage container - su expresión debe eliminar este prefijo. Una expresión que logra esto es @join(skip(split(activity('Import sales data').output.logFilePath, '/'), 1), '/').

---

6. Configure la pestaña Sink de la actividad Copiar datos para especificar la tabla [dbo].[LoadingError] utilizando su nuevo dataset de Azure SQL Database.

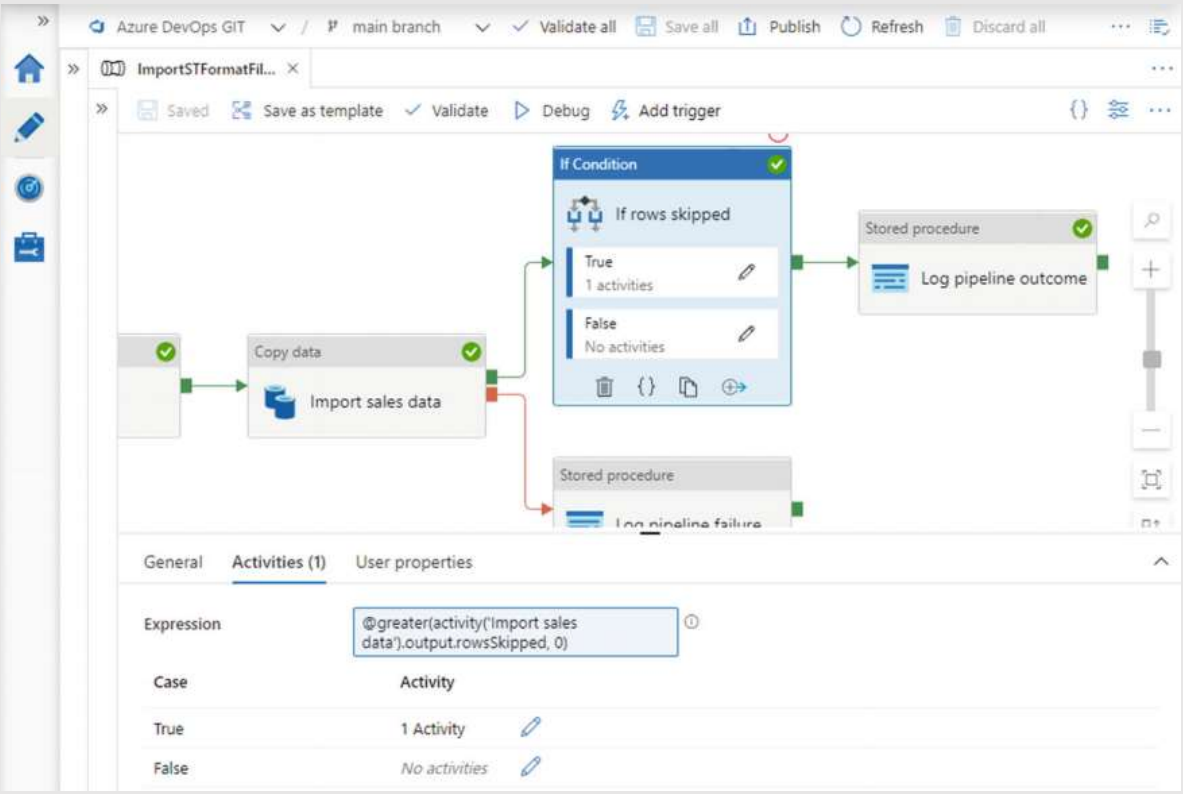


Figura 6-13 Configuración de la expresión de la actividad If Condition

Nota Cuando se selecciona una actividad en el lienzo de autoría dentro de la condición If, no se muestra ningún círculo de punto de interrupción. La UX del ADF no permite establecer un punto de interrupción en ninguna actividad anidada.

Run the Pipeline

Ejecute el pipeline de nuevo para cargar el archivo de ventas Naughty but Nice "NBN-202006.csv". Observe la pestaña de salida del panel de configuración del pipeline a medida que se ejecuta el pipeline, actualizándola regularmente para controlar el progreso. Observe que las seis actividades del pipeline aparecen en una lista, pero que el estado de la actividad If Condition permanece In progress (En curso) hasta que la actividad Copy data (Copiar datos) que contiene se pone en cola, se ejecuta y se completa.

Cuando el pipeline se haya completado con éxito, verifique que la tabla de la base de datos [dbo].[LoadingError] contiene ahora 16 filas copiadas del archivo log. La tabla de logs de la base de datos [dbo].[PipelineExecution] informa que se copiaron 16 filas menos de las que se leyeron.

Por último, ejecute el pipeline de nuevo, cargando un archivo de ventas diferente de Naughty but Nice, por ejemplo, "NBN-202004.csv". Este archivo no contiene errores de formato - la pestaña de salida del panel de configuración del pipeline muestra que la actividad If Condition es ejecutada (para permitir que su expresión sea evaluada), pero que la copia del archivo log no lo es.

6.4.3. Comprender la actividad Switch

La Condición If se compone de

- Dos ***casos*** - Verdadero y Falso - cada uno de los cuales contiene cero o más actividades.
- Una ***expresión*** que se evalúa como verdadera o falsa. Cuando esta expresión se evalúa en tiempo de ejecución, las actividades en el caso correspondiente se ejecutan.

La actividad Switch es la otra actividad condicional del ADF. Switch es una generalización de la actividad If Condition a más de dos casos. Consiste en

- Hasta 25 casos - identificados por diferentes valores de cadena - cada uno de los cuales contiene una o más actividades.
- Una expresión que se evalúa a un valor de cadena. Cuando esta expresión se evalúa en tiempo de ejecución, se ejecutan las actividades del caso correspondiente.

Además, la actividad Switch siempre contiene un caso Default. Si la expresión evaluada no coincide con ninguno de los casos identificados, se ejecutan las actividades del caso por defecto, de las cuales puede no haber ninguna.

TipADF no permite que las actividades condicionales sean utilizadas dentro de otras actividades condicionales. Si usted requiere una lógica anidada-si, la actividad Switch puede ayudarle a implementarla como una serie de casos. Una solución alternativa es crear su Condición If o Switch "interna" en un pipeline y su actividad condicional "externa" en otro - entonces puede usar la actividad Execute Pipeline para llamar al pipeline interno desde la actividad condicional externa.

La Figura 6-14 muestra una actividad Switch configurada con cuatro casos: "A", "B", "C" y Default. Cuando la expresión @variables('MyVariable') se evalúa como "A", se ejecutan las dos actividades del caso A. Las tres actividades de los casos B y C se ejecutan cuando la expresión se evalúa como "B" o "C", respectivamente. Si la expresión se evalúa con cualquier valor excepto "A", "B" o "C", se ejecuta la única actividad del caso por defecto.

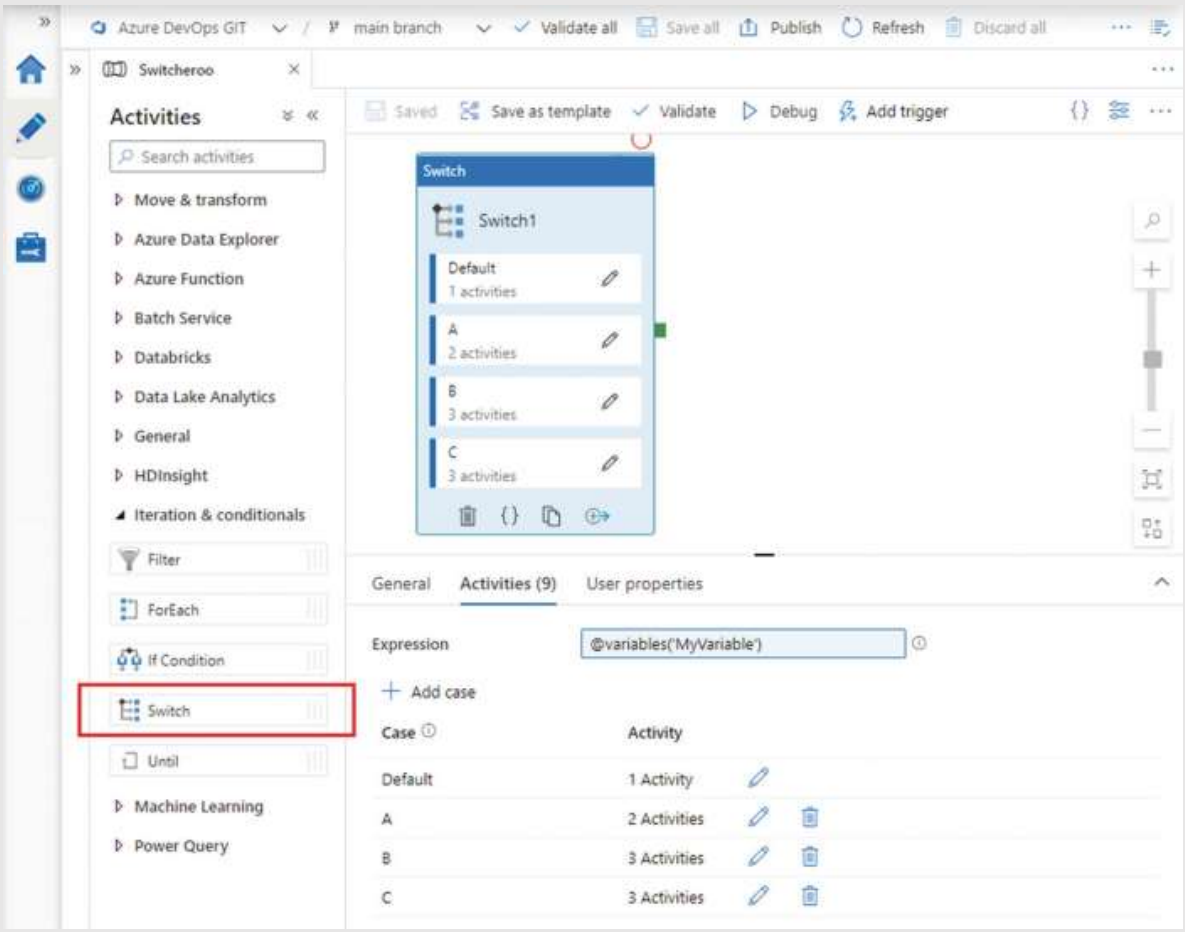


Figura 6-14 Configuración de la actividad Switch

Utilice la actividad If Condition cuando su flujo de control requiera sólo dos casos - si necesita más casos, utilice la actividad Switch.

Para los desarrolladores de SSIS SSIS no contiene ninguna tarea de flujo de control directamente comparable a las actividades If Condition o Switch, pero proporciona una funcionalidad similar utilizando expresiones en las restricciones de precedencia. En SSIS, ocasionalmente es necesario crear una tarea predecesora ficticia, para proporcionar una restricción de precedencia para implementar un comportamiento similar al de If/Switch - esto no es necesario en Azure Data Factory porque las expresiones y las dependencias de las actividades están desacopladas.

6.5. Utilizar actividades de iteración

El pipeline que ha desarrollado en la sección anterior proporciona un control de procesos y una gestión de errores más sofisticados que los de los capítulos anteriores, pero -por diseño- sólo carga un único archivo especificado. Para cargar todos los archivos de ventas de Naughty but Nice, el pipeline debe ejecutarse una vez para cada archivo. En esta sección, utilizará una de las actividades de iteración de Azure Data Factory para hacer esto automáticamente, iterando sobre una lista de archivos a cargar.

6.5.1. Utilizar la actividad Get Metadata

Antes de poder iterar sobre una lista de archivos, primero debes obtenerla - puedes hacerlo en ADF usando la actividad Get Metadata.

1. Cree un nuevo pipeline en la carpeta "Chapter6" y nómbrelo "ImportSTFormatFiles". Cree un pipeline parameter llamado "FolderPath", dándole el valor por defecto de la carpeta de datos de ventas Naughty but Nice ("azure-data-factory-by-example-main/SampleData/NaughtyButNice").

2. Arrastre una actividad "Get Metadata" -que se encuentra en la sección General de la caja de herramientas de actividades- al lienzo de creación. Nómbrela "Get file list" (Obtener lista de archivos). La actividad Get Metadata devuelve los metadatos relativos a un conjunto de datos del ADF - en su pestaña de configuración del conjunto de datos, elija el dataset "ABS_CSV".

3. Los archivos que deben cargarse son los que se encuentran en la carpeta del blob storage "NaughtyButNice" - utilizará la actividad "Get Metadata" para listar los archivos de la carpeta. Establezca la propiedad "Container" del dataset como "sampledata". Rellene la propiedad "Directory" con una expresión que devuelva el valor del parámetro "FolderPath" del pipeline.

4. El dataset "ABS_CSV" requiere un parámetro de archivo. Si no se especifica uno, el ADF utilizará el valor por defecto (".") que no coincidirá con la carpeta "NaughtyButNice". Para solucionar esto, sustituya el punto en el campo de propiedad "File" del dataset por un carácter de espacio, como se indica en la Figura 6-15.

5. Desplácese hacia abajo en la pestaña de configuración del Dataset hasta la sección de la lista de campos (también indicada en la figura). Aquí es donde se especifican los elementos de metadatos que debe devolver la actividad. Añada tres argumentos: "Exists,", "Item type," y "Item name".

6. Ejecute el pipeline. Si lo ha configurado correctamente, el JSON de salida de la actividad incluirá tres campos: exists = true, itemType = "Folder", y itemName = "NaughtyButNice". Si exists es falso, compruebe si la ruta de acceso a los archivos que ha configurado presenta errores.

7. Añada otro argumento a la lista de campos en la pestaña de configuración del Dataset de la actividad: "Child Items.". Vuelva a ejecutar el pipeline y compruebe que la salida de la actividad incluye ahora un campo array childItems que contiene la lista de archivos de la carpeta "NaughtyButNice".

Sugerencia Si el archivo o la carpeta de destino de la actividad "Get Metadata" no existe y no se especifica el argumento "Exists", la actividad fallará. Del mismo modo, si el argumento "Child Items" se especifica para una ruta que identifica un archivo en lugar de una carpeta, la actividad "Get Metadata" fallará. Retrasar el uso del argumento "Child Items" hasta el paso 7 le permite asegurarse de que la ruta configurada se refiere correctamente a una carpeta.

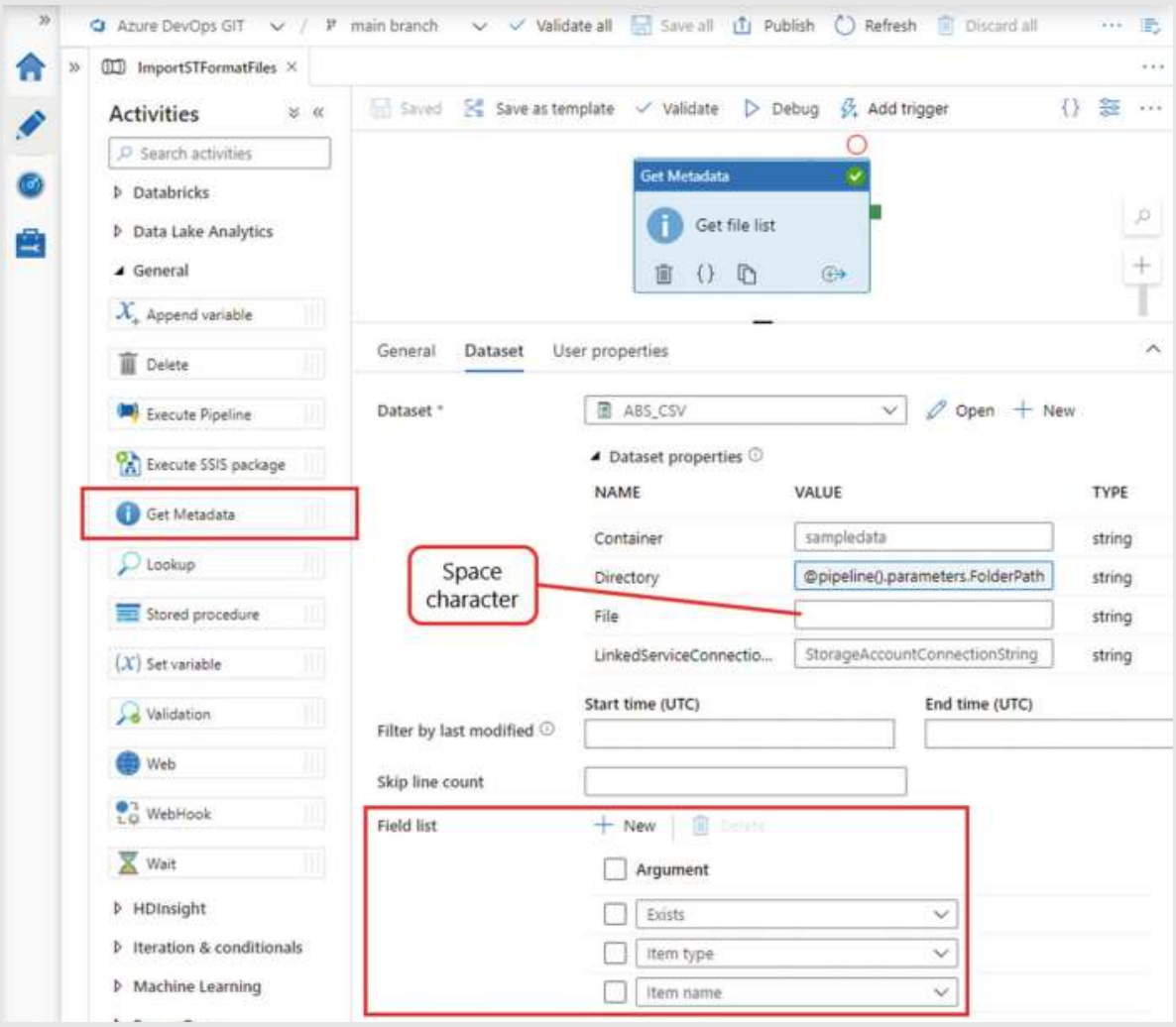


Figura 6-15 Configuración de la actividad Get Metadata

6.5.2. Utilizar la actividad ForEach

La actividad ForEach es una de las dos actividades de iteración de Azure Data Factory. Permite especificar un conjunto de actividades que se ejecutarán repetidamente, una vez por cada elemento de un array JSON dado, por ejemplo, el array de child items devuelto por la actividad Get Metadata.

1. Arrastre una actividad ForEach al lienzo de autoría, haciéndola depender de la actividad Get Metadata.

2. En la pestaña de configuración de la actividad, abra el constructor de expresiones en el campo Items.

3. Elija la salida de su actividad Get Metadata de la lista de salidas de la actividad del editor de expresiones, luego agregue ".childItems" a la expresión en el panel de expresiones. La expresión final debe ser @activity('Get file list').output.childItems.

4. Para editar el conjunto de actividades que se ejecutarán en cada iteración, haga clic en el icono del lápiz en la pestaña de configuración de Actividades. (Alternativamente, utilice el icono del lápiz que aparece en el gráfico de actividades en el lienzo de autoría).

5. Arrastre una actividad Execute Pipeline al lienzo vacío de ForEach. Abra su pestaña de configuración Settings y establezca su canalización invocada como "ImportSTFormatFile_Divert". Configure el parámetro "Directory" del pipeline invocado para que utilice el valor del parámetro "FolderPath" de este pipeline.

6. El valor del parámetro "File" del pipeline invocado debe ser el nombre del archivo para la iteración actual de la actividad ForEach. El elemento actual de la matriz de una actividad ForEach se especifica mediante la expresión item(). En este ejemplo, item() devuelve un elemento del array childItems en la salida de la actividad Get Metadata. El nombre del archivo se encuentra en el campo de nombre del objeto, por lo que la expresión completa del archivo es @item().name. La actividad Execute Pipeline configurada se muestra en la Figura 6-16.

7. Ejecute la canalización.

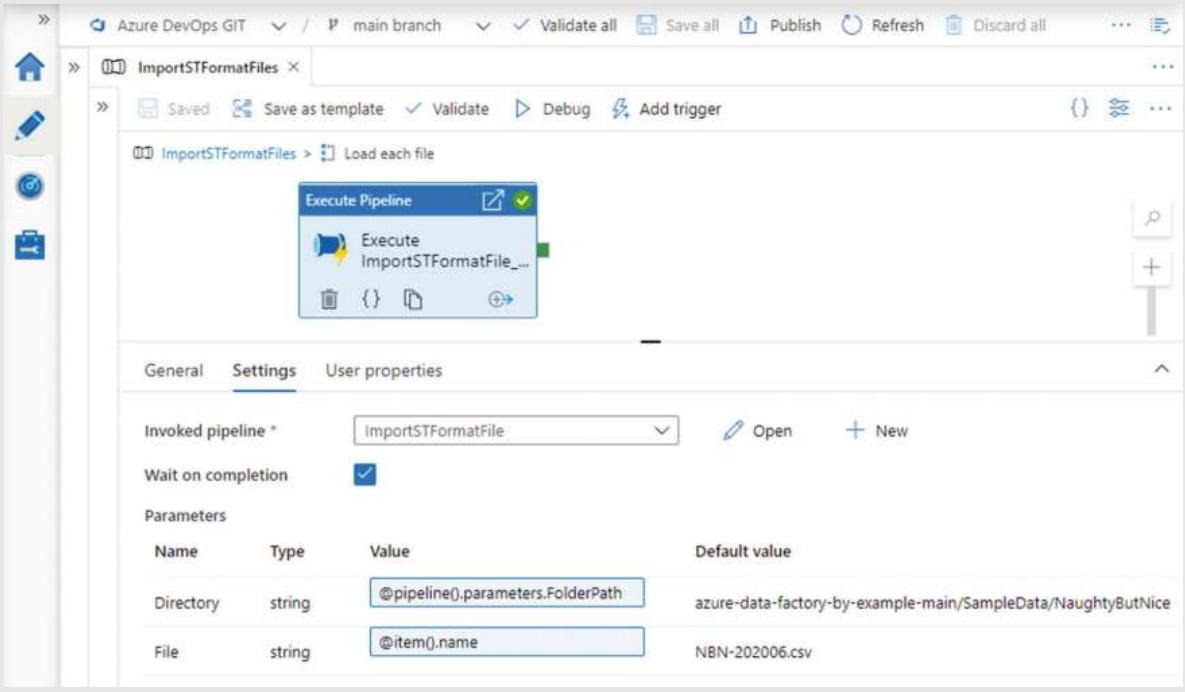


Figura 6-16 Actividad Execute Pipeline usando la expresión item() de la actividad ForEach

Cuando se ejecuta el pipeline en modo Debug, el ADF UX le advierte que "Todas las actividades dentro del bucle foreach se ejecutarán secuencialmente y cada actividad execute pipeline esperará a que se complete para propósitos de depuración". Esto contrasta con los pipelines publicados, donde el comportamiento por defecto de la actividad ForEach es ejecutar hasta 20 iteraciones simultáneamente, en paralelo. Una de las principales ventajas de Azure Data Factory sobre los servicios locales es la capacidad de escalar de esta manera, aprovechando brevemente más recursos para reducir el tiempo de procesamiento de extremo a extremo.

Cuando configuraste el array Items de la actividad ForEach (en su pestaña de configuración Settings), habrás notado dos opciones adicionales: una casilla de verificación Sequential y un campo Batch count. Estas opciones le permiten restringir el máximo paralelismo exhibido por la actividad. Si se marca la casilla de verificación Sequential se elimina la ejecución en paralelo, lo que hace que las iteraciones se ejecuten de forma secuencial. Cuando la casilla de verificación no está marcada, Batch count especifica el número máximo de iteraciones que se permite ejecutar en paralelo (hasta 50).

Consejo Debido a que las iteraciones de la actividad ForEach a menudo se ejecutan de forma secuencial en el modo de depuración, no es suficiente con probar los pipelines sólo en la UX del ADF. Los problemas de paralelización pueden no ser evidentes hasta que los pipelines se prueben en un entorno ADF publicado. La publicación de pipelines se discute en el Capítulo 10.

Utilice su cliente SQL para inspeccionar los últimos registros en la tabla de logs [dbo].[PipelineExecution]. La tabla muestra seis nuevas ejecuciones de pipeline, una para los datos de ventas de Naughty but Nice de cada mes. La columna [Comments] indica qué archivo fue cargado por cada ejecución de pipeline - la ejecución para el archivo "NBN-202006.csv" indica de nuevo que se leyeron más filas de las que se copiaron, mientras que todos los registros leídos de cada uno de los otros cinco archivos se copiaron con éxito.

Asegurar la paralelizabilidad

La primera vez que se encontró con problemas de paralelizabilidad fue al final del Capítulo 5, cuando fue necesario encadenar dos actividades de Execute Pipeline para evitar que se ejecutaran simultáneamente. Los pipelines que ha creado hasta ahora no son seguros en paralelo, porque en cada caso la secuencia de comandos de precopia de la actividad Copiar datos trunca la tabla [dbo].[Sales_LOAD]. Si se ejecutan varios pipelines simultáneos utilizando este patrón, se corre el riesgo de que cualquiera de ellos pueda truncar la tabla mientras los demás se encuentran a mitad de la copia.

La actividad ForEach no puede beneficiarse de la capacidad de escalamiento de ADF a menos que las iteraciones puedan ser aisladas unas de otras. En esta sección, modificará su pipeline por archivo para mover los datos cargados a una tabla separada, permitiendo que el pipeline se responsabilice de eliminar sólo sus propios registros de [dbo].[Sales_LOAD]. Esto reemplazará el enfoque anterior de mantener la tabla truncándola regularmente.

1. Los listados 6-3 y 6-4 proporcionan código para crear una tabla llamada [dbo].[Sales] y un procedimiento almacenado para actualizar la tabla desde [dbo].[Sales_LOAD]. Ejecute estos dos scripts en su cliente SQL para crear los dos objetos de la base de datos.

2. Edite su pipeline "ImportSTFormatFile_Divert", añadiendo una actividad de procedimiento almacenado entre la actividad Copiar datos "Import sales data" y la actividad If "If rows skipped (Si se omiten filas)".

3. Configure la actividad de procedimiento almacenado para llamar al procedimiento [dbo].[PersistLoadedSales] del Listado 6-4, utilizando la variable "RunSeqNo" del canal para proporcionar el valor del parámetro del procedimiento almacenado.

4. Edite la actividad Copiar datos, eliminando la secuencia de comandos Pre-copia de la pestaña de configuración Sink de la actividad.

```
CREATE TABLE dbo.Sales (
  RowId INT NOT NULL IDENTITY(1,1)
, Retailer NVARCHAR(255) NOT NULL
, SalesMonth DATE NOT NULL
, Product NVARCHAR(255) NOT NULL
, SalesValueUSD DECIMAL(19,2) NOT NULL
, UnitsSold INT NOT NULL
, RunSeqNo INT NOT NULL
, CONSTRAINT PK__dbo_Sales PRIMARY KEY (RowId)
);
```

Listing 6-3 Table [dbo].[Sales]

```
CREATE PROCEDURE dbo.PersistLoadedSales (
  @runSeqNo INT
) AS
DELETE tgt
FROM dbo.Sales tgt
  INNER JOIN dbo.Sales_LOAD src
    ON src.Retailer = tgt.Retailer
    AND src.SalesMonth = tgt.SalesMonth
    AND src.Product = tgt.Product
WHERE src.RunSeqNo = @runSeqNo;
DELETE
FROM [dbo].[Sales_LOAD]
OUTPUT
  deleted.[Retailer]
, deleted.[SalesMonth]
, deleted.[Product]
, deleted.[SalesValueUSD]
, deleted.[UnitsSold]
, deleted.[RunSeqNo]
INTO dbo.Sales (
  [Retailer]
, [SalesMonth]
, [Product]
```

```
, [SalesValueUSD]
, [UnitsSold]
, [RunSeqNo]
)
WHERE RunSeqNo = @runSeqNo;
```

Listado 6-4 Procedimiento [dbo].[PersistLoadedSales]

La tabla [dbo].[Sales_LOAD] estará ahora vacía la mayor parte del tiempo, excepto en los intervalos entre la carga de datos de un archivo por parte de un pipeline y su copia en [dbo].[Sales]. Para completar esta configuración, utilice su cliente SQL para truncar la tabla [dbo].[Sales_LOAD], de acuerdo con su nuevo estado de reposo.

Por último, vuelva a ejecutar el pipeline "ImportSTFormatFiles" para cargar los seis archivos de ventas de Naughty but Nice. Cuando se complete la ejecución, verifique que los datos de ventas de los seis meses entre abril y septiembre de 2020 están presentes en la tabla [dbo].[Sales] y que la tabla [dbo].[Sales_LOAD] está vacía.

Aunque la UX del ADF obliga a la actividad ForEach del pipeline a cargar los seis archivos de forma secuencial durante la depuración, el pipeline puede ahora cargar de forma segura los archivos en paralelo cuando se publica. Esto se debe a que cada ejecución del pipeline elimina de [dbo].[Sales_LOAD] sólo los datos que cargó, por lo que no interferirá con otras actividades (en este u otros pipelines) que se ejecuten al mismo tiempo.

Sugerencia Recuerde que las variables de la línea de producción tienen un alcance a nivel de la línea de producción, por lo que las actividades Set variable y Append variable de ADF nunca son seguras en paralelo. Las iteraciones ForEach que modifican las variables producirán resultados imprevisibles y deben evitarse. Una solución común es encapsular las actividades de una iteración en una actividad Execute Pipeline anidada, tal y como se ha hecho aquí.

6.5.3. Entender la actividad Until

Al igual que la actividad ForEach, la actividad Until de ADF permite especificar un conjunto de actividades a realizar repetidamente. En lugar de iterar sobre un array fijo de elementos, la actividad Until se repite indefinidamente hasta que se cumpla su condición de finalización (o se agote la actividad). La Figura 6-17 muestra un ejemplo de la expresión de la condición de finalización de una actividad Until - la actividad se repetirá hasta que el valor de una variable llamada "FilesRemaining" llegue a cero, o se alcance el tiempo de espera por defecto de siete días.

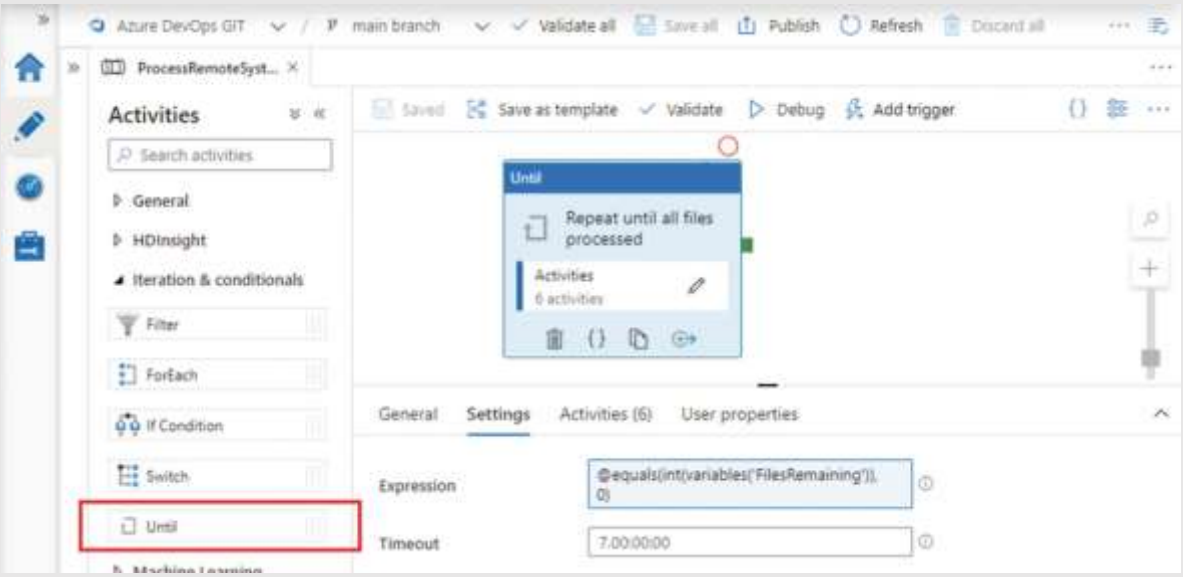


Figura 6-17 Expresión que proporciona la condición de finalización de una actividad Until

---

Sugerencia ADF no permite el anidamiento de actividades de iteración, ni permite utilizarlas dentro de actividades condicionales. Como antes, una solución es usar diferentes pipelines para implementar los bucles internos y externos de una iteración anidada.

Los casos típicos de uso de la actividad Until son situaciones que implican dependencias externas - por ejemplo, para retrasar la extracción de datos de una fuente de datos hasta que el sistema de origen haya finalizado algunas tareas internas. Una o más actividades dentro de la actividad Until usualmente tienen la tarea de reevaluar la situación en cada iteración - en el ejemplo de la Figura 6-17, esto podría consistir en recalcular el número de archivos restantes, luego usar la actividad Set variable para asignar el valor calculado a la variable "FilesRemaining".

Debido a que las iteraciones de la actividad Until no corresponden a elementos de un array, la sintaxis item() no está soportada en este contexto. Por la misma razón, las iteraciones paralelas no son posibles - la ejecución de las iteraciones de la actividad Until es siempre secuencial. La expresión de la condición de terminación de la actividad Until se reevalúa al final de cada iteración - esto significa que las actividades dentro de una actividad Until siempre se ejecutan al menos una vez.

Para los desarrolladores de SSIS, la actividad ForEach de ADF es directamente comparable al contenedor de bucles Foreach de SSIS, aunque SSIS permite además bucles anidados dentro de un mismo paquete. El patrón de iteración indefinido de la actividad Until de ADF no se admite directamente en SSIS. Un contenedor de bucle For que no especifica AssignExpression consigue un efecto similar, pero tiene la pequeña diferencia de que EvalExpression se evalúa al principio de cada iteración (por lo que es posible una ejecución con cero iteraciones).

Revisión del capítulo

En este capítulo, usted utilizó tres variedades de herramientas para controlar el flujo de ejecución del pipeline:

- **Condiciones de dependencia de la actividad**: Para determinar lo que sucede después de que se complete la ejecución de una actividad
- **Actividades condicionales**: Para ejecutar subconjuntos separados de las actividades de un pipeline bajo diferentes condiciones
- **Actividades de iteración**: Para ejecutar subconjuntos de las actividades de un proceso repetidamente.
- **Utilizado colectivamente** -incluyendo la anidación de actividades condicionales y de iteración en diferentes pipelines- esto forma un amplio arsenal de herramientas para controlar el flujo.

Conceptos clave

Condición de dependencia: Caracteriza la dependencia de una actividad. Una actividad dependiente de otra sólo se ejecuta si se cumple la condición de dependencia asociada, es decir, dependiendo de si la actividad anterior tiene éxito, falla o se omite.

- **Dependencias múltiples**: Una actividad dependiente de varias actividades sólo se ejecuta si cada actividad anterior satisface una condición de dependencia. Si una actividad especifica varias condiciones de dependencia de la misma actividad anterior, sólo es necesario que se cumpla una.
- **Actividad leaf**: Una actividad leaf es una actividad pipeline sin sucesores.
- **Actividades condicionales**: El ADF tiene dos actividades condicionales: la actividad If Condition y la actividad Switch.
- **Actividad If Condition**: Especifica una expresión que se evalúa como verdadera o falsa y dos conjuntos de actividades contenidas correspondientes. Cuando la actividad se ejecuta, su expresión se evalúa y el conjunto de actividades correspondiente se ejecuta.
- **Actividad Switch**: Especifica una expresión que se evalúa como un valor de cadena y hasta 25 conjuntos de actividades contenidos correspondientes. Cuando se ejecuta la actividad, se evalúa la expresión y se ejecuta el conjunto de actividades correspondiente, si lo hay. Si no se encuentra ningún caso coincidente, se ejecuta el conjunto de actividades por defecto.
- **Actividades de iteración**: ADF tiene dos actividades de iteración: la actividad ForEach y la actividad Until.

- **Actividad ForEach**: Especifica un array JSON y un conjunto de actividades a ejecutar una vez por cada elemento del array. El elemento actual del array se aborda en cada iteración utilizando la expresión item().

- **Paralelismo**: Por defecto, las ejecuciones de las actividades ForEach tienen lugar en paralelo, lo que requiere cuidado para asegurar que las actividades de iteraciones simultáneas no interfieran entre sí. Debe evitarse la modificación de variables, pero la actividad Execute Pipeline proporciona una forma sencilla de aislar las iteraciones. El ADF UX ejecuta las iteraciones ForEach secuencialmente en modo Debug, lo que puede hacer que los fallos de paralelismo sean difíciles de detectar en tiempo de desarrollo.

- **Actividad Until**: Especifica una condición de terminación y un conjunto de actividades que se ejecutarán repetidamente hasta que se cumpla la condición de terminación. Las actividades dentro de una actividad Until se ejecutan al menos una vez y nunca en paralelo.

- **Anidación**: Las actividades de iteración no pueden anidarse en otras actividades de iteración. Las actividades condicionales no pueden anidarse en otras actividades condicionales, aunque sí se permite el anidamiento en actividades de iteración. Una solución común es implementar las actividades internas y externas en pipelines separados, llamando a la actividad interna desde la externa a través de la actividad Execute Pipeline.

- **Puntos de interrupción (Breakpoints)**: El ADF UX no soporta puntos de interrupción dentro de las actividades de iteración o condicionales.

- **Actividad Get Metadata**: Devuelve los metadatos que describen los atributos de un dataset del ADF. No todos los atributos de metadatos son compatibles con todos los datasets: los destinos de dataset inexistentes provocarán errores a menos que se especifique el argumento "Exists"; especificar el argumento "Child Items" en un destino que no sea una carpeta hará que la actividad falle.

- **Tolerancia a los fallos (Fault tolerance)**: La actividad Copiar datos admite una gestión de errores mejorada a través de su configuración de tolerancia a fallos, lo que permite desviar filas de error individuales a un archivo de registro externo sin abandonar por completo una carga de datos.

- **Generación de errores**: En el momento de redactar el presente documento, el ADF no dispone de una actividad de "elevación de errores". Los enfoques disponibles para la fabricación de errores incluyen la definición de fundiciones de tipo ilegales en las actividades de variables Set o la explotación de la funcionalidad de elevación de errores en los servicios externos, por ejemplo, mediante el uso de las sentencias RAISERROR o THROW de SQL Server.