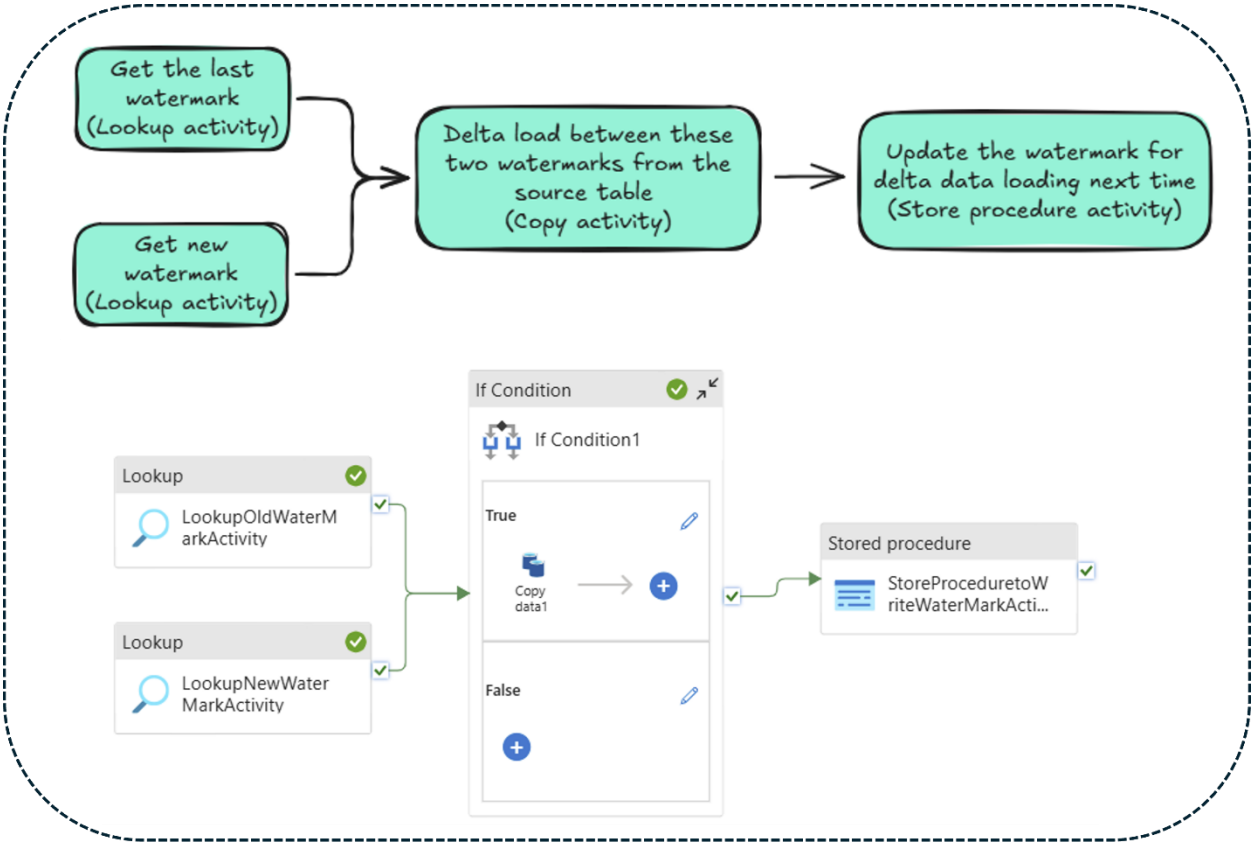
CARGA INCREMENTAL DE DATOS EN AZURE DATA FACTORY

Ing. José Carlos Reyes



17/01/2026

Carga incremental de datos en Azure Data Factory

Ing. José Carlos Reyes

Se describe enseguida el procedimiento a seguir para realizar una carga incremental de datos utilizando una herramienta de orquestación de datos de Azure, desde una base de datos en Sql-Server. El ejercicio está basado en el tutorial de Microsoft, para mayor detalle puede consultarlo en la liga: https://learn.microsoft.com/en-us/azure/data-factory/tutorial-incremental-copy-powershell

El objetico es comprender a detalle los pasos que el pipeline realiza, en cada una de sus actividades, inputs de entrada y salidas, así como realizar dos corridas del pipeline: 1 considerando los datos iniciales y una segunda al añadir datos a la fuente original. En la figura 1, se puede visualizar el diagrama del pipeline a desarrollar, el cual consiste de dos actividades Lookup para obtener la última y la nueva fecha de actualización, para la carga del delta de datos(datos nuevos no incluidos en la tabla objetivo o target) y un stored procedure para actualizar el nuevo valor de la última fecha de actualización.

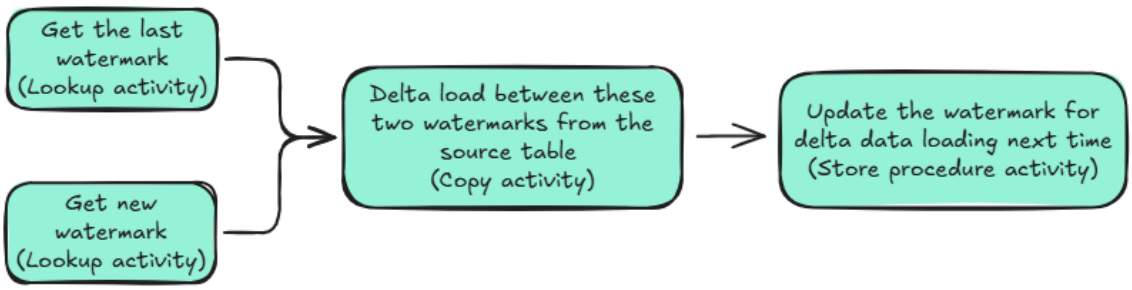


Figura 1.- Diagrama del pipeline de datos para la carga incremental desde la base de Sql-Server.

Escenario-1 o base

Partimos de lo que tenemos en nuestra base de datos de Sql-Server, en donde tenemos dos tablas una que almacena los datos de las personas(source_data_table) y una segunda que almacena la última fecha de actualización(watermarktable). Note que nuestra tabla de datos original cuenta con una columna para tener un rastreo de la fecha de modificación (columna “LastModifyDate”). Realizando una query en nuestra tabla source tenemos:

```
SELECT * FROM data_source_table
```

PersonID	Name	LastModifytime
1	aaaa	2017/09/01 00:56:00.000
2	bbbb	2017/09/02 05:23:00.000
3	cccc	2017/09/03 02:36:00.000
4	dddd	2017/09/04 03:21:00.000
5	eeee	2017/09/05 08:06:00.000

Tabla 1.- Datos almacenados en la tabla fuente.

Debemos definir una fecha inicial de última actualización, dentro de la watermark table. Realizando una query tenemos:

```
SELECT * FROM watermarktable
```

TableName	WatermarkValue
data_source_table	2010/01/01 12:00:00 AM

Para éste primer escenario, la actividad LookupOldWaterMarkActivity, ejecuta una consulta en la tabla watermarktable y obtiene la última fecha de actualización, como se muestra en la figura 2.

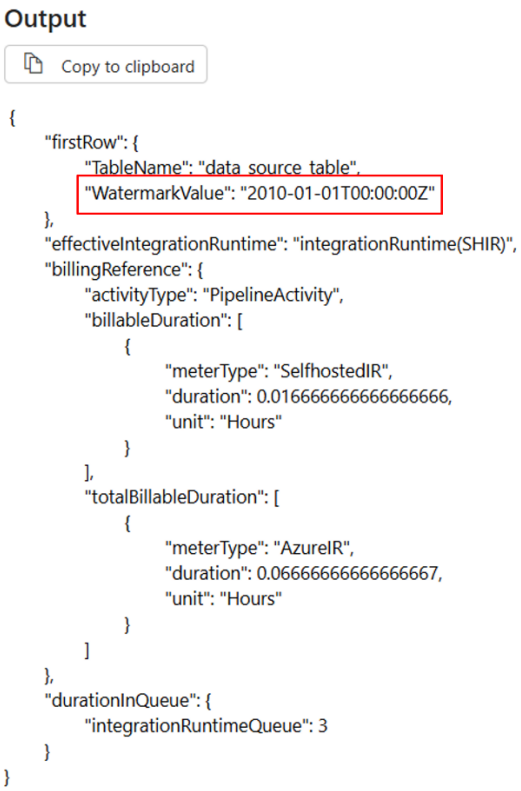


Figura 2.- Salida de la actividad 'LookupOldWaterMarkActivity' en la que se puede ver la fecha inicial de última actualización de la watermark_table, éste resultado será leído por la Copy Activity.

La actividad 'LookupOldWaterMarkActivity' ejecuta una consulta en la tabla fuente (data_source_table), obteniendo su última fecha de actualización, como resultado obtenemos el máximo valor de la columna LastModifytime (figura 3).

Figura 3.- Salida de la actividad 'LookupNewWaterMarkActivity' en la que se puede ver la fecha inicial de última actualización de la source_table, éste resultado será leído por la Copy Activity.

Para la CopyActivity, tenemos que el OldWatermarkValue equivale a "2010-01-01T00:00:00Z" mientras que la NewWatermarkVale equivale a 2017-09-05T08:06:00Z. La query que se realiza sobre la source table filtra los valores en que la columna 'LastModifytime' > OldWatermarkValue y 'LastModifytime' <= NewWatermarkValue, es decir el delta de datos por copiar esta entre valores > '2010-01-01T00:00:00Z' y <= '2017-09-05T08:06:00Z'. Para la primer corrida copiará los 5 valores dado que todos están por arriba de la fecha inicial '2010-01-01' y por debajo o iguales a la fecha '2017-09-05'. En la figura 4 podemos ver la query dentro de Sql-Server. Lo anterior podemos constatarlo en el archivo .txt guardado en el data lake de azure (figura 5).

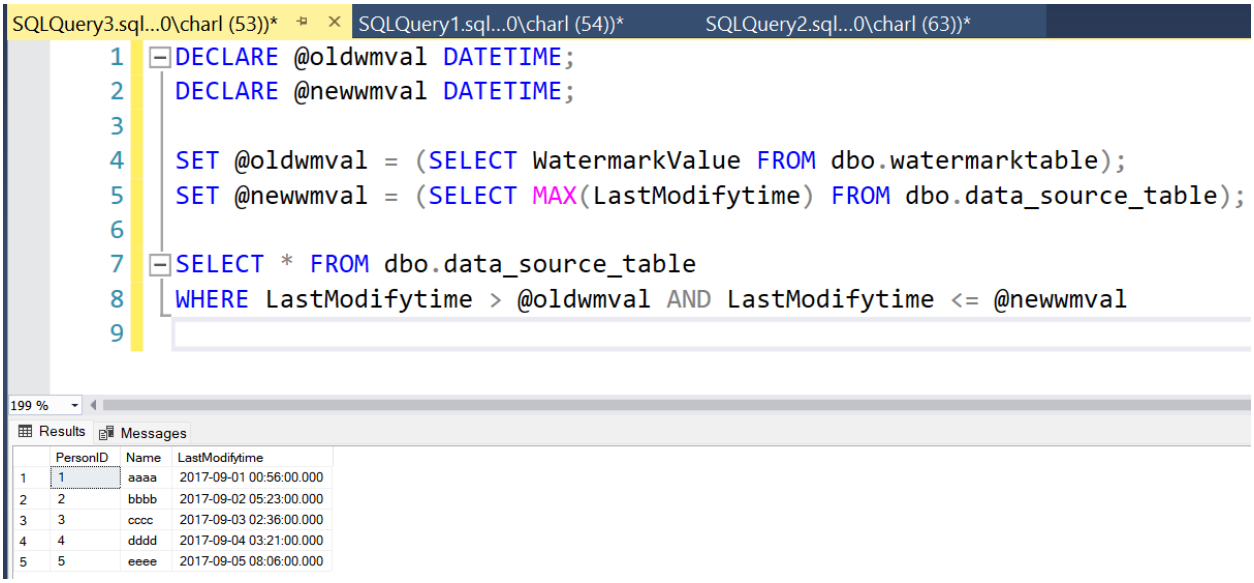


Figura 4.- Query que simula la copy activity dentro del pipeline en azure data factory.

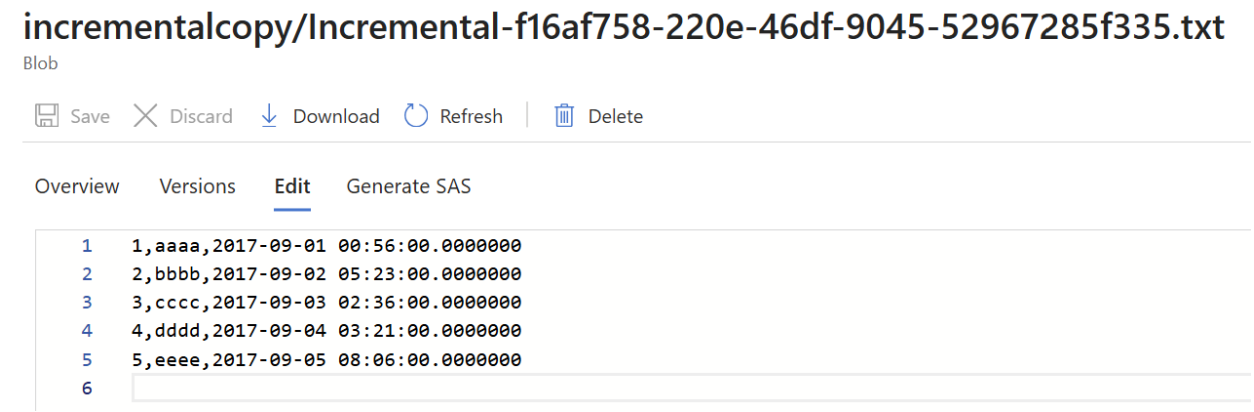


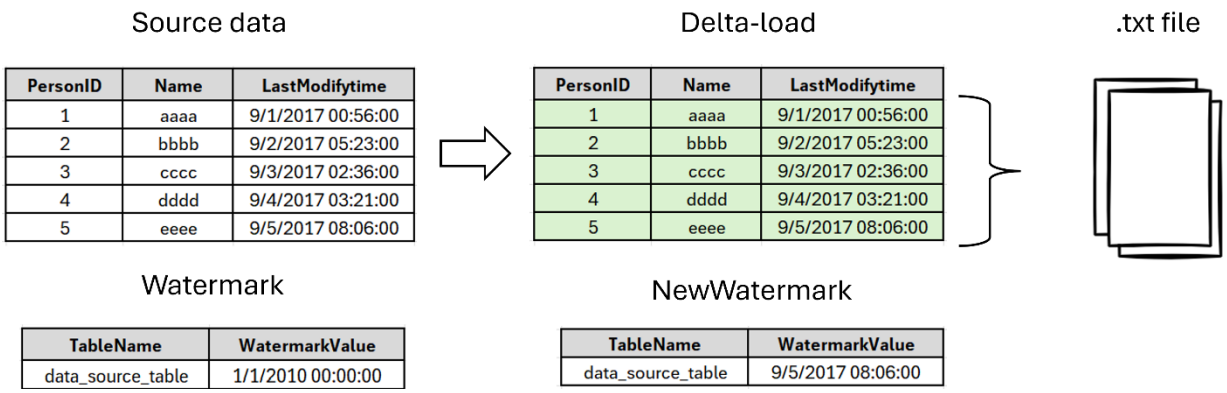
Figura 5.- Contenido del archive .txt dentro del data lake, confirmando que la correcta carga de los datos.

Finalmente, la actividad de stored procedure, realizará la ejecución del sp "usp_write_watermark", el cual, requiere 2 parámetros el @LastModifiedtime y el parámetro @TableName. El primero lo leerá del resultado de la 'LookupNewaterMarkActivity' como la última nueva fecha de actualización, en nuestro ejemplo "2017-09-05 08:06:00.00" y el nombre de la tabla lo leerá de la salida de la 'LookupOldwaterMarkActivity' que para nuestro caso es "data_source_table". Lo anterior podemos confirmarlo al realizar una query sobre la watermark table:

Figura 6.- Actualización de la fecha en la watermark table.

Ing. José Carlos Reyes

En la figura 7, podemos visualizar el proceso de la carga incremental de datos anterior, o inicial, en la cual se copian la totalidad de los datos dado que todos son mayores a la fecha watermark inicial (2010-01-01).



PersonID	Name	LastModifytime
1	aaaa	9/1/2017 00:56:00
2	bbbb	9/2/2017 05:23:00
3	cccc	9/3/2017 02:36:00
4	dddd	9/4/2017 03:21:00
5	eeee	9/5/2017 08:06:00

PersonID	Name	LastModifytime
1	aaaa	9/1/2017 00:56:00
2	bbbb	9/2/2017 05:23:00
3	cccc	9/3/2017 02:36:00
4	dddd	9/4/2017 03:21:00
5	eeee	9/5/2017 08:06:00

TableName	WatermarkValue
data_source_table	1/1/2010 00:00:00

TableName	WatermarkValue
data_source_table	9/5/2017 08:06:00

Figura 7.- Proceso ejecutado durante la primera corrida del pipeline de datos.

Si volviéramos a correr el pipeline, no habría ningún delta, dato que ahora ya no se tienen valores con fechas por arriba del día 05 de septiembre de 2017 y que sean menores e iguales a ésta misma fecha, por lo que al correrlo nos añadiría un archivo .txt en blanco (figura 7).

Aquí valdría a pena colocar un if, para en caso de no tener delta, no guardar el archivo que prácticamente está vacío.

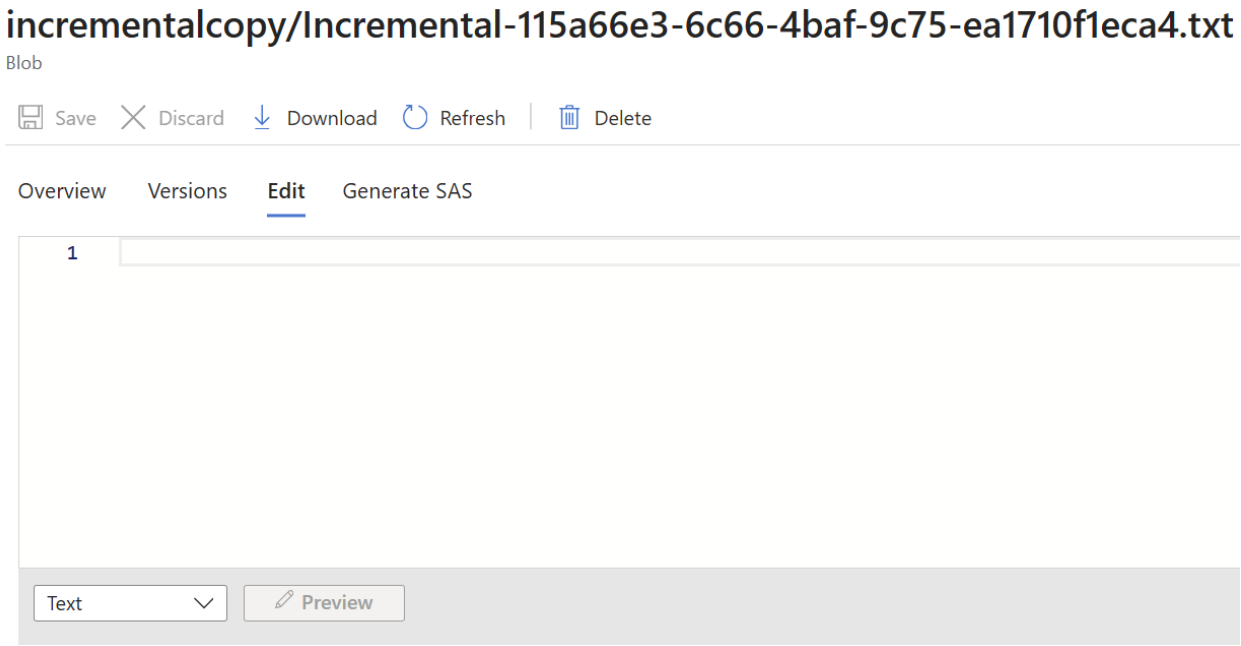


Figura 8.- Archivo vacío al aplicar una segunda corrida del pipeline sin delta o incremento en los datos.

Ing. José Carlos Reyes

Escenario-2.- Carga incremental de datos

Vamos a simular ahora la carga posterior de datos a nuestra base en Sql-Server. Dentro de SSMS ejecutamos la siguiente query para insertar datos.

```
INSERT INTO data_source_table

VALUES

(6, 'newdata','9/6/2017 2:23:00 AM'),

(7, 'newdata','9/7/2017 9:01:00 AM')
```

En la terminal ejecutamos el siguiente código para ejecutar el pipeline de datos.

```
> $resourceGroupName = "adworks-project-jc"

> $dataFactoryName = "adf-Inc-Copy-jc"

> $pipelineName = "IncrementalCopyPipeline"

> $RunId = Invoke-AzDataFactoryV2Pipeline -PipelineName $pipelineName -ResourceGroupName $resourceGroupName -dataFactoryName $dataFactoryName

> Get-AzDataFactoryV2ActivityRun -DataFactoryName $dataFactoryName -ResourceGroupName $resourceGroupName -PipelineRunId $RunId -RunStartedAfter "2026/01/17" -RunStartedBefore "2026/01/19"
```

La última línea nos muestra el status del pipeline en el cual podemos confirmar la correcta ejecución del pipeline como se muestra en las figuras 8 y 9.

```
PS C:\Users\charl> $resourceGroupName = "adworks-project-jc"
PS C:\Users\charl> $dataFactoryName = "adf-Inc-Copy-jc"
PS C:\Users\charl> $pipelineName = "IncrementalCopyPipeline"
PS C:\Users\charl> $RunId = Invoke-AzDataFactoryV2Pipeline -PipelineName $pipelineName -ResourceGroupName $resourceGroupName -dataFactoryName $dataFactoryName
PS C:\Users\charl> Get-AzDataFactoryV2ActivityRun -DataFactoryName $dataFactoryName -ResourceGroupName $resourceGroupName -PipelineRunId $RunId -RunStartedAfter "2026/01/1
7" -RunStartedBefore "2026/01/19"


ResourceGroupName    : adworks-project-jc
DataFactoryName      : adf-Inc-Copy-jc
ActivityRunId        : c18abbb9-8c16-41fd-8a26-eae1d960afd5
ActivityName         : LookupNewWaterMarkActivity
ActivityType         : Lookup
PipelineRunId        : 97d63dff-847e-4d34-8a0b-23e6835c0c12
PipelineName         : IncrementalCopyPipeline
Input                : {source, dataset}
Output               : {firstRow, effectiveIntegrationRuntime, billingReference, durationInQueue}
LinkedServiceName    :
ActivityRunStart     : 1/18/2026 6:19:47 PM
ActivityRunEnd       : 1/18/2026 6:20:01 PM
DurationInMs         : 13411
Status               : Succeeded
Error                : {errorCode, message, failureType, target...}
AdditionalProperties : {[retryAttempt, ], [iterationHash, ], [userProperties, {}], [recoveryStatus, None]...}

ResourceGroupName    : adworks-project-jc
DataFactoryName      : adf-Inc-Copy-jc
ActivityRunId        : 3b84b2bf-865e-40f1-9279-43e6e7f75e66
ActivityName         : LookupOldWaterMarkActivity
ActivityType         : Lookup
PipelineRunId        : 97d63dff-847e-4d34-8a0b-23e6835c0c12
PipelineName         : IncrementalCopyPipeline
Input                : {source, dataset}
Output               : {firstRow, effectiveIntegrationRuntime, billingReference, durationInQueue}
LinkedServiceName    :
ActivityRunStart     : 1/18/2026 6:19:47 PM
ActivityRunEnd       : 1/18/2026 6:20:00 PM
DurationInMs         : 13105
Status               : Succeeded
Error                : {errorCode, message, failureType, target...}
AdditionalProperties : {[retryAttempt, ], [iterationHash, ], [userProperties, {}], [recoveryStatus, None]...}
```

Figura 9.- Ejecución del pipeline de datos desde la terminal y resultados de la corrida.

Por último nos vamos a la carpeta dentro del data lake para asegurarnos de que se haya guardado el archivo con el delta de datos. En la figura 10, podemos observar que se tienen 2 archivos con el nombre de la corrida, la segunda corrida fue sin un delta, por lo que el archivo se encuentra vacío, valdría la pena añadirle meter la copy activity dentro de un if para que solo copie datos cuenda exista un delta y no cuando éste vacío. Además del archivo vació tenemos un tercer archivo el cual contiene los datos nuevos añadidos, confirmando que el pipeline funciona de manera correcta.

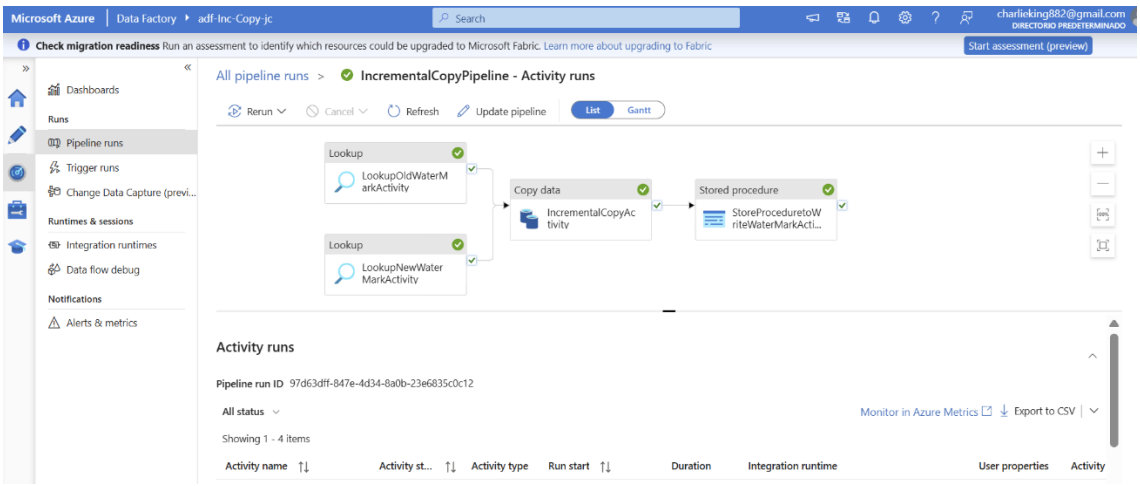


Figura 10.- Visualización del pipeline de datos en la pestaña monitor de Azure Data Factory, se observa la correcta ejecución de las actividades Lookup, Copy y Stored procedure.

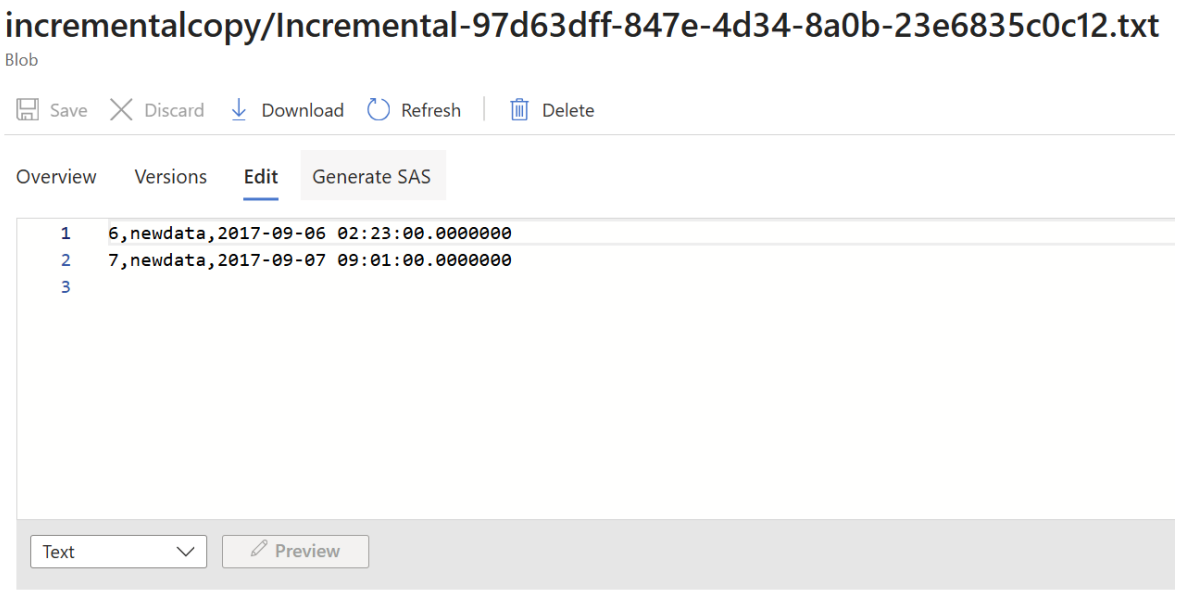
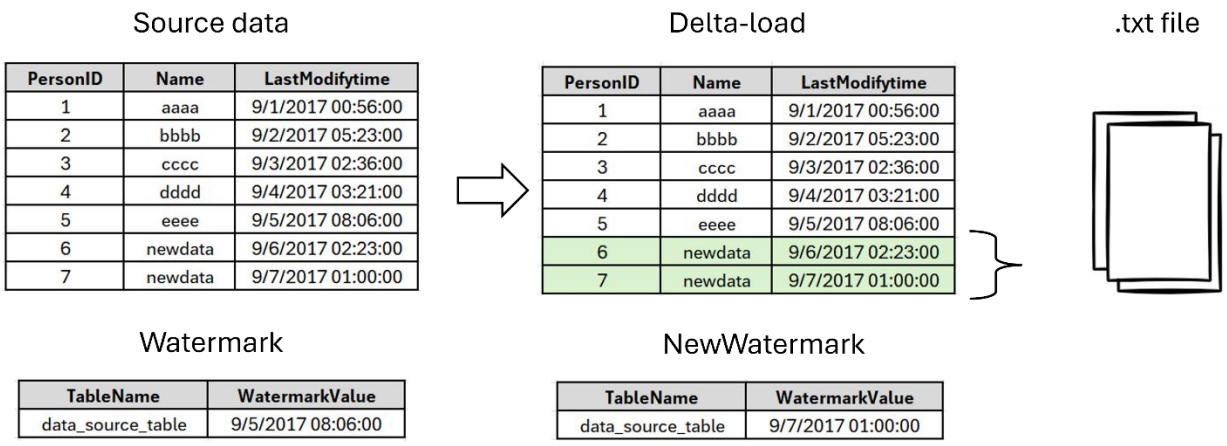


Figura 11.- Archivo dentro del data lake con los datos delta o incrementales.

En la figura 12 se visualiza el proceso ejecutado durante la segunda corrida del pipeline al añadir nuevos datos dentro de la tabla en Sql-Server, podemos ver de manera más clara los registros o filas que conforman el delta de datos o la carga incremental, los cuales se guardarán en el data lake como un archivo con extensión txt.



PersonID	Name	LastModifytime
1	aaaa	9/1/2017 00:56:00
2	bbbb	9/2/2017 05:23:00
3	cccc	9/3/2017 02:36:00
4	dddd	9/4/2017 03:21:00
5	eeee	9/5/2017 08:06:00
6	newdata	9/6/2017 02:23:00
7	newdata	9/7/2017 01:00:00

PersonID	Name	LastModifytime
1	aaaa	9/1/2017 00:56:00
2	bbbb	9/2/2017 05:23:00
3	cccc	9/3/2017 02:36:00
4	dddd	9/4/2017 03:21:00
5	eeee	9/5/2017 08:06:00
6	newdata	9/6/2017 02:23:00
7	newdata	9/7/2017 01:00:00

TableName	WatermarkValue
data_source_table	9/5/2017 08:06:00

TableName	WatermarkValue
data_source_table	9/7/2017 01:00:00

Figura 12.- Proceso ejecutado durante la segunda corrida del pipeline de datos, copiando solo el delta de datos.

Añadiendo el if-condition para añadir solo el incremental de datos

Para evitar generar archivos vacíos cuando no se tienen nuevo datos, vamos a incluir el copy activity dentro de un if-condition, el cual evalúa sí la fecha nueva de actualización (NewWaterMarkValue) es mayor que la anterior fecha de actualización (OldWaterMarkValue). En la figura 11 podemos visualizar la confición incluida dentro del if.

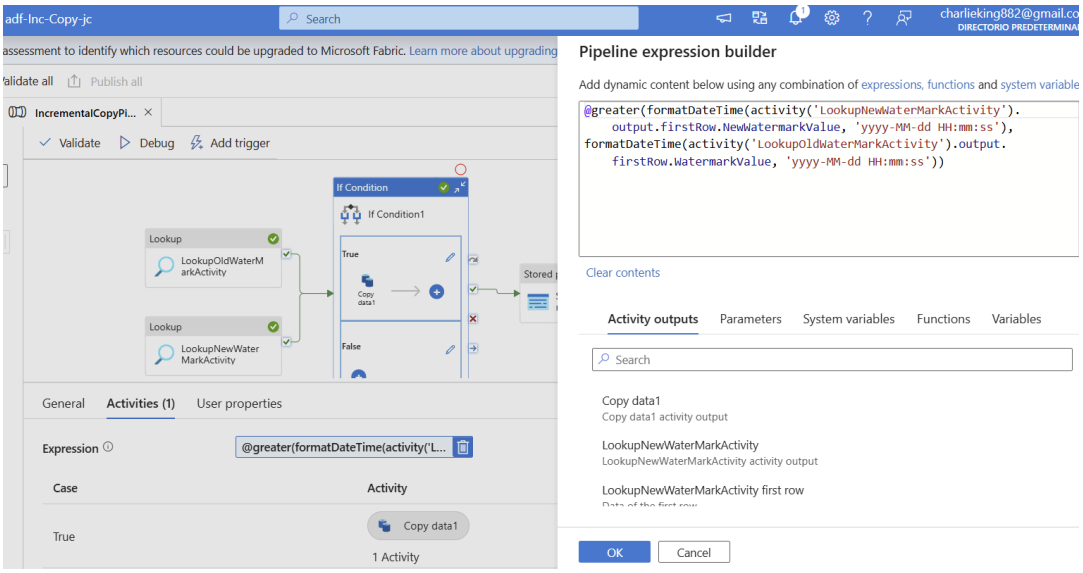


Figura 13.- Función @greater para realizar el comparativo entre la nueva y la anterior fecha de actualización de datos en el if-condition.

Dentro del if-condition colocamos el copy activity al cumplirse la condición, es decir True y en la parte de False podemos dejarlo sin actividad alguna. La figura 12 muestra la versión final del pipeline el cual solo va generando nuevos archivos al contar con nuevos registros dentro de la base de datos, es decir al tener cargas incrementales.

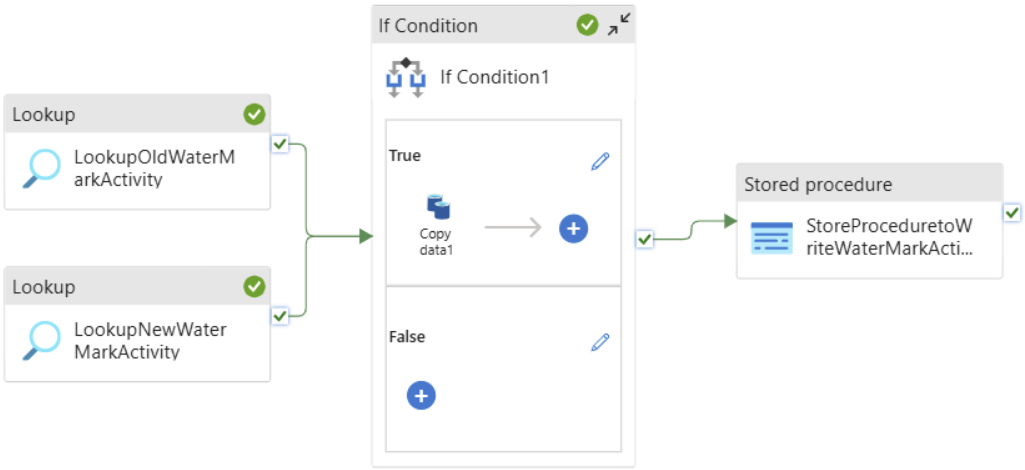


Figura 14.- Versión final del pipeline de datos para cargas incrementales desde Sql-Server.

Tras correrlo verificamos la actualización de la watermarktable con una query y confirmamos que el valor corresponde con la última fecha de actualización del registro más reciente dentro de la tabla fuente o data_source_table (figura 13).

Ing. José Carlos Reyes

Figura 15.- Último valor al añadir datos incrementales a la base, confirmando la correcta ejecución del pipeline y del stored procedure en Sql-server.

Conclusiones

Al realizar el presente tutorial se lograron diversos objetivos, como se enlistan a continuación:

1. Construcción de un pipeline de datos para cargas incrementales de datos con azure data factory y sql server, utilizando una columna 'LastModifytime' y una segunda tabla para contar con una trazabilidad de cada registro nuevo y añadir solo los nuevos.
2. De manera resumida, el pipeline consiste en 2 actividades lookup las cuales obtienen la fecha de modificación anterior y la nueva, evalúa si la nueva fecha es mayor que el último cambio y en caso de cumplirse copia los datos en el data lake como un archivo .txt. Por último una actividad stored-procedure ejecuta el procedure dentro de Sql-Server para actualizar con la nueva fecha la tabla de Watermark. La clave está en identificar los registros que han sido agregados posterior a la fecha utilizada como watermark y con fecha menor o igual a la fecha más reciente(figuras 7 y 12).
3. Al pipeline original, se le añadió una actividad de if-condition para aquellos casos en los que exista delta de datos o nuevos datos evite generar archivos .txt vacíos en el data lake.
4. Se utilizó la terminal y archivos .json para configurar los datasets, linked services y el pipeline de datos, el cual permite una mayor automatización durante la construcción del pipeline.
5. Actualmente existen otros métodos que simplifican las cargas incrementales como el Autoloader, COPY INTO o MERGE dentro de Databricks, sin embargo al realizarlos de manera manual uno puede comprender de mejor manera la lógica detrás del proceso.

Happy coding, JC.

Ing. José Carlos Reyes