

## Azure Data Factory - Part 4(a)

### Extract, Transform and Load

When you're extracting data from multiple sources, transforming it (filtering, sorting, aggregating, joining data, cleaning) and then loading the transformed data into the target location.

### Azure Data Factory Introduction –

This is a cloud based ETL tool. Can create data driven workflows which help orchestrate data movement. Also helps with transformation of the data.

The Azure Data Factory process looks somewhat like –

1. Connected to the required data sources
2. Ingest data from the source
3. Transform the data in the pipeline if required
4. Publish the data onto a destination – Azure Data Warehouse, Azure SQL Database, Cosmos DB
5. Can also monitor the pipeline as it is running

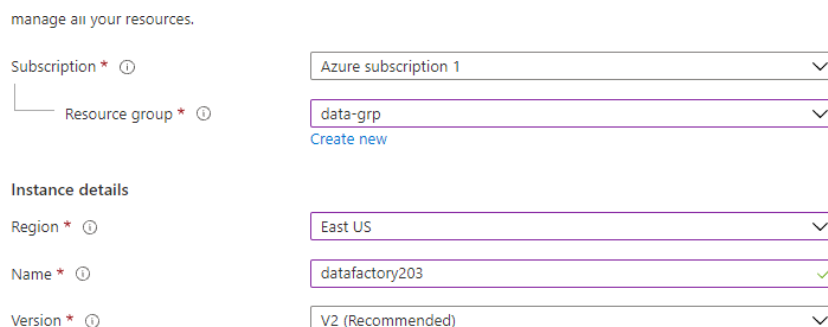
Azure Data Factory has different components –

1. Linked Service – This enables you to ingest data from a data source. The Linked Service can create the required compute resources to bring in the data from a data source.
2. Datasets – This represents the data structure within the data store that is being referenced by the linked service object
3. Activity – This is the actual transformation logic. You can also have simple copy activities to copy data from a source to a target.

In Azure Data Factory we would define our pipeline which is a logical grouping of activities in Data Factory. There could be many activities within a pipeline. For example, if you want to copy data from Azure SQL Database to Azure SQL Warehouse, you will have to create two linked services and one copy activity. Based on the requirements / activity there would be a compute infrastructure that would be managed by ADF.

### Getting started with Azure Data Factory – Copying CSV files from your data lake to the dedicated sql file

In your resource select Data Factory and hit create. Give your subscription, resource group, *unique* factory name and version.



The screenshot shows the 'Create new' form for an Azure Data Factory. It includes the following fields and values:

- Subscription \***: Azure subscription 1
- Resource group \***: data-grp (with a 'Create new' link below it)
- Instance details**
  - Region \***: East US
  - Name \***: datafactory203 (with a green checkmark)
  - Version \***: V2 (Recommended)

On the next screen select "Configure git later". Leave everything as it is and hit review and create.

Once Data Factory has been created, go to ADF Studio.

Now we would see how to copy our Log.csv file onto a table in our dedicated SQL pool from our data lake. On the ADS studio when you click on Ingest, it would open the Copy Data Tool that we saw in Azure Synapse Studio. Delete the contents of the LogData table in your dedicated sql pool and then on Copy Data tool –

- On the starting screen we select run once now
- We create a new connection for our source –
  1. We select Azure > Azure Data Lake Storage Gen2
  2. Give a name, select your subscription, select your data lake gen2 account name,
  3. Test the connection and create
- Once the connection is created, we browse for our file, remove the recursively copy option, hit next.
- Most of the settings have been auto detected like file format, column delimiter, hit next.
- Create a new connection for target. Choose Azure > Azure Synapse Analytics
  1. Give name, choose subscription, choose synapse workspace and database. For authentication, give the sql admin username for synapse that was created and the password.
  2. Test the connection and hit create.
- On the next screen select an existing table, the dbo.logdata table.
- On the next screen we see the type of the columns when they're being read and the type of the columns when they're being written onto the table.
- Add a storage account for staging area, if we try to proceed now it would give us an error because we have selected Polybase.
- To fix it, we go back and deselect type conversion and give the types of the incoming columns:
  1. Id – int32
  2. Time – Datetime
- Finish.

The table is populated in the dedicated sql pool now.

### Copying parquet data into dedicated sql pool table

Drop the contents of the logdata table.

*Side note* – in the Author section of the ADF, when you click on your pipeline, you can see all the activities part of that pipeline. Along with this each activity would have its source and target datasets which can also be viewed and reused if the dataset specifications of another activity match with this one.

Let's create a new pipeline from the Author section, click on the plus button and select pipeline.

- On the right, name the pipeline
- From the move and transform section, drag the copy data activity onto the canvas
  1. Name this activity
  2. Go to the source tab and create a new connection for the parquet files.
  3. Browse to the parquet directory after adding the Data lake. Click okay
  4. After returning to the source tab select wildcard file path and it would automatically add \*.parquet to the path
  5. Now we can open the dataset using the open button and import that schema from the schema tab and select import the schema from connection store
  6. In the sink tab we can select DestinationDataset of the previously run pipeline, give the copy method as polybase
  7. In mapping click on import schemas
  8. In settings click on enable staging and select you synapse container
- Click on Validate and it would validate the pipeline for any errors
- Hit on publish at the top
- Click on Add Trigger > Trigger now
- Head over to the SSMS to see the data in the dedicated sql pool

Sidenote – if you're not using a git repo, then you may not be able to save your pipeline at every step so you can save it ( publish it ) after it has been validated.

## Dealing with escape sequences in files

We are uploading the log.csv file but with escape characters in it. [Here](#) is the file.

We then go to ADF > Ingest to create a quick activity.

- In the source, select the previously made connection and browse and select the uploaded file
- In the next screen, if we preview the data, we can see the resource column which has the escape characters is being detected properly. *This is happening because in advanced setting on the same screen, under escape character you can specify whether it is a backslash or a forward slash or some other character or there are no escape characters at all.*
- Now go to your SSMS and drop the logdata table and recreate it with the extra column using the script –

```
CREATE TABLE [logdata]
(
    [Id] [int] ,
    [Correlationid] [varchar](200) ,
    [Operationname] [varchar](200),
    [Status] [varchar](100) ,
    [Eventcategory] [varchar](100),
    [Level] [varchar](100),
    [Time] [datetime] ,
    [Subscription] [varchar](200) ,
    [Eventinitiatedby] [varchar](1000),
    [Resourcetype] [varchar](1000) ,
    [Resourcegroup] [varchar](1000),
    [Resource] [varchar] (4000)
)
```

- Back on the Azure Data factory go to the next screen, select AzureSynapseAnalytics connection that we previously had, and select existing table > dbo.logdata
- On the next screen check the mapping, disable type conversion, give int to Id and DateTime to time columns and click next
- Next, name the pipeline and give the staging path
- Finish

Now we can go to the SSMS and query the logdata table in our dedicated sql pool to see that the Resource column has been loaded correctly.

## Generating Parquet based files

Begin by going to your Data lake gen2 and create a new directory – newparquet.

In ADF, in the author section, create a new pipeline –

- Give a name
- We'll drag and drop the copy activity onto the canvas
- Give the Copy Activity a name
- In the source tab use the data that was pointing to the Log.csv pipeline. We can also verify that it is the one by clicking on preview data. Uncheck the recursive option
- In sink, we need to create a new connection as we are creating a Parquet based file
- Select the Azure Data Lake Gen2 > Select the format of the file > Name it > select our data lake storage in linked service > browse to the newparquet folder. Select none for import schema
- In the mapping tab, click import schemas, fix the data type of the incoming columns (ID, Time)
- In settings, since we're not copying data to synapse, we would not need to enable any sort of staging area
- Publish and trigger pipeline

In our Data Lake the new parquet file will be present.

## Using a query for data transfer

Transferring data from the sql database to synapse. Go to your dedicated sql pool and delete the data from the SalesFact table. Going to ADF and selecting ingest.

- For source, Create a new connection > Azure > Azure SQL Database
- Give a name, Select Subscription, select server name, select database adventureworks, enter credentials and create
- Instead of choosing the existing tables option, we select the use query option and use the following query that we had used earlier to create the SalesFact table –

```
SELECT dt.[ProductID],dt.[SalesOrderID],dt.[OrderQty],dt.[UnitPrice],hd.[OrderDate],hd.[CustomerID],hd.[TaxAmt]
FROM [Sales].[SalesOrderDetail] dt
LEFT JOIN [Sales].[SalesOrderHeader] hd
ON dt.[SalesOrderID]=hd.[SalesOrderID]
```

- You can select the preview data button and it would show the result of your query if the query is working, click next
- Select the Azure Synapse Connection, use existing table, dbo.salesfact
- Check the column mapping in the next screen
- Next, give the name and the staging area location
- Finish

Check the SalesFact table in the Dedicated SQL pool and the data should be present.