|  |  |  |
| --- | --- | --- |
|  |  | Azure Data Ecosystem  Automating ELT with Data Bricks, Data Lake & Data Factory to SQL DB/DW. |

# Description

Most of the data movement and data migration requires a lot of manual intervention and it requires a lot of engineering effort in developing and designing the solutions which can handle data from multiple sources and the data can be of any type . This tutorial considers automating 90% of the requirement of ELT / ETL from any sources and it can be implemented for any requirement.



## Prerequisites

To apply the logic and understanding of the implementation the audience are requested to know concepts of Data Lake, Data Factory.

## Software:

1. Azure Subscription
2. Data Lake (Provided documentation for creation of installation)
3. Database (Provided documentation for creation of installation)
4. Data Factory (Provided documentation for creation of installation)
5. Azure Blob

## Requirement:

Automating ELT & ETL jobs would make the whole process very easier and would not require mundane work to be done on a daily basis. This document showcases below steps as exercises:

1. Creating Metadata Table and accessing from Data Factory.
2. Copy data from Blob to Data Lake (Landing).
3. Creating Zones in Data Lake (Process, Output Folders)
4. Dynamically Passing parameters from Data Factory.
5. Creating Automated SQL DB Table Insertion for Stage and Main Tables.
6. Automation of SQL DB Stored Procedure Calls.

## Approach:

The main backbone of the whole process would be a metadata table which would contain information about the data lake path, Blob Path and many more metadata information.

**Architecture:**

The architectural diagram shows the approach of handling any source with this approach in the below diagram, but we consider source as blob storage:

A screenshot of a cell phone

Description automatically generated

## Creating Metadata Table and Accessing from Data Factory.

In the folder we have scripts folder where we have scripts file (DBScripts)to execute . Execute each script inside it

A screenshot of a cell phone

Description automatically generated

After execution of scripts, Insert the TablesToProcessData.csv file data into Tablestoprocess table which is created from above scripts.

This table acts as a heart of the whole automation where we keep adding the new files and make the complete ETL automated and without touching the Pipelines.

Creating Required Setup:

1. Provided documentation for creation of Data Lake, Data Factory & SQL DB. Please follow each document to complete the prerequisites.
2. Once all the basic steps are done as above, we can concentrate on data movement automation.

**Copying Data from Blob to Data Lake:**

1. **Open the Data Factory and create a Linked Service (Defining the Source and Destination Connections) in connections section below.**

**A screenshot of a cell phone

Description automatically generated**

1. Add connections for Blob Storage (Before adding create the blob container and the folder path) as below:

A screenshot of a cell phone

Description automatically generated

Create a folder in azure blob container as employee and upload the Employee.csv file provided in the attachment.

1. Make sure the prerequisites have been completed for connecting Data Lake with Data Factory in the document provided in prerequisites folder.
2. Create a Linked service for Data Lake as well.
3. Create a Pipeline as below and name it as ‘AUTOMATED\_ETL\_PIPE’

A screenshot of a cell phone

Description automatically generated

1. Drag and Drop a Lookup Activity into the Pipeline Pane as below :

A screenshot of a cell phone

Description automatically generated

1. Create a Dataset for the TabletoProcess Table in Azure Data Factory and map it with Linked Service of SQL DB.

A screenshot of a cell phone

Description automatically generated

1. **Once Dataset for TablestoProcess is created map the dataset to Lookup Activity in pipeline and then past the below code under query in lookup :**

Select Id,TableName,IsActive,S3Path,RAWPath,NotebookPath,OUTPUTPATH,S3FilePath,

Load,Expression,KeyValue,JoinColumn,cast(year(getdate()) as varchar(10))+'/'+substring(convert(nvarchar(100),getutcdate(),121),6,2)+'/'+substring(convert(nvarchar(100),getutcdate(),121),9,2) as DatePath,getdate() as StartDateTime

from TablesToProcess where source = 'EFFEX'and isactive =1

A screenshot of a social media post

Description automatically generated

1. Click on preview data button next to +New link. It should show the Employee Information.
2. Publish the Pipeline and make sure no issues have occurred.
3. Drag and Drop the ForEach Activity into Pipeline and go to settings tab to pass the output of lookup into the Foreach activity as a list.

A screenshot of a cell phone

Description automatically generated

1. Double click on the For Each Activity and Drag and Drop the Copy activity into the foreach activity as below :

A screenshot of a cell phone

Description automatically generated

1. Create a Source Dataset for Blob in Azure Data Factory.

A screenshot of a social media post

Description automatically generated

1. After creating the dataset for Source Blob . Add dynamic content in Filepath and update as above for dynamically calling the parameters .
2. Same as above create new Dataset for Data Lake as below and update the filepath with dynamic content as below :

A screenshot of a cell phone

Description automatically generated

1. Add the Blob Dataset and Data Lake Dataset in copy activity as source and sink as below:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Publish the pipeline and trigger the flow to confirm if the flow is as expected . This will take the data file from Blob and copy it into the Data Lake .

A screenshot of a cell phone

Description automatically generated

Monitoring the pipeline :

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated