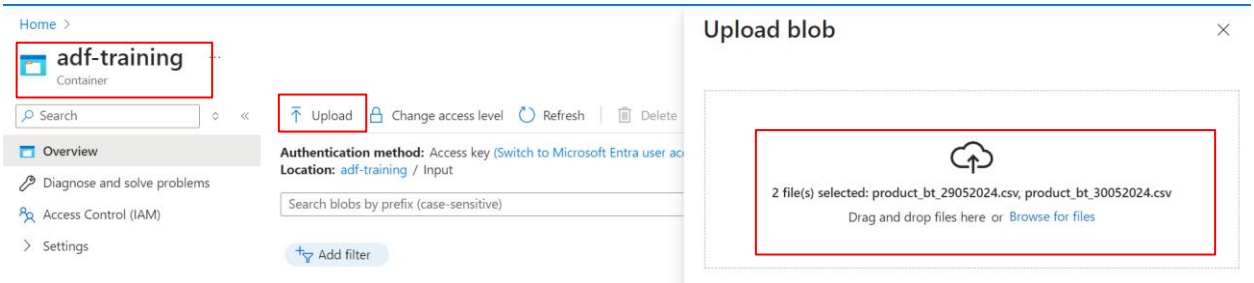
Formal Documentation of Azure Data Factory Pipeline – Training

Use Case: Loading Multiple Files Together in the Output Folder

Steps:

1. Loading the File into the Container.

- A container named adf-training was previously created in the Azure storage account.
- For this pipeline, the files were uploaded into the container, in the input folder (refer to image 1)
- File names: product_bt_29052024.csv, product_bt_30052024.csv
- Once the upload was successful, the files were added to the input folder.

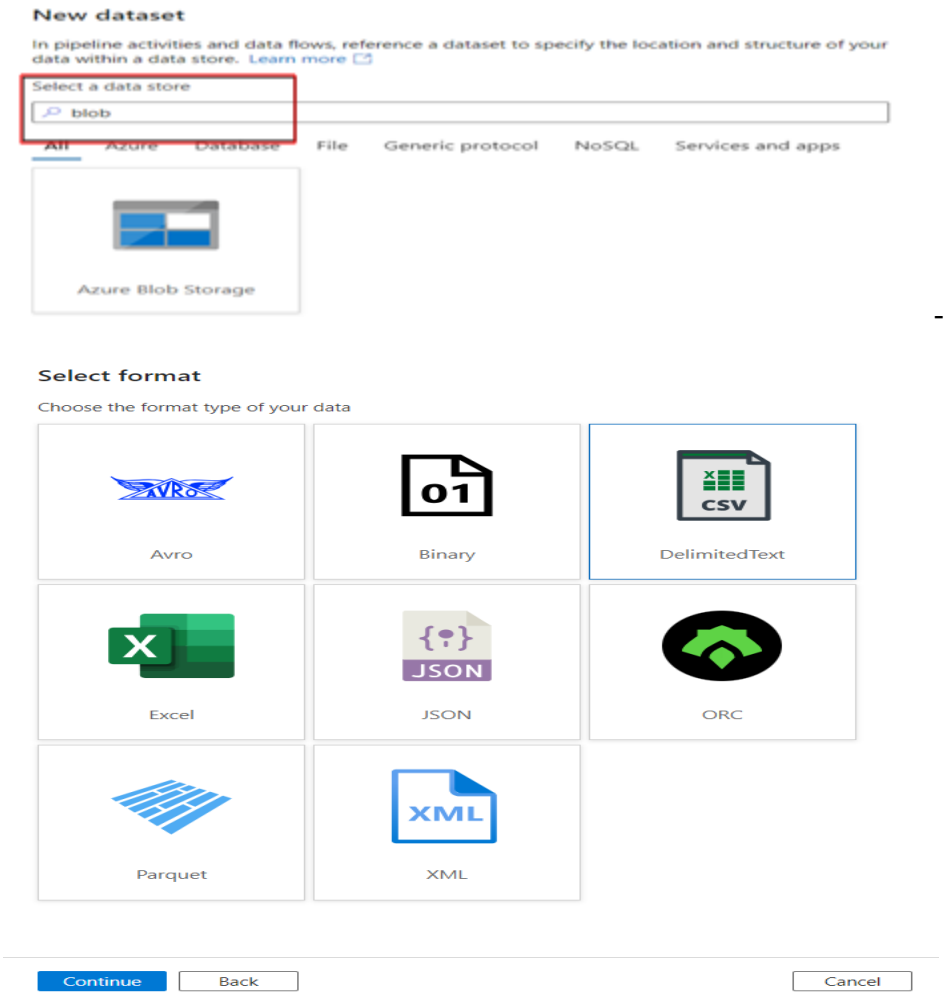


2. Dataset creation.

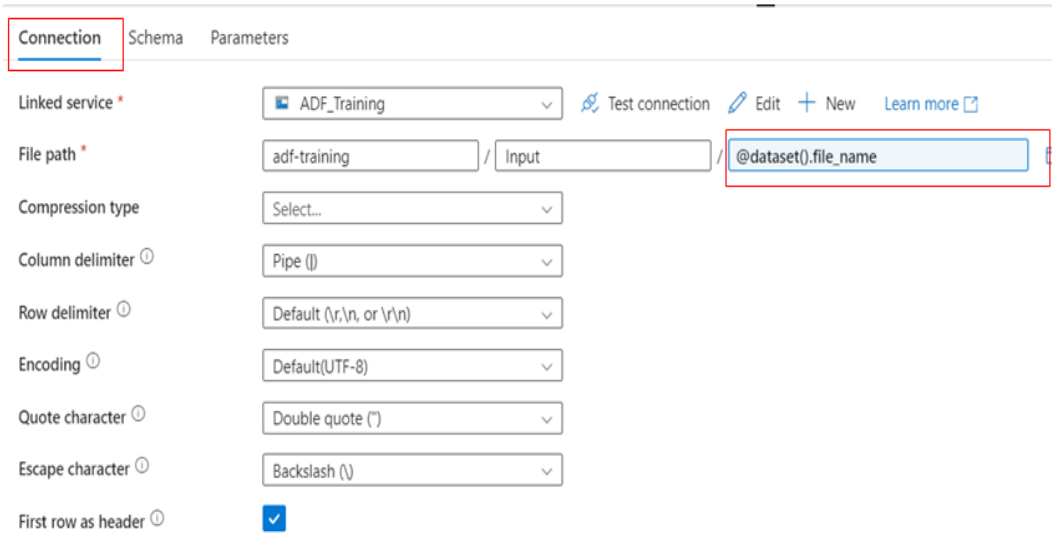
Source Dataset

- A new source dataset was created for the source data files.
- Source dataset name: bt_ds_src_ff_multi_product
- Inside Azure Data Factory, in the Author tab, I selected the Dataset option and clicked on "New Dataset" (refer to image 1).
- I chose the Azure Blob Storage option (refer to image 2,3). Next, I selected the Delimited Text file format, which brought me to the properties page where I defined the dataset name and path
- I specified the dataset name, selected the linked service, and provided the path of my input file.
- These steps created my source dataset.
- I opened my source file and updated the connection, changing the column delimiter option to pipe (|) because my CSV file is pipe delimited.
- Additionally, we will assign the file name in file path using dynamic content.
- The reason for using the dynamic function @dataset().file_name in the file path of the source file is to make the file path dynamic and dependent on the dataset properties(refer image 4)
- I selected the dynamic content option under the file name/beside Input and was able to get @dataset().file_name(refer image 5)
- Also added parameters, using parameters in the source and sink tabs of the Copy activity makes the file path dynamic and dependent on the dataset properties(refer image 6)

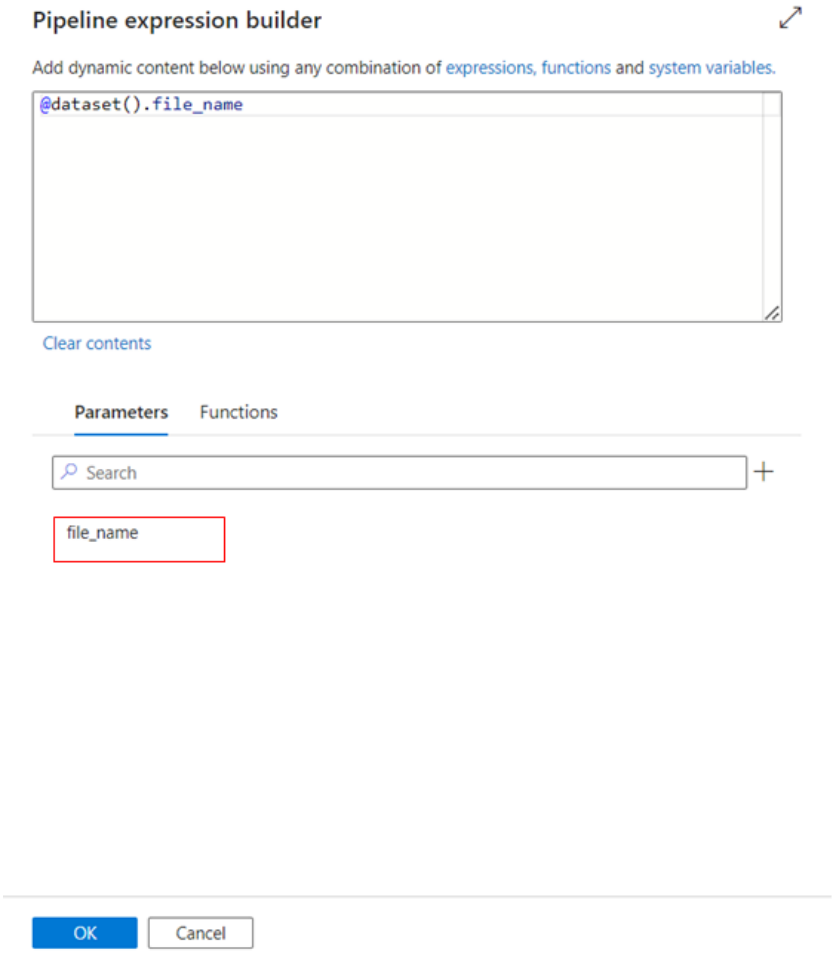
- Image 1



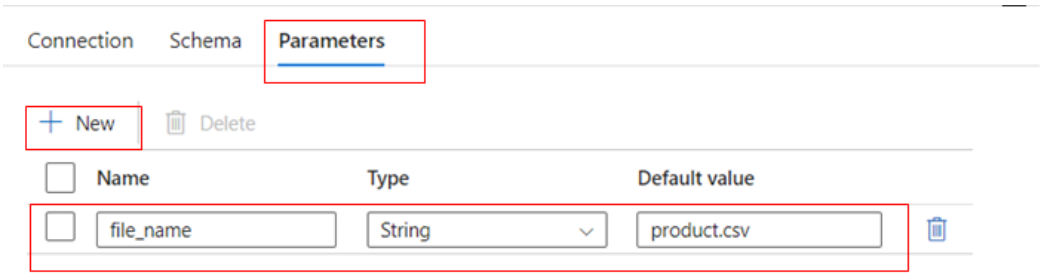
- Image 2



-Image 4



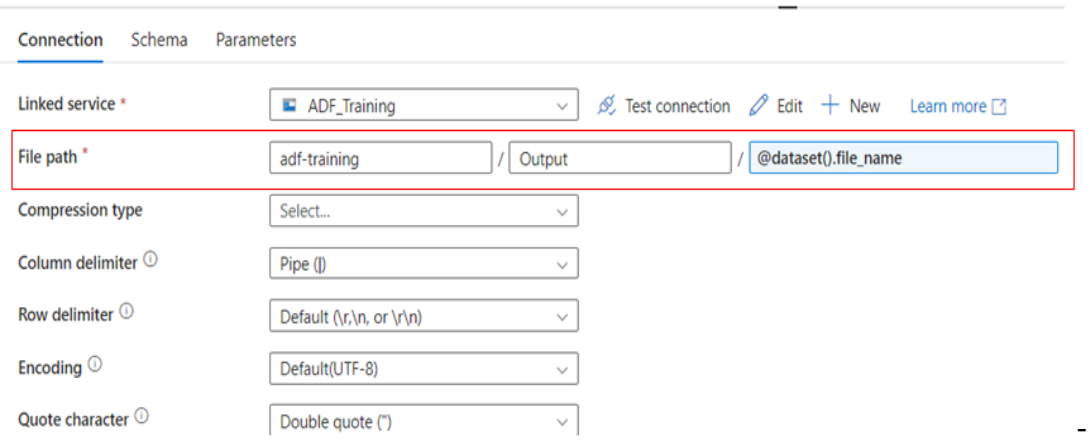
- Image 5



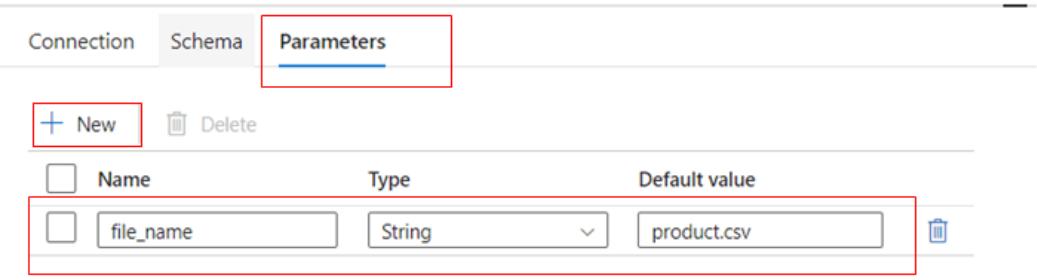
-Image 6

Target Dataset

- I followed similar steps for the target dataset. In the Author tab, I selected the Datasets option, clicked on "New Dataset," selected Azure Blob Storage, and then selected the Delimited Text format, which brought me to the properties page where I defined the dataset name and path.
- Assigned path: adf-training/output
- Target dataset name: bt_ds_tgt_ff_multi_product.
- I opened my sink file and updated the connection, changing the column delimiter option to pipe (|) because my CSV file is pipe delimited. All other options remained unchanged.
- Additionally, we will assign the file name in file path using dynamic content.
- The reason for using the dynamic function @dataset().file_name in the file path of the source file is to make the file path dynamic and dependent on the dataset properties.
- I selected the dynamic content option under the file name/beside Output and was able to get @dataset().file_name(same steps as source)(refer image 1)
- Using parameters in the source and sink tabs of the Copy activity makes the file path dynamic and dependent on the dataset properties(refer image 2).



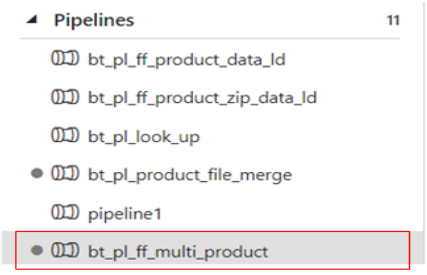
-Image 1



- Image 2

3. Pipeline Creation

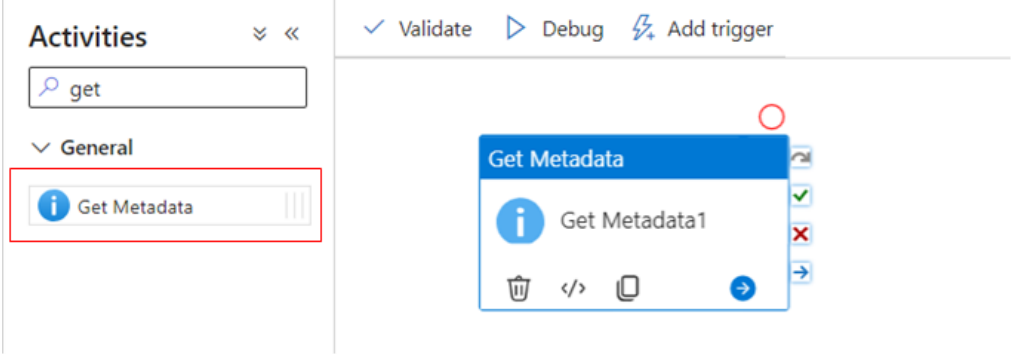
- A new pipeline named bt_pl_ff_multi_product was created in Azure Data Factory (ADF) to load the multiple file in the output folder(refer image 1).

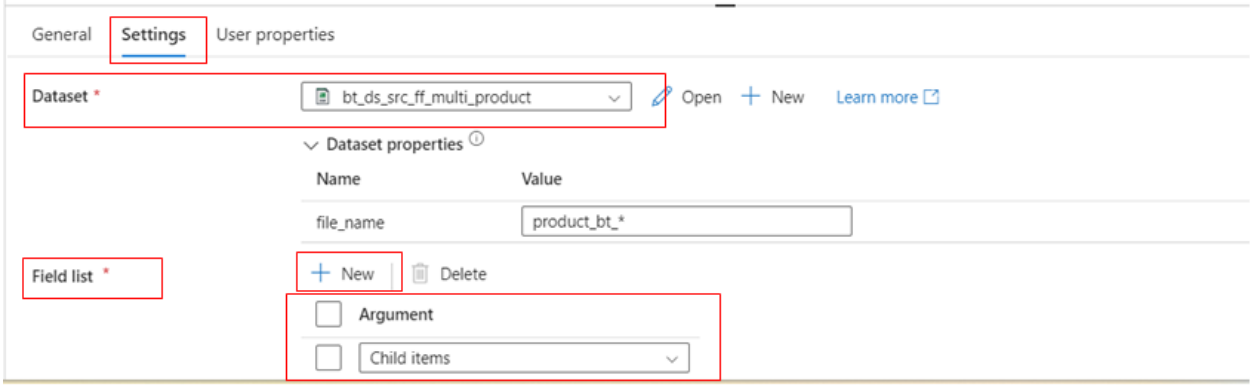


- Image 1

4. Get Metadata Activity

- I used the Get Metadata activity to get all the files/subfolders in the given path (refer image 1).
- I selected the source file and gave a common expression for the file name as “product_bt” since this is the common name in both files.
- In the field list, I selected child items. Child items are used to get a list of subfolders and files in the given folder (refer image 1,2)

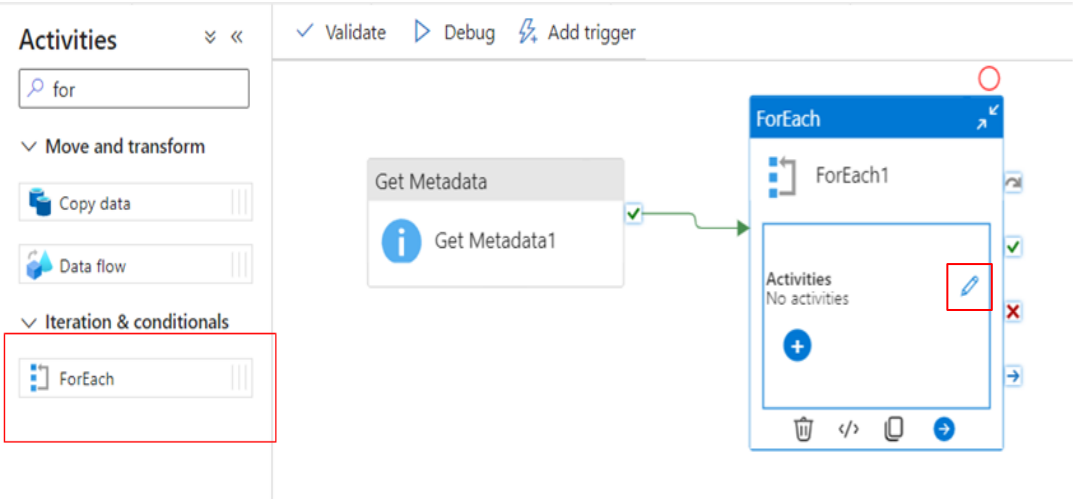




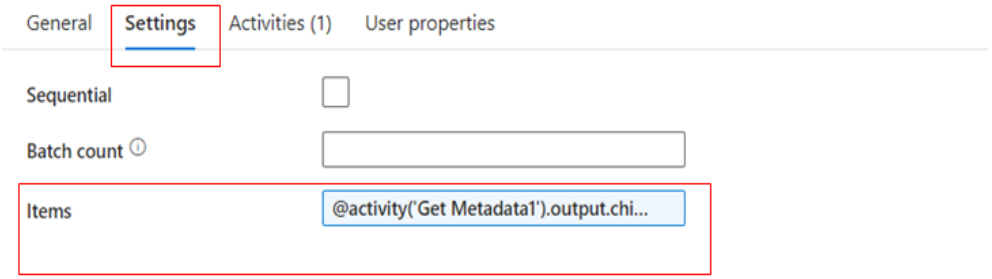
5. For Each Activity

- The For Each activity is used to iterate over a collection and execute specified activities in a loop (refer image 1)
- In the settings options, I disabled sequential as I copied the data parallelly. In parallel copying, “For Each” will copy everything to the output concurrently.
- If we select sequential, it will pass one by one; once the first file is copied successfully, it then prints the second file.

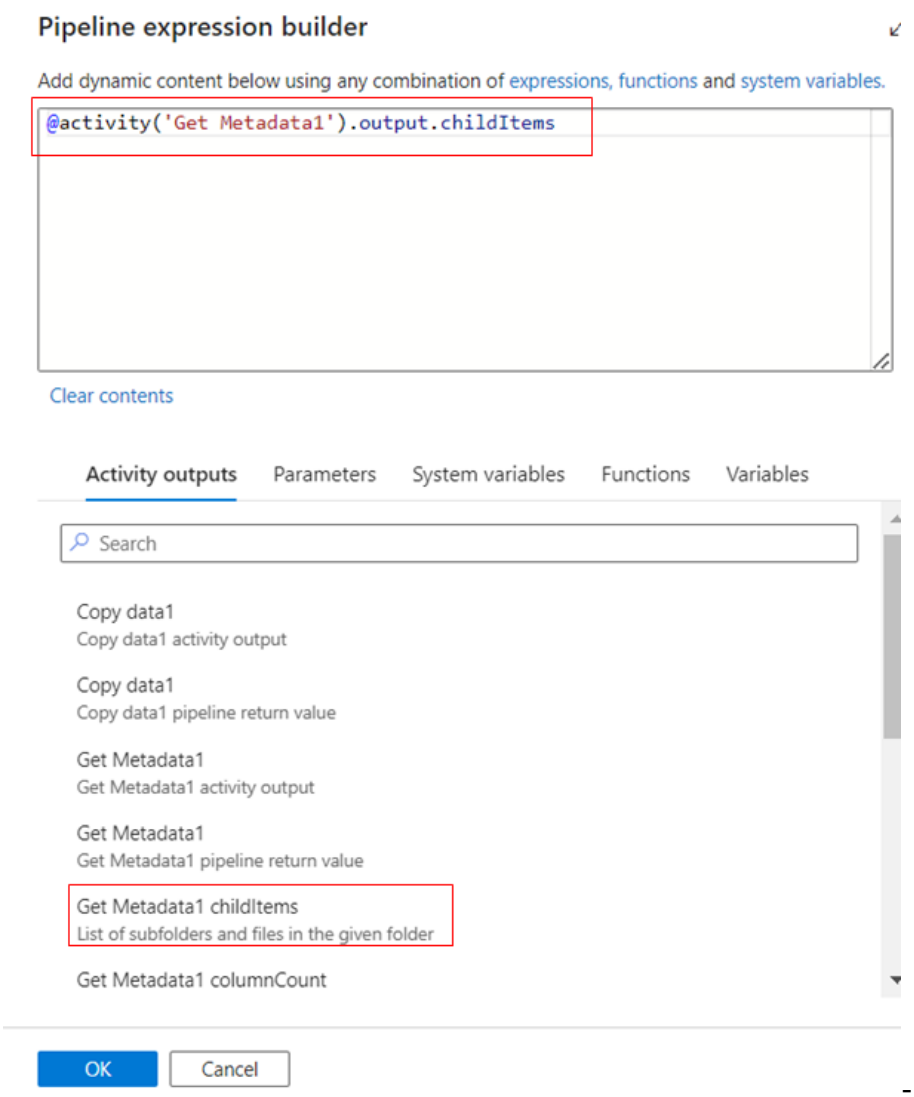
- I also did not select batch count here. Batch count is used for controlling the number of parallel executions (refer image 2).
- In items, I clicked on dynamic expression, selected Get Metadata1 child items option, and got @activity('Get Metadata1').output.childitems (refer image 3).
- Here's a breakdown of the formula:
  - @activity('Get Metadata1'): This refers to the output of an activity named "Get Metadata1". The @activity function is used to access the output of a previous activity.
  - .output: This accesses the output of the "Get Metadata1" activity.
  - .childitems: This accesses the child items of the folder. The childitems property is an array of child items, which can be files or subfolders.
- Next, I clicked on edit for each activity and dragged the copy data activity into it (refer image 4)
- I added my source file in the source and in dataset properties, I assigned a value to the file name (refer image 5) I selected "add dynamic content" and inside it selected ForEach1 option, which gave me an outcome of @item().name.
- The @item().name is used in the For Each activity in Azure Data Factory to access the name property of each item in the array being iterated over (refer image 6).
- Then I went to sink and added the sink file path and added dataset properties same as the source (refer image 7).



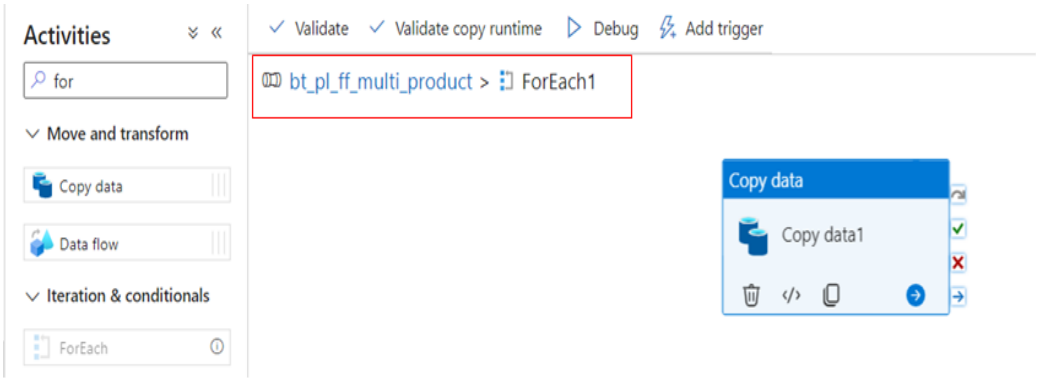
- Image 1



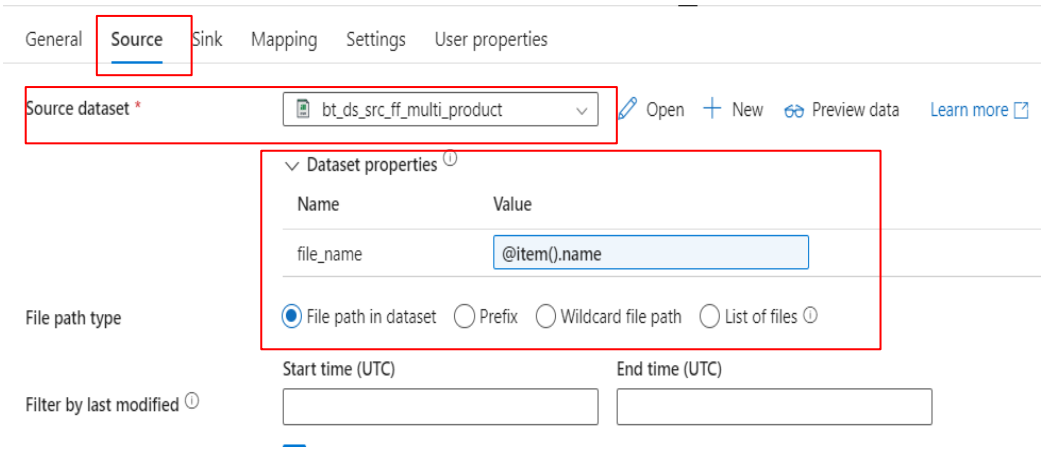
- Image 2



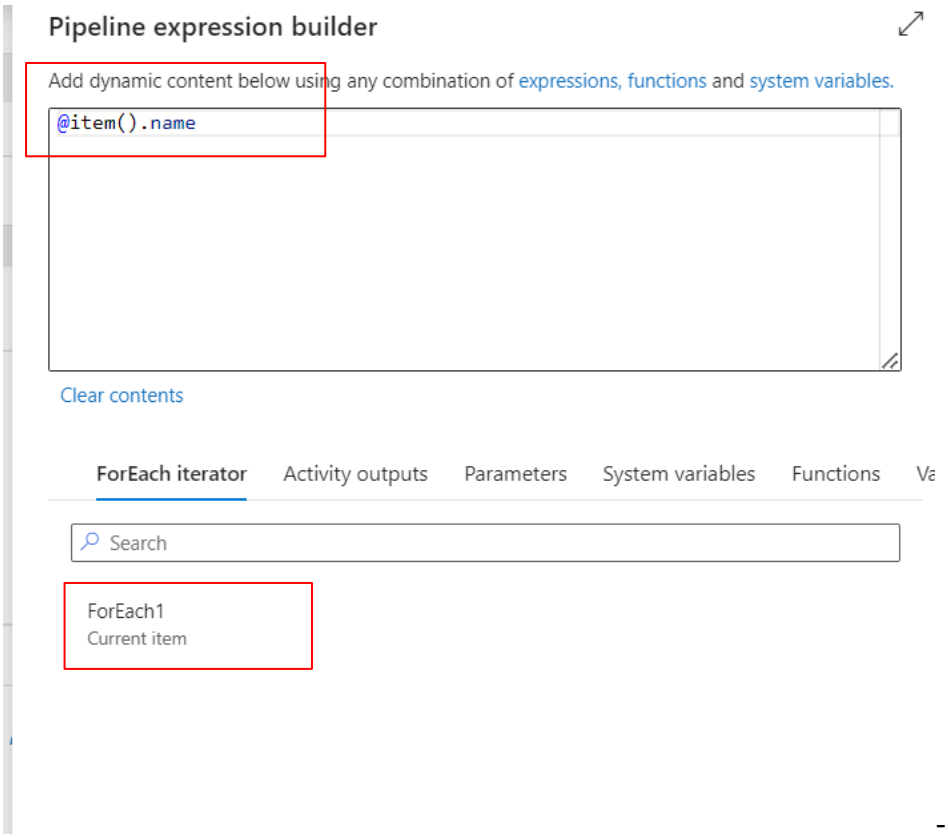
- Image 3



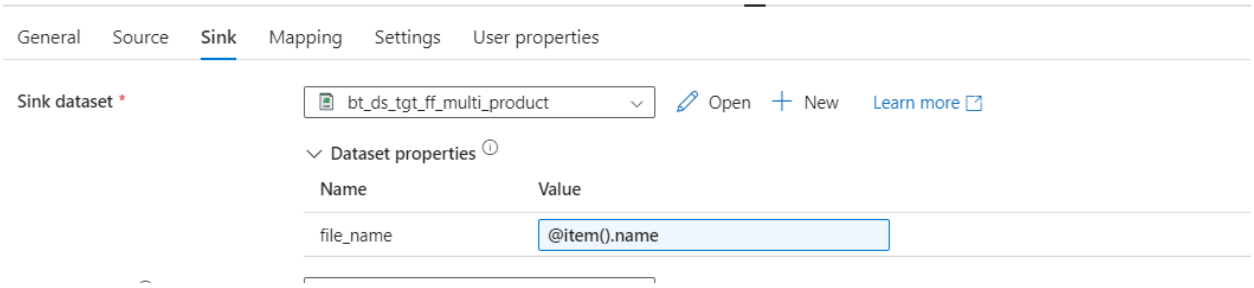
- Image 4



- Image 5



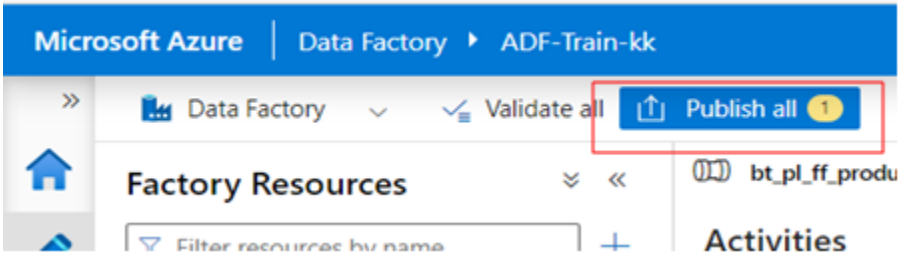
-Image 6



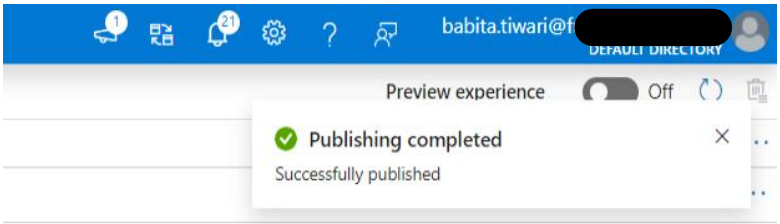
- Image 7.

6. Publishing and Executing the Pipeline

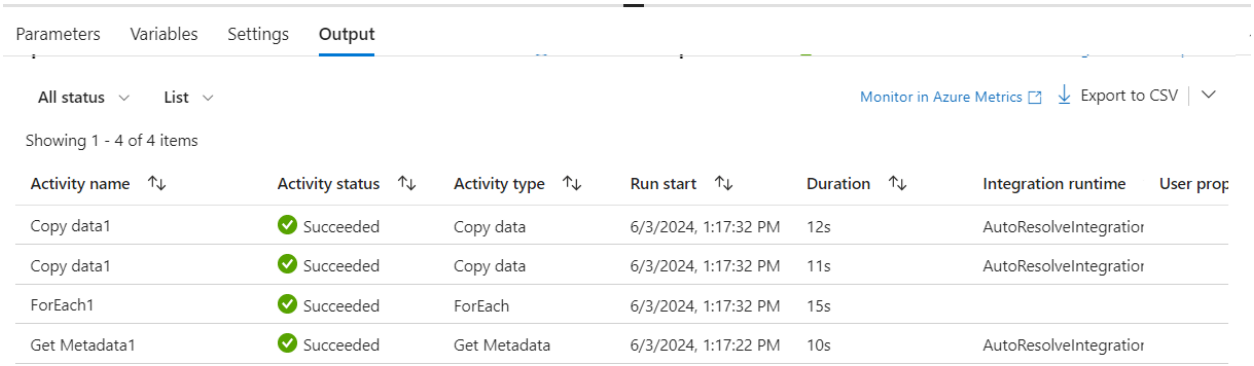
- The activities were saved/published, and all were successfully published and executed (Image 1, 2, 3).



- Image 1



- Image 2

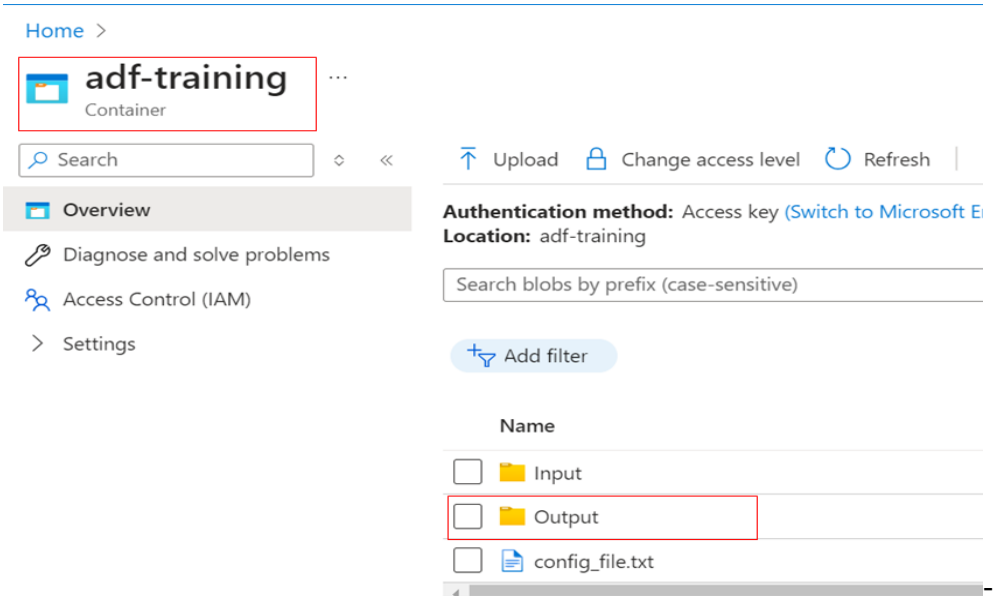


Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User prop
Copy data1	Succeeded	Copy data	6/3/2024, 1:17:32 PM	12s	AutoResolveIntegratior	
Copy data1	Succeeded	Copy data	6/3/2024, 1:17:32 PM	11s	AutoResolveIntegratior	
ForEach1	Succeeded	ForEach	6/3/2024, 1:17:32 PM	15s		
Get Metadata1	Succeeded	Get Metadata	6/3/2024, 1:17:22 PM	10s	AutoResolveIntegratior	

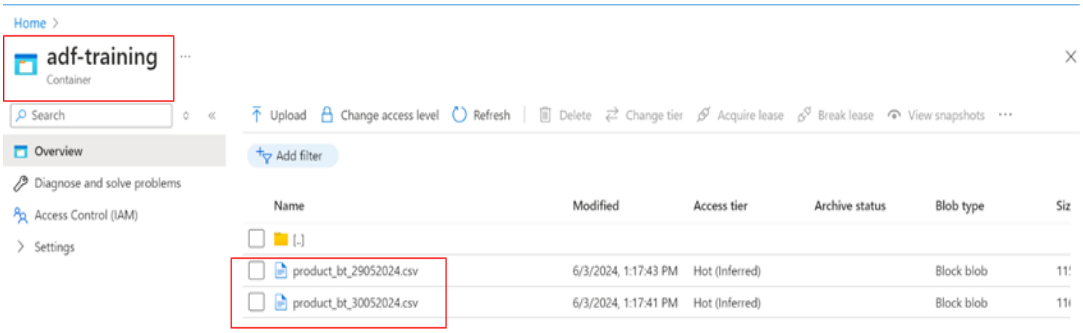
- Image 3

7. Verifying the Output

- I then visited the output folder in the adf-training container and could see that both files were successfully added to the output folder (refer image 1, 2).



- Image 1



- Image 2

Summary:

This document outlines the steps taken to create and execute an Azure Data Factory (ADF) pipeline for loading multiple files into an output folder.