

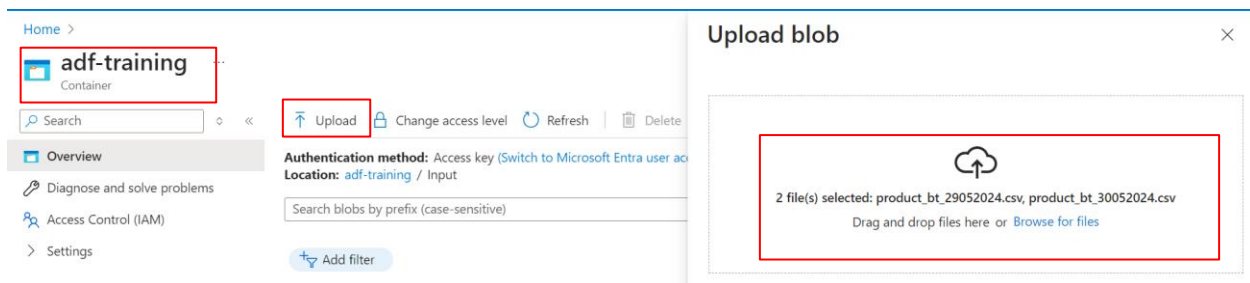
Formal Documentation of Azure Data Factory Pipeline – Training

Use Case: Loading Multiple Files Together in the Output Folder

Steps:

1. Loading the File into the Container.

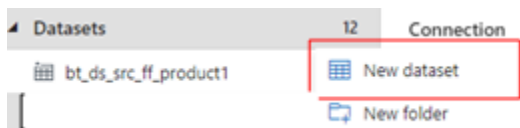
- A container named adf-training was previously created in the Azure storage account.
- For this pipeline, the files were uploaded into the container, in the input folder (refer to image 1)
- File names: product_bt_29052024.csv, product_bt_30052024.csv
- Once the upload was successful, the files were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

New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

blob

All Azure Database File Generic protocol NoSQL Services and apps



Azure Blob Storage

- Image 2

Select format

Choose the format type of your data



Avro



Binary



DelimitedText



Excel



JSON



ORC



Parquet



XML

Continue

Back

Cancel

Connection

Schema Parameters

Linked service *

ADF_Training

Test connection

Edit

New

[Learn more](#)

File path *

adf-training

/ Input

@dataset().file_name

Compression type

Select...

Column delimiter ①

Pipe (|)

Row delimiter ①

Default (\r\n, or \n)

Encoding ①

Default(UTF-8)

Quote character ①

Double quote (")

Escape character ①

Backslash (\)

First row as header ①



-Image 4

Pipeline expression builder



Add dynamic content below using any combination of [expressions](#), [functions](#) and [system variables](#).

```
@dataset().file_name
```

[Clear contents](#)

Parameters

Functions

Search



file_name

OK

Cancel

- Image 5

Connection

Schema

Parameters

New

Delete



Name

Type

Default value



file_name

String



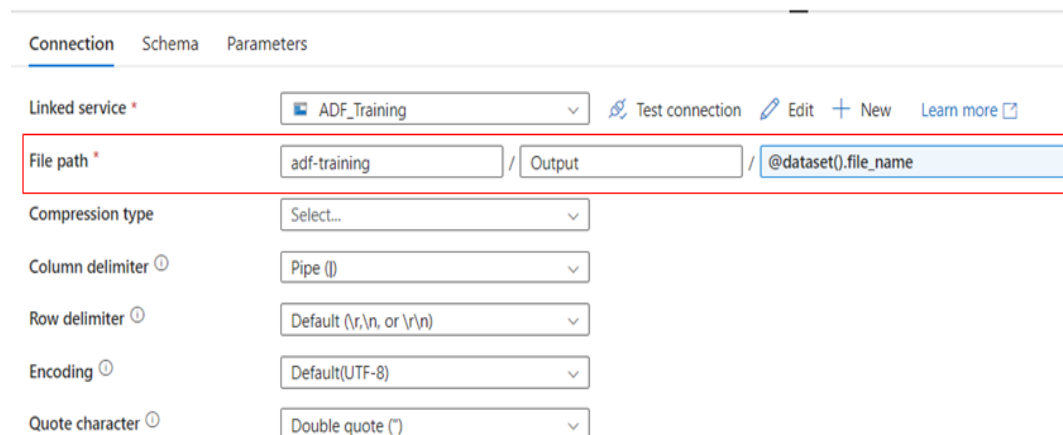
product.csv



-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).



Connection Schema Parameters

Linked service * ADF_Training [Test connection](#) [Edit](#) [+ New](#) [Learn more](#)

File path * adf-training / Output / @dataset().file_name

Compression type Select...

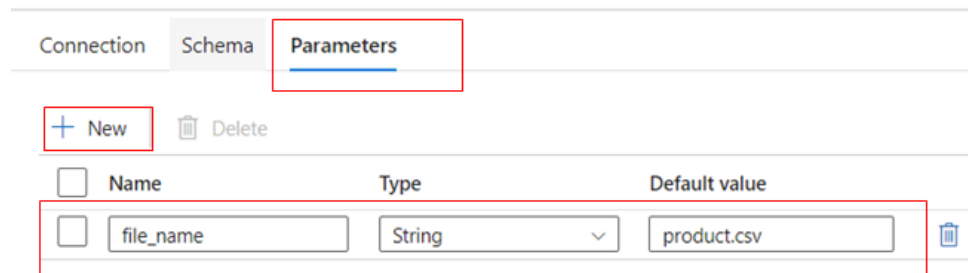
Column delimiter ⓘ Pipe (|)

Row delimiter ⓘ Default (\r,\n, or \r\n)

Encoding ⓘ Default(UTF-8)

Quote character ⓘ Double quote (")

-Image 1



Connection Schema **Parameters**

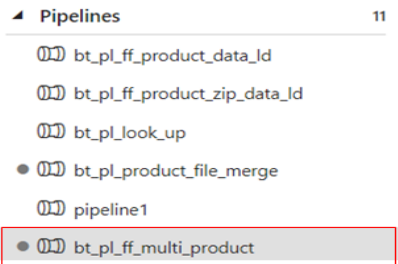
[+ New](#) [Delete](#)

<input type="checkbox"/>	Name	Type	Default value	
<input type="checkbox"/>	file_name	String	product.csv	Delete

- Image 2

3. Pipeline Creation

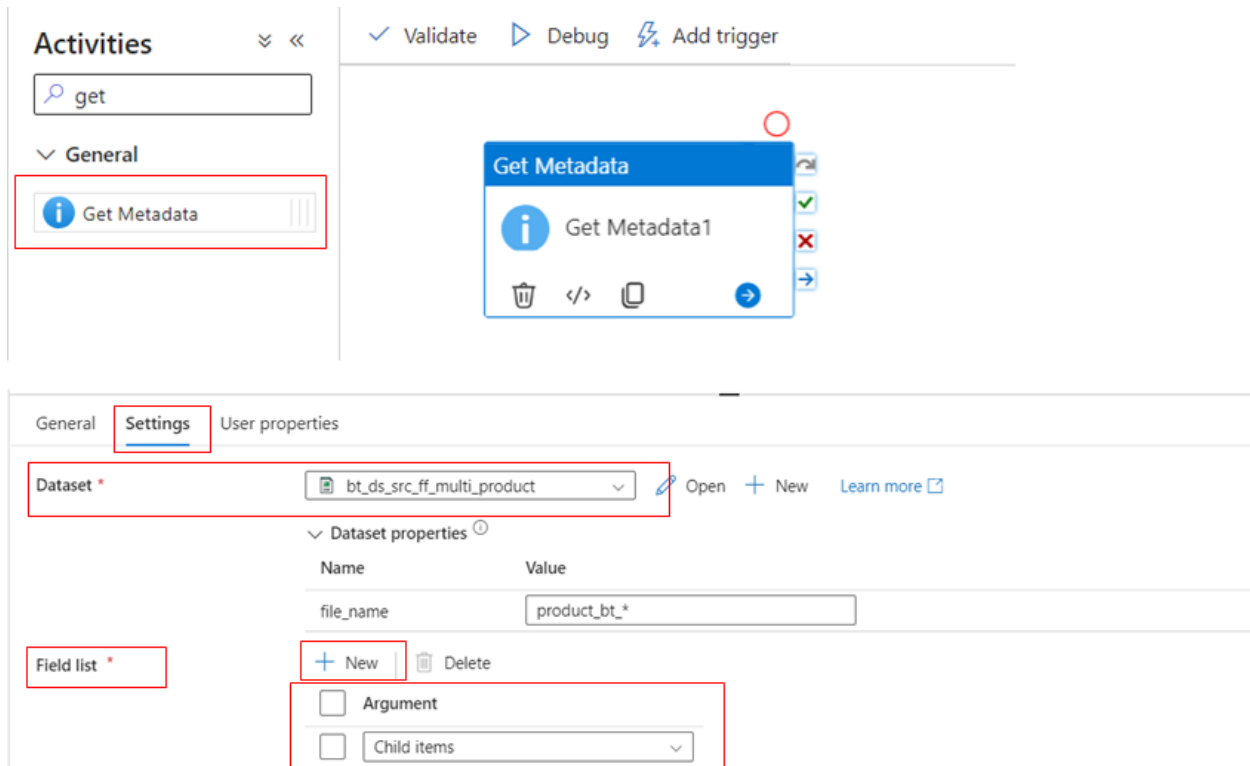
- A new pipeline named bt_pl_ff_multi_product was created in Azure Data Factory (ADF) to load the multiple file in the output folder(refer image 1).



- Image 1

4. Get Metadata Activity

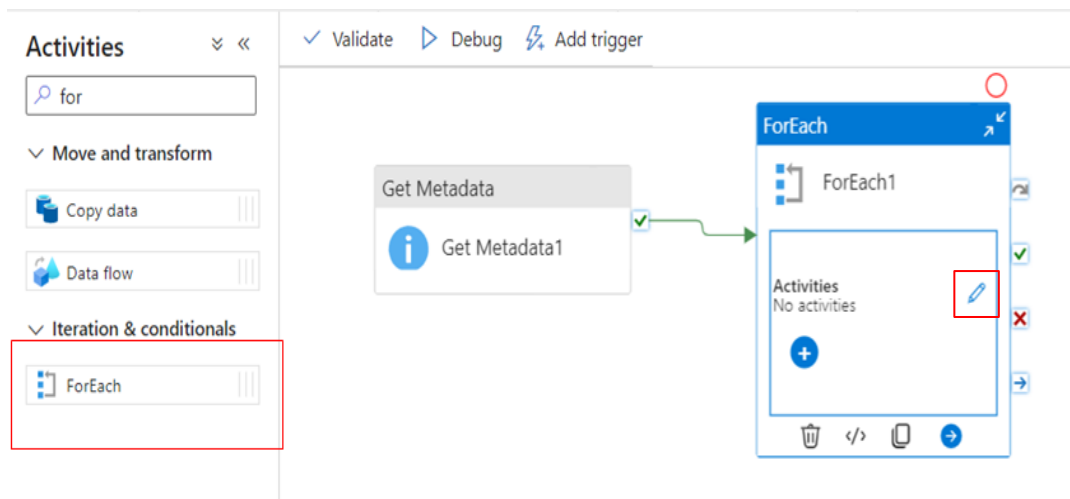
- I used the Get Metadata activity to get all the files/subfolders in the given path (refer image 1).
- I selected the source file and gave a common expression for the file name as "product_bt" since this is the common name in both files.
- In the field list, I selected child items. Child items are used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



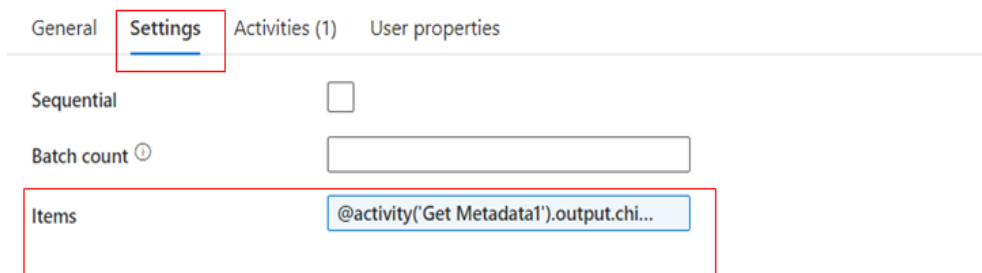
5. For Each Activity

- 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

Pipeline expression builder

Add dynamic content below using any combination of [expressions](#), [functions](#) and [system variables](#).

```
@activity('Get Metadata1').output.childItems
```

[Clear contents](#)

Activity outputs

Parameters

System variables

Functions

Variables

Search

Copy data1

Copy data1 activity output

Copy data1

Copy data1 pipeline return value

Get Metadata1

Get Metadata1 activity output

Get Metadata1

Get Metadata1 pipeline return value

Get Metadata1 childItems

List of subfolders and files in the given folder

Get Metadata1 columnCount

OK

Cancel

- Image 3

Activities

for

Move and transform

Copy data

Data flow

Iteration & conditionals

ForEach

Validate Validate copy runtime Debug Add trigger

bt_pl_ff_multi_product > ForEach1

Copy data

Copy data1

Copy data1

- Image 4

General **Source** Sink Mapping Settings User properties

Source dataset * bt_ds_src_ff_multi_product [Open](#) [New](#) [Preview data](#) [Learn more](#)

Dataset properties

Name	Value
file_name	@item().name

File path type

☒ File path in dataset ☐ Prefix ☐ Wildcard file path ☐ List of files

Filter by last modified

Start time (UTC) End time (UTC)

- Image 5

Pipeline expression builder

Add dynamic content below using any combination of [expressions](#), [functions](#) and [system variables](#).

@item().name

[Clear contents](#)

ForEach iterator Activity outputs Parameters System variables Functions Variables

ForEach1
Current item

-Image 6

General Source **Sink** Mapping Settings User properties

Sink dataset * bt_ds_tgt_ff_multi_product [Open](#) [New](#) [Learn more](#)

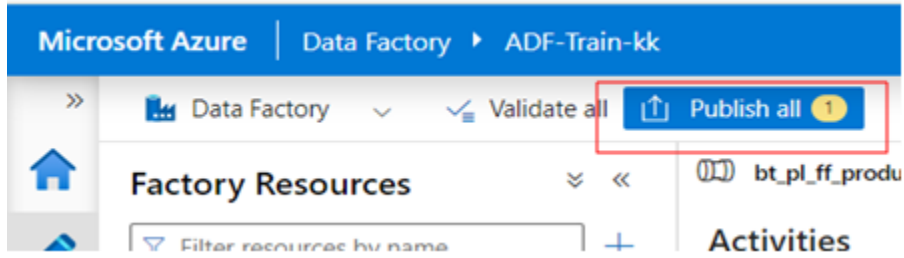
Dataset properties

Name	Value
file_name	@item().name

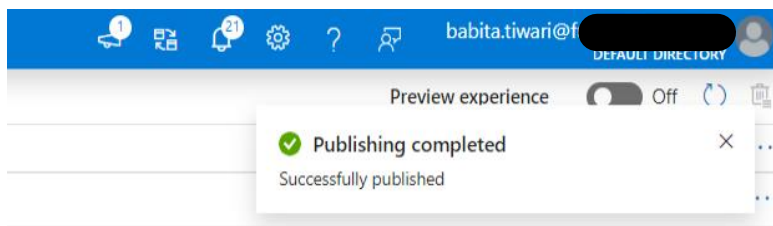
- Image 7.

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

Parameters

Variables

Settings

Output

All status

List

Monitor in Azure Metrics

Export to CSV

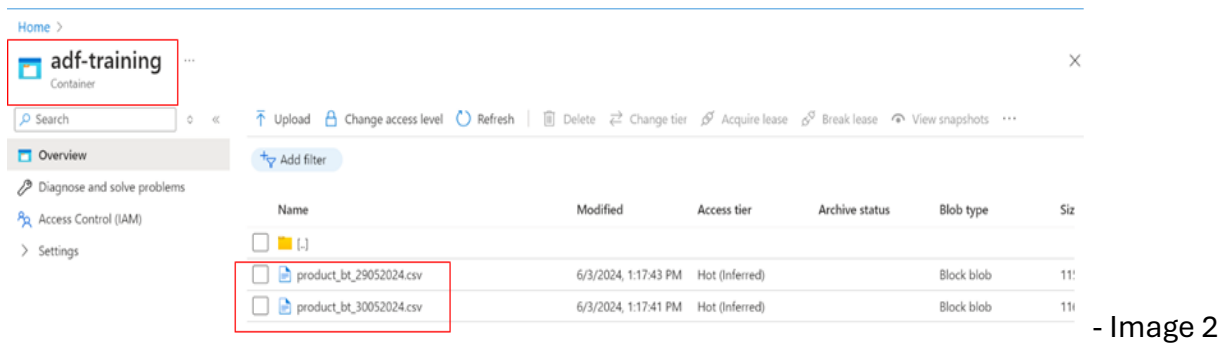
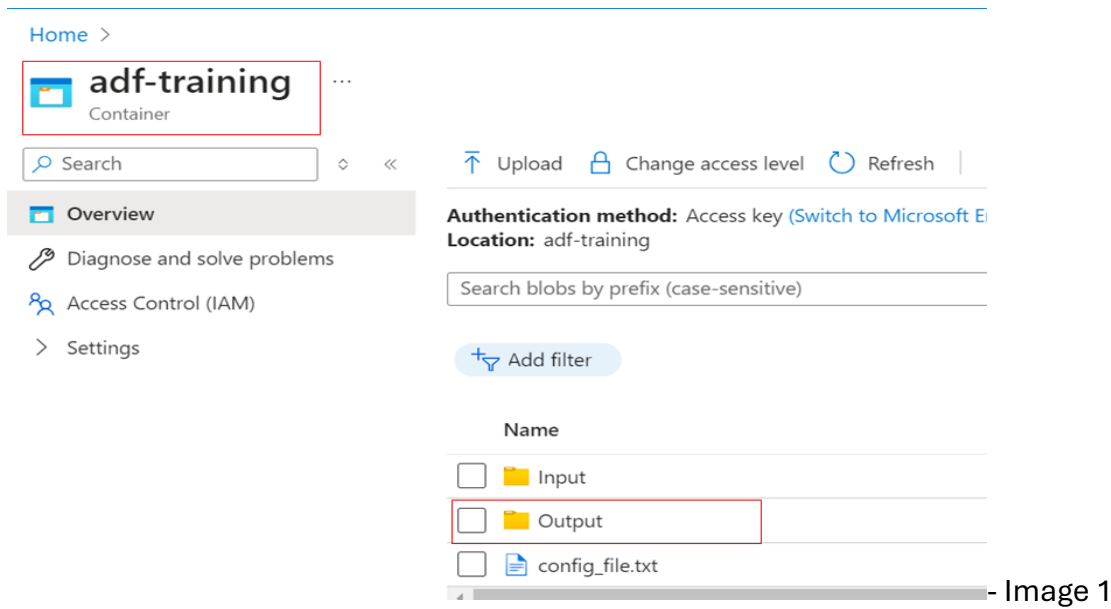
Showing 1 - 4 of 4 items

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	AutoResolveIntegration	
Copy data1	Succeeded	Copy data	6/3/2024, 1:17:32 PM	11s	AutoResolveIntegration	
ForEach1	Succeeded	ForEach	6/3/2024, 1:17:32 PM	15s		
Get Metadata1	Succeeded	Get Metadata	6/3/2024, 1:17:22 PM	10s	AutoResolveIntegration	

- 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).



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.

