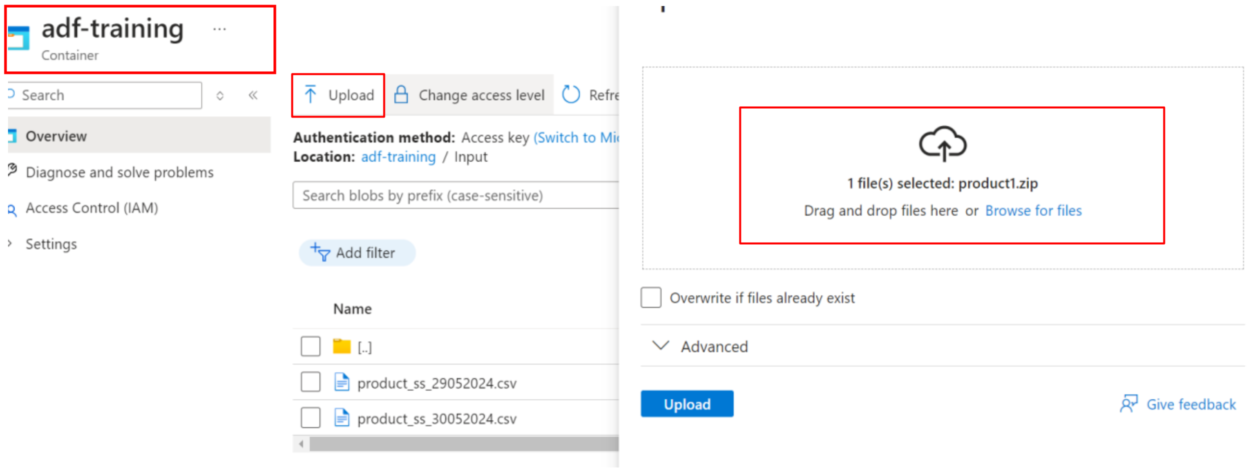
Formal Documentation of Azure Data Factory Pipeline – Training.

Usecase: Uploading zipped File from Source to Target in an Unzipped Way.

This use case involves uploading a zipped file from the source to the target in an unzipped format using the Copy Activity in Azure Data Factory. After the upload, the process checks if the file exists using the Get Metadata activity, applies an If condition to verify the existence, and if true, copies the data and then deletes the source file to avoid duplication.

1. Loading the File into the Container

- A container named adf-training was previously created in the Azure storage account.
- For this pipeline too, the file was uploaded into the container, in the input folder (refer image 1)
- Once the upload was successful, file was added in the input folder.



Authentication method: Access key (Switch to Mi
Location: adf-training / Input

Search blobs by prefix (case-sensitive)

product_ss_29052024.csv
product_ss_30052024.csv

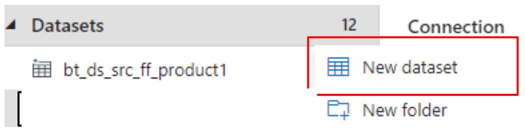
1 file(s) selected: product1.zip
Drag and drop files here or Browse for files

- Image 1

2. Dataset Creation.

Source Dataset

- A new source dataset was created for the source data file.
- Source dataset name: bt_ds_src_ff_product_zip
- 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).
- Next, I selected the Delimited Text file format, which brought me to the properties page where I defined the dataset name and path (refer to image 3).
- I specified the dataset name, selected the linked service, and provided the path of my input file: adf-training/Input/product1.csv (refer to image 4).
- These steps created my source dataset (refer to image 5).
- I opened my source file and updated the connection, changing the column delimiter option to pipe (|) because my CSV file is pipe delimited (refer image 6)
- Since my input file is zipped, I also selected the compression type and level. I chose the ZipDeflate(zip) option for compression type and optimal compression level. (refer image 6).


Datasets 12

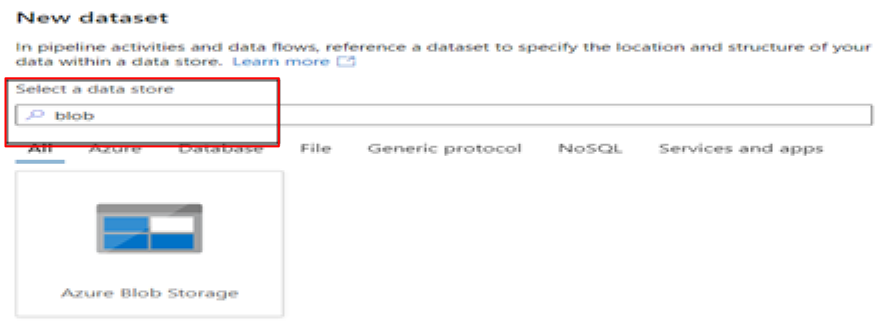
bt_ds_src_ff_product1

Connection

New dataset

New folder

- Image 1

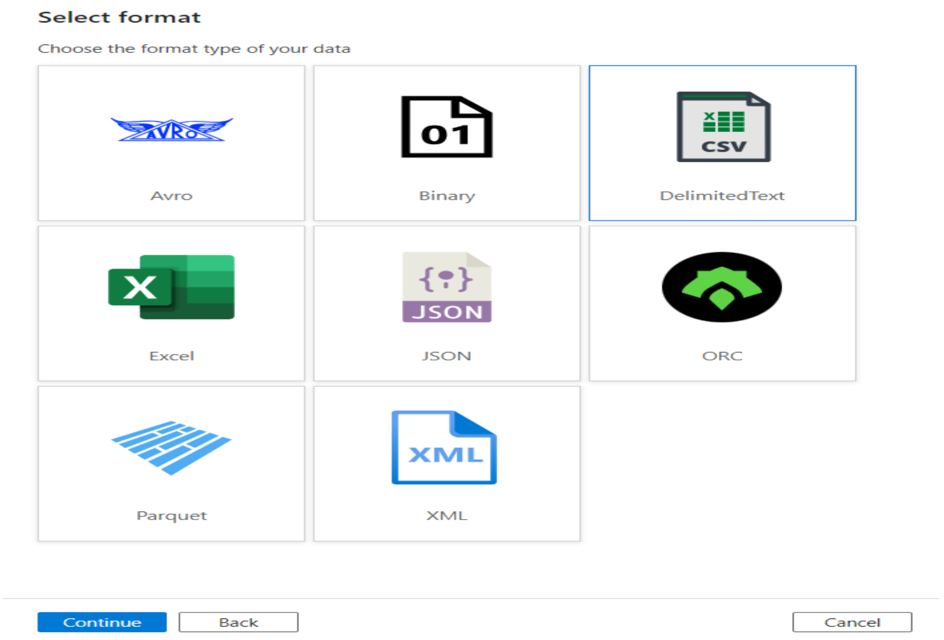

New dataset

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

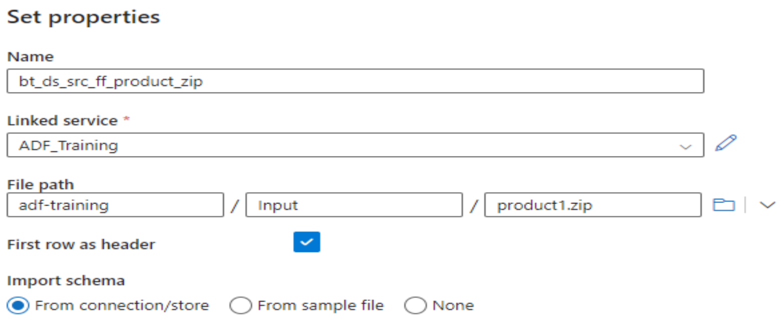
blob

- Image 2


Select format

Choose the format type of your data

- Image 3


Set properties

Name: bt_ds_src_ff_product_zip

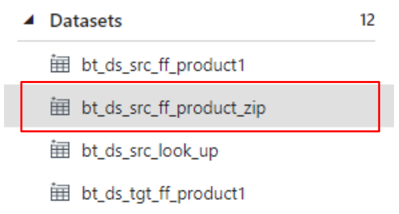
Linked service *: ADF_Training

File path: adf-training / Input / product1.zip

First row as header: ✓

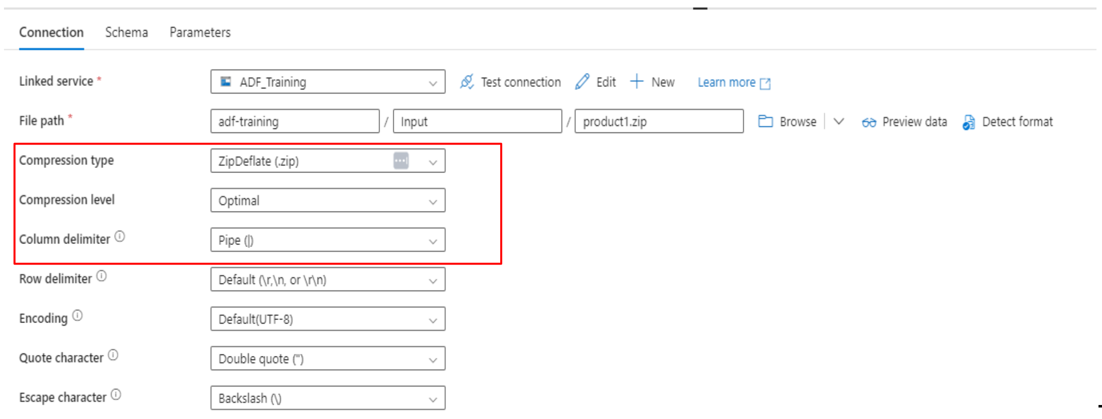
Import schema: [✓ From connection/store] From sample file None

- Image 4


Datasets 12

bt_ds_src_ff_product1
bt_ds_src_ff_product_zip
bt_ds_src_look_up
bt_ds_tgt_ff_product1

- Image 5

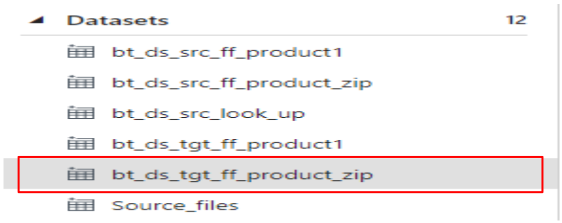


Linked service *	ADF_Training
File path *	adf-training / Input / product1.zip
Compression type	ZipDeflate (.zip)
Compression level	Optimal
Column delimiter	Pipe (|)
Row delimiter	Default (\r,\n, or \r\n)
Encoding	Default(UTF-8)
Quote character	Double quote (")
Escape character	Backslash (\)

- Image 6

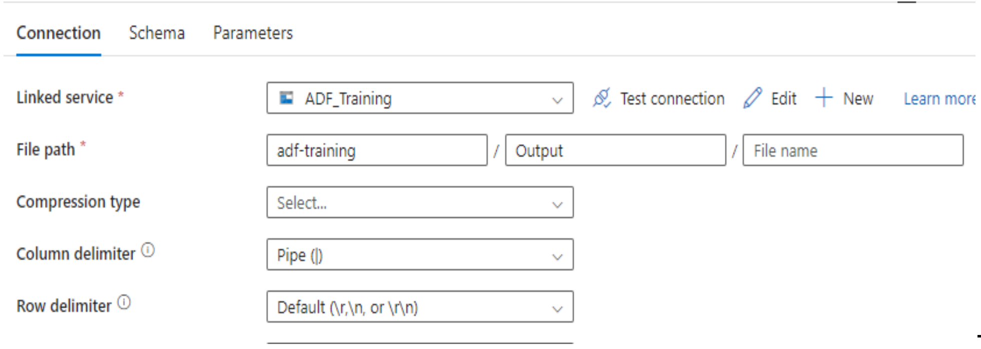
<u>Target Dataset</u>

- I followed similar steps for the target dataset. In the Author tab, I selected the datasets, 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_product_zip (refer image 1)
- I opened my source file and updated the connection, changing the column delimiter option to pipe (|) because my CSV file is pipe delimited. All other options remained unchanged (refer to image 2).
- I did not select compression type here because I didn't want my target file in a zipped format.



Datasets 12
bt_ds_src_ff_product1
bt_ds_src_ff_product_zip
bt_ds_src_look_up
bt_ds_tgt_ff_product1
bt_ds_tgt_ff_product_zip
Source_files

- Image 1



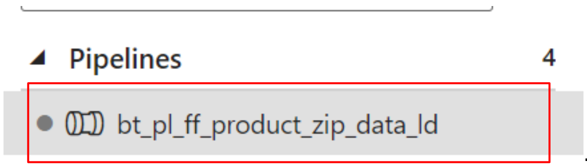
Connection | Schema | Parameters

Linked service * : ADF_Training — Test connection, Edit, New, Learn more
File path * : adf-training / Output / File name
Compression type: Select...
Column delimiter: Pipe (|)
Row delimiter: Default (\r,\n, or \r\n)

- Image 2

3. Pipeline creation.

- A new pipeline named bt_pl_ff_product_zip_data_ld was created in Azure Data Factory (ADF) to load the zipped data into the target as an unzipped file (refer image 1).

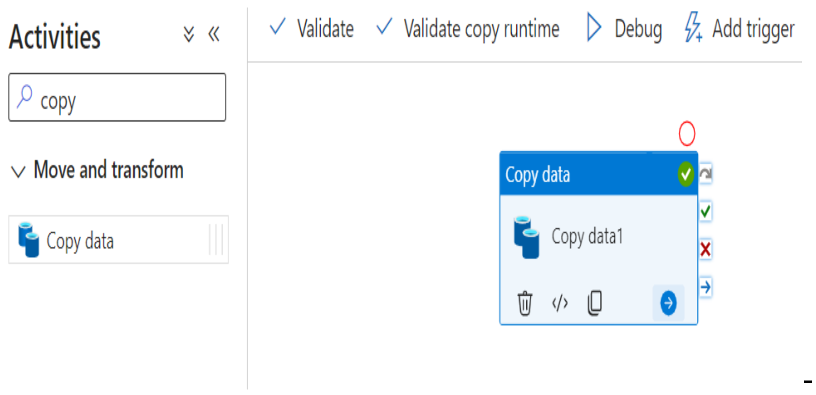


Pipelines 4
bt_pl_ff_product_zip_data_ld

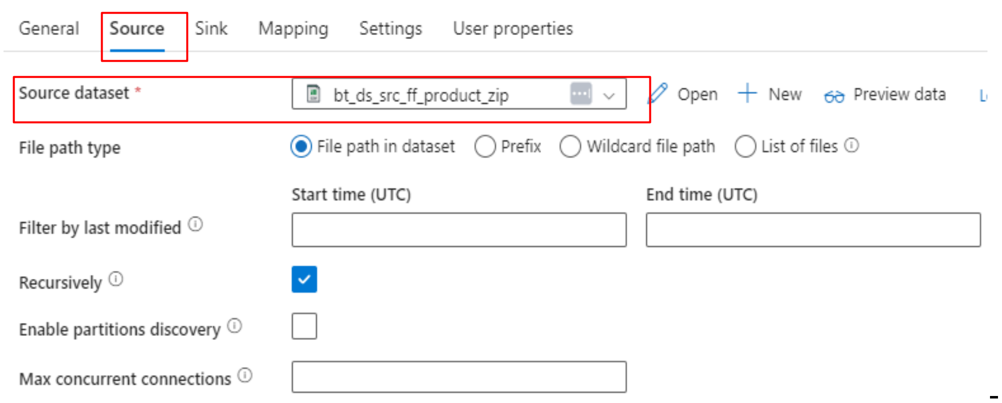
- Image 1

4. Copy Activity.

- The Copy Data activity was used to transfer data from the source to the target dataset.
- Source: bt_ds_src_ff_product1
- Sink: bt_ds_tgt_ff_product1
- I selected the Copy activity from the Activities tab (refer to image 1).
- In the source option of the Copy activity, I added the source dataset and selected the file path in the dataset. I opted for the file path in the dataset to specify a single file to copy from the source location (refer to image 2).
- Other options for the path include:
  - Prefix: Specifies a prefix for the files to copy from the source location (e.g., folder/subfolder/* to copy all files in the subfolder directory).
  - Wildcard file path: Specifies a wildcard pattern to match files to copy from the source location (e.g., *.csv to copy all CSV files).
  - List of files: Specifies a list of files to copy from the source location.
- In the sink option, I added the target dataset and changed the file extension from .txt to .csv (refer to image 3).



- Image 1



Source dataset * bt_ds_src_ff_product_zip

File path type [✓ File path in dataset] Prefix Wildcard file path List of files

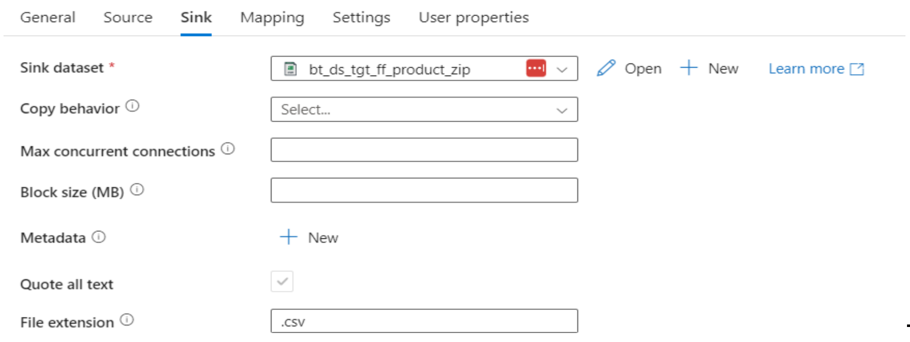
Filter by last modified — Start time (UTC): End time (UTC):

Recursively ✓

Enable partitions discovery ☐

Max concurrent connections

- Image 2



Sink dataset * bt_ds_tgt_ff_product_zip

Copy behavior Select...

Max concurrent connections

Block size (MB)

Metadata + New

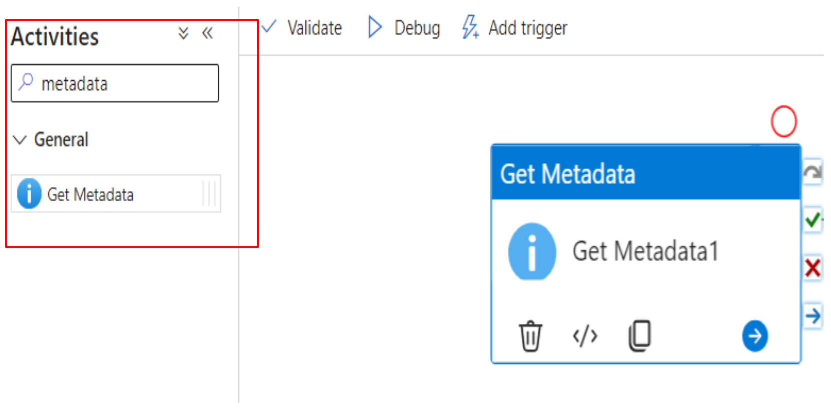
Quote all text ✓

File extension .csv

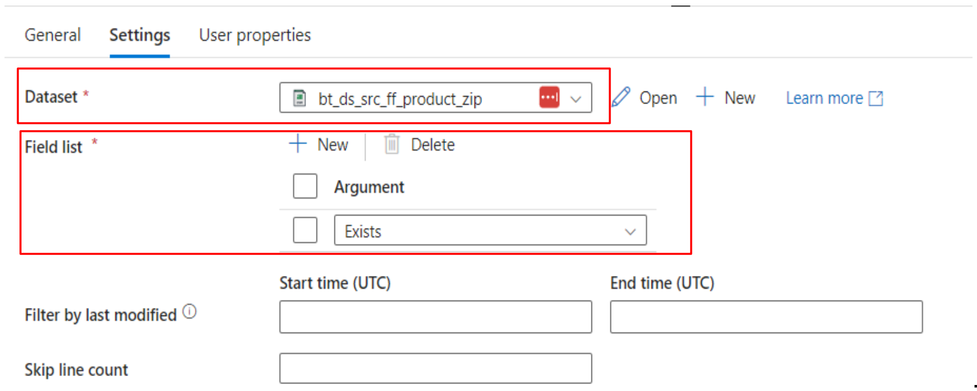
- Image 3

5. Get Metadata acticity.

- I used the GetMetadata activity to check if the file in the given path exists (refer image 1)
- In the settings option, I added the source dataset file on which I wanted to validate existence.
- Below the dataset, in the field list option, I selected the "Exists" dropdown (refer image 2)
- The "Exists" argument in the GetMetadata activity in Azure Data Factory (ADF) is used to check whether a specified item (such as a file, folder, or table) exists in the data store.
- The outcome of the existence check will be a Boolean value, either true or false.
- My file existed, so it returned a true outcome.



- Image 1



Dataset * bt_ds_src_ff_product_zip

Field list *

	Argument
☐	Exists

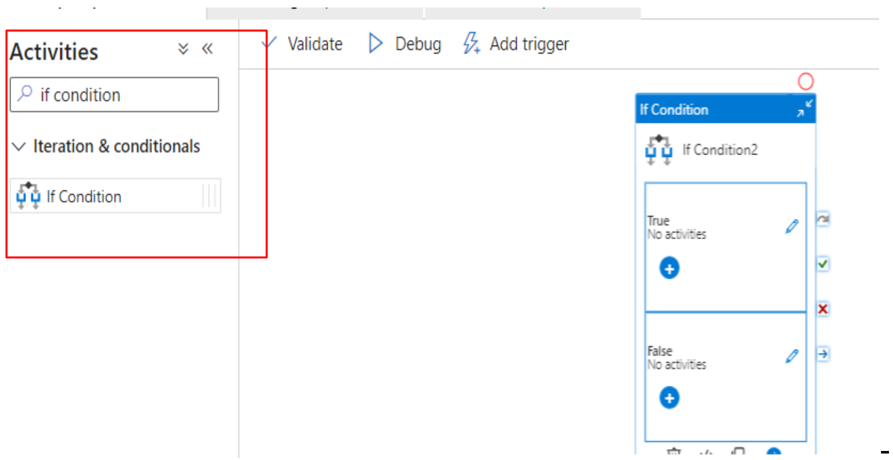
Filter by last modified — Start time (UTC): — End time (UTC):

Skip line count

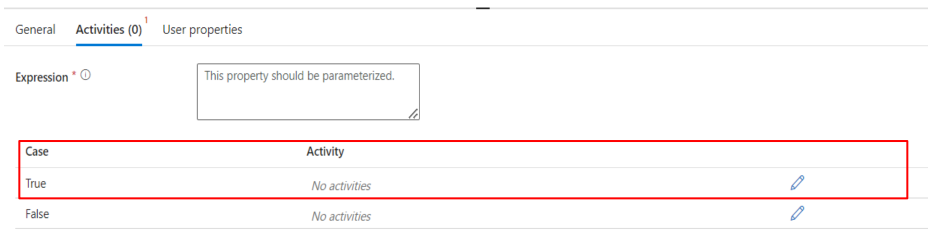
- Image 2

6. If Condition activity.

- I then used the if condition activity. (refer image 1)
- I established a connection between the Get Metadata activity and the If Condition activity (refer image 6).
- The If Condition activity was configured to copy data to the target if the "Exists" outcome was true.
- Within the true branch of the If Condition activity, I created a Copy activity where I specified the source and sink datasets (refer image 2,3,4,5)
- Additionally, in the true branch, I included a Delete activity to remove the source file from the input location once the Copy activity was completed. Provided the source path and disabled loggings. (refer image 7,8,9)



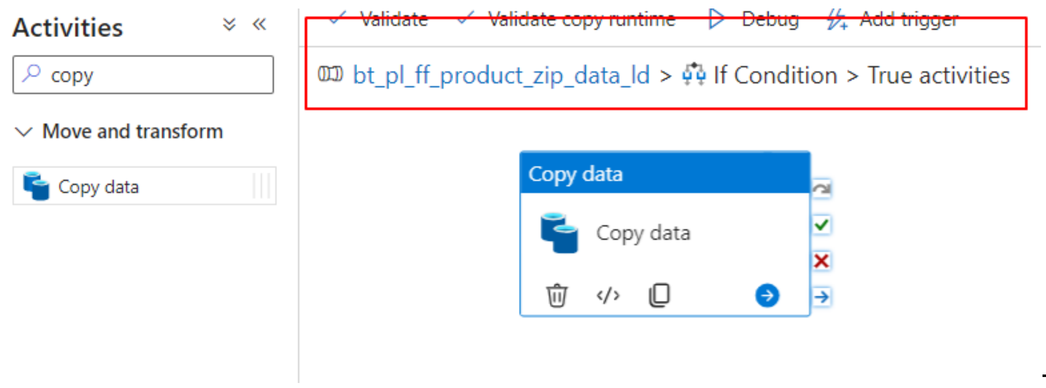
- Image 1



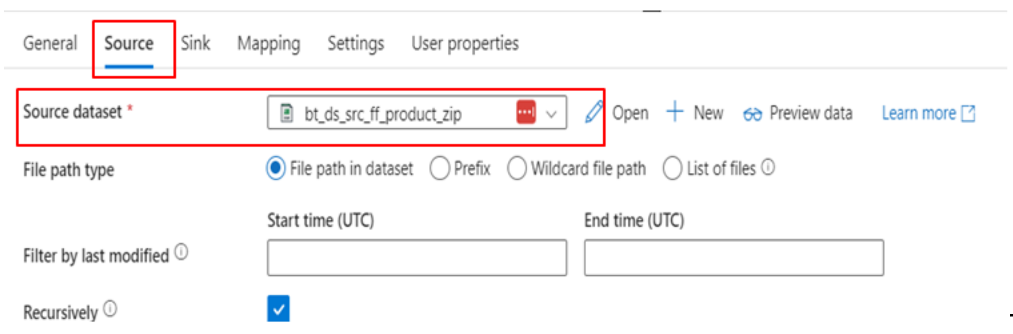
Expression * This property should be parameterized.

Case	Activity
True	*No activities*
False	*No activities*

- Image 2



- Image 3



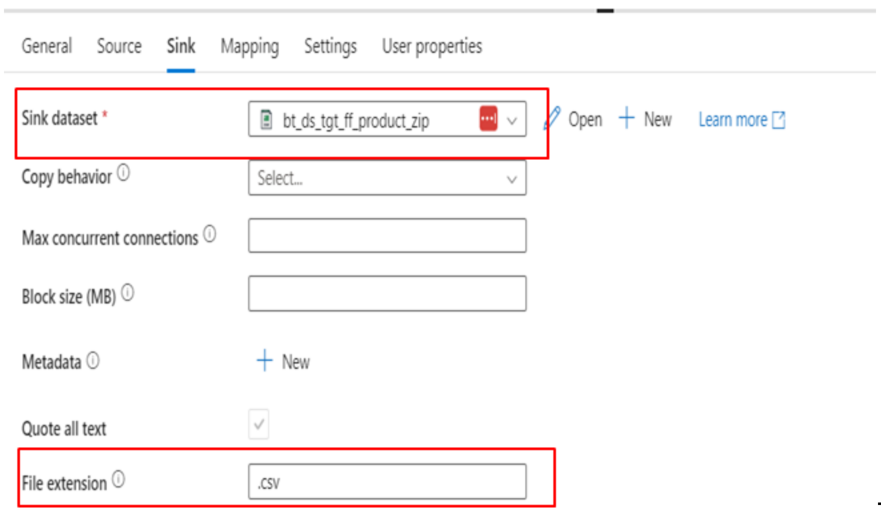
Source dataset * bt_ds_src_ff_product_zip

File path type [✓ File path in dataset] Prefix Wildcard file path List of files

Filter by last modified: Start time (UTC) End time (UTC)

Recursively ✓

- Image 4


General | Source | Sink | Mapping | Settings | User properties

Sink dataset * bt_ds_tgt_ff_product_zip

Copy behavior: Select...

Max concurrent connections

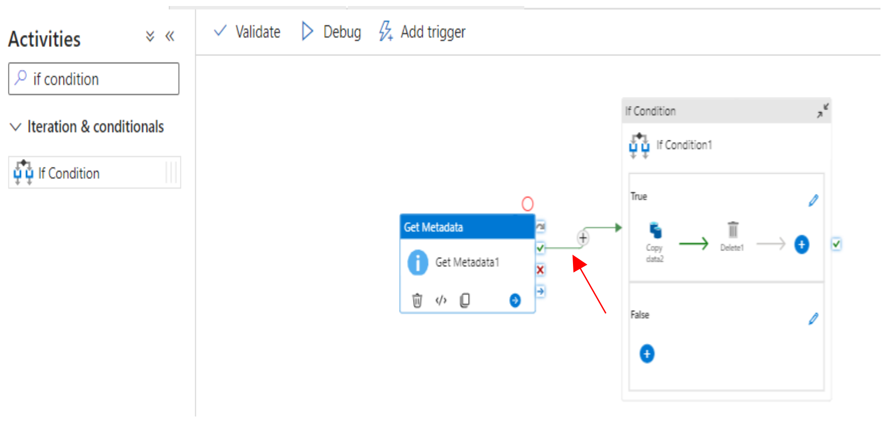
Block size (MB)

Metadata: + New

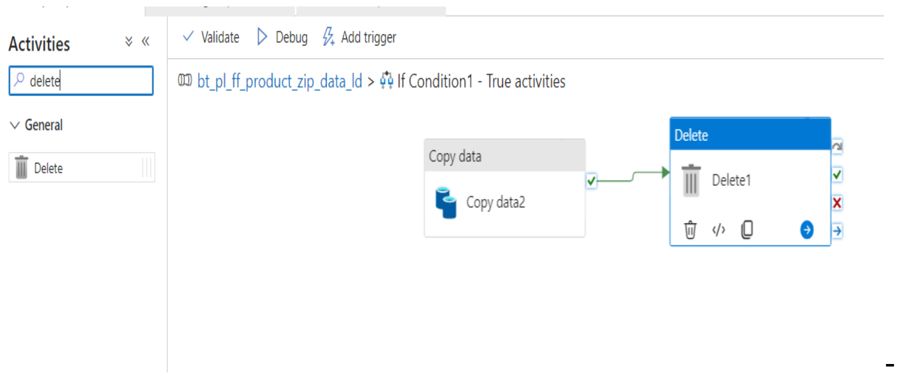
Quote all text ✓

File extension: .csv

- Image 5

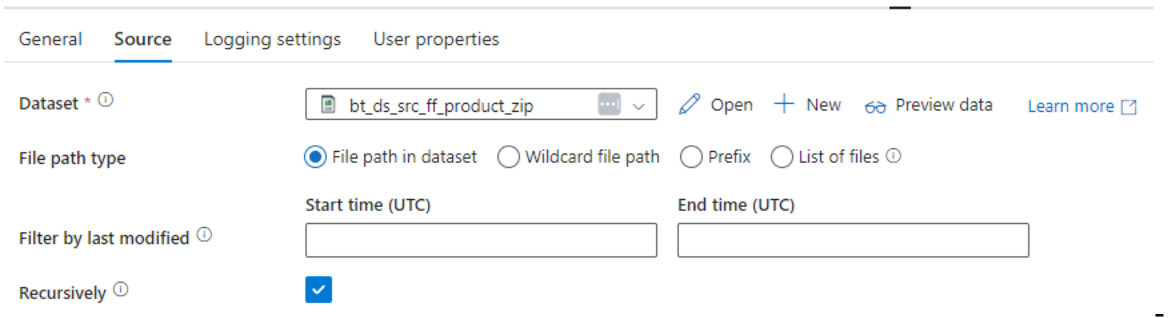


- Image 6



bt_pl_ff_product_zip_data_ld > If Condition1 - True activities

- Image 7


General | Source | Logging settings | User properties

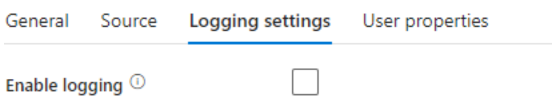
Dataset * : bt_ds_src_ff_product_zip — Open | New | Preview data | Learn more

File path type: [✓ File path in dataset] Wildcard file path | Prefix | List of files

Filter by last modified: Start time (UTC) | End time (UTC)

Recursively: ✓

Image 8

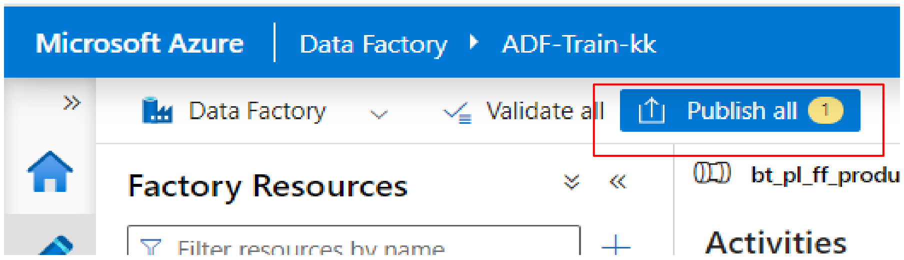

General | Source | Logging settings | User properties

Enable logging: ☐

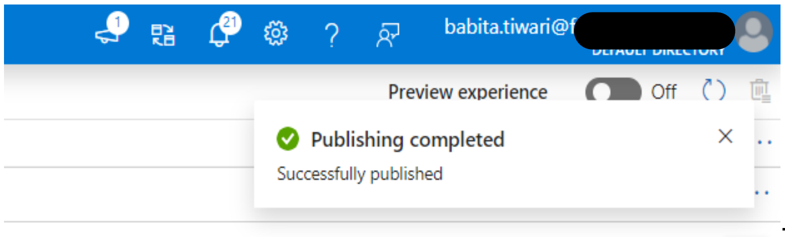
- Image 9

7. Publishing and Executing the Pipeline.

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



- Image 1

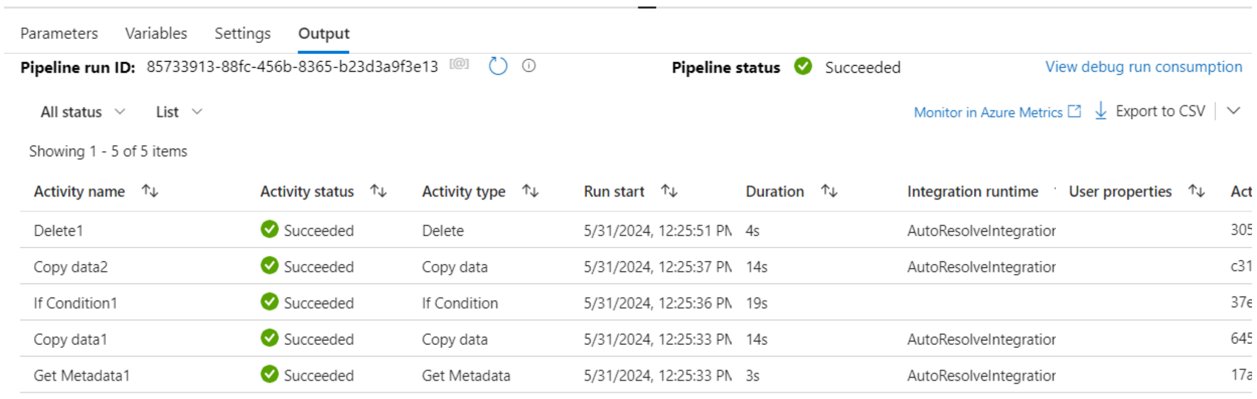

babita.tiwari@f[redacted]

Preview experience Off

Publishing completed

Successfully published

- Image 2


Parameters Variables Settings Output

Pipeline run ID: 85733913-88fc-456b-8365-b23d3a9f3e13 **Pipeline status** Succeeded View debug run consumption

All status List Monitor in Azure Metrics Export to CSV

Showing 1 - 5 of 5 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Act
Delete1	✓ Succeeded	Delete	5/31/2024, 12:25:51 PN	4s	AutoResolveIntegratior		305
Copy data2	✓ Succeeded	Copy data	5/31/2024, 12:25:37 PN	14s	AutoResolveIntegratior		c31
If Condition1	✓ Succeeded	If Condition	5/31/2024, 12:25:36 PN	19s			37e
Copy data1	✓ Succeeded	Copy data	5/31/2024, 12:25:33 PN	14s	AutoResolveIntegratior		645
Get Metadata1	✓ Succeeded	Get Metadata	5/31/2024, 12:25:33 PN	3s	AutoResolveIntegratior		17a

- Image 3

Summary:

This documentation outlines the process of uploading a zipped file to Azure using the copy activity, checking file existence with the Get Metadata activity, and using the If condition for conditional operations. This ensures efficient data transfer and prevents duplication. The steps include creating datasets, configuring a pipeline, performing copy and delete operations, and verifying the outcomes.