



Generating a Parquet File

In this lab, we're using Azure Data Factory to create a Parquet file from a CSV file. The end goal is to automate the process of converting data from one format to another for efficient storage and analysis. This involves configuring pipelines, datasets, and activities in Azure Data Factory to copy data from the CSV file to a Parquet file, ensuring compatibility and optimization for downstream analytics tasks.

1. Now before starting the lab first, we go to our storage account then go inside our parquet container and we are going to delete our log.parquet file.

Home > Resource groups > demo-resource-group > sqlstorage1010 | Containers >

parquet Container

Search Overview Diagnose and solve problems Access Control (IAM)

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Microsoft Entra user account) Location: parquet

Search blobs by prefix (case-sensitive) Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
log.parquet	4/22/2024, 5:12:03 PM	Hot (Inferred)		Block blob	660.81 KiB	Available

Shared access tokens Manage ACL Access policy Properties Metadata

2. In this lab we are going to create a parquet file using our CSV file in the data factory.
3. Now in your data factory wizard first go to Author there you can see Pipelines, now click on the three dots, and choose new Pipeline from here itself.

Home Author Monitor Manage Learning Center

Data Factory Validate all Publish all

Factory Resources

Pipelines 1

- 01-Copy-data-to-logdata
- New pipeline

Datasets

- DestinationDataset_s16
- SourceDataset_s16
- New folder

Data flows 0

Power Query 0

4. Then you will see that this type of interface layout has been generated by the wizard. First, we are going to give a name to this pipeline.

The screenshot shows the Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists Pipelines (01-Copy-data-to-logdata, 02-Copy-to-Parquet), Datasets (DestinationDataset_s16, SourceDataset_s16), Data flows (0), and Power Query (0). The main area shows the 'Activities' pane with a search bar and a list of activities: Move and transform, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, Machine Learning, and Power Query. To the right is the 'Properties' pane for the selected pipeline '02-Copy-to-Parquet'. The 'General' tab is selected, showing the 'Name' field set to '02-Copy-to-Parquet' (highlighted with a red box), a 'Description' field, and an 'Annotations' section with a 'New' button.

5. Now form the activities if you expand the move and transform section then you will see two options one for copying the data and the other for data flow.
6. Here we must choose to copy data and drag it to the canvas. So, this will allow us to copy data from our log.csv file to the parquet file which we will create moving further.

The screenshot shows the 'Activities' pane for the pipeline '02-Copy-to-Parquet'. The 'Move and transform' section is expanded, revealing the 'Copy data' activity (highlighted with a red box) and the 'Data flow' activity. Above the activities list are buttons for Validate, Debug, and Add trigger.

7. First, we are going to give a name to our activity. Then we will move to source.

The screenshot shows the 'Activities' section of the Azure Data Factory interface. A 'Copy data' activity named 'Copydatatoparquet' is selected. The 'General' tab is active, showing the name 'Copydatatoparquet' highlighted with a red box. Other tabs include Source, Sink, Mapping, Settings, and User properties. On the left sidebar, various service categories like Synapse, Azure Data Explorer, and Azure Function are listed.

8. Now in source we will choose our source dataset. It means that from where we want to pick up our data.

The screenshot shows the 'Source' tab of the Copy activity configuration. The 'Source dataset' dropdown is set to 'SourceDataset_sl6'. Below it, there are options for 'File path type': 'File path in dataset' (selected), 'Wildcard file path', and 'List of files'. Buttons for 'Open', 'New', and 'Preview data' are also present.

9. Then in the sink you must click on New. Here we will provide the destination which is our parquet-based container. For this, we need to create a new dataset.

The screenshot shows the 'Sink' tab of the Copy activity configuration. The 'Sink dataset' dropdown is set to 'Select...'. A 'New' button is visible next to it. Other tabs include General, Source, Mapping, Settings, and User properties.

10. Again, you have to choose Azure data late gen2 and click on continue.

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

All Azure Database File Generic protocol NoSQL Services and apps

 Azure Cosmos DB for NoSQL	 Azure Data Explorer (Kusto)	 Azure Data Lake Storage Gen2
---	---	--

11. Now this time around you have to choose a format for our data. So, we will choose Parquet and click on continue.

Select format

Choose the format type of your data

 Avro	 Binary	 DelimitedText
 JSON	 ORC	 Parquet

12. Then in the properties first you will give a name to your file after that in the linked service you have to choose your storage account as you can see below.

Set properties

Name

logdata_parquet

Linked service *

Select...

Filter...

+ New

sqlstorage1010_service

sqlstorage1010_service

13. Then you have to browse for the container where you want to put your file then click on the Ok button.

Set properties

Name

logdata_parquet

Linked service *

sqlstorage1010_service



File path

parquet

/ Directory

/ File name



Import schema

From connection/store From sample file None

> Advanced

14. Now if you look at your factory resources you will see that your dataset has come over here.

15. In the wizard, when we do or choose the copy data activity, there, it does not give you the facility to give a name for your datasets. But here, since we are creating our pipeline from scratch, we can now also mention what is the name of our dataset.

Factory Resources

▽ <

Filter resources by name



▲ Pipelines 2

01-Copy-data-to-logdata

● 02-Copy-to-Parquet

▷ Change Data Capture (preview) 0

▲ Datasets 3

DestinationDataset_sl6

● logdata_parquet

SourceDataset_sl6

▷ Data flows 0

▷ Power Query 0

16. So now, we have our source and our sink in place. Before we can trigger this pipeline, or say before we can run this pipeline, so in the wizard, which we have seen in the earlier labs, the wizard would go ahead and create our pipeline, publish our pipeline, and even run the pipeline.
17. But since we are doing everything from scratch here, we need to first publish our pipeline and then run our pipeline.
18. To make sure that our pipeline is correct we have to click on validate all.

Data Factory

Validate all

Publish all

2

19. Once you have clicked on it then you will see this type of message.



Your factory has been validated.

No errors were found.

20. Now you have to click on publish all. You can see that you have a pipeline and a dataset to publish click on publish.

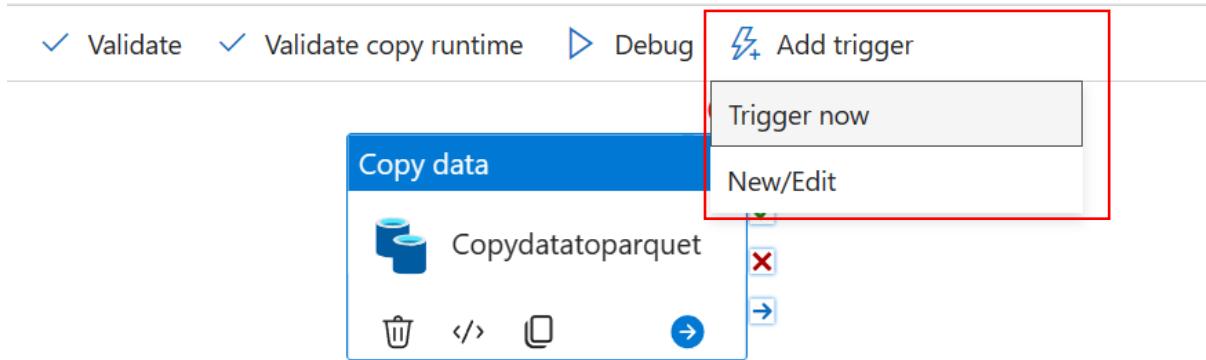
Publish all

You are about to publish all pending changes to the live environment. [Learn more](#)

Pending changes (2)

NAME	CHANGE	EXISTING
Pipelines		
02-Copy-to-Parquet	(New)	-
Datasets		
logdata_parquet	(New)	-

21. Once your publishing has completed. Click on add trigger and then on trigger now.



22. This will trigger our pipeline.

Pipeline run

! Trigger pipeline now using last published configuration.

Parameters

Name	Type	Value
No records found		

23. Now once you have triggered the pipeline after some time you will see that you are getting an error message. And if you click on view error details then you can see the error message too.

Notifications

Dismiss all

✖ Run Failed

X

Failed to run 02-Copy-to-Parquet (Pipeline).

[View pipeline run](#)

[View error details](#)

9 minutes ago

Error

↗

Operation on target Copydatatoparquet failed:

ErrorCode=ParquetInvalidColumnName,'Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=The column name is invalid. Column name cannot contain these character:[;{}]{\n\t},Source=Microsoft.DataTransfer.Common,'

24. We are getting this error message because in Parquet-based files we cannot have spaces in the column names.
25. In a log.csv file, there is a space in certain columns. For example, there's a space in the operation name, there's space in correlation ID, etcetera. This is not accepted when it comes to the Parquet-based file format, so we need to make a change.
26. In your pipeline you can see the mapping section, here you have to click on Import Schemas.

27. Below you can see the mapping.
28. Azure Data Factory is doing an automatic mapping based on its knowledge. It doesn't know about understanding whether columns have spaces in the Parquet-based destination file format. Hence, we are now making a change saying that when you're copying onto the destination, don't have spaces in the column names.

Source	Type	Destination	Type
Correlation id	abc String	Correlation id	abc UTF8
Operation name	abc String	Operation name	abc UTF8
Status	abc String	Status	abc UTF8
Event category	abc String	Event category	abc UTF8
Level	abc String	Level	abc UTF8
Time	abc String	Time	abc UTF8
Subscription	abc String	Subscription	abc UTF8
Event initiated by	abc String	Event initiated by	abc UTF8
Resource type	abc String	Resource type	abc UTF8
Resource group	abc String	Resource group	abc UTF8
Resource	abc String	Resource	abc UTF8

29. Below you can see that we have removed all the spaces.

General	Source	Sink	Mapping	Settings	User properties	
	Source	Type		Destination	Type	
	Correlation id	abc String	→	Correlationid	UTF8	+ 🗑
	Operation name	abc String	→	Operationname	UTF8	+ 🗑
	Status	abc String	→	Status	abc UTF8	+ 🗑
	Event category	abc String	→	Eventcategory	UTF8	+ 🗑
	Level	abc String	→	Level	abc UTF8	+ 🗑
	Time	abc String	→	Time	abc UTF8	+ 🗑
	Subscription	abc String	→	Subscription	abc UTF8	+ 🗑
	Event initiated by	abc String	→	Eventinitiatedby	UTF8	+ 🗑
	Resource type	abc String	→	Resourcetype	UTF8	+ 🗑
	Resource group	abc String	→	Resourcegroup	UTF8	+ 🗑
	Resource	abc String	→	Resource	abc UTF8	+ 🗑

30. Once you are done now you have to click on public again.

Publish all

You are about to publish all pending changes to the live environment. [Learn more](#) 📋

Pending changes (1)

NAME	CHANGE	EXISTING
▼ Pipelines		
02-Copy-to-Parquet	(Edited)	02-Copy-to-Parquet

31. Once the publishing is completed then you have to click on trigger now. Below you can see that the run was successful.

Run Succeeded

Successfully ran 02-Copy-to-Parquet (Pipeline).

[View pipeline run](#)

a few seconds ago

32. Now go to the monitor section and look for your pipeline it has been succeeded.

The screenshot shows the 'Activity runs' section of the pipeline run history. It displays a single activity named 'Copydatatoparquet' which has succeeded. The table below provides detailed information about this activity:

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User pr
Copydatatoparquet	Succeeded	Copy data	4/24/2024, 2:11:05 PM	19s	AutoResolveIntegration	

33. Now go to your storage account and open your parquet-based container. There you will see your parquet file in place.

The screenshot shows the Azure Storage Explorer interface. A blob named 'Log.parquet' is listed in the 'parquet' container. The blob details are as follows:

Name	Modified	Access tier	Archive status	Blob type	Size
Log.parquet	4/24/2024, 2:11:22 PM	Hot (Inferred)		Block blob	660