

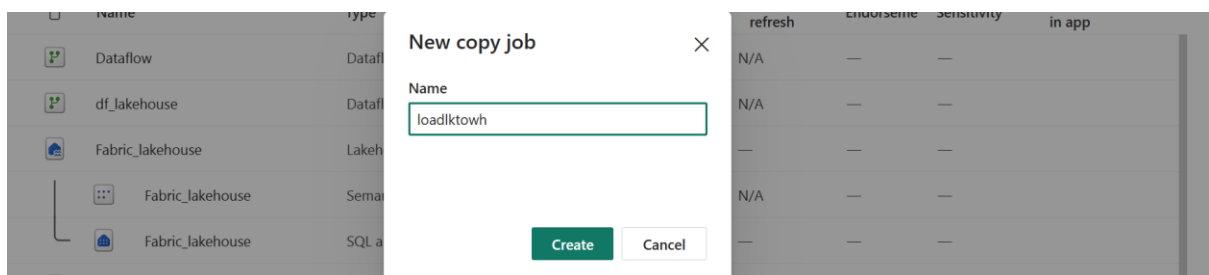
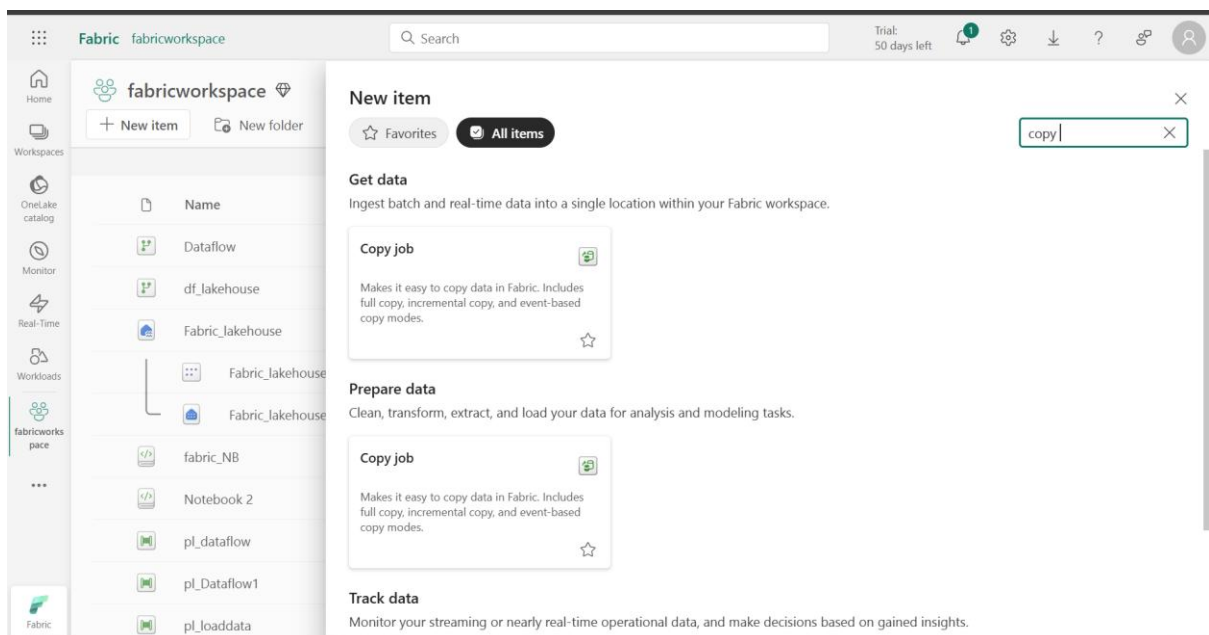
Copy Job

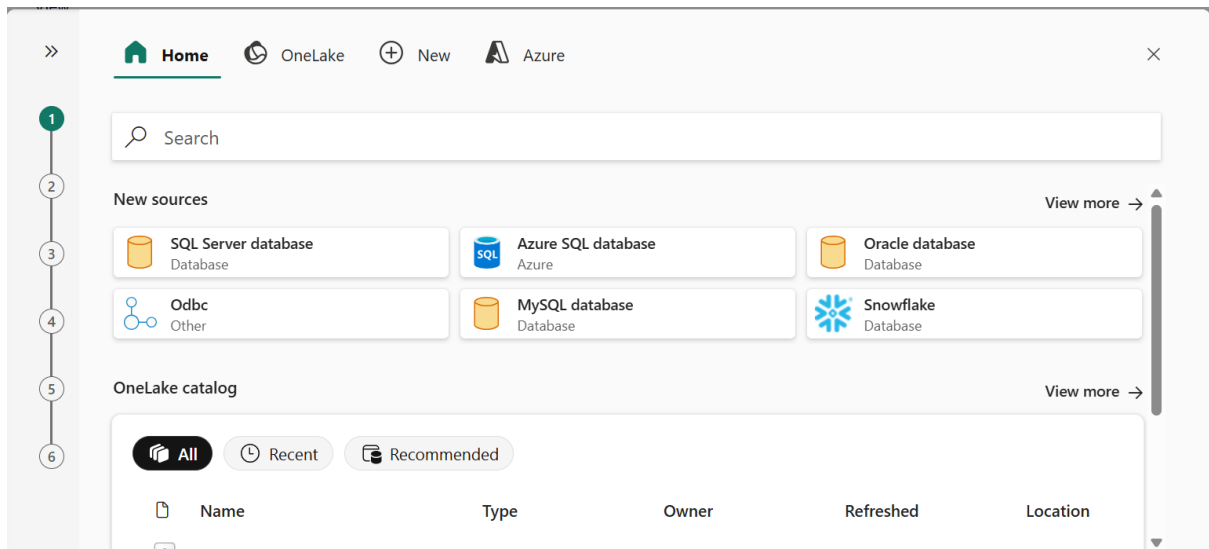
Use Copy job to load data using incremental copy from data warehouse to lake house tables

Copy Job:

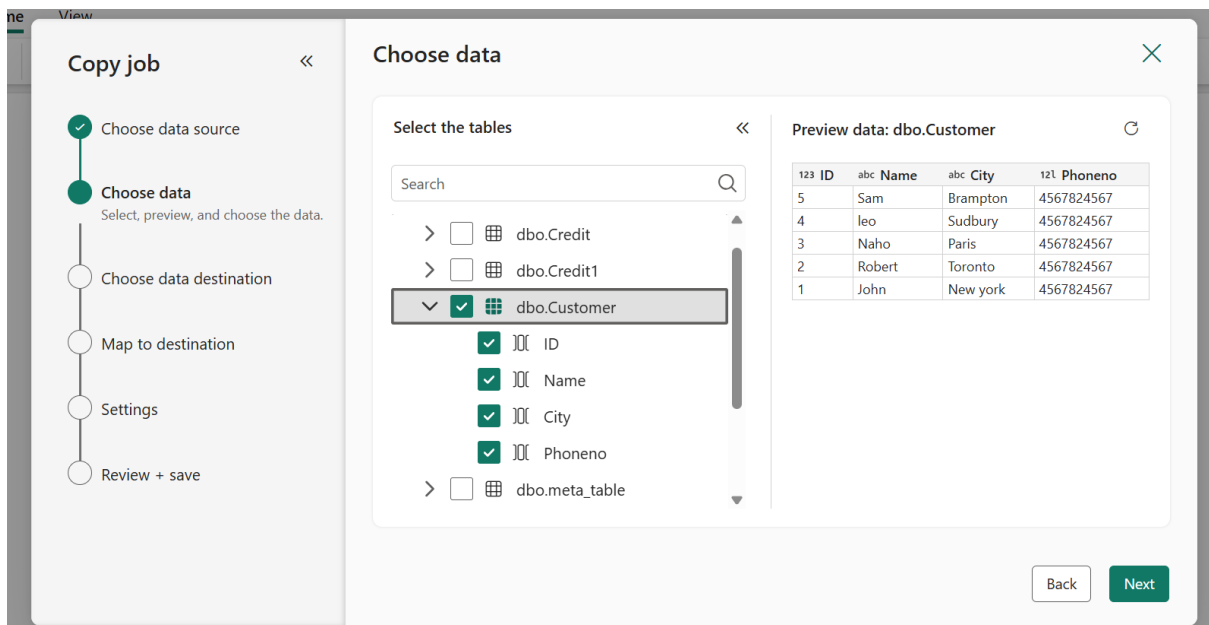
A **Copy Job** in Microsoft Fabric refers to a data movement task that facilitates the transfer of data from one or more source systems to one or more destination systems. It is commonly used within **Data Pipelines** to support ETL (Extract, Transform, Load) or ELT (Extract, Load, Transform) scenarios.

To create a copy job, go to workspace -> new item -> search for copy job





Copy job created, now we have to load source, destination and all other details.



Choosing my Source as warehouse, and loading customer table.

Copy job

Choose data source

Choose data

Choose data destination

Map to destination
 Select and map to folder path or table.

Settings

Review + save

Map to destination

Files

Tables

Append

Append new values to existing table

Edit update method

Table mapping

Filter by source name

Source	Destination
dbo.Customer	Customer_table

Back

Next

Selected my destination as lake house.

Table mapping

Filter by source name

Source

Destination

dbo.Customer



Customer_table



Edit column mapping

Copy job

Choose data source

Choose data

Choose data destination

Map to destination

Settings
 Configure copy job settings.

Review + save

Settings

Copy job mode

Choose how you want the data to be copied. This copy mode will be applied every time you run the job, which can be run once or multiple times. After this new copy job item is created, you can then schedule how often the job is run and monitor statuses.

Full copy

Copy all data at once

Incremental copy (Preview)

Initially copies all data, subsequent runs copy only changes

Stream copy

Copy data streams from real-time events

Incremental column


Refresh

Back

Next

Choosing data to load in incremental load.

Incremental column

 Refresh

Source Table

Incremental column

 dbo.Customer ⓘ

 123 ID 

ID column is being used for incremental load.

Review + save



Summary

→ Source



warehouse_fabric
Warehouse

Incremental copy →

Connection name

warehouse_fabric

Table name

dbo.Customer

↳ Destination



Fabric_lakehouse
Lakehouse

Back

Save + Run

Copy job



- ✓ Choose data source
- ✓ Choose data
- ✓ Choose data destination
- ✓ Map to destination
- ✓ Settings
- Review + save**
Confirm Copy summary

Review + save



Connection name

Fabric_lakehouse

Table name

Customer_table

☒ Start data transfer immediately ⓘ

Run options

☐ Run once ☒ Run on schedule

Repeat

Every minute

Back

Save + Run

Copy job

- Choose data source
- Choose data
- Choose data destination
- Map to destination
- Settings
- Review + save**
Confirm Copy summary

Review + save

☒ Start data transfer immediately

Run options

☐ Run once
 ☒ Run on schedule

Repeat
 By the minute

Every
 15 minute(s)

Start date and time
 4/11/2025, 9:54:48 AM

End date and time
 1/1/9999, 5:00:00 AM

Time zone
 Eastern Time (US & Canada) (UTC-5)

Back

Save + Run

As we are loading data incrementally, we can run on schedule, click save and run.

loadlktowh

Search

Trial: 50 days left

Home

View

Cancel

Schedule

Choose data source

Edit mapping

Advanced settings

View run history

Copy every 15 minute(s)
Starting April 11, 2025 9:54 AM

Source

warehouse_fabric Warehouse

Incremental copy

Destination

Fabric_Lakehouse Lakehouse

Results

Refresh

Status: In progress

Rows read: 0

Rows written: 0

Throughput: 0 byte/s

More

Source → Destination	Status	Rows read	Rows written	Duration	Run start	Run end
dbo.Customer → Custo...	In progress	-	-	1 sec	4/11/2025, 9:56:54 AM	-

loadlktowh

Search

Trial: 50 days left

Home

View

Run

Schedule

Choose data source

Edit mapping

Advanced settings

View run history

Copy every 15 minute(s)
Starting April 11, 2025 9:54 AM

Source

warehouse_fabric Warehouse

Incremental copy

Destination

Fabric_Lakehouse Lakehouse

Results

Refresh

Status: Succeeded

Rows read: 5

Rows written: 5

Throughput: 10 bytes/s

More

Source → Destination	Status	Rows read	Rows written	Duration	Run start	Run end
dbo.Customer → Custo...	Succeeded	5	5	47 sec	4/11/2025, 9:56:54 AM	4/11/2025, 9:57:41 AM

Run Succeeded

Successfully ran loadlktowh (Copy job).

Pipeline ran successfully.

Checking lake house if the data is loaded correctly or not.

The screenshot shows the Microsoft Fabric Lakehouse Explorer interface. On the left, the 'Explorer' pane shows the 'Fabric_lakehouse' structure with 'Tables' expanded, listing 'credit_table', 'customer_table', 'Merge_Op', and 'Unidentified'. The 'customer_table' is selected. The main pane displays the table data with 5 rows. The table has columns: ID, Name, City, and Phoneno.

ID	Name	City	Phoneno
5	Sam	Brampton	4567824567
4	leo	Sudbury	4567824567
3	Naho	Paris	4567824567
2	Robert	Toronto	4567824567
1	John	New york	4567824567

Data loaded successfully.

As we opted for incremental load, will add/insert some new rows to table in data warehouse to check if it will work or not.

The screenshot shows the Microsoft Fabric Lakehouse Explorer interface with a SQL query being executed. The query is:

```
1 insert into [warehouse_fabric].[dbo].[Customer] values(6,'Will','Newyork','2897653897')
2 insert into [warehouse_fabric].[dbo].[Customer] values(7,'Jin','North york','2345672445')
3 insert into [warehouse_fabric].[dbo].[Customer] values(8,'Sandra','Dandas','4267812389')
```

The 'Messages' pane shows the execution details:

```
10:02:45 AM Started executing on line 1
Statement ID: (AEEBF6FB-49B1-40DC-86F6-0C86437A2861) | Query hash: 0xA6BC570686240183 | Distributed request ID: (68ADC006-3711-4C1A-8586-D652F72B50CF) (1 records affected)
Statement ID: (41DDC105-9100-40E0-AD21-7254C27F8ADA) | Query hash: 0xA6BC570686240183 | Distributed request ID: (19FCC776-4B5E-4618-A6BF-0A0AD288AADC) (1 records affected)
```

Data inserted into table.

Pipeline will run after 15 minutes and then let's check.

The screenshot shows the Microsoft Fabric Lakehouse Explorer interface with a pipeline configuration and its execution results. The pipeline is named 'Copy every 15 minute(s)' and is scheduled to start on April 11, 2025, at 9:54 AM. The pipeline has two components: 'Source' (warehouse_fabric Warehouse) and 'Destination' (Fabric_Lakehouse Lakehouse), connected by an 'Incremental copy' activity. The 'Results' pane shows the pipeline is 'In progress' with 0 rows read and 0 rows written. The 'Run history' table shows the pipeline is 'In progress' with a duration of 3 sec and a run start time of 4/11/2025, 10:11:43 AM.

Source	Destination	Status	Rows read	Rows written	Duration	Run start	Run end
dbo.Customer	Custo...	In progress	-	-	3 sec	4/11/2025, 10:11:43 AM	-

Pipeline started running.

The screenshot shows the 'View run history' page for a pipeline. At the top, a notification states 'Copy every 15 minute(s) Starting April 11, 2025 9:54 AM'. Below this, a visual map shows the 'Source' (warehouse_fabric Warehouse) connected to the 'Destination' (Fabric_lakehouse Lakehouse) via an 'Incremental copy' activity. The 'Results' section shows a table with the following data:

Source → Destination	Status	Rows read	Rows written	Duration	Run start	Run end
dbo.Customer → Custo...	✓ Succeeded	3	3	1 min 6 sec	4/11/2025, 10:11:43 AM	4/11/2025, 10:12:50 AM

Pipeline ran successfully.

Lets check, if the data is loaded successfully.

Data is stored in delta format.

The screenshot shows the 'Explorer' view of the 'Fabric_lakehouse'. Under the 'Unidentified' folder, the 'Customer_table' is selected. The table's contents are displayed as follows:

Name	Date modified	Type	Size
_delta_log	4/11/2025, 9:5...	Folder	-
ba0e6521-e7e0-4d85-9d07-472b1cf3c2a.par...	4/11/2025, 9:5...	parquet	1 KB
e43976b0-8abc-4cfc-8580-895923cb1b3b.parquet	4/11/2025, 10:...	parquet	1 KB

Load JSON data to csv file

Upload JSON file into lake house.

The screenshot shows the 'Upload files' dialog box in the Azure Data Lake Explorer. The 'Files/' section shows a file named 'Sample.JSON'. The 'Upload' button is visible. In the background, the 'credit_table' is visible, showing a list of records with columns 'id', 'Credit_id', and 'ABC Credi'.

id	Credit_id	ABC Credi
1	5	John
2	6	Robert
3	7	Ryan
4	8	Leo
5	1	Deepthi
6	2	Rahul
7	3	Anjana
8	4	Shriya
9	1	Deepthi
10	2	Rahul
11	3	Anjana
12	4	Shriya
13	1	Deepthi
14	3	Anjana

File uploaded successfully.

The screenshot shows the Microsoft Fabric Lakehouse interface. On the left, the 'Explorer' pane displays the 'Fabric_lakehouse' structure with 'Tables', 'Files', 'container1', and 'dbo'. The main area shows a table named 'credit_table' with columns 'Credit_id', 'ABC', and 'Credi'. An 'Upload files' dialog is open on the right, showing a file named 'Sample.JSON' being uploaded to 'Fabric_lakehouse' with a size of 412 B / 412 B. The dialog also includes an 'Overwrite if files already exist' checkbox and an 'Upload' button.

Credit_id	ABC	Credi
1	5	John
2	6	Robert
3	7	Ryan
4	8	Leo
5	1	Deepthi
6	2	Rahul

Create new notebook, go to workspace-> new item -> notebook

The screenshot shows the Microsoft Fabric workspace interface. The 'New item' dialog is open, showing options to create a 'Notebook' or a 'Dataflow'. The 'Notebook' option is selected, and the dialog provides instructions on how to use notebooks for exploring, analyzing, and visualizing data. The workspace list on the left includes items like 'Dataflow', 'df_lakehouse', 'Fabric_lakehouse', 'Fabric_NB', 'loadlktowh', 'Notebook 2', 'pl_dataflow', and 'pl_Dataflow1'.

The screenshot shows the Microsoft Fabric Notebook interface. The notebook is titled 'Notebook 3' and is in the 'Saved' state. The 'Explorer' pane on the left shows 'Data items' and 'Resources'. The main area displays a code cell with the following text:

```
1 # Welcome to your new notebook
2 # Type here in the cell editor to add code!
3
```

The notebook is currently 'Not connected' and 'AutoSave: On'. The status bar at the bottom indicates 'Selected Cell 1 of 1 cells'.

Loading data from lake house.

OneLake catalog

Discover data from your org and beyond and use it to create reports

Filter by keyword

Name	Owner	Refreshed	Location	Endorsement	Sensitivity
Fabric_lakehouse	deefabric	—	fabricworkspace	—	—
second_lh	deefabric	—	newws	—	—
DataflowsStagingLakehouse	deefabric	—	fabricworkspace	—	—

```
1 from pyspark.sql.functions import explode, col
2
```

[1] ✓ 9 sec - Session ready in 9 sec 418 ms. Command executed in 483 ms by deefabric on 10:35:21 AM, 4/11/25

PySpark (Python)

```
1 json_path = "Files/Sample.JSON"
2
```

[2] ✓ <1 sec - Command executed in 408 ms by deefabric on 10:36:46 AM, 4/11/25

PySpark (Python)

```
1 df = spark.read.json(json_path, multiline=True)
2 df.show(truncate=False)
3
```

[4] ✓ <1 sec - Command executed in 922 ms by deefabric on 10:38:01 AM, 4/11/25

PySpark (Python)

Spark jobs (2 of 2 succeeded) Resources Log

id	items	name
1	[{I1, Laptop, 1200}, {I2, Mouse, 25}]	Alice
2	[{I3, Monitor, 300}]	Bob

```
1 df_exploded = df.withColumn("item", explode(col("items")))
2 df_flat = df_exploded.select(
3     col("id"),
4     col("name"),
5     col("item.id").alias("item_id"),
6     col("item.name").alias("item_name"),
7     col("item.price").alias("item_price")
8 )
9
10 df_flat.show(truncate=False)
```

[5] ✓ <1 sec - Command executed in 960 ms by deefabric on 10:38:49 AM, 4/11/25

PySpark (Python)

Spark jobs (1 of 1 succeeded) Resources Log

id	name	item_id	item_name	item_price
1	Alice	I1	Laptop	1200
1	Alice	I2	Mouse	25
2	Bob	I3	Monitor	300

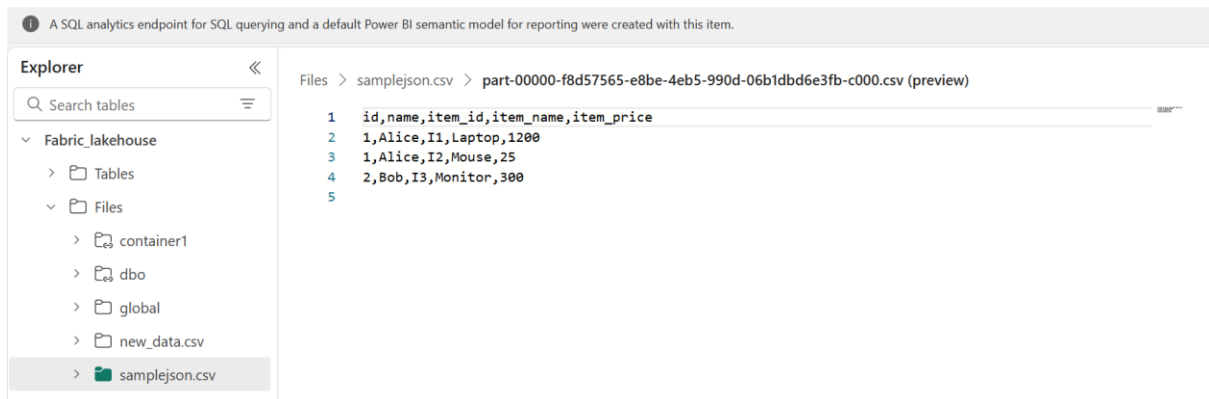
```
1 df_flat.write.mode("overwrite").option("header", "true").csv("Files/samplejson.csv")
2
```

[6] ✓ 2 sec - Command executed in 2 sec 485 ms by deefabric on 10:40:13 AM, 4/11/25

PySpark (Python)

Spark jobs (1 of 1 succeeded) Resources Log

Data loaded successfully as CSV file.



The **explode ()** function in PySpark is used to **flatten** an array or map column, turning each element into a separate row.

What It Does

- If a row contains an **array**, explode () creates a new row **for each element** in that array.
- It's commonly used to **flatten nested structures** such as arrays in JSON data.