

Implemented Unity Catalog, Auto Loader, and Automated Data Ingestion Using Databricks Workflows

Mohammed Ismail Shaikh

Prerequisites

I created five containers in ADLS Gen2: bronze, silver, gold, raw, and metastore. Using Azure Data Factory (ADF), I ingested the following files into the data lake: netflix_cast.csv, netflix_category.csv, netflix_countries.csv, and netflix_directors.csv. Additionally, I manually uploaded the netflix_titles.csv file directly into the raw container.

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

Home > netflixprojectdl08

netflixprojectdl08 | Containers

Storage account

Search

«

+ Add container

↑ Upload

↻ Refresh

🗑 Delete

🔒 Change access level

↺ Restore containers

⚙ Edit columns

Overview

Activity log

Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage browser

Partner solutions

Resource visualizer

Search containers by prefix

Only show active containers

Sorting all 6 items

<input type="checkbox"/>	Name ↑	Last modified	Anonymous access level	Lease state
<input type="checkbox"/>	\$logs	7/27/2025, 12:36:54 PM	Private	Available
<input type="checkbox"/>	bronze	7/27/2025, 12:39:33 PM	Private	Available
<input type="checkbox"/>	gold	7/27/2025, 12:39:05 PM	Private	Available
<input type="checkbox"/>	metastore	7/28/2025, 12:55:31 AM	Private	Available
<input type="checkbox"/>	raw	7/27/2025, 12:38:51 PM	Private	Available
<input type="checkbox"/>	silver	7/27/2025, 12:38:59 PM	Private	Available

bronze

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

Search blobs by prefix (case-sensitive)

Showing all 5 items

<input type="checkbox"/>	Name	Last modified
<input type="checkbox"/>	netflix_cast	7/27/2025, 2:32:49 PM
<input type="checkbox"/>	netflix_category	7/27/2025, 2:32:49 PM
<input type="checkbox"/>	netflix_countries	7/27/2025, 2:32:52 PM
<input type="checkbox"/>	netflix_directors	7/27/2025, 2:32:49 PM

raw

Authentication method: Access key (Switch to Microsoft En

Search blobs by prefix (case-sensitive)

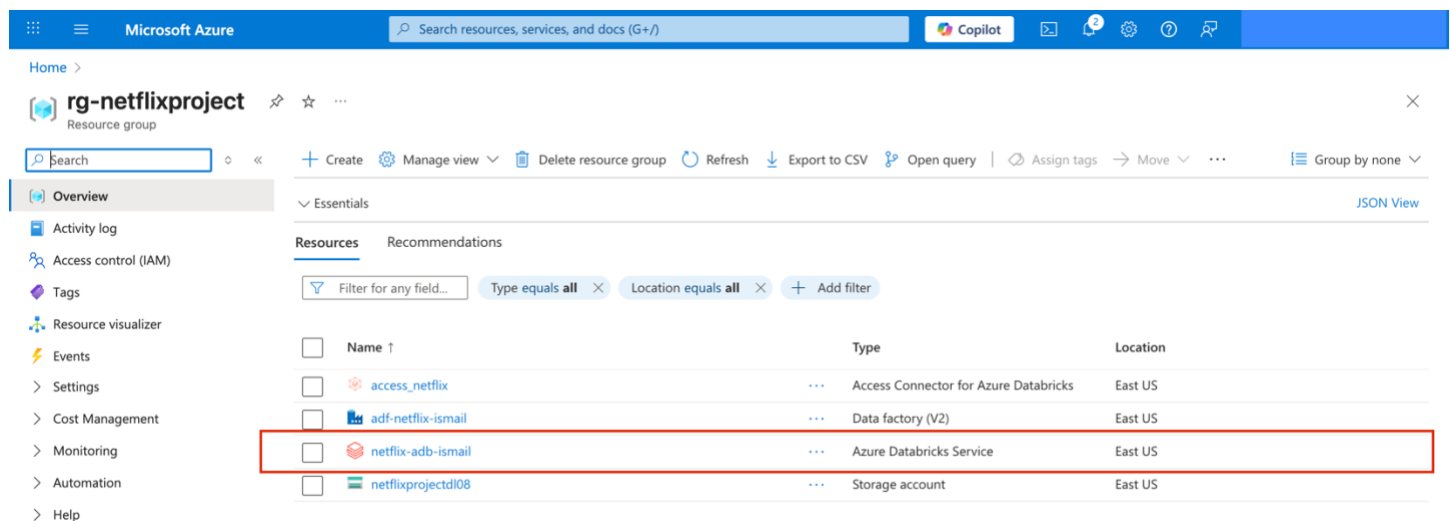
Showing all 1 items

<input type="checkbox"/>	Name
<input type="checkbox"/>	netflix_titles.csv

Setting Up Unity Catalog

Create Databricks resource:

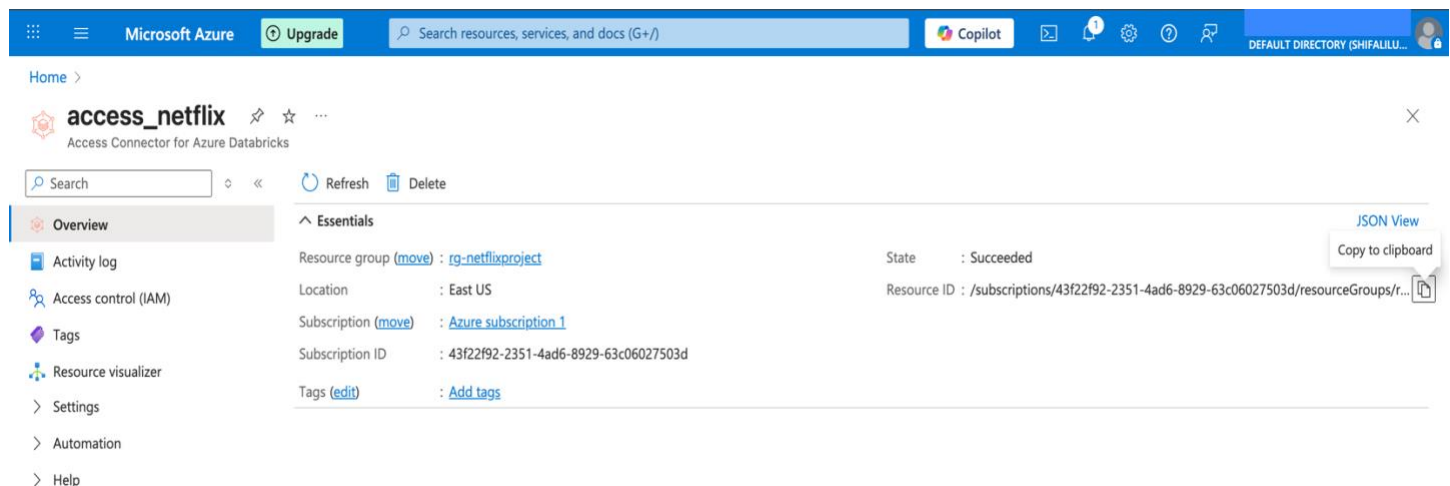
I set up a Databricks workspace called “*netflix-adb-ismail*” within my Azure resource group.



Name	Type	Location
access_netflix	Access Connector for Azure Databricks	East US
adf-netflix-ismail	Data factory (V2)	East US
netflix-adb-ismail	Azure Databricks Service	East US
netflixprojectd108	Storage account	East US

How I created the Access connector and assigned the role:

Before setting up unity catalog, I created a resource called “Access connector for Azure Databricks”, which acts as a connection between the Databricks and Azure Data Lake storage.



Resource group (move) : [rg-netflixproject](#)

Location : East US

Subscription (move) : [Azure subscription 1](#)

Subscription ID : 43f22f92-2351-4ad6-8929-63c06027503d

Tags (edit) : [Add tags](#)

State : Succeeded

Resource ID : /subscriptions/43f22f92-2351-4ad6-8929-63c06027503d/resourceGroups/r...

The Resource ID which gets generated by “Access Connector for Azure Databricks” is an authentication ID which allows the Databricks to access the Azure Data Lake.

Then I assigned the access connector the “Storage Blob Data Contributor” role to the storage account. By this databricks will be able to access the storage account.

Microsoft Azure | Upgrade | Search resources, services, and docs (G+)

Home > rg-netflixproject

rg-netflixproject | Access control (IAM) ☆ ...

Search

Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Events
Settings
Cost Management
Monitoring
Automation
Help

Check access | **Role assignments** | Roles | Deny assignments | Classic administrators

Number of role assignments for this subscription: 3 / 4000

Privileged: 1

View assignments

Search by name or email

Type: All | Role: All | Scope: All scopes | Group by: Role

All (2) | Job function roles (1) | Privileged administrator roles (1)

Name	Type	Role	Scope	Condition
Owner (1)				
ss [redacted]	User	Owner	Subscription (Inherited)	None
Storage Blob Data Contributor (1)				
access_netflix	Managed identity	Storage Blob Data Contributor	This resource	Add

Showing 1 - 2 of 2 results.

How I created Unity Catalog Metastore:

I created a Unity Catalog metastore using the Databricks account console. During the setup process, I used the Resource ID from the “Access Connector for Azure Databricks” named “*access_netflix*”, which I had created earlier. This allowed Databricks to securely access Azure Data Lake Storage.

I then configured the metastore by providing the ADLS Gen2 path that points to the “*metastore*” container. I created this container specifically to store all the metadata managed by Unity Catalog, including information about schemas, tables, and permissions.

Account

Workspaces
Catalog
Usage
User management
Cloud resources
Previews
Settings

Catalog > netflix_metastore >

netflix_metastore

Configuration | Workspaces

ADLS Gen 2 path

abfss://metastore@netflixprojectdl08.dfs.core.windows.net/8dc86b5c-c1c8-4c7c-b552-c17b82ed612d - Remove

Region

The region for this metastore. You will only be able to assign workspaces in this region to this metastore.

eastus

Metastore Admin

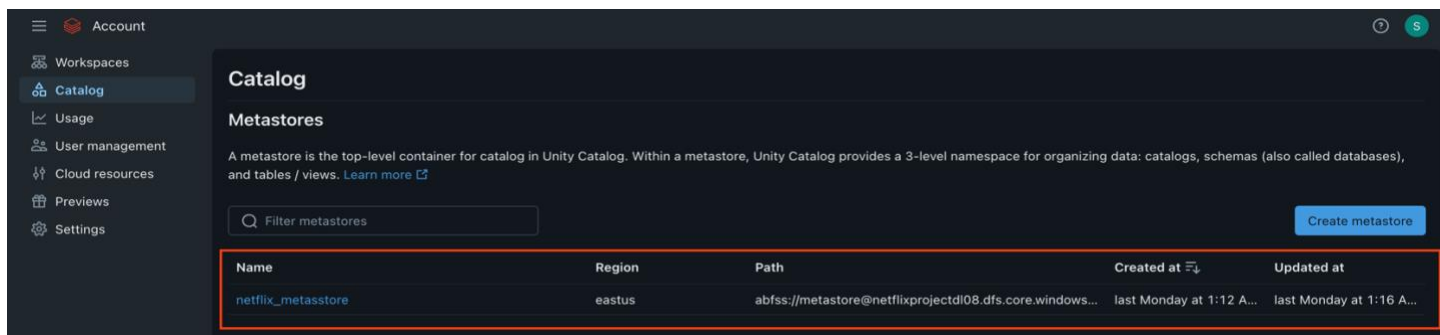
If specified, this highly privileged user or group can manage the privileges or transfer ownership of any object within the metastore. [Learn more](#)

Delta Sharing

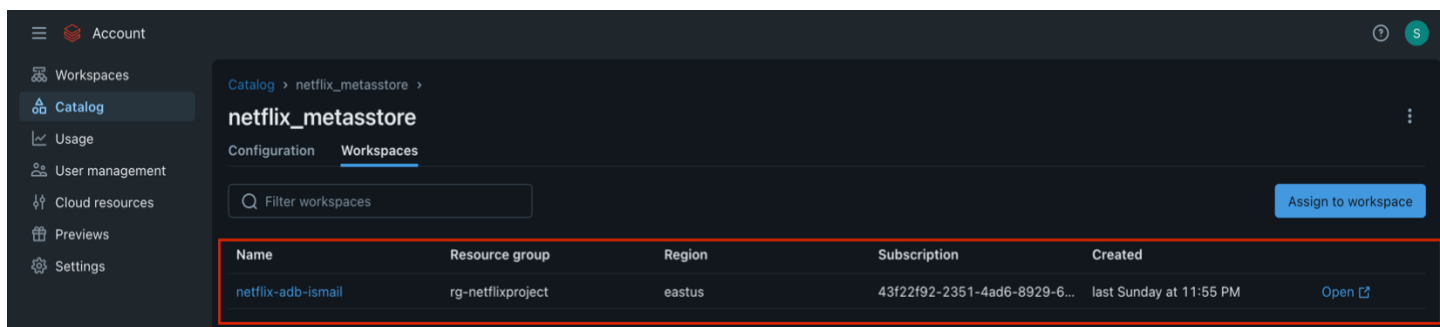
☐ Allow Delta Sharing with parties outside your organization

Workspace assignment

☐ Automatically assign new workspaces in eastus to this metastore

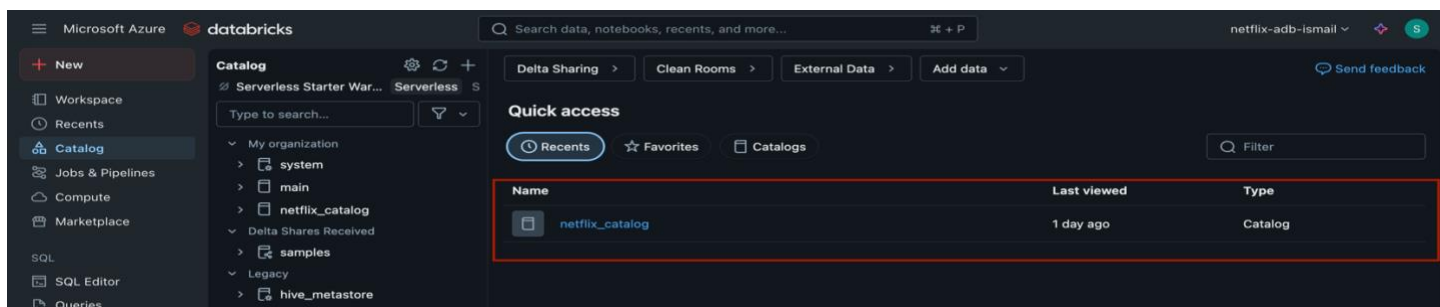


After setting it up, I assigned the Databricks workspace I created i.e. “*netflix-adb-ismail*” to this metastore so it can start using Unity Catalog for data governance and access control.

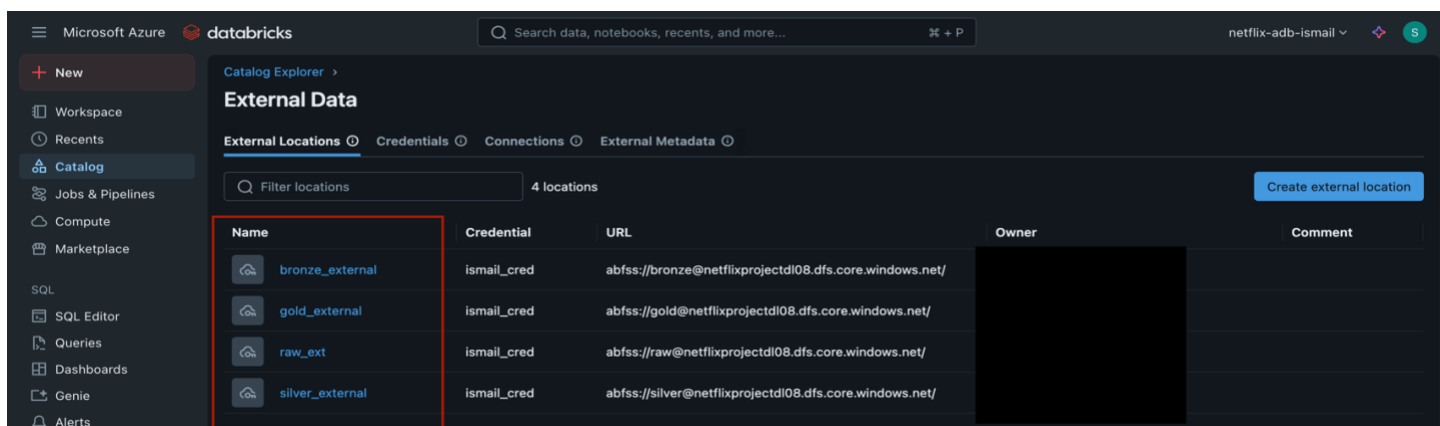


How I Created Catalog in Databricks:

After configuring the meta store in Databricks account console. I created a catalog in Databricks.

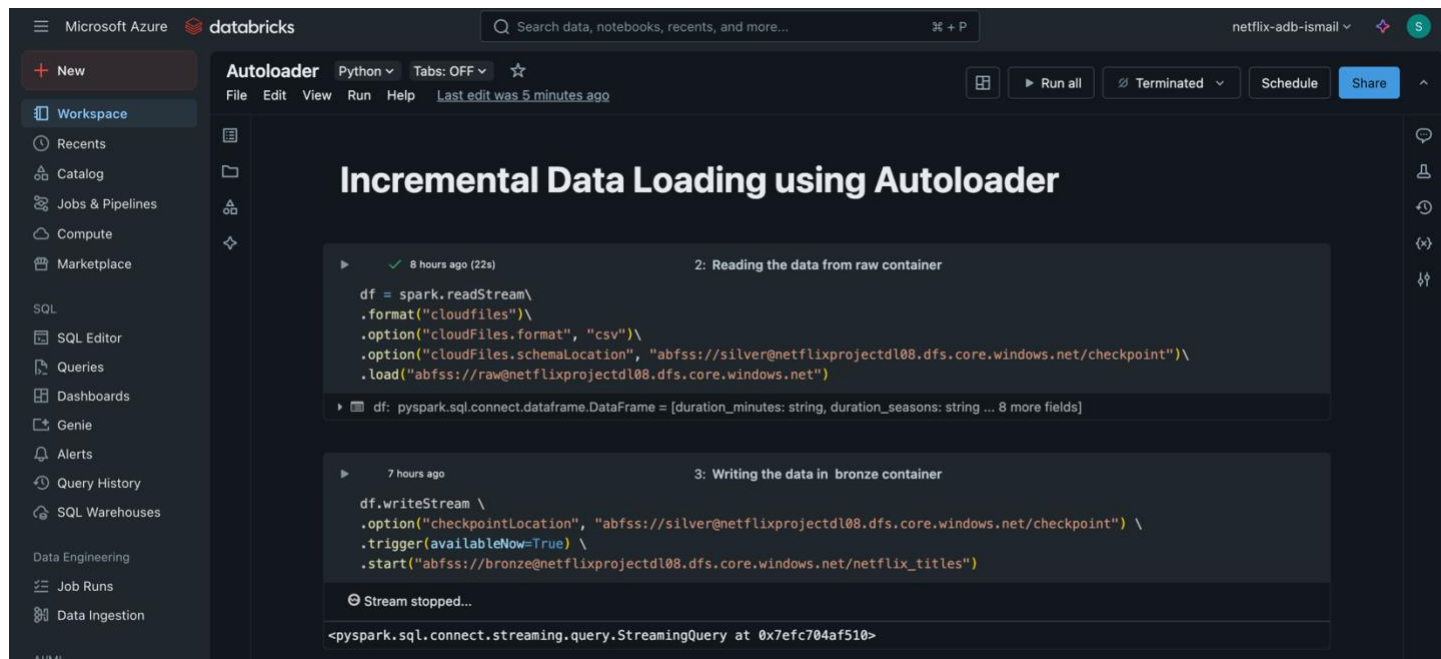


After creating the catalog, I set up external locations in ADLS Gen2 at the container level. This enabled me to access and store data in the “*bronze*”, “*silver*”, “*gold*”, and “*raw*” containers directly from Databricks.

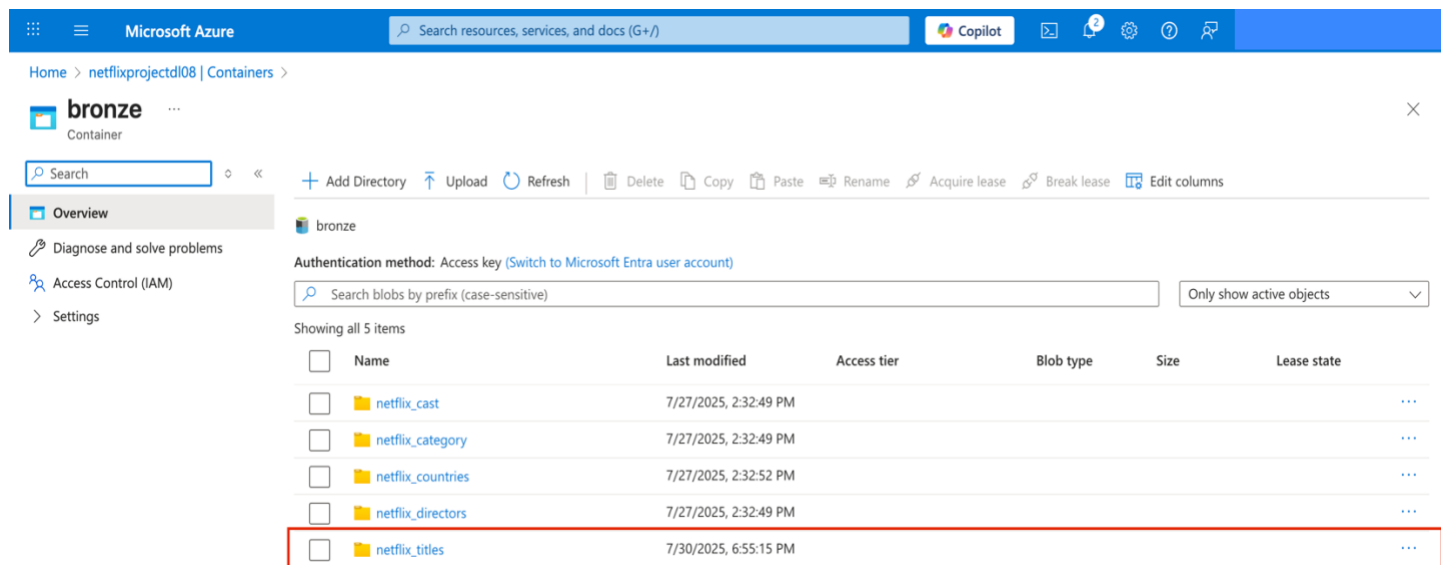


Incremental Data Loading using Autoloader

I used Databricks Auto Loader to continuously detect and ingest new CSV files arriving in the “*raw*” container. To manage schema changes and track progress, I set a checkpoint location in the “*silver*” container. This ensures only new files are processed without duplication. The incoming data is then written as a Delta table in the bronze container.



Once the incremental load ran successfully, the “*netflix_titles*” file from the “*raw*” container was ingested into the “*bronze*” container. The schema information was also captured and stored in the checkpoint folder inside the “*silver*” container to help track structure and manage future data loads.

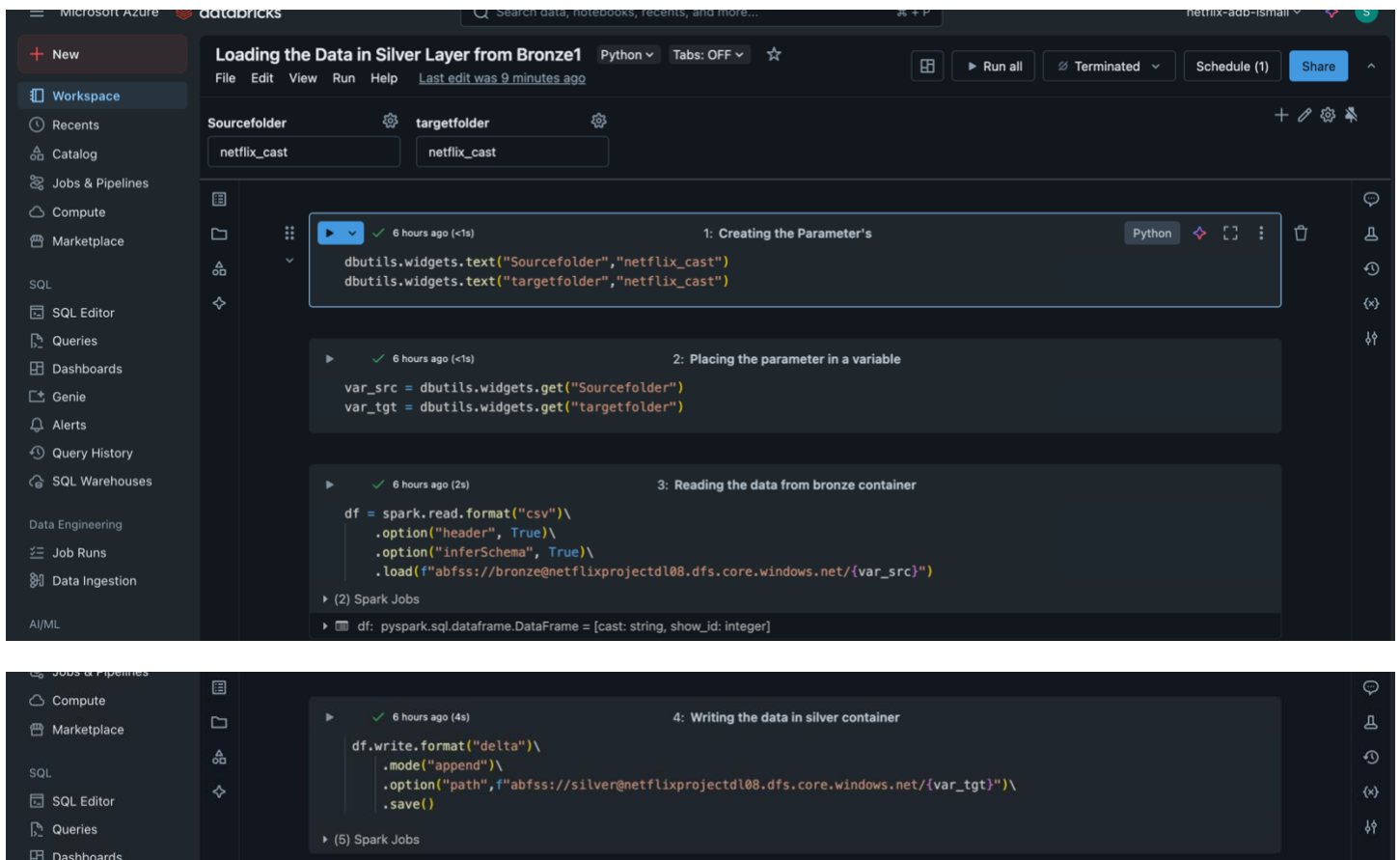


Automating Data Pipelines with Databricks Workflows

How I created the parameter in Notebook:

With all the files in the bronze layer, I needed to move them to the silver layer and perform data quality checks. Instead of processing each file manually, I set up parameters in the notebook using “*dbutils.widgets*” to automate the reading and writing tasks. This approach made the workflow more efficient and easier to manage.

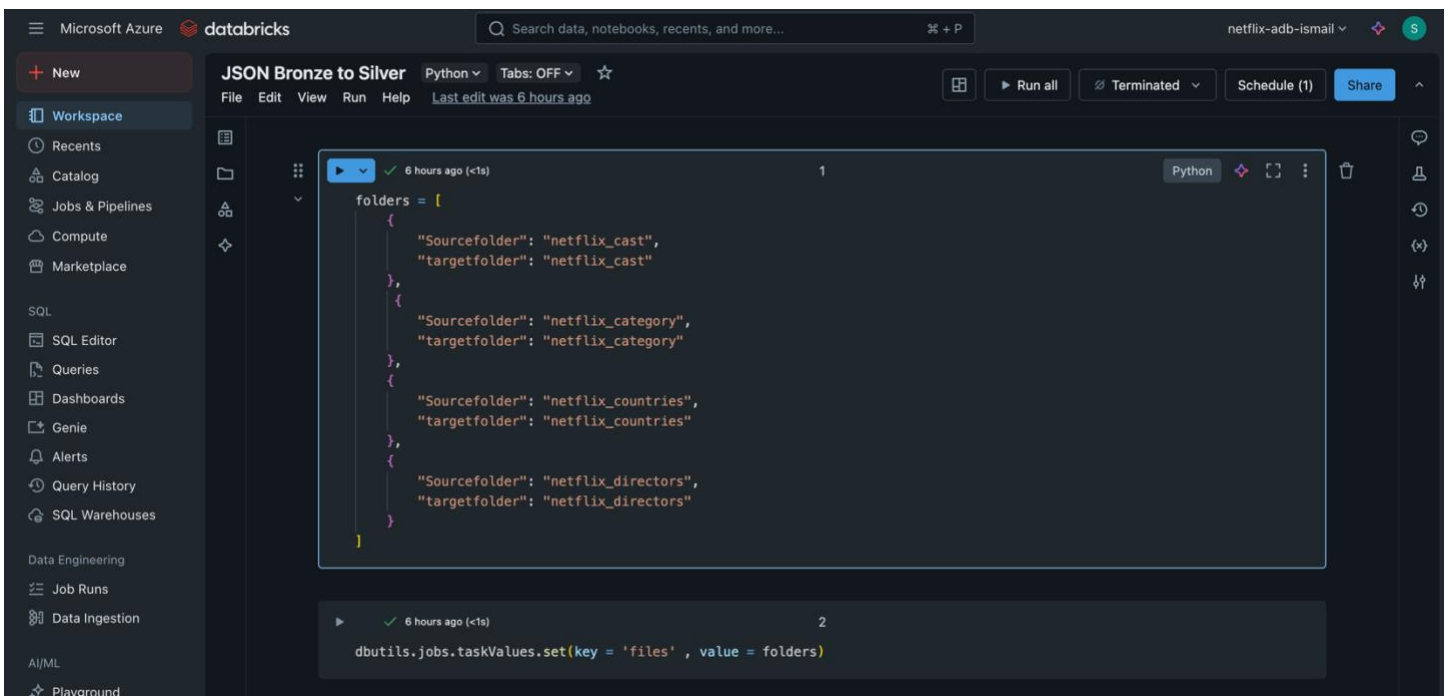
I set the default source and target parameters as “*netflix_cast*” and stored them in variables “*var_src*” and “*var_tgt*”. This allowed me to use these variables in my read and write scripts, so the data could be dynamically read from “*bronze*” container and written to the “*silver*” container.



How I created the lookup array:

With the parameters and script ready, I created a lookup array in the form of a JSON file that defined the source and target locations. This setup allowed me to easily manage multiple data paths dynamically within the pipeline.

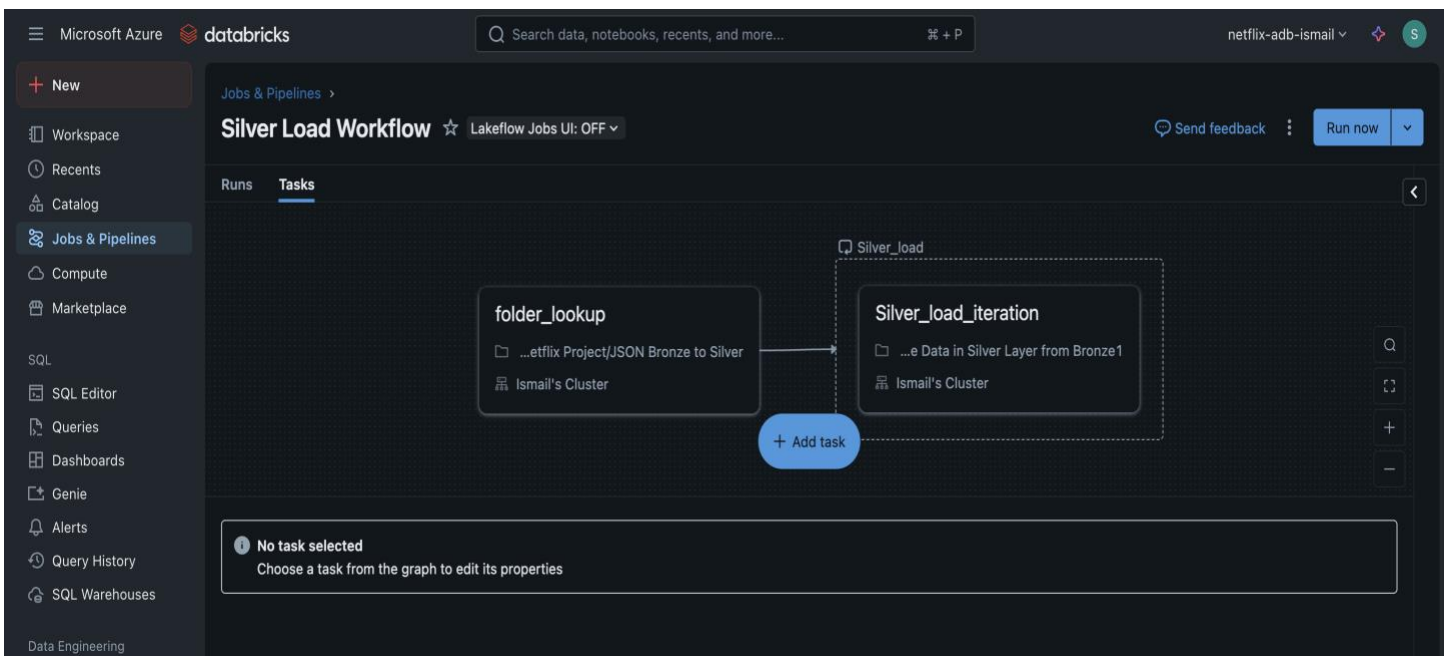
I used “*dbutils.jobs.taskValues*” to set key-value pairs, including the array name “*folders*”, which stored multiple parameters.



How I created the Databricks workflow:

I created a Databricks job called “*folder_lookup*” which runs a notebook responsible for handling the JSON file containing folder configurations. Then, I created a separate job called “*Silver_load_iteration*” that runs the notebook where I defined parameters and implemented the data read and write operations.

I added a “*lookup task*” at the start of the “*Silver_load_iteration*” job so that it runs once for every path defined in the “*folder_lookup*” notebook. This way, the job processes each folder dynamically without manual intervention.



Success

After running the pipeline, it completed successfully, and all the folders were loaded from the bronze container into the silver container.

Microsoft Azure

databricks

Search data, notebooks, recents, and more...

netflix-adb-ismail

New

Workspace

Recents

Catalog

Jobs & Pipelines

Compute

Marketplace

SQL

SQL Editor

Queries

Dashboards

Genie

Alerts

Query History

SQL Warehouses

Data Engineering

Job Runs

Data Ingestion

AI/ML

Jobs & Pipelines > New Job Jul 30, 2025, 09:46 PM >

New Job Jul 30, 2025, 09:46 PM run Lakeflow Jobs UI: OFF

Send feedback Delete job run Repair run

Graph Timeline List

folder_lookup

Succeeded · 7s <> 1

...etflix Project/JSON Bronze to Silver

Ismail's Cluster

Silver_load

Silver_load_iteration

Succeeded · 4 scheduled · 50s

...e Data in Silver Layer from Bronze1

Ismail's Cluster

Microsoft Azure

Search resources, services, and docs (G+I)

Copilot

Home > rg-netflixproject > netflixprojectdl08 | Containers >

silver

Container

Search

+ Add Directory

Upload

Refresh

Delete

Copy

Paste

Rename

Acquire lease

Break lease

Edit columns

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

silver

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

Search blobs by prefix (case-sensitive)

Only show active objects

Showing all 5 items

<input type="checkbox"/>	Name	Last modified	Access tier	Blob type	Size	Lease state
<input type="checkbox"/>	checkpoint	7/30/2025, 6:47:43 PM				...
<input type="checkbox"/>	netflix_cast	7/30/2025, 9:59:32 PM				...
<input type="checkbox"/>	netflix_category	7/30/2025, 9:59:45 PM				...
<input type="checkbox"/>	netflix_countries	7/30/2025, 9:59:58 PM				...
<input type="checkbox"/>	netflix_directors	7/30/2025, 10:00:09 PM				...

Add or remove favorites by pressing Cmd+Shift+F