**🟢 Adventure Works Data Engineering Project For My Own Company:**

**Author: Paresh Ranjan Rout**

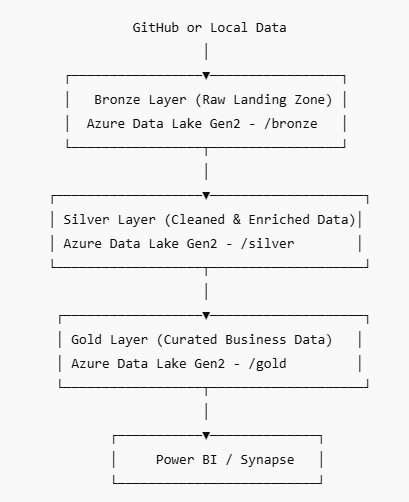
**🎯 Project Goal**

Ingest, process, and serve Adventure Works data in Azure using Medallion Architecture (Bronze, Silver, Gold layers).

**✅ Use Case:**

* Simulate a production pipeline: raw data ➡ cleaned data ➡ curated analytics-ready data.
* Show end-to-end lineage and transformations.
* Enable Power BI and downstream analytics.

**🟢 🛠️ Project Architecture (Medallion)**



**🟢 🎯 Final Best Practices Checklist**

✅ Data is always immutable in Bronze  
✅ No business logic in Bronze (only storage)  
✅ Silver is clean but still granular  
✅ Gold is aggregated and ready for consumption  
✅ Use naming conventions

💥 Perfect first step, build **the backbone of your Medallion Architecture.**  
Let’s Walk through exactly how to create your **first pipeline** to move data from **GitHub, local folder, or API ➡️ Azure Data Lake (Bronze Layer)**.

**🟢 🎯 Pipeline 1: Ingest Data to Bronze Layer**

**✅ Purpose:**

Load raw CSV or JSON files from GitHub, local system, or an API into Azure Data Lake Gen2 (/bronze/) using Azure Data Factory (ADF).

**📦 1. ✅ Decide the Data Source Type:**

Here are the common types:

**Option 1: GitHub (public CSV/JSON files)**

* Use the **HTTP connector** in ADF.
* Source URL like:  
  https://raw.githubusercontent.com/your-repo/data/sales.csv

**🛠️ 2. ✅ Create the Pipeline in ADF**

**🔹 Pipeline Name: pipeline\_ingest\_to\_bronze**

**🔹 Activities to use:**

* **Copy Activity**
  + **Source:** GitHub (HTTP) / Local Blob / REST API
  + **Sink:** Azure Data Lake Gen2 → /bronze/<entity\_name>/

**🛠️ Step-by-Step – ADF Copy Activity**

**🔹 Step 1: Create Linked Services**

**✅ Source: GitHub (HTTP)**

Go to **Manage > Linked Services > New**

* **Connector**: **HTTP**
* **Name**: **ls\_http\_github**
* **Base URL**: **https://github.com/PARESHRANJAN299/**
* **Authentication**: Anonymous (if public repo) ---🡪 **Need to do more study**
* **Test connection** → ✅(It Should)

**✅ Sink: Azure Data Lake Gen2**

Create or use an existing:

* **Connector**: Azure Data Lake Storage Gen2
* **Name**: ls\_adls\_bronze
* **Authentication method**: Managed Identity / Account Key

**🔹 Step 2: Create Dataset**

**✅ Source Dataset (CSV from HTTP)**

* Name: ds\_github\_sales
* Linked Service: ls\_http\_github
* File path: (leave empty if passing dynamically)
* Format: Delimited Text
  + First row as header: ✅ Yes
  + Column delimiter: ,

**✅ Sink Dataset (ADLS Gen2 CSV)**

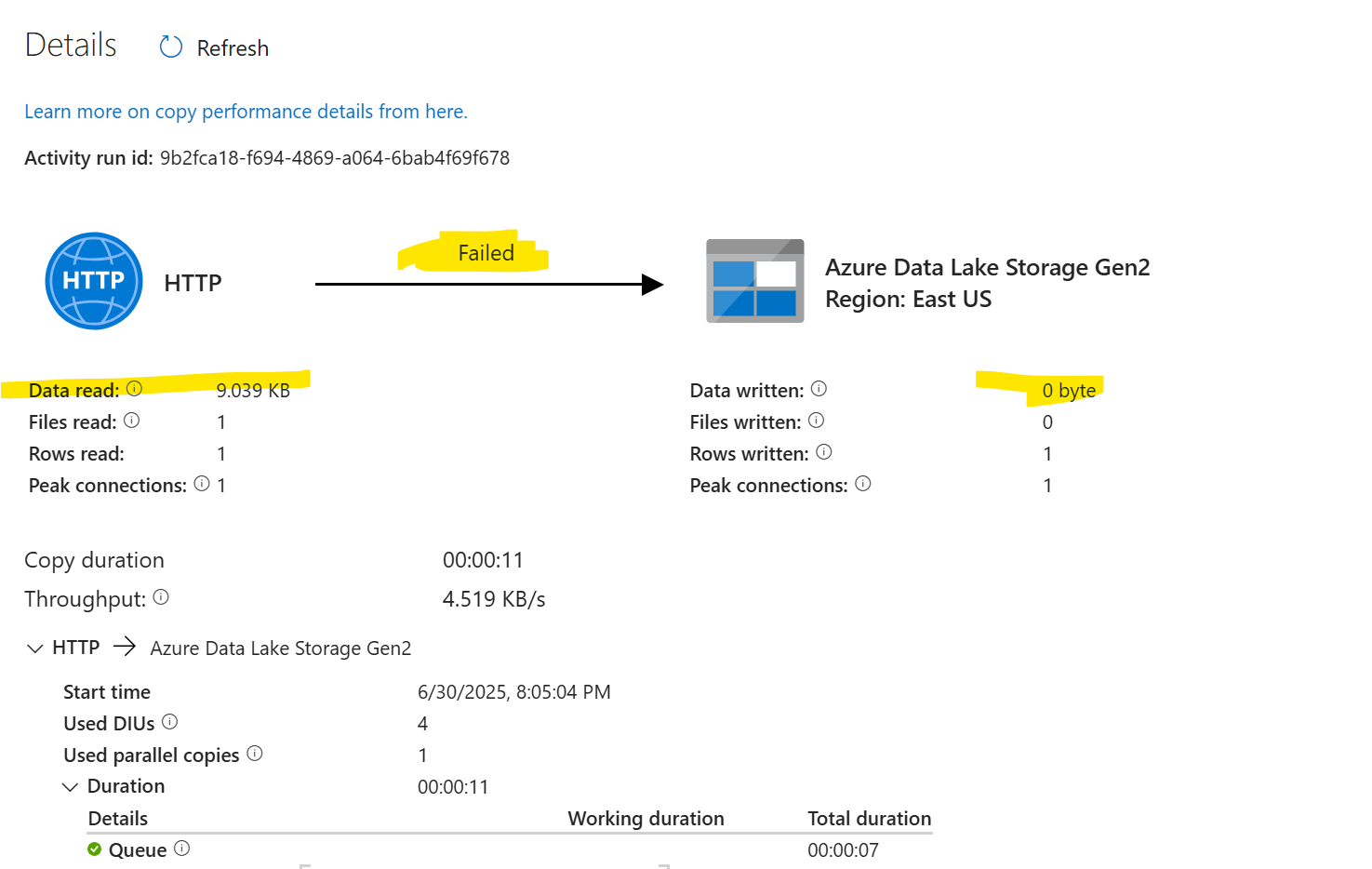
* Name: ds\_adls\_bronze\_sales
* Linked Service: ls\_adls\_bronze
* File path: bronze/sales/
* Filename: sales\_2024\_06\_08.csv
* Format: Delimited Text

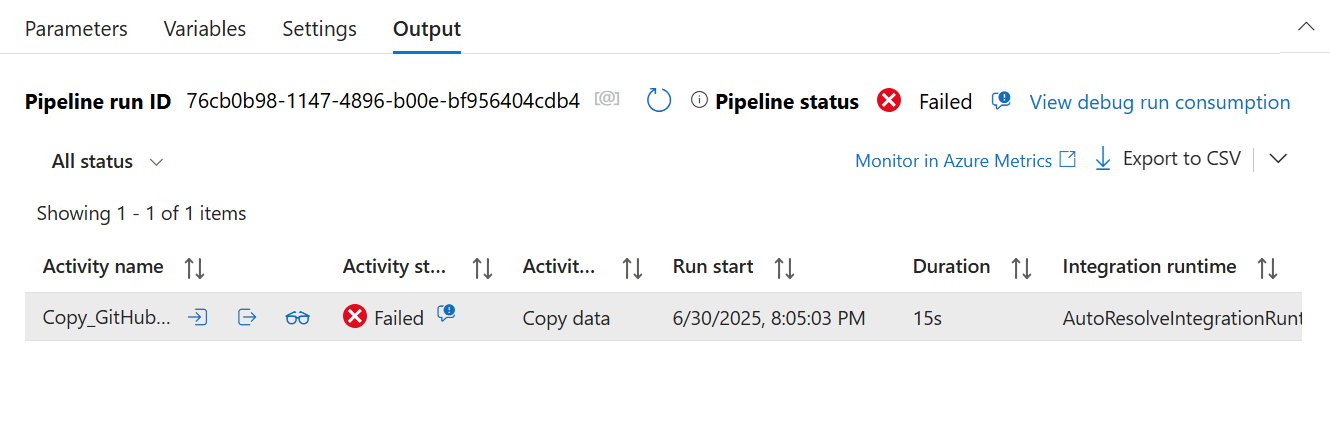
**🔹 Step 3: Add Copy Activity to Pipeline**

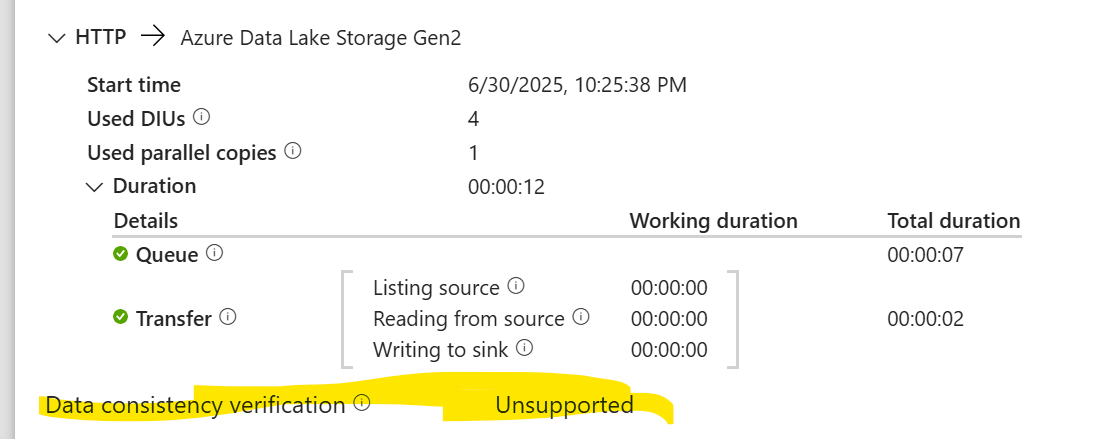
**🎯 Copy Activity Settings:**

| **Tab** | **Setting** |
| --- | --- |
| **Source** | Dataset: ds\_github\_sales |
| **Sink** | Dataset: ds\_adls\_bronze\_sales |
| **Mapping** | Let it auto-map OR specify columns manually (after preview) |
| **Fault Tolerance** | Retry count: 3, Log file path (optional) |
| **Name** | Copy\_GitHub\_To\_Bronze\_Sales |

**Error Get: -**





****

**🔍 Let’s read what you posted:**

**Observed:**

* Data read ✅: 57.828 KB
* Rows read ✅: 167
* Data written ❌: 0 bytes
* Rows written ❌: 0
* Error summary: *Data consistency verification — Unsupported*

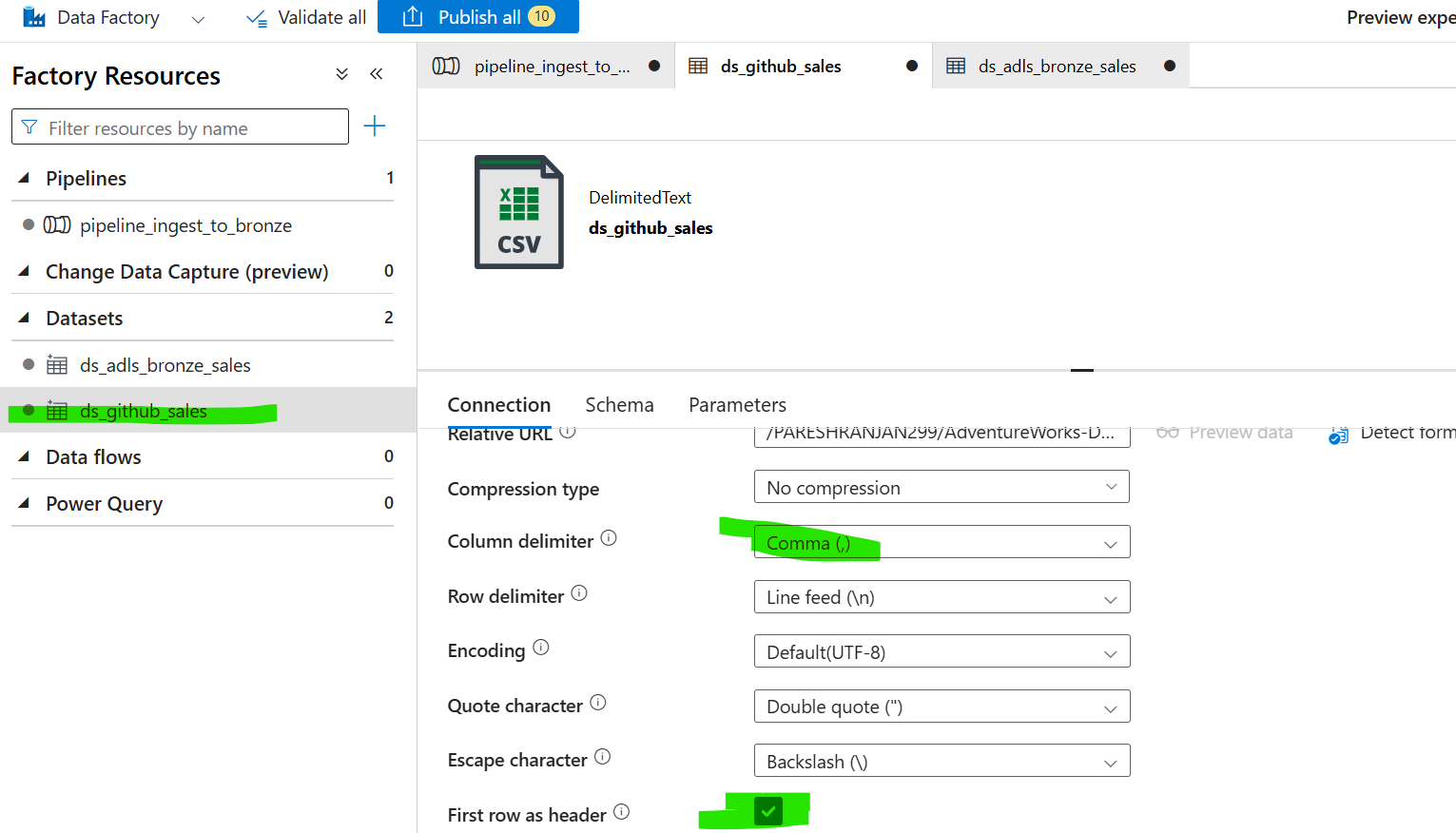
✅ This tells us:

1. **The HTTP download worked fine**—ADF successfully fetched the CSV.
2. **Writing to ADLS Gen2 failed.**

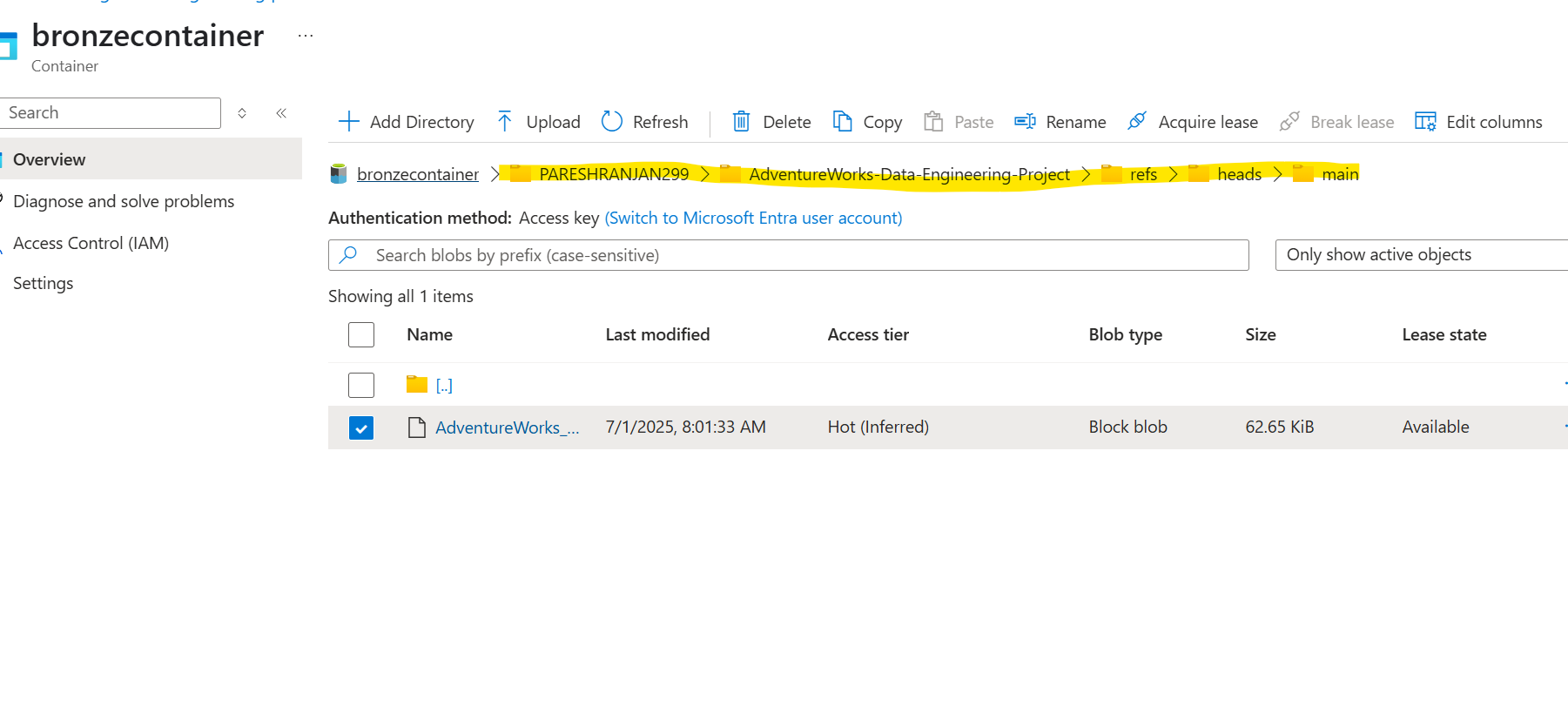
Fixed this error following below step: -

✅ Perfect—**this configuration looks correct now!**

🔹 **Column delimiter:** Comma (,)  
🔹 **Row delimiter:** Line feed (\n)  
🔹 **Encoding:** UTF-8  
🔹 **First row as header:** Enabled ✅



**Again – Multiple directories has been created’’**



**🎯 How to write to a single file or avoid multiple folders**

✅ **1️⃣ In the Sink Settings (Copy Activity)**

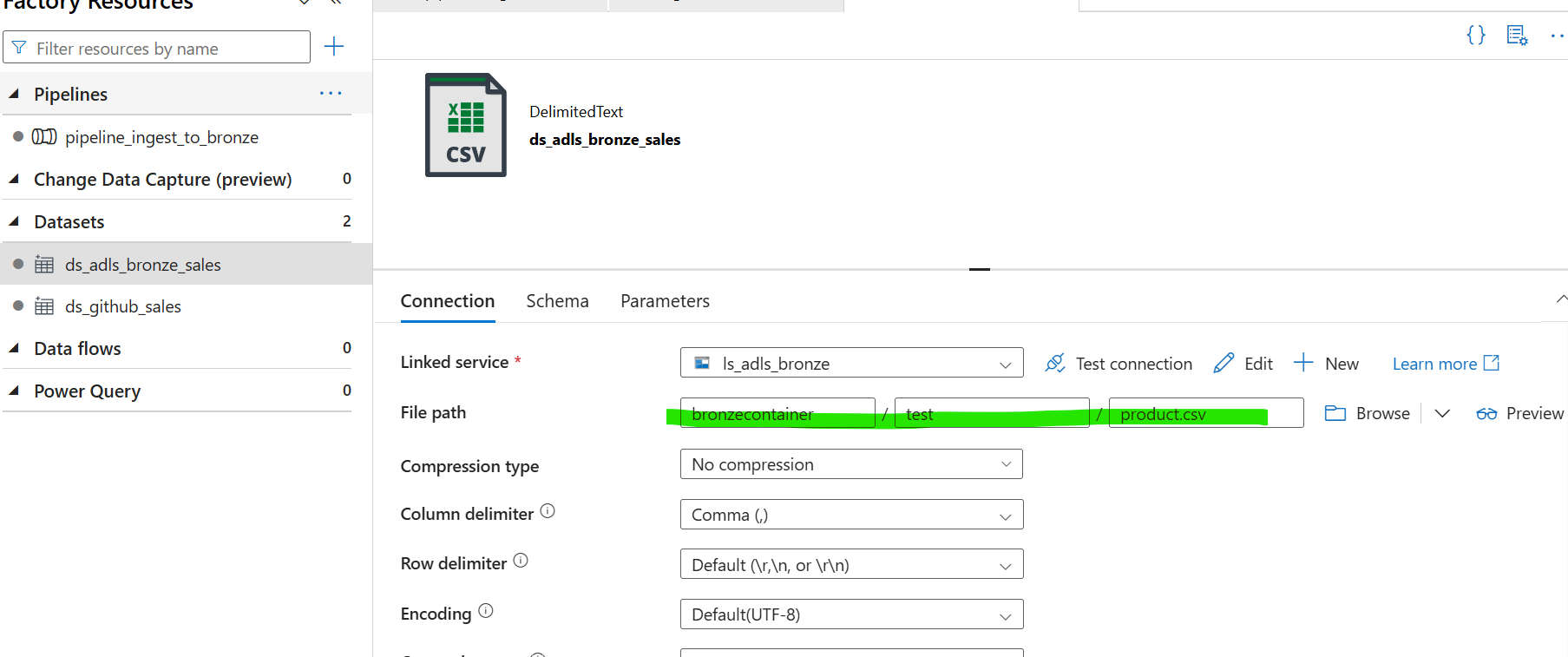
* Go to the **Sink** tab.
* Under **File path in dataset**, you’ll see something like:

container/folder/

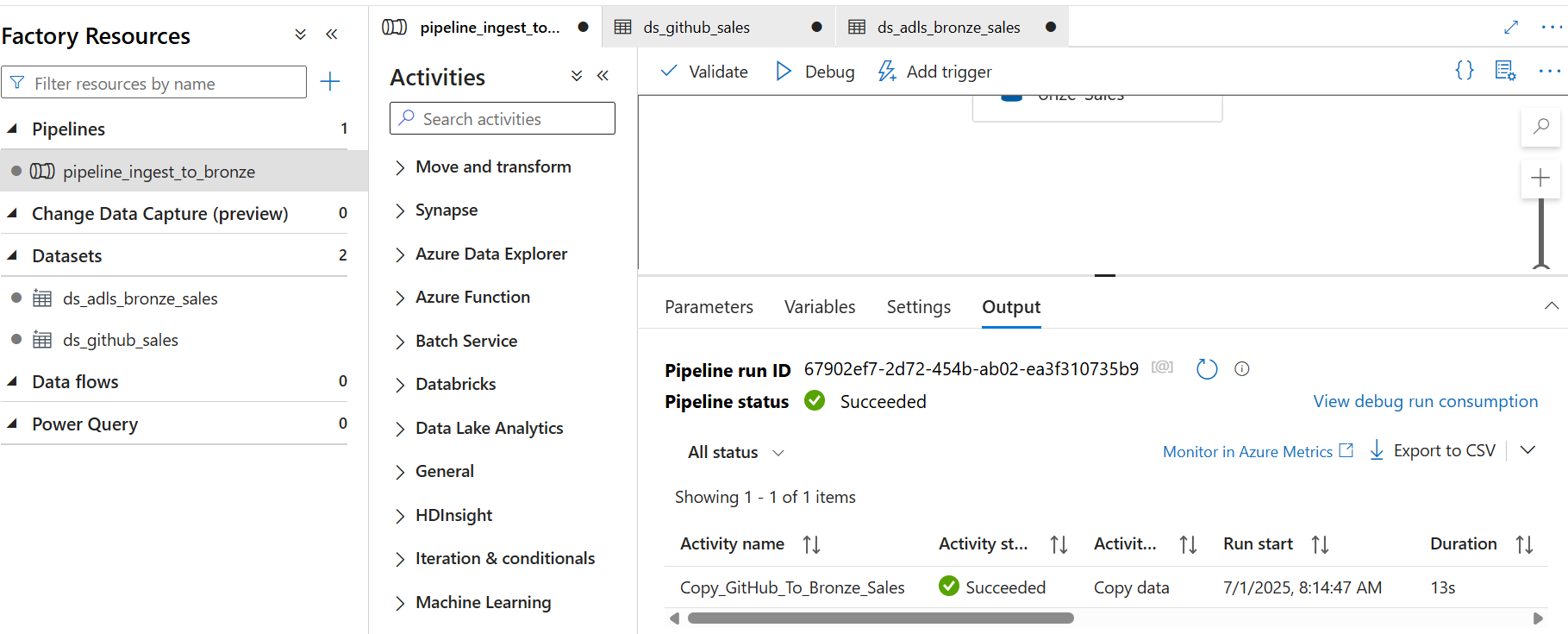
* If you leave it just like that, ADF generates subfolders named with GUIDs or partitions.

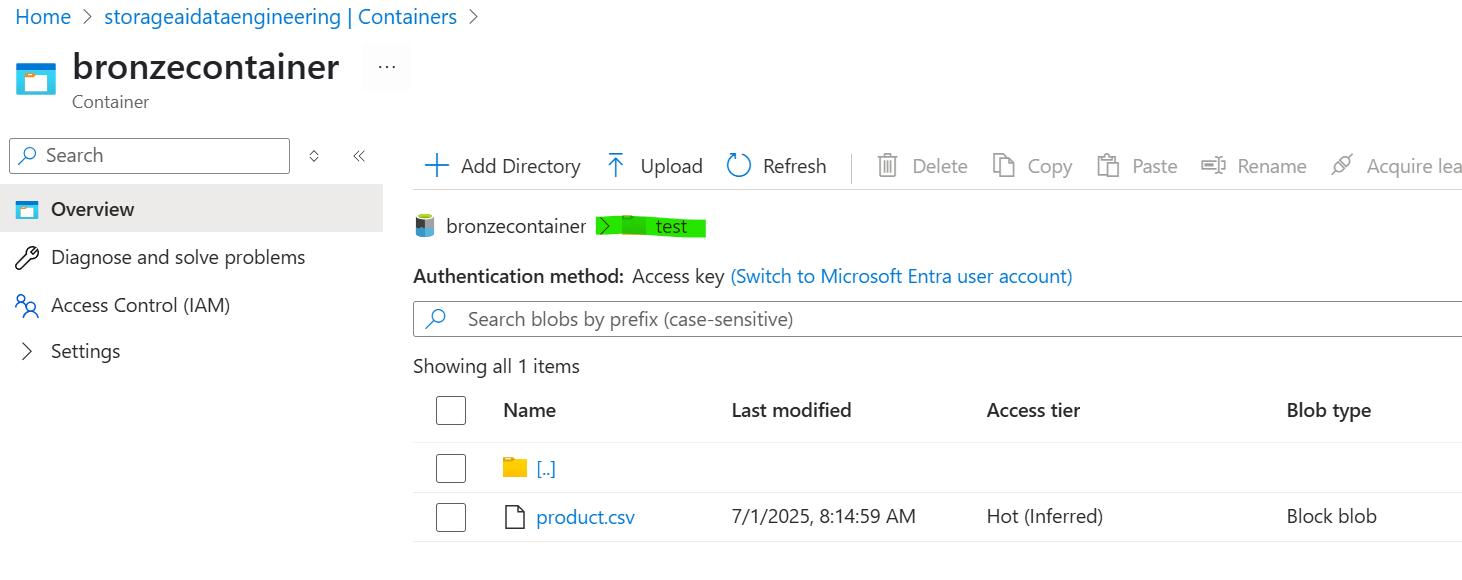
✅ **To write a single file:**

container/bronze/AdventureWorks\_Products.csv



**Fixed this error-**





**🎯** **✅ Fantastic job completing your first static pipeline successfully!  
That is a *huge* step—you now know how to connect, copy, and verify data in Azure Data Factory end-to-end. 🎉**

* **Static pipelines (hardcoded file paths, static names) are *not* scalable for real-world, production scenarios.**

**🎯 Next Step: Move from Static to Dynamic & Automated Pipelines.**

**🟢 What is a *Dynamic / Automated* pipeline?**

**This is what real enterprise projects need:**

**✔ Dynamic Paths & File Names**

* **The pipeline can process *any* file dropped in a folder.**
* **E.g., process all new files uploaded daily.**

**✔ Parameterization**

* **Instead of hardcoding:**

**@{item().name}**

* **So the file name becomes dynamic input.**

**✔ Triggers**

* **Event-based trigger: Starts automatically when a file lands in your storage.**
* **Schedule trigger: Runs every 15 mins/hour/day.**

**✔ Logging & Monitoring**

* **Stores metadata about which files were processed, when, and any errors.**

**🟢 Step-by-Step Guide to Build Your Dynamic Pipeline**

**✅ 1️⃣ Create Your Pipeline**

* **In ADF Studio:**
  + **Author > Pipelines > + New pipeline**
  + **Name it: pl\_dynamic\_ingest\_github\_to\_bronze**

🎯 1️⃣ Linked Service

You are correct:

The linked service stays the same.  
✅  
Because:

* It defines how to connect to GitHub (the HTTP endpoint).
* It does not depend on which file you want to pick.
* Same for the Sink (ADLS Gen2)—the connection doesn’t change.

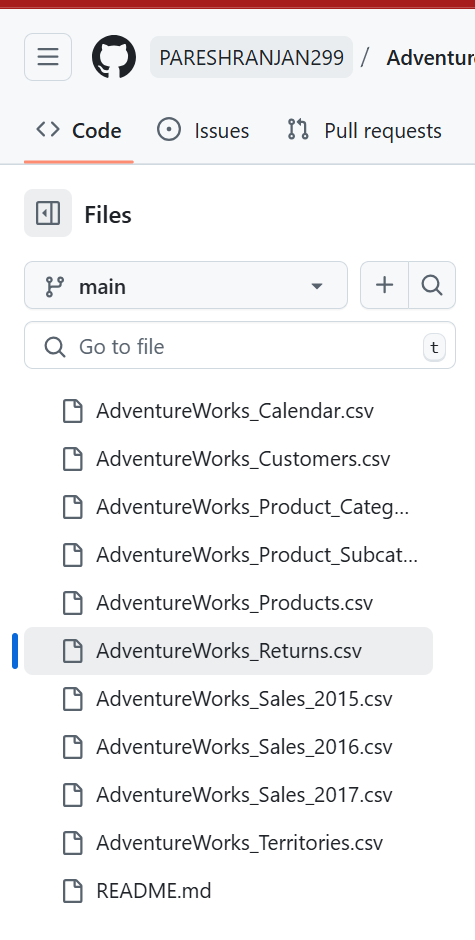
You do NOT need a separate linked service for each file.

**🎯 2️⃣ Parameterized Dataset**

**that I need a parameterized dataset**, but let’s clarify what that means:

* **I only need *one* parameterized dataset** to handle *all the files* in the folder.
* You do **not** need to create separate datasets for each file.
* This dataset:
  + Has a parameter called FileName.
  + Will dynamically receive the file name from the pipeline.

👉 **This is what makes it dynamic.**

****

* **I only need *one* parameterized dataset to handle *all the files* in the folder.**

**🎯 My Statement Recap:** In Data Engineering, ADF **dataset** is the major role—and parameterized dataset with efficiency. Source and sink configuration.

✅

Datasets **are absolutely critical** in ADF because they define:

* **What** data to read or write (file, table, collection).
* **How** to interpret it (schema, delimiter, compression).
* **Where** to locate it (path, file name).

**✅ 1️⃣ Datasets Define the *Structure* and *Location***

* For example, a **DelimitedText dataset** says:
  + “I expect a CSV file.”
  + “Comma-delimited, UTF-8, first row as header.”
  + “Located at this relative URL/path.”

Without a dataset, the Copy Activity doesn’t know how to parse or write the data.

**✅ 4️⃣ Source and Sink Configuration Controls the Movement**

* The **Source**:
  + Which dataset to read
  + Whether to use wildcards
  + Whether to skip headers
* The **Sink**:
  + Which dataset to write to
  + Dynamic partitioning paths (e.g., date folders)
  + File format and naming

**This is why “source and sink configuration” is where most data flow complexity lives.**

**✅ 5️⃣ Mapping Bridges Different Schemas**

* The Mapping tab in Copy Activity **maps source columns to sink columns.**
* Auto Mapping vs. Manual Mapping:
  + Auto Mapping: Quick, but can break if column names differ.
  + Manual Mapping: Safer for production.

**✅ 6️⃣ Additional Concepts**

Here are **other advanced elements** you’ll often combine with parameterized datasets:

🔹 **Wildcards**:

* E.g., read all .csv files with \* in the name.

🔹 **ForEach Activity**:

* Loop over the list of files dynamically.

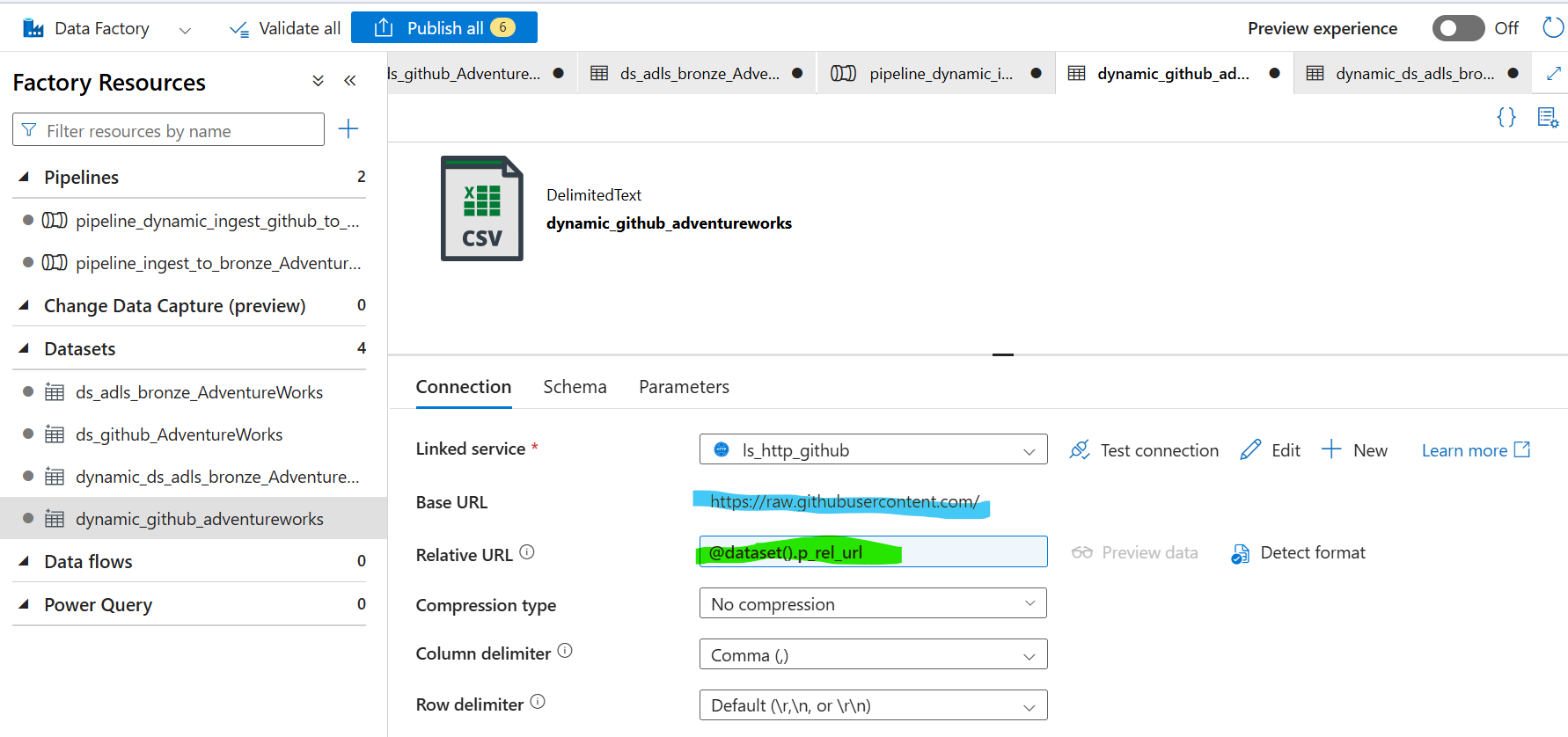
🔹 **Triggers**:

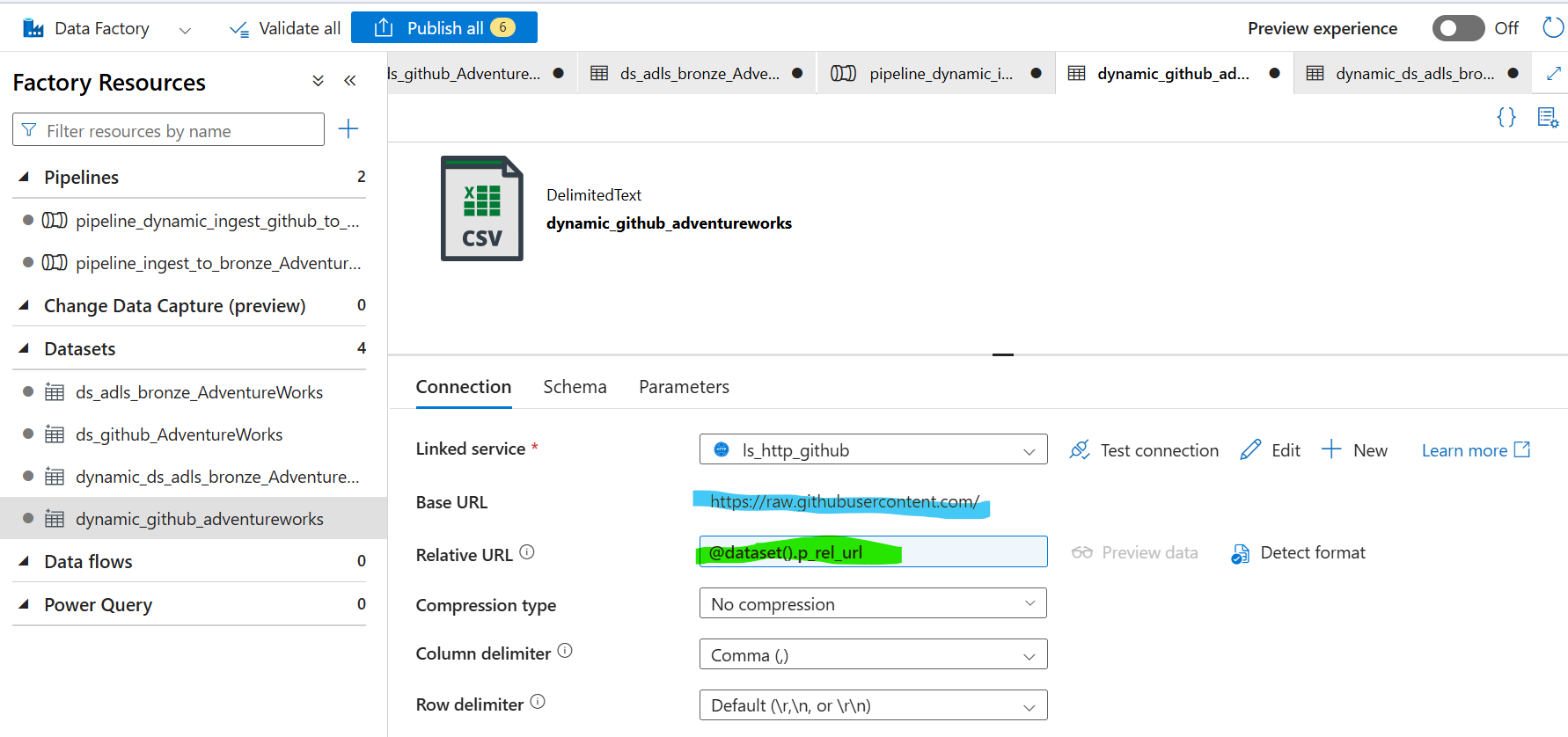
* Schedule or event-based execution.

🔹 **Metadata-driven pipelines**:

* Entire pipeline logic is driven by metadata tables (e.g., a config table with file paths, target tables).

**First, we should create a parameterized Source Dataset so the pipeline can dynamically pick different files or paths. Then we configure the Sink Dataset to either use static or dynamic paths based on our requirement.**

****

****

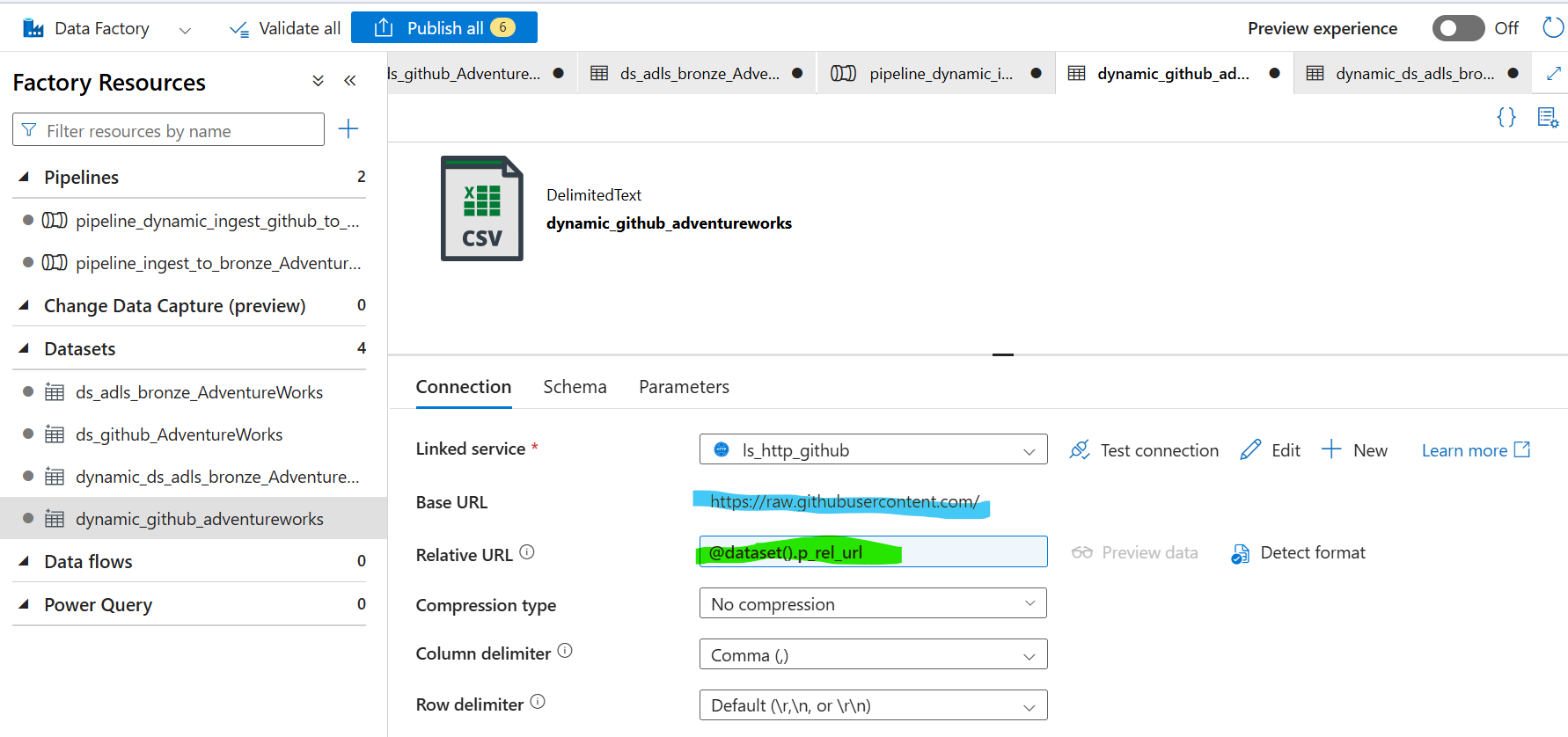
**The fully understand how Linked Service, Base URL, and Relative URL work**

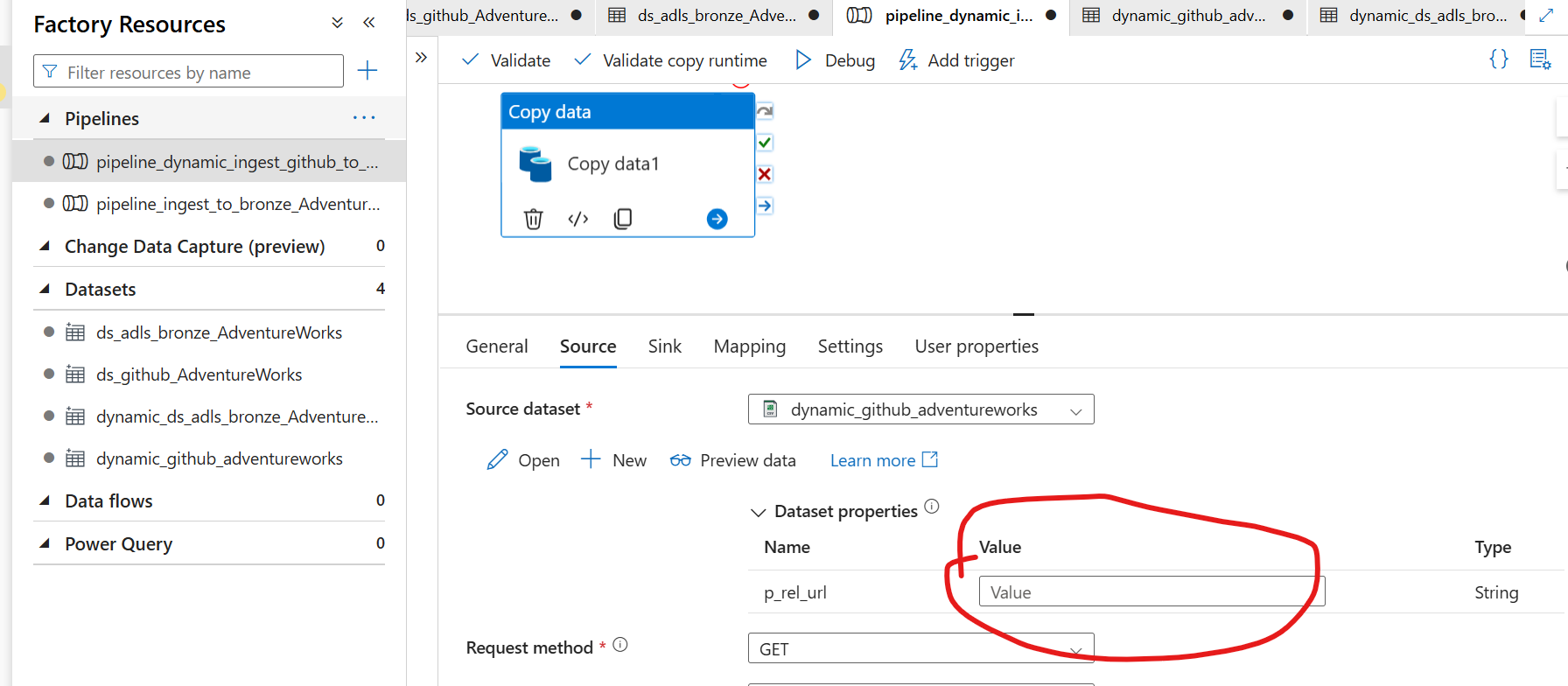
**🎯** Think of It Like This:

🔹 Linked Service = “Base URL”

* This is the *common* part of your connection.
* Example for GitHub Raw: [**https://raw.githubusercontent.com/**](https://raw.githubusercontent.com/)
* All files in your repo will come after this base URL.
* This part is the same for **every dataset in this repo**.

🔹 **Dataset = “Relative URL”**

* This is the part **after** the base URL.
* This is where you define which folder or file you want.
* For example, you might have:
* **✅ For *Products* file: PARESHRANJAN299/AdventureWorks-Data-Engineering-Project/refs/heads/main/AdventureWorks\_Products.csv**
* **✅ For *Calendar* file: PARESHRANJAN299/AdventureWorks-Data-Engineering-Project/refs/heads/main/AdventureWorks\_Calendar.csv**
* **Let’s Parameterized the source dataset within the Copy activity: lets created**
* **You created a parameterized dataset, but you haven’t assigned the value of the parameter yet.  
  You will pass that value dynamically at runtime—using ForEach or a loop**
* **Why does it show “Value not provided”?  
  Because:**
* **The dataset itself doesn’t have any hardcoded value.**
* **It’s waiting for the *pipeline* to supply the value when the pipeline runs.**



**How do you supply the parameter?**

You have **two** main ways:

✅ **1️⃣ ForEach Activity**  
**Most common in dynamic ingestion:**

* You first use **Get Metadata** to list all files in the folder.
* That returns:
* [
* {"name": "AdventureWorks\_Products.csv"},
* {"name": "AdventureWorks\_Calendar.csv"},
* {"name": "AdventureWorks\_Customers.csv"}
* ]

✅ **2️⃣ Manually Assign in the Copy Activity (for testing)**  
If you just want to debug a single file before looping:

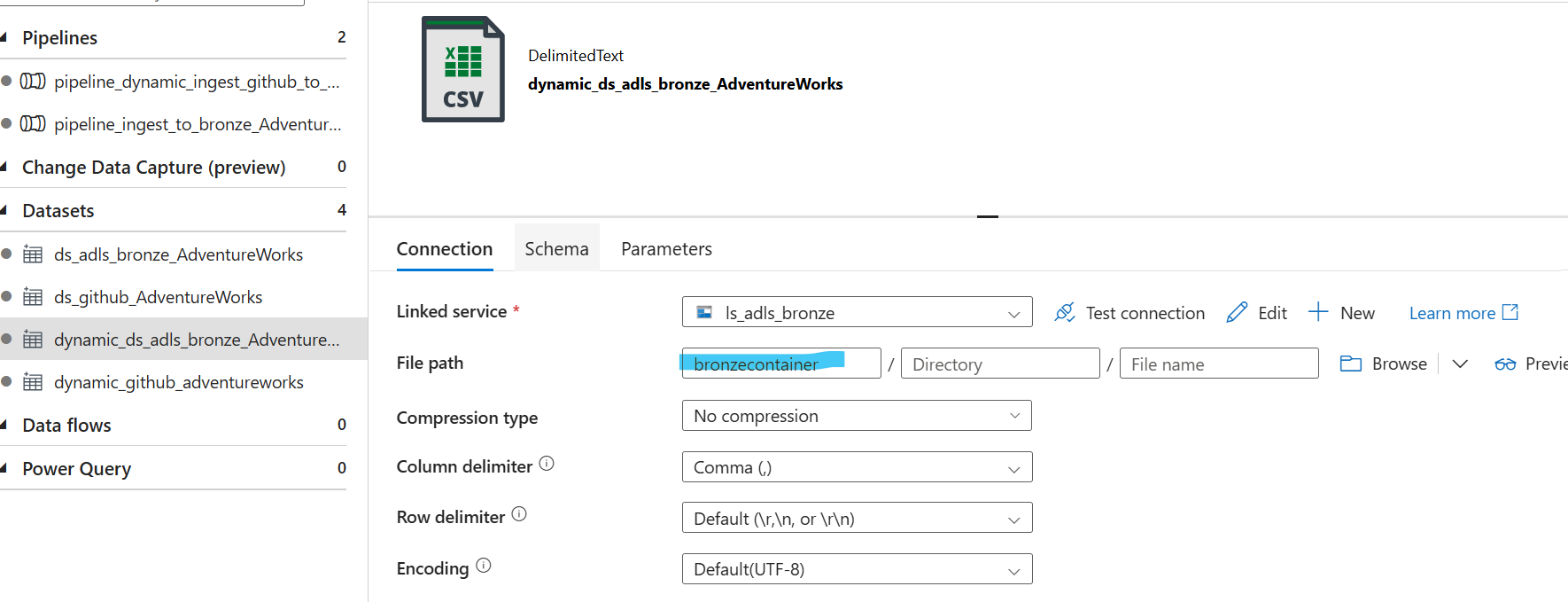
* Open the Copy Data activity.
* Click **Source**.
* You will see the parameter **RelativeUrl** field.
* Manually enter:

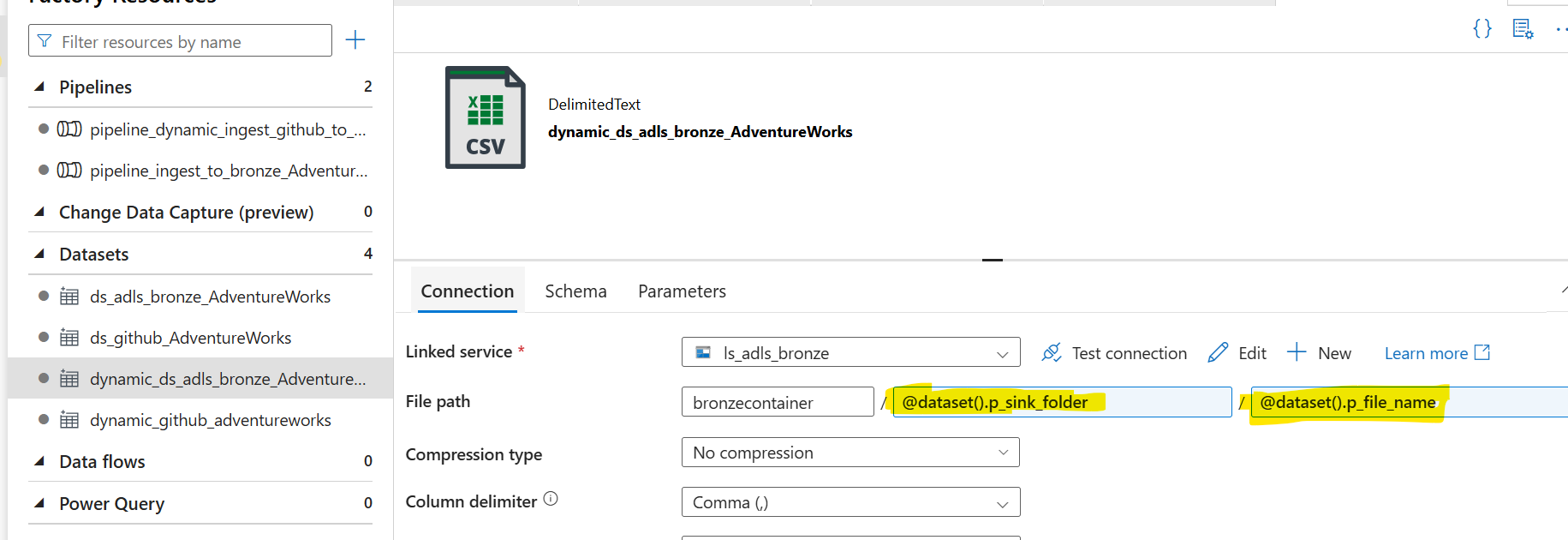
**🎯 Why Do We Parameterize the Sink?**

**You only parameterize the Sink if you want your output *destination* to be dynamic.**

**✅ Dynamic Sink scenarios:**

* **You want each file to go to a different folder.**
  + **Example:** **/bronze/sales/2024/06/AdventureWorks\_Products.csv**
  + **/bronze/sales/2024/06/AdventureWorks\_Customers.csv**

****



**🟢 🎯 YES: You Can Parameterize Both Folder Name AND File Name**

This is **exactly** how enterprises build *flexible* pipelines.

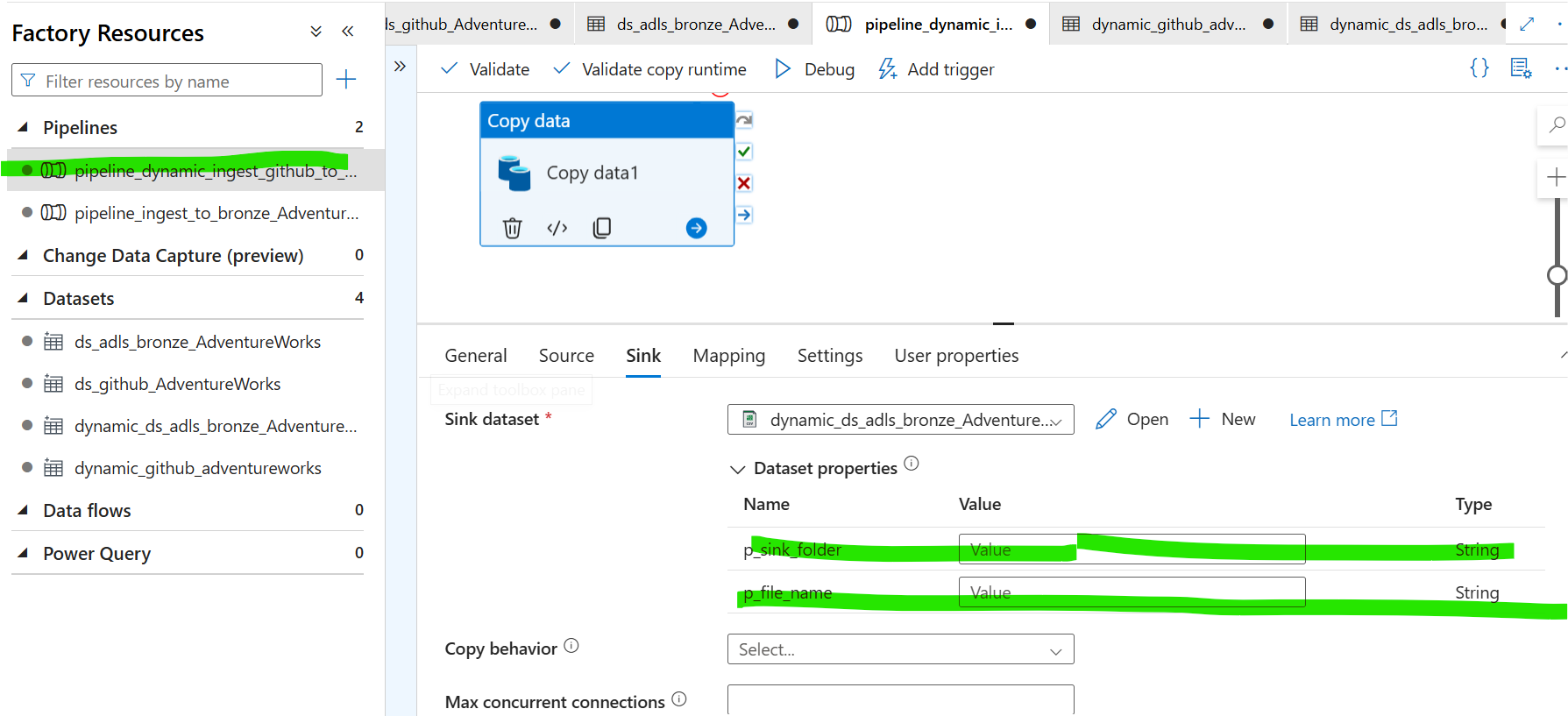
✅ In your **Sink**, you can parameterize:

1️⃣ **The Folder Path**  
2️⃣ **The File Name**

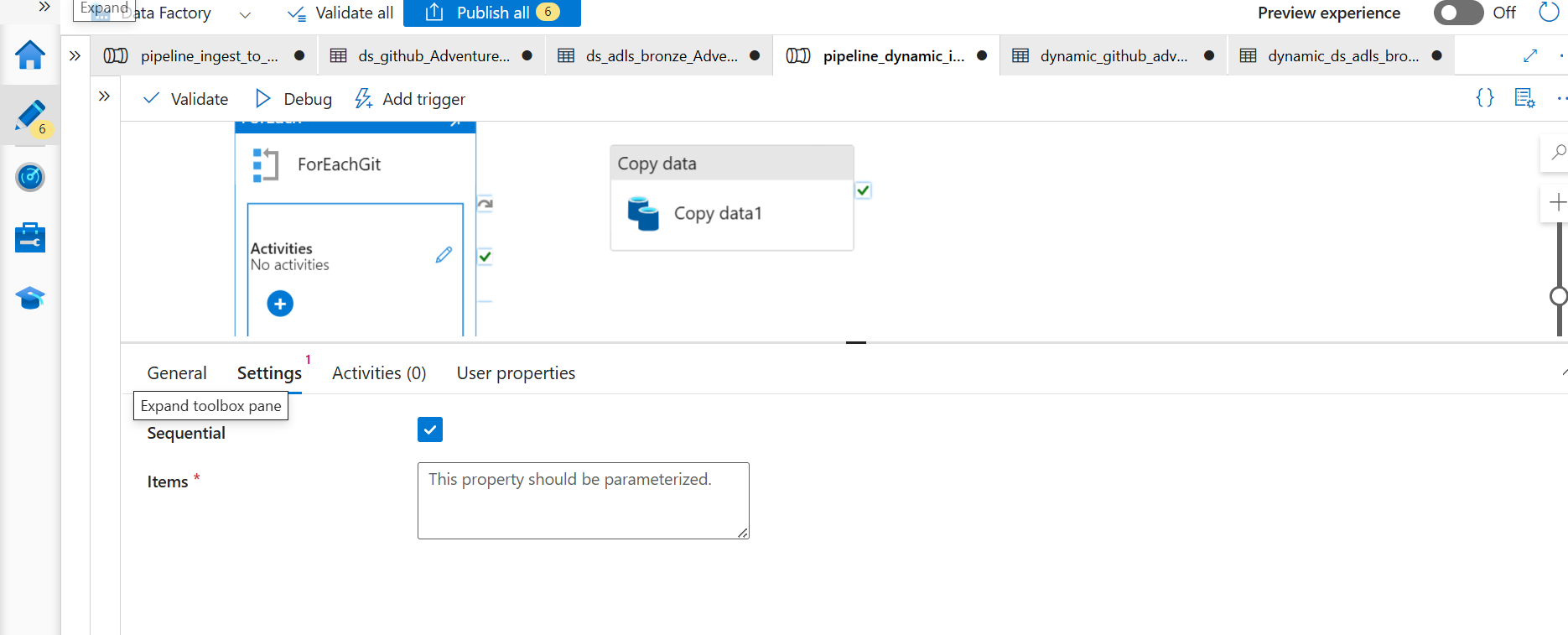
**🟢 Why Would You Do This?**

🎯 **Because you want:**

* Each file to land in its own folder (e.g., by date, by type).
* The file name to be dynamic (e.g., same name as the source)



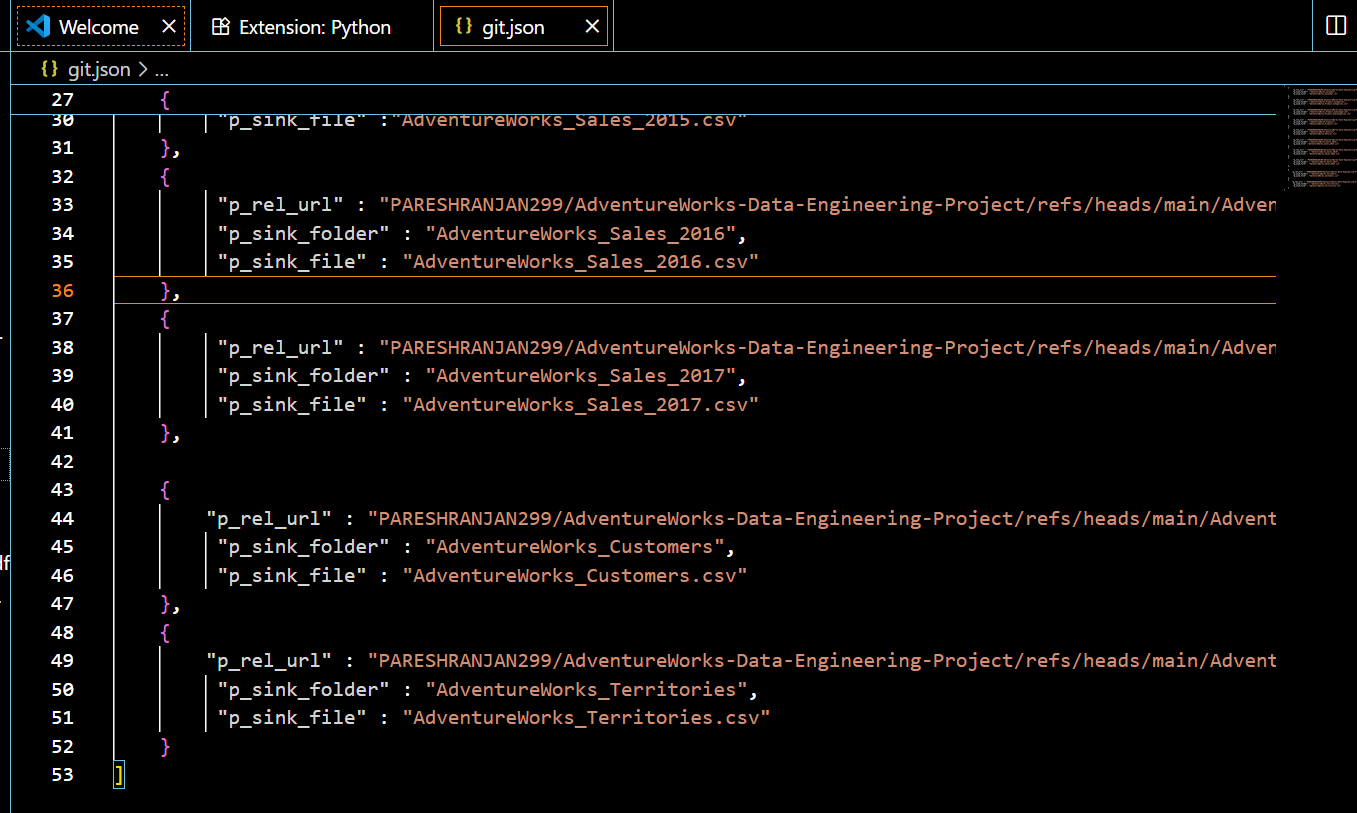
* **We’ve now parameterized both the source and the sink datasets. The only step pending is passing the parameter values dynamically (Source & Sink) at runtime through the pipeline activities.**

****

🎯 What You Did

You created an array of JSON objects, each containing:

* p\_rel\_url: the relative path to the GitHub file ✅
* p\_sink\_folder: the folder name to write to in your sink ✅
* p\_sink\_file: the filename to use in your sink ✅

****

🟢 ✅ Quick Validation Checklist

I checked carefully:

1️⃣ Well-formed JSON array  
✅ Yes.  
2️⃣ Each object has all 3 keys  
✅ Yes.  
3️⃣ No commas at the end of the last object  
✅ Yes.  
4️⃣ All string values properly quoted  
✅ Yes.

✅ So this JSON is perfectly valid.

🟢 🎯 How ForEach Works in ADF (Simple Flow)

1️⃣ Takes a list of items (array)  
2️⃣ Loops over each item, one by one  
3️⃣ Runs whatever activities you put inside it

* e.g., Copy Data
* e.g., Stored Procedure  
  4️⃣ Passes the current item’s values into those activities

🎯 **What you did (Step by Step):**

✅ **1️⃣ You created a storage container named parameters.**

* This is just a folder in your Azure Storage account.
* Inside this container, you stored your JSON file: **git.json**
* containing your list of files to ingest. (most important to understand)

✅ **2️⃣ You created a Lookup activity in your pipeline.**

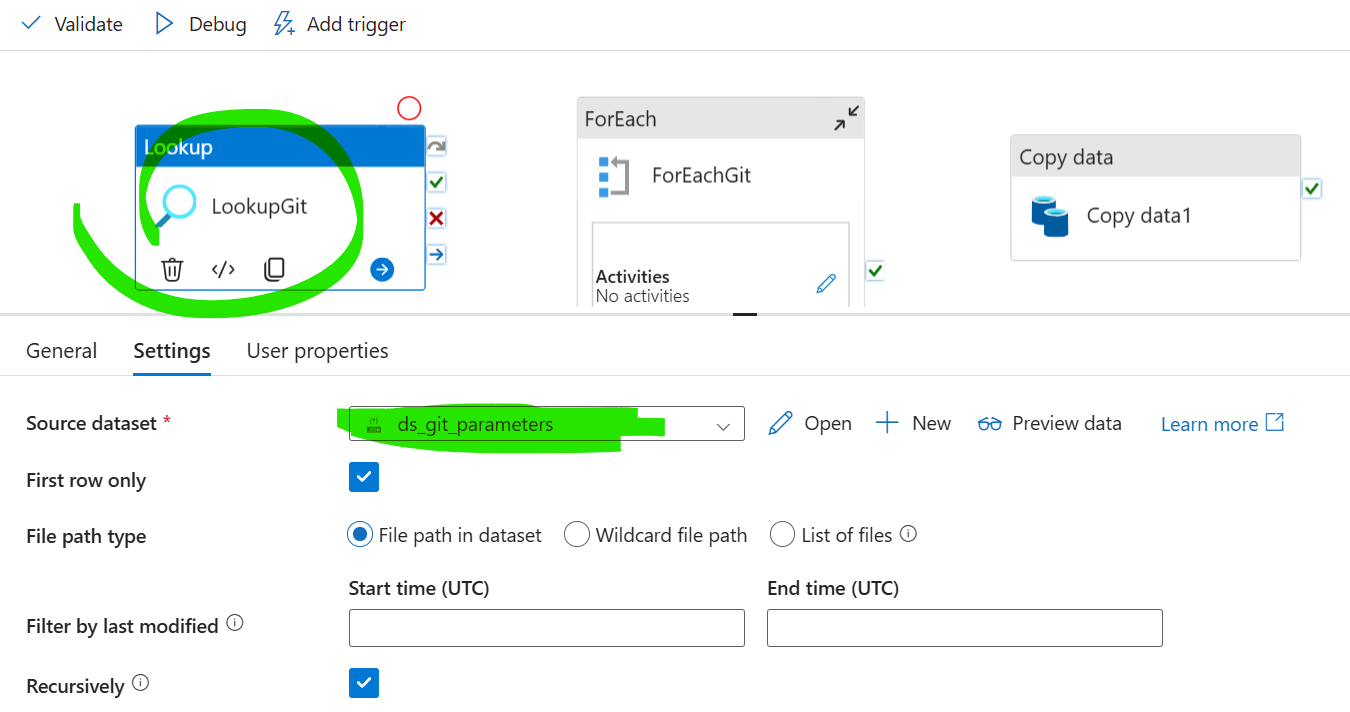
* **Purpose of Lookup:** read the contents of that JSON file.
* This is the *standard pattern* to bring metadata into ADF.

✅ **3️⃣ Inside the Lookup activity, you configured a dataset:**

* Type: **JSON dataset**
* Linked Service: your Storage account
* File Path: something like: parameters/git.json
* File format: JSON (array)

✅ **4️⃣ The Lookup activity outputs your JSON array as structured data.**

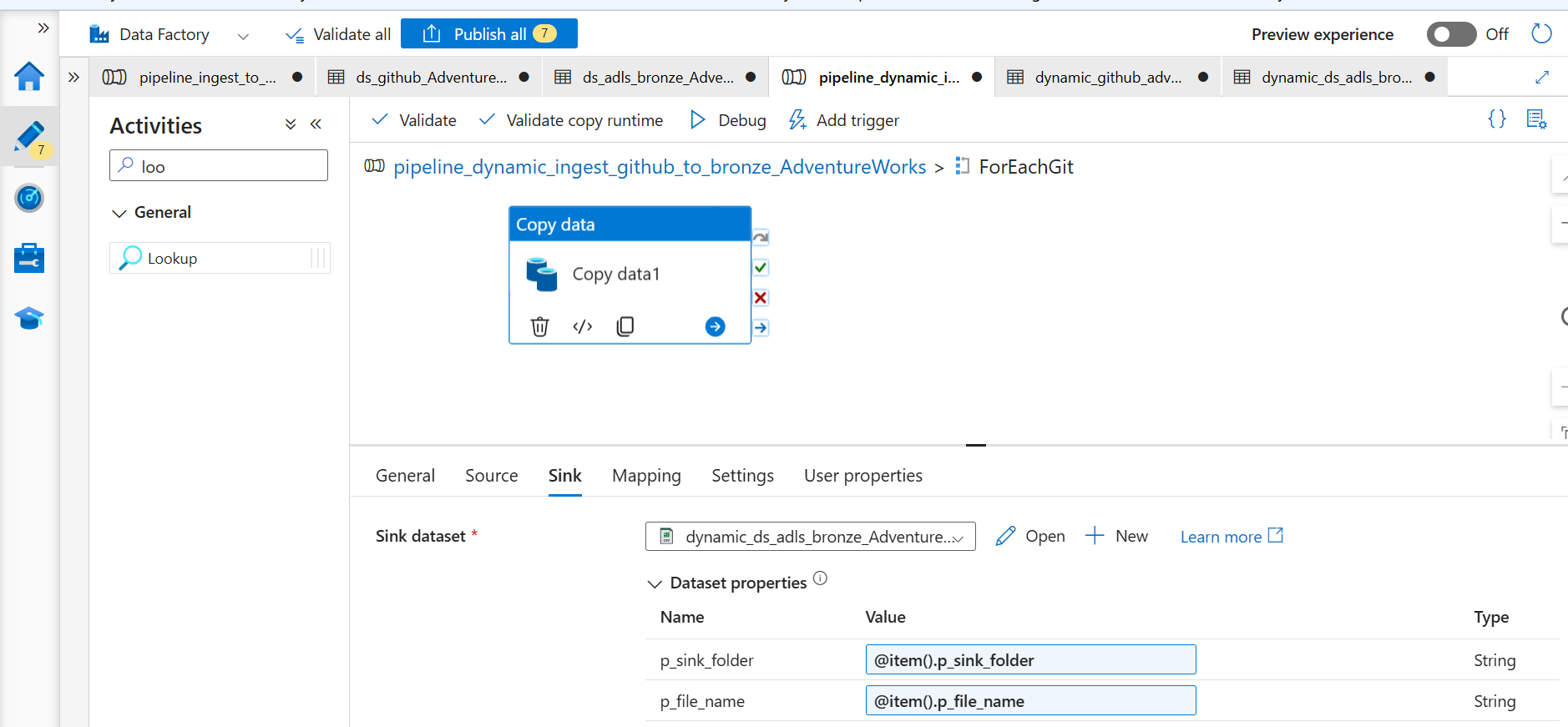
* You then pass this output to the **ForEach** activity to loop over each object.



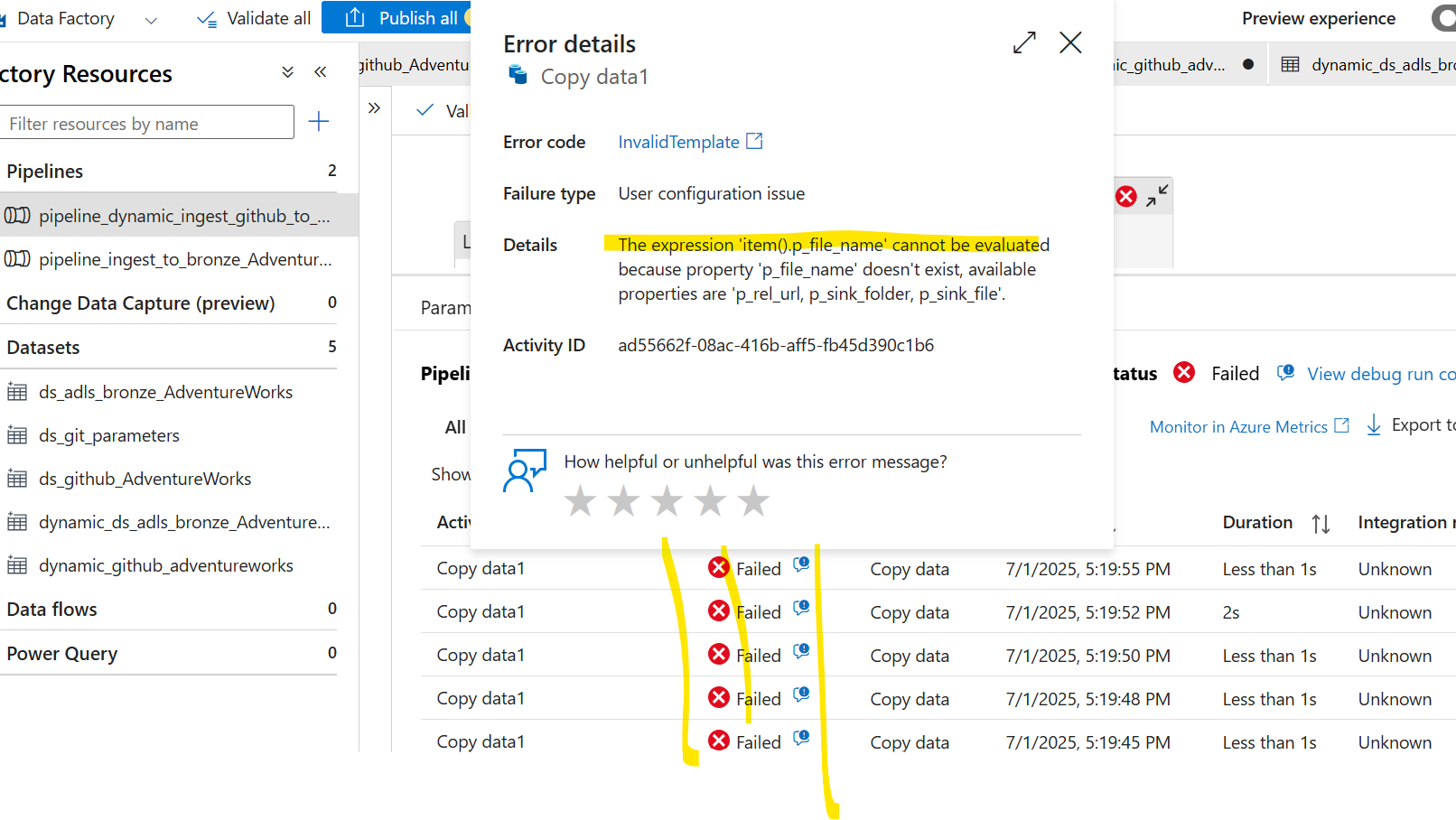
Oncheck this tick box 🡪 ‘’First Row Only’’ because it will run only one time with Lookup activity, we do not want that, we required to run all the activity.

We Bring this copy activity within the ForEach activity – and added the ‘’value’’ for **1.** Source and **2.** Sink,

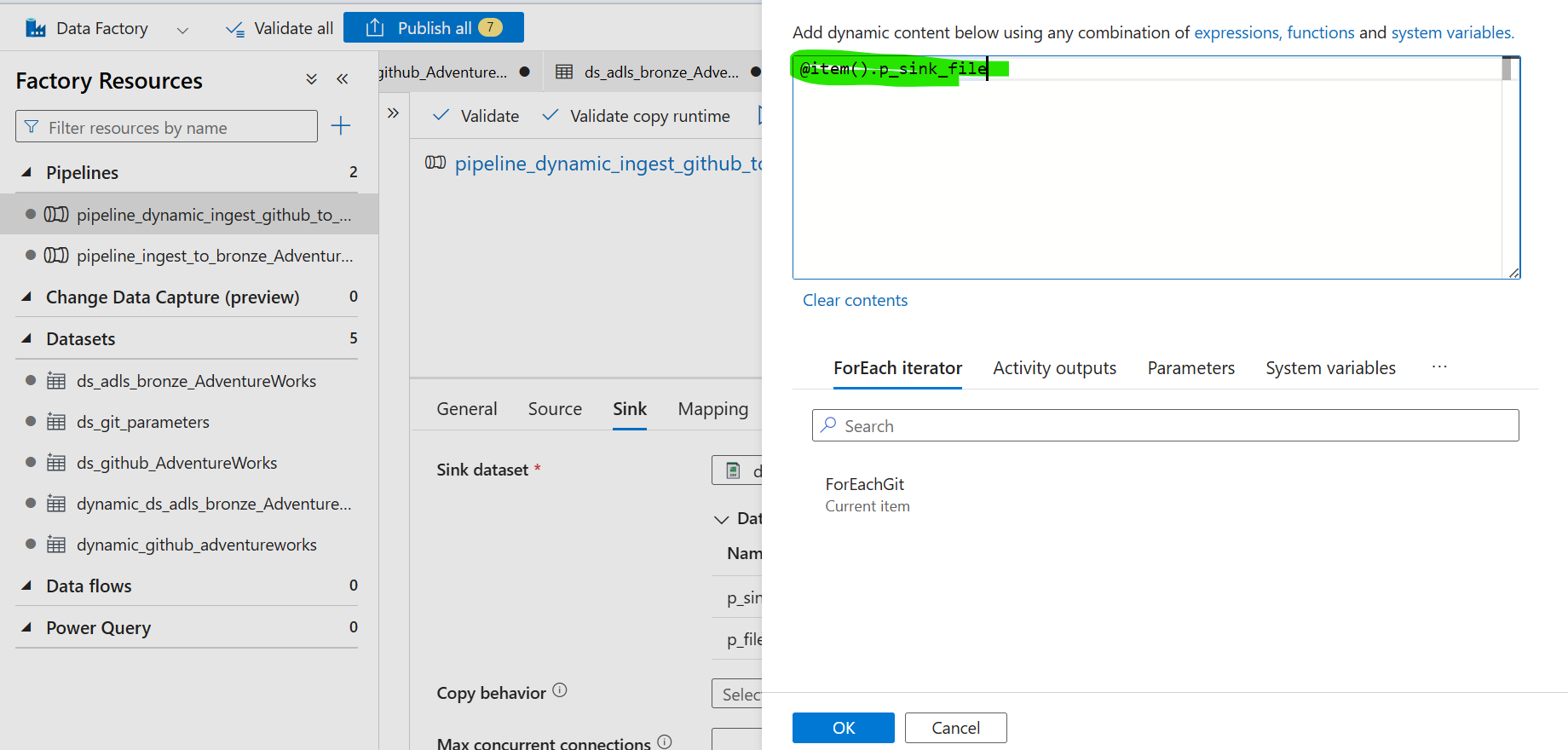
Where you get this value: - I get this from my Json file. Where I set a array from my files.

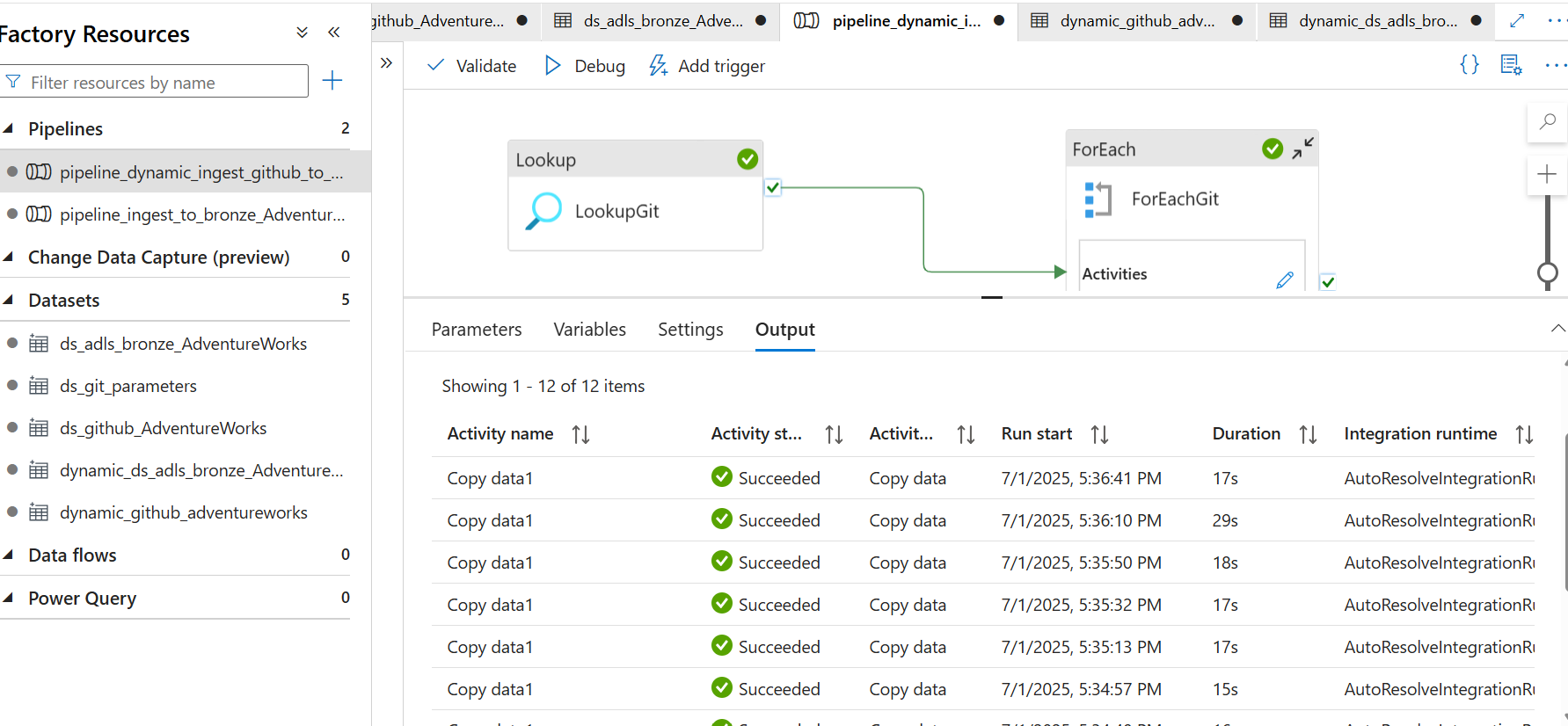


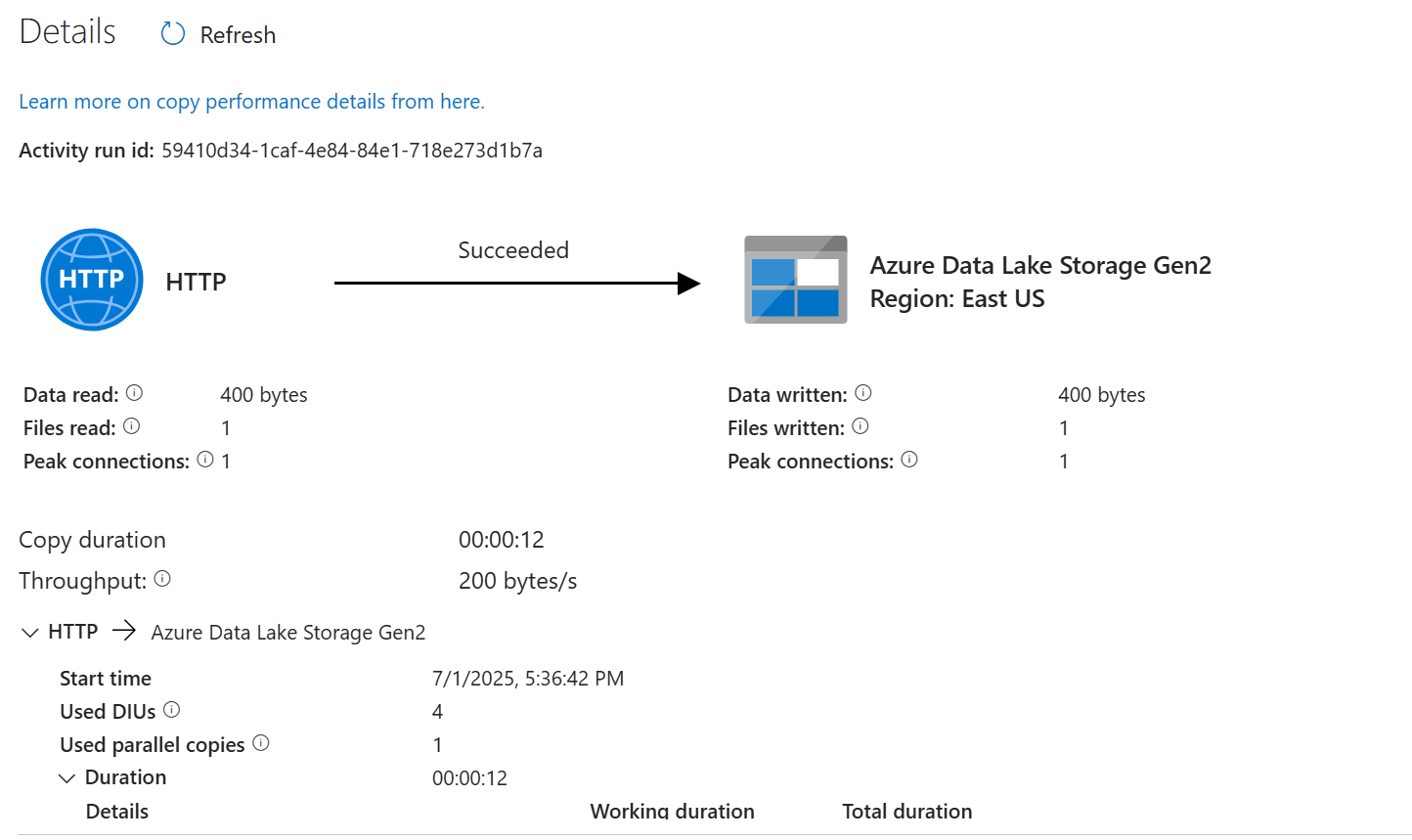
**The Expression ‘item().p\_file\_name’ cannot be:**

****

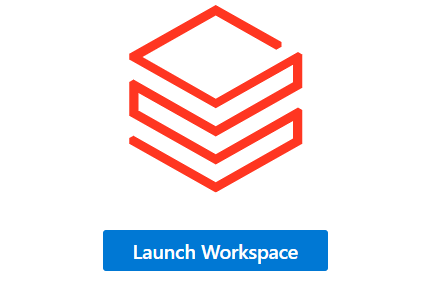
**Fixed the issue: why this occur because the sink-file name can’t not same for the value. So I changed it to ‘’@item().p\_sink\_file’’**

****

****

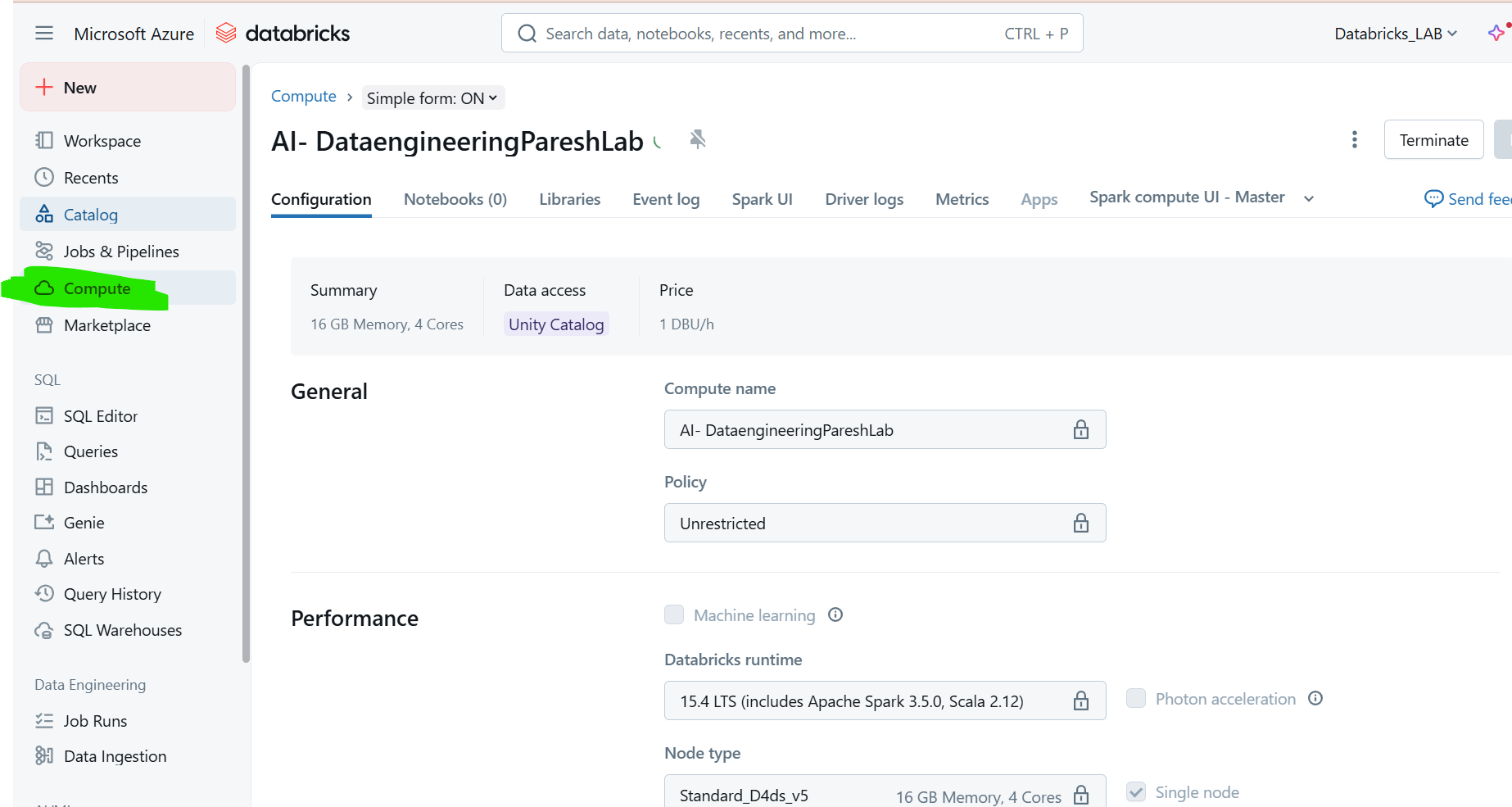
****

**Finally, I have completed my phase -1 in the Project.**

****

**In Phase 2 of the project lifecycle, we will implement data transformation pipelines to cleanse, enrich, and reshape the ingested raw data into curated datasets suitable for analysis and consumption.**

**Create an Compute on Databricks.**

****