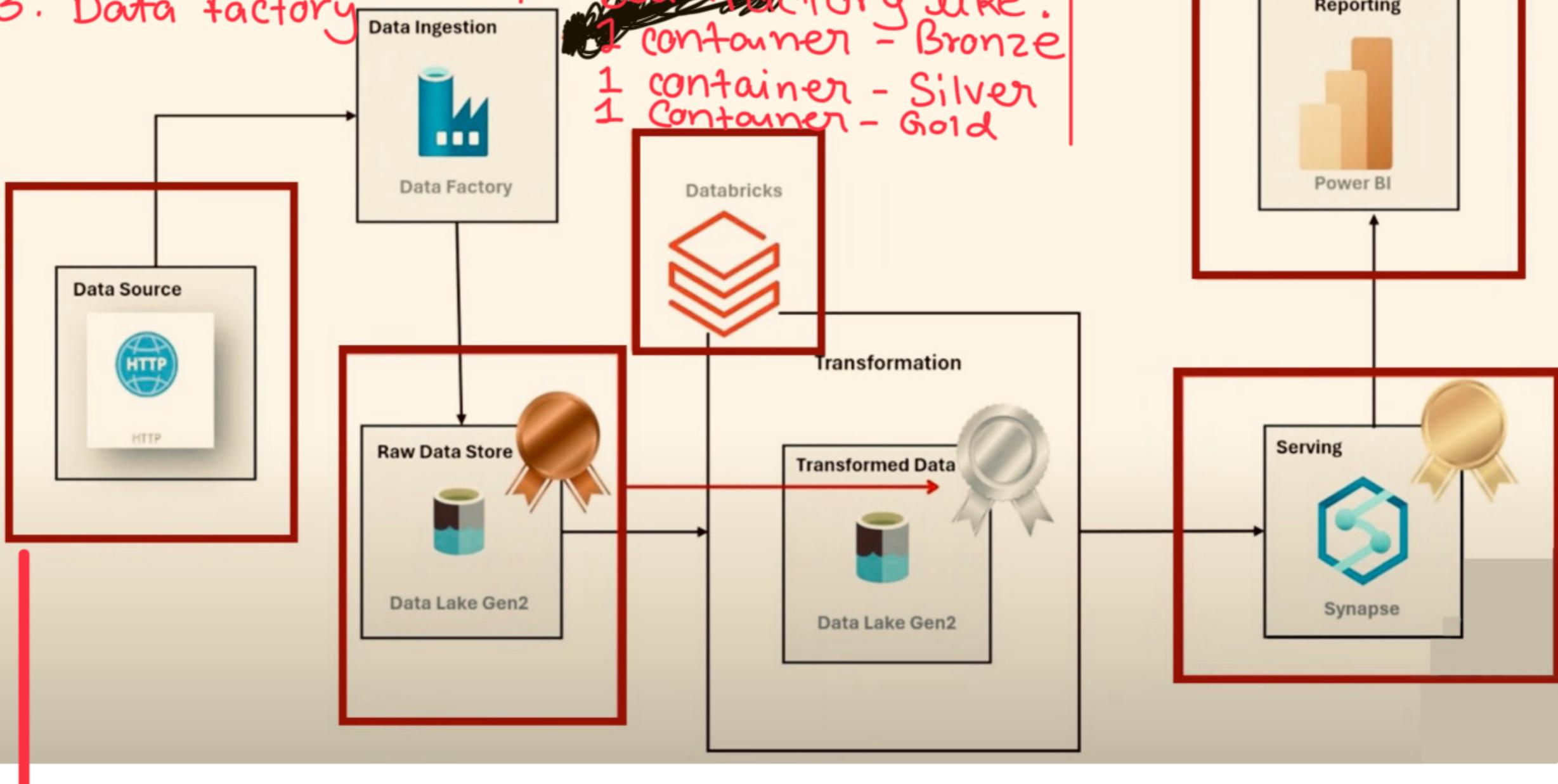


Azure End to End Data Engineering Project

Data Architecture of the Project

1. Resource groups
2. Storage account
3. Data factory



4 We will be pulling data directly out of APIS.

3 layers of data

★ Bronze layer —

Keep the data as it is available in source. No transformation will be applied on it.

★ Silver layer —

Then we will clean and transform the data present in Bronze layer and push it to Silver layer using spark.

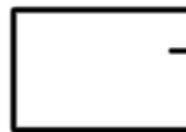
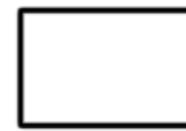
★ Gold layer —

We will build data warehouse and the most popular one right now is Azure Synapse Analytics

Difference between blob storage and datalake

blob

Datalake



and then
file

we can not create
folder in blob

we can create
folders in Datalake

▼ Data storage

Containers

File shares

Queues

Also Known as Data lake

To share all the files throughout the
organization

can be ,json data

Now, we will create 3 containers

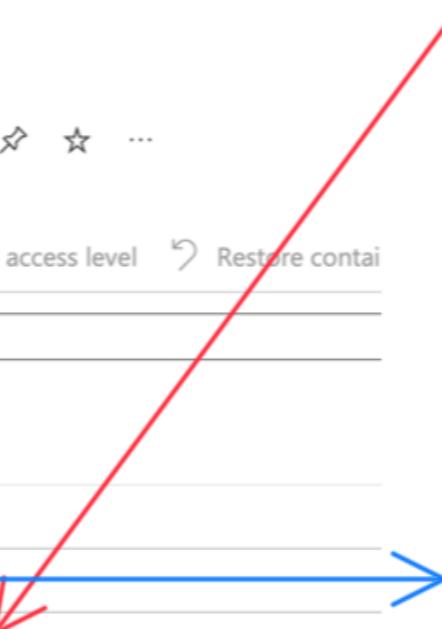
Home > awsstroagedatalakekr

awsstroagedatalakekr | Containers

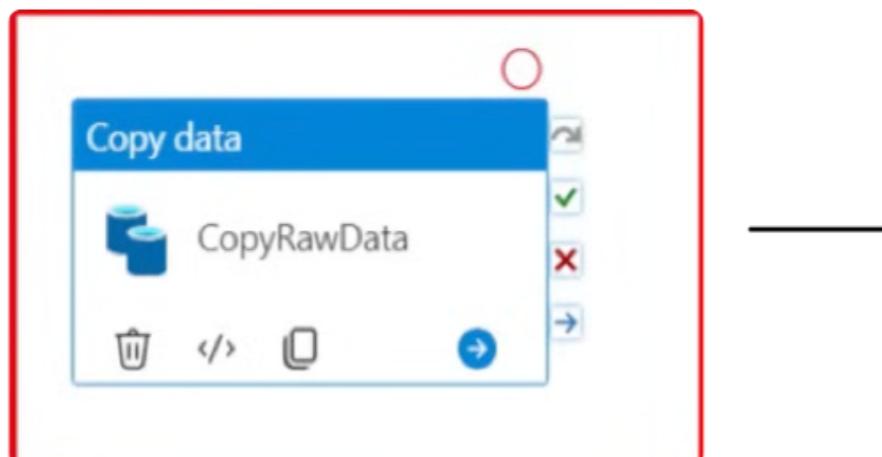
Storage account

- Search
 - Container
 - Change access level
 - Restore container
- Data migration
- Events
- Storage browser
- Partner solutions
- Resource visualizer
- Data storage
 - Containers
 - File shares
 - Queues

Name
\$logs
bronze
gold
silver



After creating 3 containers. Now, we will perform data loading in bronze layer



This will just load the data from the source and push the data to the destination.

General Source¹ Sink¹ Mapping Settings User properties

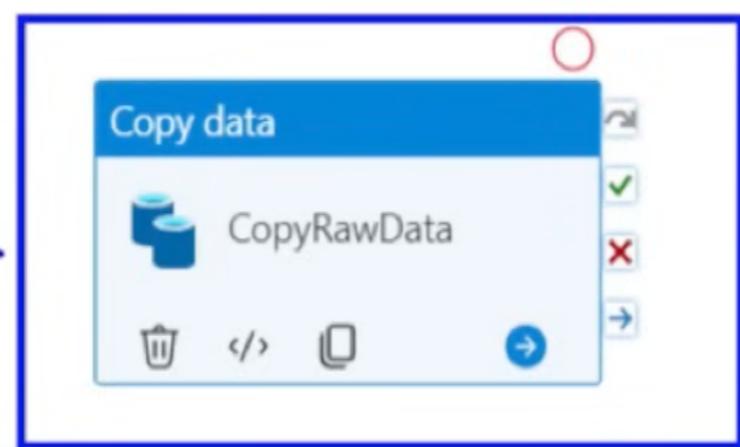
means destination

Now , we want to create a linked service for connection .

Read the data from git hub

HTTP
Source

(Build connection)



and push that data to data lake .

data lake
Destination

Linked services

Linked service defines the connection information to a data store or compute. [Learn more](#)

+ New

Filter by name

Annotations : Any

Showing 1 - 2 of 2 items

Name ↑	Type ↑	Related ↑	Annotations ↑
httplinkedservice	HTTP	0	
storagedatalake	Azure Data Lake Storage Gen2	0	

linked service has been created

Now , we will create dataset

Source Sink

Source dataset * ds_http Open New Preview data

Request method * GET

Additional headers

Request body

Request timeout

Max concurrent connections

Skip line count

Additional columns

+ New

Source Sink

Sink dataset * ds_raw Open New

Copy behavior Select...

Max concurrent connections

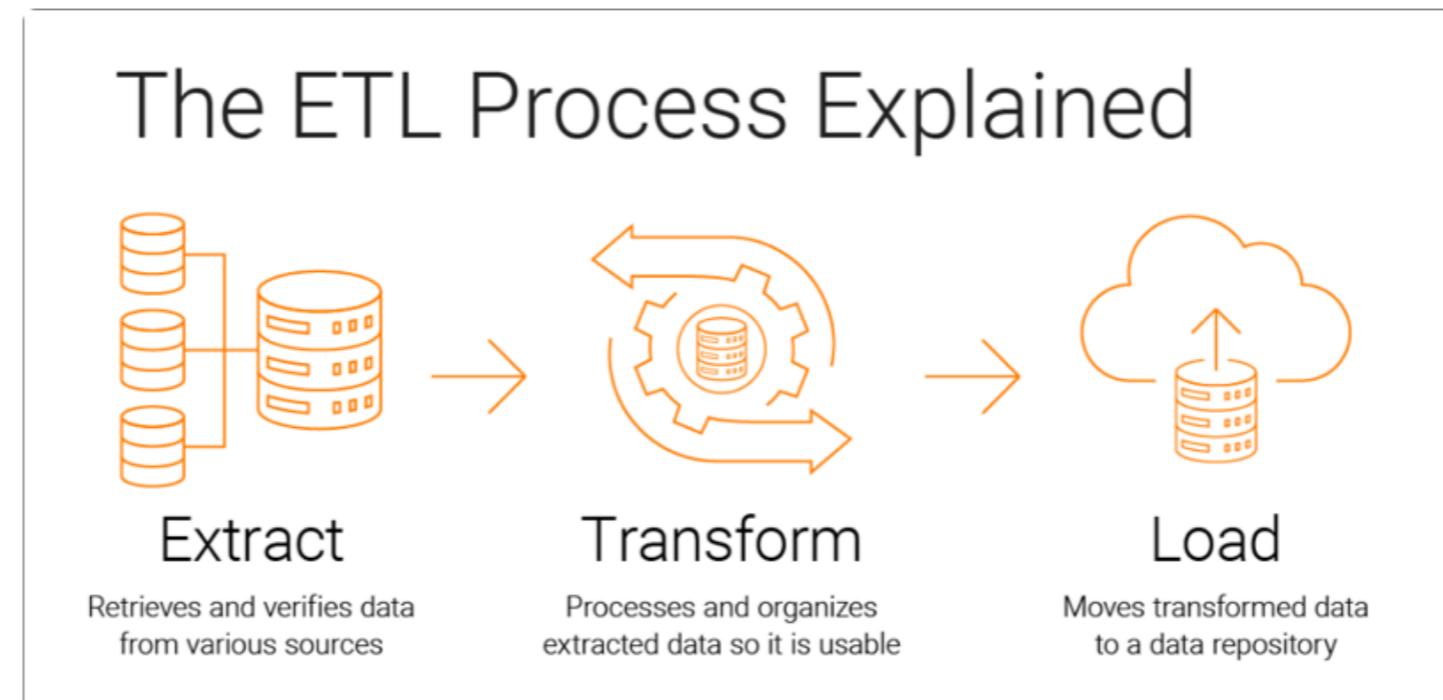
Block size (MB)

Metadata + New

Quote all text

File extension .txt

Max rows per file



Static Pipeline & dynamic Pipeline

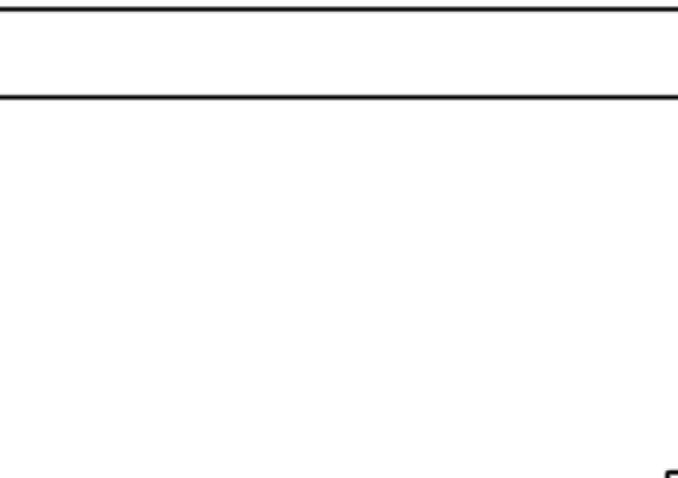
Static Pipeline :

A static pipeline has a fixed structure - its sequence of steps is predefined and does not change at runtime.

Dynamic Pipeline :

A dynamic Pipeline can change its structure during execution, often based on conditions or the characteristics of the input data.

- AdventureWorks_Calendar.csv
- AdventureWorks_Customers.csv
- AdventureWorks_Product_Categories.csv
- AdventureWorks_Product_Subcategories.csv
- AdventureWorks_Products.csv ~~███████████~~
- AdventureWorks_Returns.csv
- AdventureWorks_Sales_2015.csv
- AdventureWorks_Sales_2016.csv



COPY ACTIVITY

- [relative Url]
- (folder)
- (file)

Just these 3 values will be changing for this

Copy activity.

AdventureWorks_Sales_2017.csv

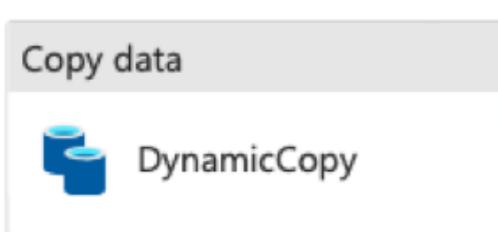
AdventureWorks_Territories.csv

(Total 10 iterations)

we want to pull this data. one way of doing is repeating the copy activity again and again. But that will be static approach which we do not want to do that.

So, we will use for loop.

So, instead of changing those 3 values everytime. we will create 3 parameters, and keep on changing the values of these 3 parameters. So everytime each iteration will have different set of parameters. and for that we will use For Each Activity.



The screenshot shows the 'Source' configuration for a CSV dataset named 'ds_git_dynamic'. The 'Connection' tab is selected, showing a linked service 'httplinkedservice' and a base URL 'https://raw.githubusercontent.com'. Other settings include relative URL '@dataset().p_rel_url', compression type 'No compression', column delimiter 'Comma (,),' row delimiter 'Default (\r,\n, or \r\n)', encoding 'Default(UTF-8)', quote character 'Double quote ("),' escape character 'Backslash (\),' and first row as header checked. A red box highlights the word 'Source'.

The screenshot shows the configuration of a Sink dataset in Azure Data Factory. At the top, there's a preview of a CSV file icon labeled "DelimitedText ds_sink_dynamic". Below it, the "Sink" tab is selected in the navigation bar. The configuration includes:

- Connection:** storagedatalake
- File path:** bronze / @dataset().p_sink_folder / @dataset().p_file_name
- Compression type:** No compression
- Column delimiter:** Comma (,)
- Row delimiter:** Default (\r,\n, or \r\n)
- Encoding:** Default(UTF-8)
- Quote character:** Double quote ("")
- Escape character:** Backslash (\)
- First row as header:** checked
- Null value:** (empty field)

Below this, the "Source" tab is selected, showing the "Source dataset" as "ds_git_dynamic". The "Sink" tab is also selected, showing the "Sink dataset" as "ds_sink_dynamic". Both sections show "Dataset properties" with a table:

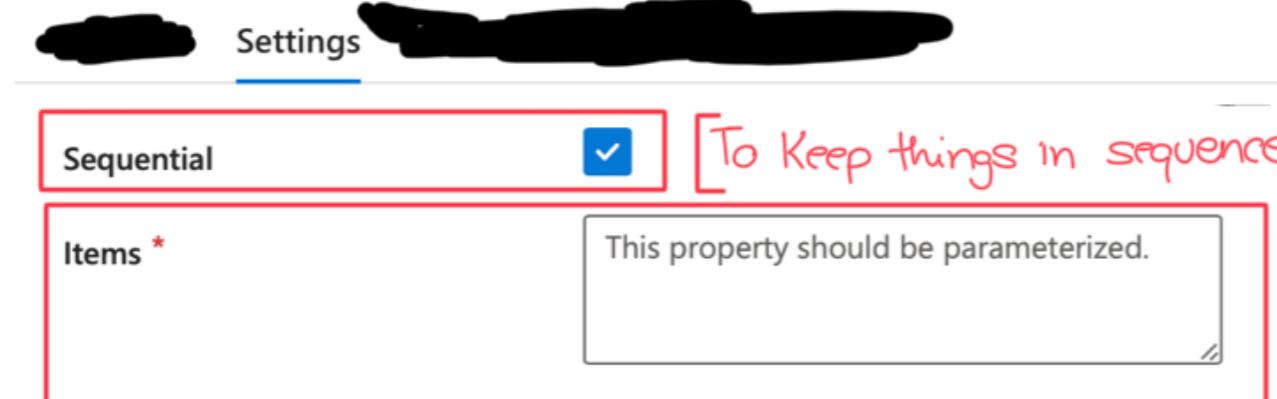
Name	Type
p_rel_url	String
p_sink_folder	String
p_file_name	String

A large red bracket on the right side of the screen groups the three rows under "p_" and points to handwritten text that reads: "we need to provide values for 3 parameters".

ForEach block diagram:

- ForEach** block (highlighted in blue) with a circular red mark above it.
- ForEachGit** block (highlighted in blue) connected to the **ForEach** block.
- Icons for trash, copy, paste, and delete.

we need to
provide
values for
3 parameters



we need to give something to run a loop. we will create a json file with arrays and Parameters inside.

Json file

```
{ git.json } X
1 [
2   {
3     "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Calendar.csv",
4     "p_sink_folder" : "AdventureWorks_Calendar",
5     "p_file_name" : "AdventureWorks_Calendar.csv"
6   },
7   {
8     "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Customers.csv",
9     "p_sink_folder" : "AdventureWorks_Customers",
10    "p_file_name" : "AdventureWorks_Customers.csv"
11  },
12  {
13    "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Product_Categories.csv",
14    "p_sink_folder" : "AdventureWorks_Product_Categories",
15    "p_file_name" : "AdventureWorks_Product_Categories.csv"
16  },
17  {
18    "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Product_Subcategories.csv",
19    "p_sink_folder" : "AdventureWorks_Product_Subcategories",
20    "p_file_name" : "AdventureWorks_Product_Subcategories.csv"
21  },
22  {
23    "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Products.csv",
24    "p_sink_folder" : "AdventureWorks_Products",
25    "p_file_name" : "AdventureWorks_Products.csv"
26  },
27  {
28    "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Returns.csv",
29    "p_sink_folder" : "AdventureWorks_Returns",
30    "p_file_name" : "AdventureWorks_Returns.csv"
31 }
```

```

32      },
33      "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Sales_2015.csv",
34      "p_sink_folder" : "AdventureWorks_Sales_2015",
35      "p_file_name" : "AdventureWorks_Sales_2015.csv"
36    },
37    {
38      "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Sales_2015.csv",
39      "p_sink_folder" : "AdventureWorks_Sales_2015",
40      "p_file_name" : "AdventureWorks_Sales_2015.csv"
41    },
42    {
43      "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Sales_2016.csv",
44      "p_sink_folder" : "AdventureWorks_Sales_2016",
45      "p_file_name" : "AdventureWorks_Sales_2016.csv"
46    },
47    {
48      "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Sales_2017.csv",
49      "p_sink_folder" : "AdventureWorks_Sales_2017",
50      "p_file_name" : "AdventureWorks_Sales_2017.csv"
51    },
52    {
53      "p_rel_url" : "KumarVaibhav27/Azure-End-to-End-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Territories.csv",
54      "p_sink_folder" : "AdventureWorks_Territories",
55      "p_file_name" : "AdventureWorks_Territories.csv"
56    }
57  ]

```

It will be feeding these 3 parameters to that activity.
and dynamically pushing the data to Azure.

Home > awsstroagedatalakekr | Containers >

parameters ...

Container

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

Overview

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

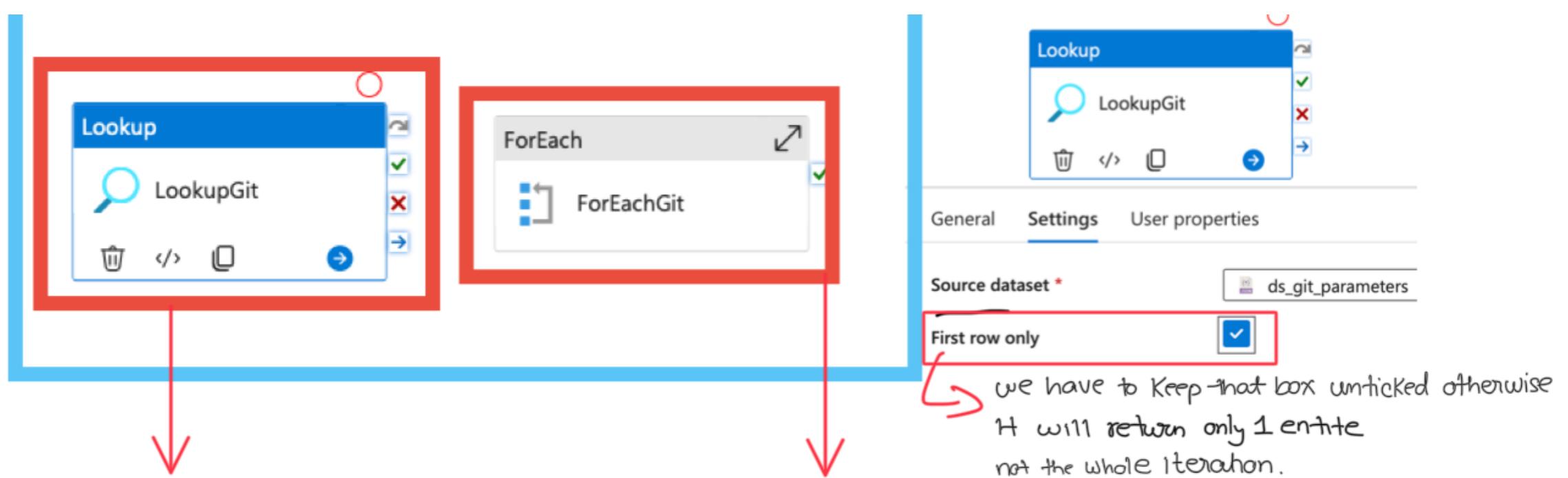
Diagnose and solve problems

Access Control (IAM)

Settings

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

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
git.json	5/28/2025, 9:21:45 PM	Hot (Inferred)		Block blob	2.8 KiB	Available



About Lookup Activity: Fetching Dynamic Data

The Lookup activity is designed to retrieve data from various sources, such as databases, files, or query execution results. The output can range from a single value to an array of objects or records.

Here are a few use cases:

Dynamically gather configuration data based on pipeline execution.
Obtain a list of file names for subsequent processing.

ForEach Loop: Iterating Through Data

The ForEach loop iterates through a collection of data, typically an array, to perform repeated tasks on each item. It executes a set of activities within the loop for each item in the collection.

Here are a few use cases:

Handle each record from a Lookup activity's output.
Copy each file in a list to a different location.
Execute separate activities for each ID retrieved.

Benefits of Lookup and ForEach activities

- The combination of Lookup and ForEach activities allows for flexible data processing based on the retrieved data.
- Efficiently handles repetitive operations on multiple records, making data workflows more streamlined.
- Breaks down complex pipelines into smaller, reusable activities, enhancing maintainability and scalability.

DynamicGitToRaw • ds_git_dynamic • ds_sink_dynamic •

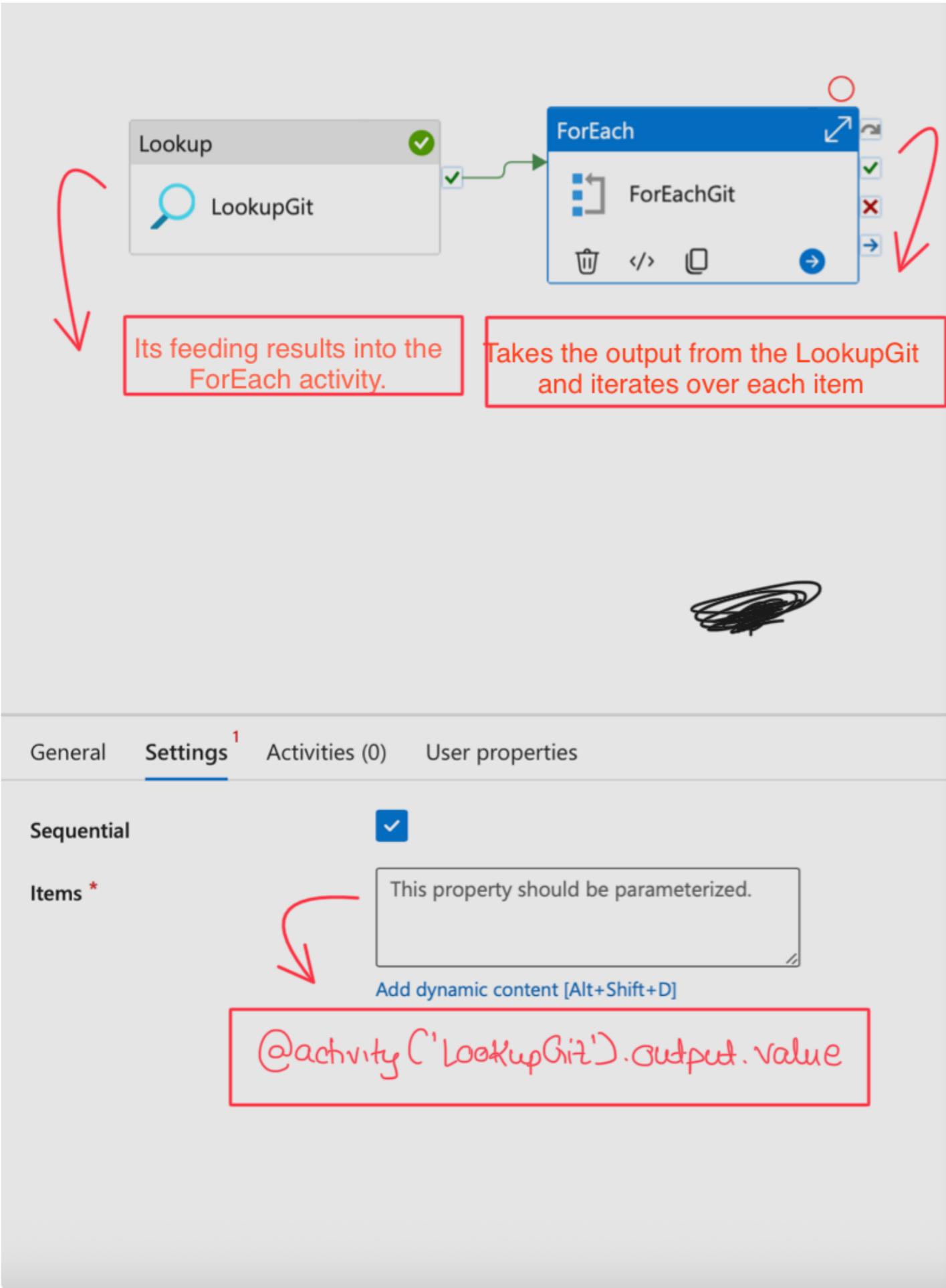
Validate

Pipeline expression builder

Add dynamic content below using any combination of:

```
@activity('LookupGit').output.value
```

output did not have



all the information, because we just want to pass the array. and Array was stored in value Key.

After that, we will cut the "Copy Data" block and paste it inside the "ForEachGit" and then we will give value for:

DynamicGitToRaw > ForEachGit

Copy data

DynamicCopy

Source

Dataset * ds_git_dynamic Open New Preview data Learn more

Dataset properties

Name	Value
p_rel_url	Value @item().p_rel_url

Sink

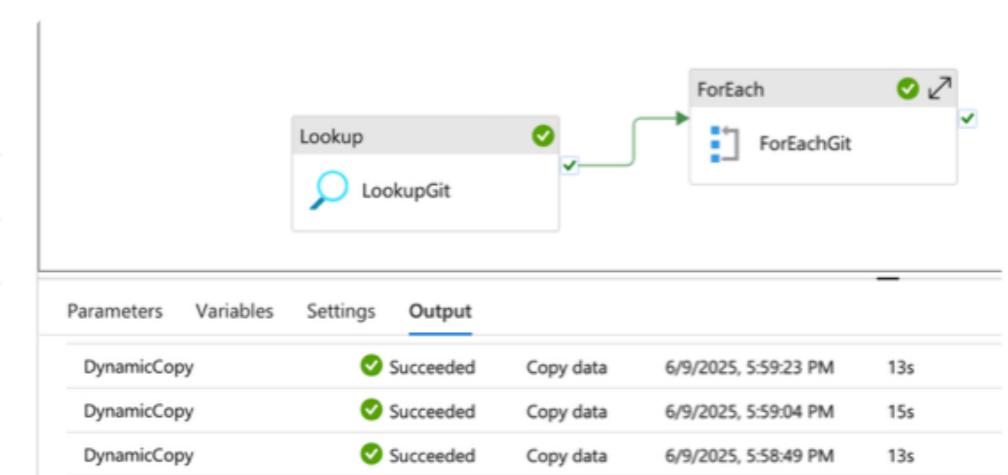
Sink dataset * ds_sink_dynamic Open New Learn more

Dataset properties

Name	Value
p_sink_folder	Value @item().p_sink_folder
p_file_name	Value @item().p_file_name

p_rel_url in Source

P_Sink_folder
P_file_name in Sink



All the tiles are successfully copied or pushed into the bronze folder.

DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:58:32 PM	15s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:58:17 PM	13s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:58:01 PM	14s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:57:46 PM	13s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:57:29 PM	16s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:57:12 PM	16s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:56:56 PM	14s
DynamicCopy	✓ Succeeded	Copy data	6/9/2025, 5:56:43 PM	12s
ForEachGit	✓ Succeeded	ForEach	6/9/2025, 5:56:42 PM	2m 56s
LookupGit	✓ Succeeded	Lookup	6/9/2025, 5:56:23 PM	19s



Now next Phase : Azure Databricks Resources

Starting with compute

Once our cluster is ready then we can begin using Databricks.

The screenshot shows the Microsoft Azure Databricks Compute configuration interface. On the left, a sidebar lists various categories: Workspace, Recents, Catalog, Workflows, Compute (which is selected and highlighted in blue), Data Engineering, Job Runs, AI/ML, Playground, Experiments, Features, Models, and Serving. The main content area is titled "Kumar Vaibhav's Cluster". It includes tabs for Configuration, Notebooks (0), Libraries, Event log, Spark UI, Driver logs, Metrics, Apps, and Spark compute UI - Master. Under Configuration, there are two radio buttons: "Multi node" (selected) and "Single node". Below that are sections for Access mode (with "Single user" selected) and User access (with "Kumar Vaibhav" listed). The Performance section includes Databricks Runtime Version (13.3 LTS), Node type (Standard_F4s, 8 GB Memory, 4 Cores), and a checkbox for "Terminate after 10 minutes of inactivity". The Tags section indicates "No custom tags". At the bottom, there are links for "Advanced options" and "Advanced options".

How to access data from datalake?



Data access - (stored in Datalake)
we will create an application,
Known as service based application
and this application will have access
to azure datalake. And this
application will be used by databricks.

Could not continue working because I didn't
have the access to **MICROSOFT ENTRA ID** ->
APP REGISTRATION.

