Dynamic Azure Data Factory Pipeline with CI/CD

This document serves as a short project overview to present some of it's procedures, methodology and final outcome.

1. Overview

The idea of this project originated from taking the course of Azure Data Engineer Associate certification on Microsoft Learning platform. As an addition to everyday learning, I decided to practice the skills and test features on my own in the Azure environment.

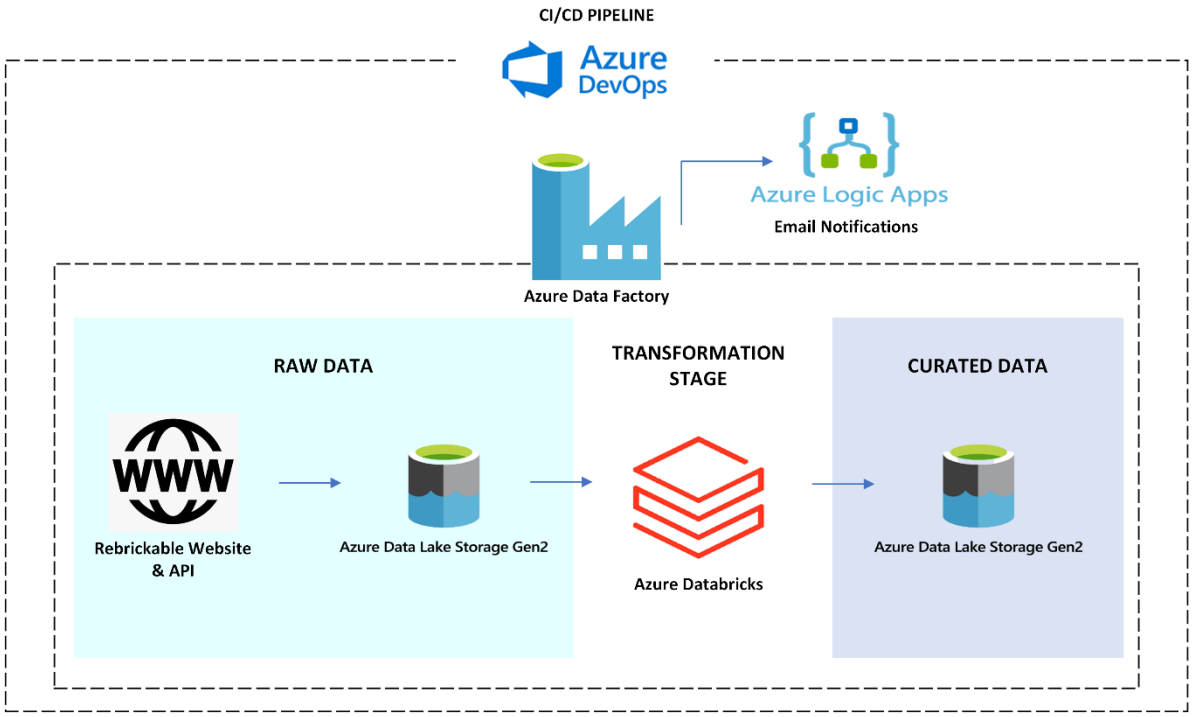
This project focuses on building a dynamic, scalable pipeline in Azure Data Factory together with the incorporation of CI/CD in Azure DevOps.

Data used in this project comes from the *Rebrickable* website (Rebrickable Website), which provides numerous files updated daily, ready for download, and also an API. Both names & number of files are customizable by JSON file in data lake. The same applies for the API files – for this purpose, two user accounts have been created to mimic a custom user-database, on base which we query data of user's Lego sets, parts, etc.

Resources & features used in this project:

- Azure Data Factory
- Azure Databricks
- Azure Logic Apps
- Azure DevOps
- Other common resources like Data Lake

2. Architecture Diagram

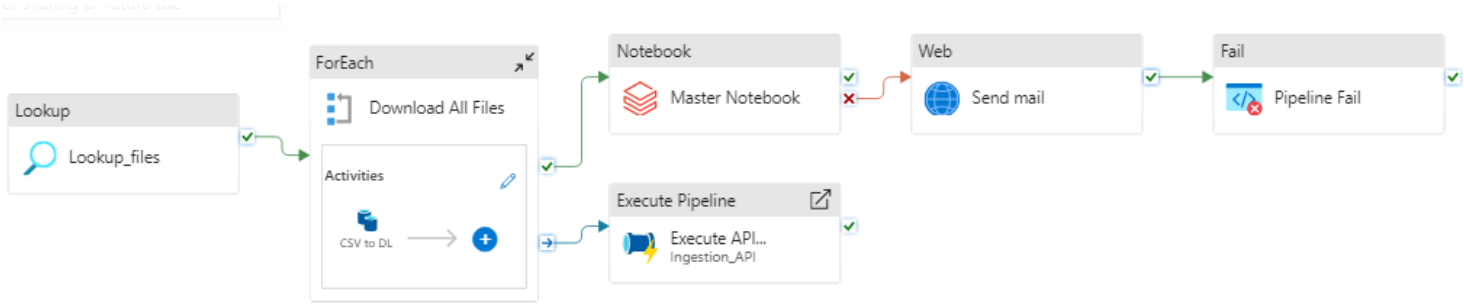


3. Ingestion phase

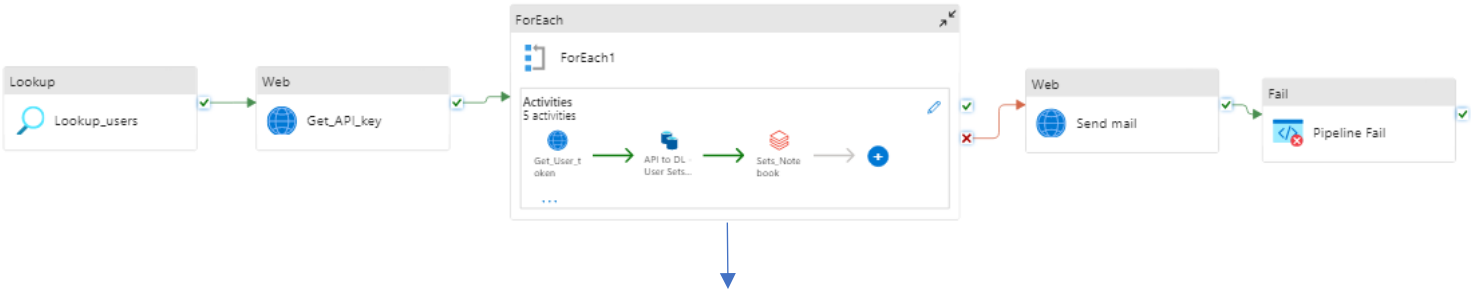
3.1 Pipelines

The Data Factory contains 2 pipelines, of which 1 is a child pipeline. Activities of them both are shown below:

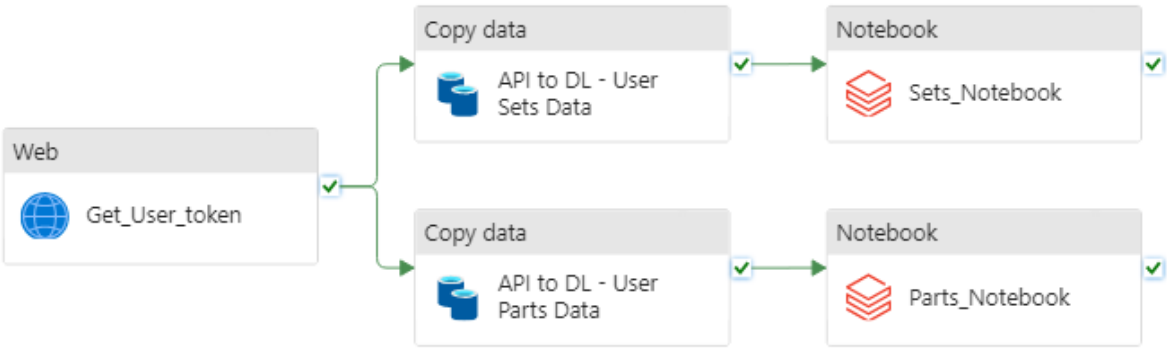
Ingestion_CSV pipeline:



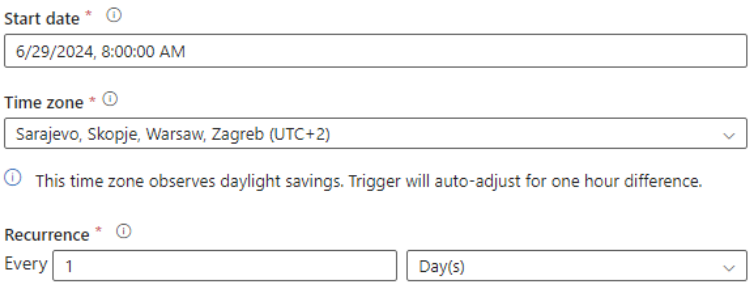
Ingestion_API pipeline:



Expanded ForEach view:

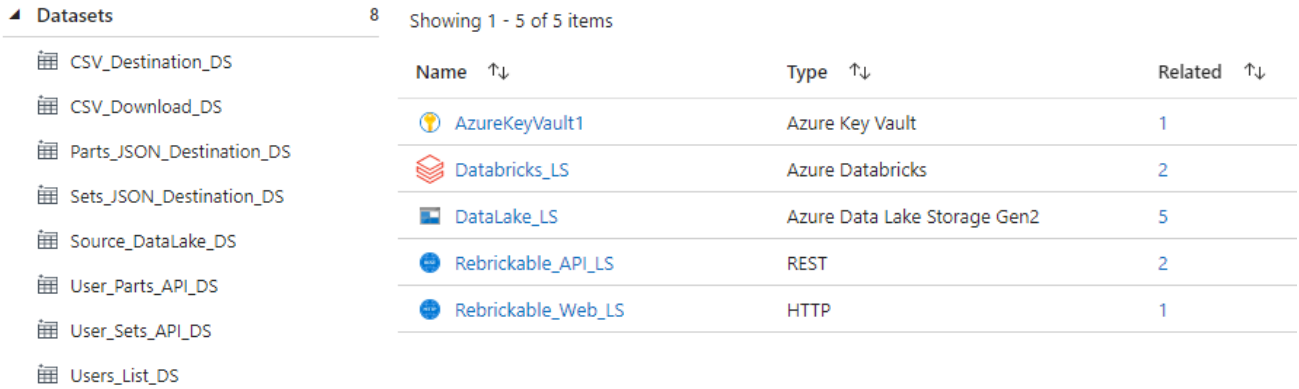


Pipelines are scheduled using a Daily Trigger:

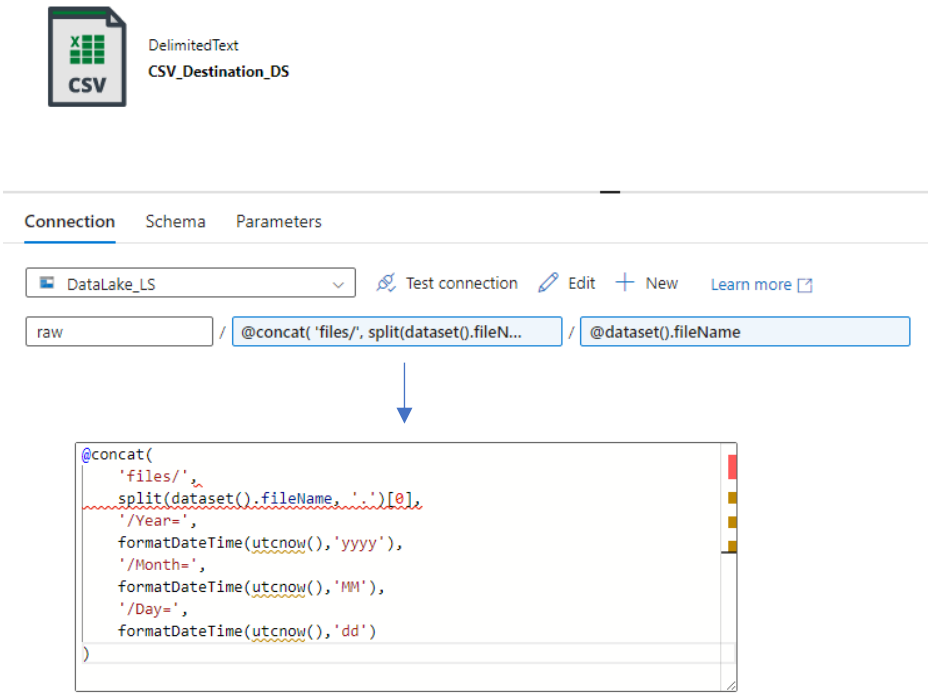


3.2 Datasets & Linked services

There are total 8 datasets and 5 linked services created.

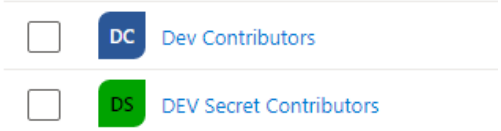


I will not dive into details of each of them, but it's worth noticing that the *CSV_Destination_DS* and both **_JSON_Destination_DS* datasets provide hierarchical structure to the raw container of the Data Lake:



When it comes to Linked services, mostly System Managed Identity was used for authentication. For Databricks, an AppRegistration was required.

I wanted to incorporate RBAC throughout whole project – to follow the best practices, I have created User Groups which got roles assigned, like Storage Blob Data Contributor.



As an exception though, at some point I've found it much better to use Access policies in the Key Vault instead of RBAC – granted Get & List permissions where it was necessary.

Showing 1 to 4 of 4 records.

Name	Email	Key Permissions	Secret Permissions
APPLICATION			
AzureDatabricks			Get, List
mateusz-lego-factory-dev			Get, List
UNKNOWN			
c24fcecb-580a-4568-95cd-5fb...			Get, List
USER			
Mateusz Halikowski	mat.halikowski_gmail.com#EXT...		Get, List, Set, Delete, Recover, ...

3.3 Functionality

Ingestion_CSV pipeline:

- The Lookup activity checks for the file list for download. As it was mentioned, it is a JSON file in the data lake:

downloadList/downloadList.json
Blob

Save Discard Download Refresh Delete

Overview Versions Edit Generate SAS

```
1  {
2    "files": [
3      "sets.csv.gz",
4      "themes.csv.gz",
5      "parts.csv.gz",
6      "part_categories.csv.gz",
7      "minifigs.csv.gz"
8    ]
9  }
10
```

- Each listed file is downloaded to ADLSg2 *raw* container
- Once downloaded, transformations are run using the Master Notebook activity, which runs multiple Databricks notebooks -> transformed data is written in delta format to *curated* container
- The child pipeline *Ingestion_API* is triggered
- In case of an error on the way, a LogicApp configured with Gmail account sends an e-mail to my personal address. Unlike the Outlook connector, the Gmail one has the option to customize the message, as I did below:

Message from mateusz-lego-factory-dev - Ingestion_CSV Odebrane ×

mat.halikowski@gmail.com 08:00 (12 godzin temu)

do mnie

This is a custom dynamic message from your pipeline - an error occured with run ID 4218ff1b-fd43-424b-a02c-e8172f78d9ba.

Ingestion_API pipeline:

- The lookup activity checks for users list – this, aswell, is a JSON file in the Data Lake. As said in the overwiew, a mini user-database has been created to allow API requests for various users LEGO collections etc.

usersList/usersList.json ...

Blob

Save Discard Download Refresh Delete

Overview Versions Edit Generate SAS

```
1 {
2   "users": [
3     "matpython720",
4     "crazymathew720"
5   ]
6 }
```

- The API key is retrieved from the connected Key Vault
- Also, the so called *User_token* is retrieved – similar to the key, just one is enough to query multiple users data. Files are downloaded according to the requests and land in ADLSg2 *raw* container. Results are paginated, and the requests are limited.

Request interval (ms) 1000

Additional headers + New Delete

	Value
Authorization	@concat('key ',activity('Get_API_key')...

Pagination rules + New Delete

Name		Value	
AbsoluteUrl		Body	next

- Transformations are run using assigned Databricks notebooks
- Same Email notification is enabled as previously

4. Transformation phase

4.1 Databricks

There are total 7 notebooks in Databricks workspace, where 1 of them, the Master Notebook, is just for running the following 3 notebooks in a sequence.

mat.halikowski@gmail.com ☆ ⋮ Share Create

Name	Type	Owner	Created at
Master Notebook	Notebook	Mateusz Halikowski	2024-06-22 11:06:30
Minifigs Notebook	Notebook	Mateusz Halikowski	2024-06-18 19:19:55
Parts Notebook	Notebook	Mateusz Halikowski	2024-06-19 11:07:45
Sets Notebook	Notebook	Mateusz Halikowski	2024-06-19 10:41:36
User Parts Notebook	Notebook	Mateusz Halikowski	2024-06-21 21:58:55
User Sets Notebook	Notebook	Mateusz Halikowski	2024-06-21 14:43:30

Each Notebook takes 4 parameters :

- Year
- Month
- Day
- Storage name

The Notebooks referring to API results also take User name parameter. All notebooks are available in my GitHub repo. A small User Parts Notebook insight to show data before & after transformations:

3 days ago (<1s) 3 Python

```
userName = dbutils.widgets.get("userName")
year = dbutils.widgets.get("year")
month = dbutils.widgets.get("month")
day = dbutils.widgets.get("day")
storageName = dbutils.widgets.get("storageName")
```

6/22/2024 (2s) 4 Python

```
user_parts_df = spark.read.json(f"abfss://raw@{storageName}.dfs.core.windows.net/user_files/{userName}/parts/Year={year}/Month={month}/Day={day}/*.json")
```

(1) Spark Jobs

6/22/2024 (1s) 5 Python

```
user_parts_df.show()
```

(1) Spark Jobs

```
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Lime]], [3...|
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Dark Tan]]...|
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Pearl Gold...|
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Black]], [...|
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Black]], [...|
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Black]], [...|
| 4017|https://rebrickab...|https://rebrickab...|[{{{{{[Medium Nou...|
```

After transformations:

Table

	partID	partName	partCat...	colorID	colorNa...	colorRGB	partURL	partIm...	quantity	
1	15533	Brick Spe...	5	71	Light Blu...	A0A5A9	https://r...	https://c...	6	155
2	973e067...	Torso, O...	60	0	Black	05131D	https://r...	https://c...	1	973
3	3001	Brick 2 x 4	11	2	Green	237841	https://r...	https://c...	5	300
4	35480	Plate Spe...	9	4	Red	C91A09	https://r...	https://c...	4	354
5	87580	Plate Spe...	9	25	Orange	FE8A18	https://r...	https://c...	2	875
6	10126	Hand Hul...	60	326	Olive Green	9B9A5A	https://r...	https://c...	1	101
7	6083	Rock Pan...	33	72	Dark Blui...	6C6E68	https://r...	https://c...	2	608
8	3622	Brick 1 x 3	11	72	Dark Blui...	6C6E68	https://r...	https://c...	19	362
9	3007	Brick 2 x 8	11	14	Yellow	F2CD37	https://r...	https://c...	2	300
10	35787	Tile 45° C...	15	320	Dark Red	720E0F	https://r...	https://c...	3	357
11	87580	Plate Spe...	9	71	Light Blu...	A0A5A9	https://r...	https://c...	22	875
12	3626cpr2049	Minifig H...	59	78	Light Nougat	F6D7B3	https://r...	https://c...	3	362
13	973c03h...	Torso Ar...	60	0	Black	05131D	https://r...	https://c...	1	973
14	11153	Slope Cu...	37	0	Black	05131D	https://r...	https://c...	28	111

The files downloaded directly from the website are cleaned, deduplicated, additional columns are added. Some datasets are joined together as well. Final output results in files amount reduction (5 to 3).

When it comes to data obtained from the API, the paginated output is merged, also cleaned, deduplicated and transformed to readable tables with additional columns. Separate datasets are created for each user.

4.2 Data Lake

Both raw and curated containers have different hierarchy. The pre-transformed data is stored in directories parametrized by file name and date components and stored as raw data, without even minor transformations. The naming parameters are obtained in the pipeline(s).

Location: raw / files / minifigs / Year=2024 / Month=06 / Day=30

Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
[..]				
minifigs.csv.gz	6/30/2024, 8:00:59 AM	Hot (Inferred)		Block blob

For the curated data, I chose delta format and directory parametrization with file name and/or user name:

Location: curated / user_files / crazymathew720 / parts

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

Name	Modified	Access tier	Archive status	Blob type
[..]				
_delta_log				
part-00000-1160390a-1ffa-4208-ae23-f5985145c786-c00...	7/1/2024, 7:18:16 PM	Hot (Inferred)		Block blob
part-00000-bf899685-d703-461d-a57d-c311cda0c0d7-c0...	6/30/2024, 8:06:40 AM	Hot (Inferred)		Block blob

Location: curated / files / minifigs

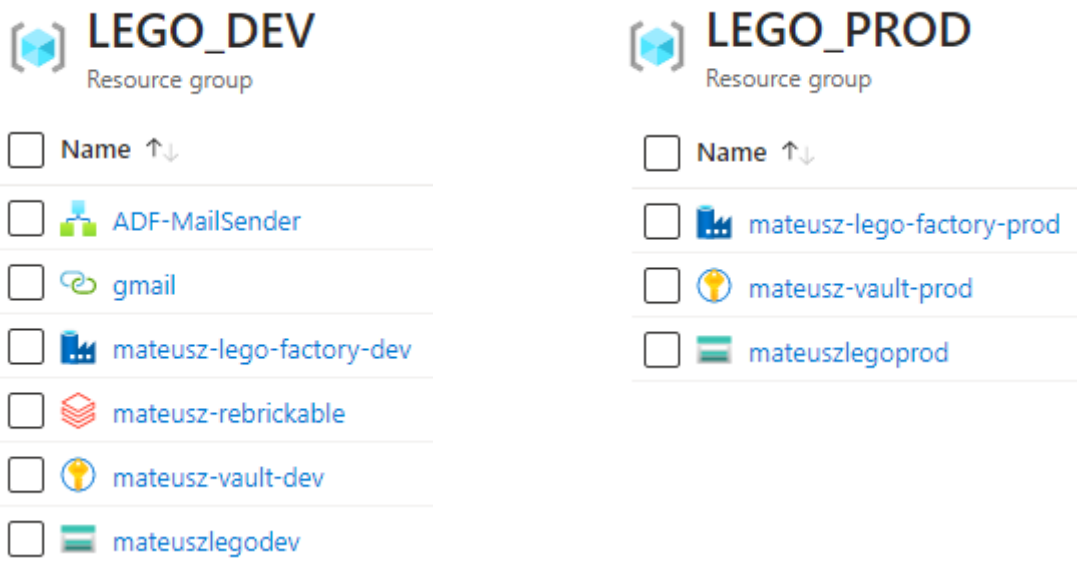
Search blobs by prefix (case-sensitive) Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
[..]				
_delta_log				
part-00000-281f0e5c-35ae-4217-a717-b274333a7423-c0...	7/2/2024, 8:09:26 AM	Hot (Inferred)		Block blob
part-00000-3f4fa9c9-5dca-4d0e-8ba9-137992b3ae56-c0...	7/1/2024, 7:18:16 PM	Hot (Inferred)		Block blob
part-00000-8f01d989-58e6-4f22-9f4f-7ad458b733a6-c00...	6/29/2024, 12:59:55 AM	Hot (Inferred)		Block blob

5. CI/CD

As a bonus to this Azure data engineering project, I decided to add CI/CD feature using Azure DevOps and ARM templates.

Two resource groups have been created, in order to enable CI/CD:

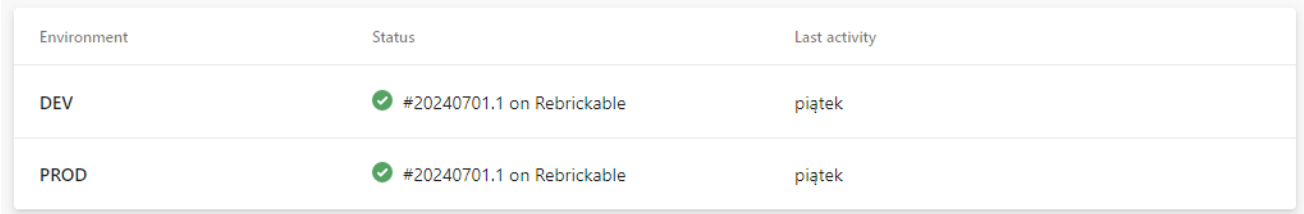


To not overcomplicate this project, I decided to keep only one instance of Databricks & Logic Apps. Other resources are duplicated as visible.

Another key points:

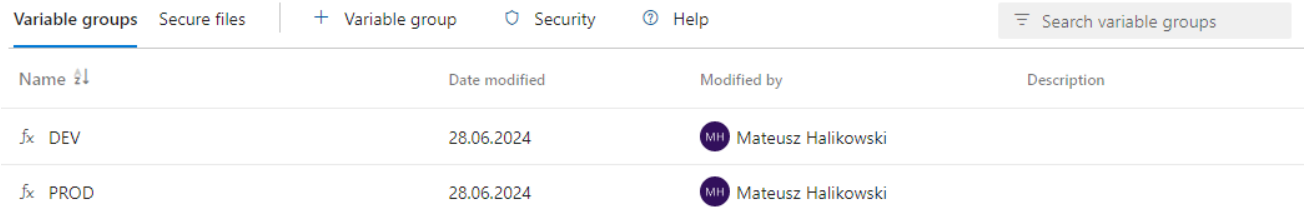
- Git repository for DEV factory was configured in DevOps
- Storage Account name and Key Vault name were added as global parameters in the Data Factory to allow proper templating
- Because of only one Databricks workspace present, an additional storage account connection was added to Databricks cluster configuration – a cluster-level connection to ADLSg2 has been set to avoid adding it in each Notebook separately.
- A plug-n-play DevOps template was used from Adam Marczak's GitHub repository [Adam Marczak's plug-n-play DevOps Templates]

Similarly to Azure resources, two Environments were created in Azure DevOps:

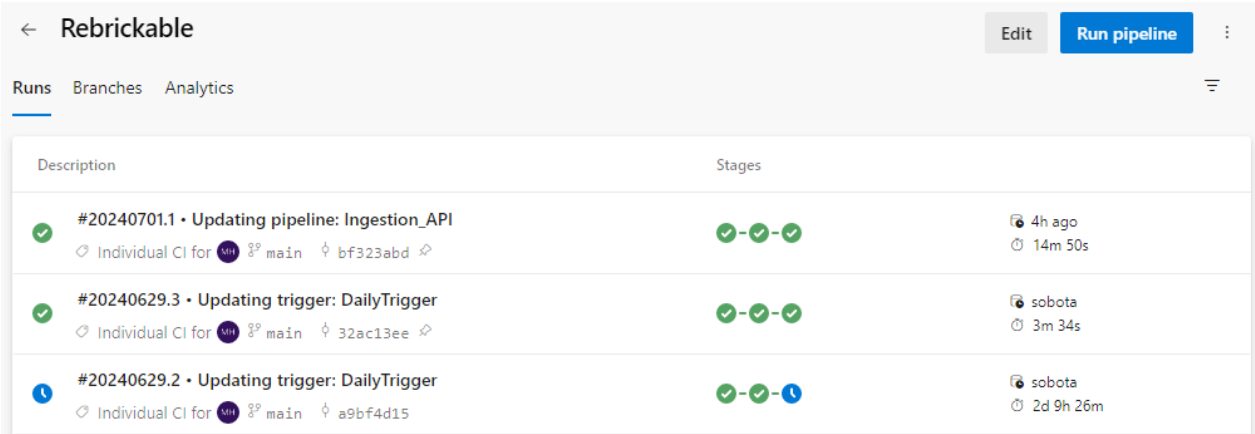


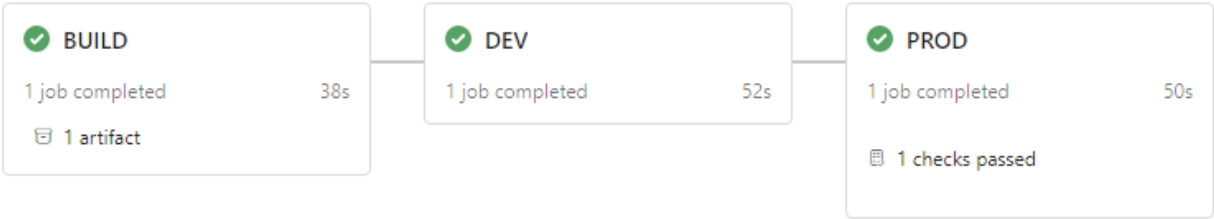
Environment	Status	Last activity
DEV	#20240701.1 on Rebrickable	piątek
PROD	#20240701.1 on Rebrickable	piątek

Also two Variable groups:



And finally a CI/CD pipeline:





6. Summary

This small project has finished successfully and helped me a lot with practicing during Azure certification. This was my first time using these services by myself and I have learned a ton, together with discovering the Azure's immense capabilities.

On the way, I have managed to solve many issues like API requests throttling, access control errors and ARM parametrization errors – and even if those were mostly my faults as a beginner, I have learned a lot from them, thus I'm glad they have happened.