

Azure Data Factory

Azure Data Factory is a **fully managed, serverless** data integration tool that is used for **data ingestion, data preparation and data transformation** at scale.

Characteristics:

- ❖ Fully Managed Service
- ❖ Serverless
- ❖ Data Integration Service
- ❖ Data Transformation Service
- ❖ Data Orchestration Service
- ❖ Data Migration Tool
- ❖ Data Streaming Service
- ❖ Suitable for complex Data transformation
- ❖ Data Storage Service

Explanation:

1. **Fully Managed Service:** A fully managed service means that Azure Data Factory takes care of all the technical stuff behind the scenes. You don't need to worry about setting up servers, managing infrastructure, or dealing with complex configurations. It's like having a team of experts handling everything for you so that you can focus on using the service without the hassle of managing it yourself.
2. **Serverless:** In the case of Azure Data Factory, being serverless means that you don't have to worry about provisioning or managing servers. You don't need to think about how much computing power you need or how to scale it. The service automatically adjusts resources based on your workload, so you can focus on your data integration tasks without getting bogged down in server-related details.
3. **Data Integration Service:** Azure Data Factory is a data integration service that helps you combine and move data from different sources. It acts like a bridge between various systems, allowing you to bring data together from different places, such as databases, files, and cloud storage. It simplifies the process of collecting and organizing data from multiple sources, making it easier to work with.
4. **Data Transformation Service:** Data transformation is the process of changing data from one format to another or applying specific operations to make the data more useful. In Azure Data Factory, the data transformation service allows you to perform these operations on your data. For example, you can clean up data, convert formats, aggregate information, or apply complex calculations to transform your data into a desired format or structure.

5. Data Orchestration Service: Data orchestration refers to the coordination and sequencing of data integration workflows. Azure Data Factory provides a data orchestration service that helps you define and manage the flow of your data pipelines. It allows you to set up dependencies, schedule activities, and ensure that data is moved and transformed in the right order and at the right time.

6. Data Migration Tool: A data migration tool helps you move data from one place to another. In the case of Azure Data Factory, it provides functionalities to migrate data from different sources, such as on-premises databases or files, to cloud-based storage or databases. It simplifies the process of transferring data securely and efficiently, reducing the effort and time required for data migration tasks.

7. Data Streaming Service: Data streaming involves the continuous and real-time processing of data as it is generated or received. Azure Data Factory offers a data streaming service that enables you to ingest and process streaming data from various sources. It allows you to handle high-volume, real-time data streams and perform near-instantaneous data transformations and analytics.

8. Suitable for Complex Data Transformation: Azure Data Factory is designed to handle complex data transformation tasks. It provides a range of powerful features and functionalities to handle intricate data manipulation scenarios. Whether you need to clean, enrich, merge, or transform your data in complex ways, Azure Data Factory can help you accomplish these tasks efficiently and effectively.

9. Data Storage Service: Data storage refers to the places where you can store your data, such as databases, files, or cloud storage. Azure Data Factory integrates with various data storage services, allowing you to read and write data from and to different storage systems. It provides seamless connectivity to popular data storage platforms, making it easy to access and work with your data wherever it is stored.

In summary, Azure Data Factory is a fully managed and serverless service that helps you integrate, transform, and orchestrate your data from different sources. It simplifies complex data tasks, provides efficient data migration capabilities, supports real-time data streaming, and seamlessly connects to various data storage services. It's like having a team of experts managing your data workflows, so you can focus on using and transforming your data without worrying about the technical details.

Project Overview

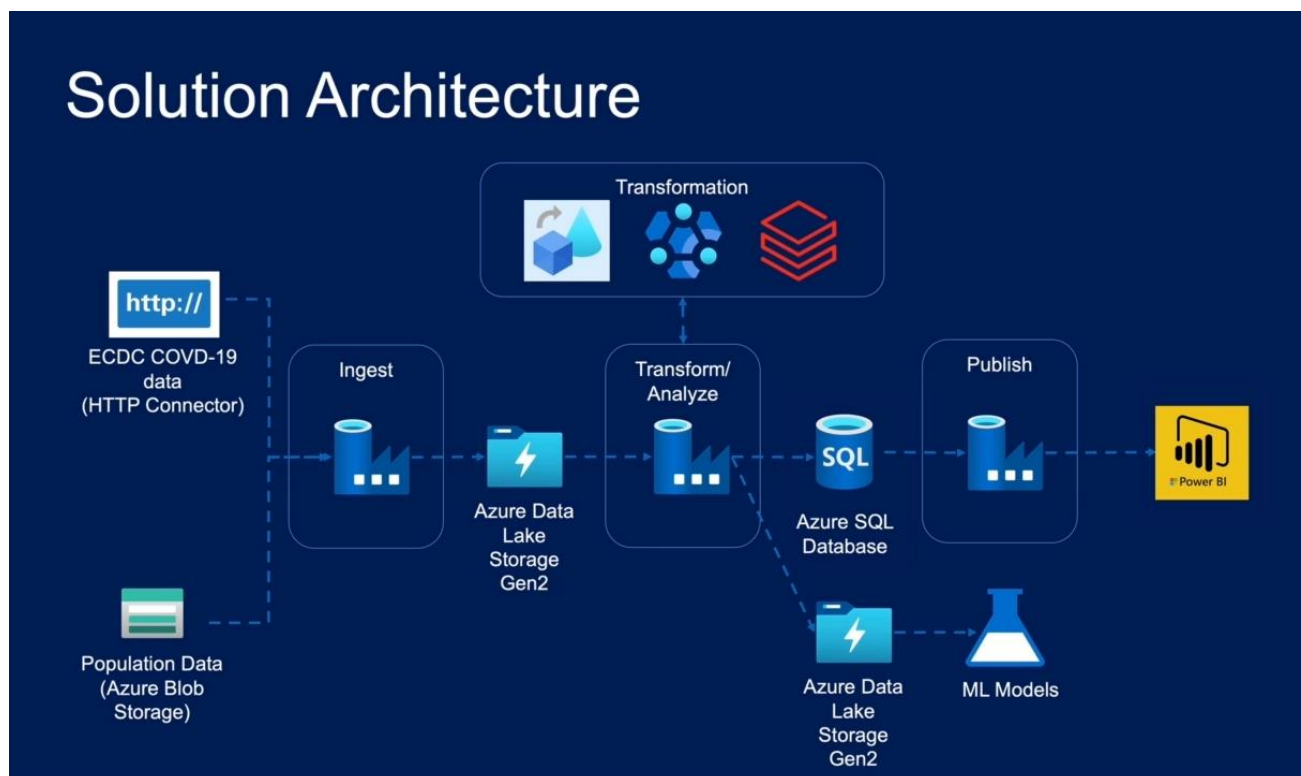
The project name is Insight Factory where I will use Azure Data Factory to prepare a data platform so that the ML Engineers can implement their algorithms on top of the data present there and the analysts can use this data to report the trends using a reporting tool like Power BI.

In this I am using data from ECDC (European Center for Disease Prevention and Control) website as the source of Covid-19 data that we will use in this project.

Also, I will use Eurostat website for population by age related data.

The main motive of this project is to completely understand the Azure Data Factory and its functionalities.

Solution Architecture Overview



For this project I will use Azure Blob Storage to store the Eurostat data and for ECDC we will ingest the data using ADF directly using HTTP Connector.

Once the data is ingested, I will dump that data in Azure Data Lake Storage (Gen2).

Now using ADF I will then transform the data using Azure Data Flow, Azure HD Insight and Azure Databricks where Azure data factory will act as a Transformation and Orchestration service.

After that I will dump the data in Azure Data Lake Storage (Gen2) for ML Predictions and Azure SQL Databases for reporting purposes.

At the end we will make a simple report using Power BI.

Environment Set Up

Resources Needed:

Azure Subscription

Data Factory

Blob Storage Account

Data Lake Storage Gen2

Azure SQL Database

Azure Databricks Cluster

HD Insight Cluster

Process

Copy data from Blob Storage to Data Lake

Eurostat population by age data is present in Azure Blob Storage (storageaccountloveleen) in population container.

We have to copy this data from azure blob storage to azure data lake gen 2 (datalakeloveleen) in raw container.

Scenario: Our customer copies their files in Azure Blob Storage at 11:00 PM every day. We want to copy that file into our Data Lake when the file is valid and if it is invalid we want to send an email to all the concern persons.

Solution:

- We will create a pipeline in Azure Data Factory and trigger that at 11:30 PM every day.
- This pipeline will first validate whether the new file has appeared in the container or not.
- When it finds a new file, it will then check whether the file has the expected number of columns or not.
- If everything goes right, it will copy the file from Blob to Data Lake and then delete the existing file from blob. (Alternative: If we do not want to move the file from Blob to Data Lake and just want to create a copy there in Data Lake then we can exclude the delete activity. Instead of that we can use get meta data activity to get the last modified date to check whether it is latest or not)
- If the file content is not as expected, then an email will be sent to all the concerned persons.

Process:

- Create linked service to Azure Blob Storage and Azure Data Lake Gen2

Edit linked service

 Azure Blob Storage [Learn more](#) 

Name *

AzureBlobStorage2

Description

Connect via integration runtime * ⓘ

AutoResolveIntegrationRuntime

Authentication type

Account key

Connection string

Azure Key Vault

Account selection method ⓘ

☐ From Azure subscription ☒ Enter manually

Storage account name *

storageaccountloveleen

Storage account key

Azure Key Vault

Storage account key *

.....

Partioned DNS enabled ⓘ

☐


Endpoint suffix

core.windows.net


 Connection successful

Apply

Cancel

 Test connection


Edit linked service

 Azure Data Lake Storage Gen2 [Learn more](#) 

Name *

LinkedServiceDataLakeGen2

Description

Connect via integration runtime * 

AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method 

☐ From Azure subscription ☒ Enter manually

URL *


https://datalakeloveleen.dfs.core.windows.net/

Storage account key

Azure Key Vault

Storage account key *

.....

Test connection 

☒ To linked service ☐ To file path

Annotations

+ New

 Connection successful


 Test connection

Save

Cancel

- Once Linked Services gets created, create datasets for source and sink.
 - Provide the reference location i.e., the location where the file should be added or is present.
 - Provide the relevant Linked Services.
 - If file is compressed give the compression type and if not give none
 - If file is delimited give relevant delimiter

ds_population_by_age | pl_ingest_population... | ds_datalake_populat... X



DelimitedText
ds_datalake_population

Connection Schema Parameters

Linked service * LinkedServiceDataLakeGen2 [Test connection](#) [Edit](#) [+ New](#) [Learn more](#)

File path * raw / Directory / population_by_age.tsv [Browse](#) | [Preview data](#) [Detect format](#)

Compression type None

Column delimiter Tab (\t)

Row delimiter Default (\r\n, or \n)


Encoding Default(UTF-8)

Quote character Double quote (")

Escape character Backslash (\)

First row as header ☒

ds_population_by_age X | pl_ingest_population... •



DelimitedText
ds_population_by_age

Connection Schema Parameters

Linked service * AzureBlobStorage2 [Test connection](#) [Edit](#) [+ New](#) [Learn more](#)

File path * population / Directory / population_by_age.tsv.gz [Browse](#) | [Preview data](#) [Detect format](#)

Compression type gzip (.gz)

Compression level Optimal

Column delimiter Tab (\t)

Row delimiter Default (\r\n, or \n)

Encoding Default(UTF-8)

Quote character Double quote (")

Escape character Backslash (\)

First row as header ☒

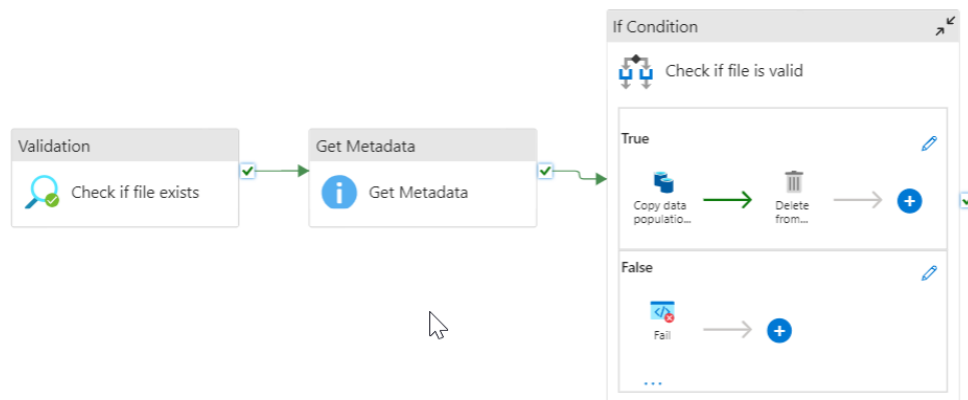
- Once datasets created create a pipeline `pl_ingest_population_data`
 - First there is a validation activity named Check if file exists that will be connected to the source dataset `ds_population_by_age` and it will check whether the file is there in source location or not
 - When it finds the file it will move to the next step
 - But if it does not find the file there it will follow the following steps:
 - I have set the timeout to be 30 seconds, sleep time 10 seconds and file size 1024 bytes
 - This activity will check for the file at source location in every 10 seconds if it finds the file it will move to next step

- If it reached timeout time it will be timed out and pipeline will fail.
- Second there is Get Meta Data activity named Get Meta Data. It will extract the exists, column count and size attributes from the location regarding the file.
- Third I have created a pipeline parameter named NumberOfColumns with default value as 13, with which the pipeline executor can pass the number of columns he is expecting that file should have.
- Fourth there is an if activity named Check if file is valid which has an expression defined as :`@equals(activity('Get Metadata').output.columnCount, pipeline().parameters.NumberOfColumns)`
 - If the value returns to be true it will then copy the file with copy data activity named Copy data population by age and then will execute the delete activity named Delete from source to delete the source file
 - If value returned as false, the pipeline would fail with a custom error message configured using fail activity named as Fail which will give Error Code: 500 and Message as Mismatched Column Number.
 - In parallel we have to send email also. For that I have used web activity.
 - But first we have to create a logic app in order to have API endpoint URL
- To create a logic App
 - Go to portal.azure.com.
 - Click on create a resource.
 - Click on Logic App.
 - Provide necessary details related to subscription, name etc.
 - Click on create.
 - Go to Logic App Designer Interface
 - Select Request Connector
 - Select When a HTTP request is received trigger.
 - In this click on use sample payload to specify the values you want to receive from ADF as arguments.
 - In the dialog box, type all the parameters in json format.


```
{
    "Pipeline_name": "",
    "Error_message": "",
    "datafactory_name": "",
    "run_id": ""
}
```
 - Here the empty strings on right side of colon specifies that the incoming value will be in string, and we are not providing any default value

- Then click on create a parameter.
 - Select method and give value as POST.
 - Then click on plus to specify an action.
 - Click on gmail connector.
 - Provide your creds for authentication with which the email would get sent.
 - Then select parameter to, email body, subject and importance and provide the parameters created before by clicking on dynamic content.
 - Click on save and copy the URL so generated.
- Once you get the URL, copy that in web activity and select the method as POST.
 - In the body section type in all the parameters that are required to be passed to logic Apps. This should be written in JSON format, and we will use system parameters there and hard-core values for error message. The content is:

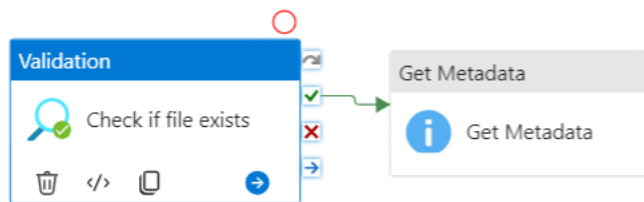

```
{
  "datafactory_name": "@{pipeline().DataFactory}",
  "Pipeline_name": "@{pipeline().Pipeline}",
  "run_id": "@{pipeline().RunId}",
  "Error_message": "Error Code 500: Column Mismatched,
    Number Of Columns Found:
    @{pipeline().parameters.NumberOfColumns}"
}
```
 - Also, in headers you should add one as Content-Type with value application/json
 - **NOTE:** if you write or pass all values in XML instead of JSON use application/XML



Parameters Variables Settings Output

+ New | Delete

<input type="checkbox"/> Name	Type	Default value	
<input type="checkbox"/> NumberOfColumns	String	13	



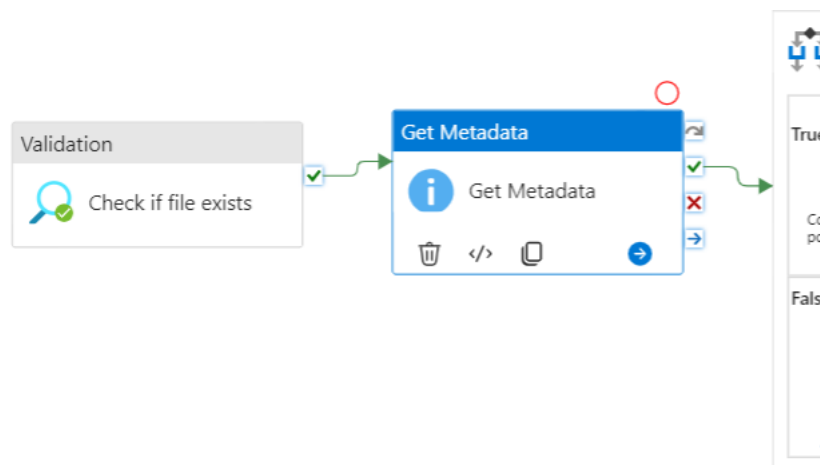
General **Settings** User properties

Dataset * ⓘ [Open](#)

Timeout ⓘ

Sleep ⓘ

Minimum size ⓘ



General **Settings** User properties

Dataset * [Open](#) [New](#) [Learn m](#)

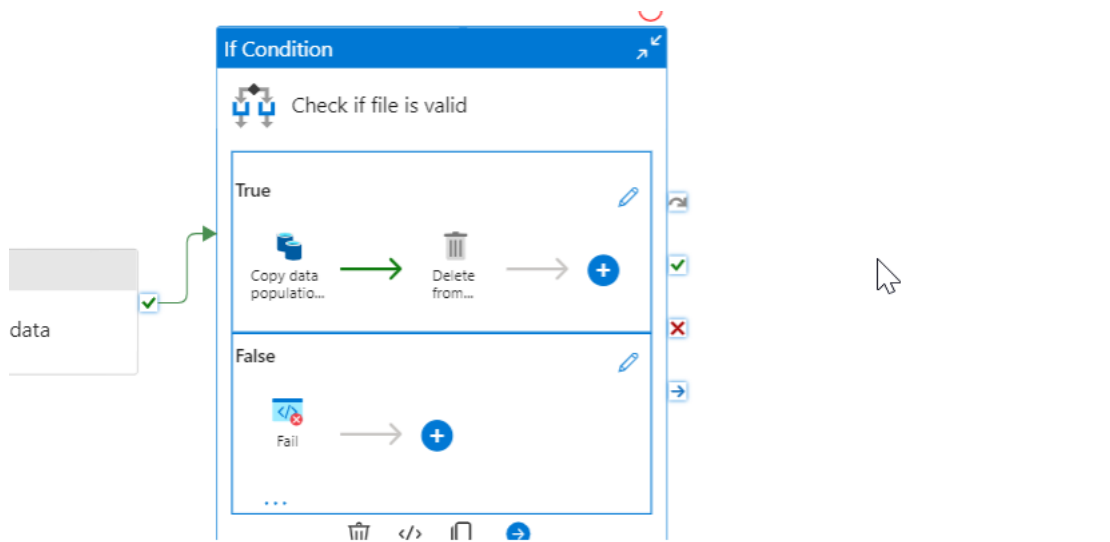
Field list * [New](#) [Delete](#)

☐ Argument

☐ Column count

☐ Exists

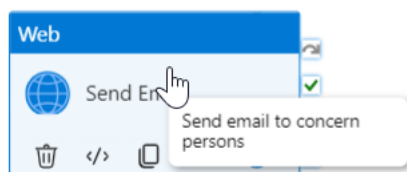
☐ Size



General **Activities (4)** User properties

Expression ⓘ @equals(activity('Get Metadata').out...

Case	Activity
True	Copy data popul... Delete from sour... 2 Activities
False	Fail 2 Activities



General **Settings** User properties

URL * ⓘ https://prod-01.southindia.logic.azure.co...
⚠ Information will be sent to the URL specified. Please ensure you trust the URL entered.

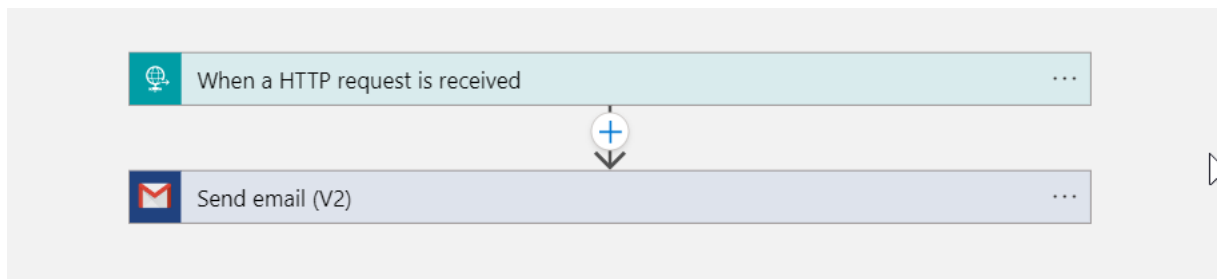
Method * ⓘ POST

Body { "datafactory_name": "@(pipeline()....

Authentication ⓘ None

Headers * ⓘ + New | Delete

<input type="checkbox"/> Name	Value
<input type="checkbox"/> Content-Type	application/json



When a HTTP request is received ⓘ ...

HTTP POST URL

<https://prod-01.southindia.logic.azure.com:443/workflows/1ab2bdf3ce8e49f9baf3213f8921a47b/tri> 📄

Request Body JSON Schema

```
{
  "datafactory_name": {
    "type": "string"
  },
  "run_id": {
    "type": "string"
  }
}
```

[Use sample payload to generate schema](#)

Method

POST

Add new parameter

Send email (V2) ...

*To

loveleenverma73@gmail.com

Subject

Pipeline Fail

Body

Font 12 **B** *I* U 🔗 🗑️

There is a column Mismatch in

Datafactory: datafactory_name x

Pipeline: Pipeline_name x

Run Id: run_id x

Error Message: Error_message x

Add new parameter