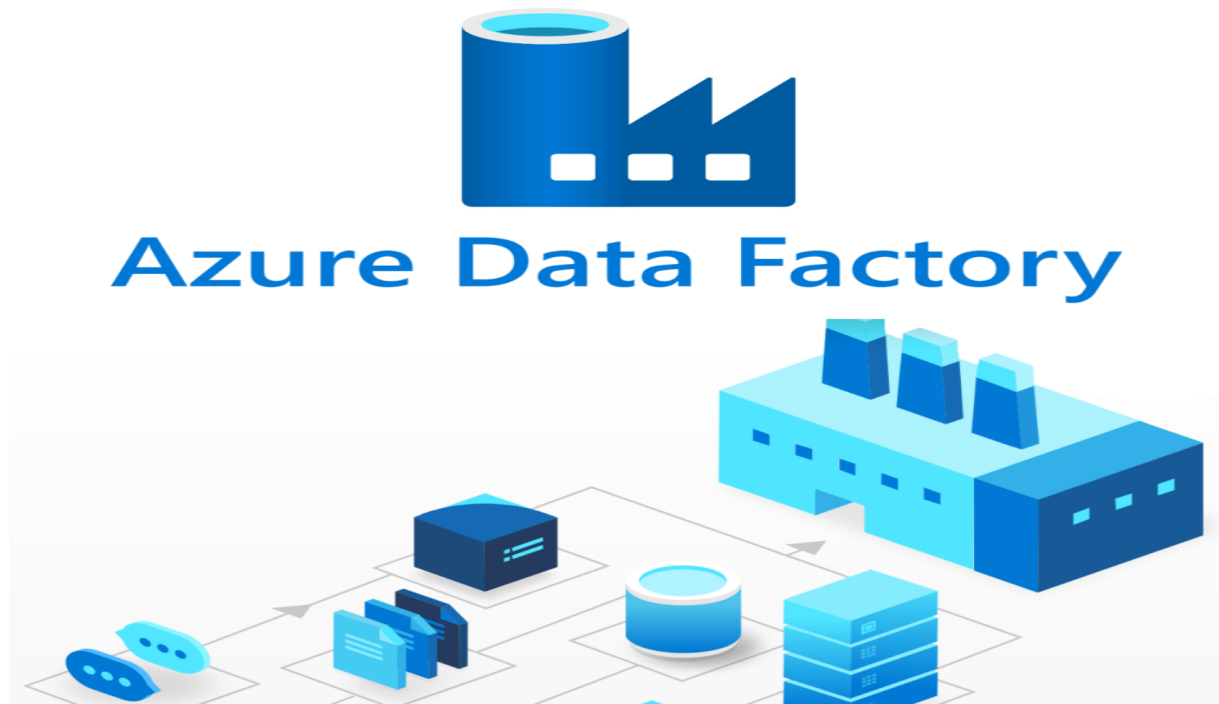


Azure data factory project.



*In this document you will find a project utilizing Azure Data Factory to extract, transform, and load data from a source and file within blob storage into a SQL database. This project involved setting up data pipelines to schedule data movement and transformation activities in data flow and databricks, as well as integrating with other Azure services such as Azure Data Lake Storage and Azure Data Lake. Through the use of Azure Data Factory, I was able to efficiently process and move data. Also, in this project we will show CI / CD techniques.*

## Contents

Azure data factory project. ....	1
1. Introduction .....	4
1.1 Project overview:.....	4
2. Solution Architecture .....	5
3. Data ingestion.....	5
3.1 Data ingestion with Azure blob storage .....	6
Step 1 - Copy activity overview.....	6
Step 2 - Linked service.....	7
Step 3 - Blob data set (source).....	8
Step 4 - Data Lake data set (sink).....	9
Step 5 - Creating an ingestion pipeline .....	10
Step 6 - Trigger (ingestion) pipeline .....	11
3.2 Data ingestion from HTTP .....	12
Step 0 - Overview copy activity from HTTP .....	12
Step 1 - Created linked services.....	12
Step 1 - Create data set (http - source) .....	13
Step 2 - Create data set (DL sink) .....	14
Step 3 - Create pipeline for data ingestion.....	15
Step 4 - Create trigger for pipeline.....	16
Step 6 - Ecdc file list data set .....	16
4. Data transformation.....	17
4.1 Data flow.....	17
Step 0 – Overview transformation in data flow.....	17
(1) Step 1 - Create transformations with data flow .....	18
(1) Step 2 - Create data set for processed data .....	18
(1) Step 3 – Create pipeline .....	19
(2) Step 4 - Create transformations with data flow .....	20
Step 5 - Create data set for processed data (daily).....	20
Step 6 - Create data set for processed data (weekly) .....	21
Step 7 - Create pipeline .....	21
4.2 Databricks .....	23
Step 0 - Databricks overview .....	23
Step 1 – Mount cluster 1 / 2 .....	23

Step 1 – Mount cluster 2 / 2 .....	24
Step 2 – transform data 1 / 2 .....	25
Step 2 – transform data 2 / 2 .....	26
Step 4 – Create pipeline .....	27
5. Load into database .....	27
Step 1 – Create SQL script .....	28
Step 2 – Create linked service .....	29
Step 3 - Create sink dataset .....	30
Step 4 – Create pipeline .....	31
Step 5 – Create pipeline hospital admission data .....	32
Step 6 - Create pipeline testing data .....	33
6. Data orchestration – Making pipelines production ready .....	34
Step 1 – Build pipeline .....	34
Step 2 – Create trigger .....	35
7. Data reporting .....	36
Step 0 – Reporting overview in Power BI .....	36
Step 1 – Create report 1 / 2 .....	36
Step 1 – Create report 2 / 2 .....	37
8. CI / CD .....	38
Step 0 – CI / CD overview .....	38
Step 0 – Git configuration within DevOps option 1 .....	39
Step 0 – Git configuration within DevOps option 2 .....	39
Step 1 – Create git repo .....	40
Step 2 – Create Tools for azure (DEV) .....	41
Step 2 – Create Tools for azure (TEST) .....	41
Step 2 – Create Tools for azure (PROD) .....	42
Step 3 - Create release pipeline option 1 .....	42
Step 4 – Create build pipeline option 2 – 1 / 2 .....	43
Step 5 – Create release pipeline option 2 .....	44

## 1. Introduction

In this project we will be building a data platform for reporting and prediction of the covid-19 outbreak.

- Data sources: European Center for Disease prevention and Control (Eurostat)
- Data flows: azure data factory
- Data transformation: HDInsight and databricks
- Ingestion of transformed data: Data Lake
- Necessary data for reporting the data trends: SQL Data warehouse
- We will orchestrate all of these pipelines using Azure data factory.

### 1.1 Project overview:

#### **Our build data lake:**

Data Lake to be built with the following data, to aid Data Scientists to predict the spread of the virus/mortality

- Confirmed cases
- Mortality
- Hospitalization/ ICU Cases
- Testing Numbers
- Country's population by age group

#### **Our build data warehouse:**

Data Warehouse to be built with the following data

- Confirmed cases
- Mortality
- Hospitalization/ ICU Cases
- Testing Numbers

#### **The data sources:**

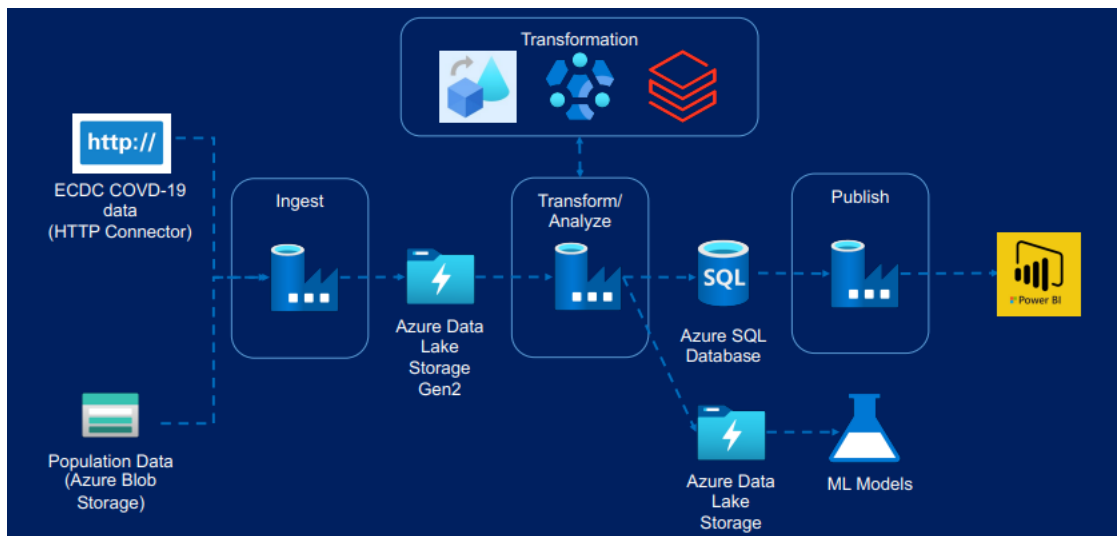
ECDC website:

- Confirmed cases
- Mortality
- Hospitalization / ICU Cases
- Testing Numbers

Eurostat Website:

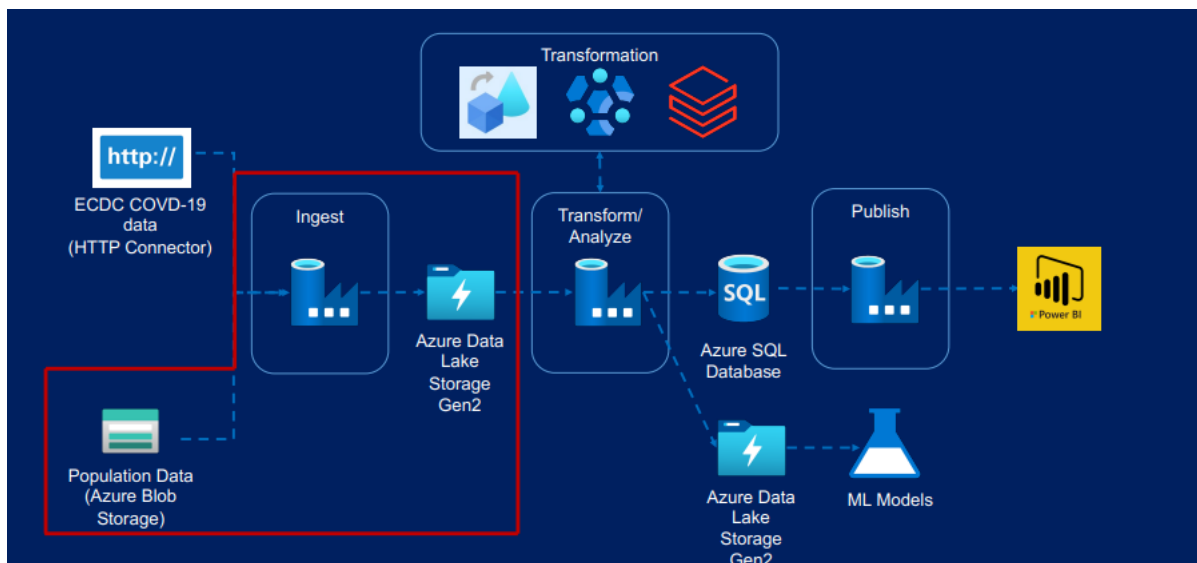
- Population by age

## 2. Solution Architecture



## 3. Data ingestion

### Overview



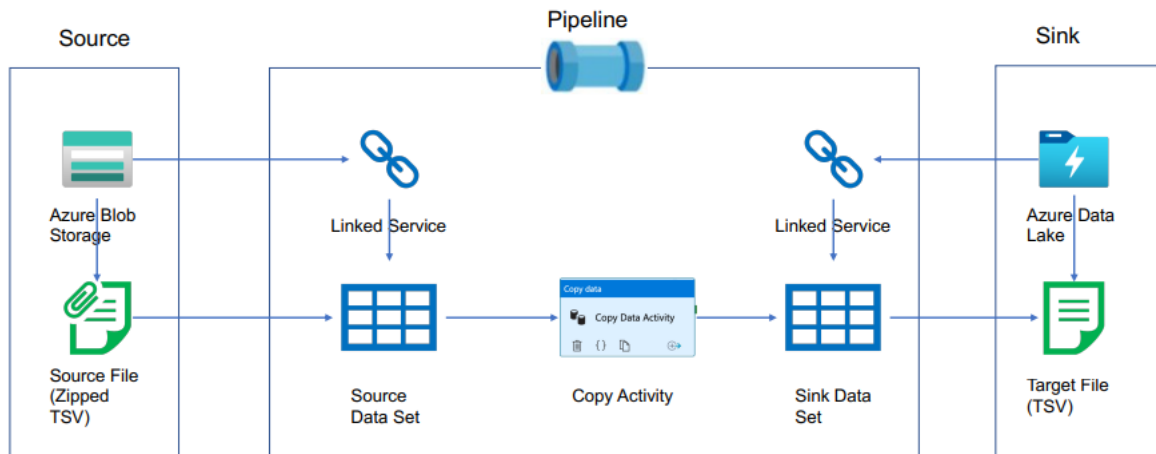
Worked with:

- Copy Activity
- Linked Services
- Datasets
- Validation Activity
- If Condition Activity
- Get Metadata Activity
- Web Activity
- Delete Activity

- Trigger



### 3.1 Data ingestion with Azure blob storage

#### Step 1 - Copy activity overview



## Step 2 - Linked service

### Edit linked service

 Azure Blob Storage [Learn more](#) 

**Name \***

ls\_ablob\_covidreportingsa

**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 \***

covidreportingsa0202

**Storage account key** **Azure Key Vault**

**Storage account key \***

.....

**Partitioned DNS enabled** ⓘ ☐

**Endpoint suffix**

core.windows.net

**Additional connection properties**

+ New

**Test connection** ⓘ

☒ To linked service ☐ To file path


**Annotations**

+ New

> Parameters

> Advanced ⓘ

Apply Cancel

 Test connection

### Step 3 - Blob data set (source)

The screenshot shows the Azure Data Factory interface. On the left, the 'Factory Resources' pane is expanded to 'Datasets' > 'raw' > 'ds\_population\_raw\_gz'. The main pane displays the configuration for this dataset, which is a 'DelimitedText' type. The configuration includes a 'Linked service' (ls\_ablob\_covidreportingsa), a 'File path' (population / Directory / population\_by\_age.tsv), and various formatting options like 'Compression type' (gzip), 'Column delimiter' (Tab), 'Row delimiter' (Default), 'Encoding' (Default), 'Escape character' (Backslash), 'Quote character' (Double quote), 'First row as header' (checked), and 'Null value'.

**Factory Resources**

- Pipelines: 9
  - Execute: 1
  - Ingest: 2
  - Process: 3
  - sqlize: 3
- Datasets: 16
  - lookup: 3
  - processed: 4
  - raw: 6
    - ds\_ecdc\_raw\_csv\_dl
    - ds\_ecdc\_raw\_csv\_http
    - ds\_population\_raw\_gz**
    - ds\_population\_raw\_tsv
    - ds\_raw\_cases\_and\_deaths
    - ds\_raw\_hospital\_admissions
  - sql: 3
- Data flows: 2
- Power Query: 0

**ds\_population\_raw\_gz**

DelimitedText  
ds\_population\_raw\_gz

**Connection** Schema Parameters

Linked service \* ls\_ablob\_covidreportingsa [Test connection](#) [Edit](#) [New](#) [Learn more](#)

File path \* population / Directory / population\_by\_age.tsv [Browse](#) [Preview data](#)

Compression type gzip (.gz)

Compression level Optimal

Column delimiter Tab (\t) [Edit](#)

Row delimiter Default (\r,\n, or \r\n) [Edit](#)

Encoding Default(UTF-8)

Escape character Backslash (\) [Edit](#)

Quote character Double quote (") [Edit](#)

First row as header ☒

Null value



## Step 4 - Data Lake data set (sink)

The screenshot shows the Azure Data Factory interface. On the left, the 'Factory Resources' pane is expanded to 'Datasets' > 'raw', where 'ds\_population\_raw\_tsv' is selected. The main pane shows a preview of the dataset as a 'DelimitedText' file named 'ds\_population\_raw\_tsv'. Below this, the 'Connection' tab is active, displaying the configuration for the 'ls\_adls\_covidreportingdl' linked service. The file path is set to 'raw / population / population\_by\_age.tsv'. The 'First row as header' checkbox is checked.

**Factory Resources**

- Pipelines: 9
  - Execute: 1
  - Ingest: 2
  - Process: 3
  - sqlize: 3
- Datasets: 16
  - lookup: 3
  - processed: 4
  - raw: 6
    - ds\_ecdc\_raw\_csv\_dl
    - ds\_ecdc\_raw\_csv\_http
    - ds\_population\_raw\_gz
    - ds\_population\_raw\_tsv**
    - ds\_raw\_cases\_and\_deaths
    - ds\_raw\_hospital\_admissions
  - sql: 3
- Data flows: 2
- Power Query: 0

**Connection** Schema Parameters

Linked service \*: ls\_adls\_covidreportingdl [Test connection](#) [Edit](#) [+ New](#) [Learn more](#)

File path \*: raw / population / population\_by\_age.tsv [Browse](#) [Preview data](#)

Compression type: None

Column delimiter: Tab (\t) [Edit](#)

Row delimiter: Default (\r\n, or \n) [Edit](#)

Encoding: Default(UTF-8)

Escape character: Backslash (\) [Edit](#)

Quote character: Double quote (") [Edit](#)

First row as header: ☒

Null value:

## Step 5 - Creating an ingestion pipeline

The screenshot displays the Azure Data Factory pipeline editor for a pipeline named `pl_ingest_populati...`. The interface includes a left-hand sidebar with an **Activities** pane, a top toolbar with **Validate**, **Debug**, and **Add trigger** options, and a bottom pane for activity configuration.

The pipeline workflow consists of the following steps:

- Validation**: A **Check if file exists** activity.
- Get Metadata**: A **Get File Metadata** activity.
- If Condition**: A conditional execution block titled **If Column Count Matches**.
  - True**: A sequence of **Copy population...** followed by **Delete source file**.
  - False**: A **Send Email** activity.

The bottom pane shows the configuration for the selected **If Condition** activity:

- Name \***: `If Column Count Matches` (with a [Learn more](#) link).
- Description**: An empty text box.

## Step 6 - Trigger (ingestion) pipeline

### Edit trigger

**Name \***

**Description**

**Type \***

BlobEventsTrigger

**Account selection method \*** ⓘ

☒ From Azure subscription ☐ Enter manually

**Azure subscription** ⓘ

Eigen abonnement (9b9a9691-6d3d-43d5-84a2-bddc02aa3789) ▾

**Storage account name \*** ⓘ

covidreportingsa0202 ▾ ↻

**Container name \*** ⓘ

/population/ ▾

**Blob path begins with** ⓘ

population\_by\_age.tsv.gz

**Blob path ends with** ⓘ

**Event \*** ⓘ

☒ Blob created ☐ Blob deleted

**Ignore empty blobs \*** ⓘ

☒ Yes ☐ No

**Annotations**

+ New

**Status** ⓘ

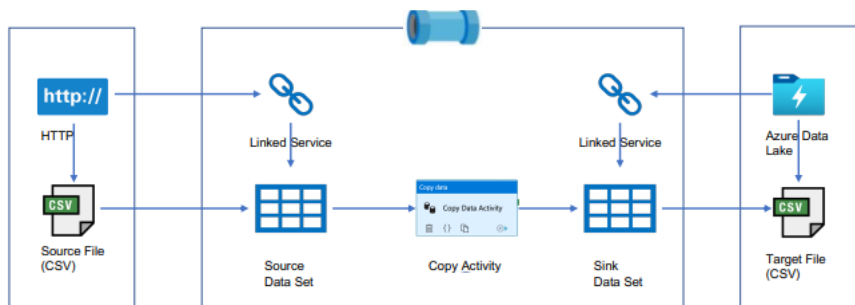
☐ Started ☒ Stopped

Continue

Cancel

## 3.2 Data ingestion from HTTP

### Step 0 - Overview copy activity from HTTP



### Step 1 - Created linked services

#### Edit linked service

HTTP [Learn more](#)

Name \*

ls\_http\_opendata\_ecdc\_europa\_eu

Description

Connect via integration runtime \*

AutoResolveIntegrationRuntime

Base URL \*

@{linkedService().sourceBaseUrl}

Information will be sent to the URL specified. Please ensure you trust the URL entered.

Server Certificate Validation

☒ Enable ☐ Disable

Authentication type \*

Anonymous

Auth headers

+ New

Annotations

+ New

Parameters

+ New | Delete

<input type="checkbox"/> Name	Type	Default value
<input type="checkbox"/> sourceBaseUrl	String	Value

> Advanced

## Step 1 - Create data set (http - source)

The screenshot displays the Microsoft Fabric Data Factory interface. On the left, the 'Factory Resources' pane shows a tree view with 'Pipelines' (9 items) and 'Datasets' (16 items). The 'raw' folder under 'Datasets' is expanded, showing several datasets, including 'ds\_ecdc\_raw\_csv\_http' which is selected.

The main workspace shows a preview of the selected dataset, 'ds\_ecdc\_raw\_csv\_http', with a 'DelimitedText' icon and a 'CSV' file icon.

Below the preview, the 'Connection' tab is active, showing the configuration for the dataset. The 'Linked service' is set to 'ls\_http\_opendata\_ecdc\_europa\_eu'. The 'Linked service properties' table is shown below:

Name	Value	Type
sourceBaseUrl	@dataset().baseUrl	string

The configuration settings for the dataset are as follows:

- Relative URL: @dataset().relativeURL
- Compression type: None
- Column delimiter: Comma (,)
- Row delimiter: Default (\r, \n, or \r\n)
- Encoding: Default(UTF-8)
- Escape character: Backslash (\)
- Quote character: Double quote (")
- First row as header: ☒
- Null value: (empty)

## Step 2 - Create data set (DL sink)

The screenshot displays the Azure Data Factory (ADF) console. On the left, the 'Factory Resources' pane shows a tree view with 'Pipelines' (9 items) and 'Datasets' (16 items). The 'raw' folder under 'Datasets' is expanded, and 'ds\_ecdc\_raw\_csv\_dl' is selected. The main workspace shows a preview of the dataset as a 'DelimitedText' file named 'ds\_ecdc\_raw\_csv\_dl'. Below this, the 'Connection' tab is active, showing the configuration for the dataset's link service.

**Connection Configuration:**

- Linked service:** ls\_adls\_covidreportingdl
- File path:** raw / ecdc / @dataset().fileName
- Compression type:** None
- Column delimiter:** Comma (,)
- Row delimiter:** Default (\r\n, or \n)
- Encoding:** Default(UTF-8)
- Escape character:** Backslash (\)
- Quote character:** Double quote (")
- First row as header:** ☒
- Null value:**

### Step 3 - Create pipeline for data ingestion

The screenshot displays the Azure Data Factory (ADF) interface for creating a pipeline. The top navigation bar includes 'Data Factory', 'Validate all', 'Publish all', and a 'Preview experience' toggle set to 'Off'. The left sidebar shows the 'Factory Resources' tree with categories like Pipelines, Datasets, and Data flows. The 'pl\_ingest\_ecdc\_data' pipeline is selected. The main canvas shows the pipeline logic: a 'Lookup' activity named 'Lookup ECDC File List' is connected to a 'ForEach' loop. The 'ForEach' loop is configured to 'Execute copy for every record' and contains a 'Copy ECDC Data' activity. The bottom panel shows the 'General' tab for the selected activity, with fields for 'Name' (set to 'Execute copy for every record') and 'Description'.

**Factory Resources**

- Pipelines (9)
  - Execute (1)
  - Ingest (2)
    - pl\_ingest\_ecdc\_data
    - pl\_ingest\_population\_data
  - Process (3)
  - sqlize (3)
- Datasets (16)
  - lookup (3)
  - processed (4)
  - raw (6)
    - ds\_ecdc\_raw\_csv\_dl
    - ds\_ecdc\_raw\_csv\_http
    - ds\_population\_raw\_gz
    - ds\_population\_raw\_tsv
    - ds\_raw\_cases\_and\_deaths
    - ds\_raw\_hospital\_admissions
  - sql (3)
- Data flows (2)
- Power Query (0)

**Activities**

- Move & transform
- Azure Data Explorer
- Azure Function
- Batch Service
- Databricks
- Data Lake Analytics
- General
- HDInsight
- Iteration & conditionals
- Machine Learning
- Power Query

**Lookup**

- Lookup ECDC File List

**ForEach**

- Execute copy for every record
- Copy ECDC Data

**General** Settings Activities (1) User properties

Name \* Execute copy for every record [Learn more](#)

Description

## Step 4 - Create trigger for pipeline

### Edit trigger

Name \*

tr\_ingest\_ecdc\_data

Description

Type \*

TumblingWindowTrigger

Start Date (UTC) \*

12/19/22 00:00:00

Recurrence \* ⓘ

Every 24

Hour(s)

☐ Specify an end date

> Advanced

Annotations

+ New

Status ⓘ

☒ Started ☐ Stopped

## Step 6 - Ecdc file list data set

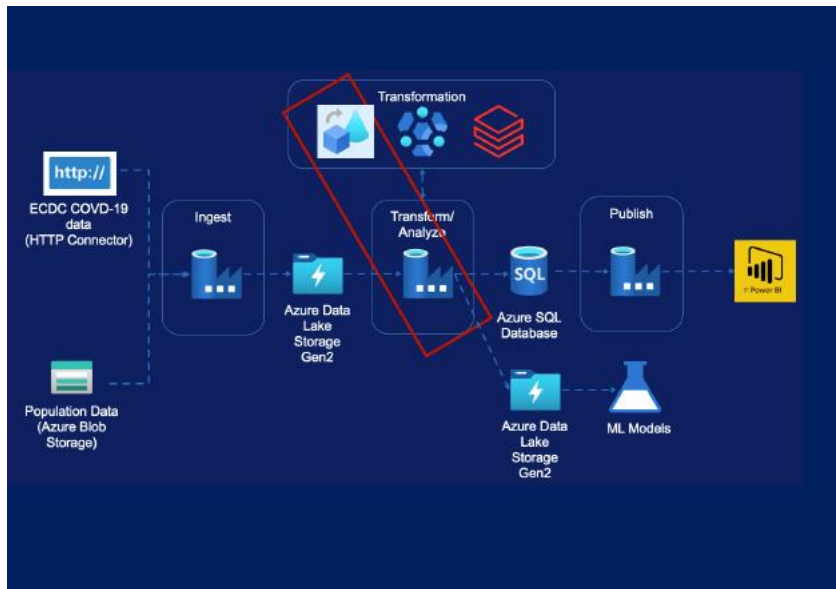
The screenshot displays the Azure Data Factory (ADF) console. On the left, the 'Factory Resources' pane shows a tree view with categories: Pipelines (9), Datasets (16), lookup (3), processed (4), raw (6), sql (3), and Data flows (2). The 'ds\_ecdc\_file\_list' dataset is selected under the 'lookup' category. The main pane shows the configuration for this dataset, which is a JSON file. The 'Connection' tab is active, showing the 'Linked service' as 'ls\_ablob\_covidreportingsa'. The 'File path' is configured as 'configs / Directory / ecdc\_file\_list.json'. The 'Compression type' is set to 'None' and the 'Encoding' is 'Default(UTF-8)'. The 'Preview experience' toggle is turned off.



## 4. Data transformation

### 4.1 Data flow

Step 0 – Overview transformation in data flow



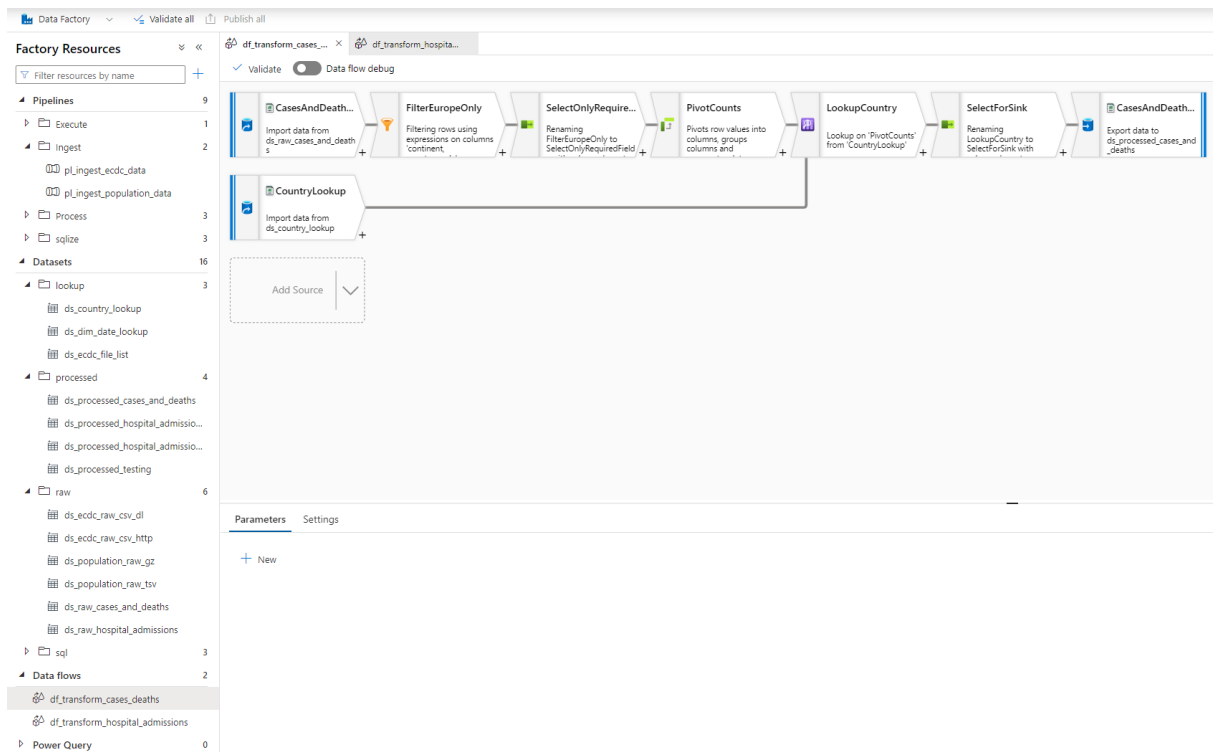
(1) Steps taken:

- Source Transformation
- Filter Transformation
- Select Transformation
- Pivot Transformation
- Lookup Transformation
- Sink Transformation
- Create Pipeline

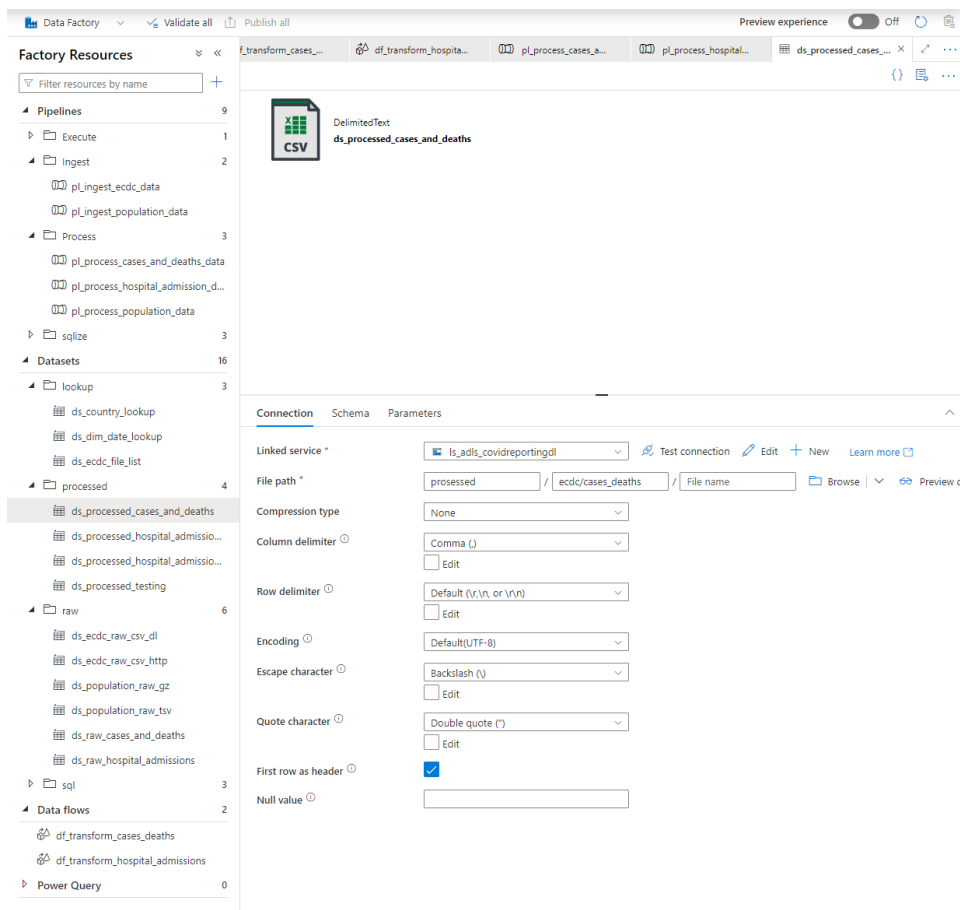
(2) Steps taken:

- Source Transformation
- Select Transformation
- Lookup Transformation
- Pivot Transformation
- Sink Transformation
- Conditional Split Transformation
- Derived Column Transformation
- Aggregate Transformation
- Sort Transformation
- Join Transformation
- Select Transformation
- Create Pipeline

## (1) Step 1 - Create transformations with data flow



## (1) Step 2 - Create data set for processed data



## (1) Step 3 – Create pipeline

The screenshot displays the Azure Data Factory (ADF) interface. On the left, the 'Factory Resources' pane shows a hierarchical view of resources, including Pipelines, Ingest, Process, Datasets, and Power Query. The 'Process' folder is expanded, showing several data flows. The 'Activities' pane in the center lists various activities such as Move & transform, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, Machine Learning, and Power Query. The right pane shows the configuration for a 'Data flow' activity, with tabs for General, Settings, Parameters, and User properties. The 'Settings' tab is active, showing options for 'Data flow', 'Run on (Azure IR)', 'Compute size', 'Logging level', 'Sink properties', and 'Staging'.

**Factory Resources**

- Pipelines: 9
  - Execute: 1
  - Ingest: 2
    - pl\_ingest\_ecdc\_data
    - pl\_ingest\_population\_data
  - Process: 3
    - pl\_process\_cases\_and\_deaths\_data
    - pl\_process\_hospital\_admission\_d...
    - pl\_process\_population\_data
  - sqlize: 3
- Datasets: 16
  - Lookup: 3
    - ds\_country\_lookup
    - ds\_dim\_date\_lookup
    - ds\_ecdc\_file\_list
  - processed: 4
    - ds\_processed\_cases\_and\_deaths
    - ds\_processed\_hospital\_admissio...
    - ds\_processed\_hospital\_admissio...
    - ds\_processed\_testing
  - raw: 6
    - ds\_ecdc\_raw\_csv\_dl
    - ds\_ecdc\_raw\_csv\_http
    - ds\_population\_raw\_gz
    - ds\_population\_raw\_tsv
    - ds\_raw\_cases\_and\_deaths
    - ds\_raw\_hospital\_admissions
  - sql: 3
- Data flows: 2
  - df\_transform\_cases\_deaths
  - df\_transform\_hospital\_admissions
- Power Query: 0

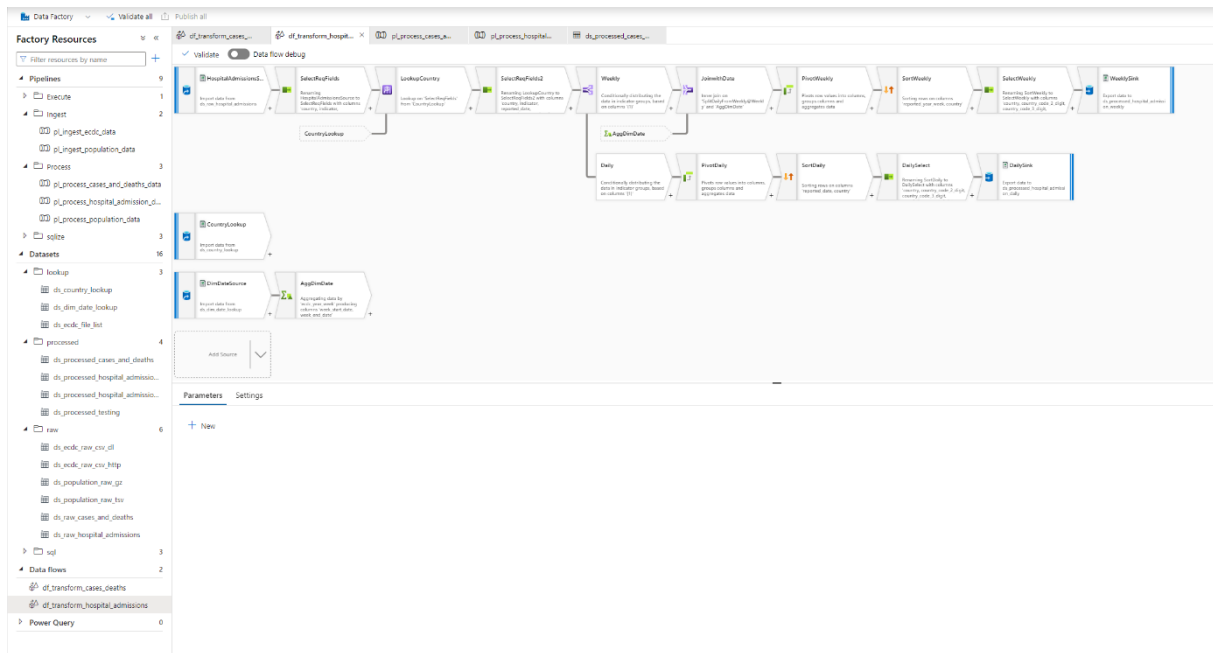
**Activities**

- Move & transform
- Azure Data Explorer
- Azure Function
- Batch Service
- Databricks
- Data Lake Analytics
- General
- HDInsight
- Iteration & conditionals
- Machine Learning
- Power Query

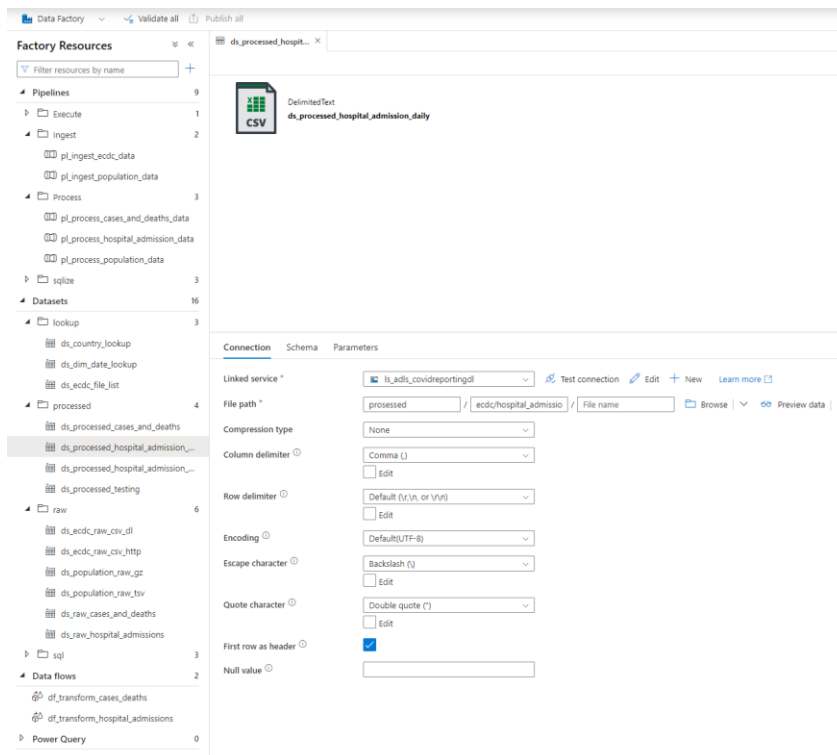
**Data flow configuration**

- General: Data flow \* (df\_transform\_cases\_deaths), Run on (Azure IR) \* (AutoResolveIntegrationRuntime), Compute size \* (Small)
- Advanced: Logging level \* (Verbose, Basic, None)
- Sink properties
- Staging

## (2) Step 4 - Create transformations with data flow



## Step 5 - Create data set for processed data (daily)



## Step 6 - Create data set for processed data (weekly)

The screenshot displays the Azure Data Factory (ADF) console. On the left, the 'Factory Resources' tree shows the hierarchy: Pipelines (9), Ingest (2), Process (3), sqsize (3), Datasets (16), Lookup (3), processed (4), raw (6), sql (3), Data flows (2), and Power Query (0). The 'processed' folder is expanded, showing datasets like 'ds\_processed\_cases\_and\_deaths', 'ds\_processed\_hospital\_admission...', and 'ds\_processed\_testing'. The 'ds\_processed\_hospital\_admission...' dataset is selected, and its configuration is shown on the right.

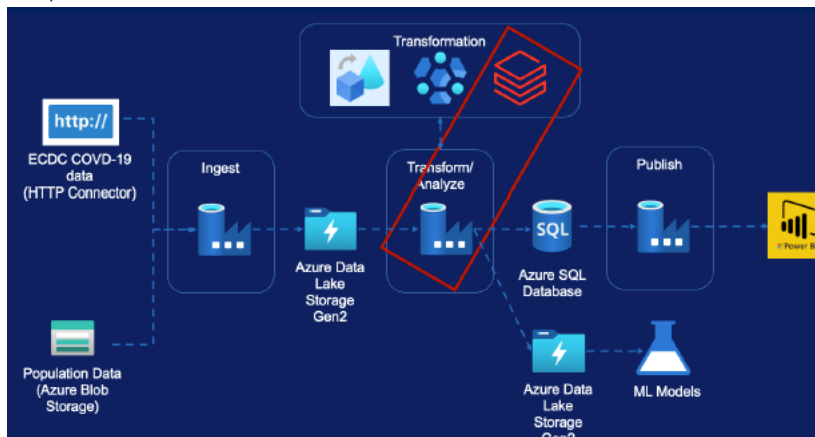
The configuration pane for the dataset 'ds\_processed\_hospital\_admission\_weekly' is visible, showing the 'Connection' tab. The 'Linked service' is set to 'ls\_adls\_covidreportingdl'. The 'File path' is configured as 'processed / ecdc/hospital\_admissio / File name'. The 'Compression type' is 'None'. The 'Column delimiter' is 'Comma (,)' with an 'Edit' link. The 'Row delimiter' is 'Default (\r\n, or \n)' with an 'Edit' link. The 'Encoding' is 'Default(UTF-8)'. The 'Escape character' is 'Backslash (\)' with an 'Edit' link. The 'Quote character' is 'Double quote (")' with an 'Edit' link. The 'First row as header' checkbox is checked. The 'Null value' field is empty.

## Step 7 - Create pipeline



## 4.2 Databricks

### Step 0 - Databricks overview



Steps taken:

- Create Databricks Service
- Create Databricks Cluster
- Mount Storage Accounts
- Transformation requirements
- Creating Pipeline

### Step 1 – Mount cluster 1 / 2

mount\_storage Python

File Edit View Run Help Last edit was 22 days ago Give feedback

Run all Unknown Schedule Share

Cmd 1

#### Mount the following data lake storage gen2 containers

1. raw
2. processed
3. lookup

Cmd 2

#### Set-up the configs

Please update the following

- application-id
- service-credential
- directory-id

Cmd 3

```
1 configs = {"fs.azure.account.auth.type": "OAuth",
2           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
3           "fs.azure.account.oauth2.client.id": "47846408-5bca-44f4-9de3-71fd3b08a9d6",
4           "fs.azure.account.oauth2.client.secret": "rty8Q-FbCdPpT37mAtANTCisANbkYI.XyOYMasI",
5           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/8603b7a6-42b5-4a50-a399-b1332d779cb6/oauth2/token"}
```

Command took 0.04 seconds -- by ossy-80@live.nl at 12/14/2022, 3:47:26 PM on Covid-reporting-cluster

Cmd 4

#### Mount the raw container

Update the storage account name before executing

Cmd 5

```
1 dbutils.fs.mount(
2   source = "abfss://raw@covidreportingdl020.dfs.core.windows.net/",
3   mount_point = "/mnt/covidreportingdl020/raw",
4   extra_configs = configs)
```

▶ (1) Spark Jobs

Out[4]: True

Command took 24.09 seconds -- by ossy-80@live.nl at 12/14/2022, 3:48:47 PM on Covid-reporting-cluster

Cmd 6

#### Mount the processed container

Update the storage account name before executing

Cmd 7

```
1 dbutils.fs.mount(
2   source = "abfss://processed@covidreportingdl020.dfs.core.windows.net/",
3   mount_point = "/mnt/covidreportingdl020/processed",
```

## Step 1 – Mount cluster 2 / 2

**mount\_storage** Python

File Edit View Run Help [Last edit was 22 days ago](#) [Give feedback](#)

▶ Run all ● Unknown ▼  Schedule Share

```
4 extra_configs = configs)
```

▶ (1) Spark Jobs

Out[8]: True

Command took 20.83 seconds -- by ossy-80@live.nl at 12/14/2022, 3:53:14 PM on Covid-reporting-cluster

Cmd 8

### Mount the lookup container

#### Update the storage account name before executing

Cmd 9

```
1 dbutils.fs.mount(
2     source = "abfss://lookup@covidreportingdl020.dfs.core.windows.net/",
3     mount_point = "/mnt/covidreportingdl020/lookup",
4     extra_configs = configs)
```

▶ (1) Spark Jobs

Out[6]: True

Command took 21.14 seconds -- by ossy-80@live.nl at 12/14/2022, 3:50:45 PM on Covid-reporting-cluster

Cmd 10

```
1 dbutils.fs.ls("/mnt/covidreportingdl020/lookup")
```

Out[9]: [FileInfo(path='dbfs:/mnt/covidreportingdl020/lookup/dim\_country/', name='dim\_country/', size=0),  
FileInfo(path='dbfs:/mnt/covidreportingdl020/lookup/dim\_date/', name='dim\_date/', size=0)]

Command took 0.26 seconds -- by ossy-80@live.nl at 12/14/2022, 3:54:21 PM on Covid-reporting-cluster

Cmd 11

```
1
```

Shift+Enter to run



## Step 2 – transform data 1 / 2

**transform\_population\_data** Python  
File Edit View Run Help Last edit was 22 days ago Give feedback

Run all Connect Schedule Share

Cmd 3

1 from pyspark.sql.functions import \*

Cmd 4

**Read the population data & create a temp view**

Cmd 5

1 df\_raw\_population = spark.read.csv("/mnt/covidreportingdl020/raw/population", sep=r'\t', header=True)  
2 df\_raw\_population = df\_raw\_population.withColumn('age\_group', regexp\_replace(split(df\_raw\_population['indic\_de,geo\time'], ',')[0], 'PC\_', ''))  
3 df\_raw\_population = df\_raw\_population.withColumn('country\_code', split(df\_raw\_population['indic\_de,geo\time'], ',')[1])  
4 df\_raw\_population = df\_raw\_population.select(col("country\_code").alias("country\_code"),  
5 col("age\_group").alias("age\_group"),  
6 col("2019 ").alias("percentage\_2019"))  
df\_raw\_population.createOrReplaceTempView("raw\_population")

Cmd 6

**Pivot the data by age group**

Cmd 7

1 # Create a data frame with pivoted percentages  
2 df\_raw\_population\_pivot = spark.sql("SELECT country\_code, age\_group, cast(regexp\_replace(percentage\_2019, '[a-z]', '') AS decimal(4,2)) AS  
percentage\_2019 FROM raw\_population WHERE length(country\_code) =  
2").groupBy("country\_code").pivot("age\_group").sum("percentage\_2019").orderBy("country\_code")  
3 df\_raw\_population\_pivot.createOrReplaceTempView("raw\_population\_pivot")

Cmd 8

**Read the country lookup**

Cmd 9

1 # Create a data frame for the country lookup  
2 df\_dim\_country = spark.read.csv("/mnt/covidreportingdl020/lookup/dim\_country", sep=r',', header=True)  
3 df\_dim\_country.createOrReplaceTempView("dim\_country")

Cmd 10

**Join population data with country lookup**

Cmd 11

1 df\_processed\_population = spark.sql("""SELECT c.country,  
2 c.country\_code\_2\_digit,  
3 c.country\_code\_3\_digit,  
4 c.population,  
5 p.Y0\_14 AS age\_group\_0\_14,  
6 p.Y15\_24 AS age\_group\_15\_24,  
7 p.Y25\_49 AS age\_group\_25\_49,  
8 p.Y50\_64 AS age\_group\_50\_64,  
9 p.Y65\_79 AS age\_group\_65\_79,  
10 p.Y80\_MAX AS age\_group\_80\_max  
11 FROM raw\_population\_pivot p  
12 JOIN dim\_country c ON c.country\_code\_2\_digit = p.country\_code\_2\_digit  
13 AND c.country\_code\_3\_digit = p.country\_code\_3\_digit""")

## Step 2 – transform data 2 / 2

**transform\_population\_data** Python

File Edit View Run Help Last edit was 22 days ago Give feedback

```
5      p.Y0_14 AS age_group_0_14,  
6      p.Y15_24 AS age_group_15_24,  
7      p.Y25_49 AS age_group_25_49,  
8      p.Y50_64 AS age_group_50_64,  
9      p.Y65_79 AS age_group_65_79,  
10     p.Y80_MAX AS age_group_80_max  
11     FROM raw_population_pivot p  
12     JOIN dim_country c ON p.country_code = country_code_2_digit  
13     ORDER BY country""")
```

Cmd 12

**Write output to the processed mount point**

Cmd 13

```
1 df_processed_population.write.format("com.databricks.spark.csv").option("header","true").option("delimiter",  
",").mode("overwrite").save("/mnt/covidreportingdl020/prosessed/population")
```

Cmd 14

```
1
```

Shift+Enter to run

## Step 3 – Create linked service

**Edit linked service**  
Azure Databricks Learn more

**Name \***  
ls\_db\_covid\_cluster

**Description**

**Connect via integration runtime \***  
AutoResolveIntegrationRuntime

**Account selection method \***  
☐ From Azure subscription ☒ Enter manually

**Databricks Workspace URL \***  
https://adb-3875266610093583.3.azuredatabricks.net

**Authentication type \***  
Access Token

Access token

Azure Key Vault

**Access token \***  
\*\*\*\*\*

**Select cluster**  
☒ New job cluster ☐ Existing interactive cluster ☐ Existing instance pool

**Cluster version \***  
7.3.x-cpu-ml-scala2.12

**Cluster node type \***  
Standard\_DS3\_v2

**Python Version \***  
3

**Worker options**  
☒ Fixed ☐ Autoscaling

**Workers \***  
1

> Additional cluster settings

Annotations

+ New

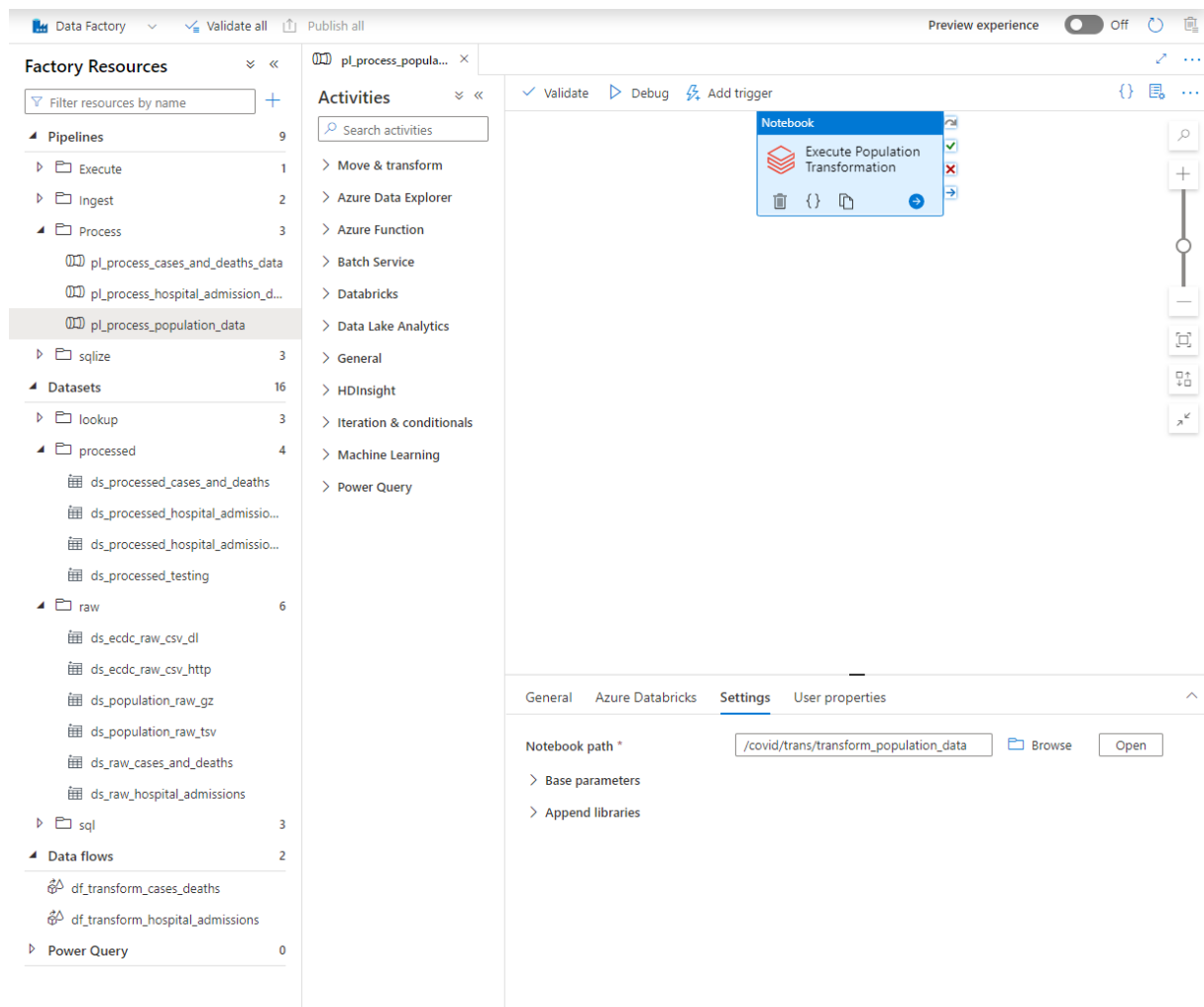
> Parameters

> Advanced

Save Cancel

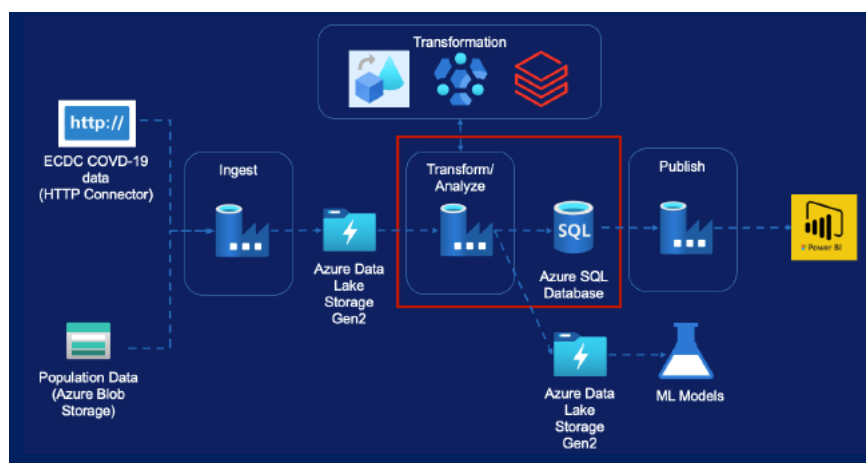
Test connection

## Step 4 – Create pipeline



## 5. Load into database

### Step 0 – Overview SQL database



Steps taken:



- Copy Cases & Deaths data
- Copy Hospital Admissions data
- Copy testing data

Step 1 – Create SQL script

```
1 CREATE SCHEMA covid_reporting
2 GO
3
4 CREATE TABLE covid_reporting.cases_and_deaths
5 (
6     country          VARCHAR(100),
7     country_code_2_digit VARCHAR(2),
8     country_code_3_digit VARCHAR(3),
9     population        BIGINT,
10    cases_count        BIGINT,
11    deaths_count        BIGINT,
12    reported_date       DATE,
13    source              VARCHAR(500)
14 )
15 GO
16
17 CREATE TABLE covid_reporting.hospital_admissions_daily
18 (
19     country          VARCHAR(100),
20     country_code_2_digit VARCHAR(2),
21     country_code_3_digit VARCHAR(3),
22     population        BIGINT,
23     reported_date       DATE,
24     hospital_occupancy_count BIGINT,
25     icu_occupancy_count  BIGINT,
26     source              VARCHAR(500)
27 )
28 GO
29
30 CREATE TABLE covid_reporting.testing
31 (
32     country          VARCHAR(100),
33     country_code_2_digit VARCHAR(2),
34     country_code_3_digit VARCHAR(3),
35     year_week         VARCHAR(8),
36     week_start_date    DATE,
37     week_end_date       DATE,
38     new_cases          BIGINT,
39     tests_done          BIGINT,
40     population         BIGINT,
41     testing_data_source VARCHAR(500)
42 )
43 GO
44
```

## Step 2 – Create linked service

### Edit linked service

 Azure SQL Database [Learn more](#) 

Name \*

ls\_sql\_covid\_db

Description

Connect via integration runtime \* 

AutoResolveIntegrationRuntime

Connection string

[Azure Key Vault](#)

Account selection method 

☐ From Azure subscription ☒ Enter manually

Fully qualified domain name \*

covid-db020.database.windows.net

Database name \*

covid-db020

Authentication type \*

SQL authentication

User name \*


admin020

Password

[Azure Key Vault](#)

Password \*

\*\*\*\*\*

Always encrypted 

☐

Additional connection properties

[+ New](#)

Annotations

[+ New](#)

[> Parameters](#)

[> Advanced](#) 

### Step 3 - Create sink dataset

The screenshot shows the Azure Data Factory (ADF) console. On the left, the 'Factory Resources' pane is expanded, showing a tree view of resources. The 'sql' folder under 'raw' is selected, and the 'ds\_sql\_cases\_and\_deaths' dataset is highlighted. The main pane displays the configuration for this dataset, showing it is an 'Azure SQL Database' sink. The 'Connection' tab is active, showing the 'Linked service' as 'ls\_sql\_covid\_db' and the 'Table' as 'covid\_reporting.cases\_and\_deaths'. The 'Preview experience' toggle is set to 'Off'.

**Factory Resources**

- Pipelines: 9
  - Execute: 1
  - Ingest: 2
  - Process: 3
  - sqlize: 3
    - pl\_sqlize\_cases\_and\_deaths\_data
    - pl\_sqlize\_hospital\_admissions\_da...
    - pl\_sqlize\_testing
- Datasets: 16
  - lookup: 3
  - processed: 4
    - ds\_processed\_cases\_and\_deaths
    - ds\_processed\_hospital\_admissio...
    - ds\_processed\_hospital\_admissio...
    - ds\_processed\_testing
  - raw: 6
    - ds\_ecdc\_raw\_csv\_dl
    - ds\_ecdc\_raw\_csv\_http
    - ds\_population\_raw\_gz
    - ds\_population\_raw\_tsv
    - ds\_raw\_cases\_and\_deaths
    - ds\_raw\_hospital\_admissions
  - sql: 3
    - ds\_sql\_cases\_and\_deaths**
    - ds\_sql\_hospital\_admissions\_daily
    - ds\_sql\_testing
- Data flows: 2
  - df\_transform\_cases\_deaths
  - df\_transform\_hospital\_admissions
- Power Query: 0

**ds\_sql\_cases\_and\_deaths**

Azure SQL Database

**Connection** Schema Parameters

Linked service \* **ls\_sql\_covid\_db** Test connection Edit + New Learn more

Table **covid\_reporting.cases\_and\_deaths** Refresh Preview data

Edit

## Step 4 – Create pipeline

The screenshot displays the Azure Data Factory (ADF) interface. The top navigation bar includes 'Data Factory', 'Validate all', 'Publish all', and a 'Preview experience' toggle set to 'Off'. The left sidebar, titled 'Factory Resources', lists various components: Pipelines (9), Datasets (16), and Data flows (2). The 'Pipelines' section is expanded, showing a list of pipelines including 'pl\_sqlize\_cases\_and\_deaths\_data'. The 'Activities' pane on the right lists various activity types such as 'Move & transform', 'Azure Data Explorer', 'Azure Function', 'Batch Service', 'Databricks', 'Data Lake Analytics', 'General', 'HDInsight', 'Iteration & conditionals', 'Machine Learning', and 'Power Query'. The main canvas shows a pipeline named 'pl\_sqlize\_cases\_and\_deaths\_data' with a single activity, 'Copy Cases and Deaths', which is a 'Copy data' activity. The 'Parameters' tab is selected at the bottom, showing a '+ New' button.

## Step 5 – Create pipeline hospital admission data

The screenshot displays the Azure Data Factory (ADF) console. The top navigation bar includes 'Data Factory', 'Validate all', 'Publish all', and a 'Preview experience' toggle set to 'Off'. The left sidebar, titled 'Factory Resources', shows a tree view of the workspace. Under 'Pipelines', the pipeline 'pl\_sqlize\_hospital\_admissions\_da...' is selected. The 'Activities' pane on the right lists various activity types, with 'Copy data' being the active selection. The main canvas shows a single activity named 'Copy Hospital Admissions Data' with a green checkmark, indicating it is configured. The bottom pane is currently set to 'Parameters' and shows a '+ New' button to add new parameters. The right edge of the canvas features a vertical toolbar with icons for zooming and other view controls.



## Step 6 - Create pipeline testing data

The screenshot displays the Azure Data Factory (ADF) interface. The top navigation bar includes 'Data Factory', 'Validate all', 'Publish all', and a 'Preview experience' toggle set to 'Off'. The left sidebar, titled 'Factory Resources', shows a tree view of resources. Under 'Pipelines', 'pl\_sqlize\_testing' is selected. The 'Activities' pane on the right lists various activity categories, with 'Copy data' selected. The main canvas shows a single 'Copy testing' activity. The bottom pane is titled 'Parameters' and contains a '+ New' button.

**Factory Resources**

- Pipelines (9)
  - Execute (1)
  - Ingest (2)
  - Process (3)
  - sqlize (3)
    - pl\_sqlize\_cases\_and\_deaths\_data
    - pl\_sqlize\_hospital\_admissions\_da...
    - pl\_sqlize\_testing
- Datasets (16)
  - lookup (3)
  - processed (4)
    - ds\_processed\_cases\_and\_deaths
    - ds\_processed\_hospital\_admissio...
    - ds\_processed\_hospital\_admissio...
    - ds\_processed\_testing
  - raw (6)
    - ds\_ecdc\_raw\_csv\_dl
    - ds\_ecdc\_raw\_csv\_http
    - ds\_population\_raw\_gz
    - ds\_population\_raw\_tsv
    - ds\_raw\_cases\_and\_deaths
    - ds\_raw\_hospital\_admissions
  - sql (3)
    - ds\_sql\_cases\_and\_deaths
    - ds\_sql\_hospital\_admissions\_daily
    - ds\_sql\_testing
- Data flows (2)
  - df\_transform\_cases\_deaths
  - df\_transform\_hospital\_admissions
- Power Query (0)

**Activities**

- Move & transform
- Azure Data Explorer
- Azure Function
- Batch Service
- Databricks
- Data Lake Analytics
- General
- HDInsight
- Iteration & conditionals
- Machine Learning
- Power Query

**Copy data**

- Copy testing

**Parameters**

- + New

## 6. Data orchestration – Making pipelines production ready

### Step 1 – Build pipeline

The screenshot displays the Azure Data Factory (ADF) interface. The top navigation bar includes 'Data Factory', 'Validate all', 'Publish all', and a 'Preview experience' toggle set to 'Off'. The left sidebar, titled 'Factory Resources', shows a tree view with 'Pipelines' (9 items) and 'Datasets' (16 items). Under 'Pipelines', the 'Execute' folder is expanded, showing 'pl\_execute\_population\_pipelines' (1 item). The main canvas, titled 'Activities', shows a pipeline named 'pl\_execute\_population\_pipelines' with two activities: 'Execute Ingest Population Data' and 'Execute Process Population Data'. The pipeline is in a 'Trigger (1)' state. The bottom panel shows the 'Parameters' tab with a '+ New' button.

## Step 2 – Create trigger

### Edit trigger

Name \*

tr\_population\_data\_arrived

Description

Type \*

BlobEventsTrigger

Account selection method \* ⓘ

☒ From Azure subscription ☐ Enter manually

Azure subscription ⓘ

Eigen abonnement (9b9a9691-6d3d-43d5-84a2-bddc02aa3789) ▾

Storage account name \* ⓘ

covidreportingsa0202 ▾

Container name \* ⓘ

population ▾

Blob path begins with ⓘ

population\_by\_age.tsv.gz

Blob path ends with ⓘ

Event \* ⓘ

☒ Blob created ☐ Blob deleted

Ignore empty blobs \* ⓘ

☒ Yes ☐ No

Annotations

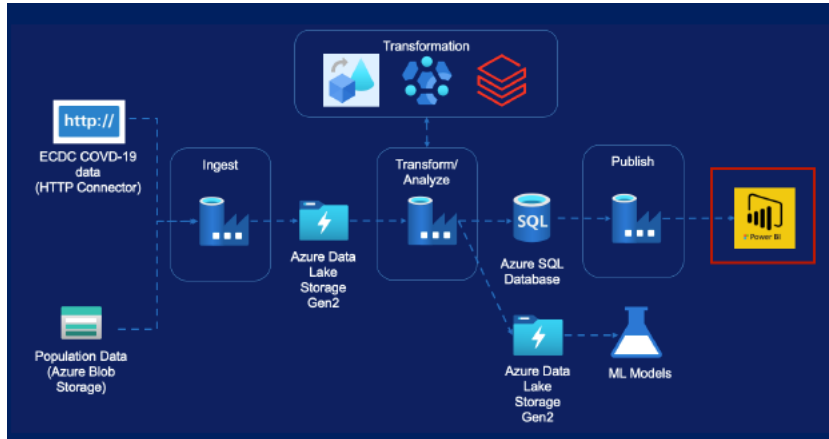
+ New

Status ⓘ

☐ Started ☒ Stopped

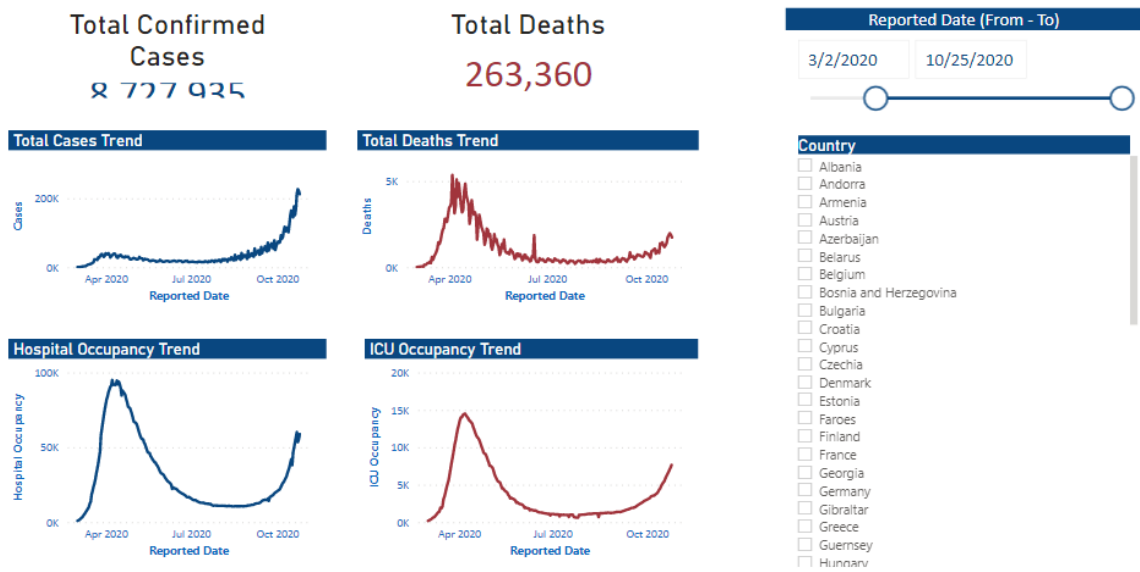
## 7. Data reporting

### Step 0 – Reporting overview in Power BI



### Step 1 – Create report 1 / 2

## Covid-19 Cases EU/EEA & UK



Step 1 – Create report 2 / 2

## Covid-19 Cases UK, France & Germany

Total Cases Trend - United Kingdom



Total Deaths Trend - United Kingdom



Reported Date (From - To)

8/24/2020

10/25/2020

Total Cases Trend - France



Total Deaths Trend - France



Total Cases Trend - Germany

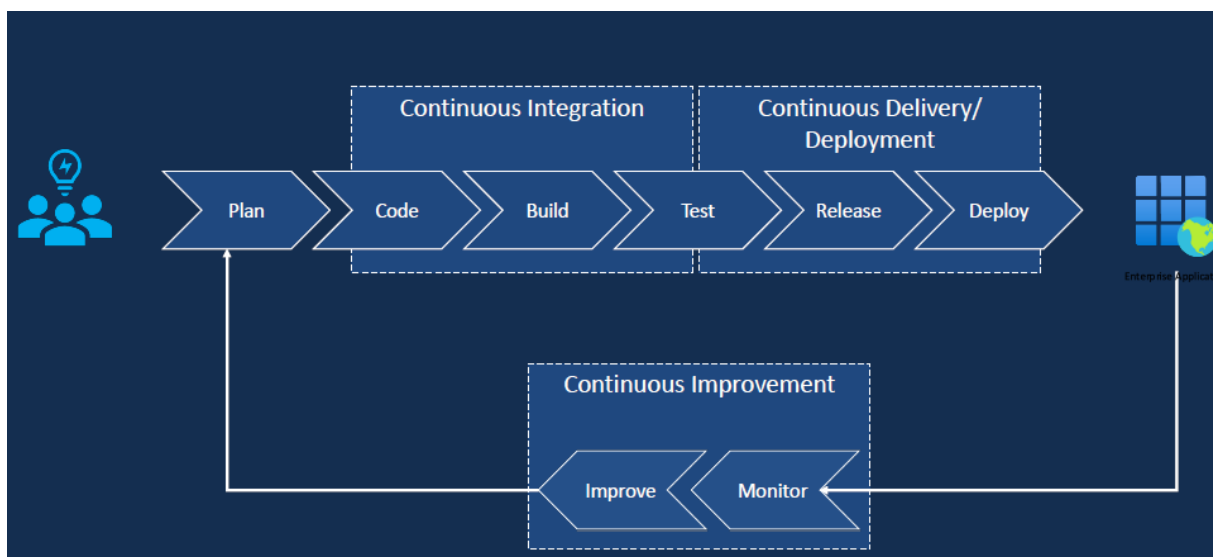
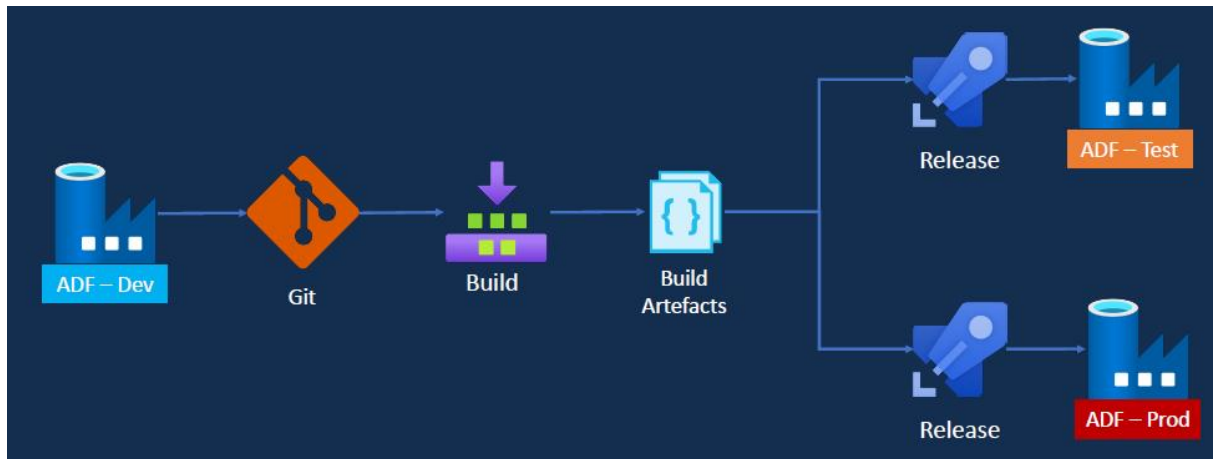


Total Deaths Trend - Germany

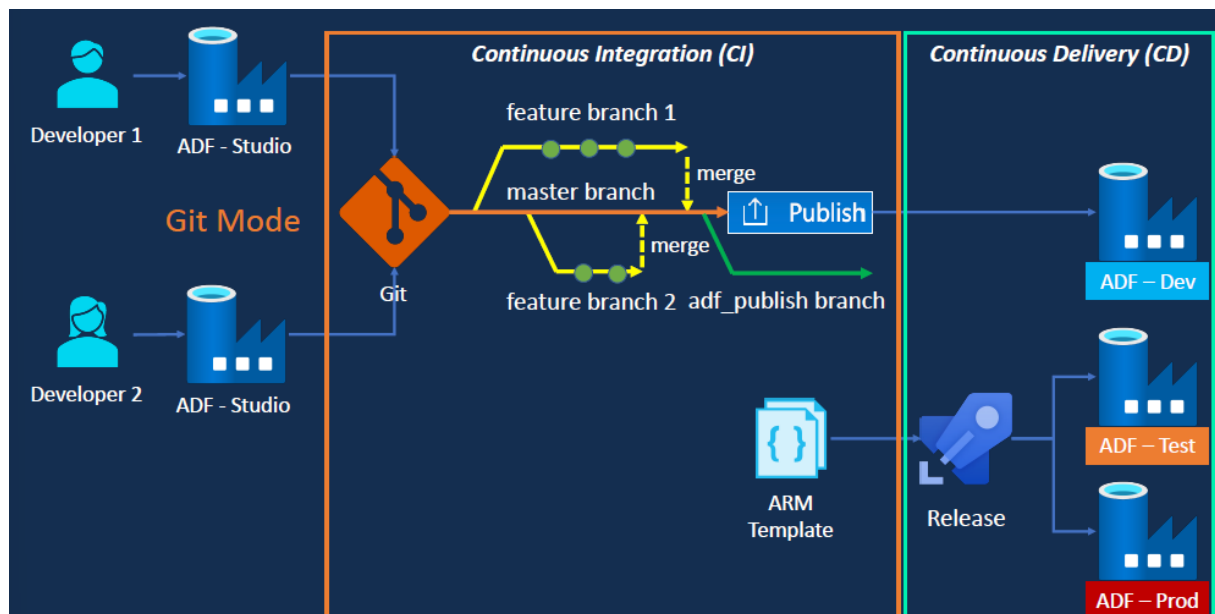


## 8. CI / CD

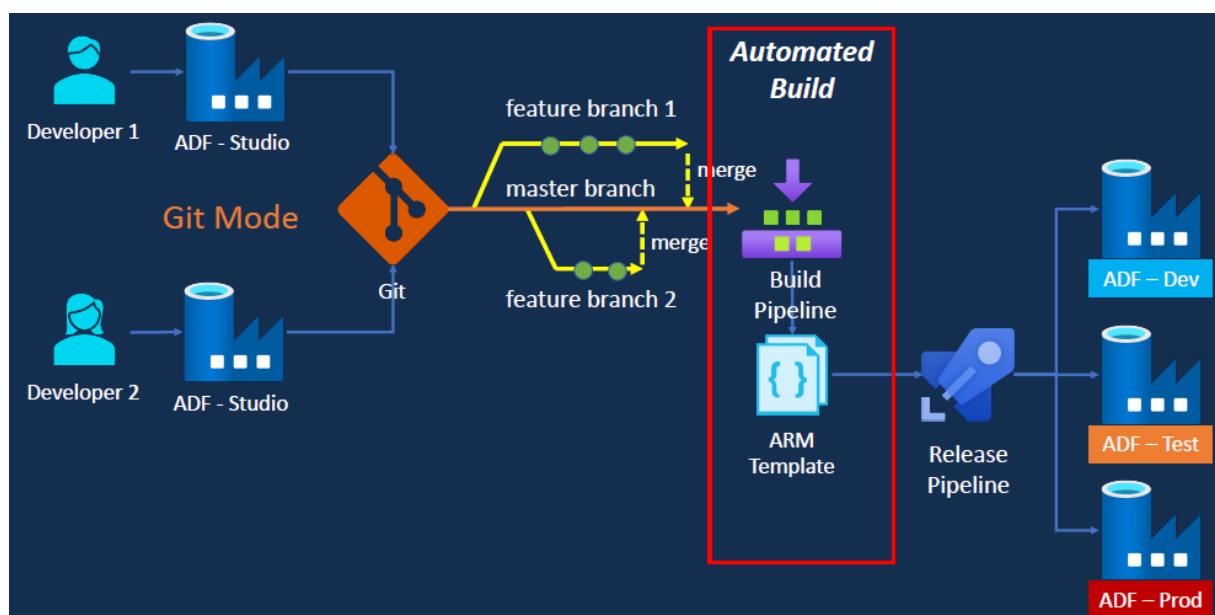
### Step 0 – CI / CD overview



## Step 0 – Git configuration within DevOps option 1



## Step 0 – Git configuration within DevOps option 2



## Step 1 – Create git repo

Azure Data Factory CI CD / Repos / Files / dev-ci-cd-adf-020

Search

BS

dev-ci-cd-adf-020

> build

> dataset

> linkedService

> pipeline

> release

> trigger

MI README.md

main

Type to find a file or folder...

Files

succeeded

Clone

Contents

History

Name ↑	Last change	Commits
build	Dec 27, 2022	4e4a2822 Added 2 files t...
dataset	Yesterday	593bfe03 Adding pipelin...
linkedService	Yesterday	93854955 Adding linkedS...
pipeline	Yesterday	593bfe03 Adding pipelin...
release	Dec 27, 2022	5a9e69e2 Added PrePost...
trigger	Dec 27, 2022	d8afe45f Updating trigg...
MI README.md	Dec 21, 2022	56ab35af Added READM...

### Introduction

TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

### Getting Started

TODO: Guide users through getting your code up and running on their own system. In this section you can talk about:

1. Installation process
2. Software dependencies
3. Latest releases
4. API references

### Build and Test

TODO: Describe and show how to build your code and run the tests.

### Contribute

TODO: Explain how other users and developers can contribute to make your code better.

If you want to learn more about creating good readme files then refer the following [guidelines](#). You can also seek inspiration from the below readme files:

- [ASP.NET Core](#)
- [Visual Studio Code](#)
- [Chakra Core](#)



## Step 2 – Create Tools for azure (DEV)

**dev-ci-cd-rg** Resource group

Search

+ Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags Move

**Overview**

- Activity log
- Access control (IAM)
- Tags
- Resource visualizer
- Events

**Settings**

- Deployments
- Security
- Policies
- Properties
- Locks

**Cost Management**

- Cost analysis
- Cost alerts (preview)

**Essentials**

Subscription (move) [Eigen abbonement](#)

Subscription ID: 9b9a9691-6d3d-43d5-84a2-bddc02aa3789

Deployments: [3 Succeeded](#)

Location: East US

Tags (edit) [Click here to add tags](#)

**Resources** Recommendations

Filter for any field... Type equals all Location equals all Add filter

Showing 1 to 3 of 3 records. Show hidden types No grouping List view

Name	Type	Location
dev-ci-cd-adf-020	Data factory (V2)	East US
dev-ci-cd-kv-020-1	Key vault	East US
devcicddl	Storage account	East US

## Step 2 – Create Tools for azure (TEST)

**test-ci-cd-rg** Resource group

Search

+ Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags Move

**Overview**

- Activity log
- Access control (IAM)
- Tags
- Resource visualizer
- Events

**Settings**

- Deployments
- Security
- Policies
- Properties
- Locks

**Cost Management**

- Cost analysis

**Essentials**

Subscription (move) [Eigen abbonement](#)

Subscription ID: 9b9a9691-6d3d-43d5-84a2-bddc02aa3789

Deployments: [5 Failed](#), [6 Succeeded](#)

Location: East US

Tags (edit) [Click here to add tags](#)

**Resources** Recommendations

Filter for any field... Type equals all Location equals all Add filter

Showing 1 to 3 of 3 records. Show hidden types No grouping List view

Name	Type	Location
test-ci-cd-adf-020	Data factory (V2)	East US
test-ci-cd-kv-020	Key vault	East US
testcicddl	Storage account	East US

## Step 2 – Create Tools for azure (PROD)

**prod-ci-cd-rg** Resource group

Search

**Essentials**

Subscription ([move](#))  
[Eigen abonnement](#)  
Subscription ID  
9b9a9691-6d3d-43d5-84a2-bddc02aa3789  
Tags ([edit](#))  
[Click here to add tags](#)

Deployments  
[3 Succeeded](#)  
Location  
East US

**Resources** Recommendations

Filter for any field... Type equals **all** X Location equals **all** X Add filter

Showing 1 to 3 of 3 records. ☐ Show hidden types ⓘ No grouping List view

Name ↑↓	Type ↑↓	Location ↑↓	
prod-ci-cd-adf-020	Data factory (V2)	East US	...
prod-ci-cd-kv-020	Key vault	East US	...
prodciiddl	Storage account	East US	...

## Step 3 - Create release pipeline option 1

All pipelines > ADF CD Option 1 Release Pi... Save Create release ...

Pipeline Tasks Variables Retention Options History

**Artifacts | + Add**

- \_dev-ci-cd-adf-020
- \_dev-ci-cd-adf-020\_main
- Schedule not set

**Stages | + Add**

- Test 1 job, 3 tasks
- Prod 1 job, 3 tasks

## Step 4 – Create build pipeline option 2 – 1 / 2

< ADF CI Option 2 Build Pipeline

Variables
Run

---

main

dev-ci-cd-adf-020 / build/adf-ci-option-2-build-pipeline.yml

```

1 # File Name: adf-ci-option-2-build-pipeline.yml
2 # Purpose: Build pipeline for the development Azure Data Factory
3 # ..... Automatically triggered on completion of a pull request and
4 # ..... Validates the Data Factory resources
5 # ..... Generates the ARM template and publishes for consumption via
6 # Build Pipeline Name: ADF CI Option 2 Build Pipeline
7
8 trigger:
9   - main
10
11 pool:
12   - vmImage: ubuntu-latest
13
14 variables:
15   - subscriptionId: '9b9a9691-6d3d-43d5-84a2-bddc02aa3789' # Use your sub
16   - resourceGroup: 'dev-ci-cd-ng' # Use the reso
17   - dataFactory: 'dev-ci-cd-adf-020' # Use your dev
18   - PackageFolder: 'build' # Use the GIT
19   - adfRootFolder: '' # Use the GIT
20
21 steps:
22
23   # Installs Node
24   Settings
25   - task: NodeTool@0
26     inputs:
27       versionSpec: '14.x'
28       displayName: 'Install Node.js'
29
30   # Installs the npm packages saved in your package.json file in the build
31   Settings
32   - task: Npm@1
33     inputs:
34       command: 'install'
35       workingDir: '$(Build.Repository.LocalPath)/$(packageFolder)' #replac
36       verbose: true
37       displayName: 'Install npm packages'
38
39   # Validates all of the Data Factory resources in the repository. You'll
40   Settings
41   - task: Npm@1
42     inputs:
43       command: 'custom'
44       workingDir: '$(Build.Repository.LocalPath)/build' #replace with the
45       customCommand: 'run build validate $(Build.Repository.LocalPath)/$(a
46       displayName: 'Validate Data Factory Resources'
47
48   # Generate the ARM template into the destination folder, which is the sa
49   Settings
50   - task: Npm@1
51     inputs:
52       command: 'custom'
53       workingDir: '$(Build.Repository.LocalPath)/build' #replace with the
54       customCommand: 'run build export $(Build.Repository.LocalPath)/$(adf
55       displayName: 'Generate ARM template'
56
57   # Publish the artifact to be used as a source for a release pipeline.
58   Settings
          
```

Tasks ☰

- .NET Core  
Build, test, package, or publish a dotnet applicatio...
- Android signing  
Sign and align Android APK files
- Ant  
Build with Apache Ant
- App Center distribute  
Distribute app builds to testers and users via Visu...
- App Center test  
Test app packages with Visual Studio App Center
- Archive files  
Compress files into .7z, .tar.gz, or .zip
- ARM template deployment  
Deploy an Azure Resource Manager (ARM) templ...
- Azure App Service deploy  
Deploy to Azure App Service a web, mobile, or AP...
- Azure App Service manage  
Start, stop, restart, slot swap, slot delete, install sit...
- Azure App Service Settings  
Update/Add App settings an Azure Web App for ...
- Azure CLI  
Run Azure CLI commands against an Azure subscri...
- Azure Cloud Service deployment  
Deploy an Azure Cloud Service
- Azure Database for MySQL deployment  
Run your scripts and make changes to your Azure...
- Azure file copy  
Copy files to Azure Blob Storage or virtual machin...
- Azure Function on Kubernetes  
Deploy Azure function to Kubernetes cluster.
- Azure Functions  
Update a function app with .NET, Python, JavaScri...
- Azure Functions for container  
Update a function app with a Docker container

## Step 5 – Create release pipeline option 2

