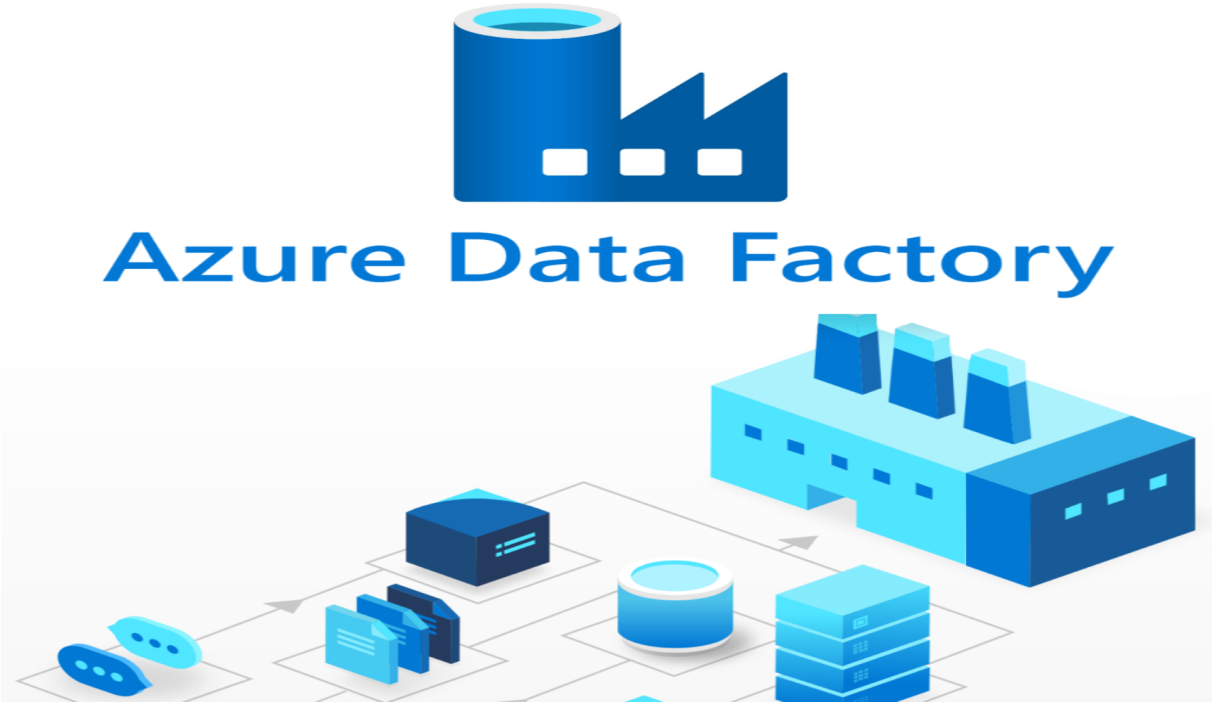
Azure data factory project.



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

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

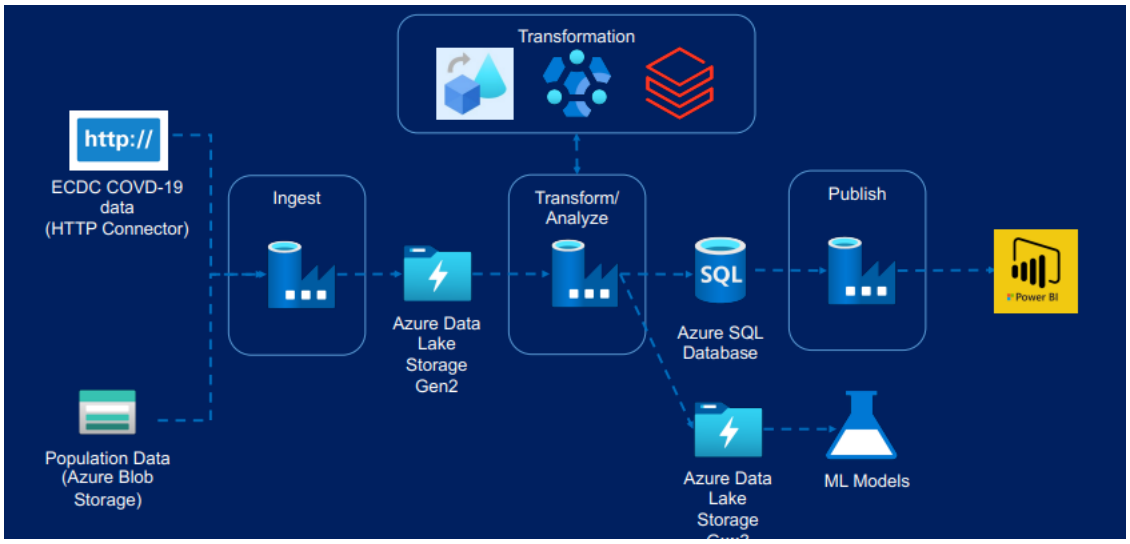
The data sources:
ECDC website:

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

Eurostat Website:

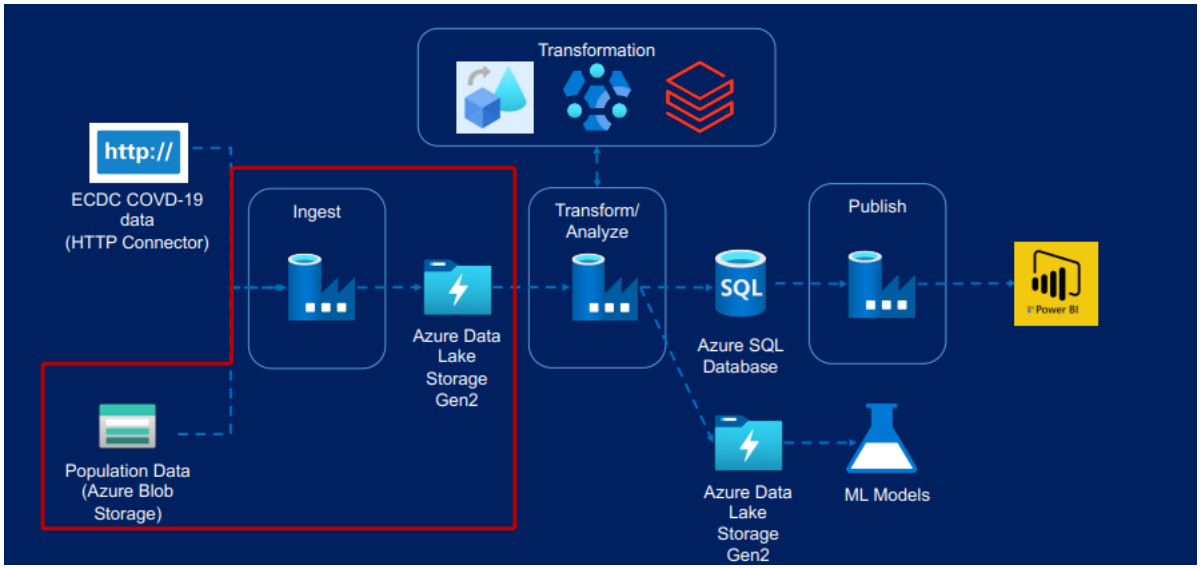
- Population by age

2. Solution Architecture



3. Data ingestion

Overview



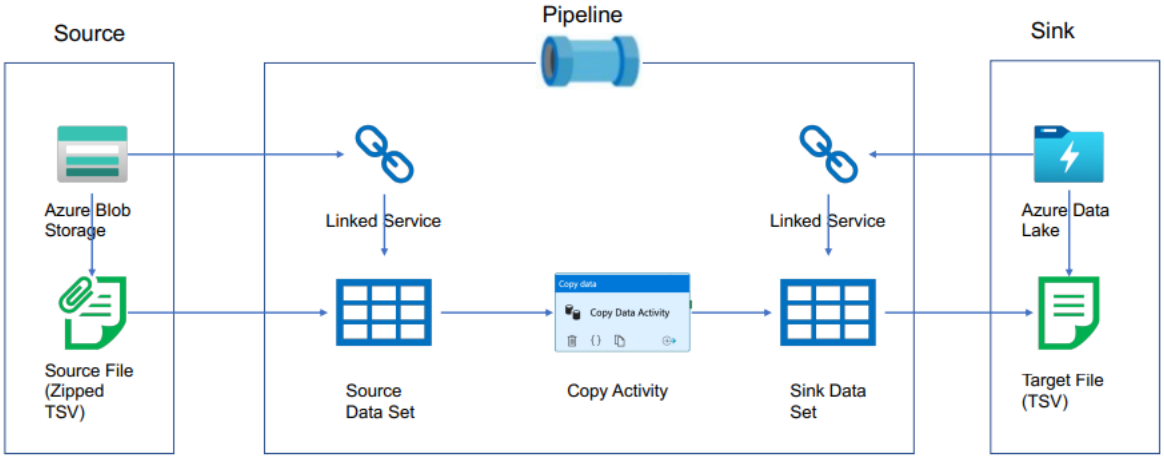
Worked with:

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

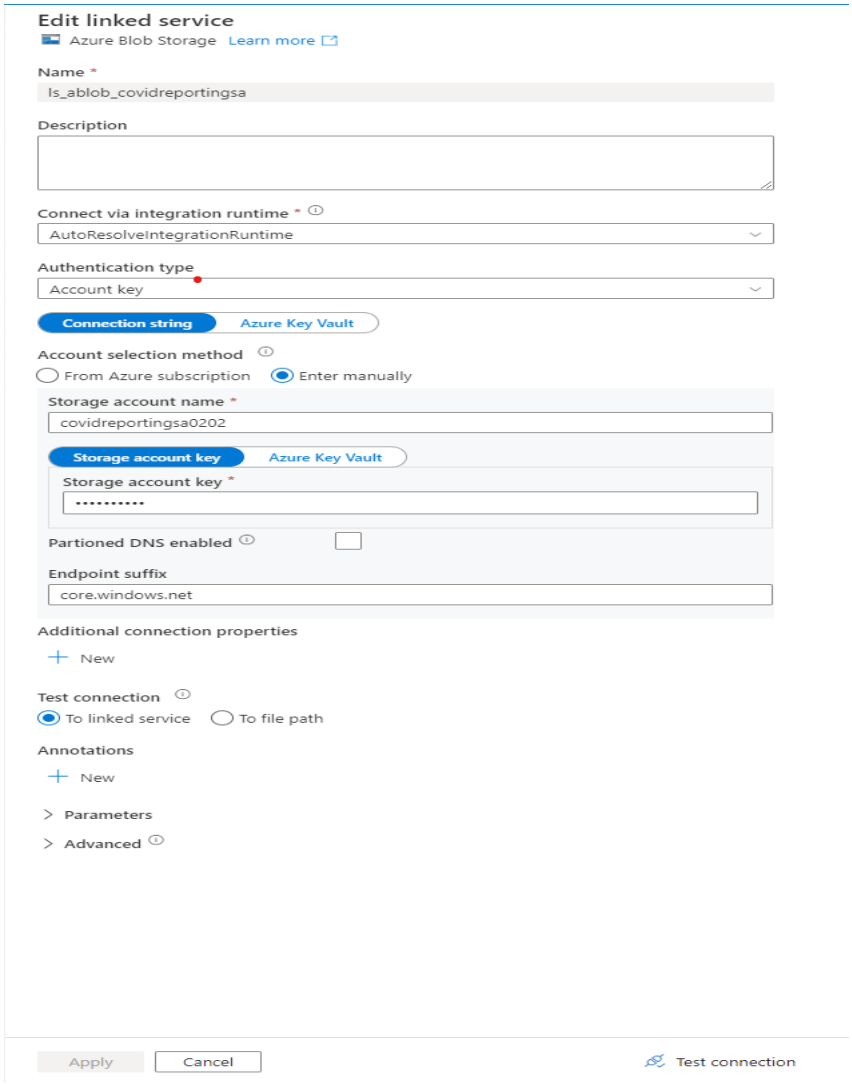
- Trigger

3.1 Data ingestion with Azure blob storage

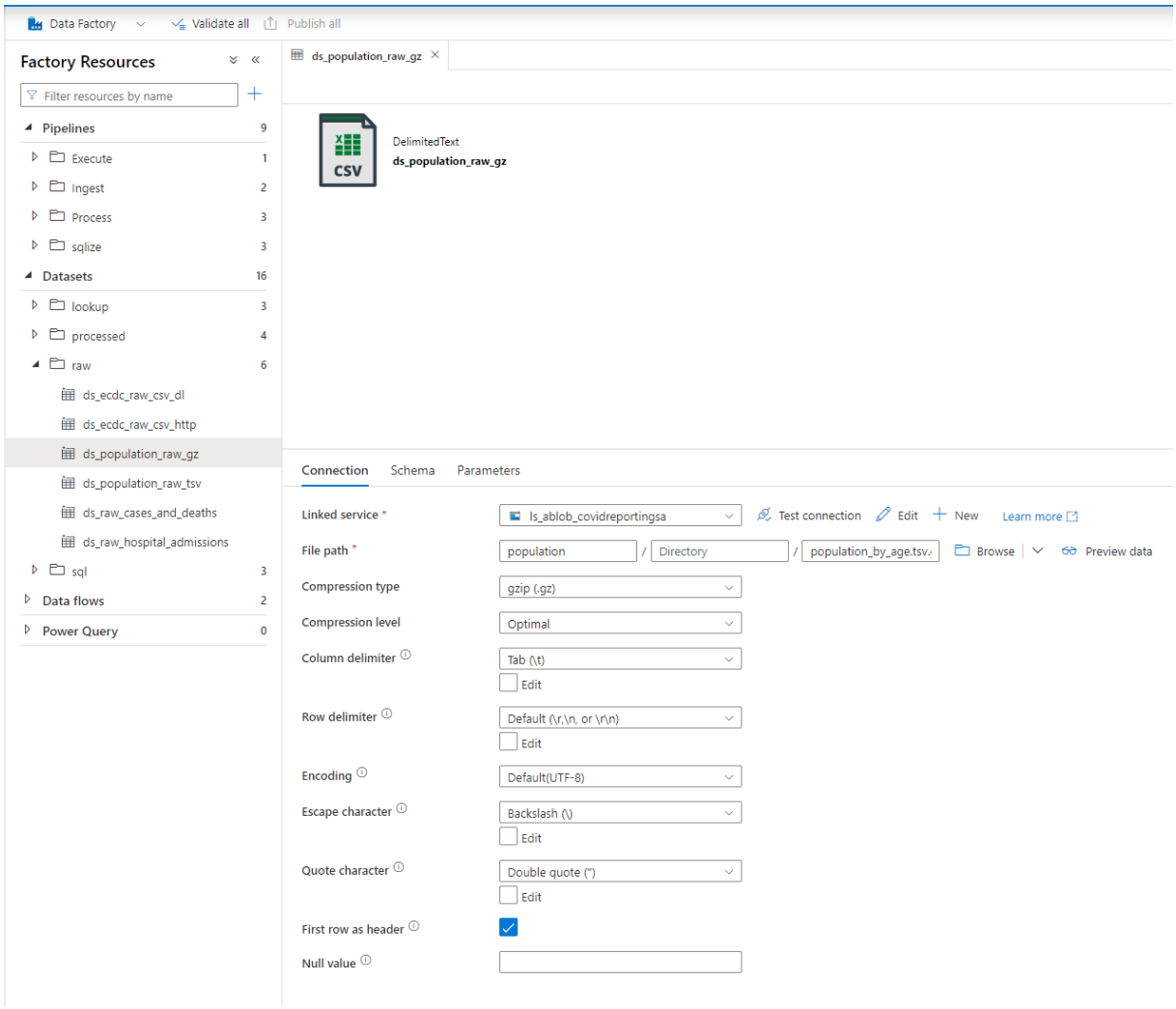
Step 1 - Copy activity overview



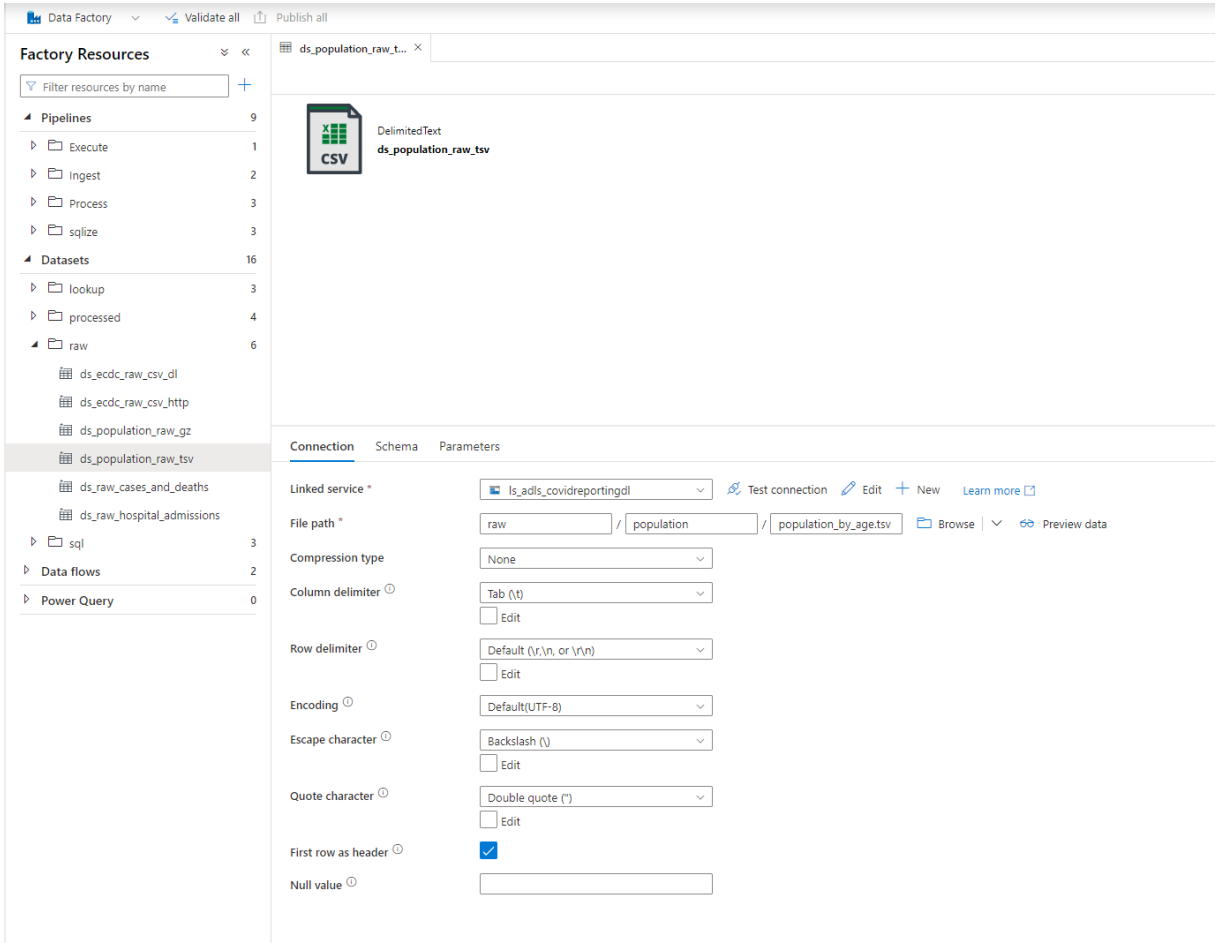
Step 2 - Linked service



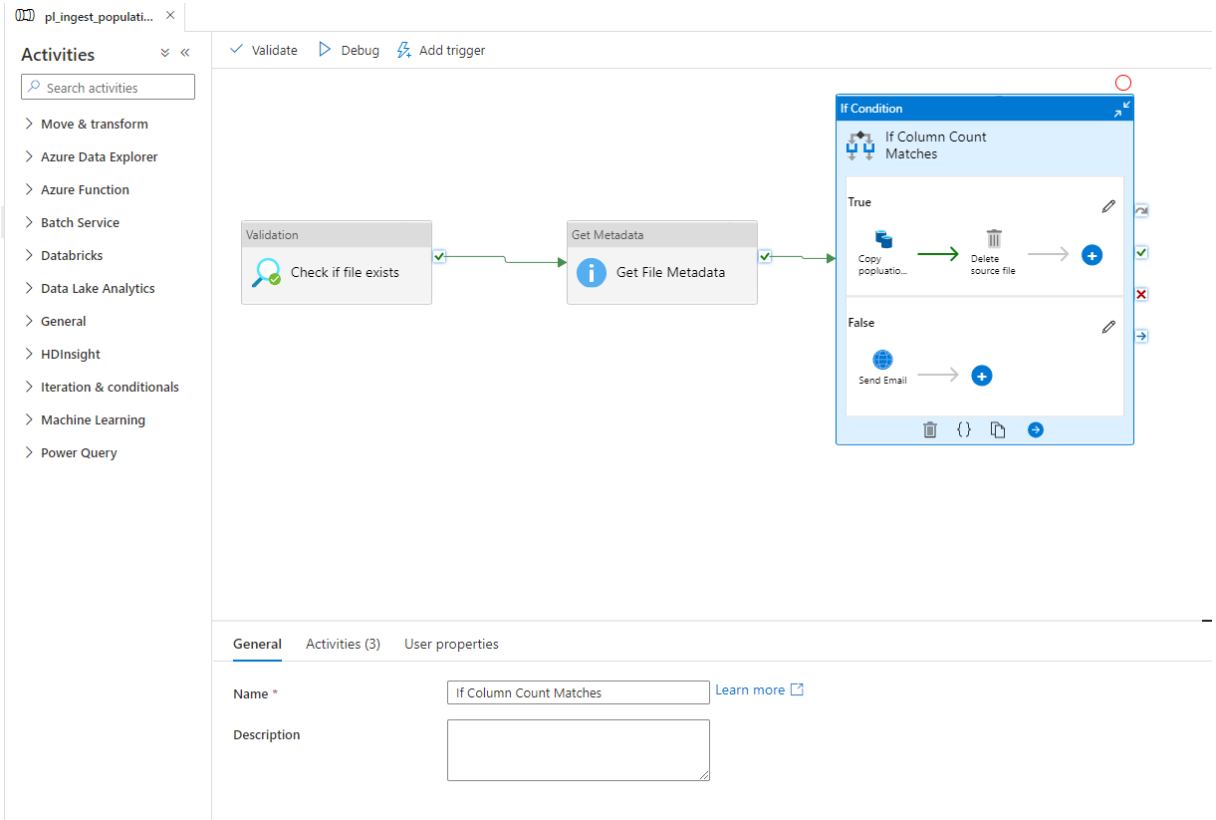
Step 3 - Blob data set (source)



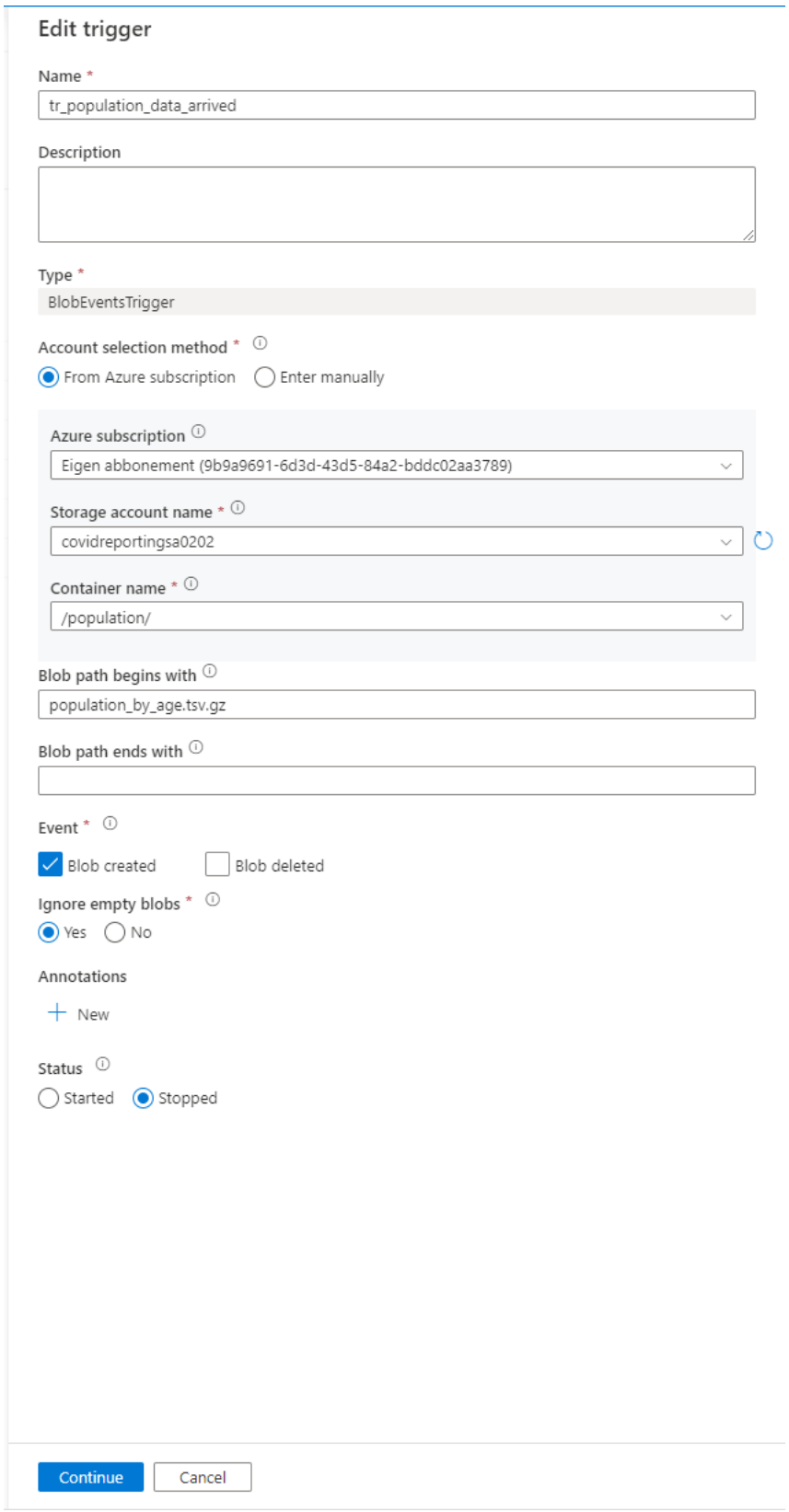
Step 4 - Data Lake data set (sink)



Step 5 - Creating an ingestion pipeline

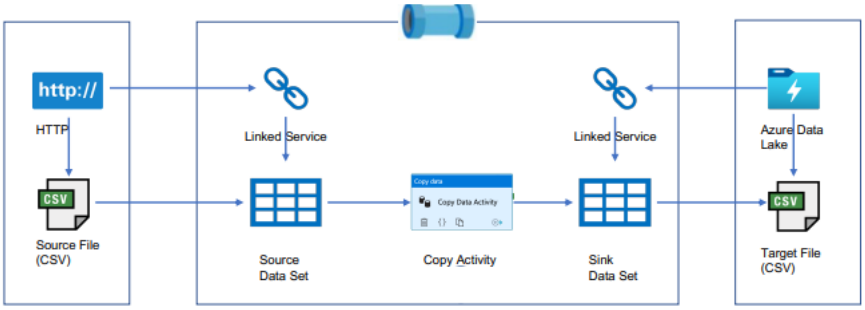


Step 6 - Trigger (ingestion) pipeline

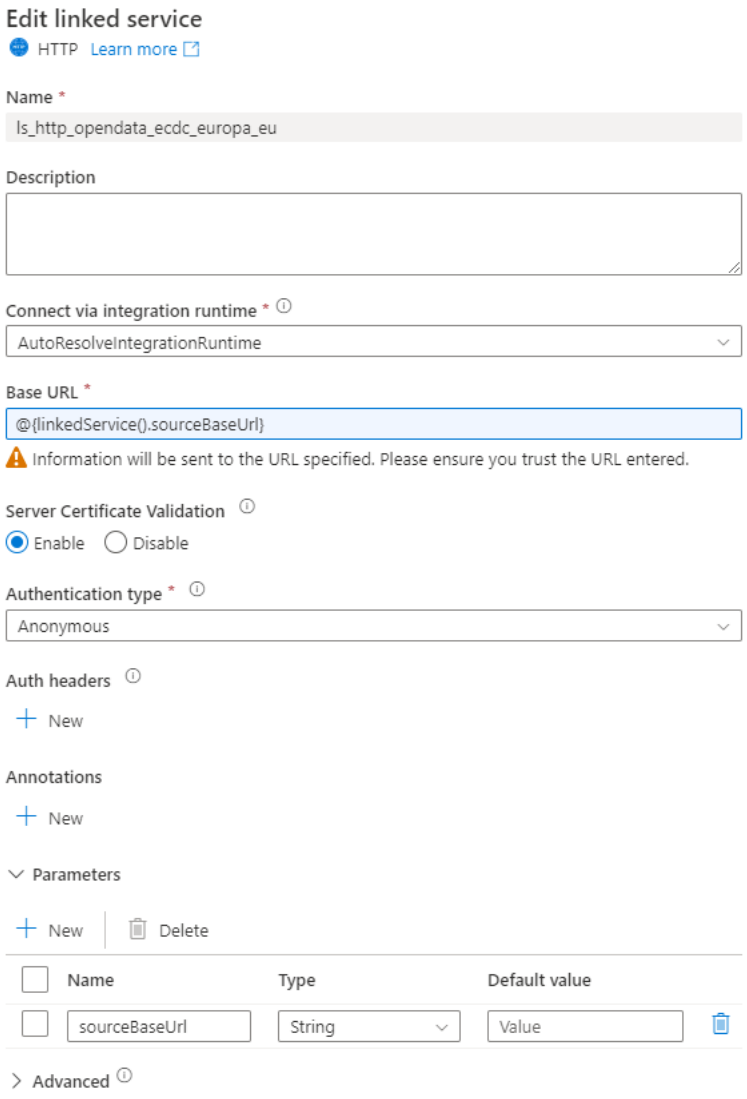


3.2 Data ingestion from HTTP

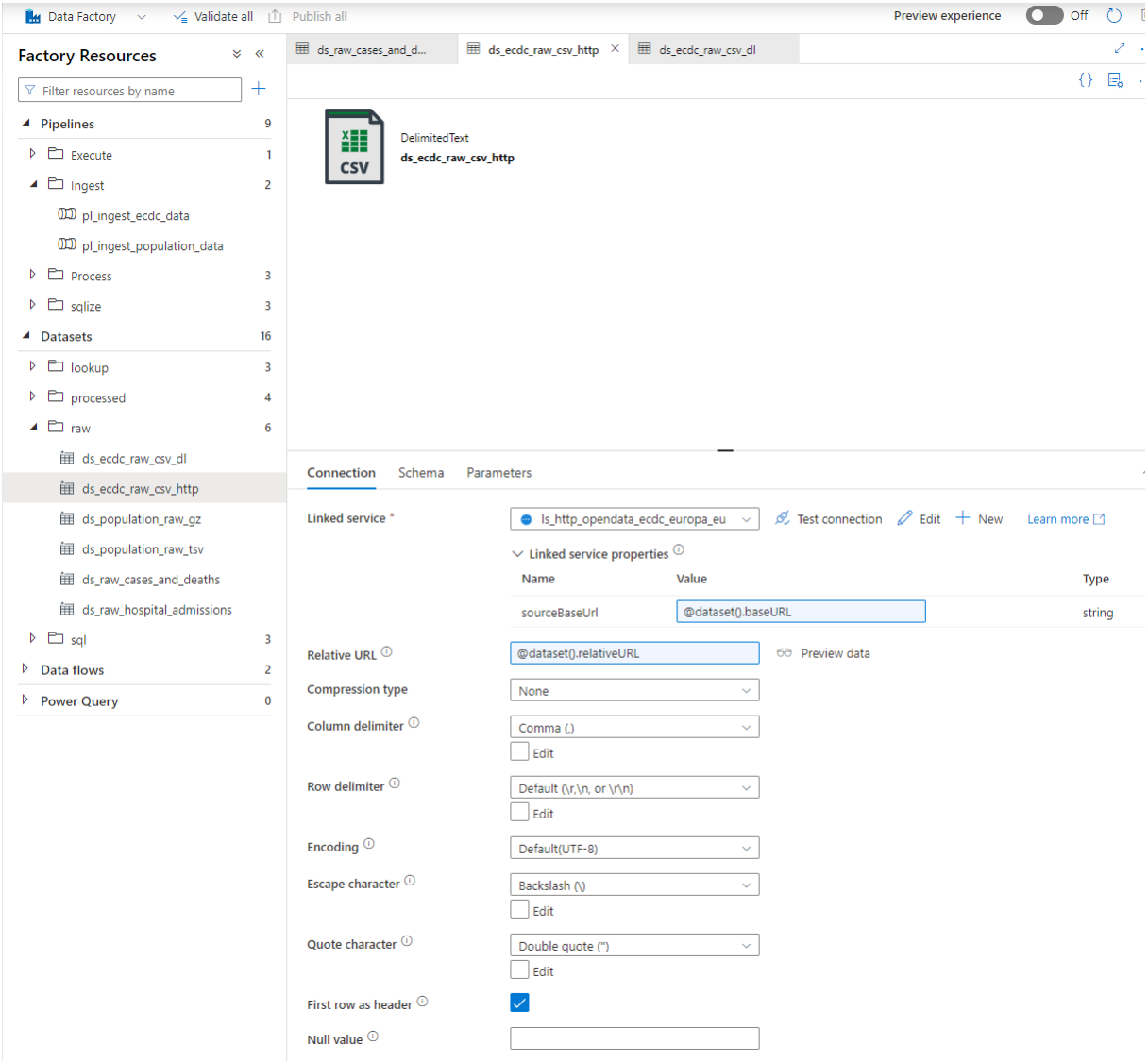
Step 0 - Overview copy activity from HTTP



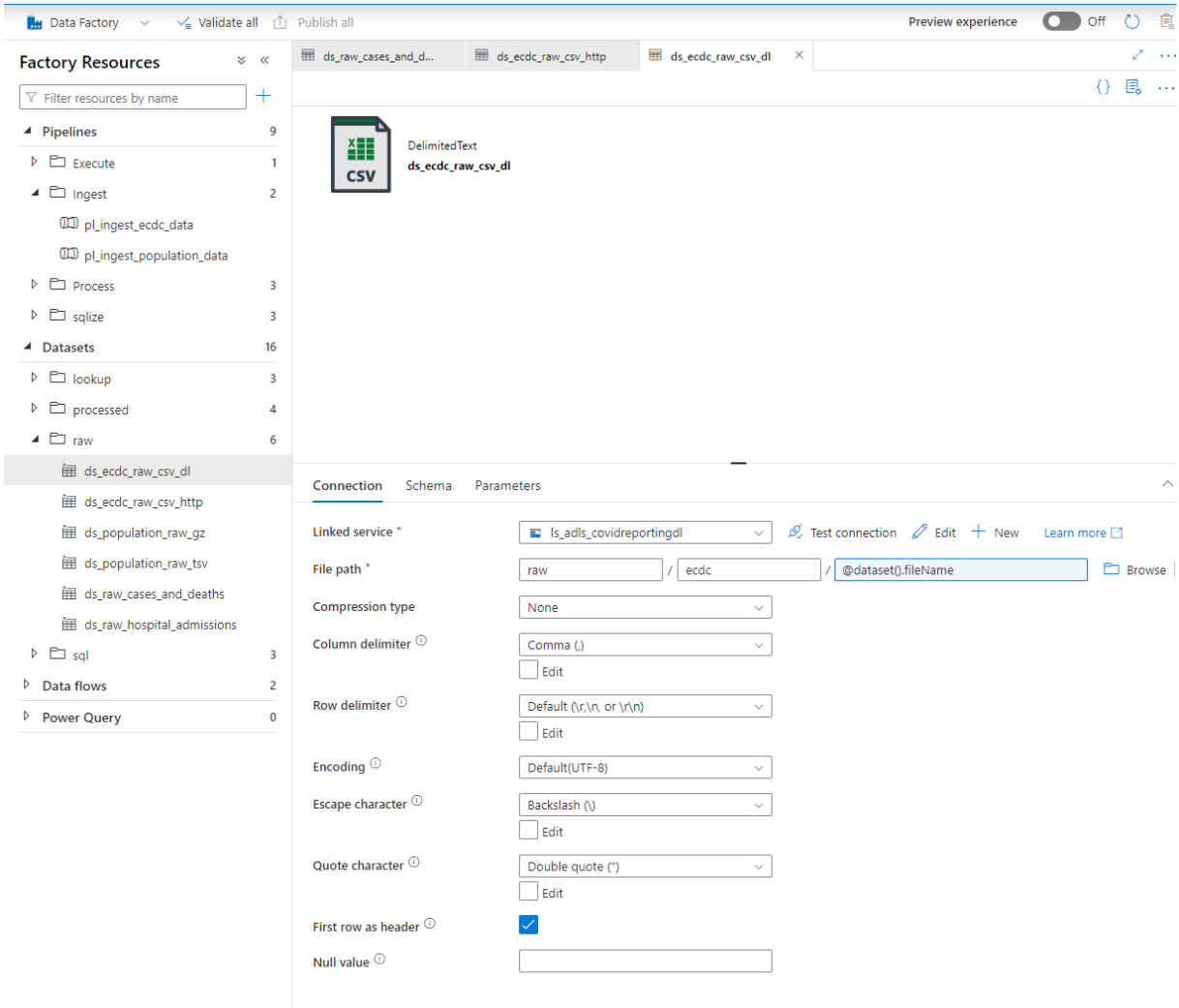
Step 1 - Created linked services



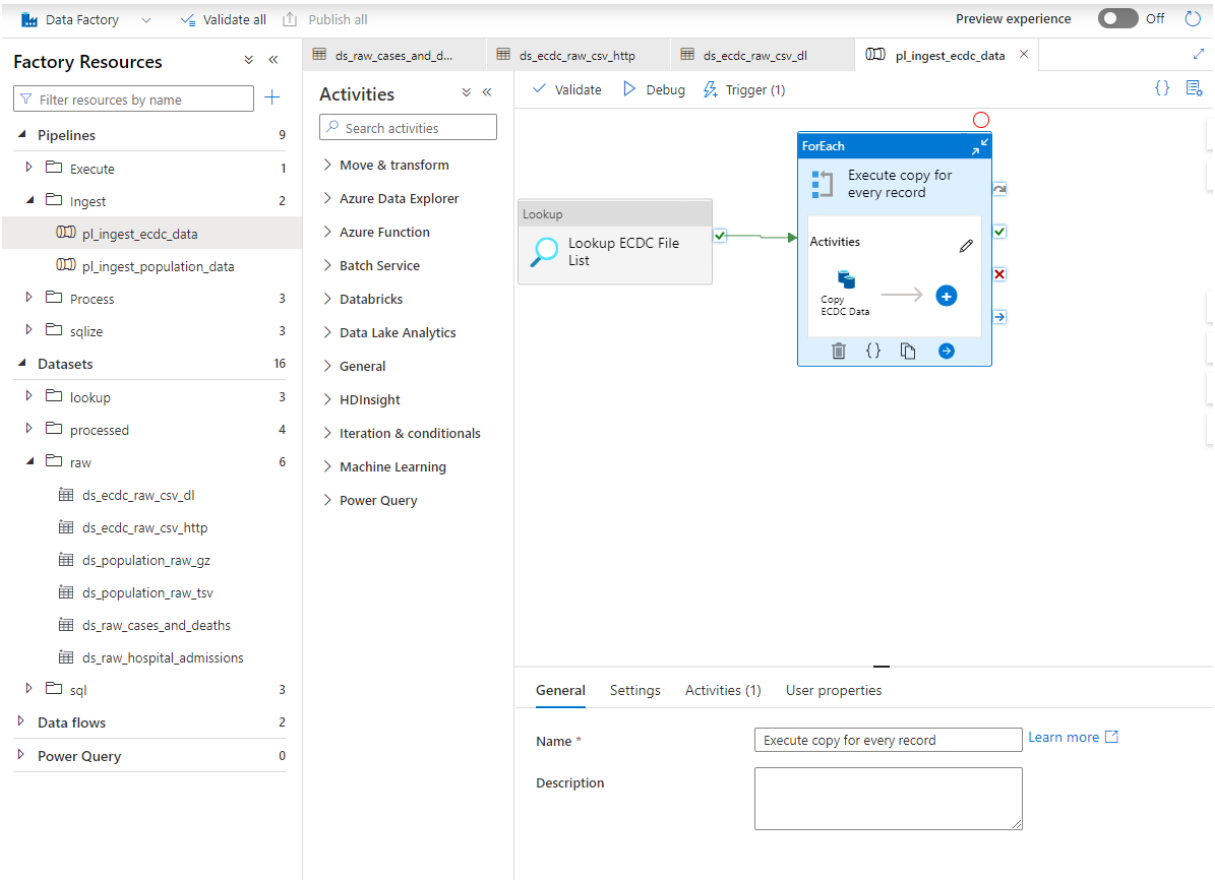
Step 1 - Create data set (http - source)



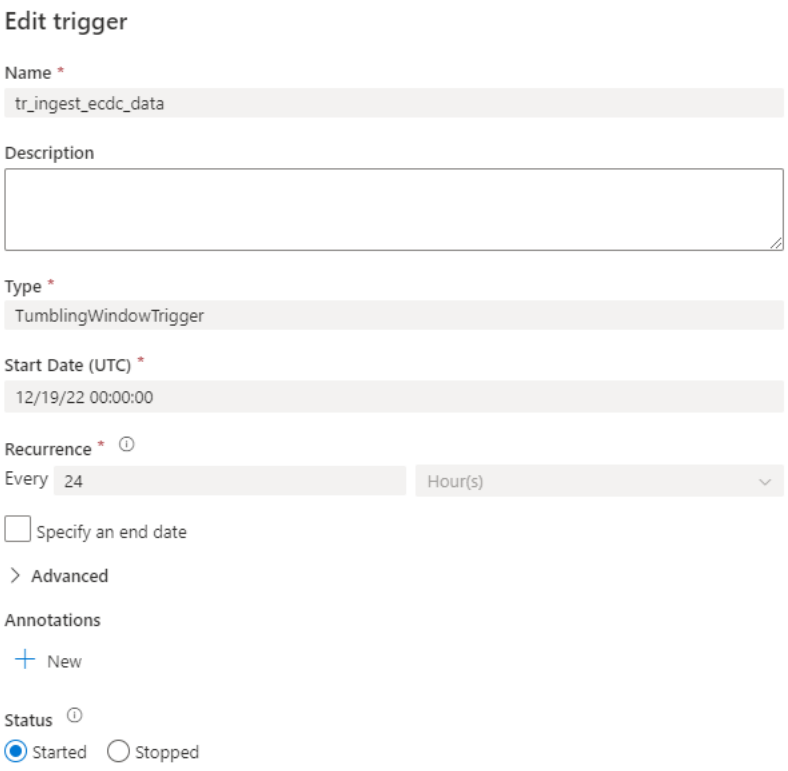
Step 2 - Create data set (DL sink)



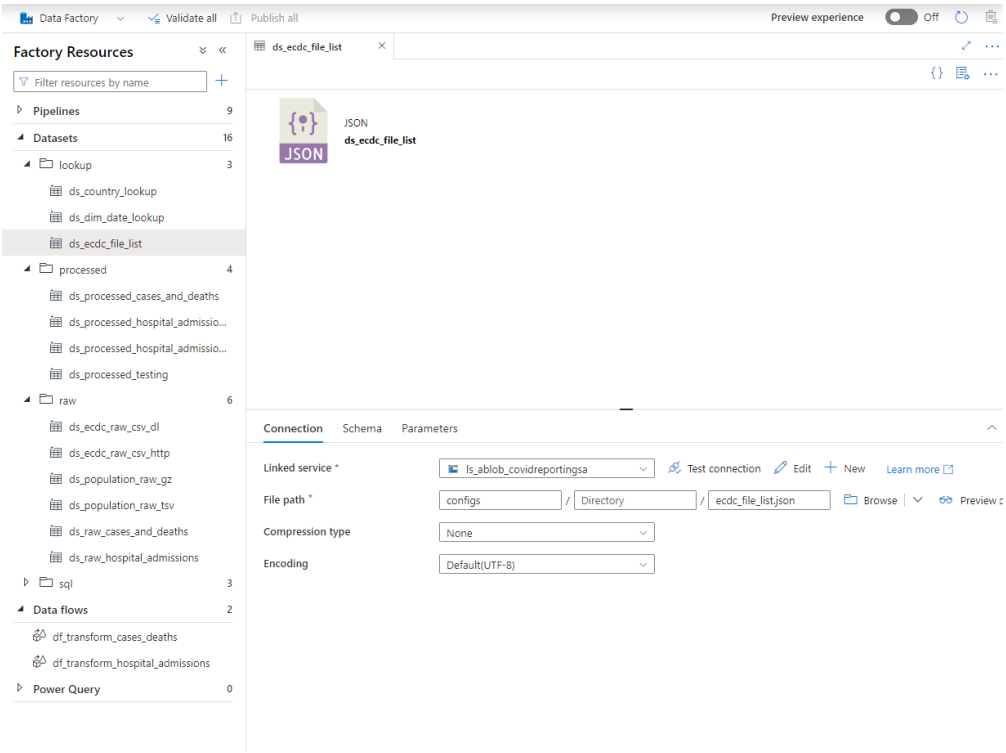
Step 3 - Create pipeline for data ingestion



Step 4 - Create trigger for pipeline



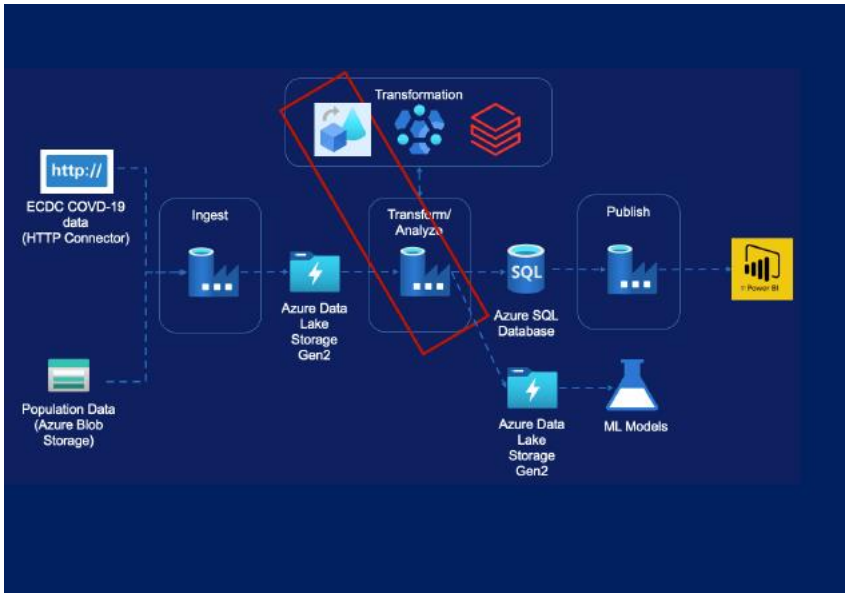
Step 6 - Ecdc file list data set



4. Data transformation

4.1 Data flow

Step 0 – Overview transformation in data flow



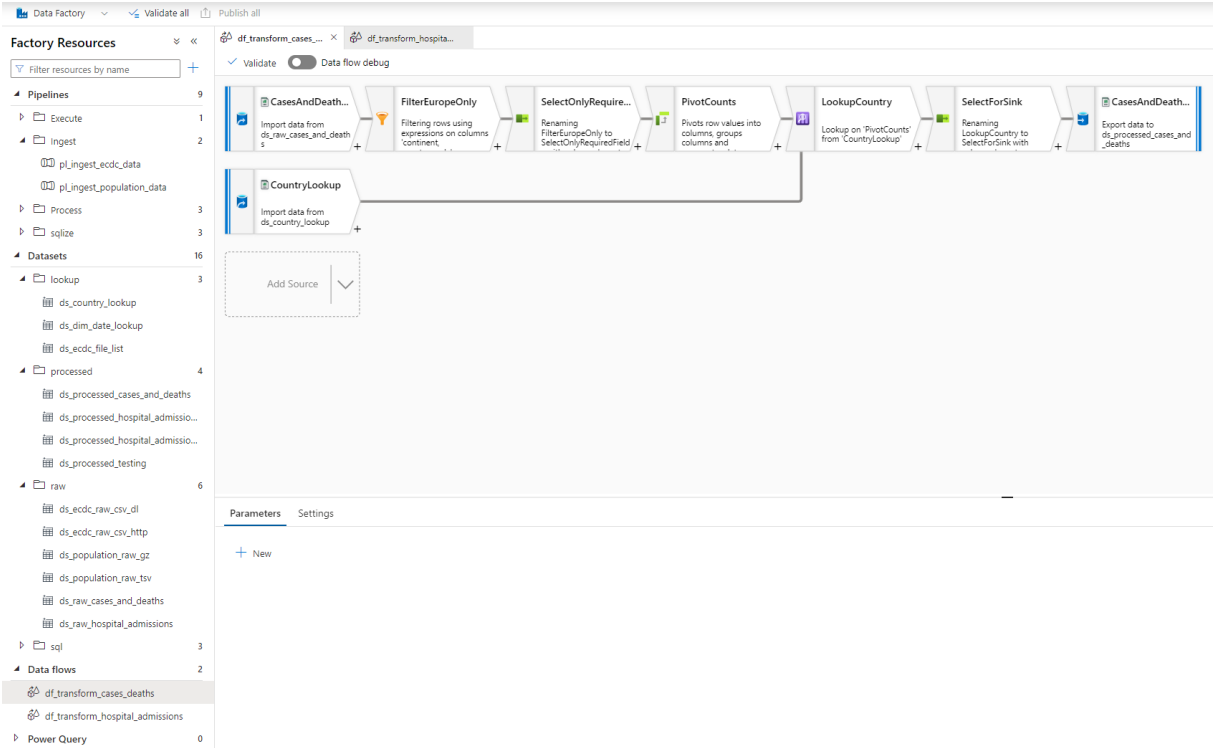
(1) Steps taken:

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

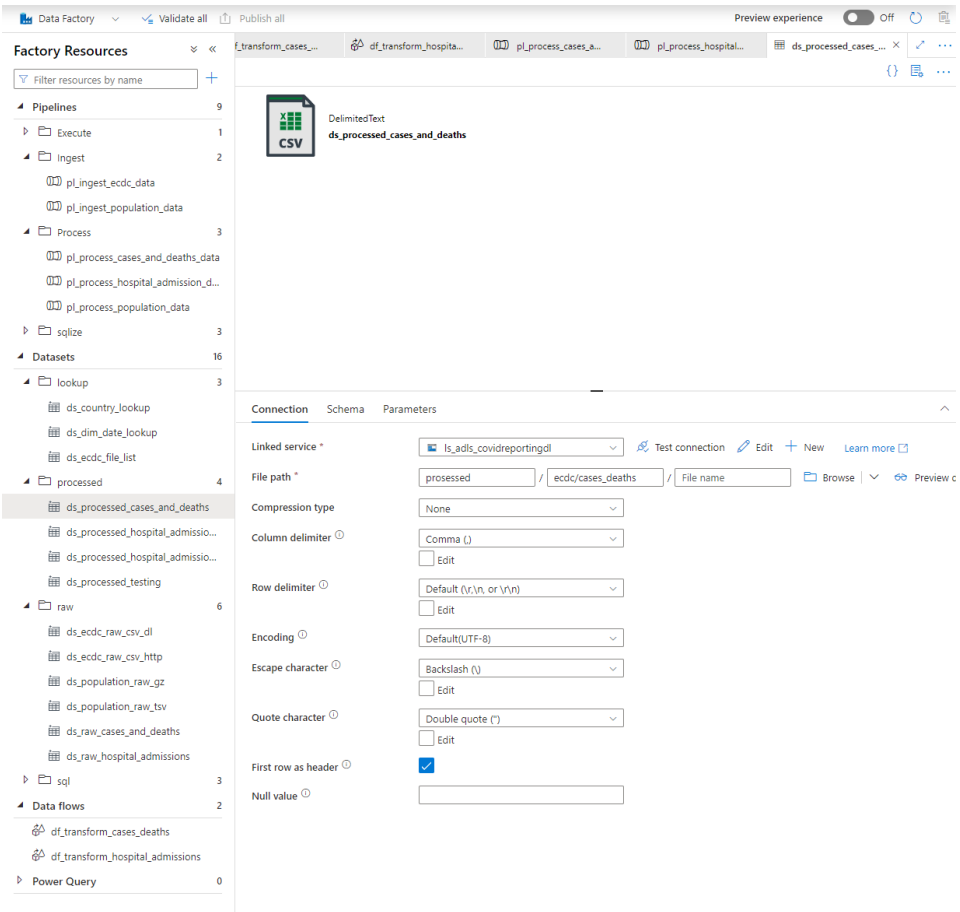
(2) Steps taken:

- 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

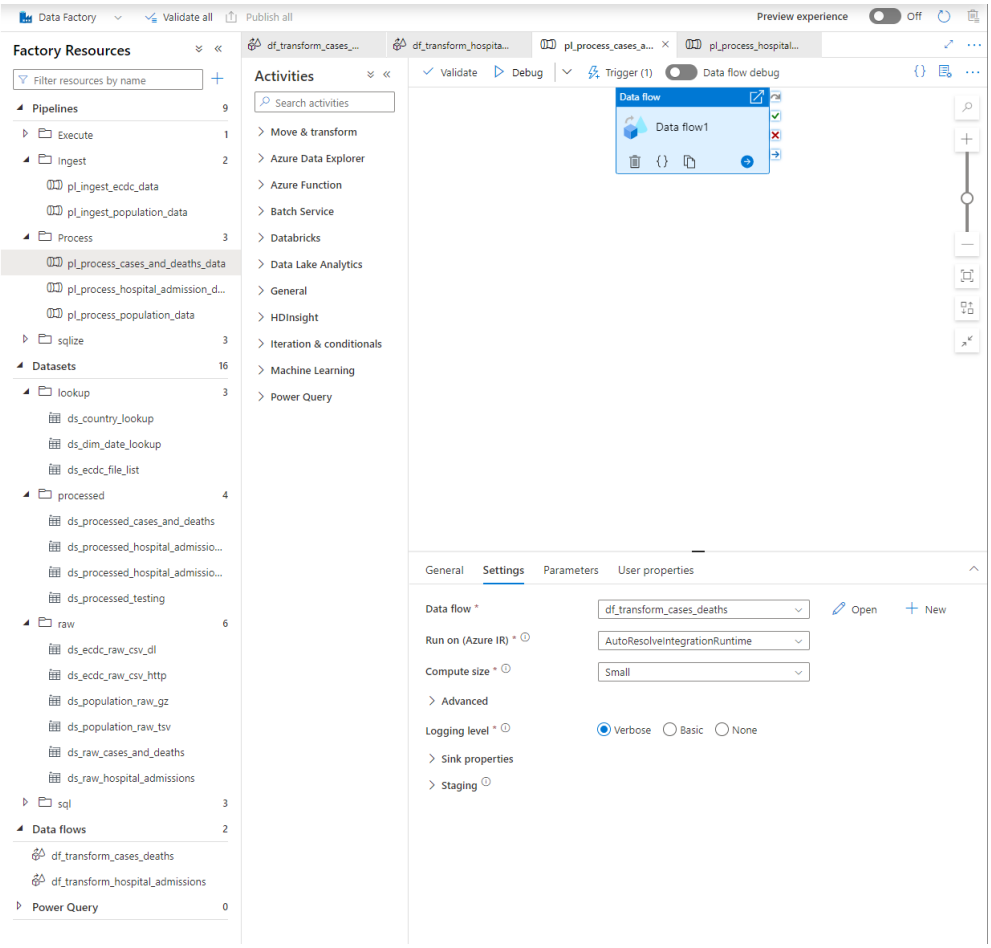
(1) Step 1 - Create transformations with data flow



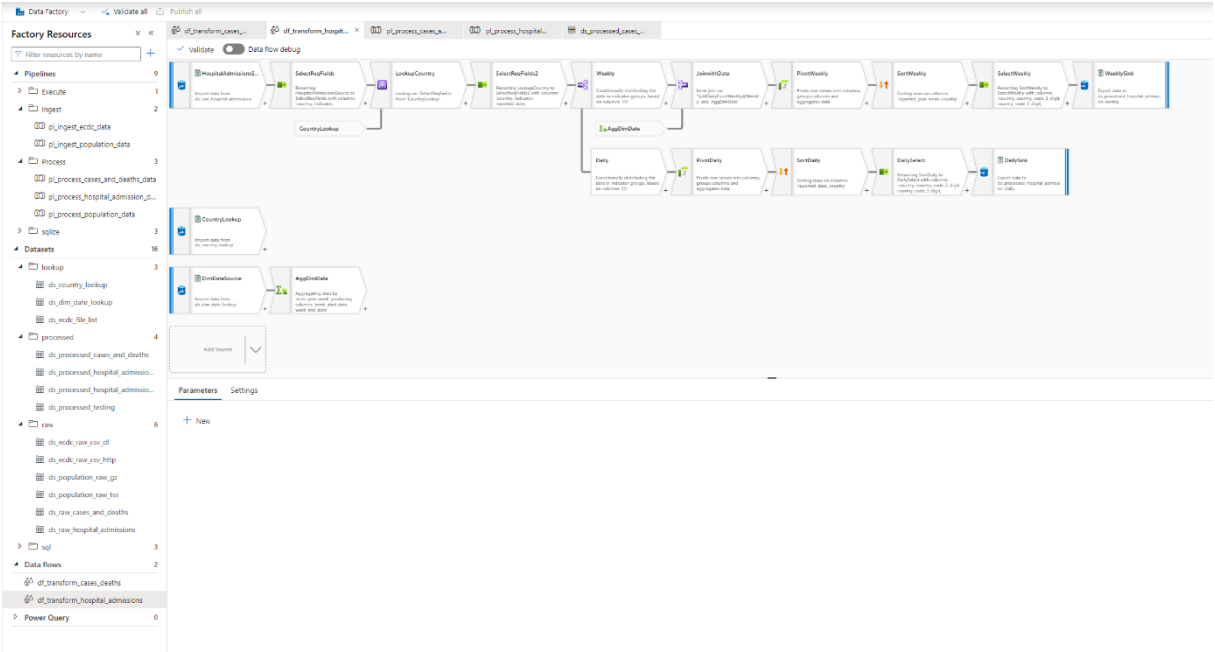
(1) Step 2 - Create data set for processed data



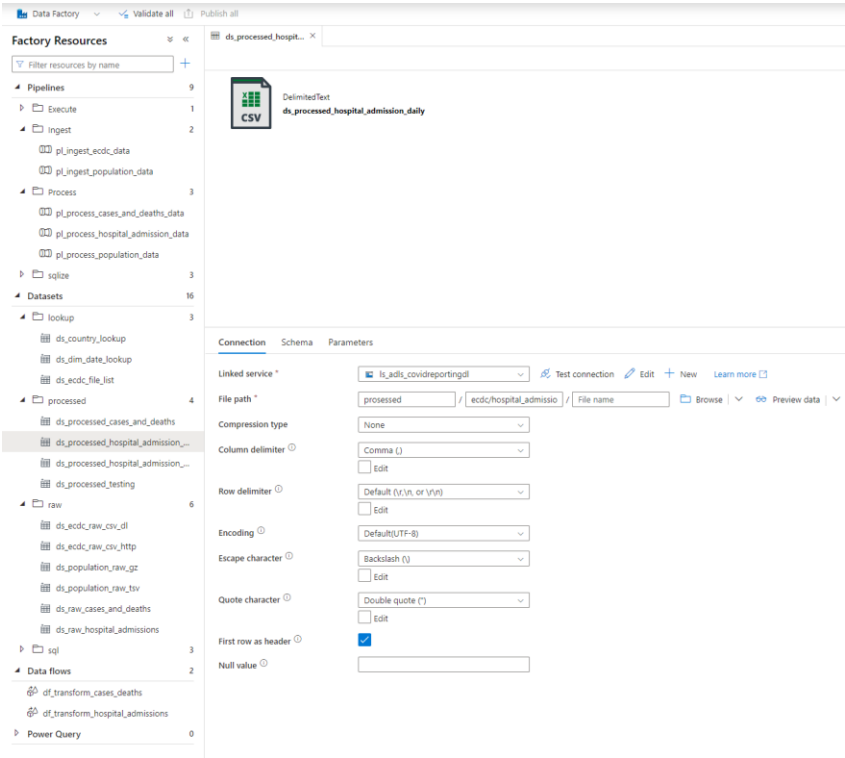
(1) Step 3 – Create pipeline



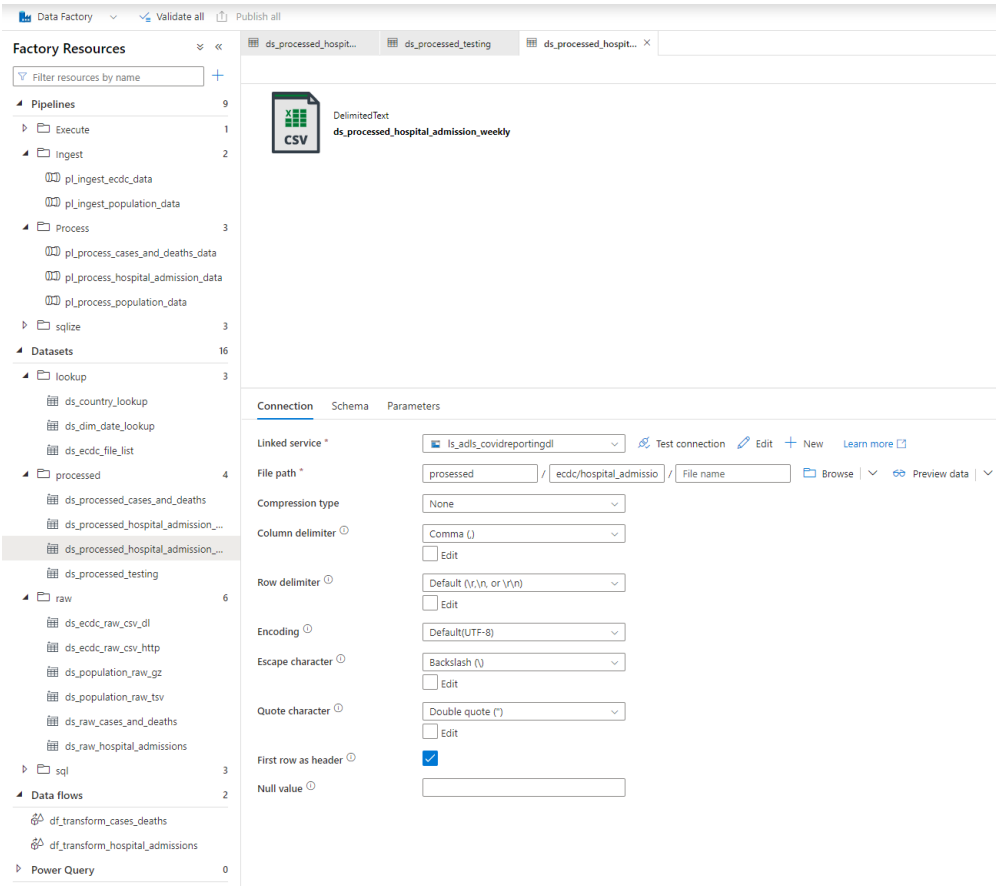
(2) Step 4 - Create transformations with data flow



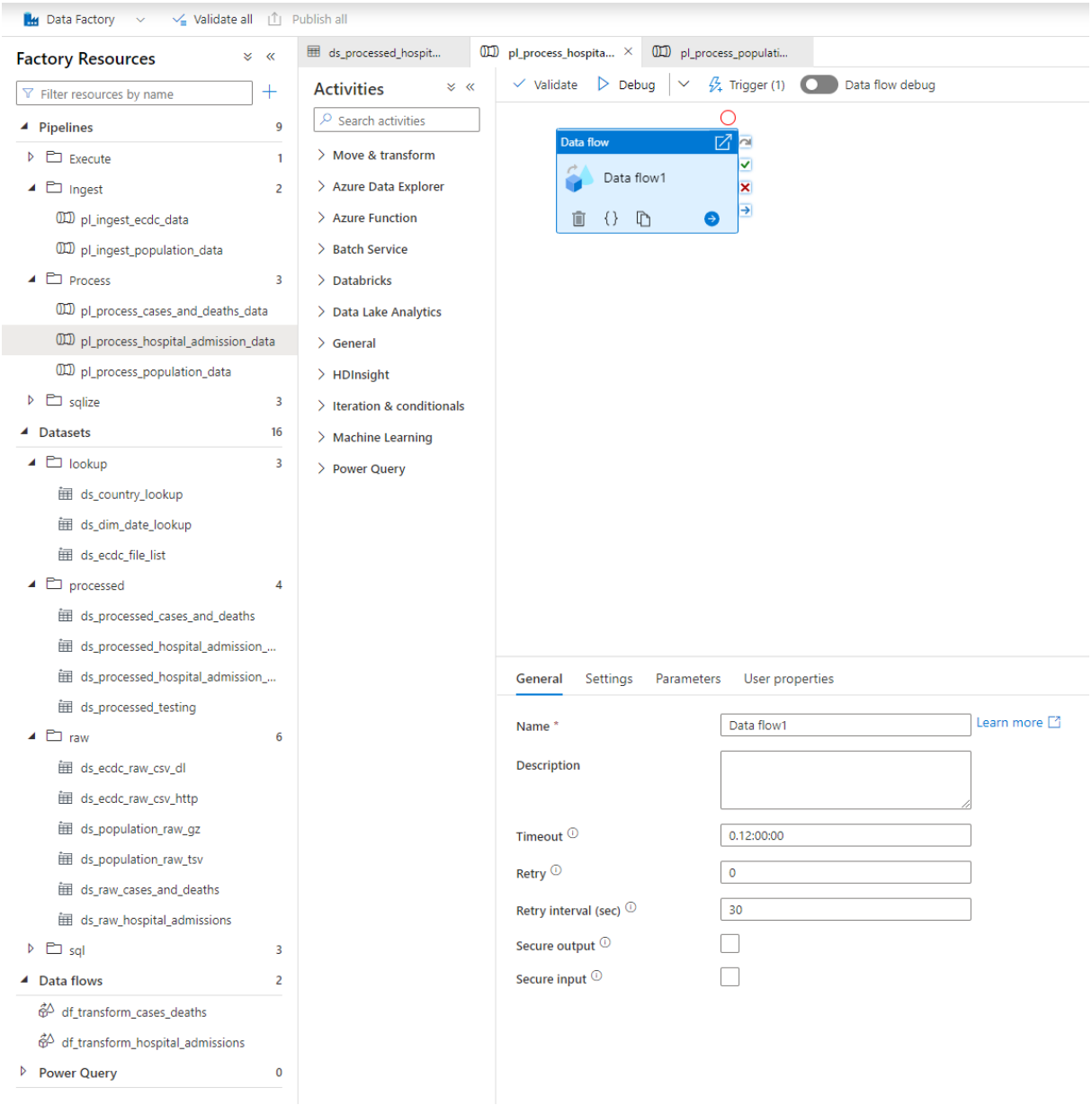
Step 5 - Create data set for processed data (daily)



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

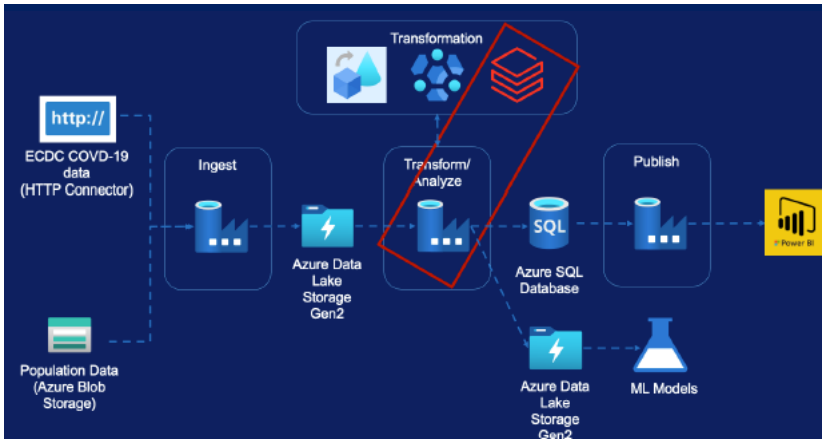


Step 7 - Create pipeline


Data Factory
Validate all
Publish all
Factory Resources
Filter resources by name
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_data
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_admission_...
ds_processed_hospital_admission_...
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
ds_processed_hospit...
pl_process_hospita...
pl_process_populati...
Activities
Search activities
Move & transform
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Validate
Debug
Trigger (1)
Data flow debug
Data flow
Data flow1
General
Settings
Parameters
User properties
Name *
Data flow1
Learn more
Description
Timeout
0.12:00:00
Retry
0
Retry interval (sec)
30
Secure output
Secure input

4.2 Databricks

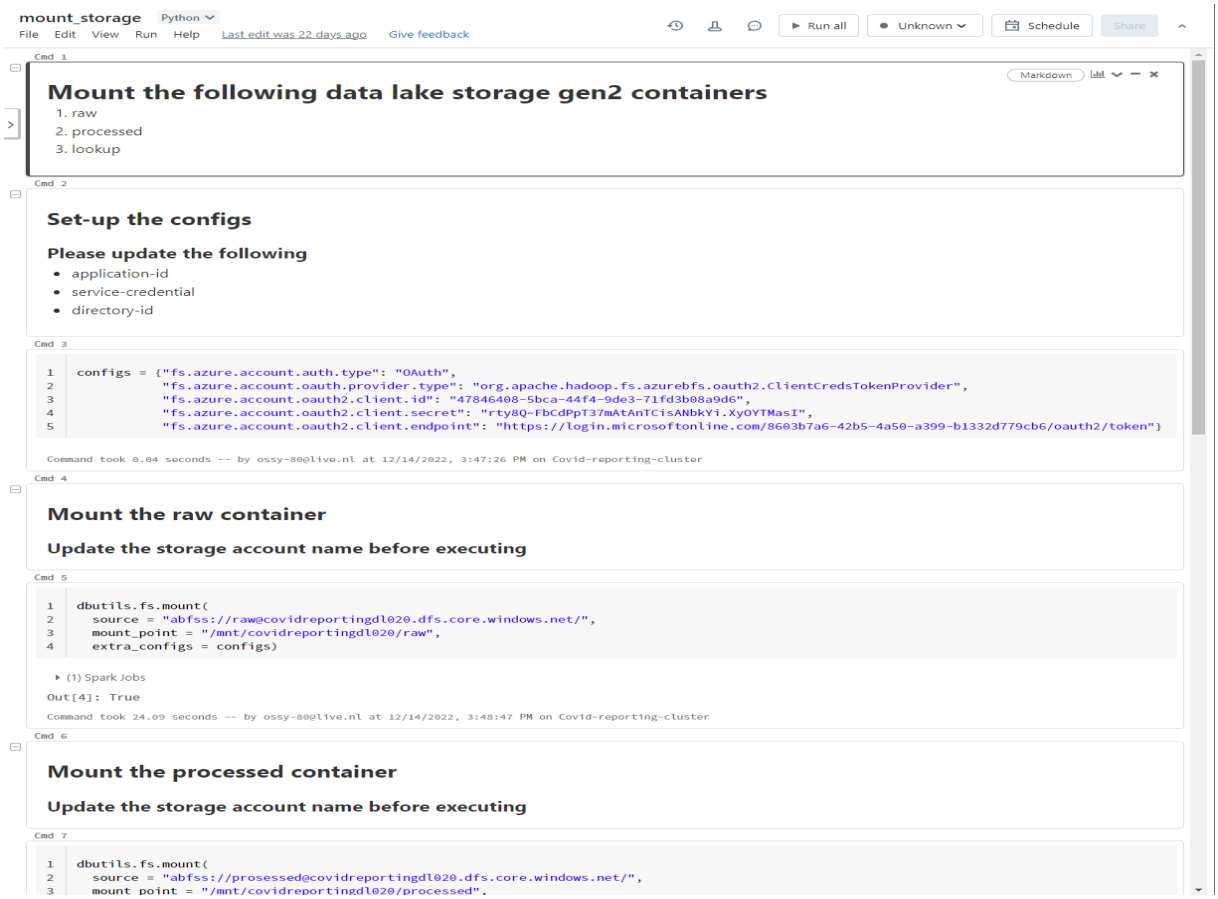
Step 0 - Databricks overview



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

Markdown

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

```
configs = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": "47846408-5bca-44f4-9de3-71fd3b08a9d6",
           "fs.azure.account.oauth2.client.secret": "rty8Q~FbCdPpT37mAtAnTCisANbkYi.XyOYTMasI",
           "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

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

```
dbutils.fs.mount(
  source = "abfss://prosessed@covidreportingdl020.dfs.core.windows.net/",
  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

```python
from pyspark.sql.functions import *
```

Cmd 4

Read the population data & create a temp view

Cmd 5

```python
df_raw_population = spark.read.csv("/mnt/covidreportingdl020/raw/population", sep=r'\t', header=True)
df_raw_population = df_raw_population.withColumn('age_group', regexp_replace(split(df_raw_population['indic_de,geo\\time'], ',')[0], 'PC_', '')).withColumn('country_code', split(df_raw_population['indic_de,geo\\time'], ',')[1])
df_raw_population = df_raw_population.select(col("country_code").alias("country_code"),
                                              col("age_group").alias("age_group"),
                                              col("2019 ").alias("percentage_2019"))
df_raw_population.createOrReplaceTempView("raw_population")
```

Cmd 6

Pivot the data by age group

Cmd 7

Python

```python
# Create a data frame with pivoted percentages
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")
df_raw_population_pivot.createOrReplaceTempView("raw_population_pivot")
```

Cmd 8

Read the country lookup

Cmd 9

```python
# Create a data frame for the country lookup
df_dim_country = spark.read.csv("/mnt/covidreportingdl020/lookup/dim_country", sep=r',', header=True)
df_dim_country.createOrReplaceTempView("dim_country")
```

Cmd 10

Join population data with country lookup

Cmd 11

```python
df_processed_population = spark.sql("""SELECT c.country,
       c.country_code_2_digit,
       c.country_code_3_digit,
       c.population,
       p.Y0_14  AS age_group_0_14,
       p.Y15_24 AS age_group_15_24,
       p.Y25_49 AS age_group_25_49,
       p.Y50_64 AS age_group_50_64,
       p.Y65_79 AS age_group_65_79,
       p.Y80_MAX AS age_group_80_max
  FROM raw_population_pivot p
```

Step 2 – transform data 2 / 2

transform_population_data Python

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

Run all Connect Schedule Share

```
        p.Y0_14  AS age_group_0_14,
        p.Y15_24 AS age_group_15_24,
        p.Y25_49 AS age_group_25_49,
        p.Y50_64 AS age_group_50_64,
        p.Y65_79 AS age_group_65_79,
        p.Y80_MAX AS age_group_80_max
  FROM raw_population_pivot p
  JOIN dim_country c ON p.country_code = country_code_2_digit
 ORDER BY country""")
```

Cmd 12

Write output to the processed mount point

Cmd 13

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

Cmd 14

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

Databrick 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

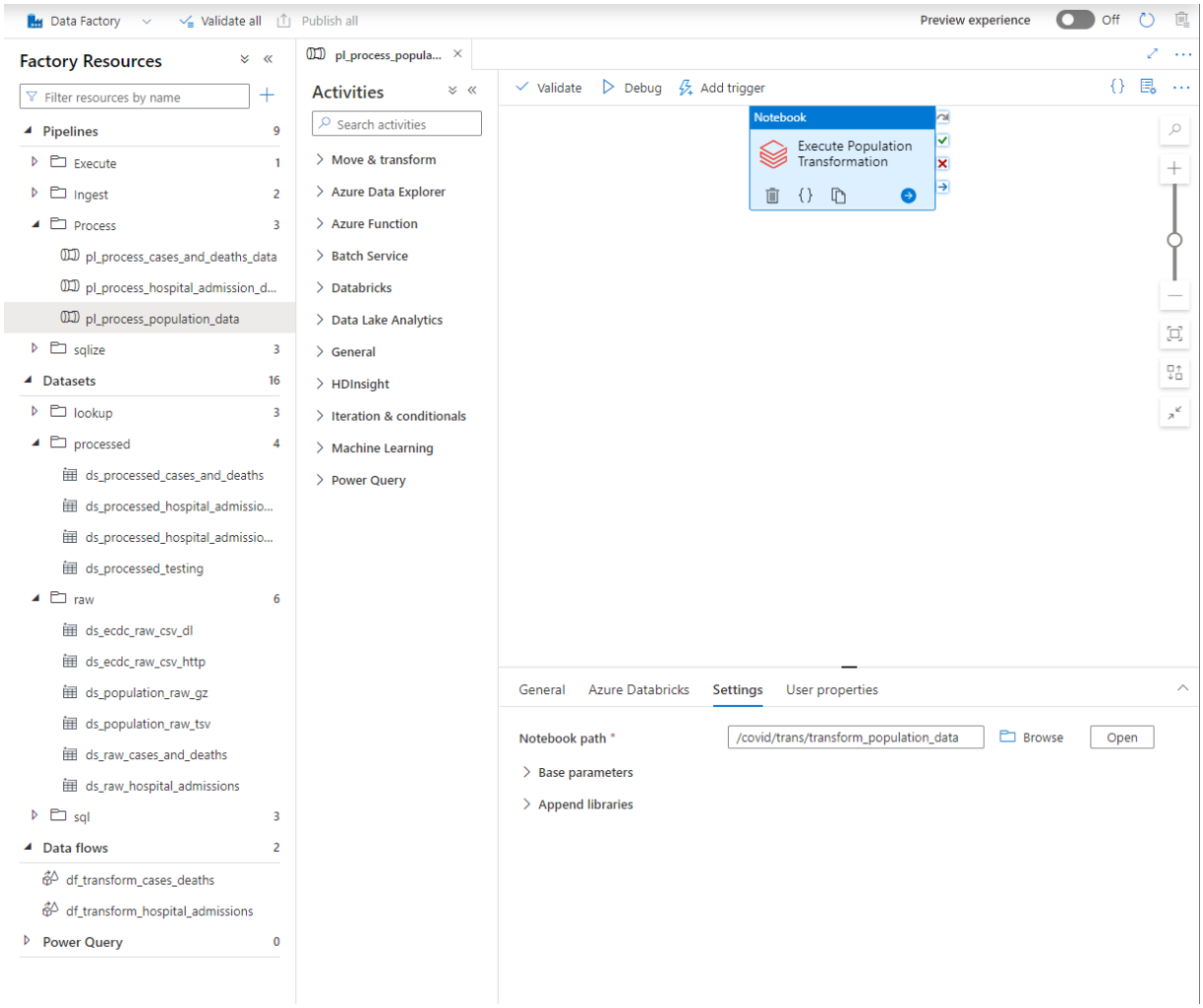
Annotations
New

Parameters

Advanced

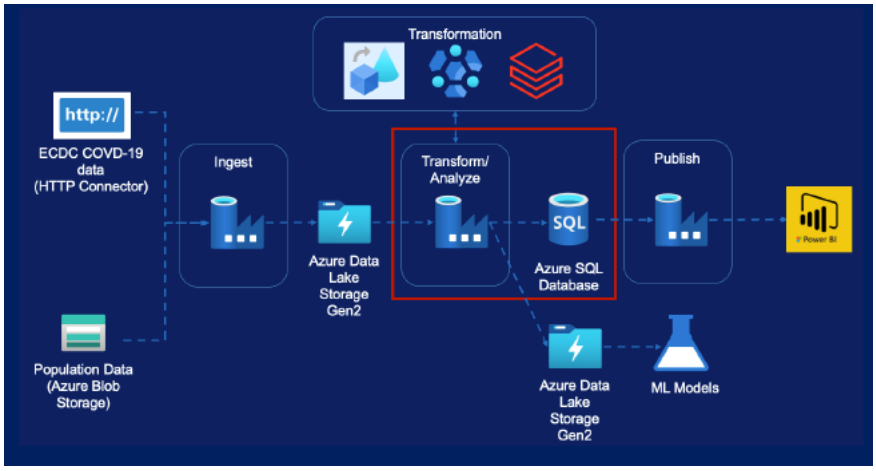
Save Cancel Test connection

Step 4 – Create pipeline



5. Load into database

Step 0 – Overview SQL database



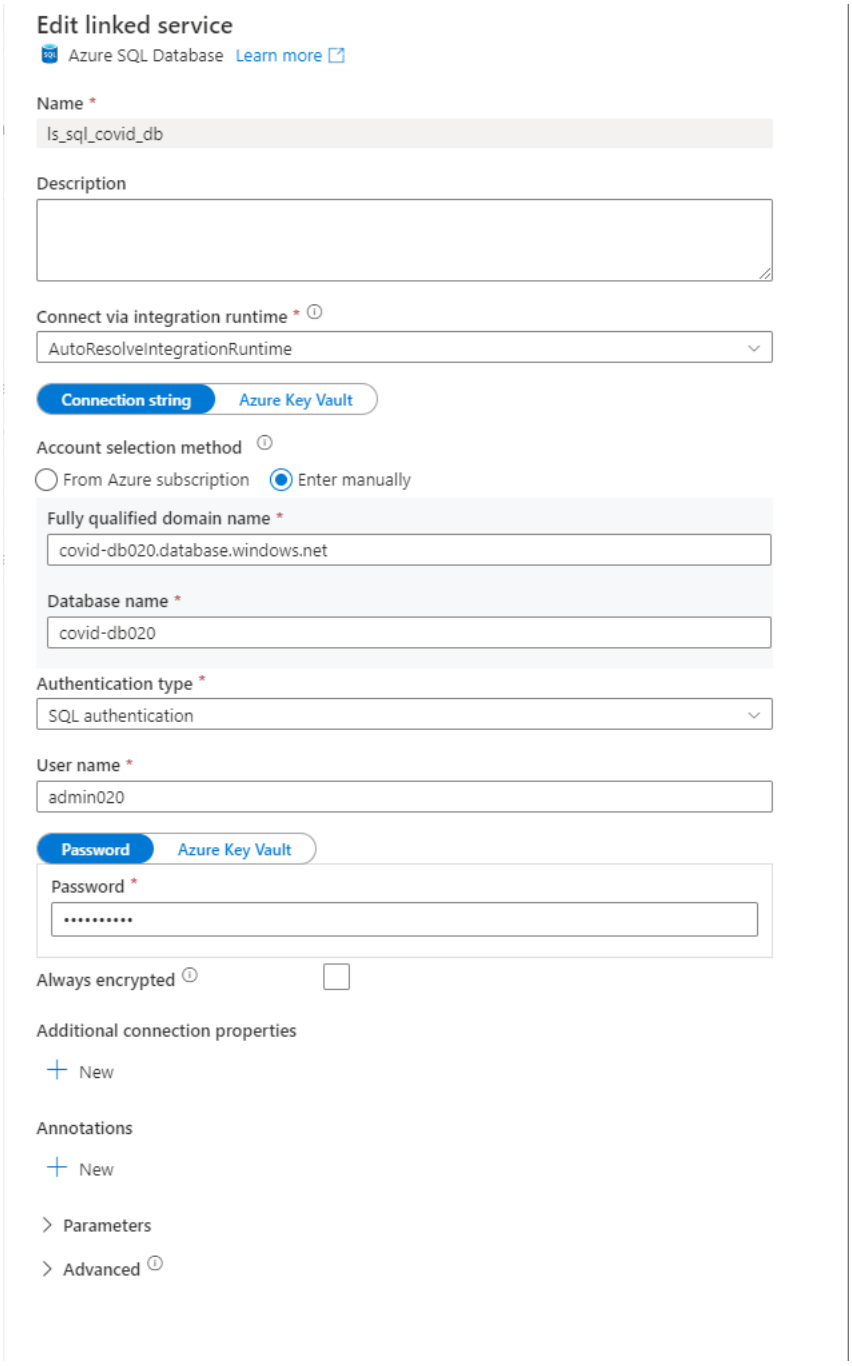
Steps taken:

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

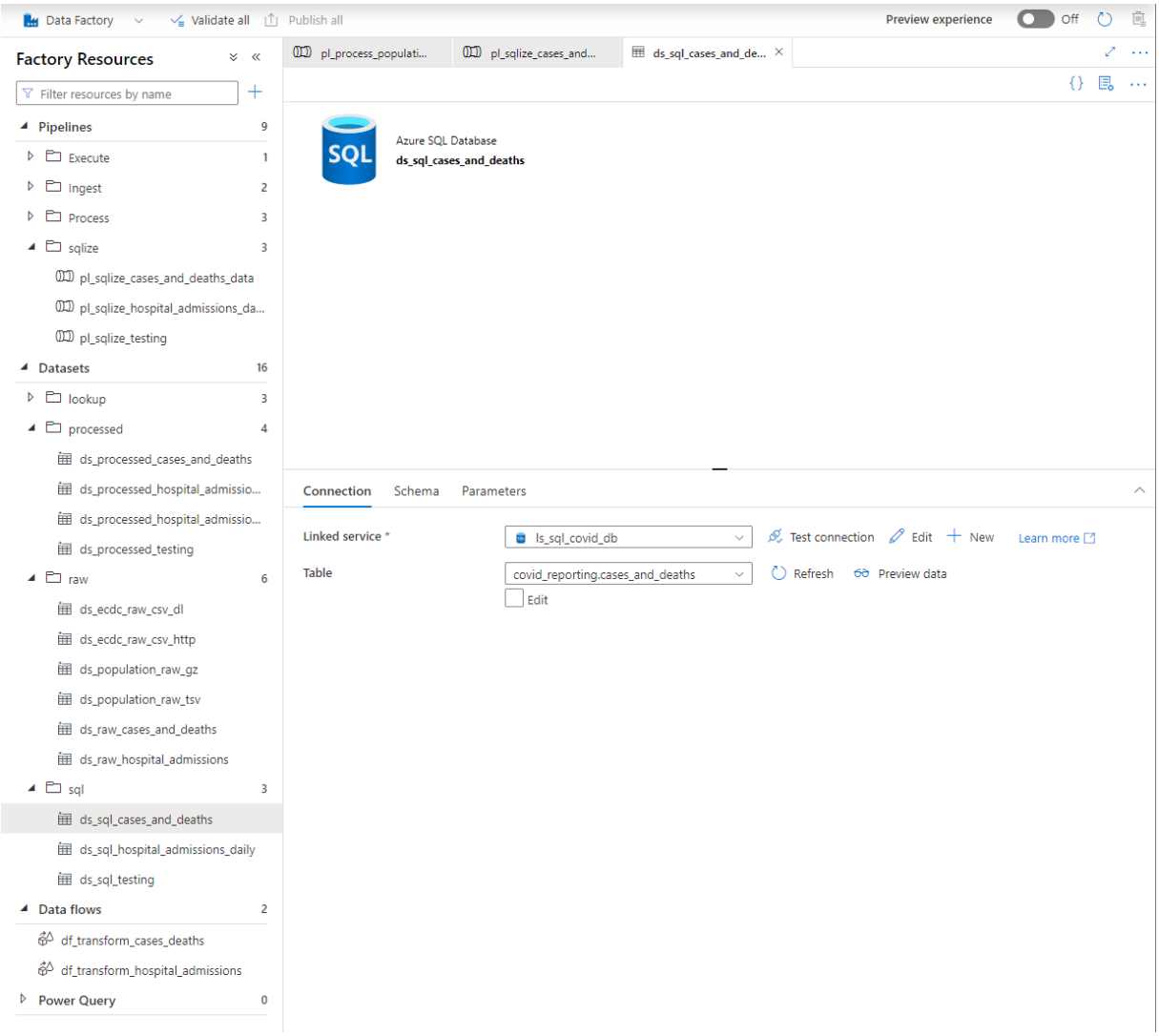
Step 1 – Create SQL script

```sql
CREATE SCHEMA covid_reporting
GO

CREATE TABLE covid_reporting.cases_and_deaths
(
    country                 VARCHAR(100),
    country_code_2_digit    VARCHAR(2),
    country_code_3_digit    VARCHAR(3),
    population              BIGINT,
    cases_count             BIGINT,
    deaths_count            BIGINT,
    reported_date           DATE,
    source                  VARCHAR(500)
)
GO

CREATE TABLE covid_reporting.hospital_admissions_daily
(
    country                 VARCHAR(100),
    country_code_2_digit    VARCHAR(2),
    country_code_3_digit    VARCHAR(3),
    population              BIGINT,
    reported_date           DATE,
    hospital_occupancy_count BIGINT,
    icu_occupancy_count      BIGINT,
    source                  VARCHAR(500)
)
GO

CREATE TABLE covid_reporting.testing
(
    country                 VARCHAR(100),
    country_code_2_digit    VARCHAR(2),
    country_code_3_digit    VARCHAR(3),
    year_week               VARCHAR(8),
    week_start_date         DATE,
    week_end_date           DATE,
    new_cases               BIGINT,
    tests_done              BIGINT,
    population              BIGINT,
    testing_data_source      VARCHAR(500)
)
GO
```

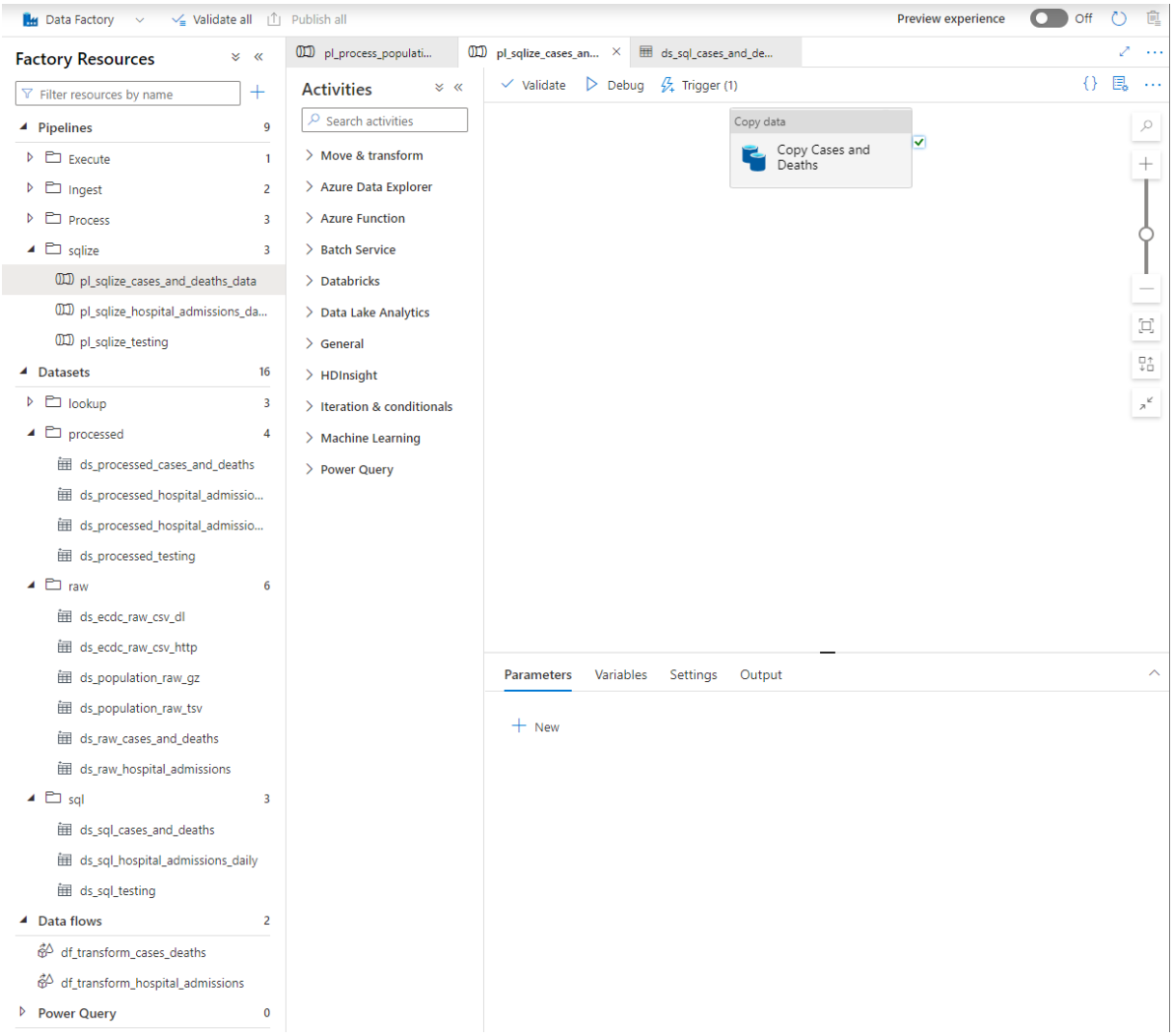
Step 2 – Create linked service



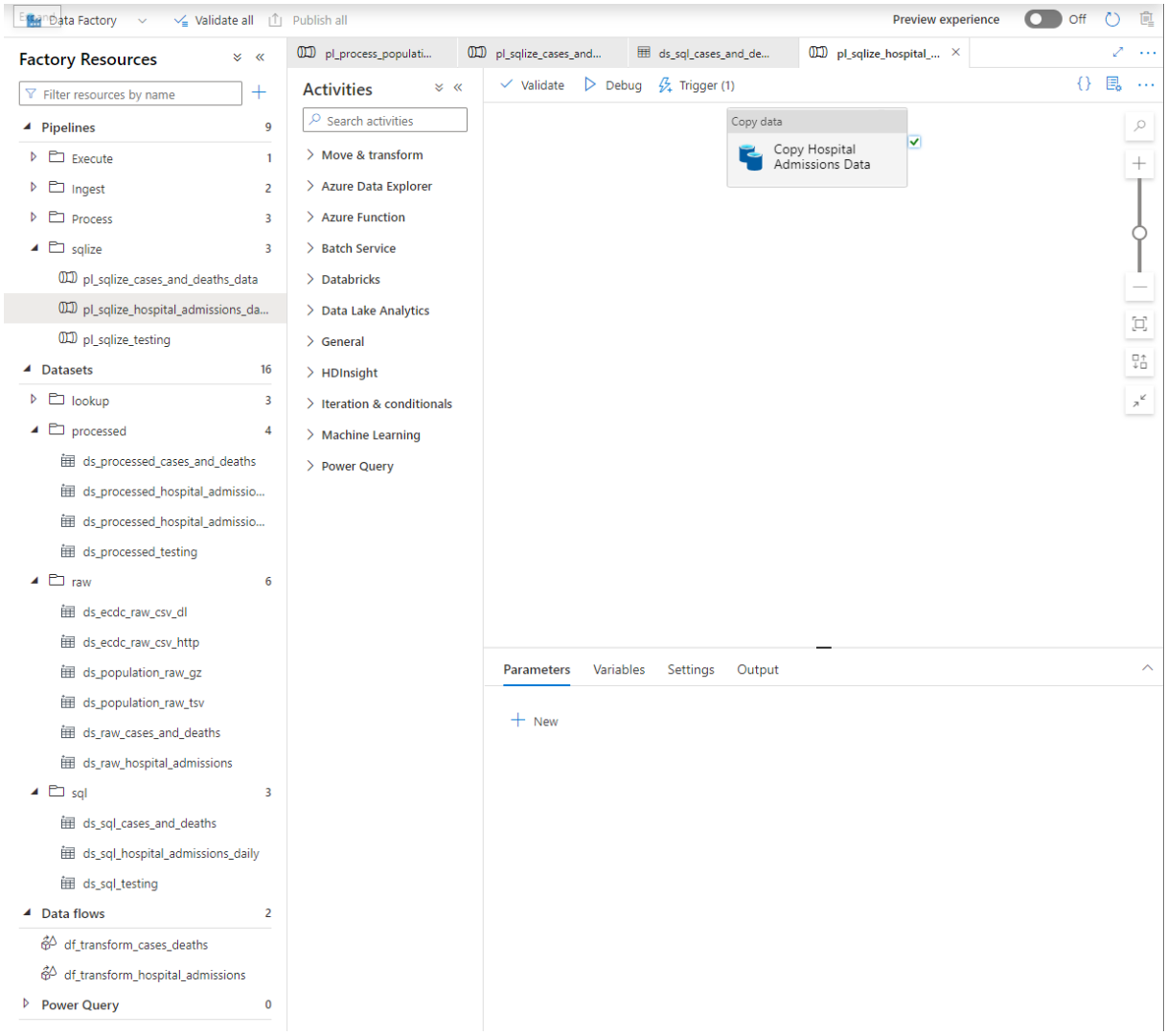
Step 3 - Create sink dataset



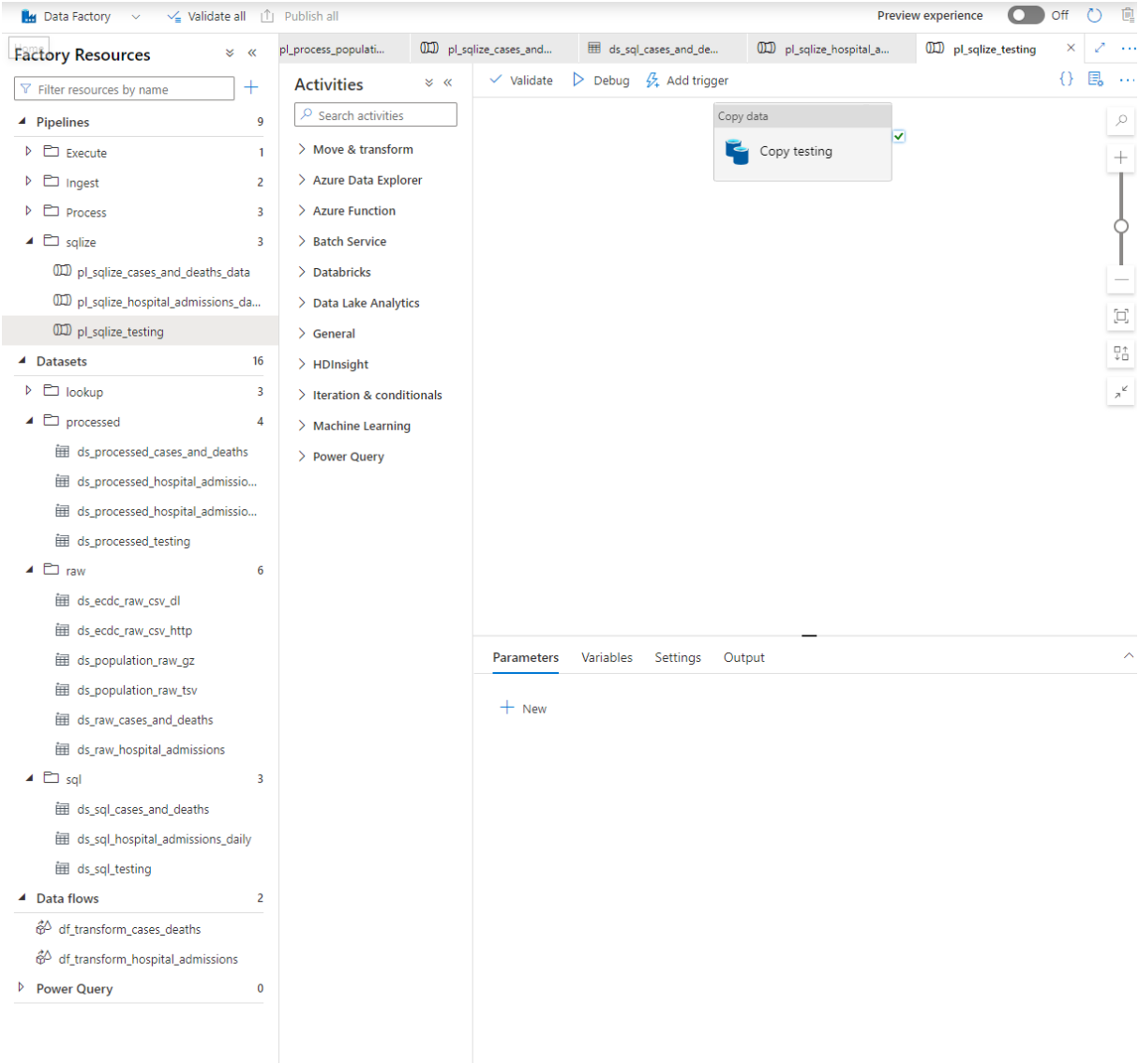
Step 4 – Create pipeline



Step 5 – Create pipeline hospital admission data

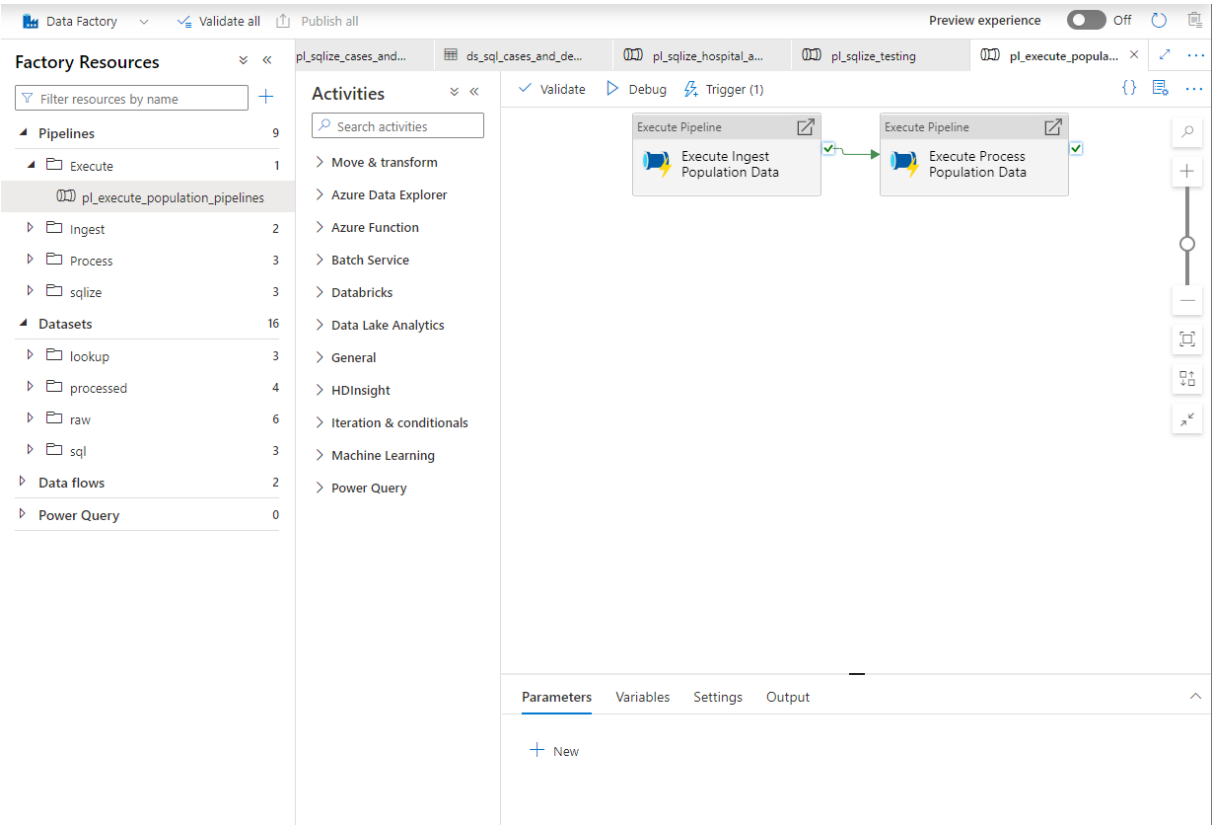


Step 6 - Create pipeline testing data



6. Data orchestration – Making pipelines production ready

Step 1 – Build pipeline



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 abbonement (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 *

[x] Blob created [ ] Blob deleted

Ignore empty blobs *

(●) Yes ( ) No

Annotations

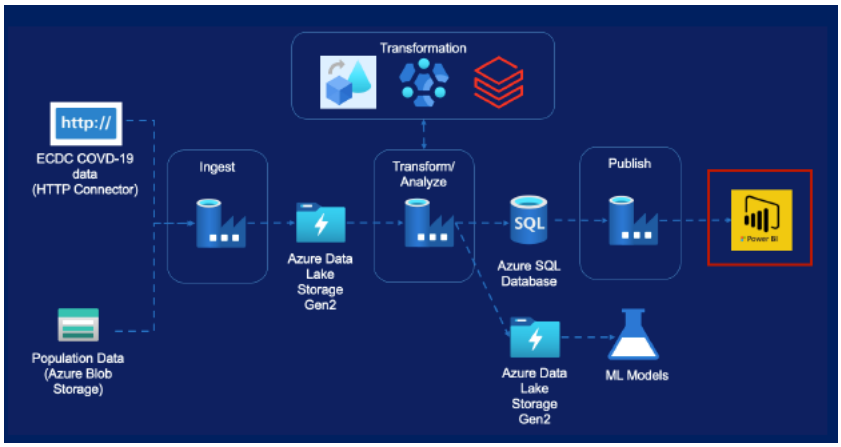
+ New

Status

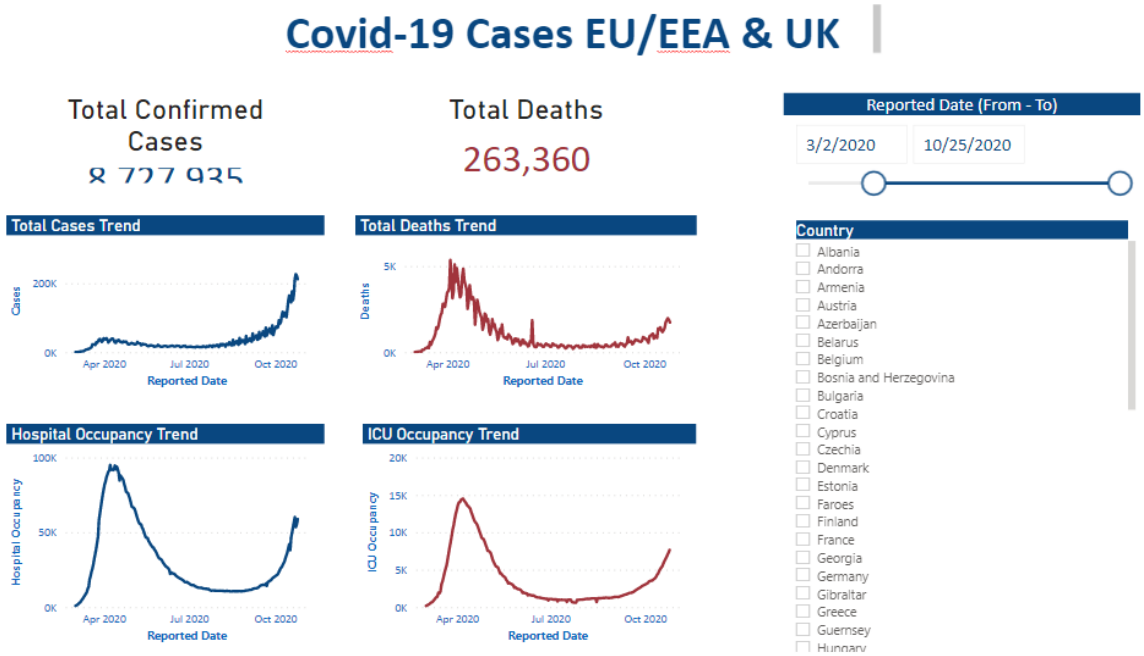
( ) Started (●) Stopped

7. Data reporting

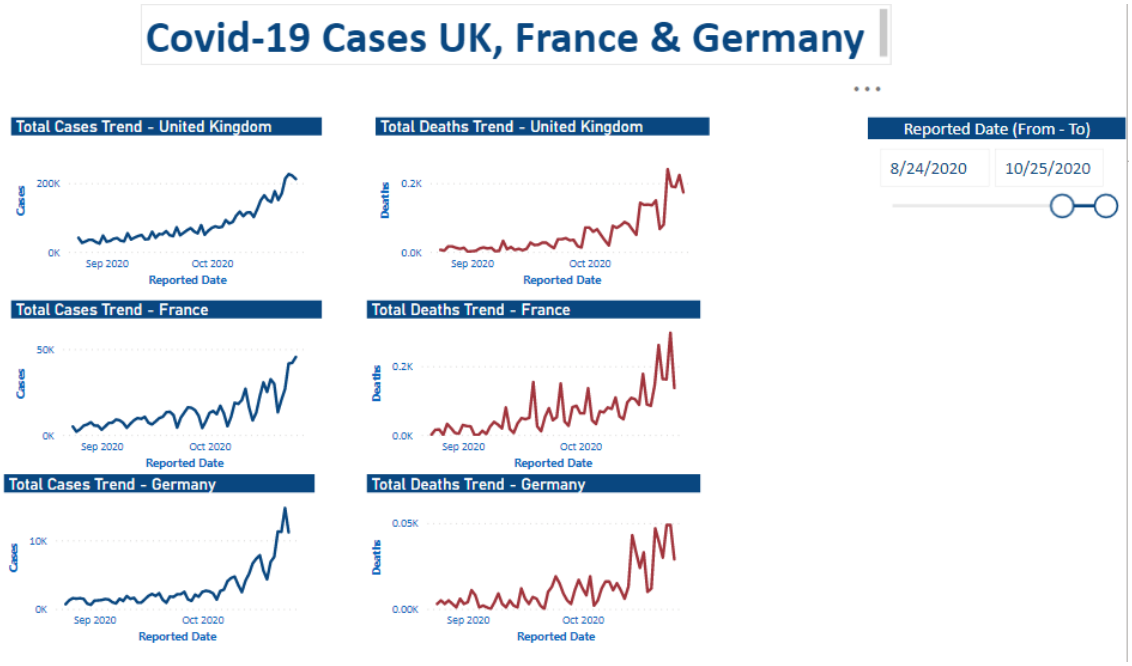
Step 0 – Reporting overview in Power BI



Step 1 – Create report 1 / 2

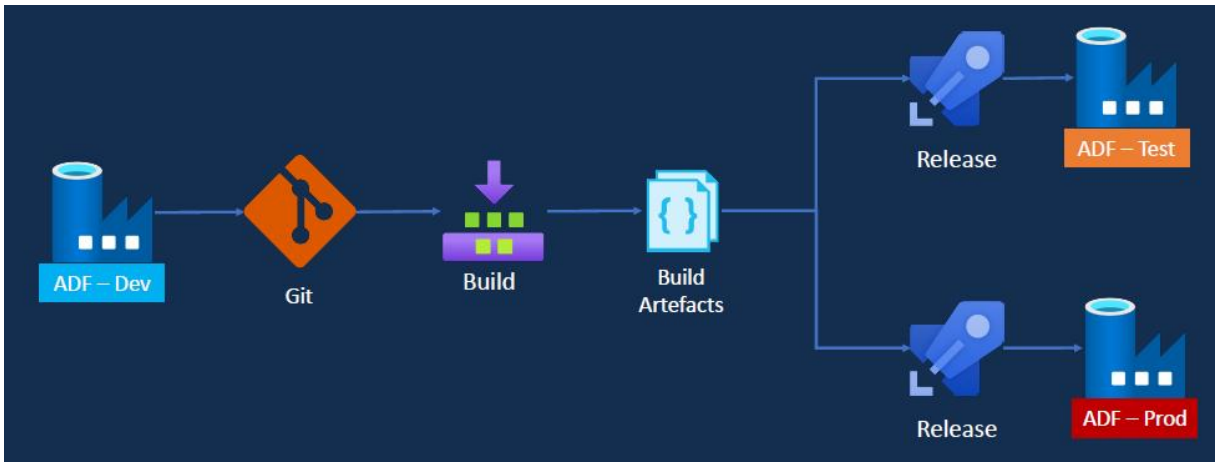


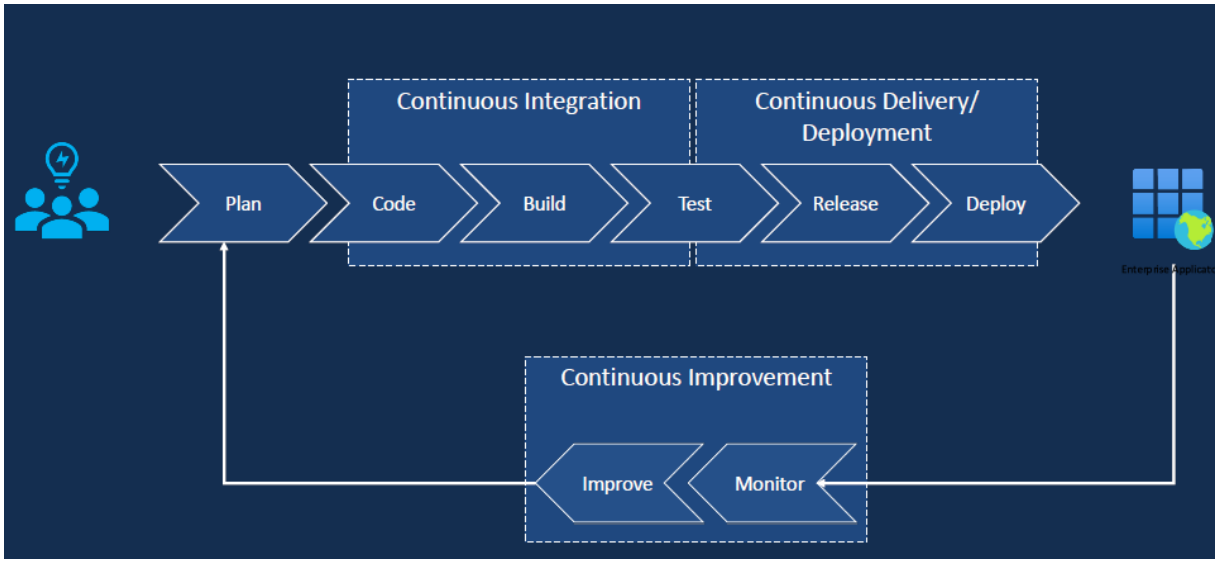
Step 1 – Create report 2 / 2



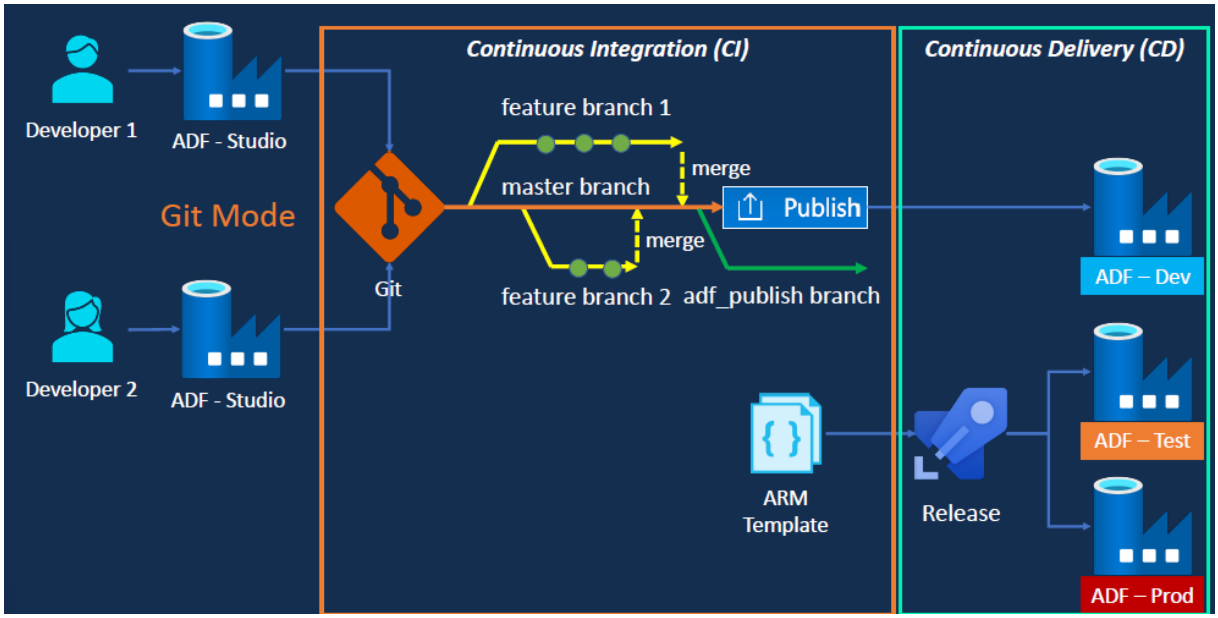
8. CI / CD

Step 0 – CI / CD overview

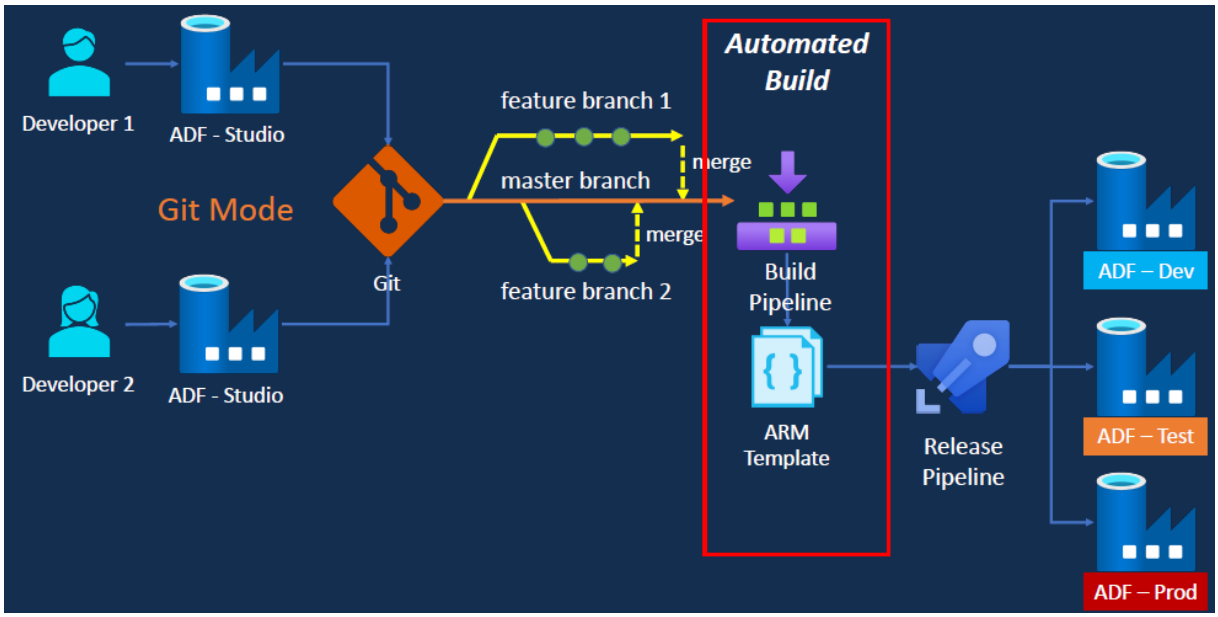




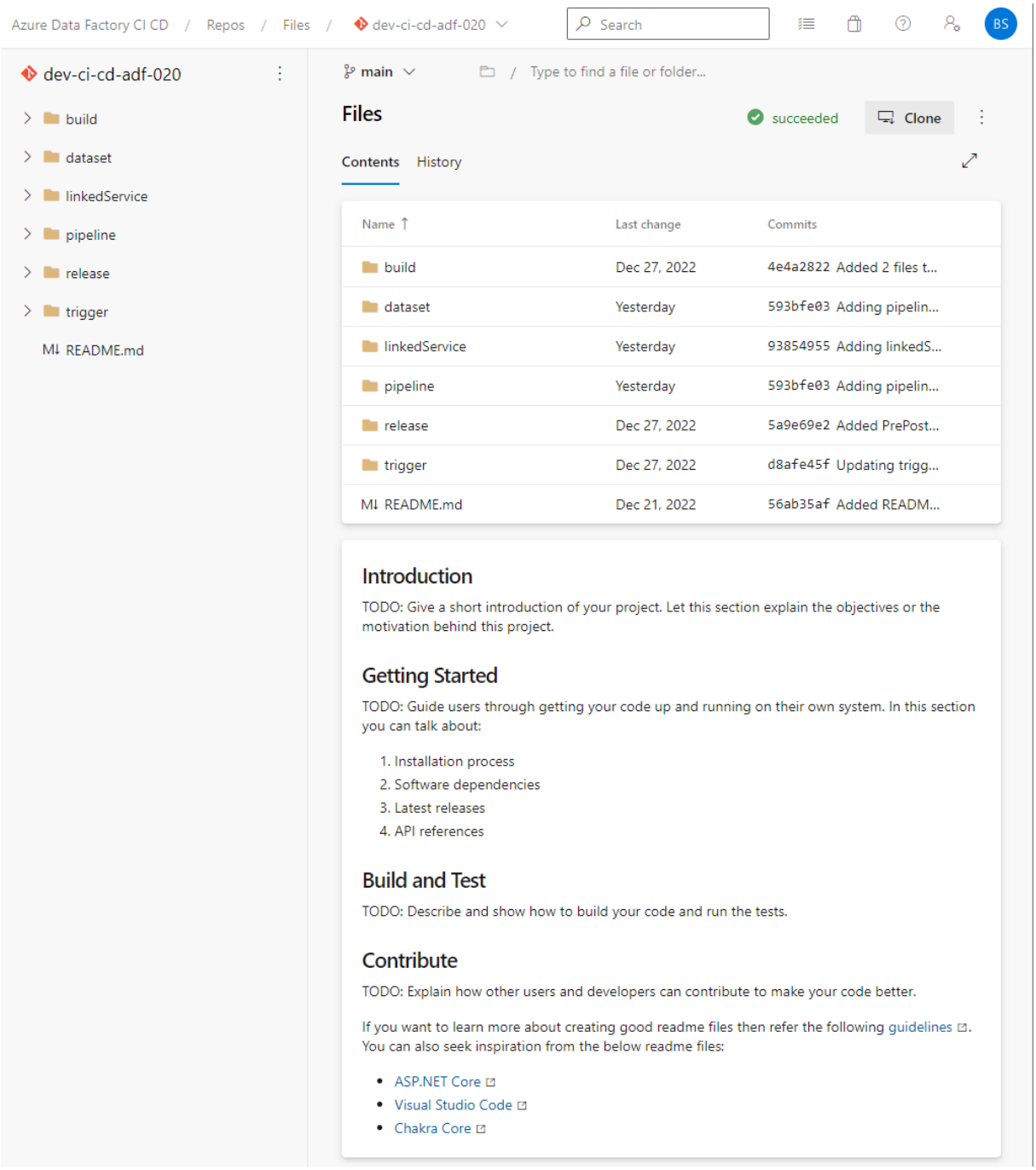
Step 0 – Git configuration within DevOps option 1



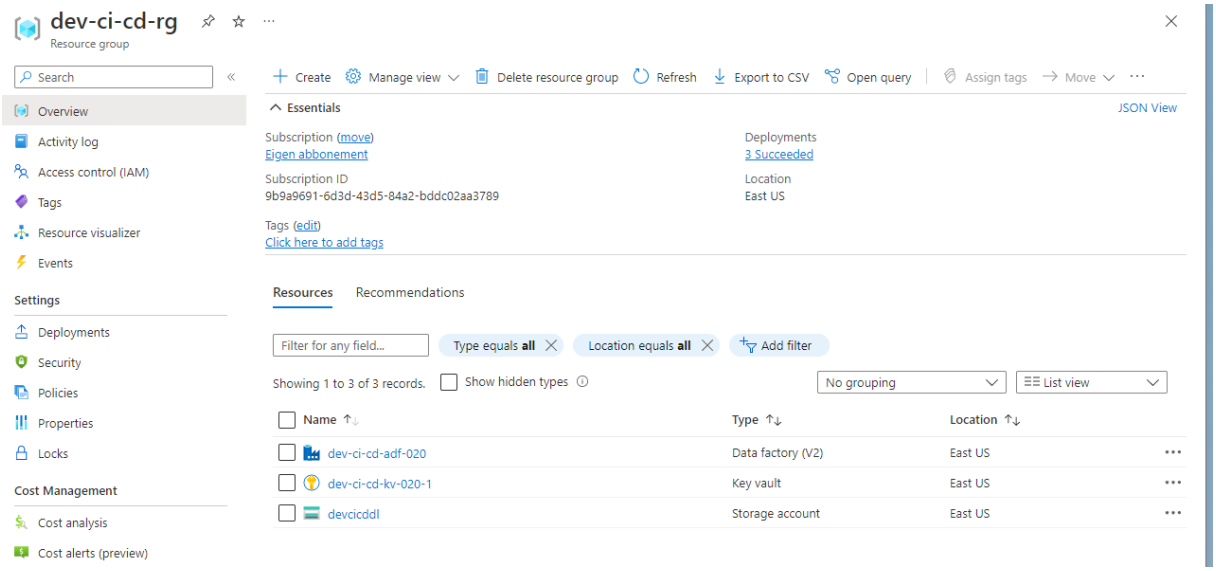
Step 0 – Git configuration within DevOps option 2



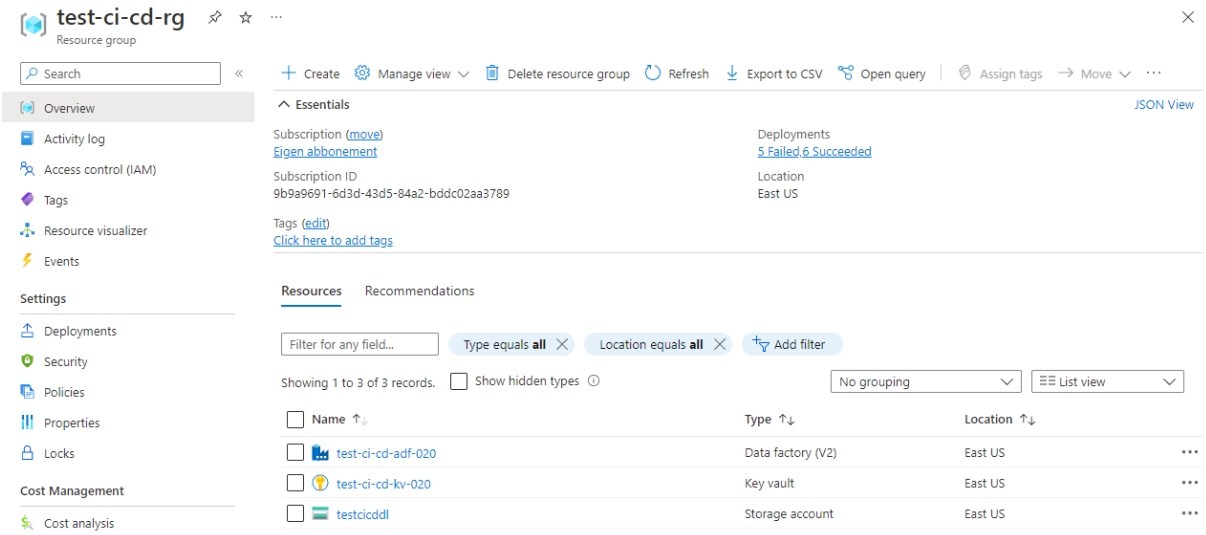
Step 1 – Create git repo



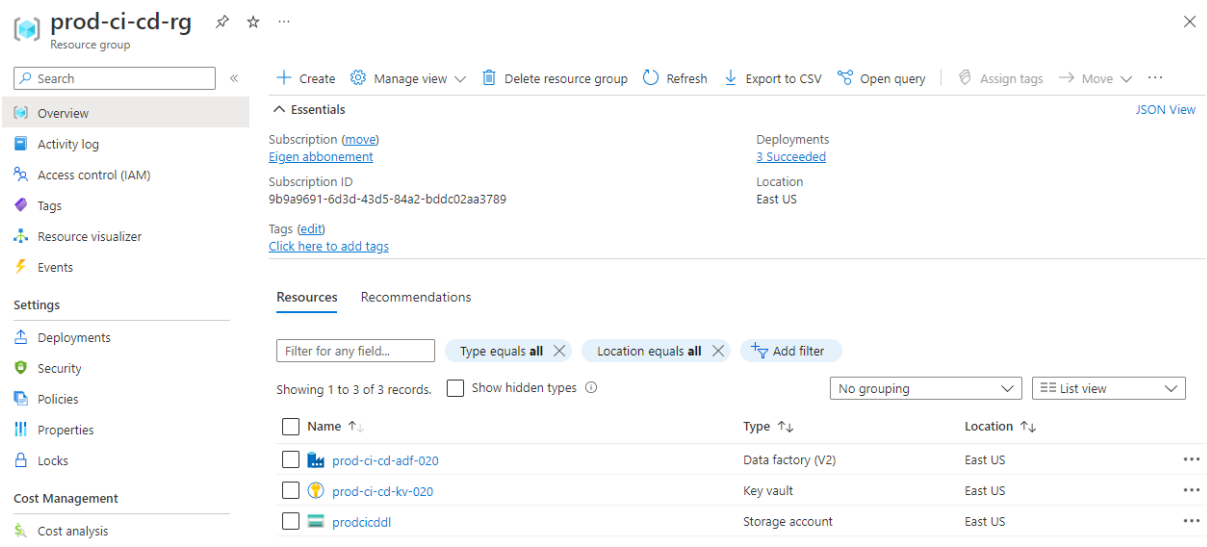
Step 2 – Create Tools for azure (DEV)



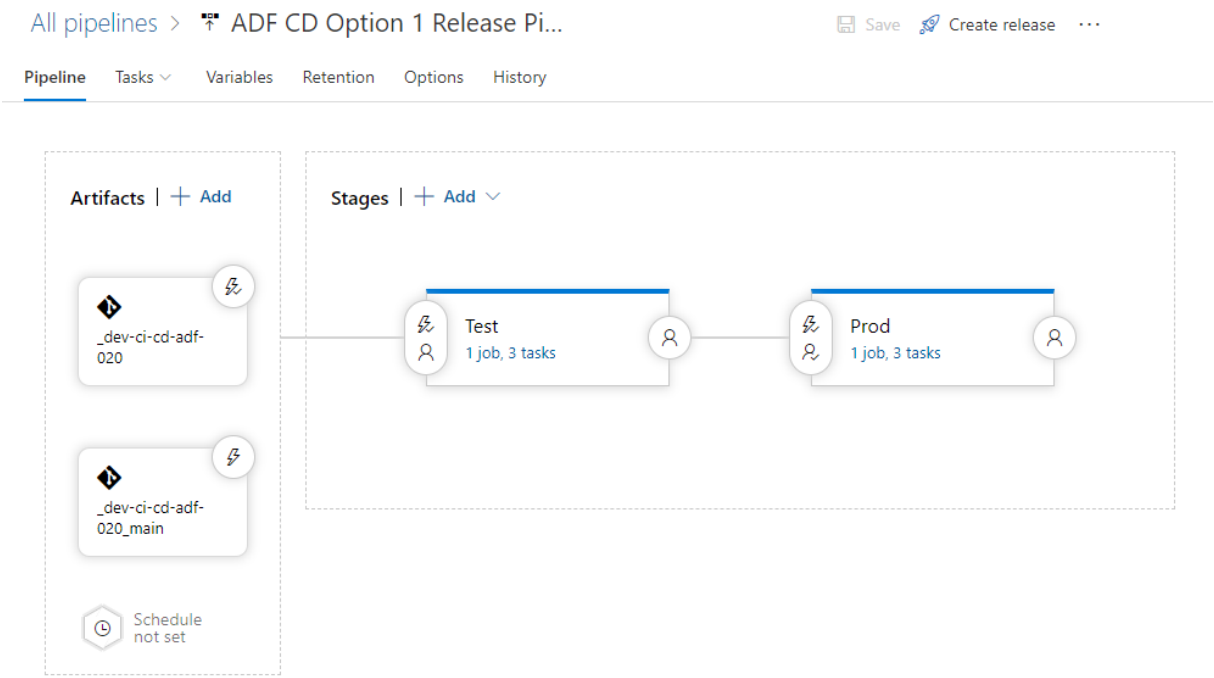
Step 2 – Create Tools for azure (TEST)



Step 2 – Create Tools for azure (PROD)



Step 3 - Create release pipeline option 1



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

```yaml
# File Name: adf-ci-option-2-build-pipeline.yml
# Purpose: Build pipeline for the development Azure Data Factory
#          Automatically triggered on completion of a pull request and
#          Validates the Data Factory resources
#          Generates the ARM template and publishes for consumption via
# Build Pipeline Name: ADF CI Option 2 Build Pipeline

trigger:
- main

pool:
  vmImage: ubuntu-latest

variables:
  subscriptionId: '9b9a9691-6d3d-43d5-84a2-bddc02aa3789'  # Use your sub
  resourceGroup: 'dev-ci-cd-rg'                           # Use the reso
  dataFactory: 'dev-ci-cd-adf-020'                        # Use your dev
  PackageFolder: 'build'                                  # Use the GIT
  adfRootFolder: ''                                       # Use the GIT

steps:

# Installs Node
Settings
- task: NodeTool@0
  inputs:
    versionSpec: '14.x'
  displayName: 'Install Node.js'

# Installs the npm packages saved in your package.json file in the build
Settings
- task: Npm@1
  inputs:
    command: 'install'
    workingDir: '$(Build.Repository.LocalPath)/$(packageFolder)' #replac
    verbose: true
  displayName: 'Install npm packages'

# Validates all of the Data Factory resources in the repository. You'll
Settings
- task: Npm@1
  inputs:
    command: 'custom'
    workingDir: '$(Build.Repository.LocalPath)/build' #replace with the
    customCommand: 'run build validate $(Build.Repository.LocalPath)/$(a
  displayName: 'Validate Data Factory Resources'

# Generate the ARM template into the destination folder, which is the sa
Settings
- task: Npm@1
  inputs:
    command: 'custom'
    workingDir: '$(Build.Repository.LocalPath)/build' #replace with the
    customCommand: 'run build export $(Build.Repository.LocalPath)/$(adf
  displayName: 'Generate ARM template'

# Publish the artifact to be used as a source for a release pipeline.
Settings
```

Tasks

Search 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 subscr...
- 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