PROFINIT
NÁSKOK DÍKY ZNALOSTEM

Azure practice

Pipeline for work with C19 opendata

Contents

1 Project description

1.1 Goals

The aim of the project is to create a pipeline in Azure Data Factory using Azure components, which after running will download the actual data about testing on Covid-19 from portal of the Ministry of Health, then perform transformations of the dataset (adding region + district names, removing columns...) and the resulting dataset will be passed to Azure Databricks notebook, where the dataset will be further processed. The resulting notebook should contain simple data analysis and visualizations.

1.2 Used technologies

The main technologies used in the project are **Azure Data Factory**, **Azure Databricks** and **Azure Storage account**. The goal is to get a feel for the basics of these technologies and try to connect them through a pipeline in Azure Data Factory.

2 Account Creation

Before starting: it is recommended to do the whole project in AJ, i.e. switch the default language in the settings.

1. Create an account via this link: https://azure.microsoft.com/en-us/free/students/. **Use your school email!**
2. After creating your account, you will be taken to Education Hub.
3. Select the option that you want to access student benefits and complete the registration.
4. On the Subscriptions page, make sure you have created a *Azure for Students* subscription.

3 Resource Preparation

`Resource` is a label used for instances of individual services, components that Azure offers, such as VMs, Web Application, Databricks, Storage account, Azure Data Factory, Azure SQL Database ...

3.1 Create Resource group

`Resource groups` are used to logically group `resource`. Good organization of `resource` then makes it easier to work; for example, when setting access and various properties common to all `resource` within a single `resource group`. For this project, we will create a single `resource group` in which we will place all the `resource` used.

1. On the Azure portal, click on the side menu and select the *resource groups* tab.
2. Create a new `resource group`.
3. Select a suitable name and nearby region, confirm it.

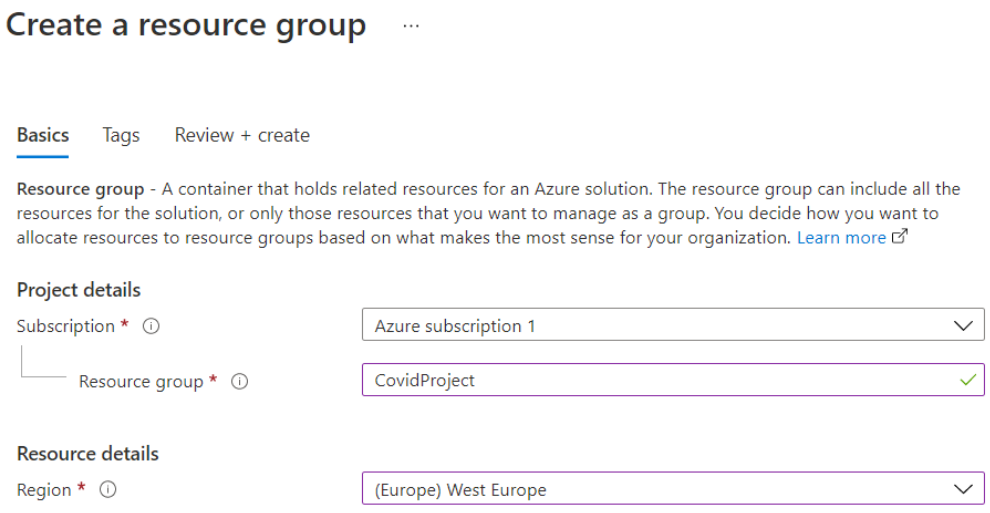

Figure 1: Creating resource group

3.2 Data Factory Creation

1. In the menu, select *Create resource* (first option).
2. Locate Data Factory `resource` and click *Create*.
3. Fill in the basic information, select V2 as the version. Select the `resource group` as the one created in the previous step.

4. In the Git configuration tab, check *Configure Git later*.
5. **Review + Create**.

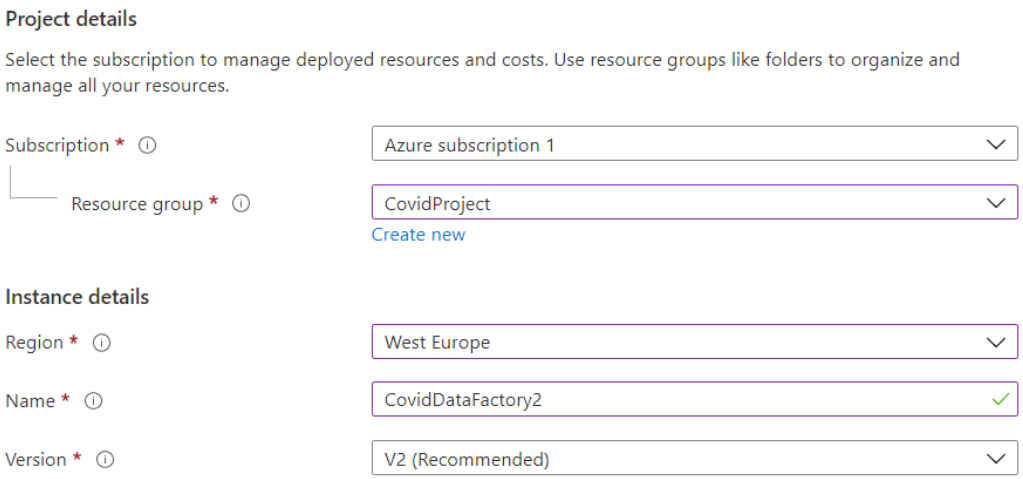

Figure 2: Creating Data Factory

3.3 Creating Storage Account

1. Similar to the previous step, create a new `resource` - Storage Account.
2. Select *Standard* performance and *LRS* redundancy.
3. **Review + Create**.

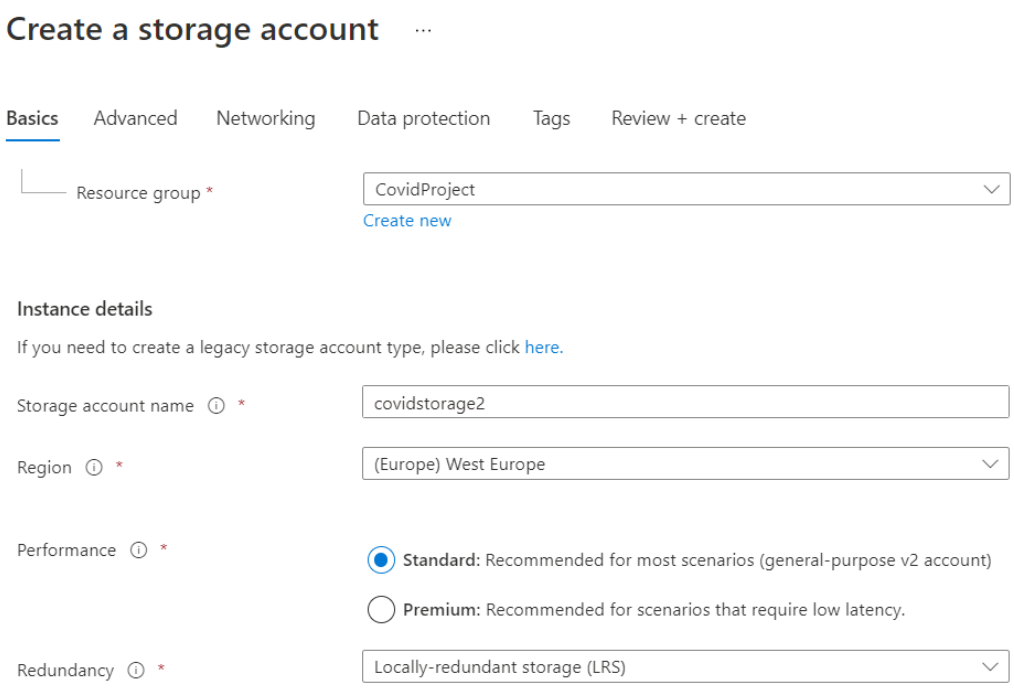

Figure 3: Creating Storage Account

3.4 Creating Azure Databricks

1. In the same way as in the previous steps, create Azure Databricks `resource`. Again, there is no need to change **almost** any settings - set **Pricing tier as Trial**.

2. Make sure you have selected the correct `resource group`.

3.5 Opening Azure Data Factory Studio

In ADF Studio we will create most of the project logic.

Process of opening ADF Studio:

1. Find the Data Factory `resource` you created in the Azure Portal.

2. Click *Open Azure Data Factory Studio.* (Image 4)

Figure 4: ADF Studio Opening

4 Pipeline

A data pipeline involves working with the Azure Data Factory Pipeline logical grouping of activities that work together on a task. In this task, we'll try a very simplified pipeline. **In our case, these will be the following activities:**

1. **Download data**
   - Create a Container inside Account Storage storage.
   - Download covid data.
   - Upload to Container.
2. **Data Flow**
   - Used to transform data using a simple GUI.
   - All optimizations happen automatically in the background.
   - Supports a wide range of operations (e.g. join, sort, filter, select...).
3. **Databricks Notebook**
   - Activity used to start a specific Databricks notebook.

4.1 Creating datasets in Storage

The dataset on the covid portal does not contain names of regions and districts, only their identifiers. The purpose of our data flow will be to join the datasets by identifiers so that for each record we also have the name of the county and district. We will then remove unnecessary columns.

Data file download:

- Download prepared population data (czech_population.csv) in the regions of the Czech Republic and their NUTS/LAU codes (CZ AREA CODES.xlsx).

Uploading files to storage:

1. In the Azure Portal, go to your Storage Account and create a new container *static-data* via **Data storage** → **Containers**. Leave the Access level as *private*.

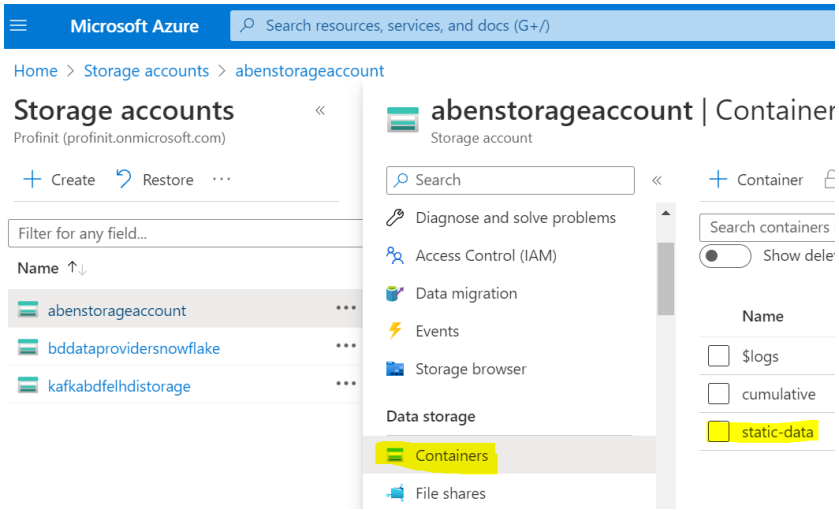

Figure 5: Static Container

2. Upload the downloaded files (csv population + excel file with edges) into the container.

The Covid portal contains many datasets. In this exercise we will work with the dataset **COVID-19: Total (cumulative) number of tests performed by regions and districts of the Czech Republic**.

1. Download the data from **https://onemocneni-aktualne.mzcr.cz/api/v2/covid-19/kraj-okres-testy.csv**.
2. Create in your Blob Container, for example **source-data**, 6.
3. Use Upload to upload the downloaded CSV.

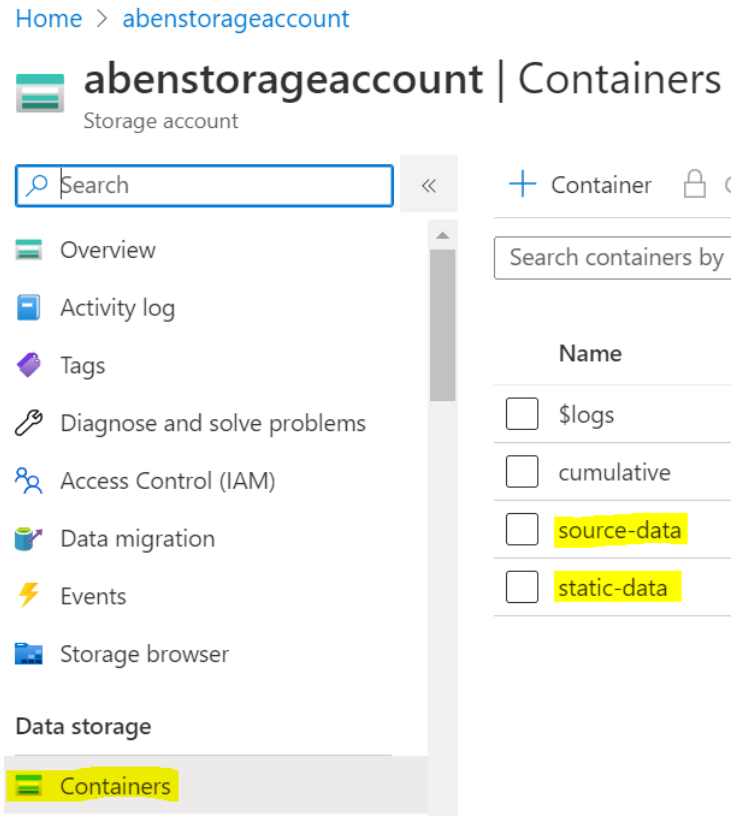

Figure 6: ADF Studio Opening

Excel file with county/district names will be used in Data Flow. So we need to create new Datasets in ADF Studio, see 4.2.

4.2 Data Flow

As already mentioned, data flows are used to transform and work with data.

1. Via **Author** → **Datasets** → **New dataset** create a new dataset.
2. Select Blob Storage, Excel dataset type and specify the file path.

3. The rest of the settings can be seen in Figure 7. Note the **Sheet name** entry to specify which Sheet to use for this dataset - one dataset can be max 1 excel page.

4. Create **two datasets** - one for districts (abbreviation LAU), the other for counties (NUTS). Be careful to select the correct sheet.

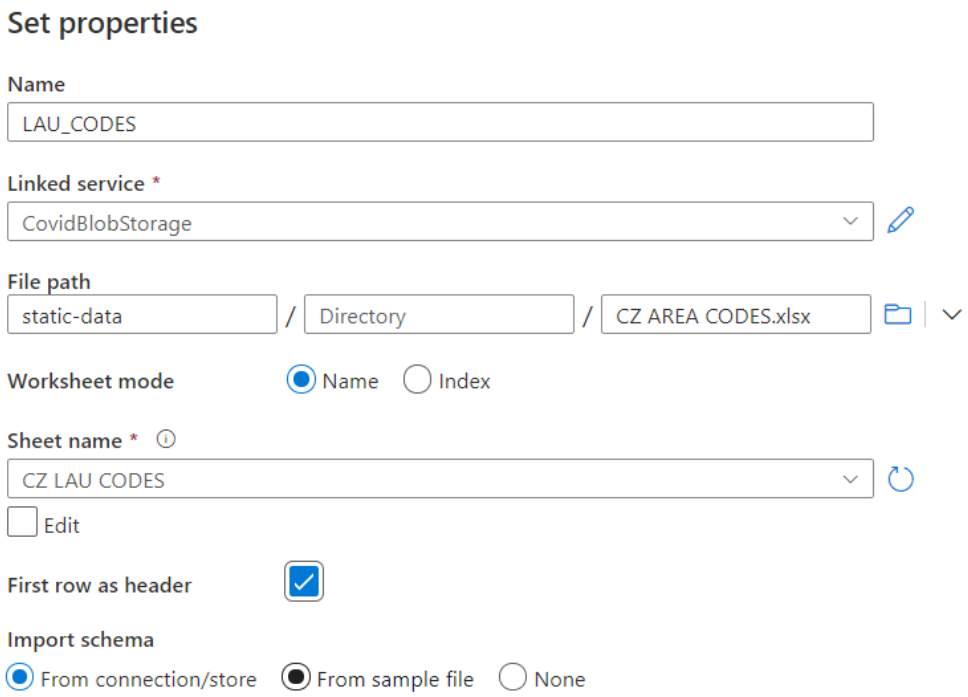

Figure 7: Creating dataset from Excel sheet

5. Import **From connection** from Schema.

6. Before proceeding, make sure you have two datasets - one for each sheet.

7. Create another **Dataset** as an Azure Blob Storage of type CSV.

8. Choose a path to the **source-dataset** Container created earlier with the cumulative data, See 8.

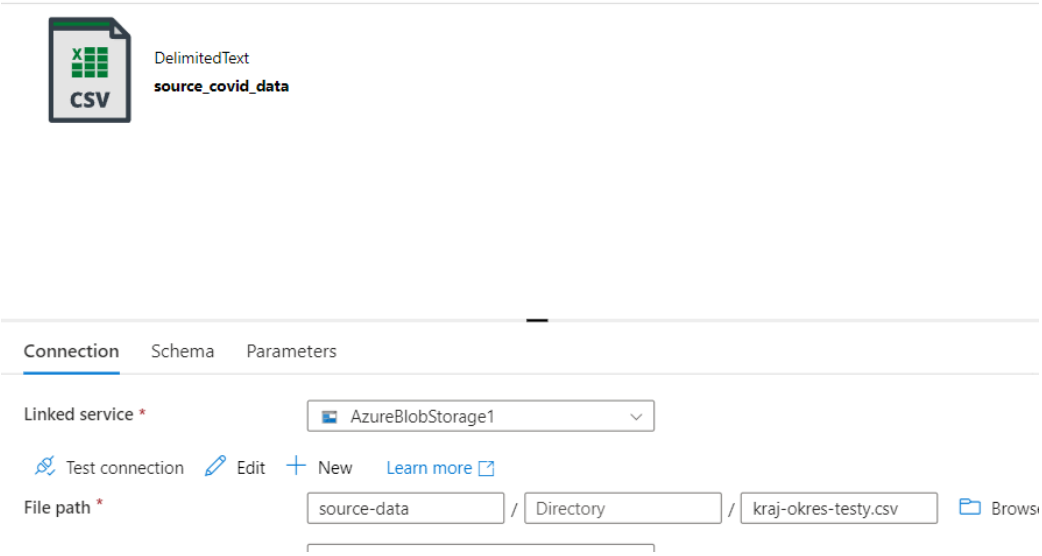

Figure 8: Creating dataset from csv

4.2.1 Downloading schema

Before we use a dataset (about tests on Covid-19) in Data Flow, we need to make sure that ADF knows its schema.

1. Navigate to the Dataset (in ADF Studio) that represents the file in Blob Storage.
2. In the schema tab, select **Import schema**.
3. If you select **From connection/store**, ADF will attempt to import the schema from the path that is specified in the dataset. Our dataset has a parameterized filename, and ADF will try to find the file with the name you specified in the default value. If you have not manually uploaded such a file to your storage, it will fail. However, it should all happen automatically. There are multiple possibilities if an error occurs:
    - then set the default value to a file name you already have in storage - e.g. the file you downloaded in step .
    - manually upload the file to storage with the name you set as the default value

- not set a default value and, as with the first option, enter the name of a file you have already downloaded, but not as a default value, but when prompted when you click import

4. Check that you managed to upload the schema. You don't need to look at the column types now - the important thing is to see what and how many there are.

4.2.2 Create Data Flow and add resources

Once we have the source datasets ready, we can move on to creating the Data Flow.

1. Click on **Author** → **Data flows** → **New data flow** to create a new Data flow.

2. Create three source datasets - LAU, NUTS and downloaded test data.

3. Just click **Add source** and select the dataset. The result should look similar to the image 9.

4. Make sure you see the number of columns for each source. If it shows you 0 columns, it means the dataset has no schema defined. In this case, load the schema (see 4.2.1).

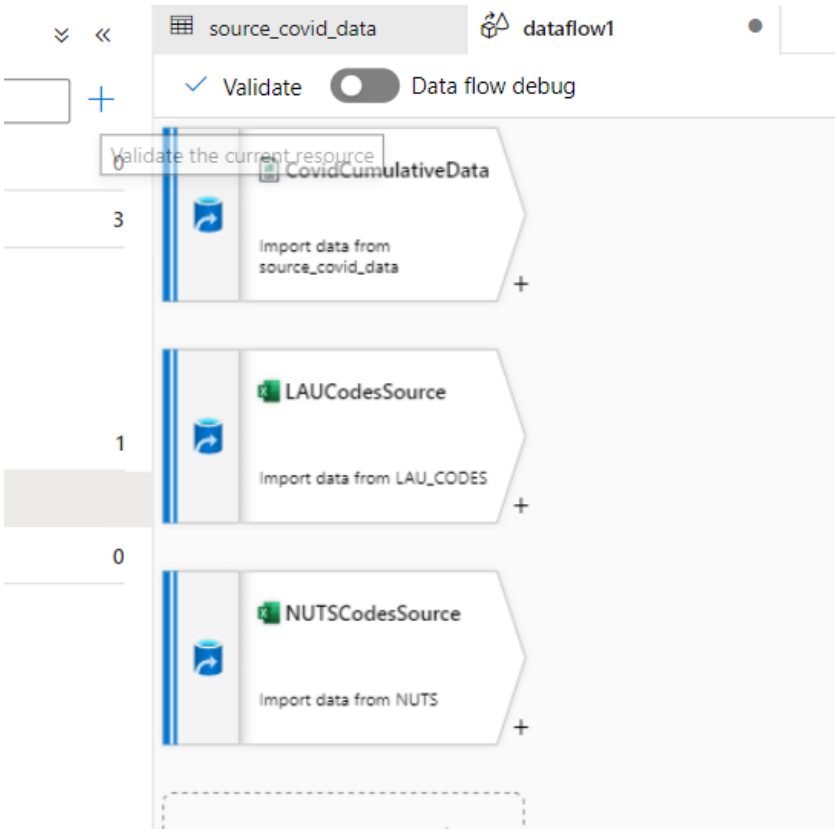

Figure 9: Data Flow source datasets

4.2.3 Adding transformations

By clicking on + next to the source dataset, a new transformation can be added and these can be chained. Click on + to see what operations can be performed on the data.

In our Data flow we will mainly need **join** and **select**. **Select** allows you to rename and delete columns. To delete a column, check the box next to the name and click on the trash can icon. To reset the settings, press **Reset**.

1. Use the `join` operation to join the Covid portal dataset with Excel datasets so that each record has a county and district name. You will need to use `join` twice. (As a reminder: LAU corresponds to NUTS4, the county code is NUTS3, sometimes shortened to NUTS only).

2. Subsequent operations are not necessary, but will speed up the JOIN process. Use the `select` operation to delete duplicate (e.g. `kraj_nuts_kod` contains the same information as the `CZ-NUTS3` column) and unnecessary columns. From the `LAUCODES` (NUTS4) dataset we do not need

`kod_okresu` or `nazev_kraje`, you can delete the columns before the join. Leave the columns `CZ-NUTS3` and `CZ-NUTS4`.

3. The ending item of the Data flow is always the `Sink` dataset operation - the destination where the transformed data should be saved, named as `SaveJoinedAndCleaned` in Figure 11.

   (a) Create a new dataset, in a similar way to the dataset where we store the downloaded data from the Covid portal. Again, we will use Blob Storage and save the dataset as a CSV

   (b) **Warning**, the dataset we use as Sink cannot have a specified filename. Set Directory as dynamic value `@dataset().directory_name`. Data can be split into multiple partitions, if we required the result of a Sink operation to be in a single file, we would have to set this explicitly in the Data flow.

   (c) However, this is not necessary, we will sort the data into partitions that have the pipeline run time in the name. **Create a parameter** *directory_name* and then use that in the *directory* field (the default value doesn't matter - we will always use the dataset with a specific value passed to it from the pipeline). Leave the *filename* field empty. See Figure 10.

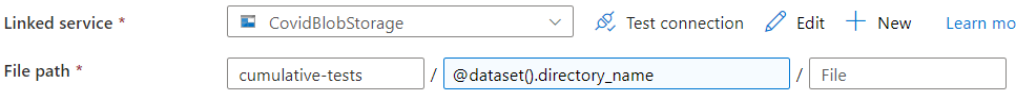

Figure 10: Path settings for the Sink dataset used in the Data flow.

   (d) You can either create a new container or use the same container where you store the downloaded datasets from the Covid portal.

4. Use the newly created dataset as a Sink dataset in your data flow.

5. The resulting data flow should look like Figure 11.

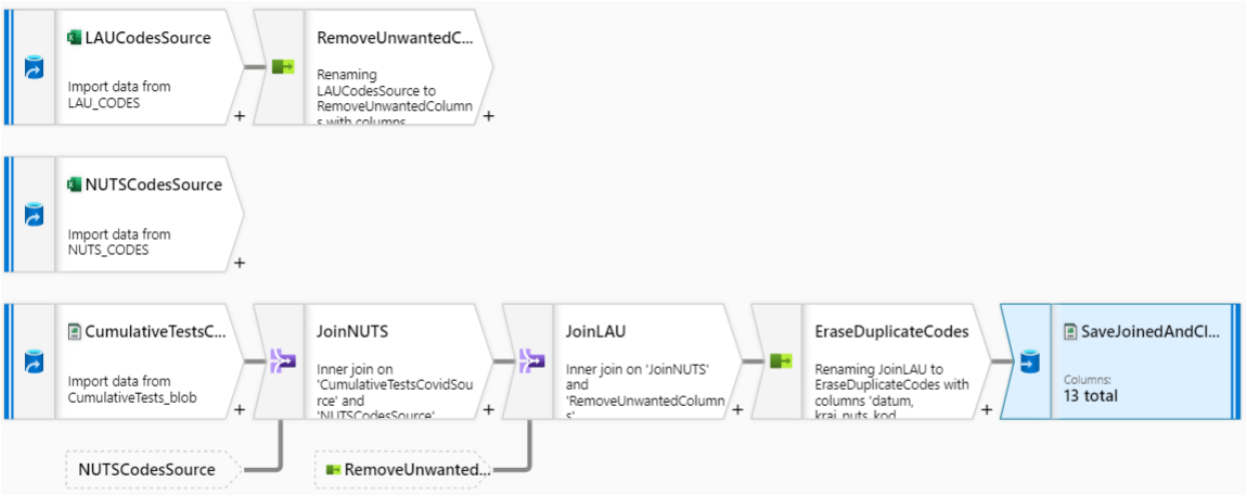

Figure 11: This is what the resulting Data flow may look like.

6. Save your changes by clicking **Publish all** → **Publish**.

4.3 Adding Data flow activity to pipeline

1. Create your pipeline and insert a Data flow activity (in the Move & Transform category) and name it appropriately.

2. In the **Settings** tab, under **Data flow**, select the Data flow you created in the previous step.

3. After selecting a specific data flow in **Settings**, fields for its dataset parameters will be added. In our case it is: *directory_name* (the name of the directory where to save the result).

   (a) **directory_name**: Construct a name from a suitable name and the pipeline start time, e.g.

   ```
   {@concat('covid_tests_joined',
           '-',replace(pipeline().TriggerTime,':','-'))}.
   ```

   - **if you copy code directly from the document, make sure it is copied correctly (i.e. quotes, underscores, no newline, no extra spaces, etc.)**

4. Save the changes by clicking on **Publish all**. To test, try running the pipeline using **Trigger now**. When the pipeline runs, see 12, check your storage to see if it contains a directory with the joined data, see 13.

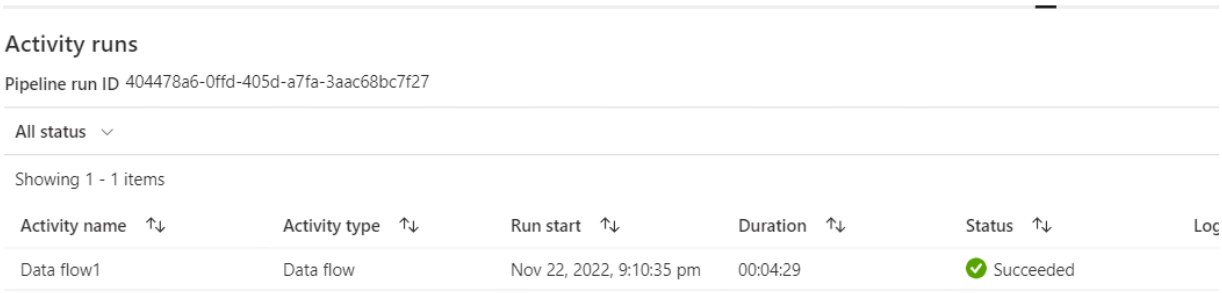

Figure 12: This is what a successful dataflow run might look like.

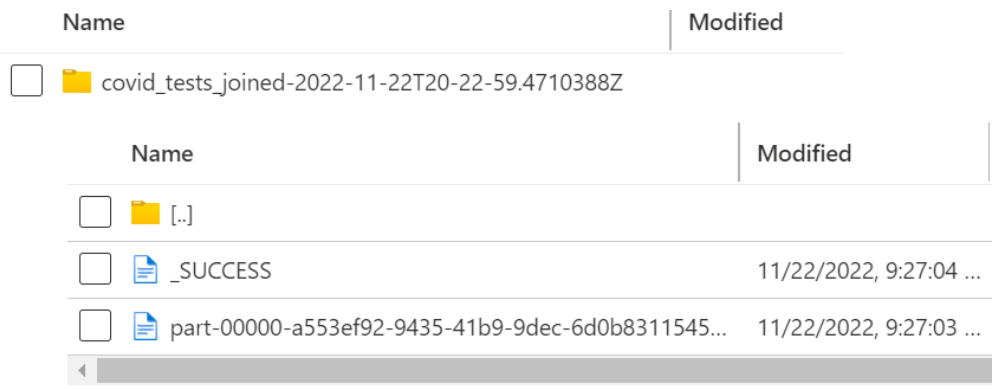

Figure 13: Joined data in cumulative containment.

5 Databricks

In the last part of the project we will use the Databrick notebook - a simple web-based document interface that contains executable code, possibly visualizations and additional comments, more here.

First, we'll create a new notebook and cluster, then link our Databricks and Data Factory `resource`, and then add a parameterized Databricks note-

book activity to the pipeline. Then most of the work will be done directly in the notebook.

5.1 Work environment opening

- Find your Databricks `resource` in the Azure Portal and then open the workspace by clicking **Launch Workspace**.

5.2 Cluster Creation

In order to perform work, calculations (e.g. commands from Databricks laptop), you need to provide computing resources - **Cluster**. Azure Databricks distinguishes 2 types of clusters:

- **all-purpose cluster**: it is created once, turned off and on on command, or when a set time has elapsed
- **job cluster**: creates automatically according to the specified parameters only when needed, e.g. when a trigger arrives

For our purposes, since we will want to test the notebook while it is being built, we will use **all-purpose cluster**. More about clusters.

1. Navigate to **Compute** → + **Create cluster**.
2. Create a cluster with the settings in Figure 14

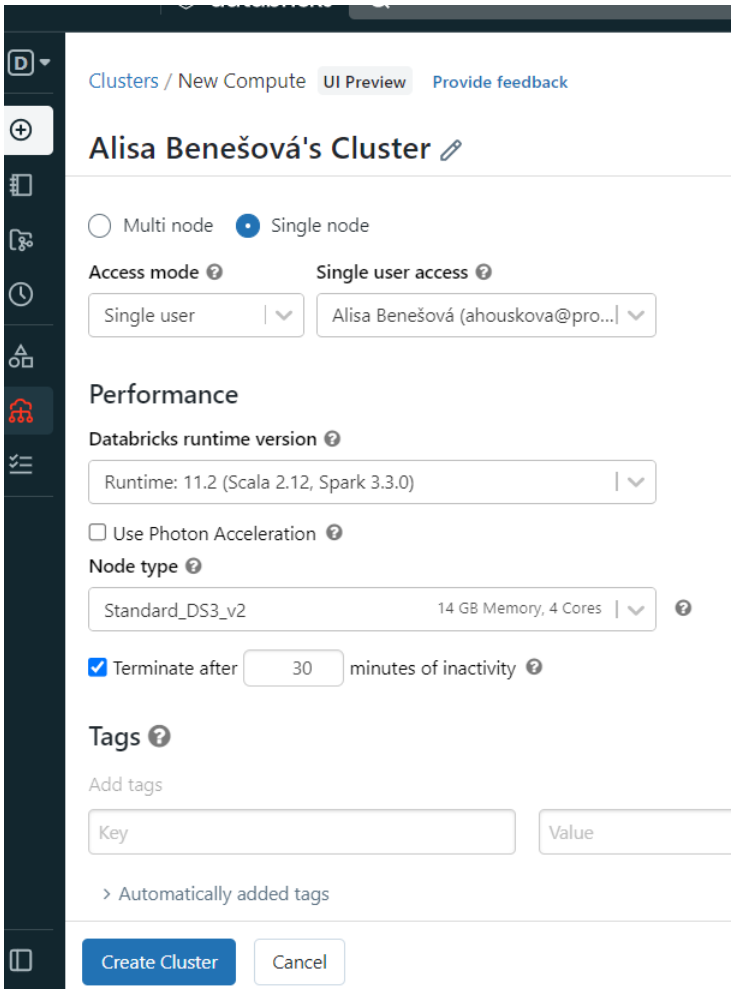

Figure 14: Cluster Configuration

5.3 Import notebook

1. Select **Workspace → Users → Your Account → Import → URL** from the menu and import your laptop from URL:https://github.com/mjanec/azure-practicetask-covid/blob/master/Covid%20Practice%20task%20template.dbc?raw=true

   In the top left drop-down menu, you can choose which cluster will execute commands from the laptop. Select your cluster - it should be running. If it is not running, start it.

2. *Not Mandatory step. You can clone the repository to your git provider and via personal access token connect to Databricks. You must use menu Repos. How to generate tokens, see github, gitlab. Choose* ***Add***

Repos *and provide your URL, pull.*

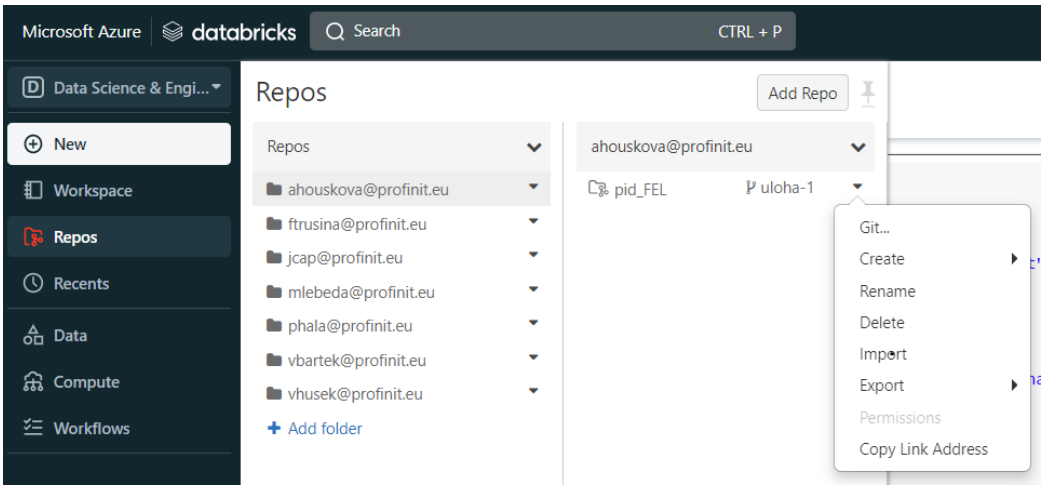

Figure 15: Repos download

5.4 Adding Azure Databricks to Connected Services

In order to run notebooks in a specific ADB `resource` through Azure Data Factory, you first need to connect these components. We create a *access token* in the ADB, which we then use to create a new linked service in the ADF.

Generate access token:

1. In the Databricks side menu, navigate to **Settings** → **User settings** → **Generate New Token**.
2. Fill in the purpose of the token and its validity - you can leave the default value.
3. Copy the token to your clipboard or save it to a file. It cannot be viewed again.

With the token in the clipboard, return to **Azure Data Factory Studio** and create a new linked service:

1. **Manage** → **Linked service** → **New**.
2. In the **Compute** tab, select Azure Databricks.

3. Fill in the name and basic information, via **From Azure subscription** select your Databricks workspace.
4. Authentication type select token and insert your token from the clipboard.
5. Select **Existing interactive cluster** and select your cluster.
6. Image 16.

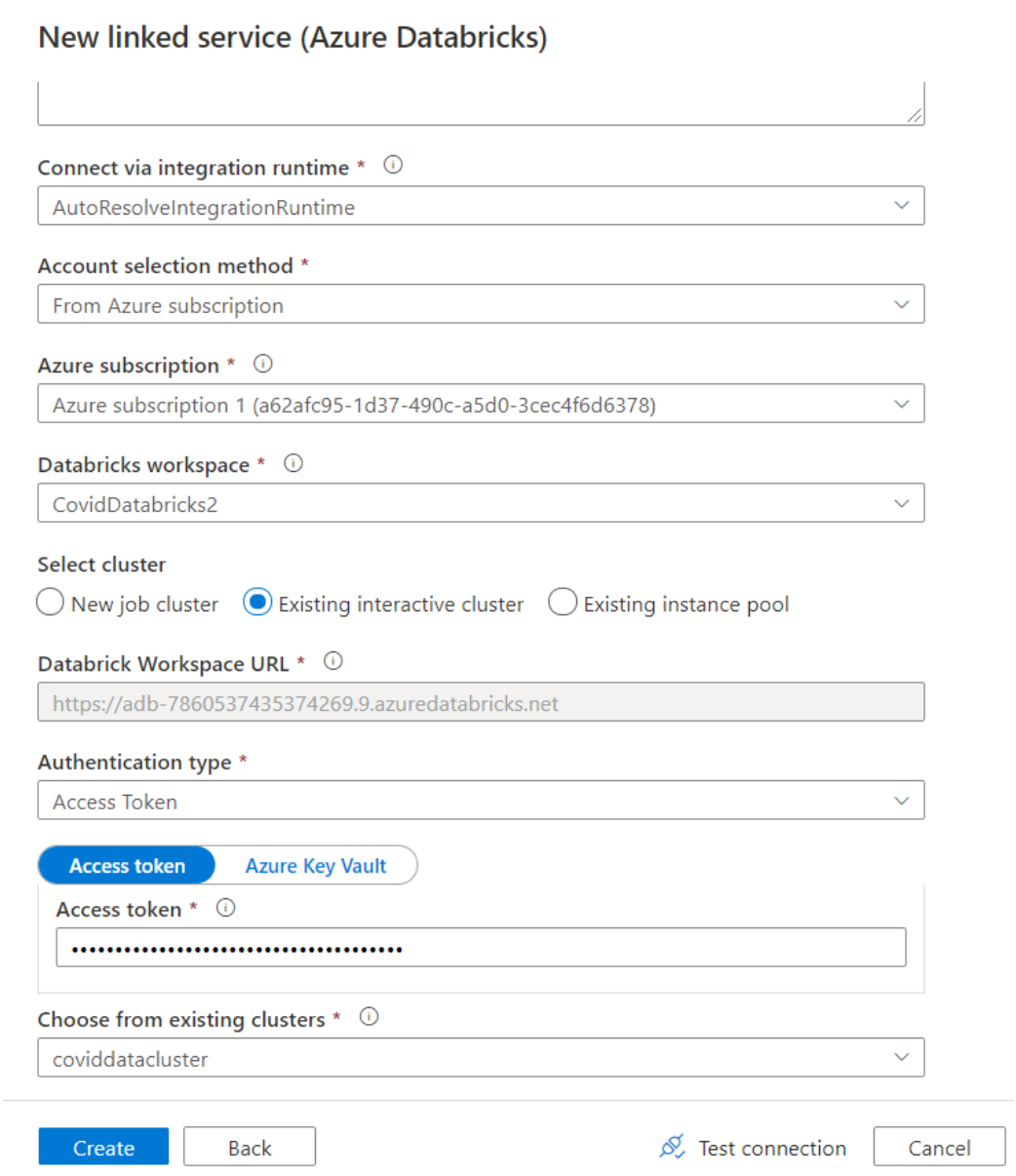

Figure 16: Example ADB linked service setup

5.5 Adding Azure Databricks notebook activity

After creating a notebook, we can add it as an activity to the Data Factory pipeline.

1. Add a new **Databricks** → **Notebook** activity to the pipeline.

2. Ensure (using the green arrow) that the activity starts only after the previous activity (data flow) has run successfully.
3. In the activity settings, select your Databricks *linked service*.
4. In the **Settings** tab of your activity, select the path to the notebook you want the activity to run.
5. On the **Settings** → **Base parameters** tab, set the parameters that will be passed to the laptop when it starts:
   - *storage_account_name*: The name of your Storage account.
   - *container*: The name of the container where you store the processed data (using data flow).
   - *directory*: The name of the directory where the data you are going to work with next is located - the value of this parameter must be the same as the value of the Sink parameter of the previous activity (we pass the address where the data was stored after the data flow ended), see 3a.
   - *storage_account_access_key*: the key you find in the Azure portal in your Storage account settings (**Security + networking** → **Access keys**) - use **key1**
     - You could also use **key2**, you can read why there are two keys here.

5.6 Show detailed view of the notebook

After you add the notebook activity to your pipeline, try to run it (via **Trigger now**).

1. Once started, go to **Monitor** → **Pipeline runs** and select the current run.
2. Click the **Details** button (see Figure 17) to display the **Run page url**.
3. Go to **run page** - the page containing the specific notebook run. You can look at the cell outputs or check (at the top of the page) if the correct parameters have been passed to the notebook.
4. Don't be scared that the notebook activity run will not be successful - the notebook is waiting to be worked out.

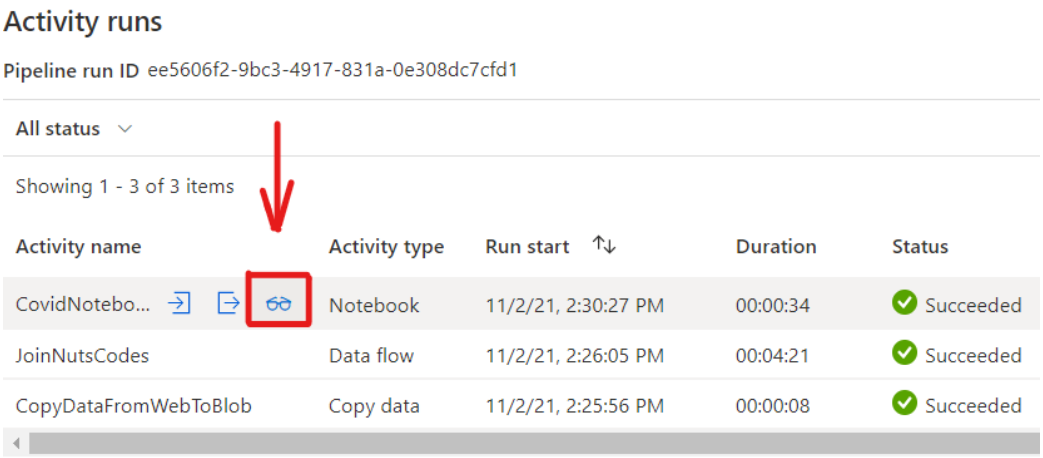

Figure 17: Display Notebook Run Details

If your notebook successfully starts with the correct parameters, go to your Azure Databricks workspace and follow the instructions in the notebook. Good luck.