Continuous Integration and Continuous Delivery with Azure Data Factory and Azure DevOps

Azure Labs by Roque Daudt (rdaudt@yahoo.com)

02 – Provision lab Environments: Azure Services

Overview of the environments' configuration

For these labs we will create three environments within a single Azure subscription: development (dev), test (tst) and production (prd). For each environment we will instantiate and configure the following services:

1. Azure Storage Account (SA)
2. Azure Key Vault (KV)
3. Azure Data Factory (ADF)

Within the subscription we will create a new organization in Azure DevOps. The image below depicts a high-level overview of the subscription, the environments and components.
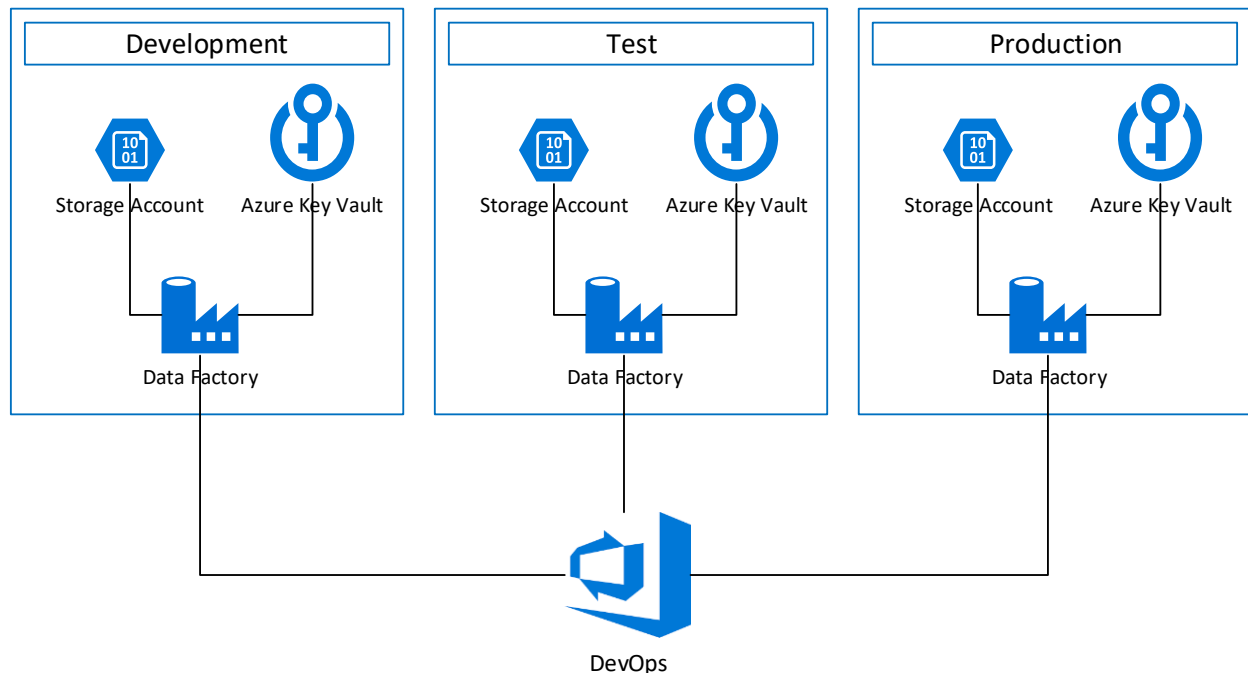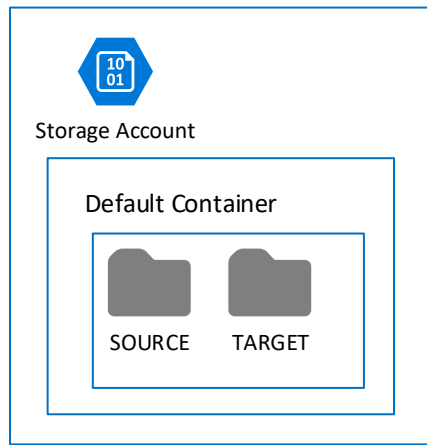

Azure Storage Account

Azure Storage Account (SA) supports several storage-related needs. Among other things, within a SA we can manage containers. In their turns, containers can have folders and files. One of these containers is the default container.

The ADF pipeline that we will work with in the labs copies one file from one folder to another in the SA. These folders are named SOURCE and TARGET and exist in the default container.

The image below depicts the SA's configuration for the labs.

As a heads up, the file that we will work with is a well-known file named **moviesDB.csv**.

Azure Key Vault (KV)

The ADF pipeline will need the ability to access the storage account, in which the files are securely stored. There are many ways to do so. In this lab, the ADF pipeline will use a connection string to access the file in the SOURCE folder.

It is a best practice to never embed keys and secrets in the artifacts that are managed by Azure DevOps, though. The recommended way to have these artifacts available to Azure services over the environments is to have them managed by KV. As one can imply, KV is Azure's service for management of keys, secrets and certificates. We will demonstrate how this best practice is implemented by having the ADF pipeline relying on KV in order to get the connection string that it needs to access the files in the SA.

In the image below we get a heads up that KV will hold a secret named **connstr**, through which ADF will be able to access the SA.
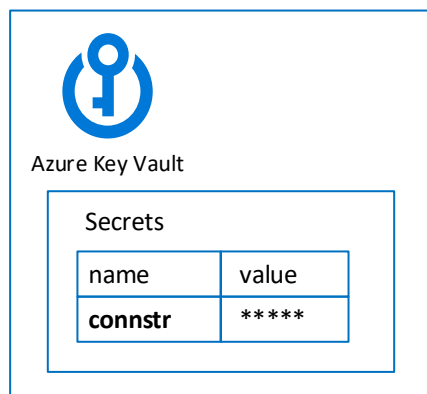
Azure Data Factory

As indicated above, the demo pipeline will have a simple feature: to copy one file from one folder to another, in a given SA.

Initially, a ADF pipeline and supporting objects will be created in the development environment, only. Through the labs we will create CICD processes that will deploy the ADF pipeline and the objects from development to higher environments: test (tst) and production (prd).
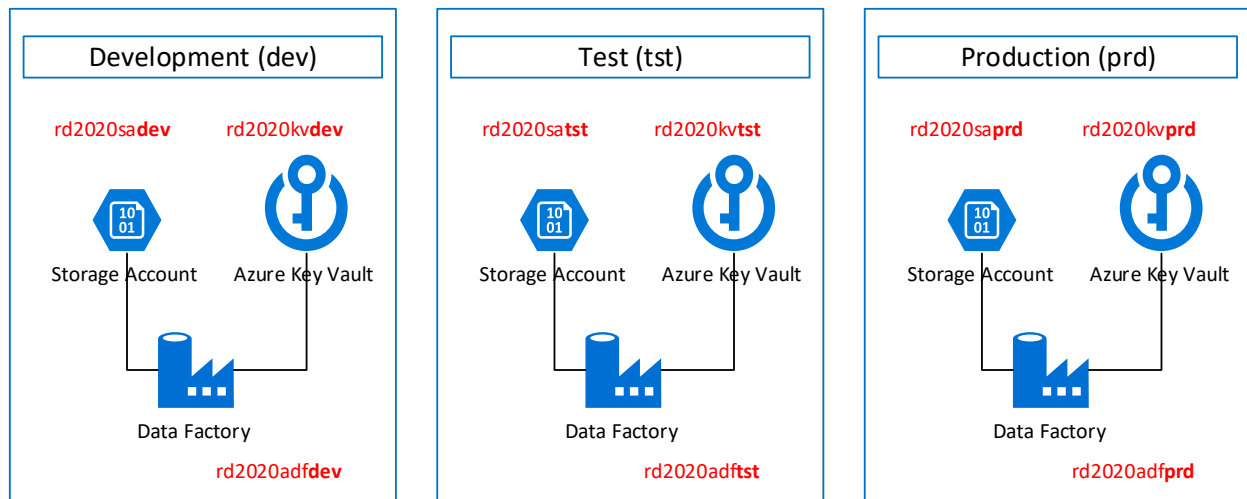
The image below depicts the demo ADF pipeline and objects
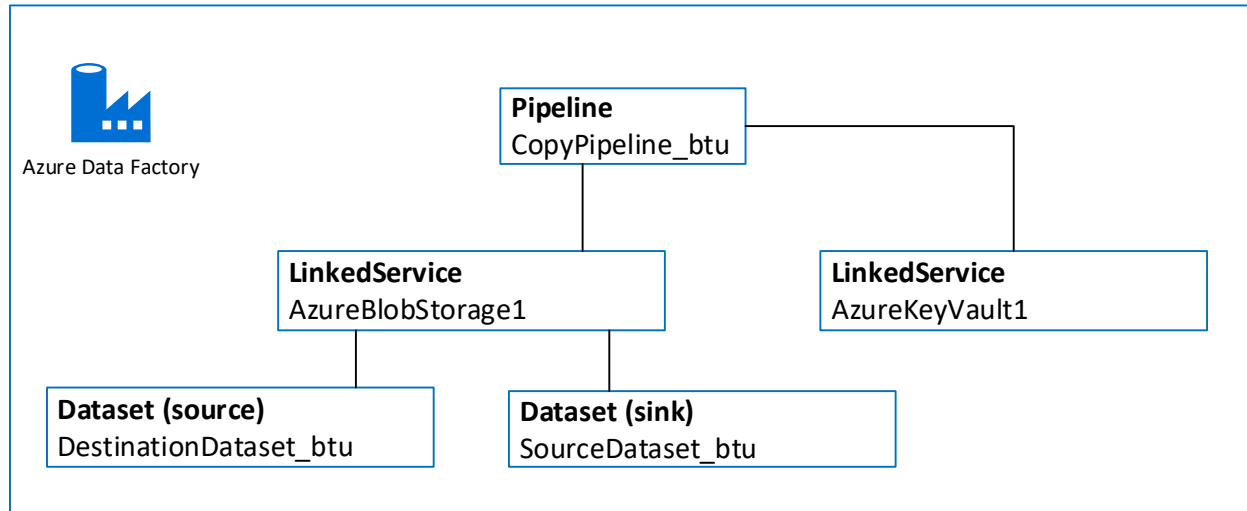
The pipeline has a single activity, the COPY activity. As it happens with ADF, the activities get to the source and sinks through linked services, which connects the activity to the datasets.

Because in this demo ADF gets access to demo files at SA level and both source and sink files are in the same SA, only one linked service to the SA suffices.

Naming convention

As a good practice and as a way to help to understand and run the labs, the environments and their components will be created using a naming convention. In this lab we will run scripts that create the environments. Before running them, you will have an opportunity to partially customize their names. Because some of the services used by these labs need to have unique names within Azure, we highly recommend that you do customize them to avoid conflicts with identifiers already in use by other Azure subscriptions. We will show how to customize them.

The default identifiers used by these labs are shown in the picture below, in red.

Development (dev)	Test (tst)	Production (prd)
rd2020sa**dev** rd2020kv**dev**	rd2020sa**tst** rd2020kv**tst**	rd2020sa**prd** rd2020kv**prd**
Storage Account Azure Key Vault	Storage Account Azure Key Vault	Storage Account Azure Key Vault
Data Factory	Data Factory	Data Factory
rd2020adf**dev**	rd2020adf**tst**	rd2020adf**prd**

Resource Groups

The resources created in each environment are created as members of Resource Groups (RG). We will create one RG for each environment. There will be an opportunity to customize the identifiers for the resource groups in your lab but this is not as important as customizing the services' names because resource groups need to be unique only within the subscription.

The default resource group identifiers used in this lab will be as follows:

Environment	Resource Group
Development	rd2020rgdev
Test	rd2020rgtst
Production	rd2020rgprd

Costs

There is no cost to create the environments. Also, my experience is that if you keep them for days not even a dollar was charged. You are safe, cost-wise, to create the environments.

Creating the lab environments

There are two major tasks for the creation of the lab environments:

1. Execution of a PowerShell script that will create the resource groups, storage accounts, key vaults and data factories
2. Creation of a new DevOps organization using the web UI

1. Creating the storage accounts, key vaults and data factories

Pre-requisite

a. Course material downloaded from … and unziped it at **C:\ADFDEVOPS** in your workstation
b. Azure PowerShell installed in your workstation

Steps

It is very easy to create the environments. We have built an ARM template that does the job for you. All that you must do is as shown below.

Configure the scripts

Open script **createEnvironments.ps1** file with any text/code editor. At line 6, replace the content of the **$prefix** variable. This is needed so that the script will give unique names to the services

```
1    # script: createEnvironments.ps1
2    # functionality:
3    # - setup azure environments in support of demo about ADF integration with Azure DevOps
4
5    # enter a different prefix. Ideally, use your initials as part of the prefix
6    $prefix = "rd2020"
```

As an example, you can use your initials and a date, such as in **trb20200928**, but this is up to you. Do not use **rd2020**.

Optionally, you can customize the location where the resources will be created by editing line 9.

```
8    # leave it as it is or enter the Azure location that better works for you
9    $loc = "West US 2"
```

Save and close the file.
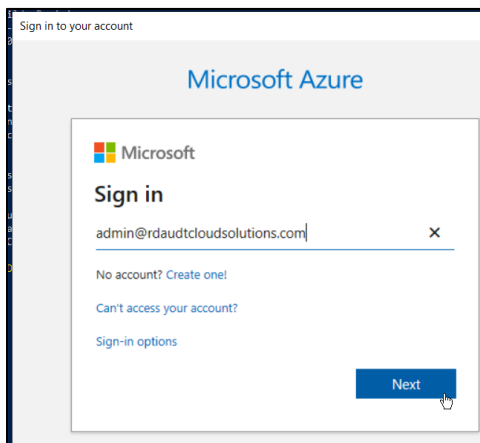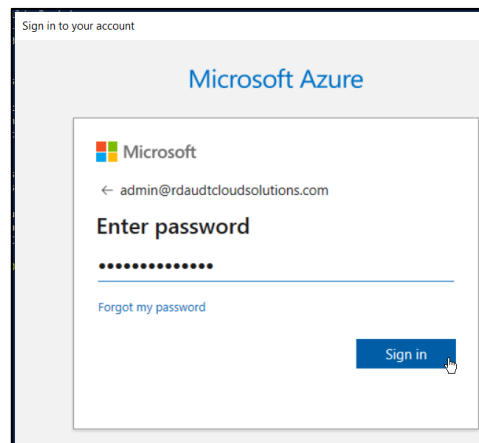
Connect to Azure

Start PowerShell and connect to Azure by running **Connect-AzAccount**.

```
PS C:\Windows\system32> Connect-AzAccount

Do you want to run software from this untrusted publisher?
File C:\Users\rdaud\OneDrive\Documents\WindowsPowerShell\Modules\Az.Accounts\1.9.4\Accounts.format.ps1xml is published
by CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond, S=Washington, C=US and is not trusted on your system.
Only run scripts from trusted publishers.
[V] Never run  [D] Do not run  [R] Run once  [A] Always run  [?] Help (default is "D"): A
```

Provide your credentials when asked to do so.

Upon a successful login you are shown the subscription name and tenant id

```
Account                        SubscriptionName        TenantId                              Environment
-------                        ----------------        --------                              -----------
admin@rdaudtcloudsolutions.com Azure subscription 1 4156de87-ab51-48cd-8f28-3bf9bfa72734 AzureCloud
```
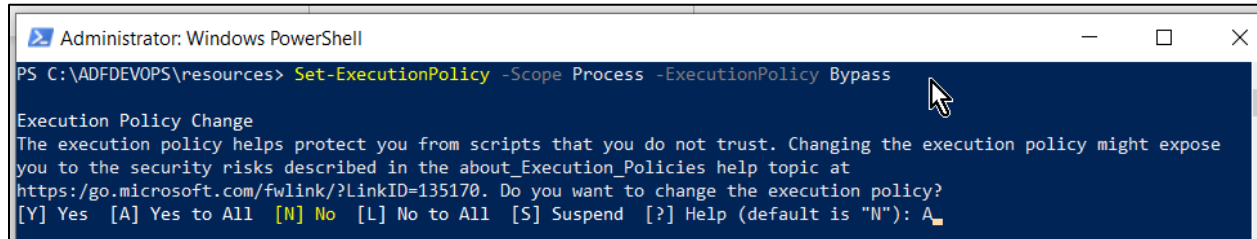
Navigate to **C:\ADFDEVOPS\resources**

```
Administrator: Windows PowerShell
PS C:\Windows\System32> cd \ADFDEVOPS\resources
```

Ensure that you will be able to run an unsigned script with the following command:
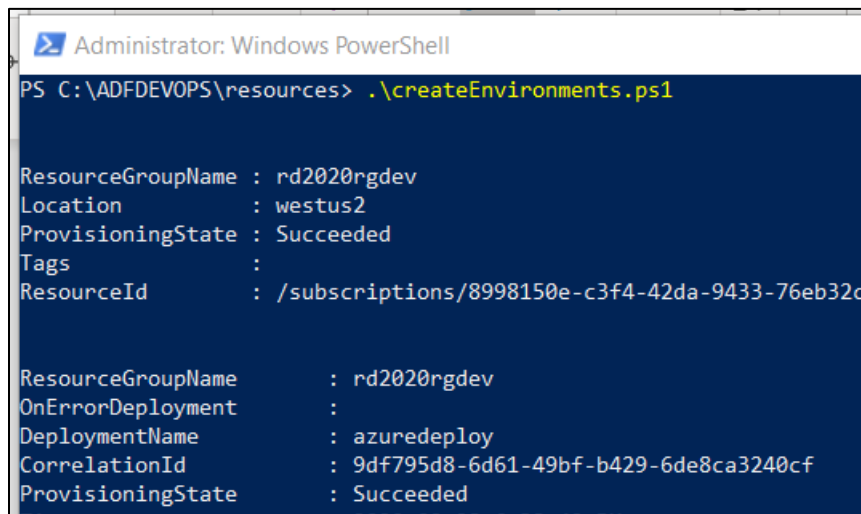
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

```
Administrator: Windows PowerShell                                          —    □    ×
PS C:\ADFDEVOPS\resources> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
```

Run **.\createEnvironments.ps1**

```
Administrator: Windows PowerShell
PS C:\ADFDEVOPS\resources> .\createEnvironments.ps1


ResourceGroupName : rd2020rgdev
Location          : westus2
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/8998150e-c3f4-42da-9433-76eb32c


ResourceGroupName      : rd2020rgdev
OnErrorDeployment      :
DeploymentName         : azuredeploy
CorrelationId          : 9df795d8-6d61-49bf-b429-6de8ca3240cf
ProvisioningState      : Succeeded
```

The script may take 2-3 minutes to run. It produces some good amount of output. Once it is done what you want to do is to scroll up and look for what it says for **Provisioning State** for the many provisioned resources.

You can also peruse the output in order to have a first look at the names of the newly created resources.

Browse the environments with Azure Portal

In your browser, navigate to https://portal.azure.com/ You should find three new resource groups. In this demo, they are named **rd2020rgdev**, **rd2020rdprd** and **rd2020rgtst**, as shown below. Yours will have a different name.
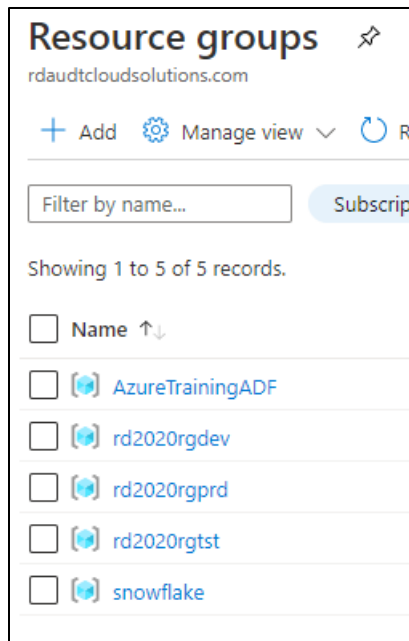
Click on each resource group to check that all the resources were successfully created. In each environment you should find one storage account, one key vault and one data factory. The image below shows how they appear in this demo, for the resource group that represents the **dev** environment, **rd2020rgdev**.
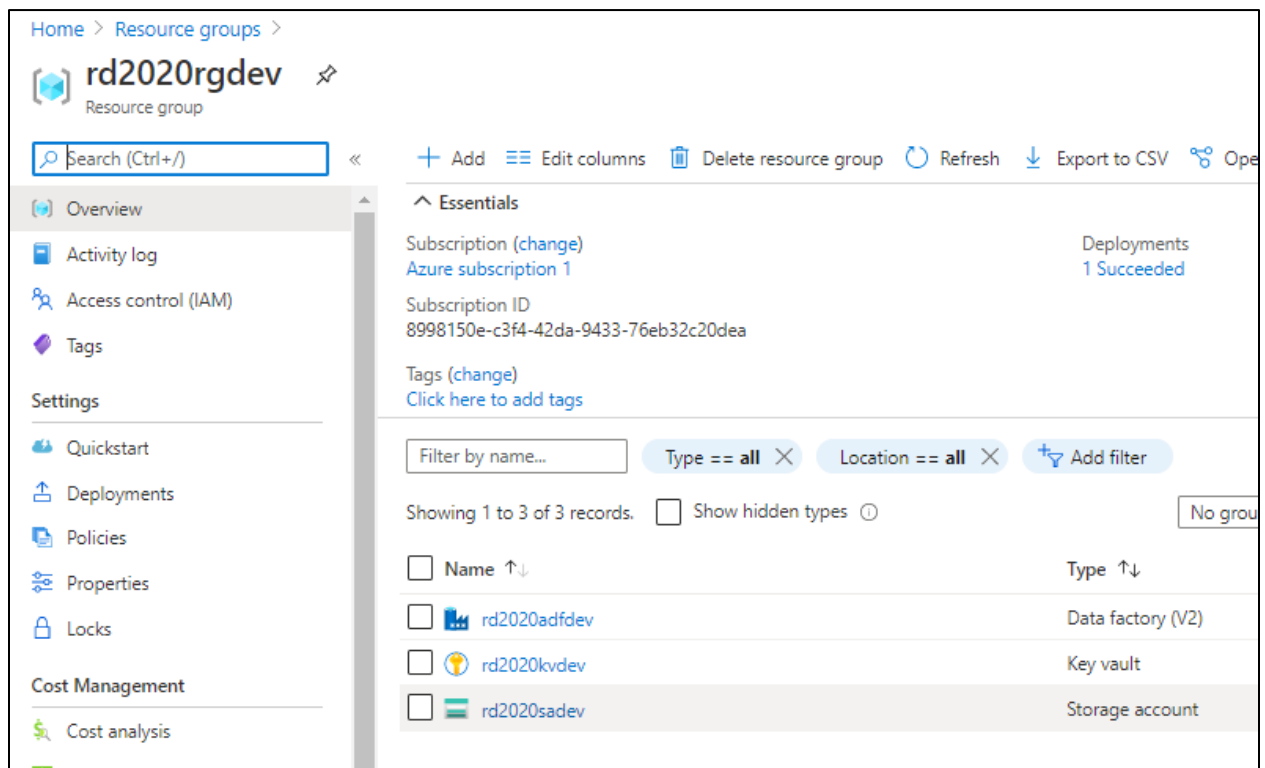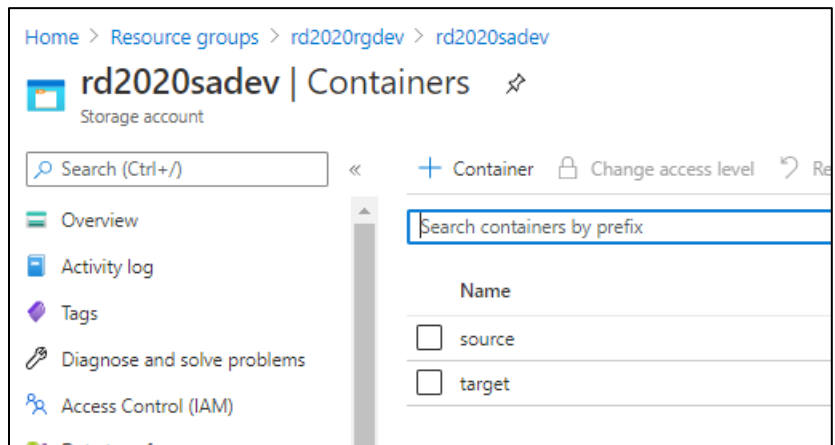

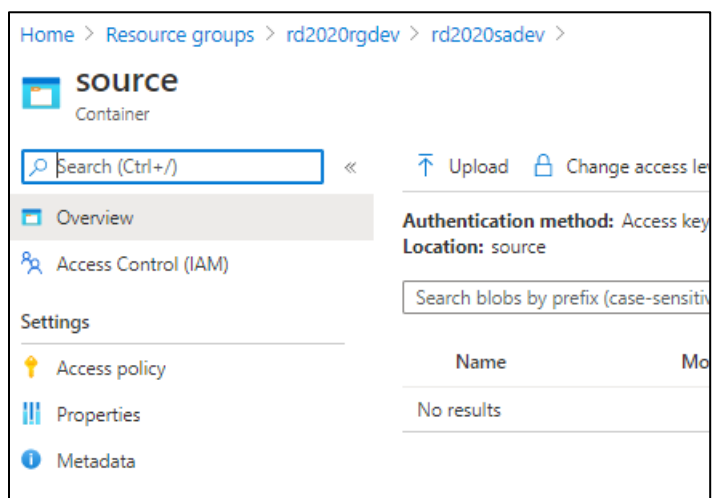## Explore the dev Azure Storage Account

Click **rd2020sadev** to explore the Azure Storage account created by the script.
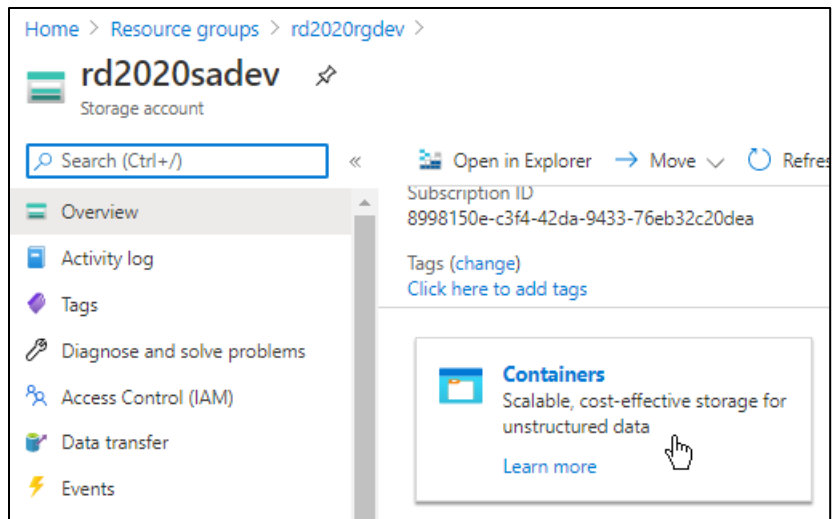
Click **Containers**. See the **source** and **target** folders in the default container.

Click **source**. See that the folder is empty.

Upload to **source** the file that will be copied by the ADF pipeline.

1. Click **Upload**
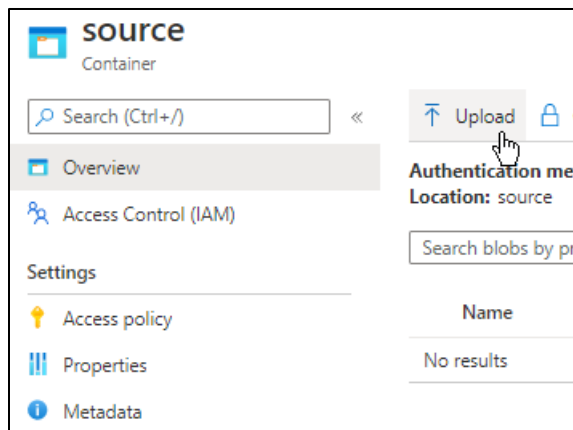
2. Click the folder icon and select **moviesDB.csv** in **C:\ADFDEVOPS\resources**
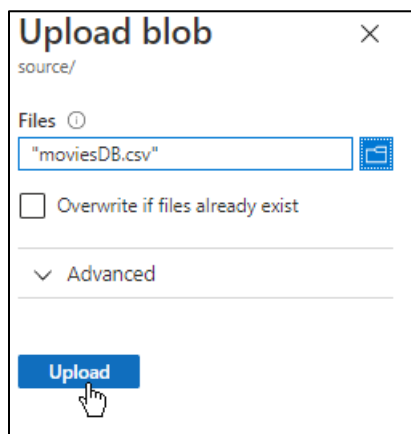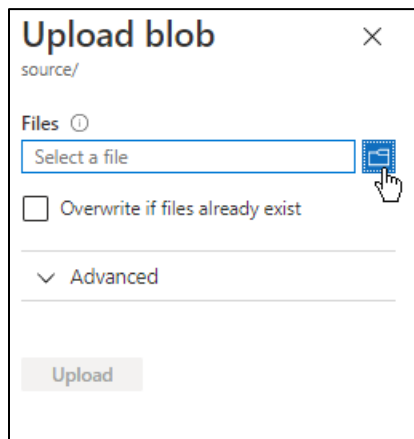
3. Click **Upload**, wait for the success message and close the **Upload blob** panel
4. See that a copy of **moviesDB.csv** is now in the **source** folder.
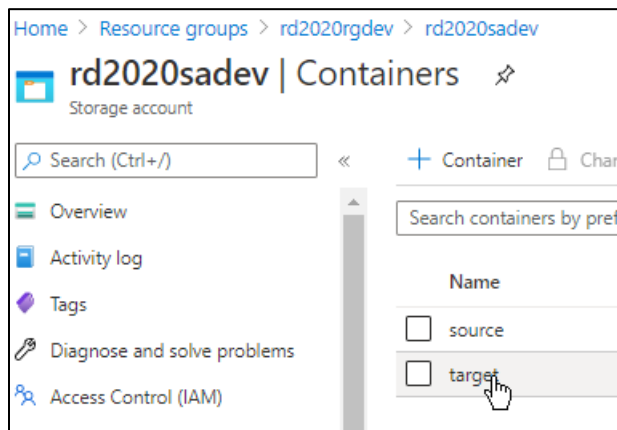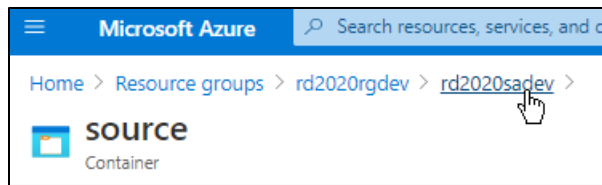
See that the **target** folder is empty.
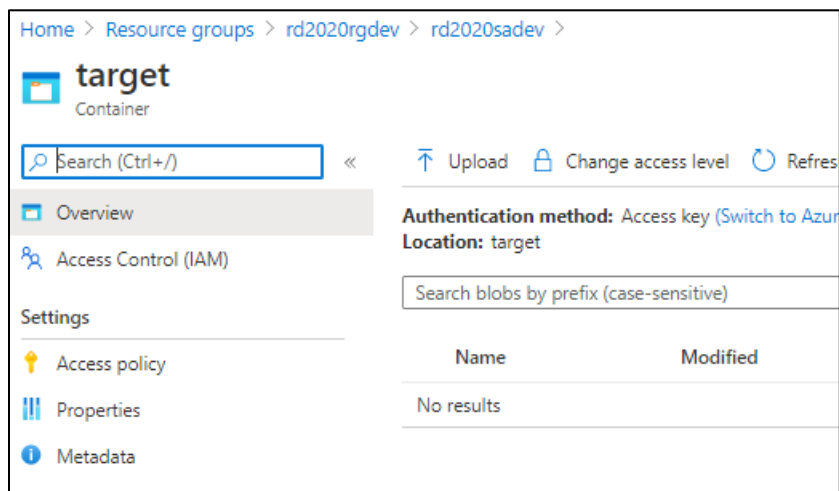
Explore the dev Azure Key Vault

Back to the resource group, click **rd2020kvdev**.
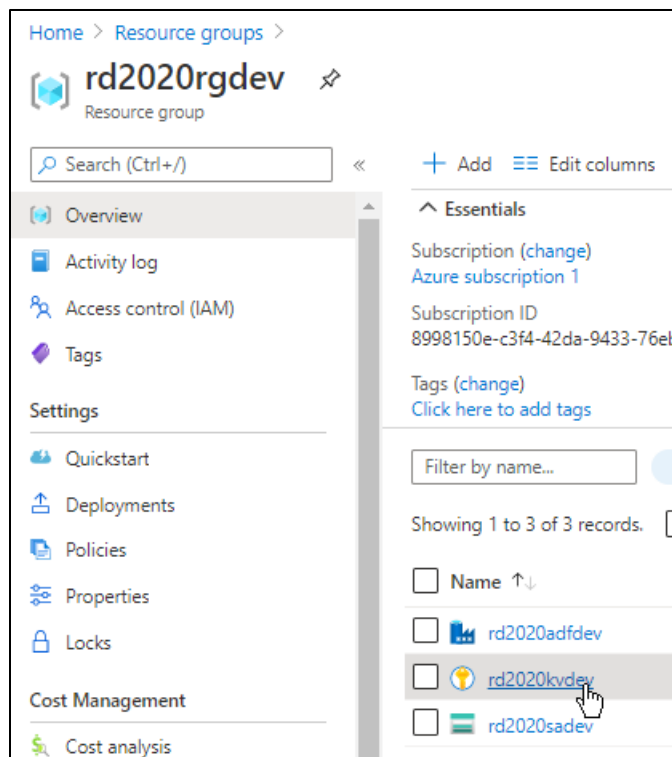
Don't worry about the message regarding soft-delete.
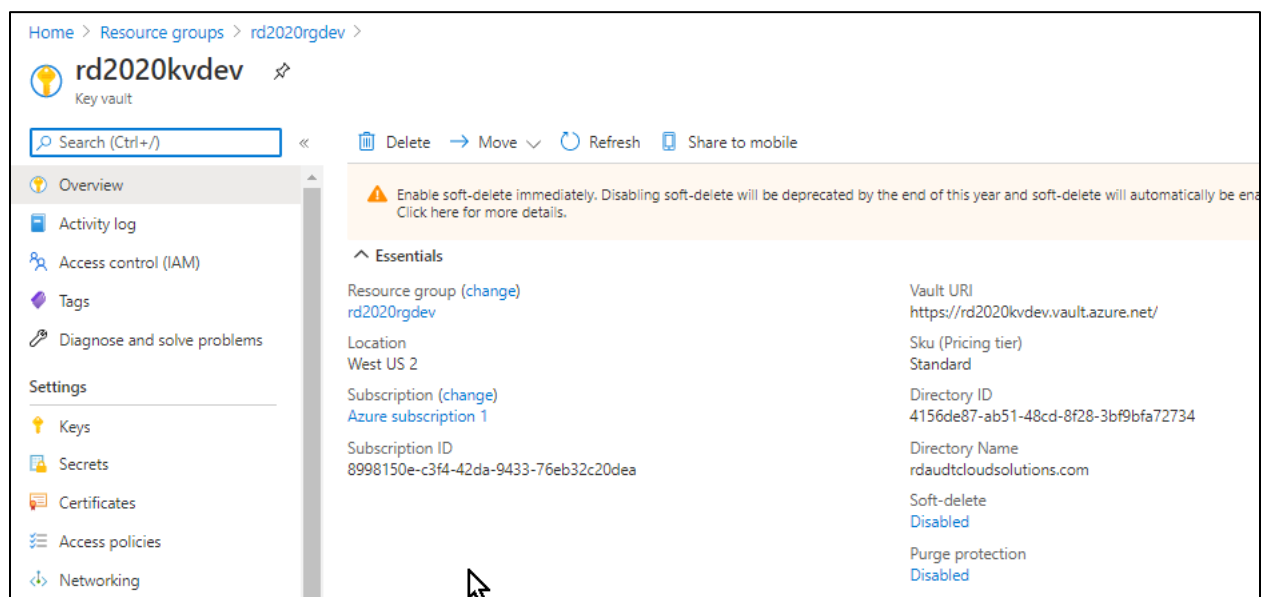
On the left menu, click **Access Policies**. ADF dev has already been granted access to Azure Key Vault but you (the reader of this lab) hasn't yet.

Click **Add Access Policy**. Select all key, secrets and certificates permissions.

Click **None selected**.

Add access policy

Add access policy

Configure from template (optional)

[⌄]

Key permissions

[16 selected   ⌄]

Secret permissions

[8 selected   ⌄]

Certificate permissions

[16 selected   ⌄]

Select principal *

None selected

Authorized application ⓘ

None selected

[Add]

Search/find your account.

Click **Select**.

Click **Add**.

Back to the previous screen, click **Save**.
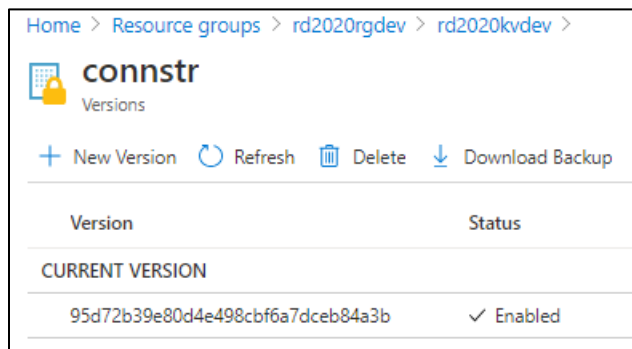
Now you have the permissions to see the secret. Click **Secrets**.

See the **connstr** secret. This is the secret that will be used by ADF to get to the files in the storage account.
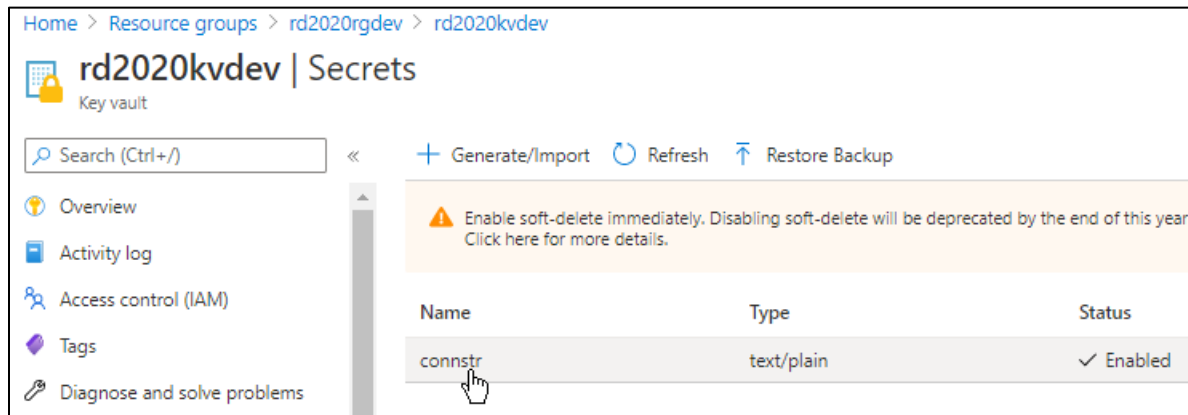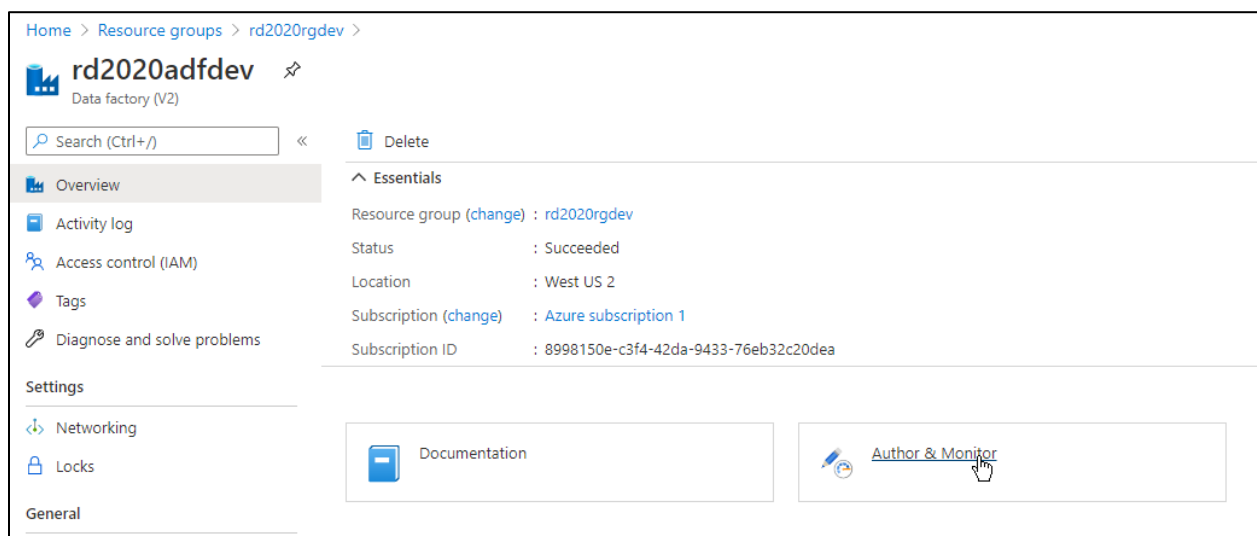

See the secret value (worry not, these services will be deleted once the lab documentation is completed).
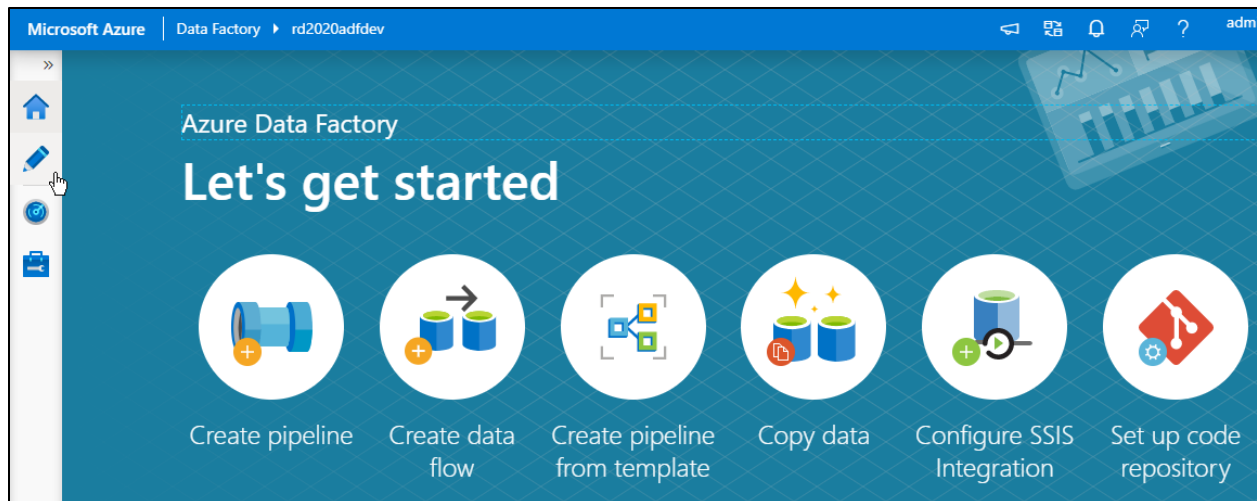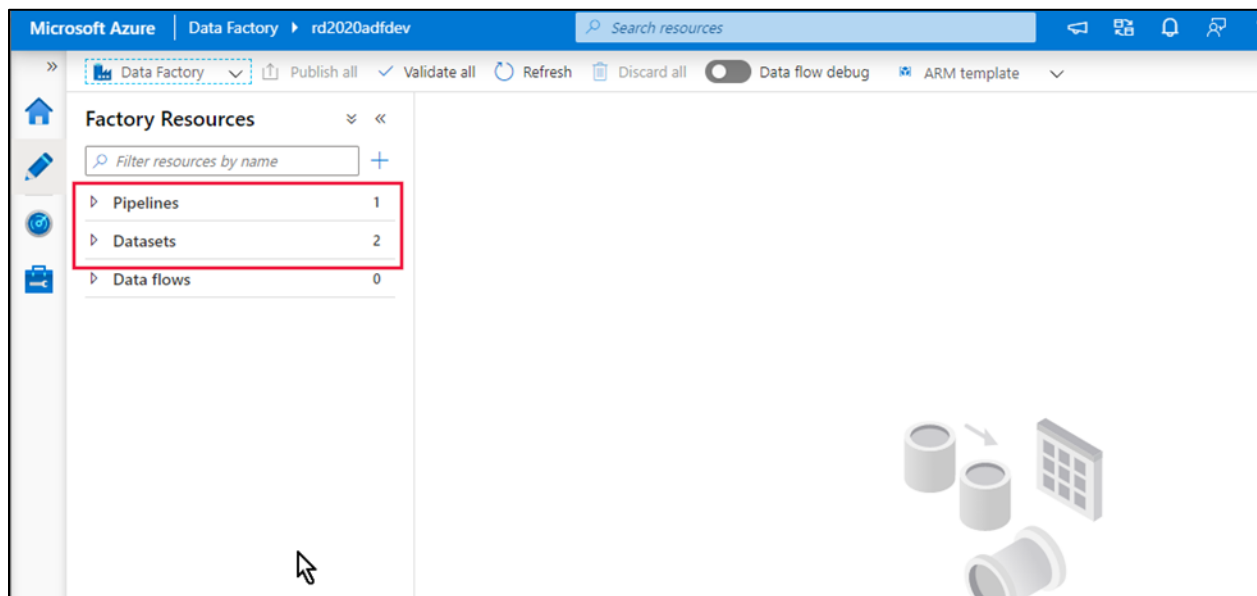

Azure Data Factory

Back to the resource group, click **rd2020adfdev** to get to the dev ADF.
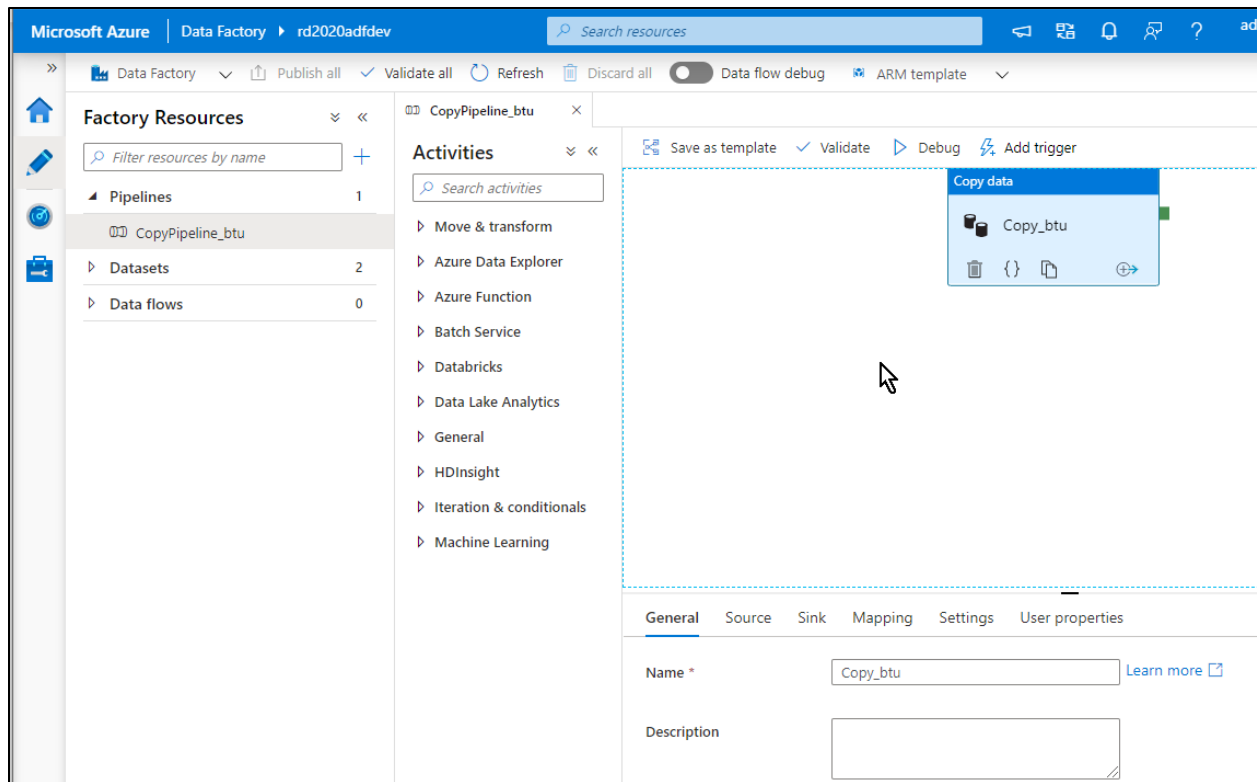

Click on **Author & Monitor**. Azure opens ADF's home page in another tab.

Click the **Author** button. You should see, as shown below, that there exists one pipeline and two data sets in ADF dev.
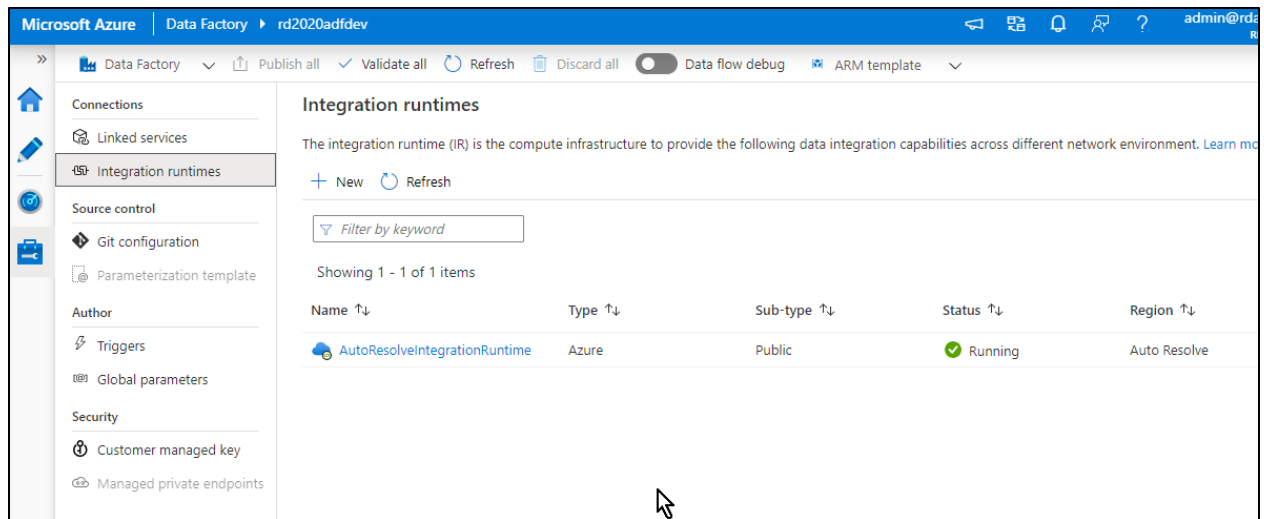

Click on **Pipelines**, on the pipeline name and then on the Copy data activity, you will see an image like the one below.

Click **Manage** and then **Linked services.** You will see that two Linked Services exists in ADF dev: **AzureBlobStorage1** and **AzureKeyVault1**.
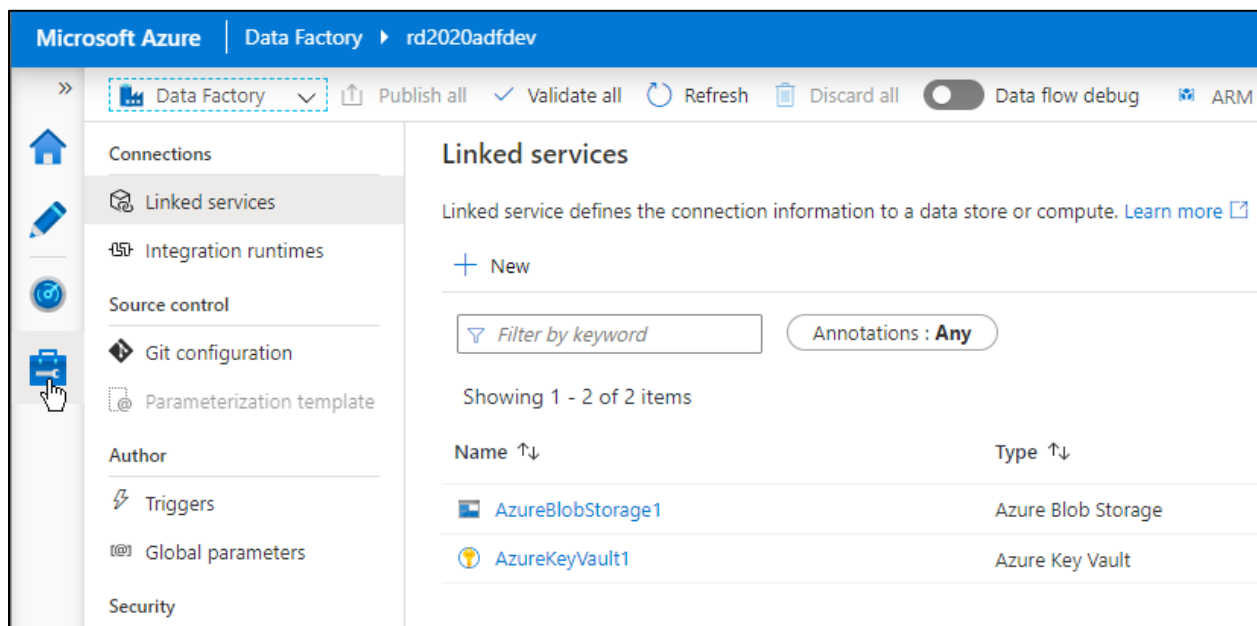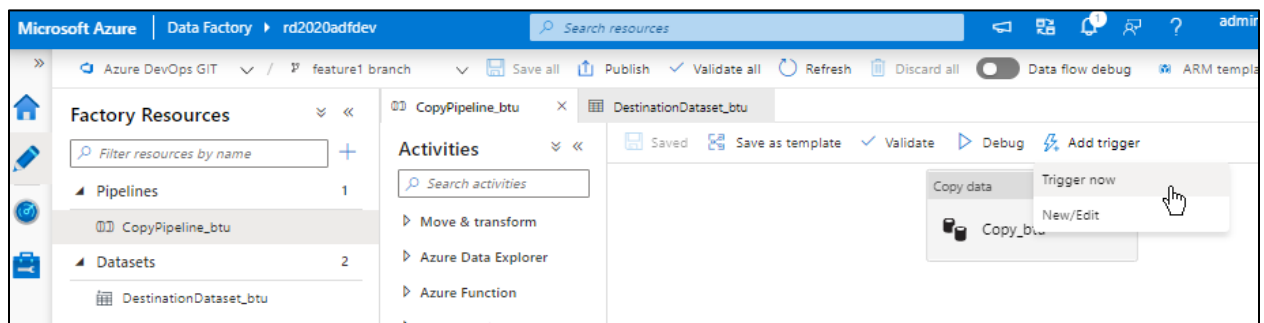

Click **Integration runtimes** you will see that one integration runtime exists.

In other words, the configuration of the ADF pipeline in DEV is as described by the pipeline's logical diagram shown previously in this document.
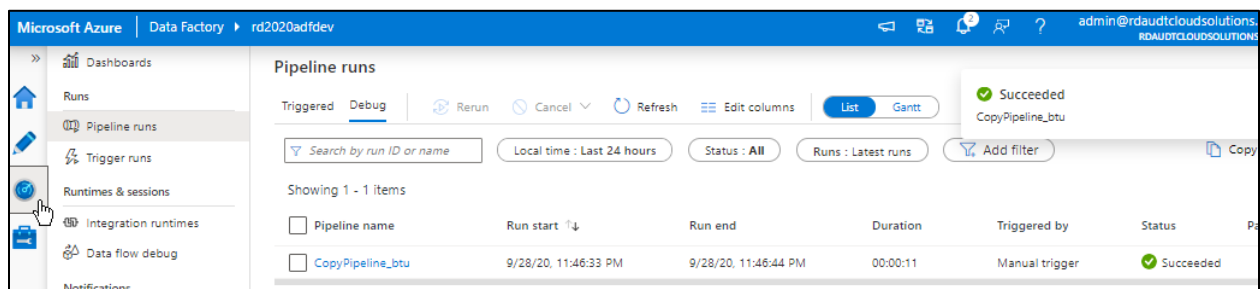
Now, let's run the pipeline to verify that it is working. Back to the Author tool, with the pipeline still open, click **Add trigger / Trigger now**.
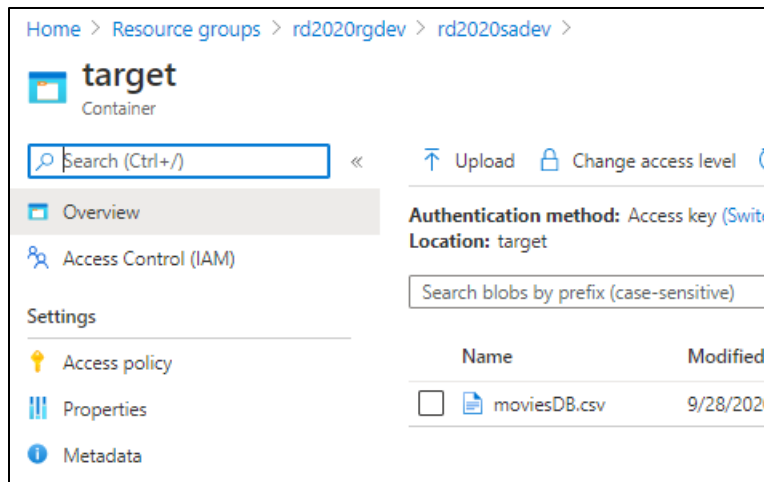

Click **OK** below.

Once you get the Succeeded message, go to **Monitor** (left menu) just to double check that the **Status** of the execution is Succeeded.

Go back to the dev Storage Account (**rd2020sadev**) and try to find file **moviesDB.csv** in the **target** folder. It is there, indeed.

Now, check that ADF test (**rd2020adftst**) and production (**rd2020adfprd**) are empty. The image below depicts **rd2020adftst**.
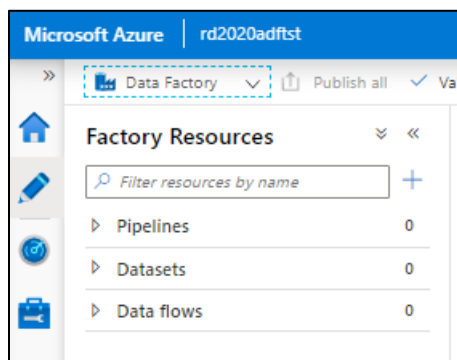
Through our labs we will create a release pipeline that promotes the ADF pipeline in dev to the two other environments.

What if something goes wrong?

If something goes wrong you will have to investigate the issue, delete the existing resource groups and run the process again, with whatever fix you have found.

You can run the script as many times as needed if you delete the resources groups before trying again.

Summary

This is a very important lab. We provisioned and explored several resources that we will use in the nexr labs.

Next

03 – Provision Lab Environments: DevOps Organization