

# Welcome [FIRSTNAME]

---

## Labs

---

[Module 1, Lesson 1, Lab 1 - Set up Azure Pass Lab](#)

[Module 1, Lesson 2, Lab 2 - Create and monitor Azure Data Factory using ARM](#)

[Module 1, Lesson 3, Lab 3 - Configure Data Factory Git integration](#)

[Module 2, Lesson 1, Lab 4 - Create and Monitor a Data Factory Pipeline](#)

[Module 2, Lesson 2, Lab 5 - Copy Data Wizard](#)

[Module 3, Lesson 1, Lab 6 - Data Transformation](#)

[Module 3, Lesson 2, Lab 7 - Control Flow](#)

[Module 5, Lesson 1, Lab 8 - Data Flow](#)

[Module 6, Lesson 1, Lab 9 - Using Azure Key Vault](#)





## WorkshopPLUS: Data AI Azure Data Factory

# Set up Azure Pass Lab

---

### **Introduction**

In this lab, you will redeem Microsoft Sponsored Azure Pass for this workshop.

### **Estimated Time**

15 minutes

### **Objectives**

At the end of this lab, you will:

- New Azure Subscription for this workshop.
- Understand how to look for balance and consumption.

### **Logon Information**

Use the following credentials to sign into the virtual environment.

- Username: **Admin**
- Password: **Passw0rd!**

## Table of Contents

---

[Exercise 1: Create a Microsoft Azure Pass Subscription](#)

[Reference: Checking your Azure Pass Balance and Usage details](#)

**Lab: Set up Azure Lab Pass**

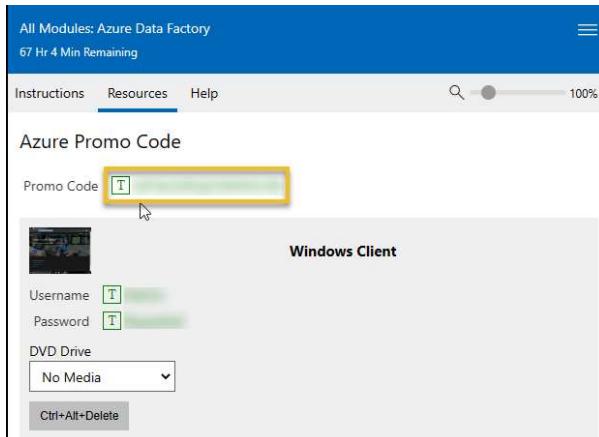
[Exercise 1: Creating a Microsoft Azure Pass Subscription](#)

This exercise shows how to create an Azure Pass subscription.

## Tasks

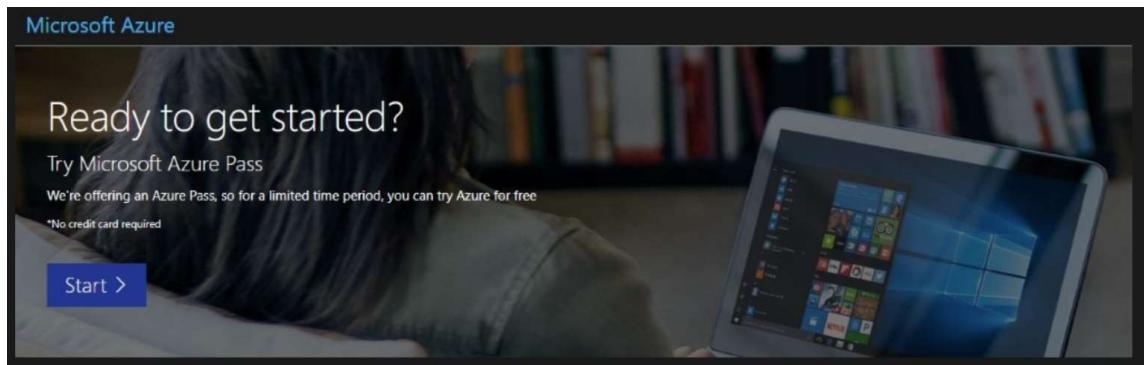
Redeem the Azure Pass using an **Microsoft Account**. It is not recommended, to use your corporate credentials.

1. Retrieve Azure Pass code from the Resource Tab in your virtual machine (right window).



2. Redeem a Microsoft Azure Pass Promo Code.

- a. From within your virtual machine, open a browser and navigate to: [www.microsoftazurepass.com](http://www.microsoftazurepass.com).
- b. Click the **start** button to **get started**.



- c. Enter your account login information and select **Sign In**.

Use your non corporate account (Microsoft Account) to login.



## Sign in

Email, phone, or Skype

[Can't access your account?](#)

[Sign-in options](#)

[Next](#)

- d. After making sure, that in the fourth line the correct email address is displayed, click **Confirm Microsoft Account**.

This is going to be the email to which the subscription is going to be tied to, so make sure, it's the non corporate account.

**Microsoft Azure**

The following Microsoft Account will be used for Azure Pass:

Given name: [redacted]  
Surname: [redacted]  
Microsoft Email: [redacted] @live.com

If the above email address is incorrect, please [sign out](#) and redeem using the correct Microsoft Account

[Confirm Microsoft Account >](#)

- e. Enter your promo code in the Promo code box, enter the captcha characters and click **Submit**.

The promo code can be found in the resources tab of the right window.

Microsoft Azure

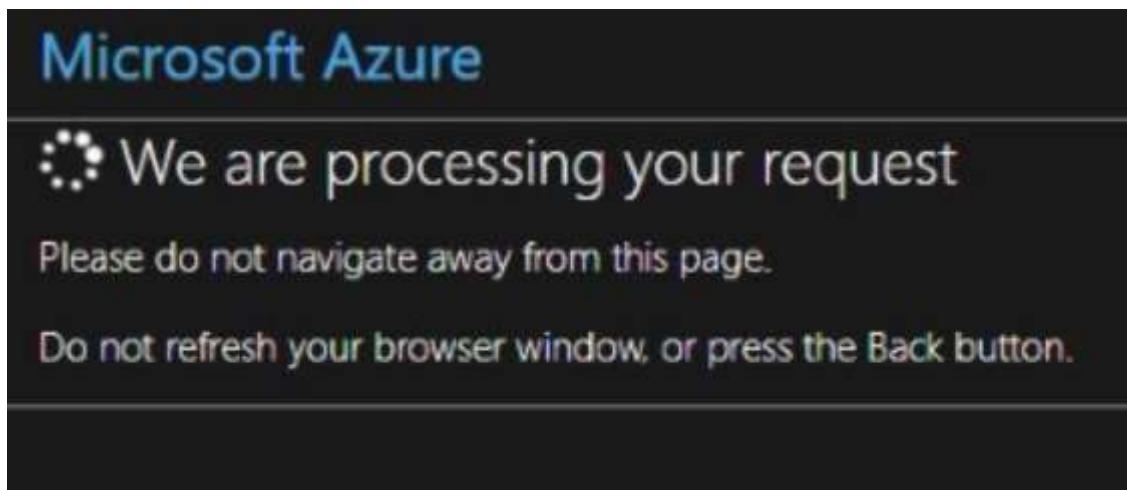
The following Microsoft Account will be used for Azure Pass:

Given name [REDACTED]  
Surname [REDACTED]  
Microsoft Email: cmrwspadf@outlook.com  
If the above email address is incorrect, please [sign out](#) and redeem using the correct Microsoft Account

Enter Promo code:  
[REDACTED]

New | Audio  
  
Enter the characters you see\*  
 Input Solution

f. It may take up to 5 minutes to process the redemption.



3. Activate your subscription.

a. When the redemption process is completed, it will redirect to the **sign up** page. Enter your account information such as region, first name, last name, a valid phone number, a valid address for the region, you selected at the top. At the bottom of the page, you'll have to check the first checkbox for the subscription agreement, the second one is optional. Click **Next**.

## Your profile



### Country/Region i

United States



Choose the location that matches your **billing address**. **You cannot change this selection later.** If your country is not listed, the offer is not available in your region. [Learn More](#)

### First name

### Middle name (Optional)

### Last name

### Email address for important notifications i

@outlook.com

### Phone i

(702) 588-4576

### Address line 1

Enter address data, check 'I agree to the subscription agreement...', the other checkbox is optional and click on **Sign Up**.

**Address line 1**

432 Carbon Blvd

**Address line 2 (Optional)****City**

Edmonton

**State**

Alaska

**ZIP code**

1233S

I understand that Microsoft may contact me about my free account.

I agree to the [subscription agreement, offer details](#).

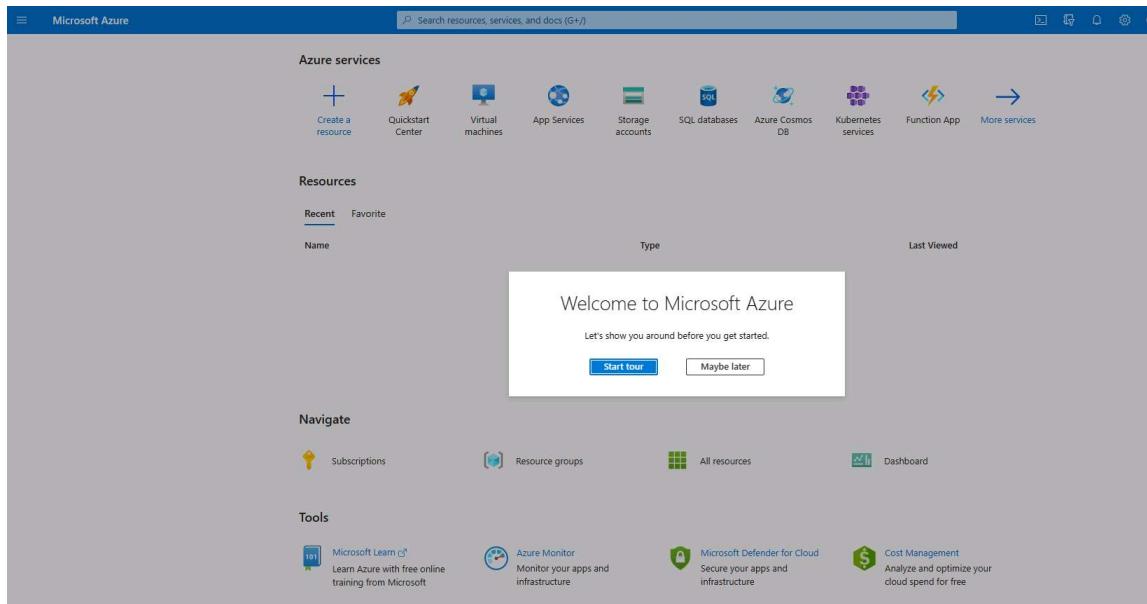
I will receive information, tips, and offers about Azure, including Azure Newsletter, Pricing updates, and other Microsoft products and services.

I would like Microsoft to share my information with select partners so I can receive relevant information about their products and services.

[Privacy Statement](#)

**Sign up**

- b. You might have to confirm the address. Now you just wait around 3-5 mins until you are redirected to your new Azure subscription.



Exercise has been completed.

## Reference: Checking your Azure Pass Balance and Usage details

After registering for an Azure Pass subscription you will have a limited dollar amount (\$100 USD) or a limited amount of time (30 days from time of registration), which ever comes first, that the subscription will be active and available to you.

1. Log in to [www.microsoftazuresponsorships.com](http://www.microsoftazuresponsorships.com) with **Microsoft Account**

- To check Balance details - click on **Check your Balance**
- To check Usage statistics - click on **Usage Details**



## WorkshopPLUS: Data AI Azure Data Factory

# Create and monitor Azure Data Factory using ARM

---

### Introduction

During this lab, you will learn how to use an ARM template to create the resources needed to complete this and future labs. You will also create and configure the services as needed to prepare for future labs.

### Estimated Time

30 minutes

### Objectives

At the end of this lab, you will be able to:

- Create an Azure Data Factory

### Logon Information

Use the following credentials to sign into the virtual environment.

- Username: **Admin**
- Password: **Passw0rd!**

## Table of Contents

---

[Exercise 1: Provision Lab Resources using an Azure Resource Manager \(ARM\) Template](#)

[Exercise 2: Create the input folder and files in container](#)

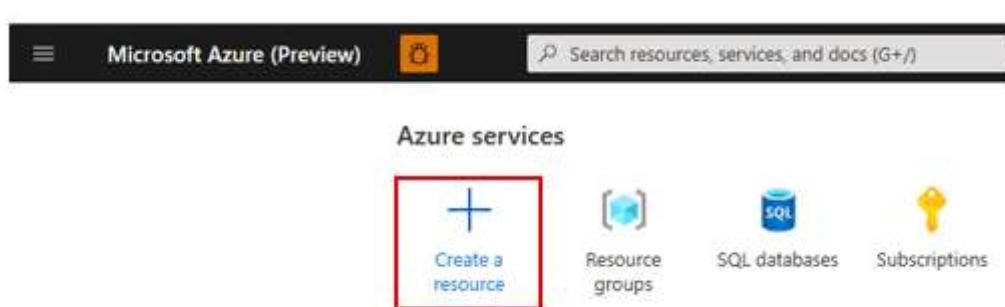
[Lab: Create and Monitor Azure Data Factory using ARM](#)

# Exercise 1: Provision Lab Resources using an Azure Resource Manager (ARM) Template

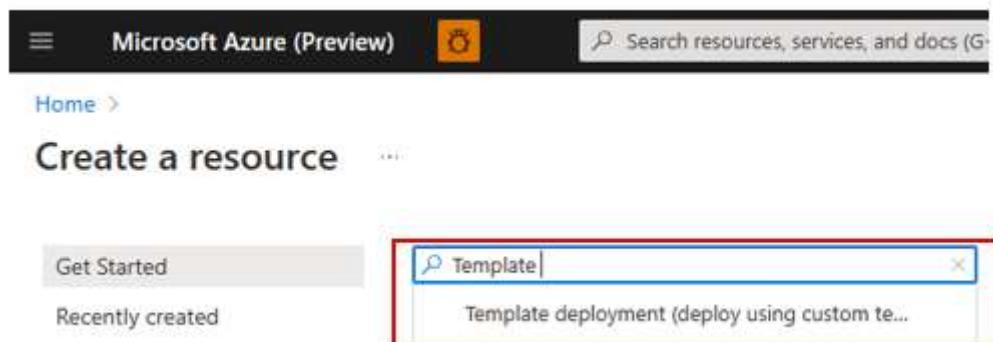
Throughout this workshop's labs, you will utilize various Azure resources outside of Data Factory. To maximize your learning about Data Factory, you will provision all necessary Azure resources in one step by leveraging ARM templates.[1]

## Tasks

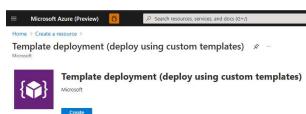
1. Connect to the Microsoft Azure Portal.
  - a. Open Edge browser and navigate to <http://portal.azure.com/> to connect to Microsoft Azure Portal.
  - b. Sign in with your credentials.
2. Deploy resources from an ARM template.
  - a. To deploy a customized template through the portal, select **Create a resource**



- b. Type in **template** on the search bar, then select **Template deployment (deploy using custom templates)**.



- c. Select **Create**.



- d. Select **Build your own template in the editor**.

Home > Create a resource > Template deployment (deploy using custom templates) >

**Custom deployment** ...

Deploy from a custom template

Select a template   Basics   Review + create

Automate deploying resources with Azure Resource Manager templates in a single, coordinated operation. Create or select a template below to get started. Learn more about template deployment.

**Build your own template in the editor**

Common templates

- Create a Linux virtual machine
- Create a Windows virtual machine
- Create a web app
- Create a SQL database

... +

e. Click **Load file** and select file

\LabFiles\M01\_L02\_Lab01\ADF\_Workshop\_ARM\_Template.json into the editor. Review the file to see the details of the items being created.



f. Click **Save** in the bottom left of the blade.

3. The Resource Template will display a page for you to enter values into for the parameters.

# Custom deployment

Deploy from a custom template

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Select a template    **Basics**    Review + create

## Template



Customized template ↗

5 resources

Edit template

Edit parameters

Visualize

## Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Azure Pass - Sponsorship

Resource group \* ⓘ

Create new

## Instance details

Region \* ⓘ

East US

Resource Location ⓘ

eastus

Uniqueness String \* ⓘ

User Name \* ⓘ

Sql Server Password \* ⓘ

Pricing Tier ⓘ

standard

Previous

Next

Review + create

- a. Your subscription details should populate.
- b. For Resource group, select **Create new** and enter a name for the new resource group (Ex: RG-ADFdemo).
- c. For Region, select **East US** or **West US**, in which the resource group will be located. You have to choose one of these regions due to a limitation of the sponsored subscription, which only provides 10 vcores in either the WestUS or EastUS region. Since you'll be creating a Azure Databricks Cluster later on, you'll need some of these vcores.
- d. For Resource Location, again select either *Eastus* or *WestUs*, in which all resources will be created. If you choose another region, you may have resource issues later on.

- e. Enter a Uniqueness String, which will be used as a prefix for resource names to help you identify them. Some of the resources require a name that is unique across all of Azure.
  - i. Do not make the Uniqueness string too long because some resource names have a character limit.
  - ii. Do not use uppercase letters or special characters because some resource names do not allow them.
- f. The user name is **datafactadmin**, this will be used as the name for the admin user for the SQL Server being created.
- g. For SQL Server Password, use **d@taFactSQL1**. Remember it, as you will use it later.
- h. For Pricing Tier, leave this set to **standard**. This reflects the pricing tier for the Azure Databricks workspace being created.
- i. Click **Review + Create**.
- j. Click **Create**.

#### 4. Verify Resources.

- a. Wait for the notification that all resources have been deployed. (this usually takes 3 - 5 mins)
- b. Select **Resource groups** in the left pane then go to your newly created resource group.
- c. Verify that the following resources were created:
  - Data Factory
  - Storage Account
  - Databricks Service
  - SQL Database
  - SQL Server



- d. If you do not see all of the resources above, it could be because your Uniqueness string was not unique enough, was too long, or included upper case or special characters. If the SQL database was not created, ensure that your password meets the rules as defined in step 2.o above.

#### 5. Add the Client IP address to the SQL Server

- a. Within your newly created resource group, click on the newly created SQL Server.

The screenshot shows the Azure Resource Group (RG-AzureDFDemo) dashboard. The left pane has a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, Resource costs, Deployments, Security, Policies, and Logs. The main pane displays resource details for a subscription named 'Azure Free - Sponsorship'. It lists five resources: 'mttdemo1' (Data Factory (V2)), 'mttdemo1blob' (Storage account), 'mttdemodatabricks' (Azure DataBricks Service), 'mttdemodbgroup' (SQL database), and 'mttdemosqlserver' (SQL server). A red arrow points to the 'Networking' link in the left pane.

b. In the left pane, under Security, Click on Networking.

The screenshot shows the Azure Security blade. On the left, there are several sections: 'Security' (highlighted with a red arrow), 'Networking' (highlighted with a red box), 'Microsoft Defender for Cloud', 'Transparent data encryption', 'Identity', and 'Auditing'.

c. Under Firewall rules section, Select the **+Add client IP** option and set **Allow access to Azure services** to **ON**.

The screenshot shows the 'Firewall rules' settings page. At the top, it says 'Allow certain public internet IP addresses to access your resource. Learn more'. Below that are two buttons: '+ Add your client IPv4 address' (highlighted with a red box and a red arrow) and '+ Add a firewall rule'. The main area shows a table with columns 'Rule name', 'Start IPv4 address', and 'End IPv4 address'. At the bottom, there's an 'Exceptions' section with a checked checkbox 'Allow Azure services and resources to access this server' (highlighted with a red box and a red arrow). At the very bottom are 'Save' and 'Discard' buttons.

d. Then click **Save**.

Exercise 1 has been completed.

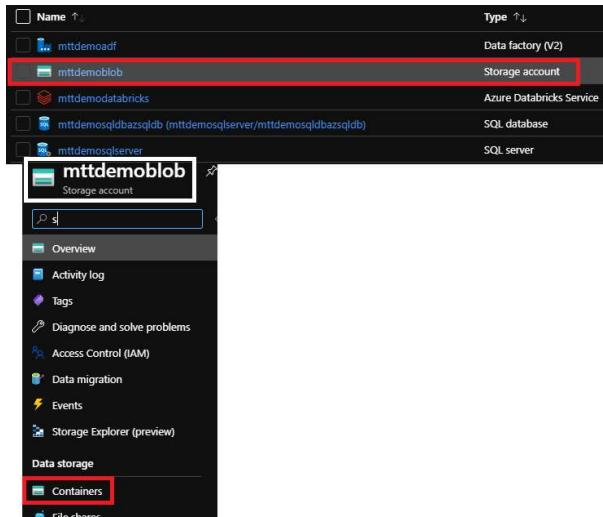
## Exercise 2: Create the input folder and files in container

In this exercise, you will select the blob storage account created via the ARM template. You create a container called **adftutorial** and a folder named **input** in the container, and then upload a sample file to that input folder.

## Tasks

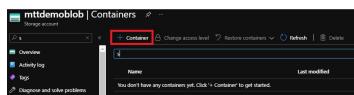
### 1. Select Storage account

Select your Storage account from the Azure portal. You can use the search bar at the top and it will end in **blob** or find it in your new Resource Group. Select your storage account, then select **Containers**.

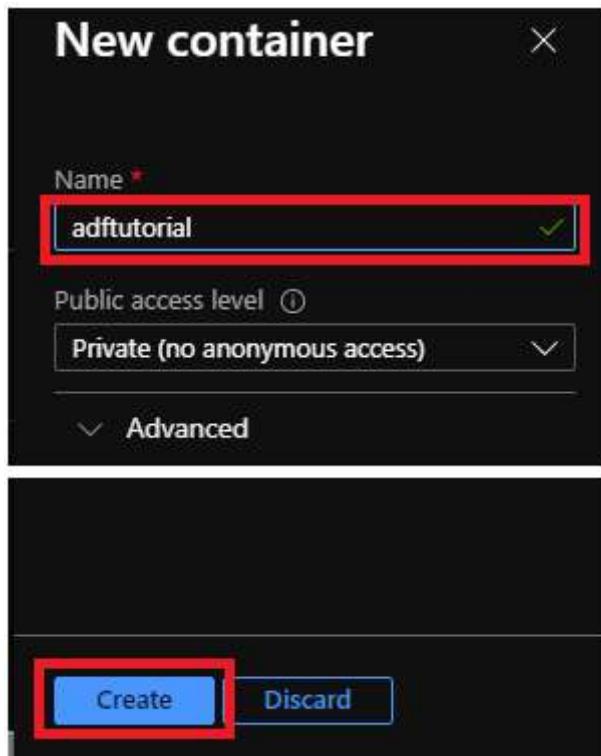


### 2. Create a container

- On Containers page, select **+Container** from the toolbar.



- In the New container dialog box, enter **adftutorial** for the name, and click the **Create** button.



3. Once created, click on the container name, **adftutorial**.

4. Upload a file

a. On the Container page, select **Upload** from the toolbar.

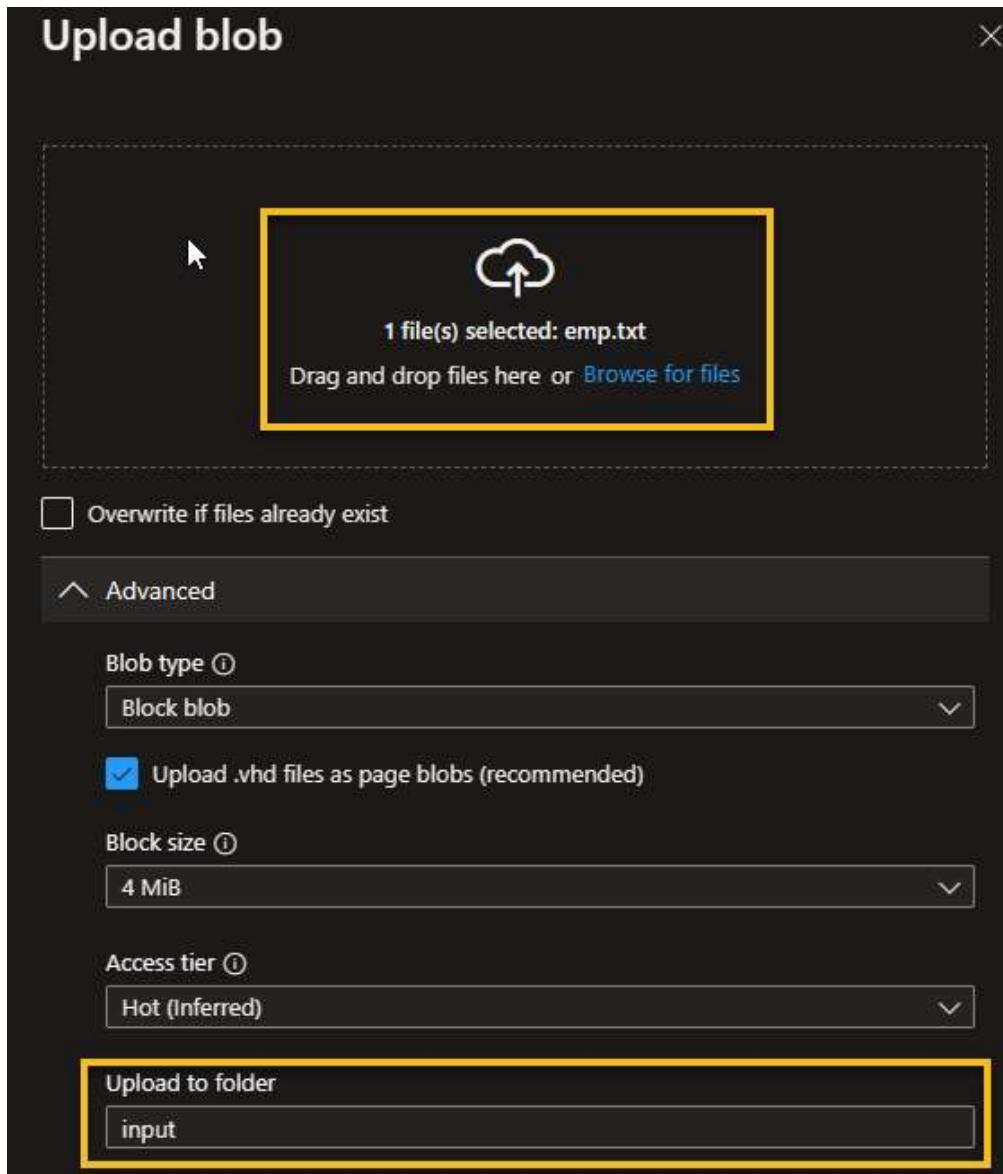
The screenshot shows the toolbar of a Container page. It includes buttons for 'Upload' (highlighted with a red box), 'Change access level', 'Refresh', 'Delete', and a search bar. Below the toolbar, it says 'Authentication method: Access key (Switch to Azure AD User Account)' and 'Location: adftutorial'. There is also a search bar with the placeholder 'Search blobs by prefix (case sensitive)'. A table below shows columns for 'NAME' and 'MODIFIED', with a note 'No blobs found.'

b. On the Upload blob page, select **Advanced**.

The screenshot shows the 'Upload blob' dialog box. It has a header 'Upload blob' and a path 'adftutorial/'. Below is a 'Files' section with a 'Select a file' input field and a folder icon. There is a checkbox for 'Overwrite if files already exist'. A section titled 'Advanced' is expanded, showing a red box around it. At the bottom is a 'Upload' button.

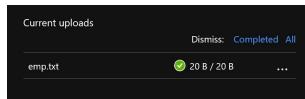
c. In the Azure portal, on the Upload blob page, click on the **folder** icon next to the **Select a file** box and select the file **\LabFiles\M01\_L02\_Lab01\emp.txt** file.

d. In the **Upload to folder** box, enter **input**.

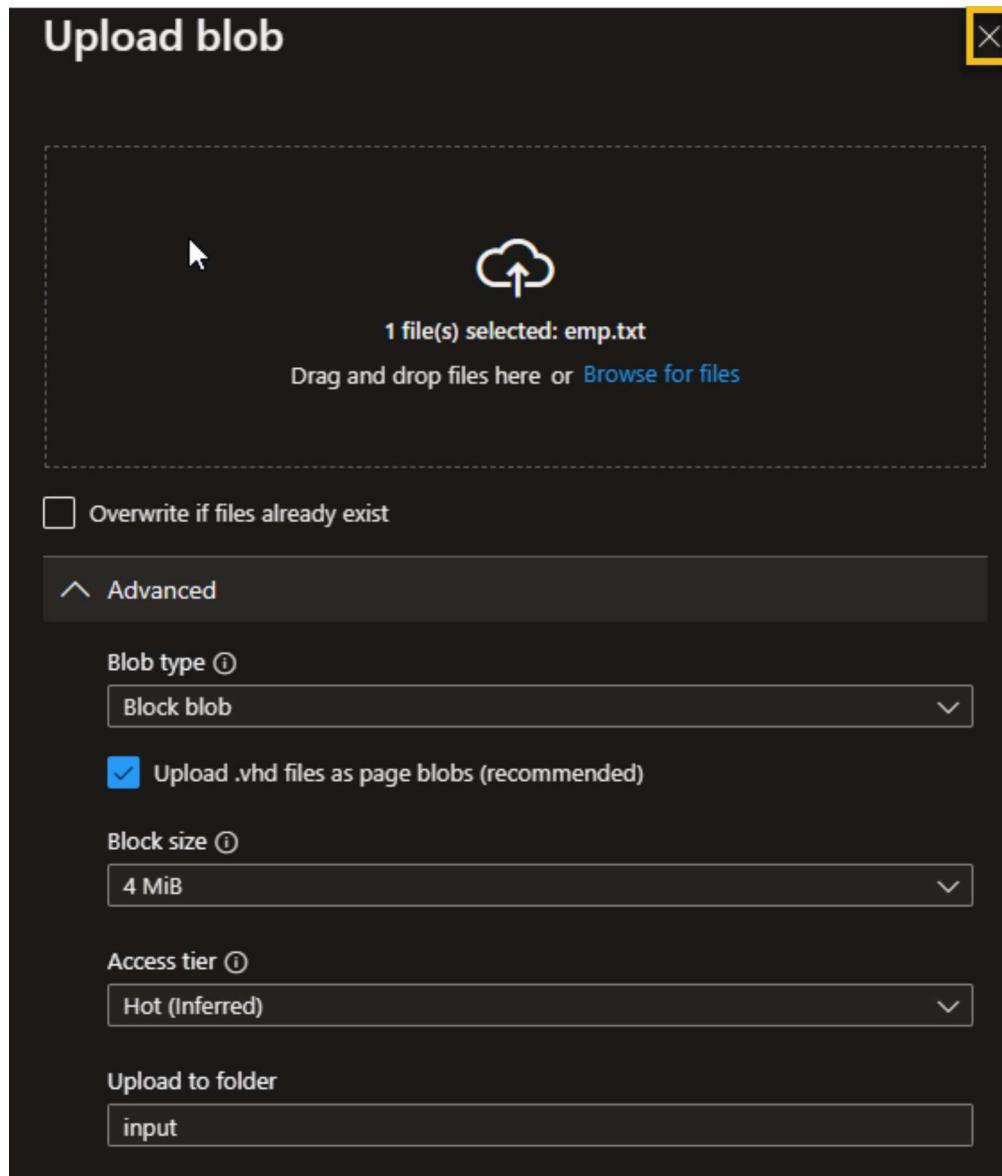


e. Click the **Upload** button.

You should see the emp.txt file and the status of the upload in the list.



f. Close the Upload blob page by clicking **X** in the corner.



Exercise 2 has been completed.

[1] An [ARM Template](#) is a JSON file that defines one or more resources to deploy to a resource group or subscription. The template can be used to deploy the resources consistently and repeatedly.



## WorkshopPLUS: Data AI Azure Data Factory

# Configure Data Factory Git Integration

---

### Introduction

During this lab, you will learn how to set Git Integration for Azure Data Factory.

### Estimated Time

30 minutes

### Objectives

At the end of this lab, you will be able to:

- Create an Azure DevOps Git Repository
- Set branches on your Repository
- Switch between Live and Git mode

### Logon Information

Use the following credentials to sign into the virtual environment.

- Username: **Admin**
- Password: **PasswOrd!**

## Table of Contents

---

[Exercise 1: Set up Azure DevOps Account](#)

[Exercise 2: Setting up Azure Data Factory Source Control](#)

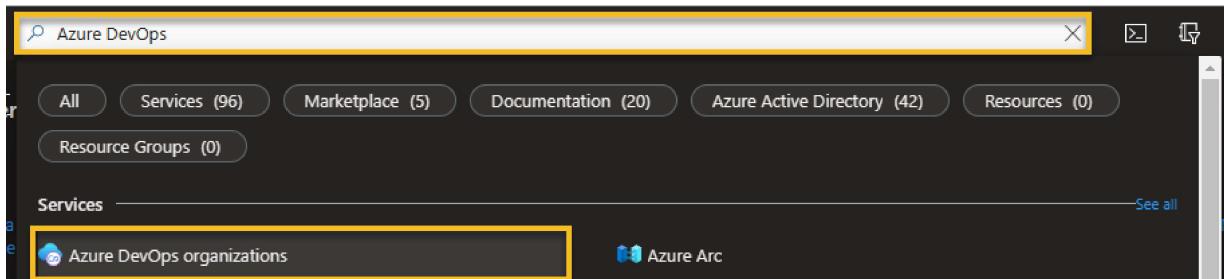
[Lab: Configure Data Factory Git Integration](#)

## Exercise 1: Set up Azure DevOps Account

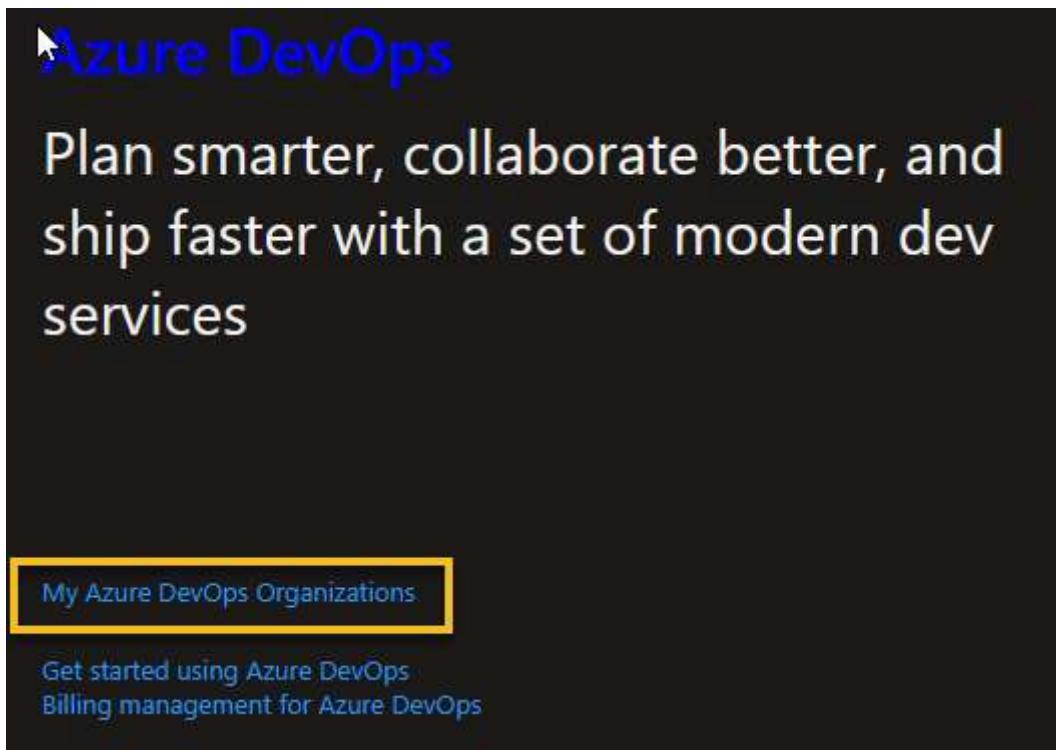
### Tasks

**Configure a DevOps Account for the same Microsoft Account you are using for the Azure Free Pass Subscription**

1. Go to <https://portal.azure.com/> and sign in with your Microsoft Account.
2. Search for **Azure DevOps** on the top and select **Azure DevOps Organization**.



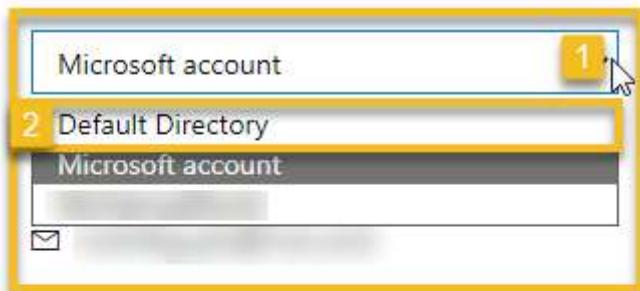
3. Go to My Azure DevOps Organization.



4. From the drop-down list, select **Default Directory**.



Mohit [REDACTED] [Edit profile](#)



## Visual Studio Dev Essentials

Get everything you need to build and deploy  
your app on any platform.

[Use your benefits](#)

5. Select **Create new organization**.



## Get started with Azure DevOps

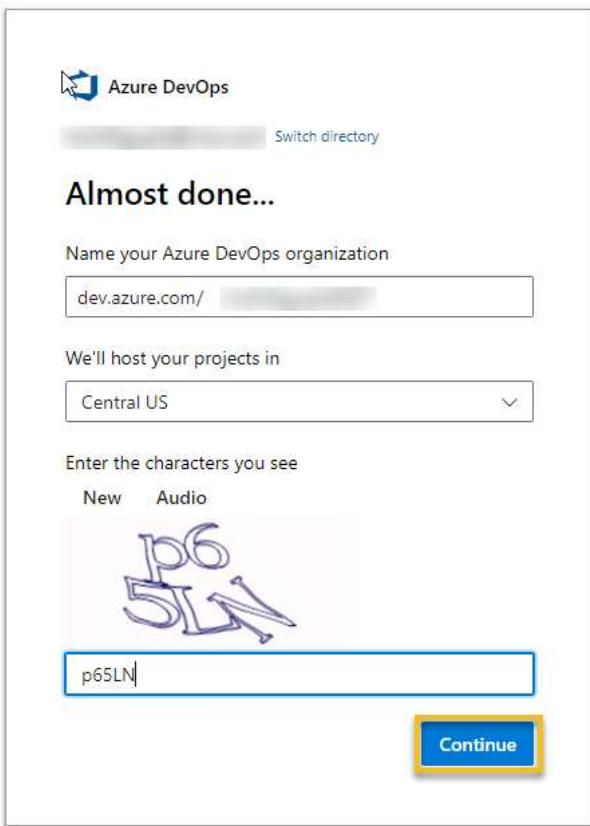
Plan better, code together, ship faster with Azure DevOps

**Create new organization**

6. Select **Continue** and unselect the checkbox.

A screenshot of the Azure DevOps 'Get started with Azure DevOps' page. At the top, there's a 'Create new organization' button with a yellow border. Below it is a 'Continue' button. A yellow box highlights the 'I would like information, tips, and offers about Azure DevOps and other Microsoft products and services. Privacy Statement.' checkbox, which is currently checked. The page also includes a 'Switch directory' link and a note about agreeing to the Terms of Service, Privacy Statement, and Code of Conduct.

7. Update the organization name, region project will be hosted and enter security code. Click **Continue**.



8. Enter project name, click + **Create project**.

+ **Create a project to get started**

Project name \*

Visibility

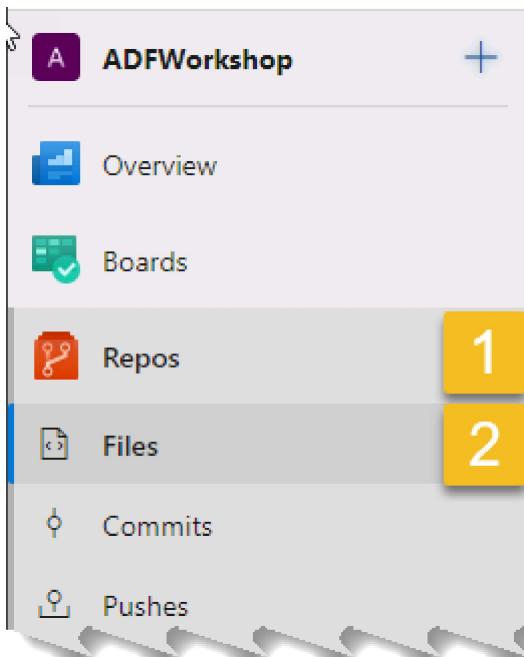
Public  
Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private  
Only people you give access to will be able to view this project.

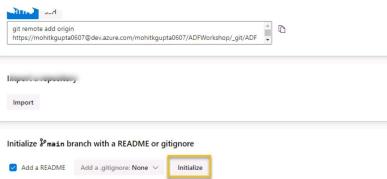
Public projects are disabled for your organization. You can turn on public visibility with organization policies.

**+ Create project**

9. Within your Project, click on **Repos** and than **Files**.



10. Then click on Files and **Initialize**.



Exercise 1 has been completed.

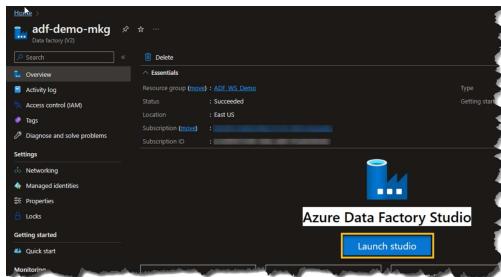
## Exercise 2: Setting up Azure Data Factory Source Control

Source Control allows developers to check-in/push Azure Data Factory code and assets to a code repository. Azure DevOps source control tracks the history of who made changes and what the changes were, as well as provides a centralized source for all your code. Source control also allows for easy cross-teams collaboration..

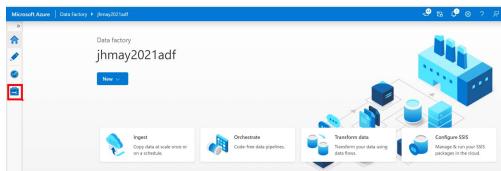
- It is **not** recommended to switch in between live mode and Git mode as the changes made into live mode may not be seen in Git mode.

### Tasks

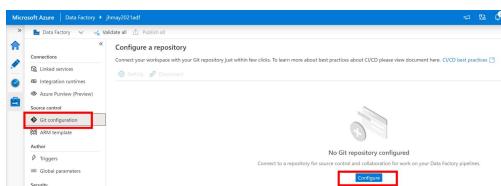
1. Login to Azure Portal, navigate to your Azure Data Factory Account and then click on Launch Studio



2. Select the Manage icon in the left panel



3. Select the Git configuration under Source Control heading then click Configure in the middle of the page.



4. In the Repository settings pane to choose the repository type. Select Azure DevOps Git and your subscription then select **Continue**.

## Configure a repository

Specify the settings that you want to use when connecting to your repository.

**Repository type \*** ⓘ

Azure DevOps Git 1

Cross tenant sign in ⓘ

Azure Active Directory ⓘ

Default Directory 2

**Continue**

**Cancel**

5. Select the organization created in [Exercise 1](#).

## Configure a repository

Default Directory [REDACTED]

Specify the settings that you want to use when connecting to your repository.

Select repository  Use repository link 

Azure DevOps organization name \* 

0607

6. This opens additional Git repository settings based on your repository type selection. Select your Project Name, and Repository Name. Select 'main' for both the Collaboration branch and Import resource into this branch. Click **Apply**.

Configure a repository

Default Directory [REDACTED]

Specify the settings that you want to use when connecting to your repository.

Select repository  Use repository link

Azure DevOps organization name \* 

0607

Project name \* 

ADFWorkshop

Repository name \* 

ADFWorkshop

Collaboration branch \* 

main

Publish branch \* 

adf\_publish

Root folder \* 

/

Import existing resources

Import existing resources to repository

Import resource into this branch 

**Apply**

Back

Cancel

If *main* branch is missing, then the repo was not initialized correctly.

7. Set working branch to main and click **Save**.

Set working branch

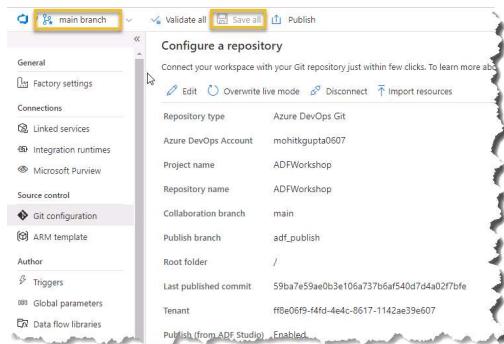
Working branch  Create new  Use existing

main

**Save**

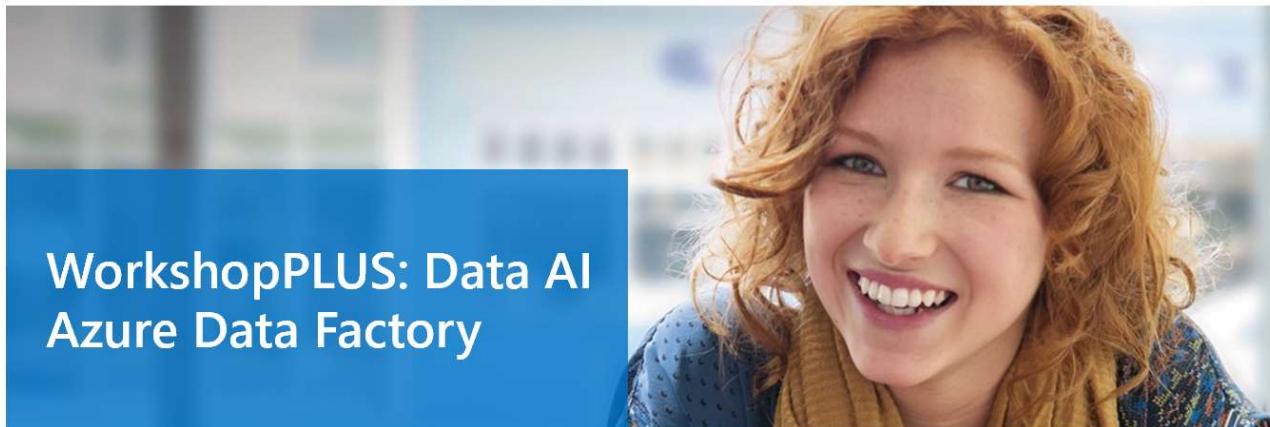
- i** If you were working with multiple developers within a Data Factory, you will create a separate branches (a.k.a feature branch) to save your code to.

8. Review your Git repository settings. Note that you are now working within the main branch rather than the directly within the Data Factory service.



- i** The **Save all** button is available now (currently not enabled because no changes have been made within the Data Factory UI). When you make a change to your Data Factory this will be enabled.

Exercise 2 has been completed.



## Create and Monitor a Data Factory Pipeline

---

### **Introduction**

During this lab, you will learn how to navigate the Azure Data Factory *Author* and *Monitor* interfaces to create and monitor a Data Factory. In this Data Factory we will create a pipeline that copies data from one folder to another folder in Azure Blob Storage.

### **Estimated Time**

30 minutes

### **Objectives**

At the end of this lab, you will be able to:

- Create a linked service
- Create datasets
- Create, debug, trigger, and monitor a Pipeline.

### **Login Information**

Use the following credentials to sign into the virtual environment:

- Username: **Admin**
- Password: **Passw0rd!**

## Table of Contents

---

[Exercise 1: Create a linked service](#)

[Exercise 2: Create datasets](#)

## [Exercise 3: Create a Pipeline](#)

## [Exercise 4: Debug and Trigger the Pipeline Manually](#)

## [Exercise 5: Monitor the Pipeline](#)

## Lab: Create and Monitor a Data Factory on Azure Portal

During this lab, you will learn how to use the Azure Data Factory *Author & Monitor* interfaces to create and monitor a data factory pipeline.

### Exercise 1: Create a linked service

In this exercise, you create a linked service to link your Azure storage account to the data factory. The linked service has the connection information that the Data Factory service uses at runtime to connect to it.

#### Tasks

1. Select your Azure Data Factory service from the Azure portal. You can use the search bar at the top and it will end in **adf** or select from your Resource Group. Select the **Launch Studio** tile to start the Azure Data Factory user interface (UI) application on a separate tab.

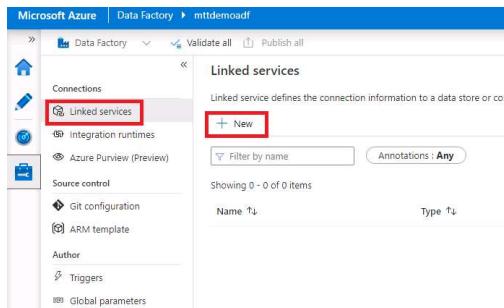


2. On the Let's get started page, switch to the **Manage** tab in the left panel.

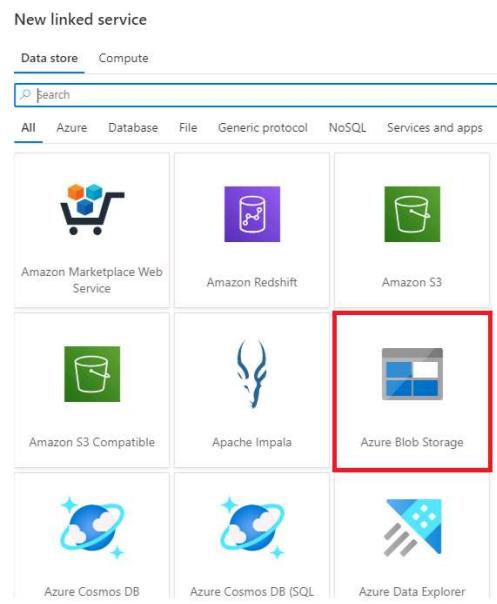


3. Create a Linked Service.

- a. Select **Linked services**, and then select the **+New** button.



b. On the New Linked Service page, select **Azure Blob Storage**, and then select **Continue**.



c. Complete the following steps:

- i. For Name, enter **AzureStorageLinkedService**.
- ii. For Authentication Method, ensure **Account Key** is selected.
- iii. For Account selection method, ensure **From Azure subscription** is selected.
- iv. Select your **Azure subscription**.
- v. Select **storage account name** from the drop downs.
- vi. Select **Test connection** to confirm that the Data Factory service can connect to the storage account.
- vii. Select **Create** to save the linked service.

New linked service (Azure Blob Storage)

Name *	AzureStorageLinkedService
Description	
Connect via integration runtime *	AutoResolveIntegrationRuntime
Authentication method	Account key
<input checked="" type="radio"/> Connection string <input type="radio"/> Azure Key Vault	
Account selection method	<input checked="" type="radio"/> From Azure subscription <input type="radio"/> Enter manually
Azure subscription	Azure Pass - Sponsorship (1)
Storage account name *	mttdemoblob
Additional connection properties	
<a href="#">New</a>	
Test connection <input checked="" type="radio"/> To linked service <input type="radio"/> To file path	
<div style="text-align: right;"> <span style="border: 1px solid red; padding: 2px;">Create</span>    <span>Back</span>    <span style="border: 1px solid red; padding: 2px;">Connection successful</span>    <span> Test connection</span>    <span>Cancel</span> </div>	

Linked service has been created.

Name	Type	Related
AzureStorageLinkedService	Azure Blob Storage	0

Exercise 1 has been completed.

## Exercise 2: Create datasets

In this exercise, you create two datasets: InputDataset and OutputDataset. These datasets are of type AzureBlob. They refer to the Azure Storage linked service that you created in the previous exercise.

The input dataset represents the source data in the input folder. In the input dataset definition, you specify the blob container (adftutorial), the folder (input), and the file (emp.txt) that contain the source data.

The output dataset represents the data that's copied to the destination. In the output dataset definition, you specify the blob container (adftutorial), the folder (output), and the file to which the data is copied. Each run of a pipeline has a unique ID associated with it. You can access this ID by using the system variable RunId. The name of the output file is dynamically evaluated based on the run ID of the pipeline.

In the linked service settings, you specified the Azure storage account that contains the source data. In the source dataset settings, you specify exactly where the source data resides (blob container, folder, and file). In the sink dataset settings, you specify where the data is copied to (blob container, folder, and file).

## Tasks

1. Click on the Pencil/Author button in the left pane.

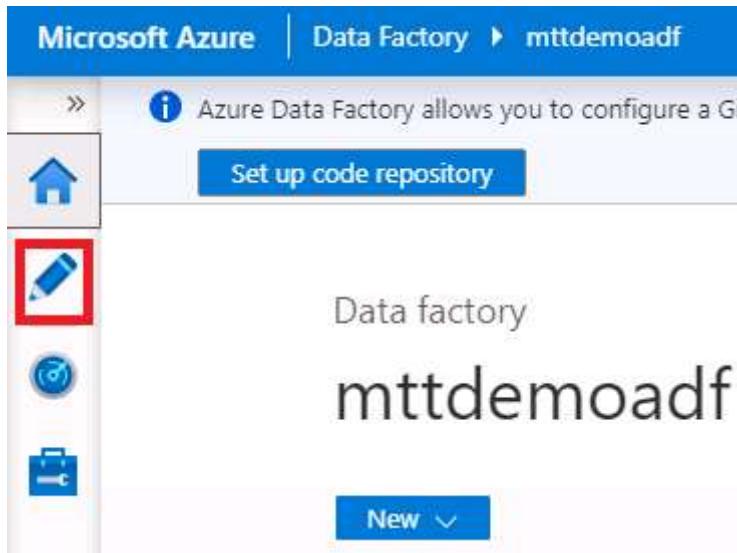
Microsoft Azure | Data Factory > mttdemoadf

» *i* Azure Data Factory allows you to configure a Gi  
Set up code repository

 Data factory

 mttdemoadf

 New ▾



2. Select the + (plus) button and then select **Dataset**

Factory Resources

+ Pipeline

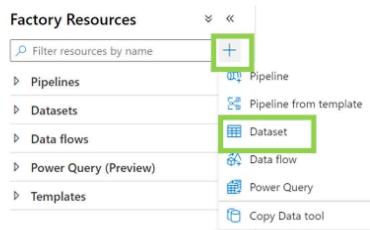
+ Dataset

+ Data flow

+ Power Query (Preview)

+ Templates

Filter resources by name



3. On the New Dataset page, select **Azure Blob Storage**, and then select **Continue**.

New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

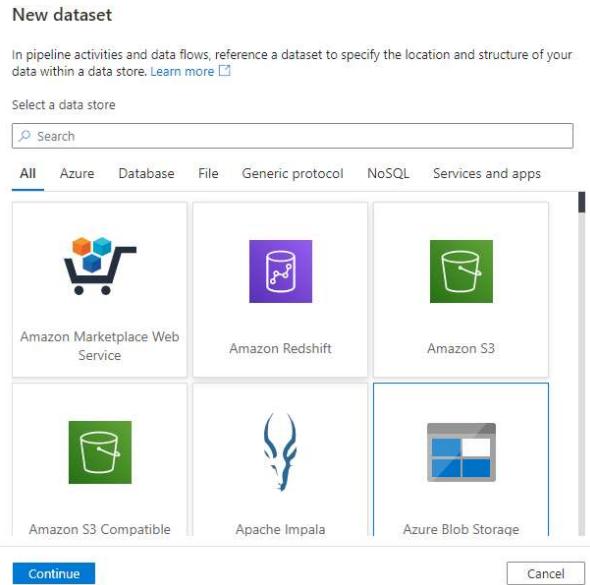
Select a data store

Search

All Azure Database File Generic protocol NoSQL Services and apps

 Amazon Marketplace Web Service	 Amazon Redshift	 Amazon S3
 Amazon S3 Compatible	 Apache Impala	 Azure Blob Storage

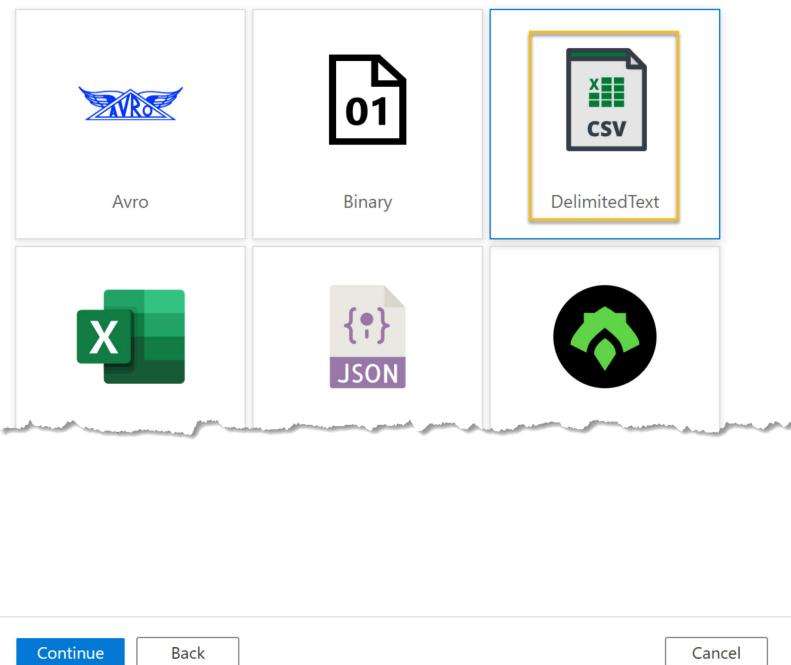
Continue Cancel



4. Since we are using a comma separated file, select **DelimitedText** and click the **Continue** button.

## Select format

Choose the format type of your data



5. Give the data set a name, such as InputDataset and select the Azure Storage linked service you created in the previous exercise.
6. Select the folder button to the right of the File path.



7. Then to navigate to the input file (adftutorial/input/emp.txt) and click the OK button.

Set properties

Name

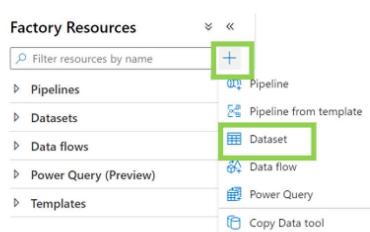
Linked service \*  [edit](#)

File path  [Browse](#) | [OK](#)

First row as header

Import schema  From connection/store  From sample file  None

8. Next create the output dataset, Select the + (plus) button, and then select Dataset.



9. On the New Dataset page, select Azure Blob Storage, and click the Continue button.
10. Select DelimitedText and click the Continue button.
11. Specify OutputDataset for the name and select the Azure Linked Service created in the previous exercise.
12. Enter adftutorial as the file path container and output for directory.
13. Select None for Import Schema then select OK.

Set properties

Name

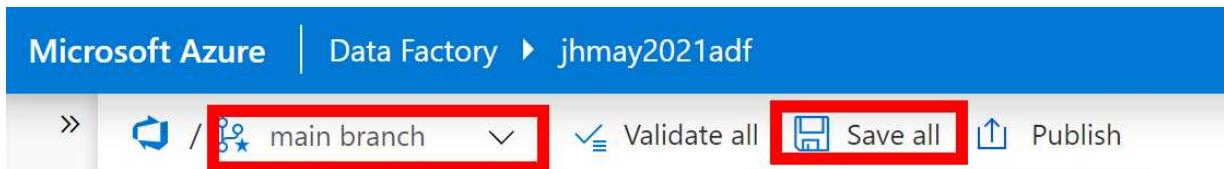
Linked service \*  [Edit](#)

File path  /  /  [Browse](#) [...](#)

First row as header

Import schema   
 From connection/store  From sample file  None

14. Click ok. Choose **Save all** from the menu bar to save your changes to your DevOps repo. If you do not see Save all, make sure you have a DevOps branch selected rather than Data Factory.



15. Select Publish All then Publish on the next screen.



Exercise 2 has been completed.

## Exercise 3: Create a Pipeline

In this exercise, you create and validate a pipeline with a Copy activity that uses the input and output datasets created in a previous exercise. The copy activity copies data from the file you specified in the input dataset settings to the file you specified in the output dataset settings. If the input dataset specifies only a folder (not the file name), the copy activity copies all the files in the source folder to the destination.

### Tasks

1. Select the + (plus) button under Factory Resources, and then select **Pipeline**.

The screenshot shows the Azure Data Factory Pipeline designer. On the left, there's a sidebar with icons for Home, Factory Resources, Pipelines, and Datasets. The 'Factory Resources' tab is selected. In the main area, there's a search bar labeled 'Filter resources by name' and a '+' button. Below it, there are two tabs: 'Pipeline' and 'Dataset'. The 'Pipeline' tab is selected, showing a pipeline named 'outputdataset'.

2. In the Properties pane to the right, specify **CopyPipeline** for Name.
3. In the Activities toolbox, expand **Move & Transform**. Drag the **Copy Data** activity from the Activities toolbox to the pipeline designer surface. You can also search for activities in the Activities toolbox. Specify **CopyFromBlobToBlob** for Name.
4. Navigate to the Source tab in the Copy Data activity settings and select **inputdataset** for Source Dataset.
5. Navigate to the Sink tab in the Copy Data activity settings and select **outputdataset** for Sink Dataset.
6. Choose **Save all** from the menu bar to save your changes to your DevOps repo. If you do not see Save all, make sure you have a DevOps branch selected rather than Data Factory.

The screenshot shows the Microsoft Azure Data Factory interface. At the top, it says 'Microsoft Azure | Data Factory > jhmay2021adf'. Below that is a toolbar with icons for Home, Data Factory, Validate all, Save all (which is highlighted with a red box), and Publish. A dropdown menu shows 'main branch'.

Exercise 3 has been completed.

## Exercise 4: Debug and Trigger the Pipeline Manually

In this exercise, you will debug the pipeline before deploying entities (linked services, datasets, pipelines) to Azure Data Factory. Then, you manually trigger a pipeline run.

### Tasks

1. On the pipeline toolbar above the canvas, select **Debug** to trigger a test run.

The screenshot shows the Microsoft Azure Data Factory interface. At the top, it says 'Microsoft Azure | Data Factory > jhmay2021adf'. Below that is a toolbar with icons for Home, Data Factory, Validate all, Save all, and Publish. A dropdown menu shows 'main branch'. The 'Activities' tab is selected in the top navigation bar. At the bottom of the screen, there's a toolbar with icons for Home, Factory Resources, Pipelines, and Datasets. The 'Activities' tab is selected. The 'Debug' button is highlighted with a red box.

2. Confirm that you see the status of the pipeline run on the **Output** tab of the pipeline settings at the bottom.

Copy data

CopyFromBlobtoBlob

Parameters Variables Settings Output

Pipeline run ID: fa876461-2ae6-4b4e-abf8-e6b6ff475b2f

Name	Type	Run start	Duration	Status
CopyFromBlobtoBlob	Copy data	2021-05-28T18:14:40.956	00:00:01	Queued

3. In a different web browser tab, go to your Azure Portal, navigate back to the Azure Storage account containing the **adftutorial** container.

4. From the Data storage section in the left pane, select **Containers**.

Storage accounts

jhmay2021blob | Containers

Overview Activity log Tags Diagnose and solve problems Access Control (IAM) Data migration Events Storage Explorer (preview)

Search containers by prefix

Name

adftutorial

5. Select the **adftutorial** container.

6. Confirm that you see the output file, emp.txt, in the output folder.

Storage accounts

jhmay2021blob | Containers

adftutorial

Overview

Access Control (IAM)

Add New

emp.txt

Name	Modified	Access tier	Block type	Size	Last state
emp.txt	5/28/2021, 22:50:00	Hot (Inferred)	Block Blob	28 B	Available

7. Run/Trigger pipeline.

8. Go back to the Data Factory portal with your Data Factory selected.

9. Before you trigger a pipeline, you must publish entities to Data Factory. To publish, select **Publish** at the top of the portal.

10. To trigger the pipeline manually, select **Add Trigger** on the pipeline toolbar, and then select **Trigger Now**.

11. Select **OK** in the Pipeline run flyout.

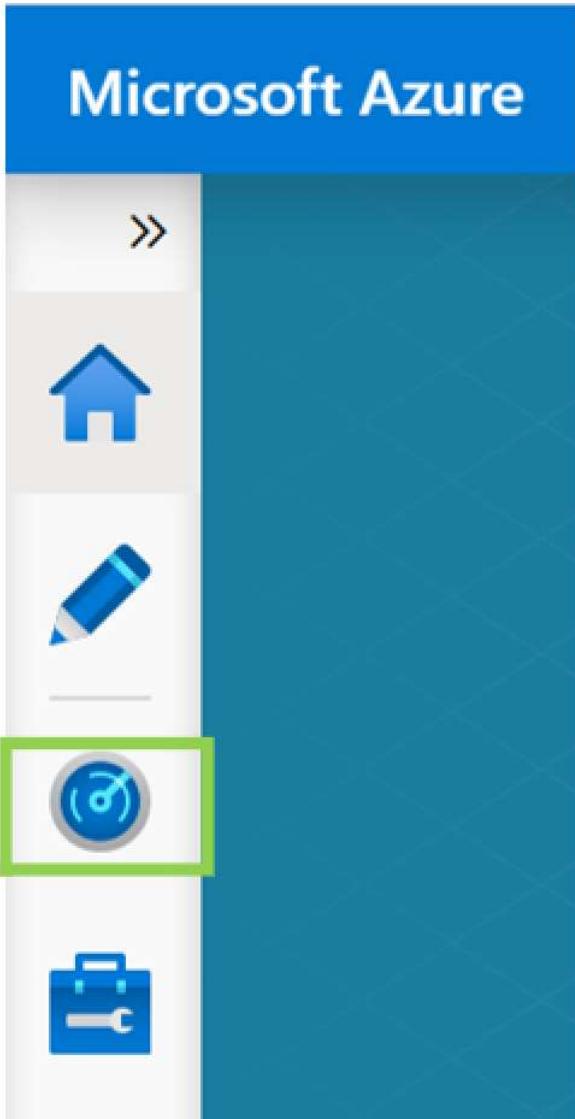
Exercise 4 has been completed.

## Exercise 5: Monitor the Pipeline

### Tasks

Monitor your pipeline.

1. Switch to the **Monitor** tab on the left.



2. Under **Pipeline runs**, under **Pipeline name**, select the **CopyPipeline** hyperlink. Here you can see the status of the copy activity run on this page.

3. To view details about the copy operation, select the **Details** (eyeglasses image) link in the Actions column. For details about the properties, see [Copy Activity overview](#).

All Succeeded In Progress Timed Out Failed Cancelled

ACTIVITY NAME	ACTIVITY TYPE	ACTIONS
CopyFromBlobtoBlob	Copy	→ ⏪ ⚡

Exercise 5 has been completed.



## Copy Data Wizard

---

### Introduction

This lab walks through the Copy Data Wizard.

In this lab we will:

- Show the graphical user interface for Copying Data

### Estimated Time

20 minutes

### Objectives

At the end of this lab, you will be able to:

- Use the copy data wizard

### Login Information

Use the following credentials to sign into the virtual environment:

- Username: **Admin**
- Password: **PasswOrd!**

## Table of Contents

---

[Exercise 1: Upload Sample Data](#)

[Exercise 2: Copy Data via the Wizard](#)

[Lab: Azure Data Factory Copy Data Wizard](#)

During this lab, you will learn how to copy data with the Copy Data wizard.

## Exercise 1: Upload Sample Data

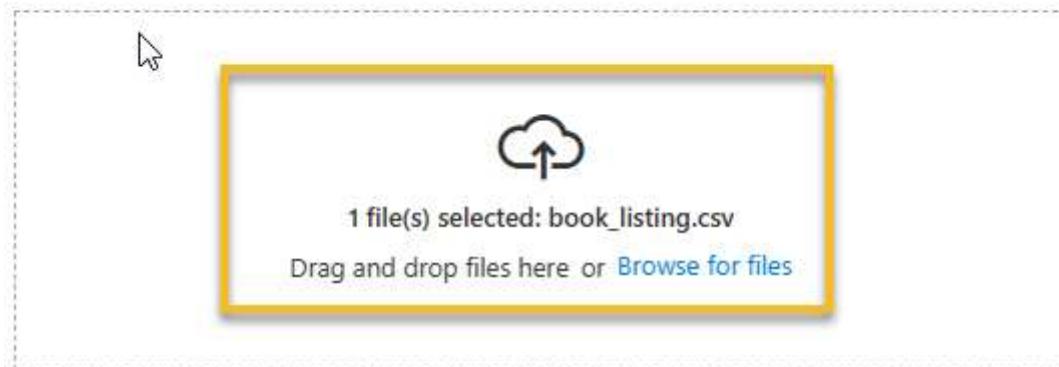
Throughout this workshop's labs, you will be utilizing various Azure resources outside of Data Factory.

### Tasks

1. From the Resource Group you created in **Module 1, Lesson 1, Lab 1**, select the **Storage Account** created in the lab.
2. On the Storage account page, select **Containers** under the **Data storage** section.
3. Select the container **adftutorial**.
4. Upload sample .txt file to Container.
  - a. On the Container page, select **Upload** on the toolbar.
  - b. On the Files selection box, click on the **folder icon** to upload a file. Navigate to the **\LabFiles\M02\_L01\_Lab02** folder and select the file **book\_listing.csv** and **Open** it.
  - c. Select **Advanced** to show additional options.
  - d. Under the **Upload to folder box** type **input/books**.
  - e. Click **Upload**.

## Upload blob

X



Overwrite if files already exist

Advanced

Blob type ⓘ

Block blob

Upload .vhdx files as page blobs (recommended)

Block size ⓘ

4 MiB

Access tier ⓘ

Hot (Inferred)

Upload to folder

input/books

- f. Confirm that the folder is input, and the file is **book\_listing.csv**, and **Upload**. You should see the book\_listing.csv file and the status of the upload in the list.

The screenshot shows the Azure Storage Blob container 'adftutorial'. The 'Overview' tab is selected. In the 'Location' field, the path 'adftutorial / input / books' is shown, with 'input / books' highlighted by a red box. Below the location field is a search bar with the placeholder 'Search blobs by prefix (case-sensitive)'. To the right of the search bar is a 'Add filter' button. On the left, there's a sidebar with 'Settings' and several options: Shared access tokens, Access policy, Properties, Metadata, and Editor (preview). The 'book\_listing.csv' file is listed under the 'Name' column.

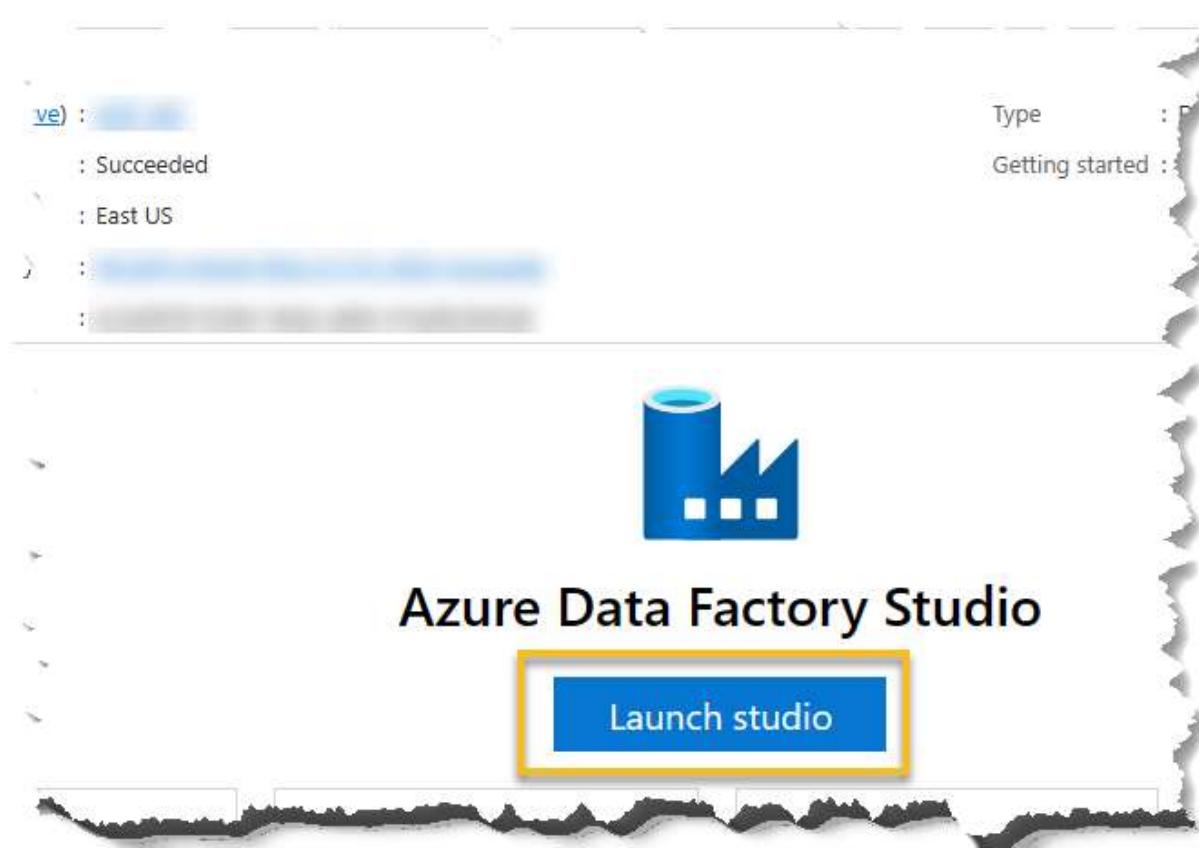
Exercise 1 has been completed.

## Exercise 2: Copy Data via the Wizard

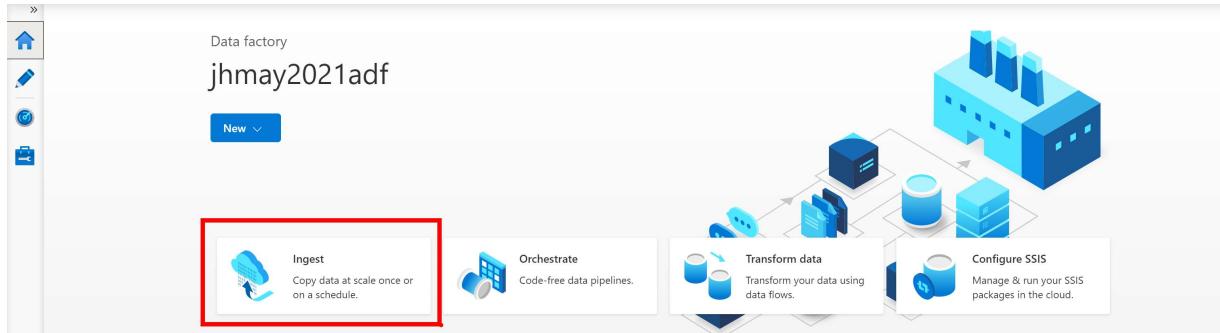
### Tasks

1. Launch your Azure Data Factory in a separate browser tab.

In the Azure portal, navigate to the ADF instance you previously provisioned, and then launch the Azure Data Factory UI by selecting Open in the ADF Overview blade.



2. On the ADF UI landing page, select Ingest.



3. On the Properties Page of the Copy Data wizard

a. Set the configurations as such:

Task Type: Choose Built-in copy task:

Task cadence or Task schedule: Choose Run once now.

4. Select Next.

1 Properties

2 Source

3 Target

4 Settings

5 Review and finish

Use Copy Data Tool to perform a one-time or scheduled data load from 90+ data sources. Follow the wizard experience to specify your data loading settings, and let the Copy Data Tool generate the artifacts for you.

### Properties

Select copy data task type and configure task schedule

**Task type**

 **Built-in copy task**  
You will get single pipeline to copy data from 90+ data source easily.

 **Metadata-driven copy task (Preview)**  
Metadata is required to be stored in external control tables to load data at large-scale.

You will get single pipeline to quickly copy objects from data source store to destination in a very intuitive manner.

**Task cadence or task schedule \***

Run once now  Schedule  Tumbling window

< Previous **Next >**

5. Configure the Source to point to the Azure Blob Storage Linked Service.

- a. On the Source dataset page, select source type Azure Blob Storage and select the AzureStorageLinkedService created in last lab
- b. Click the Folder button next to the word **Browse** to drill down to the input file **adftutorial/input/books/book\_listing.csv** and select **Choose**.
- c. Then click **Next**.

## 6. Validate the File Format settings

- Inspect the file. Notice it has the column names in the first row.

## 7. Select **Next** to finish the Source.

## 8. Configure the Destination to point to the Azure Blob Storage Linked Service.

- On the Target dataset page, select source type Azure Blob Storage and select the AzureStorageLinkedService created in last lab
- In the Folder path type in adftutorial/output/booklist
- Enter the filename of book\_listing.csv

d. Ensure Copy behavior specifies None and Max Concurrent connections is blank

e. Click next

Microsoft Azure | Data Factory > metadatagewooadf

Search

Copy Data tool

Properties

Source

Destination

Dataset

Configuration

Settings

Review and finish

Destination data store

Specify the destination data store for the copy task. You can use an existing data store connection or specify a new data store.

Destination type: Azure Blob Storage

Connection: AzureStorageLinkedService

Folder path: adftutorial/output/booklist

File name: book\_listing.csv

Copy behavior: None

Max concurrent connections: (blank)

Block size (MB): (blank)

Metadata: New

Browse

9. Validate the file format settings

a. Make sure Add header to file is chosen

b. Click next

Microsoft Azure | Data Factory > metadatagewooadf

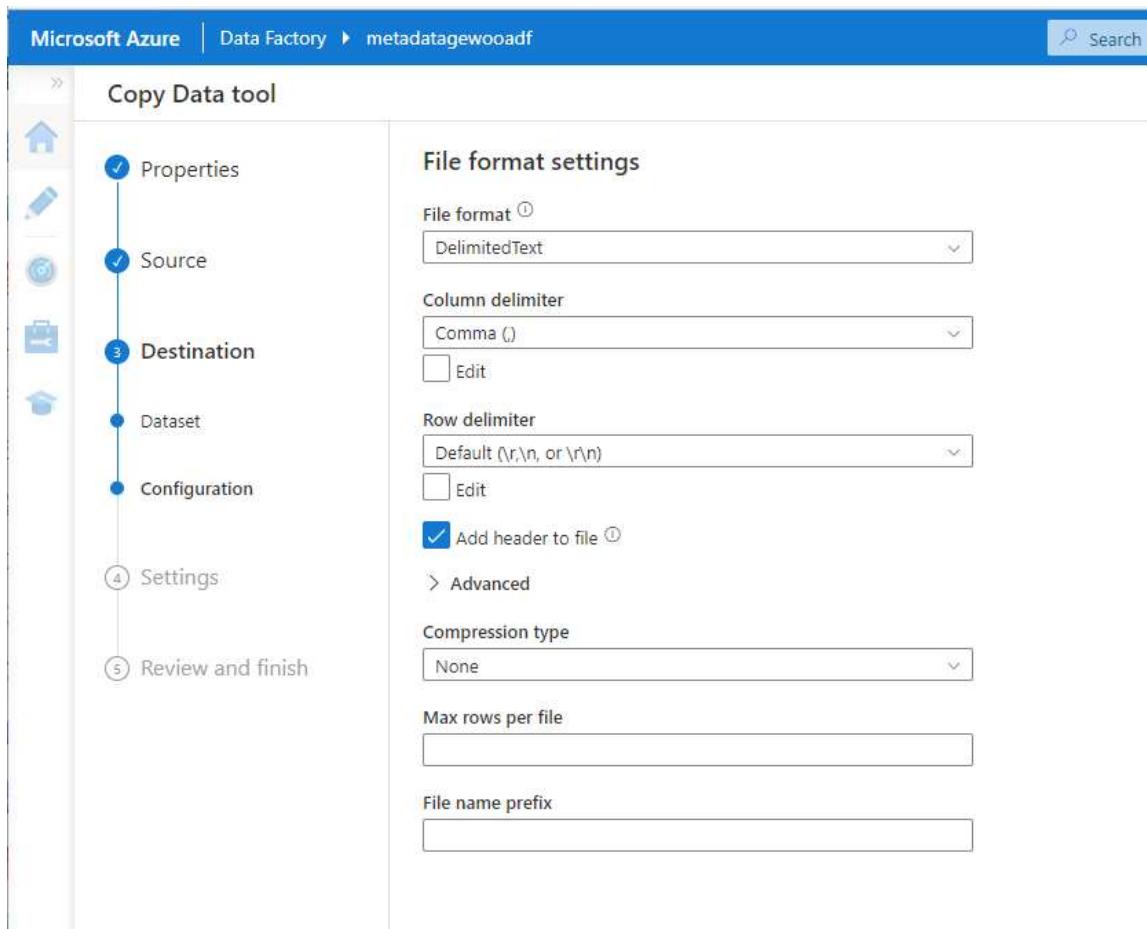
Search

### Copy Data tool

Properties  
Source  
Destination  
Dataset  
Configuration  
Settings  
Review and finish

**File format settings**

File format  Column delimiter   Edit Row delimiter   Edit  Add header to file  Advanced Compression type  Max rows per file  File name prefix



10. Name the Pipeline **CopyBooksListing** and click next.

Microsoft Azure | Data Factory > adfworkshopupg

### Copy Data tool

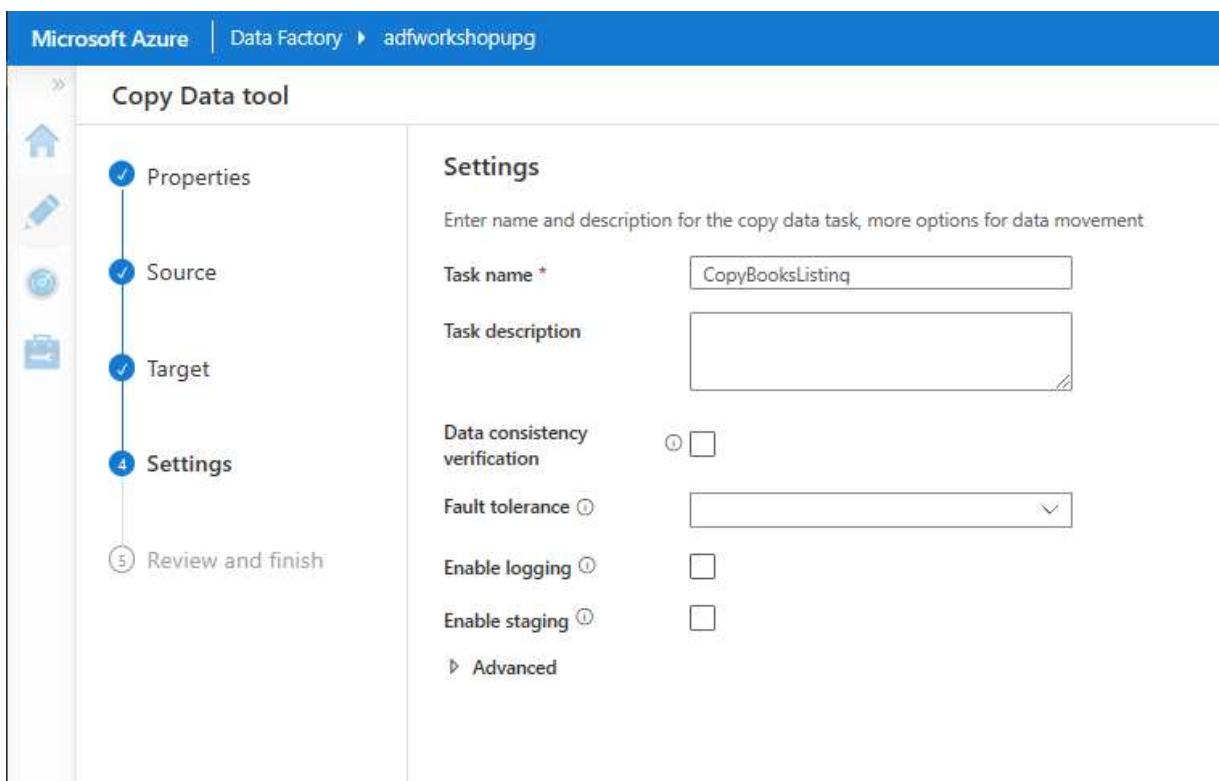
Properties  
Source  
Target  
Settings  
Review and finish

**Settings**

Enter name and description for the copy data task, more options for data movement

Task name \*  Task description

Data consistency verification  Fault tolerance   Enable logging   Enable staging  Advanced



11. View the summary and click next

Microsoft Azure | Data Factory > adfworkshopug

Copy Data tool

Properties  
Source  
Target  
Settings  
Review and finish  
Review  
Deployment

**Summary**

You are running pipeline to copy data from Azure Blob Storage to Azure Blob Storage.

Azure Blob Storage → Azure Blob Storage

**Properties**

Task name: CopyBooksListing

**Source**

Connection name: AzureStorageLinkedService

Dataset name: SourceDataset\_mh

Column delimiter: ,

Escape character: \

Quote char: "

First row as header: true

File name: book\_listing.csv

Folder path: input/books

Container: adftutorial

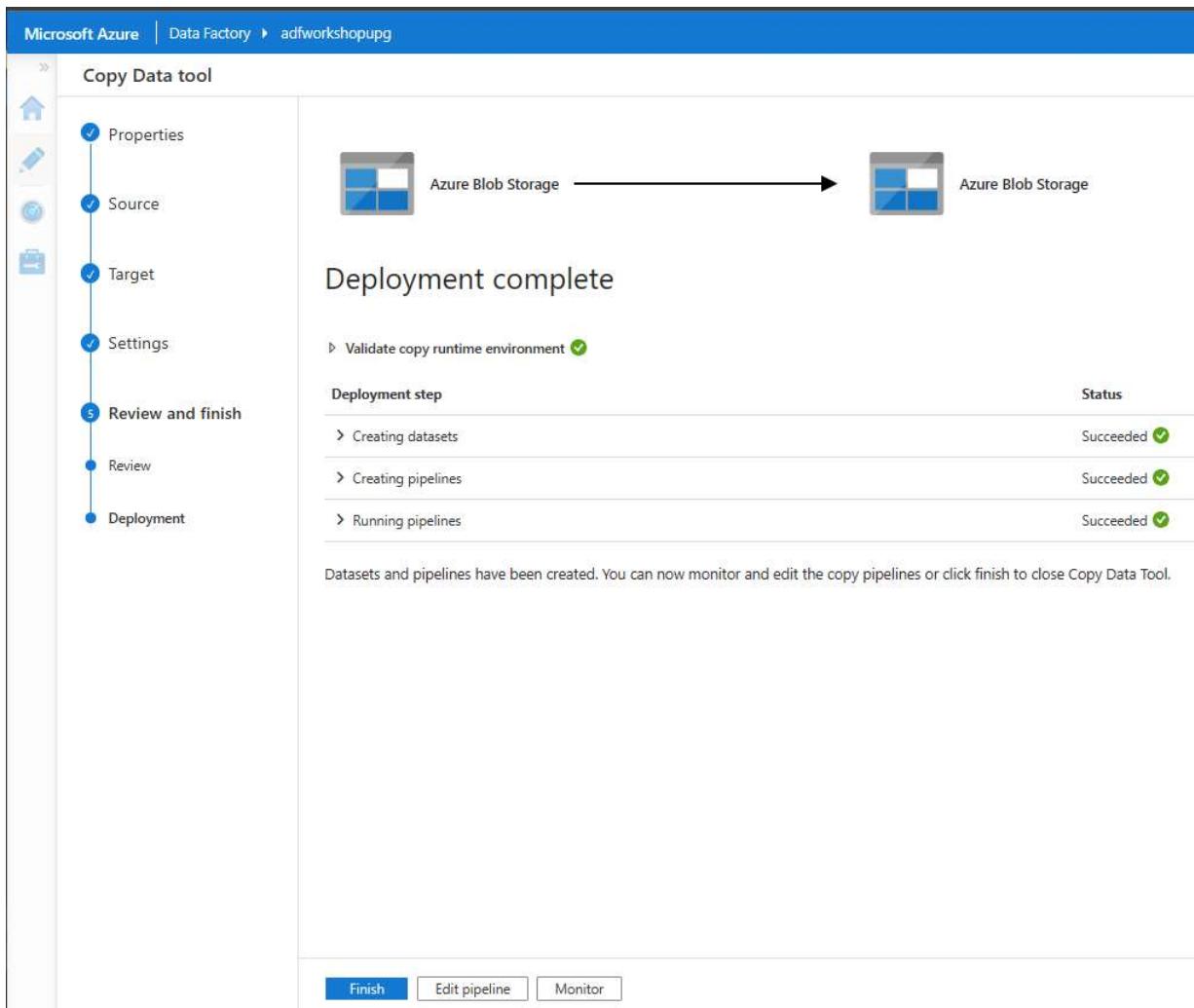
**Target**

Connection name: AzureStorageLinkedService

Dataset name: DestinationDataset\_mh

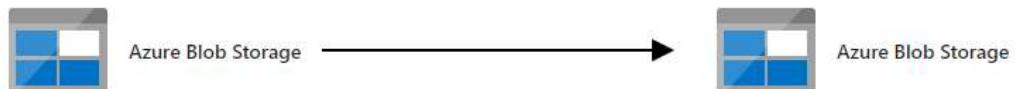
Column delimiter: ,

12. If you Do NOT have Git integration enabled in your Data Factory, click Finish and then go to step 14.



13. If you have Git enabled, you will need to manually Publish and Trigger the pipeline.

a. Verify that your datasets and pipeline has been deployed and click **Finish**.



Deployment complete

Deployment step	Status
> Creating datasets	Succeeded ✓
> Creating pipelines	Succeeded ✓

**Finish**   **Edit pipeline**   **Monitor**

- b. Select **Publish** from the menu bar.



- c. Review the **Pending changes** and click **OK**.

- d. Navigate to your **CopyBooksListing** pipeline. Select **Add trigger** from the menu bar, choose **Trigger now** and click **OK**.

14. Monitor the pipeline run

- a. Click on the Monitor button to the far-left.

# Microsoft Azure

>>



15. Click the pipeline under Pipeline Name to see the progress of the individual activities that make up the pipeline.

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has a 'Pipeline runs' item selected. The main area is titled 'Pipeline runs' and shows one item: 'CopyBooksListing'. A search bar at the top right says 'Search by run ID or name'. Below the search bar, it says 'Showing 1 - 1 items'. There are filter buttons for 'Triggered', 'Debug', and 'Rerun', with 'Triggered' being the active tab. A checkbox labeled 'Pipeline name' is checked, and the pipeline name 'CopyBooksListing' is highlighted with a red box.

16. Select the various inputs, outputs, and run icons under Activity runs, Activity Name.

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has a 'Activity runs' item selected. The main area is titled 'Activity runs' and shows one item: 'CopyBooksListing'. It displays details like 'Pipeline run ID: 8d03ab42-336b-4c0c-b79e-8f849a367070'. A table lists the activity run details:

Activity name	Activity type	Run start ↑	Duration	Status	Error	Integration runtime	User proper...	Destination
CopyBooksListing	Copy data	6/1/21, 10:49:22 AM	00:00:10	Succeeded		DefaultIntegrationRuntime (East US 2)		dwtemp/03.02/

The first three columns ('Activity name', 'Activity type', and 'Run start') have their respective icons highlighted with a red box.

Exercise 2 has been completed.



## Data Transformation

---

### Introduction

During this lab, you will learn how to perform data transformations using Azure Data Factory. This will cover data transformation using Azure Databricks and Azure Data Factory.

### Estimated Time

45 minutes

### Objectives

At the end of this lab, you will be able to:

- Configure a Data Transformation pipeline and perform ELT process

### Logon Information

Use the following credentials to sign into the virtual environment

- Username: **Admin**
- Password: **Passw0rd!**

## Table of Contents

---

### Prerequisites

[Exercise 1: Enable Microsoft.Compute](#)

[Exercise 2: Clone the Databricks Archive](#)

[Exercise 3: Data Ingestion using Azure Databricks](#)

[Exercise 4: Saving the Storage Account Name and Access Key](#)

[Exercise 5: Review Ingested Data in Azure Databricks](#)

[Exercise 6: Data Transformation in Azure Data Factory using Azure Databricks Notebook](#)

[Exercise 7: Review the Transformed Data using Azure Databricks](#)

## Lab: Data Transformations with Azure Data Factory

During this lab, you will learn how take a sample dataset to create a Data Factory pipeline and use sample notebooks in Azure Databricks to transform and analyze the data.

### Prerequisites

You should have your storage account, databricks, and data factory resources provisioned from **Module 1, Lesson 2, Lab 2** using the Azure Resource Manager (ARM) templates. If not, please refer back to that lab.

### Exercise 1: Enable Microsoft.Compute

#### Tasks

##### 1. Enable Microsoft.Compute

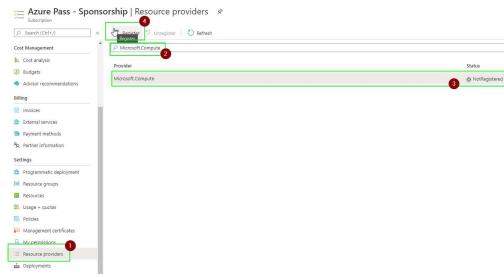
- In the Azure portal, select **Subscriptions**.



- Select **your subscription**.

The screenshot shows the 'Subscriptions' page in the Azure portal. At the top, there's a blue header bar with the Microsoft Azure logo and a 'Home >' link. Below it is a navigation bar with 'Subscriptions' highlighted. The main area displays a table of subscriptions. The first column is 'Subscription name', which has a green border around it. The second column is 'Status'. One row is selected, showing 'Azure Pass - Sponsorship' in the 'Subscription name' column and 'Unregistered' in the 'Status' column. There are also 'Add', 'My role', 'Apply', and search/filter buttons.

- Select Resource providers under Settings and search for **Microsoft.Compute**. Ensure it is registered. If not registered, **register it**. (Note that it takes a few minutes for the status to change from Unregistered to Registered. Refresh your browser to get an updated status, do not click the refresh button beside unregister.



Exercise 1 has been completed.

## Exercise 2: Clone the Databricks Archive

This exercise will create the Databricks notebooks for the lab.

## Tasks

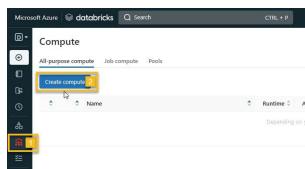
- ## 1. Launch the Databricks workspace

- a. In the Azure portal, go to the resource group you created in M01\_L02\_Lab01 and select the **Databricks service**.
    - b. Select **Launch Workspace**.



- ## 2. Create your cluster.

- a. Select **Compute** on the side panel, then +**Create Compute**.

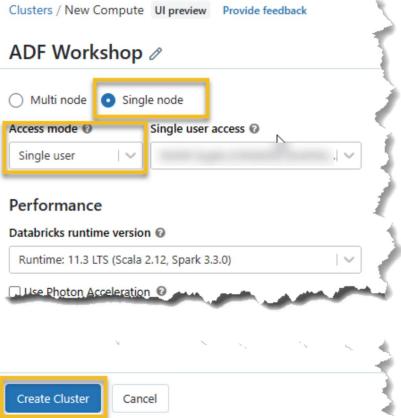


- b. Click on the pencil and Enter a Cluster Name (for example: *ADF Workshop*)

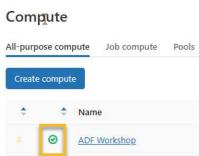
- c. Select **Single Node**.

- d. Set Access Mode to **Single user access** and it should have your email account listed

- e. Leave everything else as default and click **Create Cluster**.



f. Wait for the cluster to be created before continuing, which may take a couple of minutes.



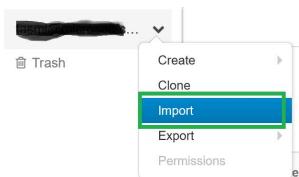
- If you get an error for number max number of IP reached. It means you might have created a multi-node cluster instead. In the sponsor tenant, you are only allowed maximum of 10 public IP addresses. Close the data bricks and delete the data bricks in your resource group to clean up everything. Create data bricks manually and start from step 2 again.

### 3. Import the notebook

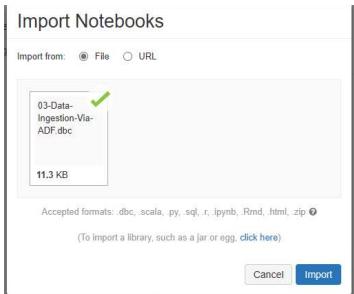
- a. On the left side, select **Workspace**, then **Users**, and select your **username** (the entry with the house icon).



- b. In the window that appears, select the downward arrow next to your name and select **Import**.



- c. In the Import Notebooks window, select import and browse to the M03\_L01\_Lab01 folder, and choose the file 03-Data-Ingestion-Via-ADFdbc



- d. Click the **Import** button.
- e. Select the folder named **03-Data-Ingestion-Via-ADF**.

**i** The folder contains 3 notebooks for this lab (two note books to be used in Databricks and one in Azure Data Factory). Run these in order, as shown below.

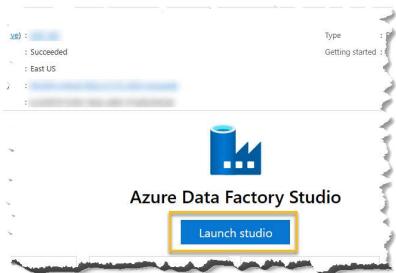
Exercise 2 has been completed.

## Exercise 3: Data Ingestion using Azure Databricks

This exercise will create a Data Factory pipeline to ingest data from a public dataset into your storage account. After the data is ingested, you use a Databricks notebook function to examine the data.

### Tasks

1. Launch your Azure Data Factory in a separate browser tab. In the Azure portal, navigate to the ADF instance you previously provisioned, and then launch the Azure Data Factory UI by selecting **Launch Studio** in the ADF Overview blade.

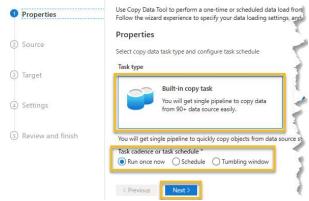


2. On the ADF UI landing page, select Ingest.



3. On the Properties Page of the Copy Data wizard

- a. Set the configurations as such:
  - i. Task Type: Choose Built-in copy task:
  - ii. Task cadence or Task schedule: Choose Run once now.
  - iii. Select Next.



b. Configure the Source to point to the Azure Blob Storage Linked Service.

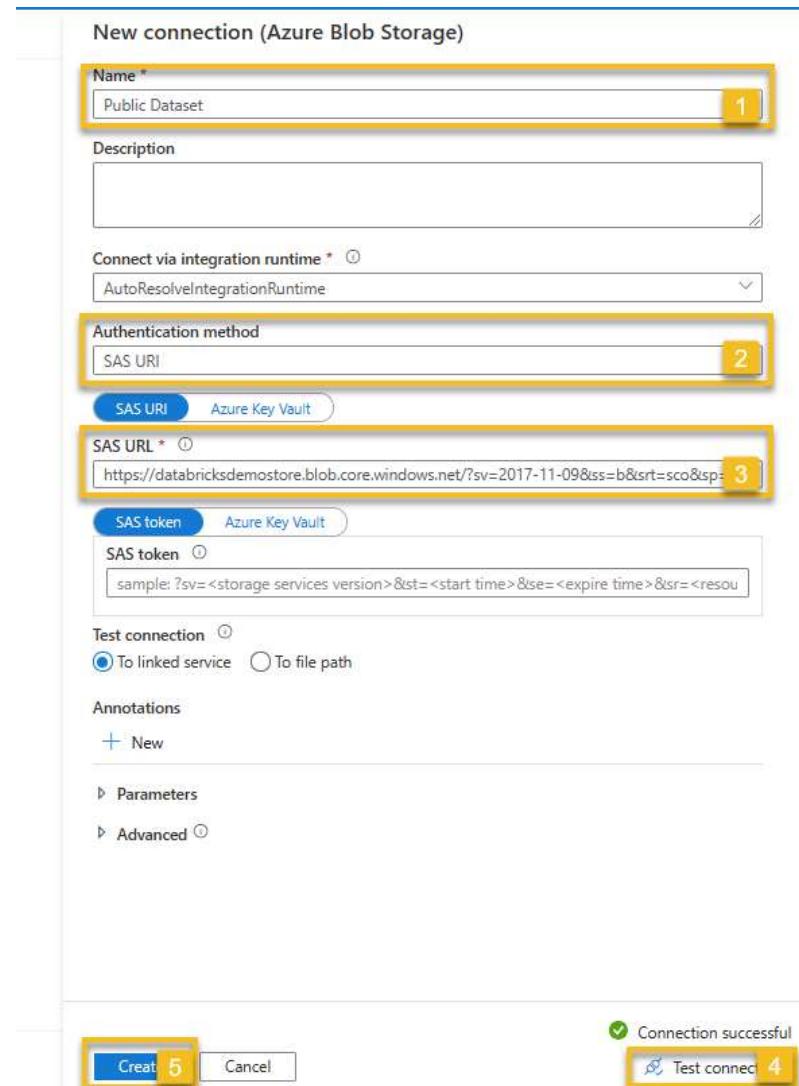
- On the Source dataset page, select source type Azure Blob Storage and select + New connection.



- Configure the New Linked Service (Azure Blob Storage).

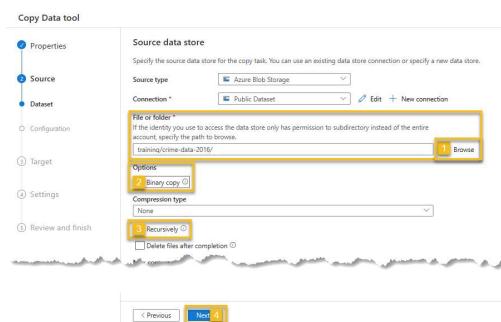
- Name: **PublicDataset**
- Authentication method: **SAS URI**
- **SAS URL:**    
 It will be helpful to type the URL into a notepad first. Before copying into to Azure Data Factory.

- Select **Test connection** and ensure a Connection successful message is displayed.
- Select **Create**.



c. Back on the Source data store page, Set the input folder.

- Folder: Use the Browse button to select the folder **training/crime-data-2016/**.
- Binary copy: Check this box.
- Recursively: Check this box.
- Select **Next**.



d. Configure the Target to point to Azure Blob Storage Account.

- i. On the Target data store page, select source type Azure Blob Storage and select + New connection.



- ii. Configure the New Linked Service (Azure Blob Storage).

- Name: **DestinationContainer**
- Authentication method: **Account key**
- Account selection method: **From Azure subscription**
- Azure Subscription: **Azure Pass - Sponsorship**
- Storage account name: Select the name of the Storage account you created previously from the list.
- Select **Test connection** and ensure a Connection successful message is displayed.
- Select **Create**.

### New connection (Azure Blob Storage)

**1** Name \*  
DestinationContainer

Description

Connect via integration runtime \* ⓘ  
AutoResolveIntegrationRuntime

Authentication method  
**2** Account key

Connection string   Azure Key Vault

Account selection method ⓘ  
**3** From Azure subscription    Enter manually

Azure subscription ⓘ  
**4**

Storage account name \*  
**5**

Additional connection properties

+ New

Test connection ⓘ  
 To linked service    To file path

Annotations

+ New

+ Parameters

+ Advanced ⓘ

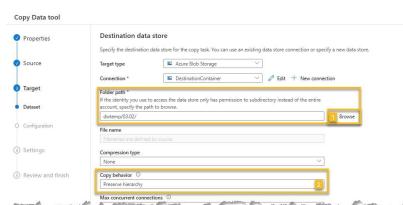
**7** Create   Cancel

**6** Test connection

✓ Connection successful

e. Configure the output file or folder:

- Folder path: dwtemp/03.02/
- File name: Leave empty.
- Copy behavior: Select **Preserve hierarchy**.
- Select **Next**.

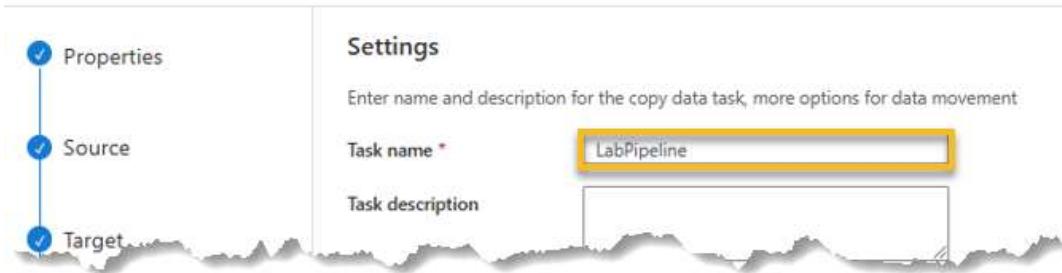


f. On the Settings page,

- i. Enter the name **LabPipeline** and leave the other default values

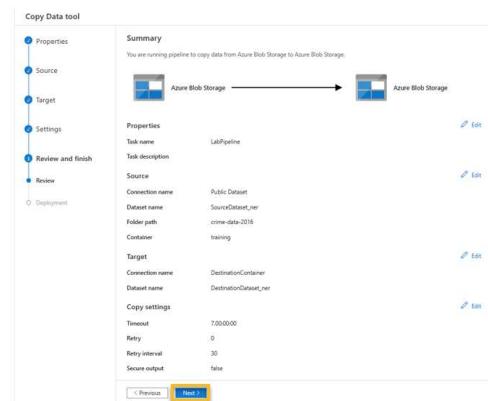
ii. Select **Next**.

### Copy Data tool



g. On the Summary page, you can review the copy pipeline settings

i. Select **Next**.

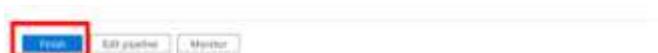


h. If you have DO NOT have Git integration enabled in your Data Factory, click Finish and then go to step 4.



i. If you HAVE Git enabled, you will need to manually Publish and Trigger the pipeline.

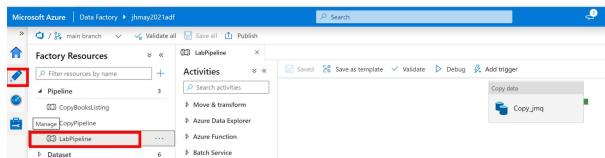
i. Select **Finish**.



ii. Click **Publish** from the top menu bar. Review changes and click **OK**.



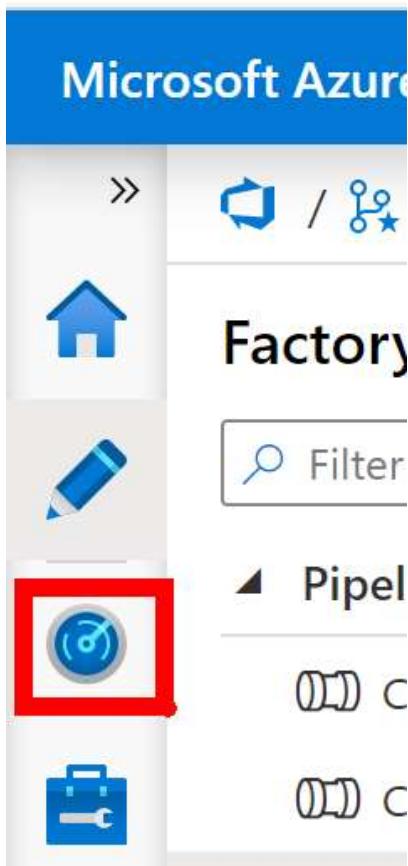
iii. Navigate and open your new pipeline, LabPipeline.



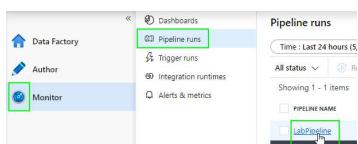
iv. Select **Add trigger** from the menu bar, choose **Trigger now** and click **OK**.



4. Click on the **Monitor** button to the far-left to monitor the pipeline run.



a. Selecting Monitor above will take you to the Pipeline Runs screen in the ADF UI, where you can monitor the status of your pipeline runs. Using the monitor dialog, you can track the completion status of pipeline runs, and access other details about the run. Click the pipeline under Pipeline Name to see the progress of the individual activities that make up the pipeline.

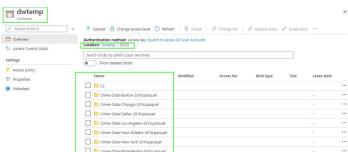


b. Select the various inputs, outputs, and run icons under Activity runs, Activity Name.



## 5. Verify files in blob storage.

- Navigate to your storage account in the Azure portal.
- Locate the **dwtemp** container.
- Locate the **03.02** folder within the dwtemp container.
- Confirm the files were copied via Azure Data Factory.



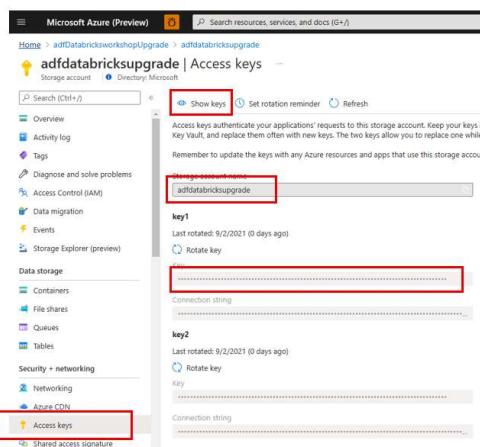
Exercise 3 has been completed.

## Exercise 4: Saving the Storage Account Name and Access Key

In the upcoming exercises you will need your Storage account name and the storage account key.

### Tasks

- The account name and key can be retrieved from the Access keys section in your storage account.
- Copy your storage account name and Select show keys and copy the value under Key 1. Save these both for a future step.



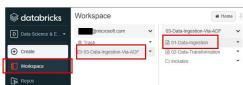
Exercise 4 has been completed.

## Exercise 5: Review Ingested Data in Azure Databricks

In this exercise we will use the Python Notebook we imported in Exercise #2 to explore the ingested data.

### Tasks

1. Navigate back to Databricks.
2. Select the **01-Data-Ingestion** notebook from the **03-Data-Ingestion-Via-ADF** Workspace.



3. Continue with the instructions in the **01-Data-Ingestion** notebook to Examine the ingested data in the notebook.

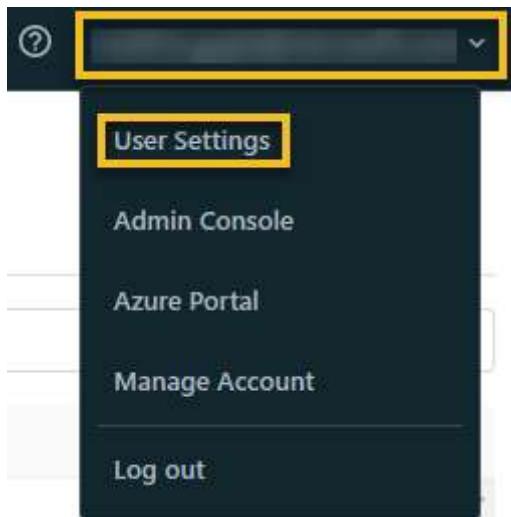
Exercise 5 has been completed.

## Exercise 6: Data Transformation in Azure Data Factory using Azure Databricks Notebook

This exercise will add a Databricks Notebook activity to your ADF pipeline that runs a sample notebook to transform and restructure your data.

### Tasks

1. Create Databricks Access Token. This step will generate an access token that will be used by ADF to access the Databricks workspace.
  - a. In the top right corner of your Databricks workspace (this window), select the Account icon, and then select User Settings.



- b. On the User Settings screen, select Generate New Token.

## User Settings

Access tokens    Git integration    Editor settings    Email preferences    Language settings

Personal access tokens can be used for secure authentication to the [Databricks API](#) instead of passwords.

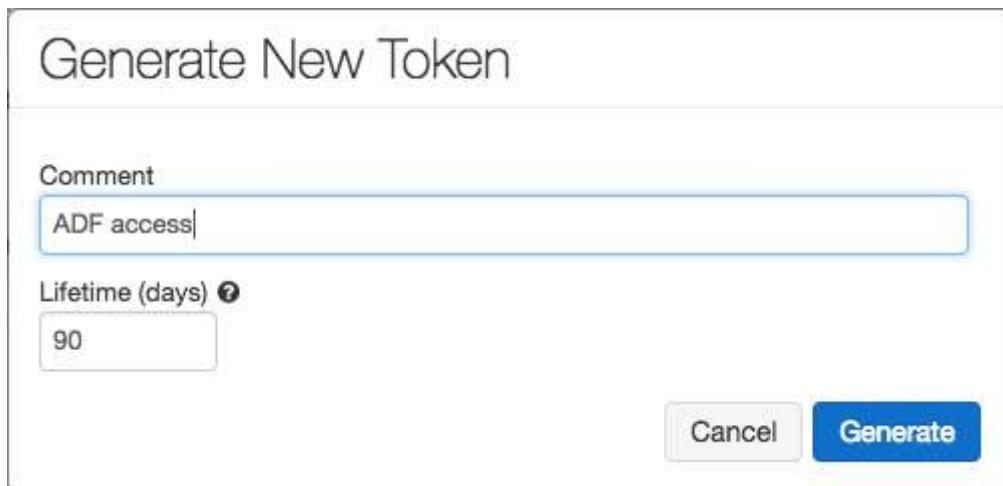
[Generate new token](#)

Comment

Creation ↑

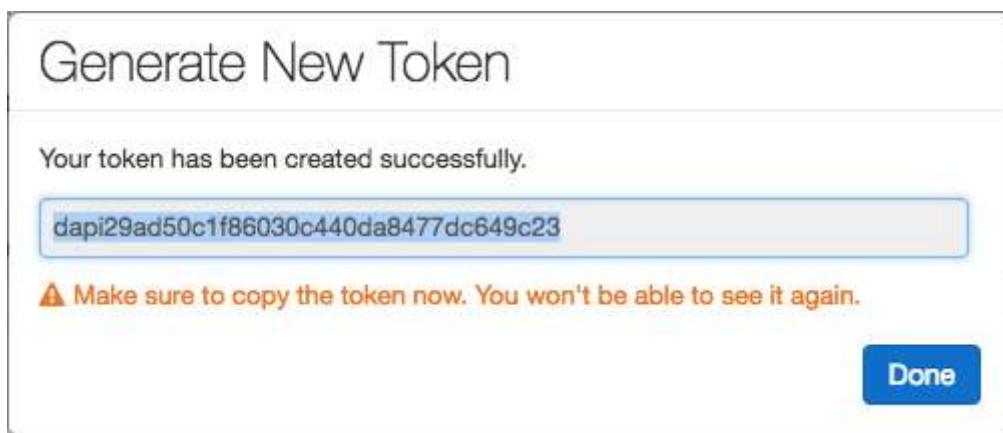
No tokens exist.

- c. In the Generate New Token dialog, enter a comment, such as "ADF access", leave the Lifetime set to 90 days, and select Generate.



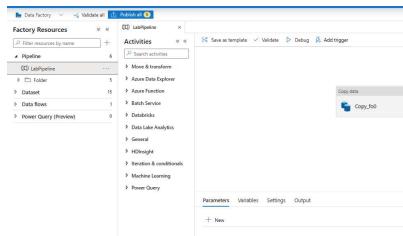
- d. Copy the generated token and paste it into a text editor, such as Notepad, for use when setting up the Databricks Linked Service in ADF.

- Do not close the dialog containing the generated token until have save it in a text editor, as it will not be visible again once you close the dialog.

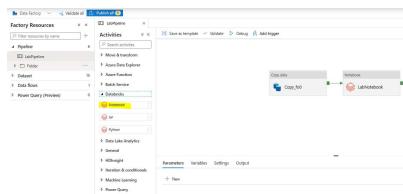


2. Add Databricks Notebook Activity to Pipeline. Navigate back to the Azure Data Factory.

- a. Select the Author (pencil) icon in the left-hand menu, and select the pipeline created in Exercise 1 (LabPipeline).



- b. Expand Databricks under Activities, then drag the Notebook activity onto the design surface, and drop it to the right of the existing copy activity.



- c. Select the Notebook activity on the design surface to display tabs containing its properties and settings at the bottom of the screen.

- d. In the General Tab enter **LabNotebook** into the Name field.

General	Azure Databricks	Settings	User properties
Name *	<input type="text" value="LabNotebook"/> <a href="#">Learn more</a>		
Description	<input type="text"/>		
Timeout ⓘ	<input type="text" value="7.00:00:00"/>		
Retry ⓘ	<input type="text" value="0"/>		
Retry interval ⓘ	<input type="text" value="30"/>		
Secure output ⓘ	<input type="checkbox"/>		
Secure input ⓘ	<input type="checkbox"/>		

- e. In the Azure Databricks Tab Select +New next to Databricks Linked Service to create a new Linked Service

The screenshot shows the 'Azure Databricks' tab selected in the top navigation bar. Below it, there is a dropdown menu labeled 'Databricks linked service \*' with a 'Select...' option and a 'New' button.

f. Configure the New Linked Service.

- Name: **AzureDatabricks**
- Azure Subscription: **Azure Pass - Sponsorship**
- Databricks workspace: Select the workspace you are using for this lab.
- Select cluster: Choose Existing interactive cluster.
- Access token: Paste the token you generated in step one into this field.
- Existing cluster id: Select the Databricks cluster you are using for this lab.

If this drop down box does not populate, ensure that you copied your Access Token correctly into the Access token field.

- Select **Test connection** and ensure you get a Connection successful message.
- Select **Create**.

The form is titled 'New linked service (Azure Databricks)'. It includes fields for Name (set to 'AzureDatabricks'), Description (empty), Connect via integration runtime (set to 'AutoResolveIntegrationRuntime'), Account selection method (set to 'Enter manually'), Databrick Workspace URL (set to 'http://adb...azuredatabricks.net'), Authentication type (set to 'Access Token'), and an Access token field containing '.....'. Below these, the 'Select cluster' section has 'Existing interactive cluster' selected and an Existing cluster ID field containing '.....'. At the bottom, there are sections for Annotations, Parameters, and Advanced settings.

g. In the Settings Tab for Notebook, select Browse, and select the Databricks-Data-Transformations notebook from the **/Users/\*[your-user-account]/\*03-Data-Ingestion-Via-ADF/includes/Databricks-Data-Transformations**.

- Make sure you select the notebook in the includes folder selected. Students commonly select the notebook in the higher level folder and then they have errors or they only select the folder and not the notebook.



h. Expand Base Parameters.

- i. Select +New, three times for three parameters we need to pass to **Databricks-Data-Transformations** notebook. The values for these are copied in Exercise #4.

Name	Value Expected
accountName	Storage Account Name
accountKey	Storage Key 1
containerName	dwtemp

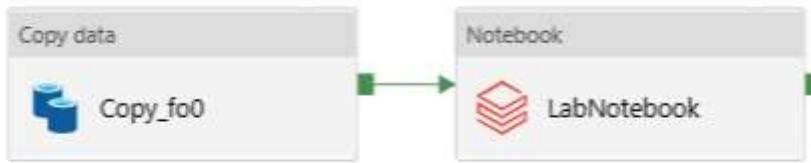
Notebook path \* /Users/[REDACTED]@microsoft.com/03-Data-Inc [Browse](#)

**Base parameters**

+ New	Delete
Name	Value
accountName	blob
accountKey	WbSzgHqO79IV7XvqaoXh22uFBei2yvl...
containerName	dwtemp

- The parameters names are case sensitive in Azure Databricks.

3. Connect Copy Activity to Notebook Activity. Select the small green box on the right-hand side of the copy activity and drag the arrow onto the Notebook activity on the design surface. Creating this connection sets the output from the copy activity as required input for the Databricks Notebook activity. What this means is that the copy activity has to successfully complete processing and generate its files in your storage account before the Notebook activity runs, ensuring the files required by that Notebook activity are in place at the time of execution.

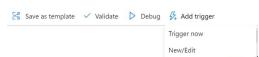


4. Publish Pipeline. In the ADF UI toolbar, if you have Git configured, select **Save all** to save your changes to your Git repo. Then select **Publish** to deploy your changes to the Azure Data Factory service.



## 5. Trigger and Monitor the Pipeline Run

a. Trigger the pipeline by selecting Add trigger, then Trigger Now on your pipeline.



b. Next, navigate to the Monitor screen by selecting the Monitor icon in the left-hand menu.



- c. Click the pipeline under Pipeline Name to see the progress of the active pipeline run. Here you can observe the ADF pipeline activities running. Notice the Databricks Notebook activity will not run until of the copy activity has completed. Once done, you will see a Succeeded message next to each activity.

The screenshot shows the 'Pipeline runs' section of the Azure Data Factory interface. On the left, there's a sidebar with 'Data Factory', 'Author', and 'Monitor' options; 'Monitor' is selected and highlighted with a green box. The main area has a header 'Pipeline runs' with a sub-header 'Time : Last 24 hours (5)'. Below this are buttons for 'All status' and a refresh icon. It says 'Showing 1 - 1 items'. Underneath is a table with a single row. The first column is a checkbox labeled 'PIPELINE NAME' with a green box around it. The second column contains a link 'LabPipeline' which is also highlighted with a green box, and a small hand cursor icon indicating it's clickable.

Activity Runs								
Pipeline Run ID 4ab10e3f-09ce-4765-add2-18e3a5ba6311								
All	Succeeded	In Progress	Failed	Cancelled				
Activity Name	Activity Type	Actions	Source	Destination	Run Start	Duration	Status	
LabNotebook	DatabricksNotebook				10/30/2018, 10:08:26 AM	00:03:03		In Progress
Copy_deu	Copy		data/03.02/	dwtemp/03.02/	10/30/2018, 10:03:12 AM	00:05:14		Succeeded

- d. To view the notebook output of the run, you can select the Output Action next to the LabNotebook activity, and then select the runPageUrl value. This will open the executed notebook within Databricks, and you can view details of the run in each cell's output.

The screenshot shows the 'Activity Runs' section of the Azure Data Factory interface. A specific run for a 'LabNotebook' activity is selected, and its details are displayed. The 'Output' tab is active, showing the run's JSON log. A red box highlights the 'runPageUrl' field, which contains a URL: "https://westus.azuredatabricks.net/?o=8161753722983864#job/1/run/1". Below the log, the run's status is shown as 'In Progress'. The run was started on 09/22/2018 at 6:48:07.

Activity Name	Activity Type	Run Start	Duration	Status
LabNotebook	DatabricksNotebook	09/22/2018, 6:48:07	00:03:03	In Progress

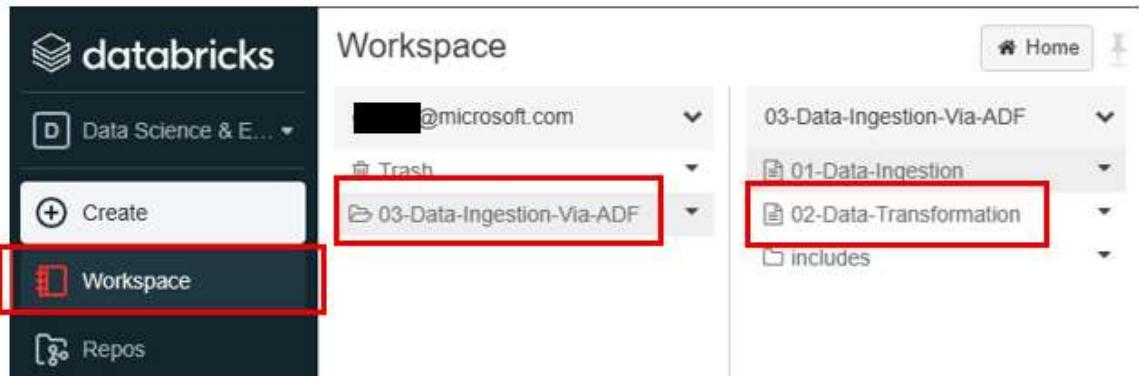
Exercise 6 has been completed.

## Exercise 7: Review the Transformed Data using Azure Databricks

In the previous exercise you prepared the data and stored in table in Azure Databricks. Now you will perform some basic aggregation on the sample dataset to generate required reports using the **02-Data-Transformation** notebook.

### Tasks

1. Navigate back to Databricks.
2. Select the **02-Data-Transformation** Databricks Notebook from the 03-Data-Ingestion-Via-ADF Workspace.



3. Continue with the instructions in the **02-Data-Transformation** notebook to **Verify transformations and aggregate the data** in the notebook.

Exercise 7 has been completed.



## Control Flow

---

### Introduction

During this lab, you will learn how to configure Control Flow within your Azure Data Factory pipelines.

### Estimated Time

90 minutes

### Objectives

At the end of this lab, you will be able to:

- Configure a Data Transformation pipeline.
- Use the Flow Control to add error handling and control the path of execution.

### Logon Information

Use the following credentials to login into virtual environment:

- Username: **Admin**
- Password: **PasswOrd!**

## Table of Contents

---

[Exercise 1: Prepare your environment](#)

[Exercise 2: Implement Control Flow](#)

[Exercise 3: Extend the Control Flow with Lookup and IF Condition activities](#)

**Lab: Control Flow with Azure Data Factory**

During this lab, you will learn how to do data transformations with Azure Data Factory.

## Exercise 1: Prepare your environment

This step will take you through preparing your environment for the exercises which will follow.

### Tasks

1. Connect to the Microsoft Azure Portal. Open Edge browser and navigate to <http://portal.azure.com/> to connect to Microsoft Azure Portal. Sign in with your subscriptions credentials.
2. Create database objects required for the lab.

- a. Go to SQL database created in **Module 1, Lesson 2, Lab 2**.

A screenshot of the Azure portal's resource list. The table has two columns: 'Name' and 'Type'. The 'mkgsqldbazsql' database is selected and highlighted with a yellow box. It is categorized as a 'SQL database'.

Name	Type
mkgadf	Data factory (V2)
mkgblob	Storage account
mkgdatabricks	Azure Databricks Service
<b>mkgsqldbazsql (mkgsqlserver/mkgsqldbazsql)</b>	<b>SQL database</b>
mkgsqlserver	SQL server

- b. Connect to **Query Editor (preview)**. Type in Login, **datafactadmin**, and Password, **d@taFactSQL1**. Click **OK** to login.

A screenshot of the 'Query editor (preview)' interface. The left sidebar shows navigation options like Overview, Activity log, Tags, etc., with 'Query editor (preview)' selected and highlighted with a yellow box. On the right, there is a 'Welcome to SQL D...' message and a 'SQL server authentication' dialog. The 'Login \*' field contains 'datafactadmin' and the 'Password \*' field contains 'd@taFactSQL1', both highlighted with yellow boxes. A large yellow box also surrounds the 'OK' button at the bottom right of the dialog.

- c. Once connected, click on **Open Query** and execute the scripts below by clicking on **Run**.

- \LabFiles\M03\_L02\_Lab1\SalesLT\_ProductStaging.sql
- \LabFiles\M03\_L02\_Lab1\SalesLT\_usp\_CheckProductStaging.sql
- \LabFiles\M03\_L02\_Lab1\SalesLT\_usp\_GetRowCountProductStaging.sql
- \LabFiles\M03\_L02\_Lab1\SalesLT\_usp\_PrepareProductStaging.sql
- \LabFiles\M03\_L02\_Lab1\SalesLT\_usp\_TruncateProductStaging.sql

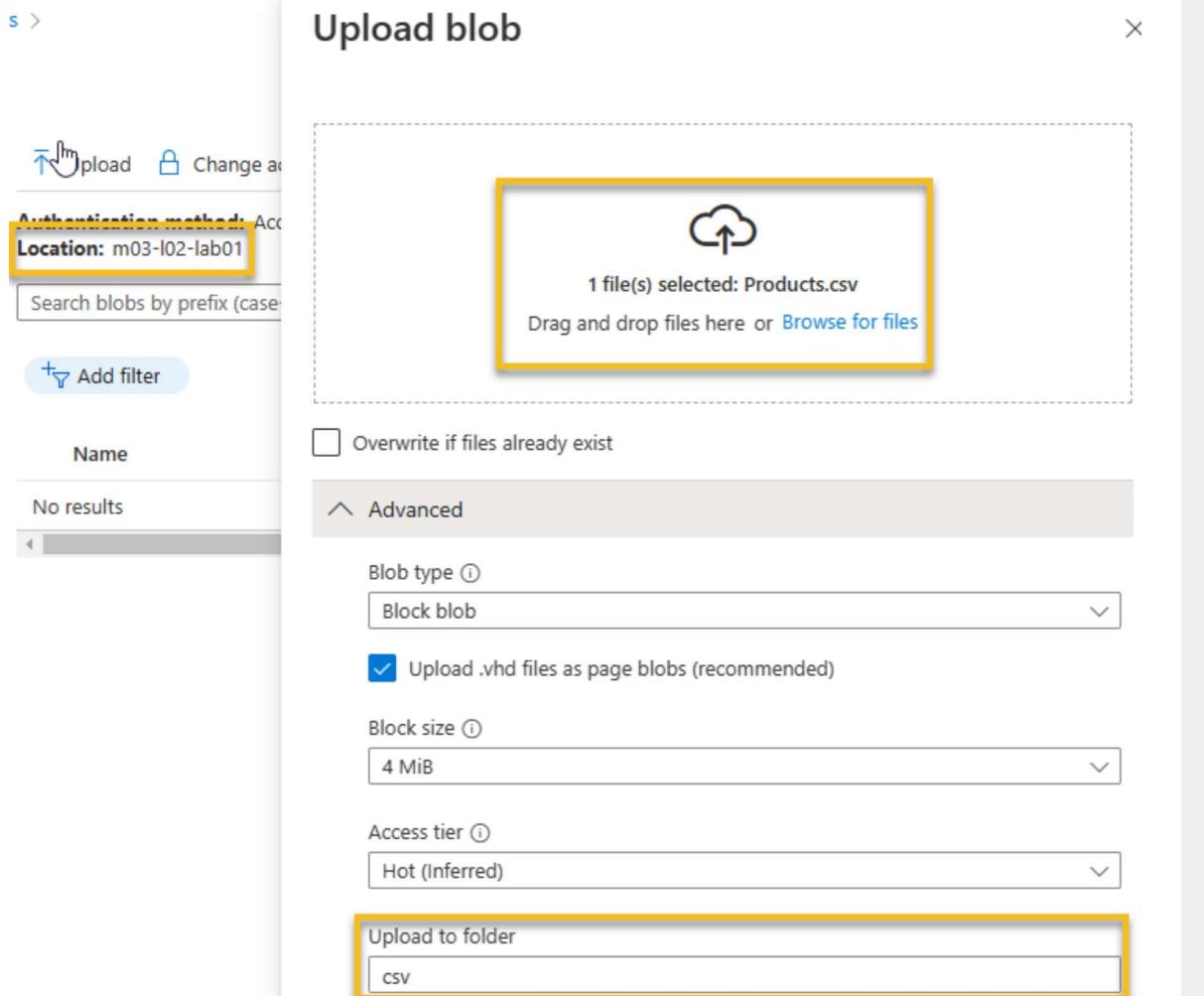
Exercise 1 has been completed.

## Exercise 2: Implement Control Flow

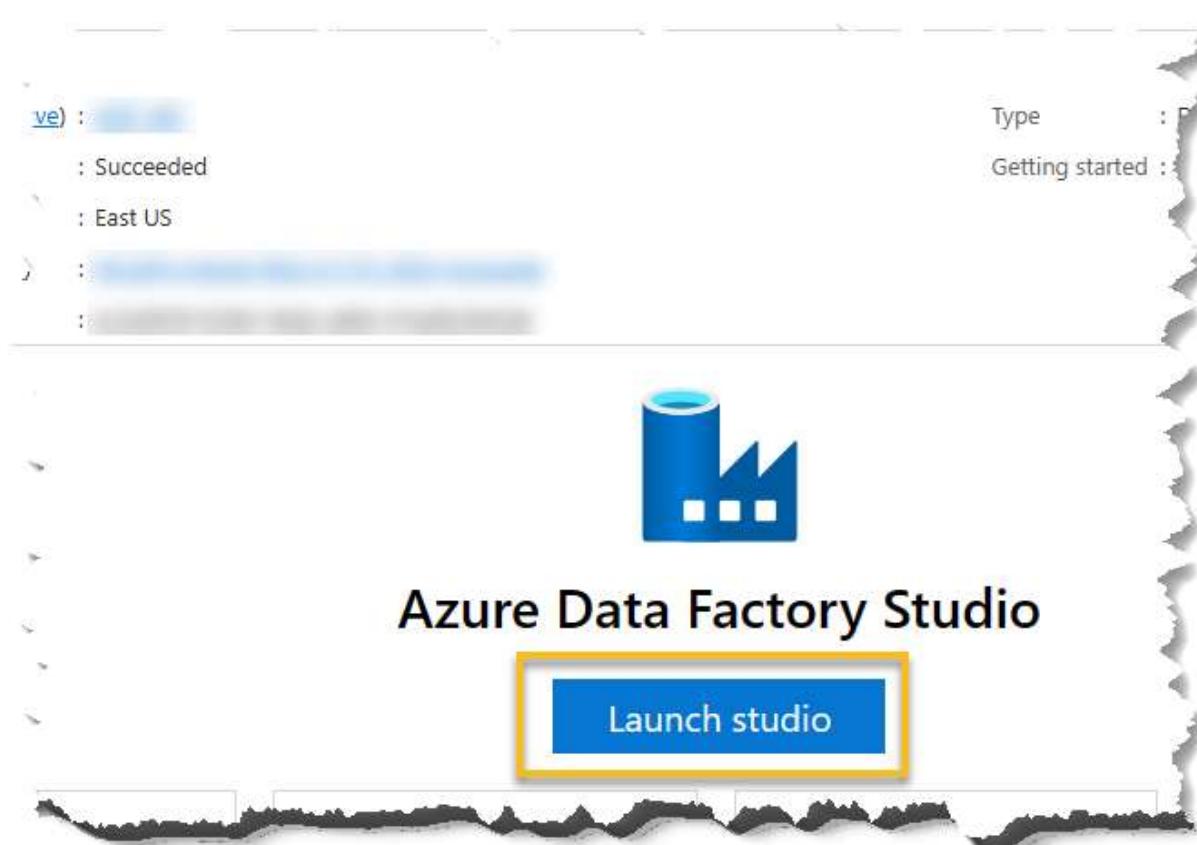
This exercise shows the benefits of using Control Flow within your pipeline.

### Tasks

1. Prepare the Azure Blob Storage
  - a. Using the Azure Portal connect to the storage account created in **Module 1, Lesson 2, Lab 2.**
  - b. Create a blob container called **m03-l02-lab01** in the root of the storage account.
  - c. Upload the input file.
- d. A sample file to use as the input has been provided for you in the lab files **\M03\_L02\_Lab01\products.csv**.
- e. Using the Azure Portal upload this file to the **csv** folder in the container **m03-l02-lab01**.



2. Log onto your Azure Data Factory by launching the **Azure Data Factory Studio**.



3. Verify that you have configured the Linked Service to the Azure Blob Storage account. If so, go to step 4; otherwise, continue with the instructions below:

- If you do not have a Linked Service already in your Data Factory to your Blob Storage account, follow the steps below to create one:
- Select the **Toolbox/Manage** button from the left panel. Under **Linked Service** and then select **+ New**.

This screenshot shows the 'Connections' blade in the Microsoft Azure Data Factory interface. On the left sidebar, under 'Connections', the 'Linked services' item is highlighted with a red box. On the right, the 'Linked services' section is displayed with a sub-section for 'Azure Blob Storage'. A red box highlights the '+ New' button. Other items in the list include 'Integration runtimes' and 'Azure Purview (Preview)'. There are also filters for 'Annotations' and 'Name'.

- Select the **Azure Blob Storage** type and click the **Continue** button.

## New Linked Service

X

Data Store Compute

blob

All Azure Database File Generic Protocol NoSQL Services and apps



Azure Blob Storage

Cancel

Continue

- d. Change the **Linked Service name** if desired (for example: use the storage account name).
- e. Select your **Subscription** then your **Azure Storage Account** from the dropdown and click the **Create** button.

## New linked service (Azure Blob Storage)

Name \*

Description

Connect via integration runtime \* ⓘ



Authentication method

 Connection string Azure Key Vault

Account selection method ⓘ

 From Azure subscription  Enter manually

Azure subscription ⓘ



Storage account name \*



Additional connection properties

Test connection ⓘ

 To linked service  To file path

Annotations

⋮ Advanced ⓘ

4. Configure the Azure SQL Database Linked Service in ADF.

- a. From the **Toolbox/Manage** button from the left panel. Under **Linked services**, select **+New**.

The screenshot shows the Microsoft Azure Data Factory interface. The top navigation bar displays 'Microsoft Azure | Data Factory > jhmay2021adf'. Below the navigation, there are several buttons: 'Validate all', 'Save all', and 'Publish'. On the left, a sidebar menu lists 'Connections' (with 'Linked services' highlighted and a red box around it), 'Integration runtimes', 'Azure Purview (Preview)', 'Source control' (with a red box around its icon), and 'Git configuration'. The main content area is titled 'Linked services' and contains the message 'Linked service defines the connection information to a data store or compute. Learn more'. A 'New' button (also highlighted with a red box) is prominently displayed. Below it are filters for 'Filter by name' and 'Annotations : Any'. The text 'Showing 1 - 4 of 4 items' is followed by two sorting options: 'Name ↑↓' and 'Type ↑↓'.

- b. On the Data Store tab, select **Azure SQL Database** and click the **Continue** button.

## New linked service

Data store   Compute



sql

All

Azure

Database

File

Generic protocol

NoSQL

Services and apps



Azure Cosmos DB (SQL API)



Azure Database for MySQL



Azure Database for PostgreSQL



Azure SQL Database



Azure SQL Database Managed Instance

Continue

Cancel

- c. Enter a name for the linked service. Something descriptive (for example: **AzureSQL\_**)
- d. Select **AutoResolveIntegrationRuntime**.

- e. Select the **Subscription**, **Azure SQL Database server** and **database**.
- f. Select **SQL authentication** for **Authentication Type**.
- g. Enter the user name, **datafactadmin**, and password, **d@taFactSQL1**.
- h. Select **Test Connection** and make sure the connection succeeds.
- i. Then select **Create**.

## New linked service

Azure SQL Database [Learn more](#)

Name \*

AzureSQL\_MyDB

Description

Connect via integration runtime \* ⓘ

AutoResolveIntegrationRuntime

[Connection string](#)

[Azure Key Vault](#)

Account selection method ⓘ

From Azure subscription  Enter manually

Azure subscription

Server name \*

Database name \*

Authentication type \*

User name \*

[Password](#)

[Azure Key Vault](#)

Password \*

Always encrypted ⓘ

[Create](#)

[Back](#)

 [Test connection](#)

[Cancel](#)

### 5. Create a Dataset - Azure Blob Storage.

a. We will need a dataset that points to the file uploaded earlier in this exercise.

b. Click on the **Author/Pencil** button in the left pane.

# Microsoft Azure

>>



c. Create a new dataset by clicking the + button and selecting **Dataset**.

## Factory Resources

▽ <<



Filter resources by name



d. Select **Azure Blob Storage** and then **Continue**.

e. Select **DelimitedText** and then **Continue**.

f. Provide a name for the dataset, such as **ProductsCSV**, and select the **Linked Service** created for Blob Storage in an earlier exercise.

g. **Browse** to the csv file uploaded (m03-l02-lab01/csv/Products.csv) and check the box for **First row as header**.

h. Click the **OK** button.

### Set Properties

X

Name

ProductsCSV

Linked service \*

csvinput

[Edit Connection](#)

File path

m03-l02-lab01

/ csv

/ Products.csv

[Browse](#)

▼

First row as header



Import schema

From connection/store

From sample file

None

6. Create a Dataset - Azure SQL Database

a. We will need a dataset that points to the Azure SQL Database so that we can use it within our activities.

b. Create a new dataset by clicking the + button and selecting **Dataset**.

c. Select the **Azure SQL Database** and then **Continue**.

d. On the Set Properties tab, change the dataset Name to **ProductStaging**.

- e. Select the **Linked Service** you created earlier in this exercise, and the table **[SalesLT].[ProductStaging]**.

## Set properties

Name  
ProductStaging

Linked service \*  
AzureSQLADF 

Table name  
SalesLT.ProductStaging  

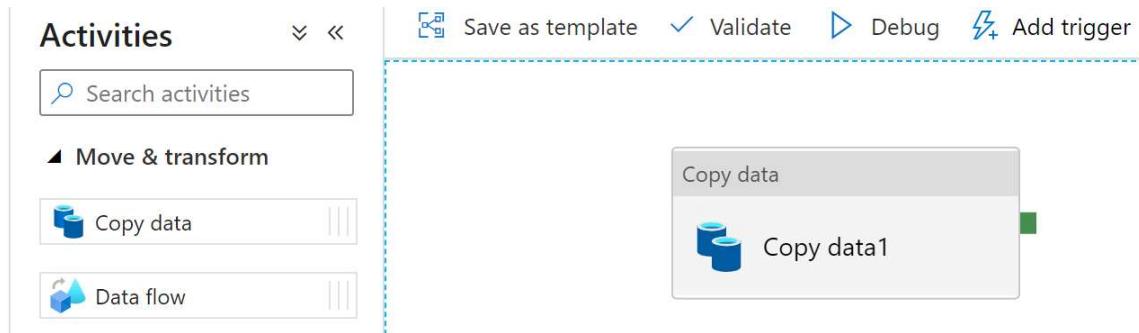
Edit

Import schema  
 From connection/store  None

f. Select **OK** to save the dataset.

## 7. Create a Pipeline and data gestation

- a. Create a new pipeline by clicking the + button and selecting **Pipeline**.
- b. Under **Activities**, expand the **Move & transform** and drag the **Copy Data** activity to the designer surface.



- c. In the Copy Data General tab, set the Name to **Import CSV**.

- d. Switch to the **Source** tab. Select the **ProductsCSV** dataset.



- e. Switch to the **Sink** tab. Select the **ProductsStaging** Linked Service for the destination we will copy to. Set the Storage Procedure Name to **None**.

Sink dataset \* ProductStaging Edit

Stored Procedure Name None Edit Refresh

f. Switch to the **Mapping** tab. Choose **Import Schemas**. Remove the **row for the StagingID column** by clicking the check box to the left and click **Delete**.

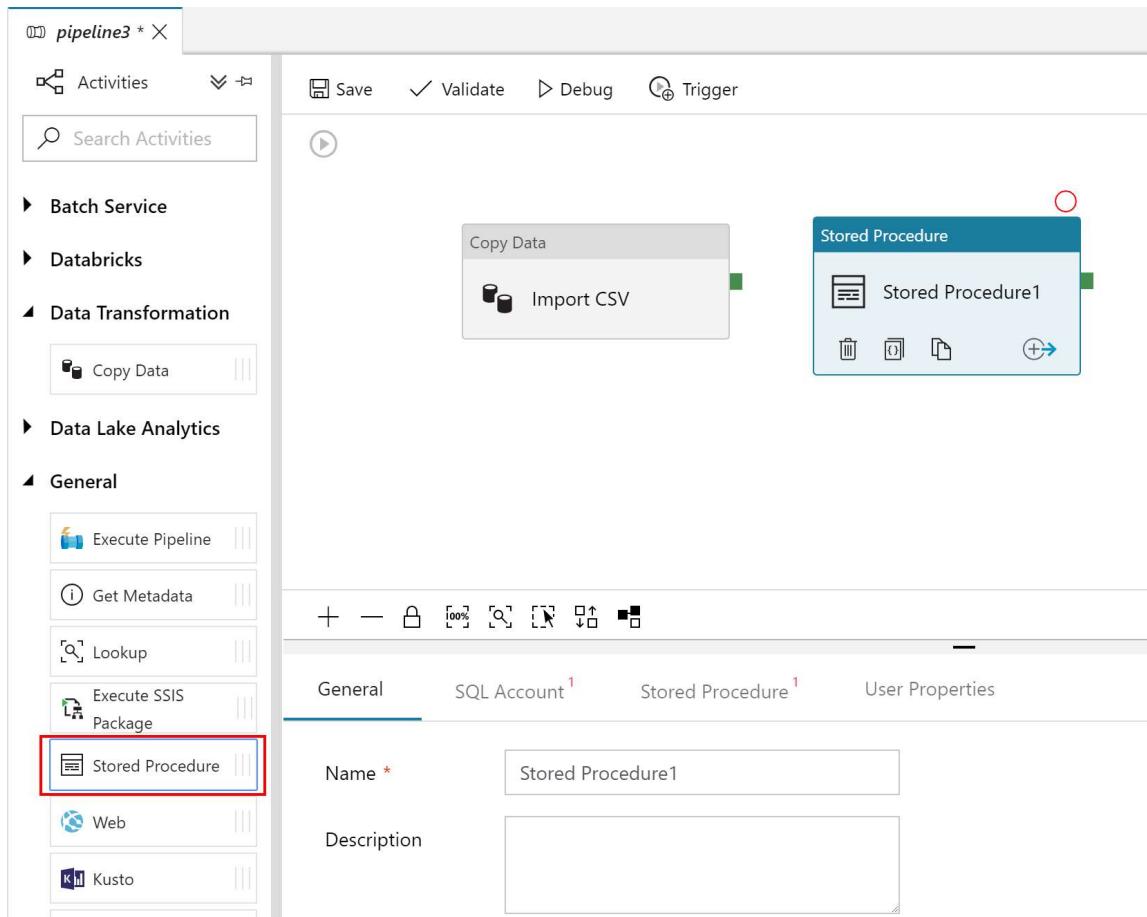
Source	Type	Destination	Type
<input checked="" type="checkbox"/> Select or edit column	Select type	StagingID	int
Mapping source is empty.			
ProductID	abc String	ProductID	int
Name	abc String	Name	nvarchar
ProductNumber	abc String	ProductNumber	nvarchar

g. If you have Git configured, click the **Save all** button to save your changes.

h. Click the **Publish** button to save the changes to the Pipeline before proceeding.

8. Add Data validation activity.

a. From the list of **General Activities**, add a **Stored Procedure** activity to the designer surface of the Pipeline you are working on.



b. Change the name of the activity to **Validate Import Rows**.

c. Switch to the **Settings** tab and select your SQL Linked Service [**SalesLT**].  
[**usp\_CheckProductStaging**] from the drop-down list.

The screenshot shows the 'Settings' tab for the 'Stored Procedure' activity. The 'General' tab is selected. The 'Linked service' dropdown is set to 'AzureSQLADF'. The 'Stored procedure name' dropdown is set to '[SalesLT].[usp\_CheckProductStaging]'. Below these fields are sections for 'Stored procedure parameters' (with 'Import' and 'New' buttons) and 'User properties'.

d. Select **Save all** (if Git configured) then **Publish** on the Pipeline to deploy changes.

9. Add Logic Apps for sending email notifications.

a. In the Azure Portal, enter **Logic app** in the search bar, then click Add or **Create**



## Logic App

Microsoft

Create

b. Select the **Resource Group** you have been using for the workshop.

c. Change type to **Consumption**.

Make sure to select consumption plan, as standard plan does not have the option to send email.

d. Give the Logical app a suitable **name** (i.e. -la-pipeline-status).

e. Choose **East US** Region.

f. Check no for Enable log analytics.

Basics Tags Review + create

Create a logic app, which lets you group workflows as a logical unit for easier management, deployment and sharing of resources. Workflows let you connect your business-critical apps and services with Azure Logic Apps, automating your workflows without writing a single line of code.

#### Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ [Redacted] ▾

Resource Group \* ⓘ adfDatabricksworkshopUpgrade ▾  
Create new

#### Instance Details

Type \*

Consumption  Standard

Looking for the classic consumption create experience? [Click here](#)

Logic App name \*

logicadfupgrade ▾

Region \*

East US ▾

Enable log analytics \*

Yes  No

[Review + create](#)

< Previous

Next : Tags >

g. Click **Review and Create**, then **Create**.

h. After it has deployed, click on **Go to resource**.

i. In the Logic App Designer, select the Http Trigger (**When a HTTP request is received**).

j. Add the text from the **\LabFiles\M03-L02-Lab01\HTTPRequestBodyJSON.txt** file to the **Request Body JSON Schema** section.

The screenshot shows the Azure Logic App builder interface. At the top, it says "When a HTTP request is received". Below that is the "HTTP POST URL": <https://prod-17.eastus.logic.azure.com:443/workflows/20d3993e793945bf9bdbf1d01d5d6433/tri...>. Under "Request Body JSON Schema", there is a JSON schema editor with the following code:

```
{  
    "properties": {  
        "DataFactoryName": {  
            "type": "string"  
        },  
        "PipelineName": {  
            "type": "string"  
        },  
        "ErrorMessage": {  
            "type": "string"  
        }  
    }  
}
```

Below the schema editor are two buttons: "Use sample payload to generate schema" and "Show advanced options ▾".

k. Select the **+New Step** use the search bar to find and select **Outlook.com - Send an email**.

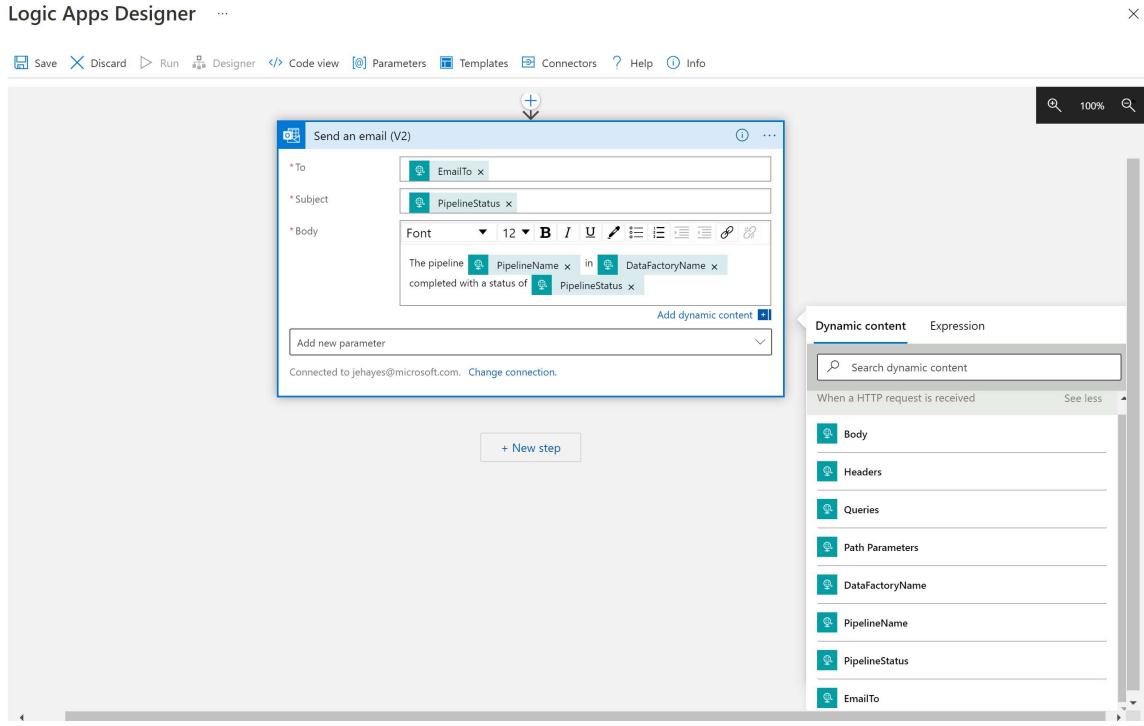
**i** This is the provider for your email account. You can use other mail provider activities such as Office 365.com, Gmail, etc. but these may have slight variations in sign-in/authentication processes and configuration)

Your browser cannot be in Incognito and/or InPrivate mode when signing in.

l. If you are using Outlook.com use your personal email address. If you are using Office 365, login using corporate email address.

The screenshot shows the "Outlook.com" step configuration. It has a title "Outlook.com" and a status message "Creating...". Below the title, it says "Sign in to create a connection to Outlook.com."

m. Enter the details of the email. You can either hard code the fields, or to make this more scalable use a combination of text and the dynamic fields selector as shown below. When you click into a field, a fly out box will appear with the available Dynamic content and Expression builder.



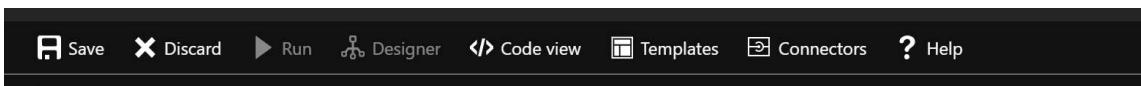
- If you do not see the json parameters when your cursor is in any of the Email required fields, click on the **See more** hyperlink in the Dynamic content section.

**Dynamic content      Expression**

**Search dynamic content**

**When a HTTP request is received** See more

n. **Save** when ready.



- o. Expand the **HTTP request** trigger and copy the **HTTP Post URL** for later reference (into Notepad or similar).

**When a HTTP request is received**

**HTTP POST URL**  
https://prod-17.eastus.logic.azure.com:443/workflows/20d3993e793945bf9bdbf1d01d5d6433/tri...

**Request Body JSON Schema**

The screenshot shows the Microsoft Flow designer interface. A step titled "Send an email" is selected. The configuration for the step includes:

- To:** EmailTo
- Subject:** DataFactoryName x pipeline PipelineName x successful
- Body:** Pipeline PipelineName x completed successfully

Below the step, there's a note: "Connected to mlavery@microsoft.com. Change connection." At the bottom left is a "+ New step" button. To the right is a sidebar titled "Dynamic content" with tabs for "Expression" and "Search dynamic content". It lists several variables:

- DataFactoryName
- EmailTo
- ErrorMessage
- PipelineName

## 10. Configure the Pipeline Parameters

- In your Data Factory, click on the whitespace of the designer surface.
- Edit the parameters of the pipeline and add **EmailTo** of type **String** and the value of your email address.

Name	Type	Default value
EmailTo	String	email@domain.com

- Add Web Activity to send failure notification
- Add a **Web** Activity to the pipeline.

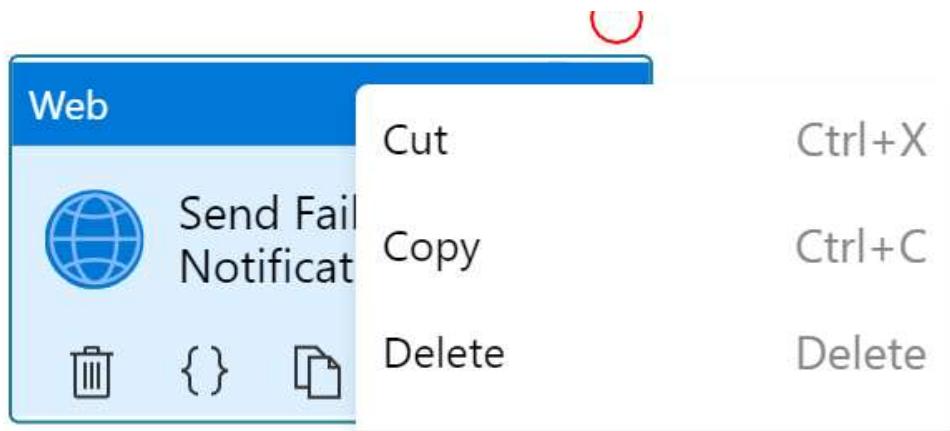
The screenshot shows the Azure Data Factory pipeline designer for pipeline "pipeline3".

- Left Sidebar (Activities):**
  - Batch Service
  - Databricks
  - Data Transformation
    - Copy Data
  - Data Lake Analytics
  - General
    - Execute Pipeline
    - Get Metadata
    - Lookup
    - Execute SSIS Package
    - Stored Procedure
    - Web
    - Kusto
    - Wait
- Center Area (Pipeline Flow):**
  - Three activities are connected sequentially: "Copy Data" (Import CSV) → "Stored Procedure" (Validate Import Rows) → "Web" (Web1).
  - The "Web" activity is highlighted with a red circle.
- Properties Panel (Bottom):**
  - General tab selected.
  - Name:** Web1
  - Description:** (empty)

- e. Set the name of the Web Activity to **Send Failure Notification**.
- f. On the Settings tab, enter the **URL** from the HTTP trigger of the relevant Logic App you created (previously copied in step **12.I**)
- g. Set the **Method** to **POST**.
- h. Click **+New** next to **Headers**.
- i. In this Header section, enter **Content-Type** for name and **application/json** for value.
- j. In the Body field add the contents of the file **\M03-L02-Lab1\WebActivityBodyFailure.txt**.

11. Copy the Web Activity to send a success notification

- a. Right click on your Send Failure Notification Web activity and select **Copy**



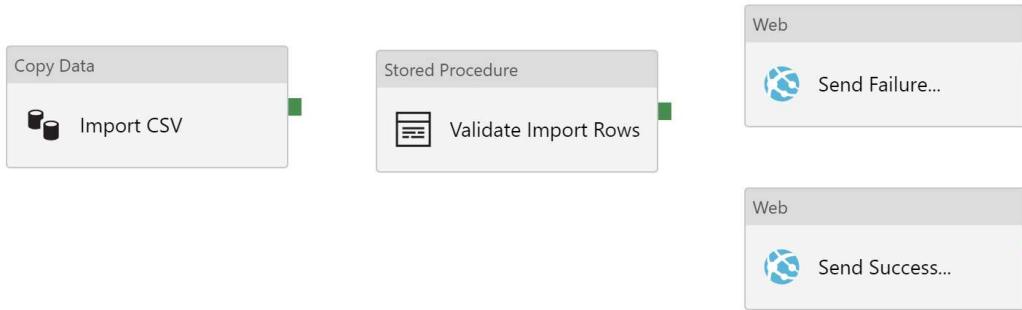
- b. Right click on the whitespace of the pipeline canvas and choose **Paste**



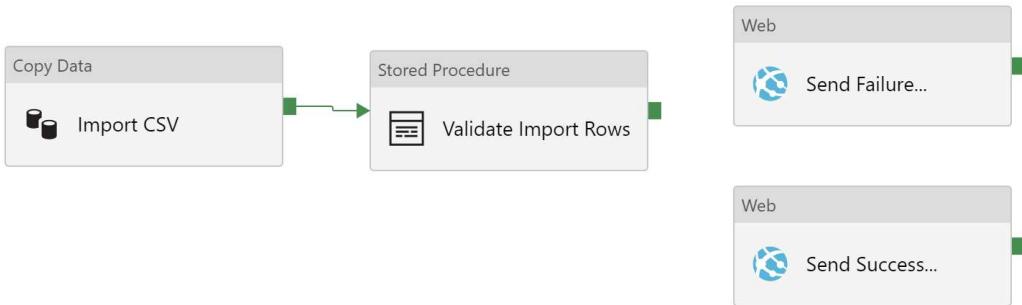
- c. Rename your new Web activity to **Send Success Notification**
- d. Go to the Settings tab and delete the code in the Body field. Replace the contents of the body field with add the contents of the file **\M03-L02-Lab1\WebActivityBodySuccess.txt**.

12. Configure the Flow Control

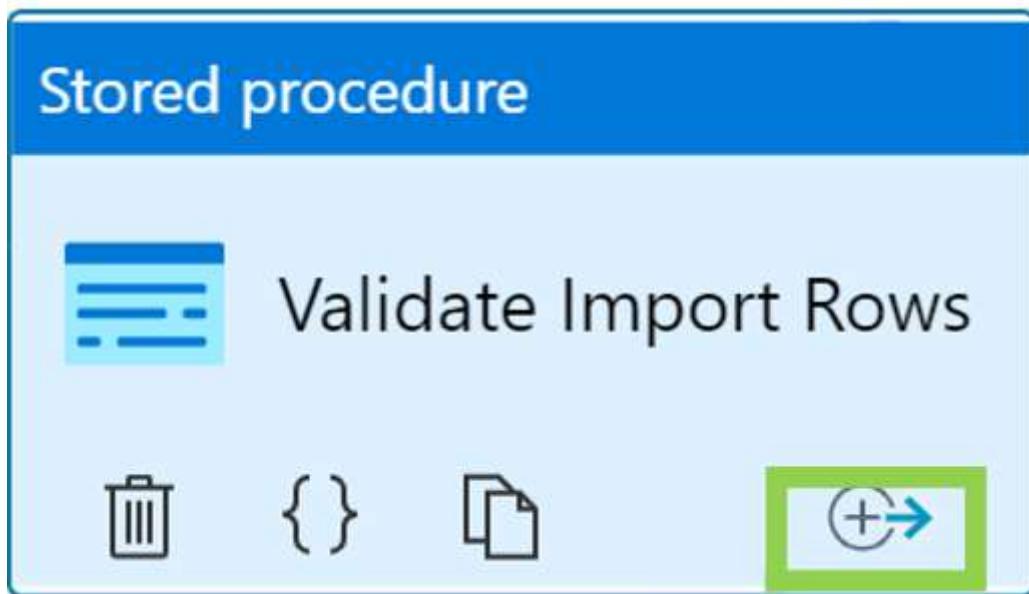
- a. The Pipeline should have 4 activities.



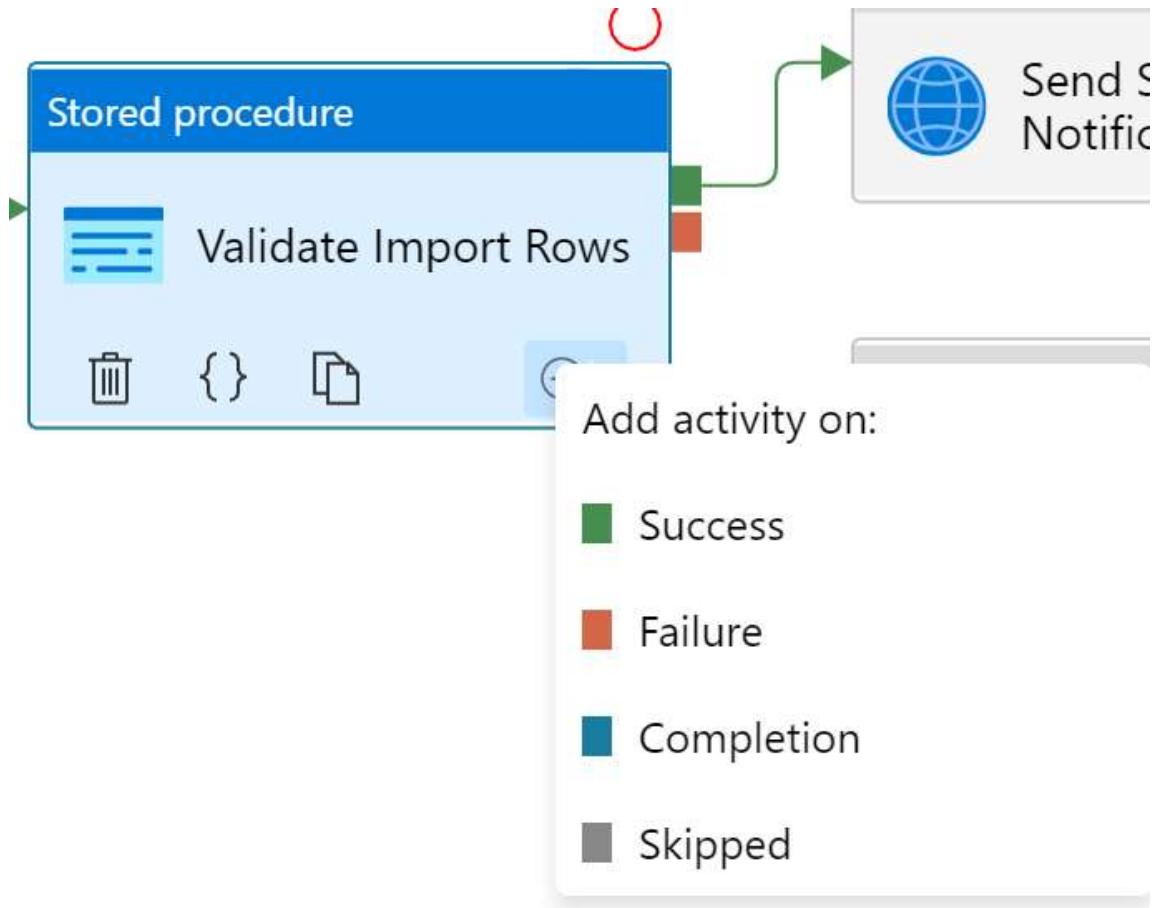
- b. Select the green box next to **Import CSV** and drag it to the **Validate Import Rows** activity.  
This will create a success path between the two activities.



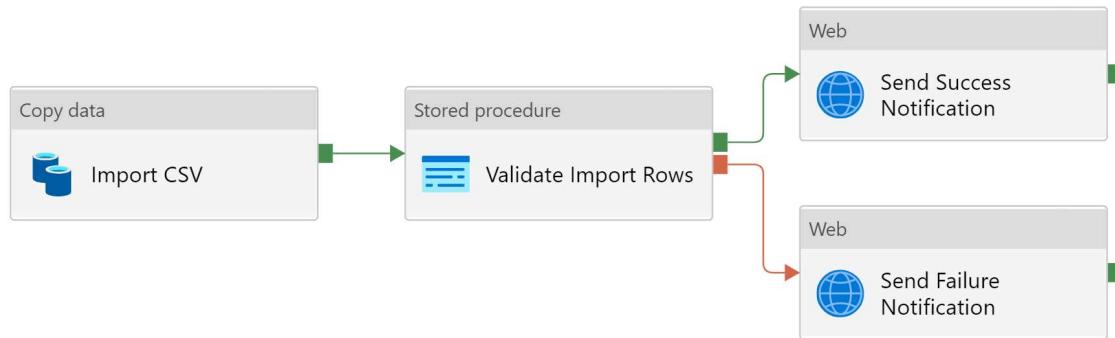
- c. Select the green box next to **Validate Import Rows** and drag it to the **Send Success** activity.  
d. Select the **Validate Import Rows** activity and notice the Add output icon in the lower right corner of the activity.



- e. Click on the Add output icon and choose **Failure**.



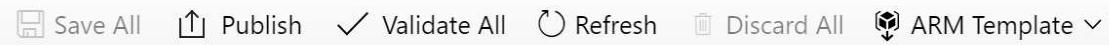
f. The pipeline will look like this.



g. If you have connected your Data Factory to a Git repository, click **Save**.

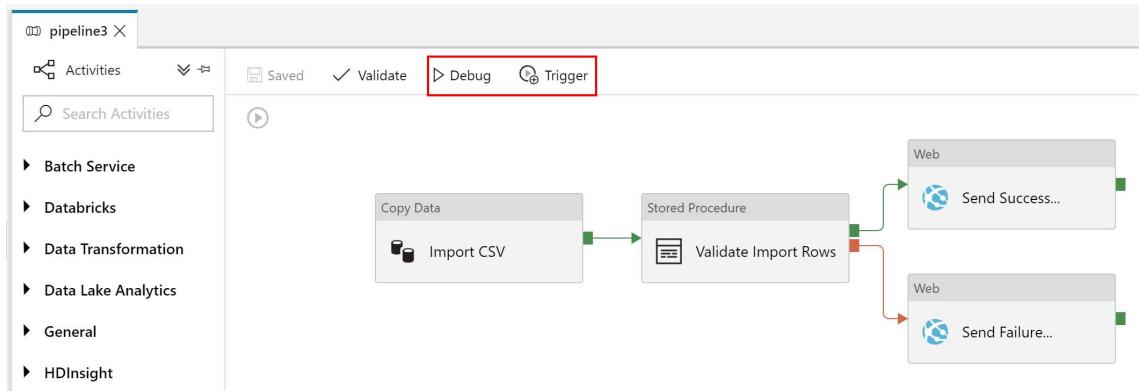
### 13. Execute the pipeline

- We are now ready to execute the pipeline. The stored procedure used in the validation phase is designed to error when there are no rows imported, as you might do in a real-world case. But it is also configured to error when there are more than 300 rows in the table, which in our case indicates that the table was not truncated before the import (we haven't implemented that step).
- First, use the **Publish** button in the toolbar to ensure the changes are pushed to the ADF service.



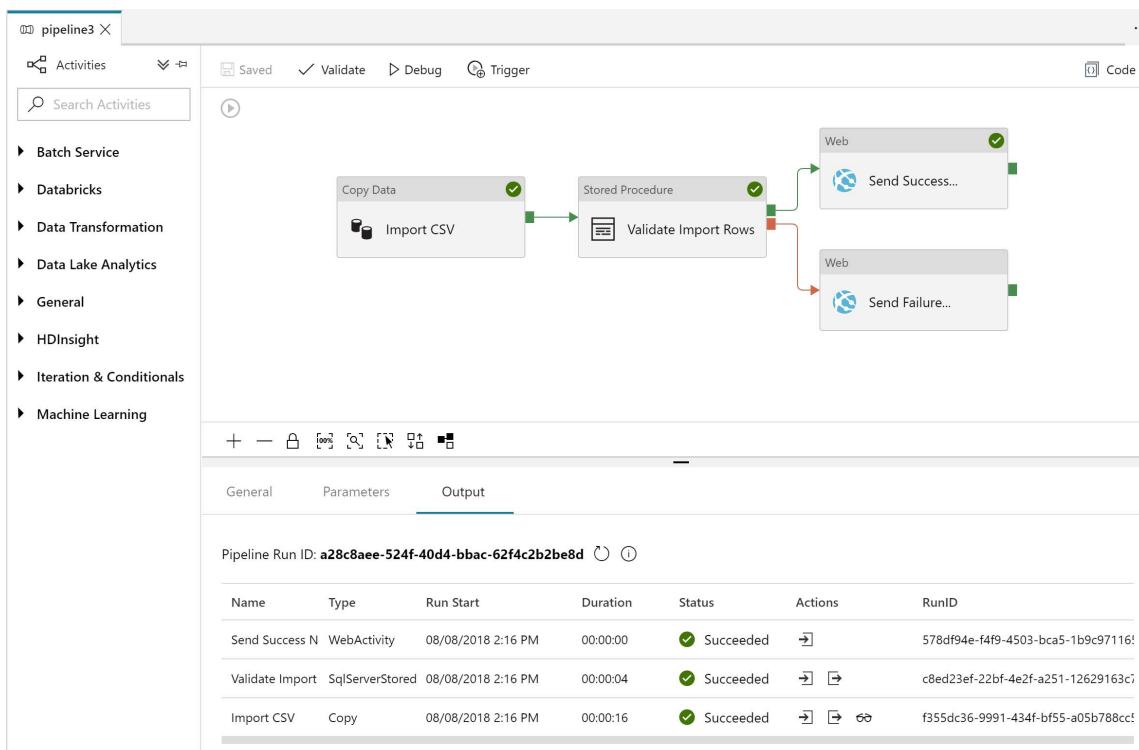
c. Make sure the publish process completes before proceeding.

d. Click either the Debug or Trigger method to execute the pipeline.

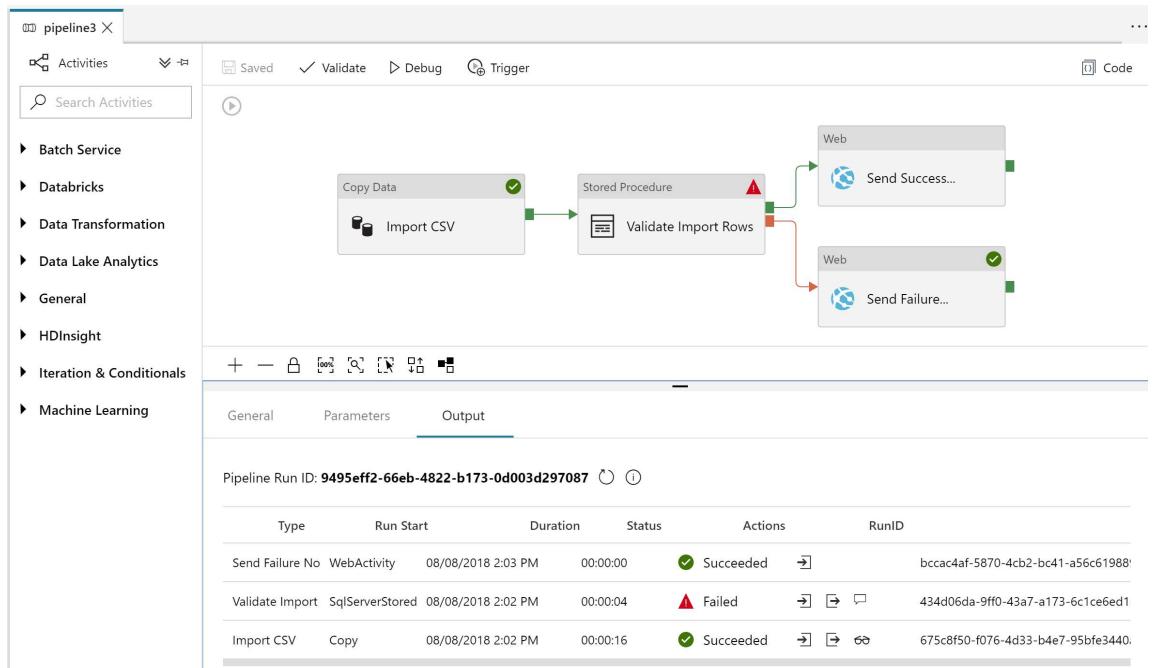


e. If using the Debug method, you should see indications in the designer as it progresses through the execution. If you used the Trigger method, then switch to the Monitoring tab and view the execution process and outcome.

f. When complete you will be able to see (in the run tasks) which pathway was taken. If using Debug method, you can also see the pathway through the icons on the designer. You should also receive an email.



g. Execute the pipeline again. This time it should follow the Failure path due to the conditions in our validation stored procedure. You should also receive an email message reporting the failure.



Exercise 2 has been completed.

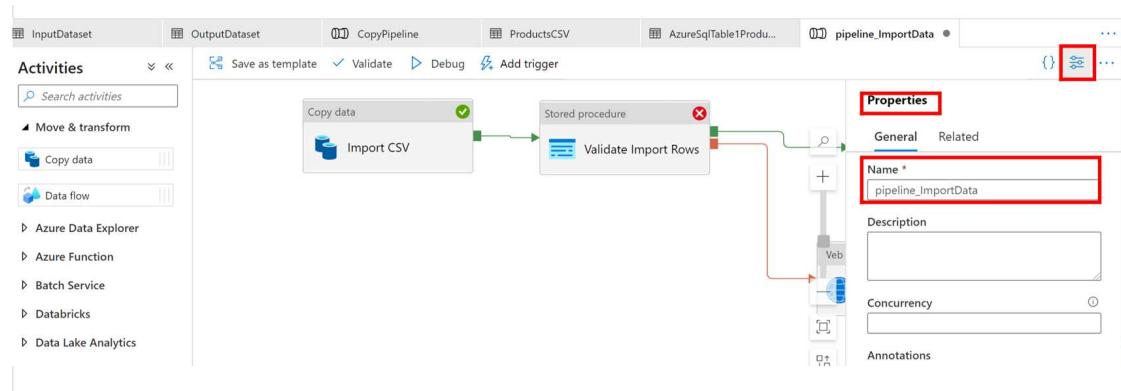
## Exercise 3: Extend the Control Flow with Lookup and IF Condition activities

This exercise shows how to extend the existing pipeline with additional Flow Control using both activity output and specifically designed activities. In this scenario we need to add additional logic to check if the staging table contains data and then perform a truncation before continuing.

### Tasks

#### 1. Rename the existing pipeline

- We will use the existing pipeline as a child to avoid duplicating activities. Edit the existing pipeline you created in the previous exercise. On the Properties panel to the right, rename the pipeline to **pipeline\_ImportData**.



- If you have Git enabled, select **Save** in the pipeline toolbar. Otherwise, click **Publish**.

#### 2. Create a new pipeline

- a. Create a new pipeline by clicking the + button under Factory Resources and selecting Pipeline.
- b. Rename the pipeline to **pipeline\_ImportData\_Control**.

**Properties**

---

**General**

---

**Name \***

pipeline\_ImportData\_Control

**Description**

**Concurrency** ⓘ

**Annotations**

+ New

---

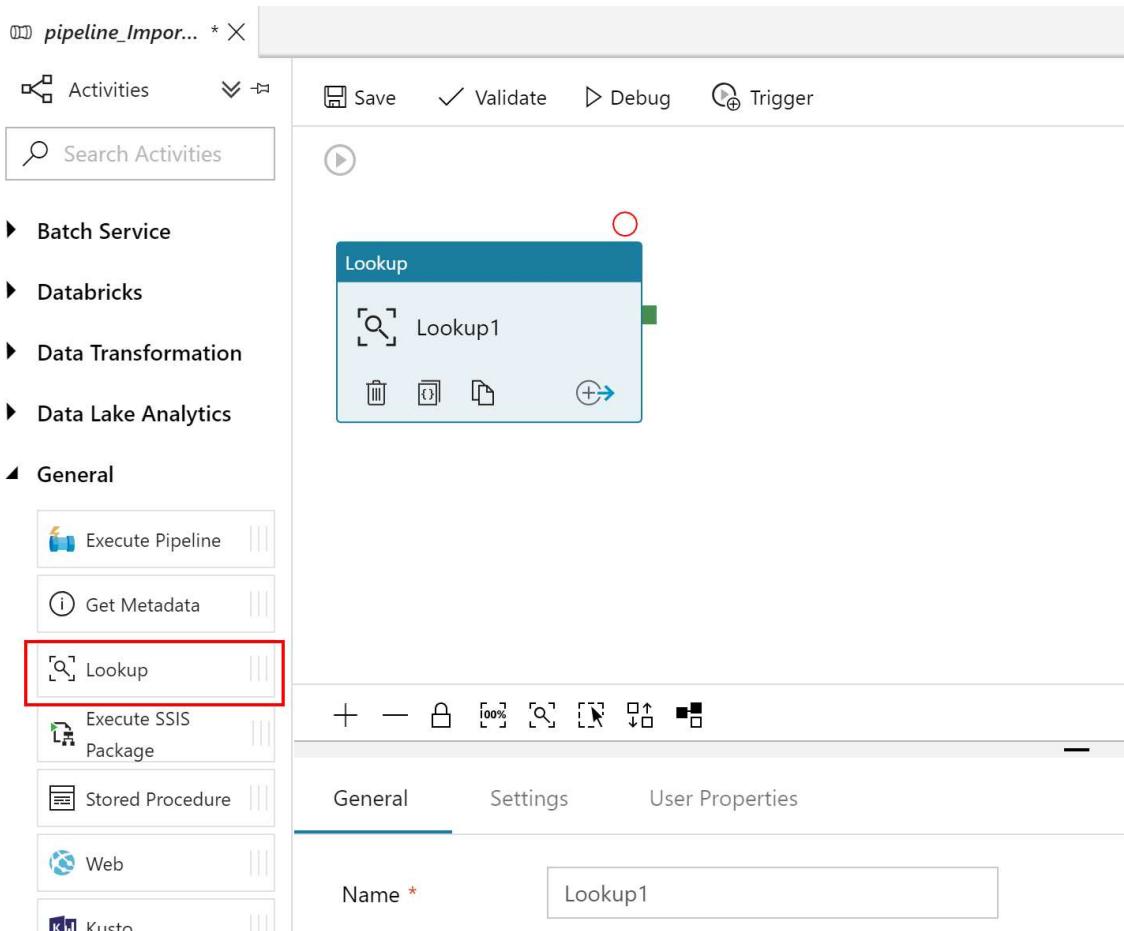
**Name**

---

c. If you have Git enabled, select **Save** in the pipeline toolbar.

3. Add a Lookup activity to the pipeline

- a. Locate the **Lookup** activity in the General section and drag it onto the designer surface.



b. Change the name of the activity to **Lookup Staging Row Count**.

General	Settings	User Properties
Name *	Lookup Staging Row Count	
Description		

c. Switch to the **Settings** tab. Select the **Source Dataset** as ProductsStaging (the Azure SQL Database dataset configured earlier).

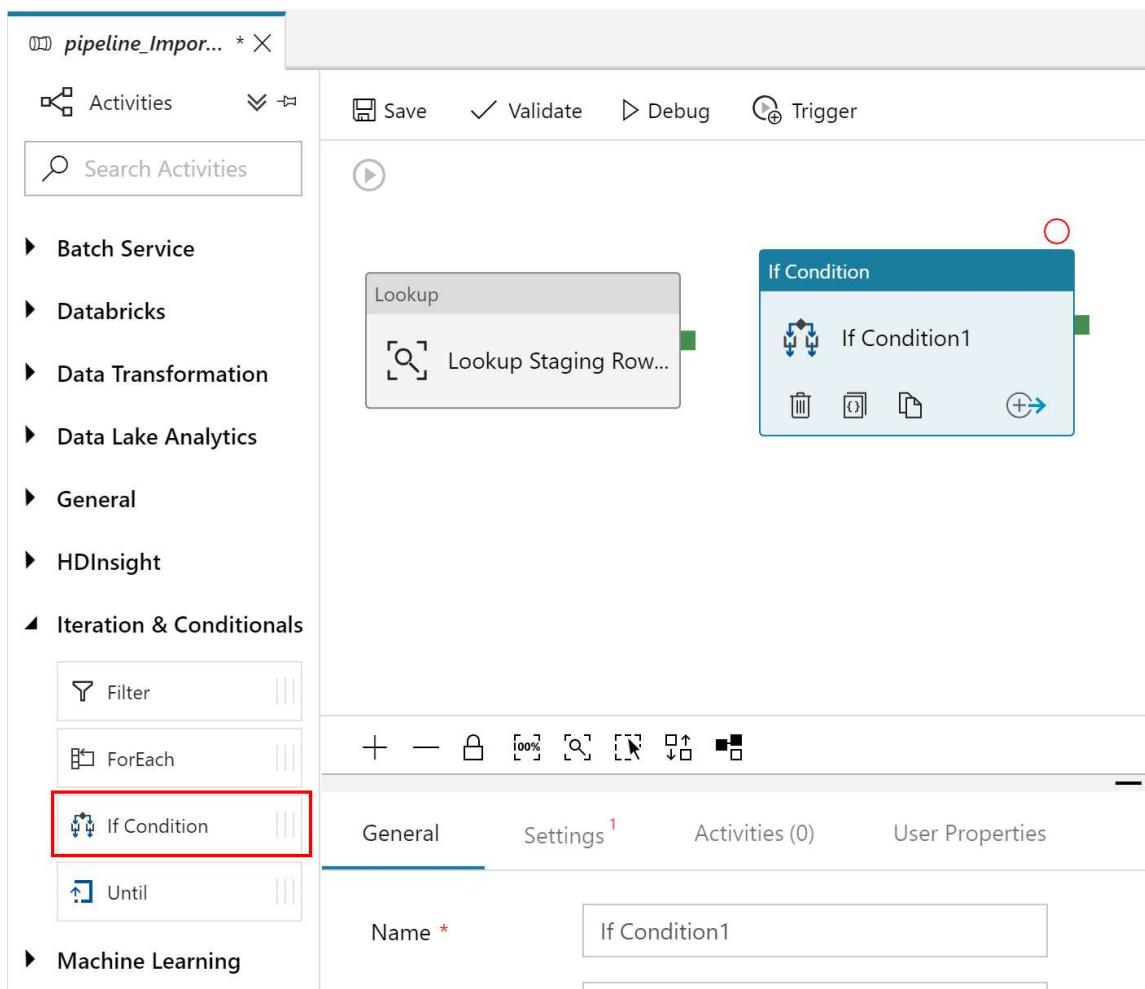
d. Set the Use Query to **Stored Procedure** and select the **[SalesLT].  
[usp\_GetRowCountProductStaging]** stored procedure from the drop-down.

e. Take note there is a **First Row Only** check box at the bottom of settings. In this case leaving that checked is ok. Be sure to review the documentation regarding the changes to behavior with that setting and a dataset.

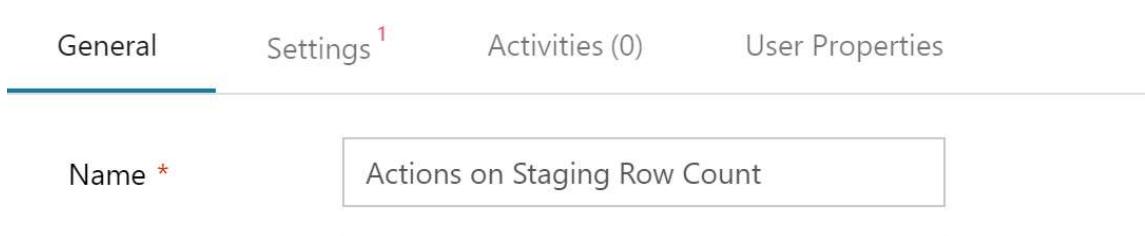
f. Select **Save** in the pipeline toolbar if you have Git enabled.

4. Add an IF Condition activity to the pipeline

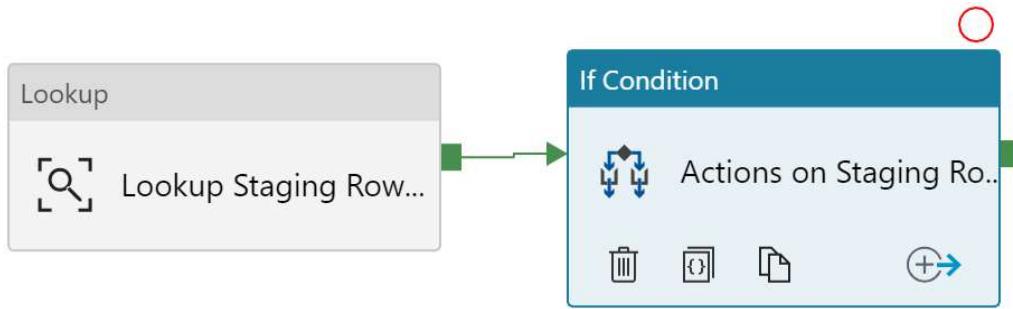
- a. Locate the **IF Connection** activity in the Iteration & Conditions section and drag it onto the designer surface.



- b. Change the name of the activity to **Actions on Staging Row Count**.



- c. Create an Activity Output Condition between the **Lookup Staging Row Count** and the **Actions on Staging Row Count** for **Success**.



d. Switch to the **Activities** tab of the **Actions on Staging Row Count** activity. Place the cursor in the Expression field, then select the **Add dynamic content** link or press **Alt+P**.

e. Enter the following into the Expression field. This will use the value from the lookup activity as part of an evaluation which returns true/false.

`T @equals(activity('Lookup Staging Row Count').output.firstRow.RowCount, 0)`

General      Settings      Activities (3)      User Properties

Expression

`@equals(activity('Lookup Staging Row Count').output.firstRow.RowCount, 0)`

(i)

f. For more on Logical Functions see <https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions>.

g. Click **Finish** to add the Dynamic content to the Expression field.

h. Click on the Pencil for to the Case = True.

**Case**

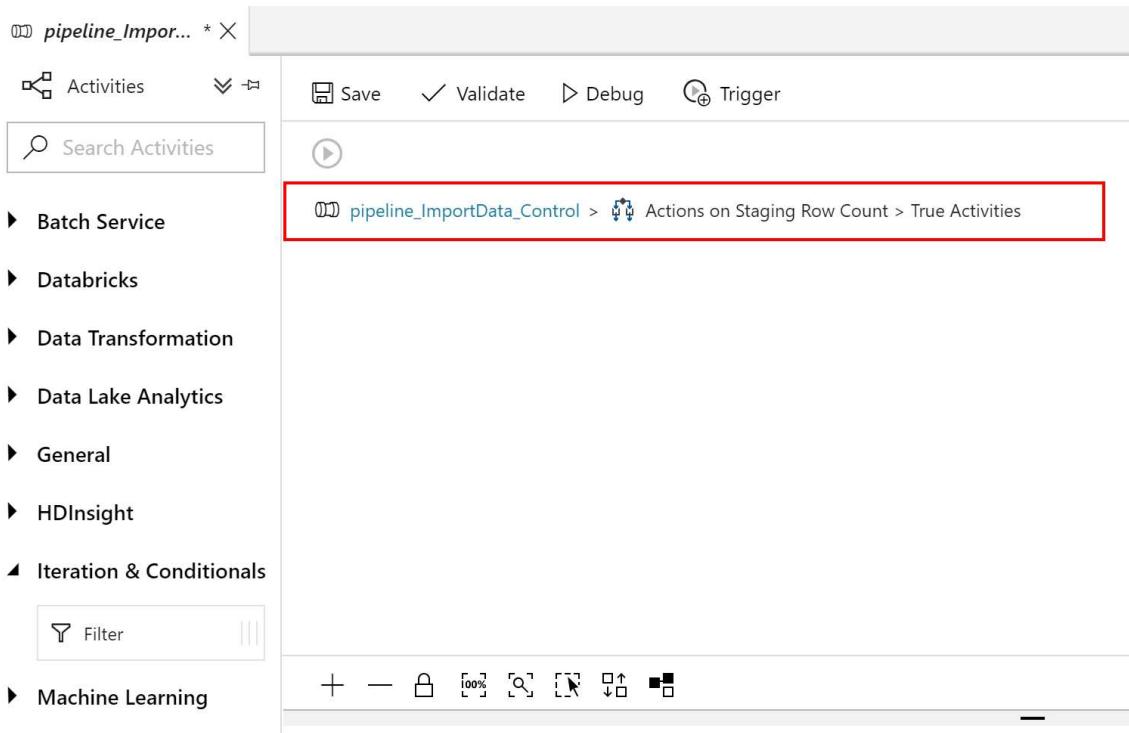
**Activity**

True

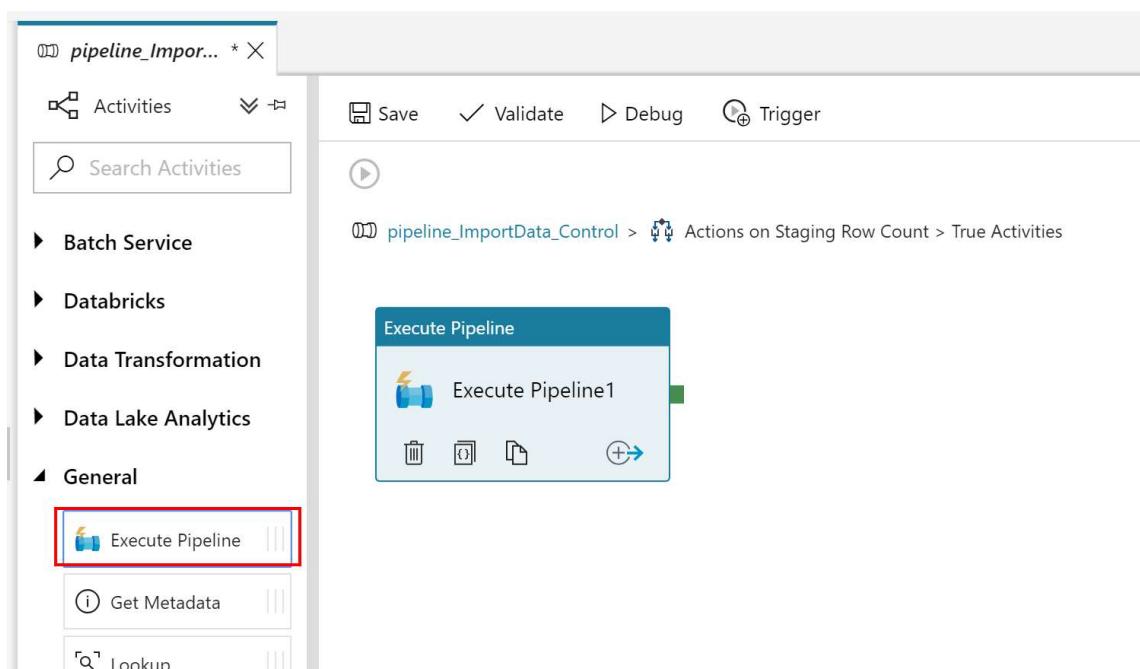
*No activities*



i. This will open a new child designer surface. Notice the context path within the pipeline above the designer surface.



- j. Here we will use the existing pipeline to avoid duplicating activities.  
Locate the **Execute Pipeline** activity in the **General** section. Drag it onto the designer surface.



- k. Rename the activity to **Import Data on True**.

- l. Switch to the **Settings** tab, select **pipeline\_ImportData** from the Invoked pipeline drop down.

General      Settings      User Properties

Invoked pipeline \*

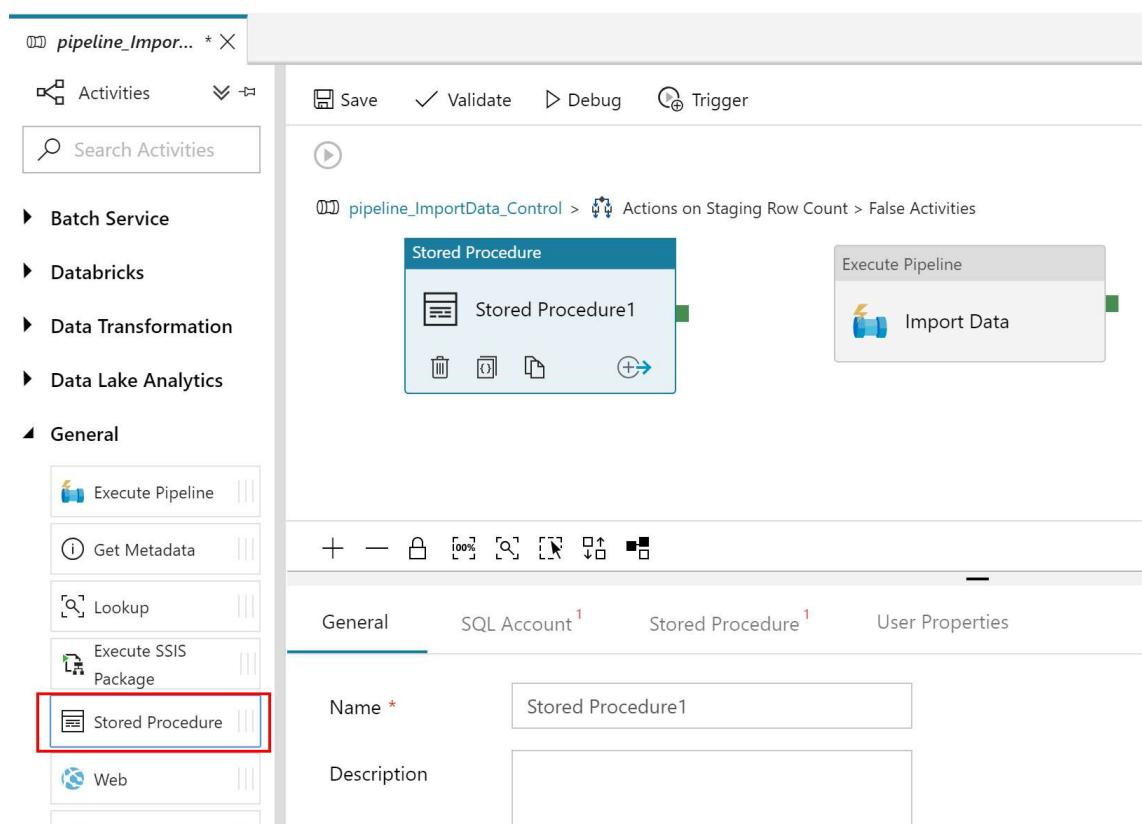
- m. Notice that the **Parameters** from the Import Data pipeline are carried over. The value could be defined by a parameter on the control pipeline, or even a looked-up value from a database table or config file. In a real-world solution you would most likely configure those settings. To keep it simple for this exercise just use the default value you have already configured (for example: hard coded configuration).

Name	Type	Value	Default value
EmailTo	string	<input type="text" value="Value"/>	je...@soft.com

- n. Click **Save** if you have Git configured.
- o. Navigate back to the main designer surface of the pipeline using the contextual path at the top of the designer surface. In other words, click on the **pipeline\_ImportData\_Control** hyperlink.

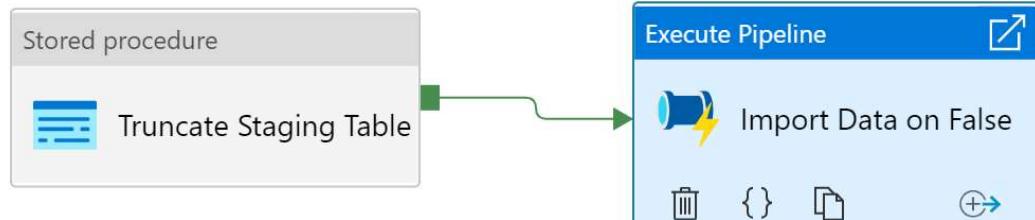


- p. Click on the Pencil for to the Case = False. Again, this opens a child designer surface.
- q. Here we will again call the **pipeline\_ImportData** pipeline. Add an Execute Pipeline activity to this designer surface as you previously did. Since activity names must be unique within a pipeline, use the name **Import Data on False**.
- r. Based on the business requirements we need to truncate the table before importing if it already has data. Locate the Stored Procedure activity in the **General** section and drag it onto the designer surface.



- s. Change the name of the activity to **Truncate Staging Table**.
- t. Switch to the **Settings** tab and select your Azure SQL Linked Service. Then select **[SalesLT]**. **[usp\_TruncateProductStaging]** from the drop-down list.
- u. Add an **Activity Output Condition** between this Stored Procedure Activity and the Execute Pipeline activity **on Success**.

pipeline\_ImportData\_Control > Actions on Staging Row Count > False activities



v. Navigate back to the main designer surface of the pipeline using the contextual path at the top of the designer surface.

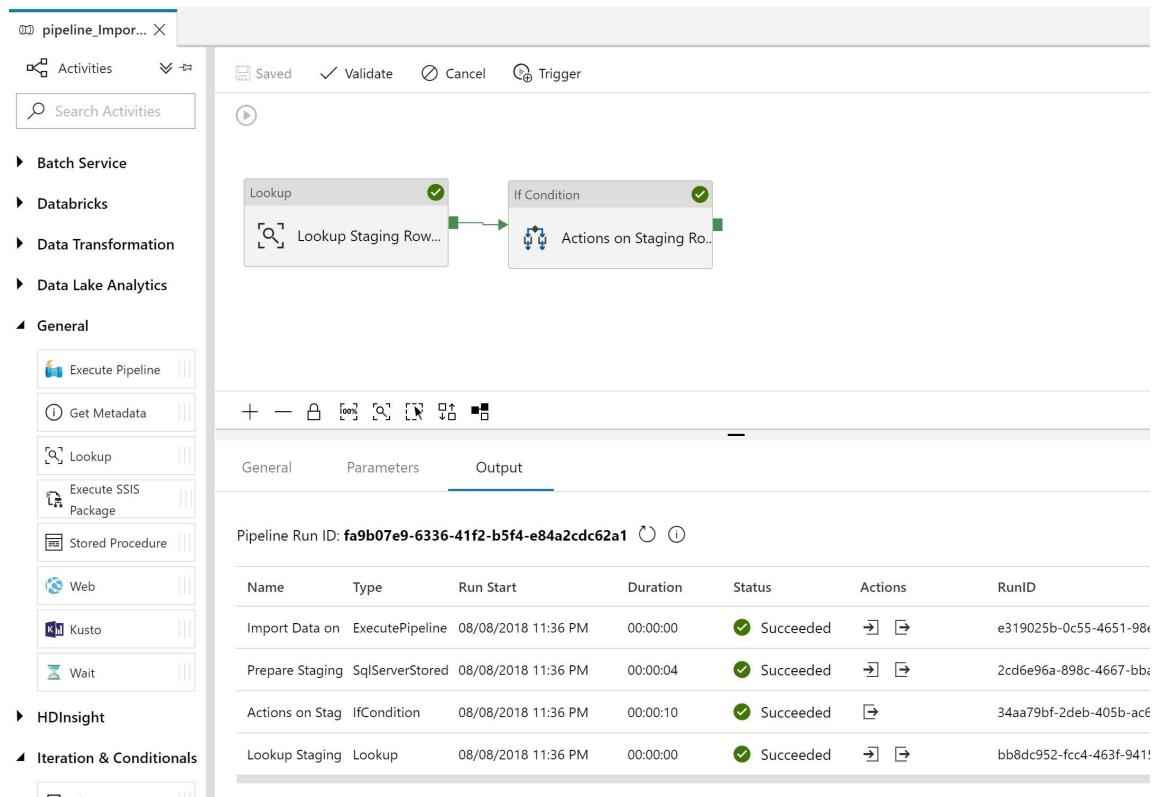
w. Select **Save** on the Pipeline to save changes, if using Git.

x. Click **Publish** to deploy your changes to the Data Factory service.

## 5. Execute the Pipeline

a. Using the same methods as in the earlier exercise execute the pipeline

**pipeline\_ImportData\_Control** and observe the paths used. In the following screenshot the execution has followed the False conditional path.



6. You could try truncating the table manually and then executing again to force the activity to follow the true path.

► `EXEC [SalesLT].[usp_TruncateProductStaging]`

Exercise 3 has been completed.





## Data Flow

---

### **Introduction**

During this lab, you will create a Data Flow task in Azure Data Factory, joining different data sources together.

### **Estimated Time**

90 minutes

### **Objectives**

At the end of this lab, you will be able to:

- Configure a Data Flow pipeline.
- Use debugging in Data Flow.
- Execute a Data Flow pipeline from an ADF Pipeline.

### **Logon Information**

Use the following credentials to login into virtual environment.

- Username: **Admin**
- Password: **Passw0rd!**

## Table of Contents

---

[Setup: Prepare your environment](#)

[Exercise 1: Implement Control Flow](#)

[Exercise 2: Create Datasets](#)

[Exercise 3: Create a Data Flow Pipeline](#)

[Exercise 4: Create a Pipeline to run the Data Flow](#)

[Exercise 5: Run the Pipeline and Data Flow](#)

## Lab: Transform, Merge and Join Data in Mapping Data Flow

During this lab, you will learn how to do data transformations with Azure Data Factory Data Flow by looking at the reading history of 4 people, Keiko Brooks, Minerva Snape, Sven Dorjadt and Trixie Belden. In addition to the books these people have read, there is a file which contains information about a number of books, many (but not all) of which have been read by our readers.

We are going to learn how to join the reading history files (some of which have different columns) to each other, as well as to the list of books.

### Setup: Prepare your environment

This step will take you through preparing your environment for the exercises which will follow.

#### Tasks

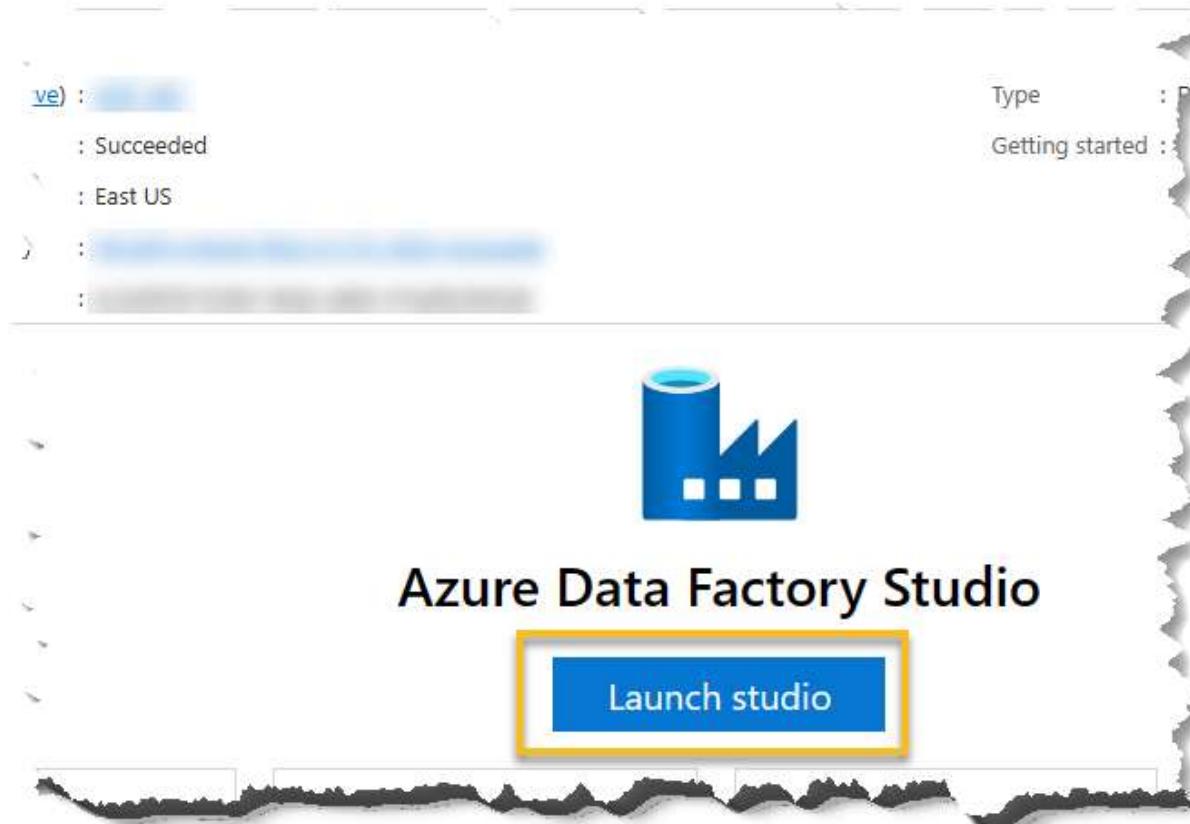
1. Open Internet Explorer and navigate to <http://portal.azure.com/> to connect to Microsoft Azure Portal. Login with your subscriptions credential.
2. Create source files in new Container.
  - a. Navigate to the storage account you have been using for this workshop, and create a new container named **module5**.
  - b. In this container, using Storage Explorer, upload the following files from LabFiles\M05\_L01\_Lab01 to a folder named **originals**:
    - Books.csv
    - Keikos\_History.csv
    - Minervas\_History.csv
    - Svens\_History.csv
    - Trixies\_History.csv

### Exercise 1: Setting up Linked Service

This exercise shows the benefits of using Control Flow within your pipeline.

#### Tasks

1. Log into the Azure Data Factory you created earlier in this workshop.
2. Select the **Launch Studio** tile to start the Azure Data Factory user interface (UI) application.



3. Configure the Azure Storage Linked Service in ADF

- a. Verify that you have configured the **Linked Service** to the **Azure Blob Storage account**. You may have an existing connection from previous lessons. If you do, skip to [Exercise 2](#).
- b. Click the **Toolbox/Manage** button on the left panel.

# Microsoft Azure

>>



c. Under **Connections** in left panel, click on **Linked services**, if not already selected, and then click on **+New** in the middle pane.

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has a 'Connections' section with 'Linked services' highlighted by a green box. Below it are 'Integration runtimes', 'Source control', 'Git configuration', 'Parameterization template', 'Author', 'Triggers', 'Global parameters', 'Security', 'Customer managed key', and 'Managed private endpoints'. The main pane is titled 'Linked services' and contains a message: 'Linked service defines the connection information to a data store or compute. Learn more'. It features a '+ New' button (also highlighted with a green box), a 'Filter by keyword' input field, and a 'Annotations : Any' dropdown. Below these are three columns: 'Name ↑', 'Type ↑↓', and 'Annotations ↑↓'. A large 'Create linked service' button with a plus sign icon is located at the bottom right. The top navigation bar includes 'Validate all', 'Save all', 'Publish', 'Refresh', 'Discard all', 'Data flow debug', and 'ARM template'.

d. Select the **Azure Blob Storage** type and click the **Continue** button.

## New linked service

Data store      Compute

---

blob

All    Azure    Database    File    Generic protocol    NoSQL    Services and apps

---



Azure Blob Storage

---

**Continue**      **Cancel**

- e. Change the Linked Service name if desired (for example: use the storage account name).
- f. Select the **Azure subscription** from the drop-down.
- g. Select the **Storage account name** from the drop-down and click on **Test Connection** test the connection.
- h. Once successfully tested, select **Create**.

## New linked service (Azure Blob Storage)

Name \*  
BlobStorage

Description

Connect via integration runtime \*  
AutoResolveIntegrationRuntime

Authentication method  
Account key

**Connection string**      Azure Key Vault

Account selection method  
 From Azure subscription       Enter manually

Azure subscription  
Select all

Storage account name \*

Additional connection properties  
+ New

Test connection  
 To linked service       To file path

If the identity you use to access the data store only has permission to subdirectory instead of the entire account, specify the path to test connection. Please make sure your self-hosted integration runtime is higher than version 4.0 if connecting via self-hosted integration runtime.

Annotations  
+ New

► Advanced 1

**Create**      **Back**       **Test connection**      **Cancel**

Exercise 1 has been completed.

## Exercise 2: Create Datasets

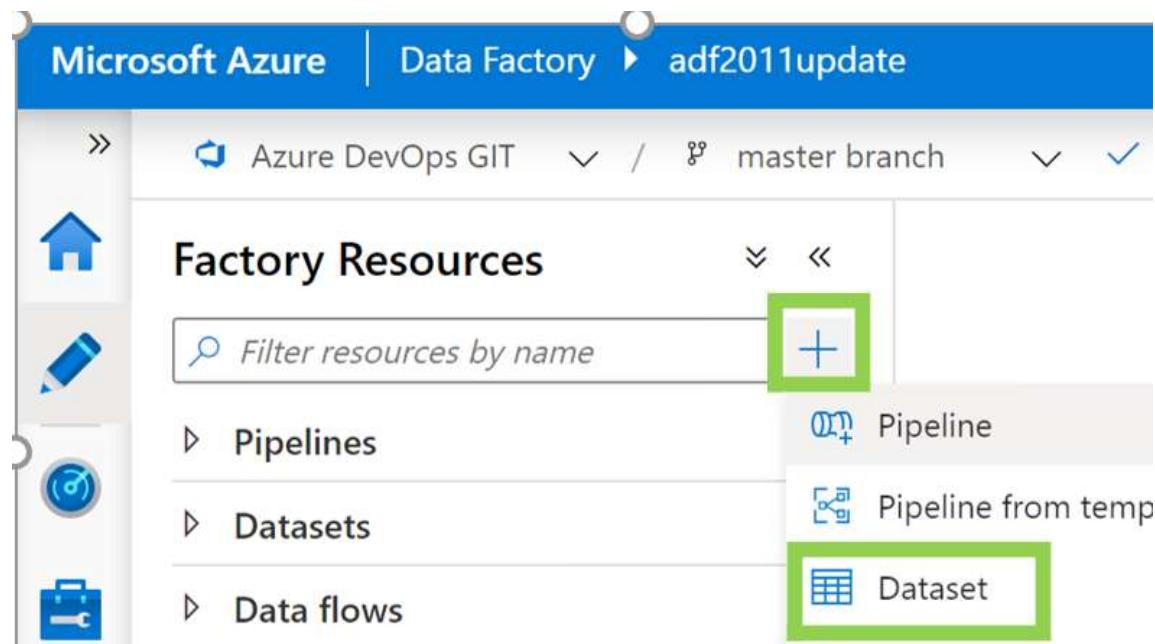
1. Create datasets which reference the files uploaded to Azure Storage earlier in this lab.
  - a. Click on the **Pencil/Edit** in the left pane.

# Microsoft Azure

>>



b. Create a new dataset by clicking the + button under **Factory Resources** and selecting **Dataset**.



c. Select **Azure Blob Storage** and click the **Continue** button.

## New dataset

Select a data store



All Azure Database File Generic protocol NoSQL Services and apps



Azure Blob Storage

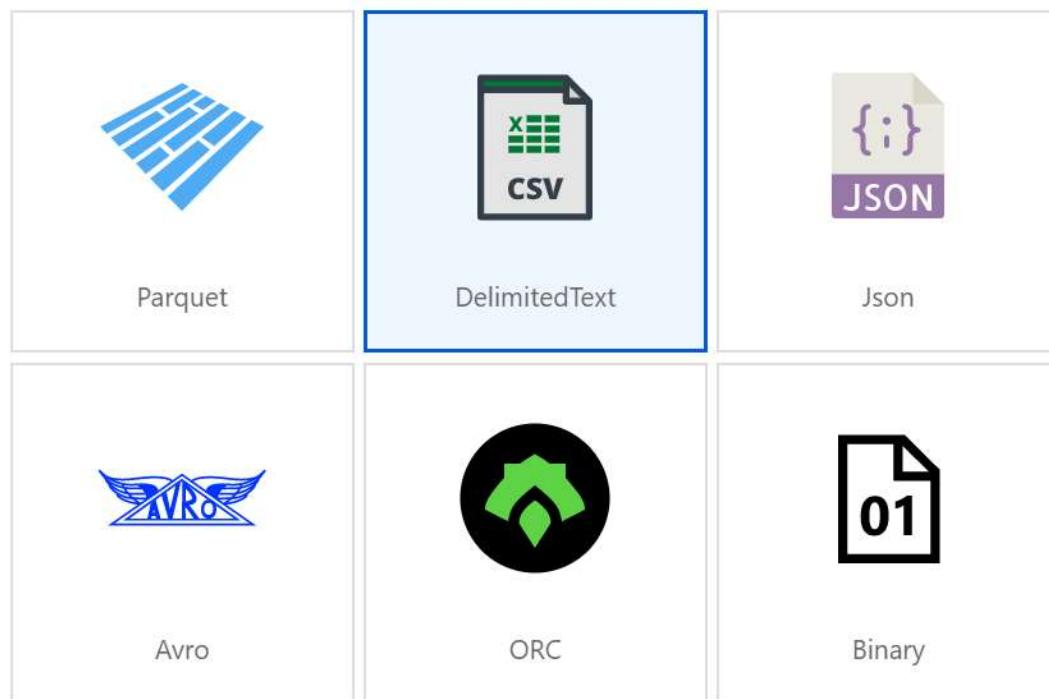
**Continue**

**Cancel**

- d. Select **DelimitedText** and click the **Continue** button.

## Select format

Choose the format type of your data



[Continue](#)

[Back](#)

[Cancel](#)

e. Enter **bookscsv** for the Name and select the **Linked Service** created for Blob Storage earlier in this exercise.

f. Click on the **Folder** button to the right of the **File path** boxes.

## Set properties

Name

bookscsv

Linked service \*

adf2011update



File path

Container

/ Directory

/ File



g. Drill down to the books.csv file uploaded earlier in the exercise by double clicking on the module5 container, double clicking the originals folder, and selecting the books.csv file. Click **OK**.

## Choose a file or folder

↑ > module5 > originals >

originals

Keikos\_History.csv

Minervas\_History.csv

Svens\_History.csv

Trixies\_History.csv

books.csv

OK

h. Checkmark the box for **First row as header**. Click the **OK** button.

## Set properties

Name

Linked service \*

File path



First row as header



Import schema

From connection/store  From sample file  None

- i. Repeat steps 1 (ii) through 1 (viii) to create Datasets for the other files in your **module5/originals** folder in your Storage account. When you are done, you should have added 5 Datasets to your Data Factory.

# Factory Resources

⌄ ⌂



Filter resources by name



## ▶ Pipelines

0

## ◀ Datasets

5

bookcsv

keikos

Minervas

svens

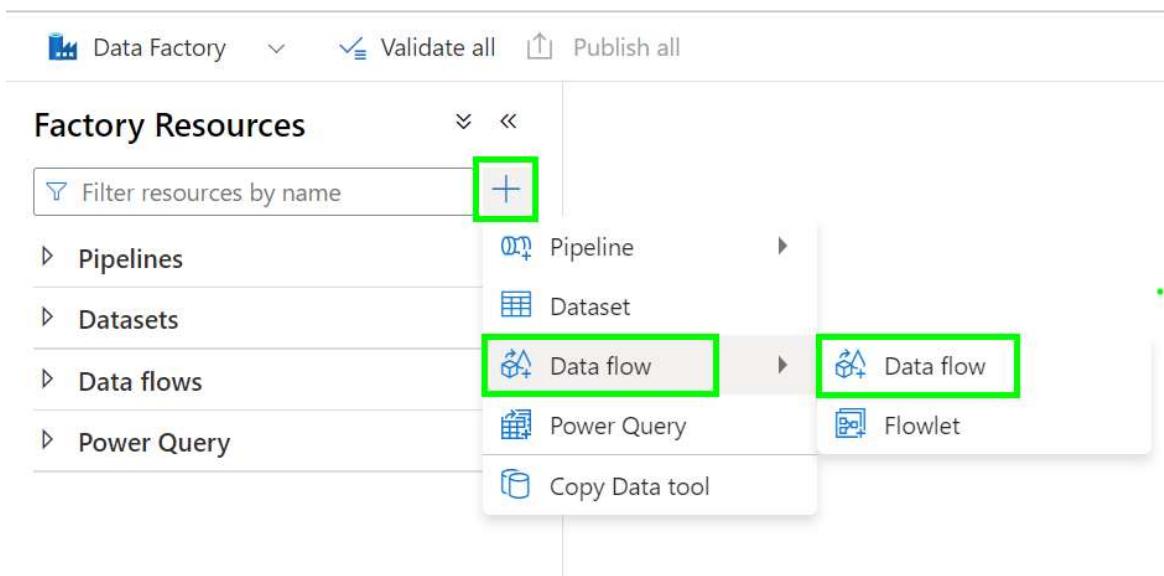
trixies

2. Now is a good time to save/publish your work.

Exercise 2 has been completed.

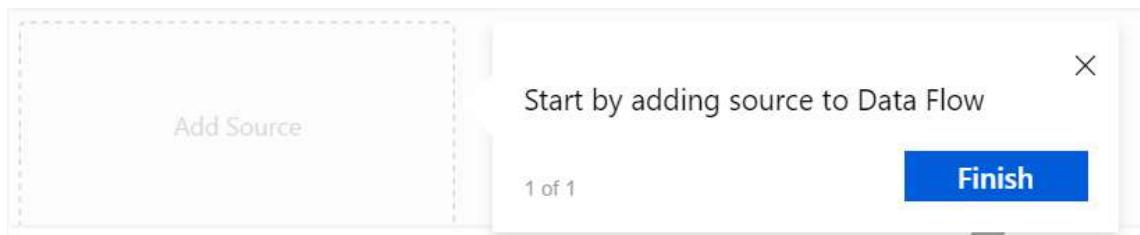
## Exercise 3: Create a Data Flow Pipeline

1. Create a new data flow by clicking the + button, hovering over **Data Flow** and selecting **Data Flow** from subsequent list.



2. In the new tab, a pop-up window may show with the text **Start by adding source to Data Flow**. If so, click the **Finish** button.

a. Similar popups will show while using Data Flow to help. Read these, including all pages if desired, and click the **Finish** button once read.



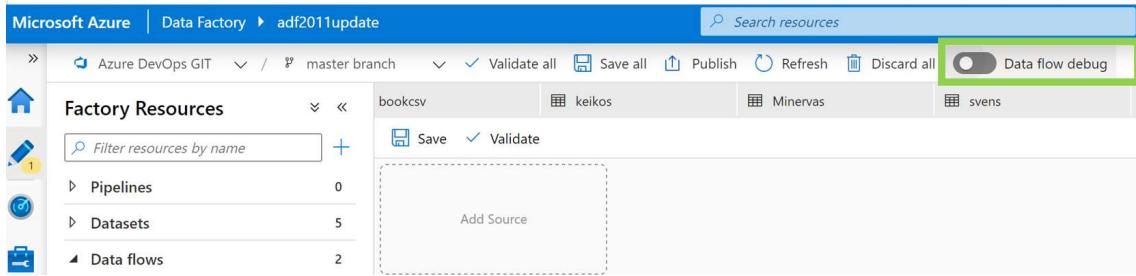
b. In the Data Flow Properties pane to the right, enter **BookReading** for the Name.

3. Enable Data flow debug

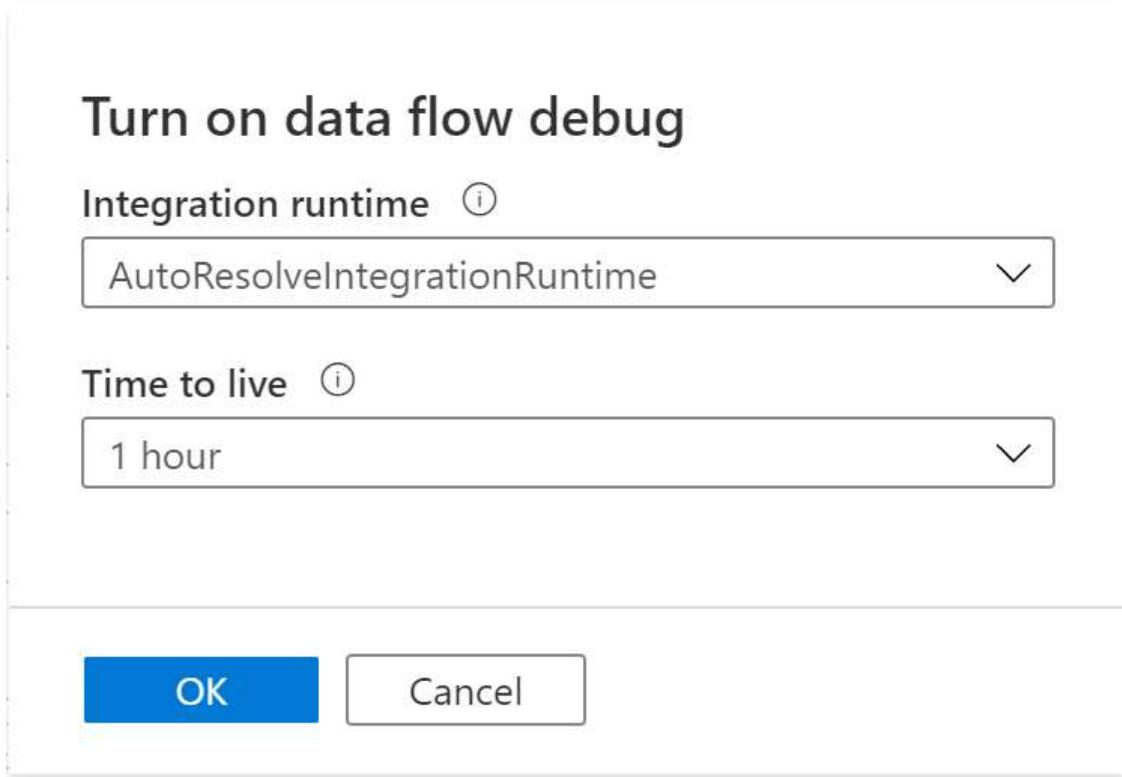
- Data flow debug allows you to preview your data while developing your Data flow. When enabled, it will start a cluster on an Azure Integration Runtime, which takes some time to

start up.

- Click on the Data flow debug slider in the ribbon.

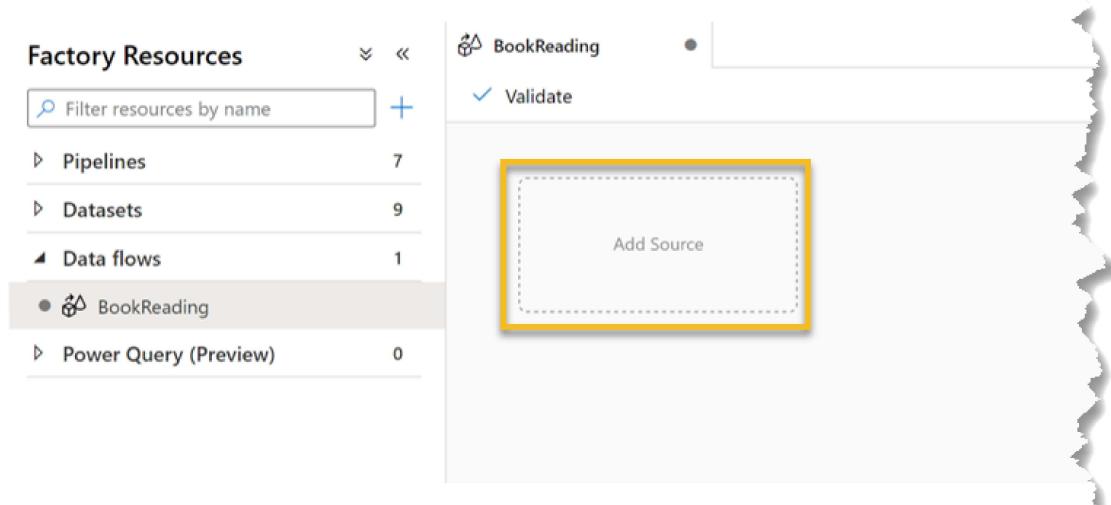


- When prompted for **Turn on data flow debug**, click **OK**.



- Add source data to the Data Flow

- Select the **Add Source** rectangle in the center pane.



- b. Click on the Dataset dropdown box and select the **bookscsv dataset** created in Step 3 for the books.csv file. Enter **BooksCSV** for the Output stream name.

Source settings    Source options    Projection    Optimize    Inspect    Data preview

---

Output stream name * <input type="text" value="BooksCSV"/> <a href="#">Learn more</a>	Source type * <input type="text" value="Dataset"/>
Dataset * <input type="text" value="bookscsv"/> <a href="#">Test connection</a> <a href="#">Open</a> <a href="#">New</a>	
Options <input checked="" type="checkbox"/> Allow schema drift <small> ⓘ </small> <input type="checkbox"/> Infer drifted column types <small> ⓘ </small> <input type="checkbox"/> Validate schema <small> ⓘ </small>	
Skip line count <input type="text"/>	
Sampling * <small> ⓘ </small> <input type="radio"/> Enable <input checked="" type="radio"/> Disable	

- c. Select the **Projection** tab and set the following columns to integer type, with a format of `#0`:

- Book\_ID
- Number\_of\_Pages
- Year\_Published
- Original\_Publication\_Year

- d. Set the following columns to float, and type 0.00 in the format box. (0.00 is not shown as an option in the dropdown list. You can manually enter your own format strings).

- Average\_Rating

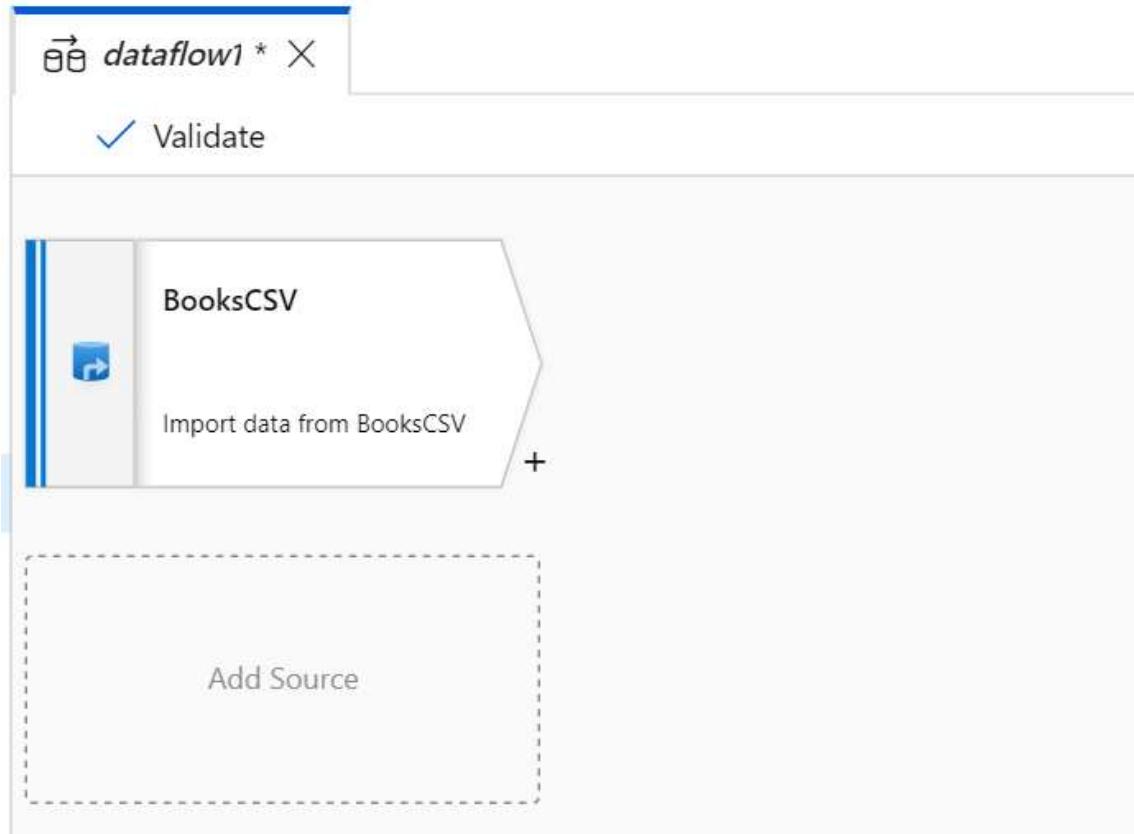
- e. Leave the remaining columns as string.

Source settings	Source options	Projection	Optimize	Inspect	Data preview
		Define default format	Detect data type	Import projection	Reset schema
Column name	Type			Format	
Book_Id	123 integer	▼	##0	▼	
ISBN	abc string	▼	Specify format	▼	
ISBN13	abc string	▼	Specify format	▼	
Title	abc string	▼	Specify format	▼	
Author	abc string	▼	Specify format	▼	
Author_l-f	abc string	▼	Specify format	▼	
Additional_Authors	abc string	▼	Specify format	▼	
Average_Rating	1.2f float	▼	0.00	▼	
Publisher	abc string	▼	Specify format	▼	
Binding	abc string	▼	Specify format	▼	
Number_of_Pages	123 integer	▼	##0	▼	
Year_Published	123 integer	▼	##0	▼	
Original_Publication_Year	123 integer	▼	##0	▼	

f. Select the **Data Preview** tab and click **Refresh** to preview the data.

- If your Debug cluster is still starting, you will see a "Please wait.." message. Click **Refresh** to display the data after the cluster has started.)

g. Select the **Add Source** area under the last source you created.



h. Under Source Settings, name the **Output Stream** TrixiesHistory and select the **dataset** **Trixies\_History** created earlier in this lab.

i. Select the Projection tab and set the following columns to integer type, with a format of ##0:

- Book\_ID
- My\_Rating
- Read\_Count

j. Set the following columns to **date**, and set the format to **dd/MM/yyyy** (this is a result of the format of the dates in the files):

- Date\_Read
- Date\_Added
- Original\_Purchase\_Date

k. Set the following columns to boolean:

- Spoiler

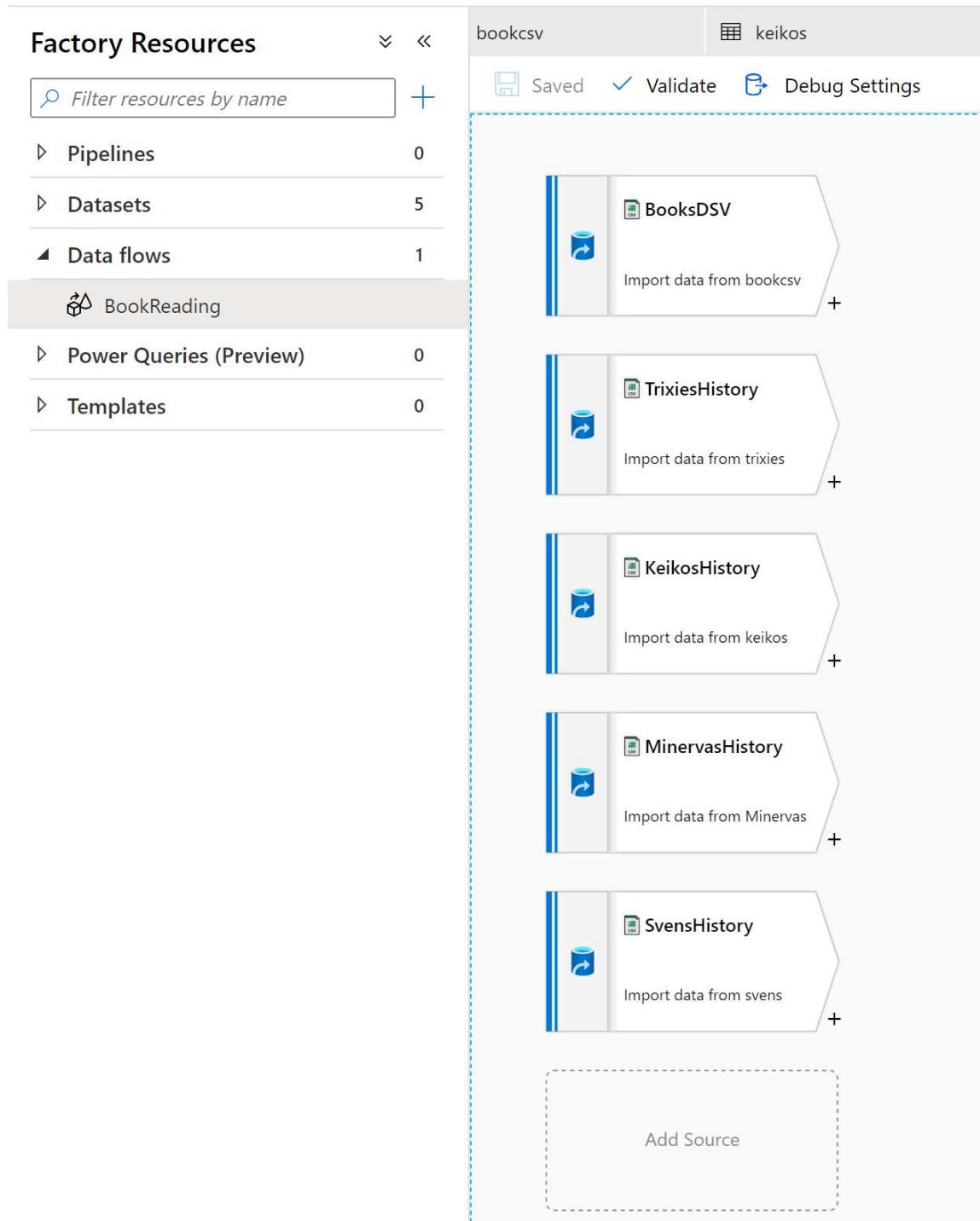
l. Leave the remaining columns as string.

Source Settings		Source Options	Projection	Optimize	Inspect	Data Preview
		Define default format	Detect data type	Reset schema		
Column name	Type			Format		
Book_Id	123 integer			##0		
ISBN	abc string			Specify format		
ISBN13	abc string			Specify format		
My_Rating	123 integer			##0		
Date_Read	date			dd/MM/yyyy		
Date_Added	date			dd/MM/yyyy		
Bookshelves	abc string			Specify format		
Exclusive_Shelf	abc string			Specify format		
My_Review	abc string			Specify format		
Spoiler	✗ boolean			Specify format		
Private_Notes	abc string			Specify format		
Read_Count	123 integer			##0		
Original_Purchase_Date	date			dd/MM/yyyy		

m. Go to the **Data Preview** tab and click Refresh to preview the data.

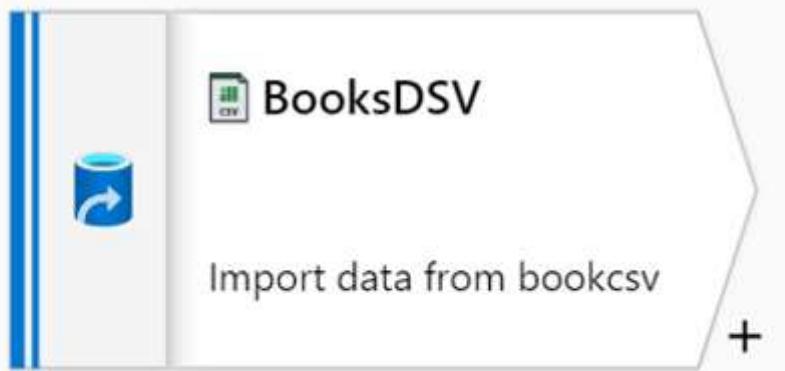
n. Repeat steps 4 (viii) through 4 (xiii) for the other svens, minervas and keikos reader history files, changing names as appropriate.

o. You should now have 5 data sources on your canvas.



5. Prepare reading history data to be joined. The intent is to join the four reading history files together, along with the books.csv file, to output the data in a single file. The four readers are not identified in their files, so we need to create a column to identify them.

a. Click the + plus sign next to TrixiesHistory on the pipeline canvas.





- b. From the popup list, select **Derived Column**.
- c. Name the output stream **AddReaderNameT**.
- d. For Columns type, enter the column name **Reader** and for the expression enter **Trixie Belden**, enclosed in single quotes.

Derived column's settings    Optimize    Inspect    Data preview ●

Output stream name \*  [Learn more](#)

Incoming stream \*

[+](#) Add   [Clone](#)   [Delete](#)   [Open expression builder](#)

Columns \* ⓘ

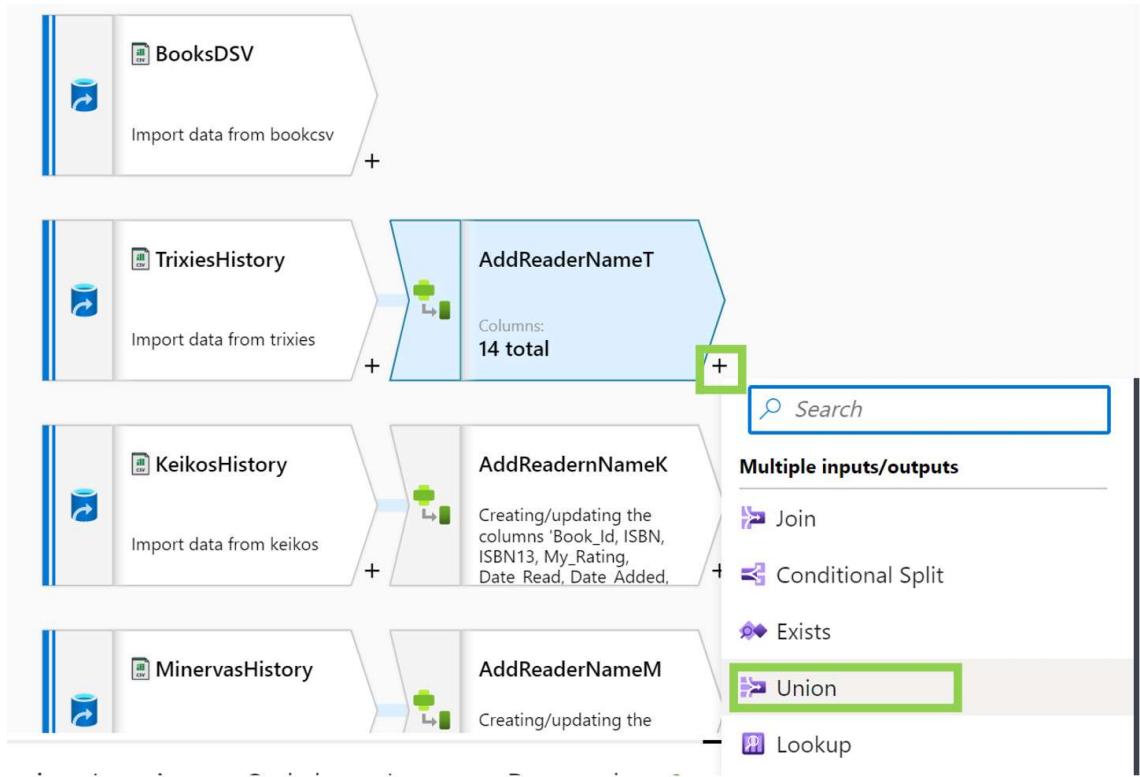
<input type="checkbox"/> Column	Expression
<input type="checkbox"/> Reader	'Trixie Belden'

- e. Repeat step 5 (i) through 5 (iv) to create a Derived Column called Reader for the other Reader History data sources, entering their full names as shown below:

Source	Output stream Name	Reader column Expression
Minerva	AddReaderNameM	Minerva Snape
Sven	AddReaderNameS	Sven Dorjadt
Keiko	AddReaderNameK	Keiko Brooks

- 6. Join all the reader history data into a single stream.

- a. Select the + **plus** sign next to **AddReaderNameT** and select **Union**.



b. Name the output stream **AllReaders**. In the **Union with** field select **AddReadernNameK**.

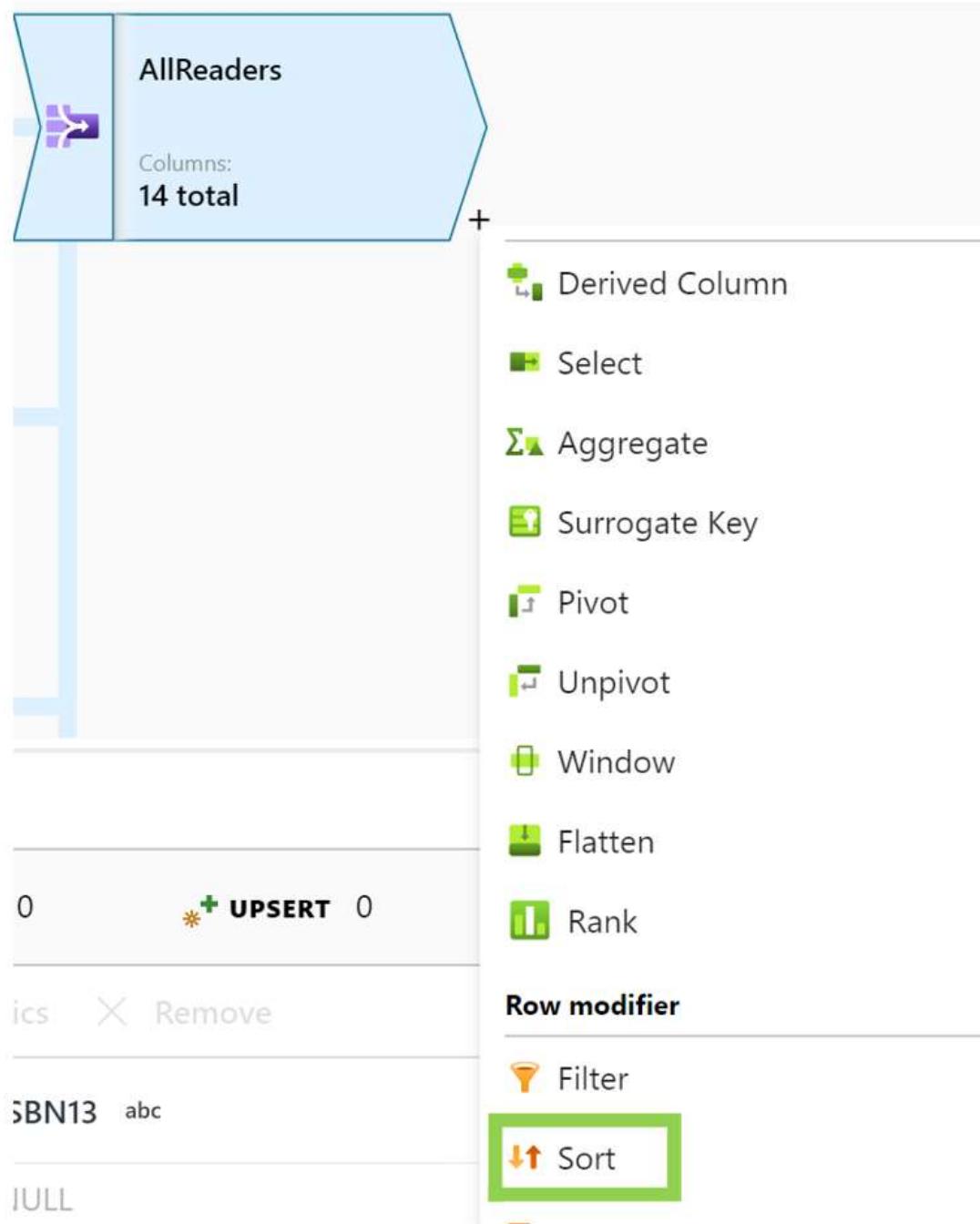
Click the plus sign and add rows for **AddReaderNameS** and **AddReaderNameM**, so all readers have their reading history in one stream.

Union settings		Optimize	Inspect	Data preview	Description
Output stream name *	AllReaders				Learn more <a href="#">↗</a>
Incoming stream *	AddReaderNameT				
Union by *	<input checked="" type="radio"/> Name <input type="radio"/> Position <a href="#">?</a>				
Union with	Streams				
	AddReadernNameK				

c. Under the Inspect tab, you can see how the columns have been aligned for the join, including the blanks from the columns missing from the files for KeikosHistory and MinervasHistory. Also note that the out of order columns are now aligned.

7. Inspect the data to be sure all 4 datasets have been joined.

a. Select the **Plus + sign** next to AllReaders and select **Sort** under the **Row modifier** heading from the popup list of transformations.



- b. Name the output stream **SortReadingHistory**. In **Sort conditions** and select the column **Book\_Id** and **Order** ascending.

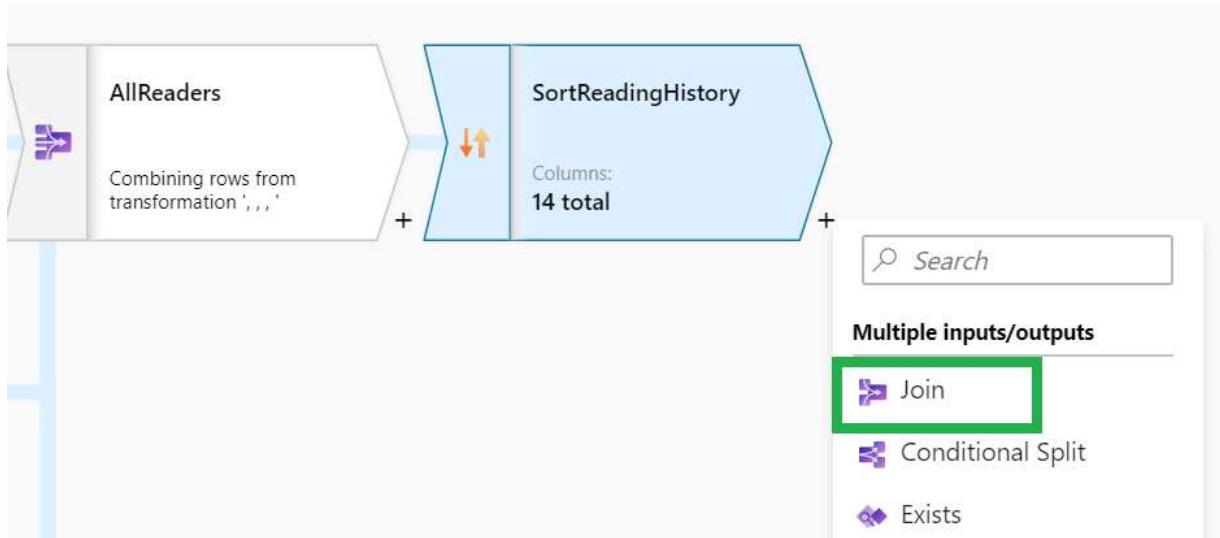
Sort conditions *	AllReaders's column	Order	Nulls first
	123 Book_Id	Ascending	<input checked="" type="checkbox"/>

**i** This isn't strictly necessary but provides an easy way within the debug mode to show that data is flowing in from different source files, given our data set.

c. Go to the **Data Preview** tab to review the data, and you should be able to find records from all 4 of our readers.

8. Join the reader and book data together.

a. Select the **Plus + sign** next to the **SortReadingHistory** Output stream and select **Join**.



b. Name the output stream **JoinReaderstoBooks** and select **BooksCSV** from the **Right stream** dropdown.

c. Set the Join type to **Inner** and add 2 columns to the Join conditions: **Book\_Id** and **ISBN**.

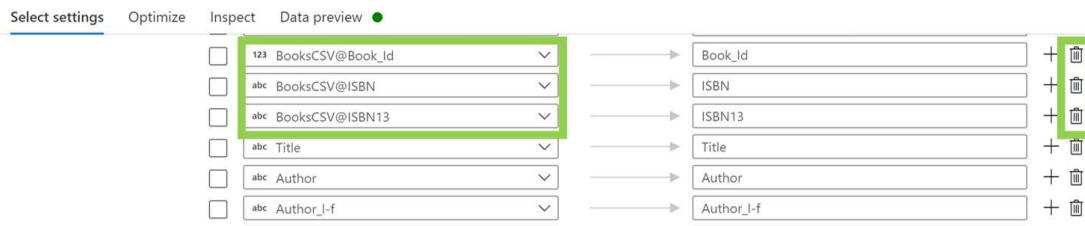
This screenshot shows the 'Join Settings' tab of the Data Flow Designer. It includes fields for 'Output stream name' (set to 'JoinReaderstoBooksInner'), 'Left stream' (set to 'SortReadingHistory'), 'Right stream' (set to 'BooksCSV'), 'Join type' (set to 'Inner', indicated by a blue selection box), and 'Join conditions'. Under 'Join conditions', there are two rows: one for 'Left: SortReadingHistory's column' (with '123 Book\_Id' and 'abc ISBN') and one for 'Right: BooksCSV's column' (with '123 Book\_Id' and 'abc ISBN').

d. Go to the **Data Preview** tab which now shows duplicate columns in the dataset, Book\_Id, ISBN and ISBN13. We should remove them to avoid any confusion for the users of our data.

9. Remove resulting duplicate columns.

a. Select the **Plus + sign** next to the **JoinReaderstoBooks** output stream and choose the **Select** transformation under the **Schema Modifier** heading.

b. Name the Output stream **RemoveDuplicateColumns**. Scroll down through the Input column list until you see the duplicate column names from the BooksCSV output stream. Remove these duplicated columns (BooksCSV@Book\_Id, BooksCSV@ISBN and BooksCSV ISBN13) from the input columns by clicking on the delete buttons at the end of each row.



10. Output the joined books and reader data

- Select the **Plus + sign** next to the **RemoveDuplicateColumns** Output stream, select the **Sink** transformation under the **Destination** heading and name it **ReadersBooksInnerJoin**.
- Select **+New** next to the **Dataset** field to create a new dataset.
- Choose **Azure Blob Storage** for the data store type.
- Select **DelimitedText** for the format. Click **Continue**.
- Enter **ReadersBooksInnerJoin**, for the name.
- Select the same Linked service used throughout this lab. Browse to your container name or manually enter module5 as the container name. (If you named your container something else, use this name instead.)
- Check **First row as header** box
- Select **None** for **Import Schema**
- Select **OK**.

### Set properties

Name  
ReaderBooksInnerJoin

Linked service \*  
adf2011update

File path  
module5 / Directory / File

First row as header

Import schema  
 From connection/store    From sample file    None

**Advanced**

- Click on the **Settings** next to Sink, select **Output to single file** and enter the filename of **InnerJoinDataset.csv**.

**i** The warning about writing large datasets to a single file. This is not a concern with our dataset.

Sink    **Settings**    Mapping    Optimize    Inspect    Data Preview

**⚠ Writing to a single file with large datasets may cause the cluster to run out of memory. Consider using other naming options to prevent data flow failures.**

Clear the folder

File name option \*  Default  Pattern  Per partition  As data in column  Output to single file

Output to single file \*

Quote All

k. On **Optimize** tab, set partition option to be **Single partition**.

Sink    Settings    Mapping    **Optimize**    Inspect    Data preview

Partition option \*  Use current partitioning  Single partition  Set Partitioning

11. Second join of the books and reader data

a. Select the **Plus + sign** next to the **BooksCSV** Source and select **Join**.

BooksCSV  
13 total

+

TrixiesHistory  
Import data from trixies

**Multiple inputs/outputs**

b. Name this output stream **JoinReadersToBooksOuter**.

c. Select **SortReaderHistory** as the **Right stream** and Right outer as the **Join Type** and add 2 columns to the Join conditions: **Book\_Id** and **ISBN**. This output stream will include all reader history records even if there is no matching record in the BooksCSV source.

12. Second removal of duplicate columns

- a. Select the **Plus + sign** next to the **JoinReadersToBooksOuter** Output stream and choose **Select**.
- b. Name this Output stream **RemoveDuplicateColumns**, and observe the error from Data Flow, which does not allow multiple transformations to have the same names. Add a **2** to the end of the name.

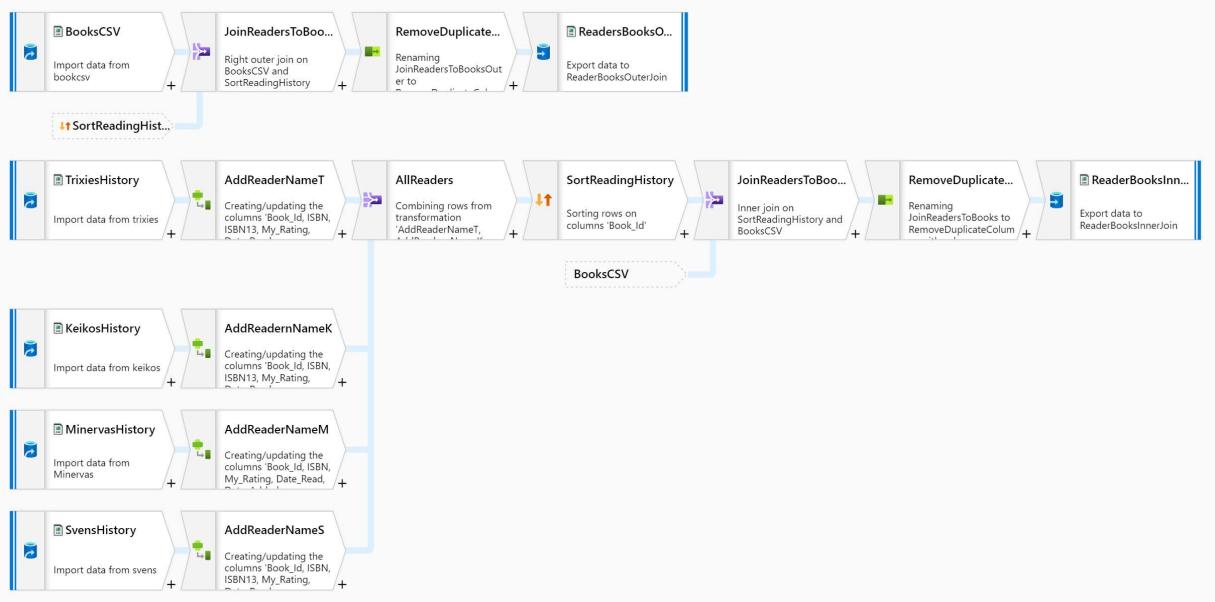
13. Again, remove the duplicated columns (Book\_Id, ISBN and ISBN13) from the Books file.

- a. Select the **Plus + sign** next to the **RemoveDuplicateColumns2** Output stream and choose **Sink**.
- b. Name this Output stream **ReadersBooksOuterJoin**.

14. Select **+New** next to the **Dataset** field to create a new dataset.

- a. Choose **Azure Blob Storage** for the data store type.
- b. Select **DelimitedText** for the format. Click **Continue**.
- c. Enter **ReadersBooksOuterJoin**, for the name.
- d. Select the same Linked service used throughout this lab. Browse to your container name or manually enter module5 as the container name. (If you named your container something else, use this name instead.)
- e. Check **First row as header** box
- f. Select **None** for **Import Schema**
- g. Select **OK**.
- h. Click on the **Settings** next to Sink, select **Output to single file** and enter the filename of **ReadersBooksOuterJoin.csv**. and select **Output to single file** name.
- i. Make sure to select **Single partition** on the **Optimize** tab.

15. Your Data Flow should look similar to the screenshot below.

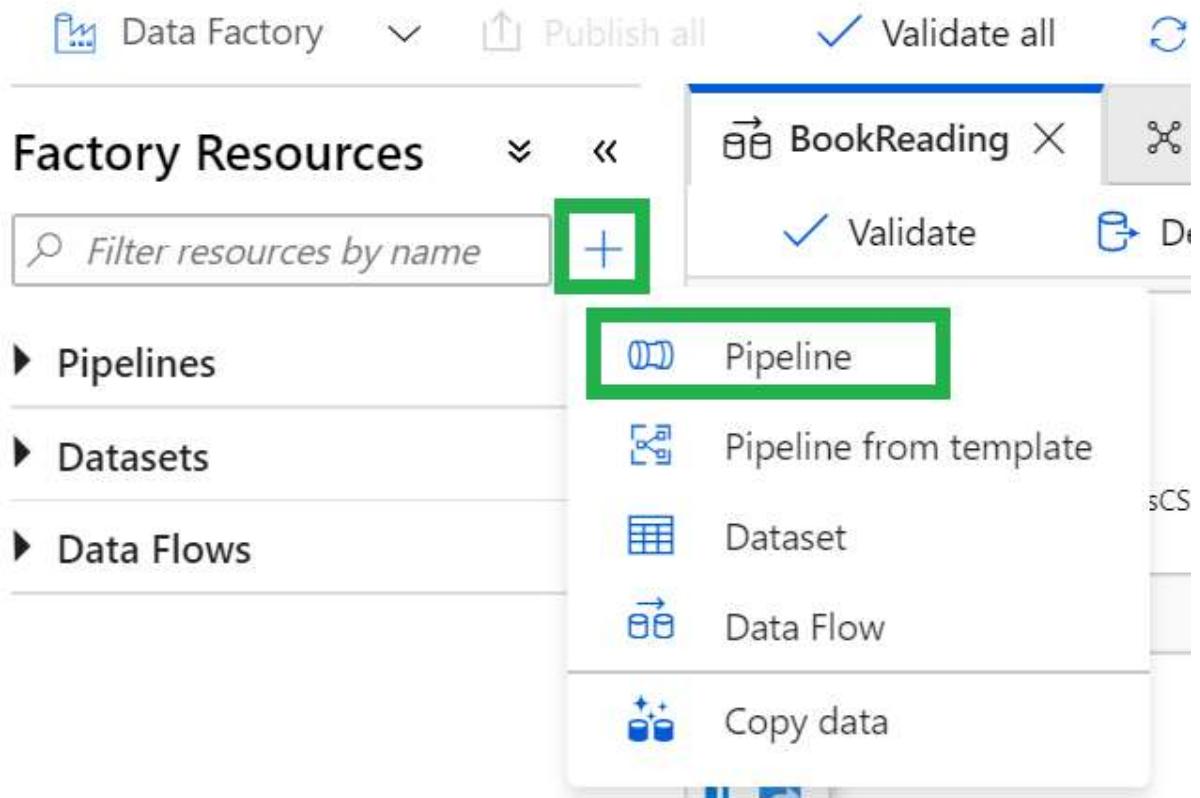


## 16. Save and Publish the Data Flow.

Exercise 3 has been completed.

## Exercise 4: Create a Pipeline to run the Data Flow

1. Select the **Plus Sign +** next to the Filter resources by name box and select **Pipeline** to create a new pipeline.



2. Rename the pipeline **pIRunBookReadingDataflow**

# Properties

## General

Name \*

plRunBookReadingDataflow

3. Expand **Move & Transform** and drag the **Data Flow** activity onto the canvas.

## Activities

▼ <<

Search activities

### ▲ Move & Transform

 Copy data

 Data Flow

4. Under General, name the activity **Run BookReading Flow**.

General    Settings <sup>1</sup>    Parameters <sup>1</sup>    User properties

Name *	Run BookReading Flow	<a href="#">Learn more</a>
Description		
Timeout	7.00:00:00	(i)
Retry	0	(i)
Retry interval	30	(i)
Secure output	<input type="checkbox"/>	(i)
Secure input	<input type="checkbox"/>	(i)

5. Click on the **Settings** tab. Select the **BookReading** data flow from the Data flow drop down. Leave the Integration Runtime default values.

General    **Settings**    Parameters <sup>1</sup>    User properties

Data flow *	BookReading	<a href="#">Open</a>	<a href="#">New</a>
Run on (Azure IR) *	AutoResolveIntegrationRuntime	(i)	
Compute type *	General purpose	(i)	
Core count *	4 (+ 4 Driver cores)	(i)	
Logging level *	<input checked="" type="radio"/> Verbose <input type="radio"/> Basic <input type="radio"/> None	(i)	
PolyBase <a href="#">(i)</a>			

6. Select **Publish all**.

Exercise 4 has been completed.

## Exercise 5: Run the Pipeline and Data Flow

- Once publishing has completed successfully, choose **Trigger**, select **Trigger now** and press **Ok**.
- Select Monitor on the left side of the portal, and see the pipeline running. Click on the **Pipeline name**.

The screenshot shows the Microsoft Azure Data Factory interface. In the left sidebar, the 'Runs' section is selected, highlighted with a green box. A single pipeline run for 'plRunBookReadingDataflow' is listed in the main pane, also highlighted with a green box. The run details are as follows:

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
plRunBookReadingDataflow	11/11/20, 9:05:12 PM	--	00:01:10	Manual trigger	In progress	Original

3. Select the glasses icon to see the **details** of the activity.

## Activity runs

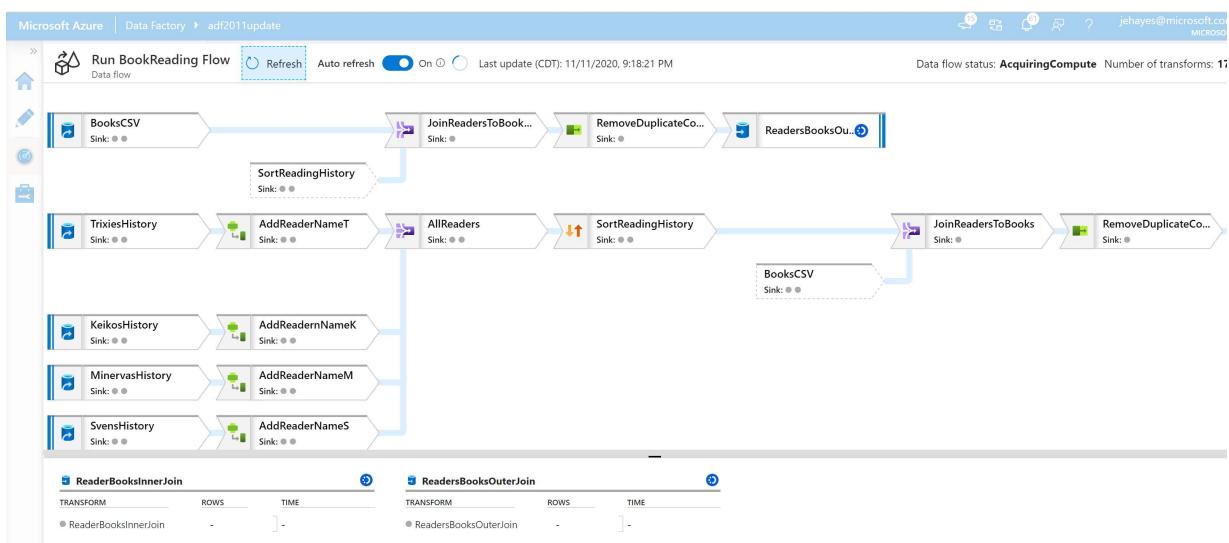
Pipeline run ID bf2ac3e4-b23c-4062-b360-ab0ae1bf7fcf

All status ▾

Showing 1 - 1 of 1 items

Activity name	Activity type	Run start ↑	
Run BookRea... ➔ ➔	ExecuteDataFlow	11/11/20, 9:05:14 PM	⋯

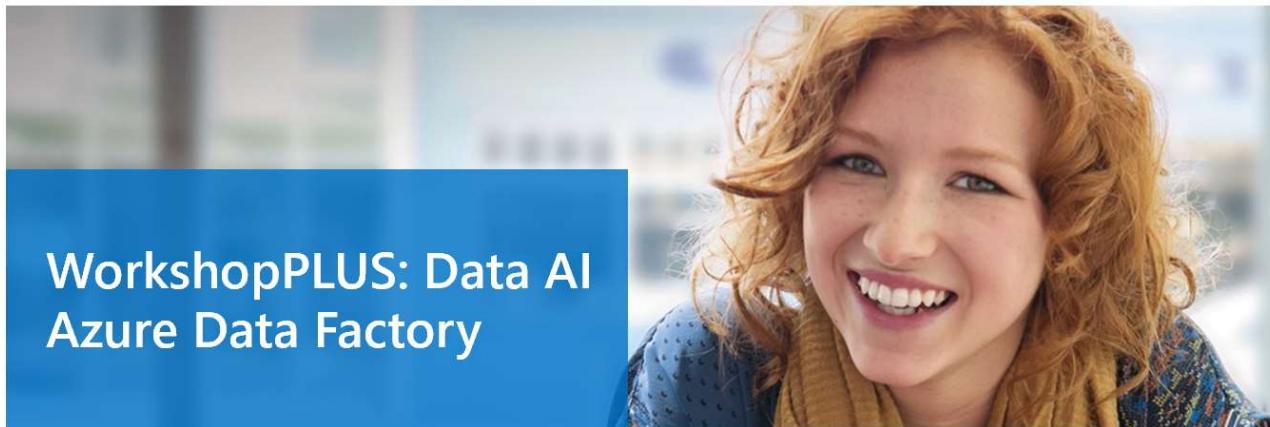
4. The screenshot below shows the transformation activities being run. The screen will refresh every 60 seconds until the run is complete if Auto refresh is on (as shown below.) Note the time AcquiringCompute in the upper right corner - this will take approximately 6 minutes.



5. After the Data flow is complete, you can see the statistics for the time and number of rows processed. Compare the outputs from the two csv files, ReadBooksInnerJoin and ReadBooksOuterJoin . Why were there differences in the rows returned in each of the csv files?

TRANSFORM	ROWS	TIME	TRANSFORM	ROWS	TIME
● AllReaders	1886		● AllReaders	1886	
● AddReaderNameT	737		● AddReaderNameT	737	
● TrixiesHistory	737		● TrixiesHistory	737	
● AddReadernNameK	304		● AddReadernNameK	304	
● KeikosHistory	304		● KeikosHistory	304	
● AddReaderNameM	775		● AddReaderNameM	775	
● MinervasHistory	775		● MinervasHistory	775	
● AddReaderNameS	70		● AddReaderNameS	70	
● SvensHistory	70		● SvensHistory	70	
● SortReadingHistory	⌚ 1886	635ms	● ReadersBooksOuterJoin	1911	635ms
● ReaderBooksInnerJoin	1265	11s 912ms	● RemoveDuplicateColum...	1911	
● RemoveDuplicateColumns	1265		● JoinReadersToBooksOuter	1911	
● JoinReadersToBooks	1265		● SortReadingHistory	⌚ 1886	11s 912ms
● BooksCSV	⌚ 2269	517ms	● BooksCSV	⌚ 2269	517ms

Exercise 5 has been completed.



## Using Azure Key Vault

---

### Introduction

During this lab, you will learn how to store and retrieve credentials used by Azure Data Factory linked services in Azure Key Vault, improving the security of your environment.

### Estimated Time

20 minutes

### Objectives

At the end of this lab, you will be able to:

- Configure Data Factory to communicate with Azure Key Vault.
- Retrieve secrets from Azure Key Vault in Azure Data Factory.

### Logon Information

Use the following credentials to sign into virtual environment.

- Username: **Admin**
- Password: **Passw0rd!**

## Table of Contents

---

[Exercise 1: Create the Data Factory Service Identity](#)

[Exercise 2: Store and Retrieve secrets in Azure Key Vault](#)

**Lab: Azure Data Factory Security**

During this lab, you will learn how to securely store secrets in Azure Key Vault for use with Azure Data Factory.

## Exercise 1: Create the Data Factory Service Identity

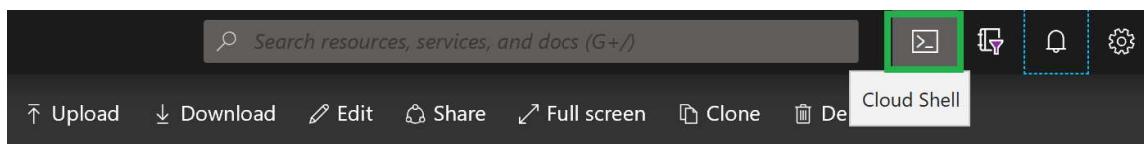
This exercise shows how to create the Azure Data Factory Service Identity.

### Tasks

1. Open Edge and navigate to <http://portal.azure.com/> to connect to the Microsoft Azure Portal. Sign in with your subscription's credentials.
2. Create Managed Identity using PowerShell.

**i** A Managed Identity is automatically created when you create the Data Factory in the Azure Portal or with PowerShell. If you create the Data Factory from an ARM template (like we did in a previous lab), you will need to run a PowerShell command which will create the Managed Identity in the existing Data Factory.

- a. Open the Cloud Shell from the Azure Portal. Launching **Cloud Shell** and select **PowerShell**, it will ask you to create a storage account.



You have no storage mounted

Azure Cloud Shell requires an Azure file share to persist files. [Learn more](#)

This will create a new storage account for you and this will incur a small monthly cost. [View pricing](#)

\* Subscription

[Show advanced settings](#)

[Create storage](#)

[Close](#)

- b. Modify and run the following command:

```
[T] Set-AzDataFactoryV2 -ResourceGroupName resource_group_name -Name ADF_name -Location eastus
```

- c. You may get a warning that says the Data Factory already exists. If you do, just select Y since this will just create the Managed Identity for the Data Factory.

**i** You will not lose your Data Factory assets (pipelines, linked services, etc.).

```
PS /home/jean> Set-AzDataFactoryV2 -ResourceGroupName adf2011update -Name adf2011adf -Location eastus2
Confirm
A data factory with the name adf2011adf in the resource group adf2011update exists.
Continuing execution may overwrite the existing one.
Are you sure you want to continue?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): Y
```

Exercise 1 has been completed.

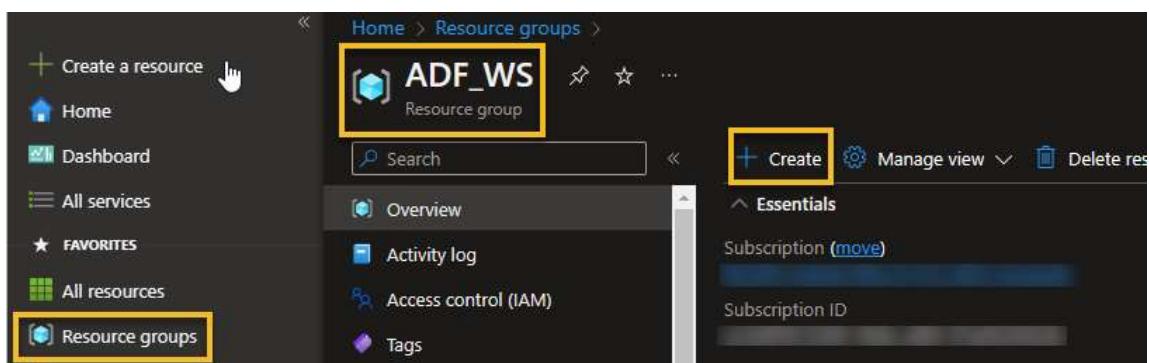
## Exercise 2: Store and Retrieve secrets in Azure Key Vault

This exercise shows how to configure Azure Data Factory to securely store secrets in Azure Key Vault. To configure the trust between the two services you will need to have completed **Module 01, Lesson 02, Lab 02**.

The method shown in this lab, using the SQL Database created in **Module 01, Lesson 02, Lab 02**, can be used with many other services where connection strings and credentials are required, such as Azure Storage, CosmosDB, etc.

### Tasks

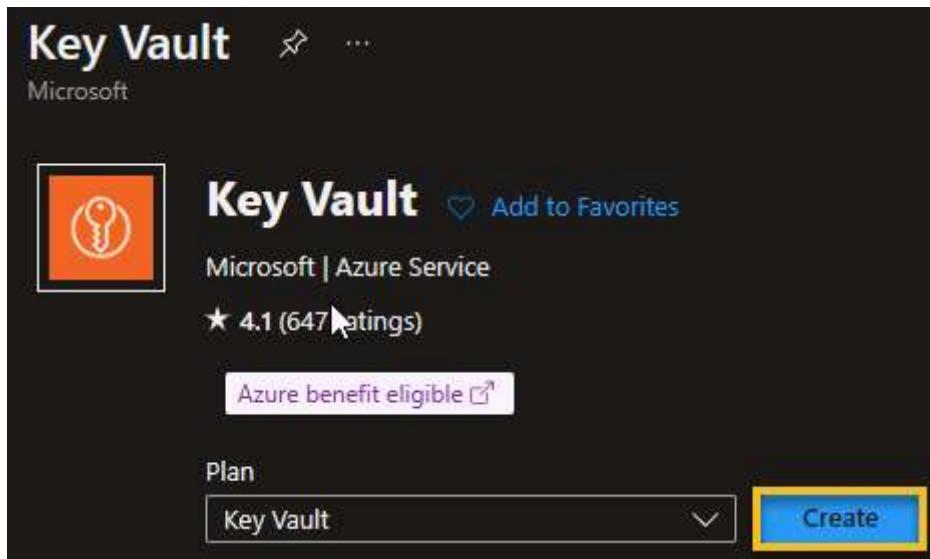
1. Create an Azure Key Vault resource.
  - a. Go to **Resource Groups**.
  - b. Go to your resource group.
  - c. Click on **+ Create** to create a new Azure Key Vault.



- d. Search for **Key Vault** and select the resource.



e. Click **Create**.



f. Confirm your resource group name, enter in unique Key Vault Name and click **Next**.

## Create a key vault ...

Basics Access configuration Networking Tags Review + create

Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance.

### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ME-MngEnv289593-chrey

Resource group \* rg-wus3-sctchbook

[Create new](#)

### Instance details

Key vault name \* kv-eus-adfwsp

Region \* East US

Pricing tier \* Standard

### Recovery options

Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or by Microsoft.

[Previous](#)

[Next](#)

[Review + create](#)

g. In **Permission Model** change radio to **Vault Access Policy**

## Create a key vault ...

Basics    **Access configuration**    Networking    Tags    Review + create

### Configure data plane access for this key vault

To access a key vault in data plane, all callers (users or applications) must have proper authentication and authorization.

#### Permission model

Grant data plane access by using a [Azure RBAC](#) or [Key Vault access policy](#)

- Azure role-based access control (recommended) ⓘ
- Vault access policy ⓘ

#### Resource access

- Azure Virtual Machines for deployment ⓘ
- Azure Resource Manager for template deployment ⓘ
- Azure Disk Encryption for volume encryption ⓘ

#### Access policies

Access policies enable you to have fine grained control over access to vault items. [Learn more](#)

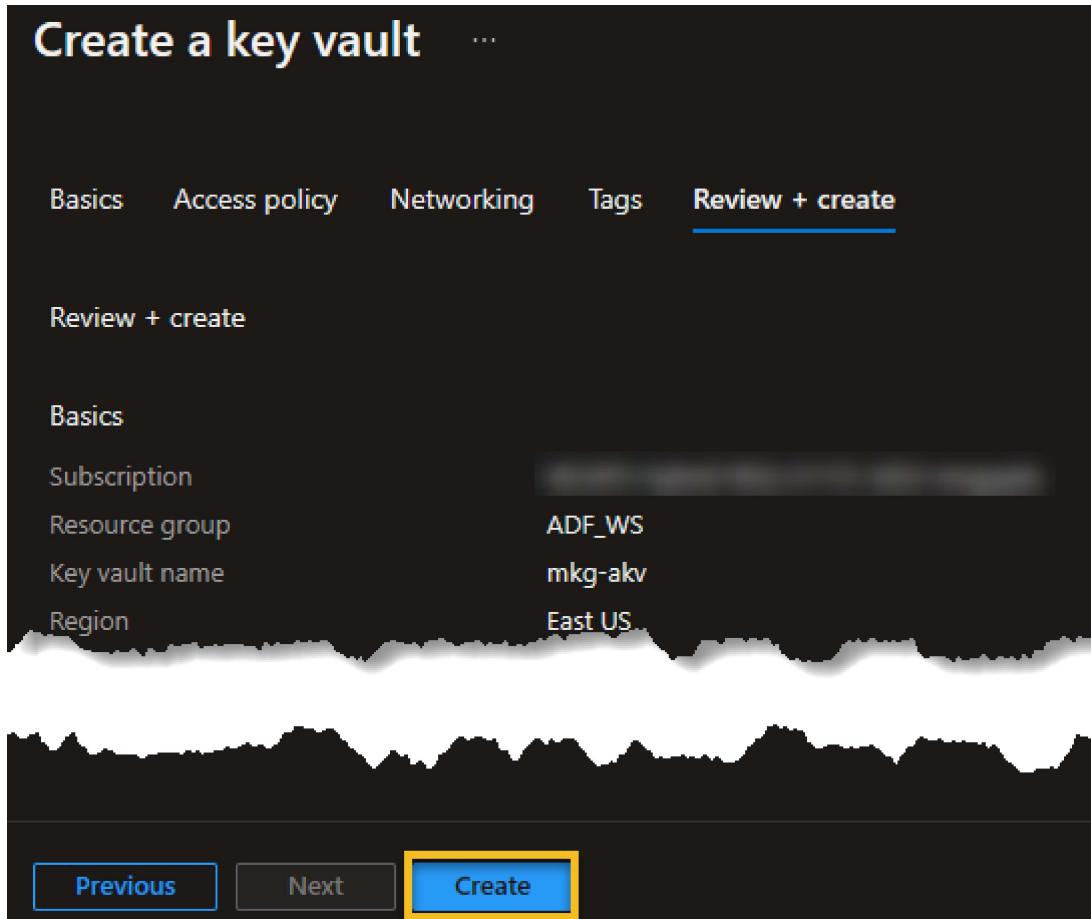
<a href="#">Create</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/> Name ↑	Email ↑	<a href="#">↑↓</a>
<input type="checkbox"/> c29b59dd-8203-473a-9bde-2e084bf92425		<a href="#">↑↓</a>

[Previous](#)

[Next](#)

**Review + create**

h. Click **Review and Create** and then **Create**



2. Once the Key Vault is provisioned, navigate to it in the Azure Portal and select Settings > **Secrets**.
3. Create a new secret to save SQL Server connection strings.

- a. Select **+Generate/Import** to create a new secret.

The screenshot shows the 'adf-tutorial - Secrets' blade in the Azure portal. The left sidebar has options: Keys, **Secrets** (which is selected and highlighted with a green box), Certificates, and Access policies. The main area shows a table with columns NAME and TYPE. The message 'There are no secrets available.' is displayed.

- b. Ensure that Upload options is set to **Manual**.
- c. Give the secret a meaningful name (for example: WSPlusMod6SqlConnectionString). Use the following text as the **Value** after you modify the parameter values to reference your Azure SQL Server, Azure SQL Database, User ID and Password.

`T Server=tcp:<serverName>.database.windows.net; Database=<dbName>; User ID=<loginName>; Password=<password>; Trusted_Connection=False; Encrypt=True;`

- Paste the above text to notepad and update the values in the <> brackets (remove the brackets) before copying to ADF.

d. Click the **Create** button.

## Create a secret

Upload options

Name \*  ✓

Secret value \*  ✓

Content type (optional)

Set activation date (i)

Set expiration date (i)

Enabled Yes No

Tags 0 tags

Create

4. Grant the Azure Data Factory access to Azure Key Vault.

a. In the Azure Key Vault resource. Select Access Policies and click on + **Create**.

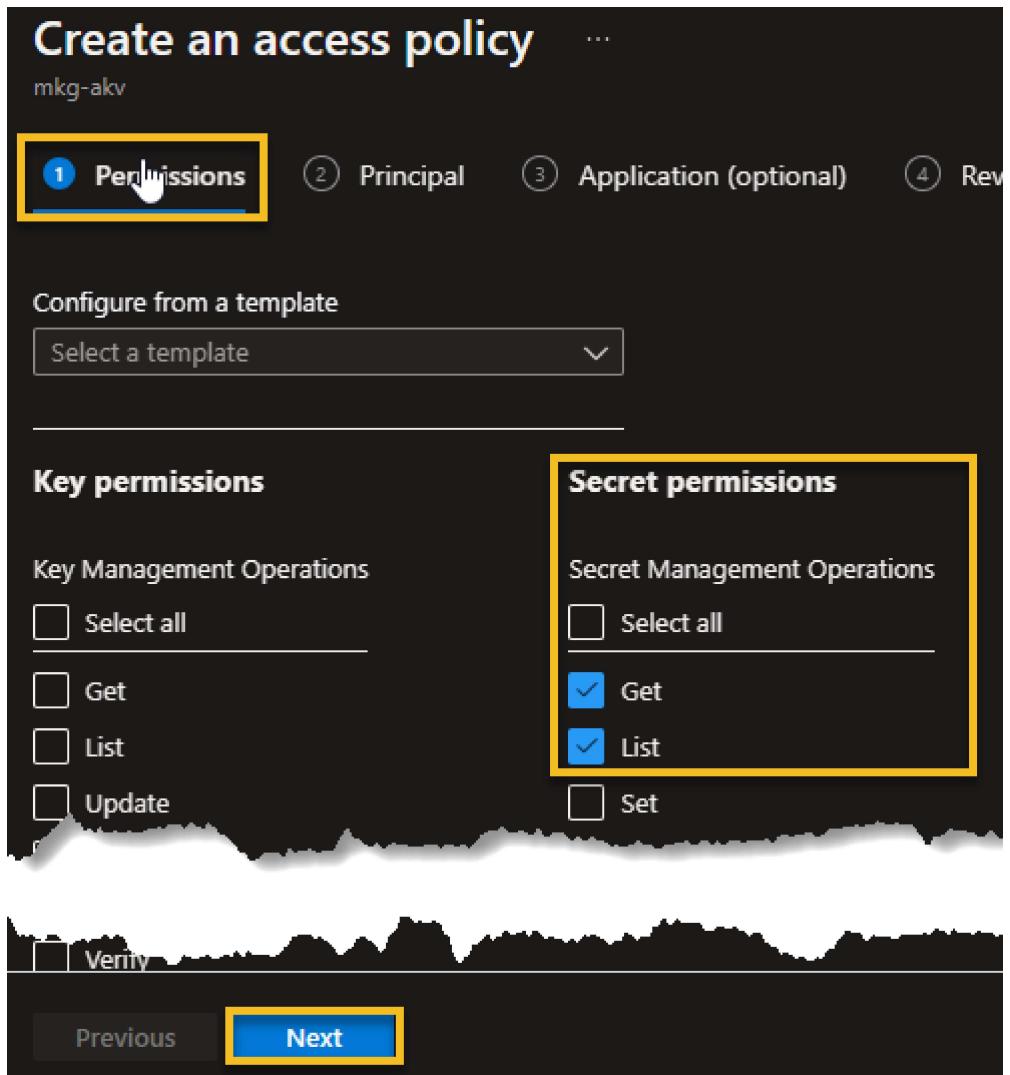
The screenshot shows the 'mkg-akv | Access policies' page in the Azure portal. On the left, a sidebar lists several options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies (which is selected and highlighted with a yellow box), and Events. At the top right, there are 'Create', 'Refresh', 'Delete', and 'Edit' buttons. A yellow box highlights the 'Create' button. Below the buttons, a message states: 'Access policies enable you to have fine grained control over access to your vault'. A search bar and a 'Permissions : All' filter are present. The main area displays a table with one record:

	Name	Email
<input type="checkbox"/>	USER	[REDACTED]

- b. Azure Key Vault allows you define permissions on each of the object types that can be saved in the Key Vault. Keys, Secrets, and Certificates. We only created a Secret. Therefore, we will grant the Azure Data Factory **Managed Identity** account permission to read the contents of the secret. From **Permission** list, select **Get** and **List**. Click **Next** to select the principal account.

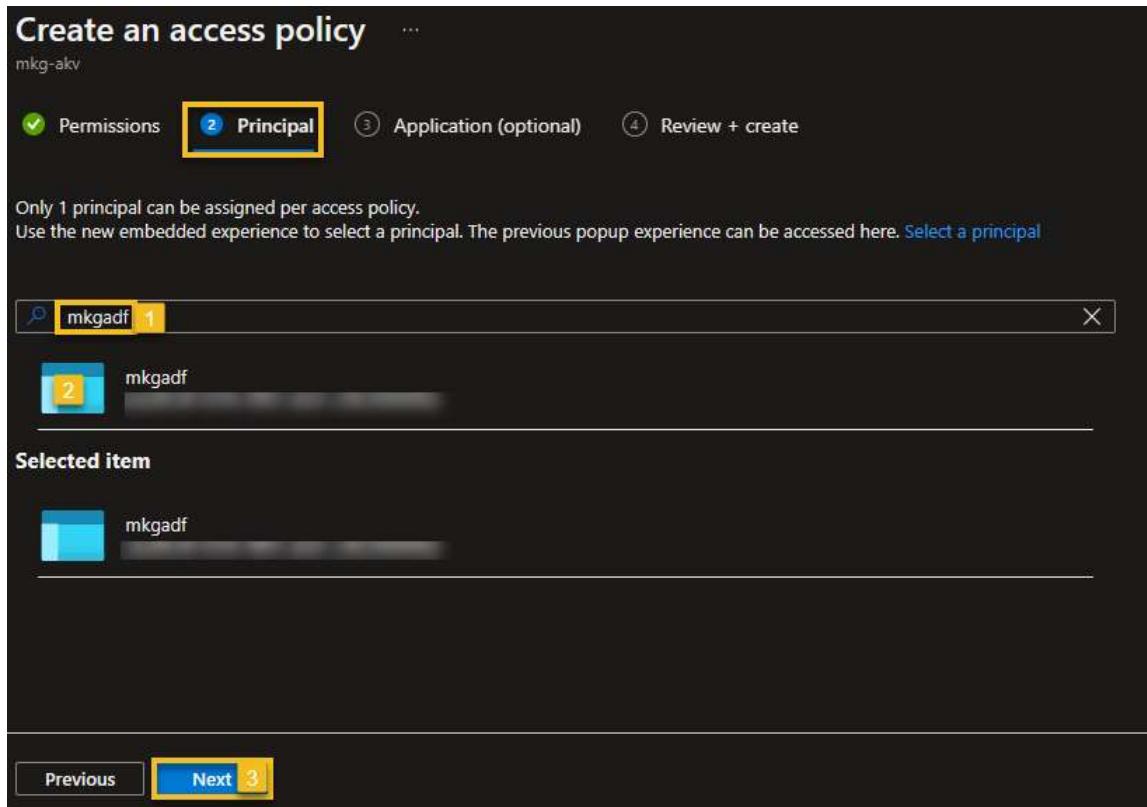
In production you will only grant **Get** permission (read). You will not expose all the secrets in the key vault with **List**.

**i** Do not use a template option, as it grants more permissions then needed.



c. Search for your Azure Data Factory name, select and click **Next**.

- If your ADF does not show up in the list, then the managed identity account is missing. Review [Exercise 1](#).



- d. Click **Next** on Step 3.
- e. Click **Create** on Step 4.
- f. You should see your ADF listed as an application after creation.

The screenshot shows the 'mkg-akv | Access policies' page. The left sidebar includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Access policies' (which is selected and highlighted with a yellow box), and 'Events'. The main area displays a table of access policies. The table has columns: 'Name' (sorted by name), 'Email' (sorted by email), 'Key Permissions', and 'Secret Permissions'. One record is shown: 'mkgadf' under the 'APPLICATION' category, with 'Get, List' permissions. A yellow box highlights the entire row for 'mkgadf'.

5. Create a Key Vault Linked Services pipeline
- a. In your Data Factory portal, select the **Manage/Toolbox** link.

The screenshot shows the Microsoft Azure Data Factory interface. At the top, it says "Microsoft Azure | Data Factory". Below that, there's a navigation bar with icons for Home, Create, and Refresh, followed by "Azure DevOps GIT". The main area is titled "Factory Resources" and contains a search bar with the placeholder "Filter resources by name". There are three main categories listed: "Pipelines", "Datasets", and "Data flows".

b. Choose **+New** under the Linked Services tab.

This screenshot shows the "Linked services" tab in the Azure portal. On the left, there's a sidebar with "Connections" and two options: "Linked services" (which is selected and highlighted in grey) and "Integration runtimes". On the right, the main area is titled "Linked services" and contains the sub-instruction: "Linked service defines the connection information to a data store or compute". Below this, there's a blue "+ New" button.

c. Select the **Azure Key Vault** linked service type and **Continue**.

# New linked service

Data store

Compute



All

Azure

Database



Azure Key Vault

- d. Give the linked service a suitable name and locate the Key Vault within your Azure subscription.
  - e. Select **Test Connection** and make sure the connection succeeds. If it doesn't succeed then you need to check the Key Vault access policies for the Service Identity.
  - f. Click **Create**.
6. Reference secrets stored in key vault

- a. We will use an Azure SQL Database to demonstrate this functionality. You should already have an Azure SQL Database from earlier labs and the pre-requisites, if not go to the Azure Portal and provision a database.
- b. From Linked services select **+New** and create a new connection to the **Azure SQL Database**.

New linked service

Data store   Compute

azure sql

All   Azure   Database   File   Generic protocol   NoSQL   Services and apps

The screenshot shows the 'New linked service' interface. At the top, there are tabs for 'Data store' and 'Compute'. Below that is a search bar with the text 'azure sql'. Underneath the search bar are navigation tabs: 'All', 'Azure', 'Database', 'File', 'Generic protocol', 'NoSQL', and 'Services and apps'. The 'Database' tab is selected. Two options are displayed in boxes: 'Azure SQL Database' (represented by a blue cylinder icon) and 'Azure SQL Database Managed Instance' (represented by a server tower icon). Both options have their names below them.

- c. Give the connection a meaningful name.
- d. Change the selection from Connection String to **Azure Key Vault**.

New linked service (Azure SQL Database)

Name \*

AzureSqlDB

Description

Connect via integration runtime \* ⓘ

AutoResolveIntegrationRuntime

Connection string    Azure Key Vault

The screenshot shows the 'New linked service (Azure SQL Database)' configuration page. It has fields for 'Name' (set to 'AzureSqlDB'), 'Description' (empty), and 'Connect via integration runtime' (set to 'AutoResolveIntegrationRuntime'). At the bottom, there are two radio buttons: 'Connection string' (grayed out) and 'Azure Key Vault' (selected, indicated by a blue background).

- e. Select the **AKV** linked service from the drop-down below.
- f. Enter the name of the secret which you created earlier in this lesson (for example: WSPlusMod6SqlConnString).

- g. Optionally enter the version you wish to use; best practice is to leave blank, which uses the latest version. This allows Data Factory to always use the most recent version of the secret without any changes needing to be made.

New linked service (Azure SQL Database)

Name \*

Description

Connect via integration runtime \*

AKV linked service \*

Secret name \*

Secret version

Authentication type \*

Authentication reference method  Inline  Credential (Preview)

Managed identity name:   
Managed identity object ID:   
Grant Data Factory service managed identity access to your Azure SQL Database.  
[Learn more](#)

Always encrypted

Annotations

Parameters

Advanced

- h. Select **Test connection** next to the Cancel button above and make sure that the connection succeeds. It may take a little longer to connect as the connection string needs to be retrieved from the Key Vault.

- i. Then click on the **Create** button.

Congratulations you have now configured Data Factory to retrieve secrets from Key Vault. From here you could configure a Dataset and Pipeline to use the connection. With the test connection success, we have demonstrated the configuration to satisfy the purpose of this lab.

Exercise 2 has been completed.

