Azure Data Services in a Day

Implement a Modern Data Platform Architecture

Table of Contents

Authors

Fabio Braga
Cloud Solution Architect – Data Platform
fabio.braga@microsofty.com

Rod Colledge
Cloud Solution Architect – Data Platform
rodcol@microsoft.com

Version History

Version	Date	Description	Author
0.1	23/04/2019	Lab Complete. Ready for Review.	Fabio Braga
0.2	25/04/2019	Review	Rod Colledge
0.3	06/05/2019	Added Lab Architecture diagram to each Lab. Fixes from Rod's review. Minor formatting adjustments.	Fabio Braga
0.4	07/05/2019	Fixed Lab2 inconsistencies and added new screenshots.	Fabio Braga

Overview

Today you will learn about the main concepts related to advanced analytics and Big Data processing and how Azure Data Services can be used to implement a modern data warehouse architecture. You will understand what Azure services you can leverage to establish a solid data platform to quickly ingest, process and visualise data from a large variety of data sources. The reference architecture you will build as part of this exercise has been proven to give you the flexibility and scalability to grow and handle large volumes of data and keep an optimal level of performance.

In the exercises in this lab you will build data pipelines using data related to New York City. The workshop was designed to progressively implement an extended modern data platform architecture starting from a traditional relational data pipeline. Then we introduce big data scenarios with large files and distributed computing. We add non-structured data and AI into the mix and finish with real-time streaming analytics. You will have done all of that by the end of the day.

Document Structure

This document contains detailed step-by-step instructions on how to implement a Modern Data Platform architecture using Azure Data Services. It's recommended you carefully read the detailed description contained in this document for a successful experience with all Azure services.

You will see the label **IMPORTANT** whenever a there is a critical step to the lab. Please pay close attention to the instructions given.

Data Source References

New York City data used in this lab was obtained from the New York City Open Data website: https://opendata.cityofnewyork.us/. The following datasets were used:

- NYPD Motor Vehicle Collisions: https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95
- TLC Yellow Taxi Trip Data: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Lab Prerequisites

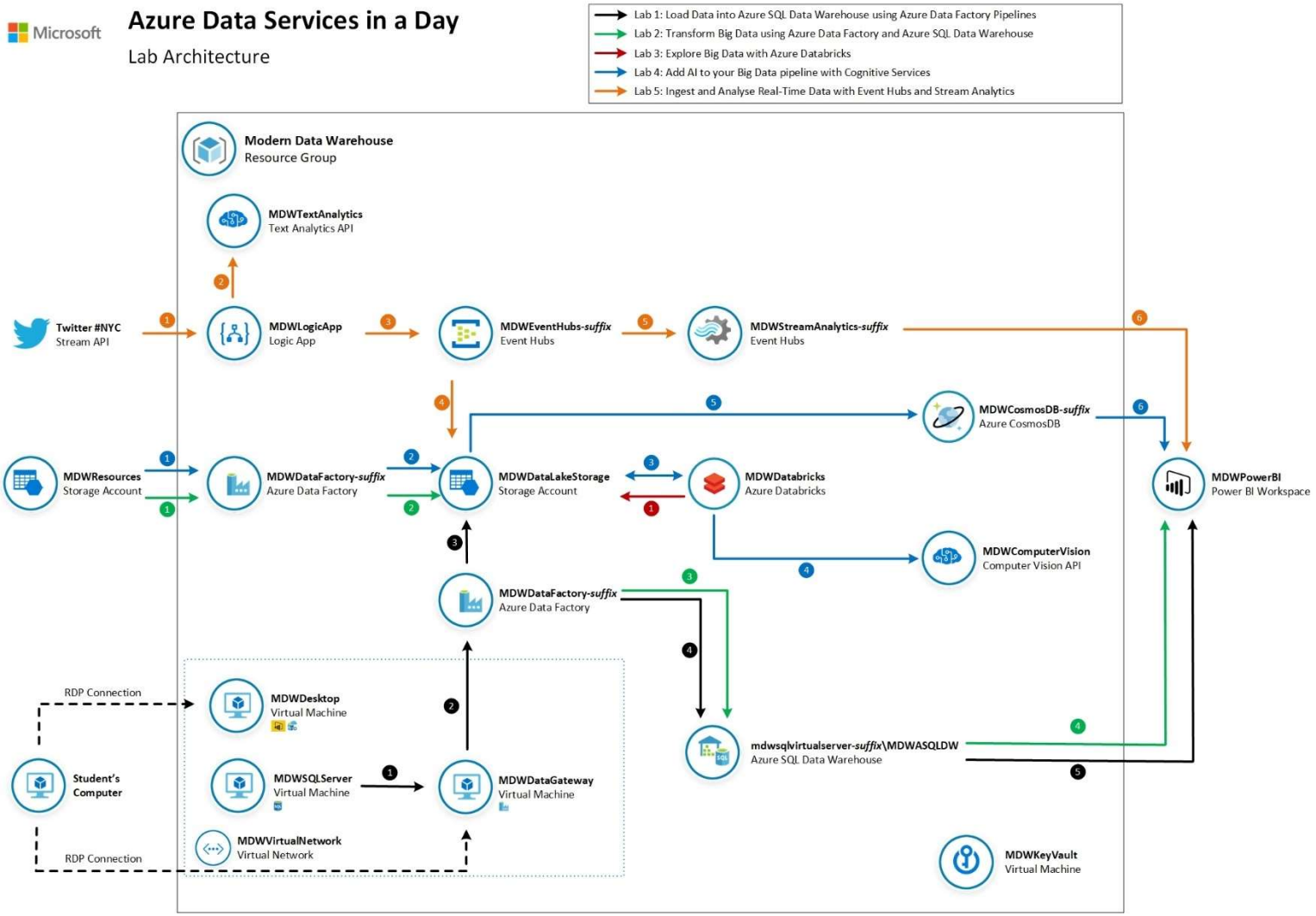
The following prerequisites must be completed before you start these labs:

- You must be connected to the internet;
- You must have an Azure account with administrator-level access to your subscription. If you don't have an account, you can sign up for free following the instructions here: https://azure.microsoft.com/en-au/free/
- Download Lab files from <insert GitHub link here> and save them in the local folder C:\ADSIAD\LabFiles;
- Lab 5 requires you to have a Twitter account. If you don't have an account you can sign up for free following the instructions here: https://twitter.com/signup.
- Lab 5 requires you to have a Power BI Pro account. If you don't have an account you can sign up for a 60-day trial for free here: https://powerbi.microsoft.com/en-us/power-bi-pro/
- You must deploy all Azure services required in each lab by running a PowerShell script as administrator on your local computer. We recommend you follow the instructions defined in ***Lab 0 – Provision Azure Resources*** for an automated deployment through ARM Templates and PowerShell. You can also deploy the resources manually through the Azure Portal by following the instructions in ***Appendix 1 – Manually Provision Azure Resources***.
- The ARM template used to automate the deployment of Azure services attempts to deploy all services required in the resource group default region. See more information on Lab 0.

IMPORTANT: A collection of Azure resources will be provisioned in your subscription and they will incur billing costs. The estimated consumption cost for the execution of the labs included in this workshop is around US $150.00.

Lab Guide

Throughout a series of 5 labs we will progressively implement the architecture referenced below:



Lab 0: Provision Azure Resources

In this lab you will automatically provision all Azure resources required to complete labs 1 though to 5. You will use a pre-defined ARM template with the definition of all Azure services used to ingest, store, process and visualise data. The estimated time to complete this lab is: 30 minutes.

Lab 1: Load Data into Azure SQL Data Warehouse using Azure Data Factory Pipelines

In this lab you will configure the Azure environment to allow relational data to be transferred from a SQL Server 2017 database to an Azure SQL Data Warehouse database using Azure Data Factory. The dataset you will use contains data about motor vehicle collisions that happened in New York City from 2012 to 2019. You will use Power BI to visualise collision data loaded from Azure SQL Data Warehouse.

1. Restore SQL Server backup from Azure Storage and Configure Azure Data Factory Self-Hosted Integration Runtime
2. Build an Azure Data Factory Pipeline to copy data from a SQL Server table
3. Use Azure Storage as a staging area for Polybase
4. Load data to a Azure SQL Data Warehouse table using Polybase
5. Visualize data from Azure SQL Data Warehouse using Power BI

Lab 2: Transform Big Data using Azure Data Factory and Azure SQL Data Warehouse

In this lab you will use Azure Data Factory to download large data files to your data lake and use Azure SQL Data Warehouse stored procedure to generate a summary dataset and store it in the final table. The dataset you will use contains detailed New York City Yellow Taxi rides for 2018. You will generate a daily aggregated summary of all rides and save the result in your data warehouse. You will then use Power BI to visualise summarised data. The estimated time to complete this lab is: xx minutes.

1. Build an Azure Data Factory Pipeline to copy big data files from shared Azure Storage
2. Save data files to your data lake
3. Use Polybase to load data into staging tables in your Azure SQL Data Warehouse.
   Call a Stored Procedure to perform data aggregations and save results in the final table.
4. Visualize data from your Azure SQL Data Warehouse using Power BI

Lab 3: Explore Big Data using Azure Databricks

In this lab you will use Azure Databricks to explore the New York Taxi data files you saved in your data lake in Lab 2. Using a Databricks notebook you will connect to the data lake and query taxi ride details. The estimated time to complete this lab is: xx minutes.

1. Build an Azure Databricks notebook to explore the data files you saved in your data lake in the previous exercise. You will use Python and SQL commands to open a connection to your data lake and query data from data files.

Lab 4: Add AI to your Big Data Pipeline with Cognitive Services

In this lab you will use Azure Data Factory to download New York City images to your data lake. Then, as part of the same pipeline, you are going to use an Azure Databricks notebook to invoke Computer Vision Cognitive Service to generate metadata documents and save them in back in your data lake. The Azure Data Factory pipeline then finishes by saving all metadata information in a Cosmos DB collection. You will use Power BI to visualise NYC images and their AI-generated metadata. The estimated time to complete this lab is: xx minutes.

1. Build an Azure Data Factory Pipeline to copy image files from shared Azure Storage
2. Save image files to your data lake
3. For each image in your data lake, invoke an Azure Databricks notebook that will take the image URL as parameter
4. For each image call the Azure Computer Vision Cognitive service to generate image metadata. Metadata files are saved back in your data lake
5. Copy metadata JSON documents into your Cosmos DB database
6. Visualize images and associated metadata using Power BI

Lab 5: Ingest and Analyse real-time data with Event Hubs and Stream Analytics

In this lab you will use an Azure Logic App to connect to Twitter and generate a stream of messages using the hashtag #NYC. The logic app will invoke the Azure Text Analytics Cognitive service to score Tweet sentiment and send the messages to Event Hubs. You will use Stream Analytics to generate the average Tweet sentiment in the last 60 seconds and send the results to a real-time dataset in Power BI. The estimated time to complete this lab is: xx minutes.

1. Build an Azure Logic App to invoke the Twitter API and retrieve Tweets with the hashtag #NYC

2. For each Tweet, invoke the Azure Text Analytics Cognitive service to detect its sentiment score
3. Format and send the Tweet's JSON message to Event Hubs
4. Save Tweet messages into your data lake for future analysis (cold path)
5. Send stream of Tweet messages to Stream Analytics for real-time analytics (hot path)
6. Visualize real-time data generated by Stream Analytics with Power BI

Lab 0: Provision Azure Resources

In this lab you will automatically provision all Azure resources required to complete labs 1 though to 5. We will use a pre-defined ARM template with the definition of all Azure services used to ingest, store, process and visualise data. The estimated time to complete this lab is: 30 minutes.

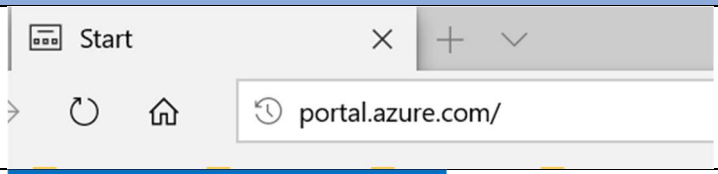
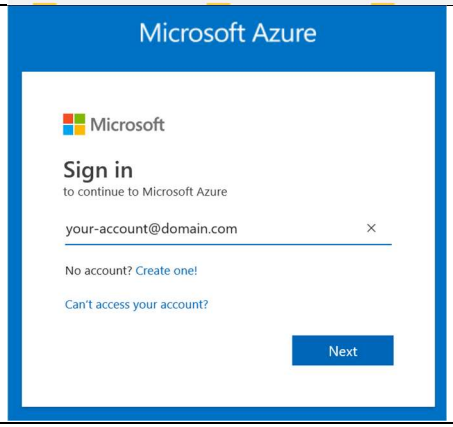
Download Lab Files from GitHub

In this section you will download all files used in this lab to deploy Azure Data Services and all other files used in labs 1 through to 5.

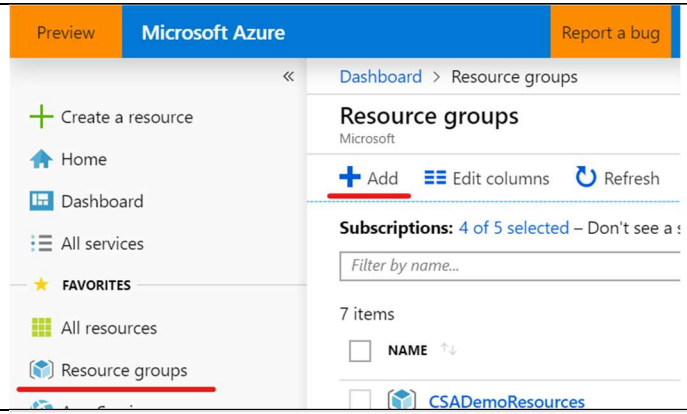
Add content here!

Prepare your Azure subscription

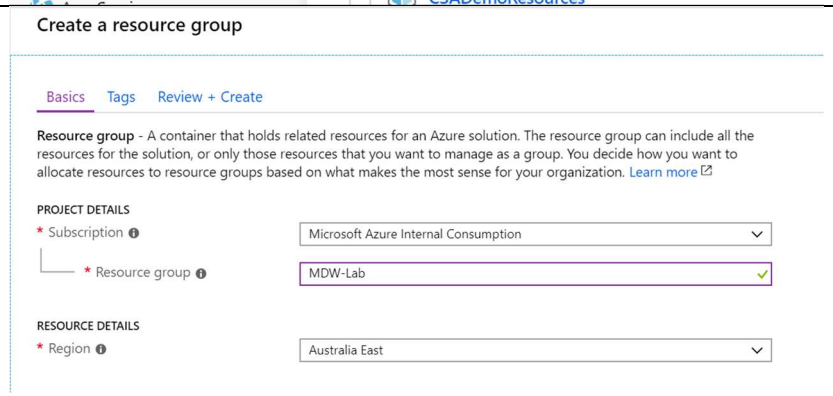
In this section you will use the Azure Portal to create a Resource Group that will host the Azure Data Services used in labs 1 through to 5.

IMPORTANT: Execute these steps on your host computer	
1. Open the browser and navigate to https://portal.azure.com	
2. Log on to Azure using your account credentials	

3. Once you have successfully logged on, locate the Favourites menu on the left-hand side panel and click the **Resource groups** item to open the Resource groups blade.
4. On the **Resource groups** blade, click the "**+ Add**" button to create a new resource group.



5. On the **Create a resource group** blade, select your subscription in **Subscription** drop down list.
6. In the **Resource group** text box enter "MDW-Lab"
7. **IMPORTANT**: The name of the resource group chosen is ***not*** relevant to the successful completion of the labs. If you choose to use a different name, then please proceed with the rest of the lab using your unique name for the resource group.



IMPORTANT

The ARM template you will use to deploy the lab components uses the Resource Group region as the default region for all services. To avoid deployment error when services are not available in the region selected, please use one of the recommended regions.

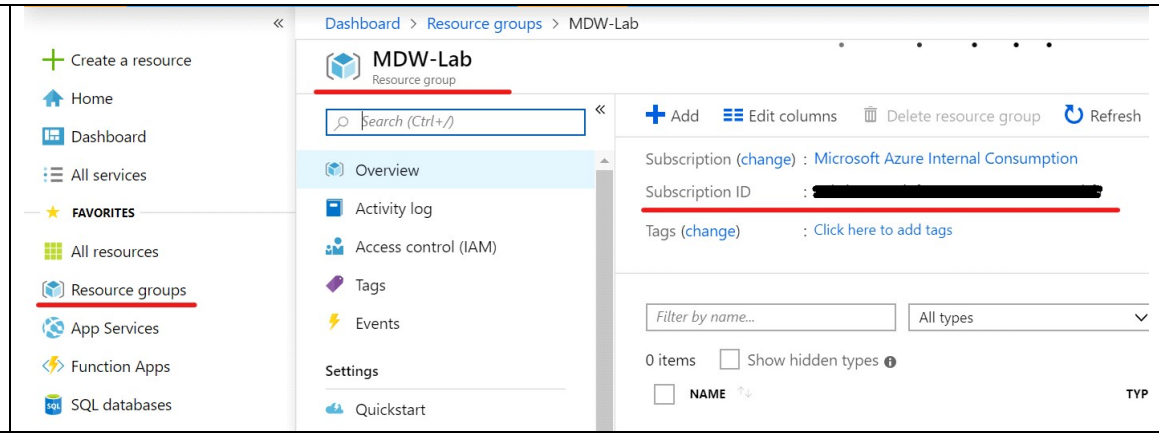
8. In the **Region** drop down list, select one of the regions from the list.

Recommended regions:

- US East
- US East 2
- US Central
- US West 2
- Europe West
- Asia Southeast
- Europe North
- US South Central
- US West
- UK South

	• Australia East • Canada Central • Japan East • Germany Central • France Central • India Central • Brazil South • Korea Central • Korea South
9. Proceed to create the resource group by clicking **Review + Create,** and then **Create**	

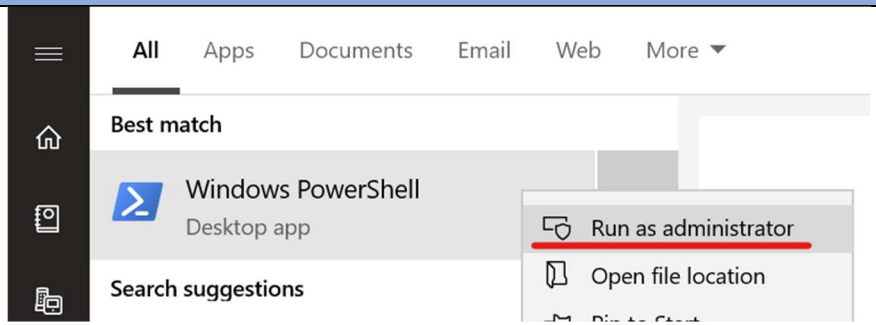
10. Once the MDW-Lab resource group has been created, navigate again to the **Resource groups** menu under **Favourites.**
11. On the **Resource groups** blade, click the MDW-Lab resource group.
12. On the **MDW-Lab Resource Group** blade, take note of your **Subscription ID.** You will need it in the next step to deploy the ARM template.



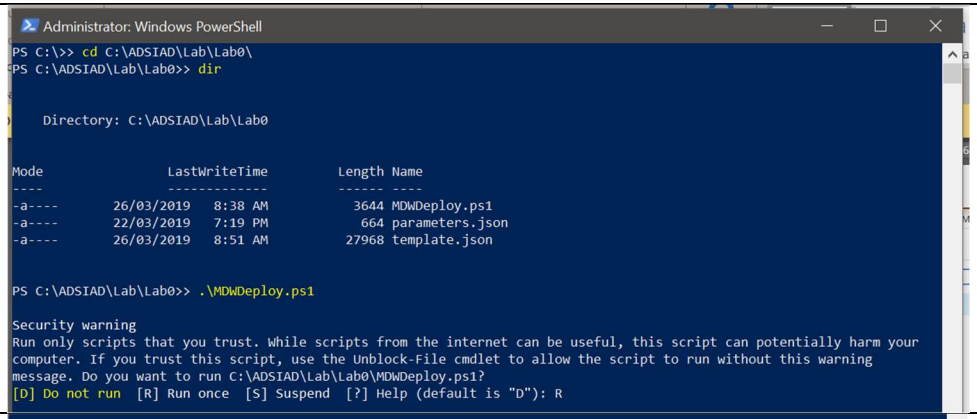
Deploy Azure Services

In this section you will use PowerShell scripts and ARM templates to automate the deployment of all Azure Data Services used in labs 1 through to 5.

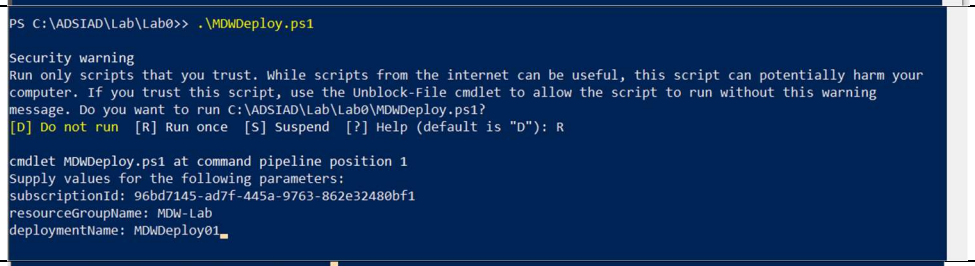
1. **IMPORTANT**: You must be a member of the local administrators group on the Student's computer.
2. Click the Windows menu logo and start typing **PowerShell until it appears, and then** right-click it and select **Run as administrator.**
3. Answer **Yes** to the question about allowing this app to make changes to your device.



4. Navigate to the C:\ADSIAD\Lab\Lab0 folder.
5. Execute the script MDWDeploy.ps1
6. Answer **R** to the confirm the execution of the deployment script.

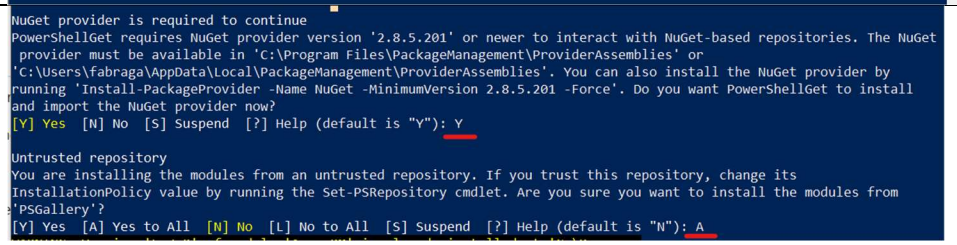


7. Paste the **Subscription ID** you retrieved from the previous step in the **subscriptionId** field.
8. Enter “MDW-Lab” in the **resourceGroupName** field.
9. Enter “MDWDeploy01” in the **deploymentName** field.

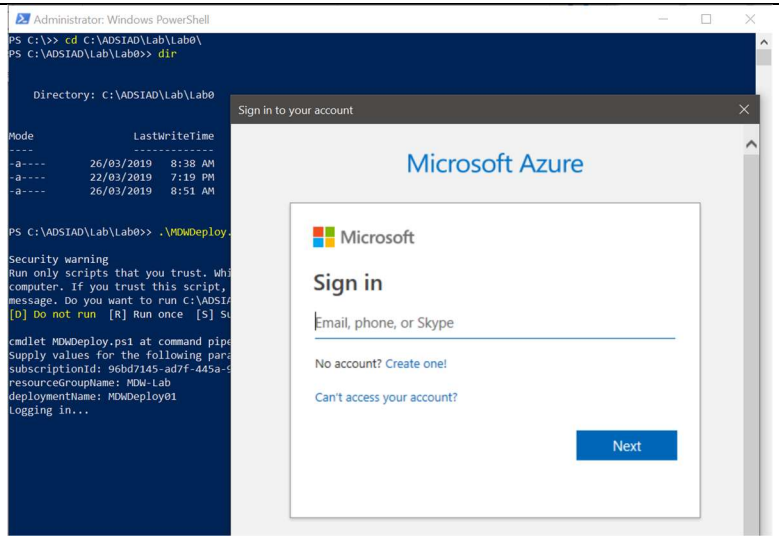


IMPORTANT: PowerShell AzureRm module is going to be installed as part of the deployment. Please find more info here: https://docs.microsoft.com/en-us/powershell/azure/install-az-ps

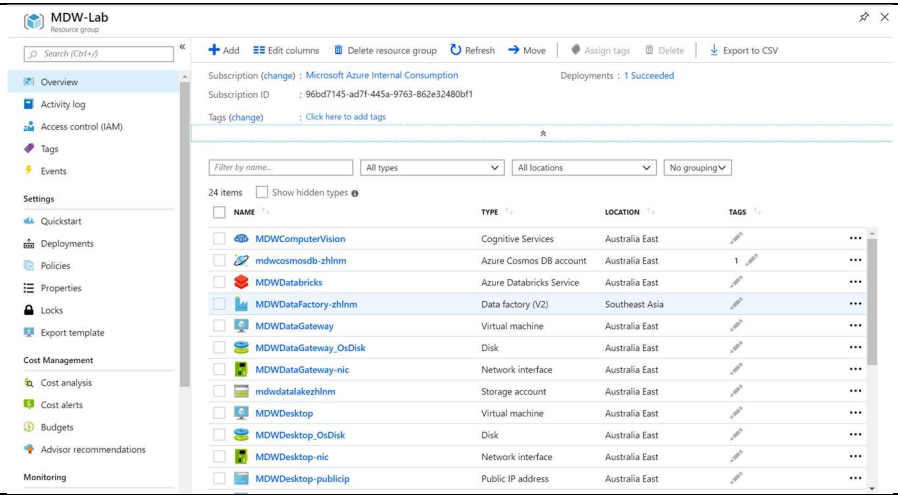
10. Answer **Y** to confirm the installation of the NuGet provider.
11. Answer **A** to proceed with the installation.



12. Log on to Azure using your credentials.
13. Once credentials are validated the script executions should last for around 15 mins.



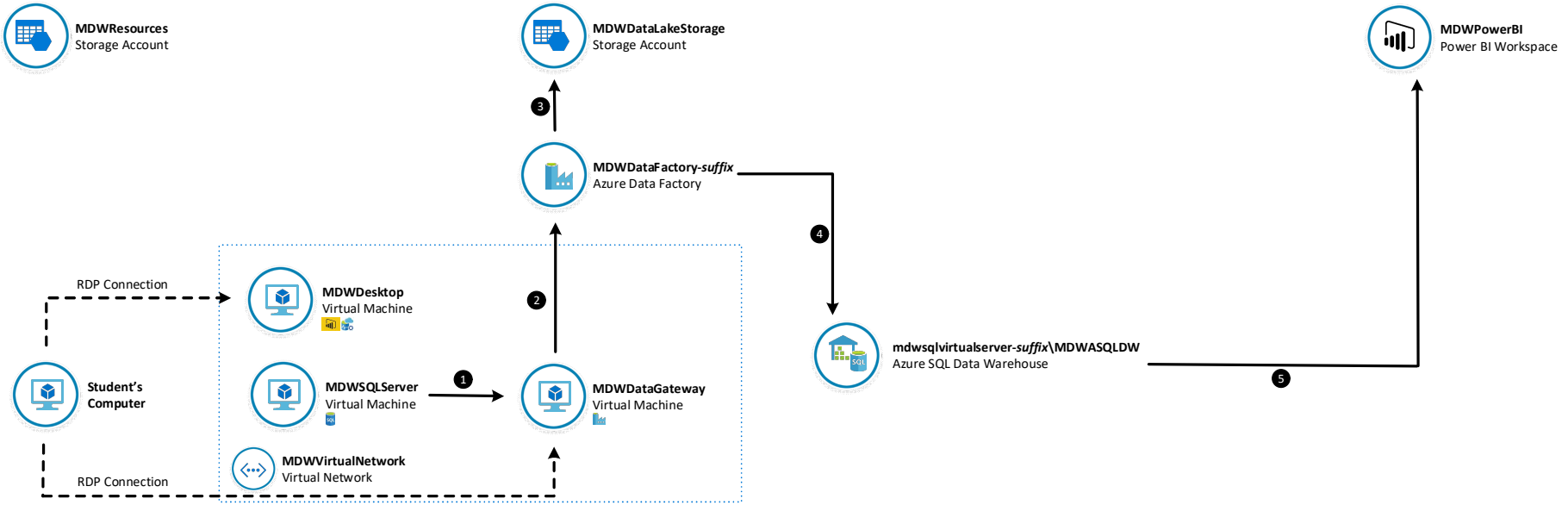
14. Once the deployment script finishes successfully, you should be able to see 25 new resources deployed in the MDW-Lab resource group.
15. **IMPORTANT:** Note that some resources had a "*-suffix*" appended to their names to guarantee uniqueness. Take note of the suffix generated as it will be used very frequently during the lab.



Lab 1: Load Data into Azure SQL Data Warehouse using Azure Data Factory Pipelines

In this lab you will configure the Azure environment to allow relational data to be transferred from a SQL Server 2017 database to an Azure SQL Data Warehouse database using Azure Data Factory. The dataset you will use contains data about motor vehicle collisions that happened in New Your City from 2012 to 2019. You will use Power BI to visualise collision data loaded from Azure SQL Data Warehouse.

Lab Architecture



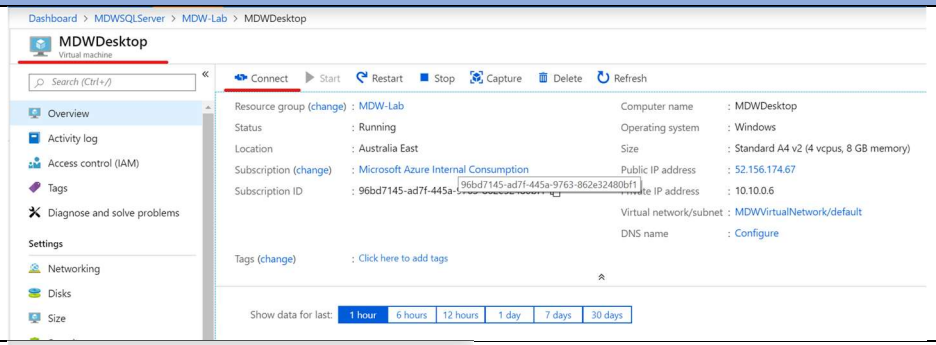
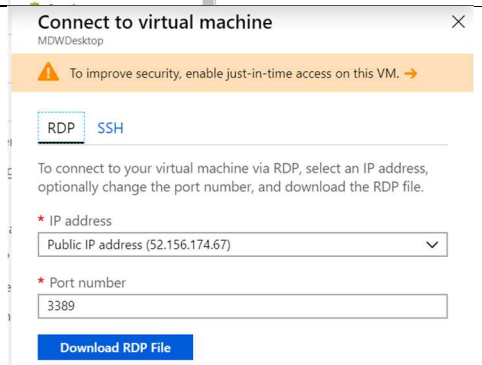
1. Restore SQL Server backup from Azure Storage and Configure Azure Data Factory Self-Hosted Integration Runtime
2. Build an Azure Data Factory Pipeline to copy data from a SQL Server table
3. Use Azure Storage as a staging area for Polybase
4. Load data to an Azure SQL Data Warehouse table using Polybase
5. Visualize data from Azure SQL Data Warehouse using Power BI

IMPORTANT: Some of the Azure services provisioned by Lab0 require globally unique name and a "*-suffix*" has been added to their names to ensure this uniqueness. Please take note of the suffix generated as you will need it for the following resources:

Name	Type
mdwcosmosdb-*suffix*	Cosmos DB account
MDWDataFactory-*suffix*	Data Factory (V2)
mdwdatalake*suffix*	Storage Account
MDWEventHubs-*suffix*	Event Hubs Namespace
MDWKeyVault-*suffix*	Key vault
mdwsqlvirtualserver-*suffix*	SQL server
MDWStreamAnalytics-suffix	Stream Analytics job

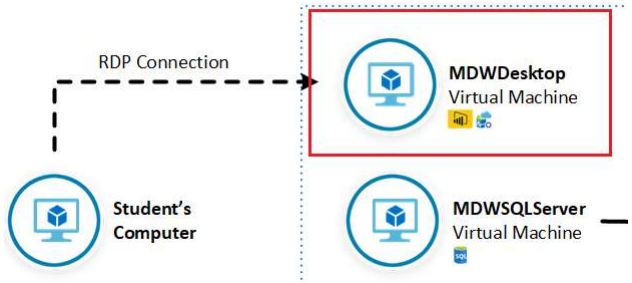
Connect to MDWDesktop

In this section you are going to establish a Remote Desktop Connection to MDWDesktop virtual machine.

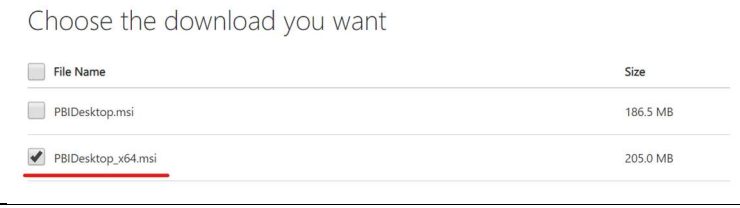
IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, navigate to the **MDW-Lab** resource group and click the **MDWDesktop** virtual machine. 2. On the **MDWDesktop** blade, from the **Overview** menu, click the **Connect** button.	
3. On the **Connect to virtual machine** blade, click **Download RDP File**. This will download a .rdp file that you can use to establish a Remote Desktop Connection with the virtual machine.	

Install required software onto MDWDesktop

In this section you are going to install Power BI Desktop and Azure Data Studio on MDWDesktop.

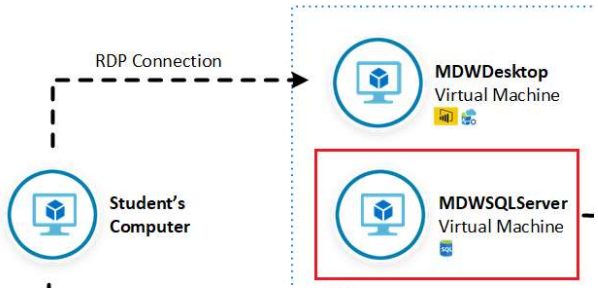


IMPORTANT: Execute these steps inside the MDWDesktop remote desktop connection	
1. Once the file is downloaded, click on file to establish the RDP connection with MDWDesktop 2. User the following credentials to authenticate: a. User Name: MDWAdmin b. Password: P@ssw0rd123! 3. Once logged in, accept the default privacy settings.	
4. Using the browser, download and install the latest version of following software. During the setup, accept all default settings.	**Azure Data Studio (User Installer)** https://docs.microsoft.com/en-us/sql/azure-data-studio/download Download and install Azure Data Studio 04/19/2019 • 4 minutes to read • Contributors all Azure Data Studio runs on Windows, macOS, and Linux. Download and install the latest release, the *April Release*: ⓘ Note If you're updating from SQL Operations Studio and want to keep your settings, keyboard shortcuts, or code snippets, see Move user settings. Platform / Download / Release date / Version Windows / User Installer (recommended) System Installer .zip / April 18, 2019 / 1.6.0 **Power BI Desktop (64-bit)**

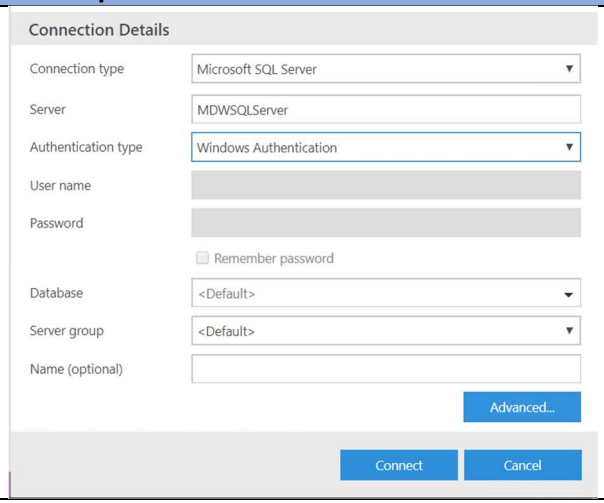
	https://www.microsoft.com/en-us/download/details.aspx?id=45331 

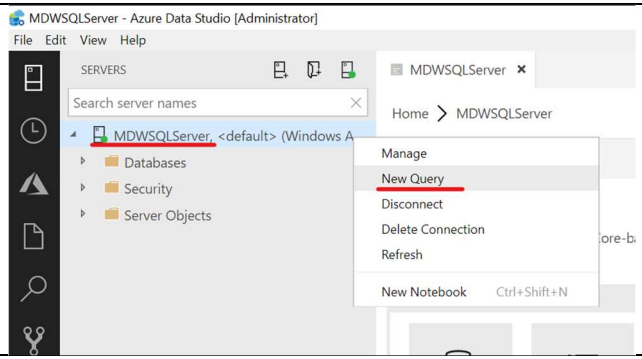
Restore NYCDataSets database onto MDWSQLServer

In this section you are going to connect to MDWSQLServer to restore the NYCDataSets database from backup stored in an Azure Storage Account.



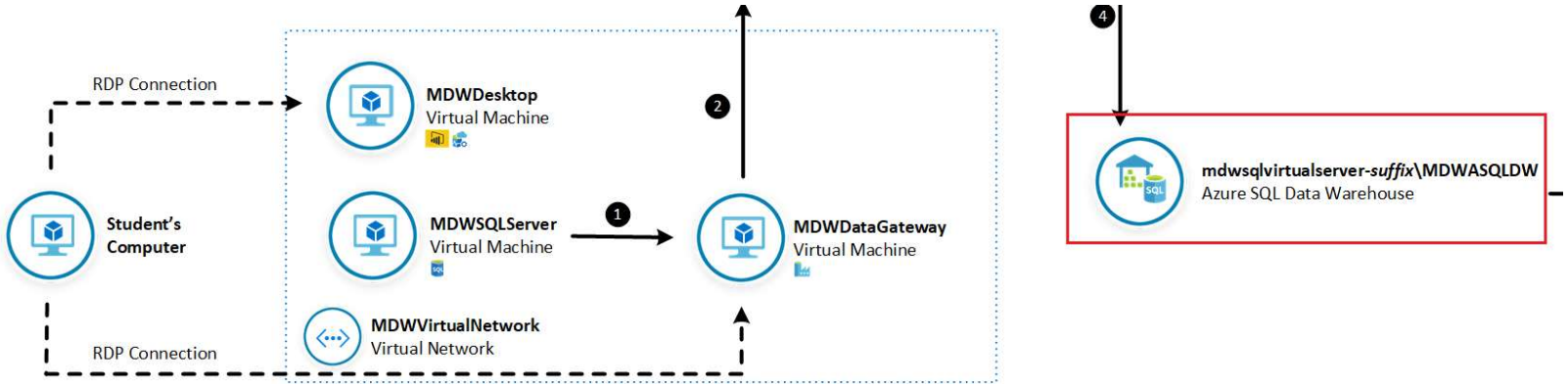
IMPORTANT: The full solution script is saved in the Student's Computer path **C:\ADSIAD\Lab\Lab1\Solution\Restore NYCDataSets.sql**.

IMPORTANT: Execute these steps inside the MDWDesktop remote desktop connection	
1. Open Azure Data Studio and establish a new connection to **MDWSQLServer** using Windows Authentication	

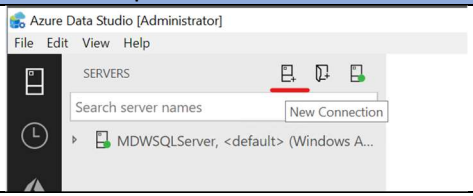
2. Press **Ctrl+G** to expand the **Servers** panel 3. Right-click the **MDWSQLServer** server name on the **SERVERS** panel and select **New Query**	
4. On the **Query Editor** window, create a new credential named [https://mdwresources.blob.core.windows.net/nycdatasets] using a Shared Access Signature (SAS). See SQL command:	**SQL command:** `create credential [https://mdwresources.blob.core.windows.net/nycdatasets] with identity = 'SHARED ACCESS SIGNATURE', secret = 'sv=2018-03-28&ss=b&srt=sco&sp=rwl&se=2050-12-30T17:25:52Z&st=2019-04-05T09:25:52Z&spr=https&sig=4qrD8NmhaSmRFu2gKja67ayohfIDEQH3LdVMa2Utykc%3D' go`
5. Restore the **NYCDataSets** database from the backup file stored in the Azure Storage Account. The backup file name is NYCDataSets.Full.bak. The restore command should move the data file to the 'F:\Data' folder and the log file to the 'F:\Log' folder. See SQL command:	**SQL command:** `restore database NYCDataSets from url = 'https://mdwresources.blob.core.windows.net/nycdatasets/NYCDataSets.Full.bak' with move 'NYCDataSets' to 'F:\Data\NYCDataSets.mdf' , move 'NYCDataSets_log' to 'F:\Log\NYCDataSets_log.ldf' Go`
6. The full solution script is saved in the Student's Computer path **C:\ADSIAD\Lab\Lab0\Solution\Restore NYCDataSets.sql**.	

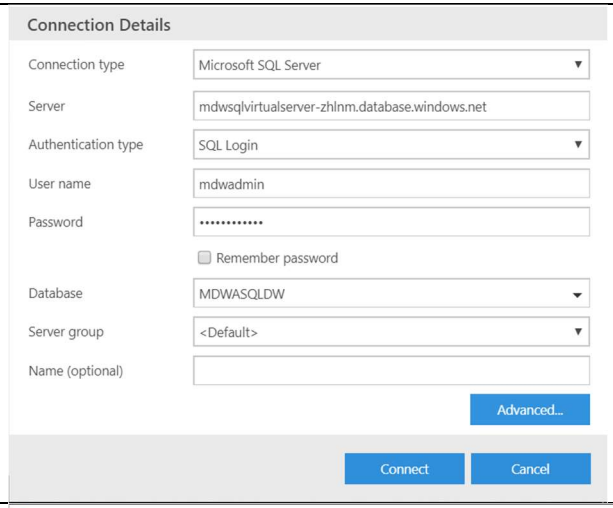
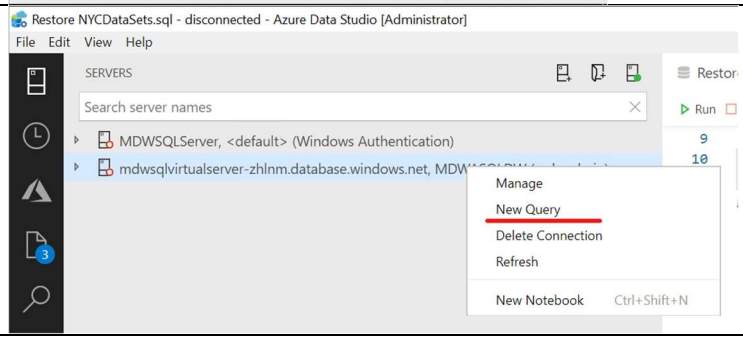
Create Azure SQL Data Warehouse database objects

In this section you will connect to Azure SQL Data Warehouse to create the database objects used to host and process data.



IMPORTANT: The full solution script is saved in the Student's Computer path **C:\ADSIAD\Lab\Lab1\Solution\Create MotorVehicleCollisions Table.sql**.

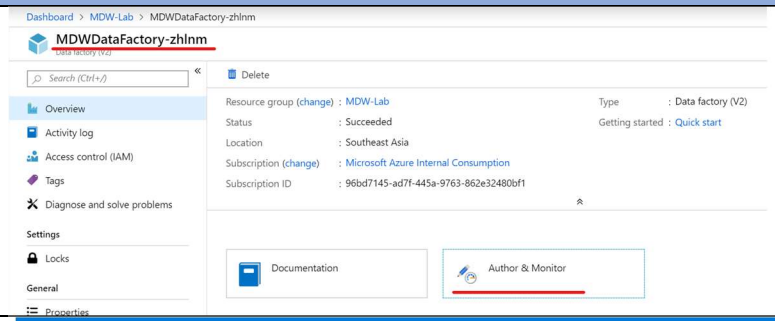
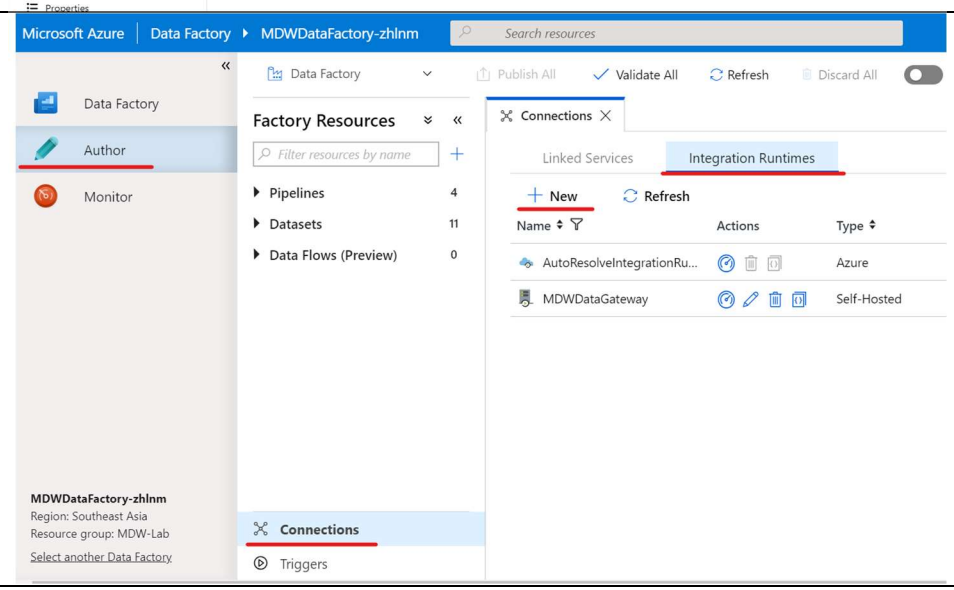
IMPORTANT: Execute these steps inside the MDWDesktop remote desktop connection	
1. Open Azure Data Studio. On the **Servers** panel, click **New Connection.**	

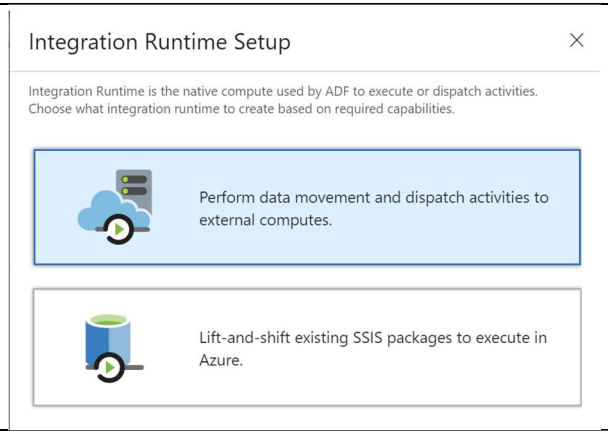
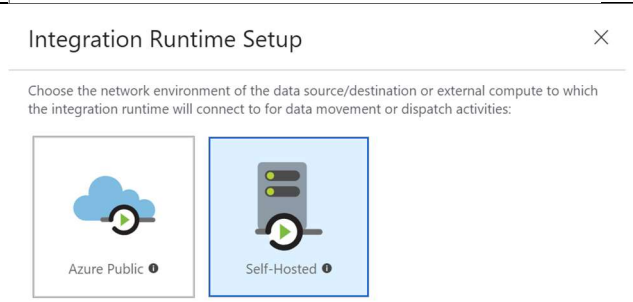
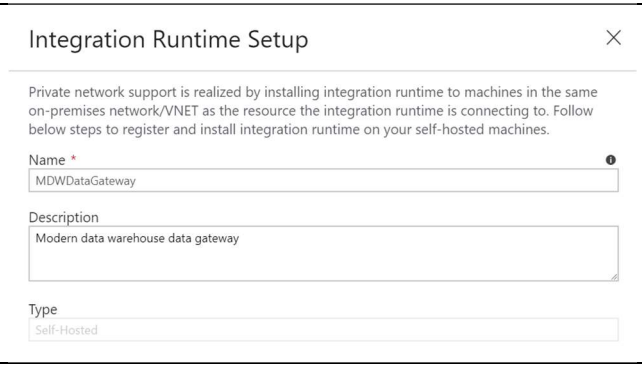
2. On the Connection Details panel, enter the following connection details: a. **Server**: mdwsqlvirtualserver-*suffix*.database.windows.net b. **Authentication Type**: SQL Login c. **User Name**: mdwadmin d. **Password**: P@ssw0rd123! e. **Database**: MDWASQLDW 3. Click **Connect.**	
4. Right-click the server name and click **New Query.**	
5. On the new query window, create a new database schema named **[NYC]**. See SQL Command.	**SQL Command:** `create schema [NYC]` `go`
6. Create a new round robin distributed table named **NYC.NYPD_MotorVehicleCollisions**, see column definitions on the SQL Command.	**SQL Command:** `create table [NYC].[NYPD_MotorVehicleCollisions](` `[UniqueKey] [int] NULL,` `[CollisionDate] [date] NULL,` `[CollisionDayOfWeek] [varchar](9) NULL,` `[CollisionTime] [time](7) NULL,` `[CollisionTimeAMPM] [varchar](2) NOT NULL,`

```
        [CollisionTimeBin] [varchar](11) NULL,
        [Borough] [varchar](200) NULL,
        [ZipCode] [varchar](20) NULL,
        [Latitude] [float] NULL,
        [Longitude] [float] NULL,
        [Location] [varchar](200) NULL,
        [OnStreetName] [varchar](200) NULL,
        [CrossStreetName] [varchar](200) NULL,
        [OffStreetName] [varchar](200) NULL,
        [NumberPersonsInjured] [int] NULL,
        [NumberPersonsKilled] [int] NULL,
        [IsFatalCollision] [int] NOT NULL,
        [NumberPedestriansInjured] [int] NULL,
        [NumberPedestriansKilled] [int] NULL,
        [NumberCyclistInjured] [int] NULL,
        [NumberCyclistKilled] [int] NULL,
        [NumberMotoristInjured] [int] NULL,
        [NumberMotoristKilled] [int] NULL,
        [ContributingFactorVehicle1] [varchar](200) NULL,
        [ContributingFactorVehicle2] [varchar](200) NULL,
        [ContributingFactorVehicle3] [varchar](200) NULL,
        [ContributingFactorVehicle4] [varchar](200) NULL,
        [ContributingFactorVehicle5] [varchar](200) NULL,
        [VehicleTypeCode1] [varchar](200) NULL,
        [VehicleTypeCode2] [varchar](200) NULL,
        [VehicleTypeCode3] [varchar](200) NULL,
        [VehicleTypeCode4] [varchar](200) NULL,
        [VehicleTypeCode5] [varchar](200) NULL
)
with (distribution = round_robin)
go
```

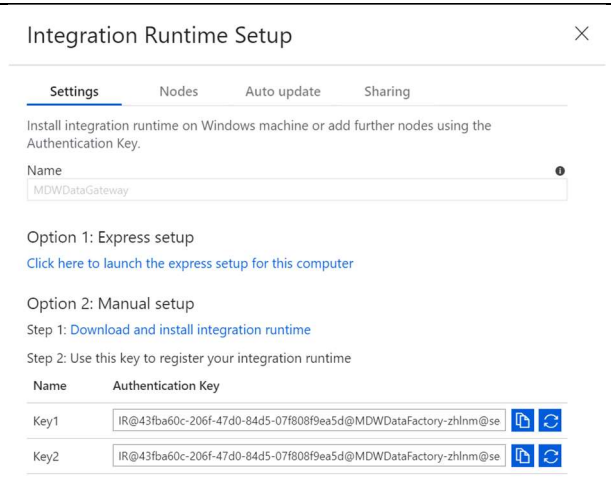
Configure the Azure Data Factory Self Hosted Integration Runtime

In this section you are going to install and configure required software onto MDWDataGateway.

IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, navigate to the MDW-Lab resource group and locate the Azure Data Factory **MDWDataFactory-*suffix***. 2. On the **MDWDataFactory-*suffix*** blade, click the **Author & Monitor** button. The Azure Data Factory portal will open on a new browser tab.	
3. On the **Azure Data Factory** portal, click the **Author (pencil icon)** button on the left-hand side menu. On the **Connections** tab, click **Integration Runtimes**. 4. Click the **+ New** button to create a new Integration Runtime.	

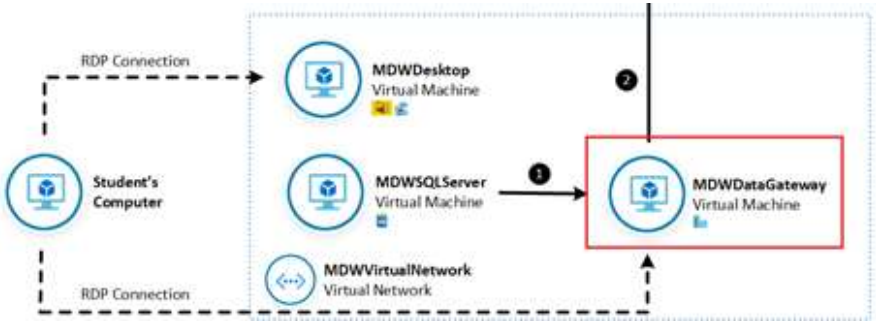
5. On the **Integration Runtime Setup** blade, select **Perform data movement and dispatch activities to external computers** and click **Next**.	
6. When prompted to choose what network environment the integration runtime will connect to, select **Self-Hosted** and click **Next**.	
7. Type **MDWDataGateway** in the **Name** text box and give it a meaningful description such as the example here. Click **Next**.	

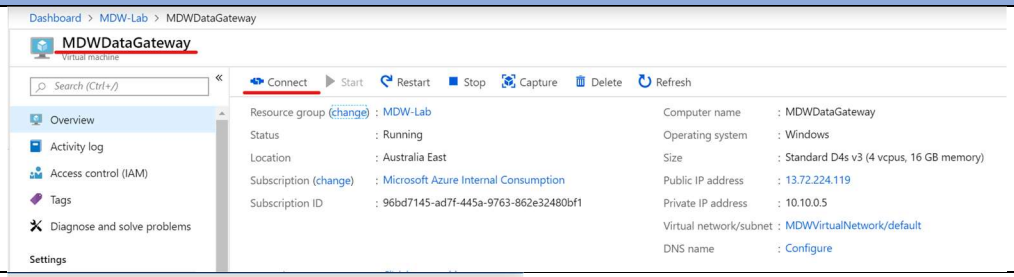
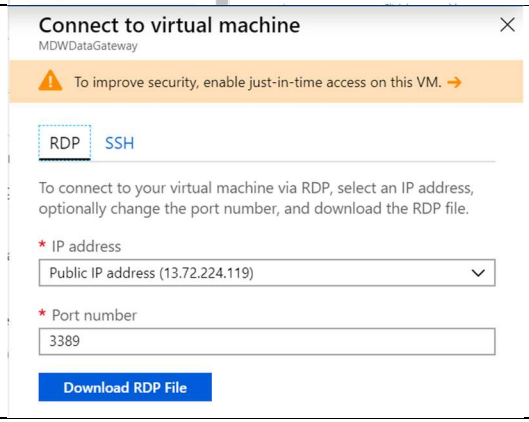
8. Copy any of the generated **Authentication Key** keys (Key 1 or Key 2) to Notepad. You are going to need it in the next step.
9. Click **Finish**.



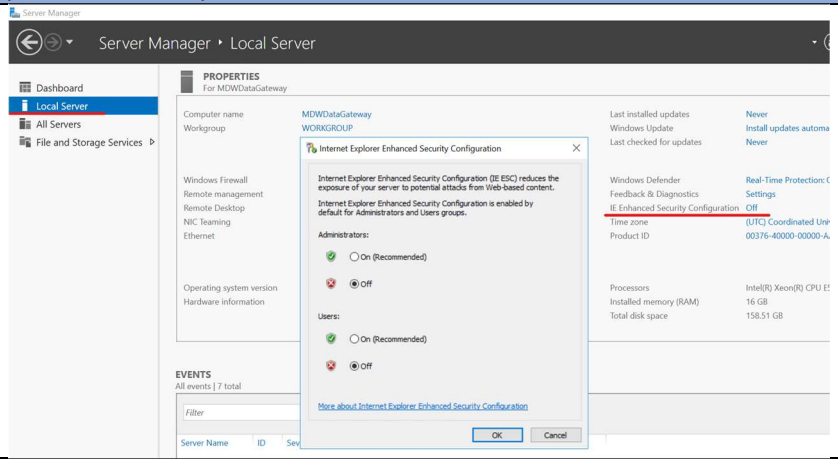
Connect to MDWDataGateway and register the Self Hosted Integration Runtime with Azure Data Factory

In this section you are going to establish a Remote Desktop Connection to MDWDataGateway virtual machine.

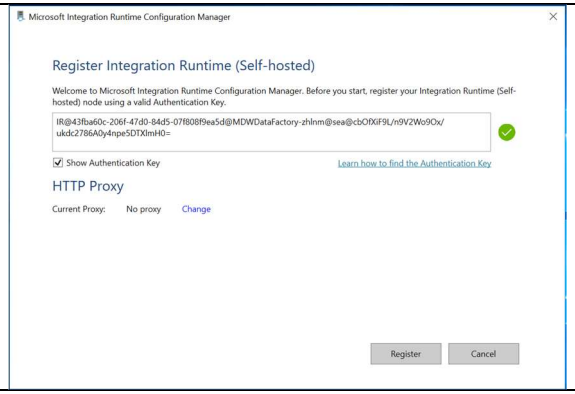


IMPORTANT: Execute these steps on your host computer	
1. On the Azure Portal, navigate to the **MDW-Lab** resource group and locate the **MDWDataGateway** virtual machine. 2. On the **MDWDataGateway** blade, from the **Overview** menu, click the **Connect** button.	
3. On the **Connect to virtual machine** blade, click **Download RDP File**. This will download a .rdp file that you can use to establish a Remote Desktop Connection with the virtual machine.	

4. Once the file is downloaded, click on file to establish the RDP connection with **MDWDataGateway** 5. User the following credentials to authenticate: a. **User Name:** MDWAdmin b. **Password:** P@ssw0rd123!	

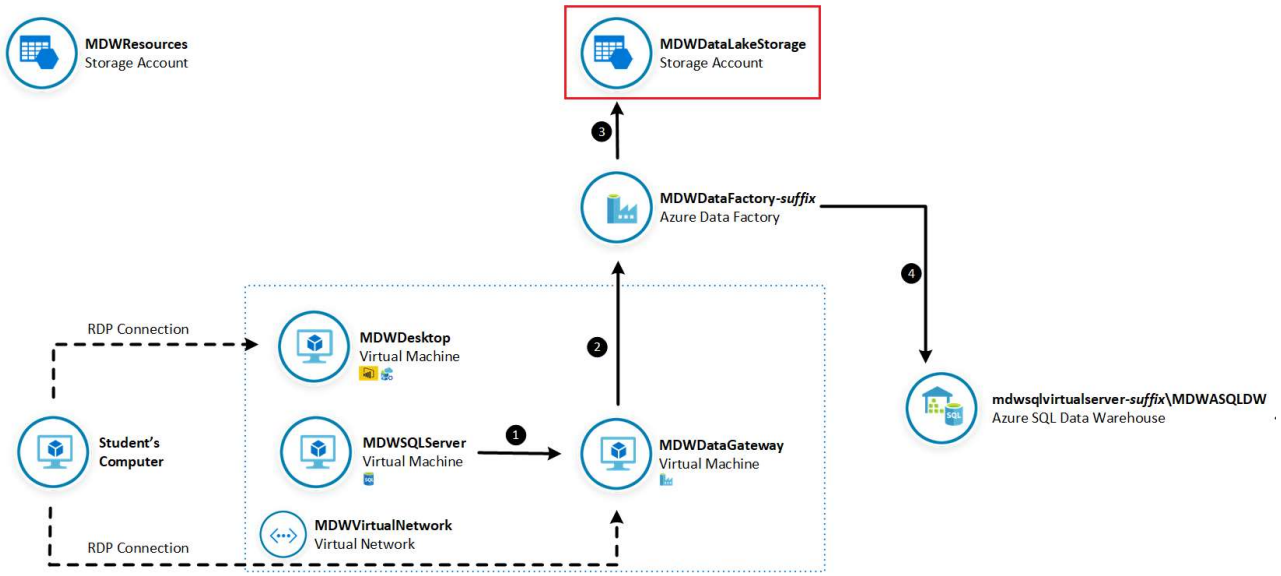
IMPORTANT: Execute these steps inside the MDWDataGateway remote desktop connection	
1. Once logged in, on the **Server Manager**, select **Local Server** on the left-hand side menu. On the right-hand side panel, locate the **IE Enhanced Security Configuration** and click the **On** link. 2. Turn the setting **Off** for both Administrators and Users. 3. Close **Server Manager**.	
4. Open the browser and download and execute the latest version of the Azure Data Factory Integration Runtime.	**Azure Data Factory Integration Runtime** https://www.microsoft.com/en-ie/download/details.aspx?id=39717

5. Accept all default options during the setup wizard. Once the setup if finished, the **Microsoft Integration Runtime Configuration Manager** will pop up asking you to enter a valid authentication key.
6. Enter the authentication key generated in the previous exercise and click **Register**.
7. Once registration is confirmed, click **Finish**.

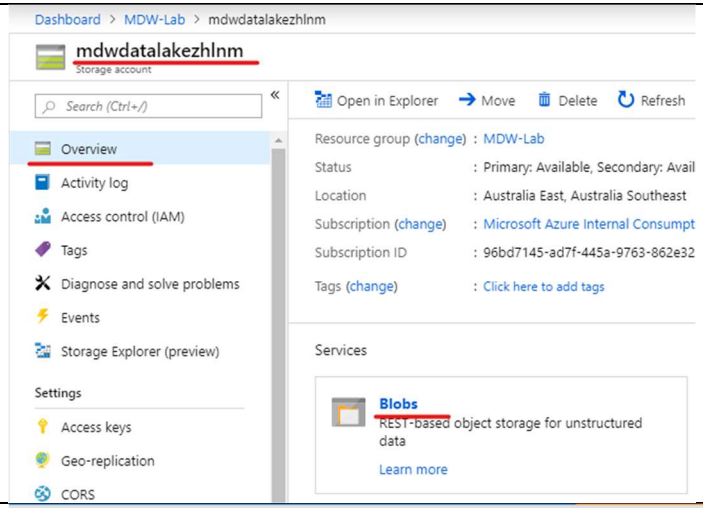
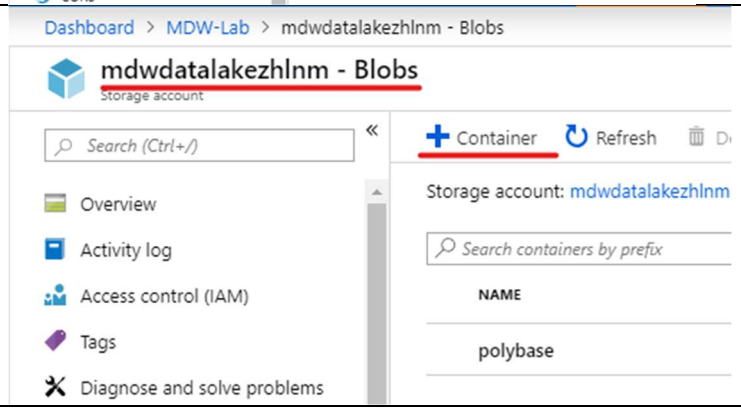


Create Staging Container on Azure Blob Storage

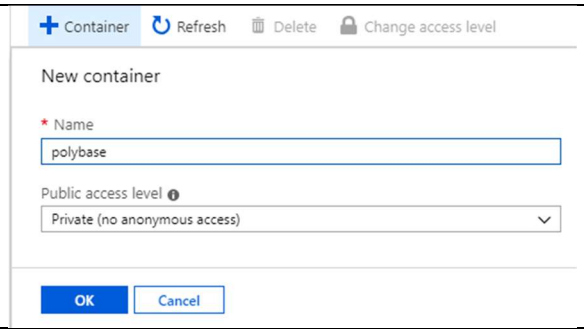
In this section you create a staging container in your MDWDataLake that will be used as a staging environment for Polybase before data can be copied to Azure SQL Data Warehouse.



IMPORTANT: Execute these steps on your host computer

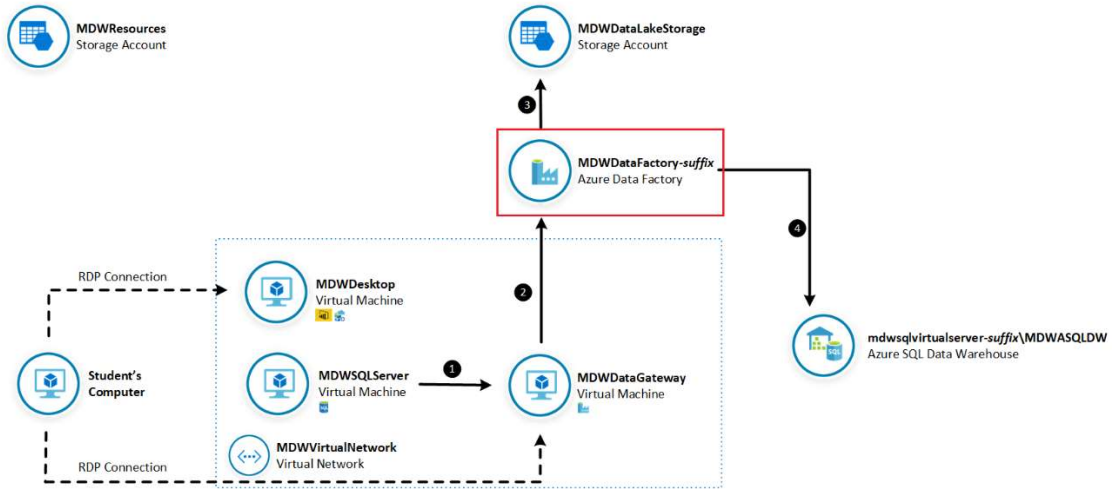
1. In the Azure Portal, go to the lab resource group and locate the Azure Storage account **mdwdatalake*suffix***. 2. On the **Overview** panel, click **Blobs**.	
3. On the mdwdalalake*suffix* – Blobs blade, click **+ Container**.	

4. On the **New container** blade, enter the following details:
    a. **Name**: polybase
    b. **Public access level**: Private (no anynymous access)
5. Click **OK** to create the new container.

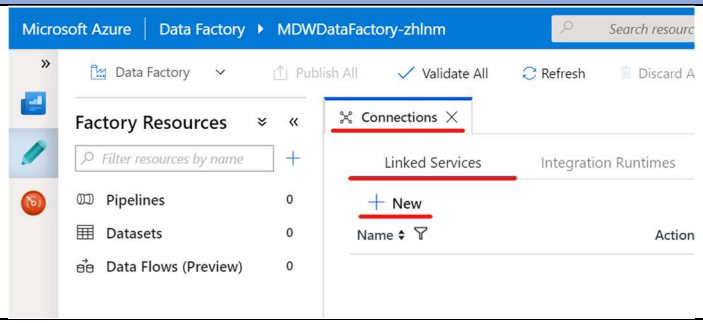


Create Azure Data Factory Pipeline to Copy Relational Data

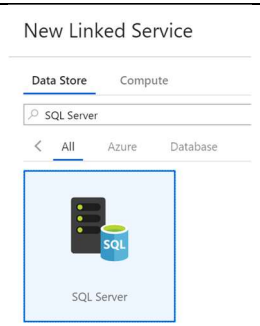
In this section you will build an Azure Data Factory pipeline to copy a table from MDWSQLServer to Azure SQL Data Warehouse.



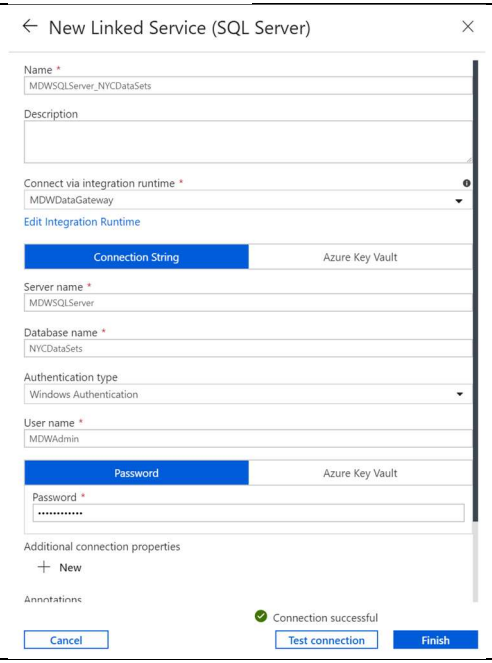
Create Linked Service connections

IMPORTANT: Execute these steps on your host computer	
1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel (pencil icon). Under **Connections** tab, click **Linked Services** and then click **+ New** to create a new linked service connection.	

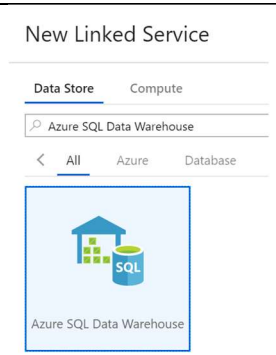
2. On the **New Linked Service** blade, type “SQL Server” in the search box to find the **SQL Server** linked service. Click **Continue**.



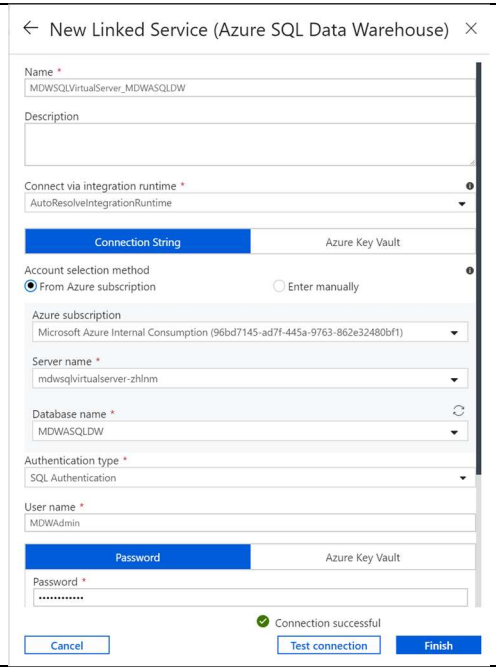
3. On the **New Linked Service (SQL Server)** blade, enter the following details:
    a. **Name**: MDWSQLServer_NYCDataSets
    b. **Connect via integration runtime**: MDWDataGateway
    c. **Server Name**: MDWSQLServer
    d. **DatabaseName**: NYCDataSets
    e. **Authentication Type**: Windows Authentication
    f. **User Name**: MDWAdmin
    g. **Password**: P@ssw0rd123!
4. Click **Test connection** to make sure you entered the correct connection details and then click Finish.



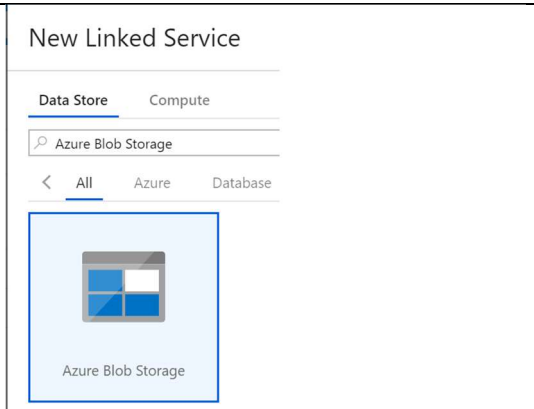
5. Repeat the process to create an Azure SQL Data Warehouse linked service connection.



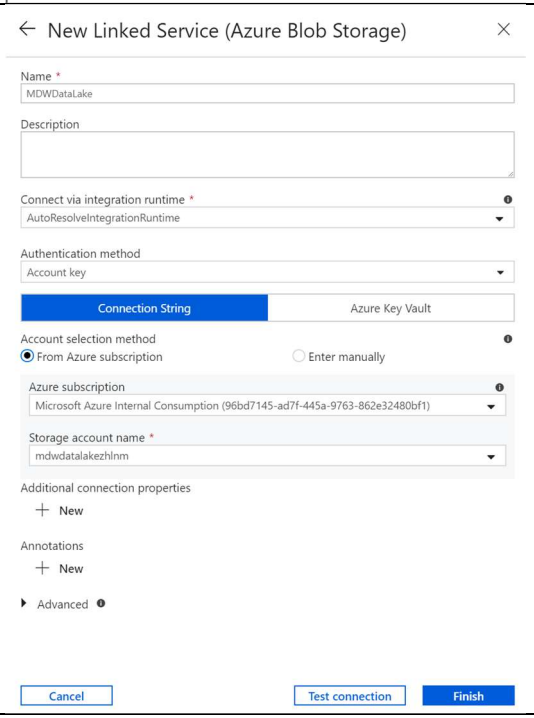
6. On the **New Linked Service (Azure SQL Data Warehouse)** blade, enter the following details:
    a. **Name**: MDWVirtualSQLServer_MDWASQLDW
    b. **Connect via integration runtime**: AutoResolveIntegrationRuntime
    c. **Account selection method**: From Azure subscription
    d. **Azure subscription**: <your subscription>
    e. **Server Name**: mdwsqlvirtualserver-*suffix*
    f. **DatabaseName**: MDWASQLDW
    g. **Authentication Type**: SQL Authentication
    h. **User Name**: MDWAdmin
    i. **Password**: P@ssw0rd123!
7. Click **Test connection** to make sure you entered the correct connection details and then click Finish.

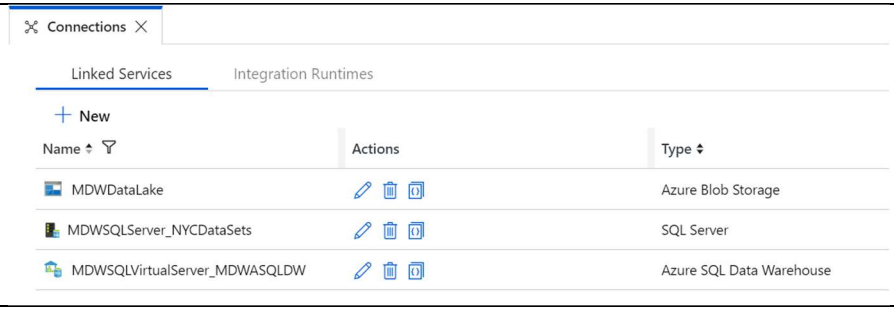


8. Repeat the process once again to create an Azure Storage linked service connection.

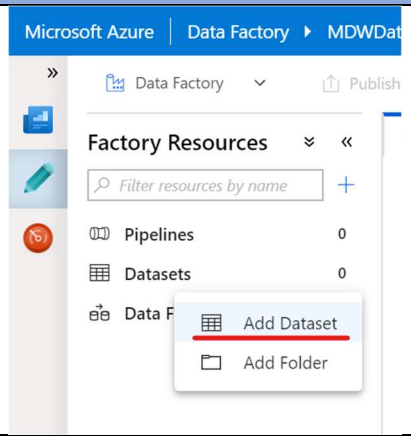


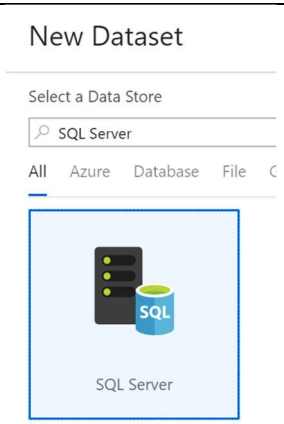
9. On the **New Linked Service (Azure Blob Storage)** blade, enter the following details:
    j. **Name**: MDWDataLake
    k. **Connect via integration runtime**: AutoResolveIntegrationRuntime
    l. **Authentication method**: Account key
    m. **Account selection method**: From Azure subscription
    n. **Azure subscription**: <your subscription>
    o. **Storage account name**: mdwdatalake*suffix*
10. Click **Test connection** to make sure you entered the correct connection details and then click Finish.

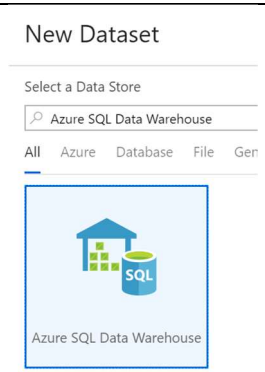
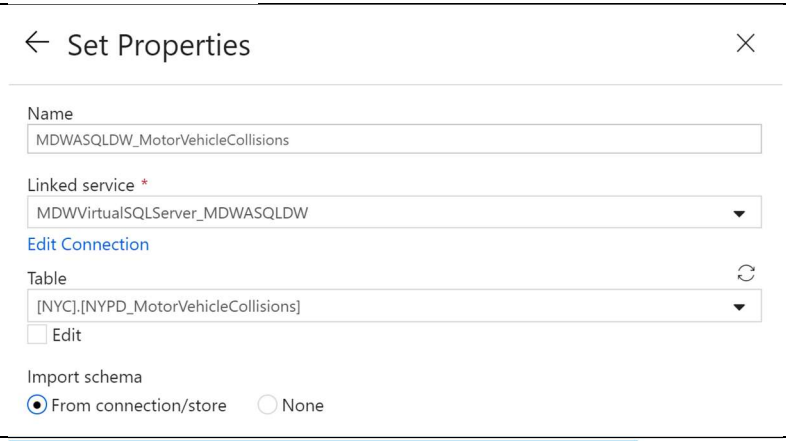
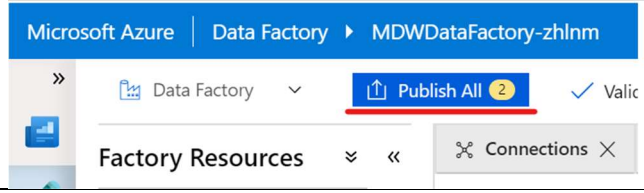


11. You should now see 3 linked services connections that will be used as source, destination and staging.	

Create Source and Destination Data Sets

IMPORTANT: Execute these steps on your host computer	
1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel (pencil icon). Under **Factory Resources** tab, click the ellipsis (**…**) next to **Datasets** and then click **Add Dataset** to create a new dataset.	

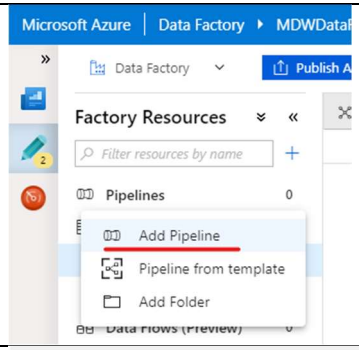
2. Type SQL Server in the search box and select **SQL Server.** Click **Finish.**	
3. On the **New Data Set** tab, enter the following details: a. **Name**: NYCDataSets_MotorVehicleCollisions b. **Linked Service:** MDWSQLServer_NYCDataSets c. **Table:** [NYC].[NYPD_MotorVehicleCollisions] 4. Leave remaining fields with default values and click Continue	

5. Repeat the process to create a new Azure SQL Data Warehouse data set.	
6. On the **New Data Set** tab, enter the following details: a. **Name**: MDWASQLDW_MotorVehicleCollisions b. **Linked Service:** MDWSQLVirtualServer_MDWASQLDW c. **Table:** [NYC].[NYPD_MotorVehicleCollisions] 7. Leave remaining fields with default values and click Continue.	
8. Publish your dataset changes by clicking the **Publish All** button on the top of the screen.	

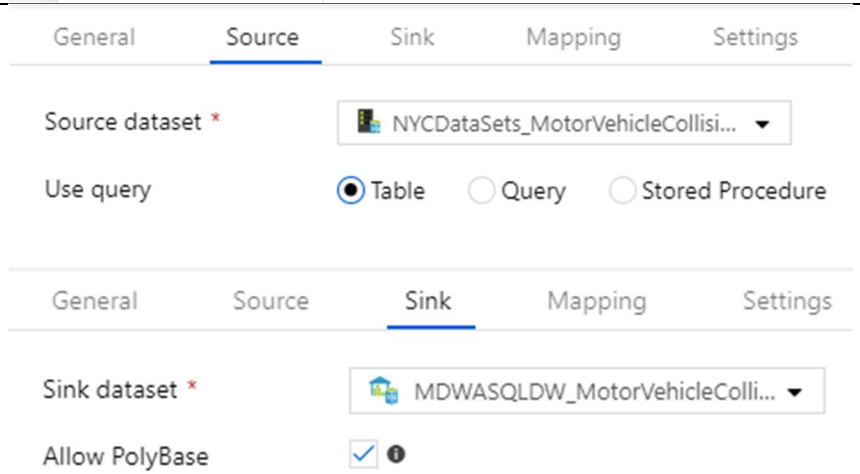
Create and Execute Pipeline

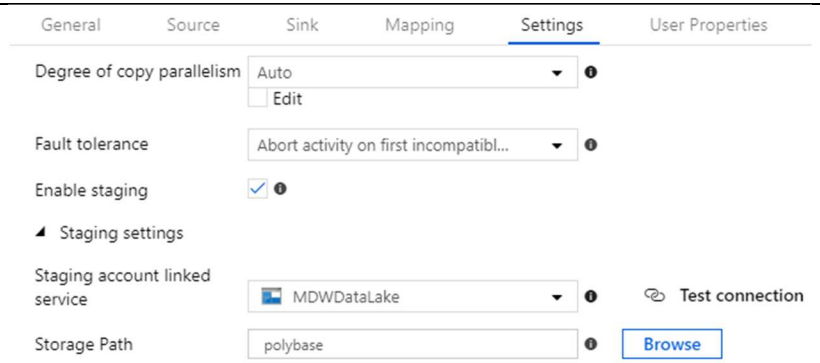
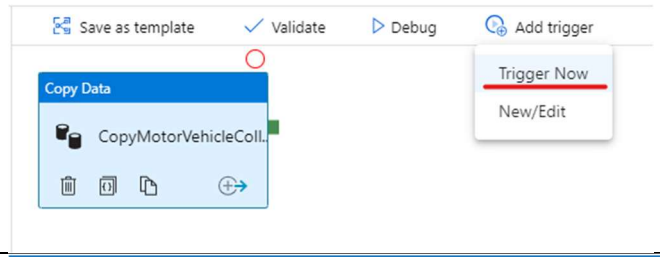
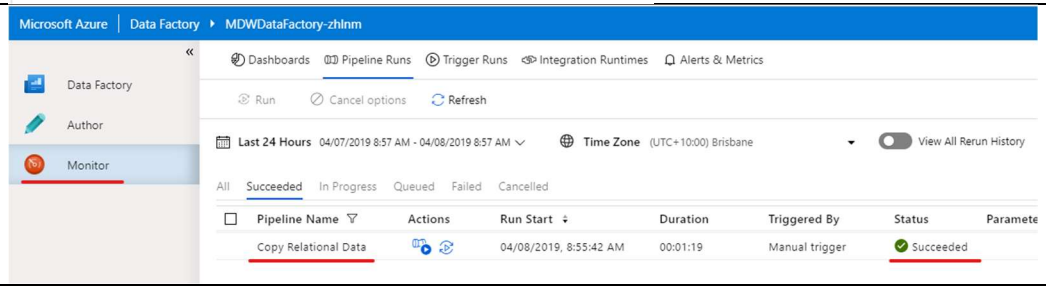
IMPORTANT: Execute these steps on your host computer

1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel (pencil icon). Under **Factory Resources** tab, click the ellipsis (**…**) next to **Pipelines** and then click **Add Pipeline** to create a new dataset.
2. On the New Pipeline tab, enter the following details:
    a. **General > Name**: Copy Relational Data
3. Leave remaining fields with default values.



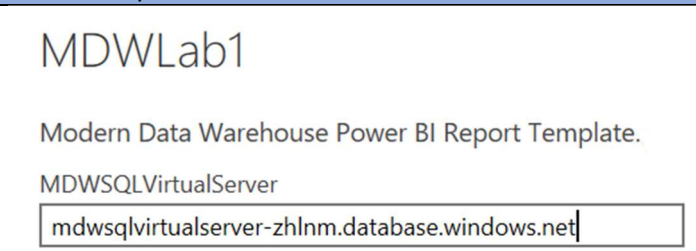
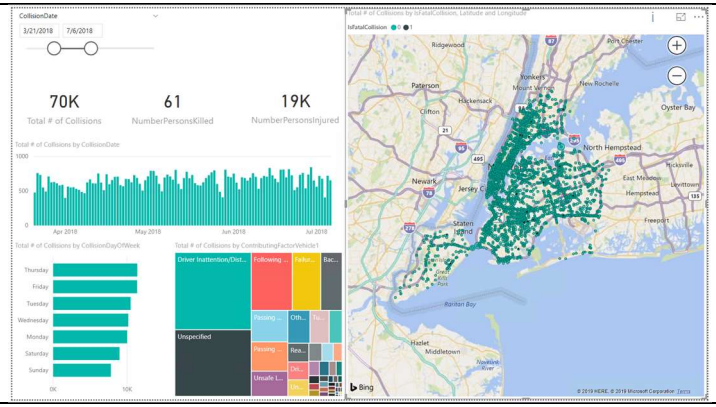
4. From the Activities panel, type “Copy Data” in the search box. Drag the Copy Data activity on to the design surface.
5. Select the Copy Data activity and enter the following details:
    a. **General > Name:** CopyMotorVehicleCollisions
    b. **Source > Source dataset:** NYCDataSets_MotorVehicleCollisions
    c. **Sink > Sink dataset:** MDWASQLDW_MotorVehicleCollisions
    d. **Sink > Allow PolyBase:** Checked
    e. **Settings > Enable staging:** Checked
    f. **Settings > Staging account linked service:** MDWDataLake
    g. **Settings > Storage Path:** polybase
6. Leave remaining fields with default values.



<table>
<tr><td></td><td> 
</td></tr>
<tr><td>7. Publish your pipeline changes by clicking the Publish all button.</td><td> 
</td></tr>
<tr><td>8. To execute the pipeline, click on Add trigger menu and then Trigger Now.
9. On the Pipeline Run blade, click Finish.</td><td> 
</td></tr>
<tr><td>10. To monitor the execution of your pipeline, click on the Monitor menu on the left-hand side panel.
11. You should be able to see the Status of your pipeline execution on the right-hand side panel.</td><td> 
</td></tr>
</table>

Visualize Data with Power BI

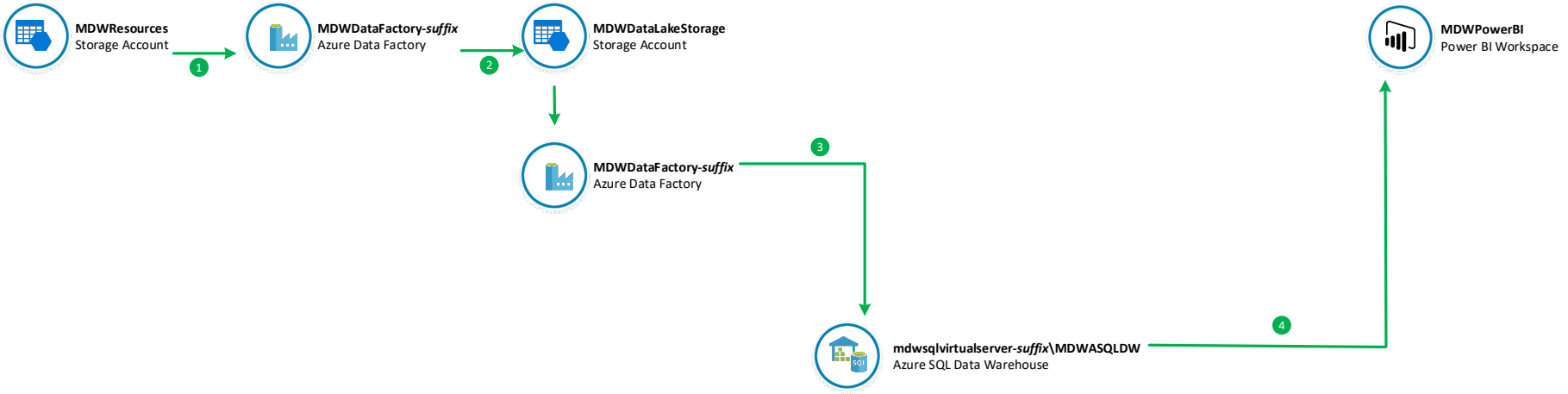
In this section you are going to use Power BI to visualize data from Azure SQL Data Warehouse. The Power BI report will use an Import connection to query Azure SQL Data Warehouse and visualise Motor Vehicle Collision data from the table you loaded in the previous exercise.

IMPORTANT: Execute these steps inside the MDWDesktop remote desktop connection	
1. On MDWDesktop, download the Power BI report from the link https://aka.ms/MDWLab1 and save it on the Desktop. 2. Open the file MDWLab1.pbit with Power BI Desktop. Optionally sign up for the Power BI tips and tricks email, or to dismiss this, click to sign in with an existing account, and then hit the escape key. 3. When prompted to enter the value of the **MDWSQLVirtualServer** parameter, type the full server name: **mdwsqlvirtualserver-*suffix*.database.windows.net** 4. Click **Load, and then Run to acknowledge the Native Database Query message** 5. When prompted, enter the credentials to connect to the database – Username is **MDWAdmin**, Password is **P@ssw0rd123!**	
6. Once the data is finished loading, interact with the report by changing the CollisionDate slicer and by clicking on the other visualisations. 7. Save your work and close Power BI Desktop.	

Lab 2: Transform Big Data using Azure Data Factory and Azure SQL Data Warehouse

In this lab you will use Azure Data Factory to download large data files to your data lake and use Azure SQL Data Warehouse to generate a summary dataset and store it. The dataset you will use contains detailed New York City Yellow Taxi rides for 2018. You will generate a daily aggregated summary of all rides and save the result in your data warehouse. You will use Power BI to visualise summarised taxi ride data.

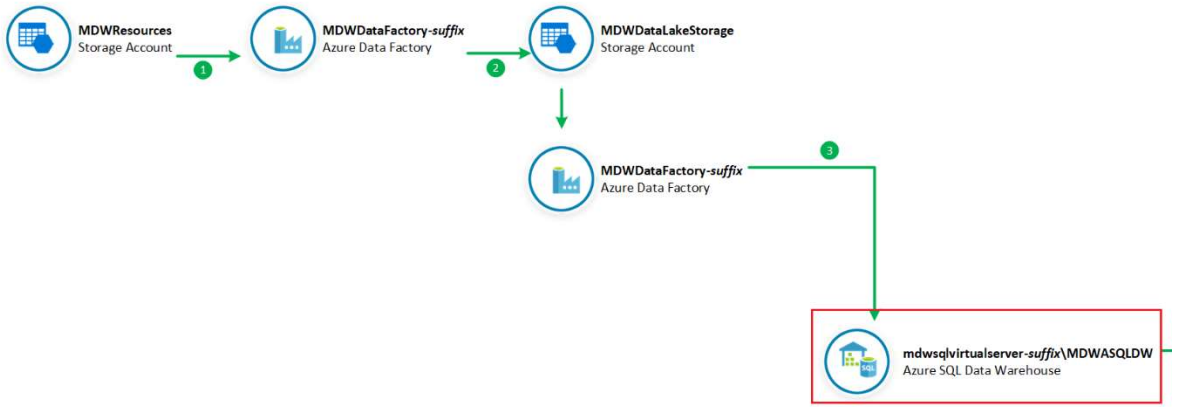
Lab Architecture



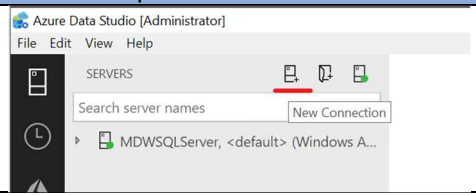
1. Build an Azure Data Factory Pipeline to copy big data files from shared Azure Storage
2. Save data files to your data lake
3. Use Polybase to load data into staging tables in your Azure SQL Data Warehouse.
   Call a Stored Procedure to perform data aggregations and save results in the final table.
4. Visualize data from your Azure SQL Data Warehouse using Power BI

Create Azure SQL Data Warehouse database objects

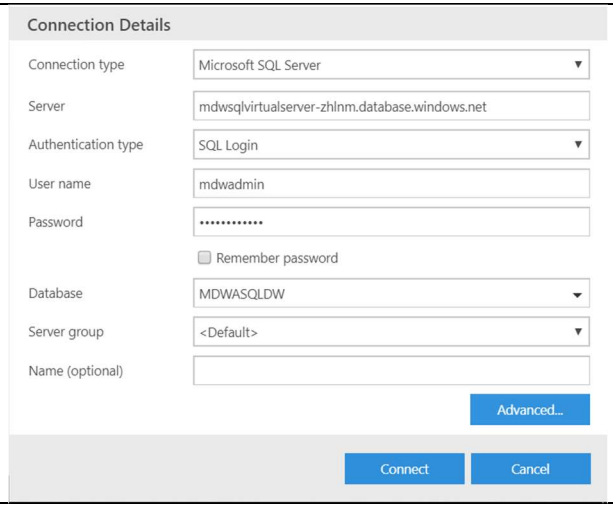
In this section you will connect to Azure SQL Data Warehouse to create the database objects used to host and process data.



IMPORTANT: The full solution script is saved in the Student's Computer path **C:\ADSIAD\Lab\Lab2\Solution\Create Staging NYCTaxiData.sql**.

IMPORTANT: Execute these steps inside the MDWDesktop remote desktop connection	
1. Open Azure Data Studio. 2. If you already have a connection to MDWSQLVirtualServer, then go to step 6. 3. On the **Servers** panel, click **New Connection.**	

4. On the Connection Details panel, enter the following connection details:
    a. **Server**: mdwsqlvirtualserver-suffix.database.windows.net
    b. **Authentication Type**: SQL Login
    c. **User Name**: mdwadmin
    d. **Password**: P@ssw0rd123!
    e. **Database**: MDWASQLDW
5. Click **Connect.**



6. Right click the MDWSQLVirtualServer name and then click **New Query**.

7. Create a new round robin distributed table named NYC.TaxiDataSummary – execute the script shown to the right

SQL Command

IMPORTANT: The full solution script is saved in the Student's Computer path **C:\ADSIAD\Lab\Lab2\Solution\Create Staging NYCTaxiData.sql**.

```
create schema [Staging]
go

create table [Staging].[NYCTaxiData]
(
    [VendorID] [int] NULL,
    [tpep_pickup_datetime] [datetime] NULL,
    [tpep_dropoff_datetime] [datetime] NULL,
    [passenger_count] [smallint] NULL,
    [trip_distance] [decimal](8, 2) NULL,
    [RatecodeID] [smallint] NULL,
    [store_and_fwd_flag] [char](1) NULL,
    [PULocationID] [int] NULL,
```

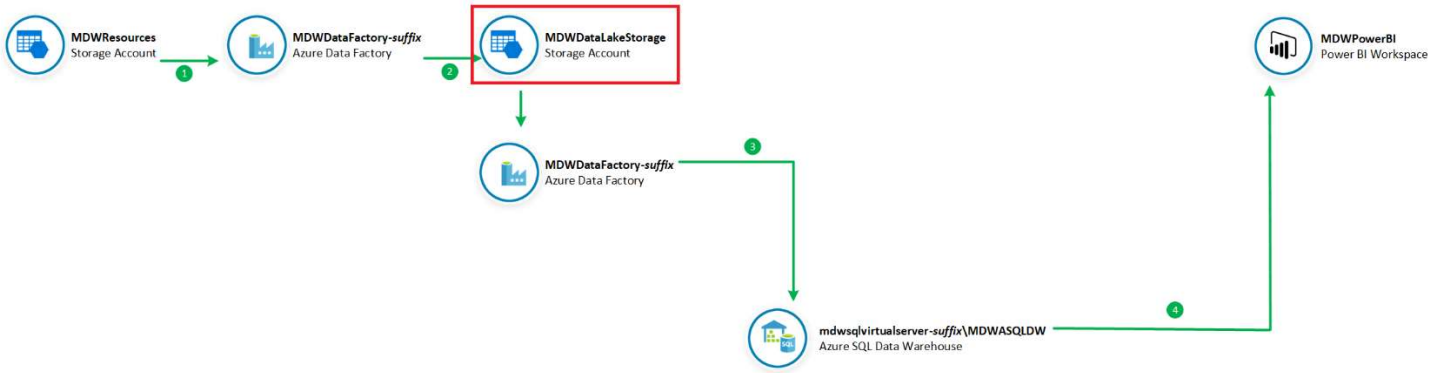
	[DOLocationID] [int] NULL, [payment_type] [smallint] NULL, [fare_amount] [decimal](10, 2) NULL, [extra] [decimal](10, 2) NULL, [mta_tax] [decimal](10, 2) NULL, [tip_amount] [decimal](10, 2) NULL, [tolls_amount] [decimal](10, 2) NULL, [improvement_surcharge] [decimal](10, 2) NULL, [total_amount] [decimal](10, 2) NULL) with (distribution = round_robin, heap) go
8. Create a new Staging table called NYCTaxiLocationLookup to host NYC taxi location lookup data – execute the script shown to the right	**SQL Command:** create table [Staging].[NYCTaxiLocationLookup] ([LocationID] [int] NULL, [Borough] [varchar](200) NULL, [Zone] [varchar](200) NULL, [service_zone] [varchar](200) NULL) with (distribution = round_robin, clustered columnstore index)
9. Create a stored procedure that will transform and aggregate data and save the result in a new table	**SQL Command:** create procedure Staging.spNYCLoadTaxiDataSummary

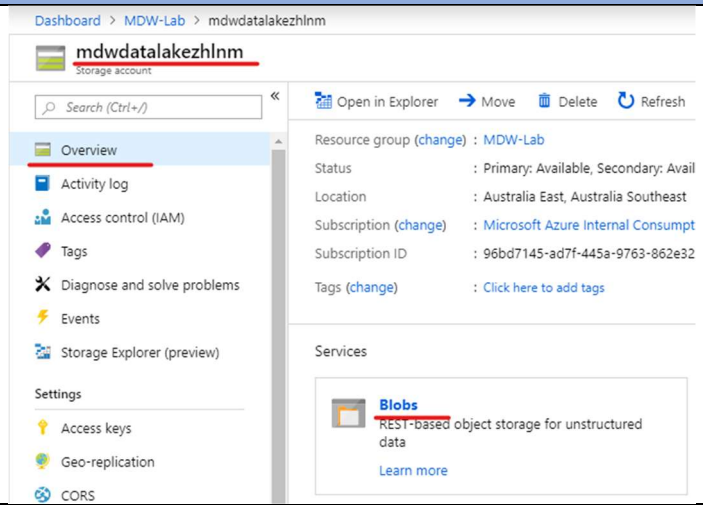
called NYC.TaxiDataSummary – execute the script shown to the right	as --Drop and re-create Staging.idx_NYCTaxiData index if (exists(select top 1 1 from sys.indexes where name = 'idx_NYCTaxiData' and object_id = object_id('Staging.NYCTaxiData'))) drop index idx_NYCTaxiData on Staging.NYCTaxiData create index idx_NYCTaxiData on Staging.NYCTaxiData(tpep_pickup_datetime, PULocationID, payment_type, passenger_count, trip_distance, tip_amount, fare_amount, total_amount) --Drop and re-create NYC.TaxiDataSummary table if (exists(select top 1 1 from sys.objects where name = 'TaxiDataSummary' and schema_id = schema_id('NYC') and type = 'U')) drop table NYC.TaxiDataSummary create table NYC.TaxiDataSummary with (distribution = round_robin) as select cast(tpep_pickup_datetime as date) as PickUpDate , PickUp.Borough as PickUpBorough , PickUp.Zone as PickUpZone , case payment_type when 1 then 'Credit card' when 2 then 'Cash' when 3 then 'No charge' when 4 then 'Dispute' when 5 then 'Unknown' when 6 then 'Voided trip'

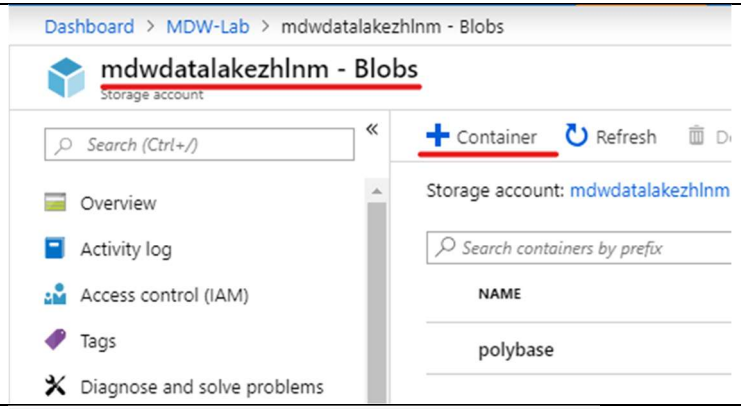
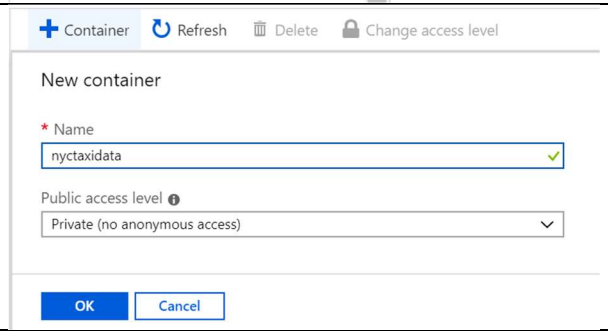
```
        end as PaymentType
        , count(*) as TotalTripCount
        , sum(passenger_count) as TotalPassengerCount
        , sum(trip_distance) as TotalDistanceTravelled
        , sum(tip_amount) as TotalTipAmount
        , sum(fare_amount) as TotalFareAmount
        , sum(total_amount) as TotalTripAmount
    from Staging.NYCTaxiData
        inner join Staging.NYCTaxiLocationLookup as PickUp
            on NYCTaxiData.PULocationID = PickUp.LocationID
    group by cast(tpep_pickup_datetime as date)
        , PickUp.Borough
        , PickUp.Zone
        , payment_type

    --drop index idx_NYCTaxiData so it does not impact future loads
    drop index idx_NYCTaxiData on Staging.NYCTaxiData
go
```

Create NYCTaxiData Container on Azure Blob Storage

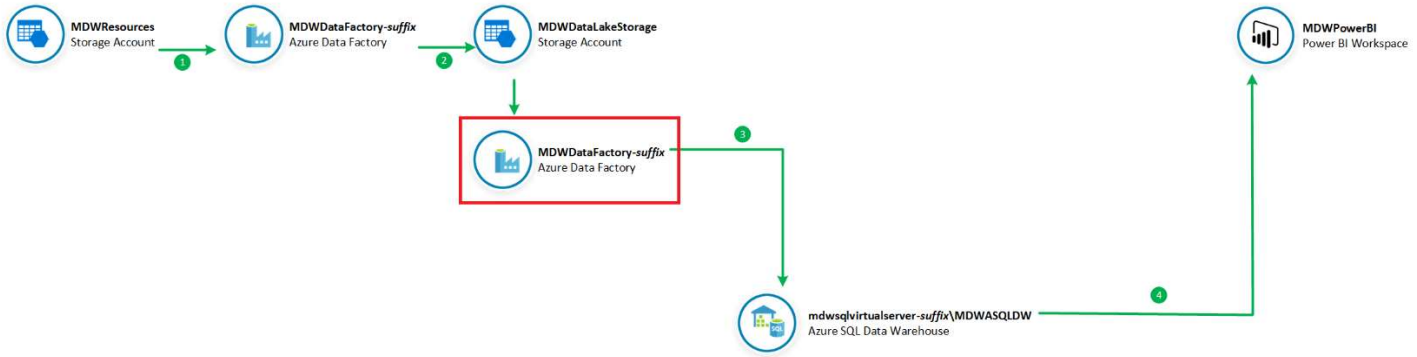
In this section you create a container in your MDWDataLake that will be used as a repository for the NYC Taxi Data files. You will copy 12 files from the MDWResources Storage Account into your NYCTaxiData container. These files contain data for all Yellow Taxi rides in 2018, one file for each month of the year.

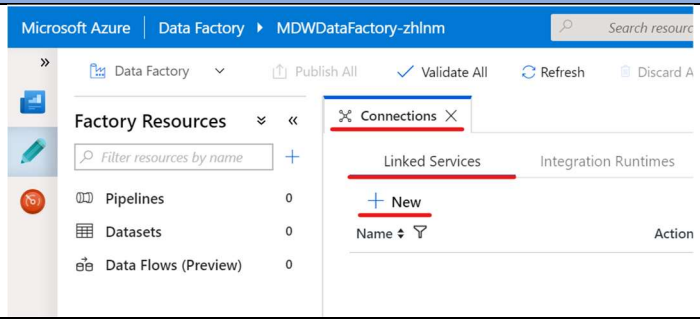


IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, go to the lab resource group and locate the Azure Storage account **mdwdatalake*suffix***. 2. On the **Overview** panel, click **Blobs**.	

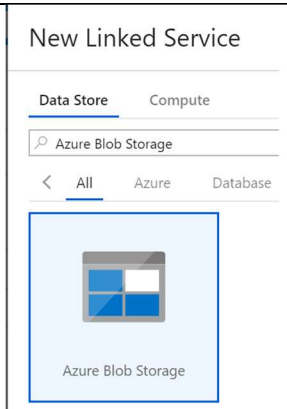
3. On the **mdwdalalake*suffix* – Blobs** blade, click **+ Container**.	
4. On the **New container** blade, enter the following details: a. **Name**: nyctaxidata b. **Public access level**: Private (no anynymous access) 5. Click **OK** to create the new container.	

Create Linked Service connection to MDWResources

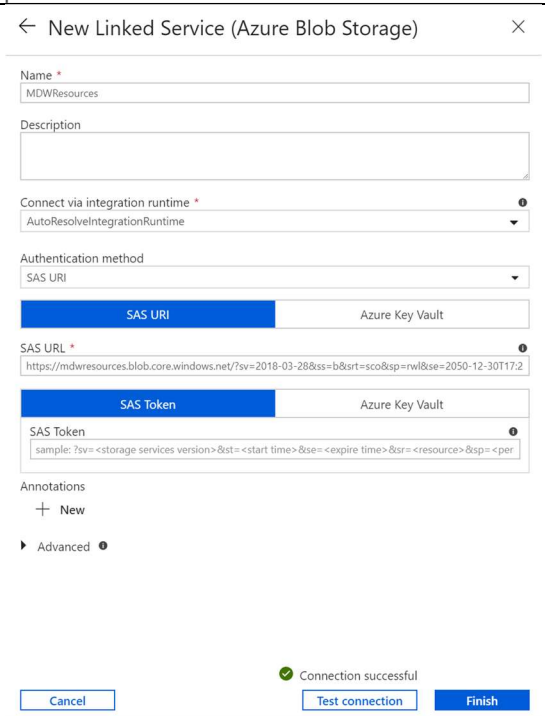


IMPORTANT: Execute these steps on your host computer	
1. Open the **Azure Data Factory** portal and click the **Author** option (pencil icon) on the left-hand side panel. Under **Connections** tab, click **Linked Services** and then click **+ New** to create a new linked service connection.	

2. On the **New Linked Service** blade, type “Azure Blob Storage” in the search box to find the **Azure Blob Storage** linked service. Click **Continue**.



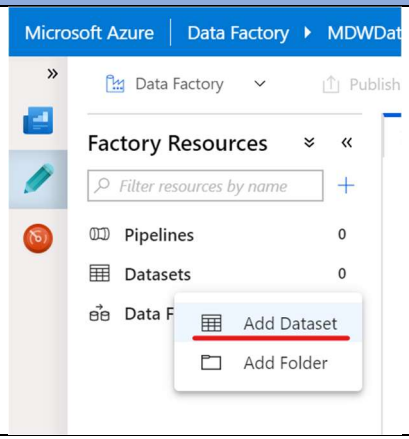
3. On the **New Linked Service (Azure Blob Storage)** blade, enter the following details:
    a. **Name**: MDWResources
    b. **Connect via integration runtime**: AutoResolveIntegrationRuntime
    c. **Authentication method**: SAS URI
    d. **SAS URL**: https://mdwresources.blob.core.windows.net/?sv=2018-03-28&ss=b&srt=sco&sp=rwl&se=2050-12-30T17:25:52Z&st=2019-04-05T09:25:52Z&spr=https&sig=4qrD8NmhaSmRFu2gKja67ayohfIDEQH3LdVMa2Utykc%3D
4. Alternatively copy SAS URL from **C:\ADSIAD\Lab\Lab2\MDWResources SAS URL.txt**
5. Click **Test connection** to make sure you entered the correct connection details and then click Finish.

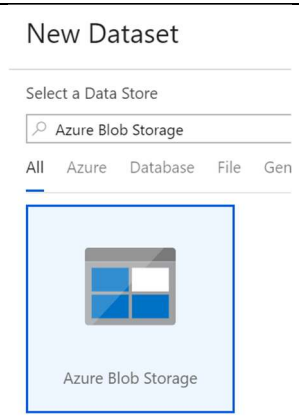
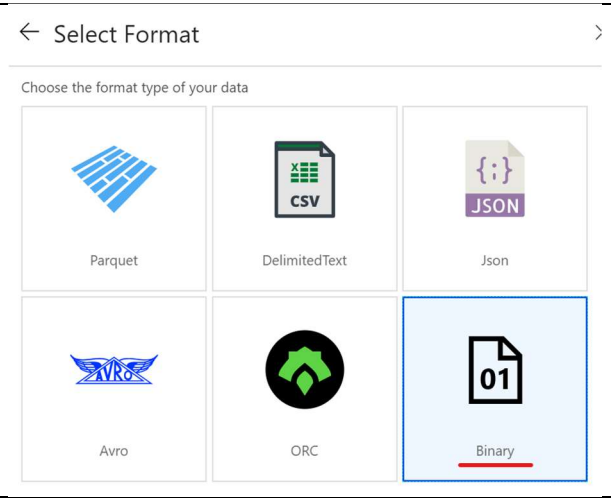


Create Source and Destination Data Sets

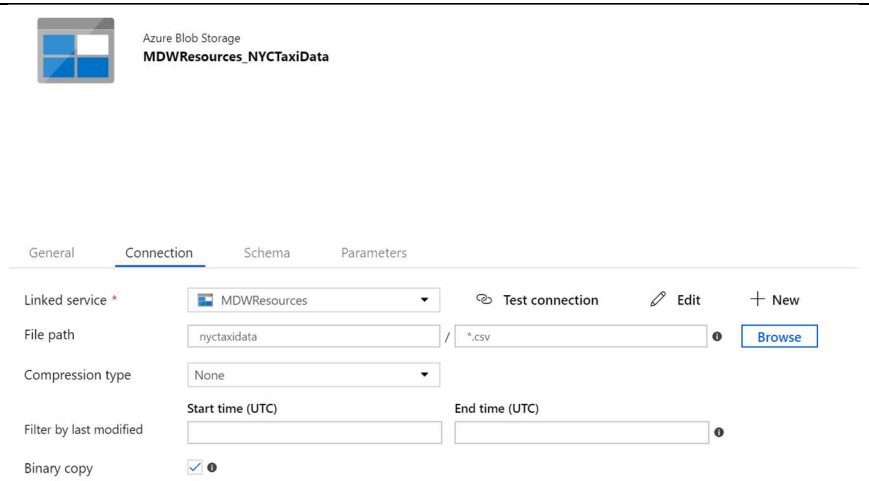
In this section you are going to create 5 datasets that will be used by your data pipeline:

- **MDWResources_NYCTaxiData**: references MDWResources storage account container that contains source data files.
- **MDWResources_NYCTaxiLookup**: references MDWResources storage account that contains a .csv file with all taxi location codes and names.
- **MDWASQLDW_StagingNYCTaxiData**: references the table Staging.NYCTaxiData in the Azure SQL Data Warehouse database MDWASQLDW.
- **MDWASQLDW_StagingNYCLocationLookup**: references the table [Staging].[NYCTaxiLocationLookup] in the Azure SQL Data Warehouse database MDWASQLDW and acts as destination of lookup data copied from **MDWResources_NYCTaxiLookup**.
- **MDWDataLake_NYCTaxiData**: references your MDWDataLake-suffix storage account. It acts as the destination of files copied from **MDWResources_NYCTaxiData** and also as a data source when copying data to **MDWASQLDW_StagingNYCTaxiData**.

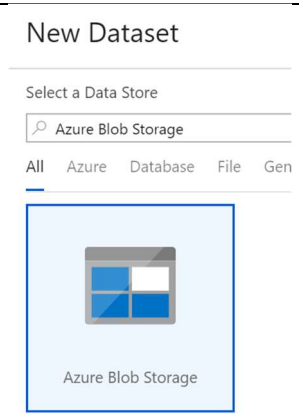
IMPORTANT: Execute these steps on your host computer	
1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel. Under **Factory Resources** tab, click the ellipsis (**…**) next to **Datasets** and then click **Add Dataset** to create a new dataset.	

2. Type “Azure Blob Storage” in the search box and select **Azure Blob Storage.** Click **Continue.**	
3. On the **Select Format** blade, select **Binary** and click **Continue.**	

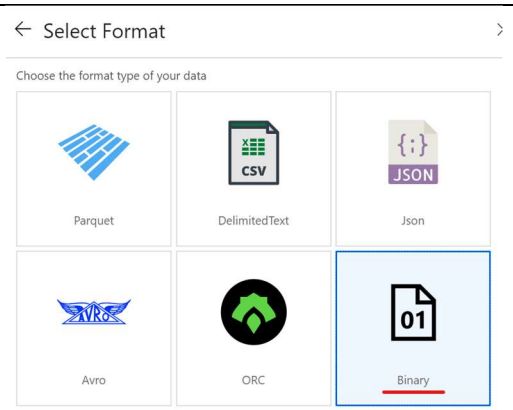
4. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWResources_NYCTaxiData
    b. **Connection > Linked Service:** MDWResources
    c. **Connection > File Path:** nyctaxidata / *.csv
    d. **Connection > Binary Copy:** Checked
5. Leave remaining fields with default values.



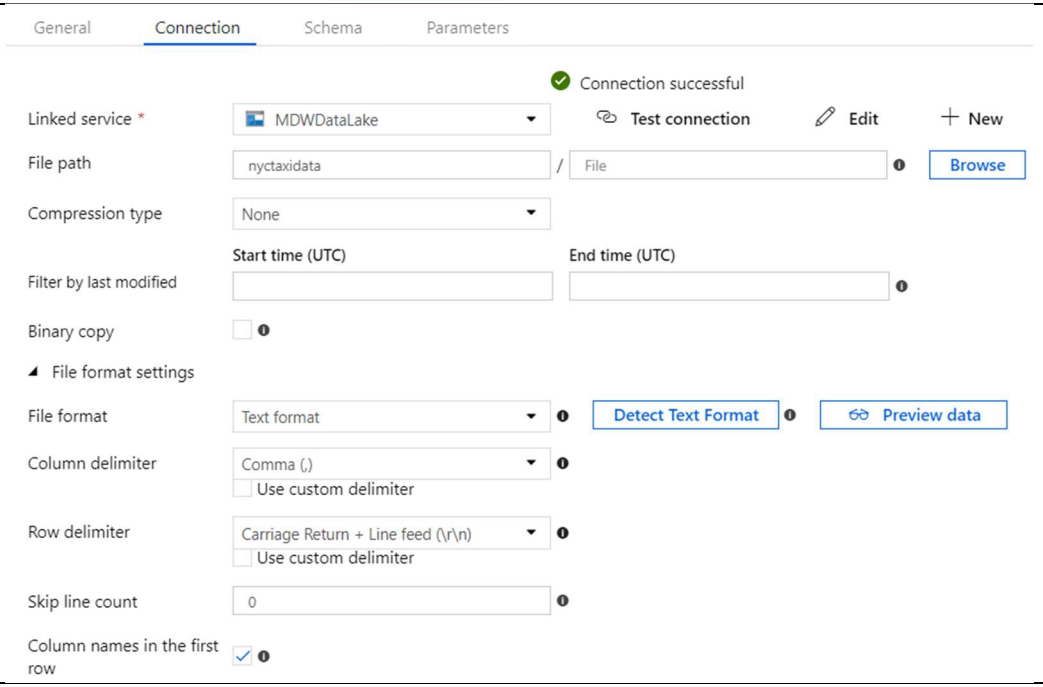
6. Repeat the process to create another dataset, this time referencing the NYCTaxiData container in your MDWDataLake storage account.
7. Type “Azure Blob Storage” in the search box and select **Azure Blob Storage.** Click **Continue.**

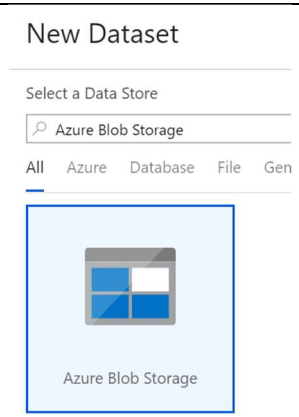
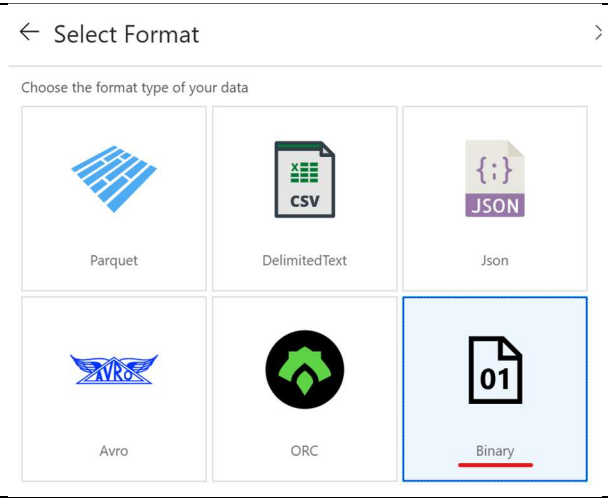


8. On the Select Format blade, select **Binary** and click **Continue**.

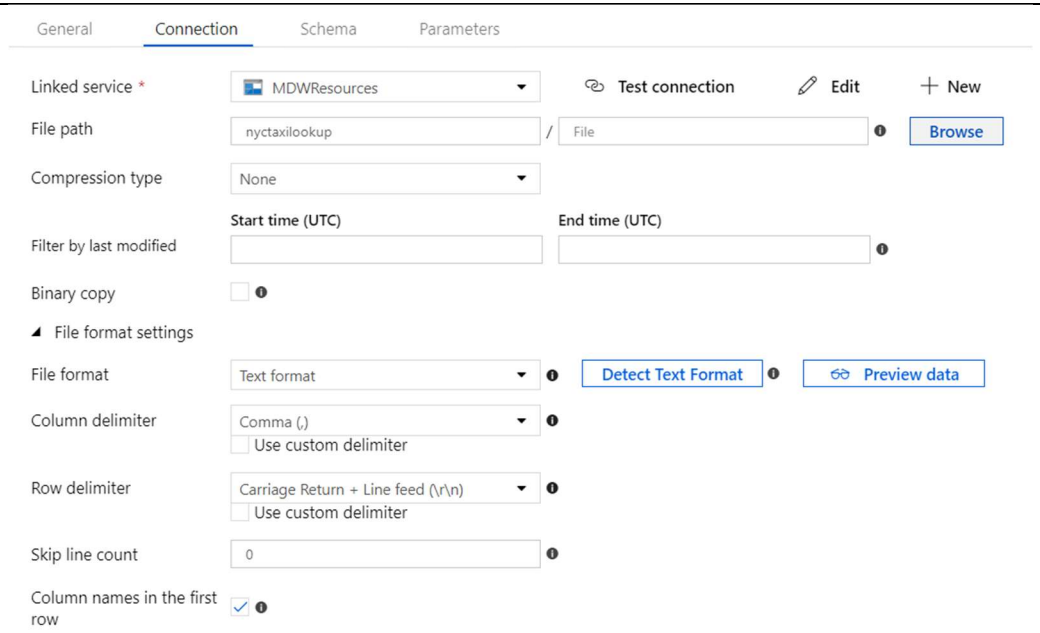


9. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWDataLake_NYCTaxiData
    b. **Connection > Linked Service:** MDWDataLake
    c. **Connection > File Path:** nyctaxidata
    d. **Connection > Binary Copy:** Unchecked
    e. **Connection > Column names in the first row:** Checked
10. Leave remaining fields with default values.

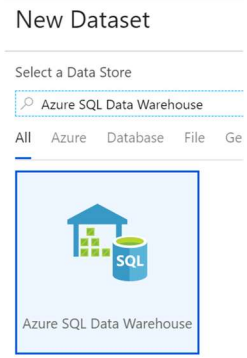


11. Repeat the process to create another dataset, this time referencing the NYCTaxiLookup container in your MDWResources storage account. 12. Type “Azure Blob Storage” in the search box and select **Azure Blob Storage.** Click **Continue.**	
13. On the Select Format blade, select **Binary** and click **Continue**.	

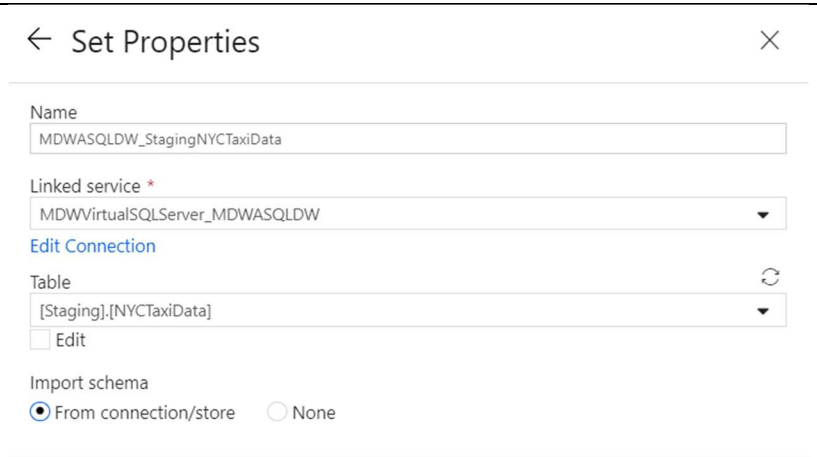
14. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWResources_NYCTaxiLookup
    b. **Connection > Linked Service:** MDWResources
    c. **Connection > File Path:** nyctaxilookup / taxi_zone_lookup.csv
    d. **Connection > Binary Copy:** Unchecked
    e. **Connection > Column names in the first row:** Checked.
    f. **Connection > Quote character:** " (double-quote)
15. Leave remaining fields with default values.



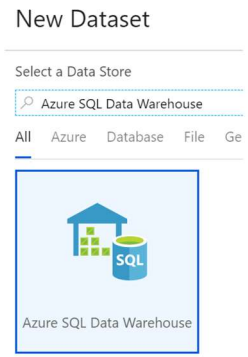
16. Repeat the process to create another dataset, this time referencing the Staging.NYCTaxiData in your Azure SQL Data Warehouse database.
17. Type "Azure SQL Data Warehouse" in the search box and select **Azure SQL Data Warehouse.** Click **Continue.**



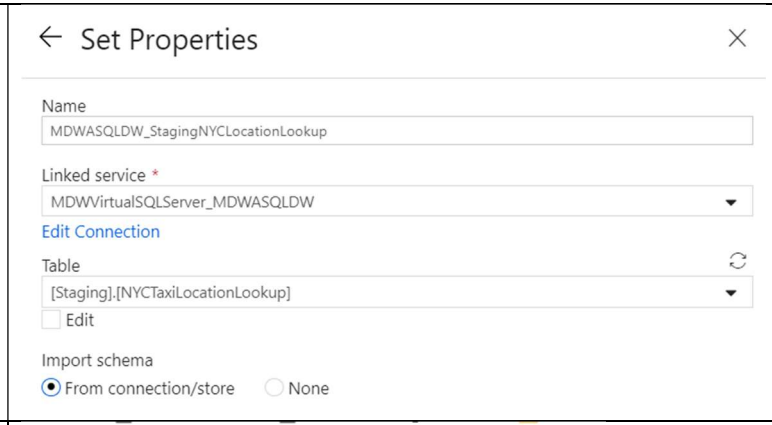
18. On the **Set Properties** blade, enter the following details:
    a. **Name**: MDWASQLDW_StagingNYCTaxiData
    b. **Linked Service:** MDWSQLVirtualServer_MDWASQLDW
    c. **Table:** [Staging].[NYCTaxiData]
19. Leave remaining fields with default values.



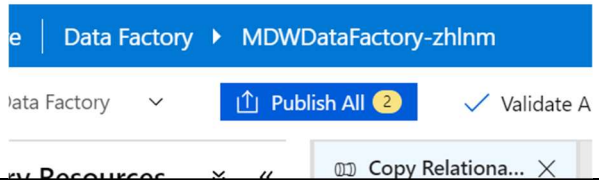
20. Repeat the process to create another dataset, this time referencing the Staging.NYCLocationLookup in your Azure SQL Data Warehouse database.
21. Type “Azure SQL Data Warehouse” in the search box and select **Azure SQL Data Warehouse.** Click **Finish.**



22. On the **Set Properties** blade, enter the following details:
    a. **Name**: MDWASQLDW_StagingNYCLocationLookup
    b. **Linked Service:** MDWSQLVirtualServer_MDWASQLDW
    c. **Table:** [Staging].[NYCTaxiLocationLookup]
23. Leave remaining fields with default values.



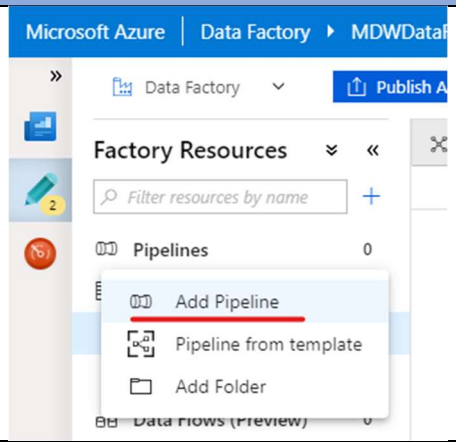
24. Publish your dataset changes by clicking the **Publish all** button.



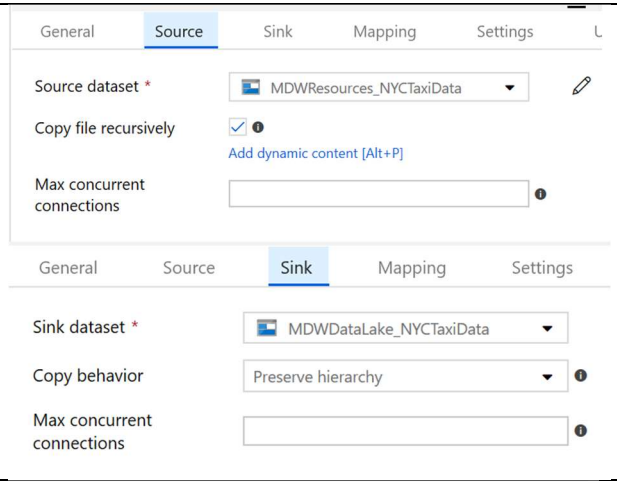
Create and Execute Pipeline

IMPORTANT: Execute these steps on your host computer

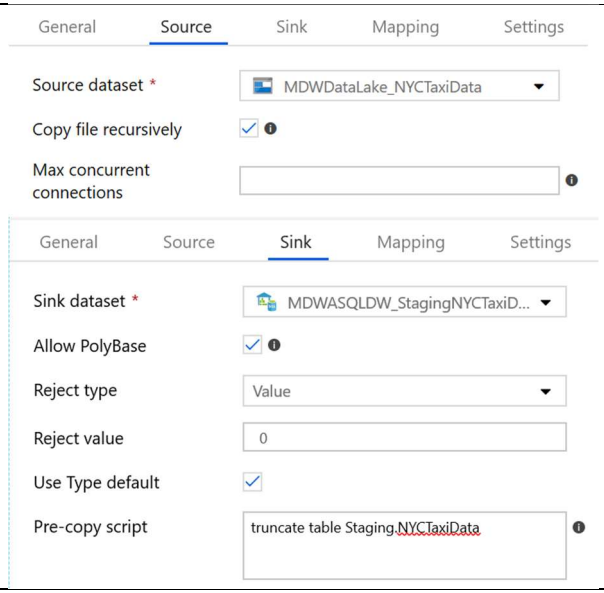
1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel. Under **Factory Resources** tab, click the ellipsis (**…**) next to **Pipelines** and then click **Add Pipeline** to create a new dataset.
2. On the New Pipeline tab, enter the following details:
    a. **General > Name**: Copy Taxi Data
3. Leave remaining fields with default values.



4. From the Activities panel, type “Copy Data” in the search box. Drag the Copy Data activity on to the design surface. This copy activity will copy data files from MDWResources to MDWDatalake.
5. Select the Copy Data activity and enter the following details:
    a. **General > Name:** CopyTaxiDataFiles
    b. **Source > Source dataset:** MDWResources_NYCTaxiData
    c. **Sink > Sink dataset:** MDWDataLake_NYCTaxiData
    d. **Sink > Copy Behavior:** Preserve Hierarchy
6. Leave remaining fields with default values.

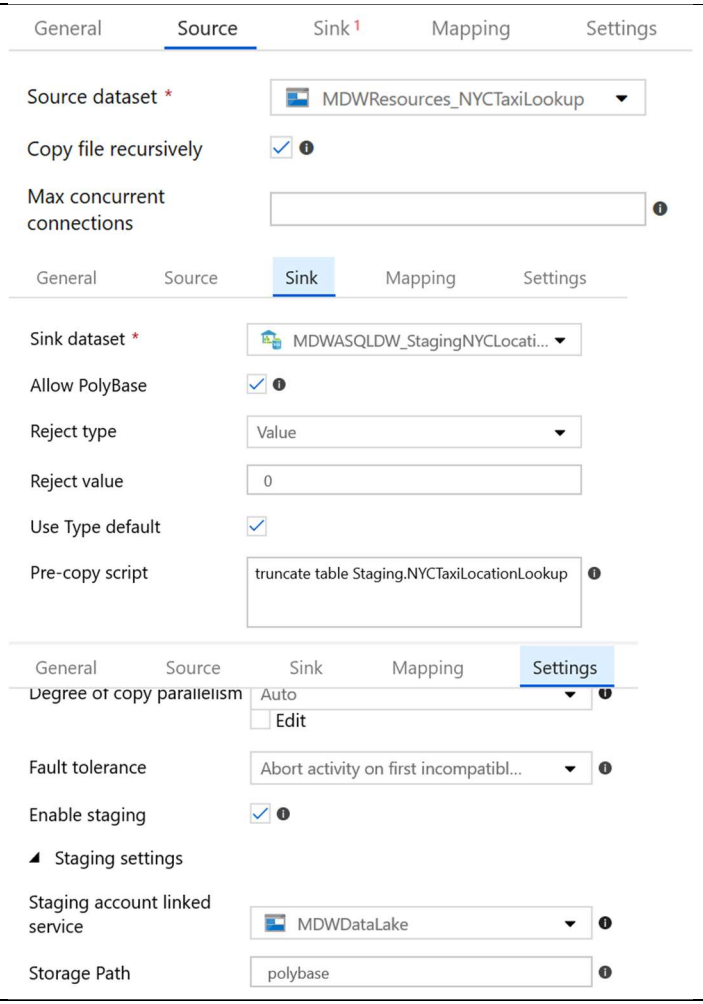


7. Repeat the process to create another Copy Data Activity, this time to copy data from the files in your data lake to your SQL Data Warehouse.
8. From the Activities panel, type “Copy Data” in the search box. Drag the Copy Data activity on to the design surface.
9. Select the Copy Data activity and enter the following details:
    a. **General > Name:** CopyTaxiDataToDW
    b. **Source > Source dataset:** MDWDataLake_NYCTaxiData
    c. **Sink > Sink dataset:** MDWASQLDW_StagingNYCTaxiData
    d. **Sink > Pre Copy Script:** truncate table Staging.NYCTaxiData
10. Leave remaining fields with default values.



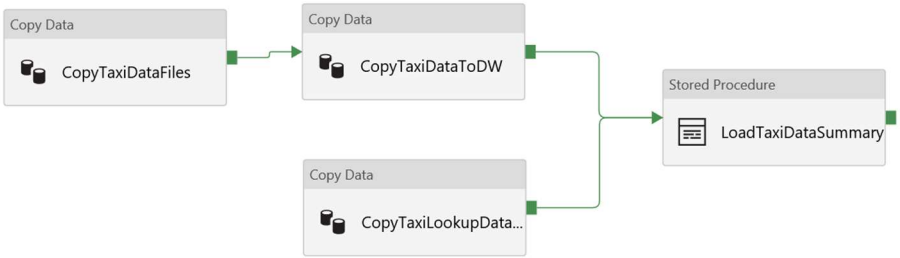
11. Create a **Success** (green) precedence constraint between CopyTaxiDataFiles and CopyTaxiDataToDW. You can do it by dragging the green connector from CopyTaxiDataFiles and landing the arrow onto CopyTaxiDataToDW.

12. Repeat the process to create another Copy Data Activity, this time to copy taxi location lookup data from MDWResources to your SQL Data Warehouse.
13. From the Activities panel, type “Copy Data” in the search box. Drag the Copy Data activity on to the design surface.
14. Select the Copy Data activity and enter the following details:
    a. **General > Name:** CopyTaxiLookupDataToDW
    b. **Source > Source dataset:** MDWResources_NYCTaxiLookup
    c. **Sink > Sink dataset:** MDWASQLDW_StagingNYCLocationLookup
    d. **Sink > Pre Copy Script:** truncate table Staging.NYCTaxiLocationLookup
    e. **Settings > Enable staging:** Checked
    f. **Settings > Staging account linked service:** MDWDataLake
    g. **Settings > Storage Path:** polybase
15. Leave remaining fields with default values.

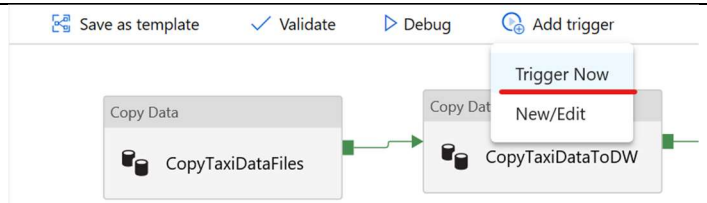
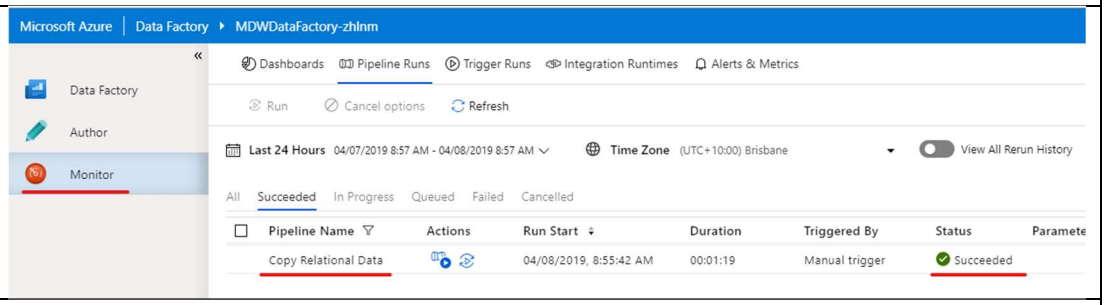
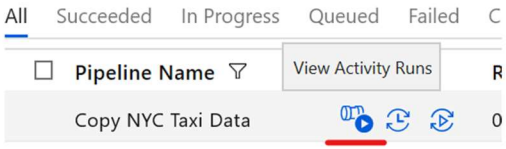


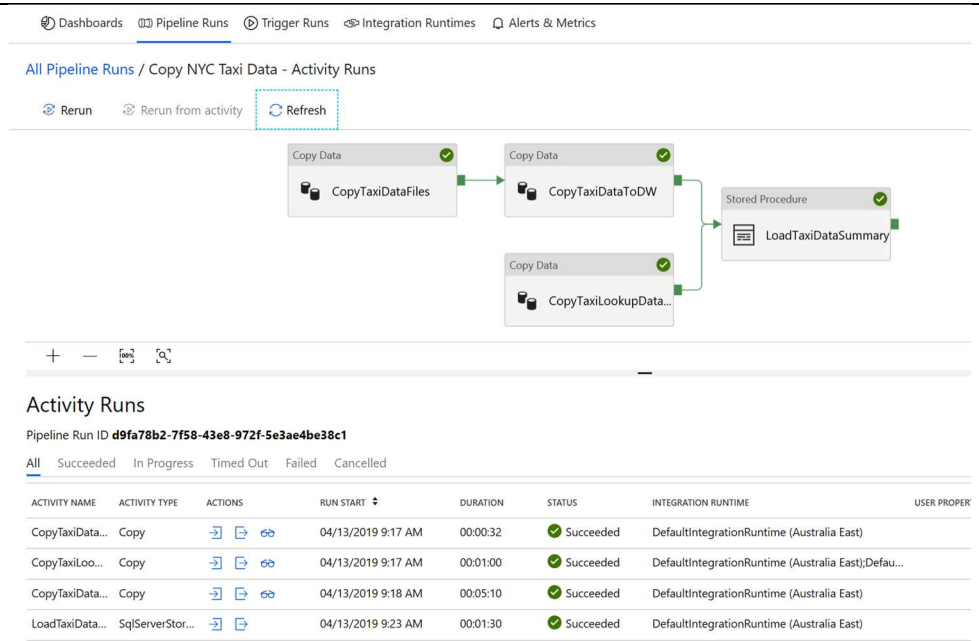
16. From the Activities panel, type “Stored Procedure” in the search box. Drag the **Stored Procedure** activity onto the design surface. This activity will execute the Staging.spNYCLoadTaxiDataSummary to generate aggregated taxi ride information.
17. Select the Stored Procedure activity and enter the following details:
    a. **General > Name:** LoadTaxiDataSummary
    b. **SQL account > Linked service:** MDWSQLVirtualServer_MDWASQLDW
    c. **Stored procedure > Stored procedure name:** [Staging].[spNYCLoadTaxiDataSummary]
18. Leave remaining fields with default values.

19. Create **Success** (green) from CopyDataToDW to LoadTaxiDataSummary and from CopyTaxiLookupData to LoadTaxiDataSummary.



20. Publish your pipeline changes by clicking the **Publish all** button.

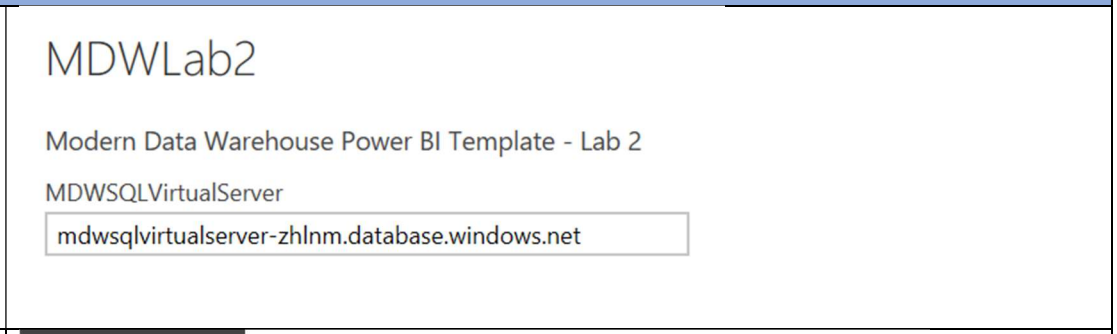
21. To execute the pipeline, click on **Add trigger** menu and then **Trigger Now**. 22. On the **Pipeline Run** blade, click **Finish**.	
23. To monitor the execution of your pipeline, click on the **Monitor** menu on the left-hand side panel. 24. You should be able to see the **Status** of your pipeline execution on the right-hand side panel.	
25. Click the **View Activity Runs** button for detailed information about each activity execution in the pipeline. The whole execution should last between 7-8 minutes.	


Dashboards
Pipeline Runs
Trigger Runs
Integration Runtimes
Alerts & Metrics
All Pipeline Runs / Copy NYC Taxi Data - Activity Runs
Rerun
Rerun from activity
Refresh
Copy Data
CopyTaxiDataFiles
Copy Data
CopyTaxiDataToDW
Stored Procedure
LoadTaxiDataSummary
Copy Data
CopyTaxiLookupData...
Activity Runs
Pipeline Run ID d9fa78b2-7f58-43e8-972f-5e3ae4be38c1
All
Succeeded
In Progress
Timed Out
Failed
Cancelled
ACTIVITY NAME
ACTIVITY TYPE
ACTIONS
RUN START
DURATION
STATUS
INTEGRATION RUNTIME
USER PROPER
CopyTaxiData... Copy 04/13/2019 9:17 AM 00:00:32 Succeeded DefaultIntegrationRuntime (Australia East)
CopyTaxiLoo... Copy 04/13/2019 9:17 AM 00:01:00 Succeeded DefaultIntegrationRuntime (Australia East);Defau...
CopyTaxiData... Copy 04/13/2019 9:18 AM 00:05:10 Succeeded DefaultIntegrationRuntime (Australia East)
LoadTaxiData... SqlServerStor... 04/13/2019 9:23 AM 00:01:30 Succeeded DefaultIntegrationRuntime (Australia East)

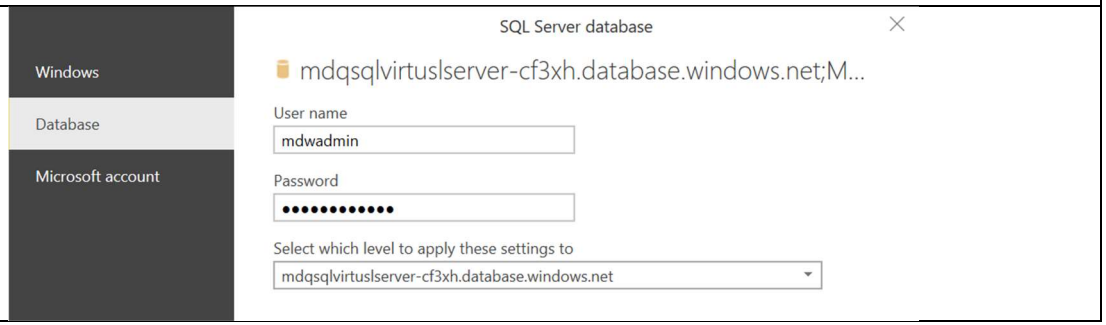
Visualize Data with Power BI

In this section you are going to use Power BI to visualize data from Azure SQL Data Warehouse. The Power BI report will use an Import connection to query Azure SQL Data Warehouse and visualise Motor Vehicle Collision data from the table you loaded in the previous exercise.

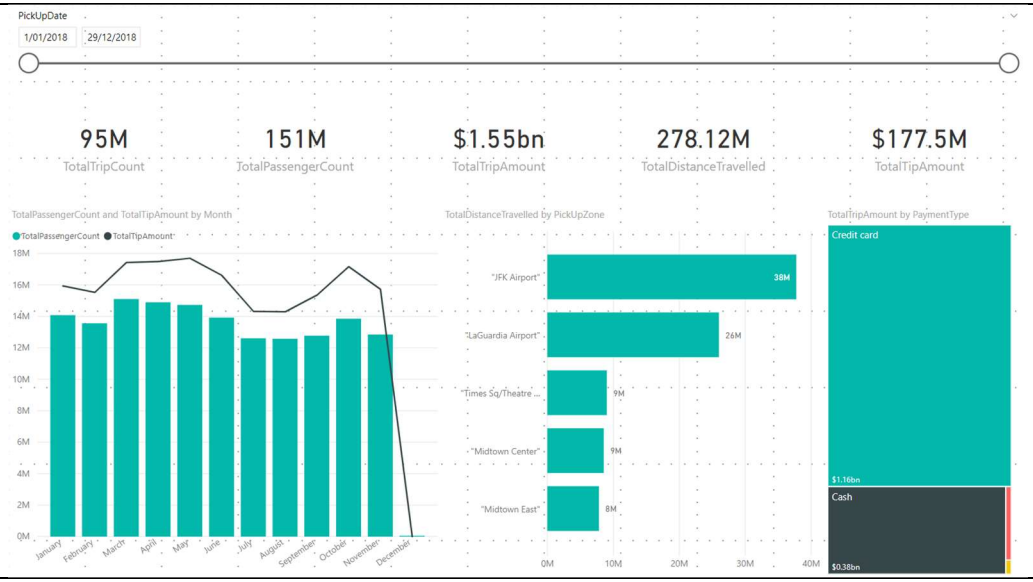
<table>
<tr><th colspan="2">IMPORTANT: Steps to be executed on MDWDesktop</th></tr>
<tr><td>
<ol start="8">
On MDWDesktop, download the Power BI report from the link https://aka.ms/MDWLab2 and save it in the Desktop.
Open the file MDWLab2.pbit with Power BI Desktop.
When prompted to enter the value of the MDWSQLVirtualServer parameter, type the full server name: mdwsqlvirtualserver-suffix.database.windows.net
Click Load.

</td><td> 
</td></tr>
<tr><td>
<ol start="12">
When prompted to enter credentials, select Database from the left-hand side panel and enter the following details:
<ol type="a">
User name: mdwadmin
Password: P@ssw0rd123!

Leave remaining fields with their default values.
Click Connect.

</td><td> 
</td></tr>
</table>

15. Once data finish loading interact with the report by changing the PickUpDate slicer and by clicking on the other visualisations.
16. Save your work and close Power BI Desktop.



Lab 3: Explore Big Data using Azure Databricks

In this lab you will use Azure Databricks to explore the New York Taxi data files you saved in your data lake in the previous exercise. Using a Databricks you will connect to the data lake and query taxi ride details.

Lab Architecture

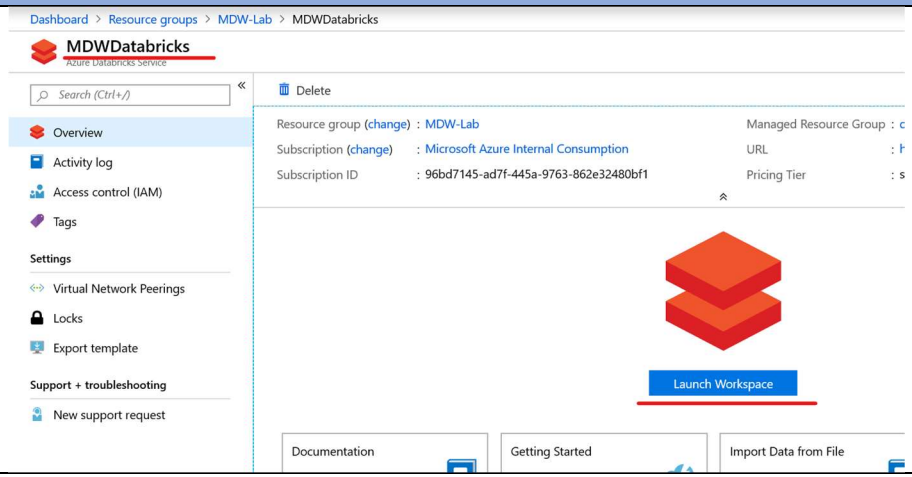
Build an Azure Databricks notebook to explore the data files you saved in your data lake in the previous exercise. You will use Python and SQL commands to open a connection to your data lake and query data from data files.

Create Azure Databricks Cluster

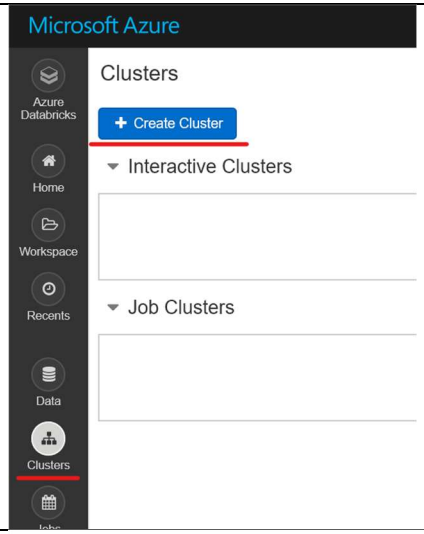
In this section you are going to create an Azure Databricks cluster that will be used to execute notebooks.

IMPORTANT: Execute these steps on your host computer

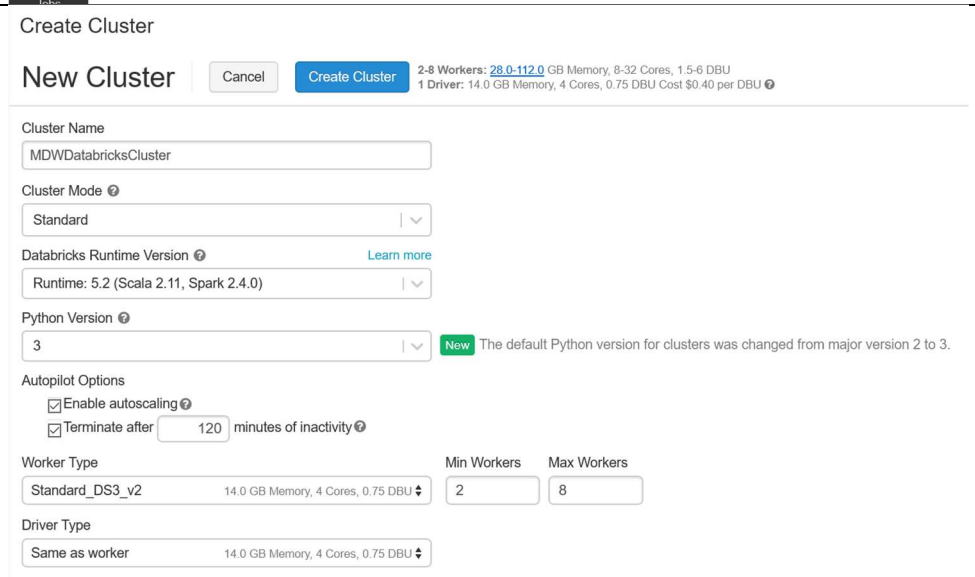
1. In the Azure Portal, navigate to the MDW-Lab resource group and locate the Azure Databricks MDWDatabricks.
2. On the **MDWDatabricks** blade, click the **Launch Workspace** button. The Azure Databricks portal will open on a new browser tab.



3. On the **Azure Databricks** portal, click the **Clusters** button on the left-hand side menu.
4. On the **Clusters** blade, click **+ Create Cluster**.



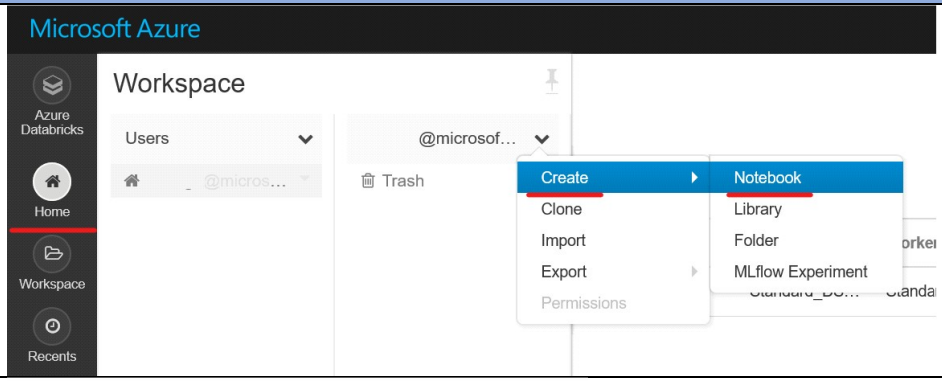
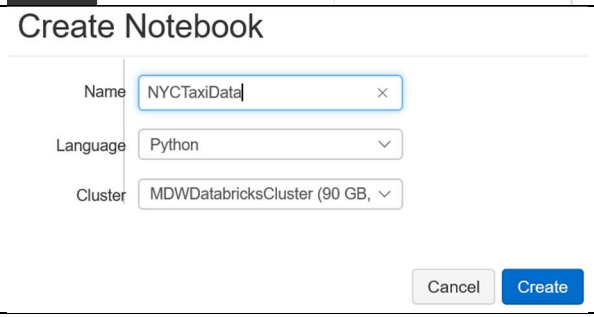
5. On the Create Cluster blade, type "MDWDatabricksCluster" in the **Cluster Name** field. Leave all other fields with their default values.
6. Click **Create Cluster**. It should take around 5 minutes for the cluster to be fully operational.

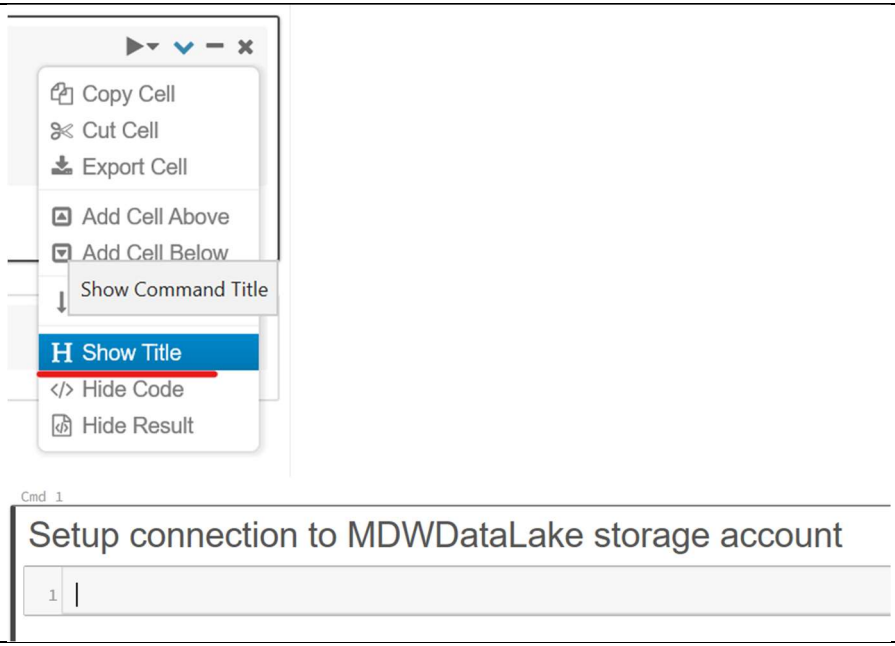


Create an Azure Databricks Notebook

In this section you are going to create an Azure Databricks notebook that will be used to explore the taxi data files you copied to your data lake in Lab 2.

IMPORTANT: The full solution scripts for each notebook command are saved in the Student's Computer path **C:\ADSIAD\Lab\Lab3\Solution**

IMPORTANT: Execute these steps on your host computer	
1. On the **Azure Databricks** portal, click the **Home** button on the left-hand side menu. 2. On the **Workspace** blade, click the down arrow next to your user name and then click **Create > Notebook**.	
3. On the Create Notebook pop-up window type "NYCTaxiData" in the **Name** field. 4. Ensure you have the **Language** field set to Python and the **Cluster** field is set to MDWDatabricksCluster. 5. Click **Create**.	

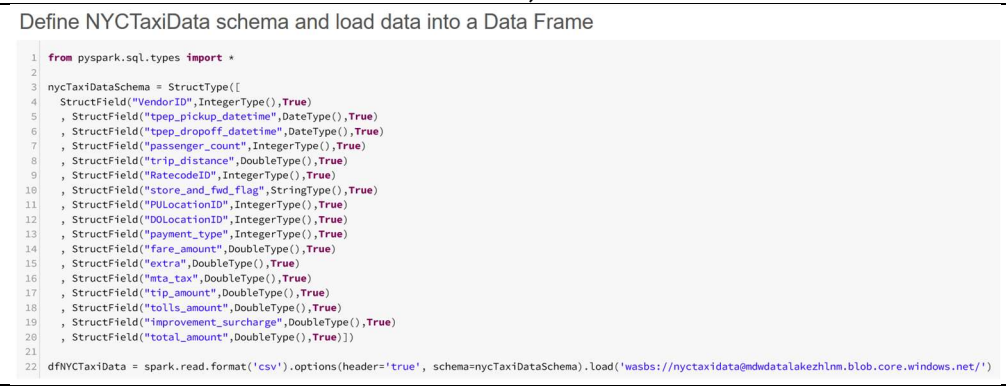
6. On the **Cmd 1** cell, click the Edit button on the top right-hand corner of the cell and then click **Show Title**. 7. Type “Setup connection to MDWDataLake storage account” in the cell title.	
8. On the **Cmd 1** cell, you will invoke the Spark API to establish a connection to your MDWDalaLake storage account. For this you will need to retrieve the name and key of your MDWDataLake storage account from the Azure Portal. 9. Remember to replace <your MDWDataLake storage account name> with **mdwdatalake*suffix*** and to replate <your MDWDataLake storage account key> with the storage account key.	**Python command:** `spark.conf.set(` `  "fs.azure.account.key.<your MDWDataLake storage account name>.blob.core.windows.net",` `  "<your MDWDataLake storage account key>")`
10. Press Shift + Enter to execute and create a new notebook cell. Set the title of the Cmd 2 cell to “Define NYCTaxiData schema and load data into a Data Frame”	

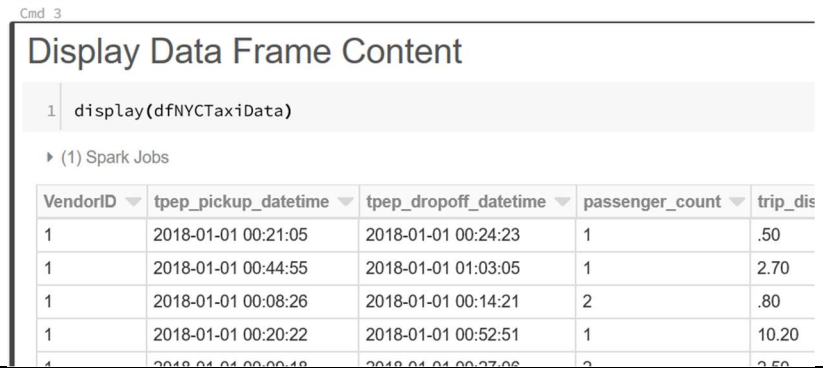
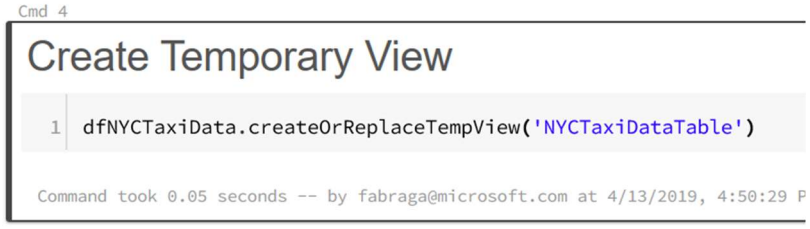
11. In the **Cmd 2** cell, define a new StructType object that will contain the definition of the data frame schema.
12. Using the schema defined above, initialise a new data frame by invoking the Spark API to read the contents of the nyctaxidata container in the MDWDataLake storage account.
13. Remember to replace <your MDWDataLake storage account name> with **mdwdatalake*suffix*** and to replate <your MDWDataLake storage account key> with the storage account key.

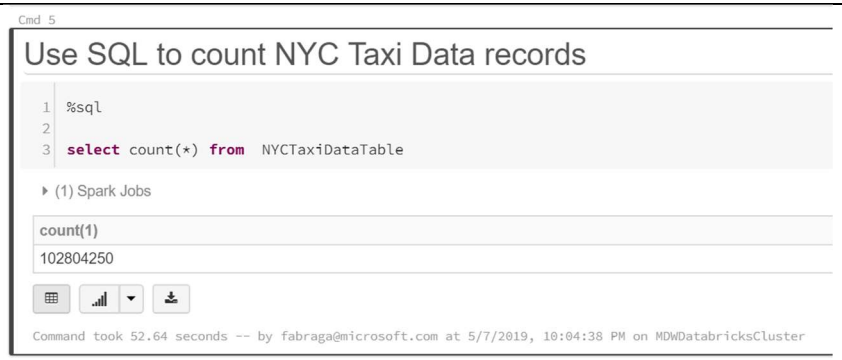
Python command:

```
from pyspark.sql.types import *

nycTaxiDataSchema = StructType([
  StructField("VendorID",IntegerType(),True)
  , StructField("tpep_pickup_datetime",DateType(),True)
  , StructField("tpep_dropoff_datetime",DateType(),True)
  , StructField("passenger_count",IntegerType(),True)
  , StructField("trip_distance",DoubleType(),True)
  , StructField("RatecodeID",IntegerType(),True)
  , StructField("store_and_fwd_flag",StringType(),True)
  , StructField("PULocationID",IntegerType(),True)
  , StructField("DOLocationID",IntegerType(),True)
  , StructField("payment_type",IntegerType(),True)
  , StructField("fare_amount",DoubleType(),True)
  , StructField("extra",DoubleType(),True)
  , StructField("mta_tax",DoubleType(),True)
  , StructField("tip_amount",DoubleType(),True)
  , StructField("tolls_amount",DoubleType(),True)
  , StructField("improvement_surcharge",DoubleType(),True)
  , StructField("total_amount",DoubleType(),True)])

dfNYCTaxiData = spark.read.format('csv').options(header='true',
schema=nycTaxiDataSchema).load('wasbs://nyctaxidata@<your MDWDataLake storage
account name>.blob.core.windows.net/')
```

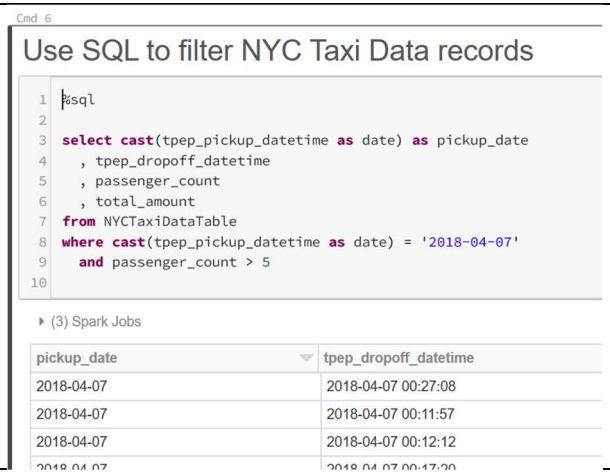
14. Hit Shift + Enter to execute the command and create a new cell.



15. Set the title of the **Cmd 3** cell to “Display Data Frame Content”.	
16. In the **Cmd 3** cell, call the **display** function to show the contents of the data frame dfNYCTaxiData.	**Python command:** `display(dfNYCTaxiData)`
17. Hit Shift + Enter to execute the command and create a new cell.	
18. Set the title of the **Cmd 4** cell to “Create Temp View”	
19. In the **Cmd 4** cell, call the createOrReplaceTempView method of the data frame object to create a temporary view of the data in memory.	**Python command:** `dfNYCTaxiData.createOrReplaceTempView('NYCTaxiDataTable')`
20. Hit Shift + Enter to execute the command and create a new cell.	
21. Set the title of the **Cmd 5** cell to “Use SQL to count NYC Taxi Data records”	
22. In the **Cmd 5** cell, change the default language to SQL using the %sql command. 23. Write a SQL query to retrieve the total number of records in the NYCTaxiDataTable view.	**Command:** `%sql` `select count(*) from NYCTaxiDataTable`

24. Hit Shift + Enter to execute the command and create a new cell.	
25. Set the title of the **Cmd 6** cell to "Use SQL to filter NYC Taxi Data records"	
26. In the **Cmd 6** cell, write a SQL query to filter taxi rides that happened on the Apr, 7th 2018 that had more than 5 passengers.	**Command** `%sql` `select cast(tpep_pickup_datetime as date) as pickup_date` `  , tpep_dropoff_datetime` `  , passenger_count` `  , total_amount` `from NYCTaxiDataTable` `where cast(tpep_pickup_datetime as date) = '2018-04-07'` `  and passenger_count > 5`

27. Hit Shift + Enter to execute the command.



28. Set the title of the **Cmd 7** cell to "Use SQL to aggregate NYC Taxi Data records and visualize data"

29. In the **Cmd 7** cell, write a SQL query to aggregate records and return total number of rides by payment type. Use the following command.
30. Hit Ctrl + Enter to execute the command. Results will be displayed in a grid in the cell.

Command

```
%sql

select case payment_type
            when 1 then 'Credit card'
            when 2 then 'Cash'
            when 3 then 'No charge'
            when 4 then 'Dispute'
            when 5 then 'Unknown'
            when 6 then 'Voided trip'
        end as PaymentType
  , count(*) as TotalRideCount
from NYCTaxiDataTable
group by payment_type
order by TotalRideCount desc
```

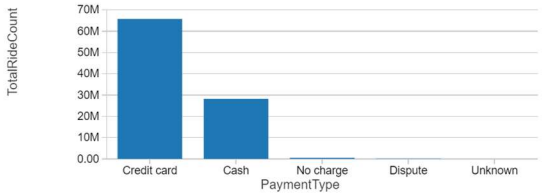
31. Click the **Bar chart** button to see results as a bar chart.

Cmd 7

Use SQL to aggregate NYC Taxi Data records and visualize data

```
%sql

select case payment_type
            when 1 then 'Credit card'
            when 2 then 'Cash'
            when 3 then 'No charge'
            when 4 then 'Dispute'
            when 5 then 'Unknown'
            when 6 then 'Voided trip'
        end as PaymentType
  , count(*) as TotalRideCount
from NYCTaxiDataTable
group by payment_type
order by TotalRideCount desc
```

▸ (1) Spark Jobs



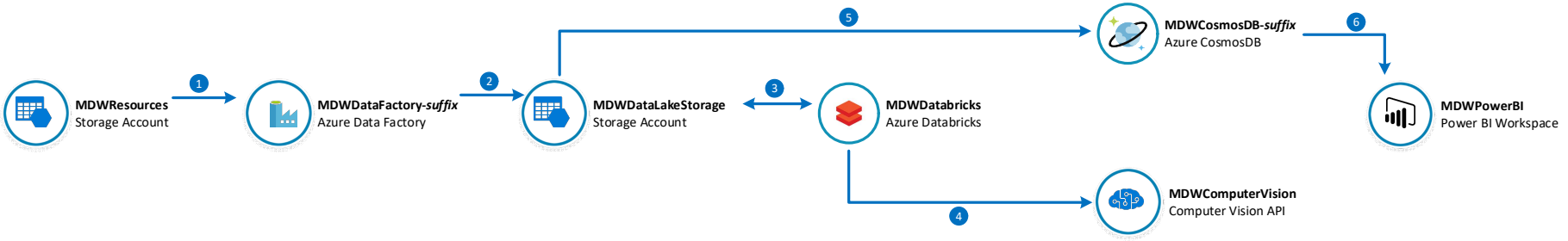
Plot Options...

Command took 56.37 seconds -- by fabraga@microsoft.com at 4/14/2019, 2:11:36 PM on MDWDatabricksCluster

Lab 4: Add AI to your Big Data Pipeline with Cognitive Services

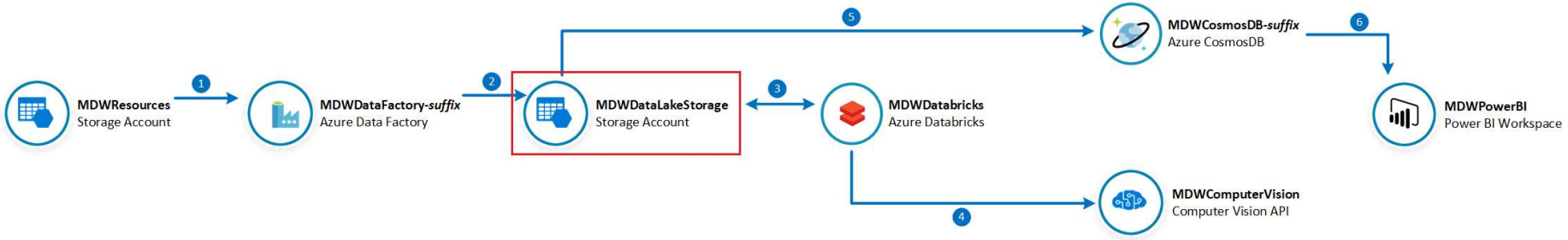
In this lab you will use Azure Data Factory to download New York City images to your data lake. Then, as part of the same pipeline, you are going to use an Azure Databricks notebook to invoke Computer Vision Cognitive Service to generate metadata documents and save them in a CosmosDB database. You will use Power BI to visualise the images and their AI-generated metadata.

Lab Architecture

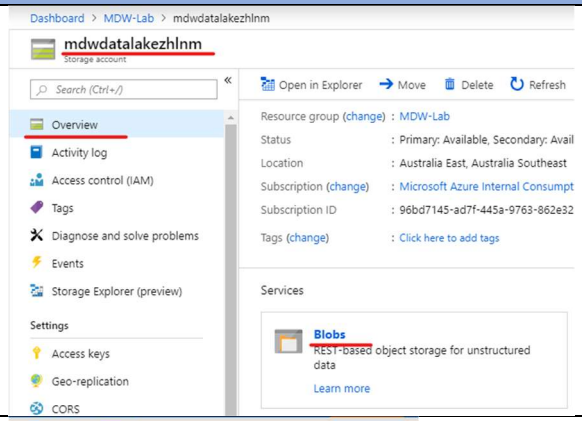
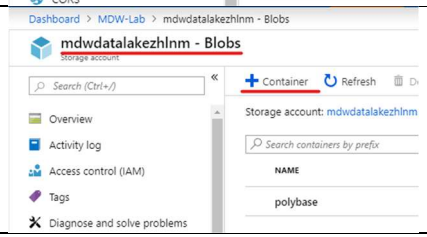


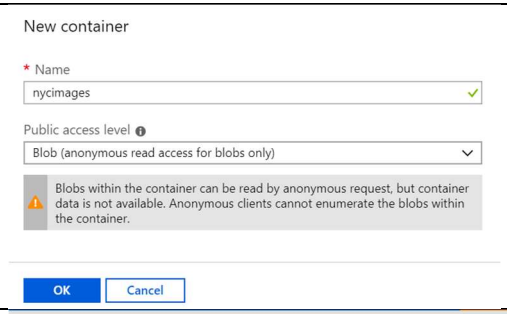
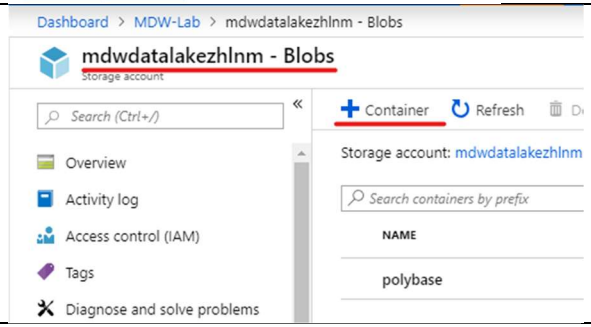
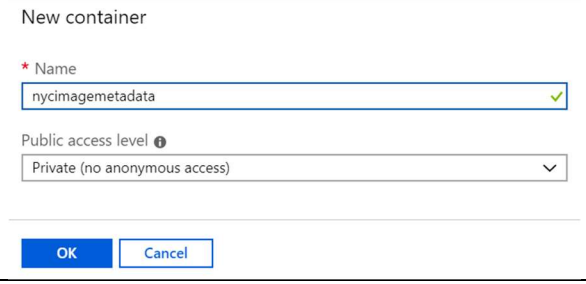
1. Build an Azure Data Factory Pipeline to copy image files from shared Azure Storage
2. Save image files to your data lake
3. For each image in your data lake, invoke an Azure Databricks notebook that will take the image URL as parameter
4. For each image call the Azure Computer Vision Cognitive service to generate image metadata. Metadata files are saved back in your data lake
5. Copy metadata JSON documents into your Cosmos DB database
6. Visualize images and associated metadata using Power BI

Create NYCImages and NYCImageMetadata Containers in Azure Blob Storage



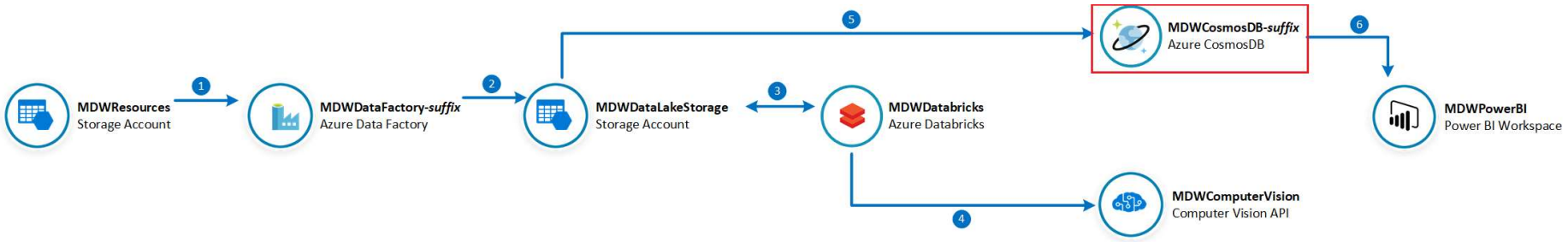
In this section you will create a container in your MDWDataLake that will be used as a repository for the NYC image files. You will copy 30 files from the MDWResources Storage Account into your NYCTaxiData container.

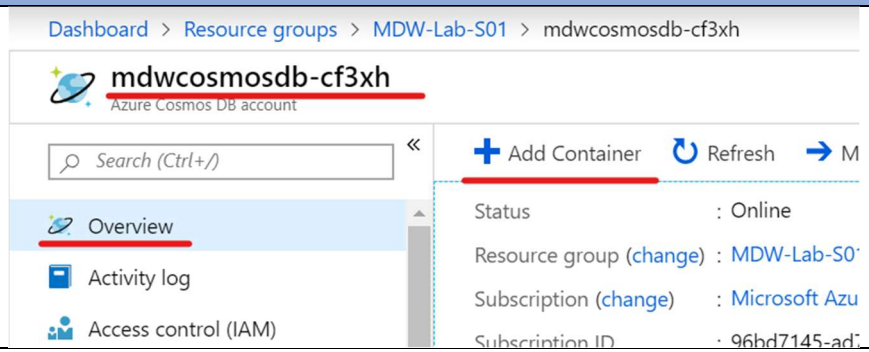
IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, go to the lab resource group and locate the Azure Storage account **mdwdatalake*suffix***. 2. On the **Overview** panel, click **Blobs**.	
3. On the **mdwdalalake*suffix* – Blobs** blade, click **+ Container**.	

4. On the **New container** blade, enter the following details: a. **Name**: nycimages b. **Public access level**: Blob (anonymous read access for blobs only) 5. Click **OK** to create the new container.	
6. Repeat the process to create the NYCImageMetadata container. This container will be used to host the metadata files generated by Cognitive Services before they can be saved in CosmosDB.	
7. On the **New container** blade, enter the following details: a. **Name**: nycimagemetadata b. **Public access level**: Private (no anonymous access) 8. Click **OK** to create the new container.	

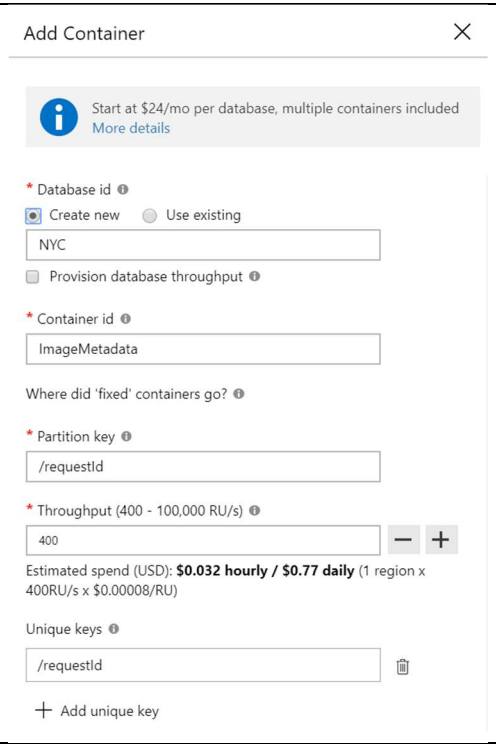
Create CosmosDB database and collection

In this section you will create a CosmosDB database called NYC and a collection called ImageMetadata that will host New York image metadata information.



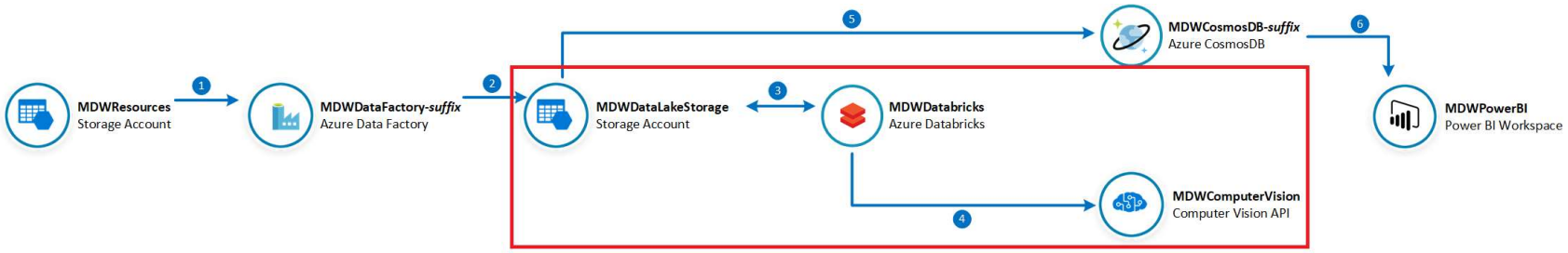
IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, go to the lab resource group and locate the CosmosDB account **MDWCosmosDB-*suffix***. 2. On the **Overview** panel, click **+ Add Container**.	

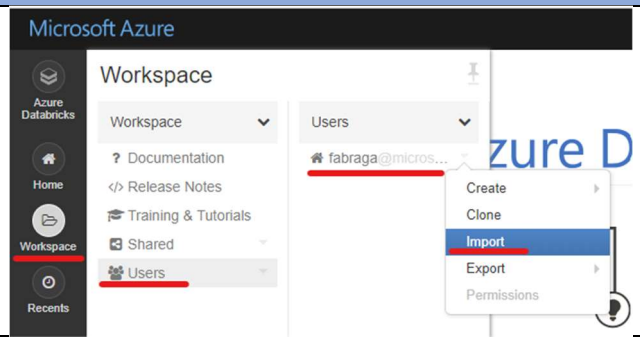
3. On the **Add Container** blade, enter the following details:
    a. **Database id > Create new**: NYC
    b. **Container id:** ImageMetadata
    c. **Partition key:** /requestId
    d. **Throughput:** 400
    e. **Unique keys:** /requestId
4. Click **OK** to create the container.



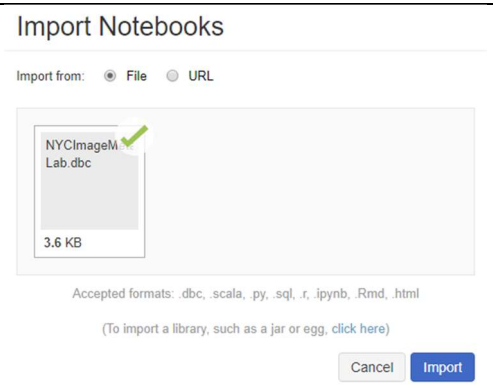
Import Databricks Notebook to Invoke Computer Vision Cognitive Services API

In this section you will import a Databricks notebook to your workspace and fill out the missing details about your Computer Vision API and your Data Lake account. This notebook will be executed from an Azure Data Factory pipeline and it will invoke the Computer Vision API to generate metadata about the images and save the result back to your data lake.

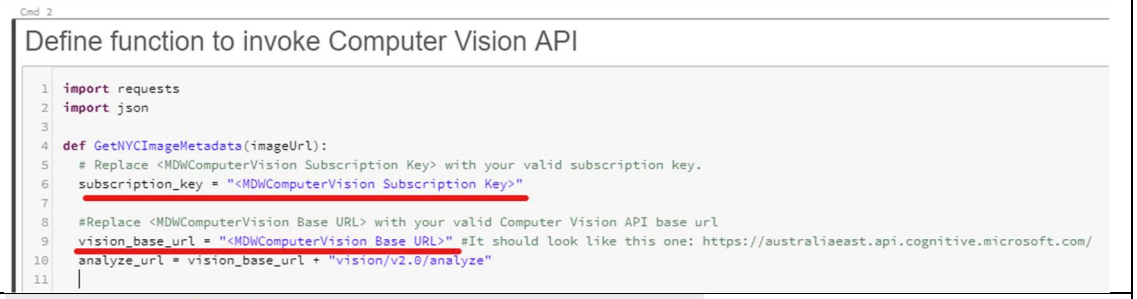


IMPORTANT: Execute these steps on your host computer	
1. On the **Azure Databricks** portal, click the **Workspace** button on the left-hand side menu. 2. On the **Workspace** blade, click **Users**. 3. On the **Users** blade, click the arrow next to your user name and then **Import**.	

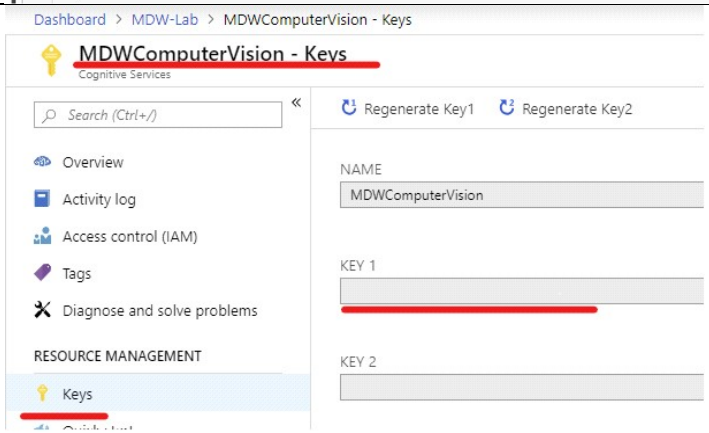
4. On the Import Notebooks pop up window, click the browse link to import the notebook file **C:\ADSIAD\Lab\Lab4\NYCImageMetadata-Lab.dbc**
5. Click **Import**.

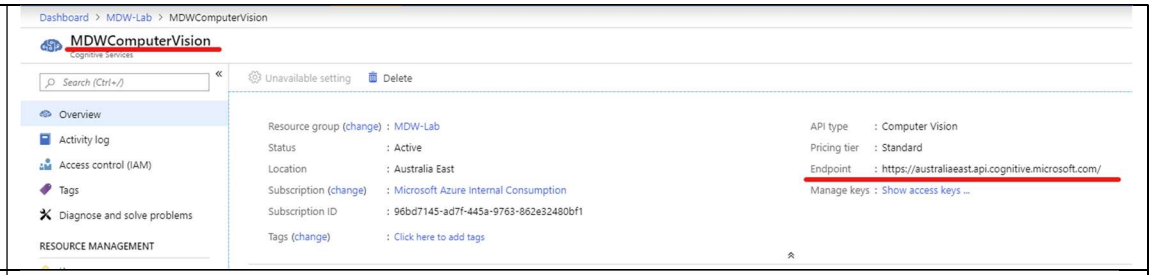
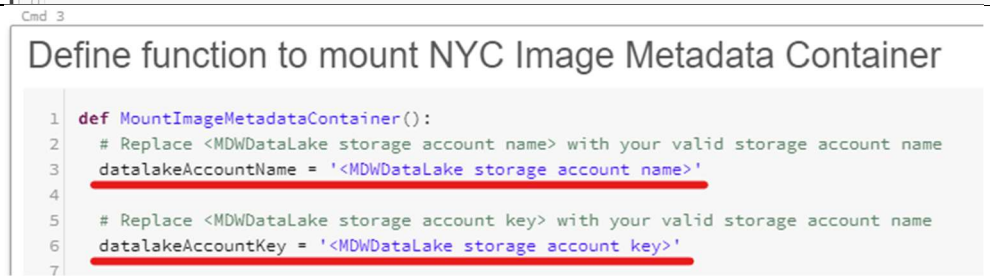
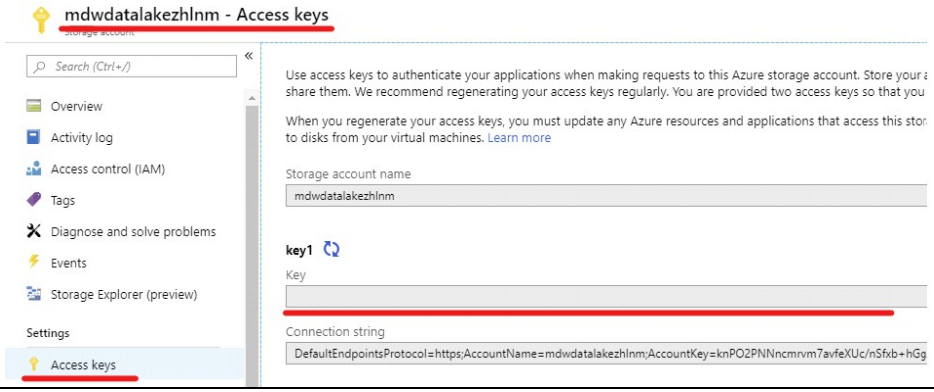


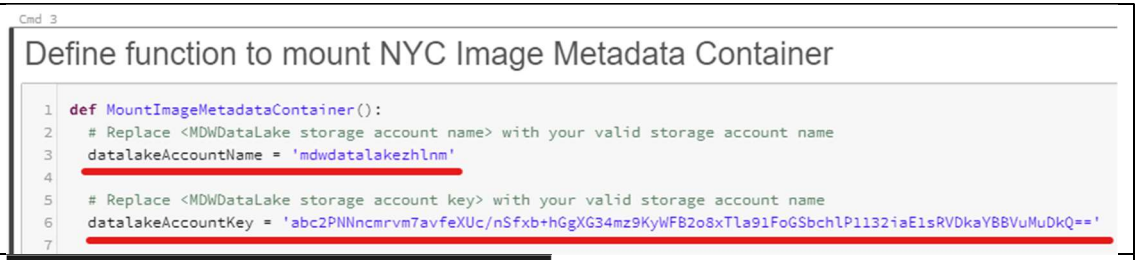
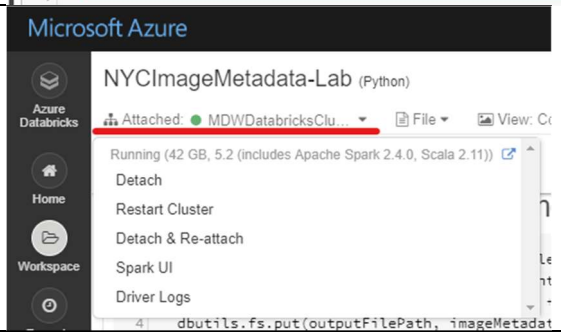
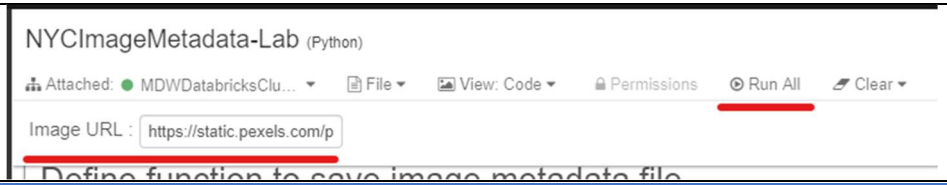
6. On the NYCImageMetadata-Lab notebook, go to the cell **Cmd 2 – Define function to invoke Computer Vision API.** You will need to change the function code to include the Computer Vision API details.



7. From the **Azure Portal**, retrieve the **MDWComputerVision** subscription key and base endpoint URL.

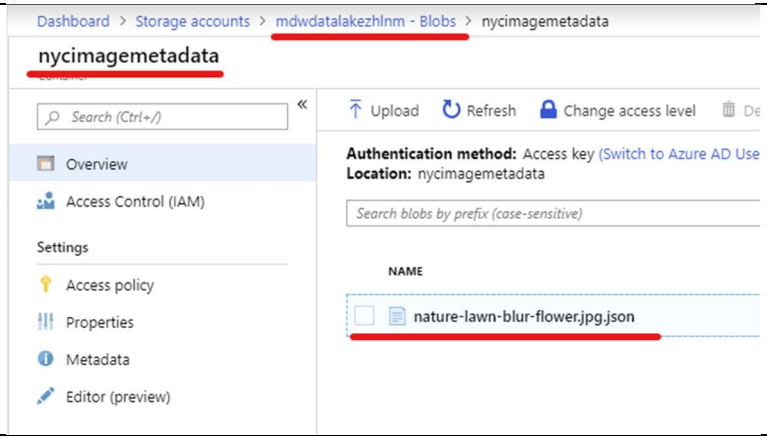


	
8. Copy and paste the Key and Endpoint values back in the Databricks notebook.	
9. On the NYCImageMetadata-Lab notebook, go to the cell **Cmd 3 – Define function to mount NYC Image Metadata Container.** You will need to change the function code to include your data lake storage account details. 10. In the dataLakeaccountName variable assignment replace **<MDWDataLake storage account name>** with **mdwdatalake*suffix*.**	
11. From the **Azure Portal**, retrieve the **MDWDataLake*suffix*** access key.	

12. Copy and paste the Access Key Databricks notebook. Replace **<MDWDataLake storage account key>** with the Access Key you got from the previous step.	
13. Attach the notebook to your previously created MDWDatabricksCluster cluster.	
14. Review the notebook code. 15. If you want to test it, you can copy any publicly available image URL and paste it in the **Image URL** notebook parameter. You can use any of the following image URLs in the list as examples.	**Test Image URLs:** https://petlifetoday.com/wp-content/uploads/2018/06/wireless-dog-fence.jpg https://static.pexels.com/photos/4204/nature-lawn-blur-flower.jpg https://image.redbull.com/rbcom/052/2017-05-22/89eef344-d24f-4520-8680-8b8f7508b264/0012/0/0/0/2428/3642/800/1/best-beginner-motocross-bikes-ktm-250-sx-f.jpg http://www.kiplinger.com/slideshow/investing/T024-S001-the-best-emerging-markets-stocks-for-2019/images/intro.jpg
16. Click **Run All** to execute the notebook.	

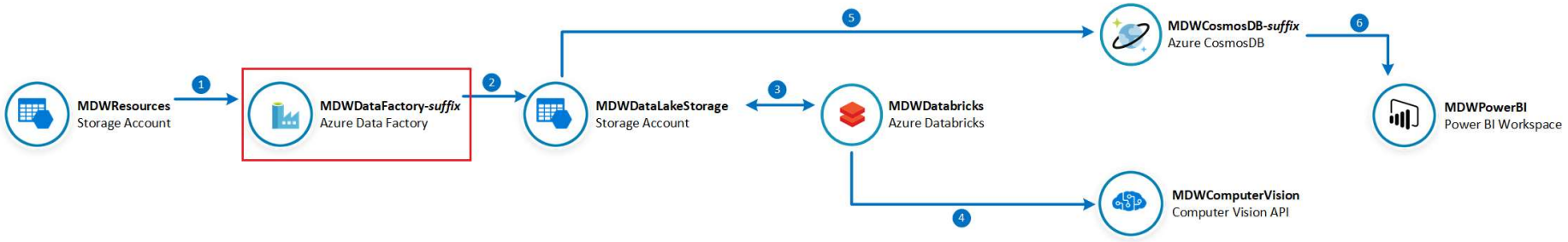
17. After a successful execution you will notice that a new JSON file has been saved in the NYCImageMetadata container in your Data Lake.

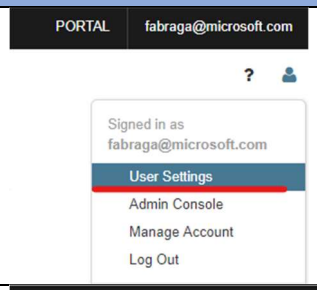
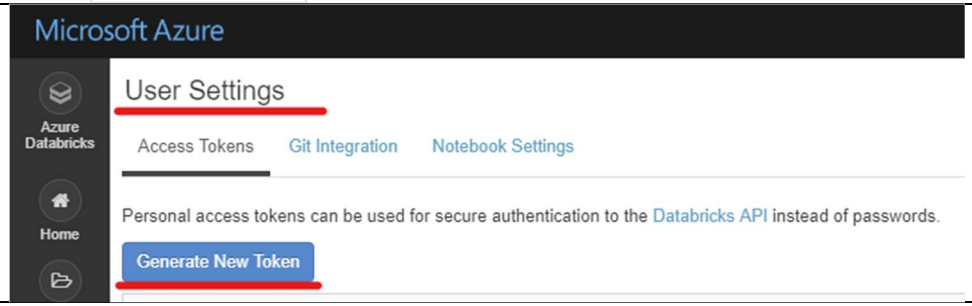
18. Navigate to Azure Portal and check the contents of the **nycimagemetadata** container in your **MDWDataLake*suffix*** storage account.
19. Download the file generated to inspect its contents.
20. **IMPORTANT**: Delete this test file before moving to next steps of this exercise.



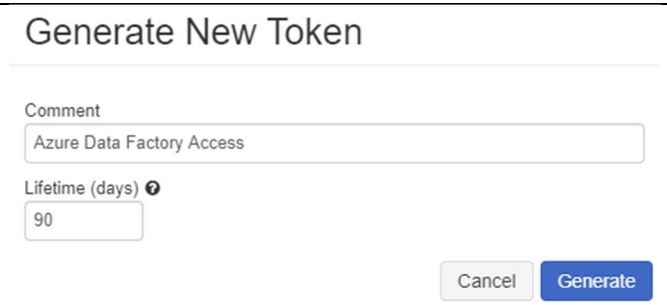
Create Databricks Linked Service in Azure Data Factory

In this section you will create a Databricks linked service in Azure Data Factory. Through this linked service you will be able to create a data pipelines to copy NYC images to your data lake and integrate Databricks notebooks to its execution.

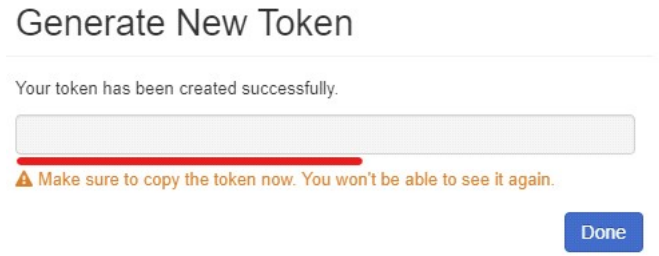


IMPORTANT: Execute these steps on your host computer	
1. On the **Azure Databricks** portal, click the **User** icon on the top right-hand corner of the screen. 2. Click on the **User Settings** menu item.	
3. On the **User Settings** blade, under the **Access Tokens** tab, click **Generate New Token**.	

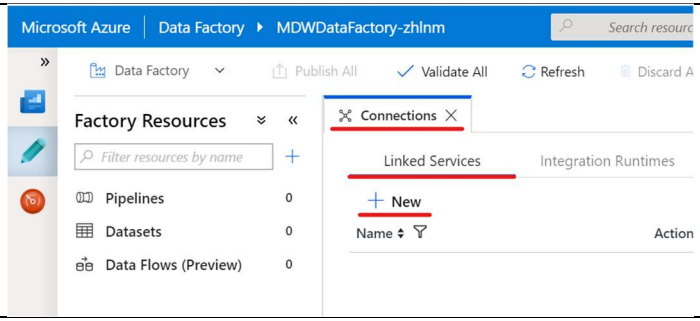
4. On the **Generate New Token** pop-up window, enter “Azure Data Factory Access” in the **Comment** field. Leave **Lifetime (days)** with the default value of 90 days.
5. Click **Generate.**



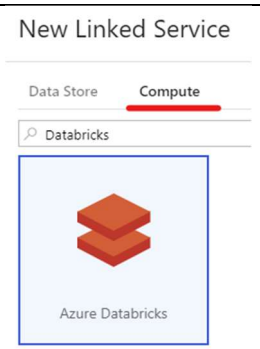
6. **IMPORTANT**: Copy the generated access token to Notepad and save it. You won’t be able to retrieve it once you close this window.



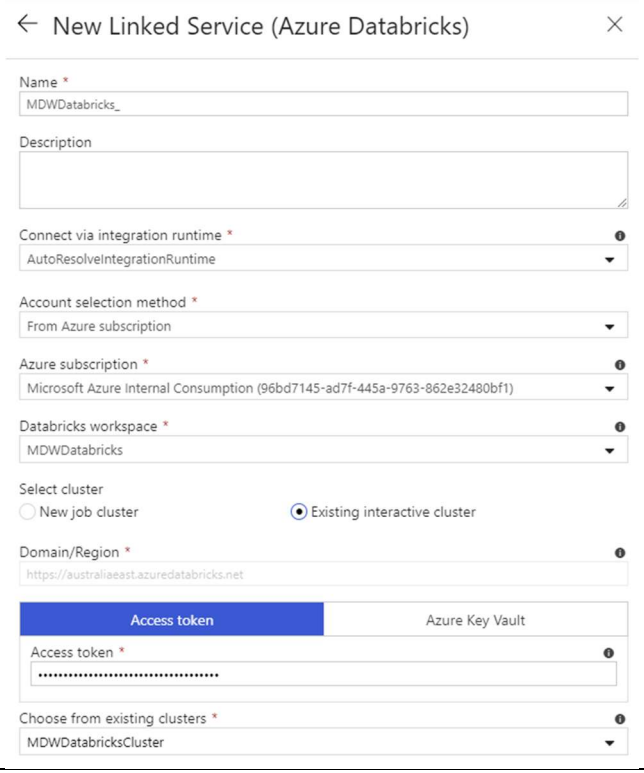
7. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel. Under **Connections** tab, click **Linked Services** and then click **+ New** to create a new linked service connection.



8. On the **New Linked Service** blade, click the **Compute** tab.
9. Type “Azure Databricks” in the search box to find the **Azure Databricks** linked service.
10. Click **Continue**.

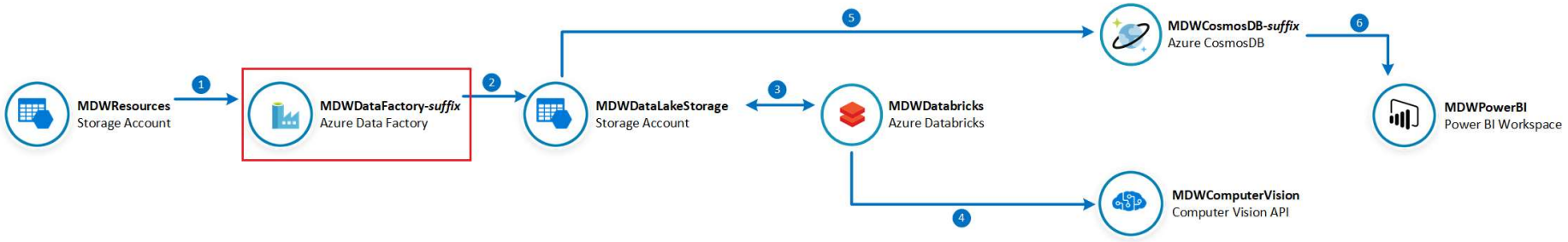


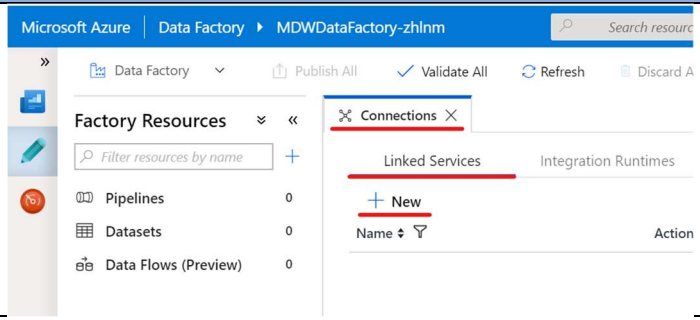
11. On the **New Linked Service (Azure Databricks)** blade, enter the following details:
    a. **Name**: MDWDatabricks
    b. **Connect via integration runtime**: AutoResolveIntegrationRuntime
    c. **Account selection method**: From Azure subscription
    d. **Azure subscription**: *<select your subscription>*
    e. **Databricks workspace**: MDWDatabricks
    f. **Select cluster**: Existing interactive cluster
    g. **Access token**: *<copy and paste access token here>*
    h. **Choose from existing clusters**: MDWDatabricksCluster
12. Click **Test connection** to make sure you entered the correct connection details. You should see a “Connection successful” message above the button.
13. If the connection was successful, then click Finish. If you got an error message, please review the connection details above and try again.



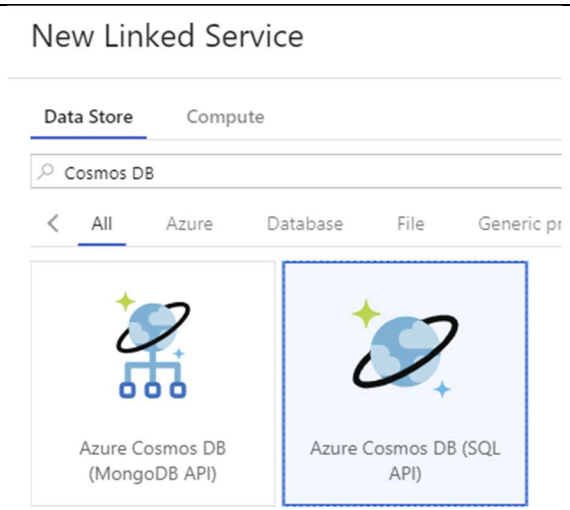
Create CosmosDB Linked Service in Azure Data Factory

In this section you will create a CosmosDB linked service in Azure Data Factory. CosmosDB will be used as the final repository of image metadata information generated by the Computer Vision API. Power BI will then be used to visualise the CosmosDB data.

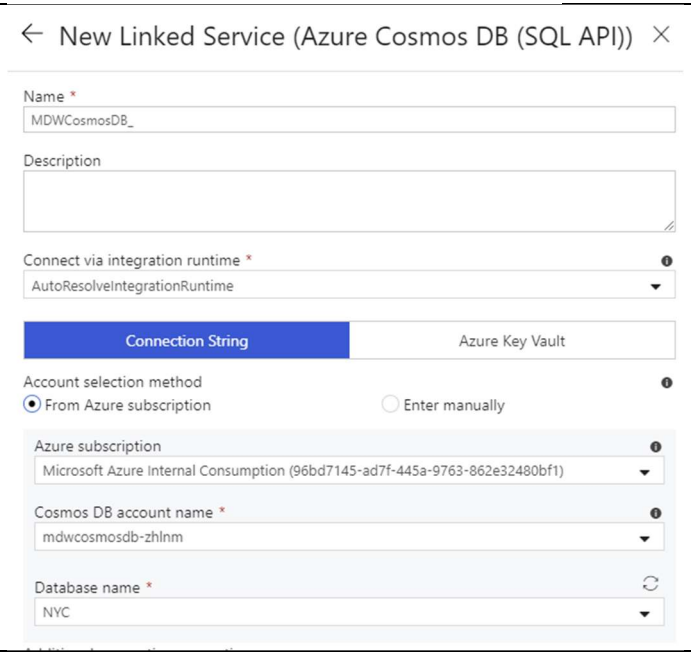


IMPORTANT: Execute these steps on your host computer	
1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel. Under **Connections** tab, click **Linked Services** and then click **+ New** to create a new linked service connection.	

2. On the **New Linked Service** blade, click the **Data Store** tab.
3. Type “Cosmos DB” in the search box to find the **Azure Cosmos DB (SQL API)** linked service.
4. Click **Continue**.

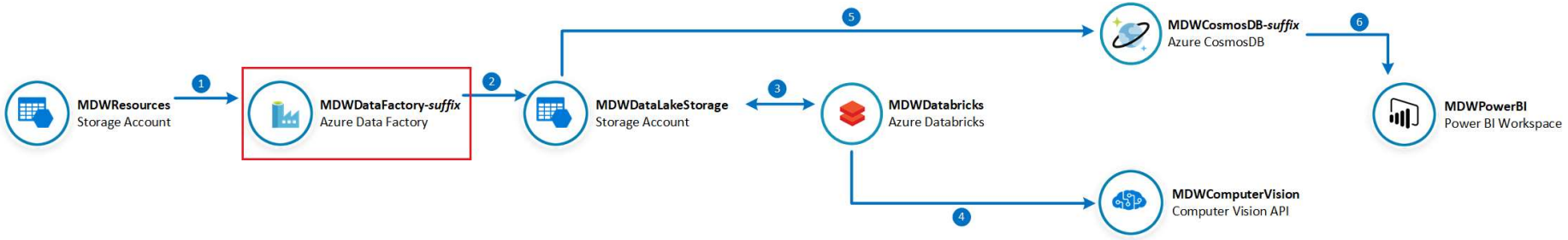


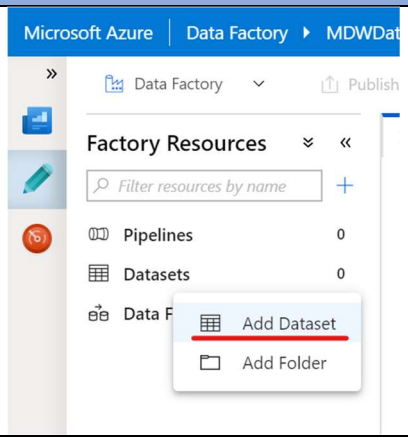
14. On the **New Linked Service (Azure Databricks)** blade, enter the following details:
    i. **Name**: MDWCosmosDB
    j. **Connect via integration runtime**: AutoResolveIntegrationRuntime
    k. **Account selection method**: From Azure subscription
    l. **Azure subscription**: *<select your subscription>*
    m. **Cosmos DB account name**: mdwcosmosdb-*suffix*
    n. **Database name**: NYC
15. Click **Test connection** to make sure you entered the correct connection details. You should see a “Connection successful” message above the button.
16. If the connection was successful, then click Finish. If you got an error message, please review the connection details above and try again.

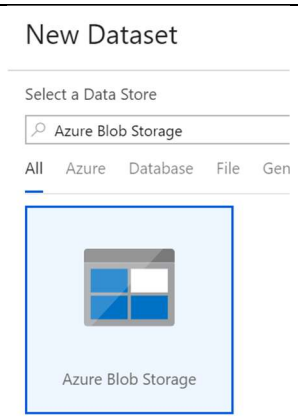
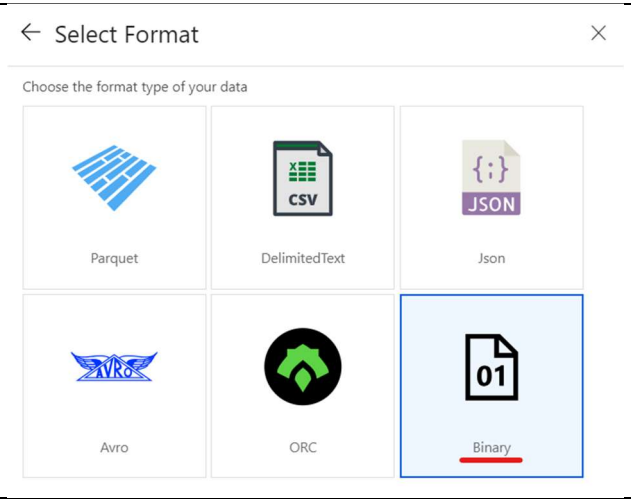


Create Azure Data Factory data sets.

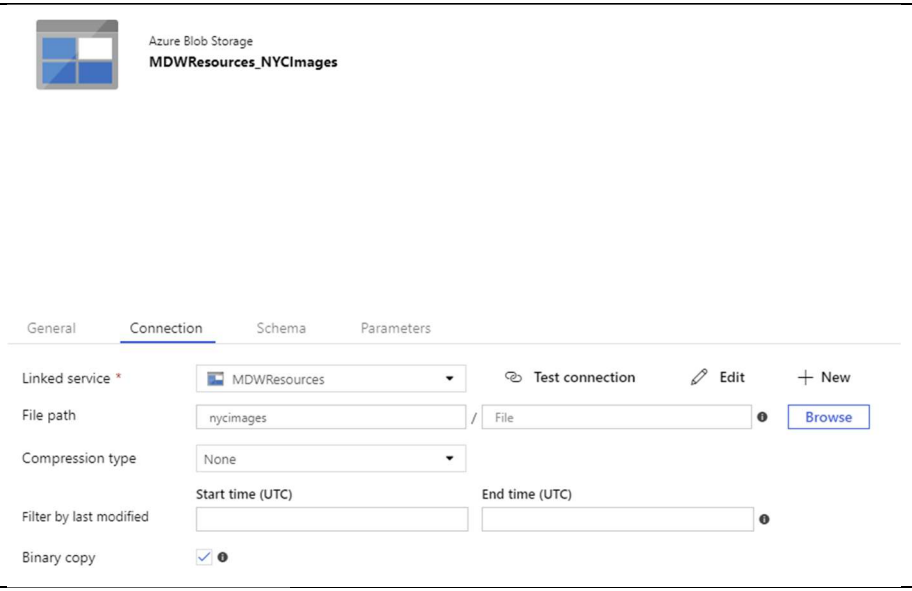
In this section you will create an Azure Data Factory data sets that will be used in the data pipeline.



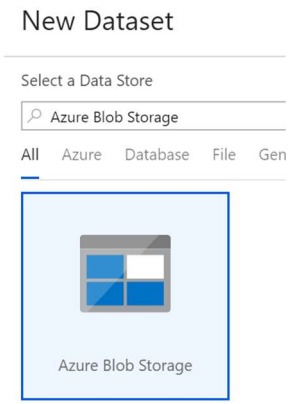
IMPORTANT: Execute these steps on your host computer	
1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel. Under **Factory Resources** tab, click the ellipsis (**…**) next to **Datasets** and then click **Add Dataset** to create a new dataset.	

2. Type “Azure Blob Storage” in the search box and select **Azure Blob Storage.** Click **Finish.**	
3. On the **Select Format** blade, select **Binary** and click **Continue**.	

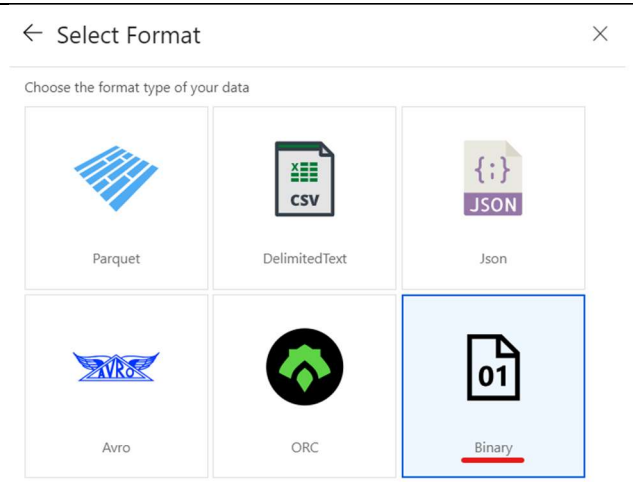
4. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWResources_NYCImages
    b. **Connection > Linked Service:** MDWResources
    c. **Connection > File Path:** nycimages
    d. **Connection > Binary Copy:** Checked
5. Leave remaining fields with default values.



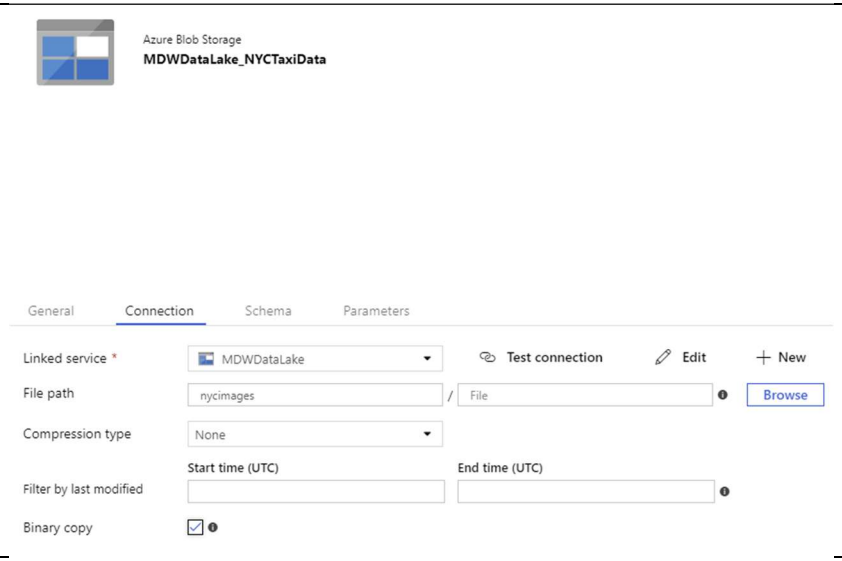
6. Repeat the process to create another dataset, this time referencing the NYCImages container in your MDWDataLake storage account.
7. Type “Azure Blob Storage” in the search box and select **Azure Blob Storage.** Click **Finish.**

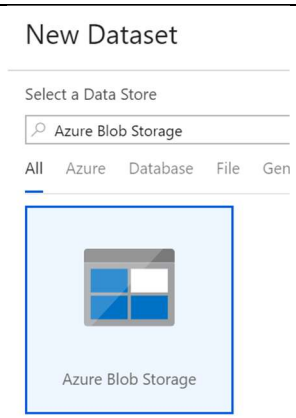
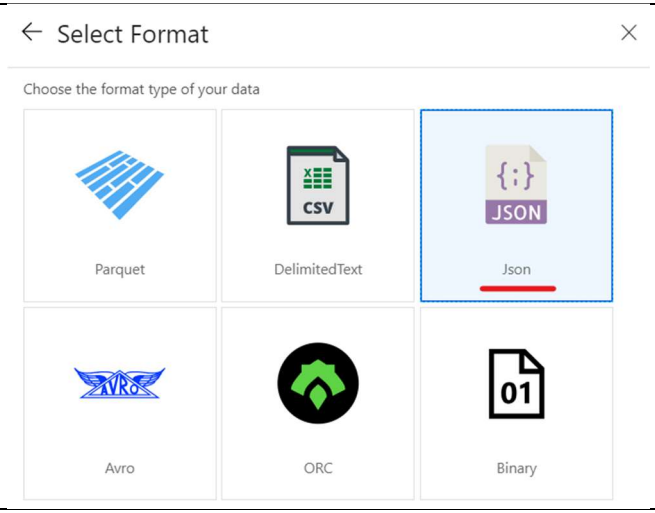


8. On the **Select Format** blade, select **Binary** and click **Continue**.

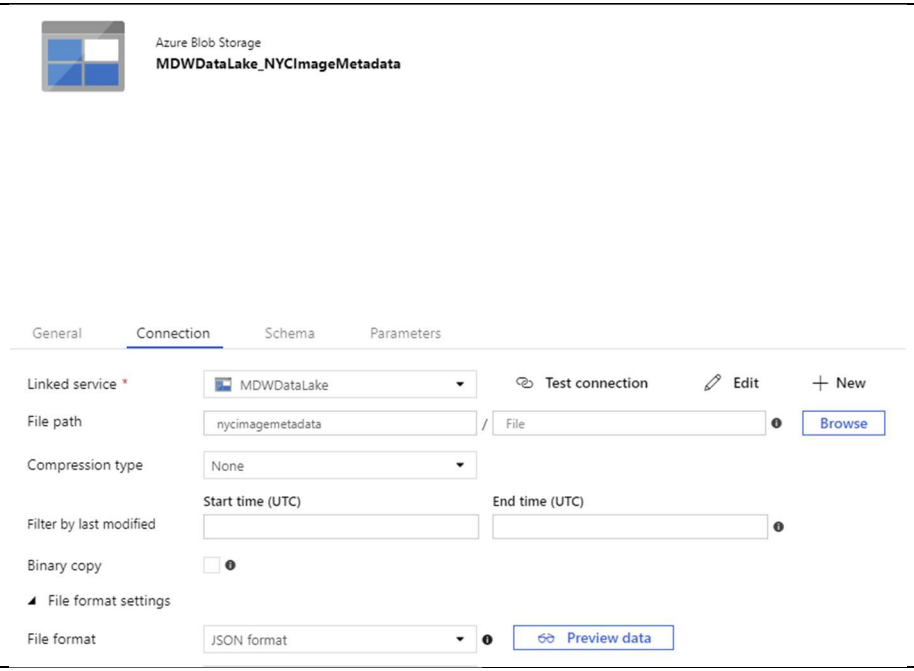


9. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWDataLake_NYCImages
    b. **Connection > Linked Service:** MDWDataLake
    c. **Connection > File Path:** nycimages
10. Leave remaining fields with default values.

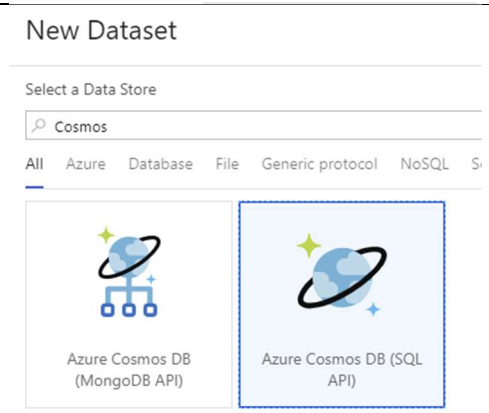


11. Repeat the process to create another dataset, this time referencing the NYCImageMetadata container in your MDWDataLake storage account. 12. Type “Azure Blob Storage” in the search box and select **Azure Blob Storage.** Click **Finish.**	
13. On the **Select Format** blade, select **JSON** and click **Continue**.	

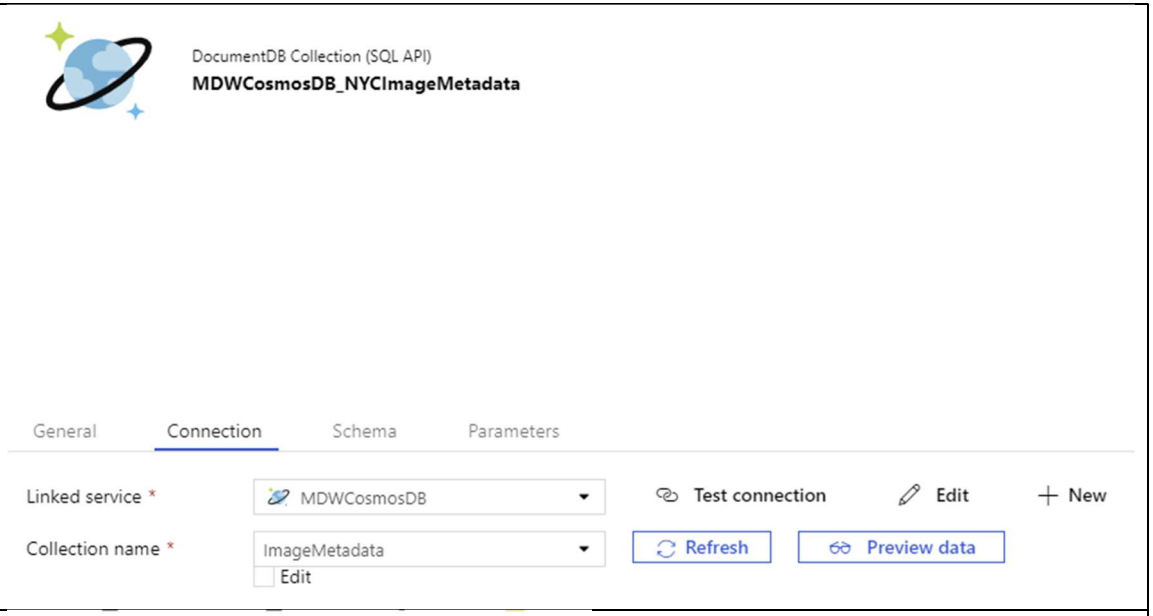
14. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWDataLake_NYCImageMetadata
    b. **Connection > Linked Service:** MDWDataLake
    c. **Connection > File Path:** nycimagemetadata
    d. **File format:** JSON format
15. Leave remaining fields with default values.



16. Repeat the process to create another dataset, this time referencing the ImageMetadata collection in your MDWCosmosDB database.
17. Type “Cosmos DB” in the search box and select **Azure Cosmos DB (SQL API).** Click **Continue.**



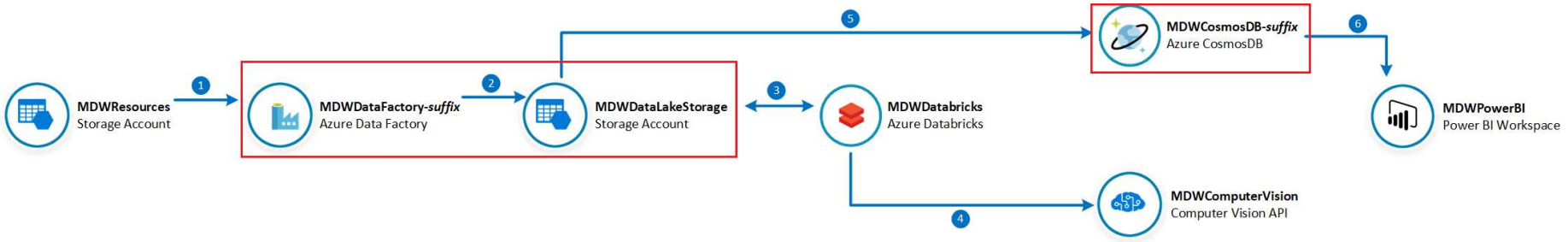
18. On the **New Data Set** tab, enter the following details:
    a. **General > Name**: MDWCosmosDB_NYCImageMetadata
    b. **Connection > Linked Service:** MDWCosmosDB
    c. **Collection name:** ImageMetadata
19. Leave remaining fields with default values.



20. Publish your dataset changes by clicking the **Publish all** button.

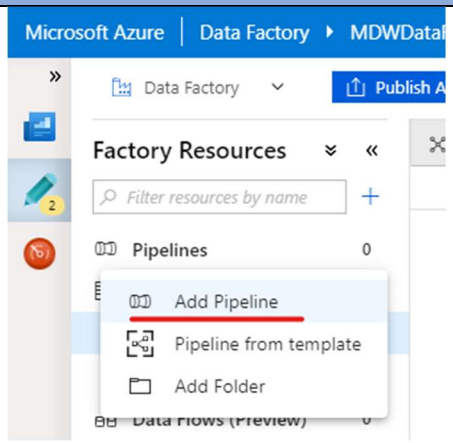
Create Azure Data Factory pipeline to generate and save image metadata to Cosmos DB.

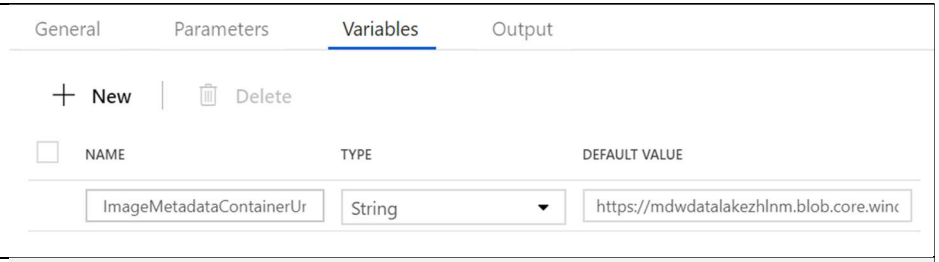
In this section you will create an Azure Data Factory pipeline to copy New York images from MDWResources into your MDWDataLakes*suffix* storage account. The pipeline will then execute a Databricks notebook for each image and generate a metadata file in the NYCImageMetadata container. The pipeline finishes by saving the image metadata content in a CosmosDB database.



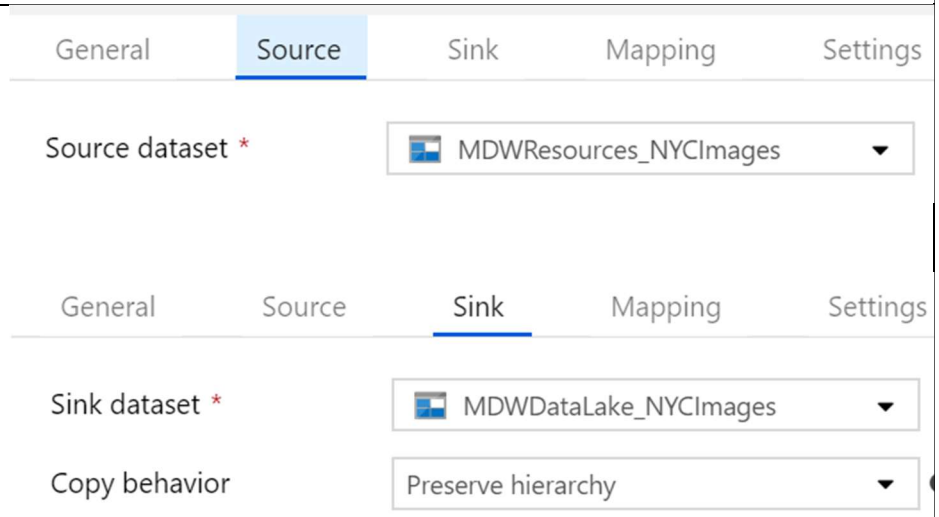
IMPORTANT: Execute these steps on your host computer

1. Open the **Azure Data Factory** portal and click the **Author** option on the left-hand side panel. Under **Factory Resources** tab, click the ellipsis (**…**) next to **Pipelines** and then click **Add Pipeline** to create a new dataset.
2. On the New Pipeline tab, enter the following details:
    a. **General > Name**: Copy NYC Images
    b. **Variables > + New >**
        i. **Name**: ImageMetadataContainerUrl
        ii. **Default Value**: https://mdwdatalakes*suffix*.blob.core.windows.net/nycimages/
3. Leave remaining fields with default values.

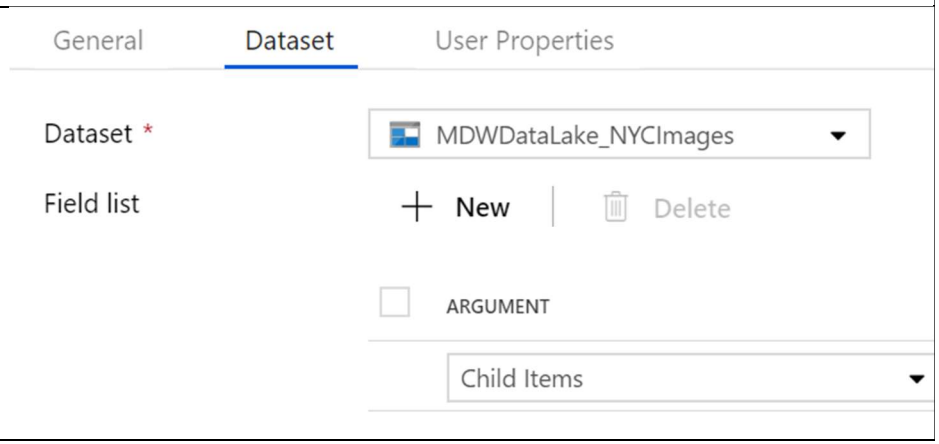




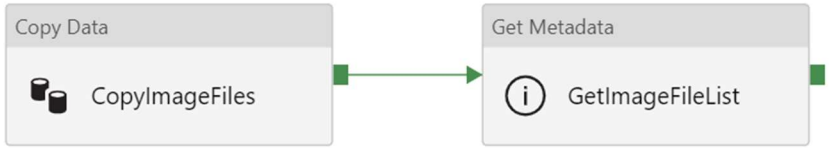
4. From the Activities panel, type “Copy Data” in the search box. Drag the Copy Data activity on to the design surface. This copy activity will copy image files from MDWResources to MDWDatalake.
5. Select the Copy Data activity and enter the following details:
    a. **General > Name:** CopyImageFiles
    b. **Source > Source dataset:** MDWResources_NYCImages
    c. **Sink > Sink dataset:** MDWDataLake_NYCImages
    d. **Sink > Copy Behavior:** Preserve Hierarchy
6. Leave remaining fields with default values.



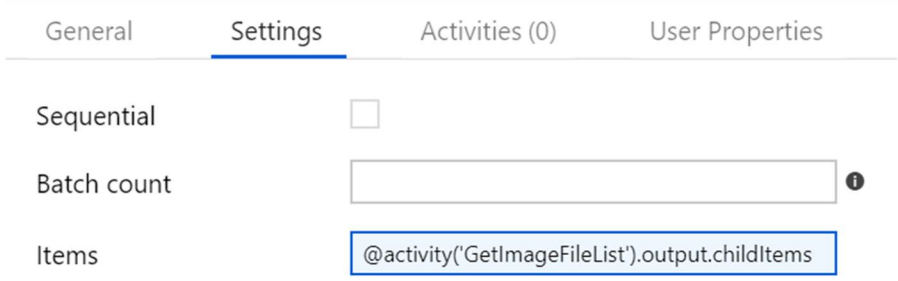
7. From the Activities panel, type “Get Metadata” in the search box. Drag the **Get Metadata** activity on to the design surface. This activity will retrieve a list of image files saved in the NYCImages container by the previous CopyImageFiles activity.
8. Select the Get Metadata activity and enter the following details:
    a. **General > Name:** GetImageFileList
    b. **Source > Source dataset:** MDWDataLake_NYCImages
    c. **Source > Field list:** Child Items
9. Leave remaining fields with default values.



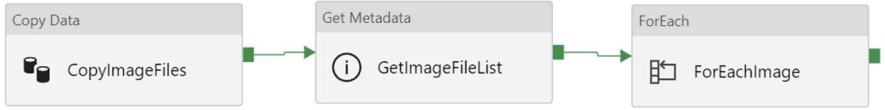
10. Create a **Success** (green) precedence constraint between CopyImageFiles and GetImageFileList activities. You can do it by dragging the green connector from CopyImageFiles and landing the arrow onto GetImageFileList.



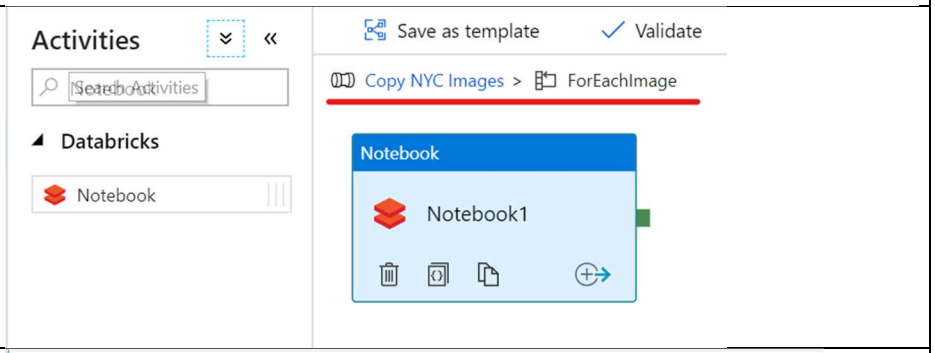
11. From the Activities panel, type “ForEach” in the search box. Drag the **ForEach** activity on to the design surface. This ForEach activity will act as a container for other activities that will be executed for each image files returned by the GetImageFileList activity.
12. Select the ForEach activity and enter the following details:
    a. **General > Name:** ForEachImage
    b. **Settings > Items:** @activity('GetImageFileList').output.childItems
13. Leave remaining fields with default values.



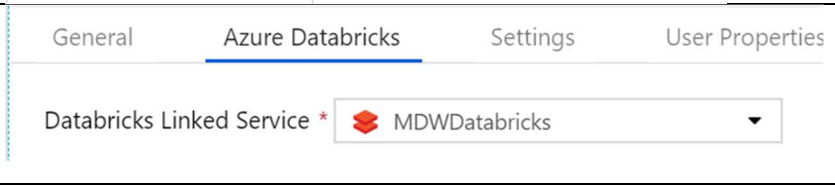
14. Create a **Success** (green) precedence constraint between GetImageFileList and ForEachImage activities. You can do it by dragging the green connector from GetImageFileList and landing the arrow onto ForEachImage.



15. Double-click the **ForEachImage** activity to edit its contents. Note the design context is displayed on the top left-hand side of the design canvas.

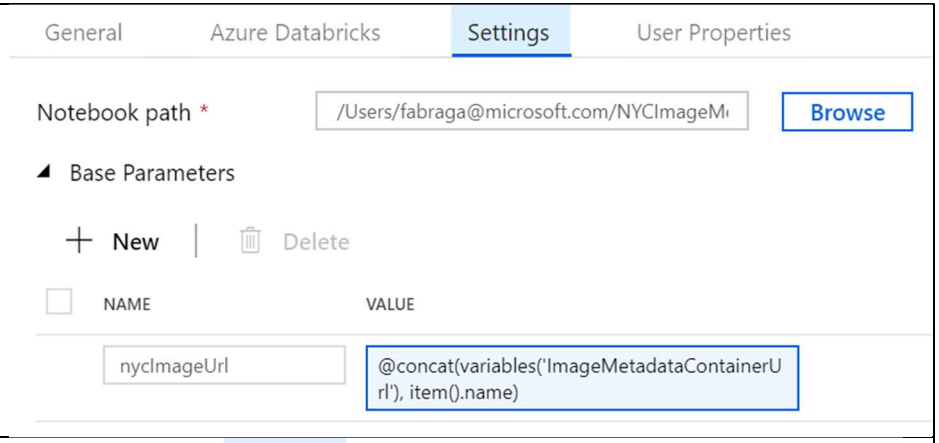


16. From the Activities panel, type “Notebook” in the search box. Drag the **Notebook** activity on to the design surface. This Notebook activity will pass the image URL as a parameter to the Databricks notebook we created previously.
17. Select the Notebook activity and enter the following details:

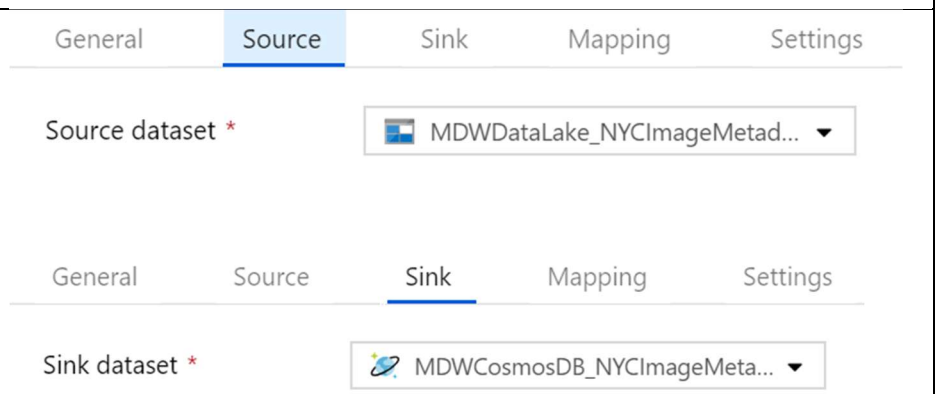


a. **General > Name:** GetImageMetadata
b. **Azure Databricks > Databricks Linked Service:** MDWDatabricks
c. **Settings > Notebook path:** Click **Browse** and navigate to /Users/*your-user-name*/NYCImageMetadata-Lab
d. **Base Parameters: + New**
    i. nycImageUrl: @concat(variables('ImageMetadataContainerUrl'), item().name)

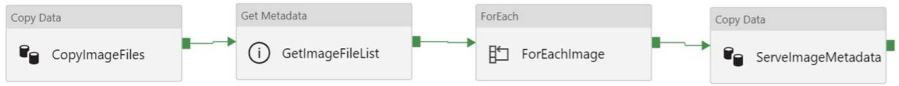
18. Leave remaining fields with default values.



19. Navigate back to the "Copy NYC Images" pipeline canvas.
20. From the Activities panel, type "Copy Data" in the search box. Drag the Copy Data activity on to the design surface. This copy activity will copy image metadata from the JSON files sitting on the NYCImageMetadata container in MDWDataLake to the ImageMetadata collection on CosmosDB.
21. Select the Copy Data activity and enter the following details:
    a. **General > Name:** ServeImageMetadata
    b. **Source > Source dataset:** MDWDataLake_NYCImageMetadata
    c. **Sink > Sink dataset:** MDWCosmosDB_NYCImageMetadata
22. Leave remaining fields with default values.

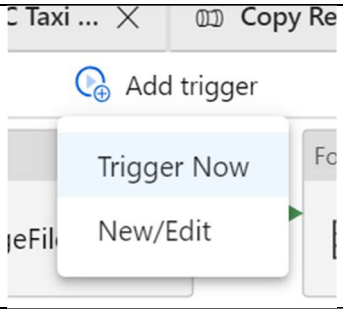


23. Create a **Success** (green) precedence constraint between GetImageFileList and ForEachImage activities. You can do it by dragging the green connector from GetImageFileList and landing the arrow onto ForEachImage.

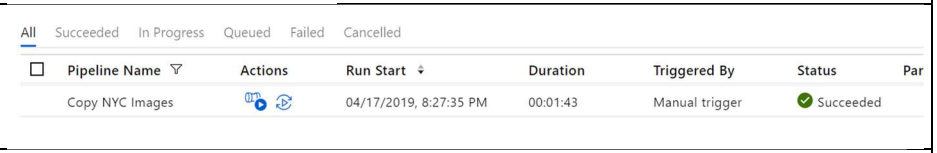


24. Publish your pipeline changes by clicking the **Publish all** button.

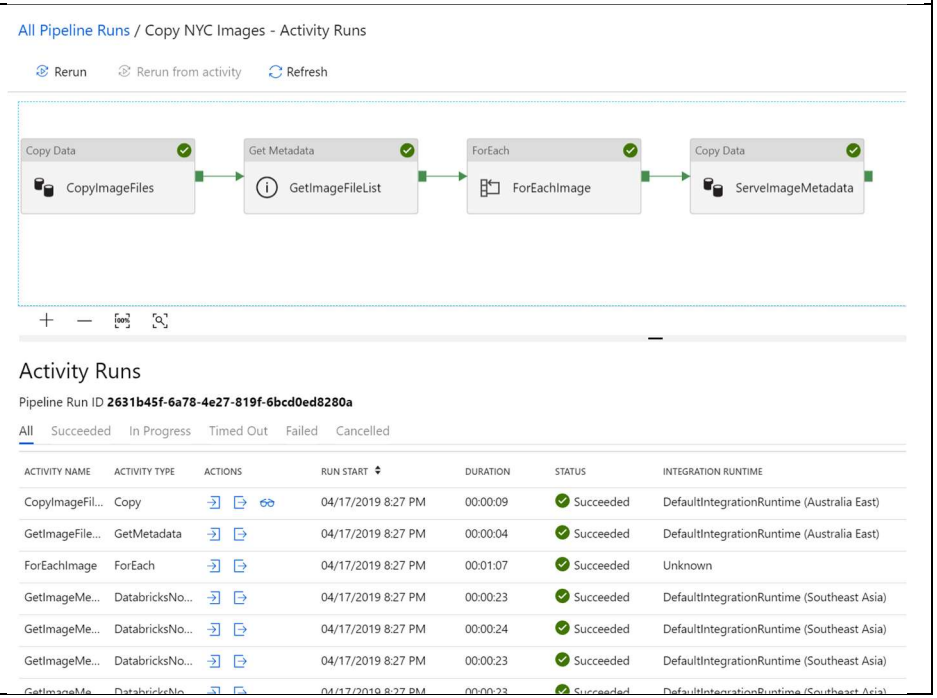
25. To execute the pipeline, click on **Add trigger** menu and then **Trigger Now**.
26. On the **Pipeline Run** blade, click **Finish**.



27. To monitor the execution of your pipeline, click on the **Monitor** menu on the left-hand side panel.
28. You should be able to see the **Status** of your pipeline execution on the right-hand side panel.



29. Click the **View Activity Runs** button for detailed information about each activity execution in the pipeline. The whole execution should last between 7-8 minutes.

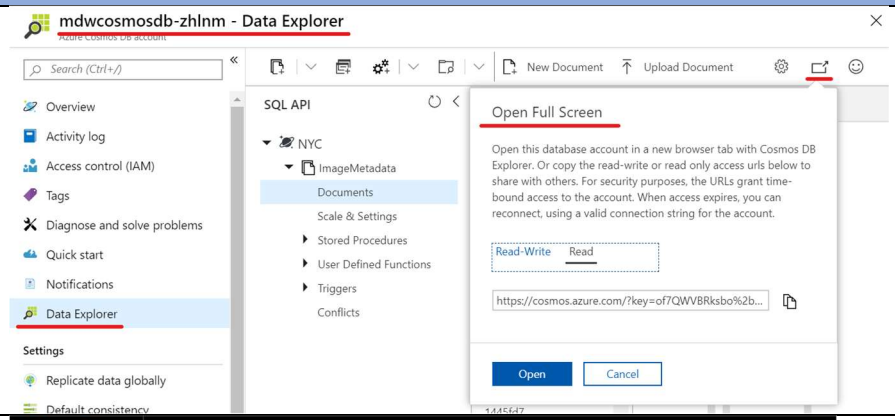


Explore Image Metadata Documents in CosmosDB

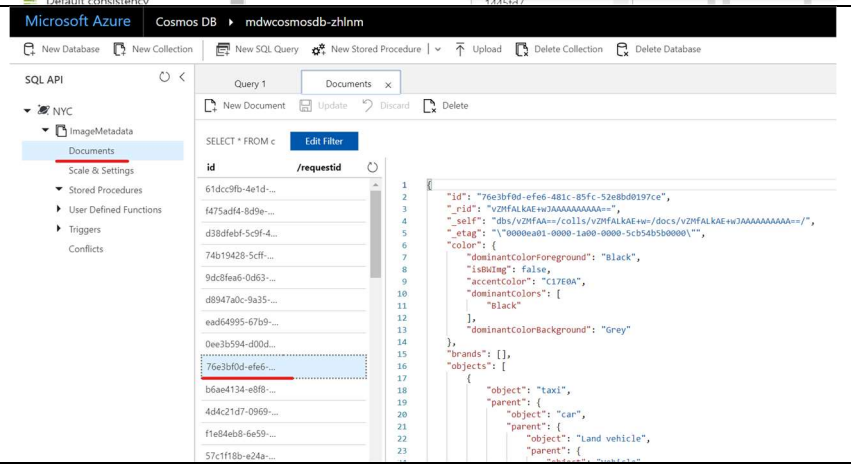
In this section you will explore the image metadata records generated by the Azure Data Factory pipeline in CosmosDB. You will use the Cosmos DB's SQL API to write SQL-like queries and retrieve data based on their criteria.

IMPORTANT: Execute these steps on your host computer

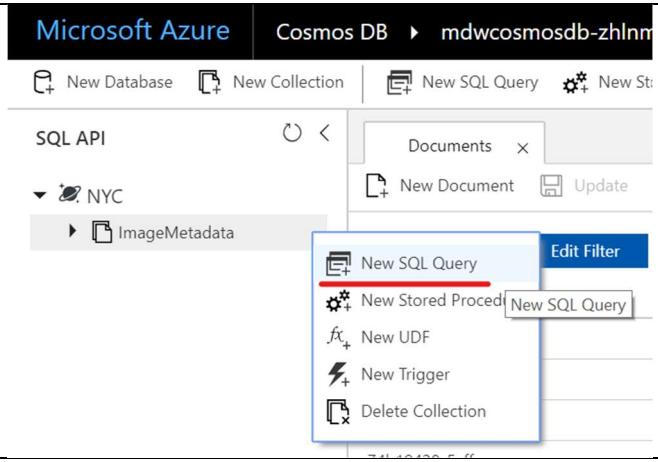
1. In the Azure Portal, go to the lab resource group and locate the CosmosDB account **MDWCosmosDB-*suffix***.
2. On the **Data Explorer** panel, click **Open Full Screen** button on the top right-hand side of the screen.
3. On the Open Full Screen pop-up window, click **Open**.



4. On the **Azure Cosmos DB Data Explorer** window, under **NYC > ImageMetadata** click **Documents** to see the full list of documents in the collection.
5. Click the document id to see the content of each document.



6. Click the ellipsis (…) next to ImageMetadata collection.
7. On the pop-up menu, click New SQL Query to open a new query tab.

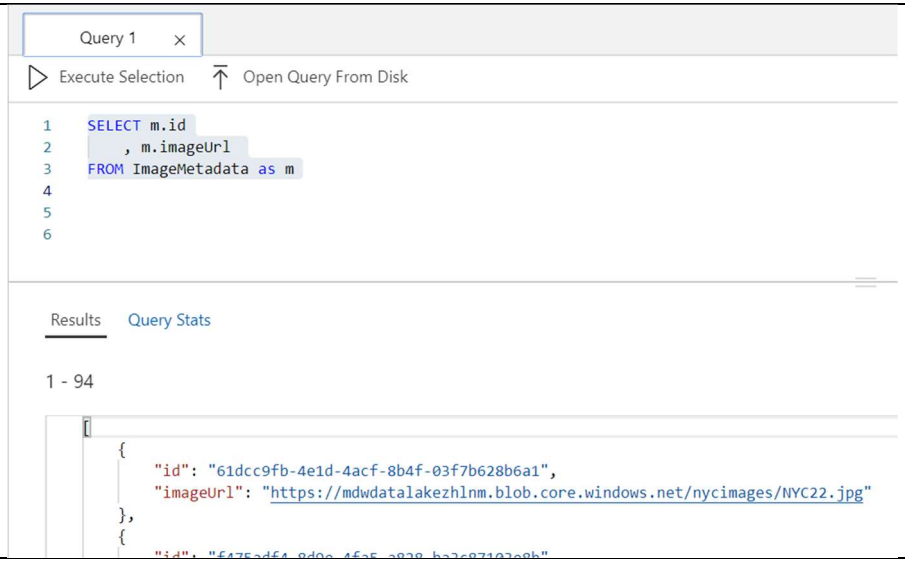


8. On the New Query 1 window, try the different SQL Commands from the list.
9. Click the **Execute Selection** button to execute your query.

SQL Commands

```
SELECT m.id
    , m.imageUrl
FROM ImageMetadata as m

SELECT m.id
    , m.imageUrl
    , tags.name
FROM ImageMetadata as m
    JOIN tags IN m.tags
WHERE tags.name = 'wedding'
```

10. Check the results in the **Results** panel.

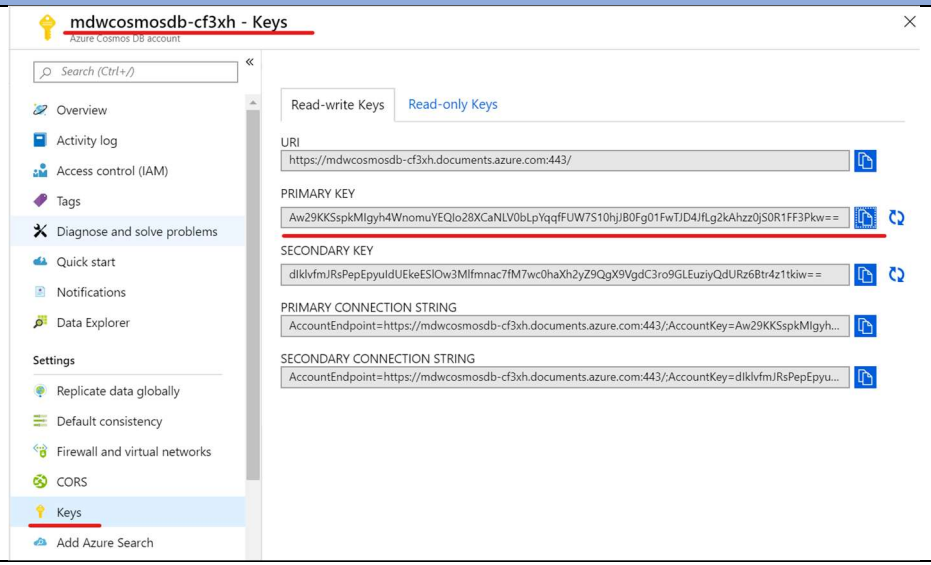


Visualize Data with Power BI

In this section you are going to use Power BI to visualize data from Cosmos DB. The Power BI report will use an Import connection to retrieve image metadata from Cosmos DB and visualise images sitting in your data lake.

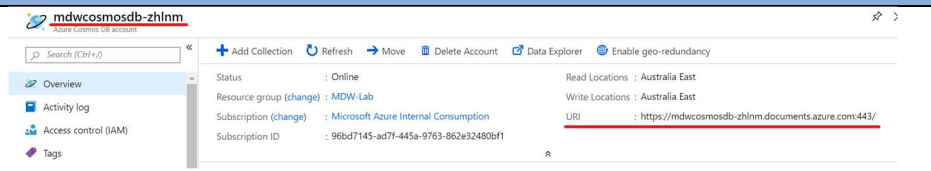
IMPORTANT: Execute these steps on your host computer

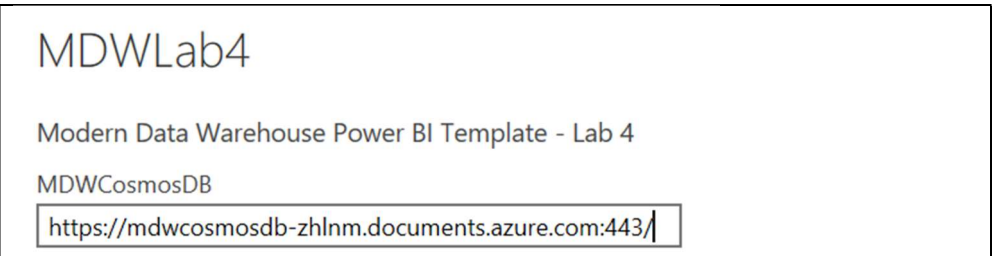
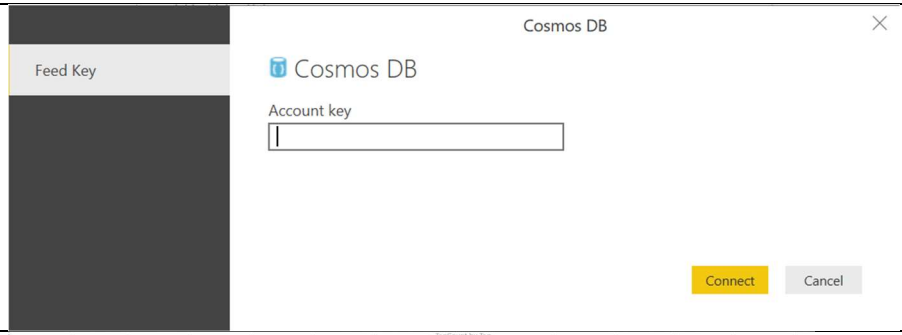
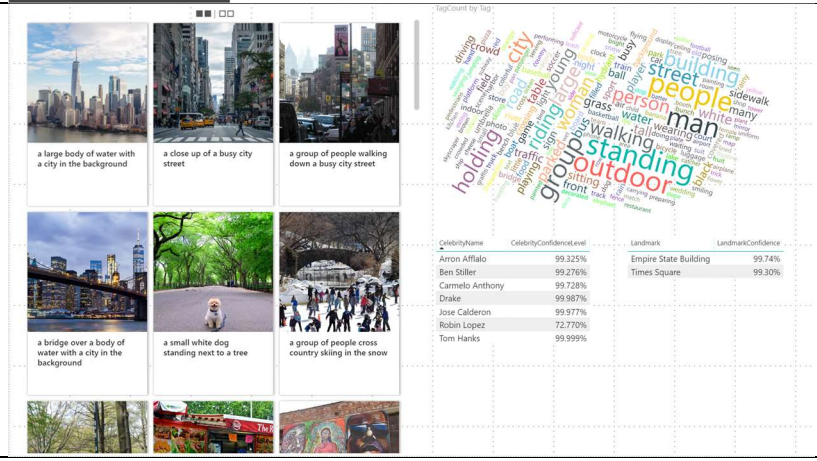
1. Navigate to the Azure Portal and retrieve the **mdwcosmosdb-*suffix*** access key.
2. Save it to notepad. You will need it in the next step.



IMPORTANT: Steps to be executed on **MDWDesktop**

3. On MDWDesktop, download the Power BI report from the link https://aka.ms/MDWLab4 and save it in the Desktop.
4. Open the file MDWLab4.pbit with Power BI Desktop.
5. When prompted to enter the value of the **MDWCosmosDB** parameter, type the full server URI: **https://mdwcosmosdb-*suffix*.documents.azure.com:443/**
6. Click **Load.**



	
7. When prompted for an Account Key, paste the MDWCosmosDB account key you retrieved in the previous exercise. 8. Click **Connect**.	
9. Once data finish loading, interact with the report by clicking on the different images displayed and check the accuracy of their associated metadata. 10. Save your work and close Power BI Desktop.	

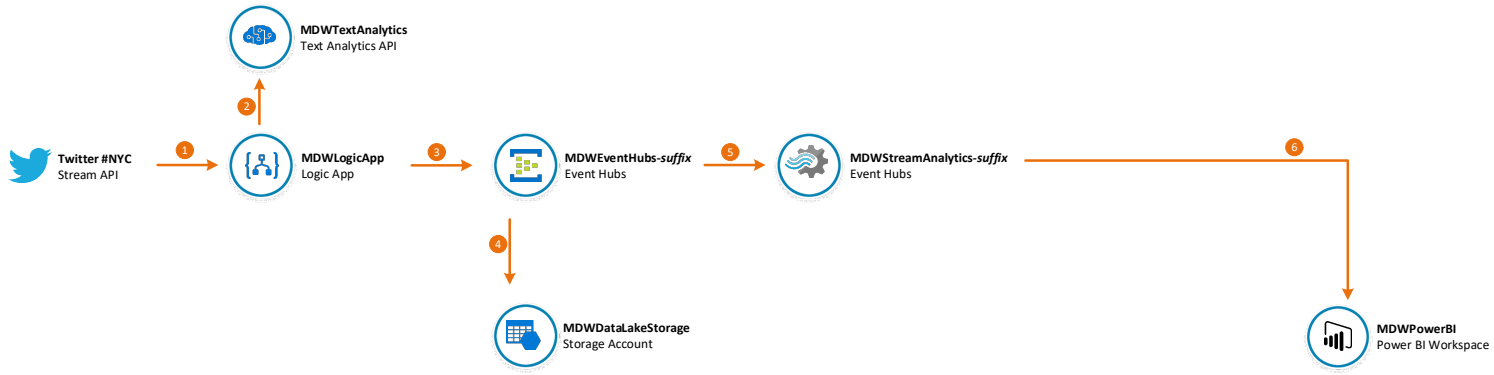
Lab 5: Ingest and Analyse real-time data with Event Hubs and Stream Analytics

In this lab you will use an Azure Logic App to connect to Twitter and generate a stream of messages using the hashtag #NYC. The logic app will invoke Azure Text Analytics Cognitive service to score Tweet sentiment and send messages to Event Hubs. You will then use Stream Analytics to generate the average tweet sentiment on the last 60 seconds and send the results to a real-time dataset in Power BI.

IMPORTANT: This lab requires you have a valid Twitter account. If you don't have one, you can sign up for free following the instructions here: https://twitter.com/signup.

IMPORTANT: This lab requires you have a valid Power BI account. If you don't have one you can register for a 60-day trial here: https://powerbi.microsoft.com/en-us/power-bi-pro/

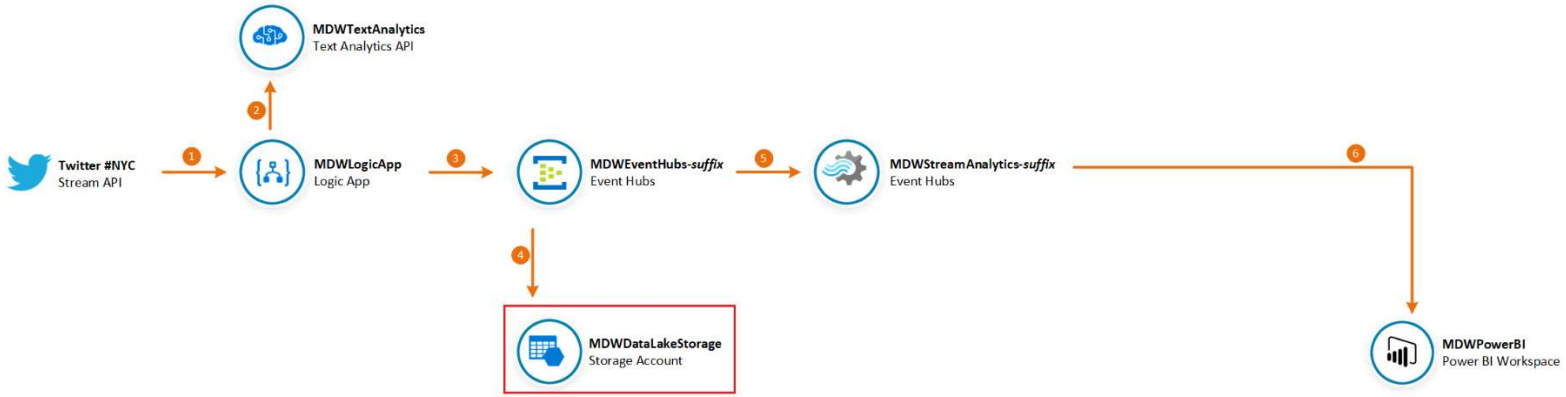
Lab Architecture



1. Build an Azure Logic App to invoke the Twitter API and retrieve Tweets with the hashtag #NYC
2. For each Tweet, invoke the Azure Text Analytics Cognitive service to detect its sentiment score
3. Format and send the Tweet's JSON message to Event Hubs
4. Save Tweet messages into your data lake for future analysis (cold path)
5. Send stream of Tweet messages to Stream Analytics for real-time analytics (hot path)
6. Visualize real-time data generated by Stream Analytics with Power BI

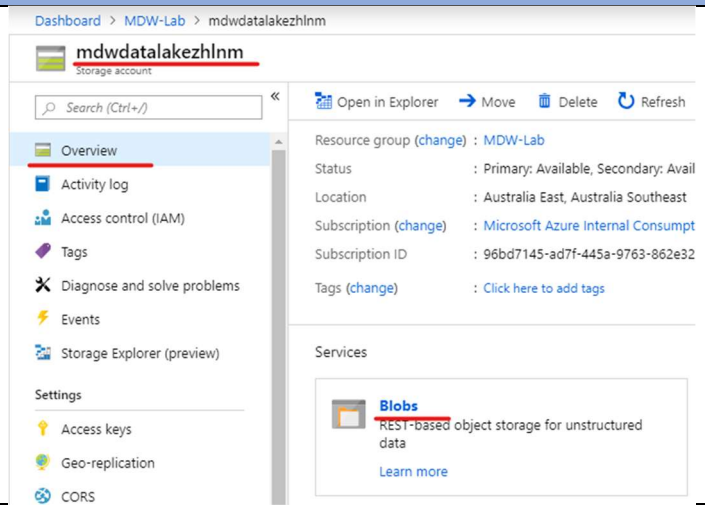
Create NYCTweets Container in Azure Blob Storage

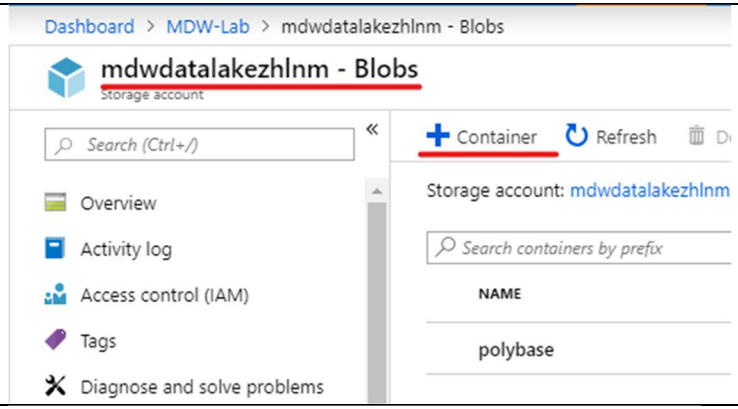
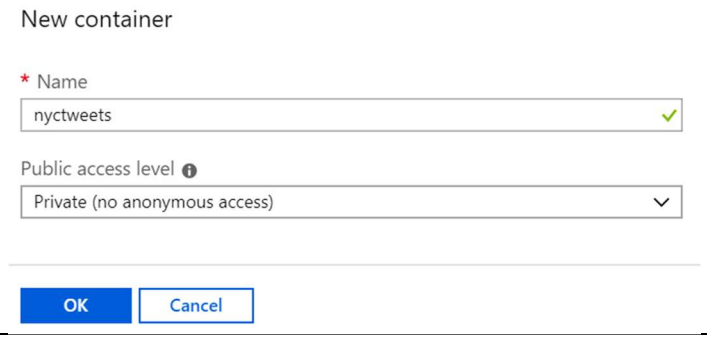
In this section you will create a container in your MDWDataLake that will be used as a repository for the NYC image files. You will copy 30 files from the MDWResources Storage Account into your NYCTaxiData container.



IMPORTANT: Execute these steps on your host computer

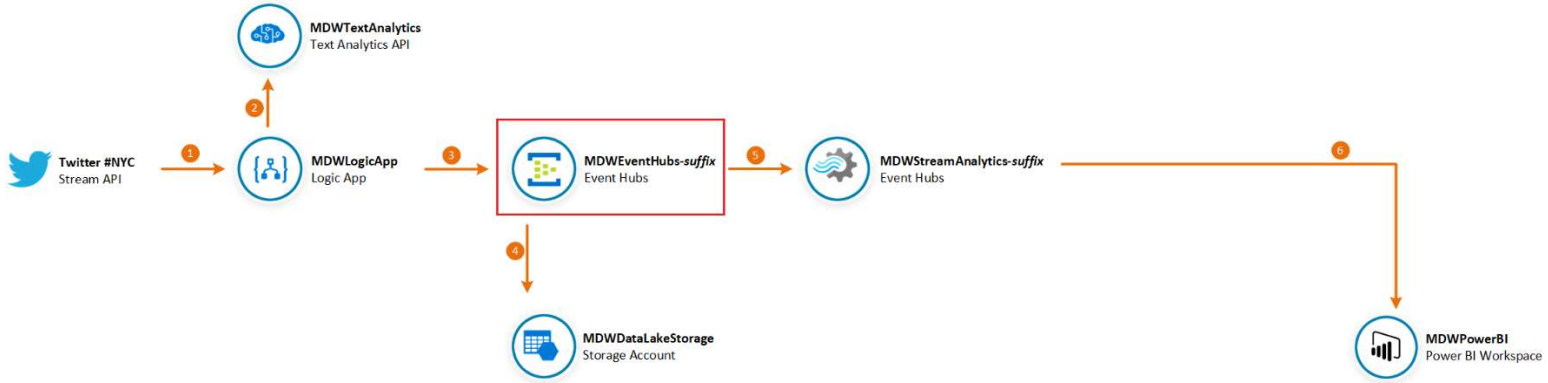
1. In the Azure Portal, go to the lab resource group and locate the Azure Storage account **mdwdatalake*suffix***.
2. On the **Overview** panel, click **Blobs**.



3. On the **mdwdalalake*suffix* – Blobs** blade, click **+ Container**.	
4. On the **New container** blade, enter the following details: a. **Name**: nyctweets b. **Public access level**: Private (no anonymous access) 5. Click **OK** to create the new container.	

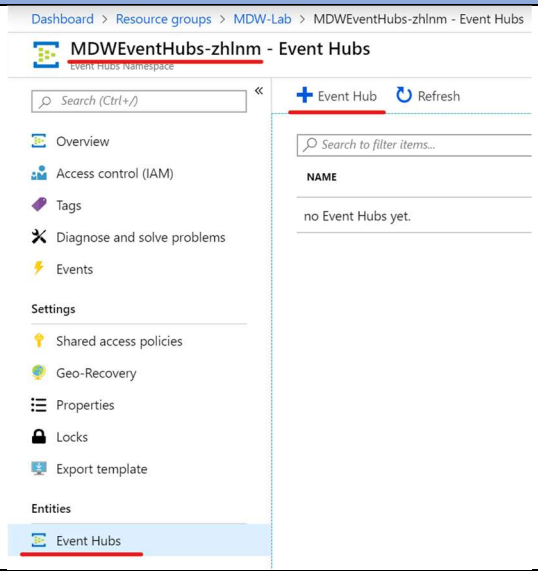
Create and Configure Event Hubs

In this section you will prepare Event Hubs to ingest Twitter data collected by the Logic App and save incoming messages to your Data Lake storage account.

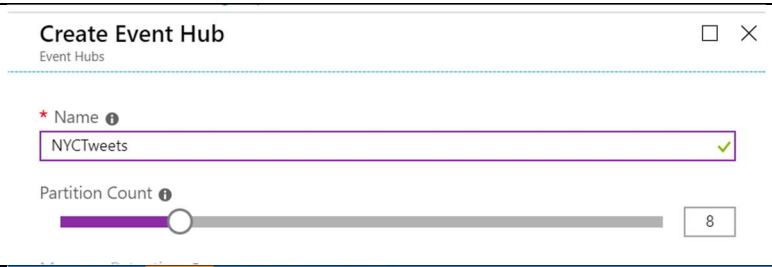


IMPORTANT: Execute these steps on your host computer

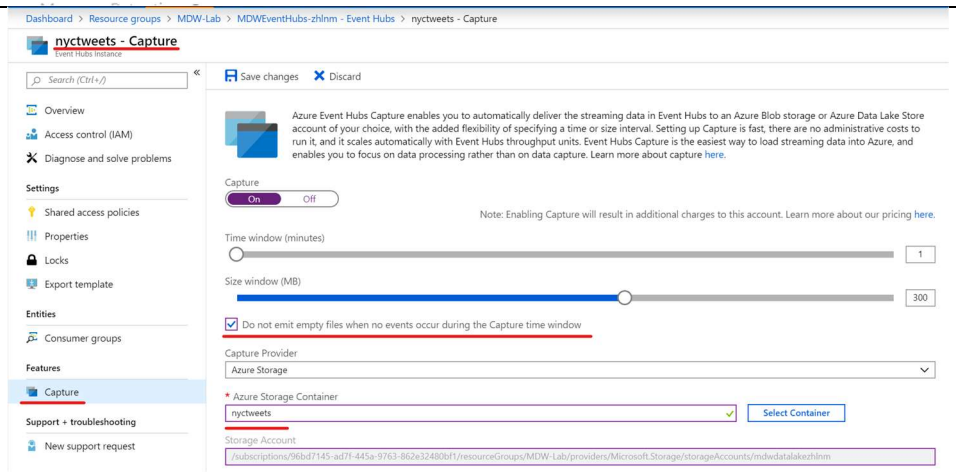
1. In the Azure Portal, go to the lab resource group and locate the Event Hubs resource **MDWEventHubs-*suffix***.
2. On the **Event Hubs** panel, click **+ Event Hub** button to create a new event hub.



3. On the **Create Event Hub** blade, type “NYCTweets” in the **Name** field and set the **Partition Count** to 8.
4. Click **Create**.

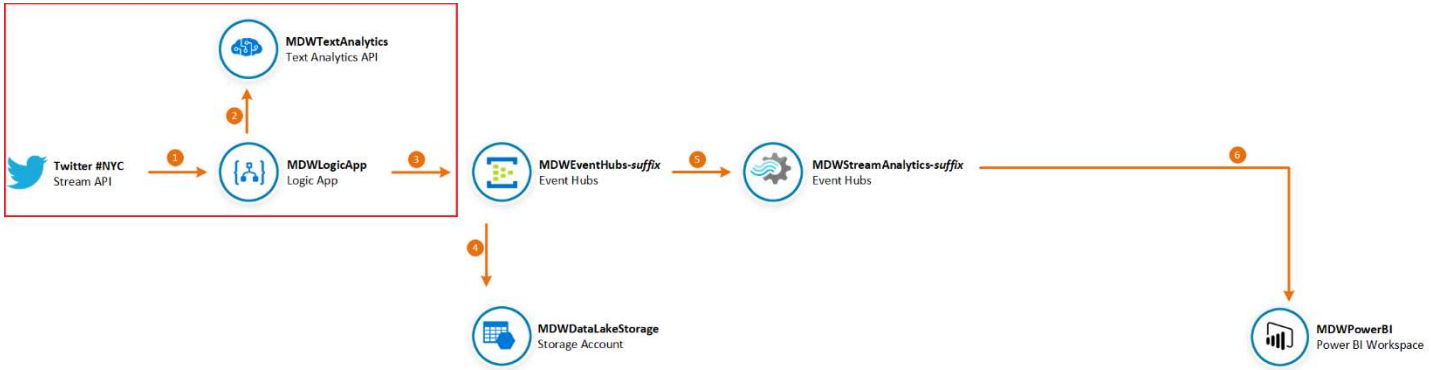


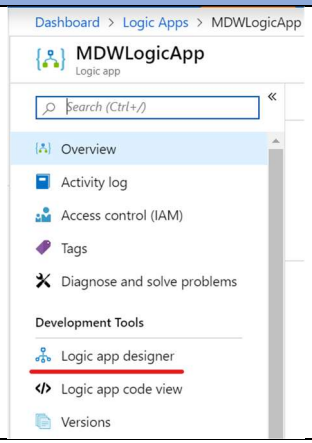
5. Once the NYCTweets event hub is created, click **Capture** on the left-hand side menu.
6. Enter the following details:
    a. **Capture**: On
    b. **Time window (minutes)**: 1
    c. **Do not emit empty files when no events occur during the Capture time window**: Checked.
    d. **Capture Provider**: Azure Storage
    e. **Azure Storage Container**: <select the nyctweets container in your **MDWDataLake*suffix*** storage account.
7. Leave remaining fields with their default values.
8. Click **Save Changes**.

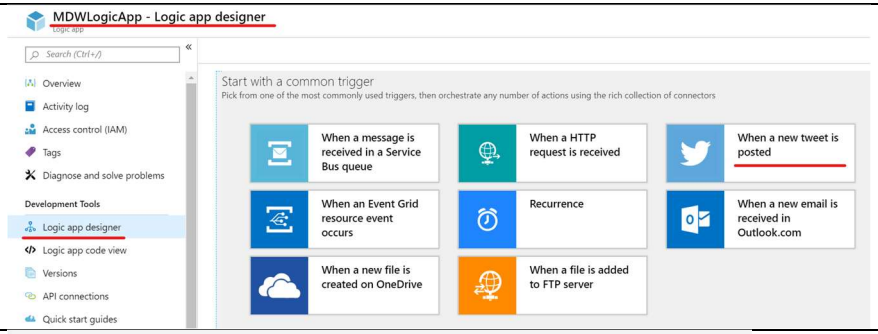
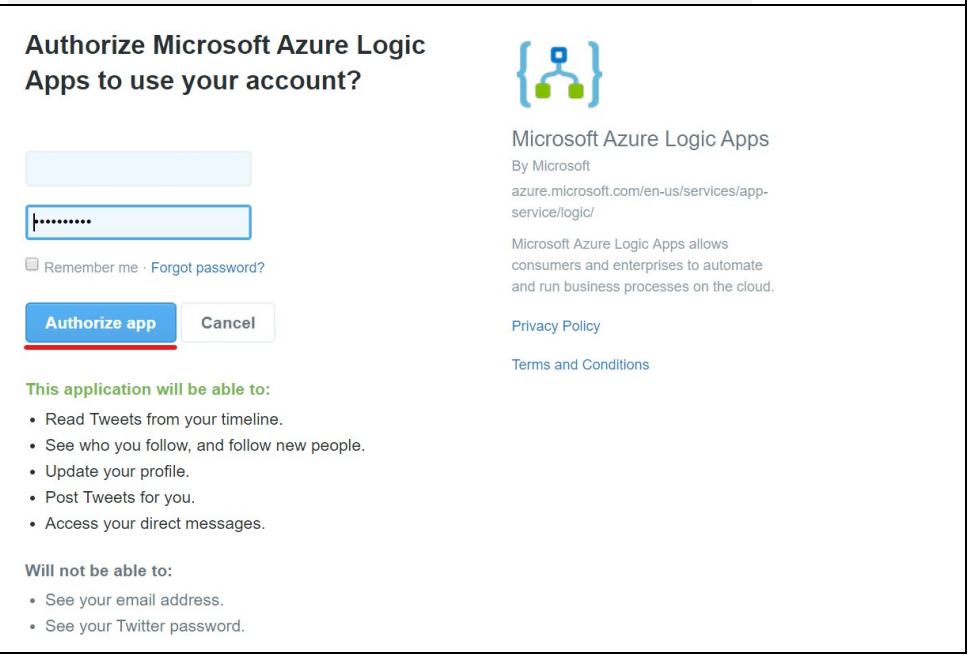


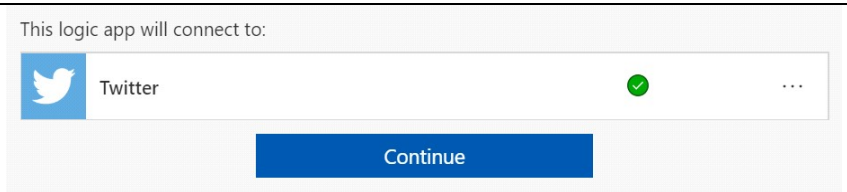
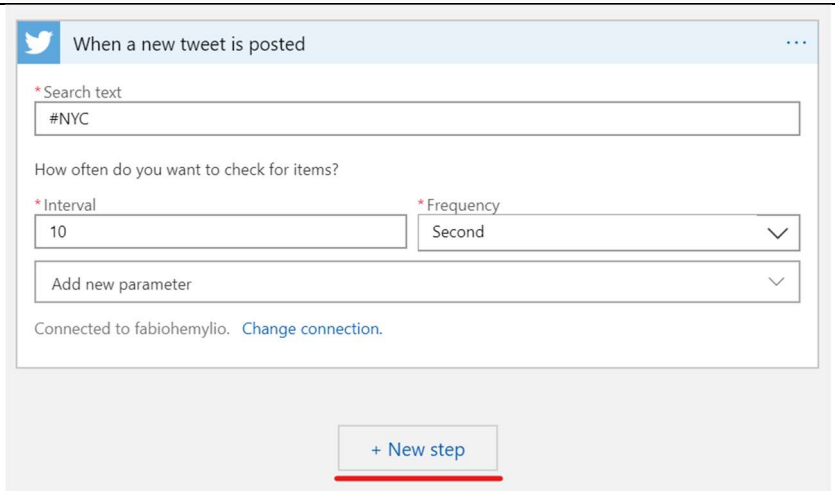
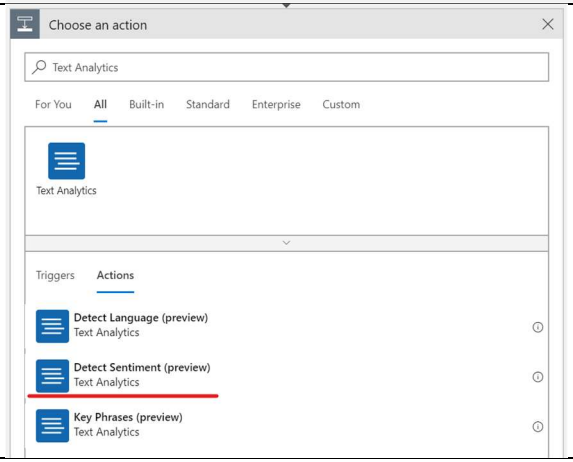
Create Azure Logic App to Read #NYC Tweets and post them to Event Hubs

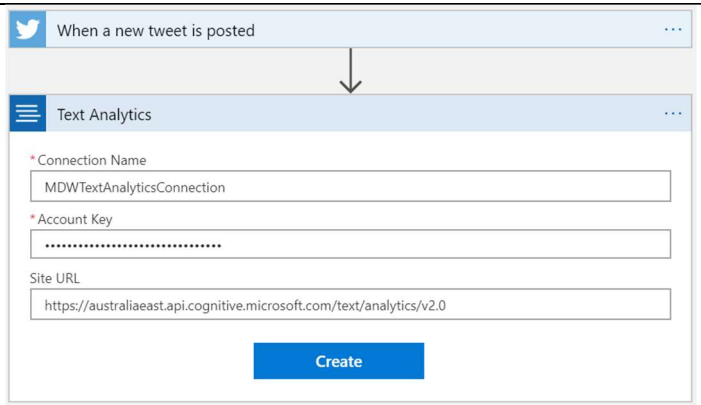
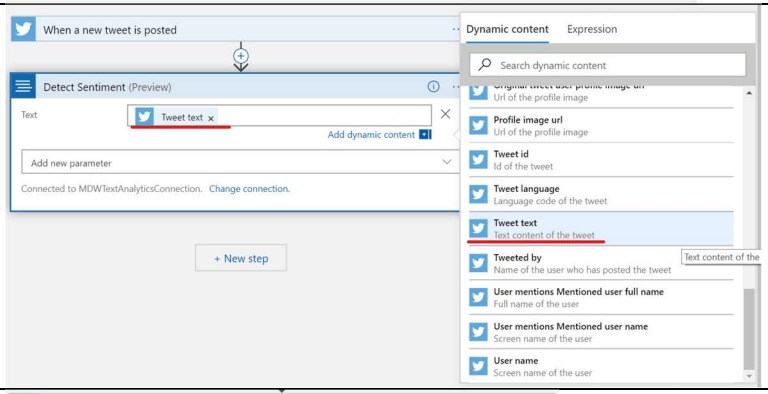
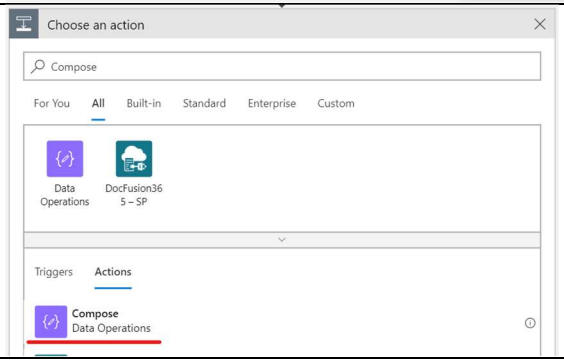
In this section you will create a Logic App to invoke the Twitter API and retrieve tweets for the hashtag #NYC. Tweets will then be formatted into a JSON message and sent to Event Hubs for processing.



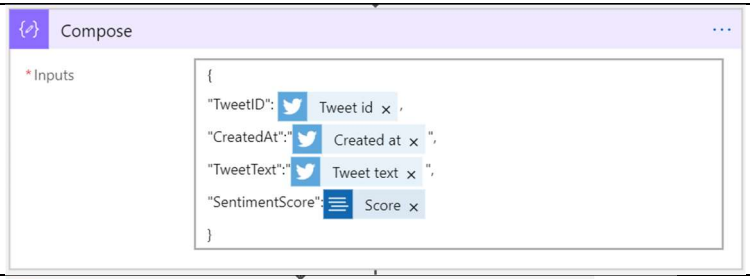
IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, go to the lab resource group and locate the Logic App resource **MDWLogicApp.** 2. On the **MDWLogicApp** menu, click **Logic app designer** to open the design blade.	

3. On the **Logic app designer** blade, scroll down to the section **Start with a common trigger**. 4. Click **When a new tweet is posted**.	
5. On the design surface you will see the Twitter connector. Click **Sign in** and use your personal Twitter account to authenticate.	
6. On the log on screen review the permissions that will be granted to logic apps. If you agree with the permissions granted then enter your credentials and click **Authorise app**. 7. On the **Confirmation Required** page, click **Allow access** to proceed with the authentication process and to grant the right permissions to Logic Apps.	

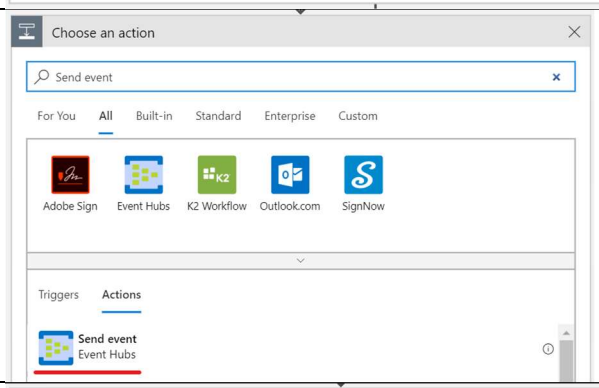
8. Once authenticated with Twitter you will notice the green tick next to your user name. 9. Click **Continue** to configure the Twitter connector.	
10. On the **When a new tweet is posted** properties, enter the follow details: a. **Search text**: #NYC b. **Interval**: 10 c. **Frequency**: Seconds 11. Leave remaining fields with their default values. 12. Click **+ New step** to create a new Logic App task.	
13. On the **Choose an action** box, type "Text Analytics" in the search field. Select **Detect Sentiment (preview)** from the **Actions** tab.	

14. On the Text Analytics properties, type “MDWTextAnalyticsConnection” in the Connection Name field. 15. Open the **Azure Portal** in a new browser tab and copy Endpoint and Key values for **MDWTextAnalytics.** 16. Paste the values in the respective fields in the **Text Analytics** properties. 17. Click **Create**.	
18. In the **Detect Sentiment (preview)** properties, click **Add new parameter**. 19. Select **Text** check box. 20. In the **Text** parameter box, select **Tweet Text** from the **Dynamic content** list. 21. Click **+ New step** to create a new task.	
22. Type “Compose” in the search box and select the **Compose Data Operation.**	

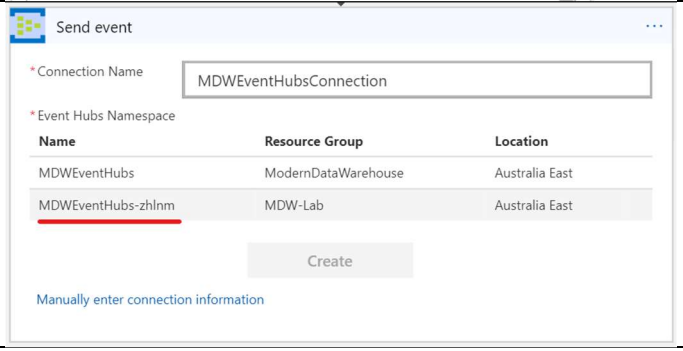
23. On the Compose properties, build a new JSON message using data elements returned by the previous tasks. Your JSON message should look like this.
24. Click **+ New step** to create a new task.



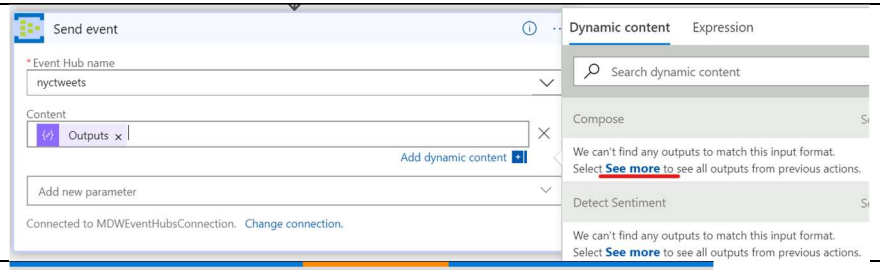
25. On the **Choose an action** box, type “Send event” in the search field. Select **Send Event** from the **Actions** tab.



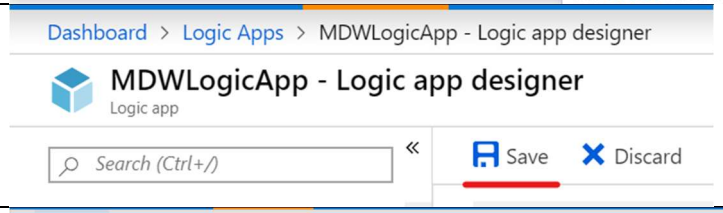
26. On the **Send event** properties, type “MDWEventHubsConnection” in the **Connection Name** field.
27. Select **MDWEventHubs-*suffix*** from the list of **Event Hubs Namespaces**.
28. Select the default access policy **RootManageSharedAccessKey.**
29. Click **Create**.



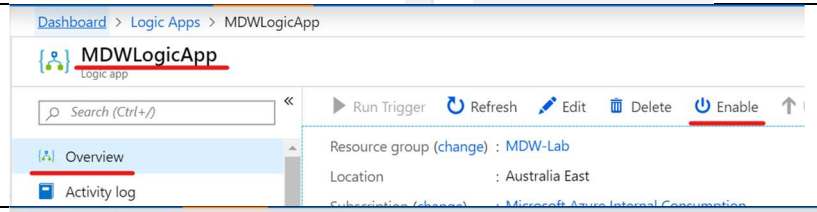
30. On the Send event properties, select nyctweets in the **Event Hub name** field.
31. Click **Add new parameter** and select **Content**.
32. Click on the **Content** field. From the **Dynamic content** pop-up window, select click the **See more** link under **Compose** to display the **Outputs** field.
33. Select **Outputs** as the value for the **Content** field.



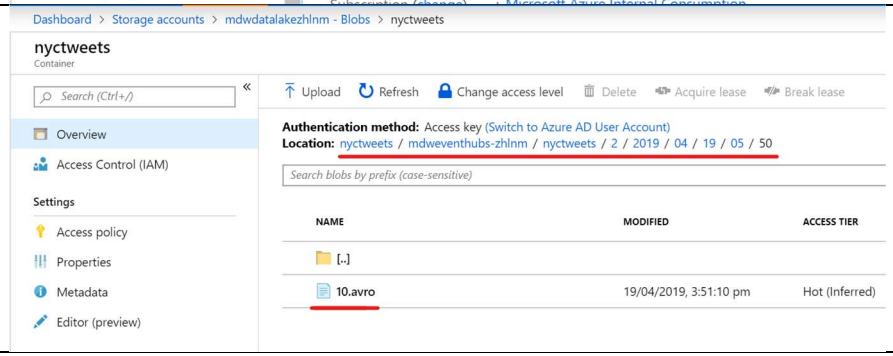
34. Click the Save button to save your Logic App.



35. On the Overview panel, click **Enable**.

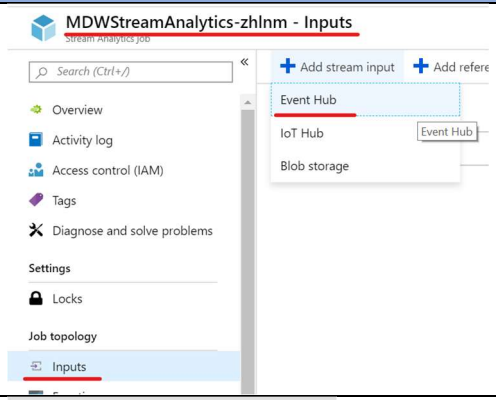
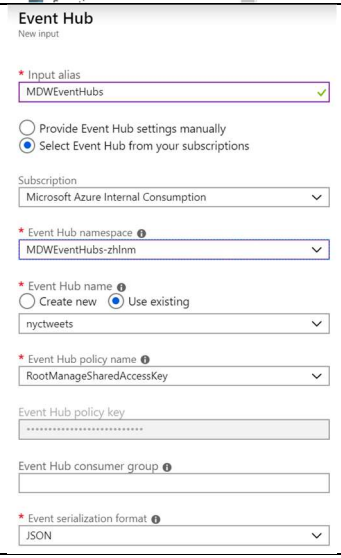


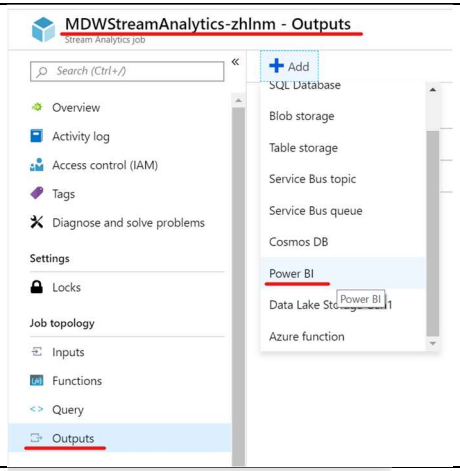
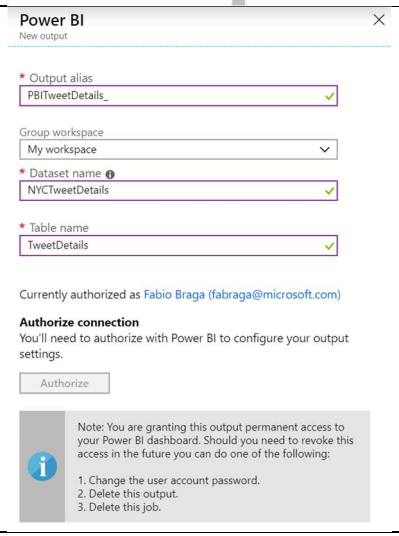
36. In the Azure Portal, navigate to the **MDWDataLake*suffix*** storage account.
37. Wait a couple of minutes and you should be able to see new files being created in the **nyctweets** container.



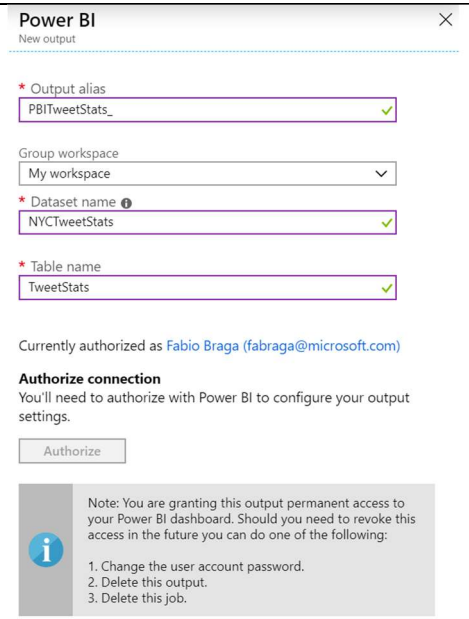
Create and Configure Stream Analytics

In this section you will configure Stream Analytics to perform analytic queries on streaming data sent by Event Hubs and generate outputs to Power BI.

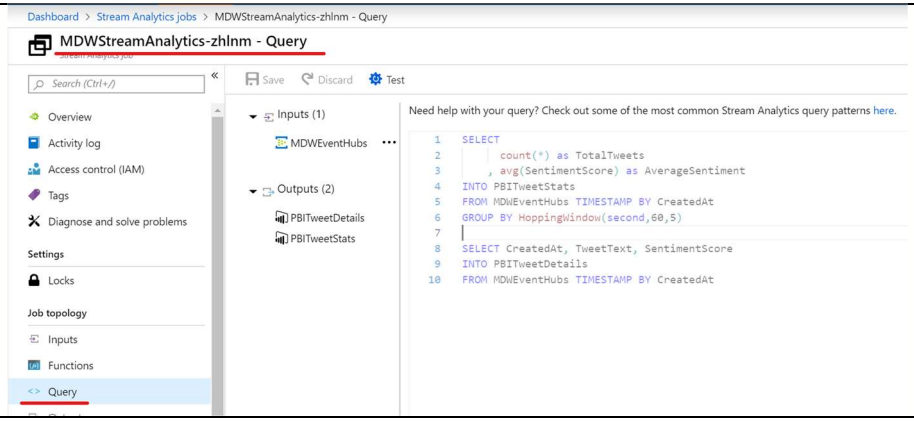
IMPORTANT: Execute these steps on your host computer	
1. In the Azure Portal, go to the lab resource group and locate the Event Hubs resource **MDWStreamAnalytics-*suffix***. 2. On the **Inputs** panel, click **+ Add stream input** button and select **Event Hub** to create a new stream input.	
3. On the **Event Hub New input** blade enter the following details: **a. Input alias:** MDWEventHubs **b. Event Hub namespace:** MDWEventHubs-*suffix* **c. Event hub name > Use existing:** nyctweets 4. Leave remaining fields with their default values.	

5. On the **Inputs** panel, click **+ Add** button and select **Power BI** to create a new stream output.	
6. On the **Power BI New Output** blade, click Authorize to authenticate with Power BI. 7. Once authenticated, enter the following details: a. **Output alias:** PBITweetDetails b. **Group Workspace:** My Workspace c. **Dataset name:** NYCTweetStats d. **Table name:** TweetStats 8. Leave remaining fields with their default values. 9. Click **Save.**	

10. Repeat the process to create another Power BI Output. This time enter the following details:
    a. **Output alias:** PBITweetStats
    b. **Group Workspace:** My Workspace
    c. **Dataset name:** NYCTweetStats
    d. **Table name:** TweetStats
11. Click **Save**.

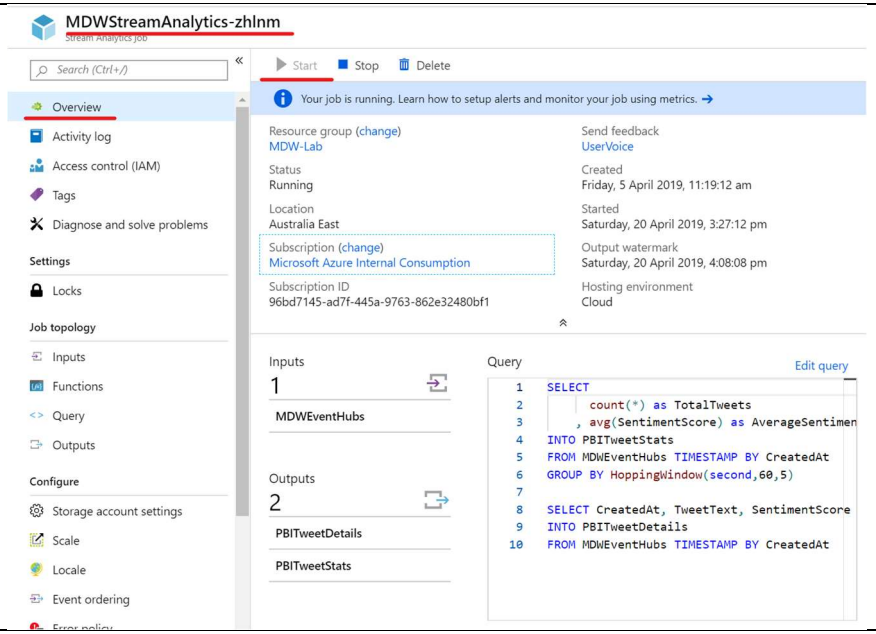


12. On the **Query** panel, note the inputs and outputs you created in the previous steps.



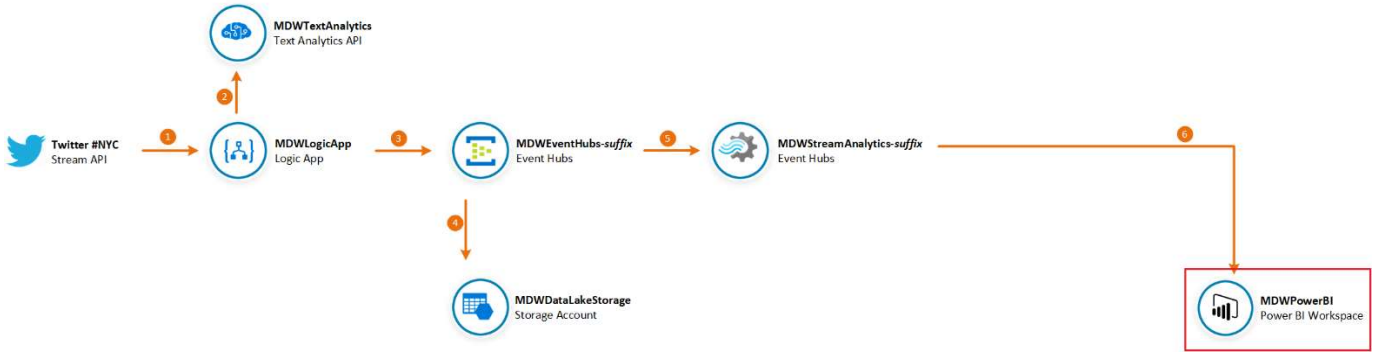
13. Enter the following SQL commands in the query window.
14. Click **Save**.

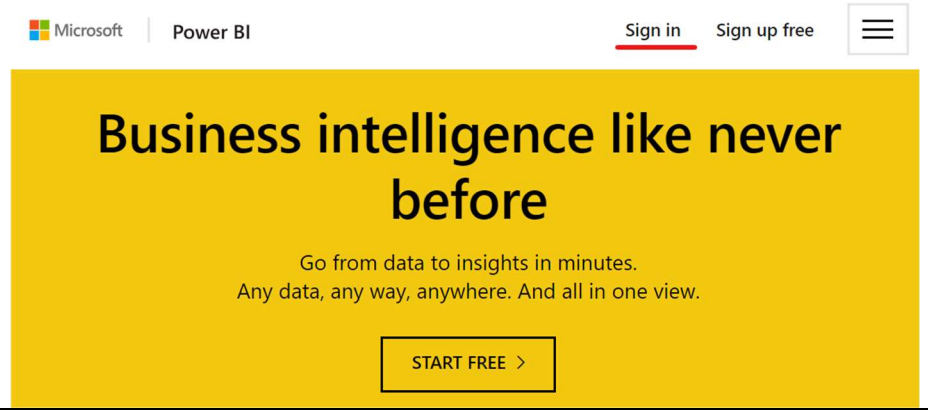
```
SELECT
      count(*) as TotalTweets
    , avg(SentimentScore) as AverageSentiment
INTO PBITweetStats
```

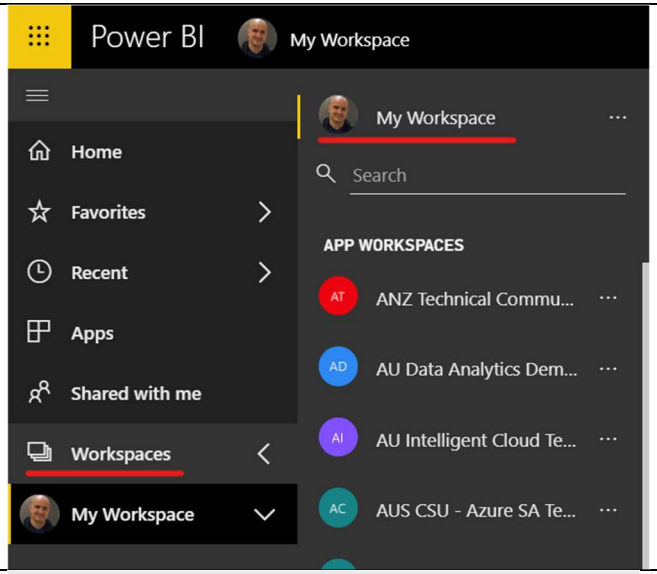
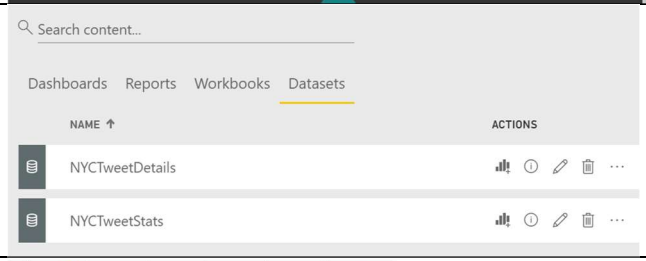
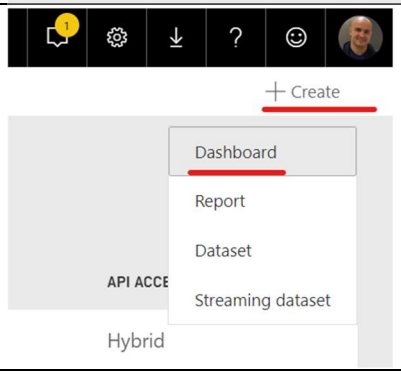
	`` FROM MDWEventHubs TIMESTAMP BY CreatedAt GROUP BY HoppingWindow(second,60,5) SELECT CreatedAt, TweetText, SentimentScore INTO PBITweetDetails FROM MDWEventHubs TIMESTAMP BY CreatedAt ```
15. On the **Overview** panel, click **Start** to start the Stream Analytics job.	

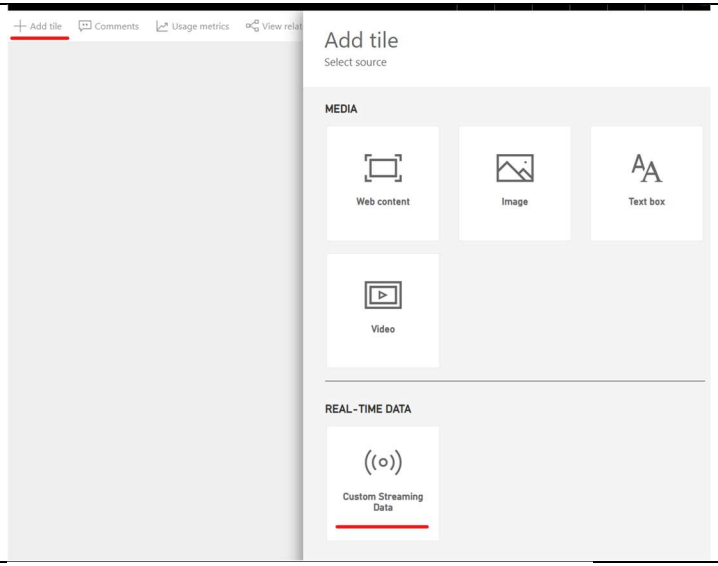
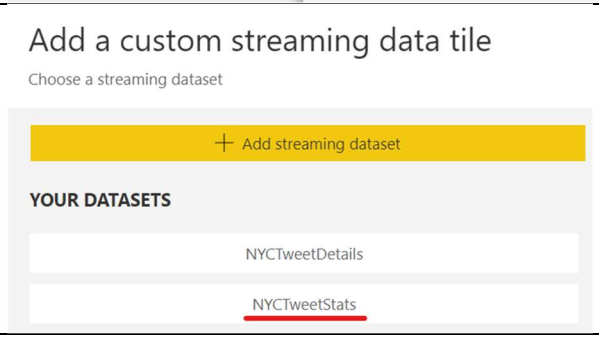
Create Power BI Dashboard to Visualise Real-Time Data

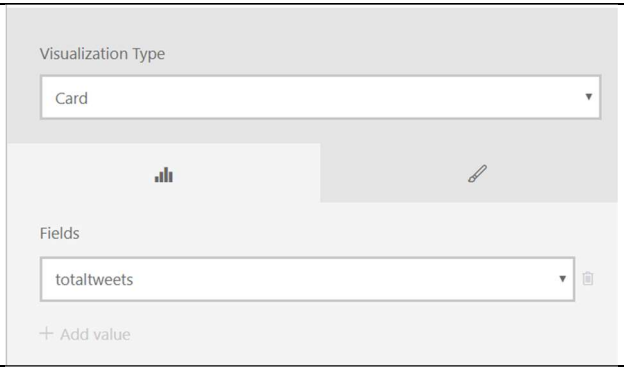
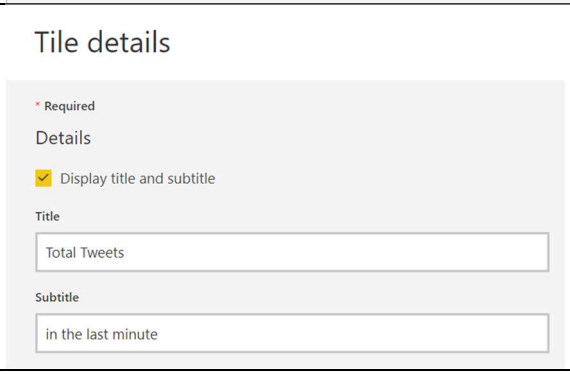
In this section you will log on to the Power BI portal to create a dashboard to visualize real-time Tweeter statistics data sent by Stream Analytics.



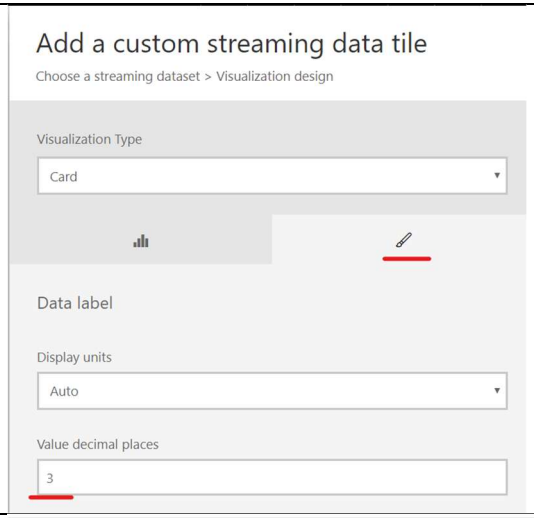
IMPORTANT: Steps to be executed on the **Student's Computer**	
1. Open a new browser tab and navigate to https://www.powerbi.com 2. Enter your credentials to authenticate with the Power BI service.	

3. Once authenticated, open the **Workspaces** menu and click **My Workspace** at the top of the **Workspaces** list.	
4. Navigate to the **Datasets** tab and verify that two datasets have been created by Stream Analytics: **NYCTweetDetails** and **NYCTweetStats.**	
5. On the top right-hand side corner click **+ Create** and then click **Dashboard** from the dropdown menu to create a new dashboard. 6. Type “NYC Tweet Stats” in the **Dashboard name** field and click **Create**.	

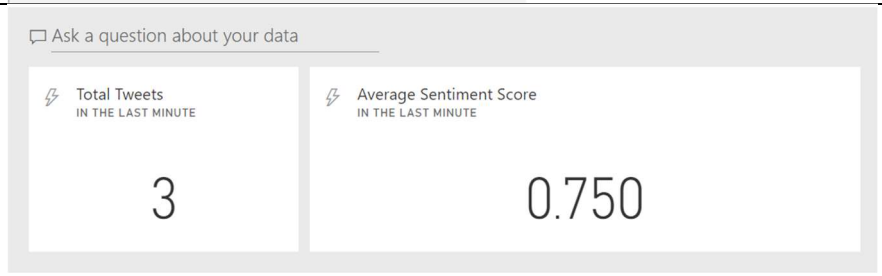
7. Click **+ Add** tile button on the toolbar. 8. On the **Add tile** blade, select **Custom Streaming Data** under the **Real-Time Data** section. 9. Click **Next**.	
10. On the Add a custom streaming data tile blade, select the **NYCTweetStats** dataset. 11. Click **Next**.	

12. On the **Visualization Type** field select **Card**. 13. On the **Fields** field select **totaltweets**. 14. Click **Next**.	
15. On the Tile details blade, enter the following details: a. **Title:** Total Tweets b. **Subtitle:** in the last minute. 16. Leave remaining fields with their default values. 17. Click **Apply**.	

18. Repeat the process to create another tile, this time to display the average sentiment score. Use the following details:
    a. **Dataset:** NYCTweetStats
    b. **Visualization Type:** Card
    c. **Fields:** averagesentiment
    d. **Data Label > Value decimal places:** 3
    e. **Details > Display Title and Subtitle:** Checked
    f. **Details > Title:** Average Sentiment Score
    g. **Details > Subtitle:** in the last minute

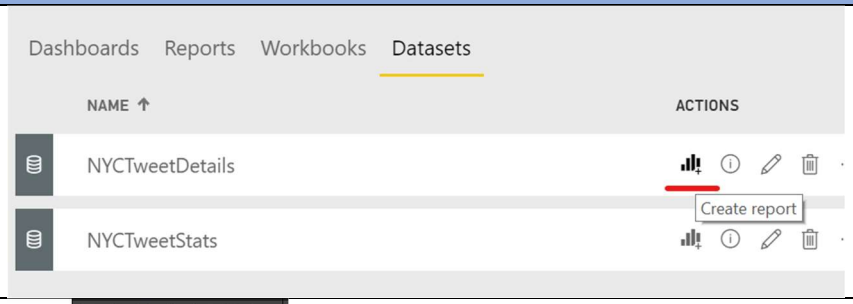
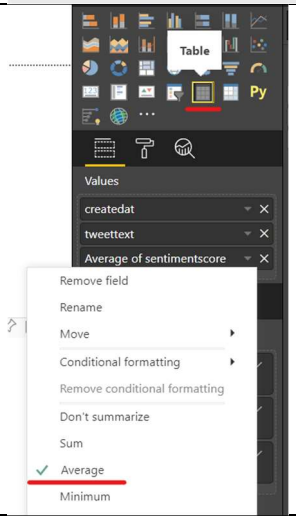


19. You should be able to see the values for both tiles changing every few seconds.
20. Try logging on to Tweeter and posting different messages with the hashtag **#NYC.** Go back to your Power BI dashboard and see the effect your messages had on the average sentiment score.

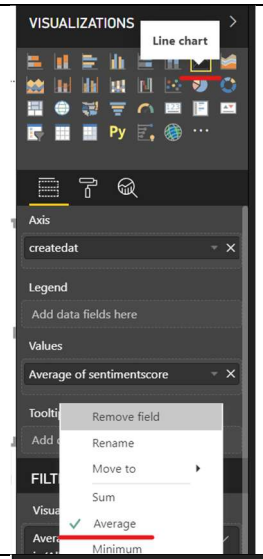


Create Power BI Report to Visualise Real-Time Data

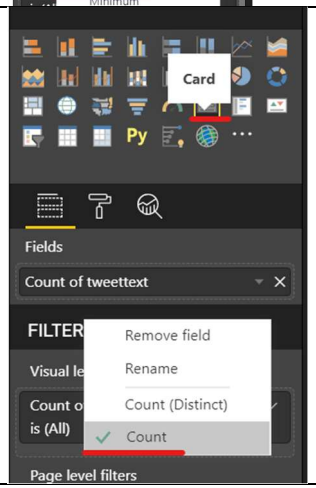
In this section you will log on to the Power BI portal to create a dashboard to visualize real-time Tweeter statistics data sent by Stream Analytics.

IMPORTANT: Steps to be executed on the **Student's Computer**	
1. On the Power BI portal navigate to **My Workspace** and click on the **Datasets** tab again. 2. Click the **Create Report** icon under **Actions** for the **NYCTweetDetails** dataset.	
3. On the report canvas add a **Table** visual and add the following dataset fields to the **Values** property: a. createdat b. tweettext c. sentimentscore 4. Change the default aggregation function for **sentimentscore** to **Average**.	

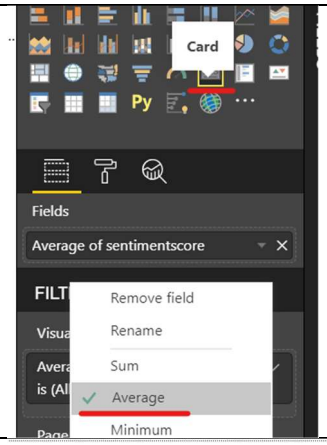
5. On the report canvas add a **Line chart** visual and add the following dataset fields:
    a. Axis: createdat
    b. Values: sentimentscore
6. Change the default aggregation function for **sentimentscore** to **Average**.



7. On the report canvas add a **Card** visual and add the following dataset fields to its **Fields** property:
    a. tweettext
8. Change the default aggregation function for **tweettext** to **Count**.



9. On the report canvas add a **Card** visual and add the following dataset fields to its **Fields** property:
    b. sentimentscore
10. Change the default aggregation function for **sentimentscore** to **Average**.



11. Place your visuals in a way you can see all of them.
12. Click the **Save** button on the toolbar to save your report.
13. Type “NYC Tweet Details” in the **Report name** field.

References

By using this demo/lab, you agree to the following terms:

The technology/functionality described in this demo/lab is provided by Microsoft Corporation for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the demo/lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this demo/lab or any portion thereof.

COPYING OR REPRODUCTION OF THE DEMO/LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS DEMO/LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS DEMO/LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

Feedback

If you give feedback about the technology features, functionality and/or concepts described in this demo/lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE DEMO/LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF DEMO/ LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE DEMO/LAB FOR ANY PURPOSE.

Disclaimer

This demo/lab contains only a portion of new features and enhancements in Microsoft Azure data services. Some of the features might change in future releases of the product. In this demo/lab, you will learn about some, but not all, new features.