Foreword

This book is not the Great American Novel. This book will not change your life.

It might, **however**, help you address your more challenging Azure Data Solution use cases.

In these pages you will find step-by-step instructions for solutions to a variety of use cases that I have experienced as a Microsoft Cloud Solution Architect.

My role title falls short of self-explanation… when asked, I explain that I exist in the grey area between various roles:

- **I sell, but I am not a sales person**… I prefer Microsoft solutions, but if a customer desires a non-Microsoft Product X, I am happy to incorporate it in the architecture
- **I serve, but I am not a service consultant**… I love firsthand development, but there is a role-defined limit to how deep I can dig (and customer resources must take over)
- **I support, but I am not a support engineer**… I will dig into any issue to help with a challenge, but Support Engineers have the knowledge and resources to go far deeper than I

My north star is Customer Success (where "success" means whatever aligns best with customer goals). I collaborate with customers to navigate the diverse landscape of resources necessary to architect a solution that they can implement and own.

This is not a book that you will read from start to finish… rather, I encourage you to **browse directly to those topics that align with your focus and interest**.

Table of Contents

Preparation

This book will not go deep into the many philosophical discussions {e.g., Naming Convention} that arise when implementing Azure; for that sort of content, I recommend the Well-Architected Framework documentation at https://docs.microsoft.com/en-us/azure/architecture/framework/

That said… I might quickly delve into brief discussions of topics like naming conventions when it will directly impact implementation activities.

Naming Convention

When deciding on a naming convention, remember that Azure inconsistently applies naming rules. Depending on the resource that you are trying to instantiate, you might see rules that:

- Prevent names with a dash ("-") or other special characters
- Requires names with only lowercase characters
- Limit total length to 128 characters

In the examples in this book, I choose names based on lowest common denominator {i.e., naming that considers all the "you can't do this or that" rules}, consistently applied to all resources.

I do not adhere religiously to the guidance in the Cloud Adoption Framework; rather, I typically use the combination of values that my customer feels is appropriate to their implementation.

For example:

- Workload / Application: "rchapler"
- Environment: "dev," "test," "prod"
- Resource Type: "dl" (as in Data Lake)

…which results in resource naming like "rchapler-test-dl"

Note: Because I have authored this book over time and in relation to various projects, you might see different naming used in different exercises. I have tried to parameterize these naming choices with "<UseCase>" so you can insert a value appropriate to your work

Resource Instantiation

The instructions in this section are mostly about the use of Azure Portal to manually instantiate the resources necessary to complete the objectives in this book. In some cases, there are also instructions for PowerShell and Bash.

I have tried to keep this section free of repetitive instructions. For example:

- When the interface says **Subscription**, use the dropdown to select the appropriate subscription
- When the interface says **Resource Group**, select the appropriate resource group
- When the interface says **Region** (or something similar like Location), select the same region used by the resource group (or a different value, if appropriate)
- When the interface asks for a **Name**, enter a meaningful name aligned with standards
- When I mention that you need to use a resource {e.g., a Key Vault}, this should imply that you will need to have instantiated a Key Vault

I have minimized image use to hold page count below 300 (the limit for LinkedIn posts).

For each of the sub-sections, start the resource instantiation process by clicking the menu icon in the upper-left of the Azure Portal interface, and then click "**+ Create a resource**."

Application Registration

(aka "Service Principal," "Client")

Create Application Registrations using **Azure Active Directory**. Complete the following steps:

- Navigate to "**Azure Active Directory**"
- Select "**App Registrations**" in the **Manage** group of the navigation pane
- Click "**+ New Registration**"
- Complete the "**Register an application**" form, enter a meaningful **Name** and then click **Register**
- Generate a Key Vault Secret for each Client Identifier and Client Secret

Data Lake

Create Resource

Complete the following steps:

- Complete the "**Create a Storage Account**" form, **Basics** tab.
- Complete the "**Create a Storage Account**" form, **Advanced** tab, check "**Enable Hierarchical Namespace**" in the "**Data Lake Storage Gen2**" grouping.
- Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Container

Complete the following steps:

- Navigate to your Data Lake, and then **Containers** in the "**Data storage**" group
- Click "**+ Container**" and enter a meaningful **Name** value in the resulting pop-out
- Click the **Create** button

Storage Explorer

Complete the following steps:

- Download and install the Microsoft Azure Storage Explorer app from https://azure.microsoft.com/en-us/features/storage-explorer/
- Launch the app and provide Azure credentials.
- Click the "**Connect to Azure storage**…" link
- On the resulting pop-up, click the "**Add an Azure Account**" radio button and then click **Next**
- Authenticate with Azure credentials

Sample Data

Complete the following steps:

- Download sample files from a site like https://filesamples.com/formats/csv
- Return to the "Microsoft Azure **Storage Explorer**" app and navigate to your container
- Click the **Upload** button and select "Upload Files" in the resulting pop-up menu
- Click the ellipses button to the right of the "**Selected Files**" box
- Navigate to **Downloads**, select your sample data files, and then click the **Open** button
- Complete the "**Upload Files**" pop-up, click the **Upload** button

Data Migration Assistant

Complete the following steps:

- Browse to https://www.microsoft.com/en-us/download/details.aspx?id=53595
- Download, open, and install the "**User Installer**" for your platform

Data Share

Note: To demonstrate sharing data to a target, you must instantiate a second ADX instance.

Complete the following steps:

- Complete the "**Create Data Share**" form
- Click the "**Review + create**" button, validate settings, and then click the **Create** button

Sent Shares

Complete the following steps:

- Navigate to your Data Share, and then "**Sent Shares**" in the "**Data Share**" group
- Click "**+ Create**" and then complete the "**Sent Shares**" form
  - On the "**1. Details**" tab, including "**Share Type**" value "**In-place**" and then click the **Continue** button
  - One the "**2. Datasets**" tab, click the "**Add datasets**," then select "**Azure Data Explorer**" on the resulting "**Select dataset type**" pop-out, and finally click **Next**
    - Complete the resulting "**Azure Data Explorer**" pop-out and then click the **Next** button
    - On the next pop-out, check the box next to your cluster and then click **Next**
    - On the final pop-out, click "**Add datasets**" and then **Continue**
  - On the "**3. Recipients**" tab, click "**Add recipient**"
    - In the resulting interface, enter an **Email** value and then click **Continue**
  - On the "**4. Review + Create**" tab, click **Create**

The designated email recipient can expect to receive an invitation email. Clicking the "**View invitation >**" link in the email will open the "**Data Share Invitations**" page where the invitee can complete the form and then click to "**Accept and Configure**."

Received Shares

Complete the following steps:

- Navigate to your Data Share, and then "**Received Shares**" in the "**Data Share**" group
- Click the Datasets tab and then check the box next to your Data Explorer cluster
- Click "**+ Map to target**", complete the resulting pop-out and then click "**Map to target**"

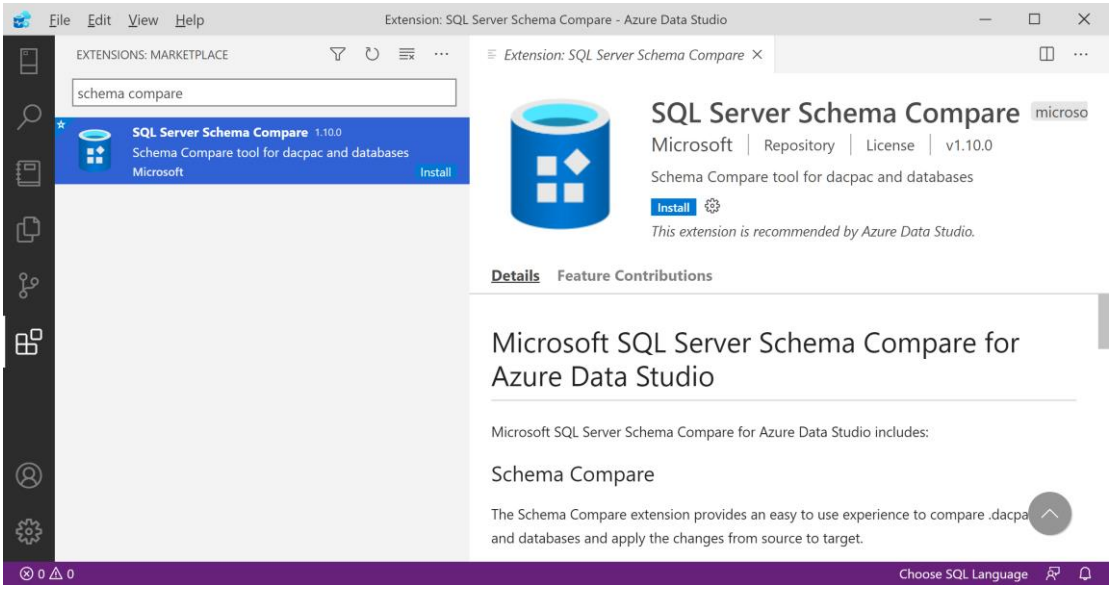
Data Studio

Complete the following steps:

- Browse to https://docs.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver15
- Download, open, and install the "**User Installer**" for your platform

Add Extensions

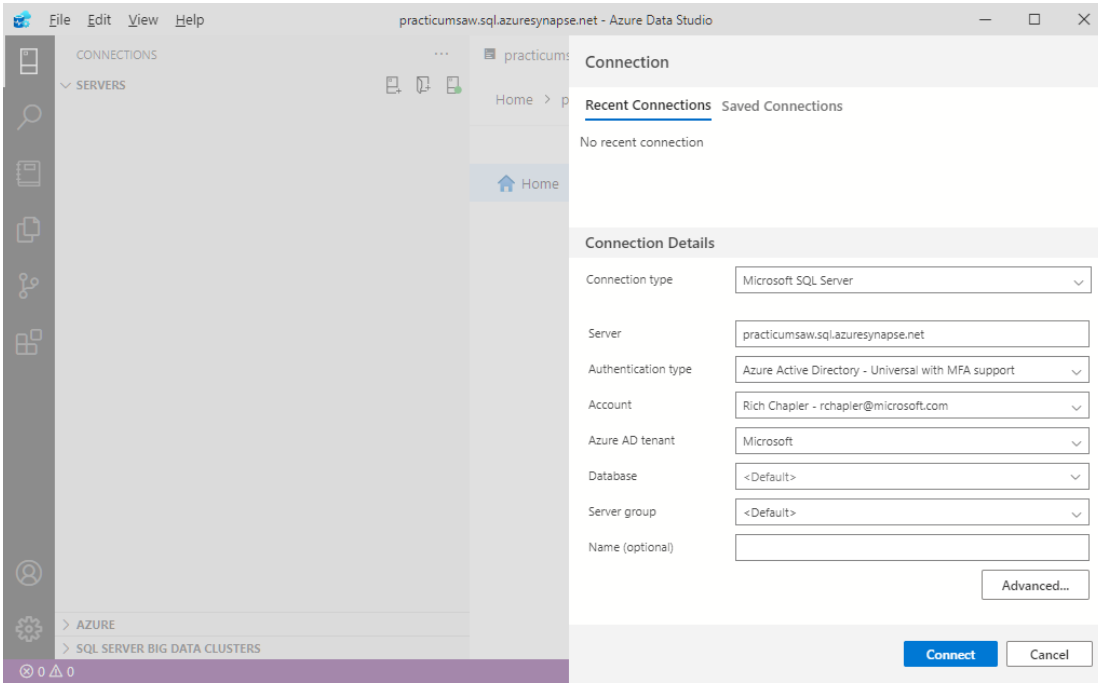
Launch Data Studio and then navigate to **Extensions** in the navigation pane.



Search for "schema compare" and install the "**SQL Server Schema Compare**" extension.

Create Connection

Navigate to **Connections** in the navigation pane.



Click the "**Create Connection**" icon (first of three) to the right of the **SERVERS** section header.

Complete the "**Connection Details**" pop-out, including:

Connection Type	Confirm default value, "**Microsoft SQL Server**"
Server	Example format: **myserver.sql.azuresynapse.net**
Authentication Type	Select "**Azure Active Directory – Universal with MFA support**"
Account	Confirm credentials
Azure AD Tenant	Confirm tenant

Click the **Connect** button.

Databricks

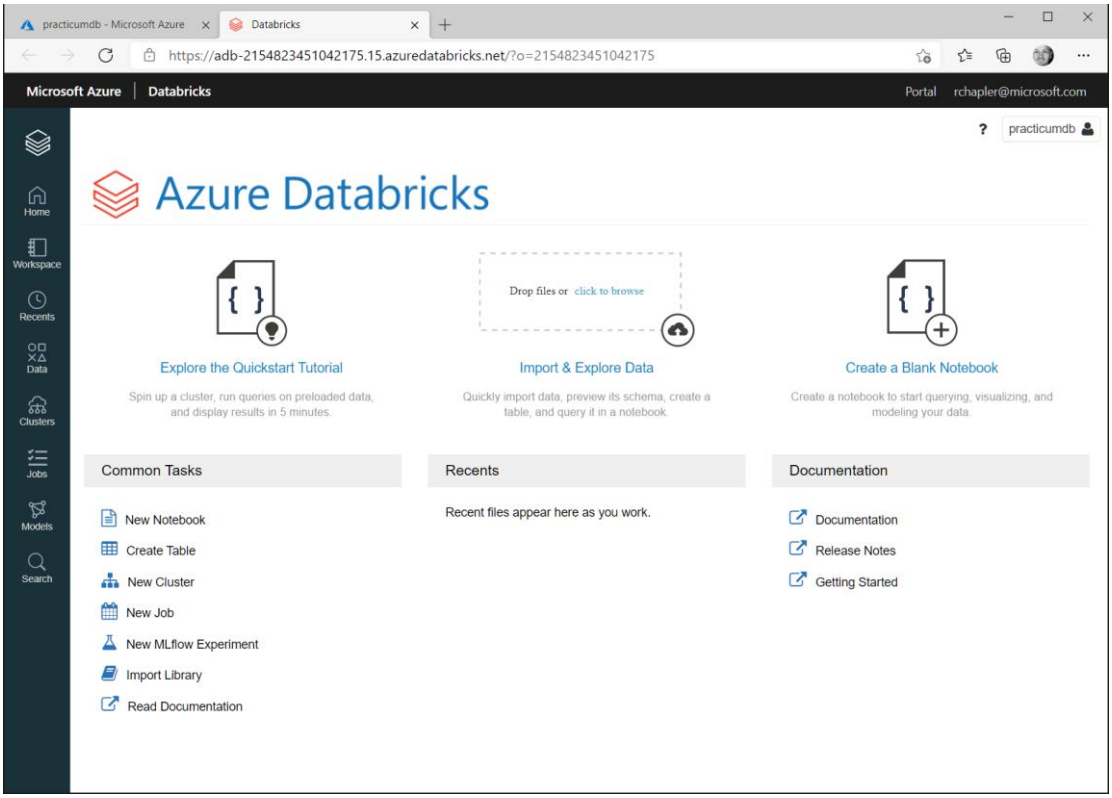
Complete the "**Create an Azure Databricks workspace**" form, including:

Pricing Tier	Select "**Premium**…" to enable features that we will need for this exercise

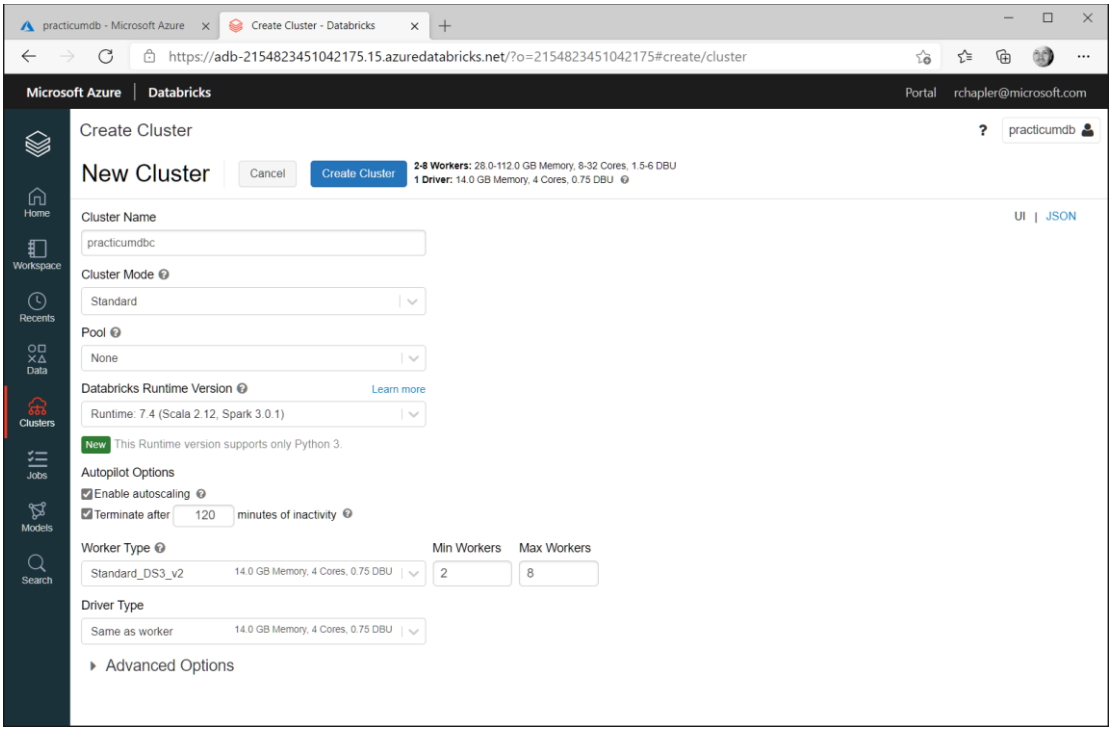
Click the "**Review + create**" button, review configuration, and then click the **Create** button.

New Cluster

Navigate to the **Overview** page in Databricks, click the "**Launch Workspace**" button, and login if required.



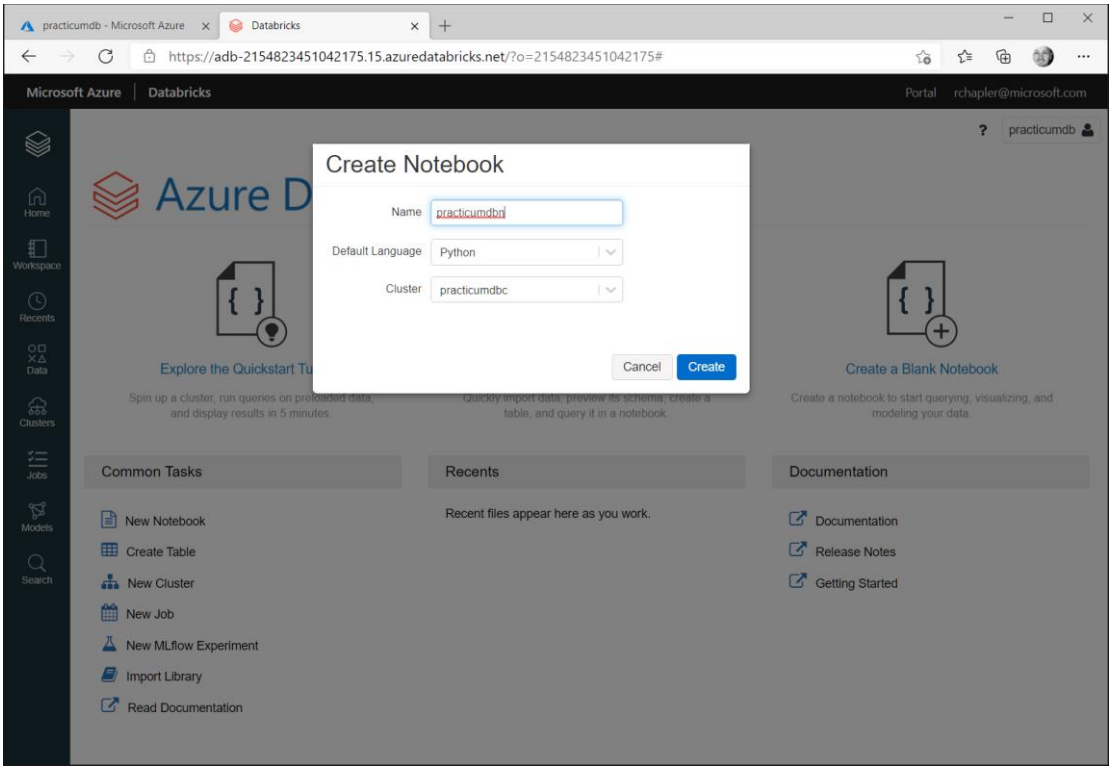
In the "**Common Tasks**" grouping, click "**New Cluster**."



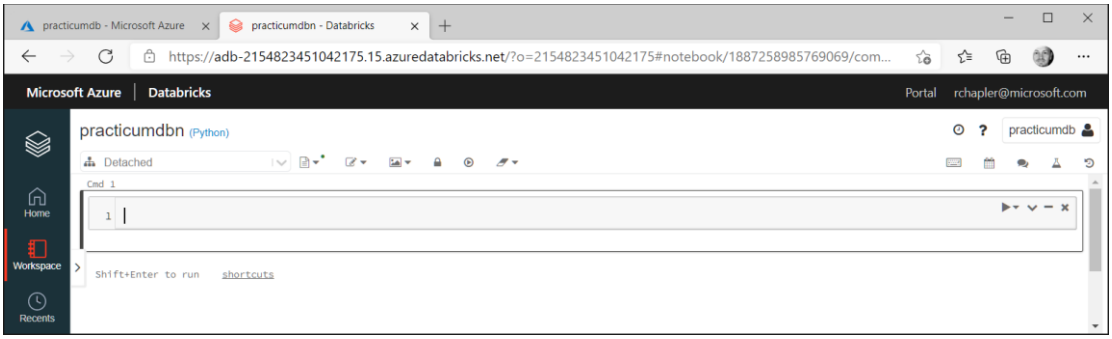
Complete the "**Create Cluster**" form and then click the "**Create Cluster**" button.

New Notebook

Return to the start page and then, in the "**Common Tasks**" grouping, click "**New Notebook**."



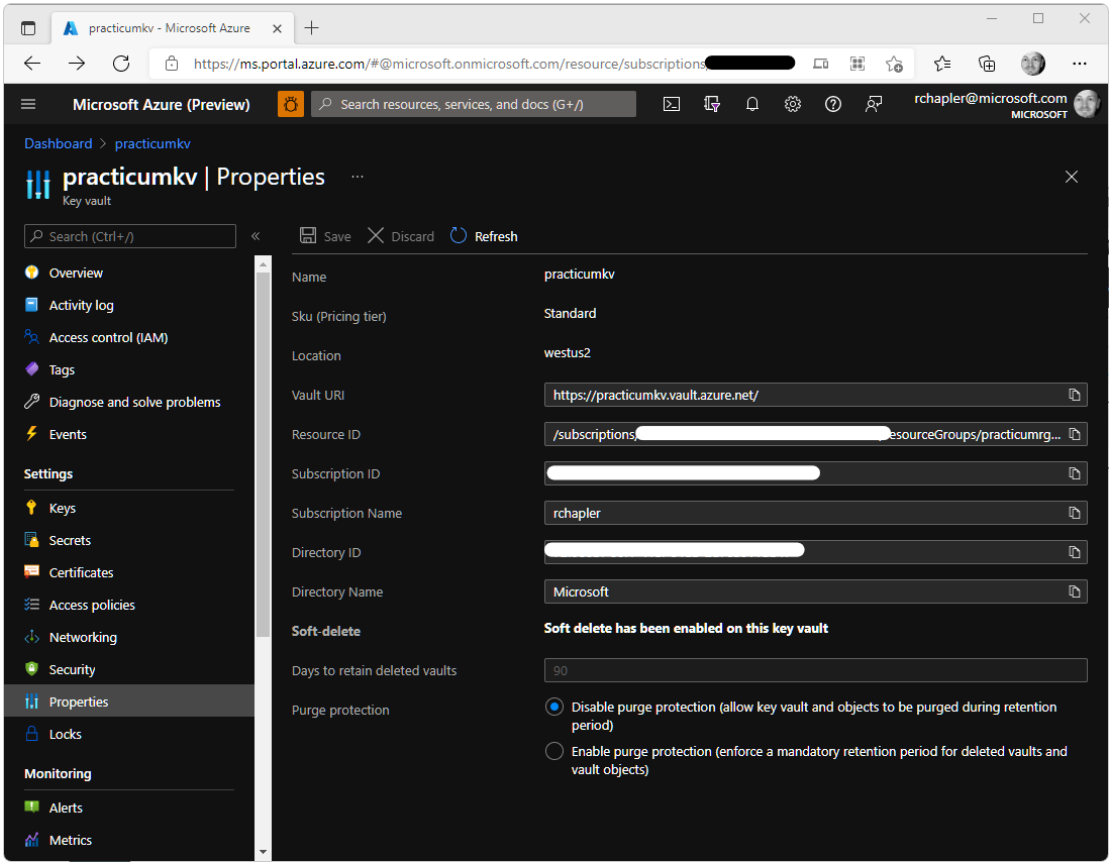
Enter a meaningful name and click the **Create** button.



Make note of the URL {i.e., http://adb-21458...} for use in the next section.

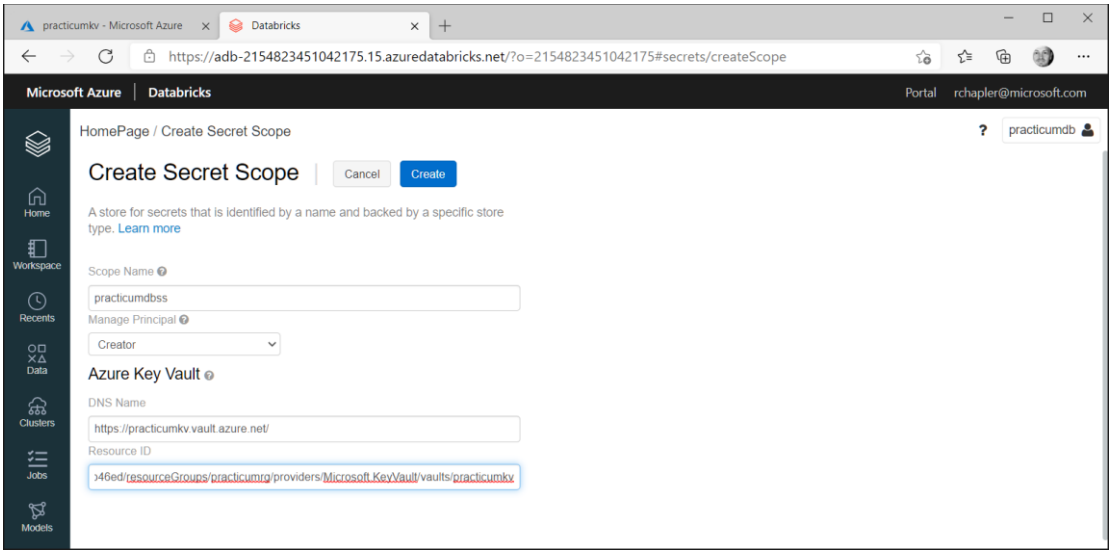
Secret Scope

Navigate to your Key Vault, and then click **Properties** in the **Settings** group of the navigation pane.



Make note of the values in "**Vault URI**" and "**Resource ID**."

You will replace {**databricksInstance**} with the start of the URL in your workspace, from my exercise, example: https://adb-2154823451042175.15.azuredatabricks.net/#secrets/createScope



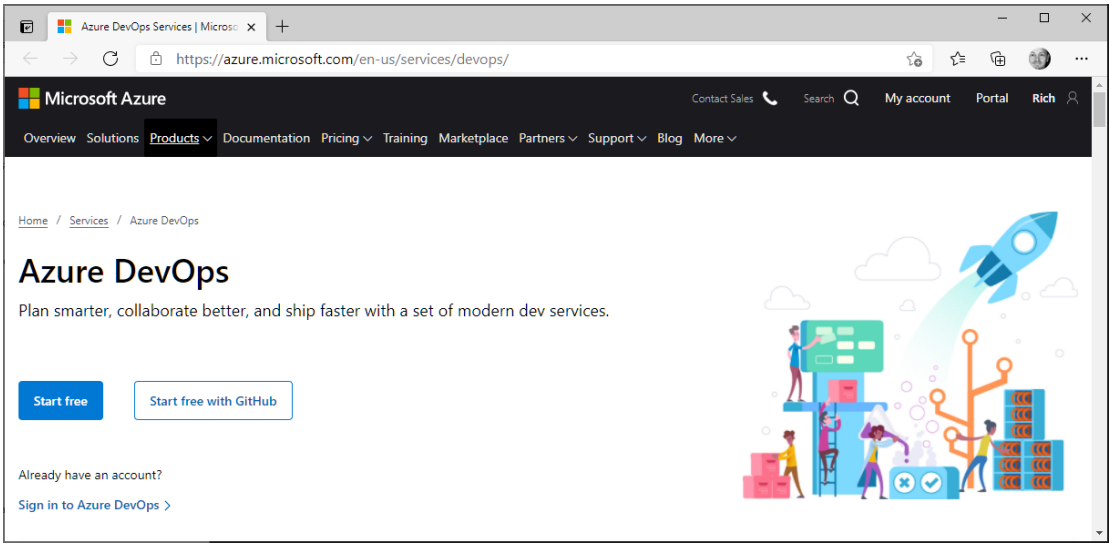
Complete the "**Create Secret Scope**" form, including values for:

DNS Name	Paste the copied "**Vault UI**" value
Resource ID	Paste the copied "**Resource ID**" value

Click the **Create** button.

DevOps

Browse to https://azure.microsoft.com/en-us/services/devops/



Click the "**Start free with GitHub**" button.

On the resulting Azure DevOps form, click the "**New organization**" link on the navigation pane.

Click the **Continue** button.

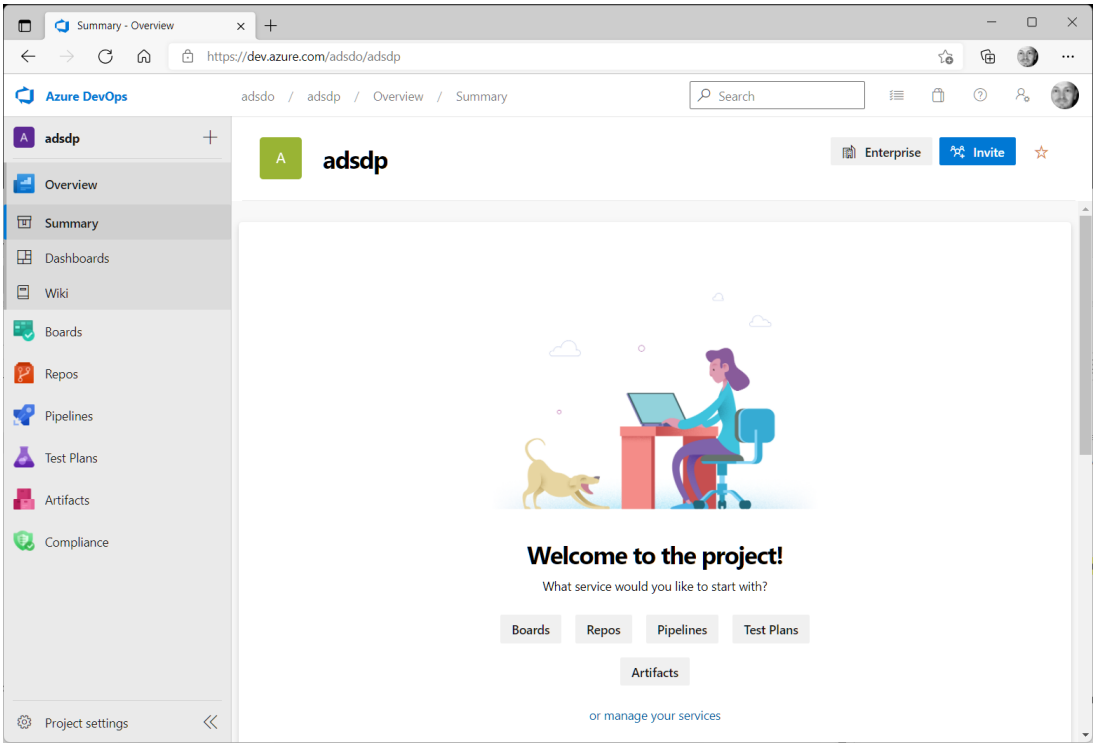
Complete the "**Almost done…**" form and then click the **Continue** button.

Create Project

Complete the "**Create a project to get started**" form, including:

Visibility	Select the value that best aligns with your requirements

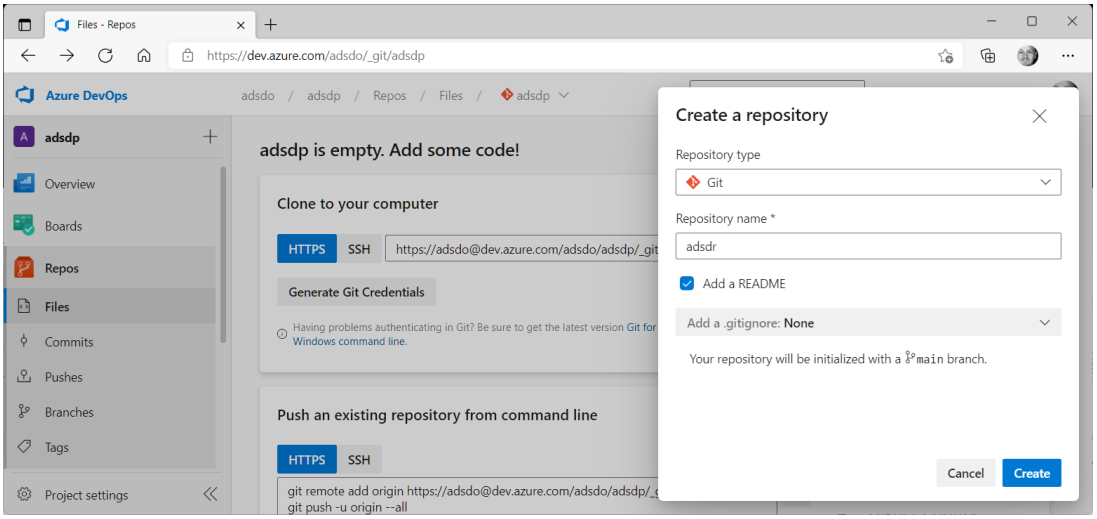
Click the "**+ Create project**" button.



When processing is complete, you will be directed to a "Welcome to the project!" screen.

Create Repository

Click Repos in the navigation pane, then click the "**+**" button at the top-right of the navigation pane and select "**New repository**" from the resulting drop-down menu.



Complete the "**Create a repository**" form, including:

Repository Type	Confirm default selection, **Git**

Click the "**Create**" button.

Event Hub

(aka “Event Hub Namespace”)

Complete the “**Create Namespace**” form, including:

Pricing Tier	Select the pricing tier appropriate for your use case

Click the “**Review + create**” button, review configuration, and then click the **Create** button.

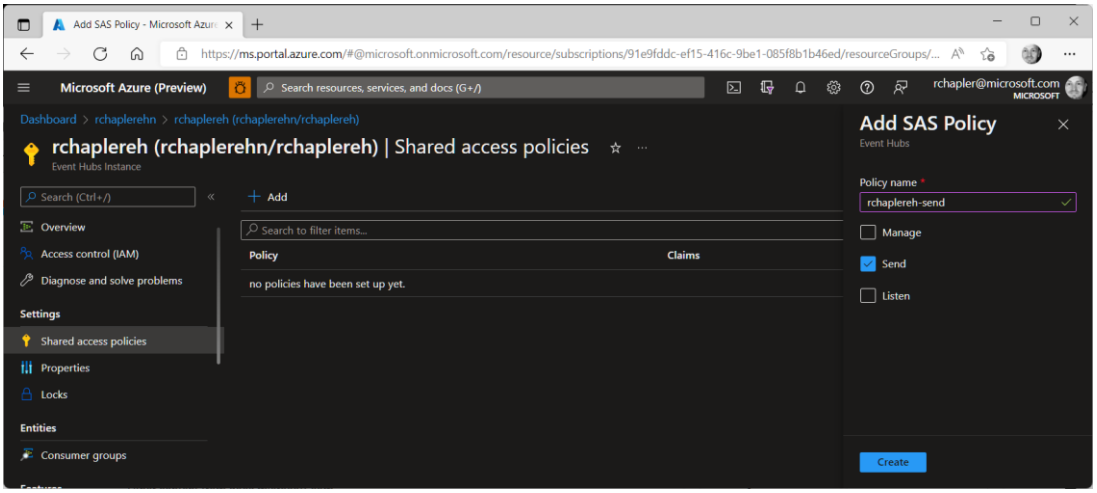
On the Overview page of the newly created Event Hub Namespace, click “+ Event Hub”

Complete the “**Create Event Hub**” form.

Click the “**Review + create**” button, review configuration, and then click the **Create** button.

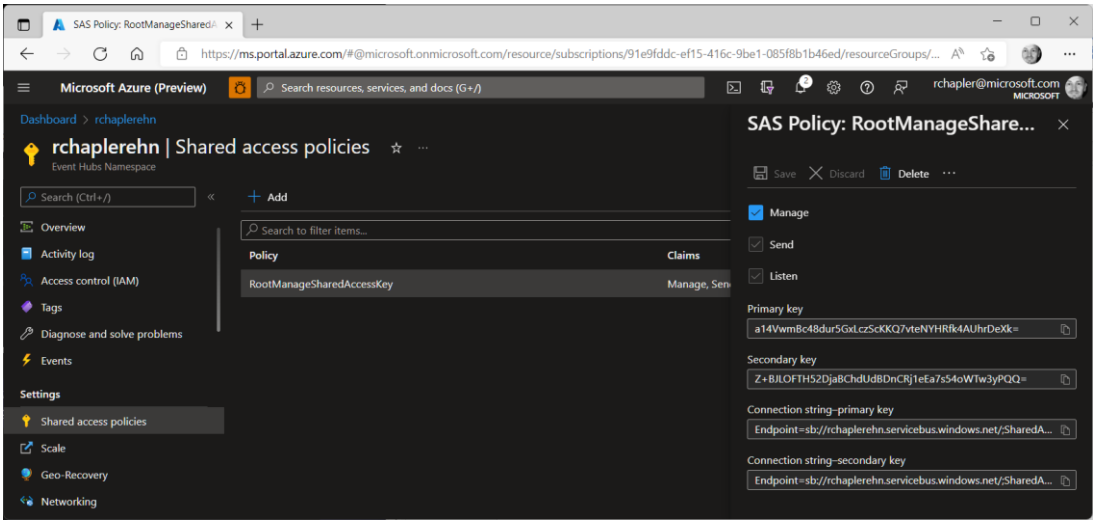
Shared Access Policies

Navigate to the Event Hub Namespace (not a specific Event Hub) and click on “**Shared access policies**” in the **Settings** group of the navigation pane. Click “+ Add.”



On the resulting pop-out, enter a meaningful value for “Policy name” and click to select appropriate permissions {e.g., **Send**}.

Click the **Create** button.

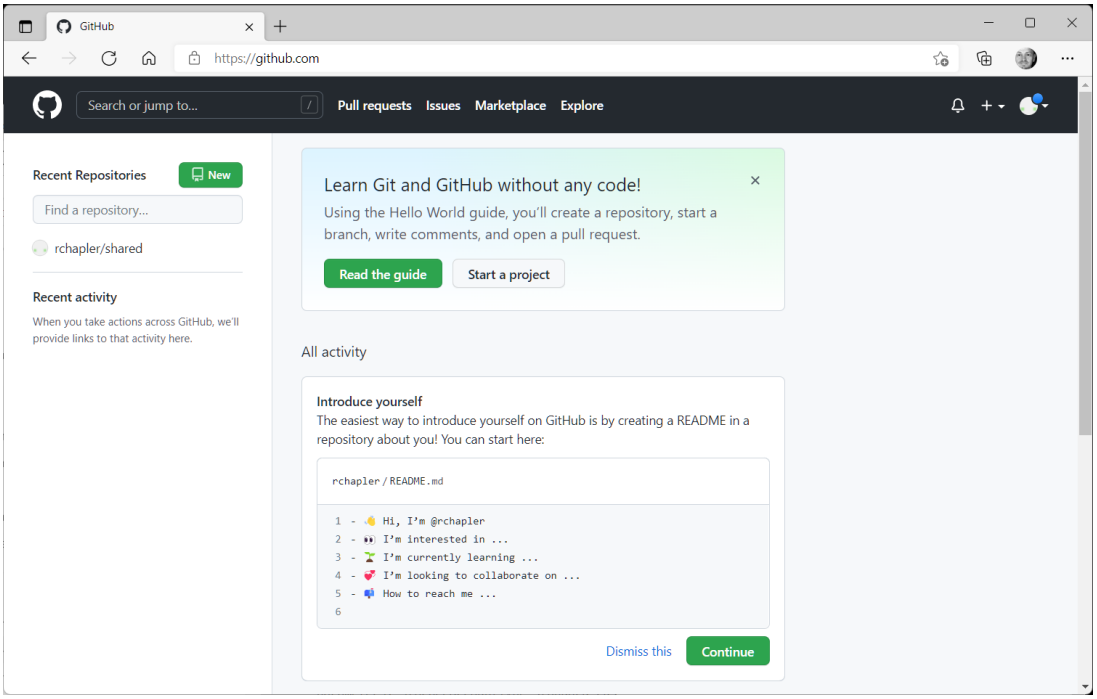


Return to the added policy to copy keys and connection strings.

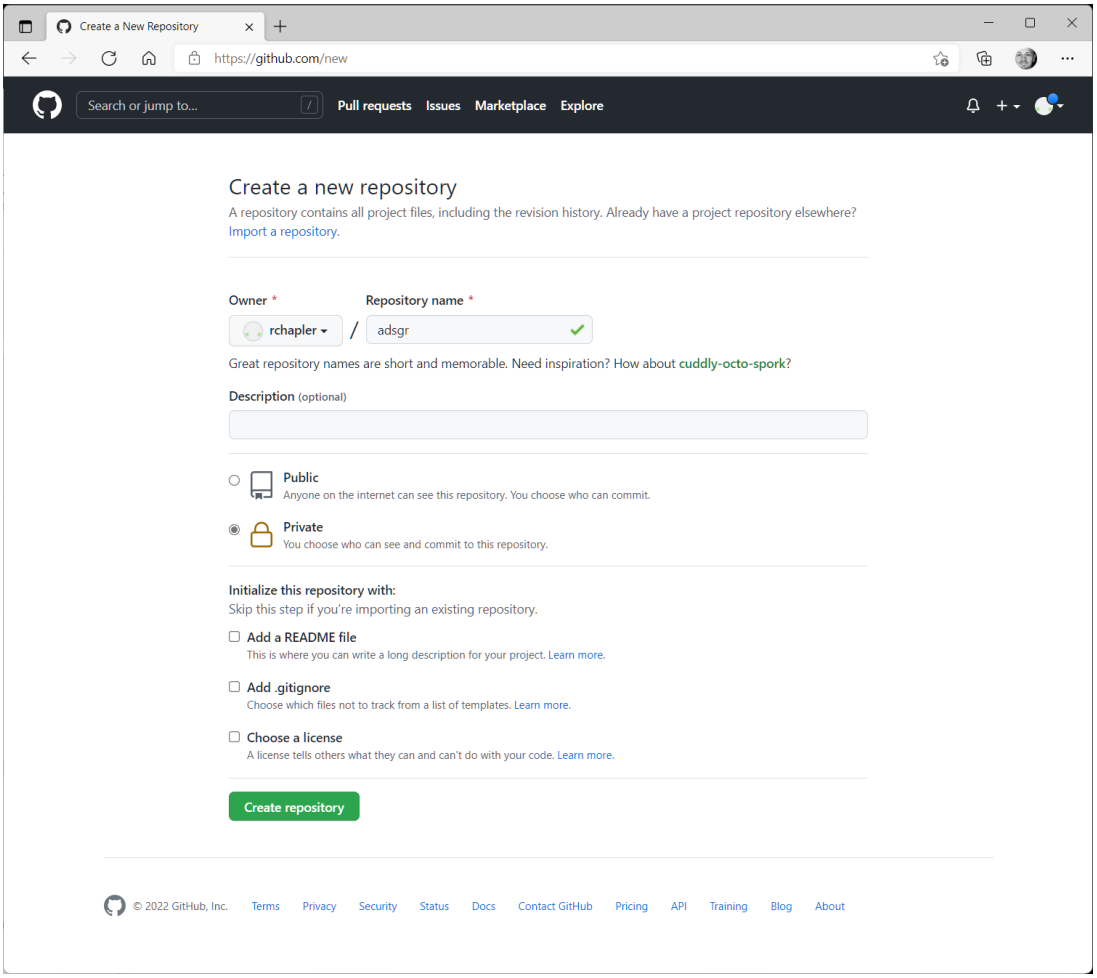
GitHub Enterprise

Browse to https://github.com/login and sign in (or sign up, as needed).

Create Repository



Click the **New** button to the right of "**Recent Repositories**" in the navigation pane.



Complete the "**Create a new repository**" form, including:

Public or **Private**	Choose Private (unless you have a specific reason for using Public)

Click the "**Create repository**" button.

Key Vault

Complete the "**Create Key Vault**" form, including:

Pricing Tier	Confirm default selection, **Standard**
Days to Retain…	Confirm default value, **90**

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Key Vault Secret

Open a new tab and navigate to your Key Vault.

Click **Secrets** in the **Settings** group of the navigation pane and then click the "**+ Generate/Import**" button.

Complete the "**Create a secret**" form, including:

Value	Paste the previously copied secret value

Click the **Create** button.

Log Analytics

Navigate to your Resource Group and click the "**+ Create**" button.

On the resulting "**Create a resource**" form, search for and select "**Log Analytics Workspace**" from the "**Search services and marketplace**" textbox.

On the resulting "**Log Analytics Workspace**" form, click the **Create** button.

Complete the resulting "**Create Log Analytics workspace**" form on the **Basics** tab.

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

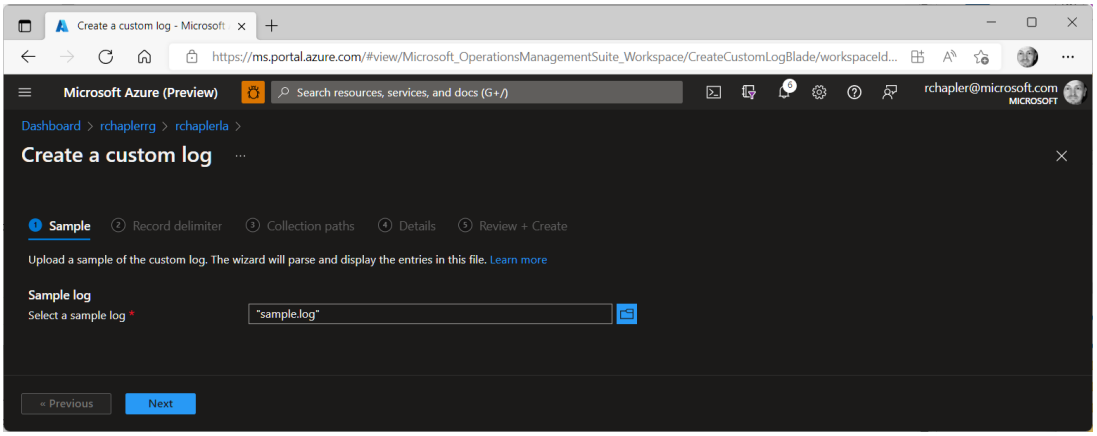
Custom Log

Note: Collection of data from custom logs implies use of an agent on an on-prem device.

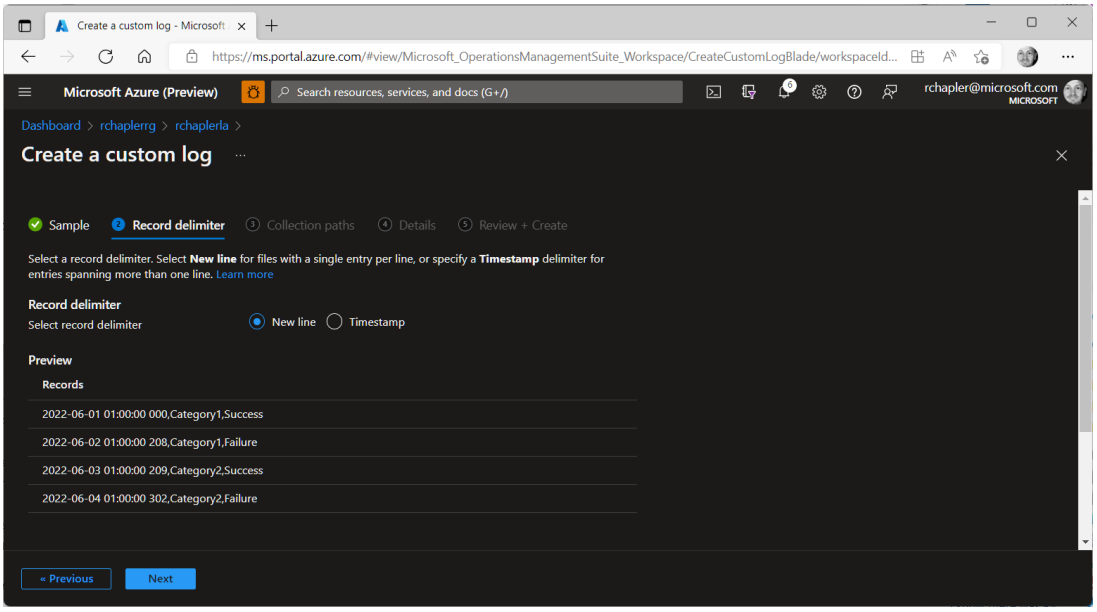
Use a text editor to create a sample log file.

Navigate to Log Analytics and then click "**Custom logs**" in the **Settings** group of the navigation pane.

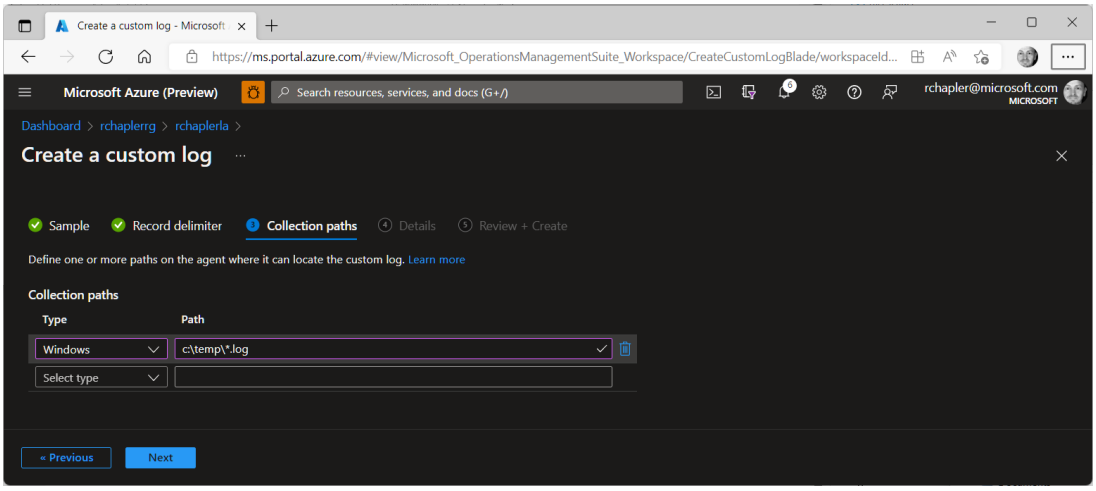
Click "**+ Add custom log**" on the resulting page.



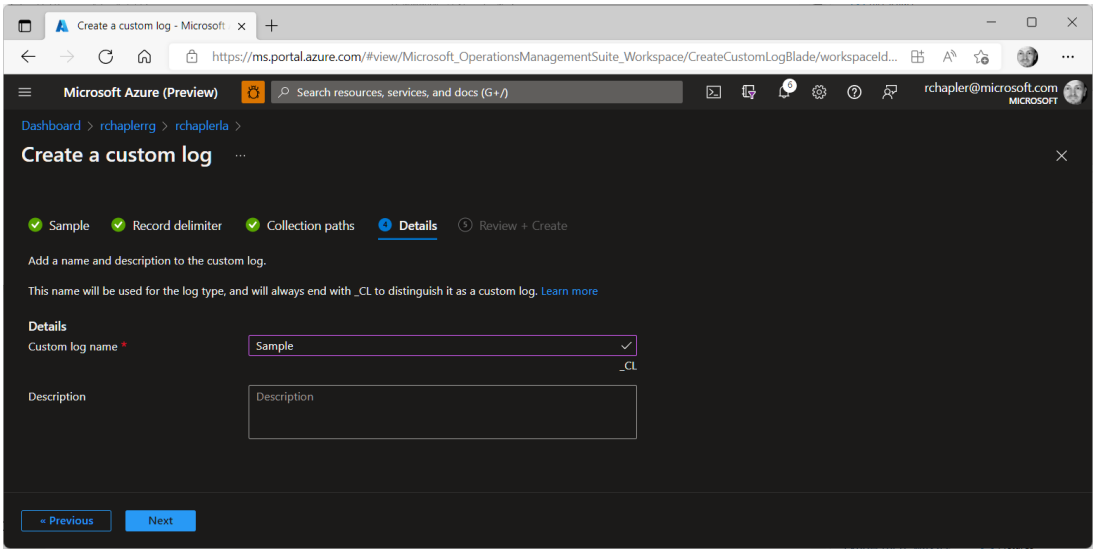
Complete the "**Create a custom log**" form, **Sample** tab, select your sample log file, confirm successful upload, and then click the **Next** button.



Complete the "**Create a custom log**" form, "**Record delimiter**" tab, confirm default setting "**New line**" and then click the **Next** button.



Complete the "**Create a custom log**" form, "**Collection paths**" tab, select values for **Type** and **Path** describing the location where log files will be found (over time). Click the **Next** button.



Complete the "**Create a custom log**" form, **Details** tab, enter a "**Custom log name**" value and then click the **Next** button.

Complete the "**Create a custom log**" form, "**Review + create**" tab, review configuration, and then click the **Create** button.

Logic App

Complete the "**Create Logic App**" form, including:

Plan	**Select the plan type appropriate for your use case** ***Consumption will work in most cases***
Windows Plan…	Click "**Create new**" and provide a name for an App Service Provider
SKU and Size	Click "**Change size**" and pick values appropriate to your use case
Zone Redundancy	Select the value appropriate for your use case ***Disabled** will work in most cases*

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Metrics Advisor

Complete the "**Create Metrics Advisor**" form, including:

Pricing Tier	**Select S0**
Bring Your Own…	Confirm default selection, "No"
I confirm…	Check the box

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Add Permissions to Data Explorer

Navigate to Data Explorer, and then click **Permissions** in the navigation pane.

Click "**+ Add**" on the resulting page

Select **AllDatabasesViewer** from the resulting drop-down menu.

Search for and then click to **Select** Metrics Advisor.

Purview

(aka “Microsoft Purview”, “Azure Purview”)

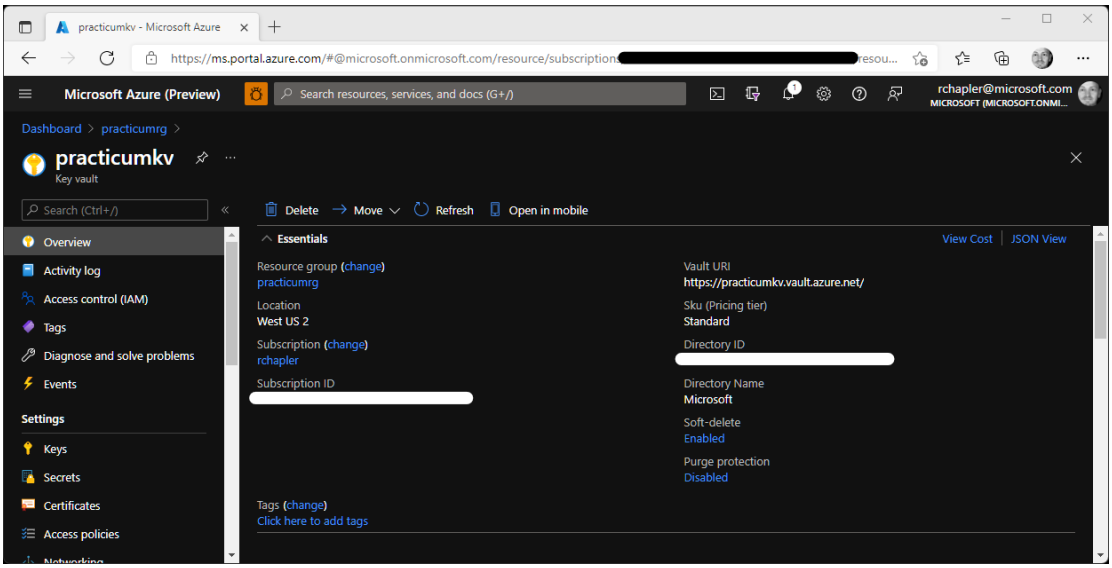
Complete the “**Create Purview account**” form.

Managed Resource Group…	This additional Resource Group will hold ancillary resources created specifically for Synapse *Note: Because you are likely to have more than one managed Resource Group, consider using a consistent naming pattern {e.g., “**<UseCase>mrg-p**” for Purview and “**<UseCase>mrg-x**” for X}*

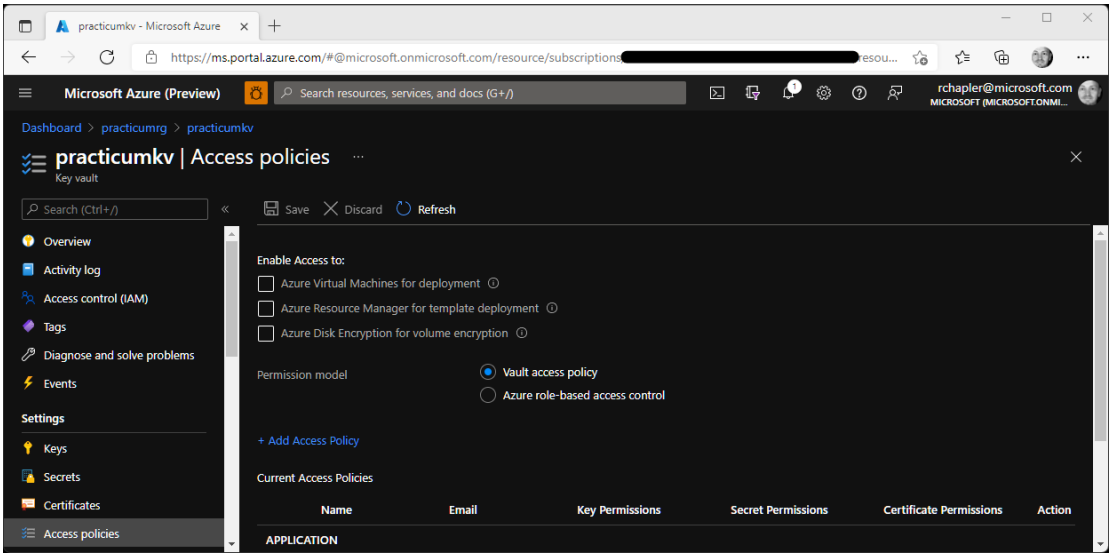
Click the “**Review + Create**” button, review configuration, and then click the **Create** button.

Add Access Policy to Key Vault

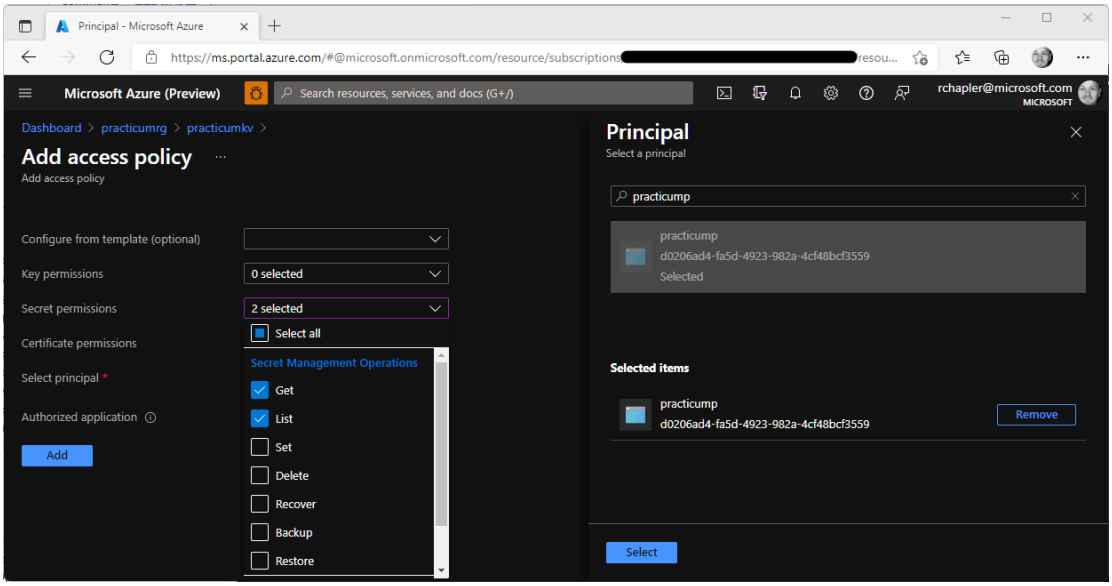
Navigate to the Key Vault.



Click “**Access Policies**” in the **Settings** group of the navigation pane.



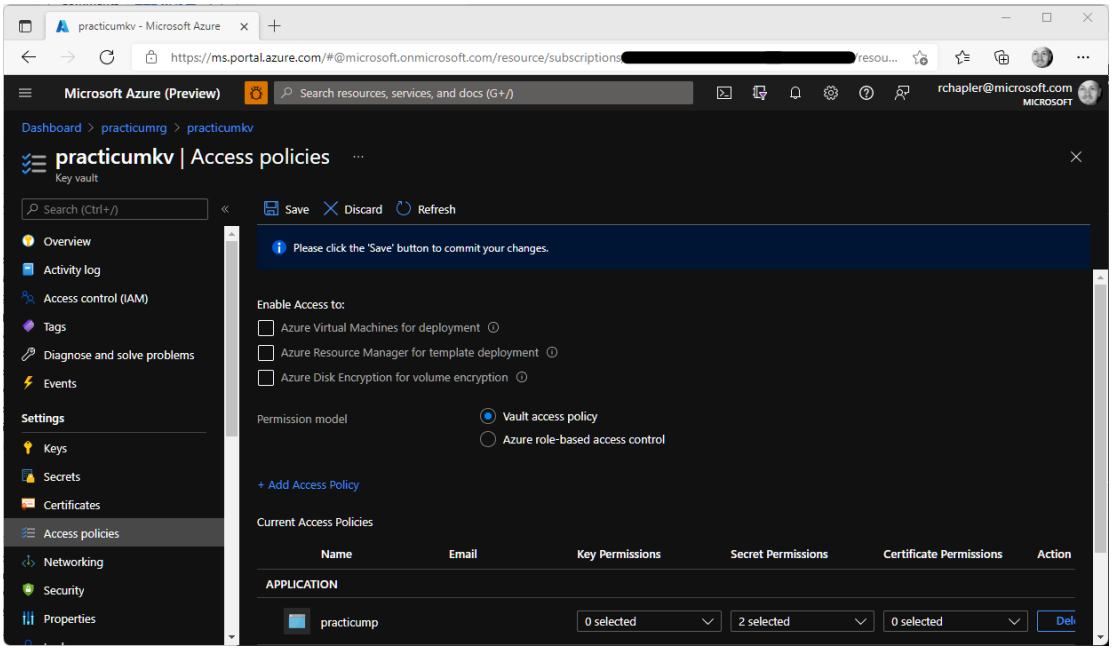
Click the "+ Add Access Policy" link.



Complete the "**Add access policy**" form, including:

Secret Permissions	Select **Get** and **List** in the drop-down menu
Select Principal	Search for, and then **Select** the managed identity for Purview

Click the **Add** button.



Click the **Save** button.

Connect Key Vault

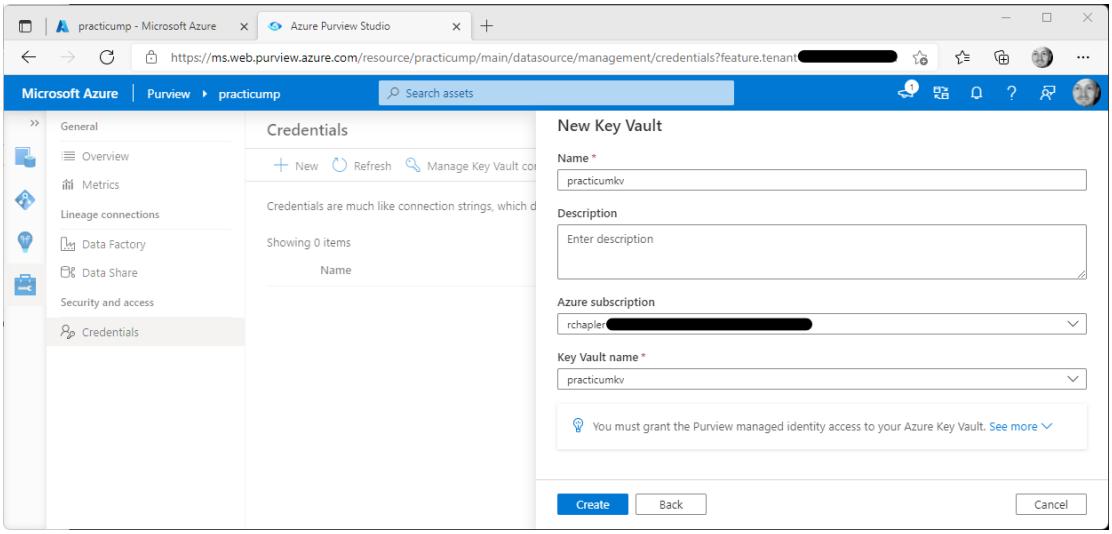
Navigate to Purview and click the "**Open Purview Studio**" button.

Select the **Management** icon on the navigation pane.

Click **Credentials** in the "**Security and access**" group of the resulting menu.

Click the "**Manage Key Vault connections**" button on the resulting page.

Click the "**+ New**" button the resulting pop-out.



Complete the "**New Key Vault**" pop-out, including:

Key Vault Name	Select your Key Vault

Click the **Create** button.

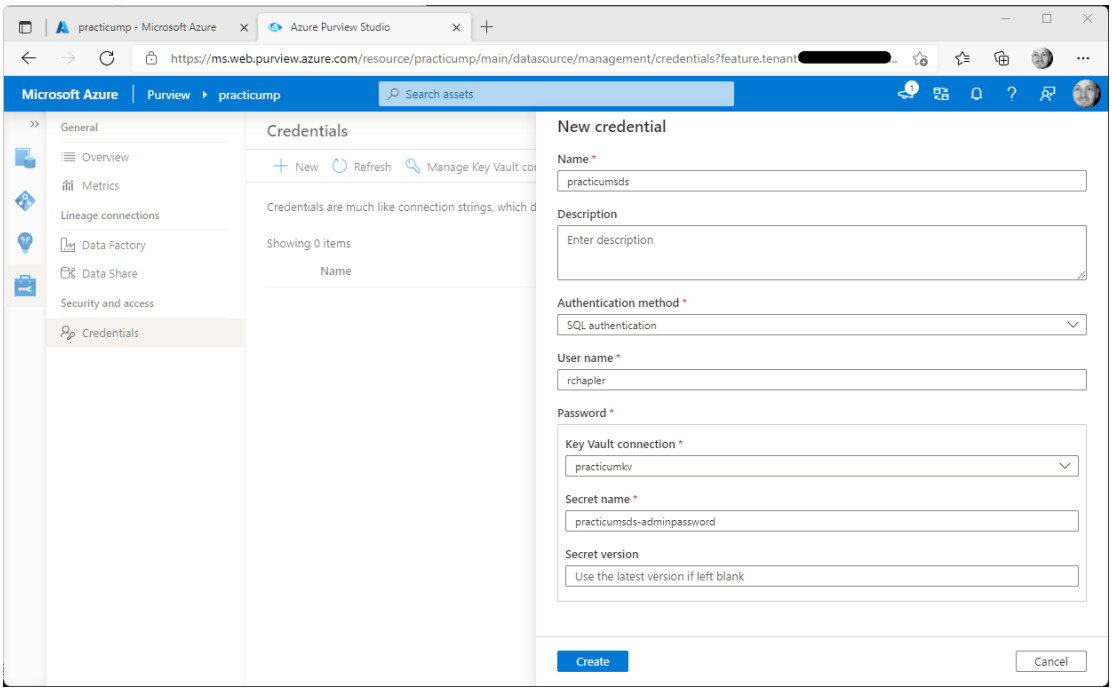
On the resulting "**Confirm granting access**" pop-up, review and then click the **Confirm** button.

Confirm the addition of Key Vault on the **"Manage Key Vault connections**" pop-out.

Click the **Close** button.

Add New Credential

On the **Credentials** form, click the "**+ New**" button.



On the resulting "**New Credential**" pop-out, including:

Authentication	Select "SQL authentication"
User Name	Enter the "server admin login" value used during creation of the SQL Database Server
Key Vault Connection	Select your key vault
Secret Name	Select your secret

Click the **Create** button.

*Note: Purview requires "**Storage Blob Data Reader**" permissions to scan a Data Lake for blobs.*

Resource Group

Complete the "**Create a Resource Group**" form, including a value for:

Region	Select a region appropriate for your situation; take into consideration that: • Some regions {e.g., West US and East US} see higher demand than others • Creation of resources in the same region offers best performance and lowest cost

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Role Assignment

Create using PowerShell

Complete the following steps:

- Navigate to Cloud Shell (mount to new / existing storage, if required)
- Select **PowerShell** from the "**Select Environment**" drop-down menu
- Update and execute the following command to create a Service Principal and give it Contributor access to the subscription:

```
az ad sp create-for-rbac --name "<UseCase>ar" --role contributor --scopes
/subscriptions/{Subscription GUID}
```

```
az ad sp create-for-rbac --name "rchaplerar" --role contributor --scopes
/subscriptions/91e9fddc-ef15-416c-9be1-085f8b1b46ed/resourceGroups/rchaplerrg
```

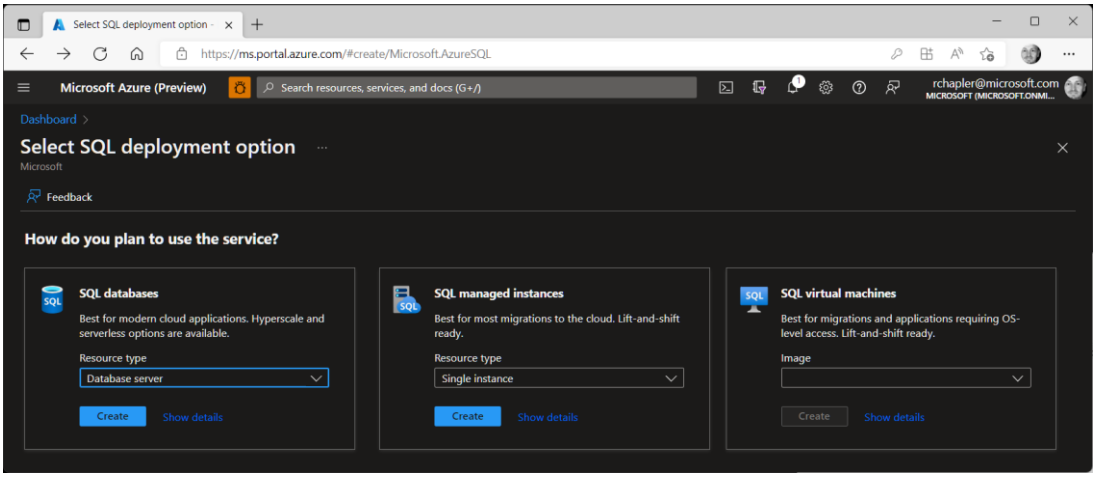
You can expect a response like the following redacted example:

```
{
  "appId": "{ClientId}",
  "displayName": "rchaplerar",
  "password": "{Password}",
  "tenant": "{TenantId}"
}
```

SQL

(aka “Azure SQL”, “Azure SQL Server”, “Azure SQL Database”)

Create Database Server



Complete the “**Select SQL deployment option**” form, “**SQL databases**” grouping, select “**Database Server**” from the Resource Type drop-down menu and then click the **Create** button.

Complete the “**Create SQL Database**” form, **Basics** tab, including:

Authentication Method	**Select “Use only Azure Active Directory (Azure AD) authentication”** ***Note: Though “SQL authentication” may be appropriate to some use cases, I do not typically recommend it***
Set Azure AD Admin	Click the “**Set admin**” link and on the resulting pop-out, search for and select the appropriate item

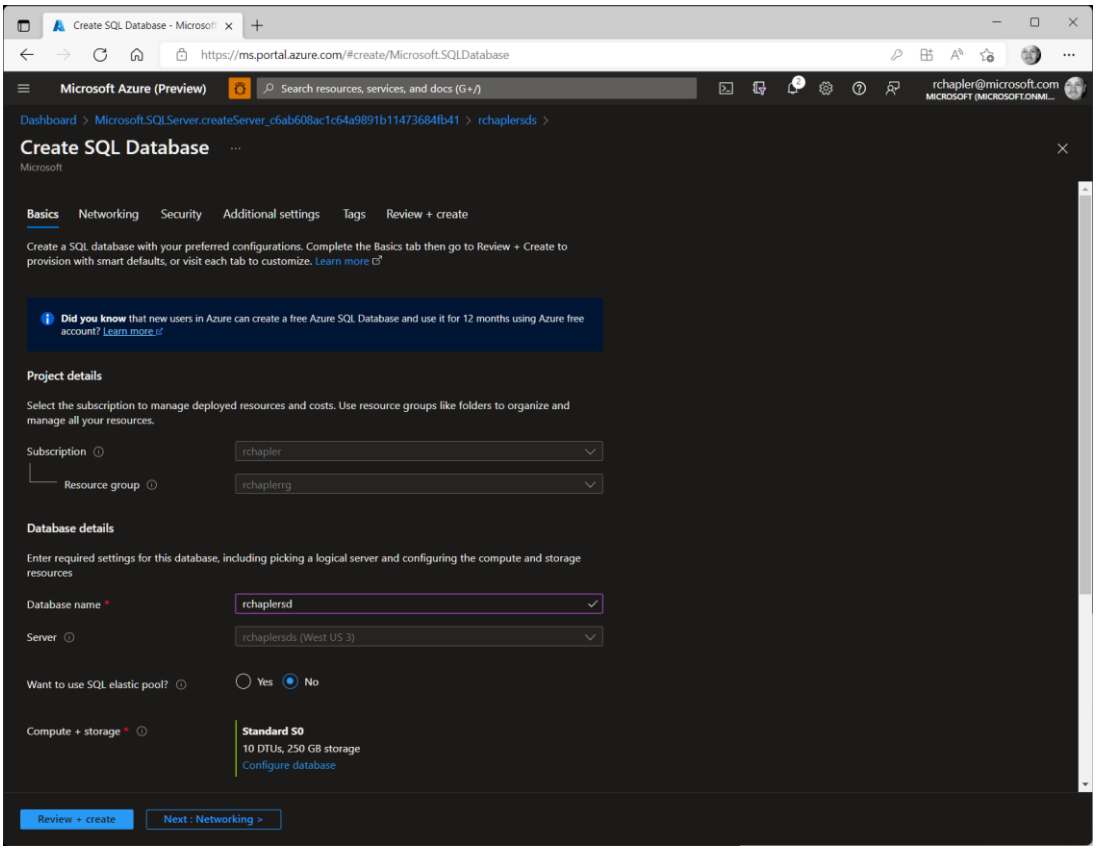
Navigate to the **Networking** tab, and including:

Allow Azure services…	Select **Yes** *Note: This setting may be inappropriate for some use cases*

Click the “**Review + create**” button, review configuration, and then click the **Create** button.

Create Database

On the **Overview** page of the new Azure SQL Database Server, click the "**+ Create database**" button.



Complete the "**Create SQL Database**" form, **Basics** tab, including:

…SQL elastic pool?	Confirm default value, **No**
Compute + Storage	Click the "**Configure database**" link and review / update default values on the resulting **Configure** page *Note: Your use case should drive your choice, but barring good reason, I typically stay with default Service Tier and minimum values for DTUs and Data Max Size*
…Storage Redundancy	Select appropriate value

Add a sample database, if needed, on the "**Additional settings**" tab.

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Create User

Note: This can be used for both System-Assigned Managed Identities and User-Assigned Managed Identities

Navigate to your Azure SQL Database and select "**Query Editor**…" from the navigation pane.

Modify and then execute the following T-SQL:

```
CREATE USER [<UseCase><ResourceAbbreviation>] FROM EXTERNAL PROVIDER

ALTER ROLE db_datareader ADD MEMBER [<UseCase><ResourceAbbreviation>]
```

Storage Account

Complete the "**Create Storage Account**" form, including:

Performance	Confirm default radio button selection, "Standard"
Account Kind	Confirm default selection, "StorageV2 (general purpose v2)"
Replication	Confirm default selection, "Read-access geo-redundant storage (RA-GRS)"

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Synapse

(aka "Azure Synapse Analytics")

Create Synapse Workspace

Complete the "**Create Synapse workspace**" form, including:

Managed Resource Group	This additional Resource Group will hold ancillary resources created specifically for Synapse *Note: Because you are likely to have more than one managed Resource Group, consider using a consistent naming pattern {e.g., "**<UseCase>mrg-s**" for Synapse and "**<UseCase>mrg-x**" for X}*
Select Data Lake Storage Gen2	Confirm default, "**From subscription**"
Account Name	Select an existing data lake or click the "**Create new**" link to create a new data lake
File System Name	Select an existing data lake or click the "**Create new**" link to create a new data lake file system
Assign myself…	Check to assign necessary permissions

Click the "**Next: Security >**" button and then select the "**Use only Azure Active Directory (Azure AD) authentication**" radio button.

Note: Though local authentication may be appropriate to some use cases, I do not typically recommend it

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Configure Repository

Navigate to Synapse Studio, click the **Manage** navigation icon, select "**Git configuration**" from the "**Source control**" grouping in the resulting navigation and then click the "**Configure**" button.

On resulting "**Configure a repository**" pop-out, including:

Repository Type	Select "**Azure DevOps Git**" (or "GitHub…") from the drop-down menu
Azure Active Directory	Select the value appropriate for your organization

Click the **Continue** button.

On the second "**Configure a repository**" pop-out, including:

Azure DevOps Organization…	Select your DevOps account
Project Name	Select your DevOps project
Repository Name	Select the repo created with your DevOps project
Collaboration Branch	Drop-down menu, click "**+ Create new**", enter an appropriate name in the resulting pop-up, and click **Create**
Publish Branch	Confirm default, **workspace_publish**
Root Folder	Confirm default, **/**
Import Existing Resources…	Unchecked

Click the **Apply** button.

Create Credentials

Note: This exercise requires a User-Assigned Managed Identity

Navigate to Synapse Studio, click the **Manage** navigation icon, select **Credentials** from the **Security** grouping in the resulting navigation. Click the "**+ New**" button.

On resulting "**New credential**" pop-out, including:

Type	Select "**User Assigned Managed Identity**"
User Assigned Managed Identities	Select your User Assigned Managed Identity

Click the **Create** button. **Publish** or **Commit** change (as appropriate).

Create Integration Runtime

Navigate to Synapse Studio, click the **Manage** navigation icon, select "**Integration runtimes**" from the **Integration** grouping in the resulting navigation. Click the "**+ New**" button.

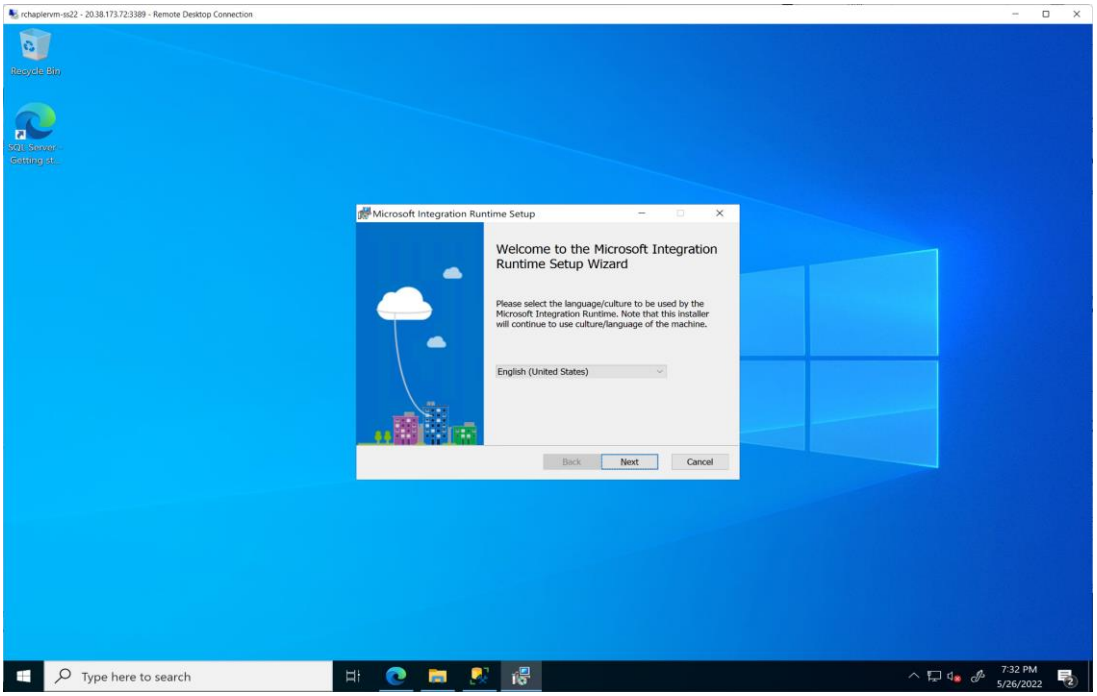
On the resulting pop-out, select "**Azure, Self-Hosted**" and then click the **Continue** button.

On the next pop-out, select "**Self-Hosted**" and then click the **Continue** button.

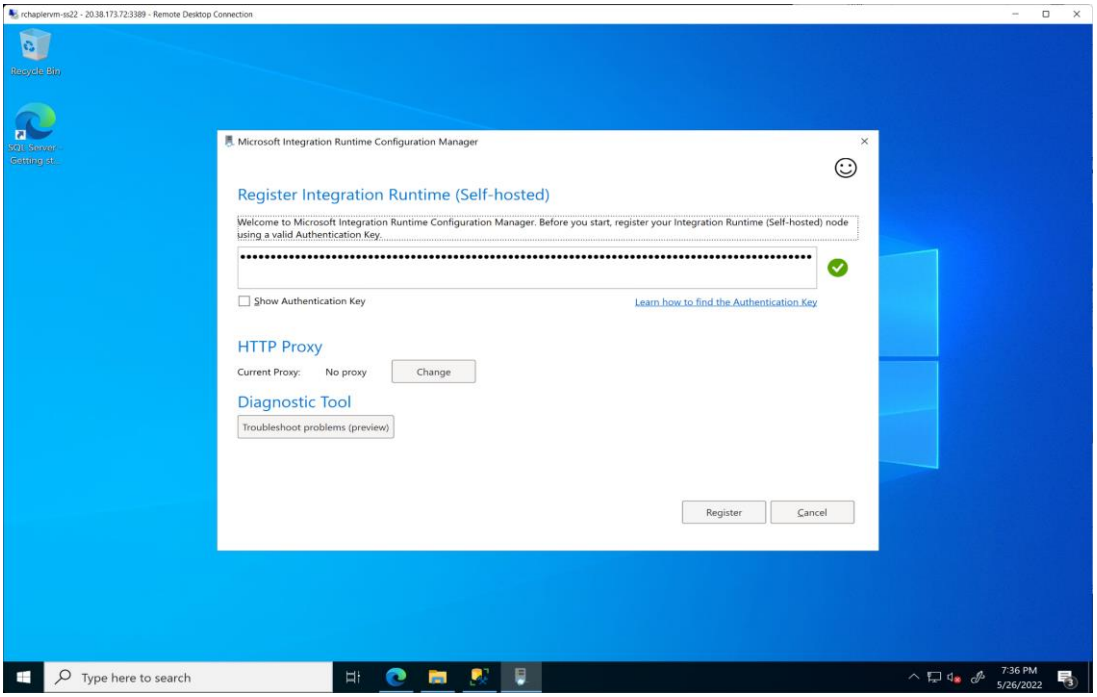
On the next pop-out, enter a value for **Name** and then click the **Create** button.

If you plan to install the Self-Hosted Integration Runtime on any machine other than the machine on which you're working, you will select Option 2. For this exercise, I am installing it on the same virtual machine that I prepared for SQL Server.

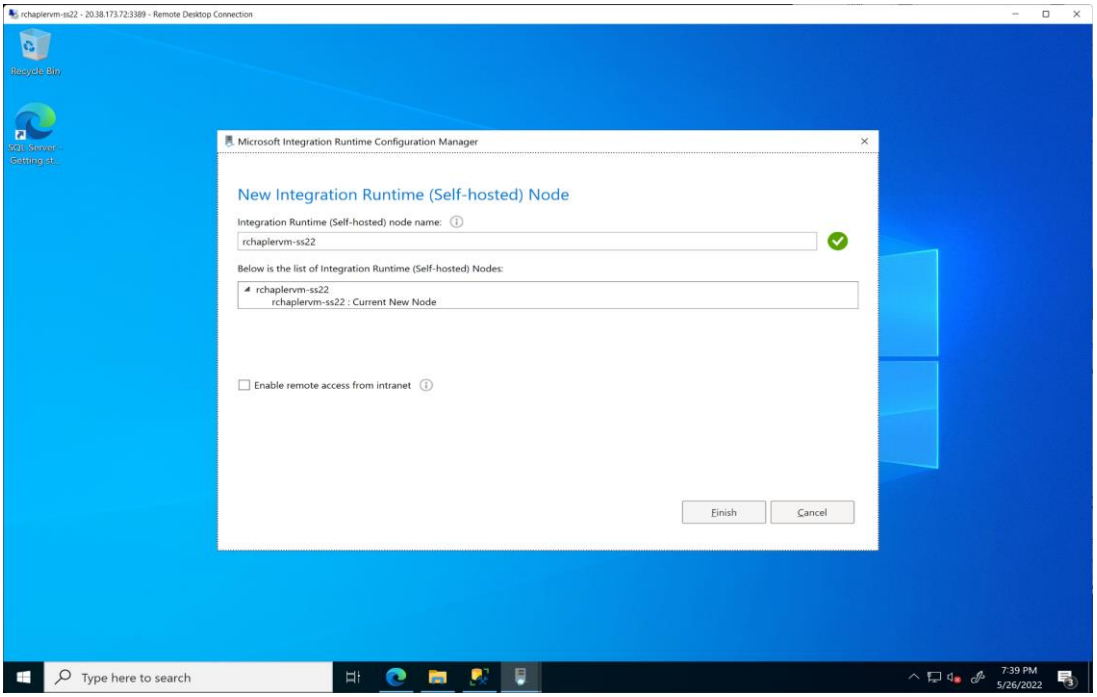
Click the "Download and install integration runtime" link and after download is complete, open the downloaded file on the Self-Hosted Integration Runtime machine.



Finish the generic installation process.



The "**Microsoft Integration Runtime Configuration Manager**" will open when installation is complete. Copy the **Key1** value from the Synapse pop-out {i.e., "**Step 2**"} and paste it into the "…**valid Authentication Key**" textbox. Click the **Register** button.



Review the final configuration values and then click the **Finish** button. After initialization is complete, you can **Close** the window.

Create Linked Services

Note: This exercise requires a User-Assigned Managed Identity, related Synapse Credentials, and a Self-Hosted Integration Runtime

Navigate to Synapse Studio, click the **Manage** navigation icon, select "**Linked services**" from the "**External connections**" grouping in the resulting navigation. Click the "**+ New**" button.

Search for and then select the resource for which you are creating a linked service; examples:

- Azure Data Explorer (Kusto)
- Azure Data Lake Storage Gen2
- Azure Key Vault
- Azure SQL Database
- SQL Server

Click the **Continue** button.

Complete the "**New linked service**…" pop-out, including:

Connect via…	Confirm default selection, "**AutoResolveIntegrationRuntime**"

And including resource specific items…

Azure Data Explorer (Kusto)

Connect via…	Confirm default selection, "**AutoResolveIntegrationRuntime**"
Authentication Method	Select "Managed Identity"
Selection Method	Select "Enter manually"
Endpoint	Enter the URI for your Data Explorer Pool {e.g., https://<UseCase>dep.<UseCase>s.kusto.azuresynapse.net}
Database	Select your Data Explorer Database

Azure Data Lake Storage Gen2

Connect via…	Confirm default selection, "**AutoResolveIntegrationRuntime**"
Selection Method	Confirm default selection, "**From Azure subscription**"
Authentication Method	Confirm default selection, "**Account key**"
Storage Account Name	Select your Data Lake

Azure Key Vault

Connect via…	Confirm default selection, "**AutoResolveIntegrationRuntime**"
Selection Method	Confirm default selection, "**From Azure subscription**"
Azure Key Vault Name	Select your Key Vault

Azure SQL Database

Connect via…	Confirm default selection, "**AutoResolveIntegrationRuntime**"
Selection Method	Confirm default selection, "**From Azure subscription**"
Server Name	Select your Azure SQL Server
Database Name	Select your Azure SQL Database
Authentication Type	Select "**System Assigned Managed Identity**" or "**User Assigned Managed Identity**" (depending on use case requirements)
Credentials	Select your Synapse Credentials

SQL Server

Connect via…	Select your Self-Hosted Integration Runtime
Server Name	Lorem
Database Name	Enter "AdventureWorks2019"
Authentication Type	Select "Windows authentication"
User Name **Password**	Enter your credentials

Click "**Test connection**" to confirm successful connection and then click the **Create** and **Publish** (or **Commit**) button.

Create Pools

Data Explorer Pool

Navigate to your Synapse Analytics workspace, click the **Open** link on the "**Open Synapse Studio**" rectangle and then click the **Manage** icon in the navigation pane.

Select "**Data Explorer pools**…" in the "**Analytics pools**" grouping of the resulting navigation and then click the "**+ New**" button.

On the resulting "**Create Data Explorer pool**" pop-out, **Basics** tab, including:

Workload	Select an appropriate option {e.g., "**Compute optimized**" as we will not have a large volume of data}
Size	Select an appropriate option {e.g., "**Extra Small**" as we will not demonstrate expensive computation}

Navigate to the "**Create Data Explorer pool**" pop-out, "**Additional settings**" tab, and then including:

Autoscale	Select an appropriate option {e.g., **Disabled** as we will not require Autoscale for this demonstration}
Number of Instances	Select an appropriate option (I usually choose the least expensive option for demonstrations)
Estimated Price	Confirm the estimated cost per hour that Azure generates based on your selected options

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Create Database

Navigate to your Synapse Analytics workspace, click the **Open** link on the "**Open Synapse Studio**" rectangle and then click the **Data** icon in the navigation, the **+** button in the header and then "**Data Explorer database**…" in the resulting drop-down menu.

On the resulting "**Create Data Explorer database**…" pop-out, including:

Pool Name	Select your Data Explorer Pool
Default Retention…	Confirm default, **365**
Default Cache…	Confirm default, **31**

Click the **Create** button.

Add Permissions

Navigate to your Data Explorer Pool and click **Permissions** in the "**Security + networking**" group in the navigation pane.

Click "**+ Add**" on the resulting page and select **AllDatabasesViewer** for read permissions or **AllDatabasesAdmin** for write, etc. permissions from the resulting drop-down menu.

Search for and select your Synapse managed instance.

Add Sample Data

Sample 1, Product Table

Navigate to your Synapse Analytics workspace, click the **Open** link on the "**Open Synapse Studio**" rectangle click the **Data** icon in the navigation, expand "**Data Explorer Databases**…", and then select your Data Explorer Pool.

Right click on your Data Explorer Database {e.g., **<UseCase>ded**} and select "New KQL script" from the resulting drop-down menu.

Execute the following logic to create a new table named Product:

```
.create table Product ( ProductId:int, ProductNumber: string, Name:string, ListPrice: decimal,
ModifiedDate:datetime )
```

Sample 2, StormEvents Data

Follow the instructions in the "Quickstart: Ingest sample data into Azure Data Explorer" article (https://docs.microsoft.com/en-us/azure/data-explorer/ingest-sample-data) to populate sample data that we can surface in Power Apps.

Familiarize yourself with the resulting data for use in later sections.

Dedicated SQL Pool

Navigate to Synapse.

Click the "+ New dedicated SQL pool" button.

Enter a meaningful name in "**Dedicated SQL pool name**" and choose a pricing tier.

Click the "**Review + create**" button, review configuration, and then click the **Create** button.

Serverless SQL Database

Navigate to your Synapse Analytics workspace and then click the **Open** link on the "**Open Synapse Studio**" rectangle.

Click the **Data** icon in the navigation pane.

Click the **Workspace** tab. Click the **+** button and "**SQL database**" in the resulting drop-down menu.

Complete the "**Create SQL database**" pop-out, including:

Select SQL Pool Type	Confirm default, **Serverless**

Click the **Create** button.

Apache Spark Pool

Navigate to your Synapse Analytics workspace and then click the **Open** link on the "**Open Synapse Studio**" rectangle.

Click the **Manage** icon and then "**Apache Spark pools**" in the navigation pane.

Click the "**New Apache Spark pool**" button and on the resulting pop-out, including:

Isolated Compute	Confirm default, **Disabled**
Node Size Family	Confirm default, "**Memory Optimized**"
Node Size	Select "Small (4 vCores / 32 GB)"
Autoscale	Select Disabled
Number of Nodes	Slide to lowest possible value (to minimize demonstration cost)
Estimated Price	Review final "Est. cost per hour" and view pricing details, as desired

Click the "**Review + create**" button, validate settings, and then click the **Create** button.

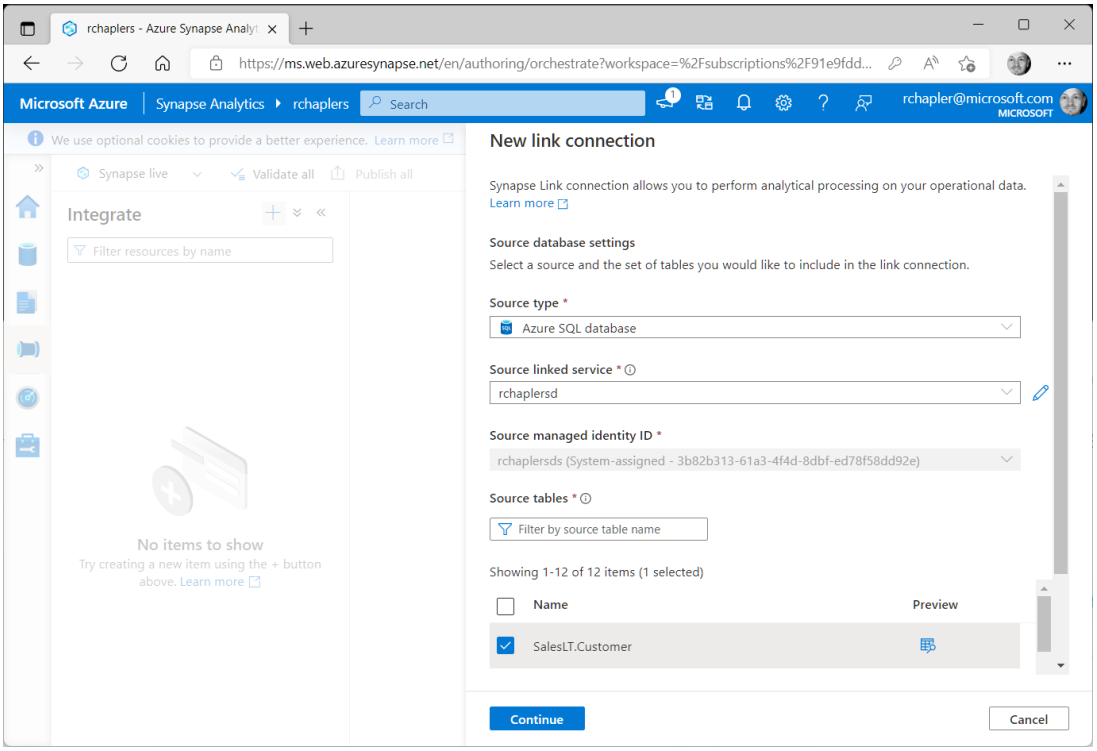
Create Link Connections

Note: These instructions require:

- *an Azure SQL Server / Database with 1) "**Pricing Tier**" set to **Premium**, 2) **Identity** >> "**System-Assigned Managed Identity**" set to **ON**, and 3) access granted {i.e., USER created / **db_datareader** ROLE altered}*
- *an Azure Synapse Linked Service for the Azure SQL Database (using a system-assigned managed identity)*
- *an Azure Synapse, Dedicated SQL Pool*

Source Database Settings

Navigate to Synapse Studio, click the **Integrate** navigation icon, click "**+**" and select "**Link Connection**…" from the resulting drop-down menu.

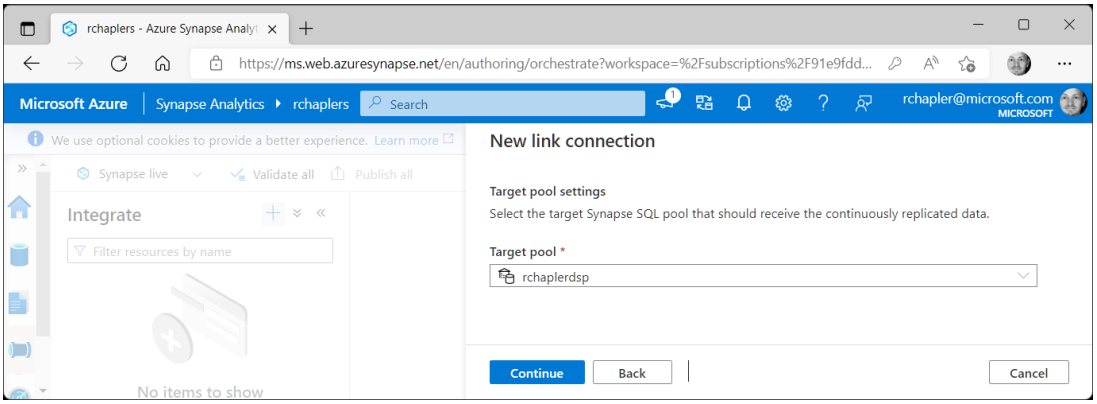


On the resulting "**New link connection**" > "**Source database settings**" pop-out, including:

Source Type	Select "**Azure SQL database**"
Source Linked Service	Select the Linked Service for your Azure SQL Database
Source Tables	Select tables for replication

Click the **Continue** button.

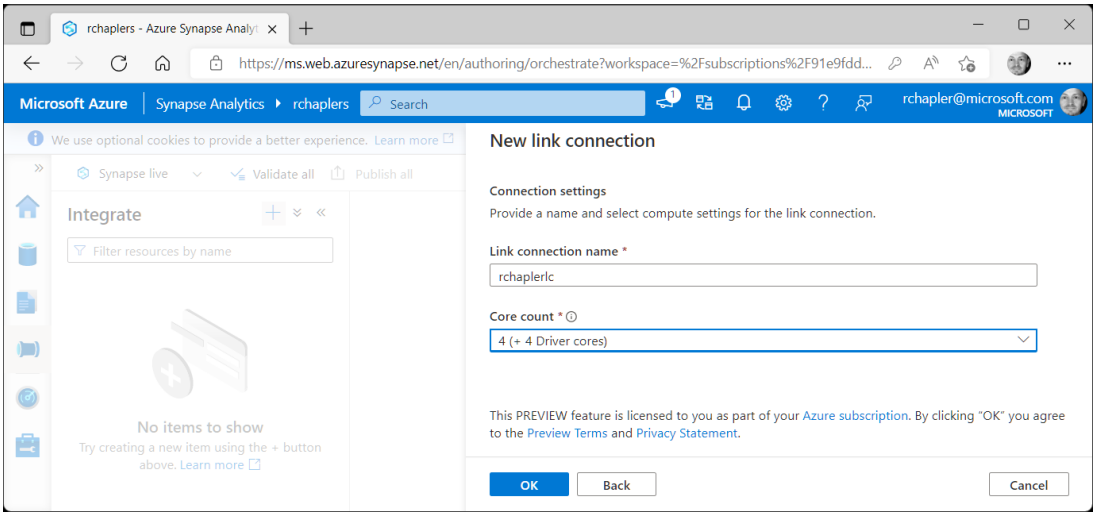
Target Pool Settings



On the resulting "**New link connection**" > "**Target pool settings**" pop-out, including:

Target Pool	Select your Dedicated SQL Pool

Click the **Continue** button.

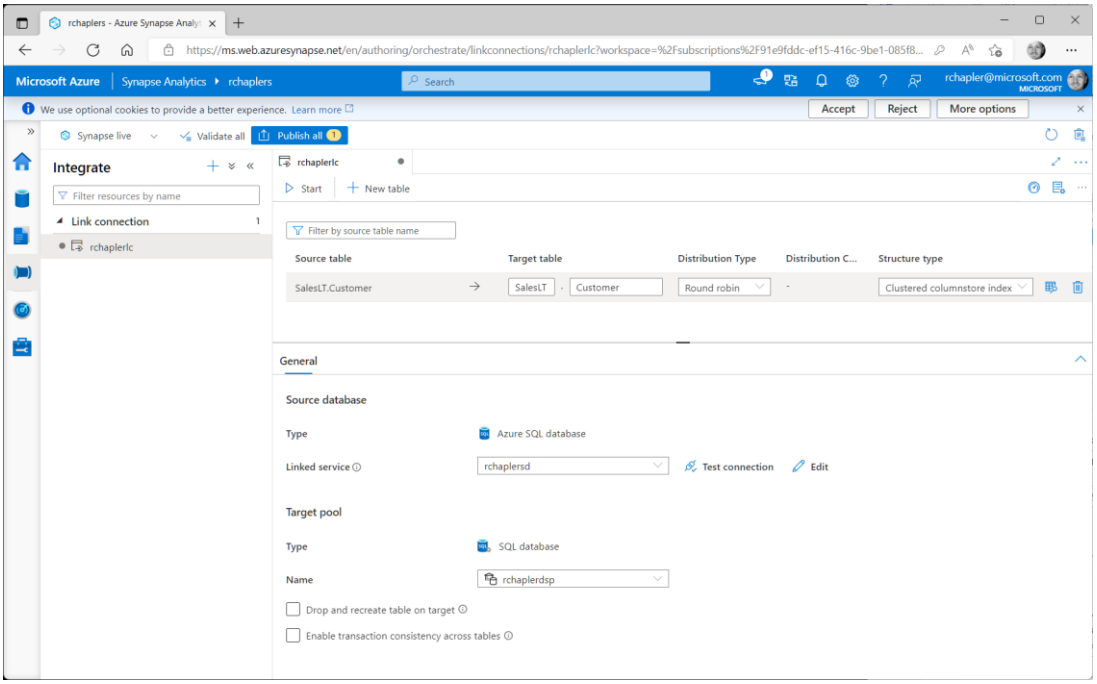


On the resulting "**New link connection**" > "**Connection settings**" pop-out, including:

Core Count	Select an appropriate value (consider the minimum to reduce cost)

Click the **OK** button.

All settings are presented in what feels like a description for recurring data movement (albeit not a Pipeline or Data Flow).



Click "**Publish All**" and then click the **Start** button.

Add Access Policy to Key Vault

Navigate to Key Vault, click "**Access Configuration**" in the **Settings** group of the navigation pane and then click the "**Go to access policies**" button on the resulting page. Click "**+ Create**" link and on the resulting "**Create an access policy**" form, **Permissions** tab, including:

Secret Permissions	Select **Get** and **List**

Click the **Next** button to navigate to the **Principal** tab, and including:

Select Principal	Search for, and then **Select** the managed identity for your Synapse Workspace

Navigate to the "**Review + Create**" tab, review configuration, and then click the **Create** button.

User-Assigned Managed Identity

Complete the "**Create User Assigned Managed Identity**" form.

Click the "**Review + create**" button, validate settings, and then click the **Create** button.

Virtual Machines

Create using Azure Portal

Use Case: Customer XYZ shared the following requirements:

- Mimic hard-to-replicate environments {e.g., on-prem SQL Server 2008 R2…}
- Create and configure using Portal

Complete the "**Create virtual machine**" form, **Basics** tab, including:

Availability Options	Confirm default, "No infrastructure redundancy required"
Security Type	Confirm default, **Standard**
Image	Search for and select the "**SQL Server 2008 R2**…" image
Azure Spot Instance	Confirm default, unchecked
Size	Select the SKU appropriate to your use case requirement (I typically start with the cheapest option)

Further down on the "**Create virtual machine**" form, **Basics** tab, including:

Username **Password**	Self-explanatory
…**Inbound Ports**	Confirm defaults, "Allow Selected Ports" and "RDP (3389)"

Complete the "**Create virtual machine**" form, "**SQL Server settings**" tab, including:

SQL Connectivity	Select "Public (Internet)"
Port	Confirm default, **1433**
SQL Authentication **Login Name** **Password**	Click to **Enable** and then confirm default (which will match your previously entered Administrator values)

Click the "**Review + create**" button, validate settings, and then click the **Create** button.

Configure SQL Server

Connect to the new virtual machine (using RDP) to complete image-specific configuration.

SQL Server Browser

Enable SQL Server Browser to allow connection to the SQL Server:

- Open **Services**, search for and double-click to open "**SQL Server Browser**"
- On the **General** tab, change "**Startup Type**" to **Automatic**, and then click the **Apply** button
- Click the **Start** button

Configure IE ESC

Disable Internet Explorer Enhanced Security Configuration to allow downloading to the demonstration VM:

- Open **Server Manager** and click the "**Configure IE ESC**" link in the "**Server Summary**" > "**Security Information**" interface grouping
    - Alternatively, navigate to "**Local Server**" and you will find "**IE Enhanced Security Configuration**" in the **Properties** grouping
- Complete the "**Internet Explorer Enhanced Security Configuration**" pop-up, click **Off** under Administrators and Users, and then click the **OK** button

Sample Database

Browse to https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure and complete the following steps:

- Download the appropriate version of the AdventureWorks sample database {e.g., "**AdventureWorks2008R2-oltp.bak**"}
- Open **SQL Server Management Studio**, right-click on **Databases**, and click "**Restore Database**" in the resulting drop-down menu
- On the "**Restore Database**" pop-up, enter **AdventureWorks** in the "**To database**" text box, "**Destination for restore**" interface grouping
- On the "**Restore Database**" pop-up, click the "**From device**" radio button in the "**Source for restore**" interface grouping
- Click the **ellipses** button and in the resulting "**Specify Backup**" pop-up, click the **Add** button
- Browse to the downloaded BAK file; you may have to copy the BAK from the Downloads folder to another location (like c:\temp) to make it accessible
- Back on the "**Restore Database**" pop-up, check the box next to added item in "**Select the backup sets to restore**" and then click the **OK** button

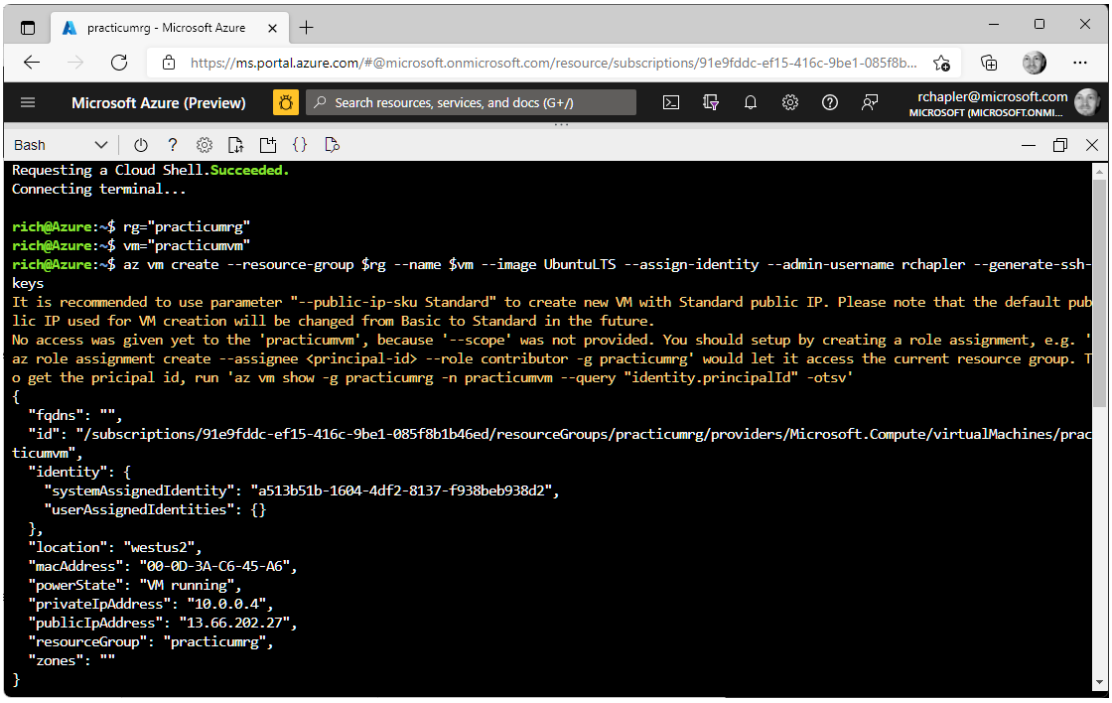
When you have successfully completed these steps, you will see a pop-up that says "**The restore of database 'AdventureWorks**" completed successfully.

Create using Cloud Shell (Bash)

Use Case: Customer XYZ shared the following requirements:

- Mimic hard-to-replicate environments {e.g., on-prem Ubuntu (Linux)}
- Create and configure using Cloud Shell

Navigate to Cloud Shell and select **Bash** from the "**Select Environment**" drop-down menu.



Update and execute the following command to set variable **rg** {i.e., the name of your Resource Group}:

```
rg="<UseCase>rg"
```

Update and execute the following command to set variable **vm** {i.e., the desired name for the new virtual machine}:

```
vm="<UseCase>vm"
```

Execute the following command to create your virtual machine:

```
az vm create --resource-group $rg --name $vm --image UbuntuLTS --assign-identity --admin-username
rchapler --generate-ssh-keys
```

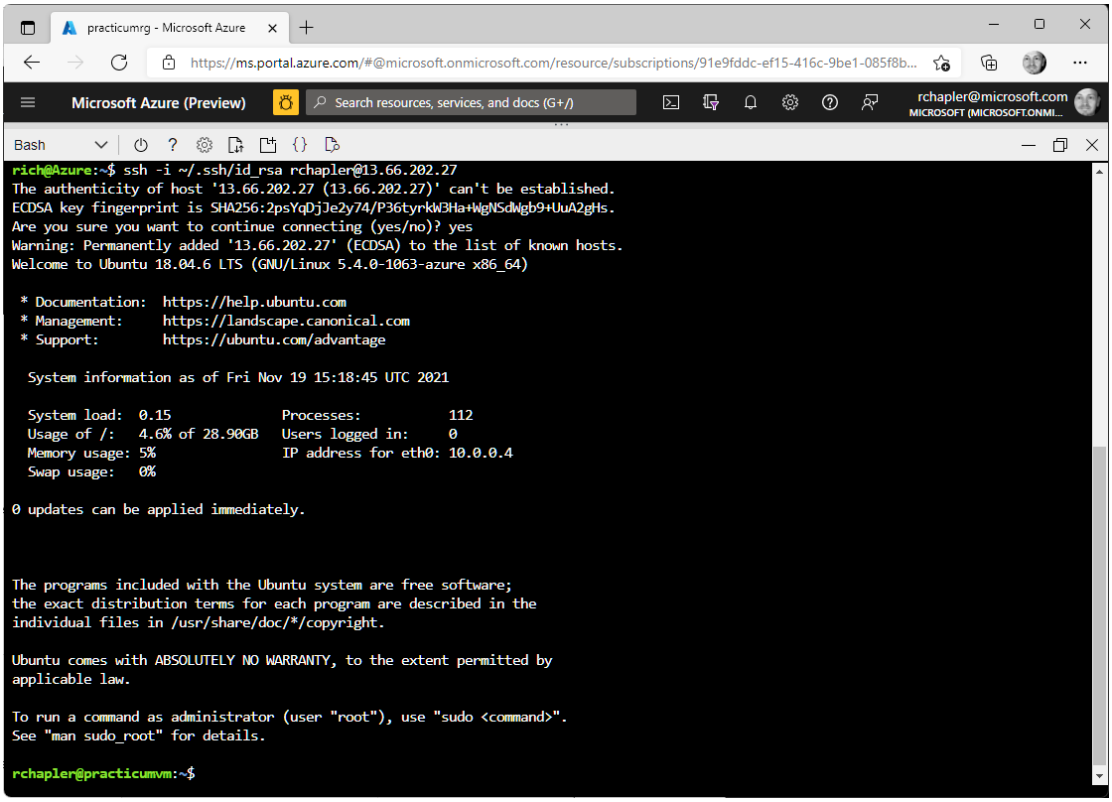
Successful execution will create the following resources:

- Virtual Machine
- Disk

- Network Security Group
- Public IP Address
- Network Interface
- Virtual Network

Update and execute the following command to connect to the server using SSH:

```
ssh -i ~/.ssh/id_rsa rchapler@{public ip address}
```



You will know you are successful if your prompt has changed to your equivalent of **rchapler@<UseCase>vm**.

Acquisition

In this section, we will discuss various methods of acquiring source data {e.g., migration, direct connection, incremental load, etc.}

Migration

Follow these instructions to migrate data from an on-prem instance of SQL Server to Azure SQL.

Use Case

Example notes:

- "We want to migrate some of many tables (schema and data) from an on-prem, SQL Server 2008 R2 database to Azure"
- "**Once migrated, data will live only in Azure**… **the on-prem source will be deprecated**"

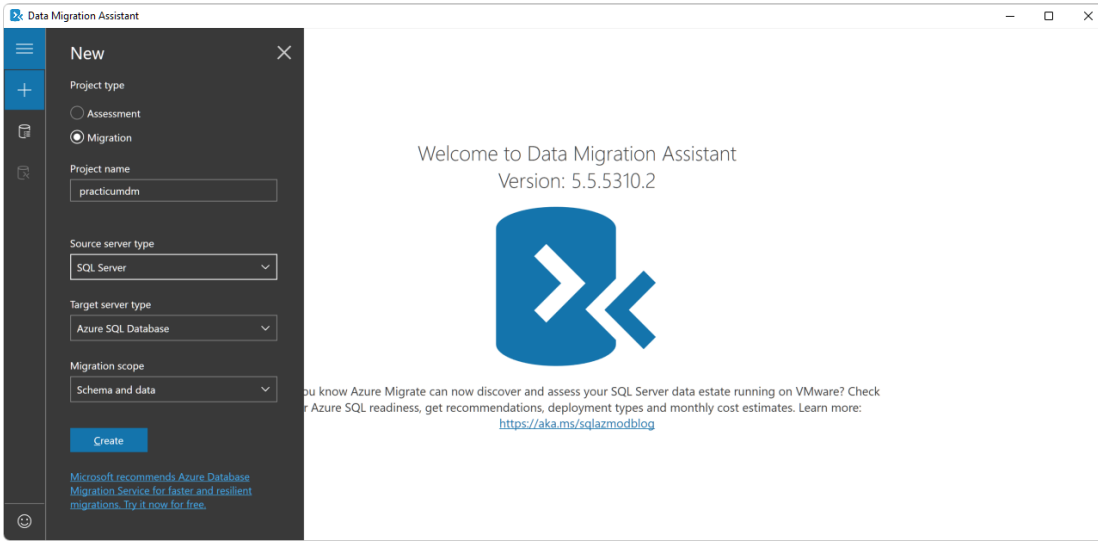
Prerequisites

This solution requires the following resources:

- Data Migration Assistant
- SQL
- Virtual Machine (with "**SQL Server 2008 R2**…" image and **AdventureWorks** sample database)

Create Migration Project

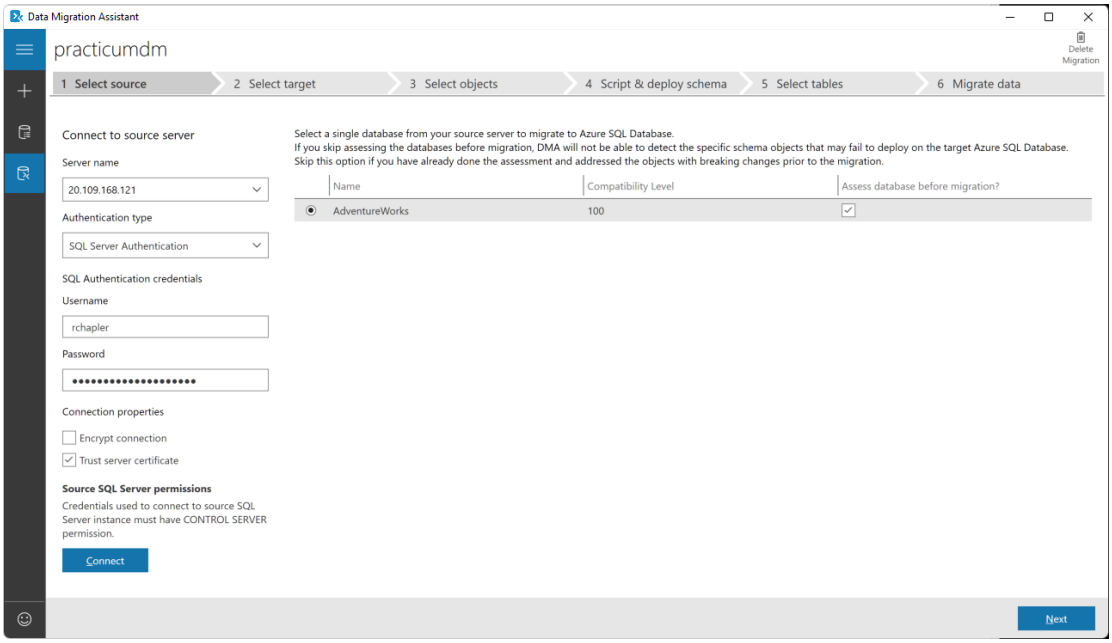
Open **Data Migration Assistant** and click **+** to create a new project



Complete the "**Create Data Share**" pop-out, including:

Project Type	Select the **Migration** radio button
Source Server Type	Select “SQL Server”
Target Server Type	Select “Azure SQL Database”
Migration Scope	Select “Schema and data”

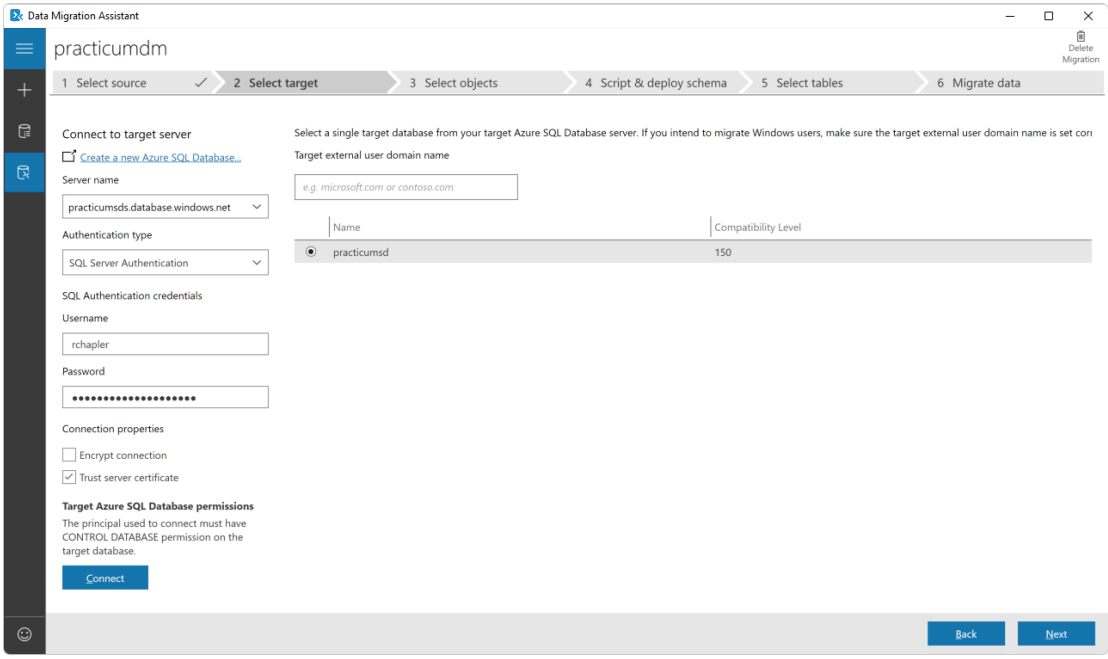
Click the **Create** button.



Complete the “**1. Select source**” form, including:

Server Name	Enter the “**Public IP address**” for your SQL Server 2008 virtual machine
Authentication Type **Username** **Password**	Select “**Windows Authentication**” and enter credentials
Encrypt Connection	Unchecked
Trust Server Certificate	Checked

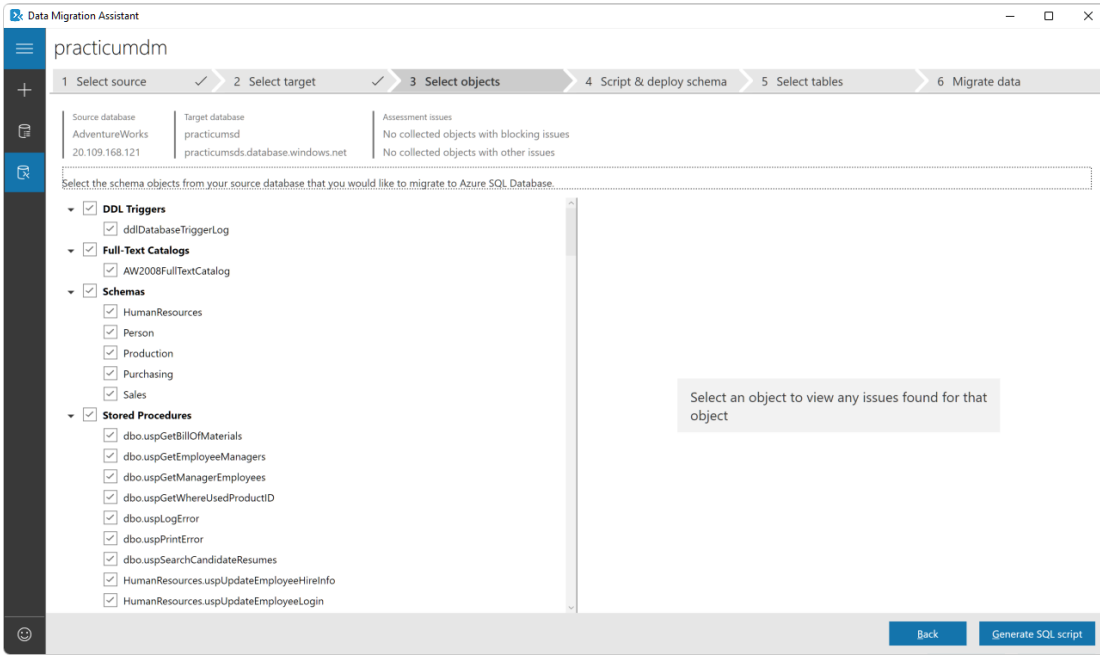
Click the **Connect** button, confirm selection of **AdventureWorks** and then click the **Next** button.



Complete the "**2. Select target**" form, including:

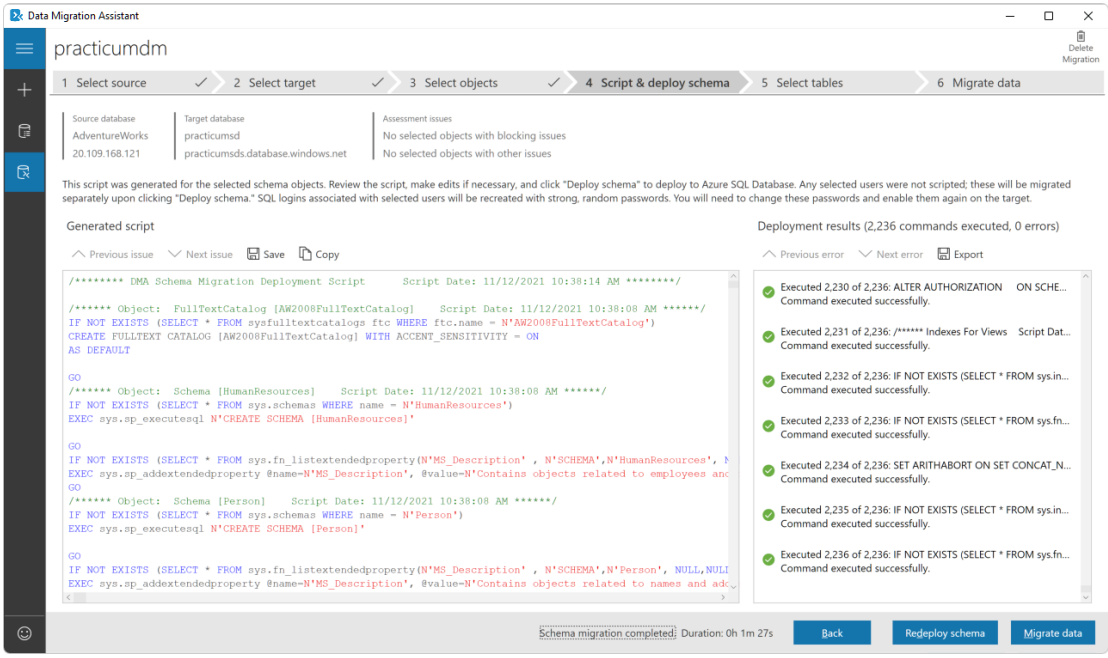
Authentication Type **Username** **Password**	Select "**SQL Server Authentication**" and enter credentials
Encrypt Connection	Unchecked
Trust Server Certificate	Checked

Click the **Connect** button, confirm selection of your SQL database, and then click the **Next** button.



Complete the "**3. Select objects**" form, review schema objects, taking special note of any flagged items.

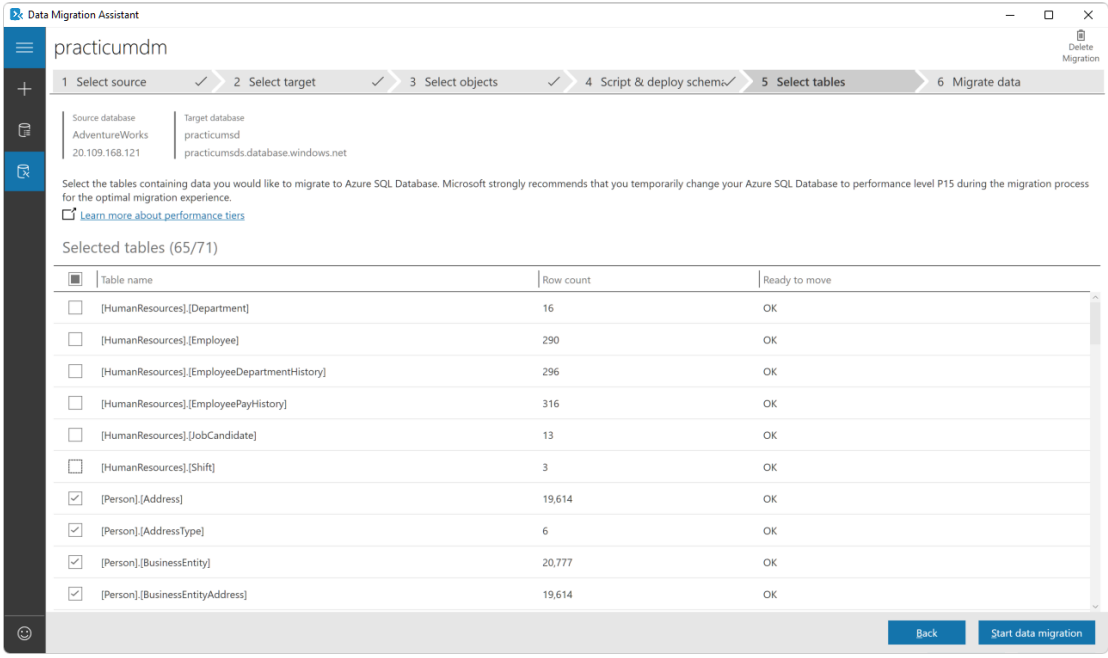
Click the "**Generate SQL script**" button in the bottom-right when you are ready to continue.



Complete the "**4. Script & deploy schema**" form, review the generated script.

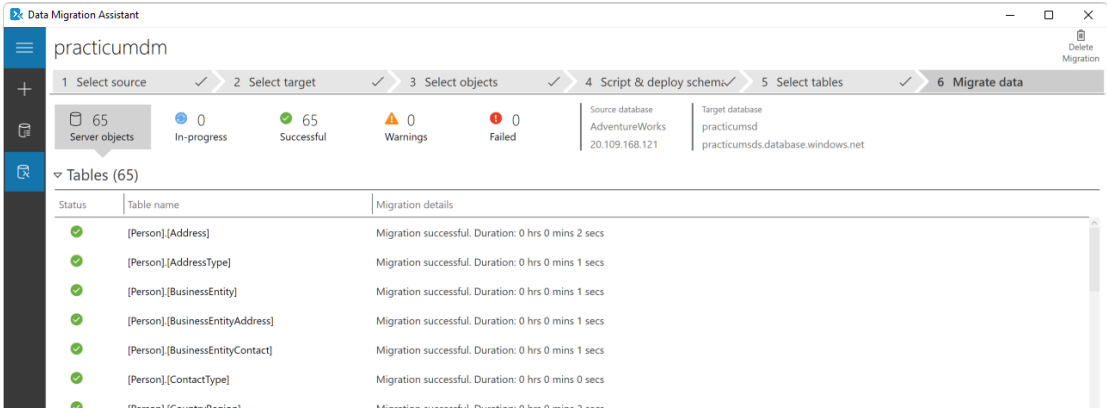
Click the “**Deploy schema**” button when you are ready to execute the script and create schema on the target database.

Click the “**Migrate data**” button when you are ready to continue.



Complete the “**5. Select tables**” form, review the list of tables and exclude items as appropriate.

Click the “**Start data migration**” button when you are ready to continue.



Allow time for the migration to successfully complete (as above).

No-Load

User connects to data in place, with no ETL process

External Data

Follow these instructions to explore methods of using data from data products external to Synapse.

Use Case

Example notes:

- “We want to use data via direct connection to the source”
- “We want to minimize investment in recurring data movement wherever possible”

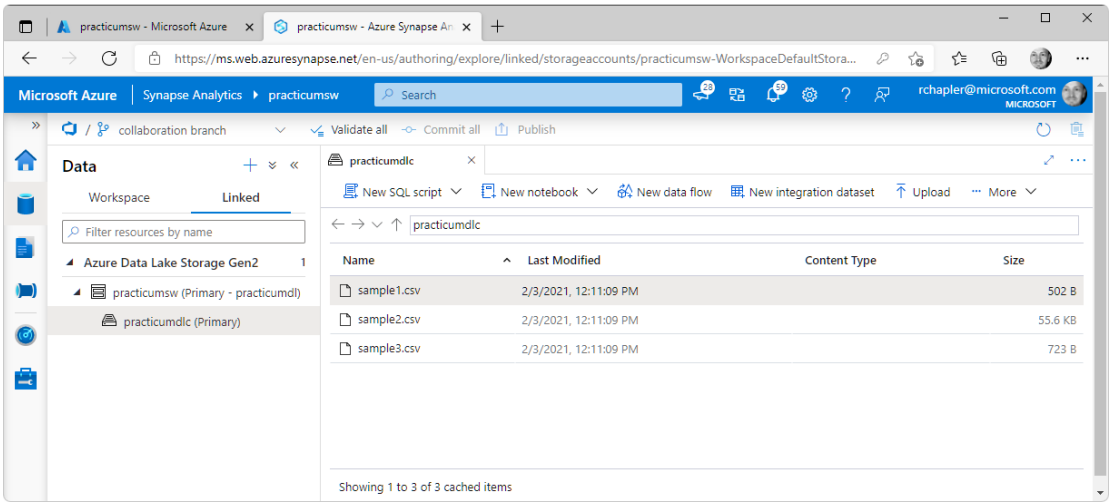
Prerequisites

This solution requires the following resources:

- Data Lake (with sample data)
- Synapse (with Serverless SQL Database and Apache Spark Pool)

Alternate Methods

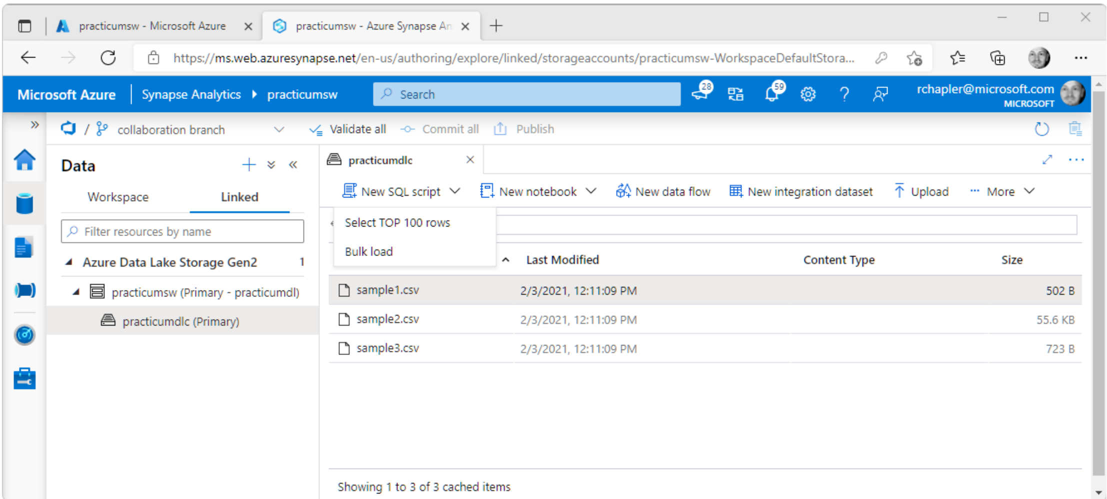
Navigate to your Synapse Analytics workspace, click the **Open** link on the “**Open Synapse Studio**” rectangle and then click the **Data** icon in the navigation pane.



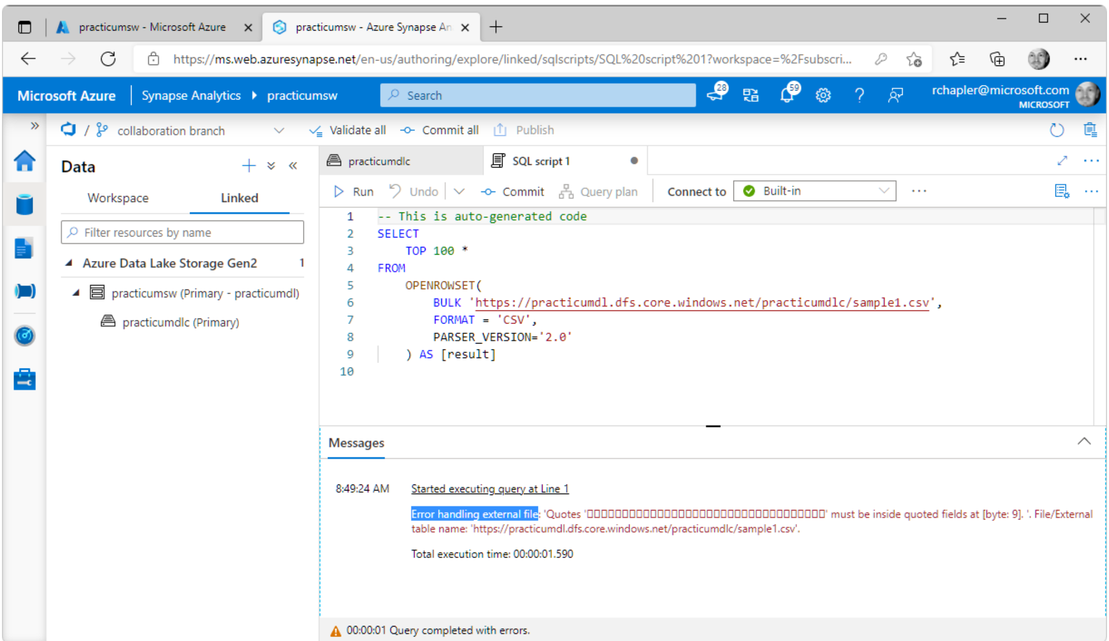
Click the **Linked** tab and Expand navigation pane.

You will see that the Azure Data Lake Storage Gen2 specified during Synapse instantiation {i.e., <UseCase>dlc} surfaces with no additional configuration.

Method 1: SQL Script (OPENROWSET)



Click the "**New SQL script**" button and "**SELECT TOP 100 rows**" from the resulting drop-down menu.



The resulting auto-generated code uses OPENROWSET(… and a built-in, serverless SQL Pool to pull data directly from the CSV file in the ADLS container.

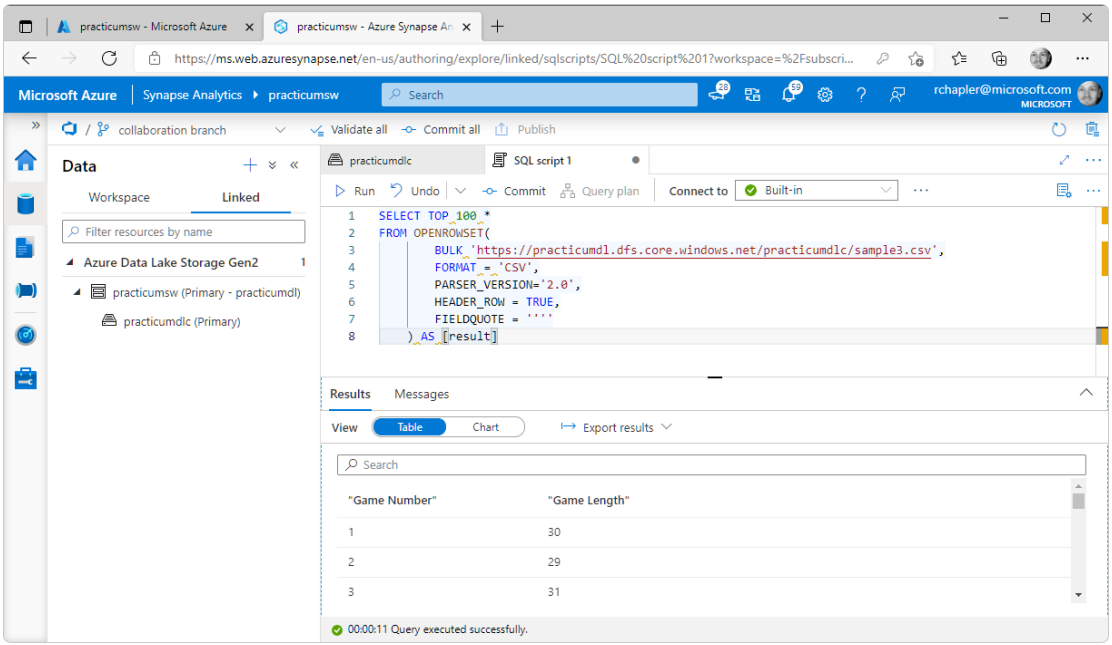
When we run the code (assuming you are using the same sample files, of course), we get an error about quote-handling.

Given the formatting of the sample file, we cannot run this without changes to the code.

Update the code to:

```
SELECT TOP 100 *
FROM OPENROWSET(
    BULK 'https://<UseCase>dl.dfs.core.windows.net/<UseCase>dlc/sample3.csv',
    FORMAT = 'CSV',
    PARSER_VERSION='2.0',
    HEADER_ROW = TRUE,
    FIELDQUOTE = ''''
  ) AS [result]
```

… and then, re-Run…

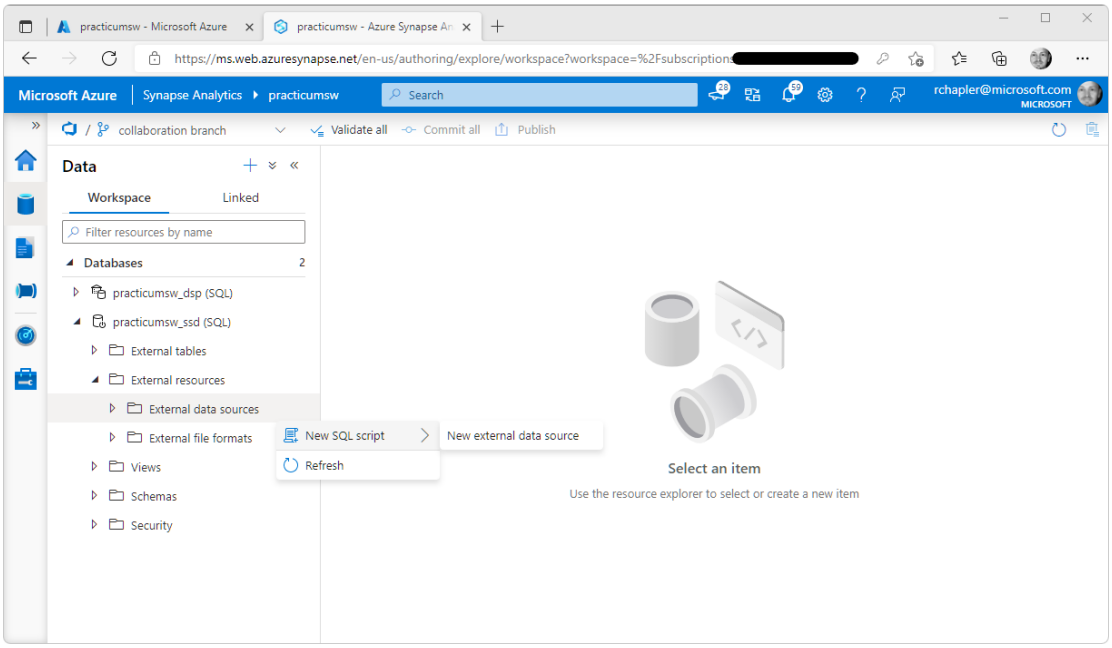


Method 2: External Table (Serverless SQL Database)

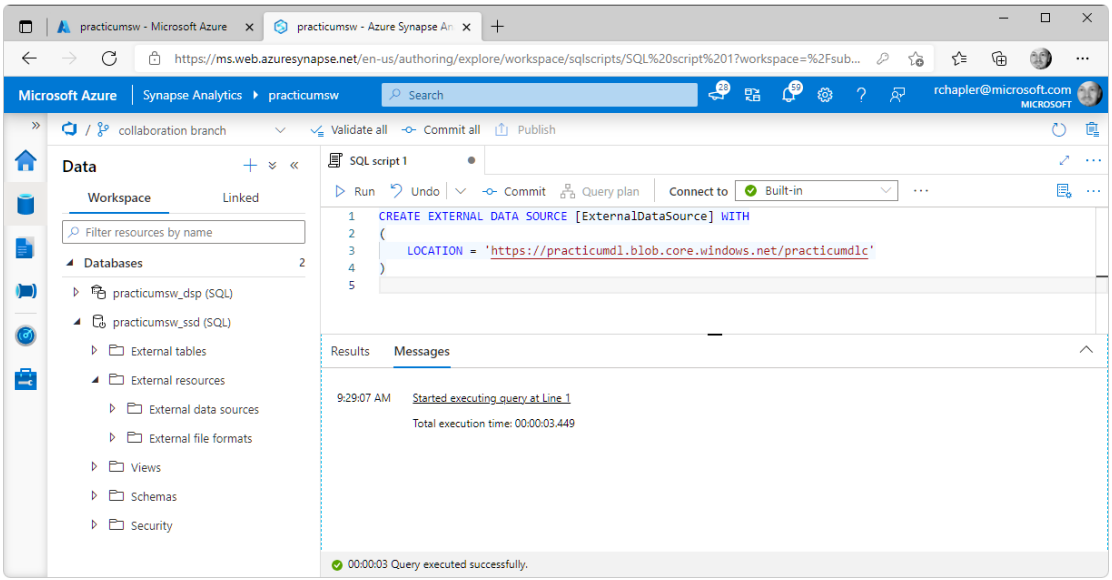
Click the **Workspace** tab.

External Data Source

Expand navigation to and then right-click on "**External data sources**."



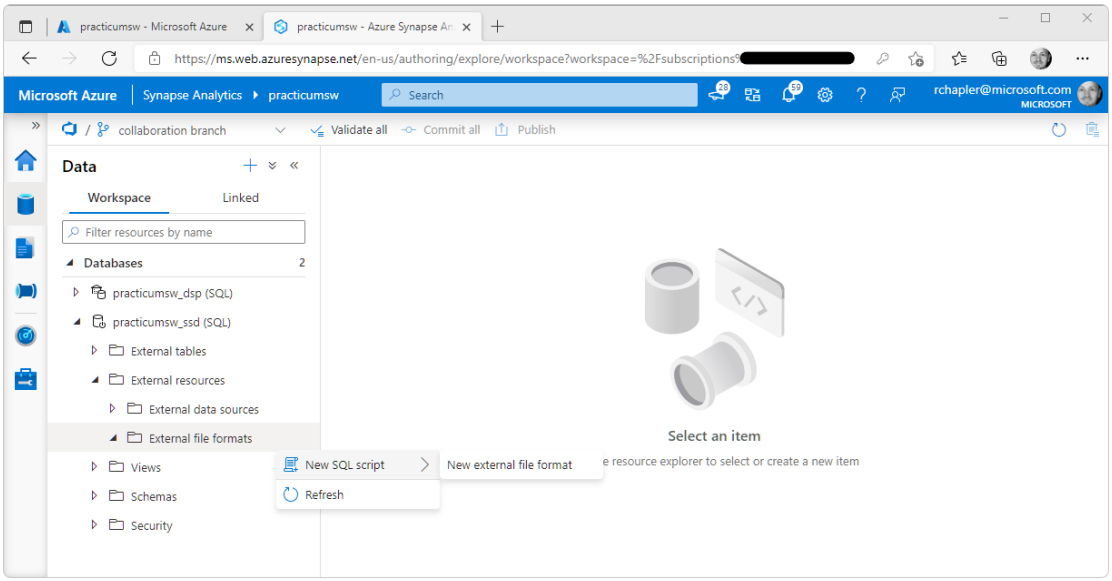
Click “**New SQL script**”, and then “**New external data source**” in the drop-down menu.



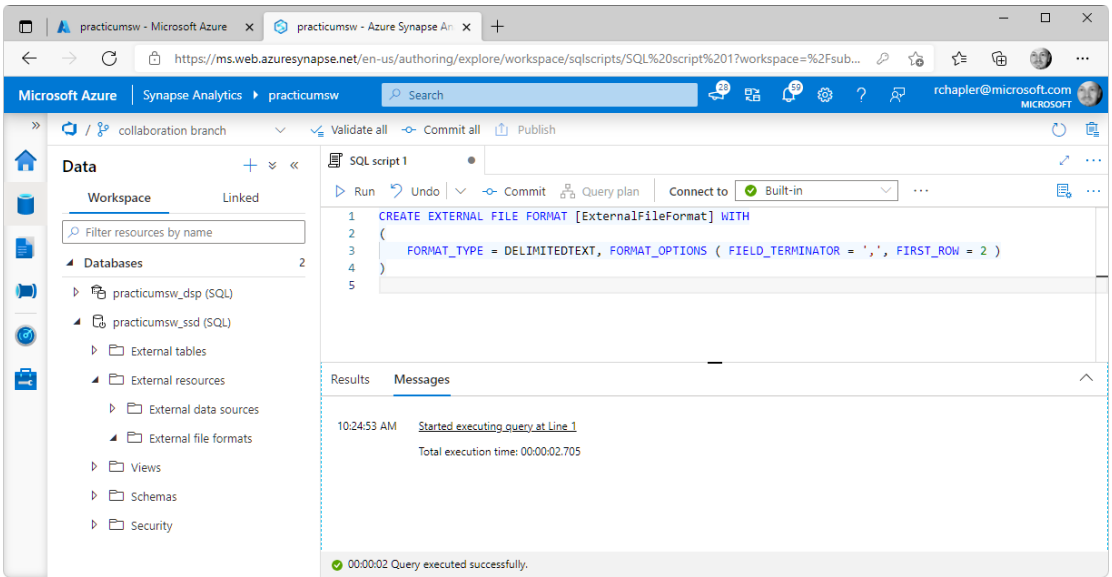
Replace <**STORAGEACCOUNT**> and <**CONTAINER**> in the auto-generated code and then **Run**.

External File Format

Expand navigation to and then right-click on “**External file formats**.”



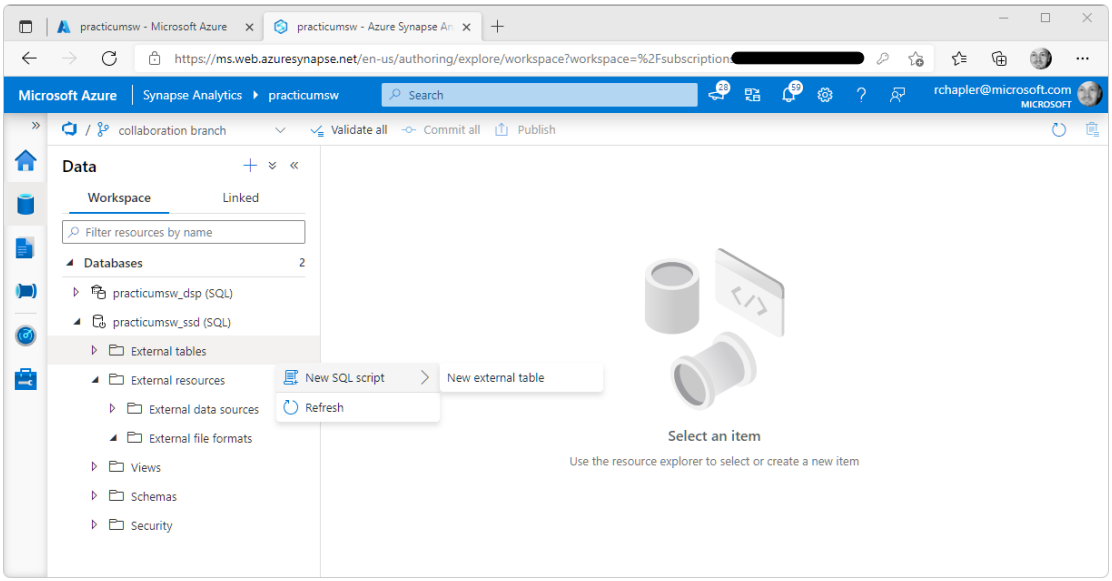
Click “**New SQL script**”, and then “**New external file format**” in the drop-down menu.



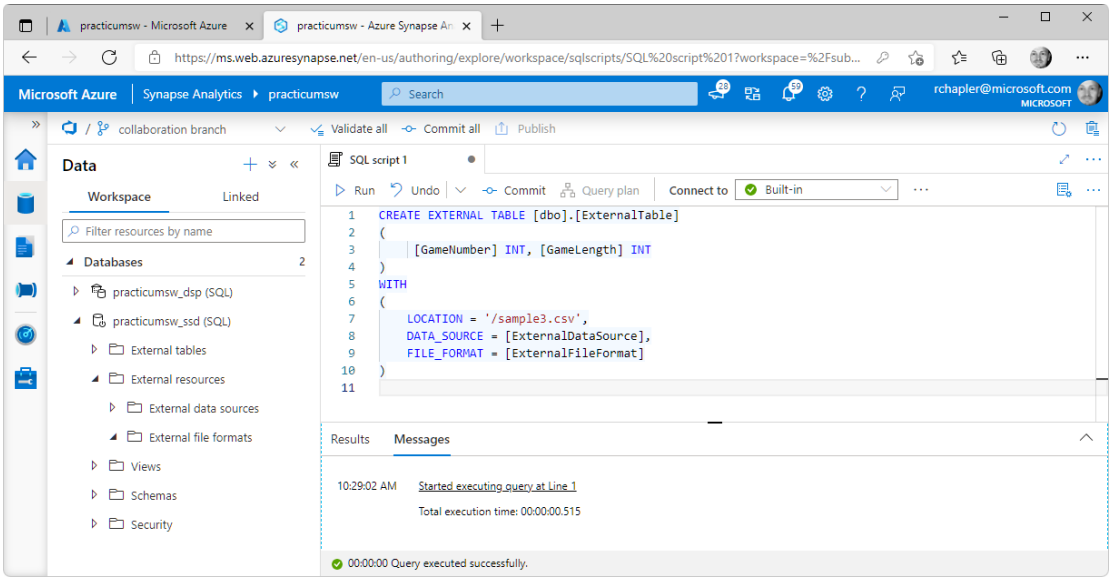
Append FORMAT_OPTIONS arguments FIELD_TERMINATOR and FIRST_ROW to the default FORMAT_TYPE argument and then **Run**.

External Table

Expand navigation to and then right-click on “**External tables**.”



Click “**New SQL script**”, and then “**New external table**” in the drop-down menu.

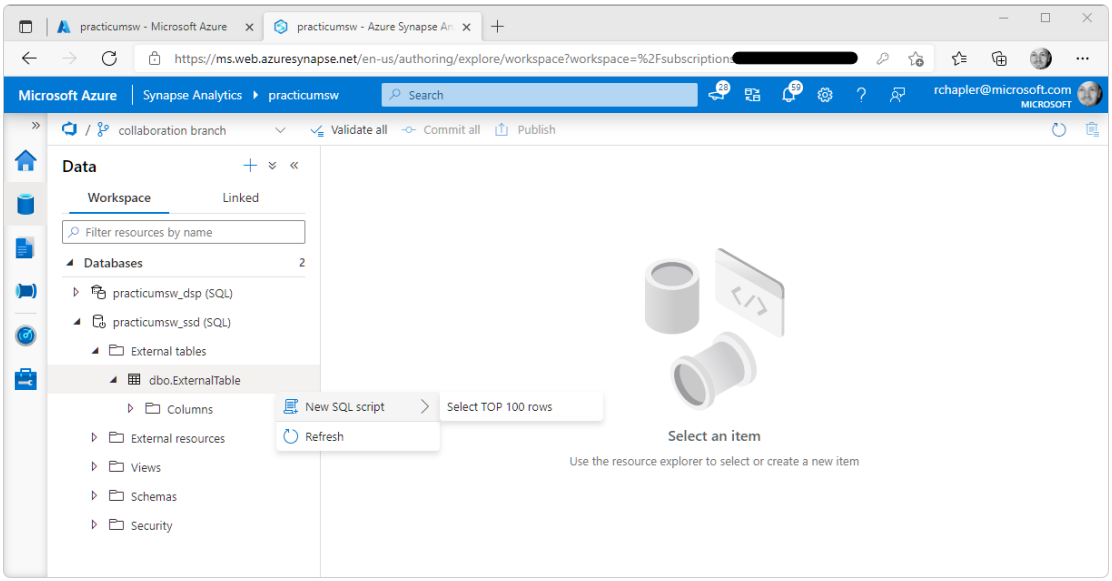


Update the following items in the default code and then **Run**.

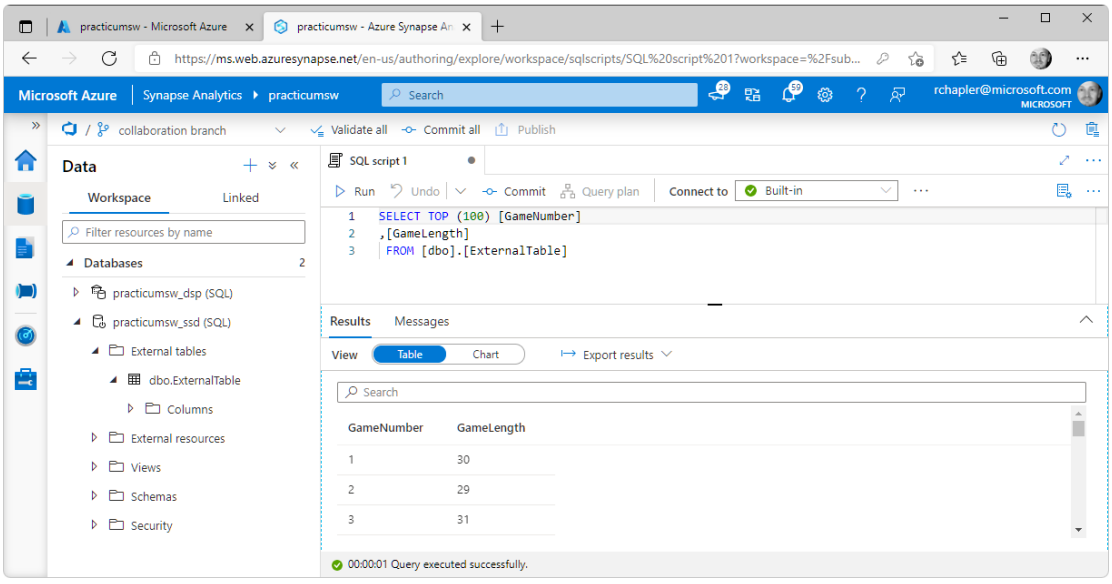
- **Schema Definition** … replace [Id] INT with columns matching the external data source
- **LOCATION** … replace ‘/folder/file’ with values matching those in your container
- **DATA_SOURCE** … replace [DataSource1] with the name used in Create External Data Source
- **FILE_FORMAT** … replace [FileFormat1] with the name used in Create External File Format

Confirm Success

Right-click on “**External tables**” and click Refresh.



Right-click on “**dbo.ExternalTable**”, click on “**New SQL script**”, and finally click on “**Select TOP 100 rows**” in the drop-down menu.



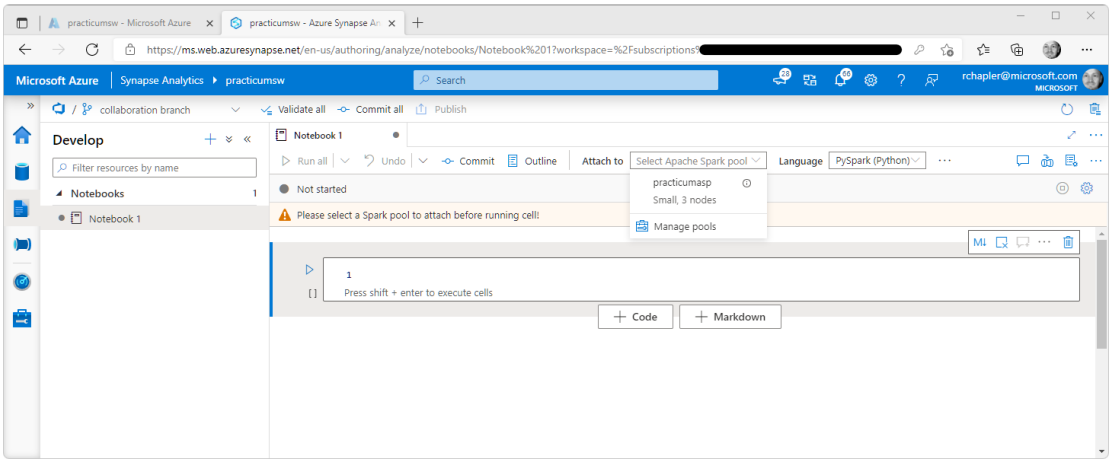
Click **Run**.

Method 3: Spark Notebook (Python)

Click the **Develop** tab.

Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

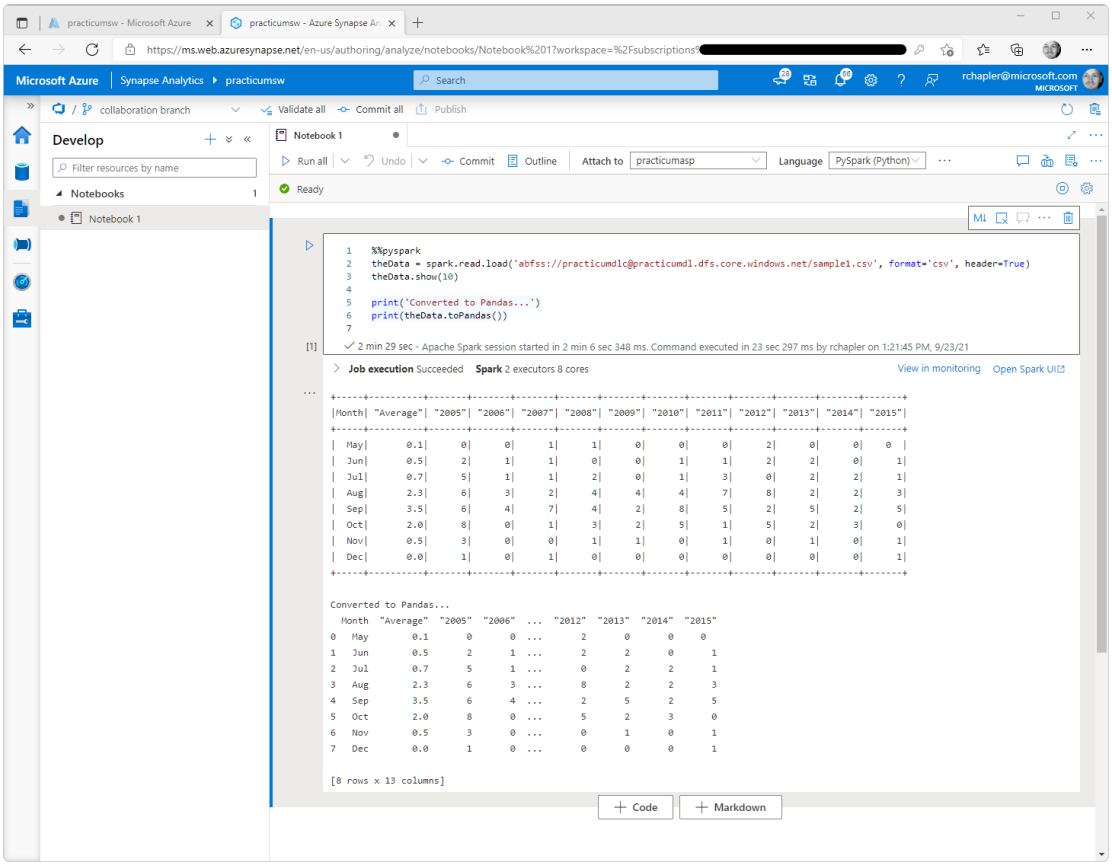
Select **Notebook** from the resulting drop-down menu.



Select your Apache Spark Pool from the "**Attach to**" drop-down menu.

Paste the following code to Cell 1 and then click "**Run Cell**":

```
%%pyspark
theData = spark.read.load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/sample1.csv',
format='csv', header=True)
theData.show(10)

print('Converted to Pandas...')
print(theData.toPandas())
```


practicumsw - Microsoft Azure
practicumsw - Azure Synapse An
https://ms.web.azuresynapse.net/en-us/authoring/analyze/notebooks/Notebook%201?workspace=%2Fsubscriptions/[redacted]
Microsoft Azure
Synapse Analytics
practicumsw
Search
rchapler@microsoft.com
MICROSOFT
collaboration branch
Validate all
Commit all
Publish
Develop
Filter resources by name
Notebooks 1
Notebook 1
Notebook 1
Run all
Undo
Commit
Outline
Attach to practicumasp
Language PySpark (Python)
Ready
1 %%pyspark
2 theData = spark.read.load('abfss://practicumdlc@practicumdl.dfs.core.windows.net/sample1.csv', format='csv', header=True)
3 theData.show(10)
4
5 print('Converted to Pandas...')
6 print(theData.toPandas())
7
[1] ✓ 2 min 29 sec - Apache Spark session started in 2 min 6 sec 348 ms. Command executed in 23 sec 297 ms by rchapler on 1:21:45 PM, 9/23/21
Job execution Succeeded
Spark 2 executors 8 cores
View in monitoring
Open Spark UI
+-----+---------+------+------+------+------+------+------+------+------+------+------+
|Month| "Average"| "2005"| "2006"| "2007"| "2008"| "2009"| "2010"| "2011"| "2012"| "2013"| "2014"| "2015"|
+-----+---------+------+------+------+------+------+------+------+------+------+------+------+
May	0.1	0	0	1	1	0	0	0	2	0	0	0
Jun	0.5	2	1	1	0	0	1	1	2	2	0	1
Jul	0.7	5	1	1	2	0	1	3	0	2	2	1
Aug	2.3	6	3	2	4	4	4	7	8	2	2	3
Sep	3.5	6	4	7	4	2	8	5	2	5	2	5
Oct	2.0	8	0	1	3	2	5	1	5	2	3	0
Nov	0.5	3	0	0	1	1	0	1	0	1	0	1
Dec	0.0	1	0	1	0	0	0	0	0	0	0	1
+-----+---------+------+------+------+------+------+------+------+------+------+------+------+
Converted to Pandas...
Month "Average" "2005" "2006" ... "2012" "2013" "2014" "2015"
0 May 0.1 0 0 ... 2 0 0 0
1 Jun 0.5 2 1 ... 2 2 0 1
2 Jul 0.7 5 1 ... 0 2 2 1
3 Aug 2.3 6 3 ... 8 2 2 3
4 Sep 3.5 6 4 ... 2 5 2 5
5 Oct 2.0 8 0 ... 5 2 3 0
6 Nov 0.5 3 0 ... 0 1 0 1
7 Dec 0.0 1 0 ... 0 0 0 1
[8 rows x 13 columns]
Code
Markdown

Mount Data Lake

Follow these instructions to establish connection to data in Azure Data Lake Storage.

Prerequisites

This solution requires the following resources:

- Data Lake (with “Storage Blob Data Reader” permissions set for your Application Registration)
- Databricks (with cluster, notebook, and secret scope)

Prepare Logic

Navigate to Databricks and the **<UseCase>dbn** notebook.

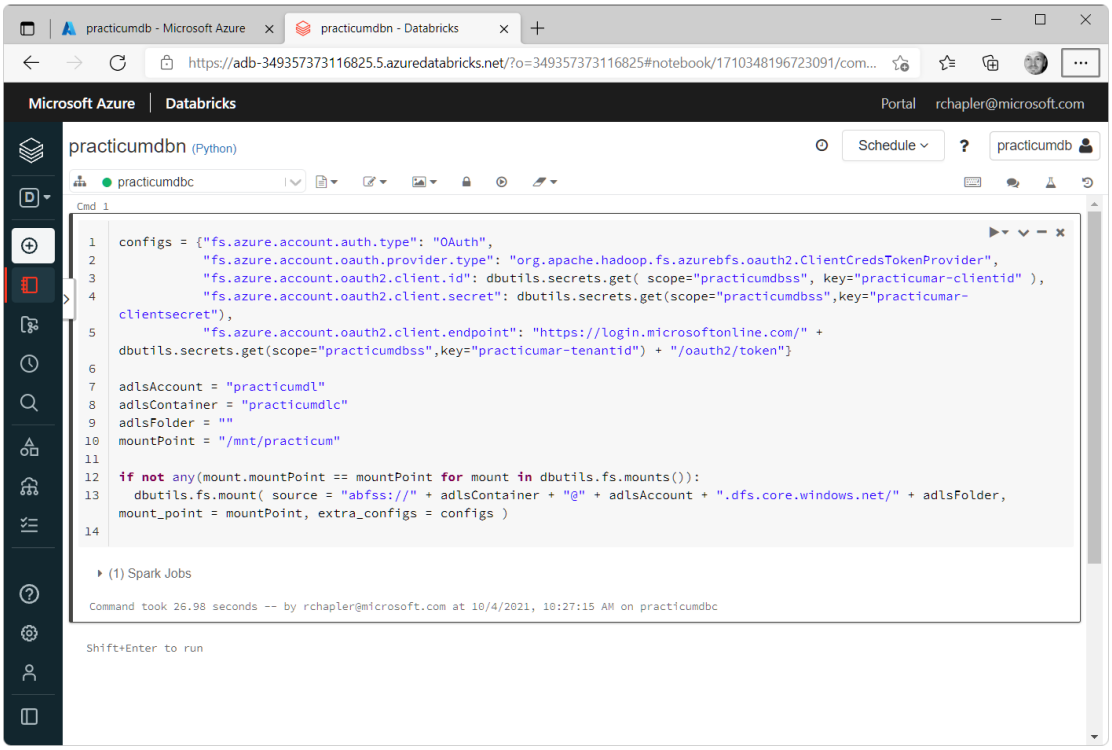
Add a new cell and paste the following code:

```
configs = {"fs.azure.account.auth.type": "OAuth",
       "fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
       "fs.azure.account.oauth2.client.id": dbutils.secrets.get( scope="<UseCase>dbss", key="
<UseCase>ar-clientid" ),
       "fs.azure.account.oauth2.client.secret": dbutils.secrets.get( scope="<UseCase>dbss",
key="<UseCase>ar-clientsecret" ),
       "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/" +
dbutils.secrets.get( scope="<UseCase>dbss", key=" <UseCase>ar-tenantid" ) + "/oauth2/token"}

adlsAccount = "<UseCase>dl"
adlsContainer = "<UseCase>dlc"
adlsFolder = ""
mountPoint = "/mnt/<UseCase>"

if not any(mount.mountPoint == mountPoint for mount in dbutils.fs.mounts()):
 dbutils.fs.mount( source = "abfss://" + adlsContainer + "@" + adlsAccount + ".dfs.core.windows.net/" +
adlsFolder, mount_point = mountPoint, extra_configs = configs )
```

(Partial) logic explanation:

- **<UseCase>dbss** … refers to the Secret Scope
- **myClientId** … refers to the Key Vault secret containing the “Application (client) ID”
- **myClientSecret** … refers to the Key Vault secret containing the “Client Secret”
- **myTenantId** … refers to the Key Vault secret containing the “Directory (tenant) ID”
- **<UseCase>dl** … refers to your Data Lake
- **<UseCase>dlc** … refers to the Data Lake Container
- **adlsFolder** … placeholder / syntax for inclusion of a folder (null because it is not applicable in this instance)

Run “**Cmd 1**.”



Quick Idea…

Consider organizing mount locations by associating them with a Databricks database entity; code example:

```
%sql
CREATE DATABASE myDatabase LOCATION "/mnt/<UseCase>/myDatabase"
```

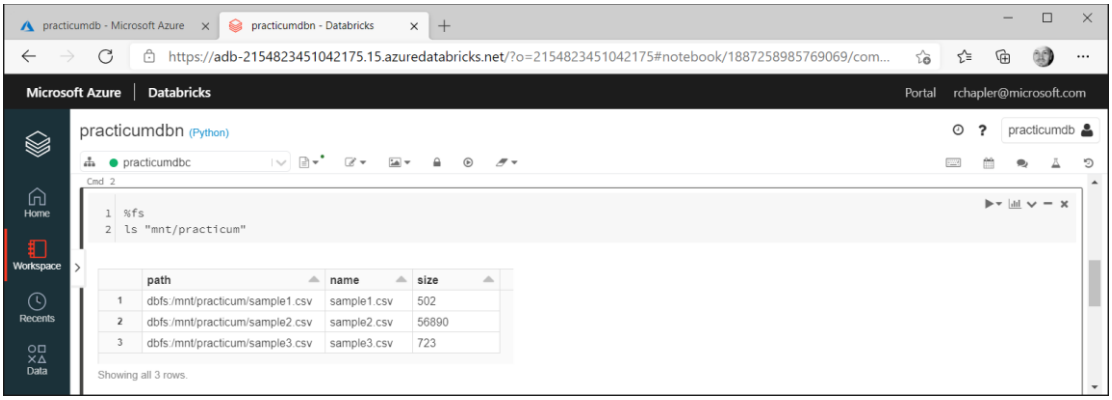
Confirm Success

Proof 1 … Sample Files at Mount Point

Note: Running this logic requires "Storage Blob Data Reader" permissions on your data lake.

Add a cell below "**Cmd 1**." In cell "**Cmd 2**", paste and run the following code:

```
%fs
ls "mnt/<UseCase>"
```

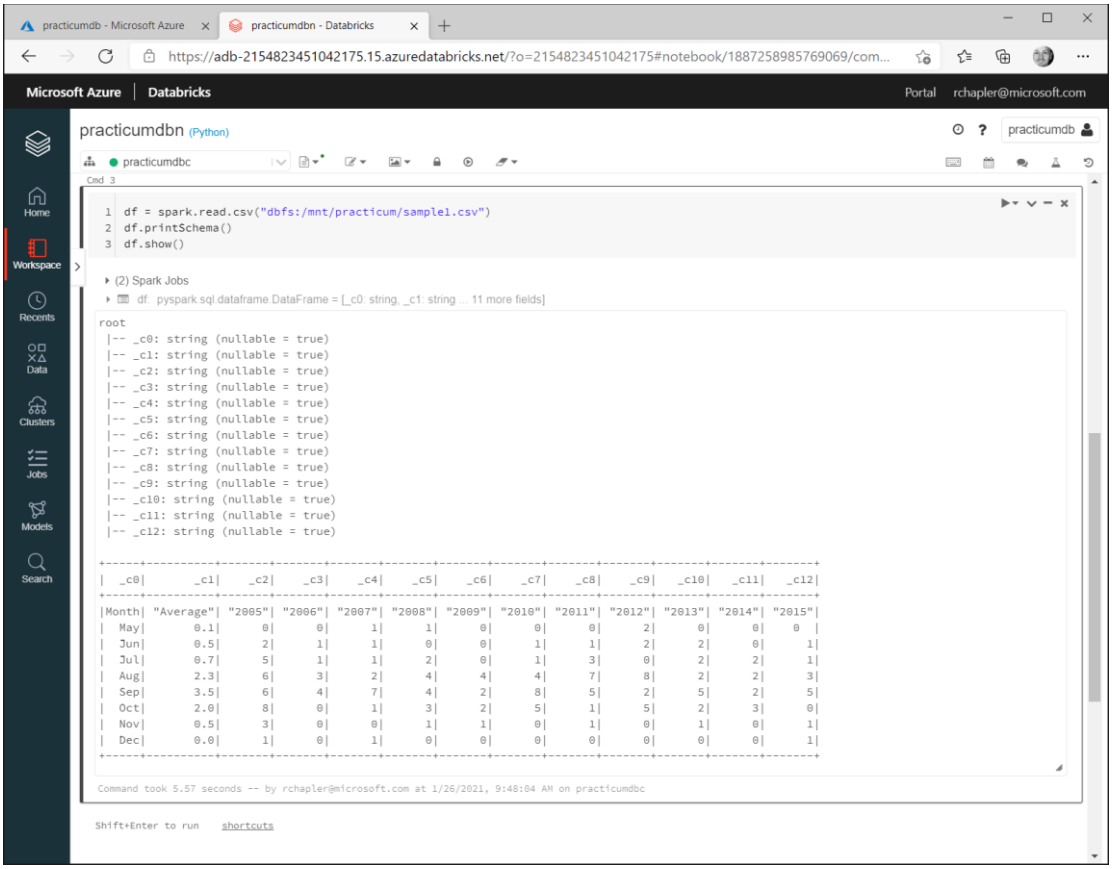


You should see your sample files at the mount point.

Proof 2 … Schema and Data

In your Databricks notebook, add a new cell, then paste and run the following code:

```
df = spark.read.csv("dbfs:/mnt/<UseCase>/sample1.csv")
df.printSchema()
df.show()
```



Observations:

- Schema … the schema is not defined, so the resulting interpretation calls out each field as data type string.
- Resultset … data formatting {e.g., first row as headers, double quotes, etc.} has not been applied.

Localize Mounted Data

Follow these instructions to localize data from a Data Lake mount.

Prerequisites

This solution requires the following resources:

- Data Lake
- Databricks

Prepare Logic

Why Localize?

Databricks File System (DBFS) is a distributed file system mounted into a Databricks workspace and available on Databricks clusters. DBFS is an abstraction on top of scalable object storage and offers the following **benefits**:

- Allows you to mount storage objects so that you can **seamlessly access data without requiring credentials**.
- Allows you to interact with object storage using directory and file semantics instead of storage URLs.
- Persists files to object storage, so you will not lose data after you terminate a cluster.

Databricks File System (DBFS) — Databricks Documentation | December 16, 2020

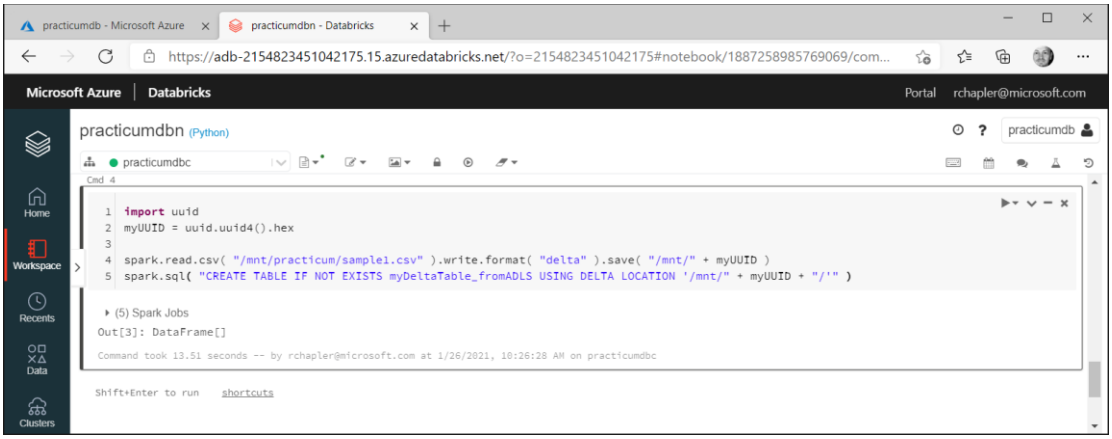
Though it is not listed here, many customers refer to the following benefits when justifying the decision to localize data from Azure Data Lake Storage to Azure Databricks:

- Performance Gains
- Decreased Compute Cost
- Leverage Benefits of Partitioning

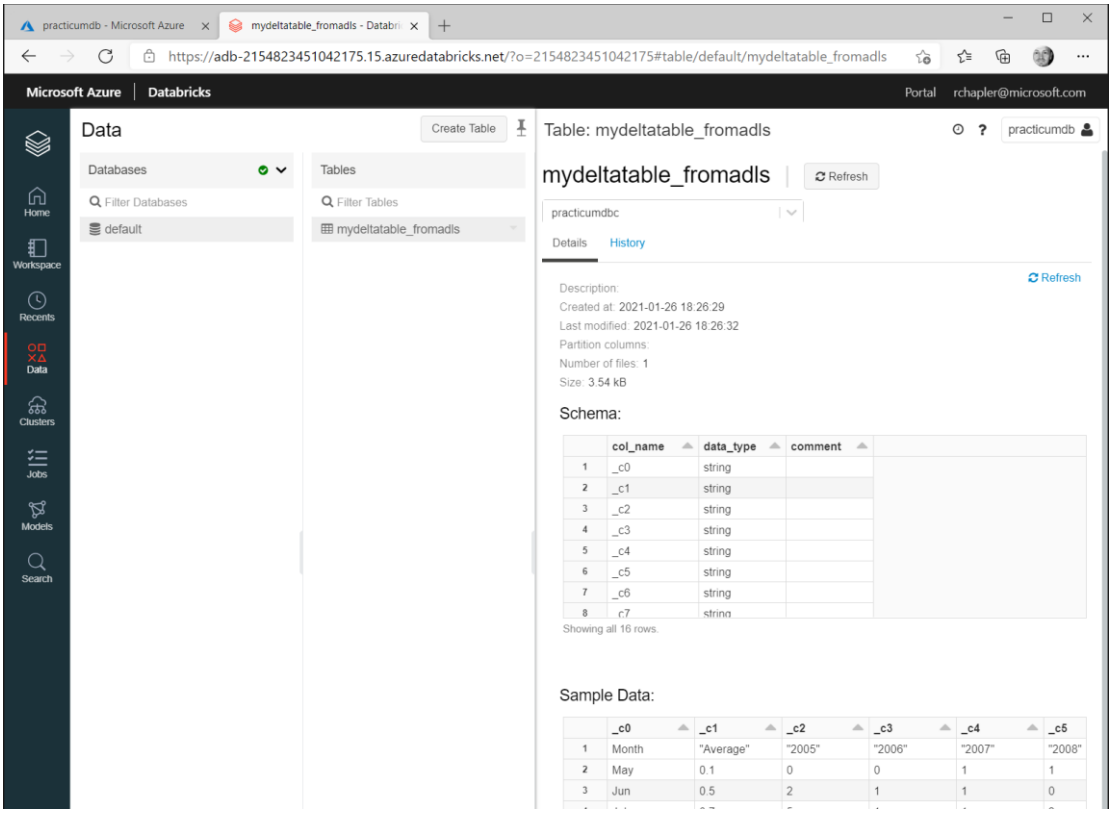
The Logic

Navigate to Databricks and the **<UseCase>dbn** notebook. Add a new cell and paste the following code:

```
import uuid
myUUID = uuid.uuid4().hex

spark.read.csv( "/mnt/<UseCase>/sample1.csv" ).write.format( "delta" ).save( "/mnt/" + myUUID )
spark.sql( "CREATE TABLE IF NOT EXISTS myDeltaTable_fromADLS USING DELTA LOCATION '/mnt/" + myUUID +
"/'" )
```



Click on **Data** in the navigation pane. Click on the **default** database, then the **mydeltatable_fromadls** table.



Observation: Previously noted schema and resultset challenges remain in the newly created table.

Simple Load

Synapse has many canned methods for loading and interacting with data.

Copy Data Tool

Follow these instructions to load data from Data Lake to Synapse.

Use Case

Example notes:

- “We want to load structured data from data lake to our Synapse SQL Pool”

Prerequisites

This solution requires the following resources:

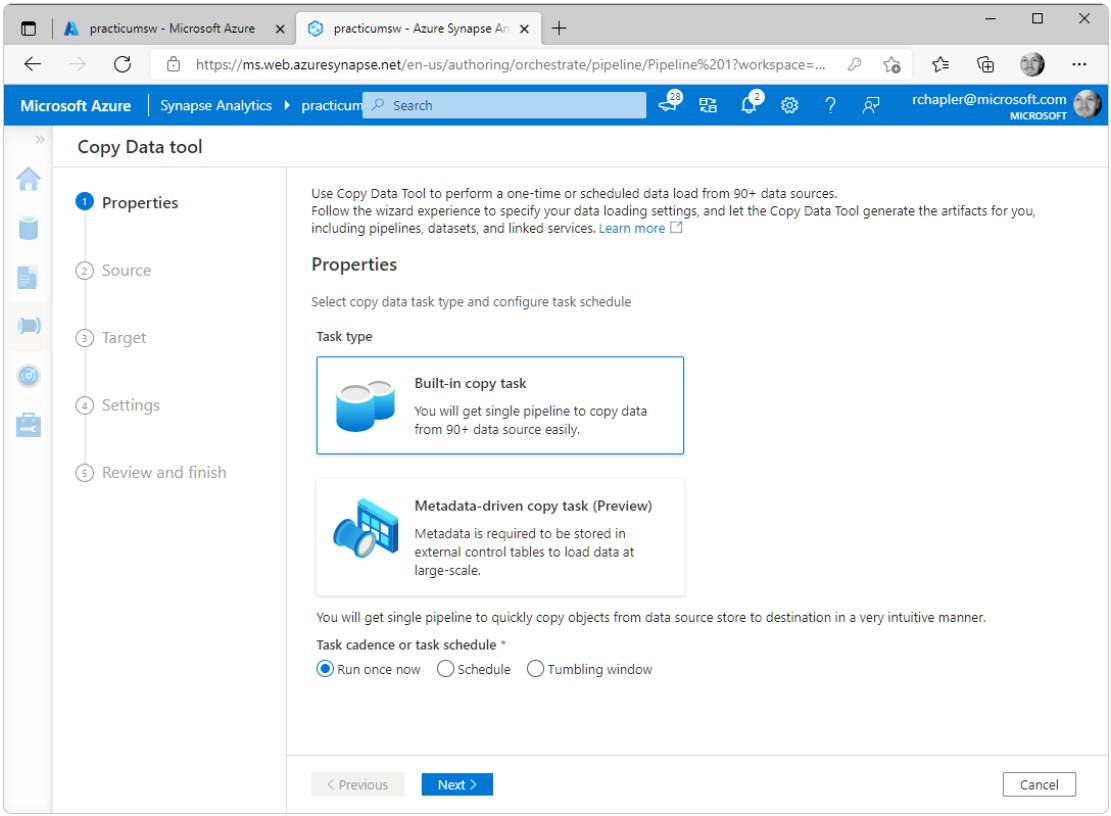
- Data Lake (with container and sample data)
- Synapse (with Dedicated SQL Pool)

Note: These instructions also apply (with minor differences) to Azure Data Factory.

Use “Copy Data tool”

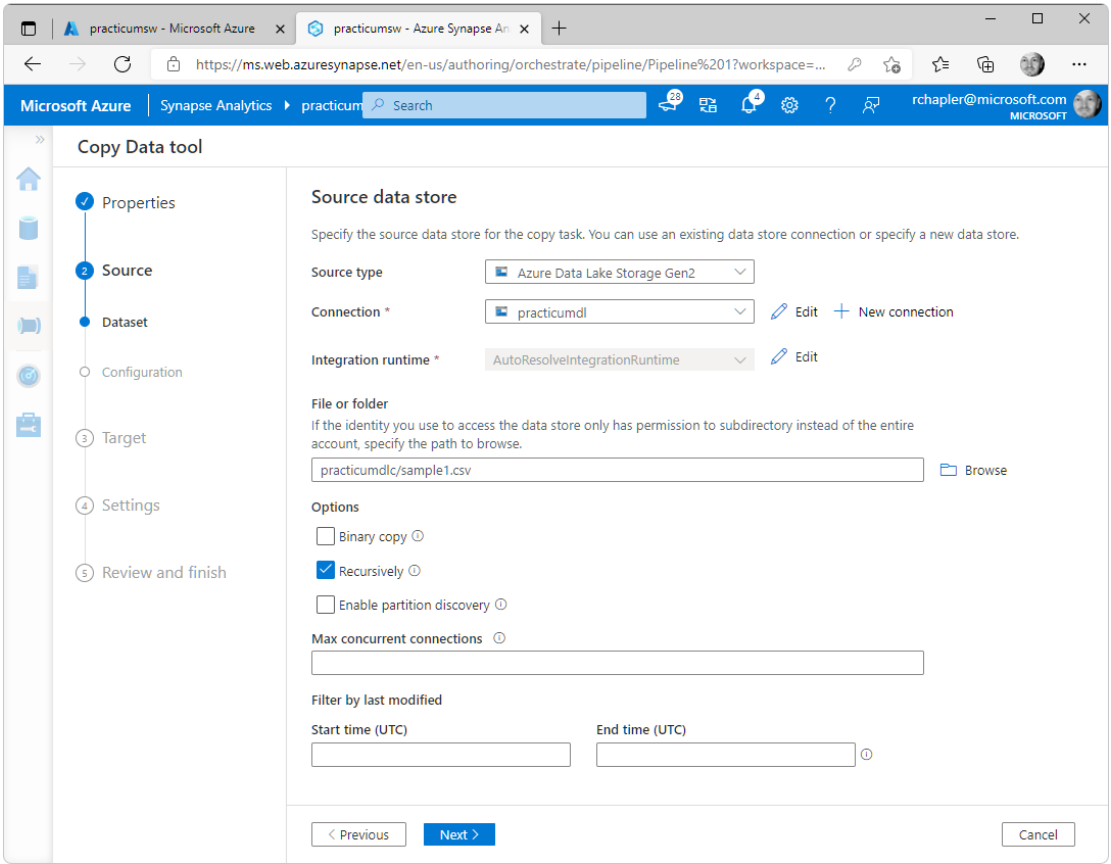
Navigate to your Synapse Analytics workspace, click the **Open** link on the “**Open Synapse Studio**” rectangle and then click the **Ingest** button.

Properties



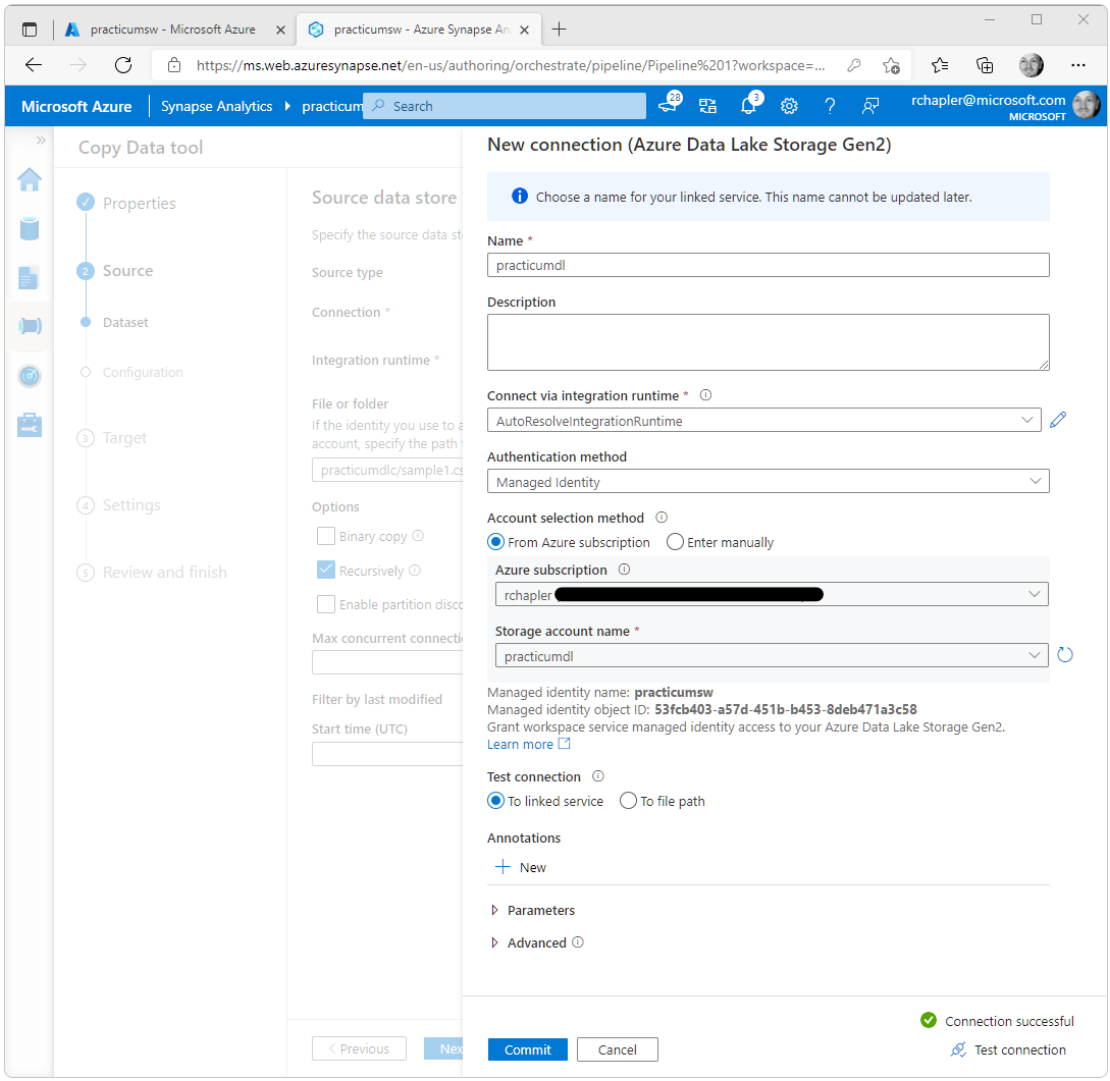
Confirm default select, “Built-in copy task” and “Run once now”, then click the “**Next >**” button.

Source > Dataset



Complete the "**Source data store**" form, including:

Source Type	Select "Azure Data Lake Storage Gen2"
Connection	Click the "**+ New connection**" button



Complete the "**New connection (Azure Data Lake Storage Gen2)**" form, including:

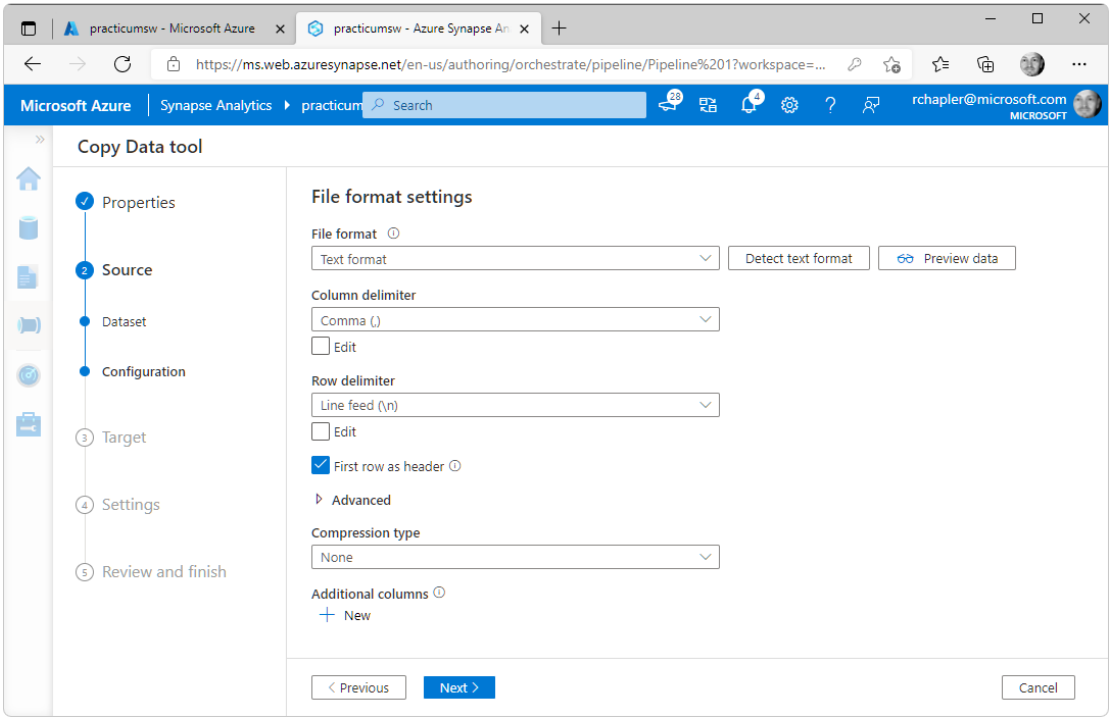
Connect via…	Confirm default selection, AutoResolveIntegrationRuntime
Account Selection Method	Confirm default selection, "**From Azure subscription**"
Storage Account Name	Select Data Lake

Click "**Test connection**" to confirm successful connection and then click the **Create** (or **Commit**) button.

Back on the “**Source data store**” form, click **Browse** to select container/file {e.g., “<UseCase>dlc/sample1.csv”}.

Click the “**Next >**” button.

Source > Configuration

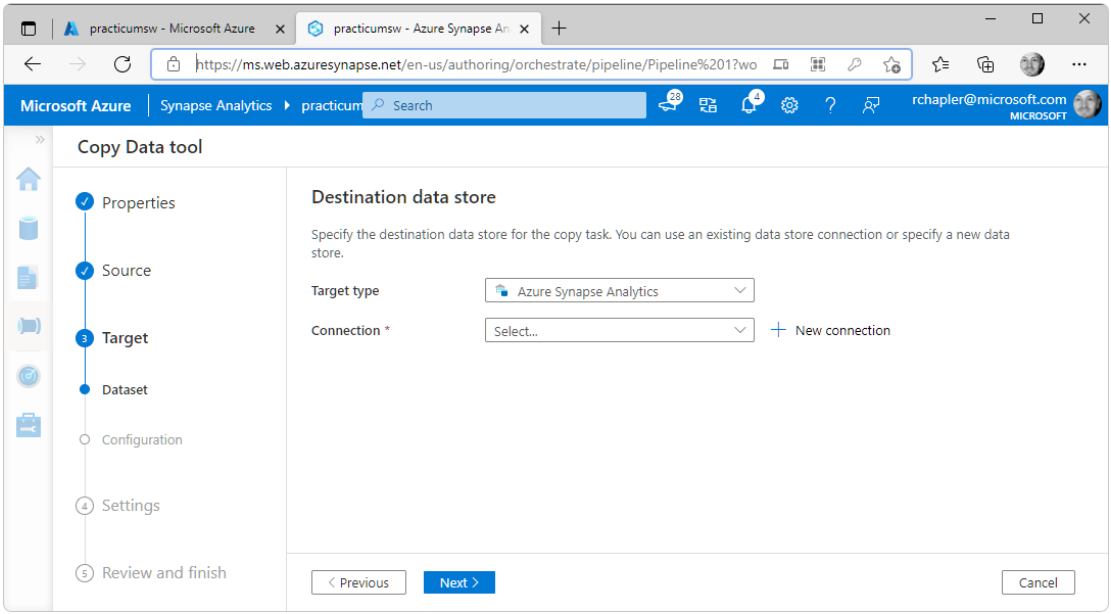


Complete the “**File format settings**” form, confirm default configuration and the resulting data preview.

Make changes if required {e.g., in this example, I check “**First row as header**” to correct handling of incoming data}.

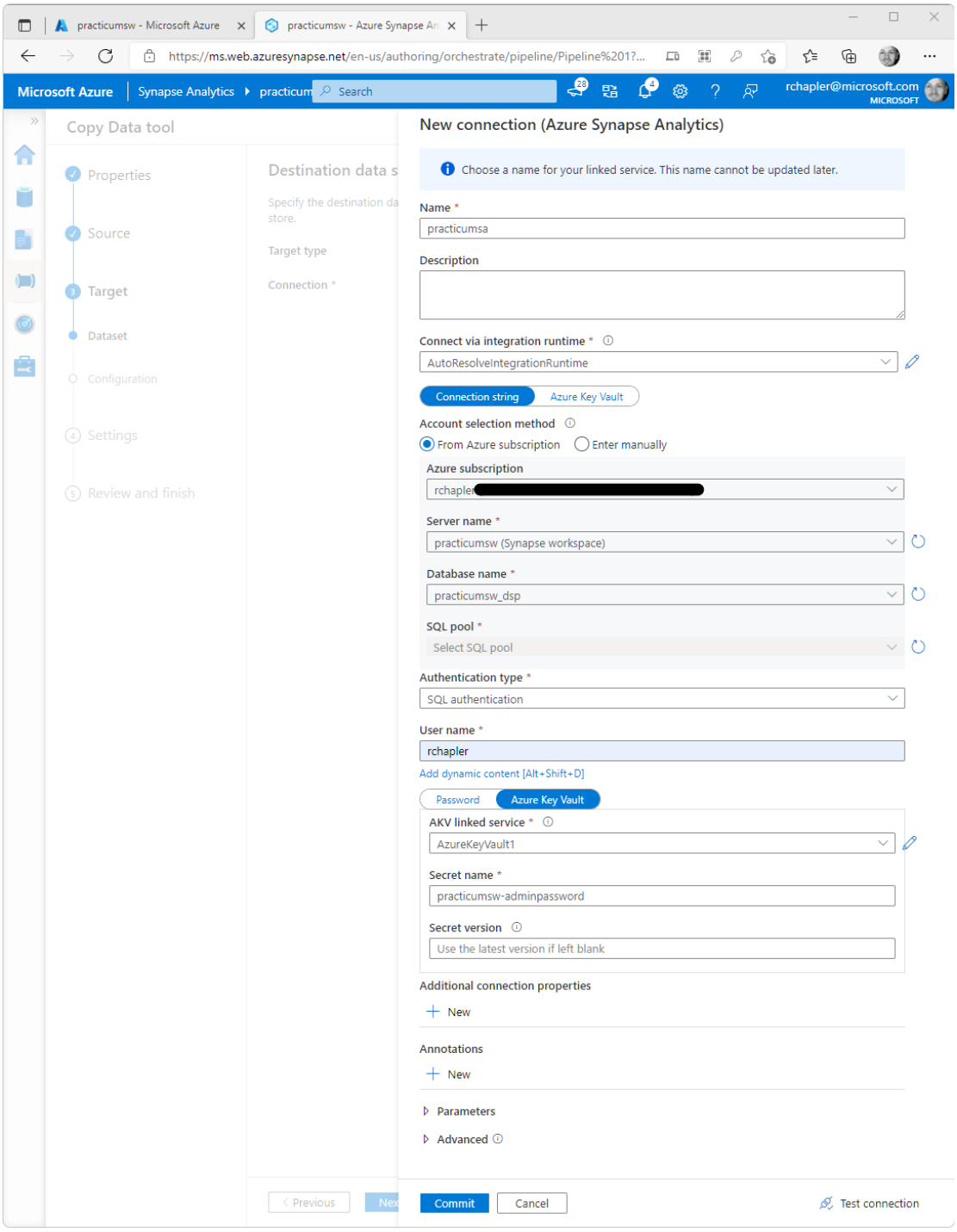
Click the “**Next >**” button.

Target > Dataset



Complete the "**Destination data store**" form, including:

Source Type	Select "Azure Synapse Analytics"
Connection	Click the "**+ New connection**" button

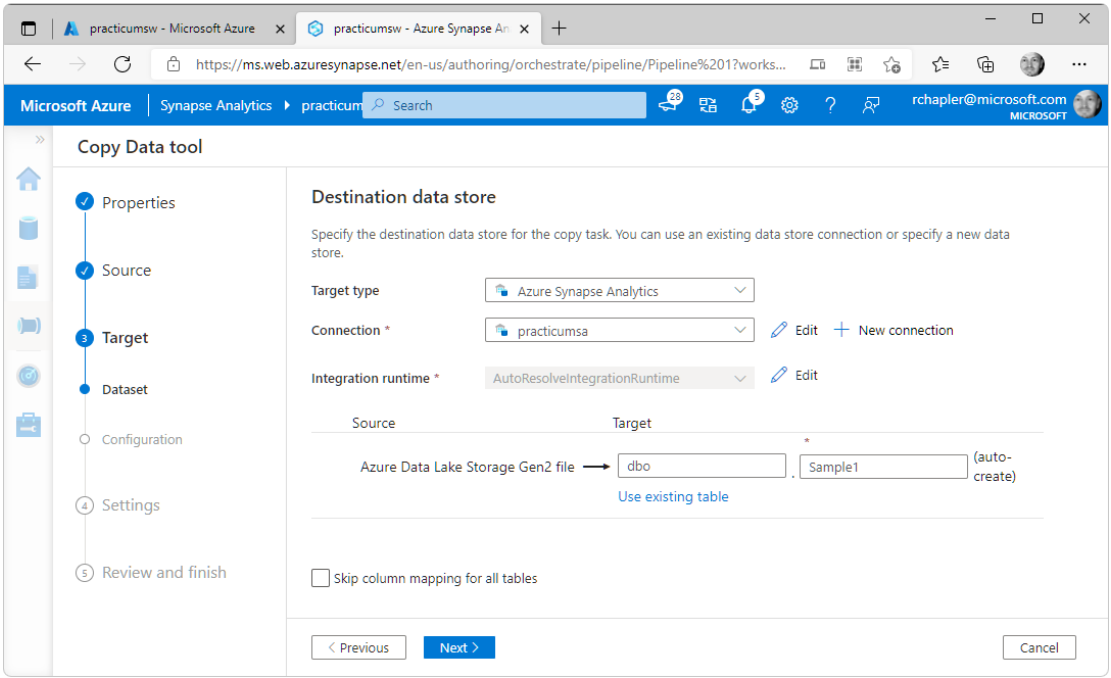


Complete the "**New connection (Azure Synapse Analytics)**" pop-out, including:

Connect via…	Confirm default selection, AutoResolveIntegrationRuntime

Account Selection Method	Confirm default selection, “**From Azure subscription**”
Server Name	Select Synapse Workspace
Database Name	Select your Dedicated SQL Pool
Authentication Type	Confirm default value, “SQL authentication”
User Name	Enter the “**SQL Server admin login**” value used during instantiation of Synapse
AKV Linked Service	Select the name of the Linked Service created for the Key Vault
Secret Name	Enter the Secret Name used to capture the Synapse Workspace administrator password

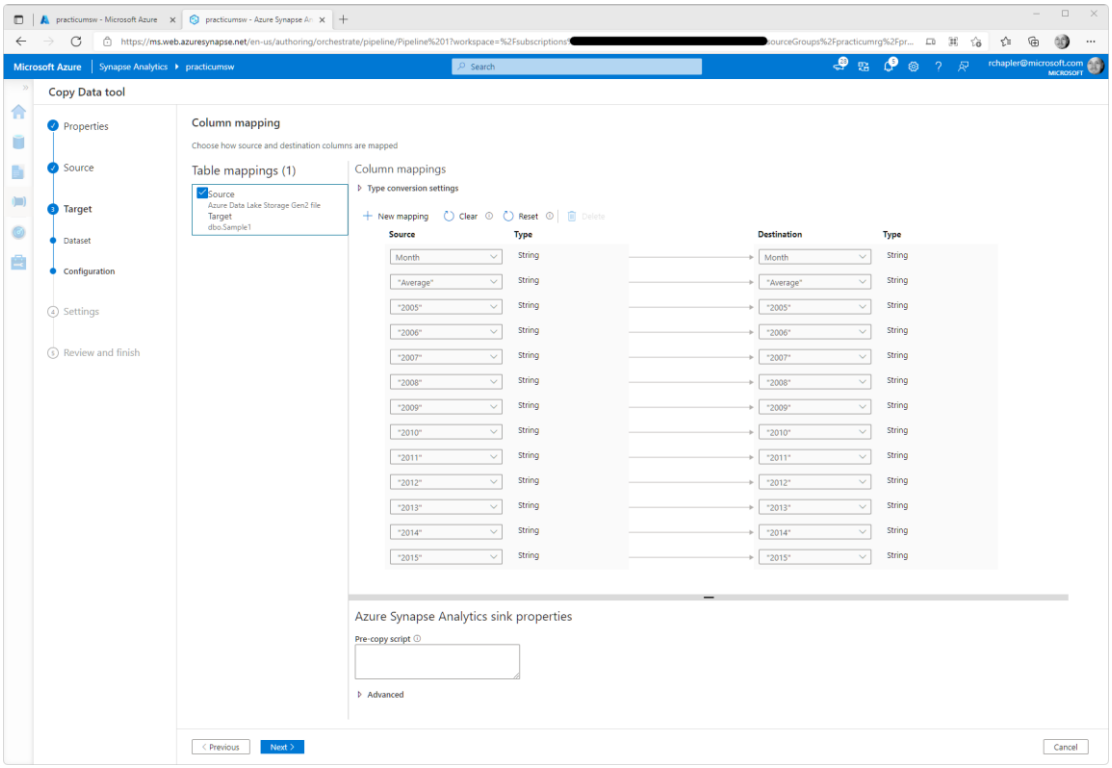
Click “**Test connection**” to confirm successful connection and then click the **Create** (or **Commit**) button.



Back on the “**Destination data source**” form, including target schema name {e.g., “dbo”} and target table name {e.g., “Sample1”}.

Click the “**Next >**” button.

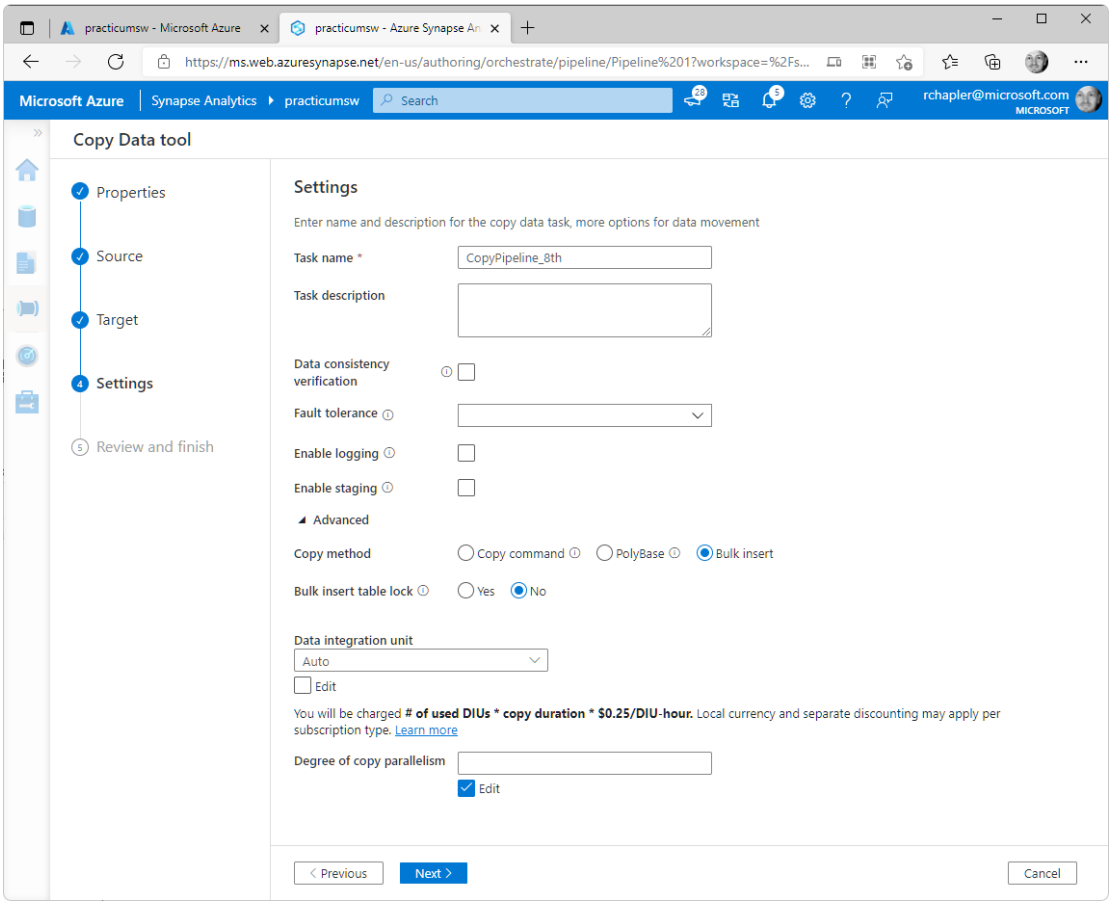
Target > Configuration



Review default column mappings; correct if required.

Click the “**Next >**” button.

Settings

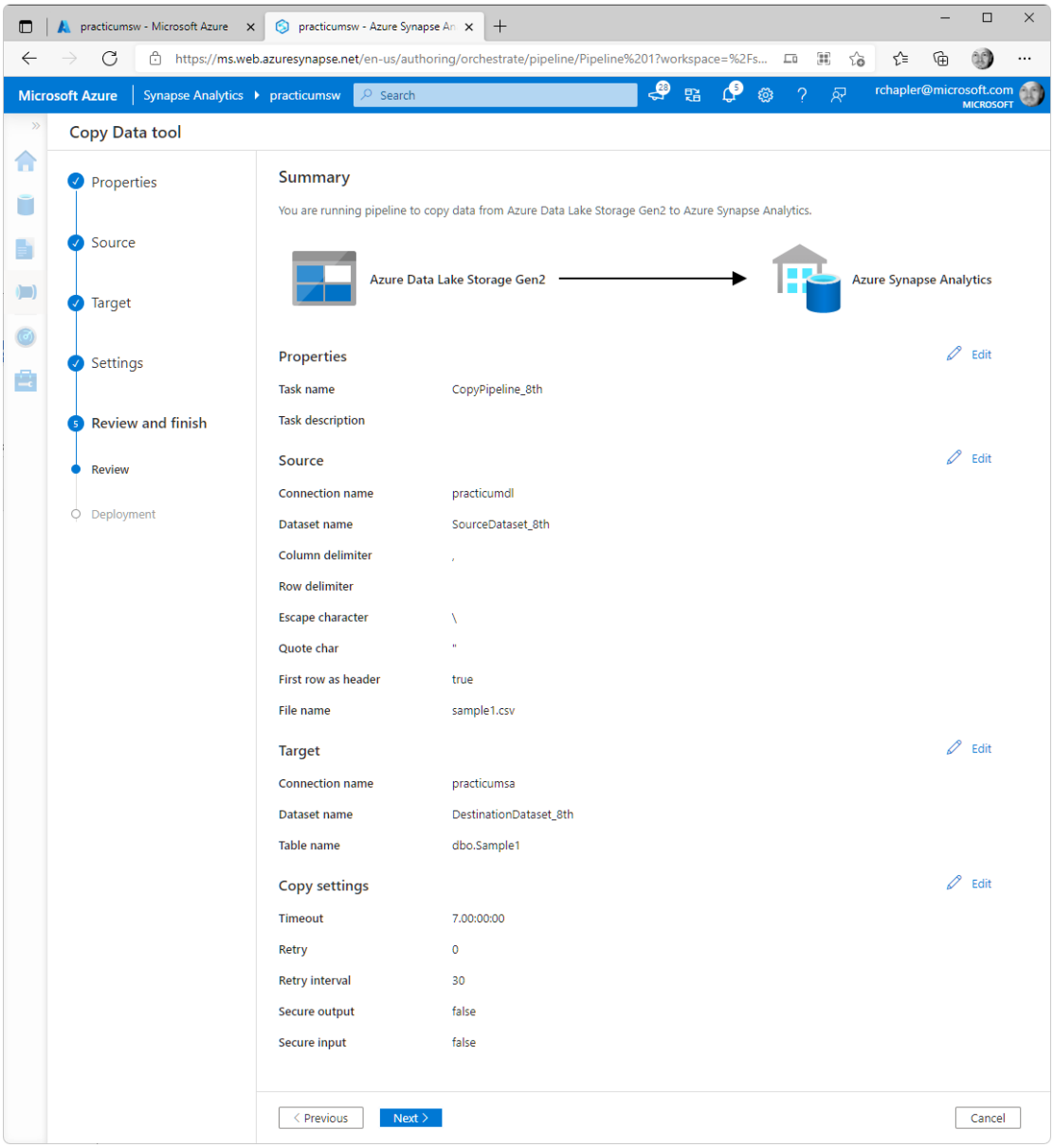


On the **Settings** form, including:

Enable Staging	Unchecked
Copy Method	Bulk insert

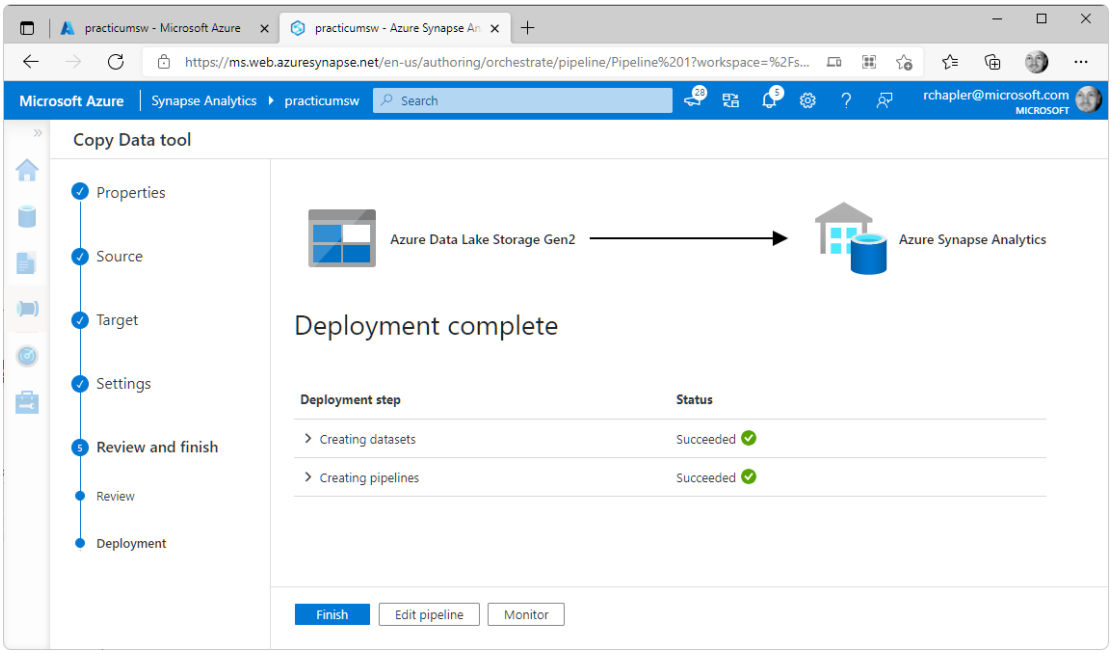
Click the "**Next >**" button.

Summary



Review configuration and then click the "**Next >**" button.

Deployment

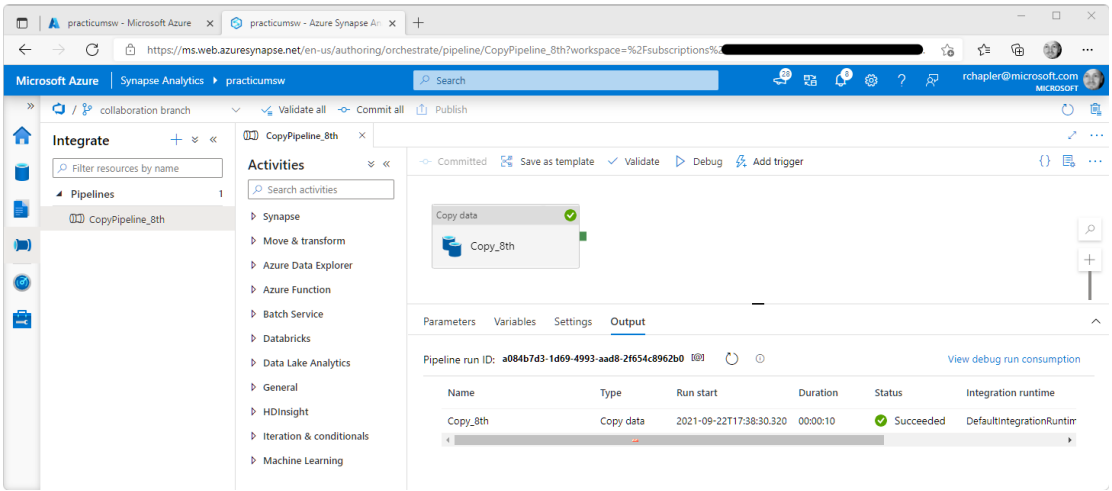


Monitor progress.

Click the **Finish** button on success.

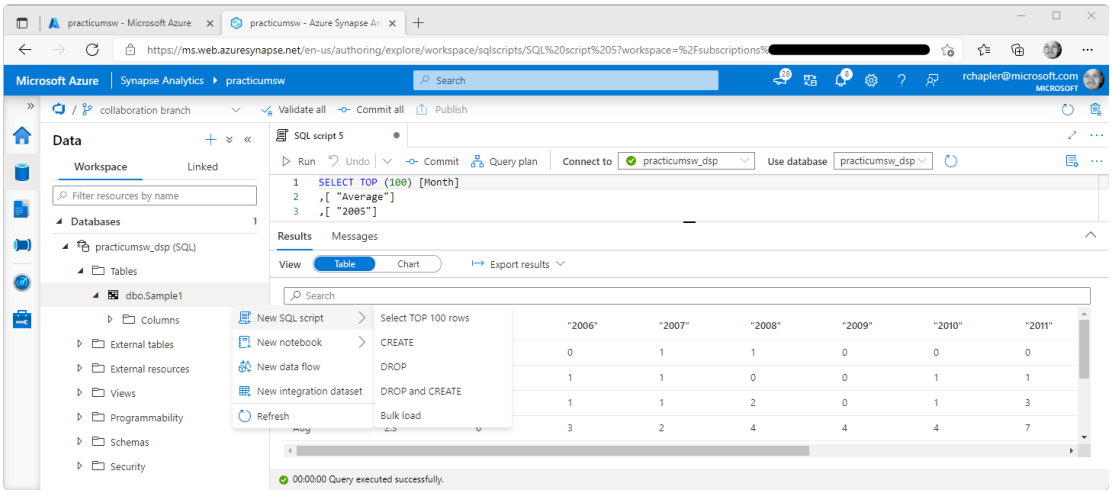
Confirm Success

"**CopyPipeline**…" will be added to the list of **Pipelines** in the Synapse, **Integrate** section.



Click **Debug** and confirm successful execution.

Click the **Data** icon in the navigation pane.



Expand **Databases** > {**Dedicated SQL Pool**} > **Tables** and confirm that you can see your new table.

Right-click on your table, select "**New SQL script**", and then "**Select TOP 100 rows**" in the drop-down menu.

Pipeline

Follow these instructions to load data from Data Explorer to Synapse.

Use Case

Example notes:

- “We want to load structured data from SQL to Data Explorer”

Prerequisites

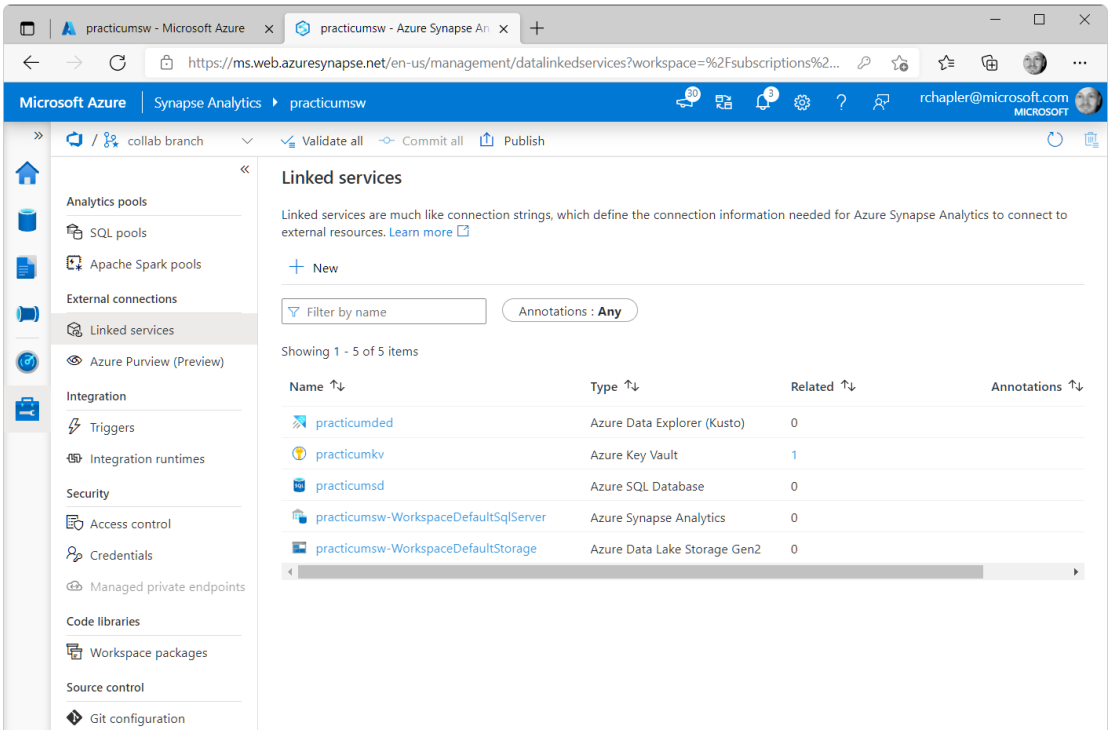
This solution requires the following resources:

- Data Explorer (with cluster, database, and sample Product table)
- SQL (with sample data)
- Synapse (with linked services)

Note: These instructions also apply (with minor differences) to Azure Data Factory.

Confirm Linked Services

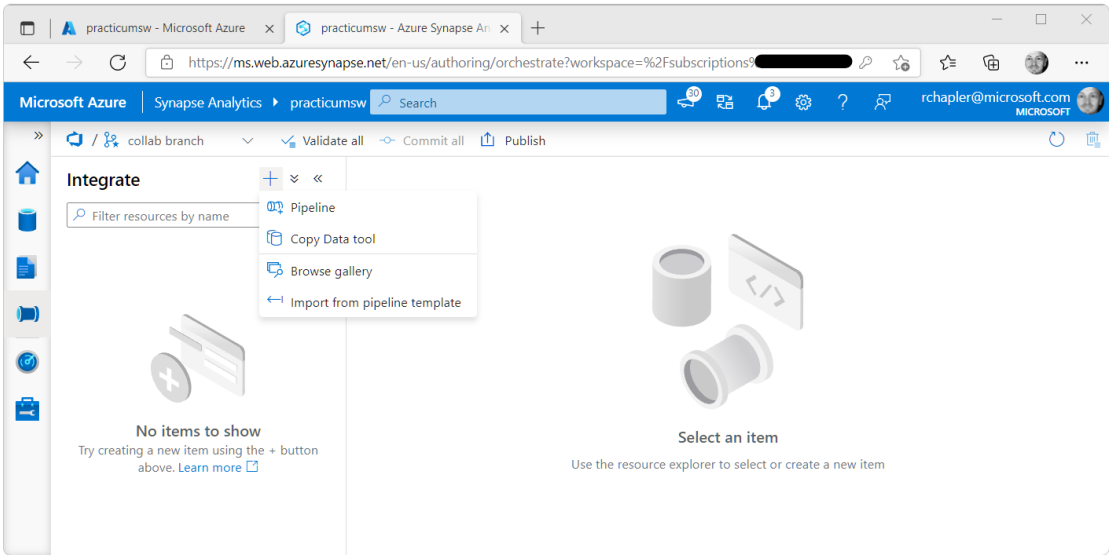
Navigate to your Synapse Analytics workspace and then click the **Open** link on the “**Open Synapse Studio**” rectangle. Select the **Manage** icon on the navigation and then “**Linked Services**” in the resulting navigation menu. Confirm linked service instantiation for: 1) Key Vault, 2) SQL and 3) Data Explorer.



Note: you are likely to see system-generated linked services {e.g., "<UseCase>sw-WorkspaceDefault…"} in addition to instantiated items.

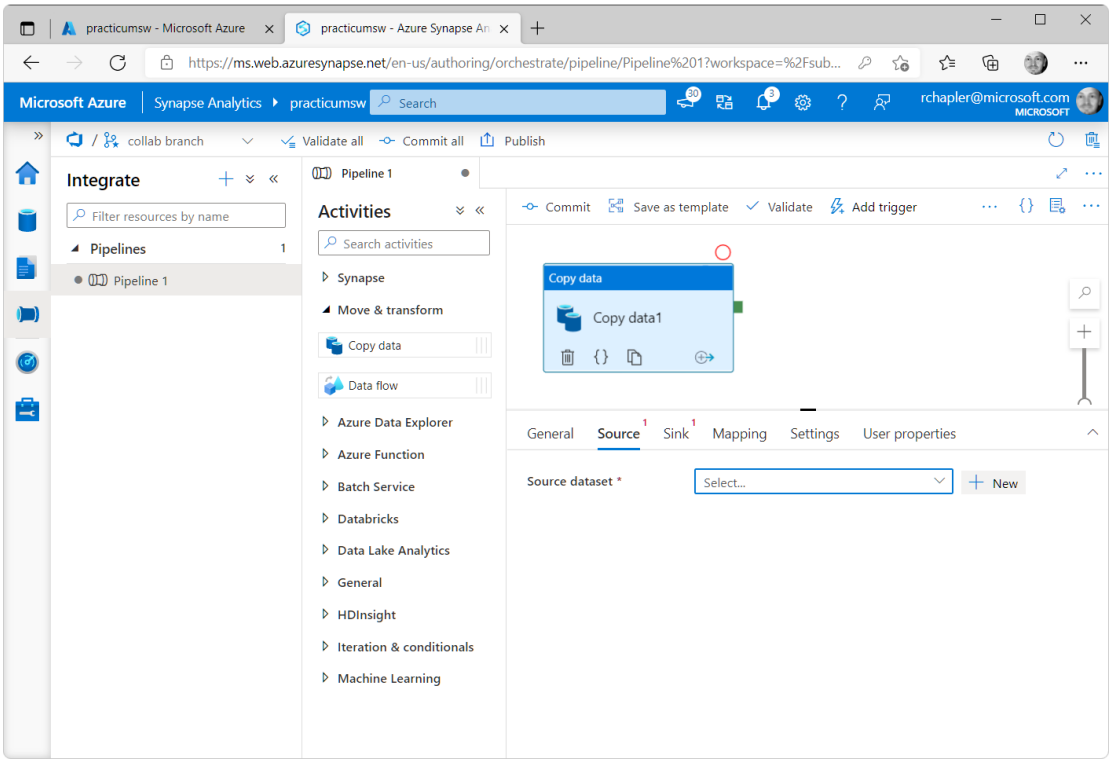
Create Pipeline

Click the **Integrate** icon in the navigation pane.



Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

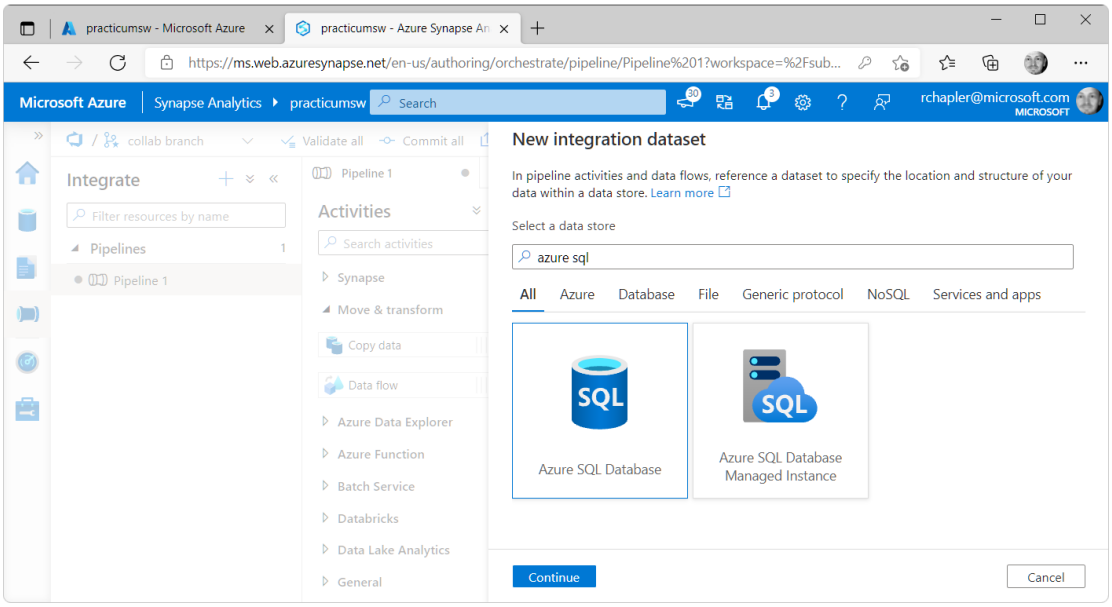
Select **Pipeline** from the resulting drop-down menu.



Add a "**Copy data**" activity.

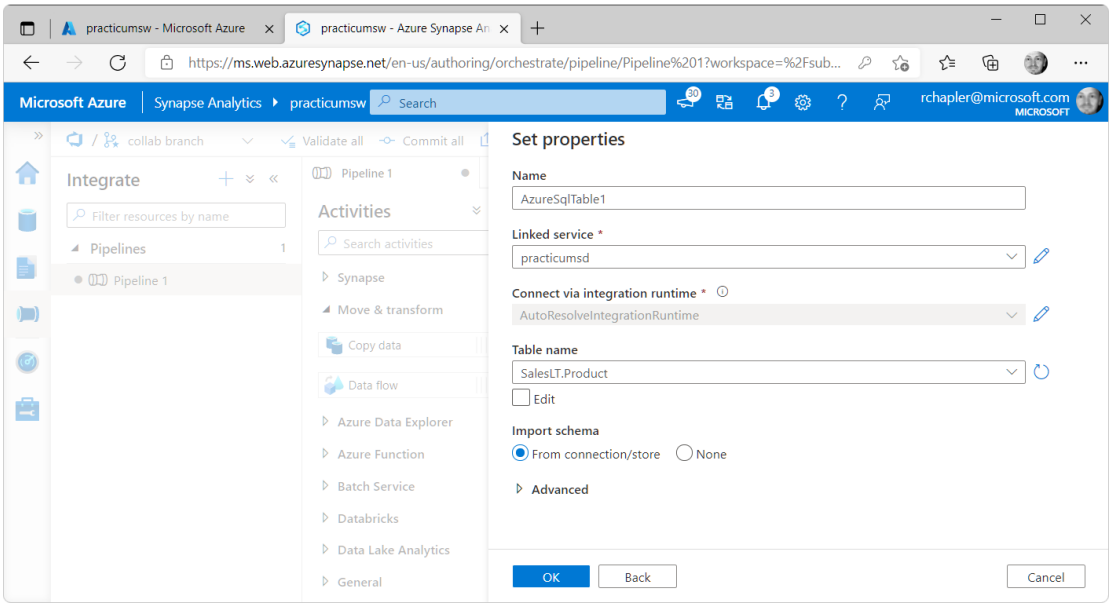
Source

On the **Source** tab, click the "**+ New**" button.



Complete the "**New integration dataset**" pop-out, search for and then select "**Azure SQL Database**."

Click the **Continue** button.

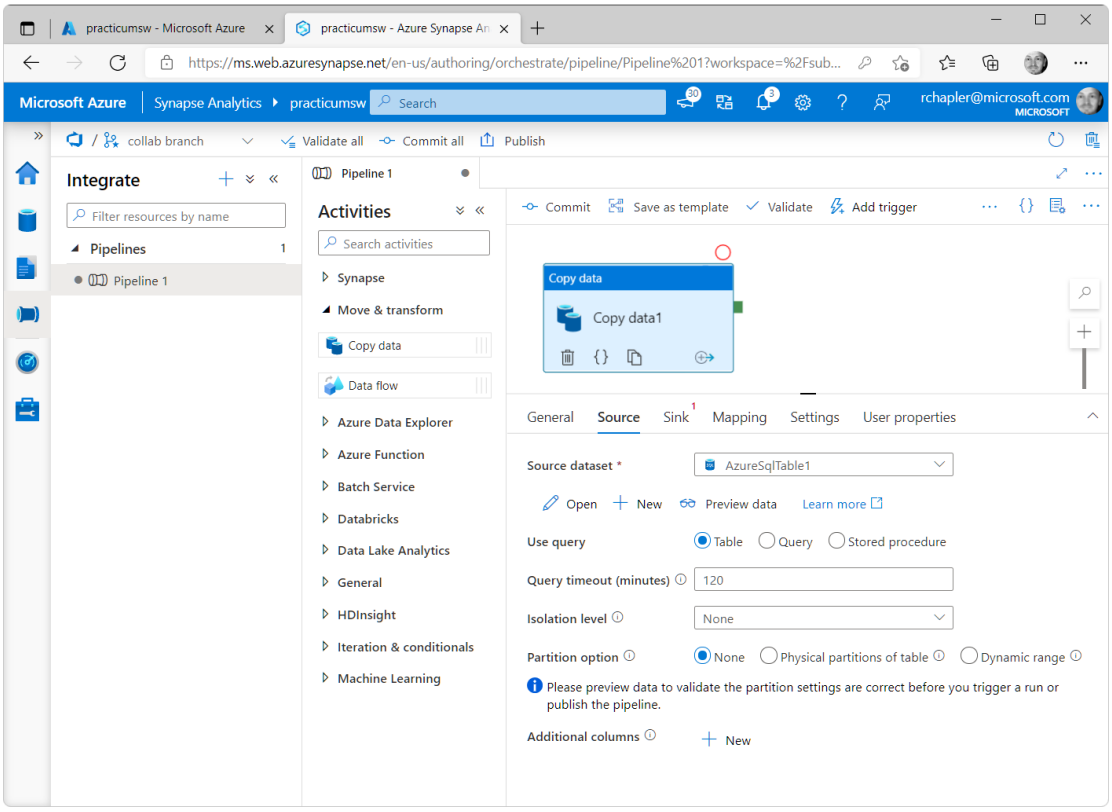


Complete the "**Set properties**" pop-out, including:

Linked Service	Select your Azure SQL linked service
Table Name	Select "SalesLT.Product"
Import Schema	Confirm default selection, "From connection/store"

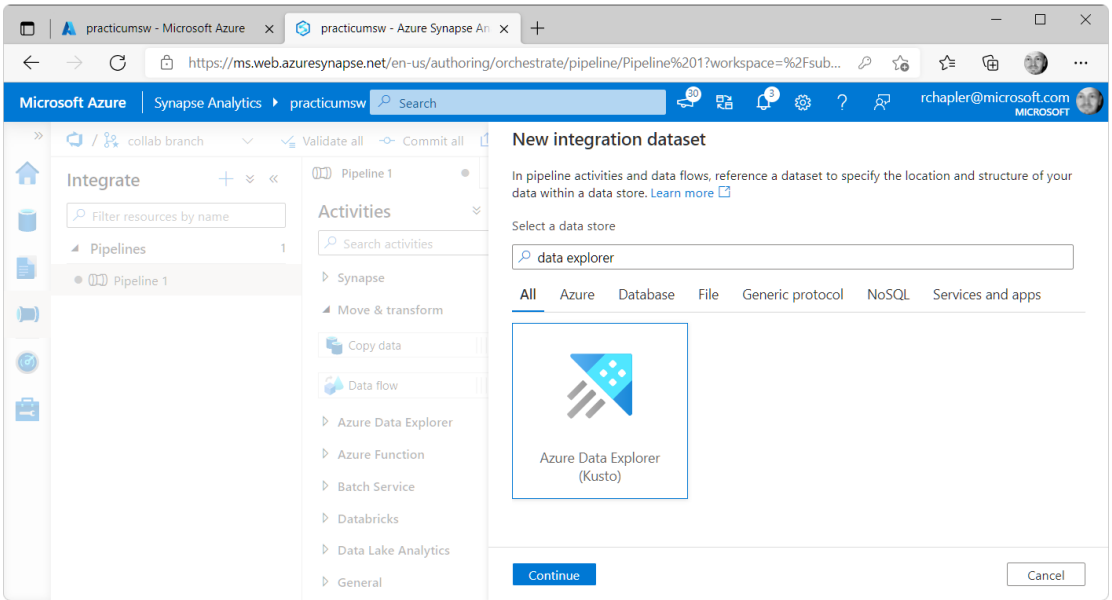
Leave all other settings with default values.

Click the **OK** button.



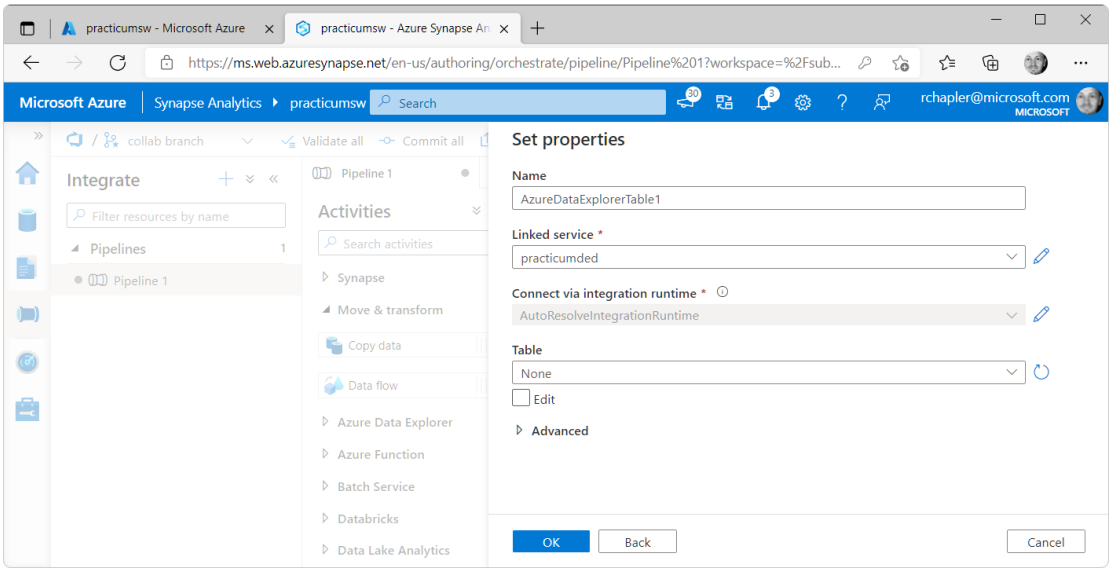
Sink

Click on the **Sink** tab and then click the "**+ New**" button.



Complete the “New integration dataset” pop-out, search for and then select “Azure Data Explorer (Kusto).”

Click the **Continue** button.

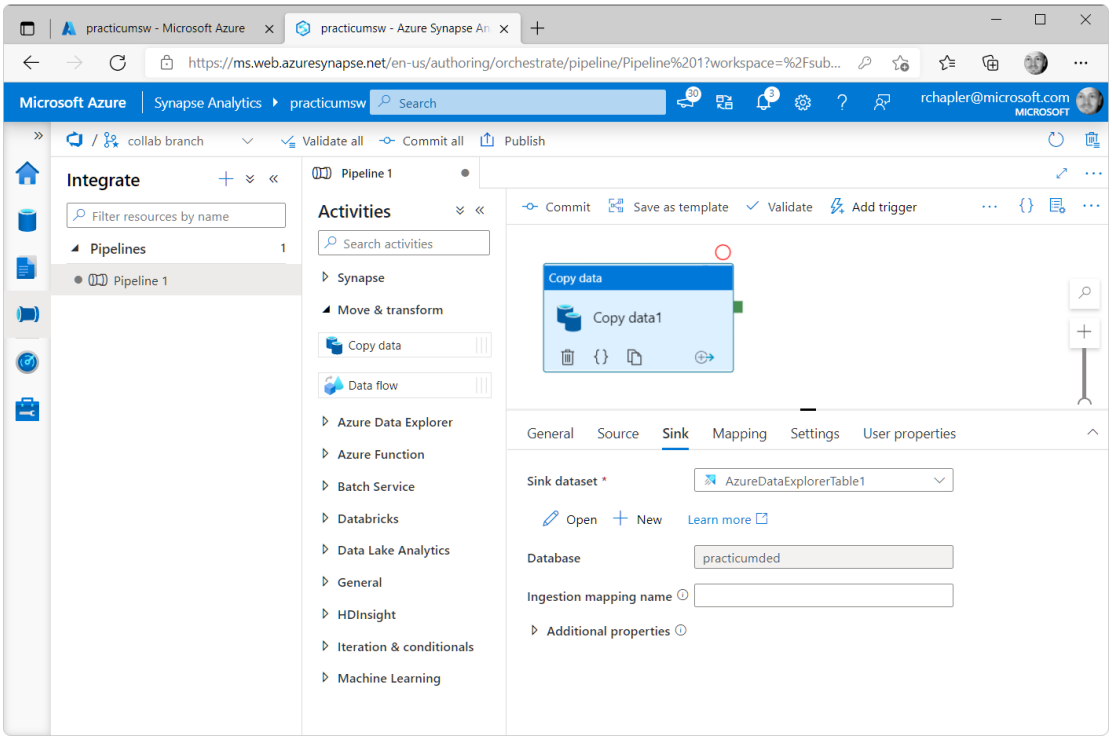


Complete the “**Set properties**” pop-out, including:

Linked Service	Select your Azure Data Explorer linked service
Table Name	Select **Product**

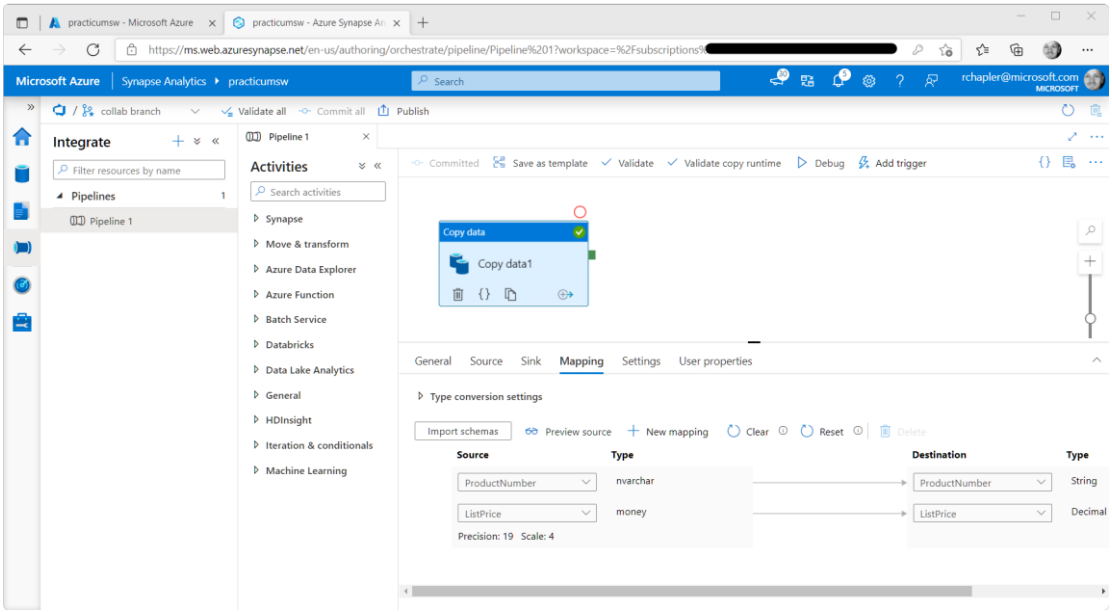
Leave all other settings with default values.

Click the **OK** button.



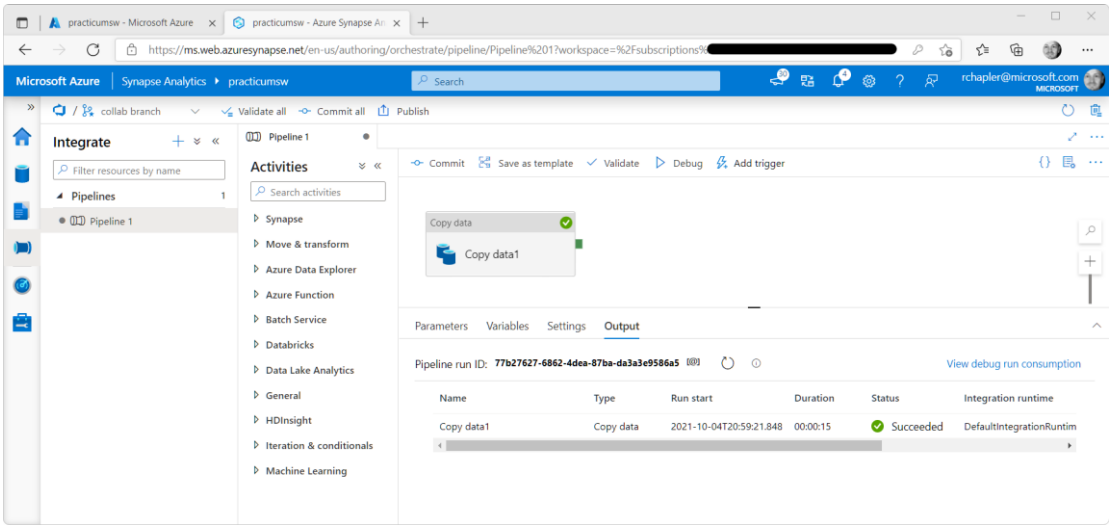
Mapping

Click on the **Mapping** tab and then click the "**Import schemas**" button.



Confirm Success

Click **Validate** to confirm that there are no errors, and then click **Debug** to confirm that the pipeline runs successfully.



Conditional Activity

Follow these instructions to demonstrate conditional logic in a pipeline.

Use Case

Example notes:

- “We want to include conditional handling in our pipeline”

Prerequisites

This solution requires the following resources:

- Synapse

Note: These instructions also apply (with minor differences) to Azure Data Factory.

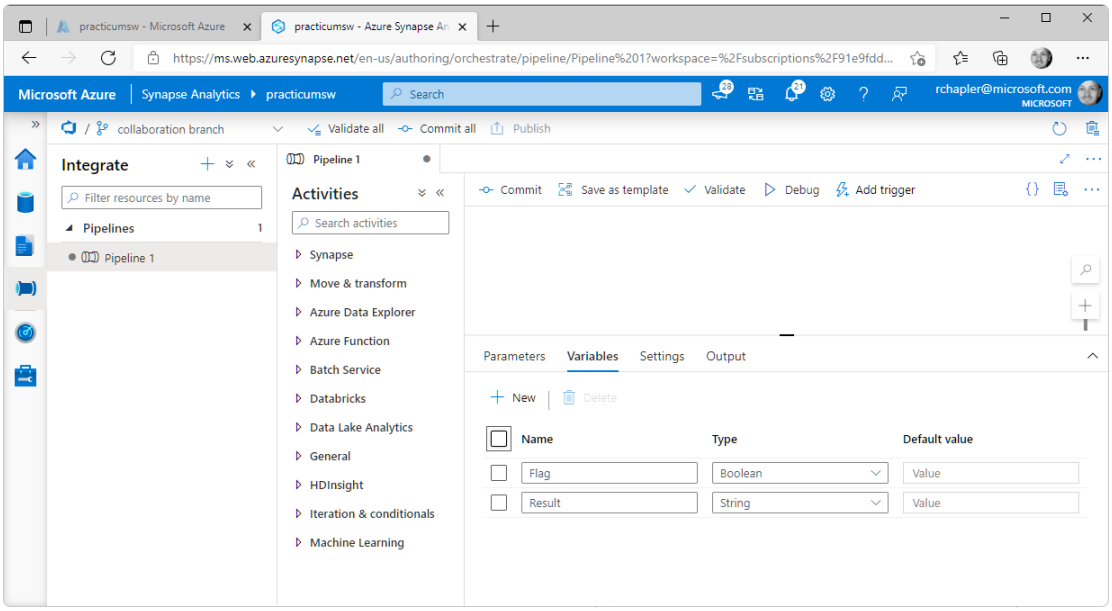
Create Pipeline

Navigate to your Synapse Analytics workspace and then click the **Open** link on the “**Open Synapse Studio**” rectangle.

Click the **Integrate** icon in the navigation pane.

Click the **+** icon just above and to the right of the “**Filter resources by name**” input.

Select **Pipeline** from the resulting drop-down menu.



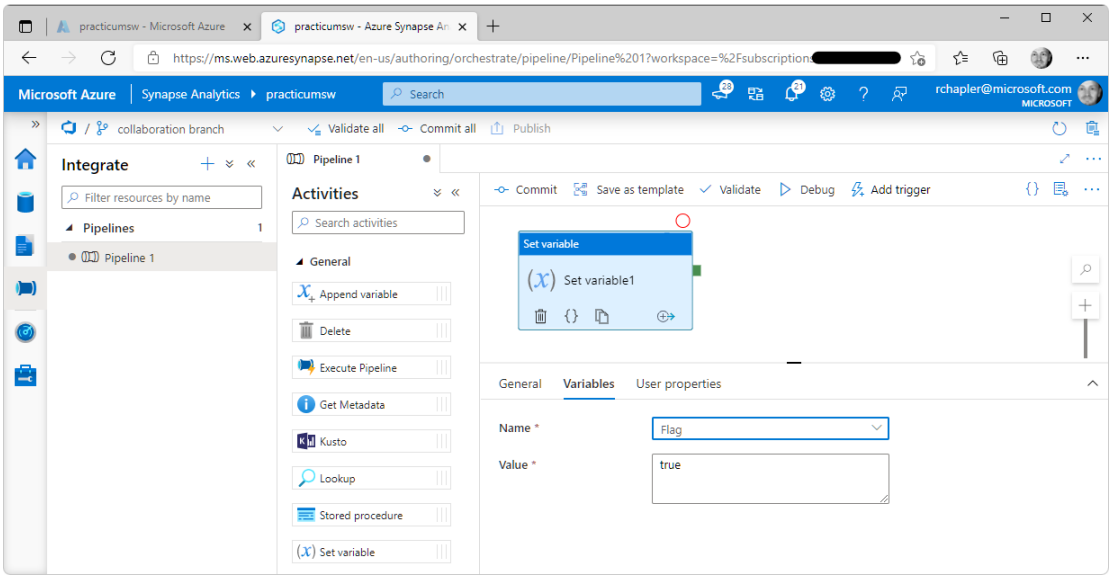
Click on the **Variables** tab.

Click “**+ New**” to add each of the following variables:

Flag (Boolean)	Will serve as a TRUE / FALSE trigger for the conditional result
Result (String)	For capturing an anecdotal result {e.g., “Success!”}

Activity 1, Set Variable

Expand **General** in the **Activities** bar, then drag-and-drop a “**Set variable**” component into the activity window.



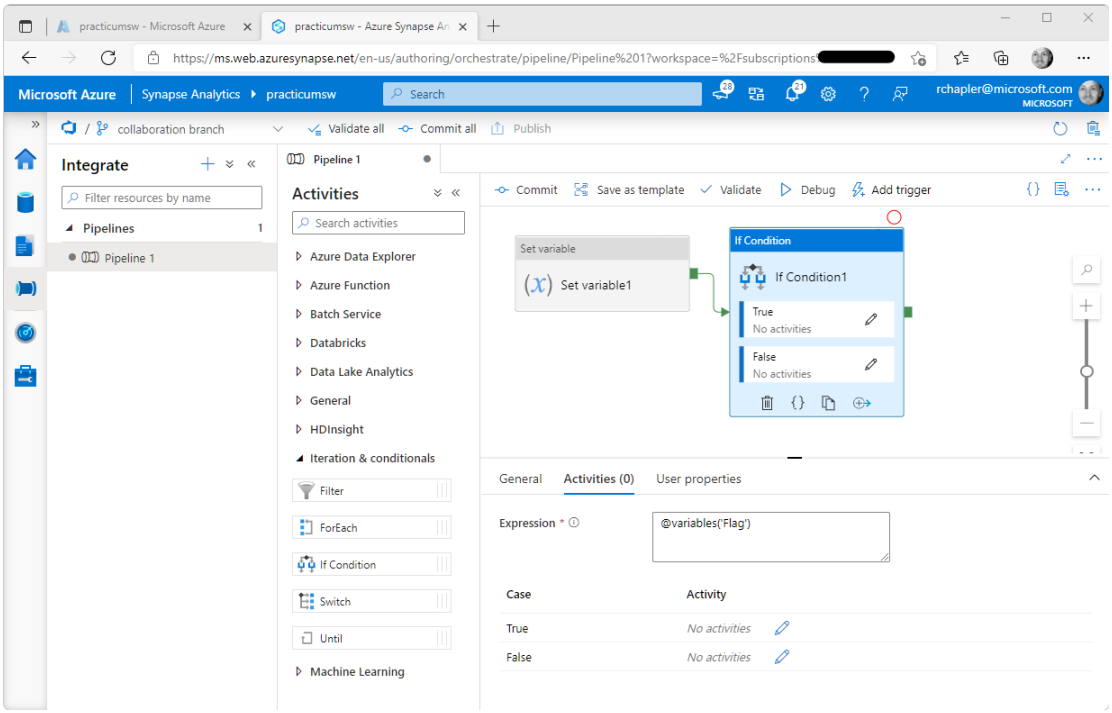
On the **Variables** tab of the “**Set Variable**…” component, including:

Name	Select **Flag** from the drop-down menu
Value	Enter **true**

Activity 2, If Condition

Expand “Iteration & conditionals” in the Activities bar.

Drag-and-drop an “**If Condition**” component into the pipeline.

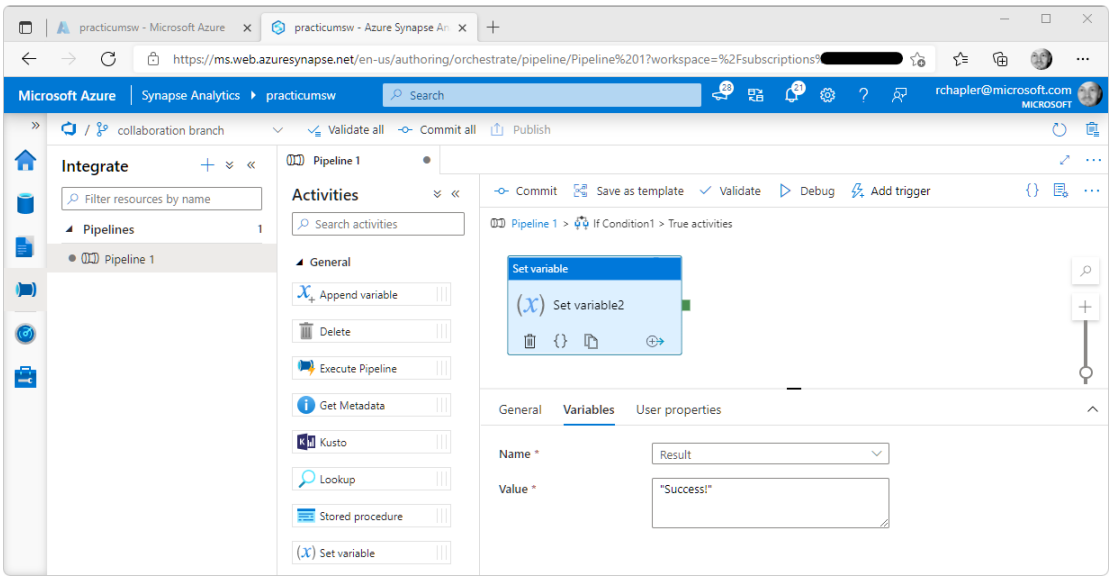


Create a dependency between the "**Set variable**" and "**If Condition**" components.

Click on the "**Activities (0)**" tab.

Click into the **Expression** textbox, and then enter the following: @variables('Flag')

Click on the pencil icon in the "**If Condition**" component, **True** selection.



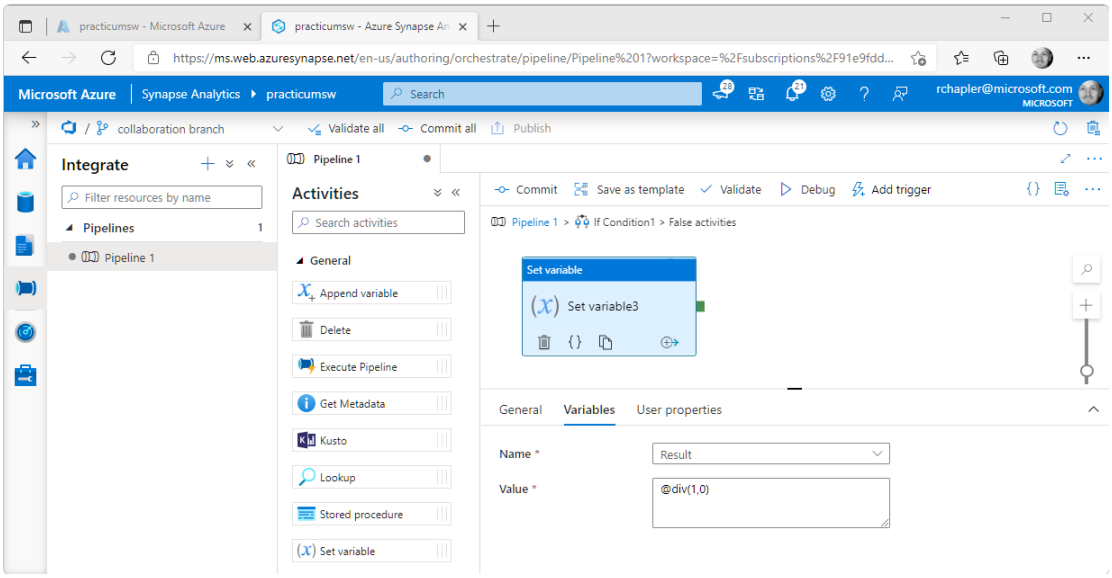
Expand **General** in the **Activities** bar, then drag-and-drop a "**Set variable**" component into the activity window.

On the **Variables** tab of the "**Set Variable**…" component, including:

Name	Select **Result** from the drop-down menu
Value	Enter "Success!"

Use cookie crumbs to navigate back to the main window.

Click on the pencil icon in the "**If Condition**" component, **False** selection.



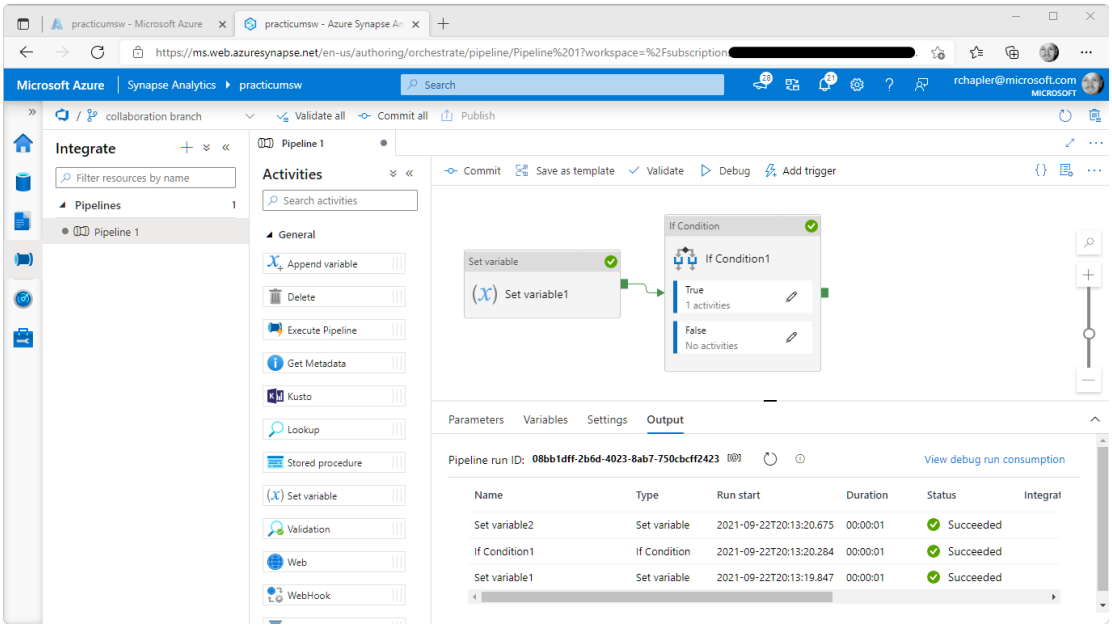
Expand **General** in the **Activities** bar, then drag-and-drop a "**Set variable**" component into the activity window.

On the **Variables** tab of the "**Set Variable**…" component, including:

Name	Select **Result** from the drop-down menu
Value	Enter **@div(1,0)**… this will force an error result

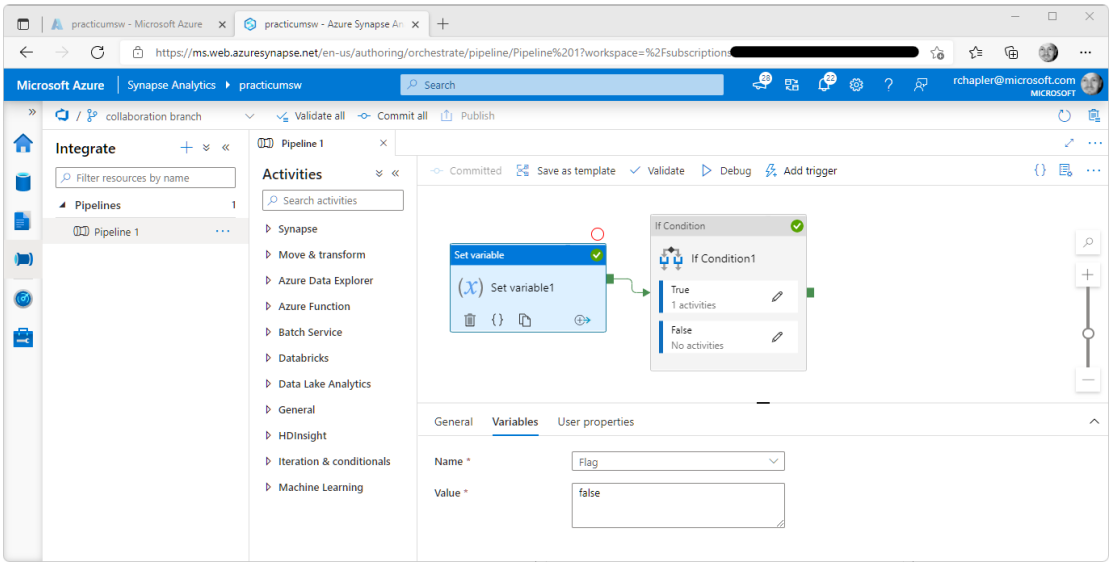
Confirm Success

Click **Debug**.



You should see **Succeeded** messages given current variable settings.

Next, we will make a pipeline change that will force pipeline failure.



Navigate to the "**Set variable1**" component, **Variables** tab.

Replace the **true** value with **false** and then click **Debug**.

You will see **Failed** messages given current variable settings.

Incremental Load

There is more than one way to incrementally load of data. Your chosen method will hinge on questions like…

- Does the source offer “delta comparison”-type functionality?
- Does the source include both identifier and waterline column? If not, could you use some form of a CHECKSUM functionality?
- Does the source include a “soft delete” column (and corresponding timestamp)?
- Does the Azure resource you plan to use {e.g., pipeline, data flow, etc.} provide for connection to the source? Is data staging required?
- Is it cheaper to separately load an initial batch of data {e.g., historical}?

The following sections explore use cases and corresponding methods (based on these criteria).

Pipeline

Follow these instructions to establish incremental load using a Synapse Pipeline to move data from a SQL database to a Data Explorer database.

Use Case

Example notes:

- “We want to efficiently load data from an **Oracle** operational data source”
- “We want to capture a time series of delete, insert, and update events and provide for “time travel” through data”

Through additional research we learn that the Oracle currently lacks a corresponding Data Flow connector, so we must use a Pipeline.

Prerequisites

This solution requires the following resources:

- SQL (with sample database) … using this **in lieu of Oracle** given complexity of instantiation
- Synapse with Data Explorer Pool, Data Explorer Database, and Linked Services for Data Explorer and SQL

Note: These instructions also apply (with minor differences) to Azure Data Factory and Azure Data Explorer.

Prepare Source

Navigate to SQL, click the "**Query editor**…" navigation icon, and authenticate. Add IP to firewall settings, if necessary. Execute the following logic:

```
CREATE TABLE [dbo].[Product_Pending] ( [_id] varchar(16), [_op] varchar(8) )
```

Prepare Destination

Navigate to Synapse Studio, click the **Develop** navigation icon, click "**+**" and select "**KQL script**" from the resulting drop-down menu.

Create Destination Table

Execute the following logic:

```
.create table _product (
  ProductId:int
  , ProductNumber:string
  , Name:string
  , ListPrice:decimal
  , ModifiedDate:datetime
  , _id:string
  , _op:string
  , _dt:datetime
  ) with ( folder = 'bronze' )
```

(Partial) logic explanation:

- Included columns should match their source counterpart (name and equivalent data type)
- The "**_id**" column provides a string-based, generic capture of various identifier data types {e.g., int, guid, etc.} and situations where unique identification may require concatenation of multiple fields {e.g., "123-456"}
- The "**_op**" column provides for flagging of **delete**, **insert**, or **update** operations
- The "**_dt**" column provides standard capture of the waterline date value for a given row
- "**with (folder = 'bronze')**" groups destination tables {i.e., the raw data} into a "**bronze**" folder

Create Materialized View ("…_Latest")

Execute the following logic:

```
.create materialized-view with ( folder = 'silver',backfill = true ) _product_latest on table _product {
  _product
  | extend _opdt = ingestion_time()
  | summarize arg_max(_opdt, *) by _id
  }
```

(Partial) logic explanation:

- "**backfill = true**" will include all existing records (as opposed to only including records ingested after view creation)
- "**extend _opdt**…" surfaces ingestion_time()
- "…**arg_max(_opdt, *) by _id**" finds and returns the row where _opdt is maximized for a given _id

Note: Materialized Views are not instantaneously populated… if you backfill a large amount of data and then run a query against the materialized view quickly after, you are likely to see incomplete results. If it is important to access the data quickly {i.e., soon after addition}, consider using a Function rather than a Materialized View.

Create User-Facing Function

Simplify user experience {i.e., "I want to see the latest data and I do not care about deleted rows!"} by preparing a Function in Data Explorer.

Execute the following logic:

```
create-or-alter function product( excludeDeletes: bool = true )
  { _product_latest | where _op != iif( excludeDeletes, 'delete', '') }
```

Create _Waterline Table

Execute the following logic:

```
.create table _product_waterline ( _id : string, _dt : datetime ) with ( folder = 'bronze' )
```

The second column might also be named and data-typed for values like "_cs" {i.e., checksum}, "_hash", "_hex" depending on what you need to use to waterline the source.

Create _Pending Function

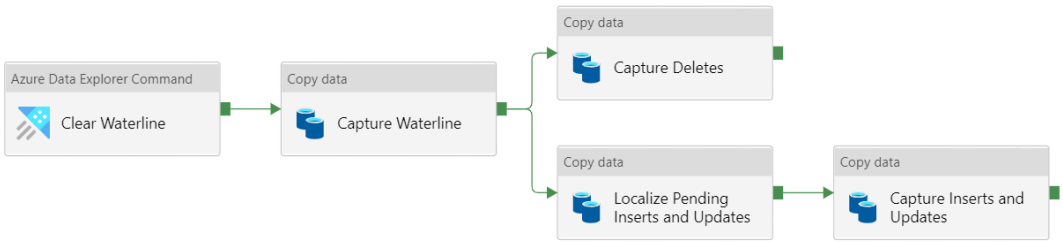
Execute the following logic:

```
.create function _product_pending with (folder='bronze')
{
    _product_latest
    | project latest_dt = _dt, latest_id = _id, latest_op = _op
    | join kind = fullouter ( _product_waterline | project waterline_id = _id, waterline_dt = _dt )
        on $left.latest_id == $right.waterline_id
    | project _id = case(
            isnull(waterline_dt) and latest_op != 'delete', latest_id // delete
            , isnull(latest_dt), waterline_id // insert
            , latest_dt != waterline_dt, waterline_id // update
            , '' )
        , _op = case(
            isnull(waterline_dt) and latest_op != 'delete', 'delete'
            , isnull(latest_dt), 'insert'
            , latest_dt != waterline_dt, 'update'
            , '' )
    | where not(isempty(_id)) and _op in ('delete', 'insert', 'update')
}
```

Note: If you are using an alternative to waterline "_dt" {e.g., checksum, hex, hash, etc.}, you may have to replace the use of ISNULL with ISEMPTY

Create Pipeline

When we are finished, our pipeline will look and function as snipped below.



Navigate to Synapse Studio, click the **Integrate** navigation icon, click "**+**" and select **Pipeline** from the resulting drop-down menu.

Activity 1: Clear Waterline

This activity will clear any previously created waterline data.

Expand "**Azure Data Explorer**" in the **Activities** bar, then drag-and-drop an "**Azure Data Explorer Command**" component into the activity window.

On the **Connection** tab, select the Data Explorer Linked Service from the drop-down menu.

On the **Command** tab, paste the following KQL in the Command textbox.

```
.clear table _product_waterline data
```

Click **Debug** and monitor result to confirm successful progress.

Activity 2: Capture Waterline

This activity will capture a dataset that will be used to identify deleted, inserted, or updated records. Capture of _waterline data to database is necessary in Pipelines because unlike Data Flow, there is no in-flow use of data.

Expand "**Move & Transform**" in the **Activities** bar, drag-and-drop a "**Copy data**" component into the activity window and then create a dependency from the "**Clear Waterline**" component.

On the **Source** tab, including:

Source Dataset	Create a new "**Azure SQL Database**" dataset using the **SQL** linked service and the "**None**" table {e.g., "**<UseCase>sd_None**"}
Use Query	Select the **Query** radio button
Query	Paste the following T-SQL: `SELECT ProductId _id,MAX(ModifiedDate) _dt FROM SalesLT.Product GROUP BY ProductId`

No additional configuration is required on this tab.

On the **Sink** tab, including:

Sink Dataset	Create a new "Azure Data Explorer (Kusto)" dataset using the Data Explorer linked service and the "_product_waterline" table {e.g., "<UseCase>ded_product_waterline"}
Database and Table	Confirm values

No additional configuration is required on this tab.

On the **Mapping** tab, click the "**Import schemas**" button.

Publish changes and then click **Debug** and monitor result to confirm successful progress.

Activity 3: Capture Deletes

This activity will **capture data associated with pending delete operations** (using already-captured data in Data Explorer since the data is no longer available at the source).

Expand "**Move & Transform**" in the **Activities** bar, drag-and-drop a "**Copy data**" component into the activity window and then create a dependency from the "**Capture Pending**" component.

On the **Source** tab, including:

Source Dataset	Select the <UseCase>ded_product_pending dataset
Use Query	Select the **Query** radio button

Query	Paste the following parameterized KQL:
	`_product_pending`
	`| where _op == 'delete'`
	`| join _product on _id`
	`| project ProductId, ProductNumber, Name, ListPrice, ModifiedDate, _id, _dt, _op = _op1`

No additional configuration is required on this tab.

On the **Sink** tab, including:

Sink Dataset	Create an "**Azure Data Explorer (Kusto)**" dataset using the **Data Explorer** linked service and the "**_product**" table {e.g., "**<UseCase>ded_product**"}
Database and Table	Confirm values

No additional configuration is required on this tab.

On the **Mapping** tab, click the "**Import schemas**" button.

Click **Debug** and monitor result to confirm successful progress.

Activity 4: Localize Pending Inserts and Updates

This activity will localize "_pending" data for delete operations to the source for a SQL JOIN.

Expand "**Move & Transform**" in the **Activities** bar, drag-and-drop a "**Copy data**" component into the activity window and then create a dependency from the "**Capture Pending**" component.

On the **Source** tab, including:

Source Dataset	Select the <UseCase>ded_product_pending dataset
Use Query	Select the **Query** radio button

Query	Paste the following KQL:
	`_product_pending`
	`| where _op in ( 'insert', 'update' )`

No additional configuration is required on this tab.

On the **Sink** tab, including:

Sink Dataset	Create a new "**Azure SQL Database**" dataset using the **SQL** linked service and the "dbo.**Product_Pending**" table {e.g., "**<UseCase>sd_Product_Pending**"}
Pre-Copy Script	Paste the following T-SQL: `TRUNCATE TABLE [dbo].[Product_Pending]`

No additional configuration is required on this tab.

On the **Mapping** tab, click the "**Import schemas**" button.

Click **Debug** and monitor result to confirm successful progress.

Activity 5: Capture Inserts and Updates

This activity will capture data associated with pending insert and update operations (using data from the source system).

Expand "**Move & Transform**" in the **Activities** bar, drag-and-drop a "**Copy data**" component into the activity window and then create a dependency from the "**Capture Pending**" component.

On the **Source** tab, including:

Source Dataset	Select <UseCase>sd_None
Use Query	Select the **Query** radio button
Query	Paste the following T-SQL: `SELECT x.[ProductId], x.[ProductNumber], x.[Name], x.[ListPrice], x.[ModifiedDate], x.[ProductId] [_id], x.[ModifiedDate] [_dt], p.[_op] FROM [SalesLT].[Product] x INNER JOIN [dbo].[Product_Pending] p ON x.[ProductID] = p.[_id]`

No additional configuration is required on this tab.

On the **Sink** tab, including:

Sink Dataset	Select <UseCase>ded_Product
Database and Table	Confirm values

No additional configuration is required on this tab.

On the **Mapping** tab, click the "**Import schemas**" button.

Click **Debug** and monitor result to confirm successful progress.

Confirm Success

The first and most obvious confirmation is that your completed pipeline runs successfully.

Then, of course, confirm that data is correctly captured.

Confirmation 1, Record Count

Navigate to Synapse Studio, click the **Data** navigation icon, expand "**Data Explorer Databases**…" and your Data Explorer Pool. Right click on your Data Explorer Database {e.g., **<UseCase>ded**} and select "**New KQL script**" from the resulting drop-down menu.

Paste and Run the following logic to validate debug runs {i.e., with a non-zero record count}:

```
product | count
```

Confirmation 2, Delete Operation

Navigate to SQL, click the "**Query editor**…" navigation icon, and authenticate (adding IP to firewall if necessary). Execute the following logic:

```
SELECT MIN( [ProductID] ) FROM [SalesLT].[Product]
```

Use the returned value to execute the following logic:

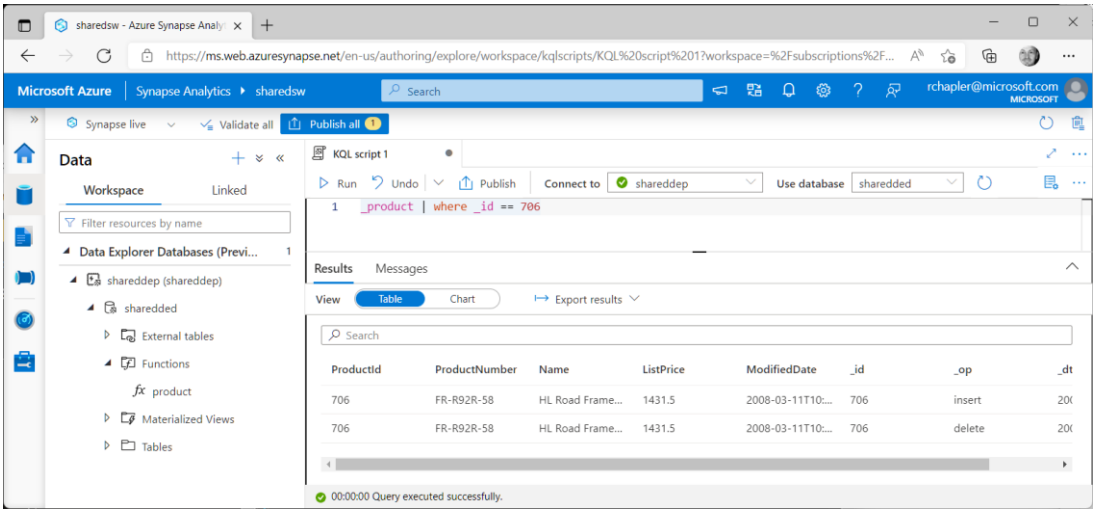
```
DELETE FROM [SalesLT].[Product] WHERE [ProductID] = {MIN ProductId value}
```

Navigate to the Product pipeline in Synapse, click **Debug** and monitor result to confirm successful progress. Execute the following logic to confirm that the "delete" record is included in the Product table:

```
_product | where _op == 'delete'
```

Execute the following logic to confirm that you can see the full history for the deleted item:

```
_product | where _id == {deleted ProductId value}
```


sharedsw - Azure Synapse Analy
https://ms.web.azuresynapse.net/en-us/authoring/explore/workspace/kqlscripts/KQL%20script%201?workspace=%2Fsubscriptions%2F...
Microsoft Azure
Synapse Analytics
sharedsw
Search
rchapler@microsoft.com
MICROSOFT
Synapse live
Validate all
Publish all 1
Data
Workspace
Linked
Filter resources by name
Data Explorer Databases (Previ... 1
shareddep (shareddep)
sharedded
External tables
Functions
product
Materialized Views
Tables
KQL script 1
Run
Undo
Publish
Connect to
shareddep
Use database
sharedded
1 _product | where _id == 706
Results
Messages
View
Table
Chart
Export results
Search
ProductId
ProductNumber
Name
ListPrice
ModifiedDate
_id
_op
_dt
706
FR-R92R-58
HL Road Frame...
1431.5
2008-03-11T10:...
706
insert
706
FR-R92R-58
HL Road Frame...
1431.5
2008-03-11T10:...
706
delete
00:00:00 Query executed successfully.

Confirmation 3, Insert Operation

Navigate to SQL, click the "**Query editor**…" navigation icon, and authenticate (adding IP to firewall if necessary). Execute the following logic:

```
INSERT INTO [SalesLT].[Product] ([Name], [ProductNumber], [Color], [StandardCost], [ListPrice],
[SellStartDate], [rowguid], [ModifiedDate])
VALUES ('New Product ABC','ABC-123','bronze',1000,1500,getdate(),newid(),getdate())
```

Navigate to the Product pipeline in Synapse, click **Debug** and monitor result to confirm successful progress. Execute the following logic to confirm that the "insert" record is included in the Product table:

```
_product | where Name contains 'New Product'
```

Confirmation 4, Update Operation

Navigate to SQL, click "**Query Editor**" in the navigation, and then login (whitelist your IP address, if required). Execute the following logic to update a record:

```
UPDATE SalesLT.Product SET ModifiedDate=GETDATE() WHERE ProductID=(SELECT MAX([ProductID]) FROM
[SalesLT].[Product])
```

Navigate to the Product pipeline in Synapse, click **Debug** and monitor result to confirm successful progress. Execute the following logic to confirm that the updated record is included in the Product table:

```
Product | where Name contains 'New Product'
```

Create Trigger

Establish a recurring schedule for your completed pipeline.

Navigate to the new pipeline in Synapse, and then click the "**Add trigger**" button and click "**New/Edit**" in the resulting drop-down menu.

On the resulting pop-out, click the "**Choose trigger**…" drop-down menu and select "**+ New**."

Complete the "**New trigger**" pop-out, including:

Type	Select "**Schedule**"
Start Date	Confirm default value (consider using round hour / minute to avoid odd timing)
Time Zone	Confirm default value
Recurrence	Choose appropriate recurrence

No additional configuration is required . Review remaining tabs and then click the **Commit** button.

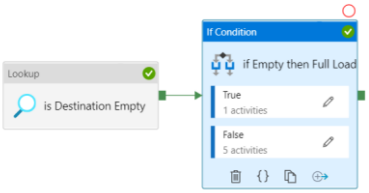
Special Circumstances

Not every source will align to the pattern described above; some of these unusual circumstances are described below.

Significant Initial Volume

Scenario: While implementing the incremental load pipeline, you recognize that processing the initial volume of source data through waterline, etc. processes is wasteful.

Proposed Solution: Consider amending your pipeline with the following precursor actions: 1) “sniff” of the destination {i.e., has any data previously been loaded}, and 2) conditional that with either initiate a simple load or an incremental load.



Lookup: Is Destination Empty?

Expand **General** in the **Activities** bar, then drag-and-drop a **Lookup** component into the activity window. On the **Settings** tab, including:

Source Dataset	Select the Data Explorer Dataset from the drop-down menu
First Row Only	Check
Query	Personalize and paste the following KQL: `_product_raw | count`

If Condition: If Empty, then Full Load

Expand “**Iteration & Conditionals**” in the **Activities** bar, then drag-and-drop an “**If Condition**” component into the activity window. Create a dependency from the “**Sniff Destination**” component. On the **Activities** tab, enter the following **Expression** value:

```
@equals( activity('Sniff Destination').output.firstRow.Count, 0 )
```

Click on the pencil icon in the “**If Condition**” component, **True** selection and paste the logic for Simple Load {i.e., copy initial batch of data}.

Click on the pencil icon in the “**If Condition**” component, **False** selection and paste the logic for Incremental Load.

Slow Source

Scenario: While implementing the incremental load pipeline, you observe that the source server is severely under-powered … running a simple waterline takes longer than a full trunc-and-load.

Proposed Solution #1: If you must evaluate delete, insert, and update events for all identifiers, consider starting the pipeline with trunc-and-load {i.e., dump and re-load of data in a destination table}. With data localized to Azure Data Explorer, all ensuing steps {e.g., waterline, pending changes, etc.} can be evaluated with functions.

Proposed Solution #2: If you only care about insert events (as in the case of a time series), consider starting the pipeline with an initial sniff of the destination table to find out the overall maximum waterline value, and then only pull those source records with a waterline value greater than that max.

Max Waterline

Expand **General** in the **Activities** bar, then drag-and-drop a **Lookup** component into the activity window. On the **Settings** tab, including:

Source Dataset	Select the Data Explorer Dataset from the drop-down menu
First Row Only	Check
Query	Personalize and paste the following KQL: `_product_raw | summarize theMax = max(modifieddate)`

Incremental Load

Expand "**Move & Transform**" in the **Activities** bar, drag-and-drop a "**Copy data**" component into the activity window and then create a dependency from the "**Sniff Destination**" component.

On the **Source** tab, including:

Source Dataset	Create a new "**Azure SQL Database**" dataset using the **SQL** linked service and the "**None**" table {e.g., "**<UseCase>sd_None**"}
Use Query	Select the **Query** radio button

Query	Paste the following dynamic expression:
	`@concat('SELECT * FROM [Product] WITH (NOLOCK)'`
	`, ' WHERE [ModifiedDate] > CAST('''`
	`, string(activity('Max Waterline').output.firstRow.theMax)`
	`, ''' AS DATETIME)')`

No additional configuration is required on this tab.

CHECKSUM” vs. Waterline

If your source data lacks a waterline column, you will need to use “checksum”-type logic that can evaluate the data in one or more columns, return a coded value representing that data and provide for comparison against data in future evaluations of the one or more columns.

For example, your Source Query in “Capture Waterline” might look something like:

```
SELECT CONCAT(x.COL1, '') AS ID, STANDARD_HASH( x.COL2 || x. COL3 || x. COL4, 'SHA256' ) AS CS
FROM SCHEMA.TABLE x
```

This change will need to be considered for all related functions, etc.

Data Flow

Follow these instructions to demonstrate incremental load using Synapse Data Flows and Delta Lake.

Prerequisites

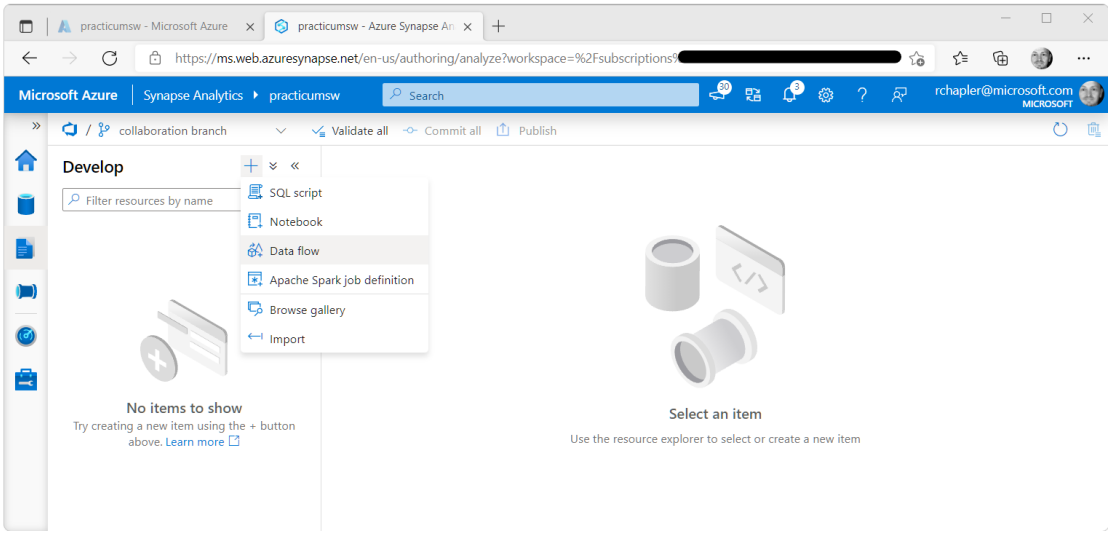
This solution requires the following resources:

- Data Lake
- SQL (with sample database)
- Synapse (with linked service for SQL and Apache Spark Pool)

Note: These instructions also apply (with minor differences) to Azure Data Factory.

Initial Load

This section describes how to load existing source data into Delta Lake (as a primer for Incremental Load).



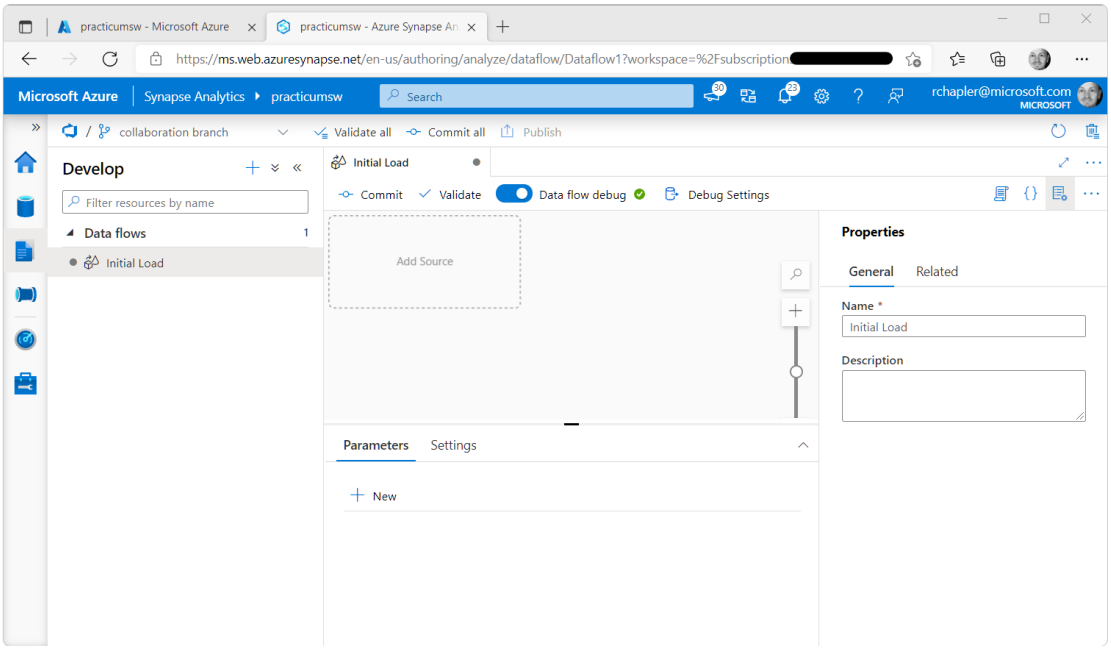
Navigate to your Synapse Analytics workspace and then click the **Open** link on the "**Open Synapse Studio**" rectangle.

Create Data Flow

Click the **Develop** icon in the navigation pane.

Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

Select "**Data flow**" from the resulting drop-down menu.

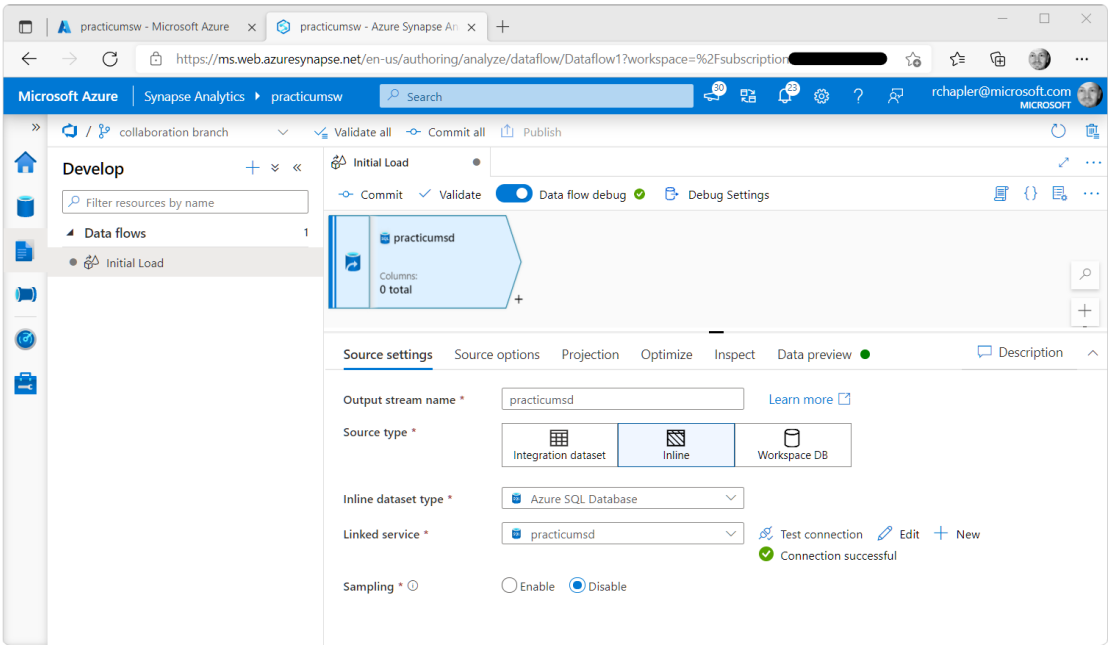


Complete the **Properties** pop-out form.

Activate "**Data flow debug**" to avoid wait time later in the process.

Source (sample database)

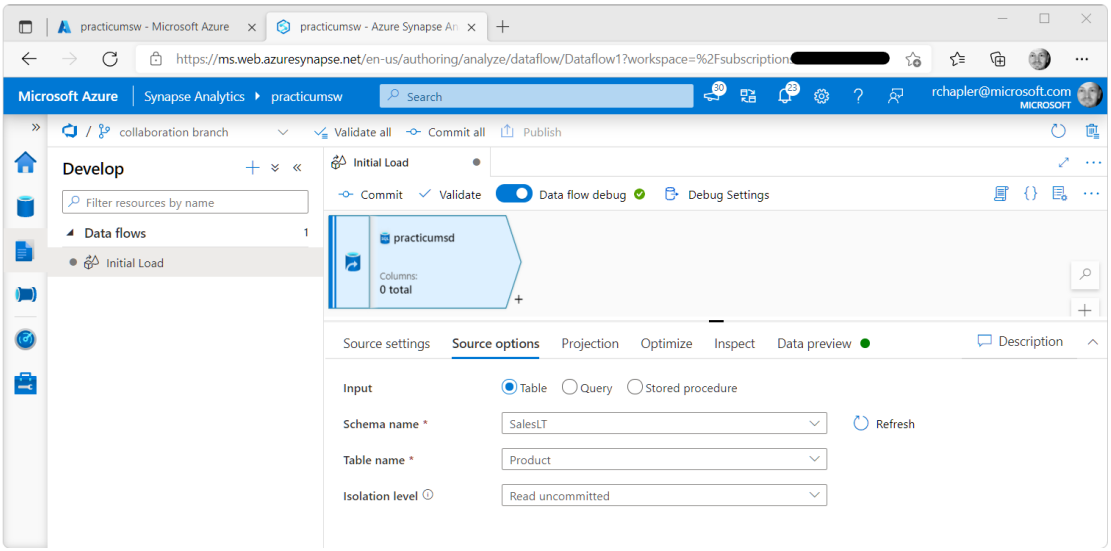
Click on "**Add Source**" in the dashed rectangle.



Complete the "**Source settings**" tab, including:

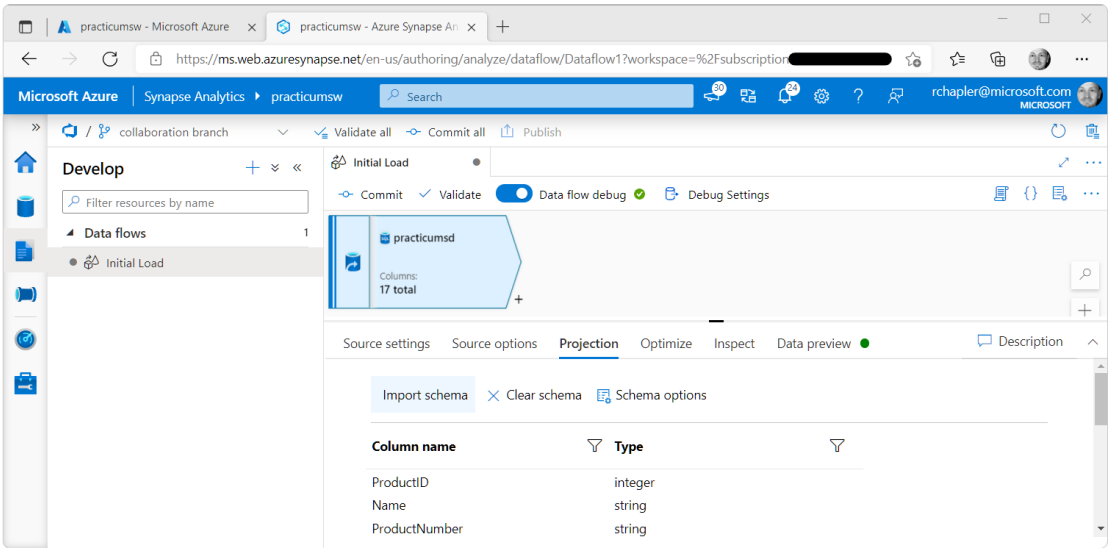
Source Type	Select **Inline**
Inline Dataset Type	Select "**Azure SQL Database**"
Linked Service	Select **<UseCase>sd**
Sampling	Confirm default setting, **Disable**

Click "**Test connection**" to confirm successful configuration.

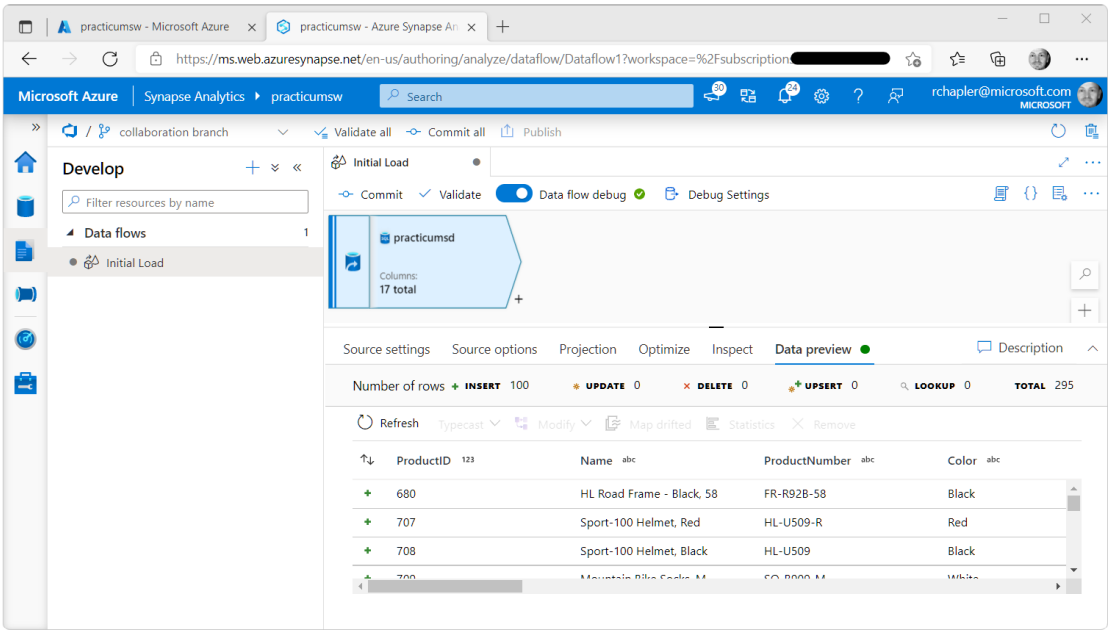


Complete the "**Source options**" tab, including:

Input	Confirm default setting, **Table**
Schema Name	Enter SalesLT
Table Name	Enter Product



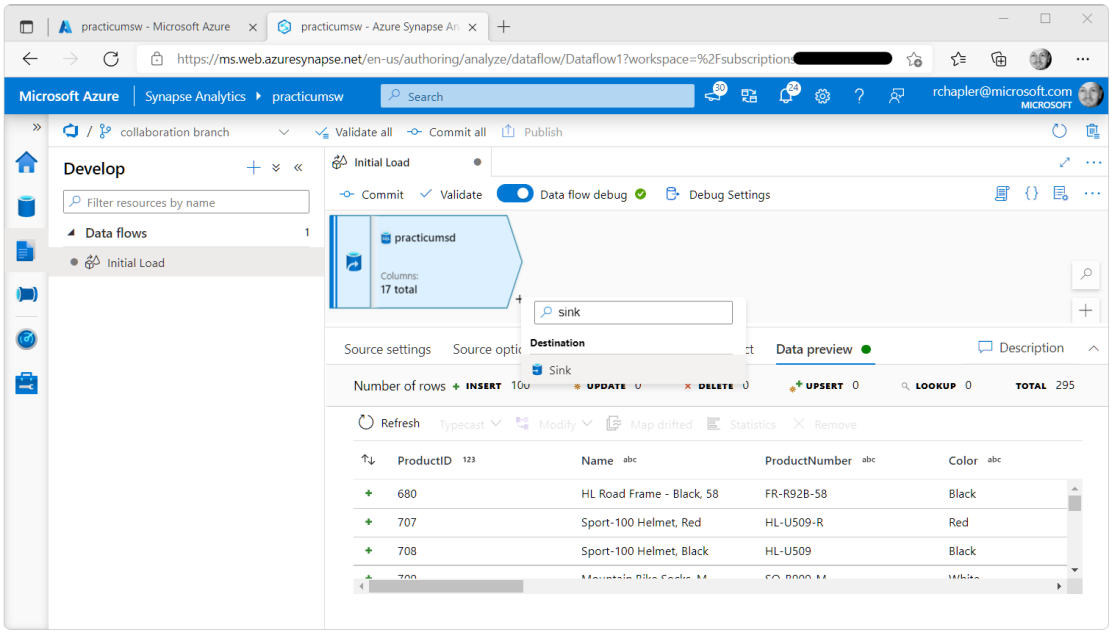
On the **Projection** tab, click "**Import schema**" and then **Import** on the resulting pop-out.



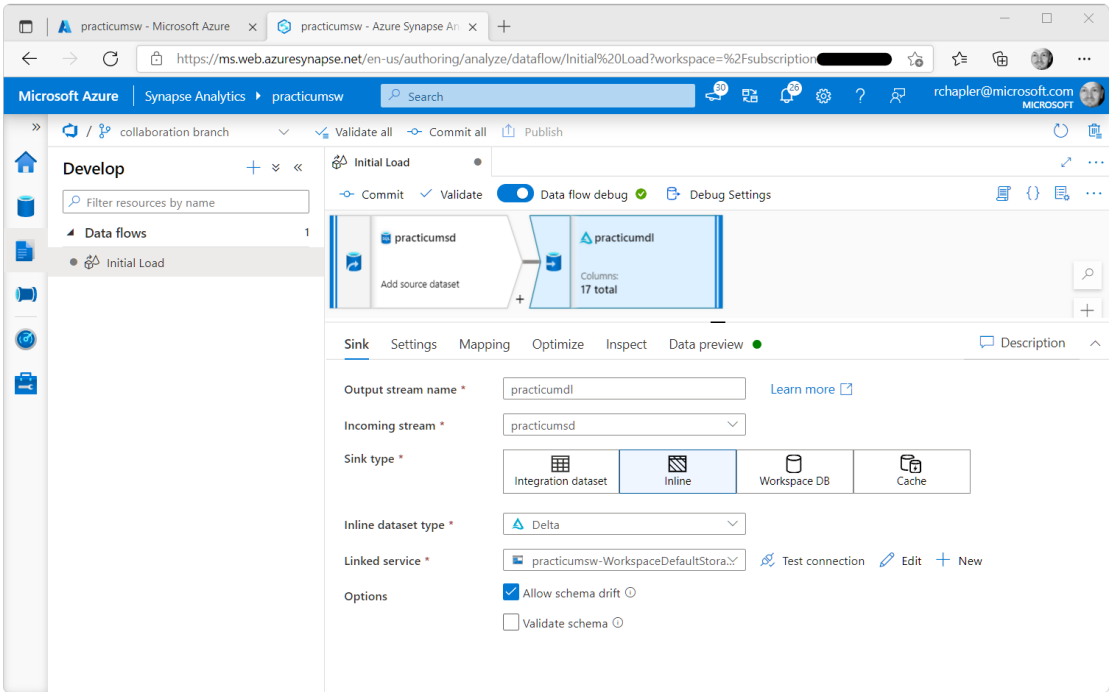
Navigate to the "**Data preview**" tab and then click **Refresh**.

Click the **Commit** button.

Sink (delta lake)

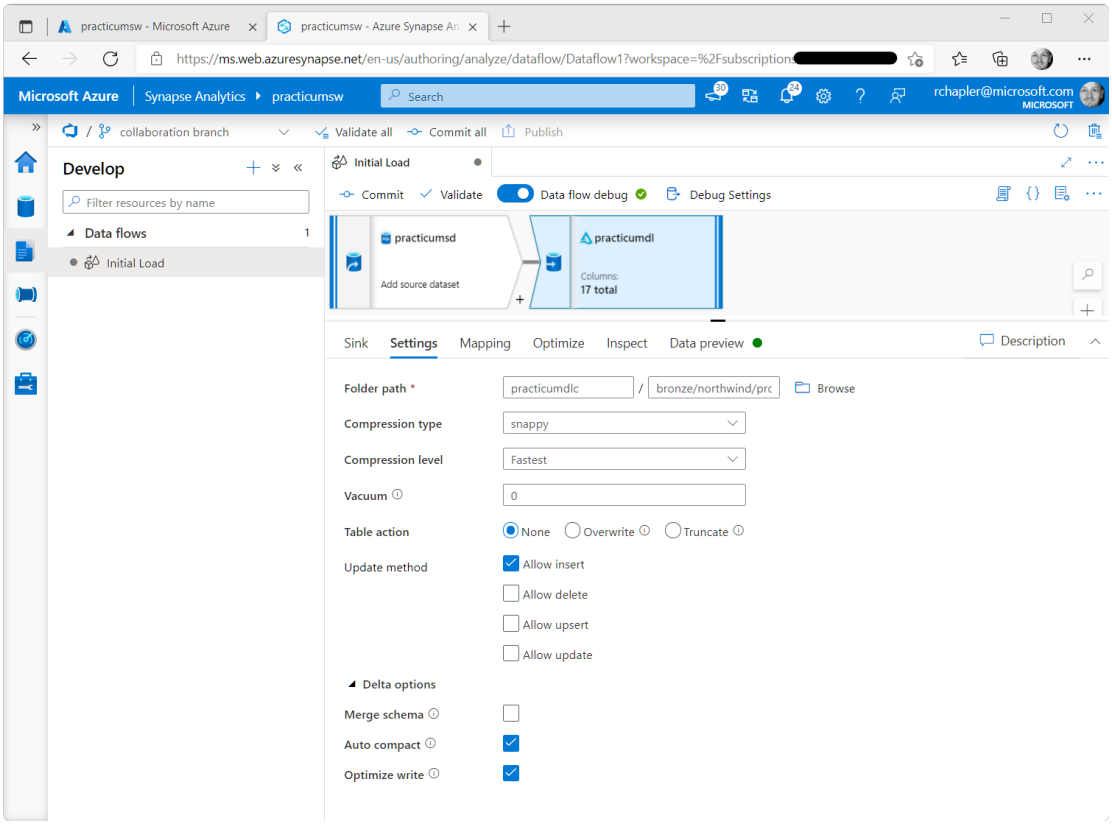


Click the **+** in the bottom right of the Data Lake source, and then search for and select **Sink** from the resulting pop-up list.



On the **Sink** tab, including:

Incoming Stream	Confirm default, **<UseCase>sd**
Sink Type	Select **Inline**
Linked Service	Select <UseCase>sw-WorkspaceDefaultStorage
Options	Confirm that "**Allow schema drift**" is checked

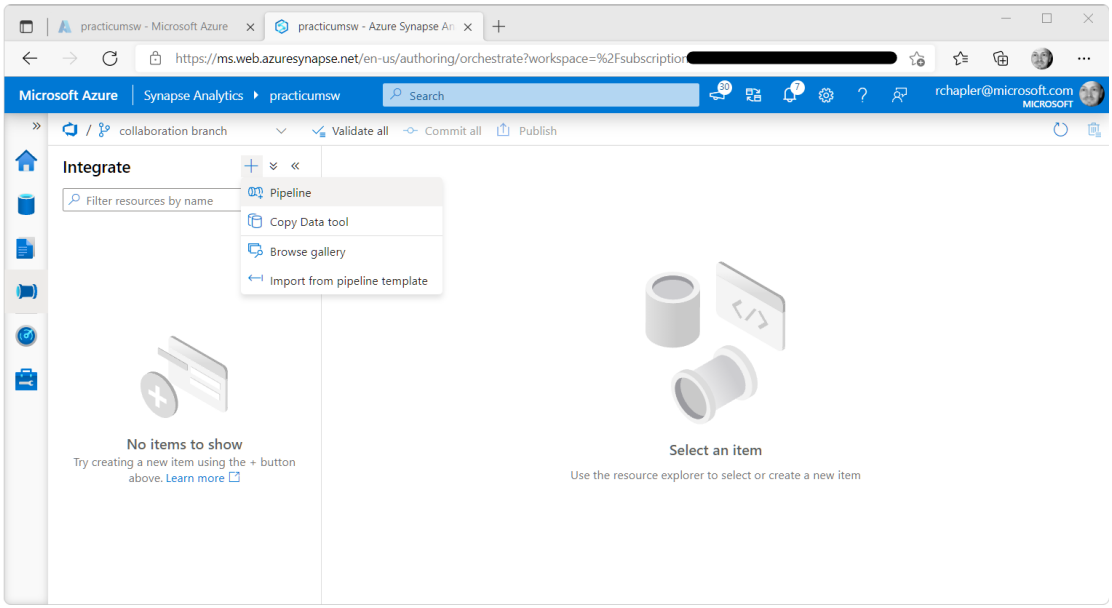


On the **Settings** tab, including:

File System	Enter <UseCase>dlc
Folder Path	Enter "bronze/northwind/product"
Compression Type	Select **snappy**
Compression Level	Select **Fastest**
Auto Compact	Checked
Optimize Write	Checked

Click the **Commit** button.

Create Pipeline

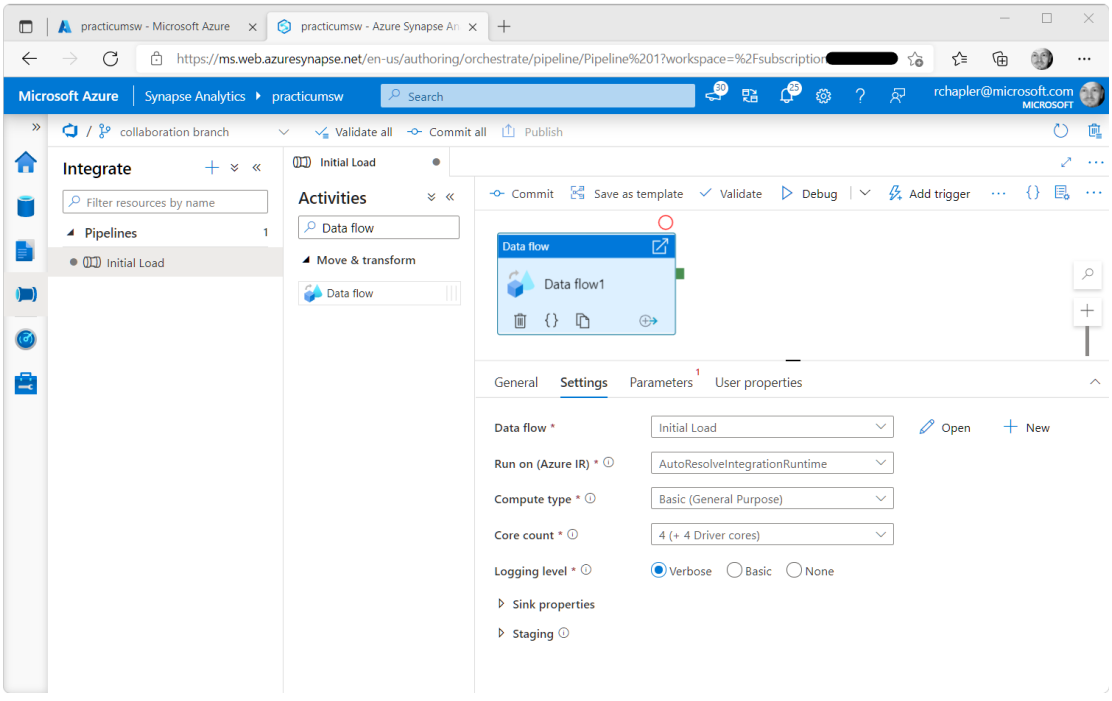


Click the **Integrate** icon in the navigation pane.

Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

Select "**Pipeline**" from the resulting drop-down menu.

On the **Properties** pop-out, enter Name, "**Initial Load**."



Search for, and then drag-and-drop a “**Data flow**” component into the activity window.

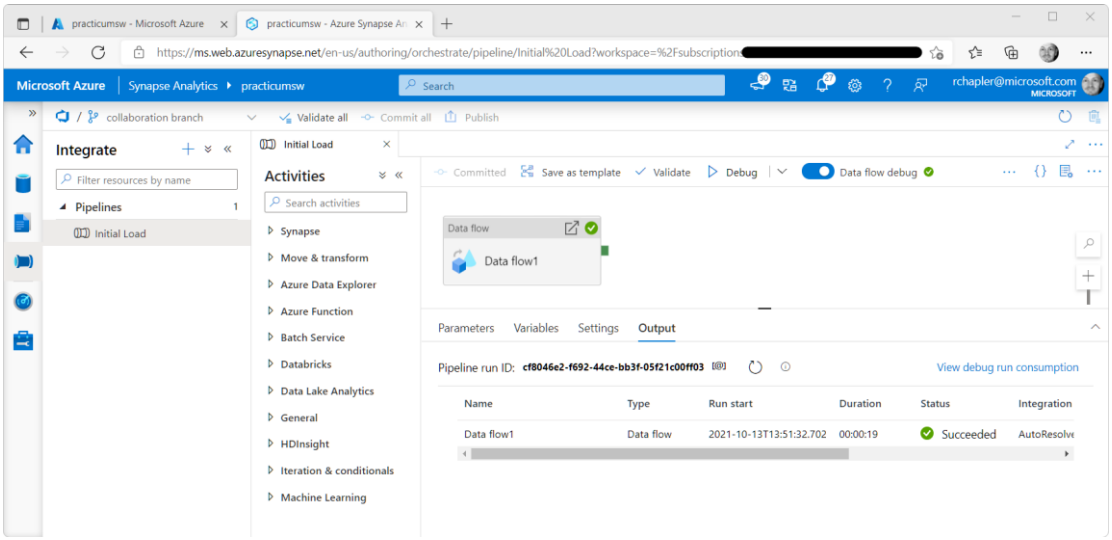
On the **Settings** tab of the “**Data flow**…” component, select your data flow.

Click the **Commit** button.

Confirm Success

Debug Pipeline / Data Flow

Click **Debug** and confirm successful execution.



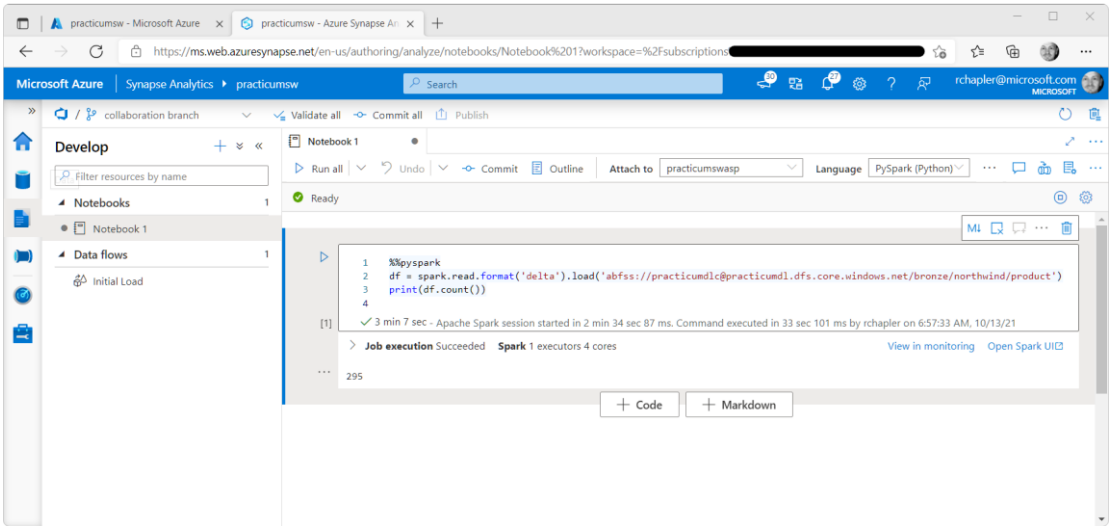
Query Delta Lake

Navigate to Synapse, and then click the **Develop** icon in the navigation pane.

Click the **+** icon just above and to the right of the “**Filter resources by name**” input.

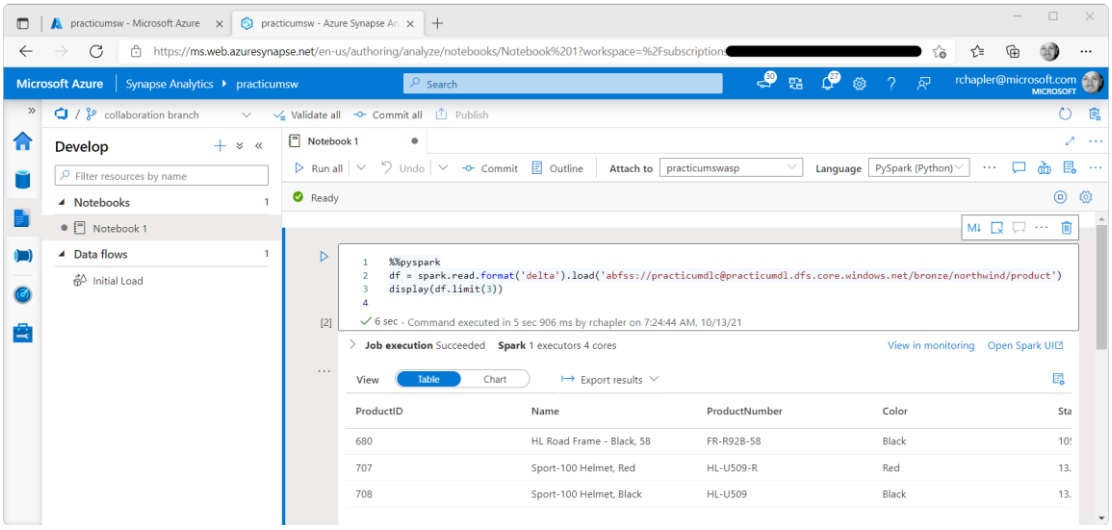
Select “**Notebook**” from the resulting drop-down menu.

In your new notebook, select your Apache Spark pool (if required) from the “**Attach to**” drop-down menu.



Run the following code to produce a count of records in the new Delta Lake:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/bronze/northwind/
product')
print(df.count())
```



Run the following code to display a few records in the new Delta Lake:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/bronze/northwind/
product')
display(df.limit(3))
```

Incremental Load

This section describes how to provide for recurring capture of changes to source data.

Create Data Flow

Click the **Develop** icon in the navigation pane.

Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

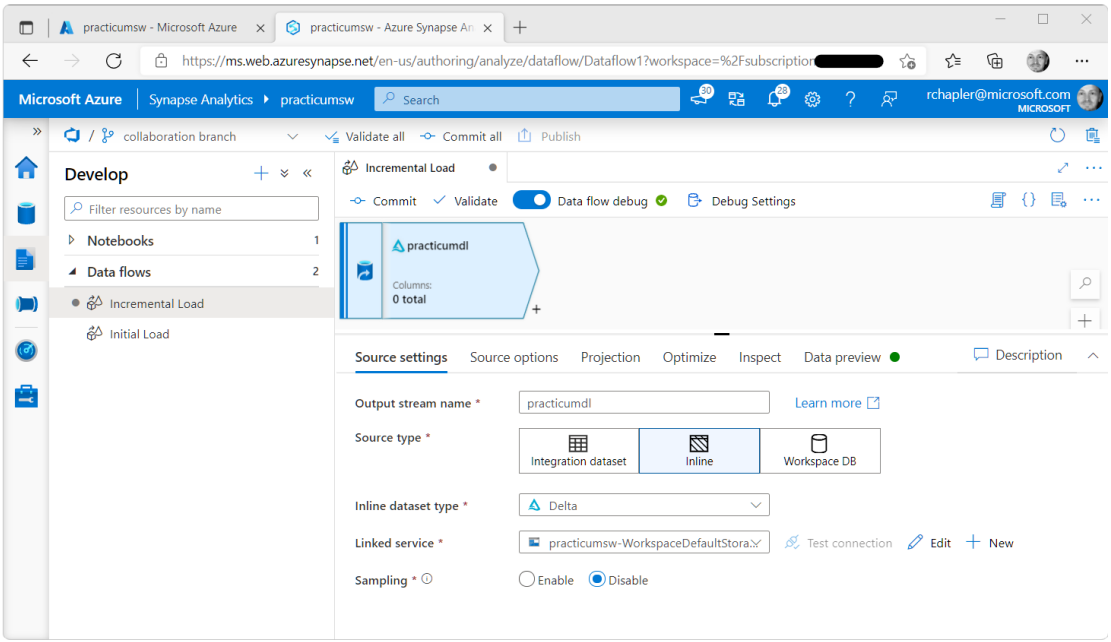
Select "**Data flow**" from the resulting drop-down menu.

Name your data flow "**Incremental Load**"

Source (delta lake)

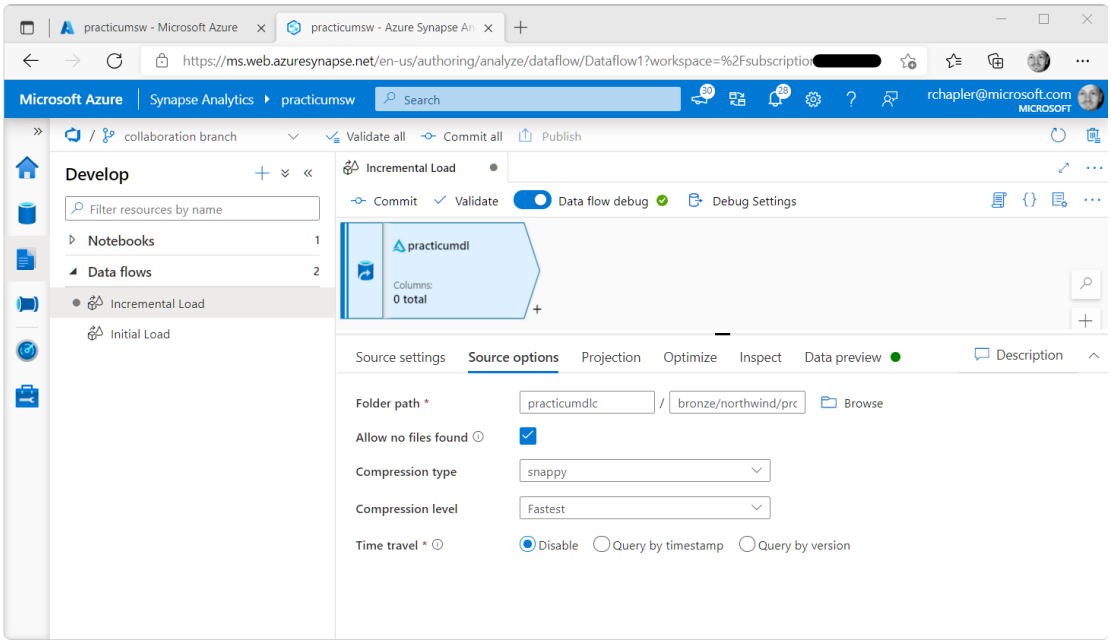
In this step, we will add the Delta Lake (created in the previous section) as a source.

Click on "**Add Source**" in the dashed rectangle.



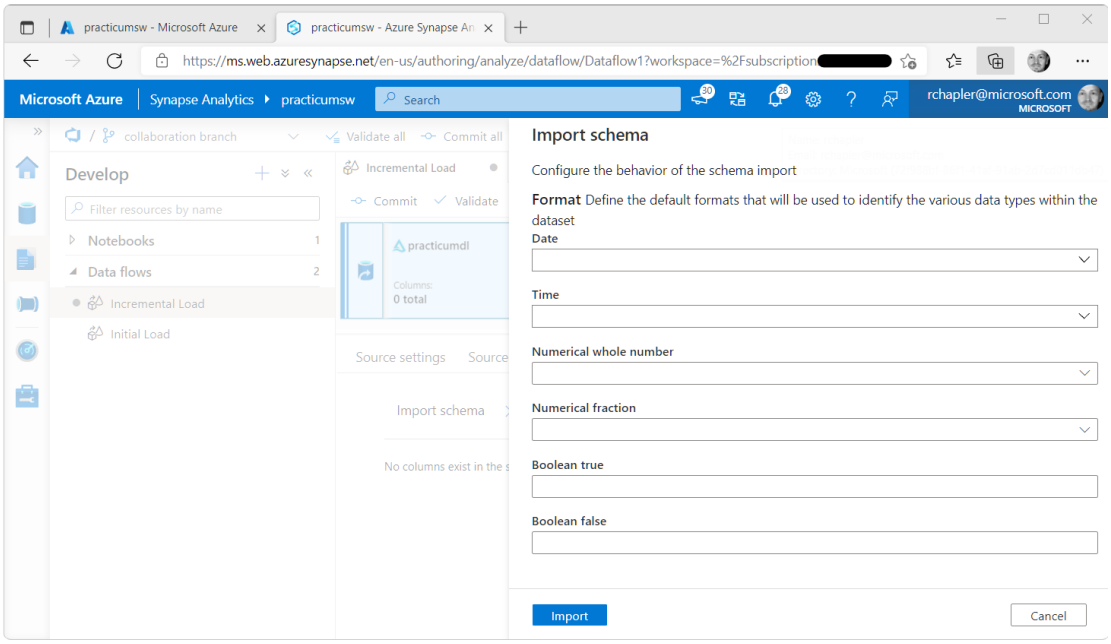
Complete the "**Source settings**" tab, including:

Source Type	Select **Inline**
Inline Dataset Type	Select **Delta**
Linked Service	Select **<UseCase>sw-WorkspaceDefaultStorage**
Sampling	Confirm default setting, **Disable**



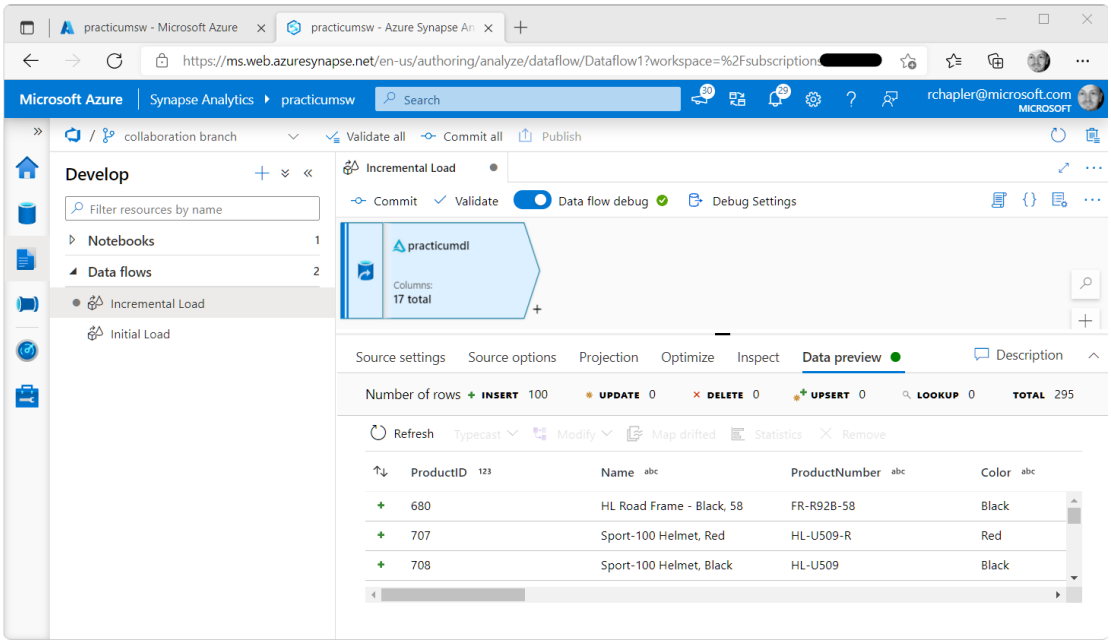
Complete the "**Source options**" tab, including:

File System	Enter <UseCase>dlc
Folder Path	Enter "bronze/northwind/product"
Allow No Files Found	Checked
Compression Type	Select **snappy**
Compression Level	Select **Fastest**
Time Travel	Confirm default, **Disable**



On the **Projection** tab, click "**Import schema**" and then **Import** on the resulting pop-out.

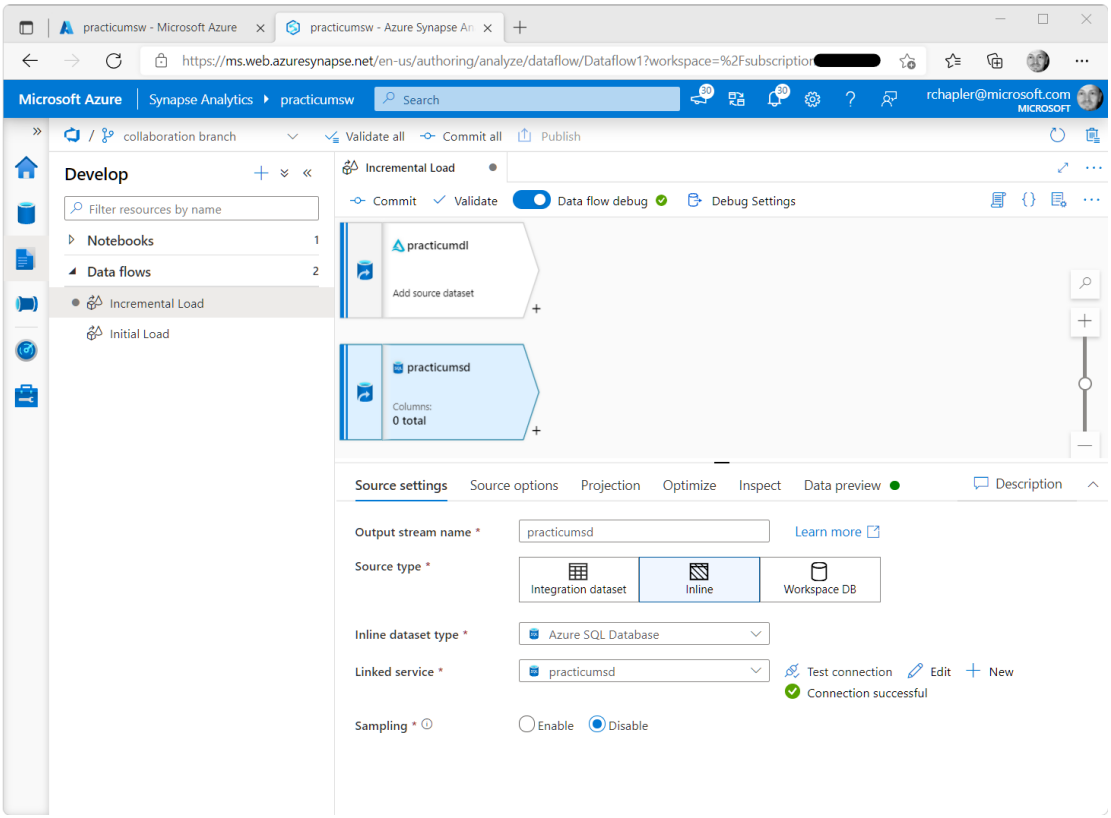
Navigate to the "**Data preview**" tab and then click **Refresh**.



Click the **Commit** button.

Source (sample database for this exercise)

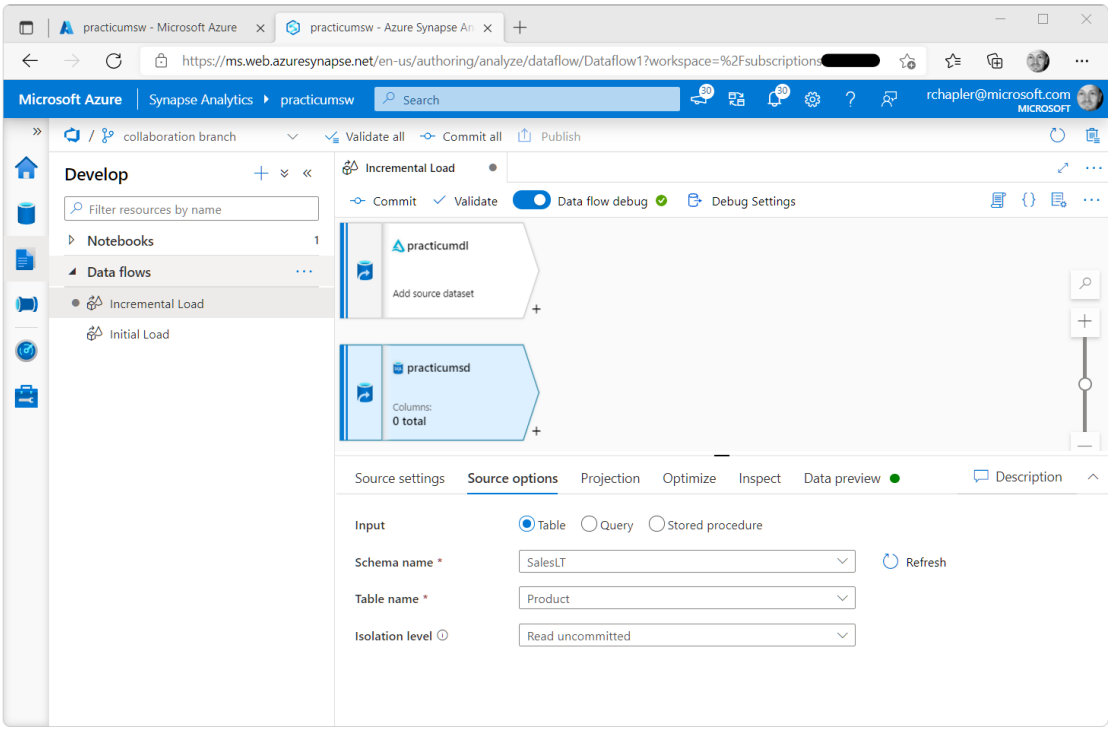
In this step, we will add the SQL Northwind sample database (our primary source for this exercise). Click on "**Add Source**" in the dashed rectangle.



Complete the "**Source settings**" tab, including:

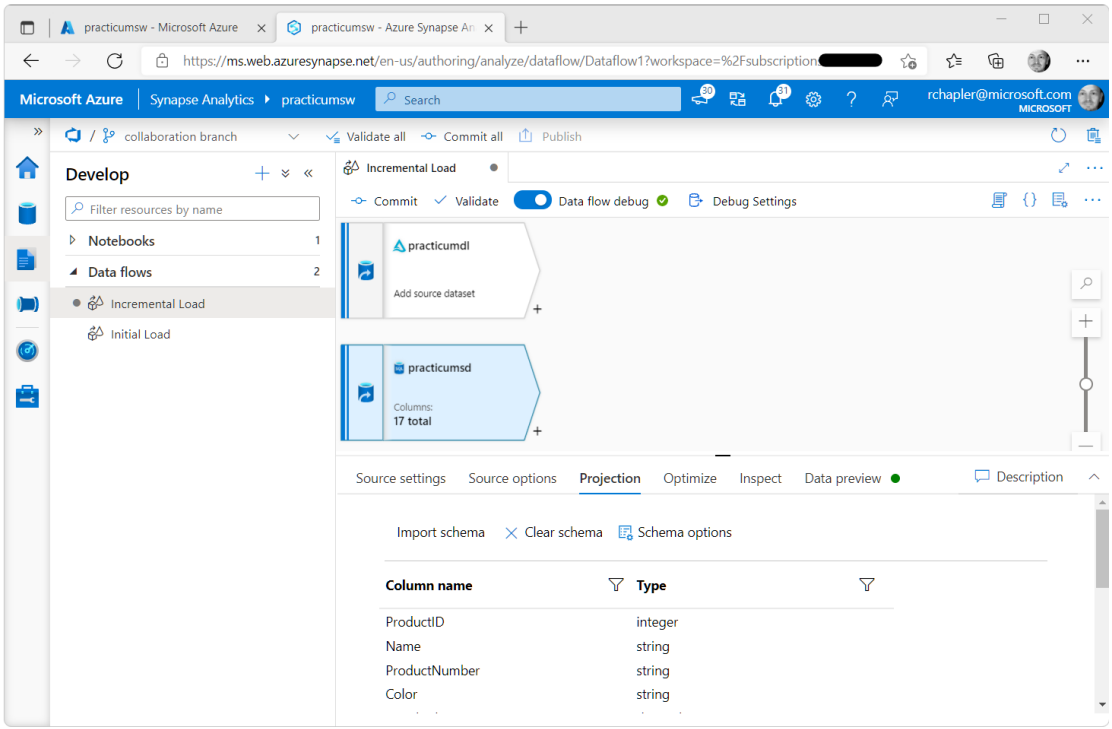
Source Type	Select **Inline**
Inline Dataset Type	Select "**Azure SQL Database**"
Linked Service	Select **<UseCase>sd**
Sampling	Confirm default setting, **Disable**

Click "**Test connection**" to confirm successful configuration.

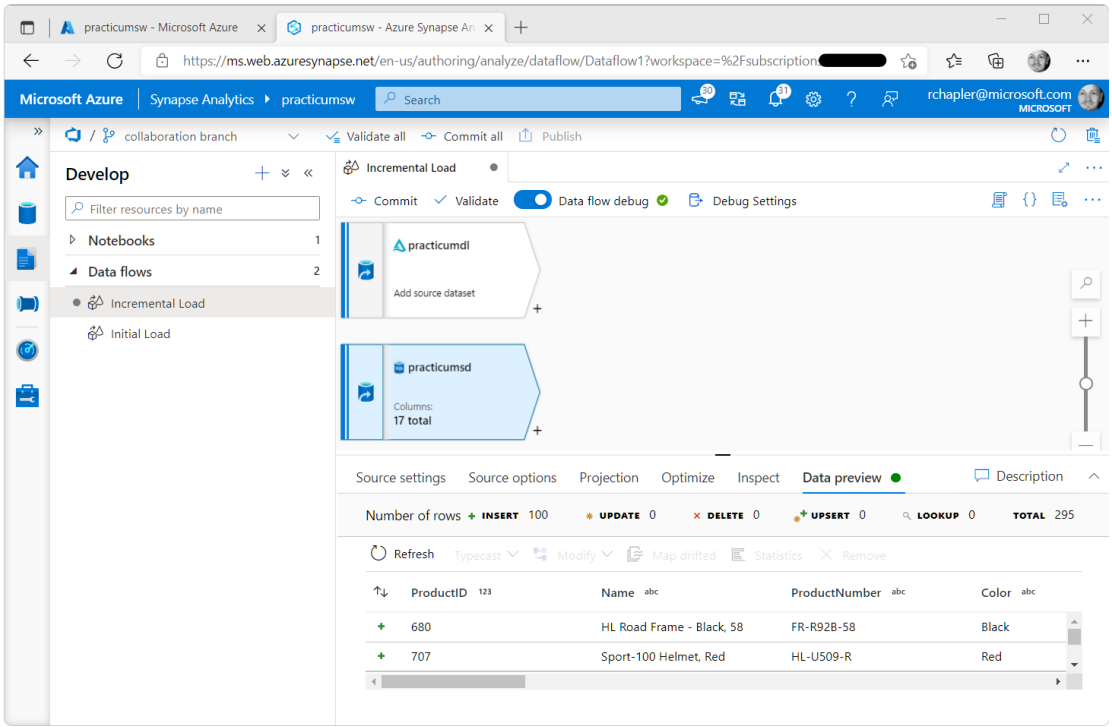


Complete the "**Source options**" tab, including:

Input	Confirm default setting, **Table**
Schema Name	Enter SalesLT
Table Name	Enter Product



On the **Projection** tab, click "**Import schema**" and then **Import** on the resulting pop-out.

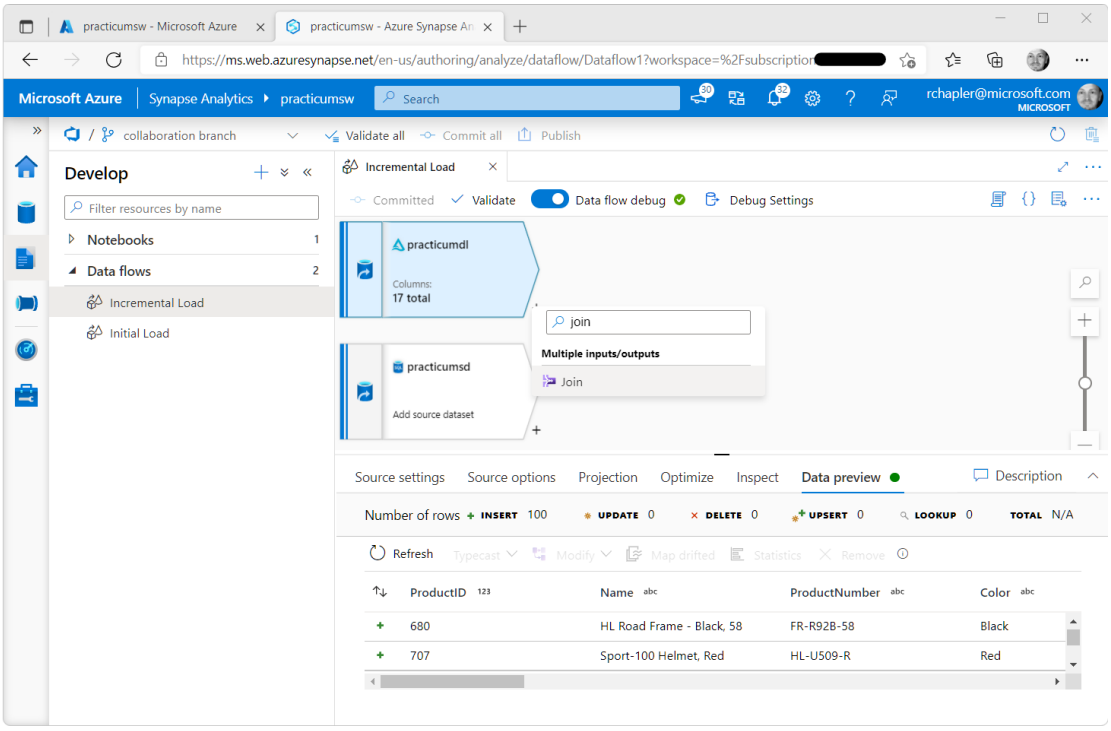


Navigate to the “**Data preview**” tab and then click **Refresh**.

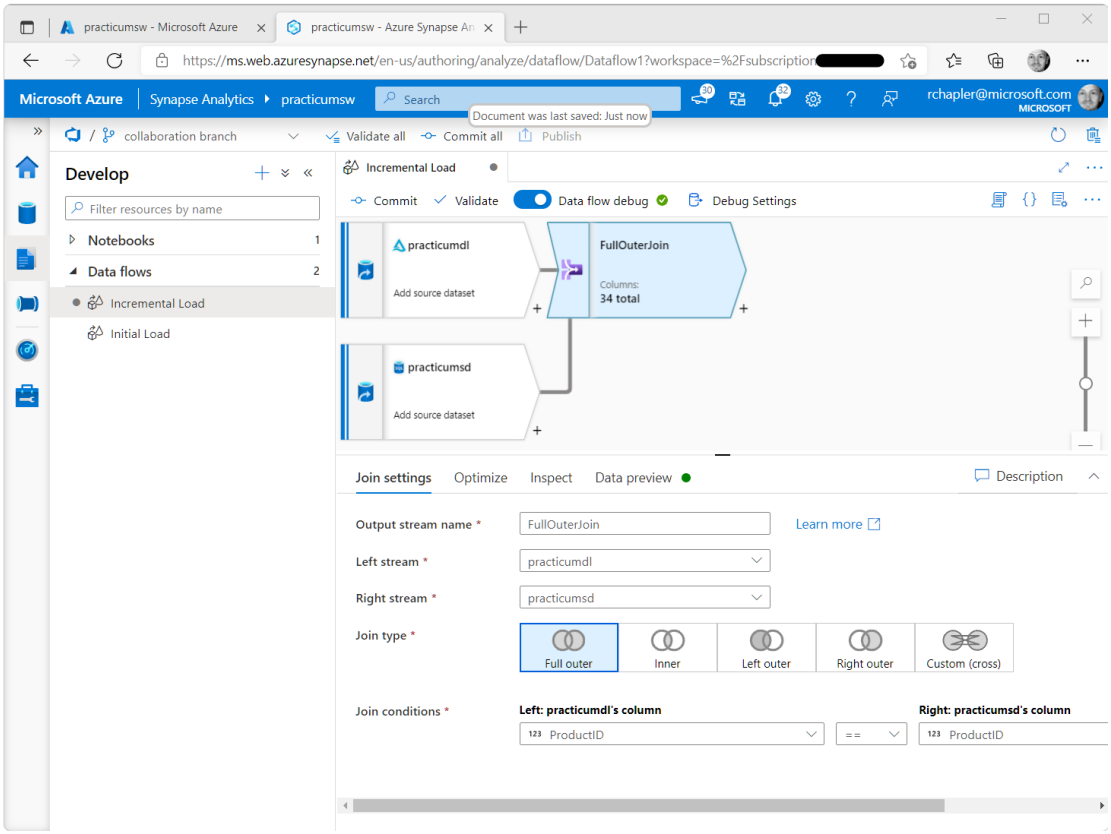
Click the **Commit** button.

Join Sources

In this step, we will perform a full outer join between source datasets.



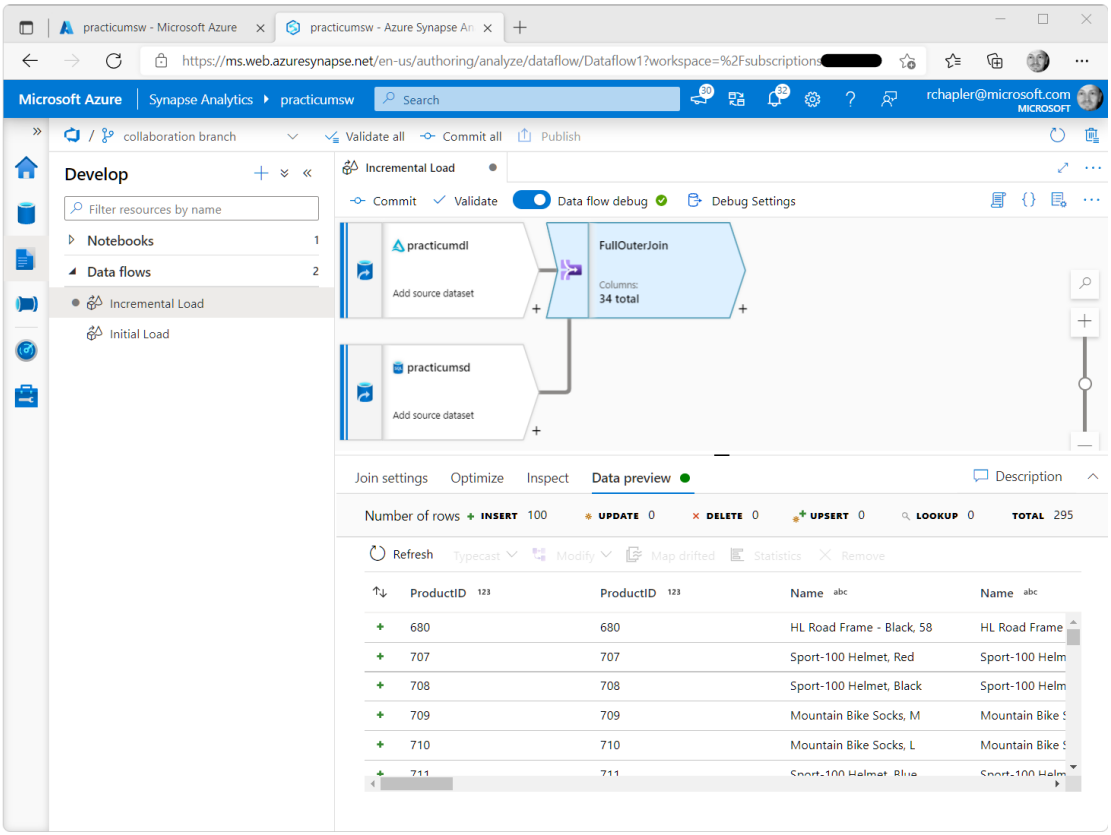
Click the **+** in the bottom right of the SQL source, and then select **Join** from the resulting pop-up list.



Complete the "**Join settings**" tab, including:

Left Stream	Select **<UseCase>sd**
Right Stream	Select **AggregateWaterline**
Join Type	Select "**Full outer**"
Join Conditions	Select **ProductID** for both Left… and Right… column

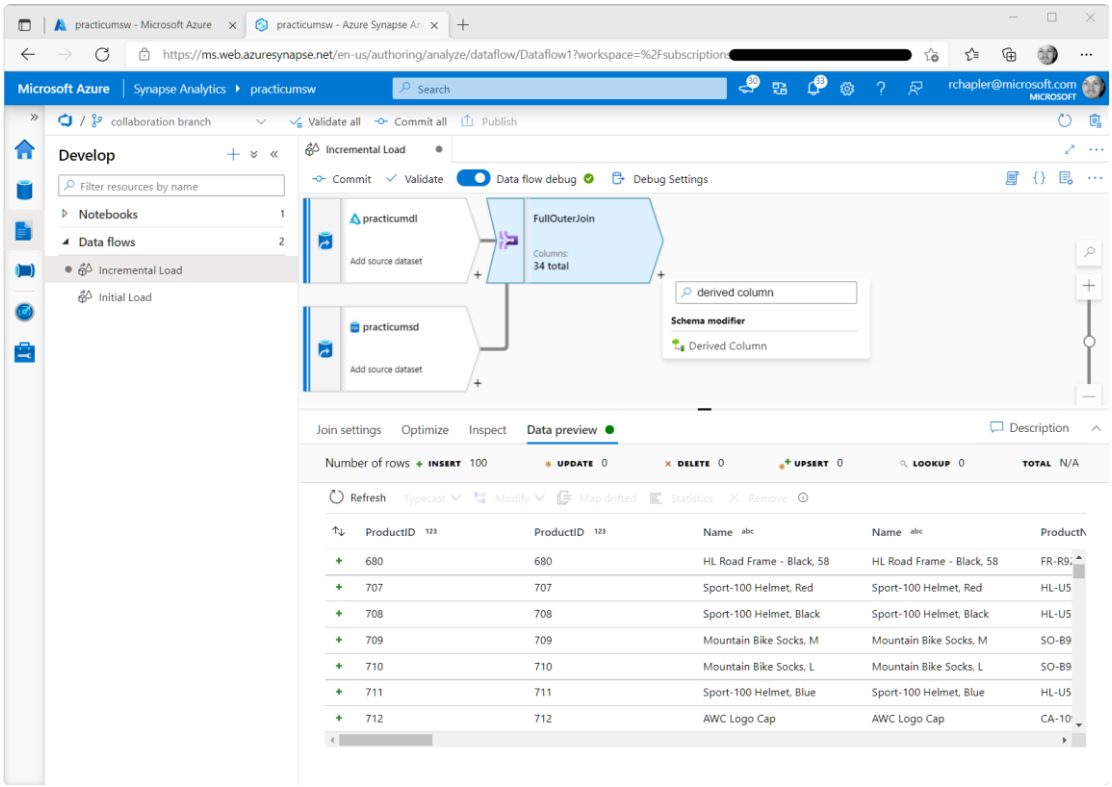
Navigate to the "**Data preview**" tab and then click **Refresh**.



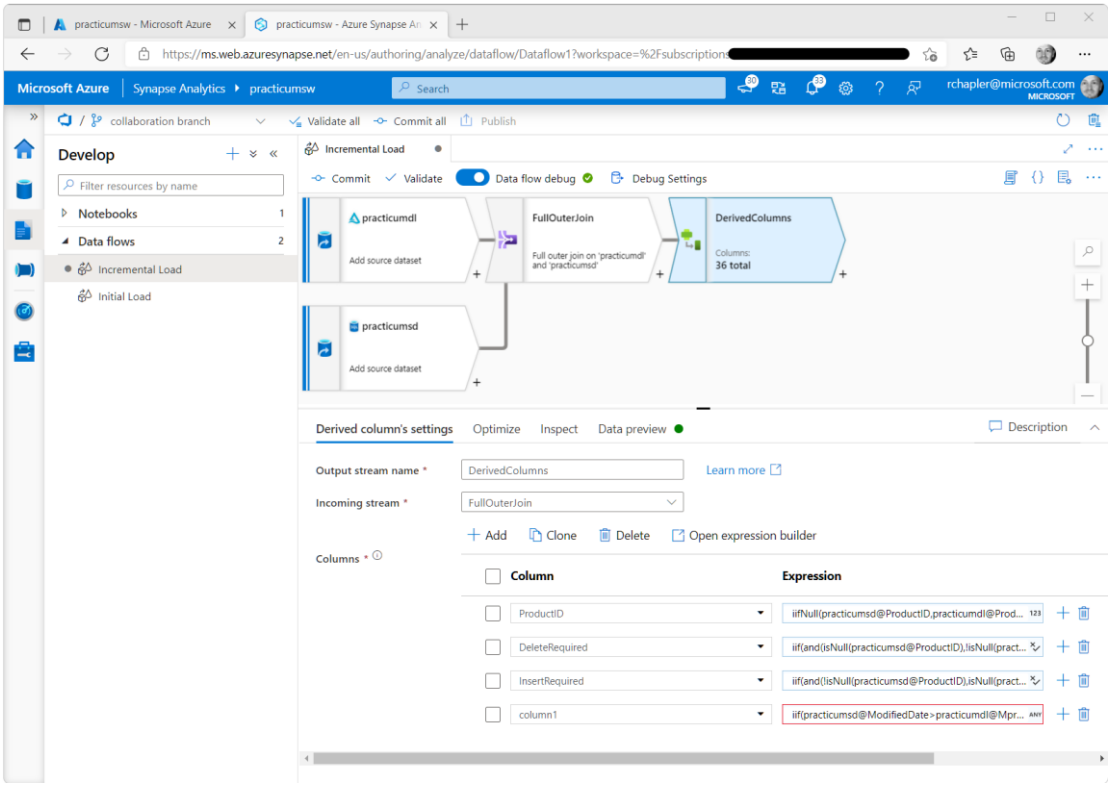
Click the **Commit** button.

Derive Columns

In this step, we will add columns designed to characterize required Delta Lake changes.



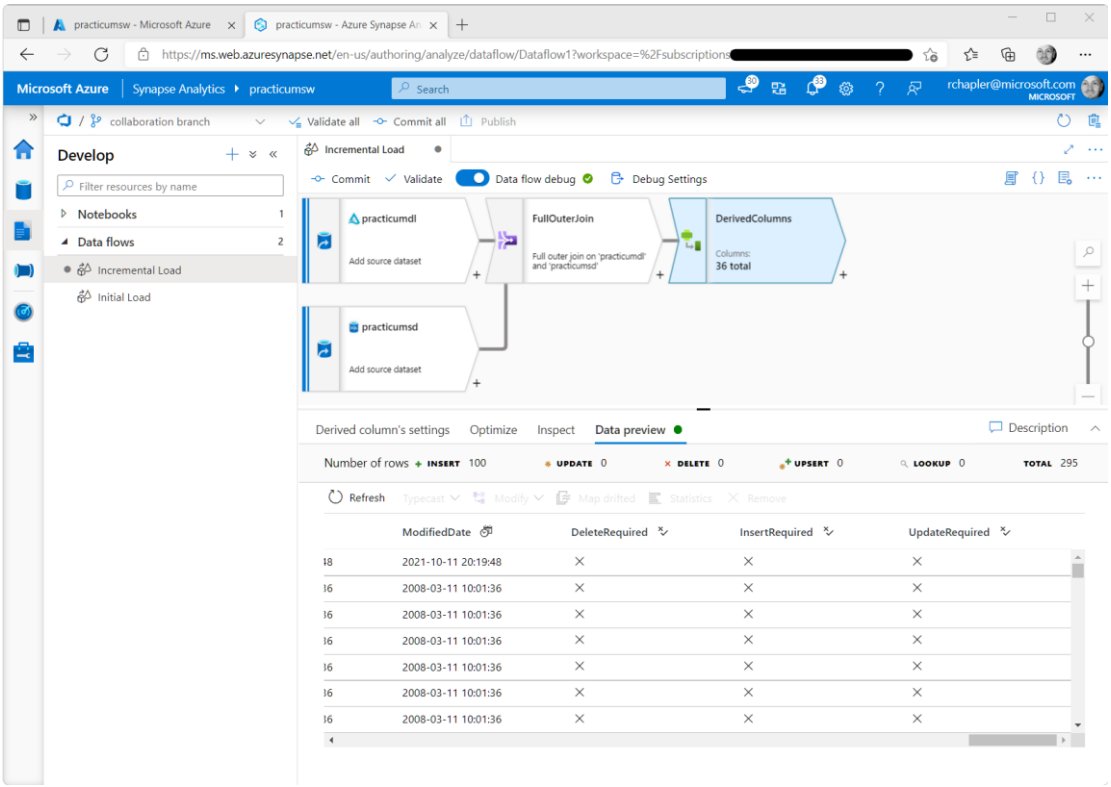
Click the **+** in the bottom right of the SQL source, and then select **Derived Column** from the resulting pop-up list.



Complete the "**Derived column's settings**" tab, including:

Incoming Stream	Confirm default selection
Columns	ProductID iifNull(<UseCase>sd@ProductID,<UseCase>dl@ProductID) DeleteRequired iif(and(isNull(<UseCase>sd@ProductID),!isNull(<UseCase>dl@ProductID)),true(),false()) InsertRequired iif(and(!isNull(<UseCase>sd@ProductID),isNull(<UseCase>dl@ProductID)),true(),false()) UpdateRequired iif(<UseCase>sd@ModifiedDate><UseCase>dl@ModifiedDate,true(),false())

Navigate to the "**Data preview**" tab and then click **Refresh**.

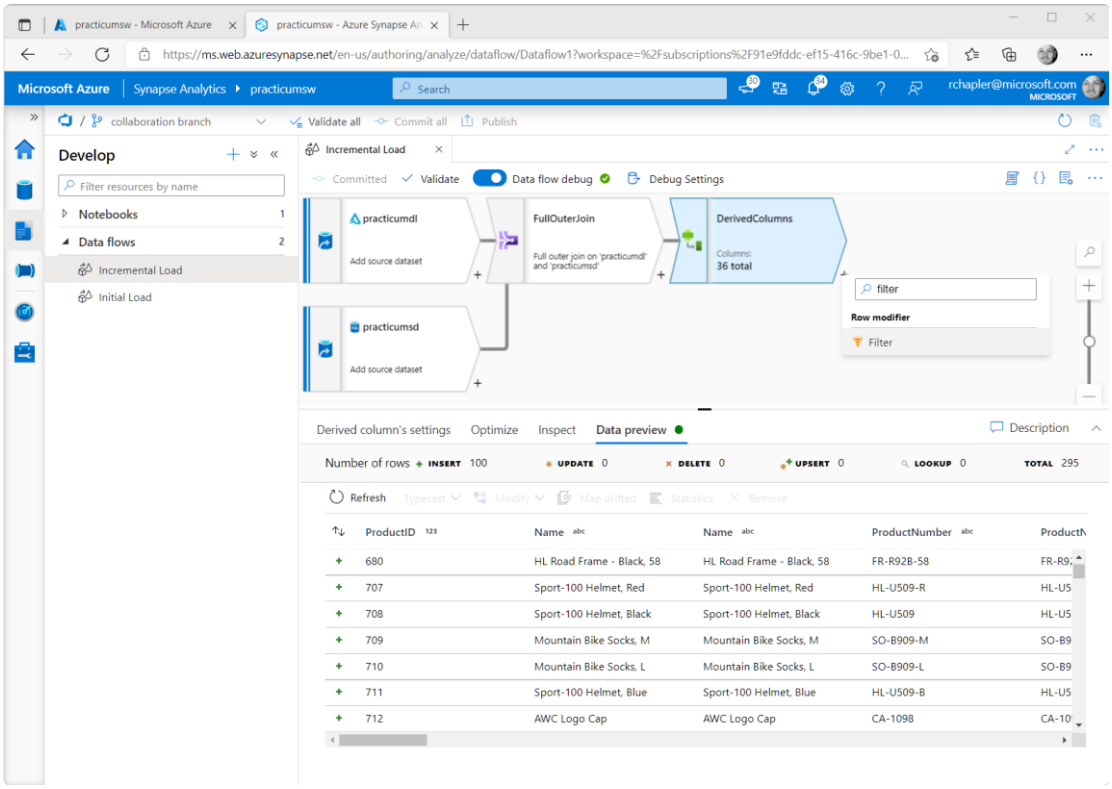


Click the **Commit** button.

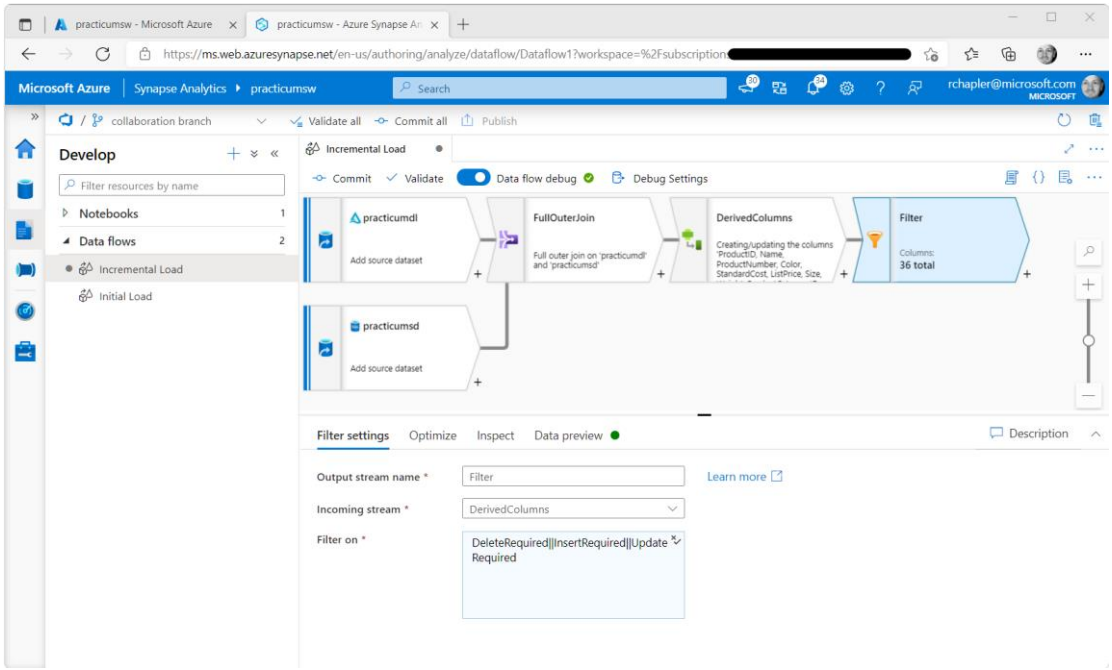
Note: I included this step to provide a straightforward way to preview and provide for the delete, insert and update of data… arguably, this step could be consolidated into Filter or AlterRows.

Filter Non-Actionable

In this step, we will remove rows that have no required action.



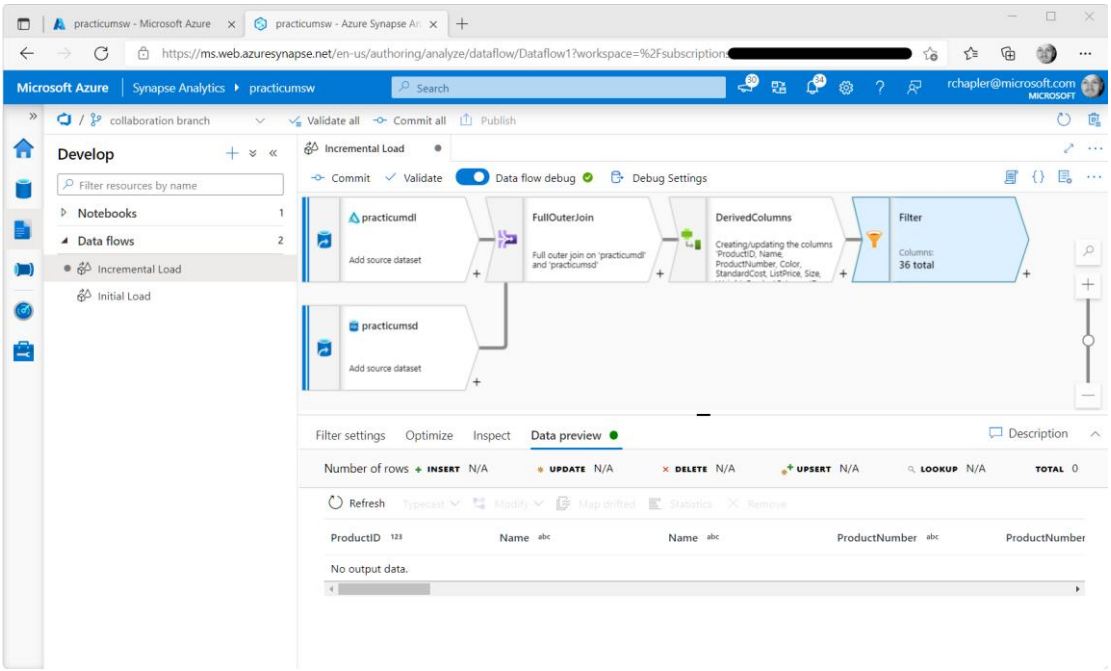
Click the **+** in the bottom right of the SQL source, and then select **Filter** from the resulting pop-up list.



Complete the “**Filter settings**” tab, including:

Incoming Stream	Confirm default selection
Filter On	DeleteRequired||InsertRequired||UpdateRequired

Navigate to the “**Data preview**” tab and then click **Refresh**.



Click the **Commit** button.

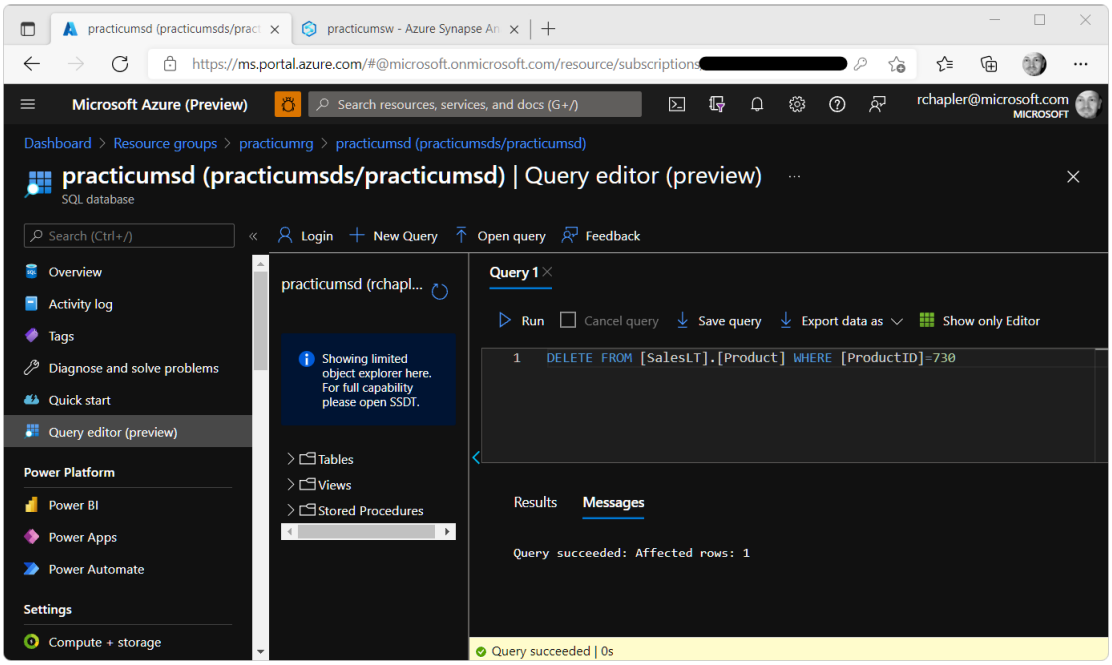
Note: Because we are doing an incremental load immediately after an initial load, on a sample database which doesn’t have unexpected change, we see no current results in the Data Preview.

Change Examples

To make the remaining steps a bit more interesting, we will force changes to the sample database.

Navigate to your Azure SQL Database, click the “**Query editor (preview)**” link in the navigation, and login.

Delete Example

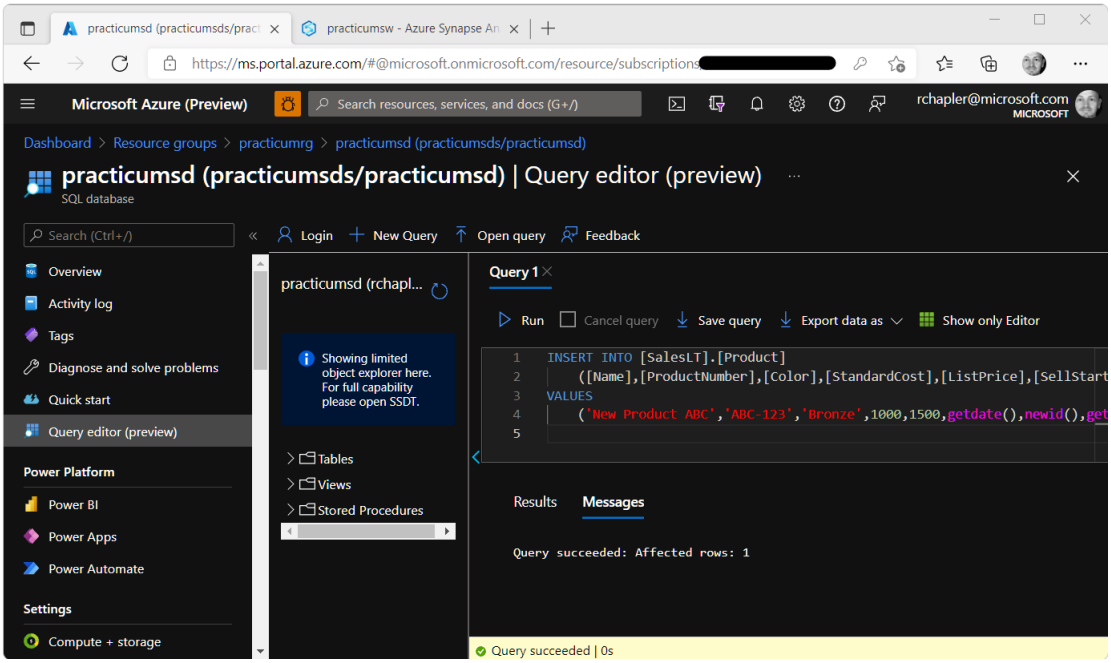


Execute the following T-SQL:

```
DELETE FROM [SalesLT].[Product] WHERE [ProductID]=730
```

Note: I chose 730 because it did not trigger constraints; any row deletion will work.

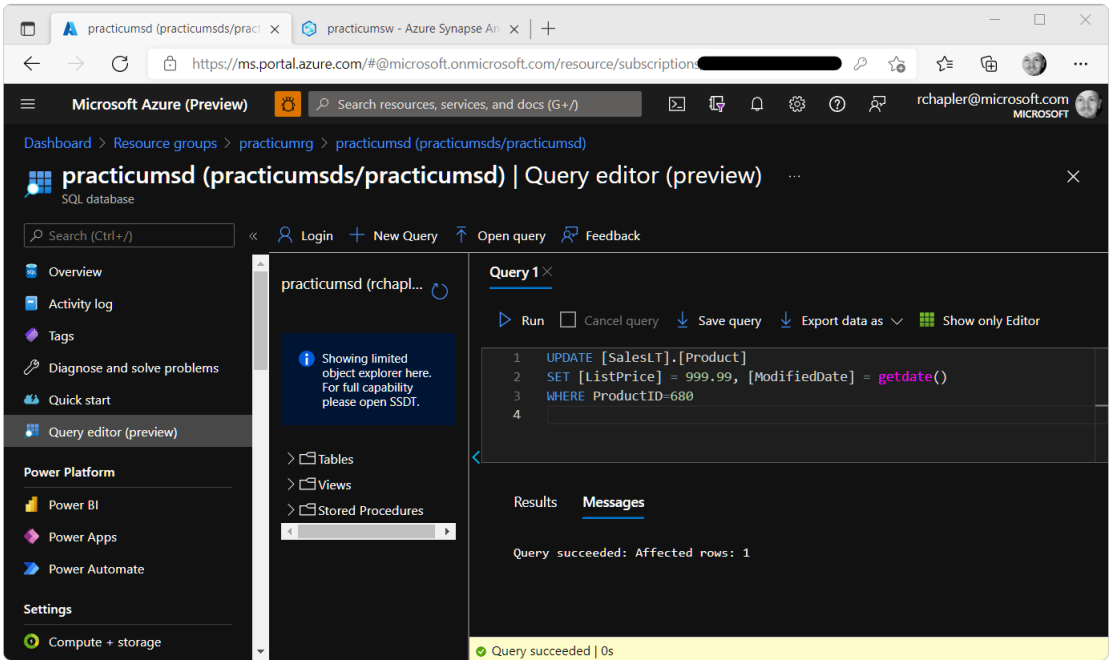
Insert Example



Execute the following T-SQL:

```
INSERT INTO [SalesLT].[Product]
 ([Name],[ProductNumber],[Color],[StandardCost],[ListPrice],[SellStartDate],[rowguid],[ModifiedDate])
VALUES
 ('New Product ABC','ABC-123','bronze',1000,1500,getdate(),newid(),getdate())
```

Update Example

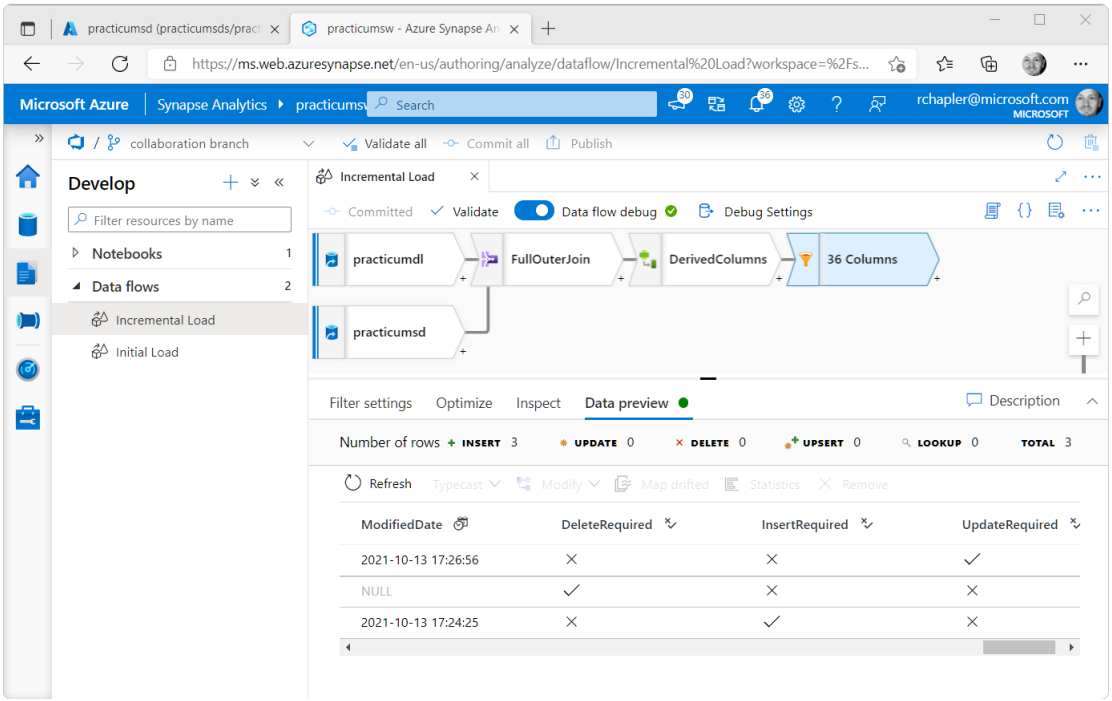


Execute the following T-SQL:

```
UPDATE [SalesLT].[Product]
SET [ListPrice] = 999.99, [ModifiedDate] = getdate()
WHERE ProductID=680
```

Confirm Changes

Return to the "**Incremental Load**" data flow and sequentially update with "**Data preview**" > **Refresh**.

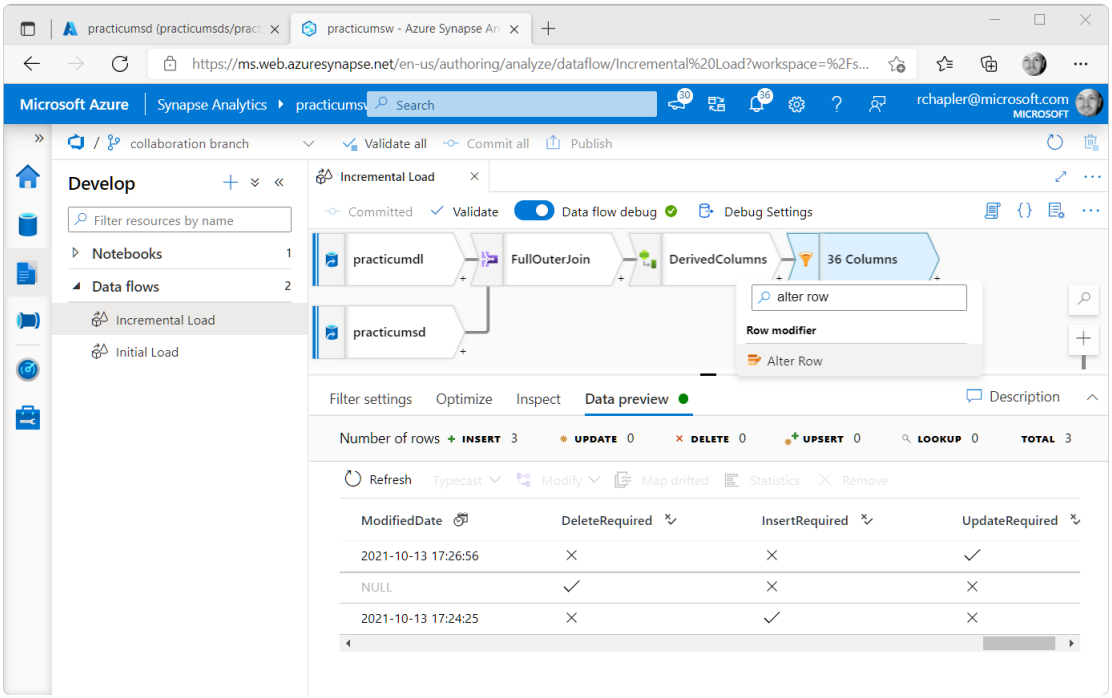


After update, you should see the three actions listed in **Filter** > "**Data preview**."

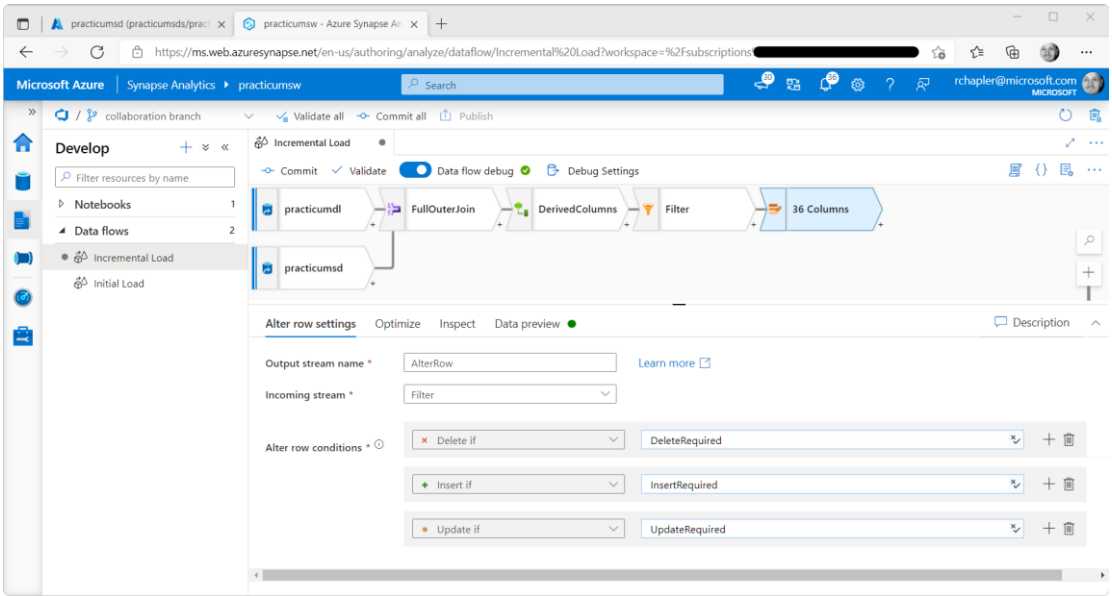
Finalize Data Flow

Alter Row

In this step, we will apply delete, insert, and update policies to the rows.



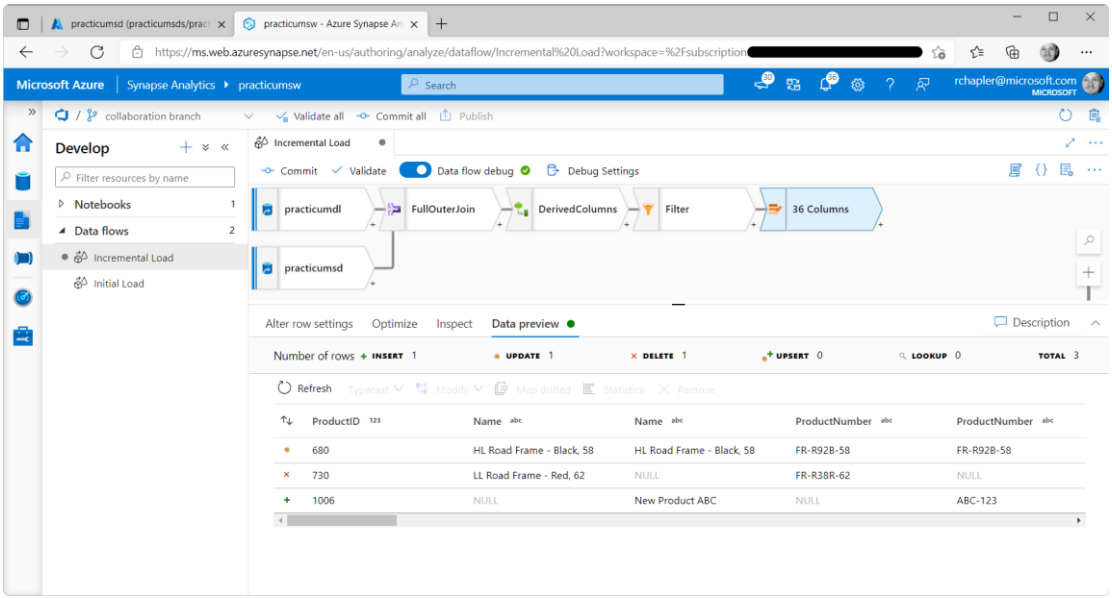
Click the **+** in the bottom right of the SQL source, and then select "**Alter Row**" from the resulting pop-up list.



Complete the “**Alter row settings**” tab, including:

Incoming Stream	Confirm default selection
Alter Row Conditions	“Delete if” DeleteRequired “Insert if” InsertRequired “Update if” UpdateRequired

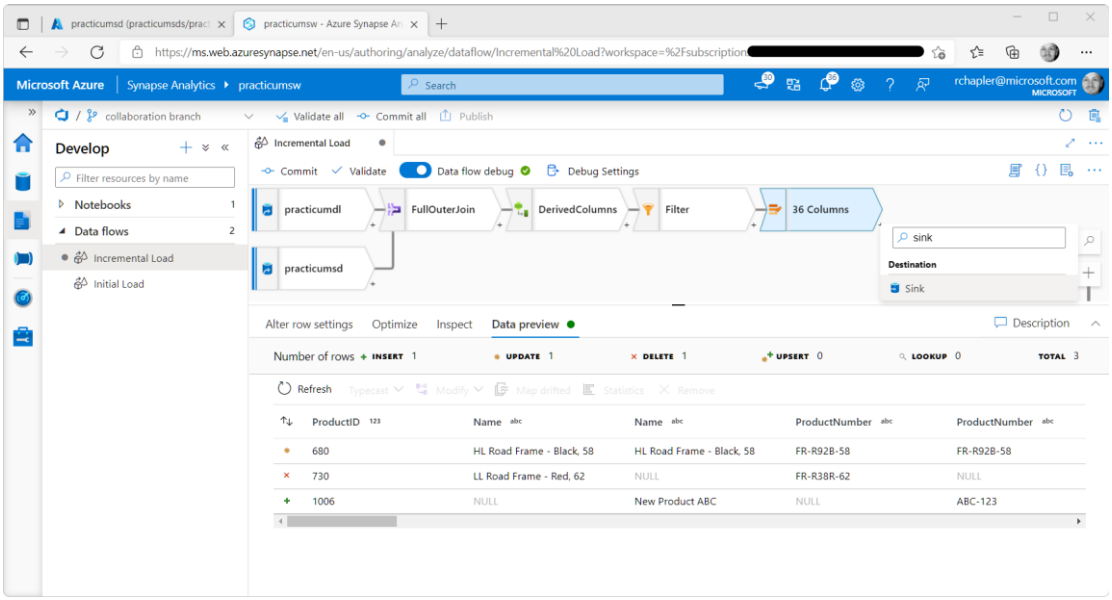
Navigate to the “**Data preview**” tab and then click **Refresh**.



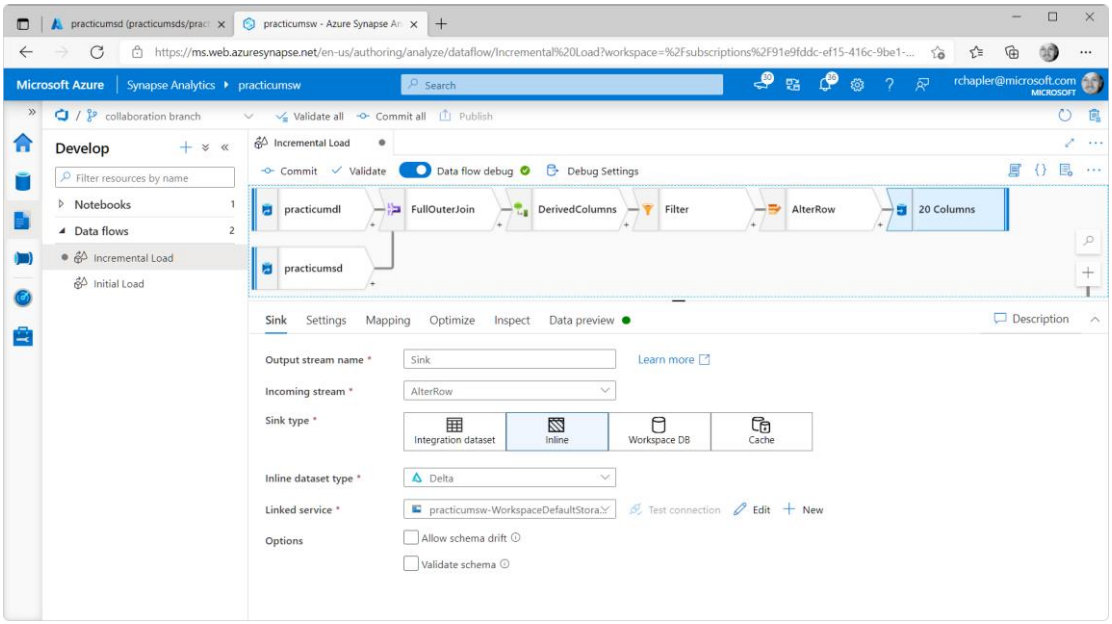
Click the **Commit** button.

Sink (delta lake)

In this step, we finalize the data flow.

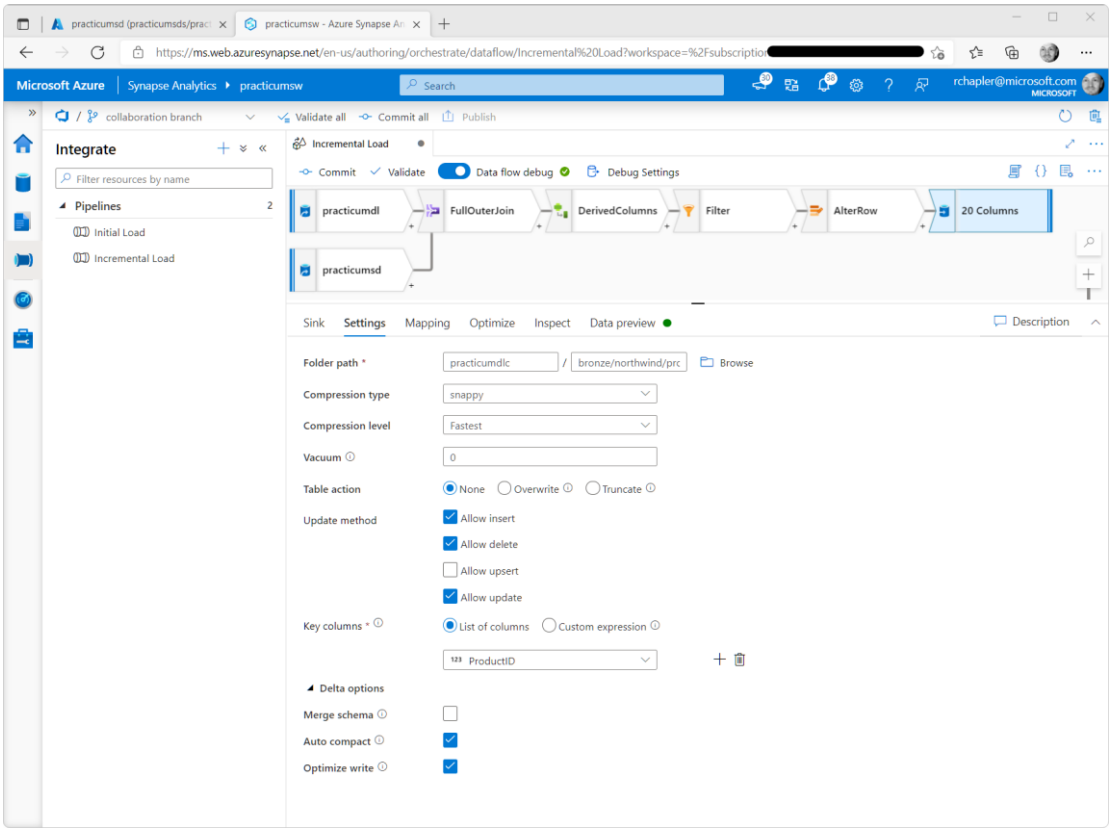


Click the **+** in the bottom right of the SQL source, and then select **Sink** from the resulting pop-up list.



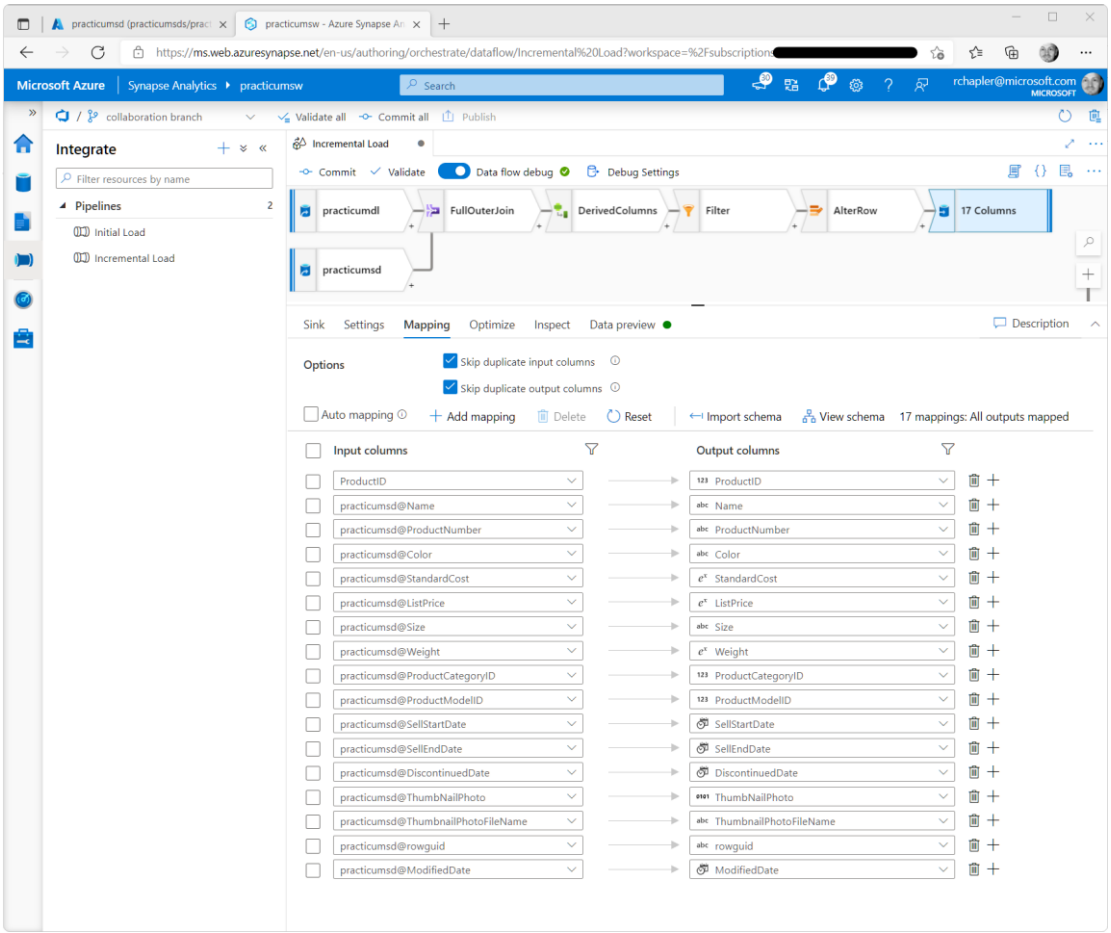
On the **Sink** tab, including:

Sink Type	Select **Inline**
Inline Dataset Type	Select **Delta**
Linked Service	Select **<UseCase>sw-WorkspaceDefaultStorage**



On the **Settings** tab, including:

File System	Enter <UseCase>dlc
Folder Path	Enter “bronze/northwind/product”
Compression Type	Select **snappy**
Compression Level	Select **Fastest**
Update Method	“Allow insert”, “Allow delete”, and “Allow update” checked
Key Columns	Confirm default selection, “**List of columns**” and select **ProductID**
Auto Compact	Checked
Optimize Write	Checked

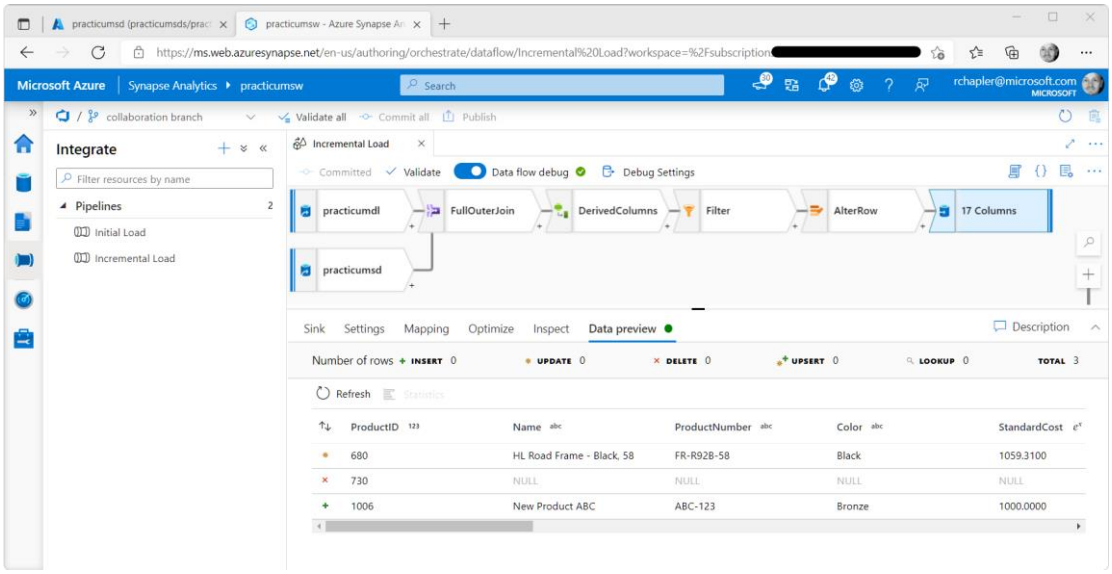


On the **Mapping** tab, un-check “Auto mapping”, click “**Import schema**” and then **Import** on the resulting pop-out.

Review the auto-generated list of columns and remove:

- Rows where “**Input columns**” begin with “**<UseCase>dl**…” (need data changes from the source, not the waterline dataset)
- Derived Columns DeleteRequired, InsertRequired, and UpdateRequired

Navigate to the “**Data preview**” tab and then click **Refresh**.



Click the **Commit** button.

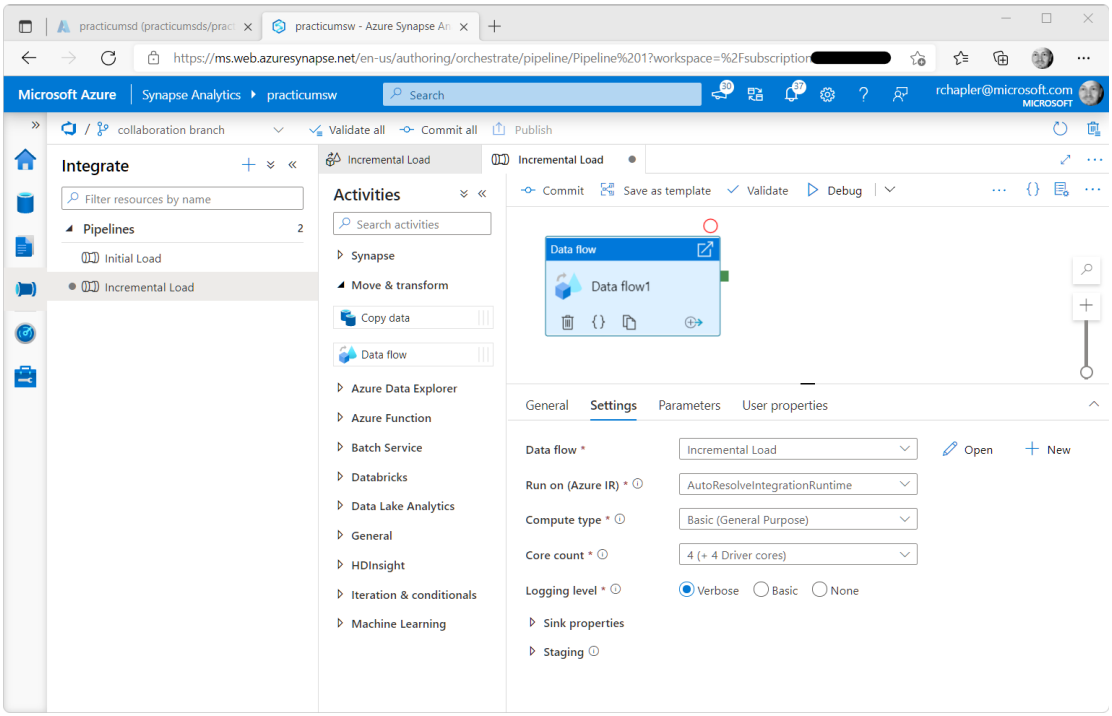
Create Pipeline

Click the **Integrate** icon in the navigation pane.

Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

Select "**Pipeline**" from the resulting drop-down menu.

On the **Properties** pop-out, enter Name, "**Incremental Load**."



Search for, and then drag-and-drop a "**Data flow**" component into the activity window.

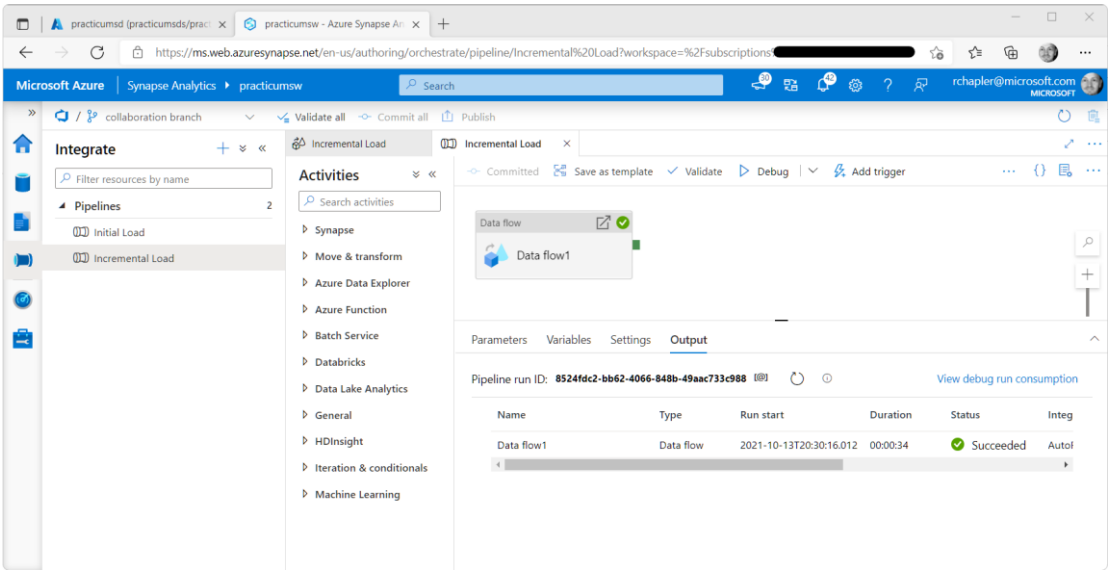
On the **Settings** tab of the "**Data flow**…" component, select your data flow.

Click the **Commit** button.

Confirm Success

Debug Pipeline / Data Flow

Click **Debug** and confirm successful execution.

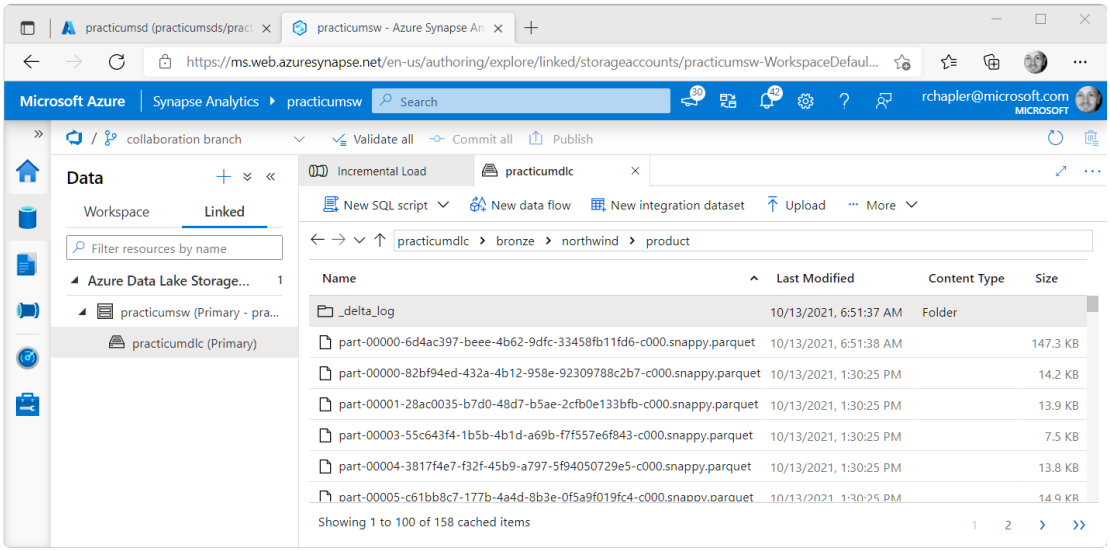


Review Produced Files

Click the **Data** icon in the navigation, and then the **Linked** tab.

Expand “Azure Data Lake Storage Gen2”, then <UseCase>sw to <UseCase>dlc.

Click to navigate to “<UseCase>dlc > bronze > northwind > product”



Review the files produced by delta lake.

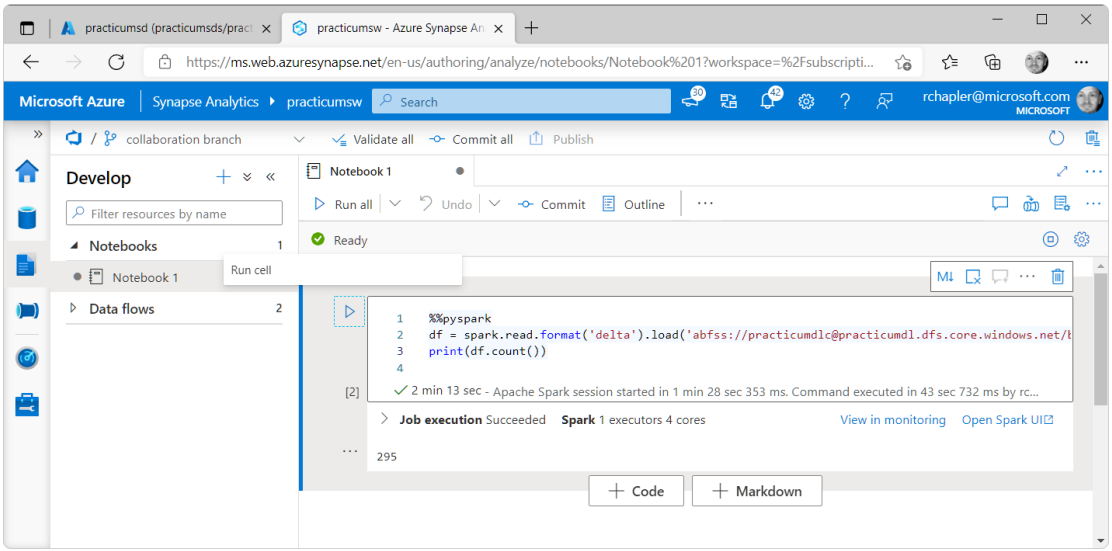
Query Record Count

Navigate to Synapse, and then click the **Develop** icon in the navigation pane.

Click the **+** icon just above and to the right of the "**Filter resources by name**" input.

Select "**Notebook**" from the resulting drop-down menu.

In your new notebook, select your Apache Spark pool (if required) from the "**Attach to**" drop-down menu.

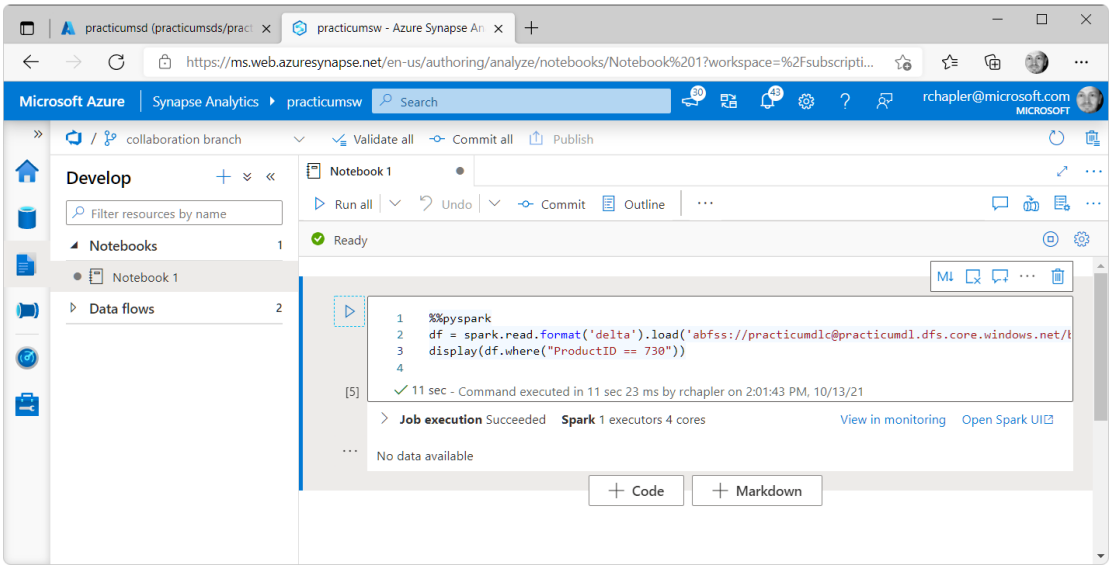


Run the following code to produce a count of records in the new Delta Lake:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/bronze/northwind/
product')
print(df.count())
```

Note: Expected value is 295… we started with 295, deleted one and inserted one.

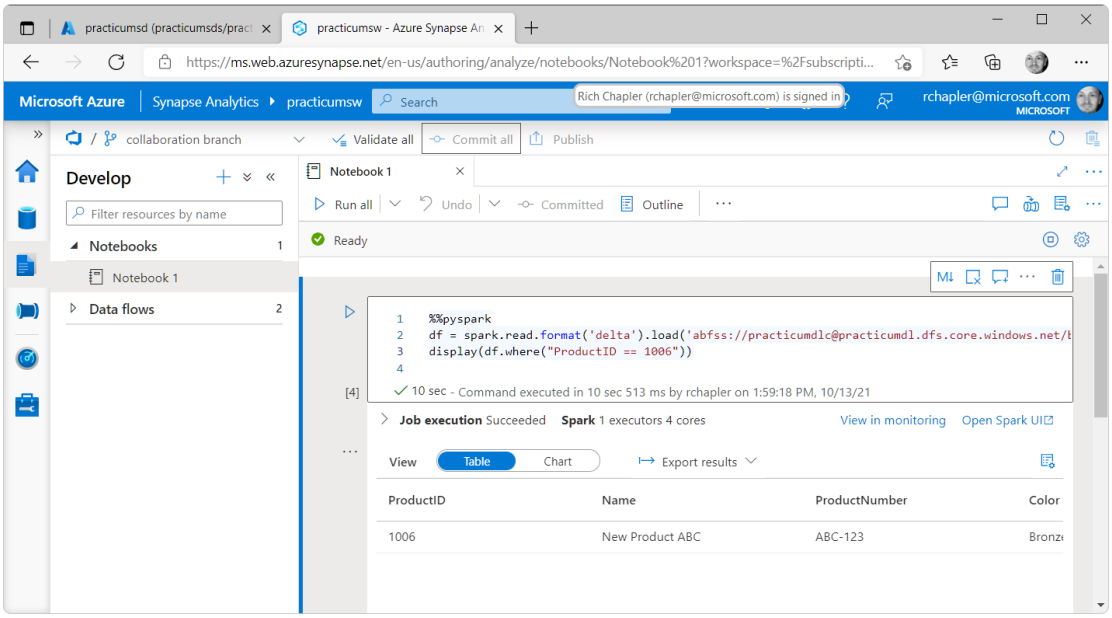
Confirm Delete Example



Run the following code to confirm our update example:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/bronze/northwind/
product')
display(df.where("ProductID == 730"))
```

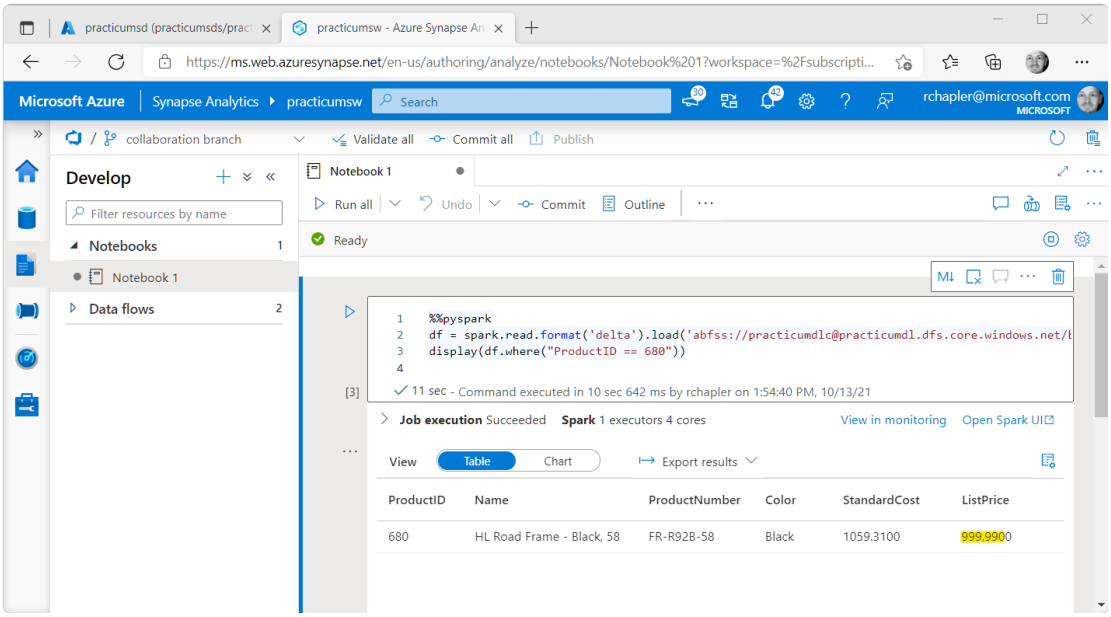
Confirm Insert Example



Run the following code to confirm our update example:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/bronze/northwind/
product')
display(df.where(“ProductID == 1006”))
```

Confirm Update Example



Run the following code to confirm our update example:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://<UseCase>dlc@<UseCase>dl.dfs.core.windows.net/bronze/northwind/
product')
display(df.where(“ProductID == 680”))
```

API Load

Logic Apps > Event Hub > Data Explorer

Use Case

Example notes:

- “We want to stream weather data from a publicly available API”

Prerequisites

This solution requires the following resources:

- Event Hub (with Primary Key associated with Shared Access Policy for “send”)
- Logic App (using Consumption plan)
- Data Explorer Cluster (or Synapse Data Explorer Pool) and Data Explorer Database

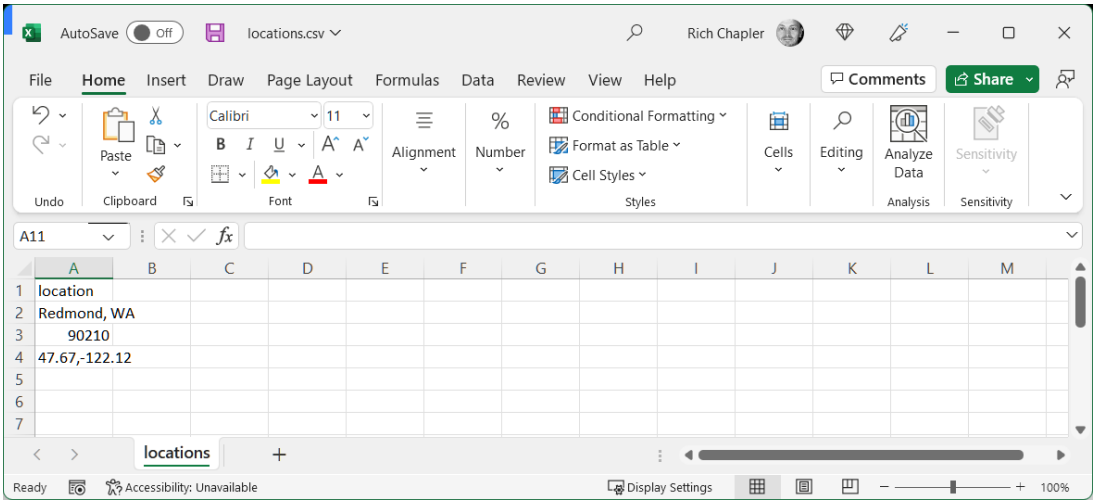
Prepare Destination

We will need two tables for this exercise:

- **Locations** … a proxy for a table that might exist in your environment; for example: you have a “Service Calls” table that includes location information regarding service calls (for which you want to correlate weather data)
- **Weather** … a table that will be home to regularly captured data from MSN Weather

In this section, we will prepare Locations.

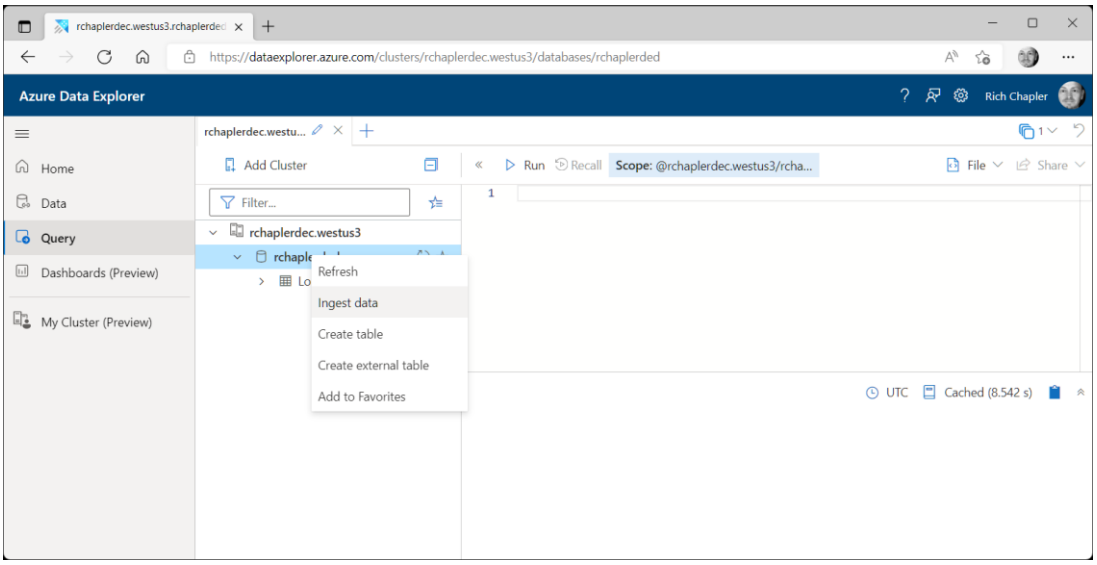
Open Excel, create a blank workbook and add a few rows with location information… valid entries include City, Region, State, Country, Landmark, PostalCode and Lat.Long.



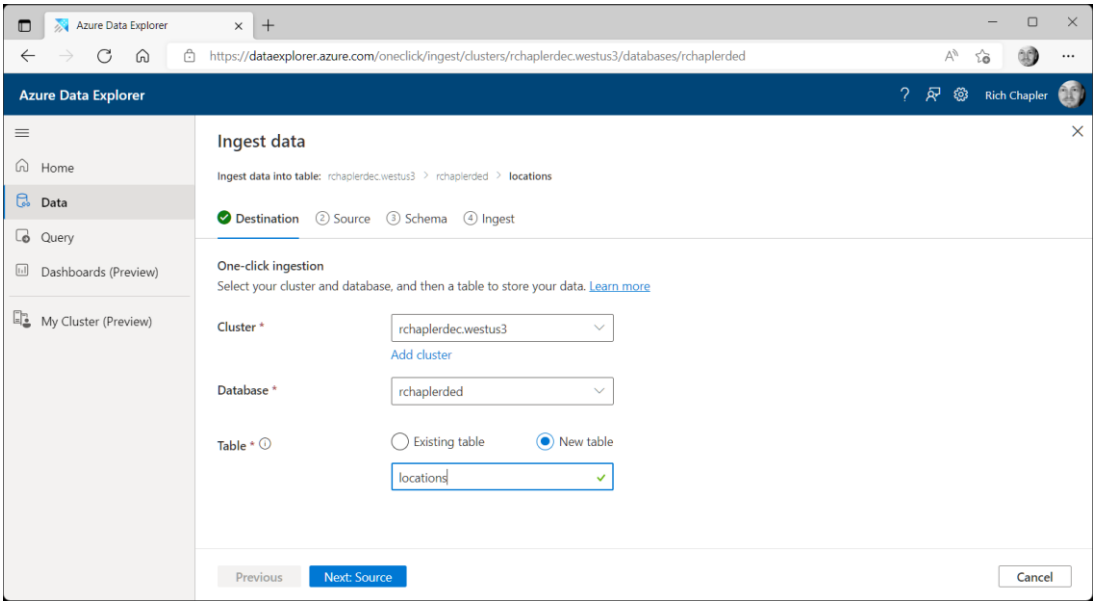
Note that I have included a header value in cell A1 as column name.

Save as “**locations.csv**” on your Desktop.

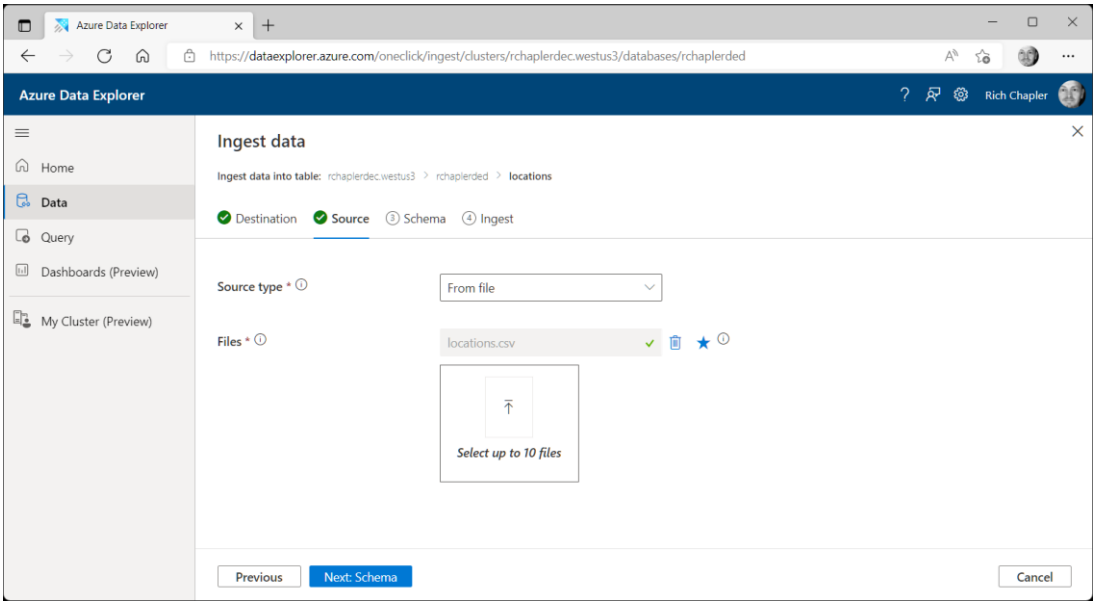
Navigate to https://dataexplorer.azure.com/



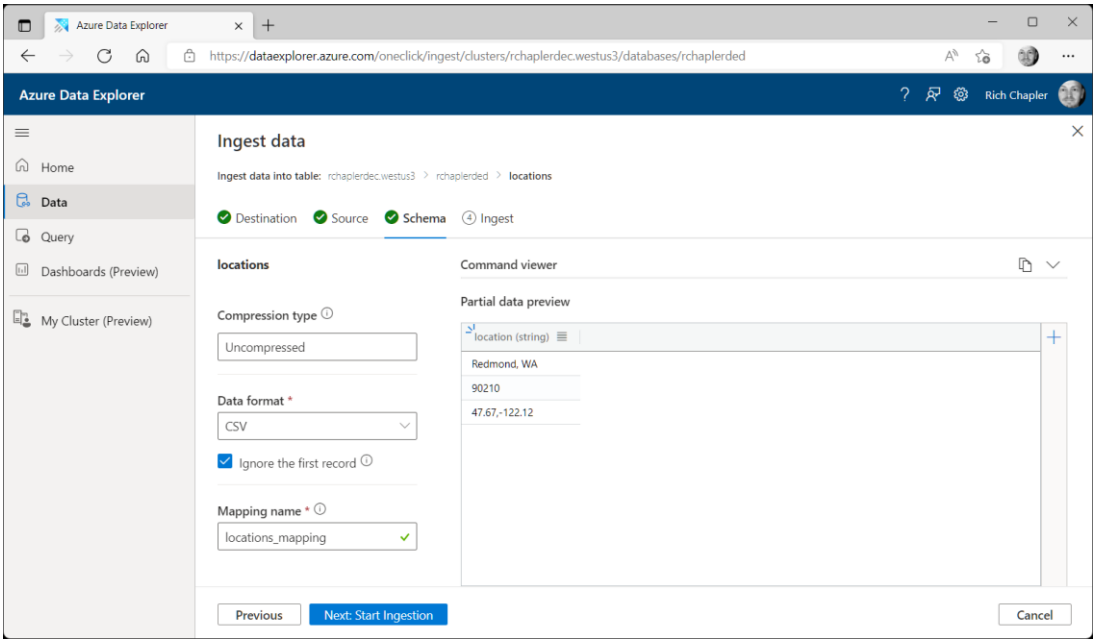
Right-click on your database and select "**Ingest data**" from the resulting drop-down menu.



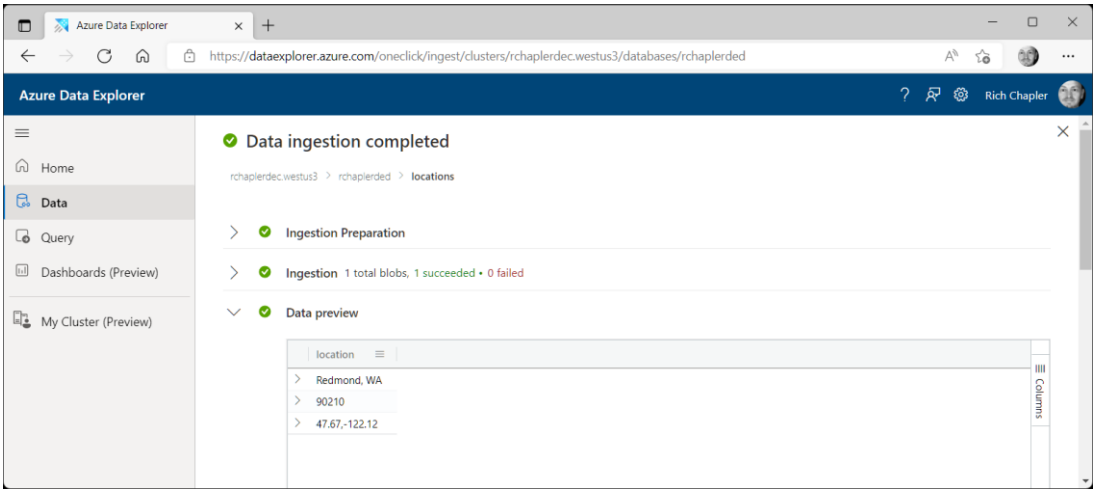
Complete the "**Ingest data**" form, **Destination** tab, confirm Cluster and Database values, select the "**New table**" radio button, enter "**locations**" in the **Table** textbox, and then click the "**Next: Source**" button.



Complete the "**Ingest data**" form, **Source** tab, confirm Source type "**From file**", browse to and select the previously created "locations.csv", and then click the "**Next: Schema**" button.

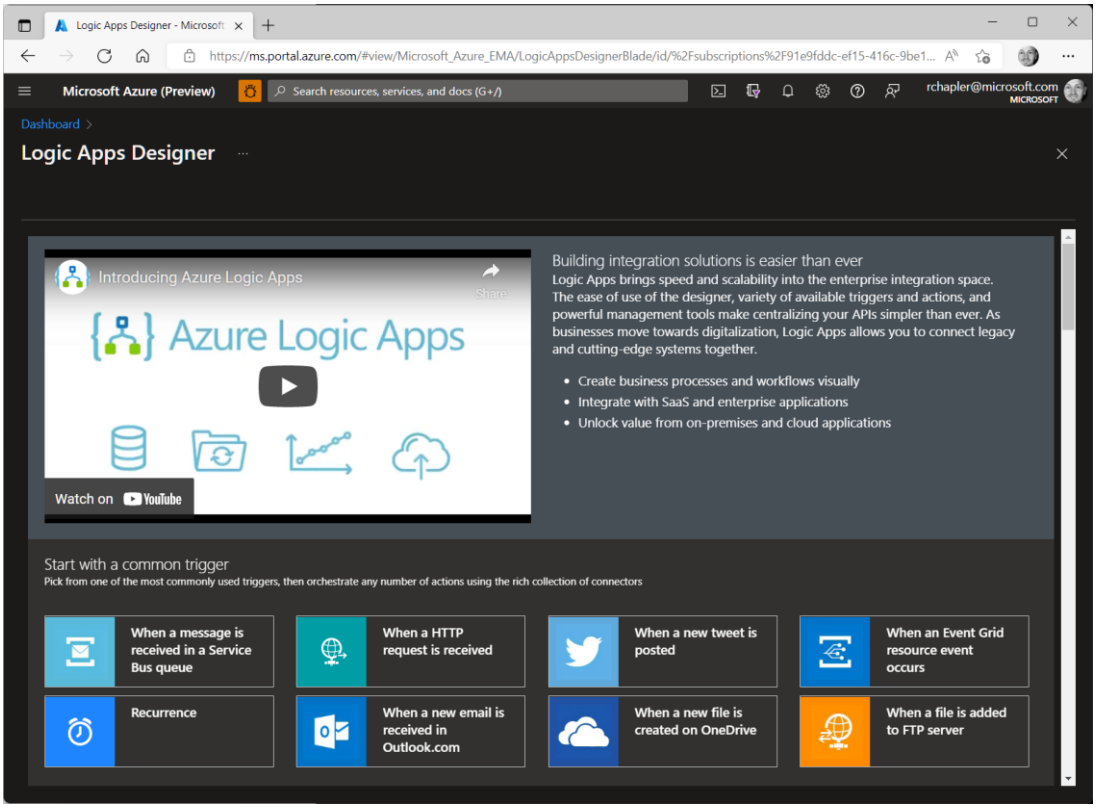


Complete the "**Ingest data**" form, **Schema** tab, review / update the generated schema definition (as needed) and then click the "**Next: Start Ingestion**" button.

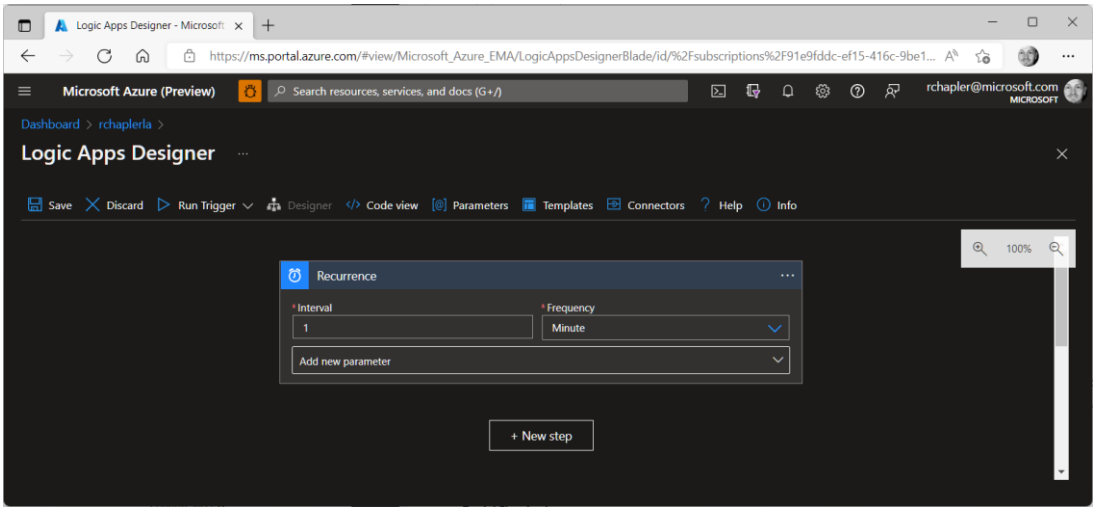


Confirm successful data ingestion.

Request Data



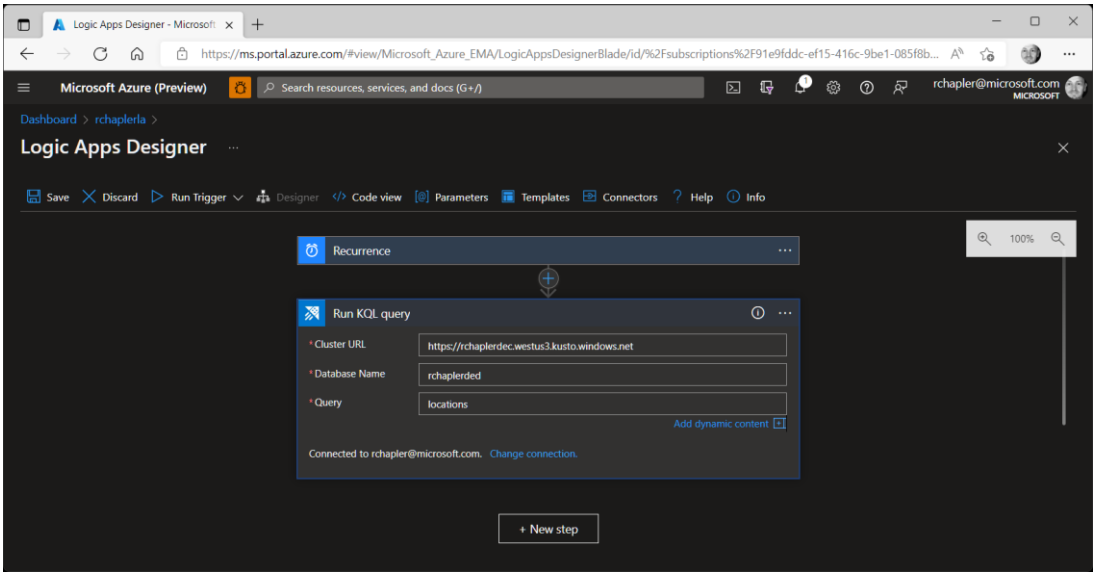
Navigate to your Logic App and select **Recurrence** from the "**Start with a common trigger**" page.



Complete the "**Logic Apps Designer**" form, **Recurrence** trigger, including:

Interval	Enter 1
Frequency	Select **Minute**

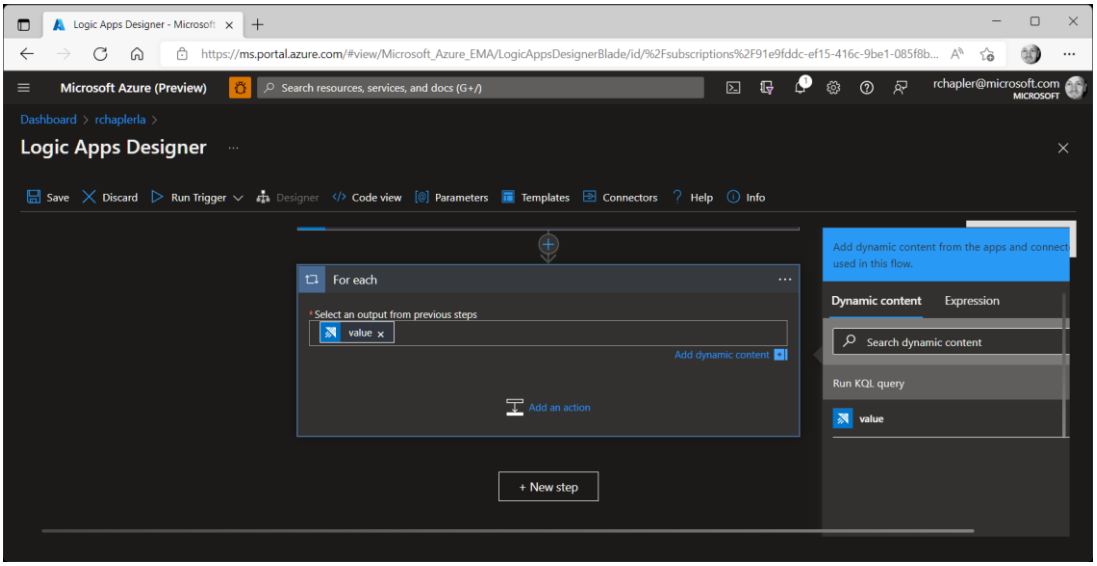
Click the "**+ New step**" button. On the resulting "**Choose an operation**" component, search for and select "**Azure Data Explorer**." On the **Actions** tab, select "**Run KQL query**." Select your **Tenant** and sign in, as needed.



Complete the "**Logic Apps Designer**" form, "**Run KQL query**" action, including:

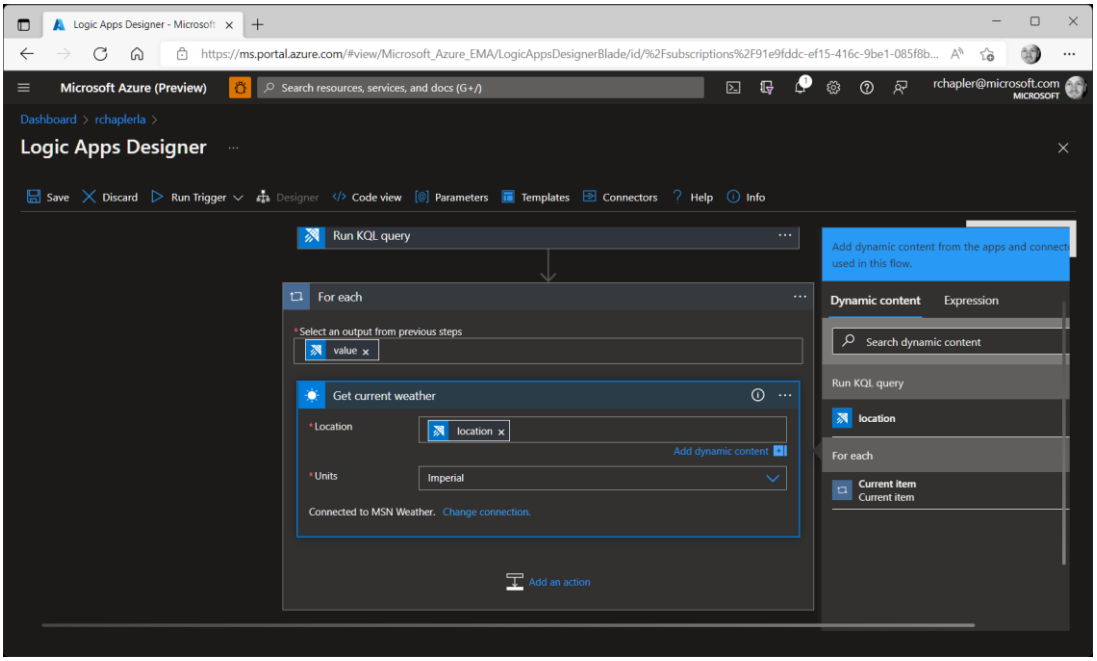
Cluster Name	Enter the URI of your Data Explorer cluster (or pool)
Database Name	Enter the name of your Data Explorer database
Query	Enter `locations`

Click the "**+ New step**" button.



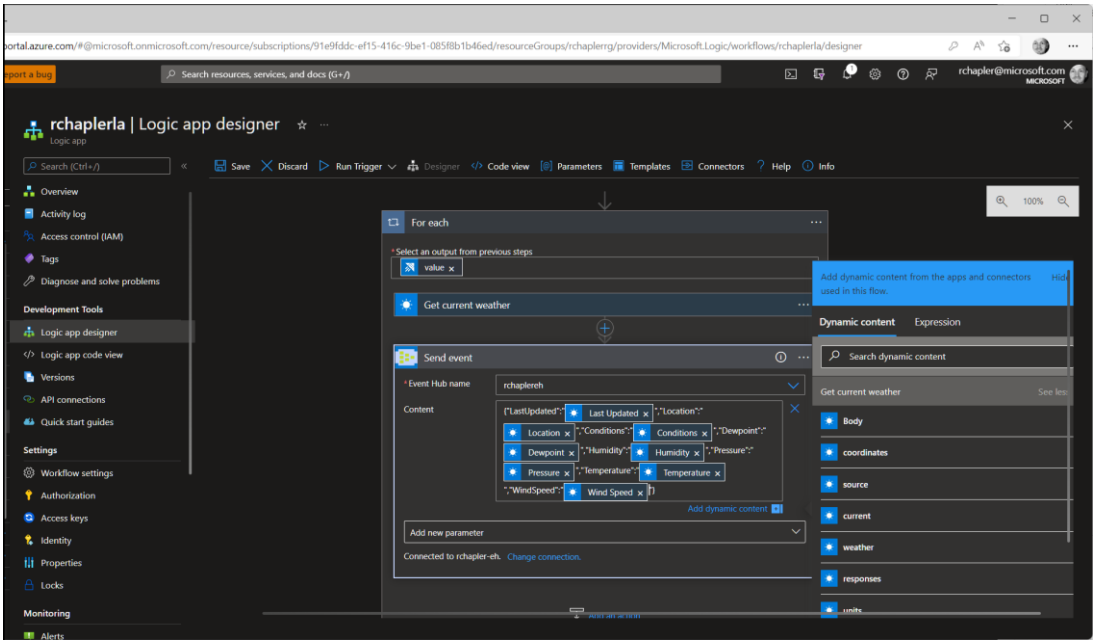
On the resulting "**Choose an operation**" component, search for and select "**Control**." On the **Actions** tab, select "**For each**." Click on the "**Select an output from previous steps**" textbox and then select dynamic content selection "**value**" from the "**Run KQL query**" group of the resulting "**Dynamic content**" pop-out.

Click the "**Add an action**" link.



On the resulting "**Choose an operation**" component, search for and select "**MSN Weather**." On the **Actions** tab, select "**Get current weather**." In the Location textbox, select dynamic content selection "**location**" from the "**Run KQL query**" group of the resulting "**Dynamic content**" pop-out.

Click the "**Add an action**" link.



On the resulting "**Choose an operation**" component, search for and select "**Event Hubs**." On the **Actions** tab, select "**Send event**." In the Location textbox, select dynamic content selection "**location**" from the "**Run KQL query**" group of the resulting "**Dynamic content**" pop-out.

Complete the "**Logic Apps Designer**" form, "**Event Hubs**" action, including:

Authentication Type	Select "**Access Key**"
Connection String	Enter the connection string for your Event Hub *Note: browse to the Shared Access Key on the Event Hub Namespace to easily copy the Event Hub Connection String*

Click the **Create** button. When processing is complete, select your Event Hub from the "**Event Hub Name**" drop-down menu. Next, click the "Add new parameter" drop-down menu, and select **Content**.

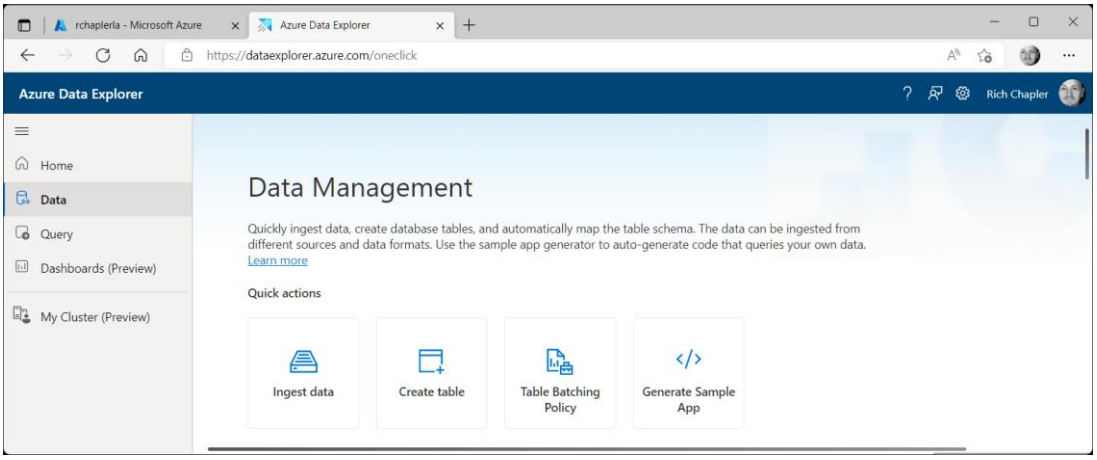
In the resulting **Content** textbox, construct a string using the following base string with bracketed values replaced with dynamic content.

```
{"LastUpdated":"{LastUpdated}","Location":"{Location}","Conditions":"{Conditions}","Dewpoint":"{Dewpoint}","Humidity":"{Humidity}","Pressure":"{Pressure}","Temperature":"{Temperature}","WindSpeed":"{WindSpeed}"}
```

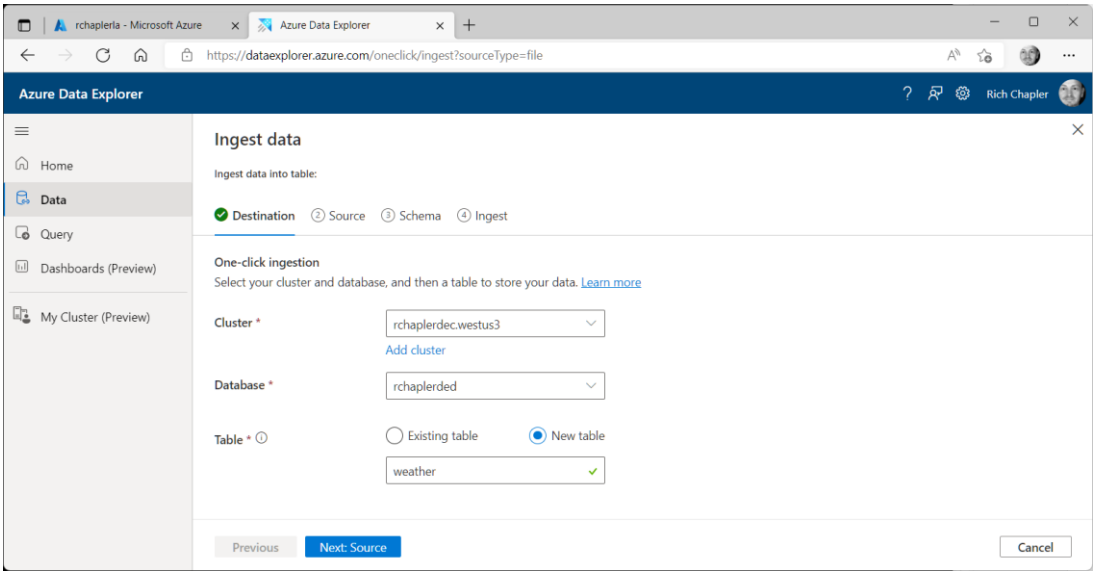
Click **Save** and then "**Run Trigger**" > **Run** to test. If everything is correct, the run should finish quickly, and you will see green checkmark circles next to all boxes.

Ingest Data

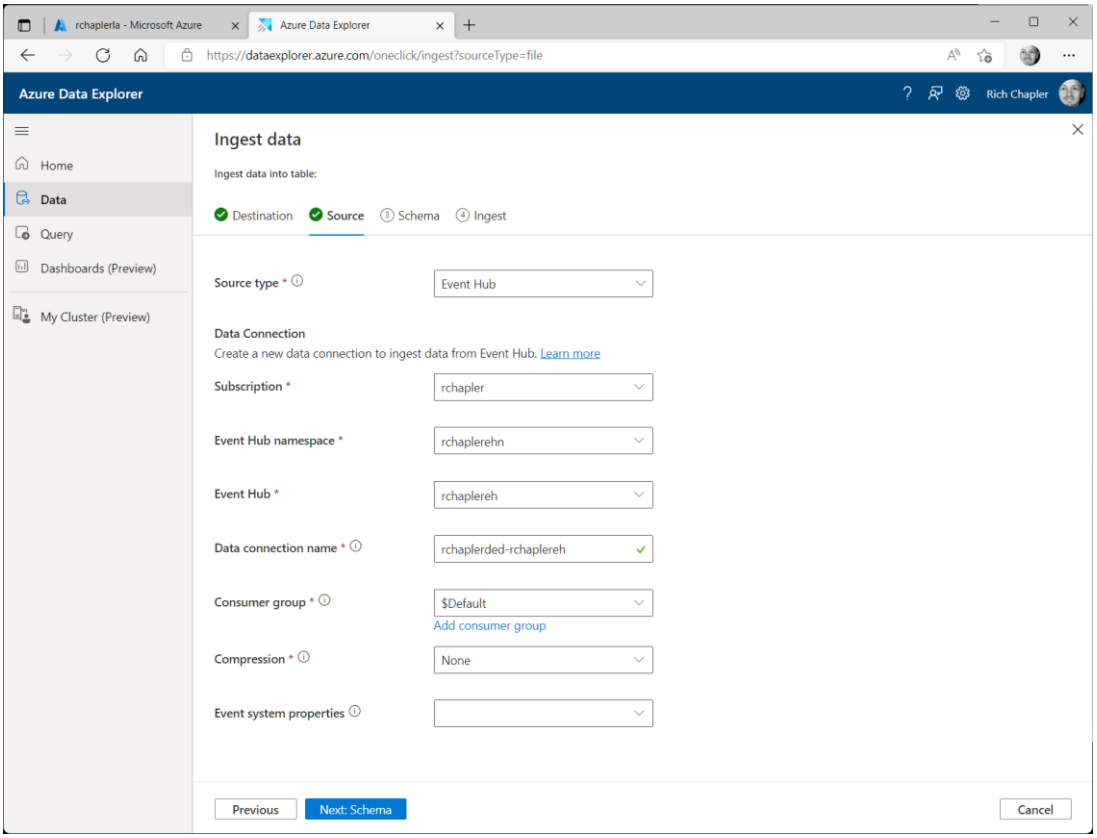
Navigate to https://dataexplorer.azure.com/



Click **Data** in the navigation pane and then the "**Ingest Data**" button on the resulting "**Data Management**" page.



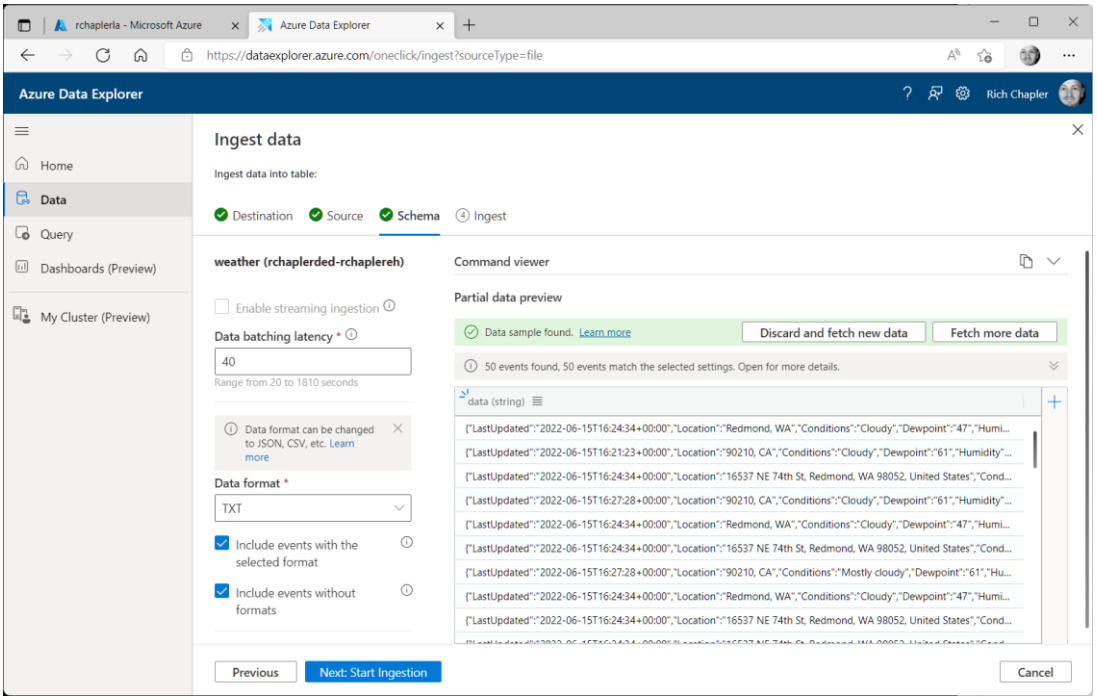
Complete the "**Ingest data**" form, **Destination** tab, confirm Cluster and Database values, select the "**New table**" radio button, enter "**weather**" in the **Table** textbox, and then click the "**Next: Source**" button.



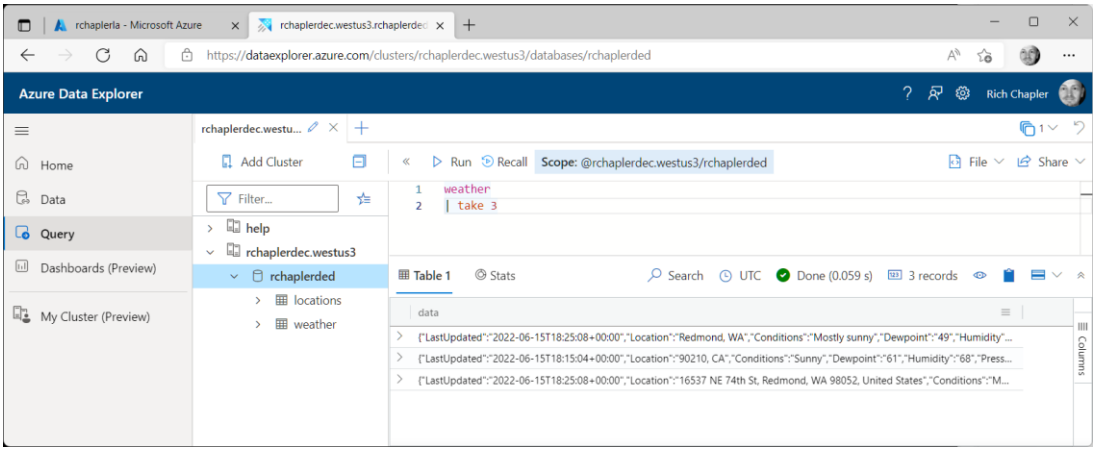
Complete the "**Ingest data**" form, **Source** tab, including:

Source Type	Select "Event Hub"
Event Hub Namespace	Select your Event Hub Namespace
Event Hub	Select your Event Hub
Data Connection Name	Confirm system-selected value
Consumer Group	Confirm default selection, "**$Default**"
Compression	Confirm default selection, **None**

Click the "**Next: Schema**" button.



Complete the "**Ingest data**" form, **Schema** tab, review / update the generated schema definition (as needed) and then click the "**Next: Start Ingestion**" button.



Confirm successful data ingestion.

…using Databricks

Follow these instructions to **source data from a REST API**.

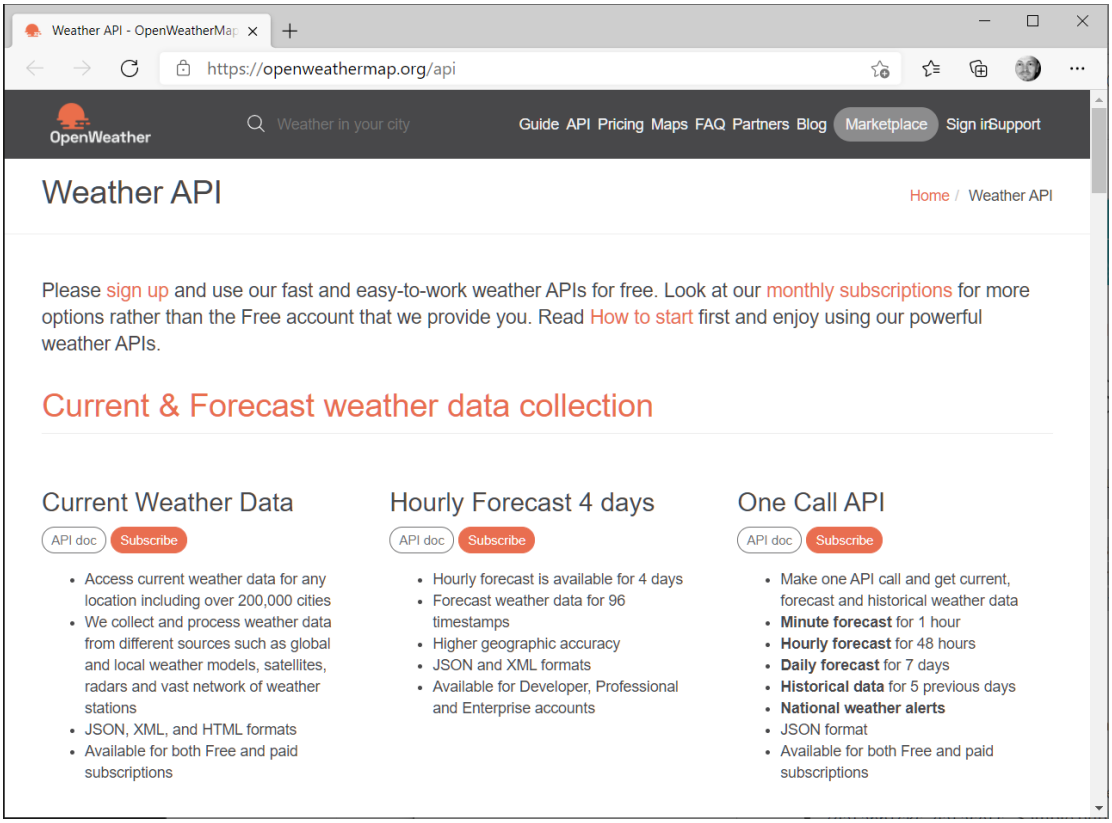
Prerequisites

This solution requires the following resources:

- Databricks
- Key Vault

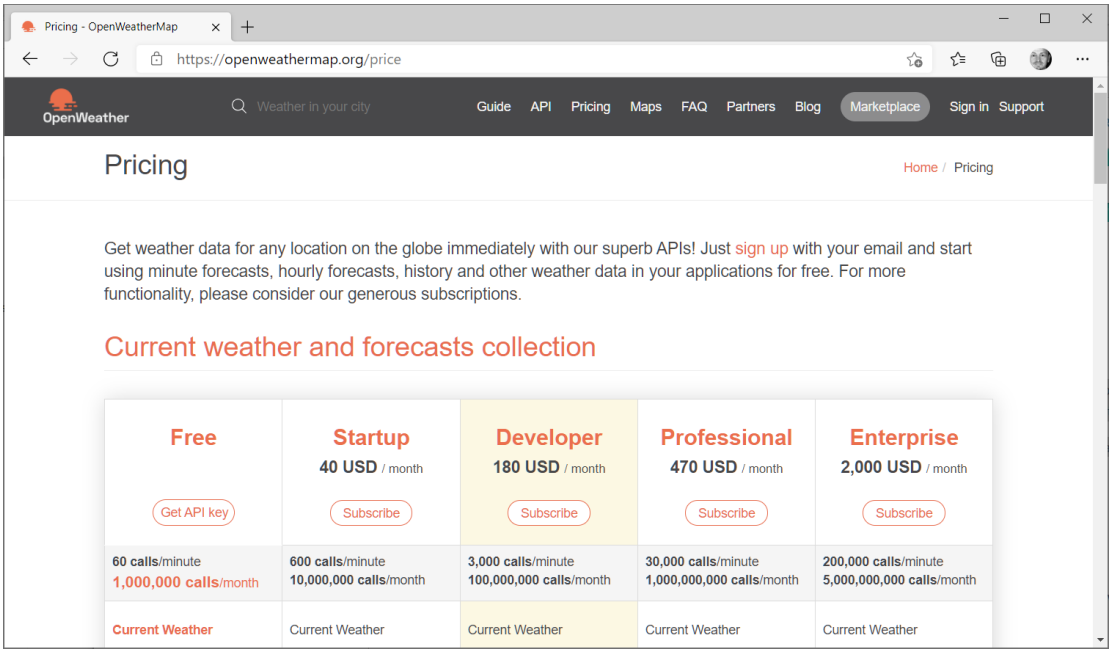
Stage Resources

For this exercise, we will use "**Current Weather**" data from https://openweathermap.org/api.



I chose this as my sample source because they have a Free subscription, their data is relatable, and it is easy to use.

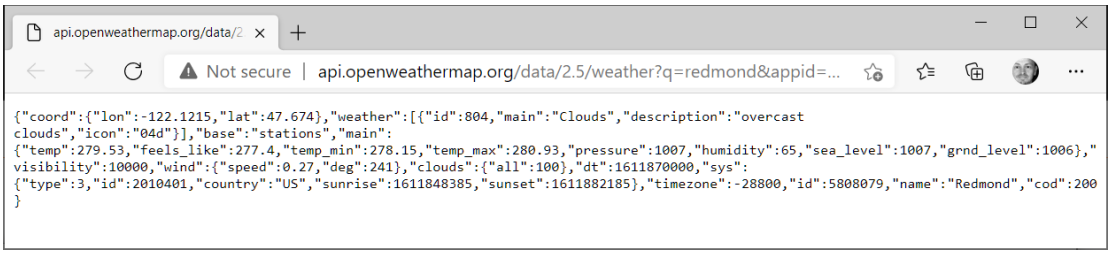
Navigate to https://openweathermap.org/price.



Click the "**Get API key**" button in the **Free** section and complete the "**Create a New Account**" process.

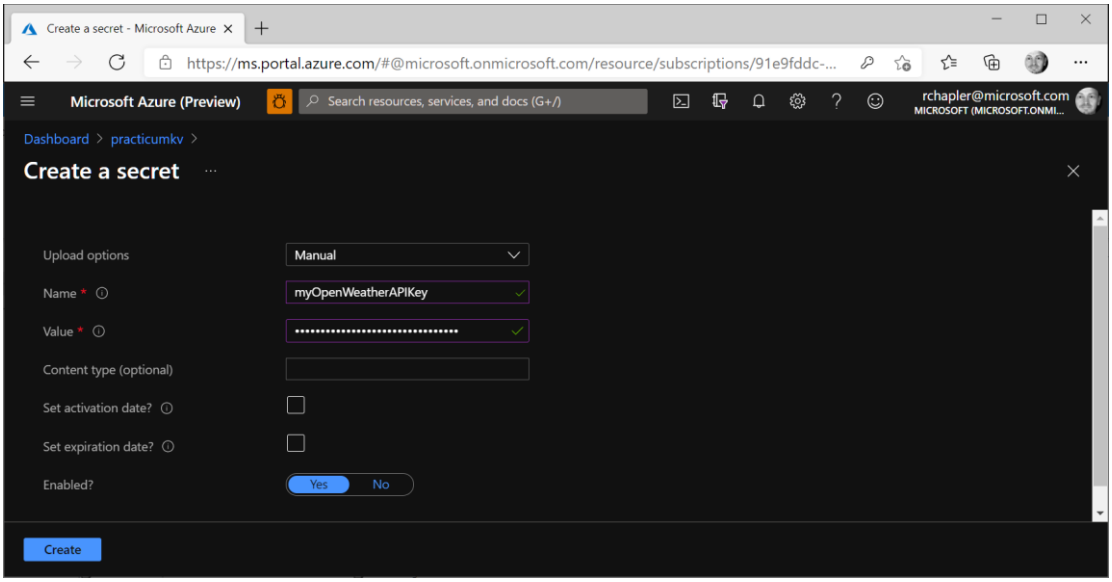
When you're successfully created an account, making an API call for "**Current Weather**" data is as simple as:

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```



Add Secret to Key Vault

Add your API Key to Key Vault.

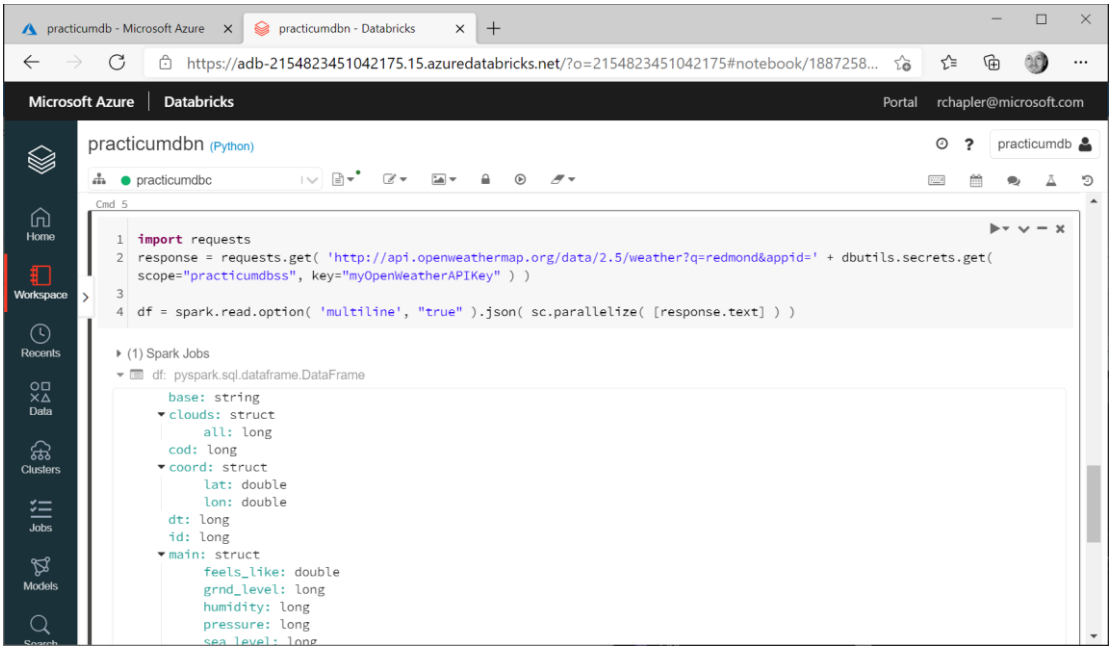


Prepare Logic

Navigate to Databricks and the **<UseCase>dbn** notebook. Add a new cell and paste the following code:

```
import requests
response = requests.get( 'http://api.openweathermap.org/data/2.5/weather?q=redmond&appid=' +
dbutils.secrets.get( scope="<UseCase>dbss", key="myOpenWeatherAPIKey" ) )

df = spark.read.option( 'multiline', "true" ).json( sc.parallelize( [response.text] ) )
```

Run the cell.



Another Python example (including POST, headers, and body):

```
import json
stringJSON = '{"data":[{"1212":0,"1227":0,…"ZZ9":0}]}'
theJSON = json.loads(stringJSON)
response = requests.post(
'http://eedfb2cf-deb1-4260-971d-d7a6c308d9b2.eastus.azurecontainer.io/score'
, headers={'CONTENT-TYPE': 'application/json'}
, json = theJSON
)
response.json()
```

…from Purview

Use Case

Example notes:

- “Our current governance solution lacks connectors for some Azure resource types”
- “We want to use Purview to collect source metadata from Azure resources”
- “Can we port Purview-collected metadata into our corporate-approved solution?”

Prerequisites

This solution requires the following resources:

- Application Registration (with a client secret)
- Data Lake (with Container)
- Purview
- Synapse (with linked service and dataset for your Data Lake)

Prepare Purview

Navigate to **Purview Governance Portal**, then **Data Map**, and then **Collections**.

Select the collection that is home to the metadata you plan to query.

Select the **Role Assignments** tab and add your Application Registration to “**Collection admins**”, “**Data source admins**”, and “**Data curators**.”

Create Linked Service

Navigate to Synapse Studio, click the **Manage** navigation icon, select “**Linked services**” from the “**External connections**” grouping in the resulting navigation. Click the “**+ New**” button.

Complete the “**New linked service**…” pop-out, including:

Connect via…	Confirm default selection, “**AutoResolveIntegrationRuntime**”

Base URL	Modify and enter:
	`https://{Purview Instance Name}.purview.azure.com/scan/datasources/{Purview Data Source Name}?api-version=2022-02-01-preview`
	
	Example:
	`https://rchaplerp.purview.azure.com/scan/datasources/rchaplerdec?api-version=2022-02-01-preview`

Authentication Type	Select **Anonymous**

Click “**Test connection**” to confirm successful connection and then click the **Save** button.

Create Dataset

Navigate to Synapse Studio and click the **Data** navigation icon.

Select "**Integration datasets**" from the **Linked** tab in the resulting navigation.

Click "**+**" and select "**Integration dataset**" from the resulting drop-down menu.

Complete the "**New integration dataset**" pop-out, search for and select **REST**.

Click the **Continue** button.

Complete the "**Set properties**" pop-out, including:

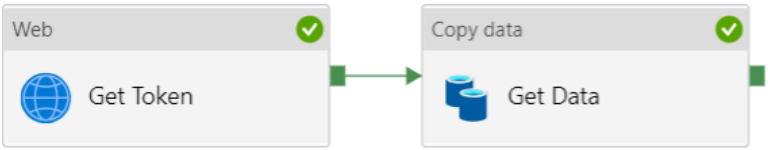
Linked Service	Select the REST Linked Service (created in the prior step)

Click the **OK** button.

On the resulting screen, click "**Test connection**" to confirm successful connection.

Create Pipeline

When we are finished, our Synapse Pipeline will look and function as snipped below.



Navigate to Synapse Studio, click the **Integrate** navigation icon, click "**+**" and select **Pipeline** from the resulting drop-down menu.

Activity 1: Get Token

This activity will make a REST API call to http://login.microsoftonline.com and get a bearer token.

Expand **General** in the **Activities** bar, then drag-and-drop a **Web** component into the activity window. On the **Settings** tab, including:

URL	Modify and enter: `https://login.microsoftonline.com/{TenantId}/oauth2/token`
Method	Select **POST**
Headers	Click "**+ Add**" and enter key-value pair: `content-type, application/x-www-form-urlencoded`
Body	Modify and enter: `grant_type=client_credentials&client_id={Client Identifier}&client_secret={Client Secret}& resource=https://purview.azure.net`

Click **Debug** and monitor result to confirm successful progress.

Activity 2: Get Data

For this exercise, focused simply on demonstrating end-to-end solution functionality, we are going to pull information about data sources and capture the result in a file in the Data Lake.

Many other API options are discussed in the Purview API documentation:
https://docs.microsoft.com/en-us/rest/api/purview/

Expand "**Move & Transform**" in the **Activities** bar, drag-and-drop a "**Copy data**" component into the activity window and then create a dependency from the "**Get Bearer Token**" component.

On the **Source** tab, including:

Source Dataset	Select your REST dataset
Request Method	Select **GET**

Additional Headers	Click "**+ Add**" and enter key-value pair:
	`content-type, application/json;charset=utf-8`
	
	Click "**+ Add**" and enter key-value pair (with dynamic content):
	`Authorization, @concat('Bearer ',activity('Get Token').output.access_token)`

No additional configuration is required on this tab.

On the **Sink** tab, including:

Sink Dataset	Select your Data Lake dataset

No additional configuration is required on this tab.

Confirm Success

Click **Debug** and monitor result to confirm successful progress.

Special Load

Batch Upsert

Follow these instructions to provide for recurring upsert of data.

Prerequisites

This solution requires the following resources:

- Data Lake
- Databricks

CREATE TABLE

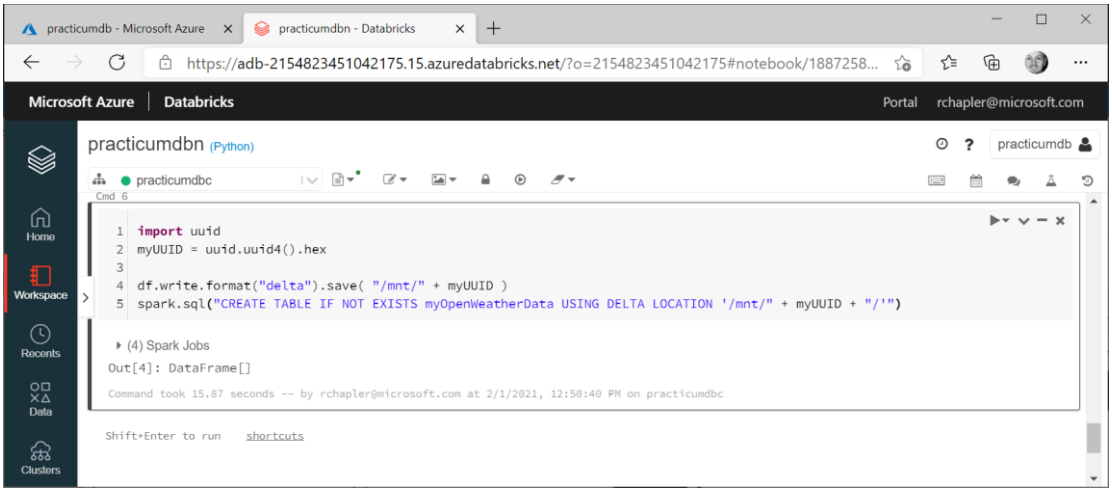
Navigate to Databricks and the **<UseCase>dbn** notebook.

Add a new cell and paste the following code:

```
import uuid
myUUID = uuid.uuid4().hex

df.write.format("delta").save( "/mnt/" + myUUID )
spark.sql("CREATE TABLE IF NOT EXISTS myOpenWeatherData USING DELTA LOCATION '/mnt/" + myUUID + "/'")
```

Notes:

- We are using dataframe **df** created in Objective 3 … The Logic
- We are using format **delta** to provide for future UPSERT operations.

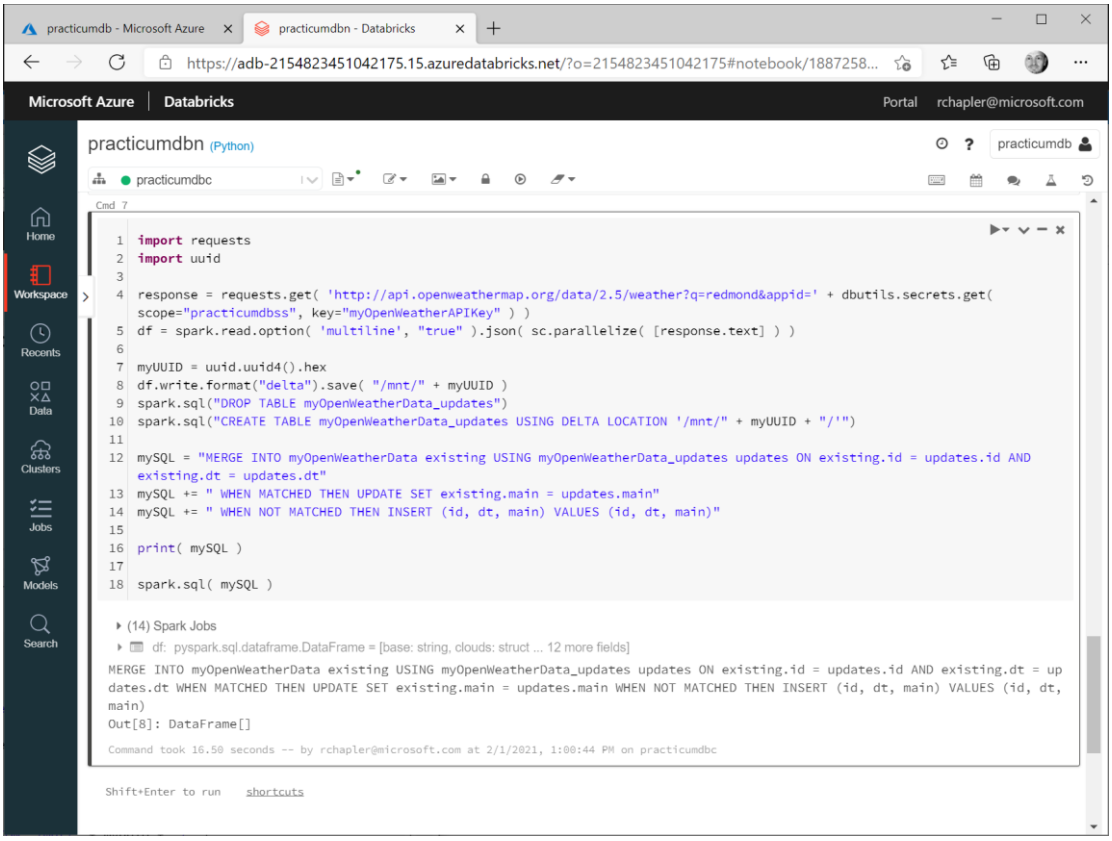


MERGE INTO

Navigate to Databricks and the **<UseCase>dbn** notebook. Add a new cell and paste the following code:

```
import requests
import uuid

response = requests.get( 'http://api.openweathermap.org/data/2.5/weather?q=redmond&appid=' +
dbutils.secrets.get( scope="<UseCase>dbss", key="myOpenWeatherAPIKey" ) )
df = spark.read.option( 'multiline', "true" ).json( sc.parallelize( [response.text] ) )

myUUID = uuid.uuid4().hex
df.write.format("delta").save( "/mnt/" + myUUID )
spark.sql("DROP TABLE myOpenWeatherData_updates")
spark.sql("CREATE TABLE myOpenWeatherData_updates USING DELTA LOCATION '/mnt/" + myUUID + "/'")

mySQL = "MERGE INTO myOpenWeatherData existing USING myOpenWeatherData_updates updates ON existing.id =
updates.id AND existing.dt = updates.dt"
mySQL += " WHEN MATCHED THEN UPDATE SET existing.main = updates.main"
mySQL += " WHEN NOT MATCHED THEN INSERT (id, dt, main) VALUES (id, dt, main)"

print( mySQL )

spark.sql( mySQL )
```

Run the cell.

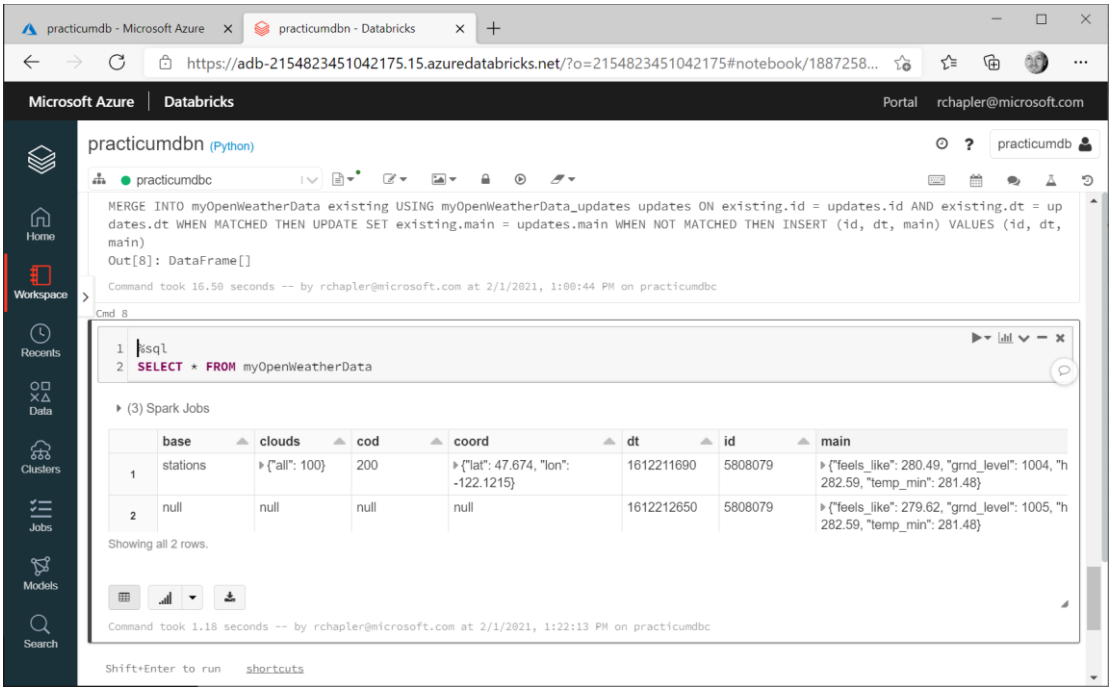


Confirm Success

Navigate to Databricks and the **<UseCase>dbn** notebook. Add a new cell and paste the following code:

```
%sql
SELECT * FROM myOpenWeatherData
```

Run the cell.



Note that the row created using MERGE INTO has a limited number of populated cells since only some of the possible cells were included in WHEN NOT MATCHED THEN INSERT (id, dt, main)… we can, of course, expand this to include all fields about which we care.

Synchronize Unstructured Data

Server-to-Cloud

(featuring Linux and AzCopy)

Follow these instructions to automate recurring synchronization of unstructured data {e.g., image files} from an on-prem Linux server to Azure Storage.

Use Case: Customer XYZ shared the following requirements:

- **Unstructured data** (images) are regularly loaded to an **on-prem Linux server**
    - This configuration is owned by a separate group and cannot be changed to load directly to Azure
- Synchronization of new data must occur **as soon as it becomes available**

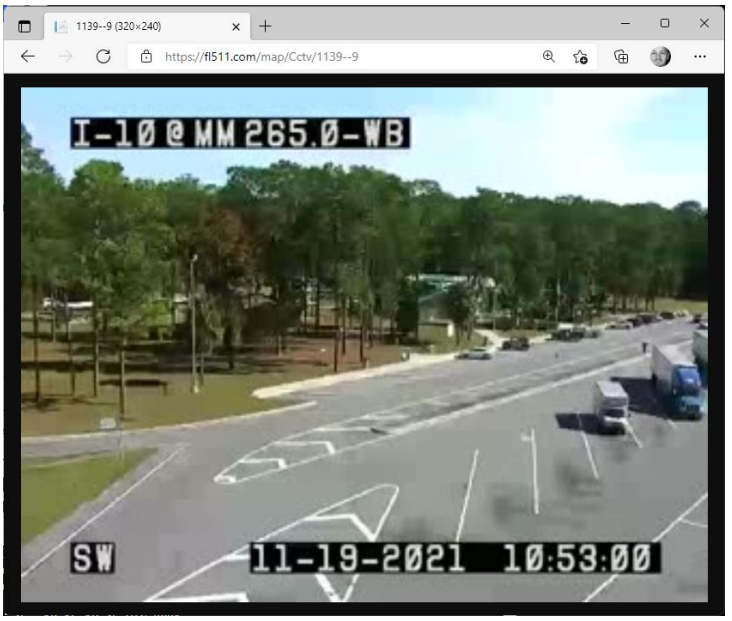
Prerequisites

This solution requires the following resources:

- Storage Account with container named **images** and Shared Access Token (with **Read**, **Write**, **Delete**, **List** permissions)
- Virtual Machine (Linux/Ubuntu) … intended to mimic the customers' on-prem configuration

Stage Sample Data

In this section, we will load three sample images from an online traffic cam source (1139--9 (320×240) (fl511.com)) onto our Linux virtual machine.



Navigate to Azure Cloud Shell.

Connect using SSH

Update and execute the following command to connect to the server using SSH:

```
ssh -i ~/.ssh/id_rsa rchapler@{public ip address}
```

Create Images Directory

Execute the following command to create a directory for our sample images:

```
mkdir images
```

Get Sample Images

Execute the following command to get a sample image from the traffic cam source:

```
wget -O images/01.jpg https://fl511.com/map/cctv/1139--9
```

Update and execute the previous command to get additional, updated sample images (every ~minute); example:

```
wget -O images/02.jpg https://fl511.com/map/cctv/1139--9
wget -O images/03.jpg https://fl511.com/map/cctv/1139--9
```

Execute the following commands to list sample files in the **images** directory:

```
cd images
ls
```

Prepare AzCopy

In this section, we will download, install, and test AzCopy on our Linux virtual machine.

Download AzCopy

Execute the following command to download the AzCopy installer:

```
wget https://aka.ms/downloadazcopy-v10-linux
```

You can expect a response like:

```
--2021-11-23 16:21:50-- https://aka.ms/downloadazcopy-v10-linux
Resolving aka.ms (aka.ms)... 23.63.47.52
Connecting to aka.ms (aka.ms)|23.63.47.52|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://azcopyvnext.azureedge.net/release20211027/azcopy_linux_amd64_10.13.0.tar.gz
[following]
--2021-11-23 16:21:50--
https://azcopyvnext.azureedge.net/release20211027/azcopy_linux_amd64_10.13.0.tar.gz
Resolving azcopyvnext.azureedge.net (azcopyvnext.azureedge.net)... 104.86.182.19, 104.86.182.43,
2600:1409:9800:21::17d8:93b8, ...
Connecting to azcopyvnext.azureedge.net (azcopyvnext.azureedge.net)|104.86.182.19|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11876012 (11M) [application/gzip]
```

```
Saving to: ‘downloadazcopy-v10-linux’

downloadazcopy-v10-linux
100%[=============================================================================================================
==>] 11.33M --.-KB/s  in 0.08s

2021-11-23 16:21:50 (141 MB/s) - ‘downloadazcopy-v10-linux’ saved [11876012/11876012]
```

Extract Download

Execute the following command to extract the downloaded GZIP file:

```
tar -xvf downloadazcopy-v10-linux
```

You can expect a response like:

```
azcopy_linux_amd64_10.13.0/
azcopy_linux_amd64_10.13.0/NOTICE.txt
azcopy_linux_amd64_10.13.0/azcopy
```

AzCopy Sync

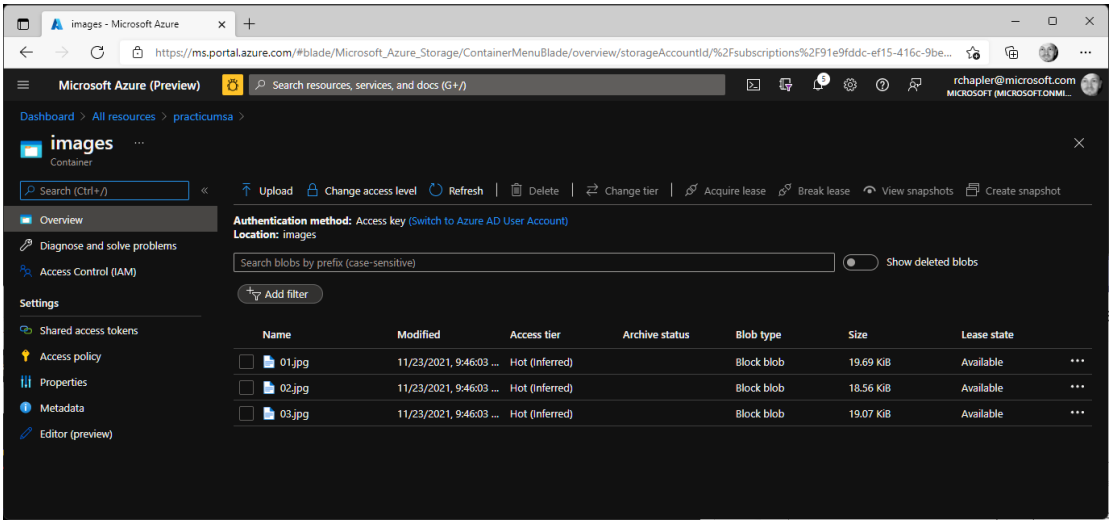
Update and execute the following command to test file synchronization using AzCopy:

```
./azcopy_linux_amd64_10.13.0/azcopy sync "/home/rchapler/images"
"https://<UseCase>sa.blob.core.windows.net/images?sp=rwdl&st=2021-11-23T20:23:01Z&se=2022-10-
01T03:23:01Z&spr=https&sv=2020-08-04&sr=c&sig=U3o98QJ1NJ8wAldkjBBJv8Eken4N3Rlrr5OgLxsq6PE%3D" --
recursive
```

You can expect a response like:

```
INFO: Any empty folders will not be processed, because source and/or destination doesn't have full
folder support

Job ffac3e11-2fc7-ee4f-5216-95fd028b1a51 has started
Log file is located at: /home/rchapler/.azcopy/ffac3e11-2fc7-ee4f-5216-95fd028b1a51.log

3 Files Scanned at Source, 0 Files Scanned at Destination

Job ffac3e11-2fc7-ee4f-5216-95fd028b1a51 Summary
Files Scanned at Source: 3
Files Scanned at Destination: 0
Elapsed Time (Minutes): 0.0334
Number of Copy Transfers for Files: 3
Number of Copy Transfers for Folder Properties: 0
Total Number Of Copy Transfers: 3
Number of Copy Transfers Completed: 3
Number of Copy Transfers Failed: 0
Number of Deletions at Destination: 0
Total Number of Bytes Transferred: 58702
Total Number of Bytes Enumerated: 58702
Final Job Status: Completed
```

Navigate to the **images** container in your Storage Account and confirm file synchronization.



Prepare Script

In this section, we will prepare a script and confirm success.

Create Script with Vim

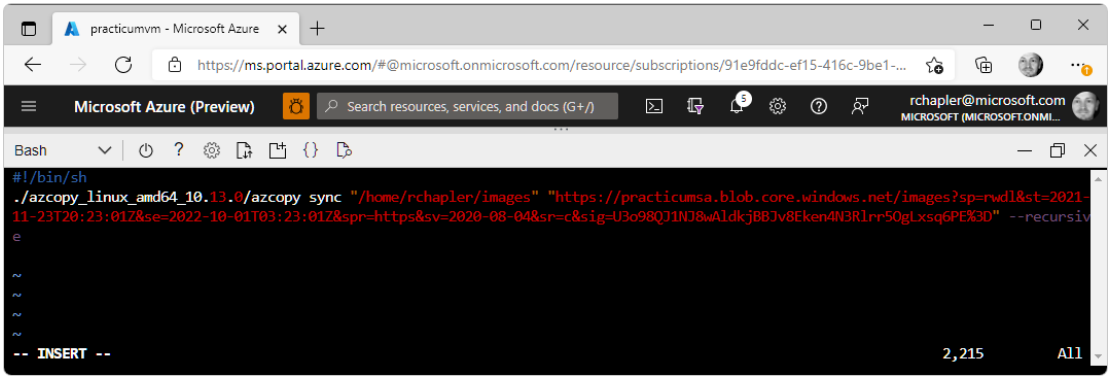
Execute the following command to open a text editor:

```
vim script.sh
```

In the editor, type **i** to change to **INSERT** mode.

Update and paste the following command lines into the text editor:

```
#!/bin/sh
./azcopy_linux_amd64_10.13.0/azcopy sync "/home/rchapler/images"
"https://<UseCase>sa.blob.core.windows.net/images?sp=rwdl&st=2021-11-23T20:23:01Z&se=2022-10-
01T03:23:01Z&spr=https&sv=2020-08-04&sr=c&sig=U3o98QJ1NJ8wAldkjBBJv8Eken4N3Rlrr5OgLxsq6PE%3D" --
recursive
```



Switch to command mode by pressing the **ESC** key.

Press the **:** (colon) key to open the prompt bar in the bottom left corner of the window.

Type **x** after the colon and press the **Enter** key to save changes and exit.

Back in the Azure Cloud Shell, update and execute the following command to grant permissions to the new script:

```
chmod +x /home/rchapler/script.sh
```

Confirm Success

In the Cloud Shell, update and paste the following command line.

```
/home/rchapler/script.sh
```

Successful processing of the new script will produce a response like:

```
INFO: Any empty folders will not be processed, because source and/or destination doesn't have full
folder support

Job cb947969-d113-624c-6ac6-56274d66baac has started
Log file is located at: /home/rchapler/.azcopy/cb947969-d113-624c-6ac6-56274d66baac.log

4 Files Scanned at Source, 0 Files Scanned at Destination

Job cb947969-d113-624c-6ac6-56274d66baac Summary
Files Scanned at Source: 4
Files Scanned at Destination: 4
Elapsed Time (Minutes): 0.0334
Number of Copy Transfers for Files: 0
Number of Copy Transfers for Folder Properties: 0
Total Number Of Copy Transfers: 0
Number of Copy Transfers Completed: 0
Number of Copy Transfers Failed: 0
Number of Deletions at Destination: 0
Total Number of Bytes Transferred: 0
Total Number of Bytes Enumerated: 0
Final Job Status: Completed
```

Schedule Cron Job

In this section, we will schedule a cron job and confirm success.

Schedule Cron Job

In the Cloud Shell, execute the following command to **open the cron table** {i.e., job scheduler}:

```
crontab -e
```

In Crontab, update and paste the following command line.

```
* * * * * /home/rchapler/script.sh
```

(Partial) logic explanation:

- **Schedule** … "* * * * *" (in this case, "every minute")
- **Command** … "/home/rchapler/script.sh"

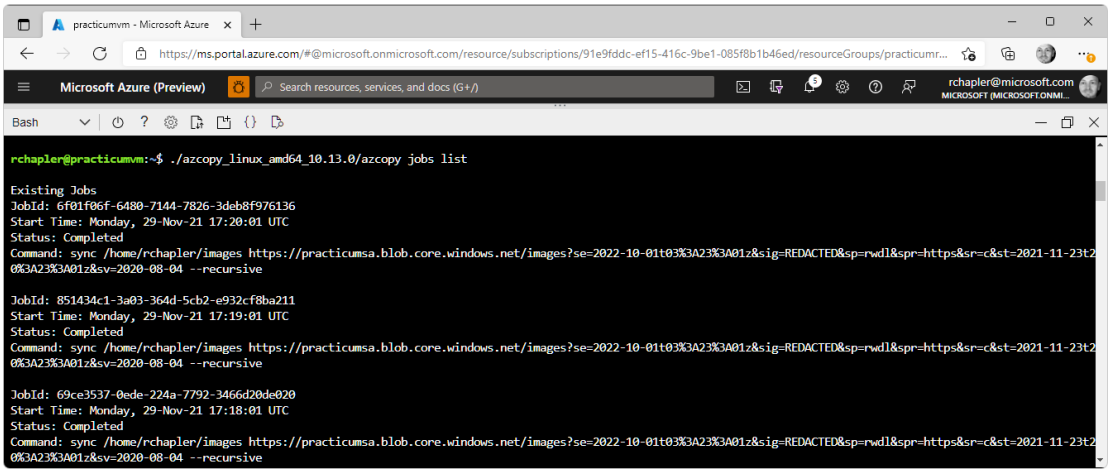
Press CTRL-X on the keyboard to exit crontab; save when prompted.

Back in the Cloud Shell, you should see the message "crontab: installing new crontab"

Confirm Success

In the Cloud Shell, update and execute the following command line to review recent runs of AzCopy.

```
./azcopy_linux_amd64_10.13.0/azcopy jobs list
```



More recent executions are shown first {i.e., you will want to scroll the top of the results to see the most recent execution}.

Confirm Synchronization

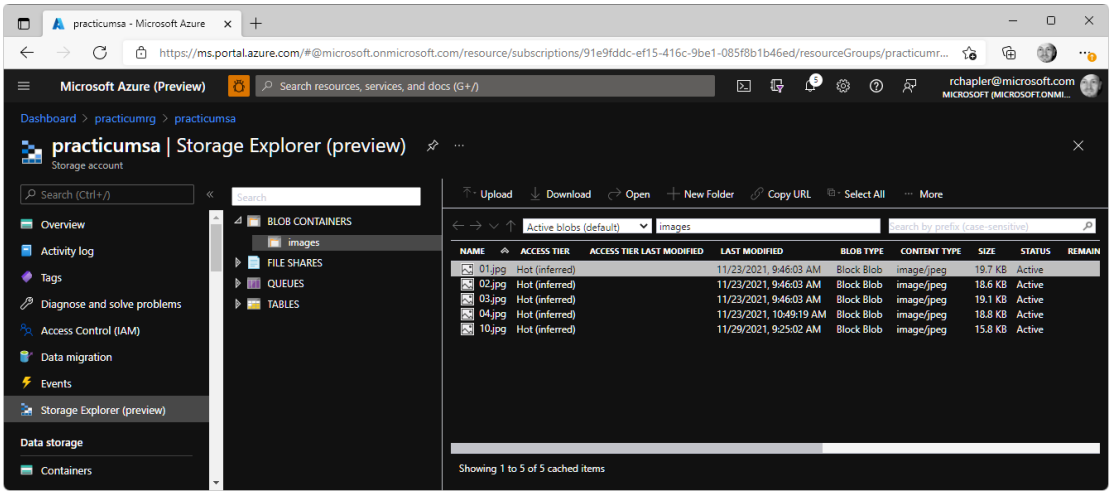
Execute the following command to get a new sample image from the traffic cam source:

```
wget -O images/10.jpg https://fl511.com/map/cctv/1139--9
```

You can expect a response like:

```
--2021-11-29 17:24:45-- https://fl511.com/map/cctv/1139--9
Resolving fl511.com (fl511.com)... 172.67.4.44, 104.22.25.76, 104.22.24.76, ...
Connecting to fl511.com (fl511.com)|172.67.4.44|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16149 (16K) [image/jpeg]
Saving to: ‘images/10.jpg’

images/10.jpg
100%[=======================================================================================>] 15.77K --.-
KB/s  in 0.06s

2021-11-29 17:24:45 (251 KB/s) - ‘images/10.jpg’ saved [16149/16149]
```

After waiting a minute for the scheduled synchronization, navigate to your storage account, and then **Storage Explorer**.



You should see the new file {i.e., “10.jpg”} included in the **images** container.

On-Prem to On-Prem

Follow these instructions to migrate files using Azure File Sync and a storage account.

Use Case

Work with your customer to understand and document requirements; example notes:

- Implementing a third-party Content Management System (CMS)
- Must migrate **petabytes of unstructured data**, including media, image, and Portable Document Format (PDF) files
- **Source** server is an **on-prem** server, and **destination** server is a **virtual machine**
- It is critical that **file metadata remains unchanged** during migration {e.g., Created On datetime}

Prerequisites

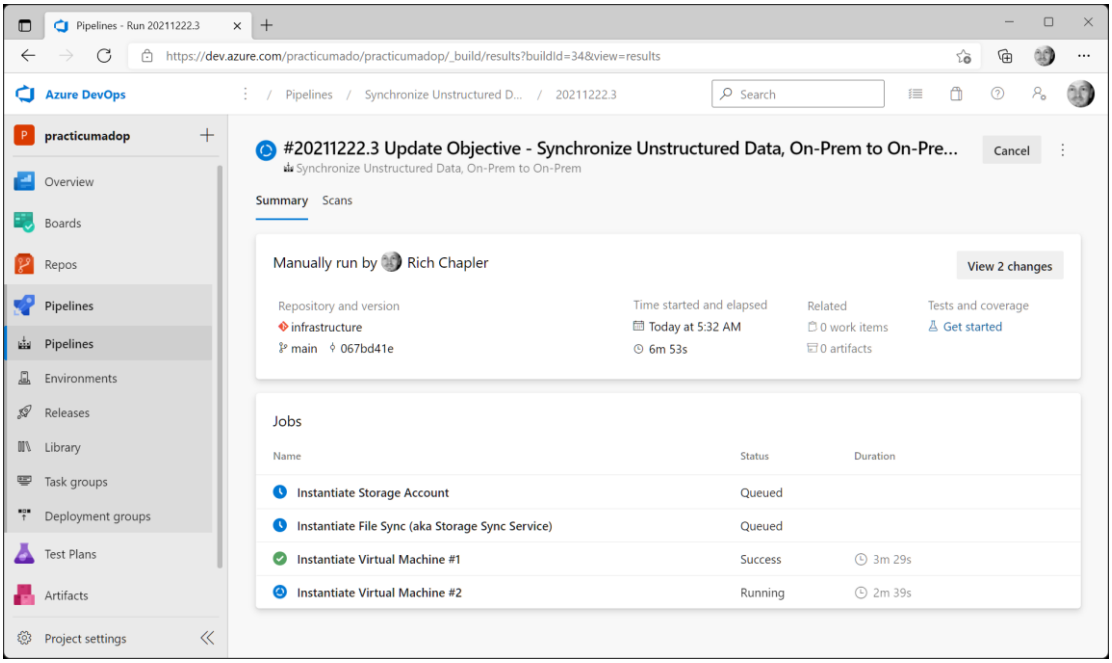
This solution requires the following resources:

- File Sync (aka Storage Sync Services)
- Storage Account
- Virtual Machine 1… to mimic the customers' source server (on-prem)
- Virtual Machine 2… to mimic the customers' destination server (virtual machine)

Follow the instructions in the "Infrastructure-as-Code " section to create templates, create a pipeline using the following YAML, and then deploy necessary pre-requisites:

```
trigger:
 branches:
  include:
  - refs/heads/main
 paths:
  include:
  - ARMTemplates
jobs:
- job: sa
 displayName: 'Instantiate Storage Account'
 pool:
  vmImage: windows-latest
 steps:
 - task: AzureResourceManagerTemplateDeployment@3
  inputs:
   ConnectedServiceName: a775d3ec-e4f1-46ed-905b-ddd26a59356c
   subscriptionName: 91e9fddc-ef15-416c-9be1-085f8b1b46ed
   resourceGroupName: <UseCase>
   location: West US
   csmFile: ARM Templates/StorageAccount.json
- job: sss
 displayName: 'Instantiate File Sync (aka Storage Sync Service)'
 pool:
  vmImage: windows-latest
 steps:
 - task: AzureResourceManagerTemplateDeployment@3
  inputs:
   ConnectedServiceName: a775d3ec-e4f1-46ed-905b-ddd26a59356c
   subscriptionName: 91e9fddc-ef15-416c-9be1-085f8b1b46ed
   resourceGroupName: <UseCase>
```

```
    location: West US
    csmFile: ARM Templates/FileSync.json
- job: vm1
 displayName: 'Instantiate Virtual Machine #1'
 pool:
  vmImage: windows-latest
 steps:
 - task: AzureResourceManagerTemplateDeployment@3
  displayName: 'ARM Template deployment: <UseCase>vm01'
  inputs:
   ConnectedServiceName: a775d3ec-e4f1-46ed-905b-ddd26a59356c
   subscriptionName: 91e9fddc-ef15-416c-9be1-085f8b1b46ed
   resourceGroupName: <UseCase>vm01
   location: West US
   csmFile: ARM Templates/VirtualMachine.json
- job: vm2
 displayName: 'Instantiate Virtual Machine #2'
 pool:
  vmImage: windows-latest
 steps:
 - task: AzureResourceManagerTemplateDeployment@3
  displayName: 'ARM Template deployment: <UseCase>vm02'
  inputs:
   ConnectedServiceName: a775d3ec-e4f1-46ed-905b-ddd26a59356c
   subscriptionName: 91e9fddc-ef15-416c-9be1-085f8b1b46ed
   resourceGroupName: <UseCase>vm02
   location: West US
   csmFile: ARM Templates/VirtualMachine.json
```



Install Agent

Navigate to the first virtual machine, click the **Connect** button, and select **RDP** from the resulting drop-down menu.

On the resulting "**Connect with RDP**" form, click the "**Download RDP File**" button.

Click the "**Open file**" link under your RDP file in the **Downloads** drop-down menu.

Click the **Connect** button on the resulting "**Remote Desktop Connection**" pop-up.

In the "**Windows Security**" pop-up, click the "**More choices**" link, then "**Use a different account**" in the resulting choices, then click **OK**.

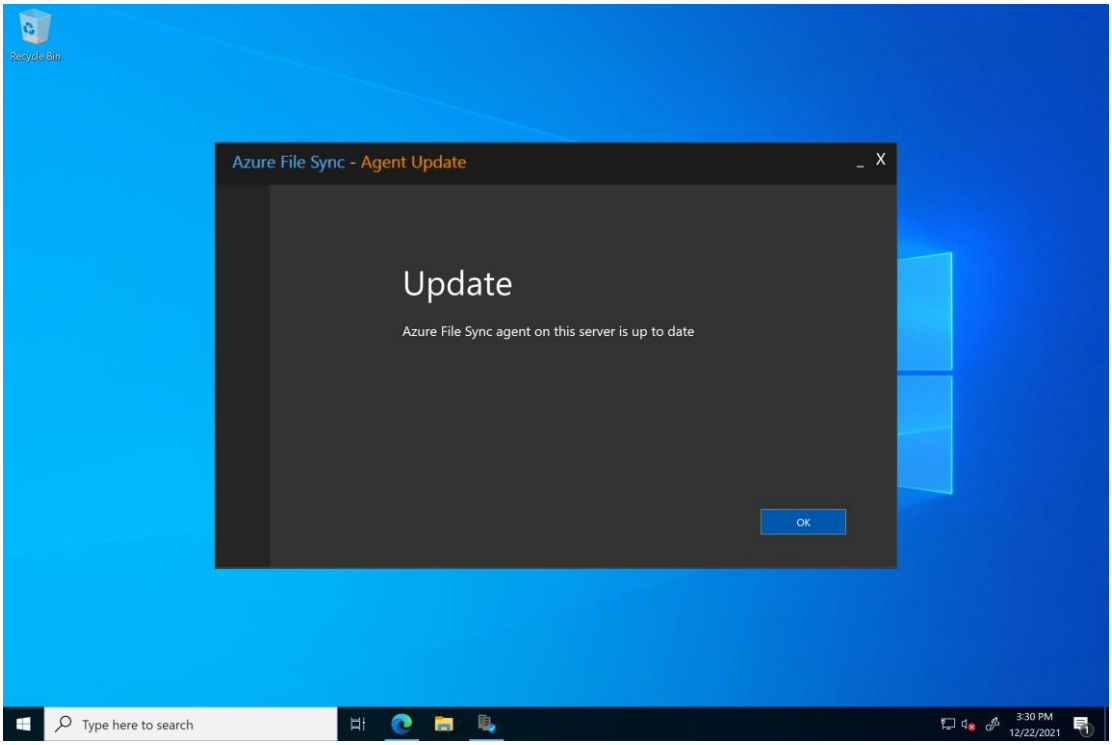
Back at the top of the "**Windows Security**" pop-up, enter credentials included in the YAML template.

Note: "User name" should include the name of the server {e.g., "<UseCase>vm01\Admin123"}.

Once connected to the virtual machine, browse to [Download Azure File Sync Agent from Official Microsoft Download Center](#).

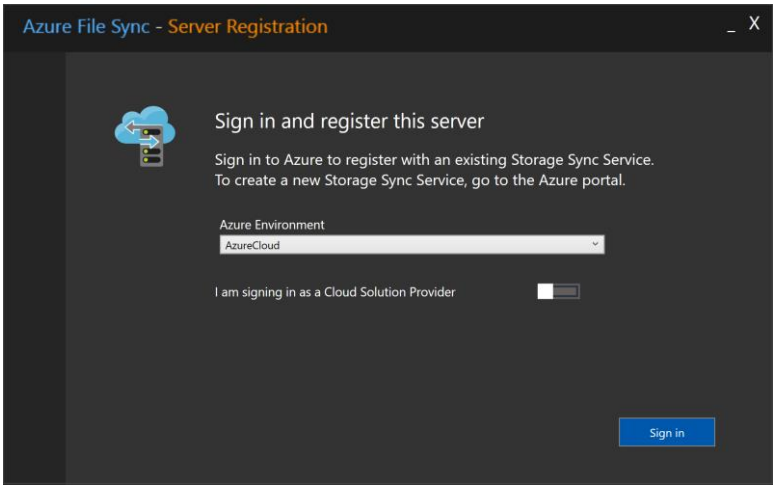
Click the **Download** button, and select the appropriate download {e.g., **StorageSyncAgent_WS2022.msi**} and click the **Next** button.

Open the downloaded file and complete the installation.



The Azure File Sync, server registration utility will open.

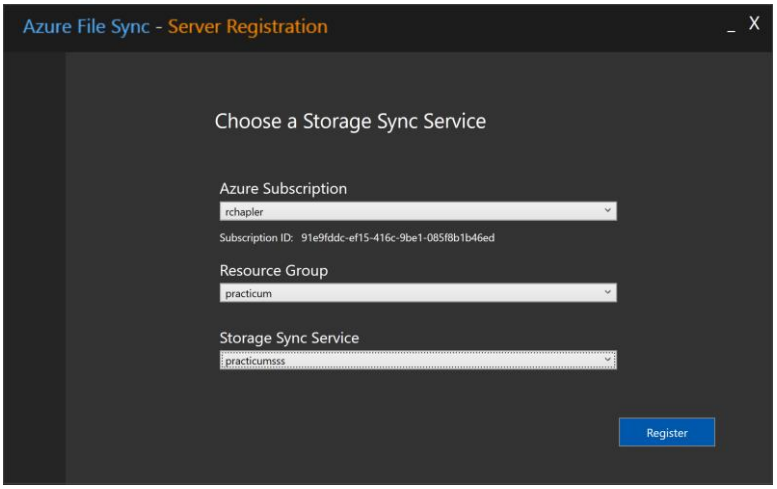
Click the OK button on the Update page.



Confirm "**Azure Environment**" selection and click the "**Sign in**" button on the "**Sign in and register this server**" page.

Complete the sign in process.

Note: You will likely have to add trusted sites and repeat the sign in process since this is a new, unconfigured server.



Complete the "**Choose a Storage Sync Service**" form, including:

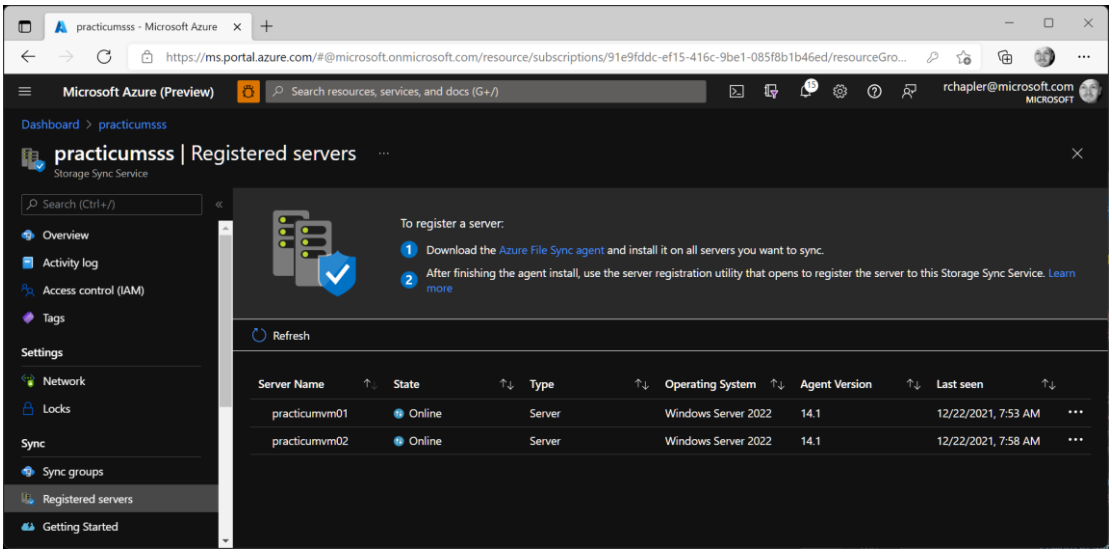
Storage Sync Service	Select your storage sync service

Click the Register button and allow time for the "Network connectivity test" to reach "Passed. You are all set."

Close (or minimize) the "**Remote Desktop Connection**" window.

Repeat this process for the second virtual machine.

Confirm Success



Navigate to Storage Sync Service, then "**Registered Servers**" in the **Sync** group of the navigation pane.

You should see your two registered servers.

Create File Share

Open your Storage Account, then "**File Shares**" in the "**Data storage**" group of the navigation pane.

Click the "**+ File share**" button and enter a meaningful **Name** value in the resulting pop-out.

Click the **Create** button.

Create Sync Group

Navigate to Storage Sync Service, then "**Registered Servers**" in the **Sync** group of the navigation pane.

Click the "**+ Sync group**" button.

Complete the "**Choose a Storage Sync Service**" form, including:

Storage Account	Click the "**Select storage account**" button and select your storage account on the resulting screen
Azure File Share	Select your storage account, file share

Click the **Create** button.

Add Server Endpoint

Click into the new Sync Group and then click the "**Add server endpoint**" button at the top of the screen.

On the resulting "**Add server endpoint**" pop-out, including:

Registered Server	Select your first virtual machine
Path	Enter "**C:\Temp**"

Click the **Create** button and then repeat the process for the second virtual machine.

Confirm Success

Connect to your first virtual machine and then open File Explorer.

Navigate to C:\Temp, right-click and select New > Bitmap Image in the resulting drop-down menus.

Take note of the metadata {e.g., **Created** datetime}.

Connect to your second virtual machine and then open File Explorer.

You should see a synchronized copy of the “New Bitmap Image.bmp” file and metadata will match.

Repeat this exercise with a file update {e.g., rename} and a file delete.

Deployment

The following sections describe instantiation and deployment of the Azure resources necessary for code deployment.

Some Basics

Before we dive into full deployment of a data solution, some basics must be covered:

- Branches
- Pull Requests
- ARM Templates
- Schema Comparison

This solution requires the following resources:

- *Service Principal*
- *A code repository… DevOps (with project and repository) or GitHub (with repository)*
- *Synapse or Data Factory (with configured repository)*

Branches

Every developer should begin work on an assigned Task with the creation of a branch {e.g., "rchapler_12345" where "rchapler" refers to the developer alias and "12345" refers to the Task Identifier} based on the latest version of the code in the collaboration branch {e.g., "development"}.

Branches can be created using DevOps, GitHub, Synapse, or Data Factory. For this exercise, we will create a branch using Synapse.

Navigate to Synapse Studio, click the **Integrate** navigation icon, click the branch drop-down menu and select "**+ New branch**."

On the resulting "Create a new branch" pop-up, including:

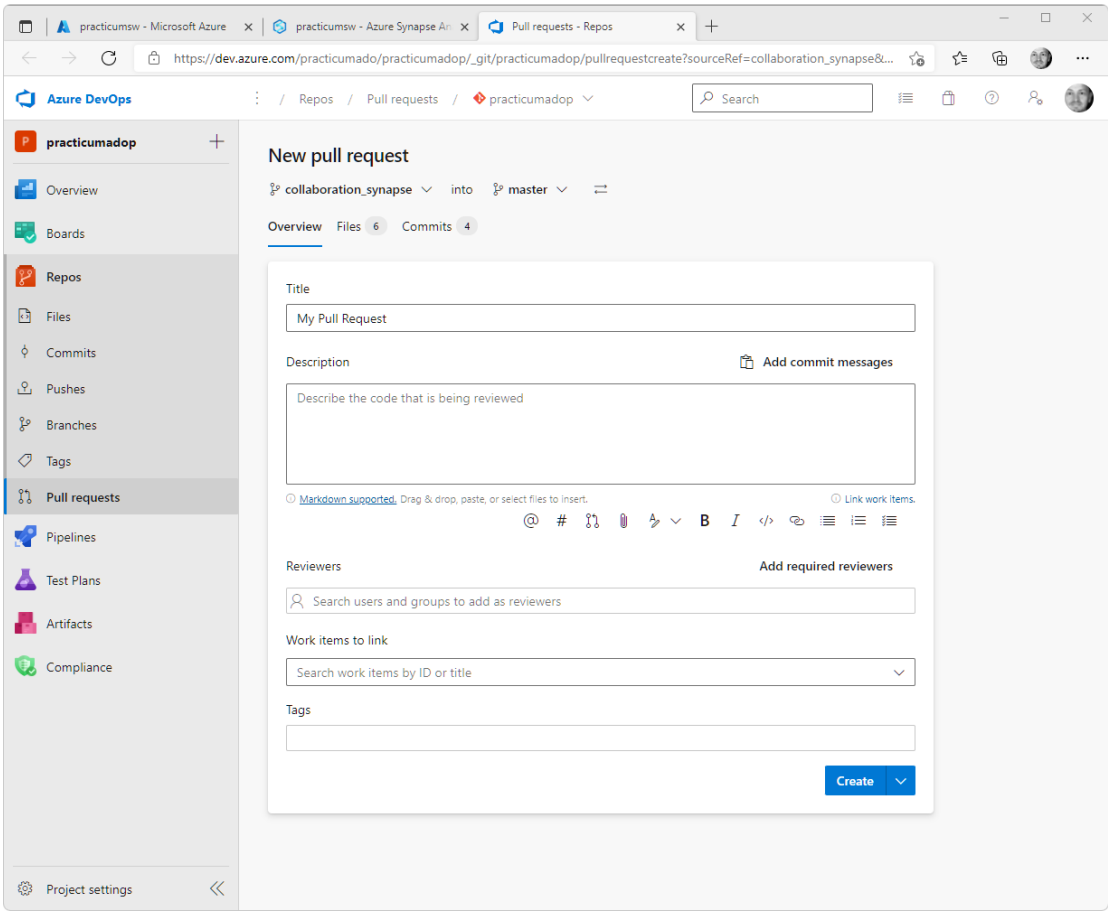
Branch Name	Consider a branch name that will inform the Development and Deployment Managers {e.g., begin with an email alias like "rchapler", include a timestamp like "20220601", and maybe include a Task Identifier like "12345"}
Base On	Select the appropriate starting point for work on the assigned task; in most cases Developers will select the collaboration branch

Click the **Create** button.

To follow the subsequent instructions for creating a Pull Request, consider making a minor change to the code in your branch.

Pull Requests

Navigate to Synapse Studio, click the **Integrate** navigation icon, click the branch drop-down menu and select "**Create pull request**." A third browser tab (pointing to https://dev.azure.com...) will open.

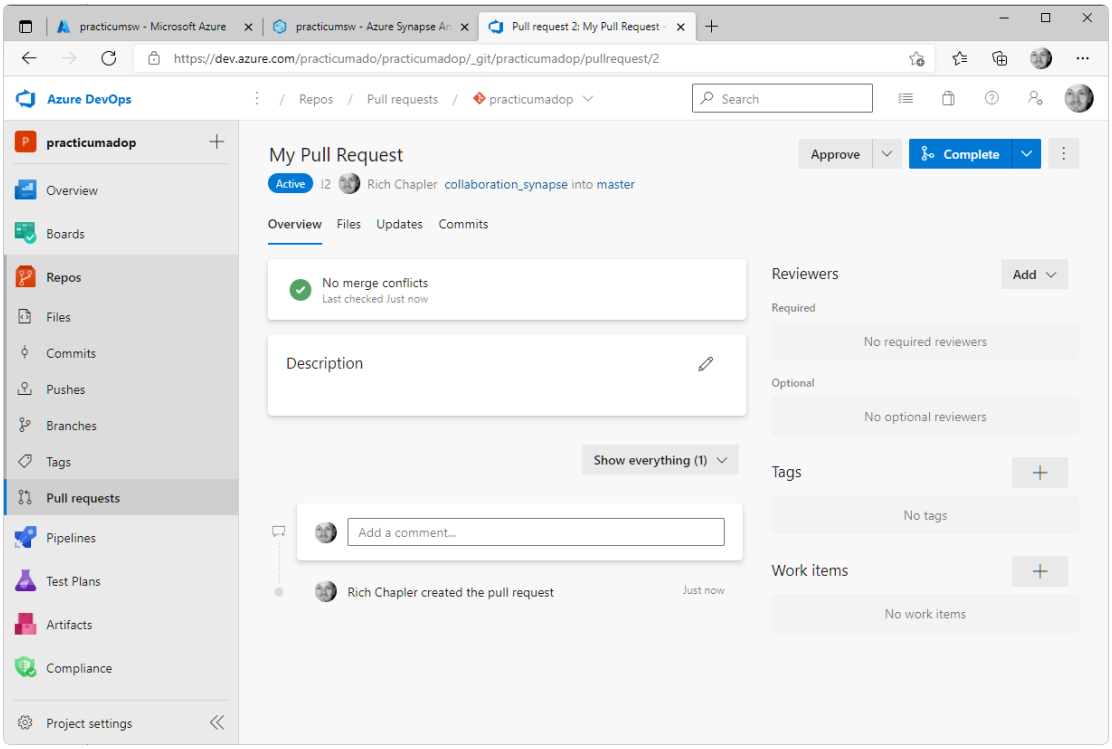


On the second "**New pull request**" form, including:

Branches	{**collaboration** branch} into {**master** branch} … you may have to create a master branch if one does not exist
Title	Enter a title that is meaningful for you (and aligned with your naming standards)

Including items like **Reviewers** and "**Work items**…" as appropriate.

Click the **Create** button.

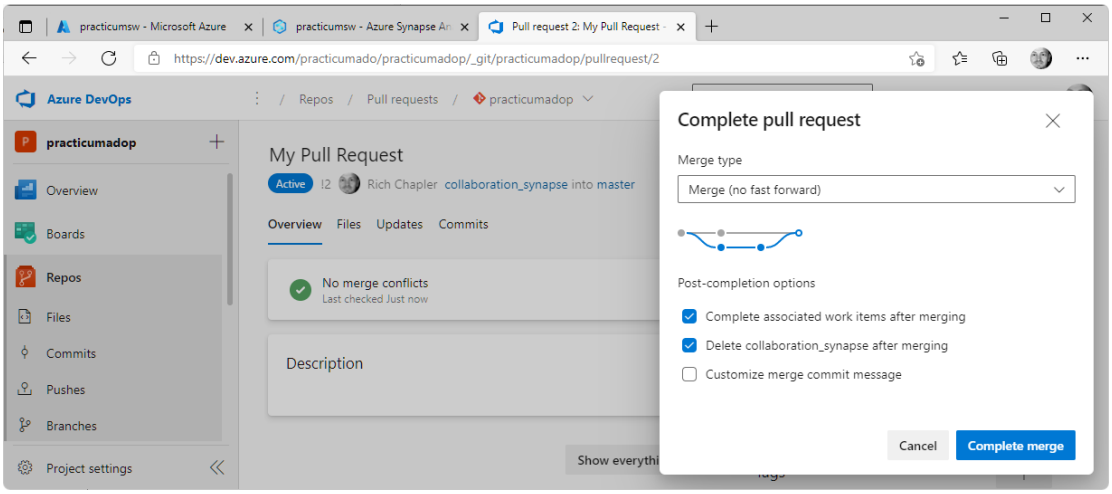


At this point (and if specified), designated reviewers would be notified by email, and they will need to review and approve for the Pull Request to move forward.

Additional configuration can provide for validation of other gating criteria {e.g., inclusion of text patterns that might be secret like an account key}.

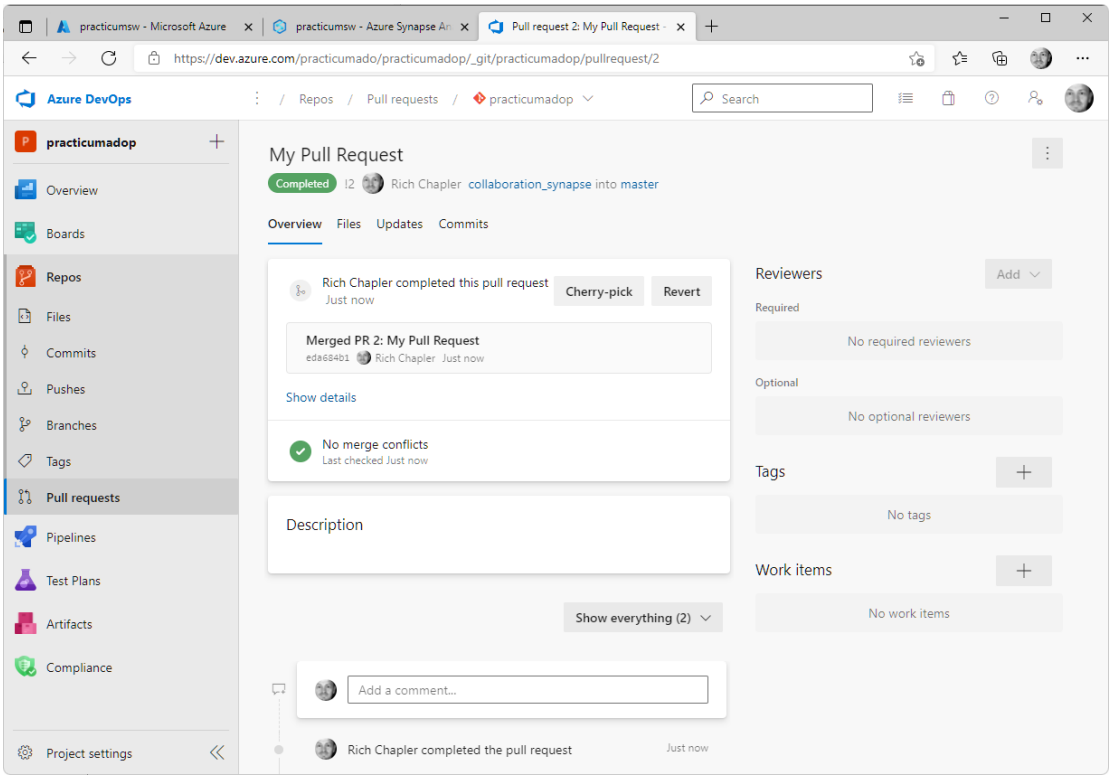
We have not included gating factors in this example.

Click the **Complete** button.



Confirm selections in the “**Complete pull request**” pop-out.

Click the “**Complete merge**” button.



Note: Post-completion options such as “Cherry-pick” and Revert provide for operational control even after a code merge.

ARM Templates

In this section, we will prepare a bare-minimum template, and then manually deploy that template.

Create Template

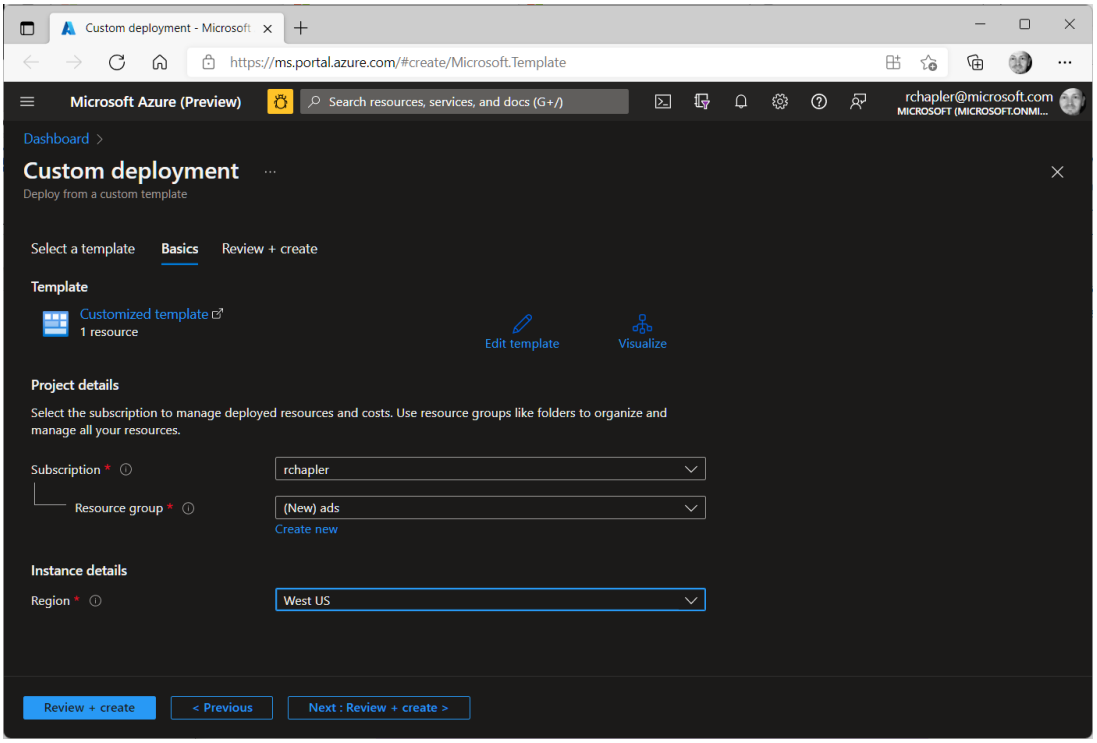
Navigate to https://ms.portal.azure.com/#create/Microsoft.Template and then click on the "**Build your own template in the editor**" link. On the resulting "**Edit template**" screen, replace the default JSON with the following prepared, "bare minimum" JSON:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "[providers('Microsoft.Storage','storageAccounts').apiVersions[0]]",
      "name": "[concat(resourceGroup().name,'sa')]",
      "location": "[resourceGroup().location]",
      "sku": { "name": "Standard_LRS" },
      "kind": "StorageV2"
    }
  ]
}
```

(Partial) logic explanation:

[concat…	Concatenation of the Resource Group name and acronym for the resource {i.e., "**sa**" for storage account}
Location	Use of "**[resourceGroup().location]**" ensures common location specification
AccountType	"**Standard_LRS**" is the cheapest option (and fine for demonstration) Every resource will have specific properties and codes best derived from online documentation

Review the pasted JSON and then click the **Save** button.



Complete the "**Custom deployment**" form, including:

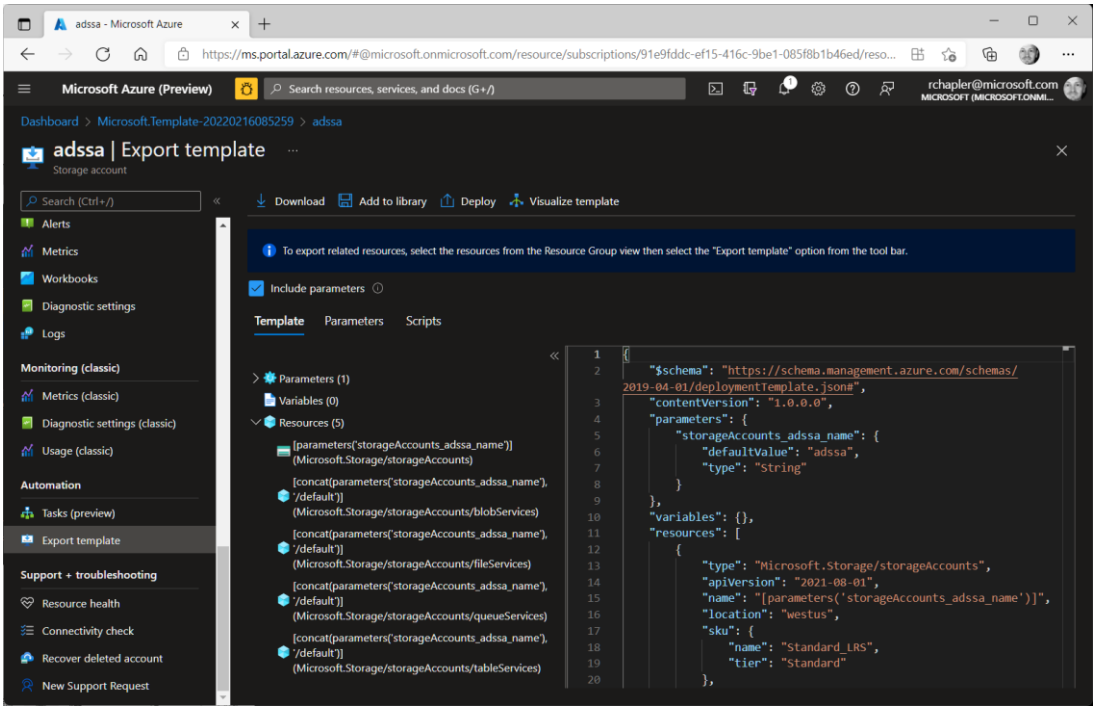
Resource Group	Select existing or "**Create new**"
Region	Select a region appropriate for your situation

Click the "**Review + create**" button, validate settings, and then click the **Create** button. When the deployment is complete, click the "**Go to resource**" button to review the created Storage Account.

Export Template

In this section, we will export the deployed template to see what Azure adds.

Navigate to the newly created Storage Account, and then click "**Export template**" in the **Automation** group of the navigation pane.



There is far more JSON than we included in the bare bones JSON!

Complete the following steps:

- Click the **Download** button and a file named "ExportedTemplate-ads.zip" will be written to the **Downloads** folder on your device
- Open this ZIP file and extract the "**template.json**" and "**parameters.json**" files
- Navigate to https://ms.portal.azure.com/#create/Microsoft.Template
- Click on the "**Build your own template in the editor**" link
- Click "**Load file**"
- Navigate to the "**template.json**" file and click the **Open** button

When you have successfully completed these steps, you will be able to review and re-use the exported template content.

Deploy Template

…via DevOps Pipeline

In this section, we will create, save, and run a DevOps Pipeline that deploys an ARM template.

Upload Template

Upload the ARM Template to the DevOps repo by completing the following steps:

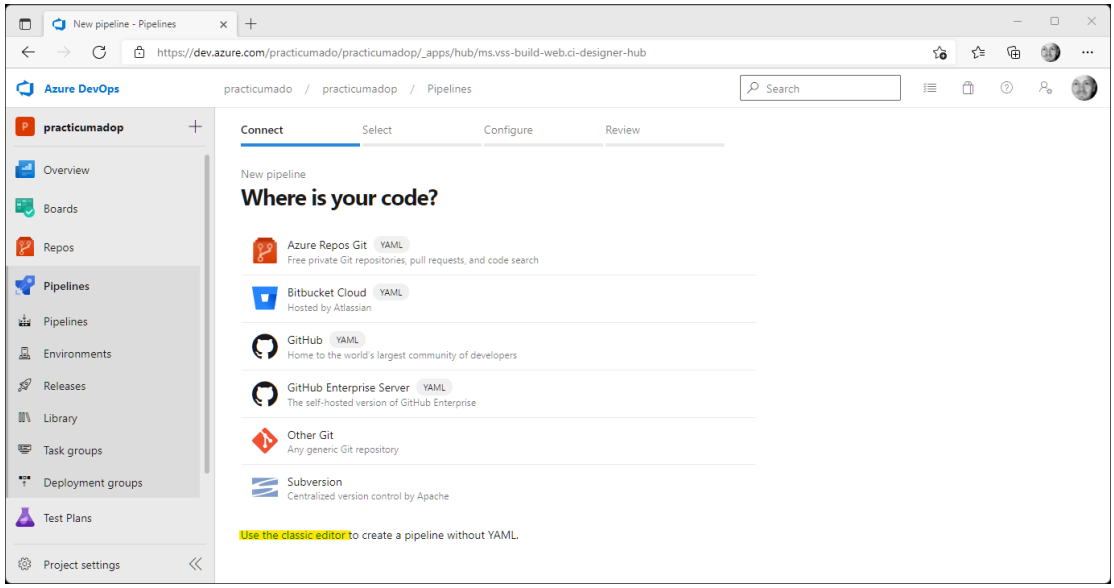
- Navigate to DevOps, select your project, and then click on **Repos** in the navigation pane
- Click the **+** icon just to the right of your project name and select "**New repository**" from the resulting drop-down menu
- In the resulting "**Create a repository**" pop-out, enter a "**Repository name**" {e.g., "infrastructure"} and then click the **Create** button
- Initialize the new repository with a ReadMe file; you can delete this file once additional files have been uploaded
- Click the vertical ellipses button in the upper-right of the screen, and then "**Upload file(s)**" in the resulting drop-down menu
- Drag-and-drop the **StorageAccount.json** template and click the **Commit** button

On successful completion, you can expect to receive an email reporting "**Build succeeded**."

Navigate to your Resource Group and confirm Storage Account creation.

Create Pipeline

Navigate to DevOps, click on **Pipelines** in the navigation pane, and then click the "**New Pipeline**" button.

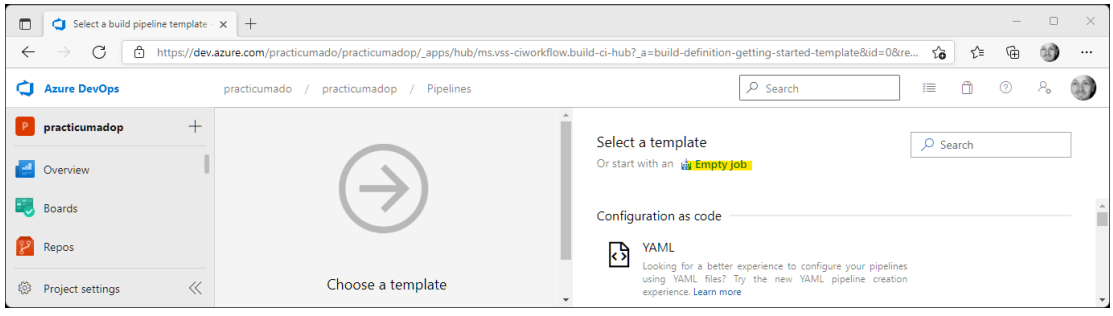


On the Connect tab {i.e., "New Pipeline … Where is your code?"} form, click the "**Use classic editor**" link.

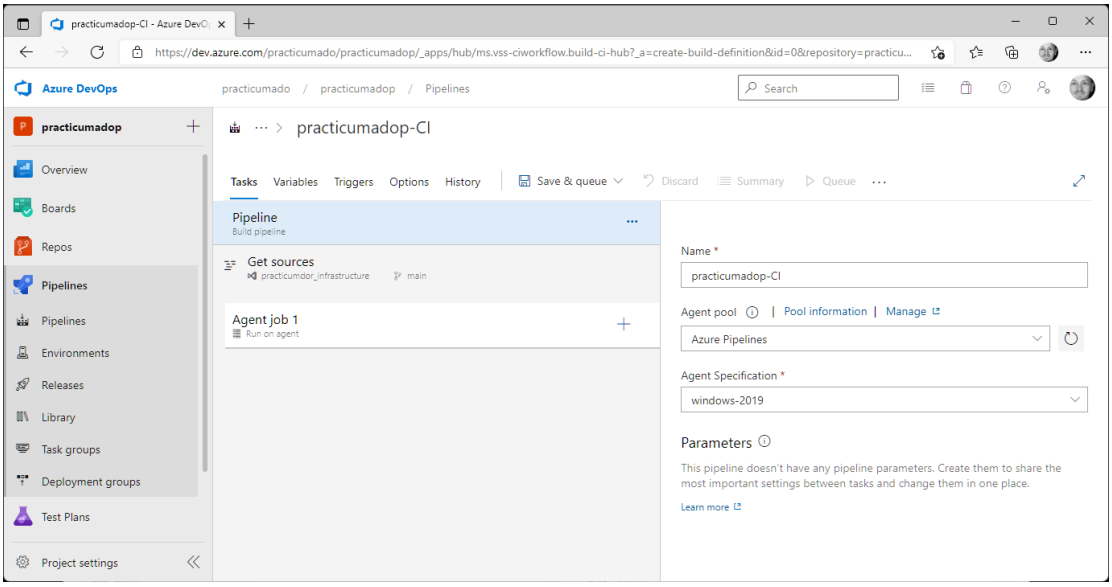
Complete the "**Select a source**" pop-out, select "**Azure Repos Git**" and then including:

Team Project	Select your DevOps project
Repository	Select your DevOps repository {e.g., **infrastructure**}
Default Branch…	Confirm default selection

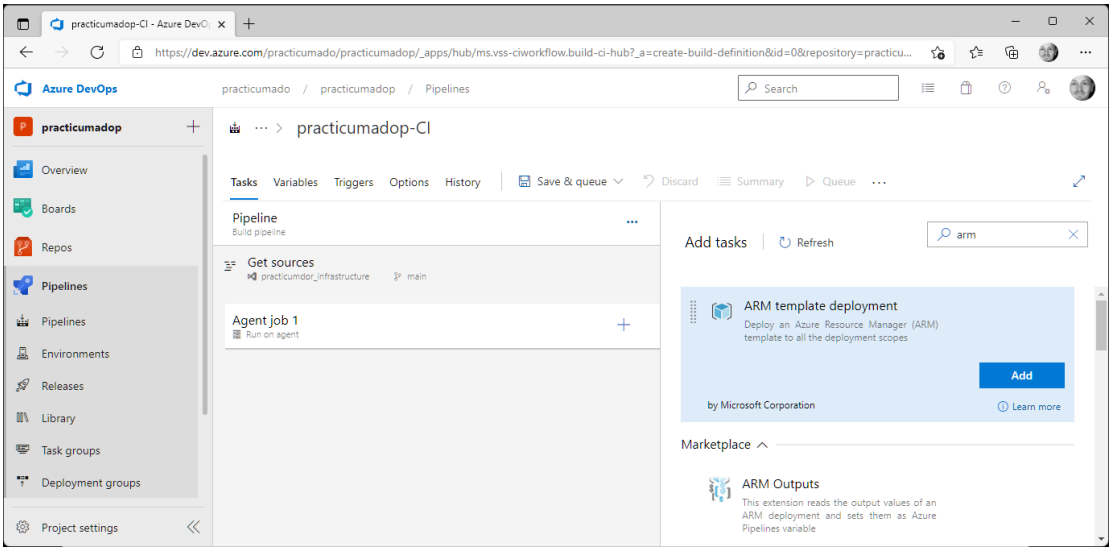
Click the **Continue** button.



Click “**Empty job**” on the “**Select a template**” pop-out.



Confirm default values for **Name**, **Agent Pool,** and **Agent Specification**.



Click the “**+**” button to the right of “**Agent Job 1**.”

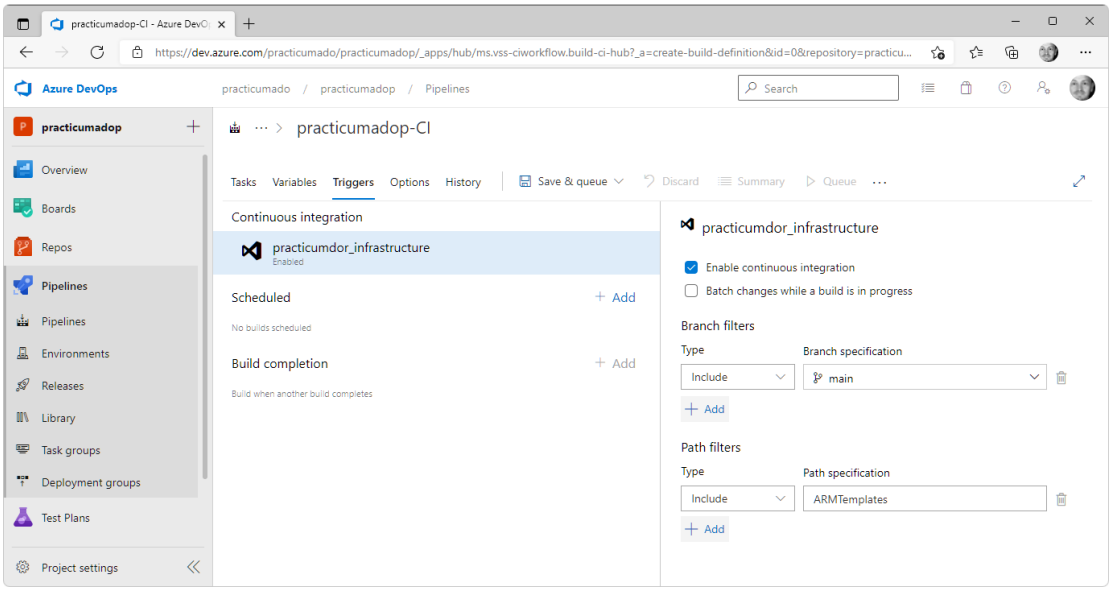
Search for and then select “**ARM template deployment**.”

Click the **Add** button.

Complete the “**ARM template deployment**” pop-out, including:

Deployment Scope	Confirm default, “**Resource Group**”
Action	Select “Create or update Resource Group”
Template	Browse to and select the “**StorageAccount.json**” template created in the prior section
Deployment Mode	Confirm selection, **Incremental**

Click on the **Triggers** tab.



Check the “**Enable continuous integration**” checkbox; new input controls will surface.

Branch Filter	Confirm defaults, Type: **Include** and Branch Specification: **Main**
Path Filter	Add Type: **Include** and Path Specification: “**ARMTemplates**”

Click on the “**Save & queue**” button, and then click “**Save & queue**” in the resulting drop-down menu.

Confirm values on the resulting “**Run pipeline**” pop-out, and then click the “**Save and run**” button.

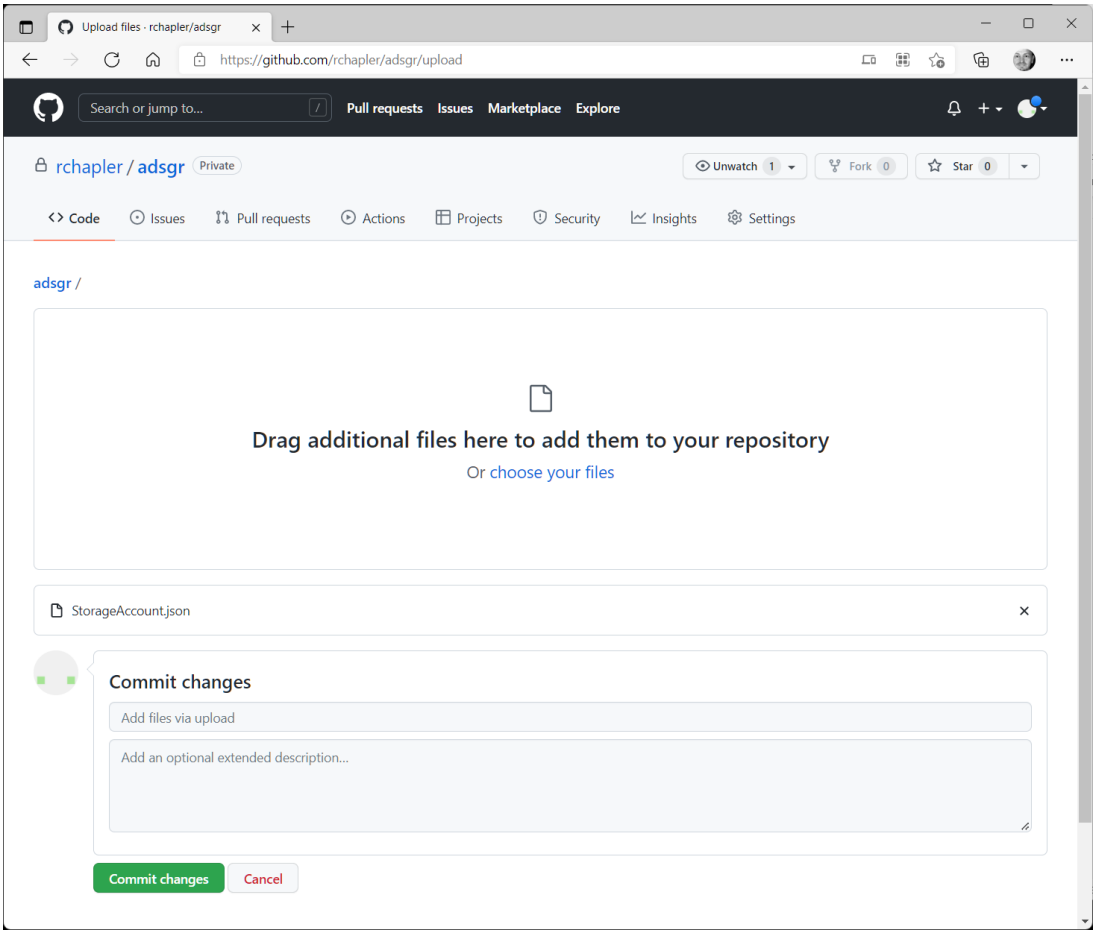
…via GitHub Action

In this section, we will create, save, and run a GitHub Action that deploys an ARM template.

Upload Template

Modify (with your values) and navigate to the following URL:

https://github.com/rchapler/<UseCase>gr/upload



Click the "**choose your files**" link, select the "StorageAccount.json" file created in Create Template, and finally, click the "**Commit changes**" button at the bottom of the page.

Prepare Credentials

Use a text editor to modify the following string:

```
{
"clientId": "{App GUID, aka Client Id}",
"clientSecret": "{Password String, aka Client Secret “rbac”}",
"subscriptionId": "{SubscriptionId GUID}",
"tenantId": "{TenantId GUID}"
}
```

You should use the values produced when you created the Service Principal.

Create Secret

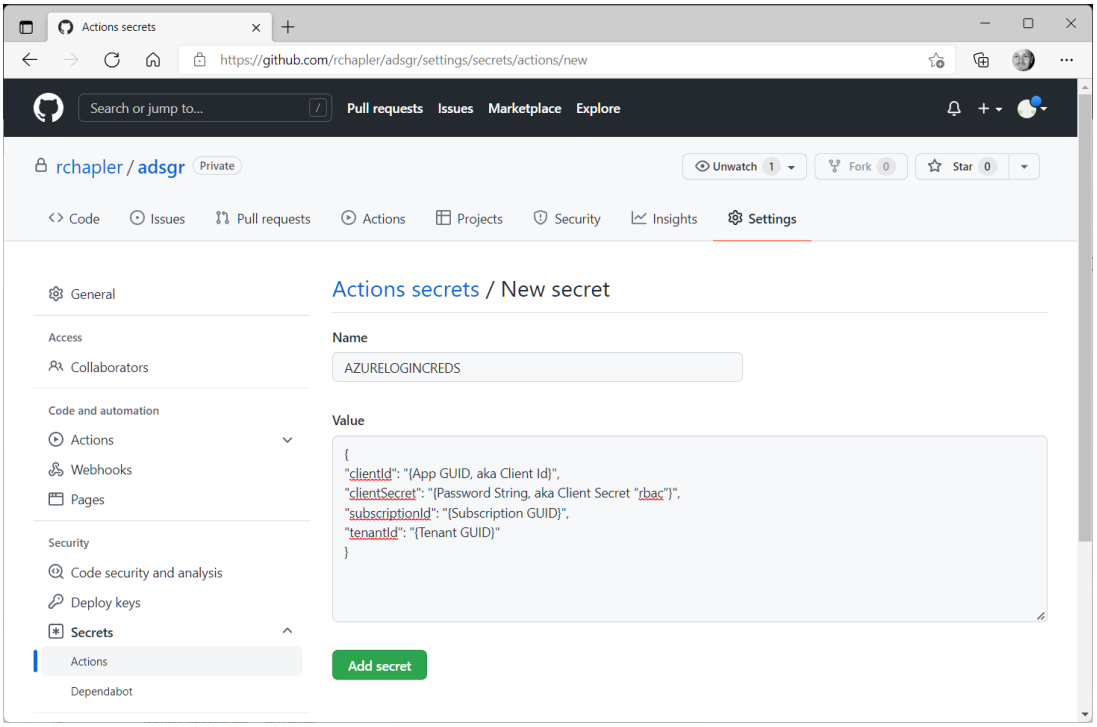
Modify and navigate to the following URL:

https://github.com/rchapler/shared/settings/secrets/actions.

Click the “**New repository secret**” button.

Complete the “**Actions secrets / new secret**” form, including:

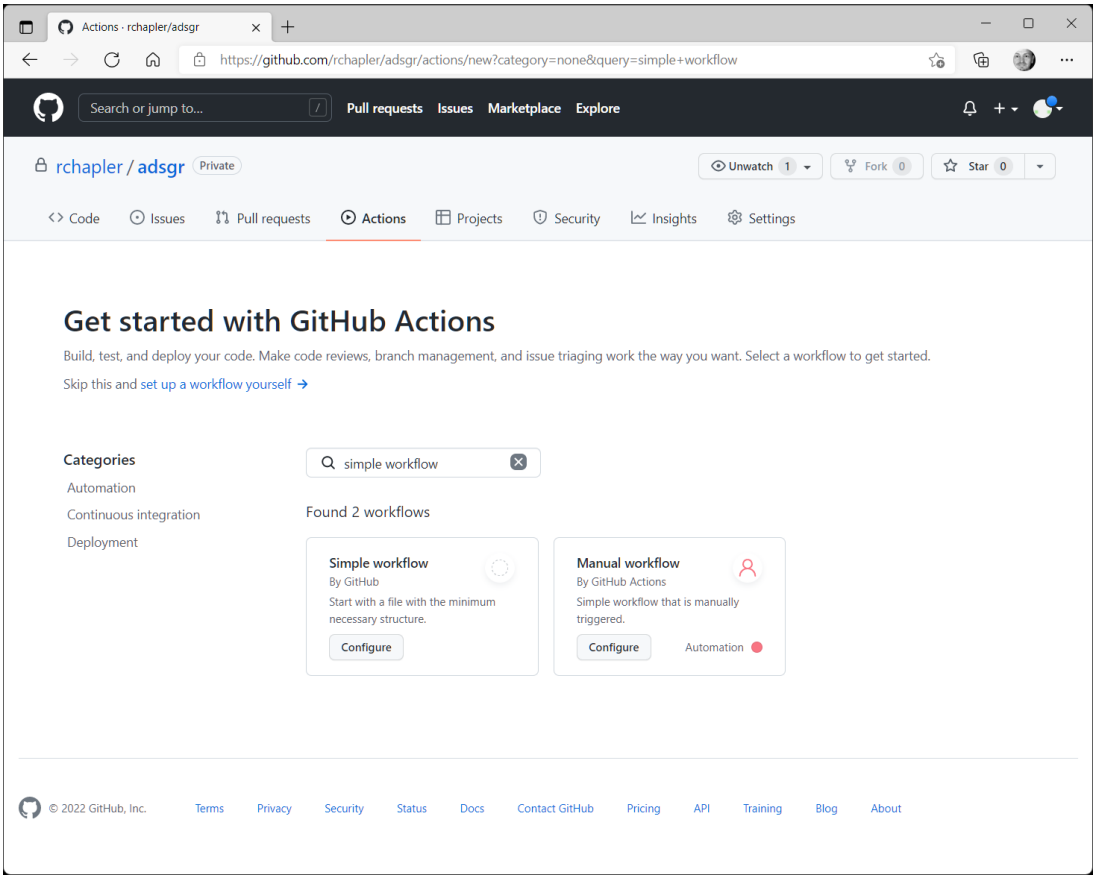
Value	Paste the credential string prepared in the prior step



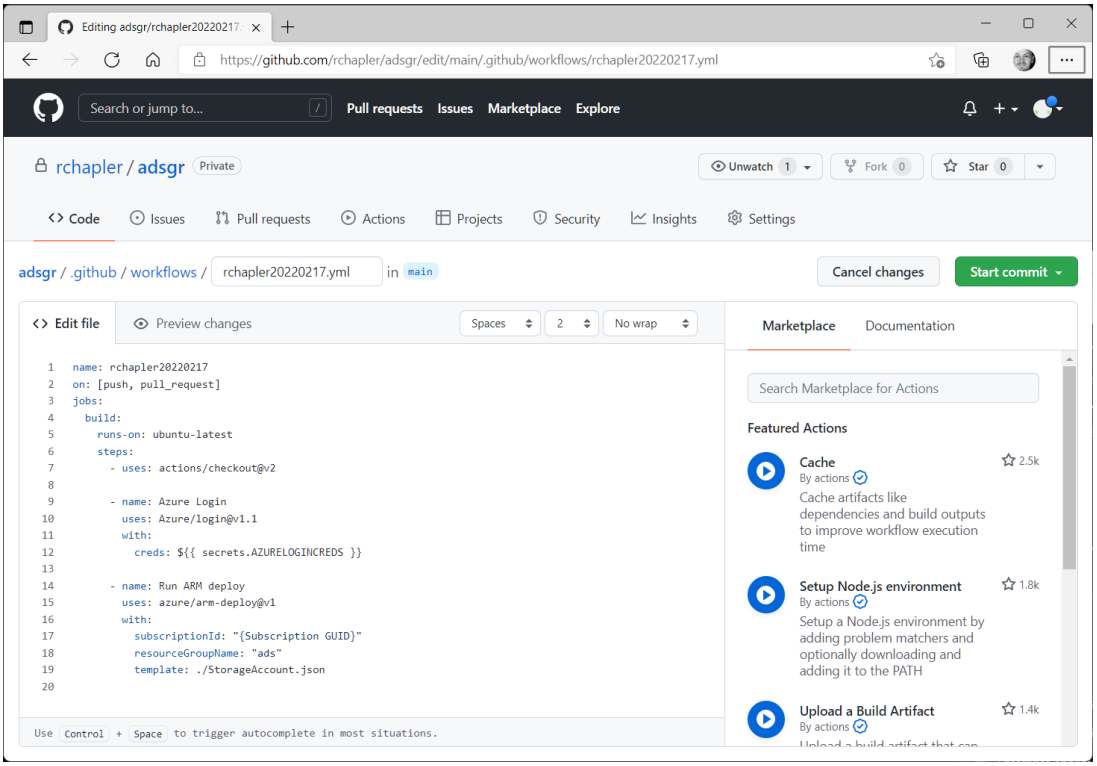
Click the “**Add secret**” button.

Repeat for “SUBSCRIPTIONID” secret.

Create Workflow



Click **Actions** in the menubar, search for "**Simple workflow**", and then click the **Configure** button.

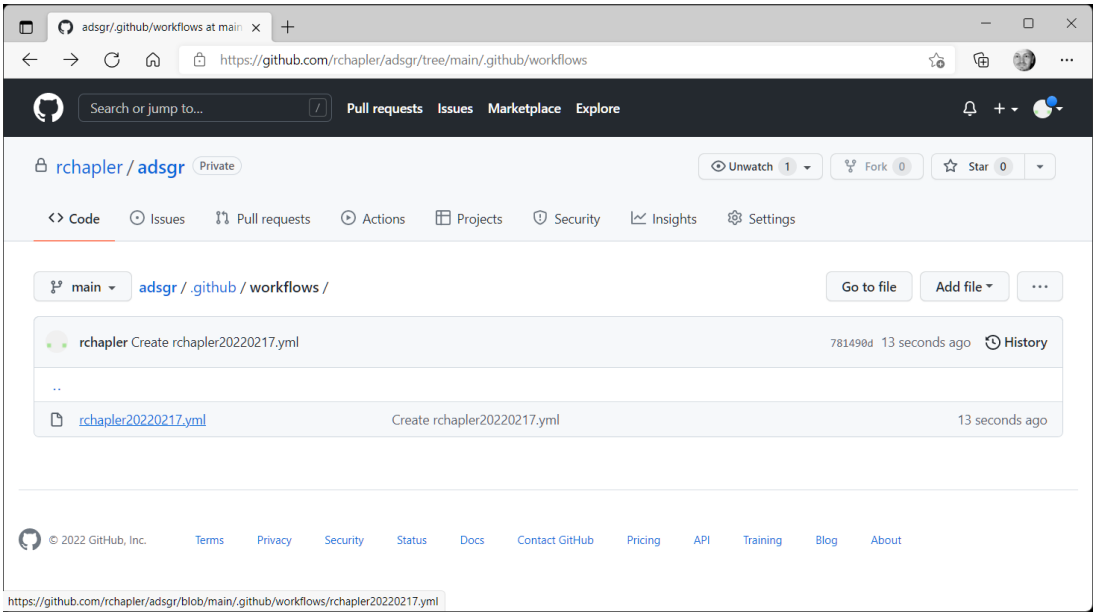


Enter a meaningful name, then modify and paste the following YAML.

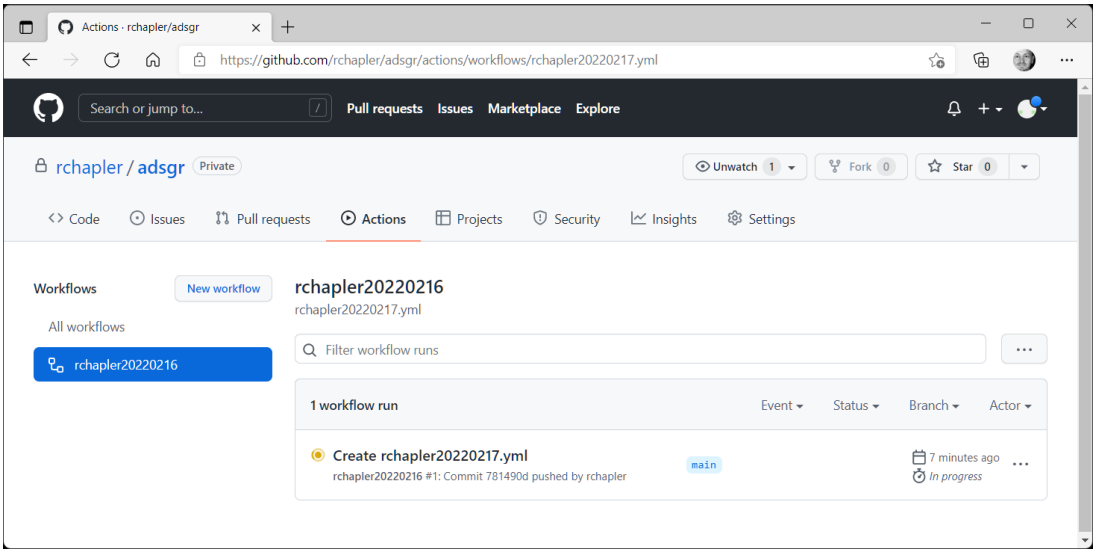
```
name: rchapler20220216
on: workflow_dispatch
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v2

   - name: AzureLogin
    uses: Azure/login@v1.1
    with:
     creds: ${{ secrets.AZURELOGINCREDS }}

   - name: DeployARM_StorageAccount
    uses: azure/arm-deploy@v1
    with:
     subscriptionId: "{Subscription GUID}"
     resourceGroupName: "rchapler"
     template: ./StorageAccount.json
```

*Note: Your implementation of GitHub Enterprise may not have a ready image for "**ubuntu-latest**" and require an update or other work-around.*

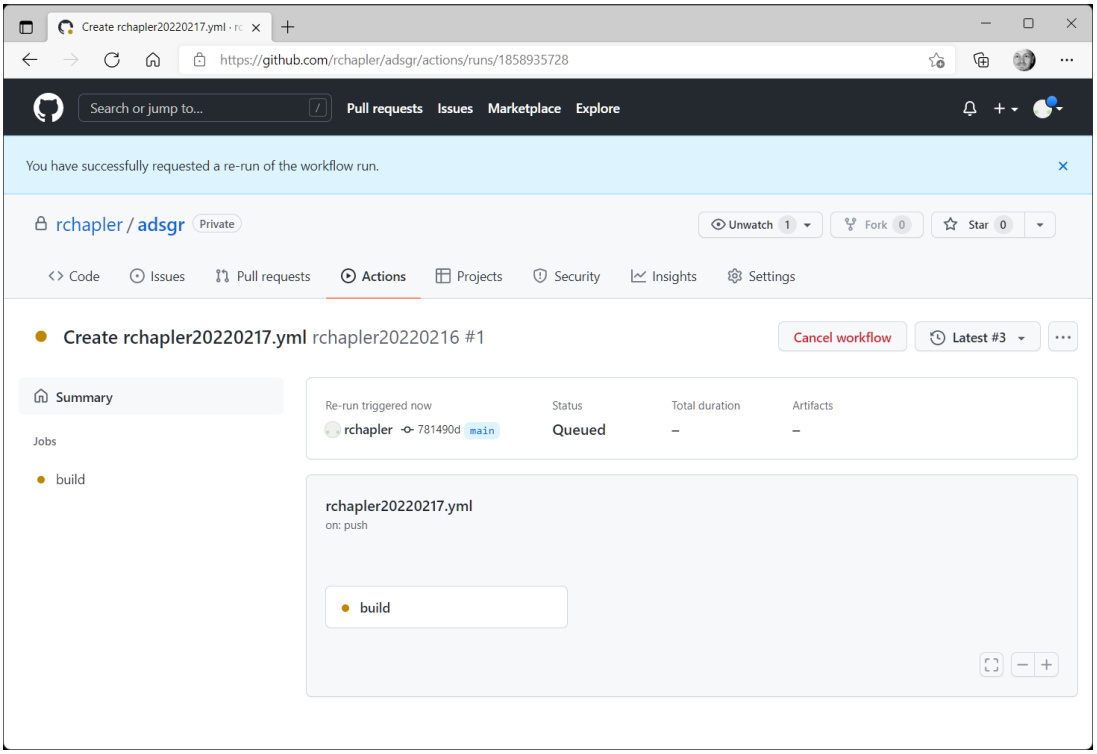
Click the "**Start commit**" button and then the "**Commit new file**" button on the resulting pop-up.



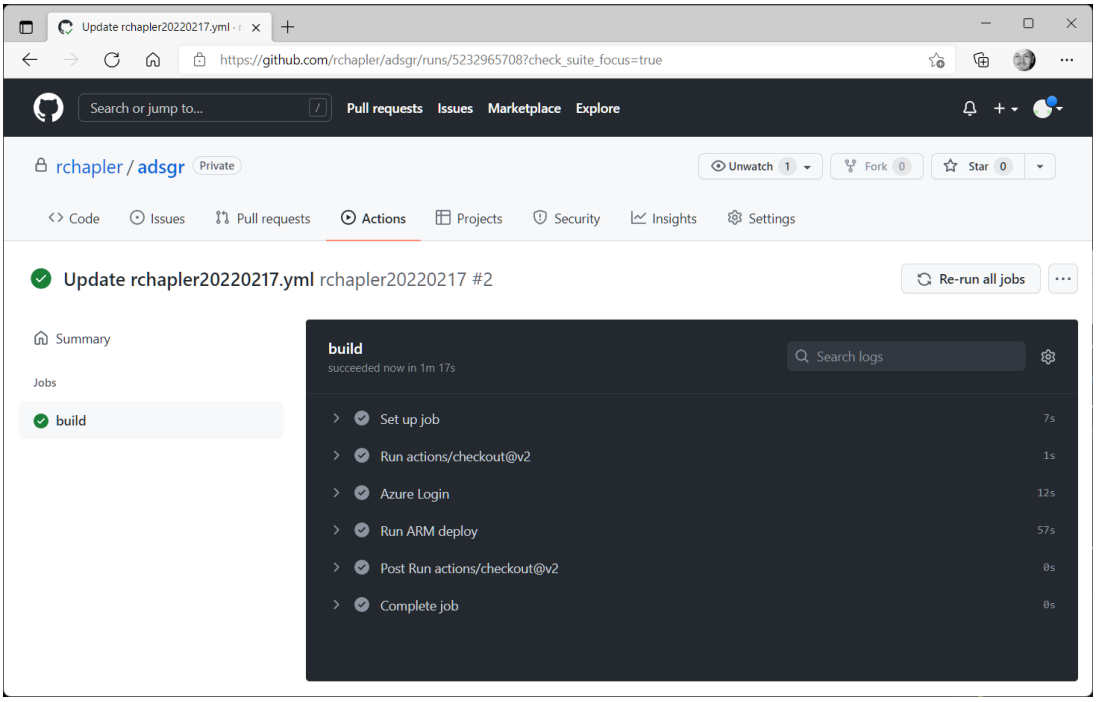
On the resulting form, click on the link for the committed YAML file.



Click on the run {i.e., “Create rchapler20220217.YAML”}.

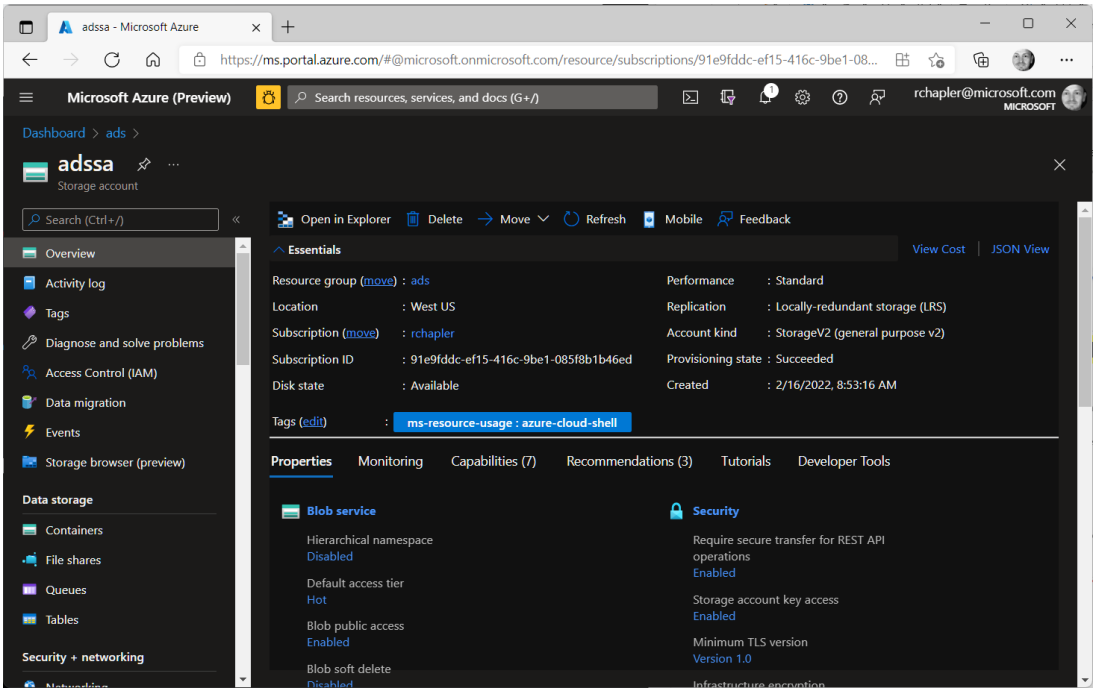


Click on the **build** box.



Confirm Success

We can confirm successful deployment (using either DevOps Pipeline or GitHub Action) simply by navigating to Azure and confirming that the Storage Account was created.



Schema Comparison

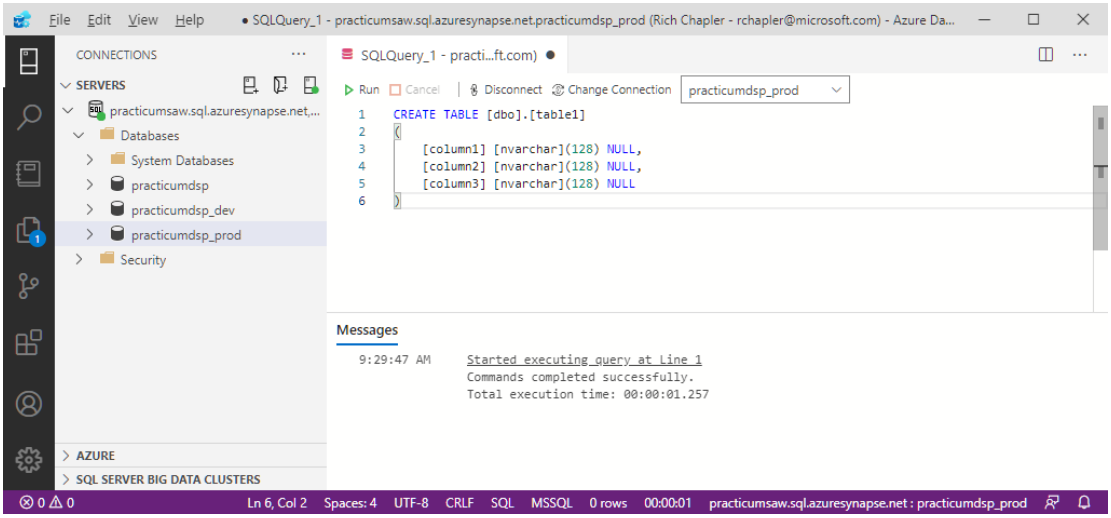
This solution requires the following resources:

- *Data Studio*
- *Synapse (with two SQL pool; e.g., <UseCase>dev and <UseCase>test)*

Although this example focuses on Synapse, the same solution applies to SQL.

Stage Resources

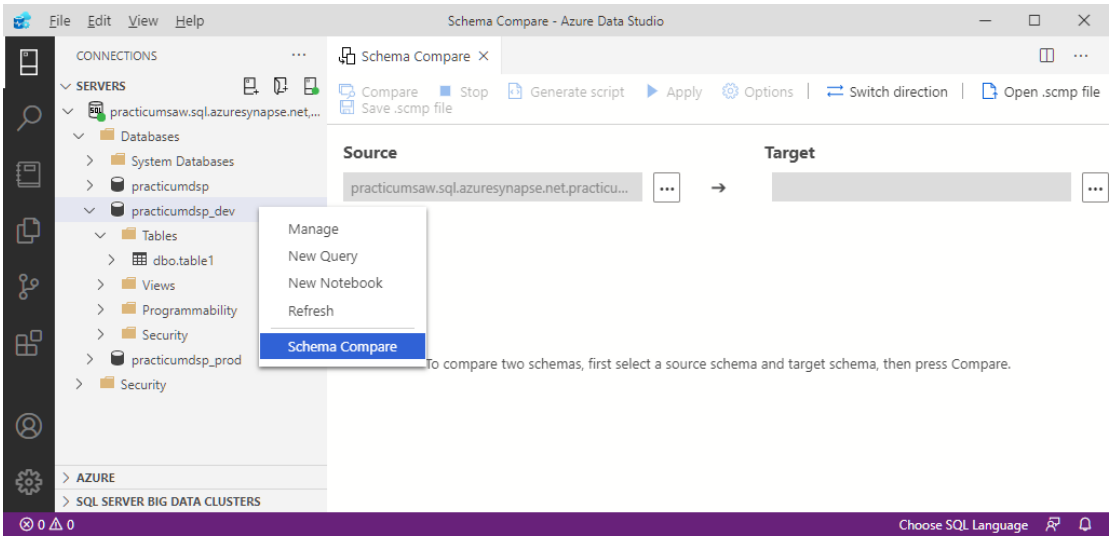
Open Azure Data Studio.



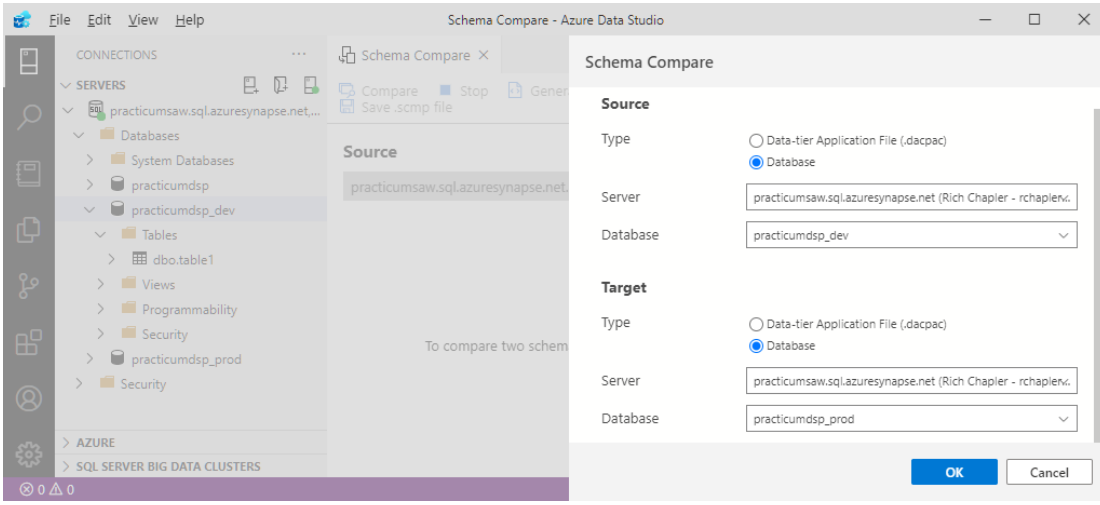
Execute the following T-SQL on <UseCase>dsp_dev:

```
CREATE TABLE [dbo].[table1]
(
        [column1] [nvarchar](128) NULL,
        [column2] [nvarchar](128) NULL,
        [column3] [nvarchar](128) NULL
)
```

Schema Compare



Right-click on the SQL Pool and click "**Schema Compare**" in the resulting drop-down menu.



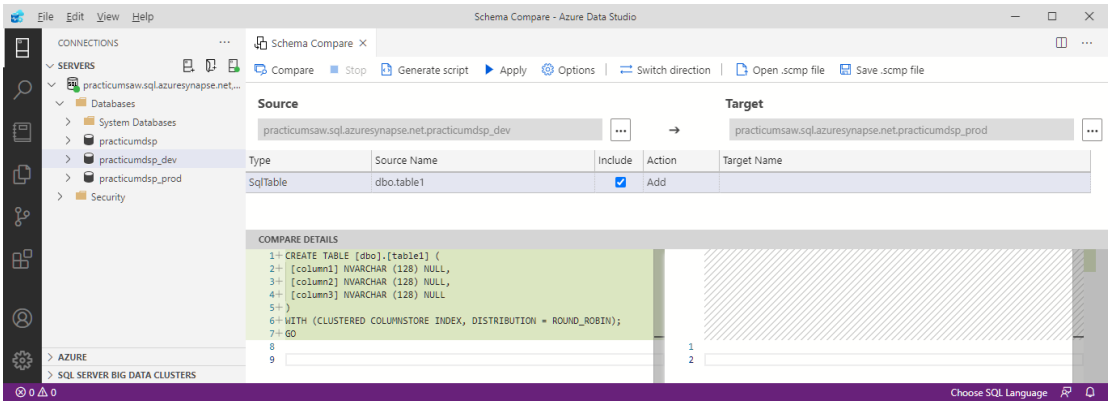
Click the ellipses button.

Complete the "**Schema Compare**" pop-out, including:

Source | **Type**	Select the **Database** radio button
Source | **Server** and **Database**	Confirm selection of your Synapse server and the **<UseCase>dsp_dev** database
Target | **Type**	Select the **Database** radio button

Target | **Server** and **Database**	Confirm selection of your Synapse server and the **<UseCase>dsp_prod** database

Click the **OK** button.

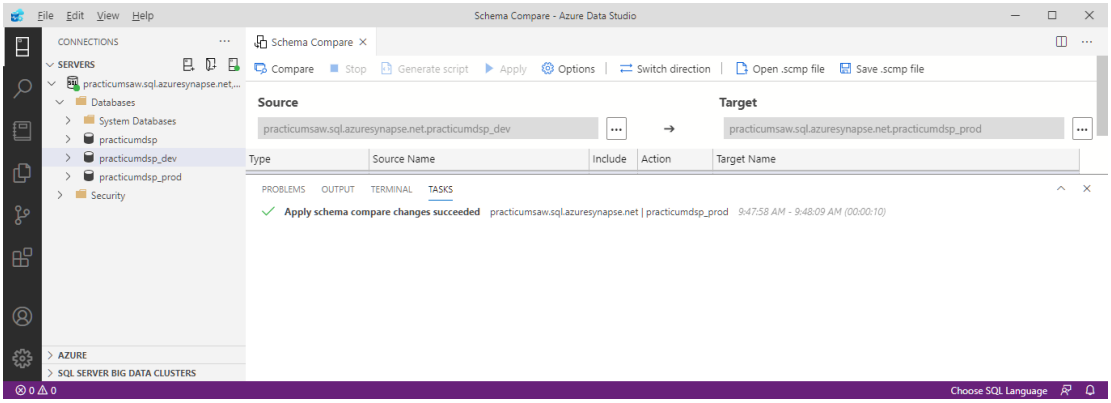


Complete the "**Schema Compare**" tab, click the **Compare** button.

We only created a table on the development database, so Schema Compare should surface that table as something missing from the production database.

Click on the row to see T-SQL for the identified item.

Click the **Apply** button to update the target.



Solution Deployment

Now that we have a grasp of the basics, we will discuss full data solution deployment.

Use Case

Example notes:

- "…deploy, **in measurable steps**, each of the core elements of our Azure Data Solution to the PROD environment"
- "…use **GitHub Actions** and establish a **repeatable template**"
- "…minimize code included in created templates"
- "…use programmatic rather than fixed values {e.g., the latest API version rather than a specific dated version} wherever possible"
- "…assume that all resources naming will follow the convention of {Resource Group name} + {acronym for resource type}"

Prerequisites

This solution requires the following resources:

- GitHub Enterprise (with repository and AZURELOGINCREDS secret)

Infrastructure

This section describes deployment and configuration of data solution infrastructure {e.g., resource group, data lake, synapse, etc.}. These tasks will be completed infrequently.

Resource Group

We will start by deploying a Resource Group because it is a **prerequisite for all other resources**.

Resource Groups must be deployed at the subscription-level {i.e., scope = “subscription”}.

Reference the detailed instructions in the “**Infrastructure-as-Code**” section, as required.

First, upload the following ARM Template JSON to GitHub (modified, as required).

```
{
    "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": { "ResourceGroupName": { "type": "string" } },
    "resources": [
        {
            "type": "Microsoft.Resources/resourceGroups",
            "apiVersion": "[providers('Microsoft.Resources','resourceGroups').apiVersions[0]]",
            "name": "[parameters('ResourceGroupName')]",
            "location": "eastus"
        }
    ]
}
```

Second, create a secret for RESOURCEGROUPNAME_TEST and …_PROD (where the suffix refers to the related environment). Repeat for SUBSCRIPTIONID_TEST and …PROD.

Third, create a workflow using the following YAML (modified, as required):

```
name: Deploy Resource Group
on: workflow_dispatch
jobs:
 build:
  runs-on: self-hosted
  steps:
   - uses: actions/checkout@v2

   - name: Login to Azure
     uses: azure/login@v1.1
     with:
       creds: ${{ secrets.AZURELOGINCREDS }}

   - name: Create Resource Group
     uses: azure/arm-deploy@v1
     with:
       scope: subscription
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       region: "eastus"
       template: ./arm/resourcegroup.json
       parameters:
         ResourceGroupName=${{ secrets.RESOURCEGROUPNAME_TEST }}
```

Fourth, run the workflow and monitor for success. When complete, confirm success by navigating to your TEST subscription and confirming that the Resource Group has been created.

Assuming success and when you are ready to proceed, repeat the process for the PROD.

Role Assignment

Repeat the process with the following ARM Template JSON:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "variables": {
        "RoleDefinitionId_Owner": "8e3af657-a8ff-443c-a75c-2fe8c4bcb635"
        , "RoleDefinitionId_Contributor": "b24988ac-6180-42a0-ab88-20f7382dd24c"
        , "ObjectId_GroupOrUser": "29757728-136d-4ab7-96ff-4f902f750975"
    },
    "resources": [
        {
            "type": "Microsoft.Authorization/roleAssignments",
            "apiVersion": "[providers('Microsoft.Authorization','roleAssignments').apiVersions[0]]",
            "name": "[guid(concat(variables('ObjectId_GroupOrUser'),'-
',variables('RoleDefinitionId_Contributor')))]",
            "properties": {
                "roleDefinitionId": "[resourceId('Microsoft.Authorization/roleDefinitions',
variables('RoleDefinitionId_Contributor'))]",
                "principalId": "[variables('ObjectId_GroupOrUser')]"
            }
        }
    ]
}
```

Notes:

- *The RoleDefinitionId for built-in role "Owner" is always 8e3af657-a8ff-443c-a75c-2fe8c4bcb635*
    - *…for role "Contributor" the guid is b24988ac-6180-42a0-ab88-20f7382dd24c*
- *This template assumes the existence of and GUID for a group or user {i.e., PrincipalId}*

…and workflow YAML:

```
name: Deploy Role Assignment
on: workflow_dispatch
jobs:
 build:
  runs-on: self-hosted
  steps:
   - uses: actions/checkout@v2

   - name: Login to Azure
     uses: azure/login@v1.1
     with:
       creds: ${{ secrets.AZURELOGINCREDS }}

   - name: Create Role Assignment
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME_TEST }}
       template: ./arm/roleassignment.json
```

Data Lake

Repeat the process with the following ARM Template JSON:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "[providers('Microsoft.Storage','storageAccounts').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'dl')]",
            "location": "[resourceGroup().location]",
            "sku": { "name": "Standard_LRS" },
            "kind": "StorageV2",
            "properties": { "isHnsEnabled": true }
        }
    ]
}
```

Notes:

- *Create a data lake using the Storage Account template plus the **isHnsEnabled** property*
- *I am not bothering to consider Gen1 data lakes in my use of "dl" for the suffix*

…and workflow YAML:

```
name: Deploy Data Lake
on: workflow_dispatch
jobs:
 build:
  runs-on: self-hosted
  steps:
   - uses: actions/checkout@v2

   - name: Login to Azure
     uses: azure/login@v1.1
     with:
       creds: ${{ secrets.AZURELOGINCREDS }}

   - name: Create Data Lake (pre-requisite for Synapse)
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME_TEST }}
       template: ./arm/datalake.json
```

File Share

Repeat the process with the following ARM Template JSON:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts/fileServices",
            "apiVersion": "[providers('Microsoft.Storage','storageAccounts').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'dl/default')]"
        },
        {
            "type": "Microsoft.Storage/storageAccounts/fileServices/shares",
            "apiVersion": "[providers('Microsoft.Storage','storageAccounts').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'dl/default/',resourceGroup().name,'dlfs')]",
            "dependsOn": [

"[resourceId('Microsoft.Storage/storageAccounts/fileServices',concat(resourceGroup().name,'dl'),'default
')]"
            ]
        }
    ]
}
```

Note: This template is dependent on pre-instantiation of a Data Lake

…and workflow YAML:

```
name: Deploy File Share
on: workflow_dispatch
jobs:
 build:
  runs-on: self-hosted
  steps:
   - uses: actions/checkout@v2

   - name: Login to Azure
     uses: azure/login@v1.1
     with:
       creds: ${{ secrets.AZURELOGINCREDS }}

   - name: Create Data Lake, File Share (pre-requisite for Synapse)
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME_TEST }}
       template: ./arm/datalake_fileshare.json
```

Synapse

Notes:

- *This section depends on pre-instantiation of a Data Lake and a code repository (DevOps or GitHub)*
- *To instantiate Synapse on a new subscription, we must ensure that Subscription > Resource Provider > Microsoft.SQL has been registered… logic for this is baked into the "**All-In YAML Example**" section*
- *Parameters for "ResourceId_Administrator", "SynapseSQL_AdministratorName", and "SynapseSQL_AdministratorPassword"… these values should be passed in from the deployment template (and captured there as secrets)*
- *"workspaceRepositoryConfiguration" {i.e., DevOps / Git configuration} **is not included** because it is not possible to programmatically set the related permissions {e.g., GitHub Personal Access Token} … I am treating this entire, particularly key step, as manual-by-necessity*
- *If you disable "publicNetworkAccess", you must also provide VNET configuration*

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters":
        {
            "SynapseSQL_AdministratorName": { "type": "string" },
            "SynapseSQL_AdministratorPassword": { "type": "secureString" }
        },
    "resources": [
        {
            "type": "Microsoft.Synapse/workspaces",
            "apiVersion": "[providers('Microsoft.Synapse','workspaces').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'s')]",
            "location": "[resourceGroup().location]",
            "identity": { "type": "SystemAssigned" },
            "properties": {
                "defaultDataLakeStorage": {
                    "resourceId":
"[resourceId('Microsoft.Storage/storageAccounts',concat(resourceGroup().name,'dl'))]",
                    "createManagedPrivateEndpoint": false,
                    "accountUrl": "[concat('https://',resourceGroup().name,'dl.dfs.core.windows.net')]",
                    "filesystem": "concat(resourceGroup().name,'dlfs')"
                },
                "connectivityEndpoints": {
                    "web":
"[concat('https://web.azuresynapse.net?workspace=%2fsubscriptions%2f',subscription().Id,'%2fresourceGrou
ps%2f',resourceGroup().name,'%2fproviders%2fMicrosoft.Synapse%2fworkspaces%2f',resourceGroup().name,'s')
]",
                    "dev": "[concat('https://',resourceGroup().name,'s','.dev.azuresynapse.net')]",
                    "sqlOnDemand": "[concat(resourceGroup().name,'s','-
ondemand.sql.azuresynapse.net')]",
                    "sql": "[concat(resourceGroup().name,'s','.sql.azuresynapse.net')]"
                },
                "managedResourceGroupName": "[concat(resourceGroup().name,'mrg-s')]",
                "sqlAdministratorLogin": "[parameters('SynapseSQL_AdministratorName')]",
                "sqlAdministratorLoginPassword": "[parameters('SynapseSQL_AdministratorPassword')]",
                "publicNetworkAccess": "Enabled"
            }
        }
    ]
}
```

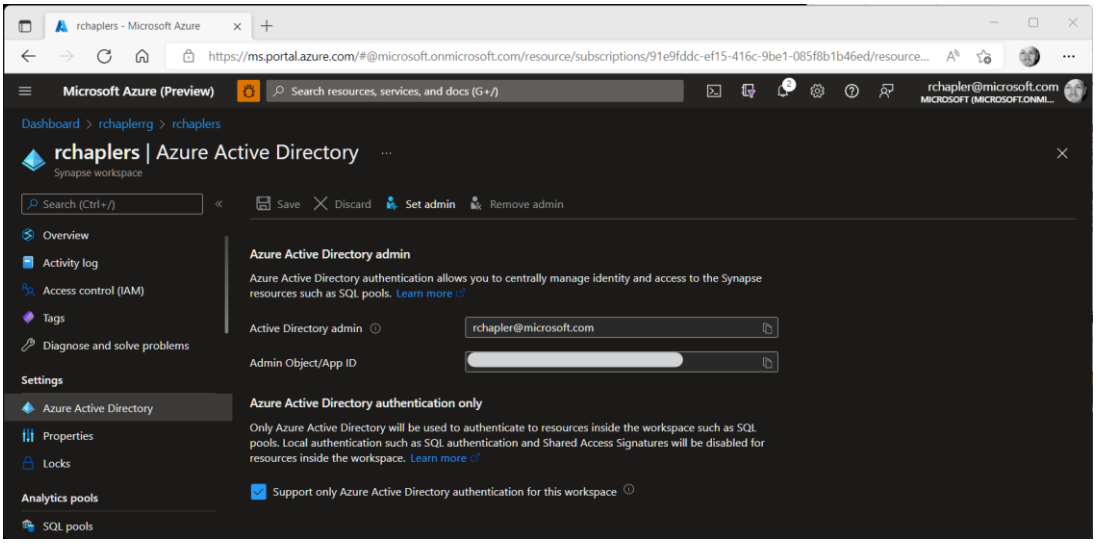
Firewall Rules

Use this template to add Firewall Rules to Synapse.

Rules should align with your security policies and network configuration.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.Synapse/workspaces/firewallRules",
            "apiVersion": "[providers('Microsoft.Synapse','workspaces').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'s','/AllowAllWindowsAzureIps')]",
            "properties": { "startIpAddress": "0.0.0.0", "endIpAddress": "0.0.0.0" }
        },
        {
            "type": "Microsoft.Synapse/workspaces/firewallRules",
            "apiVersion": "[providers('Microsoft.Synapse','workspaces').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'s','/allowRange')]",
            "properties": { "startIpAddress": "#.#.0.0", "endIpAddress": "#.#.255.255" }
        }
    ]
}
```

Azure Active Directory >> Set Admin



Once deployment of a Synapse Analytics Workspace is complete, consider what entity should be set as the Azure Active Directory admin.

Repository

The “how to” is already described in the Preparation > Synapse > Configure Repository section. Inclusion here is intended only to call out two deployment-related considerations:

- **Repository** – each Synapse instantiation {i.e., DEV, TEST, and PROD} must be manually configured after deployment of infrastructure-as-code …. **programmatic setting of Synapse + GitHub Personal Access Token is not yet an available feature**
- **Branch** – the collaboration branch for each Synapse instance should be configured with the corresponding branch {e.g., “development”, “test”, and “production”}in each repository

Synapse Studio, Role-Based Access Control (RBAC)

A person that creates a Synapse workspace is initialized as a Workspace Administrator, allowing full access to Synapse Studio, and granting the ability to manage role assignments.

When we create a Synapse workspace using an ARM template, however, there is no way to grant Synapse roles via the Studio interface.

And an aside… ARM templates are not a supported method for changing Synapse Studio roles.

So, the Deployment Manager will have to manually update Synapse RBAC settings as below.

Manage Access with Cloud Shell (PowerShell)

Navigate to Cloud Shell and select **PowerShell** from the "**Select Environment**" drop-down menu.

List Role Definitions

Update and execute the following command to detail Synapse Studio roles (in preparation for future commands):

```
az synapse role definition list --workspace-name <synapseWorkspaceName>
```

(Partial) logic explanation:

- --workspace-name <synapseWorkspaceName>
  …where <synapseWorkspaceName> refers to the name of your Synapse instance {e.g., <UseCase>s}

The JSON response will include role definition data like the snippet below:

```
  {
    "availabilityStatus": "Available",
    "description": "Full Synapse access to serverless SQL pools, Apache Spark pools and Integration
runtimes.  Includes create, read, update and delete access to all published code artifacts.  Includes
Compute Operator, Linked Data Manager, and Credential User permissions on the workspace system identity
credential.  Includes granting access.  Azure permissions are required to create, delete, or manage
compute resources.",
    "id": "6e4bf58a-b8e1-4cc3-bbf9-d73143322b78",
    "isBuiltIn": true,
    "name": "Synapse Administrator",
...
```

Capture the **id** and **name** values for later use.

Create Role Assignment

Update and execute the following command to create a Synapse Studio role assignment:

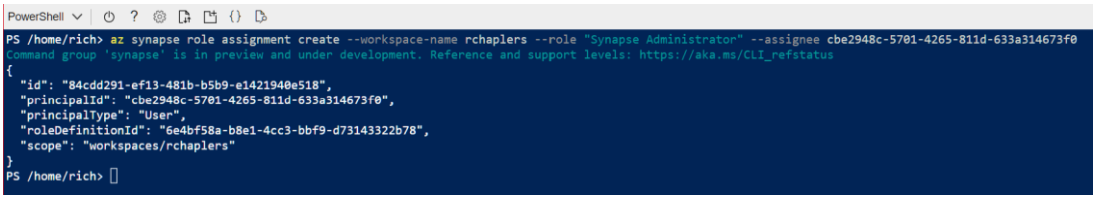
```
az synapse role assignment create --workspace-name <SynapseWorkspaceName} --role <roleNameOrId> --
assignee <aadPrincipalId>
```

Example:

```
az synapse role assignment create --workspace-name rchaplers --role "Synapse
Administrator" --assignee cbe2948c-5701-4265-811d-633a314673f0
```

(Partial) logic explanation:

- --workspace-name <synapseWorkspaceName>
  …where <synapseWorkspaceName> refers to the name of your Synapse instance {e.g., <UseCase>s}



Note: Considering adding various Synapse RBAC Roles; for example:

- *"Synapse Artifact Publisher" to allow you to publish*
- *"Synapse Linked Data Manager" to allow you to create a Linked Server*

Data Explorer Pool

The Data Explorer Pool template requires a "**workspaceUID**" value (that will help us uniquely identify the Synapse workspace created in the prior step). You can get this value using the REST API call described at the following link: [Workspaces - List - REST API (Azure Synapse) | Microsoft Docs]

Use the "**Try It**" button and resulting form to personalize and submit the following REST API request:

```
GET
https://management.azure.com/subscriptions/{subscriptionId}/providers/Microsoft.Synapse/workspaces?api-version=2021-06-01
```

The response Body will include a "**workspaceUID**" value that you can bake into the template below:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.Synapse/workspaces/kustoPools",
            "apiVersion": "[providers('Microsoft.Synapse','workspaces/kustoPools').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'s/',resourceGroup().name,'dep')]",
            "location": "[resourceGroup().location]",
            "sku": {
                "name": "Compute optimized",
                "size": "Extra small",
                "capacity": 2
            },
            "properties": {
                "enableStreamingIngest": false,
                "enablePurge": false,
                "workspaceUID": "db162ad0-8a4e-4aa7-a7df-60d6b044999a"
            }
        }
    ]
}
```

Data Explorer Database

Use this template to add a database to a Data Explorer database. This template can be used with a Data Explorer Pool or Data Explorer Cluster.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.Synapse/workspaces/kustoPools/Databases",
            "apiVersion":
"[providers('Microsoft.Synapse','workspaces/kustoPools/Databases').apiVersions[0]]",
            "name":
"[concat(resourceGroup().name,'s/',resourceGroup().name,'dep/',resourceGroup().name,'ded')]",
            "location": "[resourceGroup().location]",
            "kind": "ReadWrite",
            "properties": { "softDeletePeriod": "P365D", "hotCachePeriod": "P31D" }
        }
    ]
}
```

Code

This section describes deployment of data solution "code" {i.e., pipelines, datasets, linked services, etc.}. These tasks will be completed more frequently.

Once Synapse is re-configured, we shift to recurring tasks.

Deployment process can be remarkably simple or quite complex depending on your team and operational requirements. Consider the following activities when designing your process:

1. **Developer begins work on a development task** by creating a new branch {i.e., "Branch X"} based on the collaboration branch
2. **Developer completes development task**, tests changes, and ensures quality of commits in their branch ("Branch X")
3. **Developer creates a pull request** ("Branch X" >> "Collaboration" branch), and assigns necessary reviewers / approvers (individually or as part of a group)
4. **Peers review**, provide feedback and approve (or reject) the pull request
5. On a regular schedule, **Development Manager creates "Release" branch** including only those commits that should make their way through the deployment process
6. **Development Manager creates a pull request** ("Release" branch >> "TEST" branch), and assigns to Deployment Manager
7. **Deployment Manager reviews "Release" branch** for obvious issues
8. Assuming no issues, **Deployment Manager creates a pull request** ("Release" branch >> "PROD" branch)

A few notes:

- *Before deploying "Release" branch commits {e.g., to a "TEST" branch}, consider replacing the contents of the "TEST" branch with a copy of the contents of the "PROD" branch*
- *When you begin your formal deployment process, and IF you have a previously created "PROD" branch … consider purging any questionable content to make "PROD" as perfect as possible*

Deploy Schema

Before turning on the pipelines and triggers to move data, we must prepare the destination database (specific to the deployment environment) by creating or updating the schema.

There are many options for semi-automated deployment of database schema elements; for now, we will discuss two: 1) via GitHub Action or 2) via Synapse Pipeline.

GitHub Action

If you are using an Azure Data Explorer Cluster, you can use this YAML in a GitHub Action to execute KQL Logic such as ".create table"

```
name: Azure Data Solution, Synapse Data Explorer, Schema for TableX
on: workflow_dispatch
jobs:
 build:
  runs-on: self-hosted
  steps:
   - uses: actions/checkout@v2
   - uses: elstudio/action-install-azure-cli@main

   - name: Login to Azure
     uses: azure/login@v1.1
     with:
       creds: ${{ secrets.AZURELOGINCREDS_PROD }}

   - name: _TableX
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_PROD }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME_PROD }}
       template: ./templates/dataexplorerscript.json
       parameters:
         kqlScript=".create table _TableX ( Col1: int, Col2: string, _id: string, _dt: datetime, _op:
string ) with ( folder = 'Bronze' )"
```

[Trigger an Azure Pipelines run from GitHub Actions - Azure Pipelines | Microsoft Docs]

Bonus Materials

All-In YAML Example

The following YAML will deploy ARM templates for each of the resources necessary for a Data Solution team starting use of a new Azure subscription:

```
name: Data Solution Architecture, Deployment
on: workflow_dispatch
jobs:
 build:
  runs-on: self-hosted
  steps:
   - uses: actions/checkout@v2

   - name: Login to Azure
     uses: azure/login@v1.1
     with:
       creds: ${{ secrets.AZURELOGINCREDS }}

  # ######################### ######################### Only needs to be uncommented for initial
deployment
  #  - name: Register Resource Provider Microsoft.Sql (pre-requisite for Synapse)
  #    run: az provider register --namespace Microsoft.Sql

  #  - name: Pause for completion of Resource Provider registration
  #    run: sleep 5m
  #    shell: bash

   - name: Create Synapse Analytics Workspace
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME }}
       template: ./arm/synapse.json
       parameters:
         SynapseSQL_AdministratorName=${{ secrets.SYNAPSESQL_ADMINISTRATORNAME }}
         SynapseSQL_AdministratorPassword=${{ secrets.SYNAPSESQL_ADMINISTRATORPASSWORD }}

   - name: Create Synapse, Firewall Rules
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME }}
       template: ./arm/synapse_firewallrules.json

   - name: Create Synapse, Data Explorer Pool
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME }}
       template: ./arm/synapse_dataexplorerpool.json

   - name: Create Data Explorer Database
     uses: azure/arm-deploy@v1
     with:
       subscriptionId: ${{ secrets.SUBSCRIPTIONID_TEST }}
       resourceGroupName: ${{ secrets.RESOURCEGROUPNAME }}
       template: ./arm/dataexplorerdatabase.json
```

More Templates

There are some excellent ARM templates at: https://azure.microsoft.com/en-us/resources/templates

File Sync (ARM)

(aka Storage Sync Service)

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "microsoft.storagesync/storageSyncServices",
      "apiVersion": "[providers('Microsoft.StorageSync','storageSyncServices').apiVersions[0]]",
      "name": "[concat(resourceGroup().name,'sss')]",
      "location": "[resourceGroup().location]",
      "properties": {
        "incomingTrafficPolicy": "AllowAllTraffic"
      }
    }
  ]
}
```

Key Vault (ARM)

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.KeyVault/vaults",
            "apiVersion": "[providers('Microsoft.KeyVault','vaults').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'kv')]",
            "location": "westus",
            "properties": {
                "sku": {
                    "family": "A",
                    "name": "Standard"
                },
                "tenantId": "[subscription().tenantId]",
                "accessPolicies": [
                    {
                        "tenantId": "[subscription().tenantId]",
                        "objectId": "{ObjectId for Azure Active Directory User]}",
                        "permissions": {
                            "keys": [
                                "Get",
                                "List",
                                "Update",
                                "Create",
                                "Delete"
                            ],
                            "secrets": [
                                "Get",
                                "List",
                                "Set",
                                "Delete"
                            ]
                        }
                    }
                ]
            }
        }
```

```
    ]
}
```

KQL Script (ARM)

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "kqlScript": {
            "defaultValue": ".create table _bt_info ( nlfdkbnr: int, sbaureihe: string, spnr8: string,
dtinsert:datetime, _id: string, _dt: datetime, _op: string ) with ( folder = 'Bronze' )",
            "type": "String"
        }
    },
    "resources": [
        {
            "type": "Microsoft.Kusto/Clusters/Databases/Scripts",
            "apiVersion": "[providers('Microsoft.Kusto','Clusters/Databases/Scripts').apiVersions[0]]",
            "name":
"[concat(resourceGroup().name,'dec/',resourceGroup().name,'ded/',guid(resourceGroup().id,
deployment().name))]",
            "properties": {
                "scriptContent": "[parameters('kqlScript')]",
                "continueOnErrors": false
            }
        }
    ]
}
```

Storage Account (ARM)

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "[providers('Microsoft.Storage','storageAccounts').apiVersions[0]]",
      "name": "[concat(resourceGroup().name,'sa')]",
      "location": "[resourceGroup().location]",
      "sku": { "name": "Standard_LRS" },
      "kind": "StorageV2"
    }
  ]
}
```

Virtual Machine (ARM)

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "[providers('Microsoft.Network','publicIPAddresses').apiVersions[0]]",
      "name": "[concat(resourceGroup().name,'-pia')]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Basic"
      },
      "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "dnsSettings": {
          "domainNameLabel": "[toLower(format('{0}-{1}', resourceGroup().name,
uniqueString(resourceGroup().id, resourceGroup().name)))]"
        }
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "[providers('Microsoft.Network','networkSecurityGroups').apiVersions[0]]",
      "name": "[concat(resourceGroup().name,'-nsg')]",
      "location": "[resourceGroup().location]",
      "properties": {
        "securityRules": [
          {
            "name": "default-allow-3389",
            "properties": {
              "priority": 1000,
              "access": "Allow",
              "direction": "Inbound",
              "destinationPortRange": "3389",
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "[providers('Microsoft.Network','virtualNetworks').apiVersions[0]]",
      "name": "[concat(resourceGroup().name,'-vn')]",
      "location": "[resourceGroup().location]",
```

```
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "10.0.0.0/16"
          ]
        },
        "subnets": [
          {
            "name": "Subnet",
            "properties": {
              "addressPrefix": "10.0.0.0/24",
              "networkSecurityGroup": {
                "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
concat(resourceGroup().name,'-nsg'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups', concat(resourceGroup().name,'-nsg'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-02-01",
      "name": "[concat(resourceGroup().name,'-ni')]",
      "location": "[resourceGroup().location]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses', concat(resourceGroup().name,'-
pia'))]"
              },
              "subnet": {
                "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
concat(resourceGroup().name,'-vn'), 'Subnet')]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses', concat(resourceGroup().name,'-pia'))]",
        "[resourceId('Microsoft.Network/virtualNetworks', concat(resourceGroup().name,'-vn'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "[providers('Microsoft.Compute','virtualMachines').apiVersions[0]]",
      "name": "[resourceGroup().name]",
      "location": "[resourceGroup().location]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "Standard_D2_v3"
        },
        "osProfile": {
          "computerName": "[resourceGroup().name]",
          "adminUsername": "Admin123",
          "adminPassword": "Password123!"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
```

```
            "sku": "2022-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "name":"[concat(resourceGroup().name,'-d-os')]",
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "StandardSSD_LRS"
            }
          },
          "dataDisks": [
            {
              "name":"[concat(resourceGroup().name,'-d-data')]",
              "diskSizeGB": 1023,
              "lun": 0,
              "createOption": "Empty"
            }
          ]
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces', concat(resourceGroup().name,'-
ni'))]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces', concat(resourceGroup().name,'-ni'))]"
      ]
    }
  ],
  "outputs": {
    "hostname": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Network/publicIPAddresses',
concat(resourceGroup().name,'-pia'))).dnsSettings.fqdn]"
    }
  }
}
```

User Assigned Managed Identity (ARM)

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "variables": {},
    "resources": [
        {
            "type": "Microsoft.ManagedIdentity/userAssignedIdentities",
            "apiVersion":
"[providers('Microsoft.ManagedIdentity','userAssignedIdentities').apiVersions[0]]",
            "name": "[concat(resourceGroup().name,'uami')]",
            "location": "[resourceGroup().location]"
        }
    ]
}
```

Protect Resources

Use Case

Example notes:

- “We need to prevent the accidental deletion of our production Data Lake”
- “Nothing in the production instance should be deleted without appropriate permissions”

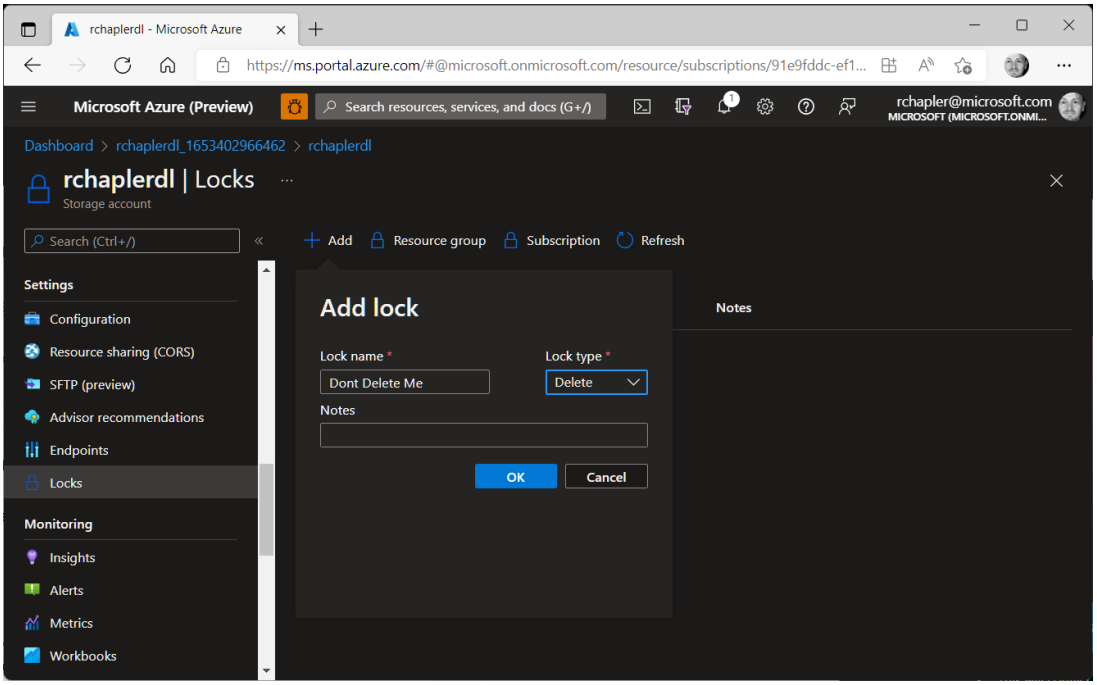
Prerequisites

This solution requires the following resources:

- Data Lake

Lock Storage Account

Navigate to your Data Lake, then click **Locks** in the **Settings** group of the navigation pane.



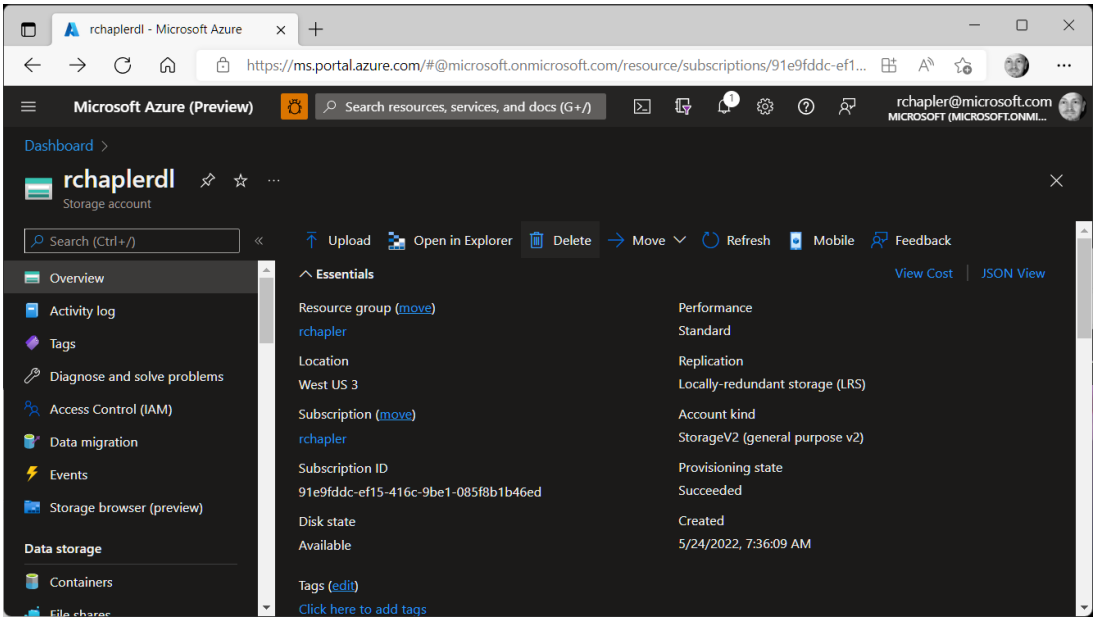
Click the “**+ Add**” button. On the resulting “**Add lock**” pop-up, including:

Lock Type	Select **Delete**

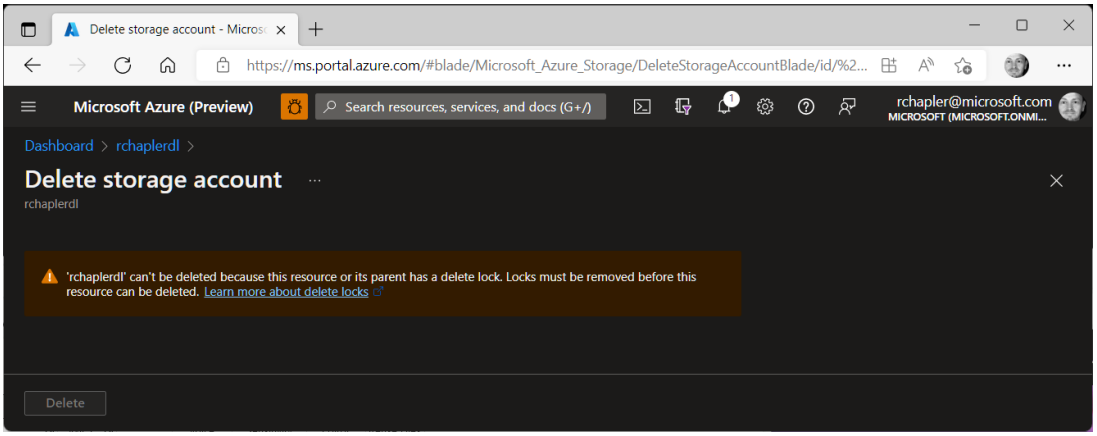
Click the **OK** button.

Confirm Success

Click Overview at the top of the navigation pane.



Click the **Delete** button.



You should receive a message indicating that the resource cannot be deleted.

Presentation

Query from On-Prem

Follow these instructions to **query from an on-prem client**.

Prerequisites

This solution requires the following resources:

- Azure CLI
- Data Explorer
- Postman

Get Token

In a Command Prompt window (with Azure CLI installed), execute the following command: az login

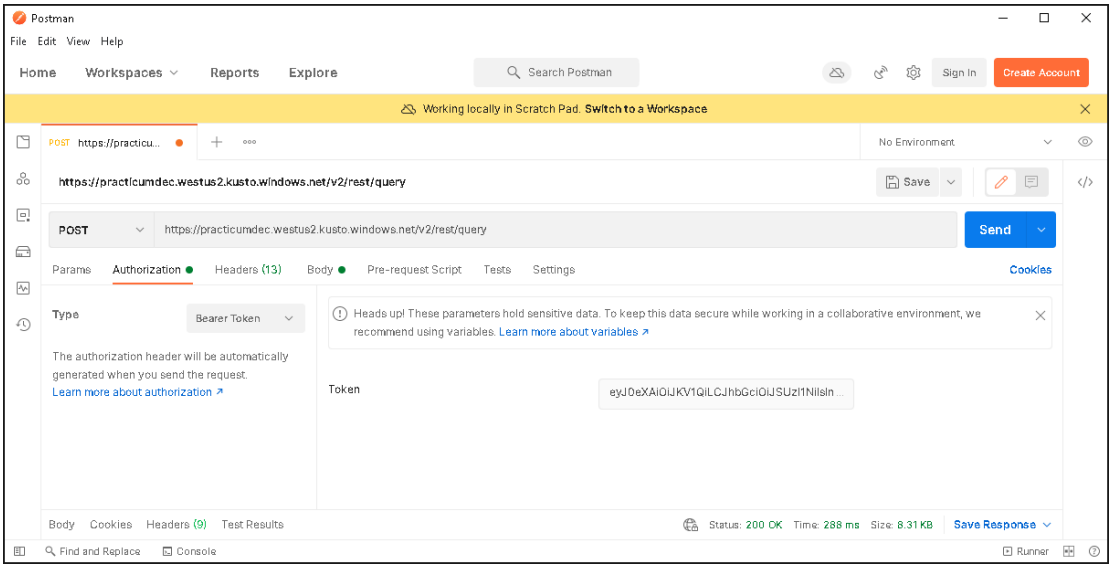
Use the resulting browser window to provide credentials.

In a Command Prompt window (with Azure CLI installed), execute the following command: az account get-access-token --resource https://<UseCase>dec.westus2.kusto.windows.net --query accessToken --output tsv

Capture the returned token for use later in this section.

Post API Request

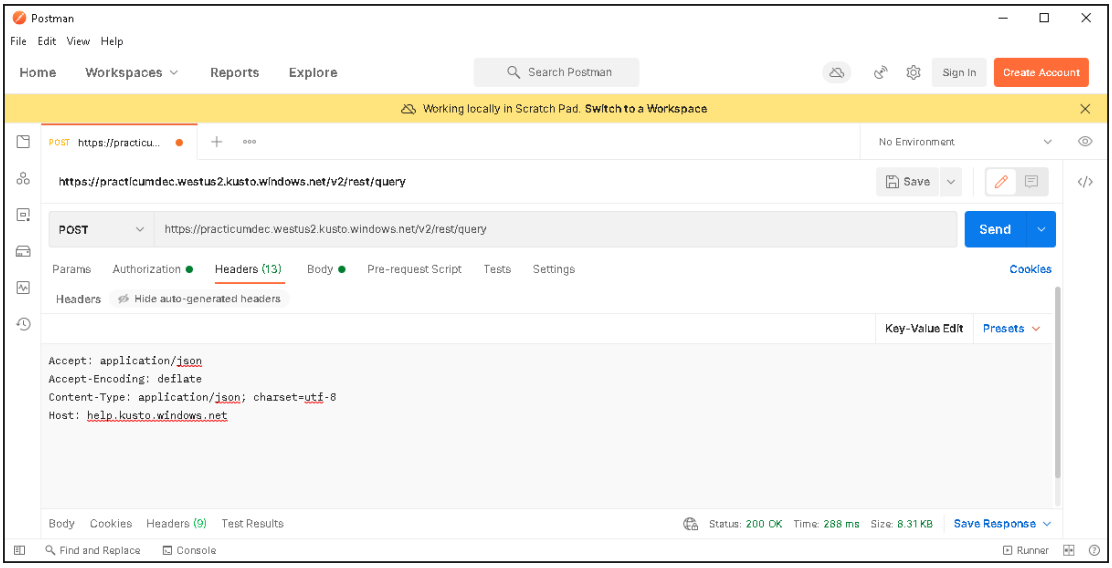
Open a new request in Postman.



Select **POST** from the drop-down menu. Enter a hostname (with the name you used for your cluster); example:

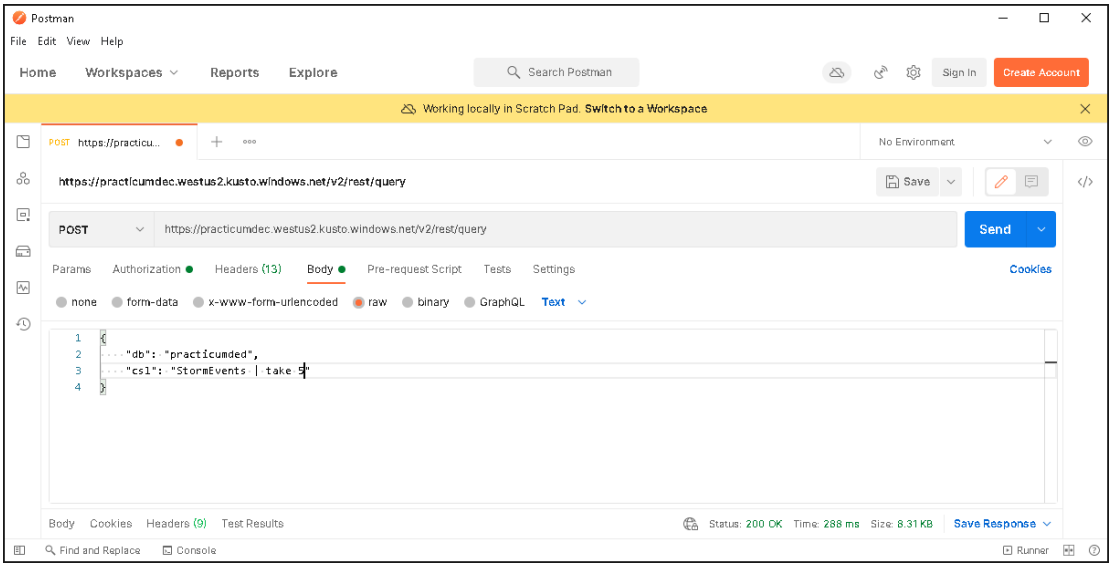
`https://<UseCase>dec.westus2.kusto.windows.net/v2/rest/query`

On the **Authorization** tab, select "**Bearer Token**" from the drop-down menu and paste your previously copied token in the resulting interface.



On the **Headers** tab, enter the following items:

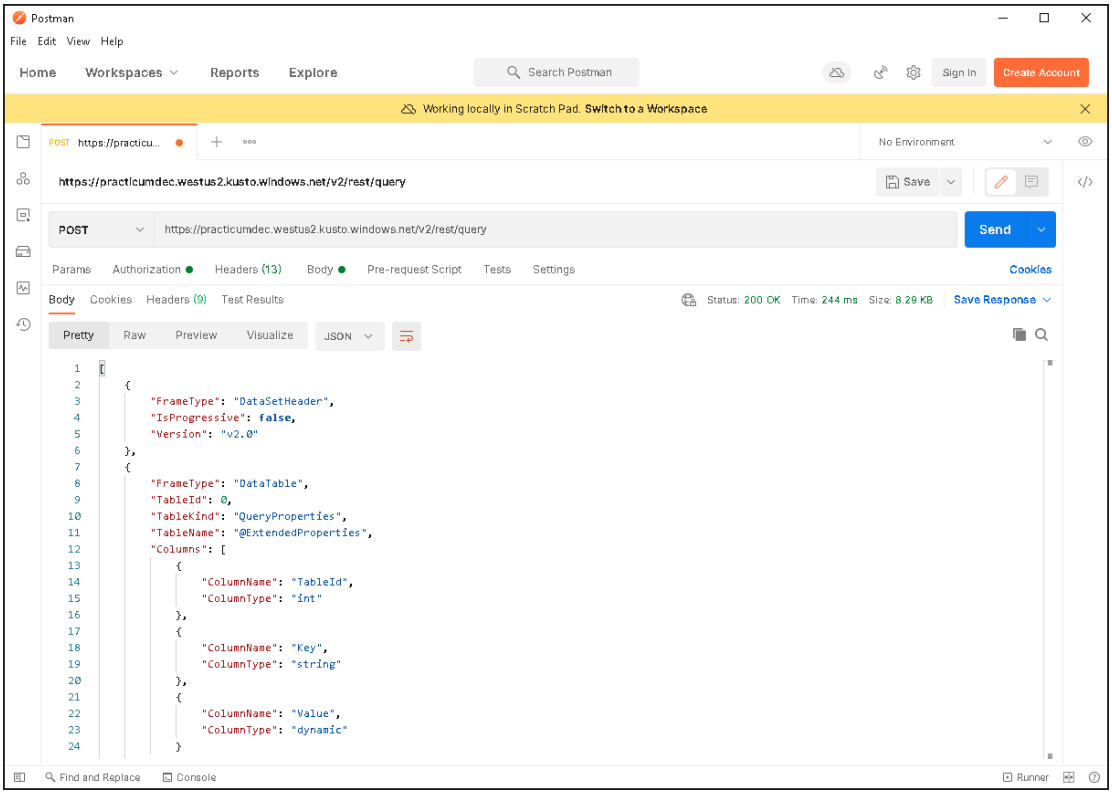
Accept	application/json
Accept-Encoding	deflate
Content-Type	application/json; charset=utf-8
Host	help.kusto.windows.net



On the **Body** tab, paste the following:

```
{
 "db": "<UseCase>ded",
 "csl": "StormEvents | take 5"
}
```

Click the **Send** button.



You should see JSON with both schema and data included.

Detect Anomalies

Use Case: Customer XYZ shared the following requirements:

- Detect data anomalies for Metric 123
- When significant anomalies are detected, the solution should support investigation

Follow these instructions to on-board data feeds and investigate findings.

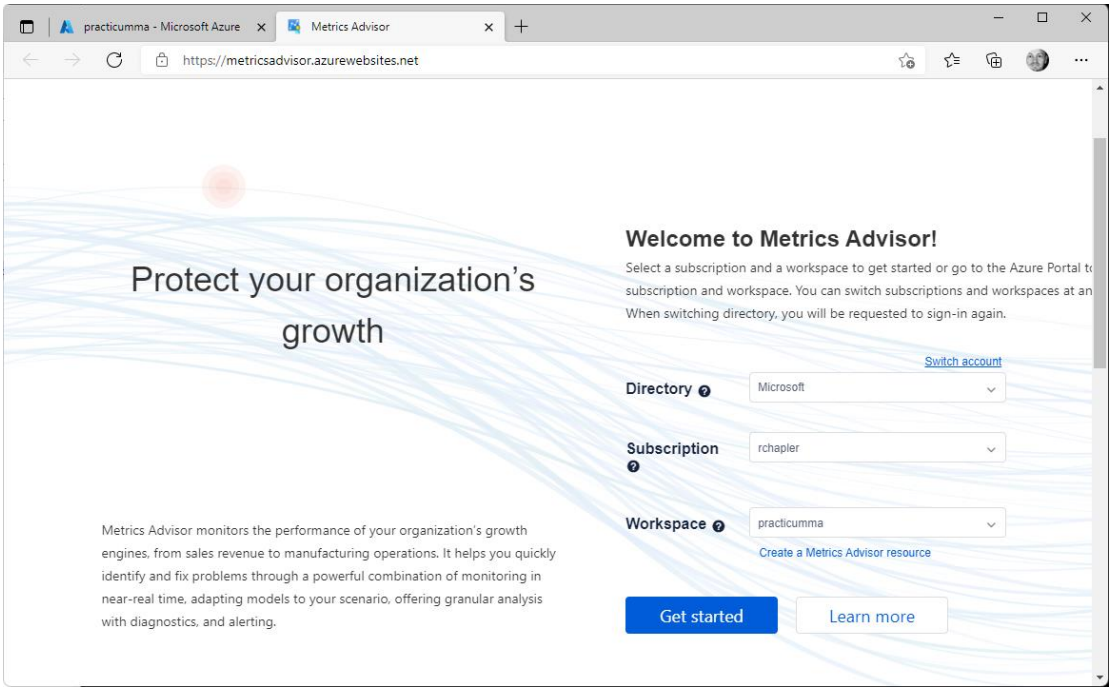
Prerequisites

This solution requires the following resources:

- Data Explorer
- Metrics Advisor

Onboard Data Feed

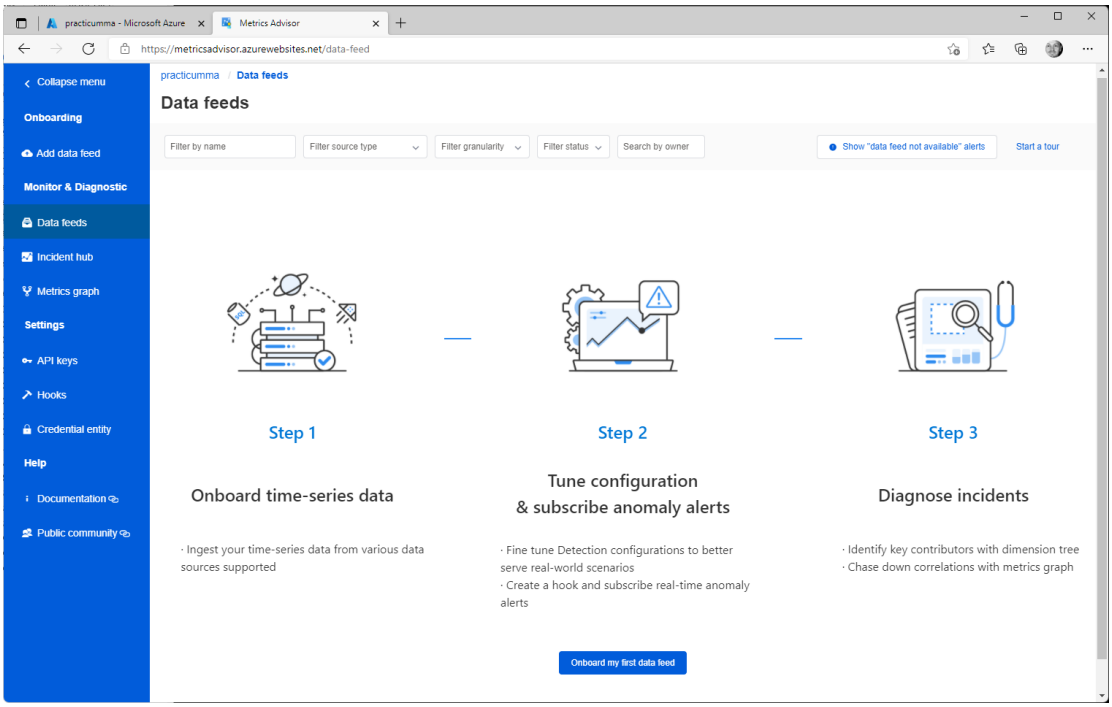
Navigate to Metrics Advisor and then click the "**Go to your workspace**" link.



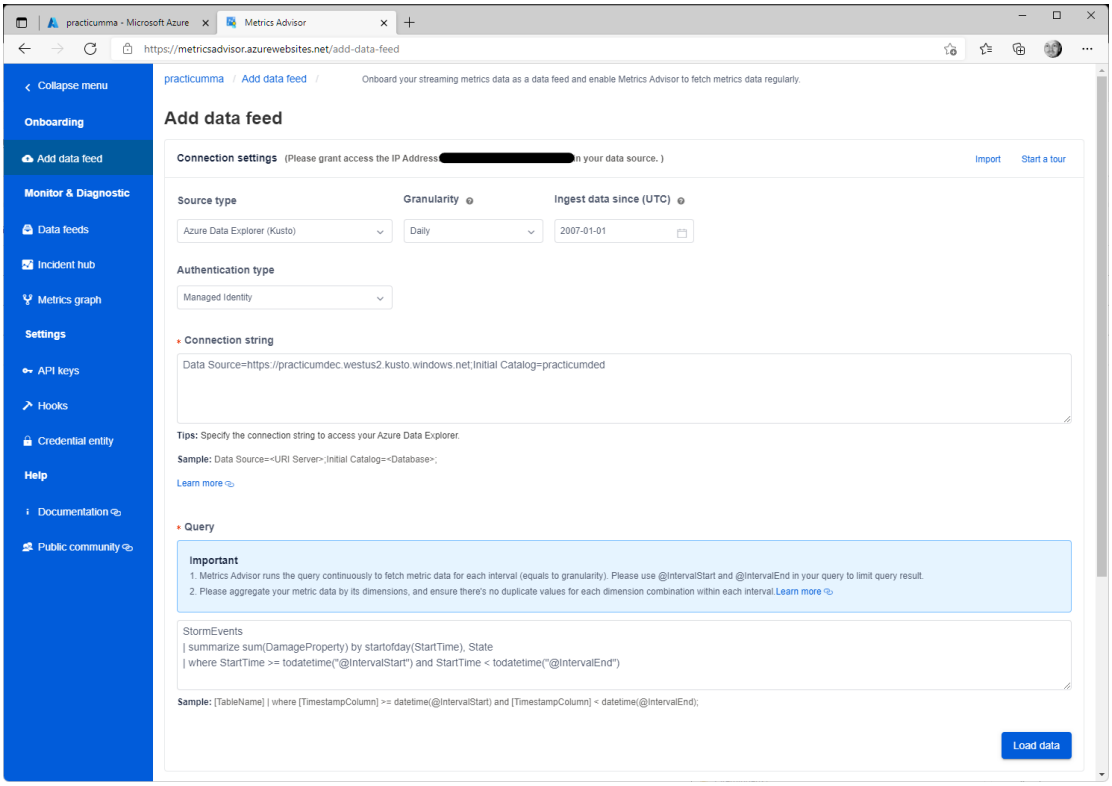
Complete the "**Welcome to Metrics Advisor!**" form, including:

Directory	Select your directory
Workspace	Select Metrics Advisor

Click the “**Get started**” button.



Click the “Onboard my first data feed” button.

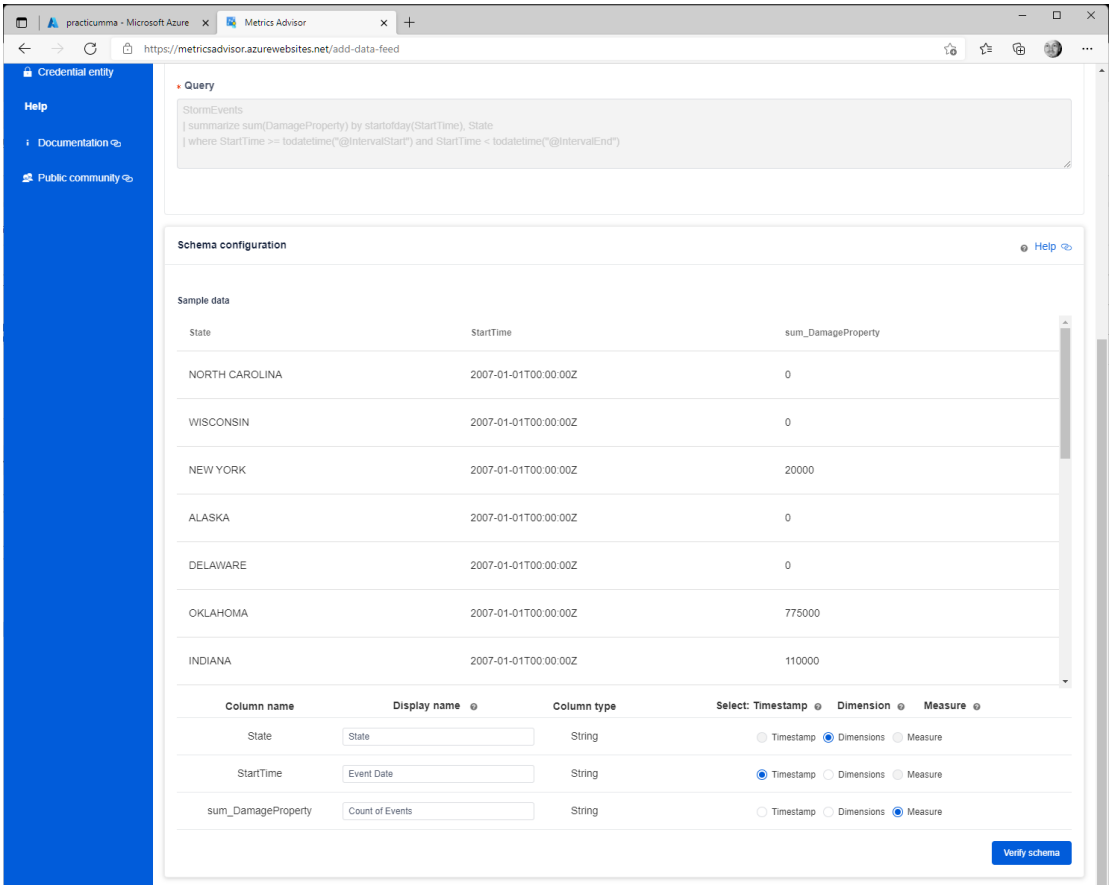


Complete the "**Add data feed**" form, including:

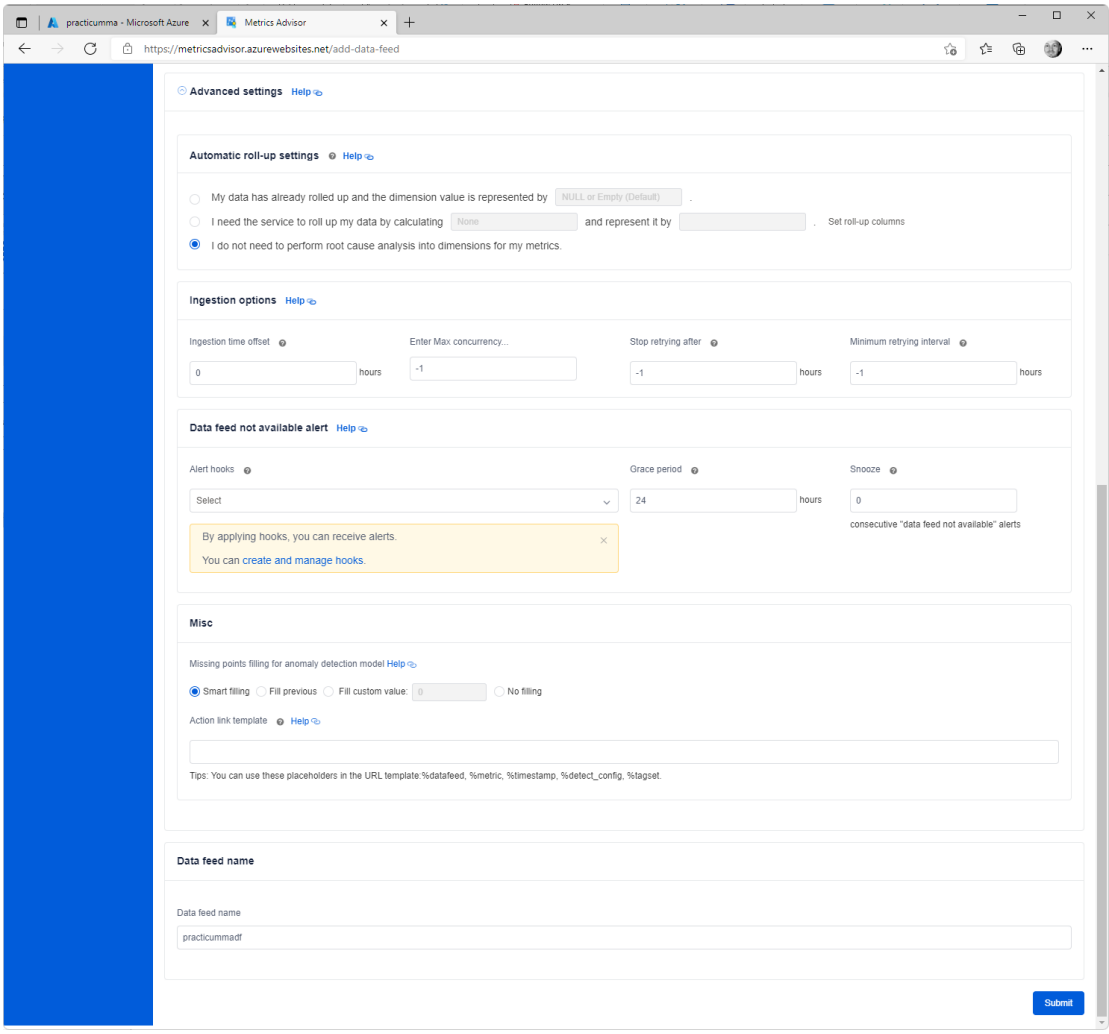
Source Type	Select "Azure Data Explorer (Kusto)"
Granularity	Confirm default selection, **Daily**
Ingest Data Since (UTC)	Select a date appropriate for your data set
Authentication Type	Select "Managed Identity"
Connection String	Enter in form: `Data Source=https://{ADX Cluster}.{region}.kusto.windows.net;Initial Catalog={ADX Database}`

Query	Create and test a query in your database, then parameterize; example below:
	
	`StormEvents`
	`| summarize sum(DamageProperty) by startofday(StartTime), State`
	`| where StartTime >= todatetime("@IntervalStart") and StartTime < todatetime("@IntervalEnd")`

Click the "**Load Data**" button.

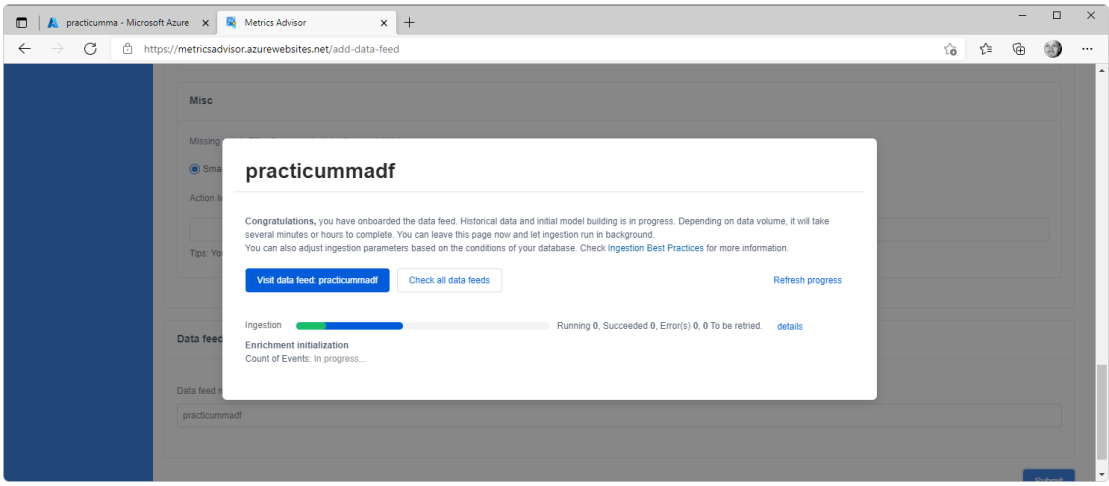


Click the “**Verify Schema**” button.



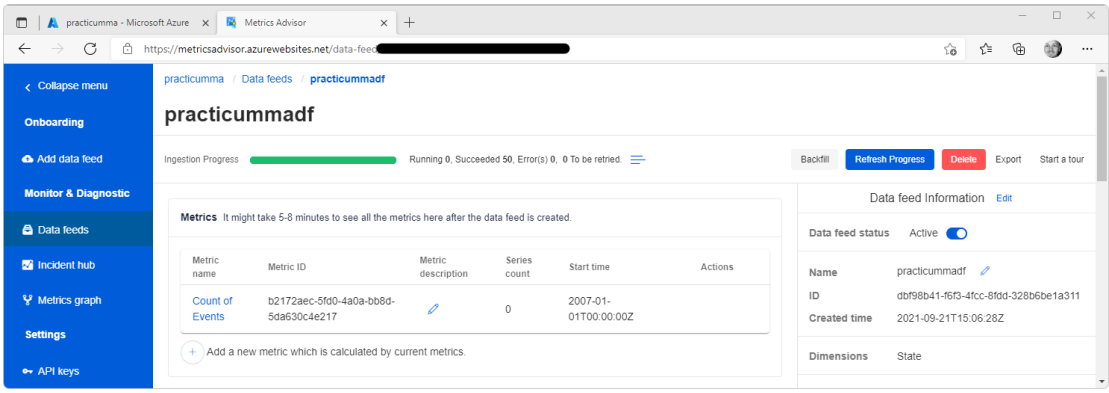
No changes are required to default values in “**Advanced Settings**.”

Enter a “**Data Feed Name**” and then click the Submit button.



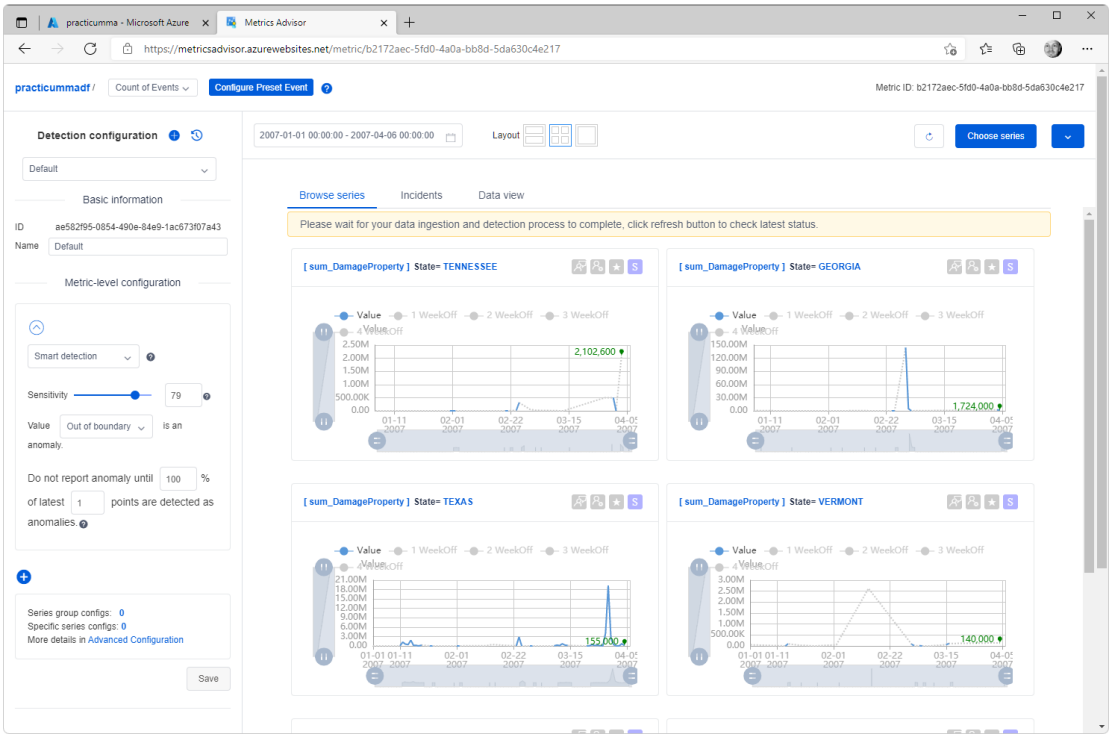
You will receive a "**Congratulations**…" message.

Click the "**Visit data feed**…" button.



Monitor progress on the data feed page.

Click into the Metric Name link to see analysis.


practicumma - Microsoft Azure
Metrics Advisor
https://metricsadvisor.azurewebsites.net/metric/b2172aec-5fd0-4a0a-bb8d-5da630c4e217
practicummadf /
Count of Events
Configure Preset Event
Metric ID: b2172aec-5fd0-4a0a-bb8d-5da630c4e217
Detection configuration
Default
Basic information
ID ae582f95-0854-490e-84e9-1ac673f07a43
Name Default
Metric-level configuration
Smart detection
Sensitivity 79
Value Out of boundary is an anomaly.
Do not report anomaly until 100 % of latest 1 points are detected as anomalies.
Series group configs: 0
Specific series configs: 0
More details in Advanced Configuration
Save
2007-01-01 00:00:00 - 2007-04-06 00:00:00
Layout
Choose series
Browse series
Incidents
Data view
Please wait for your data ingestion and detection process to complete, click refresh button to check latest status.
[sum_DamageProperty] State= TENNESSEE
[sum_DamageProperty] State= GEORGIA
[sum_DamageProperty] State= TEXAS
[sum_DamageProperty] State= VERMONT
Value
1 WeekOff
2 WeekOff
3 WeekOff
4 WeekOff
2,102,600
1,724,000
155,000
140,000

Application+ AI

Use Case: Company XYZ wants to produce a customer-facing app that can be used to capture images and metadata for an insurance claim {e.g., a vehicle with damage to a headlight}.

Follow these instructions to build an application that employs artificial intelligence with minimal code.

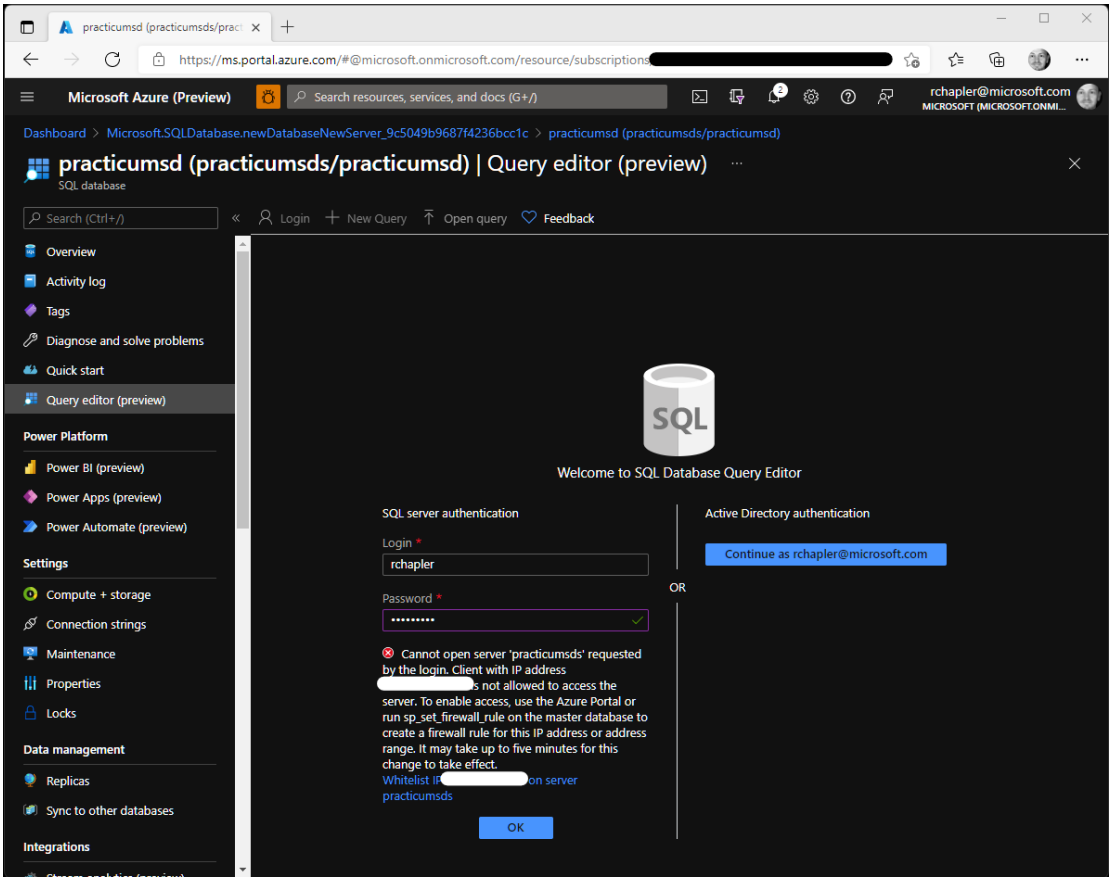
Prerequisites

This solution requires the following resources:

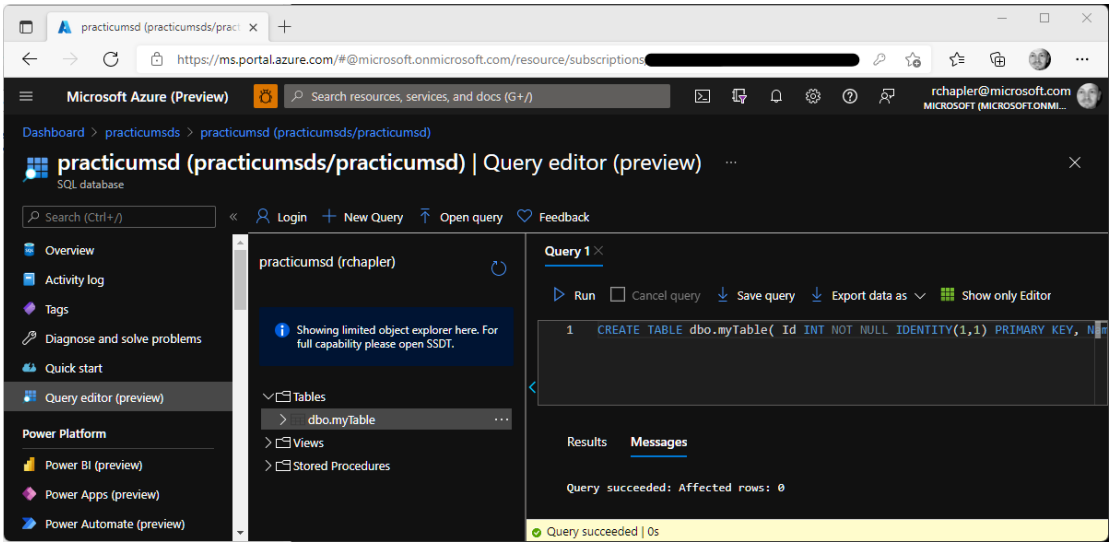
- Azure SQL
- PowerApps

Create Target Table

Navigate to SQL.



Navigate to “**Query editor**…” and login. Whitelist your IP address as appropriate.



Execute the following T-SQL:

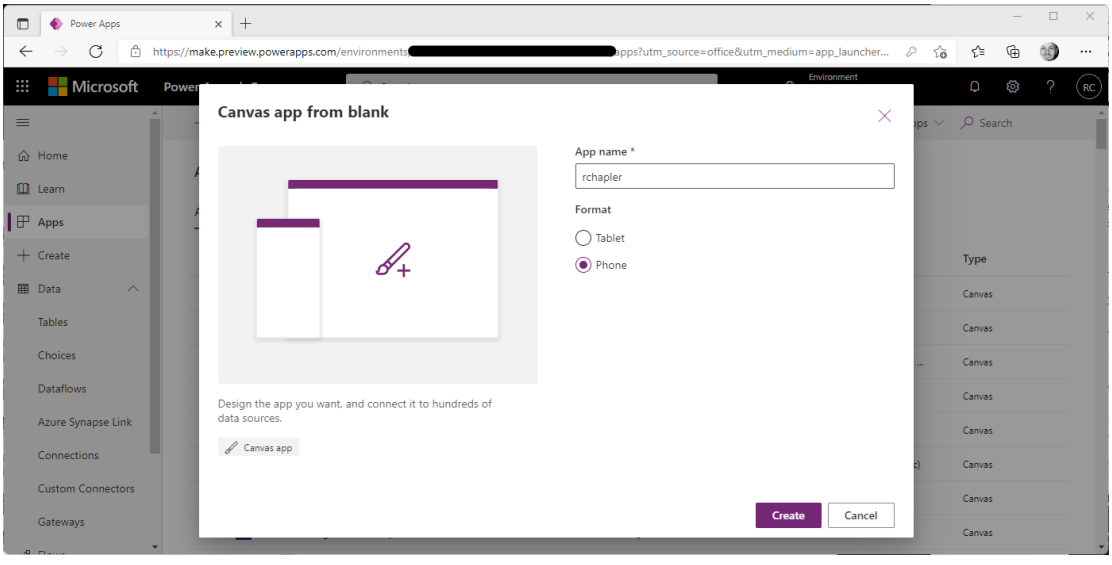
```
CREATE TABLE dbo.myTable( Id INT NOT NULL IDENTITY(1,1) PRIMARY KEY, Name VARCHAR(64), Picture IMAGE )
```

Create Canvas App

Navigate to Power Apps (https://make.preview.Power Apps.com/).

Click on **Apps** in the navigation pane.

At the top of the “**Apps**” form, click “**+ New app**” and select **Canvas** from the drop-down menu.



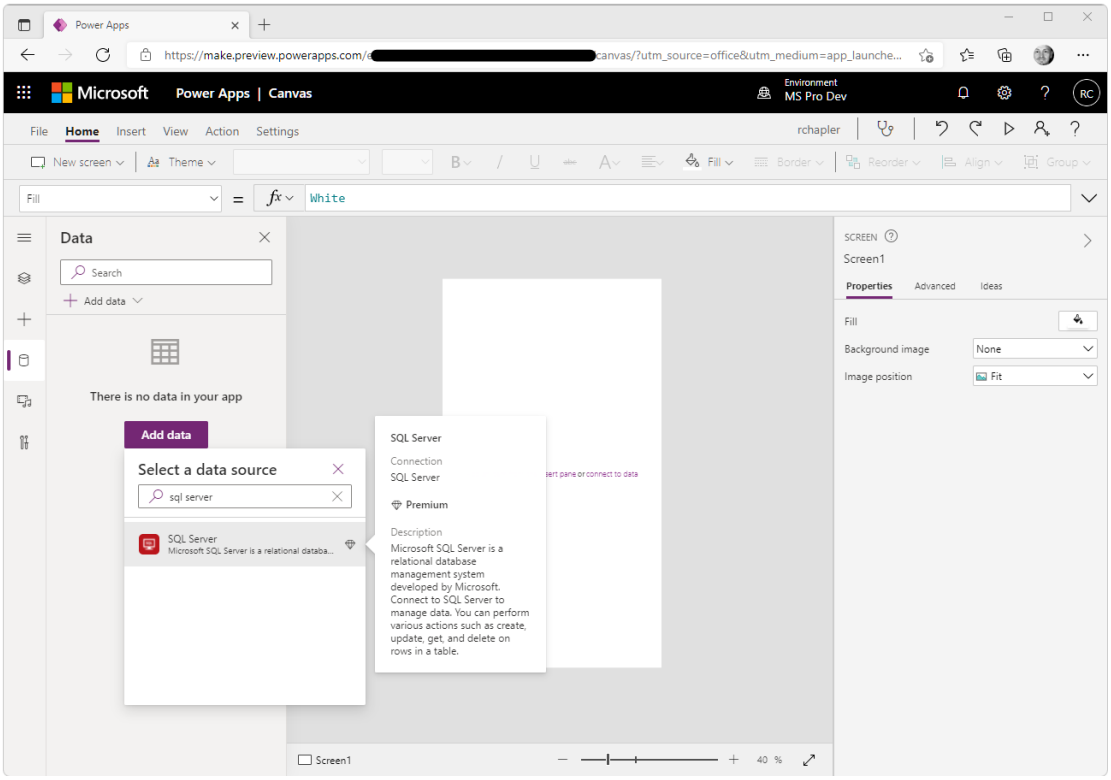
Complete the “**Canvas app from blank**” pop-up, enter the following items:

Format	Select the **Phone** radio button

Click the **Create** button.

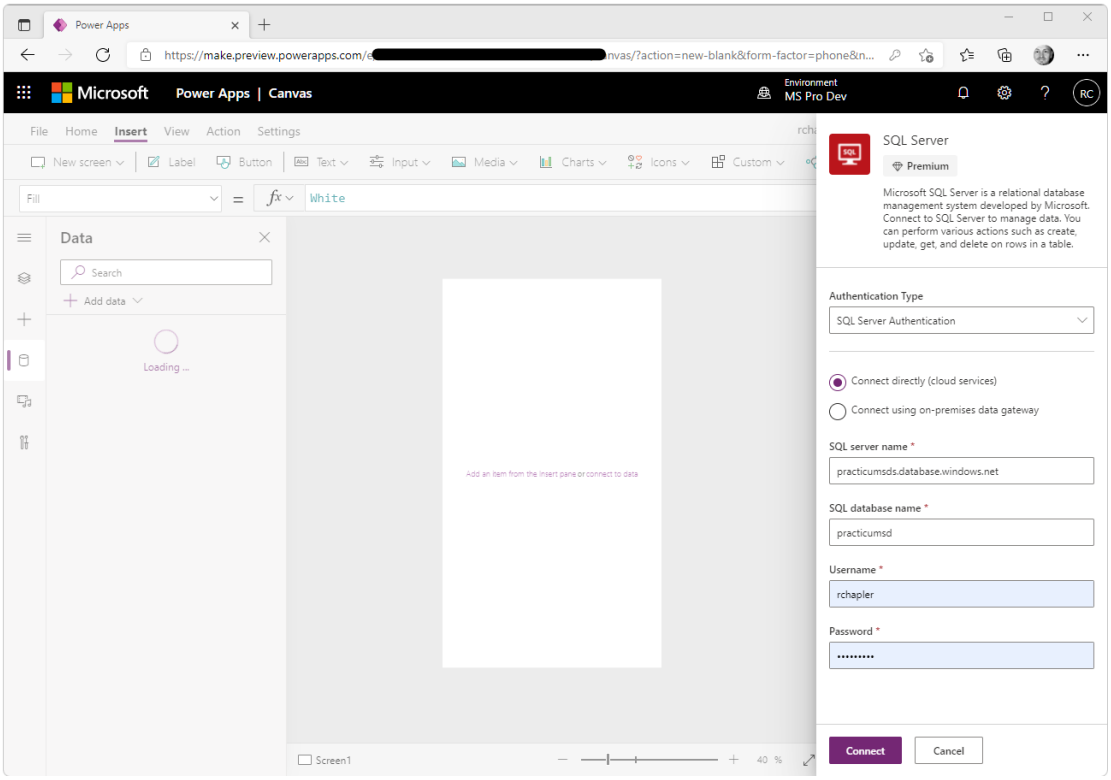
Add Data

Click on **Data** in the navigation pane. Click on the “**Add data**” button.



In “**Select a data source**”, enter “**sql server**” in the search input and select “**SQL Server**” from the results.

Click “**+ Add a connection**” on the resulting drop-down menu.

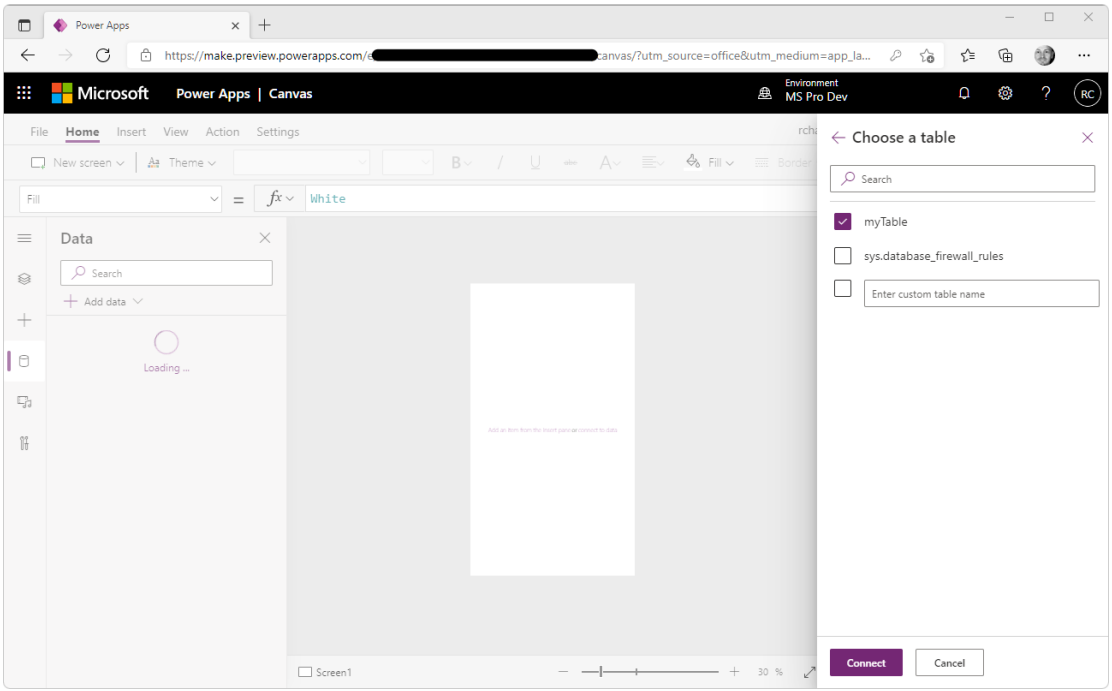


Complete the "**SQL Server**" pop-out, enter the following items:

Authentication Type	SQL Server Authentication
SQL Server Name **SQL Database Name** **Username** **Password**	Enter the values employed during database server creation

Click the **Connect** button.

Note: Resolve error messages "We weren't able to add this connection…" and "Client with IP address #.#.#.# is not allowed…", by adding your IP address to the Azure SQL Server firewall.



Complete the "**Choose a table**" pop-out, check the table created in Create Target Table.

Click the **Connect** button.

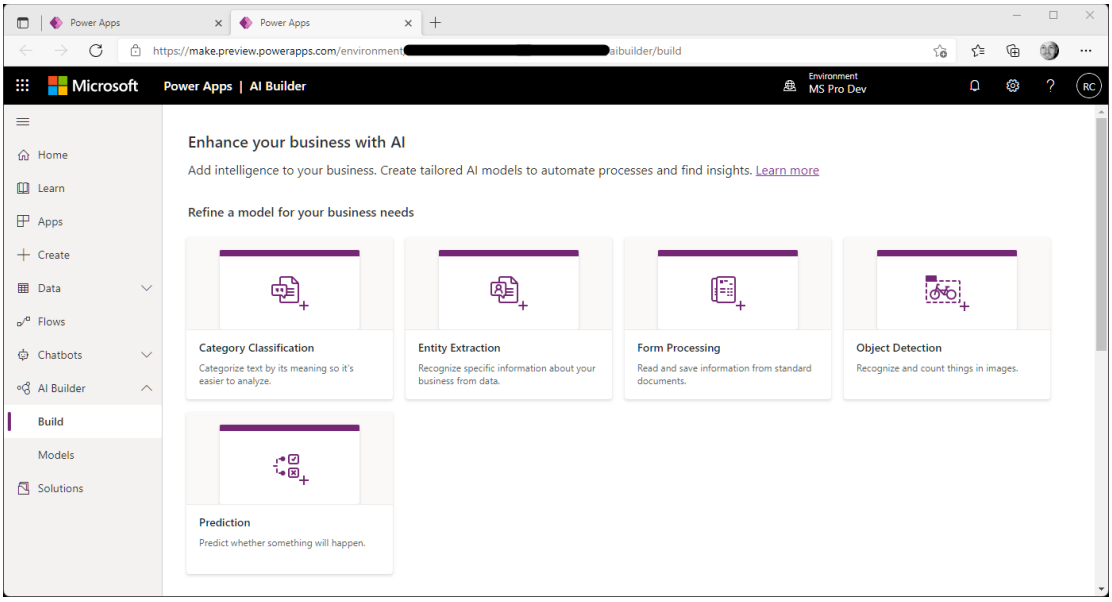
Insert Object Detector

Click the "Add an item from the Insert pane" link.

On the resulting **Insert** menu, expand "**AI Builder**" and select "**Object detector**."

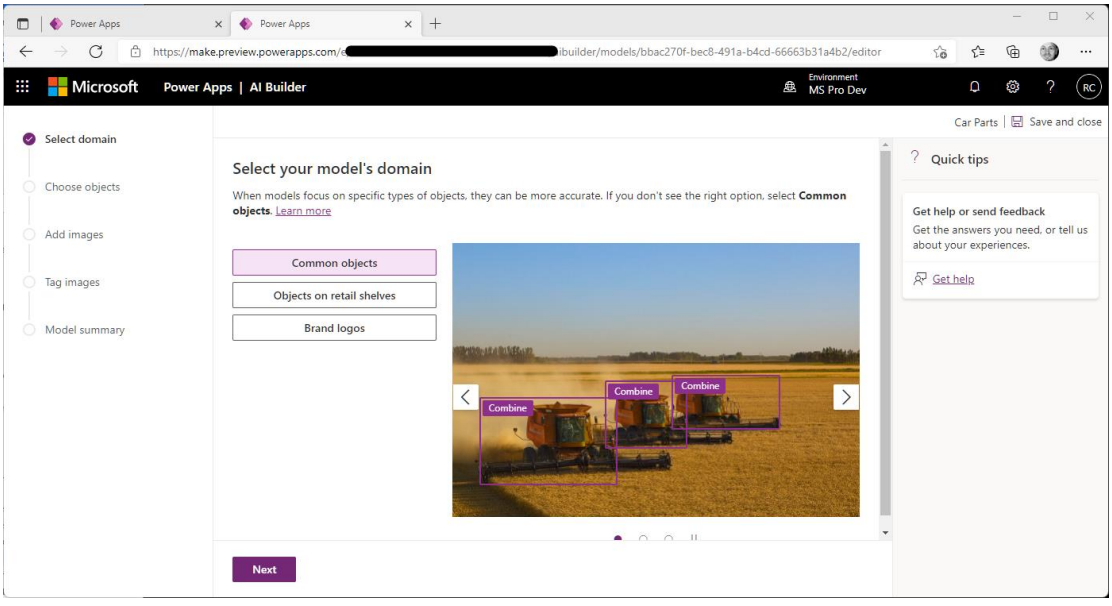
On the resulting pop-out, click "**+ New model**."

Create AI Model

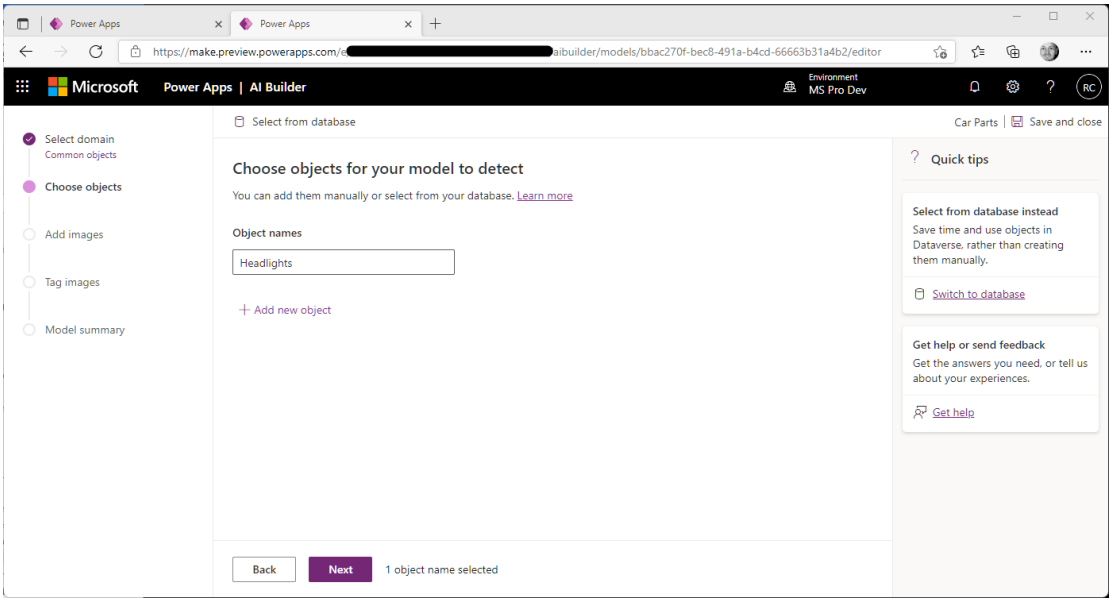


On the new “Power Apps | AI Builder”, “Enhance your business with AI” form, click the “Object Detection” button.

Name the AI model and then click the **Create** button.



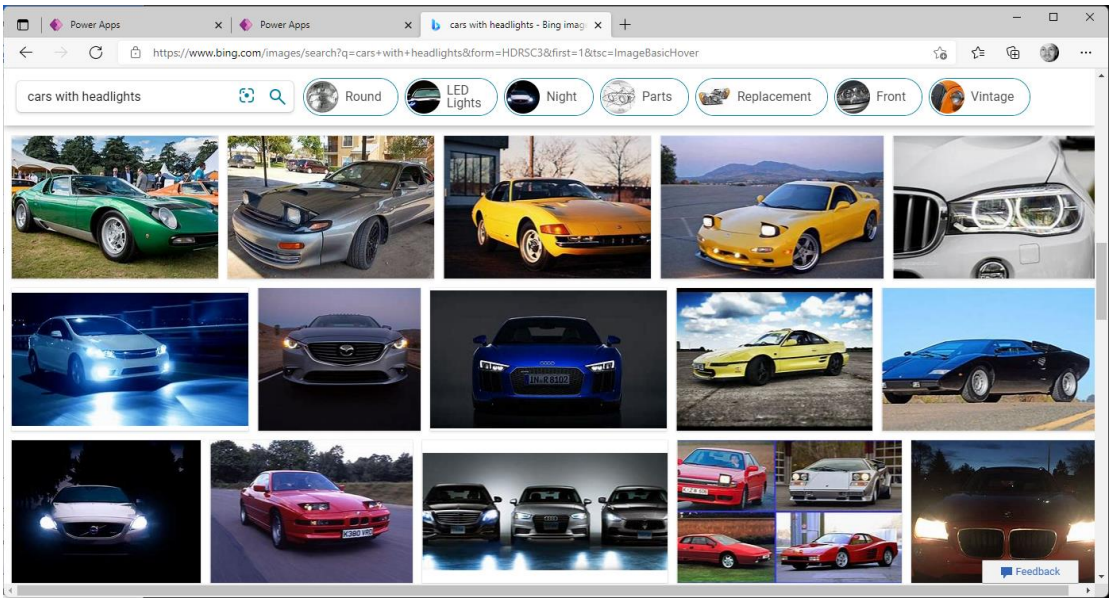
Complete the “**Select your model’s domain**” form, click the “**Common objects**” and then the “**Next**” button.



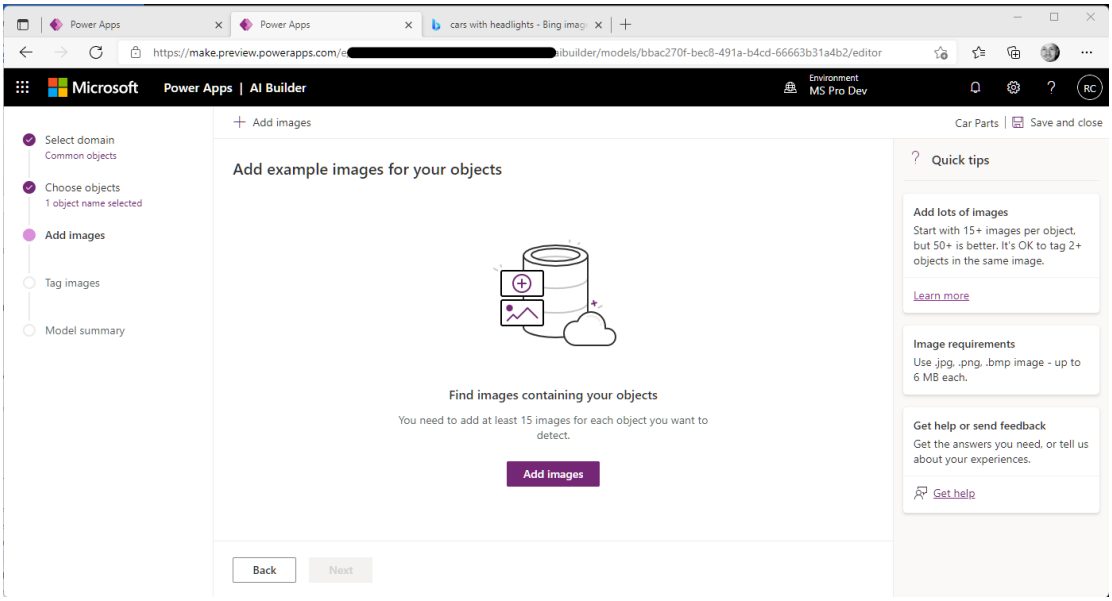
Complete the "**Choose objects for your model to detect**" form, click "**+ Add new object**", enter an object name and then click the "**Next**" button.

Complete the "**Add example images for your objects**" form, we will add image files that will be used to train the AI model.

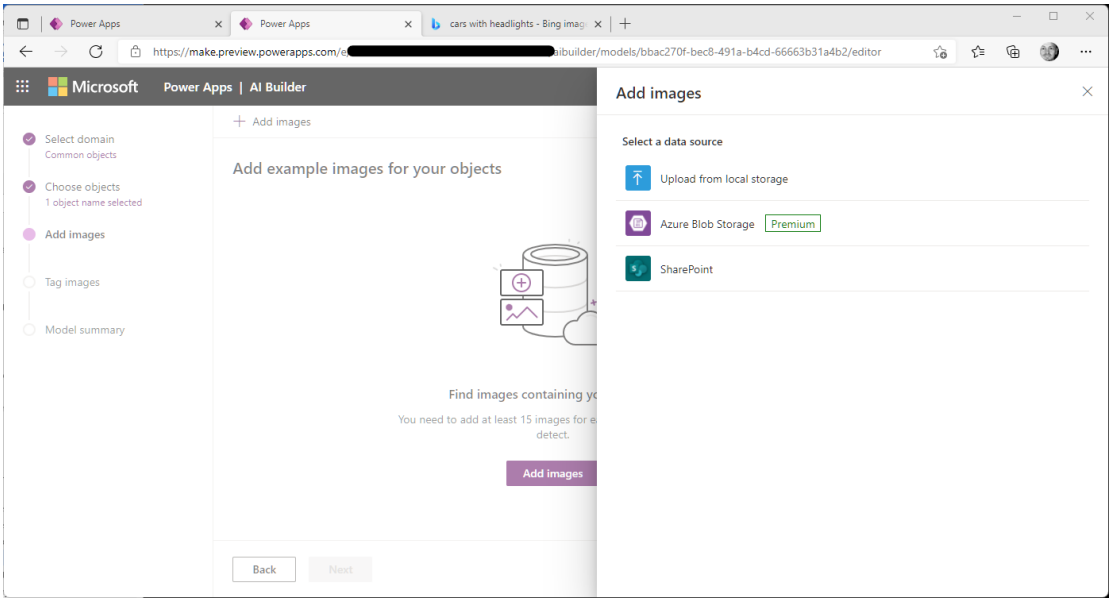
For this exercise, we can pull images of cars (with headlights) from a search engine.



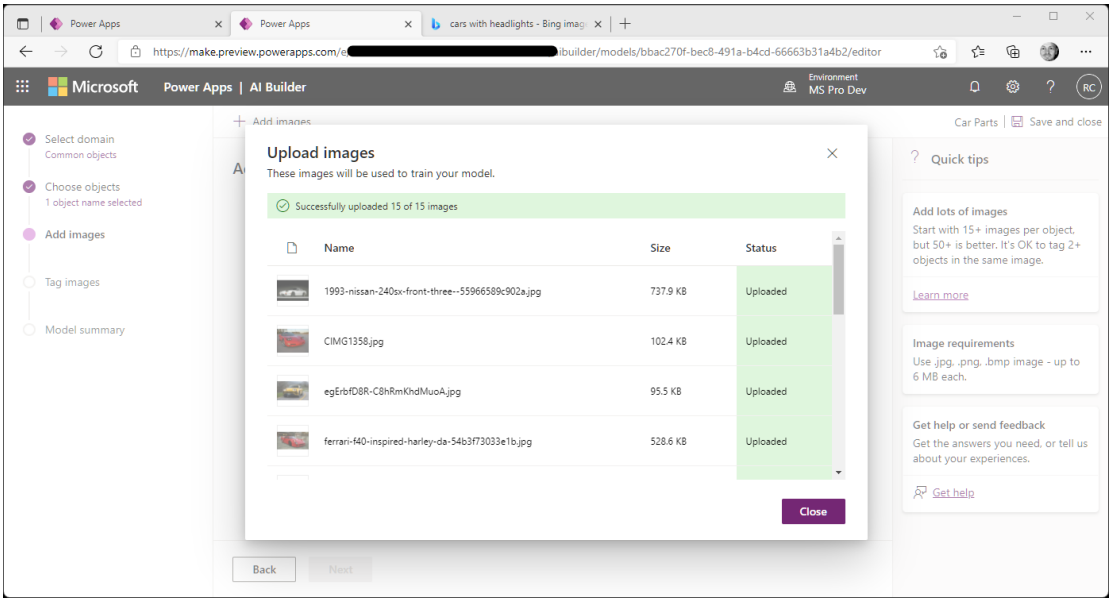
For each of the required 15 images, download an image file to a temporary folder on the Windows desktop.



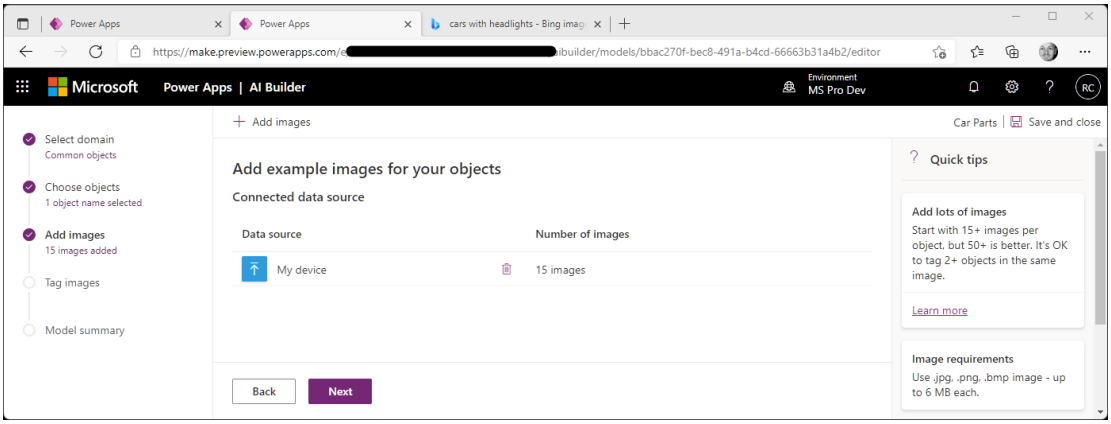
Complete the “Add example images for your objects” form, click the “Add images” button.



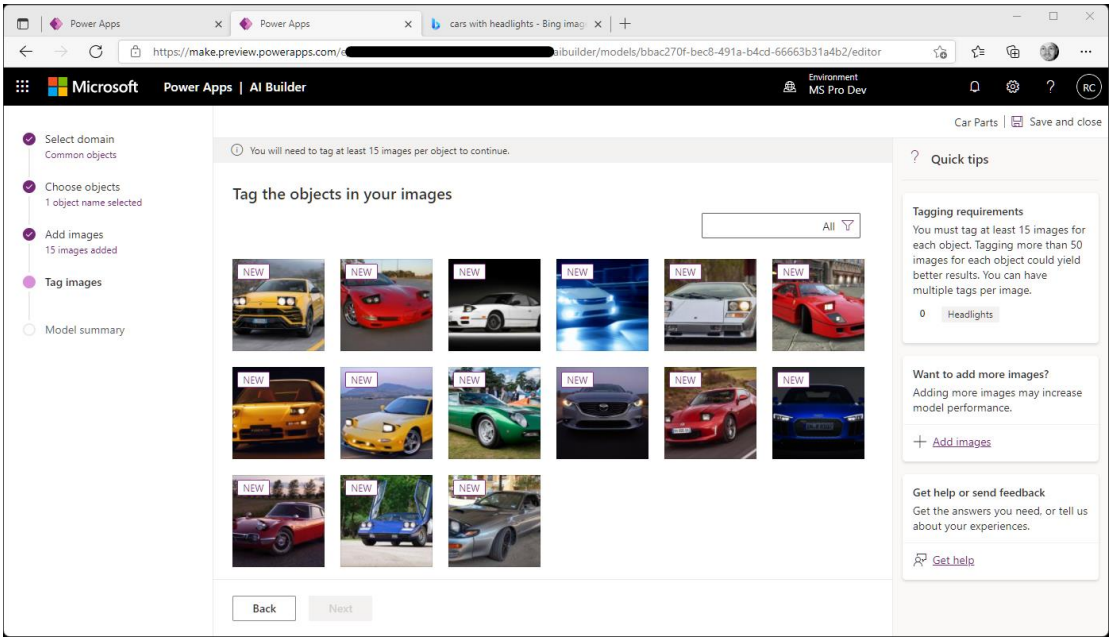
On the resulting “**Add images**” pop-out, click “**Upload from local storage**.” Select the Example Images downloaded to your desktop.



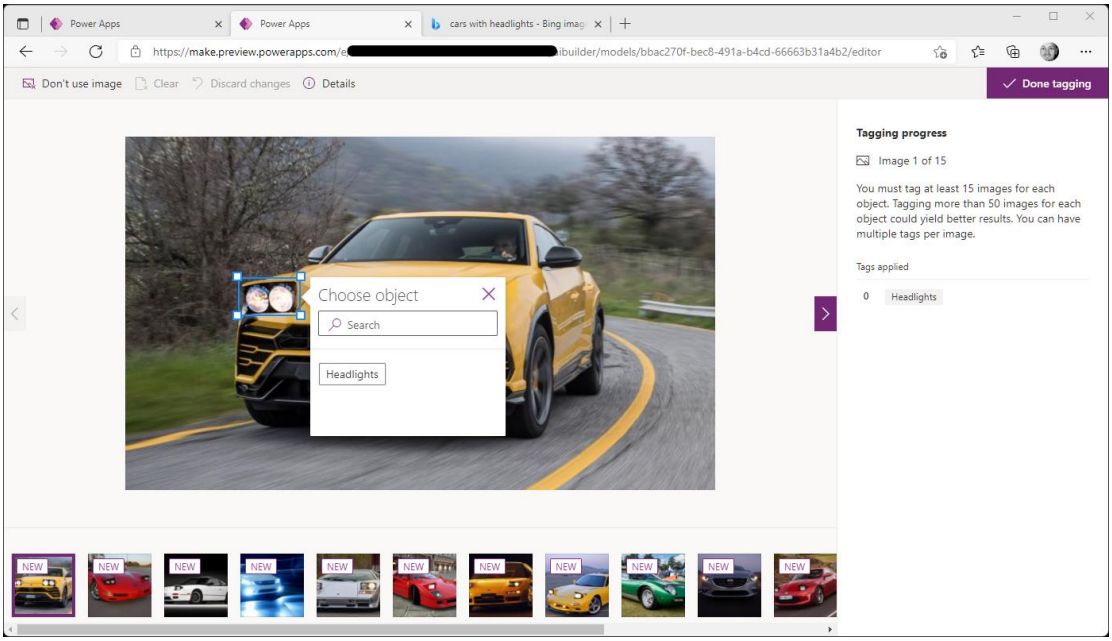
Review the items in the "**Upload images**" pop-up, then click the "**Upload 15 images**" button. Click **Close**.



Back on the "**Add example images for your desktop**" form, confirm "**Number of images**" shows 15 or more images, then click **Next**.

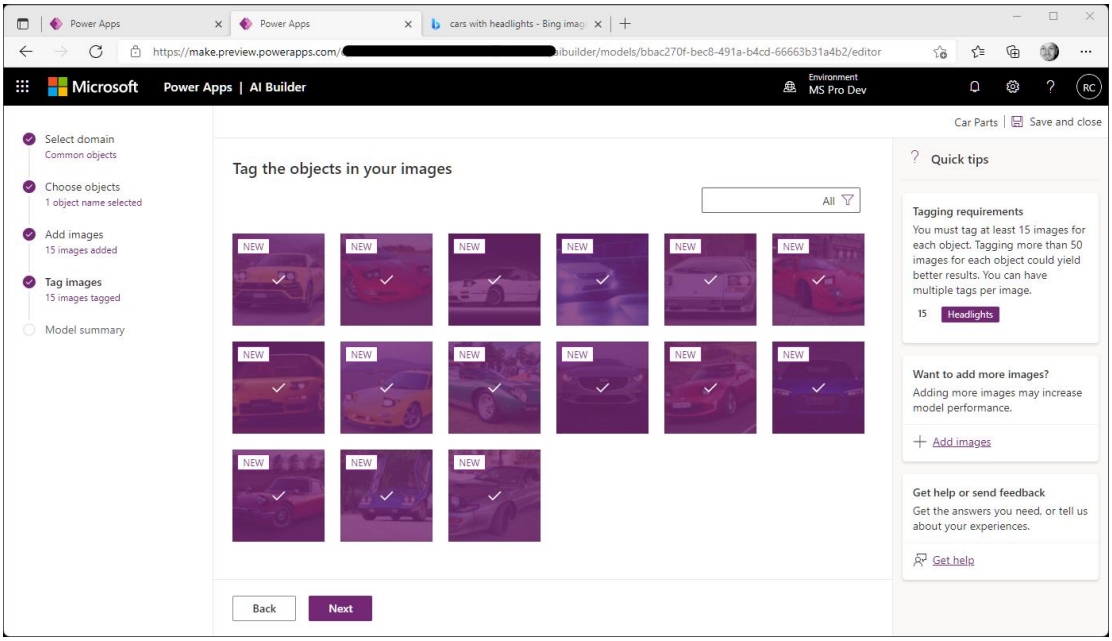


Select the first image on the "**Tag the objects in your images**" page.

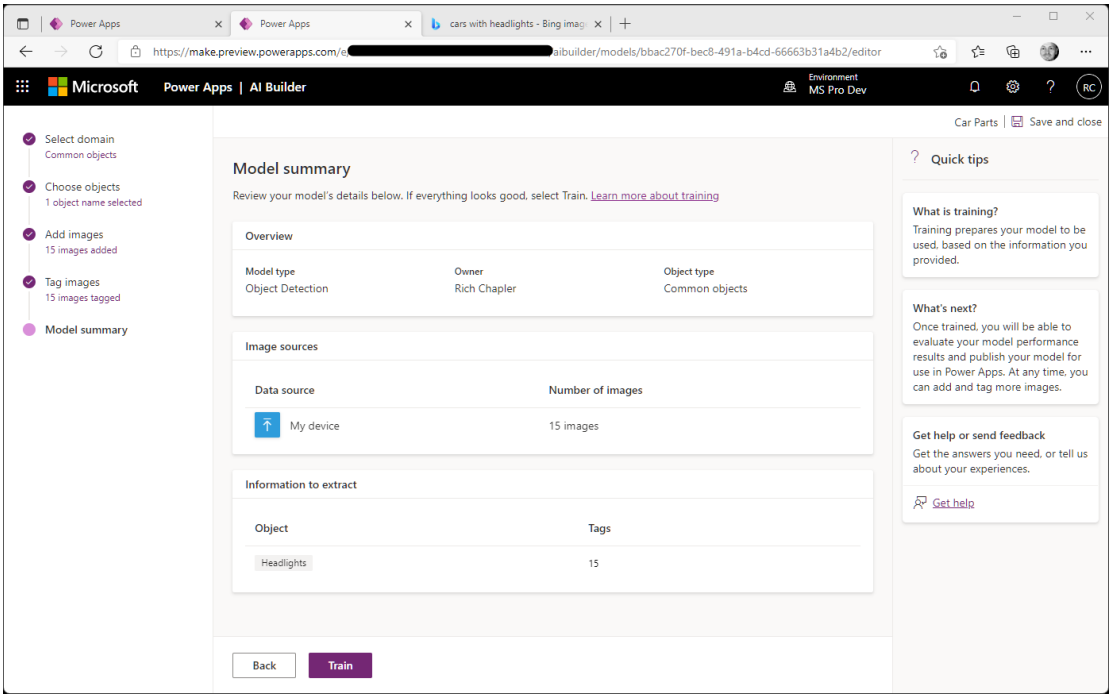


Drag a rectangle around headlights in the image, then click the Headlights button on the "**Choose object**" pop-up.

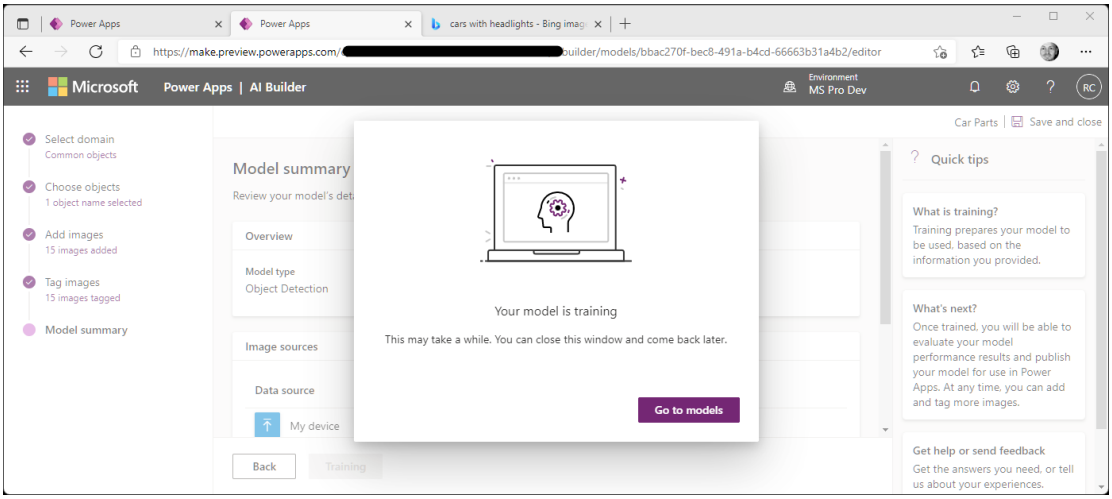
Repeat for all taggable items, click **>** to move to through images, and then click the "**Done tagging**" button in the upper-right.



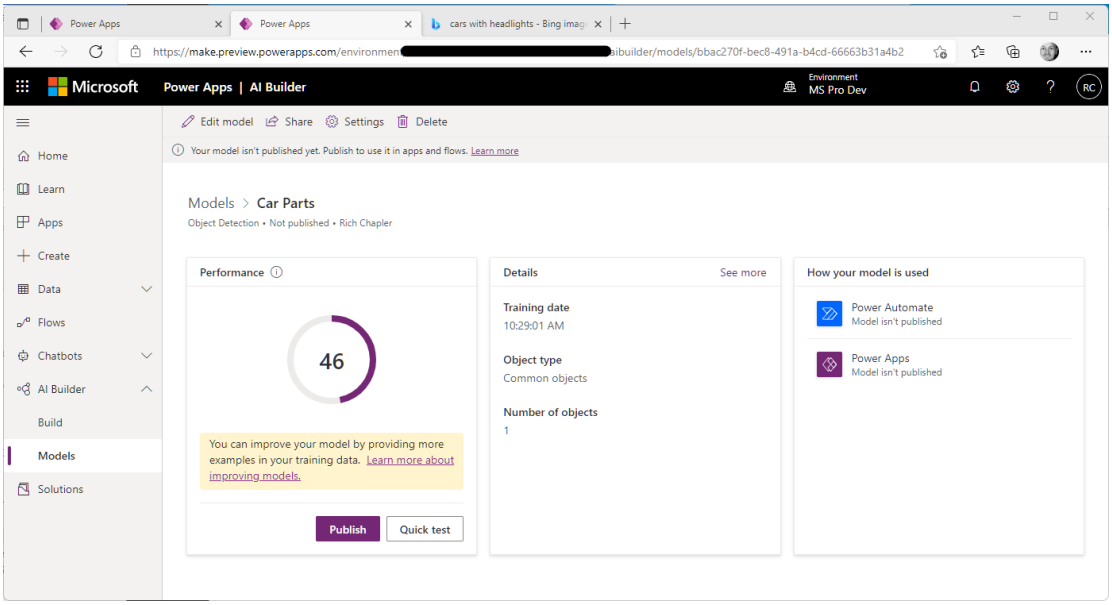
Click the **Next** button.



Review the "**Model Summary**" and then click the **Train** button.



Click the "**Go to models**" button. After Status changes to **Trained**, navigate to the model.

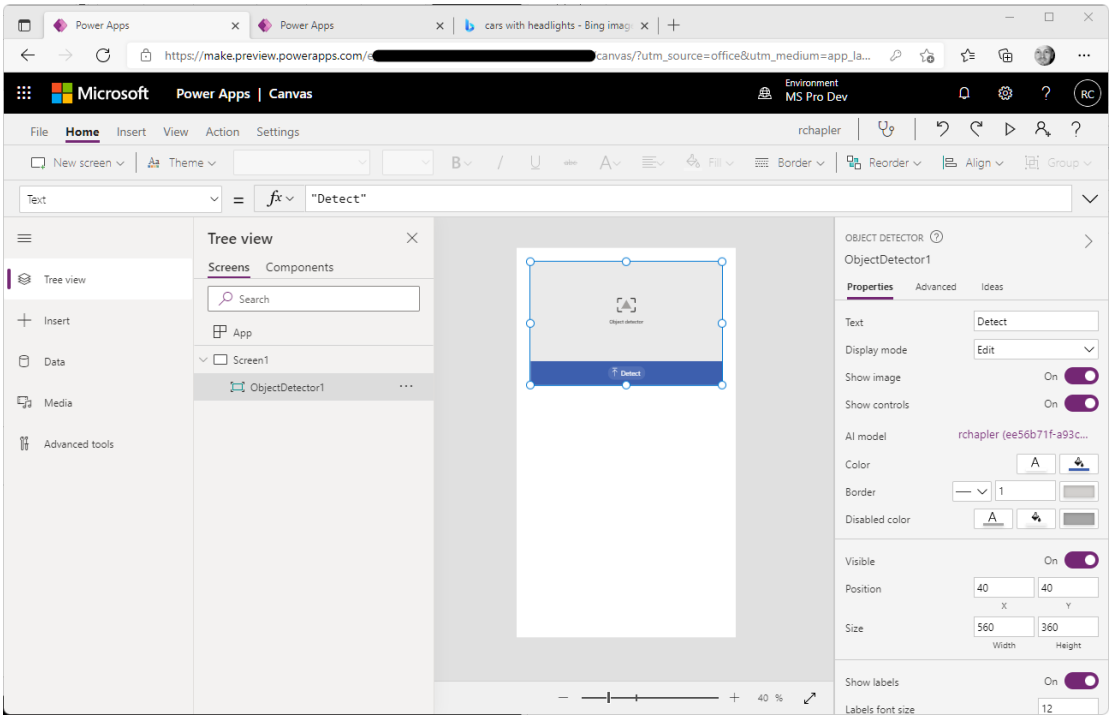


In the snip above, we see that **Performance** {i.e., model performance expectation %} of 46 for the example images included in training.

In the future, you can improve model quality by including more images and re-training the model.

Click the **Publish** button.

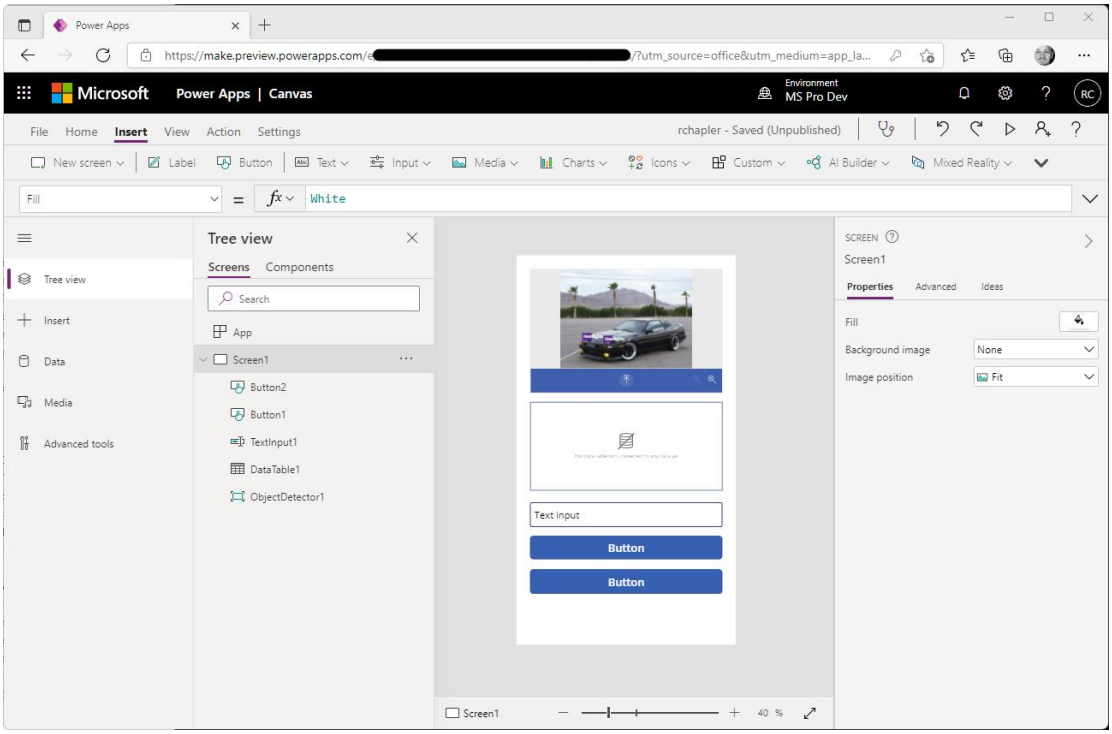
Return to Power Apps and complete the OjbectDetector1 component by selecting your newly created model.



Note: Consider resizing the component to improve visibility.

Insert / Configure Controls

Return by closing Preview Mode.

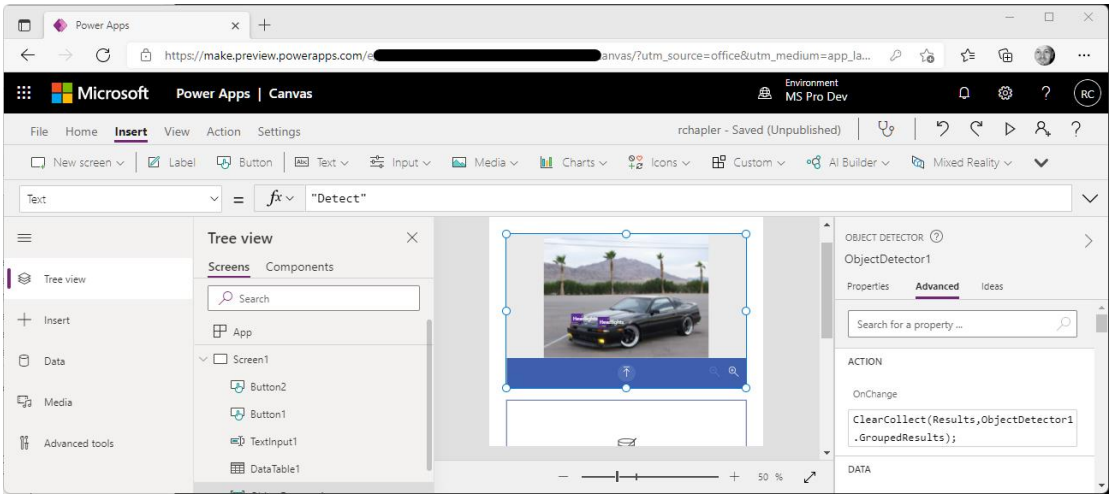


Insert the following controls:

- **Data Table** … users will use this control to view the results generated by AI Builder
- **Text Input** … users will use this control to enter comments about the analyzed image
- "Submit" **Button** … users will use this control to save the image and comment to Azure SQL
- "Reset" **Button** … users will use this control to clear previously entered values

You might note that I employed formatting {e.g., control width, border size, color, etc.} to enhance usability.

ObjectDetector1

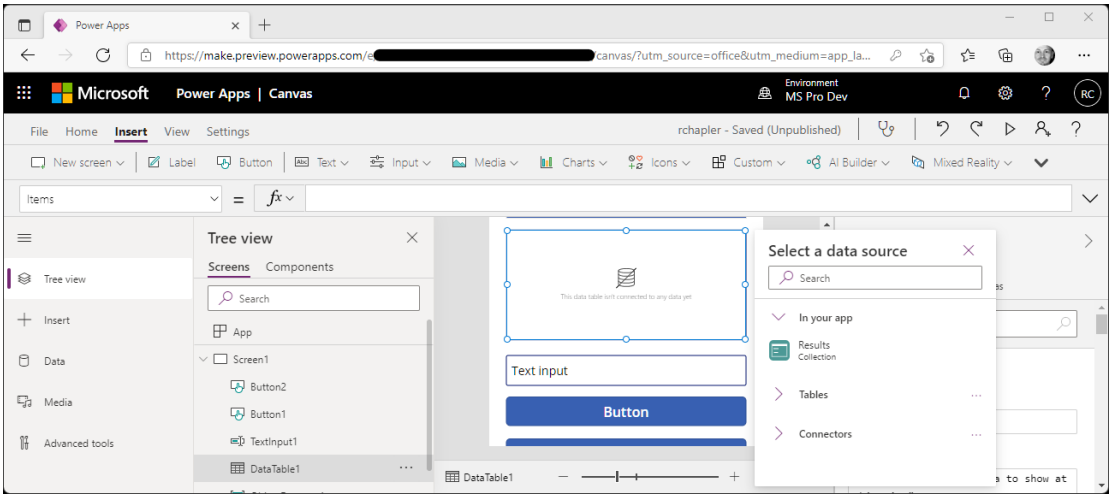


Click on **ObjectDetector1** in the "**Tree view**" pane. On the resulting pop-out, click the **Advanced** tab.

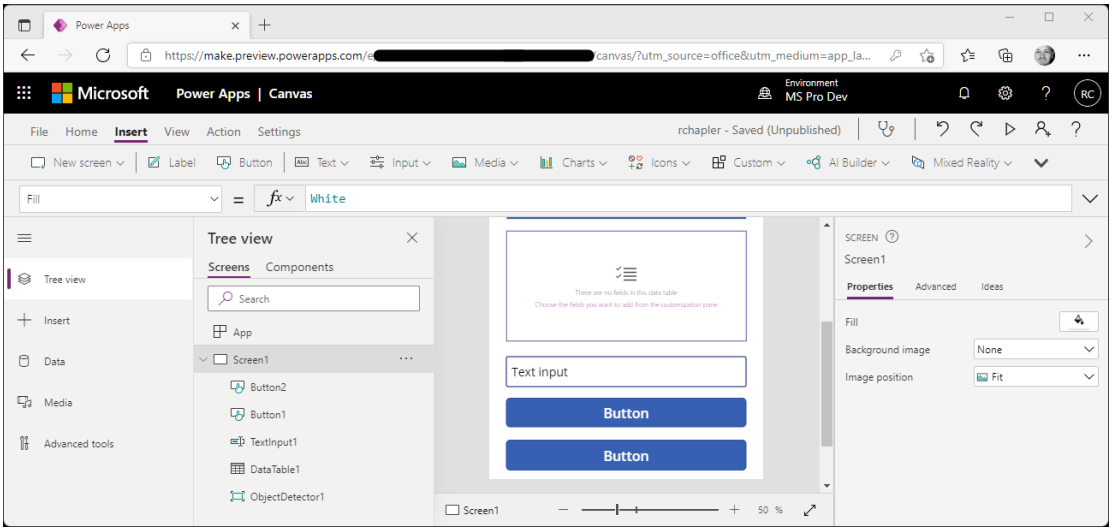
Paste the following logic into **Action** | **OnChange**:

```
ClearCollect(Results,ObjectDetector1.GroupedResults);
```

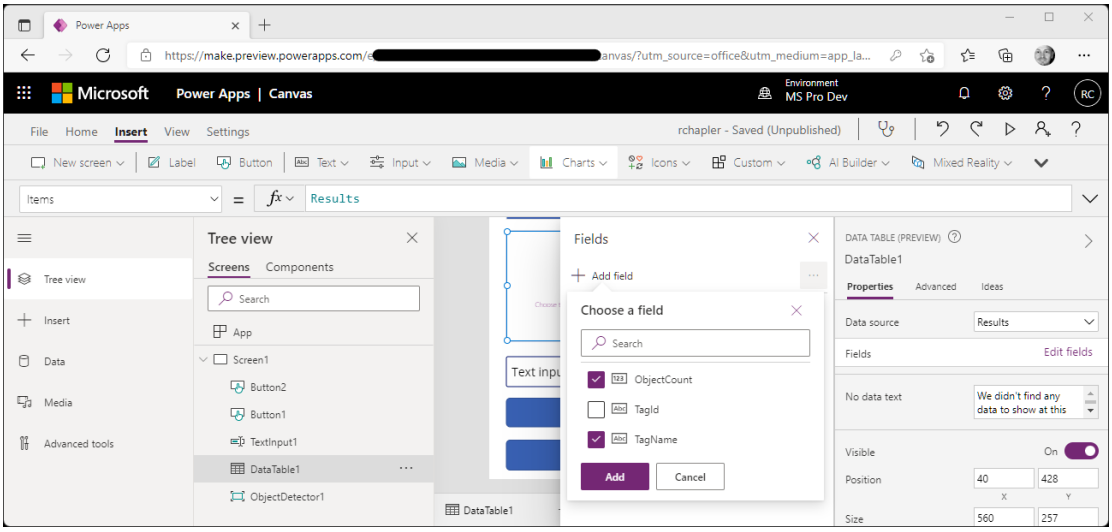
DataTable1



Click on **DataTable1** in the "**Tree view**" pane. On the resulting pop-up, select "**In your app**" | **Results**.



On the updated control, click the "**Choose the fields you want to add…**" link.

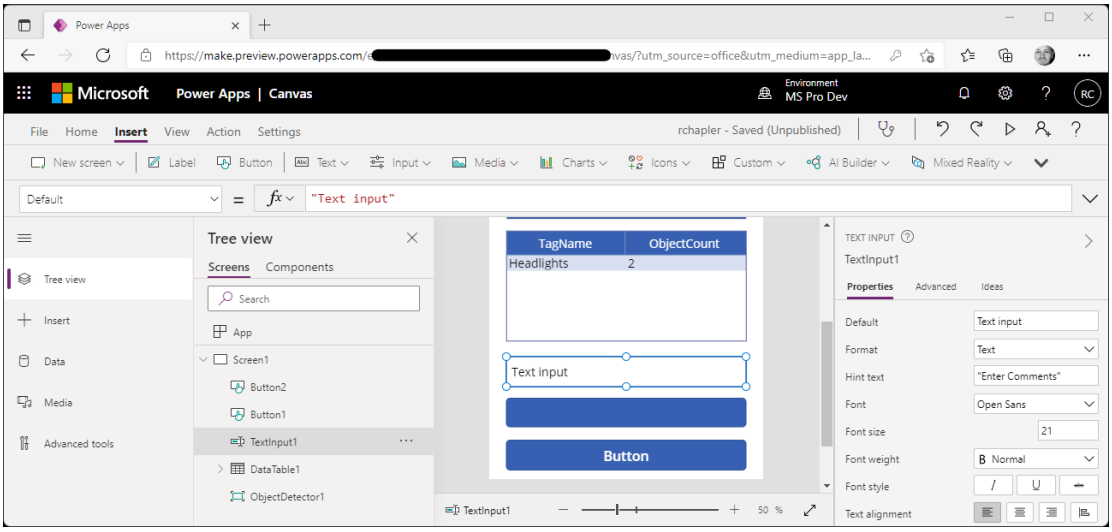


On the resulting **Fields** pop-out, click "**+ Add field**."

In the resulting "**Choose a field**" pop-up, check **TagName** and then **ObjectCount**.

Click the **Add** button.

TextInput1

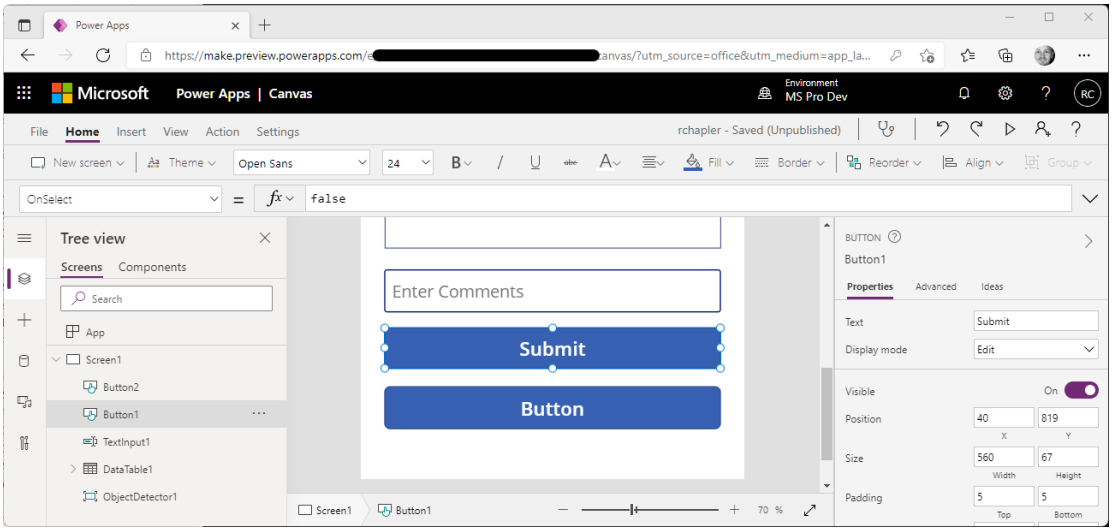


Click on **TextInput1** in the "**Tree view**" pane.

On the resulting "**TEXT INPUT**" pop-out, **Advanced** tab, enter the following items:

Default	""
Hint Text	"Enter Comments"

Button1



Click on **Button1** in the "**Tree view**" pane.

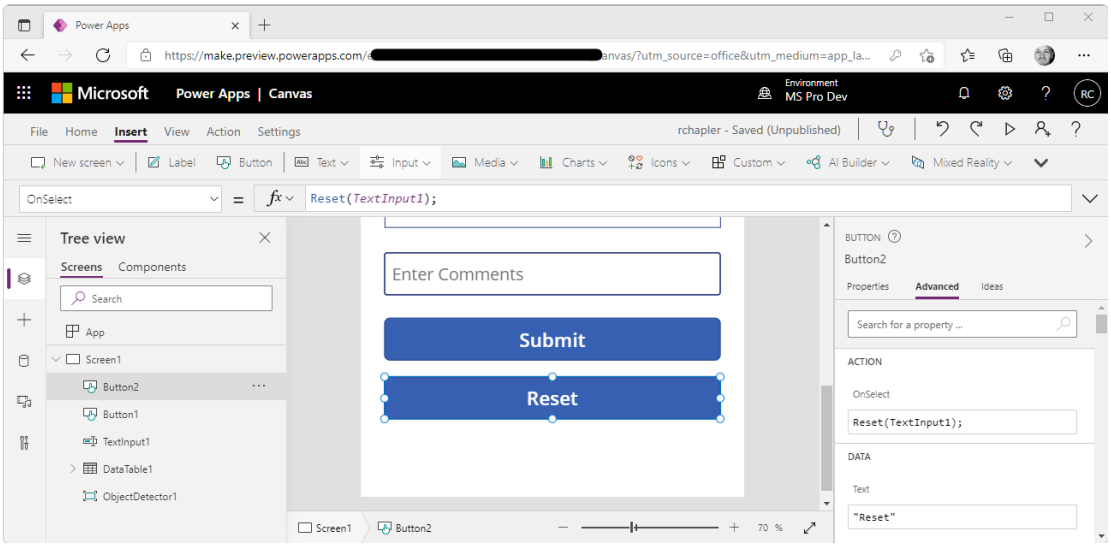
On the resulting pop-out, click the **Properties** tab, and enter the word "Submit" in the **Text** input.

On the resulting pop-out, click the **Advanced** tab,

Paste the following logic into **Action** | **OnSelect**:

```
Patch(myTable,Defaults(myTable),{Name:TextInput1.Text,Picture:ObjectDetector1.OriginalImage})
```

Button2



Click on **Button2** in the "**Tree view**" pane.

On the resulting pop-out, click the **Properties** tab, and enter the word "Reset" in the **Text** input.

On the resulting pop-out, click the **Advanced** tab,

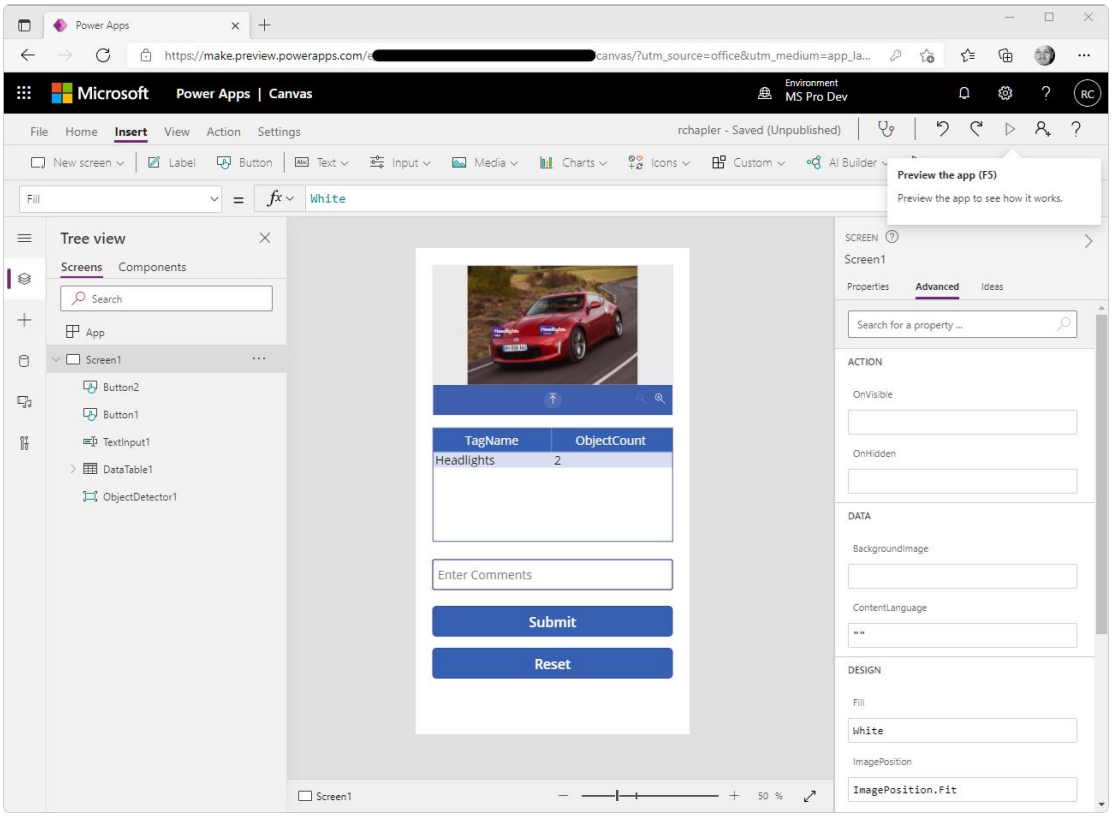
Paste the following logic into **Action** | **OnSelect**:

```
Reset(TextInput1);
```

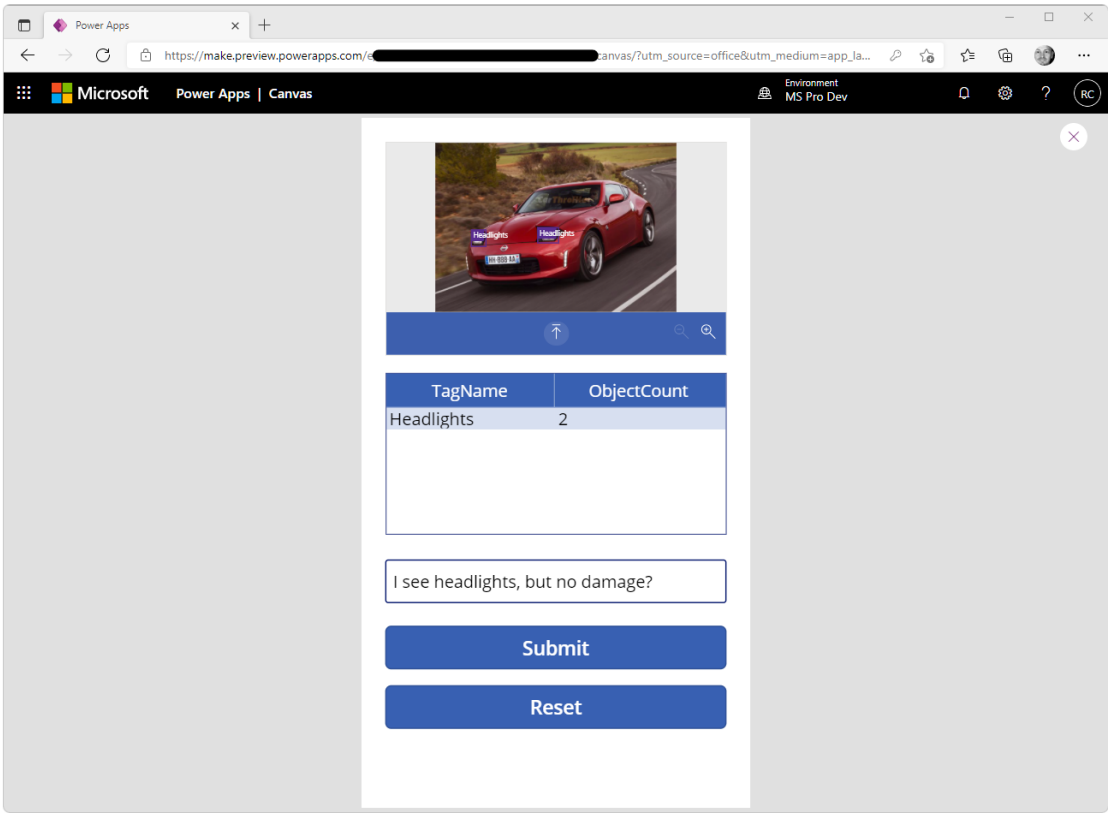
Confirm Success

Click **File** in the menu bar, then "**Save As**", confirm the app name, and finally, click the **Save** button in the bottom-right.

Return to the main form, click the "**Preview the app**" button {i.e., Play icon} in the upper-right.



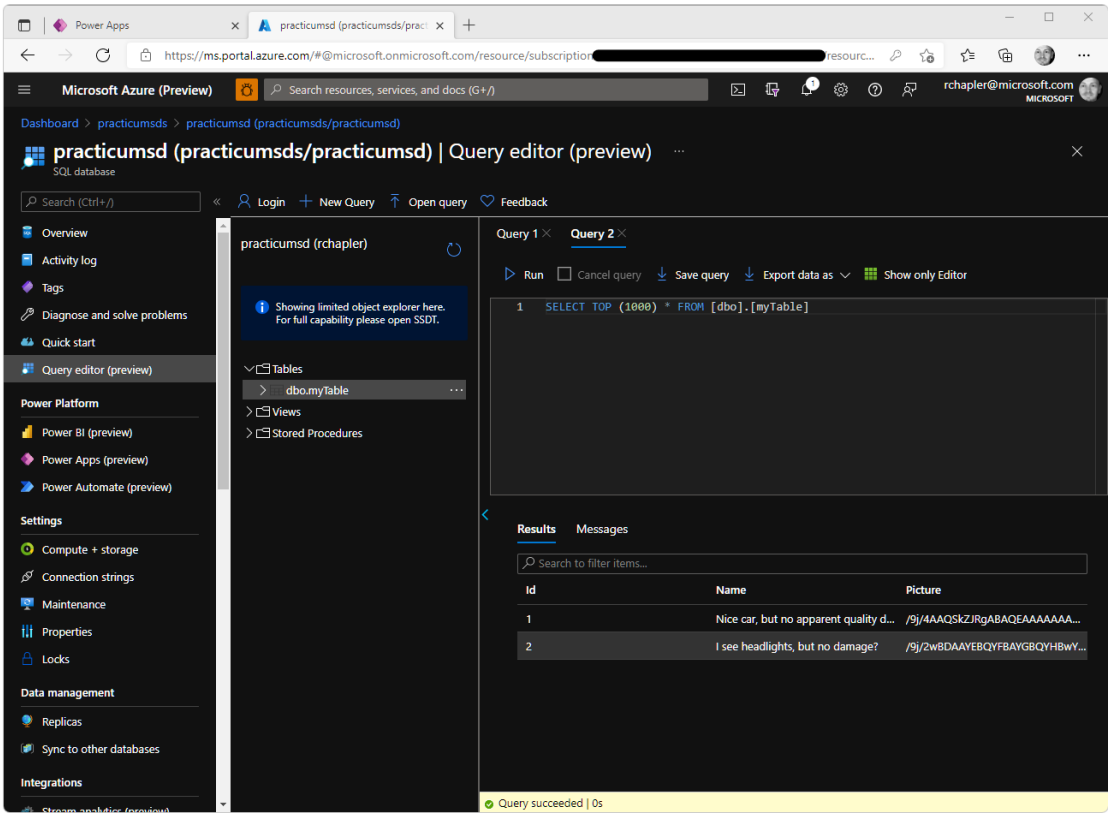
Your app will be presented in a phone simulation.



Click the **Detect** button and select an image file to test objection detection.

Review the resulting values in the data table.

Enter a comment in the text input and click the **Submit** button.



Navigate to **Query Editor** in the Azure SQL Database and confirm that your comment has been added as a new row.

Consider publishing your app and loading it on your phone as one additional confirmation of success.

Audit Usage

Follow these instructions to support regular audit of and alerting on key data products.

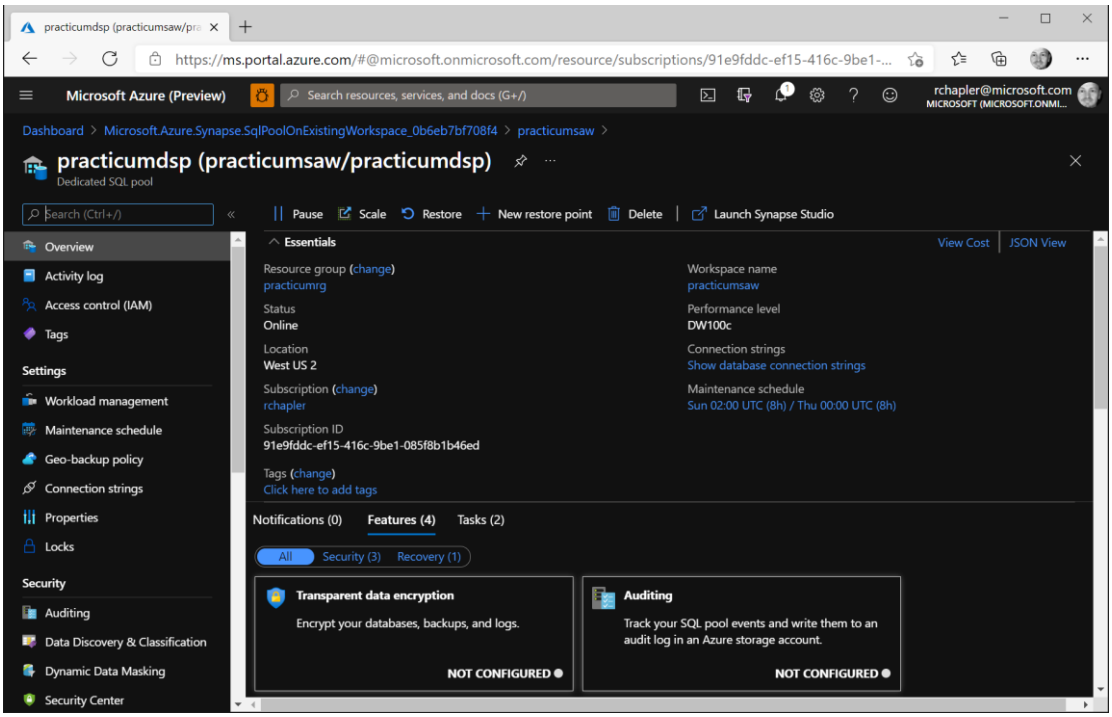
Prerequisites

This solution requires the following resources:

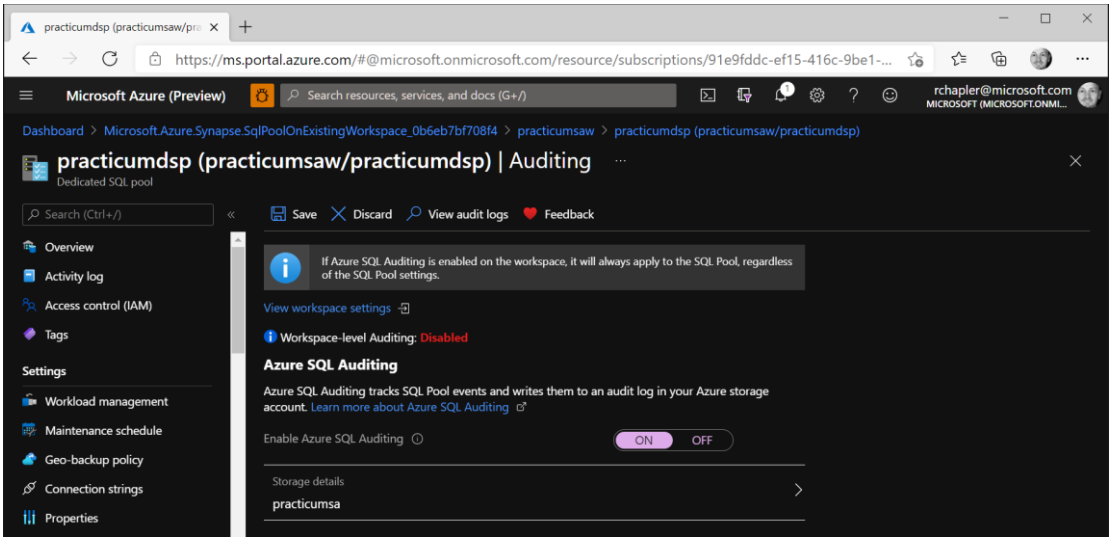
- Synapse

Configure Auditing

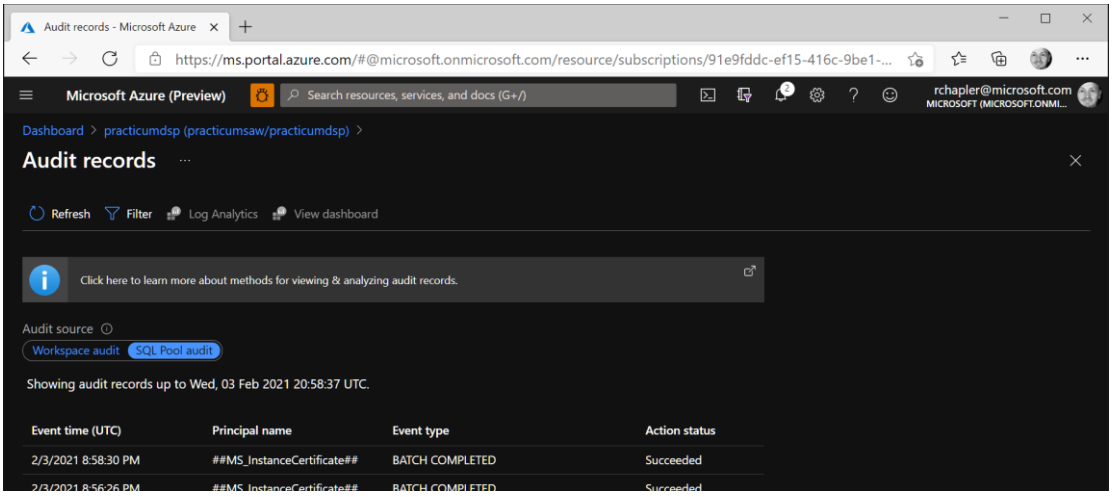
Navigate to the **Overview** page in Synapse, Dedicated SQL Pool.



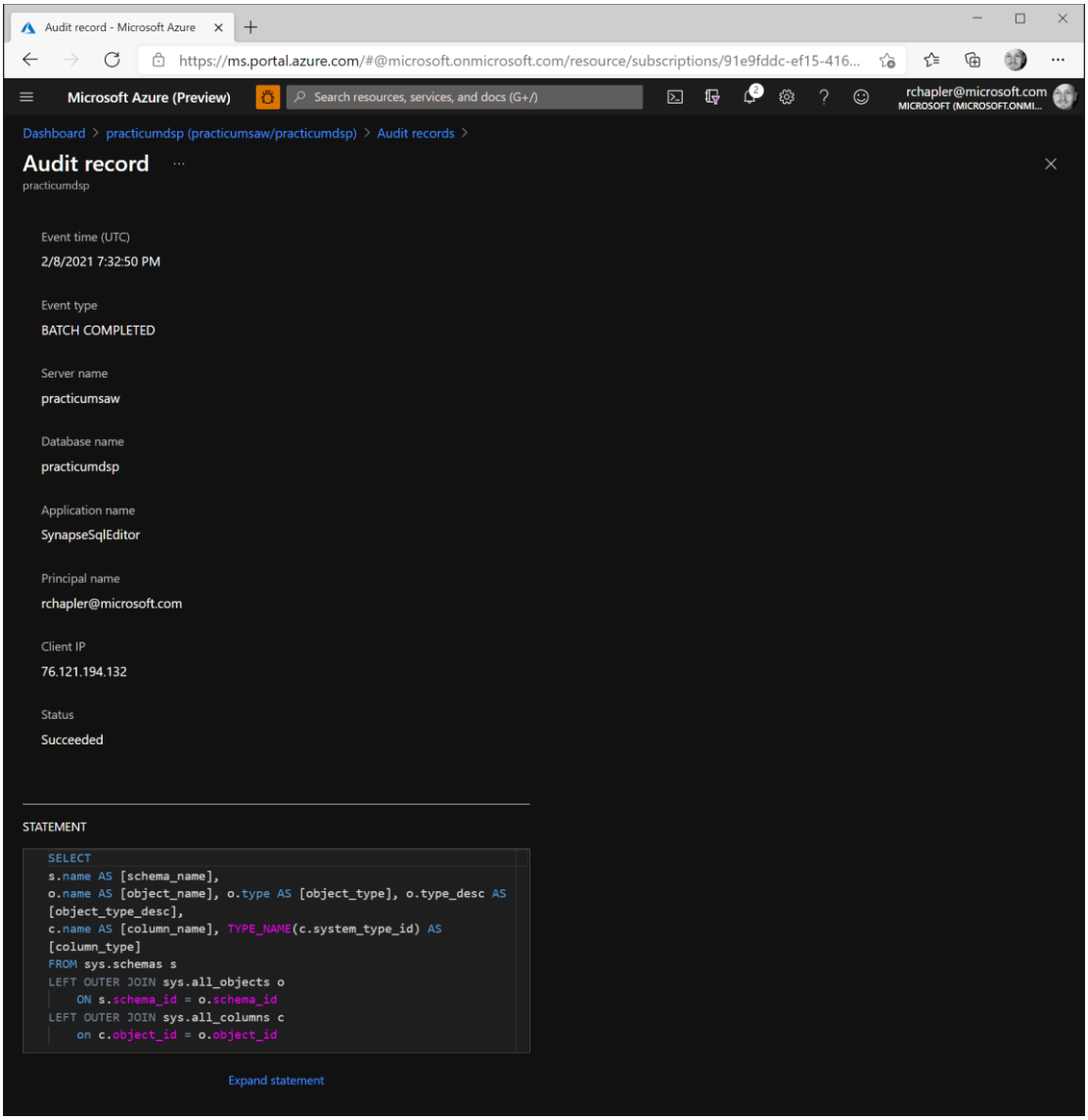
Click the **Auditing** button.



Flip "**Enable Azure SQL Auditing**" switch to **ON** position. Click to configure "**Storage details**." **Save** changes and then click "**View audit logs**."



Click into one of "**BATCH COMPLETED**" audit records.


Audit record - Microsoft Azure
https://ms.portal.azure.com/#@microsoft.onmicrosoft.com/resource/subscriptions/91e9fddc-ef15-416...
Microsoft Azure (Preview)
Search resources, services, and docs (G+/)
rchapler@microsoft.com
MICROSOFT (MICROSOFT.ONMI...
Dashboard > practicumdsp (practicumsaw/practicumdsp) > Audit records >
Audit record
practicumdsp
Event time (UTC)
2/8/2021 7:32:50 PM
Event type
BATCH COMPLETED
Server name
practicumsaw
Database name
practicumdsp
Application name
SynapseSqlEditor
Principal name
rchapler@microsoft.com
Client IP
76.121.194.132
Status
Succeeded
STATEMENT
SELECT
s.name AS [schema_name],
o.name AS [object_name], o.type AS [object_type], o.type_desc AS
[object_type_desc],
c.name AS [column_name], TYPE_NAME(c.system_type_id) AS
[column_type]
FROM sys.schemas s
LEFT OUTER JOIN sys.all_objects o
ON s.schema_id = o.schema_id
LEFT OUTER JOIN sys.all_columns c
on c.object_id = o.object_id
Expand statement

Governance

Discover Data

Data discovery is “an organic process based on communal knowledge”, characterized by the following questions:

- **Search** … “does the data that I need exist?”
- **Location** … “how do I connect to Data Product X?”
- **Related** … “are there related data products that I might find valuable?”
- **Contacts** … “who are the owners and experts for Data Product X?”

In this section, we will configure Purview and successfully respond to these questions.

Prerequisites

This solution requires the following resources:

- Purview
- SQL

Register Source

Navigate to Purview in the Azure portal.

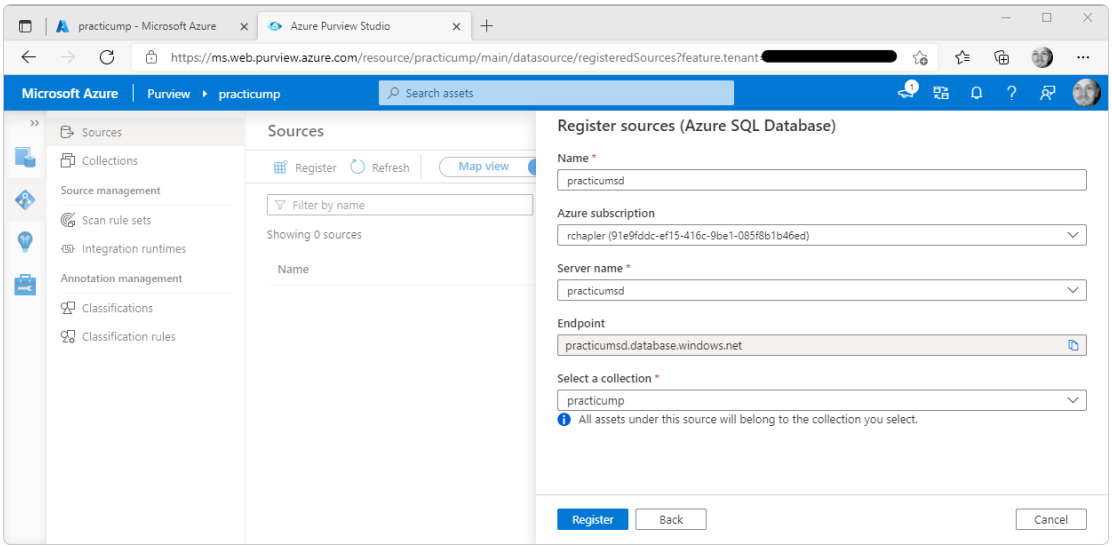
Click the “**Open Purview Studio**” button.

Click the **Data Map** icon on the navigation pane.

Click the **Register** button on the resulting page.

Click the **Azure** tab on the “**Register sources**” pop-out.

Click to check the “**Azure SQL Database**” option and then click the **Continue** button.



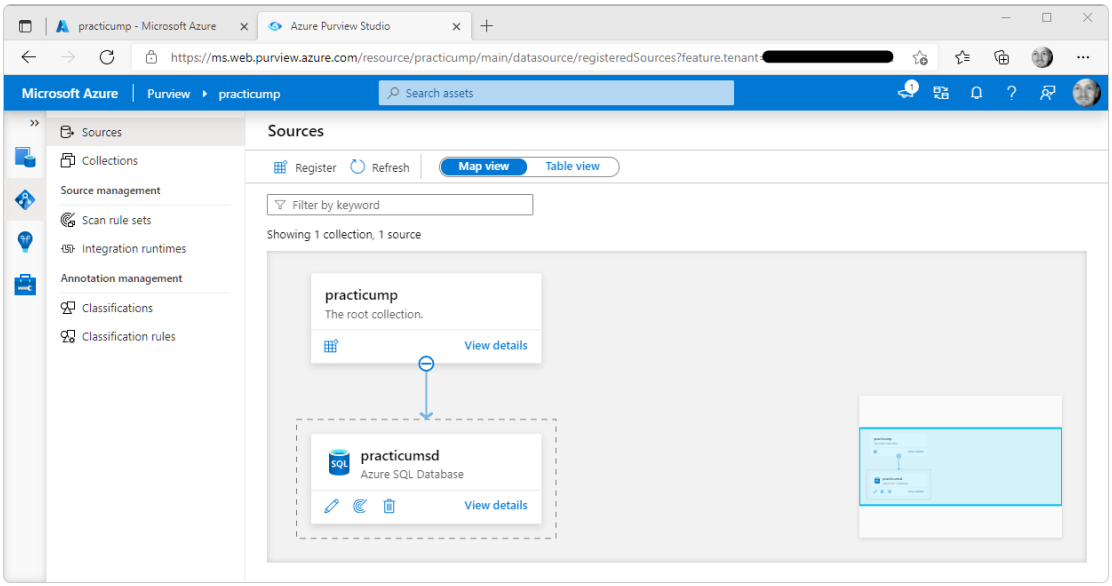
Complete the "**Register sources (Azure SQL Database)**" pop-out, including:

Server Name	Enter the name of your SQL Database Server
Endpoint	Confirm the populated value
Select a Collection	Leave default value

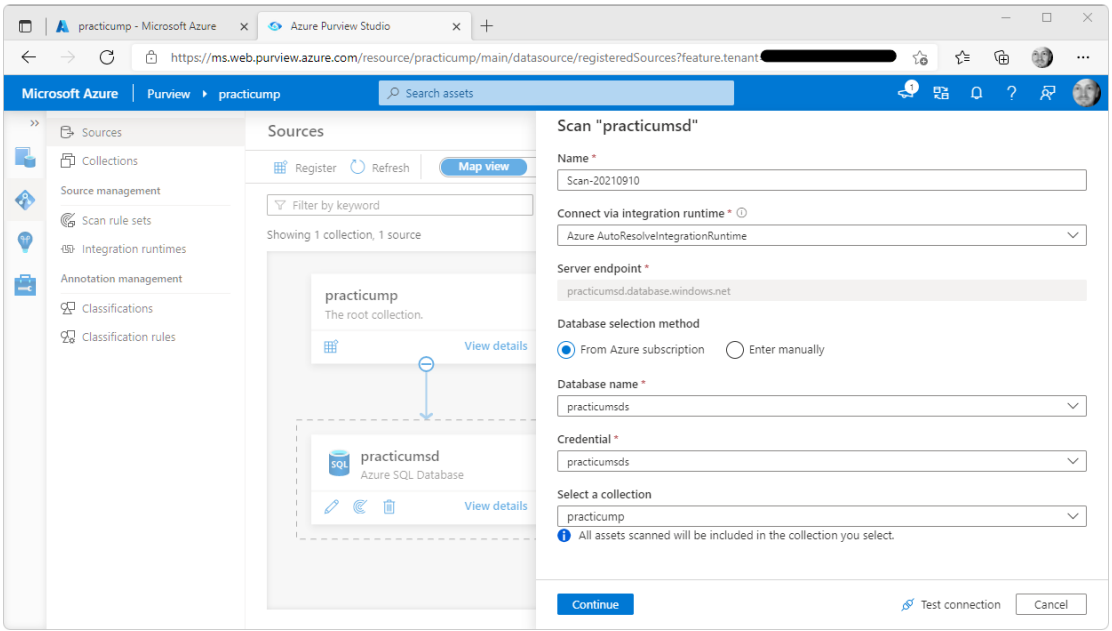
Click the **Register** button.

Scan Data

Navigate to **Sources**, **“Map view**.”



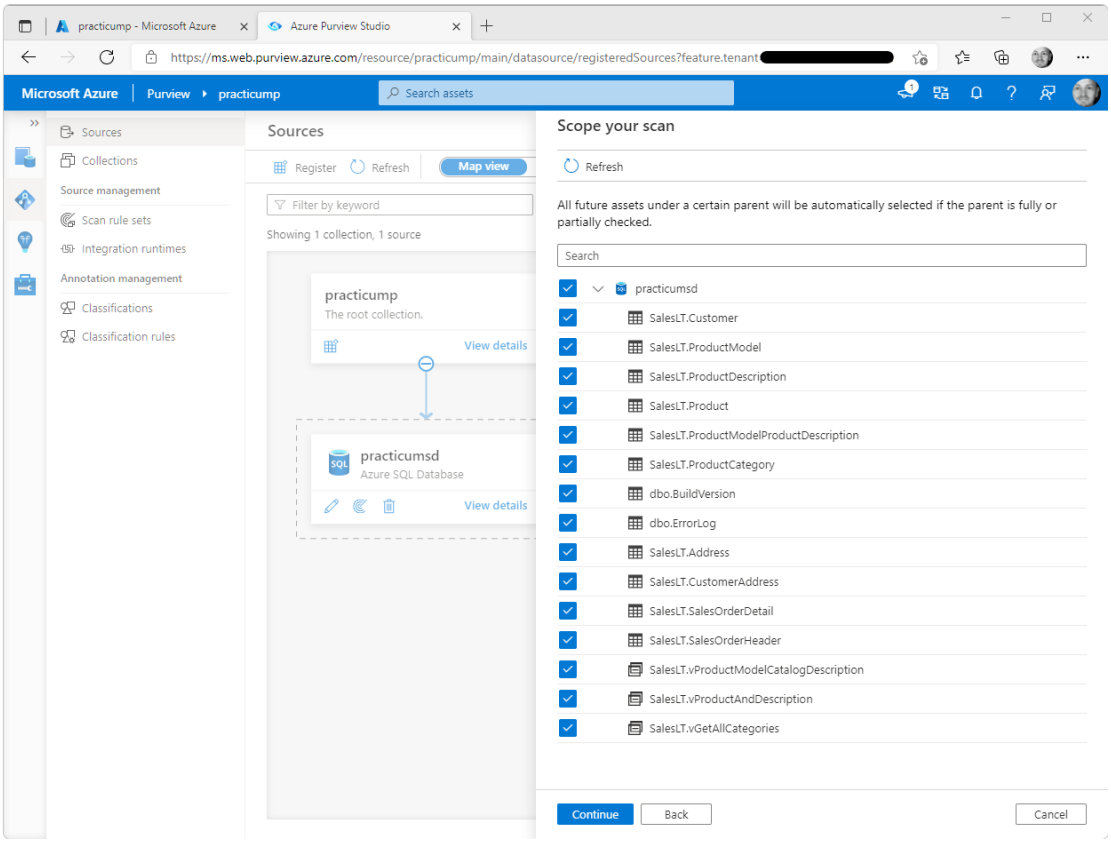
Click the “**New scan**” icon on the newly created item.



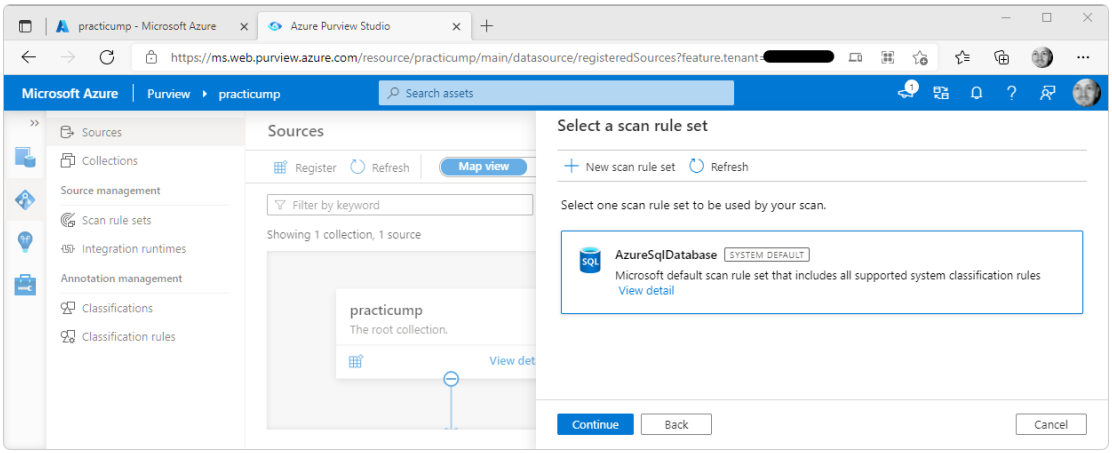
Complete the “**Scan**…” pop-out, including:

Database Selection Method	Select the “**From Azure subscription**” radio button
Database Name	Select your Azure SQL database
Credential	Select your “**SQL Authentication**” credential

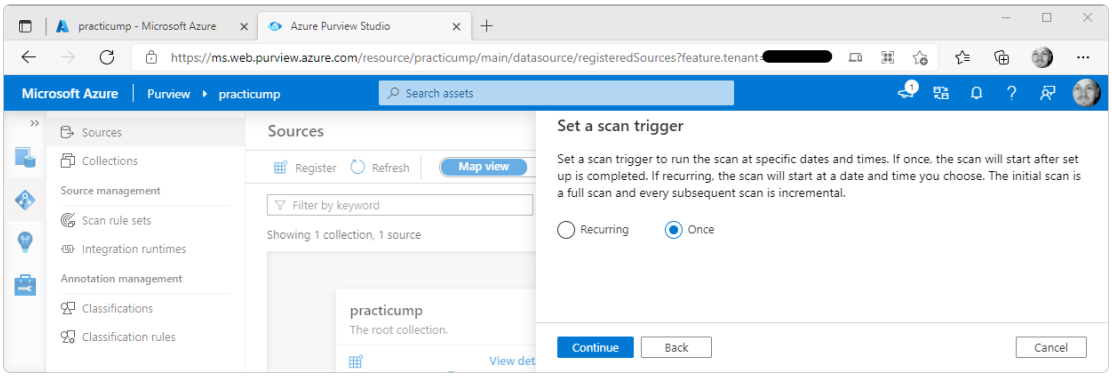
Click “**Test connection**” to confirm successful configuration and then click the **Continue** button.



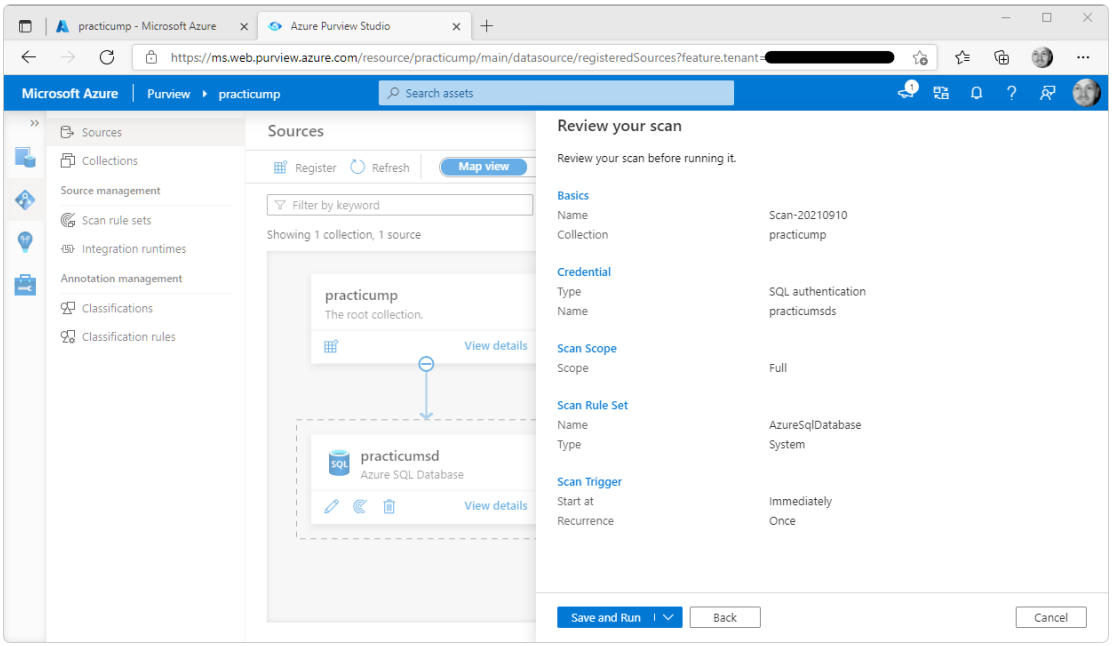
Review and select desired data products from those listed in the “**Scope your scan**” pop-out and then click the **Continue** button.



Confirm selection of the default Scan Rule Set on the "**Select a scan rule set**" pop-out and then click the **Continue** button.



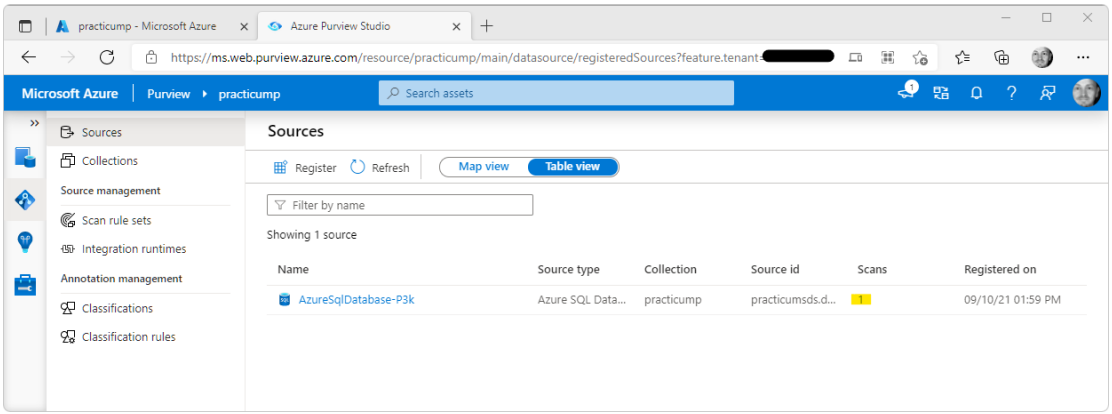
Click the **Once** radio button on the "**Set a scan trigger**" pop-out and then click the **Continue** button.



Review scan settings and then click the “**Save and Run**” button.

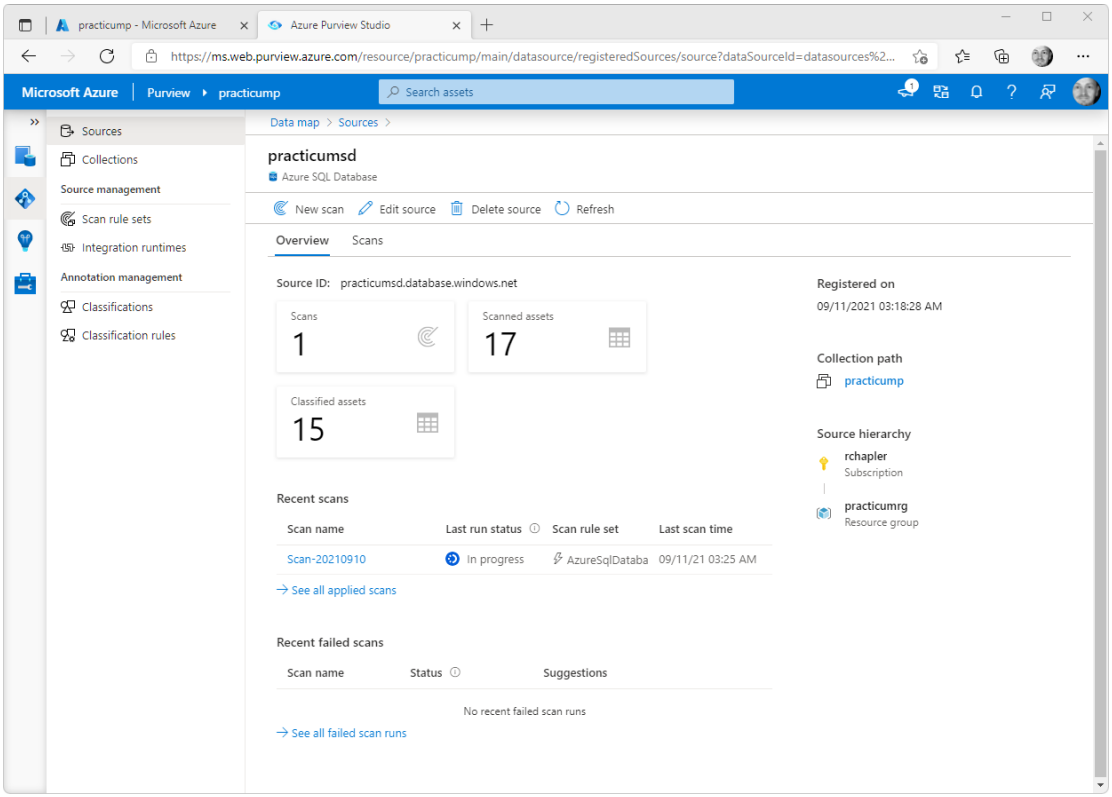
Confirm Scan

Navigate to Sources, “Table view.”



Note that the **Scans** column now shows 1.

Click on the **Name** link to drill through to details. Allow time for processing.

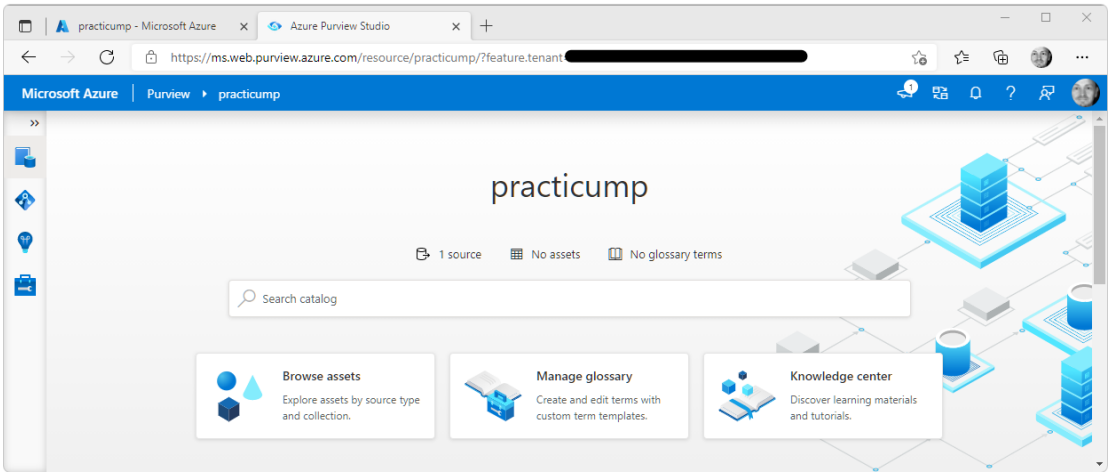


Browse Assets

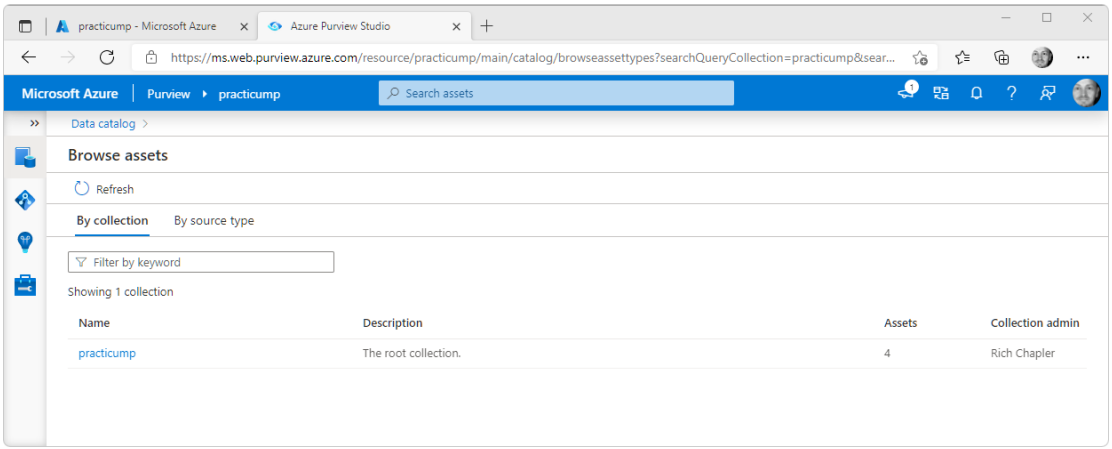
After scanning, it takes time for **Asset Insights** to reflect new data products.

We do not have to wait to browse data products, however.

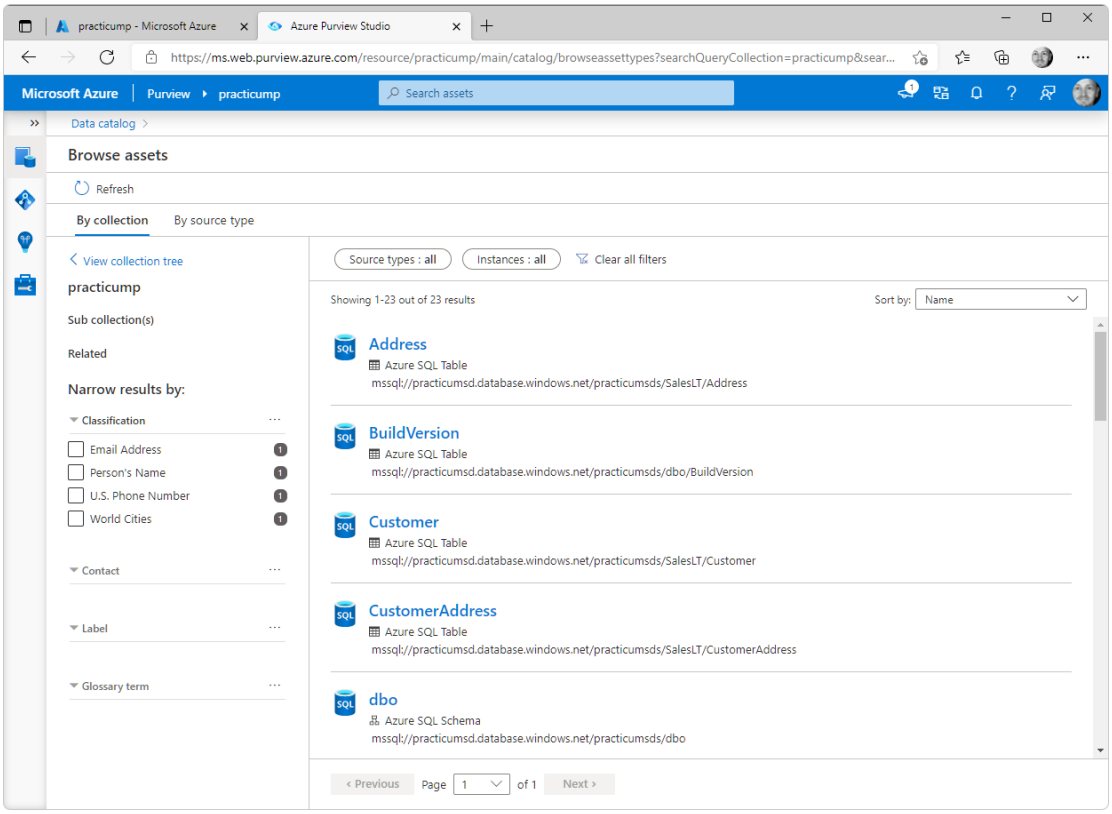
Click the **Data Catalog** icon on the navigation pane.



Click on the "**Browse assets**" button.

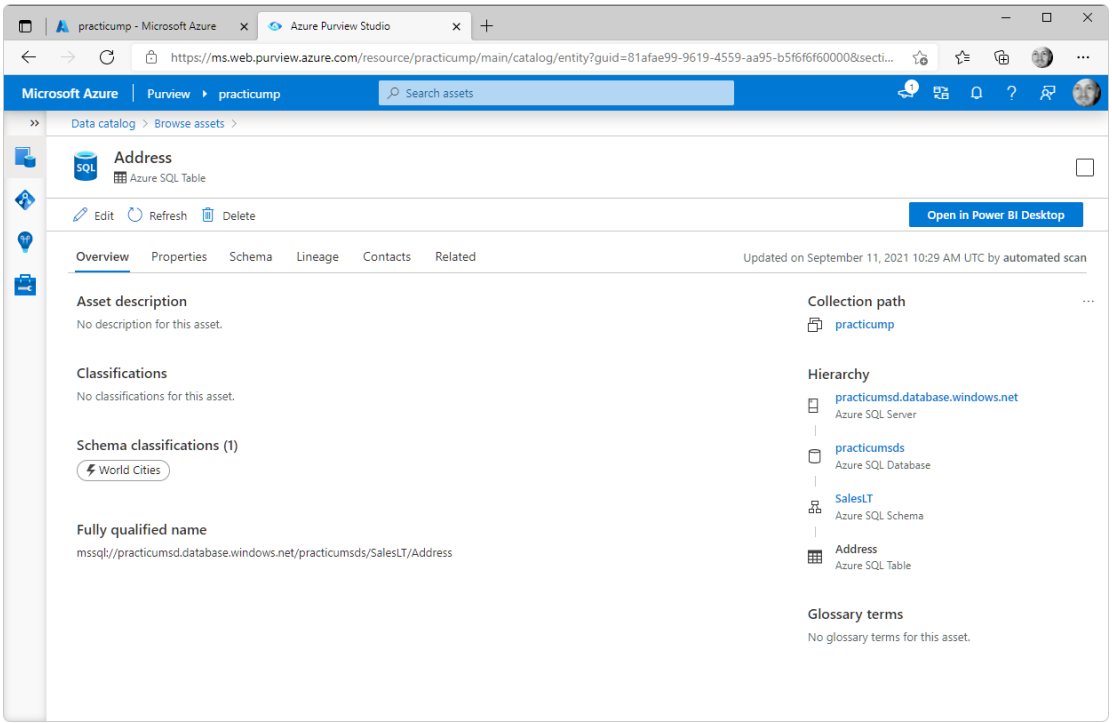


Click on the Root Collection link {i.e., name}.



Explore results.

Click on the first item, **Address**.



Notice the automatic "**World Cities**" classification.

Drill-through the tabs {e.g., Properties, Schema, Lineage, etc.}.

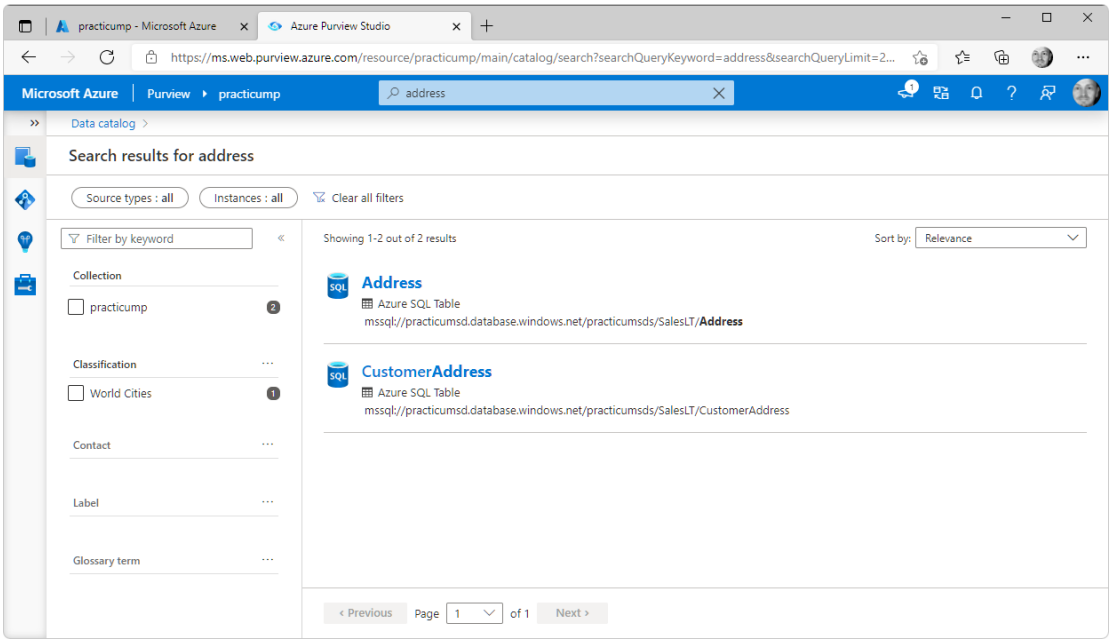
Confirm Success

Now that we have implemented the basics in Purview, can we answer the questions previously posed?

Search

Question: “Does the data that I need exist?”

Enter a phrase into the search box at the top center of the window {e.g., “address”}.

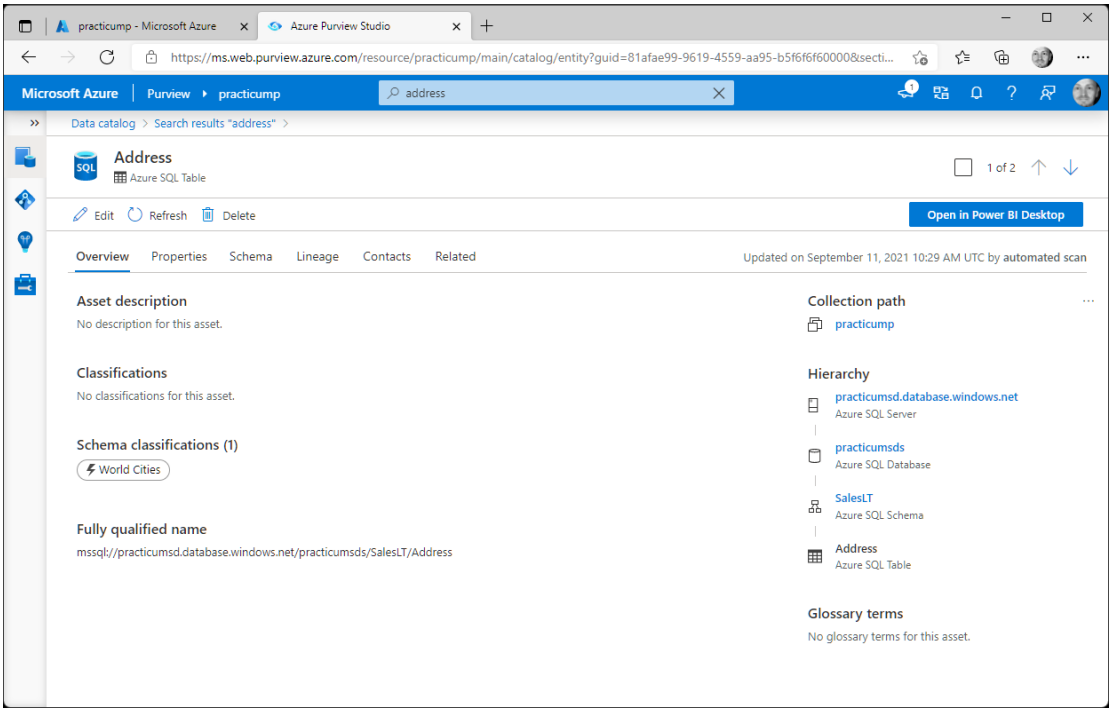


Returned search results will include all data products that include the requested phrase.

Location

Question: “how do I connect to Data Product X?”

Drill into the first search result, Address.

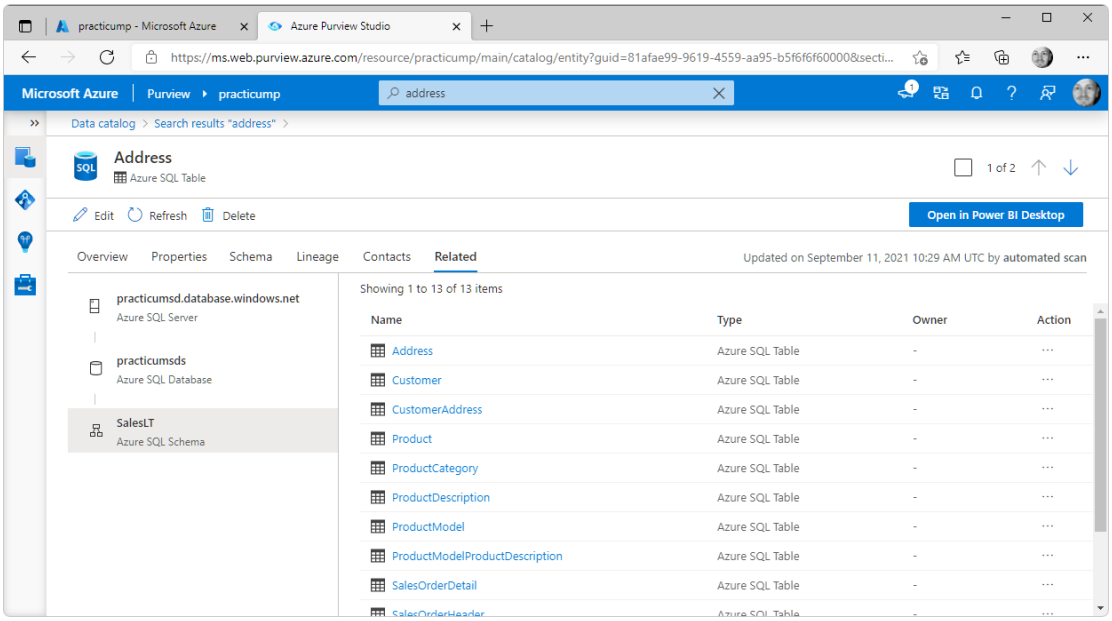


On the **Overview** tab, you will see **Hierarchy** information which includes the Fully Qualified Domain Name of the Azure SQL Server.

Related

Question: “are there related data products that I might find valuable?”

Continue in the search result for phrase “Address.”

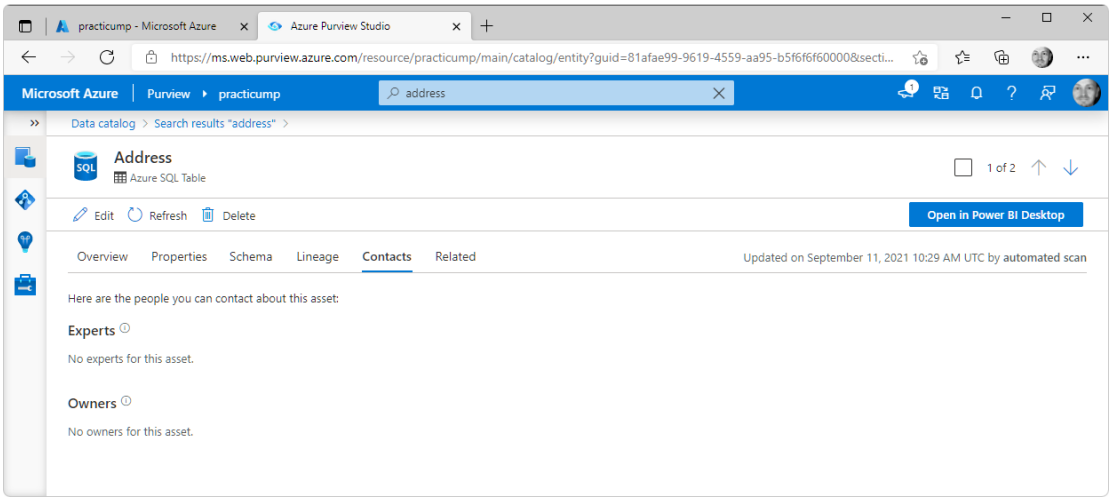


On the **Related** tab, you will see a list of related data products (as identified by Purview).

Contacts

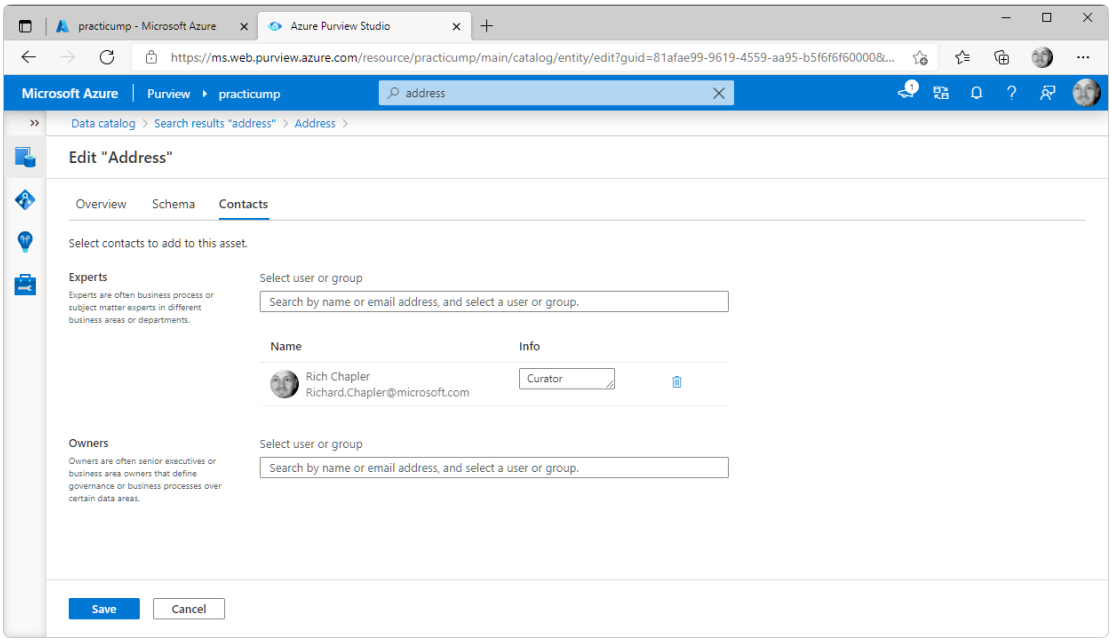
Question: “who are the owners and experts for Data Product X?”

Continue in the search result for phrase “Address.”



On the **Contacts** tab, you will see lists of **Experts** and **Owners** you can contact about the data product (as identified by Purview).

If, as in the snip above, there are no Contacts and you can add information not already captured, click the **Edit** button.



In either the **Experts** or the **Owners** drop-down menu, search for the user or group in your directory. Click **Save**.

Classify Data

Data classification is a process of annotating data products to help organizations answer the following questions:

- **Definition** … “how does data map to a business context?” and “what is the intended use of Data Product X?”
- **Documentation** … “where is the documentation of Data Product X?
- **Vocabulary** … “what else {e.g., synonyms, acronyms, etc.} might help me understand Data Product X?”
- **Standardization** … “how do I classify all data products in a standard way?”

In this section, we will configure Purview and successfully respond to these questions.

Prerequisites

This solution requires the following resources:

- Purview
- SQL

Create Term

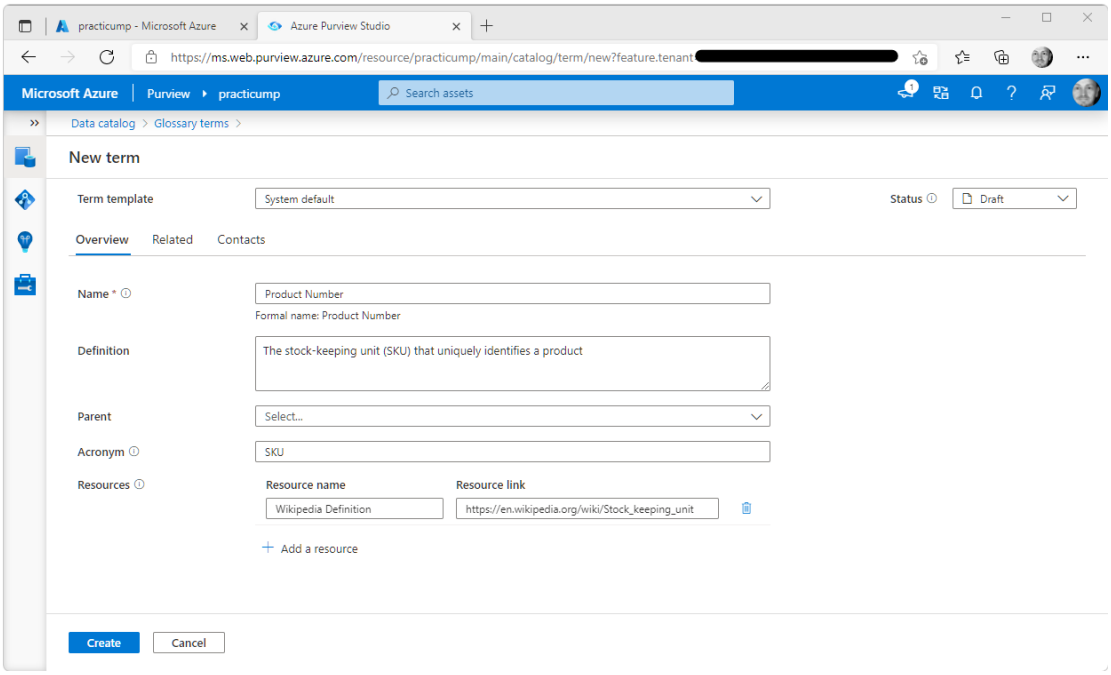
Navigate to Purview in the Azure portal.

Click the “**Open Purview Studio**” button.

Click the “**Manage Glossary**” button.

Click the “**+ New term**” button on the resulting page.

On the resulting pop-out, confirm selection of “**System default**” and then click the **Continue** button.



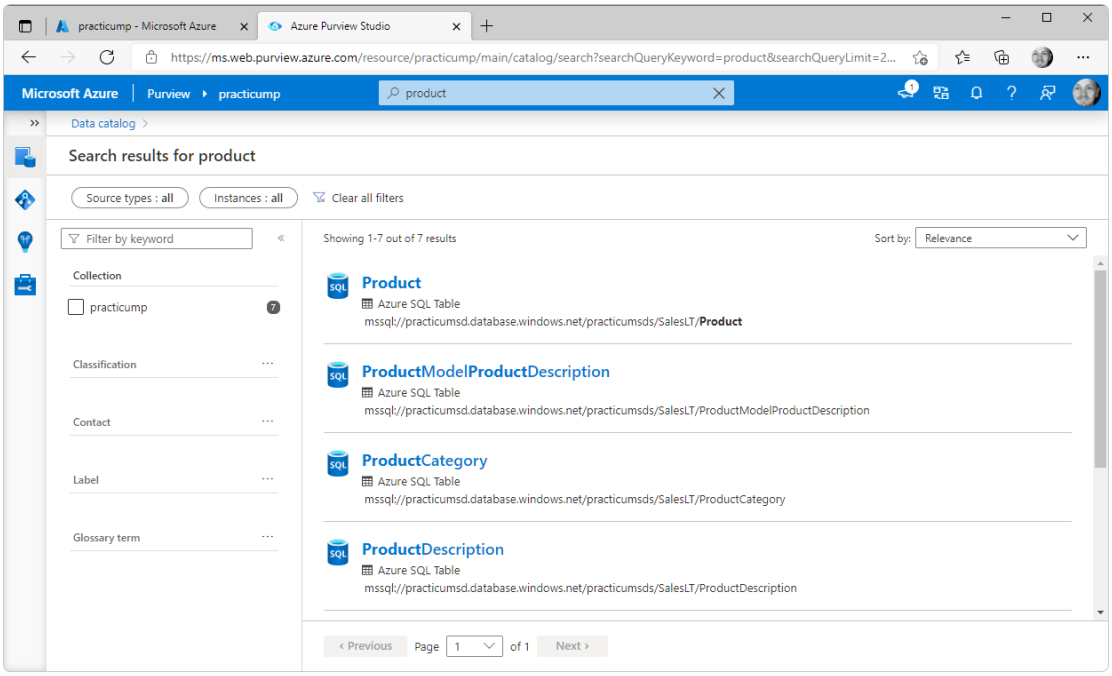
Complete the "**New term**" form, including:

Parent	Select a parent term, if applicable
Acronym	Capture abbreviation, if applicable
Resources	Click "+ Add a resource" to capture hyperlinks with descriptive content

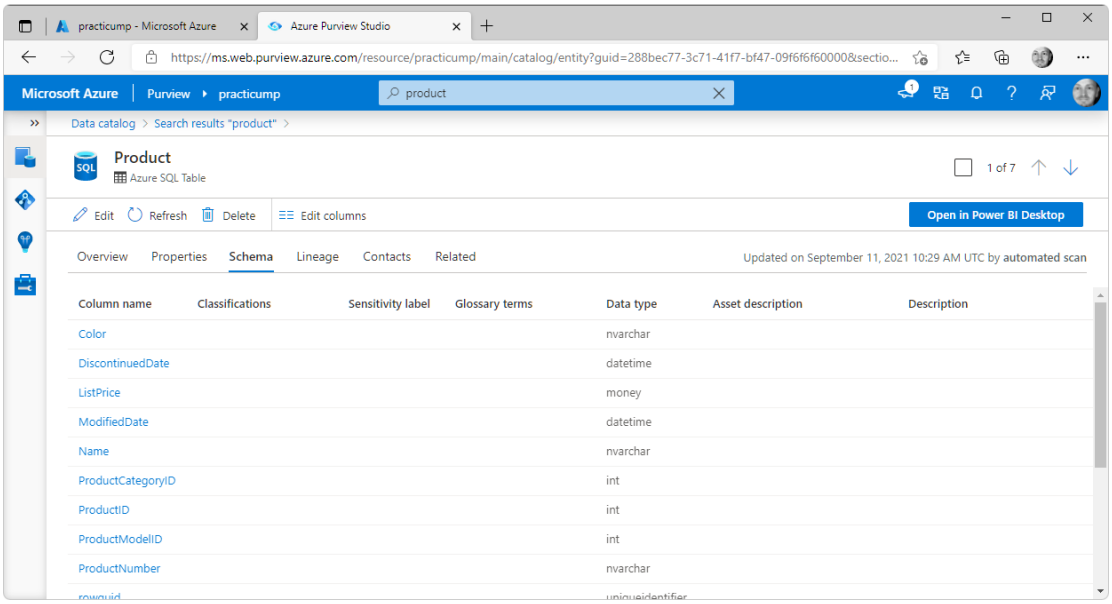
Click the **Create** button.

Use Term

Search for "product" to identify data products that use terms like Product Number.

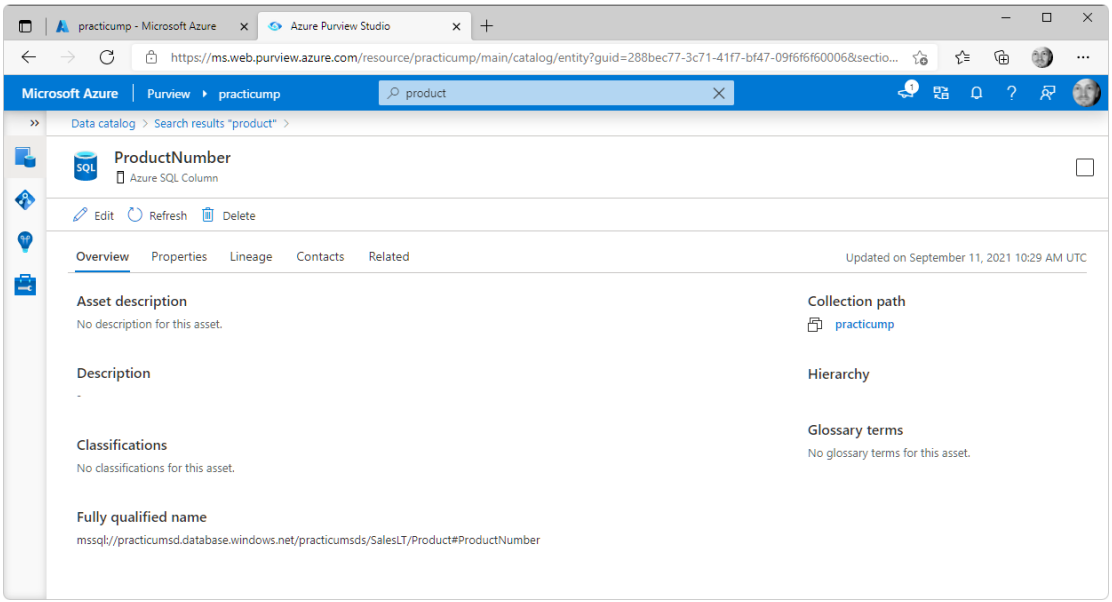


Click **Product**.

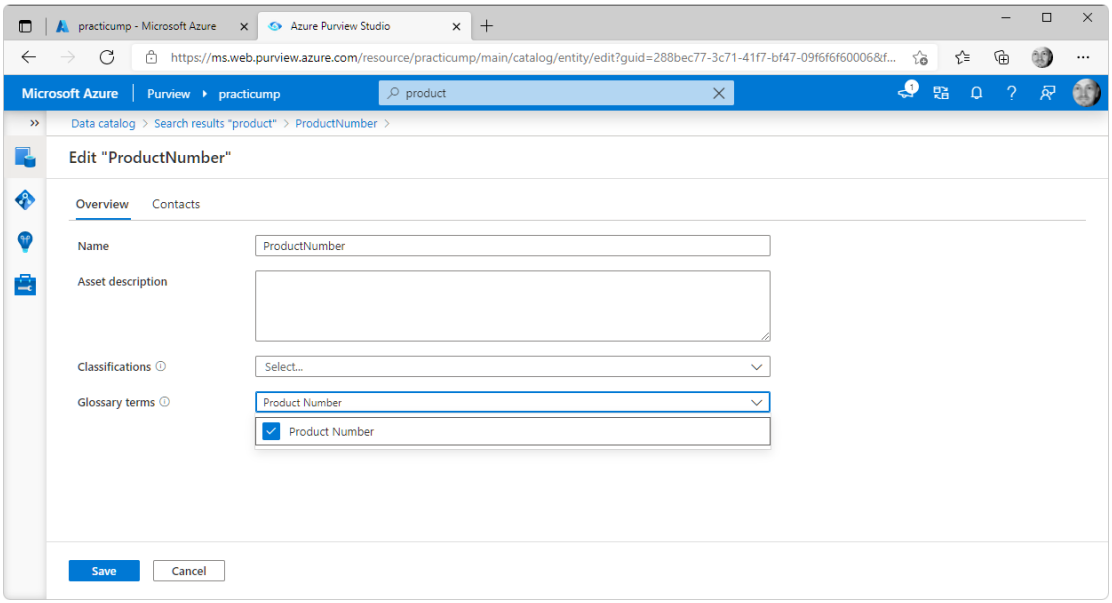


Click the **Schema** tab.

Click on the **ProductNumber** link.



Click the **Edit** button.



Select the new term, "**Product Number**" from the "**Glossary terms**" drop-down menu.

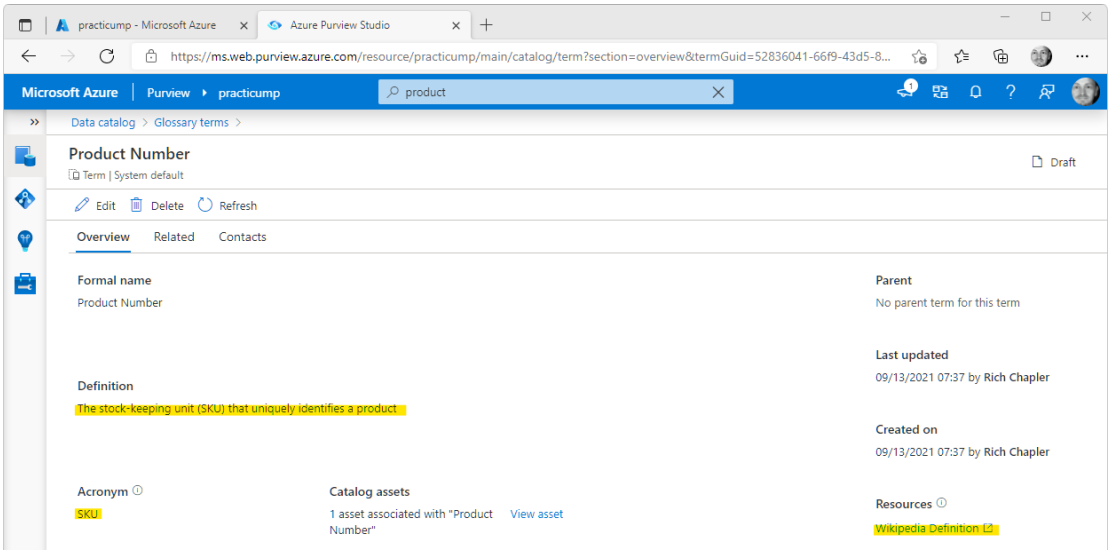
Click the **Save** button.

Confirm Success

Now that we have implemented the basics in Purview, can we answer the questions previously posed?

Definition

Questions: “how does data map to a business context?” and “what is the intended use of Data Product X?”

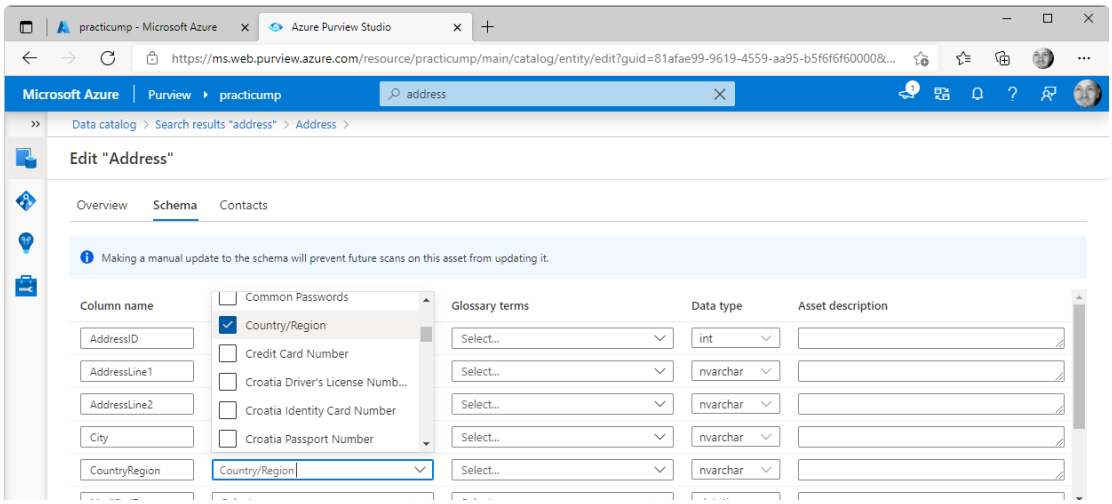


This also demonstrates response to questions:

- **Documentation** … “where is the documentation of Data Product X?
- Links to documentation can be included in **Resources**
- **Vocabulary** … “what synonyms, acronyms, etc. might help me understand Data Product X?”

Standardization

Question: “how do I classify all data products in a standard way?”



Use “**Column level classification**” options to characterize all country- / region-type schema elements as a standard classification.

Understand Lineage

Data lineage describes provenance {i.e., where data comes from, where it ends up and what happens to it along the way}.

Understanding data lineage enables stakeholders to answer the following question:

- **Accountability** (aka Impact Analysis) … "who and what would be affected by a change to or problem with Data Product X?"

In this section, we will configure Purview and successfully respond to this question.

Prerequisites

This solution requires the following resources:

- Data Factory
- Data Share
- Purview
- SQL

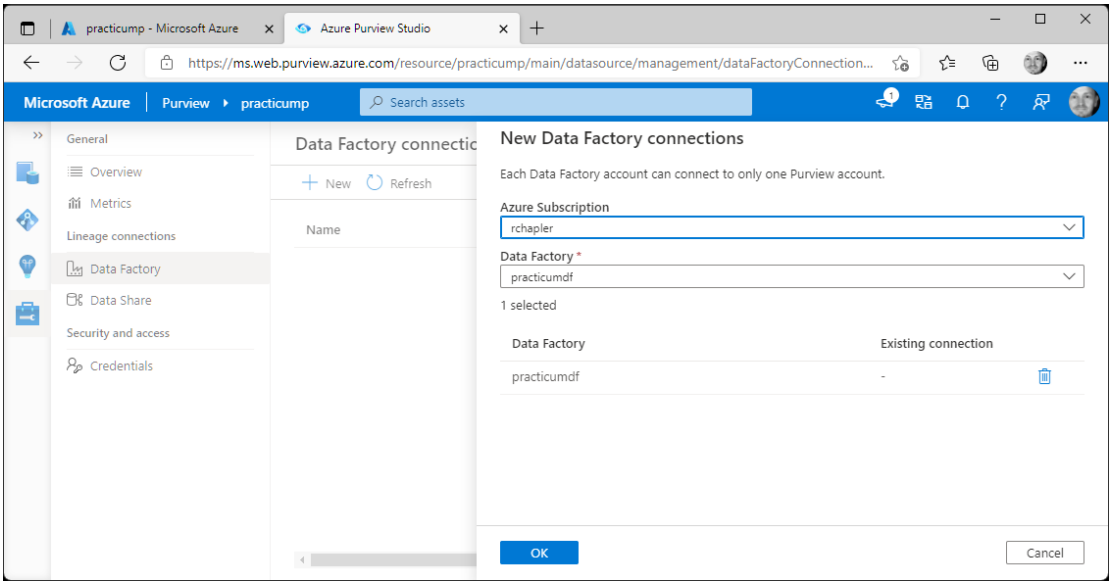
Configure Data Factory

Navigate to Purview in the Azure portal.

Click the "**Open Purview Studio**" button.

Click the **Management** icon on the navigation and then "**Data Factory**" from the resulting menu.

Click the "**+ New**" button on the resulting page.

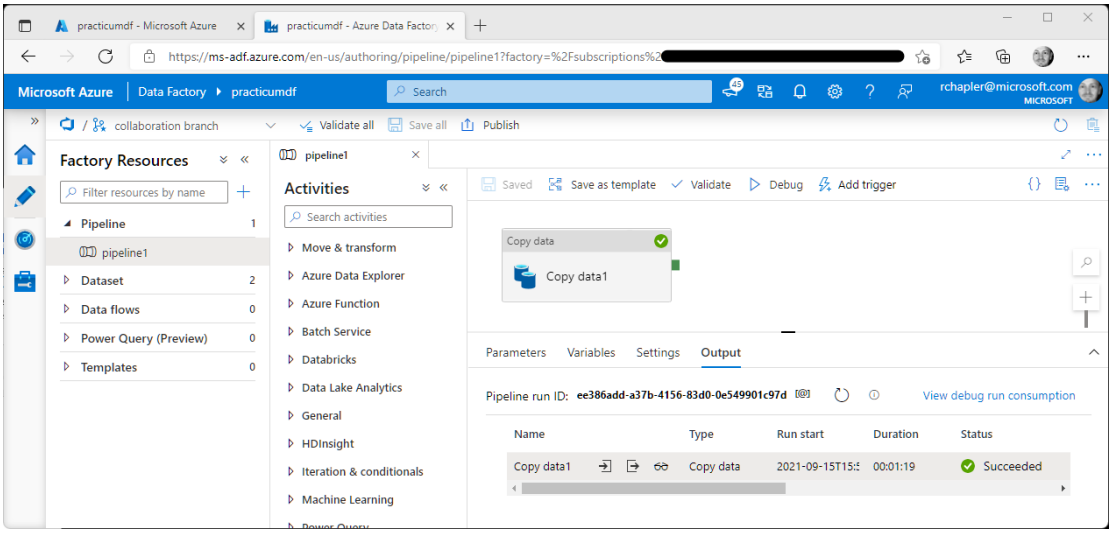


Complete the “**New Data Factory connections**” pop-out, select your Data Factory and then click the **OK** button. Confirm connection.

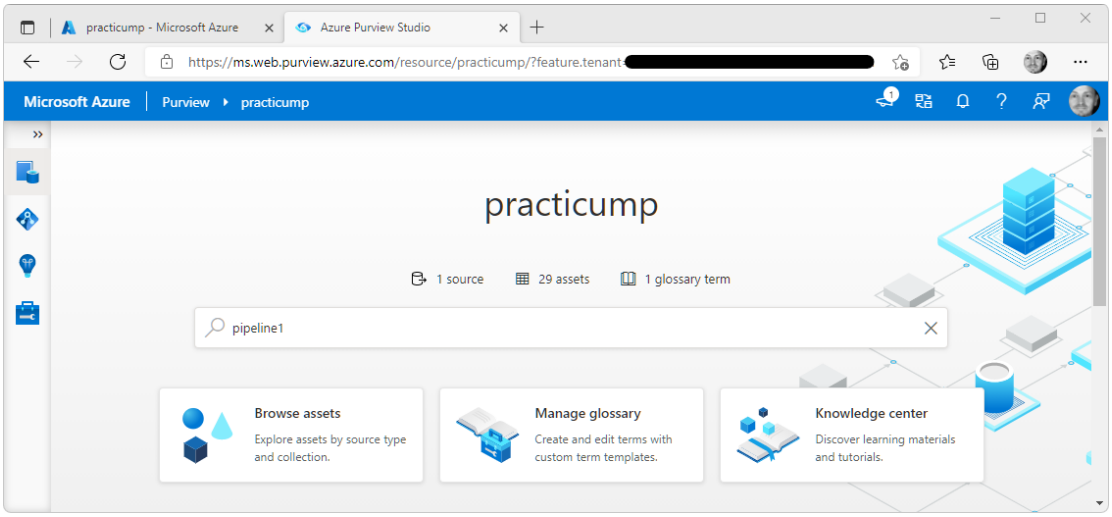
Confirm Success

Purview will automatically detect data movement and capture data about pipelines executing in a connected instance of Synapse or Data Factory.

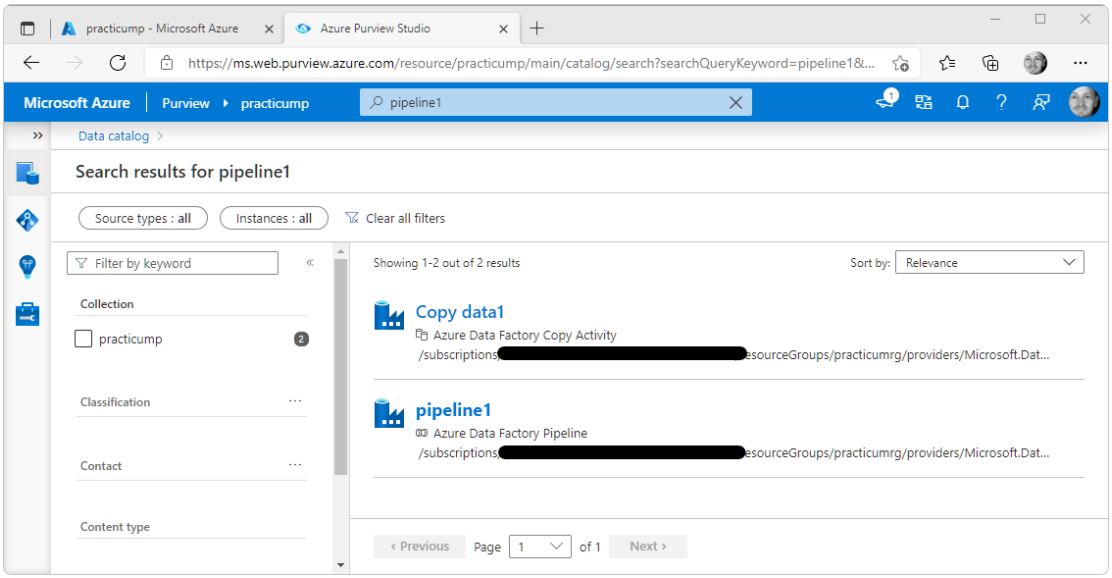
Navigate to Data Factory and execute your sample pipeline.



After successful execution, navigate to Purview.

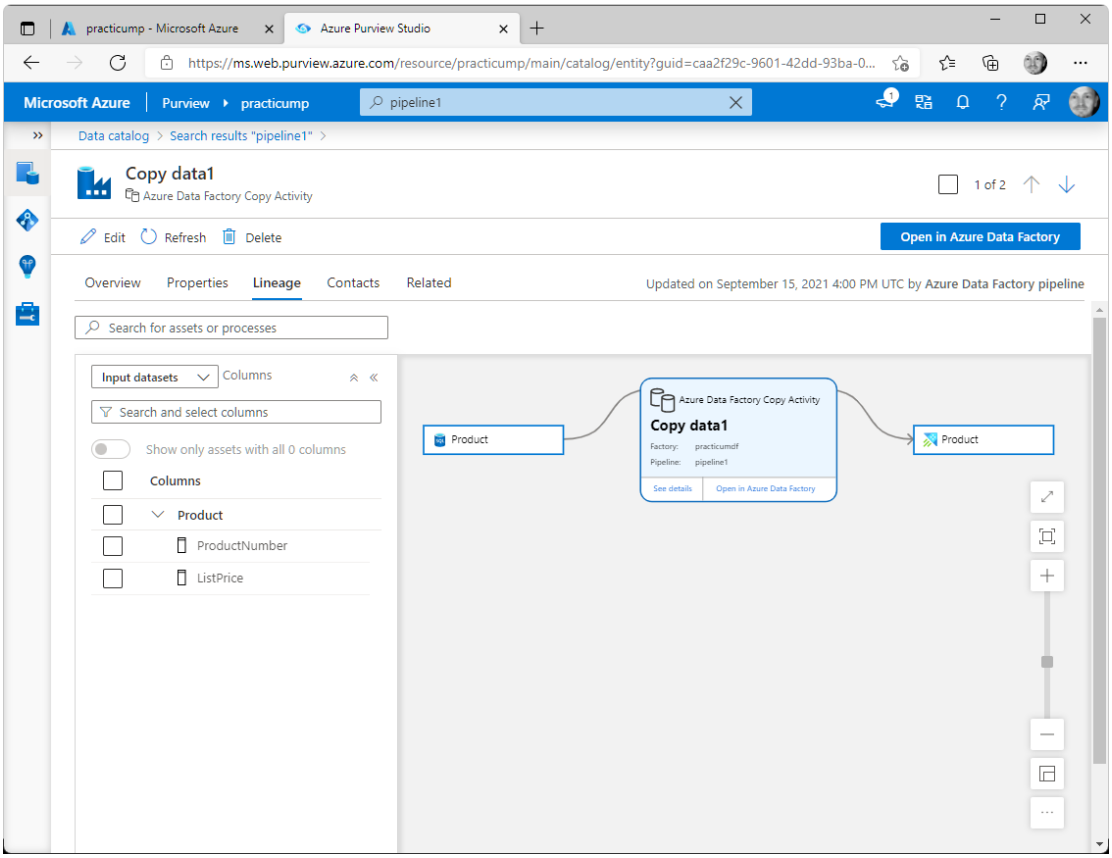


Search for “pipeline1.”



Click on the "**Copy data1**" result.

Click on the **Lineage** tab.



Continue to drill-through elements to better understand the capture of Lineage data.

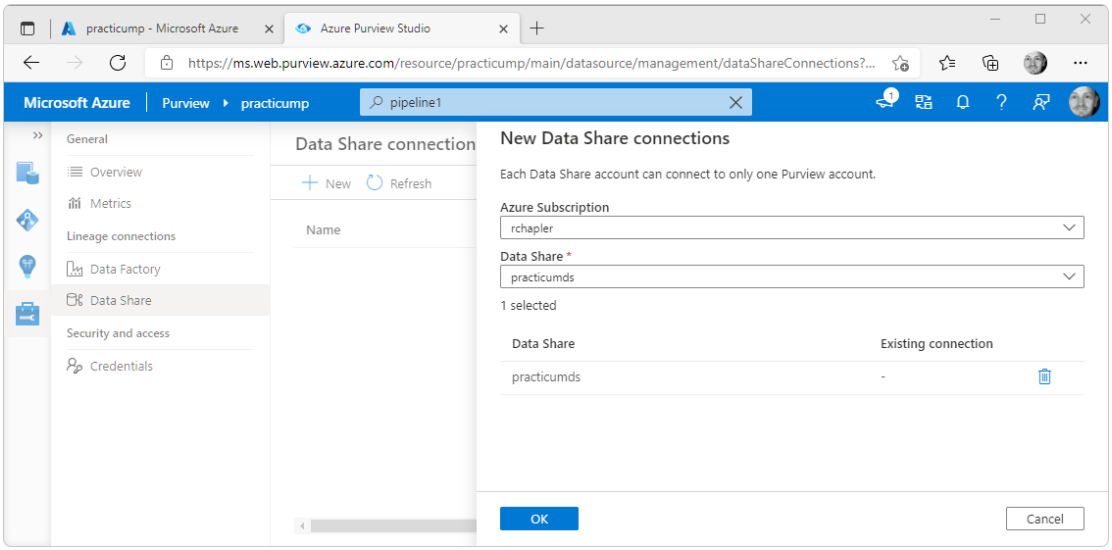
Configure Data Share

Navigate to Purview in the Azure portal.

Click the "**Open Purview Studio**" button.

Click the **Management** icon on the navigation and then "**Data Share**" from the resulting menu.

Click the "**+ New**" button on the resulting page.



Complete the "**New Data Factory connections**" pop-out, select your Data Share and then click **OK**.

Article: https://docs.microsoft.com/en-us/azure/purview/how-to-link-azure-data-share

Note from this article… "For Data Share products to show in Purview, a snapshot job must be run after you connect your Data Share to Purview"