Foreword

This book is essentially a collection of objectives with step-by-step instructions not appropriately launched with a flowery introduction. That said, I feel like it needs *something*. Do not worry, I intend to be brief…

The title for my role, Cloud Solution Architect, falls short of self-explanation. So, when asked, I explain that I exist in the grey area between various roles:

- **I sell, but I am not a sales person**… I am certainly biased in favor of 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 hands-on solution development, but there is a role-defined limit to how deep I can dig (and after that is where customer resources take over)
- **I support, but I am not a support engineer**… I will dig into just about any issue to help with a challenge, but Support Engineers have the knowledge and resources to go far deeper than I
- **I train, but I am not a training professional**… I work with customers, whenever possible, directly in their subscriptions and environments; as such, I spend more time coaching than coding

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

Many times, customer ask questions or need solutions where I must learn more. This book is an opportunistic capture of the knowledge that I have gained {i.e., a collection of step-by-step instructions describing solution implementation}.

Content is organized into **product- | activity- focused objectives** and **discipline-specific sections**; consider browsing directly to those objectives that align with your focus and interest.

Table of Contents

Prepare

Naming Convention

Consider best practice guidance from the Azure Cloud Adoption Framework:

[Define your naming convention - Cloud Adoption Framework | Microsoft Docs]

When setting on a naming convention, also remember that:

- Some resources will not allow names with a dash ("-") or other special characters
- Some resources will allow names with capitals and some only lowercase

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

I do not adhere to the guidance in the Cloud Adoption Framework; rather, I typically use a combination of project name as prefix and resource type acronym as suffix. For example:

- Suffix: "ads" (as in Azure Data Solutions)
- Prefix: "dl" (as in Data Lake)
- Resource Name: "adsdl"

Because this book has been written over time and for various projects, you might see different naming used in different exercises.

Infrastructure

The following sections describe various ways {e.g., Azure Portal, Bash, PowerShell, etc.} to manually instantiate the Azure resources necessary to complete the objectives in this book.

Application Registration (aka Service Principal)

This exercise requires the following resource(s):

- Key Vault

Consider using a User-Assigned Managed Identity instead of a Service Principal.

Create using Azure Portal

Navigate to "**Azure Active Directory**", select "**App Registrations**" in the **Manage** group of the navigation pane, and then click the "**+ New Registration**" button.

On the "**Register an application**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards

No additional changes are required. Review settings on the remaining tabs.

Click the **Register** button.

Key Vault Secrets, Tenant, and Client Identifiers

Open a new tab and navigate to Key Vault.

Click **Secrets** in the **Settings** group of the navigation pane.

Click the "**+ Generate/Import**" button.

On the "**Create a secret**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Value	Paste the previously copied value

No additional changes are required. Review settings on the remaining tabs.

Click the **Create** button.

Repeat for **Tenant Id**.

Client Secret

Navigate to “**Certificates & secrets**” in the **Manage** group in the navigation pane.

Click the “**+ New client secret**” button, and then in the resulting “**Add a client secret**” popout, click the **Add** button.

Copy **Value**.

Key Vault Secret, Client 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.

On the “**Create a secret**” page, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Value	Paste the previously copied **Value**

No additional changes are required. Review settings on the remaining tabs.

Click the **Create** button.

Create using Cloud Shell (PowerShell)

Navigate to Cloud Shell and select **PowerShell** from the "**Select Environment**" dropdown.

Update and execute the following command create a Service Principal and give it Contributor access to the subscription:

```
az ad sp create-for-rbac --name "adssp" --role contributor --scopes /subscriptions/{Subscription GUID}
```

Logic explained…

- `az ad sp create-for-rbac`
  …creates a service principal and configures access (az ad sp | Microsoft Docs)

- `--name`
  …self-explanatory

- `--role contributor`
  …role of the service principal

- `--scopes /subscriptions/{Subscription GUID}`
  …replace with your Subscription GUID
  …scope the service principal's role assignment applies to

You can expect a response like the following redacted example:

```
{
 "appId": "{App GUID, aka Client Id}",
 "displayName": "adssp",
 "password": "{Password String, aka Client Secret "rbac"}",
 "tenant": "{Tenant GUID}"
}
```

These values will be useful in many exercises; keep them handy.

Data Lake

Create using Azure Portal

This exercise requires the following resource(s):

- Resource Group
- Storage Account

On the "**Create Storage Account**" page, enter values for the following items:

Subscription	Select your subscription
Resource Group	Select your resource group
Storage Account Name	Enter a meaningful name aligned with standards
Location	Match resource group selection
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)"

Navigate to the **Advanced** tab.

Click on the **Enabled** radio button in the "**Data Lake Storage Gen2**" grouping.

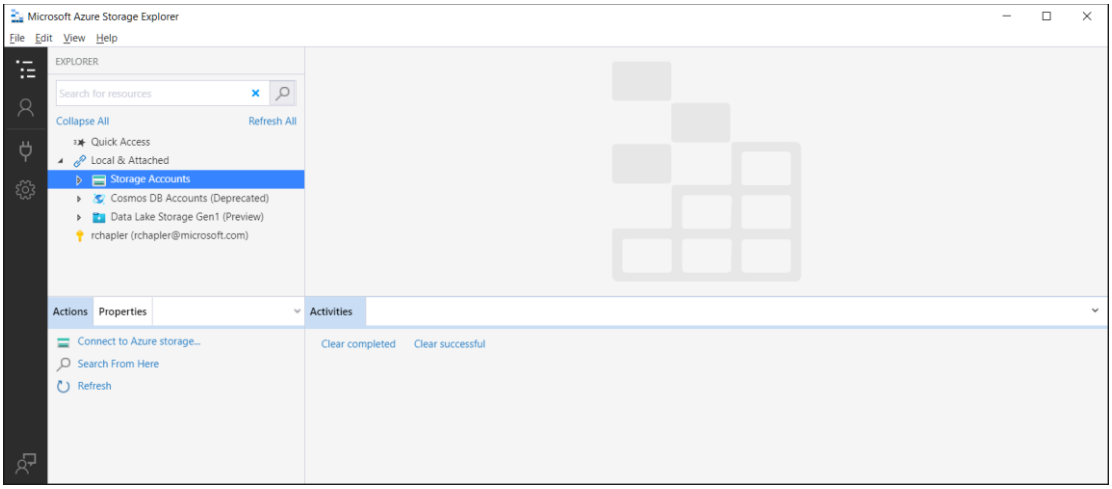
No additional changes are required. Review settings on the remaining tabs.

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

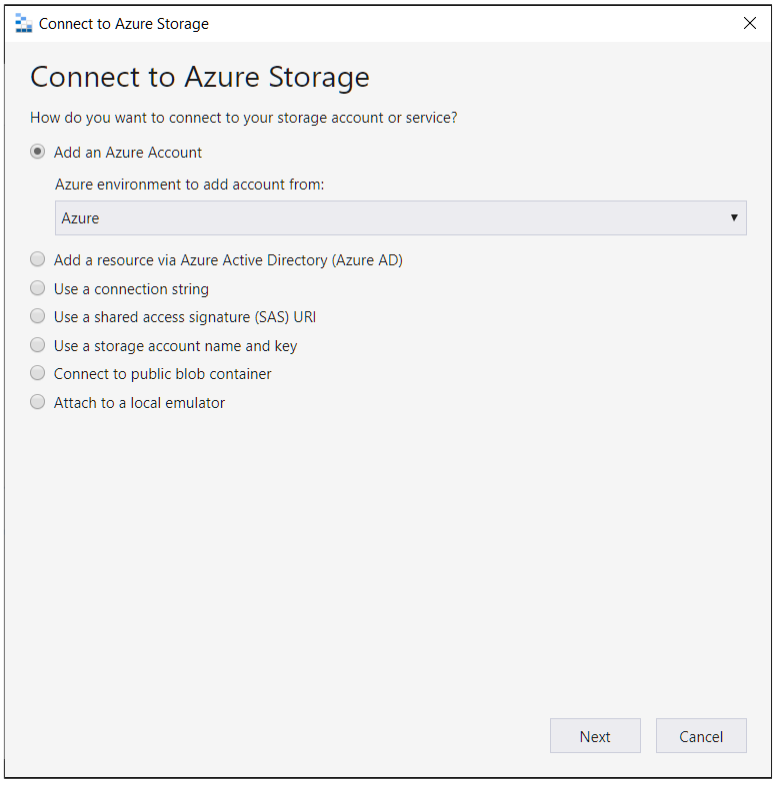
Storage Explorer

Download and install the Microsoft Azure Storage Explorer app (Azure Storage Explorer – cloud storage management | Microsoft Azure).

Launch the app and provide Azure credentials.



Click the “Connect to Azure storage…” link.

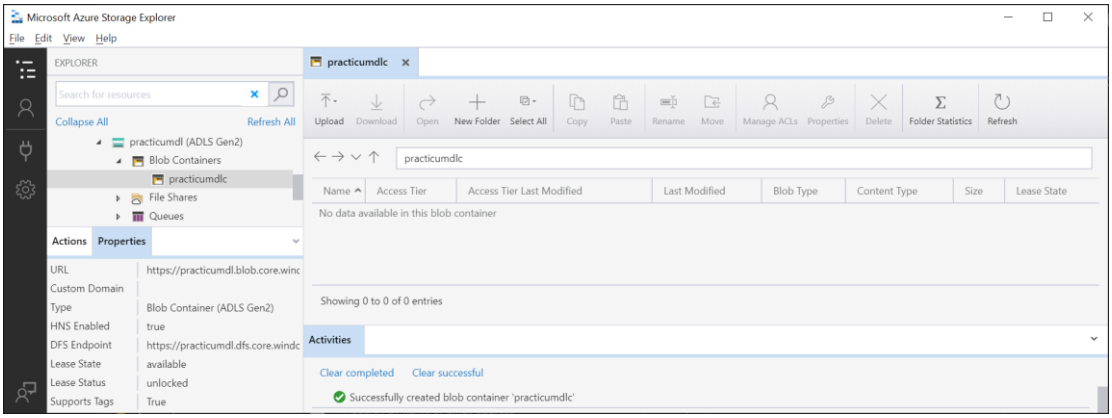


On the resulting popup, click the “**Add an Azure Account**” radio button. Click the **Next** button. Authenticate with Azure credentials.

Container

Continue with the “Microsoft Azure Storage Explorer” app.

Use the search bar to focus on your Storage Account.



Right-click on “**Blob Containers**” and click “**Create Blob Container**” in the resulting popup menu. Enter a meaningful name.

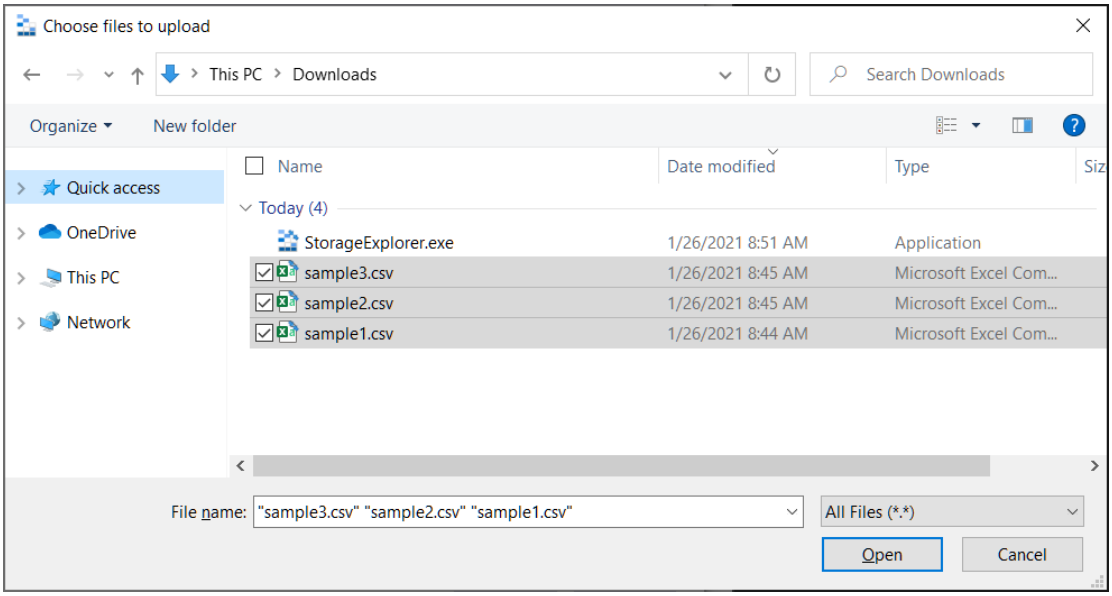
Sample Data

Download sample files from a site like [Sample CSV Files Download - Get Examples Instantly (filesamples.com)]

Return to the “Microsoft Azure **Storage Explorer**” app and navigate to your container. Click the **Upload** button and select “Upload Files” in the resulting popup menu.

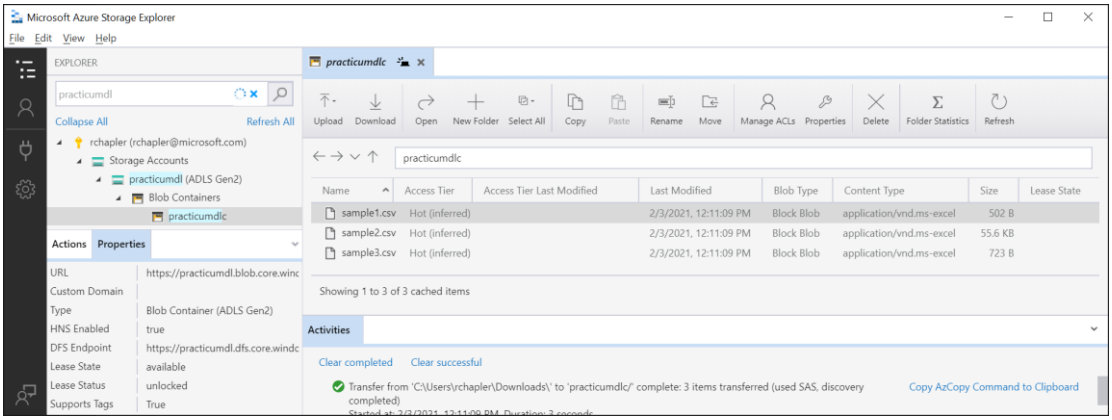
Click the ellipses button to the right of the “**Selected Files**” box.

Navigate to **Downloads** and select your sample data files.



Click the **Open** button.

On the "**Upload Files**" popup, click the **Upload** button. Allow time for file transfer.



Data Migration Assistant

Browse to [Download Microsoft® Data Migration Assistant v5.5 from Official Microsoft Download Center](#).

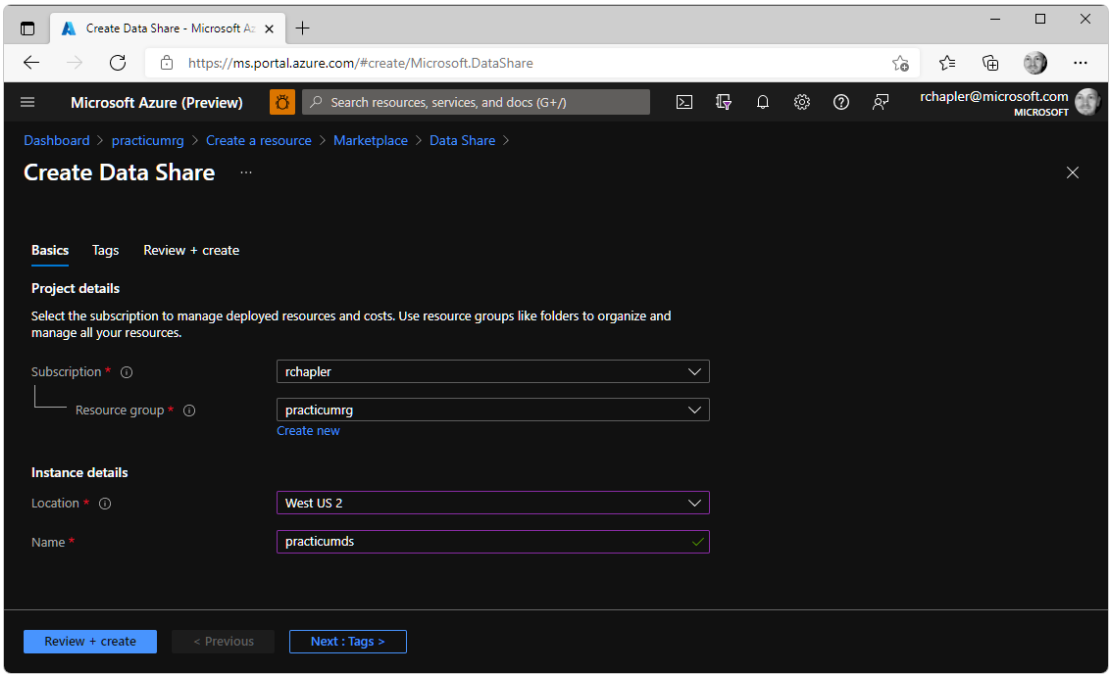
Download, open, and install the "**User Installer**" for your platform.

Data Share

Create using Azure Portal

This exercise requires the following resource(s):

- Data Explorer (2 instances)
- Resource Group



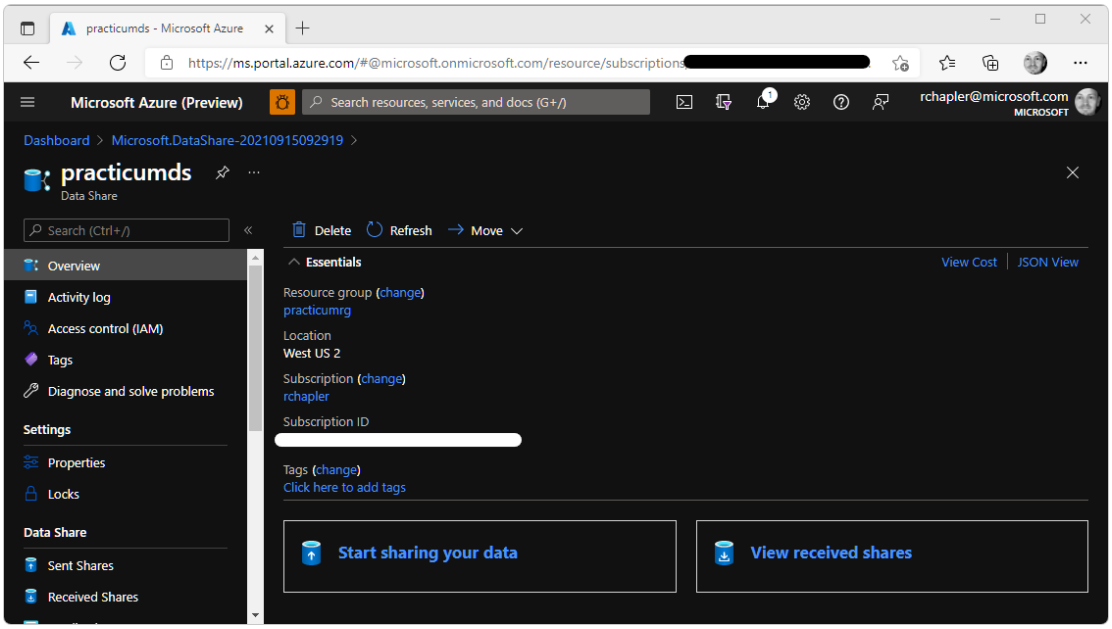
On the "**Create Data Share**" page, enter values for the following items:

Subscription	Select your subscription
Resource Group	Select your resource group
Location	Match resource group selection
Name	Enter a meaningful name aligned with standards

No additional changes are required. Review settings on the remaining tabs.

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

After deployment is complete, click the “**Go to resource**” button.

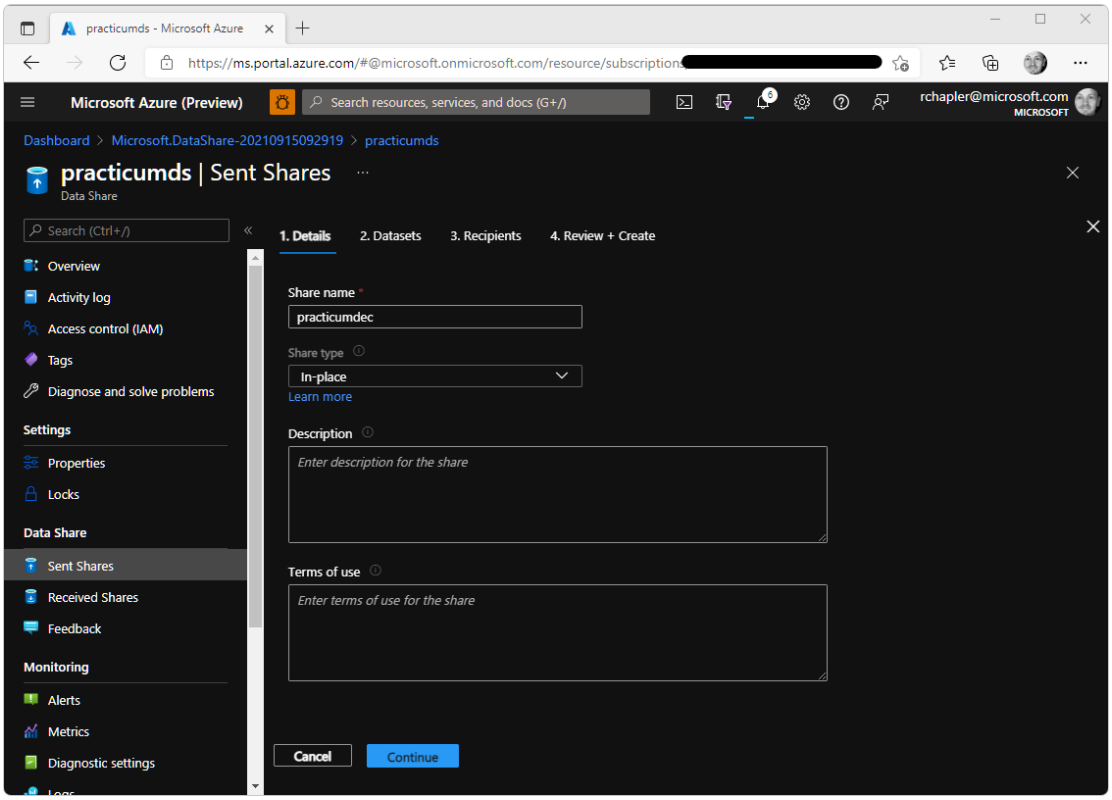


Sample Share

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

Click the “Start sharing your data” button.

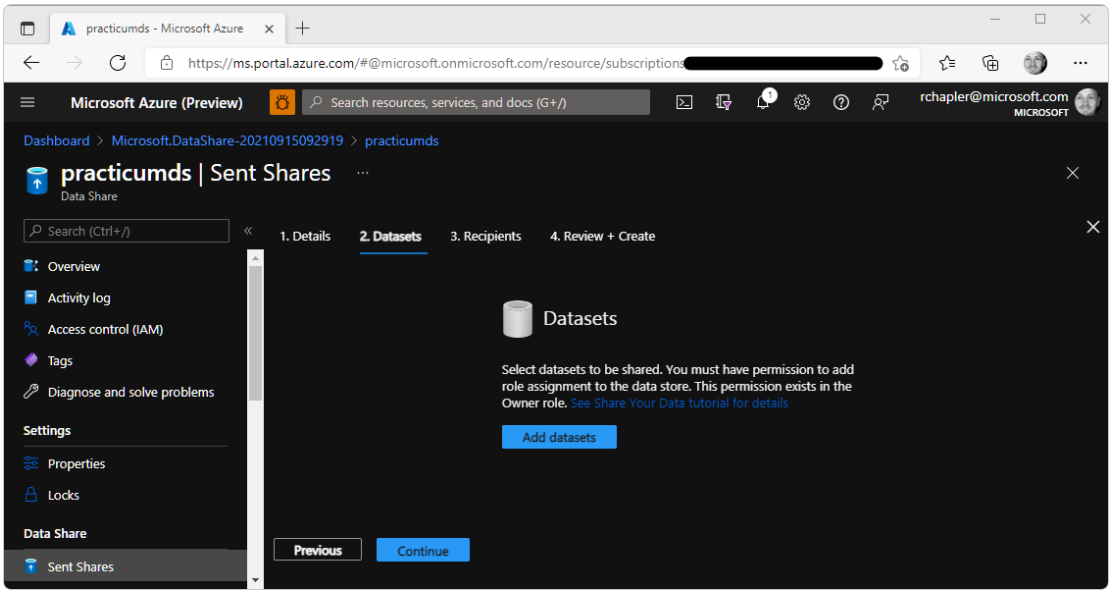
On the resulting “**Sent Shares**” page, click the “**+ Create**” button.



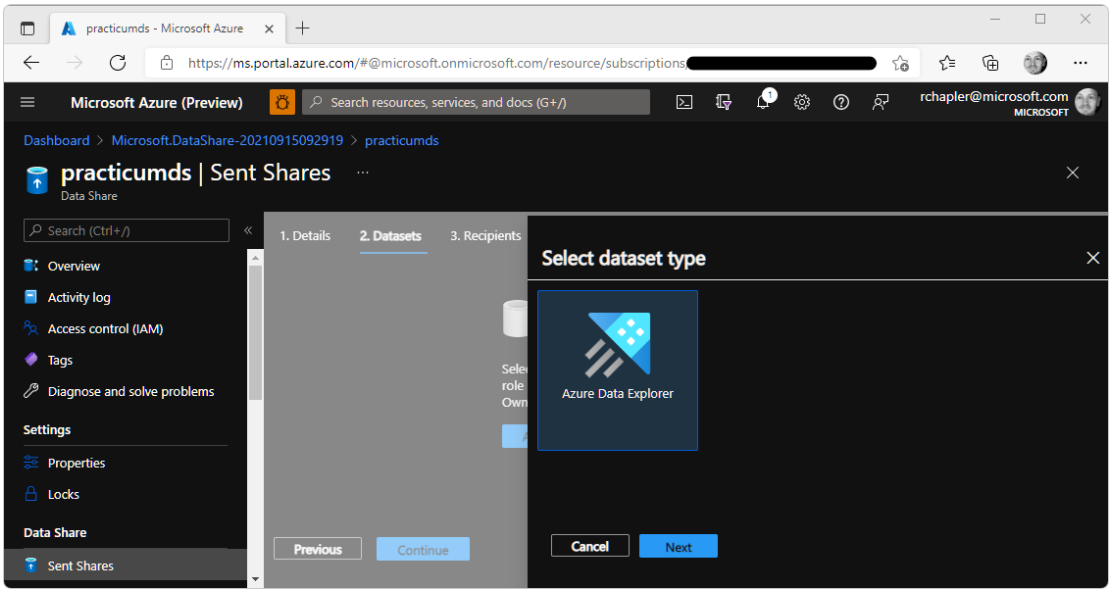
On the "**Sent Shares**" page, "**1. Details**" tab, enter values for the following items:

Share Name	Enter a meaningful name aligned with standards
Share Type	Select "In-place" from the dropdown menu

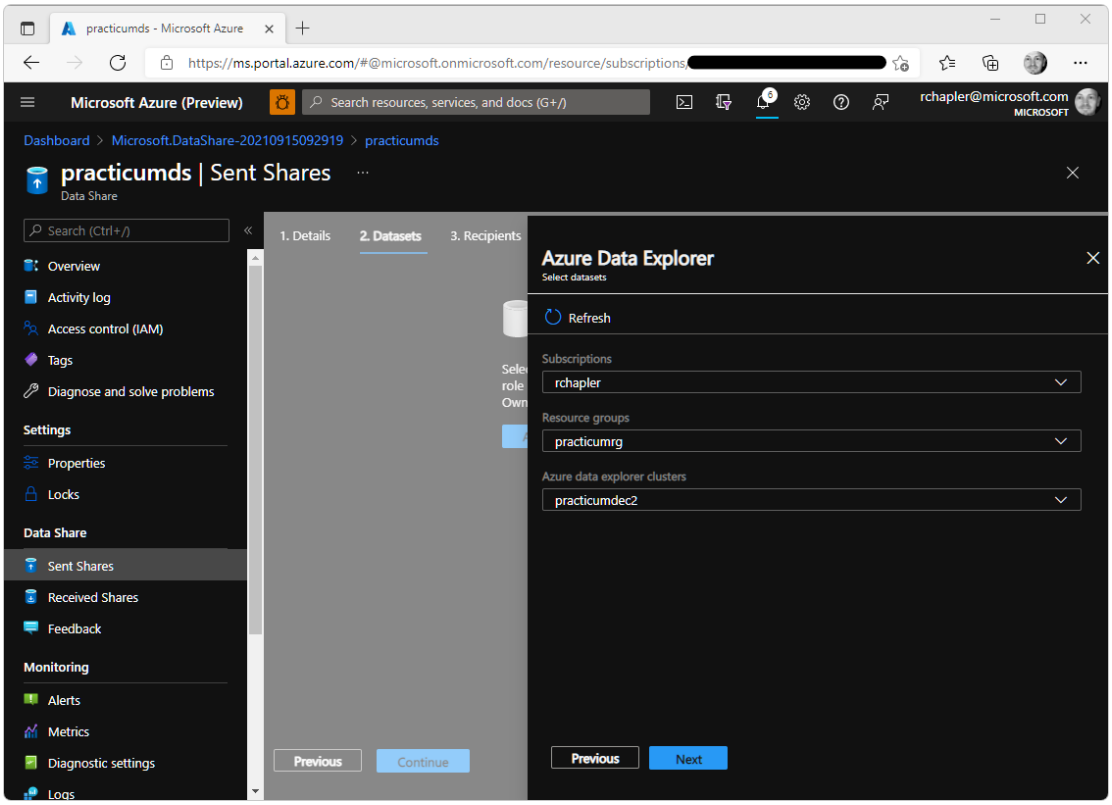
Click the **Continue** button.



On the "**Sent Shares**" page, "**2. Datasets**" tab, click the "**Add datasets**" button.



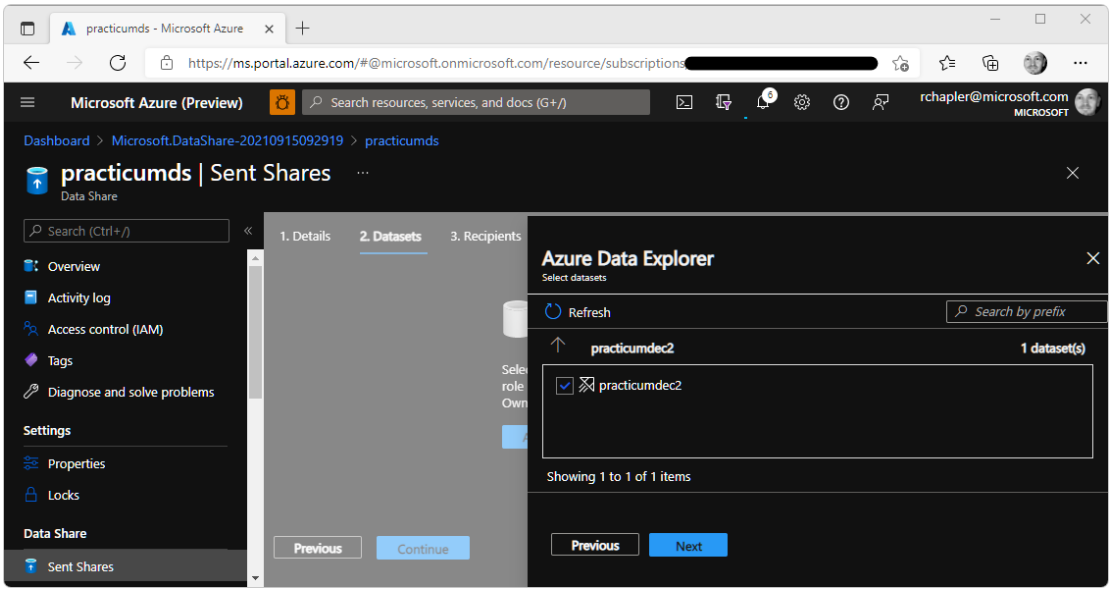
On the resulting "**Select dataset type**" popout, click the "**Azure Data Explorer**" button, then click the **Next** button.



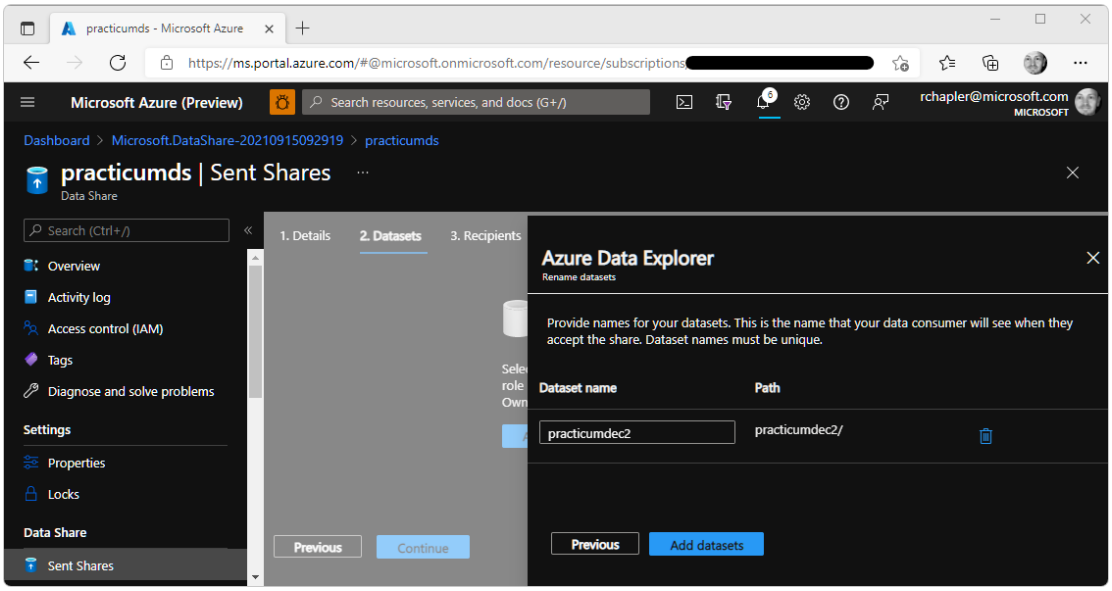
On the resulting "**Azure Data Explorer**" popout, enter values for the following items:

Subscriptions	Select your subscription
Resource Groups	Select your resource group
Azure Data Explorer Clusters	Select your Data Explorer cluster
Name	Enter a meaningful name aligned with standards

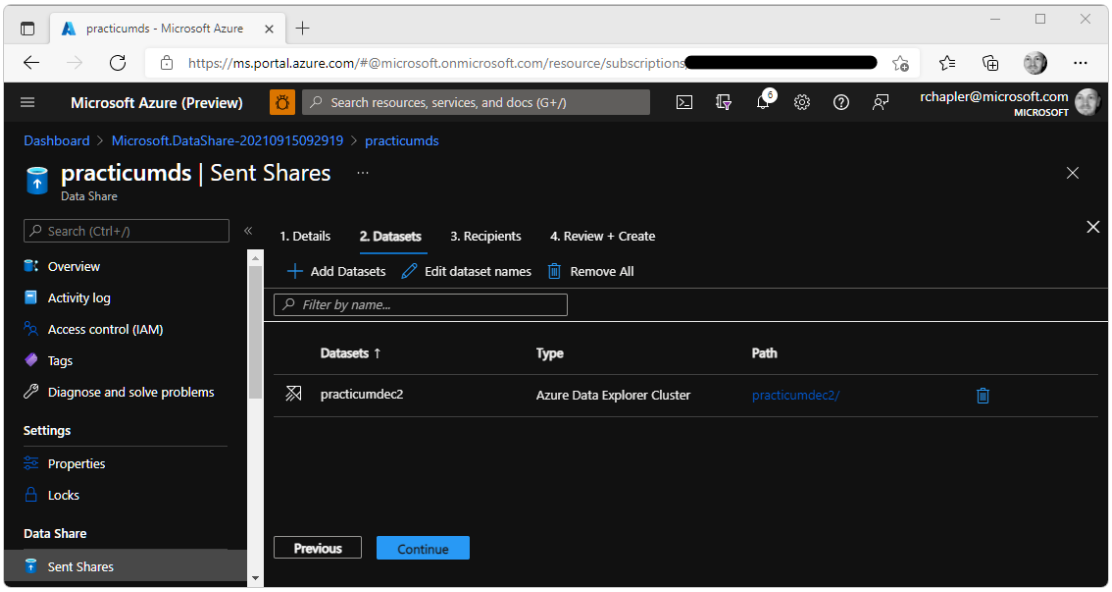
Click the **Next** button.



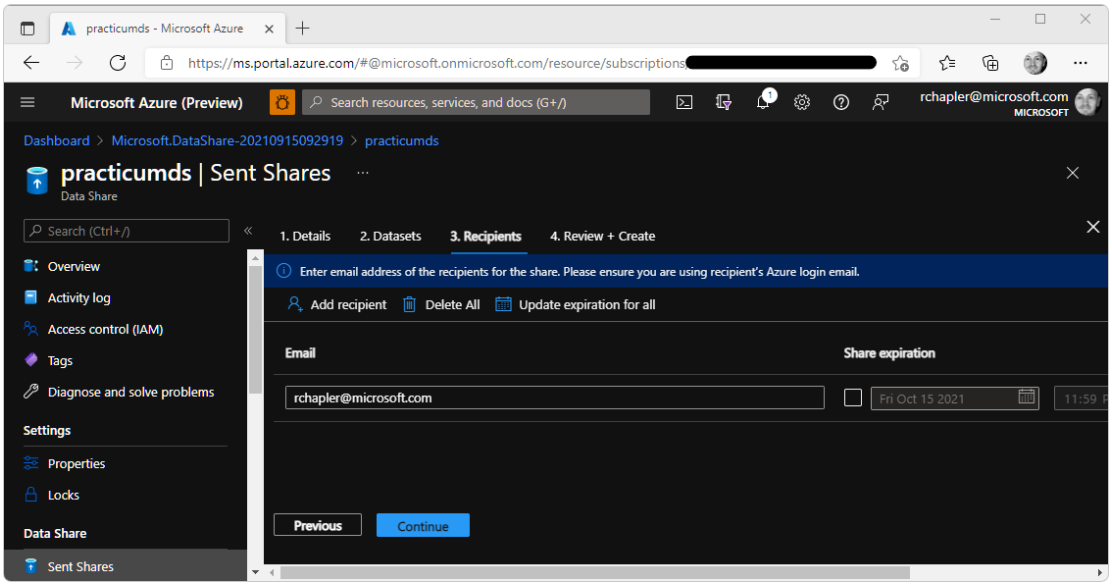
On the second "**…Select datasets**" popout, check the box next to your cluster and then click the **Next** button.



On the "**…Rename datasets**" popout, confirm the automatically generated "Dataset name" value. Click the "**Add datasets**" button.



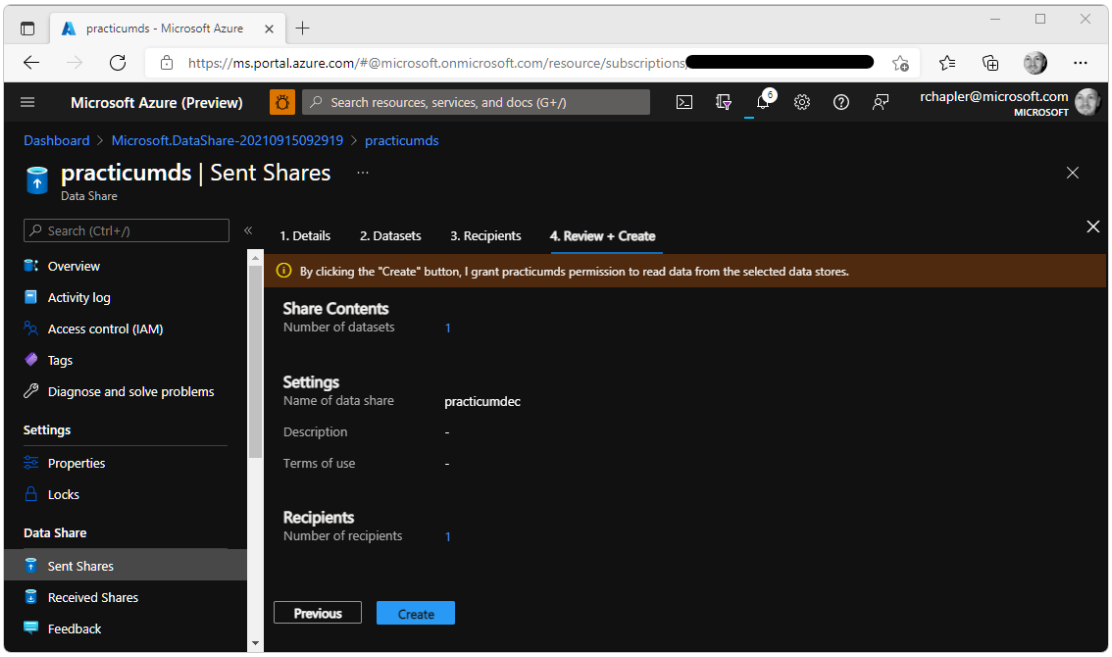
Back on the "**Sent Shares**" page, "**2. Datasets**" tab, click the **Continue** button.



On the "**Sent Shares**" page, "**3. Recipients**" tab, click the "**Add recipient**" button.

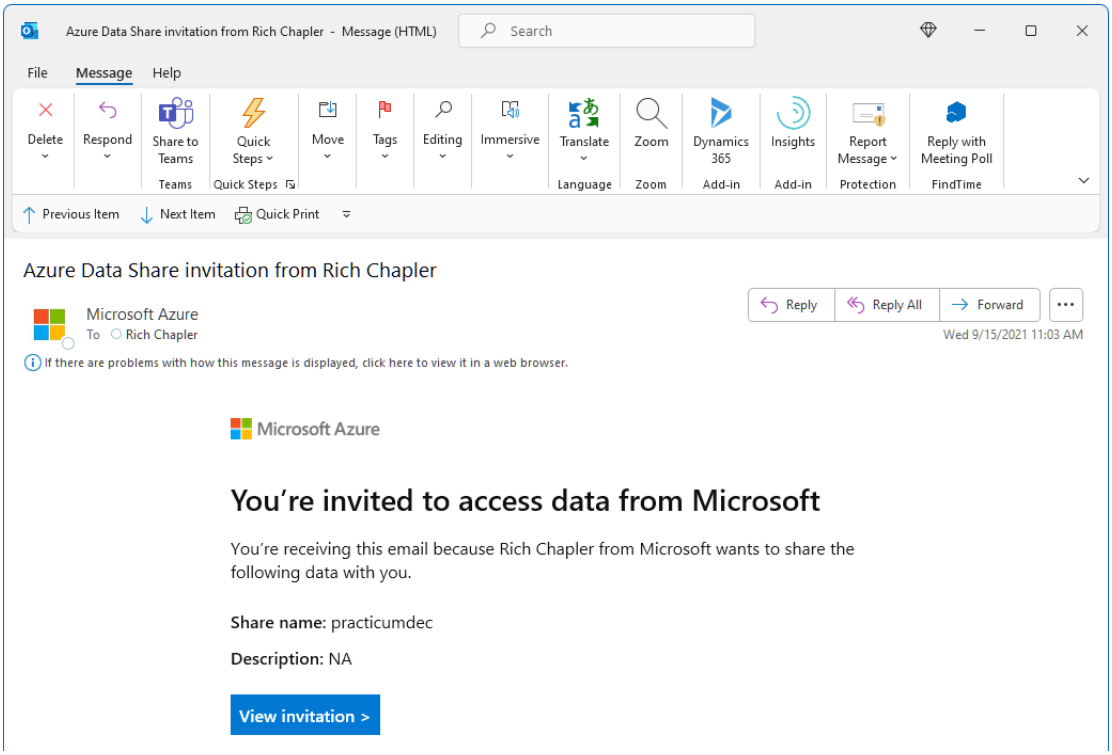
In the resulting interface, enter an **Email** value.

Click the **Continue** button.

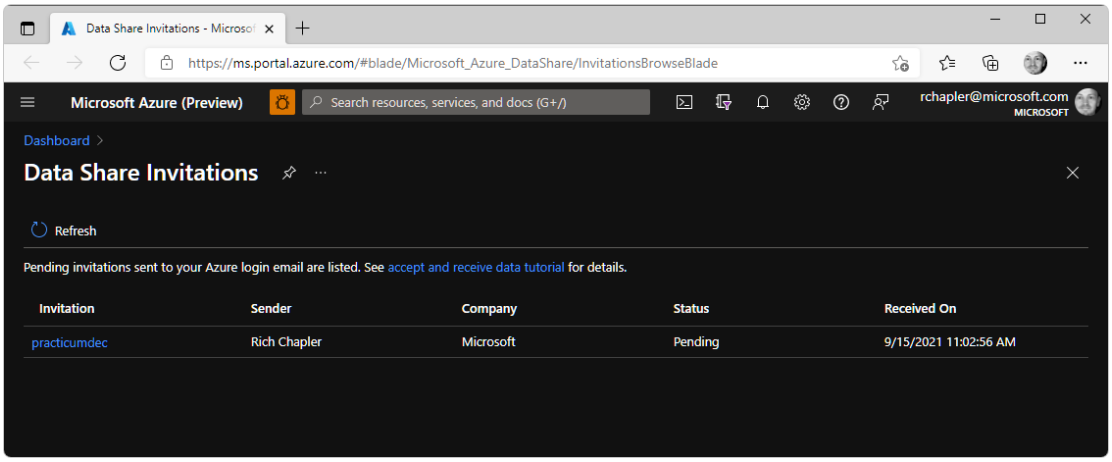


On the "**Sent Shares**" page, "**4. Review + Create**" tab, click the **Create** button.

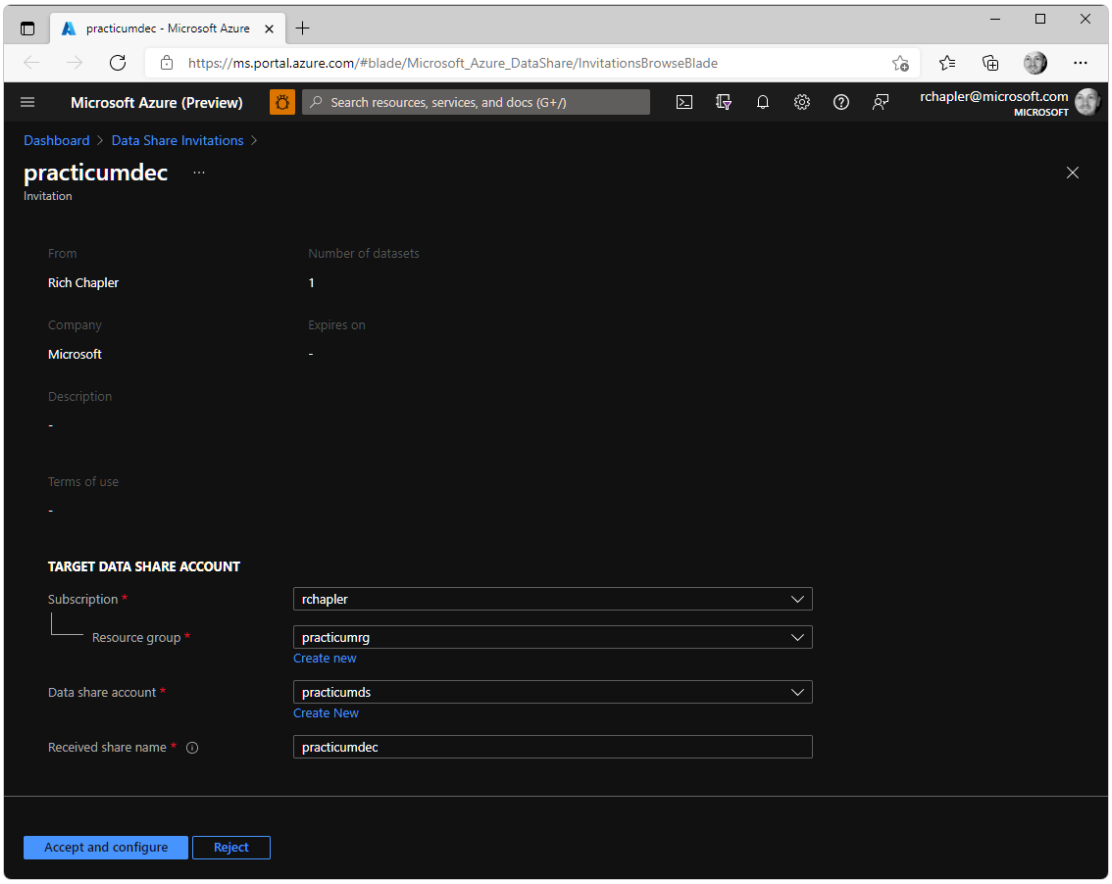
The designated email recipient can expect to receive an email like the one below:



Click the “**View invitation >**” link.



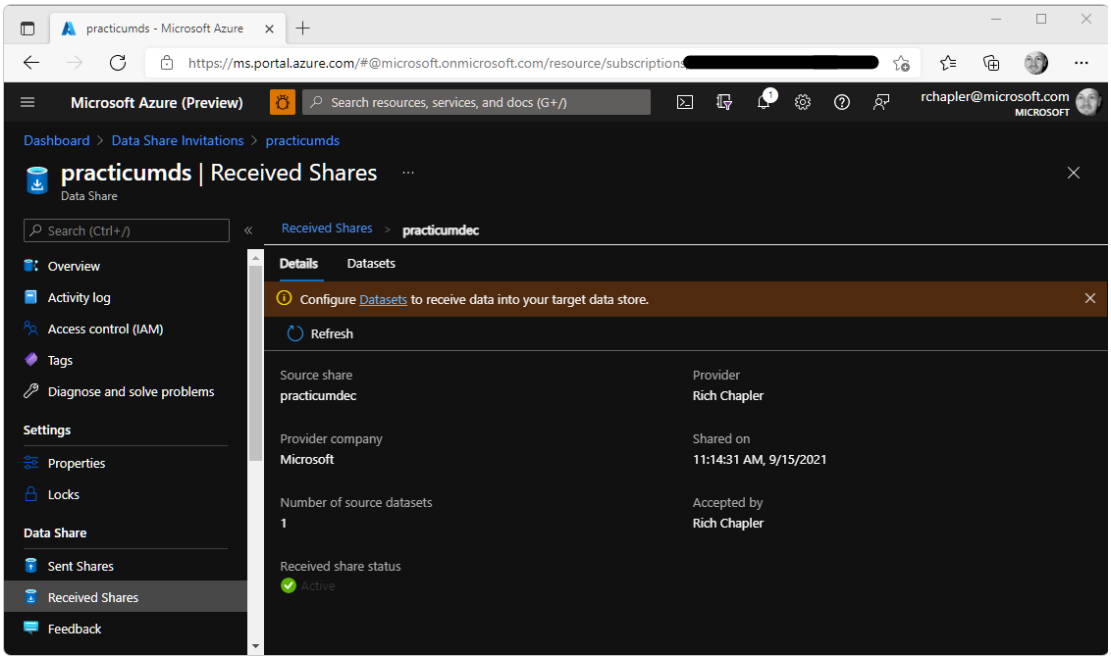
Click the **Invitation** link.



On the resulting “**…Invitation**” page, enter values for the following items:

Subscriptions	Select your subscription
Resource Groups	Select your resource group
Data Share Account	Select your Data Share account
Received Share Name	Confirm default value

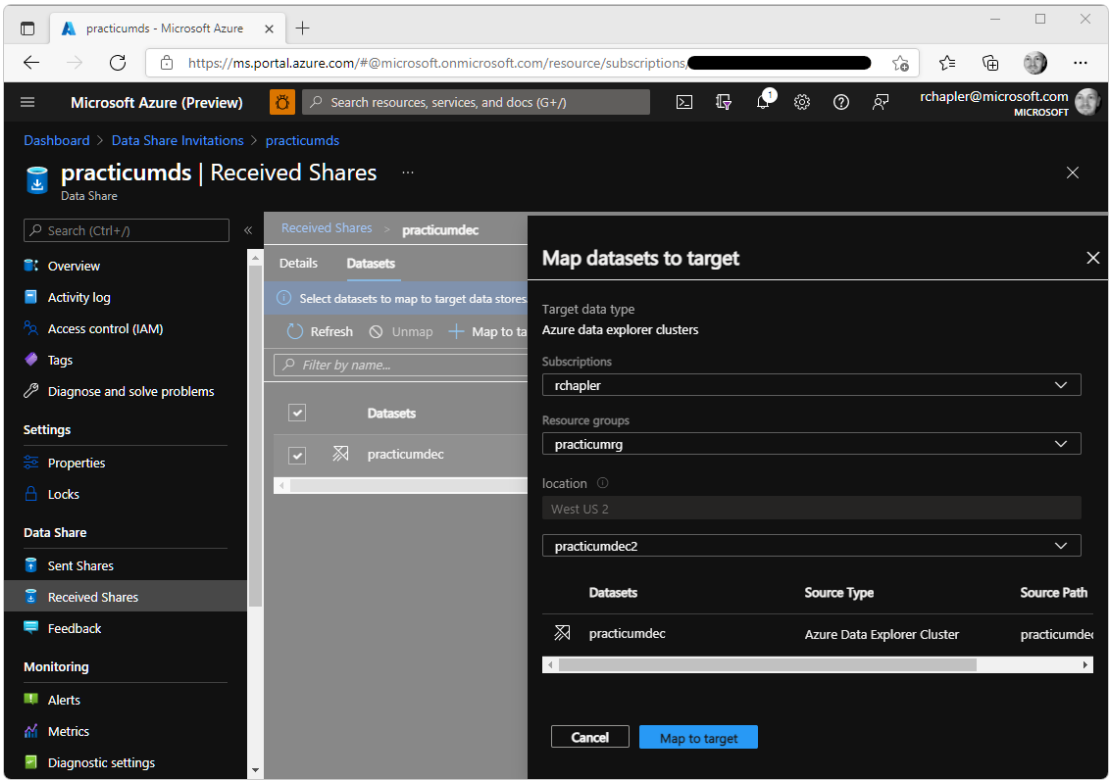
Click the "**Accept and configure**" button.



Click the **Datasets** link.

On the resulting "**Received Shares**…" page, **Datasets** tab, check the box next to your Data Explorer cluster.

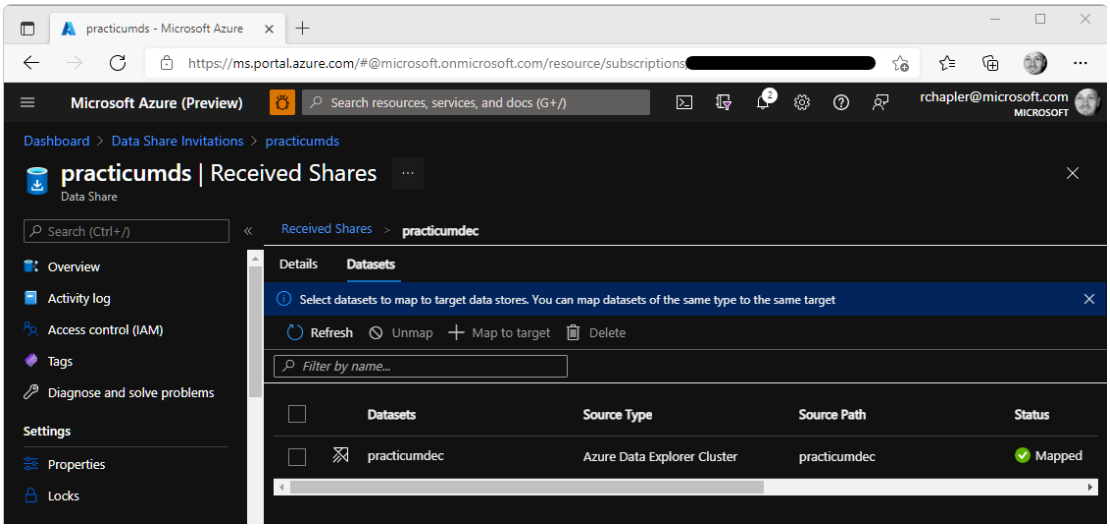
Click the "**+ Map to target**" button.



On the resulting "**Map datasets to target**" popout, enter values for the following items:

Subscriptions	Select your subscription
Resource Groups	Select your resource group
Select Kusto clusters	Select your Data Explorer cluster

Click the "**Map to target**" button.



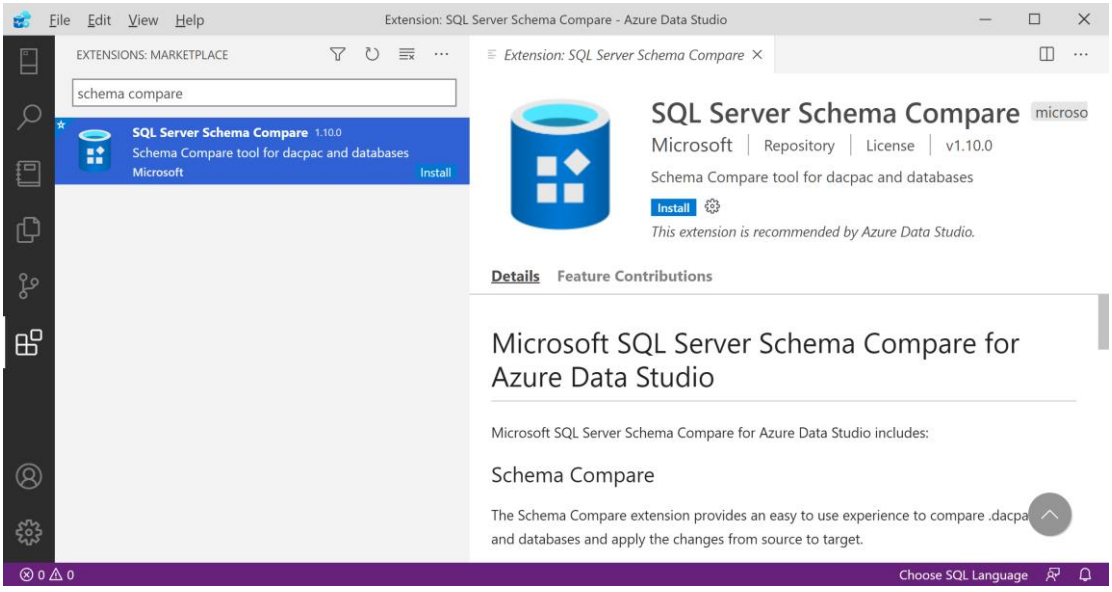
Data Studio

Browse to [Download and install Azure Data Studio - Azure Data Studio | Microsoft Docs]

Download, open, and install the "**User Installer**" for your platform.

Add Extensions

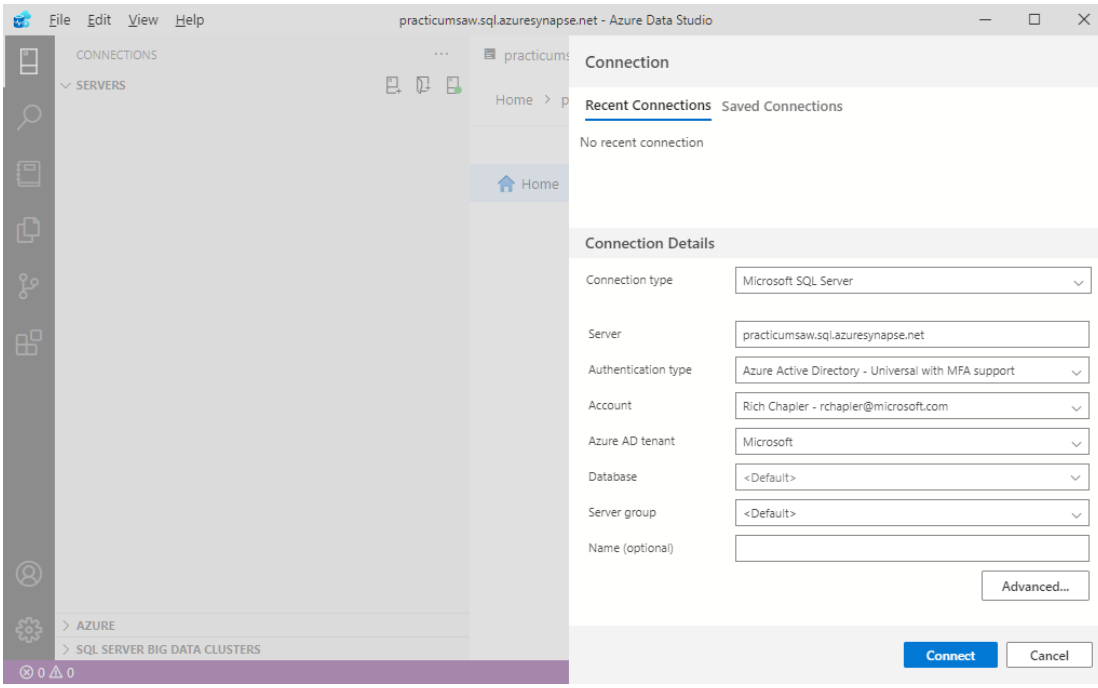
Launch Data Studio and then navigate to **Extensions** in the navigation pane.



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.

On the "**Connection Details**" popout, enter values for the following items:

Connection Type	Confirm default value, "Microsoft SQL Server"
Server	Paste the "**Dedicated SQL endpoint**" value copied in Instantiate Resources | Synapse
Authentication Type	Select "Azure Active Directory – Universal with MFA support"
Account	Confirm credentials
Azure AD Tenant	Confirm tenant

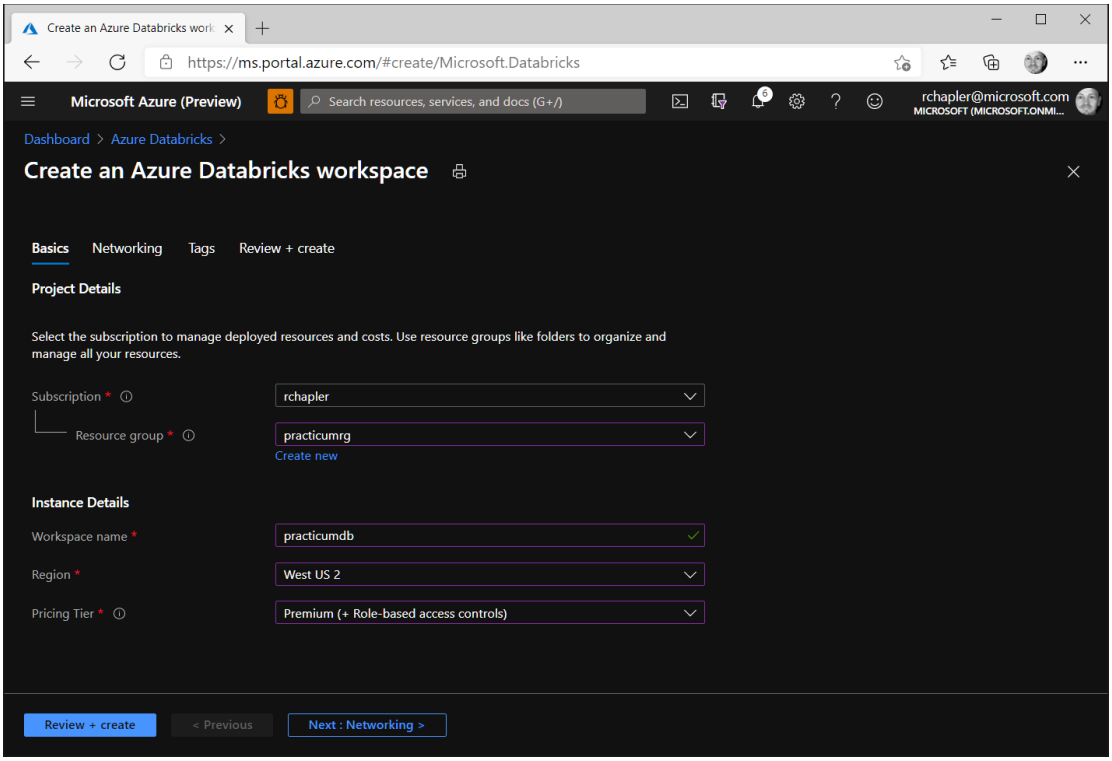
Click the **Connect** button.

Databricks

Create using Azure Portal

This exercise requires the following resource(s):

- Resource Group



On the "**Create an Azure Databricks workspace**" page, enter values for the following items:

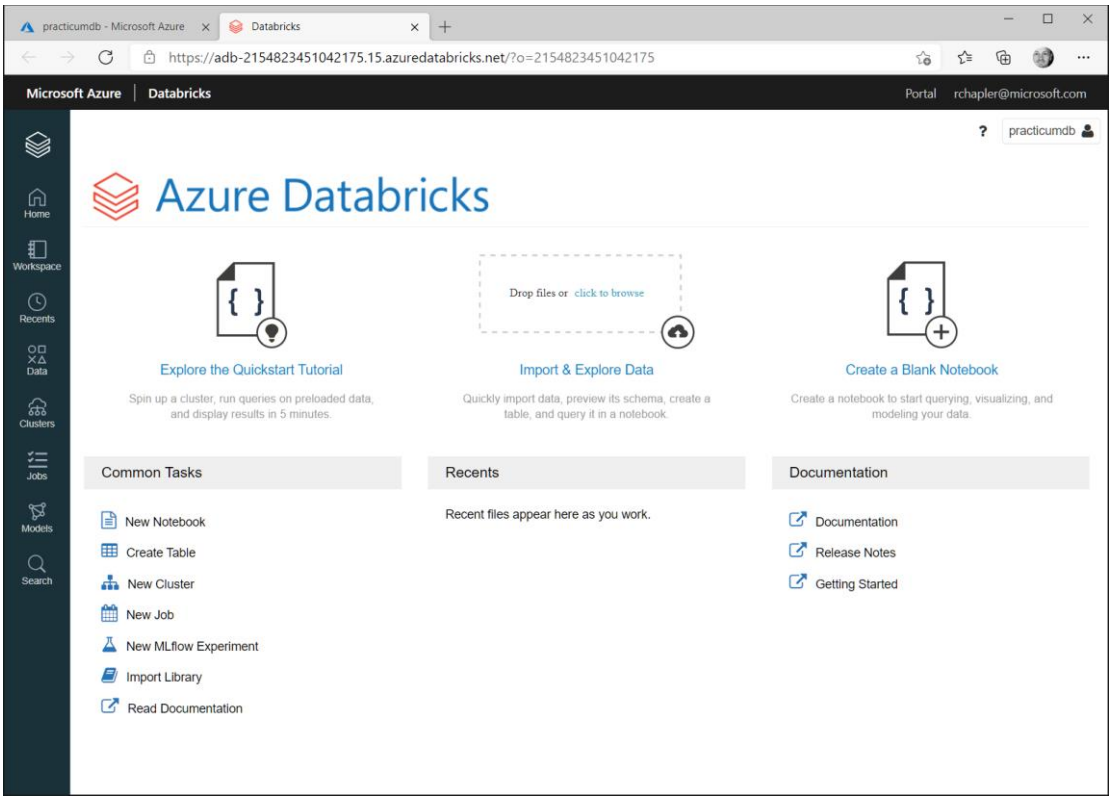
Subscription	Select your subscription
Resource Group	Select your resource group
Workspace Name	Enter a meaningful name aligned with standards
Region	Match resource group selection
Pricing Tier	Select "**Premium**…" to enable features that we will need for this exercise

No additional changes are required. Review settings on the remaining tabs.

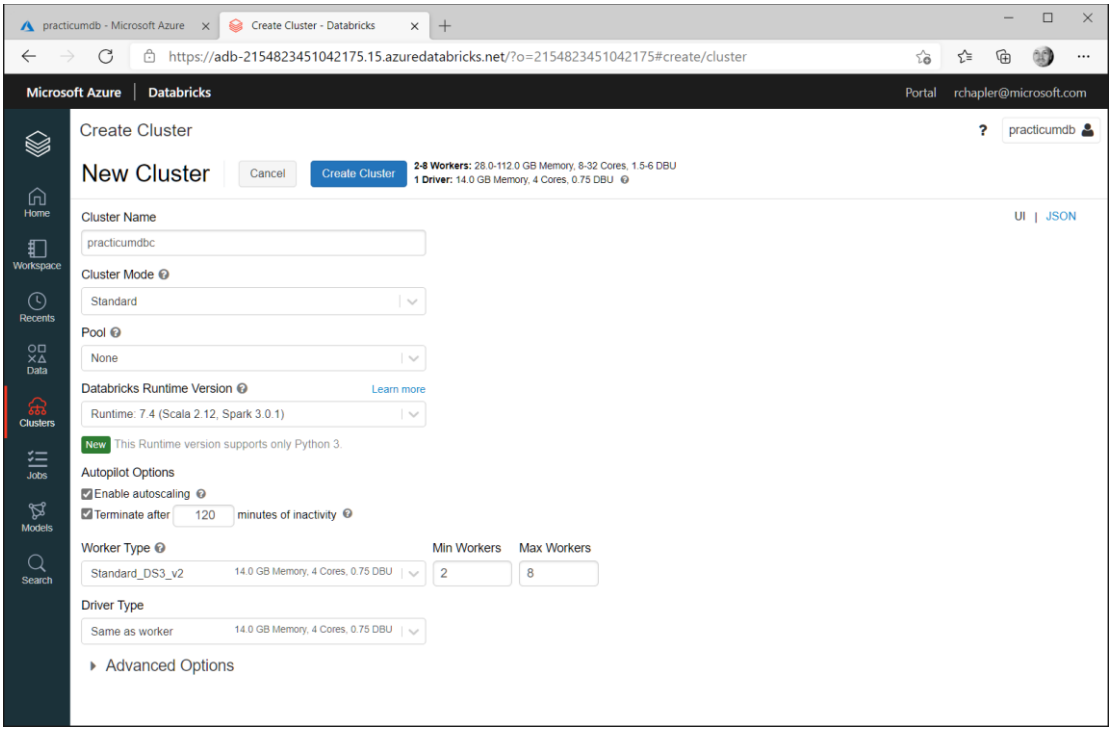
Click the "**Review + create**" button, review configuration, and then click the **Create** button.

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**".



On the "**Create Cluster**" page, enter values for the following items:

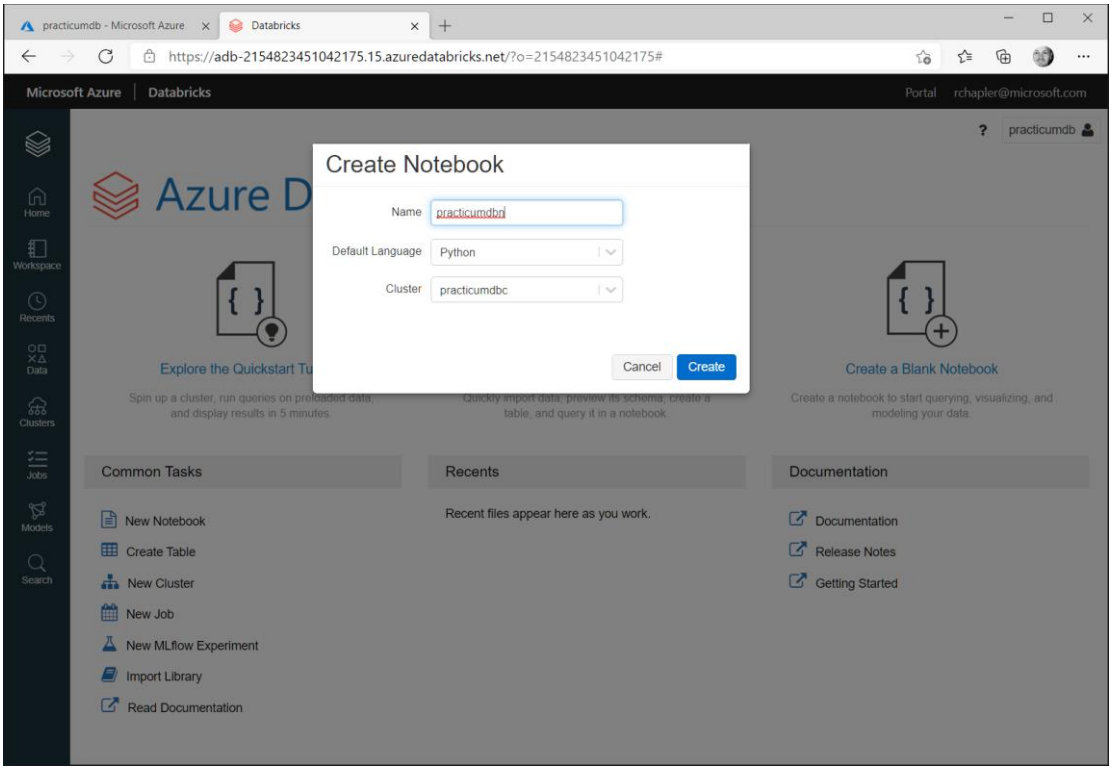
Cluster Name	Enter a meaningful name aligned with standards

No additional changes are required. Review settings on the remaining tabs.

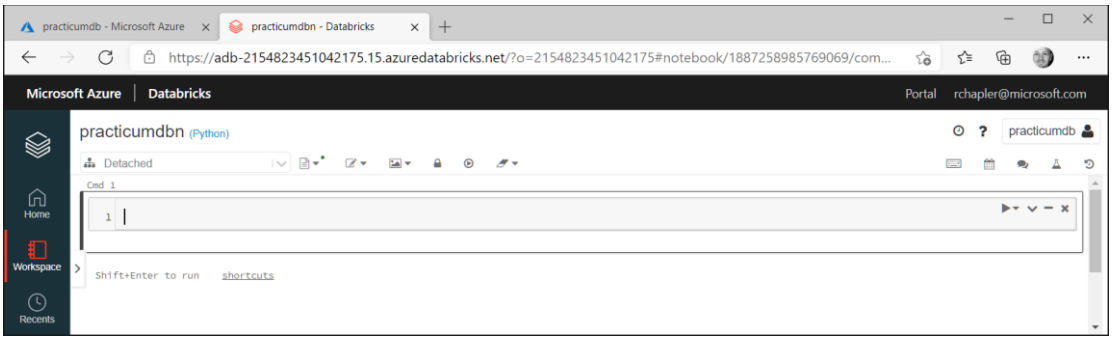
Click the "**Create Cluster**" button.

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.

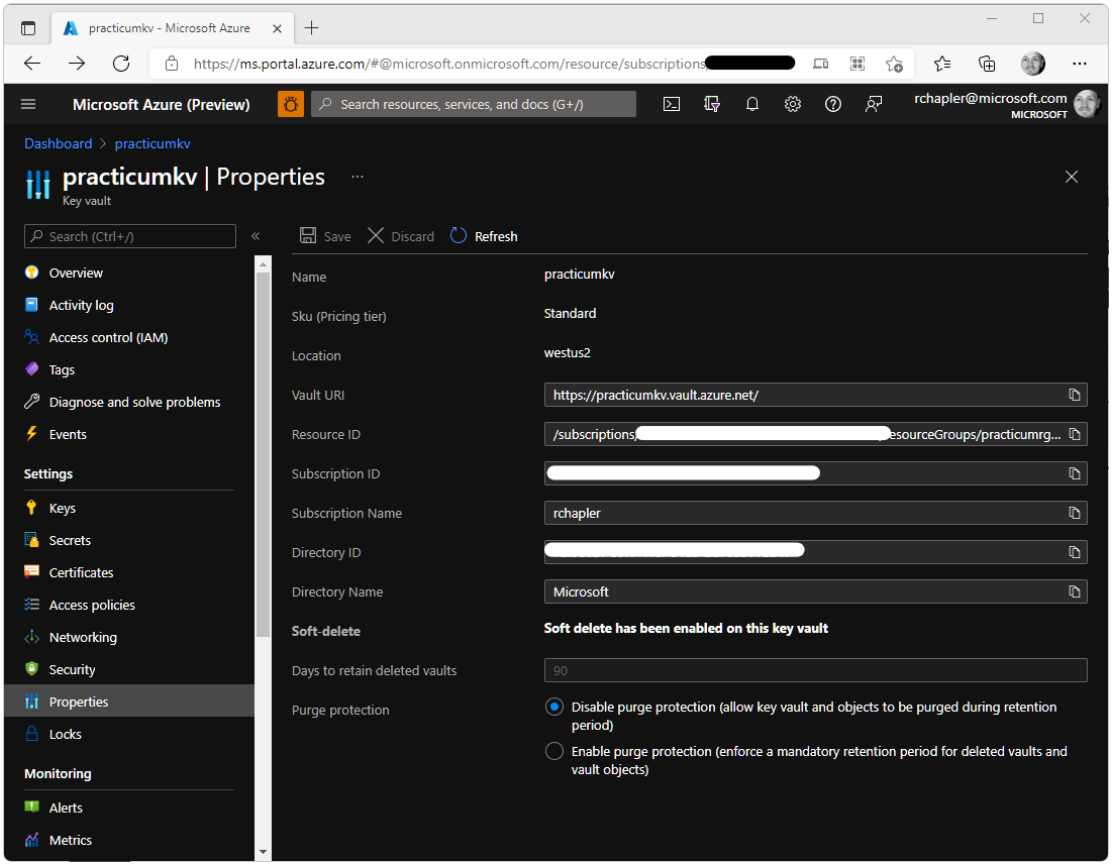


Make note of the URL {i.e., http://adb-21458...} for use in the next section.

Secret Scope

Navigate to Key Vault.

Click **Properties** in the **Settings** group of the navigation pane.

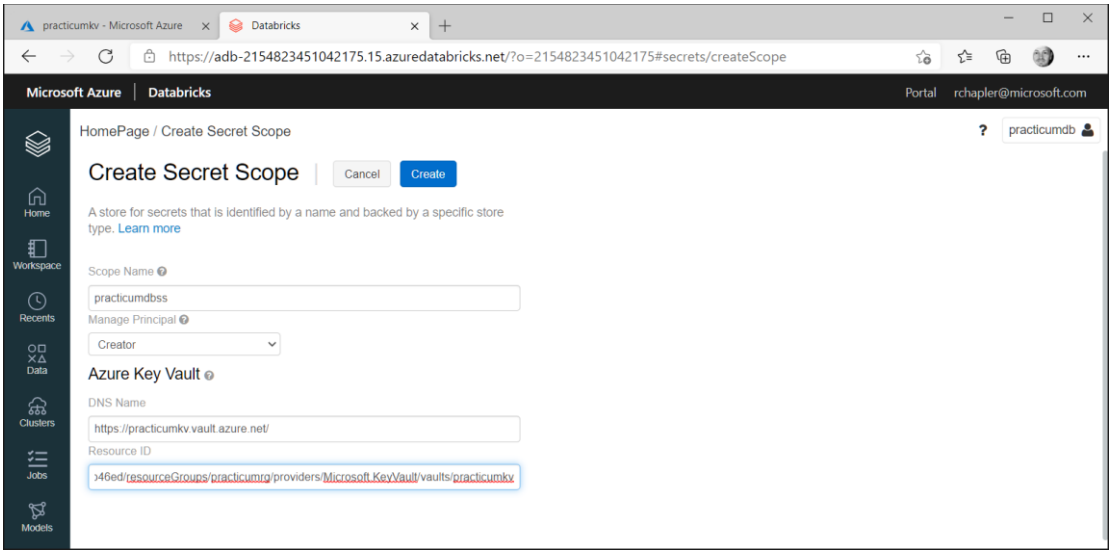


Make note of the values in “**Vault URI**” and “**Resource ID**”.

On a new tab, navigate to https://{databricksInstance}#secrets/createScope.

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



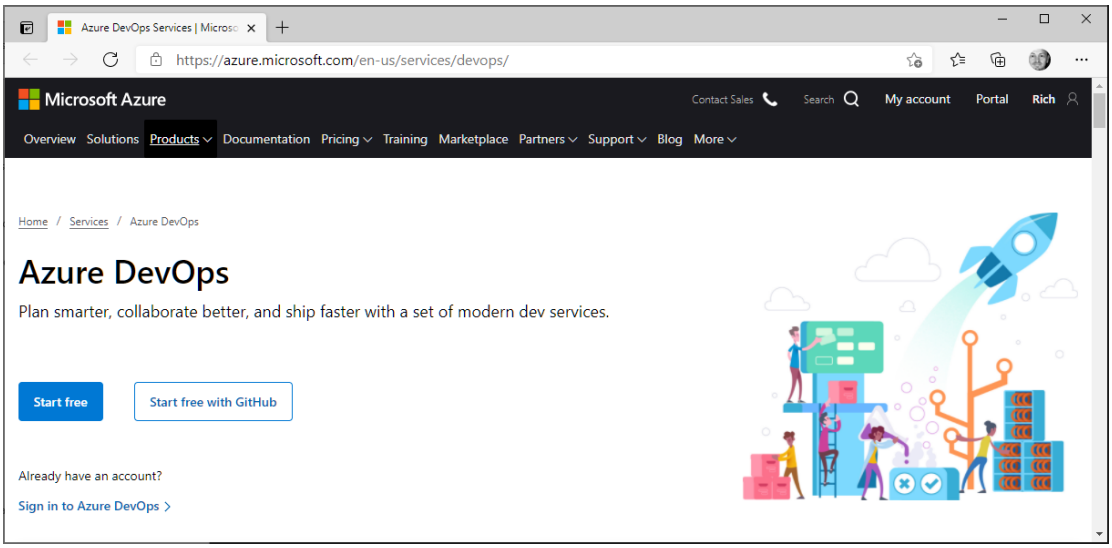
On the "**Create Secret Scope**" page, enter values for the following items:

Scope Name	Enter a meaningful name aligned with standards
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 page, click the "**New organization**" link on the navigation pane.

Click the **Continue** button.

On the "**Almost done…**" page, enter values for the following items:

Name your Azure DevOps organization	Enter a meaningful name aligned with standards Example: "adsd0" where "ads" refers to "Azure Data Solutions" and "do" is an acronym for "DevOps Organization"
We will host your projects in	Match resource group selection

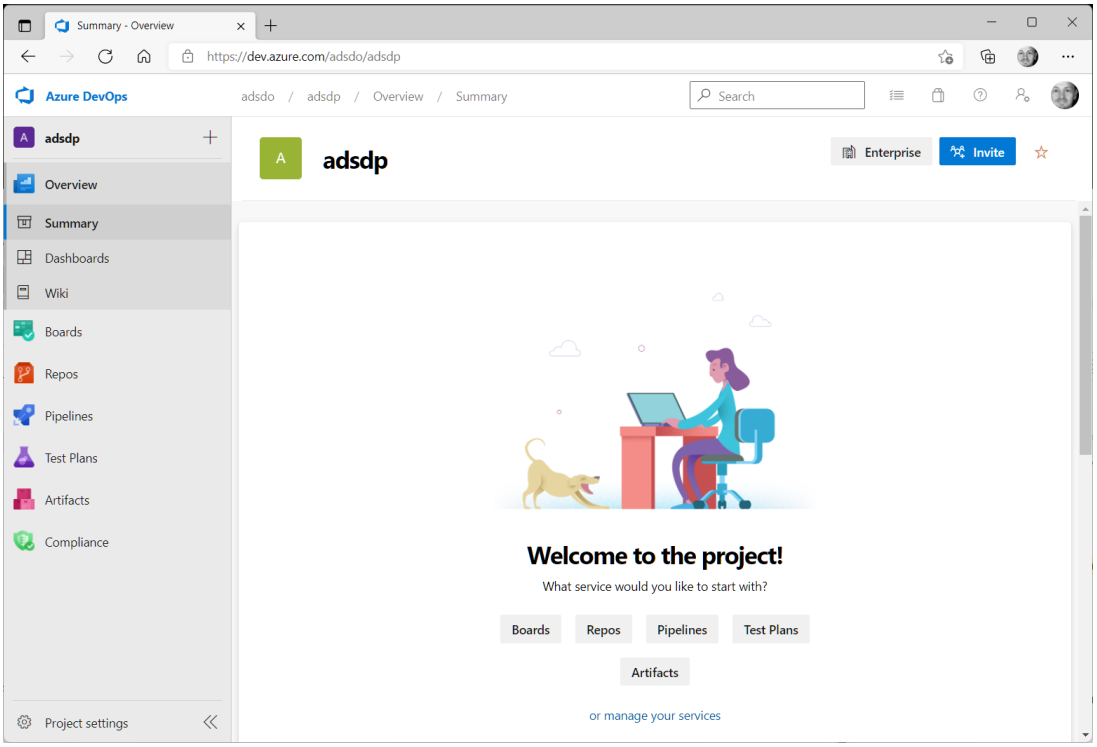
Click the **Continue** button.

Create Project

On the "**Create a project to get started**" page, enter values for the following items:

Project Name	Enter a meaningful name aligned with standards Example: "adsdp" where "ads" refers to "Azure Data Solutions" and "dp" is an acronym for "DevOps Project"
Visibility	Select the value that best aligns with your requirements

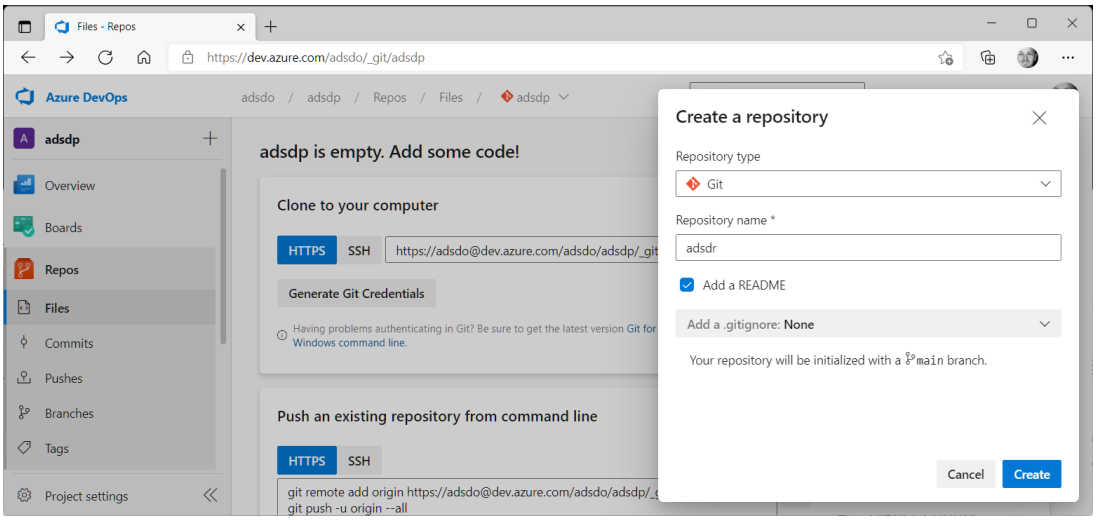
Click the "**+ Create project**" button.



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 dropdown.



On the "**Create a repository**" page, enter values for the following items:

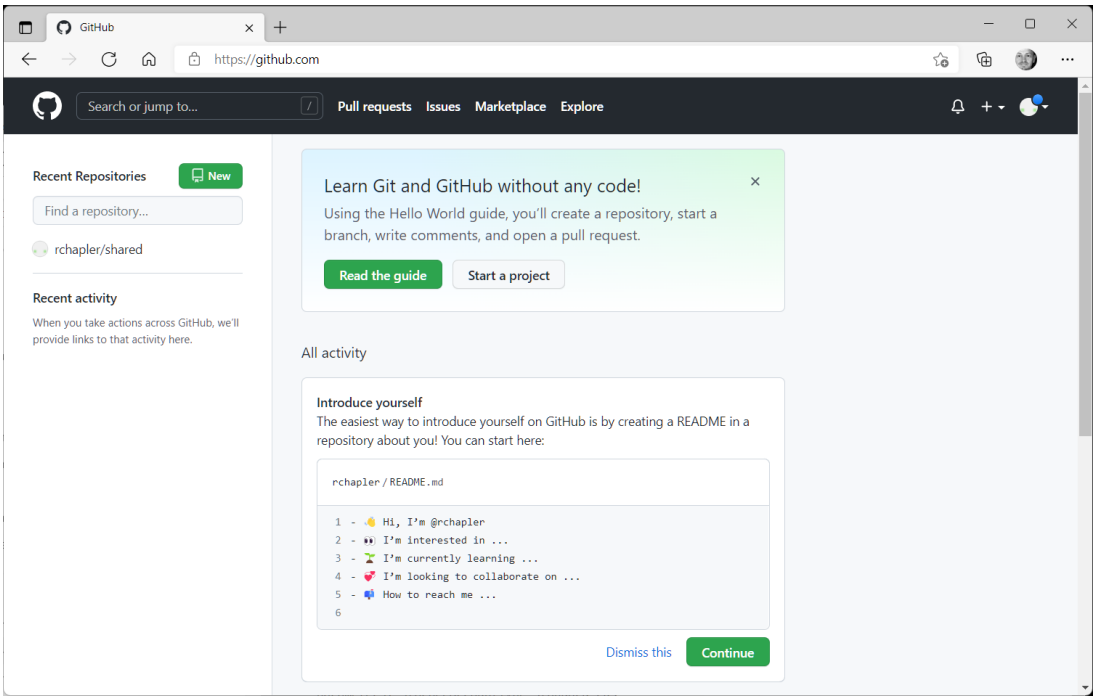
Repository Type	Confirm default selection, **Git**
Repository Name	Enter a meaningful name aligned with standards Example: "adsdr" where "ads" refers to "Azure Data Solutions" and "dr" is an acronym for "DevOps Repository"

Click the "**Create**" button.

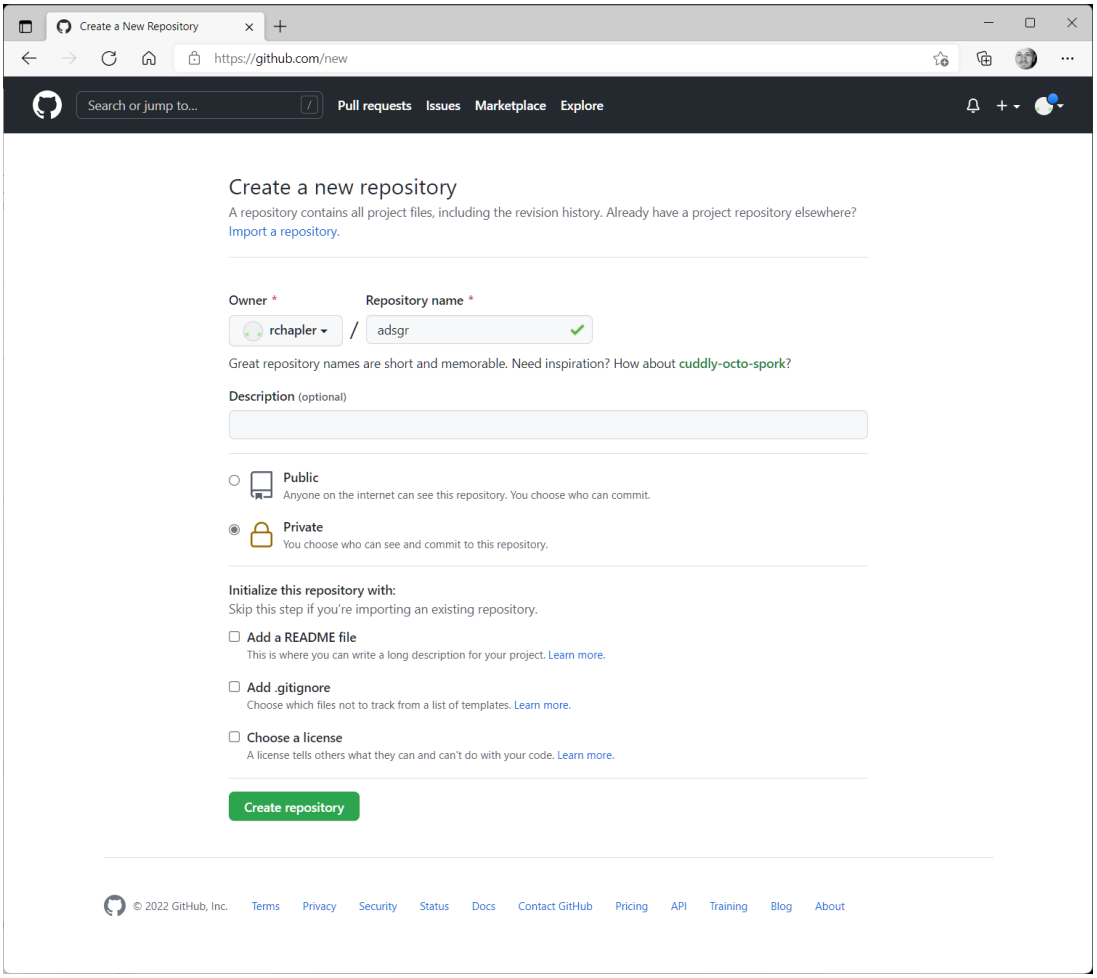
GitHub Enterprise

Browse to [Sign in to GitHub · GitHub] and sign in (or sign up, as needed).

Create Repository



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



On the "**Create a new repository**" page, enter values for the following items:

Repository Name	Enter a meaningful name aligned with standards Example: "adsgr" where "ads" refers to "Azure Data Solutions" and "gr" is an acronym for "GitHub Repository"
Public or **Private**	Choose Private (unless you have a specific reason for using Public)

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

Key Vault

Create using Azure Portal

On the "**Create Key Vault**" page, enter values for the following items:

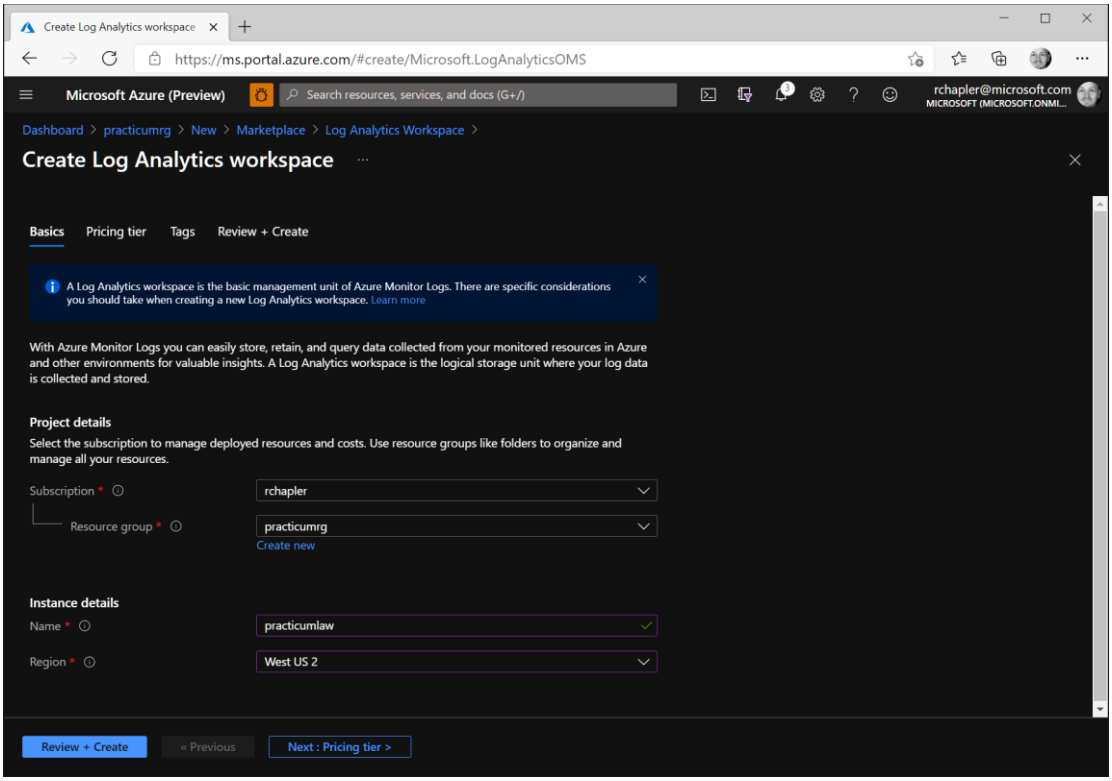
Subscription	Select your subscription
Resource Group	Select your resource group
Key Vault Name	Enter a meaningful name aligned with standards {e.g., **practicumkv**}
Region	Match resource group selection
Pricing Tier	Confirm default selection, "Standard"

No additional changes are required. Review settings on the remaining tabs.

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

Log Analytics

Create using Azure Portal



On the "**Create Log Analytics workspace**" page, enter values for the following items:

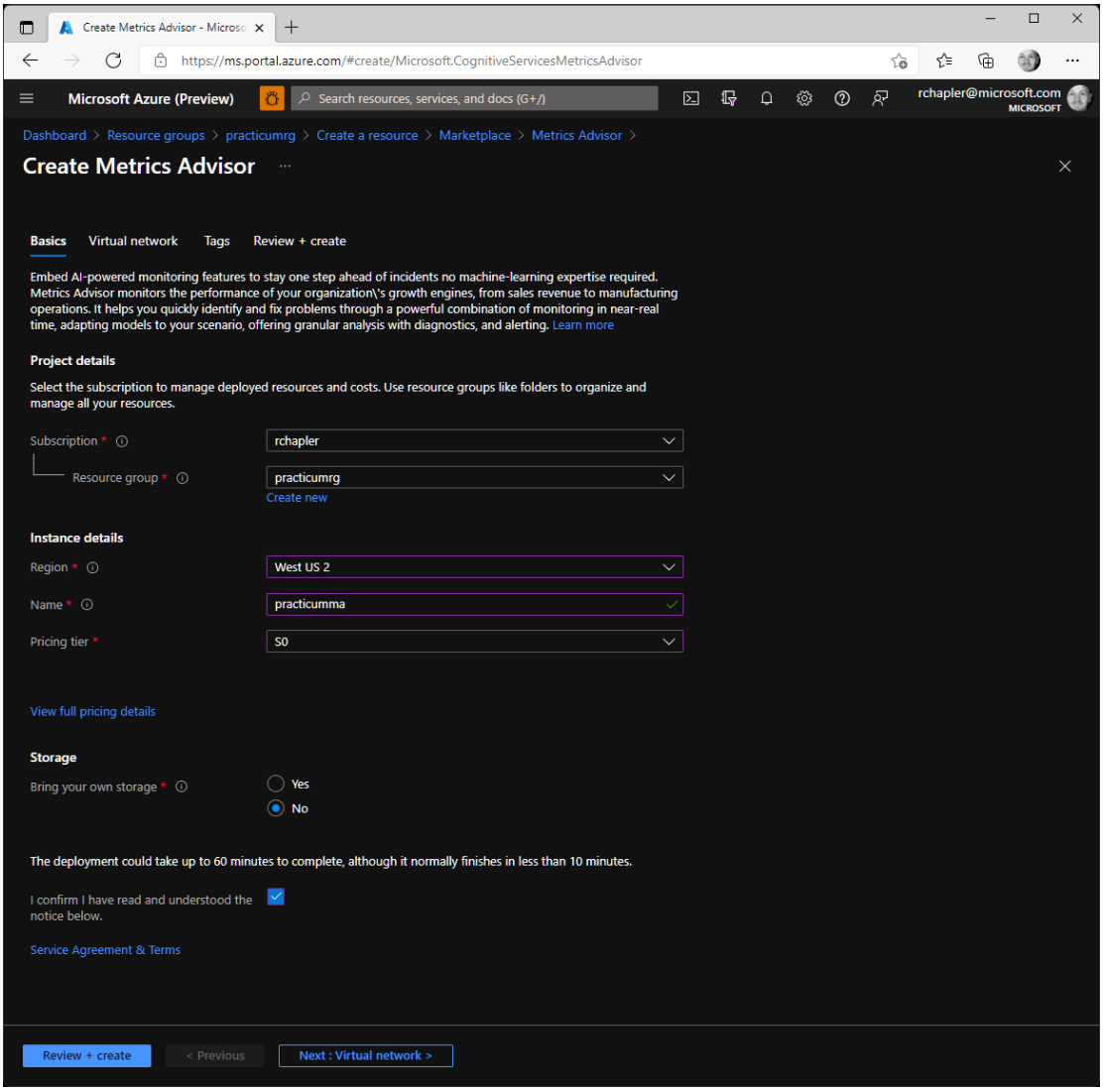
Subscription	Select your subscription
Resource Group	Select your resource group
Name	Enter a meaningful name aligned with standards
Region	Match resource group selection

No additional changes are required. Review settings on the remaining tabs.

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

Metrics Advisor

Create using Azure Portal



On the "**Create Metrics Advisor**" page, enter values for the following items:

Subscription	Select your subscription
Resource Group	Select your resource group
Region	Match resource group selection
Name	Enter a meaningful name aligned with standards
Pricing Tier	Select S0

Bring Your Own Storage	Confirm default selection, “No”
I confirm…	Check the box

No additional changes are required. Review settings on the remaining tabs.

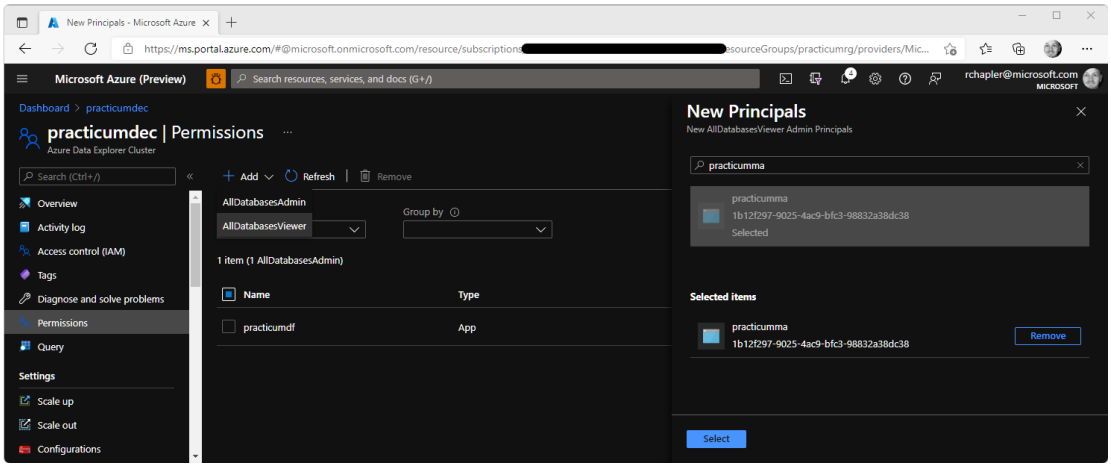
Click the “**Review + create**” button, review configuration, and then click the **Create** button.

Add Permissions to Data Explorer

This exercise requires the following resource(s):

- Data Explorer

Navigate to Data Explorer.



Click **Permissions** in the navigation pane.

Click “**+ Add**” on the resulting page.

Select **AllDatabasesViewer** from the resulting dropdown.

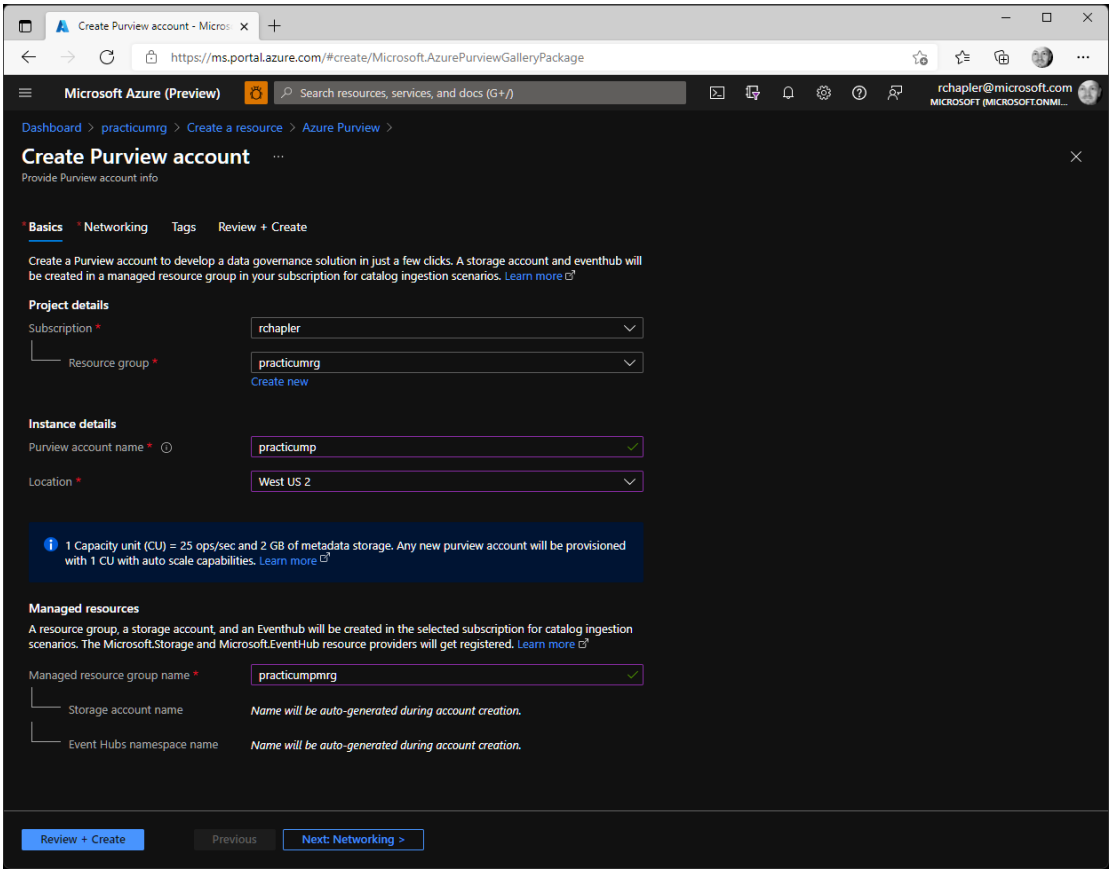
Search for and then click to **Select** Metrics Advisor.

Purview

Create using Azure Portal

This exercise requires the following resource(s):

- Key Vault
- Resource Group



On the "**Create Purview account**" page, enter values for the following items:

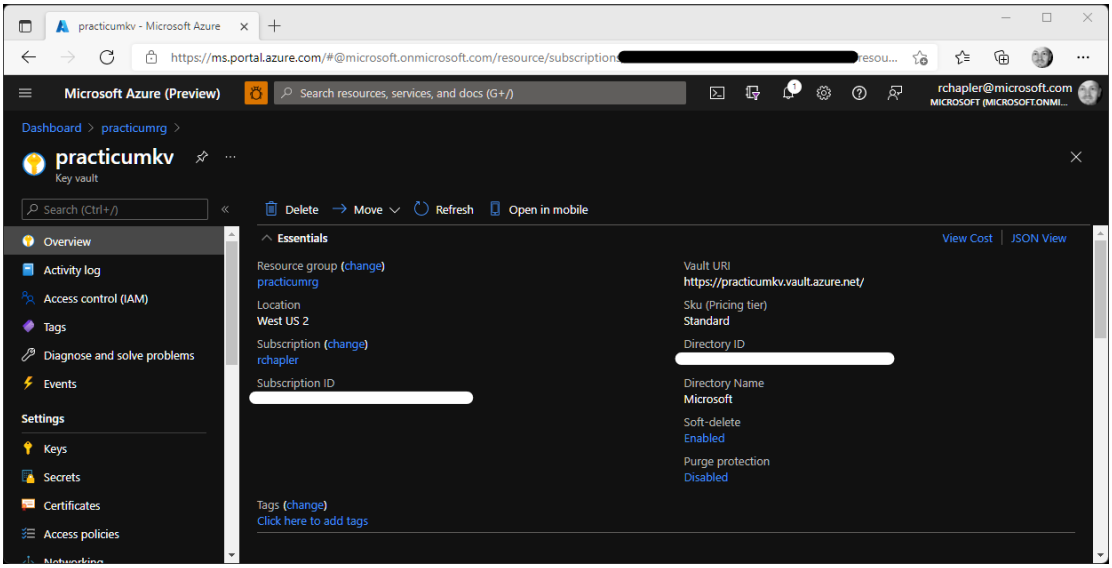
Subscription	Select your subscription
Resource Group	Select your resource group
Purview Account Name	Enter a meaningful name aligned with standards
Location	Match resource group selection
Managed Resource Group…	Self-explanatory

No additional changes are required. Review settings on the remaining tabs.

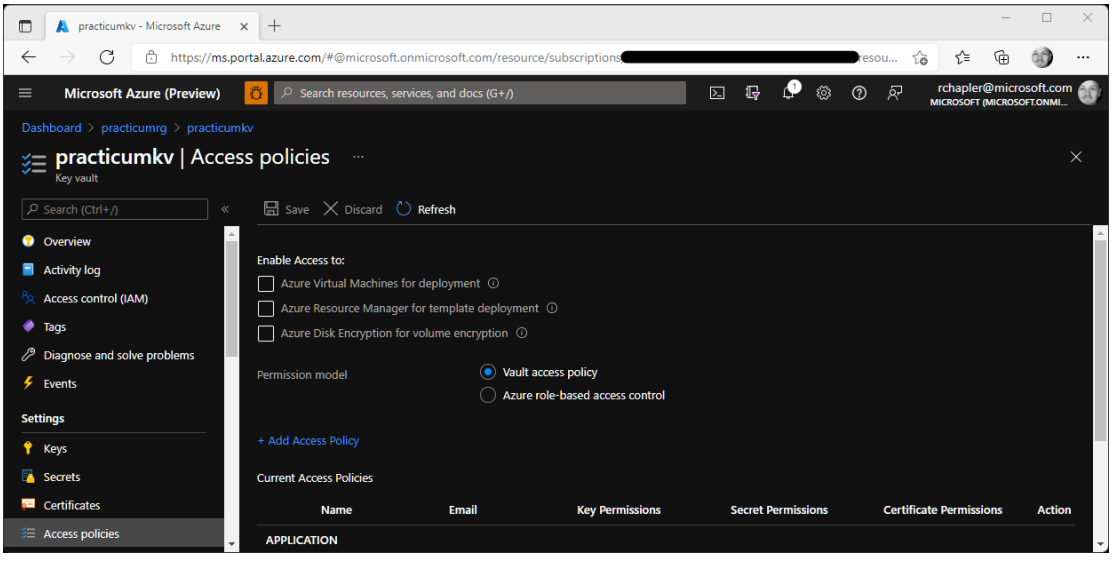
Click the “**Review + Create**” button, review configuration, and then click the **Create** button.

Add Access Policy to Key Vault

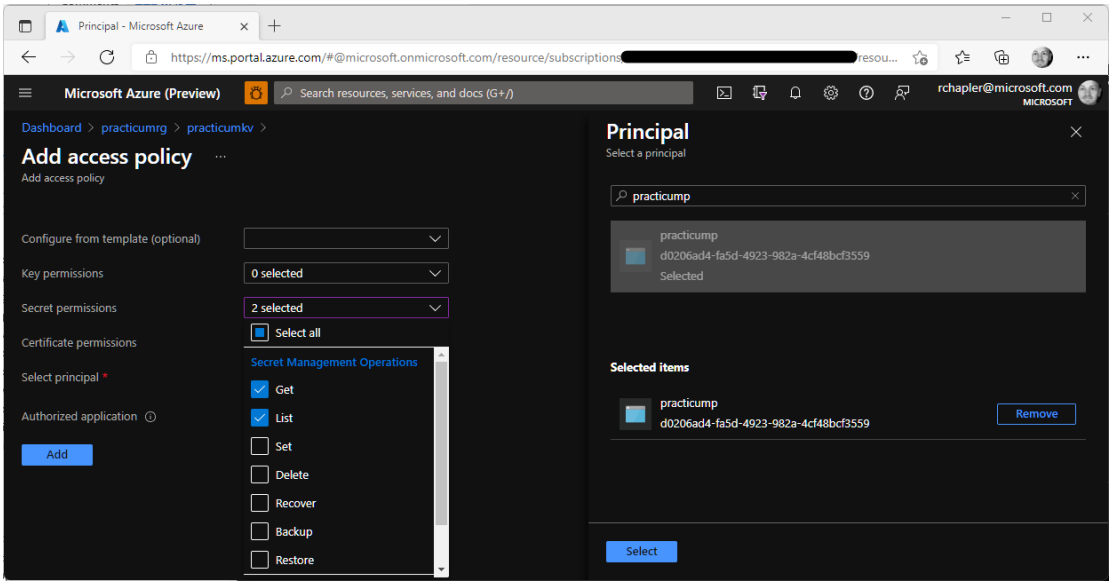
Navigate to the Key Vault.



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



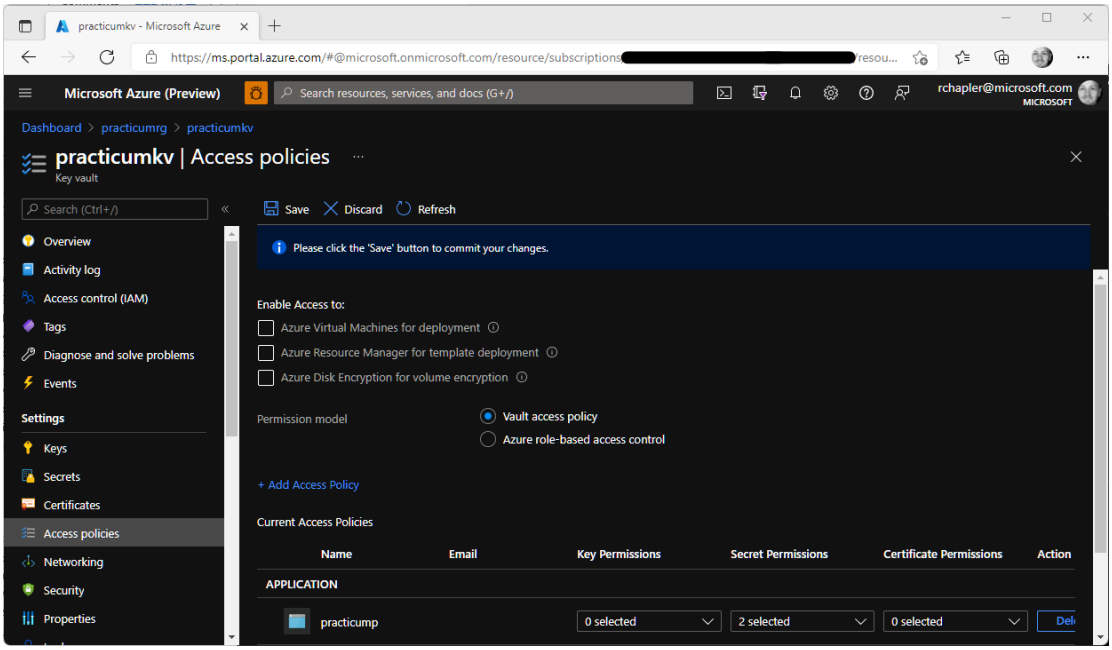
Click the “+ Add Access Policy” link.



On the "**Add access policy**" page, enter values for the following items:

Secret Permissions	Select **Get** and **List** in the dropdown
Select Principal	Search for, and then **Select** the managed identity for Purview

Click the **Add** button.



Click the **Save** button.

Connect Key Vault

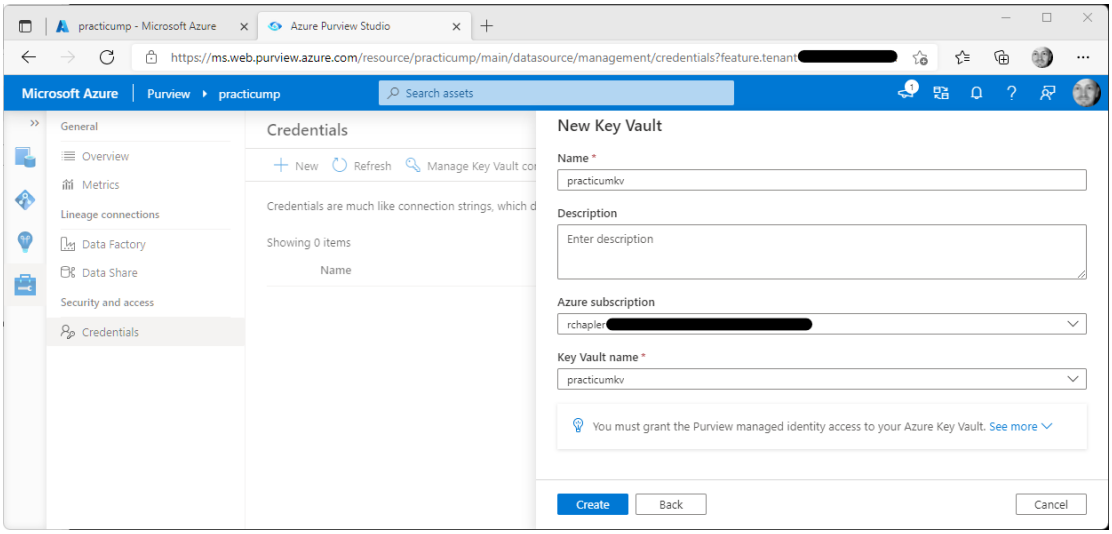
Navigate to Purview and click on the "**Open Purview Studio**" button.

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 popout.



On the "**New Key Vault**" popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Subscription	Select your subscription
Key Vault Name	Select your Key Vault

Click the **Create** button.

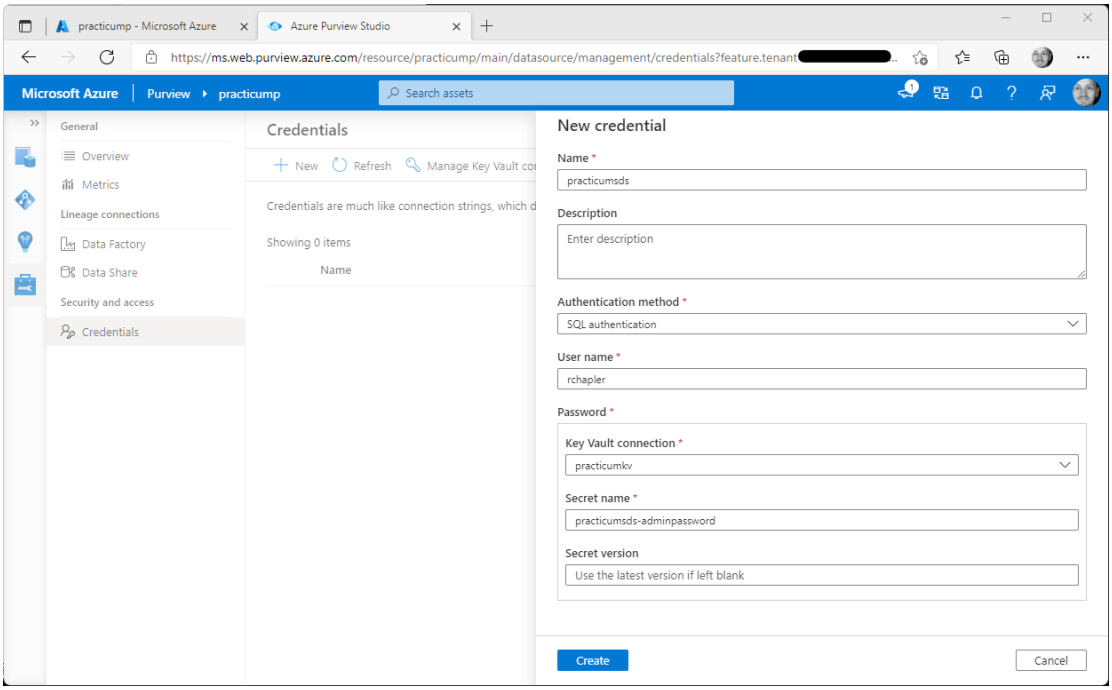
On the resulting "**Confirm granting access**" popup, review and then click the **Confirm** button.

Confirm the addition of Key Vault on the **"Manage Key Vault connections**" popout.

Click the **Close** button.

Add New Credential

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



On the resulting "**New Credential**" popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
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

No additional changes are required. Review settings on the remaining tabs.

Click the **Create** button.

Resource Group

Create using Azure Portal

On the "**Create a Resource Group**" page, enter values for the following items:

Subscription	Select your subscription
Resource Group	Enter a meaningful name aligned with standards {e.g., **shared**}
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

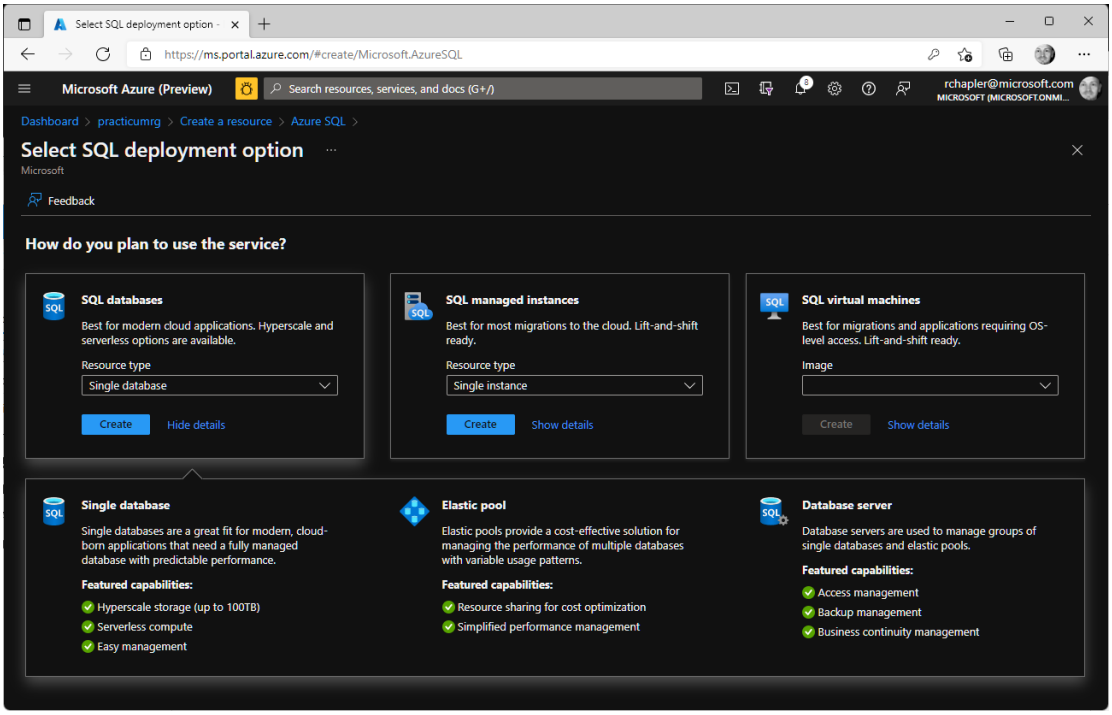
No additional changes are required. Review settings on the remaining tabs.

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

SQL

Create using Azure Portal

Step 1: Create Resource



On the "**Select SQL deployment option**" page, select the deployment option appropriate to your use case. For most examples, we will typically select "**Single database**" in the "**SQL databases**" > "**Resource type**" dropdown.

Click the **Create** button.

On the "**Create SQL Database**" page, **Basics** tab, enter values for the following items:

Resource Group	Select your resource group
Database Name	Enter a meaningful name aligned with standards {e.g., **sharedsd**}
Server	Click the "**Create new**" link, enter values in the resulting "**New server**" popout and then click the **OK** button
…use SQL elastic pool?	Confirm default value, **No**
Compute + Storage	Click the "Configure database" link

On the resulting **Configure** page, enter values for the following items:

Service Tier	Confirm default value, "**General Purpose**…"
Compute Tier	Click the **Serverless** radio button
Max vCores	Confirm minimum value
Min vCores	Confirm minimum value
Enable Auto-Pause	Confirm default value, **checked**
Days | **Hours** | **Minutes**	Confirm default values, 0 | 1 | 0
Data Max Size (GB)	Set minimum appropriate value

Review the automatically generated "**Cost Summary**" and then click the **Apply** button.

Back on and further down on the "**Create SQL Database**" page, **Basics** tab, enter values for the following items:

Backup Storage Redundancy	Confirm default value, "Locally-redundant backup storage"

If you want to add a sample database, navigate to the "**Additional settings**" tab and enter values for the following items:

Use Existing Data	Select the "Sample" option

No additional changes are required. Review settings on the remaining tabs.

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

Step 2: Prepare Key Vault

Create Secret for Admin Password

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

Click the "**+ Generate/Import**" button.

On the "**Create a secret**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards {e.g., "**sharedsds-adminpassword**"}
Value	Enter the value used when creating the SQL Database Server

No additional changes are required. Review settings on the remaining tabs.

Click the **Create** button.

Step 3: Configure Firewall

Navigate to your Azure SQL Server and click the "**Show firewall settings**" link on the **Overview** page. Set "Allow Azure services and resources to access this server" to "Yes". Click the **Save** button.

Storage Account

Create using Azure Portal

On the "**Create Storage Account**" page, enter values for the following items:

Subscription	Select your subscription
Resource Group	Select your resource group
Storage Account Name	Enter a meaningful name aligned with standards
Location	Match resource group selection
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)"

No additional changes are required. Review settings on the remaining tabs.

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

Synapse

Create using Azure Portal

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Key Vault
- Resource Group
- SQL

Step 2: Create Resource

On the "**Create Synapse workspace**" page, enter values for the following items:

Subscription	Select your subscription
Resource Group	Select your resource group
Managed Resource Group	This additional Resource Group will hold ancillary resources created specifically for Synapse Because you are likely to have more than one managed resource group, consider using a consistent naming pattern {e.g., "**sharedmrg-s**" for Synapse and "sharedmrg-x" for X}
Workspace Name	Enter a meaningful name aligned with standards {e.g., **practicums**}
Region	Match resource group selection
Select Data Lake Storage Gen2	Confirm default, "**From subscription**"
Account Name	Select or create a data lake {e.g., **shareddl**}
File System Name	Select or create a data lake file system {e.g., **shareddlfs**}
Assign myself…	Check to assign necessary permissions

Click the "**Next: Security >**" button and enter values for "**SQL Server Admin Login**" and "**SQL Password**".

No additional changes are required. Review settings on the remaining tabs.

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

Step 3: Prepare Key Vault

Create Secret for Admin Password

Navigate to Key Vault.

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

On the "**Create a secret**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards {e.g., "**practicums-adminpassword**"}
Value	Enter the value used when creating the Synapse Workspace

No additional changes are required. Review settings on the remaining tabs.

Click the **Create** button.

Add Access Policy to Key Vault

Navigate to Key Vault and then click "**Access Policies**" in the **Settings** group of the navigation pane.

Click the "**+ Add Access Policy**" link and on the resulting "**Add access policy**" page, enter values for the following items:

Secret Permissions	Select **Get** and **List** in the dropdown
Select Principal	Search for, and then **Select** the managed identity for your Synapse Workspace

Click the **Add** button and then click the **Save** button.

Step 4: Add Permissions to Data Explorer

This exercise requires the following resource(s):

- Data Explorer

Navigate to Data Explorer.

Click **Permissions** in the navigation pane.

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

Select **AllDatabasesViewer** for read permissions and **AllDatabasesAdmin** for write, etc. permissions from the resulting dropdown.

Search for and then click to **Select** Synapse.

Step 5: 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**" popout, **Basics** tab, enter values for the following items:

Data Explorer Pool Name	Enter a meaningful name aligned with standards {e.g., **shareddep**}
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**" popout, "**Additional settings**" tab, and then enter values for the following items:

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

No additional changes are required. Review settings on the remaining tabs.

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

Create Data Explorer 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 dropdown.

On the resulting "**Create Data Explorer database**…" popout, enter values for the following items:

Pool Name	Enter a meaningful name aligned with standards {e.g., **shareddep**}
Name	Enter a meaningful name aligned with standards {e.g., **sharedded**}
Default Retention…	Confirm default, **365**
Default Cache…	Confirm default, **31**

Click the **Create** button.

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.

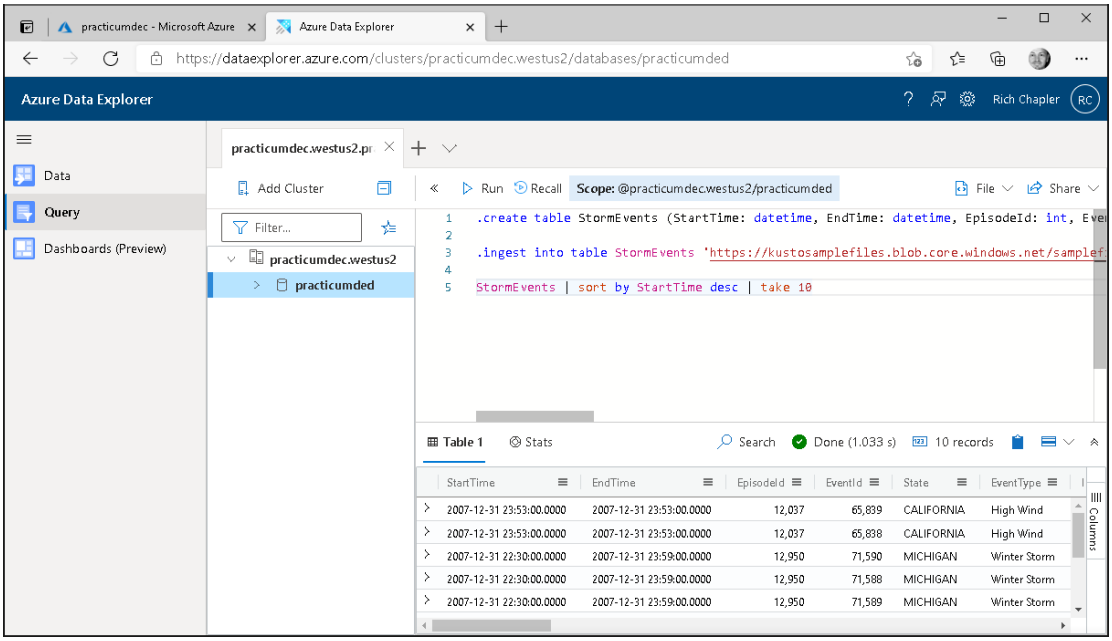
Right click on your Data Explorer Database {e.g., **sharedded**} and select "New KQL script" from the resulting dropdown.

Execute the following query 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 dropdown.

On the "**Create SQL database**" popout, enter values for the following items:

Select SQL Pool Type	Confirm default, **Serverless**
Database	Enter a meaningful name aligned with standards

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 popout, enter values for the following items:

Apache Spark Pool Name	Enter a meaningful name aligned with standards
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.

Step 6: Create Linked Services

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

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

Select "**Linked services**" in the "**External connections**" grouping of the resulting navigation and then click the "**+ New**" button.

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

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

Click the **Continue** button.

On the "**New linked service**…" popout, enter values for the following common items:

Name	Enter a meaningful name aligned with standards
Connect via…	Confirm default selection, "**AutoResolveIntegrationRuntime**"

And enter values for resource specific items…

Data Explorer

Authentication Method	Select "Managed Identity"
Selection Method	Select "Enter manually"
Endpoint	Enter the URI for your Data Explorer Pool {e.g., https://shareddep.practicums.kusto.azuresynapse.net}
Database	Select your Data Explorer Database

Data Lake

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

Key Vault

Selection Method	Confirm default selection, "**From Azure subscription**"

Subscription	Select your subscription
Azure Key Vault Name	Select your Key Vault

SQL

Selection Method	Confirm default selection, "**From Azure subscription**"
Subscription	Select your subscription
Server Name	Select your Azure SQL Server {e.g., **sharedsds**}
Database Name	Select your Azure SQL Database {e.g., **sharedsd**}
Authentication Type	Select "SQL authentication"
User Name	Enter the "**server admin login**" value used during instantiation of the SQL Database Server
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 Azure SQL Server administrator password

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

On the "**Create virtual machine**" page, **Basics** tab, enter values for the following items:

Subscription	Select your subscription
Resource Group	Select your resource group
Virtual Machine Name	Enter a meaningful name aligned with standards
Region	Match resource group selection
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**" page, **Basics** tab, enter values for the following items:

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

On the "**Create virtual machine**" page, "**SQL Server settings**" tab, enter values for the following items:

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)

No additional changes are required. Review settings on the remaining tabs.

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.

On the "Internet Explorer Enhanced Security Configuration" popup, click Off under Administrators and Users, and then click the OK button.

Sample Database

Browse to AdventureWorks sample databases - SQL Server | Microsoft Docs 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 dropdown
- On the "**Restore Database**" popup, enter AdventureWorks in the "**To database**" text box, "**Destination for restore**" interface grouping
- On the "**Restore Database**" popup, click the "**From device**" radio button in the "**Source for restore**" interface grouping
- Click the **ellipses** button and in the resulting "**Specify Backup**" popup, 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**" popup, check the box next to new item in "**Select the backup sets to restore**" and then click the **OK** button

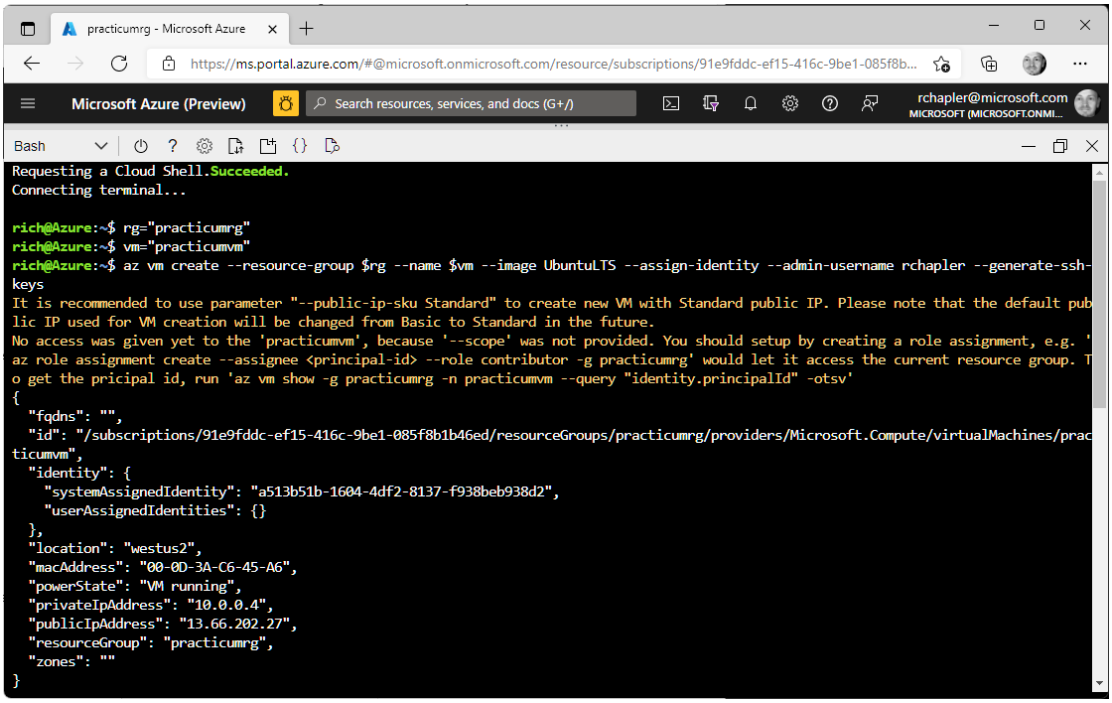
When you have successfully completed these steps, you will see a popup that says "**The restore of database 'AdventureWorks**" completed successfully.

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**" dropdown.



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

```
rg="practicumrg"
```

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

```
vm="practicumvm"
```

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
```

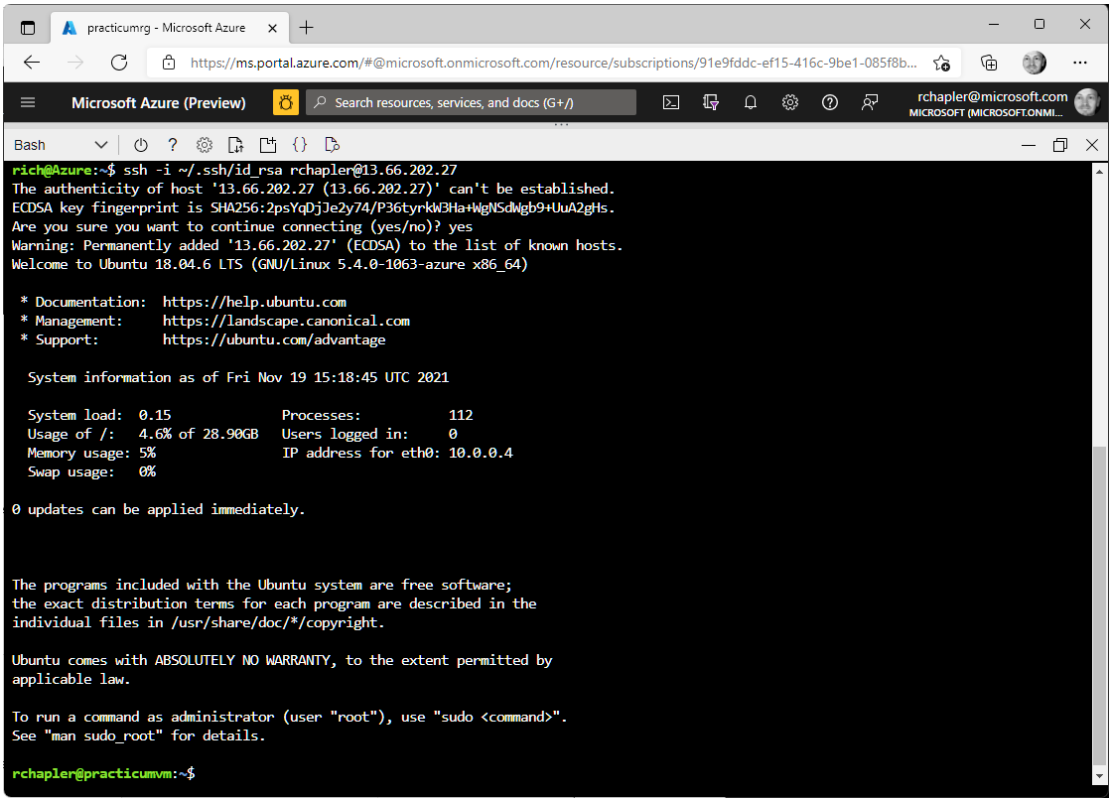
Successful execution will create the following resources:

- 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@practicumvm**.

Source

Migration from On-Prem

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

Step 1: Understand Use Case

Work with your customer to understand requirements; 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 in Azure, and the on-prem source will be deprecated over time”

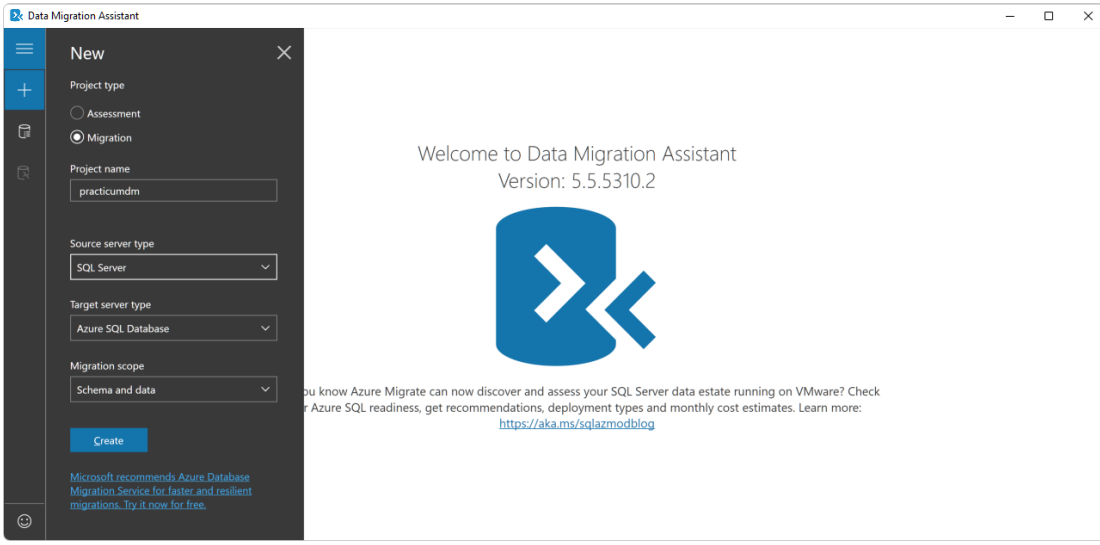
Step 2: Instantiate Prerequisites

This exercise requires the following resource(s):

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

Step 3: Create Migration Project

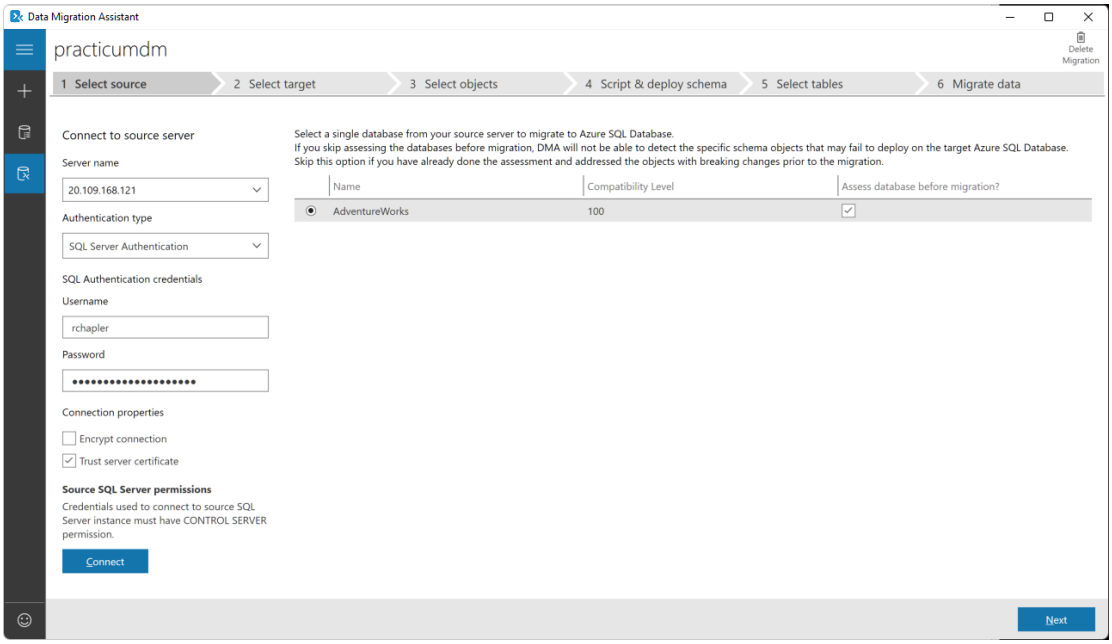
Open **Data Migration Assistant** and click **+** to create a new project



On the "**Create Data Share**" popout, enter values for the following items:

Project Type	Select the **Migration** radio button
Project Name	Enter a meaningful name aligned with standards
Source Server Type	Select "SQL Server"
Target Server Type	Select "Azure SQL Database"
Migration Scope	Select "Schema and data"

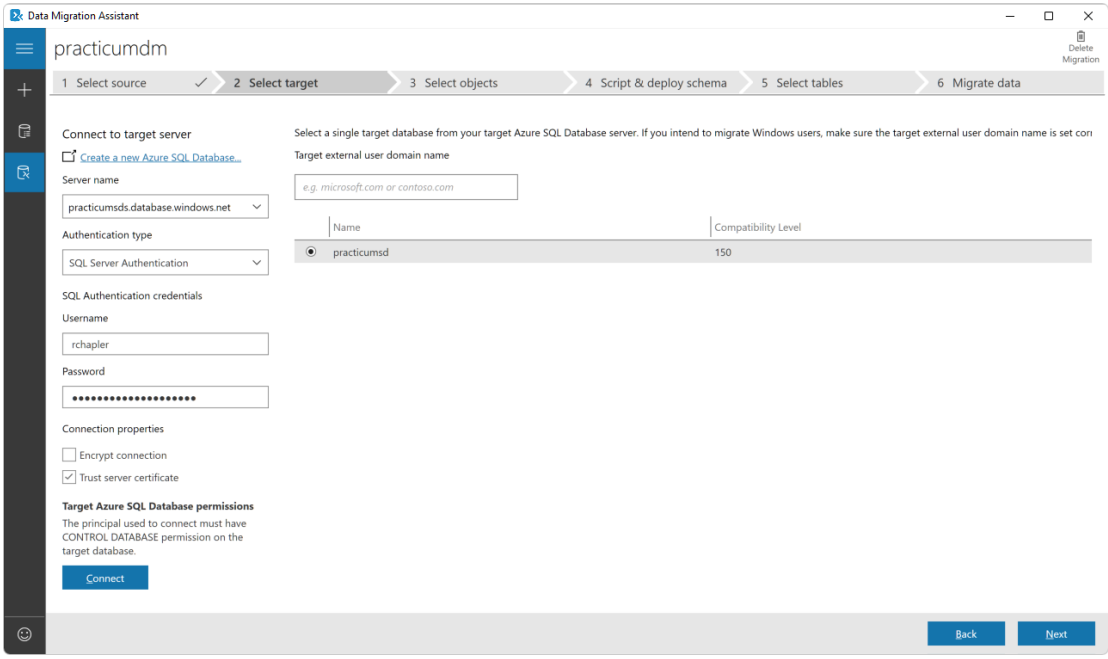
Click the **Create** button.



On the "**1. Select source**" page, enter values for the following items:

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

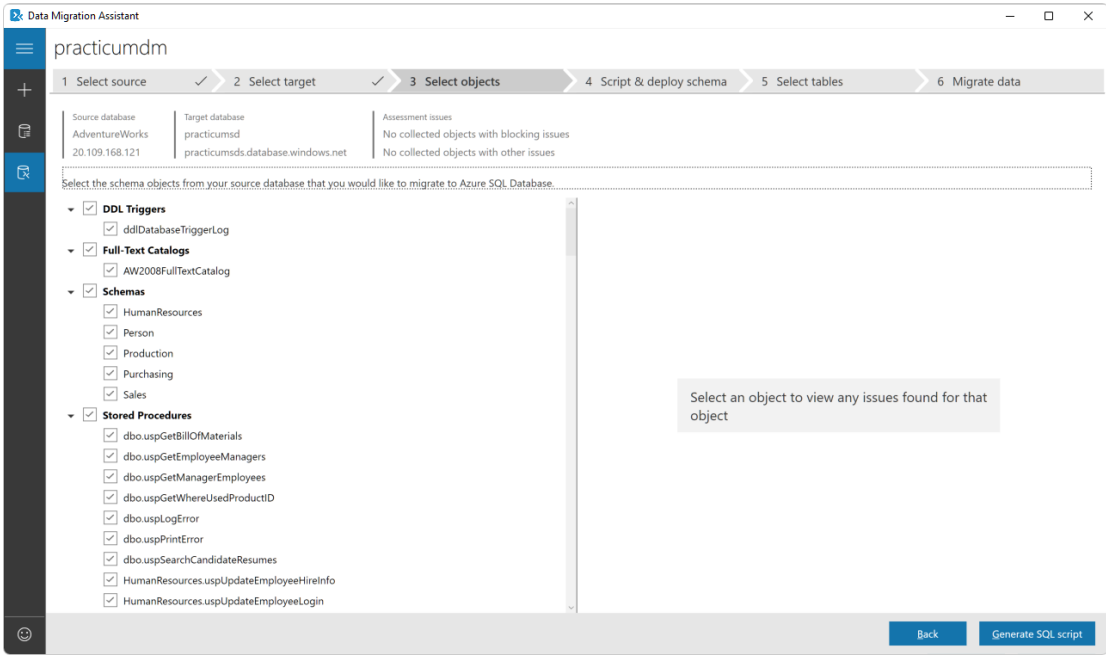
Click the **Connect** button, confirm selection of **AdventureWorks** and then click the **Next** button.



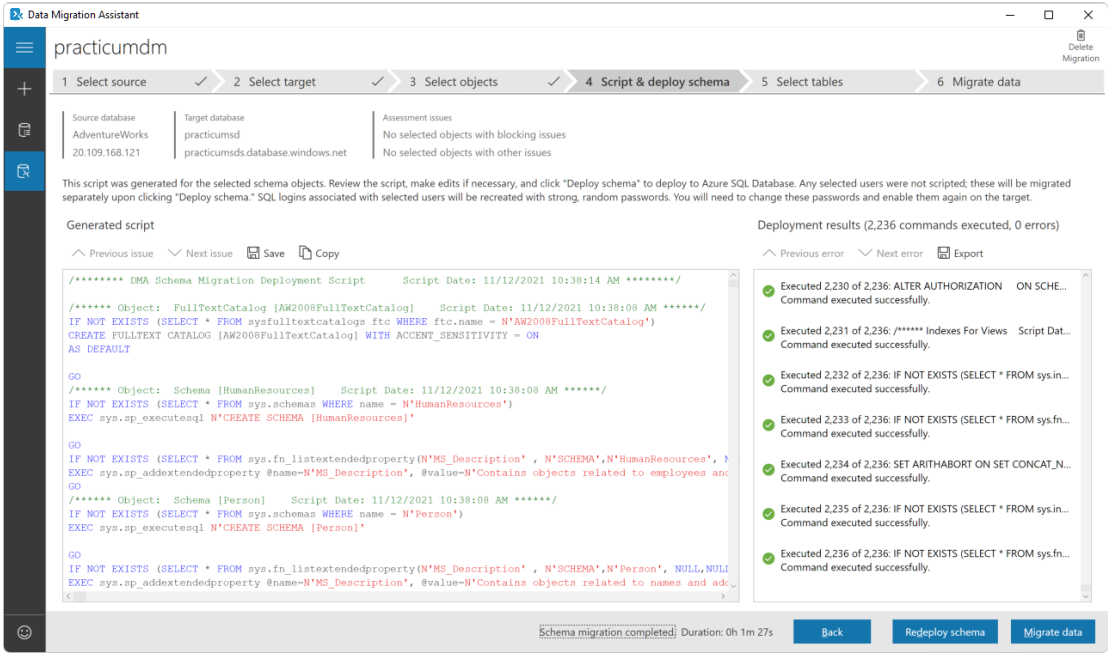
On the "**2. Select target**" page, enter values for the following items:

Server Name	Enter a meaningful name aligned with standards
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.



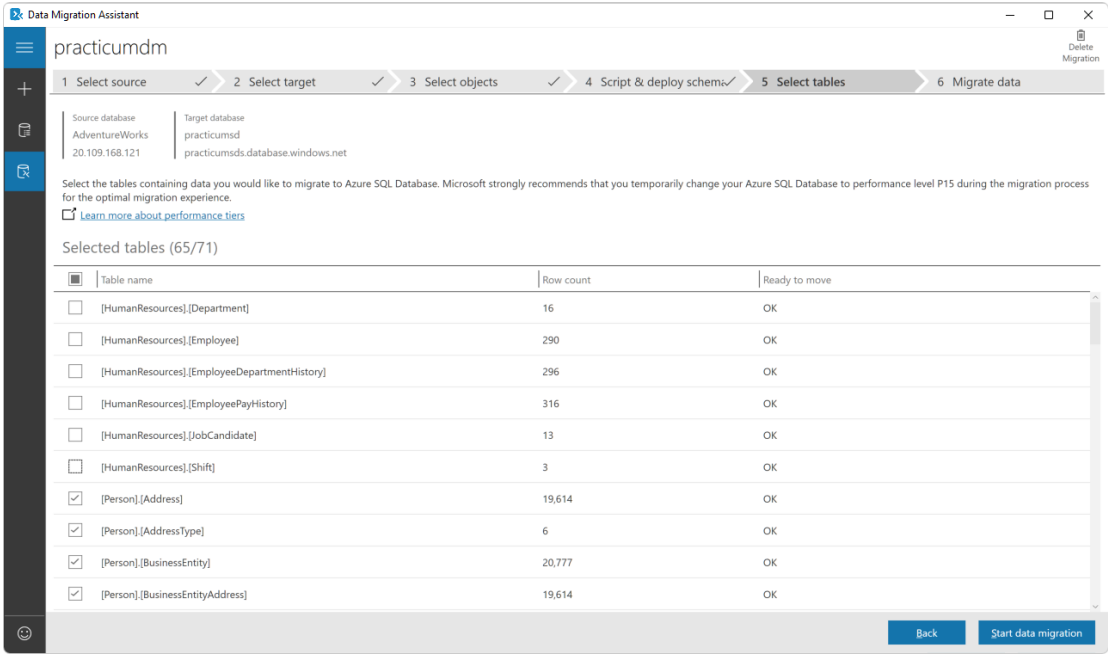
On the "**3. Select objects**" page, review schema objects, taking special note of any flagged items. Click the "**Generate SQL script**" button in the bottom-right when you are ready to continue.



On the "**4. Script & deploy schema**" page, review the generated script.

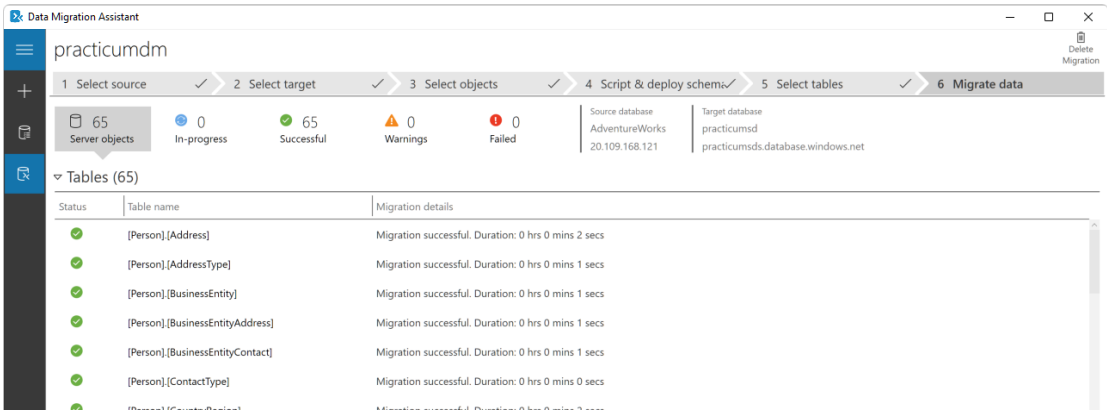
Click the “**Deploy schema**” button when you are ready to execute the script and create schema on the target database.

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



On the “**5. Select tables**” page, 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).

Objective Complete… congratulations!

External Data

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

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

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

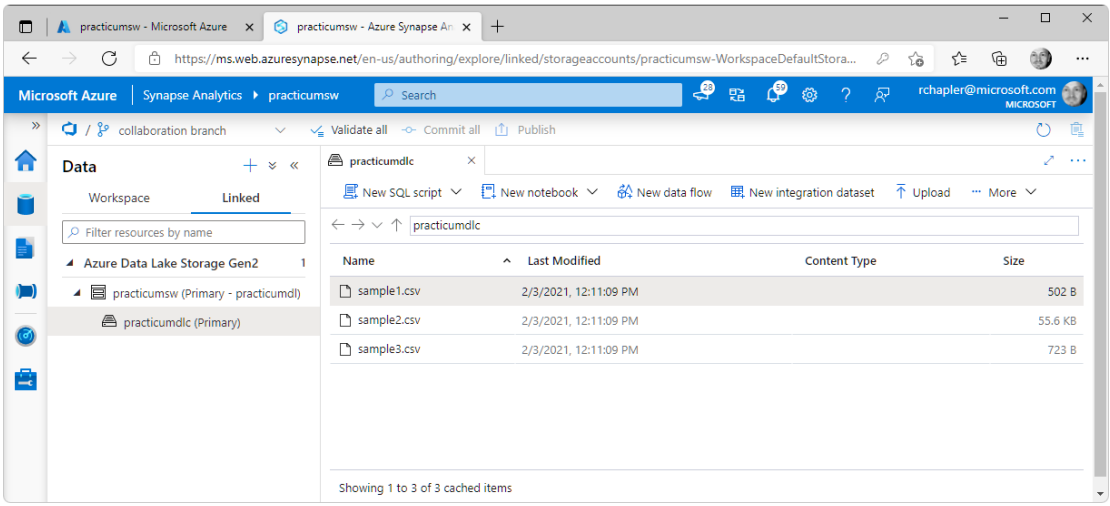
Step 2: Instantiate Prerequisites

This exercise requires the following resource(s):

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

Step 3: Explore Alternate Methods

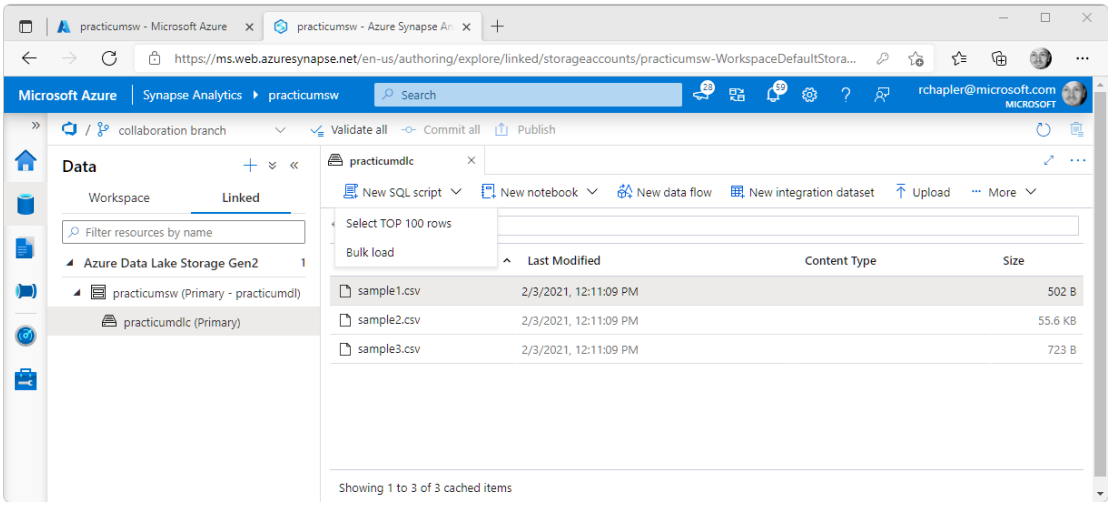
Navigate to your Synapse Analytics workspace, click the **Open** link on the “**Open Synapse Studio**” rectangle and then click the **Data** icon in the navigation pane.



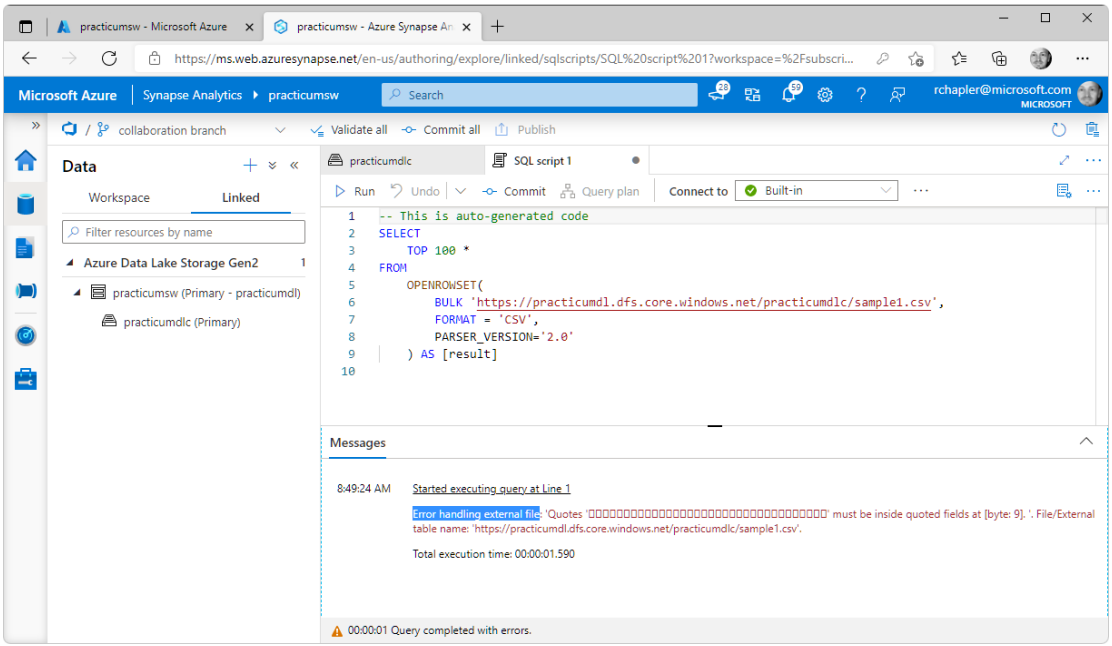
Click the **Linked** tab and Expand navigation pane.

You will see that the Azure Data Lake Storage Gen2 specified during Synapse instantiation {i.e., shareddlc} is surfaced with no additional configuration.

Method 1: SQL Script (OPENROWSET)



Click the "**New SQL script**" button and "**SELECT TOP 100 rows**" from the resulting dropdown.



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.

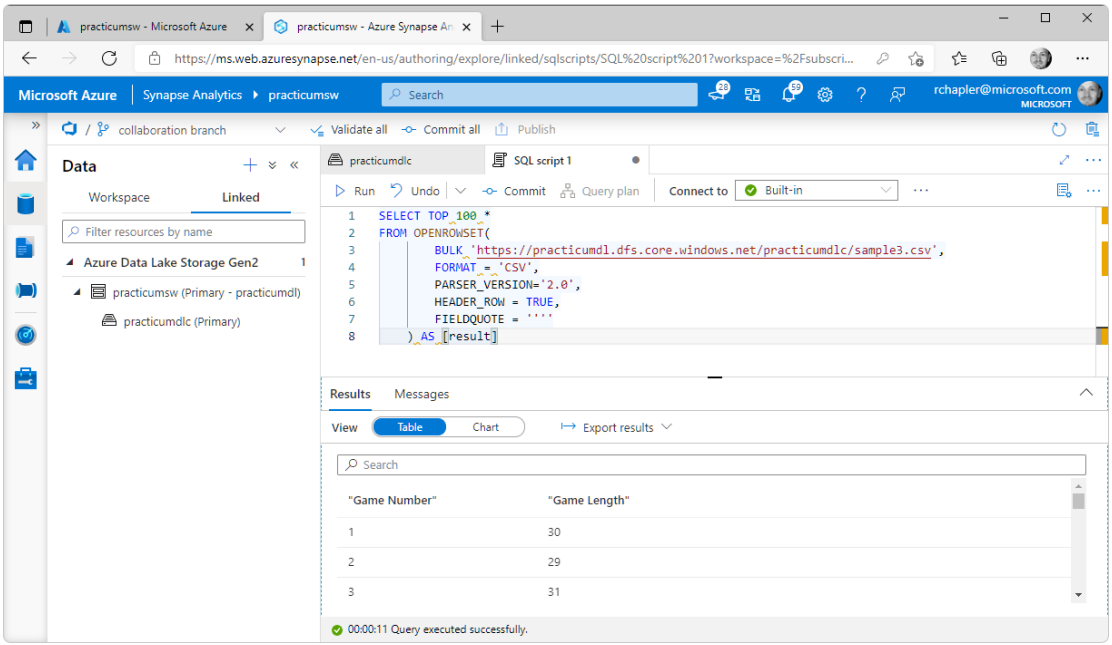
When we run the code (assuming you are using the same sample files, of course), we get an error about quote-handling.

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://shareddl.dfs.core.windows.net/shareddlc/sample3.csv',
    FORMAT = 'CSV',
    PARSER_VERSION='2.0',
    HEADER_ROW = TRUE,
    FIELDQUOTE = ''''
  ) AS [result]
```

… and then, re-Run…

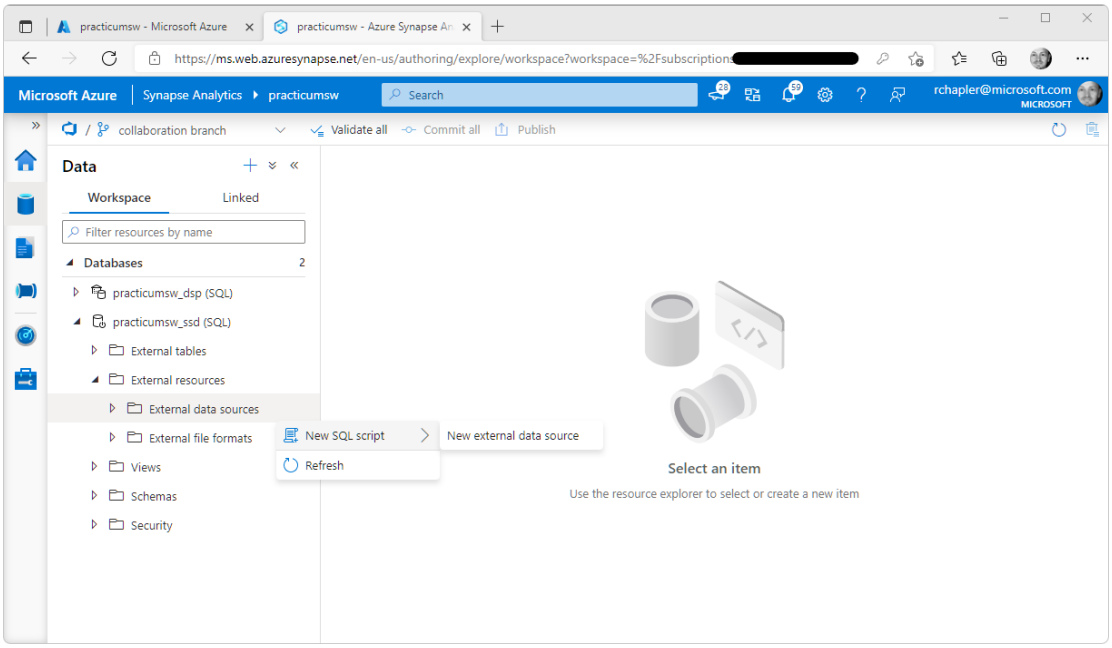


Method 2: External Table (Serverless SQL Database)

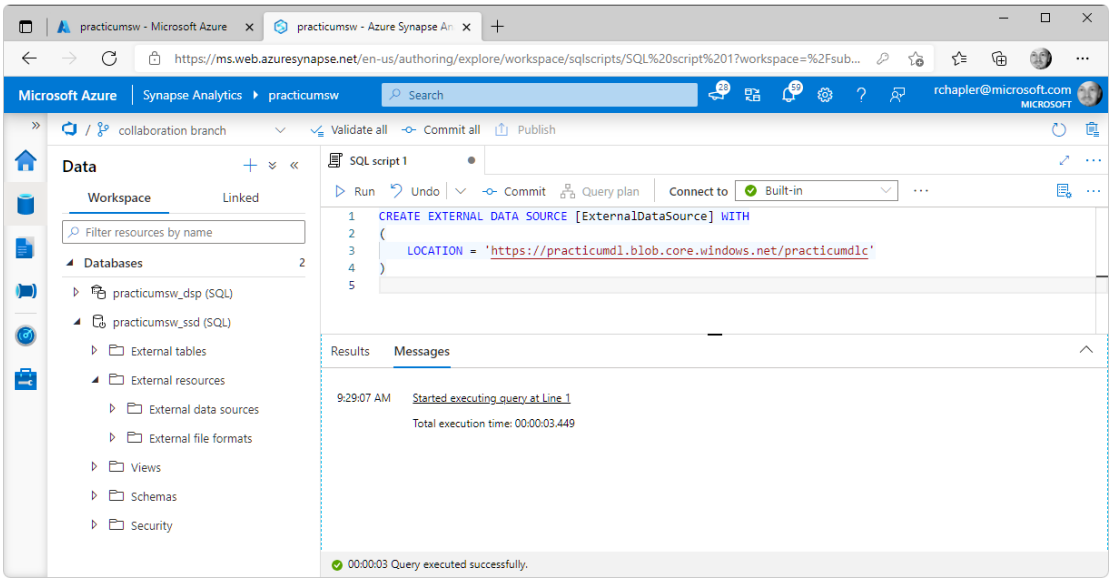
Click the **Workspace** tab.

External Data Source

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



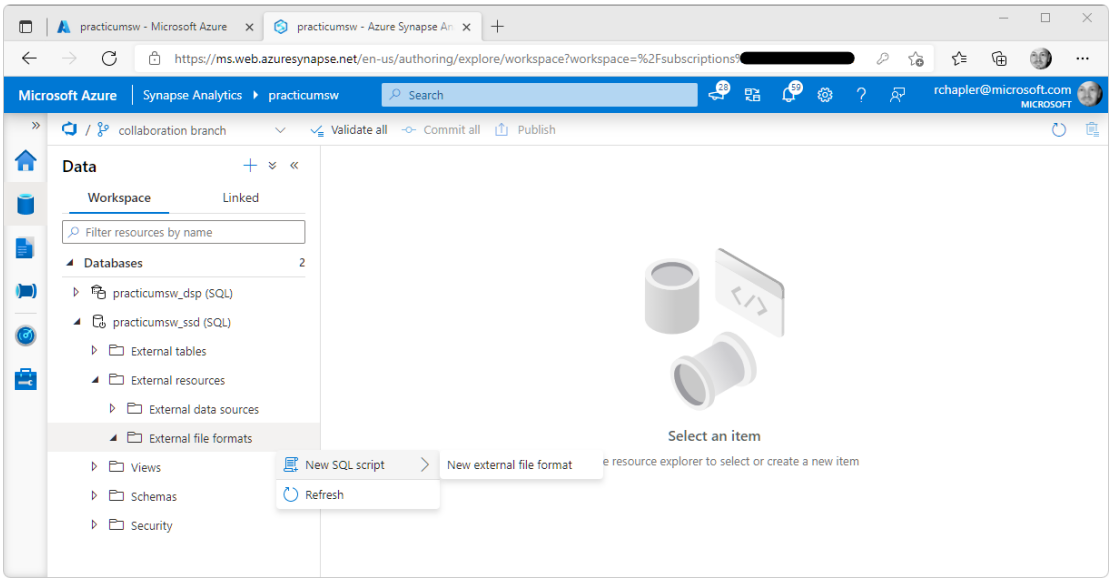
Click “**New SQL script**”, and then “**New external data source**” in the dropdown.



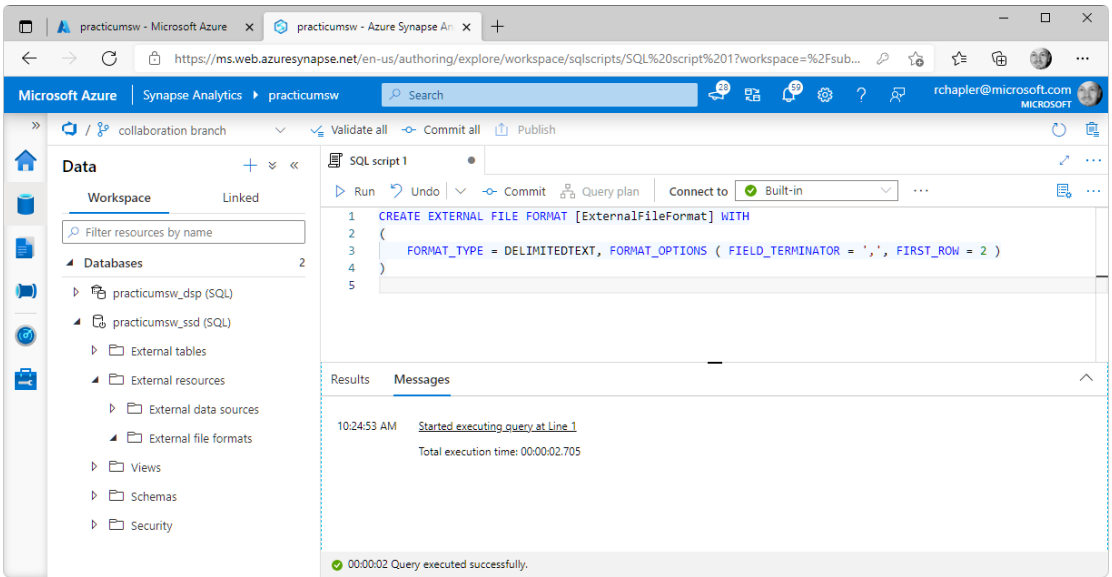
Replace <**STORAGEACCOUNT**> and <**CONTAINER**> in the auto-generated code and then **Run**.

External File Format

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



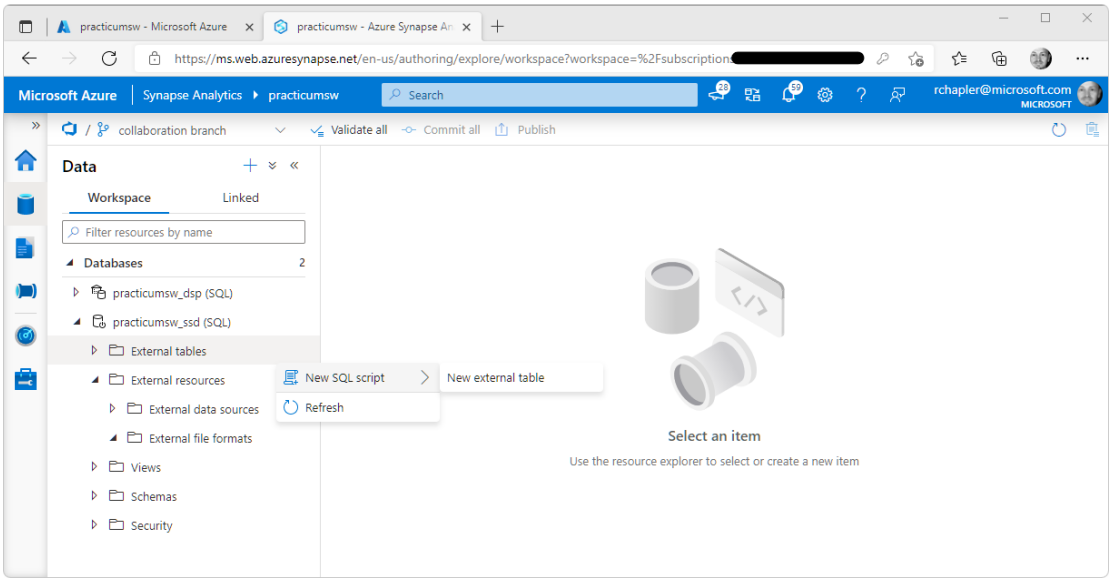
Click “**New SQL script**”, and then “**New external file format**” in the dropdown.



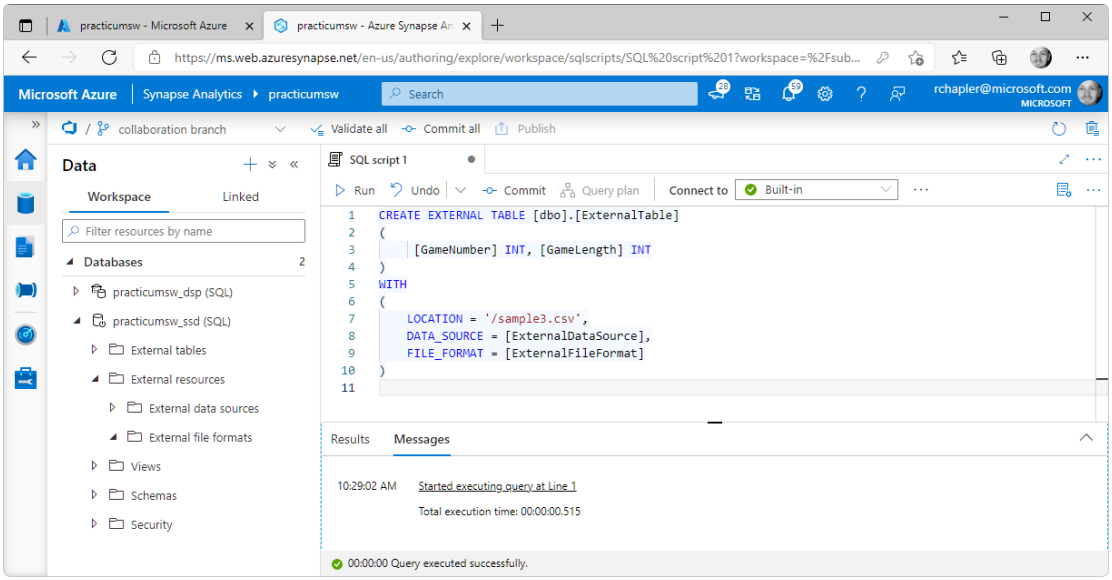
Append FORMAT_OPTIONS arguments FIELD_TERMINATOR and FIRST_ROW to the default FORMAT_TYPE argument and then **Run**.

External Table

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



Click “**New SQL script**”, and then “**New external table**” in the dropdown.

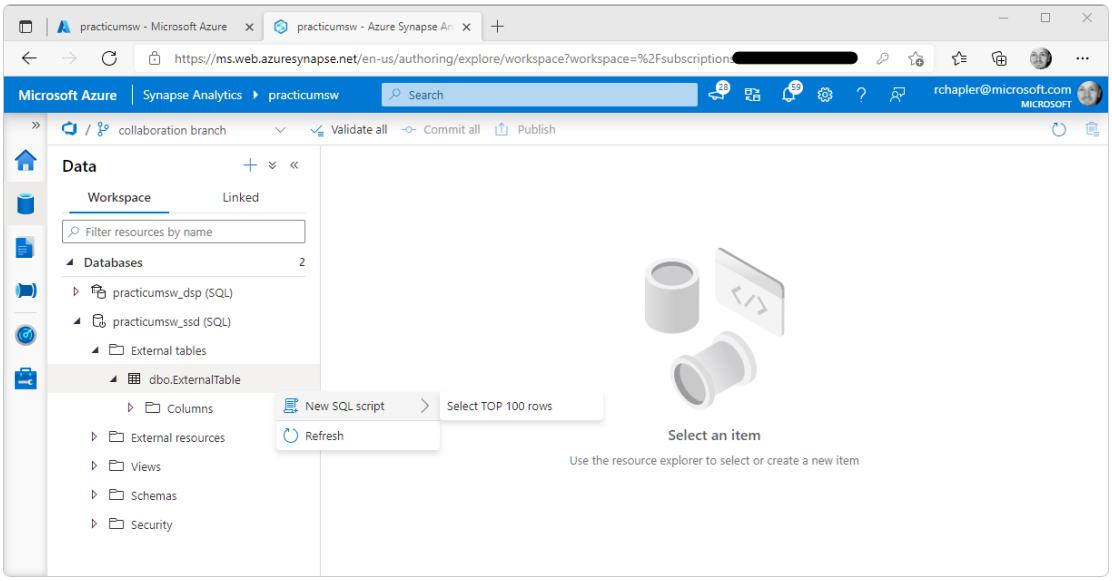


Update the following items in the default code and then **Run**.

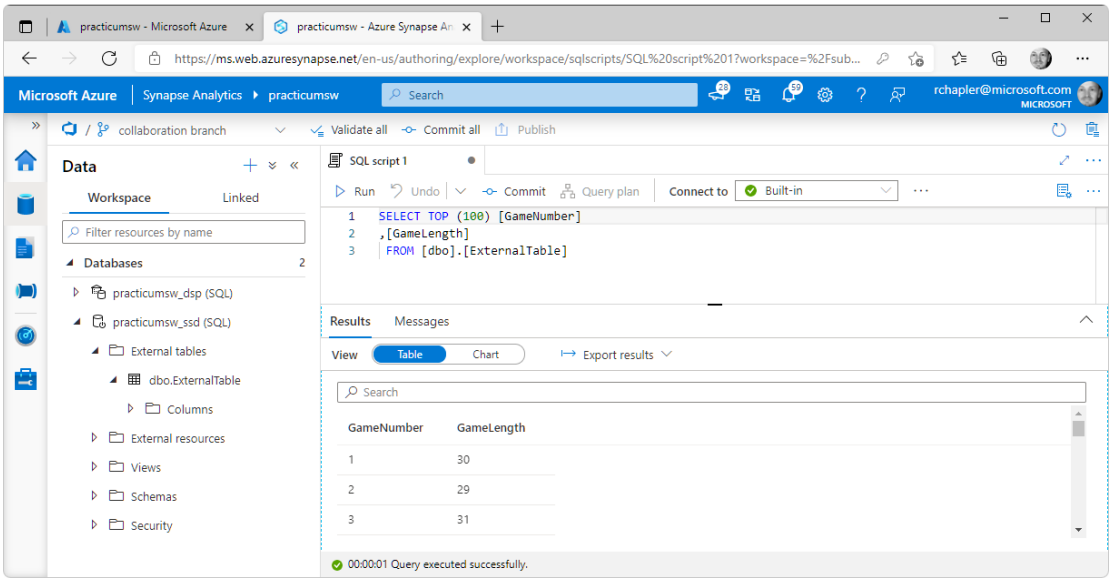
- **Schema Definition** … replace `[Id] INT` with columns matching the external data source
- **LOCATION** … replace `‘/folder/file’` with values matching those in your container
- **DATA_SOURCE** … replace `[DataSource1]` with the name used in Create External Data Source
- **FILE_FORMAT** … replace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



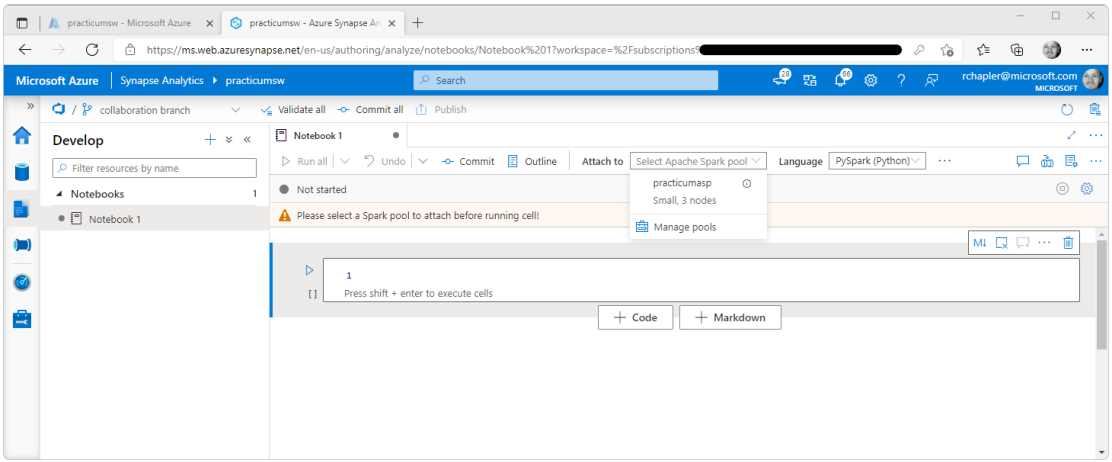
Click **Run**.

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.

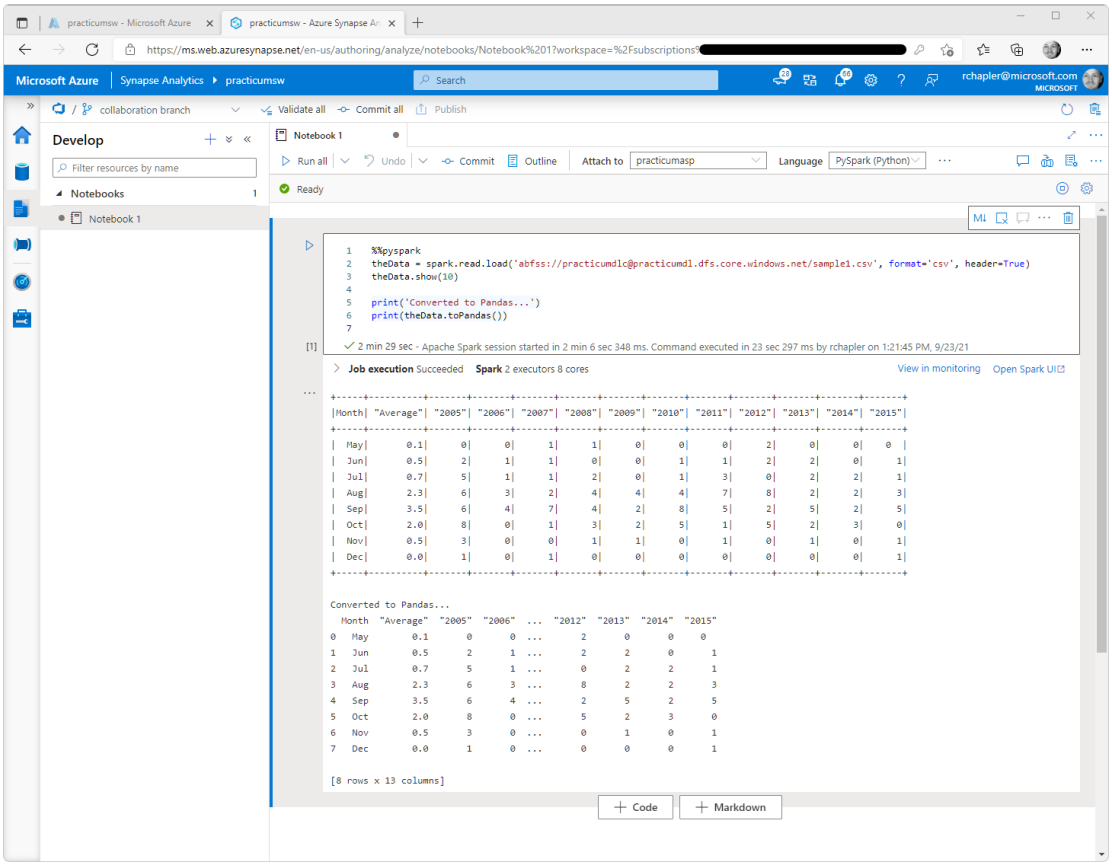
Select **Notebook** from the resulting dropdown.



Select your Apache Spark Pool from the "**Attach to**" dropdown.

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

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

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



Objective Complete... congratulations!

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.

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

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

Step 2: Instantiate Prerequisites

This exercise requires the following resource(s):

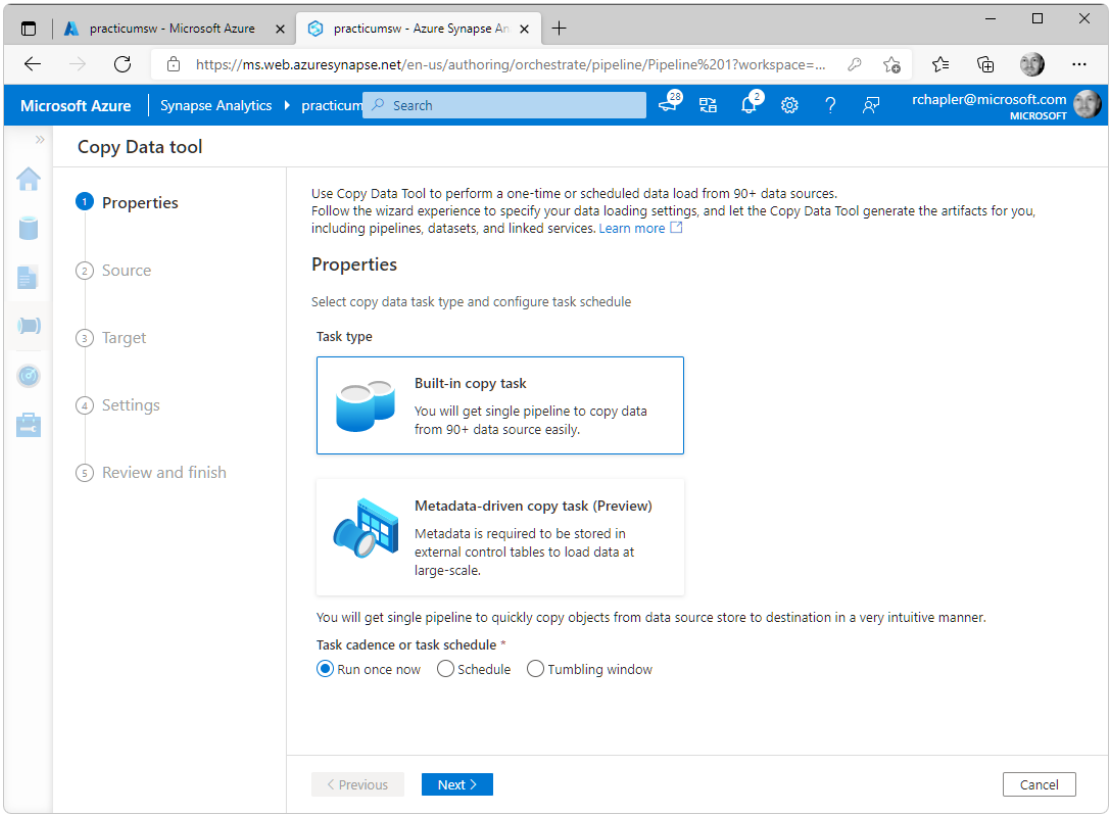
- Data Lake (with container and sample data)
- Synapse (with Dedicated SQL Pool)

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

Step 3: Use "Copy Data tool"

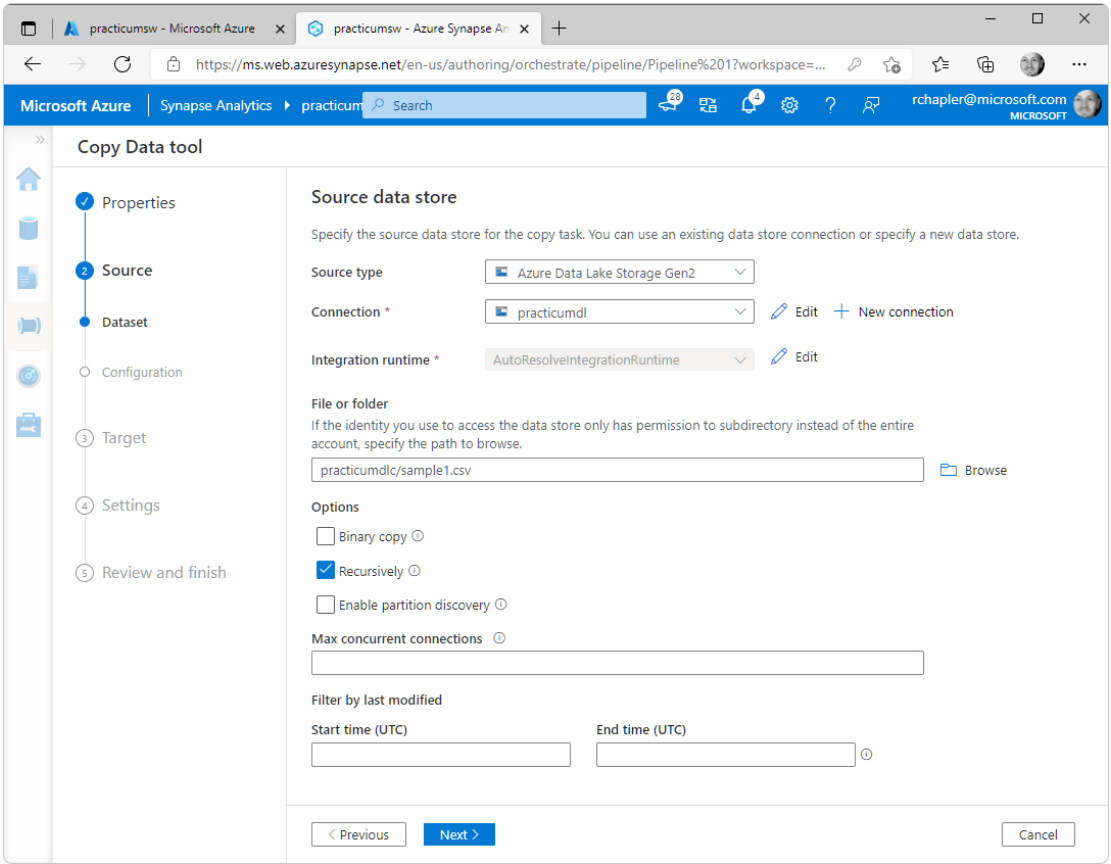
Navigate to your Synapse Analytics workspace, click the **Open** link on the "**Open Synapse Studio**" rectangle and then click the **Ingest** button.

Properties



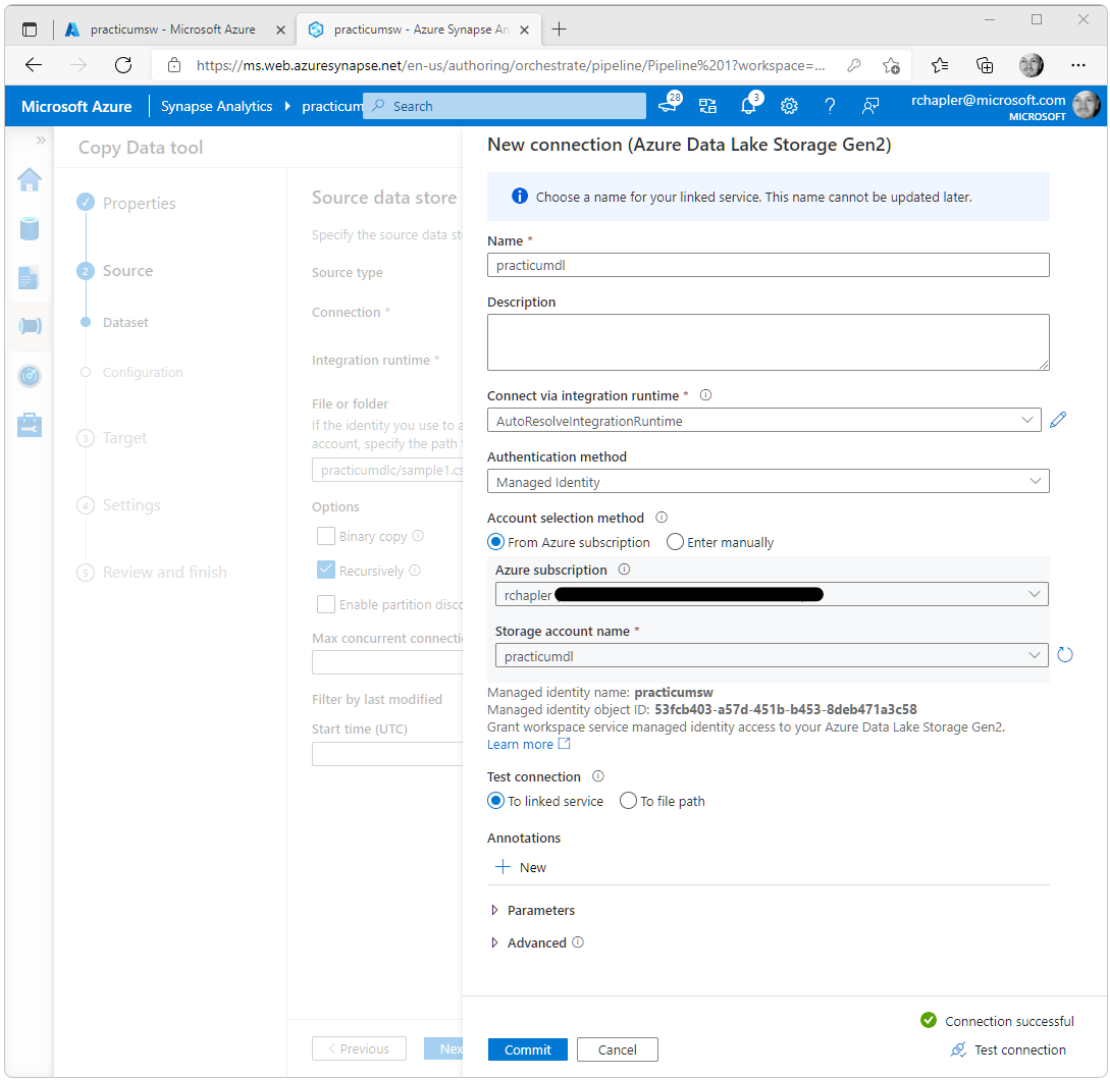
Confirm default select, “Built-in copy task” and “Run once now”, then click the “**Next >**” button.

Source > Dataset



On the "**Source data store**" page, enter values for the following items:

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



On the "**New connection (Azure Data Lake Storage Gen2)**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Connect via…	Confirm default selection, AutoResolveIntegrationRuntime
Account Selection Method	Confirm default selection, "**From Azure subscription**"
Azure Subscription	Select your subscription
Storage Account Name	Select Data Lake

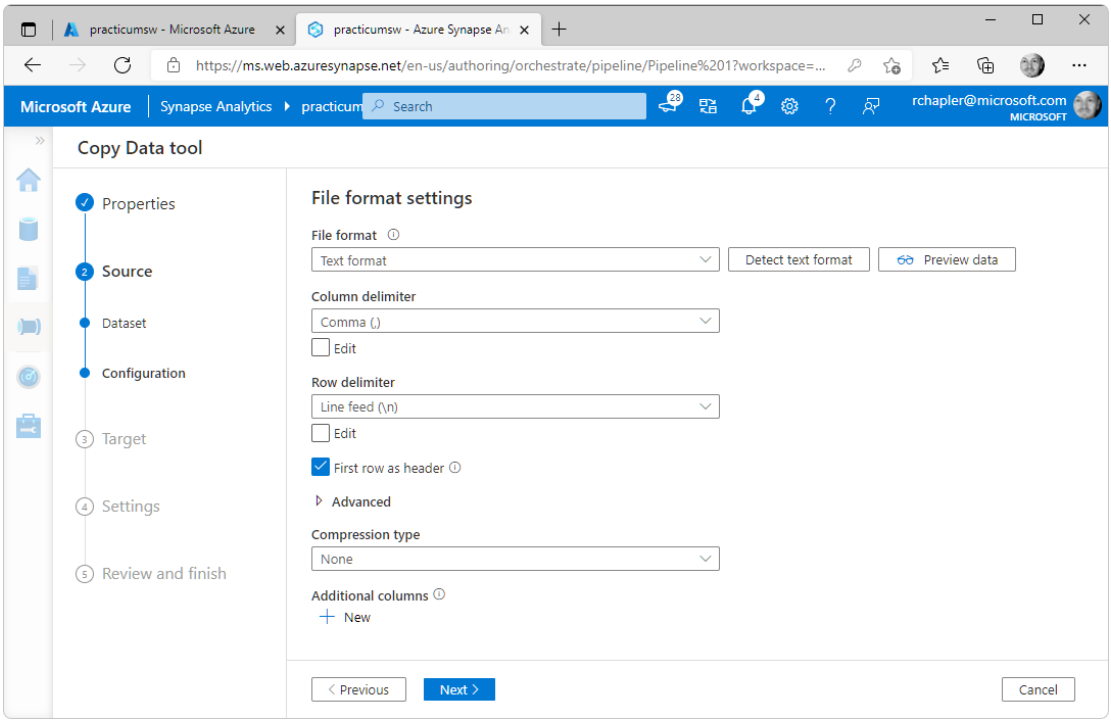
Click “**Test connection**” to confirm successful connection and then click the **Create** (or **Commit**) button.

Back on the “**Source data store**” page, click **Browse** to select container/file {e.g., “shareddlc/sample1.csv”}.

No additional changes are required. Review settings on the remaining tabs.

Click the “**Next >**” button.

Source > Configuration

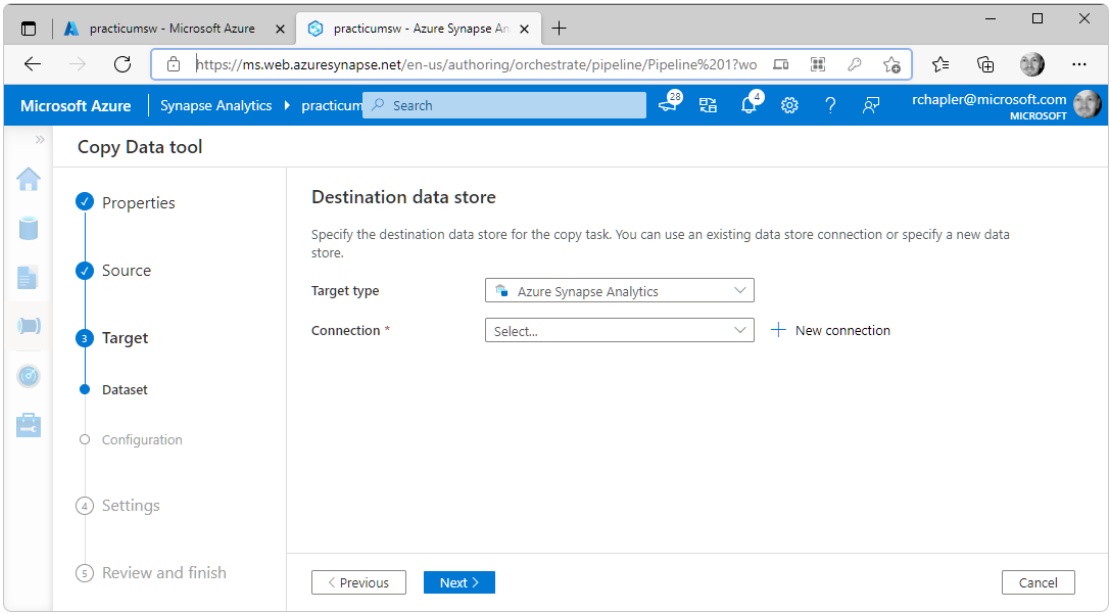


On the “**File format settings**” page, confirm default configuration and the resulting data preview.

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

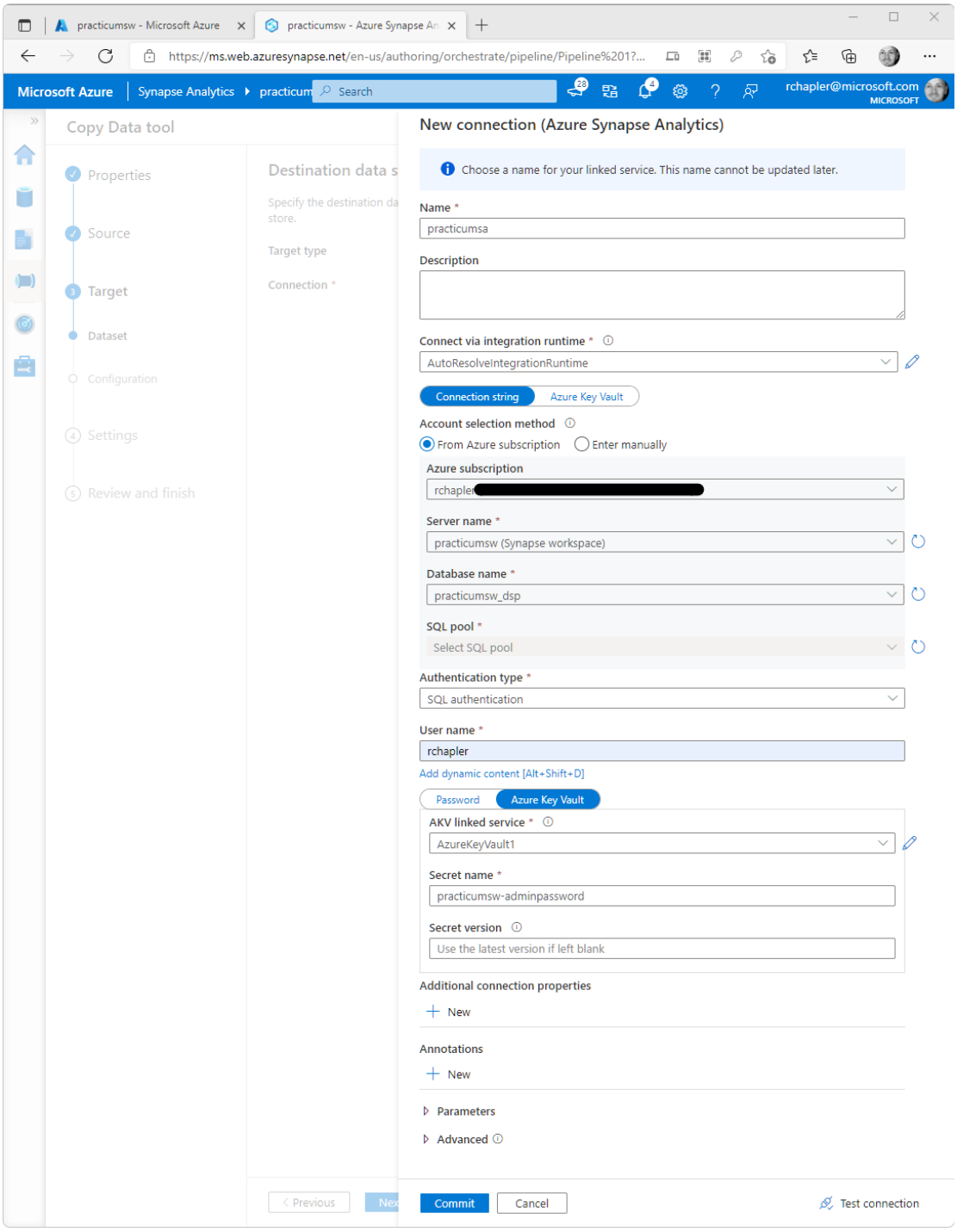
Click the “**Next >**” button.

Target > Dataset



On the "**Destination data store**" page, enter values for the following items:

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

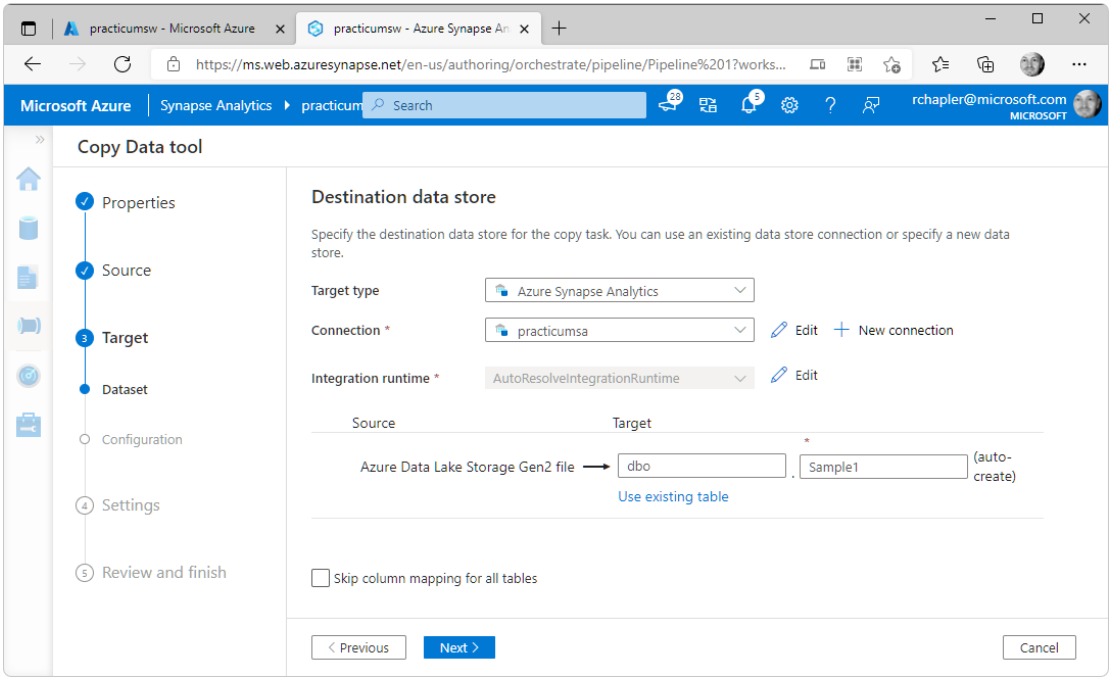


On the "**New connection (Azure Synapse Analytics)**" popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Connect via…	Confirm default selection, AutoResolveIntegrationRuntime

Account Selection Method	Confirm default selection, “**From Azure subscription**”
Azure Subscription	Select your 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

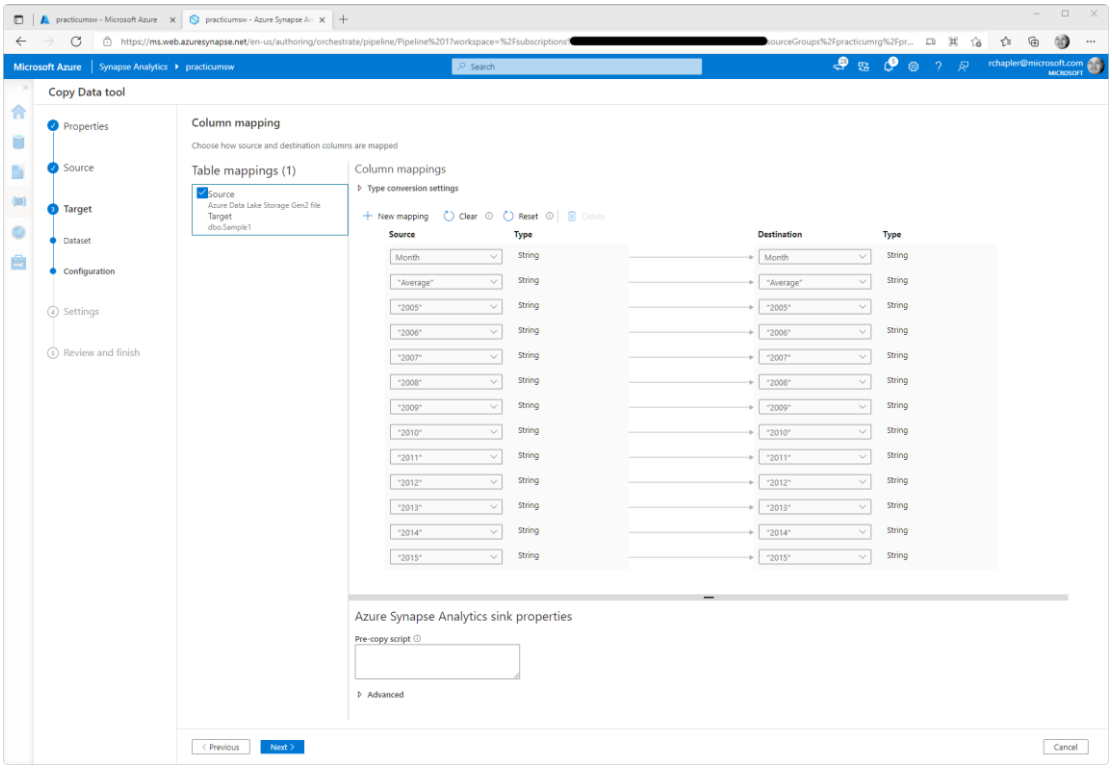
Click “**Test connection**” to confirm successful connection and then click the **Create** (or **Commit**) button.



Back on the “**Destination data source**” page, enter values for target schema name {e.g., “dbo”} and target table name {e.g., “Sample1”}.

Click the “**Next >**” button.

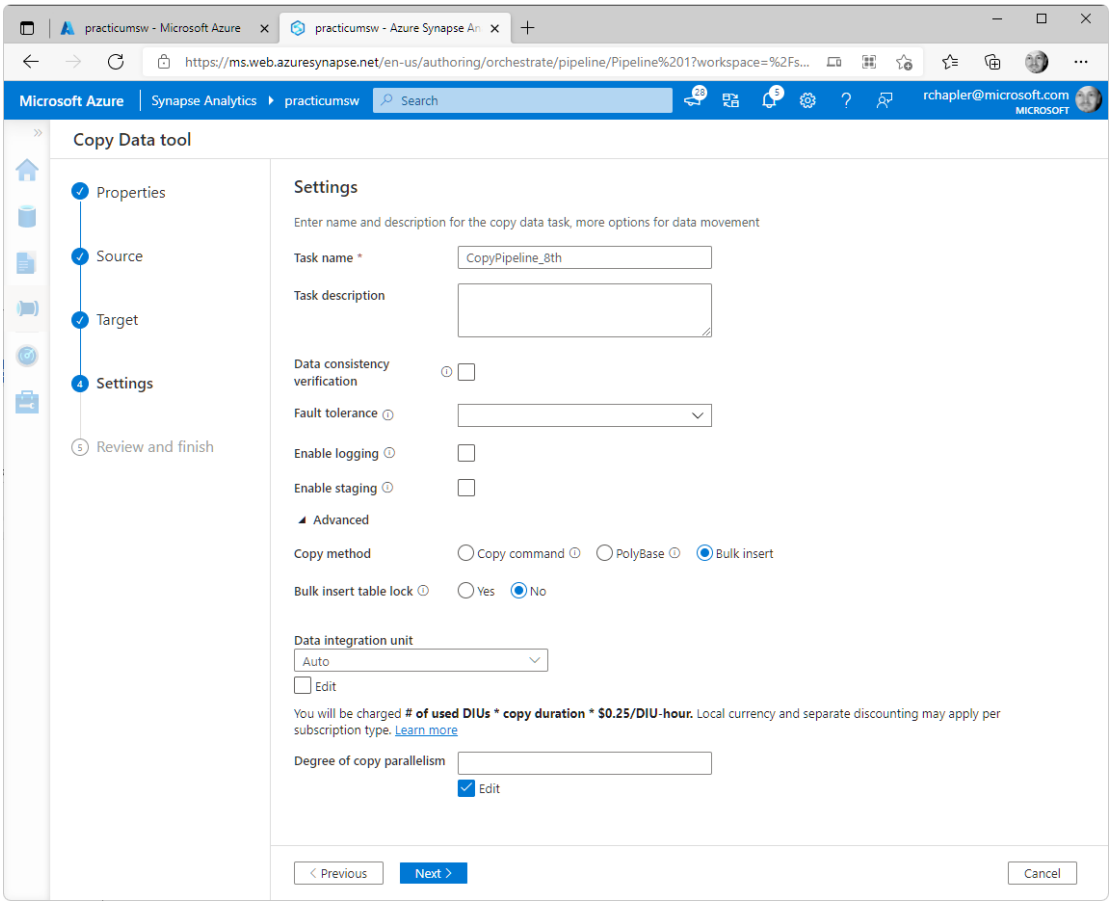
Target > Configuration



Review default column mappings; correct if required.

Click the “**Next >**” button.

Settings

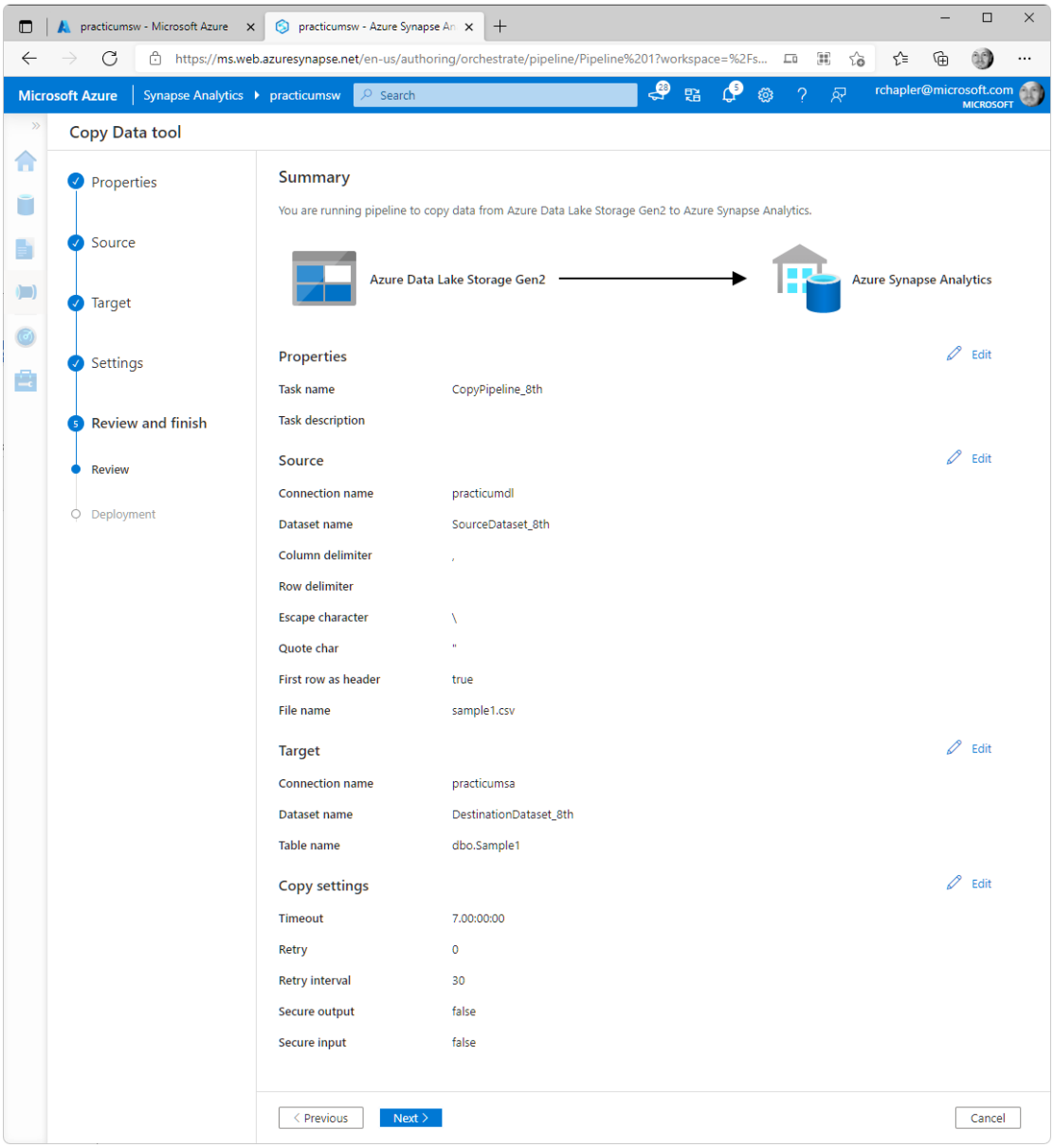


On the **Settings** page, enter values for the following items:

Enable Staging	Unchecked
Copy Method	Bulk insert

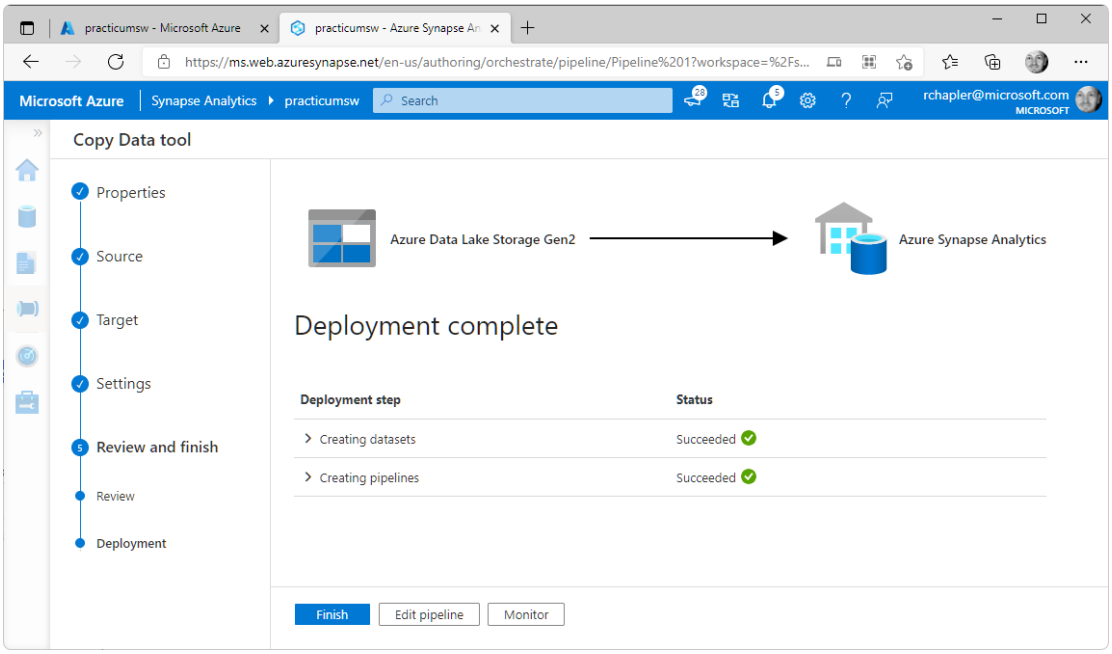
Click the "**Next >**" button.

Summary



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

Deployment

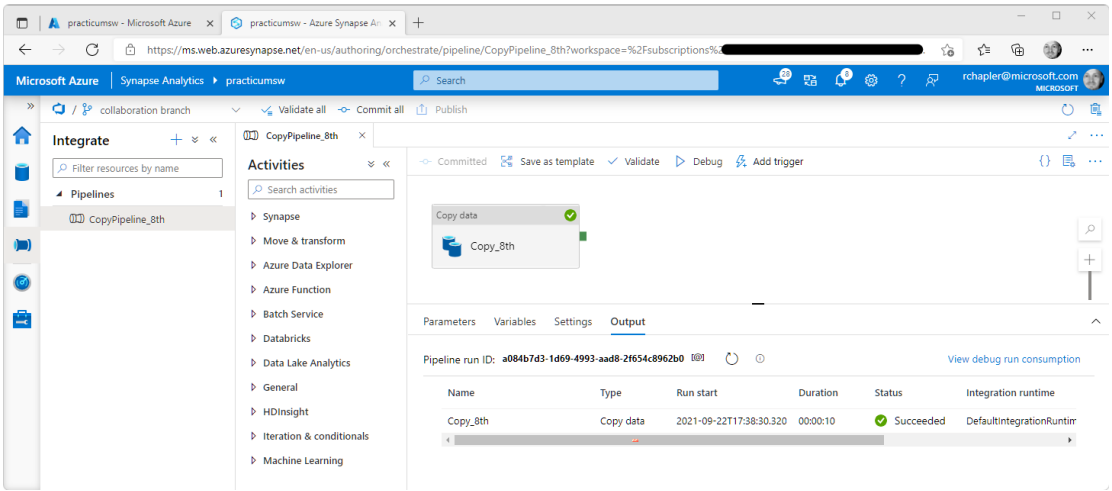


Monitor progress.

Click the **Finish** button on success.

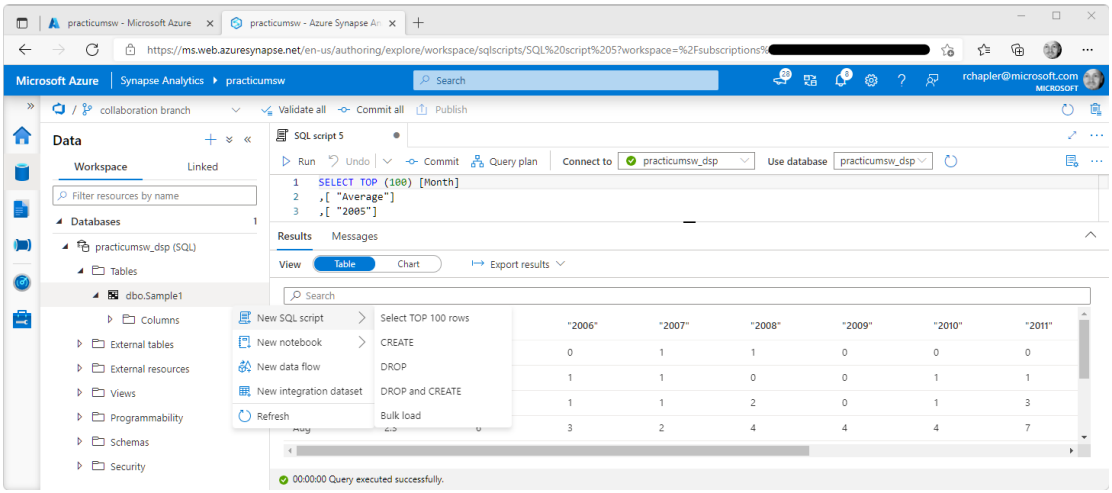
Step 3: Confirm Success

"**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 dropdown menu.

Objective Complete... congratulations!

Pipeline

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

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

- "We want to load structured data from SQL to Data Explorer"

Step 2: Instantiate Prerequisites

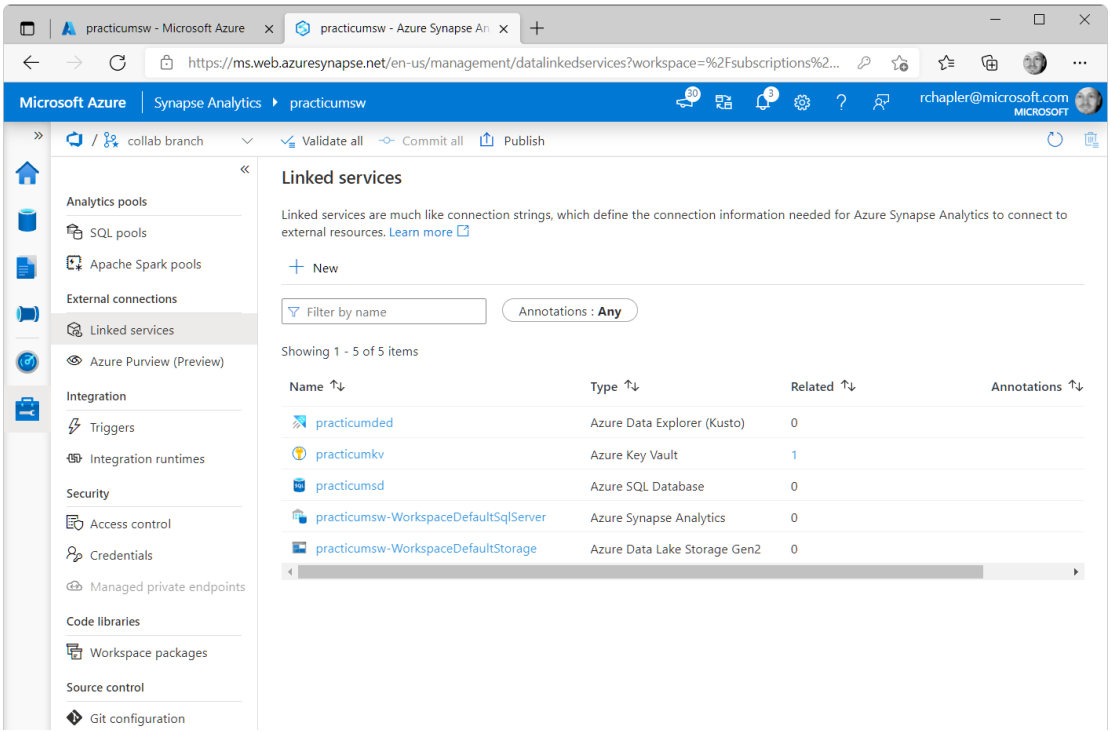
This exercise requires the following resource(s):

- 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.

Step 3: Confirm Linked Services

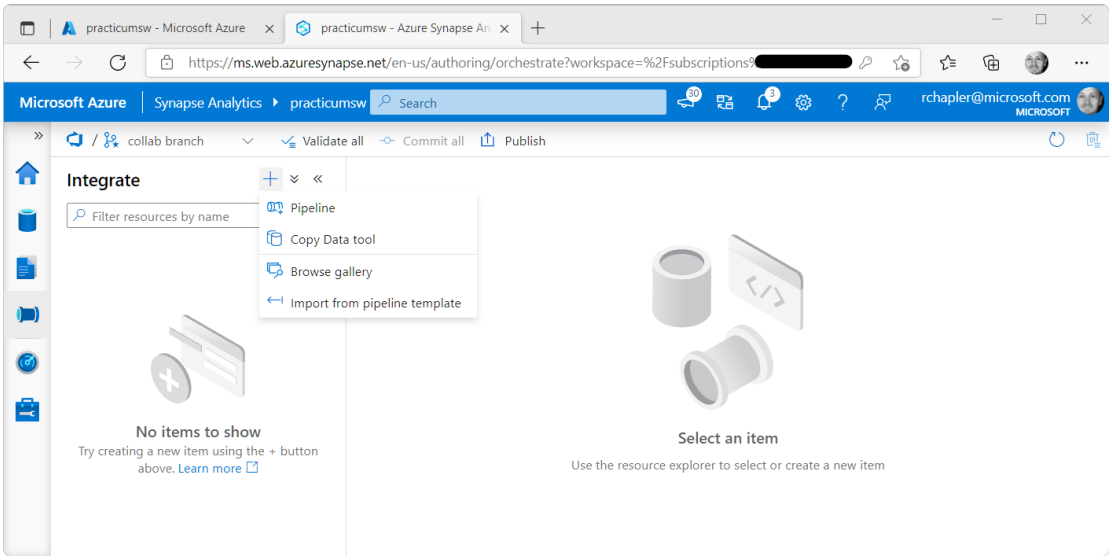
Navigate to your Synapse Analytics workspace and then click the **Open** link on the "**Open Synapse Studio**" rectangle. Select the **Manage** icon on the navigation and then "**Linked Services**" in the resulting navigation menu. Confirm linked service instantiation for: 1) Key Vault, 2) SQL and 3)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



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

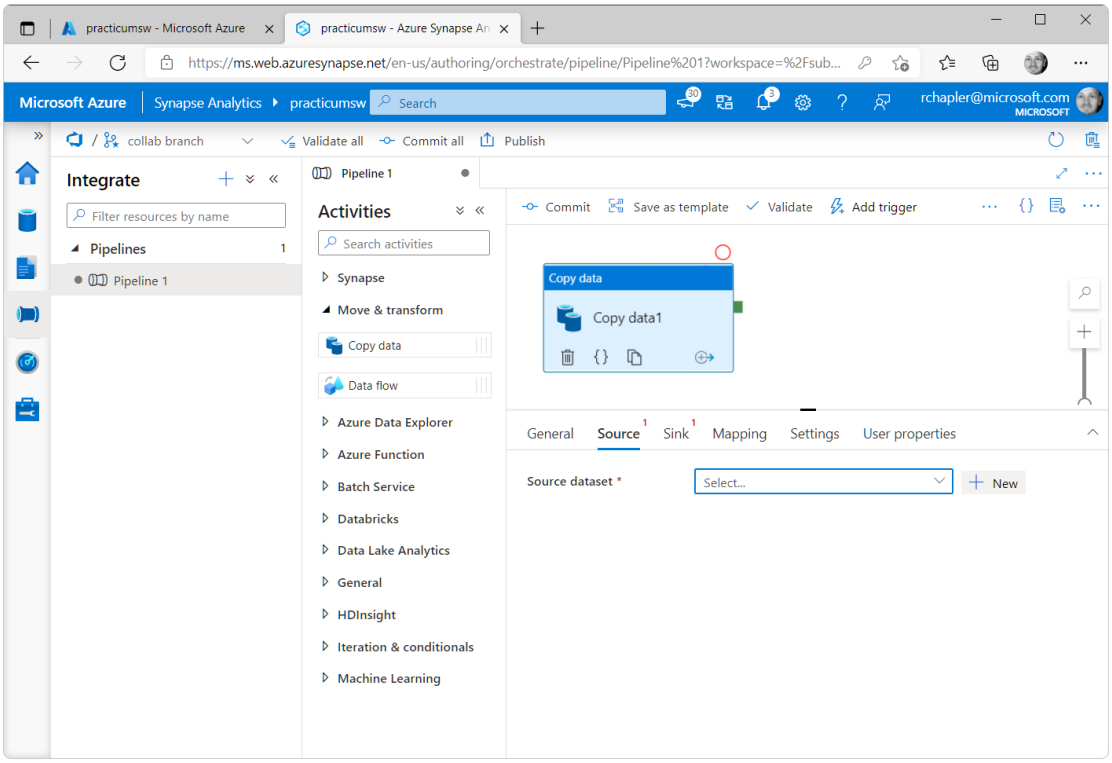
Step 4: Create Pipeline

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



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

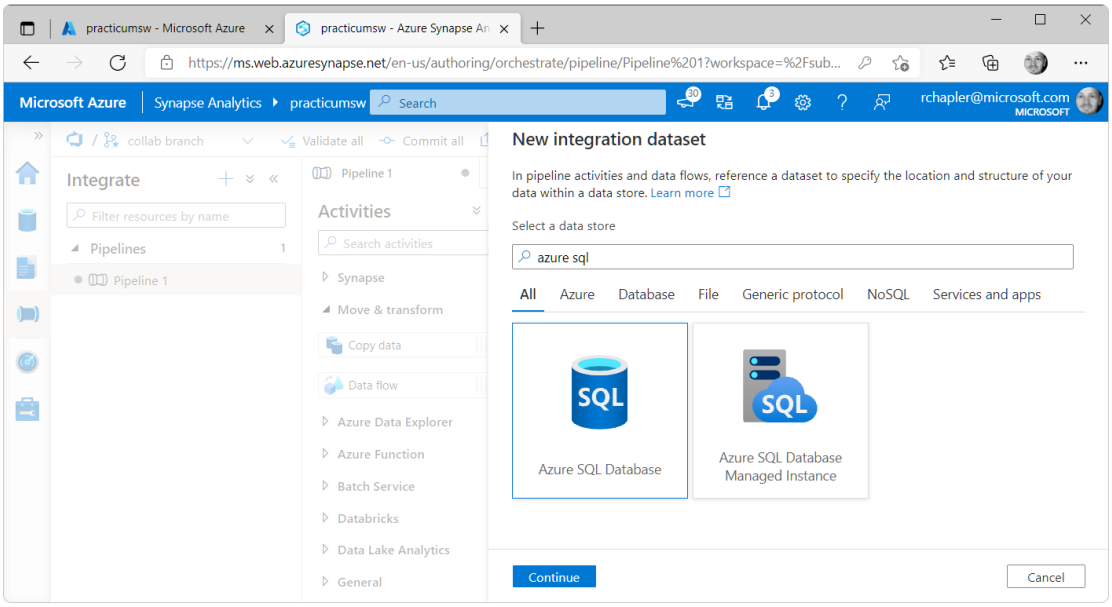
Select **Pipeline** from the resulting dropdown.



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

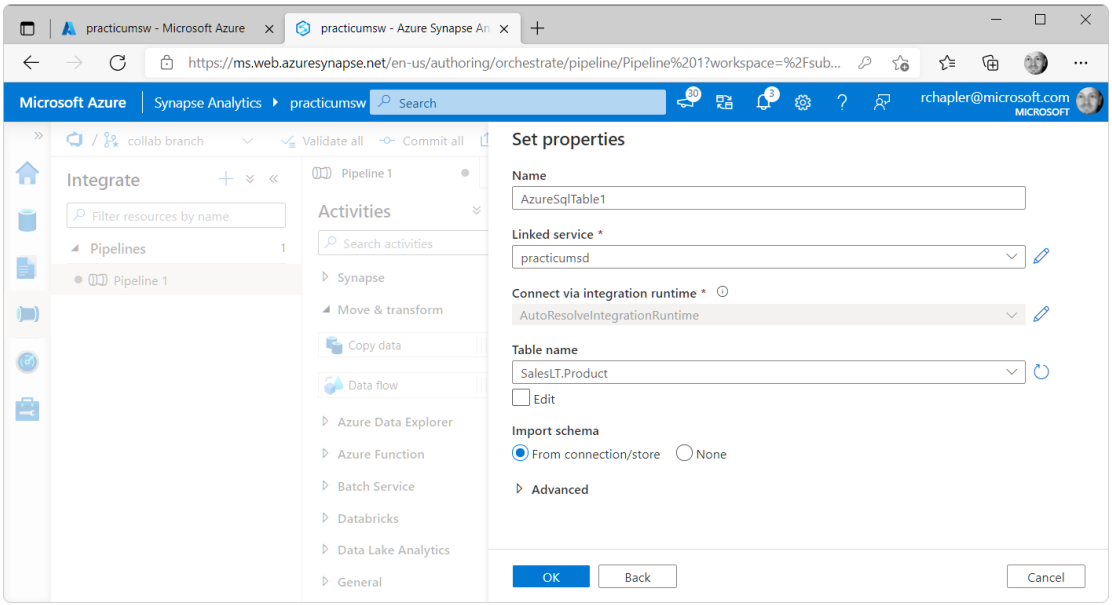
Source

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



On the “**New integration dataset**” popout, search for and then select “**Azure SQL Database**”.

Click the **Continue** button.

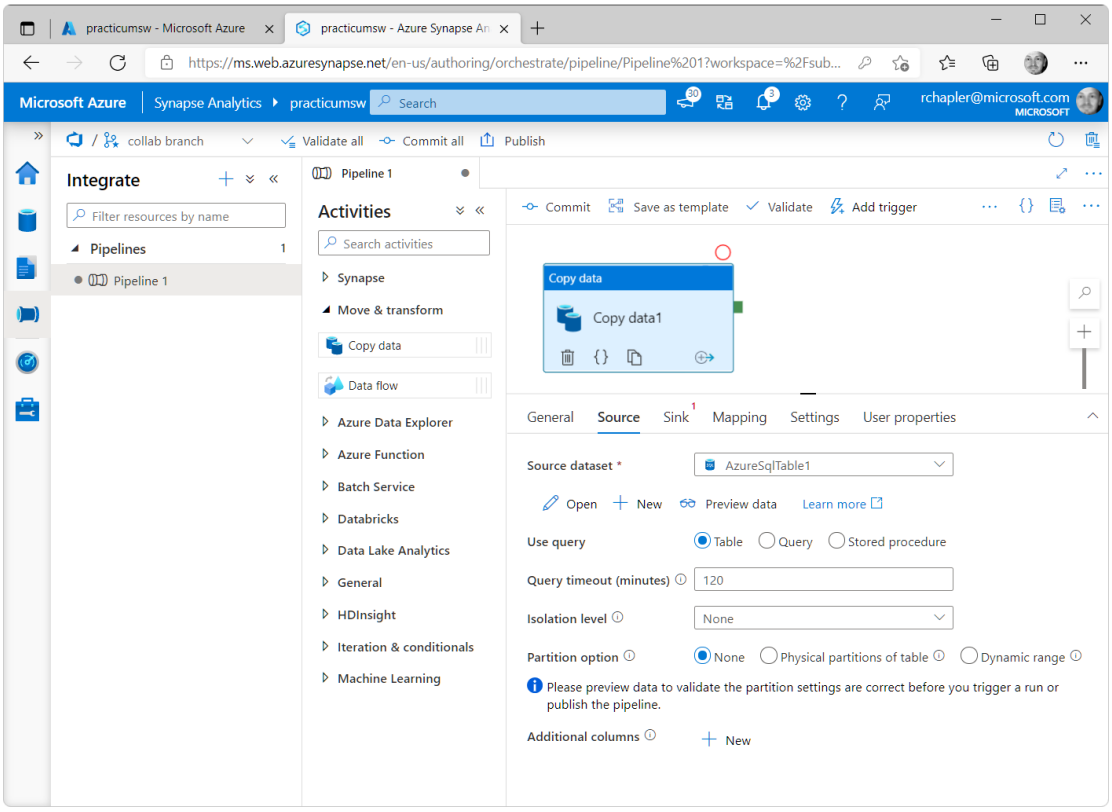


On the “**Set properties**” popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Linked Service	Select your Azure SQL linked service
Table Name	Select “SalesLT.Product”
Import Schema	Confirm default selection, “From connection/store”

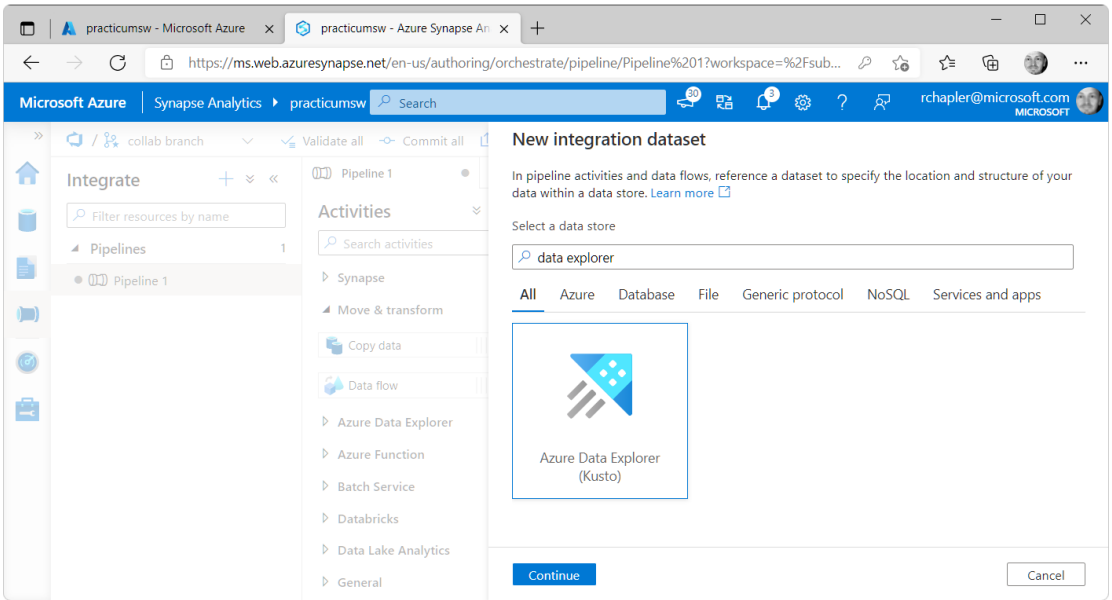
Leave all other settings with default values.

Click the **OK** button.

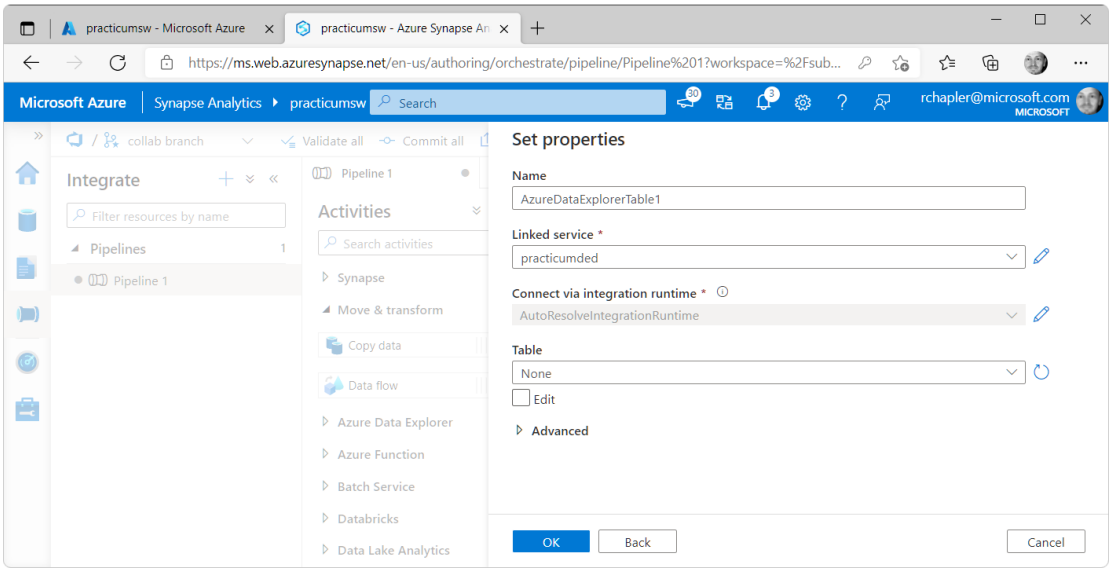


Sink

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



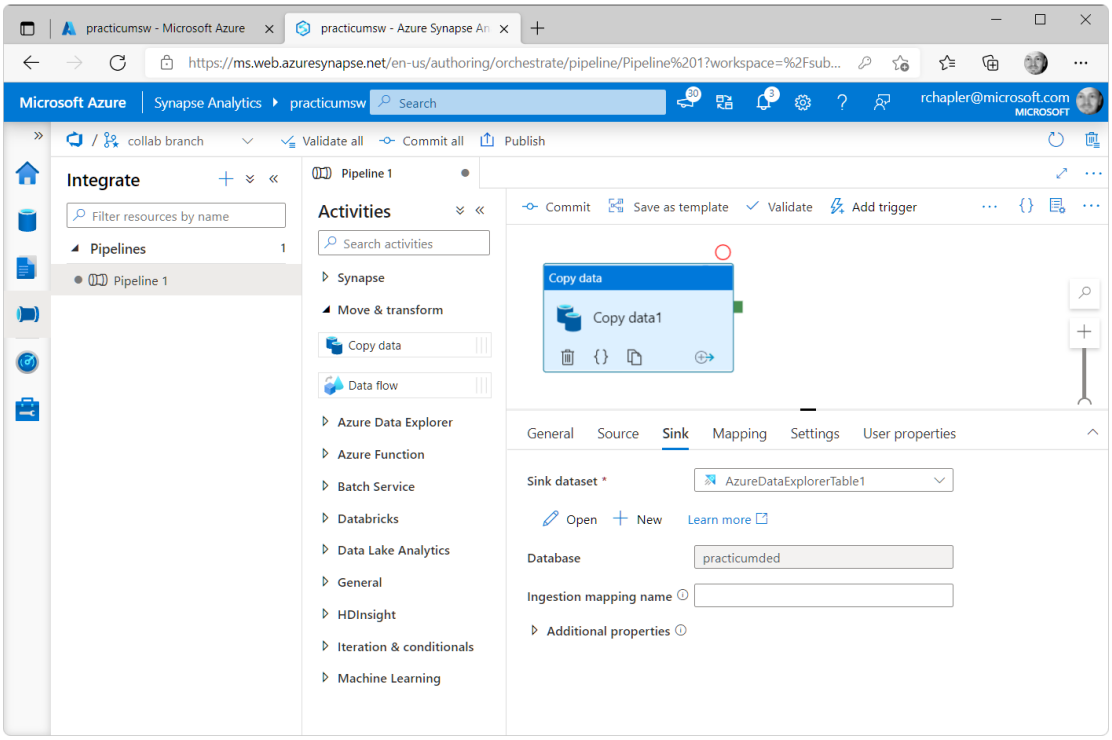
On the “New integration dataset” popout, search for and then select “Azure Data Explorer (Kusto)”.
Click the **Continue** button.



On the “**Set properties**” popout, enter values for the following items:

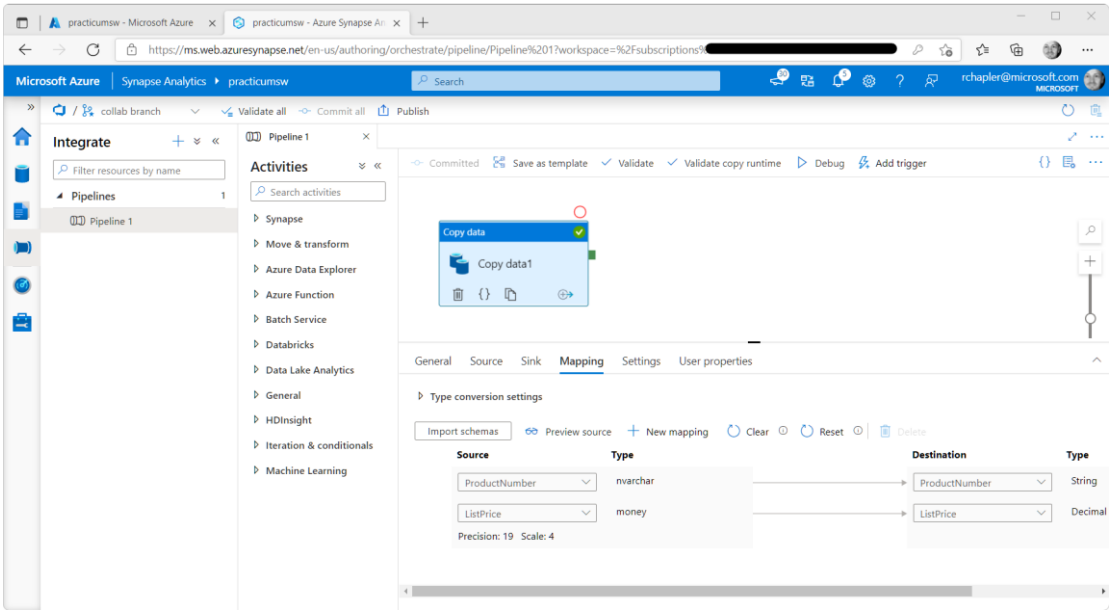
Name	Enter a meaningful name aligned with standards
Linked Service	Select your Azure Data Explorer linked service
Table Name	Select **Product**

Leave all other settings with default values.
Click the **OK** button.



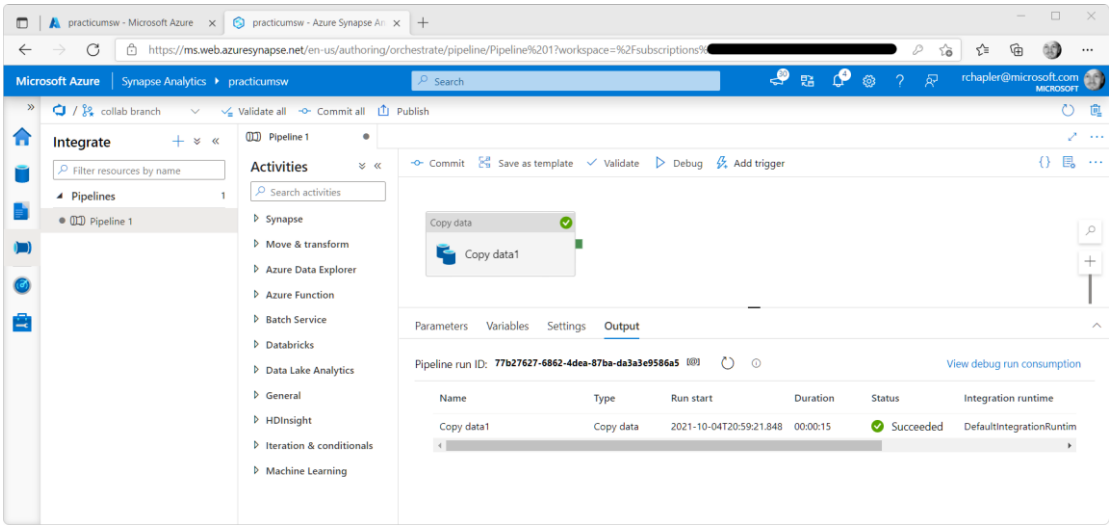
Mapping

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



Step 5: Confirm Success

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



Objective Complete... congratulations!

Conditional Activity

Follow these instructions to demonstrate conditional logic in a pipeline.

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

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

Step 2: Instantiate Prerequisites

This exercise requires the following resource(s):

- Synapse

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

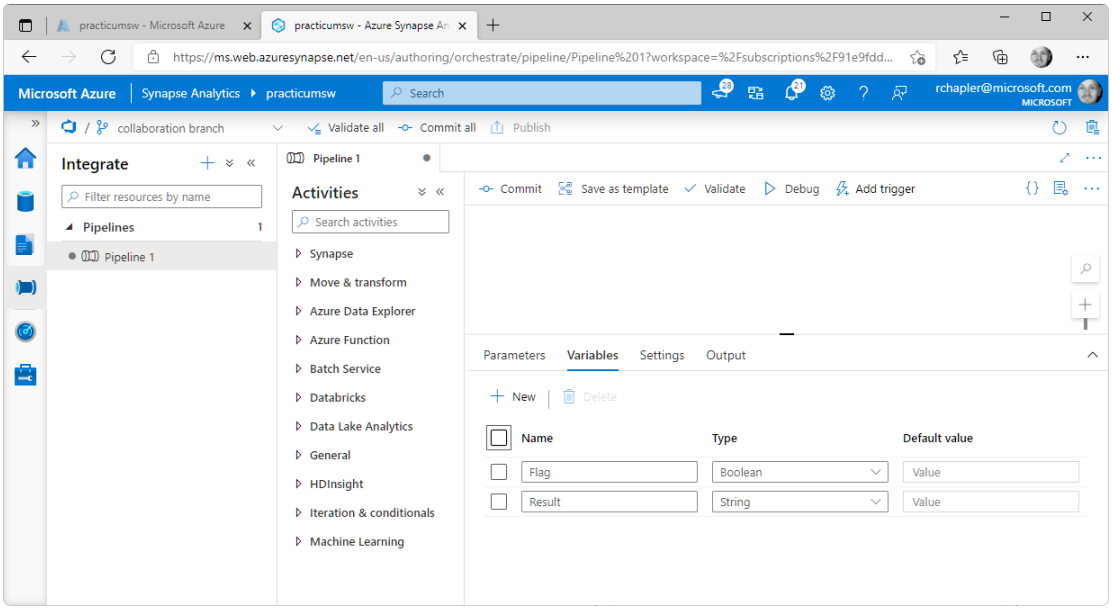
Step 3: Create Pipeline

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 dropdown.



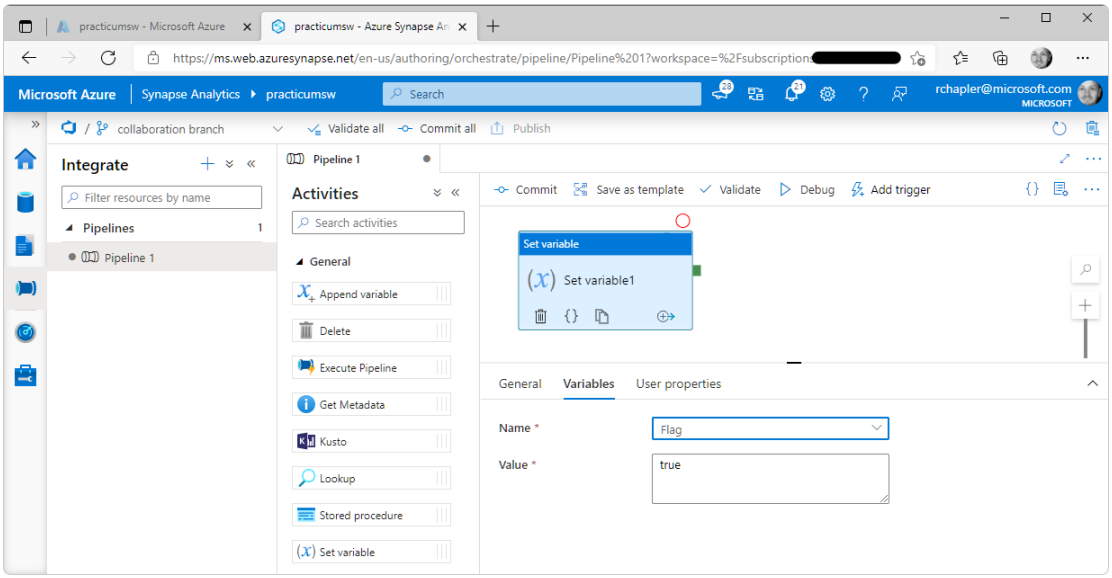
Click on the **Variables** tab.

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, and then drag-and-drop a “**Set variable**” component into the activity window.



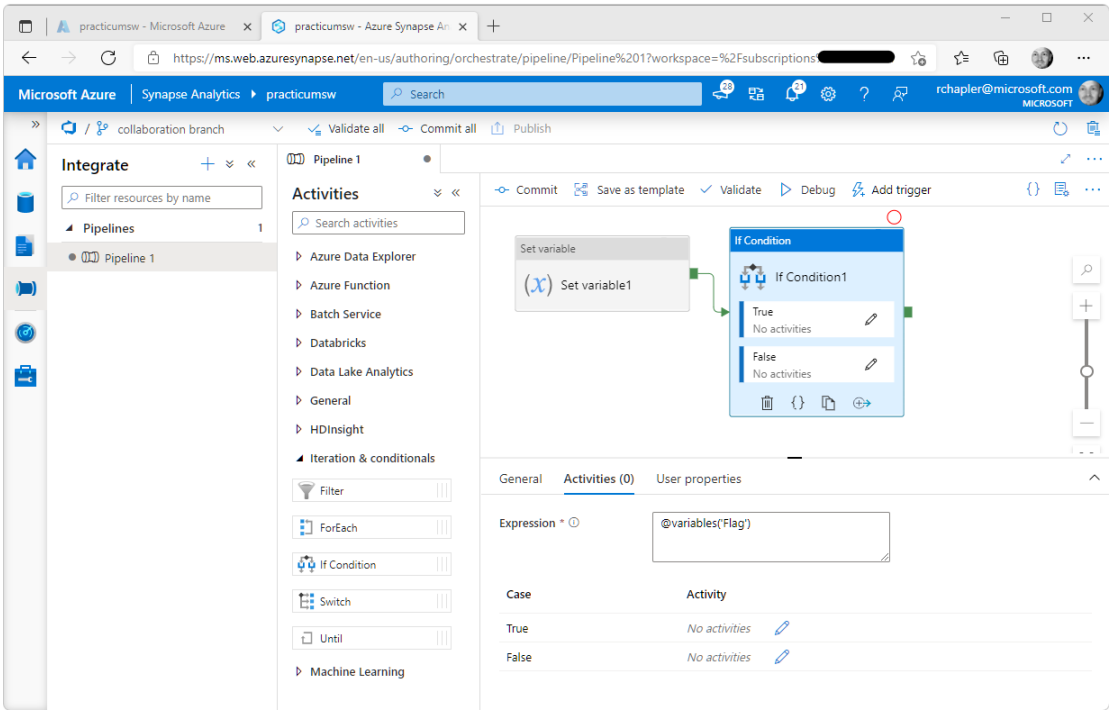
On the **Variables** tab of the “**Set Variable**…” component, enter values for the following items:

Name	Select **Flag** from the dropdown
Value	Enter **true**

Activity 2, If Condition

Expand “Iteration & conditionals” in the Activities bar.

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

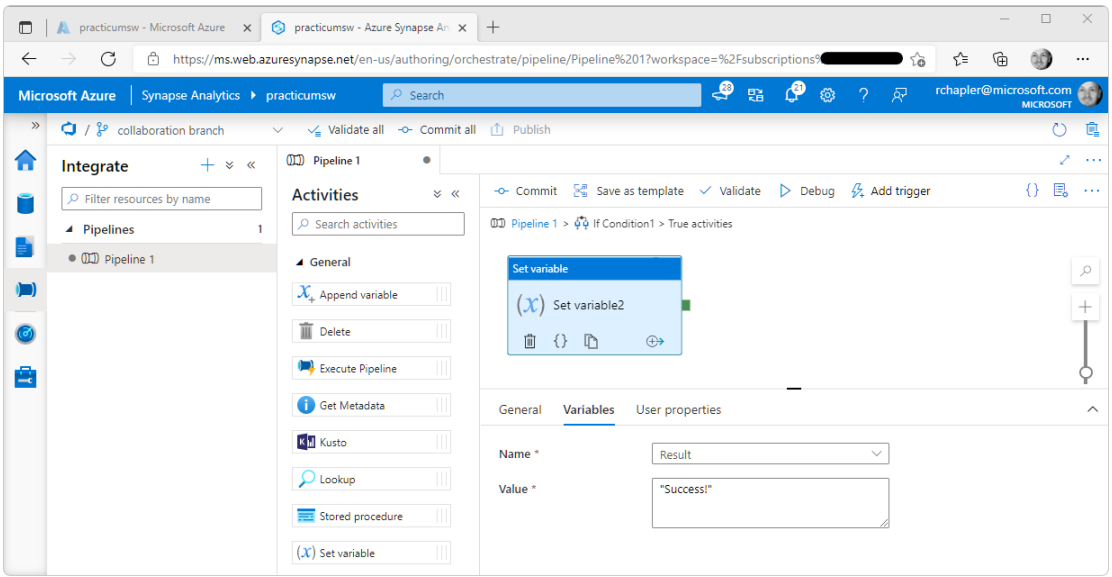


Create a dependency between the "**Set variable**" and "**If Condition**" components.

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.



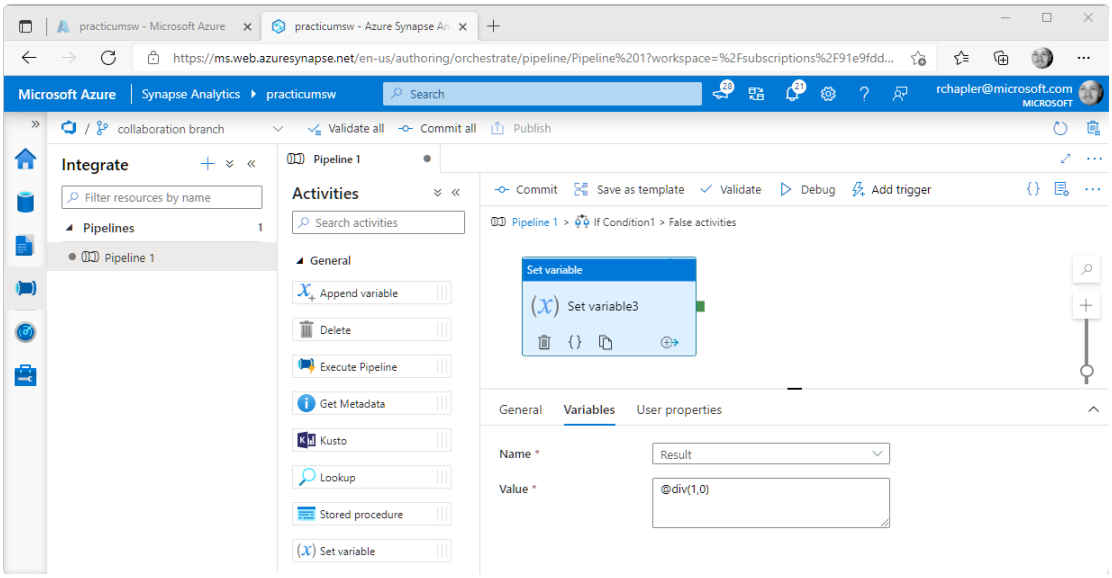
Expand **General** in the **Activities** bar, and then drag-and-drop a "**Set variable**" component into the activity window.

On the **Variables** tab of the "**Set Variable**..." component, enter values for the following items:

Name	Select **Result** from the dropdown
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.



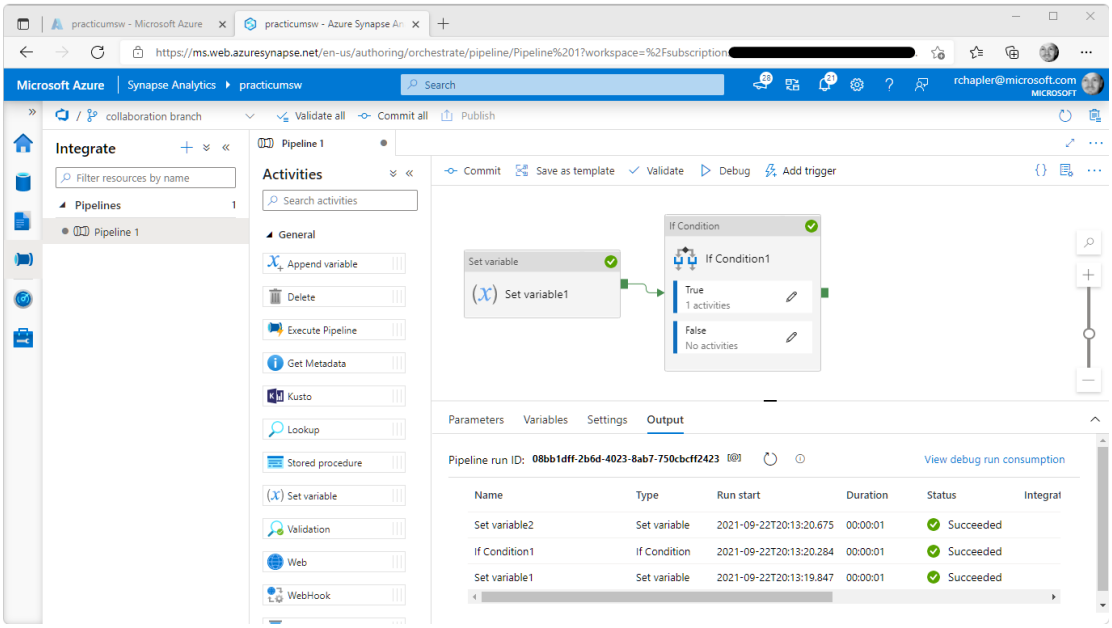
Expand **General** in the **Activities** bar, and then drag-and-drop a "**Set variable**" component into the activity window.

On the **Variables** tab of the "**Set Variable**..." component, enter values for the following items:

Name	Select **Result** from the dropdown
Value	Enter **@div(1,0)**... this will force an error result

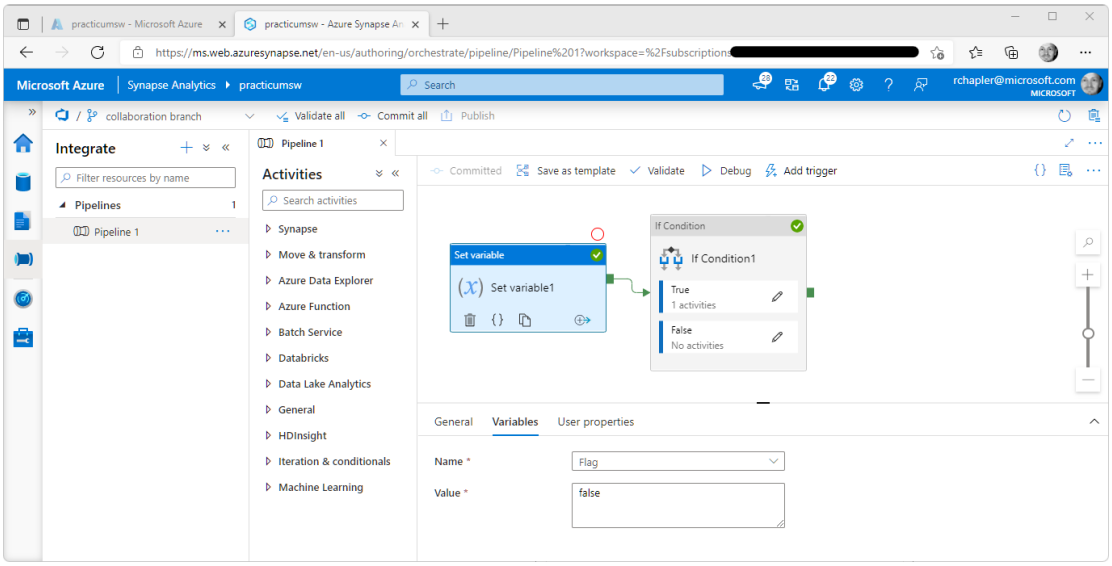
Step 4: Confirm Success

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.

Objective Complete... congratulations!

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.

Step 1: Understand Use Case

Work with your customer to understand requirements; 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.

Step 2: Instantiate Prerequisites

This exercise requires the following resource(s):

- 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.

Step 3: Prepare Source

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

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

Step 4: Prepare Destination

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

Create Destination Table

Execute the following query:

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

Logic explained…

- 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 query:

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

Logic explained…

- "**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

Create 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 query:

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

Create _Waterline Table

Execute the following query:

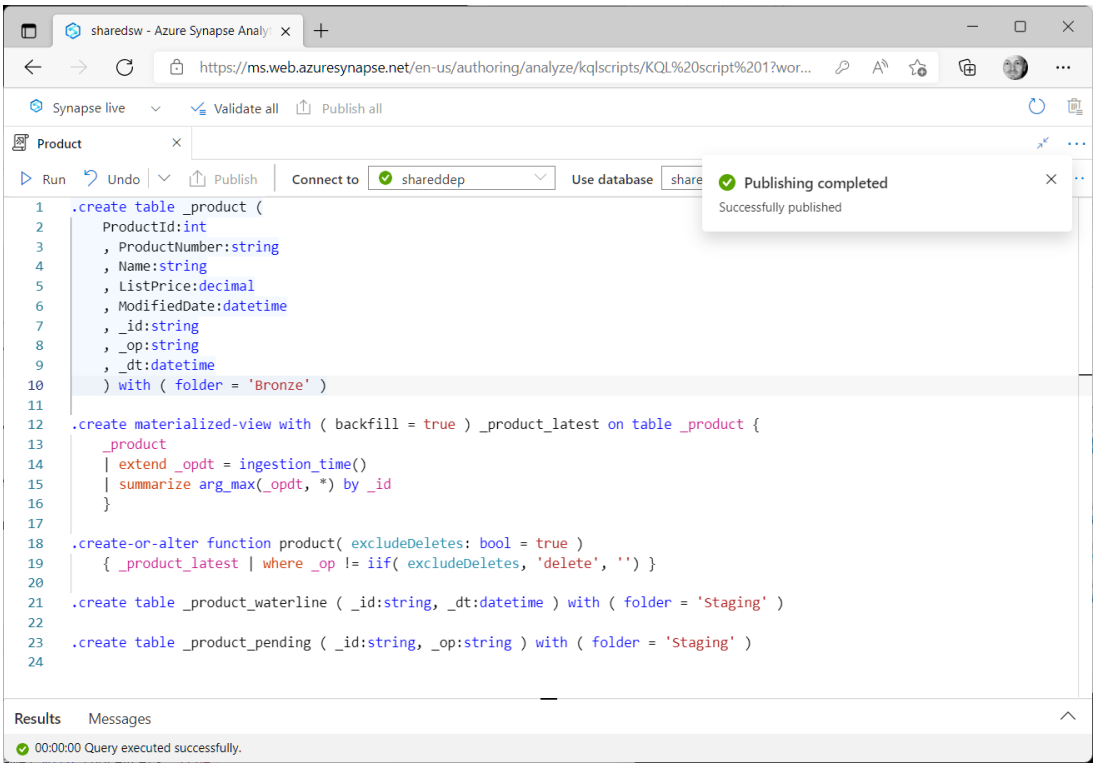
```
.create table _product_waterline ( _id:string, _dt:datetime ) with ( folder = 'Staging' )
```

Create _Pending Table

Execute the following query:

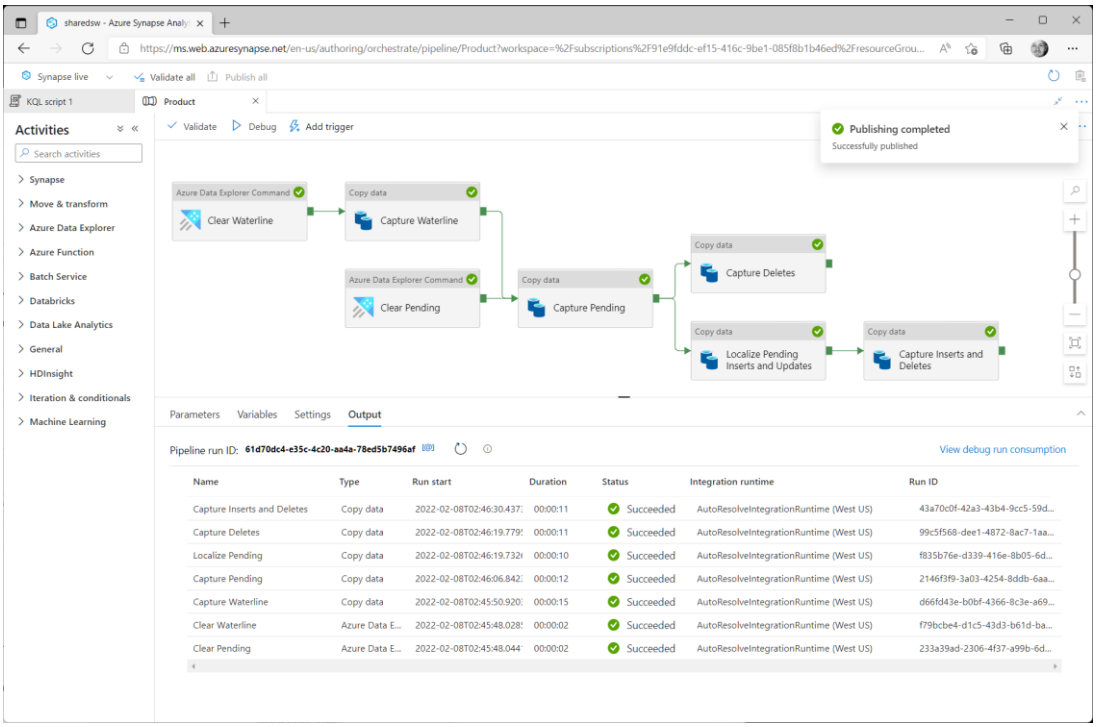
```
.create table _product_pending ( _id:string, _op:string ) with ( folder = 'Staging' )
```

Name, commit and / or publish your script.



Step 5: 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 dropdown.

Activity 1: Clear Waterline

This activity will clear any previously created waterline data.

Expand "**Azure Data Explorer**" in the **Activities** bar, and 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 dropdown.

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, enter values for the following items:

Source Dataset	Create a new "**Azure SQL Database**" dataset using the **SQL** linked service and the "**None**" table {e.g., "**sharedsd_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 changes are required on this tab.

On the **Sink** tab, enter values for the following items:

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

No additional changes are 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: Clear Pending

This activity will clear any previously created pending data.

Expand "**Azure Data Explorer**" in the **Activities** bar, and 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 dropdown.

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

```
.clear table _product_pending data;
```

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

Activity 4: Capture Pending

This activity will capture a dataset to the destination database that will be used to characterize pending delete operations.

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 Waterline**" and "**Clear Pending**" components.

On the **Source** tab, enter values for the following items:

Source Dataset	Select sharedded_product_waterline
Use Query	Select the **Query** radio button

Query	Paste the following KQL:
	`_product_latest | project _product_latest_dt = _dt, _product_latest_id = _id, _product_latest_op = _op`
	`| join kind = fullouter ( _product_waterline | project _product_waterline_id = _id, _product_waterline_dt = _dt )`
	`on $left._product_latest_id == $right._product_waterline_id`
	`| project _id = case(`
	`isnull(_product_waterline_dt) and _product_latest_op != 'delete', _product_latest_id`
	`,_product_waterline_id`
	`)`
	`, _op = case(`
	`isnull(_product_waterline_dt) and _product_latest_op != 'delete', 'delete'`
	`, isnull(_product_latest_dt), 'insert'`
	`, _product_latest_dt < _product_waterline_dt, 'update'`
	`, ''`
	`)`
	`| where _op in ('delete', 'insert', 'update')`
	●

No additional changes are required on this tab.

On the **Sink** tab, enter values for the following items:

Sink Dataset	Create an “Azure Data Explorer (Kusto)” dataset using the Data Explorer linked service and the “_product_pending” table {e.g., “sharedded_product_pending”}
Database and Table	Confirm values

No additional changes are 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 5: Capture Deletes (from destination)

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, enter values for the following items:

Source Dataset	Select the sharedded_product_pending dataset
Use Query	Select the **Query** radio button

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

No additional changes are required on this tab.

On the **Sink** tab, enter values for the following items:

Sink Dataset	Create an “**Azure Data Explorer (Kusto)**” dataset using the **Data Explorer** linked service and the “**_product**” table {e.g., “**sharedded_product**”}
Database and Table	Confirm values

No additional changes are required on this tab.

On the **Mapping** tab, click the “**Import schemas**” button.

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

Activity 6: 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, enter values for the following items:

Source Dataset	Select the sharedded_product_pending dataset
Use Query	Select the **Query** radio button
Query	Paste the following parameterized KQL: `_product_pending | where _op in ( 'insert', 'update' )` •

No additional changes are required on this tab.

On the **Sink** tab, enter values for the following items:

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

No additional changes are required on this tab.

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

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

Activity 7: Capture Inserts and Updates (from source)

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, enter values for the following items:

Source Dataset	Select sharedsd_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 changes are required on this tab.

On the **Sink** tab, enter values for the following items:

Sink Dataset	Select sharedded_Product
Database and Table	Confirm values

No additional changes are required on this tab.

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

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

Step 6: 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., **sharedded**} and select "**New KQL script**" from the resulting dropdown.

Paste and Run the following query 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 query:

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

Use the returned value to execute the following query:

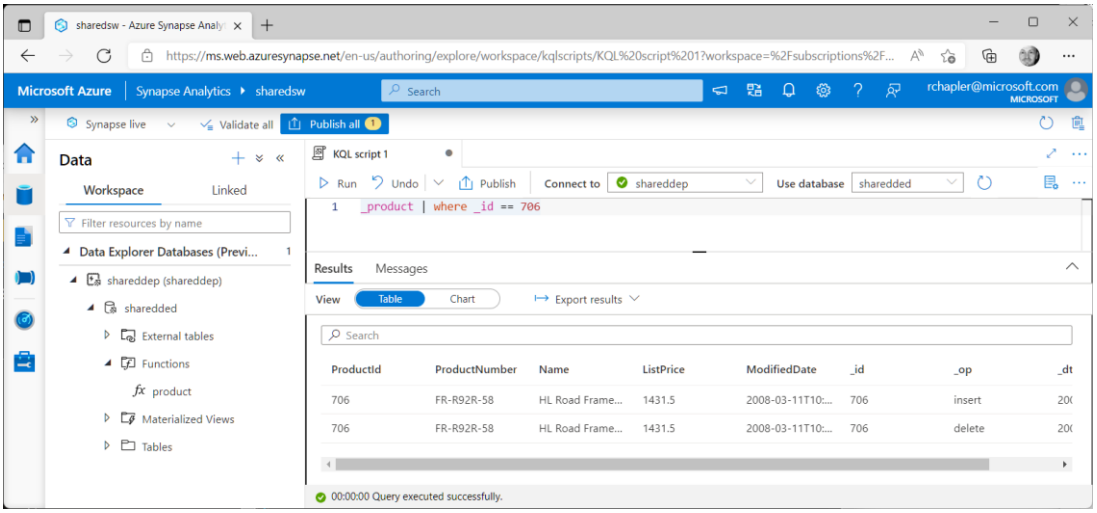
```
DELETE FROM [SalesLT].[Product] WHERE [ProductID] =  {MIN ProductId value}
```

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

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

Execute the following query 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 query:

```
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 query 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 query 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 query to confirm that the updated record is included in the Product table:

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

Step 6: 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 dropdown.

On the resulting popout, click the "**Choose trigger**…" dropdown and select "**+ New**".

On the "**New trigger**" popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
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 changes are required. Review remaining tabs and then click the **Commit** button.

Objective Complete… congratulations!

Data Flow

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

Step 1: Instantiate Prerequisites

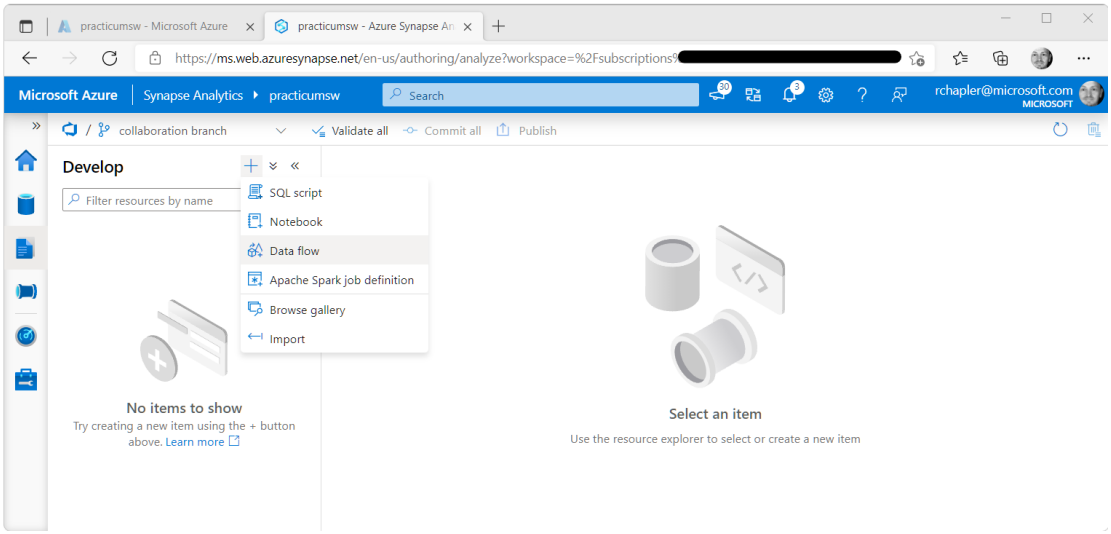
This exercise requires the following resource(s):

- 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.

Step 2: Initial Load

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



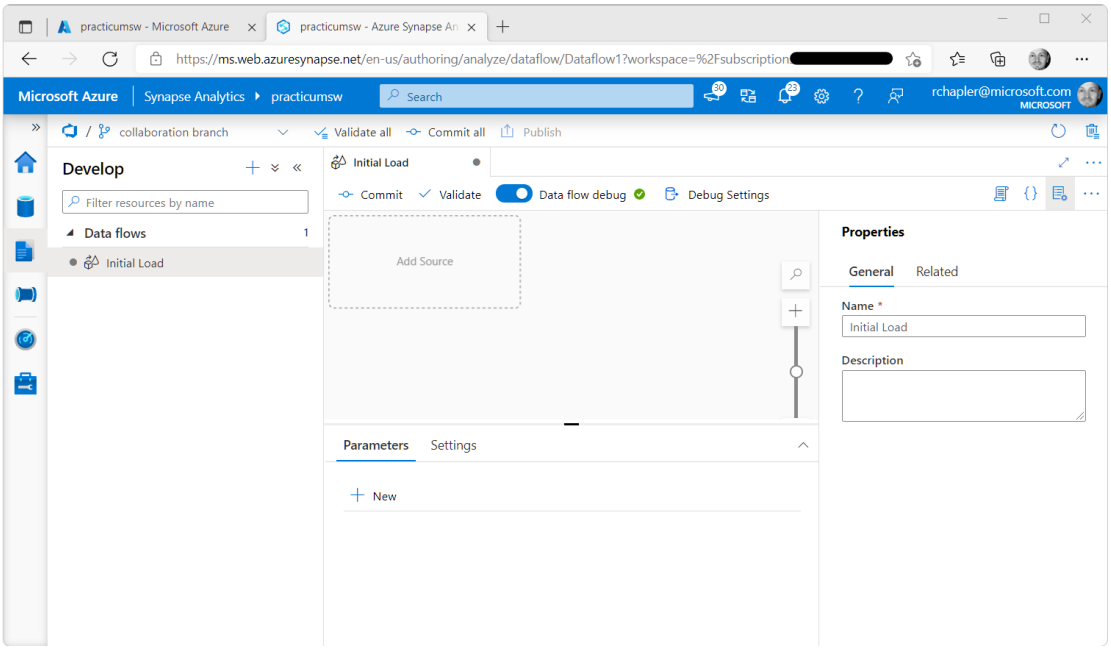
Navigate to your Synapse Analytics workspace and then click the **Open** link on the "**Open Synapse Studio**" rectangle.

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 dropdown.



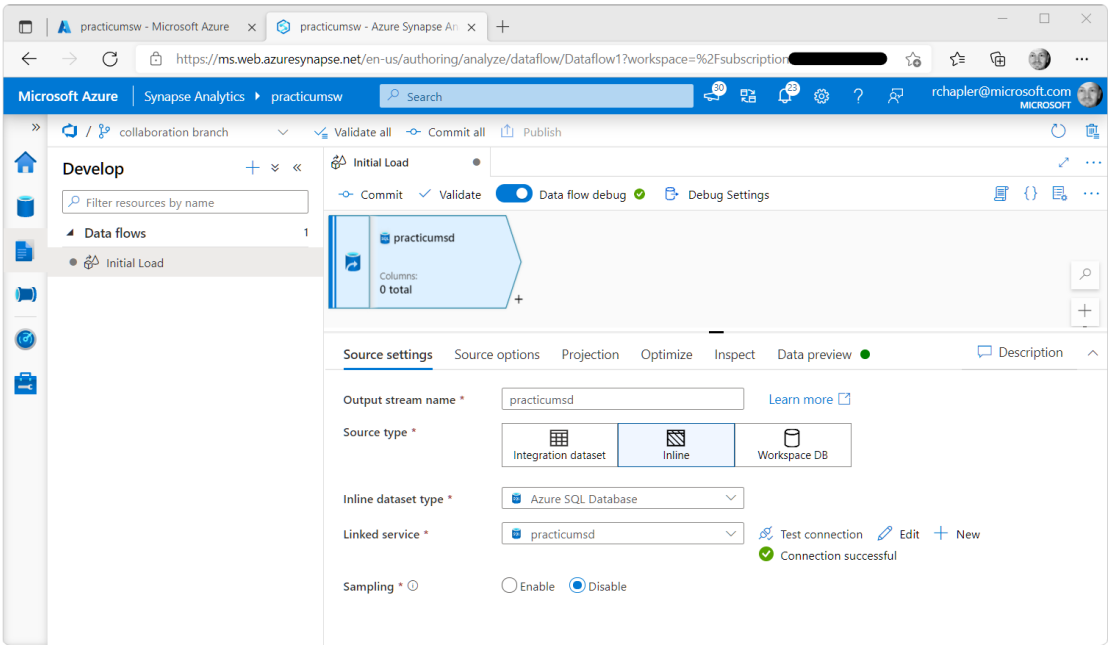
On the **Properties** popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards

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

Source (sample database)

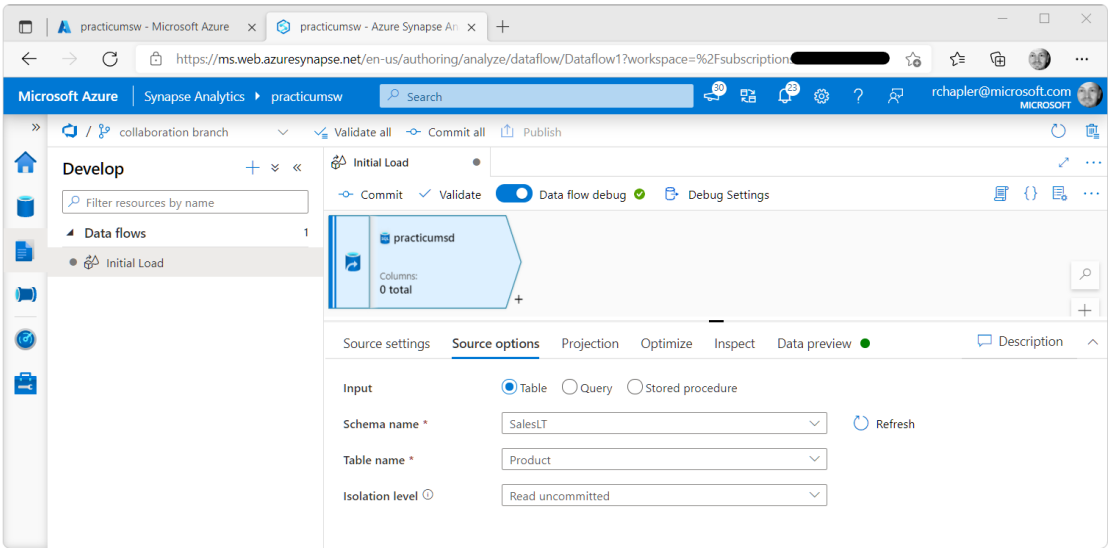
Click on "**Add Source**" in the dashed rectangle.



On the "**Source settings**" tab, enter values for the following items:

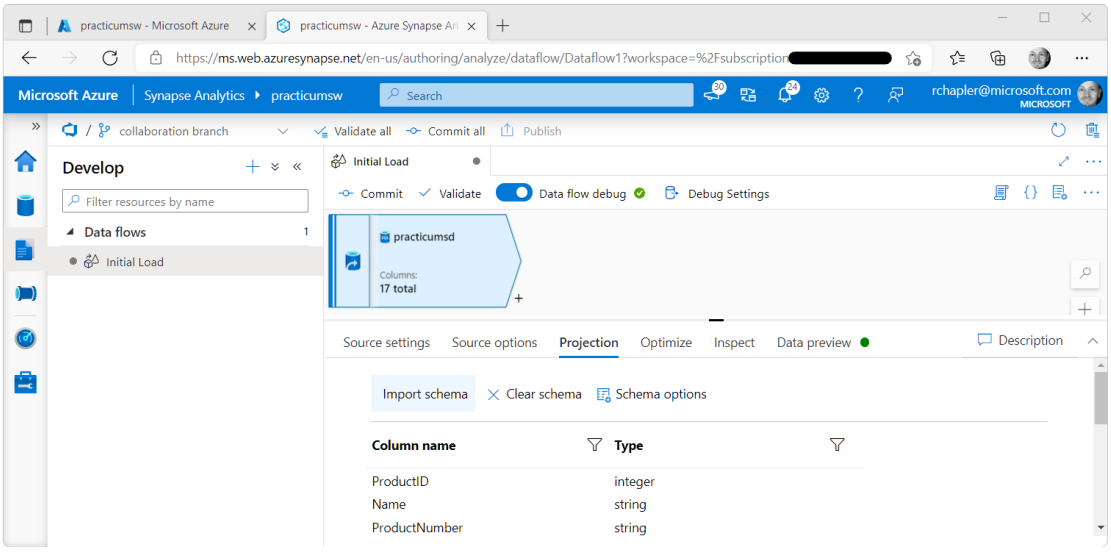
Output Stream Name	Enter a meaningful name aligned with standards
Source Type	Select Inline
Inline Dataset Type	Select "Azure SQL Database"
Linked Service	Select sharedsd
Sampling	Confirm default setting, **Disable**

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

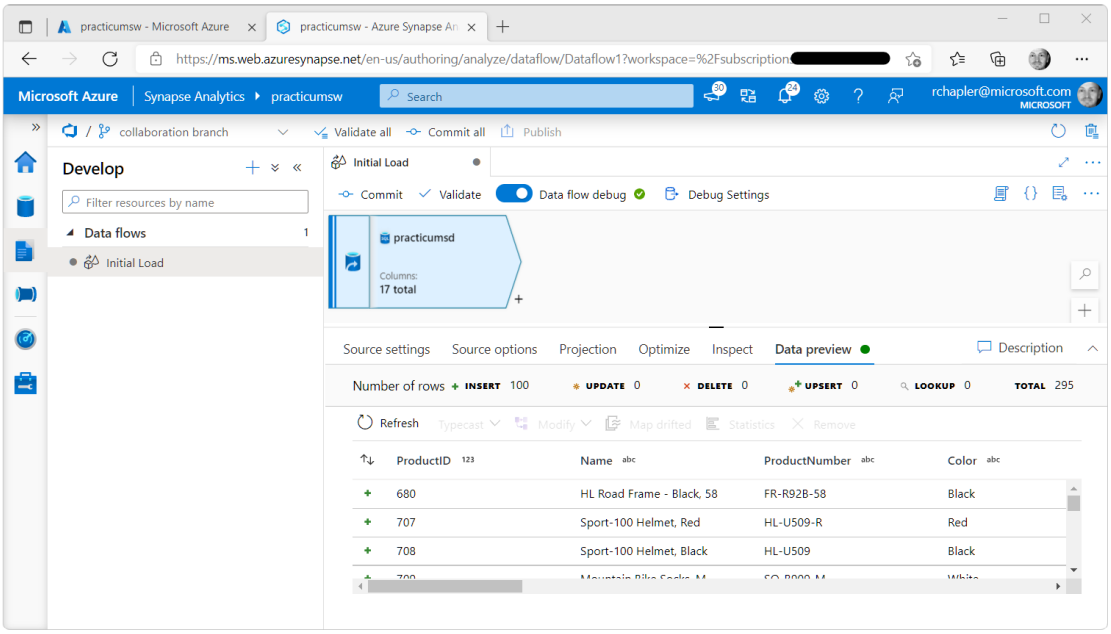


On the "**Source options**" tab, enter values for the following items:

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



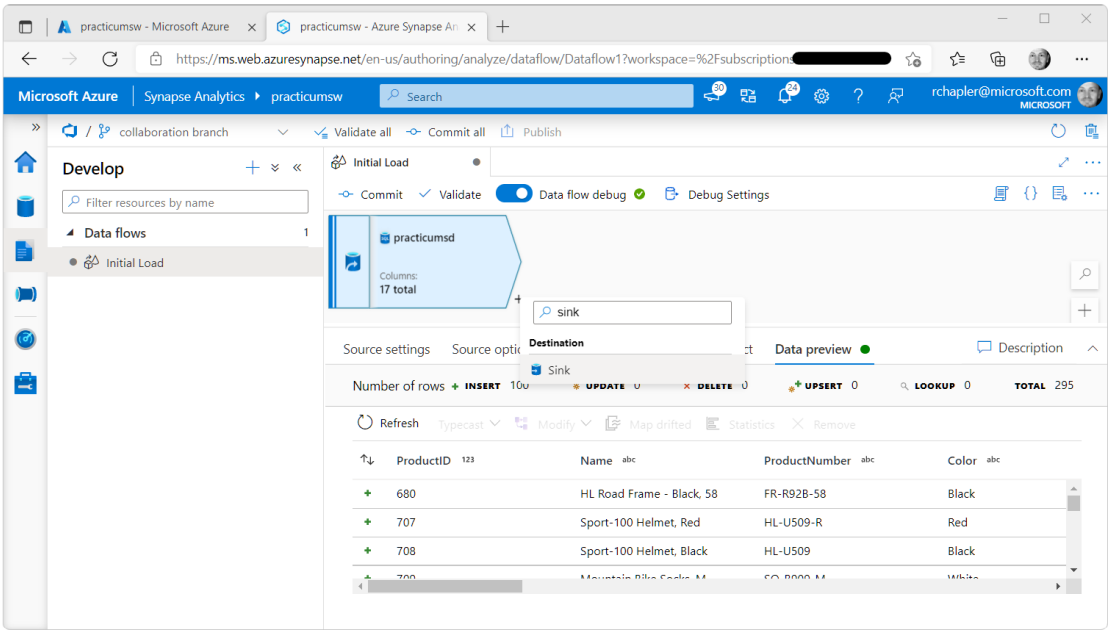
On the **Projection** tab, click "**Import schema**" and then **Import** on the resulting popout.



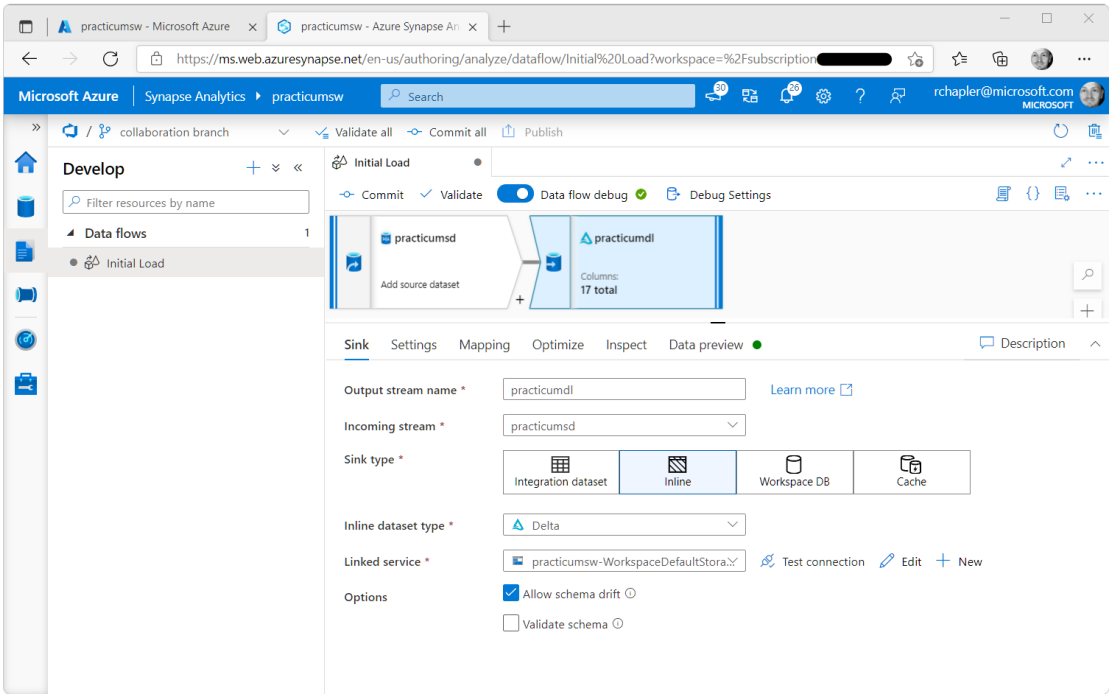
Navigate to the "**Data preview**" tab and then click **Refresh**.

Click the **Commit** button.

Sink (delta lake)

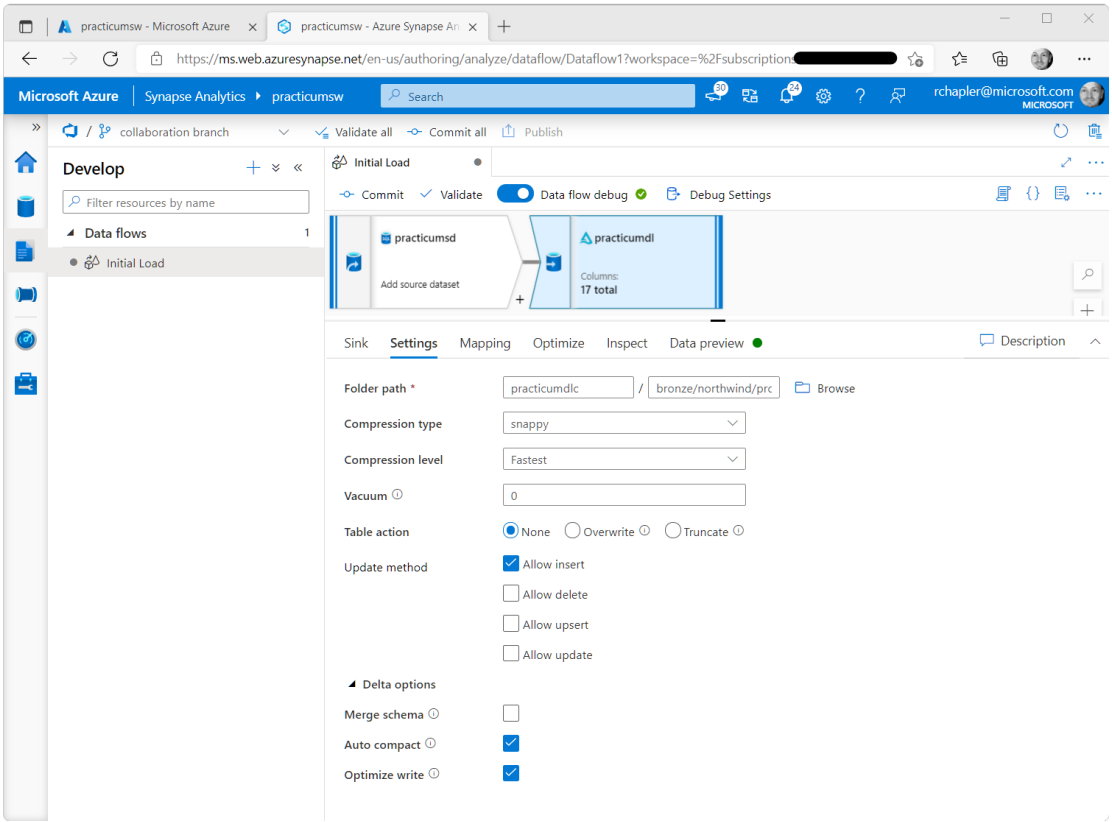


Click the **+** in the bottom right of the Data Lake source, and then search for and select **Sink** from the resulting popup list.



On the **Sink** tab, enter values for the following items:

Output Stream Name	Enter a meaningful name aligned with standards
Incoming Stream	Confirm default, **sharedsd**
Sink Type	Select **Inline**
Linked Service	Select practicumsw-WorkspaceDefaultStorage
Options	Confirm that “**Allow schema drift**” is checked



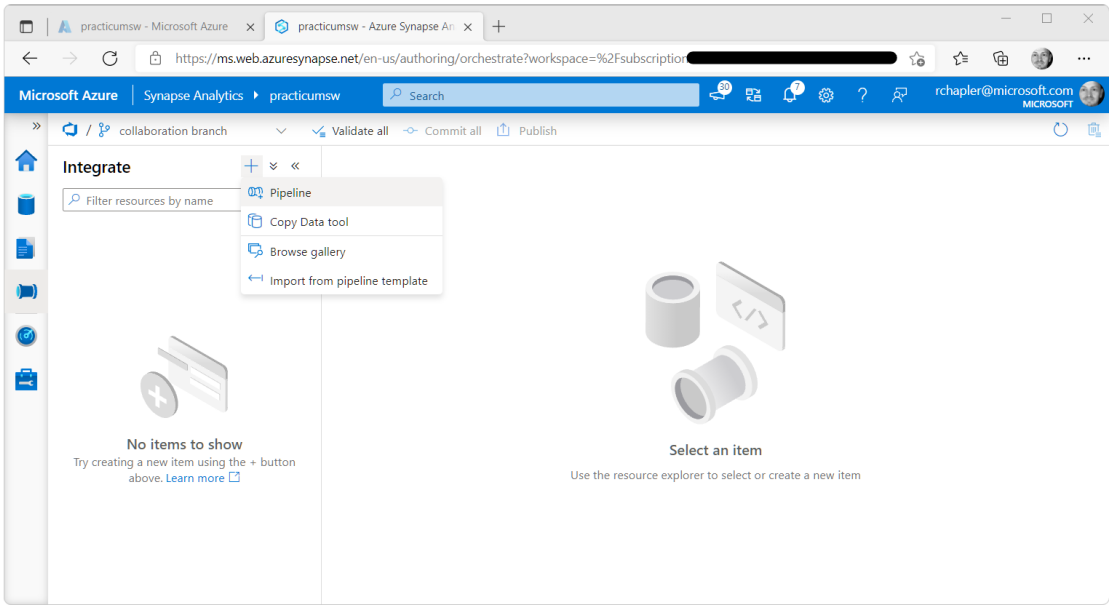
On the **Settings** tab, enter values for the following items:

File System	Enter shareddlc
Folder Path	Enter “bronze/northwind/product”
Compression Type	Select **snappy**
Compression Level	Select **Fastest**
Auto Compact	Checked
Optimize Write	Checked

No additional changes are required. Review settings on the remaining tabs.

Click the **Commit** button.

Create Pipeline

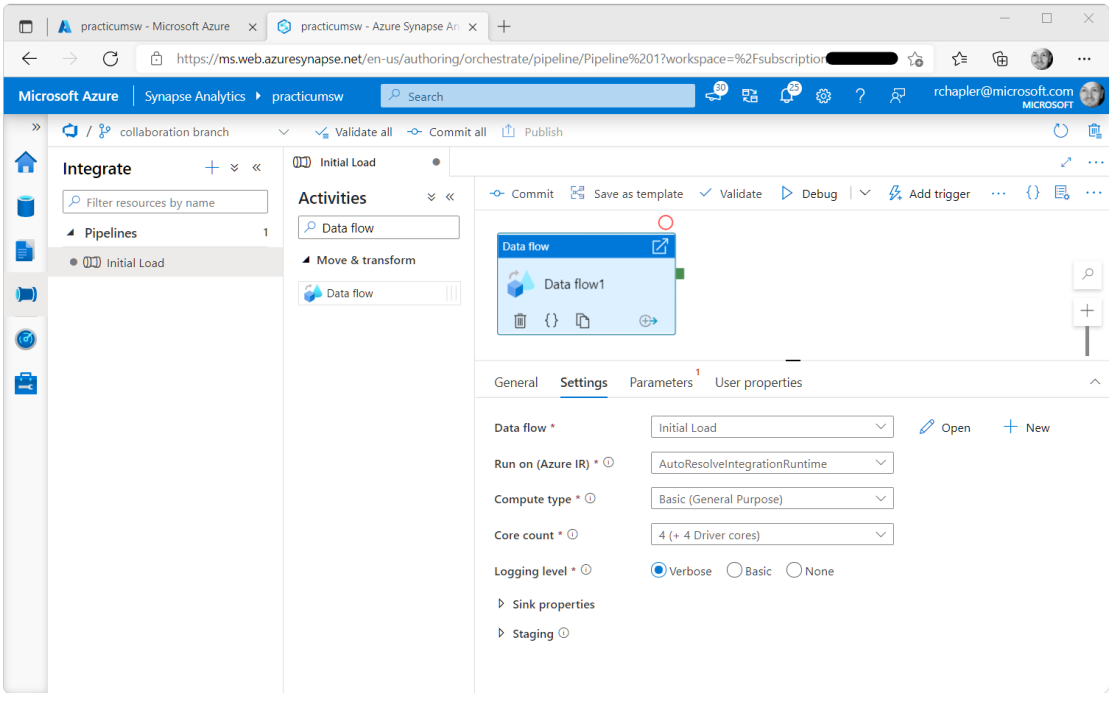


Click the **Integrate** icon in the navigation pane.

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

Select "**Pipeline**" from the resulting dropdown.

On the **Properties** popout, 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.

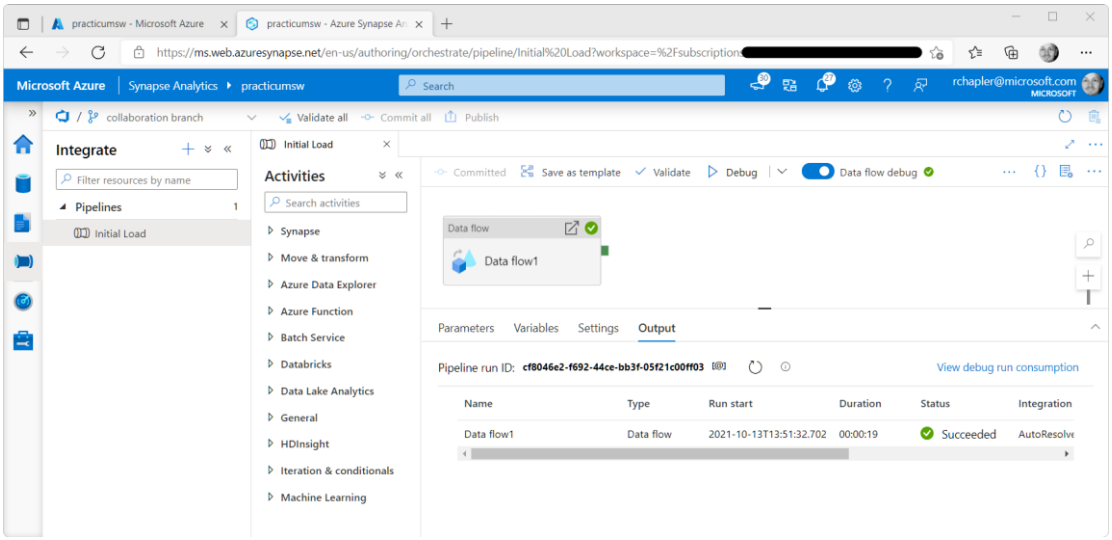
No additional changes are required. Review settings on the remaining tabs.

Click the **Commit** button.

Confirm Success

Debug Pipeline / Data Flow

Click **Debug** and confirm successful execution.



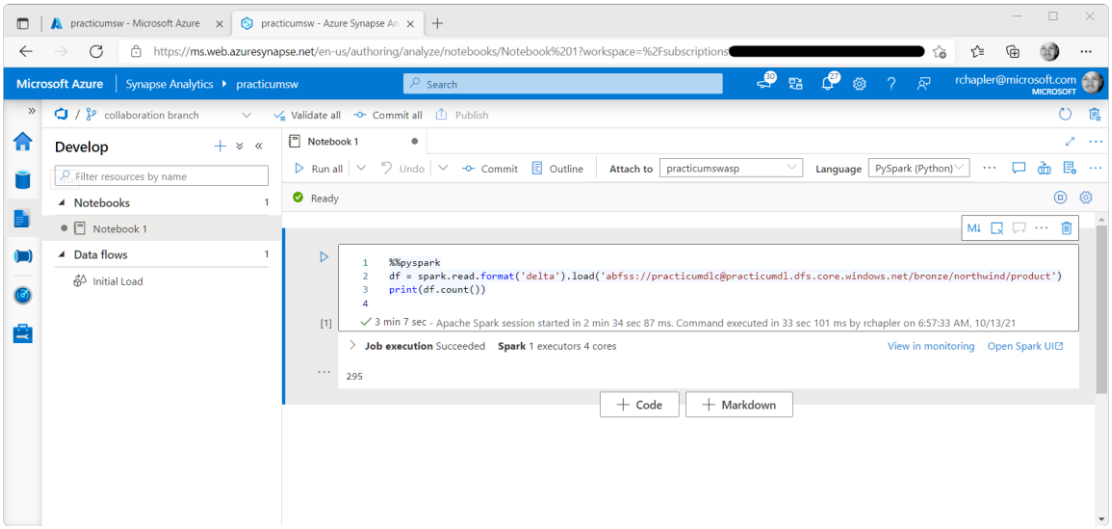
Query Delta Lake

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.

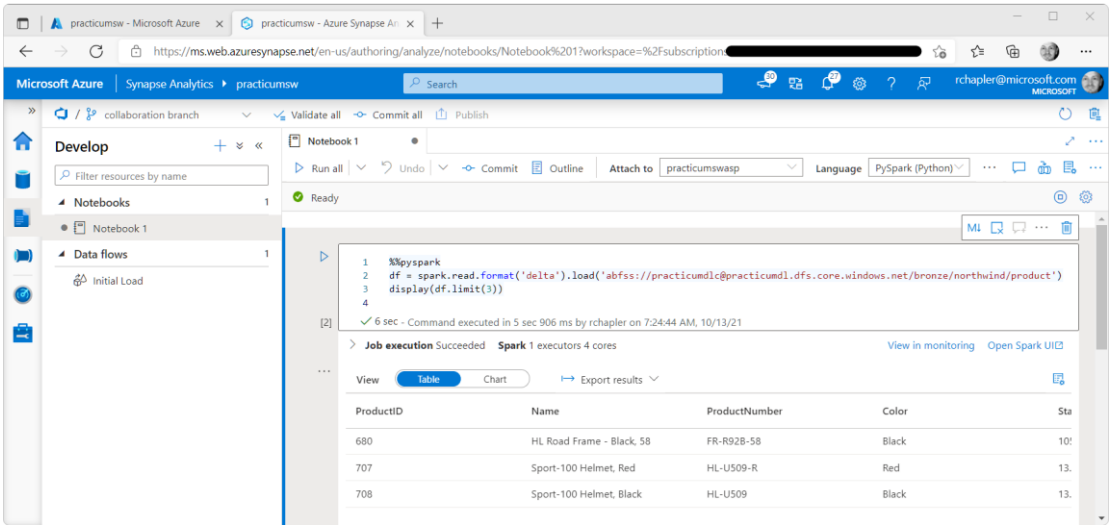
Select “**Notebook**” from the resulting dropdown.

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



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

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

Step 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.

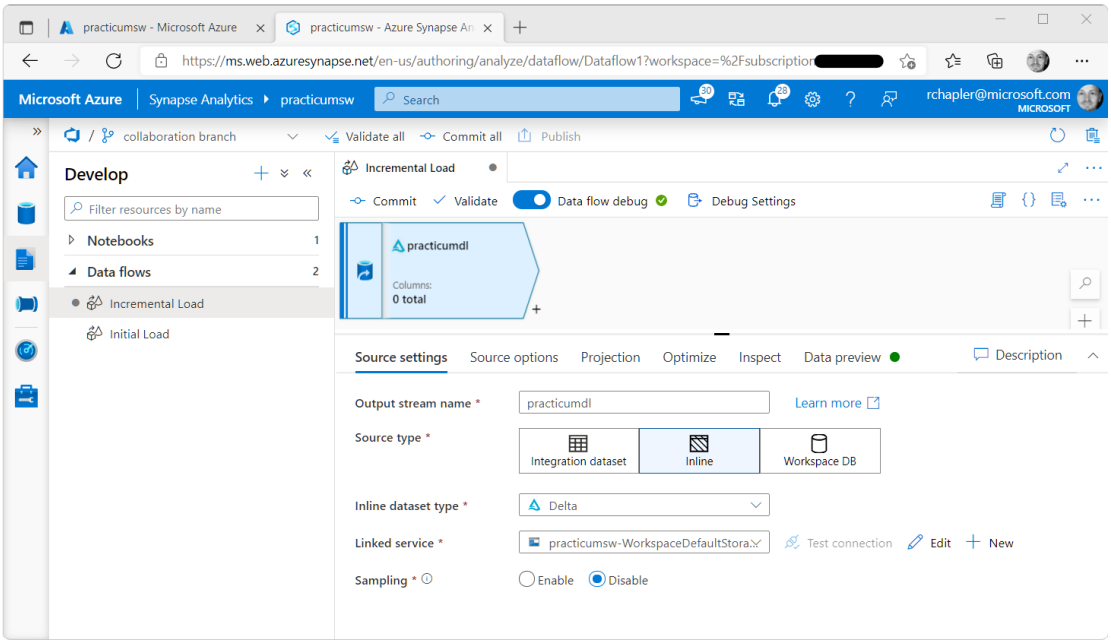
Select "**Data flow**" from the resulting dropdown.

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.

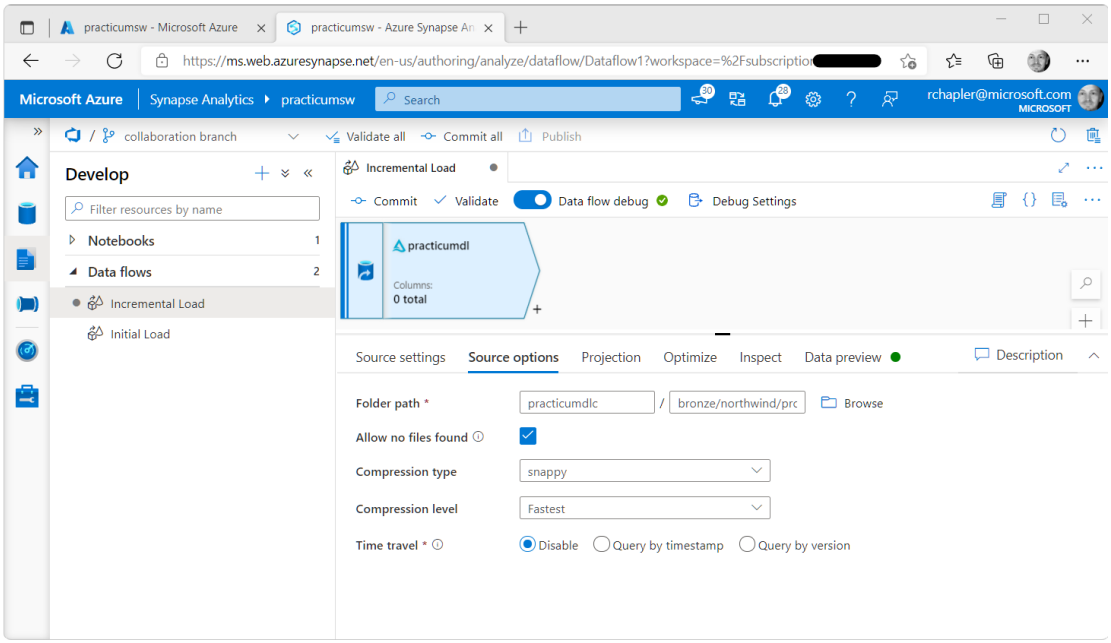
Click on "**Add Source**" in the dashed rectangle.



On the "**Source settings**" tab, enter values for the following items:

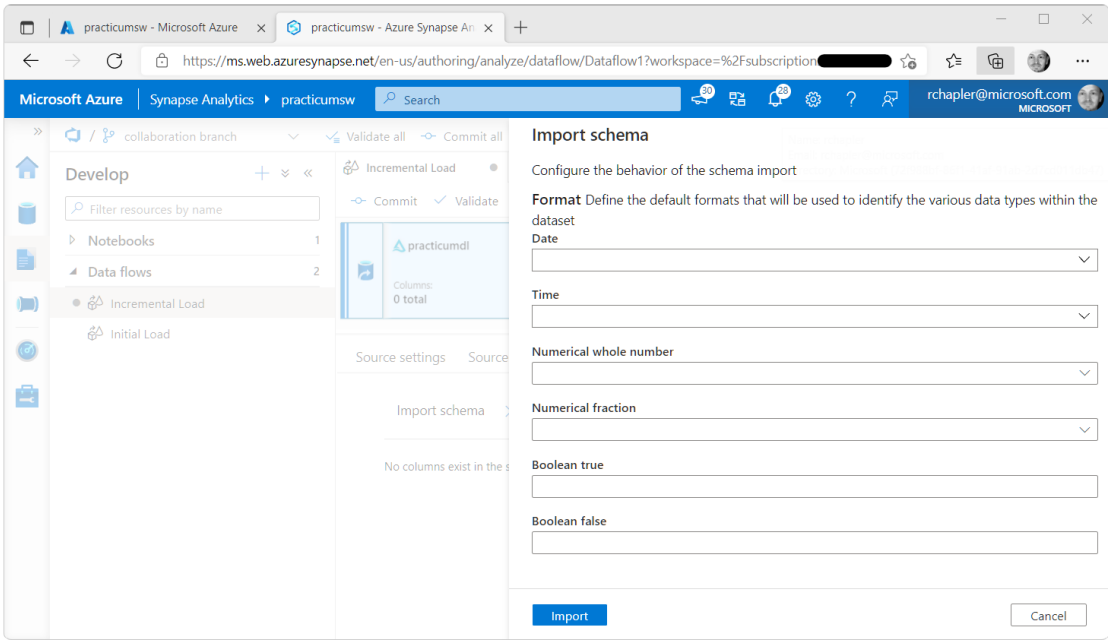
Output Stream Name	Enter a meaningful name aligned with standards
Source Type	Select Inline
Inline Dataset Type	Select **Delta**
Linked Service	Select practicumsw-WorkspaceDefaultStorage

Sampling	Confirm default setting, **Disable**



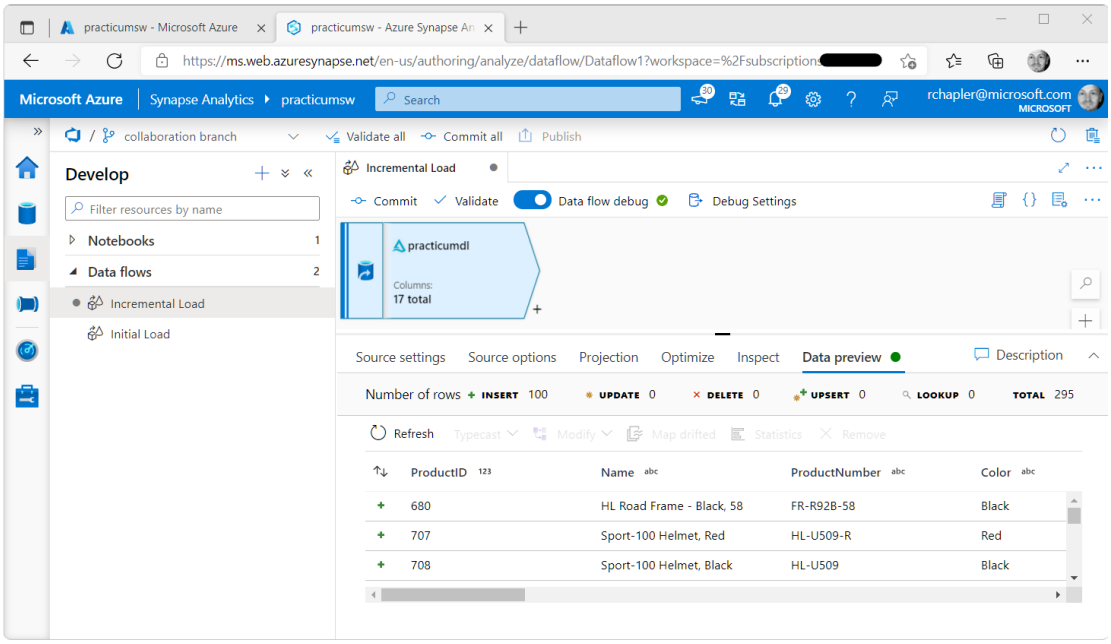
On the "**Source options**" tab, enter values for the following items:

File System	Enter shareddlc
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 popout.

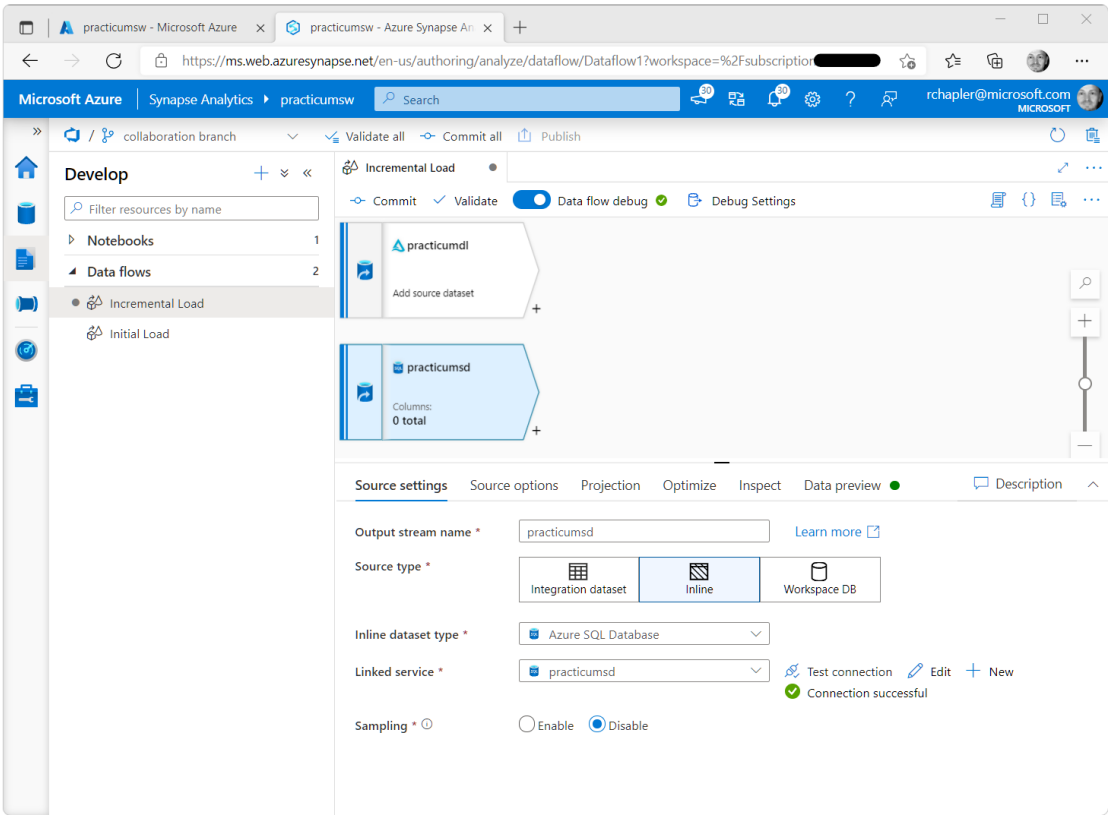
Navigate to the "**Data preview**" tab and then click **Refresh**.



Click the **Commit** button.

Source (sample database for this exercise)

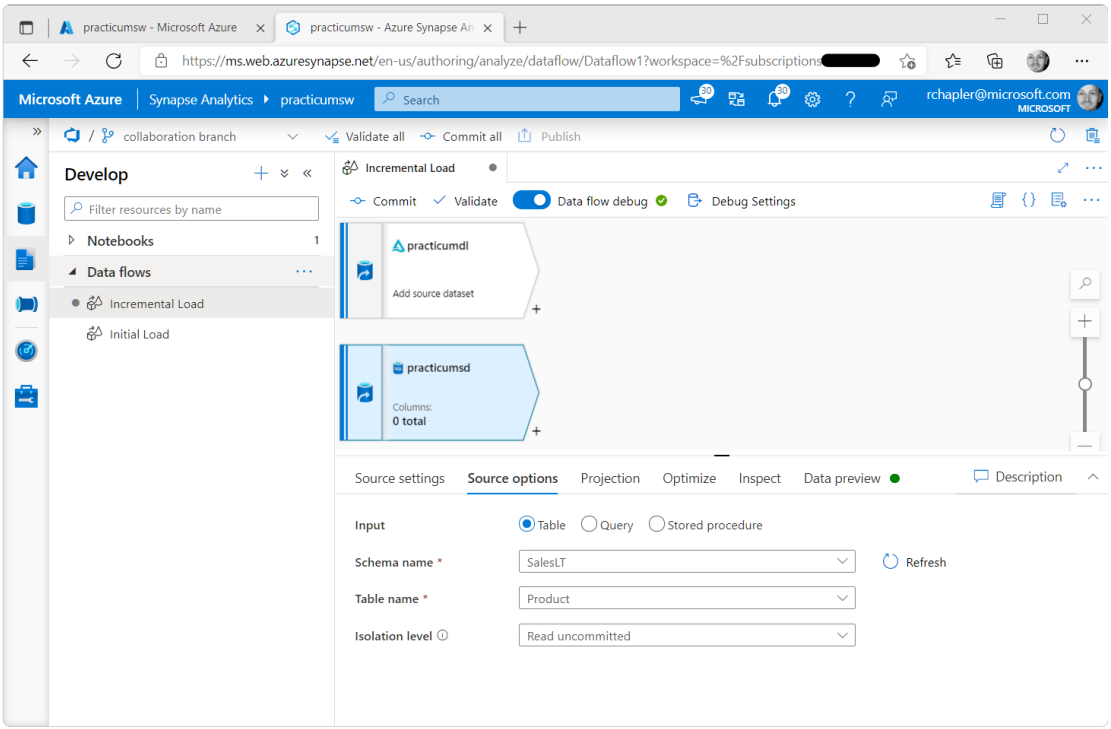
In this step, we will add the SQL Northwind sample database (our primary source for this exercise). Click on "**Add Source**" in the dashed rectangle.



On the "**Source settings**" tab, enter values for the following items:

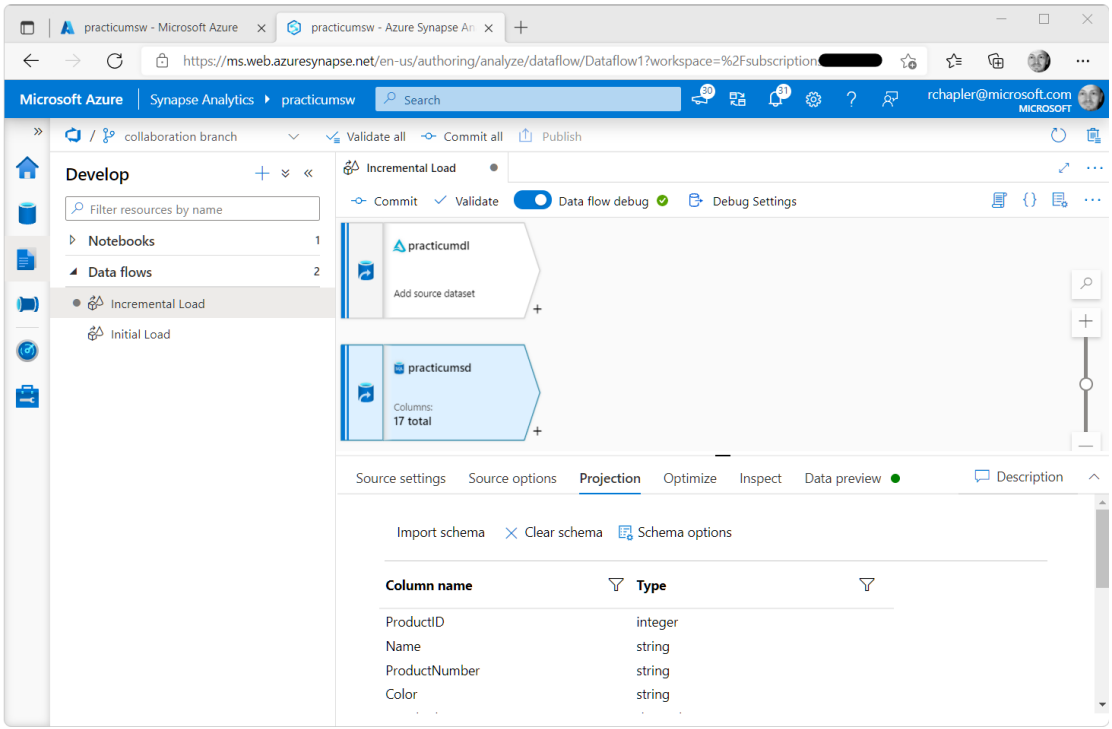
Output Stream Name	Enter a meaningful name aligned with standards
Source Type	Select Inline
Inline Dataset Type	Select "Azure SQL Database"
Linked Service	Select sharedsd
Sampling	Confirm default setting, **Disable**

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

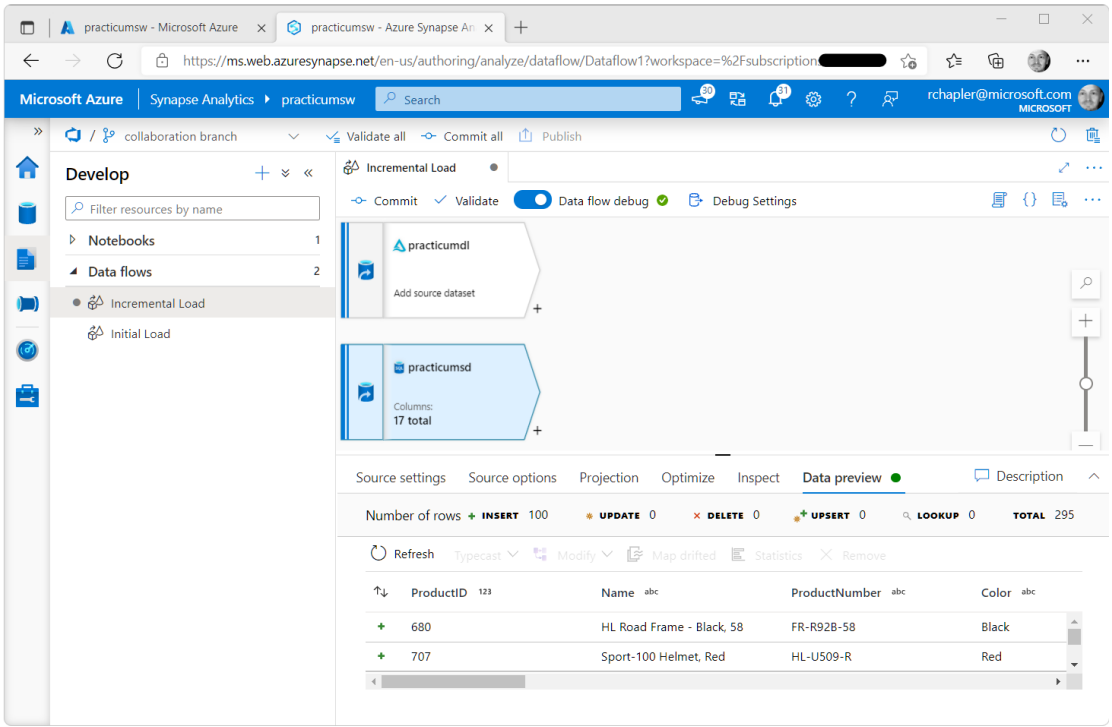


On the "**Source options**" tab, enter values for the following items:

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 popout.

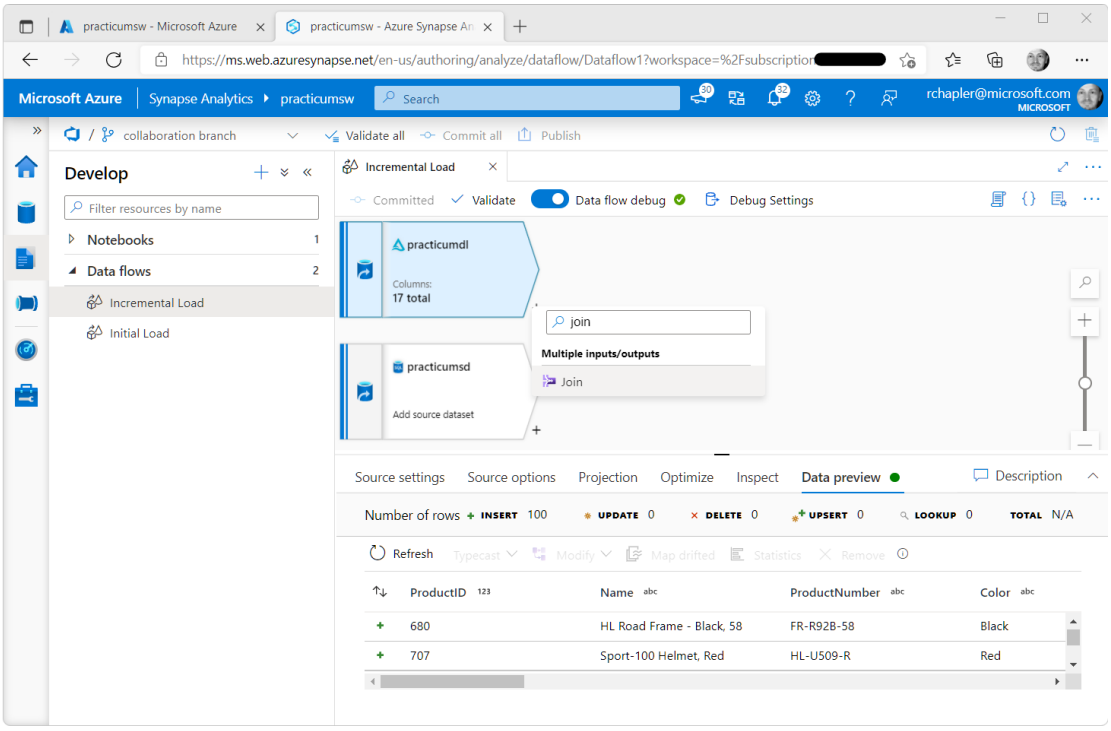


Navigate to the “**Data preview**” tab and then click **Refresh**.

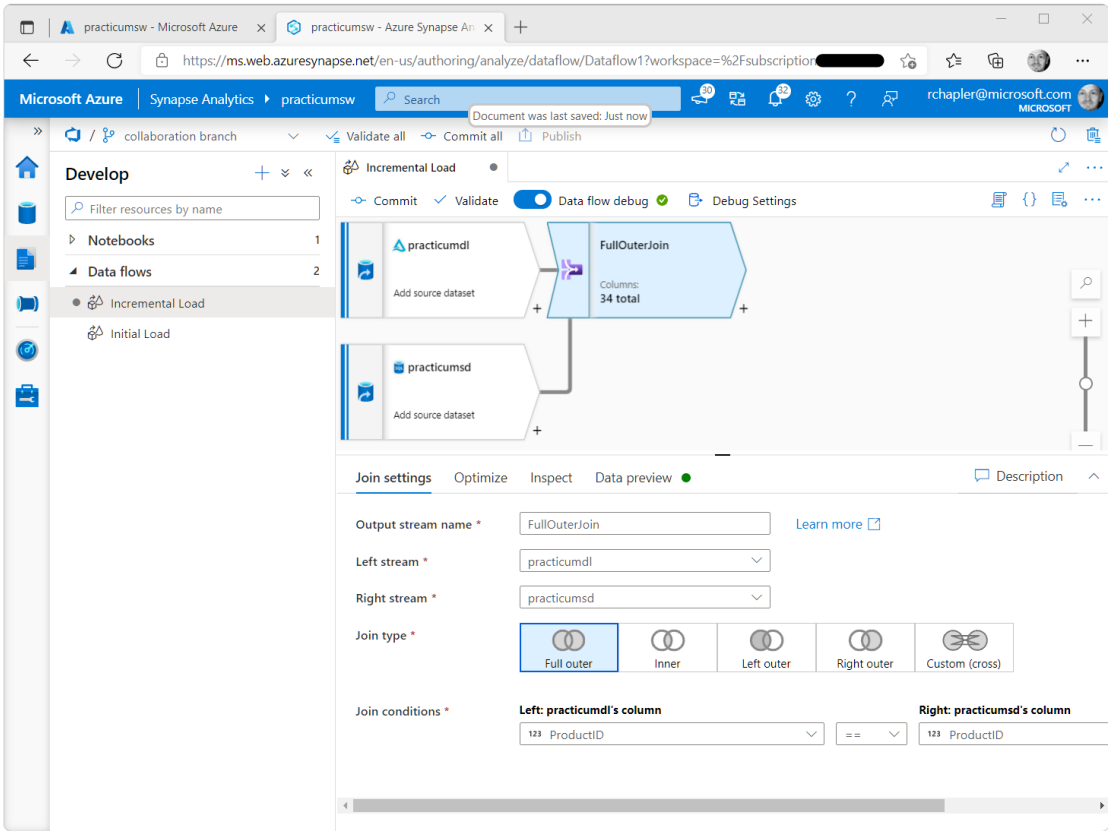
Click the **Commit** button.

Join Sources

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



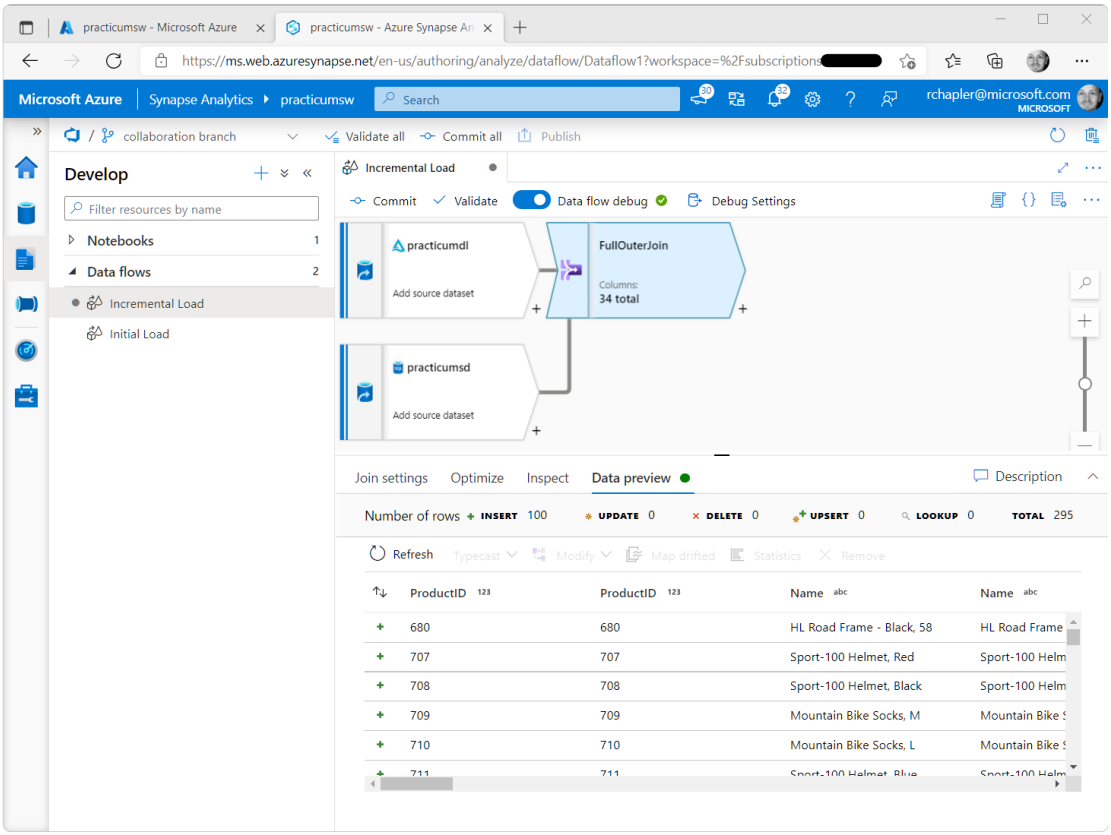
Click the **+** in the bottom right of the SQL source, and then select **Join** from the resulting popup list.



On the "**Join settings**" tab, enter values for the following items:

Output Stream Name	Enter a meaningful name aligned with standards
Left Stream	Select sharedsd
Right Stream	Select AggregateWaterline
Join Type	Select "Full outer"
Join Conditions	Select **ProductID** for both Left… and Right… column

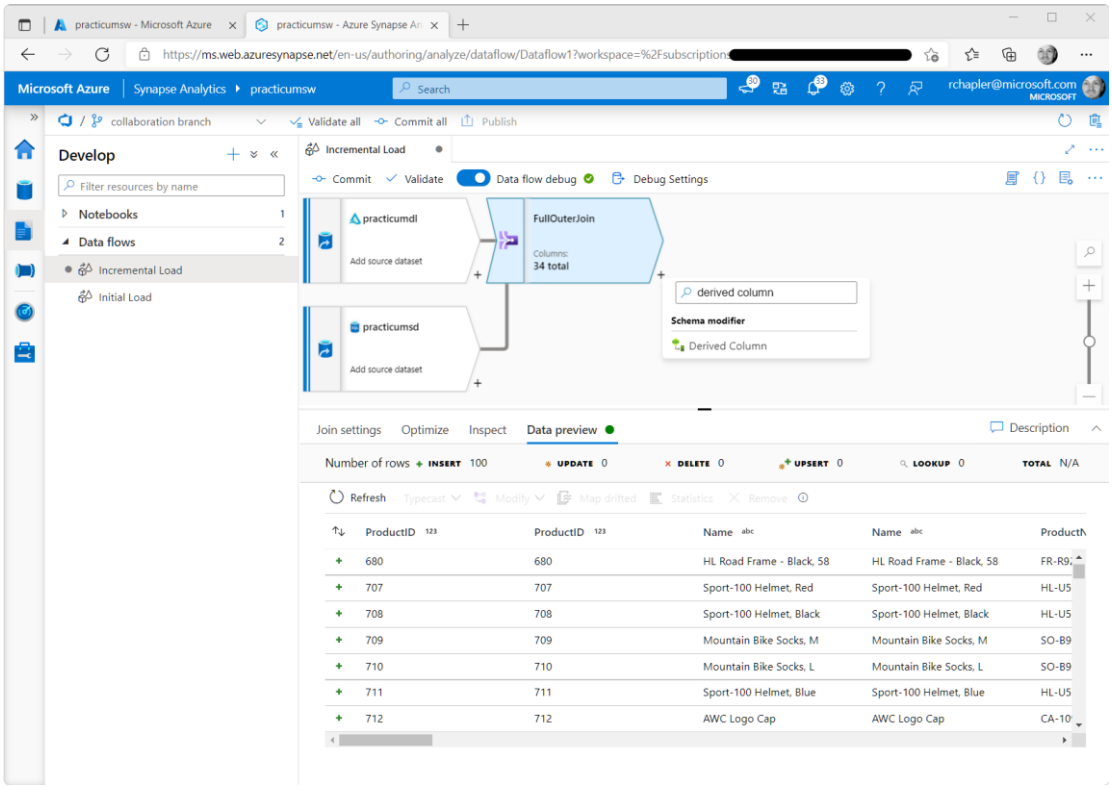
Navigate to the "**Data preview**" tab and then click **Refresh**.



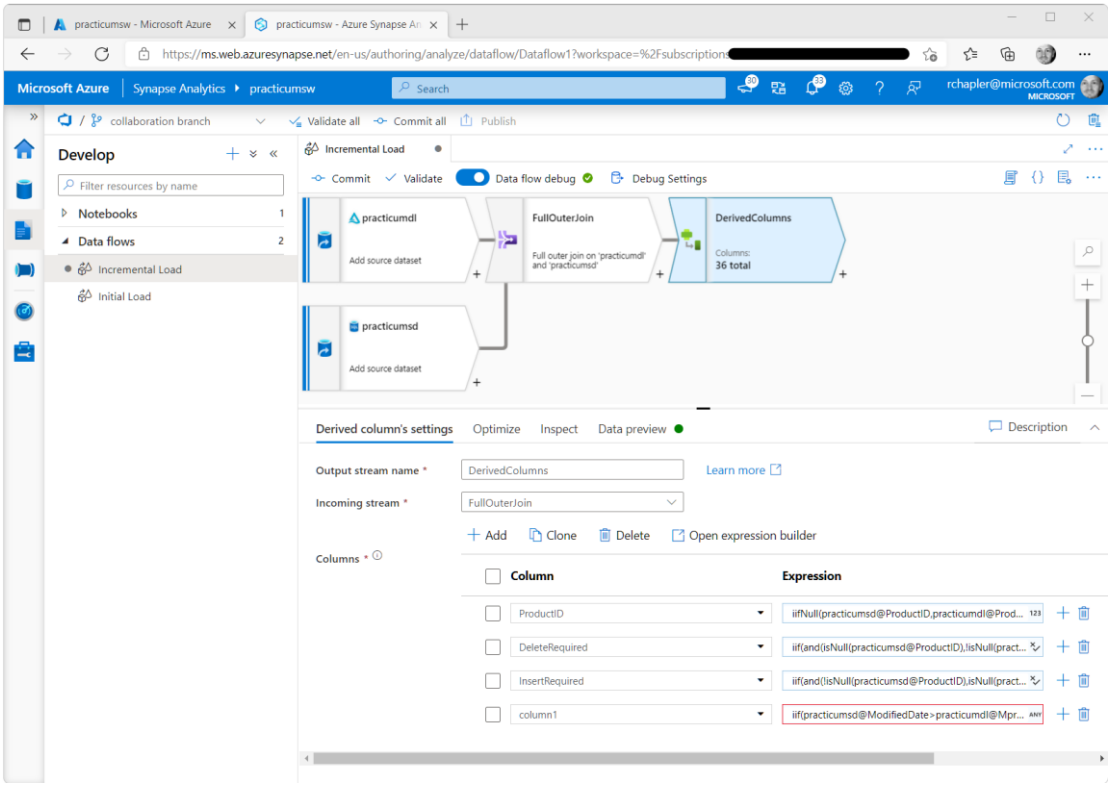
Click the **Commit** button.

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 popup list.

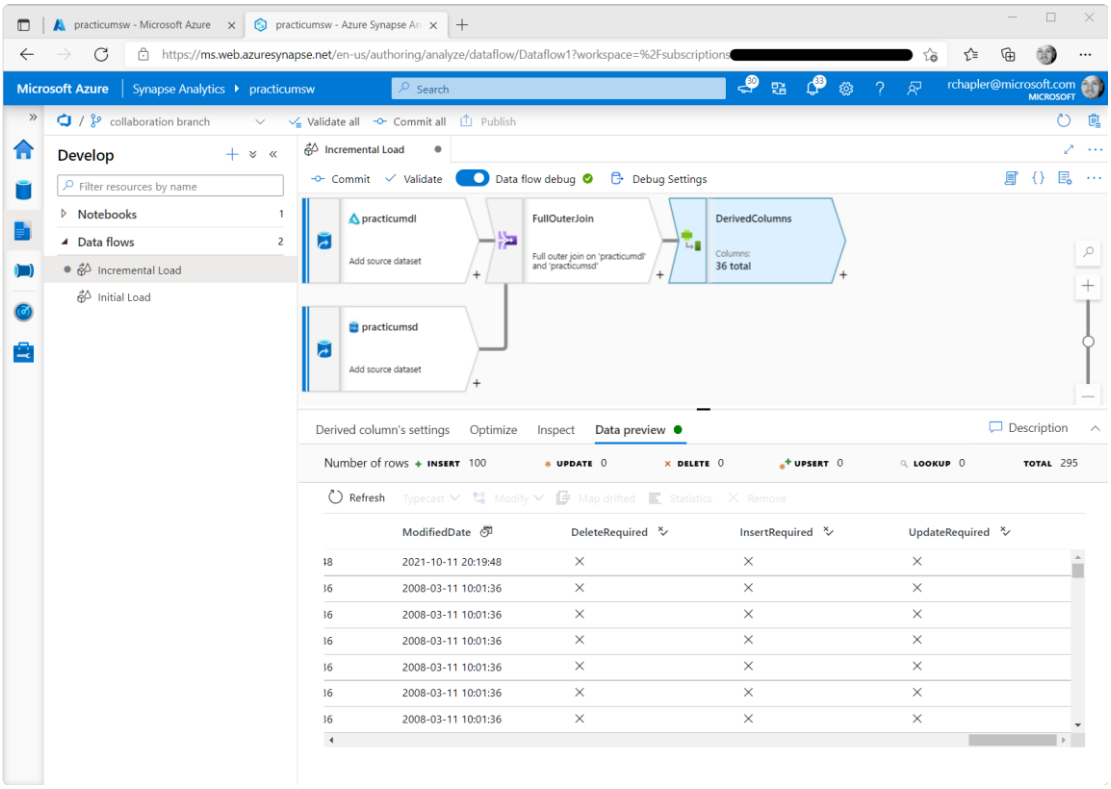


On the "**Derived column's settings**" tab, enter values for the following items:

Output Stream Name	Enter a meaningful name aligned with standards
Incoming Stream	Confirm default selection

Columns	ProductID
	`iifNull(sharedsd@ProductID,shareddl@ProductID)`
	
	DeleteRequired
	`iif(and(isNull(sharedsd@ProductID),!isNull(shareddl@ProductID)),true(),false())`
	
	InsertRequired
	`iif(and(!isNull(sharedsd@ProductID),isNull(shareddl@ProductID)),true(),false())`
	
	UpdateRequired
	`iif(sharedsd@ModifiedDate>shareddl@ModifiedDate,true(),false())`

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

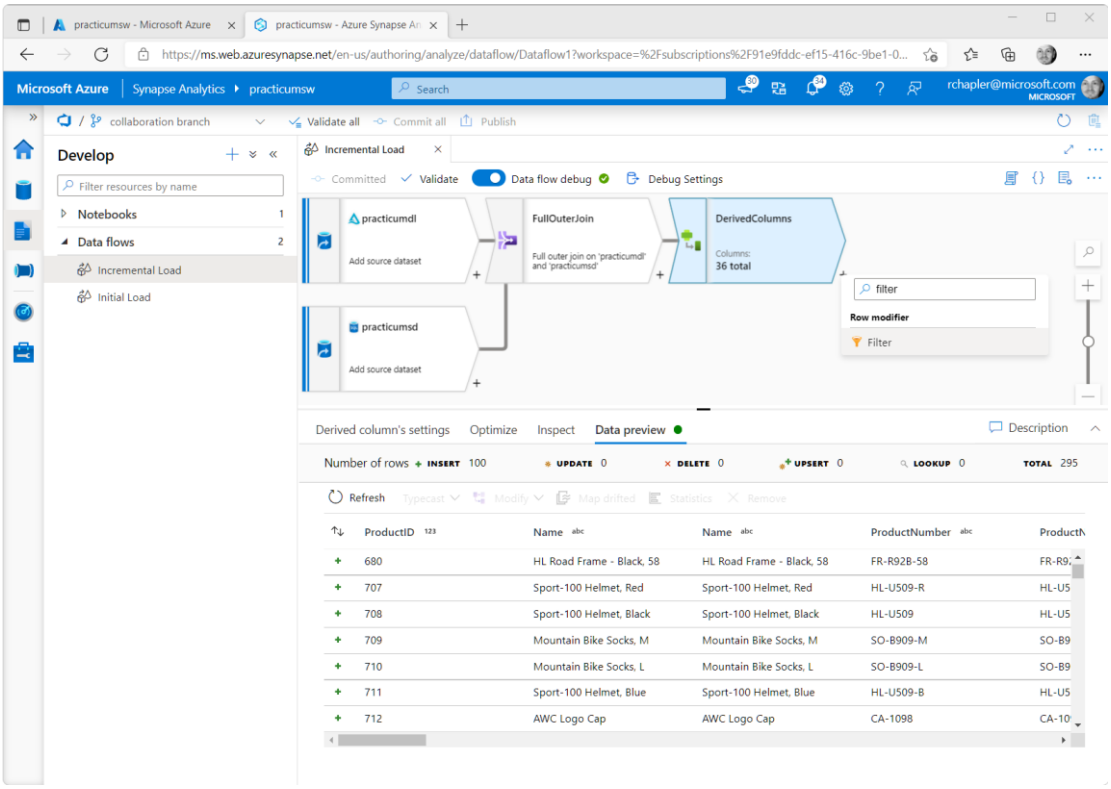


Click the **Commit** button.

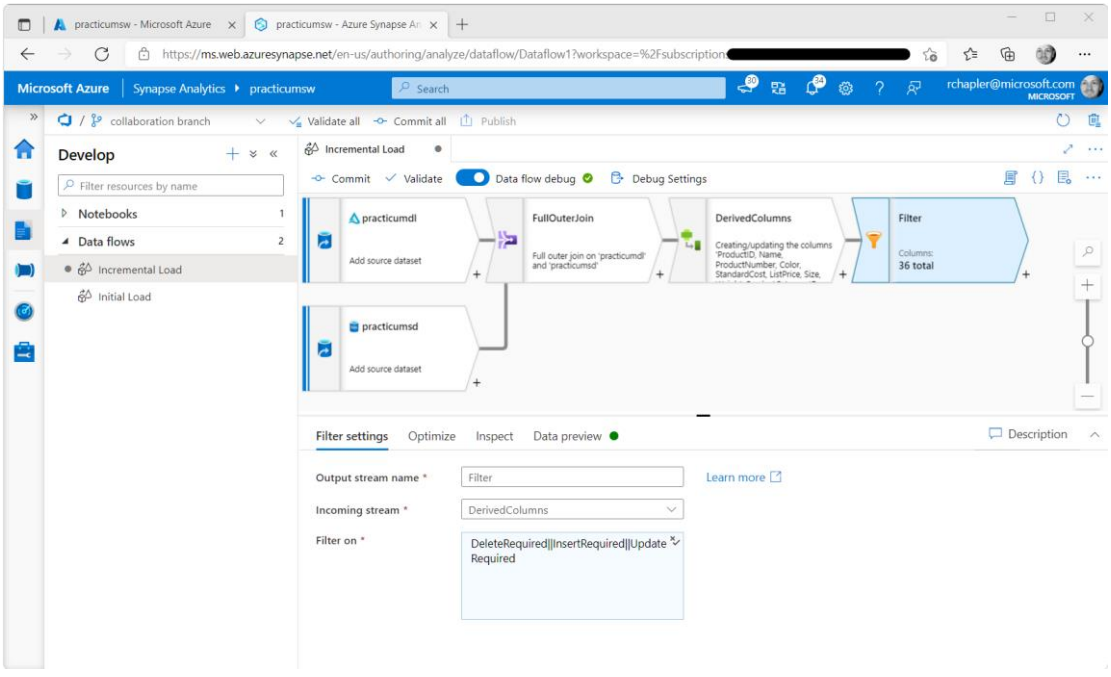
Note: I included this step to provide an easy way to preview and provide for the delete, insert and update of data… arguably, this step could be consolidated into Filter or AlterRows.

Filter Non-Actionable

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



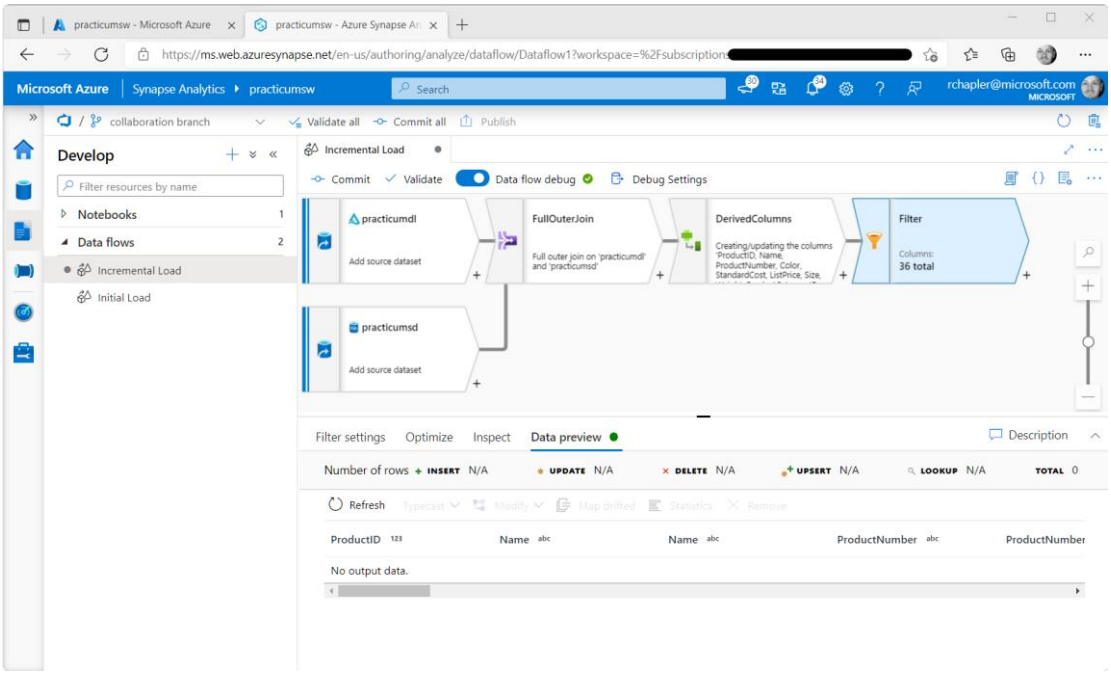
Click the **+** in the bottom right of the SQL source, and then select **Filter** from the resulting popup list.



On the “**Filter settings**” tab, enter values for the following items:

Output Stream Name	Enter a meaningful name aligned with standards
Incoming Stream	Confirm default selection
Filter On	`DeleteRequired||InsertRequired||UpdateRequired`

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



Click the **Commit** button.

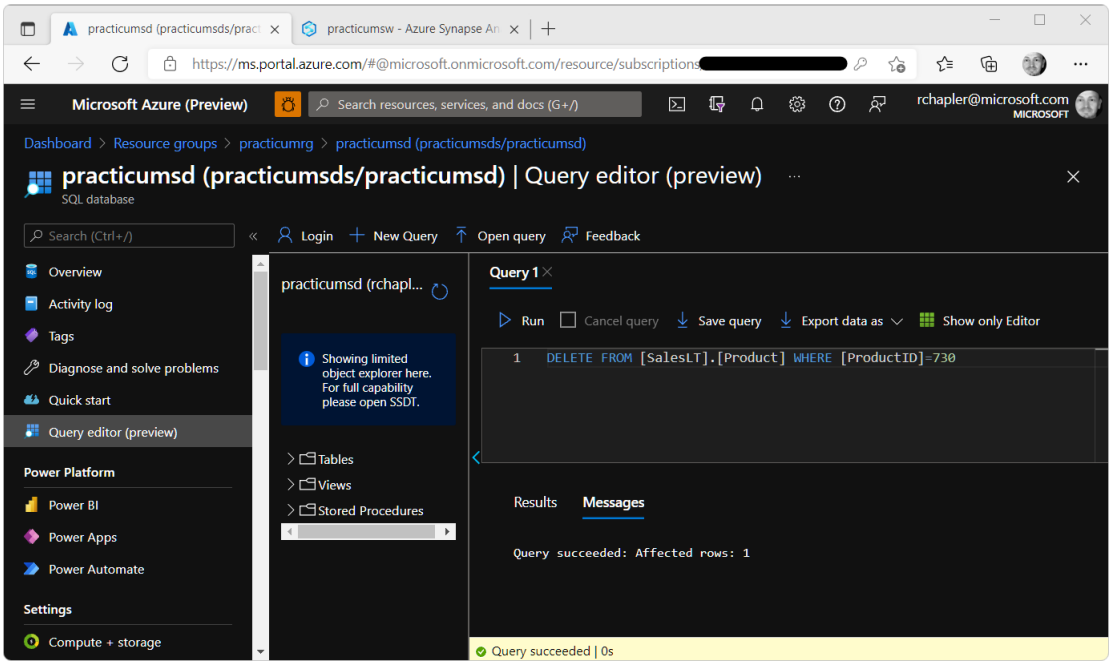
Note: Because we are doing an incremental load immediately after an initial load, on a sample database which doesn’t have unexpected change, we see no current results in the Data Preview.

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

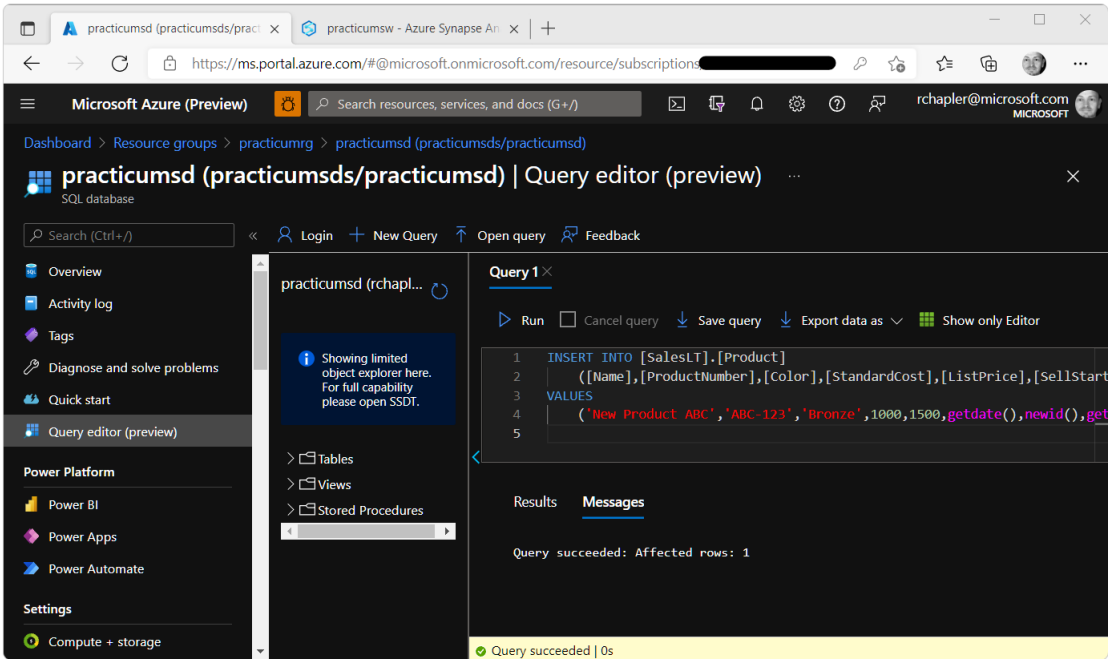


Execute the following T-SQL:

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

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

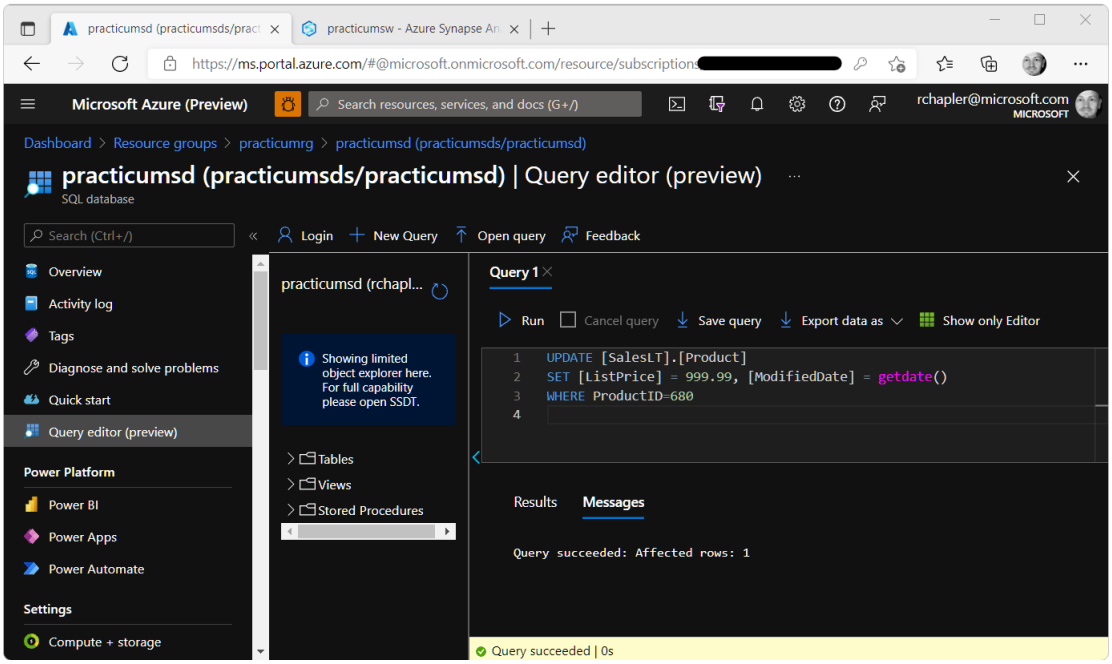
Insert Example



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

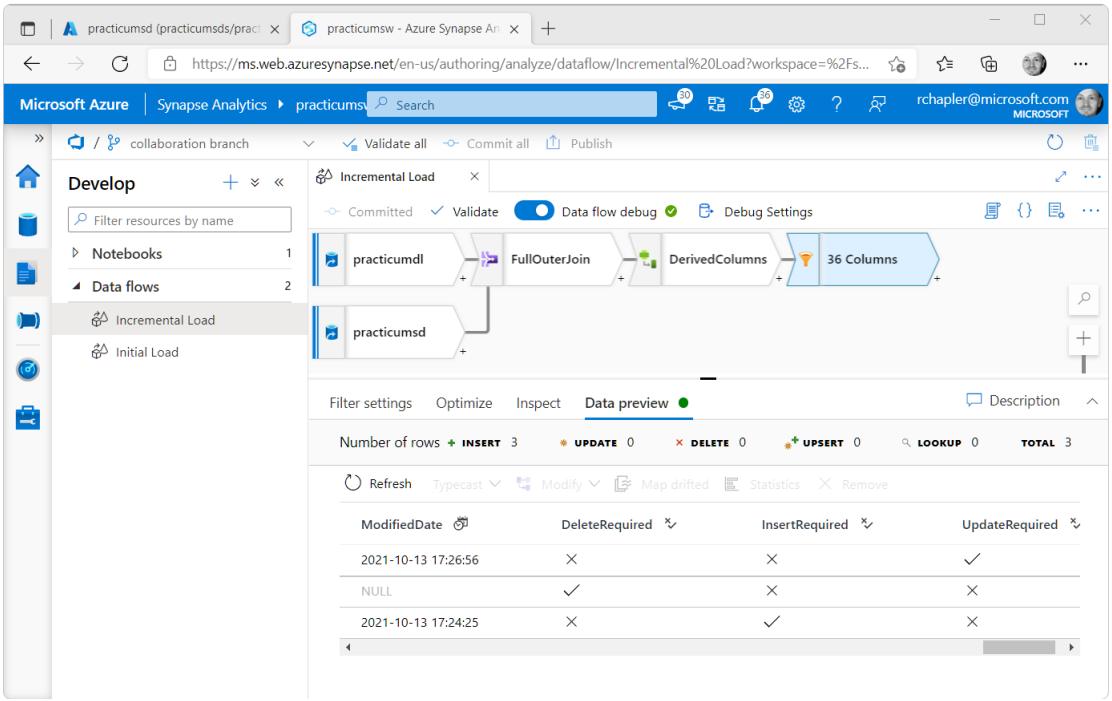


Execute the following T-SQL:

```
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**.

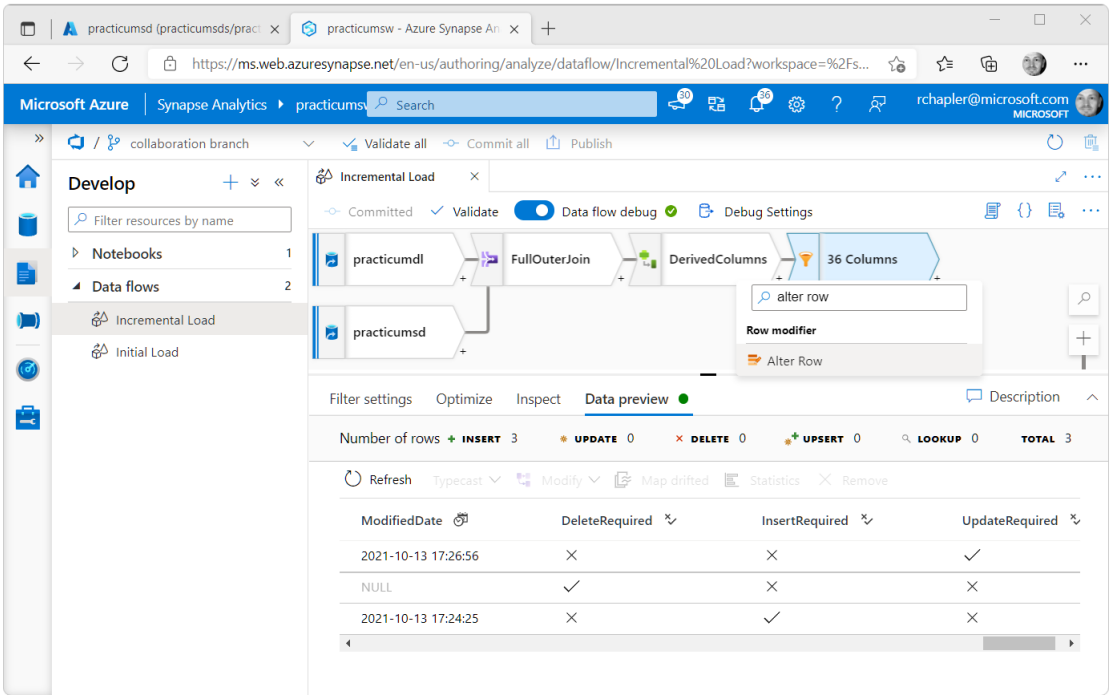


After update, you should see the three actions listed in **Filter** > "**Data preview**".

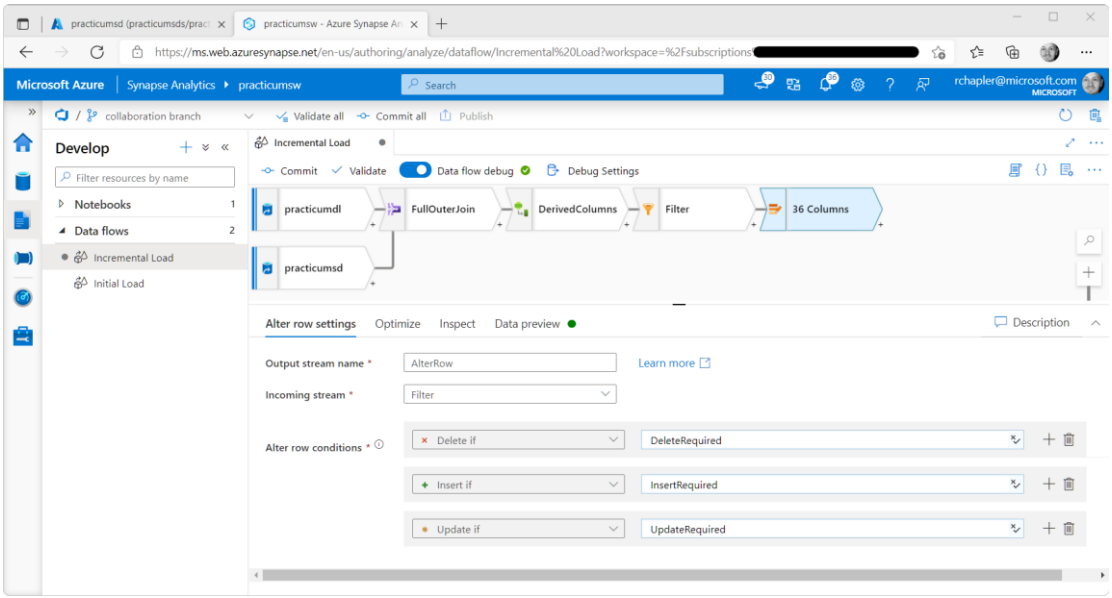
Finalize Data Flow

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 popup list.



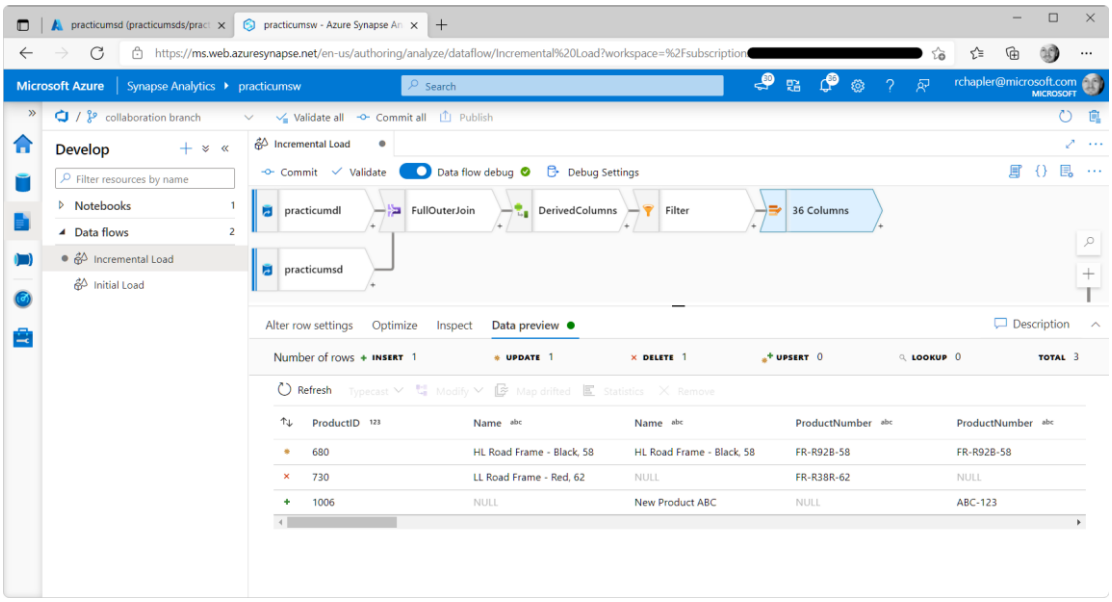
On the "**Alter row settings**" tab, enter values for the following items:

Output Stream Name	Enter a meaningful name aligned with standards

Incoming Stream	Confirm default selection

Alter Row Conditions	“Delete if”
	`DeleteRequired`
	
	“Insert if”
	`InsertRequired`
	
	“Update if”
	`UpdateRequired`

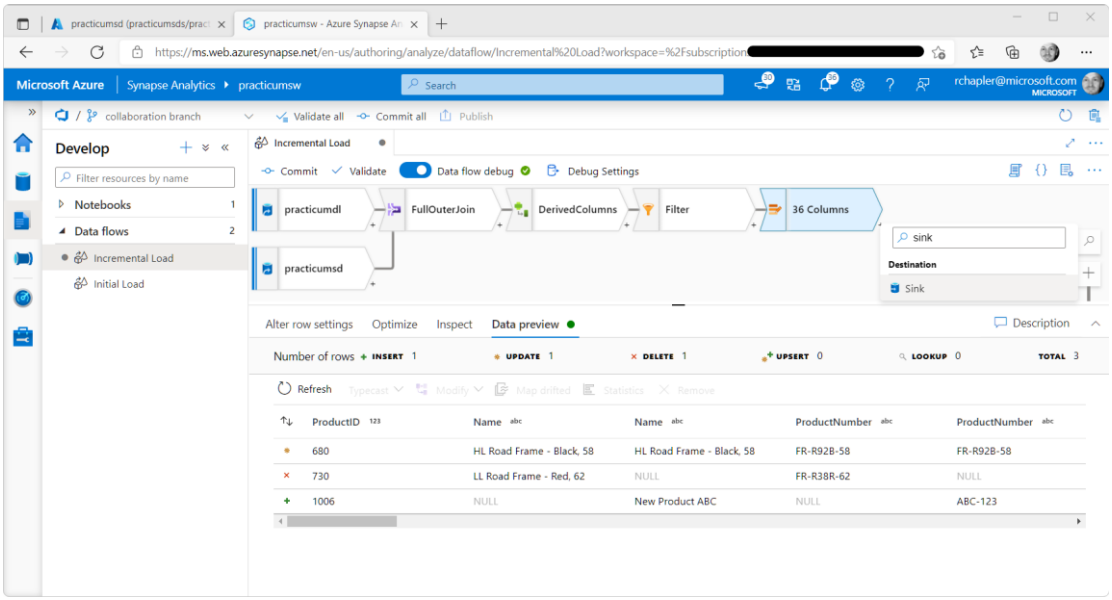
Navigate to the “**Data preview**” tab and then click **Refresh**.



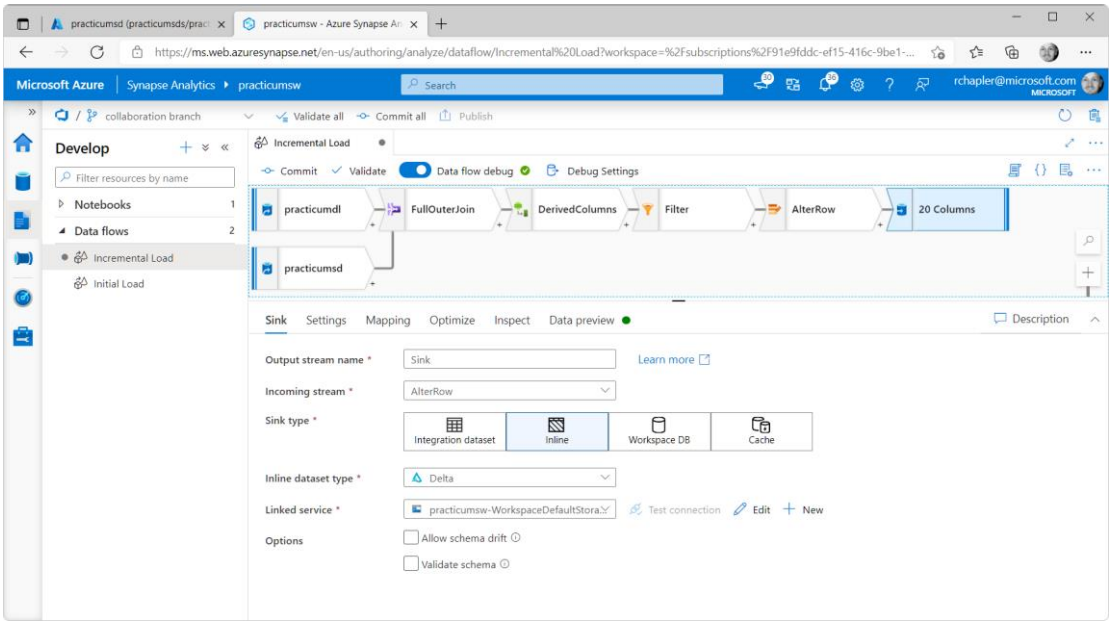
Click the **Commit** button.

Sink (delta lake)

In this step, we finalize the data flow.



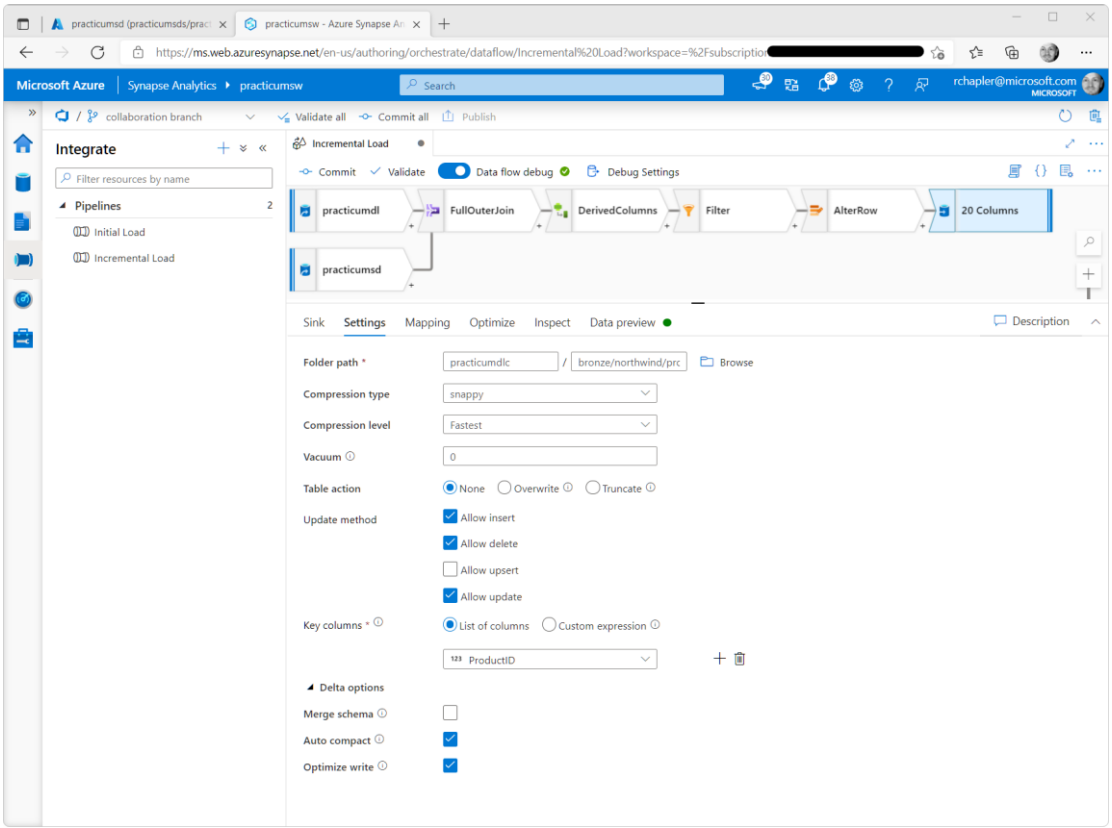
Click the **+** in the bottom right of the SQL source, and then select **Sink** from the resulting popup list.



On the **Sink** tab, enter values for the following items:

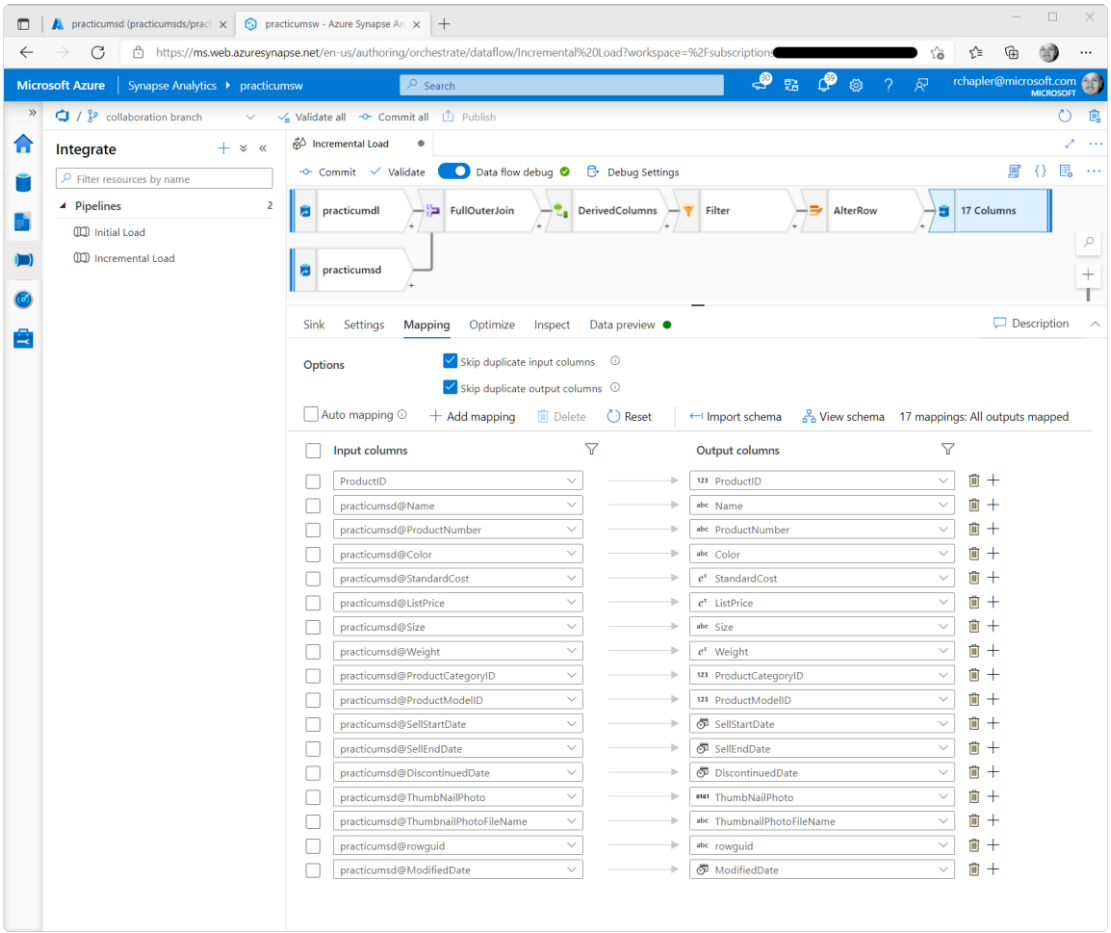
Output Stream Name	Enter a meaningful name aligned with standards
Sink Type	Select **Inline**
Inline Dataset Type	Select **Delta**

Linked Service	Select practicumsw-WorkspaceDefaultStorage



On the **Settings** tab, enter values for the following items:

File System	Enter shareddlc
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

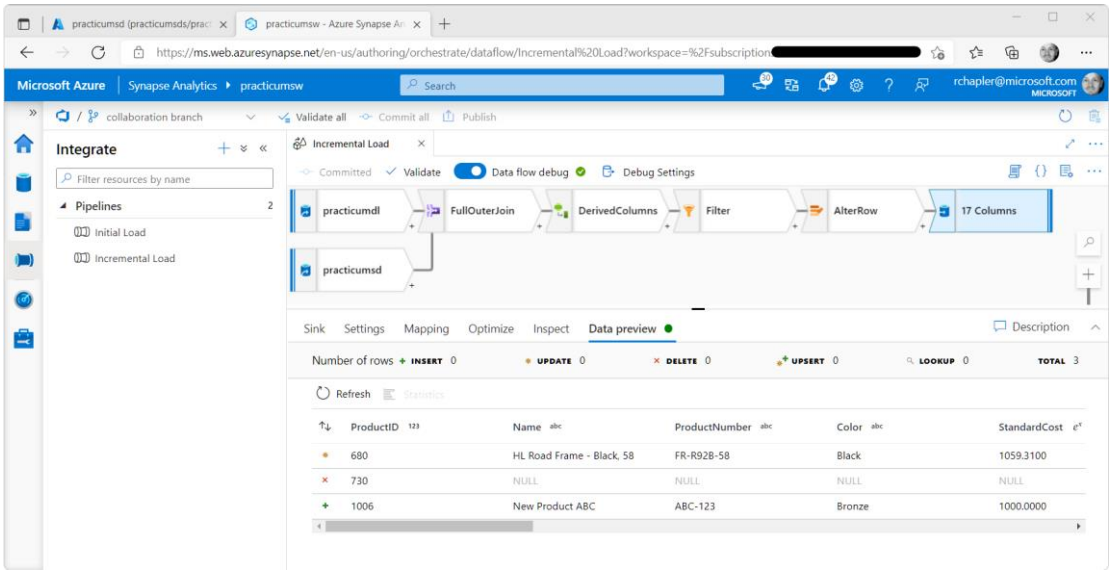


On the **Mapping** tab, un-check “Auto mapping”, click “**Import schema**” and then **Import** on the resulting popout.

Review the auto-generated list of columns and remove:

- Rows where “**Input columns**” begin with “**shareddl**…” (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.

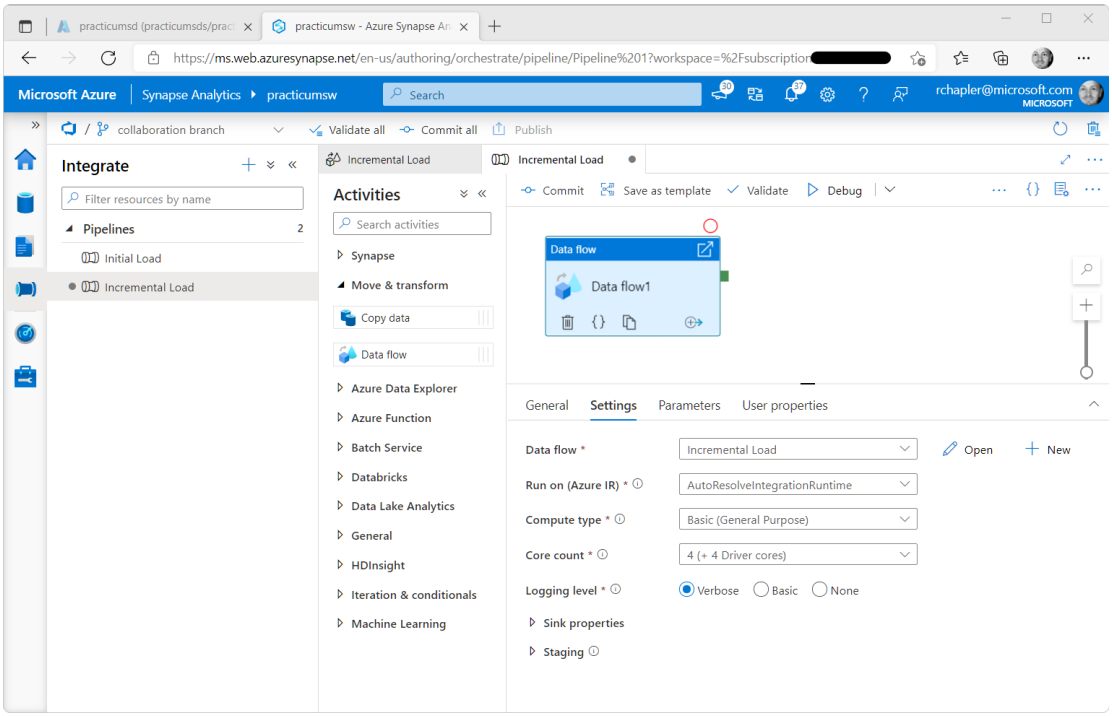
Create Pipeline

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 dropdown.

On the **Properties** popout, 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.

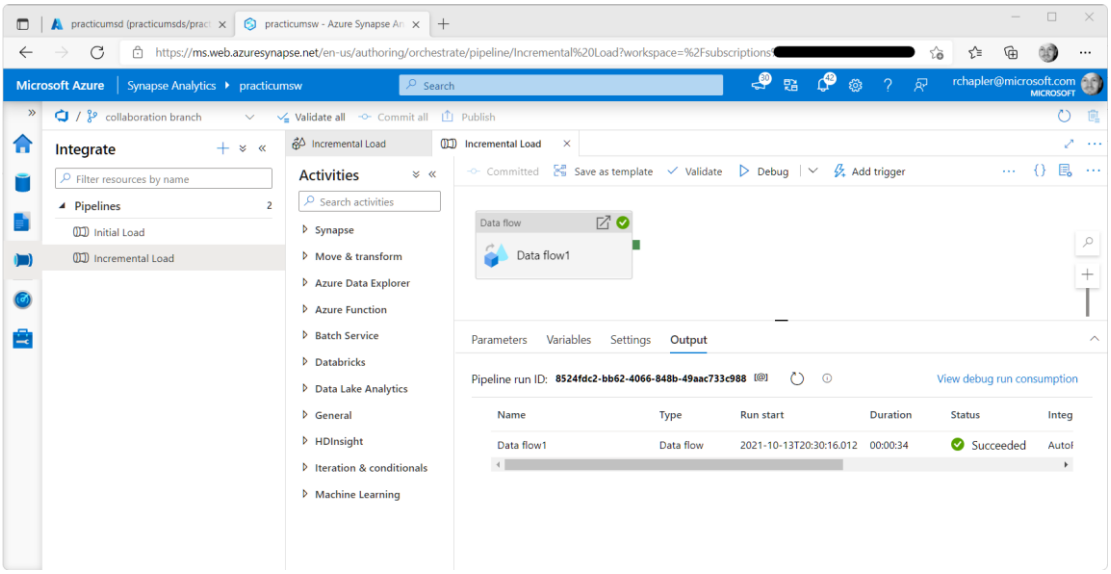
No additional changes are required. Review settings on the remaining tabs.

Click the **Commit** button.

Confirm Success

Debug Pipeline / Data Flow

Click **Debug** and confirm successful execution.

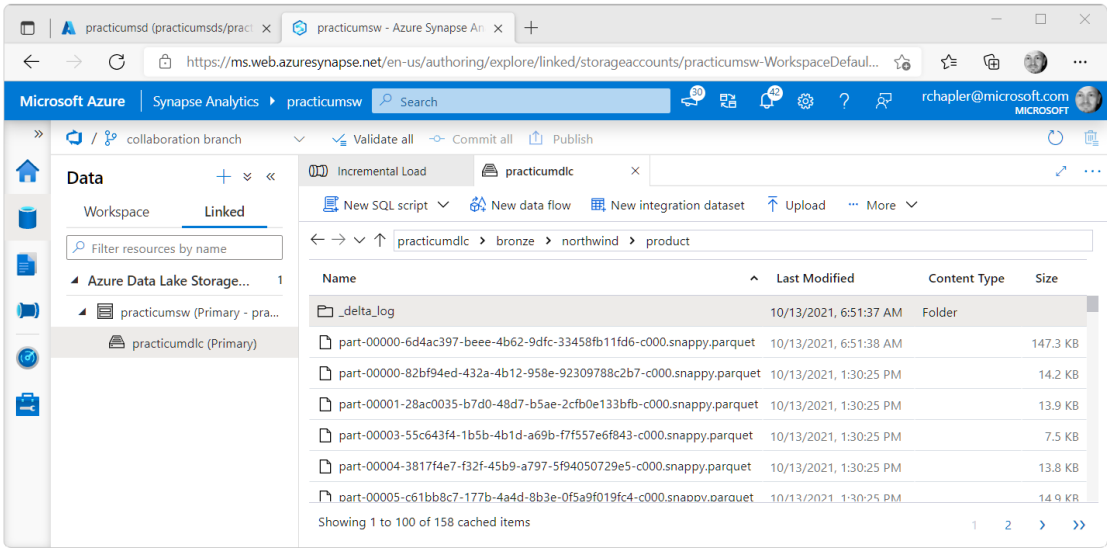


Review Produced Files

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

Expand “Azure Data Lake Storage Gen2”, then practicumsw to shareddlc.

Click to navigate to “shareddlc > bronze > northwind > product”



Review the files produced by delta lake.

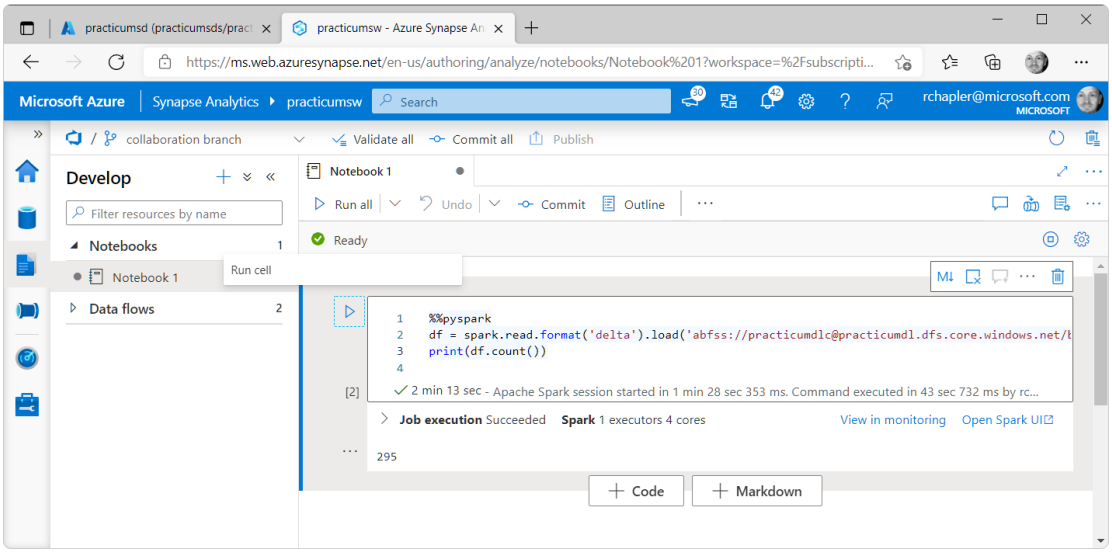
Query Record Count

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 dropdown.

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

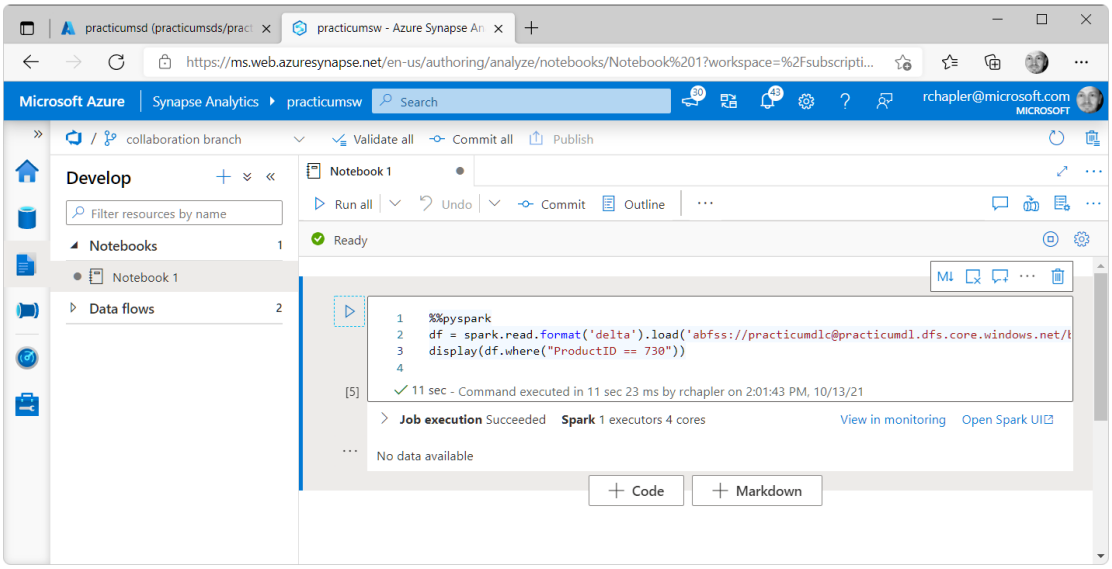


Run the following code to produce a count of records in the new Delta Lake:

```
%%pyspark
df = spark.read.format('delta').load('abfss://shareddlc@shareddl.dfs.core.windows.net/bronze/northwind/product')
print(df.count())
```

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

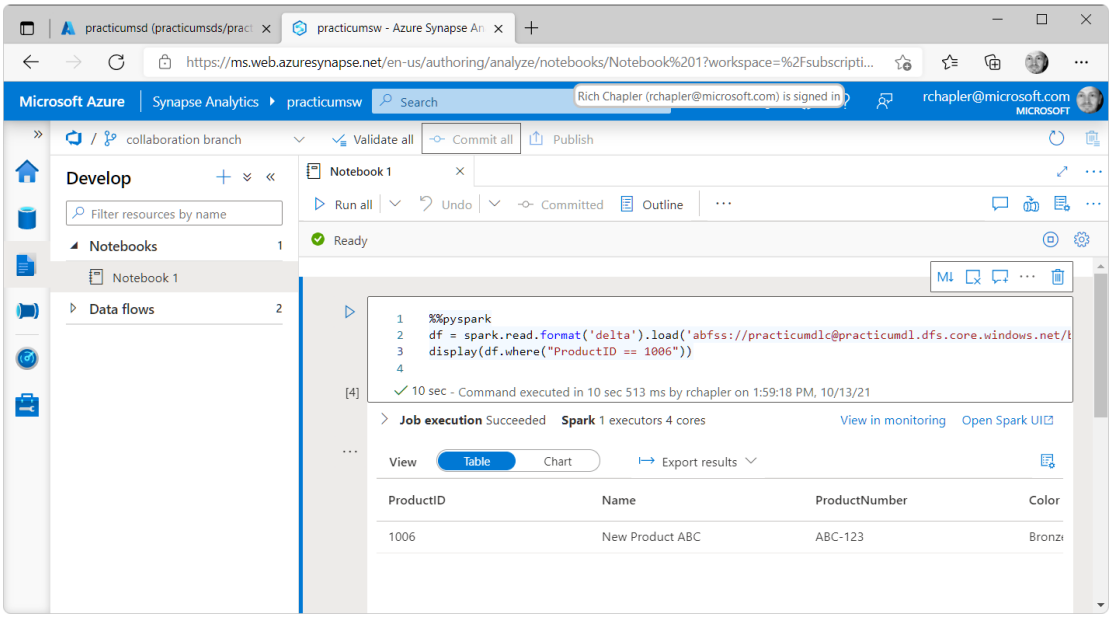
Confirm Delete Example



Run the following code to confirm our update example:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://shareddlc@shareddl.dfs.core.windows.net/bronze/northwind/produc
t')
display(df.where("ProductID == 730"))
```

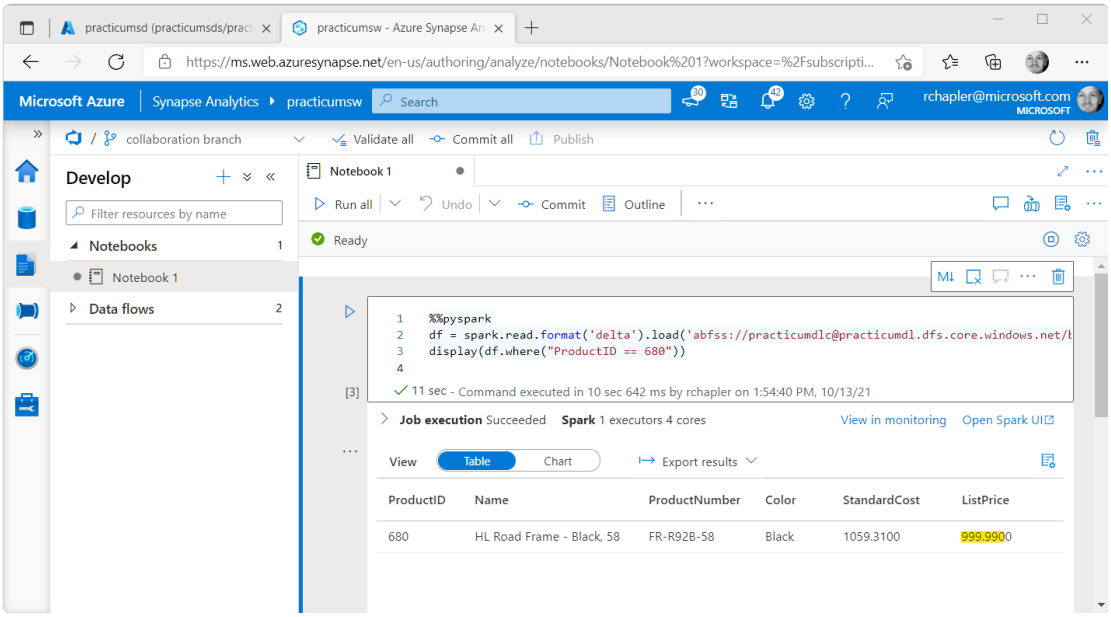
Confirm Insert Example



Run the following code to confirm our update example:

```
%%pyspark
df =
spark.read.format('delta').load('abfss://shareddlc@shareddl.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://shareddlc@shareddl.dfs.core.windows.net/bronze/northwind/product')
display(df.where("ProductID == 680"))
```

Objective Complete... congratulations!

Mount Data Lake

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

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

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

Step 2: Prepare Logic

Navigate to Databricks and the **practicumdbn** notebook.

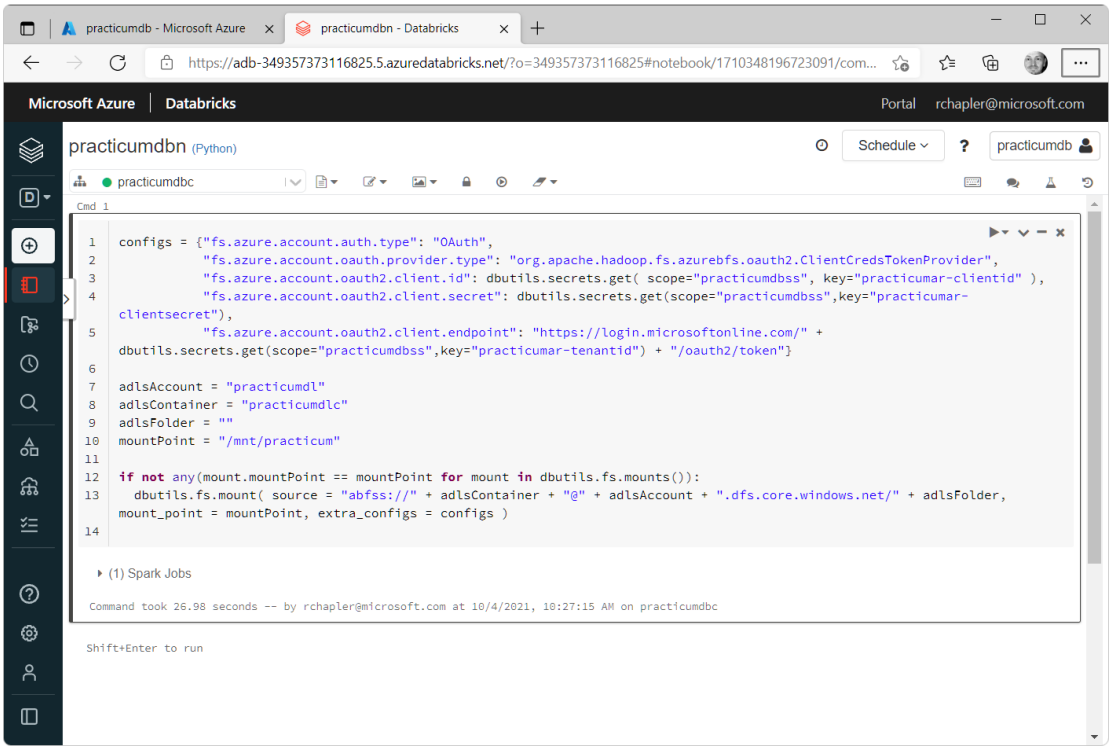
Add a new cell and paste the following code:

```
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="practicumdbss", key="
practicumar-clientid" ),
      "fs.azure.account.oauth2.client.secret": dbutils.secrets.get( scope="practicumdbss",
key="practicumar-clientsecret" ),
      "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/" +
dbutils.secrets.get( scope="practicumdbss", key=" practicumar-tenantid" ) + "/oauth2/token"}

adlsAccount = "shareddl"
adlsContainer = "shareddlc"
adlsFolder = ""
mountPoint = "/mnt/practicum"

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 )
```

Some callouts…

- **practicumdbss** … 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”
- **shareddl** … refers to your Data Lake
- **shareddlc** … 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/practicum/myDatabase"
```

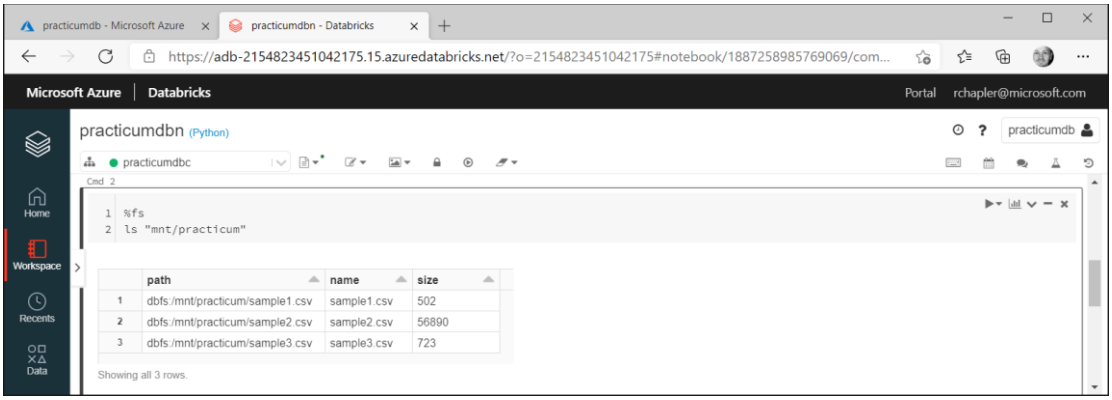
Step 3: Confirm Success

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/practicum"
```

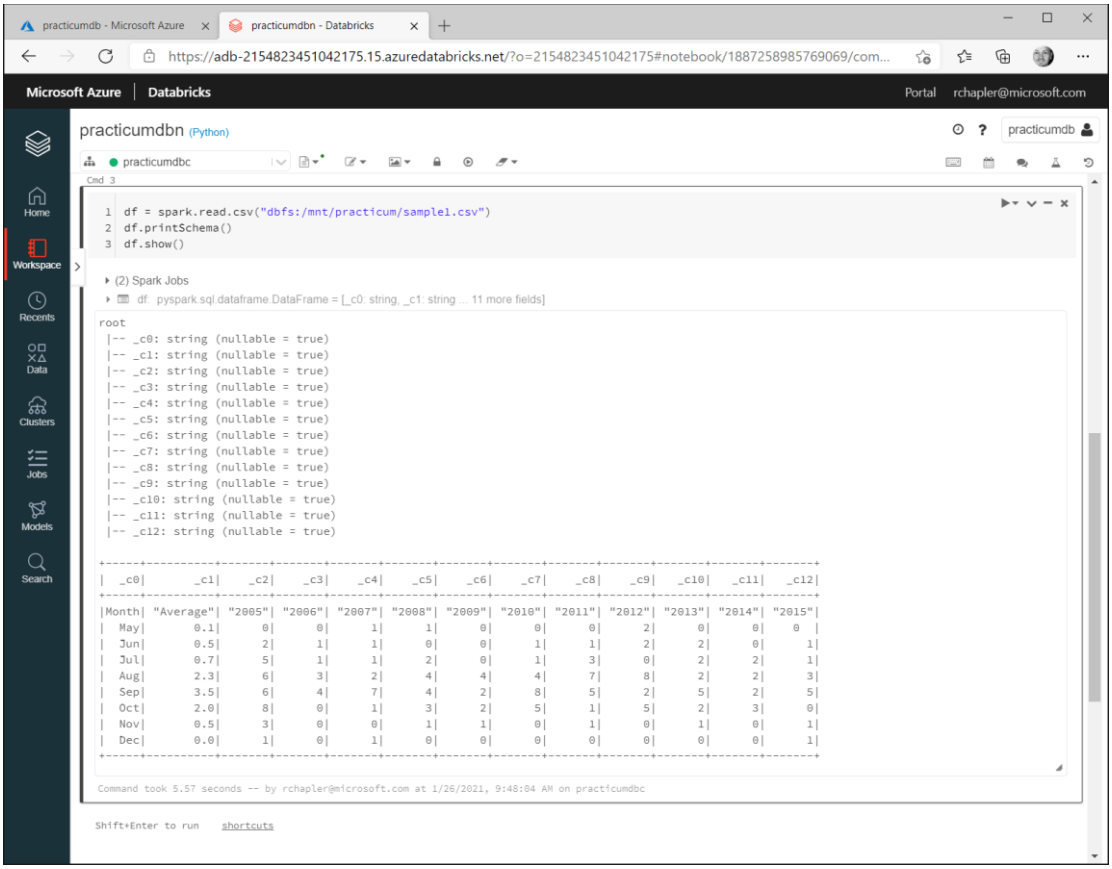


You should see your sample files at the mount point.

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/practicum/sample1.csv")
df.printSchema()
df.show()
```



Some 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.

Objective Complete… congratulations!

Localize Mounted Data

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

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Data Lake
- Databricks

Step 2: 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

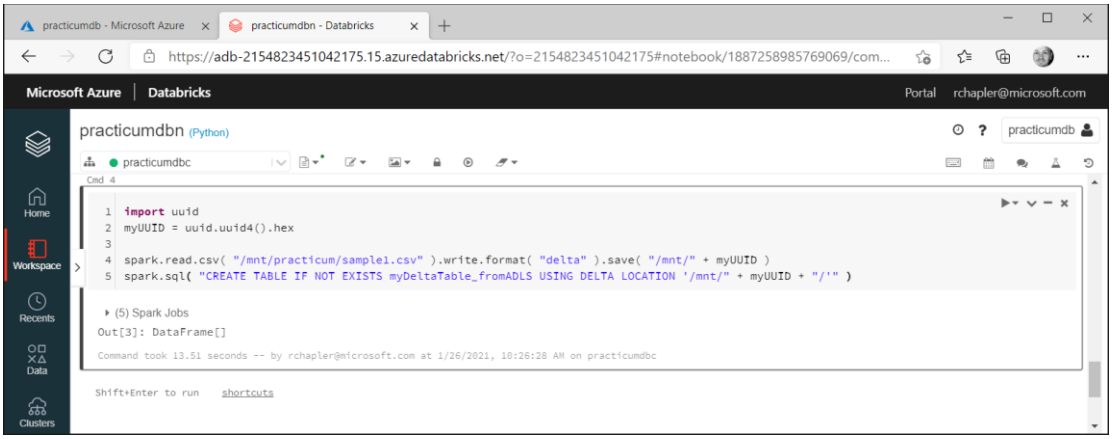
Though it is not listed here, many customers refer to the following benefits when justifying the decision to localize data from Azure Data Lake Storage to Azure Databricks:

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

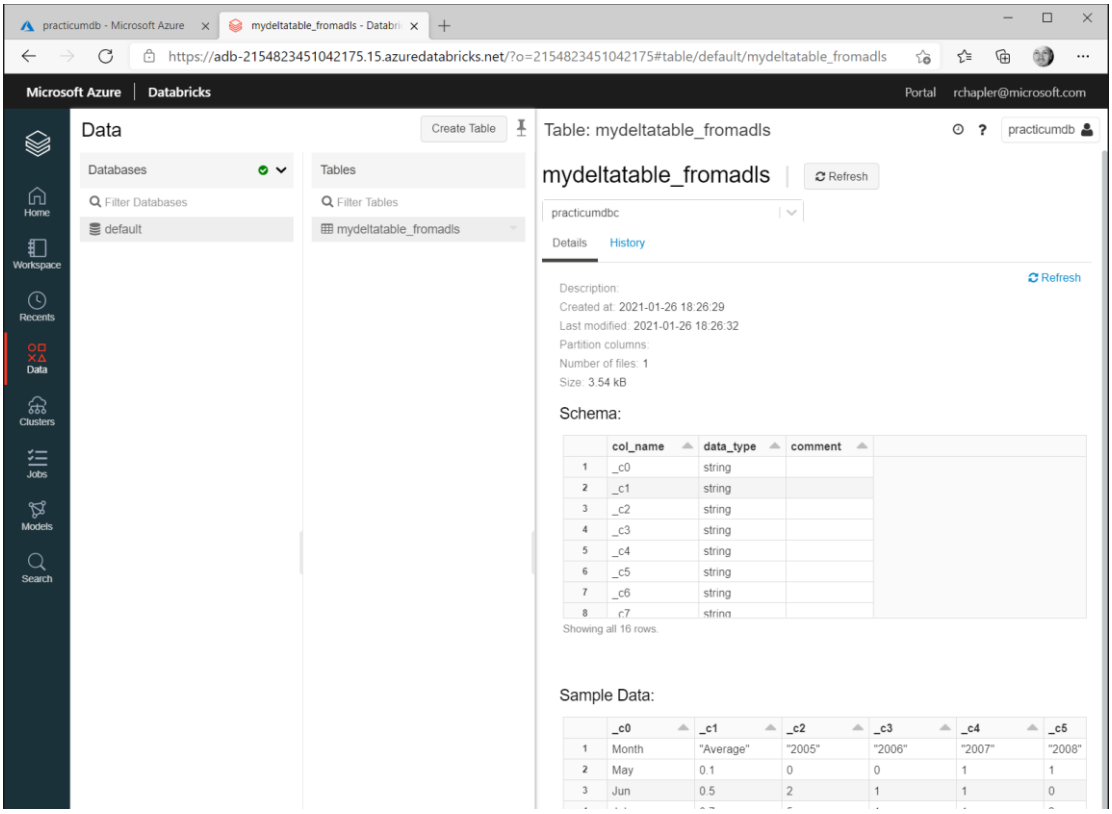
The Logic

Navigate to Databricks and the **practicumdbn** notebook. Add a new cell and paste the following code:

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

spark.read.csv( "/mnt/practicum/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.

Objective Complete... congratulations!

Sourcing from APIs

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

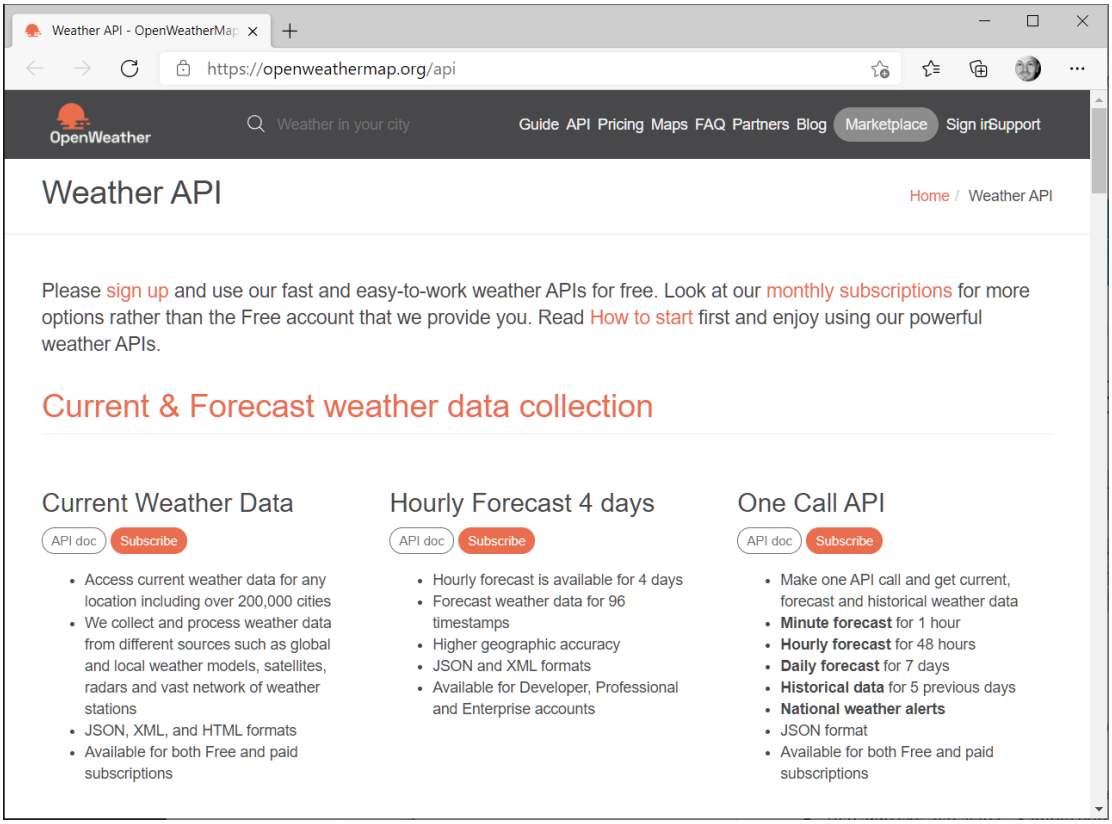
Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Databricks
- Key Vault

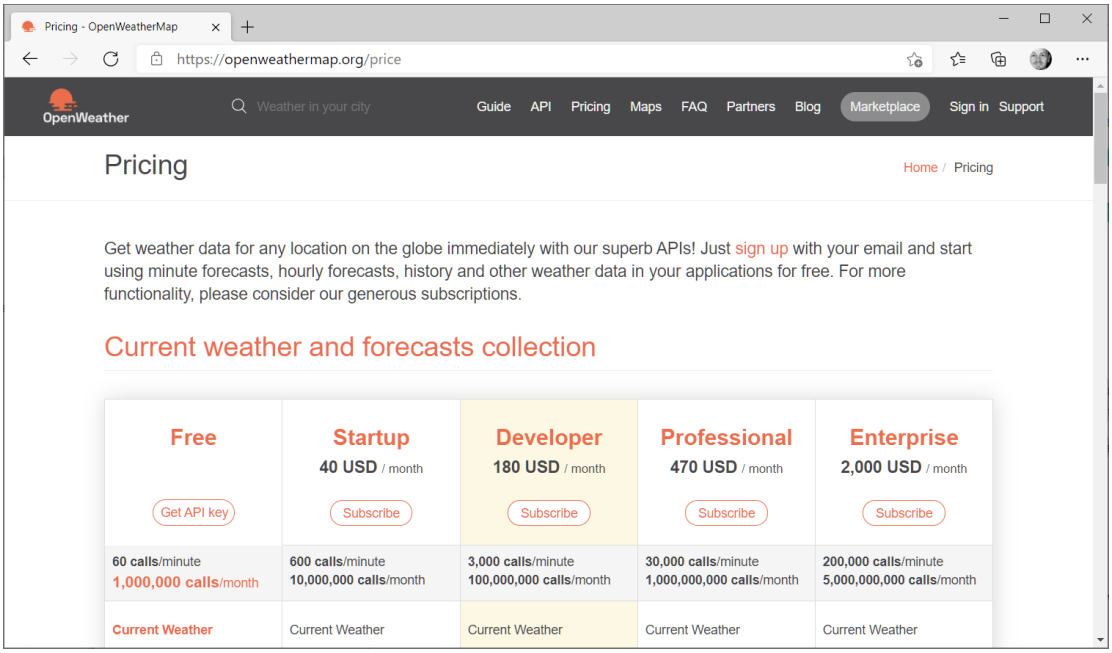
Step 2: Stage Resources

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.

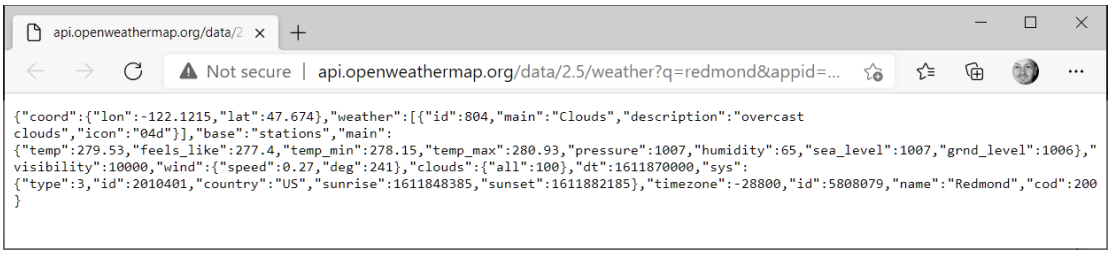
Navigate to https://openweathermap.org/price.



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

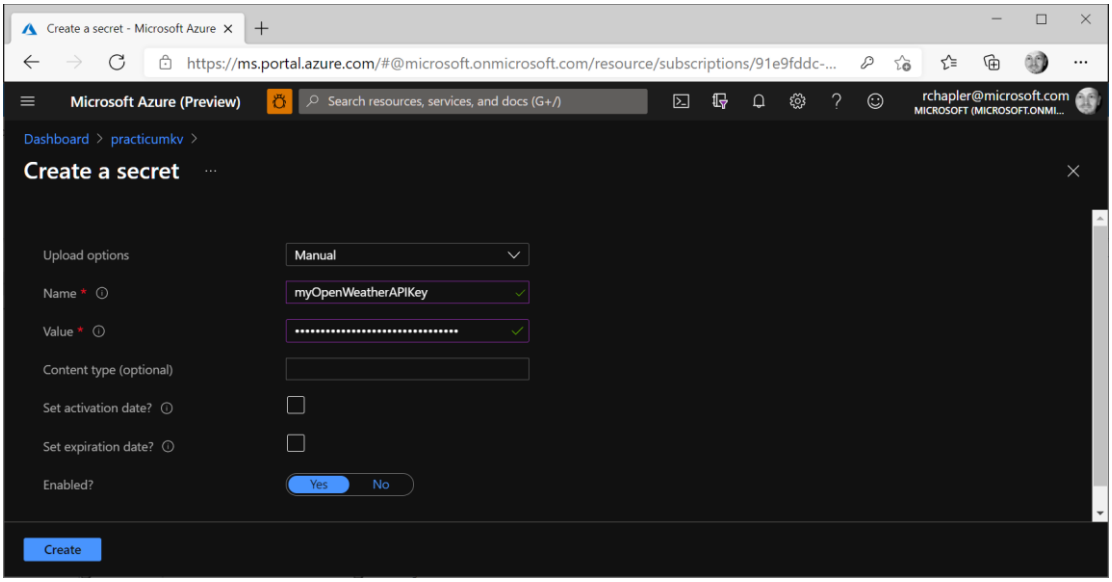
When you're successfully created an account, making an API call for "**Current Weather**" data is as simple as:

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



Step 3: Add Secret to Key Vault

Add your API Key to Key Vault.

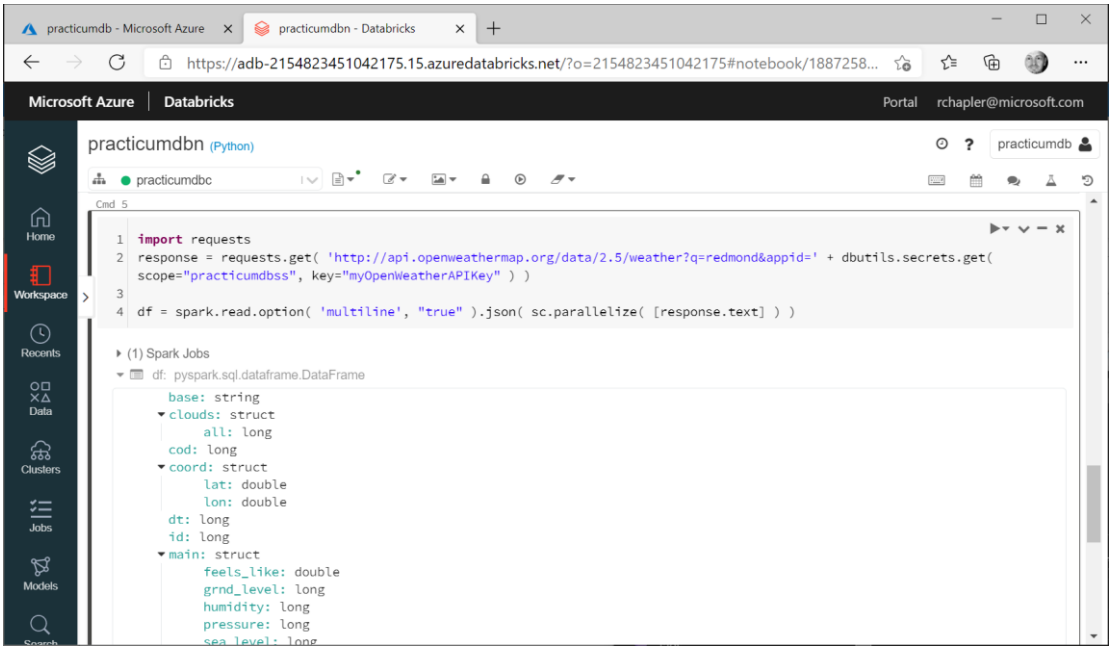


Step 4: Prepare Logic

Navigate to Databricks and the **practicumdbn** 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="practicumdbss", 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()
```

Objective Complete... congratulations!

Batch Upsert

Follow these instructions to provide for recurring upsert of data.

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Data Lake
- Databricks

Step 2: CREATE TABLE

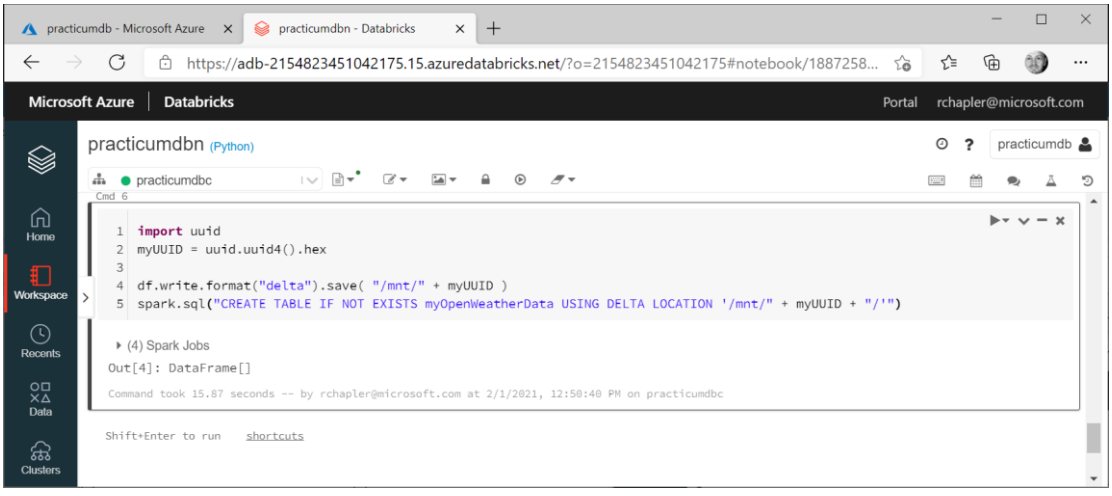
Navigate to Databricks and the **practicumdbn** notebook.

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.

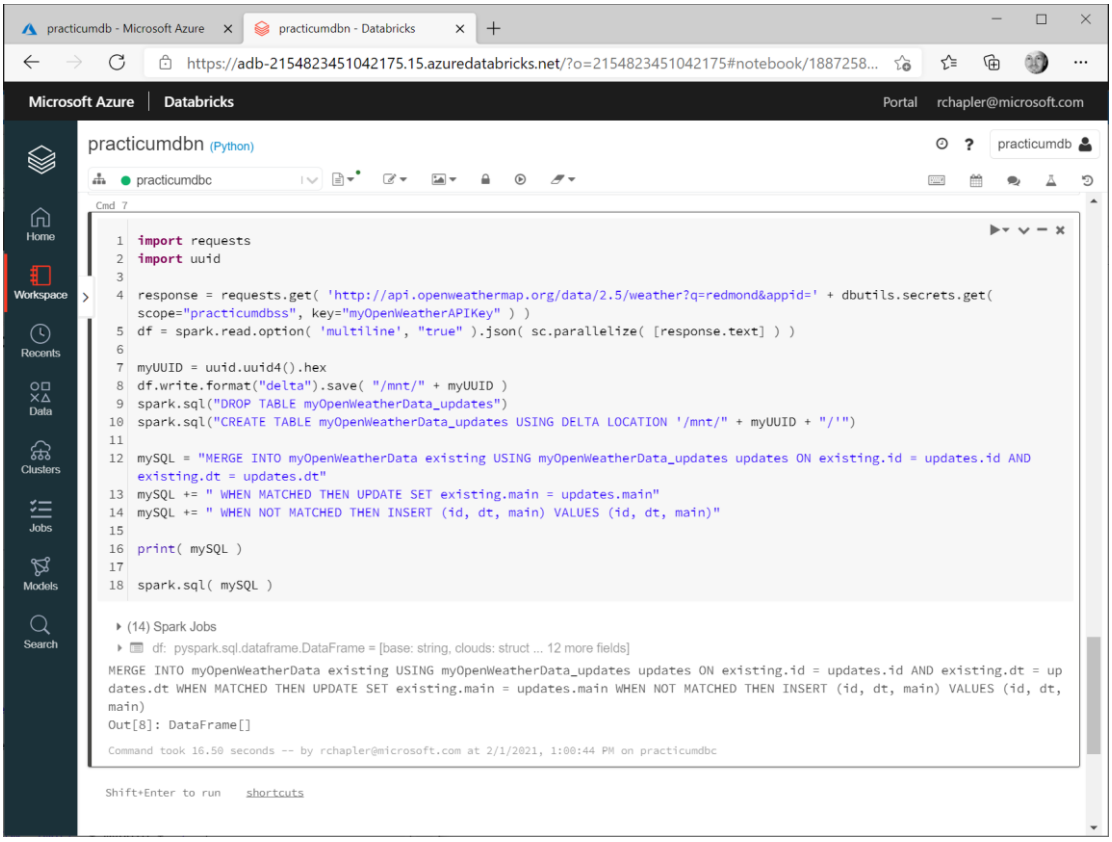


Step 2: MERGE INTO

Navigate to Databricks and the **practicumdbn** 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="practicumdbss", 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.

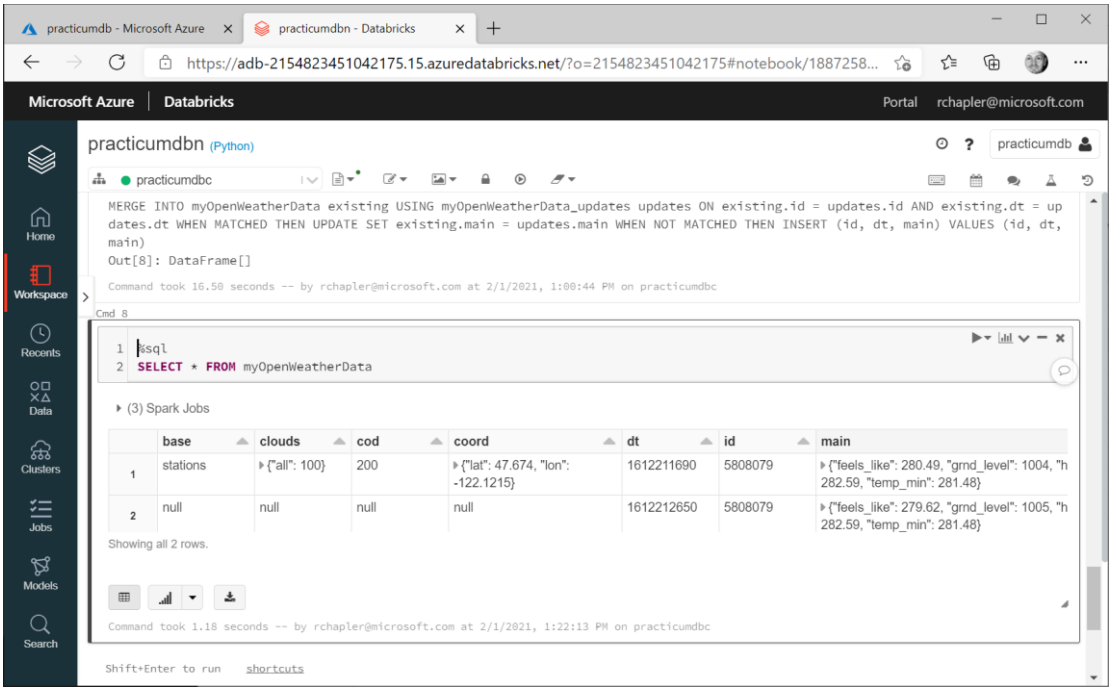


Step 3: Confirm Success

Navigate to Databricks and the **practicumdbn** 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.

Objective Complete… congratulations!

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**

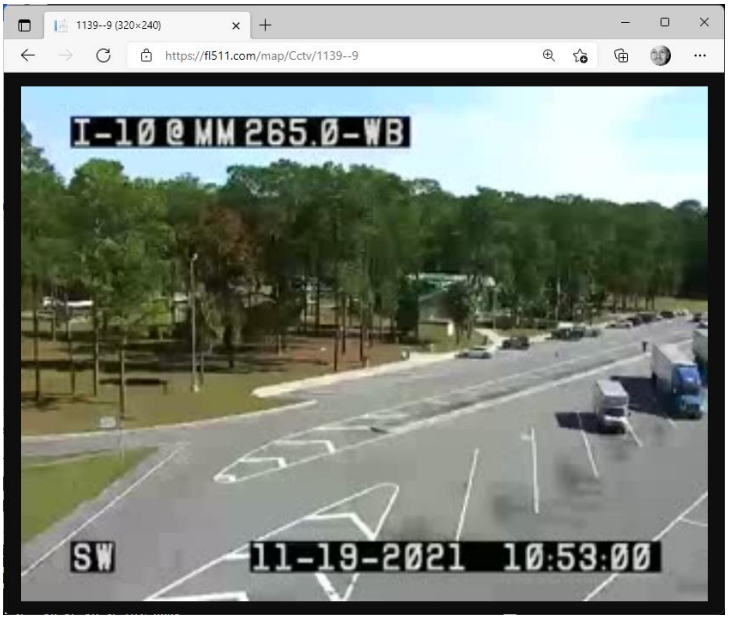
Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- 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

Step 2: 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.

Step 1: 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}
```

Step 2: Create Images Directory

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

```
mkdir images
```

Step 3: 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
```

Step 3: 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

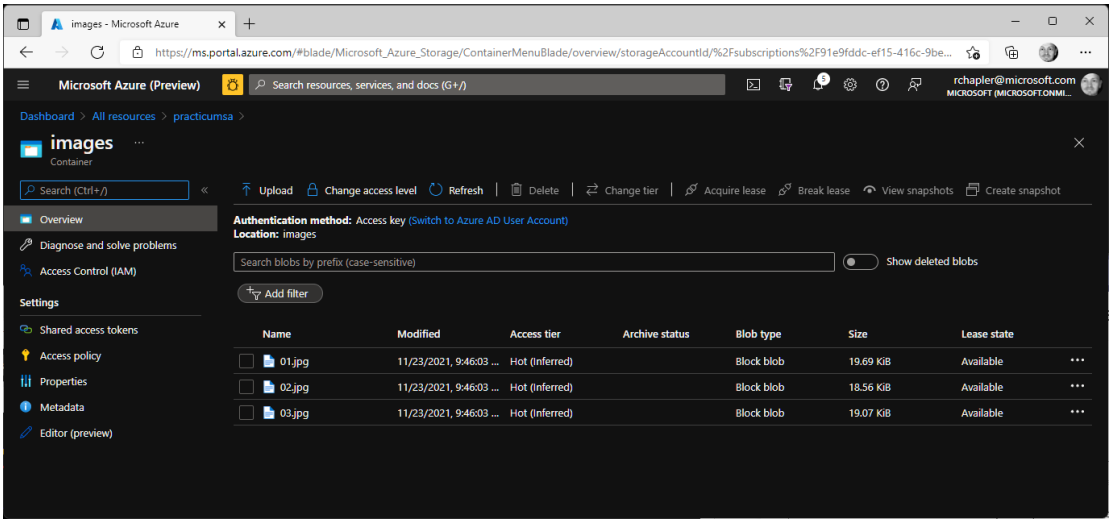
Update and execute the following command to test file synchronization using AzCopy:

```
./azcopy_linux_amd64_10.13.0/azcopy sync "/home/rchapler/images"
"https://practicumsa.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.



Step 4: Prepare Script

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

Create Script with Vim

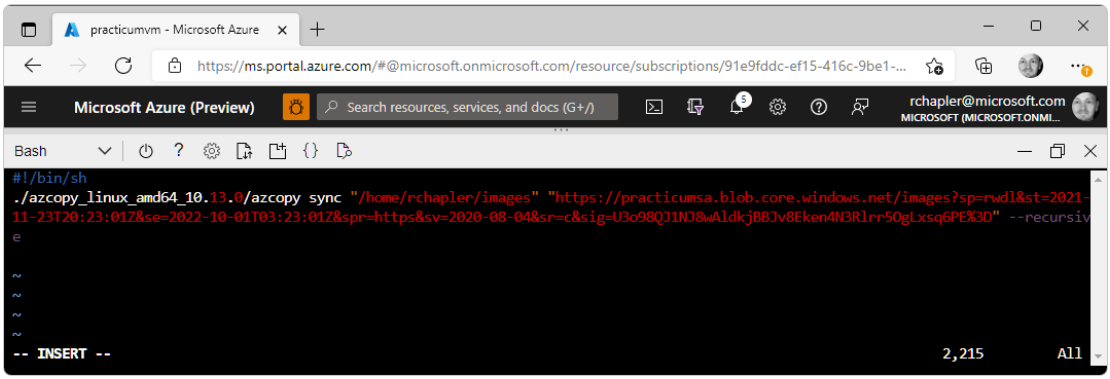
Execute the following command to open a text editor:

```
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://practicumsa.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
```

Step 5: 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
```

Logic explained…

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

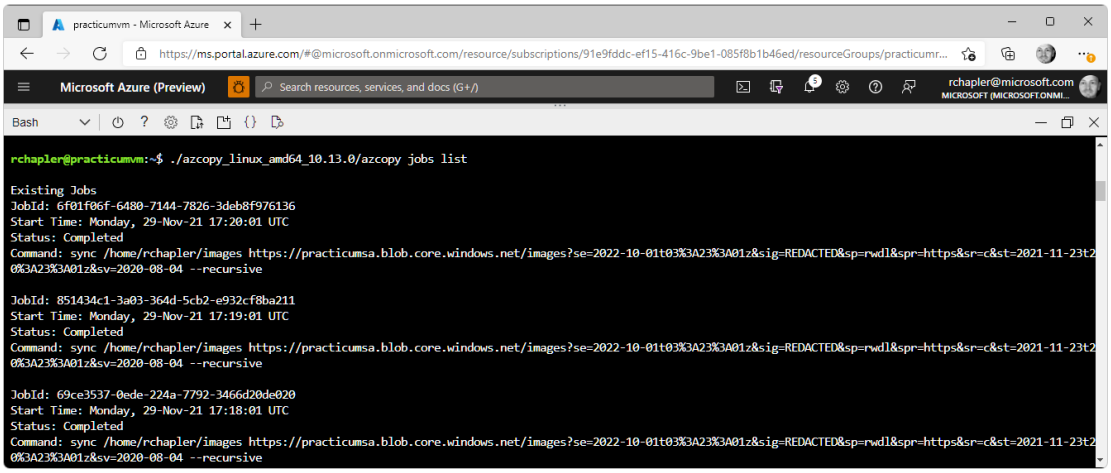
Press CTRL-X on the keyboard to exit crontab; save when prompted.

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}.

Step 6: Confirm Synchronization

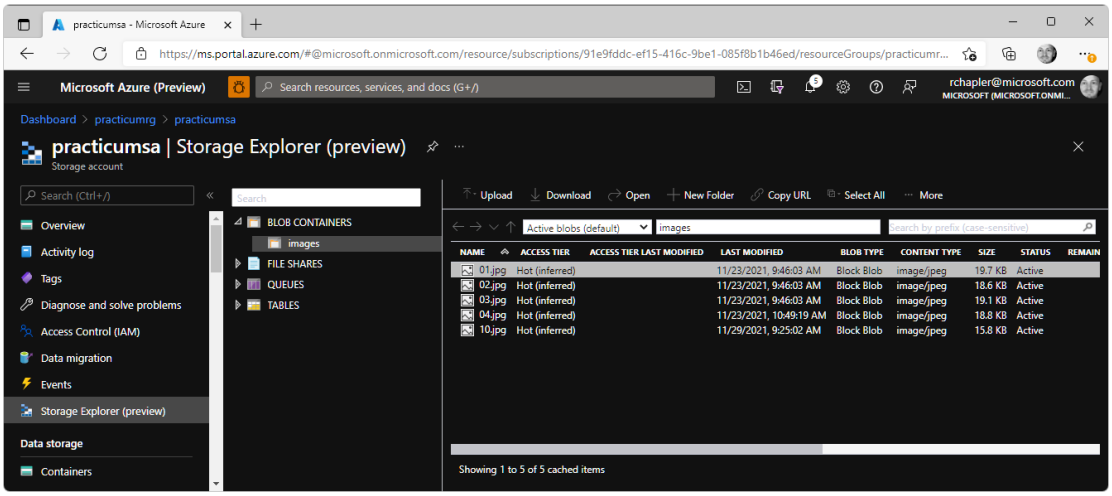
Execute the following command to get a new sample image from the traffic cam source:

```
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.

Objective Complete… congratulations!

On-Prem to On-Prem

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

Step 1: Understand 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}

Step 2: Instantiate Prerequisites

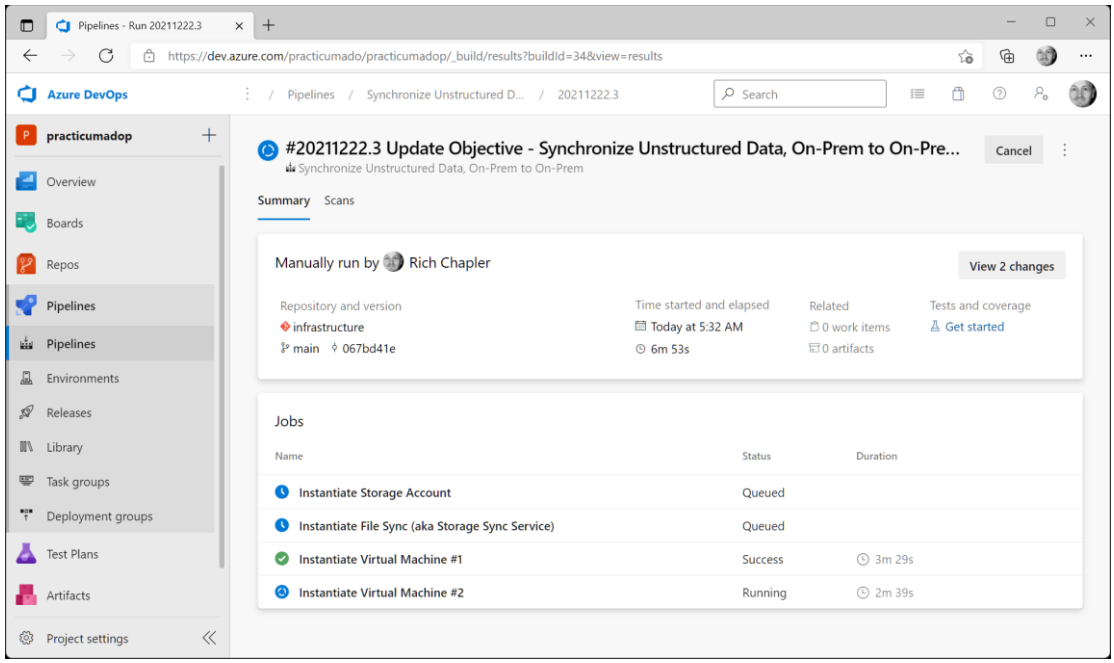
This exercise requires the following resource(s):

- 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 Deploy Infrastructure-as-Code (feat. DevOps + ARM Templates) instructions 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: practicum
   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: practicum
   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: practicumvm01'
  inputs:
   ConnectedServiceName: a775d3ec-e4f1-46ed-905b-ddd26a59356c
   subscriptionName: 91e9fddc-ef15-416c-9be1-085f8b1b46ed
   resourceGroupName: practicumvm01
   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: practicumvm02'
  inputs:
   ConnectedServiceName: a775d3ec-e4f1-46ed-905b-ddd26a59356c
   subscriptionName: 91e9fddc-ef15-416c-9be1-085f8b1b46ed
   resourceGroupName: practicumvm02
   location: West US
   csmFile: ARM Templates/VirtualMachine.json
```



Step 3: Install Agent

Navigate to the first virtual machine, click the **Connect** button, and select **RDP** from the resulting dropdown.

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

Click the "**Open file**" link under your RDP file in the **Downloads** dropdown.

Click the **Connect** button on the resulting "**Remote Desktop Connection**" popup.

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

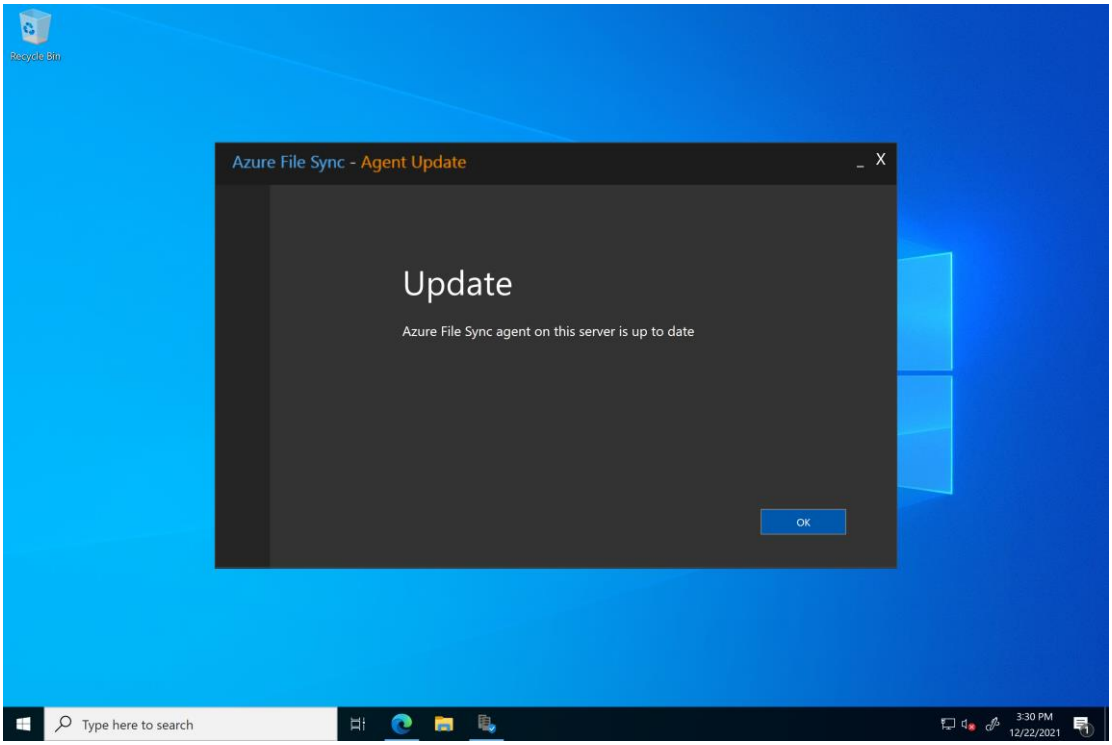
Back at the top of the "**Windows Security**" popup, enter credentials included in the YAML template.

Note: "User name" should include the name of the server {e.g., "practicumvm01\Admin123"}.

Once connected to the virtual machine, browse to Download Azure File Sync Agent from Official Microsoft Download Center.

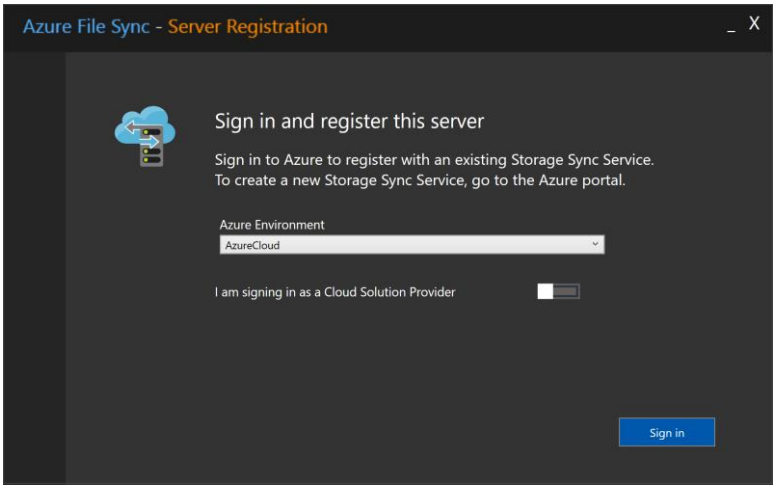
Click the **Download** button, and select the appropriate download {e.g., **StorageSyncAgent_WS2022.msi**} and click the **Next** button.

Open the downloaded file and complete the installation.



The Azure File Sync, server registration utility will open.

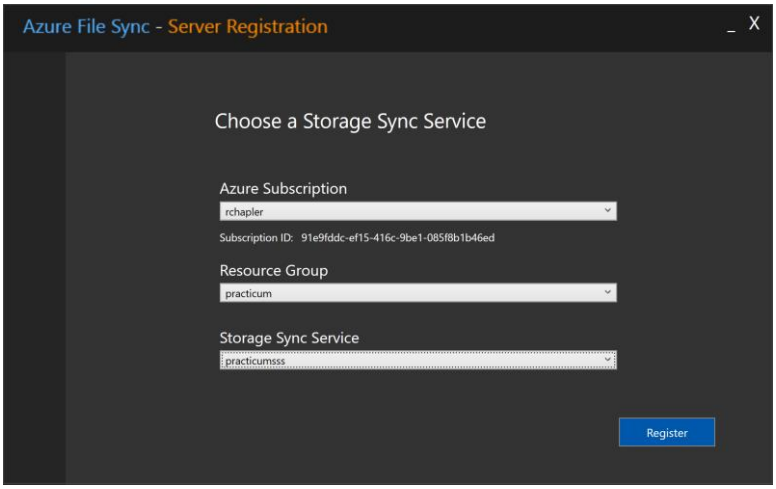
Click the OK button on the Update page.



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.



On the "**Choose a Storage Sync Service**" page, enter values for the following items:

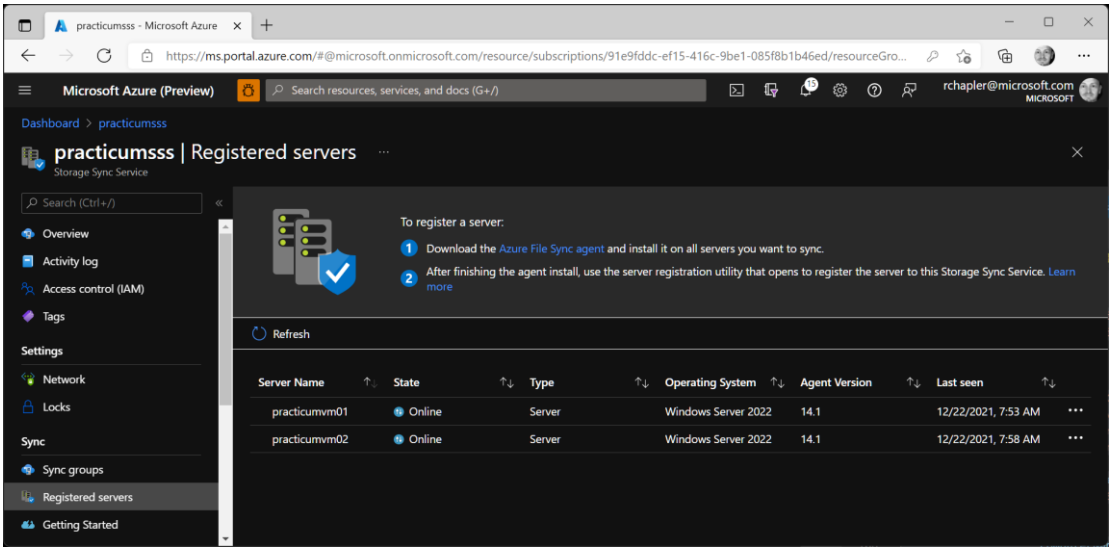
Azure Subscription	Select your subscription
Resource Group	Select your resource group
Storage Sync Service	Select your storage sync service

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 bar. You should see your two registered servers.

Step 4: Create File Share

Open your Storage Account, then "**File Shares**" in the "**Data storage**" group of the navigation bar.
Click the "**+ File share**" button and enter a meaningful **Name** value in the resulting popout.
Click the **Create** button.

Step 5: Create Sync Group

Navigate to Storage Sync Service, then "**Registered Servers**" in the **Sync** group of the navigation bar.
Click the "**+ Sync group**" button.

On the "**Choose a Storage Sync Service**" page, enter values for the following items:

Sync Group Name	Enter a meaningful name aligned with standards {e.g., practicumssssg}
Subscription	Select your subscription
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.

Step 6: 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**" popout, enter values for the following items:

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.

Step 7: 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 dropdowns.

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.

Objective Complete… congratulations!

Surface

Query from On-Prem

Follow these instructions to **query from an on-prem client**.

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Azure CLI
- Data Explorer
- Postman

Step 2: 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.

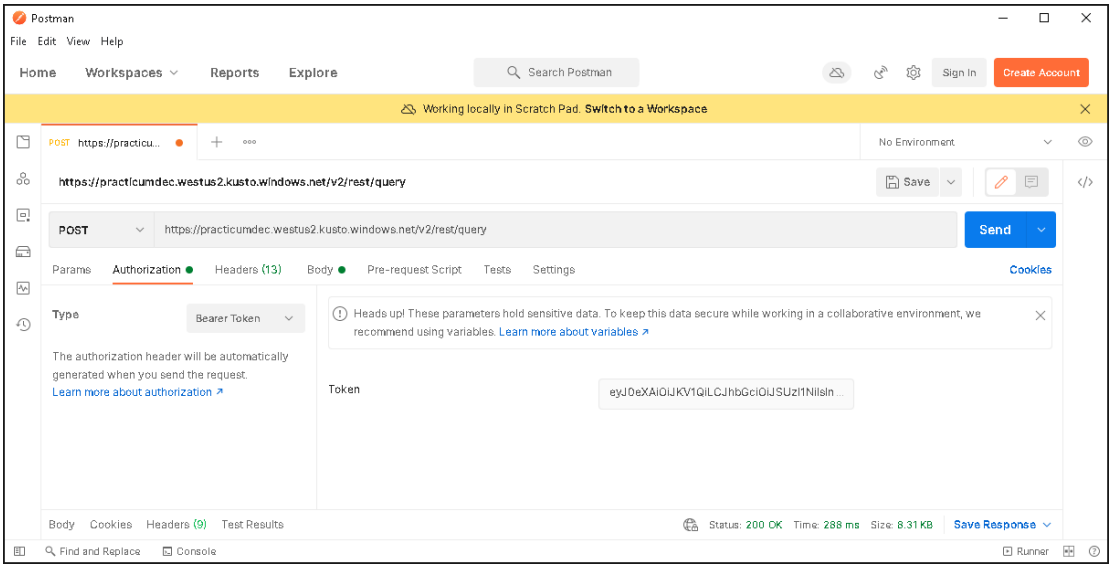
In a Command Prompt window (with Azure CLI installed), execute the following command: `az account get-access-token --resource https://practicumdec.westus2.kusto.windows.net --query accessToken --output tsv`

Capture the returned token for use later in this section.


Command Prompt
C:\Users\rchapler>az account get-access-token --resource https://practicumdec.westus2.kusto.windows.net --query accessToken --output tsv
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Im5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyIsImtpZCI6Im5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyJ9.eyJhdWQiOiJodHRwczovL3ByYWN0aWN1bWRlYy53ZXN0dXMyLmt1c3RvLndpbmRvd3MubmV0IiwiaXNzIjoiaHR0cHM6Ly9zdHMud2luZG93cy5uZXQvNzJmOTg4YmYtODZmMS00MWFmLTkxYWItMmQ3Y2QwMTFkYjQ3LyIsImlhdCI6MTYxODk1NjI1NCwibmJmIjoxNjE4OTU2MjU0LCJleHAiOjE2MTg5NjAxNTQsImFjciI6IjEiLCJhaW8iOiJBVVFBdS84VEFBQUF5YlNLemdTcmVRNmFrcmhlLy8xU3lxZmVXVUtyOEUwVVh5SlgwUXZ3cGhwSVNOSFhuaXYrc25ROEp6YTNVeWpTUW5QOUlJRmdTeUpIMFBWWlM3Q1Zydz09IiwiYW1yIjpbInJzYSIsIm1mYSJdLCJhcHBpZCI6IjA0YjA3Nzk1LThkZGItNDYxYS1iYmVlLTAyZjllMWJmN2I0NiIsImFwcGlkYWNyIjoiMCIsImRldmljZWlkIjoiMjc3MzgzMWEtMTQ2NS00N2NhLThiODItMDg5NWM1MGNlMWVlIiwiZmFtaWx5X25hbWUiOiJDaGFwbGVyIiwiZ2l2ZW5fbmFtZSI6IlJpY2giLCJpcGFkZHIiOiI3Ni4xMjEuMTk0LjEzMiIsIm5hbWUiOiJSaWNoIENoYXBsZXIiLCJvaWQiOiJjYmUyOTQ4Yy01NzAxLTQyNjUtODExZC02MzNhMzE0NjczZjAiLCJvbnByZW1fc2lkIjoiUy0xLTUtMjEtMjEyNzUyMTE4NC0xNjA0MDEyOTIwLTE4ODc5Mjc1MjctNDY2NzUyOCIsInB1aWQiOiIxMDAzN0ZGRTgwMUFDODYxIiwicmgiOiIwLkFSb0F2NGo1Y3ZHR3IwR1JxeTE4MEJIYlI1VjNzQVRialJwR3UtNEMtZUdfZTBZYUFEVS4iLCJzY3AiOiJ1c2VyX2ltcGVyc29uYXRpb24iLCJzdWIiOiJsRk9zQ0xkdGFidHBoU0U1c3VyTktqX3lqam84aXo4S2ZaM0Z4dHdnV3ZNIiwidGlkIjoiNzJmOTg4YmYtODZmMS00MWFmLTkxYWItMmQ3Y2QwMTFkYjQ3IiwidW5pcXVlX25hbWUiOiJyY2hhcGxlckBtaWNyb3NvZnQuY29tIiwidXBuIjoicmNoYXBsZXJAbWljcm9zb2Z0LmNvbSIsInV0aSI6IktqdVlhbEdYQlVhcndHajNBNDB3QUEiLCJ2ZXIiOiIxLjAifQ.U6yTE35AdDTQhJgUEjjUPFW6raM9ORfO7SzFya4azbPzrwKntGBrekyjLzqaR5C-cJzPw9c8dp5Gn-giHw0cpXnZTbrA-Mv9-g4Gwsto3I1LxqXkH3bBmhGHqaDvQ3j1D-N_I1SU1whxj6FjgsgzfwBlNgCo1gt0nDyEWHrfGvO3lPkmwNTFLu3-hUqRQDxmiEu519Uf4cxMm00cxSaKegNs2eco0aYAUxCjgD2cdjVlEgbioL9ZdCCNJRblAgHhJCxHVGAfPBTyL03XL_Xngc-DpZdyC6MMp1SF4-BEc72S9Nli4lJHNpGq-hy8DCB2_5bduHwXMGOTnflwbq14PQ
C:\Users\rchapler>

Step 3: Post API Request

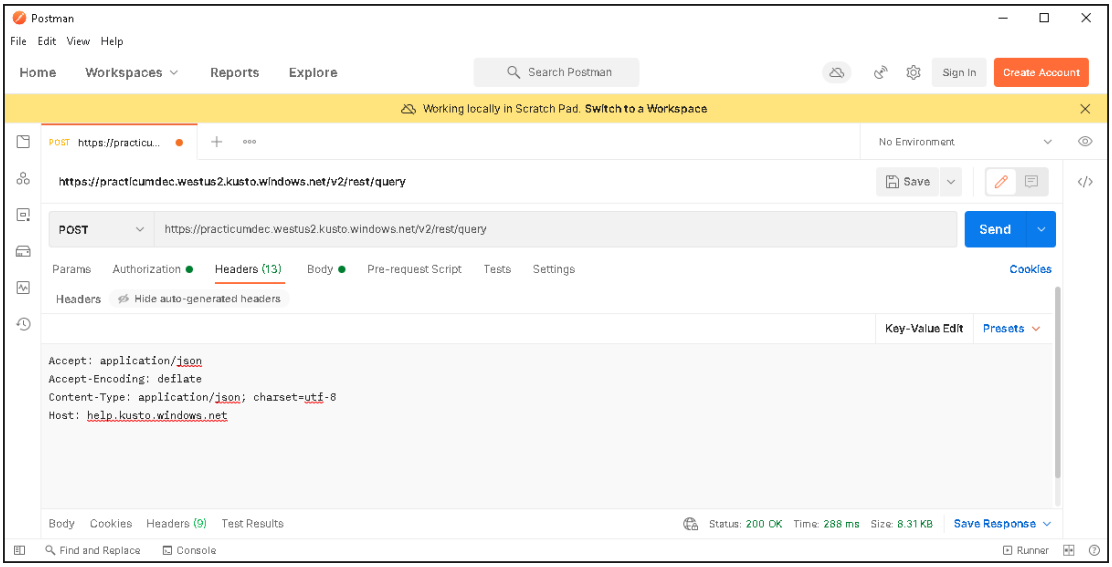
Open a new request in Postman.



Select **POST** from the dropdown. Enter a hostname (with the name you used for your cluster); example:

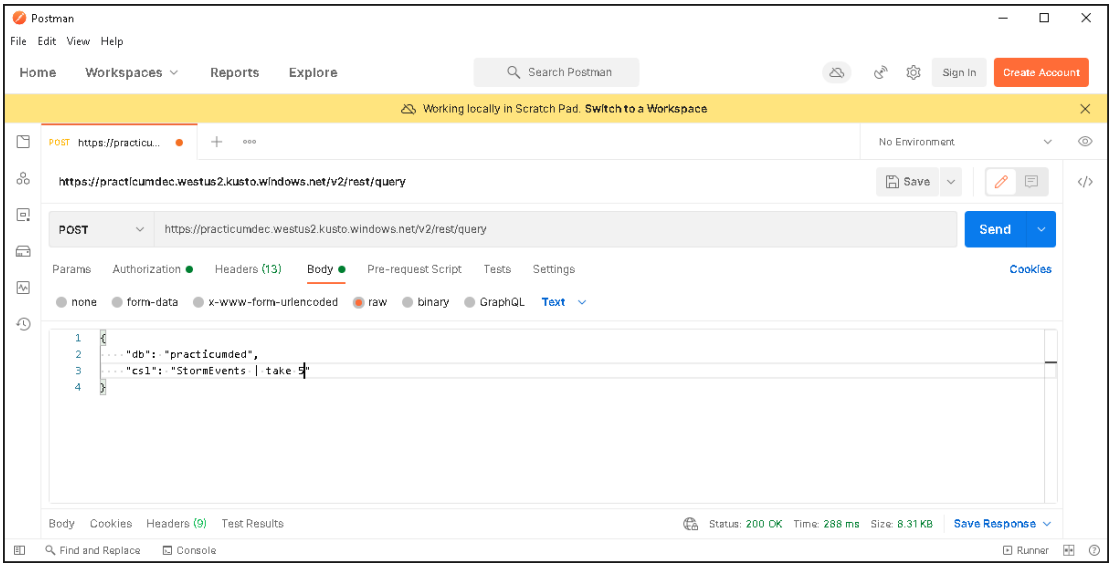
```
https://practicumdec.westus2.kusto.windows.net/v2/rest/query
```

On the **Authorization** tab, select "**Bearer Token**" from the dropdown and paste your previously copied token in the resulting interface.



On the **Headers** tab, enter the following items:

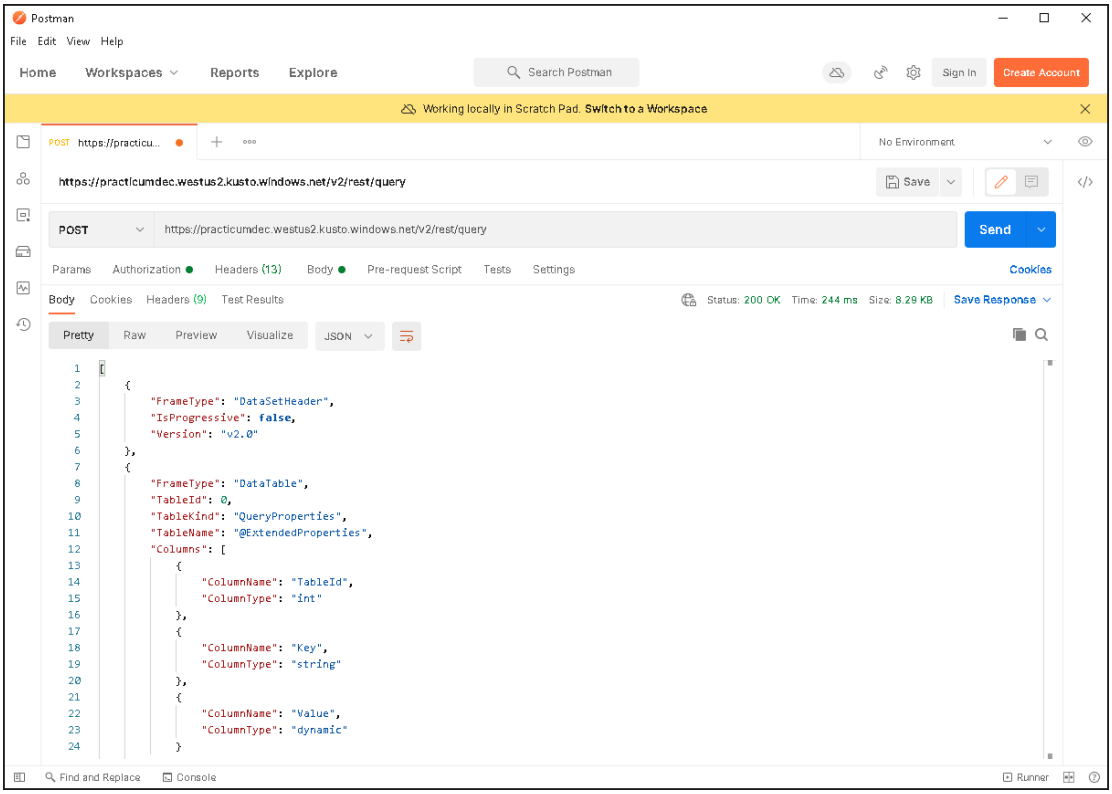
Accept	application/json
Accept-Encoding	deflate
Content-Type	application/json; charset=utf-8
Host	help.kusto.windows.net



On the **Body** tab, paste the following:

```
{
 "db": "sharedded",
 "csl": "StormEvents | take 5"
}
```

Click the **Send** button.



You should see JSON with both schema and data included.

Objective Complete... congratulations!

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.

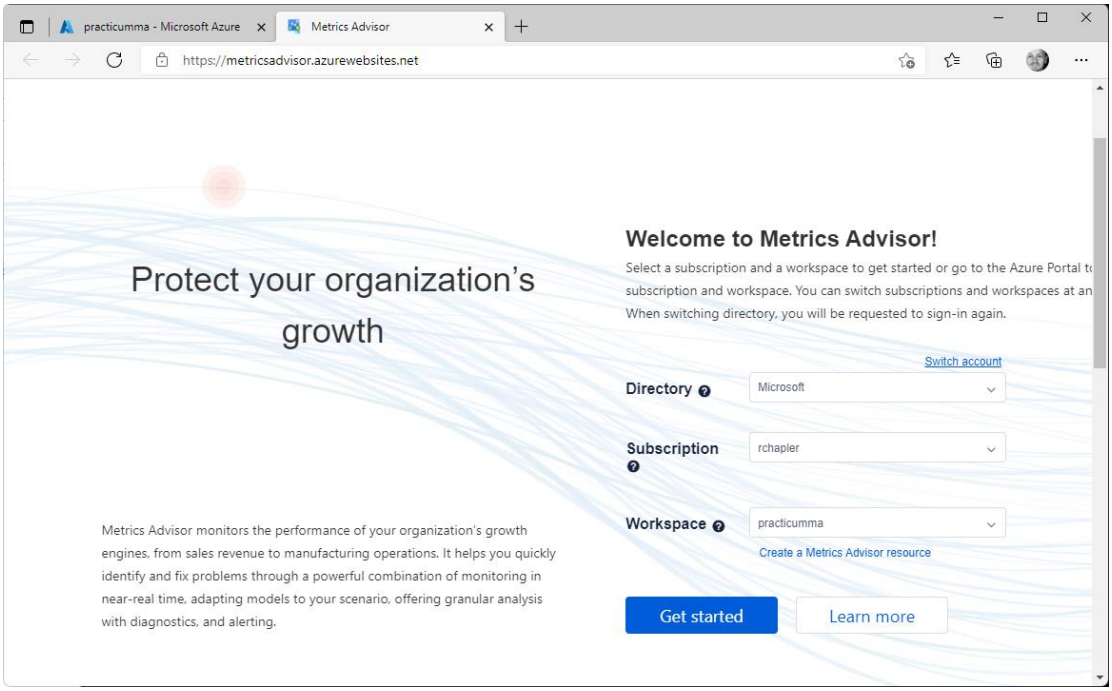
Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Data Explorer
- Metrics Advisor

Step 2: Onboard Data Feed

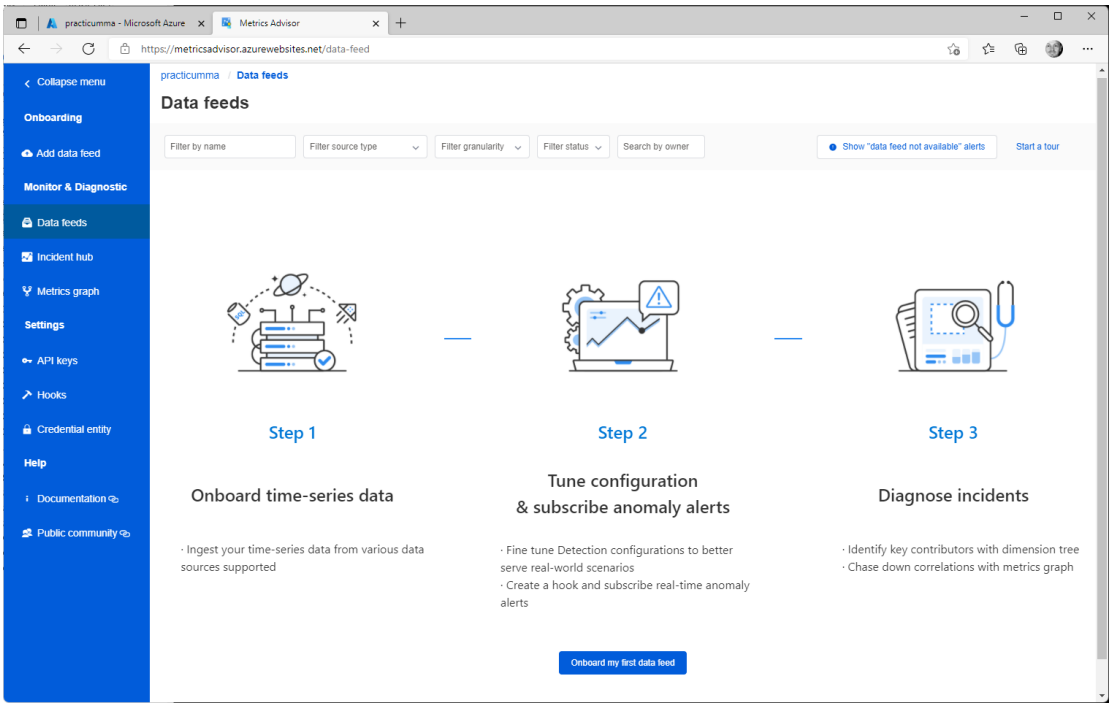
Navigate to Metrics Advisor and then click the "**Go to your workspace**" link.



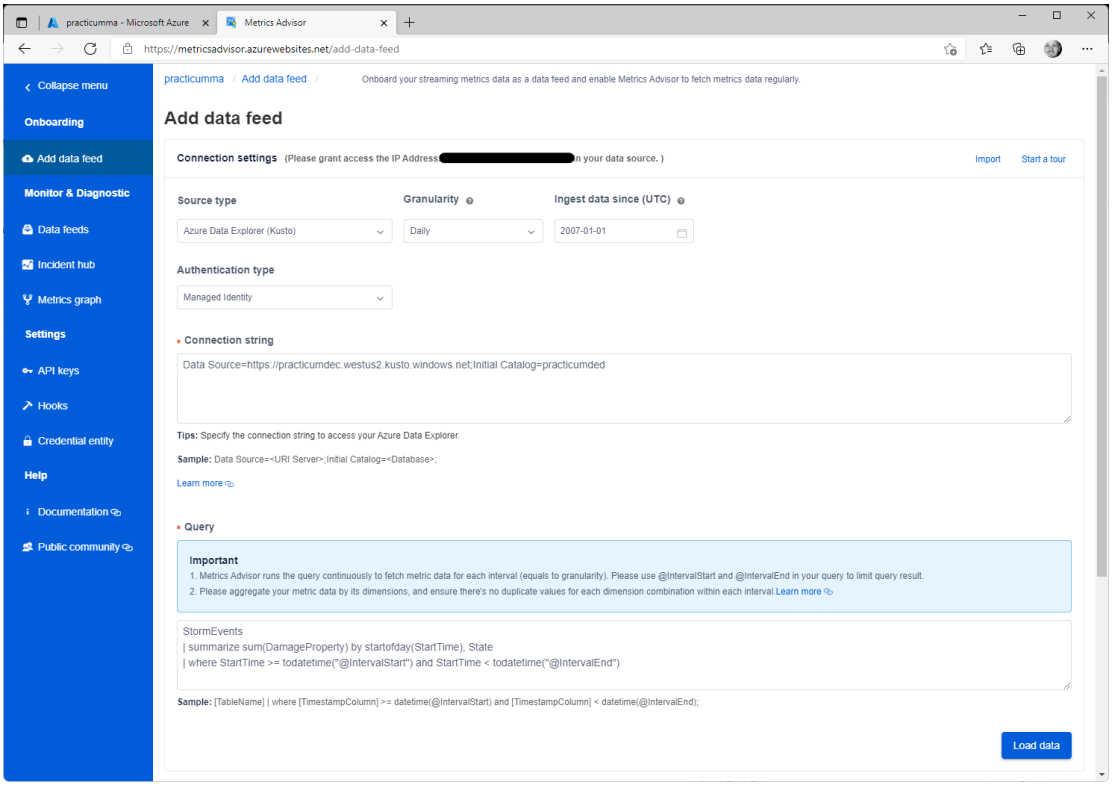
On the "**Welcome to Metrics Advisor!**" page, enter values for the following items:

Directory	Select your directory
Subscription	Select your subscription
Workspace	Select Metrics Advisor

Click the “**Get started**” button.



Click the “Onboard my first data feed” button.

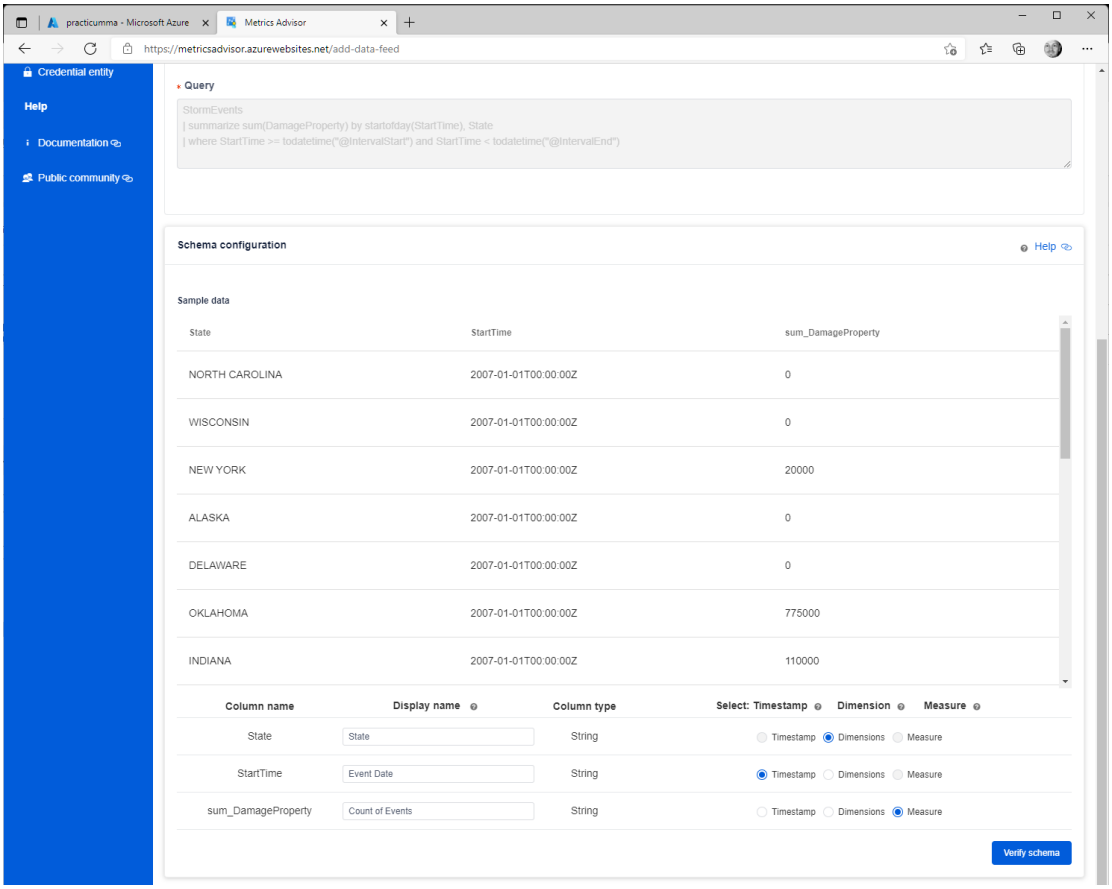


On the "**Add data feed**" page, enter values for the following items:

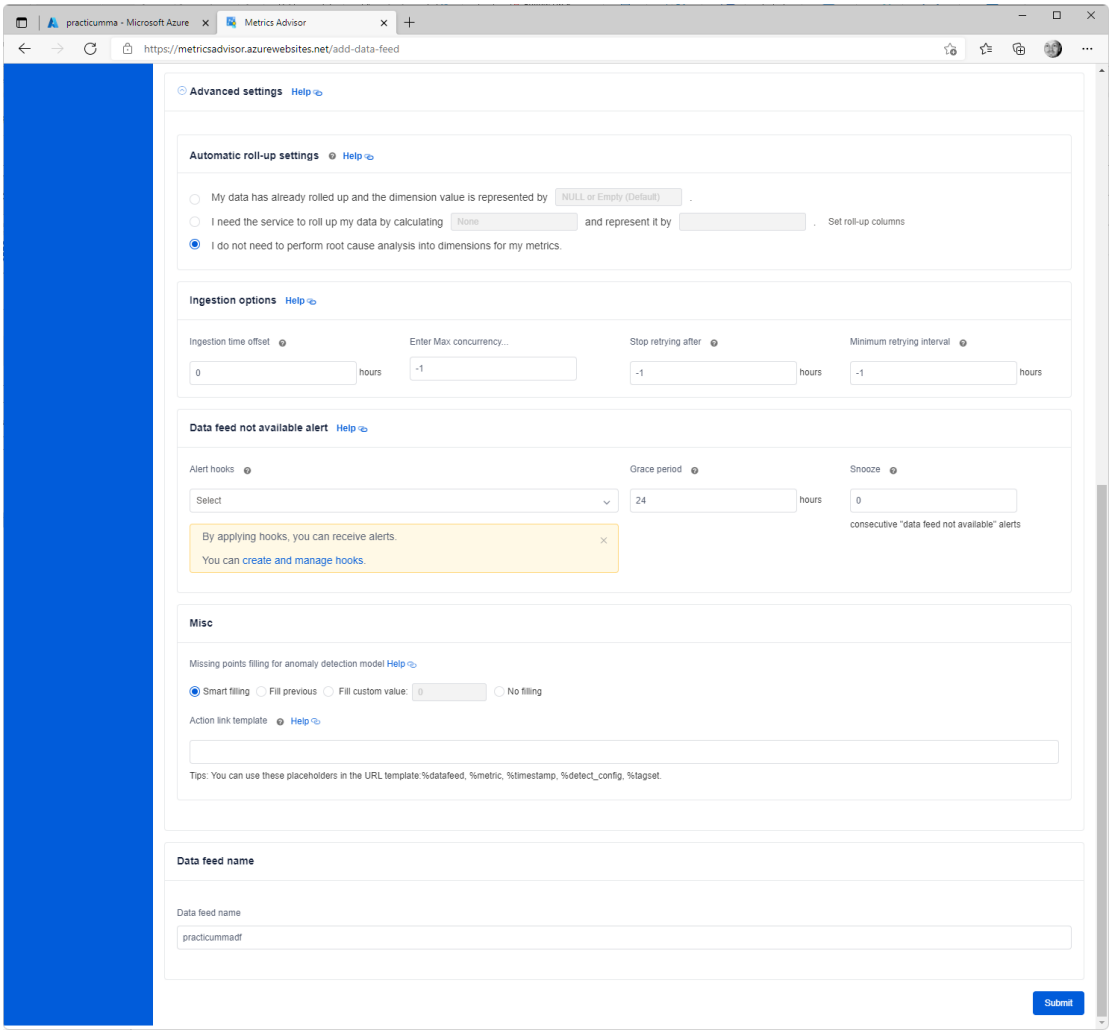
Source Type	Select "Azure Data Explorer (Kusto)"
Granularity	Confirm default selection, **Daily**
Ingest Data Since (UTC)	Select a date appropriate for your data set
Authentication Type	Select "Managed Identity"
Connection String	Enter in form: `Data Source=https://{ADX Cluster}.{region}.kusto.windows.net;Initial 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.

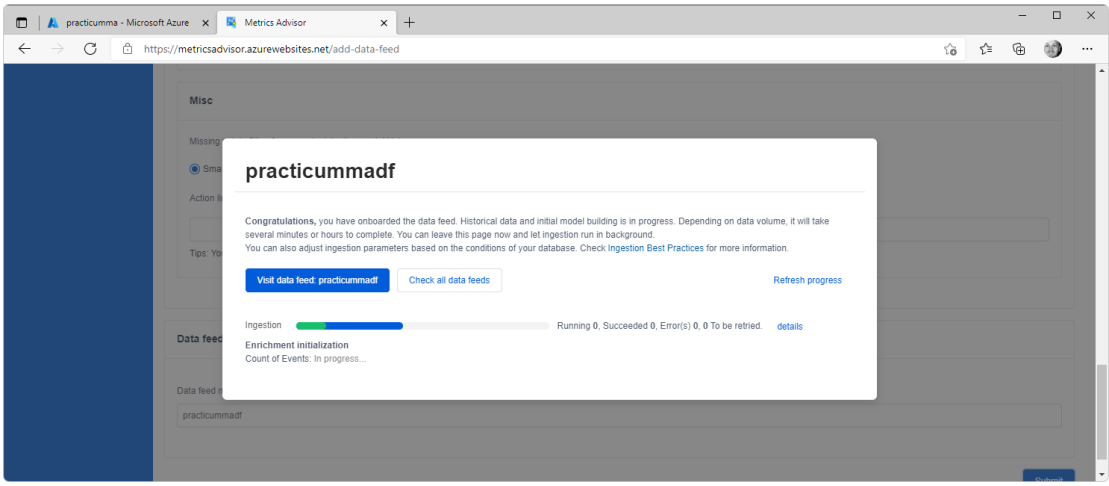


Click the “**Verify Schema**” button.



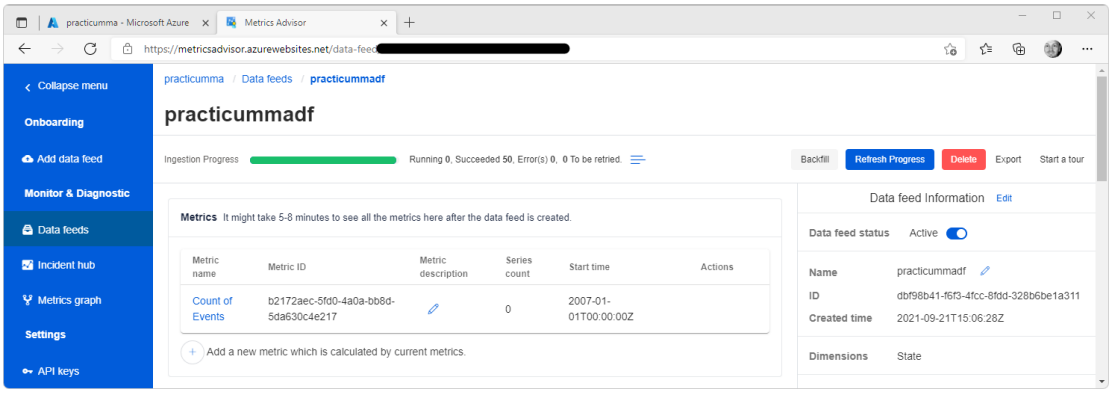
No changes are required to default values in “**Advanced Settings**”.

Enter a “**Data Feed Name**” and then click the Submit button.



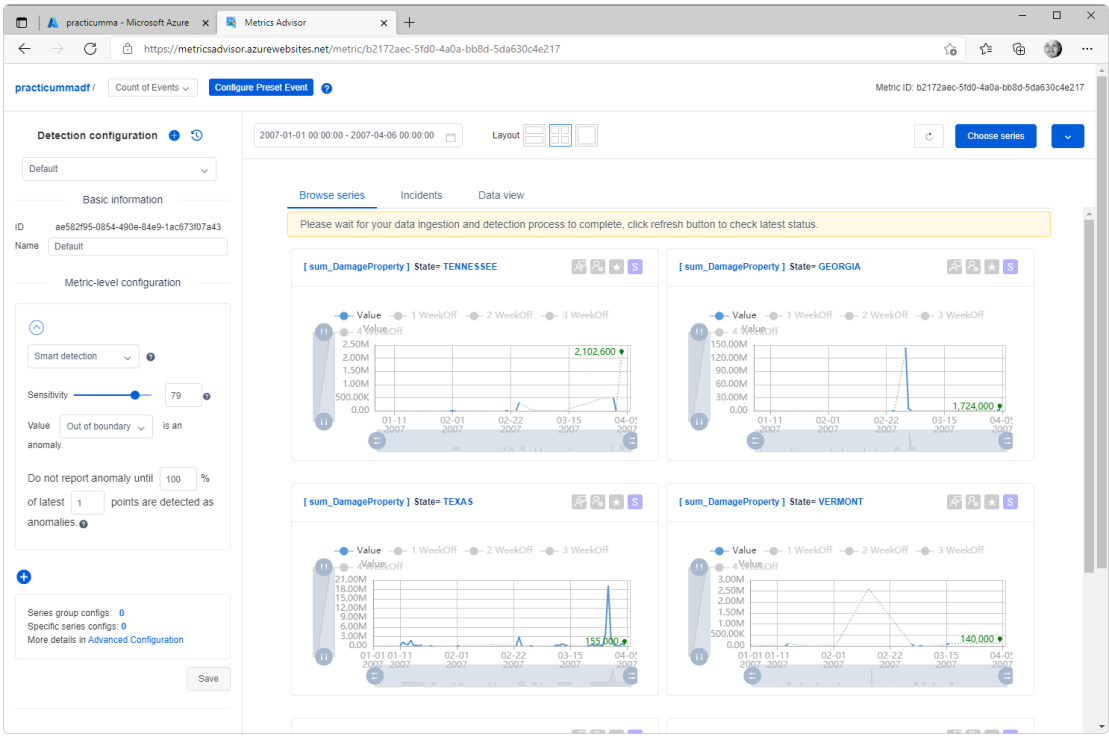
You will receive a "**Congratulations**…" message.

Click the "**Visit data feed**…" button.



Monitor progress on the data feed page.

Click into the Metric Name link to see analysis.



Objective Complete... congratulations!

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.

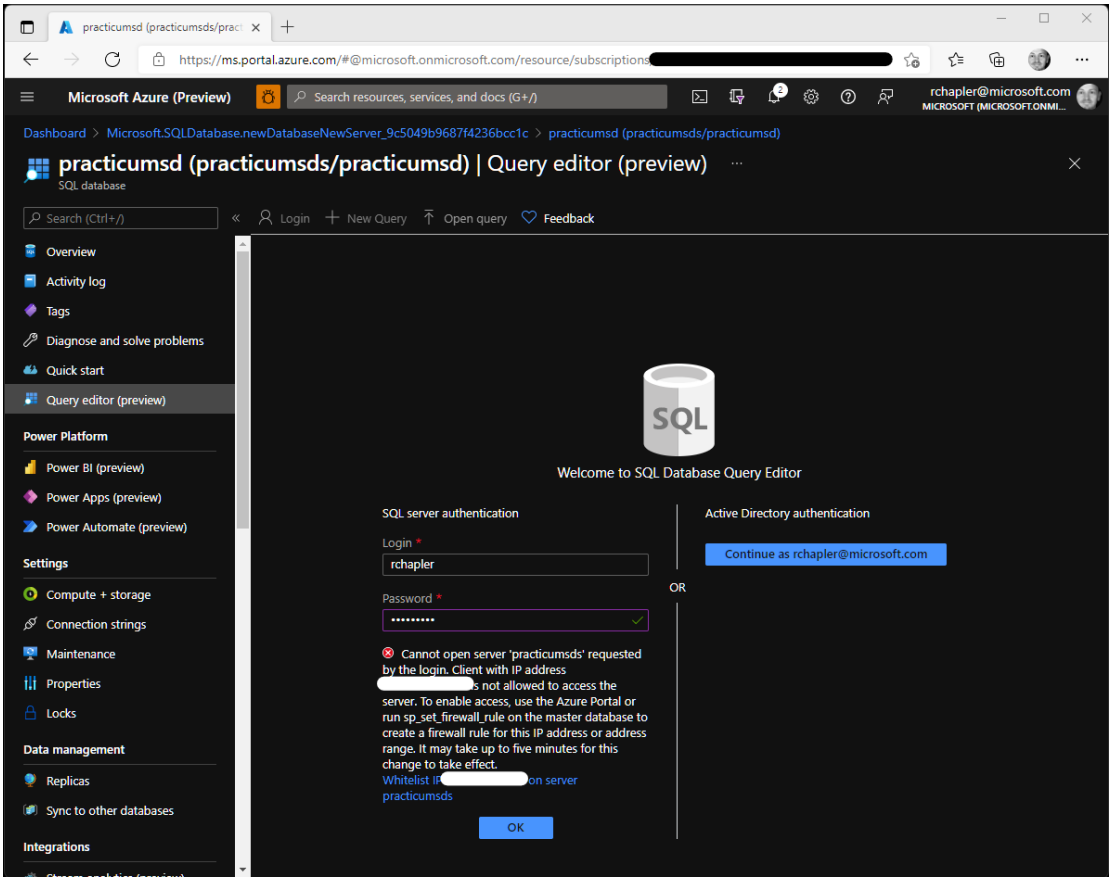
Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

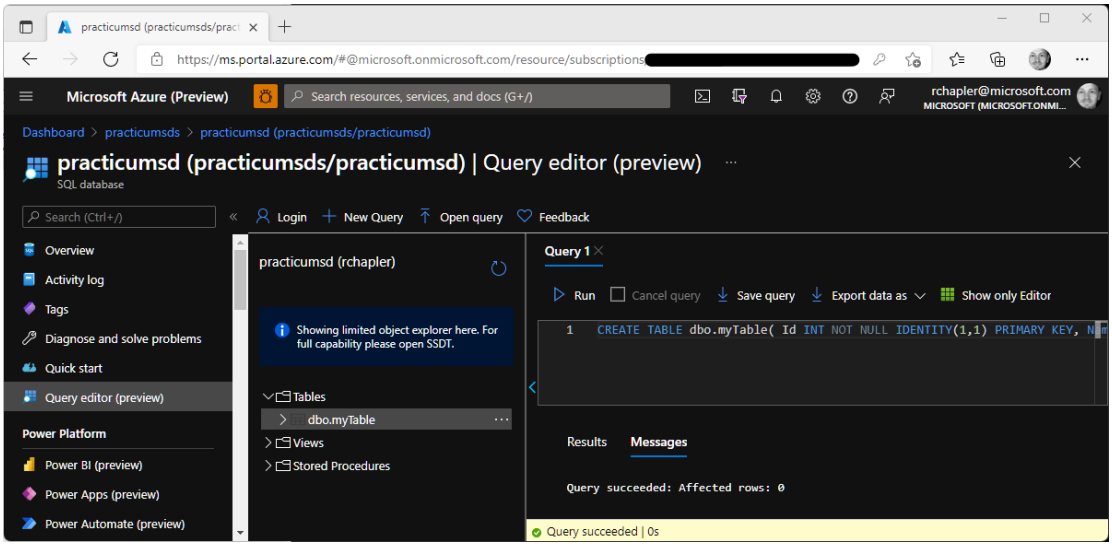
- Azure SQL
- PowerApps

Step 2: Create Target Table

Navigate to SQL.



Navigate to “**Query editor**…” and login. Whitelist your IP address as appropriate.



Execute the following T-SQL:

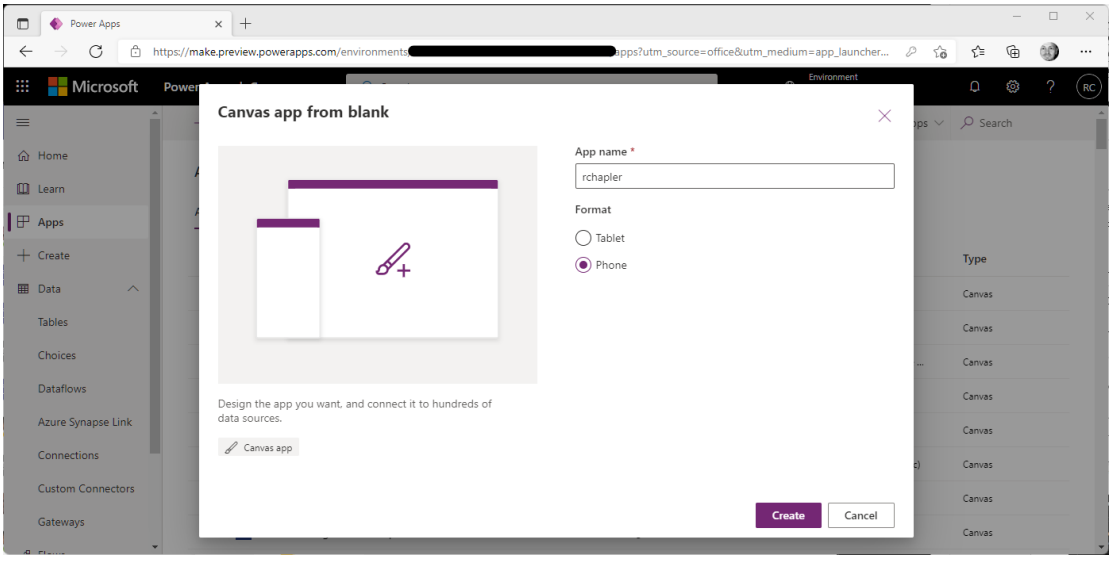
```
CREATE TABLE dbo.myTable( Id INT NOT NULL IDENTITY(1,1) PRIMARY KEY, Name VARCHAR(64), Picture IMAGE )
```

Step 3: 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**” page, click “**+ New app**” and select **Canvas** from the dropdown.



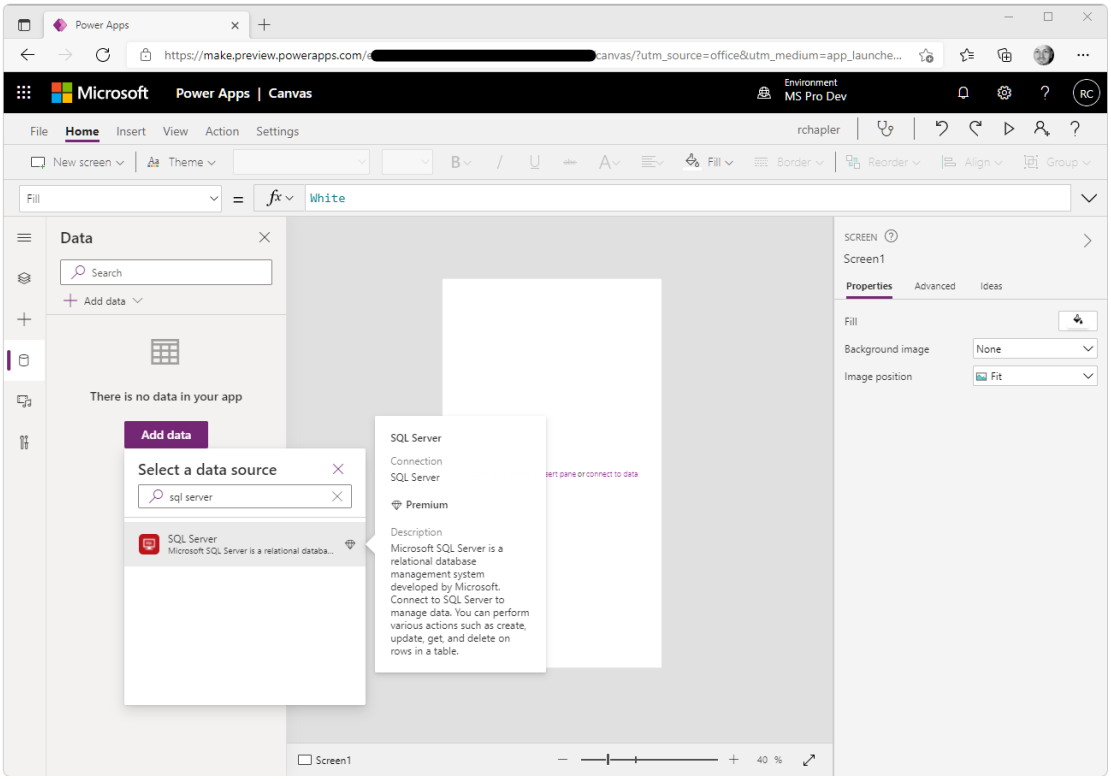
On the “**Canvas app from blank**” popup, enter the following items:

App Name	Enter a meaningful name aligned with standards
Format	Select the **Phone** radio button

Click the **Create** button.

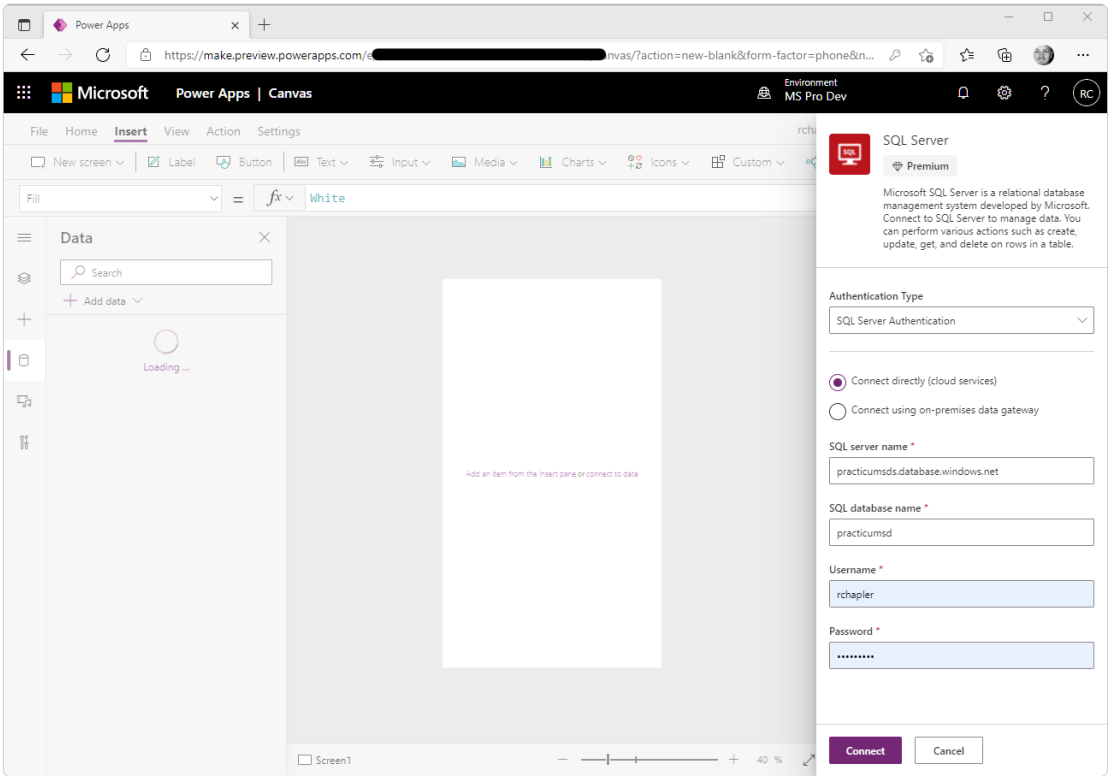
Step 4: Add Data

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 dropdown.

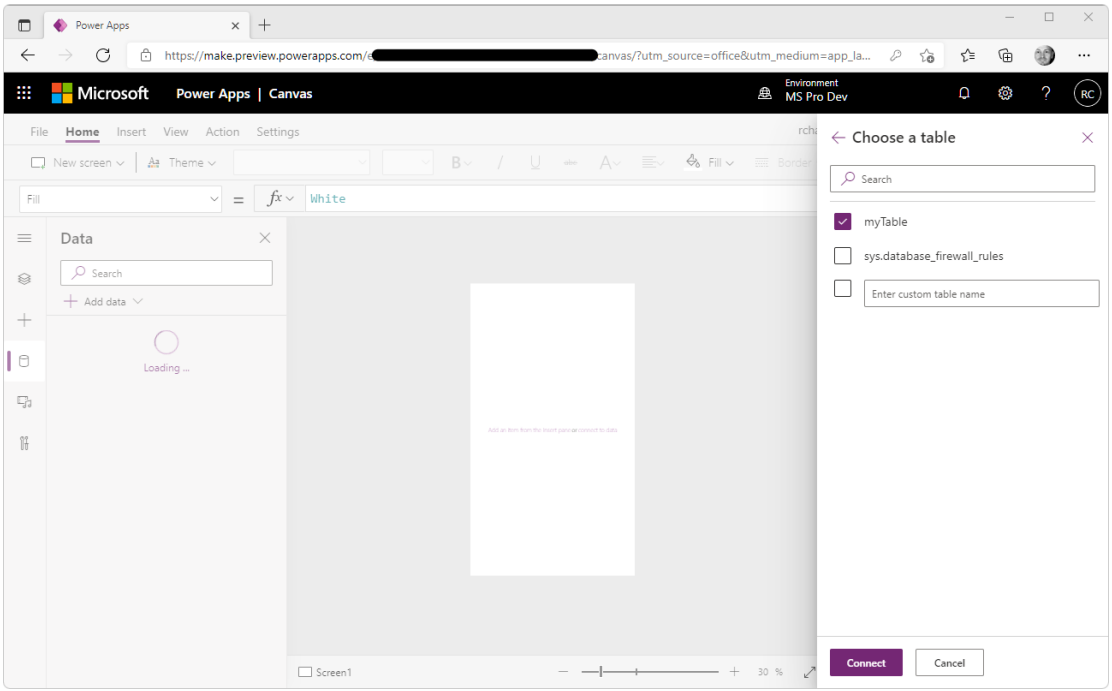


On the "**SQL Server**" popout, enter the following items:

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.



On the “**Choose a table**” popout, check the table created in [Create Target Table].

Click the **Connect** button.

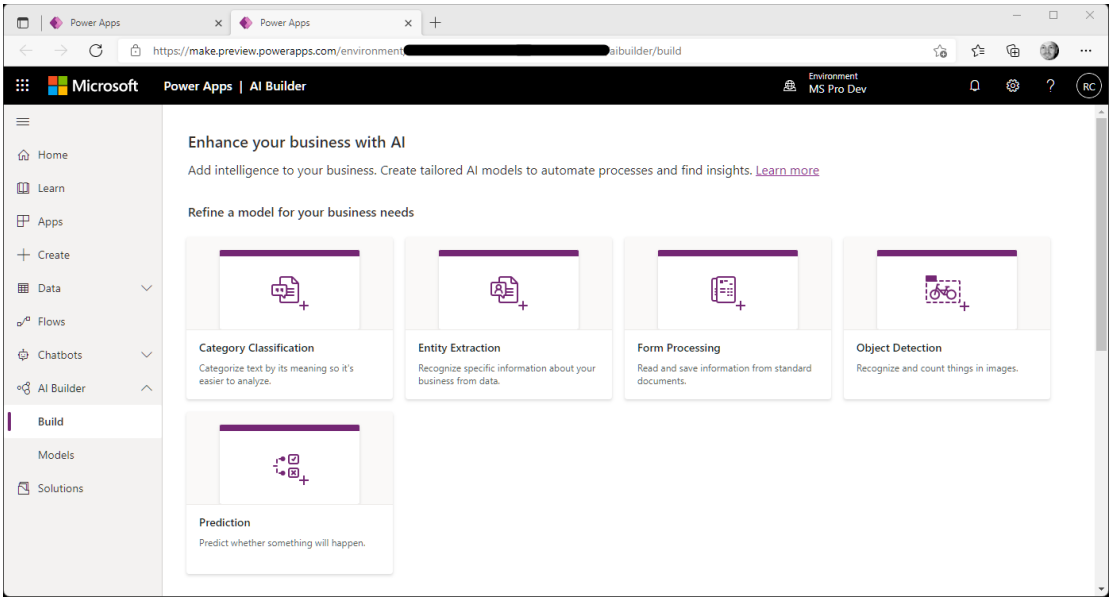
Step 6: Insert Object Detector

Click the “Add an item from the Insert pane” link.

On the resulting, left-hand **Insert** menu, expand “**AI Builder**” and select “**Object detector**”.

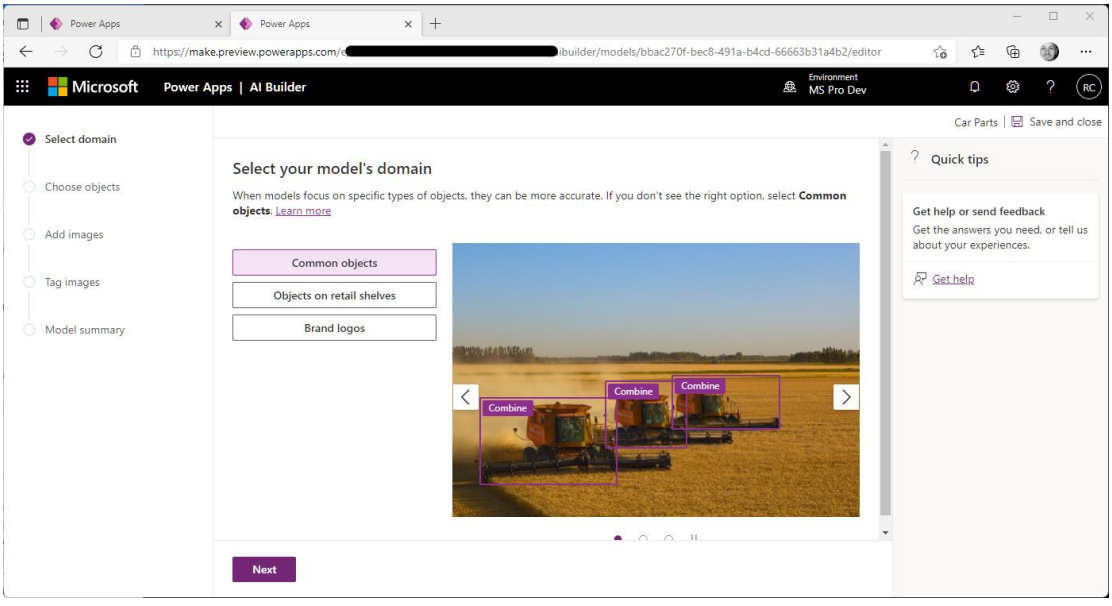
On the resulting, right-hand popout, click “**+ New model**”.

Step 7: Create AI Model

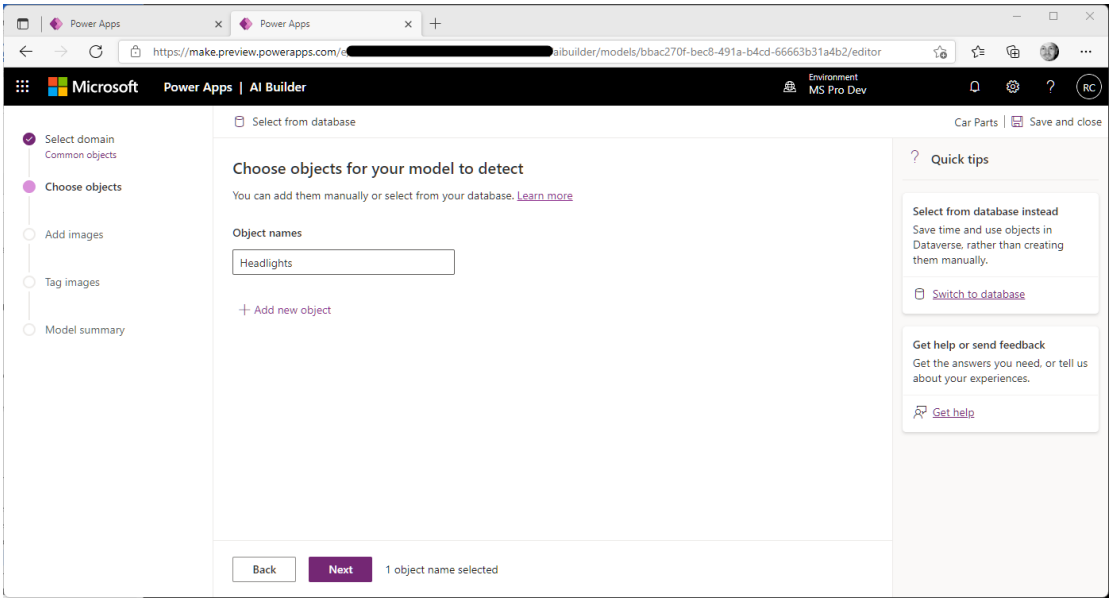


On the new “Power Apps | AI Builder”, “Enhance your business with AI” page, click the “Object Detection” button.

Name the AI model and then click the **Create** button.



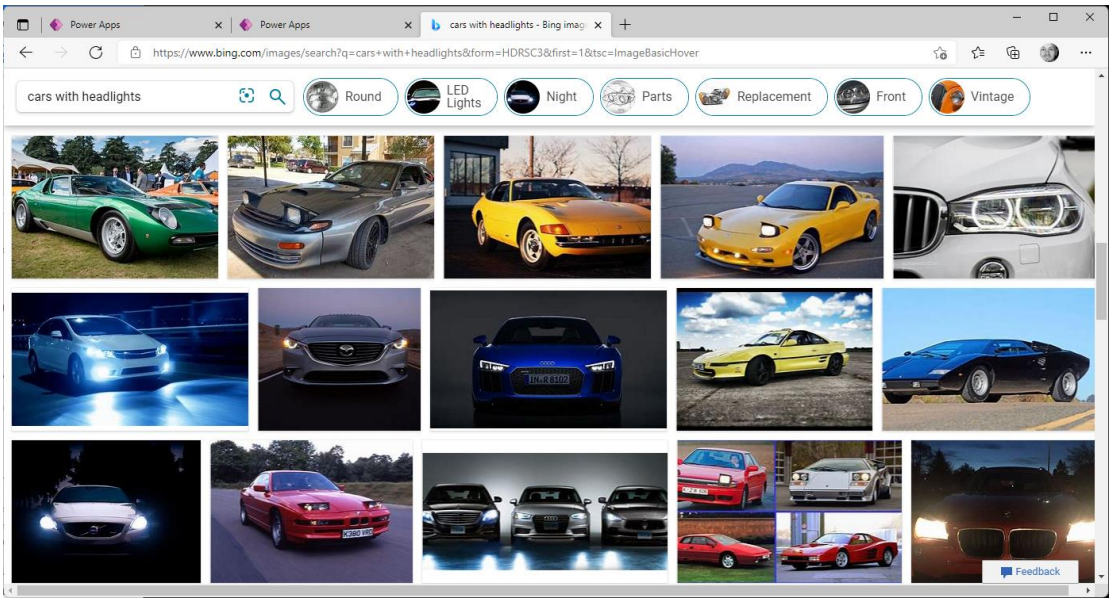
On the “**Select your model’s domain**” page, click the “**Common objects**” and then the “**Next**” button.



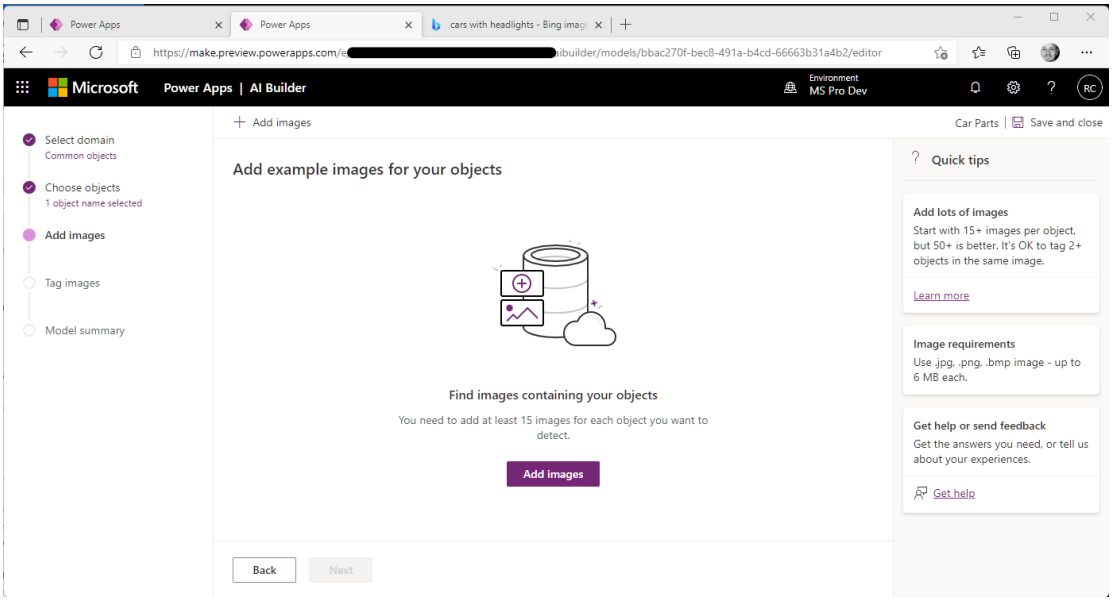
On the "**Choose objects for your model to detect**" page, click "**+ Add new object**", enter an object name and then click the "**Next**" button.

On the "**Add example images for your objects**" page, we will add image files that will be used to train the AI model.

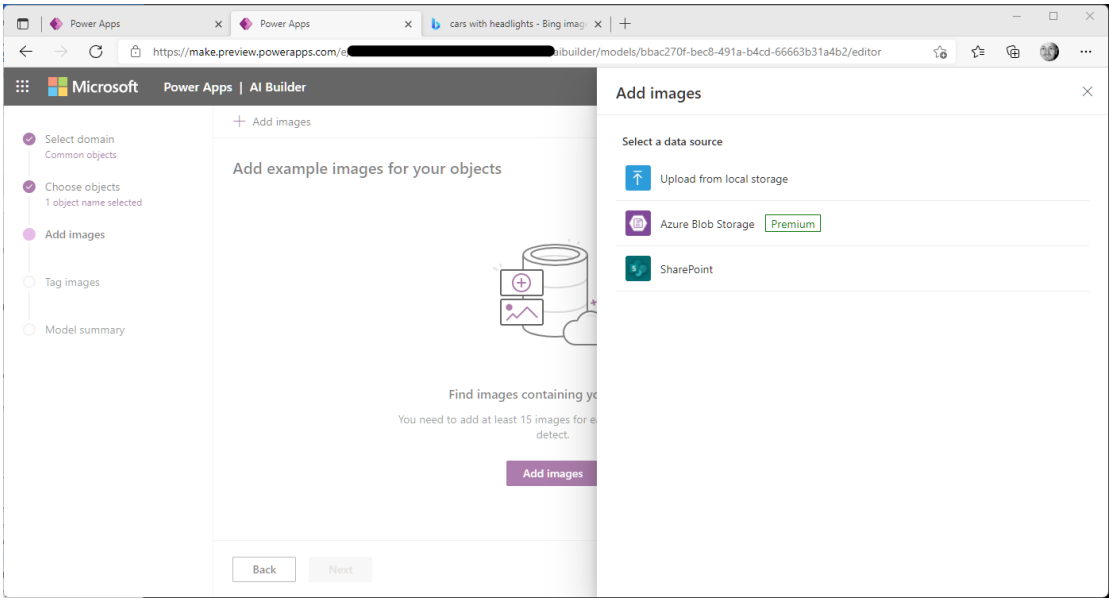
For this exercise, we can pull images of cars (with headlights) from a search engine.



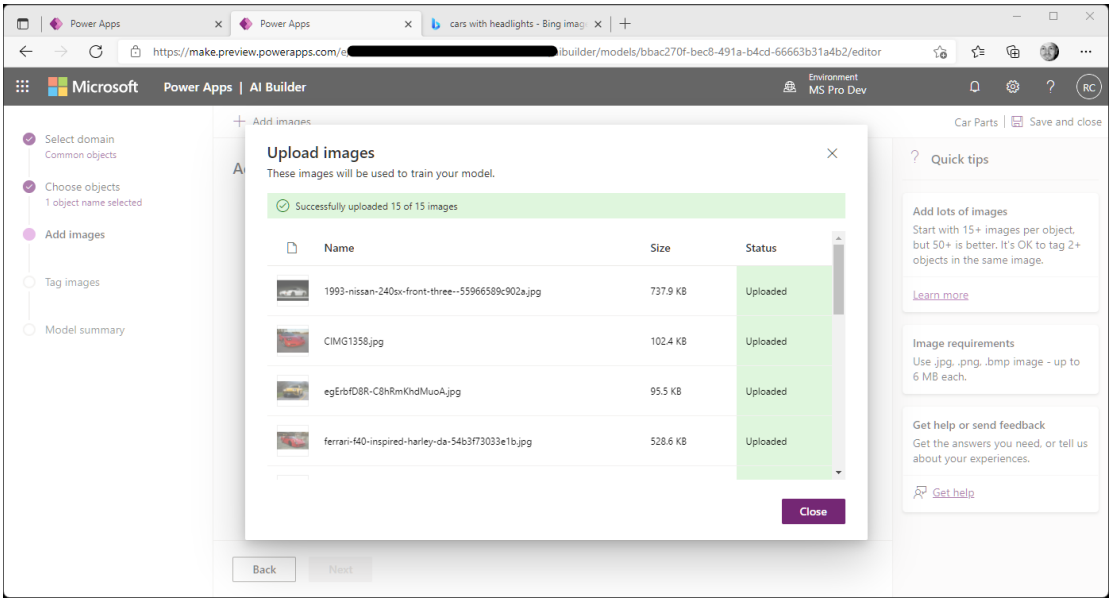
For each of the required 15 images, download an image file to a temporary folder on the Windows desktop.



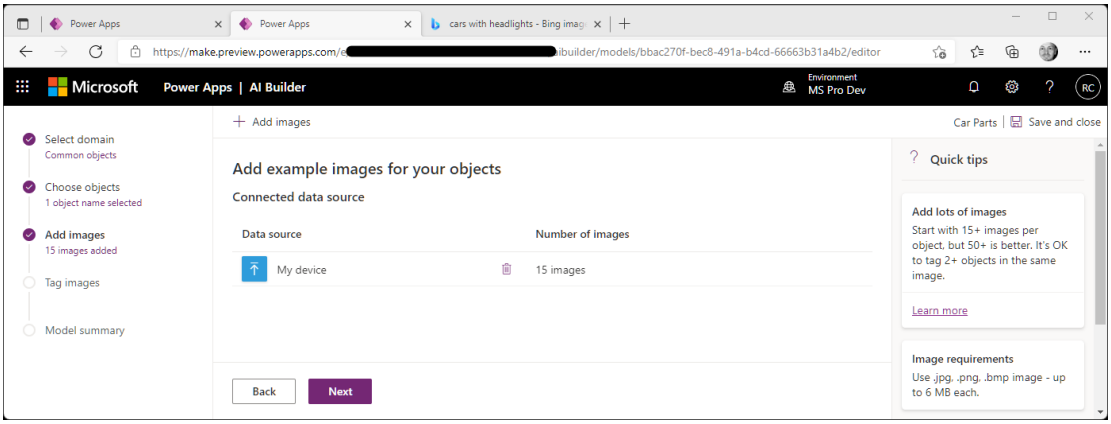
On the “Add example images for your objects” page, click the “Add images” button.



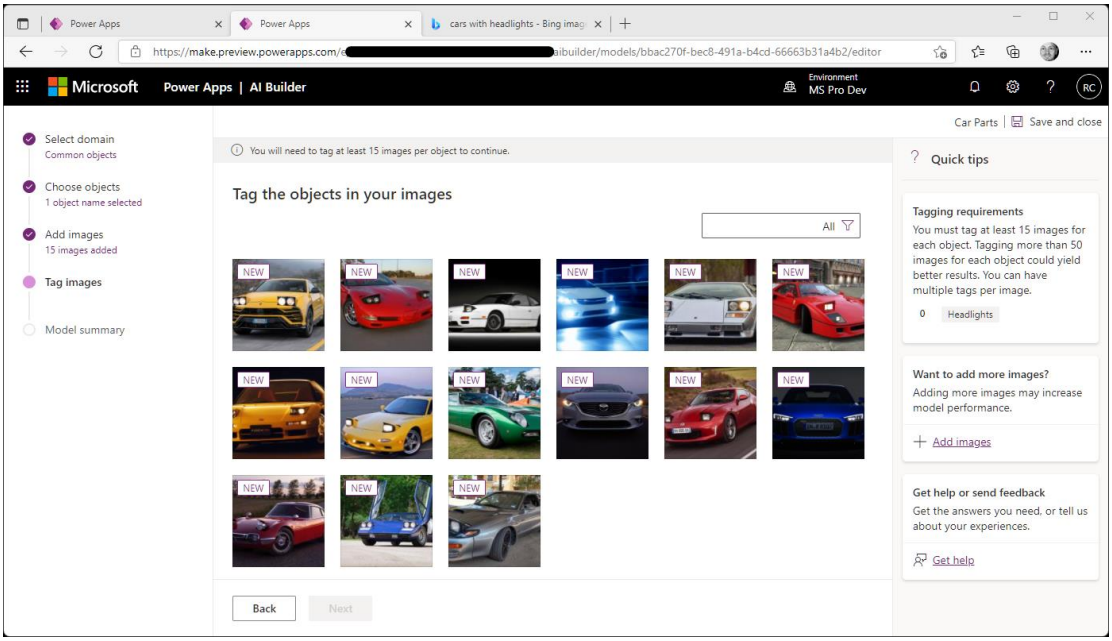
On the resulting “**Add images**” popout, click “**Upload from local storage**”. Select the Example Images downloaded to your desktop.



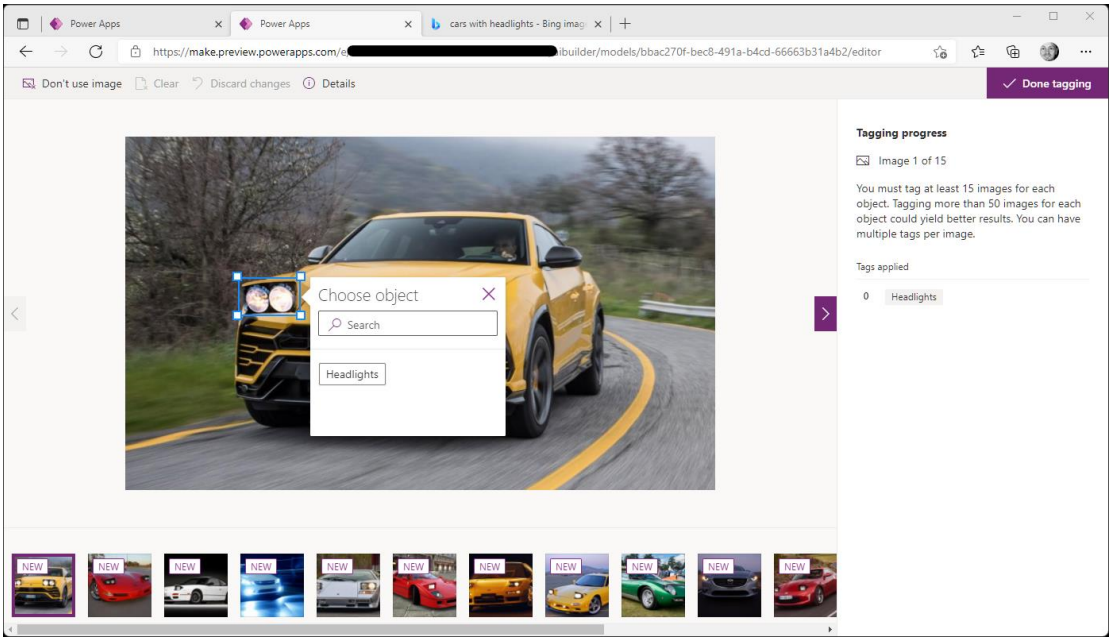
Review the items in the "**Upload images**" popup, then click the "**Upload 15 images**" button. Click **Close**.



Back on the "**Add example images for your desktop**" page, confirm "**Number of images**" shows 15 or more images, then click **Next**.

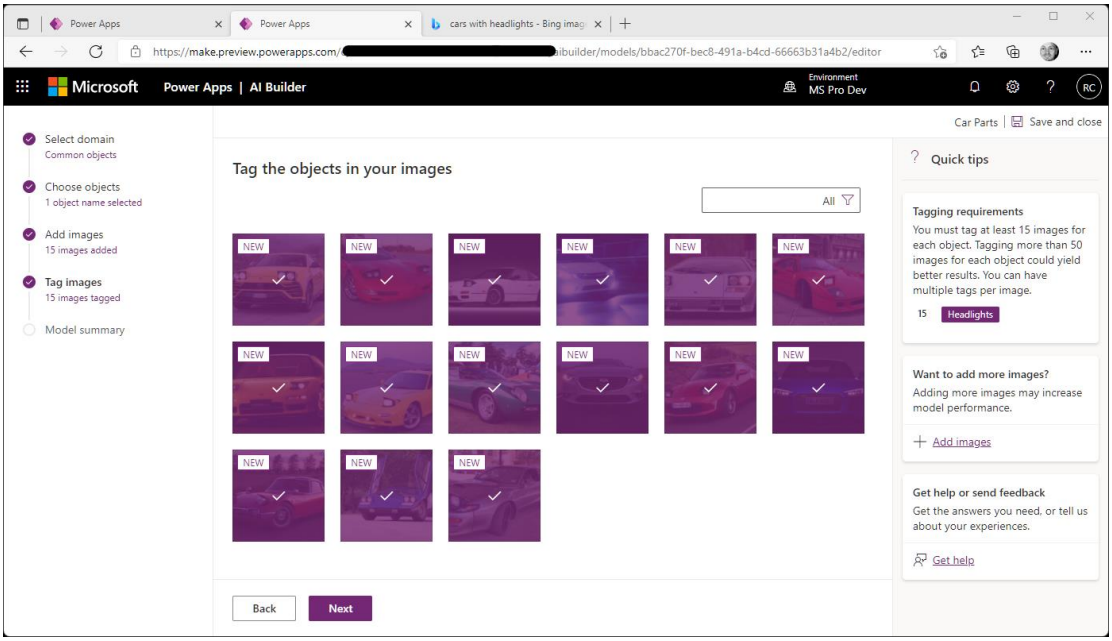


Select the first image on the "**Tag the objects in your images**" page.

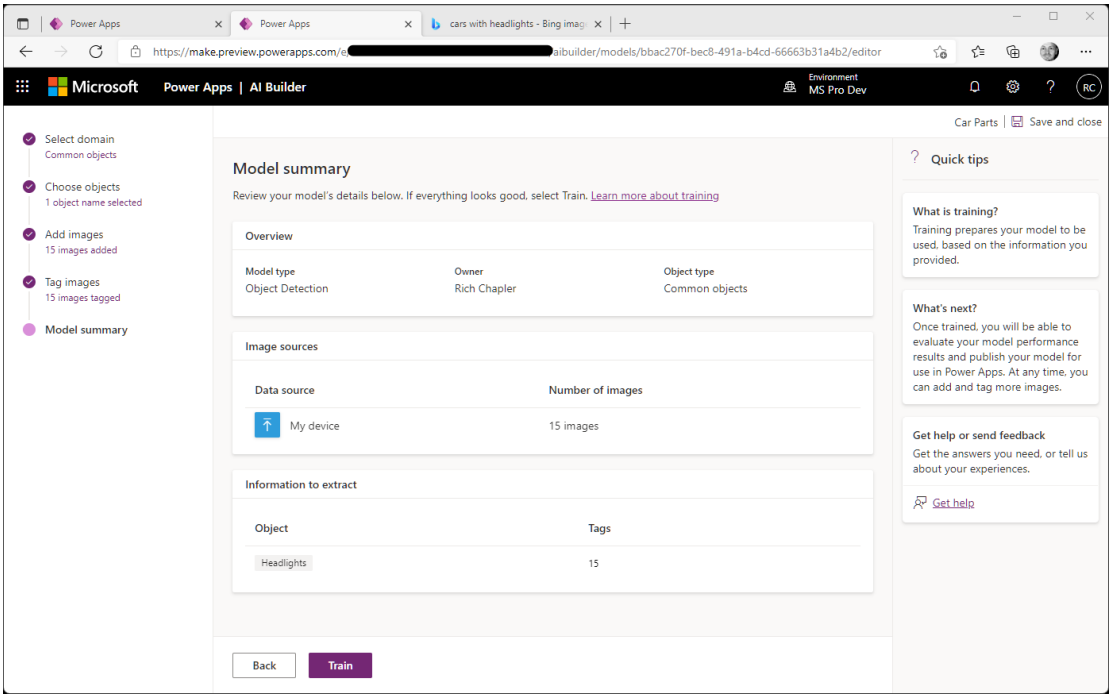


Drag a rectangle around headlights in the image, then click the Headlights button on the "**Choose object**" popup.

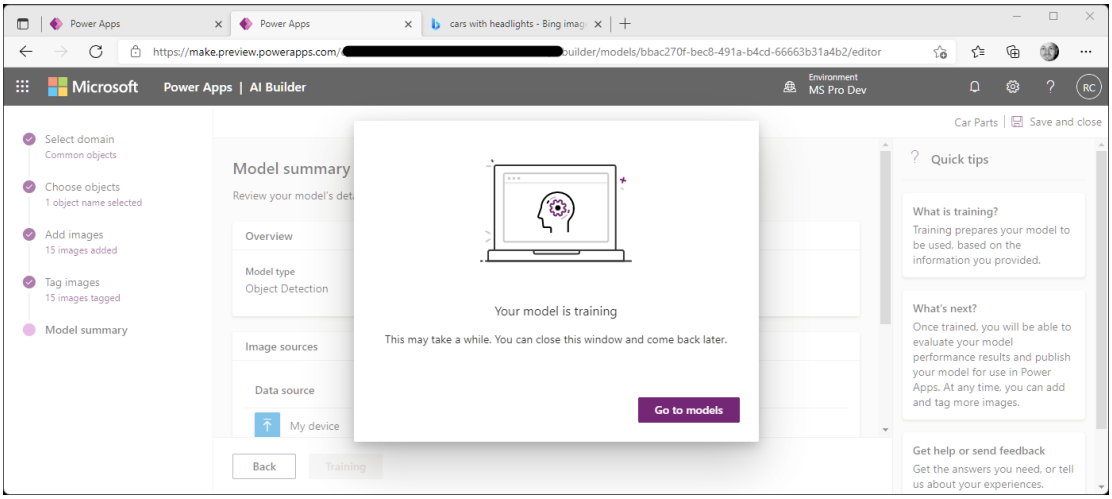
Repeat for all taggable items, click **>** to move to through images, and then click the "**Done tagging**" button in the upper-right.



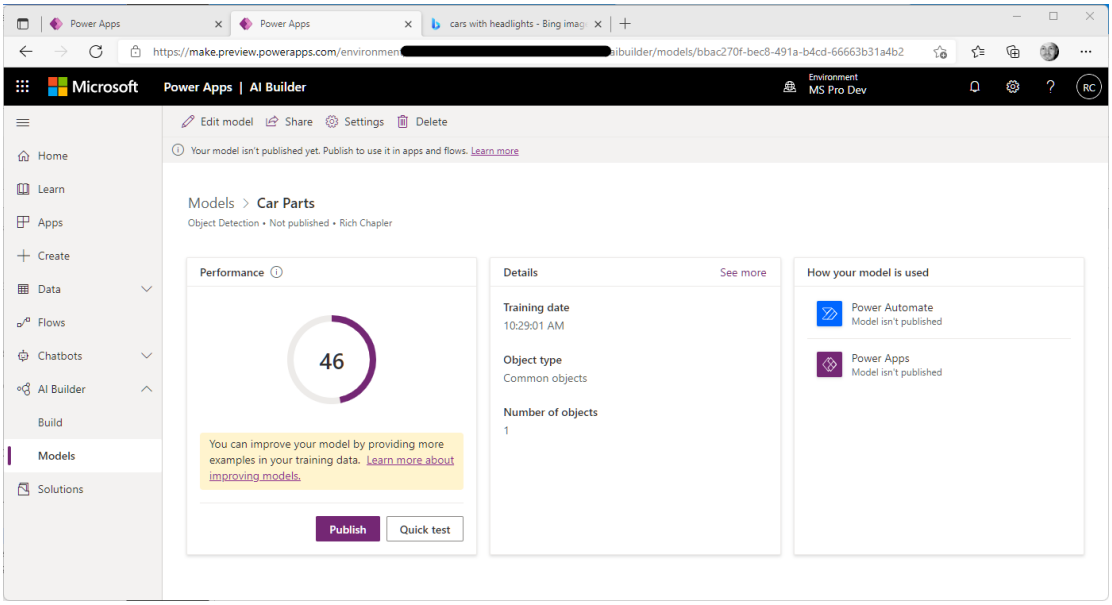
Click the **Next** button.



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.

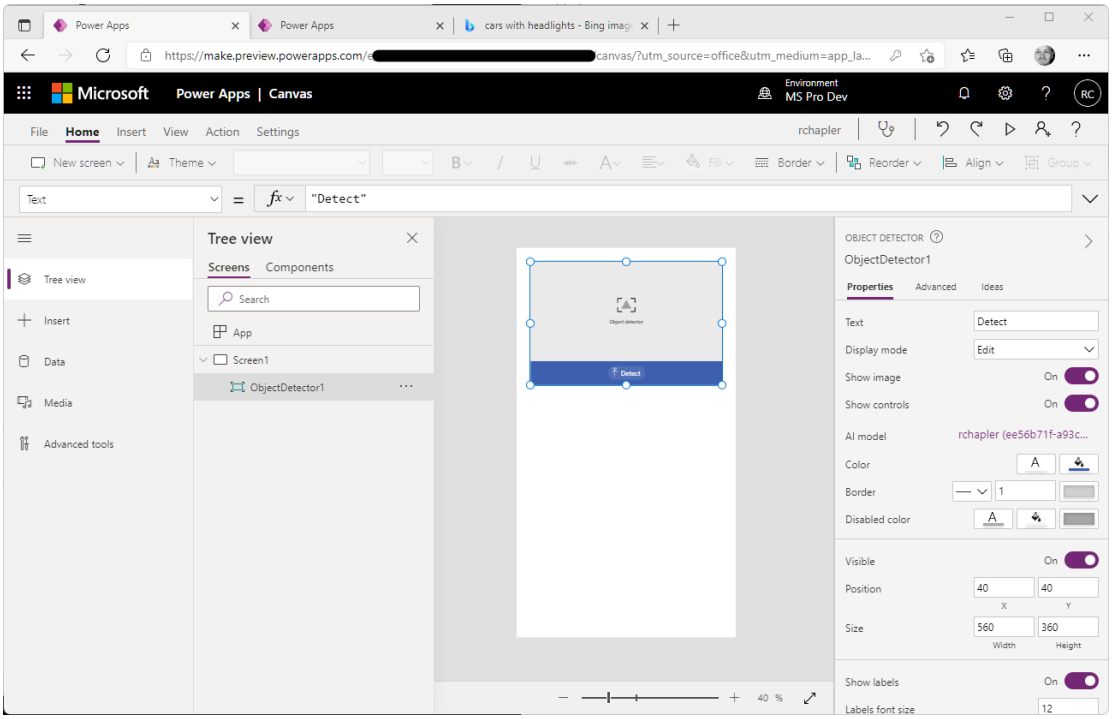


In the snip above, we see that **Performance** {i.e., model performance expectation %} of 46 for the example images included in training.

In the future, you can improve model quality by including more images and re-training the model.

Click the **Publish** button.

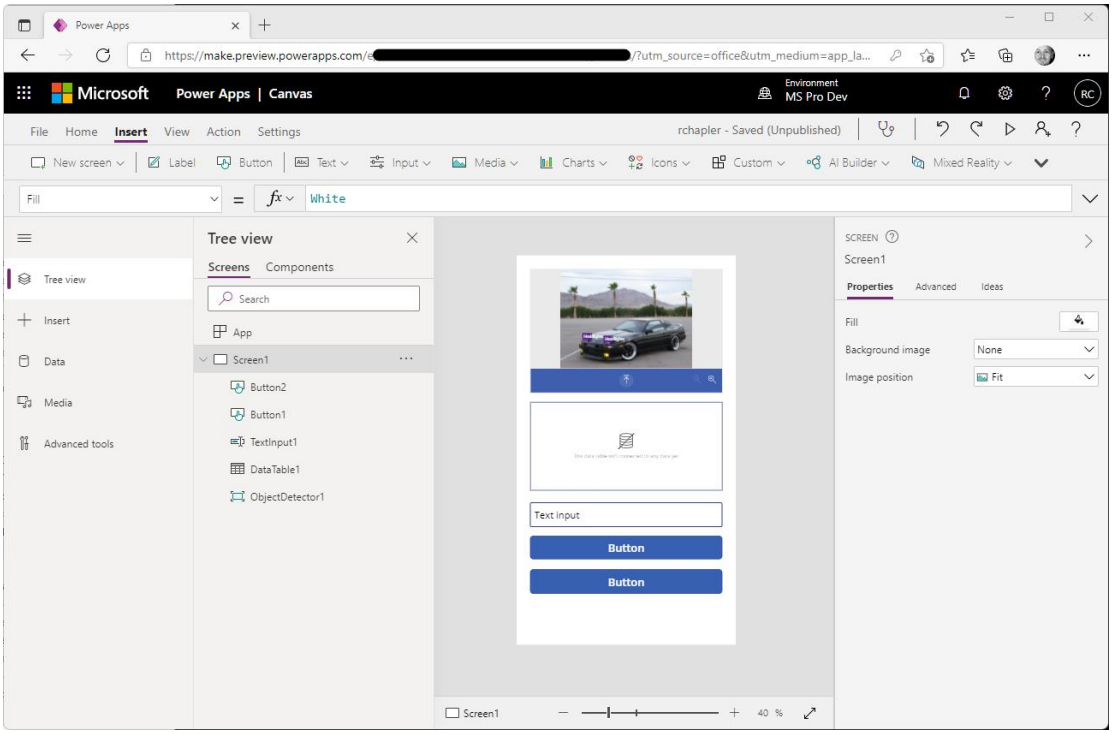
Return to Power Apps and complete the OjbectDetector1 component by selecting your newly created model.



Note: Consider resizing the component to improve visibility.

Step 8: Insert / Configure Controls

Return by closing Preview Mode.

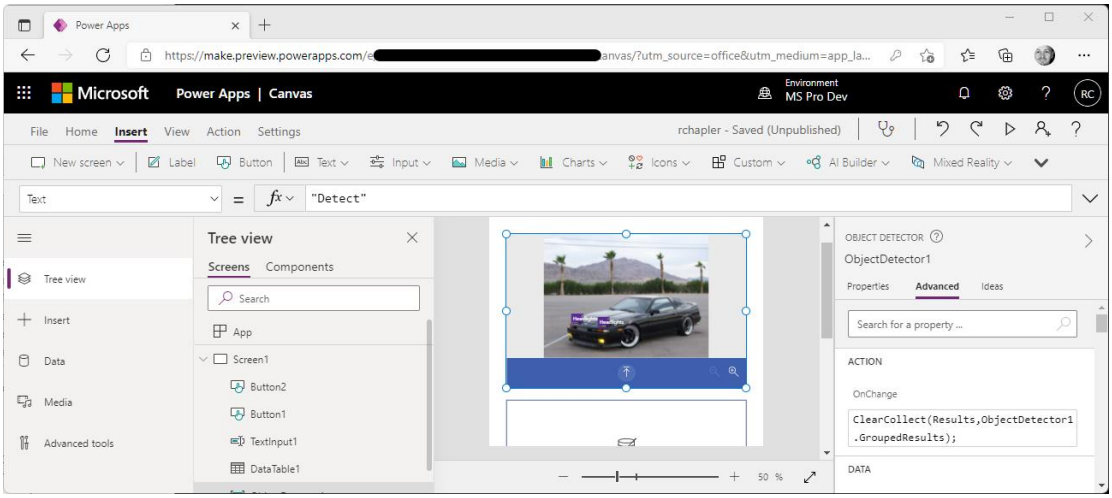


Insert the following controls:

- **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

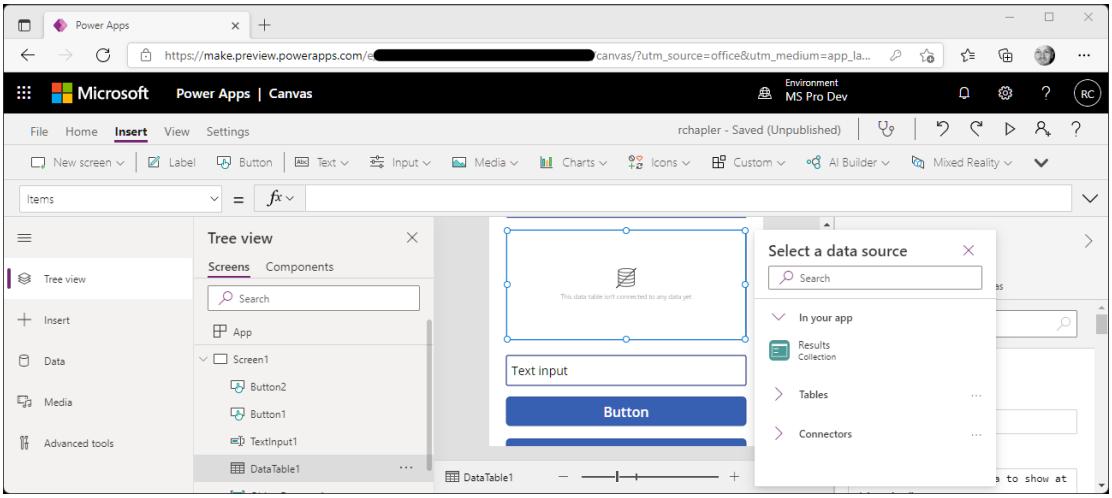


Click on **ObjectDetector1** in the left-hand **Tree View**. On the resulting right-hand popout, click the **Advanced** tab.

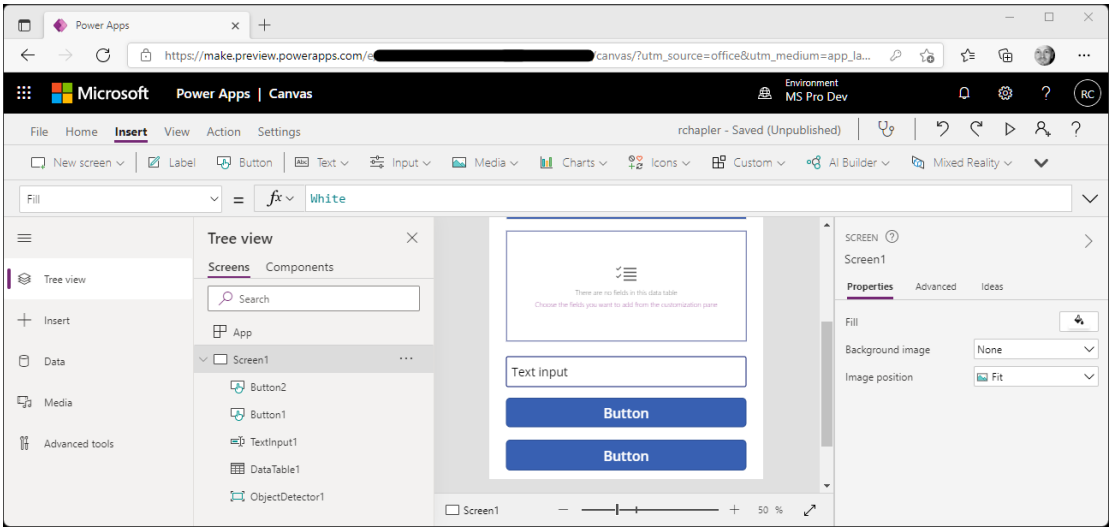
Paste the following logic into **Action** | **OnChange**:

```
ClearCollect(Results,ObjectDetector1.GroupedResults);
```

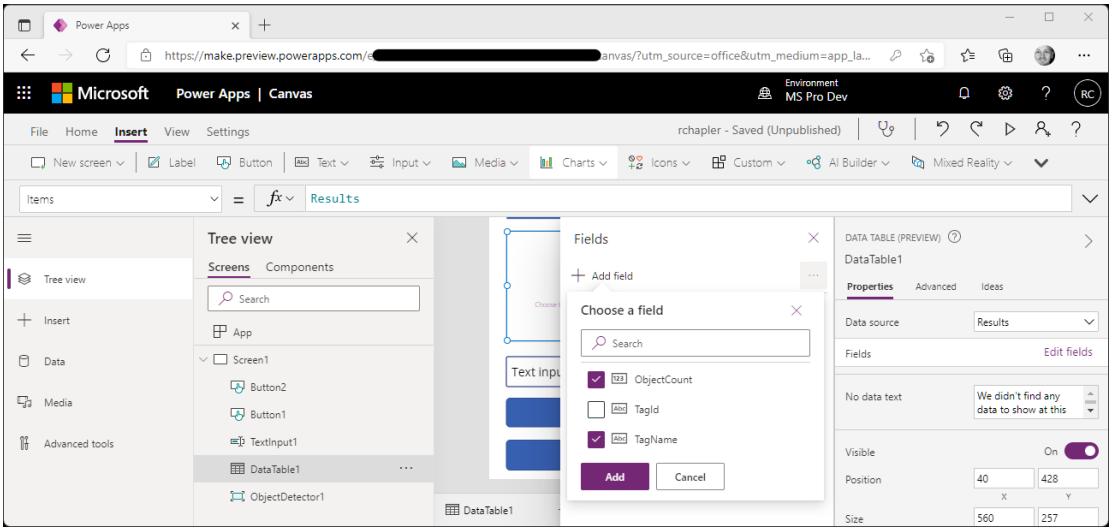
DataTable1



Click on **DataTable1** in the left-hand **Tree View**. On the resulting popup, select "**In your app**" | **Results**.



On the updated control, click the "**Choose the fields you want to add…**" link.

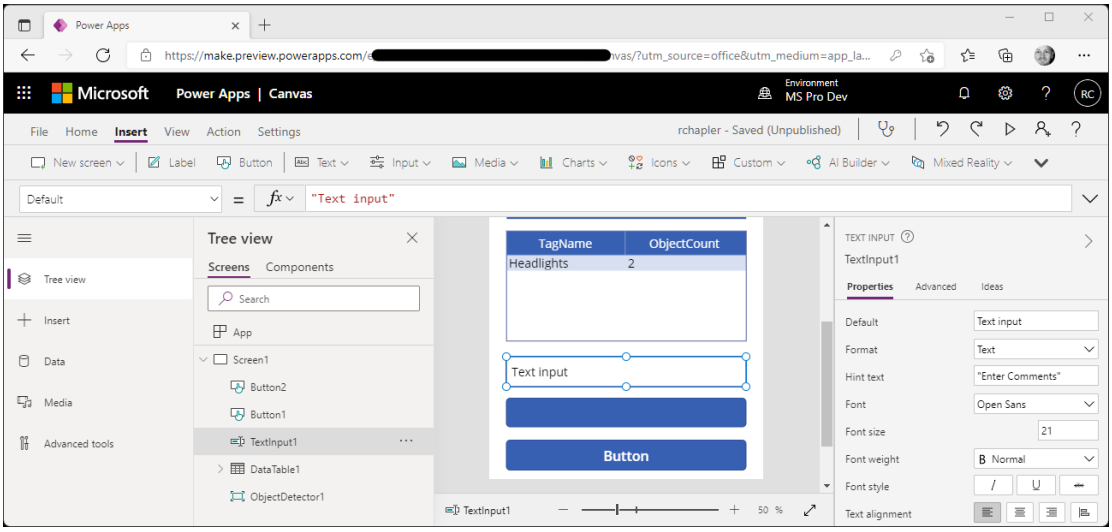


On the resulting **Fields** popout, click "**+ Add field**".

In the resulting "**Choose a field**" popup, check **TagName** and then **ObjectCount**.

Click the **Add** button.

TextInput1

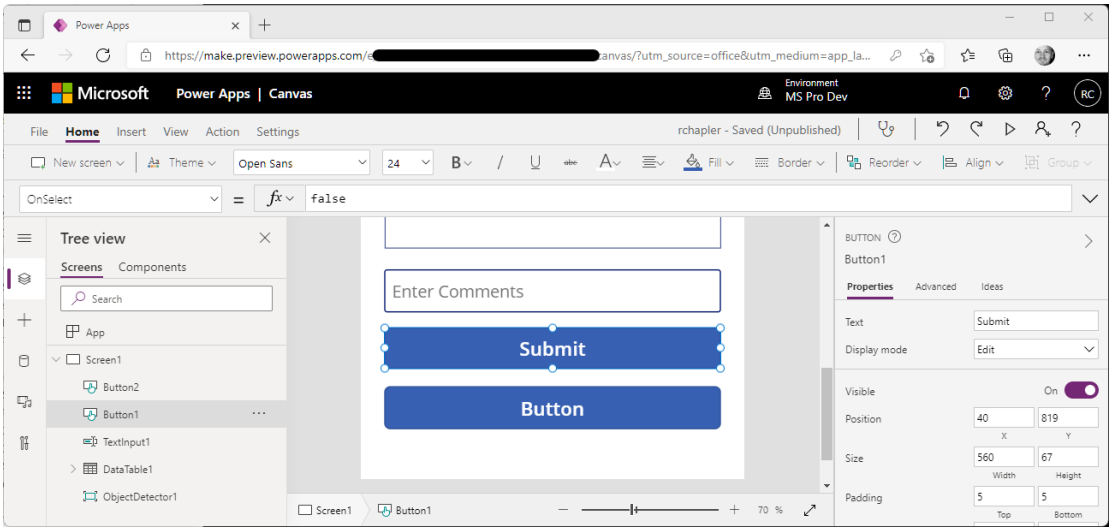


Click on **TextInput1** in the left-hand **Tree View**.

On the resulting "**TEXT INPUT**" popout, **Advanced** tab, enter the following items:

Default	""
Hint Text	"Enter Comments"

Button1



Click on **Button1** in the left-hand **Tree View**.

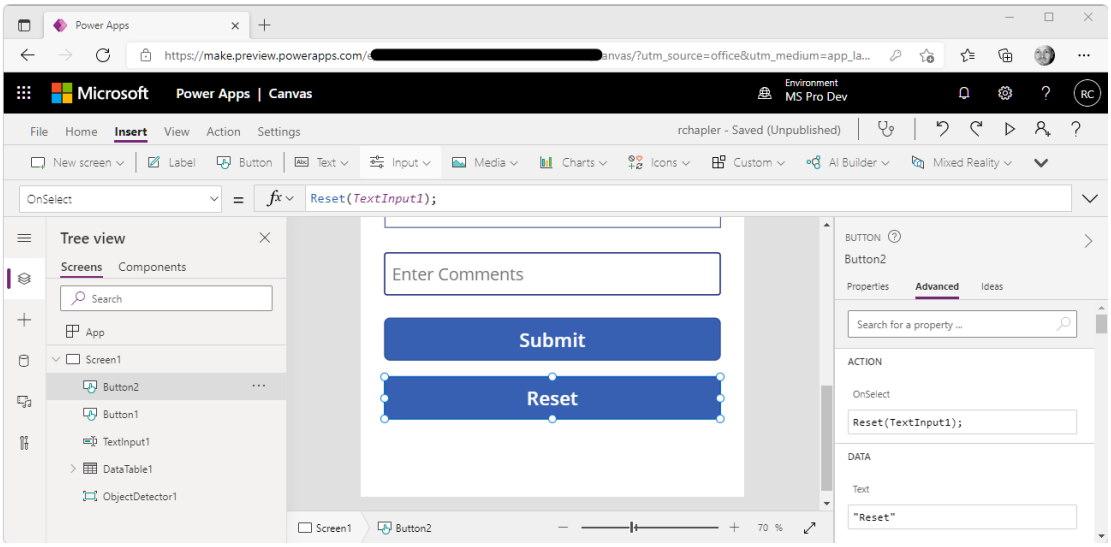
On the resulting right-hand popout, click the **Properties** tab, and enter the word “Submit” in the **Text** input.

On the resulting right-hand popout, 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 left-hand **Tree View**.

On the resulting right-hand popout, click the **Properties** tab, and enter the word “Reset” in the **Text** input.

On the resulting right-hand popout, click the **Advanced** tab,

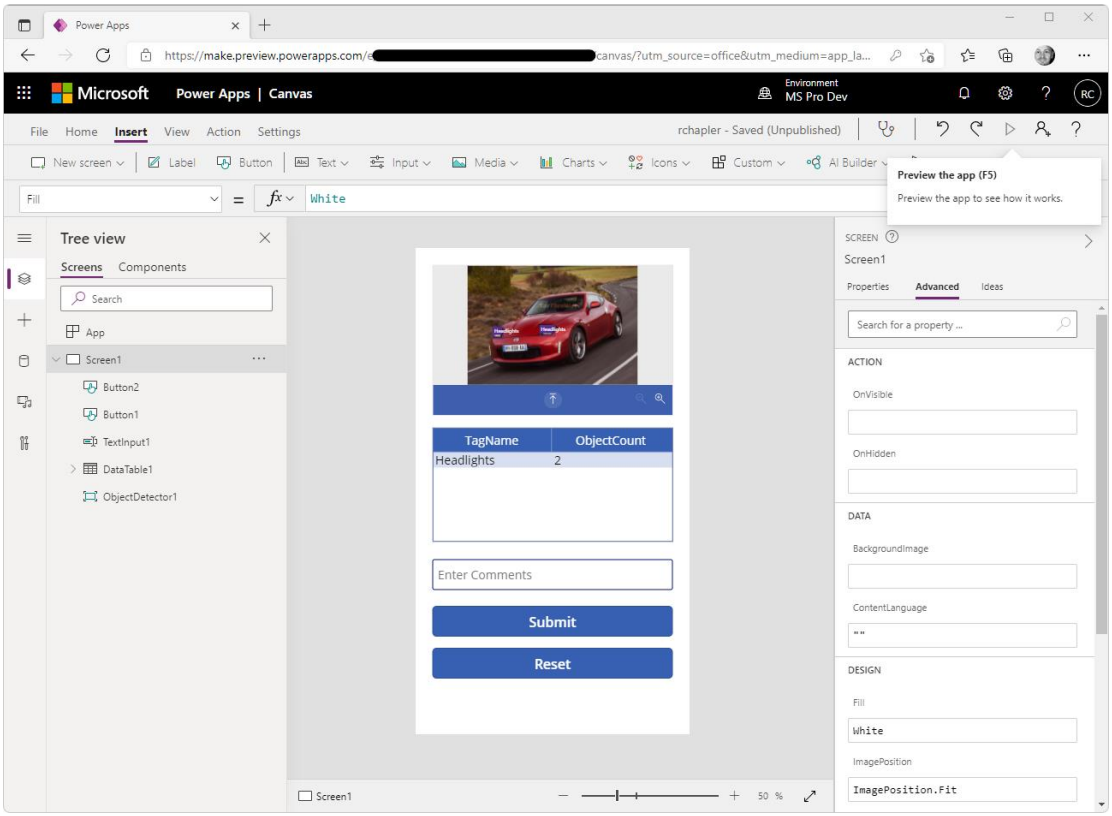
Paste the following logic into **Action** | **OnSelect**:

```
Reset(TextInput1);
```

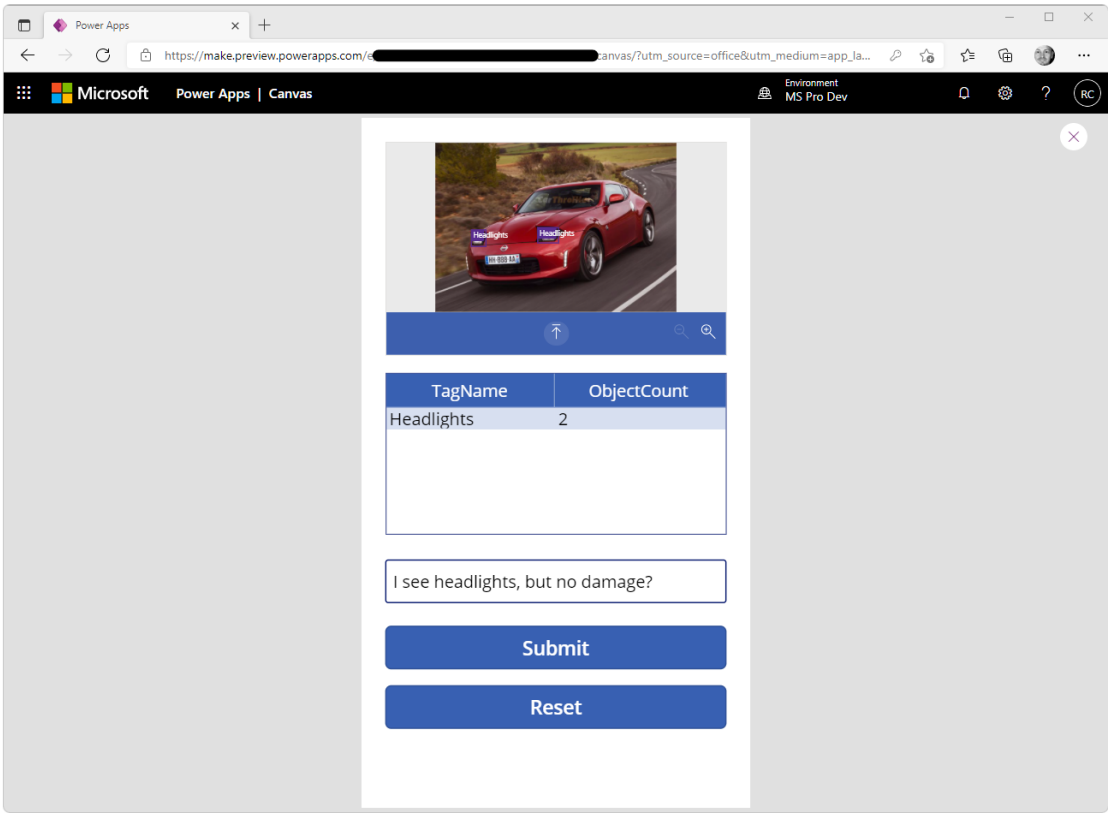
Step 9: Confirm Success

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 page, click the “**Preview the app**” button {i.e., Play icon} in the upper-right.



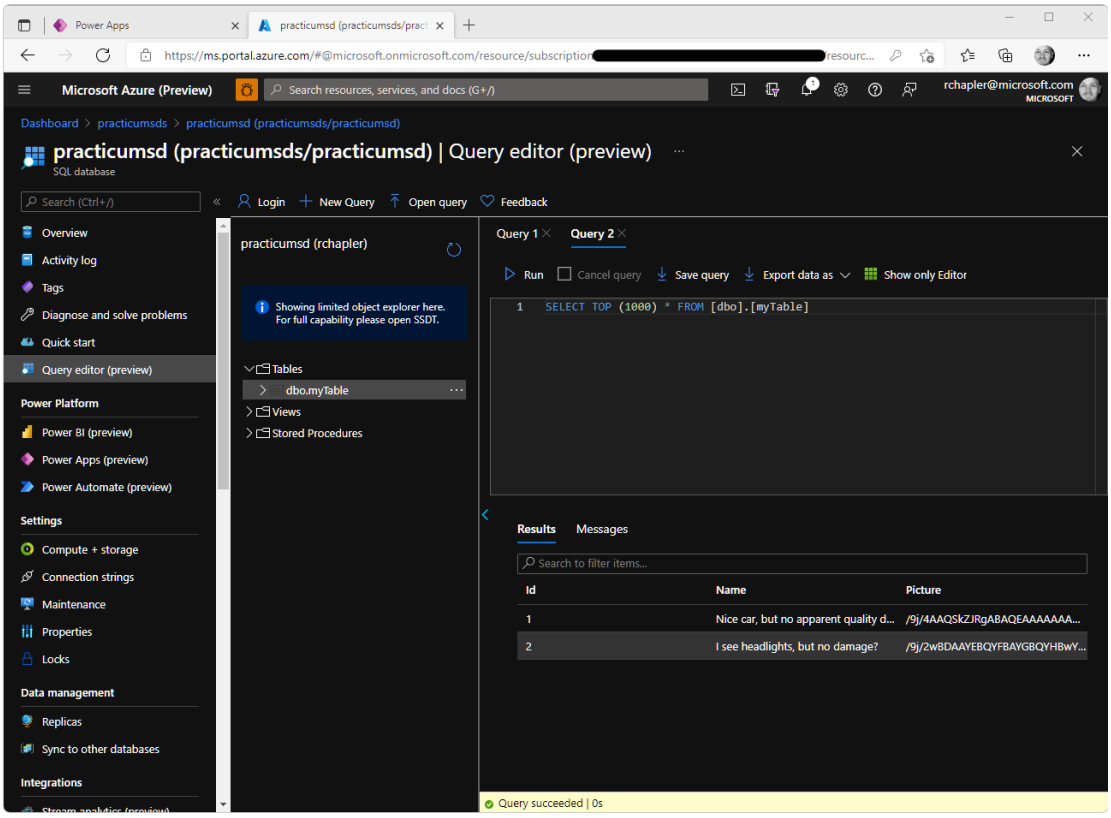
Your app will be presented in a phone simulation.



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.

Objective Complete... congratulations!

Audit Usage

Follow these instructions to support regular audit of and alerting on key data products.

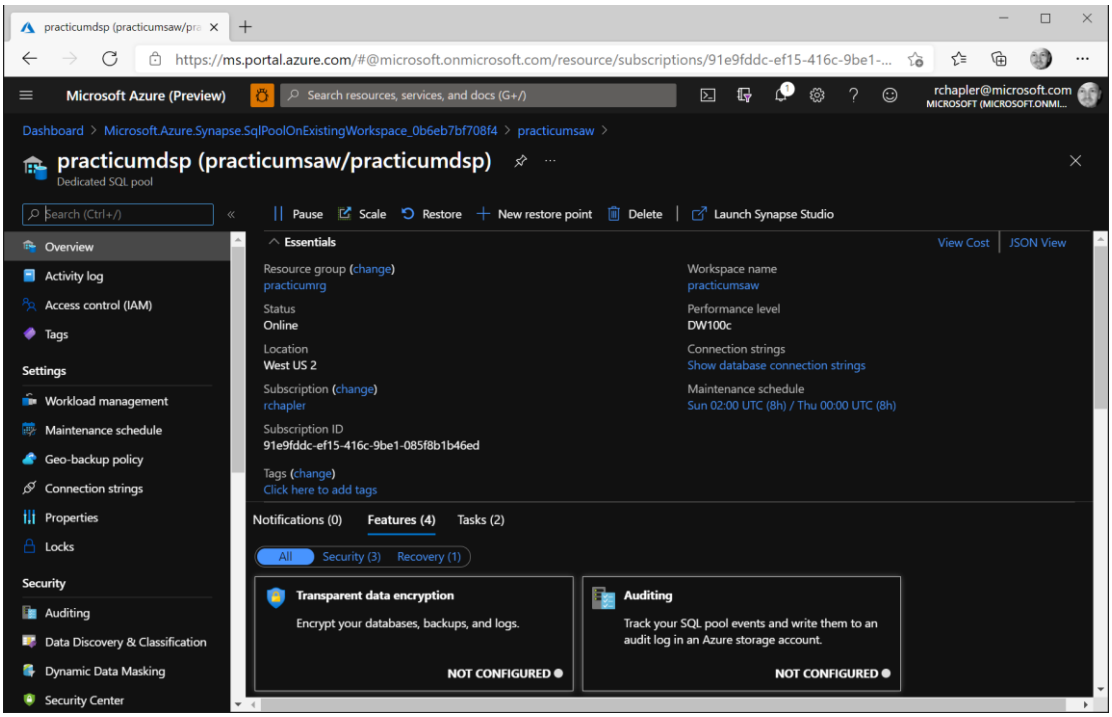
Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

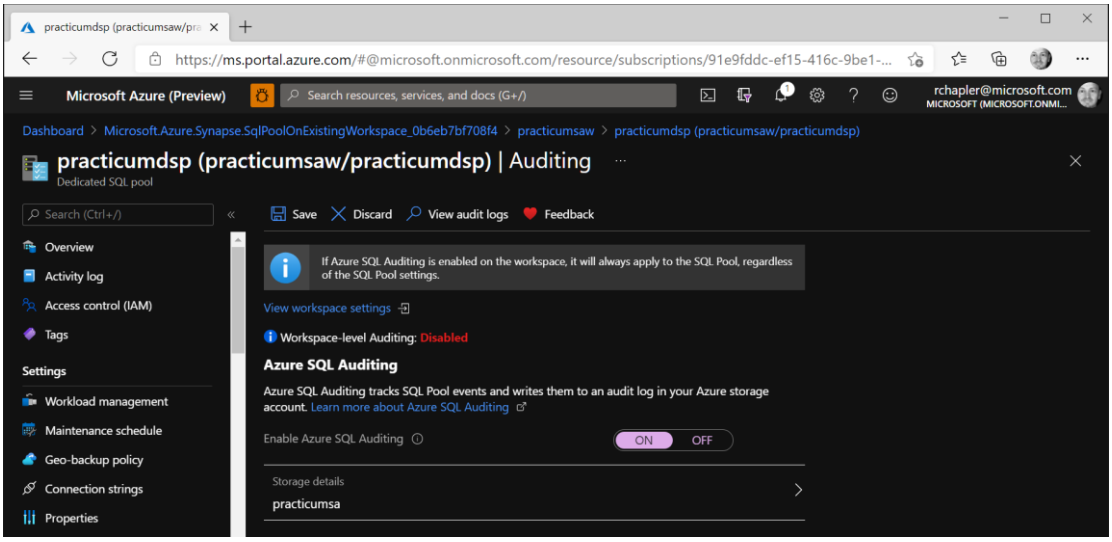
- Synapse

Step 2: Configure Auditing

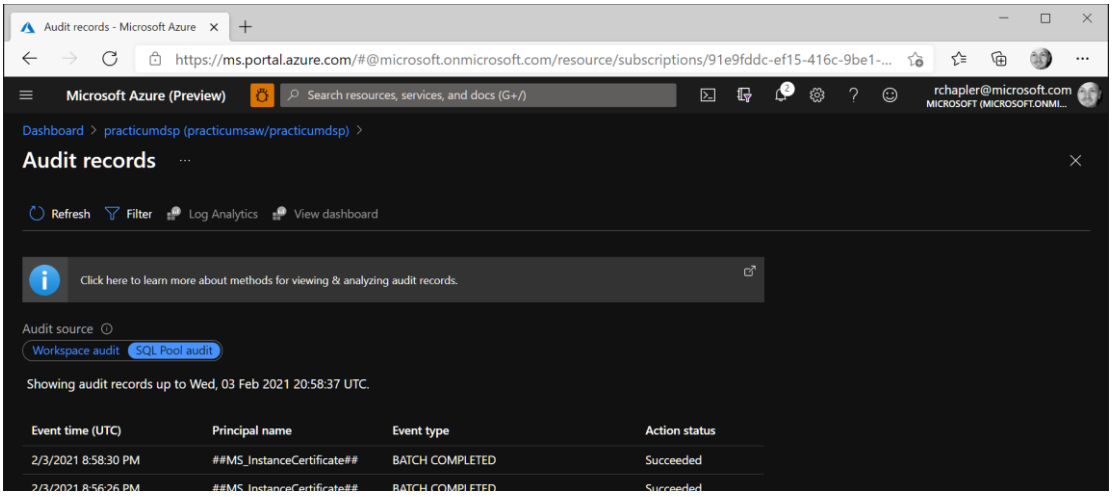
Navigate to the **Overview** page in Synapse, Dedicated SQL Pool.



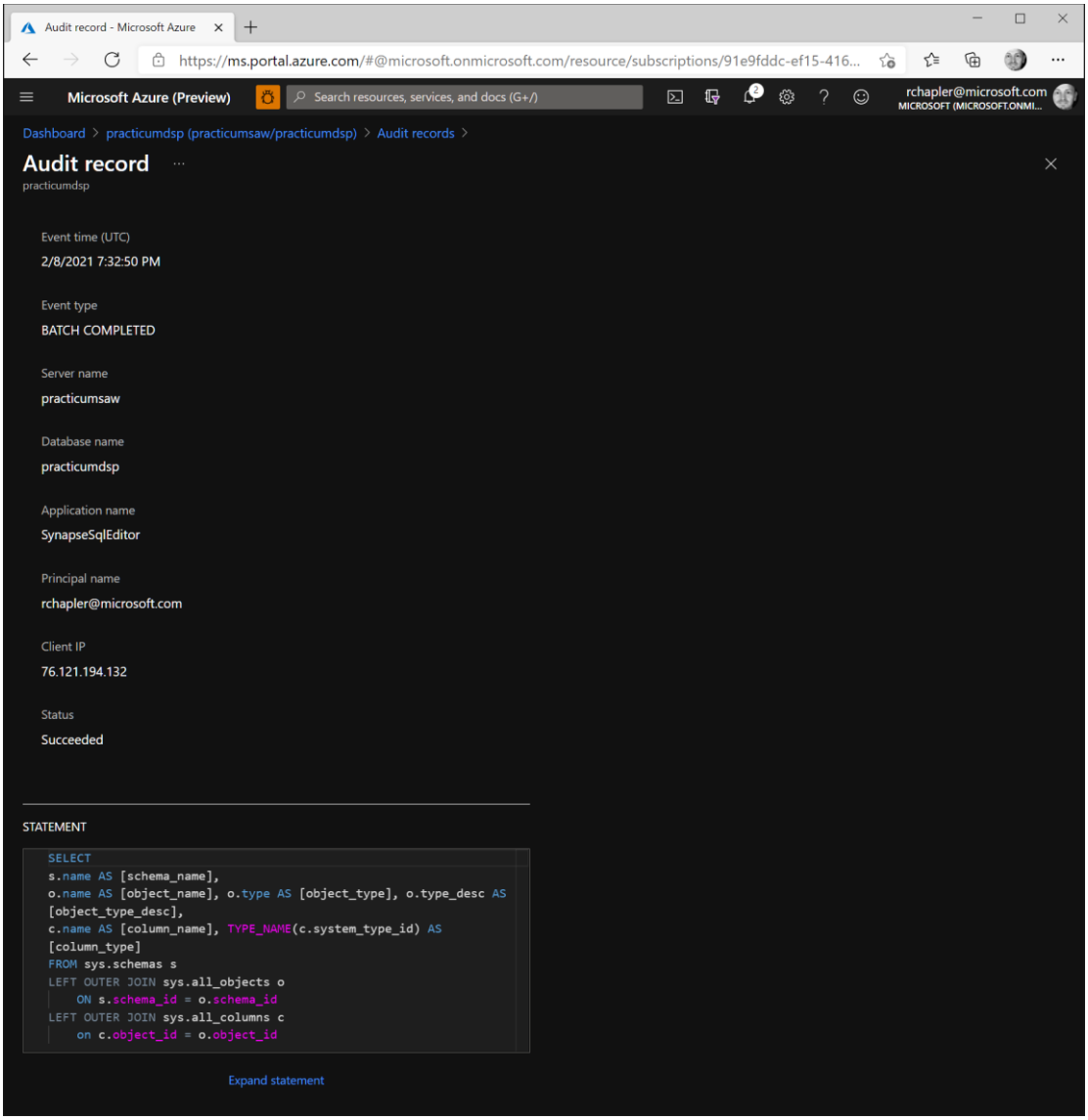
Click the **Auditing** button.



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.



Objective Complete... congratulations!

Deploy

The following sections describe automatic instantiation and deployment of the Azure resources necessary for the objectives in this book.

Infrastructure-as-Code

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

- "We want to **automate infrastructure deployment** to TEST and PROD environments"
- "We want to **protect our investment** in infrastructure / configuration with source control"

Step 2: Instantiate Prerequisites

This exercise requires either of the following resource(s):

- DevOps (with project and repository)
- GitHub Enterprise (with repository)
- Service Principal (with contributor permissions to the subscription)

Step 3: Create Template

In this section, we will prepare a bare-minimum template, and then manually deploy that template.

Create Custom Template

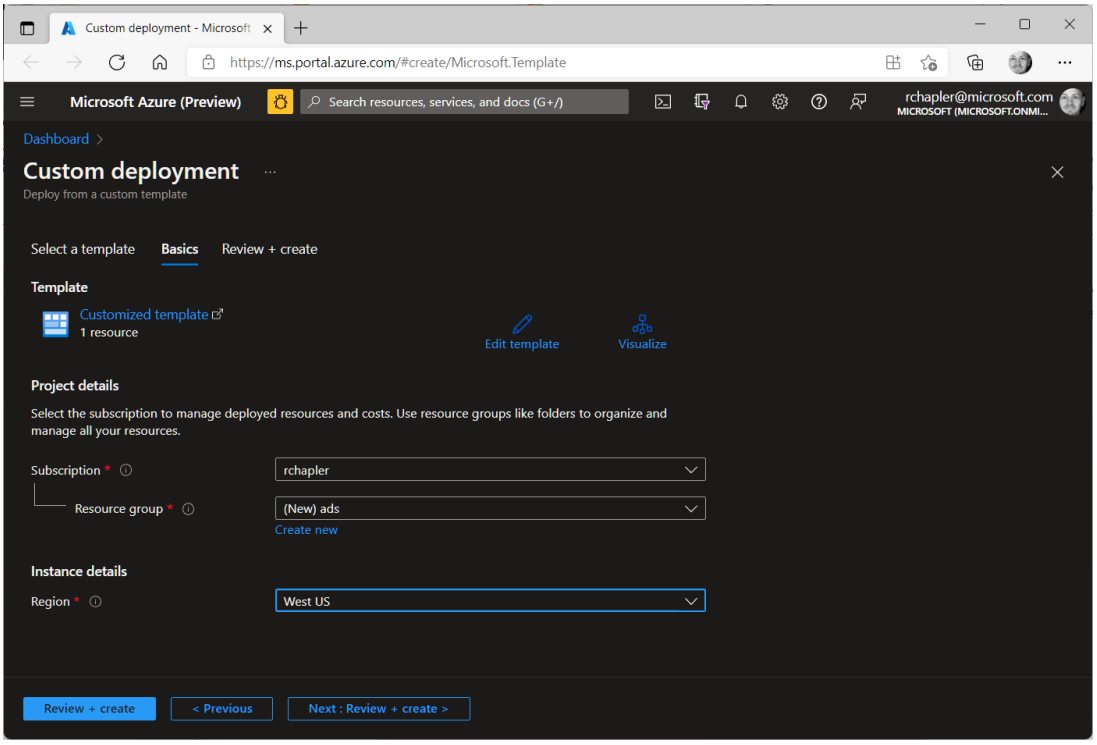
Navigate to https://ms.portal.azure.com/#create/Microsoft.Template and then click on the "**Build your own template in the editor**" link. On the resulting "**Edit template**" screen, replace the default JSON with the following prepared, "bare minimum" JSON:

```
{
  "$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"
    }
  ]
}
```

Special logic callouts…

[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.



On the “**Custom deployment**” page, enter values for the following items:

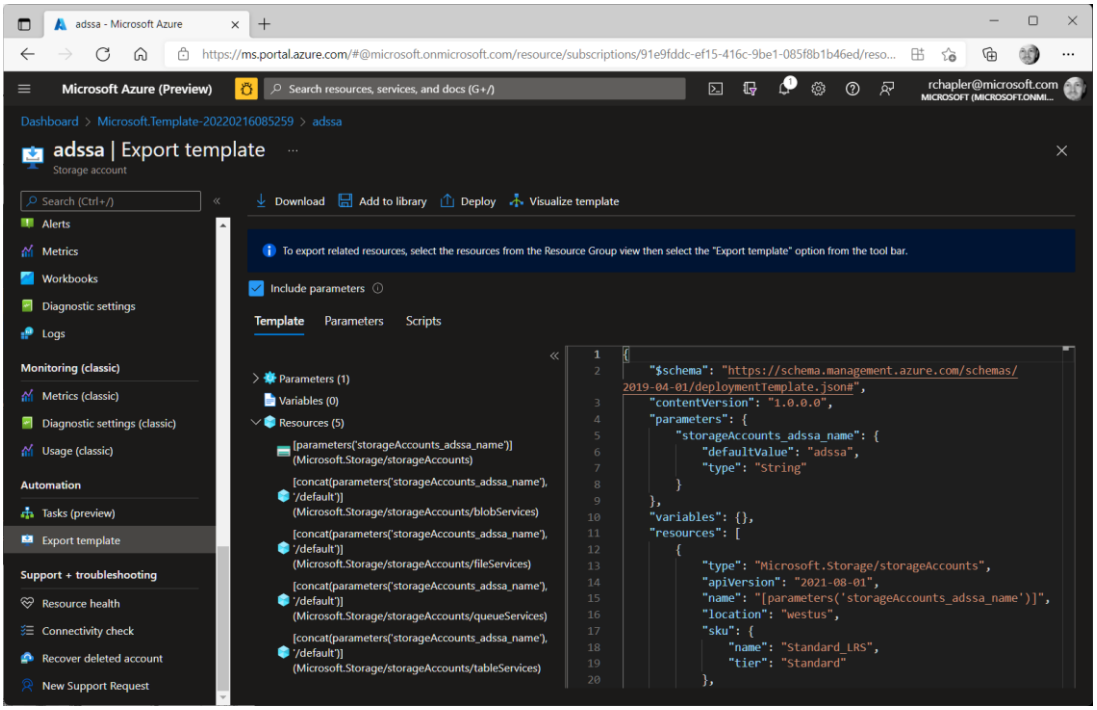
Resource Group	Select existing or “**Create new**”
Region	Select a region appropriate for your situation

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 from Existing Resource

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.

Step 4: Deploy Template

…via DevOps Pipeline

In this section, we will create, save, and run a DevOps Pipeline that deploys an ARM template.

Step 1: Upload Template

Upload the ARM Template to the DevOps repo by completing the following steps:

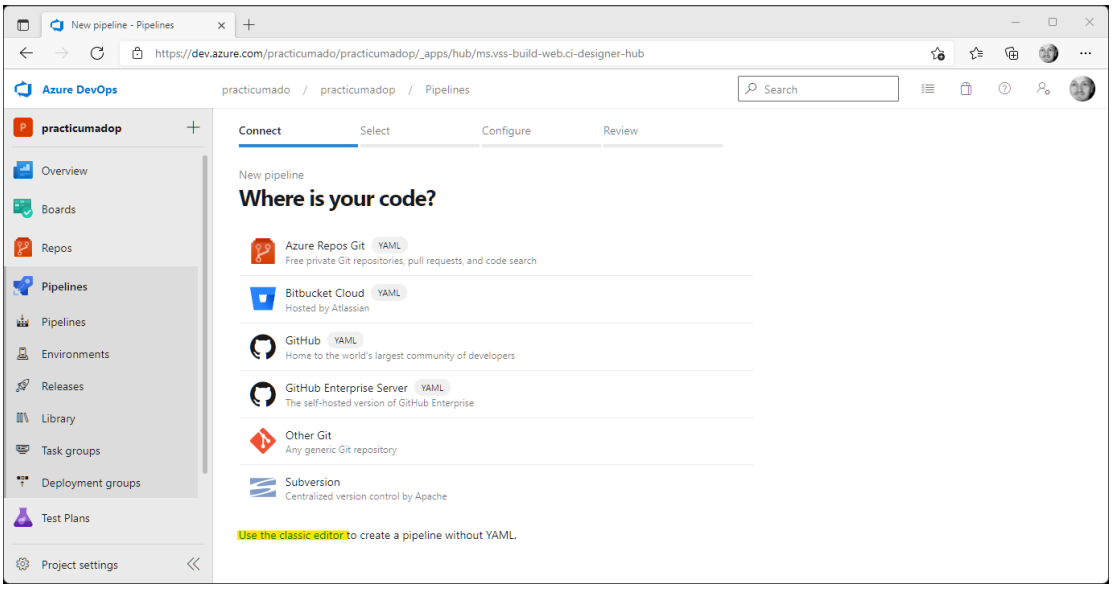
- Navigate to DevOps, select your project, and then click on **Repos** in the navigation pane
- Click the **+** icon just to the right of your project name and select "**New repository**" from the resulting dropdown
- In the resulting "**Create a repository**" popout, enter a "**Repository name**" {e.g., "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 dropdown
- 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.

Step 2: Create Pipeline

Navigate to DevOps, click on **Pipelines** in the navigation pane, and then click the "**New Pipeline**" button.

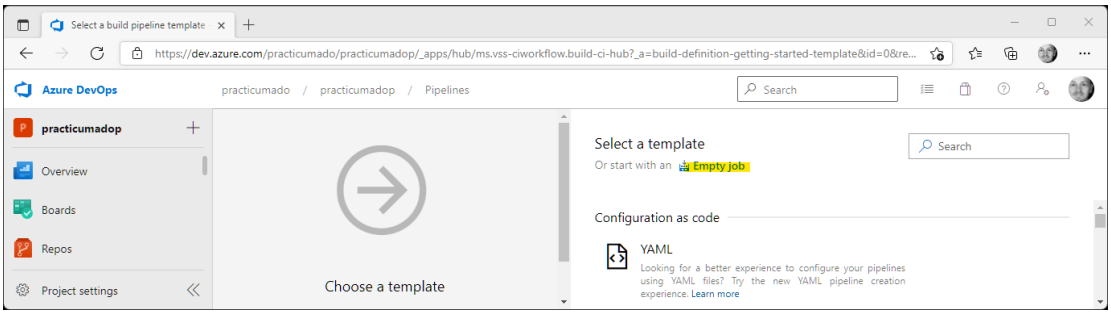


On the Connect tab {i.e., "New Pipeline … Where is your code?"} page, click the "**Use classic editor**" link.

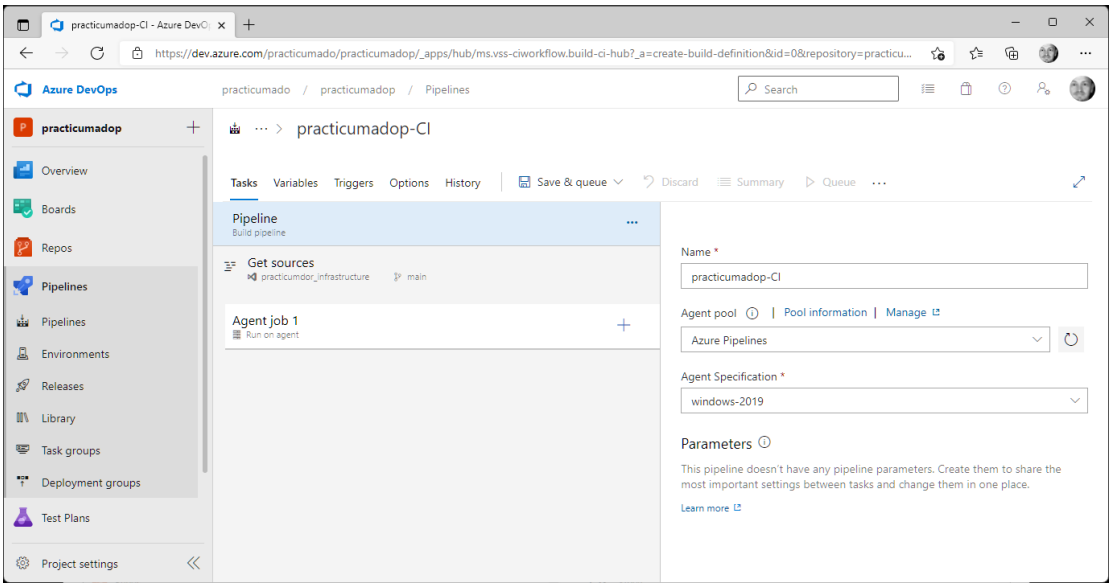
On the "**Select a source**" popout, select "**Azure Repos Git**" and then enter values for the following items:

Team Project	Select your DevOps project
Repository	Select your DevOps repository {e.g., **infrastructure**}
Default Branch…	Confirm default selection, **main**

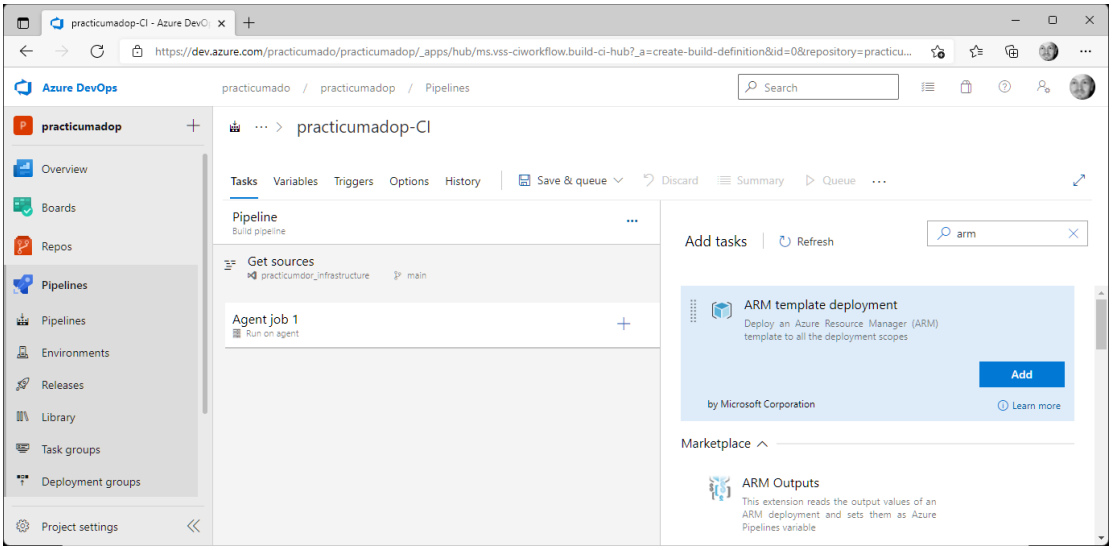
Click the **Continue** button.



Click “**Empty job**” on the “**Select a template**” popout.



Confirm default values for **Name**, **Agent Pool,** and **Agent Specification**.



Click the “**+**” button to the right of “**Agent Job 1**”.

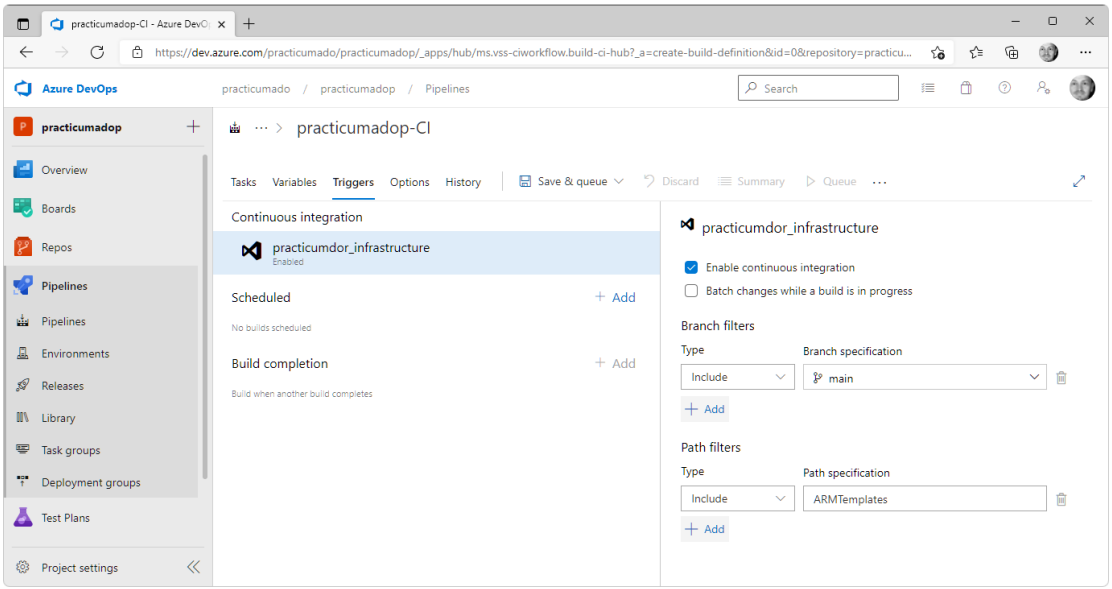
Search for and then select “**ARM template deployment**”.

Click the **Add** button.

On the “**ARM template deployment**” popout, enter values for the following items:

Deployment Scope	Confirm default, “**Resource Group**”
Azure Resource Manager…	Select your subscription
Subscription	Select your subscription
Action	Select “Create or update resource group”
Resource Group	Select your resource group
Location	Match resource group selection
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 dropdown.

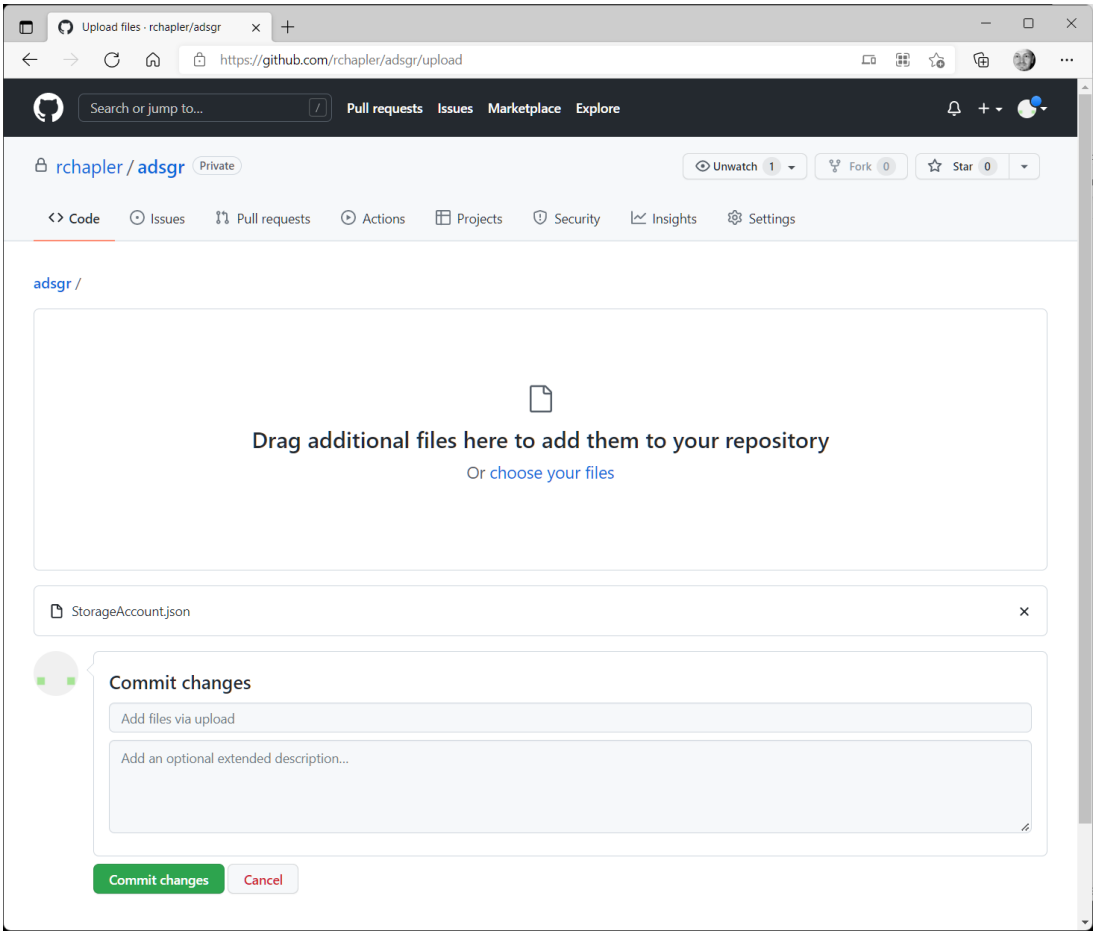
Confirm values on the resulting “**Run pipeline**” popout, and then click the “**Save and run**” button.

…via GitHub Action

In this section, we will create, save, and run a GitHub Action that deploys an ARM template.

Step 1: Upload Template

Modify (with your values) and navigate to the following URL:
https://github.com/rchapler/adsgr/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.

Step 2: 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": "{Subscription GUID}",
"tenantId": "{Tenant GUID}"
}
```

You should use the values produced when you created the Service Principal.

Step 3: Create Secret

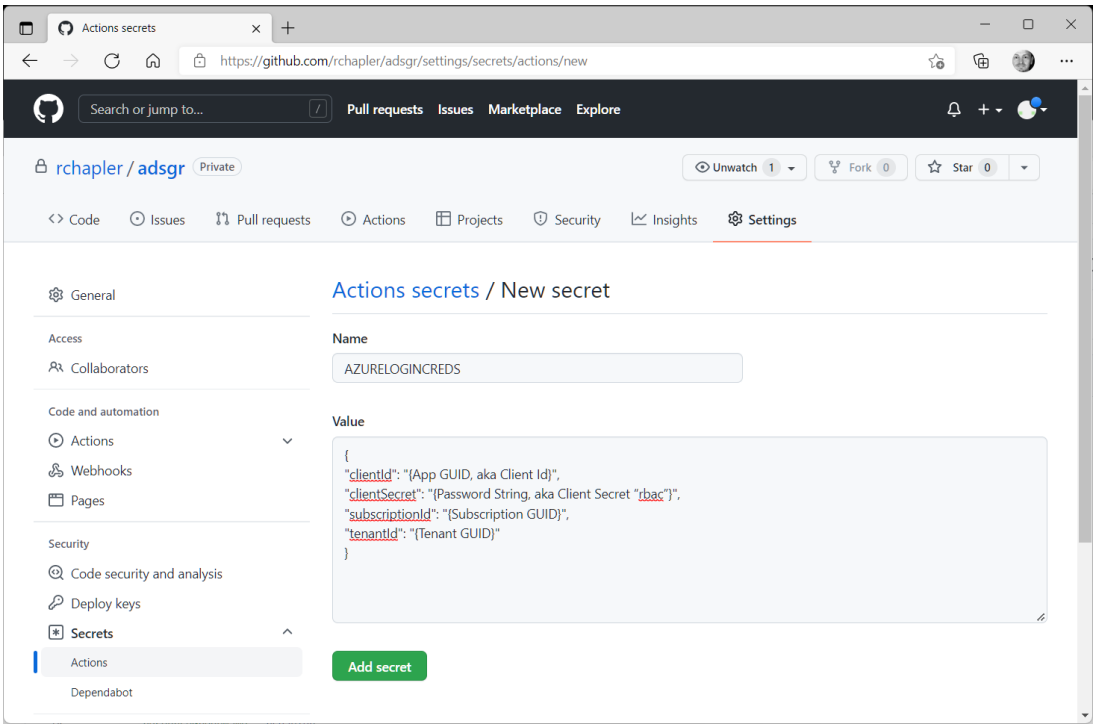
Modify and navigate to the following URL:

https://github.com/rchapler/shared/settings/secrets/actions.

Click the "**New repository secret**" button.

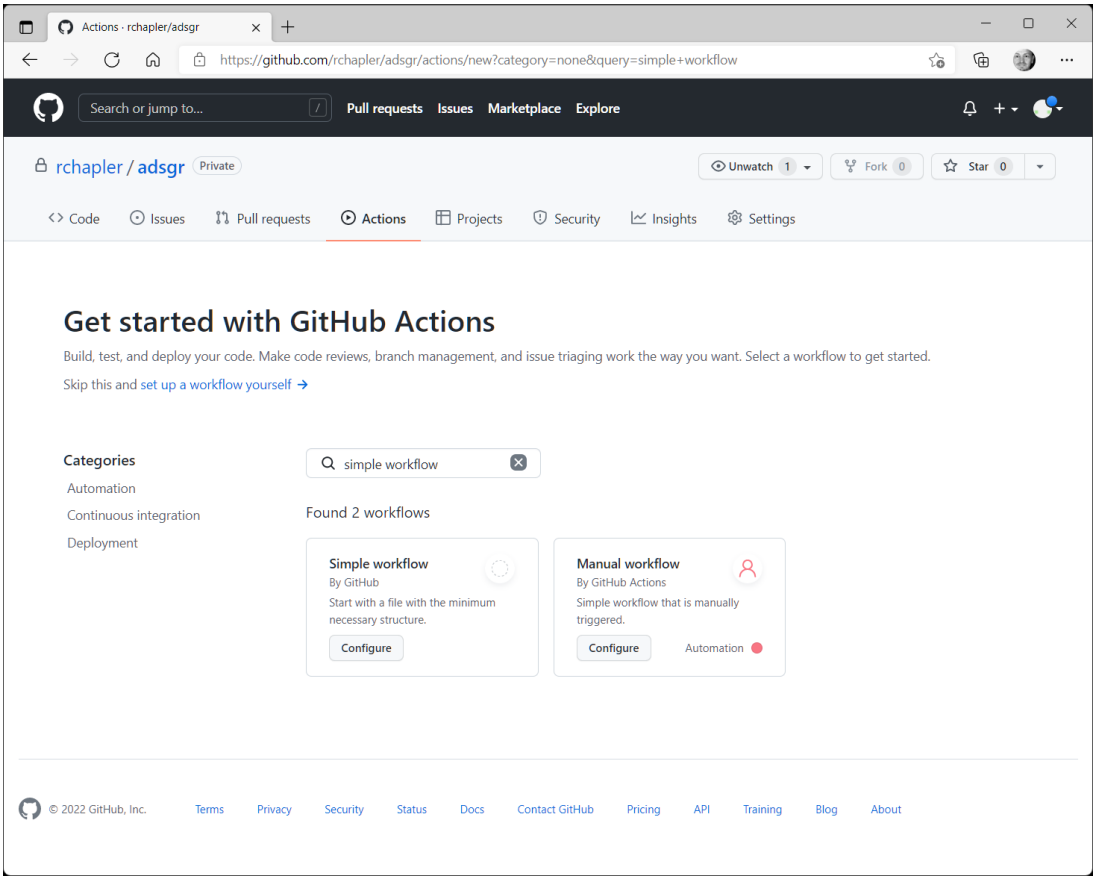
On the "**Actions secrets / new secret**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Value	Paste the credential string prepared in the prior step

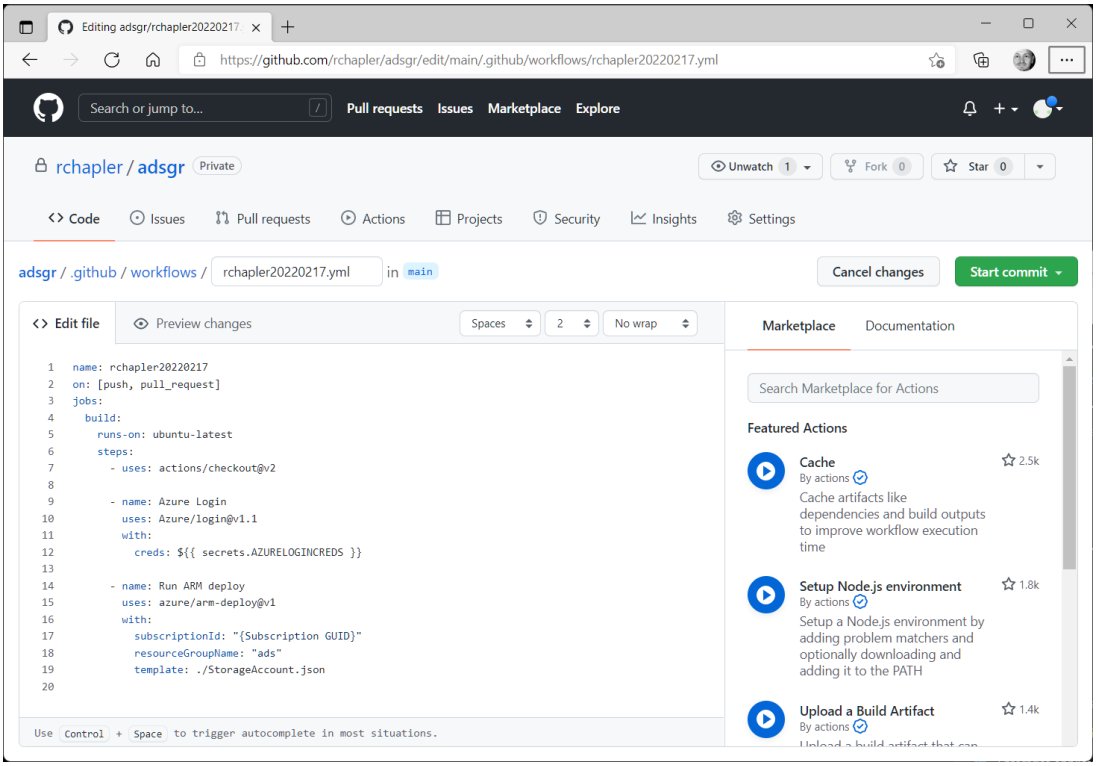


Click the "**Add secret**" button.

Step 4: Create Workflow



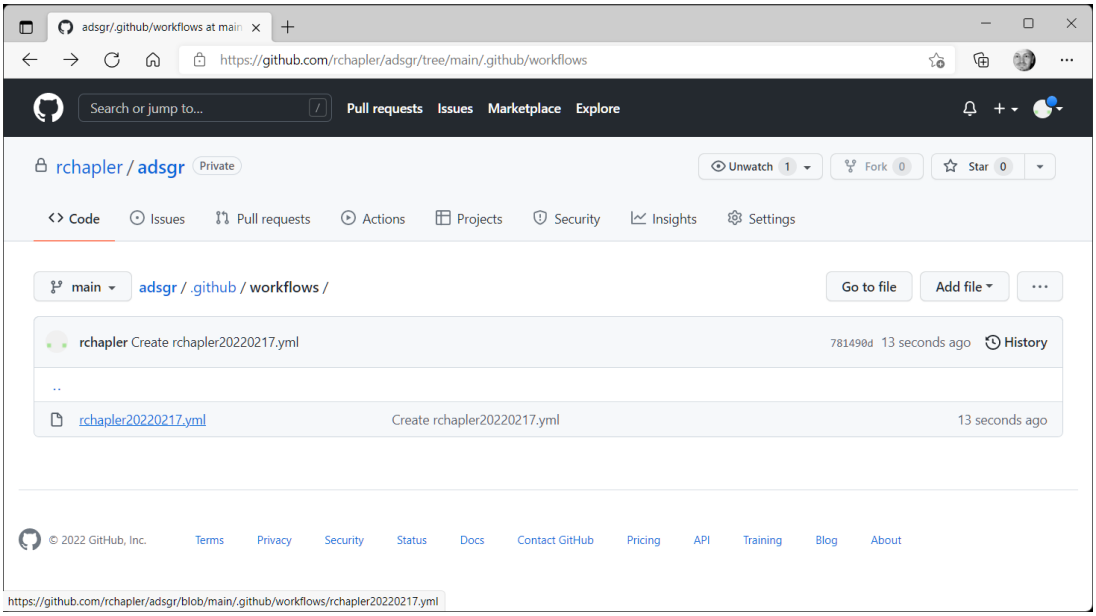
Click **Actions** in the menubar, search for "**Simple workflow**", and then click the **Configure** button.



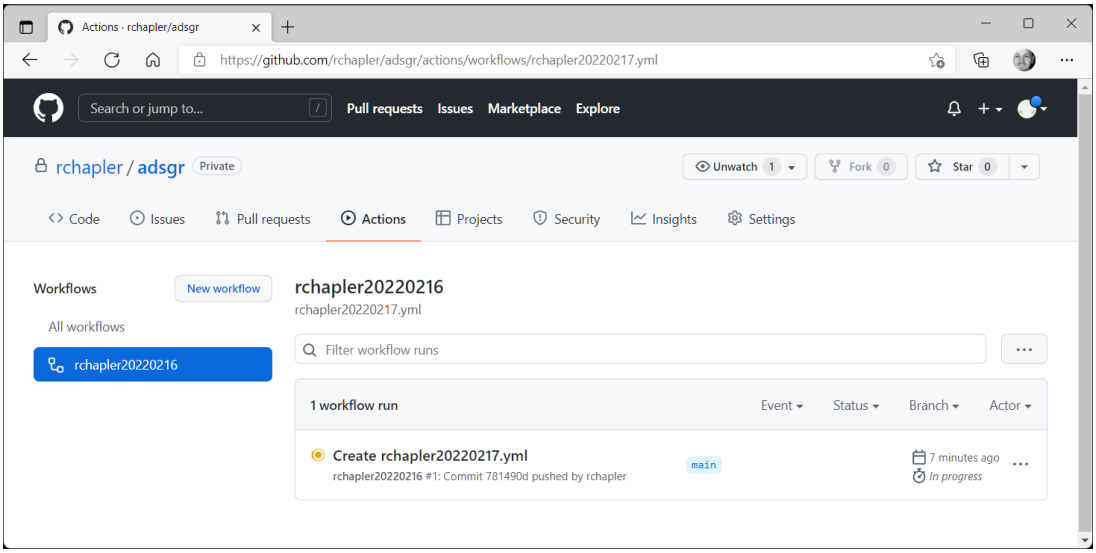
Enter a meaningful name, then modify and paste the following YML.

```
name: rchapler20220216
on: [push, pull_request]
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v2

   - name: Azure Login
    uses: Azure/login@v1.1
    with:
     creds: ${{ secrets.AZURELOGINCREDS }}

   - name: Run ARM deploy
    uses: azure/arm-deploy@v1
    with:
     subscriptionId: "{Subscription GUID}"
     resourceGroupName: "ads"
     template: ./StorageAccount.json
```

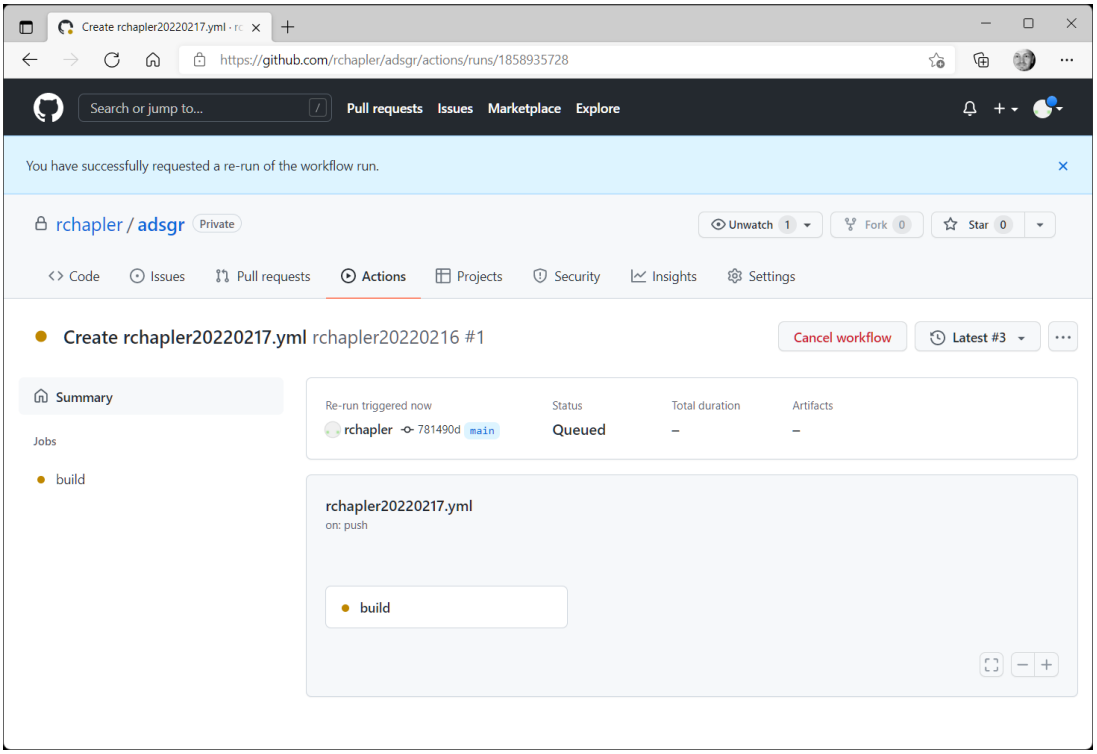
Click the “**Start commit**” button and then the “**Commit new file**” button on the resulting popup.



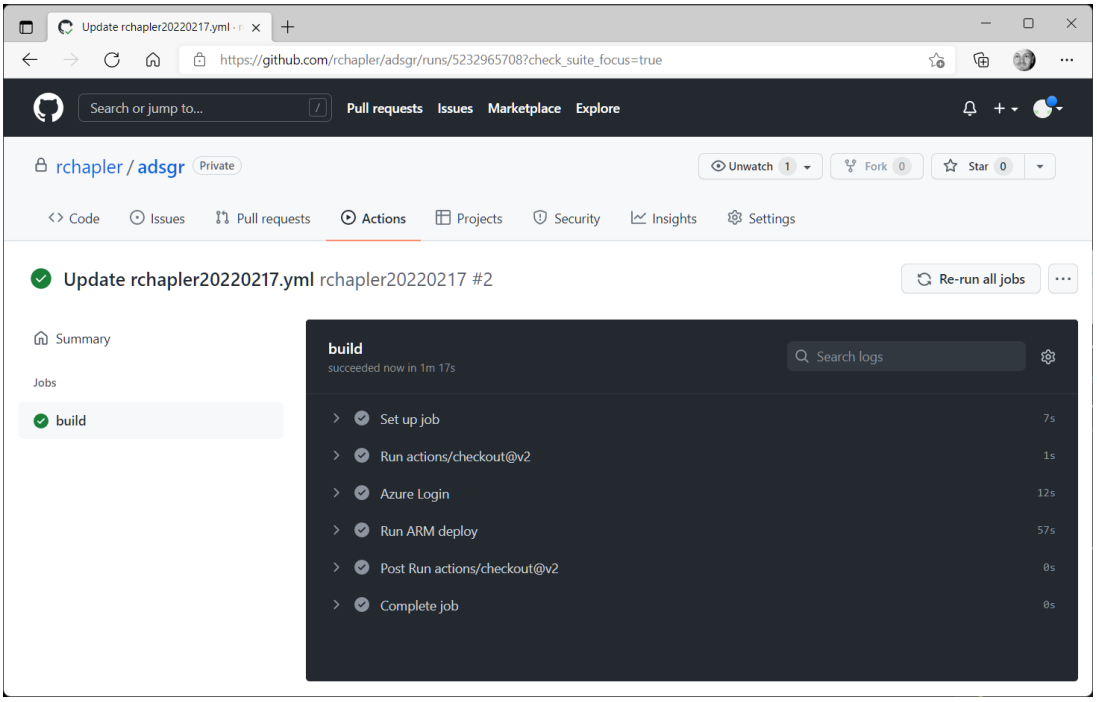
On the resulting page, click on the link for the committed YML file.



Click on the run {i.e., “Create rchapler20220217.yml”}.

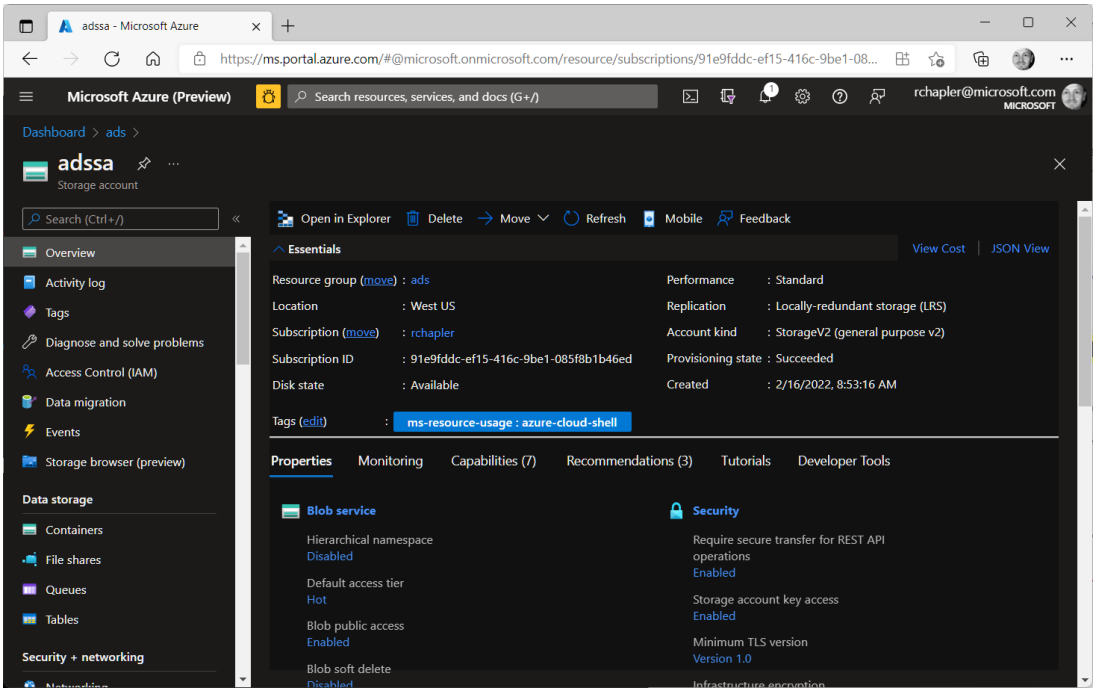


Click on the **build** box.



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.



Objective Complete... congratulations!

Step 5: Incorporate Additional Templates

The templates in this section…

- …are “bare bones” {i.e., include only the minimum amount of code necessary for instantiation}
- …use the latest API version rather than a specific dated version; as such, schema might require update over time

File Sync (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"
      }
    }
  ]
}
```

Storage Account

```
{
  "$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

```
{
  "$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]"
    }
  }
}
```

Source Control

Follow these instructions to connect Synapse to DevOps, create branches, and pull requests.

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

- "We need multiple developers to collaborate on parts of a large use case"

Step 2: Instantiate Prerequisites

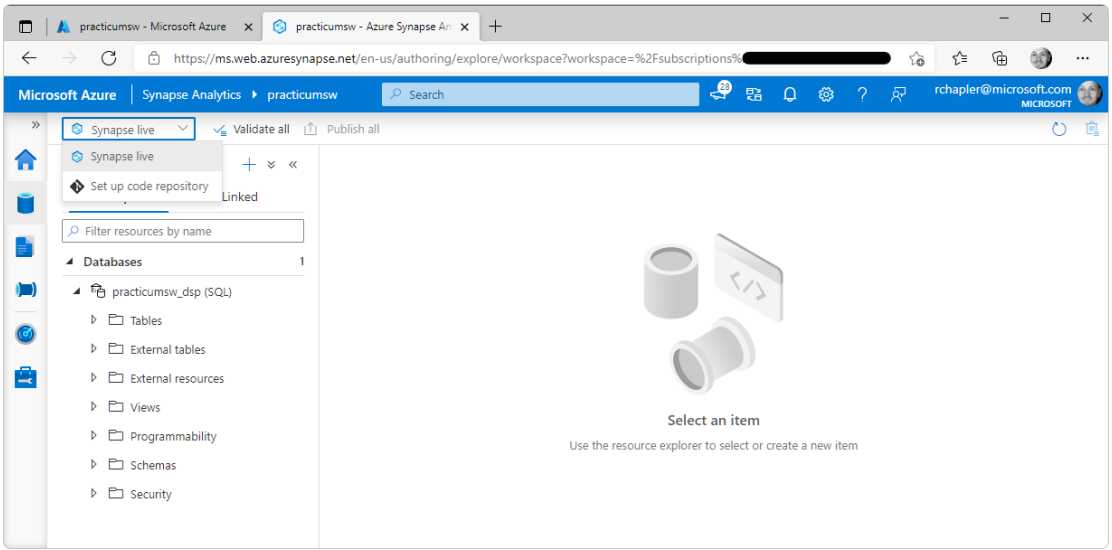
This exercise requires the following resource(s):

- DevOps
- Synapse (with Dedicated SQL Pool)

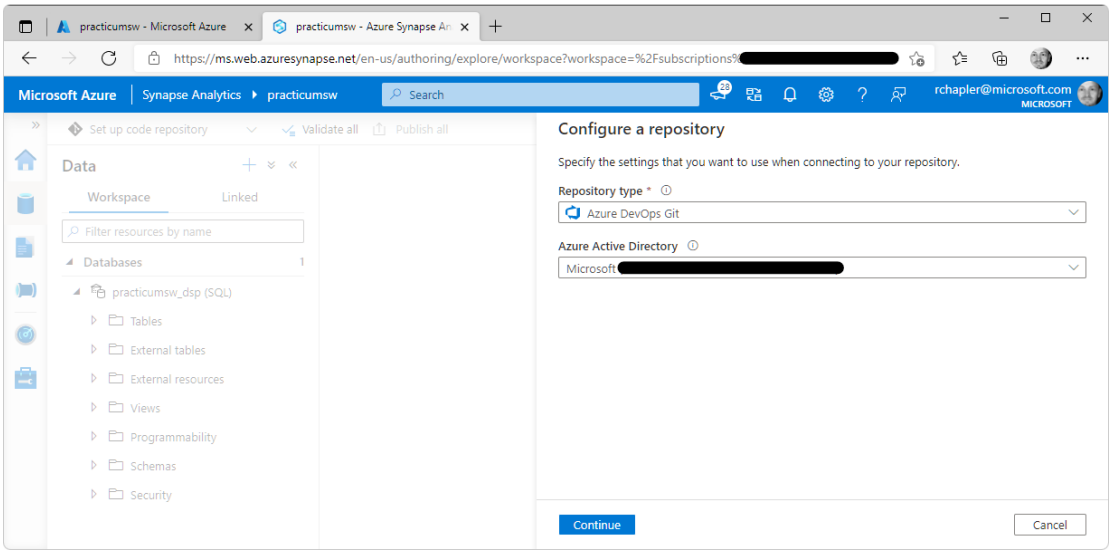
Note: These instructions also apply (with minor differences) to Azure Data Factory.

Step 3: Setup Code Repository

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



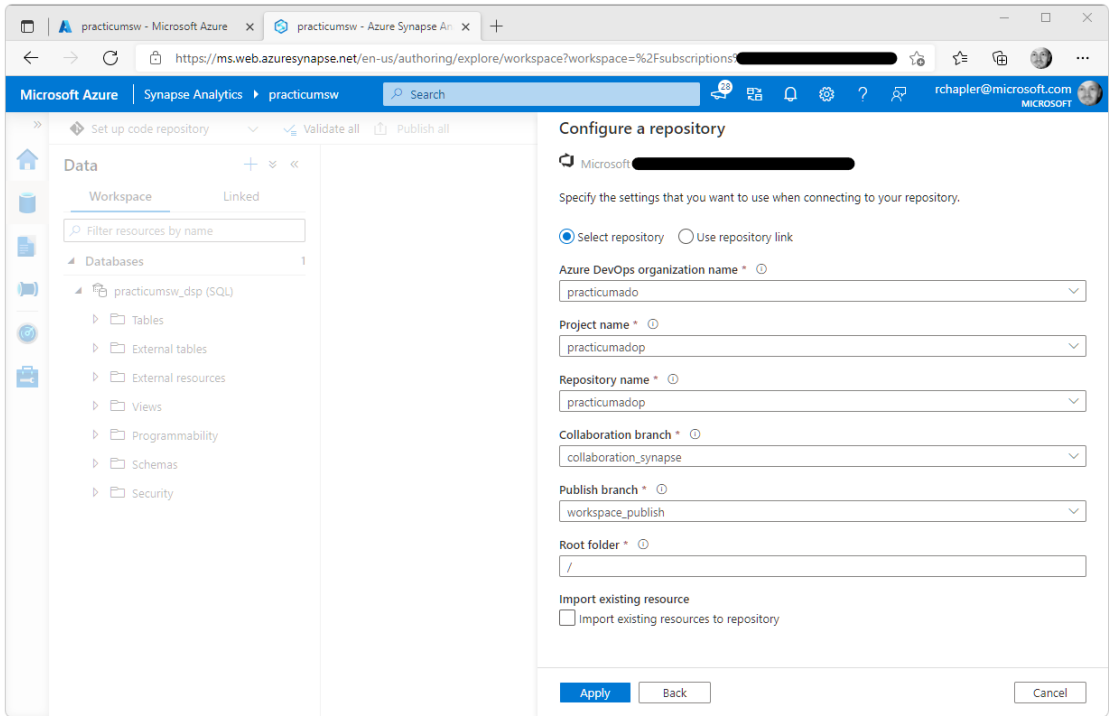
Click "**Synapse live**" and then select "**Set up code repository**" from the resulting dropdown.



On the "**Configure a repository**" popout, enter values for the following items:

Repository Type	Select "**Azure DevOps Git**" from the dropdown
Azure Active Directory	Select the value appropriate for your organization

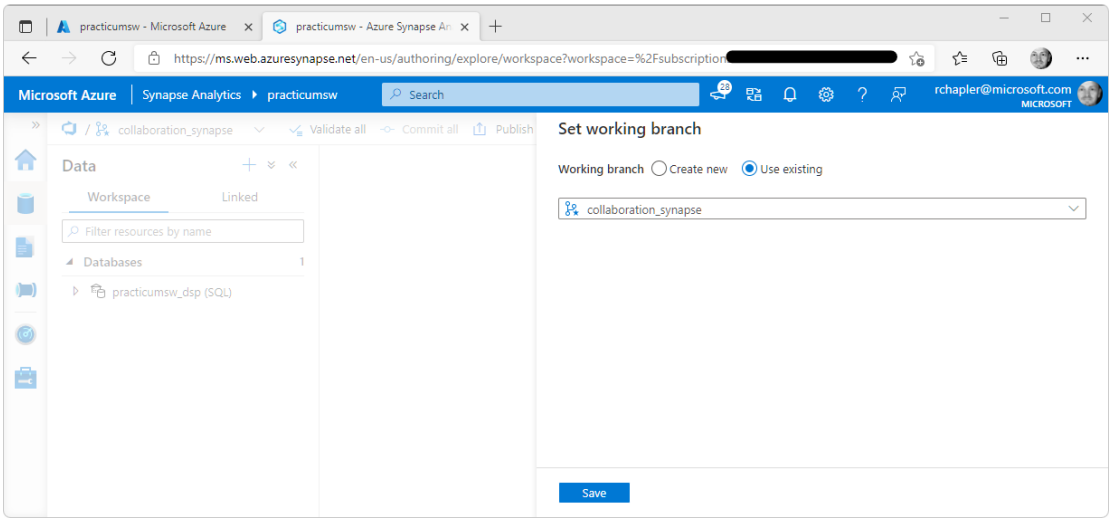
Click the **Continue** button.



On the second "**Configure a repository**" popout, enter values for the following items:

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	Dropdown, click "**+ Create new**", enter an appropriate name in the resulting popup, and click **Create**
Publish Branch	Confirm default, **workspace_publish**
Root Folder	Confirm default, **/**
Import Existing Resources…	Unchecked

Click the **Apply** button.



Confirm values and then click the **Save** button.

When processing is complete, you will note that we are now working in the new branch.

Step 4: Confirm Success

In this section, we will confirm success by simulating a deployment workflow {i.e., modification, and then pull request}.

Modify one or more of the following:

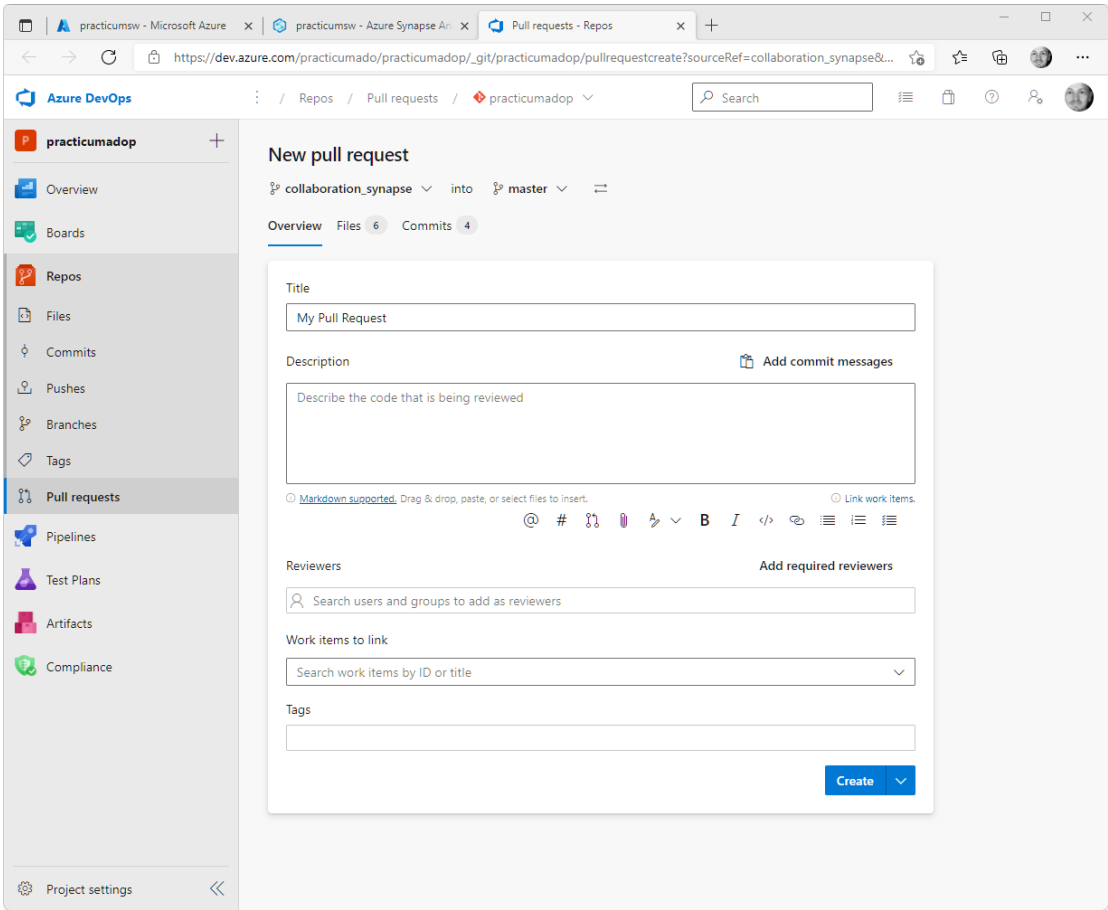
- SQL Script
- Spark Notebook
- Data Flow
- Pipeline

Pull Request

Next, we will promote changes from the collaboration branch to the master branch via Pull Request.

Click on the "**{collaboration} branch**" dropdown control and select "**Create pull request**…" from the resulting dropdown.

A third tab in your browser (pointing to https://dev.azure.com...) will open.



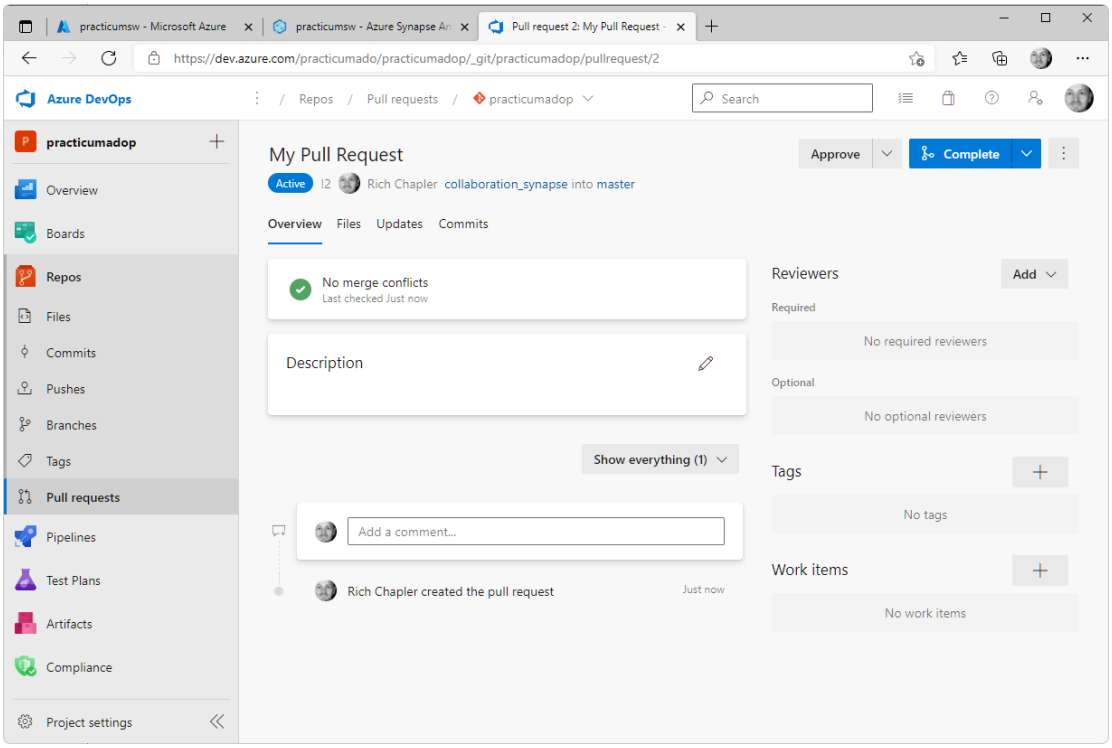
On the second "**New pull request**" page, enter values for the following items:

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)

Enter values for items like **Reviewers** and "**Work items**…" as appropriate.

No additional changes are required. Review settings on the remaining tabs.

Click the **Create** button.

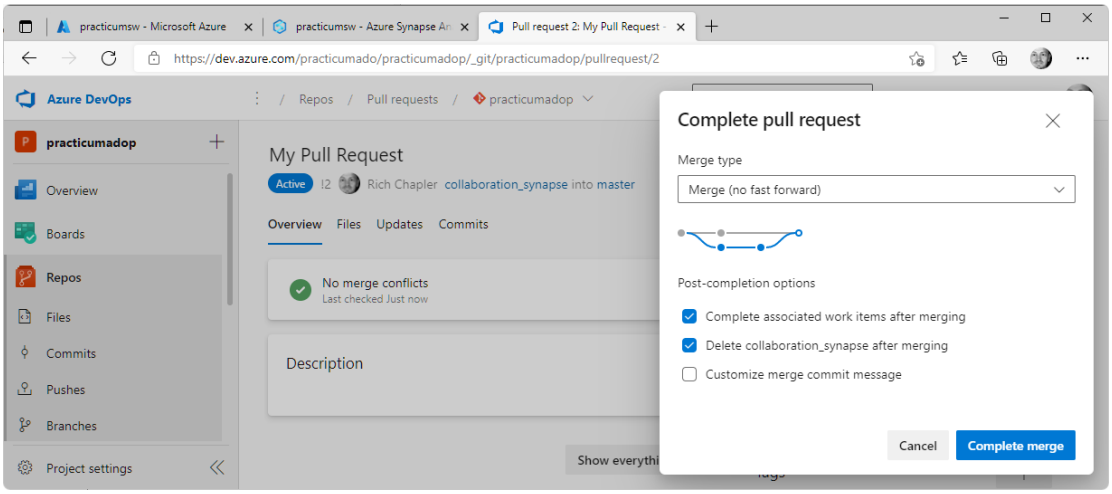


At this point (and if specified), designated reviewers would be notified by email, and they will need to review and approve for the Pull Request to move forward.

Additional configuration can provide for validation of other gating criteria {e.g., inclusion of text patterns that might be secret like an account key}.

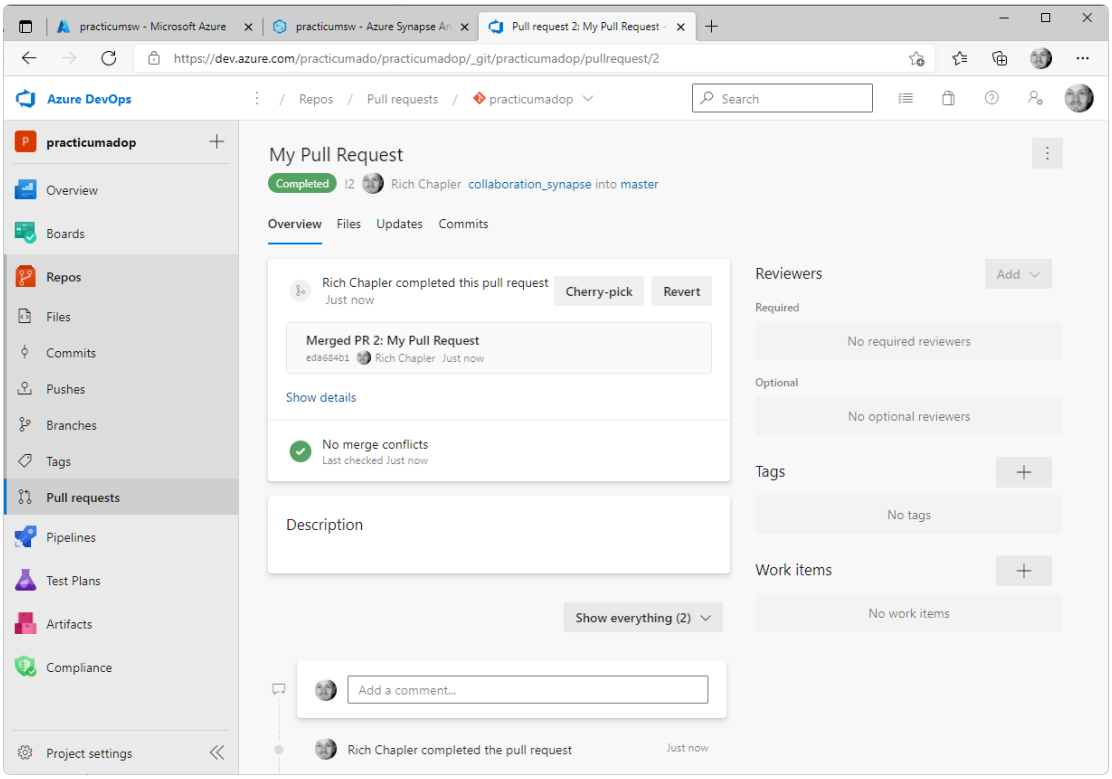
We have not included gating factors in this example.

Click the **Complete** button.



Confirm selections in the “**Complete pull request**” popout.

Click the “**Complete merge**” button.



Note: Post-completion options such as “Cherry-pick” and Revert provide for operational control even after a code merge.

Objective Complete… congratulations!

Schema Comparison

Follow these instructions to **compare two database schemas**.

Step 1: Understand Use Case

Work with your customer to understand requirements; example notes:

- “We need to articulate changes to the database schema in the development environment (as compared to the database schema in the production environment)”

Step 2: Instantiate Prerequisites

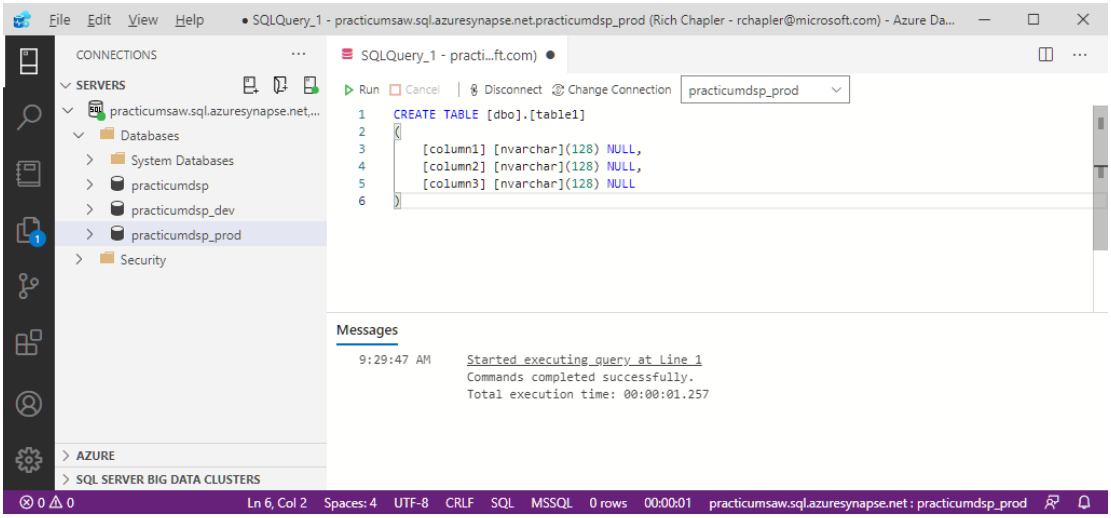
This exercise requires the following resource(s):

- Data Studio
- Synapse (with two SQL pools named **practicumdsp_dev** and **practicumdsp_prod**)

Although this example focuses on Synapse, the same solution applies to SQL.

Step 3: Stage Resources

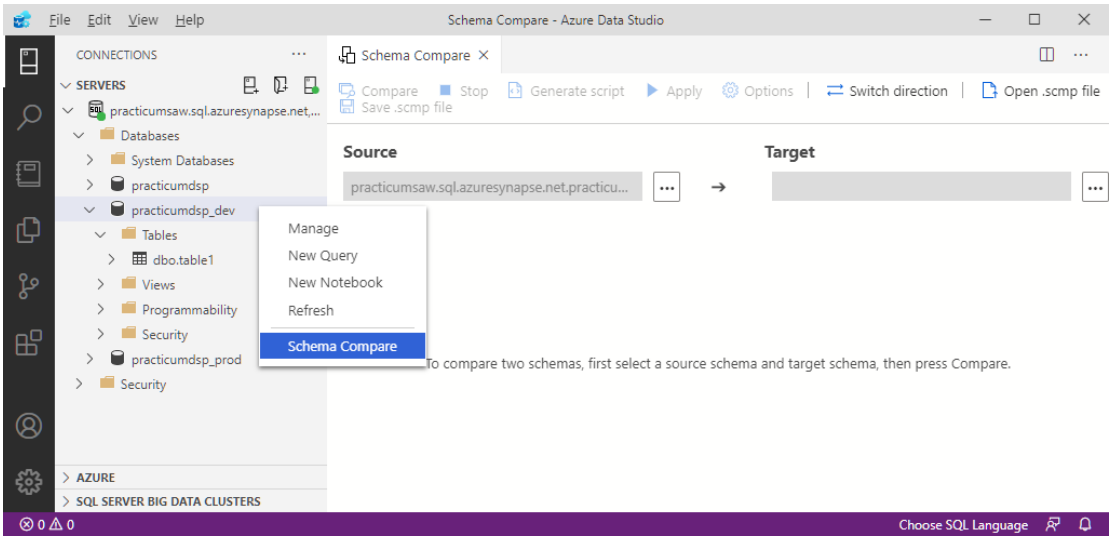
Open Azure Data Studio.



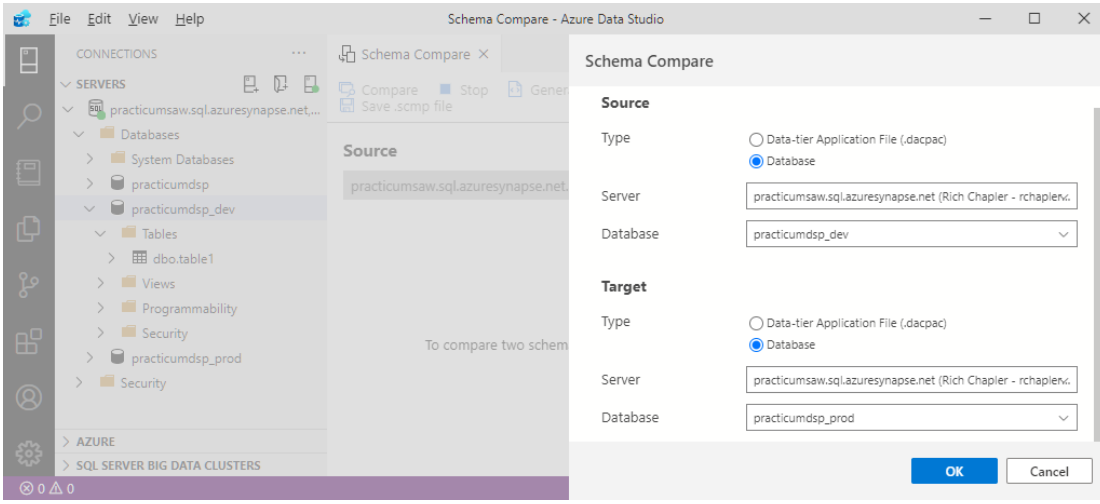
Execute the following T-SQL on practicumdsp_dev:

```
CREATE TABLE [dbo].[table1]
(
        [column1] [nvarchar](128) NULL,
        [column2] [nvarchar](128) NULL,
        [column3] [nvarchar](128) NULL
)
```

Step 4: Schema Compare



Right-click on the SQL Pool and click "**Schema Compare**" in the resulting dropdown.



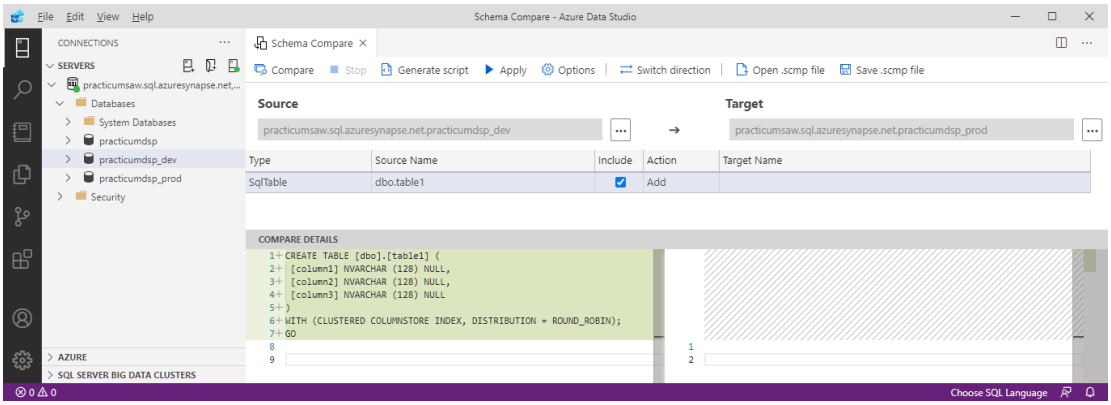
Click the ellipses button.

On the "**Schema Compare**" popout, enter values for the following items:

Source | **Type**	Select the **Database** radio button
Source | **Server** and **Database**	Confirm selection of your Synapse server and the **practicumdsp_dev** database
Target | **Type**	Select the **Database** radio button

Target | **Server** and **Database**	Confirm selection of your Synapse server and the **practicumdsp_prod** database

Click the **OK** button.

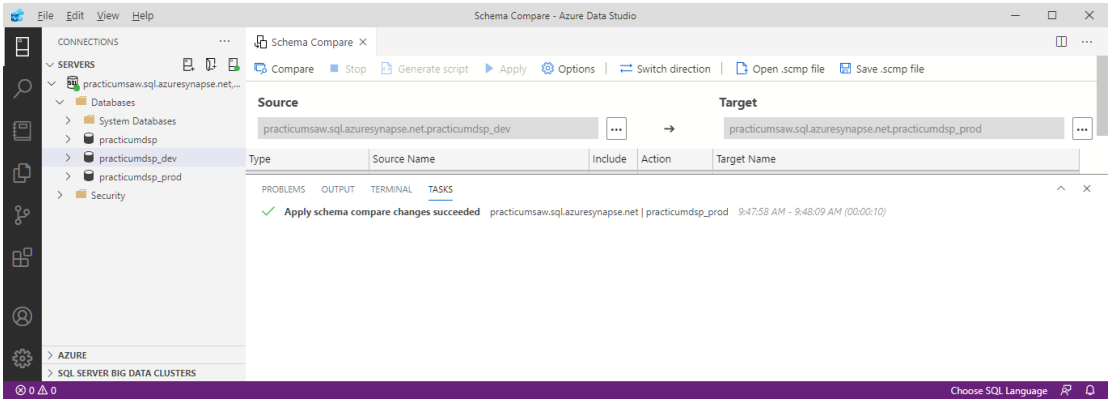


On the "**Schema Compare**" tab, click the **Compare** button.

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.



Objective Complete... congratulations!

Govern

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.

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Purview
- SQL

Step 2: Register Source

Navigate to Purview in the Azure portal.

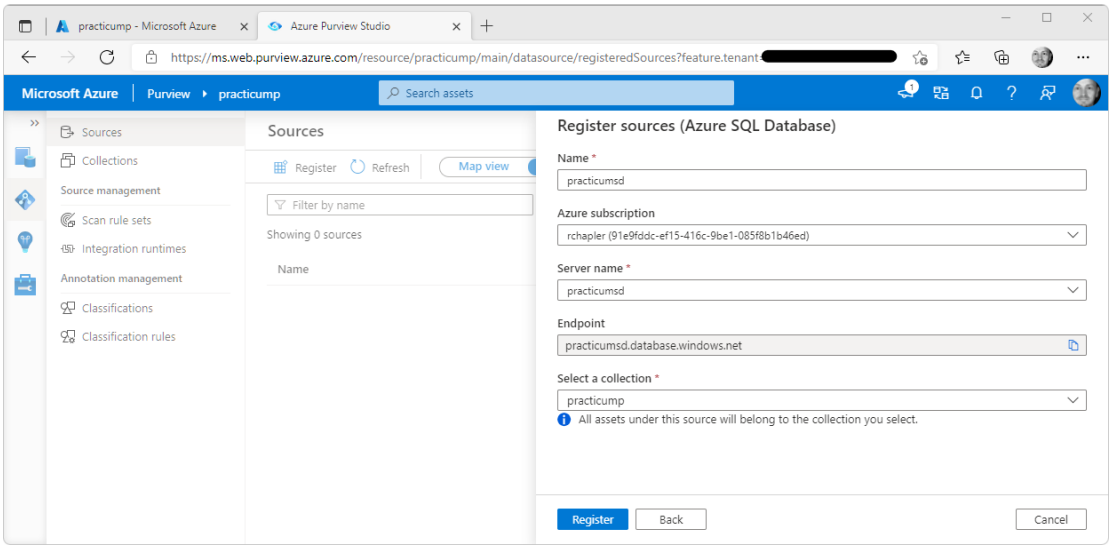
Click the “**Open Purview Studio**” button.

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**” popout.

Click to check the “**Azure SQL Database**” option and then click the **Continue** button.



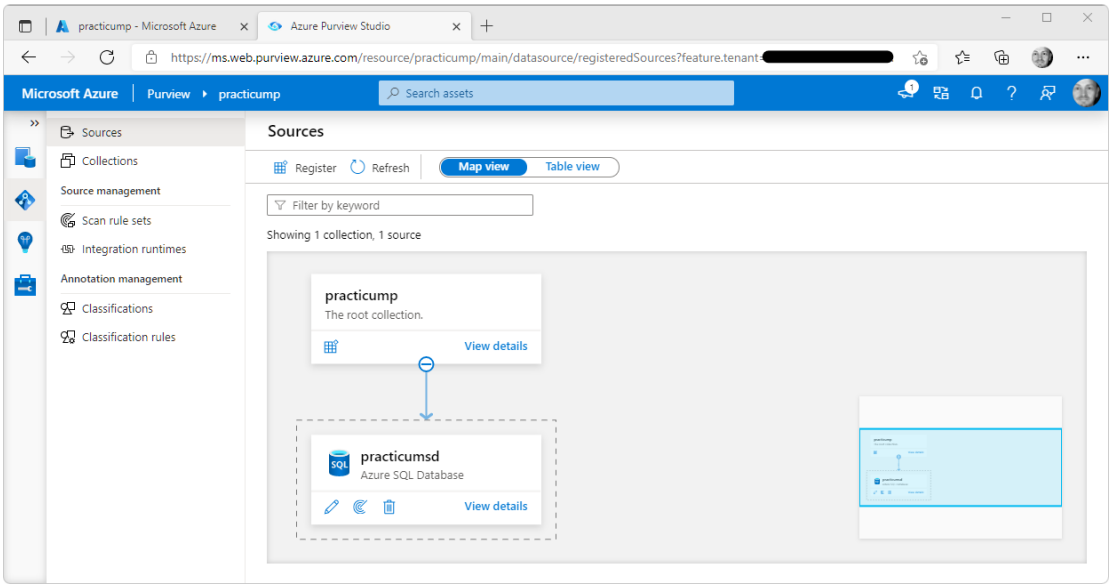
On the "**Register sources (Azure SQL Database)**" popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Subscription	Select your subscription
Server Name	Enter the name of your SQL Database Server
Endpoint	Confirm the populated value
Select a Collection	Leave default value

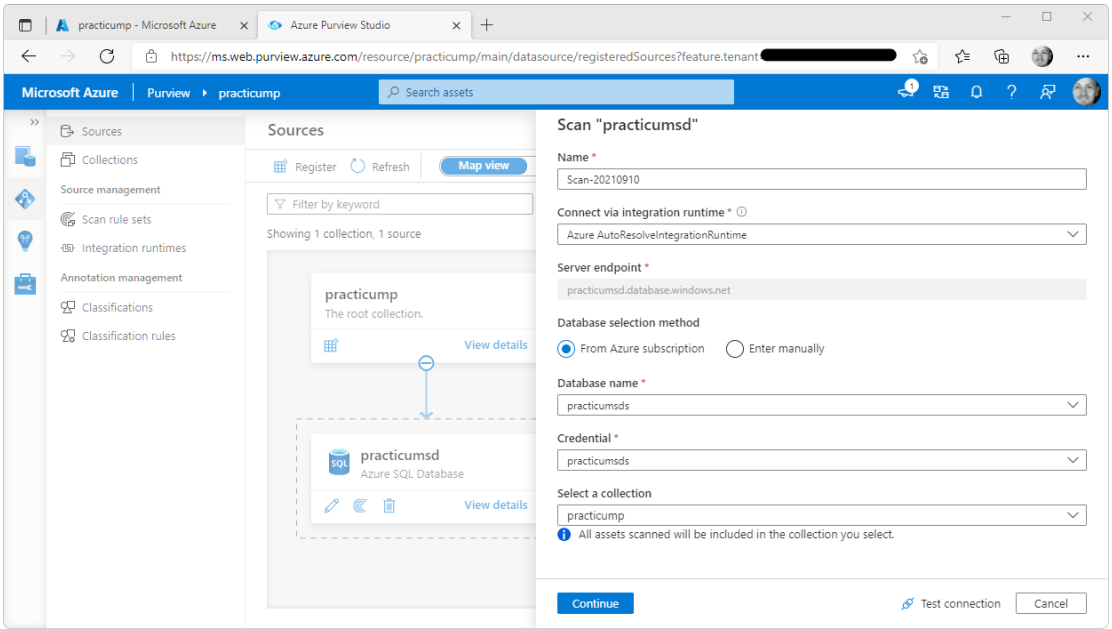
Click the **Register** button.

Step 3: Scan Data

Navigate to **Sources**, **"Map view"**.



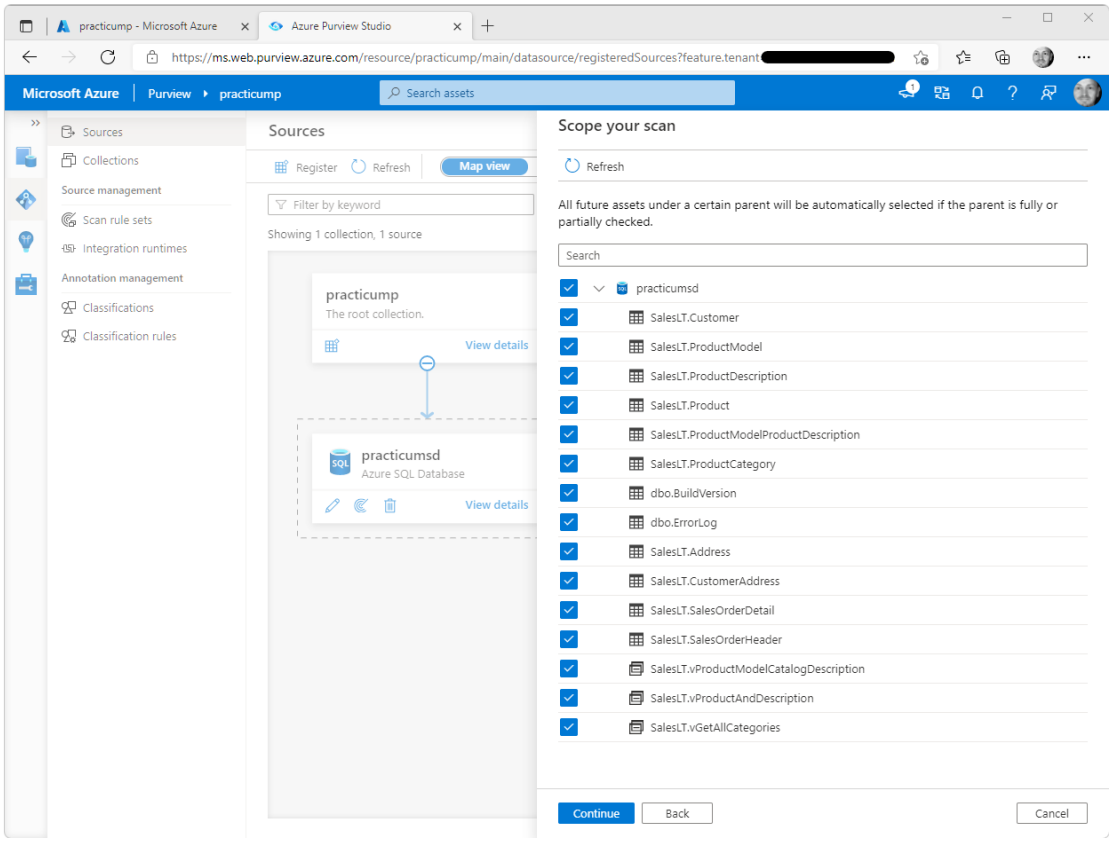
Click the “**New scan**” icon on the newly created item.



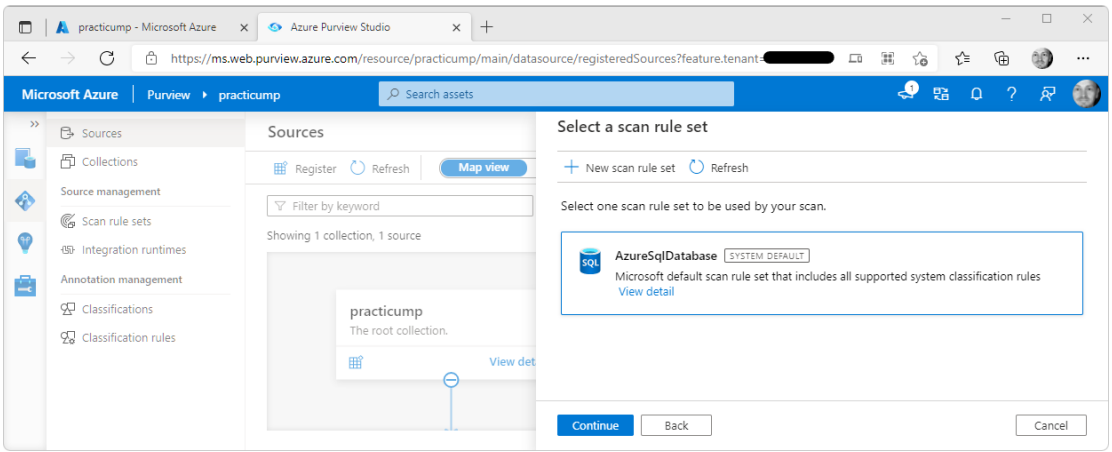
On the “**Scan**…” popout, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Database Selection Method	Select the “**From Azure subscription**” radio button
Database Name	Select your Azure SQL database
Credential	Select your “**SQL Authentication**” credential

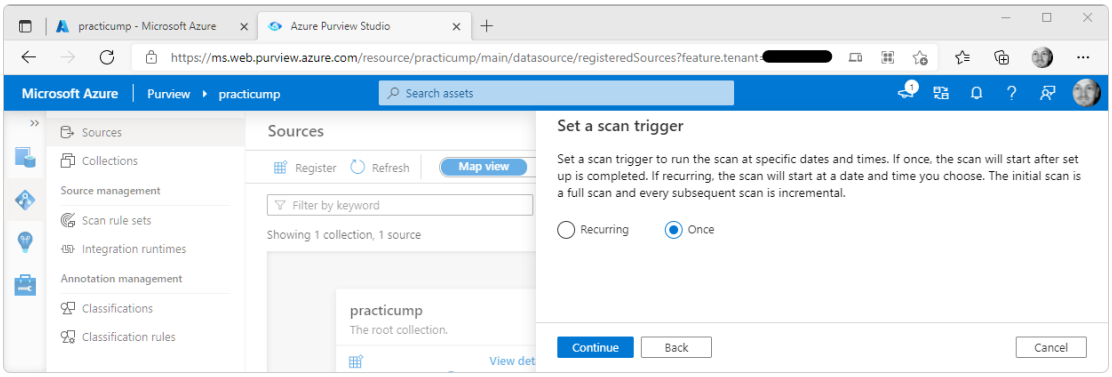
Click “**Test connection**” to confirm successful configuration and then click the **Continue** button.



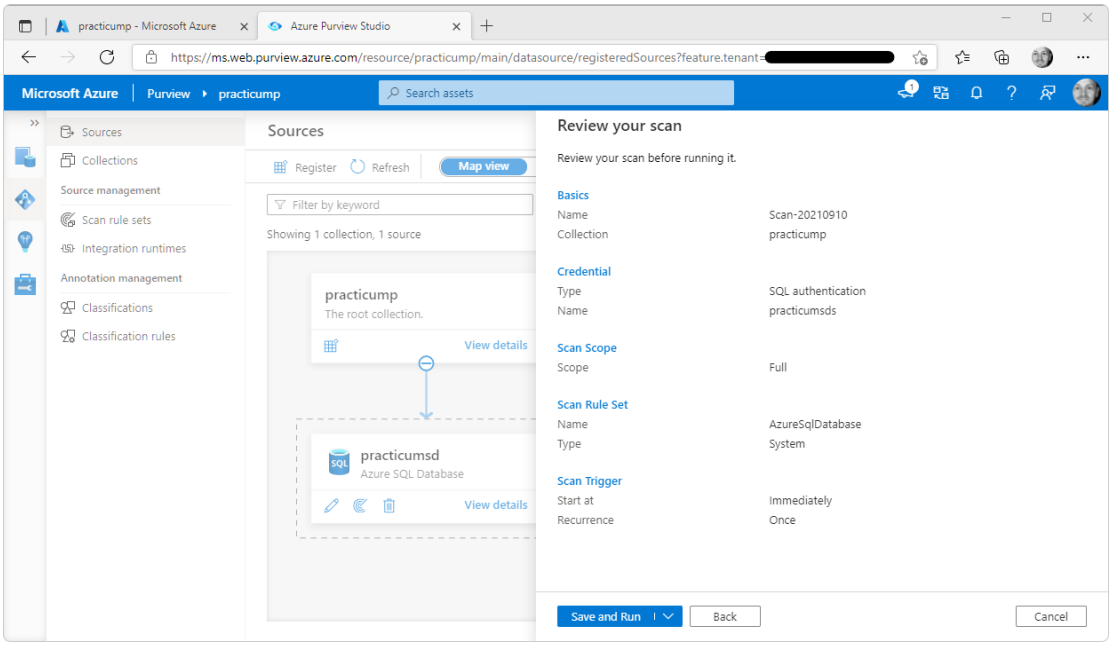
Review and select desired data products from those listed in the “**Scope your scan**” popout and then click the **Continue** button.



Confirm selection of the default Scan Rule Set on the “**Select a scan rule set**” popout and then click the **Continue** button.



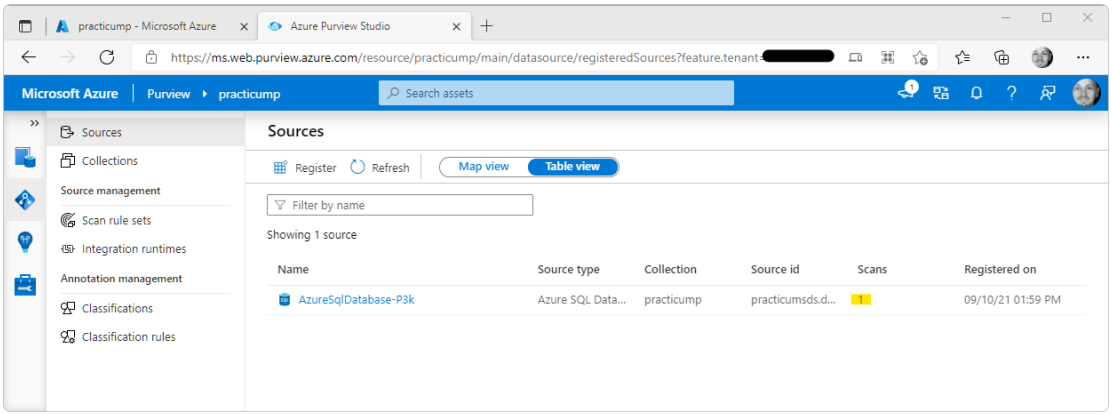
Click the **Once** radio button on the “**Set a scan trigger**” popout and then click the **Continue** button.



Review scan settings and then click the “**Save and Run**” button.

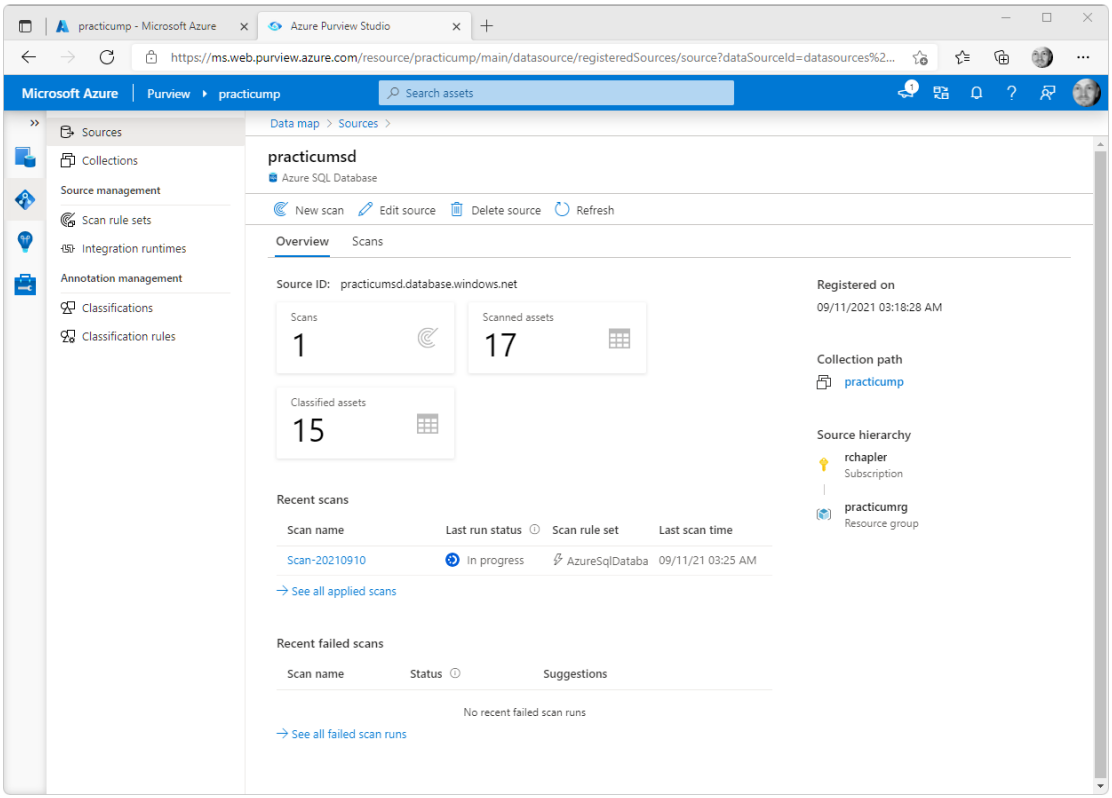
Confirm Scan

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.

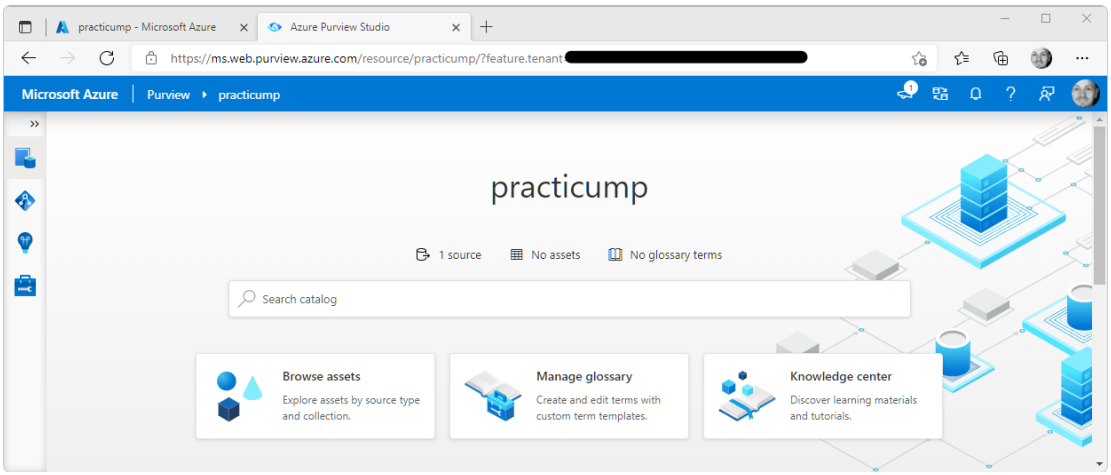


Browse Assets

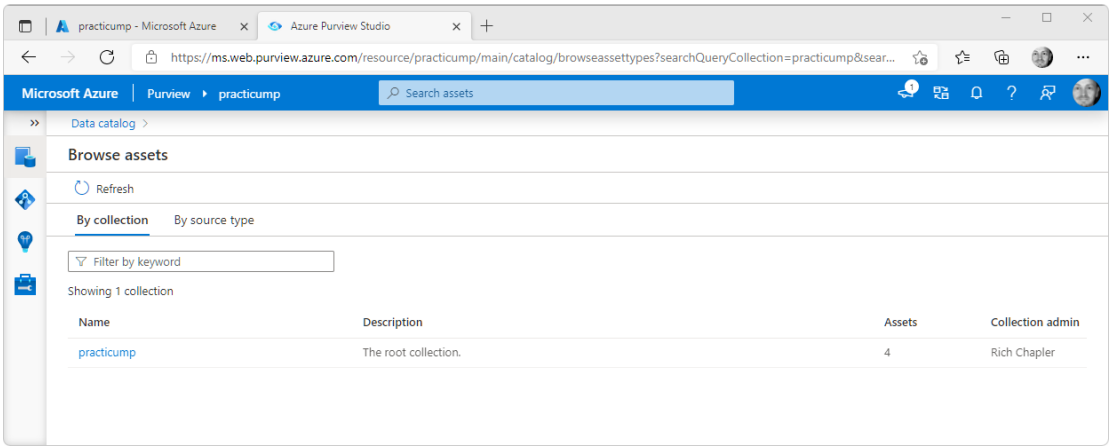
After scanning, it takes time for **Asset Insights** to reflect new data products.

We do not have to wait to browse data products, however.

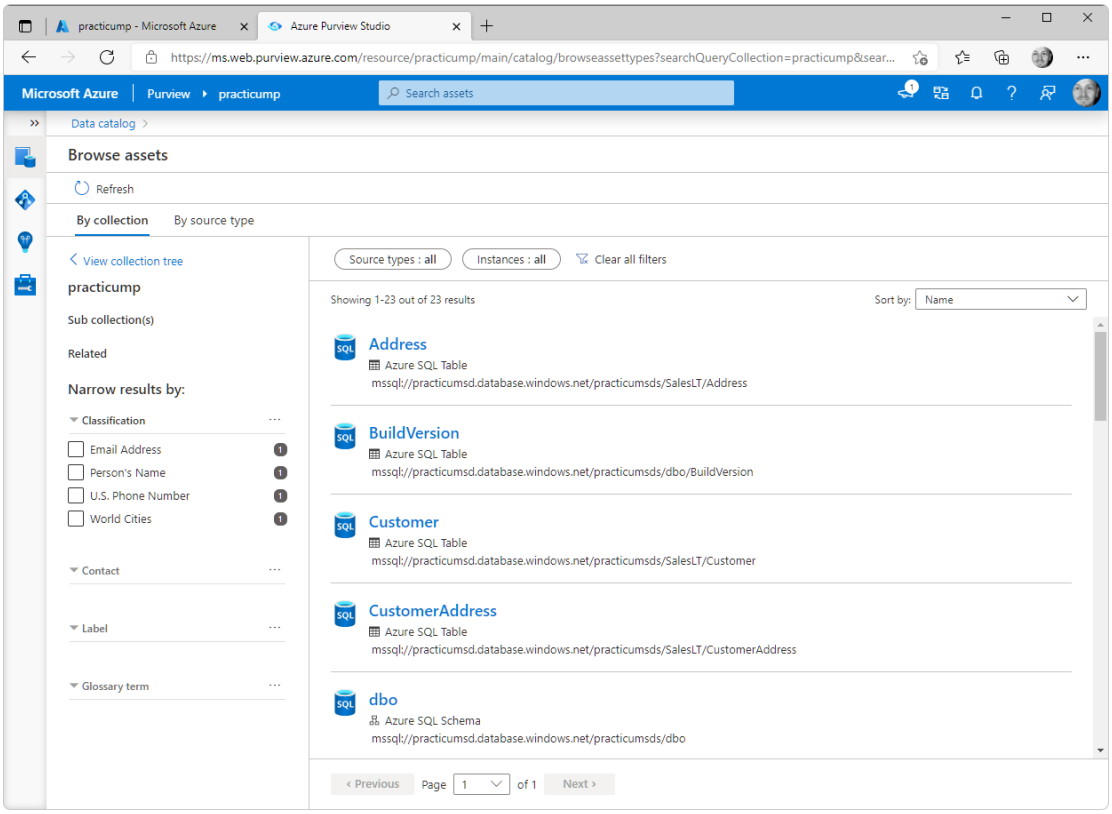
Click the **Data Catalog** icon on the navigation pane.



Click on the "**Browse assets**" button.

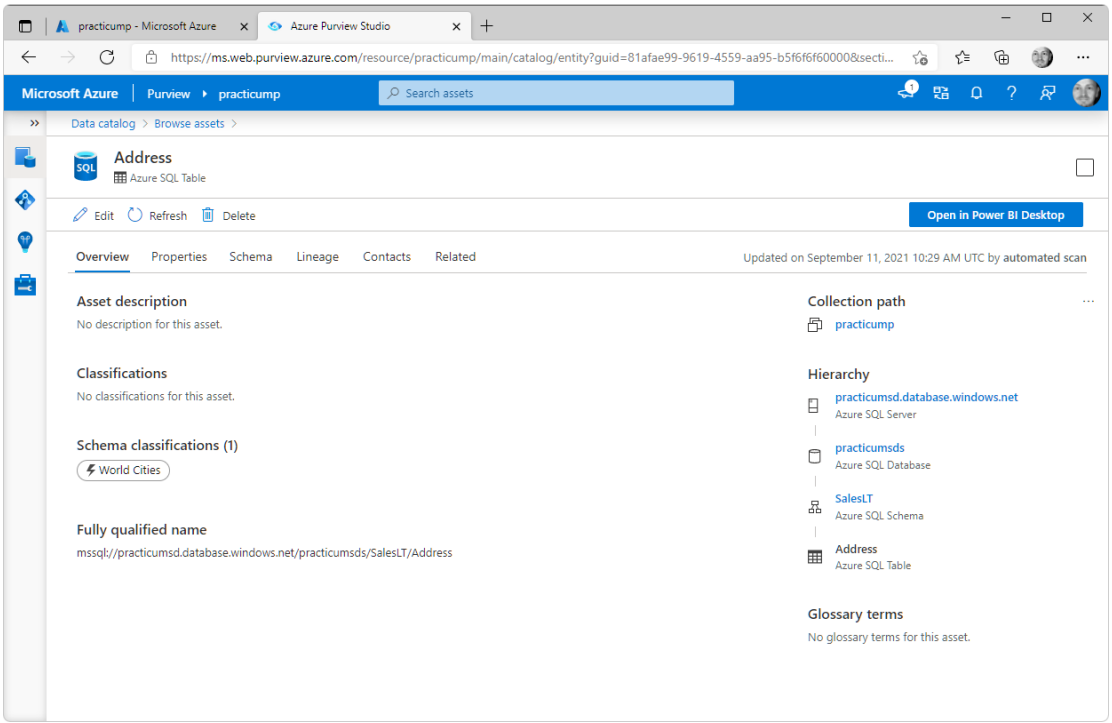


Click on the Root Collection link {i.e., name}.



Explore results.

Click on the first item, **Address**.



Notice the automatic "**World Cities**" classification.

Drill-through the tabs {e.g., Properties, Schema, Lineage, etc.}.

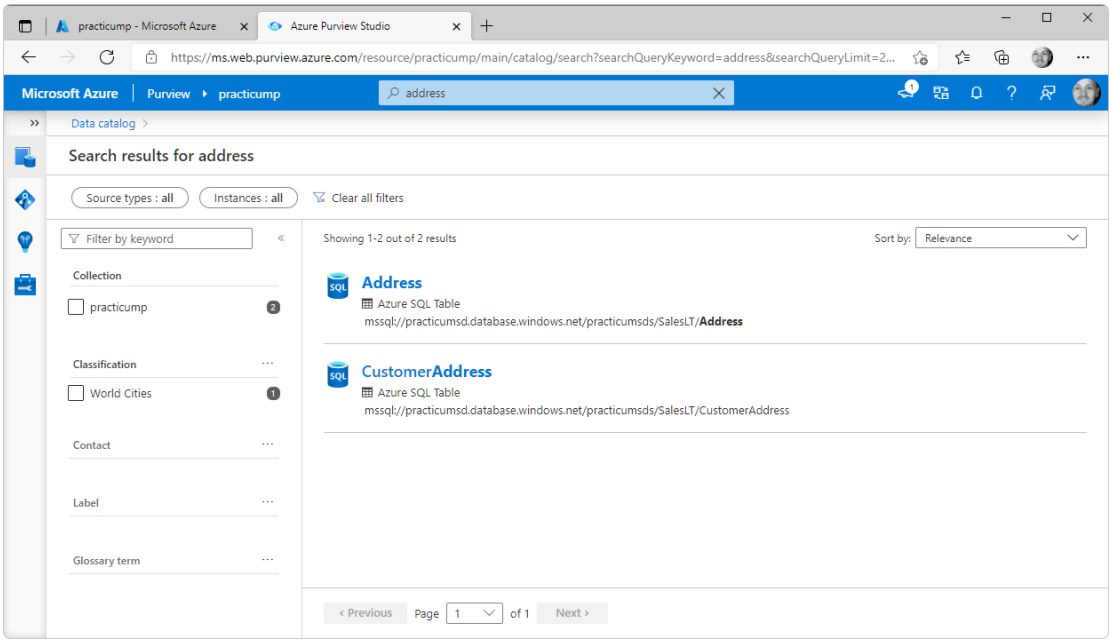
Step 4: Confirm Success

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"}.

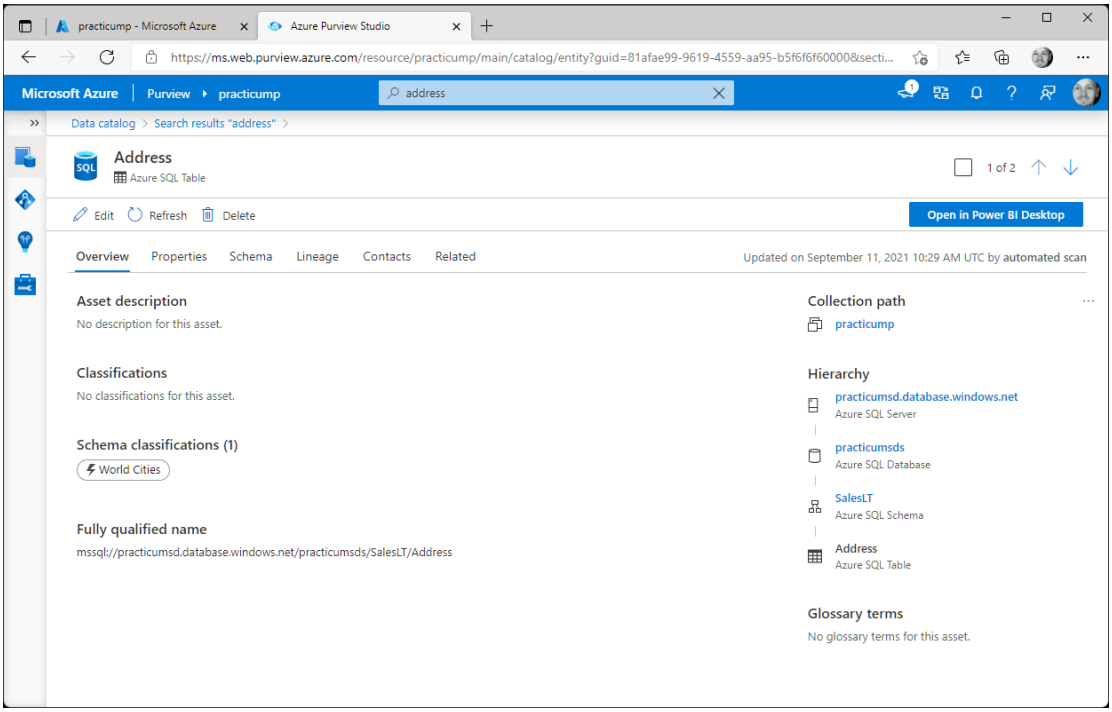


Returned search results will include all data products that include the requested phrase.

Location

Question: “how do I connect to Data Product X?”

Drill into the first search result, Address.

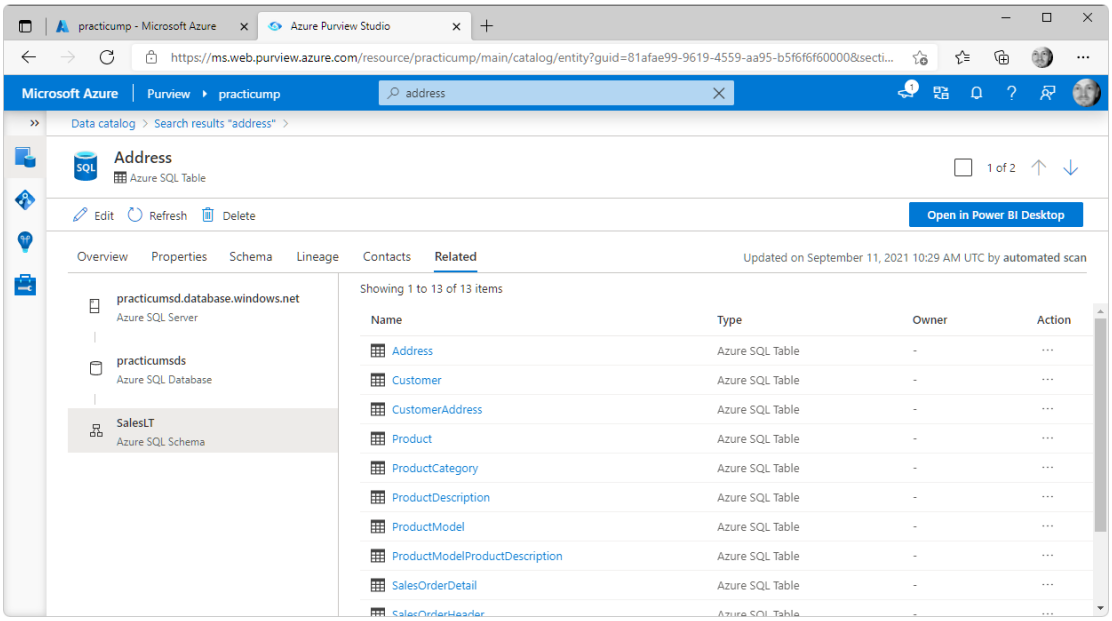


On the **Overview** tab, you will see **Hierarchy** information which includes the Fully Qualified Domain Name of the Azure SQL Server.

Related

Question: “are there related data products that I might find valuable?”

Continue in the search result for phrase “Address”.

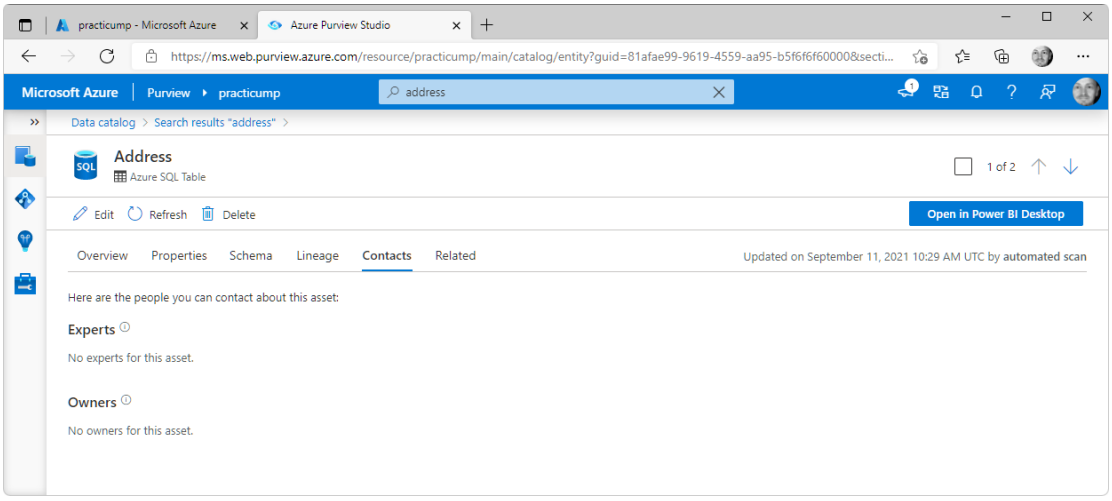


On the **Related** tab, you will see a list of related data products (as identified by Purview).

Contacts

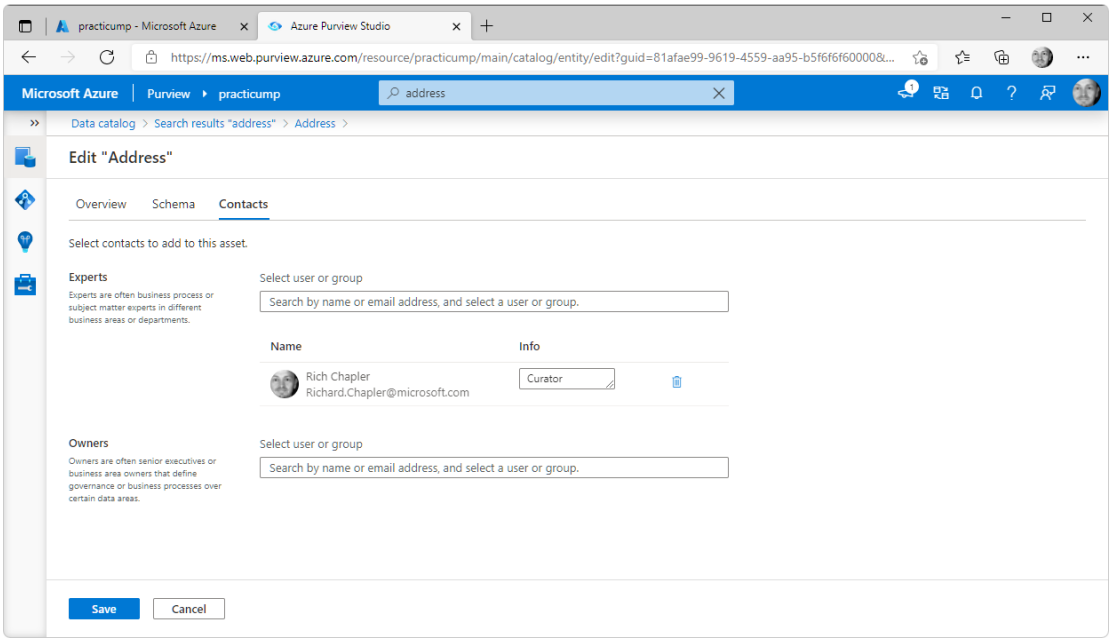
Question: “who are the owners and experts for Data Product X?”

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** dropdown, search for the user or group in your directory. Click **Save**.

Objective Complete... congratulations!

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.

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Purview
- SQL

Step 2: Create Term

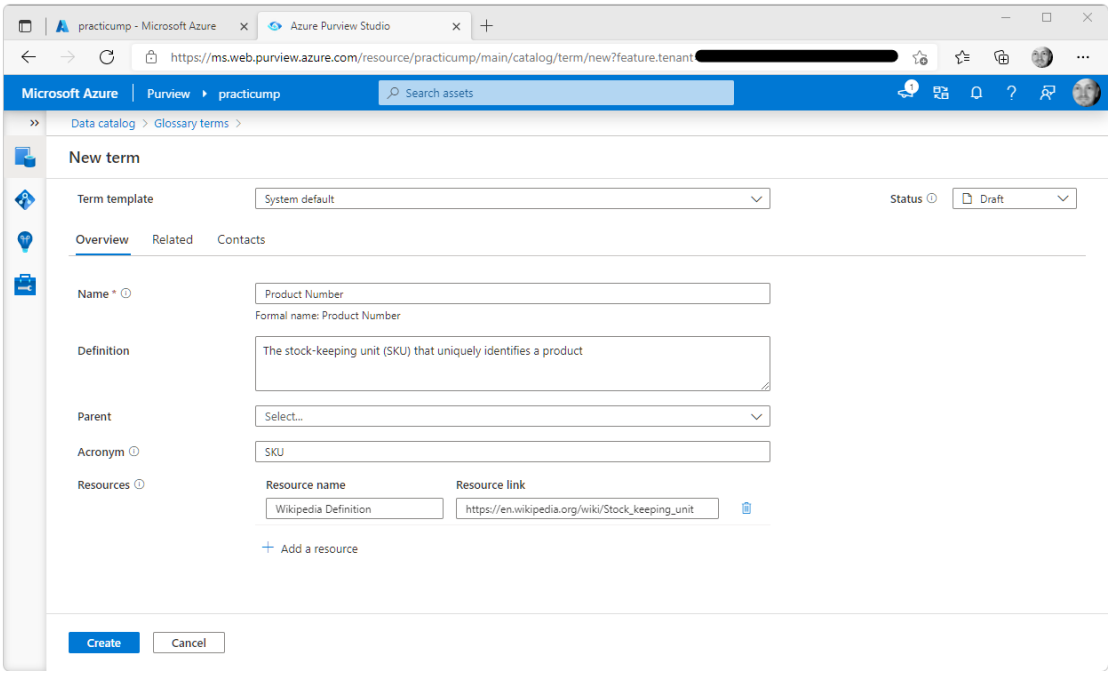
Navigate to Purview in the Azure portal.

Click the "**Open Purview Studio**" button.

Click the "**Manage Glossary**" button.

Click the "**+ New term**" button on the resulting page.

On the resulting popout, confirm selection of "**System default**" and then click the **Continue** button.



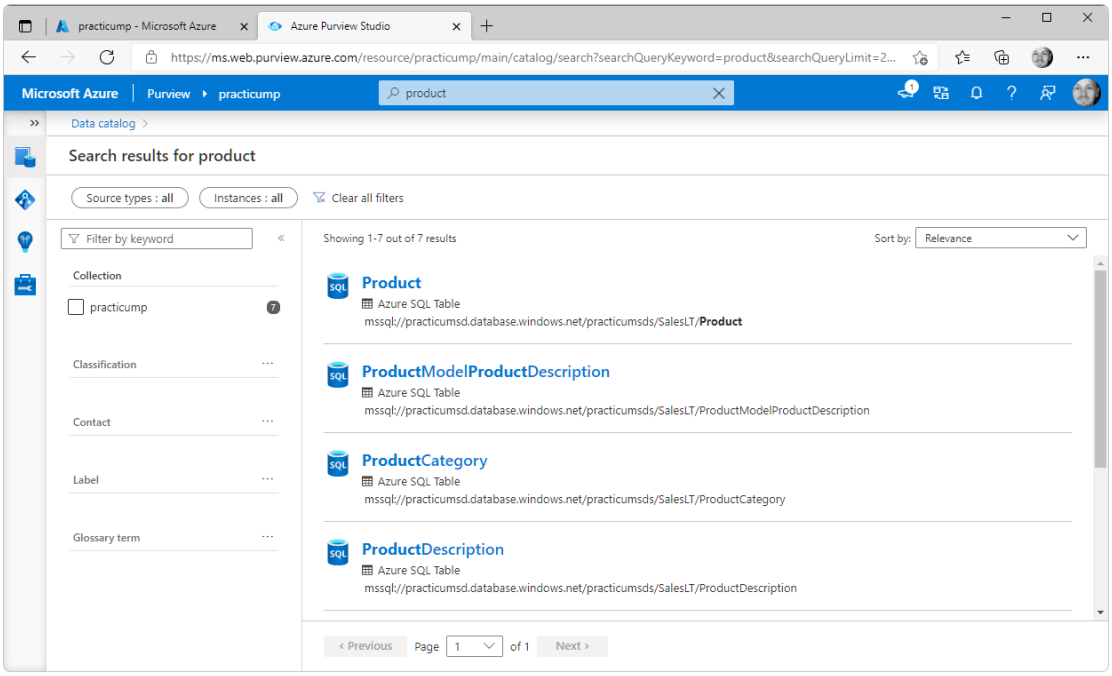
On the "**New term**" page, enter values for the following items:

Name	Enter a meaningful name aligned with standards
Definition	Select your subscription
Parent	Select a parent term, if applicable
Acronym	Capture abbreviation, if applicable
Resources	Click "+ Add a resource" to capture hyperlinks with descriptive content

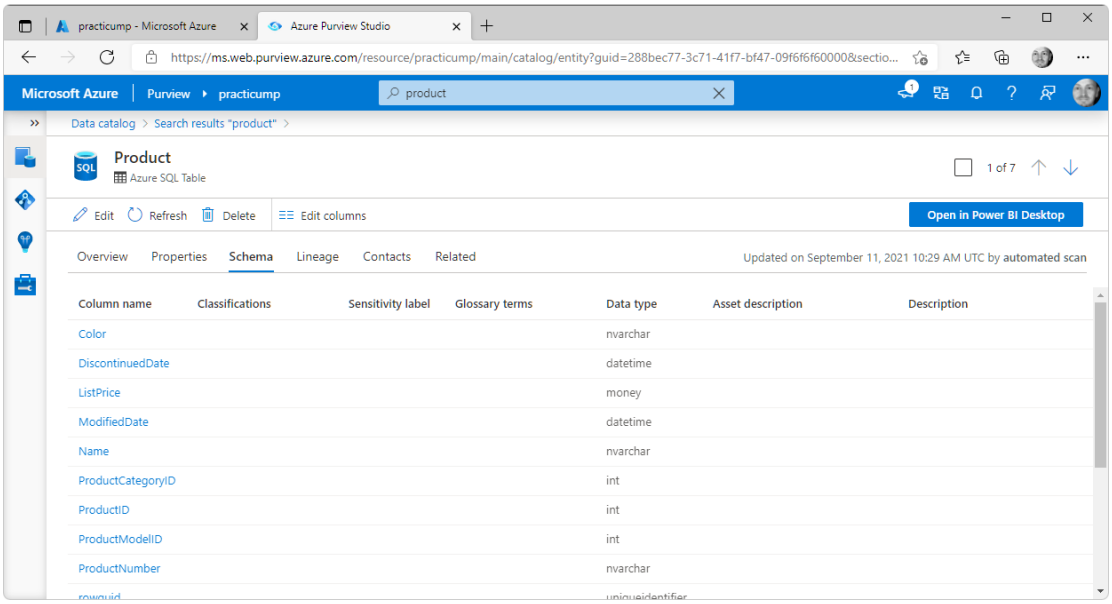
Click the **Create** button.

Step 3: Use Term

Search for "product" to identify data products that use terms like Product Number.

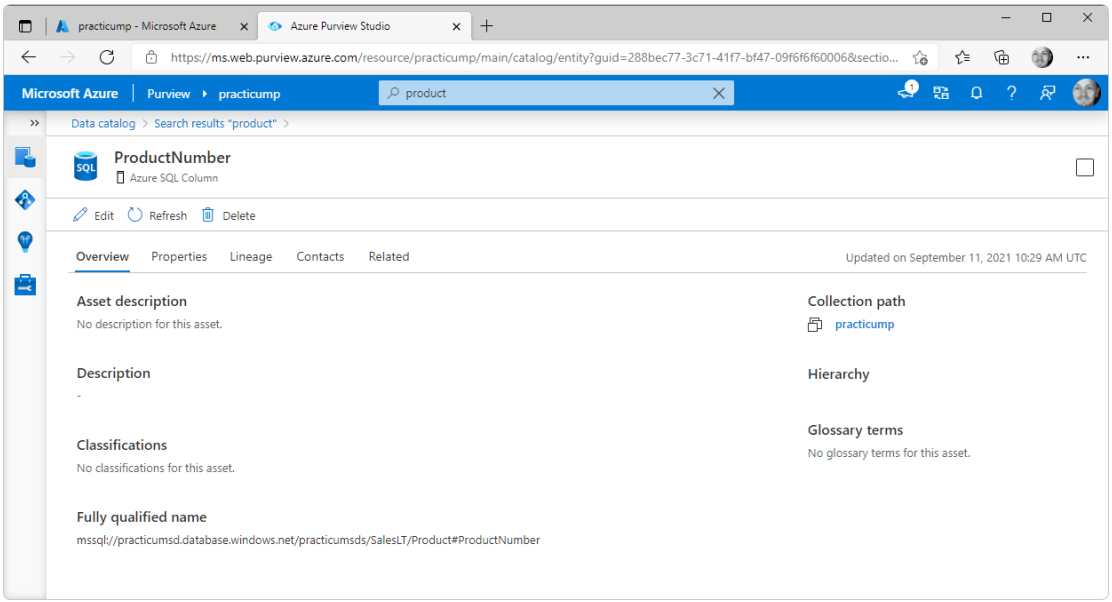


Click **Product**.

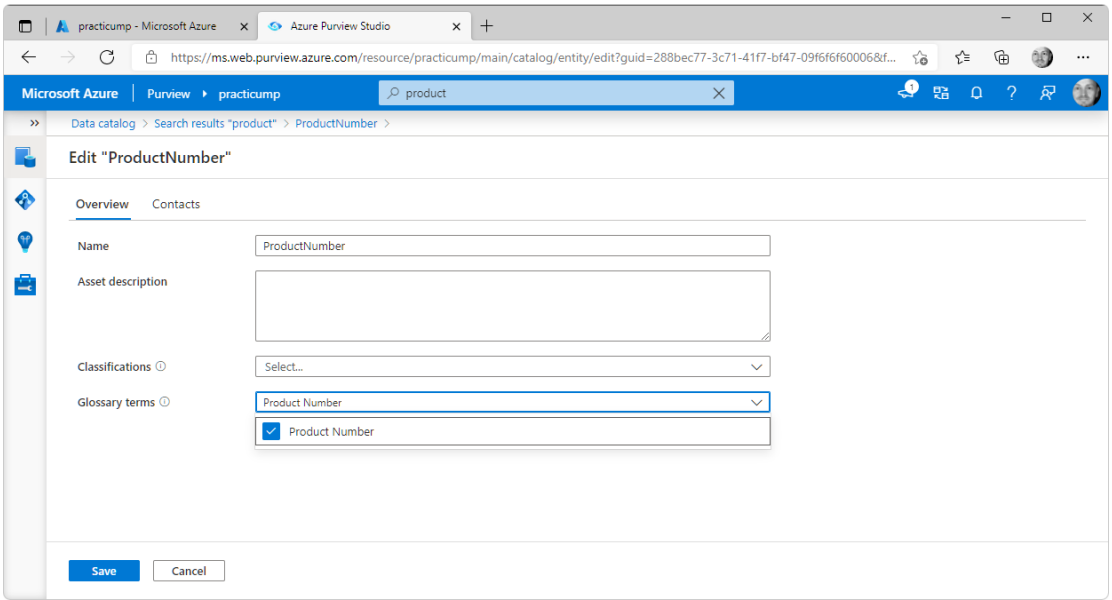


Click the **Schema** tab.

Click on the **ProductNumber** link.



Click the **Edit** button.



Select the new term, "**Product Number**" from the "**Glossary terms**" dropdown.

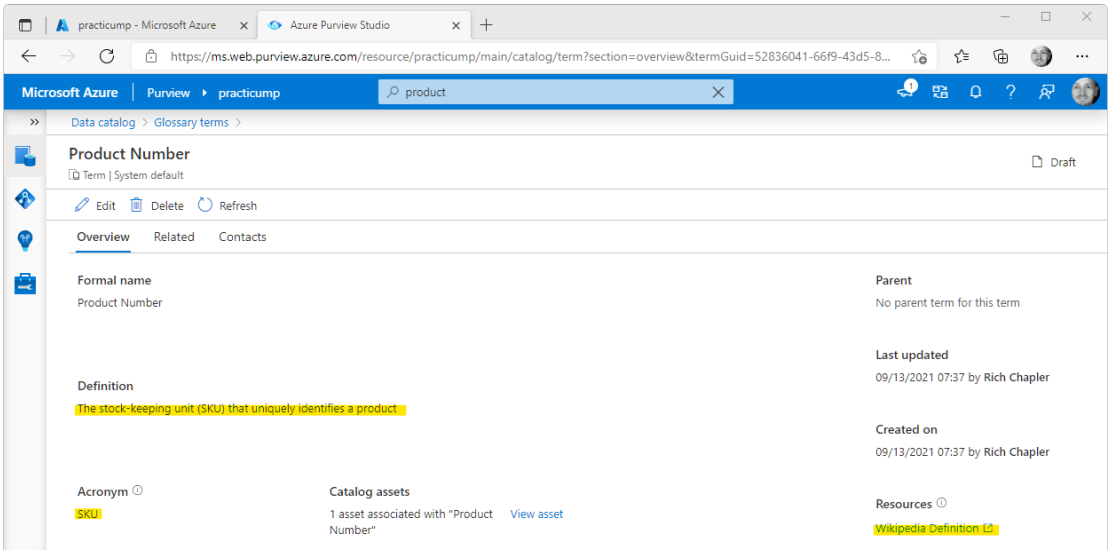
Click the **Save** button.

Step 4: 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?”

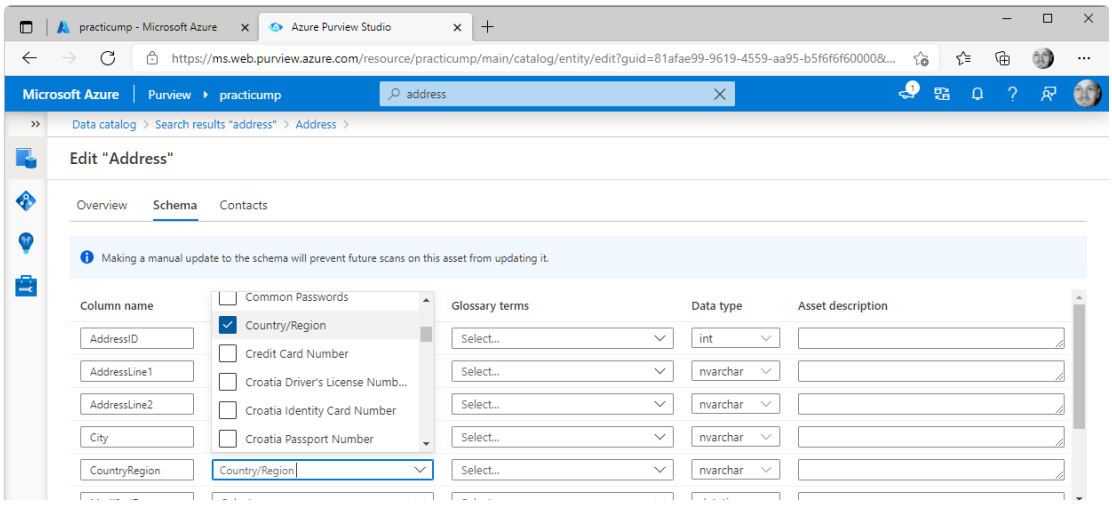


This also demonstrates response to questions:

- **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.

Objective Complete… congratulations!

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.

Step 1: Instantiate Prerequisites

This exercise requires the following resource(s):

- Data Factory
- Data Share
- Purview
- SQL

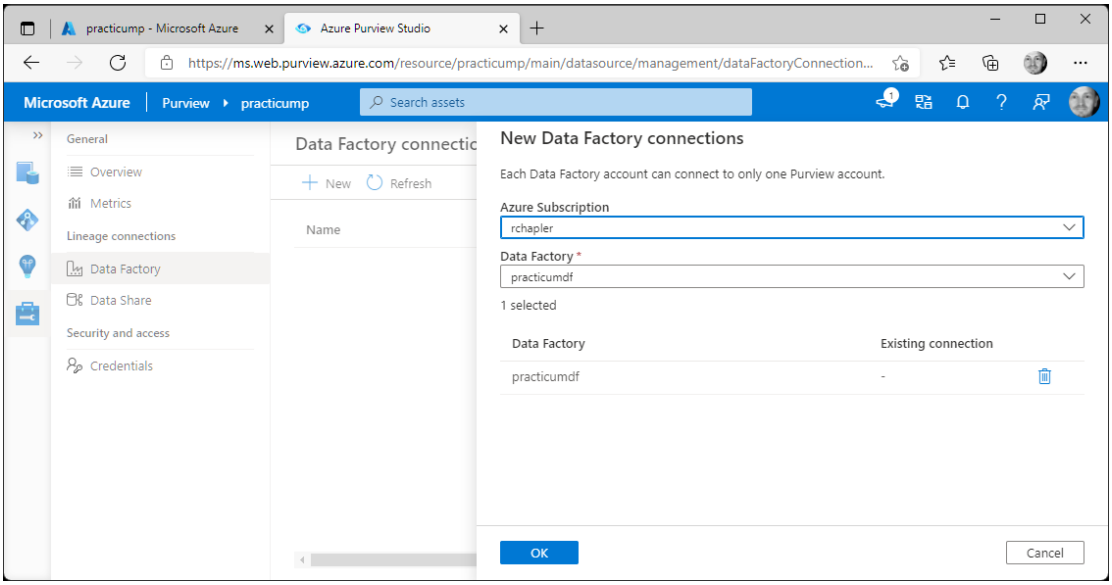
Step 2: Configure Data Factory

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.



On the "**New Data Factory connections**" popout, enter values for the following items:

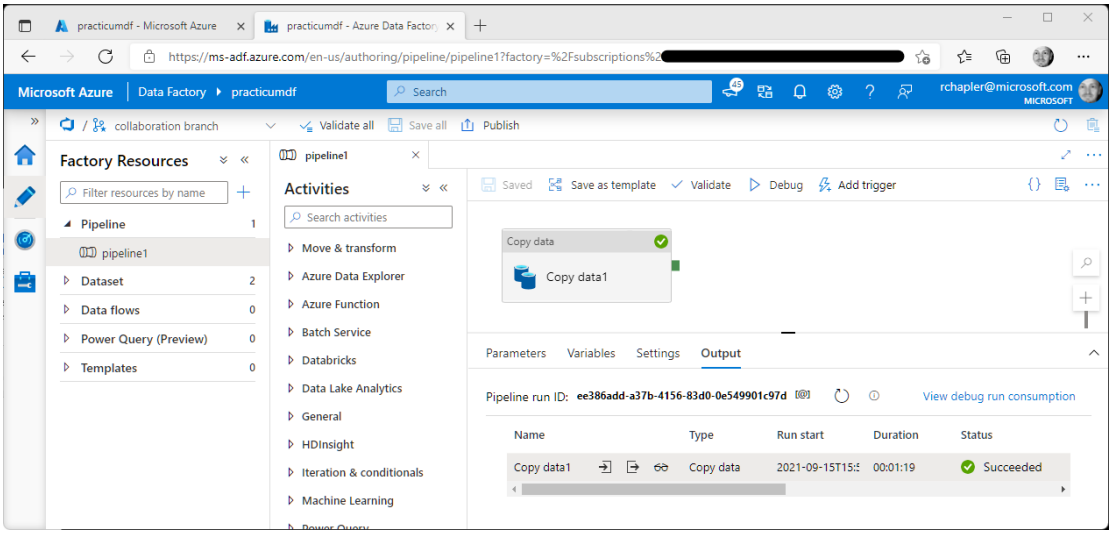
Azure Subscription	Select your subscription
Data Factory	Select data factory

Click the **OK** button. Confirm connection.

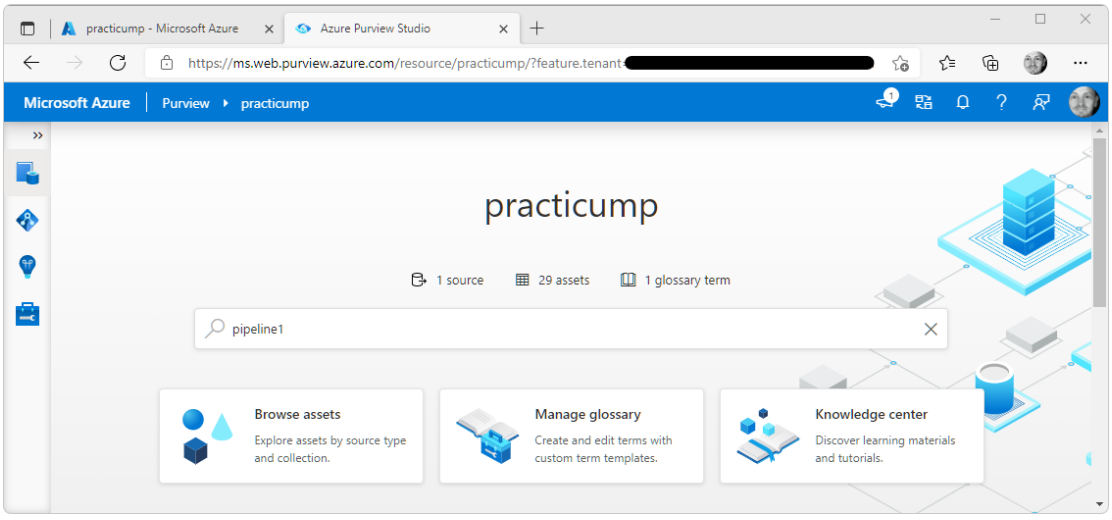
Confirm Success

Purview will automatically detect data movement and capture data about a pipeline when it is executed in a connected data factory.

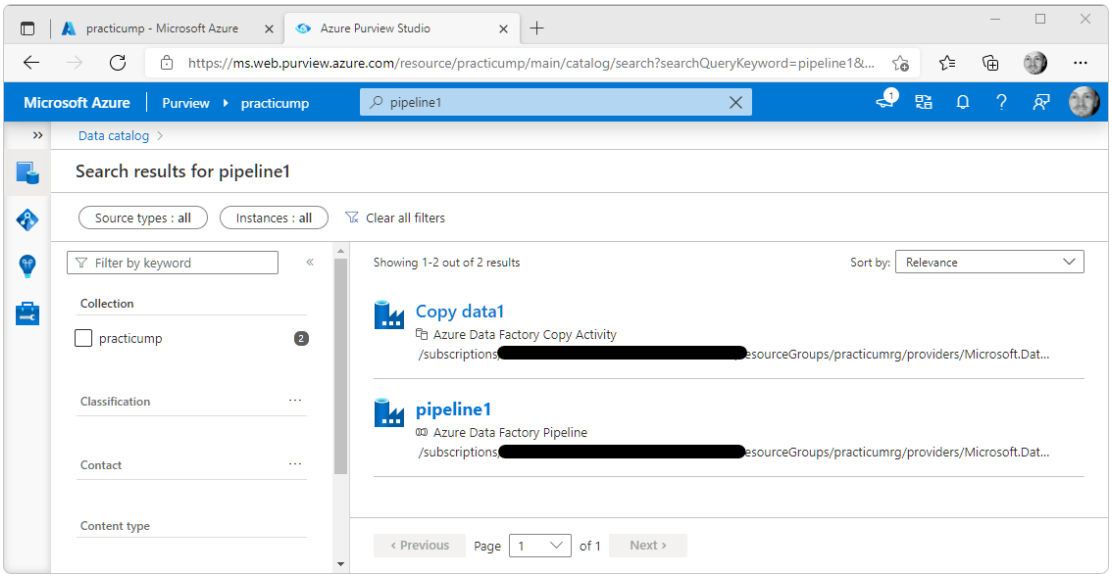
Navigate to Data Factory and execute your sample pipeline.



After successful execution, navigate to Purview.

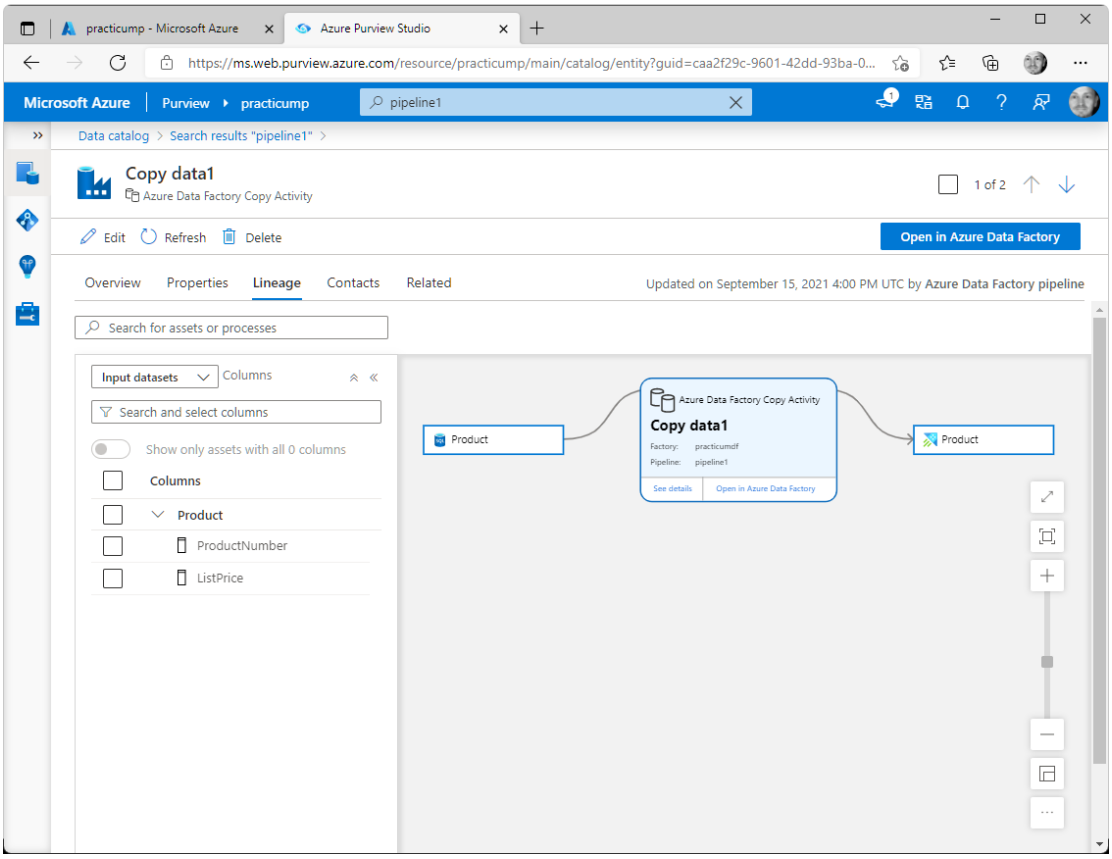


Search for “pipeline1”.



Click on the “**Copy data1**” result.

Click on the **Lineage** tab.



Continue to drill-through elements to better understand the capture of Lineage data.

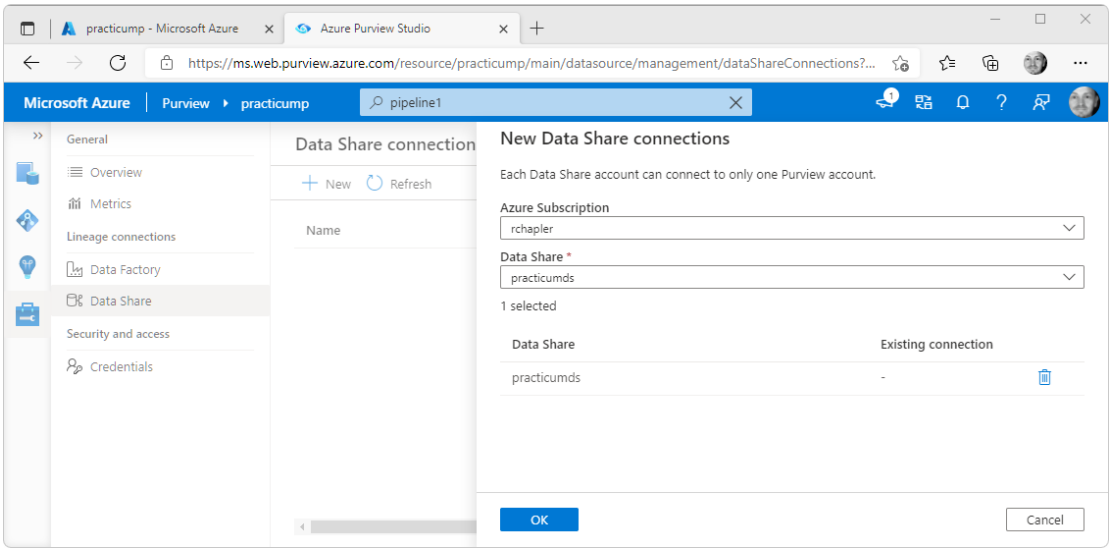
Step 3: Configure Data Share

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.



On the "**New Data Factory connections**" popout, enter values for the following items:

Azure Subscription	Select your subscription
Data Share	Select data share

Click the **OK** button.

Article: [Connect to Azure Data Share - Azure Purview | Microsoft Docs]

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"

Objective Complete… congratulations!

Quick Answers

This section provides quick answers to questions that simply do not merit an extended write-up.

Subscription Identifier

Verbose: “What is my Subscription Identifier?”

Enter the following command in the Cloud Shell:

```
Get-AzADServicePrincipal -DisplayName {Subscription Name}
```

Tenant Identifier

Verbose: “What is my Tenant Identifier?”

Enter the following command in the Cloud Shell:

```
Get-AzADServicePrincipal -DisplayName {Subscription Name}
```