Microsoft Fabric Real Time Demo

Microsoft Fabric Real Time Demo

Table of Contents

Introduction

This repository contains a program that will simulate bank transactions like a checking account with deposits, withdrawals and transaction categories. Thes transactions will simulate real time data you can surface in Microsoft Fabric and use to do real time demos.

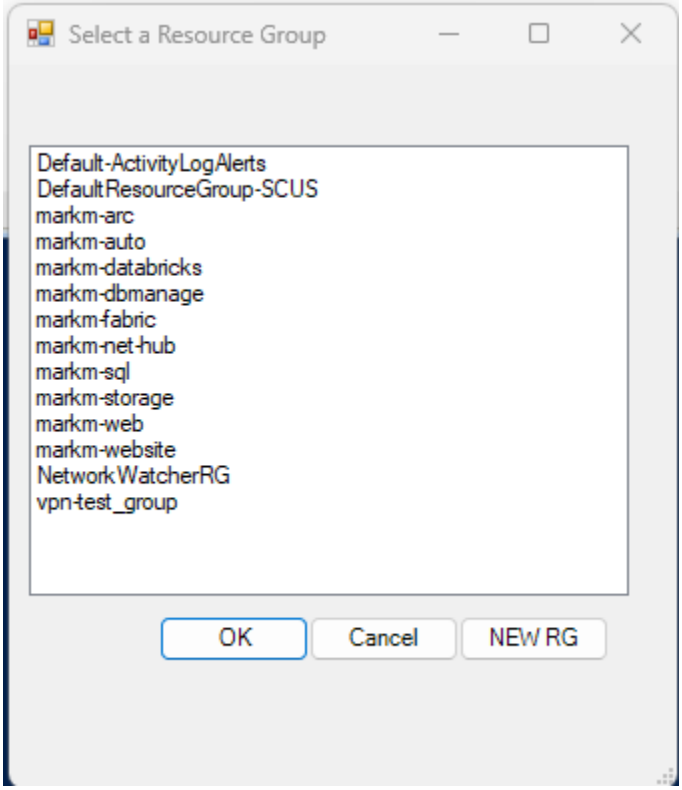
The transactions are sent to an Azure Event Hub that you can use to pull the transactions into Microsoft Fabric.

This repo contains a PowerShell script that will create an event hub and output the information you will need to modify the application code to write to that event hub. There are two versions of the application you can use based on your preferred programming language. One is written in C# .Net the other is written in Python.

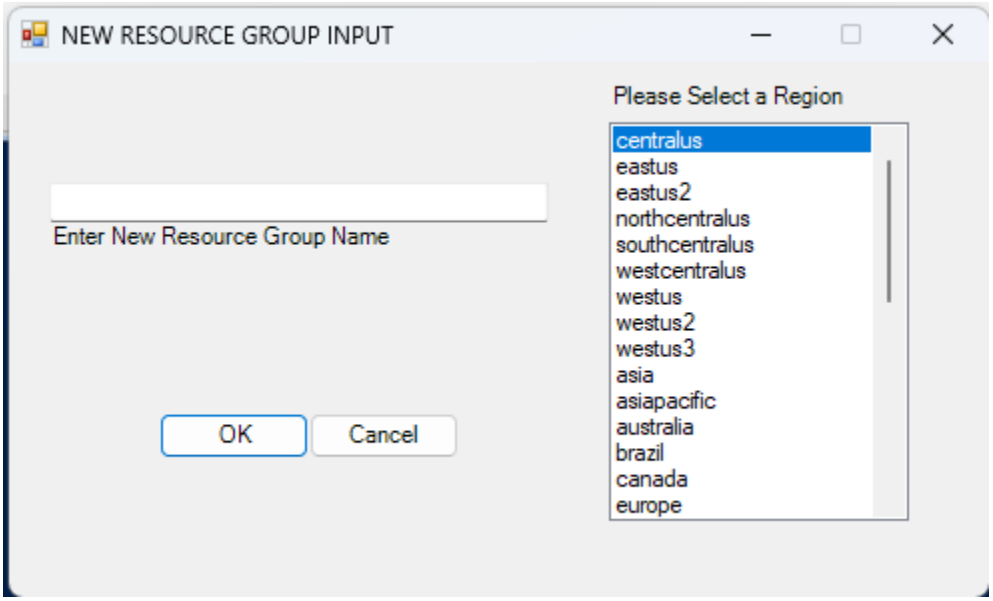
Creating the Event Hub

You will find a PowerShell script called CreateEventHub.ps1 in the root directory of this repository. This script will prompt for user input using Windows Forms. When you run the script it will appear that nothing is happening while the script loads the Windows Forms assembly. Be patient, it shouldn't take more than a minute or two before the first form is displayed. If nothing happens after a few minutes it is possible the windows form window is behind another open windows.

The first window will ask you to select an existing resource group. If you want to create a new resource group click on the NEW RG button.

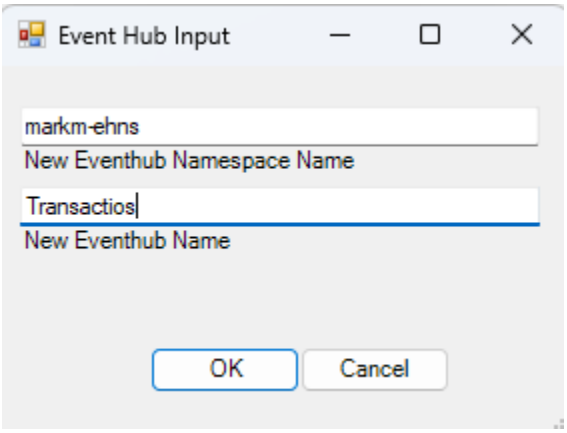


If you want to create a new resource group the following window will appear.



Type the name of the new resource group and select a region where you want it created. In the next step you will be prompted to enter Event Hub information. I will place the new Event Hub in the region where the resource group exists.

In the next screen you will be prompted for an Event Hub Namespace and an Event Hub name.



This is all of the input you will need to enter the Namespace and Event Hub will be created for you and when the script is complete you will see the following output which you can cut and paste into the application. Which output you choose will depend on the version of the application you have chosen to use either C# or Python.

```
Copy the following lines of code and replace Lines 21 through 24 in the C# program with them
-----------------------------------------------------------------------
--                              C#                                   --
-----------------------------------------------------------------------

private static readonly string EHNamespace = "markm-FabricRTEH";
private static readonly string EHName = "Transactions";
private static readonly string EHKeyname = "RootManageSharedAccessKey";
private static readonly string EHKey = "xG16gV[redacted]=";


-----------------------------------------------------------------------
--                             Python                                --
-----------------------------------------------------------------------

EHNamespace = "markm-FabricRTEH"
EHName = "Transactions"
EHKeyname = "RootManageSharedAccessKey"
EHKey = "xG16[redacted]="

PS C:\Users\markm>
```

If you are familiar with using Event Hubs and would like to set up an Event Hub in an existing namespace you can do that, you will need to create the event hub in the portal and provide the values for the namespace, event hub name, key name and key value in the application in the above format.

Getting the program ready to run

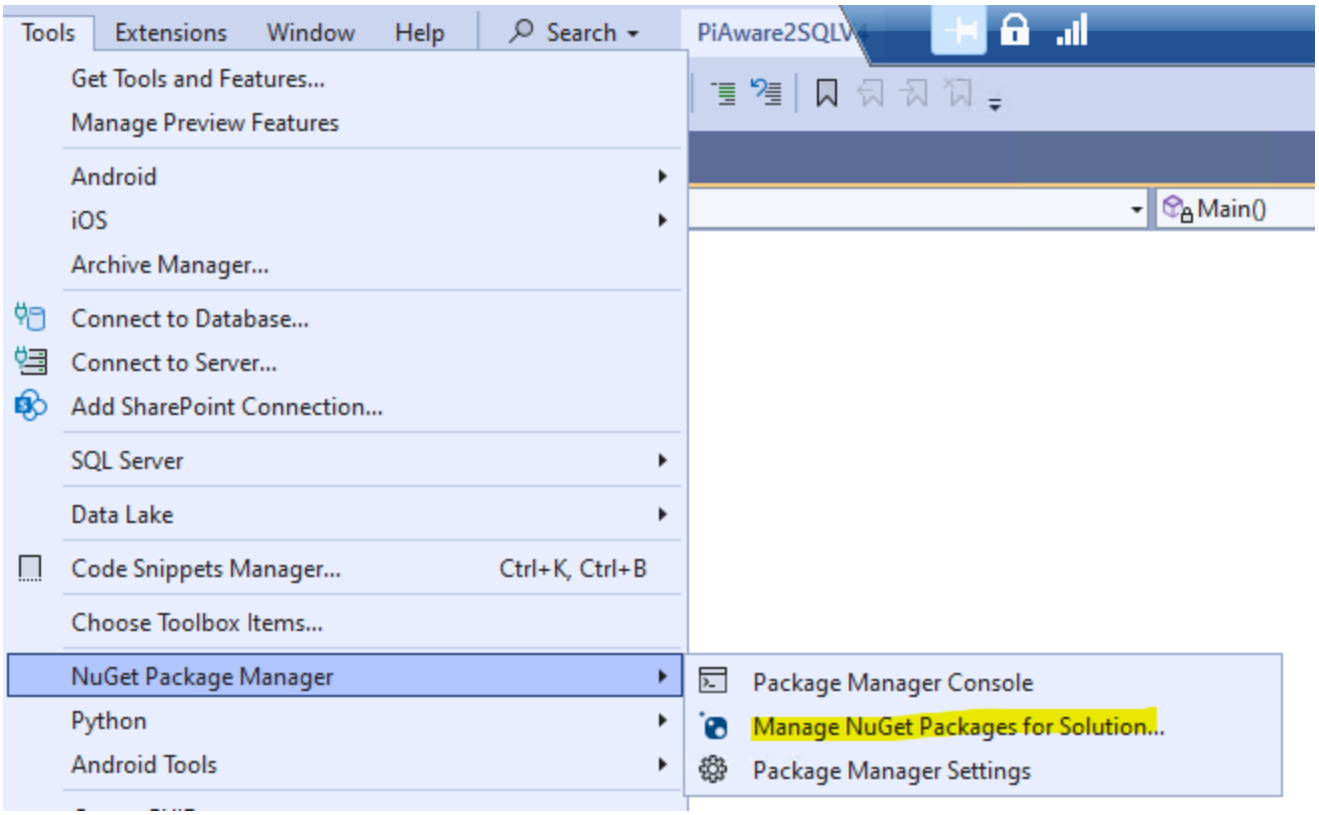
C#

```
1   /*
2    * This is a sample program that will randomly generate transactions as you would see in a checkbook alo
3    * If the balance falls below the next transaction amount + $500, it will make a random deposit amount t
4    *
5    * In this specific example, I am writing the transactions to an Azure Event Hub, however, it would be v
6    *
7    * 08/12/2024
8    * Mark Moore
9    */
10
11   using System;
12   using System.Threading.Tasks;
13   using Azure.Messaging.EventHubs;
14   using Azure.Messaging.EventHubs.Producer;
15   using Newtonsoft.Json.Linq;
16
     0 references
17   public class Program
18   {
19       // Modify the next four lines of code to include values for your Eventhub Namespace, Name, Keyname,
20       private static readonly string EHNamespace = "markm-eh";
21       private static readonly string EHName = "transactions";
22       private static readonly string EHKeyname = "RootManageSharedAccessKey";
23       private static readonly string EHKey = "tMG2u4sBAz+0q87JNMfCRz94+beos15E3+AEhApoAoY=";
```

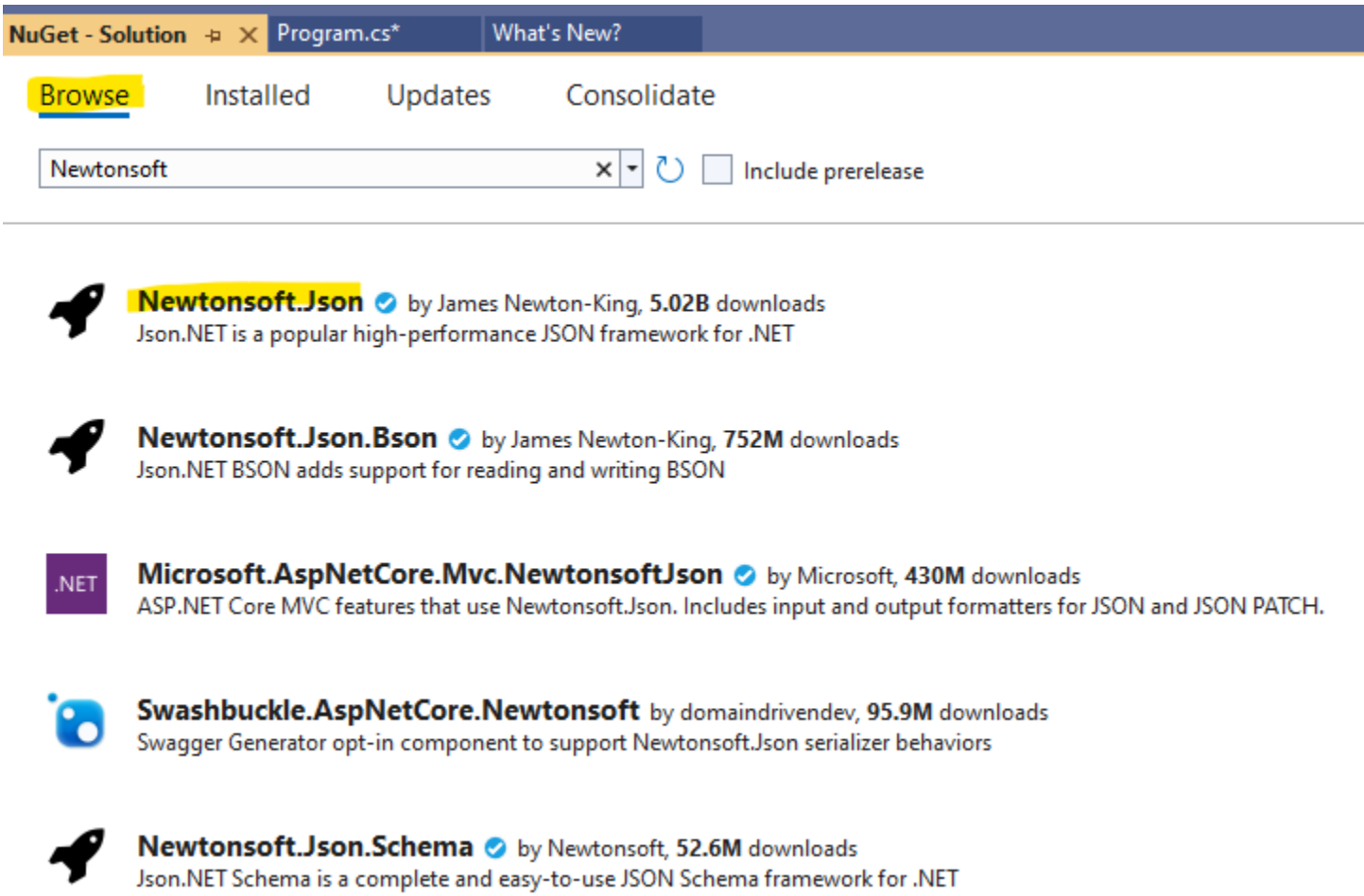
You only need to perform a few steps to get this program to run.

Notice on lines 13, 14 and 15 you have red lines under components in the using statements. This means that those components are referenced in the program but are not available to use because they have not been loaded into your solution.

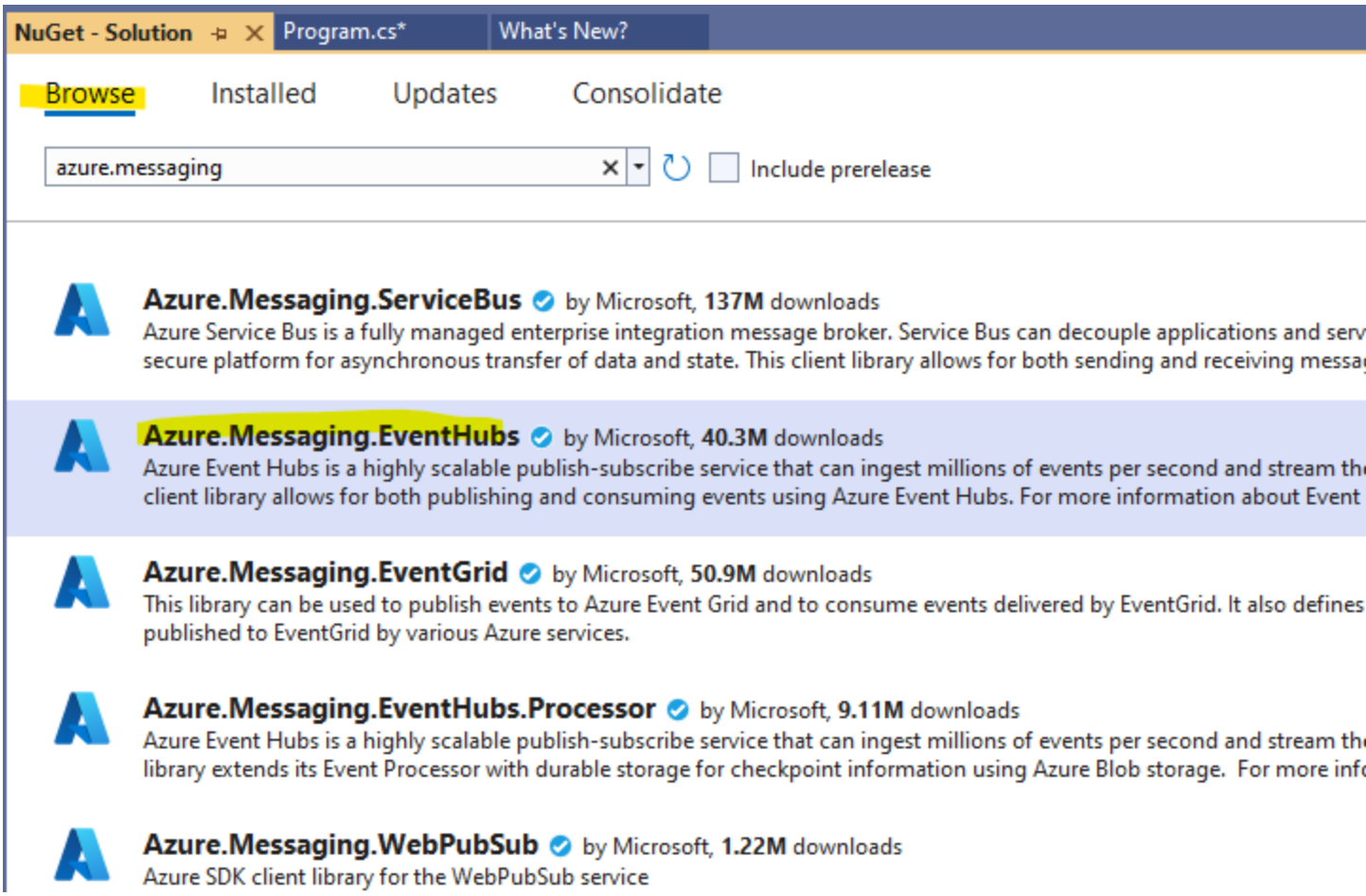
To load these components, go to the Tools menu in Visual Studio and select NuGet Package Manager and then select Manage NuGet Packages for Solution.



Go to the Browse tab and then search for Newtonsoft select Newtonsoft.Json put a check mark next to your Project name if you have named it something other than Transactions2EH. Click install and apply.



Repeat the process for Azure.Messaging.EventHubs



Next you will need to modify lines 21 through 24 with the values used when creating your Event Hub. I provide these values in the Power Shell script to create the event hub as complete lines of code, so you can copy those lines from the output of the script and replace lines 21 through 24 with those lines.

```
0 references
public class Program
{
    // Modify the next four lines of code to include values for your Eventhub Namespace, Name, Keyname,
    private static readonly string EHNamespace = "YourEHNamespace";
    private static readonly string EHName = "YourEHName";
    private static readonly string EHKeyname = "YourKeyName";
    private static readonly string EHKey = "YourKeyValue";
```

Run the program and your output should look like this:


```
    "Deposit": "",
    "Withdrawal": "494.60",
    "Balance": "1667.96"
}
{
    "DateTime": "2024-08-19T11:54:48.026540",
    "Category": "Insurance",
    "Deposit": "",
    "Withdrawal": "396.18",
    "Balance": "1271.78"
}
{
    "DateTime": "2024-08-19T11:54:49.061847",
    "Category": "Insurance",
    "Deposit": "",
    "Withdrawal": "200.30",
    "Balance": "1071.48"
}
{
    "DateTime": "2024-08-19T11:54:50.093212",
    "Category": "Deposit",
    "Deposit": 6935.13,
    "Withdrawal": "",
    "Balance": "8006.61"
}
```

Python

You will need to load the event hub library into your Development Environment using the following command:

pip install azure-eventhub

Next you will need to replace the following lines of code with those generated by the PowerShell script

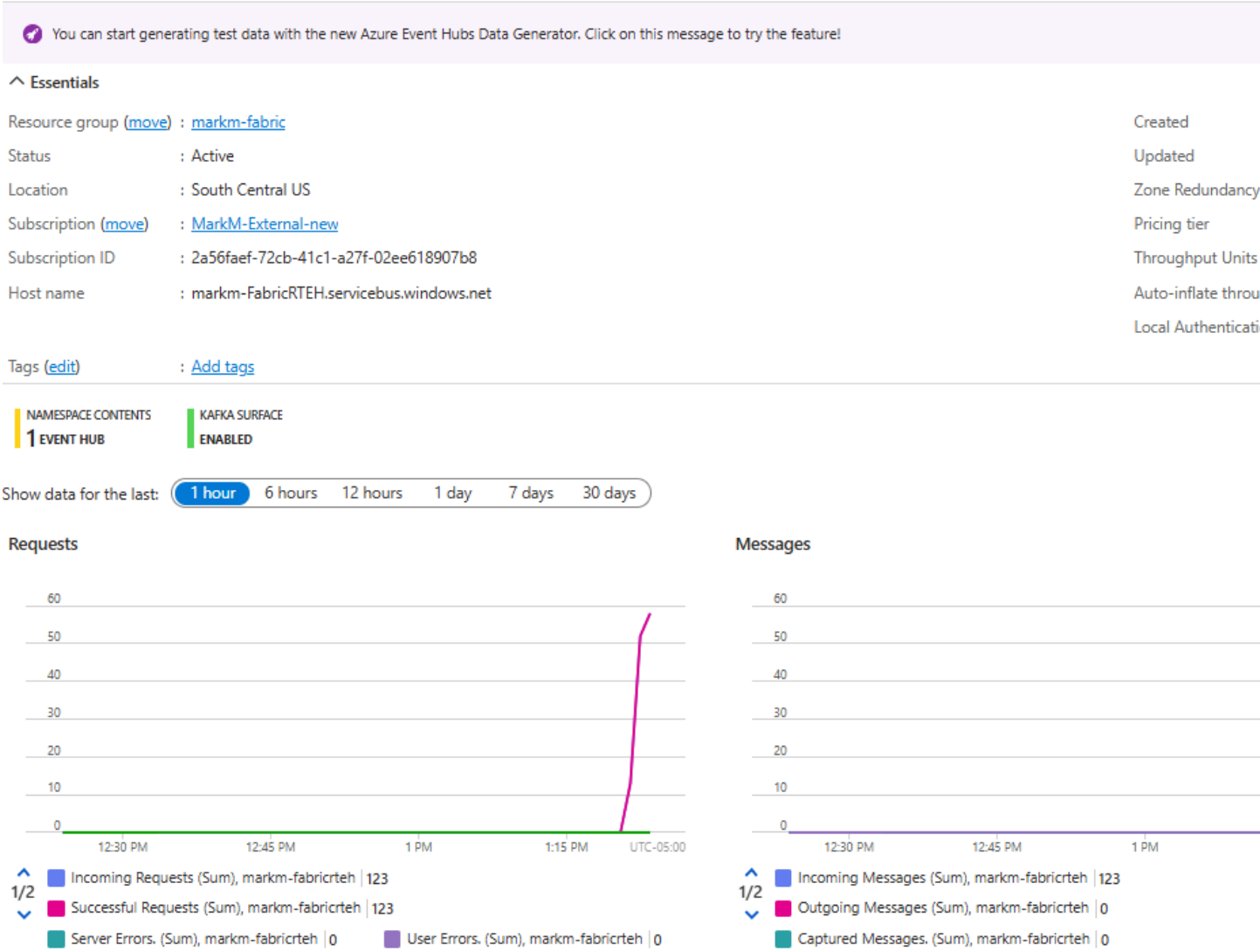
```
EHNamespace = "YourEHNamespace"
EHName = "YourEHName"
EHKeyname = "YourKeyName"
EHKey = "YourKeyValue"
```

Run your program, the output should look like this:


```
    "Deposit": "",
    "Withdrawal": "494.60",
    "Balance": "1667.96"
}
{
    "DateTime": "2024-08-19T11:54:48.026540",
    "Category": "Insurance",
    "Deposit": "",
    "Withdrawal": "396.18",
    "Balance": "1271.78"
}
{
    "DateTime": "2024-08-19T11:54:49.061847",
    "Category": "Insurance",
    "Deposit": "",
    "Withdrawal": "200.30",
    "Balance": "1071.48"
}
{
    "DateTime": "2024-08-19T11:54:50.093212",
    "Category": "Deposit",
    "Deposit": 6935.13,
    "Withdrawal": "",
    "Balance": "8006.61"
}
```

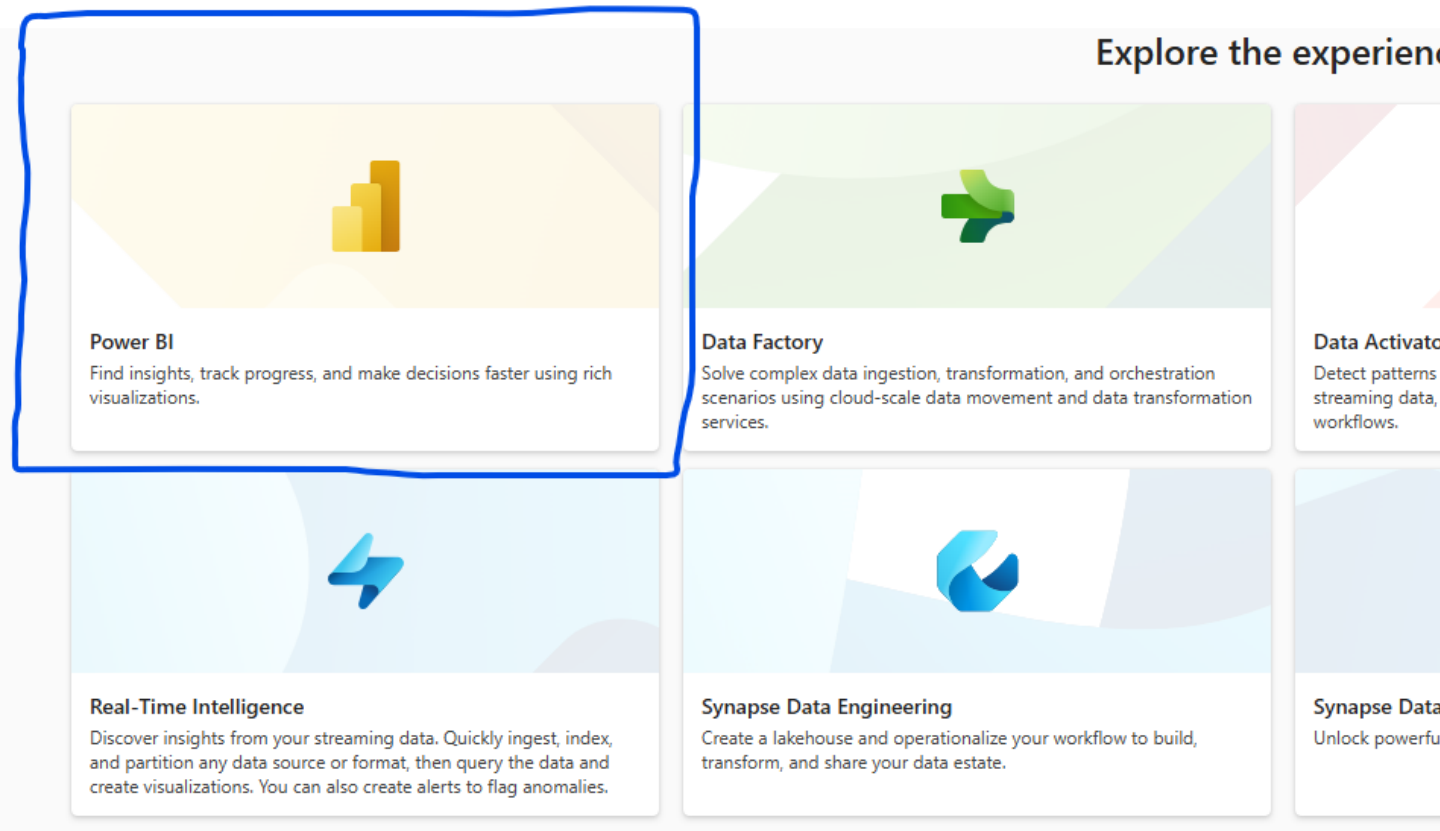
Validating that your program is writing to your event hub.

Log in to the Azure Portal and click on your event hub namespace. You should see activity in the line charts on this page indicating that data is being received by your event hub.

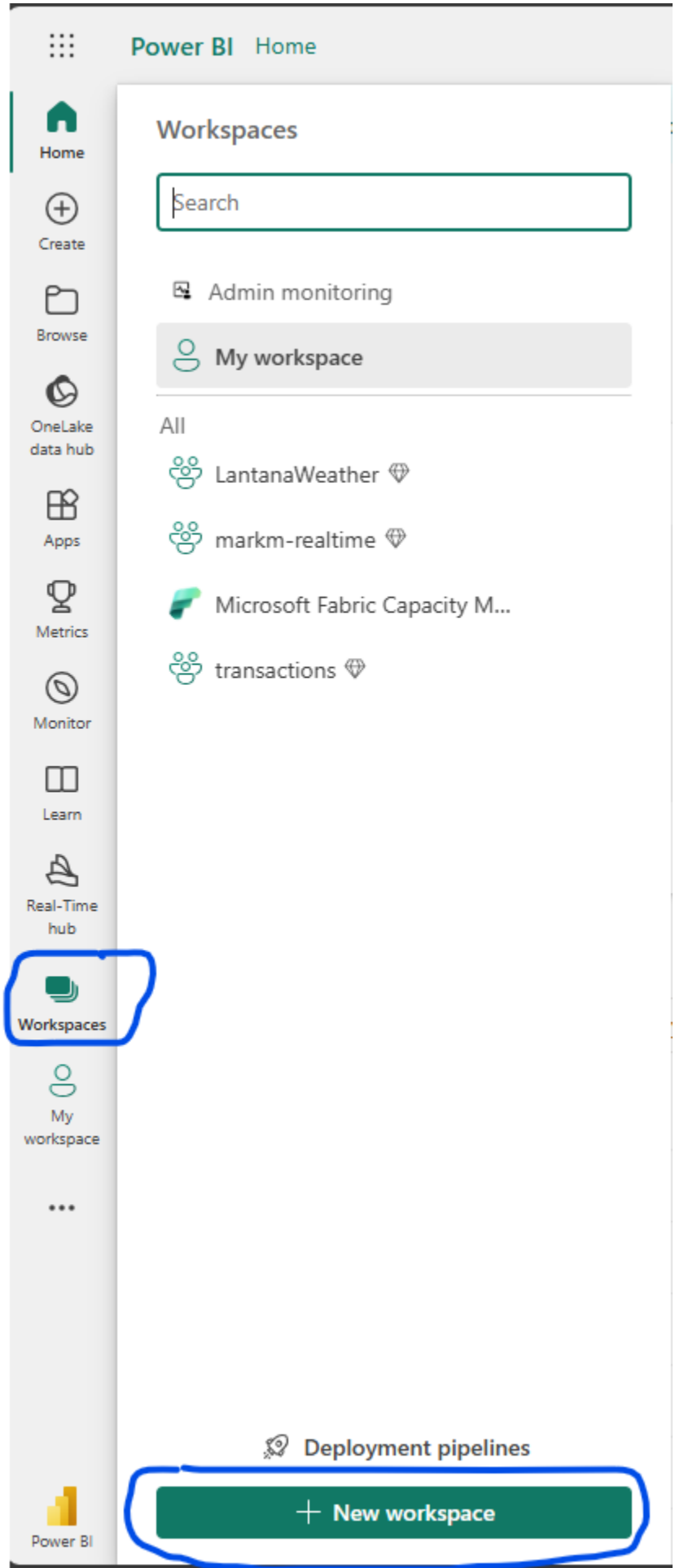


Pulling events from your event hub into Fabric.

Log into https://app.fabric.microsoft.com and select the Power BI persona.



Click on Workspaces and create a new workspace


Power BI
Home
Home
Create
Browse
OneLake data hub
Apps
Metrics
Monitor
Learn
Real-Time hub
Workspaces
My workspace
Power BI
Workspaces
Search
Admin monitoring
My workspace
All
LantanaWeather
markm-realtime
Microsoft Fabric Capacity M...
transactions
Deployment pipelines
New workspace

Type the name of your new workspace and select a Fabric Capacity under the Advanced Dropdown.

Create a workspace

Name *

FabricRTDemo

This name is available

Description

Describe this workspace

Domain

Assign to a domain (optional)

Learn more about workspace settings

Workspace image

Upload

Reset

Advanced

Contact list *

admin (Owner)

License mode

Pro

Select Pro to use basic Power BI features and collaborate on reports, dashboards, and scorecards. To access a Pro workspace, users need Pro per-user licenses. Learn more

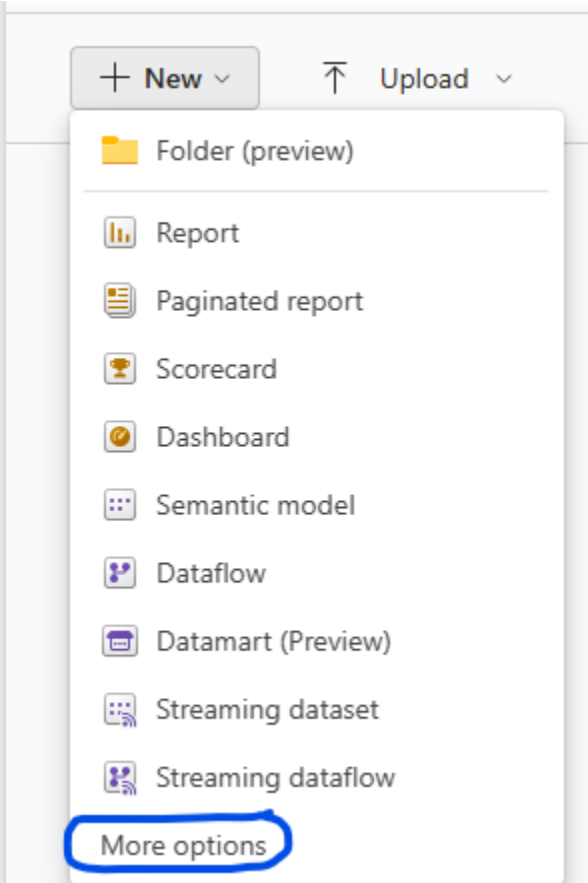
Trial

Select the free trial per-user license to try all the new features and experiences in Microsoft Fabric for 60 days. A Microsoft Fabric trial license allows users to create Microsoft Fabric items and collaborate with others in a Microsoft Fabric trial capacity. Explore new capabilities in Power BI, Data Factory, Data Engineering, and Real-Time Intelligence, among others. Learn more

Apply Cancel

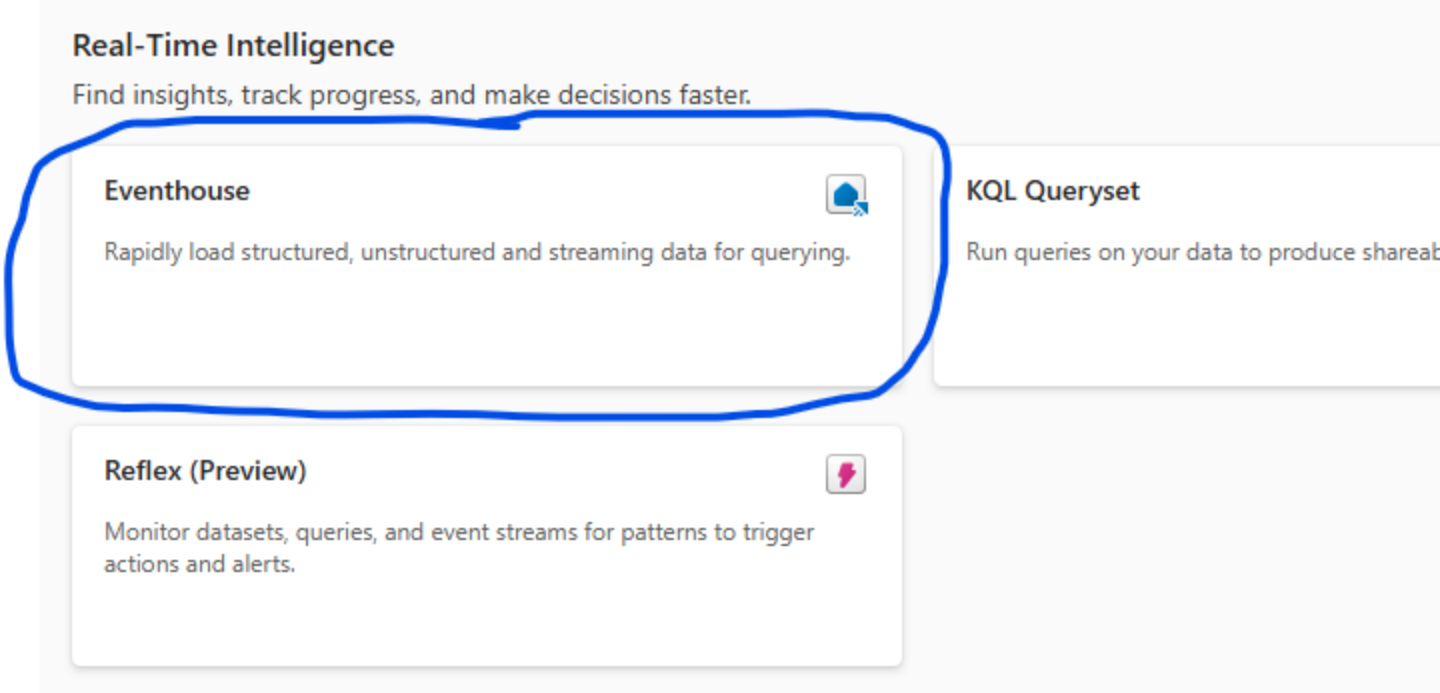
Click Apply and you will be taken to the new workspace.

Click New and select more options at the bottom of the dropdown.

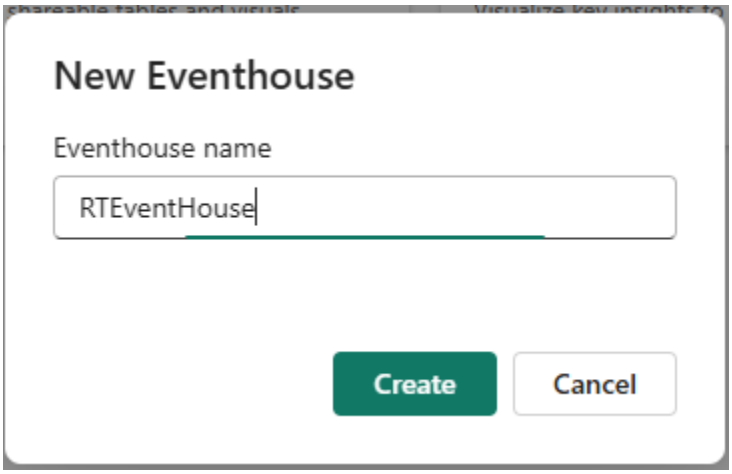


Scroll to the bottom of the options in New and Realtime artifacts will be at the bottom of the page.

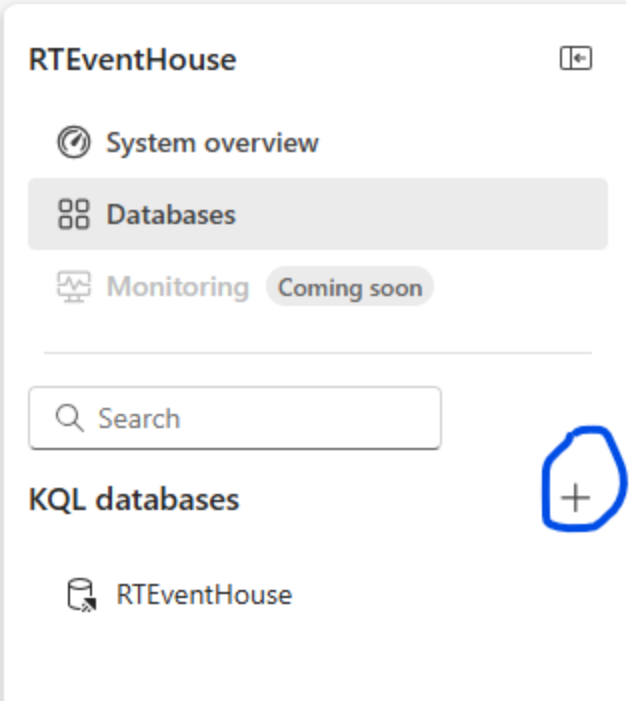
Create an Event House. You transactions from your event hub will land here. An event house is your KQL cluster where you will pull events for real time processing as well as contain a historical store for your real time events.



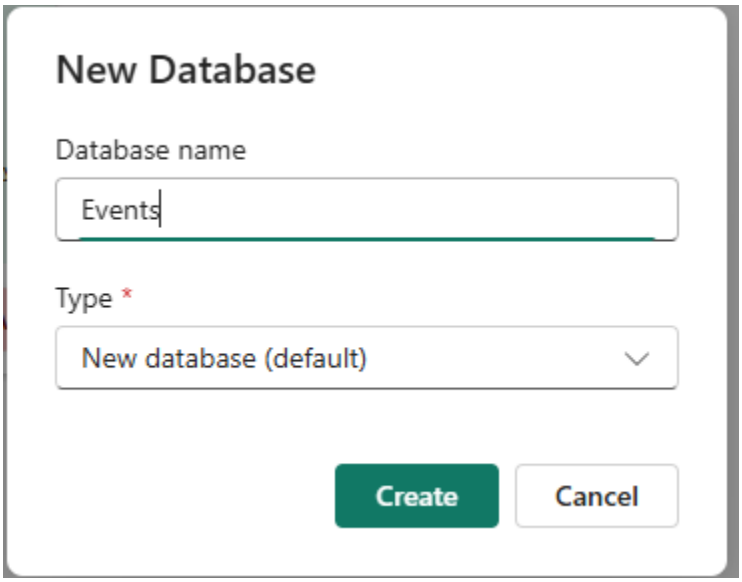
Provide a name for the Eventhouse.



Create a new database in the event house by clicking on the plus sign next to KQL databases.

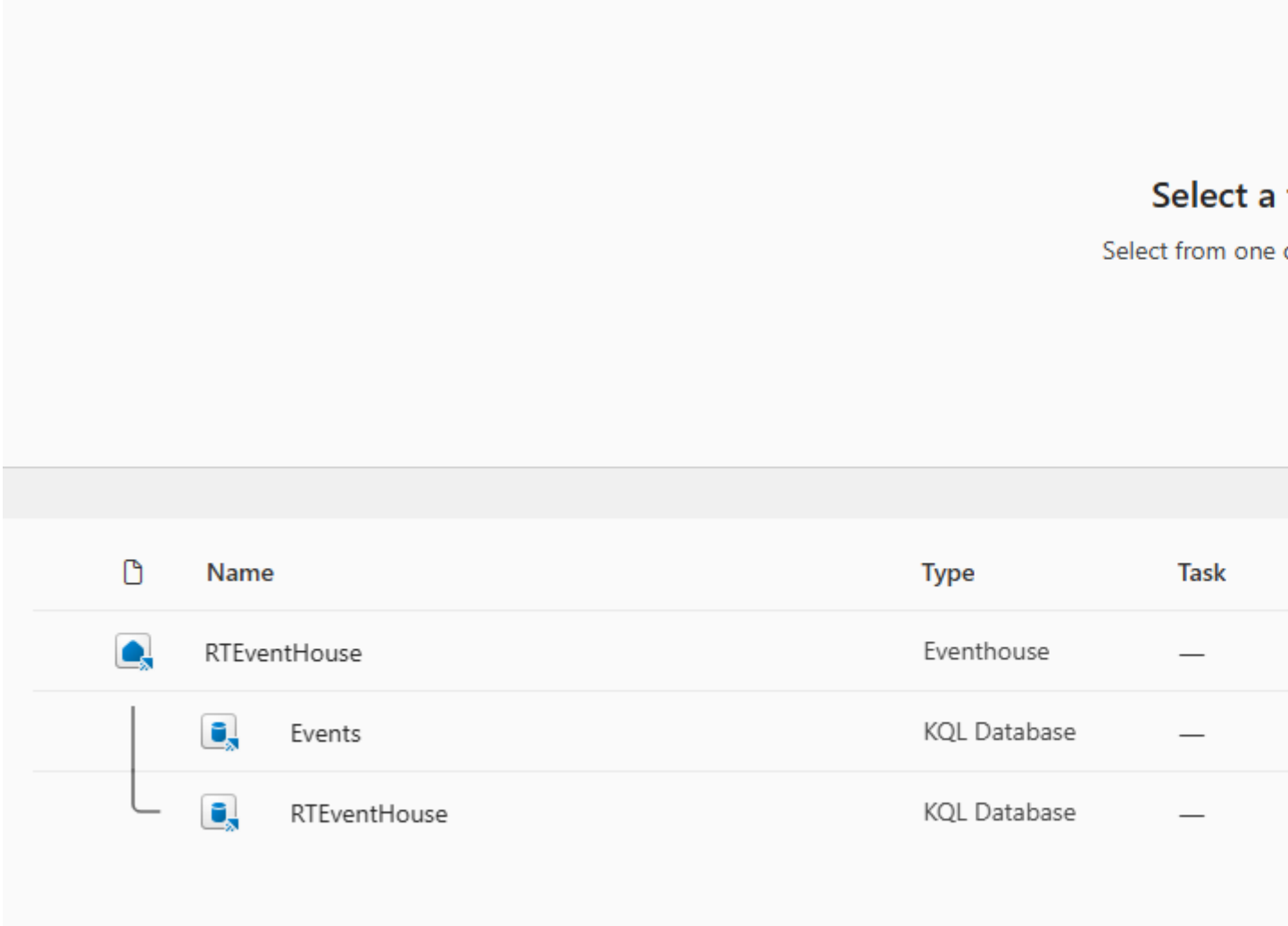


Enter a Database name and select New database. Click create.

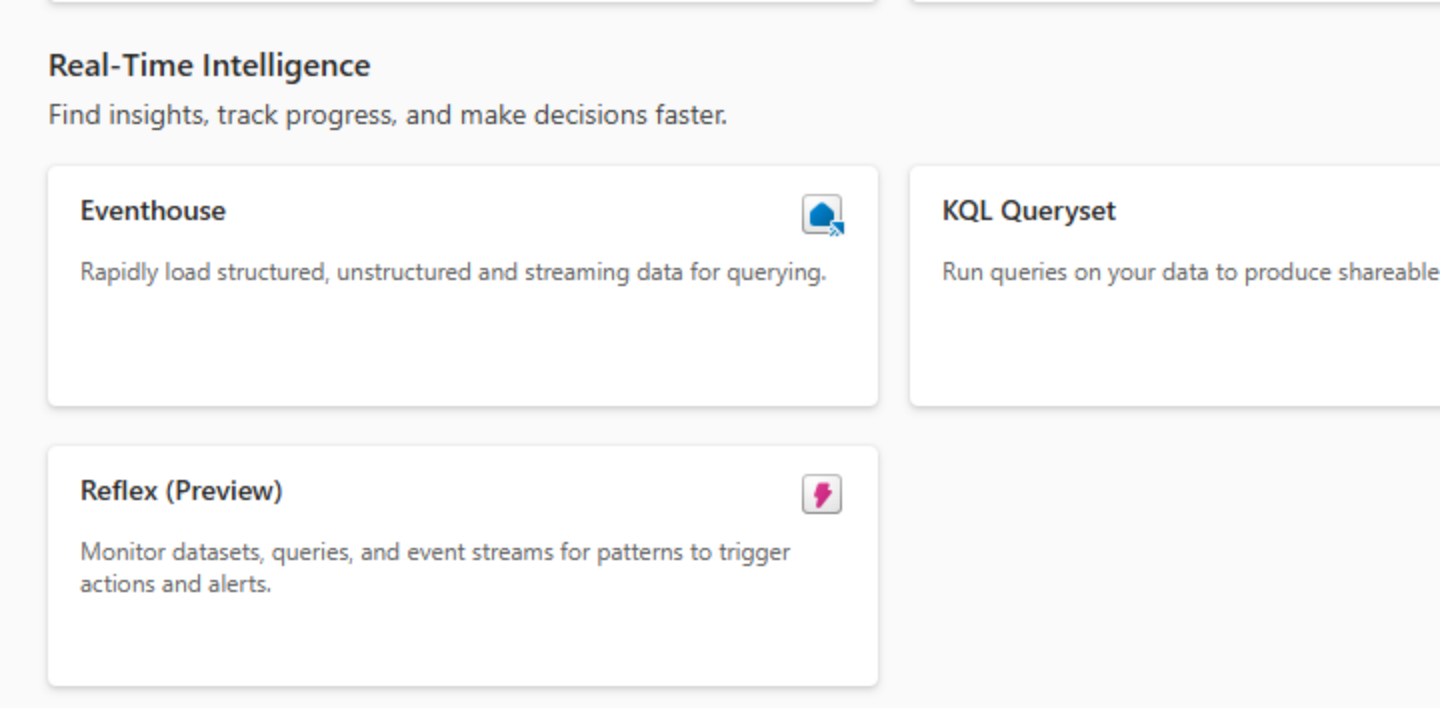


Go back to the root of your new workspace. You should see your newly created EventHouse.

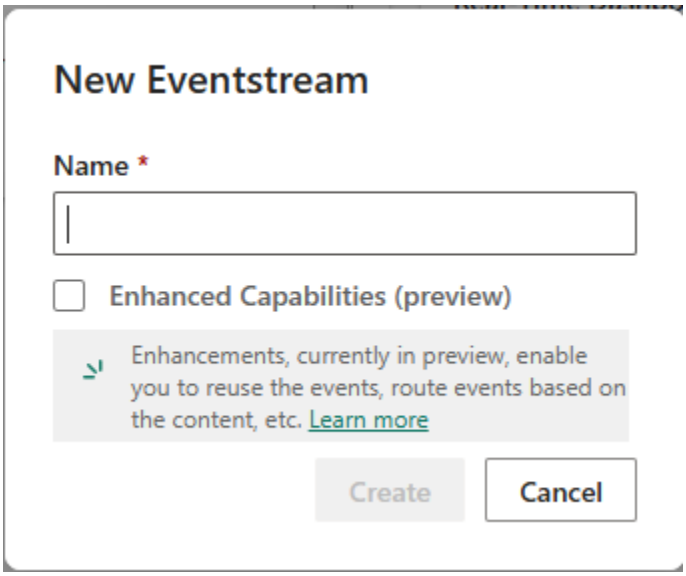

Monitor
Learn
Workspaces
FabricRTDemo
Events
RTEventHouse
RTEventHouse



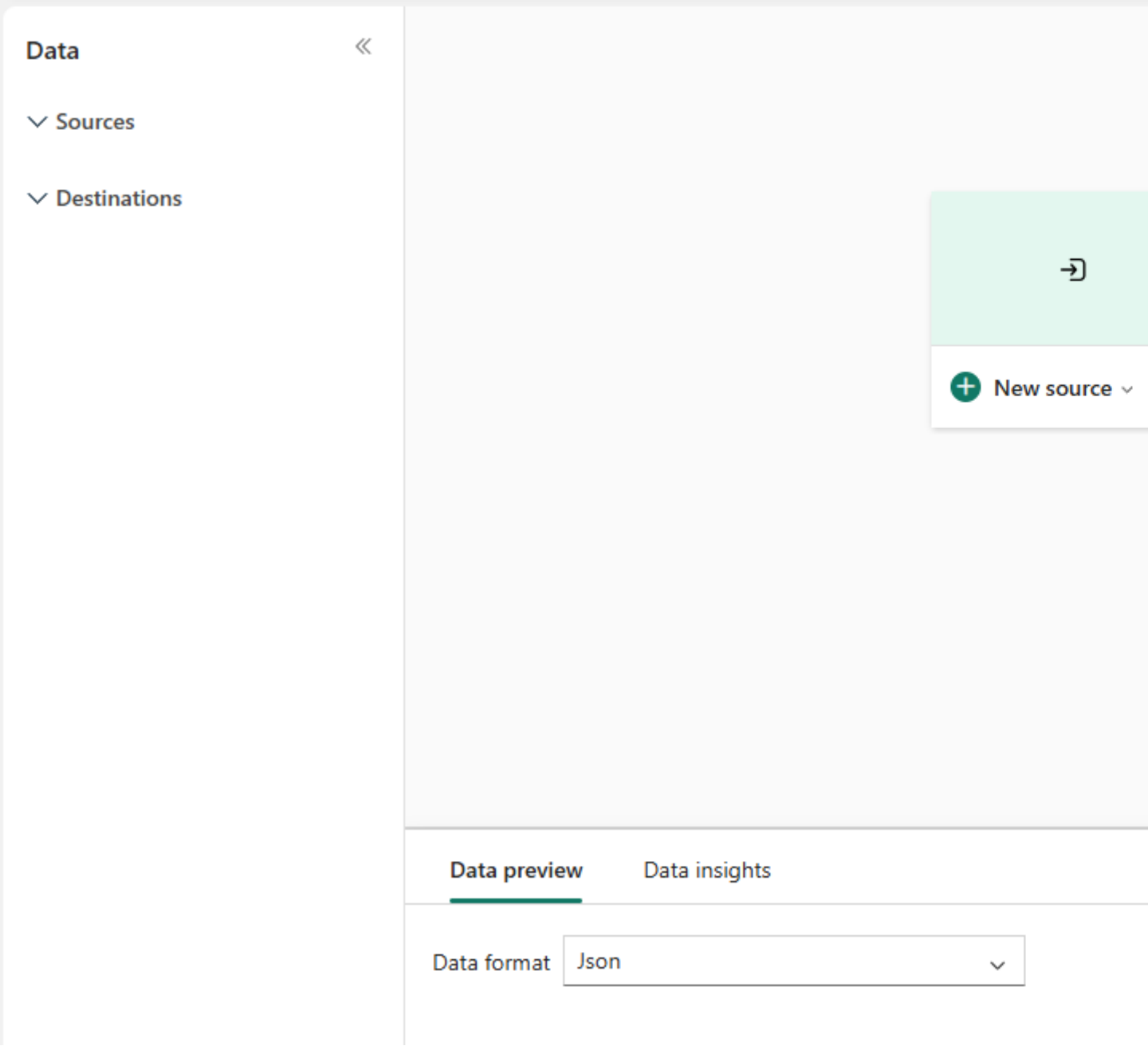
Click New again select more options and scroll to the bottom of the page. This time select Eventstream.



Enter the name of the Eventstream. Note you can select the Enhanced Capabilities (preview) option however, in this demo I am not using any of the new capabilities that option provides.

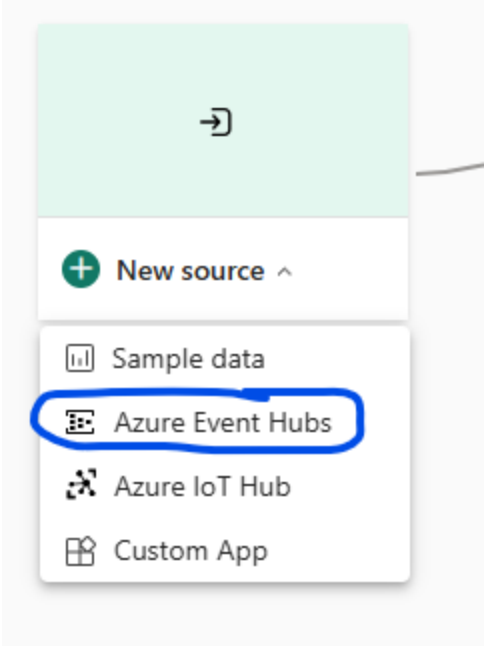


You should now see the following screen.

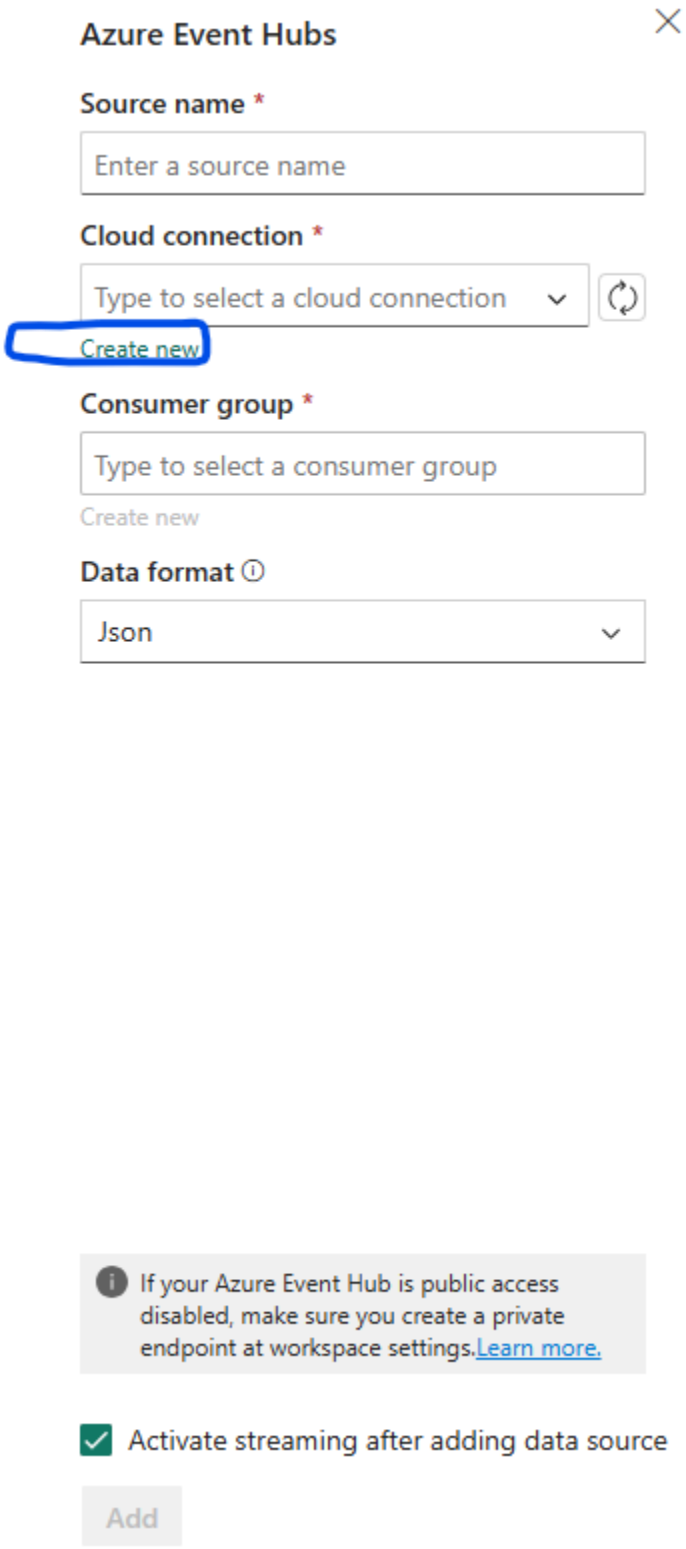


JSON is the format the events are in when the arrive from the Event Hub. Make sure Json is selected.

Click on New Source and select Azure Event Hubs



You will see the following dialog box. Type in an a Source name and under Cloud connection select create new to connect your event hub to Fabric.



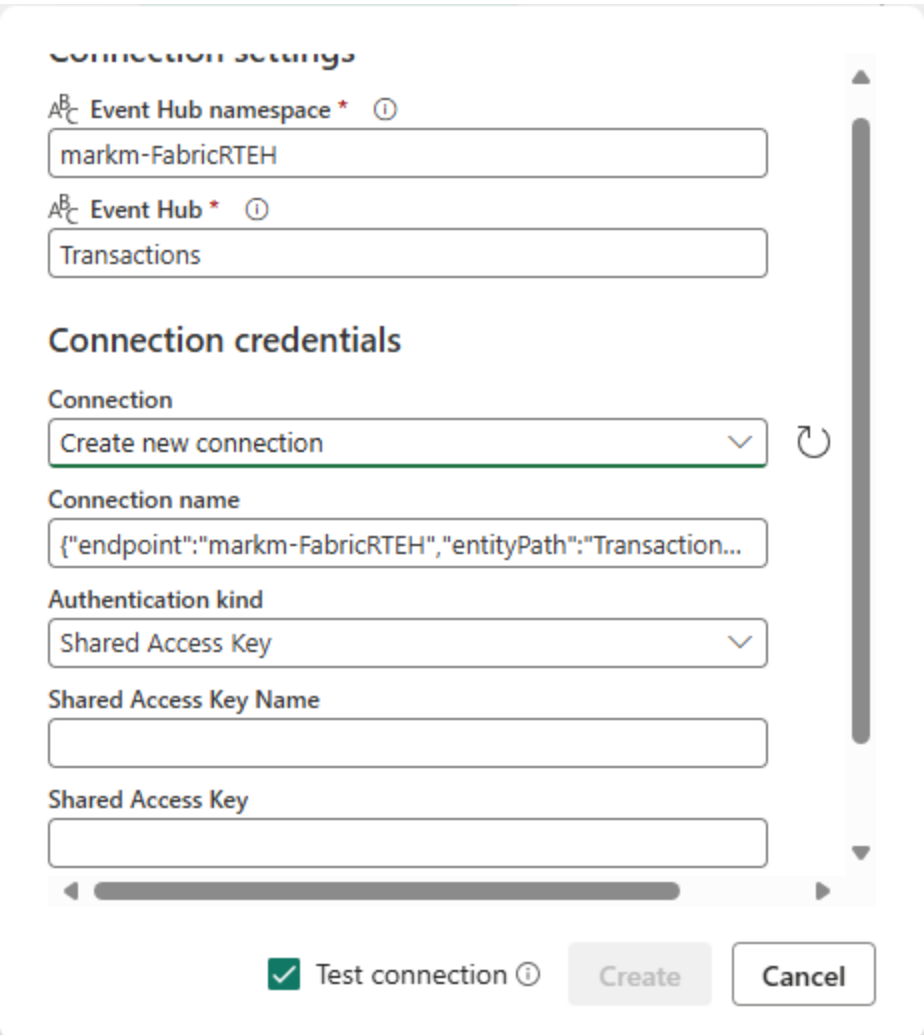
You will see the following dialog to set up a new connection to your event hub.

You will need to type in the Namepace name and Event Hub name that you created with the PowerShell script.

```
Copy the following lines of code and replace Lines 21 through 24 in the C# program with them
-----------------------------------------------------------------------
--                              C#                                   --
-----------------------------------------------------------------------

private static readonly string EHNamespace = "markm-FabricRTEH";
private static readonly string EHName = "Transactions";
private static readonly string EHKeyname = "RootManageSharedAccessKey";
private static readonly string EHKey = "xG16gV[redacted]=";


-----------------------------------------------------------------------
--                            Python                                 --
-----------------------------------------------------------------------

EHNamespace = "markm-FabricRTEH"
EHName = "Transactions"
EHKeyname = "RootManageSharedAccessKey"
EHKey = "xG16[redacted]="

PS C:\Users\markm>
```

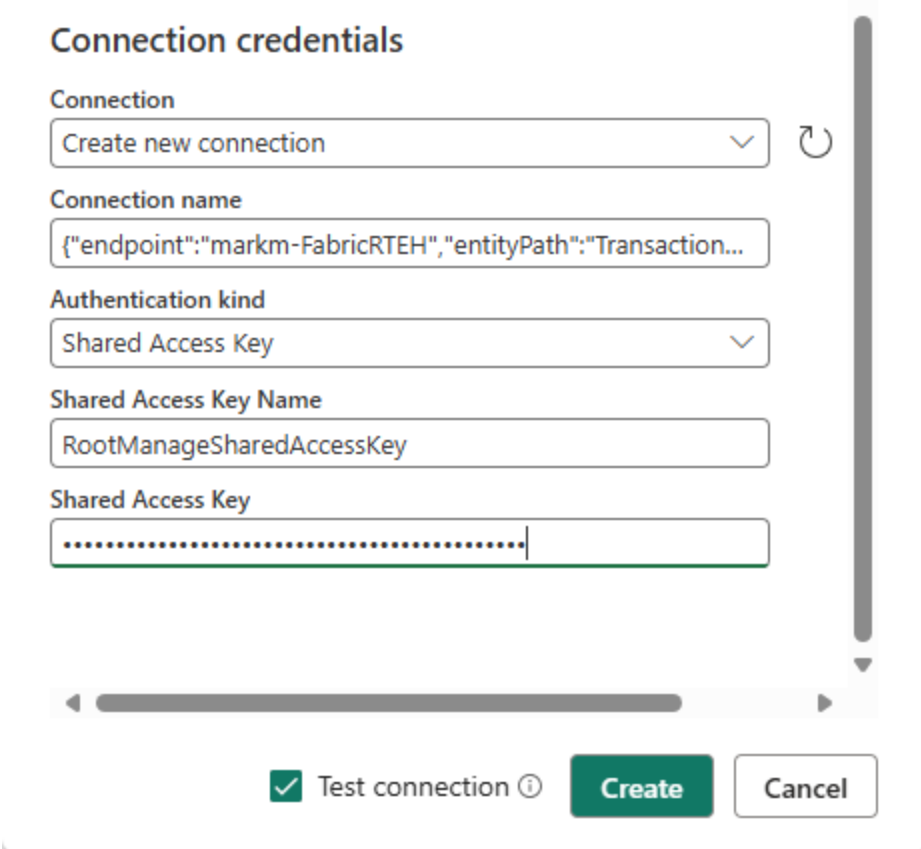
Type in your namespace name and Event Hub name, the connection string will be built for you and will show up in Connection name.

Select the Connection dropdown and select Create new connection.



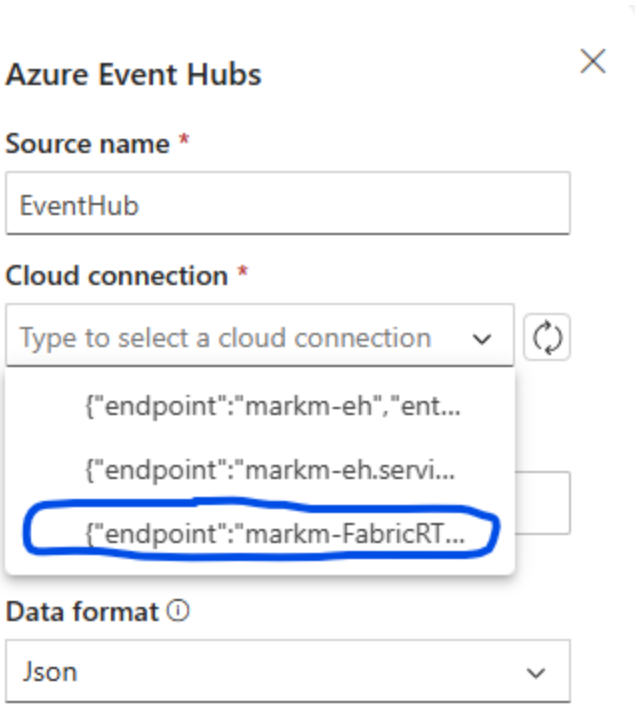
In the Authentication kind field, select Shared Access Key

Type in your Key name and key information from the PS1 Script output.



Click create.

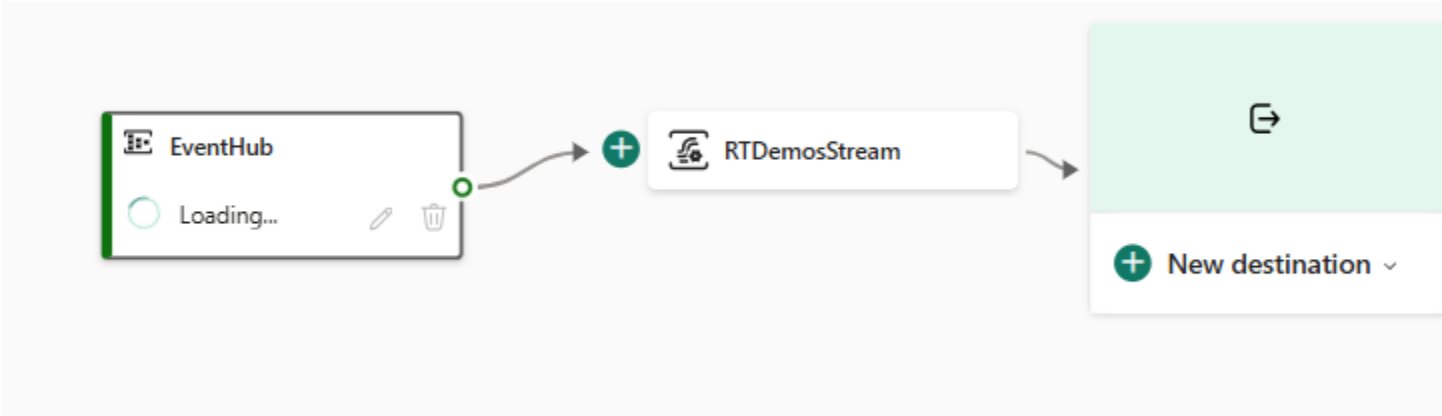
Your new Event hub is now connected to Fabric and you can select it in the Cloud connection drop down. If this is the first event hub you have connected to Fabric it will be the only one in the list. In this case I have a few so I will select the new one.



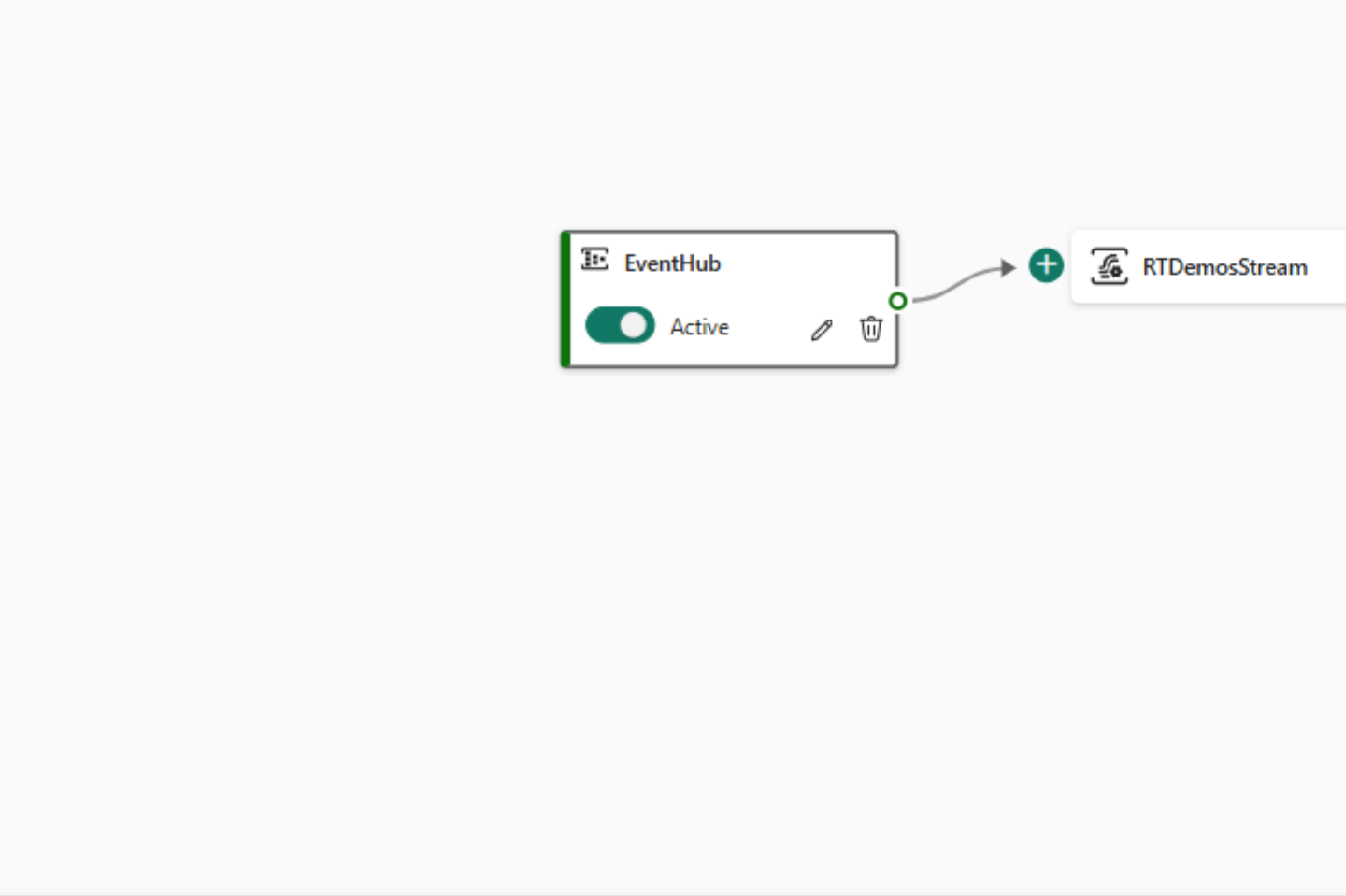
The Consumer Group is listed in the dropdown for Consumer Group and the default name is $Default

Select the consumer group, click the checkbox for Activate streaming after adding data source and click Add

You will see Loading in the EventHub box in your Event Stream



Once that changes to Active, you are now streaming data and you can preview that data by select Data preview.



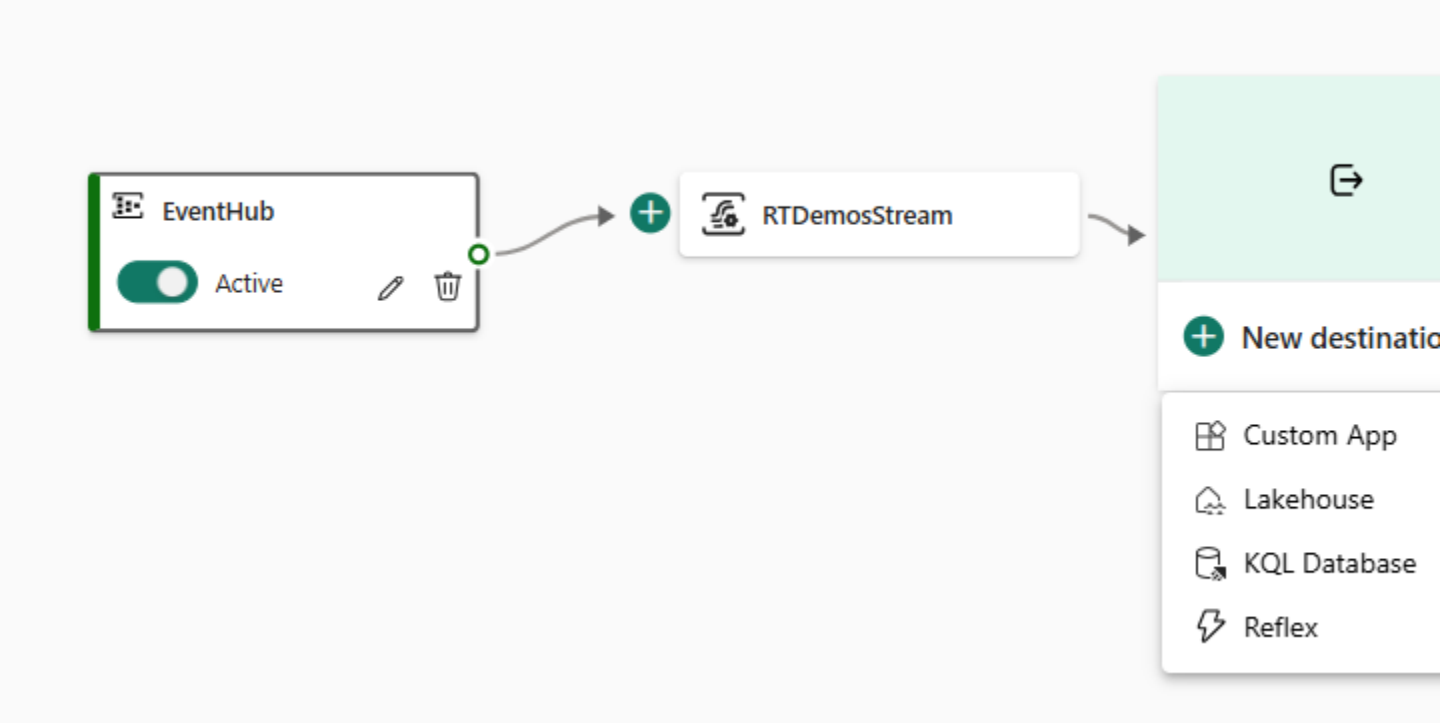
Details | **Data preview** | Data insights | Runtime logs

ast refreshed 08/19/24 02:23:11 PM

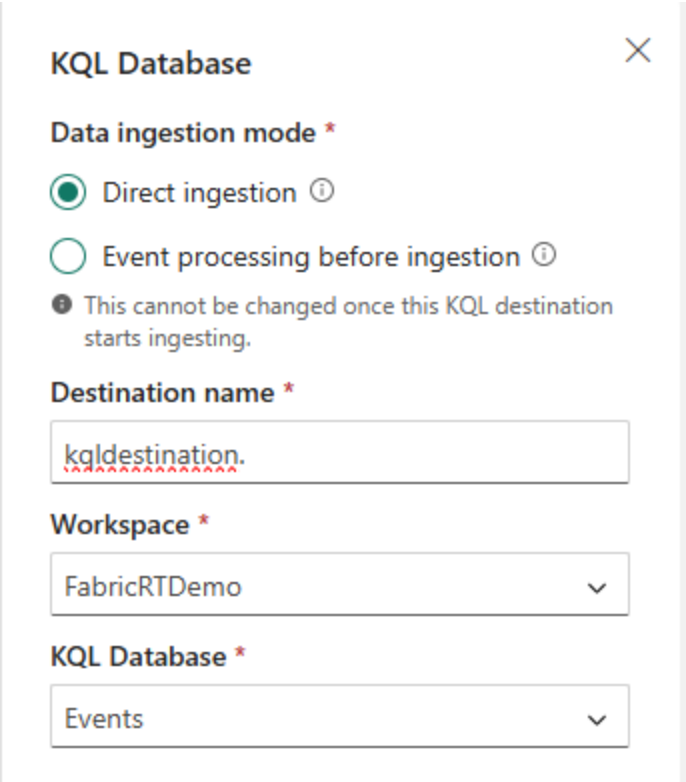
DateTime	Category	Deposit	Withdrawal
2024-08-19T18:53:07.3555893	Entertainment		751.95
2024-08-19T18:53:08.3890609	Clothing		635.94
2024-08-19T18:53:09.4215975	Personal Care		850.74
2024-08-19T18:53:10.4562655	Personal Care		592.10
2024-08-19T18:53:12.5291942	Clothing		53.65
2024-08-19T18:53:16.6802182	Deposit	8528.4	

Notice that the DateTime column written by the application and the Event ProcessedUtcTime and EventEnqueuedUtcTime are not all the same exact time. This will come up later in this document.

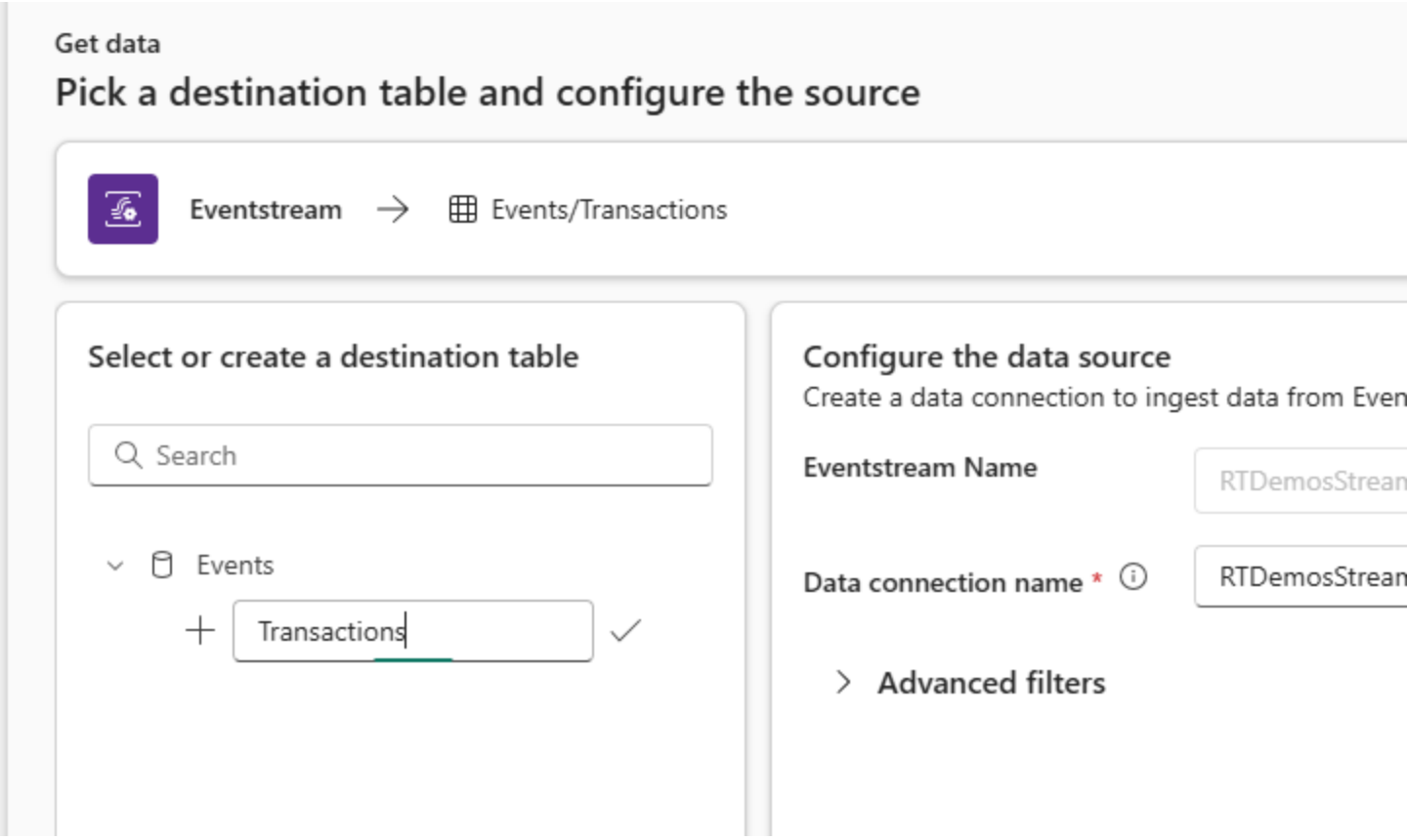
Now we need to select a destination for the events. In this case we will select the EventHouse we created in a previous step which is a KQL Database.



Select the KQL Database and then select Direct ingestion. Select the Workspace you used to create your Event House and select the database name.



Click Add and configure

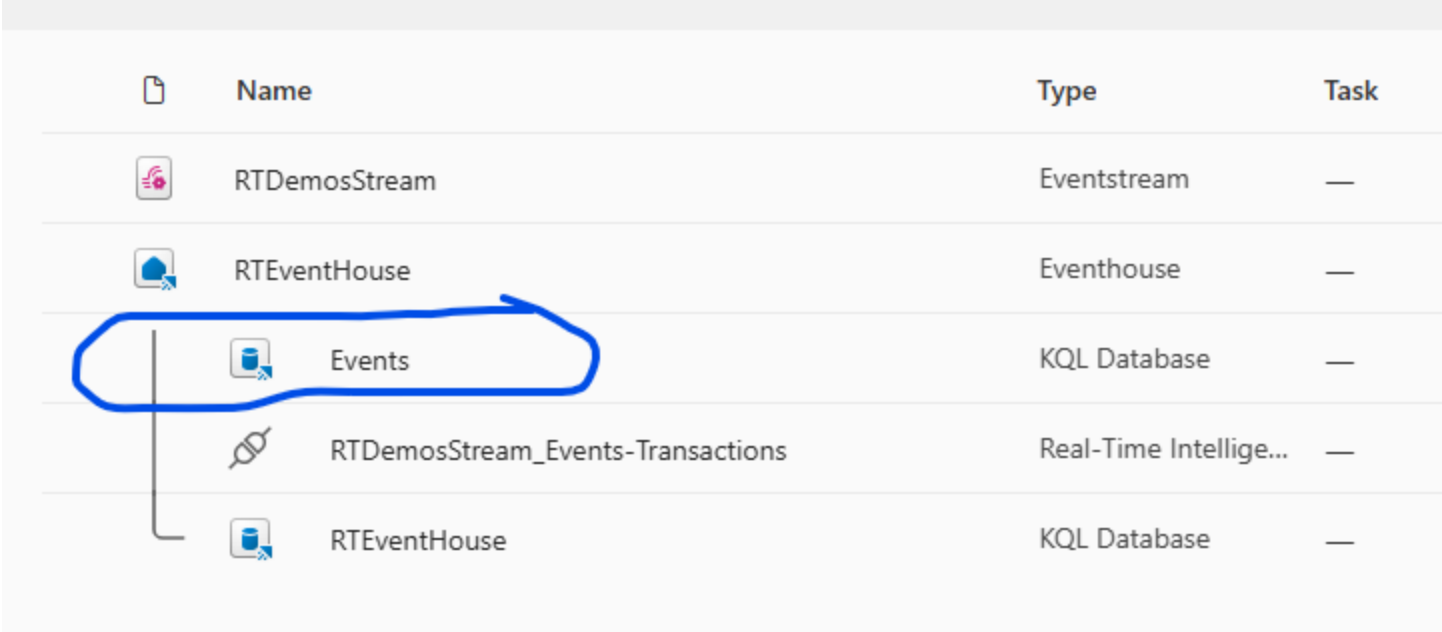


You will be prompted to add a new table, click on New Table and Enter a Name

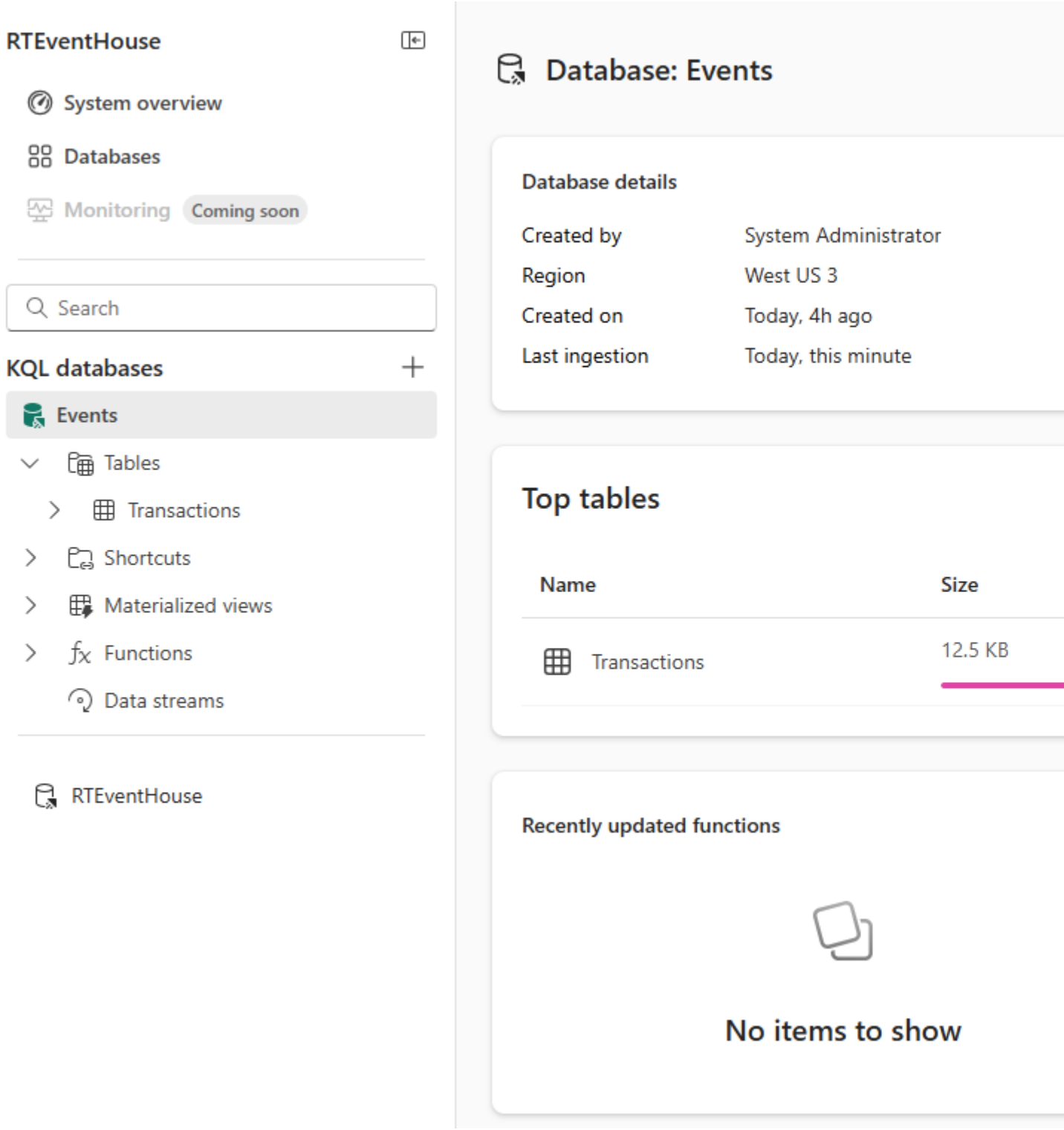
Click Next

You will see events that have been pulled in from Event Hub. After you click Finish, events will be streamed into the KQL table from the Event Hub through your Event Stream. Click Finish and click Close.

Go back to your workspace and click on your KQL Database



You will see information about your database and there is an explore your data button on the top right-hand side of the screen. Click on Explore your data



You will see some sample queries you can explore at a later time. For the purposes of this demo use the following query.

Transactions

| where DateTime >= ago(30s)

This query will pull the last 30 seconds of data. Everytime you issue this query it will pull the last 30 seconds of data and data is continuously being written to the KQL table so this query is essentially and 30 window from to 30 seconds ago of data. Each refresh will drop records older than 30 seconds and pick up new ones.

Explore your data

Run Save as KQL queryset

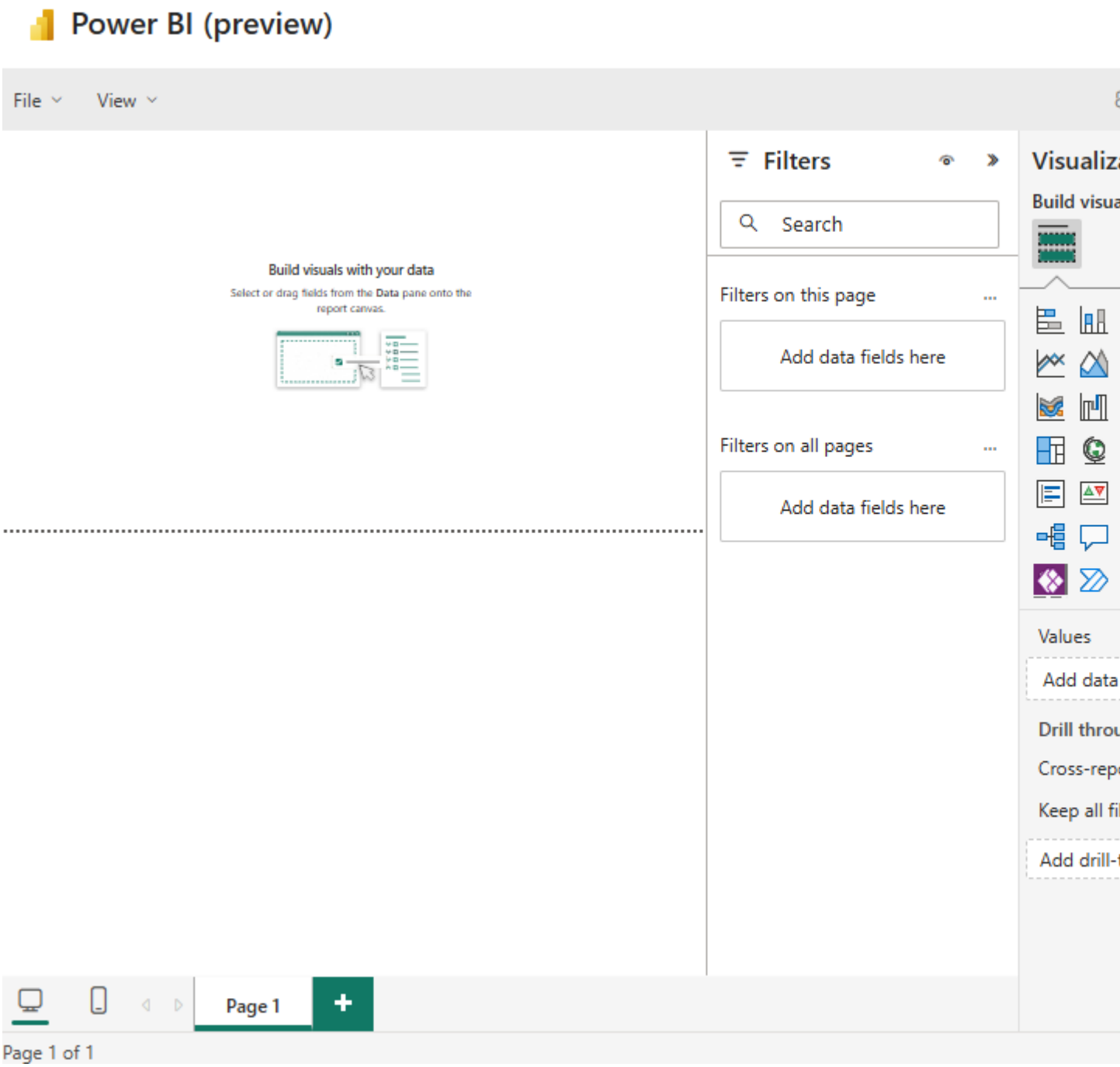
```
1  Transactions
2  | where DateTime >= ago(30s)
3
4
```

Table 1 Stats Search Done (

DateTime	Category	Deposit	Withdrawal	Balance	EventProcessedU
2024-08-19 22:39:15.8290	Utilities		224.38	1,595.77	2024-08-19 22:39
2024-08-19 22:39:16.8640	Transportation		774.69	821.08	2024-08-19 22:39
2024-08-19 22:39:17.8970	Deposit	6640.49		7,461.57	2024-08-19 22:39
2024-08-19 22:39:18.9300	Personal Care		660.82	6,800.75	2024-08-19 22:39
2024-08-19 22:39:19.9710	Education		155.75	6,645	2024-08-19 22:39
2024-08-19 22:39:21.0060	Utilities		547.35	6,097.65	2024-08-19 22:39
2024-08-19 22:39:22.0390	Home Maintenance		929.39	5,168.26	2024-08-19 22:39
2024-08-19 22:39:23.0770	Personal Care		192.12	4,976.14	2024-08-19 22:39
2024-08-19 22:39:24.1140	Savings		605.99	4,370.15	2024-08-19 22:39
2024-08-19 22:39:25.1490	Clothing		150.61	4,219.54	2024-08-19 22:39

In the upper right-hand corner click on Power BI. This will take the Kusto Query and imbed it into a Power Query query that is part of a semantic model and allow you to build a report using that query.

After you click in the Power BI button you will be dropped into the online report authoring page in Fabric.

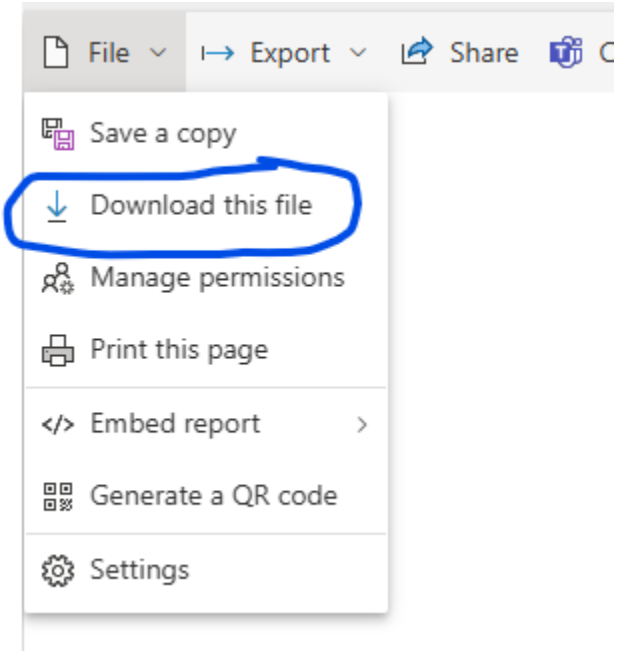


Click on File and save the report to the workspace with the rest of the demo artifacts. The save option will default to your “My Workspace” so if your forget to select a workspace the report will be in your My Workspace.

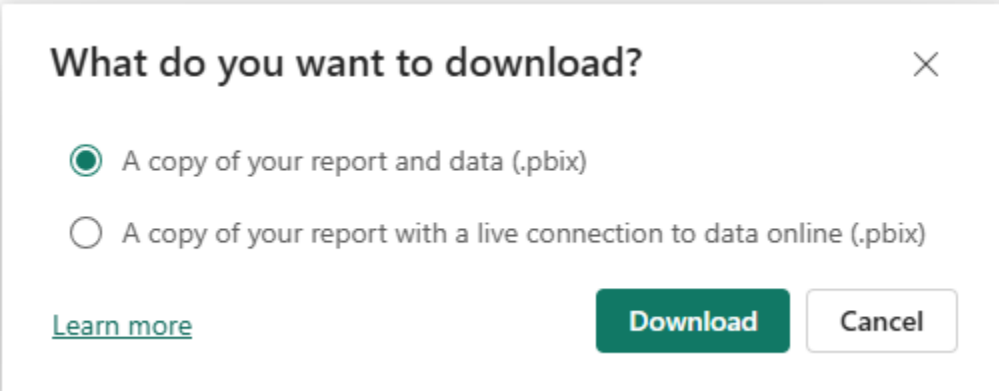
Go to the workspace with he report open the report and download the PBIX file we will finish authoring the report and do the demo from Power BI desktop.

Why not in Fabric you ask? Remember when I pointed out the different times in the data and the query we used to pull the data is the Time Stamp written by the application running on our desktop? The reason I author the report this way is that my demo application and Power BI report are now using the same clock. If I did this in Fabric (which you can do) I would need to pick a time from a clock in Fabric, usually that is the EventEnqueuedUtcTime column. If my clocks are not in sync between the time in the data and the Power BI report. I may or may not get a full 30 seconds of data.

Open the Report in your workspace and click on File and select Download this file.

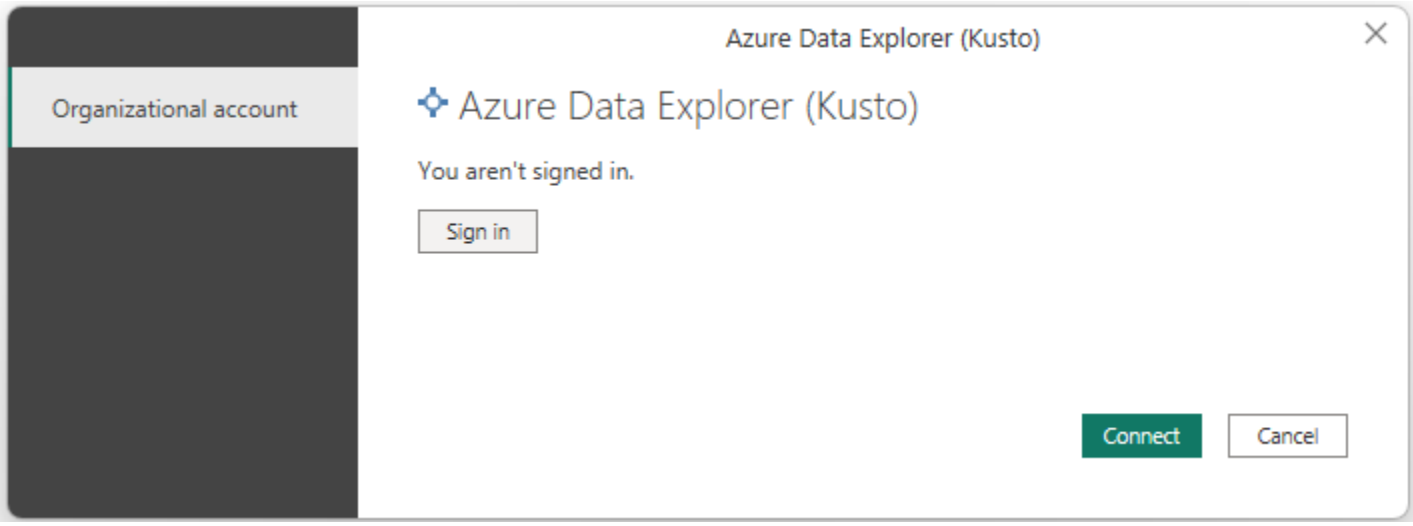


Select A copy of your report with data, not the live connection.



Open the report on your desktop in Power BI desktop. There is a new feature we will be using, make sure you have a recent copy of the Power BI desktop app.

Once the report is opened, you will be prompted to sign into your Kusto Database.



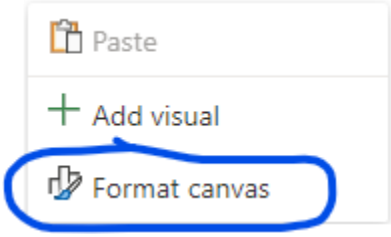
Once you have signed in click on Connect.

At this point you can create any visual in the gallery to display your real time data. Lets start with a simple grid that looks like this:

DateTime	Category	Deposit	Sum of Withdrawal	Sum of Balance
8/19/2024 11:05:47 PM	Savings		988.62	3,257.65
8/19/2024 11:05:48 PM	Internet		917.74	2,339.91
8/19/2024 11:05:29 PM	Eating Out		897.60	3,444.93
8/19/2024 11:05:41 PM	Fitness		880.74	6,571.96
8/19/2024 11:05:53 PM	Fitness		857.60	910.98
8/19/2024 11:05:33 PM	Mortgage		759.54	1,414.75
8/19/2024 11:05:44 PM	Groceries		691.72	5,017.55
8/19/2024 11:05:55 PM	Savings		660.73	6,995.62
8/19/2024 11:05:42 PM	Home Maintenance		547.56	6,024.40
8/19/2024 11:05:31 PM	Personal Care		541.63	2,607.73
8/19/2024 11:05:40 PM	Gifts		526.43	7,452.70
8/19/2024 11:05:45 PM	Insurance		448.77	4,568.78
Total			**11,796.73**	**98,818.80**

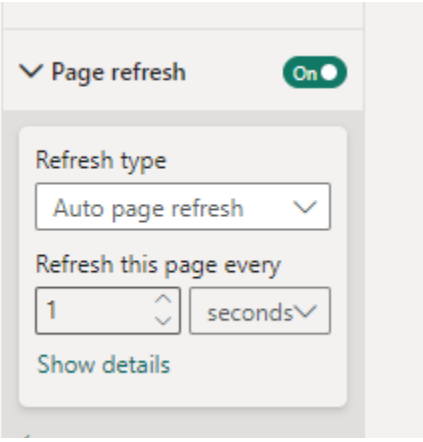
You will notice that once you drag a numeric field to the visual, the visual will switch to a pivot table. Just drag the Deposit, Withdrawal and Balance columns over and switch visual back to a grid.

Once you have this visual built, you will also notice that your data is not updating in real time. This is the new feature I mentioned above. Right Click anywhere on the canvas of the report that is blank.



Select Format canvas

Turn Page refresh on and click the drop down to configure it. Configure it to look like this. Once you do your report will automatically refresh once per second.

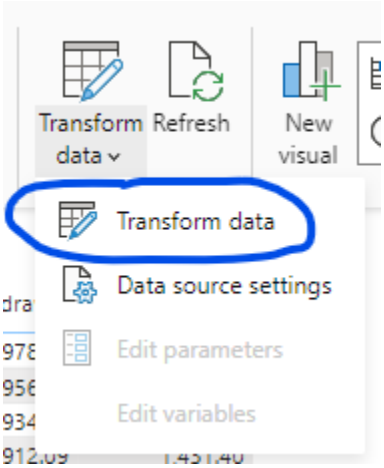


You can continue to build more visuals, everything on the report will refresh in 1 second intervals and they will all refresh at the same time.

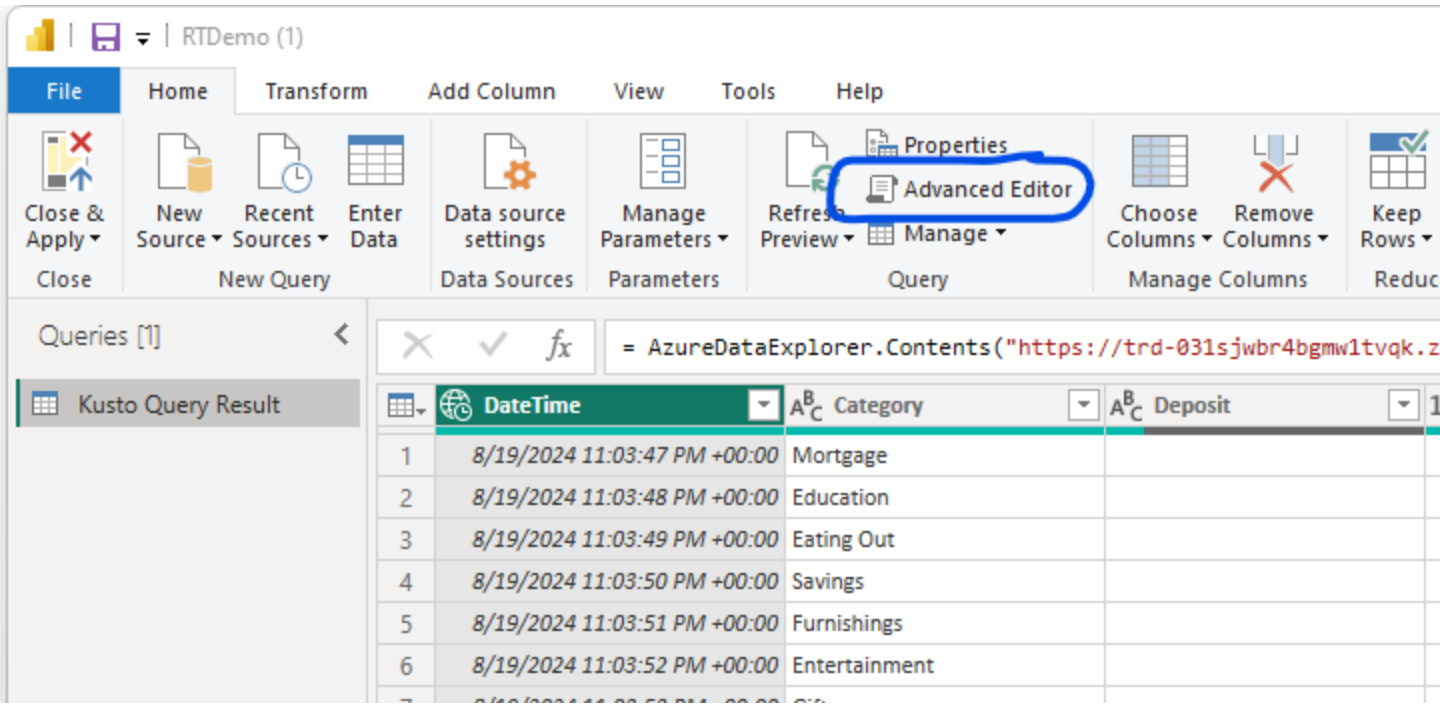
Congratulations you are now ready to do the demo.

Let's take a moment to look under the covers.

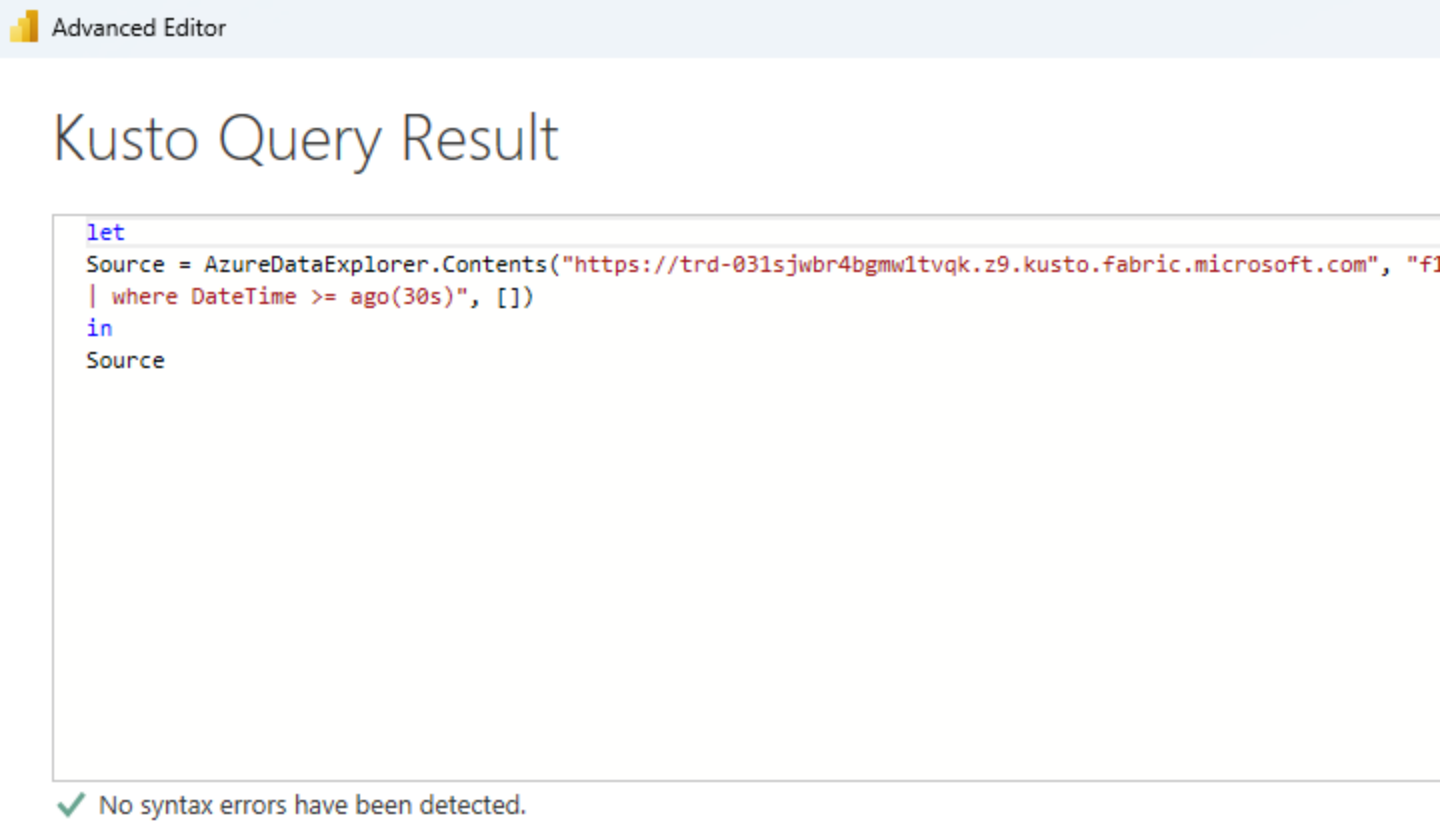
Click on Transform Data in the ribbon on top of the report, Select the Transform data from the dropdown.



This will bring you into Power Query, now select Advanced Editor.



In the Advanced Editor screen you will see your query. What is happening is that every second this query is being issued to your Kusto Table and pulling back the results. For each refresh we pull back the results of the query which will give us the last 30 seconds of data.



Let Format the Query so that it is more readable to understand what is happening here.

After reformatting the query looks like this.

```
let
Source = AzureDataExplorer.Contents
(
  "https://trd-031sjwbr4bgmw1tvqk.z9.kusto.fabric.microsoft.com",
  "f12d8cbe-1698-4e08-a478-8b9e3a481381",
  "Transactions
  | where DateTime >= ago(30s)",
[]
)
in
Source
```

You can see that we have a query that is calling AzureDataExplorer and issuing a Kusto Query.

The first line in the query is the full URL to your kusto Event House (Cluster)

"https://trd-031sjwbr4bgmw1tvqk.z9.kusto.fabric.microsoft.com",

The second line is the Database ID

"f12d8cbe-1698-4e08-a478-8b9e3a481381",

The 3rd and 4th lines are your Kusto query as you wrote them in Kusto under Explore your data (Data Explorer)

"Transactions

| where DateTime >= ago(30s)",

In the program I am throttling new events to be written once per second so every second I get a new event going through the system.

What you should notice in the report is that roughly every second I am adding a new record to the top of the grid chart and one record is falling off the bottom of the grid since it is now 31 seconds old.

This is possible because I am using the same clock to write my date time stamp as I am using in for Power BI desktop.

If you choose to do the entire demo in Fabric, you will need to play around with your Kusto query to pull a little more that 30 seconds of data to get the same effect.

This is not an actual real-time report since it take time for an event to move from the Event Hub into Fabric, through your Event Stream and into your Kusto Database. The data you are seeing is most likely delayed by a second or two from the time it was written to the Event Hub.

It is important to note that when referring to this demo it is a Near Real Time demo since there is a slight delay from the time an event is written to the time it is displayed in a PBI report.