Kafka/Event Hub to Event Hub demo

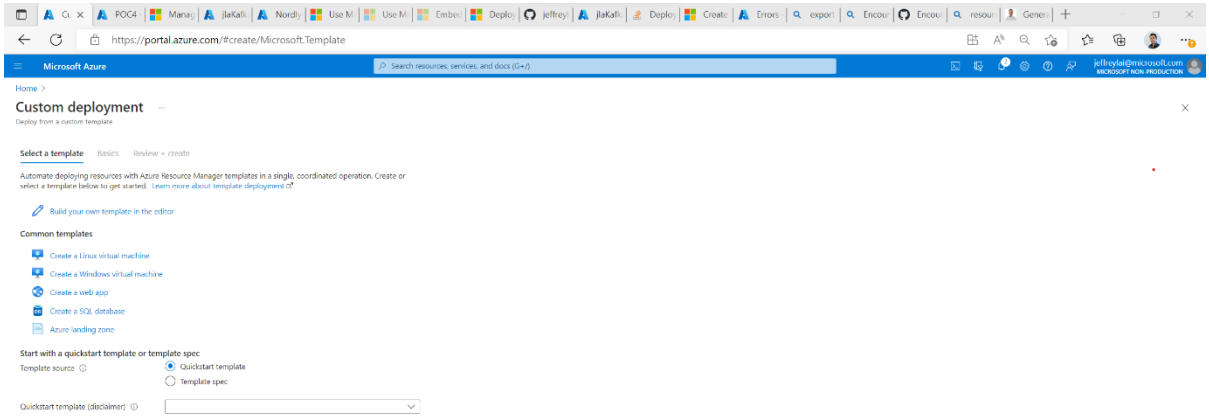
This document is a guided lab for: 7effrey89/Kafka-Topic-Replication-To-Kafka-EventHub:

Contents

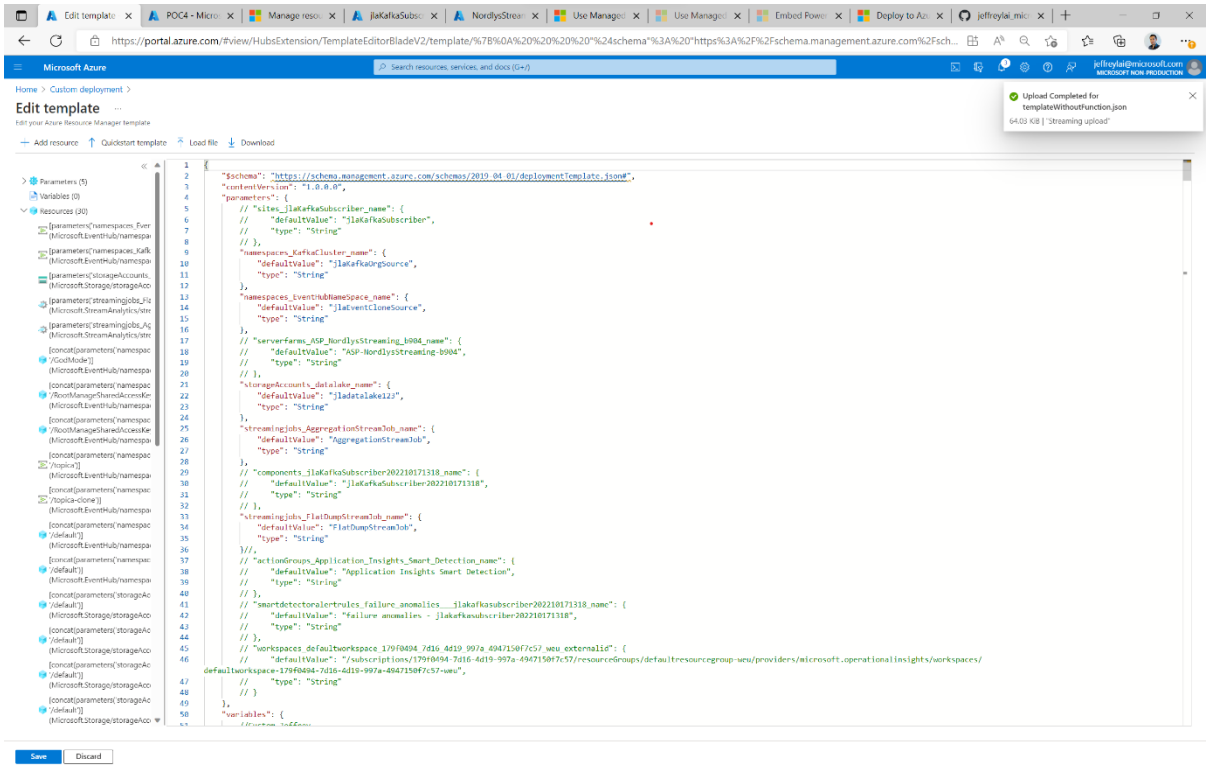
Arm template deployment

Background: Below steps will deploy some pre-created resources in order to safe time of the lab participant. The lab will create two EventHub Namespaces with 1 Event Hub, one storage account, and two stream analytics jobs that are half configured.

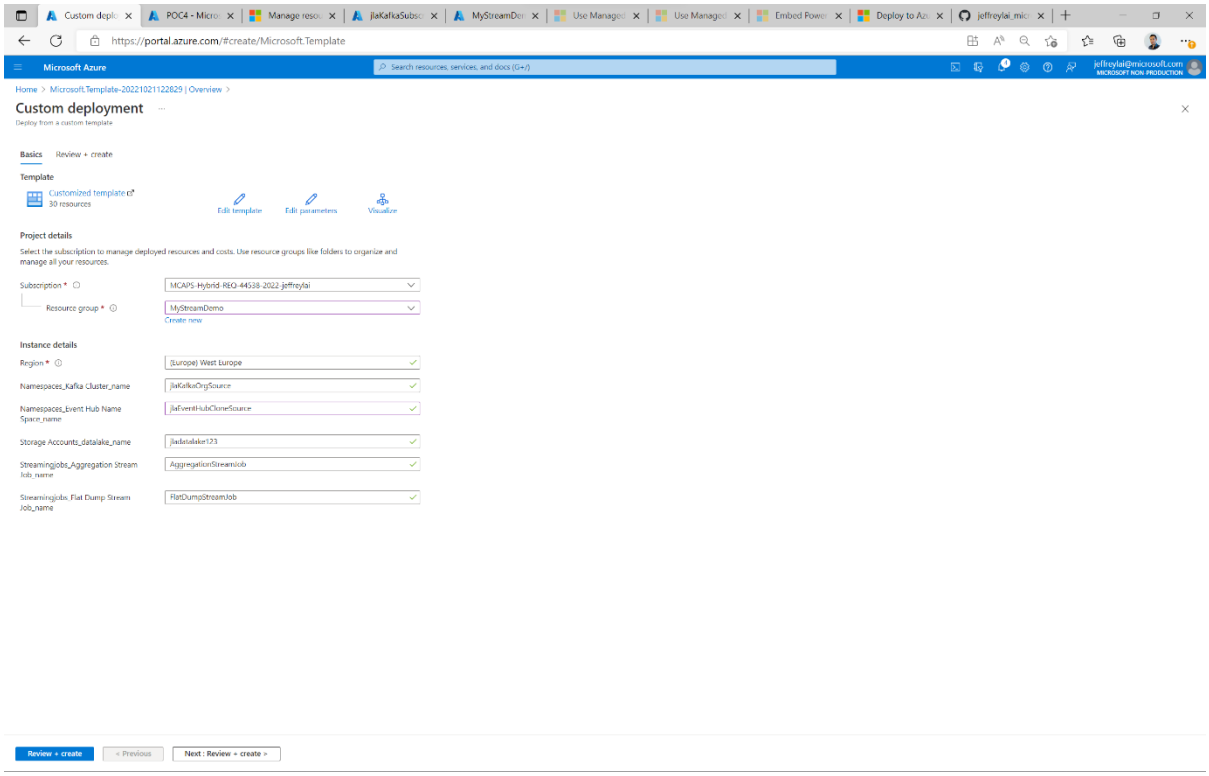
1. In the search bar, search for “Deploy from a custom template”.
2. Click on “Build your own template in the editor”
3. Click on “Load File”



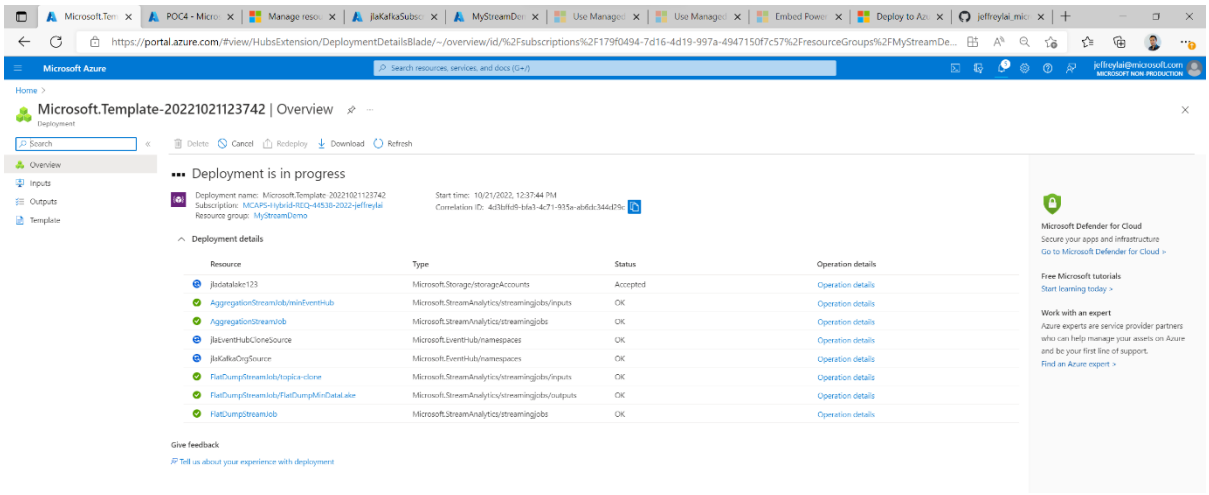
4. Load the json file in the folder: “Arm Template”



5. Fill out the parameters for the Azure Resources to be created. Please make them unique.

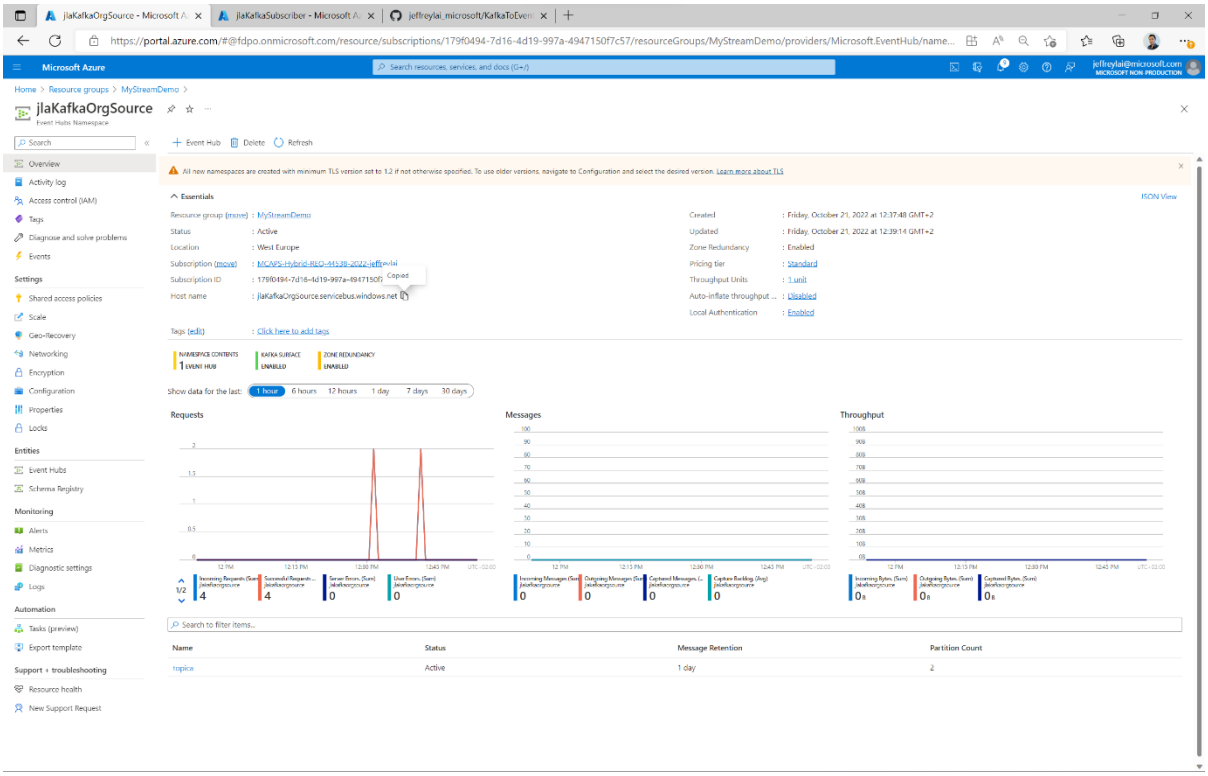


6. Await the deployment to finish.

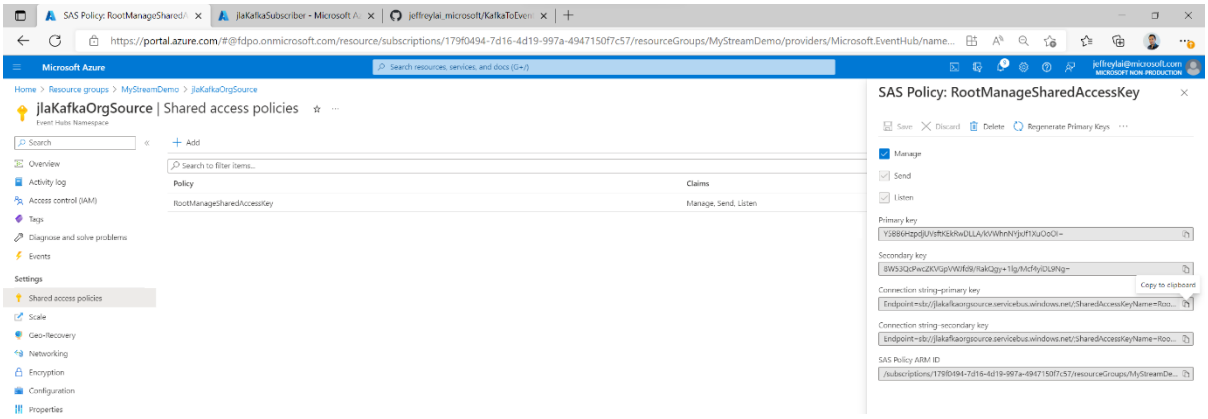


7. When the deployment has finished. Go to the resource group, and find the Event Hub namespace that represent: “JlaKafkaOrgSource”.

8. Note down the Host name of the resource in a notepad document. You will need it later.



9. On the left panel, click on “Share Access Policies”, and copy the “Connection string-primary key” to a notepad document. You will need it later.

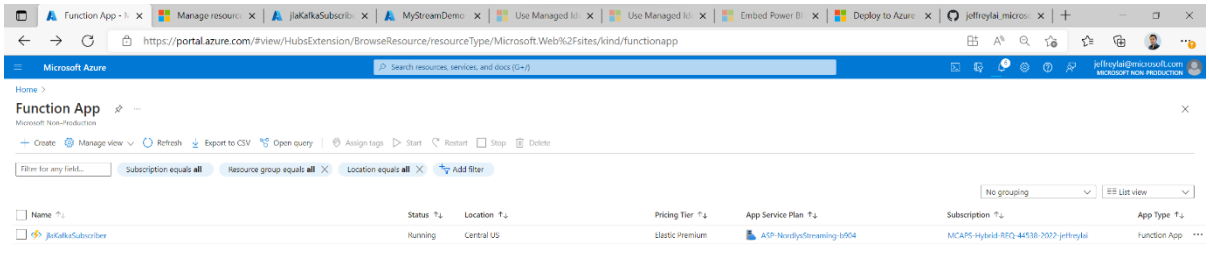


10. Repeat step 8 and 9 for the other Event Hub that was created (“jlaEventHubCloneSource”).

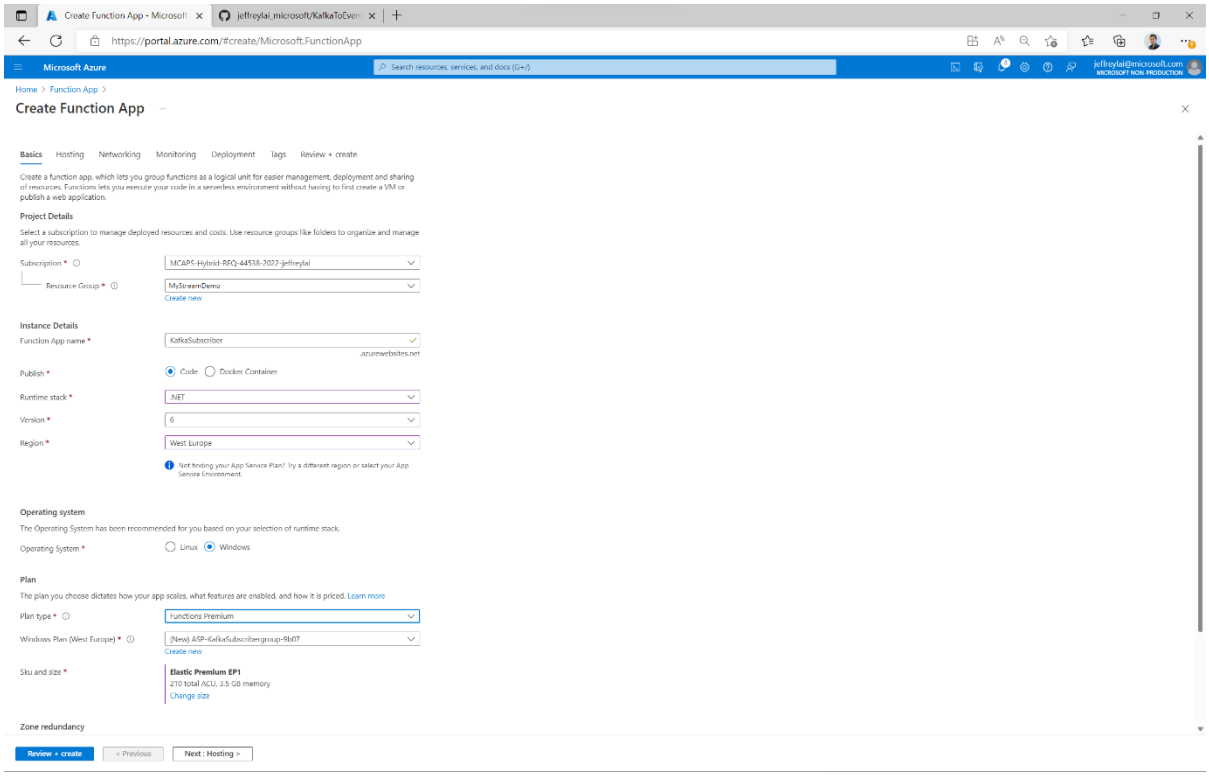
Creating Azure Function

Background: In this section we will create a Function that will automatically trigger when a message is received in a specific topic in Kafka Cluster. The function will then send the message to an Event Hub.

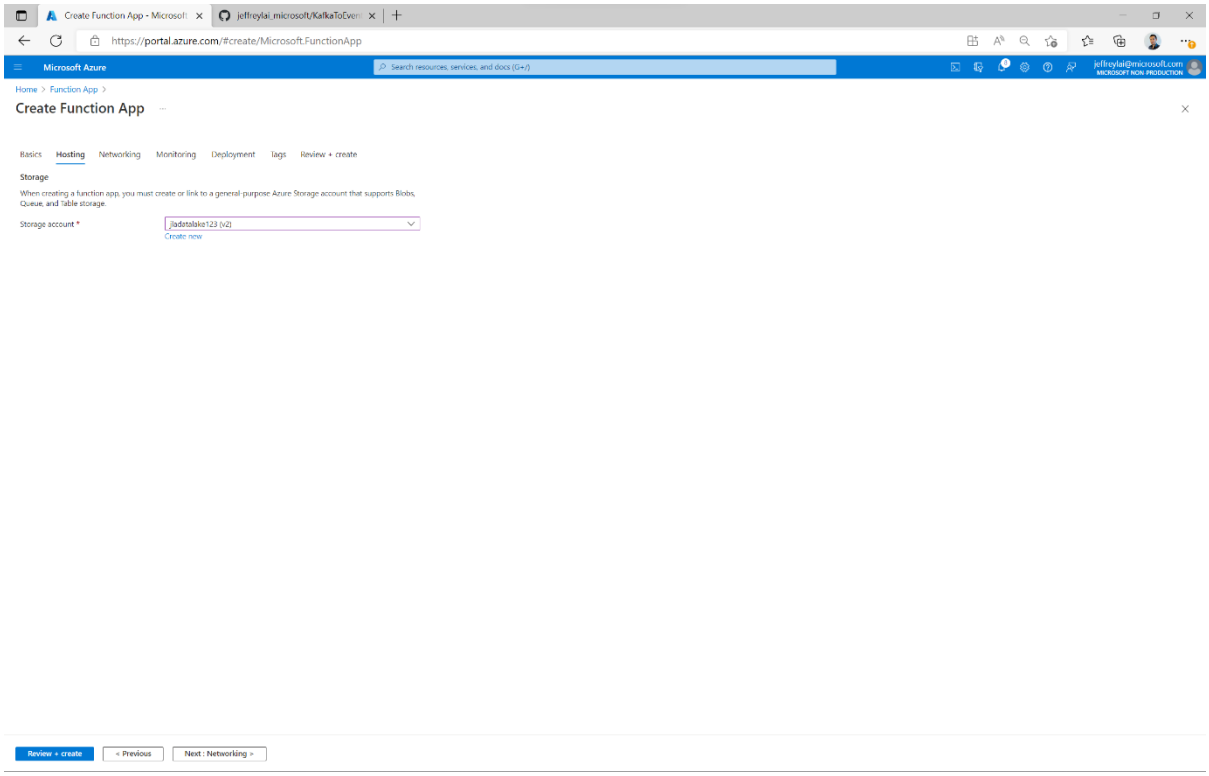
1. In the search bar, search for “Azure Function”.
2. Click “Create”



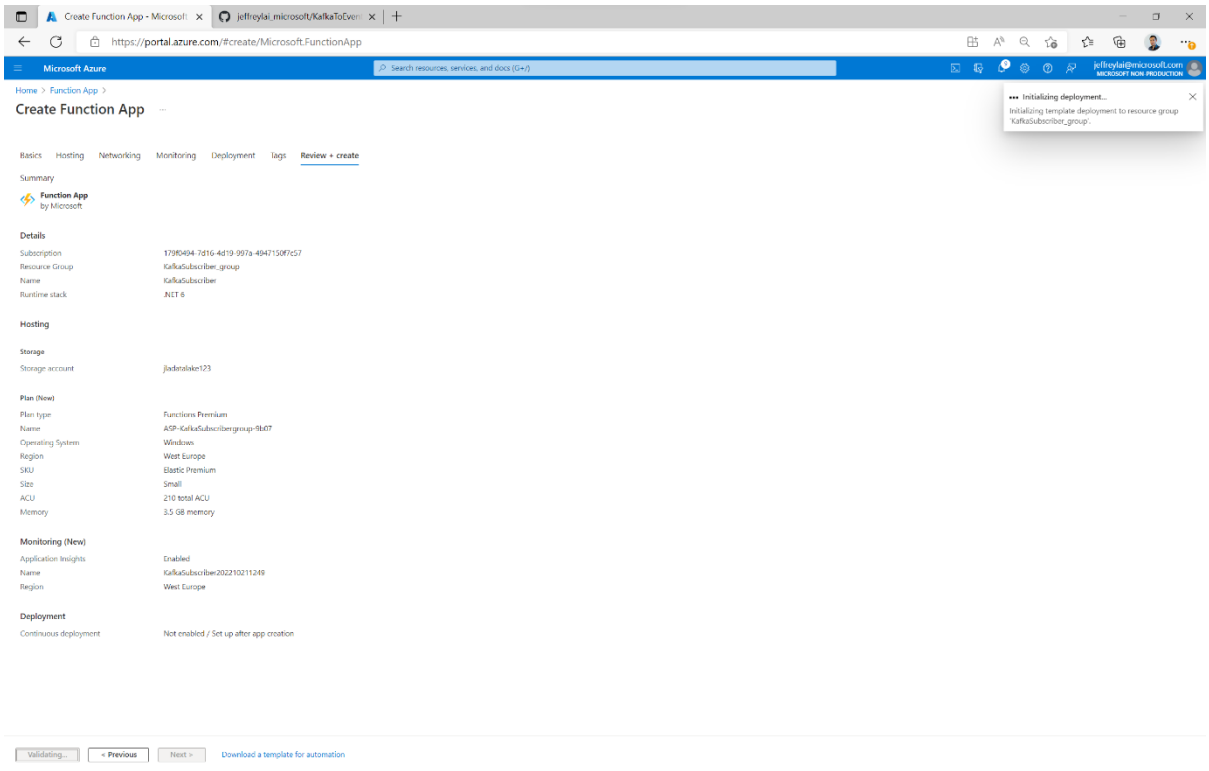
3. Fill out the details accordingly to the screenshot below. Please assign a unique name to the function app name.



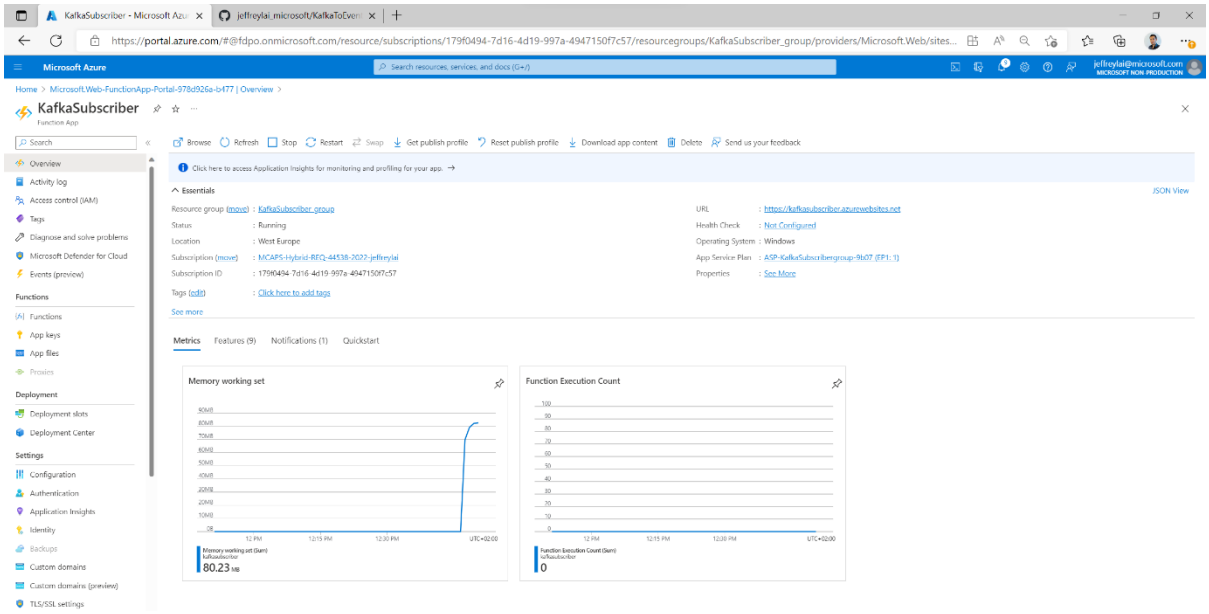
4. Go to “Hosting”, and choose the storage account (Data Lake) that was created by the arm template.



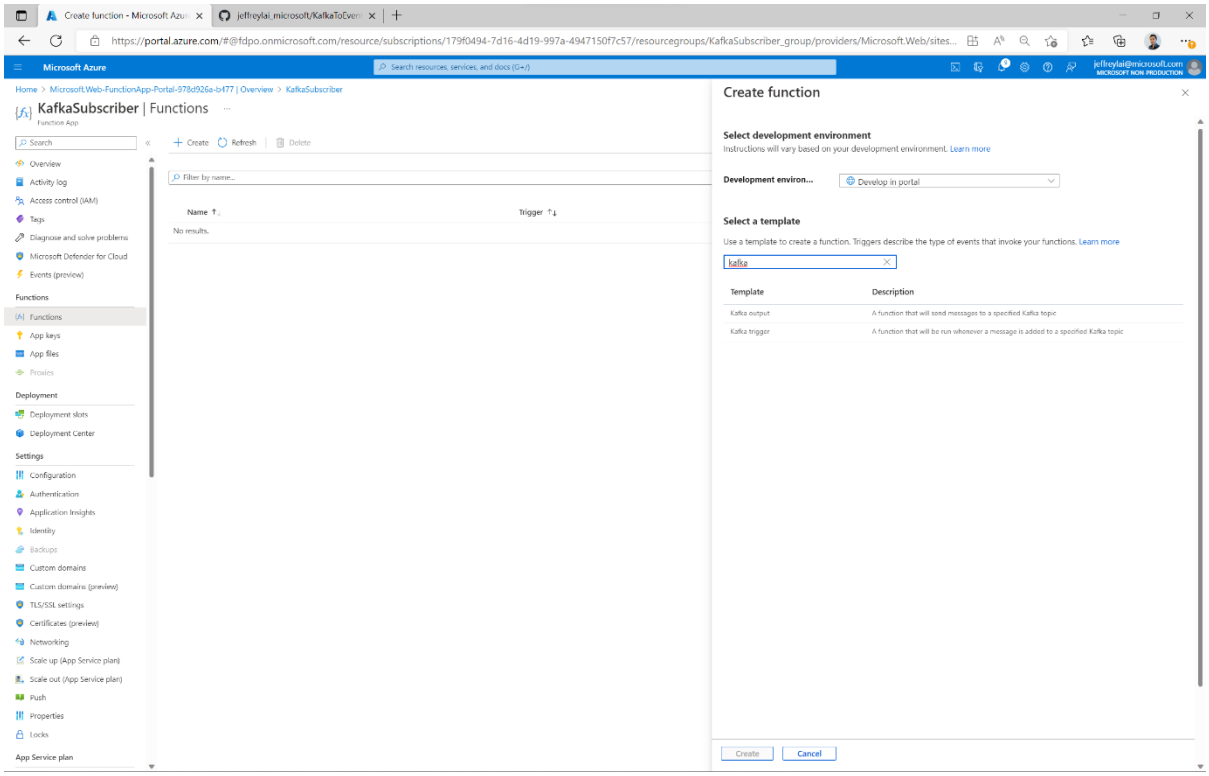
5. Click on “Review + Create”



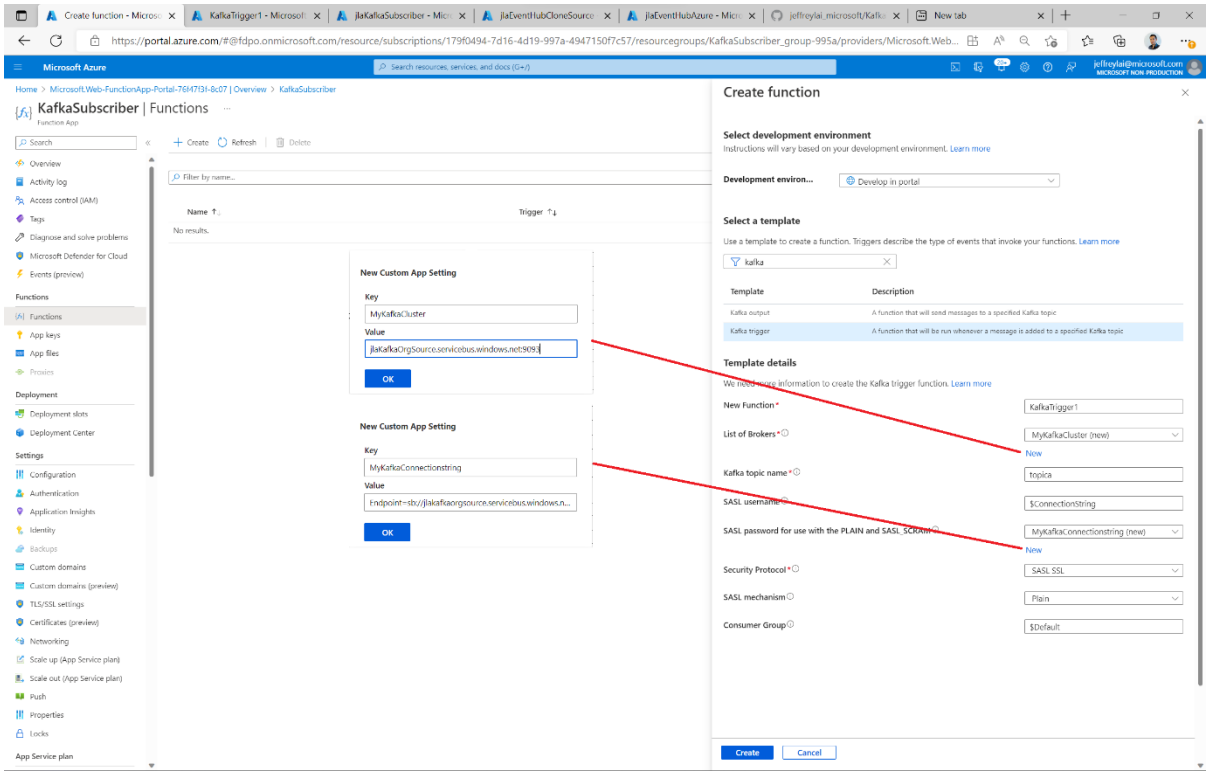
6. Once the resource is created, it will look like below:



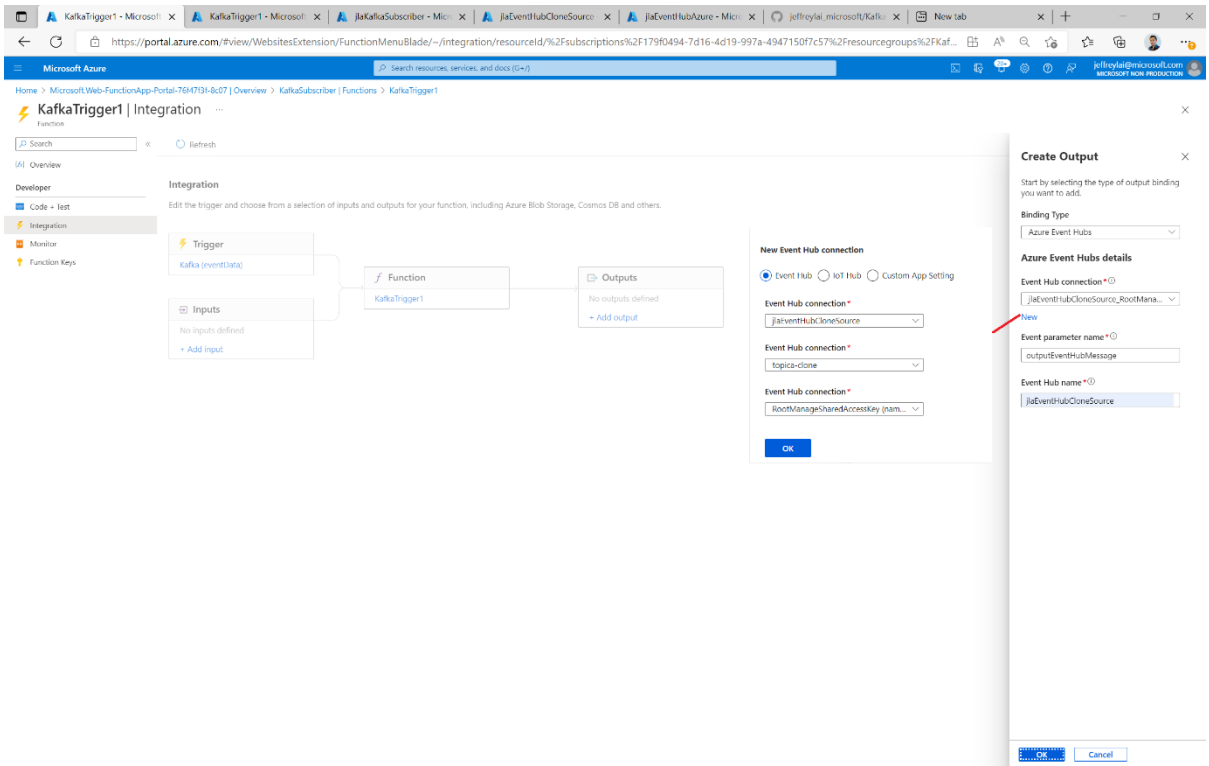
7. Click on “Functions” at the left panel, and search for “Kafka trigger”.



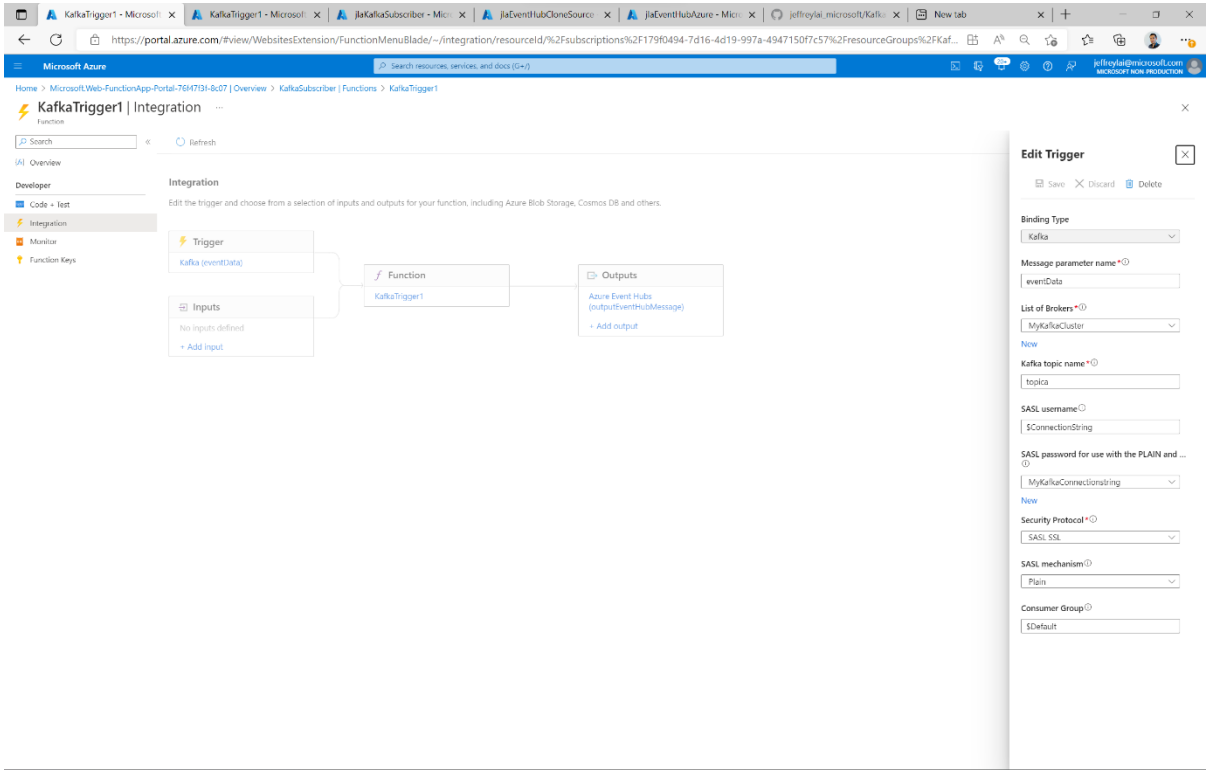
8. Fill out the details of the trigger template as below.
    a. For “List of Brokers” use the value that you pasted in notepad, and append “:9093”
    b. For “SASL password..” use the connection string that you pasted in notepad for *“JlaKafkaOrgSource” – e.g.* `Endpoint=sb://`*JlaKafkaOrgSource*`.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=/Ogx7qlAk6j8a6VNJAD3Ngf9+mwnPSaJ+RdSrJtwaNo=`



9. Create the Output for the function by clicking on “Integration”. *If you do not see this screenshot, go back to the function, and click on “Functions” on the left panel.*
   Fill out the details as the screenshot below to let the function write to the Event Hub.
   Note: It’s not recommended to give root permissions to any service you are connecting to. Always follow the least privilege principle when it comes to connecting devices. For more information visit: [Increase application security with the principle of least privilege - Microsoft Entra | Microsoft Learn]



10. Confirm that the “trigger” is configured with the details below :



11. On the left panel, click on “Code + Test”, and paste code below:

```
#r "Microsoft.Azure.WebJobs.Extensions.Kafka"

using System;
using System.Text;
using Microsoft.Azure.WebJobs.Extensions.Kafka;

public static void Run(KafkaEventData<string> eventData, ILogger log, out string outputEventHubMessage)
{
    //remember to add "out string outputEventHubMessage" as parameter for Run in the line above

    //Logs the message received from kafka
    log.LogInformation($"C# Kafka trigger function processed a message: {eventData.Value}");
    //Sends the message to event hub.
    outputEventHubMessage = eventData.Value;
}
```

12. Congratulations, you have now created an Azure Function that consumes from a Kafka Cluster and Produces to an Event Hub. Let’s test it in the steps below.

Running the demo

The code for the C# application used in this lab is based on regular kafka consumer/producer code samples to emphasize that an EventHub can act as an alternative to a Kafka Cluster without any code configuration to existing kafka consumer/producers.

Links to the original kafka consumer/producer are found below:

- https://www.thecodebuzz.com/apache-kafka-net-client-producer-consumer-csharp-confluent-examples-i/
- https://www.thecodebuzz.com/apache-kafka-net-client-producer-consumer-csharp-confluent-examples-ii/

In this demo, additional code has been added to:

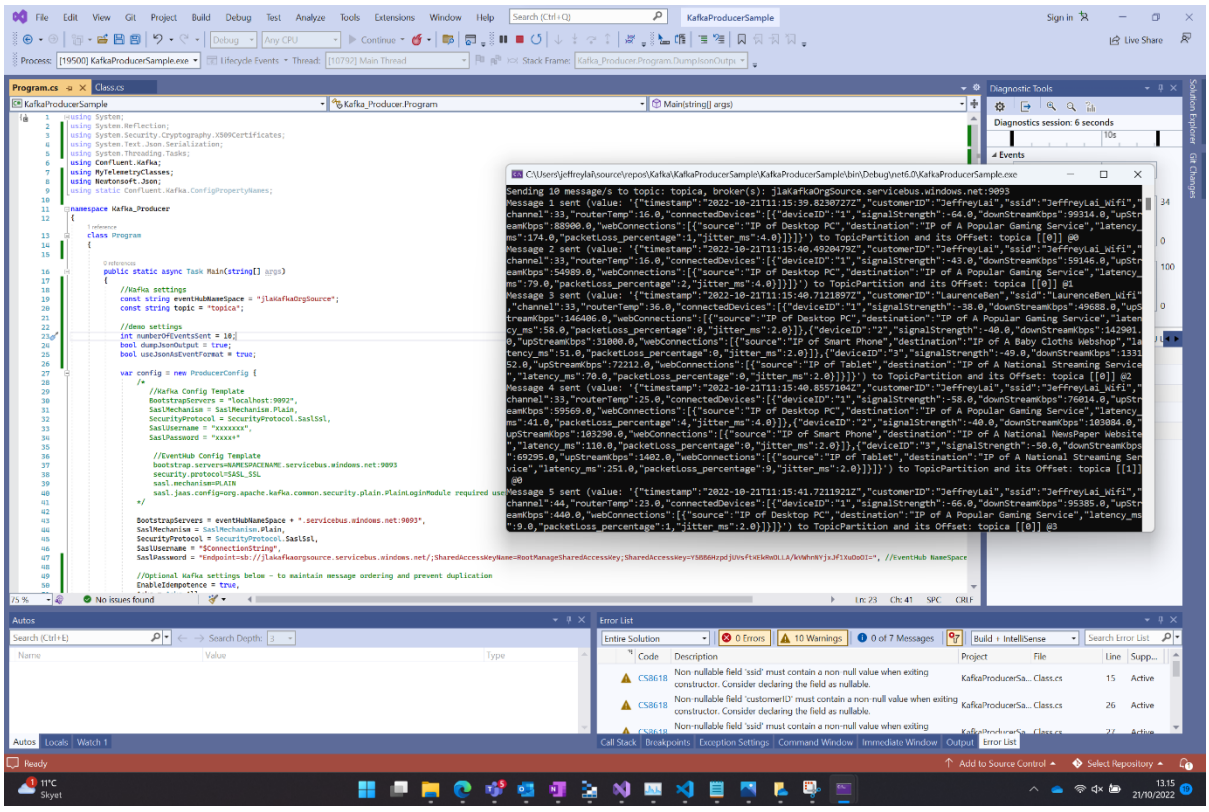
- Simulate randomized telemetry data instead of sending a specific message for the kafka producer.

1. Producing Events:

Following step will send events to the Kafka Cluster/EventHub Namespace to the topic/eventHub: "topica".

Open the C# project: "KafkaProducerSample", and configure the following variables in the code accordingly. Needs to be those reflecting the EventHub: "*JlaKafkaOrgSource*":

- `const string eventHubNameSpace = "jlaKafkaOrgSource";`
- `SaslPassword = "Endpoint=sb://jlakafkaorgsource.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=Y5BB6HzpdjUVsftKEkRwDLLA/kVWhnNYjxJf1XuOoOI=", //EventHub NameSpace Key1`
- Build and execute the console app

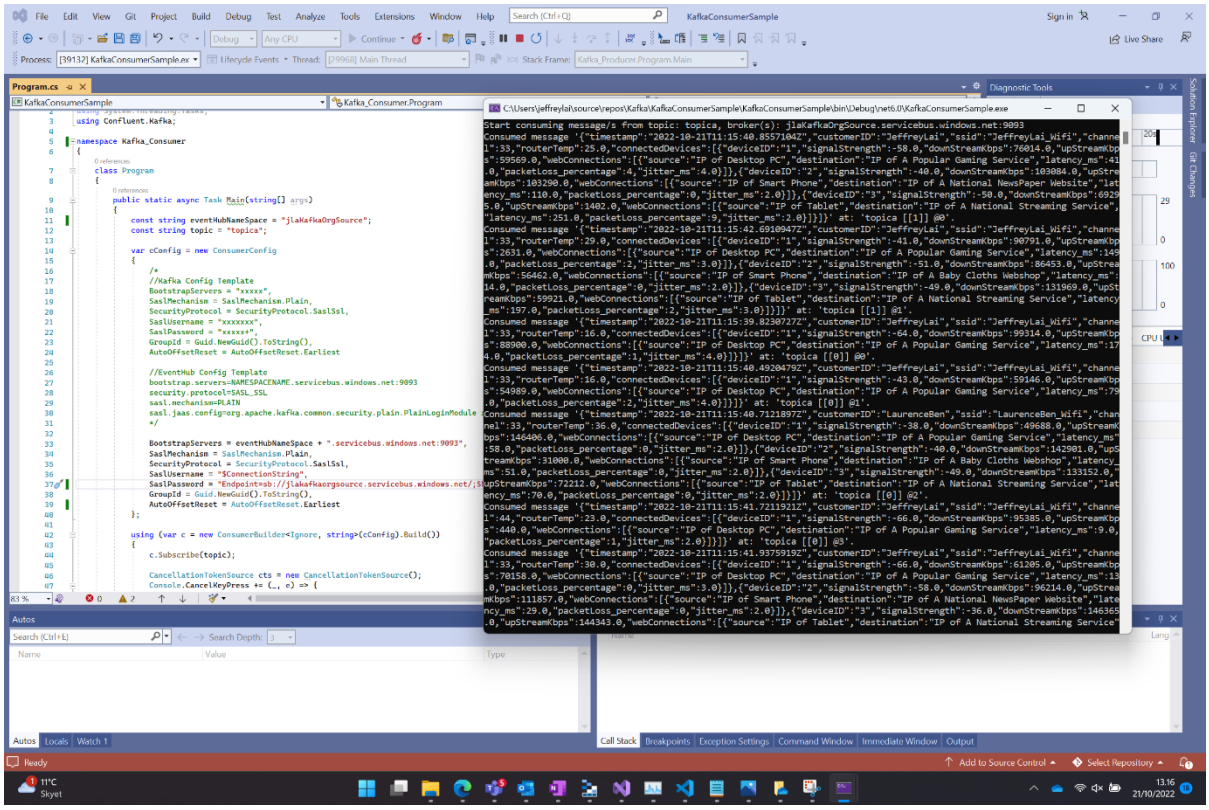


2. Consuming and verifying the events.

The following step will read all the events captured in the topic/eventhub: "topica".

Open the C# project: "KafkaConsumerSample", and configure the following variables in the code accordingly. Needs to be those reflecting the EventHub: "*JlaKafkaOrgSource*":

- `const string eventHubNameSpace = "jlaKafkaOrgSource";`
- `SaslPassword = "Endpoint=sb://jlakafkaorgsource.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=Y5BB6HzpdjUVsftKEkRwDLLA/kVWhnNYjxJf1XuOoOI=", //EventHub NameSpace Key1`
- Build and execute the console app

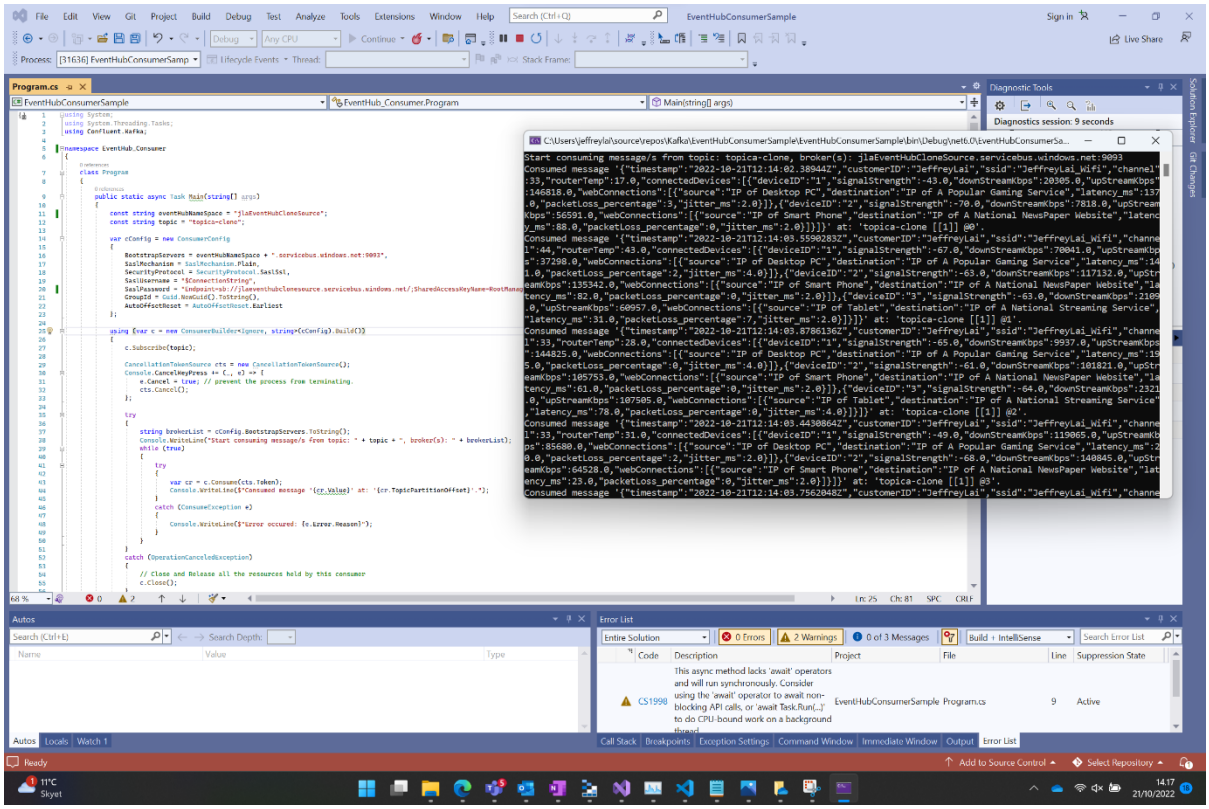


3. Consuming and verifying the events on *EventHubConsumerSample*.

The following step will read all the events captured in the topic/eventhub: "topica-clone" that was read and produced by the Azure Function.

Open the C# project: "EventHubConsumerSample", and configure the following variables in the code accordingly. Needs to be those reflecting the EventHub: "*jlaEventHubCloneSource*":

- `const string eventHubNameSpace = " jlaEventHubCloneSource";`
- `SaslPassword = "Endpoint=sb://jlaeventhubclonesource.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=azHBQNu71yJV0ujCppWGbByZ0bH6d+a+16GbQ4+8yh8=", //EventHub NameSpace Key1`
- Build and execute the console app



Congratulations! You have now made a solution that entails:

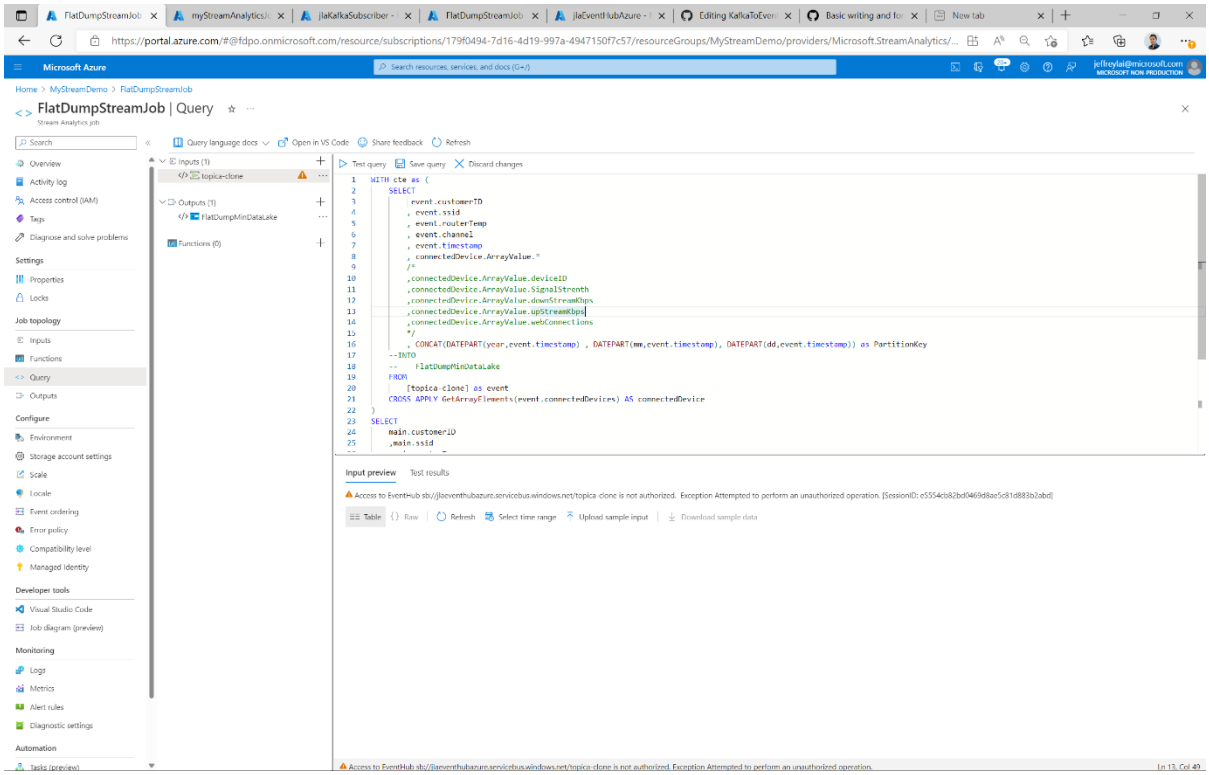
- A Kafka Producer that sends events to a topic in a Kafka Cluster/EventHub with Kafka Support using Kafka protocols
- A Kafka Consumer that subscribes to a topic in a Kafka Cluster/EventHub with Kafka Support using Kafka protocols
- An Azure Function that subscribes to a topic in a Kafka Cluster using Kafka protocols, and sends the event to an Event Hub.
- A Kafka Consumer that subscribes to an Event Hub to verify our data replication.

Optional Step1:

Description: The following step will finish a predeployed Stream Analytics job that will gather the events in real time from an EventHub and then sink it as a .csv file to blob storage (data lake) for later analysis. This job can easily be modified to sink to other data sources such as tables in a Azure SQL Database.

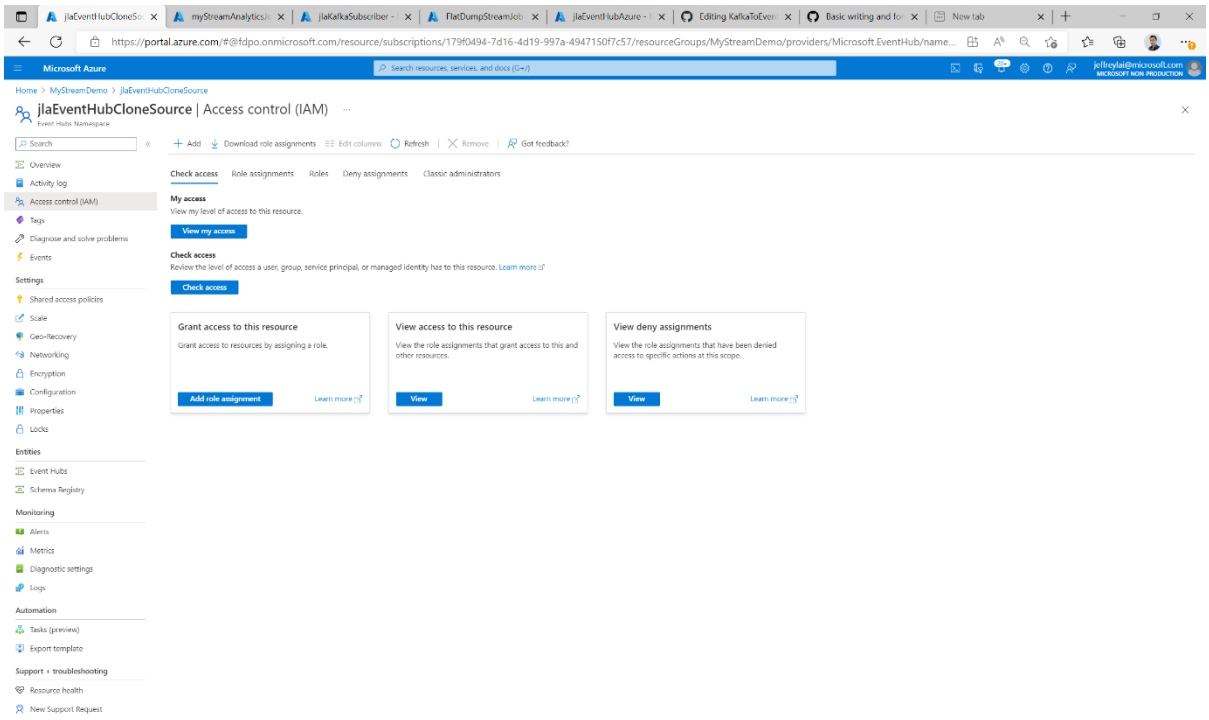
1. In the search bar, search for “Stream Analytics Jobs”, and click on the Job: “FlatDumpStreamJob”.
2. Click on “Query” on the left panel, and paste the following code:

```
WITH CTE AS (SELECT
       event.customerID
      , event.ssid
      , event.routerTemp
      , event.channel
      , event.timestamp
      , connectedDevice.ArrayValue.*
      /*
      ,connectedDevice.ArrayValue.deviceID
      ,connectedDevice.ArrayValue.SignalStrenth
      ,connectedDevice.ArrayValue.downStreamKbps
      ,connectedDevice.ArrayValue.upStreamKbps
      ,connectedDevice.ArrayValue.webConnections
      */
      , CONCAT(DATEPART(year,event.timestamp) , DATEPART(mm,event.timestamp), DATEPART(dd,event.timestamp)) as PartitionKey
   --INTO
   --    FlatDumpMinDataLake
   FROM
      [topica-clone] as event
   CROSS APPLY GetArrayElements(event.connectedDevices) AS connectedDevice
)
SELECT
   main.customerID
   ,main.ssid
   ,main.routerTemp
   ,main.channel
   ,main.timestamp
   ,main.deviceID
   ,main.SignalStrenth
   ,main.downStreamKbps
   ,main.upStreamKbps
   ,main.PartitionKey
   ,webConnection.ArrayValue.*
INTO
   FlatDumpMinDataLake
FROM cte as main
CROSS APPLY GetArrayElements(main.webConnections) AS webConnection
```

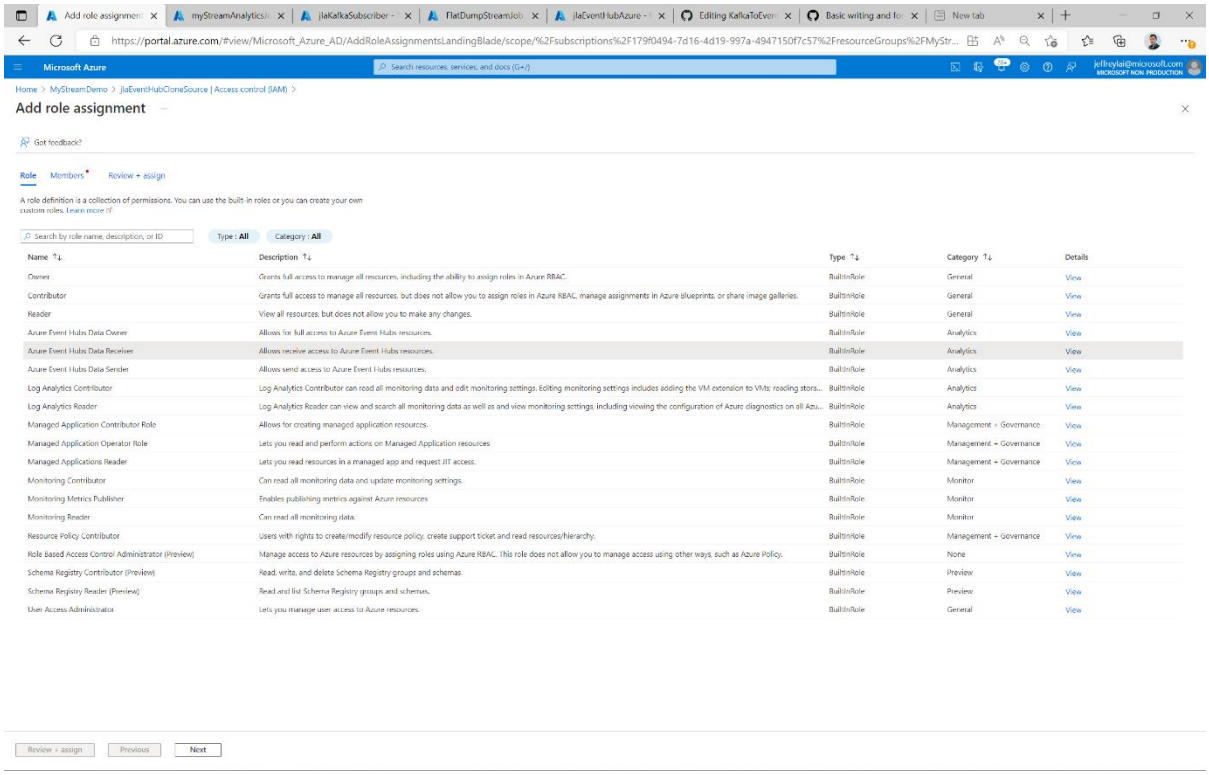


Note: If the orange triangle appears next to a source (input/output) it means that stream analytics doesn't have the right permissions to access the resource. Do as below:

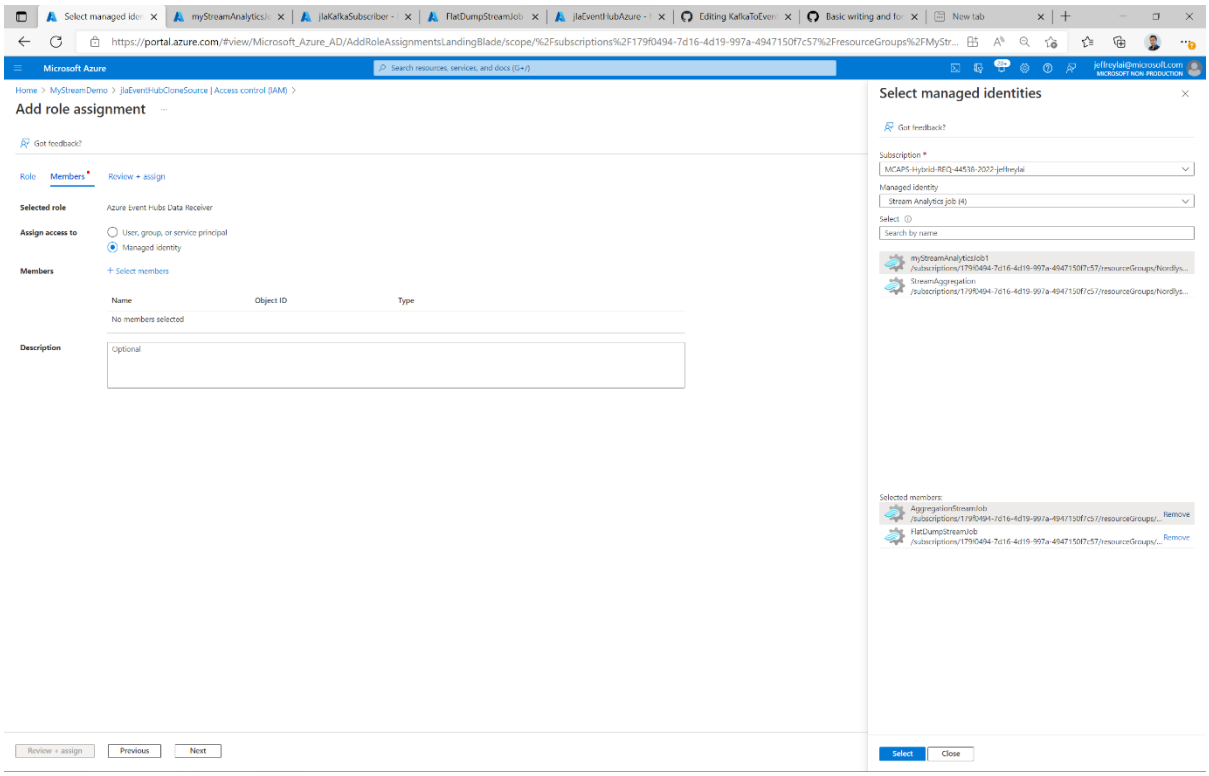
3. Go to the EventHub (jlaEventHubCloneSource), and click on "Access control (IAM), and then "Add role Assignment".



4. Choose e.g. Azure Event Hub Data Receiver to allow Stream Analytics read access.

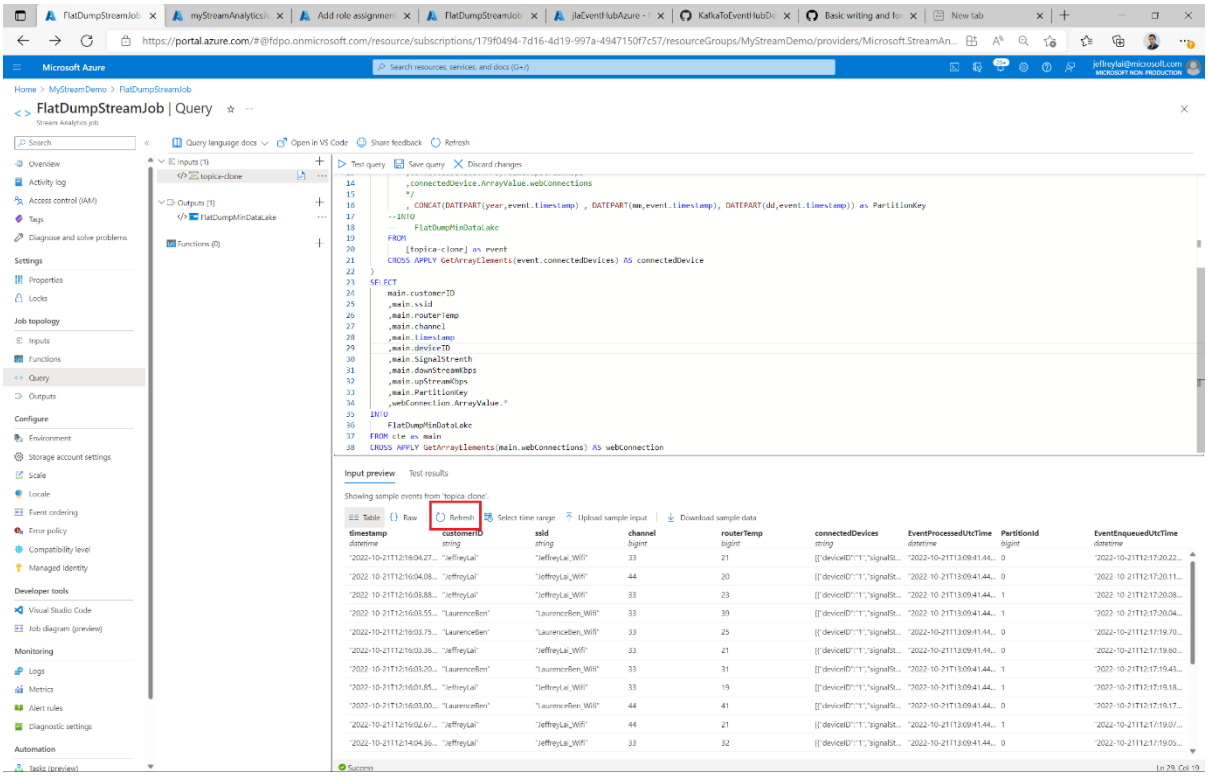


5. Click next and choose “Managed Identity” and click “Select members”. At the panel to the right, choose the Stream Analytics jobs that will get this permission.

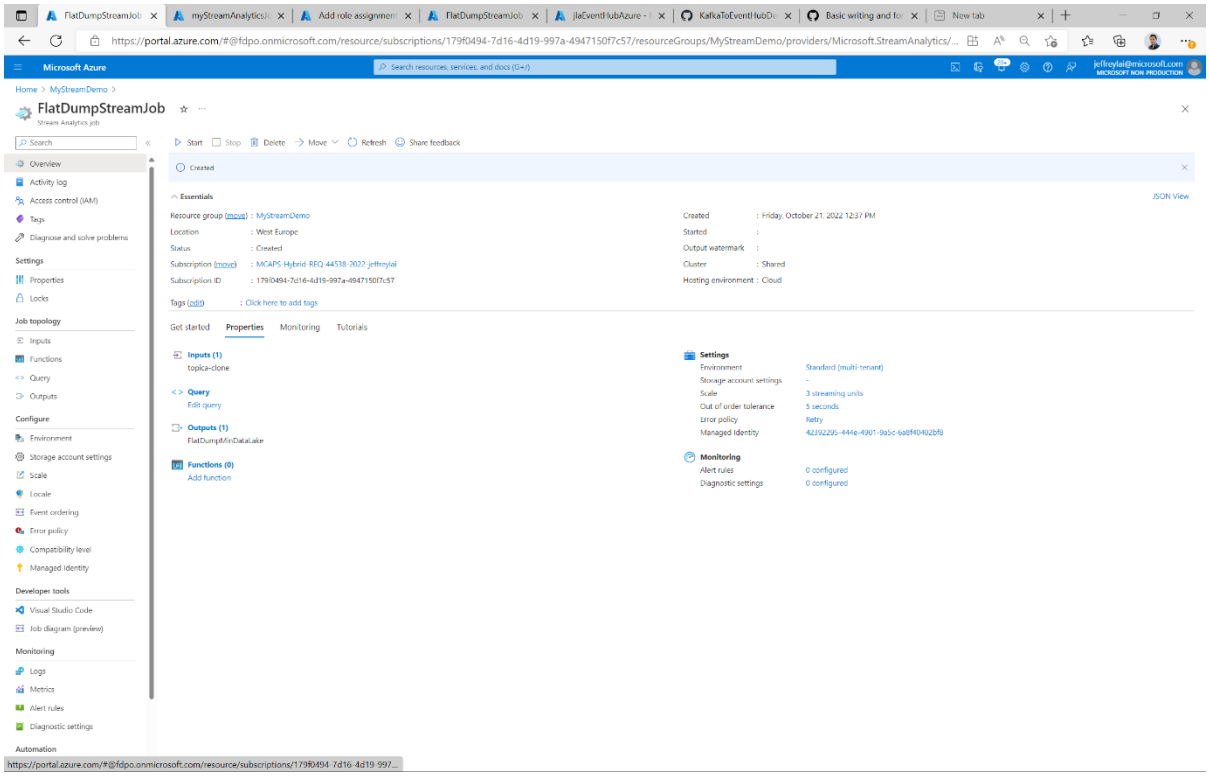


6. Go back to the stream analytics job, and refresh the input preview for the input source. It should now have the appropriate permissions to connect to the event hub. Also try to “test

query” to validate the code.

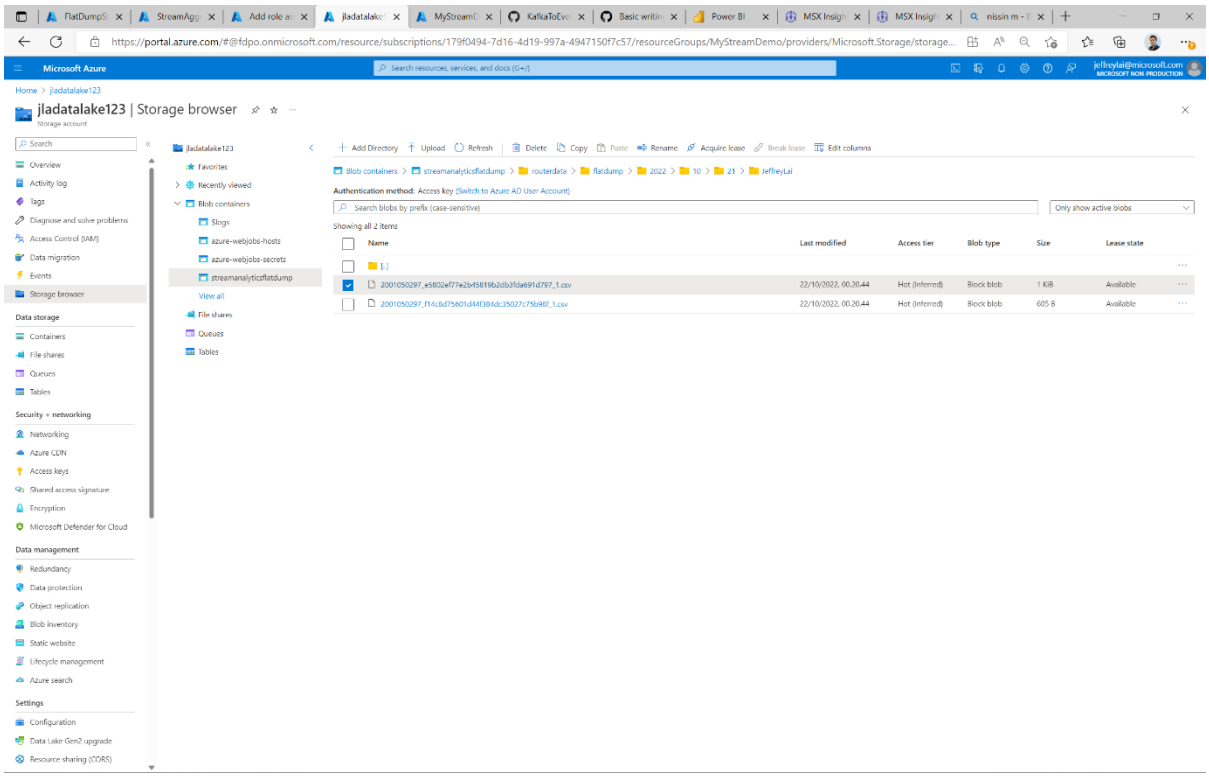


7. Go to the “Overview” for Stream Analytics job, and click “Start”.

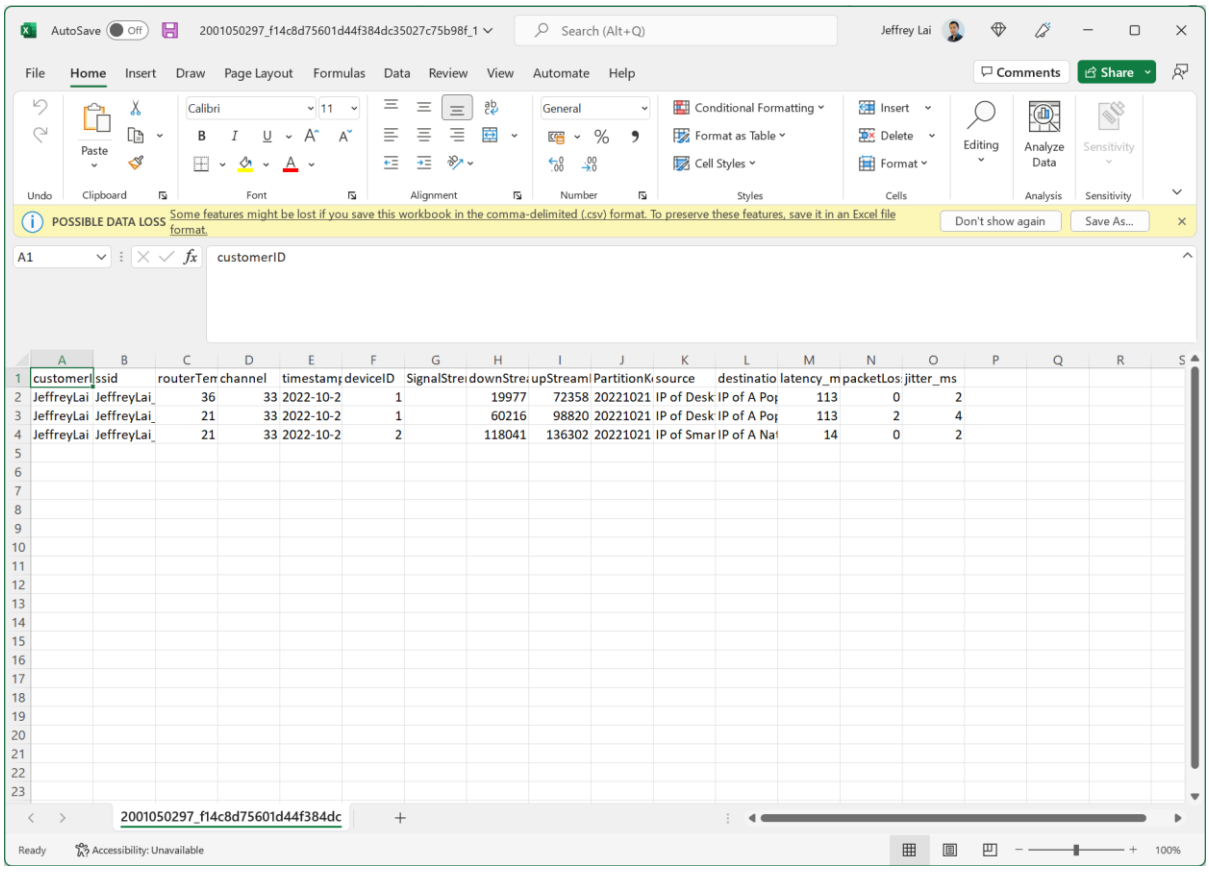


8. Execute the console app: KafkaProducerSample to produce some events to the Event Hub
9. Verify the output of the Stream Analytics job that is designed to produce flatfiles in the deployed storage account (data lake)
    a. This can be done by searching for Storage account, and then clicking on the specified storage account. On the left panel click on “Storage browser, and locate the blob

container “streamanalyticsflatdump”. Then click through the created folders to find the csv files:



Example of the flat files:

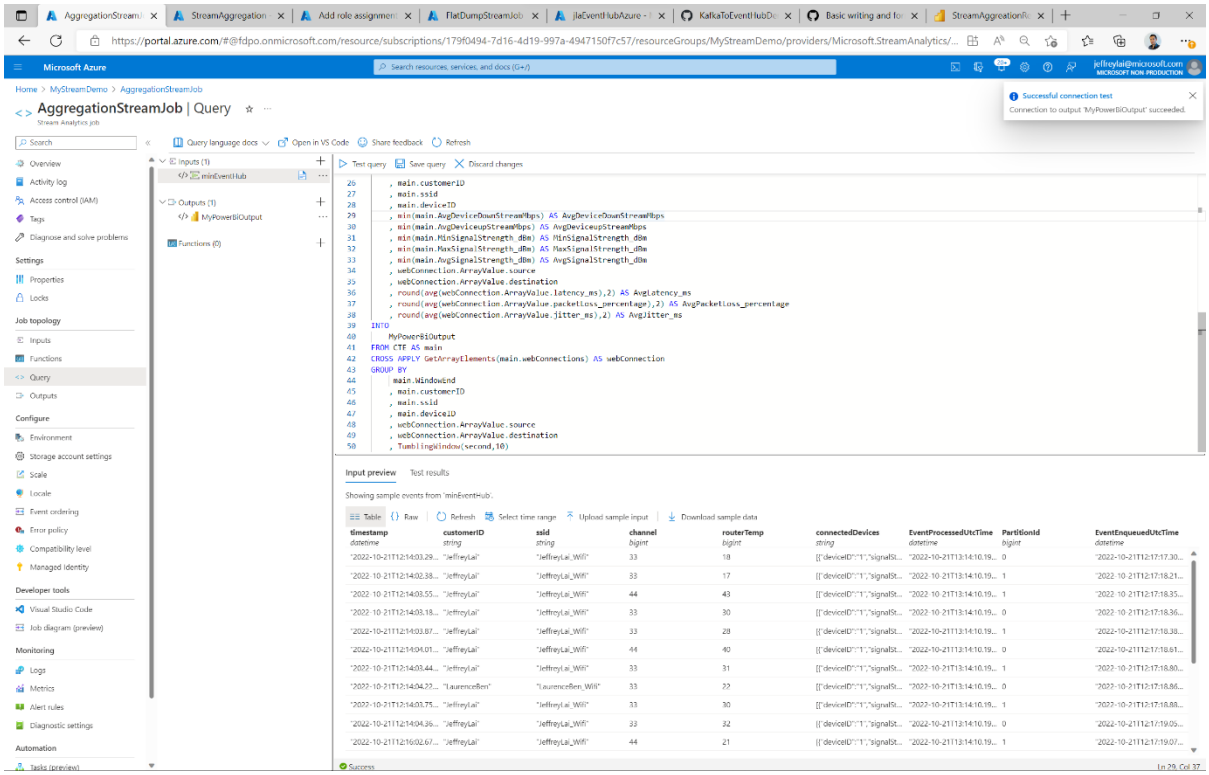


Optional step2:

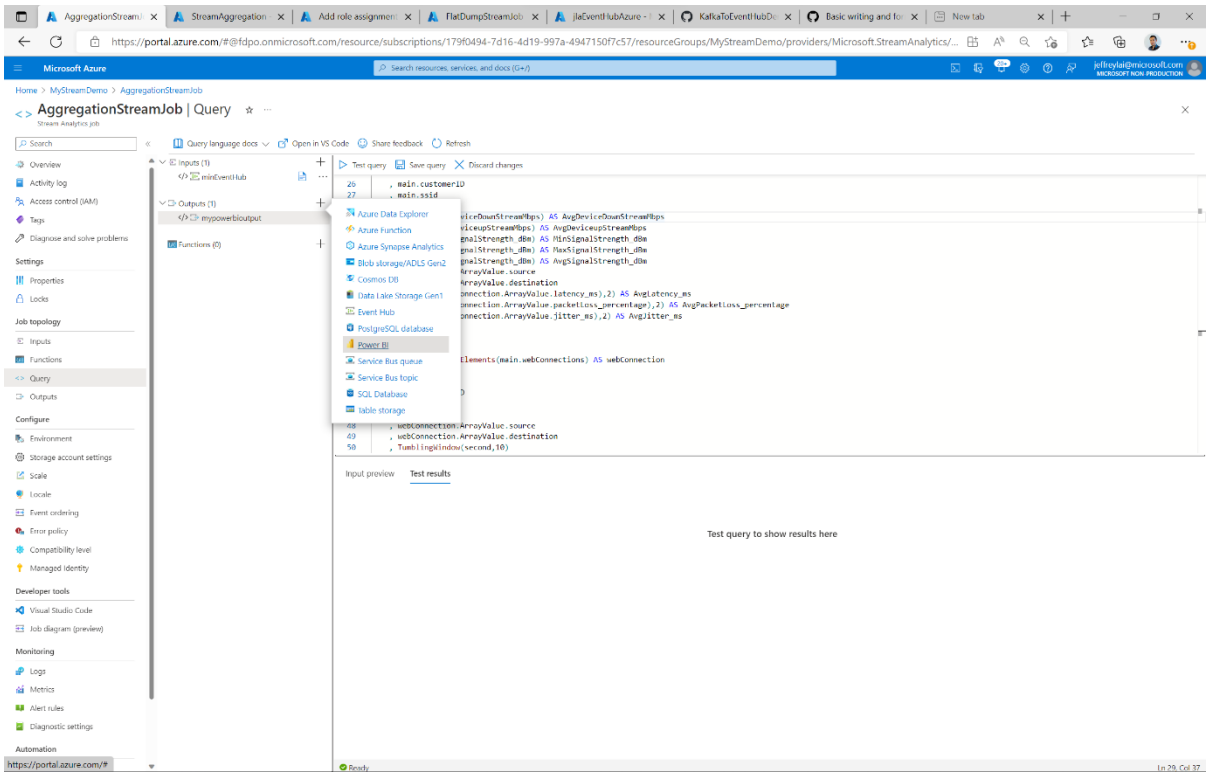
Description: The following step will finish a predeployed Stream Analytics job that will gather the events in real time from an EventHub and stream it to a dataset in PowerBi.

1. In the search bar, search for “Stream Analytics Jobs”, and click on the Job: “StreamAggreation”.
2. Click on “Query” on the left panel, and paste the following code:

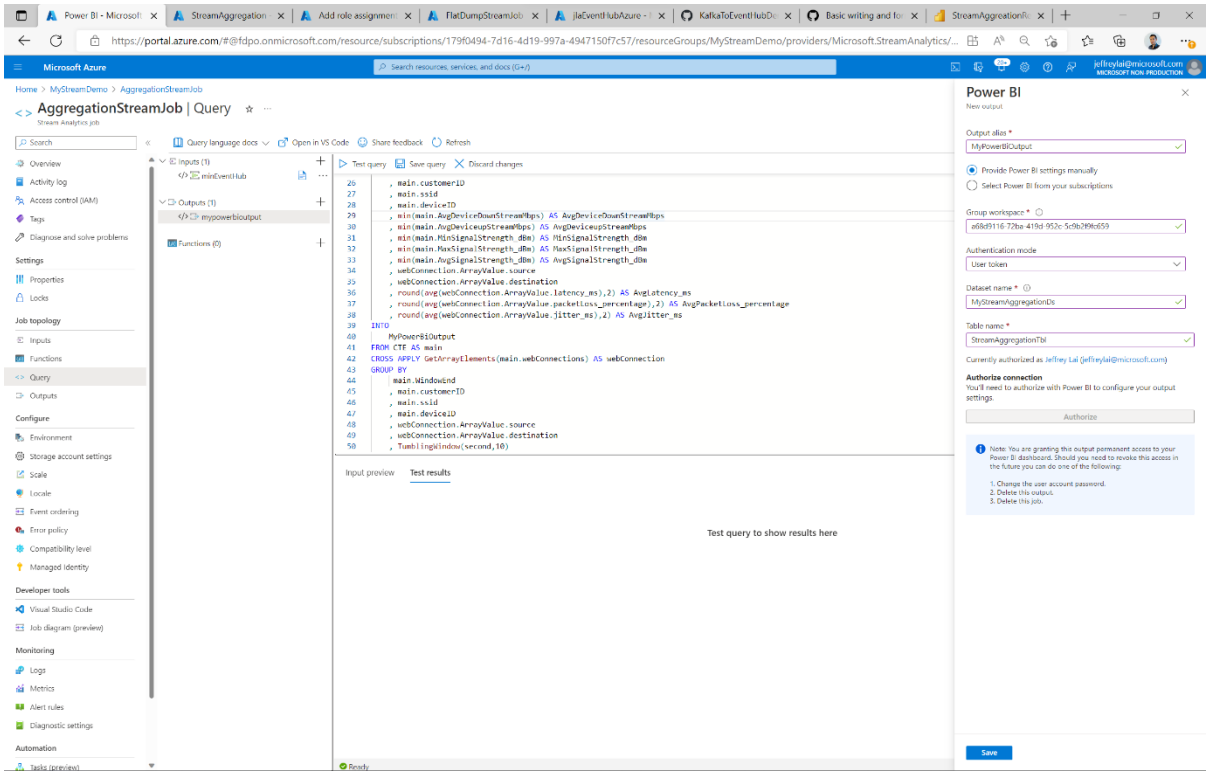
```
  WITH CTE AS (SELECT
     System.Timestamp() AS WindowEnd
     , event.customerID
     , event.ssid
     , connectedDevice.ArrayValue.deviceID
     , round(avg(connectedDevice.ArrayValue.downStreamKbps)/1024,2) AS AvgDeviceDownStreamMbps
     , round(avg(connectedDevice.ArrayValue.upStreamKbps)/1024,2) AS AvgDeviceupStreamMbps
     , round(min(connectedDevice.ArrayValue.signalStrength),2) AS MinSignalStrength_dBm
     , round(max(connectedDevice.ArrayValue.signalStrength),2) AS MaxSignalStrength_dBm
     , round(avg(connectedDevice.ArrayValue.signalStrength),2) AS AvgSignalStrength_dBm
     , connectedDevice.ArrayValue.webConnections

  FROM
     [minEventHub] AS event TIMESTAMP BY event.timestamp
  CROSS APPLY GetArrayElements(event.connectedDevices) AS connectedDevice
  GROUP BY
     event.customerID
     , event.ssid
     , connectedDevice.ArrayValue.deviceID
     , connectedDevice.ArrayValue.webConnections
     , TumblingWindow(second,10)
)
SELECT
   main.WindowEnd
   , main.customerID
   , main.ssid
   , main.deviceID
   , min(main.AvgDeviceDownStreamMbps) AS AvgDeviceDownStreamMbps
   , min(main.AvgDeviceupStreamMbps) AS AvgDeviceupStreamMbps
   , min(main.MinSignalStrength_dBm) AS MinSignalStrength_dBm
   , min(main.MaxSignalStrength_dBm) AS MaxSignalStrength_dBm
   , min(main.AvgSignalStrength_dBm) AS AvgSignalStrength_dBm
   , webConnection.ArrayValue.source
   , webConnection.ArrayValue.destination
   , round(avg(webConnection.ArrayValue.latency_ms),2) AS AvgLatency_ms
   , round(avg(webConnection.ArrayValue.packetLoss_percentage),2) AS AvgPacketLoss_percentage
   , round(avg(webConnection.ArrayValue.jitter_ms),2) AS AvgJitter_ms
INTO
   MyPowerBiOutput
FROM CTE AS main
CROSS APPLY GetArrayElements(main.webConnections) AS webConnection
GROUP BY
    main.WindowEnd
   , main.customerID
   , main.ssid
   , main.deviceID
   , webConnection.ArrayValue.source
   , webConnection.ArrayValue.destination
   , TumblingWindow(second,10)
```



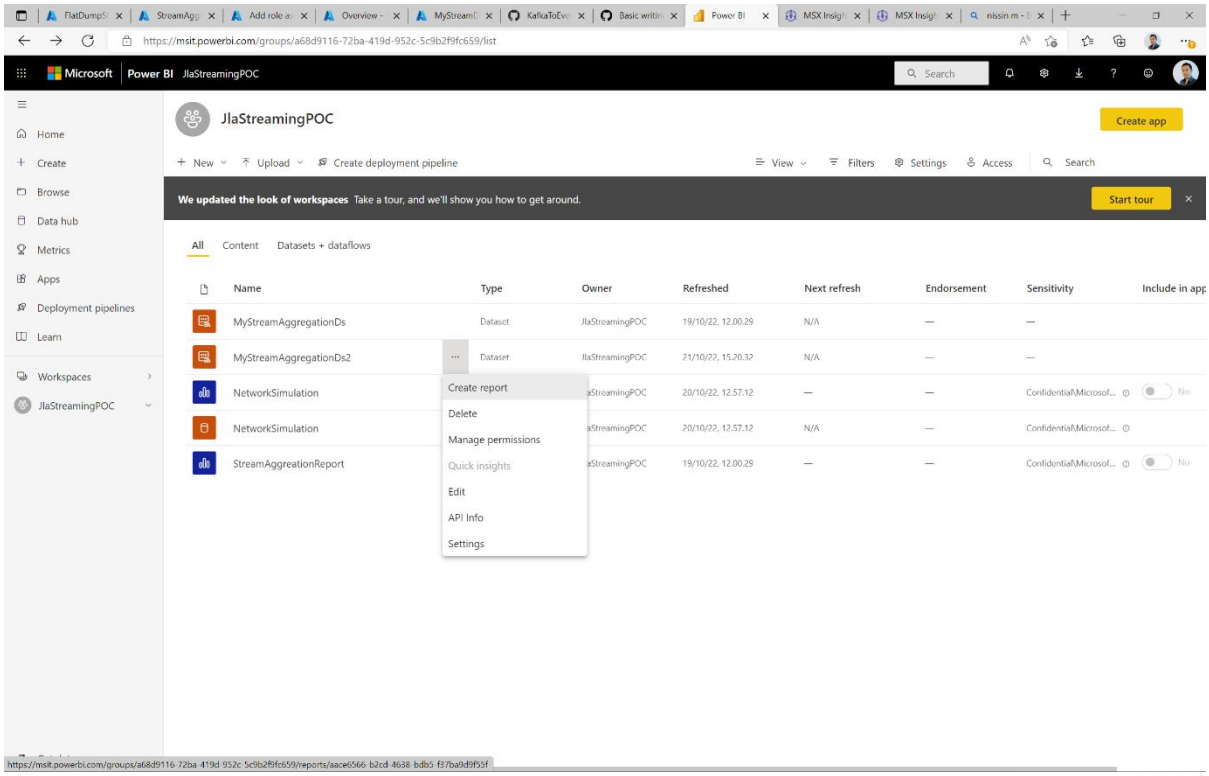
3. Configure the output by clicking in the “+” sign. Choose “Power Bi”



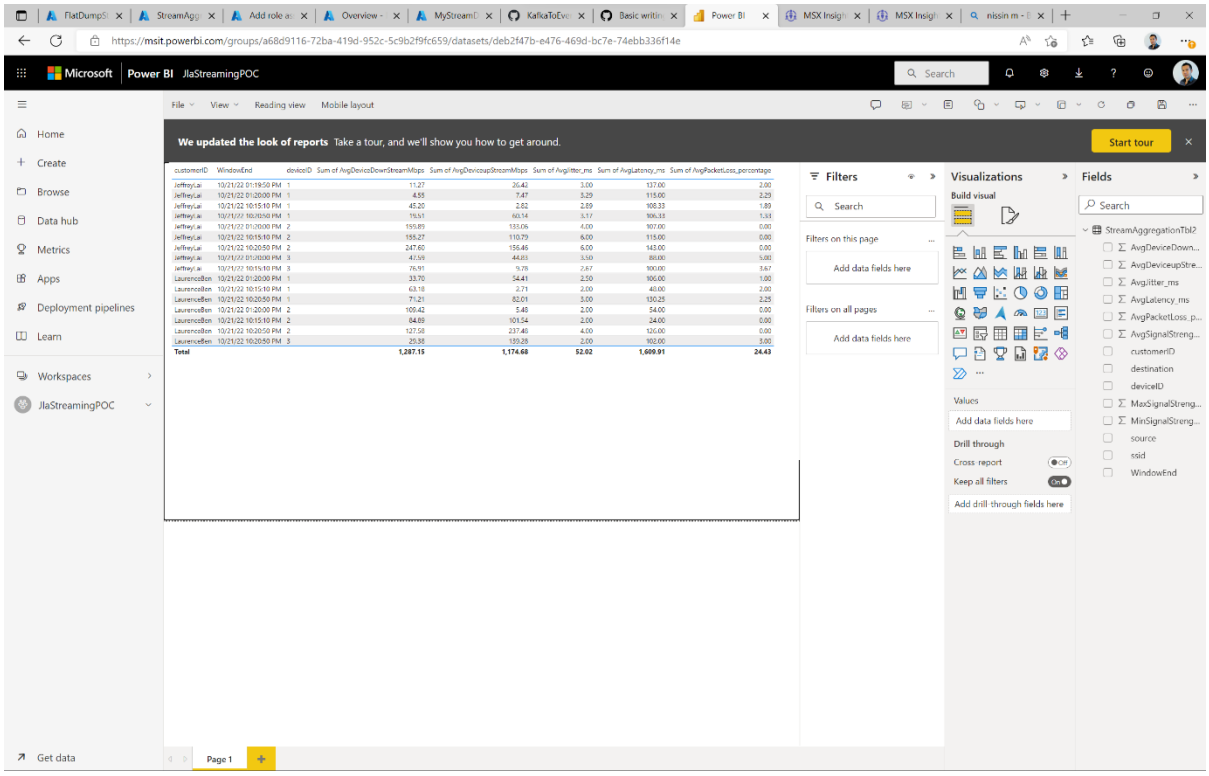
4. Configure the output to match the desired Workspace to publish to with the “Group workspace” GUID, and define names for dataset table that will be created for this stream. Finalize the process by clicking on “Authorize”. This will allow Stream Analytics to write to Power BI.



5. Execute the console app: KafkaProducerSample to produce some events to the Event Hub
6. Verify output of the job by going to your PowerBi workspace, and confirm a dataset has been created: “MyStreamAggregationDs”.



7. Click on “Create Report” on the dataset, and it will look like below:



8. Congratulations, you can now create dash board based on real time data.