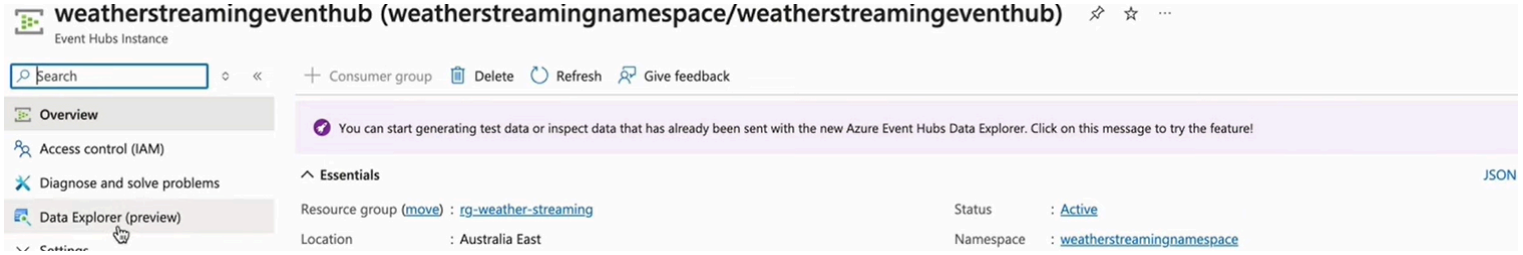
Data Ingestion:

1. Create and setup Event Hub: Azure **Event Hub** is a **real-time data streaming platform**. It ingests millions of events per second from sources like apps, IoT devices, or logs, and passes them to consumers like Databricks, Azure Stream Analytics, or Azure Functions. It's like a **message pipeline** for large-scale streaming data.

2. Create Cluster in databricks: Launch databricks. Create personal compute:: Configure it(databricks version, Node type)

3. Install event Hub libraries in cluster: go in your cluster: click libraries: select Pypi radio as its pypi library. Install it. Restart cluster

4. Sending test Event from DB to eventHUB: in your Azure event hub inspect this:



Notifies events.
Event Hubs Data Explorer is a feature within the Azure Portal that allows users to interact with data flowing through an Event Hub without writing any code. It provides a browser-based interface to manually send events to an Event Hub or view the messages that have been received. This tool is especially useful for testing, debugging, and verifying the setup of event producers and consumers. By using the Data Explorer, users can monitor the real-time flow of events, examine specific partitions, and check offsets or sequence numbers. It helps ensure that events are being correctly published and consumed, making it a valuable feature during development and troubleshooting.

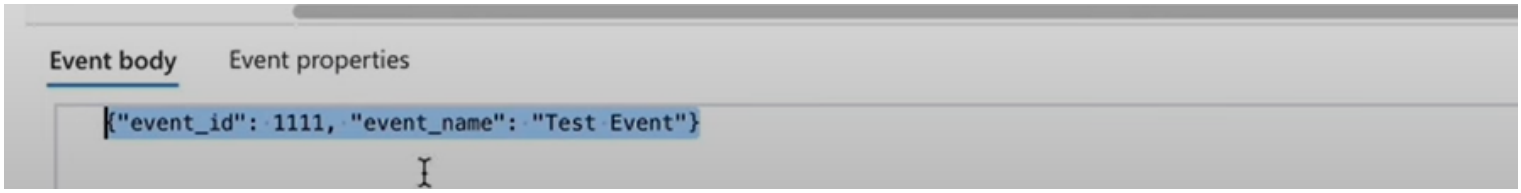
A. create notebook in databricks: attach cluster to it.

```
from azure.eventhub import EventHubProducerClient, EventData
import json

# Event Hub configuration
EVENT_HUB_CONNECTION_STRING = "XXXX"     – put key here
EVENT_HUB_NAME = "XXXX"  – put here

# Initialize the Event Hub producer
producer = EventHubProducerClient.from_connection_string(
    conn_str=EVENT_HUB_CONNECTION_STRING,
    eventhub_name=EVENT_HUB_NAME
)

# Function to send events to Event Hub
def send_event(event):
    event_data_batch = producer.create_batch()
    event_data_batch.add(EventData(json.dumps(event)))
    producer.send_batch(event_data_batch)

# Sample JSON event
event = {
    "event_id": 1111,
    "event_name": "Test Event",
}

# Send the event
```

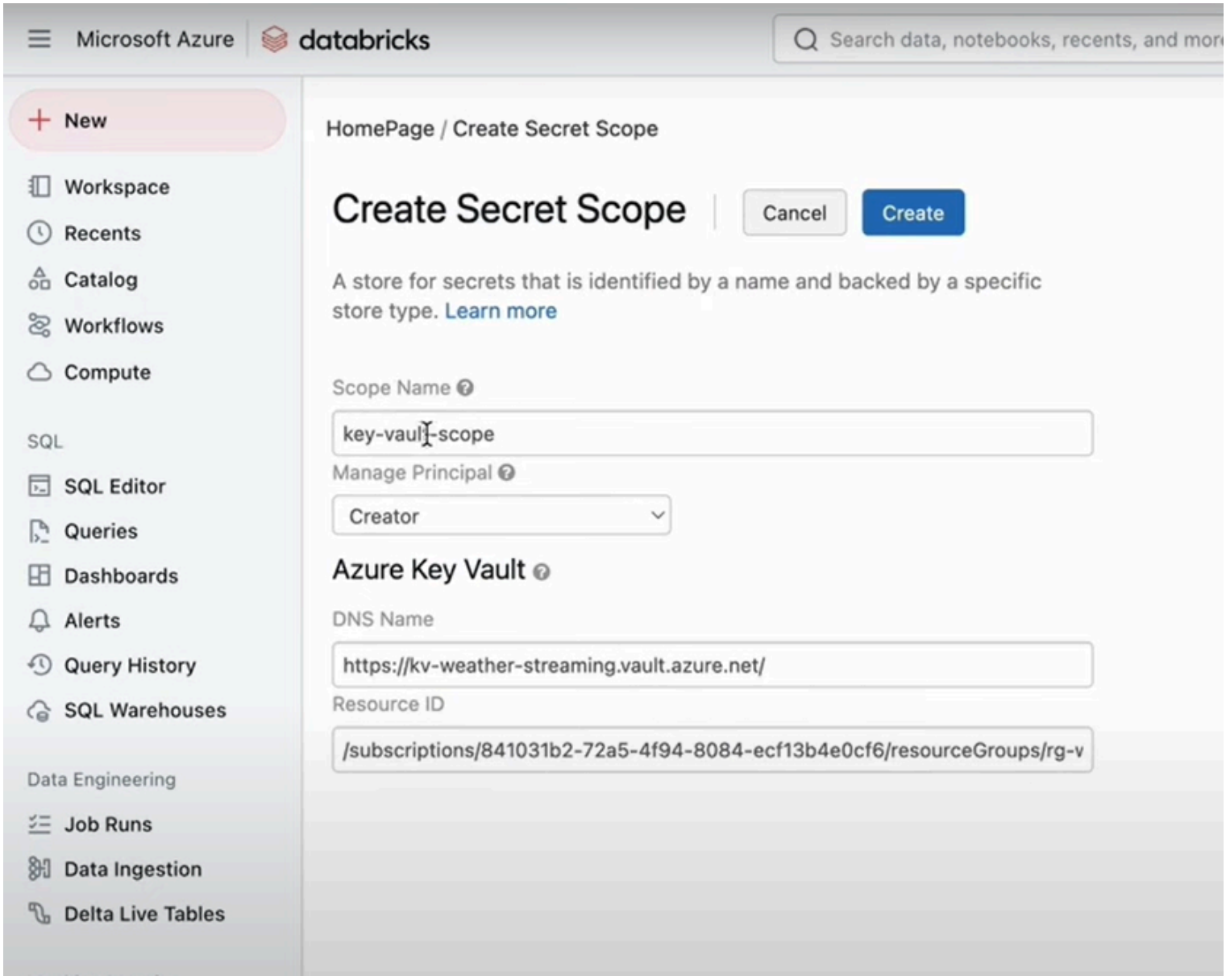
```
send_event(event)

# Close the producer
producer.close()
```

Run it.

B.
Go to data explorer in event hub:: you will see your event body.



5. Configuring keyvaults in Azure databricks so that in the above script you dont need to put key explicitly.
Go to url from databricks documentation for creating secrets.

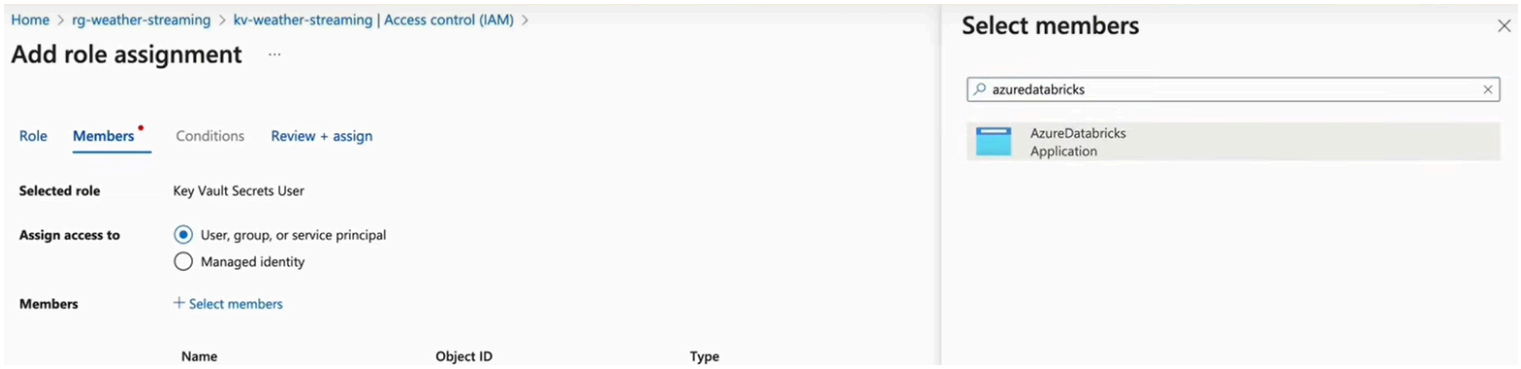


Update code now.

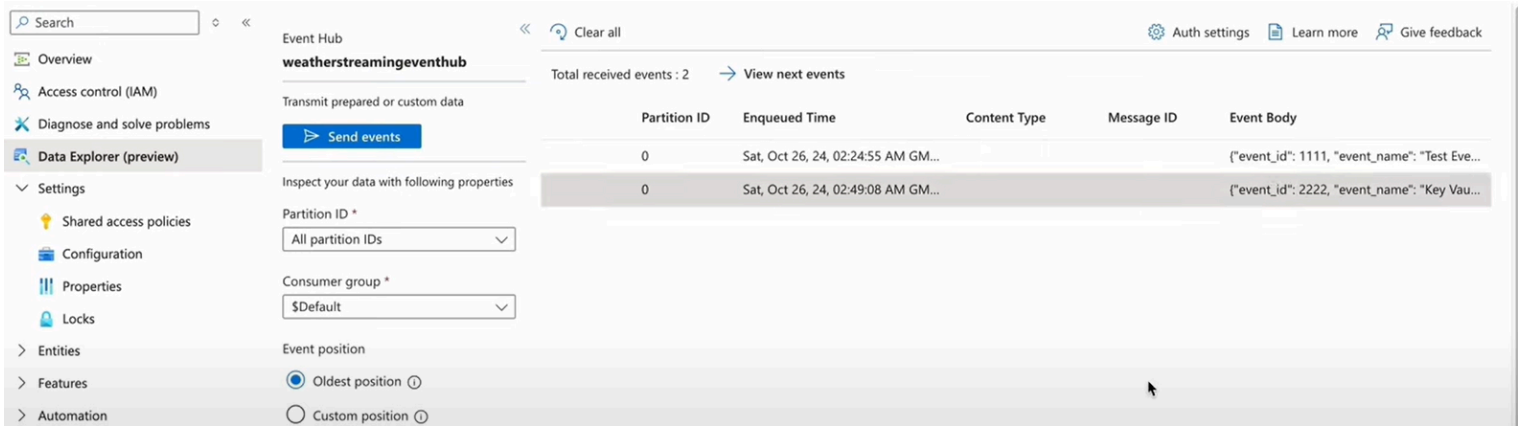
```
# Getting secret value from Key Vault
eventhub_connection_string = dbutils.secrets.get(scope="key-vault-scope", key="eventhub-connection-string")
```

Now provide access from azure keyvault: go to IAM in key vault: select key vault secret us

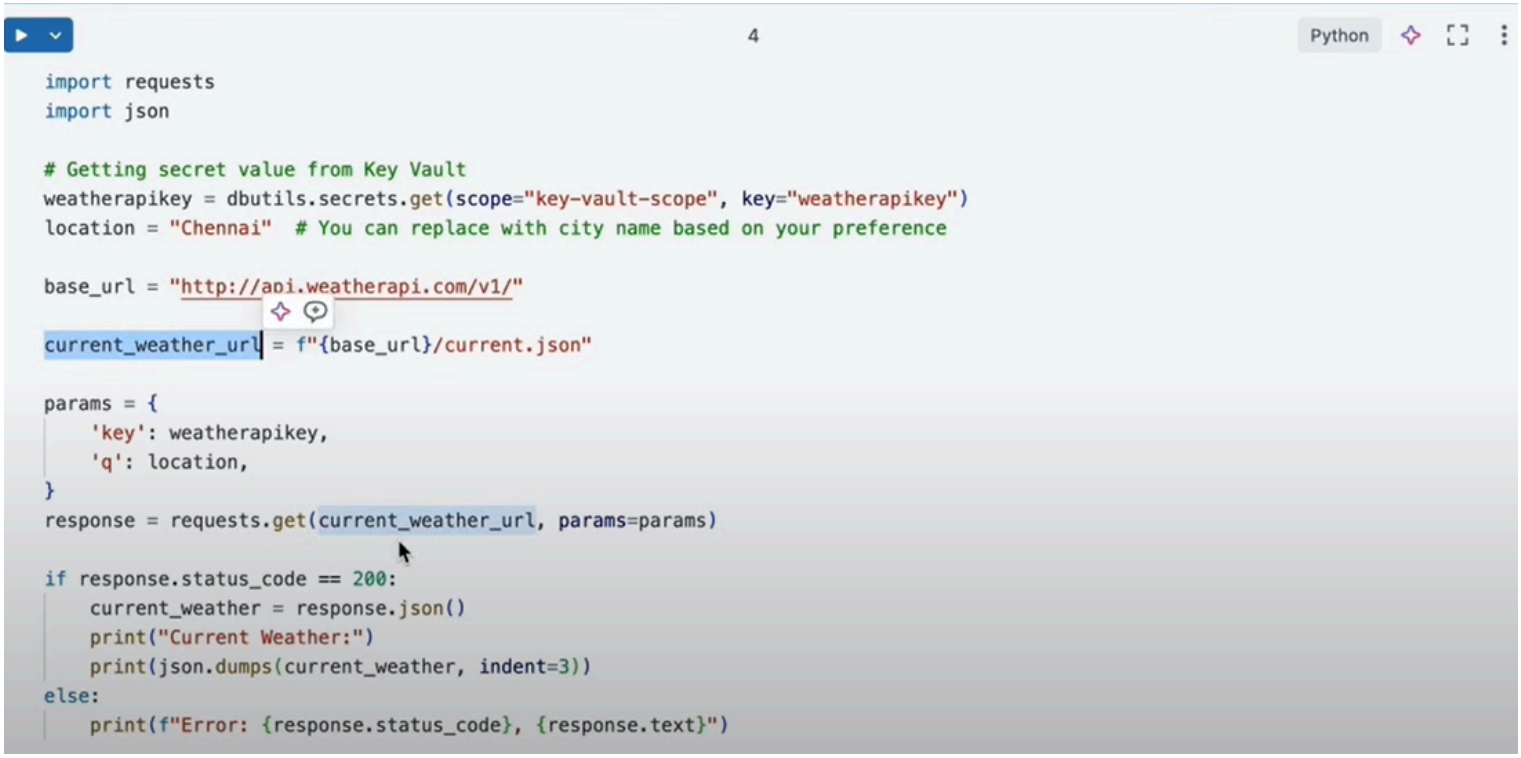
In the members: you need to put Azuredatabricks



So access needs to be granted from both sides: databricks to azure and vice versa.
Now go to databricks notebook: change event id to 222 from 1111



6. Weather api testing in DB.
Go in azure notebook



Key is fetched from keyvault
Read proper documentation from api.

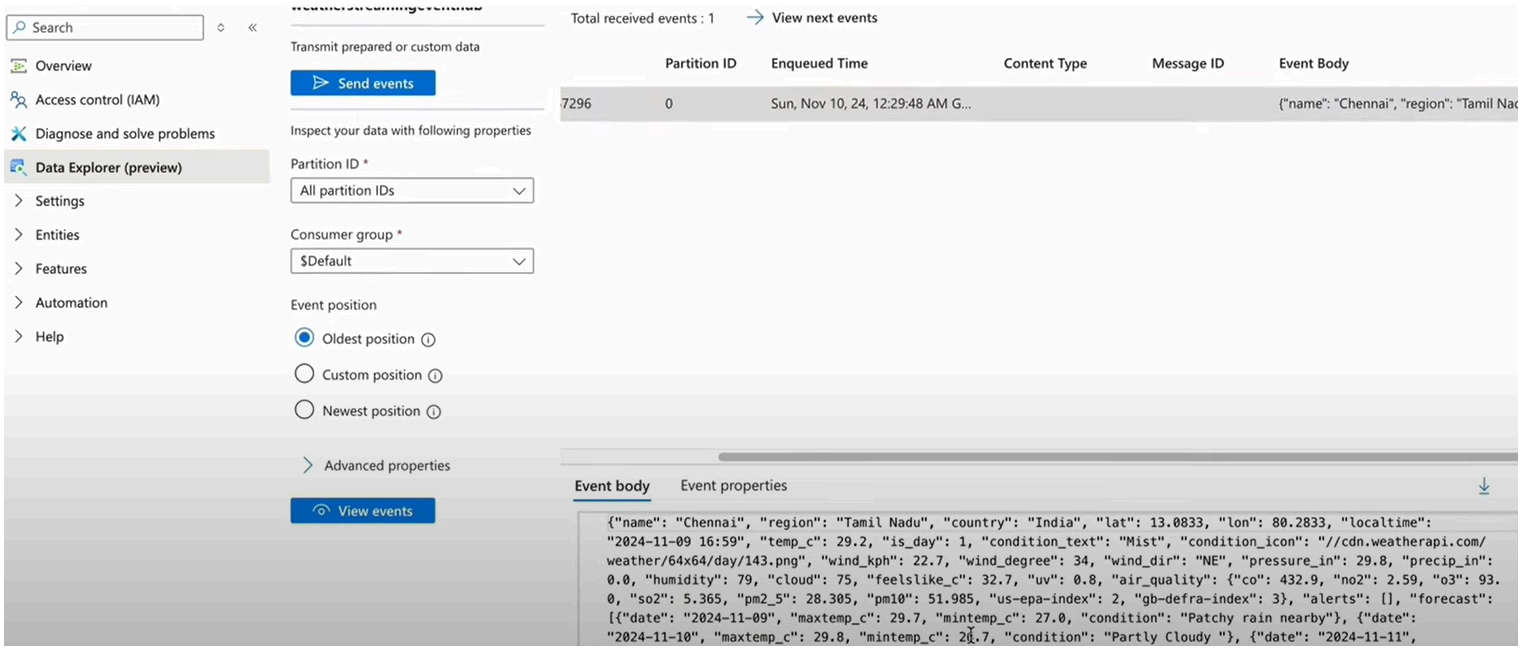
Basically just invoking the api and storing json to current weather variable.

```
    "lat": 13.0833,
    "lon": 80.2833,
    "tz_id": "Asia/Kolkata",
    "localtime_epoch": 1730553270,
    "localtime": "2024-11-02 18:44"
  },
  "current": {
    "last_updated_epoch": 1730552400,
    "last_updated": "2024-11-02 18:30",
    "temp_c": 28.1,
    "temp_f": 82.7,
    "is_day": 0,
    "condition": {
      "text": "Partly Cloudy",
      "icon": "//cdn.weatherapi.com/weather/64x64/night/116.png",
      "code": 1003
    },
    "wind_mph": 6.9,
    "wind_kph": 11.2,
    "wind_degree": 88,
    "wind_dir": "E",
```


7. Developing complete code to different types of weather data. : write code in datbricks(any data is fine.)

8. Final data ingestion from azure databricks to event hub.

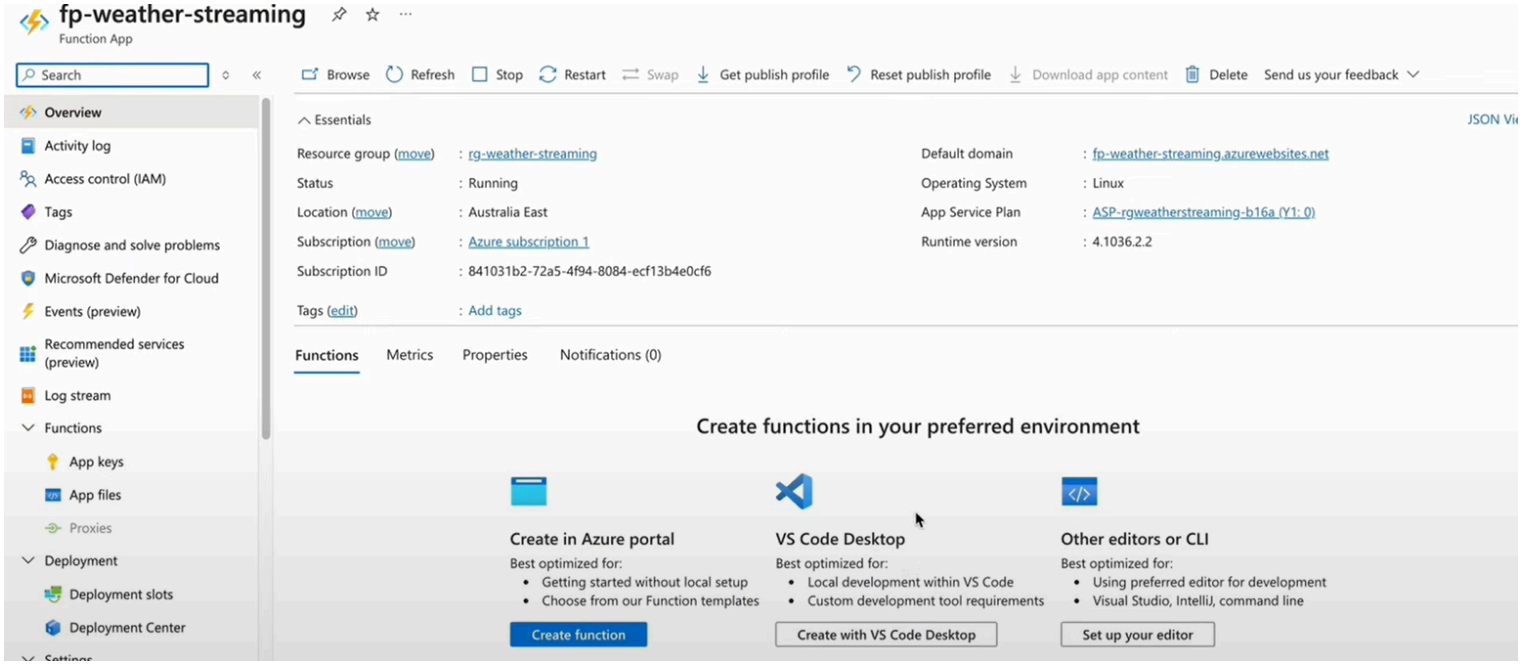
Combine the event code and api response, so that the response api can be sent to event hub as an event.



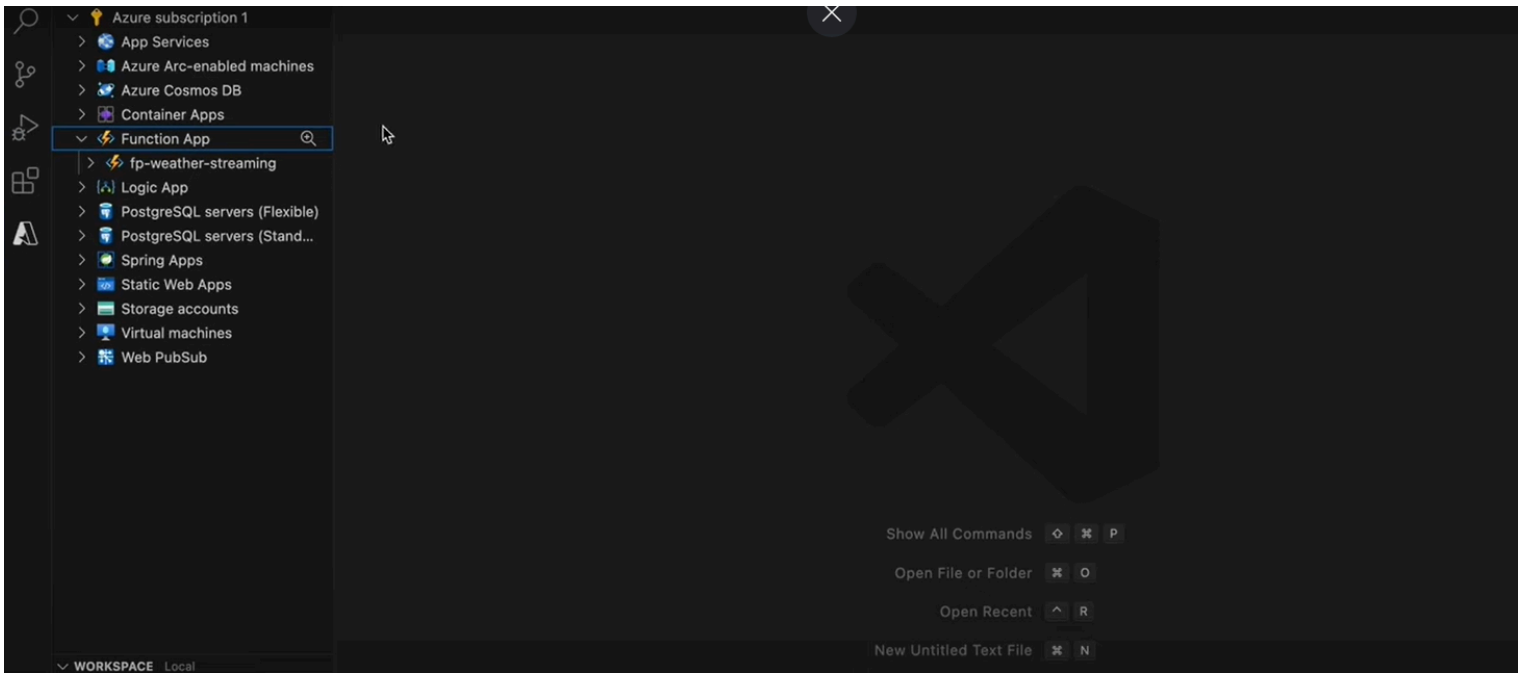
Go to Databricks and set up a streaming job that continuously pushes data to Azure Event Hub in batches. Configure the stream to emit data at a rate of 30 rows per second. This process should run automatically until the source API has no more data or all data has been consumed. Each output batch should be sent as a single event to Event Hub.

```python
# Main program
def process_batch(batch_df, batch_id):
    try:
        # Fetch weather data
        weather_data = fetch_weather_data()

        # Send the weather data (current weather part)
        send_event(weather_data)

    except Exception as e:
        print(f"Error sending events in batch {batch_id}: {str(e)}")
        raise e

# Set up a streaming source (for example, rate source for testing purposes)
streaming_df = spark.readStream.format("rate").option("rowsPerSecond", 1).load()

# Write the streaming data using foreachBatch to send weather data to Event Hub
query = streaming_df.writeStream.foreachBatch(process_batch).start()
```

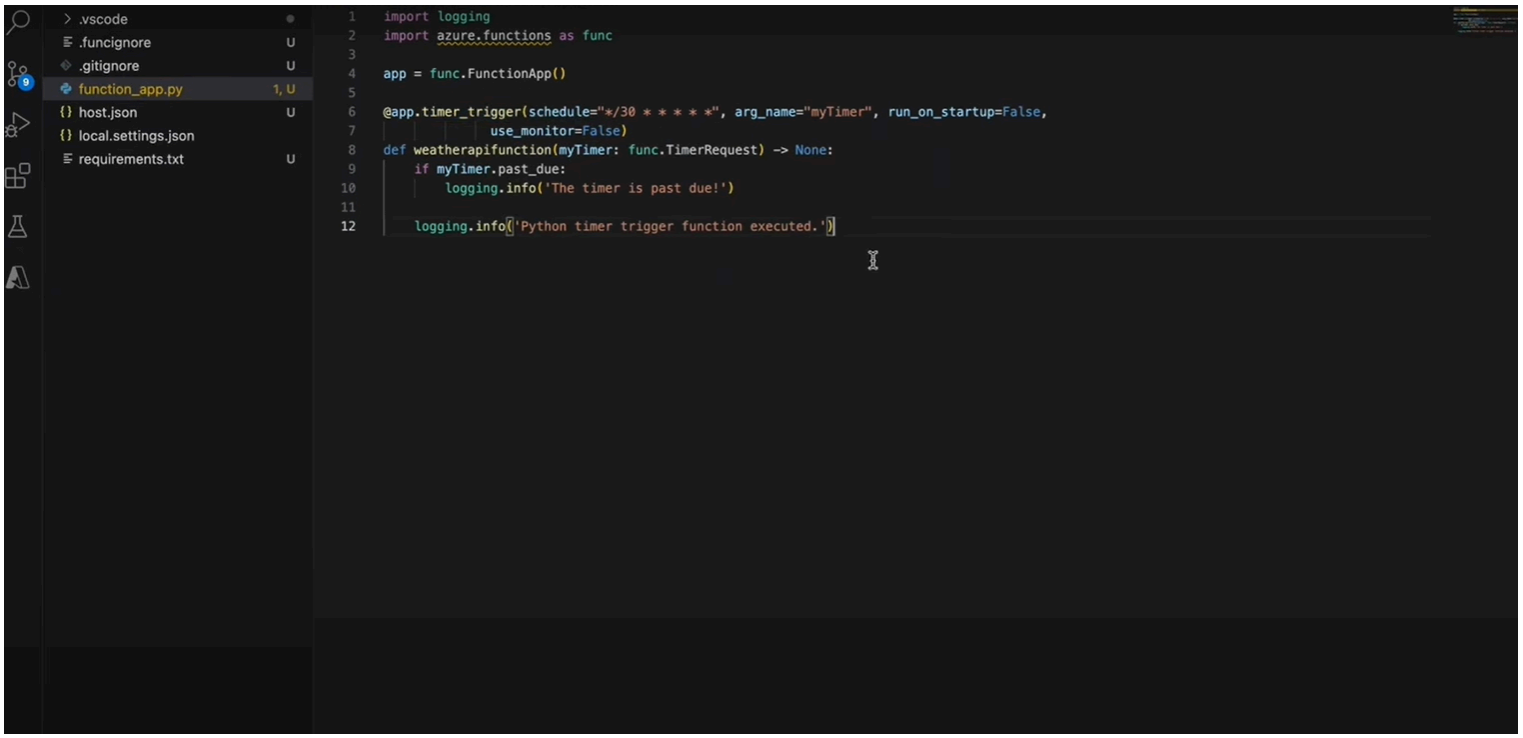
Same data Ingestion using Azure function App.



Same like AWS lambda.
Here you can write code is VS-code. -> in vs code extensions -> azure functions. -> sign in to azure through vscode.
Note thai this VS-code is different than usual, it has some operations which you see in the drop down wherever in it.



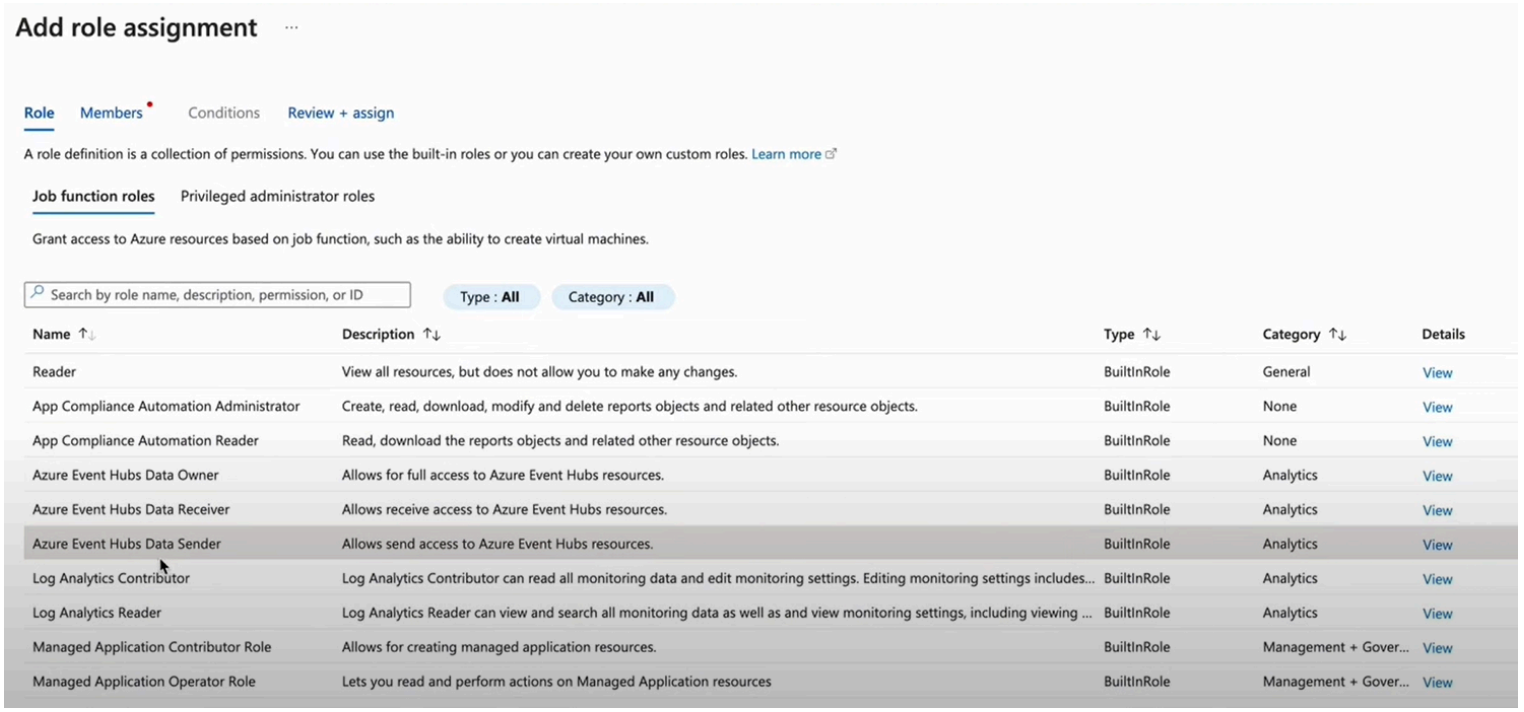
First write the code in local and deploy it to cloud. Hence configure a local directory for this.
This vs-code has time trigger functionality already in dropdown in vs code- put cron equation for after 30 seconds.



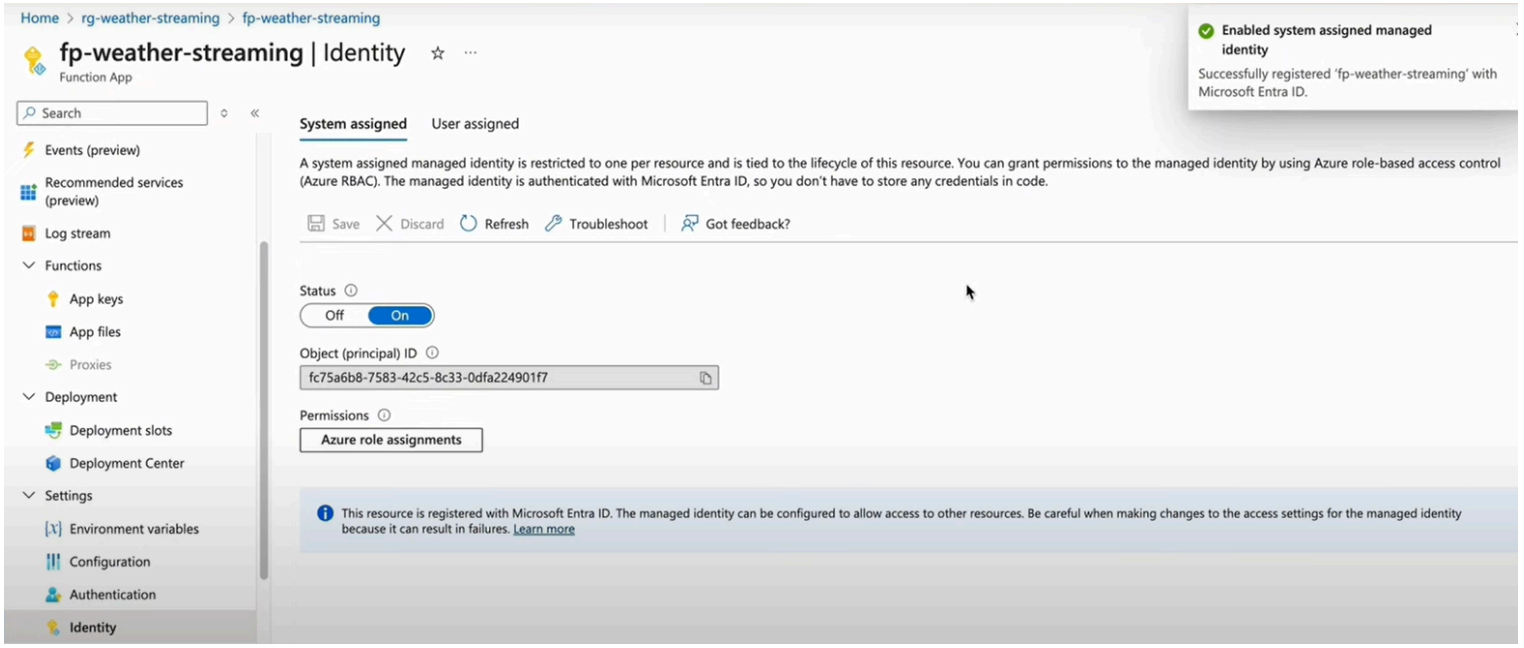
Requirements.txt has prewritten libraries like azure functions. Add event hub.

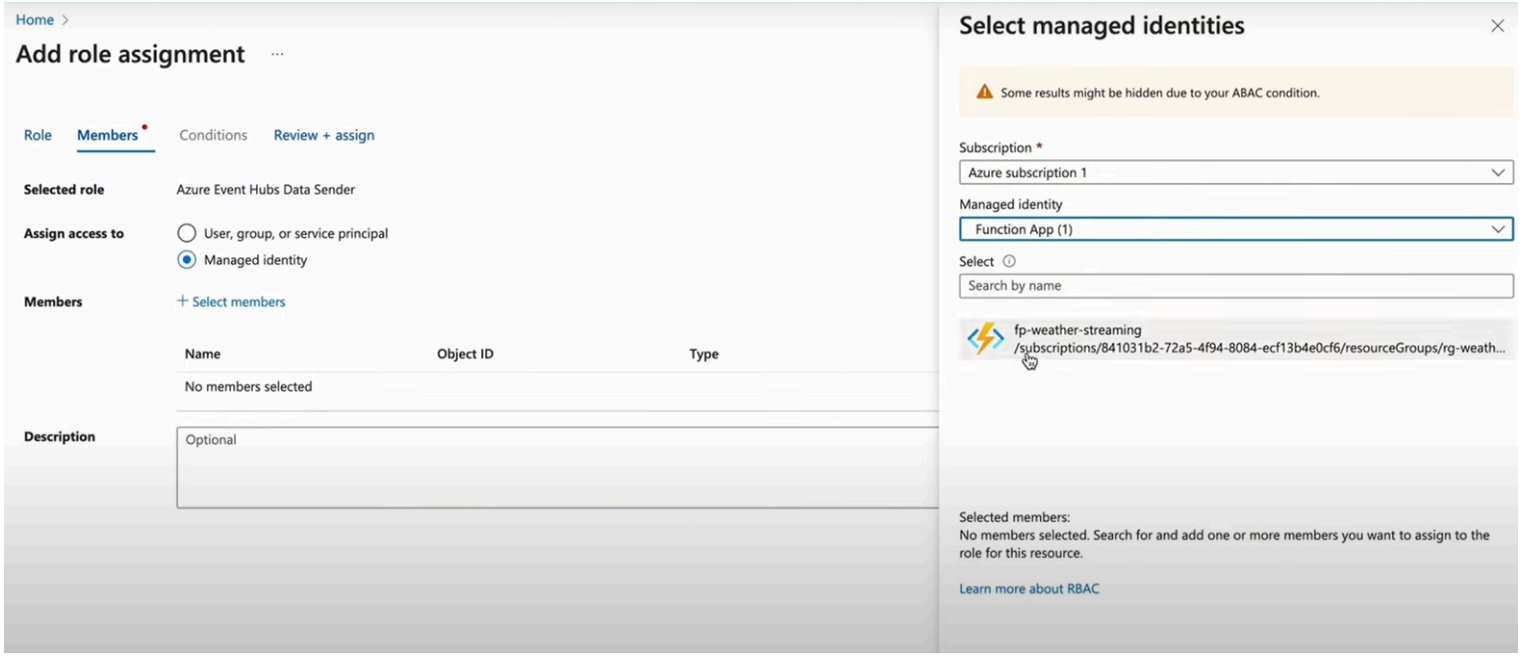
Copy paste databricks code: only now conn needs to be changes. Here you dont needs keys and all. This process is bit easy.

BELOW IAM NEEDS TO BE DONE IN EVENTS HUB IAM



Go to IAM and create managed identity:





Hence now the azure functions has the access to events hub.

Now remove connection string, from databricks which you used in vs code.

Add this managed identity package to vs code as we created managed identity in event hubs IAM

```
from azure.identity import DefaultAzureCredential
```

In vs code create its own new producer, that databricks producer does not work,as it has that connection string Which we dont need as we created the mayonnaise identity connection in IAM of event hub

```
# Event Hub configuration

EVENT_HUB_NAME = "weatherstreamingeventhub"

EVENT_HUB_NAMESPACE = "weatherstreamingnamespace.servicebus.windows.net"

# Uses Managed Identity of Function App
credential = DefaultAzureCredential()

# Initialize the Event Hub producer
producer = EventHubProducerClient(
    fully_qualified_namespace=EVENT_HUB_NAMESPACE,
    eventhub_name=EVENT_HUB_NAME,
    credential=credential
)
```

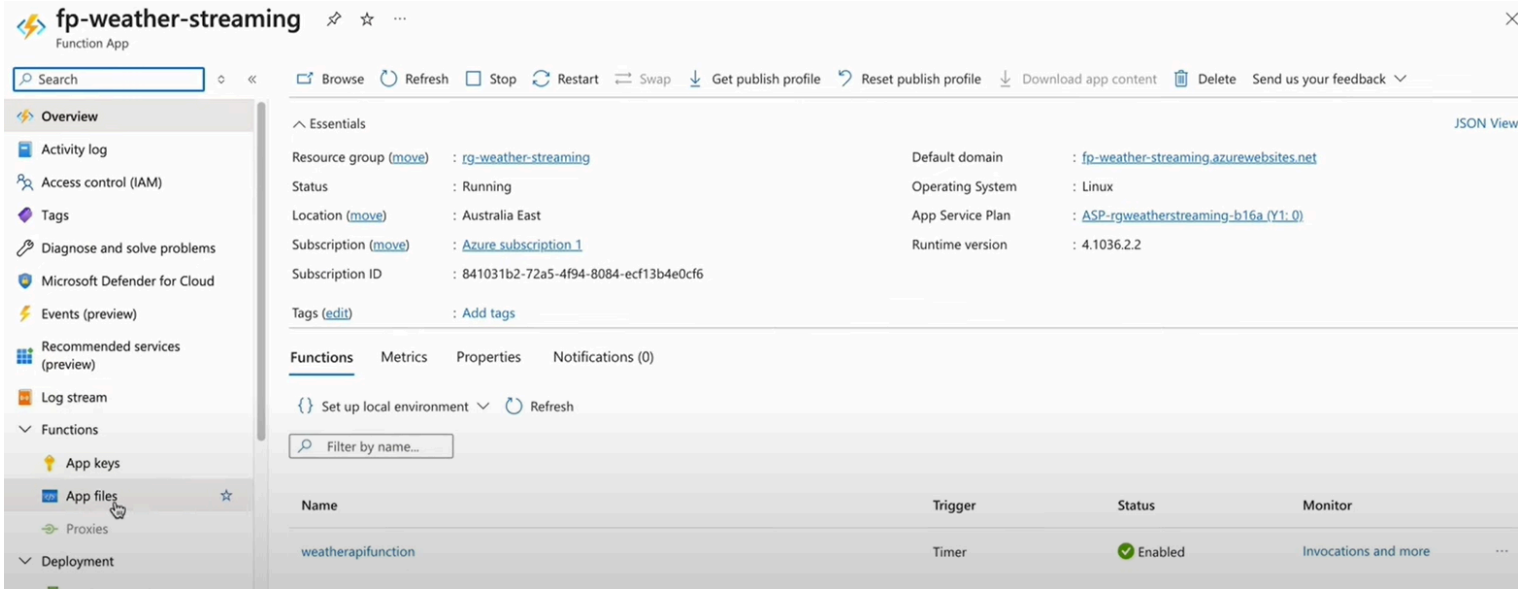
Now we're using couple of secrets as there are secrets from the EPI stored in the keyboard hence to access the api we also need to connect the azure function with the secrets in the secret vault hence for that you need to go in the im of secret world and put the managed identity as azure function from the top down as the same we have configured for the previous step basically by doing this the secrets would be accessed by the function app in the vs code

Also, add this in code.

```
# Fetch the API key from Key Vault
VAULT_URL = "https://kv-weather-streaming.vault.azure.net/"
API_KEY_SECRET_NAME = "weatherapikey"
weatherapikey = get_secret_from_keyvault(VAULT_URL, API_KEY_SECRET_NAME)
```

Deploy to azure

You will be able to see the deployment as a azure function:



Go to event hub, you will see events in the event hub.

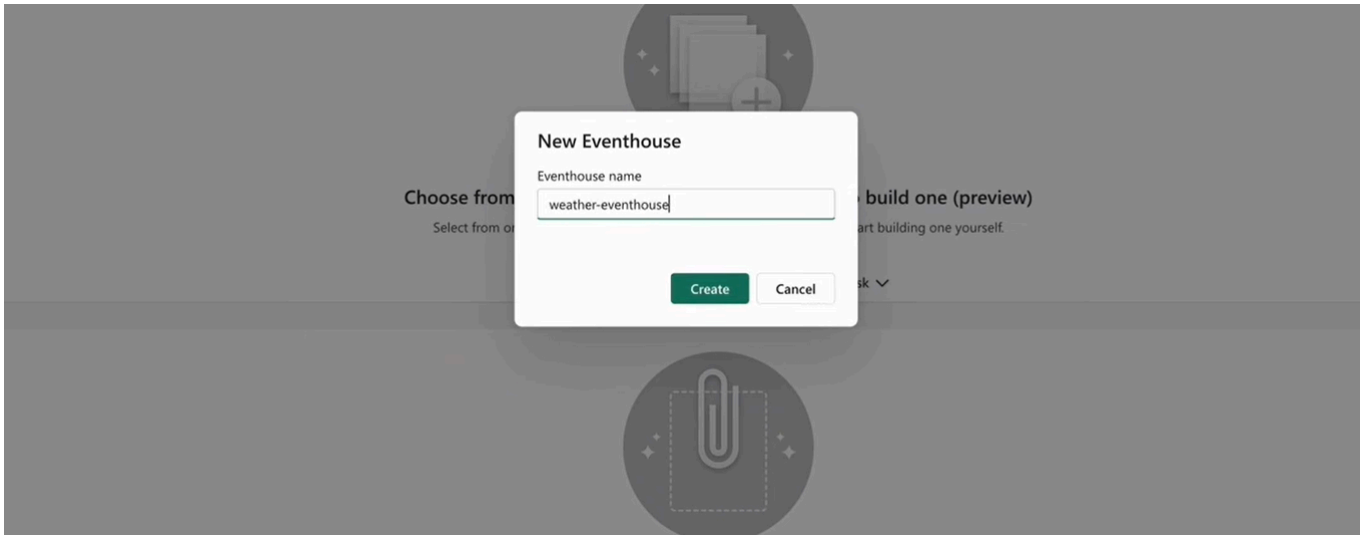
Pricing calculator like amazon in Azure

Now pushing data from Event hub to fabric database.

Go to fabric: create eventhouse:
Eventhouse is a **new preview feature in Microsoft Fabric** designed to enable **real-time event streaming and processing** within the Fabric ecosystem. It's part of the **Real-Time Analytics** workload.

An Eventhouse acts as a **container** or **workspace** that lets you:

- Collect data from sources like Event Hubs, IoT Hubs, or custom APIs
- Process and route that data using **Eventstreams**
- Store, analyze, and visualize streaming data in near real-time

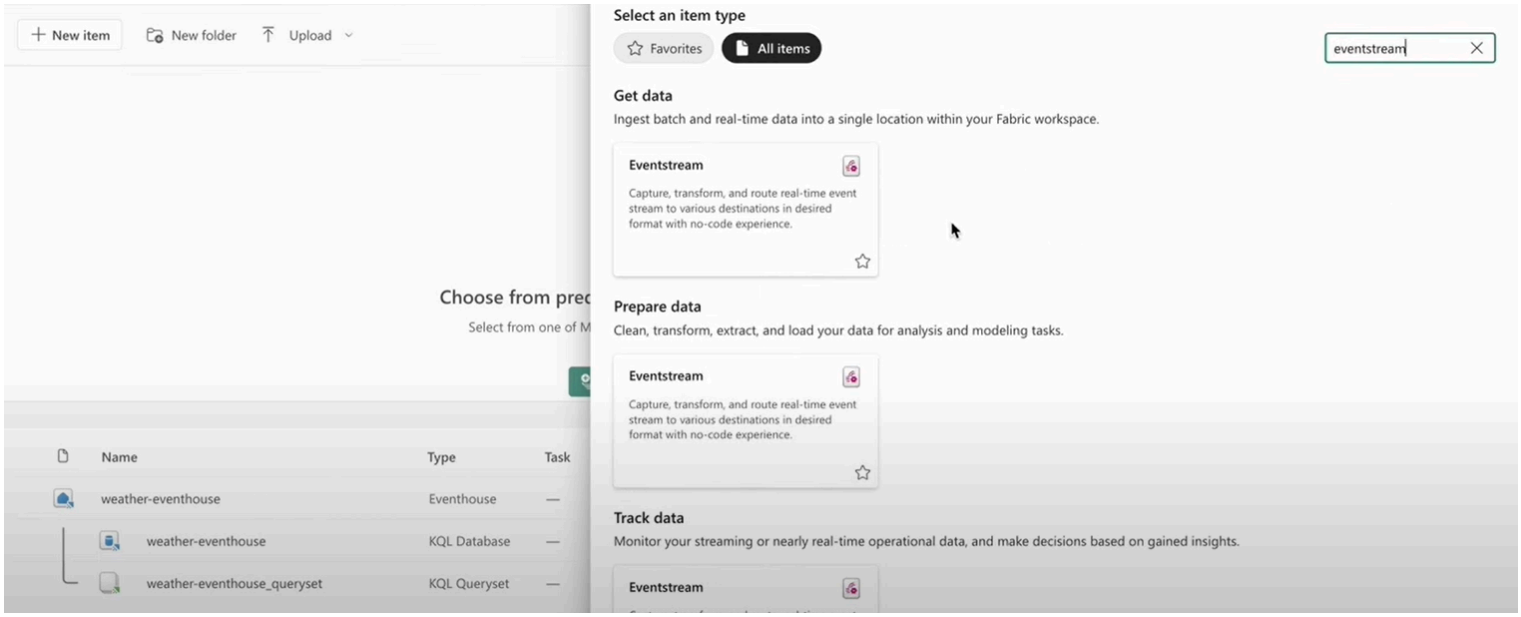


It Creates K Q L database
A **KQL Database** in an **Eventhouse** within **Microsoft Fabric** is a special type of database designed for **real-time analytics and log/event data exploration** using the **Kusto Query Language (KQL)**.

In the context of an Eventhouse, the KQL Database acts as the **destination** where incoming streaming data is stored and made queryable. As events are ingested through sources like Azure Event Hub or IoT Hub via an Eventstream, they are routed into tables within this KQL Database. Once data lands in the KQL Database, you can run KQL queries to analyze trends, detect anomalies, and monitor activity in real time.

This setup is especially powerful for scenarios involving telemetry, application monitoring, security logs, or IoT event streams — where data is high in volume, arrives continuously, and must be queried instantly. The KQL Database integrates seamlessly with other Fabric components, allowing data to be visualized directly in Power BI or used in further downstream analytics.

Now create eventstream:

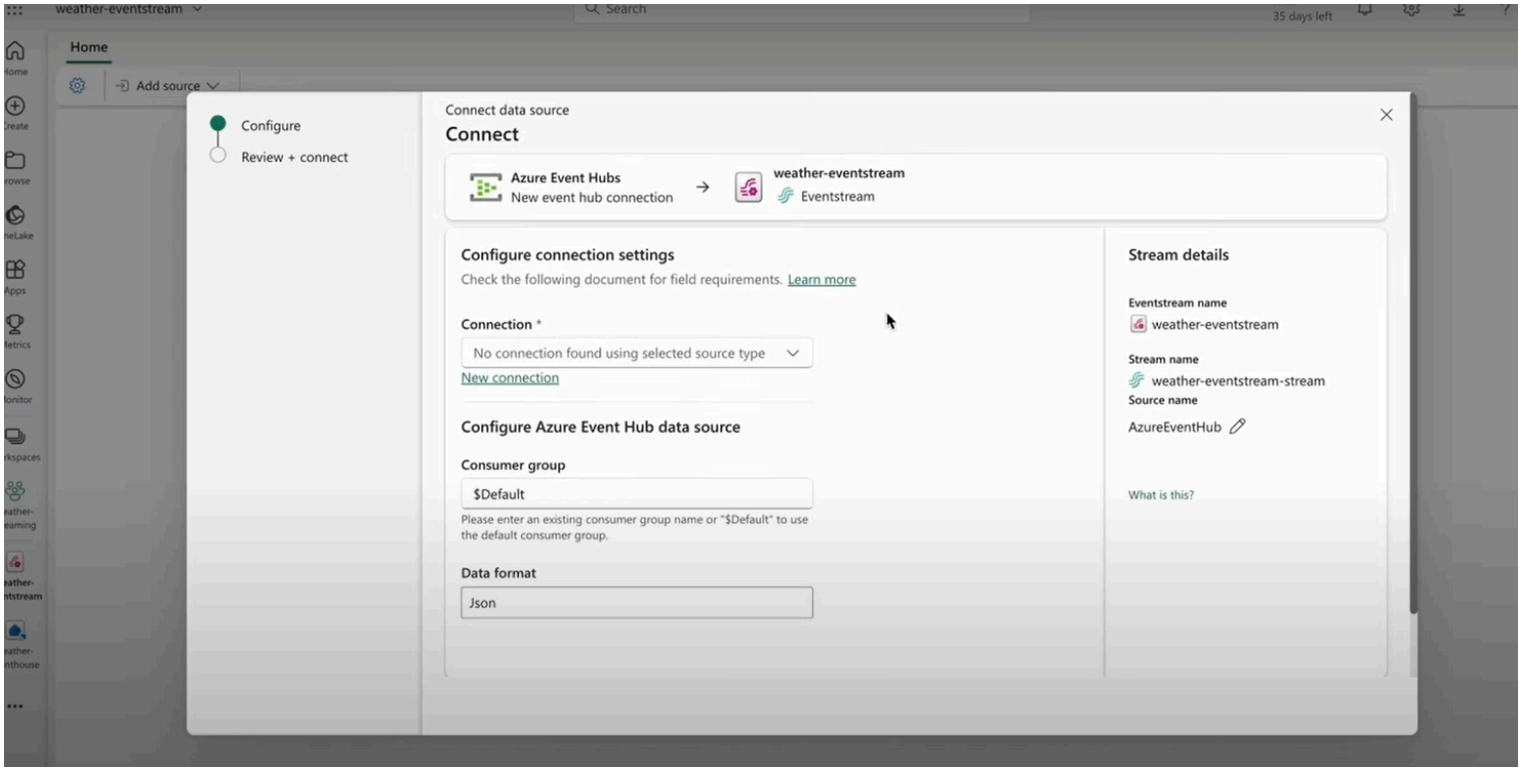


An **Eventstream** in Microsoft Fabric is a real-time data pipeline that **ingests, transforms, and routes streaming data** from various sources to destinations within the Fabric ecosystem.

It allows you to capture data continuously from sources like **Azure Event Hubs**, **IoT Hubs**, or custom APIs and then direct that data to destinations such as a **KQL Database**, **Lakehouse**, or **Power BI** for real-time analytics and visualization.

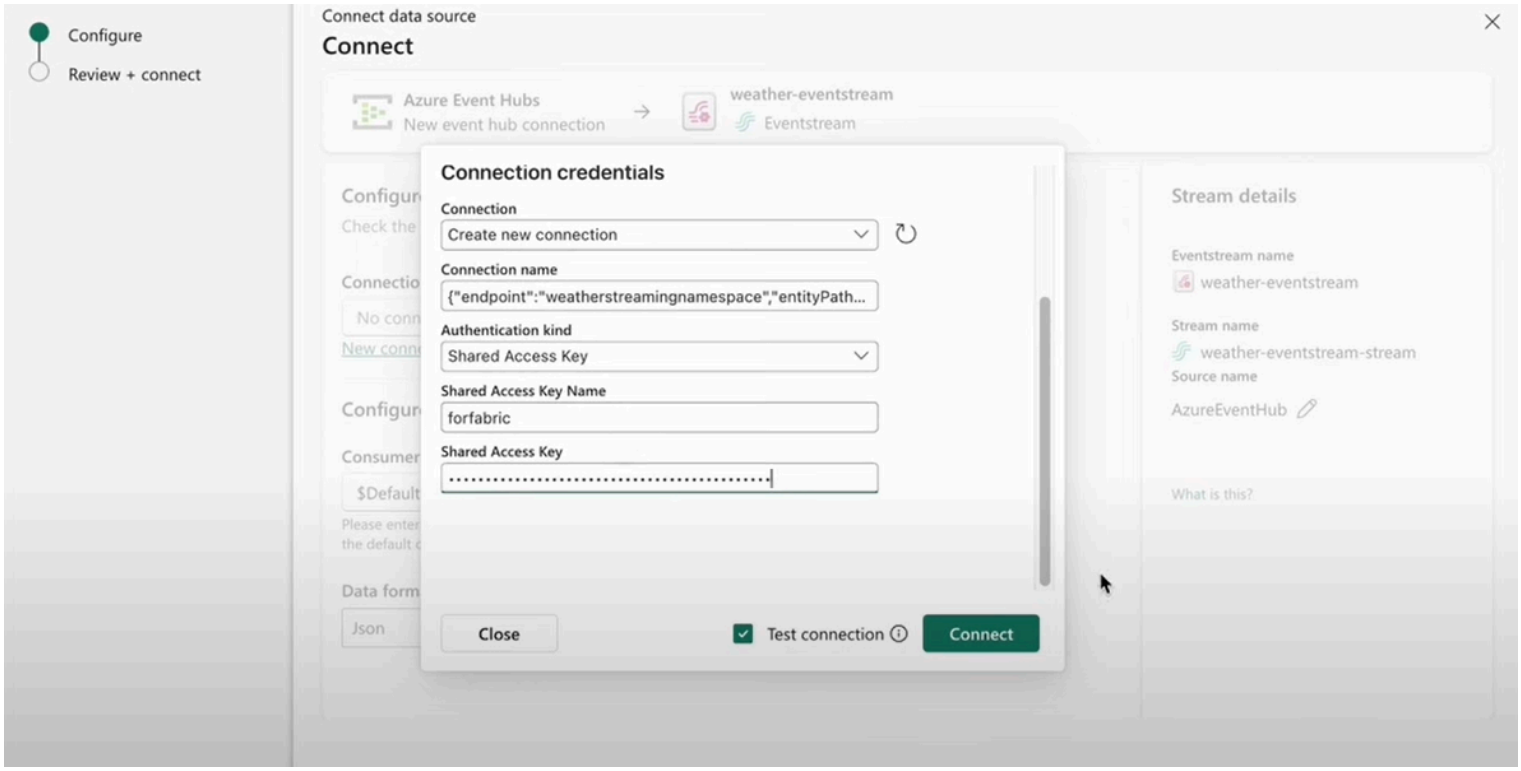
Eventstreams support lightweight transformation logic, such as filtering or enriching data before sending it forward. They play a central role in enabling **event-driven architectures**, **live dashboards**, and **real-time monitoring** inside Microsoft Fabric, all without the need to write complex code or manage infrastructure manually.

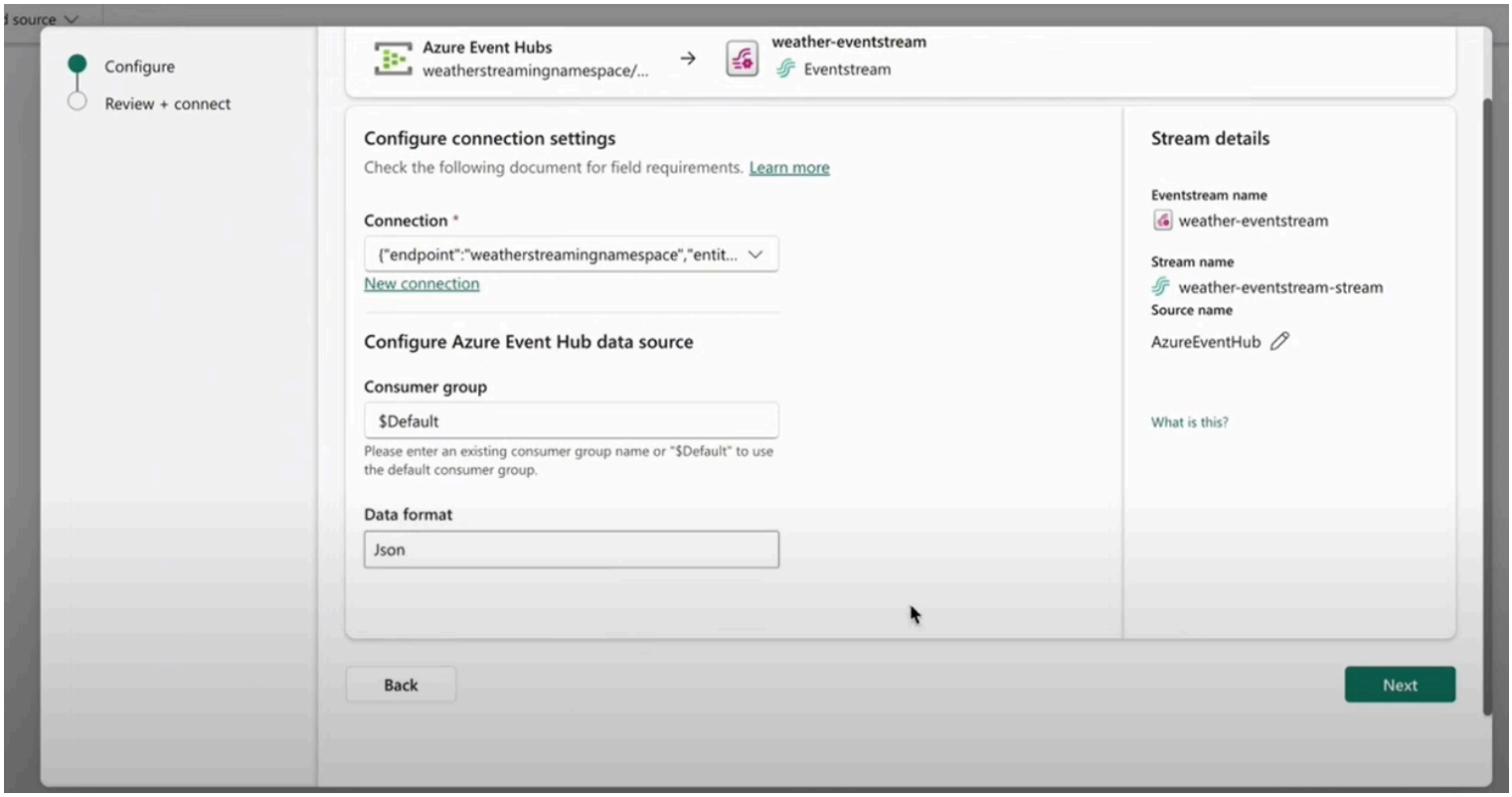
Click On add source in event stream and configure the event hub there in the below screenshot



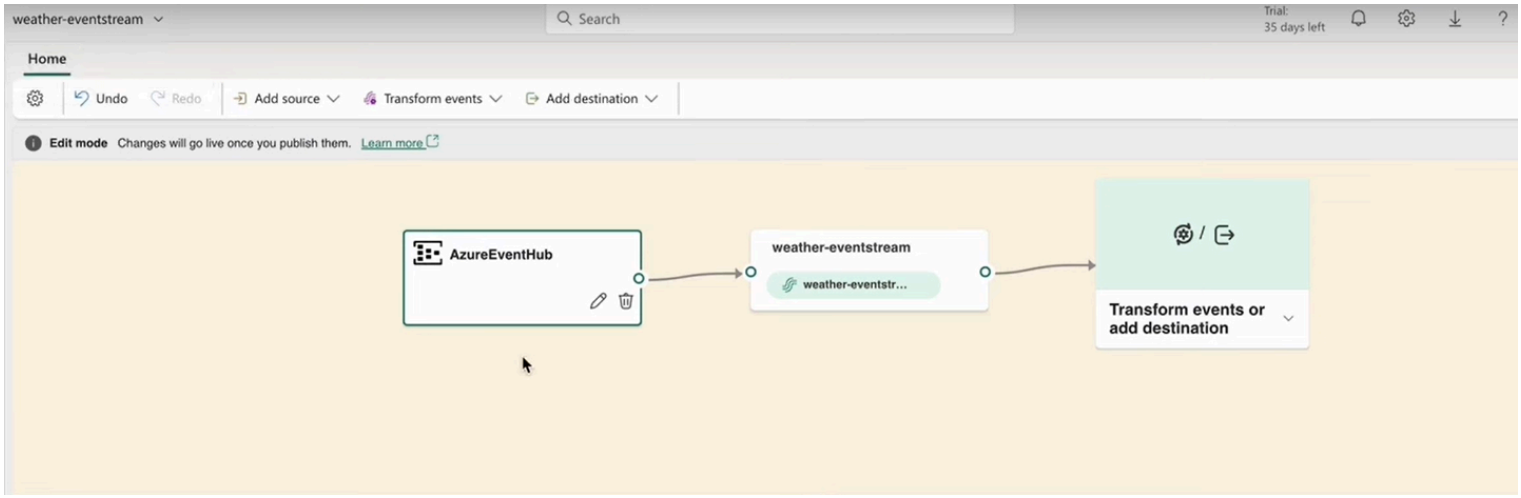
For connecting the future steps you need to create a key in shared access policies in the settings for fabric All this needs to be done in the event hub settings shared access policy stamp as you are connecting the event hub to the event stream and eventually to the kql database in the eventhouse


Search
Add
Overview
Access control (IAM)
Diagnose and solve problems
Data Explorer
Settings
Shared access policies
Configuration
Properties
Locks
Entities
Features
Automation
Help
Search to filter items...
Policy
Claims
fordatabricks
Send
forfabric
Listen

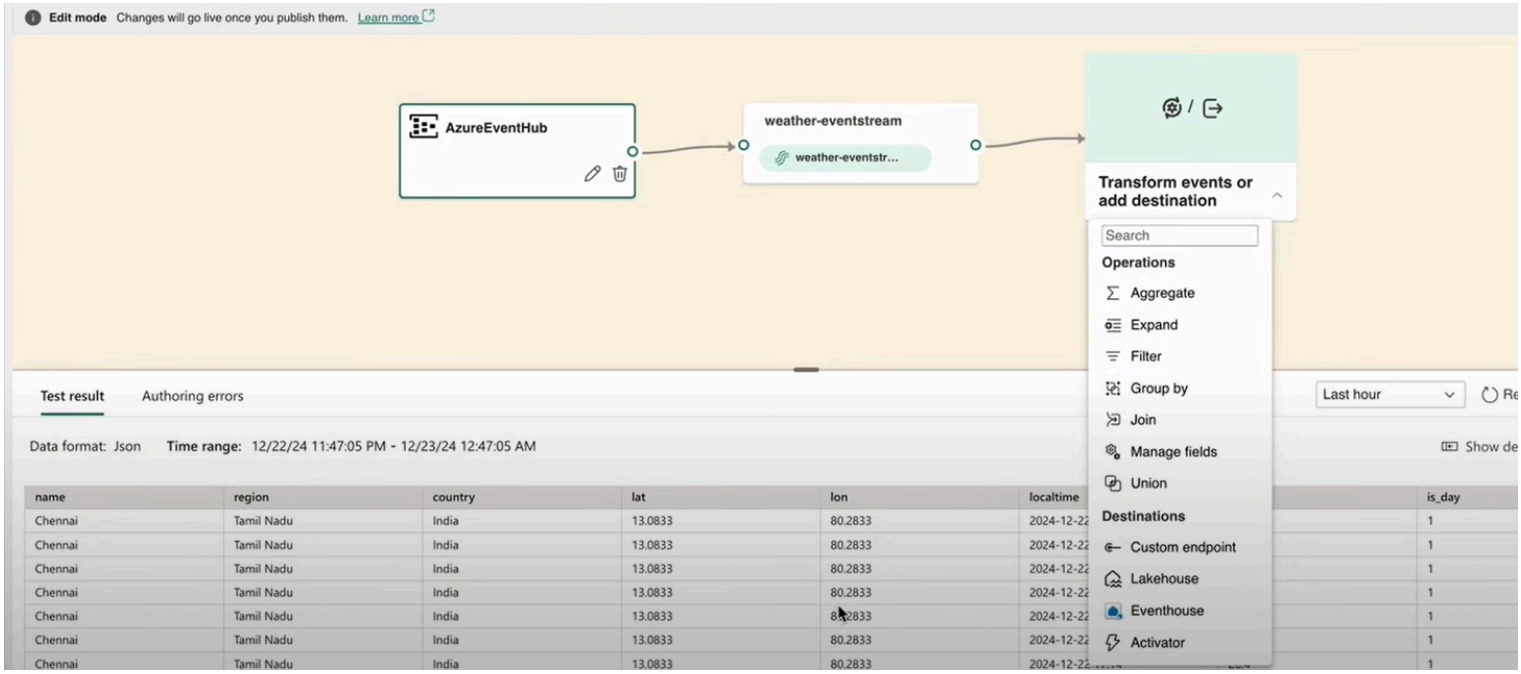

Configure
Review + connect
Connect data source
Connect
Azure Event Hubs
New event hub connection
weather-eventstream
Eventstream
Connection credentials
Connection
Create new connection
Connection name
{"endpoint":"weatherstreamingnamespace","entityPath...
Authentication kind
Shared Access Key
Shared Access Key Name
forfabric
Shared Access Key
Close
Test connection
Connect
Stream details
Eventstream name
weather-eventstream
Stream name
weather-eventstream-stream
Source name
AzureEventHub
What is this?

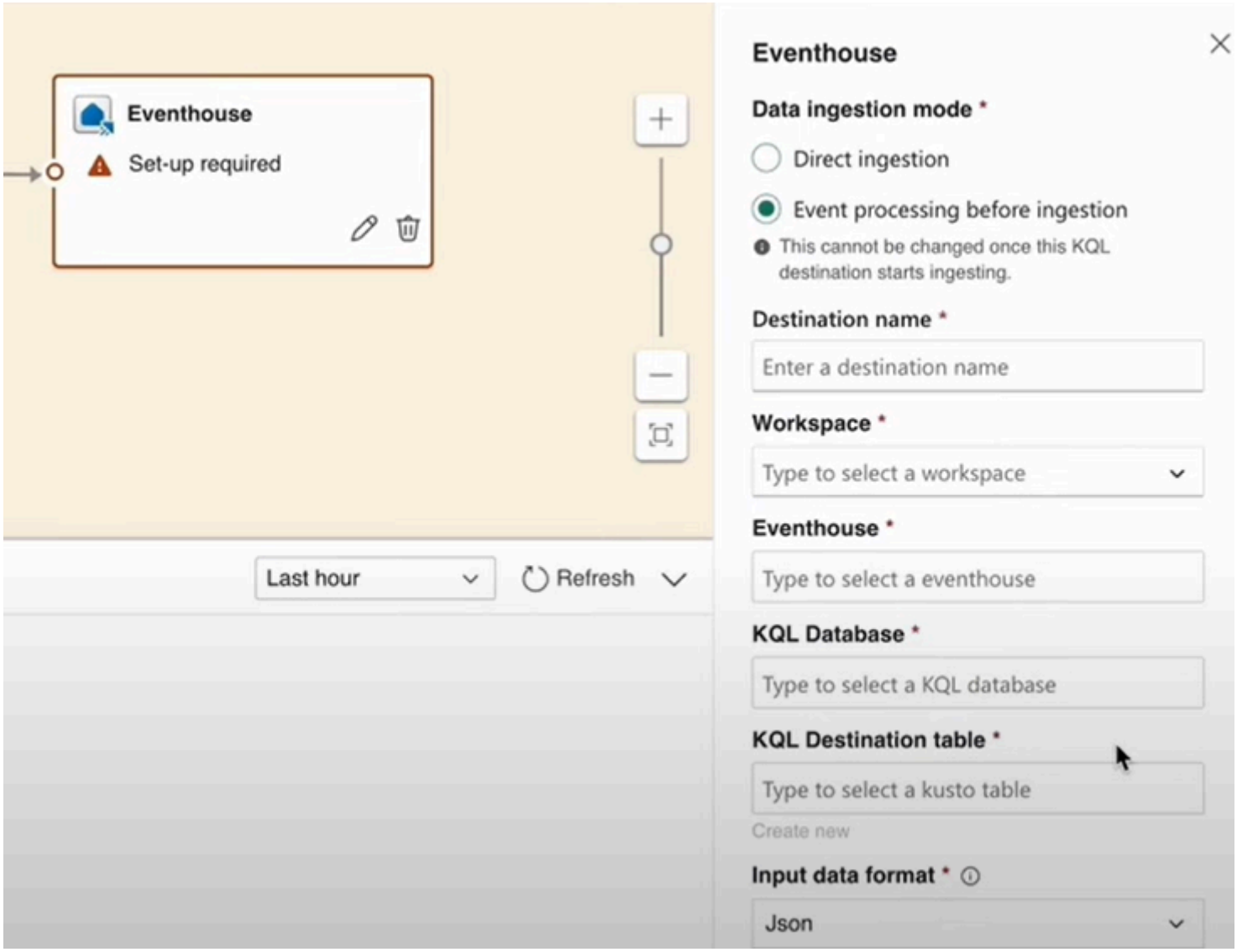

Configure
Review + connect
Azure Event Hubs
weatherstreamingnamespace/...
weather-eventstream
Eventstream
Configure connection settings
Check the following document for field requirements. Learn more
Connection *
{"endpoint":"weatherstreamingnamespace","entit...
New connection
Configure Azure Event Hub data source
Consumer group
$Default
Please enter an existing consumer group name or "$Default" to use the default consumer group.
Data format
Json
Stream details
Eventstream name
weather-eventstream
Stream name
weather-eventstream-stream
Source name
AzureEventHub
What is this?
Back
Next

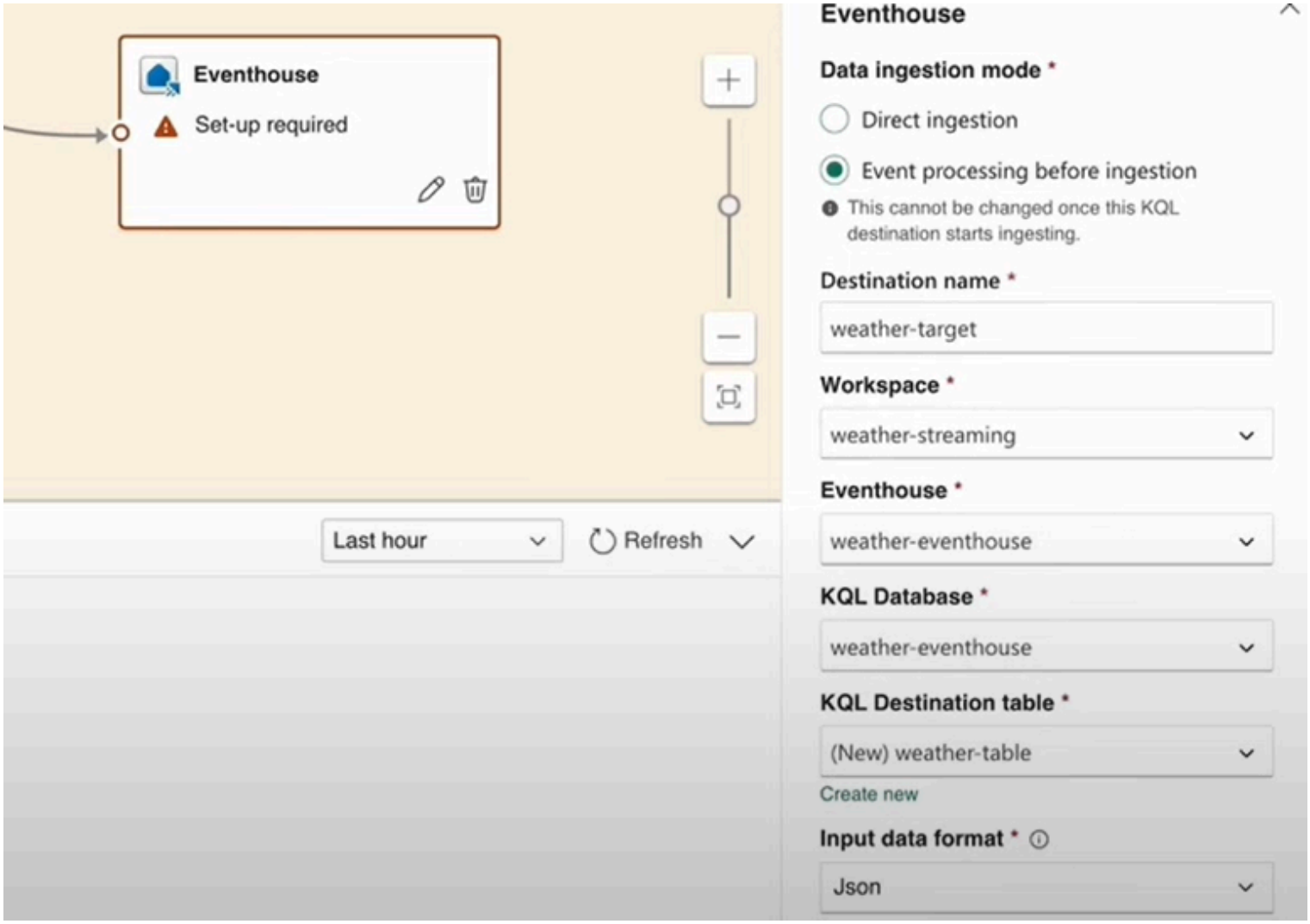
Default consumer group is same in the event stream which is consumer and which is same in the event hub



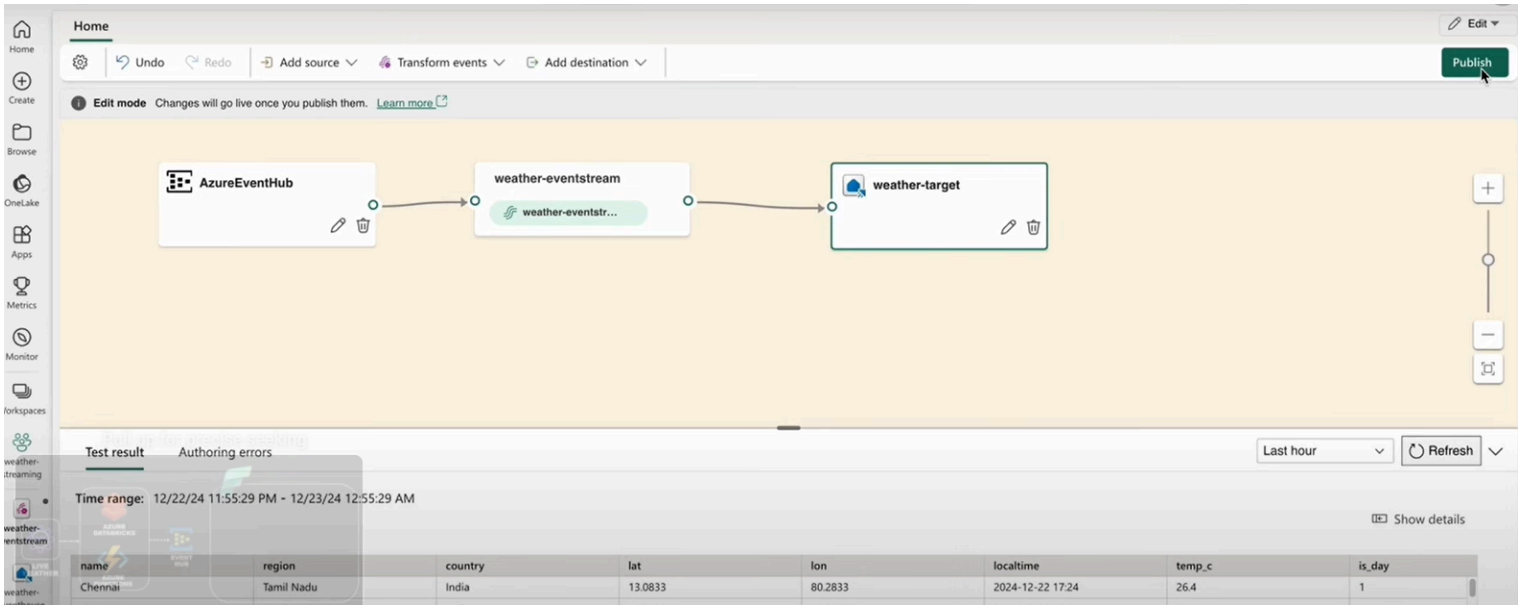
Now let's configure the target which is the kql database in event house




Eventhouse
Set-up required
Last hour
Refresh
Eventhouse
Data ingestion mode *
Direct ingestion
Event processing before ingestion
This cannot be changed once this KQL destination starts ingesting.
Destination name *
Enter a destination name
Workspace *
Type to select a workspace
Eventhouse *
Type to select a eventhouse
KQL Database *
Type to select a KQL database
KQL Destination table *
Type to select a kusto table
Create new
Input data format *
Json



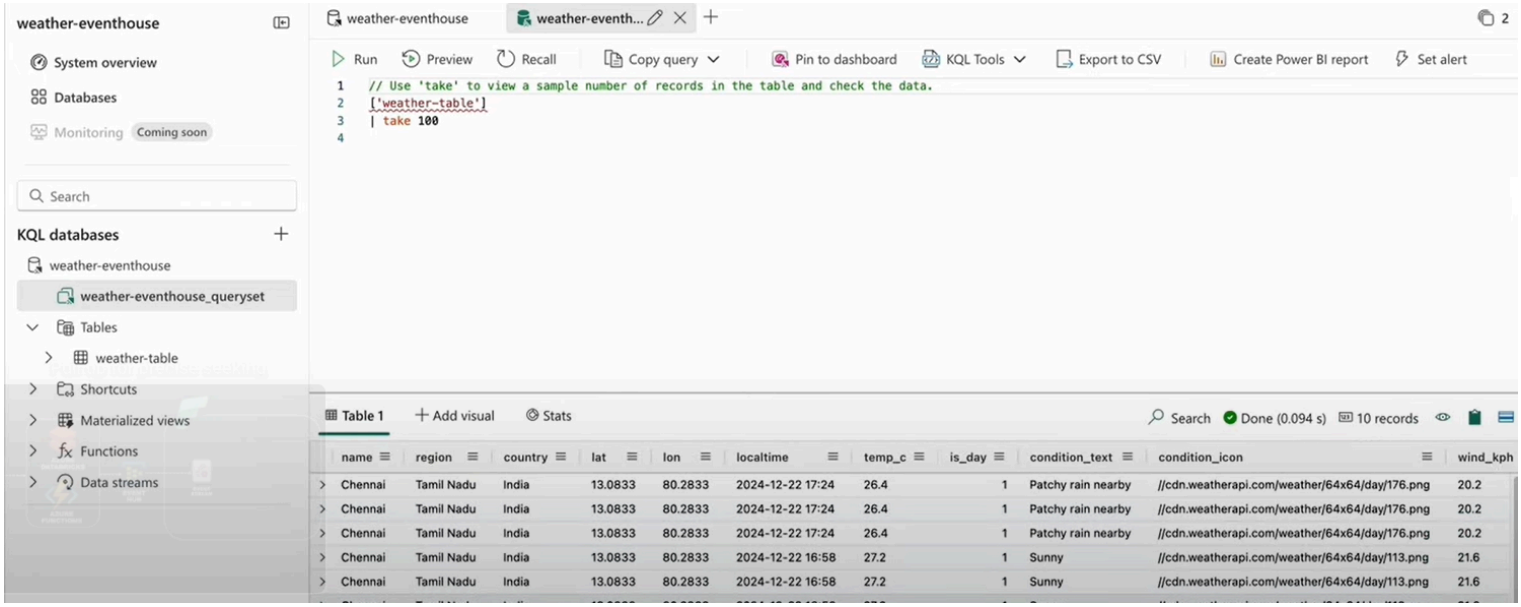
Publish once done.



This process will run 24 by 7 as the Azure function has been coded in the BS code to connect with the weather API and a code involves fetching the data in batches with a 30 second delay which is then connected to the event hub and that event hub is then connected to the azure fabric and in the azure fabric that

event hub is connected to event stream and that event stream is connected to the kql database in the event house.

Kql db screenshot



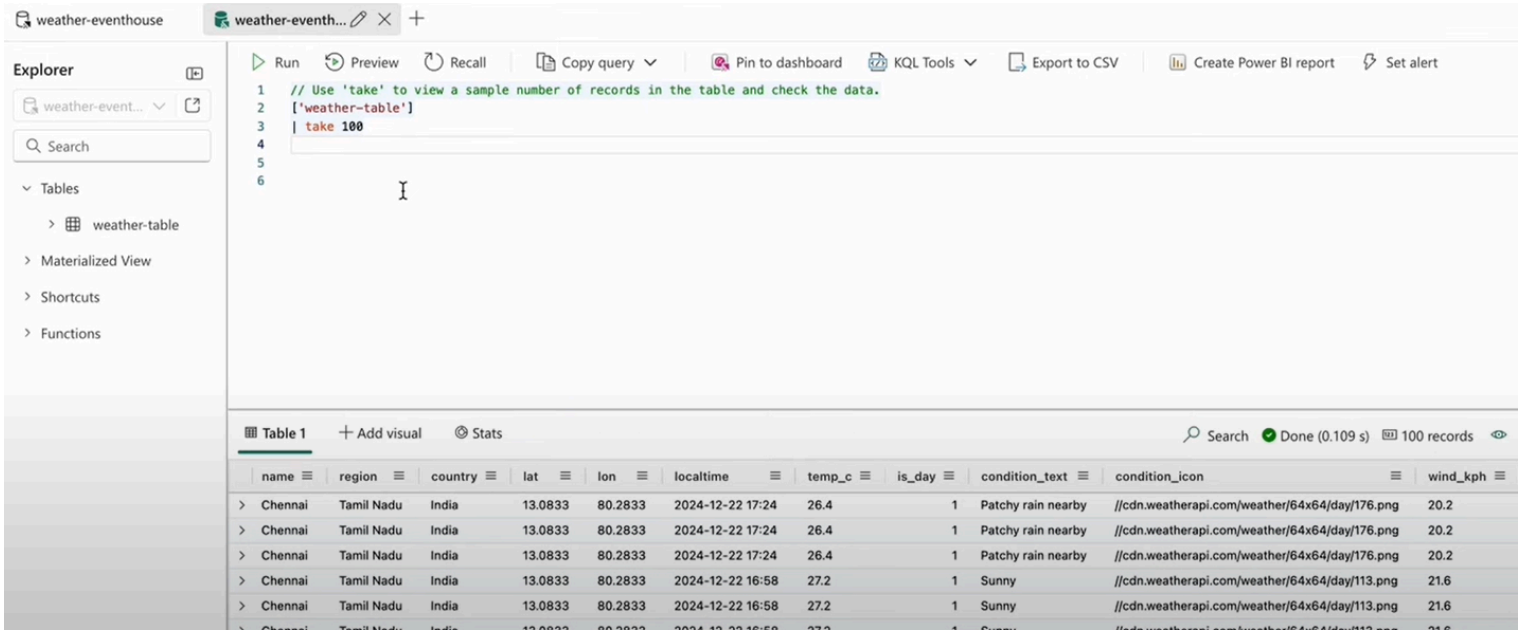
– general queryset in kql info

A **Query Set** in **KQL (Kusto Query Language)** refers to a **collection of saved KQL queries** that you organize together within a workspace, like in Microsoft Fabric or Azure Data Explorer.

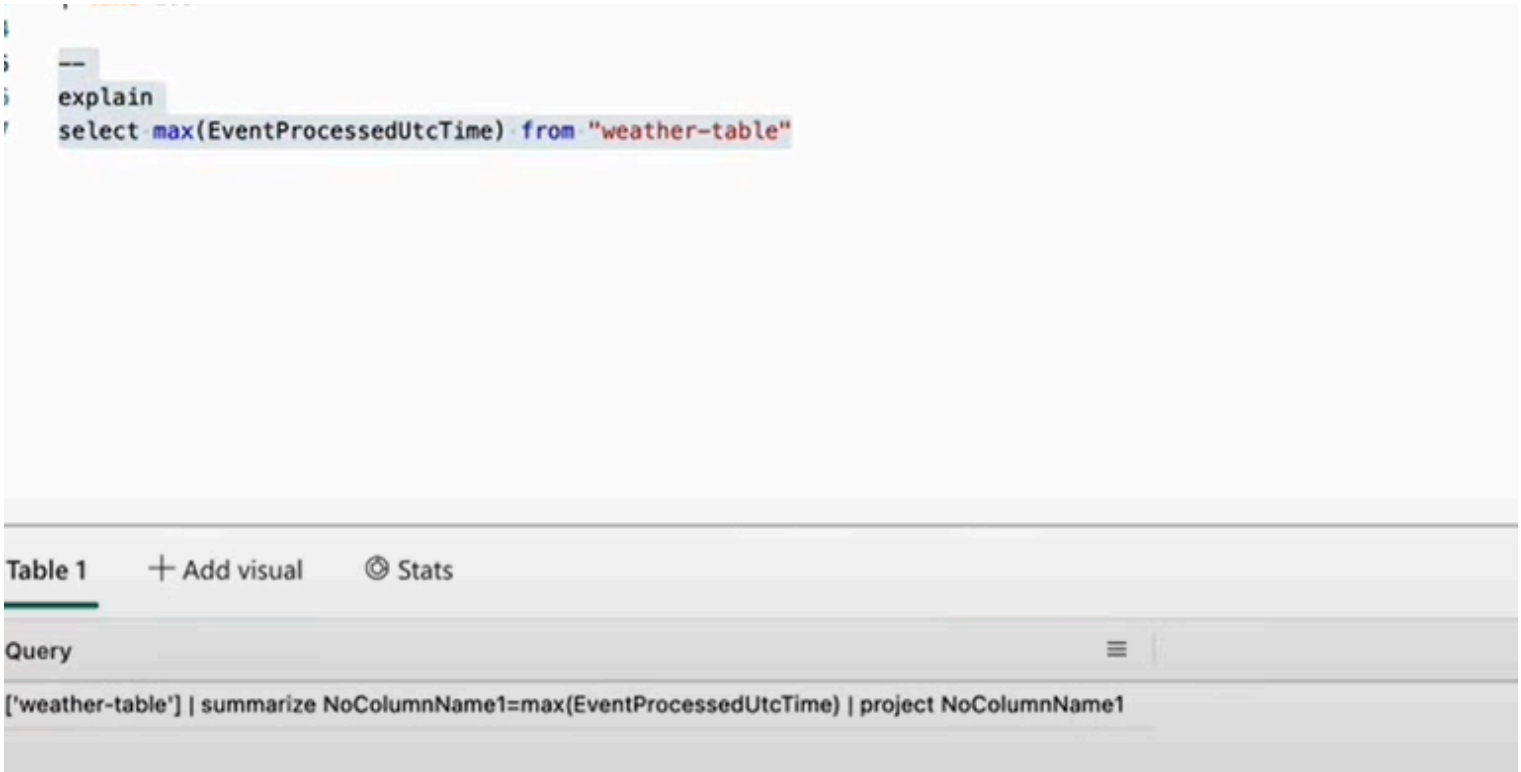
It allows users to:

- **Group related queries** for easier access and reuse.
- **Share and manage queries** as a team.
- Quickly **run, edit, or schedule** commonly used queries from a single place.

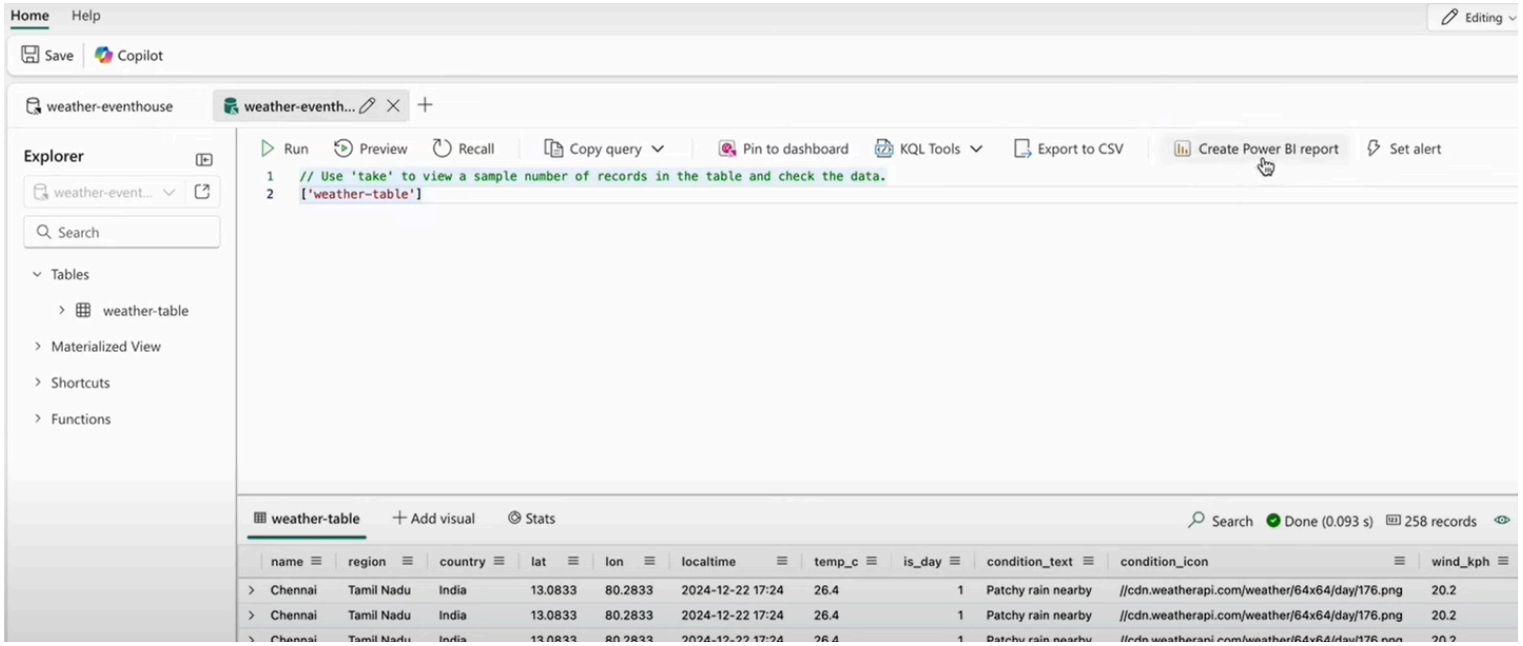
Each query in a query set can target the same or different KQL databases, and these sets are especially helpful for operational dashboards, monitoring tasks, or repeat analysis workflows. Query sets improve productivity and collaboration by storing analytic logic in a structured and reusable format.

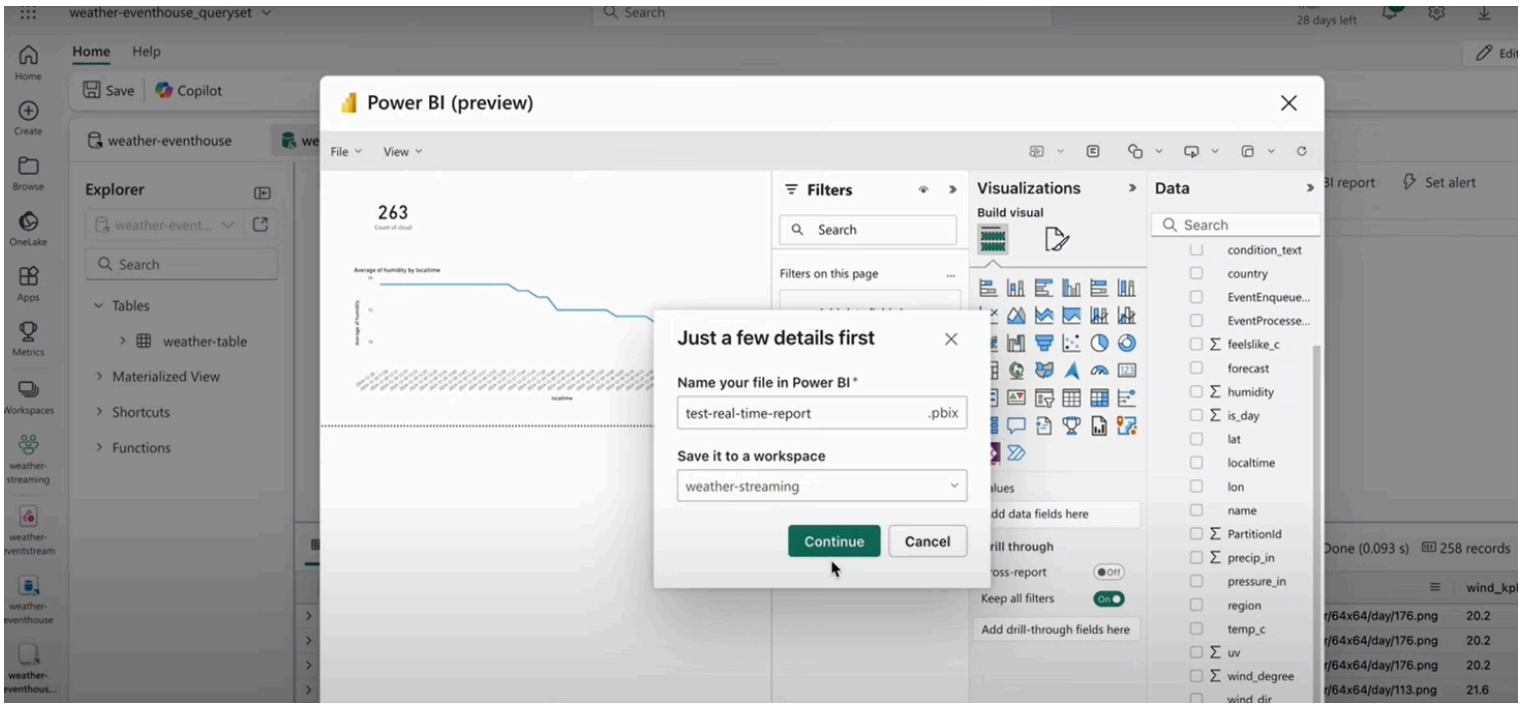


If you want to query kql query set you need to know the kql language but if you don't know it there is a simple way to convert the sql into kql suppose you want to select the maximum number from the event processed UTC time column then what you need to do is just write two dashes enter explain enter select Max SQL query whatever it is after we press shift enter in the terminal the SQL query gets converted into KQL query then copy that KQL query from the terminal and paste it in the editor and you will see your output.



Now for the generated output for that particular query if you have to generate a power bi report there is a button which says power bear report in the query set of the cql query database which is in the eventhouse let me paste the screenshot for your reference.

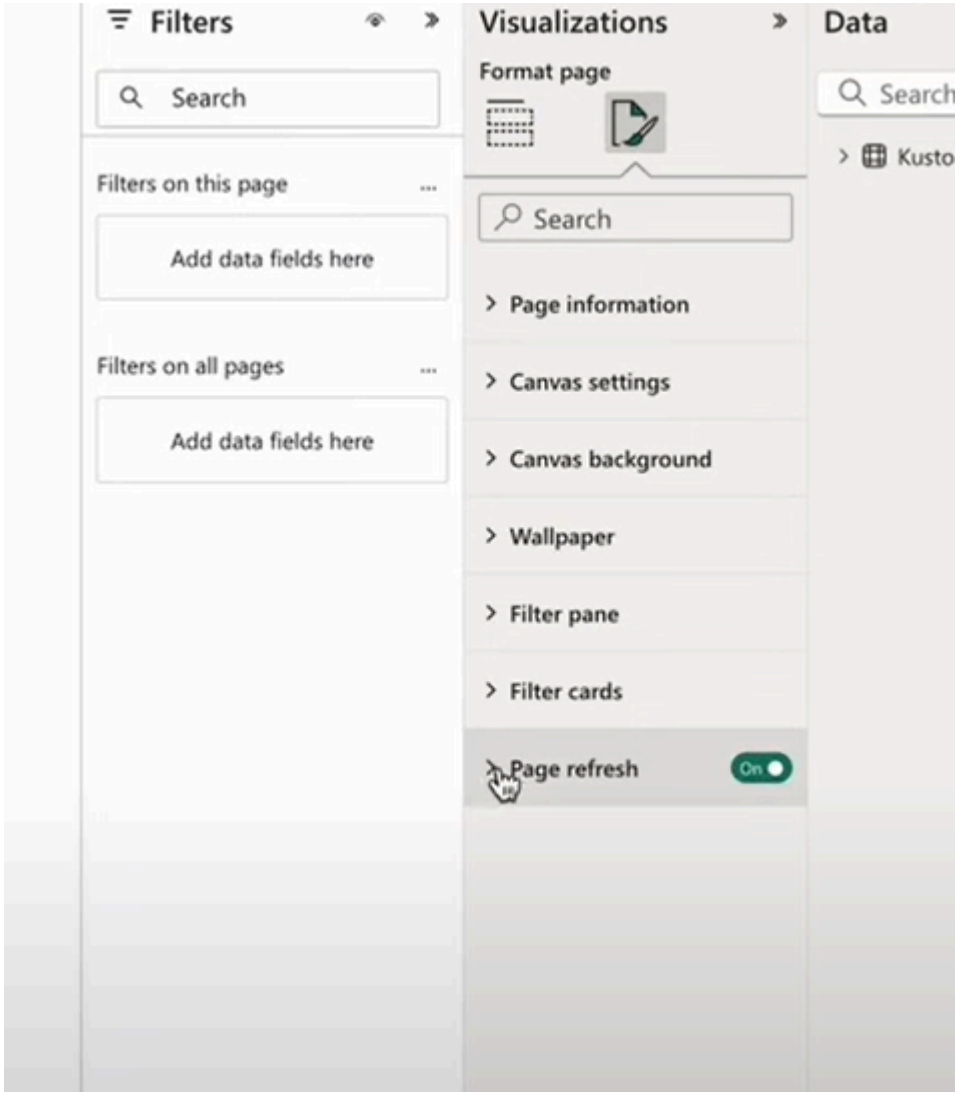




You'll see that the report which is created in power bi is there in the part of the weather streaming workspace if you refresh the page. you will also see the semantic model generated there with the power bi report.

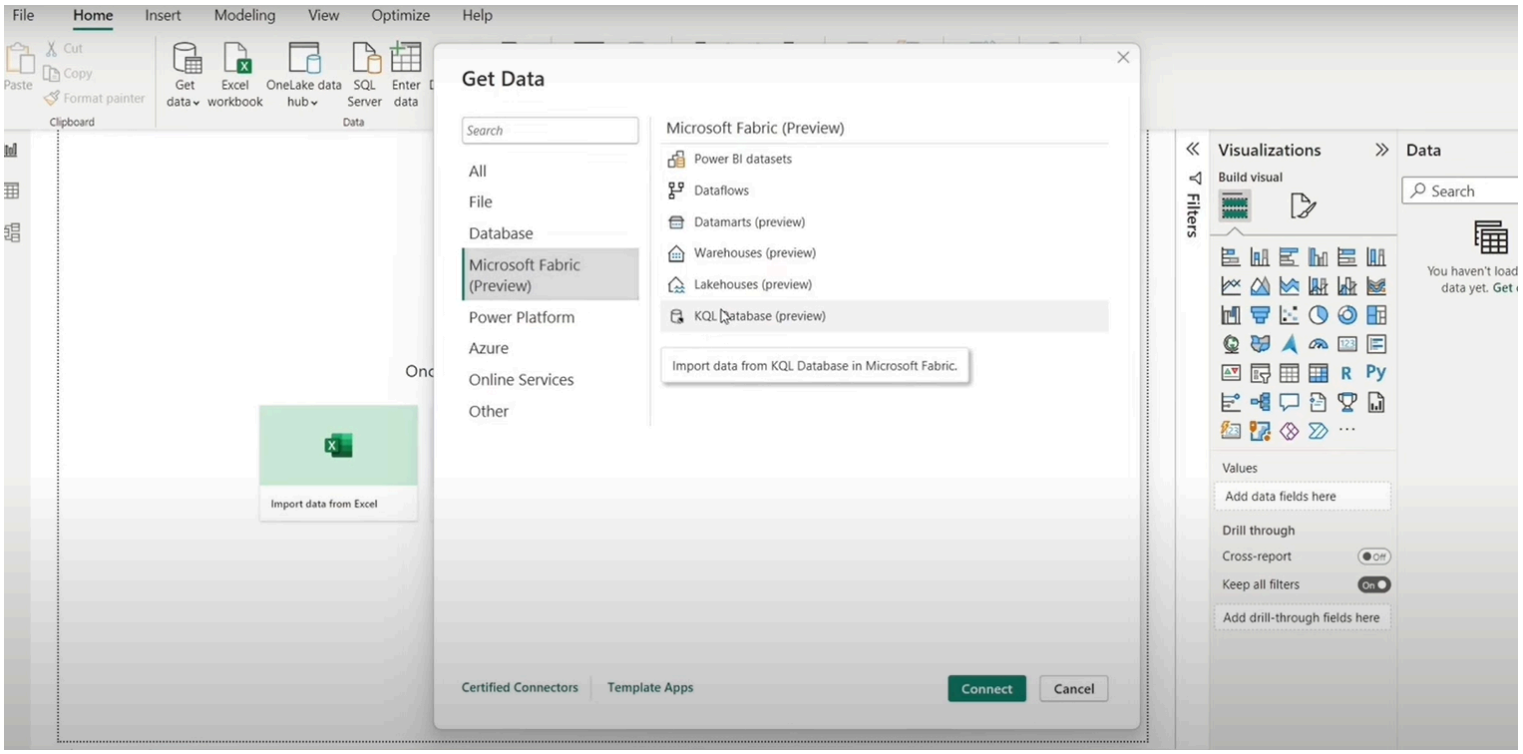
Most important thing about the Arabia report which gets generated is whenever the data is changed from the API it automatically flows through the pipeline which we created and it updates the power BR report automatically so whenever the API data changes for the weather the power BI report because of the pipeline we created in Azure.

That is a button in power bi which is called page refresh so the use of this feature is that the user doesn't need to explicitly refresh the power BR report if the page refresh is enabled whenever the user is on that report it automatically refreshes the page as the data is refreshed through our pipe line.



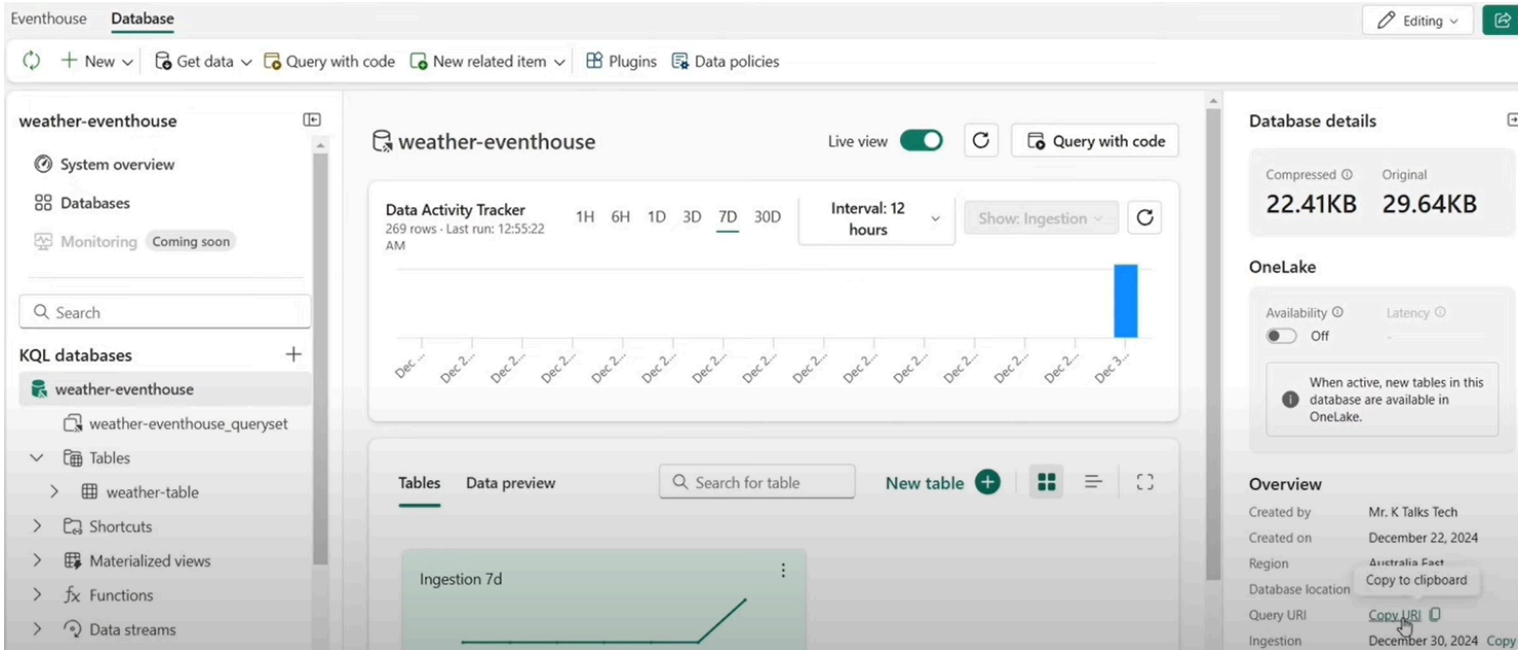
Now if you have to connect your kkl database with the power via desktop then here are the procedures.

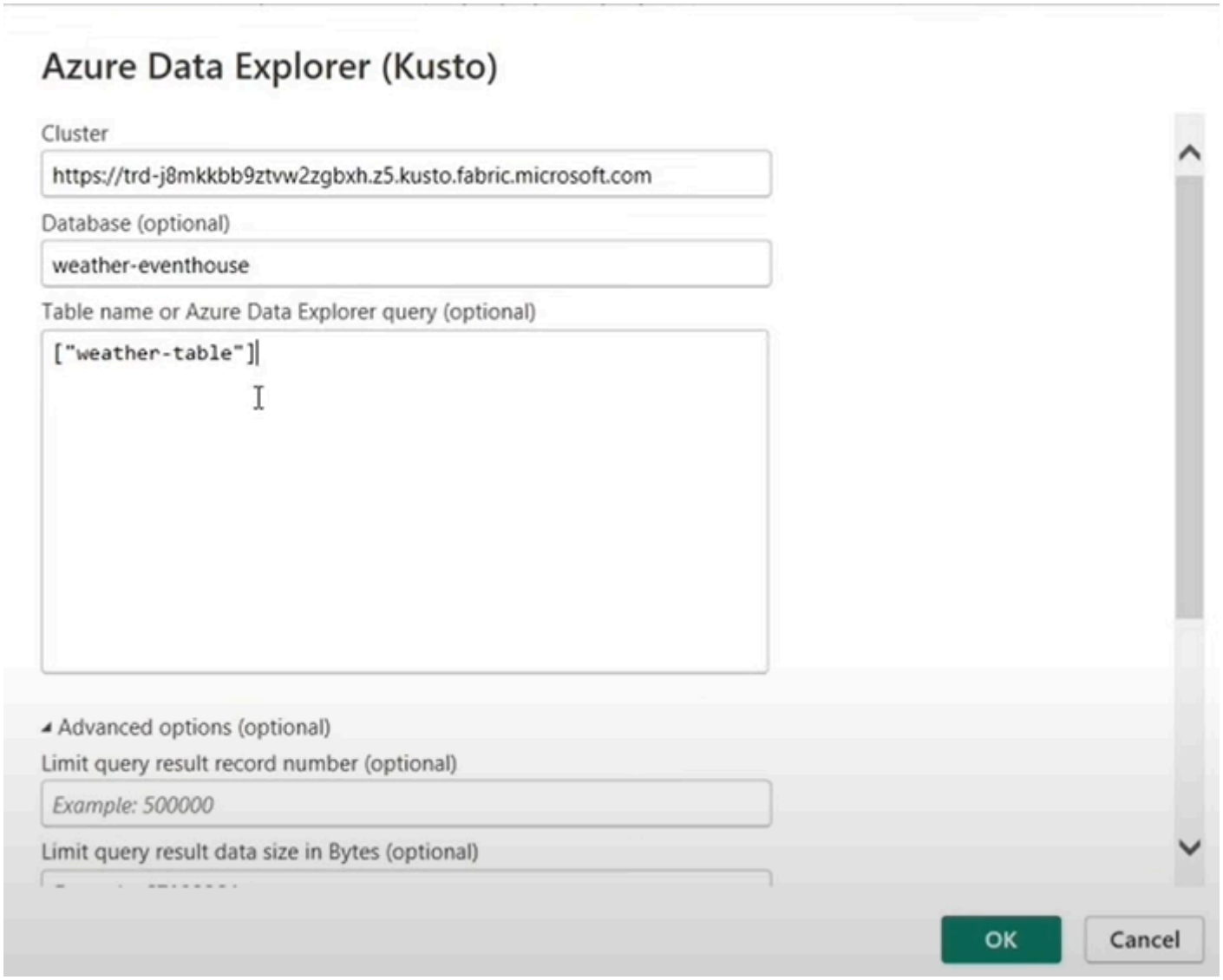
Powerbi desktop open it.



Now we need to connect our kq database with the power bi desktop so you need to go to the event house where you have the KQA database and in that there is the query URI copy that query URI and paste that in

the connect data option of the KQL database in the power BI desktop





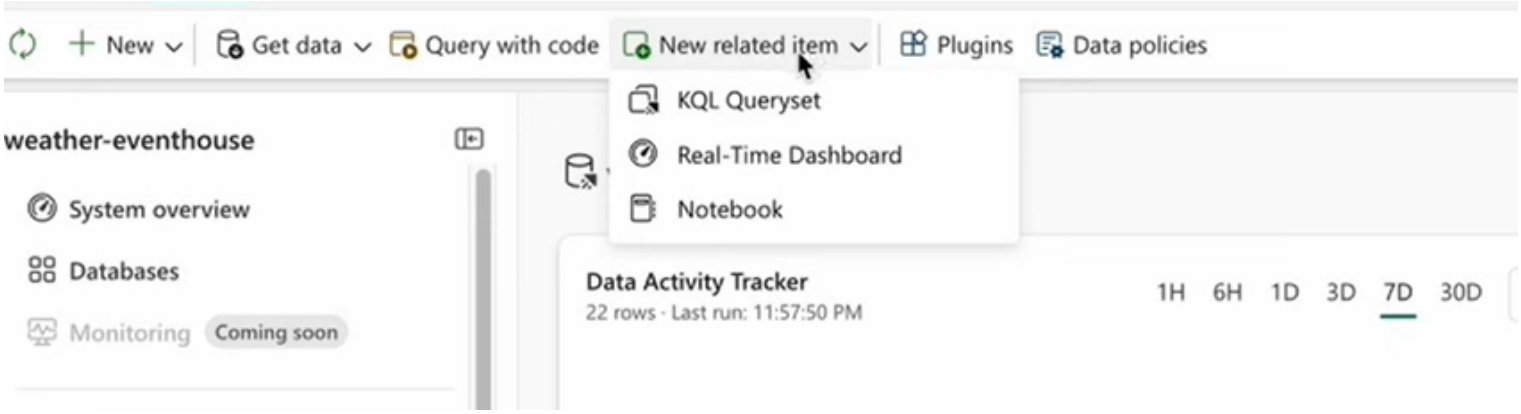
Here you can't schedule the refresh here you need to click on the refresh button then only you will able to see the recent data.

For scheduling manual refresh you can Schedule data refresh manually for 48 times a day in Microsoft Fabric account on the semantic model

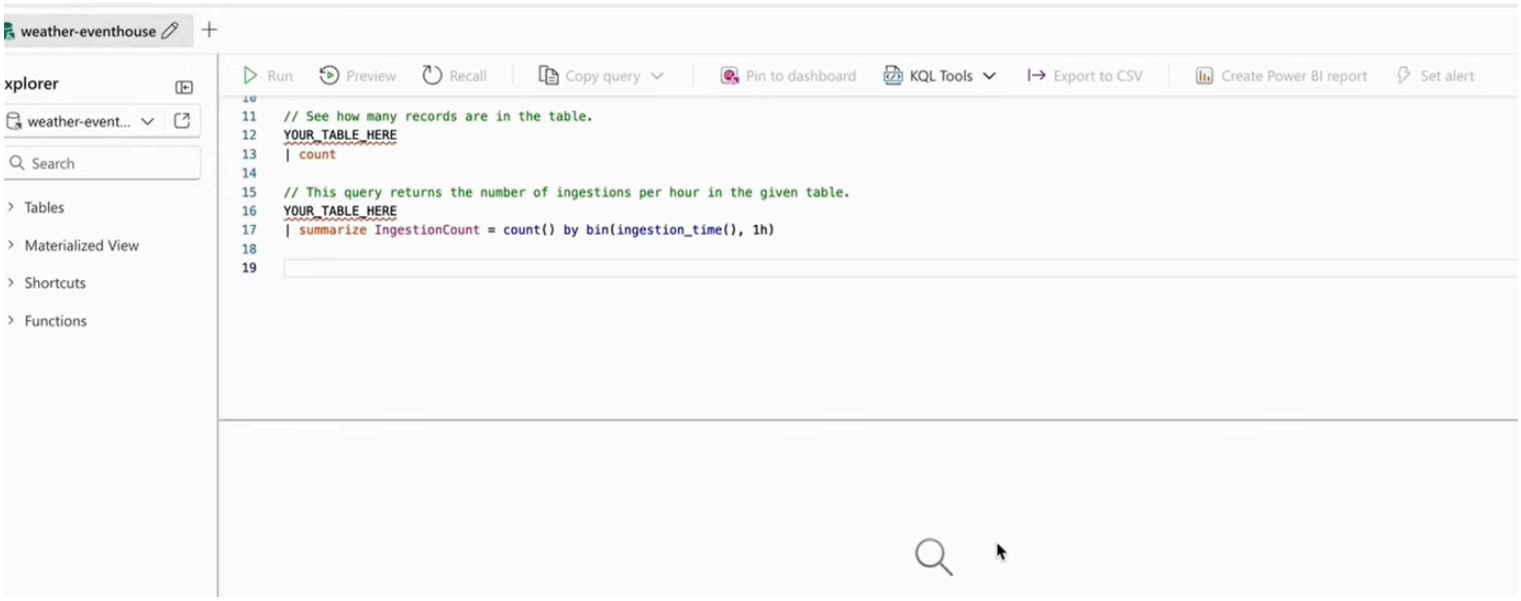
Now configuring alerts::

Firstly in the kkr data base you have to create a new query set and there we have to write the logic for what needs to be pushed in the alerts

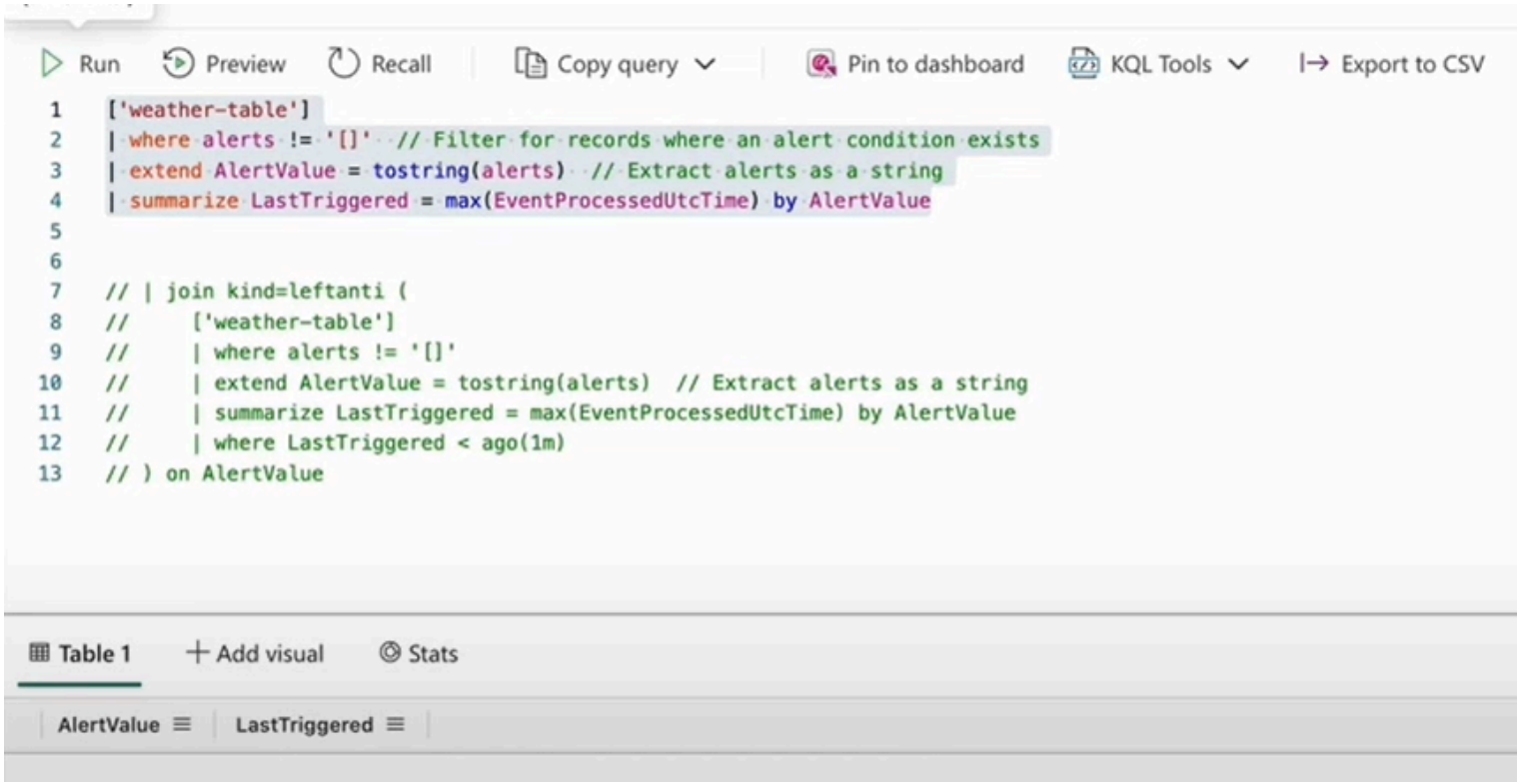
.



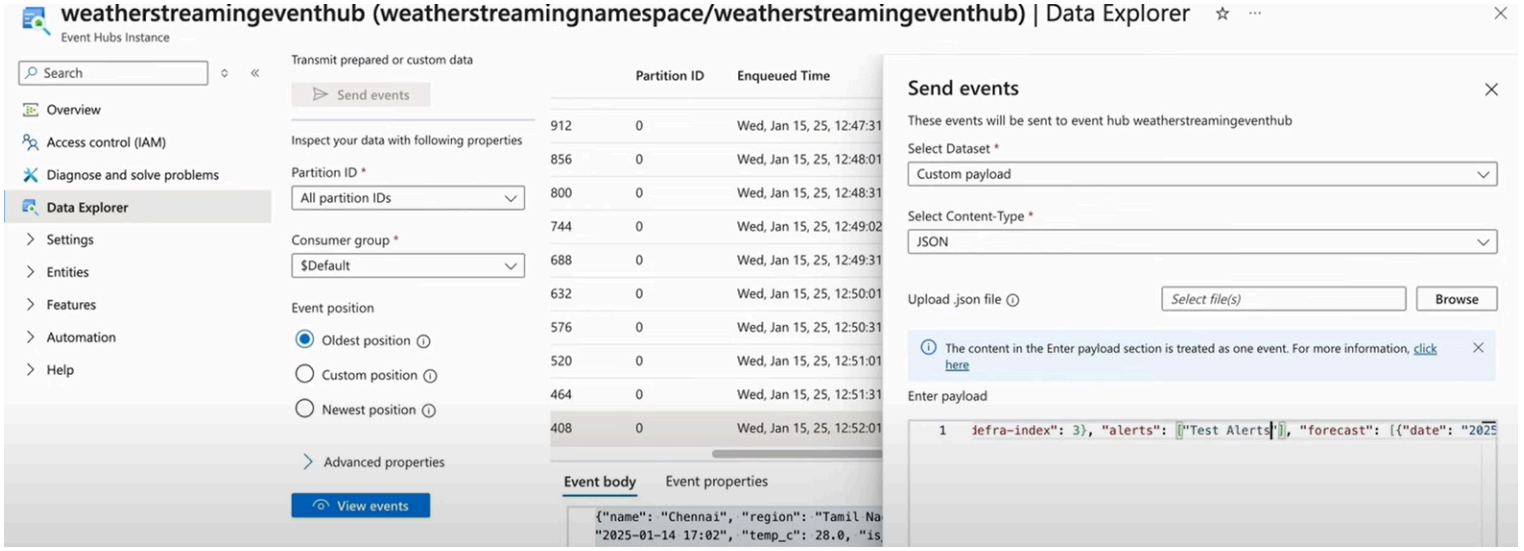
Below is the newly created query set in same kql database



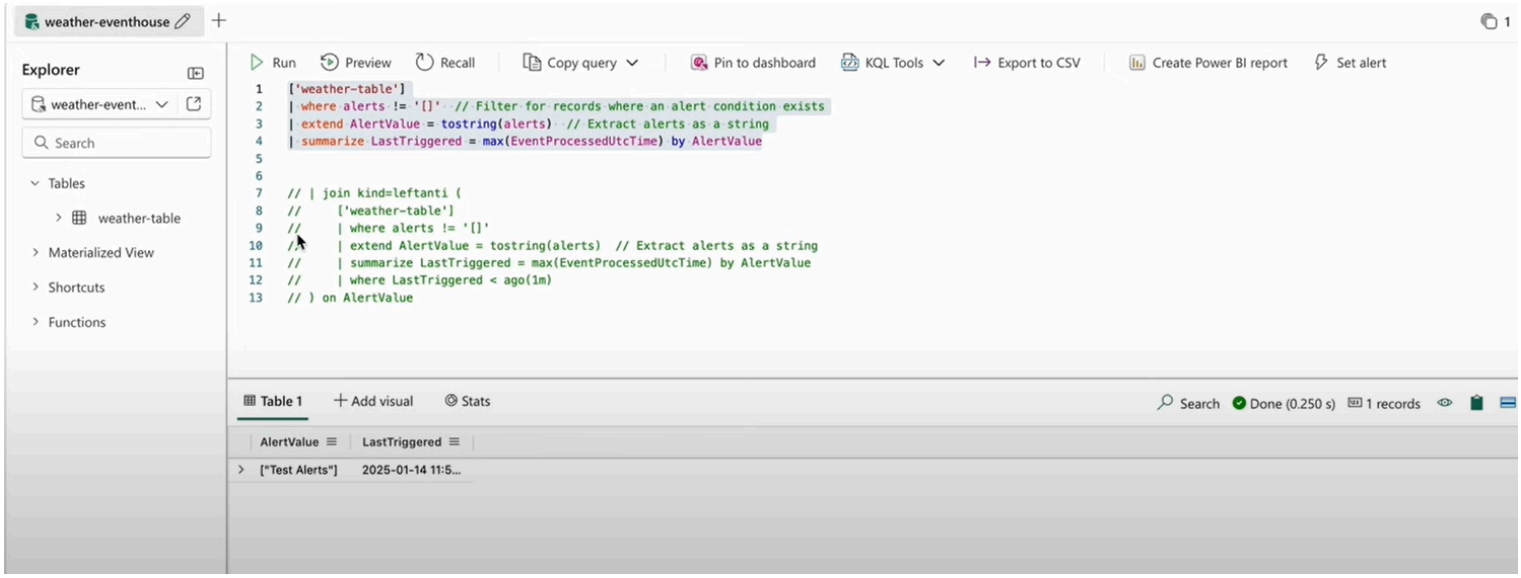
Now in this database it is a column called alerts if there is any message coming from the api in the alerts column that means there is some alert so we will track the alert now in this query set.



So there is a sand events button where if you have to explicitly send event in the event hub so there you can explicitly catch any record and put a custom payload with some alert for any record and click on send

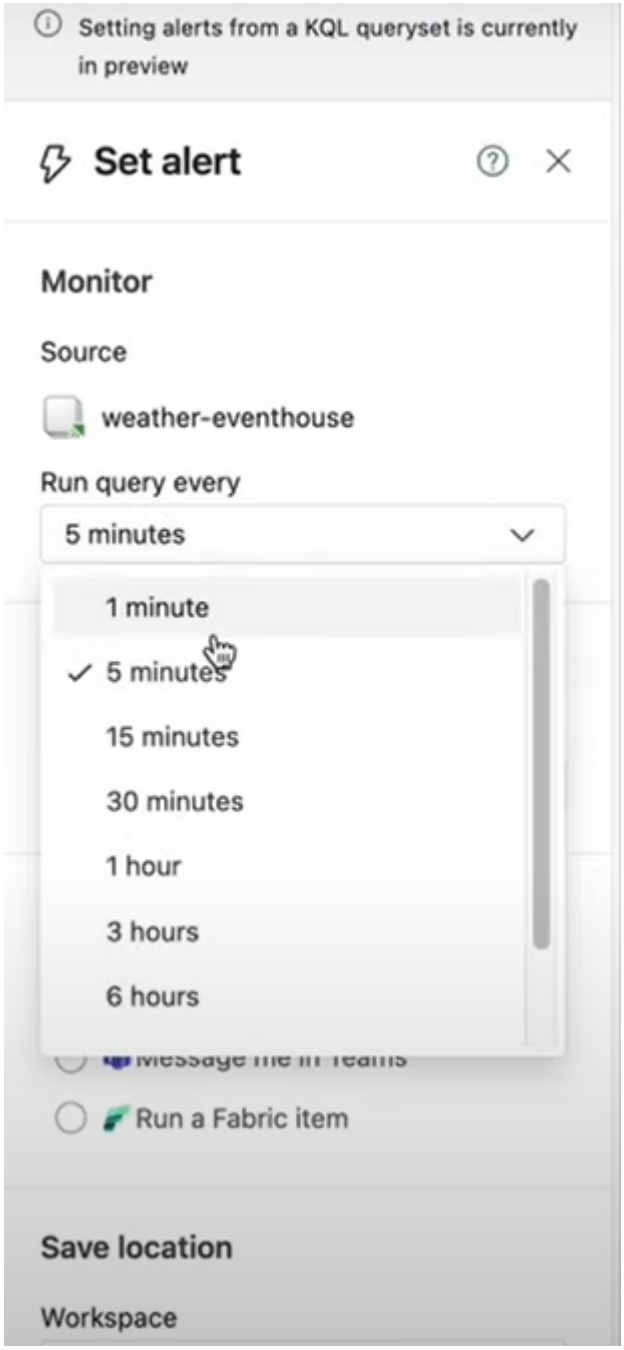


You're in the kql database you will see that the query captured that alert and now you are able to see that in the

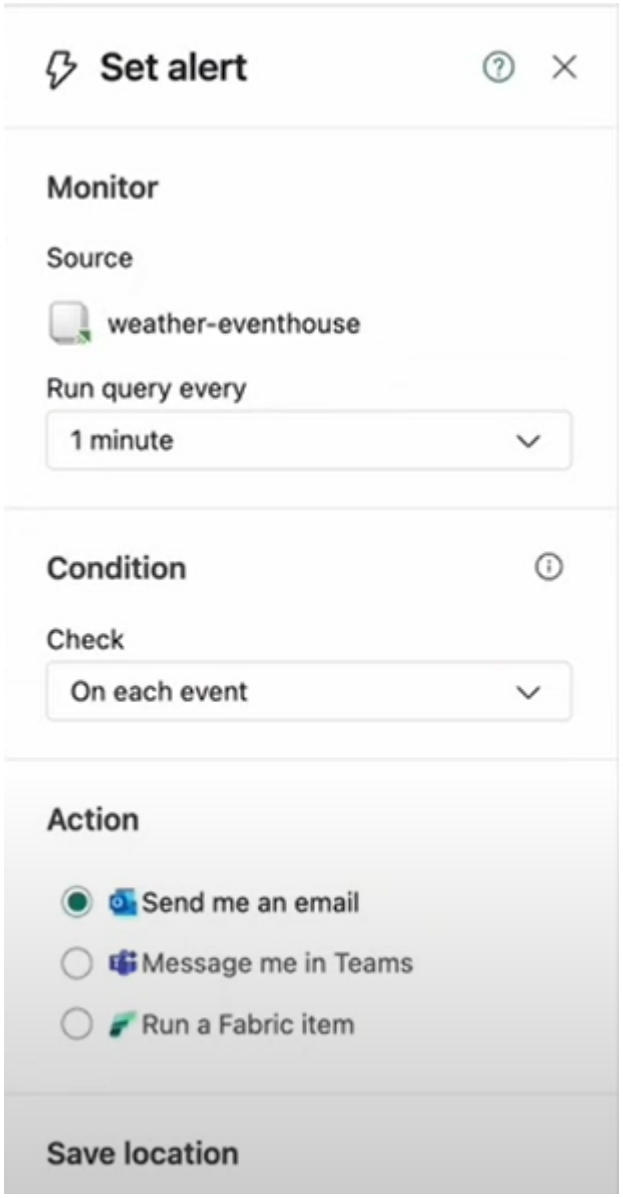


In the KQL query set which we created the 2nd one we have option to run every query and automate that query and there are options to run every query for one minute 5 15 minute so basically if you want to run any query in the query set you can set an automator for that query so that it executes every five minutes 1 minute 10 minute whatever you want

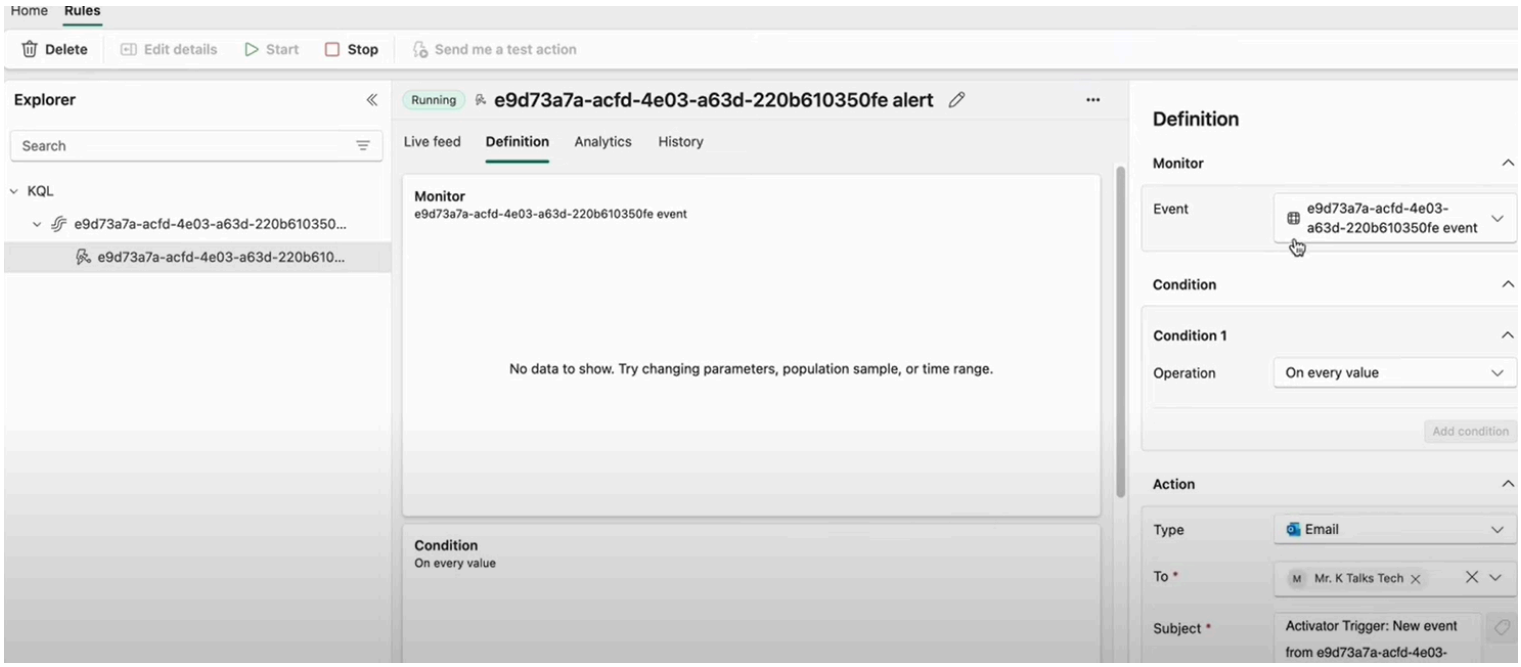
.



There is signed email option in the set alert in the KQL will query set



When you create an alert automatically creates a data activator where you can see what has triggered that query and what is happening and like the monitoring feature in Azure you see the logs and various visualizations for that alert which you have set



Event activity has the function of type the content of the email and configure it