

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:

The screenshot shows the Azure Event Hubs Overview page for a resource named 'weatherstreamingeventhub'. The page includes a search bar, navigation links for consumer groups, delete, refresh, and feedback, and a message about generating test data. The main content area displays the 'Essentials' section with details: Resource group (move) : rg-weather-streaming, Status : Active, Location : Australia East, Namespace : weatherstreamingnamespace. A JSON link is also present. On the left sidebar, 'Data Explorer (preview)' is selected, along with other options like Access control (IAM), Diagnose and solve problems, and Container.

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.

The screenshot shows the Azure Data Explorer interface with the 'Event body' tab selected. The content pane displays the JSON event body: {"event_id": 1111, "event_name": "Test Event"}. The interface includes a search bar and navigation buttons for other tabs like 'Event properties'.

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.

The screenshot shows the 'Create Secret Scope' dialog in the Azure Databricks interface. The left sidebar shows various workspace options like Workspace, Recents, Catalog, Workflows, Compute, SQL, and others. The main dialog title is 'Create Secret Scope' with 'Cancel' and 'Create' buttons. It explains what a secret scope is: 'A store for secrets that is identified by a name and backed by a specific store type.' Below this, there's a 'Scope Name' input field containing 'key-vault-scope', a 'Manage Principal' dropdown set to 'Creator', and sections for 'Azure Key Vault' (with 'DNS Name' as 'https://kv-weather-streaming.vault.azure.net/' and 'Resource ID' as '/subscriptions/841031b2-72a5-4f94-8084-ecf13b4e0cf6/resourceGroups/rg-v').

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

The screenshot shows the 'Add role assignment' page in the Azure portal. In the top navigation bar, it says 'Home > rg-weather-streaming > kv-weather-streaming | Access control (IAM) > Add role assignment ...'. The 'Members' tab is selected. A search bar at the top right contains the text 'azuredatabricks'. Below the search bar, a result for 'AzureDatabricks Application' is shown with a blue folder icon.

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

The screenshot shows the 'Data Explorer (preview)' section of the Azure Event Hubs interface. On the left, there's a sidebar with options like Overview, Access control (IAM), and Data Explorer (preview). The main area shows an 'Event Hub' named 'weatherstreamingeventhub'. It displays two received events:

Partition ID	Enqueued Time	Content Type	Message ID	Event Body
0	Sat, Oct 26, 24, 02:24:55 AM GM...			{"event_id": 1111, "event_name": "Test Eve..."}
0	Sat, Oct 26, 24, 02:49:08 AM GM...			{"event_id": 2222, "event_name": "Key Vau..."}

6. Weather api testing in DB.

Go in azure notebook

The screenshot shows an Azure Notebook cell with the following Python code:

```
import requests
import json

# Getting secret value from Key Vault
weatherapikey = dbutils.secrets.get(scope="key-vault-scope", key="weatherapikey")
location = "Chennai" # You can replace with city name based on your preference

base_url = "http://api.weatherapi.com/v1/"
current_weather_url = f"{base_url}/current.json"

params = {
    'key': weatherapikey,
    'q': location,
}
response = requests.get(current_weather_url, params=params)

if response.status_code == 200:
    current_weather = response.json()
    print("Current Weather:")
    print(json.dumps(current_weather, indent=3))
else:
    print(f"Error: {response.status_code}, {response.text}")
```

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 databricks(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.

The screenshot shows the Azure Event Hub 'Send events' interface. On the left, there's a sidebar with options like Overview, Access control (IAM), Diagnose and solve problems, Data Explorer (preview) (which is selected), Settings, Entities, Features, Automation, Help, and a Help section. The main area has a search bar and a 'Transmit prepared or custom data' button. Below it, a table lists received events: Partition ID (7296), Enqueued Time (Sun, Nov 10, 24, 12:29:48 AM G...), Content Type (Message ID), and Event Body ({"name": "Chennai", "region": "Tamil Nadu"}). The 'Event body' column shows the JSON response from the weather API. At the bottom, there are tabs for 'Event body' and 'Event properties', and a 'View events' button.

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.

```

# 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.

fp-weather-streaming Function App

Search | Browse | Refresh | Stop | Restart | Swap | Get publish profile | Reset publish profile | Download app content | Delete | Send us your feedback

Overview

Resource group (move) : rg-weather-streaming
Status : Running
Location (move) : Australia East
Subscription (move) : Azure subscription 1
Subscription ID : 841031b2-72a5-4f94-8084-ecf13b4e0cf6
Tags (edit) : Add tags

Default domain : fp-weather-streaming.azurewebsites.net
Operating System : Linux
App Service Plan : ASP-rgweatherstreaming-b16a(Y1:0)
Runtime version : 4.1036.2.2

Functions Metrics Properties Notifications (0)

Create functions in your preferred environment

- Create in Azure portal
- VS Code Desktop
- Other editors or CLI

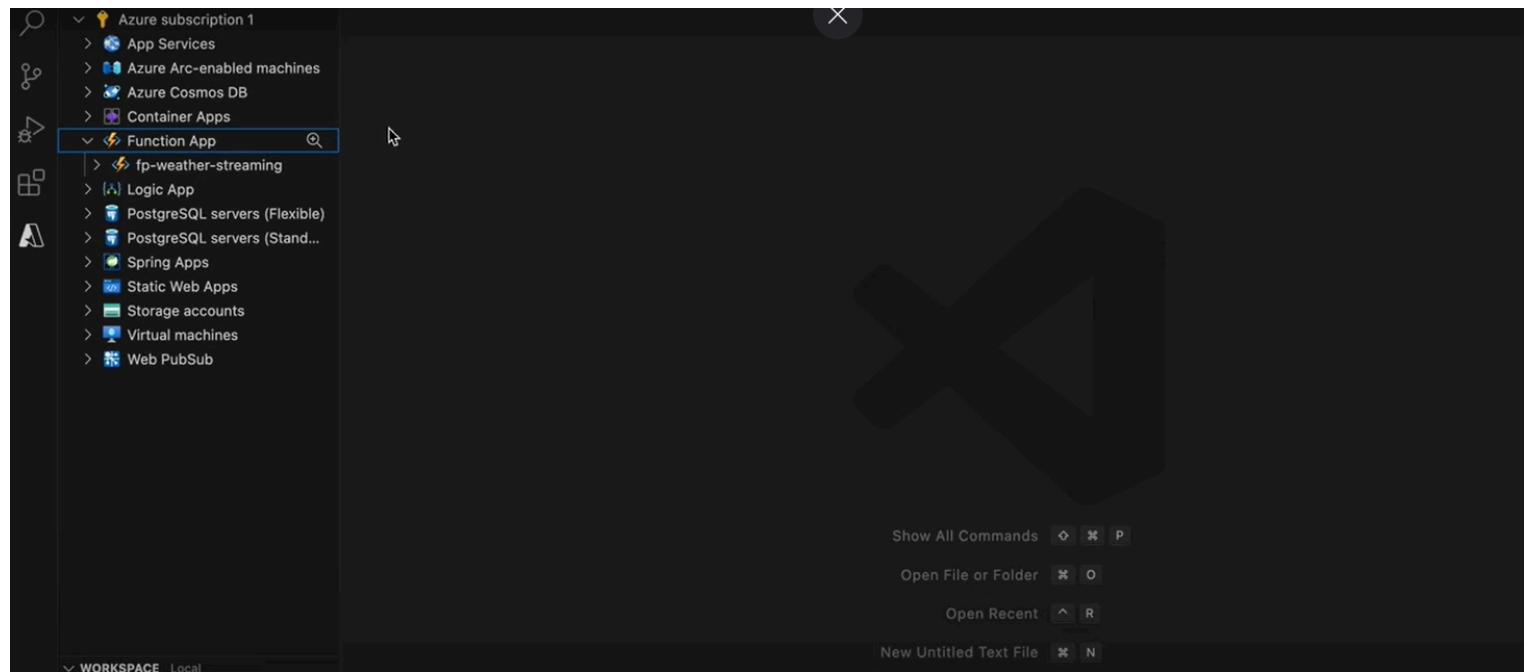
Best optimized for:

- Getting started without local setup
- Choose from our Function templates
- Local development within VS Code
- Custom development tool requirements
- Using preferred editor for development
- Visual Studio, IntelliJ, command line

Create function | Create with VS Code Desktop | Set up your editor

Same like AWS lambda.

Here you can write code in VS-code. -> in vs code extensions -> azure functions. -> sign in to azure through vscode. Note that 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.

```
1 import logging
2 import azure.functions as func
3
4 app = func.FunctionApp()
5
6 @app.timer_trigger(schedule="*/30 * * * *", arg_name="myTimer", run_on_startup=False,
7                     use_monitor=False)
8 def weatherapifunction(myTimer: func.TimerRequest) -> None:
9     if myTimer.past_due:
10         logging.info('The timer is past due!')
11
12     logging.info('Python timer trigger function executed.')
```

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

Add role assignment

Role Members * Conditions Review + assign

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. [Learn more](#)

Job function roles Privileged administrator roles

Grant access to Azure resources based on job function, such as the ability to create virtual machines.

Name ↑↓	Description ↑↓	Type ↑↓	Category ↑↓	Details
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General	View
App Compliance Automation Administrator	Create, read, download, modify and delete reports objects and related other resource objects.	BuiltInRole	None	View
App Compliance Automation Reader	Read, download the reports objects and related other resource objects.	BuiltInRole	None	View
Azure Event Hubs Data Owner	Allows for full access to Azure Event Hubs resources.	BuiltInRole	Analytics	View
Azure Event Hubs Data Receiver	Allows receive access to Azure Event Hubs resources.	BuiltInRole	Analytics	View
Azure Event Hubs Data Sender	Allows send access to Azure Event Hubs resources.	BuiltInRole	Analytics	View
Log Analytics Contributor	Log Analytics Contributor can read all monitoring data and edit monitoring settings. Editing monitoring settings includes...	BuiltInRole	Analytics	View
Log Analytics Reader	Log Analytics Reader can view and search all monitoring data as well as and view monitoring settings, including viewing ...	BuiltInRole	Analytics	View
Managed Application Contributor Role	Allows for creating managed application resources.	BuiltInRole	Management + Gover...	View
Managed Application Operator Role	Lets you read and perform actions on Managed Application resources	BuiltInRole	Management + Gover...	View

Go to IAM and create managed identity:

fp-weather-streaming | Identity

System assigned User assigned

Status: On

Object (principal) ID: fc75a6b8-7583-42c5-8c33-0dfa224901f7

Permissions: Azure role assignments

This resource is registered with Microsoft Entra ID. The managed identity can be configured to allow access to other resources. Be careful when making changes to the access settings for the managed identity because it can result in failures. [Learn more](#)

Add role assignment

Selected role: Azure Event Hubs Data Sender

Assign access to: Managed identity

Members: No members selected

Description: Optional

Select managed identities

Subscription: Azure subscription 1

Managed identity: Function App (1)

Search by name

Selected members: No members selected. Search for and add one or more members you want to assign to the role for this resource.

Learn more about RBAC

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.eventhub import EventHubProducerClient, EventData
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()

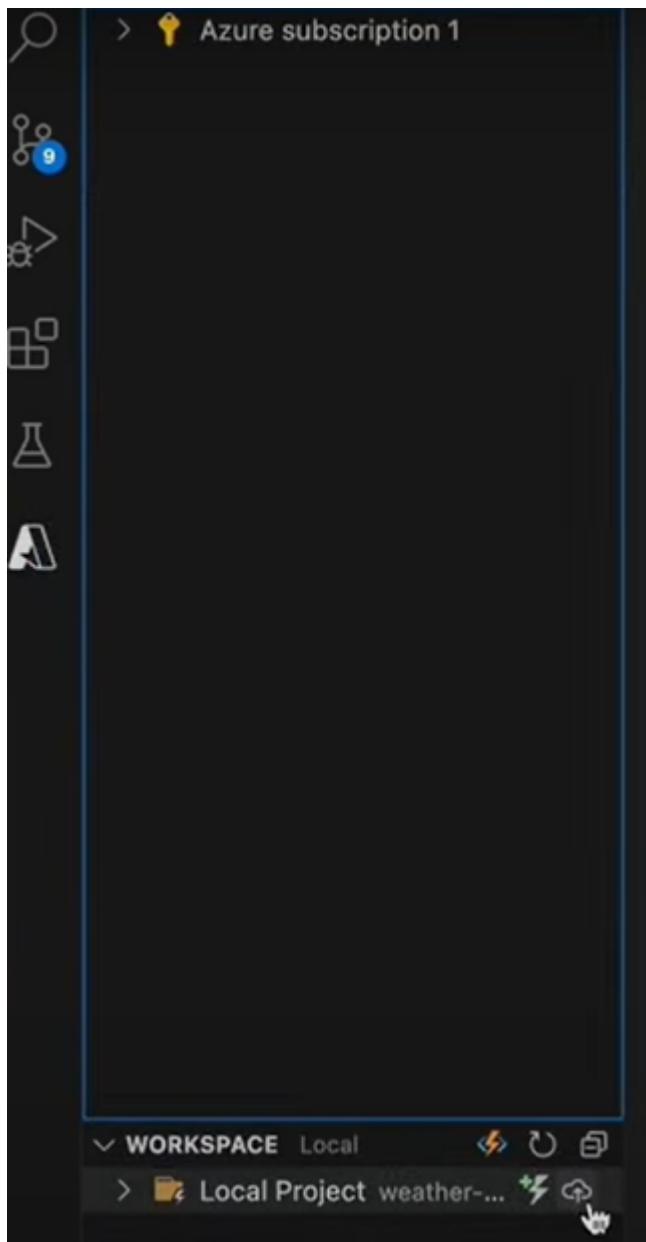
# Initialize the Event Hub producer
producer = EventHubProducerClient(
    fully_qualified_namespace=EVENT_HUB_NAMESPACE,
    eventhub_name=EVENT_HUB_NAME,
    credential=credential
)
```

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:

fp-weather-streaming

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Events (preview)

Recommended services (preview)

Log stream

Functions

App keys

App files

Proxies

Deployment

Resource group (move) : rg-weather-streaming

Status : Running

Location (move) : Australia East

Subscription (move) : Azure subscription 1

Subscription ID : 841031b2-72a5-4f94-8084-ecf13b4e0cf6

Tags (edit) : Add tags

Default domain : fp-weather-streaming.azurewebsites.net

Operating System : Linux

App Service Plan : ASP-rgweatherstreaming-b16a (Y1: 0)

Runtime version : 4.1036.2.2

weatherapifunction

Name : weatherapifunction

Trigger : Timer

Status : Enabled

Monitor : Invocations and more

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

Pricing calculator like amazon in Azure

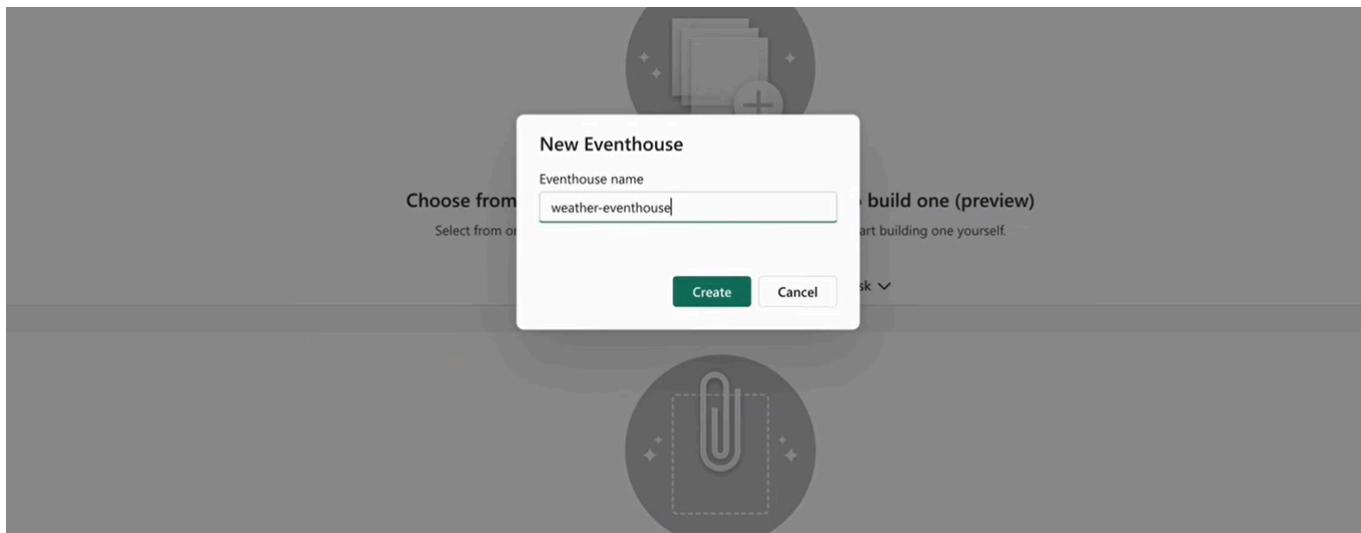
Now pushing data from Event hub to fabric database.

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:

The screenshot shows the Microsoft Fabric workspace interface. At the top, there are buttons for 'New item', 'New folder', 'Upload', and a search bar containing 'eventstream'. Below the search bar, a modal window titled 'Select an item type' is open, showing the 'Eventstream' option under the 'Get data' category. The 'Eventstream' section is described as 'Capture, transform, and route real-time event stream to various destinations in desired format with no-code experience.' On the left, a sidebar lists existing resources: 'weather-eventhouse' (Eventhouse), 'weather-eventhouse' (KQL Database), and 'weather-eventhouse_queryset' (KQL Queryset). A 'Choose from previous' dropdown is also visible.

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

The screenshot shows the 'Configure' step of the event stream creation process. The main title is 'Connect data source' with a 'Connect' button. Below it, a connection is shown between 'Azure Event Hubs' (New event hub connection) and 'weather-eventstream' (Eventstream). The 'Configure connection settings' section includes a dropdown for 'Connection' (set to 'No connection found using selected source type') and a link to 'New connection'. The 'Configure Azure Event Hub data source' section includes a 'Consumer group' field set to '\$Default' with a note: 'Please enter an existing consumer group name or "\$Default" to use the default consumer group.' The 'Data format' field is set to 'Json'. On the right, the 'Stream details' section shows the 'Eventstream name' as 'weather-eventstream', 'Stream name' as 'weather-eventstream-stream', and 'Source name' as 'AzureEventHub'. There is also a 'What is this?' link.

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

Policy

Claims

fordatabricks

Send

forfabric

Listen

Configure Review + connect

Connect data source Connect

Azure Event Hubs → weather-eventstream

New event hub connection

Eventstream

Connection credentials

Connection Create new connection

Check the configuration

Connection name {"endpoint": "weatherstreamingnamespace", "entityPath": ...}

Authentication kind Shared Access Key

Shared Access Key Name forfabric

Shared Access Key|

Consumer group \$Default

Please enter the default consumer group name.

Data format Json

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 → weather-eventstream

weatherstreamingnamespace/...

Eventstream

Configure connection settings

Check the following document for field requirements. [Learn more](#)

Connection *

{"endpoint": "weatherstreamingnamespace", "entity..."} 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

Back Next

Stream details

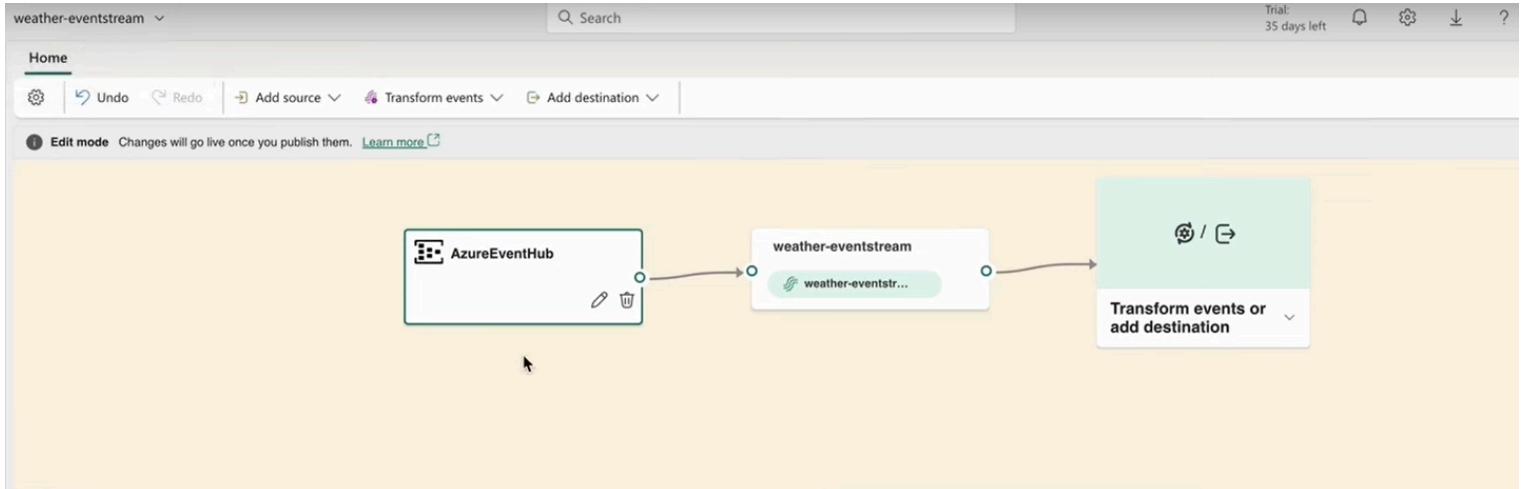
Eventstream name weather-eventstream

Stream name weather-eventstream-stream

Source name AzureEventHub

What is this?

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

This screenshot shows the configuration of a target destination. On the left, there's a "Test result" section with a table of data and an "Authoring errors" section. The table has columns: name, region, country, lat, lon, and localtime. The data shows multiple rows for Chennai, Tamil Nadu, India, with coordinates (13.0833, 80.2833) and timestamp 2024-12-22T11:47:05Z. Below the table, it says "Data format: Json" and "Time range: 12/22/24 11:47:05 PM - 12/23/24 12:47:05 AM". On the right, a large context menu is open over the "Transform events or add destination" button. The menu is titled "Transform events or add destination" and contains sections for "Operations" (Aggregate, Expand, Filter, Group by, Join, Manage fields, Union), "Destinations" (Custom endpoint, Lakehouse, Eventhouse, Activator), and a "Last hour" dropdown. The "Destinations" section is currently active, showing options like "Custom endpoint", "Lakehouse", "Eventhouse", and "Activator".

The screenshot shows the Eventhouse configuration interface. At the top left is a summary card with the title "Eventhouse" and a warning icon "Set-up required". Below the card are three icons: a pencil for edit, a plus sign for add, and a trash can for delete. To the right of the card is a vertical toolbar with a plus sign at the top, followed by a minus sign and a zoom-in/crop icon. At the bottom of the interface is a timeline navigation bar with the text "Last hour" and a refresh button.

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

The screenshot shows the Eventhouse configuration interface. On the left, there is a summary card for "Eventhouse" with a warning icon and the message "Set-up required". Below it is a timeline with "Last hour" and a "Refresh" button. To the right is a detailed configuration panel:

- Data ingestion mode ***: Event processing before ingestion (selected)
- Destination name ***: weather-target
- Workspace ***: weather-streaming
- Eventhouse ***: weather-eventhouse
- KQL Database ***: weather-eventhouse
- KQL Destination table ***: (New) weather-table
- Create new**
- Input data format ***: Json

Publish once done.

The screenshot shows the Azure Stream Analytics studio. The top navigation bar includes "Home", "Edit mode" (highlighted), "Undo", "Redo", "Add source", "Transform events", "Add destination", and a "Publish" button. The main area displays a pipeline diagram with three stages: "AzureEventHub" (source), "weather-eventstream" (transform), and "weather-target" (sink). Below the diagram is a "Test result" section showing a preview of the output data:

name	region	country	lat	lon	localtime	temp_c	is_day
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1

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

The screenshot shows the Microsoft Fabric KQL interface. On the left, there's a sidebar with 'weather-eventhouse' selected. Under 'KQL databases', 'weather-eventhouse_queryset' is highlighted. The main area shows a query editor with the following KQL code:

```
1 // Use 'take' to view a sample number of records in the table and check the data.
2 ['weather-table']
3 | take 100
4
```

Below the code, a table titled 'Table 1' displays the results of the query. The columns are: name, region, country, lat, lon, localtime, temp_c, is_day, condition_text, condition_icon, and wind_kph. The data shows five rows for Chennai, India, with coordinates 13.0833, 80.2833, and localtime 2024-12-22 17:24. The temperature ranges from 26.4 to 27.2 degrees Celsius, and the wind speed is between 20.2 and 21.6 kph.

name	region	country	lat	lon	localtime	temp_c	is_day	condition_text	condition_icon	wind_kph
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1	Patchy rain nearby	cdn.weatherapi.com/weather/64x64/day/176.png	20.2
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1	Patchy rain nearby	cdn.weatherapi.com/weather/64x64/day/176.png	20.2
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1	Patchy rain nearby	cdn.weatherapi.com/weather/64x64/day/176.png	20.2
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 16:58	27.2	1	Sunny	cdn.weatherapi.com/weather/64x64/day/113.png	21.6
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 16:58	27.2	1	Sunny	cdn.weatherapi.com/weather/64x64/day/113.png	21.6

– 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.

```

Run Preview Recall Copy query Pin to dashboard KQL Tools Export to CSV Create Power BI report Set alert
1 // Use 'take' to view a sample number of records in the table and check the data.
2 ['weather-table']
3 | take 100
4
5
6

```

Table 1 + Add visual Stats

Search Done (0.109 s) 100 records

name	region	country	lat	lon	localtime	temp_c	is_day	condition_text	condition_icon	wind_kph
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1	Patchy rain nearby	cdn.weatherapi.com/weather/64x64/day/176.png	20.2
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1	Patchy rain nearby	cdn.weatherapi.com/weather/64x64/day/176.png	20.2
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 17:24	26.4	1	Patchy rain nearby	cdn.weatherapi.com/weather/64x64/day/176.png	20.2
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 16:58	27.2	1	Sunny	cdn.weatherapi.com/weather/64x64/day/113.png	21.6
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 16:58	27.2	1	Sunny	cdn.weatherapi.com/weather/64x64/day/113.png	21.6
Chennai	Tamil Nadu	India	13.0833	80.2833	2024-12-22 16:58	27.2	1	Sunny	cdn.weatherapi.com/weather/64x64/day/113.png	21.6

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.

```

-- explain
select max(EventProcessedUtcTime) from "weather-table"

```

Table 1 + Add visual Stats

Query

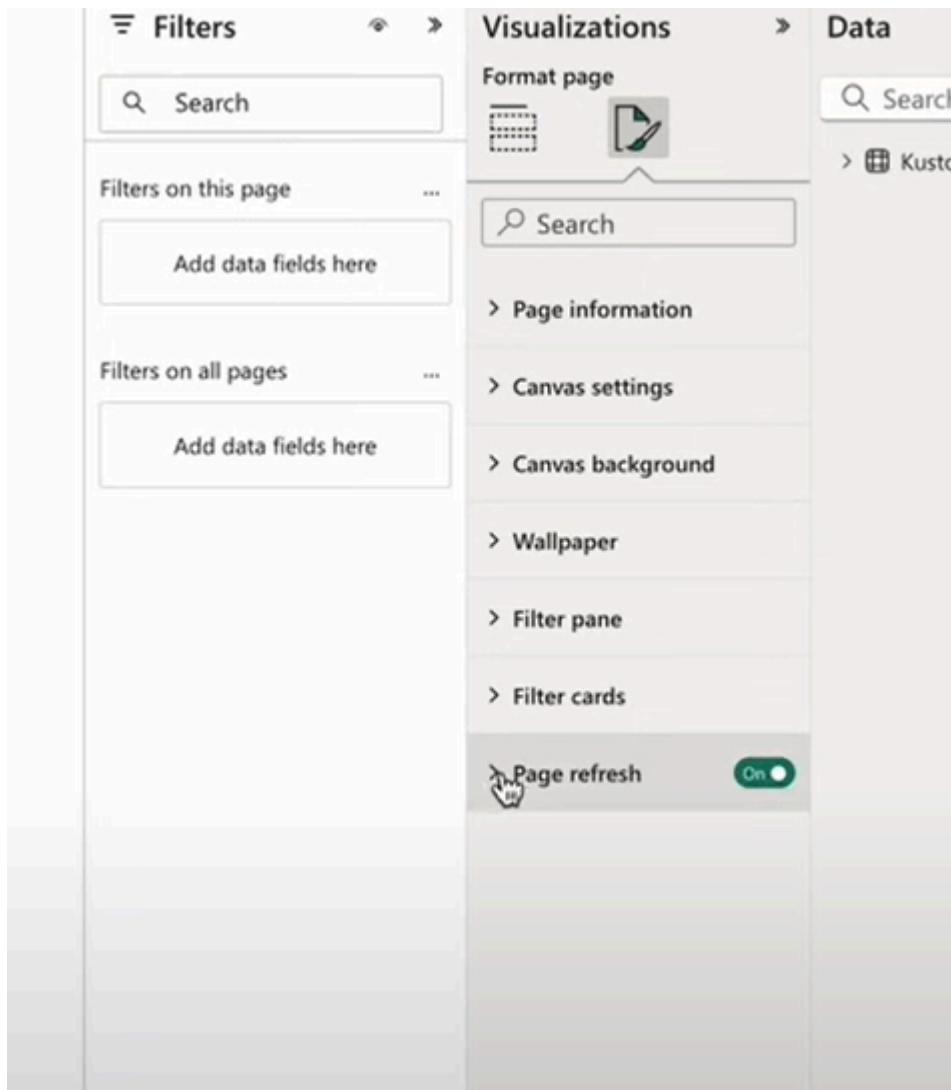
```
[weather-table] | summarize NoColumnName1=max(EventProcessedUtcTime) | project NoColumnName1
```

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.

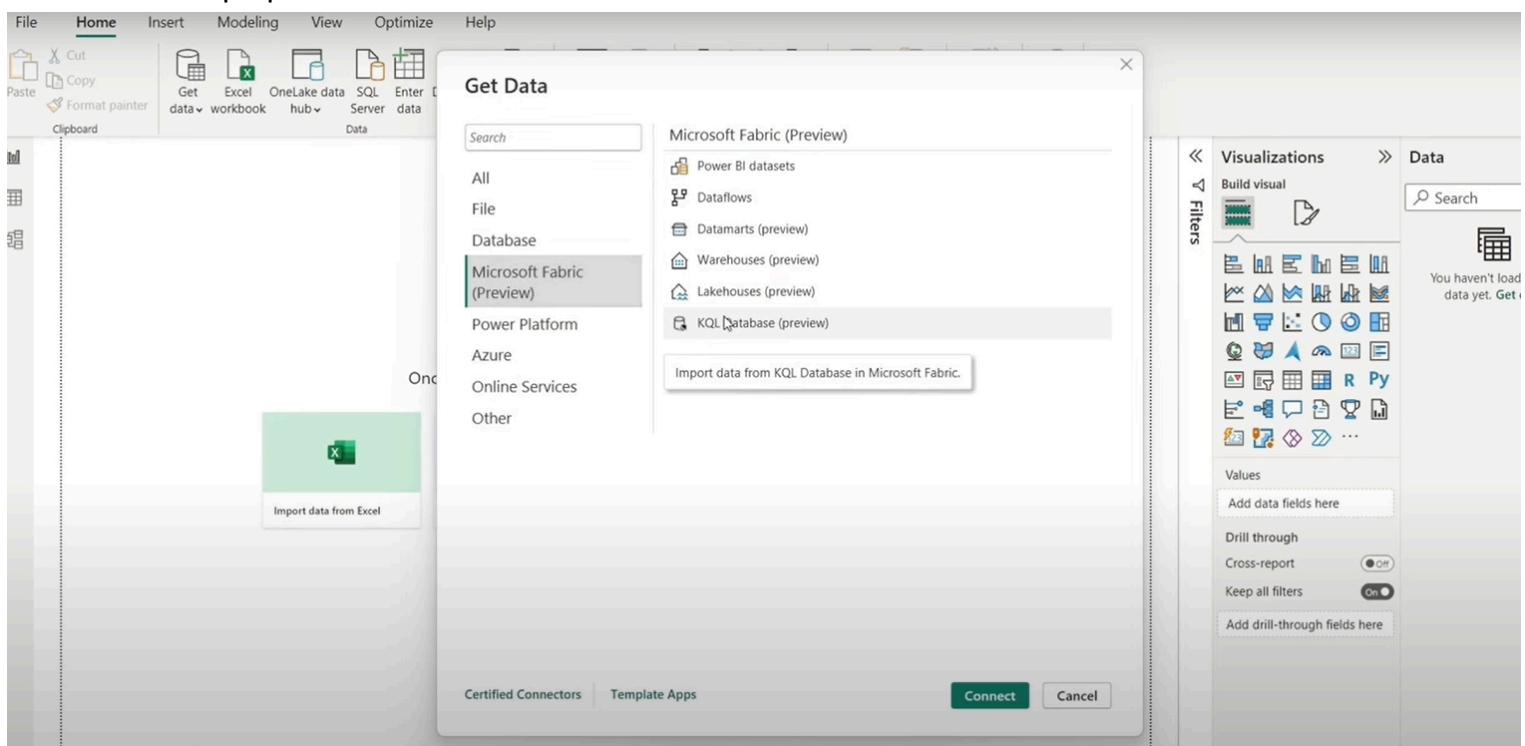
Most important thing about the Arabia report which gets generated is whenever the data is changed from the API it automatically flows through the pipeline which we created and it updates the power BR report automatically so whenever the API data changes for the weather the power BI report because of 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 kql database with the power via desktop then here are the procedures.

Powerbi desktop open it.



Now we need to connect our kql 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

The screenshot shows the Power BI Desktop interface with the 'weather-eventhouse' dataset selected. The left sidebar displays the KQL databases, and the main area shows the Data Activity Tracker and a table preview. The 'OneLake' section indicates availability is off, and the 'Overview' section provides details about the database creation.

Azure Data Explorer (Kusto)

Cluster
https://trd-j8mkkbb9ztvw2zgbxh.z5.kusto.fabric.microsoft.com

Database (optional)
weather-eventhouse

Table name or Azure Data Explorer query (optional)
["weather-table"]

Advanced options (optional)

Limit query result record number (optional)
Example: 500000

Limit query result data size in Bytes (optional)

OK Cancel

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

The screenshot shows the Microsoft Fabric interface for the 'weather-eventhouse' database. At the top, there are several navigation icons: a green circle, '+ New', 'Get data', 'Query with code', 'New related item' (which is currently selected and has a dropdown menu), 'Plugins', and 'Data policies'. Below the navigation bar, the database name 'weather-eventhouse' is displayed, along with links for 'System overview', 'Databases', and 'Monitoring' (with a note 'Coming soon'). A central panel displays a 'Data Activity Tracker' with 22 rows, last run at 11:57:50 PM. To the right of the tracker are time range filters: '1H', '6H', '1D', '3D', '7D' (which is selected and highlighted in green), and '30D'. A dropdown menu is open over the 'New related item' icon, listing 'KQL Queryset', 'Real-Time Dashboard', and 'Notebook'.

Below is the newly created query set in same kql database

The screenshot shows the Microsoft Fabric KQL editor for the 'weather-eventhouse' database. On the left, the 'explorer' sidebar lists 'Tables', 'Materialized View', 'Shortcuts', and 'Functions'. The main area contains a KQL query:11 // See how many records are in the table.
12 YOUR_TABLE_HERE
13 | count
14
15 // This query returns the number of ingestions per hour in the given table.
16 YOUR_TABLE_HERE
17 | summarize IngestionCount = count() by bin(ingestion_time(), 1h)
18
19A toolbar at the top includes 'Run', 'Preview', 'Recall', 'Copy query', 'Pin to dashboard', 'KQL Tools', 'Export to CSV', 'Create Power BI report', and 'Set alert'. A search bar is located at the bottom right.

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.

Run Preview Recall Copy query Pin to dashboard KQL Tools Export to CSV

```

1  ['weather-table']
2  | where alerts != [] // Filter for records where an alert condition exists
3  | extend AlertValue = tostring(alerts) // Extract alerts as a string
4  | summarize LastTriggered = max(EventProcessedUtcTime) by AlertValue
5
6
7  // | join kind=leftanti (
8  //   ['weather-table']
9  //   | where alerts != []
10 //    | extend AlertValue = tostring(alerts) // Extract alerts as a string
11 //    | summarize LastTriggered = max(EventProcessedUtcTime) by AlertValue
12 //    | where LastTriggered < ago(1m)
13 // ) on AlertValue

```

Table 1 + Add visual Stats

AlertValue LastTriggered

So there is a send 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

weatherstreamingeventhub (weatherstreamingnamespace/weatherstreamingeventhub) | Data Explorer

Partition ID	Enqueued Time
912	0 Wed, Jan 15, 25, 12:47:31
856	0 Wed, Jan 15, 25, 12:48:01
800	0 Wed, Jan 15, 25, 12:48:31
744	0 Wed, Jan 15, 25, 12:49:02
688	0 Wed, Jan 15, 25, 12:49:31
632	0 Wed, Jan 15, 25, 12:50:01
576	0 Wed, Jan 15, 25, 12:50:31
520	0 Wed, Jan 15, 25, 12:51:01
464	0 Wed, Jan 15, 25, 12:51:31
408	0 Wed, Jan 15, 25, 12:52:01

Send events

These events will be sent to event hub weatherstreamingeventhub

Select Dataset * Custom payload

Select Content-Type * JSON

Upload json file (Select file(s) Browse)

The content in the Enter payload section is treated as one event. For more information, click here.

Enter payload

```

1  { "efra-index": 3 }, "alerts": [ "Test Alerts" ], "forecast": [ { "date": "2025-01-14 17:02", "temp_c": 28.0, "is_night": false } ]

```

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

The screenshot shows the Kusto Query Editor interface. On the left, the 'Explorer' sidebar lists 'Tables' (weather-table), 'Materialized View', 'Shortcuts', and 'Functions'. The main area displays a KQL query set:

```
1  /*weather-table*/
2  | where alerts != '[]' // Filter for records where an alert condition exists
3  | extend AlertValue = tostring(alerts) // Extract alerts as a string
4  | summarize LastTriggered = max(EventProcessedUtcTime) by AlertValue
5
6
7  // | join kind=leftanti (
8  //   ['weather-table']
9  //   | where alerts != '[]'
10 //   | extend AlertValue = tostring(alerts) // Extract alerts as a string
11 //   | summarize LastTriggered = max(EventProcessedUtcTime) by AlertValue
12 //   | where LastTriggered < ago(1m)
13 // ) on AlertValue
```

Below the query, a table named 'Table 1' is shown with one row:

AlertValue	LastTriggered
["Test Alerts"]	2025-01-14 11:5...

At the top right, there are buttons for 'Run', 'Preview', 'Recall', 'Copy query', 'Pin to dashboard', 'KQL Tools', 'Export to CSV', 'Create Power BI report', and 'Set alert'.

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

ⓘ Setting alerts from a KQL queryset is currently in preview

⚡ Set alert

?

X

Monitor

Source

 weather-eventhouse

Run query every

5 minutes



1 minute

5 minutes

15 minutes

30 minutes

1 hour

3 hours

6 hours

 Message me in Teams

 Run a Fabric item

Save location

Workspace

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

⚡ Set alert

?

X

Monitor

Source



Run query every

1 minute

i

Condition

Check

On each event

Action

Send me an email

Message me in Teams

Run a Fabric item

Save location

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

Home Rules

Delete Edit details Start Stop Send me a test action

Explorer

Search

KQL

e9d73a7a-acfd-4e03-a63d-220b610350fe

e9d73a7a-acfd-4e03-a63d-220b610350fe alert

Live feed Definition Analytics History

Monitor e9d73a7a-acfd-4e03-a63d-220b610350fe event

No data to show. Try changing parameters, population sample, or time range.

Condition On every value

Definition

Monitor

Event e9d73a7a-acfd-4e03-a63d-220b610350fe event

Condition 1

Operation On every value

Action

Type Email

To Mr. K Talks Tech

Subject Activator Trigger: New event from e9d73a7a-acfd-4e03-

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

