

Kafka/Event Hub to Event Hub demo

This document is a guided lab for: [Jeffrey89/Kafka-Topic-Replication-To-Kafka-EventHub](#):

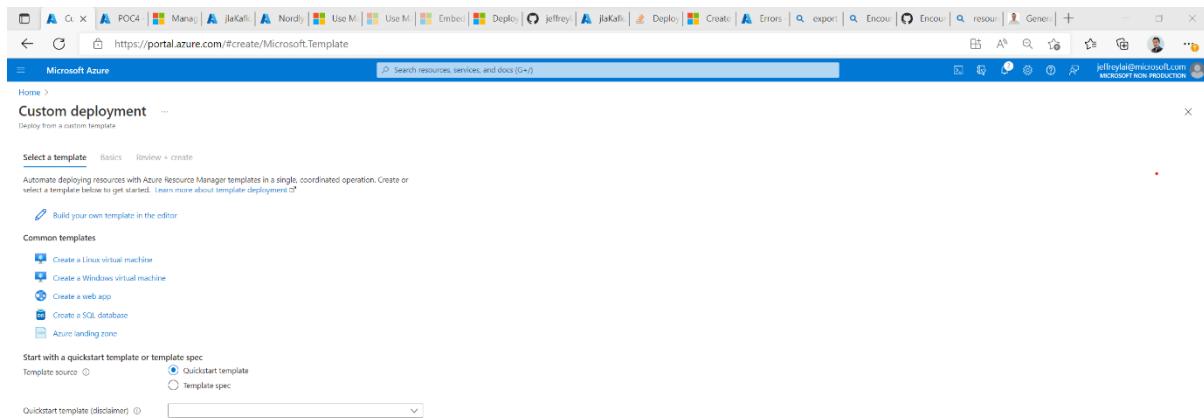
Contents

Kafka/Event Hub to Event Hub demo	1
Arm template deployment	2
Creating Azure Function.....	5
Running the demo.....	10
1. Producing Events:	10
2. Consuming and verifying the events.....	11
3. Consuming and verifying the events on <i>EventHubConsumerSample</i>	11
Optional Step1	13
Optional step2:	18

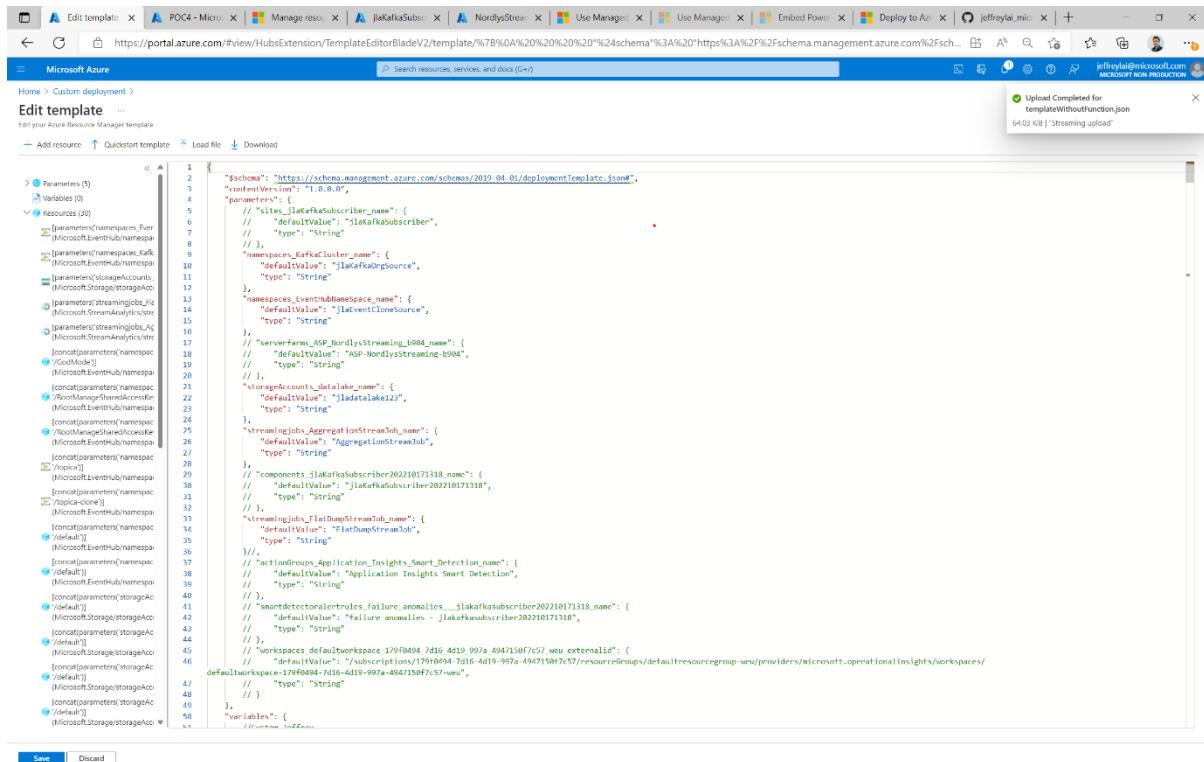
Arm template deployment

Background: Below steps will deploy some pre-created resources in order to save 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”



```
$schema": "https://schema.management.azure.com/schemas/2019_04_01/deploymentTemplate.json#",
contentVersion": "1.0.0.0",
parameters: {
  "sites_IotHubSubscriber_name": {
    "defaultValue": "IotHubSubscriber",
    "type": "string"
  },
  "namespaces_KafkaCluster_name": {
    "defaultValue": "IotHubKafkaSource",
    "type": "string"
  },
  "namespaces_EventHubNameSpace_name": {
    "defaultValue": "IotEventHubSource",
    "type": "string"
  },
  "serverFarms_ASP_NordlysStreaming_b904_name": {
    "defaultValue": "ASP-NordlysStreaming-b904",
    "type": "string"
  },
  "storageAccounts_datalake_name": {
    "defaultValue": "Iotdatalake12",
    "type": "string"
  },
  "streamingJobs_aggregationStreamJob_name": {
    "defaultValue": "AggregationStreamJob",
    "type": "string"
  },
  "components_IotHubSubscriber902210171310_name": {
    "defaultValue": "IotHubSubscriber902210171310",
    "type": "string"
  },
  "streamingJobs_FlatDampStreamJob_name": {
    "defaultValue": "FlatDampStreamJob",
    "type": "string"
  },
  "actionGroups_Application_Insights_Smart_Detection_name": {
    "defaultValue": "Application Insights Smart Detection",
    "type": "string"
  },
  "smartDetectorAlertRules_failure_anomalies__IotKafkaSubscriber202210171318_name": {
    "defaultValue": "failure anomalies - IotKafkaSubscriber202210171318",
    "type": "string"
  },
  "workspaces_defaultworkspace_179f0494_7d16_4d19_997a_4947150f7c57_weu_external": {
    "defaultValue": "/subscriptions/179f0494-7d16-4d19-997a-4947150f7c57/resourceGroups/defaultresourcegroup-weu/providers/microsoft.operationalinsights/workspaces/defaultworkspace_179f0494-7d16-4d19-997a-4947150f7c57",
    "type": "string"
  }
},
variables: {}
```

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

The screenshot shows the 'Custom deployment' blade for a Microsoft.Template resource. Key settings include:

- Subscription:** MCAFS-Hybrid-REQ-44538-2022-jeffreyai
- Resource group:** MyStreamDemo
- Instance details:**
 - Region: Europe West Europe
 - Namespaces_Kafka_Cluster_name: JlaKafkaOrgSource
 - Namespaces_Event_Hub_Name_Spanner_name: JlaEventHubCloneSource
 - Storage_Account_datalake_name: Jladatalake123
 - StreamingJobs_Aggregation_Stream_Job_name: AggregationStreamJob
 - StreamingJobs_Flat_Dump_Stream_Job_name: FlatDumpStreamJob

6. Await the deployment to finish.

The screenshot shows the 'Overview' blade for the Microsoft.Template deployment. Deployment status:

- Deployment name:** Microsoft.Template_20221021123742
- Start time:** 10/21/2022, 12:37:44 PM
- Correlation ID:** 4c9bfe6b-1a1-471-935e-eb6d344d55

Deployment details:

Resource	Type	Status	Operation details
Jladatalake123	Microsoft.Storage/storageAccounts	Accepted	Operation details
AggregationStreamJob/minEventHub	Microsoft.StreamAnalytics/streamingJobs/inputs	OK	Operation details
AggregationStreamJob	Microsoft.StreamAnalytics/streamingJobs	OK	Operation details
JlaEventHubCloneSource	Microsoft.EventHub/namespaces	OK	Operation details
JlaKafkaOrgSource	Microsoft.EventHub/namespaces	OK	Operation details
FlatDumpStreamJob/topic-clone	Microsoft.StreamAnalytics/streamingJobs/inputs	OK	Operation details
FlatDumpStreamJob/FlatDumpMinDataLake	Microsoft.StreamAnalytics/streamingJobs/outputs	OK	Operation details
FlatDumpStreamJob	Microsoft.StreamAnalytics/streamingJobs	OK	Operation details

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.

The screenshot shows the Azure portal's Event Hub Overview page for the resource group 'MyStreamDemo'. The left sidebar includes options like Overview, Activity log, Tags, Diagnose and solve problems, Events, Settings, Shared access policies, Scale, Geo-recovery, Networking, Encryption, Configuration, Properties, and Locks. The main area shows the 'Essentials' section with details such as Resource group (name: MyStreamDemo), Status (Active), Location (West Europe), Subscription (jlapo), Subscription ID (179f0494-7d16-4d19-997a-4947150f7c57), Host name (jlaKafkaOrgSource.servicebus.windows.net), and various pricing and throughput settings. Below this is a chart showing 'Messages' and 'Throughput' over a 24-hour period. At the bottom, there is a table for topics, showing one entry for 'topic' with Active status, 1 day message retention, and 2 partitions.

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.

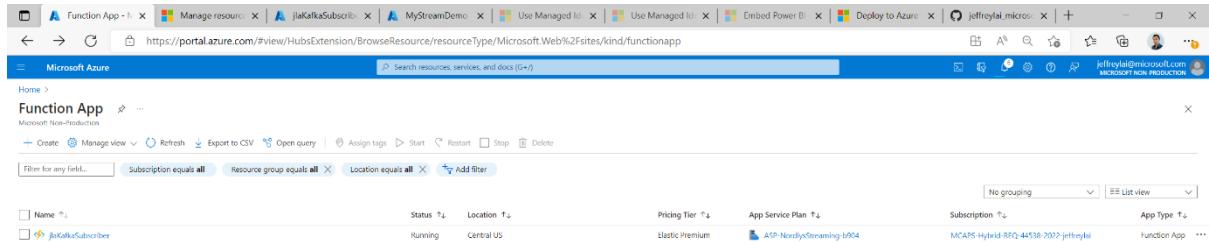
The screenshot shows the Azure portal's Shared access policies page for the resource group 'MyStreamDemo'. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Shared access policies, Scale, Geo-recovery, Networking, Encryption, Configuration, and Properties. The main area shows a table for policies, with one row named 'RootManageSharedAccessKey'. A modal window titled 'SAS Policy: RootManageSharedAccessKey' is open, showing fields for Primary key, Secondary key, Connection string-primary key, Connection string-secondary key, and SAS Policy ARM ID. The primary key value is highlighted with a red box.

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

Creating Azure Function

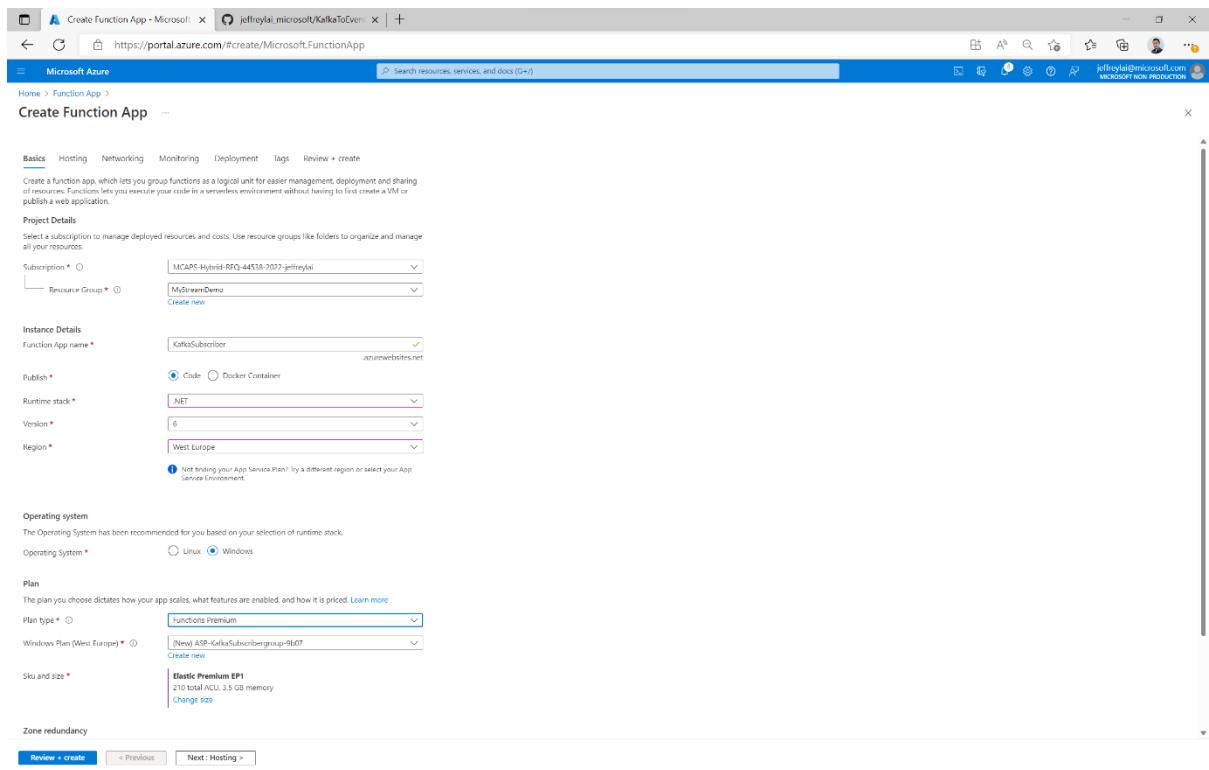
Background: In this section we will create a Function that will automatically trigger when a message is received in a specific topic in Kafka Cluster. The function will then send the message to an Event Hub.

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



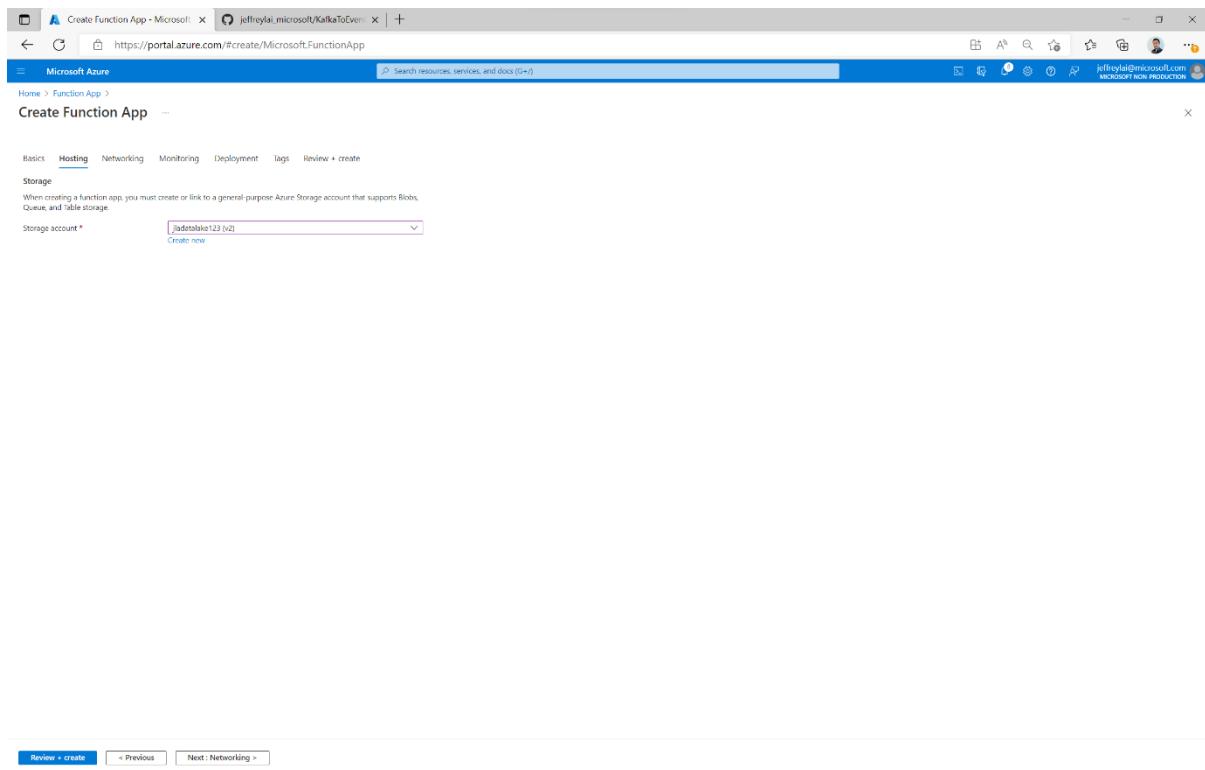
The screenshot shows the Microsoft Azure portal interface. The user is navigating through the 'Function App' section. A search bar at the top has 'Search resources, services, and docs (G+)' entered. Below the search bar, there are several tabs: 'Create', 'Manage view', 'Refresh', 'Export to CSV', 'Open query', 'Assign tags', 'Start', 'Restart', 'Stop', and 'Delete'. There is also a filter bar with 'Filter for any field...' and dropdowns for 'Subscription equals all', 'Resource group equals all', and 'Location equals all'. The main table lists one function app: 'Name': 'KafkaSubscriber', 'Status': 'Running', 'Location': 'Central US', 'Pricing Tier': 'Elastic Premium', 'App Service Plan': 'AS-
MCAPS-Hybrid-RFQ-44518-2022-jeffreylai', 'Subscription': 'MCAPS-Hybrid-RFQ-44518-2022-jeffreylai', and 'App Type': 'Function App'. The table has columns for Name, Status, Location, Pricing Tier, App Service Plan, Subscription, and App Type.

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

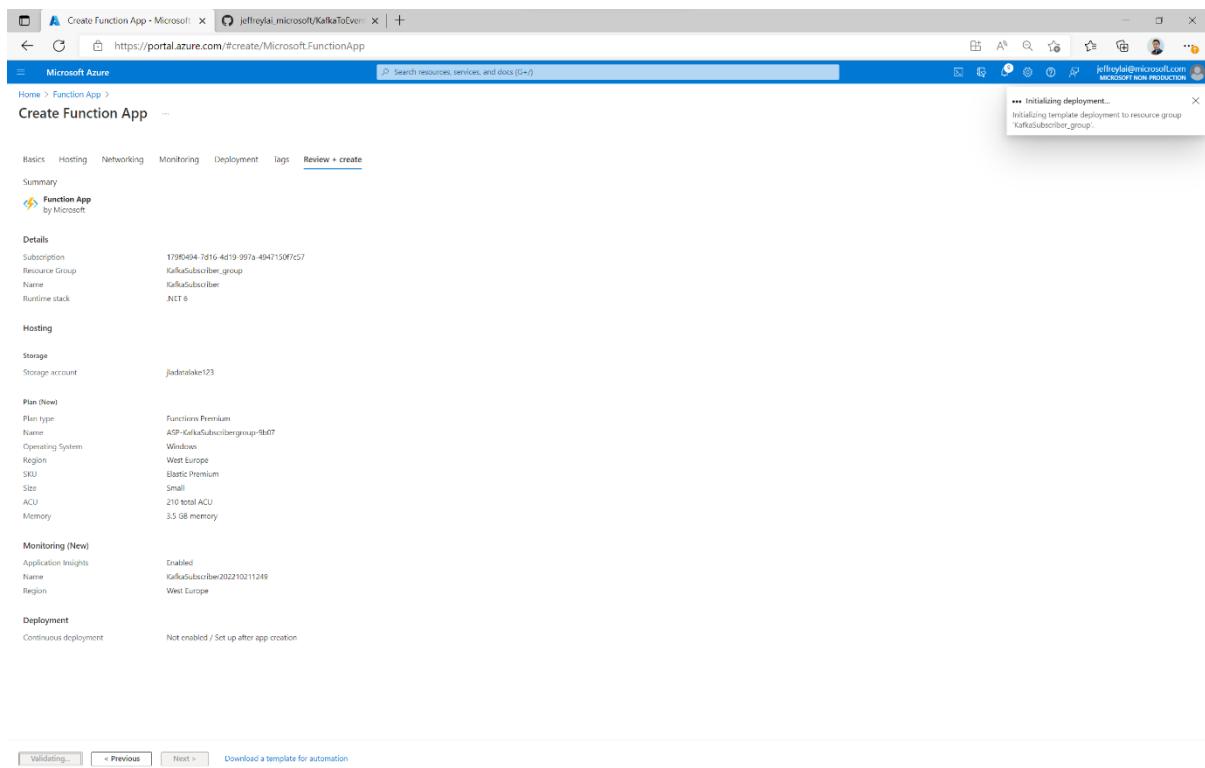


The screenshot shows the 'Create Function App' wizard on the 'Basic' step. The 'Function App name' field is filled with 'KafkaSubscriber'. Other fields include 'Runtime stack' set to '.NET', 'Version' set to '6', and 'Region' set to 'West Europe'. Under 'Operating System', 'Windows' is selected. Under 'Plan', 'Functions Premium' is chosen, and the 'Windows Plan (West Europe)' dropdown shows '(New) ASP-KafkaSubscribergroup-94d07'. At the bottom, there are buttons for 'Review + create' and 'Next : Hosting >'. The URL in the browser is https://portal.azure.com/#create/Microsoft.FunctionApp.

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:

The screenshot shows the Microsoft Azure portal with the URL https://portal.azure.com/#@fdpo.onmicrosoft.com/resource/subscriptions/1790494-7d16-4d19-997a-4947150f7c57/resourcegroups/KafkaSubscriber_group/providers/Microsoft.Web/sites/KafkaSubscriber. The page title is "KafkaSubscriber - Microsoft Azure". The left sidebar shows the "Functions" section under "Microsoft Azure". The main content area shows the "Overview" tab for the KafkaSubscriber function. It includes sections for "Essentials" (Resource group: KafkaSubscriber_group, Status: Running, Location: West Europe, Subscription ID: 1790494-7d16-4d19-997a-4947150f7c57), "Metrics" (Memory working set and Function Execution Count charts), and "Logs" (Activity log, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview)).

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

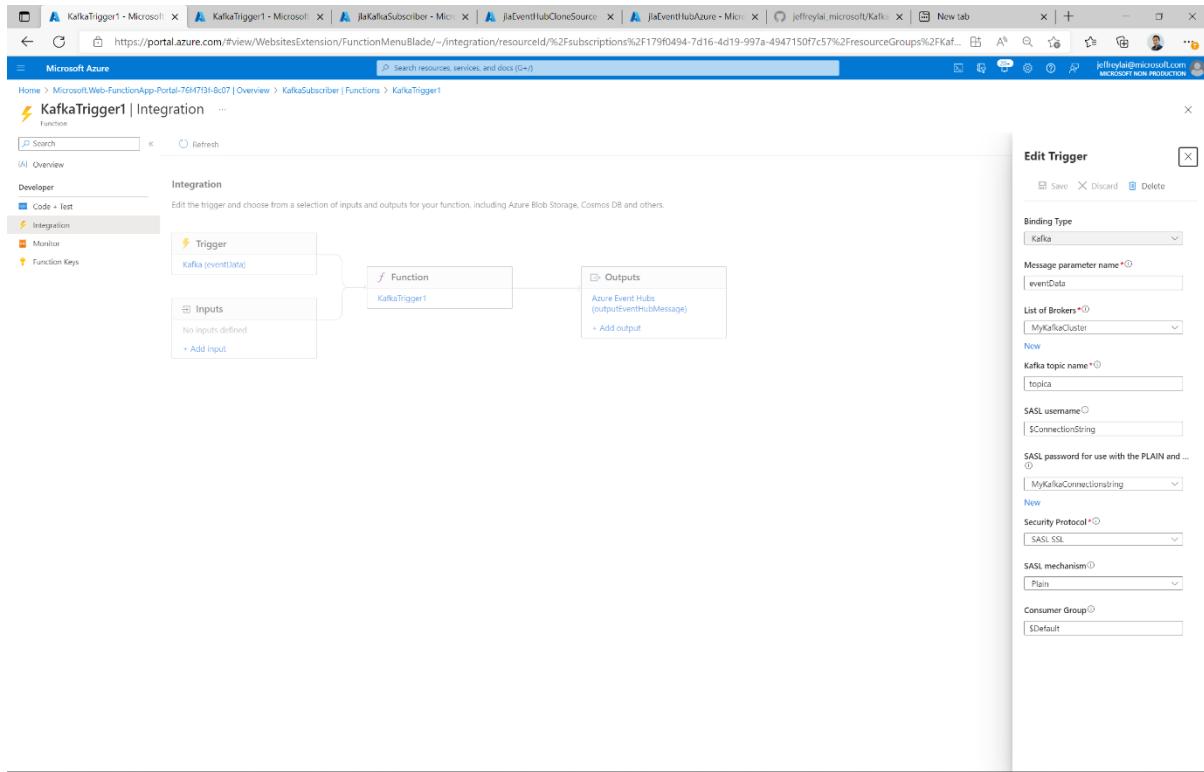
The screenshot shows the Microsoft Azure portal with the URL https://portal.azure.com/#@fdpo.onmicrosoft.com/resource/subscriptions/1790494-7d16-4d19-997a-4947150f7c57/resourcegroups/KafkaSubscriber_group/providers/Microsoft.Web/sites/KafkaSubscriber. The page title is "Create function - Microsoft Azure". The left sidebar shows the "Functions" section under "Microsoft Azure". The right pane shows the "Create function" dialog. It has sections for "Select development environment" (Development env.: Develop in portal) and "Select a template" (Template: Kafka trigger). The "Kafka trigger" template is selected. The "Description" table shows two entries: "Kafka output" (A function that will send message to a specified Kafka topic) and "Kafka trigger" (A function that will be run whenever a message is added to a specified Kafka topic). At the bottom are "Create" and "Cancel" buttons.

8. Fill out the details of the trigger template as below.

- For “List of Brokers” use the value that you pasted in notepad, and append “:9093”
- 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=/0gx7qlAk6j8a6VNJAD3Ngf9+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

```
using System;
using System.Security.Cryptography.X509Certificates;
using System.Text.Json.Serialization;
using System.Threading;
using Confluent.Kafka;
using MyTelemetryClasses;
using System.Collections.Generic;
using static Confluent.Kafka.ConfigPropertyNames;

namespace Kafka_Producer
{
    class Program
    {
        [STAThread]
        public static async Task Main(string[] args)
        {
            //Kafka settings
            const string eventHubNameSpace = "jlaKafkaOrgSource";
            const string topic = "topica";
            //Event settings
            int partitionCount = 10;
            bool dumpJsonOutput = true;
            bool useCloudEventFormat = true;
            var config = new ProducerConfig
            {
                //Kafka Config Template
                BootstrapServers = "localhost:9992",
                SaslMechanism = SaslMechanism.Plain,
                SecurityProtocol = SecurityProtocol.SaslSsl,
                SaslUsername = "xxxxxx",
                SaslPassword = "xxxxxx"
            };
            //EventHub Config Template
            eventHubNameSpace += ".servicebus.windows.net:9993";
            SecurityProtocol = SecurityProtocol.Sasl_Ssl,
            SaslMechanism = SaslMechanism.Plain;
            SaslUsername = "jlaKafkaOrgSource";
            SaslPassword = "jlaKafkaOrgSource";
            //Optional kafka settings below - to maintain message ordering and prevent duplication
            EnableTimestamp = true;
        }
    }
}
```

Output window showing 10 messages sent to topic: topica, broker(0): jlaKafkaOrgSource.servicebus.windows.net:9993. The messages contain fields like timestamp, customerID, ssid, signalStrength, downStreamBps, upStreamBps, packetLossPercentage, and jitter_ms.

```
Message 1 sent (value: {"timestamp": "2022-10-21T11:15:39.823072Z", "customerID": "JeffreyLai", "ssid": "JeffreyLai_Wifi", "channel": "33", "routeTemp": "16.0", "connectedDevices": [{"deviceID": "1", "signalStrength": "-64.0", "downStreamBps": "99314.0", "upStreamBps": "8896.0", "webConnections": [{"source": "IP of Desktop PC", "destination": "IP of A Popular Gaming Service", "latency_ms": "17.0", "packetLoss_Percentage": "0", "jitter_ms": "4.0"}]}, {"deviceID": "2", "signalStrength": "-49.0", "downStreamBps": "49204.0", "upStreamBps": "49204.0", "webConnections": [{"source": "IP of Desktop PC", "destination": "IP of A Baby Cloths Webshop", "latency_ms": "29.0", "packetLoss_Percentage": "0", "jitter_ms": "4.0"}]}, {"deviceID": "3", "signalStrength": "-43.0", "downStreamBps": "59146.0", "upStreamBps": "5498.0", "webConnections": [{"source": "IP of Desktop PC", "destination": "IP of A Popular Gaming Service", "latency_ms": "79.0", "packetLoss_Percentage": "0", "jitter_ms": "4.0"}]}], "topicOffset": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], "topicPartition": 0, "topic": "topica [0] #0"}]

Message 2 sent (value: {"timestamp": "2022-10-21T11:15:40.492079Z", "customerID": "JeffreyLai", "ssid": "JeffreyLai_Wifi", "channel": "33", "routeTemp": "16.0", "connectedDevices": [{"deviceID": "1", "signalStrength": "-64.0", "downStreamBps": "99314.0", "upStreamBps": "8896.0", "webConnections": [{"source": "IP of Desktop PC", "destination": "IP of A Popular Gaming Service", "latency_ms": "17.0", "packetLoss_Percentage": "0", "jitter_ms": "4.0"}]}, {"deviceID": "2", "signalStrength": "-49.0", "downStreamBps": "49204.0", "upStreamBps": "49204.0", "webConnections": [{"source": "IP of Desktop PC", "destination": "IP of A Baby Cloths Webshop", "latency_ms": "29.0", "packetLoss_Percentage": "0", "jitter_ms": "4.0"}]}, {"deviceID": "3", "signalStrength": "-43.0", "downStreamBps": "59146.0", "upStreamBps": "5498.0", "webConnections": [{"source": "IP of Desktop PC", "destination": "IP of A Popular Gaming Service", "latency_ms": "79.0", "packetLoss_Percentage": "0", "jitter_ms": "4.0"}]}], "topicOffset": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], "topicPartition": 0, "topic": "topica [0] #0"}]
```

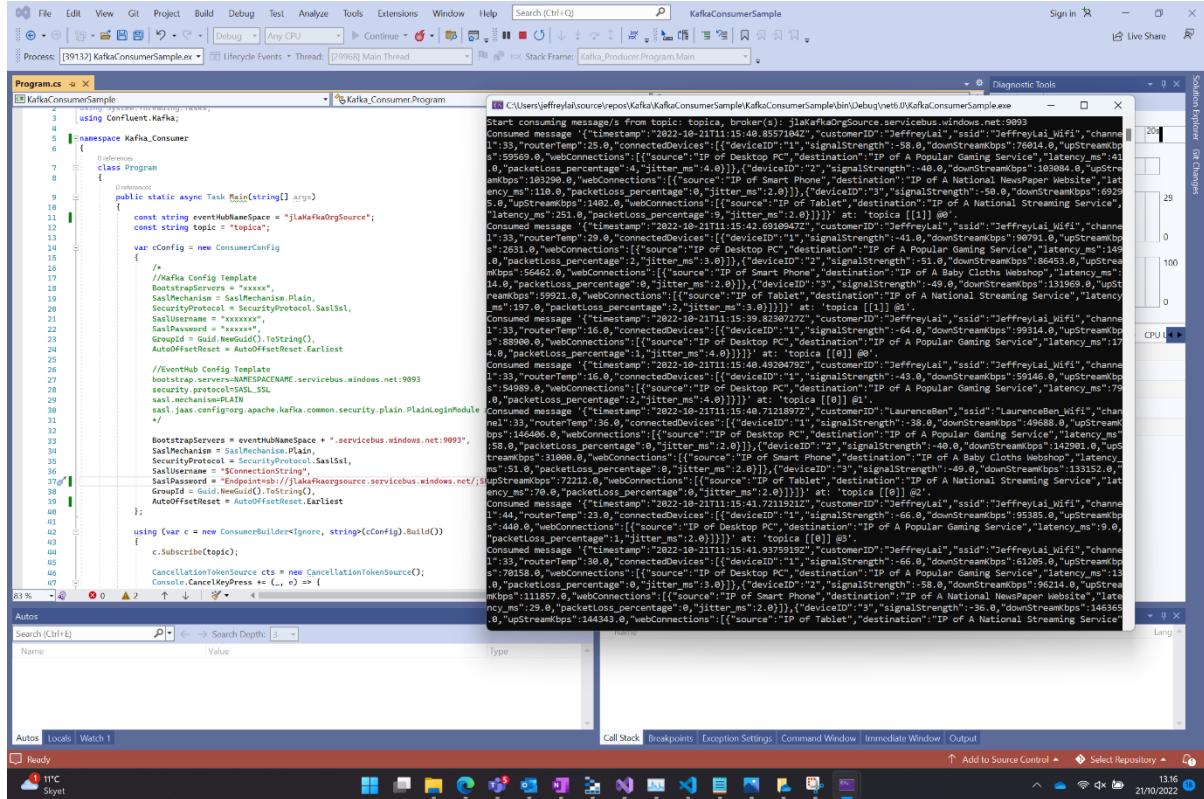
Error List window showing two CS8618 warnings: Non-nullable field 'ssid' must contain a non null value when exiting constructor. Consider declaring the field as nullable. and Non-nullable field 'customerID' must contain a non null value when exiting constructor. Consider declaring the field as nullable.

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/kVWhnNYxJF1XuOoOI=", //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=azHBQNu71yJV0ujCppWGByZ0bH6d+a+16GbQ4+8yh8=", //EventHub NameSpace Key1`
- Build and execute the console app

The screenshot shows the Microsoft Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure with files: Program.cs, EventHub.Consumer.Program.cs, and Main(string[] args).
- Error List:** Displays one error: CS1999: The 'await' operator can only be used within an async method.
- Task List:** Lists various tasks such as Build, Run, Clean, and Publish.
- Output Window:** Shows the build log output.
- Diagnostic Tools:** A Fiddler capture window showing network traffic. It displays multiple messages being consumed from the topic 'topic-clone'. Each message includes fields like timestamp, customer ID, signal strength, and up/downstream bandwidth. The traffic shows data being sent from a Kafka cluster to an Event Hub.

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.SignalStrength
    ,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.SignalStrength
    ,main.downStreamKbps
    ,main.upStreamKbps
    ,main.PartitionKey
    ,webConnection.ArrayValue.*
INTO
    FlatDumpMinDataLake
FROM cte as main
CROSS APPLY GetArrayElements(main.webConnections) AS webConnection
```

```

WITH cte AS (
    SELECT
        event.customerID,
        event.ssId,
        event.eventType,
        event.channel,
        event.timestamp,
        connectedDevice.ArrayValue.[
            connectedDevice.ArrayValue.deviceId,
            connectedDevice.ArrayValue.SignalStrength,
            connectedDevice.ArrayValue.downStreamBps,
            connectedDevice.ArrayValue.upStreamBps,
            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,
    ...

```

Input preview Test results

Access to EventHub://[eventhubname].servicebus.windows.net/topicA clone is not authorized. Exception Attempted to perform an unauthorized operation. [SessionID: e5554ca82bd046d8ae5c81d883a2bd]

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”.

My access

Check access

Add role assignment View Learn more (?)

View access to this resource

View deny assignments

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

Add role assignment

Role Members 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 [F]

Name	Description	Type	Category	Details
Owner	Grants full access to manage all resources, including the ability to assign roles in Azure RBAC.	BuiltInRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Azure RBAC, manage assignments in Azure blueprints, or share image galleries.	BuiltInRole	General	View
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General	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 adding the VM extension to VMs; reading store...	BuiltInRole	Analytics	View
Log Analytics Reader	Log Analytics Reader can view and search all monitoring data as well as view monitoring settings, including viewing the configuration of Azure diagnostics on all Azur...	BuiltInRole	Analytics	View
Managed Application Contributor Role	Allows for creating managed application resources.	BuiltInRole	Management + Governance	View
Managed Application Operator Role	Lets you read and perform actions on Managed Application resources.	BuiltInRole	Management + Governance	View
Managed Applications Reader	Lets you read resources in a managed app and request JIT access.	BuiltInRole	Management + Governance	View
Monitoring Contributor	Can read all monitoring data and update monitoring settings.	BuiltInRole	Monitor	View
Monitoring Metrics Publisher	Enables publishing metrics against Azure resources.	BuiltInRole	Monitor	View
Monitoring Reader	Can read all monitoring data.	BuiltInRole	Monitor	View
Resource Policy Contributor	Users with rights to create/modify resource policy, create support ticket and read resources/hierarchy.	BuiltInRole	Management + Governance	View
Role Based Access Control Administrator (Preview)	Manage access to Azure resources by assigning roles using Azure RBAC. This role does not allow you to manage access using other ways, such as Azure Policy.	BuiltInRole	None	View
Scheme Registry Contributor (Preview)	Read, write, and delete Schema Registry groups and schemas.	BuiltInRole	Preview	View
Scheme Registry Reader (Preview)	Read and list Schema Registry groups and schemas.	BuiltInRole	Preview	View
User Access Administrator	Lets you manage user access to Azure resources.	BuiltInRole	General	View

Review + assign Previous Next

- 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.

Select managed identities

Subscription: MCAPS-Hybrid-RFQ-44510-2022-jrileyal

Managed identity: Stream Analytics job (4)

Selected role: Azure Event Hubs Data Receiver

Assign access to: Managed identity

Members: + Select members

Description: Optional

Select Close

- 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.

The screenshot shows the Azure Stream Analytics job configuration page for 'FlatDumpStreamJob'. The left sidebar lists various settings like Overview, Activity log, Access control (IAM), Tags, Diagnostic and solve problems, Settings, Properties, Locks, Job topology, Inputs, Functions, Outputs, Configure, Environment, Storage account settings, Scale, Locale, Event ordering, Error policy, Compatibility level, Managed identity, Developer tools, Visual Studio Code, Job diagram (preview), Monitoring, Logs, Metrics, Alert rules, Diagnostic settings, and Automation. The 'Query' section is selected. The main area contains the Stream Analytics job configuration. The 'Inputs' section shows one input named 'topic-clone'. The 'Outputs' section shows one output named 'FlatDumpMinDataLake'. The 'Functions' section is empty. The 'Query' section displays the following T-SQL code:

```

    SELECT
        [topic-clone].event,
        [topic-clone].connectedDevice.ArrayValue.webConnections
    FROM
        [topic-clone] AS event
    CROSS APPLY GetArrayElements(event.connectedDevices) AS connectedDevice
    SELECT
        [topic-clone].event,
        [topic-clone].connectedDevice.main.customerID,
        [topic-clone].connectedDevice.main.xssid,
        [topic-clone].connectedDevice.main.routerID,
        [topic-clone].connectedDevice.main.channel,
        [topic-clone].connectedDevice.main.deviceID,
        [topic-clone].connectedDevice.main.SignalStrength,
        [topic-clone].connectedDevice.main.downstreamBps,
        [topic-clone].connectedDevice.main.upstreamBps,
        [topic-clone].connectedDevice.main.PartitionKey,
        [topic-clone].connectedDevice.ArrayValue,
        [topic-clone].connectedDevice.main.webConnections
    FROM
        FlatDumpMinDataLake
    CROSS APPLY GetArrayElements(main.webConnections) AS webconnection

```

The 'Input preview' section shows sample events from 'topic-clone'. The table has columns: timestamp, customerID, ssid, channel, routerID, connectedDevices, EventProcessedUtcTime, PartitionId, and EventEnqueuedUtcTime. The data is as follows:

timestamp	customerID	ssid	channel	routerID	connectedDevices	EventProcessedUtcTime	PartitionId	EventEnqueuedUtcTime
2022-10-21T12:16:04.27...	"jeffrey\air"	"JeffreyLe_WiFi"	33	21	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	0	"2022-10-21T12:17:20.11..."
2022-10-21T12:16:04.08...	"jeffrey\air"	"JeffreyLe_WiFi"	44	20	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	0	"2022-10-21T12:17:20.08..."
2022-10-21T12:16:03.55...	"Laurence\air"	"LaurenceBen_WiFi"	33	30	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	1	"2022-10-21T12:17:20.04..."
2022-10-21T12:16:03.79...	"Laurence\air"	"LaurenceBen_WiFi"	33	25	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	0	"2022-10-21T12:17:19.70..."
2022-10-21T12:16:03.26...	"jeffrey\air"	"JeffreyLe_WiFi"	33	21	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	0	"2022-10-21T12:17:19.60..."
2022-10-21T12:16:03.20...	"Laurence\air"	"LaurenceBen_WiFi"	33	31	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	1	"2022-10-21T12:17:19.43..."
2022-10-21T12:16:01.85...	"jeffrey\air"	"JeffreyLe_WiFi"	33	19	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	1	"2022-10-21T12:17:19.18..."
2022-10-21T12:16:00.00...	"Laurence\air"	"LaurenceBen_WiFi"	44	41	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	0	"2022-10-21T12:17:19.17..."
2022-10-21T12:16:02.67...	"jeffrey\air"	"JeffreyLe_WiFi"	44	21	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	1	"2022-10-21T12:17:19.01..."
2022-10-21T12:14:04.36...	"jeffrey\air"	"JeffreyLe_WiFi"	33	32	["deviceID":1,"signalSt...]	"2022-10-21T13:09:41.44..."	0	"2022-10-21T12:17:19.05..."

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

The screenshot shows the Azure Stream Analytics job overview page for 'FlatDumpStreamJob'. The left sidebar lists various settings like Overview, Activity log, Access control (IAM), Tags, Diagnostic and solve problems, Settings, Properties, Locks, Job topology, Inputs, Functions, Outputs, Configure, Environment, Storage account settings, Scale, Locale, Event ordering, Error policy, Compatibility level, Managed identity, Developer tools, Visual Studio Code, Job diagram (preview), Monitoring, Logs, Metrics, Alert rules, Diagnostic settings, and Automation. The 'Properties' tab is selected. The 'Created' section shows the job was created on Friday, October 21, 2022 12:37 PM. The 'Inputs' section shows one input named 'topic-clone'. The 'Outputs' section shows one output named 'FlatDumpMinDataLake'. The 'Functions' section is empty. The 'Properties' section shows the environment is Standard (multi-tenant), storage account settings are linked to 'myStreamAnalytics', scale is 3 streaming units, out of order tolerance is 5 seconds, error policy is Retry, and managed identity is '42392295-44fe-4901-9d5c-6a8f40402fbf'. The 'Monitoring' section shows 0 alert rules and 0 diagnostic settings configured.

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)

- 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:

The screenshot shows the Microsoft Azure Storage browser interface for the 'jladatalake123' account. The left sidebar contains navigation links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, and Storage browser. Under Storage browser, there are sections for Data storage (Containers, File shares, Queues, Tables) and Security + networking (Networking, Azure CDN, Access keys, Shared access signature, Encryption, Microsoft Defender for Cloud). The main content area displays a hierarchical tree view of blob containers: 'blob containers' > 'streamanalyticflfddump' > 'routedata' > 'flfddump' > '2022' > '10' > '21' > 'JeffreyLai'. A search bar at the top right allows searching by prefix (case-sensitive). Below the tree view is a table listing blobs:

Name	Last modified	Access tier	Block type	Size	Lease state
2001050297.e5802af77e2b45819b2db3fae91d797.1.csv	22/10/2022, 00:20:44	Hot (Inferred)	Block blob	1 KB	Available
2001050297.f14cd75801d44f384dc3502775a9bf_1.csv	22/10/2022, 00:20:44	Hot (Inferred)	Block blob	605 B	Available

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: “StreamAggregation”.
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)
```

Successful connection test
Connection to output 'MyPowerBIDOutput' succeeded.

```

26 , main.customerID
27 , main.sessionID
28 , main.deviceID
29 , min(main.AvgDeviceDownStreamMbps) AS AvgDeviceDownStreamMbps
30 , min(main.AvgDeviceUpStreamMbps) AS AvgDeviceUpStreamMbps
31 , min(main.SignalStrength_dbm) AS MinSignalStrength_dbm
32 , min(main.SignalStrength_dbm) AS AvgSignalStrength_dbm
33 , min(main.SignalStrength_dbm) AS AvgSignalStrength_dbm
34 , webConnection.ArrayValue.source
35 , webConnection.ArrayValue.destination
36 , round(avg(webConnection.ArrayValue.latency_ms),2) AS AvgLatency_ms
37 , round(avg(webConnection.ArrayValue.packetLoss_Percentage),2) AS AvgPacketLoss_Percentage
38 , round(avg(webConnection.ArrayValue.jitter_ms),2) AS AvgJitter_ms
39 INTRO
40 MyPowerBIDOutput
41 FROM #T AS main
42 CROSS APPLY GetArrayElements(main.webConnections) AS webConnection
43 GROUP BY
44     main.WindowEnd
45     , main.connectorID
46     , main.sessionID
47     , main.deviceID
48     , webConnection.ArrayValue.source
49     , webConnection.ArrayValue.destination
50     , TumblingWindow(second,10)

```

Input preview Test results

Showing sample events from minEventHub:

timestamp	customerId	sessionId	channel	routerTemp	connectedDevices	EventProcessedUtcTime	PartitionId	EventEnqueuedUtcTime
2022-10-21T12:14:03.29...	"jeffreyal"	"JeffreyAl_Wifi"	33	10	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	0	2022-10-21T12:17:17:30...
2022-10-21T12:14:03.38...	"jeffreyal"	"JeffreyAl_Wifi"	33	17	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	1	2022-10-21T12:17:18:21...
2022-10-21T12:14:03.55...	"jeffreyal"	"JeffreyAl_Wifi"	44	43	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	1	2022-10-21T12:17:18:35...
2022-10-21T12:14:03.18...	"jeffreyal"	"JeffreyAl_Wifi"	33	30	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	0	2022-10-21T12:17:18:36...
2022-10-21T12:14:03.07...	"jeffreyal"	"JeffreyAl_Wifi"	33	28	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	1	2022-10-21T12:17:18:36...
2022-10-21T12:14:01.01...	"jeffreyal"	"JeffreyAl_Wifi"	44	40	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	0	2022-10-21T12:17:18:37...
2022-10-21T12:14:03.44...	"jeffreyal"	"JeffreyAl_Wifi"	33	31	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	1	2022-10-21T12:17:18:38...
2022-10-21T12:14:04.22...	"LaureneReen"	"LaureneReen_Wifi"	33	22	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	0	2022-10-21T12:17:18:38...
2022-10-21T12:14:03.75...	"jeffreyal"	"JeffreyAl_Wifi"	33	30	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	1	2022-10-21T12:17:18:38...
2022-10-21T12:14:04.36...	"jeffreyal"	"JeffreyAl_Wifi"	33	32	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	0	2022-10-21T12:17:19:05...
2022-10-21T12:16:02.67...	"jeffreyal"	"JeffreyAl_Wifi"	44	21	[{"deviceID": "1", "signalSt...}	"2022-10-21T13:14:10.19..."	1	2022-10-21T12:17:19:05...

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

Test query to show results here

```

26 , main.customerID
27 , main.sessionID
28 , main.deviceID
29 , min(main.AvgDeviceDownStreamMbps) AS AvgDeviceDownStreamMbps
30 , min(main.AvgDeviceUpStreamMbps) AS AvgDeviceUpStreamMbps
31 , min(main.SignalStrength_dbm) AS MinSignalStrength_dbm
32 , min(main.SignalStrength_dbm) AS AvgSignalStrength_dbm
33 , min(main.SignalStrength_dbm) AS AvgSignalStrength_dbm
34 , webConnection.ArrayValue.source
35 , webConnection.ArrayValue.destination
36 , round(avg(webConnection.ArrayValue.latency_ms),2) AS AvgLatency_ms
37 , round(avg(webConnection.ArrayValue.packetLoss_Percentage),2) AS AvgPacketLoss_Percentage
38 , round(avg(webConnection.ArrayValue.jitter_ms),2) AS AvgJitter_ms
39 INTRO
40 MyPowerBIOutput
41 FROM #T AS main
42 CROSS APPLY GetArrayElements(main.webConnections) AS webConnection
43 GROUP BY
44     main.WindowEnd
45     , main.connectorID
46     , main.sessionID
47     , main.deviceID
48     , webConnection.ArrayValue.source
49     , webConnection.ArrayValue.destination
50     , TumblingWindow(second,10)

```

Input preview Test results

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.

The screenshot shows the Azure Stream Analytics job configuration page. On the left, the navigation pane includes sections like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Properties, Locks, Job topology, Inputs, Outputs, Configure, Environment, Storage account settings, Scale, Locale, Event ordering, Error policy, Compatibility level, Managed identity, Developer tools (Visual Studio Code, Job diagram (preview)), Monitoring (Logs, Metrics, Alert rules, Diagnostic settings), Automation, and Tools (preview). The main area displays the StreamAggregationJob configuration, specifically the Query section. The query language document shows a complex SQL-like query for aggregating data from various inputs (main.eventhub, main.webconnection) and calculating metrics like average device strength, average packet loss percentage, and average jitter. To the right, the Power BI interface is shown, allowing users to define an output alias (MyPowerBIOutput), select a workspace (a68d9116-72ba-419d-952c-5c9b2f9fc659), and authorize the connection. A note at the bottom indicates that the user is granting full management access to their Power BI dataset.

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".

The screenshot shows the Microsoft Power BI workspace interface. The left sidebar includes Home, Create, Browse, Data hub, Metrics, Apps, Deployment pipelines, Learn, and Workspaces (JaStreamingPOC). The main area displays the 'JaStreamingPOC' workspace. A message at the top says "We updated the look of workspaces Take a tour, and we'll show you how to get around." Below this, a table lists datasets and dataflows. The 'MyStreamAggregationDs' dataset is listed under the All tab. A context menu is open over this dataset, showing options: Create report, Delete, Manage permissions, Quick insights, Edit, API Info, and Settings. The URL in the browser bar is https://msit.powerbi.com/groups/a68d9116-72ba-419d-952c-5c9b2f9fc659/list.

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

Microsoft Power BI JlaStreamingPOC

We updated the look of reports. Take a tour, and we'll show you how to get around.

Start tour

Filters

Visualizations

Fields

Values

Drill through

Cross report

Keep all filters

Add drill through fields here

Get data

Page 1

customerID	WindowEnd	deviceID	Sum of AvgDeviceDownStreamMbps	Sum of AvgDeviceUpStreamMbps	Sum of AvgLatency_ms	Sum of AvgLatency_ms	Sum of AvgPacketsLoss_Percentage
jeffrey.a	10/21/22 01:19:59 PM	1	1.27	26.42	3.00	137.00	2.00
jeffrey.a	10/21/22 01:20:00 PM	1	4.55	7.47	2.95	115.00	2.9
jeffrey.a	10/21/22 01:20:01 PM	1	2.52	2.52	3.97	100.53	1.09
jeffrey.a	10/21/22 01:20:02 PM	1	19.51	60.14	3.17	90.18	1.33
jeffrey.a	10/21/22 01:20:00 PM	2	150.89	133.08	4.00	907.00	0.03
jeffrey.a	10/21/22 01:19:59 PM	2	155.27	110.79	6.00	115.60	0.05
jeffrey.a	10/21/22 01:20:01 PM	2	24.76	155.49	5.00	145.00	2.00
jeffrey.a	10/21/22 01:20:02 PM	3	41.59	44.89	3.90	80.00	5.00
jeffrey.a	10/21/22 01:19:59 PM	3	76.91	9.78	2.67	90.00	1.67
LaurenceBen	10/21/22 01:20:00 PM	1	23.70	54.41	2.00	900.00	1.00
LaurenceBen	10/21/22 01:20:01 PM	1	63.11	3.71	1.00	40.00	1.00
LaurenceBen	10/21/22 01:20:02 PM	1	71.21	83.01	3.00	130.25	2.25
LaurenceBen	10/21/22 01:20:00 PM	2	105.42	54.43	2.00	54.00	0.00
LaurenceBen	10/21/22 01:19:59 PM	2	64.89	101.54	2.00	24.00	0.00
LaurenceBen	10/21/22 01:20:01 PM	2	127.55	237.46	4.00	150.00	2.00
LaurenceBen	10/21/22 01:20:02 PM	3	25.35	159.28	2.00	100.00	3.00
Total			1,287.15	1,174.68	52.02	1,609.91	24.43

Get data

Page 1

Customer Details

Page 2 of 2

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

Microsoft Power BI JlaStreamingPOC

We updated the look of reports. Take a tour, and we'll show you how to get around.

Start tour

Visualizations

Fields

Values

Drill through

Cross-report

Keep all filters

Add drill through fields here

Get data

Page 1

Customer Details

Page 2 of 2