

A dark blue vertical bar runs along the left edge of the slide. A blue arrow-shaped banner points to the right from this bar, containing the date. Below the banner, several thin, curved lines in shades of blue and grey sweep upwards from the bottom left corner.

05/03/2019

Real-Time Data Ingestion with Azure Stream Analytics

Roy, Sandip (Cognizant)
COGNIZANT

Table of Contents

1. Introduction	3
2. Solution Components	3
3. Solution	3
4. Detailed Steps	4
5. References	14

1. Introduction

This document outlines how to ingest real-time streaming data (structured or semi structured) using Azure Stream Analytics and store them in persistent storage (e.g. Azure SQL DB) for any future analytical purposes.

2. Solution Components

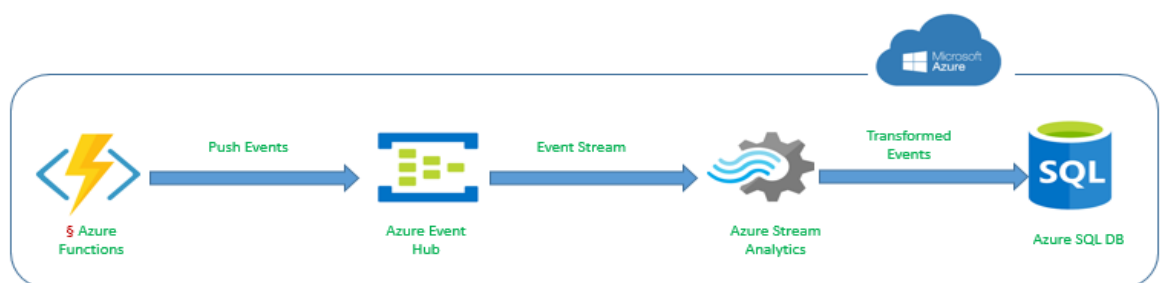
Azure Event Hubs is a big data streaming platform and event ingestion service. It can receive and process millions of events per second. Data sent to an event hub can be transformed and stored by using any real-time analytics provider or batching/storage adapters.

Azure Stream Analytics is an event-processing engine that allows you to examine high volumes of data streaming from devices. Incoming data can be from devices, sensors, web sites, social media feeds, applications, and more. It also supports extracting information from data streams, identifying patterns, and relationships.

Azure SQL Database is a managed cloud database provided as part of Microsoft Azure. A cloud database is a database that runs on a cloud computing platform, and access to it is provided as a service. Managed database services take care of scalability, backup, and high availability of the database.

Azure Function App is a solution for easily running small pieces of code, or "functions," in the cloud. You can write just the code you need for the problem at hand, without worrying about a whole application or the infrastructure to run it.

3. Solution



§ Used in this demo for simulation only. In practical cases, we can have different producer pushing events into Event Hub

4. Detailed Steps

Step 1 – Create an Azure Event Hub.

Firstly, create Namespace/Service Bus component under which your Event Hub will reside.



Create Namespace
Event Hubs

* Name

* Pricing tier (View full pricing details)

☐ Enable Kafka ⓘ

☐ Make this namespace zone redundant ⓘ

* Subscription

* Resource group [Create new](#)

* Location

* Throughput Units

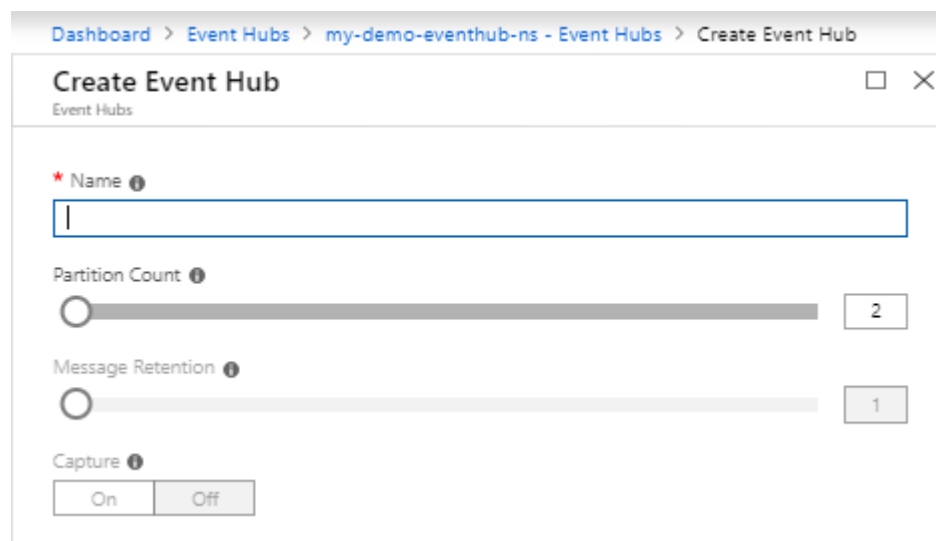
☐ Enable Auto-Inflate ⓘ

Create

1 items

NAME	STATUS	TIER	LOCATION	THROUGHPUT UNITS	RESOURCE GROUP	SUBSCRIPTION
my-demo-eventhub-ns	Active	Basic	West Europe	1	my-demo-rg	Free Trial

Now create Event Hub under Namespace created above.



Dashboard > Event Hubs > my-demo-eventhub-ns - Event Hubs > Create Event Hub

Create Event Hub
Event Hubs

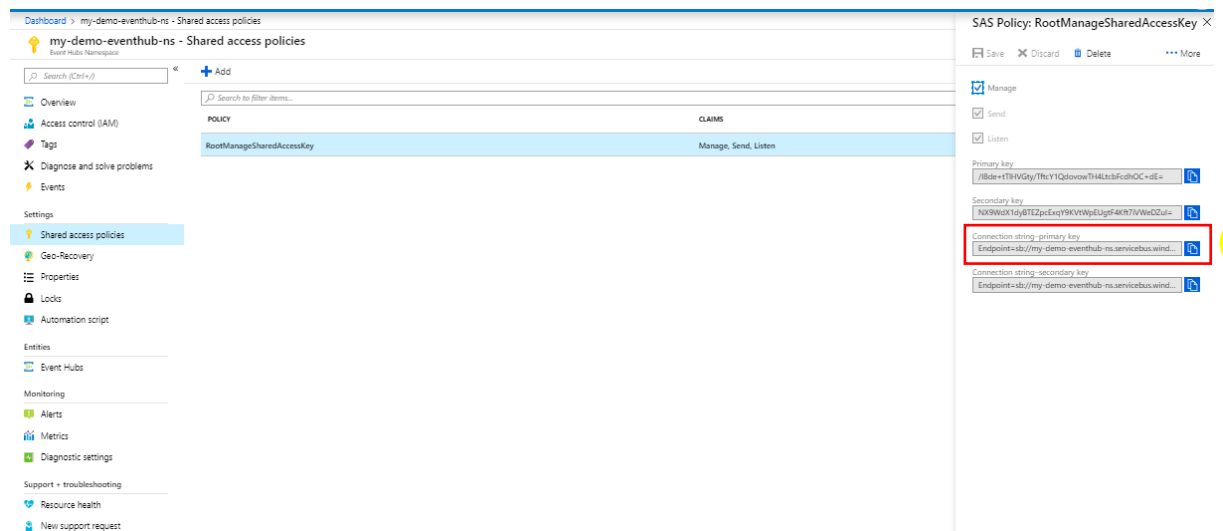
* Name

Partition Count

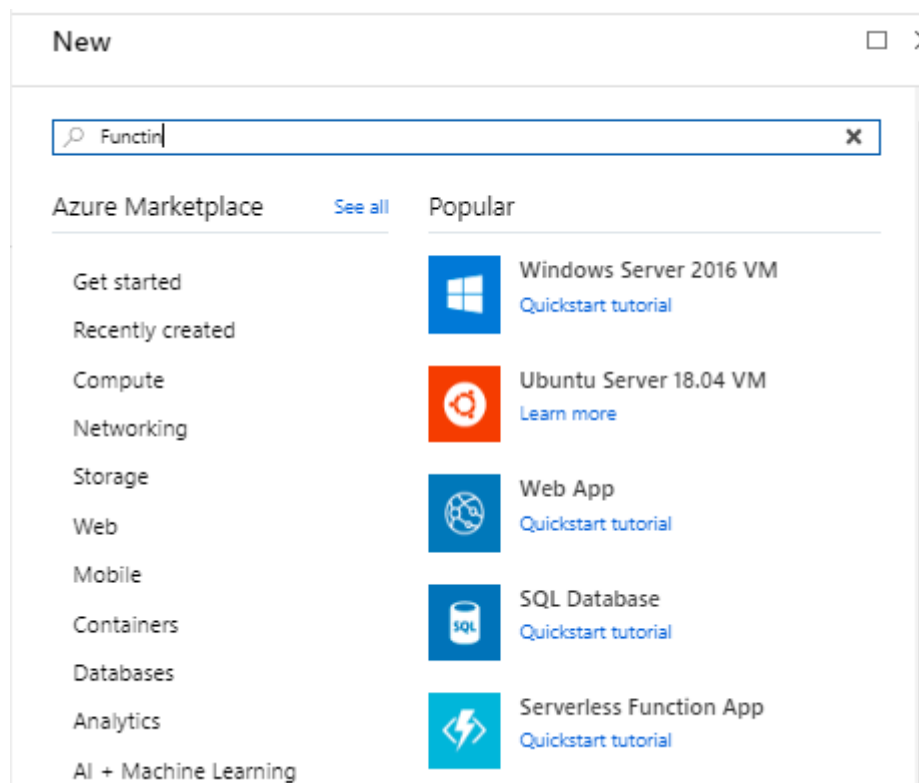
Message Retention

Capture

Once Event Hub is created go to parent namespace and check Shares access policies under Settings to take a note of the SAS (Shared Access Signature) policy of the namespace – this is required in order to access the namespace and its decedent resources (Event Hub) with required permissions. Make sure you have send and listen permission on to this as per requirement.



Step 2 – Create an Azure Function (Function App) which would send JSON stream of event payload into Azure Event Hub in periodic manner (let say every 2 minutes or so).



Function App

Microsoft

✈ □ ✕

Write any function in minutes – whether to run a simple job that cleans up a database or build a more complex architecture. Creating functions is easier than ever before, whatever your chosen OS, platform, or development method.

🔍 Save for later

PUBLISHER

Microsoft

USEFUL LINKS

[Documentation](#)
[Solution Overview](#)
[Pricing Details](#)

Create

Function App

Create

□ ✕

* App name

.azurewebsites.net

* Subscription

Free Trial ▾

* Resource Group ⓘ

☒ Create new ☐ Use existing

* OS

Windows Linux (Preview)

* Hosting Plan ⓘ

Consumption Plan ▾

* Location

Central US ▾

* Runtime Stack

.NET ▾

* Storage ⓘ

☒ Create new ☐ Use existing

✓

Application Insights

Disabled >

Create

Automation options

sandip-app
Function Apps

Search: "sandip-app" ✕

All subscriptions ▾

- Function Apps
 - sandip-app** ⌂ ⌵
 - Functions +
 - TimerTrigger1
 - Integrate
 - Manage
 - Monitor
 - Proxies
 - Slots (preview)

Overview Platform features

Stop Swap Restart Get publish profile Reset publish profile Download app content Delete

Status Running	Subscription Free Trial	Resource group my-demo-rg	URL https://sandip-app.azurewebsites.net
	Subscription ID e46bee66-4fc6-4188-8d80-08f6385a2659	Location West Europe	App Service plan / pricing tier WestEuropePlan (Consumption)

Configured features

- Function app settings
- Application settings
- Application Insights

Select the in-portal option to develop the simple C# (you can Node JS or Java script method also) method.

Overview Platform features Quickstart ✕

Azure Functions for .NET - getting started

Follow our Quickstart guidance to author and publish a function [Learn more](#)

1

2

CHOOSE A DEVELOPMENT ENVIRONMENT CREATE A FUNCTION

Visual Studio

Use Visual Studio to author, build, and run .NET functions

VS Code

Use Visual Studio Code to author your functions

Any editor + Core Tools

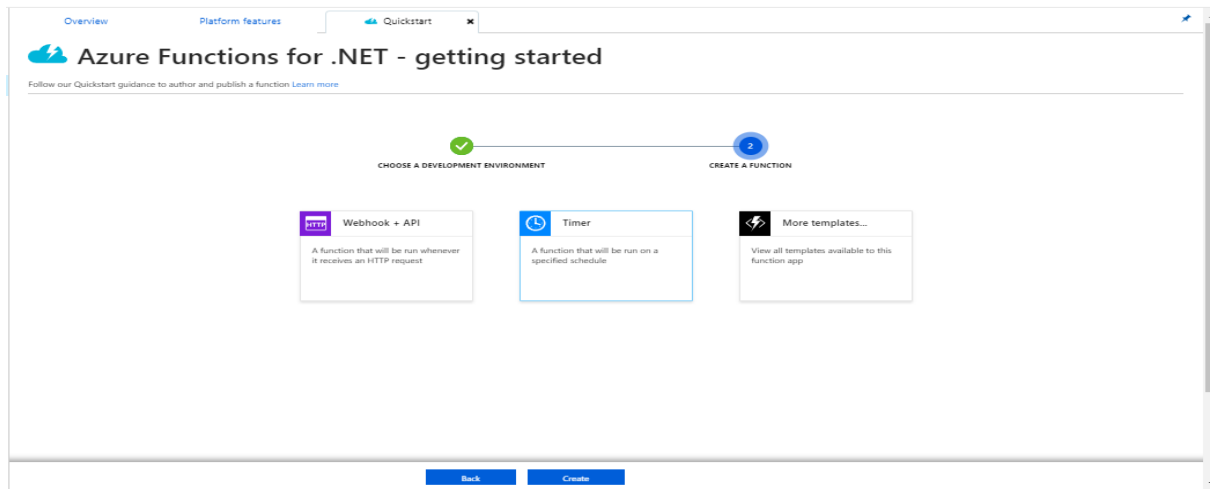
Write functions using your favorite editor and the Azure Functions Core Tools

In-portal

Author functions quickly in the portal

[Continue](#)

Select the required template for the method. We will go for Timer option in case as we would like to get our method executed in certain duration automatically so that it works as simulator for producing real-time data into Event Hub.



Once the Function is created (it is created with basic Function body), we will configure the Event Hub endpoint (i.e. SAS Primary Key that we highlighted above) as connection string in Azure Functions.

Connection strings

Connection strings should only be used with a function app if you are using entity framework. For other scenarios use App Settings. Click to learn more.

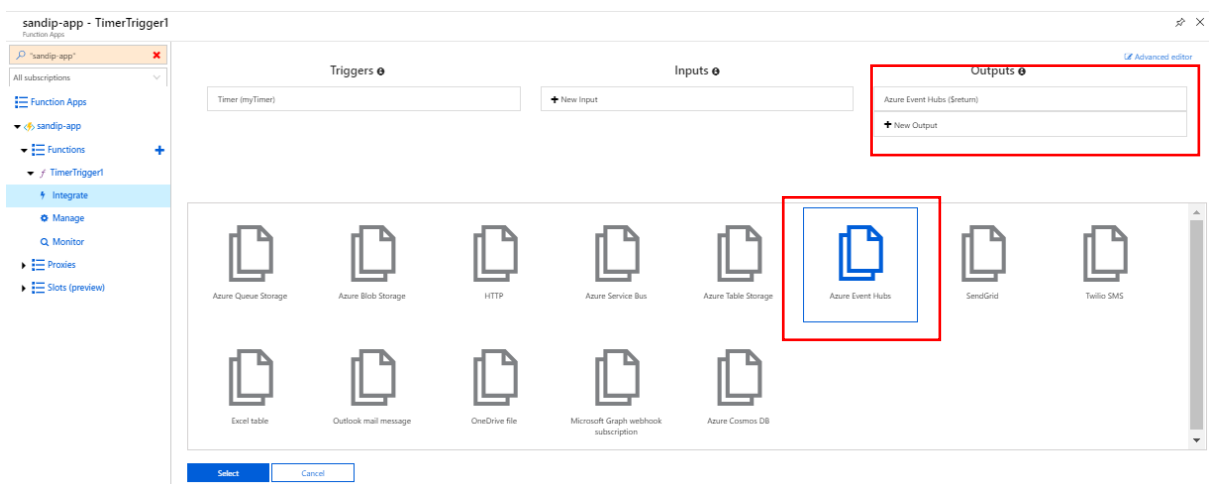
Connection Strings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below.

Hide Values Show Values

CONNECTION STRING NAME	VALUE	TYPE	SLOT SETTING	DELETE
demo-conn	Endpoint=sb://my-demo-eventhub-ins.servicebus.windows.net/SharedAccessKeyName=RootManageSharedAccessKey/SharedAcc...	Custom		

1

Now go to Integrate option and select the input/output option for this Function App. We will add Azure Event Hub as outputs. Make sure you use return value from Azure Function as Event Hub input (yellow circled as 2).



Triggers **+** Inputs **+** Outputs **+** [Advanced editor](#)

Timer (myTimer) **+ New Input**

Azure Event Hubs (\$return) **+ New Output**

Azure Event Hubs output

Event parameter name **ⓘ**
 1
☒ Use function return value

Event Hub name **ⓘ**
 2

Event Hub connection **ⓘ** [show value](#)
 new

Actions
 Create a new function triggered by this output

Connection

Event Hub IoT hub Custom

Namespace Event Hub Policy

Update the execution schedule for this timer based Function App. E.g. Here it is chosen 2 minutes' interval using CRON expression.

sandip-app - TimerTrigger1 [Advanced editor](#)

Triggers **+** Inputs **+** Outputs **+**

Timer (myTimer) **+ New Input**

Azure Event Hubs (\$return) **+ New Output**

Timer trigger **x delete**

Timestamp parameter name **ⓘ**

Schedule **ⓘ**

[Documentation](#)

Finally update the Azure Function to implement your code as per requirement. E.g. Below is simple code to return hard-coded JSON string which will be eventually sent to Event Hub as Events.

sandip-app **x**

Function Apps

sandip-app

Functions **+**

TimerTrigger1

Integrate Manage Monitor Proxies Slots (preview)

run.csx

```

1 static int invocationCount = 0;
2
3 public static string Run(TimerInfo myTimer, ILogger log)
4 {
5     return $"{{EventName: \"{Event{++invocationCount}\"\", EventTimestamp: \"{DateTime.Now.ToString(\"dd-MM-yyyy HH:mm:ss\")}\"}}";
6 }
  
```

Sample message: {"EventName":"EVENT1","EventTimestamp":"18-02-2019 12:38:00"}

Step 3 – Create the Stream Analytics Job.

The screenshot shows the 'New Stream Analytics job' configuration page in the Azure portal. The left sidebar displays 'Stream Analytics jobs' with a list containing 'my-demo-stream-analytics-job'. The main panel is titled 'New Stream Analytics job' and contains the following fields:

- Job name:** A text input field with the placeholder 'Enter job name'.
- Subscription:** A dropdown menu currently showing 'Free Trial'.
- Resource group:** A dropdown menu with 'Select existing...' and a 'Create new' link below it.
- Location:** A dropdown menu currently showing 'West Europe'.
- Hosting environment:** Two radio buttons, 'Cloud' (selected) and 'Edge'.
- Streaming units (1 to 120):** A slider control with a value of 3.

At the bottom of the configuration panel, there is a blue 'Create' button and a link for 'Automation options'.

Once the Stream Analytics job is created, add Event Hub as stream input.

The screenshot shows the 'demo-test - Inputs' configuration page for a Stream Analytics job. The left sidebar lists various settings like Overview, Activity log, Access control (IAM), Tags, and Job topology. The 'Inputs' section is selected. The main panel shows a table for adding inputs:

SOURCE TYPE	SOURCE
Event Hub	
IoT Hub	
Blob storage	

Buttons for '+ Add stream input' and '+ Add reference input' are located at the top of the table.

Make sure you choose Event message serialization format correctly (please note Stream Analytics job supports seriation format like JSON, Avro, CSV etc.) otherwise we would get error retrieving the message from Event Hub.

Event Hub

New input

✕

Input alias

Provide Event Hub settings manually

Select Event Hub from your subscriptions

Subscription

Free Trial

Event Hub namespace

my-demo-eventhub-ns

Event Hub name

Create new

Use existing

my-demo-eventhub

Event Hub policy name

RootManageSharedAccessKey

Event Hub policy key

Event Hub consumer group

Event serialization format

JSON

Encoding

UTF-8

Event compression type

None

Save

my-demo-stream-analytics-job - Inputs

Stream Analytics job

✕

Search (Ctrl+J)

«

+ Add stream input

+ Add reference input

Overview

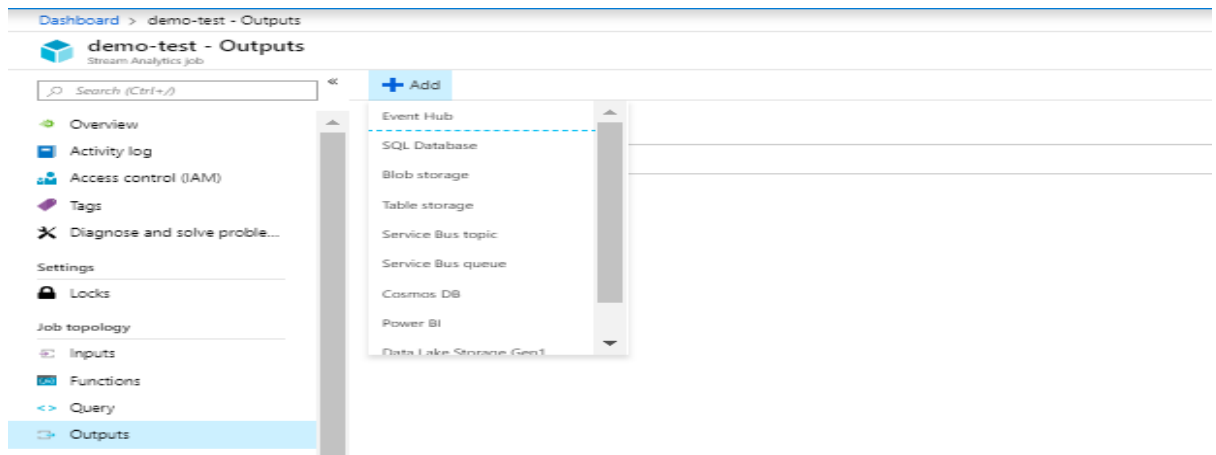
Activity log

Access control (IAM)

NAME	SOURCE TYPE	SOURCE
my-demo-event-hub-conn-alias	Stream	Event Hub

Sample dat ...

Add outputs to Stream Analytics job. In our case we would have Azure SQL DB as target.



Specify the DB details (database, credentials, target table etc.)

SQL Database

New output

×

★ Output alias

☐

Provide SQL Database settings manually

☒

Select SQL Database from your subscriptions

Subscription

Free Trial

▼

Database ⓘ

my-demo-sql-db

▼

Server name

my-demo-sql-db-server.database.windows.net

★ Username

★ Password

★ Table

☒

Merge all input partitions into a single writer

☐

Inherit partition scheme of previous query step or input

Max batch count ⓘ

10000

Save

my-demo-stream-analytics-job - Outputs

×

Stream Analytics job

Search (Ctrl+J)

«

+ Add

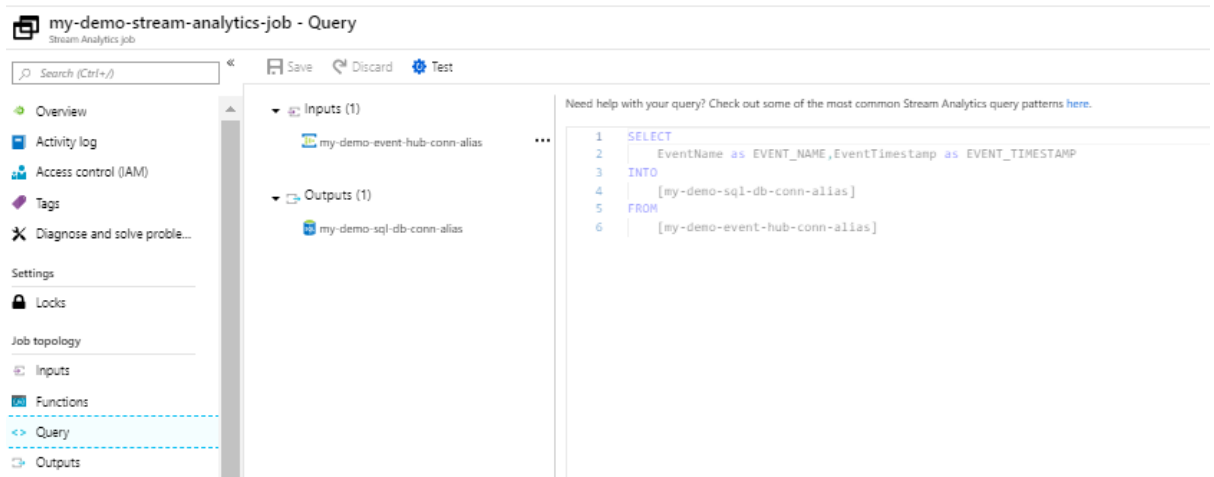
Overview

Activity log

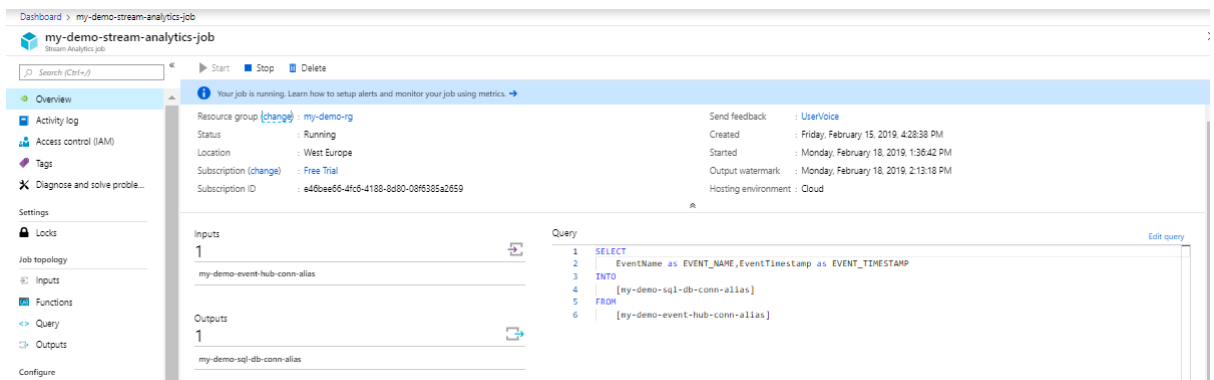
Access control (IAM)

NAME	SINK	
my-demo-sql-db-conn-alias	SQL Database	...

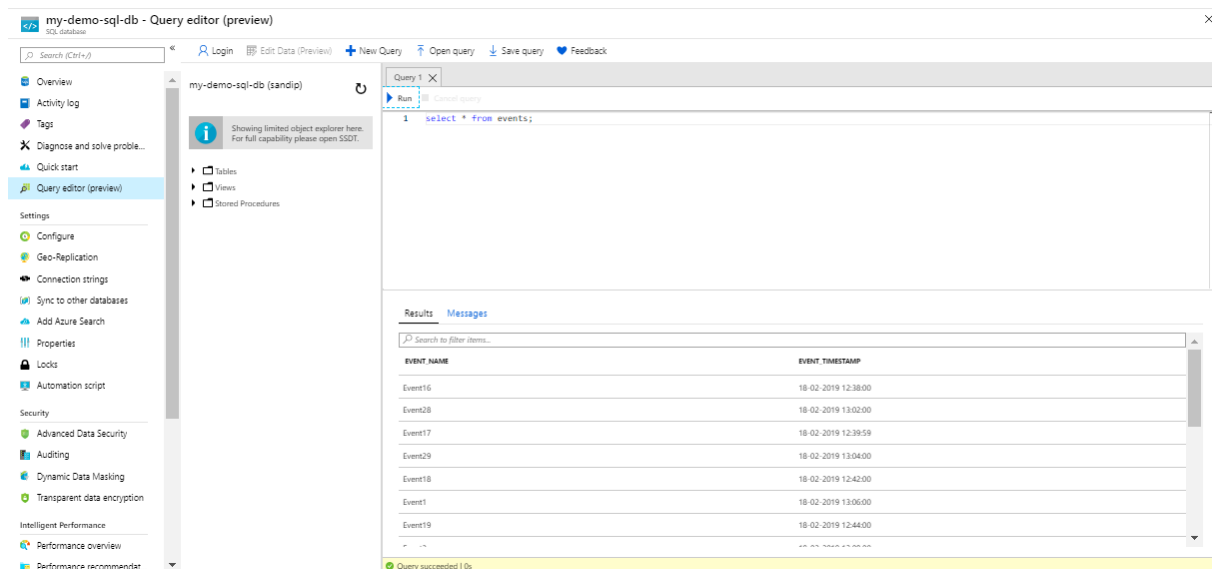
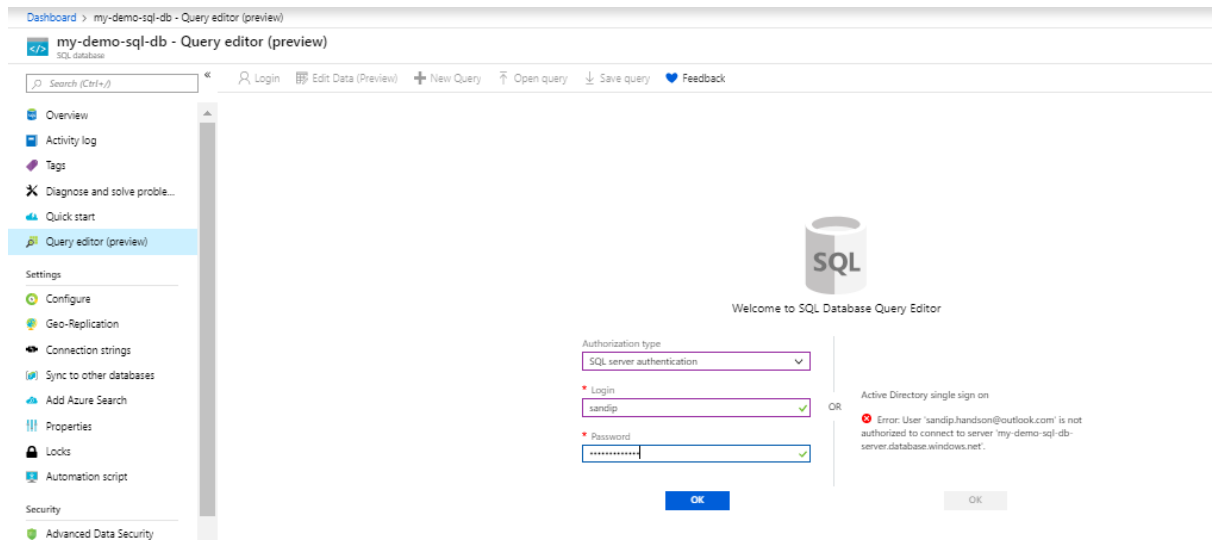
Once both input and outputs are defined for Stream Analytics job, it's time to write the query (actual business logic/mapping/transformation to populate target – to insert records into SQL DB). Please note that Azure Stream Analytics offers a SQL-like query language for performing transformations and computations over streams of events.



Once the job is ready, just start it.



Step 4 – Go to SQL DB and notice that the streaming data is getting logged into target table.



5. References

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-scheduled-function>

<https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-create>

<https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-quick-create-portal>

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-single-database-get-started>