



ALEXANDRE B BARRETO

MICROSOFT AZURE HANDS-ONS

AZURE ANALYTICS

AUTHOR

Alexandre B Barreto
barretoabb@tec.mx

IOT



GOAL

This exercise provides a practical example of how to use Azure Analytics to filter and analyze the data, creating the business report. During this exercise, the students will learn Azure IoT Analytics.

This tutorial is based on Veneri, Giacomo, and Antonio Capasso. Hands-on industrial Internet of Things: create a robust industrial IoT infrastructure using industry 4.0. Packt Publishing Ltd, 2018.

AZURE ANALYSIS SERVICE

GOAL

Azure Analysis Services is a fully managed platform as a service (PaaS) that provides enterprise-grade data models in the cloud. Use advanced mashup and modeling features to combine data from multiple data sources, define metrics, and secure your data in a single, trusted tabular semantic data model.

To use this capability, it is required to store the collected information in a highly scalable repository with which to store data for later analysis or visualization. One of the repositories supported by Azure is **Data Lake Storage**. We will use **Data Lake Storage Gen1** to store the results of our stream analytics, which we will look at in more detail in the Stream Analytics section.

Azure Data Lake Storage Gen1 is an enterprise-wide hyper-scale repository for big data analytics workloads. Azure Data Lake lets you capture data of any size, type, and ingest speed in one place for operational and exploratory analysis.

Data Lake Storage Gen1 can be accessed from Hadoop (available with the HDInsight cluster) using WebHDFS-compatible REST APIs. It was developed to allow an analysis of stored data and is tuned to work in data analysis scenarios. Data Lake Storage Gen1 includes all enterprise-class functionality: security, manageability, scalability, reliability, and availability.

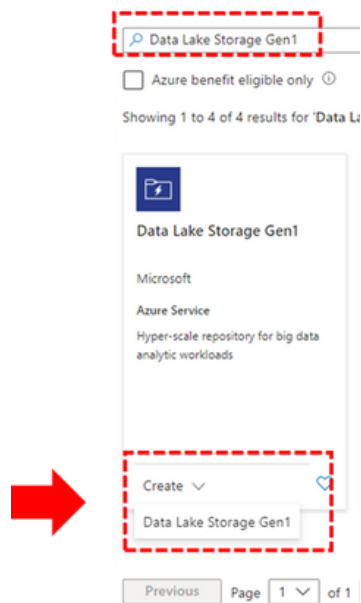


Azure Analysis Services

TASK 1

SETTING UP DATA LAKE STORAGE

To prepare our data lake repository, open the main Dashboard and create a new resource. Look for **Data Lake Storage Gen1** and click on **Create** button.



Next, you need to set the resource group (**iot_6_2_4**) and the service name (**datalake624**). The other information uses the default parameters. Click on the **Pricing** button.

Basic Pricing Encryption Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure for Students

Resource group * iot_6_2_4

Location * East US 2

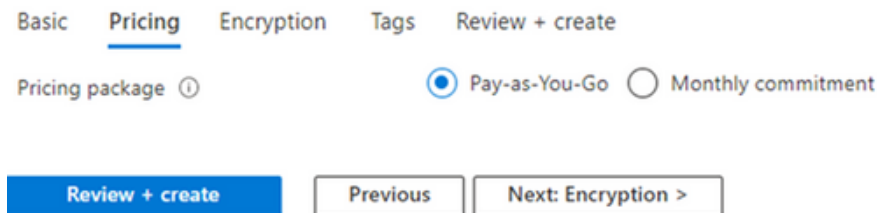
Instance details

Name * datalake624

datalake624.azuredatastore.net

Review + create Previous Next: Pricing >

In the Pricing panel, select Pay-as-You-Go and click on the **Review + Create** button. Azure will check if the configuration is ok, if it is ok a new button Create is shown. Click on it.



The screenshot shows the 'Pricing' tab selected in the Azure portal. Below the tabs, the 'Pricing package' section has two radio buttons: 'Pay-as-You-Go' (selected) and 'Monthly commitment'. At the bottom, there are three buttons: 'Review + create' (blue), 'Previous', and 'Next: Encryption >'.

STREAM ANALYTICS

Azure provides three different frameworks that can be used to work with data: Stream Analytics, Data Lake Analytics, and ML Analytics.



Microsoft Azure Stream Analytics is a serverless scalable complex event processing engine by Microsoft that enables users to develop and run real-time analytics on multiple streams of data from sources such as devices, sensors, websites, social media, and other applications. Users can set up alerts to detect anomalies, predict trends, trigger necessary workflows when certain conditions are observed, and make data available to other downstream applications and services for presentation, archiving, or further analysis.

It enables to build of streaming pipelines in minutes, running complex analytics with no need to learn new processing frameworks or provision virtual machines (VMs) or clusters. It uses familiar SQL language that is extensible with JavaScript and C# custom code for more advanced use cases. Easily enable scenarios like low-latency dashboarding, streaming ETL, and real-time alerting with one-click integration across sources and sinks.

TASK 2 USING STREAMING ANALYTICS

To use this feature, open the main Dashboard and create a new resource. Look for **Streaming Analytics Job** and click on **Create** button. As in the previous task, we set some properties and keep the default values in another. The resource group is the same (**iot_6_2_4**) and the instance name you set is **stream_job_624**. Also, confirm that the hosting environment is on Cloud.

New Stream Analytics job ...

⚠ Changes on this step may reset later selections you have made. Review all options prior to deployment.

Azure SQL DB & Cosmos DB, remote monitoring, predictive maintenance, and more. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure for Students ▼

Resource group * ⓘ iot_6_2_4 ▼

[Create new](#)

Instance details

Name * stream_job_624 ✓

Region * ⓘ East US ▼

Hosting environment ⓘ

☒ Cloud

☐ Edge

Streaming unit details

Streaming units (SUs) represents the computing resources that are allocated to execute a Stream Analytics job. The higher the number of SUs, the more CPU and memory resources are allocated for your job. The number of SUs can be modified once you create the job. You will be charged for the job's Streaming Units only when the job runs.

[Learn more](#)

Streaming units * 3 ▼

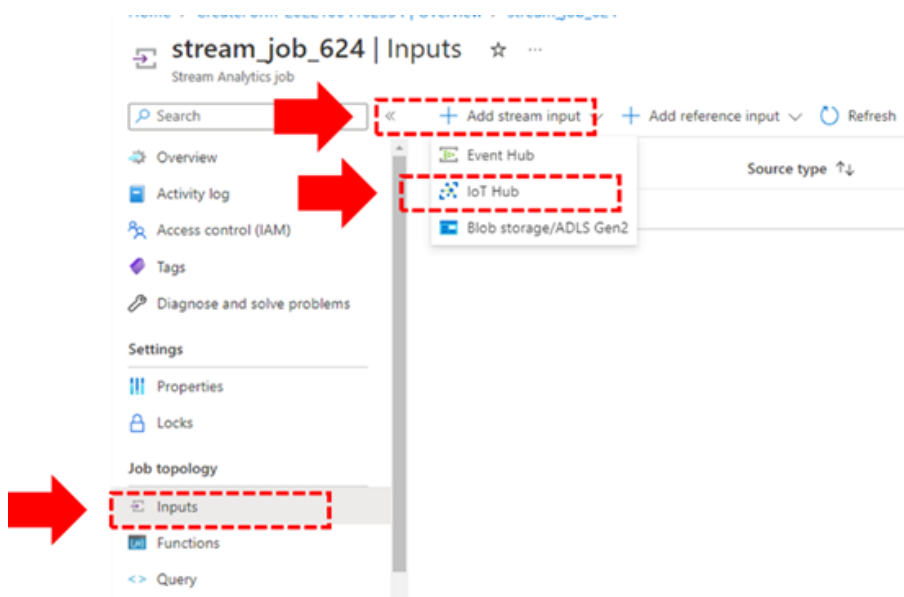
[Review + create](#)

[< Previous](#)

[Next : Storage >](#)

Click on the **Review + Create** button. Azure will check if the configuration is ok, if it is ok a new button Create is shown. Click on it.

Once this is done, we need to define an input. From the Stream Analytics page, click on our job, then click on **+Add stream input** from the Inputs tab to add our IoT Hub input.



Define the name of the input alias to "iothub" and assure that the IoT Hub name is correct. After it, click on the **Save** button.

IoT Hub

×

New input

Input alias ^{*}

iothub ✓

☐ Provide IoT Hub settings manually

☒ Select IoT Hub from your subscriptions

Subscription

Azure for Students ▼

IoT Hub ^{*} ⓘ

iot-hub-624 ▼

Consumer group ^{*} ⓘ

\$Default ▼

Shared access policy name ^{*} ⓘ

iothubowner ▼

Shared access policy key ⓘ

.....

Endpoint ⓘ

Messaging ▼

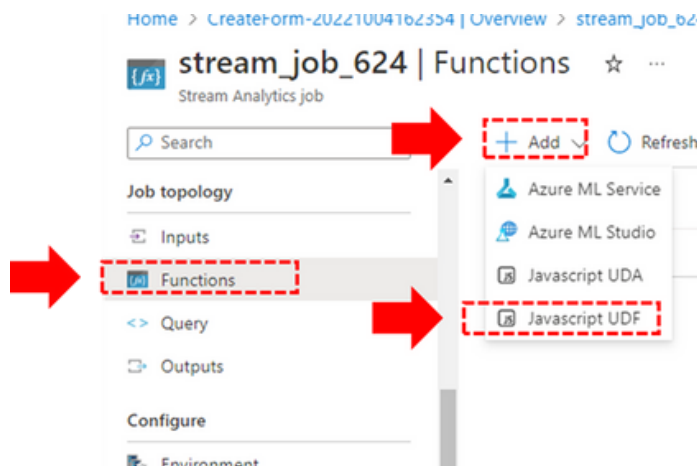
Partition key ⓘ

Event serialization format ^{*} ⓘ

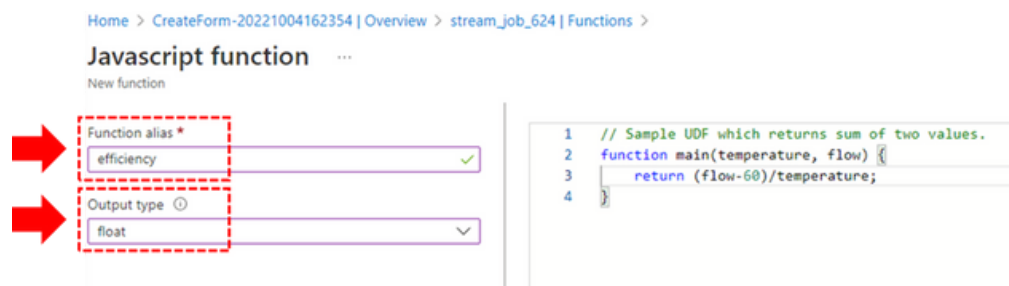
JSON ▼

Save

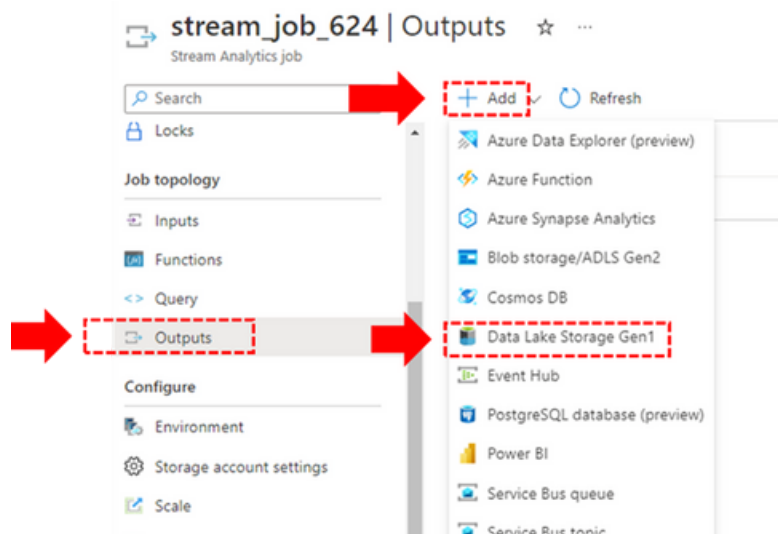
The next step is the creation of a simple **user-defined function (UDF)** to calculate the efficiency, we can click on the Functions tab in the menu on the left, and create a JavaScript function, called efficiency.



The next step is the definition of the JavaScript code and sets that the output is float.



Following, we connect our output. From the Outputs tab, click on **+Add** and then **Data Lake Store**.



We must provide a name (**efficiencyoutput**), and a path prefix of **out/logs/[date]/[time]**. We need to change the Event serialization format option to CSV and the Delimiter option to tab. After that, click on **Save**.

The screenshot shows the 'Data Lake Storage Gen1' configuration window. Four red arrows point to the following fields:

- Output alias ***: Set to 'efficiencyoutput'.
- Account name ***: Set to 'datalake624'.
- Path prefix pattern ***: Set to 'out/logs/[date]/[time]'.
- Event serialization format ***: Set to 'tab'.

Other visible fields include:

- Subscription**: Azure for Students
- Authentication mode**: Create system assigned managed identity
- Date format**: YYYY/MM/DD
- Time format**: HH
- Encoding**: UTF-8

A 'Save' button is at the bottom left. A note at the bottom right states: 'The selected resource and the stream analytics job are located in different regions. You will be billed for cross data hub usage.'

We are now ready to connect our input with our UDF and our output. From the Query tab, we can define this as follows. Remember to save the query.

SELECT

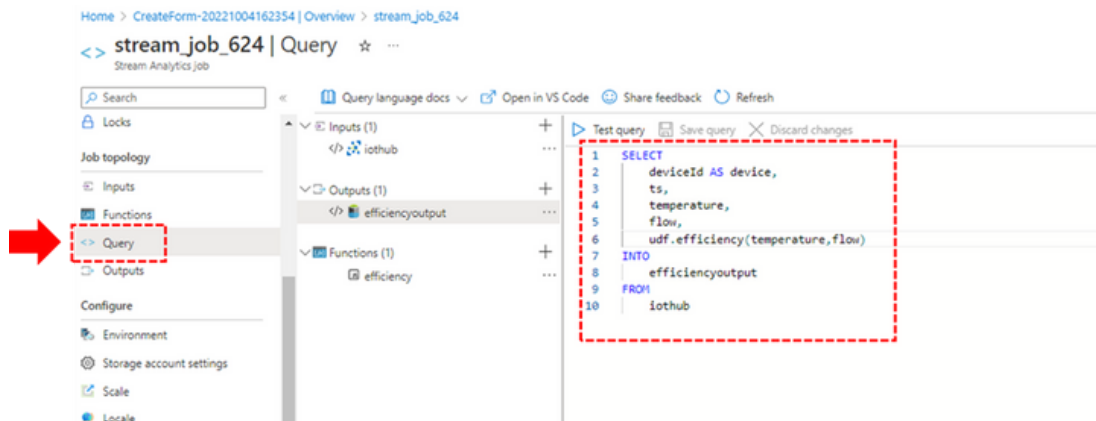
deviceId AS device,
ts,
temperature,
flow,
udf.efficiency(temperature,flow)

INTO

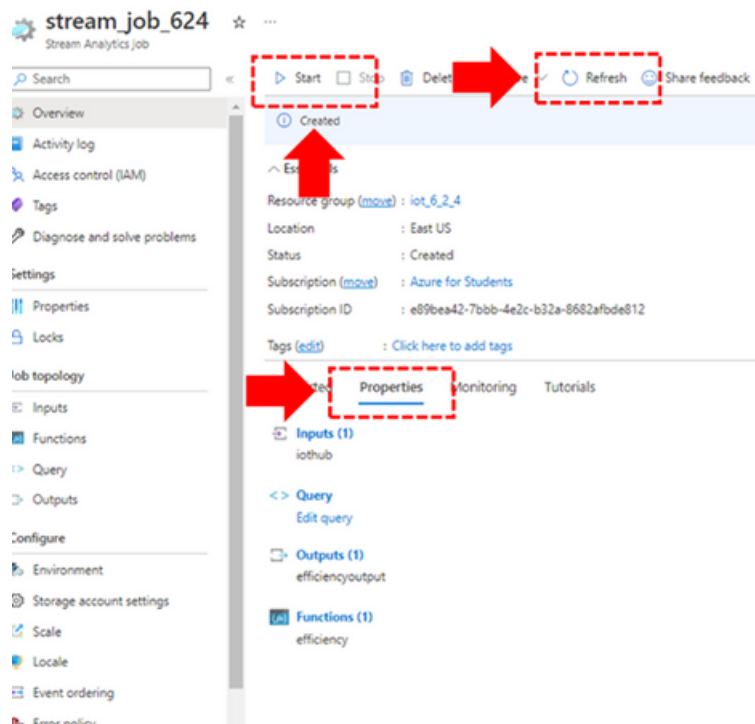
efficiencyoutput

FROM

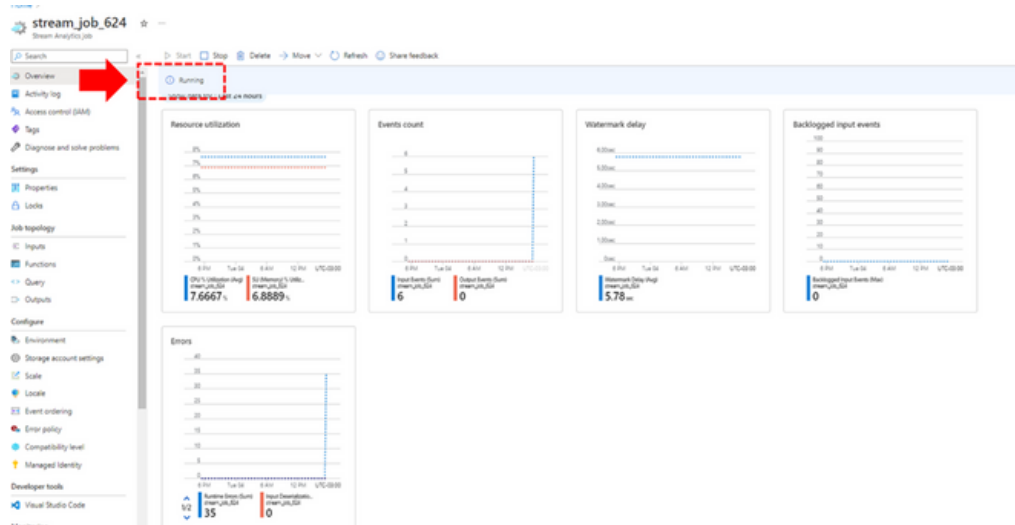
iothub



Return to the Dashboard of the Streaming Job, click on the **Property** panel, and check if the Input, Output, or Function appears. If they are not shown, click on **Refresh**. When the configuration is OK, click on the **Start** button to initiate the job.



After you click on the **Start** button the job will be prepared. You need to confirm and click on the new **Start** button that appears on the right side of the Dashboard. After you click the button, the Dashboard is changed. If you stop the JavaScript app that you created in the last exercise, re-run and analyze the Dashboard information. If the information does not change, click again on the **Refresh** button.



After you test the functionality, remember to click on the Stop button, to stop the analytics from running to avoid unwanted billing, as in the JavaScript app.