

Internet of Things Workshop

Lab 3

Azure Stream Analytics

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 9/16/2016 | Steve Busby | 1.0 | Initial draft |
|  |  |  |  |

Table of Contents

[Change Record 2](#_Toc461796314)

[Introduction 4](#_Toc461796315)

[1.) Create a Stream Analytics Job 5](#_Toc461796316)

[2.) Defining the Input 7](#_Toc461796317)

[3.) Creating the output EventHub queue 9](#_Toc461796318)

[4) Defining the Stream Analytics Output 13](#_Toc461796319)

[5) Writing the Streaming Query 15](#_Toc461796320)

Introduction

This lab is focused on configuring an Azure Stream Analytics for filtering the incoming device data stream for alert conditions. An alert condition, in the context of this lab, is a high temperature violation of a defined temperature threshold This lab is the third in a series that walks through building an end-to-end Internet of Things prototype for doing temperature monitoring.

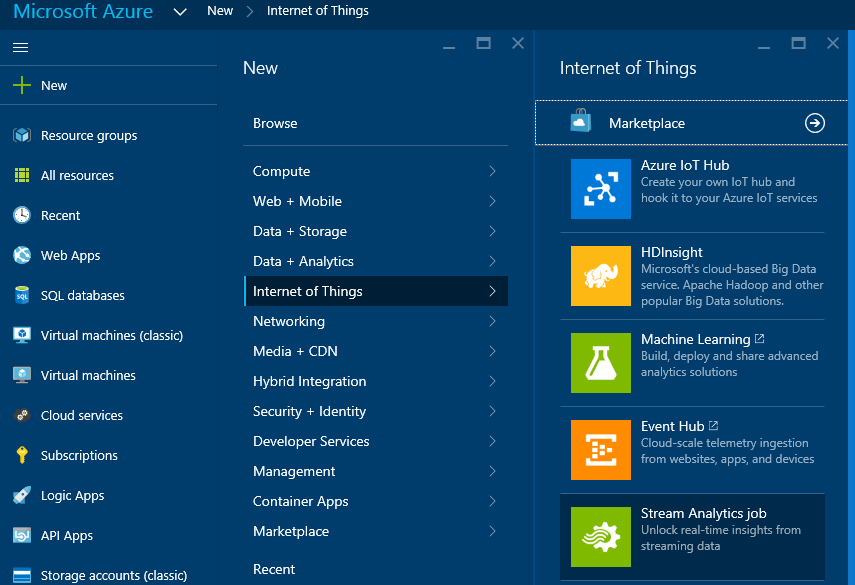
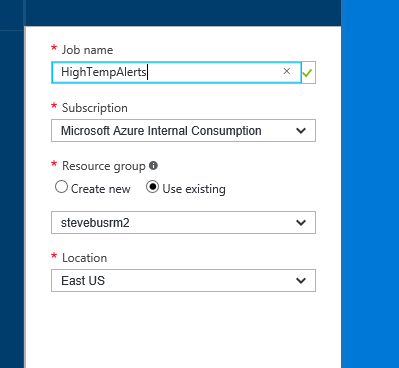
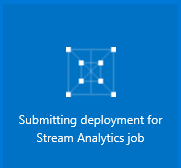
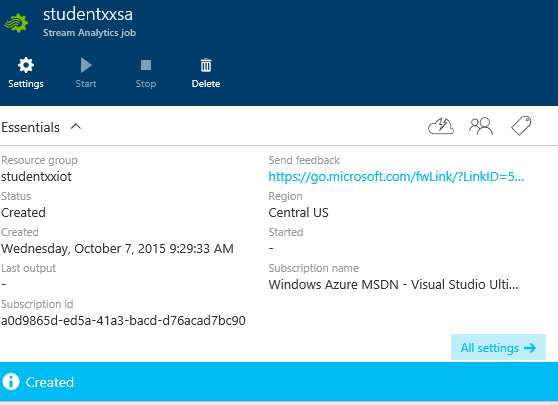
In this series of labs, you will:

1. Create and navigate the Azure IoT Remote Monitoring Pre-Configured Solution (RM-PCS)
2. Create a device to read a temperature and humidity sensor (and optionally, a light sensor) and send that data to the RM-PCS for display
3. Create a Stream Analytics job that looks for ‘high temperature’ alerts and outputs that alert to a queue for further processing
4. Create an Azure Function that takes that alert, and sends a command to the device to turn on or off an LED depending on the alert condition.

At the end of this lab you will have written the streaming query to identify alerts and put them in a queue for further processing.

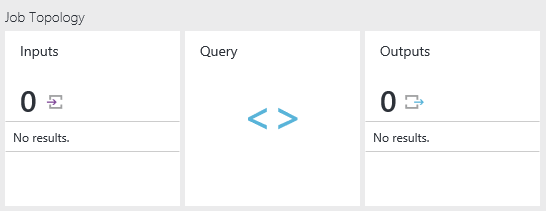
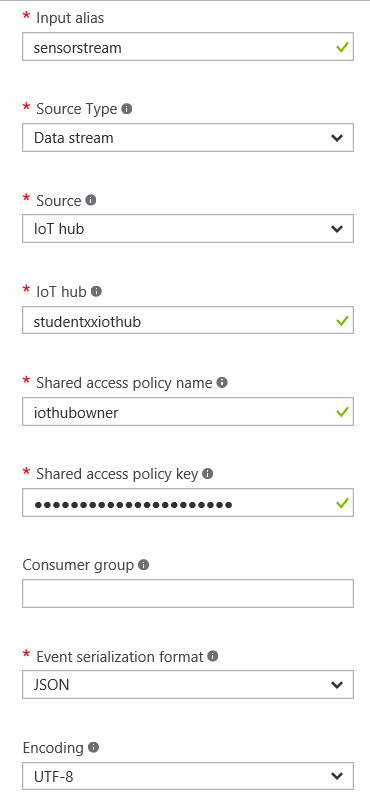
1.) Create a Stream Analytics Job

In this section you will create a new Stream Analytics Job and validate events are flowing properly.

1. Using a web browser navigate to portal.azure.com
2. Create a new Stream Analytics Job from the portal navigation using + New -> Internet of Things -> Stream Analytics Job.  
   
3. You’ll be presented with the create pane for Azure Stream Analytics. For the Job Name field, enter a name for your job that you can remember. For “Resource Group”, choose “use existing”. In the drop down list box, you should see a resource group named the same thing as your chosen solution name from lab 1. Choose that. Make sure that the “location” is the same as chosen in Lab 1 (e.g. “East US”, most likely). Click on the “Create” button.
4. You will be presented with a progress indicator while the job is creating.  
   
5. Upon completion you’ll be presented the main blade for your stream analytics job.  
   

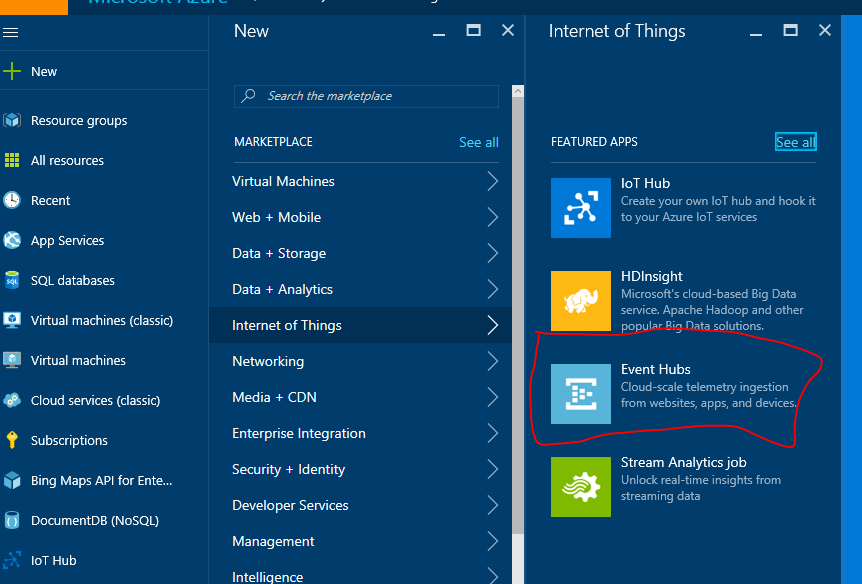
2.) Defining the Input

In this section we’ll define the input to the Azure Stream Analytics job. This configuration binds to the IoTHub that was automatically created for you as part of your RM-PCS solution.

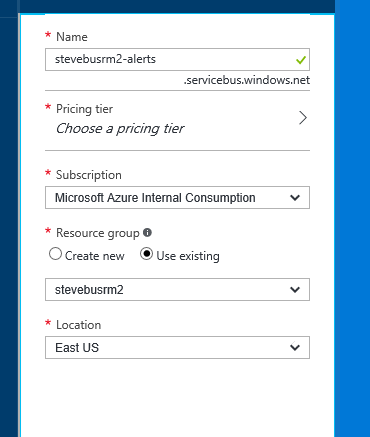
1. Under “Job Topology” click on “Inputs”.  
   
2. You’ll be presented a blank list of inputs. Click the Add button .
3. On the “New input” screen complete the details based on your specific solution names. A few notes:
   1. For input alias, enter ‘sensorstream’ (we will use this later in the query, so if you name it something different, you’ll need to change the query later)
   2. For Source Type, choose Data stream
   3. For Source, choose IoT Hub
   4. Since we have only one IoTHub in our subscription at this point, it should auto-fill the rest of the fields.  
      
4. Click the button.

3.) Creating the output EventHub queue

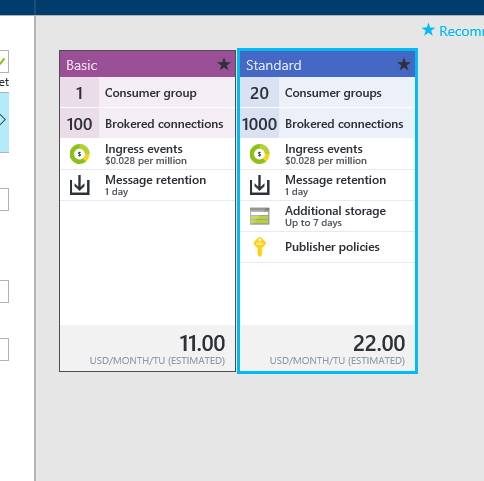
1. In this section, we will create the EventHub that will act as in internal queue for Alerts. This queue will be the output of our Stream Analytics Job, and the input of our Azure Function we will write in Lab 4 to process the alerts
2. In the Azure Portal, choose New -> Internet of Things -> EventHub



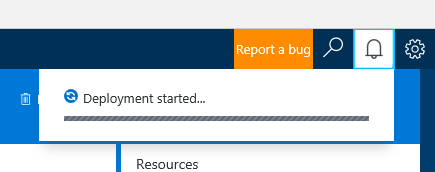
1. The first thing that we need to create is a namespace to hold our eventhub. A namespace is essentially just a URI that we will use to reference our eventhub from the public internet (although in our lab use case, we will only be referencing it internally within Azure)
2. On the “Create Namespace” blade, enter a unique name for your eventhub namespace. A good practice, for this lab, would be to just use the name of your solution chosen in Lab 1 and add “-Alerts” on the end. Choose the same existing resource group as you chose in the previous step and ensure the location is the same as previously chosen



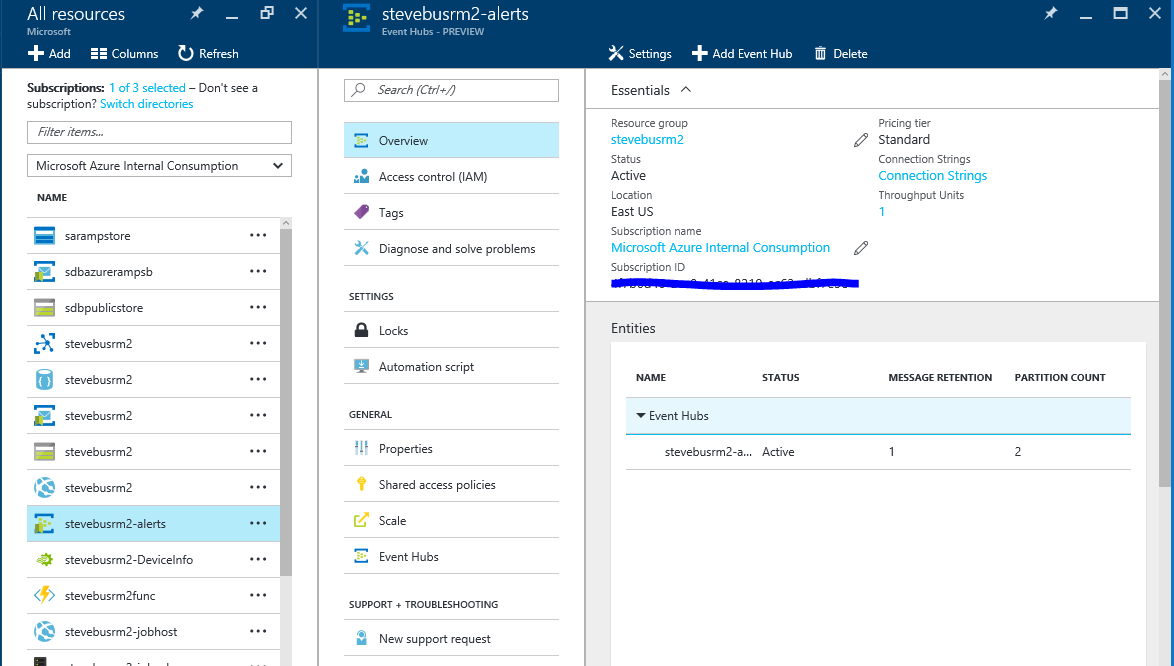
1. Click on the arrow next to “choose a pricing tier”



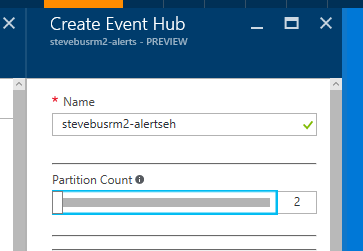
1. Choose “standard” and hit the ‘select’ button
2. Hit the “create” button to create your EventHub. You will see a pop-up that says “deployment started”, but it will go away. Click on the little bell shaped icon to drop it back down and you can see the status. Wait until the deployment is complete before starting the next step



1. Now we need to add an EventHub to our namespace. From the left-hand Nav, choose “Resource Groups”, then click on the resource group that was created for your solution in Lab 1, and choose the EventHub namespace you just created.



1. Click “Add EventHub”
2. For the EventHub name, use the same name as your namespace, just append ‘eh’ on the end of it (technically you can name it anything you want, but this makes it easier for the lab)

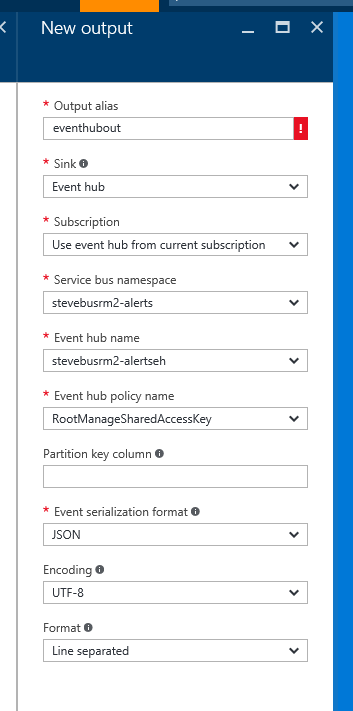


1. Take the defaults for all other settings and click “Create”
2. Wait until the eventhub is created before proceeding to the next step.

4) Defining the Stream Analytics Output

In this section, we will go back to our Azure Stream Analytics (ASA) job and ‘wire up’ the EventHub created in Step 3 as the output of our job

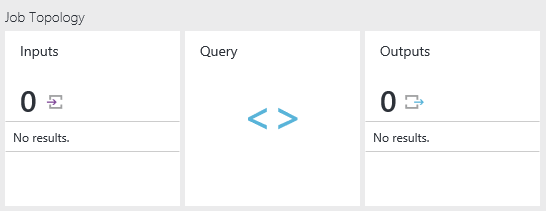
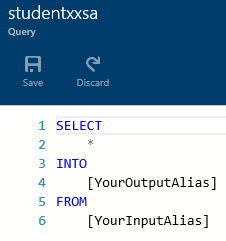
1. In the Azure Portal, on the left hand nav, click on “Stream Analytics Jobs” and then open the job you created above
2. On the blade for your ASA job, click “Outputs” and then click “+ Add” to add an output
3. On the “New Output” blade, give your output a name (if you use ‘eventhubout’, you won’t need to edit the query in the next step). Choose “EventHub” as the type of Sink. Choose “use event hub from the current subscription”. Choose the Service Bus Namespace and Event Hub Name created above. The rest of the fields should auto-fill.



1. Accept the rest of the defaults and hit ‘Create’. Wait until the output is created to move to the next step.
2. Now that we have wired up an IotHub as an input, and EventHub queue as an Output, we are now ready to specify our query to identify the Alerts we are looking for.

5) Writing the Streaming Query

In this section you will write a streaming query that defines how you identify High Temperature Alerts. Stream Analytics will read the events as they stream in from the device and execute this query. Any time an event satisfies the query, an output row will be generated and sent as a message to the downstream EventHub.

1. Click on the “Query” button from the “Job Topology” screen.  
   
2. You’ll now be presented a basic starting point for a query.  
   
3. For our query, we will be looking for cases where the temperature data flowing from the device transitions from a LOW to a HIGH state, or back. We will use this in the next step to send a command to the device to toggle a “high temp warning” LED on and off. The following query will provide this capability to us.

-- this section creates a subquery that a) filters out the 'deviceinfo' messages (by checking for a missing ObjectType property), b) creates a column called "TempState" that is 'HIGH' if temperature is > 80, otherwise 'LOW',and c) adds a timestamp to the output

WITH TemperatureCTE as (

SELECT

DeviceId,

Temperature,

CASE

WHEN cast(Temperature as bigint) >= 80 THEN 'HIGH'

ELSE 'LOW'

END as TempState,

System.TimeStamp as eventdatetime

FROM sensorstream

WHERE

ObjectType is NULL

AND

Temperature is not NULL

AND

TRY\_CAST( Temperature AS bigint) IS NOT NULL)

-- this query takes the output of the subquery above and looks for cases in which the "temp state" in the current row is different than the "temp state" in the previous row (i.e. we've transitioned from 'high to low' or 'low to high'). The LAG function gives us access to the previous row for a particular device (the PARTITION BY DeviceID part). This kicks out a row into the output queue

SELECT \*

INTO eventhubout

FROM TemperatureCTE

WHERE

LAG(TempState, 1) OVER (PARTITION BY DeviceID LIMIT DURATION(minute, 10)) <> TempState

1. Click on the  button to store the query.
2. We are done with the Stream Analytics job, but we will not start it running until we have our Azure Function in place to process the events.