## References

* Service Bus

<https://azure.microsoft.com/en-us/services/service-bus/>

* How to send cloud-to-device messages with IoT Hub and .Net

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-csharp-csharp-c2d>

* Service Bus Explorer

<https://github.com/paolosalvatori/ServiceBusExplorer>

## Requirements

* Complete the second hands-on lab.
* NuGet packages
  + Newtonsoft.Json for JSON in C#
  + WindowsAzure.ServiceBus for Service Bus
  + Microsoft.Azure.Devices (Service SDK for Azure IoT Devices) for Cloud to Device message of IoT Hub

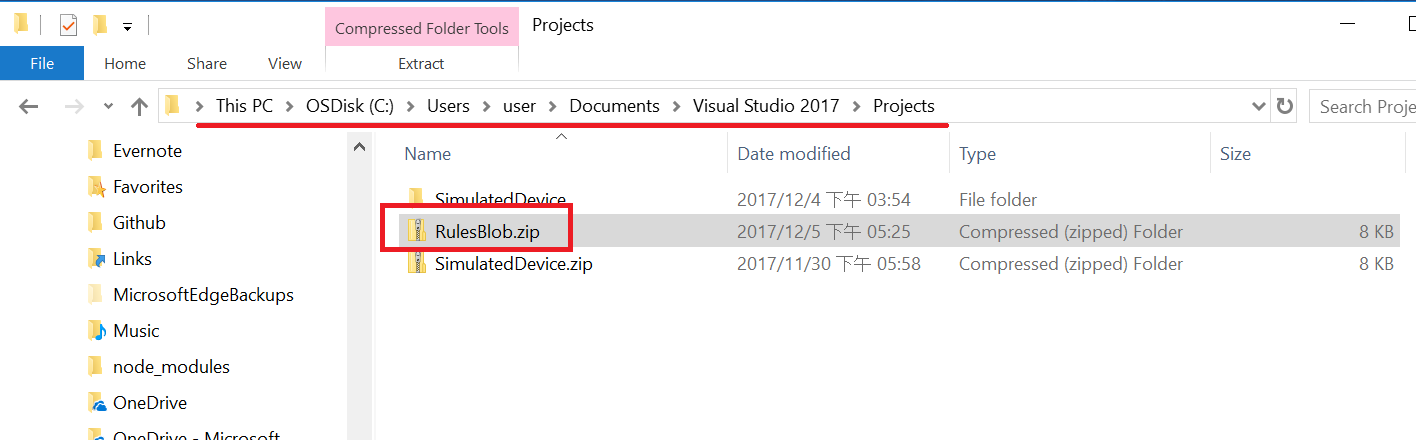
## Goals

* Create and upload a device rule in Blob
* Create a **Service Bus** and its receiver
  + Receive the alarm message from a Stream Analytics Job.
  + Send Cloud to Device command to the corresponding devices.
* Create an **Azure Stream Analytics Job**
  + Processing the telemetry data with device rules.
  + Trigger a message to **Service Bus** when the temperature alert was detected.

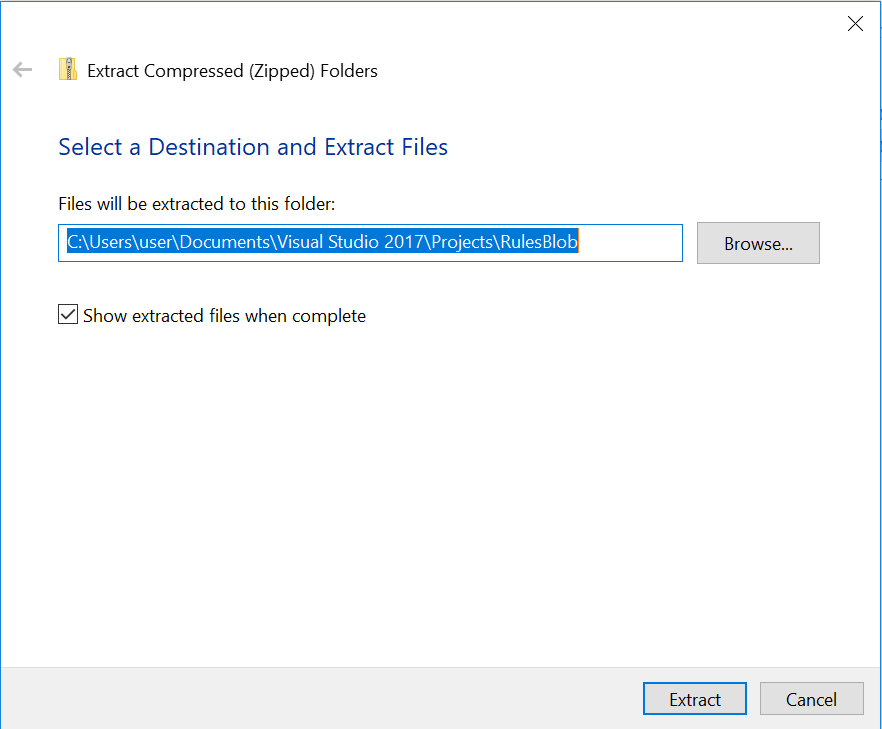
## Step 1: Create and upload a device rule in Blob

* Build the Console App for Rules Blob
  + Copy the **RulesBlob.zip** from **03-HOL/Sample** folder to the projects of Visual Studio.

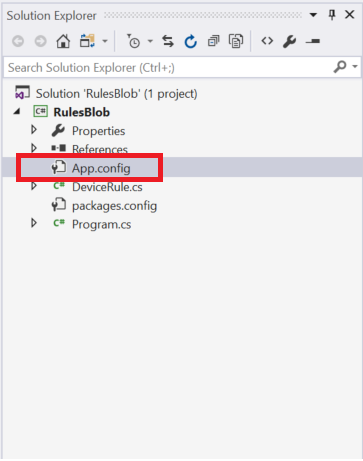
Path: **C:\Users\<username>\Documents\Visual Studio 2017\Projects\**



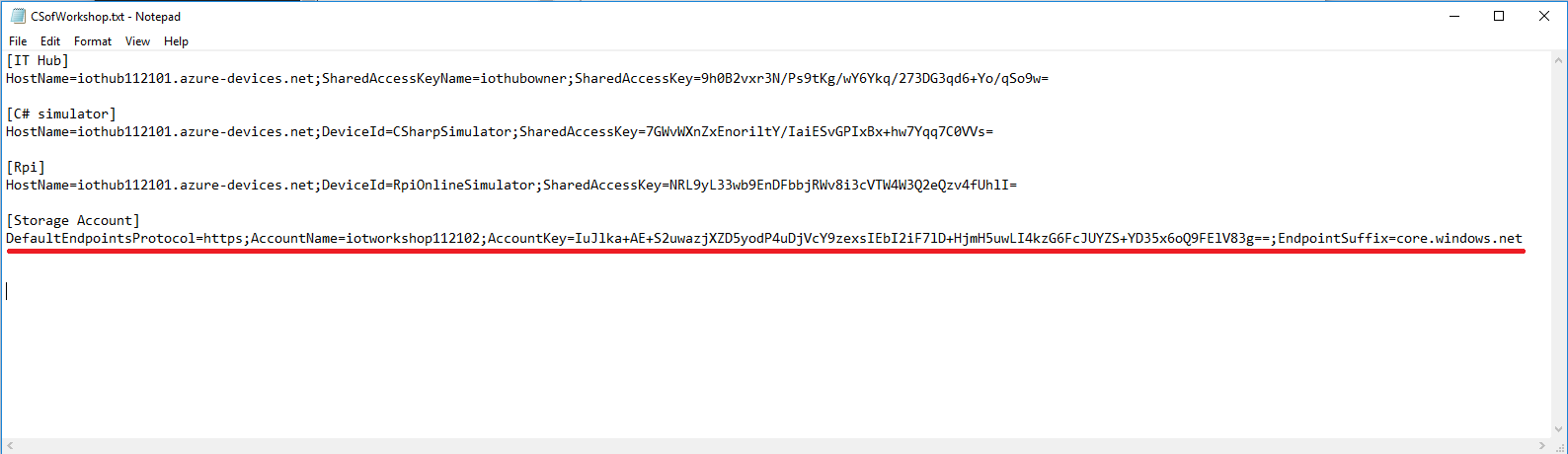
* + Unzip



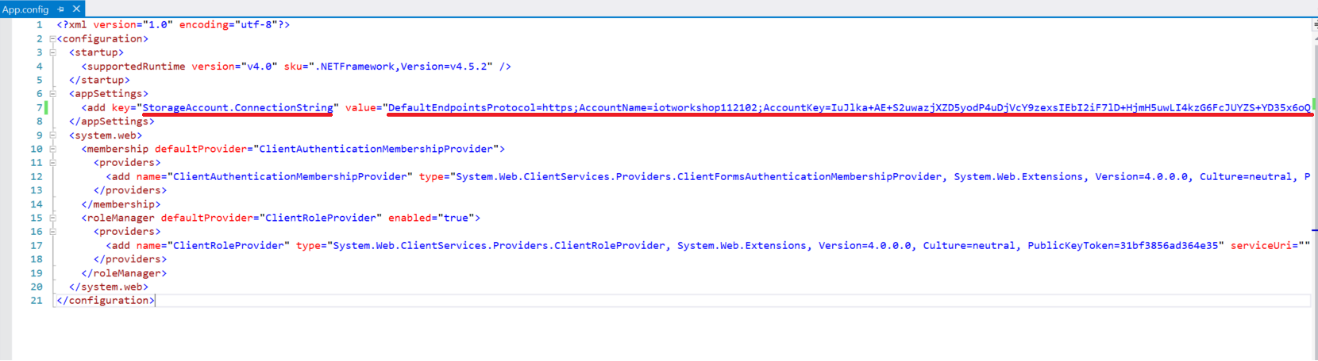
* + Open the RuleBlob.sln file in Visual Studio.
  + Open the App.config



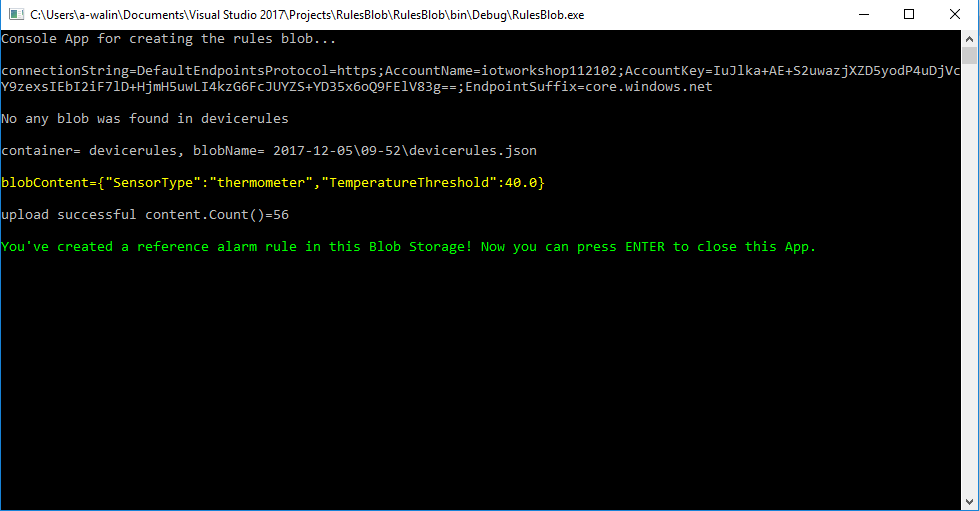
* + Get the **Connection String** of **Storage Account** from the previous you wrote.



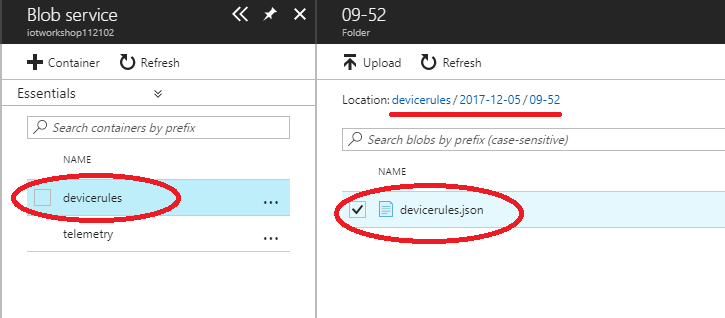
* + Update it into the value of StorageAccount.ConnectionString in App.config.



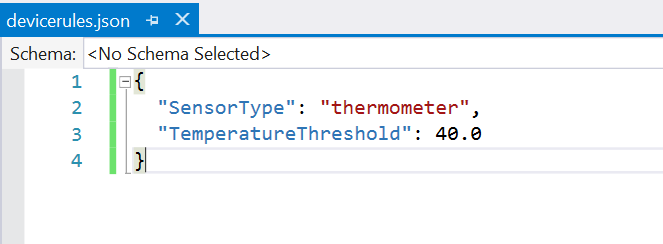
* + Press **F5** to build and debug the project
  + Check the output of console



* Download and open this Blob file. (Container: devicerules)

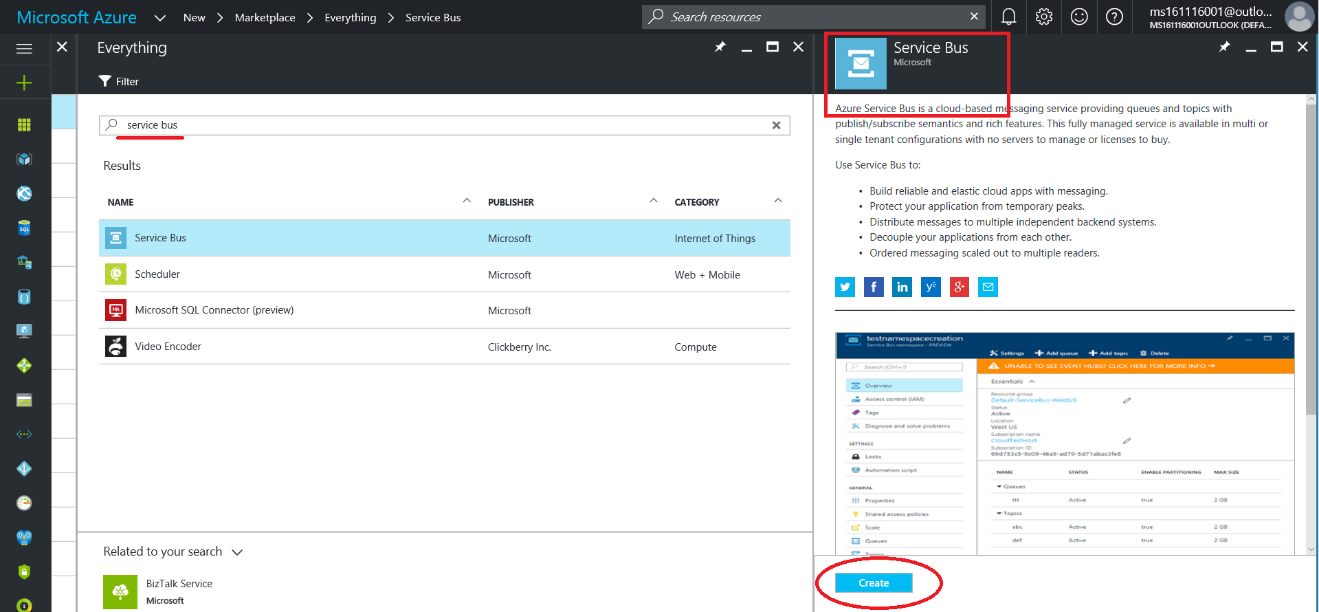


* + JSON Content

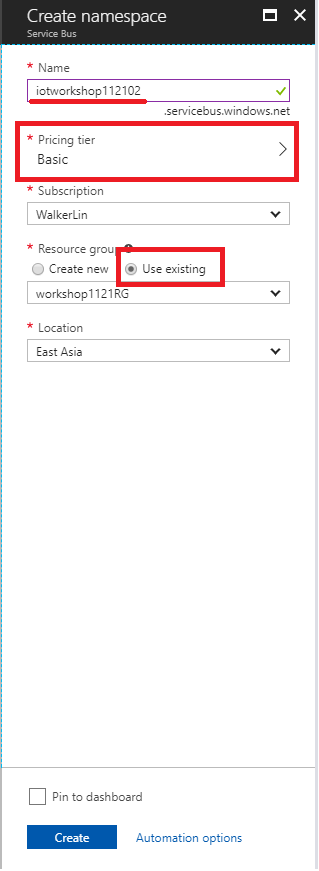


## Step 2: Create a Service Bus for Temperature Alert

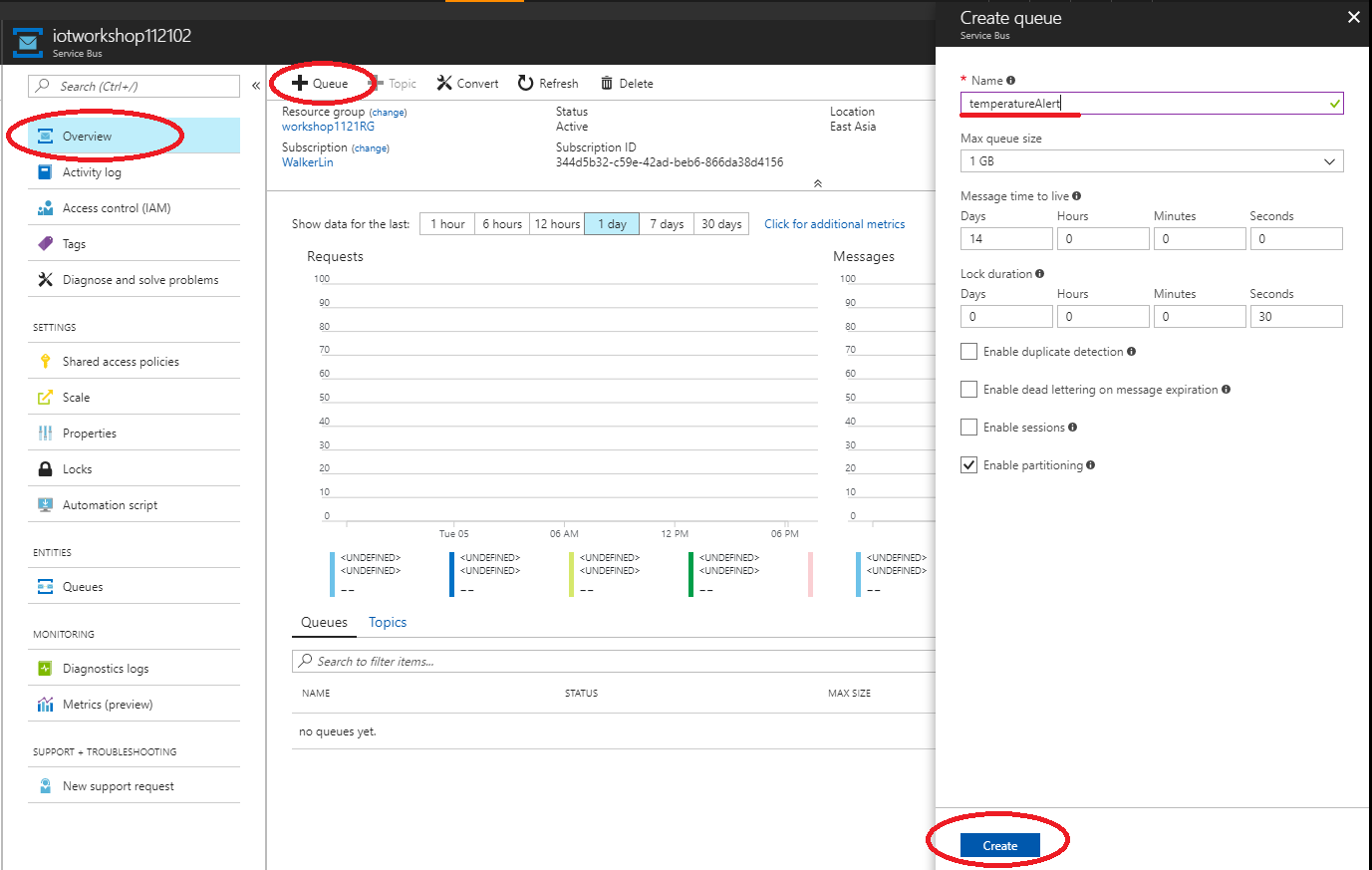
* Search and **Create** the **Service Bus**



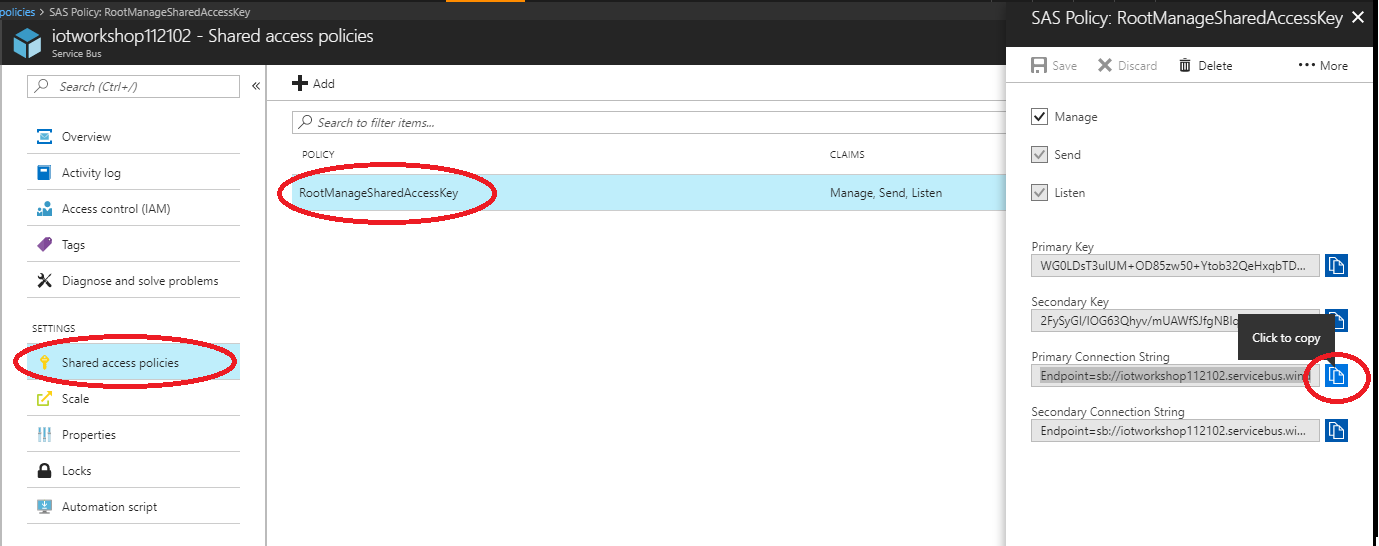
* Create a namespace
  + Name: **iotworkshop112102** (for example)
  + Pricing tier: **Basic**
  + Use the existing **Resource Group**.



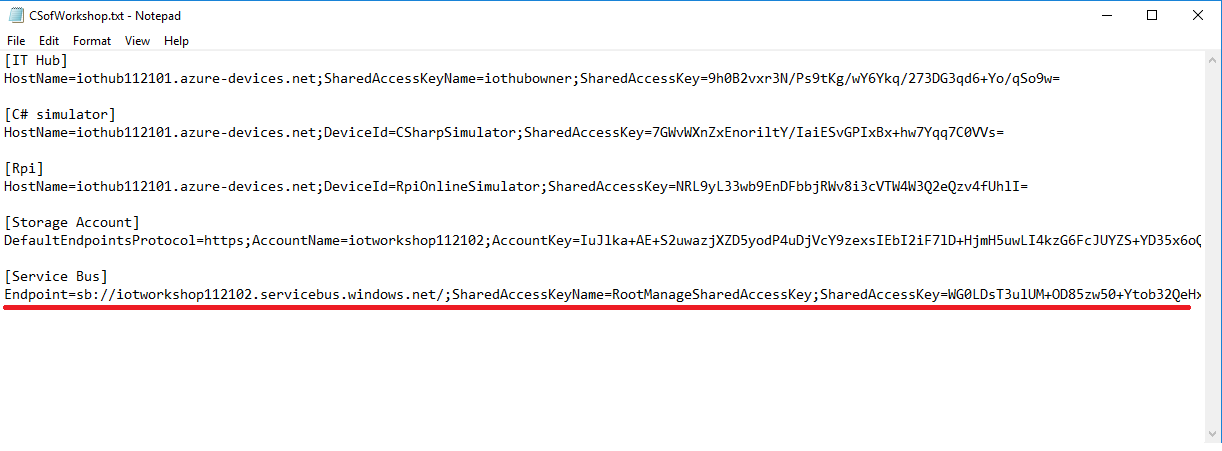
* Create a **Queue**
  + Name: **temperatureAlert** (must be fixed)



* Get the **connection string of Service Bus**

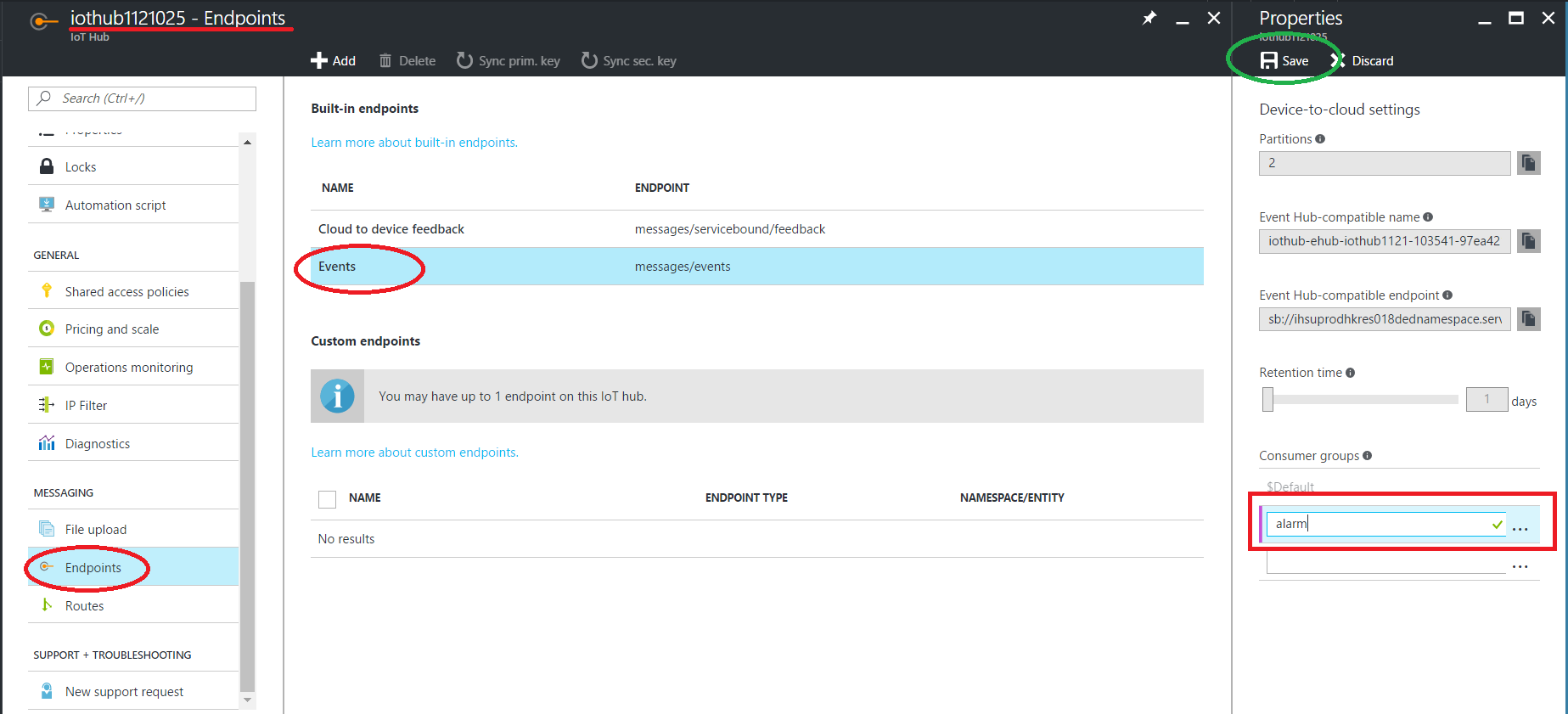


* Save the **connection string of Service Bus** for the later used.

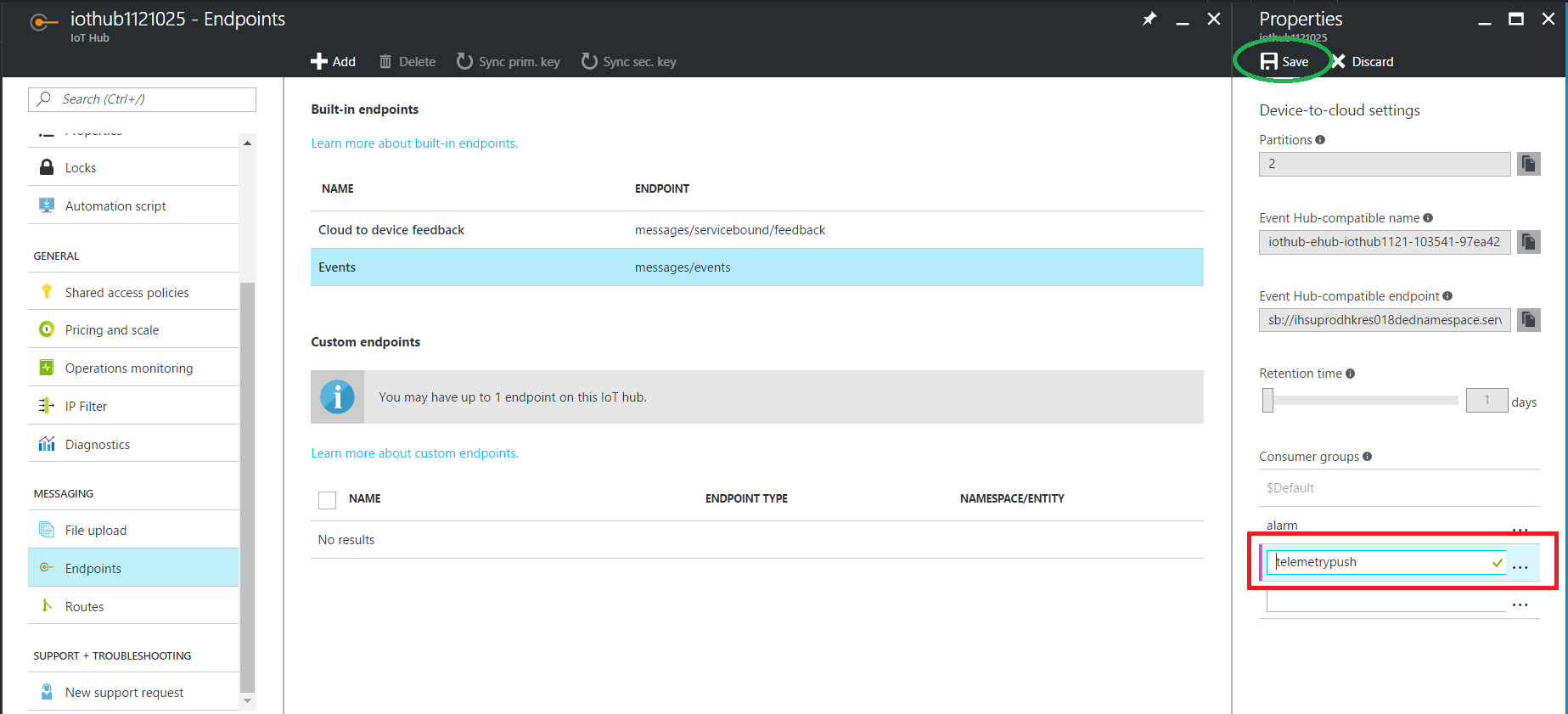


## Step 3: Create new consumer groups of IoT Hub

* Find your **IoT Hub**
* Navigate to “**Endpoints”** of MESSAGEING, then click the “**Events**” to add a new consumer group.
* Add a new one for Alarm of Stream Analytics Job
  + Name: **alarm** (must be fixed)



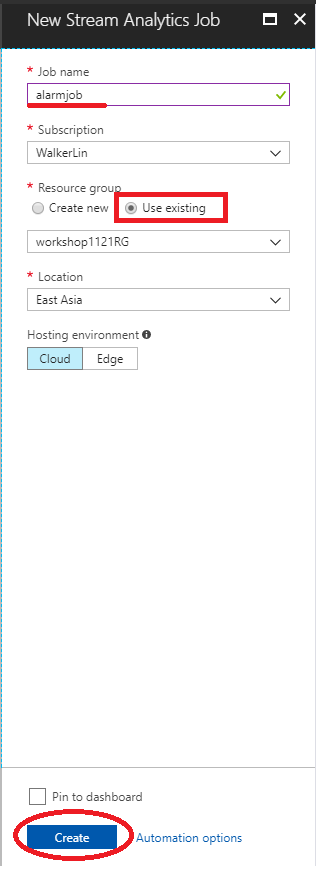
* Add another consumer group for the telemetry push by Event Processor Host, and it will be used in the next HOL.
  + Name: **telemetrypush** (must be fixed)



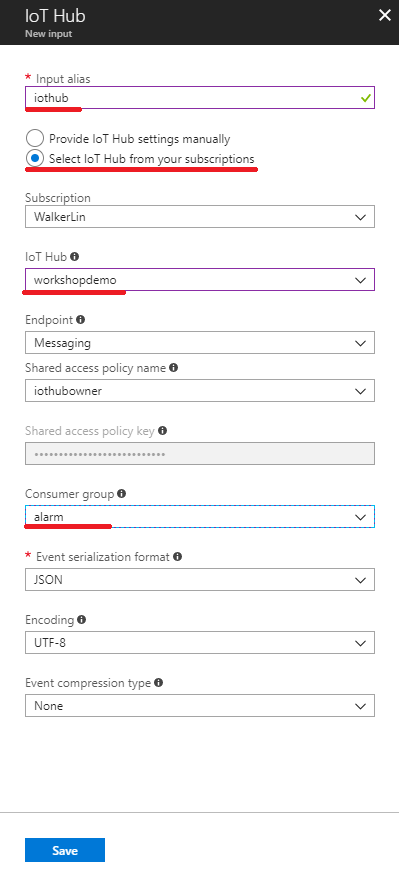
* + Don’t forget to **Save** the changed properties.

## Step 4: Create a Stream Analytics Job for Alarm Message

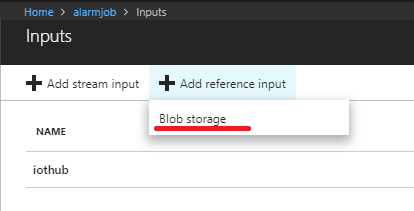
* New a **Stream Analytics Job**
  + Job Name: **alarmjob**



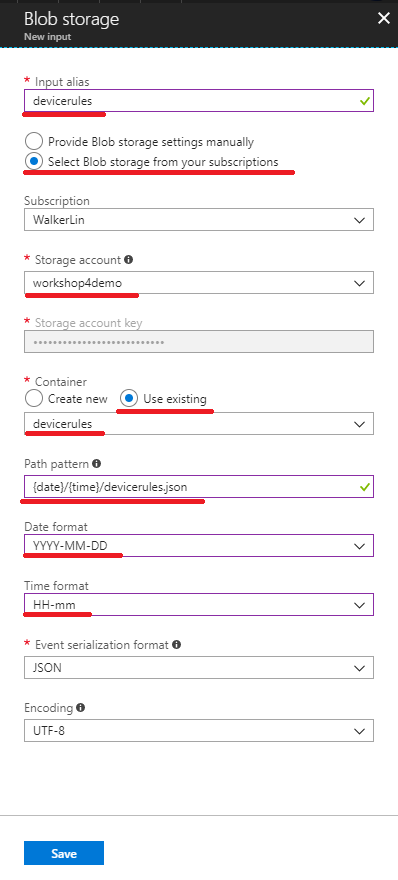
* + Add an **IoT Hub** as the **input**
    - Input alias: **iothub** (must be fixed)
    - IoT Hub: select the used IoT Hub
    - Consumer group: **alarm (not $Default)**
    - The others should be set as below



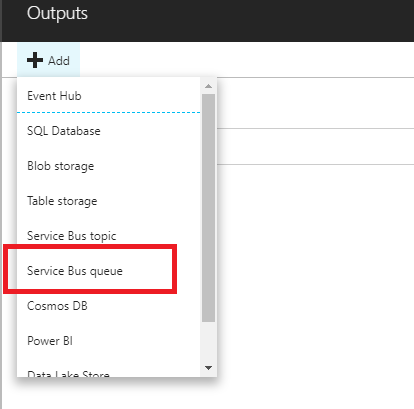
* + Add **reference input** from **Blob storage**



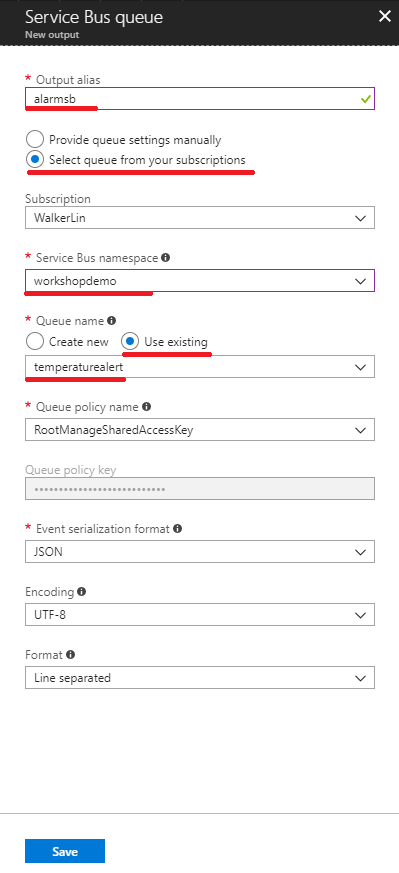
* + Input
    - Input alias: **devicerules** (must be fixed)
    - Storage account: select the storage account as you provisioned
    - Container: **devicerules** (select this one we created from step 1)
    - Path pattern: **{date}/{time}/devicerules.json** (must be fixed)
    - Date format: **YYYY-MM-DD** (This format must be fixed)
    - Time format: **HH-mm** (This format must be fixed)
    - **JSON** and **UTF-8** also should be fixed.



* + Add a Service Bus queue as the output



* + Output
    - Output alias: **alarmsb** (must be fixed)
    - Service bus namespace: select what you created
    - Queue name: **temperatureAlert**



* + Add a **Query** for processing the alarm message
    - Paste the following script to the query of Stream Analytics.

WITH StreamAvgData AS

(

SELECT

Stream.[deviceId],

'thermometer' AS [SensorType],

AVG(CAST(Stream.[temperature] AS Float)) AS AvgTemperature,

System.TimeStamp AS CreatedAt

FROM [iothub] Stream TIMESTAMP BY time

WHERE GetMetadataPropertyValue(Stream, '[User].[SensorType]') = 'thermometer'

GROUP BY

Stream.[deviceId],

TumblingWindow(second, 30)

),

AlarmData AS

(

SELECT

TempAvgData.deviceId AS IoTHubDeviceID,

'TempAlert' as [AlarmType],

TempAvgData.AvgTemperature as [Reading],

Ref.[TemperatureThreshold] as [Threshold],

TempAvgData.CreatedAt as [CreatedAt]

FROM [StreamAvgData] TempAvgData

JOIN [devicerules] Ref

ON

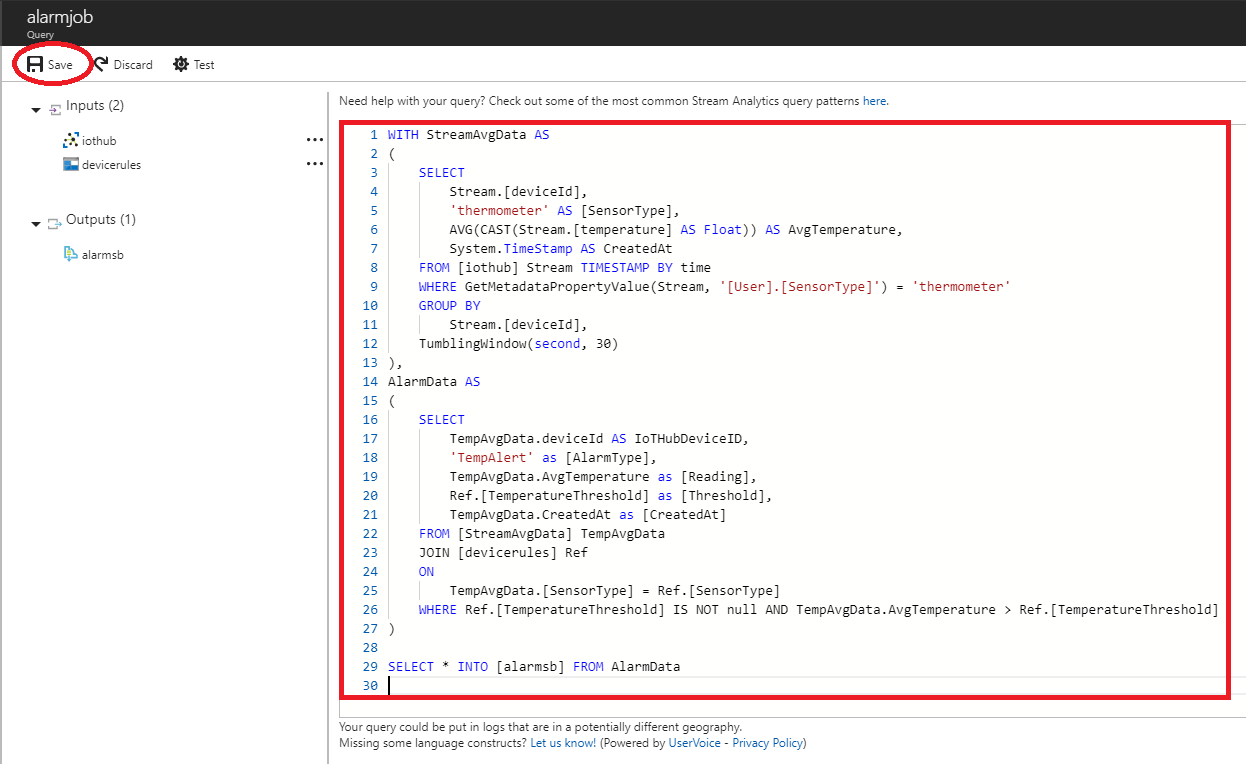
TempAvgData.[SensorType] = Ref.[SensorType]

WHERE Ref.[TemperatureThreshold] IS NOT null AND TempAvgData.AvgTemperature > Ref.[TemperatureThreshold]

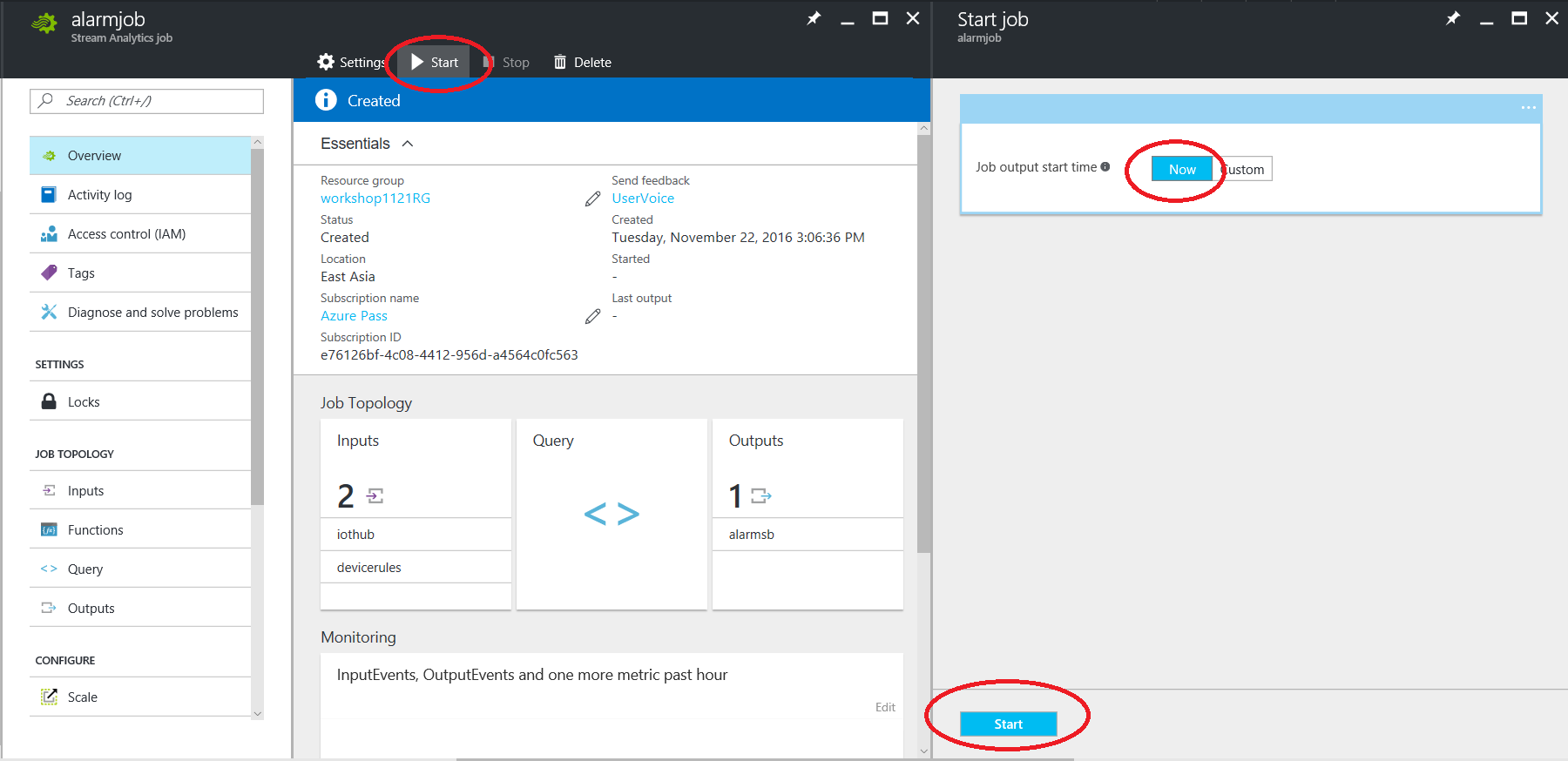
)

SELECT \* INTO [alarmsb] FROM AlarmData

* + - Save



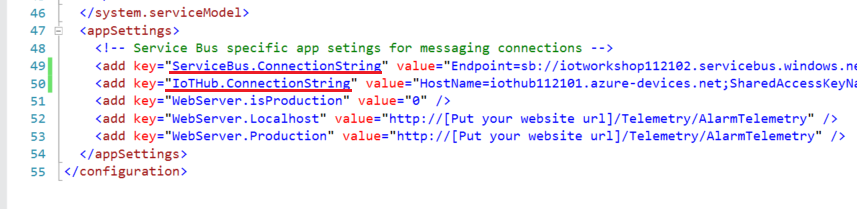
* **Start** the Stream Analytics Job
  + Start now



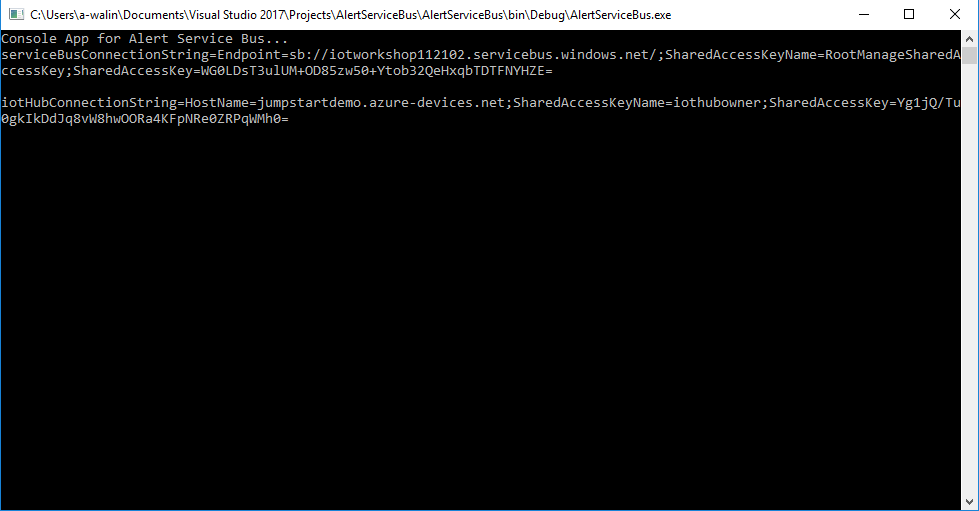
* + It may take a 1~2 minutes for starting.

## Step 5: Run the Alert Service Bus Console App

* Build the Alert Service Bus Console App (03-HOL/Sample)
  + Unzip the **AlertServiceBus.zip** file and open the solution in Visual Studio.
  + Update the **connection string** of **App.config**
    - **ServiceBus.ConnectionString**: the connection string of Service Bus
    - **IoTHub.ConnectionString**: the connection string of IoT Hub (for Cloud to Device command)

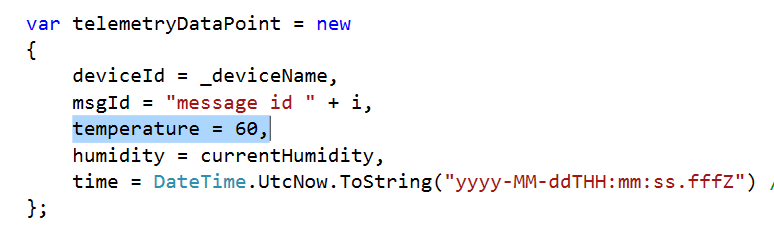


* + Press **F5** to build and debug the console App

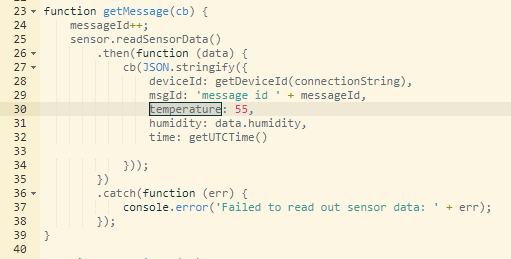


## Step 6: Run the simulated devices

* Before running these devices, you can modify the temperature so that it is greater than the threshold (The default value is 40).
  + C# Simulator

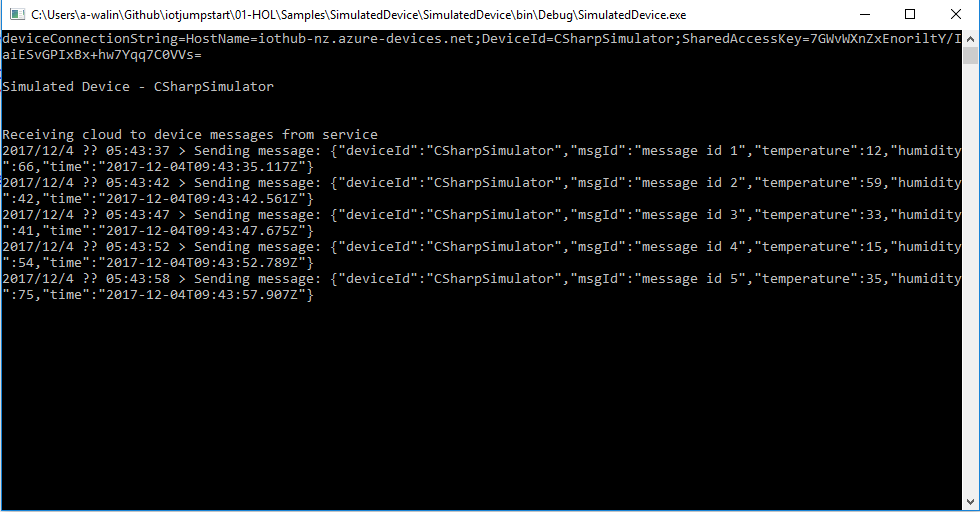


* + Raspberry Pi Azure IoT Online Simulator

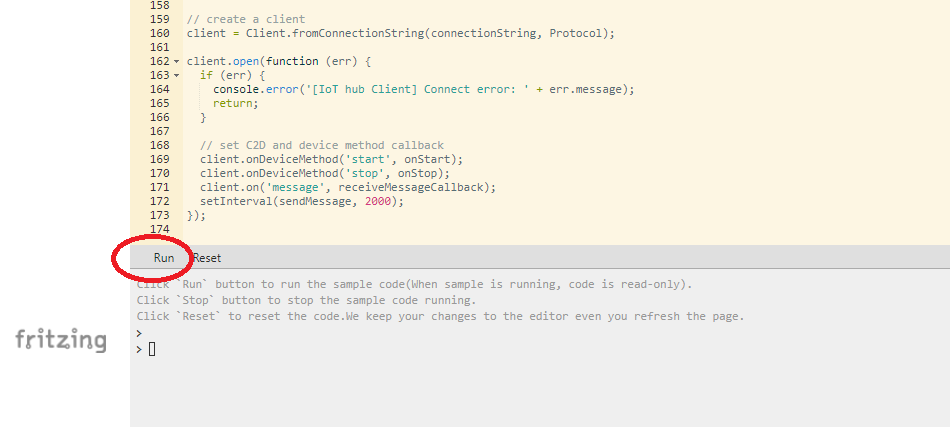


* Run the executable file (SimulatedDevice.exe) or press F5 to debug in the Visual Studio.
  + The executable file should be located at

C:\Users\<username>\Documents\Visual Studio 2017\Projects\SimulatedDevice\SimulatedDevice\bin\Debug

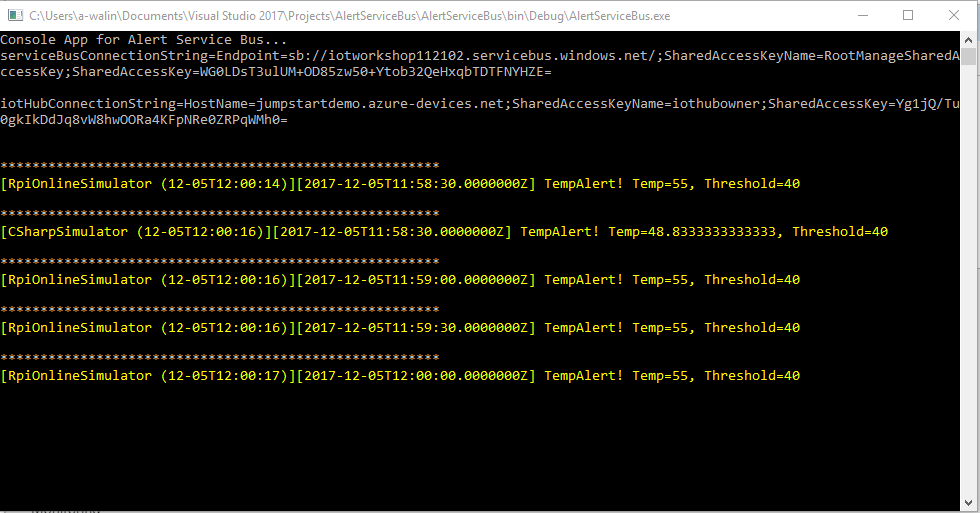


* Run your code on [Raspberry Pi Azure Online Simulator](https://azure-samples.github.io/raspberry-pi-web-simulator/).

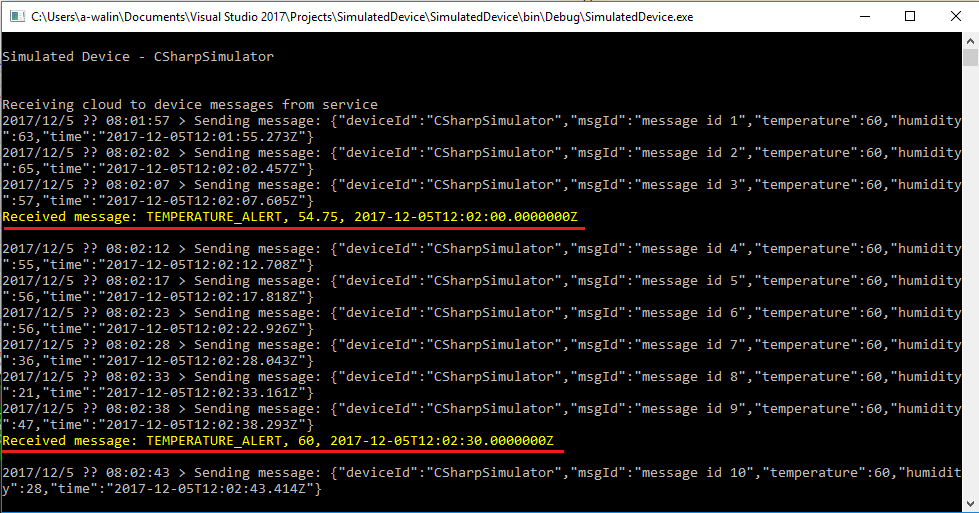


## Step 7: Observe the result of alert

* Check the console output of Service Bus. The alert should be shown when the temperature alert was triggered in every 30 seconds
  + Average temperature > Temperature threshold (The default is 40)



* Also, the devices got a Cloud-to-Device command from IoT Hub when the alert was triggered.
  + The output of C# simulator



* + The output of Raspberry Pi Azure IoT Online Simulator



* *The HOL 3 has been completed. We use another Azure Stream Analytics Job to do an alert trigger that this alert message was sent to a queue of Service Bus when the rule was detected. Therefore, we use a windows console app to receive these messages from Service Bus and send a C2D commands to the device for the further used.*
* *The next hands-on lab, we are going to enjoy the power of data visualization in Power BI.*

*Let’s quickly make a dashboard from IoT data.*