## References

* Service Bus

<https://azure.microsoft.com/zh-tw/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

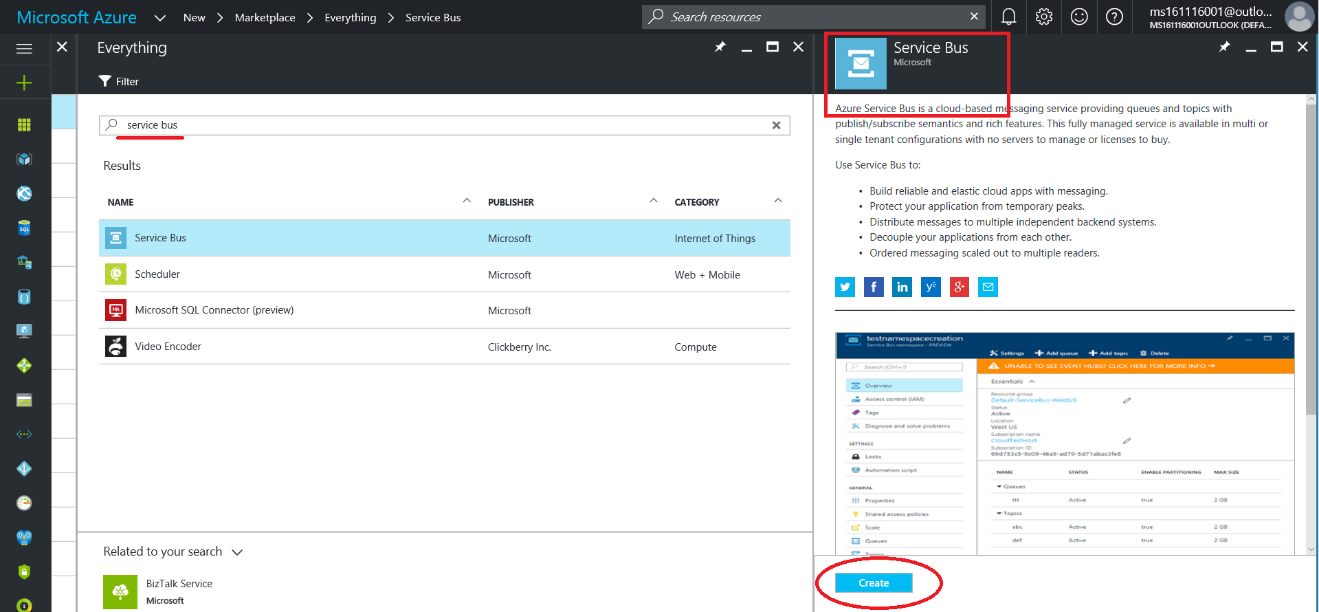
* Finished the part 4 of HOL
* Two simulated wind turbines
* NuGet packages
  + Newtonsoft.Json for JSON in C#
  + WindowsAzure.ServiceBus for Service Bus
  + Microsoft.Azure.Devices (Service SDK for Azure IoT Devices) for Cloud to Device message of IoT Hub

## Goals

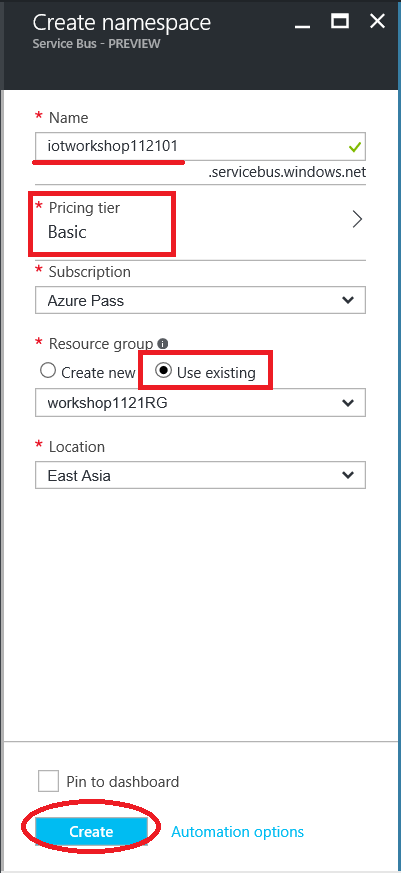
* Create a **Service Bus**
  + Receive the alarm message from a Stream Analytics Job.
  + Send Cloud to Device command to the corresponding devices.
* Create an **Azure Stream Analytics Job**
  + Processing the telemetry data with device rules.
  + Trigger an alarm message to **Service Bus** when the alarm was detected.

## Step 1: Create a Service Bus for Alarm Message

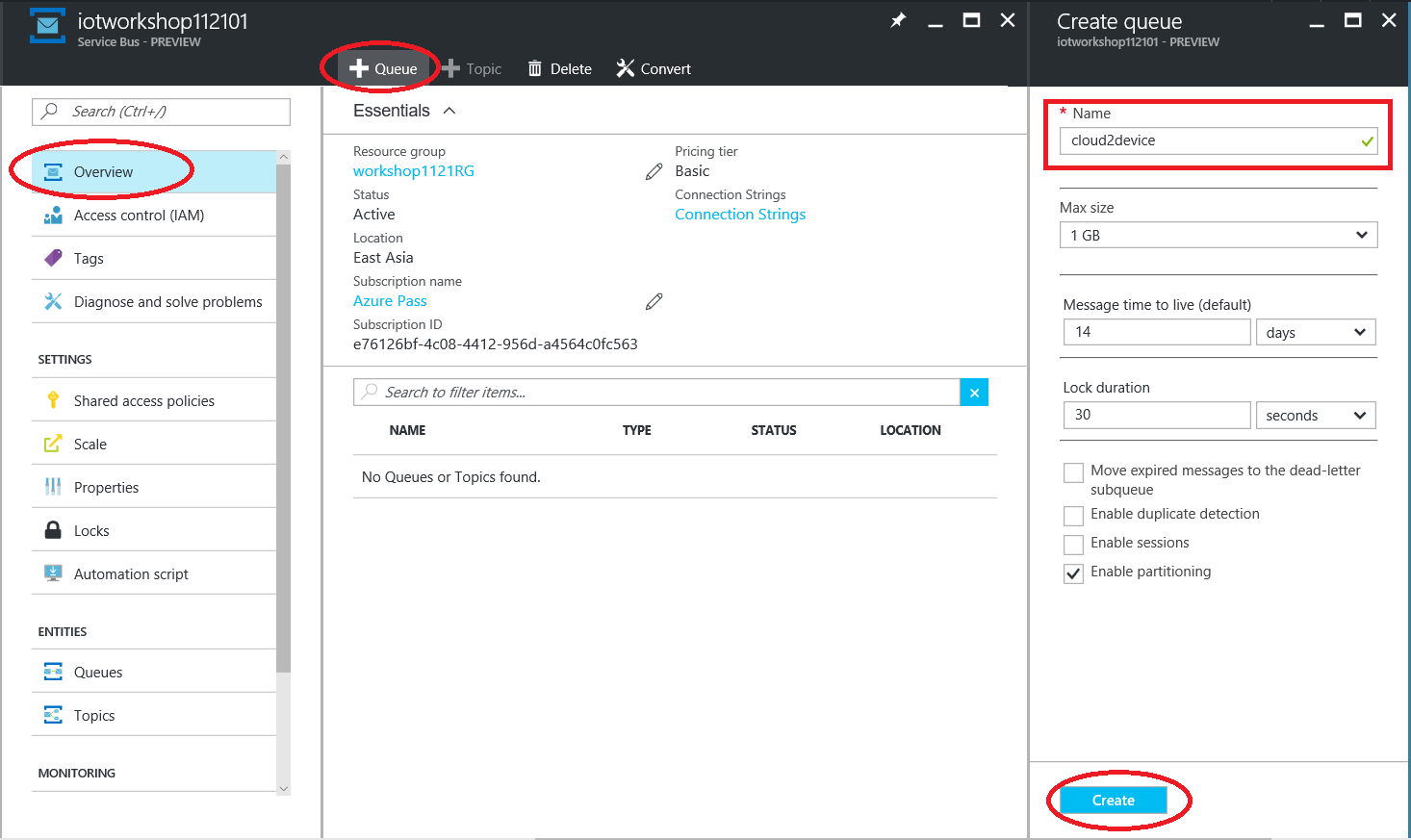
* Search and **Create** the **Service Bus**



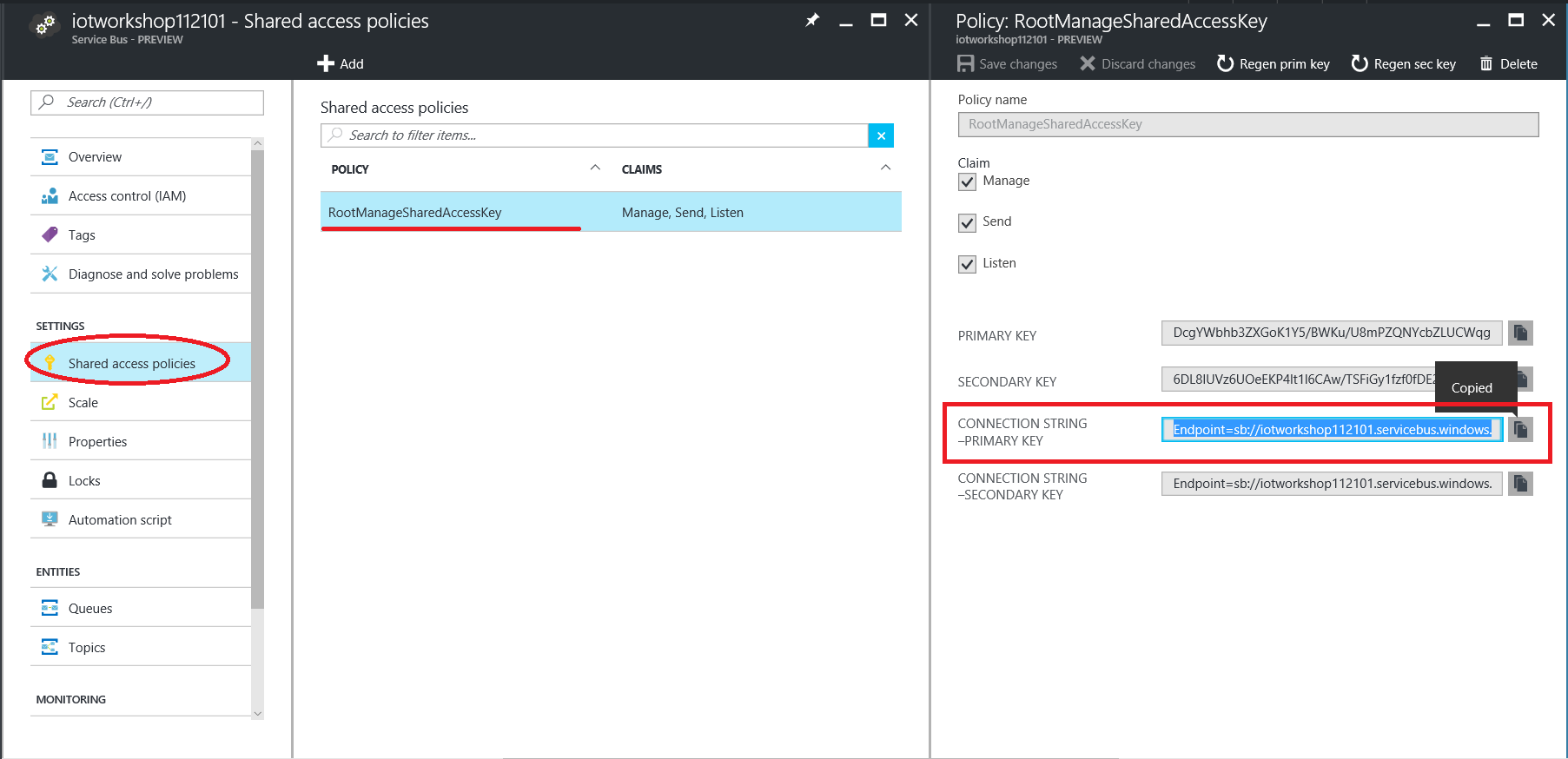
* Create a namespace
  + Name: **iotworkshop112101** (for example)
  + Pricing tier: **Basic**
  + Use the existing **Resource Group**.



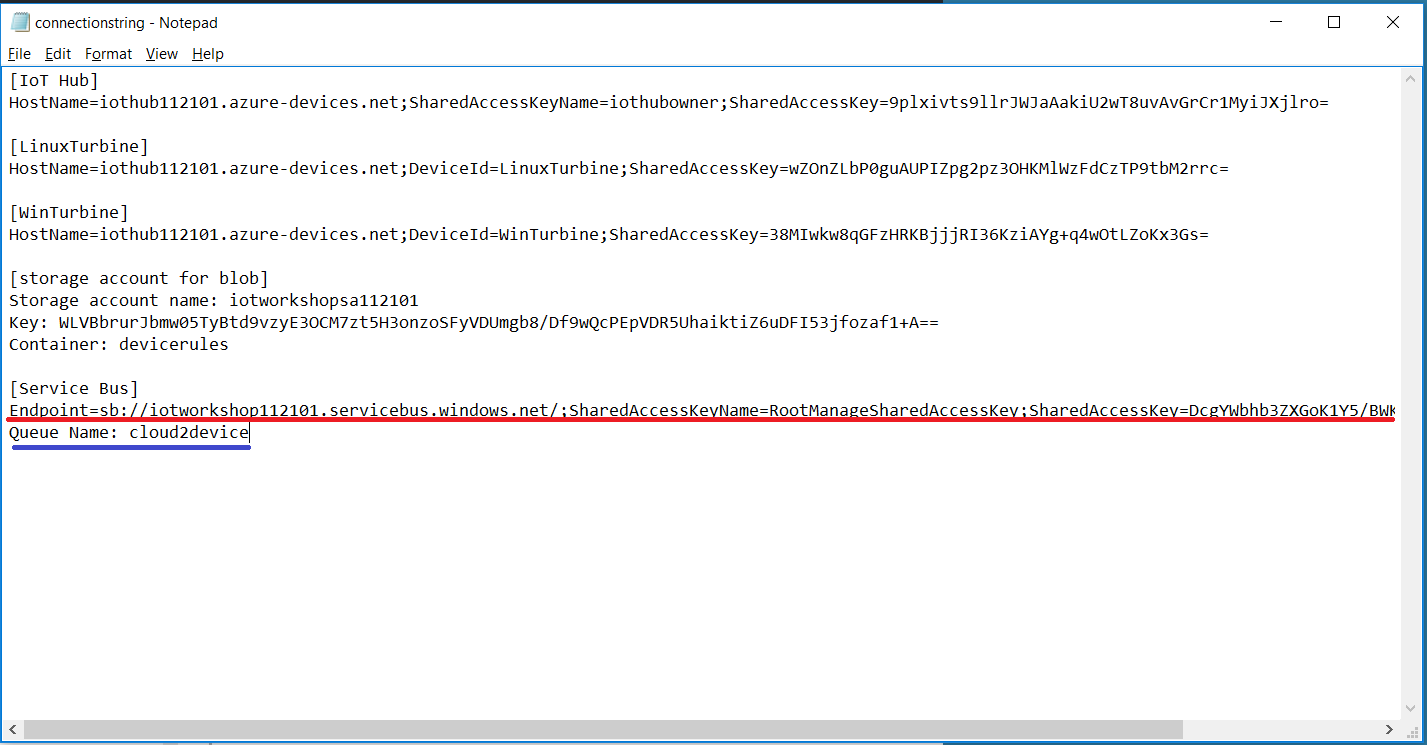
* Create a **Queue**
  + Name: **cloud2device** (must be fixed)



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

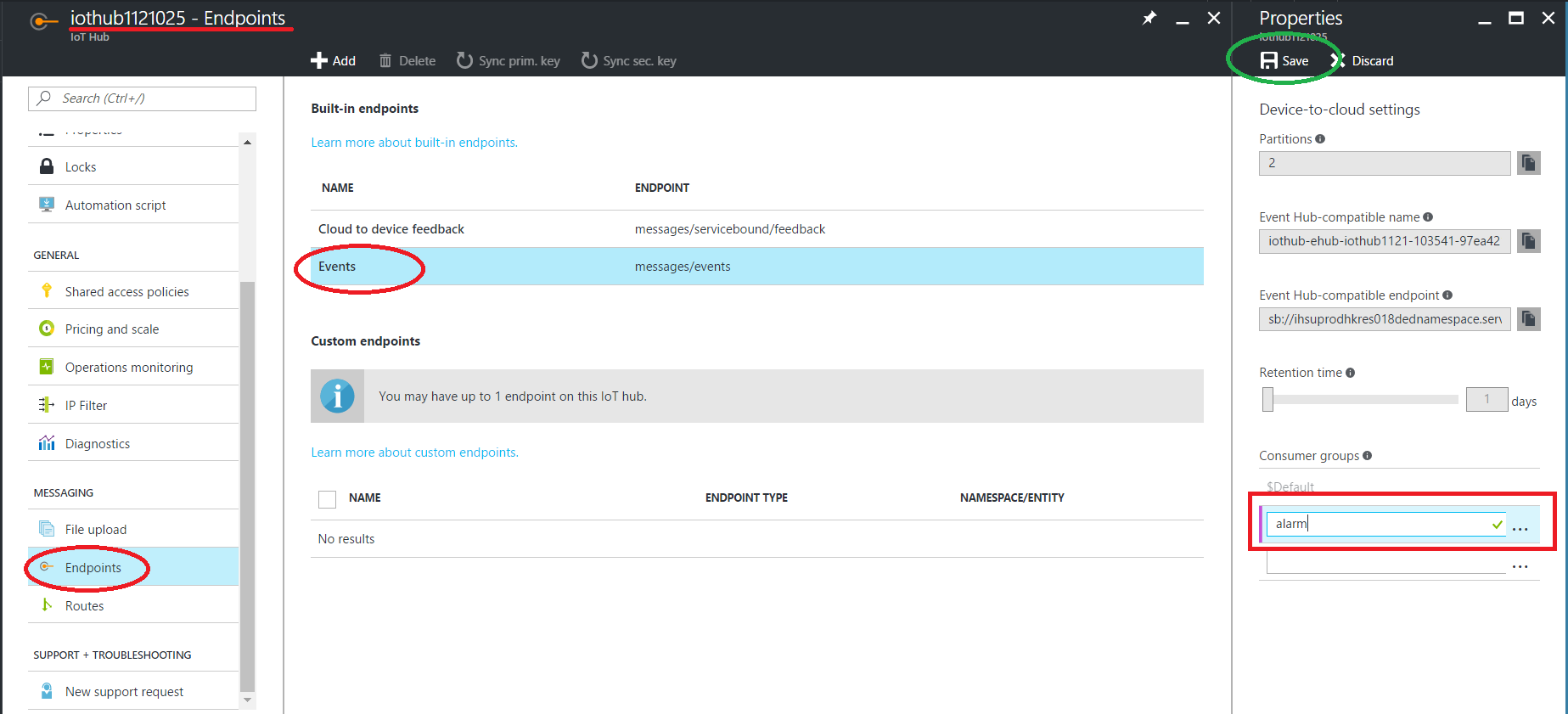


* Save the **connection string of Service Bus** for the later used.

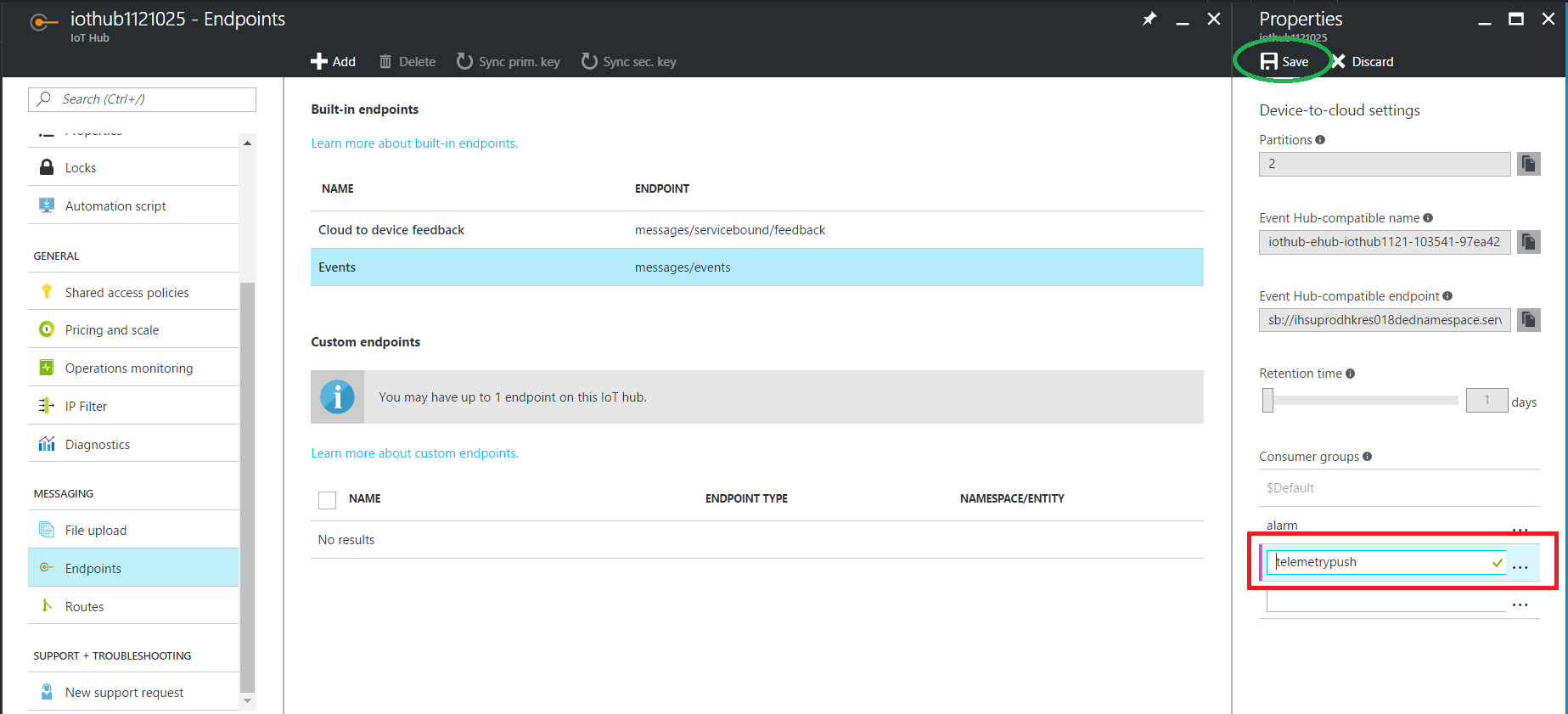


## Step 2: Create new consumer groups of IoT Hub

* 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)



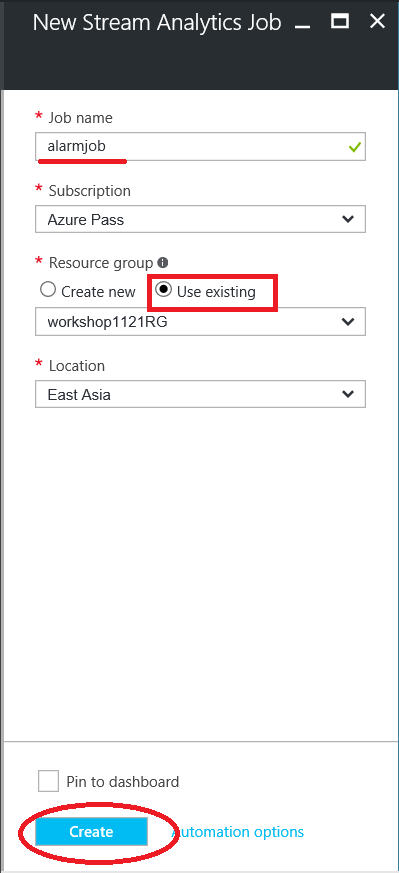
* Add another consumer group for the telemetry push by Event Processor Host, and it will be used in the next HOL.
  + Name: **telemetrypush** (must be fixed)



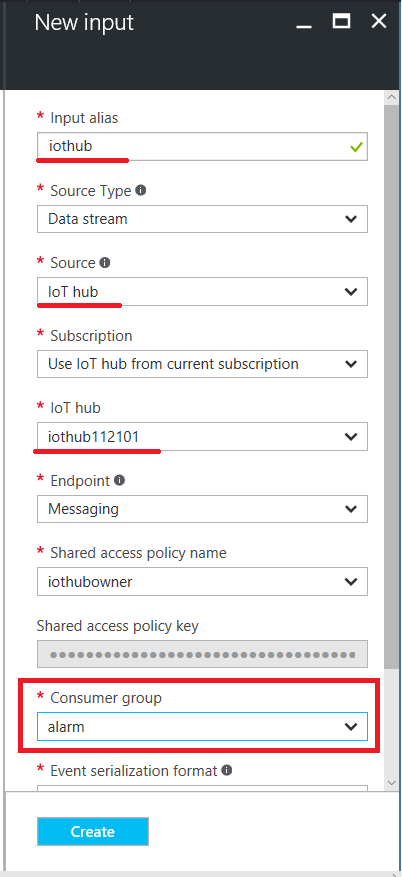
* + Don’t forget to **Save** the changed properties.

## Step 3: Create a Stream Analytics Job for Alarm Message

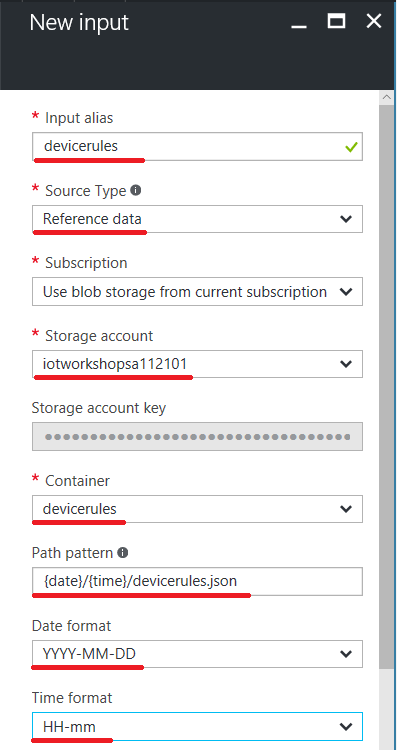
* New a **Stream Analytics Job**
  + Job Name: **alarmjob**

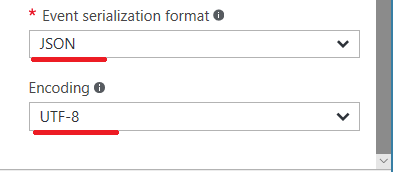


* + Add an **IoT Hub** as the **input**
    - Input alias: **iothub** (must be fixed)
    - Source Type: **Data stream**
    - Source: **IoT Hub**
    - IoT Hub: select the used IoT Hub
    - Consumer group: **alarm (not $Default)**
    - The others should be set as below

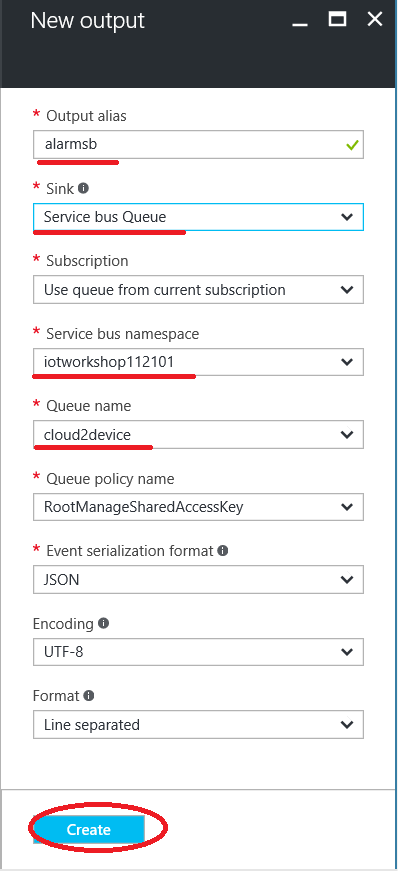


* + It’s same as the HOL-4, also add a rule **Blob** as another one of **inputs**.
    - Input alias: **devicerules** (must be fixed)
    - Source Type: **Reference data**
    - Storage account: select the storage account as you provisioned
    - Container: **devicerules** (must be fixed)
    - Path 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.





* Press the **Create** button.
  + Add a Service Bus as the output
    - Output alias: **alarmsb** (must be fixed)
    - Sink: **Service bus Queue**
    - Service bus namespace: select what you created
    - Queue name: **cloud2device**



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

WITH StreamAvgData AS

(

SELECT

deviceId,

AVG(CAST(speed AS BIGINT)) AS AvgSpeed,

AVG(CAST(depreciation AS float)) AS AvgDepreciation,

System.TimeStamp AS LocalTime,

System.TimeStamp AS CreatedAt

FROM [iothub] TIMESTAMP BY time

GROUP BY

deviceId,

TumblingWindow(second, 30)

),

AlarmData AS

(

SELECT

Stream.deviceId AS IoTHubDeviceID,

'CutOutSpeed MessageID' AS [MessageID],

'CutOutSpeed' as [AlarmType],

Stream.AvgSpeed as [Reading],

Ref.[CutOutSpeed] as [Threshold],

Stream.LocalTime,

Stream.CreatedAt

FROM [StreamAvgData] Stream

JOIN [devicerules] Ref

ON

Stream.deviceId = Ref.[DeviceID]

WHERE Ref.[CutOutSpeed] IS NOT null AND Stream.AvgSpeed > Ref.[CutOutSpeed]

UNION ALL

SELECT

Stream2.deviceId AS IoTHubDeviceID,

'Repair MessageID' AS [MessageID],

'Repair' as [AlarmType],

Stream2.AvgDepreciation as [Reading],

Ref2.[Repair] as [Threshold],

Stream2.LocalTime,

Stream2.CreatedAt

FROM [StreamAvgData] Stream2

JOIN [devicerules] Ref2

ON

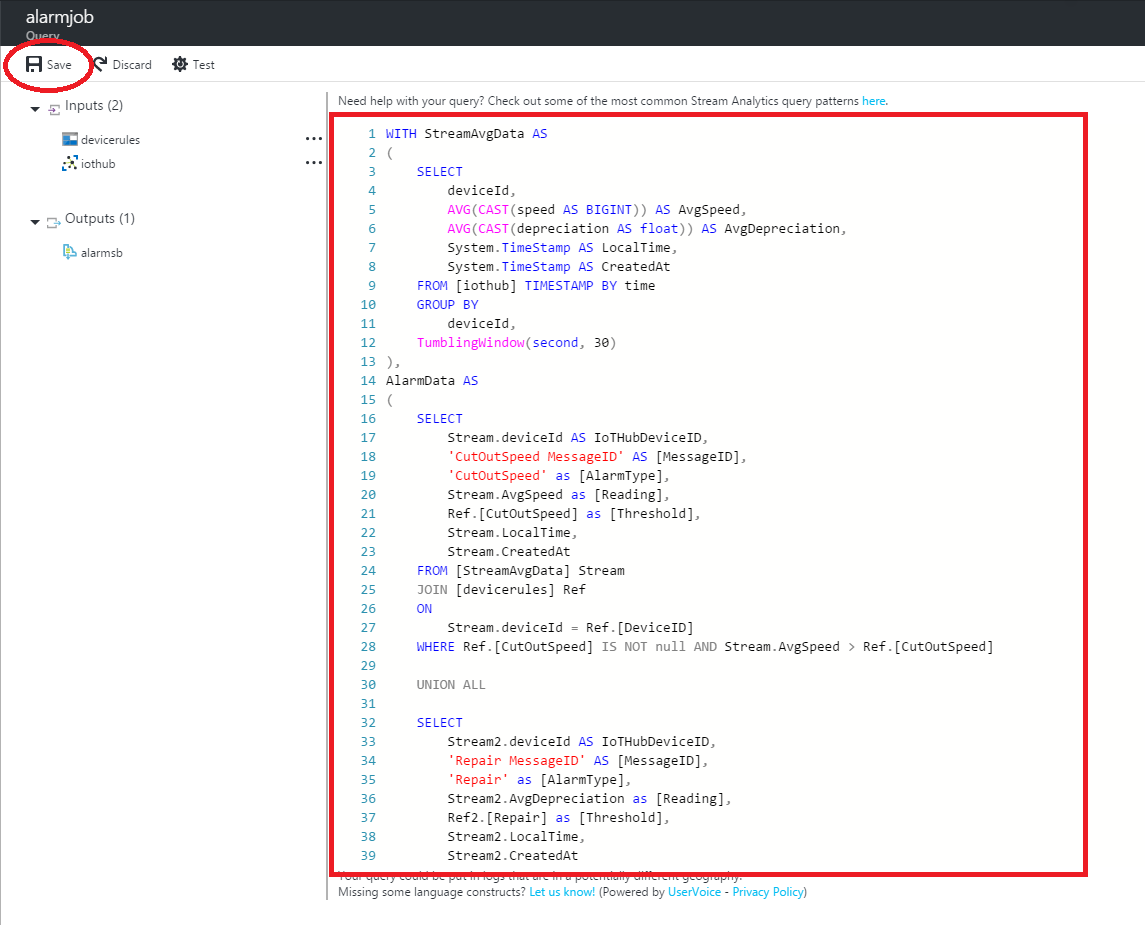
Stream2.deviceId = Ref2.[DeviceID]

WHERE Ref2.[Repair] IS NOT null AND Stream2.AvgDepreciation < Ref2.[Repair]

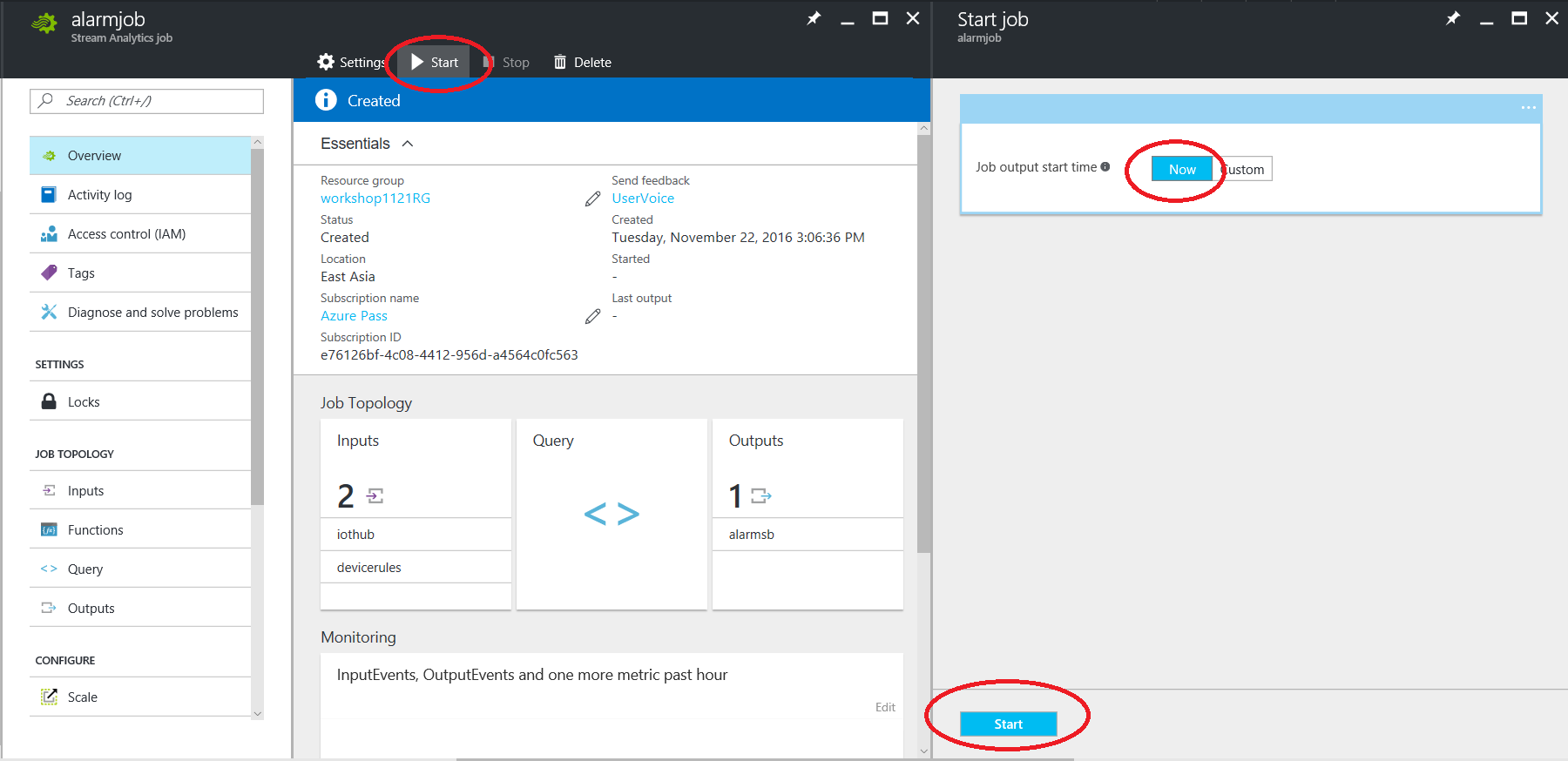
)

SELECT \* INTO [alarmsb] FROM AlarmData

* + - Save



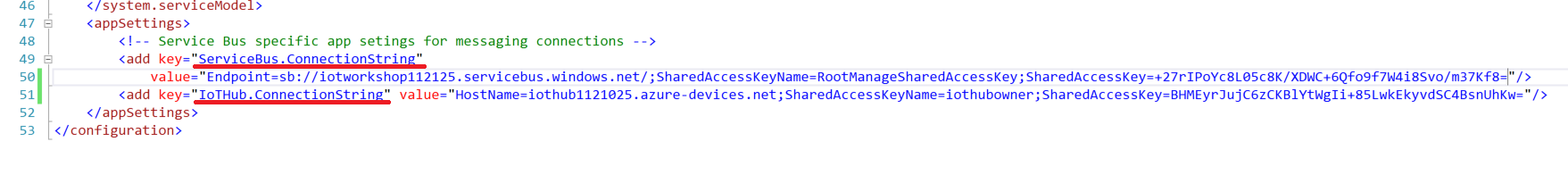
* **Start** the Stream Analytics Job
  + Start now



* + Wait for the deployment

## Step 4: Run the Alert Service Bus Console App

* Build the Alert Service Bus Console App (05-HOL/Sample)
  + Unzip the **AlertServiceBus.zip** file and open the solution in VS
  + 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 5: Run the simulated Linux and Windows wind turbines

* LinuxTurbine - Run C Simple Sample of AMQP in SDK

cd ~/azure-iot-sdk-c/cmake/iotsdk\_linux/serializer/samples/simplesample\_amqp/

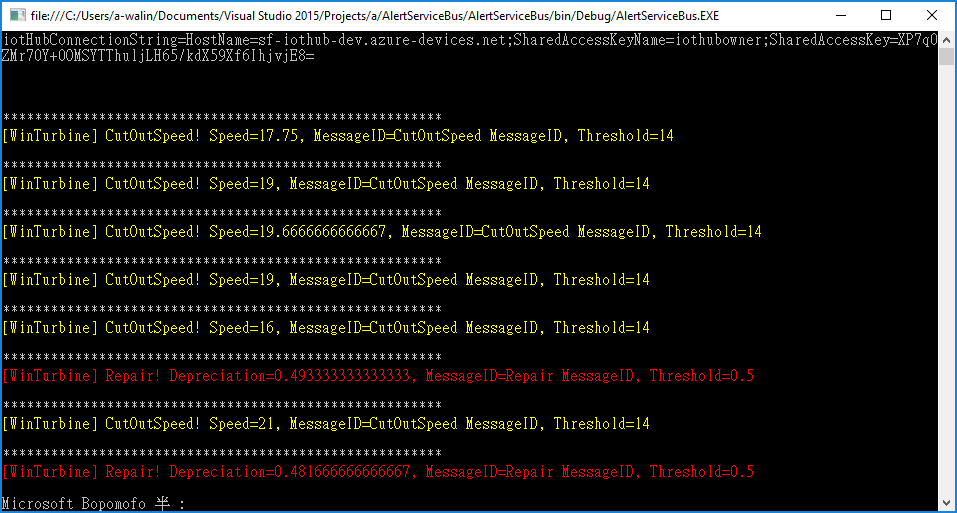
./simplesample\_amqp

* WinTurbine - Run the executable file (SimulatedWindTurbine.exe) or debugging in the VS
  + The executable file should be located at

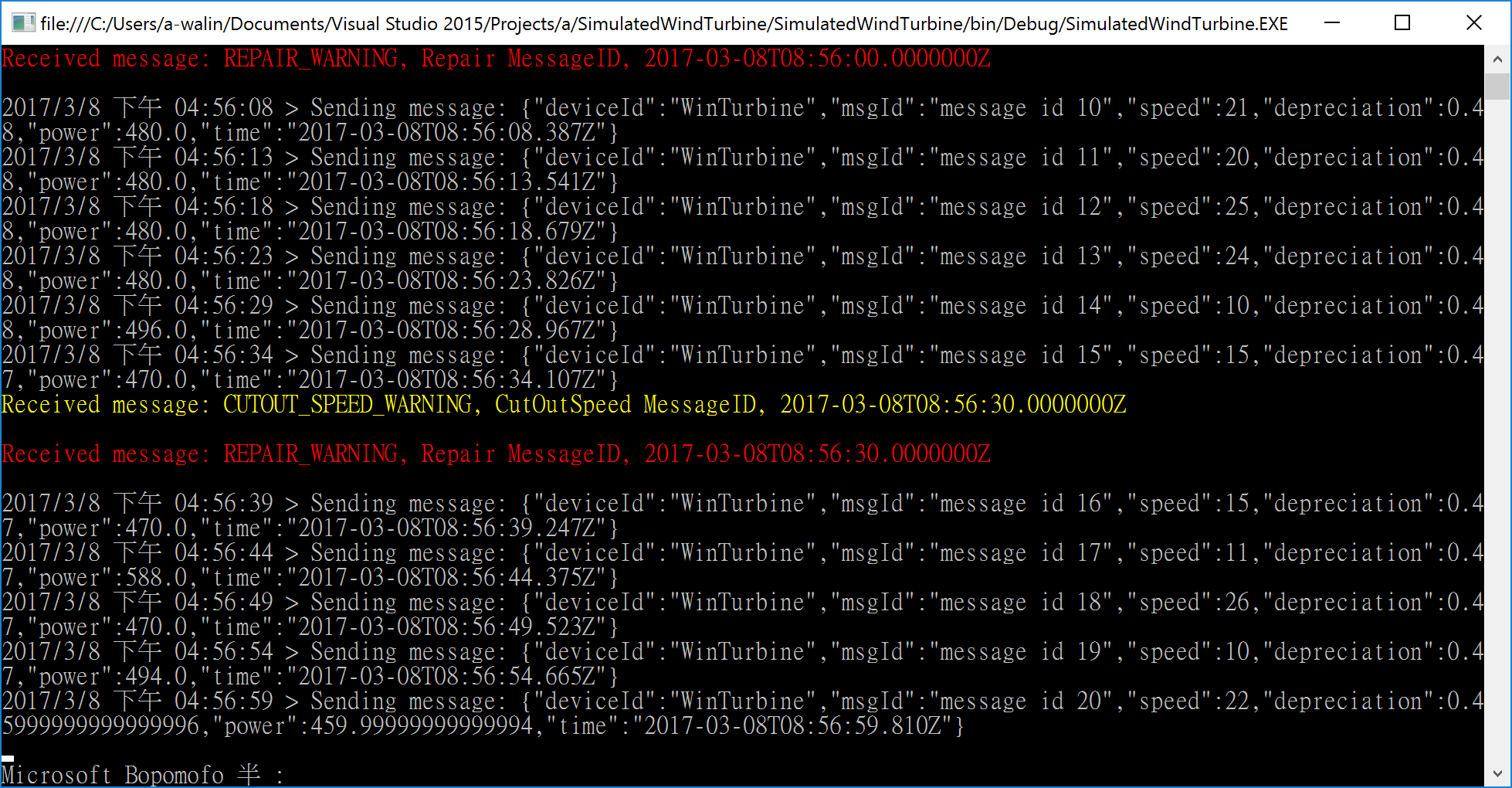
C:\Users\user\Documents\Visual Studio 2015\Projects\SimulatedWindTurbine\SimulatedWindTurbine\bin\Debug

## Step 7: Observe the results

* Check the console output of Service Bus. The alert should be shown when these conditions are triggered within 30 seconds:
  + Average wind speed > Cut-out Speed threshold (The default is 14)
  + Average depreciation rate < Repair threshold (The default is 0.5)



* Also, the device gets the Cloud-to-Device commands from IoT Hub when the alert is triggered.
  + The output of device



* *The HOL 5 has been completed. We use another Azure Stream Analytics Job to do that to trigger on an alarm message when the device rule is detected in a real time. the alarm message will be sent to a queue of Service Bus for messaging queue. Then, we use a windows console app to receive these alarm messages from the queue and send C2D commands to the devices for the further used.*
* *The next hands-on lab, we are going to enjoy the power of data presentation in Power BI.*

*Let’s try to find out the insights from data.*