Weather Station POV documentation

References used:

https://microsoft.github.io/azure-iot-developer-kit/

https://github.com/AzureArchitecture

https://www.c-sharpcorner.com/topics/azure-function

https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-your-first-function-visual-studio

https://microsoft.github.io/azure-iot-developer-kit/docs/get-started/

https://www.10thmagnitude.com/step-step-guide-creating-functions-within-azures-iot-hub/

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-arduino-iot-devkit-az3166-get-started

https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-with-azure-functions

https://microsoft.github.io/azure-iot-developer-kit/versions/

https://github.com/microsoft/vscode-iot-workbench/blob/master/docs/iot-devkit/devkit-get-started.md

https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-troubleshoot-input

https://azure.microsoft.com/en-us/resources/samples/azureiotlabs/

https://cosmosdb.github.io/labs/

https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-documentdb-output

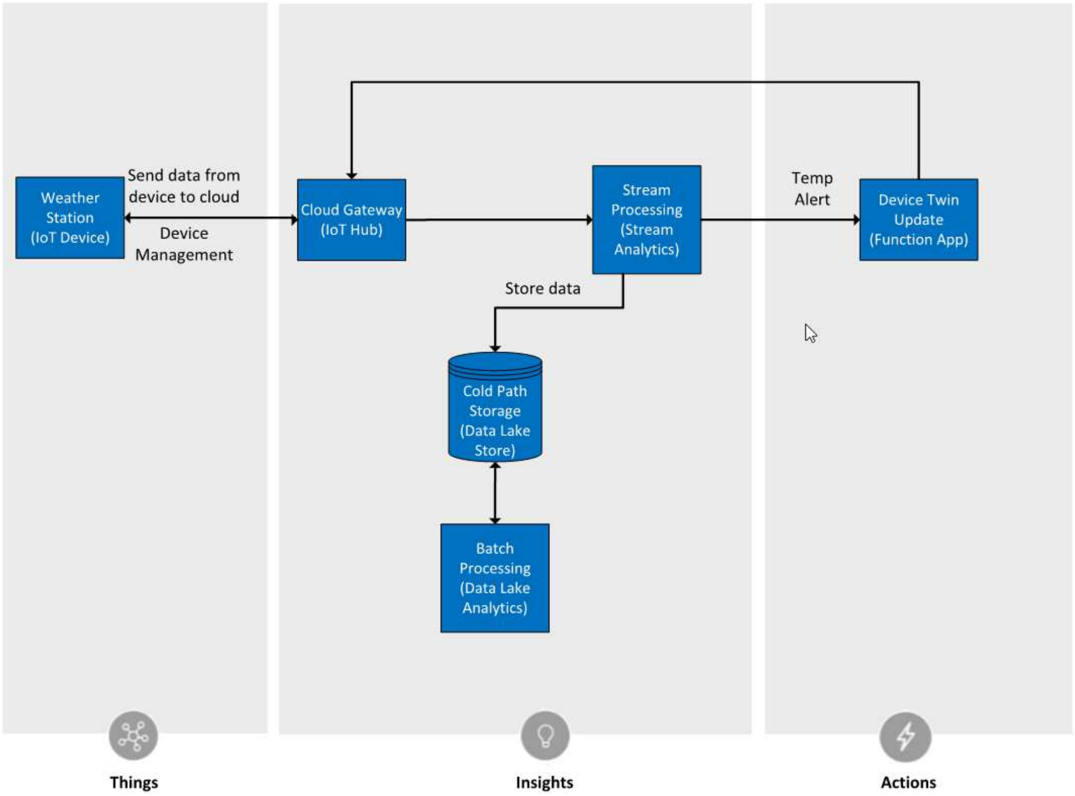
http://azurefabric.com/cosmosdb-some-setup-and-quick-get-started-tips/

https://darenmay.com/

https://www.axonize.com/blog/iot-technology/the-advantages-and-disadvantages-of-using-azure-stream-analytics-for-iot-applications/

https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-function-vs-code

1. Create an Architecture Diagram for a solution that satisfies the requirements (use PowerPoint or Visio)


Weather Station (IoT Device)
Send data from device to cloud
Device Management
Cloud Gateway (IoT Hub)
Stream Processing (Stream Analytics)
Temp Alert
Device Twin Update (Function App)
Store data
Cold Path Storage (Data Lake Store)
Batch Processing (Data Lake Analytics)
Things
Insights
Actions

2. Create a threat model and note any key factors that drive configuration of the solution

Reference:

Course: DEV301x IoT Architecture Design and Business Planning

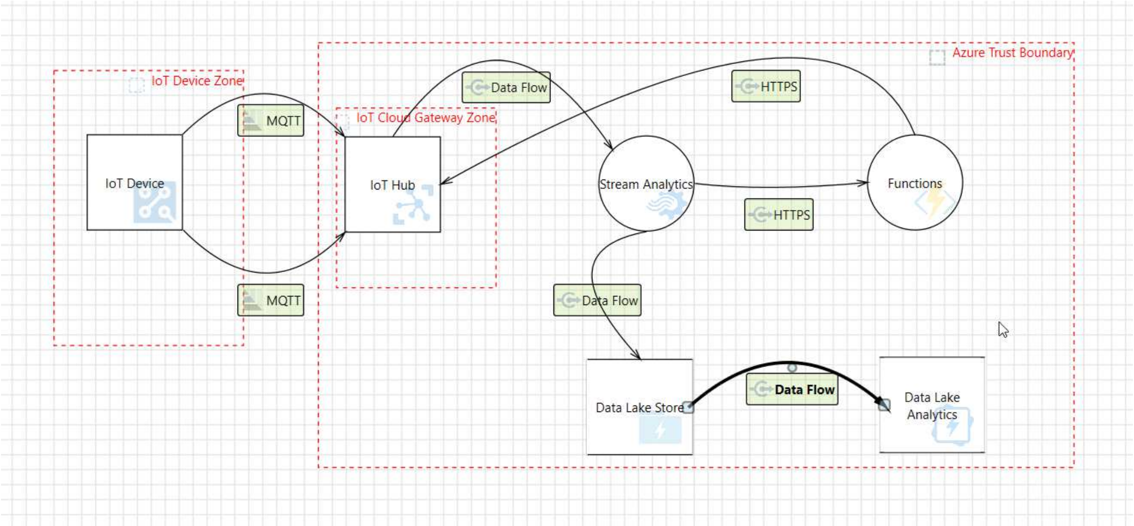
Module: Understanding the Azure IoT Reference Architecture

Lab: Reference Architecture SubSystems and Security

Topics: Threat Modeling the Azure IoT Reference Architecture and Microsoft Threat Modeling Tool 2016 Review

Threat Modeling Tool 2016 Getting Started Guide.docx

2a. Install 2016 version and link the Azure template v3.



Design View

Plan of Attack

a) Create resources based on design from left to right

b) Configure MX Chip but leave the flashing device.ino file later

c) Create IOT Hub

d) Create Azure Stream Analytics

e) Create Azure Functions

f) Test output with stream analytics

g) Create Azure Data Lake Store

h) Test output with stream analytics

i) Create Azure Data Lake Analytics

3. Preparing the Device and Connecting to Azure

Course: DEV325x Introduction to Device Programming for IoT: C Edition

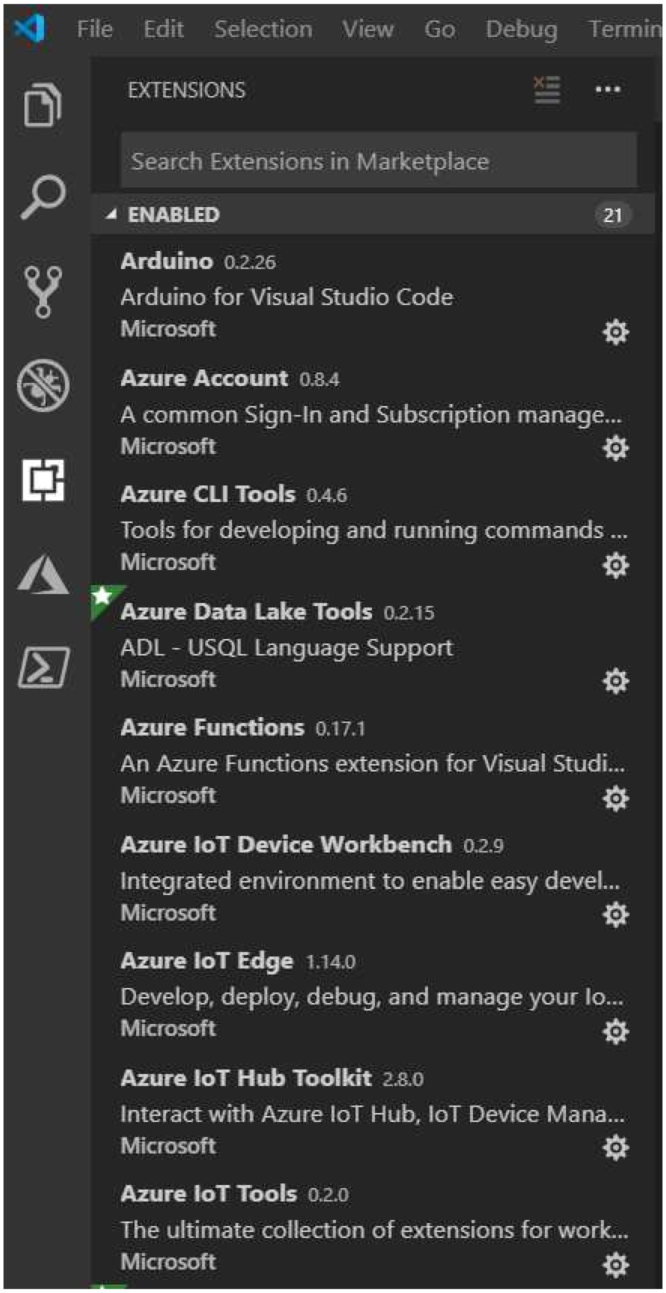
Module: Data and Device Inputs

Lab: Configure the MXChip Development Environment

4. Setup Your MXChip Device

5. Configure Your Environment

6. Ensure have proper VS Studio extensions installed:



Arduino 0.2.26
Arduino for Visual Studio Code
Microsoft

Azure Account 0.8.4
A common Sign-In and Subscription manage...
Microsoft

Azure CLI Tools 0.4.6
Tools for developing and running commands ...
Microsoft

Azure Data Lake Tools 0.2.15
ADL - USQL Language Support
Microsoft

Azure Functions 0.17.1
An Azure Functions extension for Visual Studi...
Microsoft

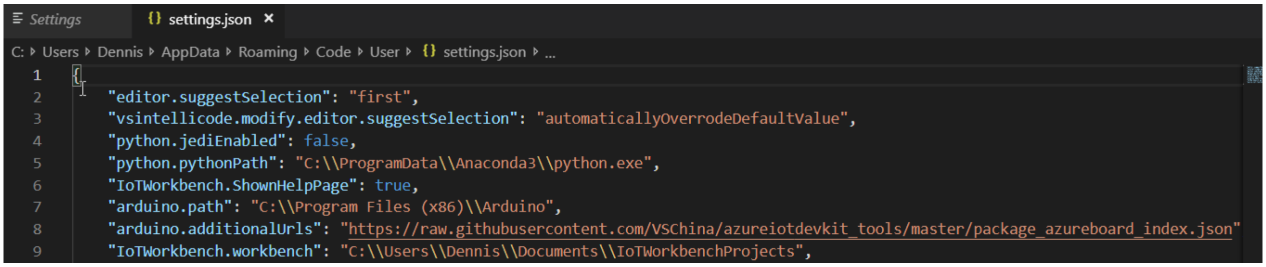
Azure IoT Device Workbench 0.2.9
Integrated environment to enable easy devel...
Microsoft

Azure IoT Edge 1.14.0
Develop, deploy, debug, and manage your Io...
Microsoft

Azure IoT Hub Toolkit 2.8.0
Interact with Azure IoT Hub, IoT Device Mana...
Microsoft

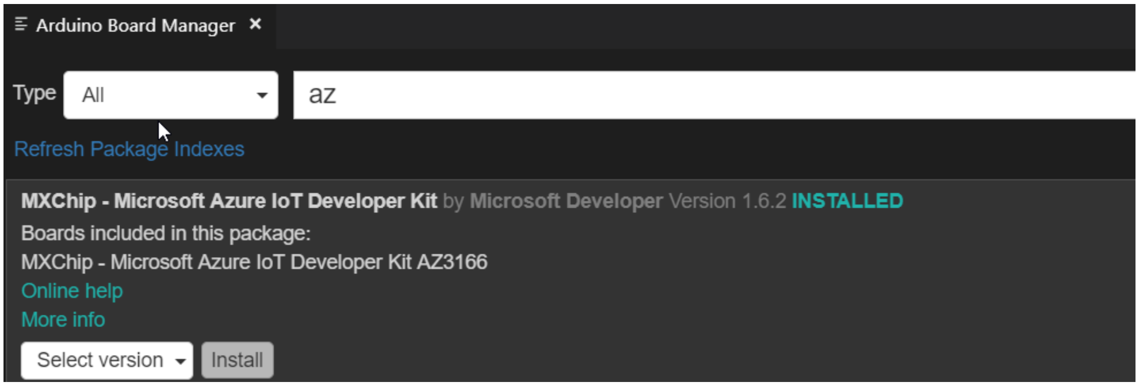
Azure IoT Tools 0.2.0
The ultimate collection of extensions for work...
Microsoft

7. The following JSON to your settings file.


Settings
settings.json
C: › Users › Dennis › AppData › Roaming › Code › User › settings.json › ...

```
{
    "editor.suggestSelection": "first",
    "vsintellicode.modify.editor.suggestSelection": "automaticallyOverrodeDefaultValue",
    "python.jediEnabled": false,
    "python.pythonPath": "C:\\ProgramData\\Anaconda3\\python.exe",
    "IoTWorkbench.ShownHelpPage": true,
    "arduino.path": "C:\\Program Files (x86)\\Arduino",
    "arduino.additionalUrls": "https://raw.githubusercontent.com/VSChina/azureiotdevkit_tools/master/package_azureboard_index.json"
    "IoTWorkbench.workbench": "C:\\Users\\Dennis\\Documents\\IoTWorkbenchProjects",
```

8. Arduino: Board Manager – **Make sure Board version must match the MXChip firmware**



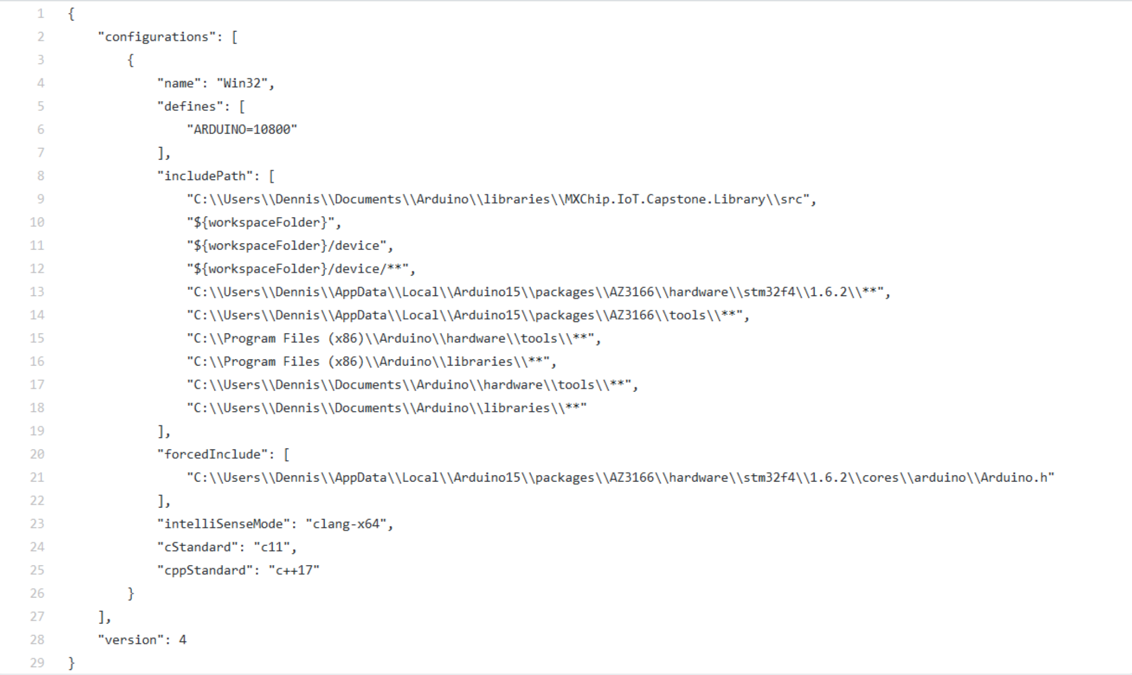
Type: All

az

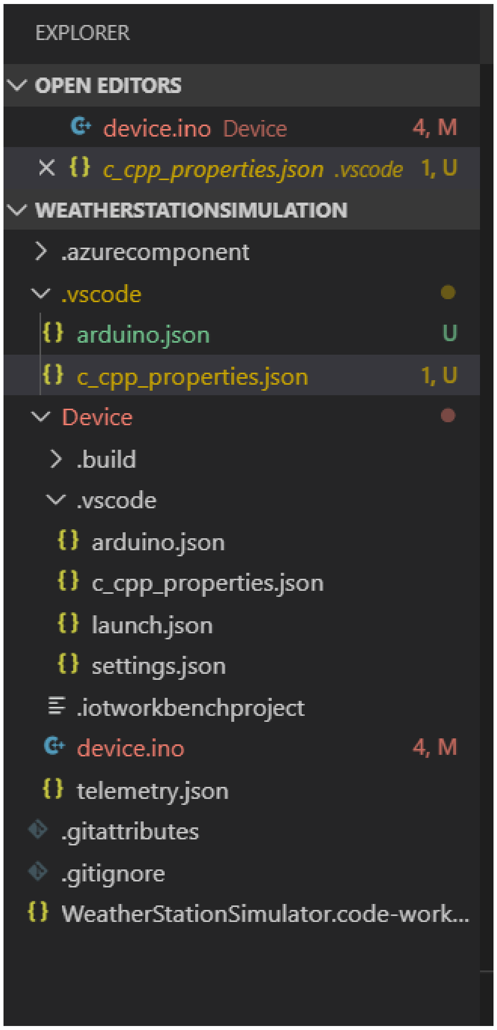
MXChip - Microsoft Azure IoT Developer Kit by Microsoft Developer Version 1.6.2 INSTALLED
Boards included in this package:
MXChip - Microsoft Azure IoT Developer Kit AZ3166

9. The USB interface used to communicate with your MXChip is **ST-Link.**

Sample Code for cppproperties.json (There are 2 c_cpp_properties.json)



```
{
    "configurations": [
        {
            "name": "Win32",
            "defines": [
                "ARDUINO=10800"
            ],
            "includePath": [
                "C:\\Users\\Dennis\\Documents\\Arduino\\libraries\\MXChip.IoT.Capstone.Library\\src",
                "${workspaceFolder}",
                "${workspaceFolder}/device",
                "${workspaceFolder}/device/**",
                "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\hardware\\stm32f4\\1.6.2\\**",
                "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\tools\\**",
                "C:\\Program Files (x86)\\Arduino\\hardware\\tools\\**",
                "C:\\Program Files (x86)\\Arduino\\libraries\\**",
                "C:\\Users\\Dennis\\Documents\\Arduino\\hardware\\tools\\**",
                "C:\\Users\\Dennis\\Documents\\Arduino\\libraries\\**"
            ],
            "forcedInclude": [
                "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\hardware\\stm32f4\\1.6.2\\cores\\arduino\\Arduino.h"
            ],
            "intelliSenseMode": "clang-x64",
            "cStandard": "c11",
            "cppStandard": "c++17"
        }
    ],
    "version": 4
}
```



```
EXPLORER
∨ OPEN EDITORS
      device.ino  Device                          4, M
  ×  {} c_cpp_properties.json  .vscode            1, U
∨ WEATHERSTATIONSIMULATION
  > .azurecomponent
  ∨ .vscode                                          ●
    {} arduino.json                                 U
    {} c_cpp_properties.json                       1, U
  ∨ Device                                           ●
    > .build
    ∨ .vscode
      {} arduino.json
      {} c_cpp_properties.json
      {} launch.json
      {} settings.json
    .iotworkbenchproject
    device.ino                                     4, M
    {} telemetry.json
  .gitattributes
  .gitignore
  {} WeatherStationSimulator.code-work...
```

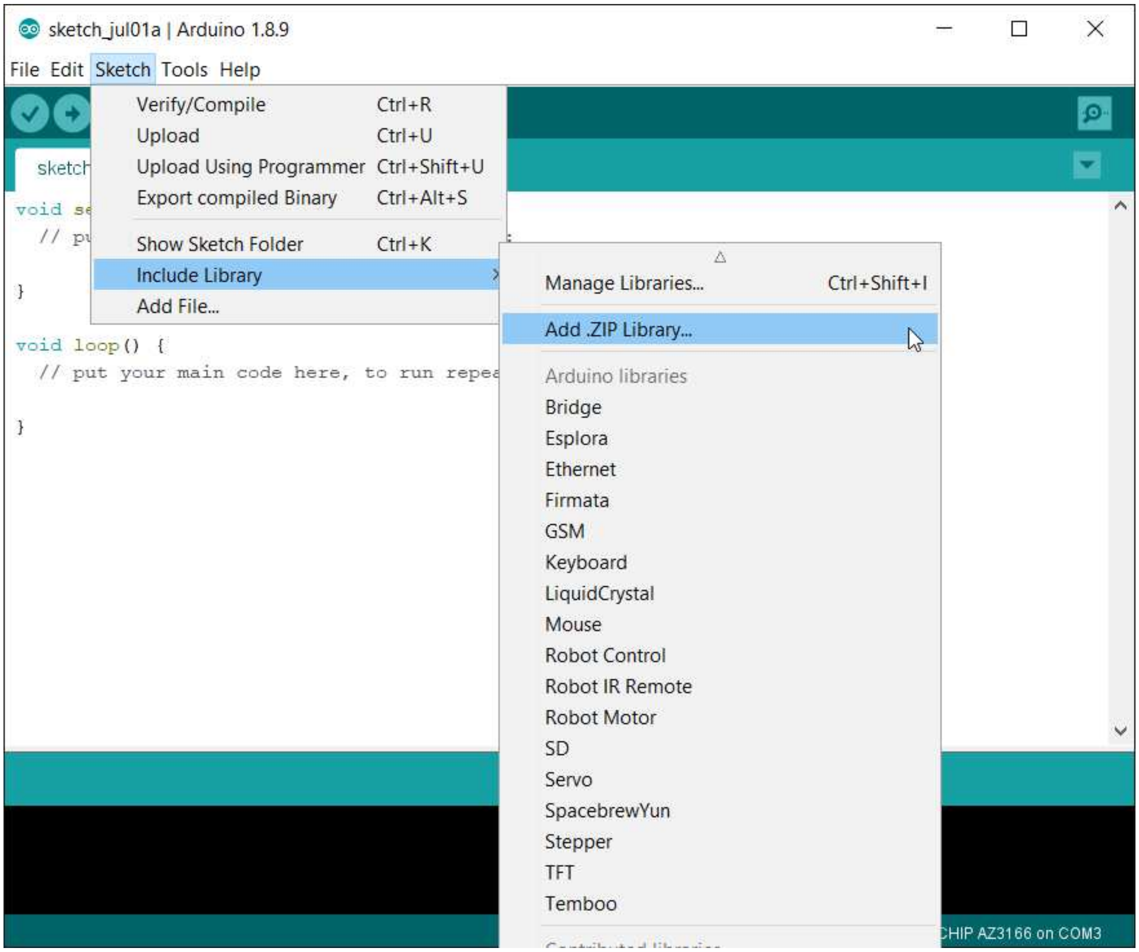
10. Reference (NOT USED YET):

Course: DEV297x IoT Device Configuration and Communication: C Edition

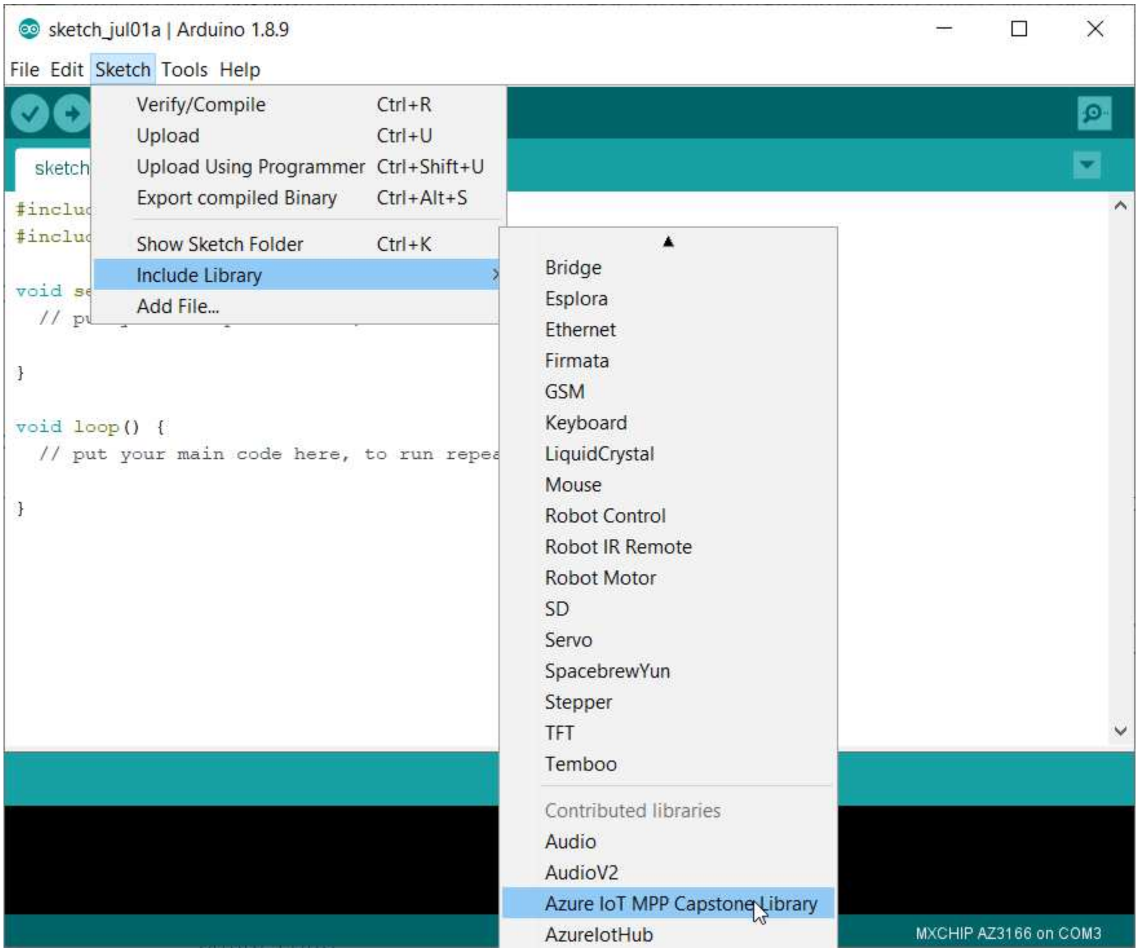
Module: Manage Your Devices

Lab: Automating Device Configuration and Management

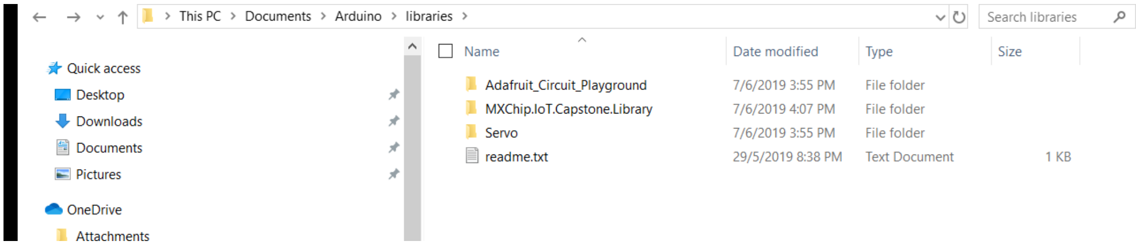
11. Import the MXChip.IoT.Capstone.Library into the Arduino development environment



```
void se
  // pu

}

void loop() {
  // put your main code here, to run repea

}
```



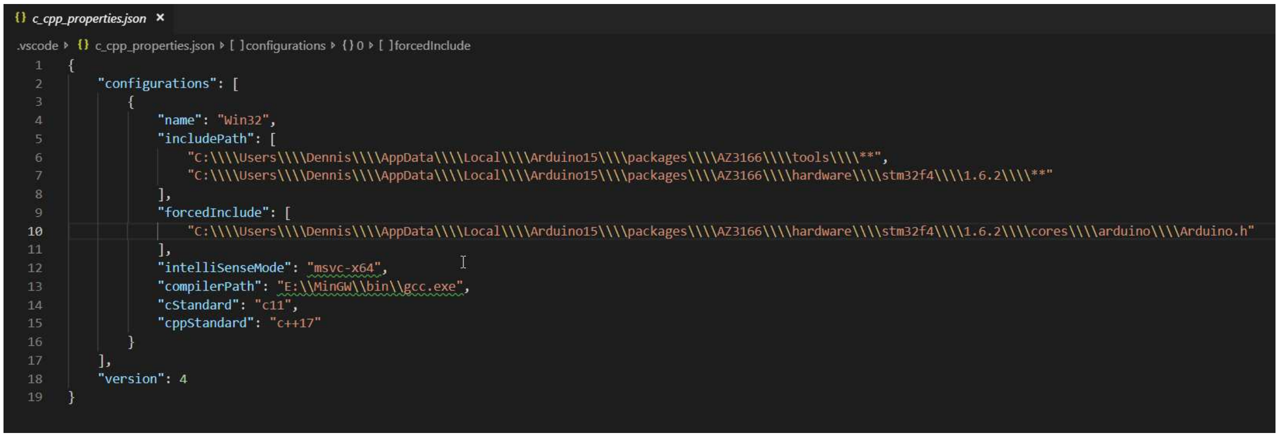
```
#incluc
#incluc

void se
  // pu

}

void loop() {
  // put your main code here, to run repea

}
```



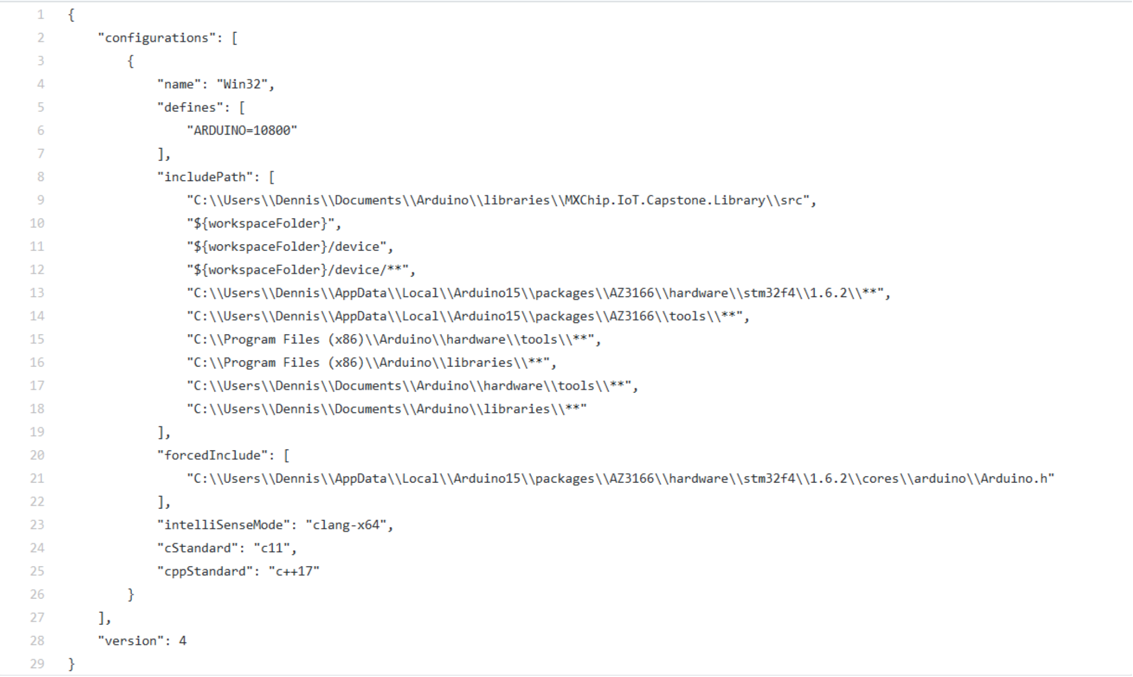
Name	Date modified	Type	Size
Adafruit_Circuit_Playground	7/6/2019 3:55 PM	File folder	
MXChip.IoT.Capstone.Library	7/6/2019 4:07 PM	File folder	
Servo	7/6/2019 3:55 PM	File folder	
readme.txt	29/5/2019 8:38 PM	Text Document	1 KB

12. Extract the Weather Station Simulator source code from the WeatherStationSimulator.zip file

13. Update the paths in the `includePath` property within the **c_cpp_properties.json** file to your local paths

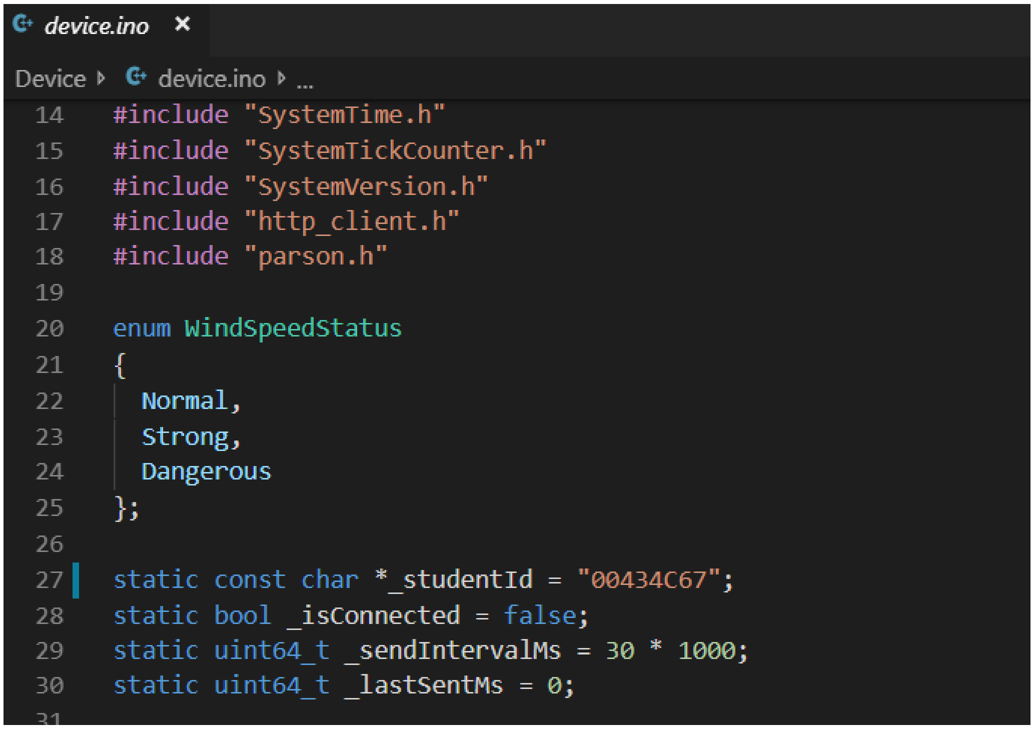

{} c_cpp_properties.json ×
.vscode ▸ {} c_cpp_properties.json ▸ [ ]configurations ▸ { } 0 ▸ [ ]forcedInclude

```
{
    "configurations": [
        {
            "name": "Win32",
            "includePath": [
                "C:\\\\Users\\\\Dennis\\\\AppData\\\\Local\\\\Arduino15\\\\packages\\\\AZ3166\\\\tools\\\\**",
                "C:\\\\Users\\\\Dennis\\\\AppData\\\\Local\\\\Arduino15\\\\packages\\\\AZ3166\\\\hardware\\\\stm32f4\\\\1.6.2\\\\**"
            ],
            "forcedInclude": [
                "C:\\\\Users\\\\Dennis\\\\AppData\\\\Local\\\\Arduino15\\\\packages\\\\AZ3166\\\\hardware\\\\stm32f4\\\\1.6.2\\\\cores\\\\arduino\\\\Arduino.h"
            ],
            "intelliSenseMode": "msvc-x64",
            "compilerPath": "E:\\MinGW\\bin\\gcc.exe",
            "cStandard": "c11",
            "cppStandard": "c++17"
        }
    ],
    "version": 4
}
```



```
{
    "configurations": [
        {
            "name": "Win32",
            "defines": [
                "ARDUINO=10800"
            ],
            "includePath": [
                "C:\\Users\\Dennis\\Documents\\Arduino\\libraries\\MXChip.IoT.Capstone.Library\\src",
                "${workspaceFolder}",
                "${workspaceFolder}/device",
                "${workspaceFolder}/device/**",
                "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\hardware\\stm32f4\\1.6.2\\**",
                "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\tools\\**",
                "C:\\Program Files (x86)\\Arduino\\hardware\\tools\\**",
                "C:\\Program Files (x86)\\Arduino\\libraries\\**",
                "C:\\Users\\Dennis\\Documents\\Arduino\\hardware\\tools\\**",
                "C:\\Users\\Dennis\\Documents\\Arduino\\libraries\\**"
            ],
            "forcedInclude": [
                "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\hardware\\stm32f4\\1.6.2\\cores\\arduino\\Arduino.h"
            ],
            "intelliSenseMode": "clang-x64",
            "cStandard": "c11",
            "cppStandard": "c++17"
        }
    ],
    "version": 4
}
```

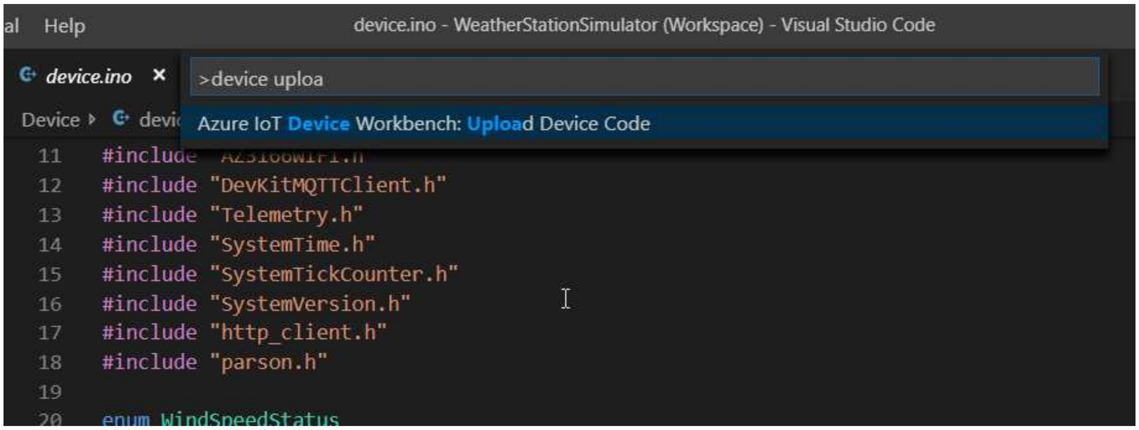
14. Update the Student ID constant value with your ID in **device.ino**



```
#include "SystemTime.h"
#include "SystemTickCounter.h"
#include "SystemVersion.h"
#include "http_client.h"
#include "parson.h"

enum WindSpeedStatus
{
  Normal,
  Strong,
  Dangerous
};

static const char *_studentId = "00434C67";
static bool _isConnected = false;
static uint64_t _sendIntervalMs = 30 * 1000;
static uint64_t _lastSentMs = 0;
```

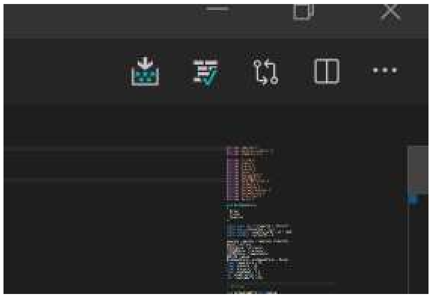
15. Upload device code to AZ3166. Note this may be repeated after device is registered with IOT Hub

(DO NOT DO THIS FIRST) – Must Config Device Connection String see sections below


al Help
device.ino - WeatherStationSimulator (Workspace) - Visual Studio Code
device.ino ×
Device ▸ devic
>device uploa
Azure IoT Device Workbench: Upload Device Code

```
#include "AZ3166WiFi.h"
#include "DevKitMQTTClient.h"
#include "Telemetry.h"
#include "SystemTime.h"
#include "SystemTickCounter.h"
#include "SystemVersion.h"
#include "http_client.h"
#include "parson.h"

enum WindSpeedStatus
```

Another method:



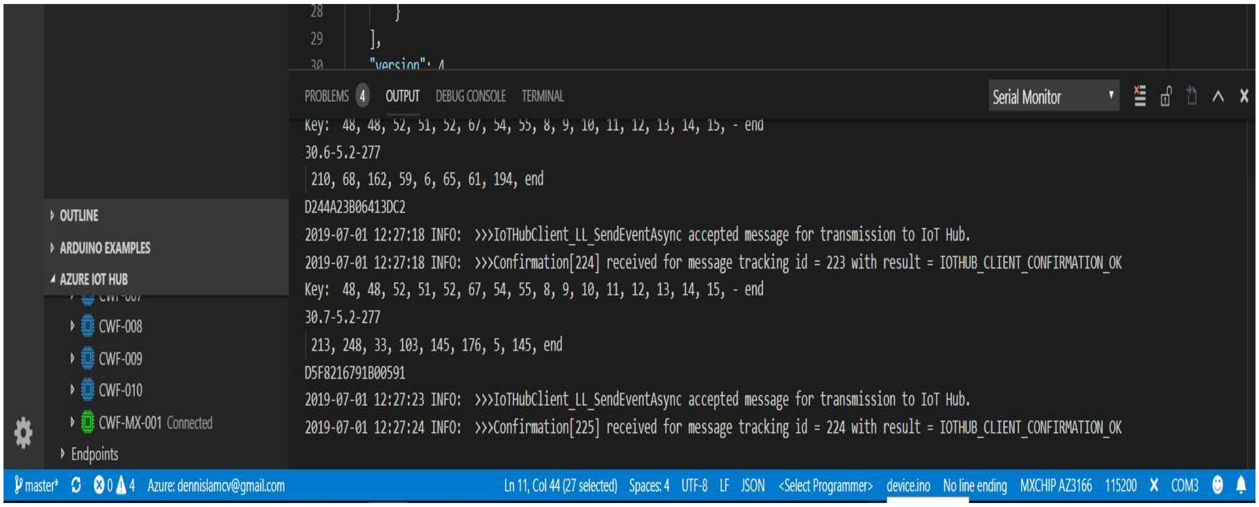
16. In the Visual Studio Code terminal window, notice that (after a few seconds) you are prompted to set your AZ3166 device into configuration mode.

17. On your device, press and hold the A button, and then push and release the Reset button.

18. The task completes a number of Arduino and MXChip AZ3166 verification steps and then begins the process of building and uploading your Arduino sketch.

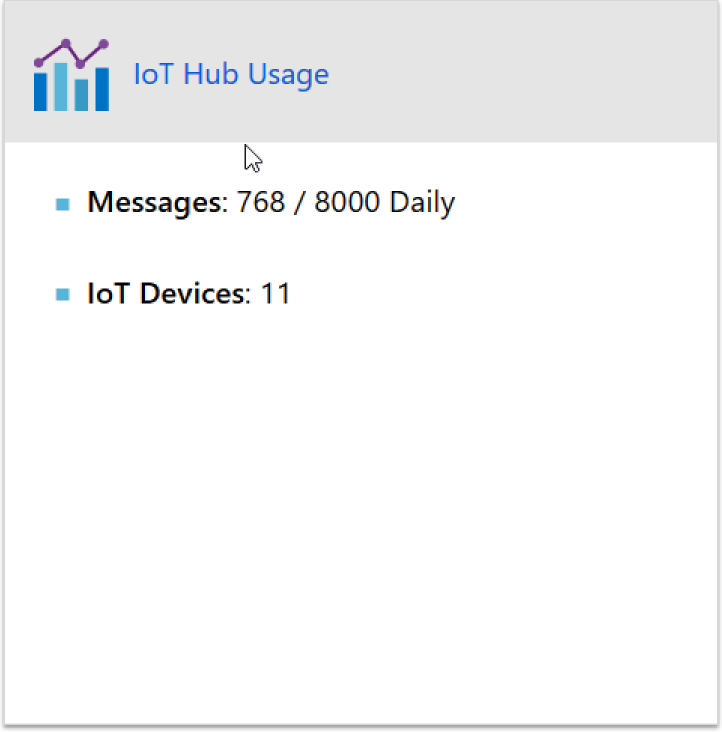
19. After 15-20 seconds, your device will reboot and you should see a message indicating that the Build and Upload process for your Arduino sketch completed successfully.

20. Verify that device-to-cloud communication is taking place by using the Serial Monitor in Visual Studio Code and checking your Azure Portal.



```
        }
    ],
    "version": 4
```

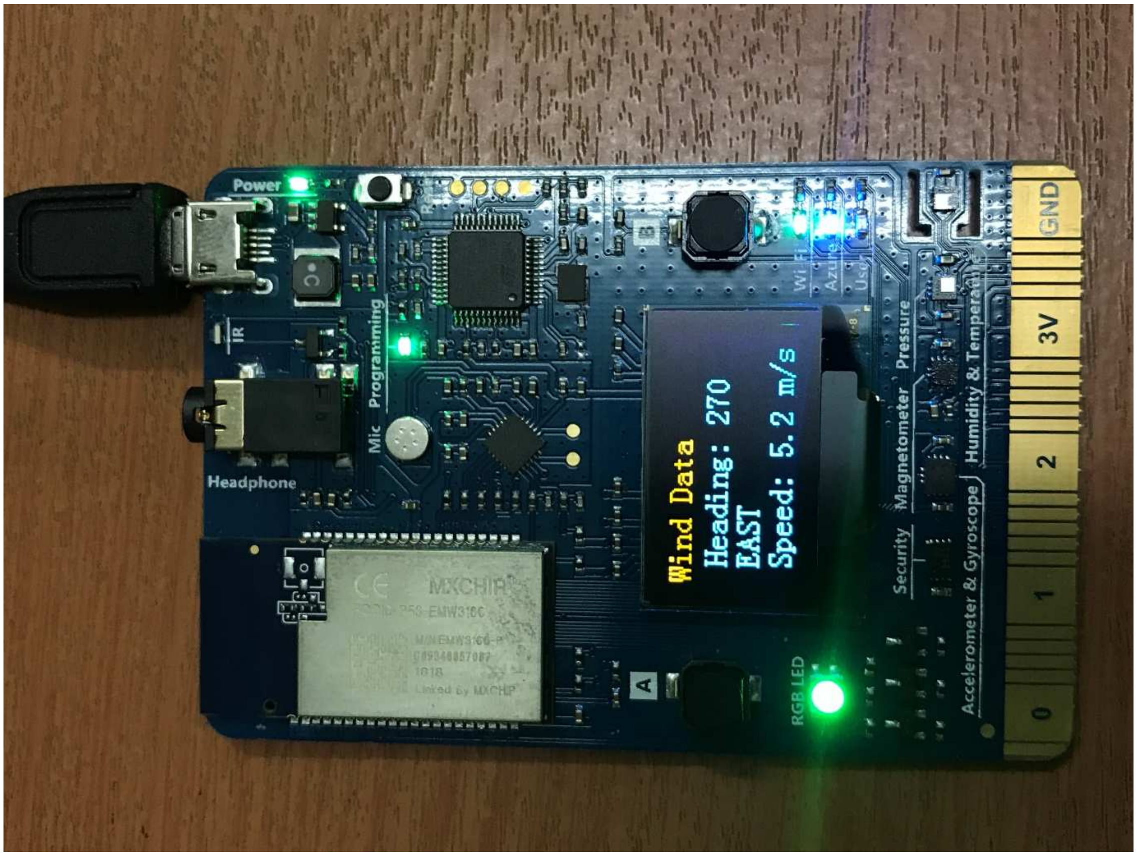
```
Key:  48, 48, 52, 51, 52, 67, 54, 55, 8, 9, 10, 11, 12, 13, 14, 15, - end
30.6-5.2-277
 210, 68, 162, 59, 6, 65, 61, 194, end
D244A23B06413DC2
2019-07-01 12:27:18 INFO:  >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub.
2019-07-01 12:27:18 INFO:  >>>Confirmation[224] received for message tracking id = 223 with result = IOTHUB_CLIENT_CONFIRMATION_OK
Key:  48, 48, 52, 51, 52, 67, 54, 55, 8, 9, 10, 11, 12, 13, 14, 15, - end
30.7-5.2-277
 213, 248, 33, 103, 145, 176, 5, 145, end
D5F8216791B00591
2019-07-01 12:27:23 INFO:  >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub.
2019-07-01 12:27:24 INFO:  >>>Confirmation[225] received for message tracking id = 224 with result = IOTHUB_CLIENT_CONFIRMATION_OK
```



IoT Hub Usage

- **Messages**: 768 / 8000 Daily
- **IoT Devices**: 11

Working Device after update with A and B can change windspeed



21. Create **standard S1** IOT Hub in Azure

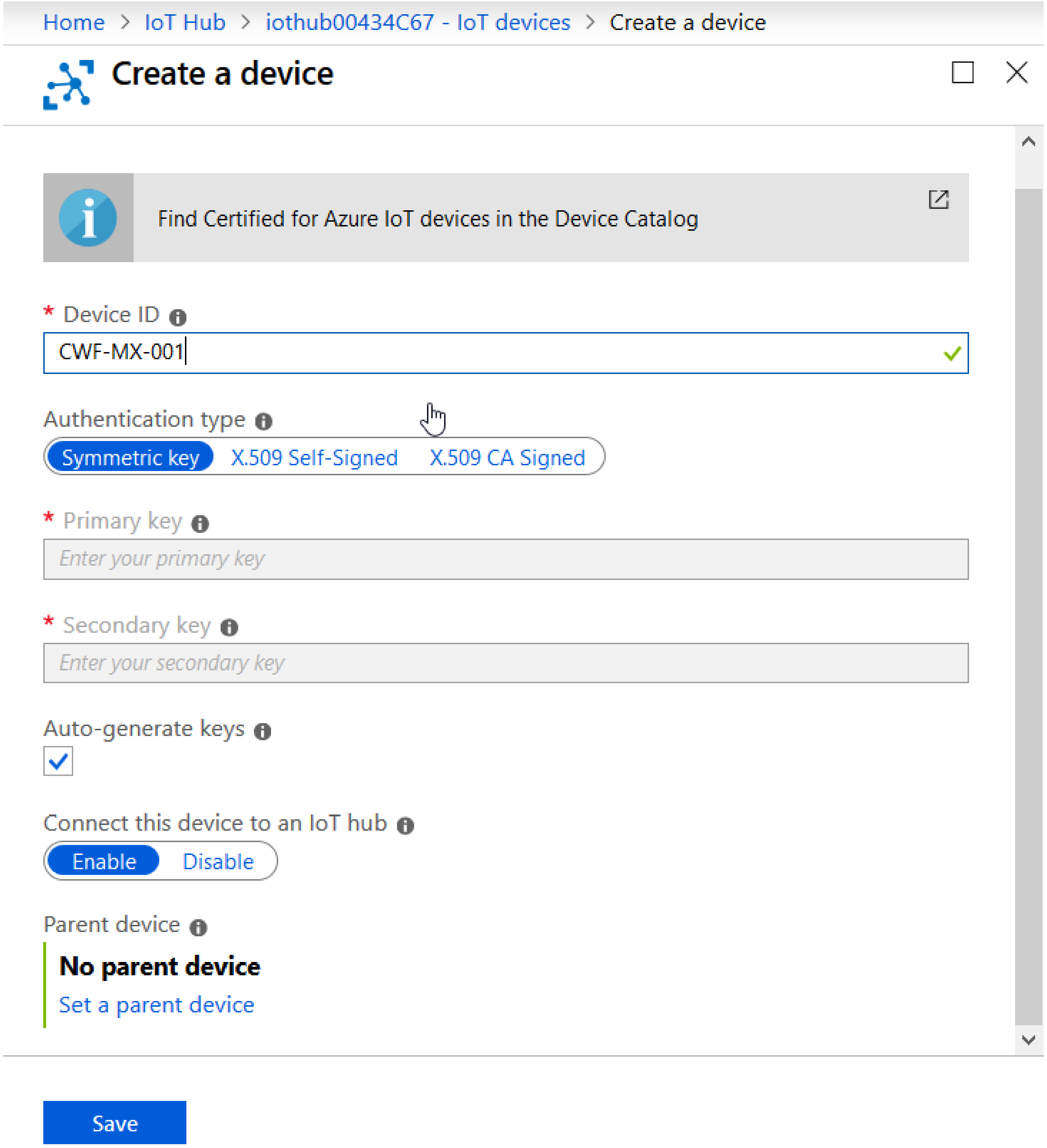
22. Register your MXChip AZ3166 device with your IoT Hub

23. Ensure that you have the Example sample open in Visual Studio Code and that your MXChip AZ3166 device is connected to your PC

24. Provision Azure IoT Hub and device

24a. Add a new Device. Click **IoT Devices** on the left pane under **Explorers**

24b. Use the name **CWF-MX-001**



Home > IoT Hub > iothub00434C67 - IoT devices > Create a device

Create a device

Find Certified for Azure IoT devices in the Device Catalog

* Device ID: CWF-MX-001

Authentication type: [✓ Symmetric key] X.509 Self-Signed X.509 CA Signed

* Primary key: Enter your primary key

* Secondary key: Enter your secondary key

Auto-generate keys ☑

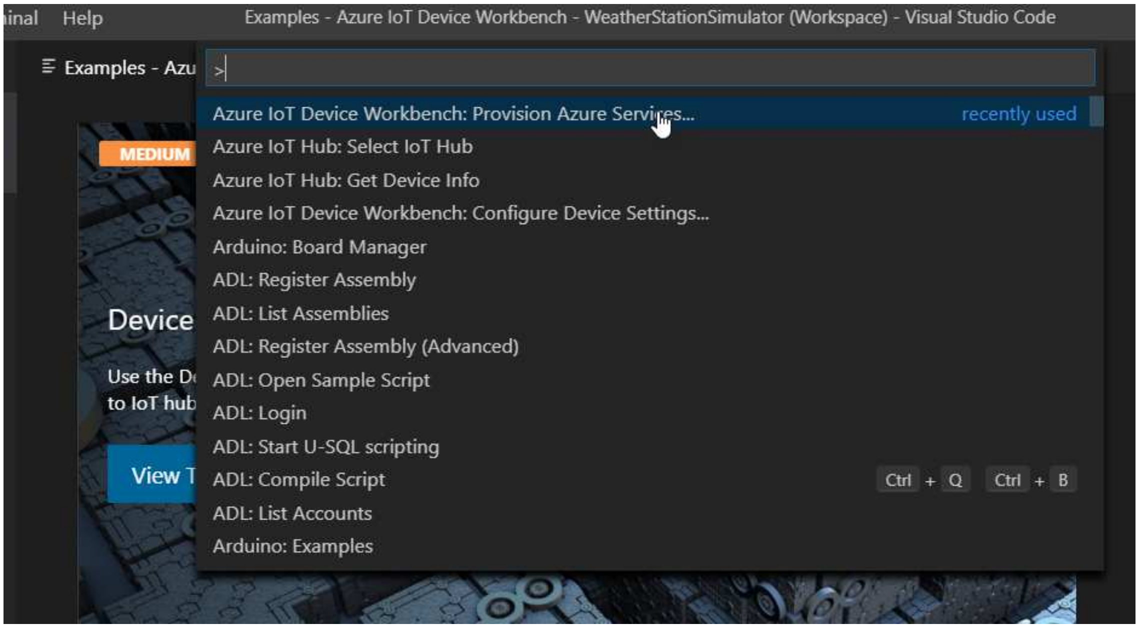
Connect this device to an IoT hub: [✓ Enable] Disable

Parent device: **No parent device**

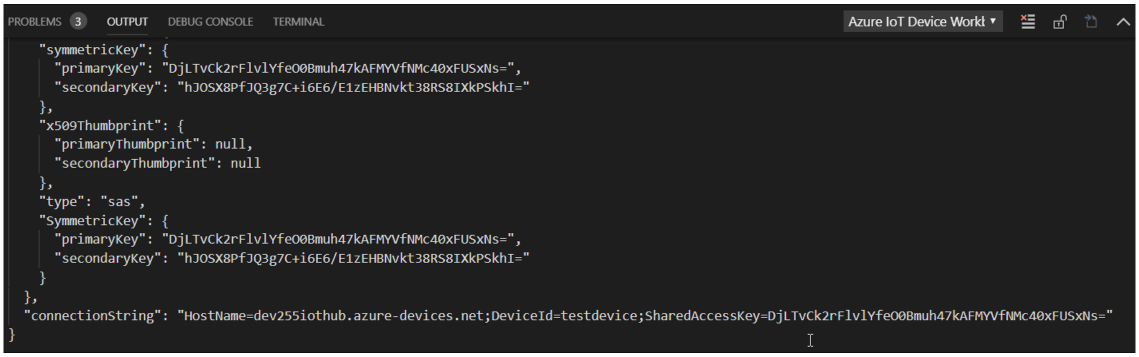
Set a parent device

Save

25. In the new opened project window, click `F1` to open the command palette, type and select **Azure IoT Device Workbench: Provision Azure Services...**. Follow the step by step guide to finish provisioning your Azure IoT Hub and creating the IoT Hub device

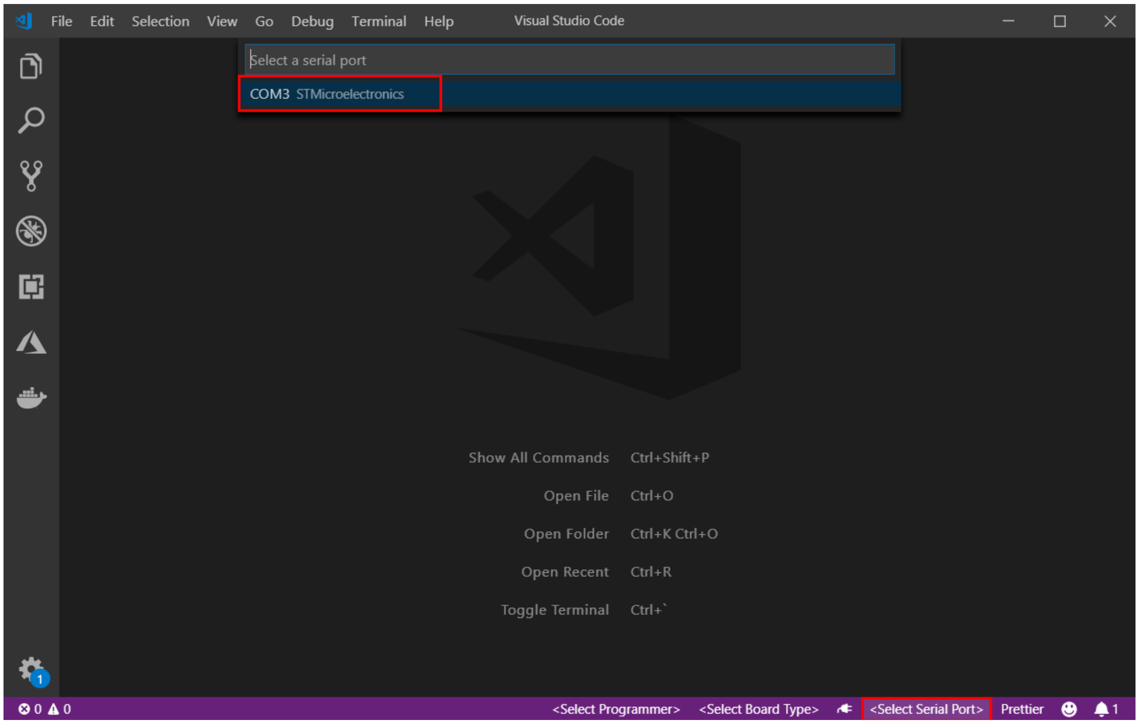

inal
Help
Examples - Azure IoT Device Workbench - WeatherStationSimulator (Workspace) - Visual Studio Code
Examples - Azu
>
Azure IoT Device Workbench: Provision Azure Services...
recently used
Azure IoT Hub: Select IoT Hub
Azure IoT Hub: Get Device Info
Azure IoT Device Workbench: Configure Device Settings...
Arduino: Board Manager
ADL: Register Assembly
ADL: List Assemblies
ADL: Register Assembly (Advanced)
ADL: Open Sample Script
ADL: Login
ADL: Start U-SQL scripting
ADL: Compile Script
Ctrl + Q
Ctrl + B
ADL: List Accounts
Arduino: Examples
MEDIUM
Device
Use the D
to IoT hub
View T

26. Now you have Azure IoT Hub provisioned and device created in it. Also the device connection string will be saved in VS Code for configuring the IoT DevKit later.

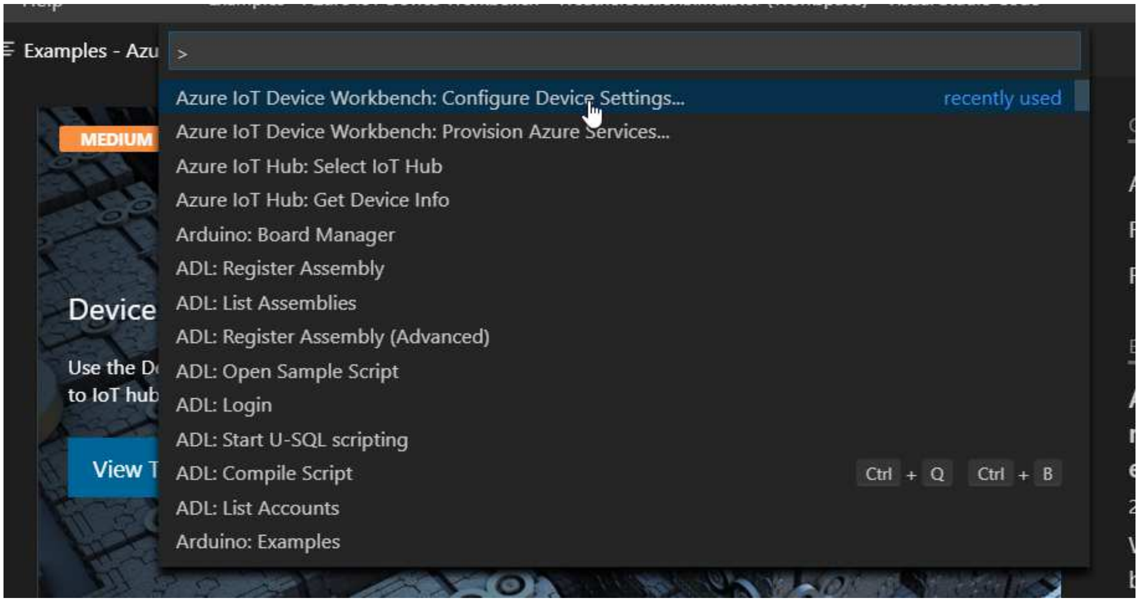


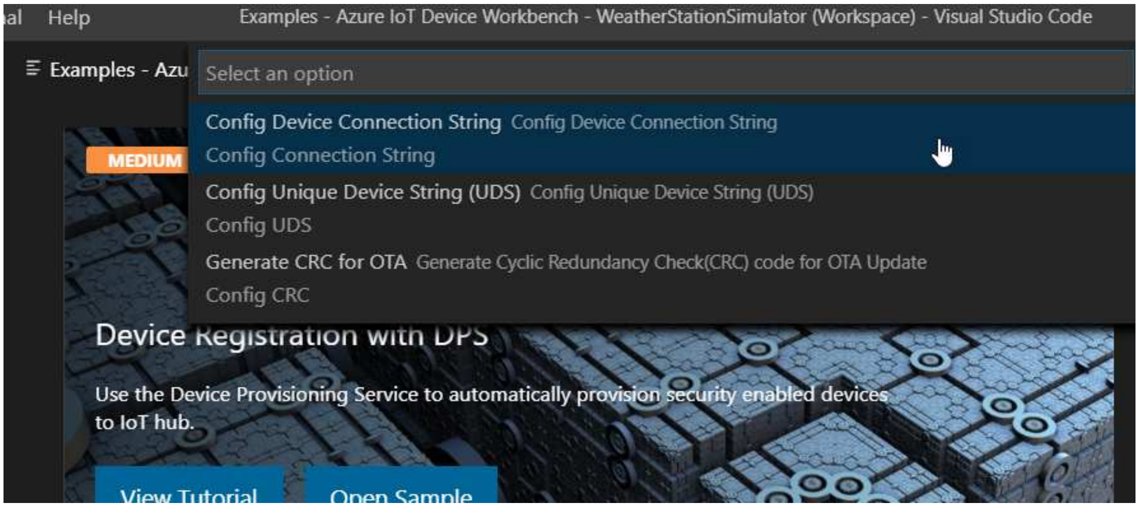
```
    "symmetricKey": {
      "primaryKey": "DjLTvCk2rFlvlYfeO0Bmuh47kAFMYVfNMc40xFUSxNs=",
      "secondaryKey": "hJOSX8PfJQ3g7C+i6E6/E1zEHBNvkt38RS8IXkPSkhI="
    },
    "x509Thumbprint": {
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    },
    "type": "sas",
    "SymmetricKey": {
      "primaryKey": "DjLTvCk2rFlvlYfeO0Bmuh47kAFMYVfNMc40xFUSxNs=",
      "secondaryKey": "hJOSX8PfJQ3g7C+i6E6/E1zEHBNvkt38RS8IXkPSkhI="
    }
  },
  "connectionString": "HostName=dev255iothub.azure-devices.net;DeviceId=testdevice;SharedAccessKey=DjLTvCk2rFlvlYfeO0Bmuh47kAFMYVfNMc40xFUSxNs="
}
```

27. In the bottom-right status bar, check the **MXCHIP AZ3166** is shown as selected board and serial port with **STMicroelectronics** is used.

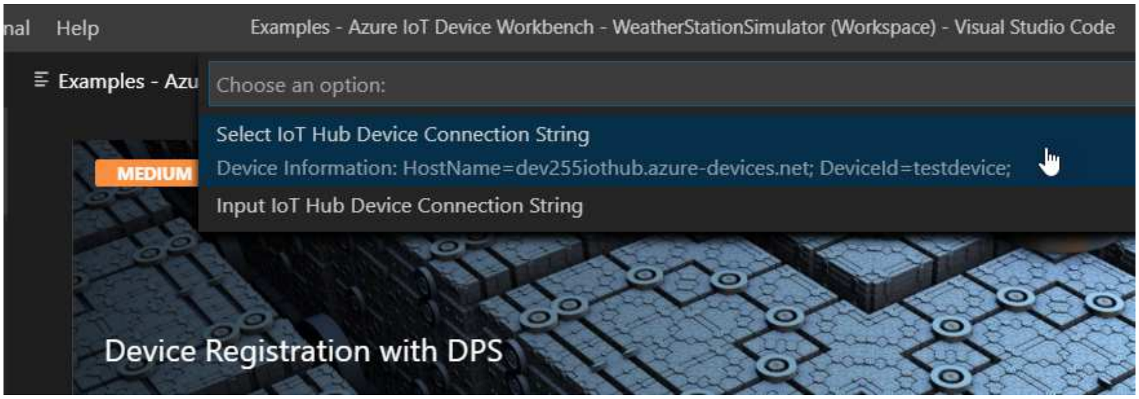


28. Click `F1` to open the command palette, type and select **Azure IoT Device Workbench: Configure Device Settings...**, then select **Config Device Connection String > Select IoT Hub Device Connection String**.

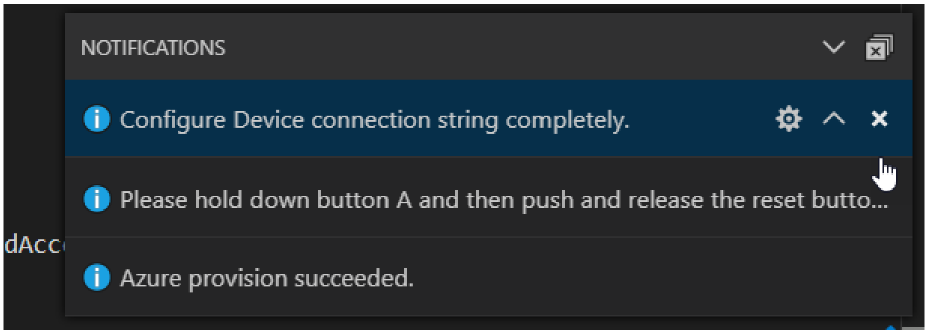

Examples - Azu
>
Azure IoT Device Workbench: Configure Device Settings... recently used
Azure IoT Device Workbench: Provision Azure Services...
Azure IoT Hub: Select IoT Hub
Azure IoT Hub: Get Device Info
Arduino: Board Manager
ADL: Register Assembly
ADL: List Assemblies
ADL: Register Assembly (Advanced)
ADL: Open Sample Script
ADL: Login
ADL: Start U-SQL scripting
ADL: Compile Script Ctrl + Q Ctrl + B
ADL: List Accounts
Arduino: Examples
MEDIUM
Device
Use the D
to IoT hub
View T


al Help
Examples - Azure IoT Device Workbench - WeatherStationSimulator (Workspace) - Visual Studio Code
Examples - Azu
Select an option
Config Device Connection String Config Device Connection String
Config Connection String
Config Unique Device String (UDS) Config Unique Device String (UDS)
Config UDS
Generate CRC for OTA Generate Cyclic Redundancy Check(CRC) code for OTA Update
Config CRC
MEDIUM
Device Registration with DPS
View Tutorial
Open Sample

Use the Device Provisioning Service to automatically provision security enabled devices to IoT hub.


nal Help
Examples - Azure IoT Device Workbench - WeatherStationSimulator (Workspace) - Visual Studio Code
Examples - Azu
Choose an option:
Select IoT Hub Device Connection String
Device Information: HostName=dev255iothub.azure-devices.net; DeviceId=testdevice;
Input IoT Hub Device Connection String
MEDIUM
Device Registration with DPS

29. On DevKit, hold down **button A**, push and release the **reset** button, and then release **button A**. Your DevKit enters configuration mode and saves the connection string.

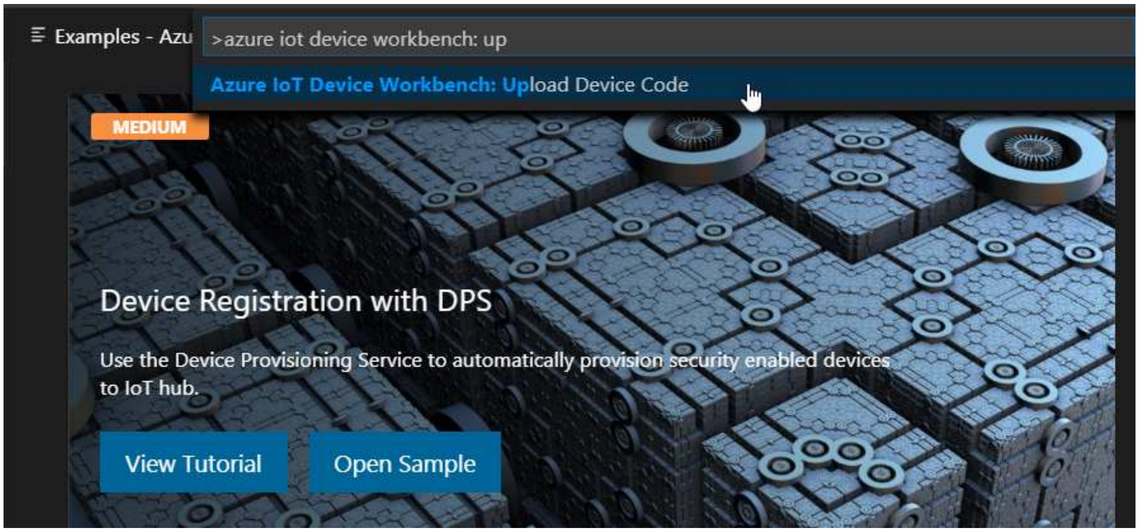


Configure Device connection string completely.

Please hold down button A and then push and release the reset butto...

Azure provision succeeded.

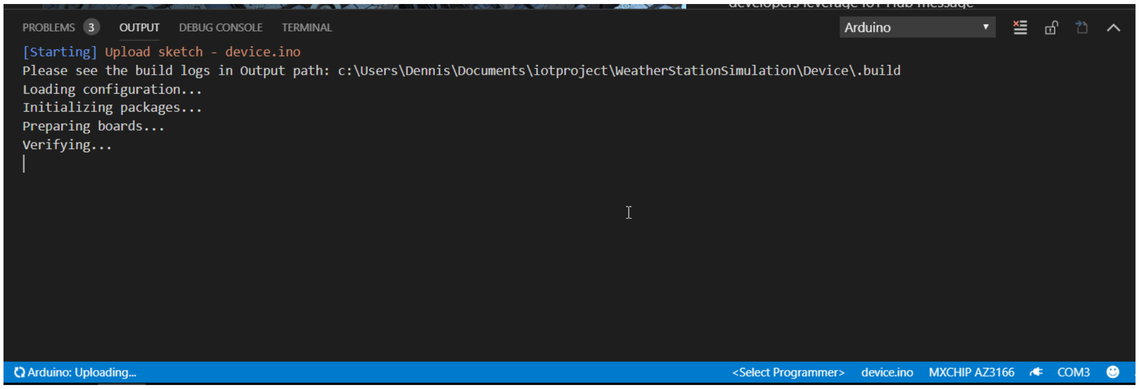
30. Click `F1` again, type and select **Azure IoT Device Workbench: Upload Device Code**. It starts compile and upload the code to DevKit.



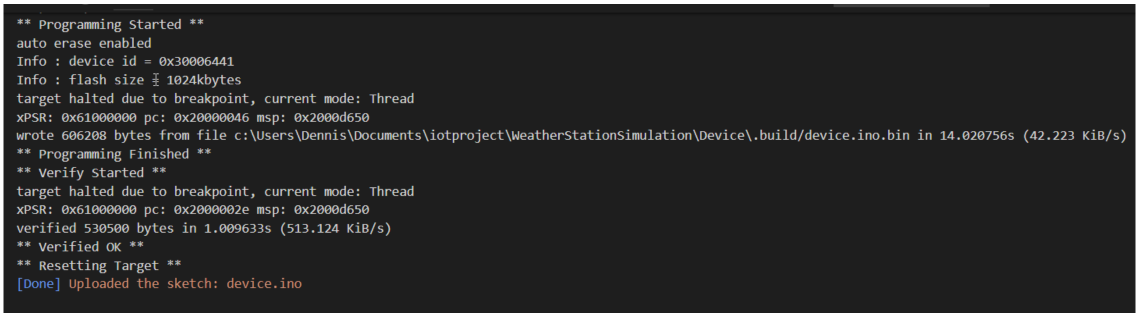
```
>azure iot device workbench: up
```

Examples - Azu
Azure IoT Device Workbench: Upload Device Code
MEDIUM
Device Registration with DPS
Use the Device Provisioning Service to automatically provision security enabled devices to IoT hub.
View Tutorial
Open Sample

The DevKit reboots and starts running the code.

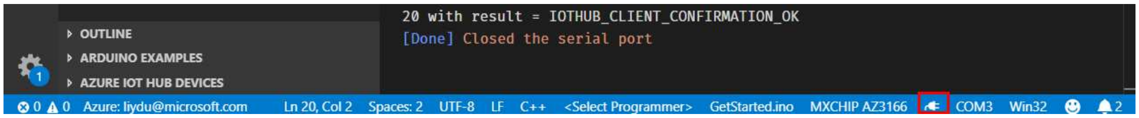


```
[Starting] Upload sketch - device.ino
Please see the build logs in Output path: c:\Users\Dennis\Documents\iotproject\WeatherStationSimulation\Device\.build
Loading configuration...
Initializing packages...
Preparing boards...
Verifying...
```



```
** Programming Started **
auto erase enabled
Info : device id = 0x30006441
Info : flash size = 1024kbytes
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x20000046 msp: 0x2000d650
wrote 606208 bytes from file c:\Users\Dennis\Documents\iotproject\WeatherStationSimulation\Device\.build/device.ino.bin in 14.020756s (42.223 KiB/s)
** Programming Finished **
** Verify Started **
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x2000d650
verified 530500 bytes in 1.009633s (513.124 KiB/s)
** Verified OK **
** Resetting Target **
[Done] Uploaded the sketch: device.ino
```

31. Click the power plug icon on the status bar to open the Serial Monitor:

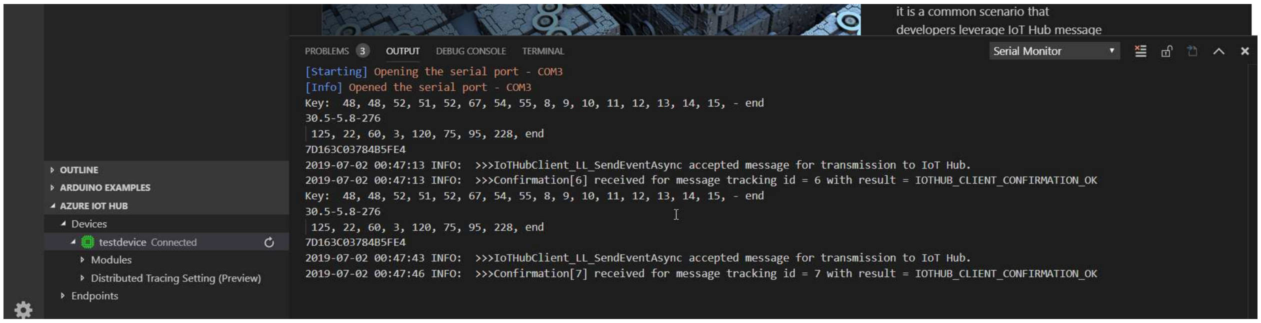


```
20 with result = IOTHUB_CLIENT_CONFIRMATION_OK
[Done] Closed the serial port
```

32. The application is running successfully when you see the following results:

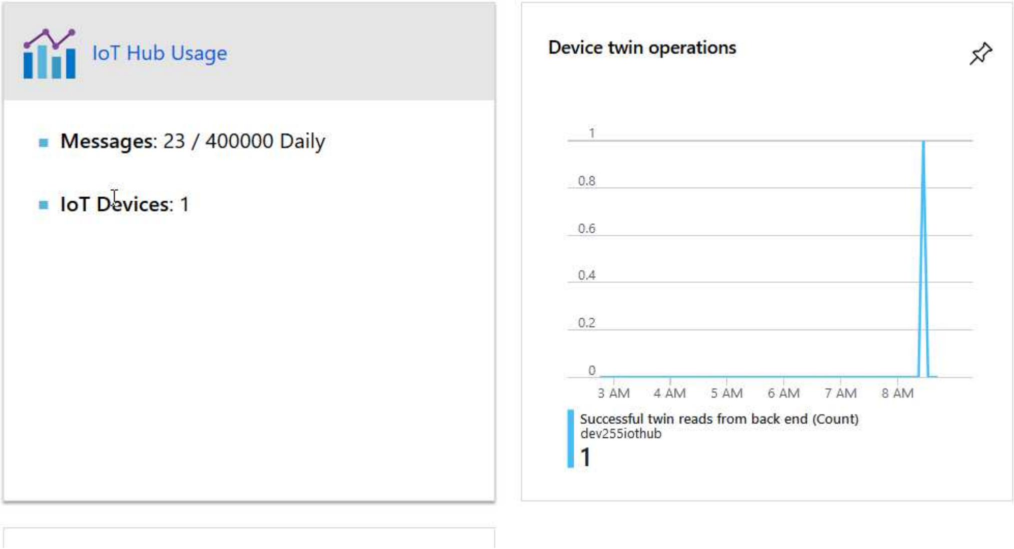
The Serial Monitor displays the message sent to the IoT Hub.

The LED on the MXChip IoT DevKit is blinking.



```
[Starting] Opening the serial port - COM3
[Info] Opened the serial port - COM3
Key:  48, 48, 52, 51, 52, 67, 54, 55, 8, 9, 10, 11, 12, 13, 14, 15, - end
30.5-5.8-276
 125, 22, 60, 3, 120, 75, 95, 228, end
7D163C03784B5FE4
2019-07-02 00:47:13 INFO:  >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub.
2019-07-02 00:47:13 INFO:  >>>Confirmation[6] received for message tracking id = 6 with result = IOTHUB_CLIENT_CONFIRMATION_OK
Key:  48, 48, 52, 51, 52, 67, 54, 55, 8, 9, 10, 11, 12, 13, 14, 15, - end
30.5-5.8-276
 125, 22, 60, 3, 120, 75, 95, 228, end
7D163C03784B5FE4
2019-07-02 00:47:43 INFO:  >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub.
2019-07-02 00:47:46 INFO:  >>>Confirmation[7] received for message tracking id = 7 with result = IOTHUB_CLIENT_CONFIRMATION_OK
```

33. Go to IOT Hub overview to double check

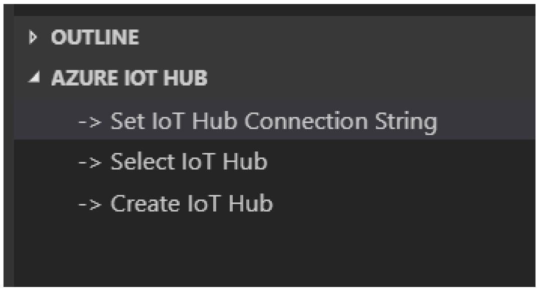


IoT Hub Usage

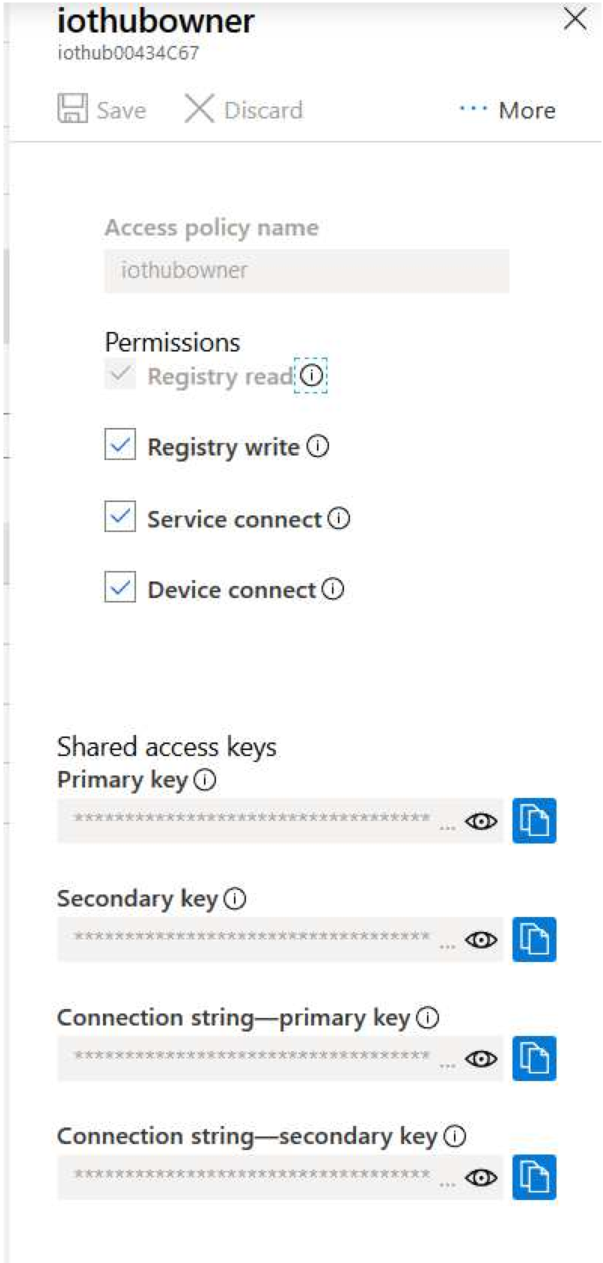
- **Messages**: 23 / 400000 Daily
- **IoT Devices**: 1

Update Nov 2019

We need to link the device to proper IOT Hub in VS Code here:



Set IOT Hub Connection String:



iothubowner

iothub00434C67

Access policy name

iothubowner

Permissions

☑ Registry read ⓘ

☑ Registry write ⓘ

☑ Service connect ⓘ

☑ Device connect ⓘ

Shared access keys

Primary key ⓘ

************************************ ...

Secondary key ⓘ

************************************ ...

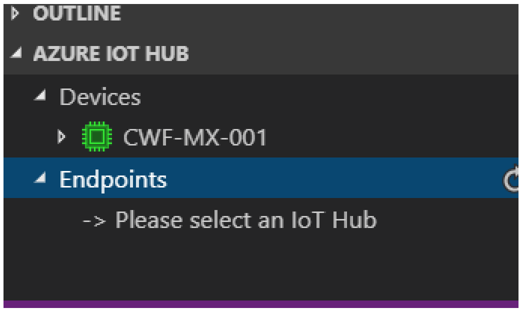
Connection string—primary key ⓘ

************************************ ...

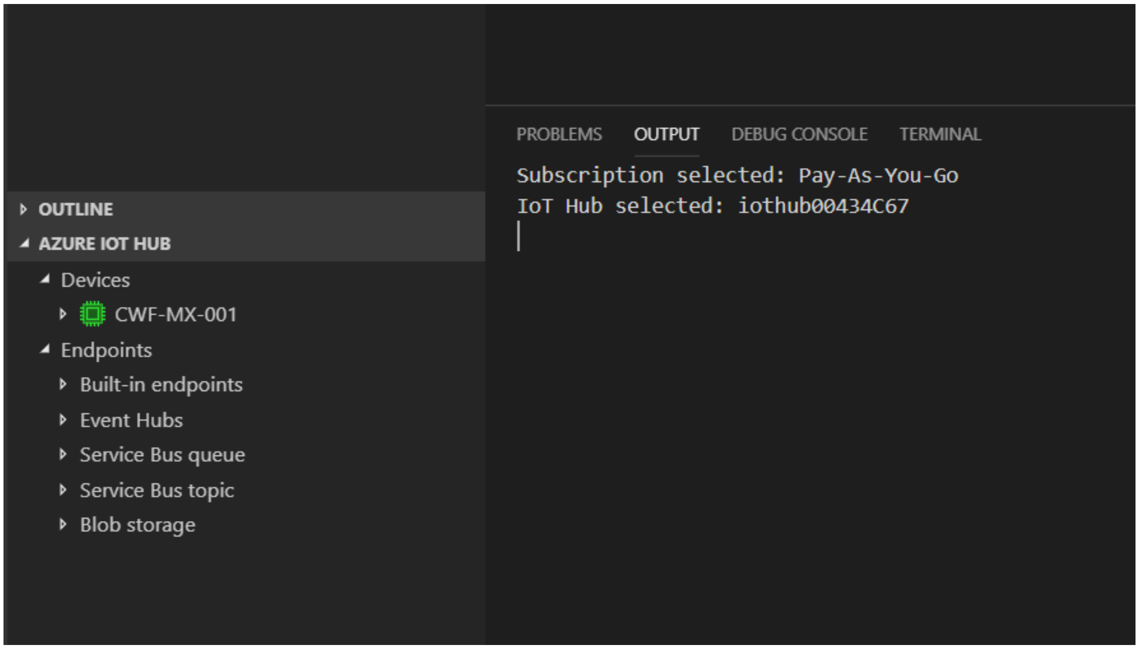
Connection string—secondary key ⓘ

************************************ ...

We get a Green icon

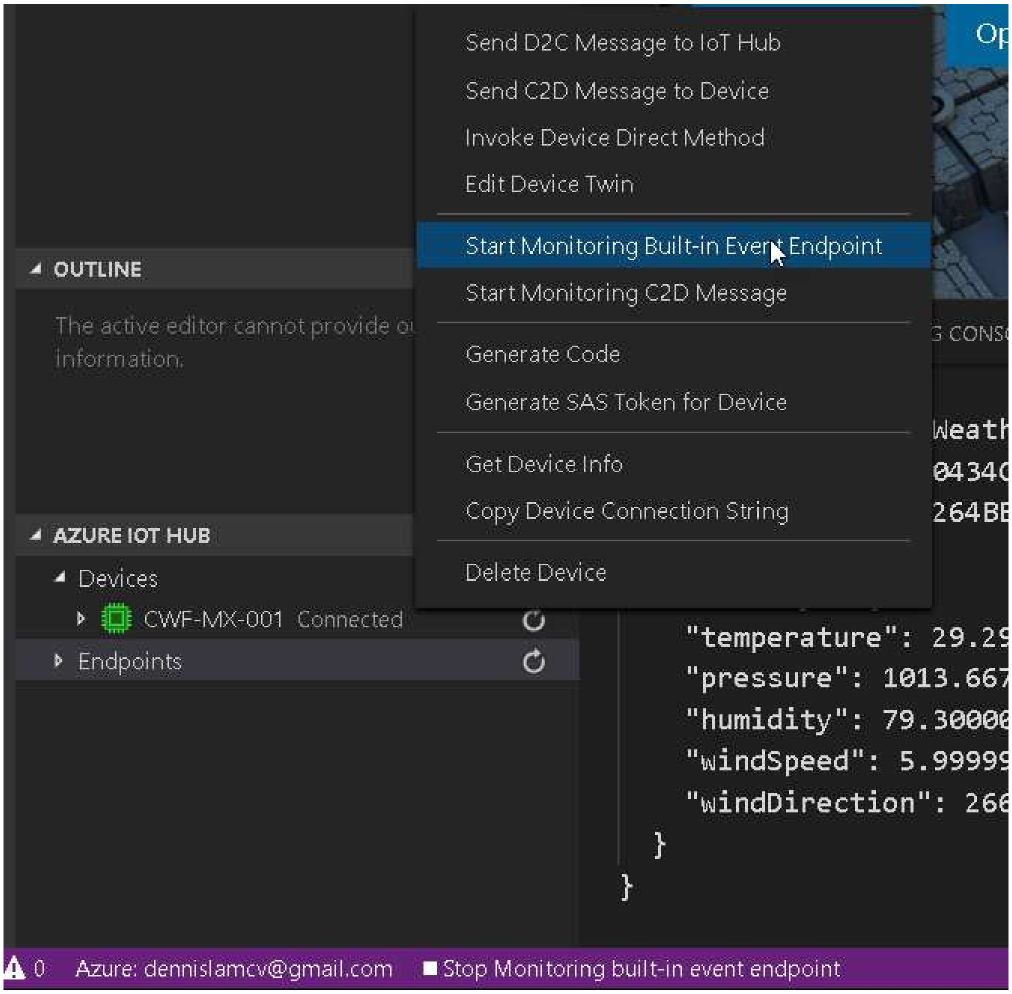


Select IOT Hub



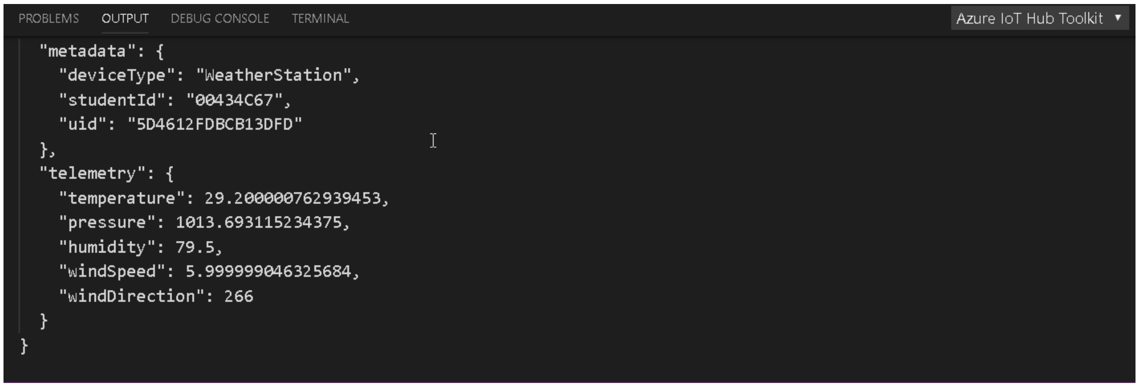
```
Subscription selected: Pay-As-You-Go
IoT Hub selected: iothub00434C67
```

You can also check by using VS Studio Code. Under CWF-MX-001, right click and select Start monitoring built in event.


Send D2C Message to IoT Hub
Send C2D Message to Device
Invoke Device Direct Method
Edit Device Twin
Start Monitoring Built-in Event Endpoint
Start Monitoring C2D Message
Generate Code
Generate SAS Token for Device
Get Device Info
Copy Device Connection String
Delete Device
OUTLINE
The active editor cannot provide ou
information.
AZURE IOT HUB
Devices
CWF-MX-001 Connected
Endpoints
⚠ 0
Azure: dennislamcv@gmail.com
■ Stop Monitoring built-in event endpoint

```
    "temperature": 29.29
    "pressure": 1013.667
    "humidity": 79.30000
    "windSpeed": 5.99999
    "windDirection": 266
  }
}
```

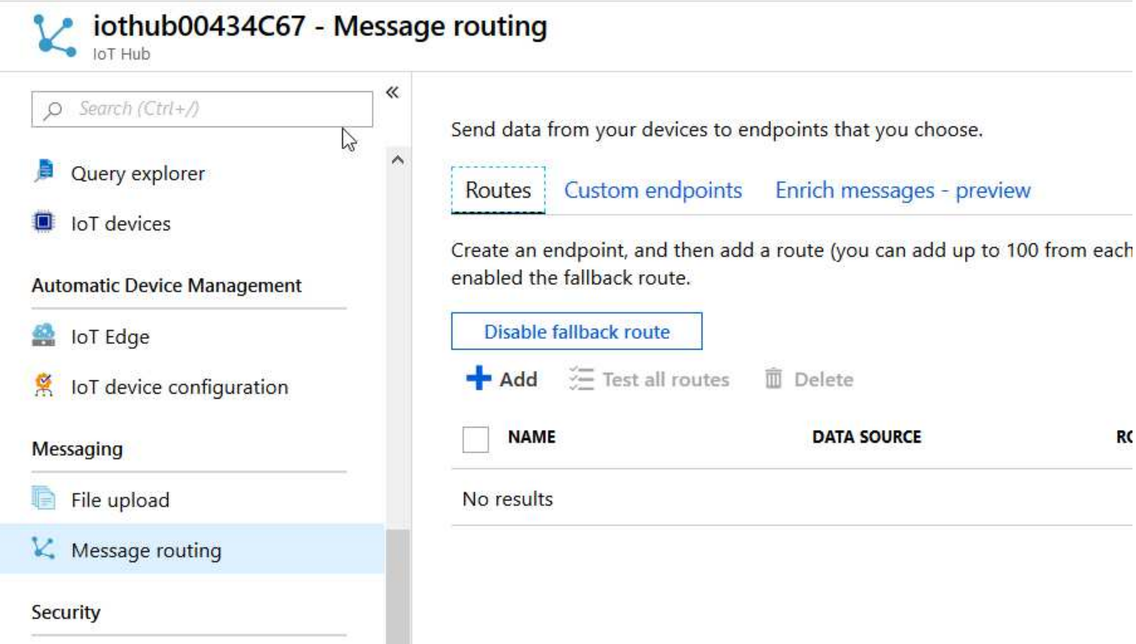
On the output you will see telemetry being sent as per in the requirements,



```
  "metadata": {
    "deviceType": "WeatherStation",
    "studentId": "00434C67",
    "uid": "5D4612FDBCB13DFD"
  },
  "telemetry": {
    "temperature": 29.200000762939453,
    "pressure": 1013.693115234375,
    "humidity": 79.5,
    "windSpeed": 5.999999046325684,
    "windDirection": 266
  }
}
```

You can use Send D2C messages to IOT Hub to check connectivity.

33a: Configure Message Routing in IOT Hub



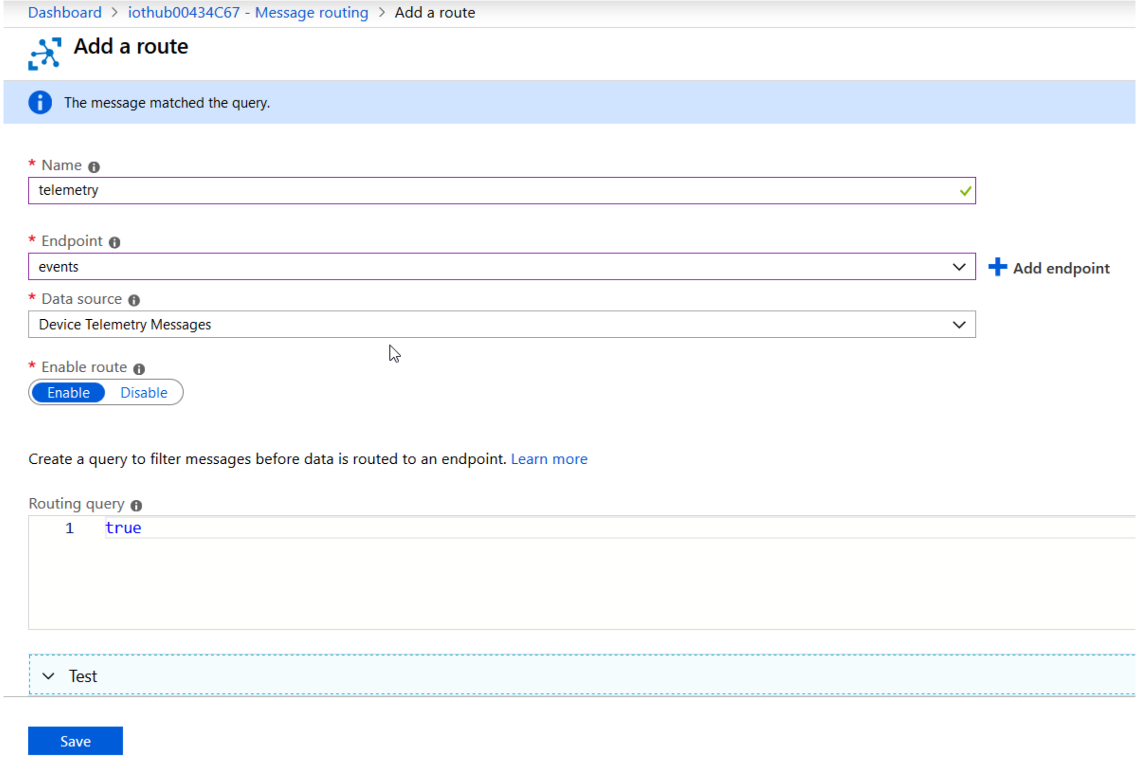
iothub00434C67 - Message routing

IoT Hub

Send data from your devices to endpoints that you choose.

Create an endpoint, and then add a route (you can add up to 100 from each
enabled the fallback route.

☐	NAME	DATA SOURCE	RO
	No results		



Add a route

The message matched the query.

* Name: telemetry

* Endpoint: events

* Data source: Device Telemetry Messages

* Enable route: Enable | Disable

Create a query to filter messages before data is routed to an endpoint. Learn more

Routing query

```
true
```

Do a test to verify its working.

34. On the IoT Hub device settings, configure the device twin by adding `windSpeedStatus` and `sendFrequencySeconds` to the **desired** properties.

Review the following materials:

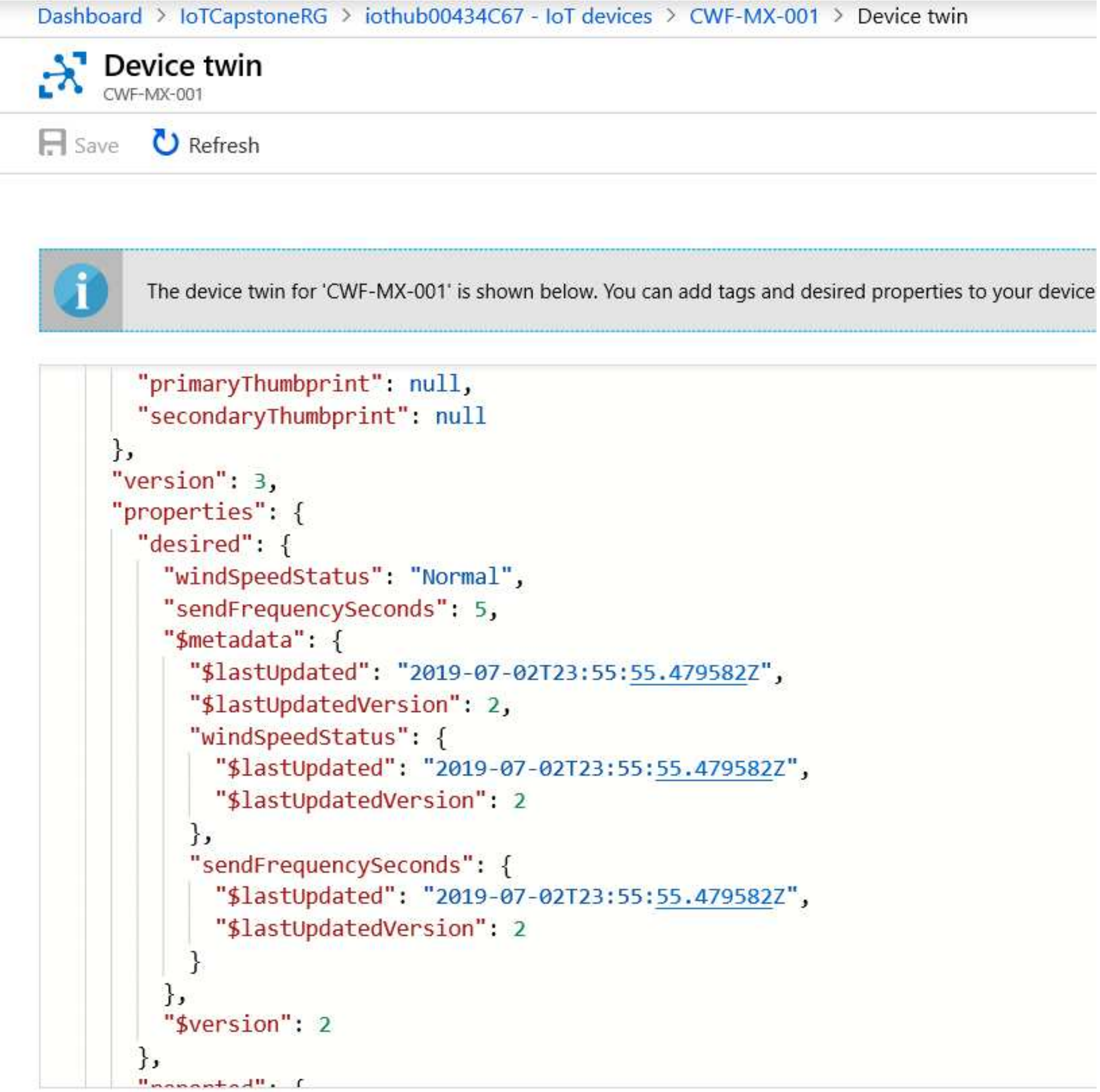
Course: DEV297x IoT Device Configuration and Communication: C Edition

Module Implement Device Communications

Lab: Configuring and Securing IoT Hub Devices

Topic: Access Device Twin Properties from the Back End

35. On Device Twin in IOT Hub, add the properties, desired


Dashboard > IoTCapstoneRG > iothub00434C67 - IoT devices > CWF-MX-001 > Device twin
Device twin
CWF-MX-001
Save
Refresh

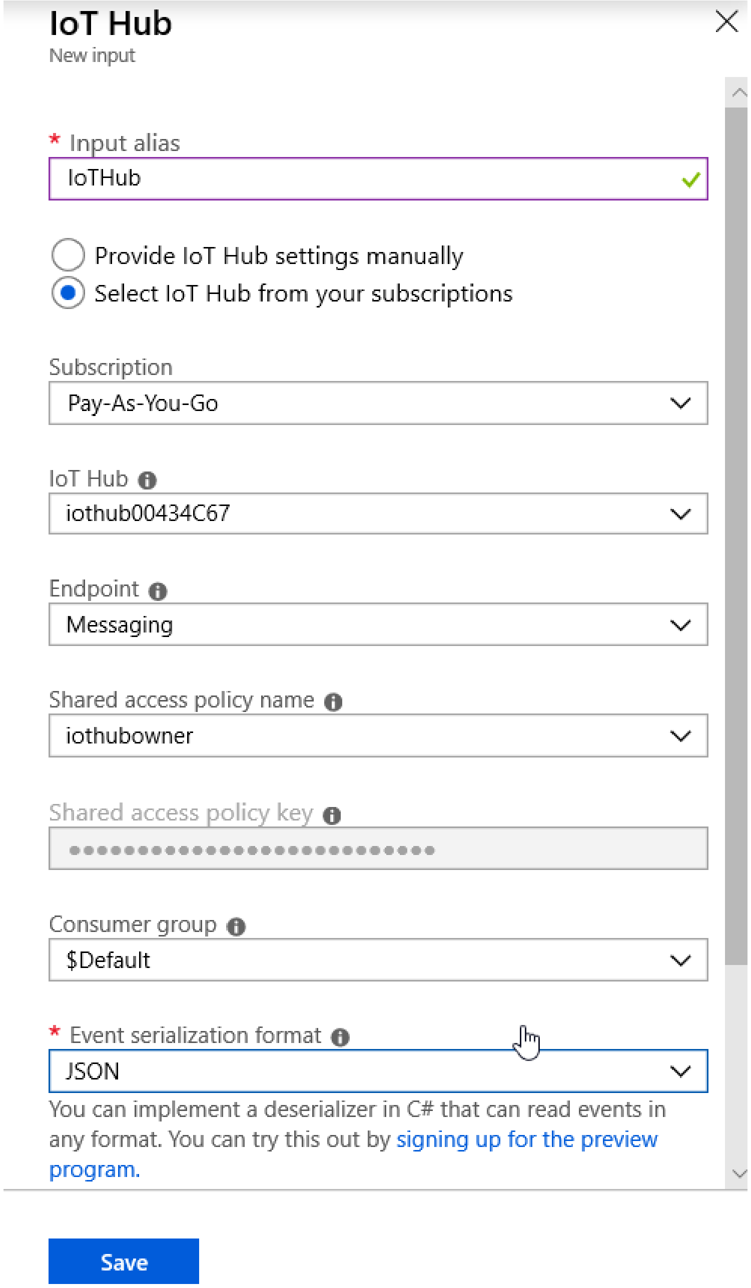
The device twin for 'CWF-MX-001' is shown below. You can add tags and desired properties to your device

```
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    },
    "version": 3,
    "properties": {
      "desired": {
        "windSpeedStatus": "Normal",
        "sendFrequencySeconds": 5,
        "$metadata": {
          "$lastUpdated": "2019-07-02T23:55:55.479582Z",
          "$lastUpdatedVersion": 2,
          "windSpeedStatus": {
            "$lastUpdated": "2019-07-02T23:55:55.479582Z",
            "$lastUpdatedVersion": 2
          },
          "sendFrequencySeconds": {
            "$lastUpdated": "2019-07-02T23:55:55.479582Z",
            "$lastUpdatedVersion": 2
          }
        },
        "$version": 2
      },
```

36. Set up Azure Stream Analytics. Under Hosting environment, select Cloud. Under Streaming units, change the setting to 1.

37. On the Stream Analytics job blade in the left hand nav area, under Job topology, click Inputs.

38. In the Inputs pane, click Add stream input and then select IoT Hub.



IoT Hub

New input

* Input alias

IoTHub ✓

○ Provide IoT Hub settings manually

● Select IoT Hub from your subscriptions

Subscription

Pay-As-You-Go

IoT Hub

iothub00434C67

Endpoint

Messaging

Shared access policy name

iothubowner

Shared access policy key

●●●●●●●●●●●●●●●●●●●●●●●●●●●

Consumer group

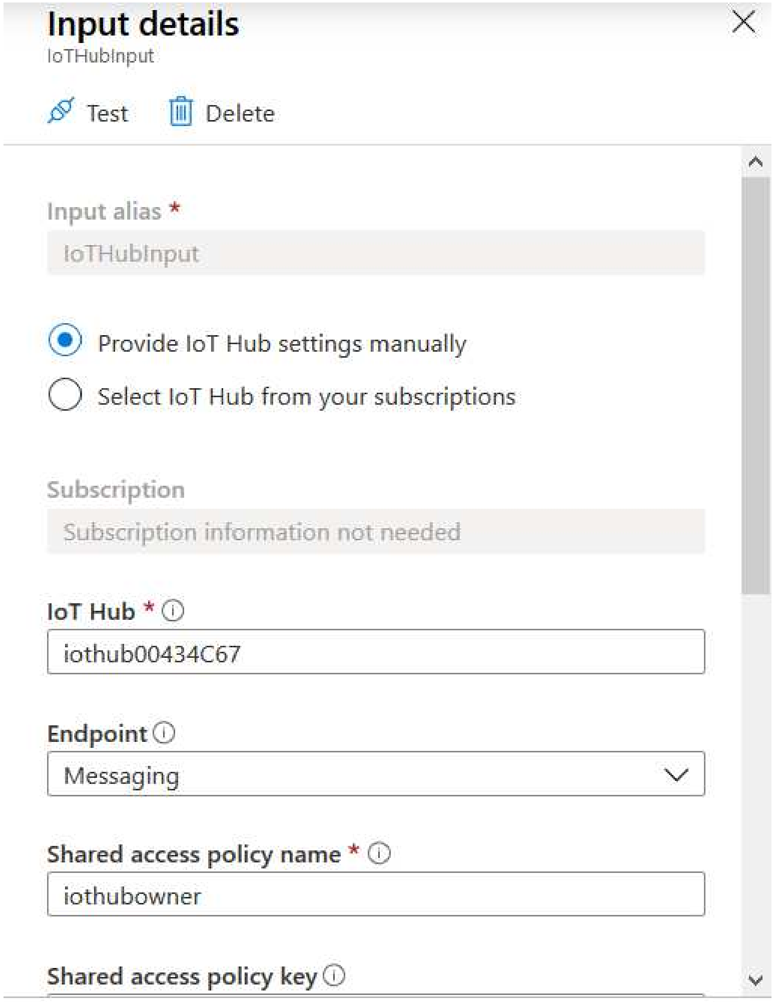
$Default

* Event serialization format

JSON

You can implement a deserializer in C# that can read events in any format. You can try this out by signing up for the preview program.

NOV Update: Changed to IoTHubInput to prevent clash.



Input details

IoTHubInput

Input alias *

IoTHubInput

● Provide IoT Hub settings manually

○ Select IoT Hub from your subscriptions

Subscription

Subscription information not needed

IoT Hub *

iothub00434C67

Endpoint

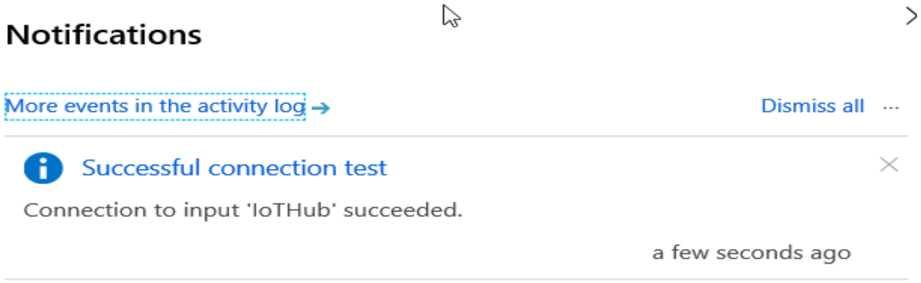
Messaging

Shared access policy name *

iothubowner

Shared access policy key

39. Make sure connection test successful.



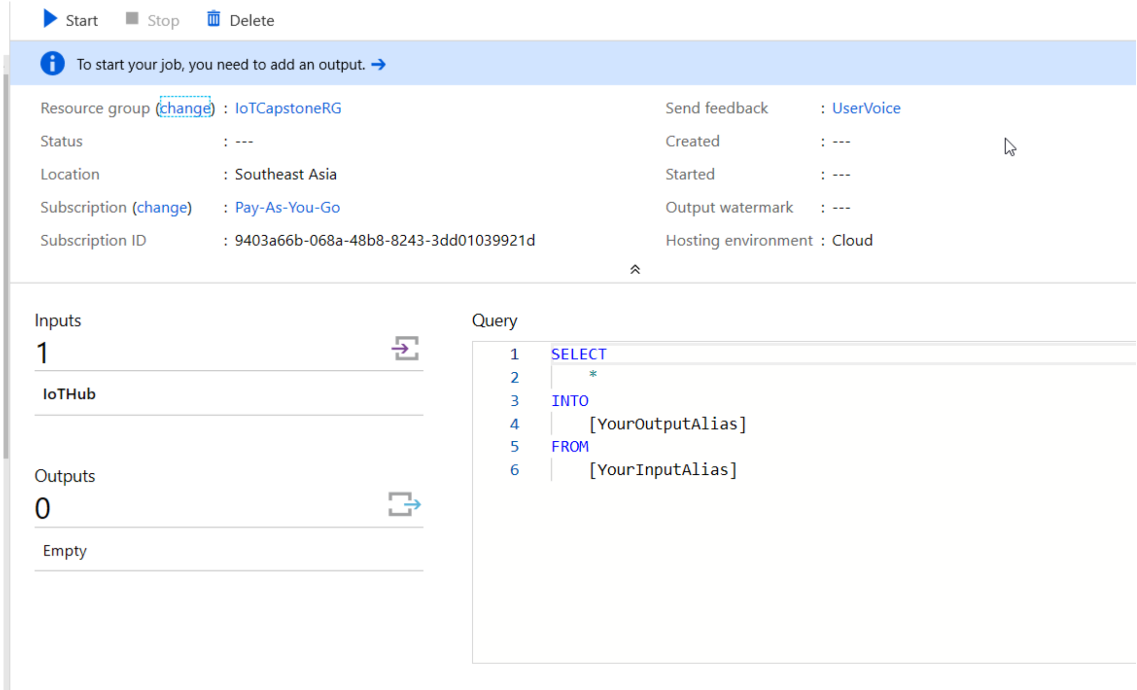
Notifications

Successful connection test

Connection to input 'IoTHub' succeeded.

a few seconds ago

At this point we have only Input.



To start your job, you need to add an output. →

Resource group (change) : IoTCapstoneRG

Status : ---

Location : Southeast Asia

Subscription (change) : Pay-As-You-Go

Subscription ID : 9403a66b-068a-48b8-8243-3dd01039921d

Send feedback : UserVoice

Created : ---

Started : ---

Output watermark : ---

Hosting environment : Cloud

Inputs

1

IoTHub

Outputs

0

Empty

Query

```
SELECT
    *
INTO
    [YourOutputAlias]
FROM
    [YourInputAlias]
```

40. Create an Azure Function

The source for the Azure function can be found here - CapstoneAzFunctions.zip

Review the following materials:

Course: DEV301x IoT Architecture Design and Business Planning

Module PoV and Rollout

Lab: Planning a PoV

Topic: Stream Analytics and Azure Functions (particularly the sections "Create an Azure Function" and "Create a Stream Analytics Job Output" in this topic)

NOV Capstone Changes

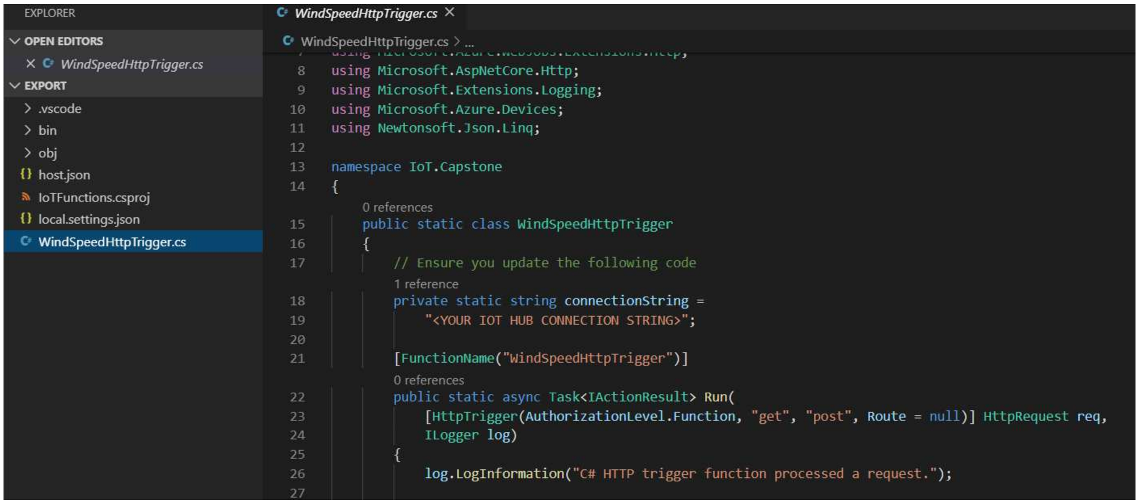
Tip: The supplied Azure function should be deployed via **Visual Studio Code**, not via the built-in method available in the Azure Portal. If you are unclear how to perform this, complete the following steps:

Configure your Visual Studio Code environment as detailed here: **https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-function-vs-code**.

Extract CapstoneAzFunctions.zip to a local folder.

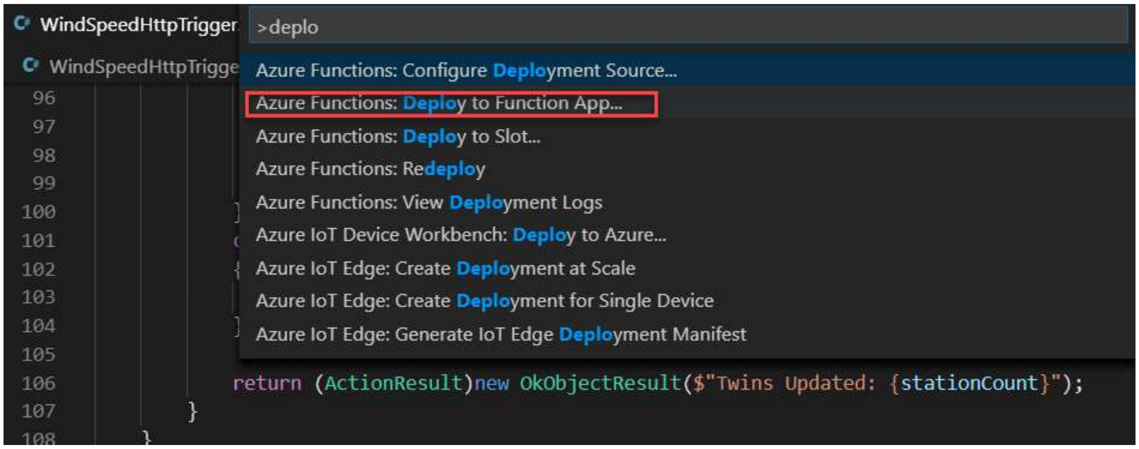
In Visual Studio Code, open the folder you created above.

Update the connectionString value in WindSpeedHttpTrigger.cs.



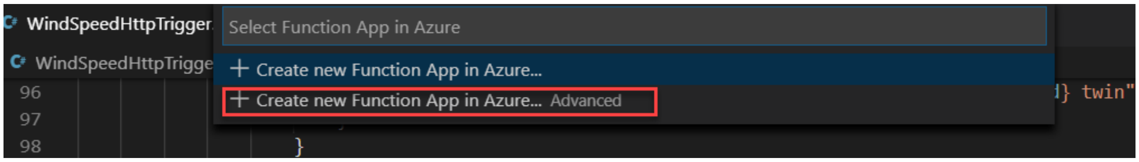
```
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Azure.Devices;
using Newtonsoft.Json.Linq;

namespace IoT.Capstone
{
    public static class WindSpeedHttpTrigger
    {
        // Ensure you update the following code
        private static string connectionString =
            "<YOUR IOT HUB CONNECTION STRING>";

        [FunctionName("WindSpeedHttpTrigger")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
            ILogger log)
        {
            log.LogInformation("C# HTTP trigger function processed a request.");

```

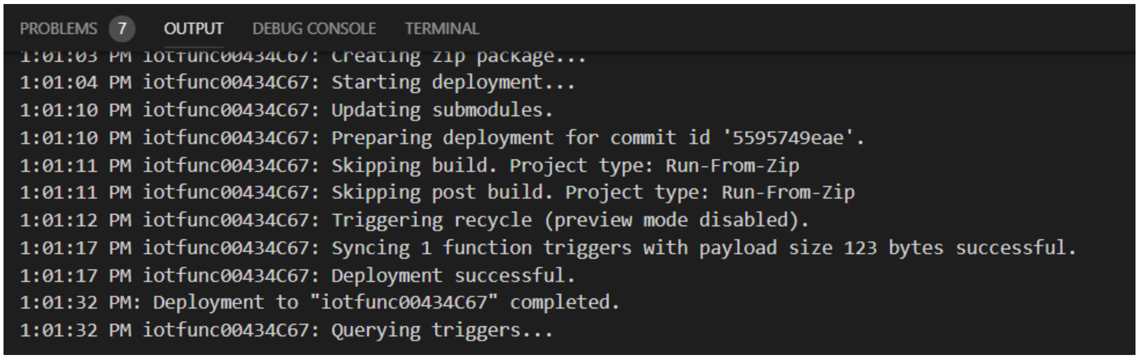
Press F1 to open the Command Palette, then type Azure Functions: Deploy to a function app and select the command.


WindSpeedHttpTrigger
WindSpeedHttpTrigge
>deplo
Azure Functions: Configure Deployment Source...
Azure Functions: Deploy to Function App...
Azure Functions: Deploy to Slot...
Azure Functions: Redeploy
Azure Functions: View Deployment Logs
Azure IoT Device Workbench: Deploy to Azure...
Azure IoT Edge: Create Deployment at Scale
Azure IoT Edge: Create Deployment for Single Device
Azure IoT Edge: Generate IoT Edge Deployment Manifest

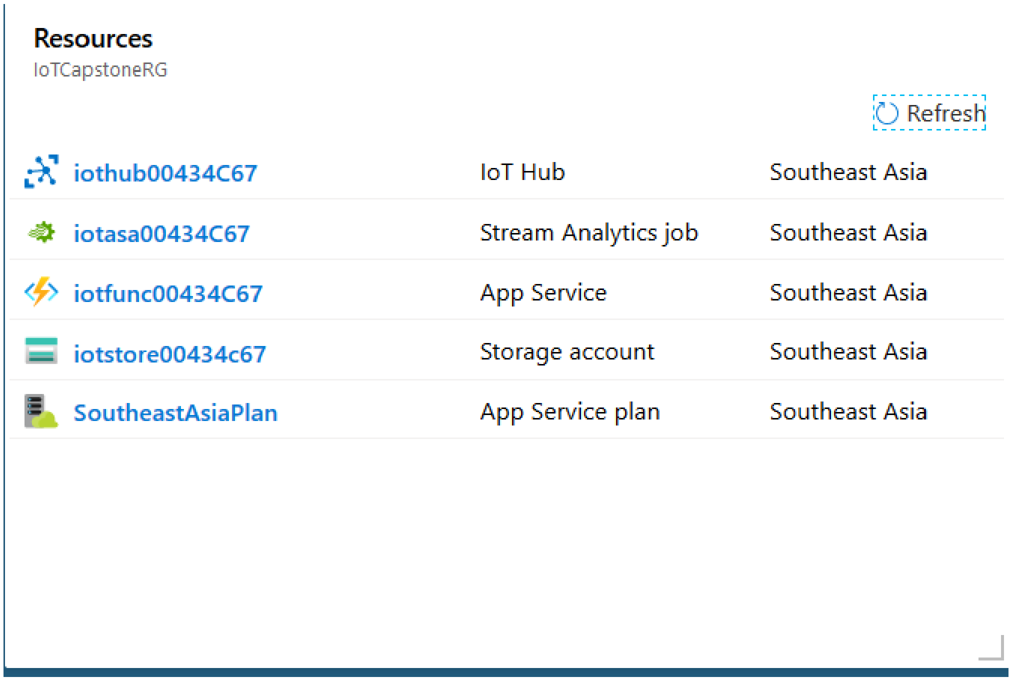
```
                return (ActionResult)new OkObjectResult($"Twins Updated: {stationCount}");
            }
        }
```

Follow the steps to deploy the function - ensure you choose Create new function app advanced so you can specify the Resource Group, etc.


WindSpeedHttpTrigger
WindSpeedHttpTrigge
Select Function App in Azure
+ Create new Function App in Azure...
+ Create new Function App in Azure... Advanced
}} twin"



```
1:01:03 PM iotfunc00434C67: Creating zip package...
1:01:04 PM iotfunc00434C67: Starting deployment...
1:01:10 PM iotfunc00434C67: Updating submodules.
1:01:10 PM iotfunc00434C67: Preparing deployment for commit id '5595749eae'.
1:01:11 PM iotfunc00434C67: Skipping build. Project type: Run-From-Zip
1:01:11 PM iotfunc00434C67: Skipping post build. Project type: Run-From-Zip
1:01:12 PM iotfunc00434C67: Triggering recycle (preview mode disabled).
1:01:17 PM iotfunc00434C67: Syncing 1 function triggers with payload size 123 bytes successful.
1:01:17 PM iotfunc00434C67: Deployment successful.
1:01:32 PM: Deployment to "iotfunc00434C67" completed.
1:01:32 PM iotfunc00434C67: Querying triggers...
```

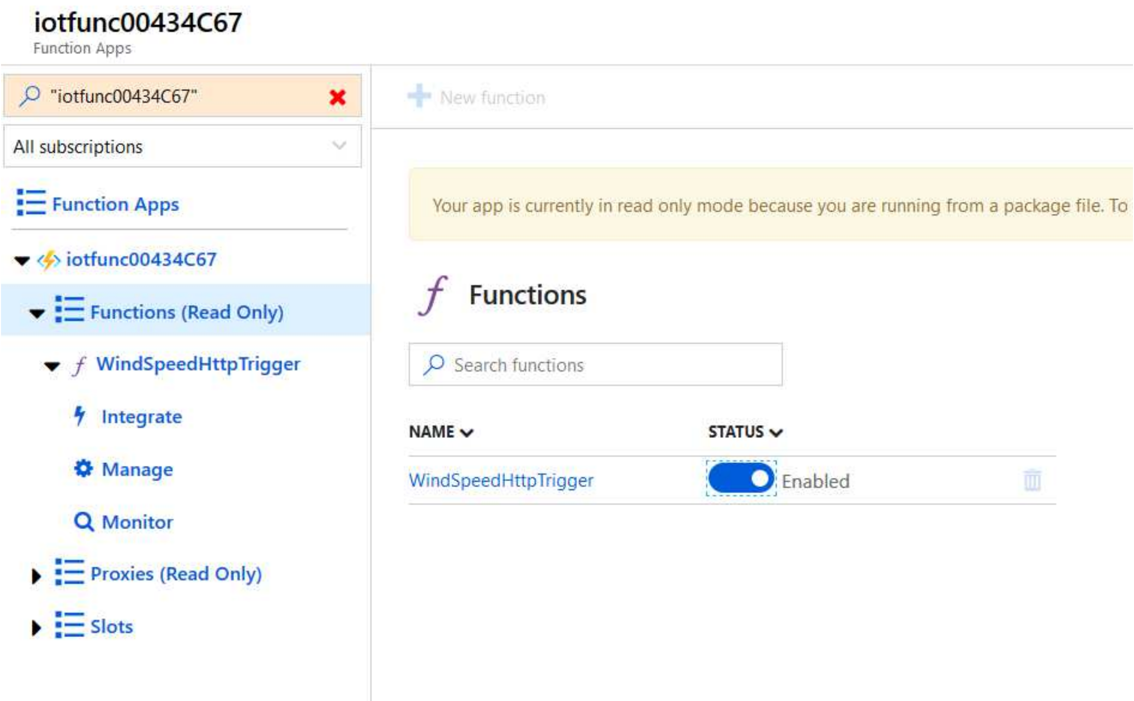


Resources

IoTCapstoneRG

Refresh

iothub00434C67	IoT Hub	Southeast Asia
iotasa00434C67	Stream Analytics job	Southeast Asia
iotfunc00434C67	App Service	Southeast Asia
iotstore00434c67	Storage account	Southeast Asia
SoutheastAsiaPlan	App Service plan	Southeast Asia



iotfunc00434C67

Function Apps

"iotfunc00434C67"

All subscriptions

Your app is currently in read only mode because you are running from a package file. To

Functions

Search functions

NAME	STATUS
WindSpeedHttpTrigger	Enabled

----

41. In the Search the Marketplace field, enter Function App and select Function App.

42. Under OS, select Windows.

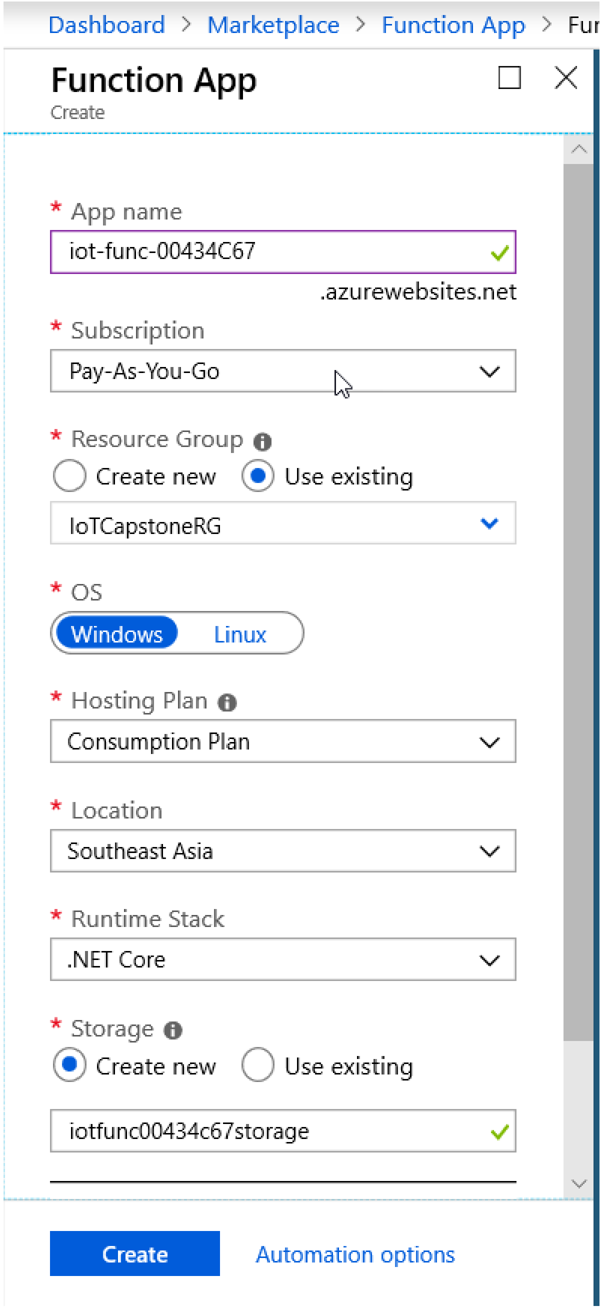
43. Under Hosting Plan, choose Consumption Plan.

44. Under Location, choose a location close to you.

45. Under Runtime Stack, choose .NET.

46. Under Storage, create New.

47. Under Application Insights, select Off.



Dashboard > Marketplace > Function App > Fur

Function App

Create

* App name

iot-func-00434C67 ✓

.azurewebsites.net

* Subscription

Pay-As-You-Go

* Resource Group

○ Create new ● Use existing

IoTCapstoneRG

* OS

Windows | Linux

* Hosting Plan

Consumption Plan

* Location

Southeast Asia

* Runtime Stack

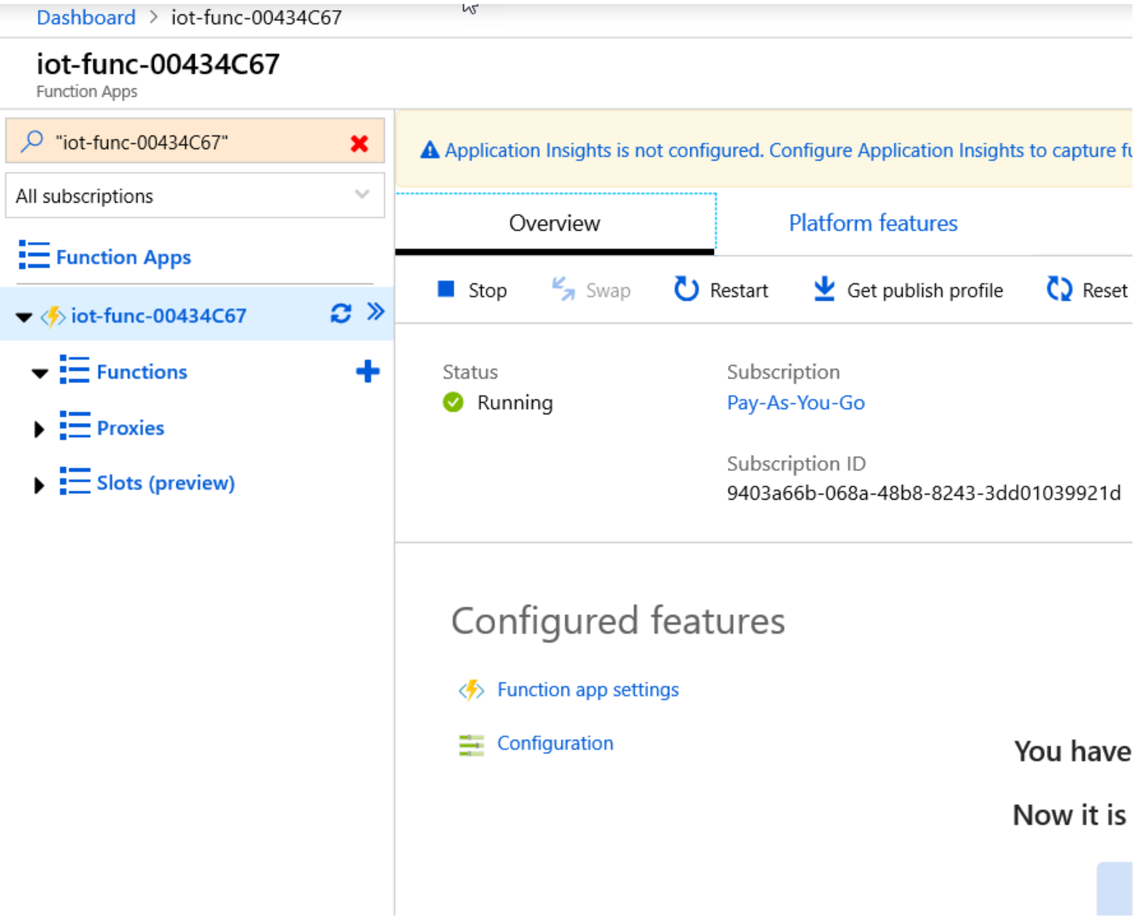
.NET Core

* Storage

● Create new ○ Use existing

iotfunc00434c67storage ✓

48. After function deployment it look something like this:



Dashboard > iot-func-00434C67

iot-func-00434C67
Function Apps

"iot-func-00434C67"

All subscriptions

Function Apps

iot-func-00434C67

Functions

Proxies

Slots (preview)

Application Insights is not configured. Configure Application Insights to capture fu

Overview | Platform features

Stop | Swap | Restart | Get publish profile | Reset

Status
Running

Subscription
Pay-As-You-Go

Subscription ID
9403a66b-068a-48b8-8243-3dd01039921d

Configured features

Function app settings

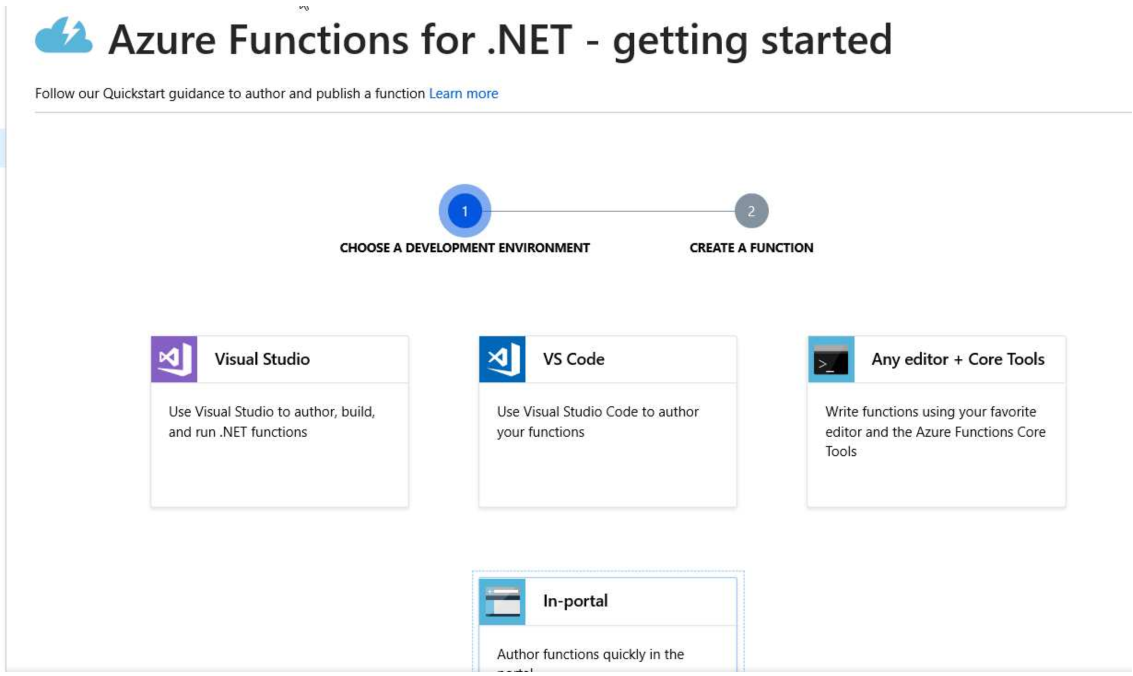
Configuration

You have

Now it is

49. On the Function Apps blade, in the left hand nav area, you will see the Function App we just created listed in a tree view. To the right of the Functions node, click + to add a new function.

50. Choose In-Portal (**Confirm Not Working**)



Azure Functions for .NET - getting started

Follow our Quickstart guidance to author and publish a function Learn more

1 CHOOSE A DEVELOPMENT ENVIRONMENT

2 CREATE A FUNCTION

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

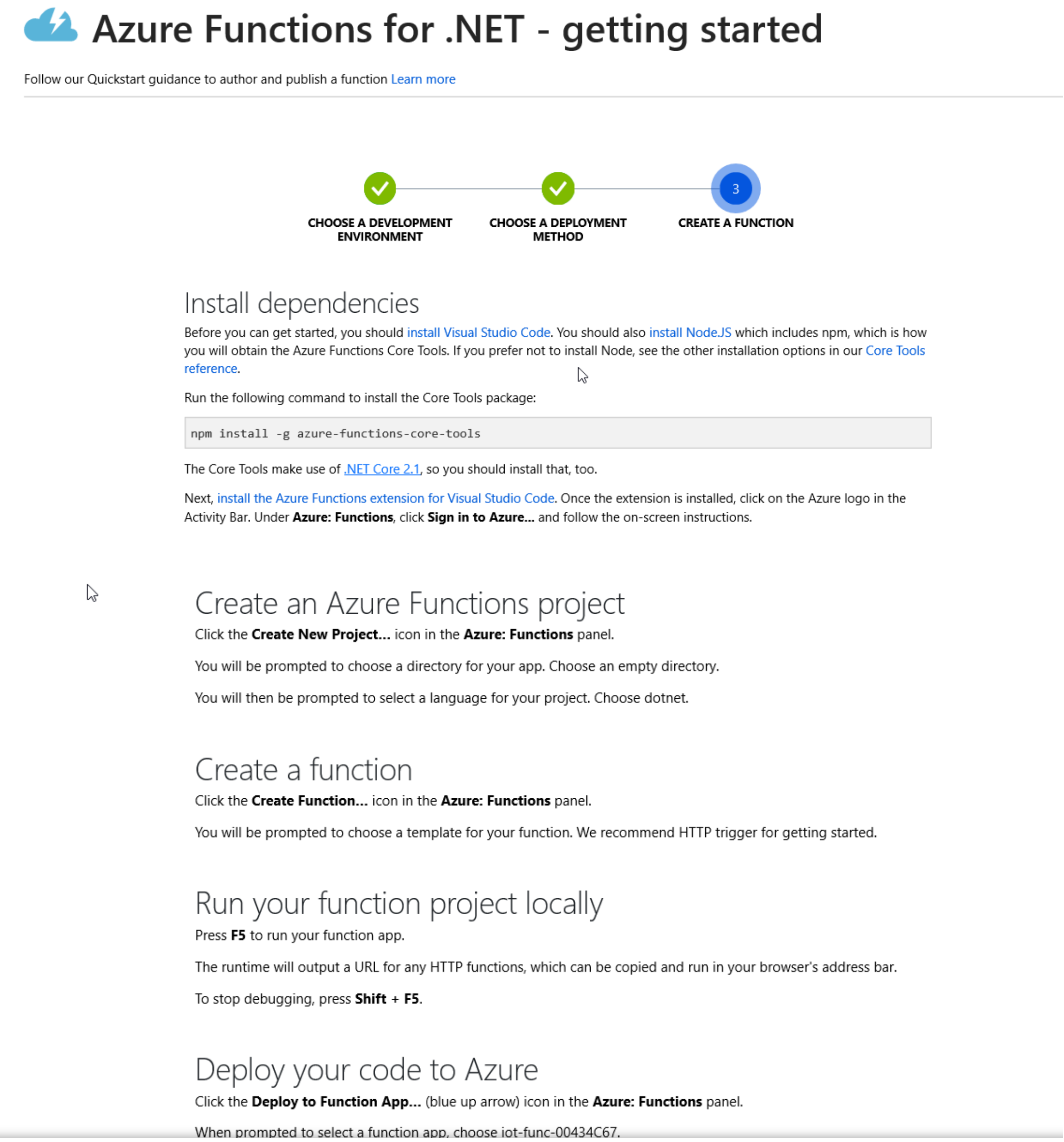
VS Code
Use Visual Studio Code to author your functions

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

In-portal
Author functions quickly in the portal

51. From the template list, choose HTTP trigger.

51a: Use VS Code to deploy the function.



Azure Functions for .NET - getting started

Follow our Quickstart guidance to author and publish a function Learn more

Install dependencies

Before you can get started, you should install Visual Studio Code. You should also install Node.JS which includes npm, which is how you will obtain the Azure Functions Core Tools. If you prefer not to install Node, see the other installation options in our Core Tools reference.

Run the following command to install the Core Tools package:

```
npm install -g azure-functions-core-tools
```

The Core Tools make use of .NET Core 2.1, so you should install that, too.

Next, install the Azure Functions extension for Visual Studio Code. Once the extension is installed, click on the Azure logo in the Activity Bar. Under **Azure: Functions**, click **Sign in to Azure...** and follow the on-screen instructions.

Create an Azure Functions project

Click the **Create New Project...** icon in the **Azure: Functions** panel.

You will be prompted to choose a directory for your app. Choose an empty directory.

You will then be prompted to select a language for your project. Choose dotnet.

Create a function

Click the **Create Function...** icon in the **Azure: Functions** panel.

You will be prompted to choose a template for your function. We recommend HTTP trigger for getting started.

Run your function project locally

Press **F5** to run your function app.

The runtime will output a URL for any HTTP functions, which can be copied and run in your browser's address bar.

To stop debugging, press **Shift** + **F5**.

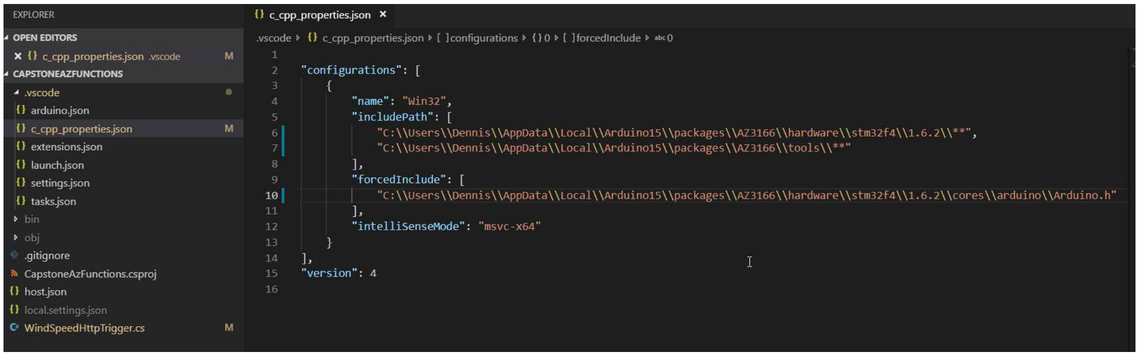
Deploy your code to Azure

Click the **Deploy to Function App...** (blue up arrow) icon in the **Azure: Functions** panel.

When prompted to select a function app, choose iot-func-00434C67.

Open the function using VS Code

Update **c_cpp_properties.json** with your include files, with your appropriate paths used.



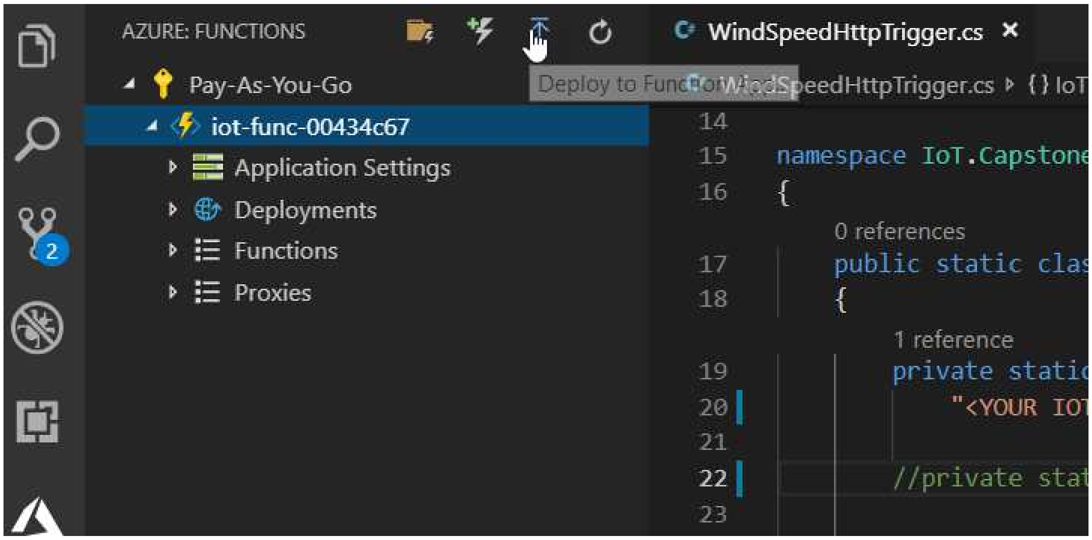
```

"configurations": [
    {
        "name": "Win32",
        "includePath": [
            "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\hardware\\stm32f4\\1.6.2\\**",
            "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\tools\\**"
        ],
        "forcedInclude": [
            "C:\\Users\\Dennis\\AppData\\Local\\Arduino15\\packages\\AZ3166\\hardware\\stm32f4\\1.6.2\\cores\\arduino\\Arduino.h"
        ],
        "intelliSenseMode": "msvc-x64"
    }
],
"version": 4

```

Comment out: "**using System.Text;**"

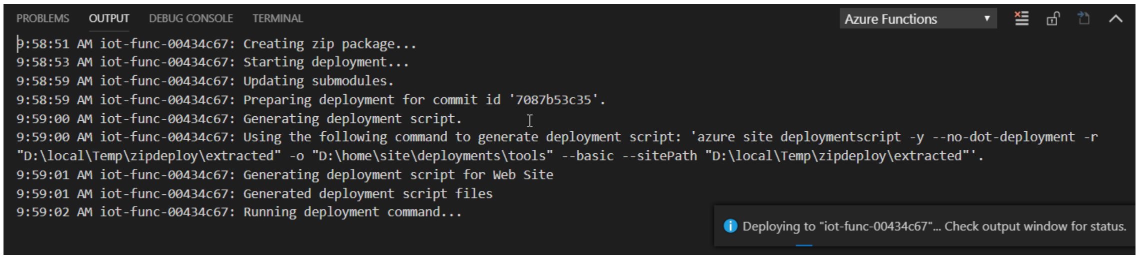
Comment out: "**private static TransportType transportType = TransportType.Amqp**;"

(Not needed, and besides, the MXChip uses MQTT, not Amqp, so not sure what this line of code is doing here. (not needed)

Make sure Azure Function extension is installed and select **Deploy to Function App.**



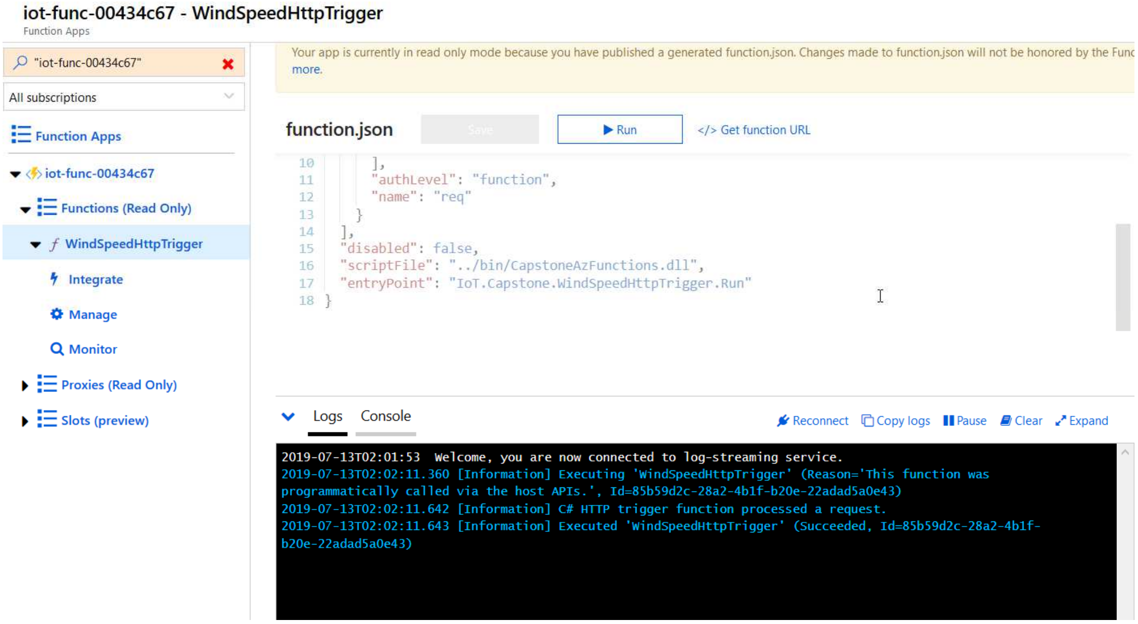
```
namespace IoT.Capstone
{
    0 references
    public static clas
    {
        1 reference
        private static
            "<YOUR IOT

        //private stat
```



```
9:58:51 AM iot-func-00434c67: Creating zip package...
9:58:53 AM iot-func-00434c67: Starting deployment...
9:58:59 AM iot-func-00434c67: Updating submodules.
9:58:59 AM iot-func-00434c67: Preparing deployment for commit id '7087b53c35'.
9:59:00 AM iot-func-00434c67: Generating deployment script.
9:59:00 AM iot-func-00434c67: Using the following command to generate deployment script: 'azure site deploymentscript -y --no-dot-deployment -r
"D:\local\Temp\zipdeploy\extracted" -o "D:\home\site\deployments\tools" --basic --sitePath "D:\local\Temp\zipdeploy\extracted"'.
9:59:01 AM iot-func-00434c67: Generating deployment script for Web Site
9:59:01 AM iot-func-00434c67: Generated deployment script files
9:59:02 AM iot-func-00434c67: Running deployment command...
```

Deploying to "iot-func-00434c67"... Check output window for status.

On Azure Portal, it is read-only mode



Your app is currently in read only mode because you have published a generated function.json. Changes made to function.json will not be honored by the Func
more.

```
            ],
            "authLevel": "function",
            "name": "req"
        }
    ],
    "disabled": false,
    "scriptFile": "../bin/CapstoneAzFunctions.dll",
    "entryPoint": "IoT.Capstone.WindSpeedHttpTrigger.Run"
}
```

```
2019-07-13T02:01:53  Welcome, you are now connected to log-streaming service.
2019-07-13T02:02:11.360 [Information] Executing 'WindSpeedHttpTrigger' (Reason='This function was
programmatically called via the host APIs.', Id=85b59d2c-28a2-4b1f-b20e-22adad5a0e43)
2019-07-13T02:02:11.642 [Information] C# HTTP trigger function processed a request.
2019-07-13T02:02:11.643 [Information] Executed 'WindSpeedHttpTrigger' (Succeeded, Id=85b59d2c-28a2-4b1f-
b20e-22adad5a0e43)
```

To get the Function-Device Twins-Stream Analytics Query-MXChip to all handshake correctly, you need to:

. have correct data telemetry names (expected by the Function) and correct JSON format

The ASA query is where you confirm names are all correct prior to sending to output sources.

(“ConnectionDeviceId" must be renamed to "deviceId").

. data sent to Function via ASA must not be nested (like it is when sent from MXChip)

. When testing with code snippets in the "Request body" in the Test tab, you must include

square brackets. The ASA query, of course, will not include square brackets.

. Test your output for the Function by first sending to either a Blob or Data Lake store and

Inspect the telemetry JSON data to confirm it meets the naming and format requirements.

. Using the VS Code approach means you will not be able to edit the

WindSpeedHttpTrigger.cs in the Portal. Each change (but there really shouldn't be any

after doing the 3 changes above) must be done back in VS Code, with a new deployment

to Azure, using the Command Palette.

. While testing the code snippet inputs to the Function, you might have to toggle in and out

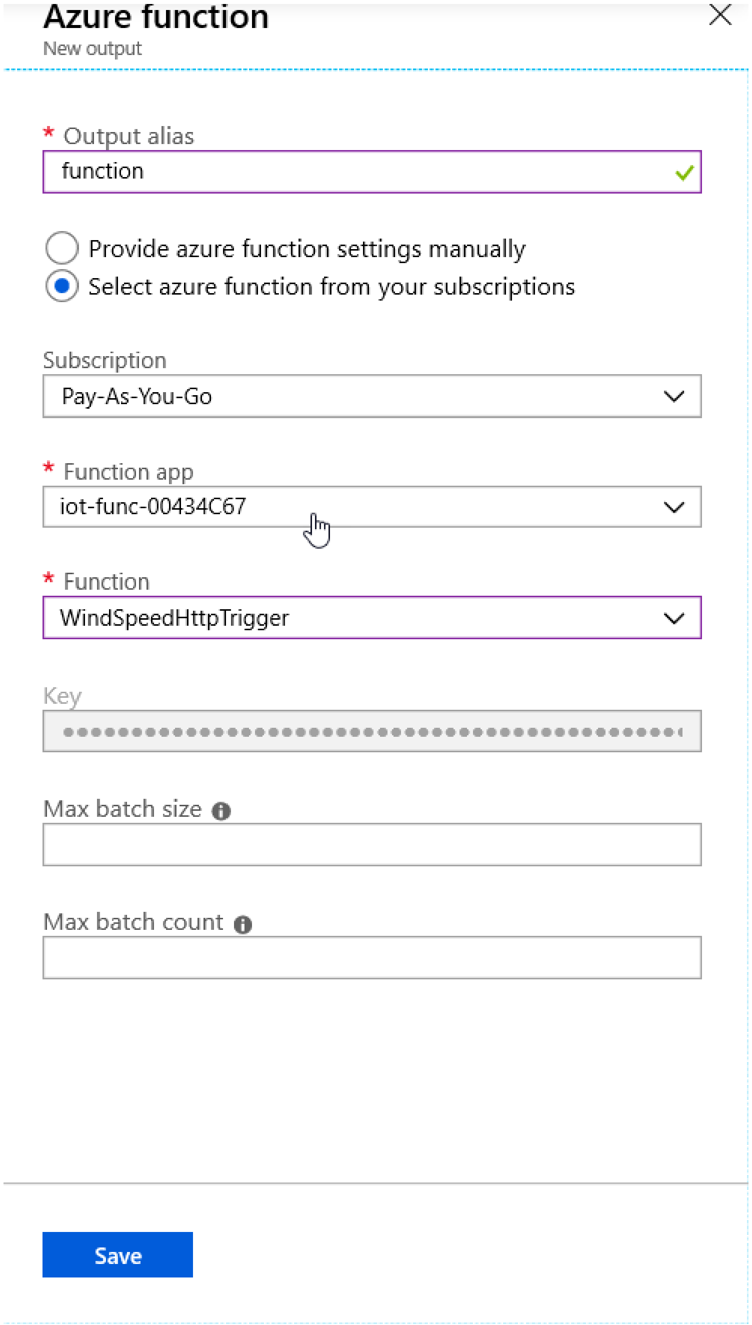
of the Function from time to time – it seems to suspend sometimes.

52. Create a Stream Analytics Job Output

53. In the Stream Analytics Job blade left hand nav area, under Job topology, click Outputs.

54. At the top of the Outputs pane, click Add and select Azure Function

55. Leave Max batch size and Max batch count empty so that the default values are used.



Azure function

New output

* Output alias

function ✓

○ Provide azure function settings manually

● Select azure function from your subscriptions

Subscription

Pay-As-You-Go

* Function app

iot-func-00434C67

* Function

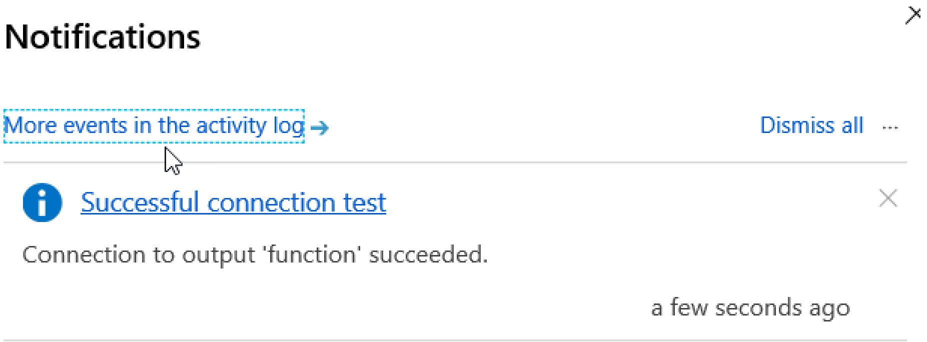
WindSpeedHttpTrigger

Key

●●●

Max batch size

Max batch count



Notifications

More events in the activity log →

Dismiss all ···

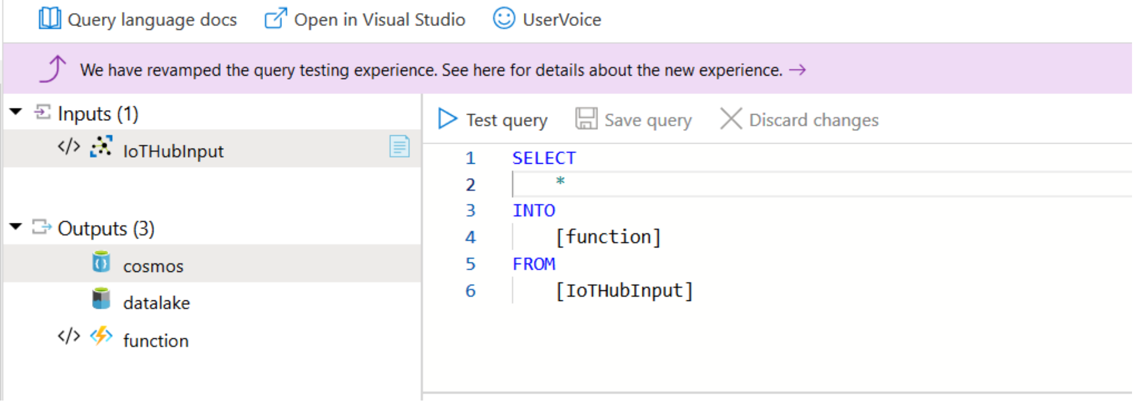
Successful connection test

Connection to output 'function' succeeded.

a few seconds ago

56. In the Stream Analytics Job blade left hand nav area, under Job topology, click Query.

57. In the Query pane, replace the default query with the following:



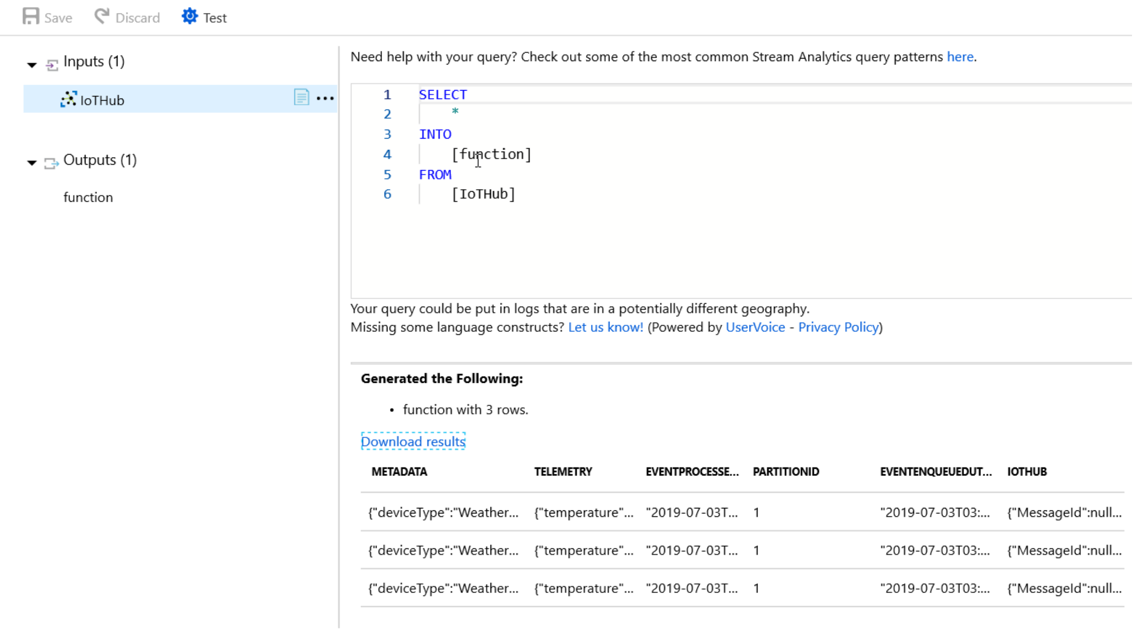
We have revamped the query testing experience. See here for details about the new experience. →

```
SELECT
    *
INTO
    [function]
FROM
    [IoTHubInput]
```

58. To capture test data, in the Query pane, under Inputs click the ellipsis (...) to the right of IoTHubInput and select Sample data from input.

59. The sample data displays. Accept the default values and click OK.

60. Once the sampling has completed, click Test in the toolbar.



Need help with your query? Check out some of the most common Stream Analytics query patterns here.

```
SELECT
    *
INTO
    [function]
FROM
    [IoTHub]
```

Your query could be put in logs that are in a potentially different geography.
Missing some language constructs? Let us know! (Powered by UserVoice - Privacy Policy)

Generated the Following:

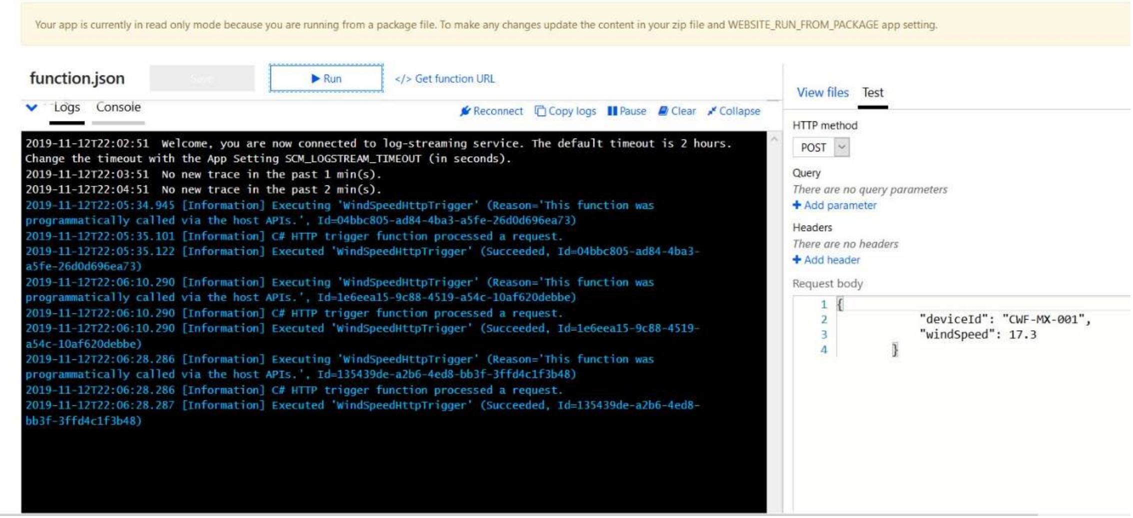
- function with 3 rows.

METADATA	TELEMETRY	EVENTPROCESSE...	PARTITIONID	EVENTENQUEUEDUT...	IOTHUB
{"deviceType":"Weather...	{"temperature"...	"2019-07-03T...	1	"2019-07-03T03:...	{"MessageId":null...
{"deviceType":"Weather...	{"temperature"...	"2019-07-03T...	1	"2019-07-03T03:...	{"MessageId":null...
{"deviceType":"Weather...	{"temperature"...	"2019-07-03T...	1	"2019-07-03T03:...	{"MessageId":null...

----------------------Nov Update--

We need to do a query test to make sure function works.

Under Function, go to function.json and click Run with the simple test POST command: It will trigger the azure function if successful.



Your app is currently in read only mode because you are running from a package file. To make any changes update the content in your zip file and WEBSITE_RUN_FROM_PACKAGE app setting.

```
2019-11-12T22:02:51  Welcome, you are now connected to log-streaming service. The default timeout is 2 hours.
Change the timeout with the App Setting SCM_LOGSTREAM_TIMEOUT (in seconds).
2019-11-12T22:03:51  No new trace in the past 1 min(s).
2019-11-12T22:04:51  No new trace in the past 2 min(s).
2019-11-12T22:05:34.945 [Information] Executing 'WindSpeedHttpTrigger' (Reason='This function was
programmatically called via the host APIs.', Id=04bbc805-ad84-4ba3-a5fe-26d0d696ea73)
2019-11-12T22:05:35.101 [Information] C# HTTP trigger function processed a request.
2019-11-12T22:05:35.122 [Information] Executed 'WindSpeedHttpTrigger' (Succeeded, Id=04bbc805-ad84-4ba3-
a5fe-26d0d696ea73)
2019-11-12T22:06:10.290 [Information] Executing 'WindSpeedHttpTrigger' (Reason='This function was
programmatically called via the host APIs.', Id=1e6eea15-9c88-4519-a54c-10af620debbe)
2019-11-12T22:06:10.290 [Information] C# HTTP trigger function processed a request.
2019-11-12T22:06:10.290 [Information] Executed 'WindSpeedHttpTrigger' (Succeeded, Id=1e6eea15-9c88-4519-
a54c-10af620debbe)
2019-11-12T22:06:28.286 [Information] Executing 'WindSpeedHttpTrigger' (Reason='This function was
programmatically called via the host APIs.', Id=135439de-a2b6-4ed8-bb3f-3ffd4c1f3b48)
2019-11-12T22:06:28.286 [Information] C# HTTP trigger function processed a request.
2019-11-12T22:06:28.287 [Information] Executed 'WindSpeedHttpTrigger' (Succeeded, Id=135439de-a2b6-4ed8-
bb3f-3ffd4c1f3b48)
```

HTTP method: POST

Query
There are no query parameters
+ Add parameter

Headers
There are no headers
+ Add header

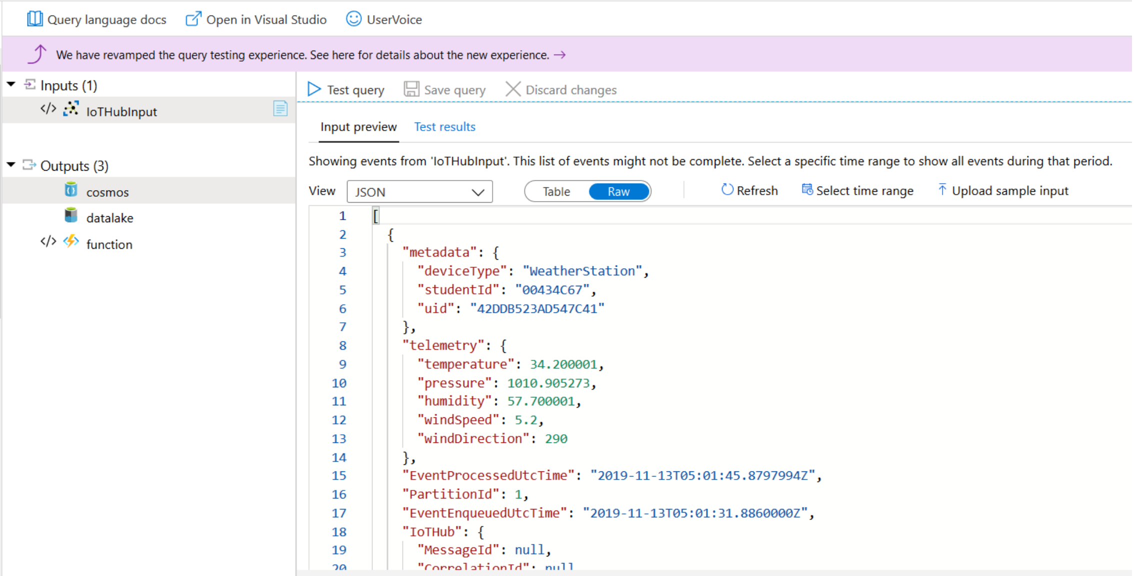
Request body

```
{
            "deviceId": "CWF-MX-001",
            "windSpeed": 17.3
        }
```

Under Query In ASA: in JSON format, make sure telemetry is sent to IoTHubInput

- The Azure Function expects to receive JSON data in the following format:

```
{
            "deviceId": "CWF-MX-001",
            "windSpeed": 17.3
        }
```



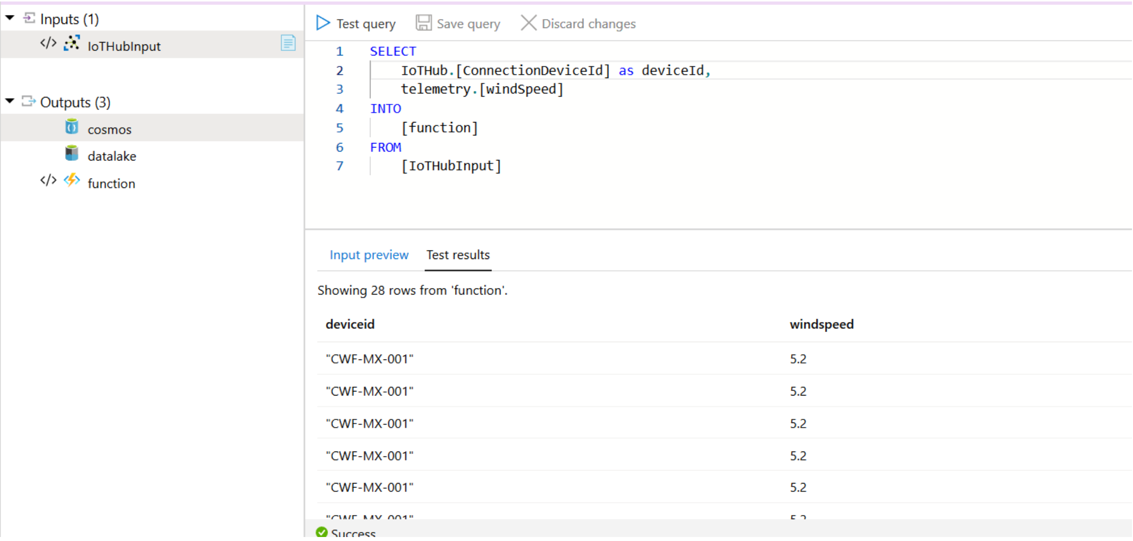
We have revamped the query testing experience. See here for details about the new experience. →

Showing events from 'IoTHubInput'. This list of events might not be complete. Select a specific time range to show all events during that period.

View: JSON

```
[
  {
    "metadata": {
      "deviceType": "WeatherStation",
      "studentId": "00434C67",
      "uid": "42DDB523AD547C41"
    },
    "telemetry": {
      "temperature": 34.200001,
      "pressure": 1010.905273,
      "humidity": 57.700001,
      "windSpeed": 5.2,
      "windDirection": 290
    },
    "EventProcessedUtcTime": "2019-11-13T05:01:45.8797994Z",
    "PartitionId": 1,
    "EventEnqueuedUtcTime": "2019-11-13T05:01:31.8860000Z",
    "IoTHub": {
      "MessageId": null,
      "CorrelationId": null
```

The query language for function will be this:



```
SELECT
    IoTHub.[ConnectionDeviceId] as deviceId,
    telemetry.[windSpeed]
INTO
    [function]
FROM
    [IoTHubInput]
```

Showing 28 rows from 'function'.

deviceid	windspeed
"CWF-MX-001"	5.2
"CWF-MX-001"	5.2
"CWF-MX-001"	5.2
"CWF-MX-001"	5.2
"CWF-MX-001"	5.2
"CWF-MX-001"	5.2

Success

---

61. Send data to Data Lake Storage Gen 1. Review the following materials:

Course: DEV326x IoT Data Analytics and Storage

Module Getting Started with Data Lake Storage and Analytics

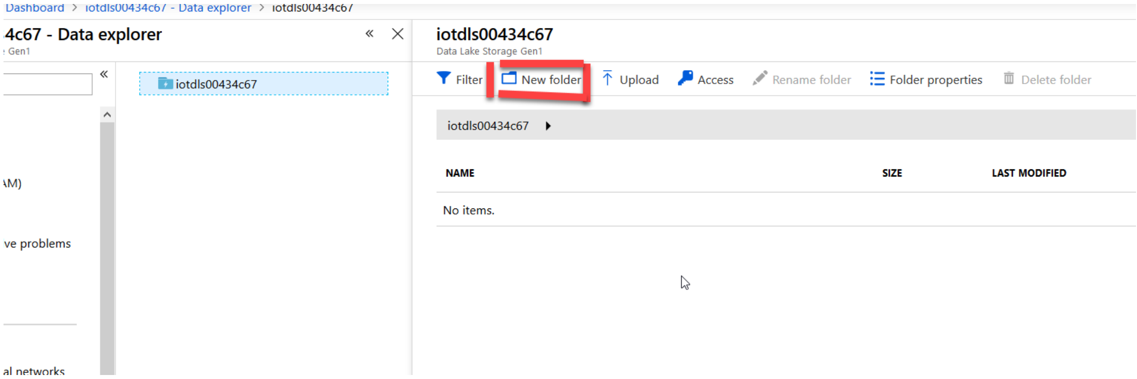
Lab: IoT Analytics and Cold Storage

Topic: Set up a Cold Storage Repository with Azure Data Lake Storage

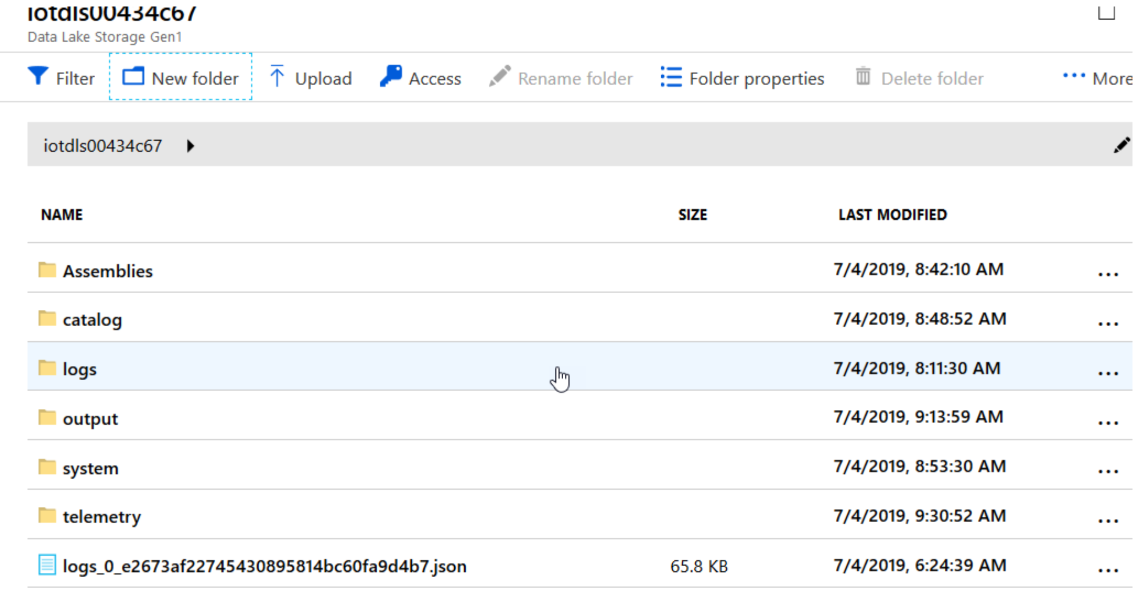
Tip: The default data format for JSON data from Azure Streaming Analytics is LineSeparated - ensure you update the format to use Array.

Tip: Use the following Path prefix pattern for the Data Lake Gen 1 output in stream analytics: telemetry/{date}

61a. Create **telemetry** folder to store Streaming data coming from your device through IoTHub using Stream Analytics Job



NAME	SIZE	LAST MODIFIED
No items.		

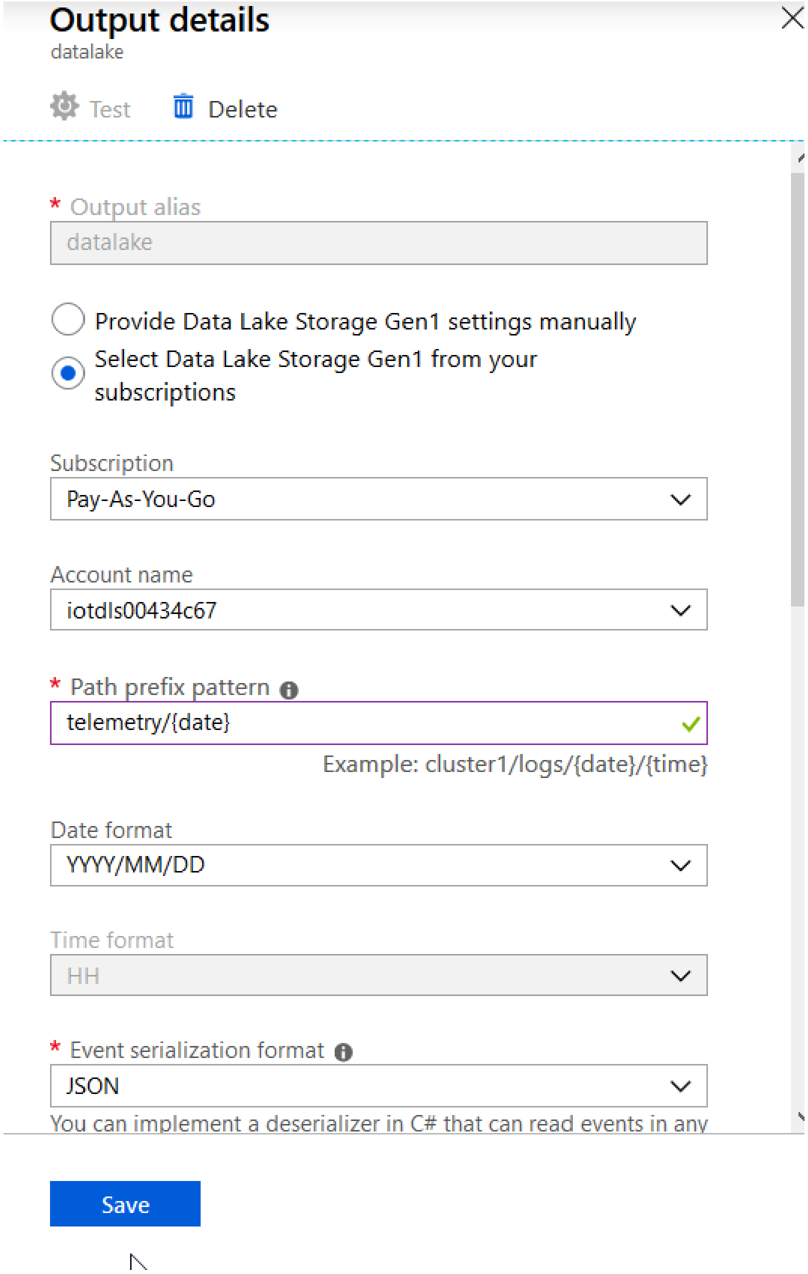


NAME	SIZE	LAST MODIFIED
Assemblies		7/4/2019, 8:42:10 AM
catalog		7/4/2019, 8:48:52 AM
logs		7/4/2019, 8:11:30 AM
output		7/4/2019, 9:13:59 AM
system		7/4/2019, 8:53:30 AM
telemetry		7/4/2019, 9:30:52 AM
logs_0_e2673af22745430895814bc60fa9d4b7.json	65.8 KB	7/4/2019, 6:24:39 AM

62. Once the deployment has complete, navigate to the Stream Analytics job that you created.

63. On the Overview blade of your Stream Analytics job, click **Outputs**

64. In the upper left corner of the Outputs blade, click + Add., click **Data Lake Store**



Output details

datalake

* Output alias

datalake

○ Provide Data Lake Storage Gen1 settings manually

● Select Data Lake Storage Gen1 from your subscriptions

Subscription

Pay-As-You-Go

Account name

iotdls00434c67

* Path prefix pattern

telemetry/{date} ✓

Example: cluster1/logs/{date}/{time}

Date format

YYYY/MM/DD

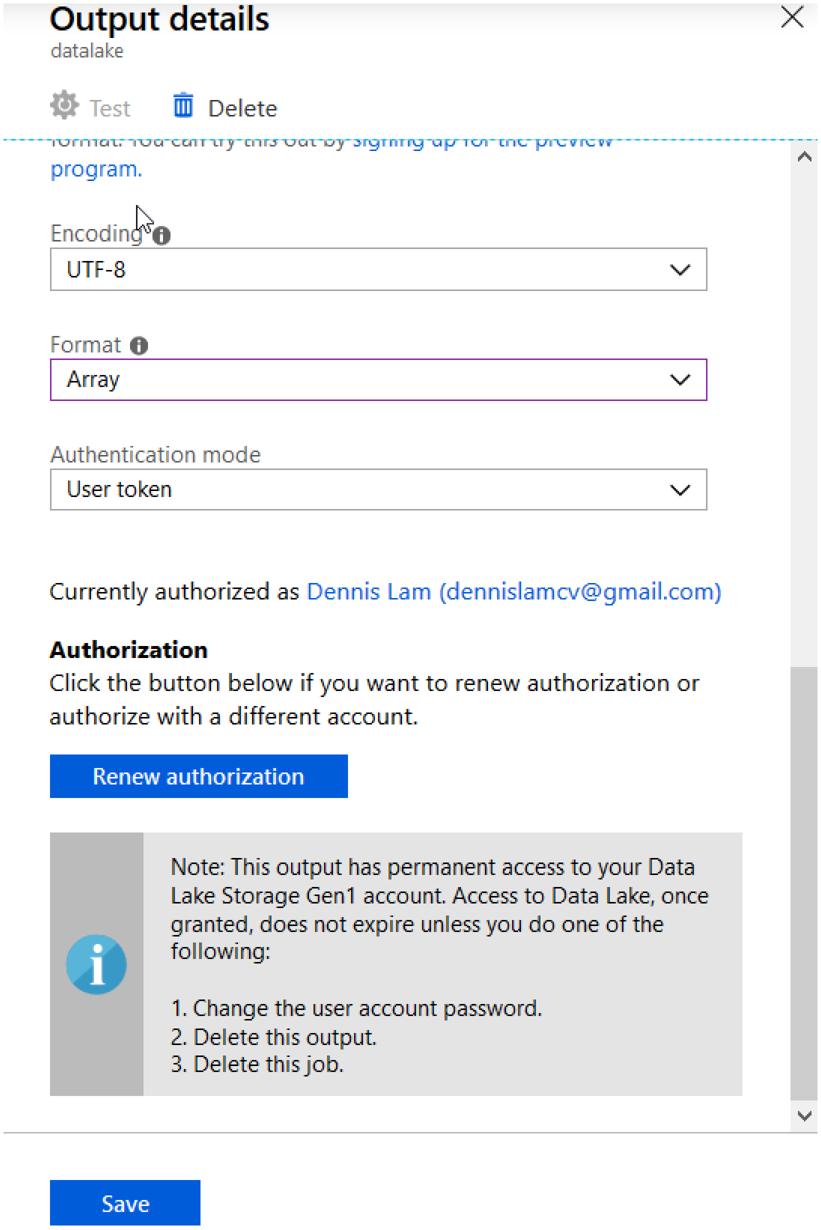
Time format

HH

* Event serialization format

JSON

You can implement a deserializer in C# that can read events in any



Output details

datalake

format. You can try this out by signing up for the preview program.

Encoding: UTF-8

Format: Array

Authentication mode: User token

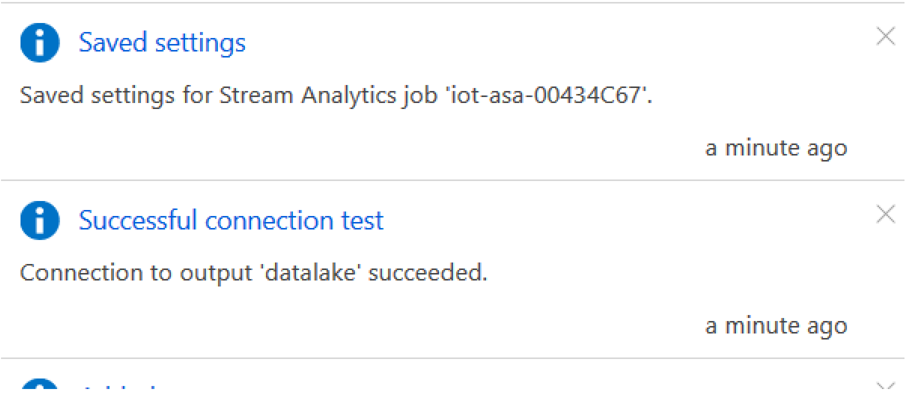
Currently authorized as Dennis Lam (dennislamcv@gmail.com)

Authorization

Click the button below if you want to renew authorization or authorize with a different account.

Note: This output has permanent access to your Data Lake Storage Gen1 account. Access to Data Lake, once granted, does not expire unless you do one of the following:

1. Change the user account password.
2. Delete this output.
3. Delete this job.



Saved settings

Saved settings for Stream Analytics job 'iot-asa-00434C67'.

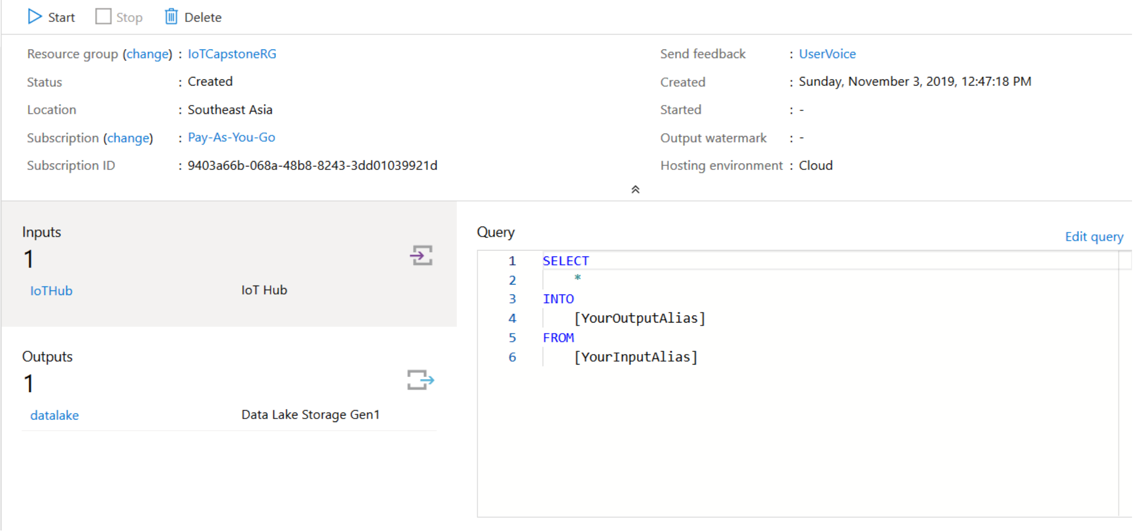
a minute ago

Successful connection test

Connection to output 'datalake' succeeded.

a minute ago

Connected Data Lake Store:



Resource group (change) : IoTCapstoneRG
Status : Created
Location : Southeast Asia
Subscription (change) : Pay-As-You-Go
Subscription ID : 9403a66b-068a-48b8-8243-3dd01039921d

Send feedback : UserVoice
Created : Sunday, November 3, 2019, 12:47:18 PM
Started : -
Output watermark : -
Hosting environment : Cloud

Inputs
1

IoTHub	IoT Hub

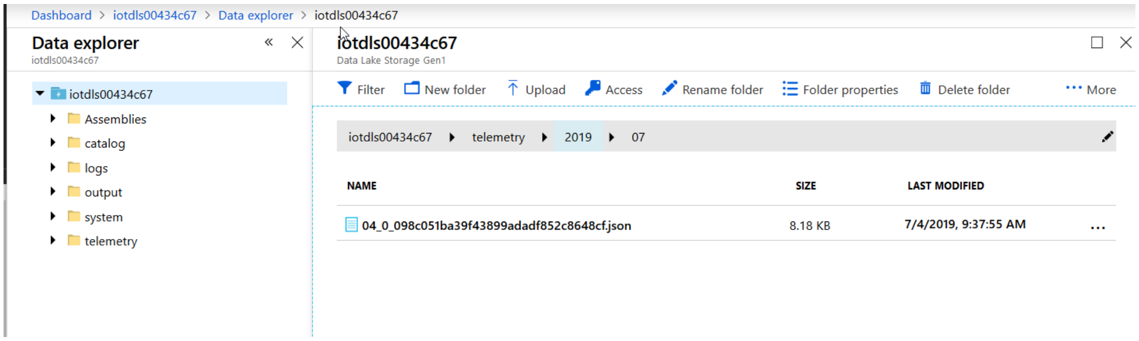
Outputs
1

datalake	Data Lake Storage Gen1

Query

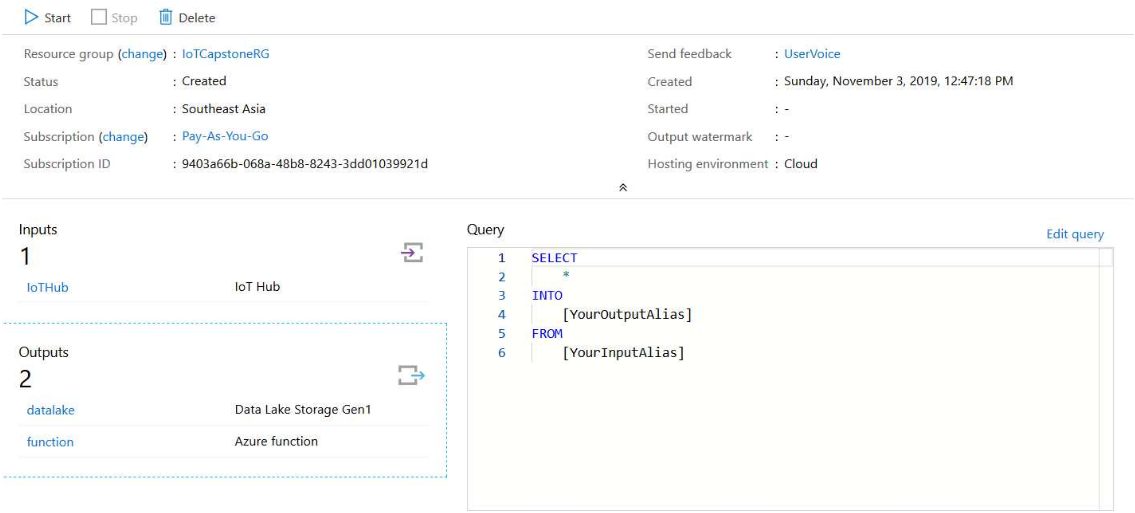
```
SELECT
    *
INTO
    [YourOutputAlias]
FROM
    [YourInputAlias]
```

The json file appears here:



NAME	SIZE	LAST MODIFIED
04_0_098c051ba39f43899adadf852c8648cf.json	8.18 KB	7/4/2019, 9:37:55 AM

Add Azure Function as Stream Analytics Output:



Resource group (change) : IoTCapstoneRG
Status : Created
Location : Southeast Asia
Subscription (change) : Pay-As-You-Go
Subscription ID : 9403a66b-068a-48b8-8243-3dd01039921d

Send feedback : UserVoice
Created : Sunday, November 3, 2019, 12:47:18 PM
Started : -
Output watermark : -
Hosting environment : Cloud

Inputs
1

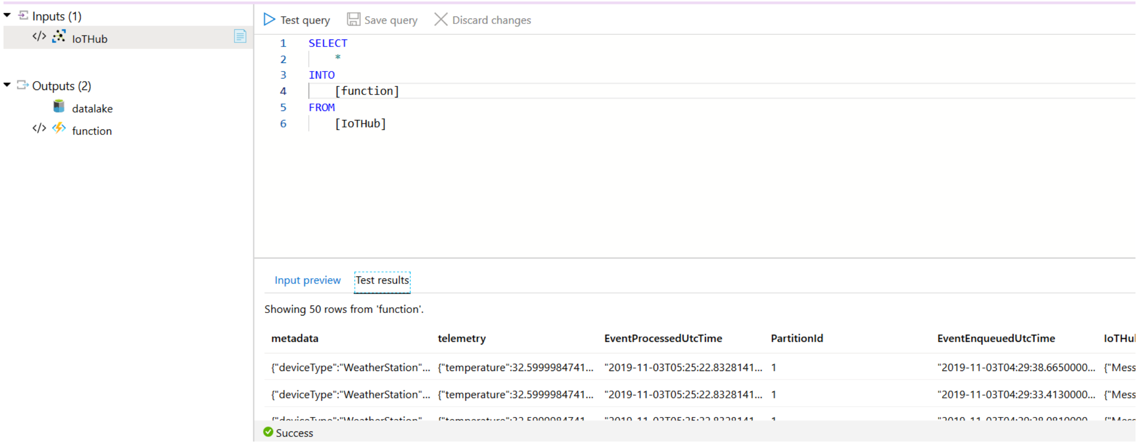
IoTHub	IoT Hub

Outputs
2

datalake	Data Lake Storage Gen1
function	Azure function

Query

```
SELECT
    *
INTO
    [YourOutputAlias]
FROM
    [YourInputAlias]
```



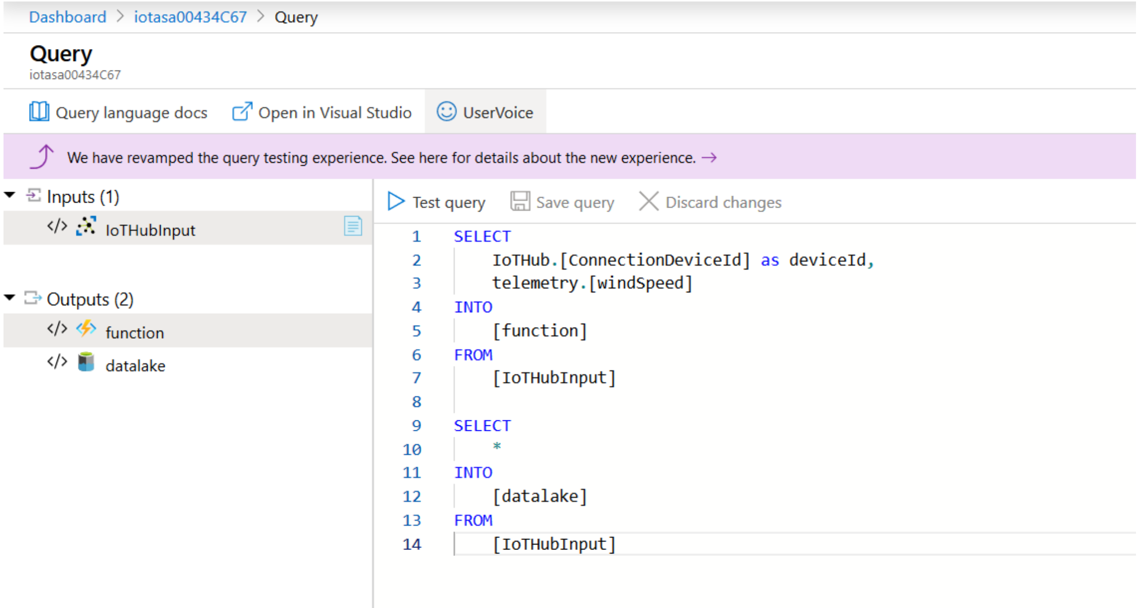
```
SELECT
    *
INTO
    [function]
FROM
    [IoTHub]
```

Showing 50 rows from 'function'.

metadata	telemetry	EventProcessedUtcTime	PartitionId	EventEnqueuedUtcTime	IoTHu
{"deviceType":"WeatherStation"...	{"temperature":32.5999984741...	"2019-11-03T05:25:22.8328141...	1	"2019-11-03T04:29:38.6650000...	{"Mess
{"deviceType":"WeatherStation"...	{"temperature":32.5999984741...	"2019-11-03T05:25:22.8328141...	1	"2019-11-03T04:29:33.4130000...	{"Mess
{"deviceType":"WeatherStation"...	{"temperature":32.5999984741...	"2019-11-03T05:25:22.8328141...	1	"2019-11-03T04:29:28.0810000...	{"Mess

Success

---------------------------**Nov Update**---



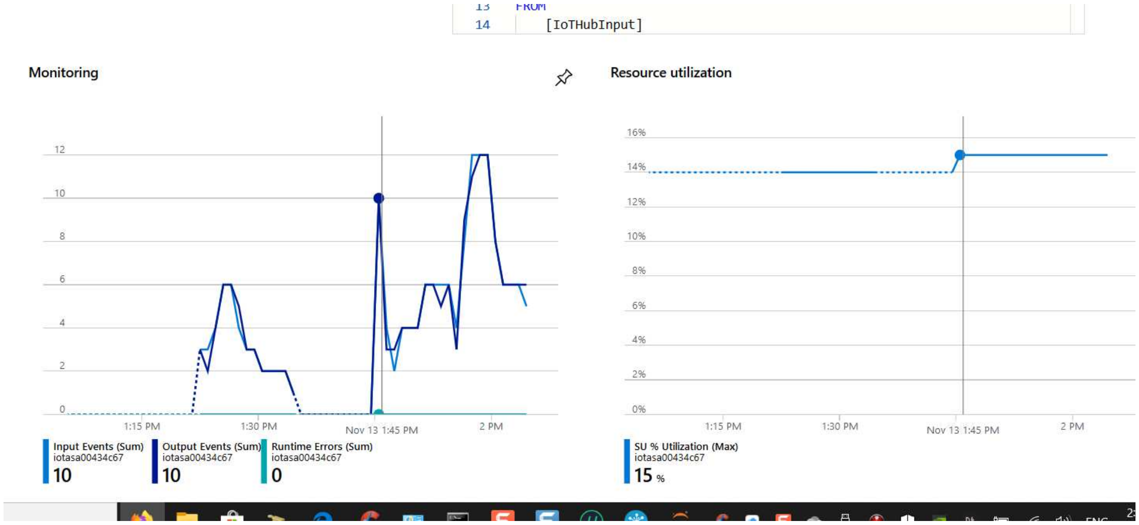
We have revamped the query testing experience. See here for details about the new experience. →

```
SELECT
    IoTHub.[ConnectionDeviceId] as deviceId,
    telemetry.[windSpeed]
INTO
    [function]
FROM
    [IoTHubInput]

SELECT
    *
INTO
    [datalake]
FROM
    [IoTHubInput]
```

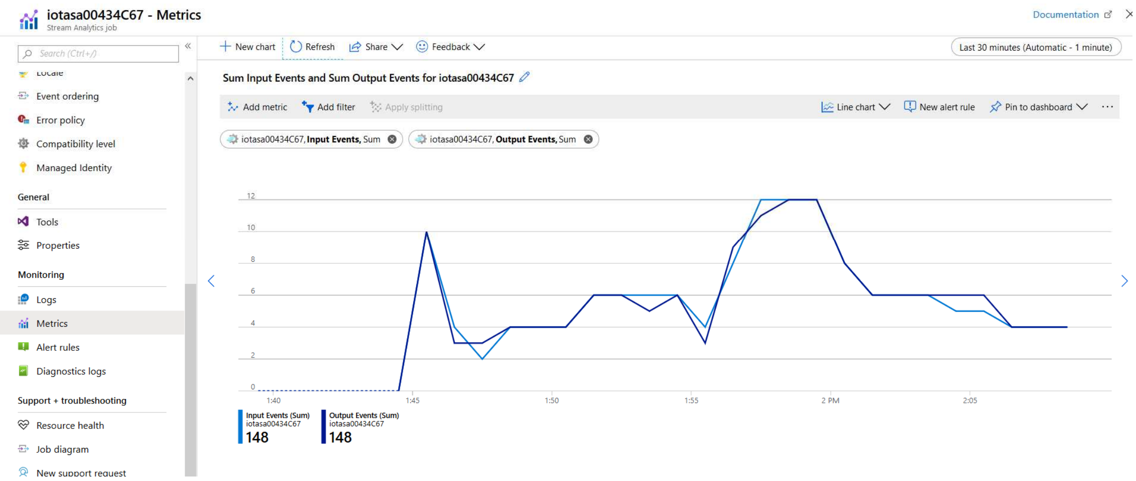
Need to combine 2 outputs.

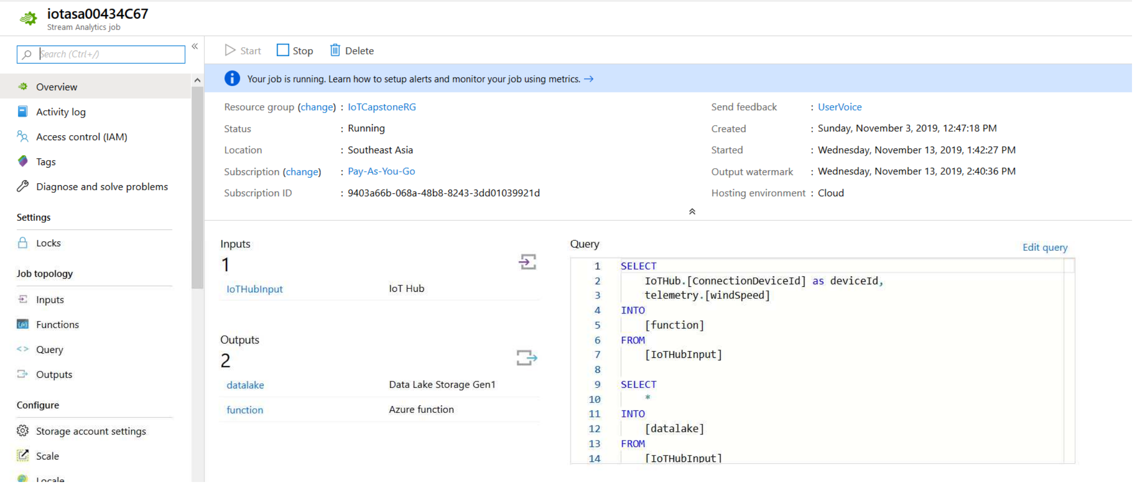
65. Leave it run at least 1 hour. During streaming press buttons A and B to check if Azure Function and RGB LED is working.

You can monitor the streaming in ASA main panel:



```
FROM
    [IoTHubInput]
```





Your job is running. Learn how to setup alerts and monitor your job using metrics. →

Resource group (change) : IoTCapstoneRG
Status : Running
Location : Southeast Asia
Subscription (change) : Pay-As-You-Go
Subscription ID : 9403a66b-068a-48b8-8243-3dd01039921d

Send feedback : UserVoice
Created : Sunday, November 3, 2019, 12:47:18 PM
Started : Wednesday, November 13, 2019, 1:42:27 PM
Output watermark : Wednesday, November 13, 2019, 2:40:36 PM
Hosting environment : Cloud

Inputs
1

IoTHubInput	IoT Hub

Outputs
2

datalake	Data Lake Storage Gen1
function	Azure function

Query

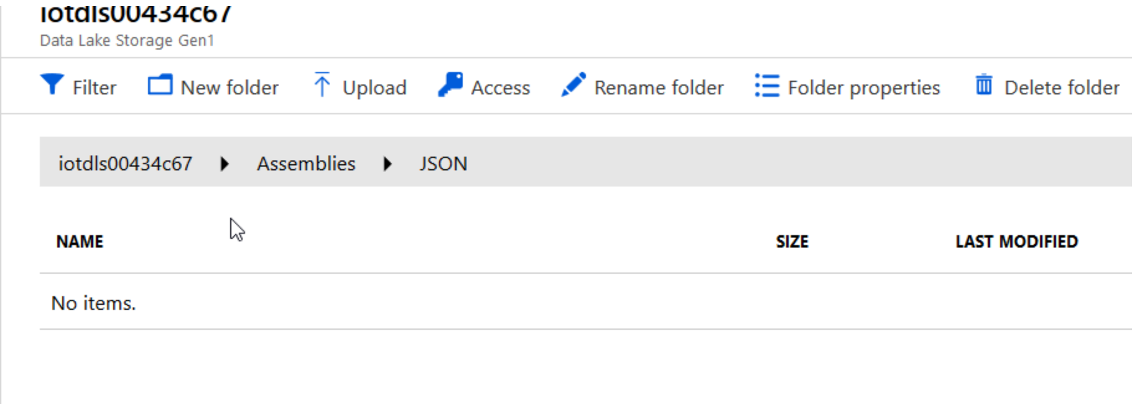
Edit query

```
SELECT
    IoTHub.[ConnectionDeviceId] as deviceId,
    telemetry.[windSpeed]
INTO
    [function]
FROM
    [IoTHubInput]

SELECT
    *
INTO
    [datalake]
FROM
    [IoTHubInput]
```

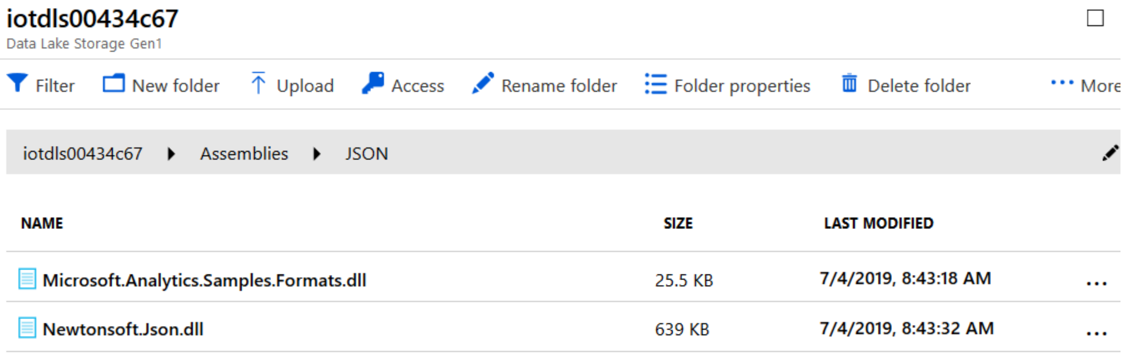
66. Create a Database in Azure Data Lake

Create a U-SQL database called IoTCapstoneDB and register the Newtonsoft.Json.dll and Microsoft.Analytics.Samples.Formats.dll assemblies in order to be able to query the JSON telemetry.

67. Upload your two DLLs from the bin folder above to your desired location in Azure Data Lake Store. You'll need to create a folder for them first. I used the following path: **\Assemblies\JSON.**

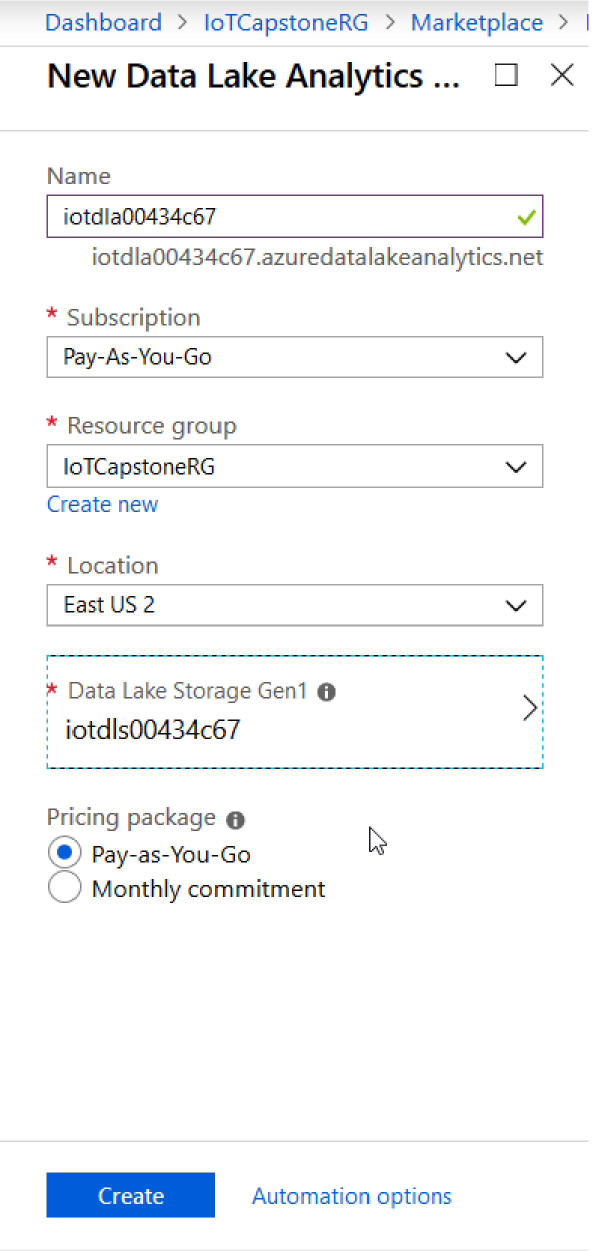


NAME	SIZE	LAST MODIFIED
No items.		



NAME	SIZE	LAST MODIFIED
Microsoft.Analytics.Samples.Formats.dll	25.5 KB	7/4/2019, 8:43:18 AM
Newtonsoft.Json.dll	639 KB	7/4/2019, 8:43:32 AM

68. Create Azure Data Lake Analytics (Take note same location as Data Lake Store.


Dashboard > IoTCapstoneRG > Marketplace >

New Data Lake Analytics ...

Name: iotdla00434c67 ✓
iotdla00434c67.azuredatalakeanalytics.net

* Subscription: Pay-As-You-Go

* Resource group: IoTCapstoneRG

Create new

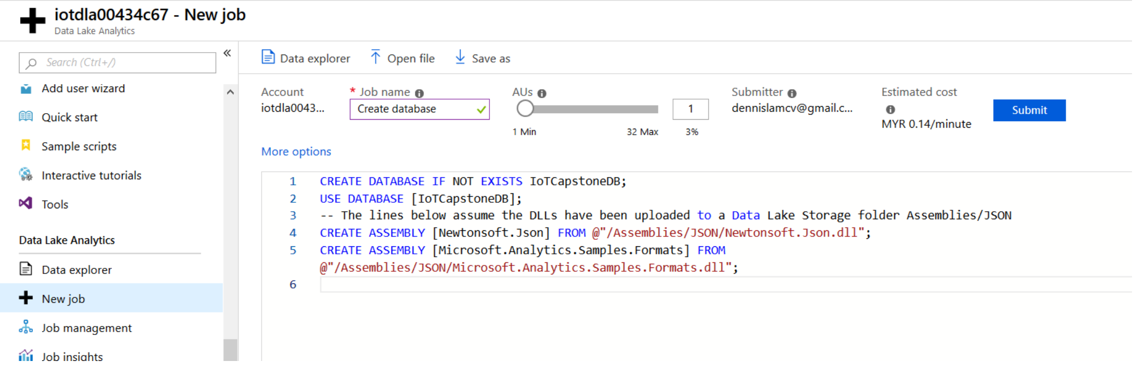
* Location: East US 2

* Data Lake Storage Gen1: iotdls00434c67

Pricing package
● Pay-as-You-Go
○ Monthly commitment

69. Create and run the following U-SQL job:

```
CREATE DATABASE IF NOT EXISTS IoTCapstoneDB;
USE DATABASE [IoTCapstoneDB];
// The lines below assume the DLLs have been uploaded to a Data Lake
Storage folder Assemblies/JSON
CREATE ASSEMBLY [Newtonsoft.Json] FROM
@"/Assemblies/JSON/Newtonsoft.Json.dll";
CREATE ASSEMBLY [Microsoft.Analytics.Samples.Formats] FROM
@"/Assemblies/JSON/Microsoft.Analytics.Samples.Formats.dll";
```



Account: iotdla0043...

* Job name: Create database

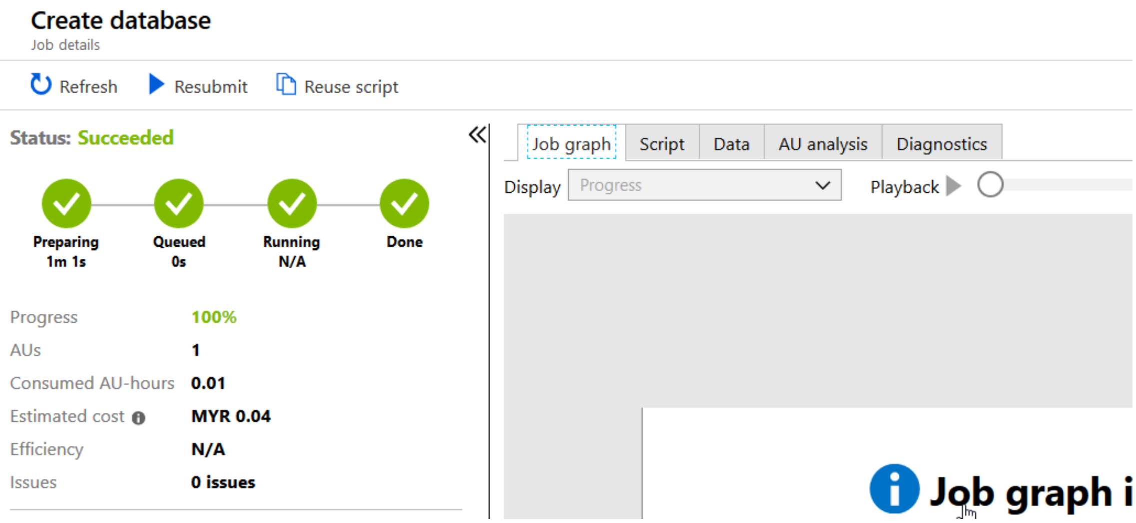
AUs: 1

Submitter: dennislamcv@gmail.c...

Estimated cost: MYR 0.14/minute

```
1   CREATE DATABASE IF NOT EXISTS IoTCapstoneDB;
2   USE DATABASE [IoTCapstoneDB];
3   -- The lines below assume the DLLs have been uploaded to a Data Lake Storage folder Assemblies/JSON
4   CREATE ASSEMBLY [Newtonsoft.Json] FROM @"/Assemblies/JSON/Newtonsoft.Json.dll";
5   CREATE ASSEMBLY [Microsoft.Analytics.Samples.Formats] FROM
    @"/Assemblies/JSON/Microsoft.Analytics.Samples.Formats.dll";
6
```

Note: Remove line 3 completely else error.

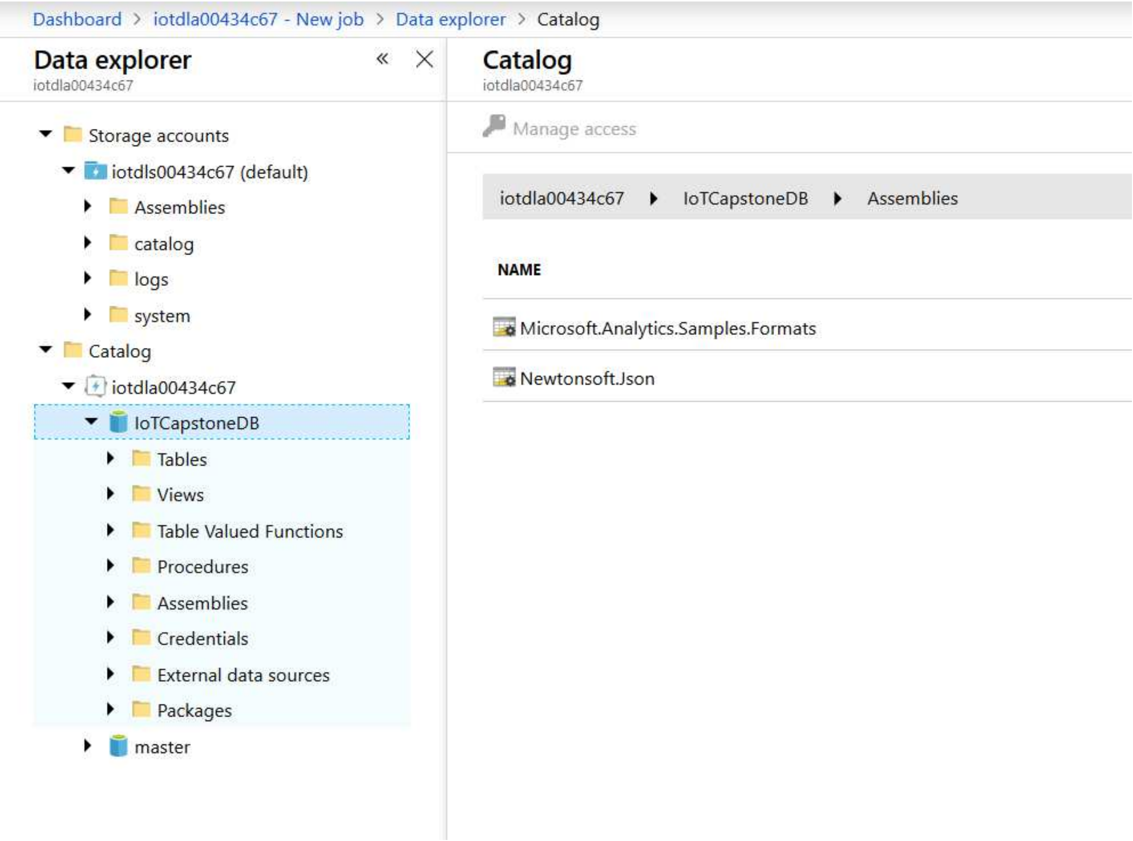


Status: Succeeded

Preparing 1m 1s — Queued 0s — Running N/A — Done

Progress	100%
AUs	1
Consumed AU-hours	0.01
Estimated cost	MYR 0.04
Efficiency	N/A
Issues	0 issues

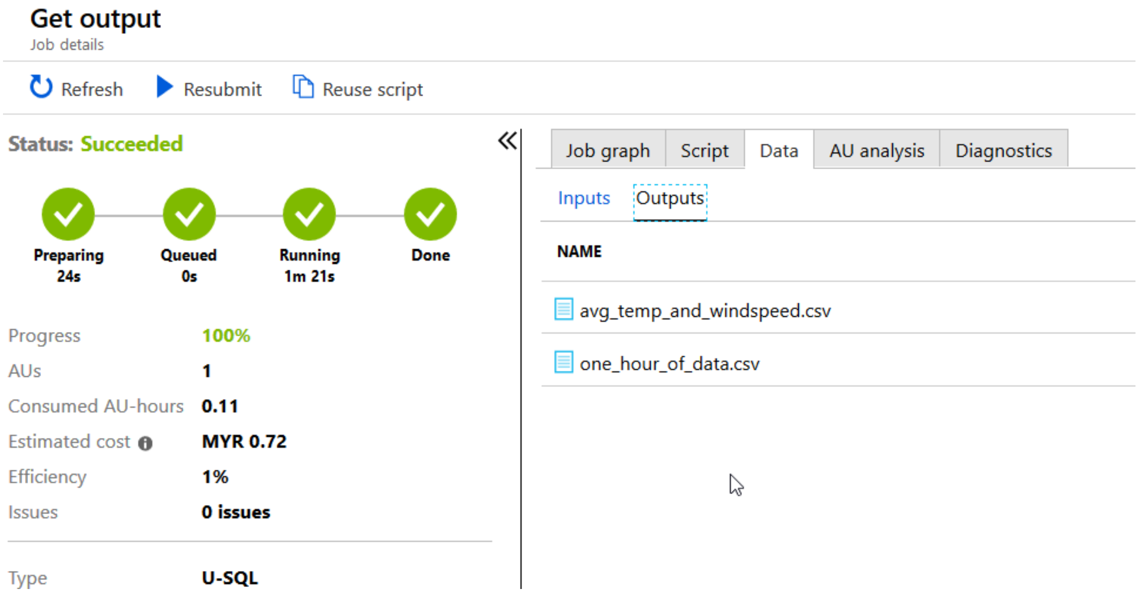
70. If you now browse under **IOTCapstoneDB** database in the **Data Explorer** in Data Lake Analytics, you should see the two assemblies are now listed under **Assemblies**.



NAME
Microsoft.Analytics.Samples.Formats
Newtonsoft.Json

71. Write a query in **Data Lake Analytics** that calculates the average wind speed and temperature for the last hour of telemetry received from the weather station.

```
REFERENCE ASSEMBLY IoTCapstoneDB.[Newtonsoft.Json];
REFERENCE ASSEMBLY IoTCapstoneDB.[Microsoft.Analytics.Samples.Formats];

USING Microsoft.Analytics.Samples.Formats.Json;

DECLARE @InputPath string =
"/telemetry/{date:yyyy}/{date:MM}/{date:dd}_{*}.json";

DECLARE @OutputOneHourFile string = "/output/one_hour_of_data.csv";
DECLARE @OutputAvgFile string = "/output/avg_temp_and_windspeed.csv";

// Extract all data and convert from JSON
@json =
EXTRACT
    date DateTime,
    EventProcessedUtcTime DateTime,
    PartitionId int,
    EventEnqueuedUtcTime DateTime,
    metadata_deviceType string,
    metadata_studentId string,
    metadata_uid string,
    telemetry_temperature double,
    telemetry_pressure double,
    telemetry_humidity double,
    telemetry_windSpeed double,
    telemetry_windDirection double,
    IoTHub_ConnectionDeviceId string
FROM
    @InputPath
USING new MultiLevelJsonExtractor(null,
    true,
    "EventProcessedUtcTime",
    "PartitionId",
    "EventEnqueuedUtcTime",
    "metadata.deviceType",
    "metadata.studentId",
    "metadata.uid",
    "telemetry.temperature",
    "telemetry.pressure",
    "telemetry.humidity",
    "telemetry.windSpeed",
    "telemetry.windDirection",
    "IoTHub.ConnectionDeviceId"
     );

// Restrict data to last hour
@lastHour =
    SELECT
        *
    FROM
        @json
    WHERE
        EventProcessedUtcTime > (DateTime.UtcNow - TimeSpan.FromHours(1));

// Output intermediate data set for grading
OUTPUT @lastHour
TO @OutputOneHourFile
USING Outputters.Csv(outputHeader:true);

// Determine the average temperature and windspeed for each
IoTHub_ConnectionDeviceId
// Output should be 3 columns:
//      IoTHub_ConnectionDeviceId
//      avg_temp
//      avg_windspeed

@avgdata =
    SELECT
        IoTHub_ConnectionDeviceId,
        AVG(telemetry_temperature) AS avg_temp,
        AVG(telemetry_windSpeed) AS avg_windspeed
    FROM @lastHour
    GROUP BY IoTHub_ConnectionDeviceId;

// Output averaged values for assessment
OUTPUT @avgdata
TO @OutputAvgFile
USING Outputters.Csv(outputHeader:true);
```



Get output

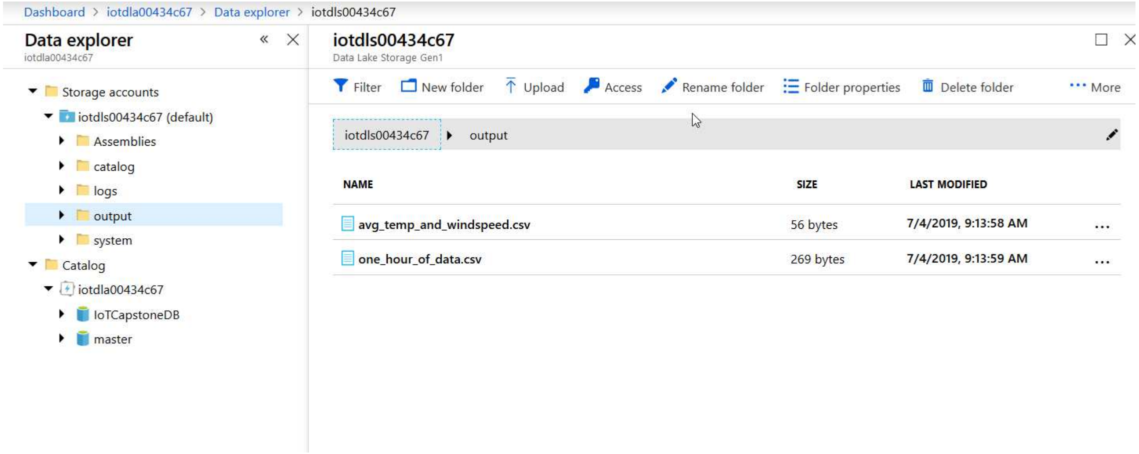
Job details

Status: Succeeded

Progress	100%
AUs	1
Consumed AU-hours	0.11
Estimated cost	MYR 0.72
Efficiency	1%
Issues	0 issues
Type	U-SQL

NAME
avg_temp_and_windspeed.csv
one_hour_of_data.csv

72. Download the csv files and open in excel or notepad to check



NAME	SIZE	LAST MODIFIED
avg_temp_and_windspeed.csv	56 bytes	7/4/2019, 9:13:58 AM
one_hour_of_data.csv	269 bytes	7/4/2019, 9:13:59 AM

73. Finally export the whole setup as a Azure template for grading.