
Table of Contents

Introduction	1.1
Connect Raspberry Pi with Node.js	1.2
Connect Raspberry Pi online simulator to Azure IoT Hub	1.3
Manage Cloud Device message with IoT Hub Explorer	1.4
Save messages to Azure Storage	1.5
Data Visualization in Power BI	1.6
Data Visualization with Web App	1.7
Weather forecast using Azure Machine Learning	1.8
Device Management using IoT Hub Explorer	1.9
Remote Monitoring and Notifications	1.10

Quick Solution

Azure IoT Suite	2.1
-----------------	-----

Azure IoT Workshop

The Workshop Manual

This book is the Workshop's Manual for the Azure IoT Course.

เทคโนโลยีเพล็กโกลในยุคปัจจุบันที่ทุกอย่างเชื่อมโยง

ลึกลับ หมวด จะเปลี่ยนวิธีการดำเนินธุรกิจแบบเดิมๆ

อุดสาหกรรม และ การใช้ชีวิตประจำวัน ทุกวันนี้เราอยู่ใน โลกของ “smart products” และ “intelligent business services” เทคโนโลยีที่มี ความก้าวหน้าอย่าง เช่น

Internet of Things (IoT), cloud computing, mobile, social และ big data ทำให้ทุกหน่วยงานจำเป็นต้องปรับกระบวนการ

ทำงานที่ทำอยู่ให้มีการเชื่อมโยงเข้ากับเครือข่าย

คอมพิวเตอร์มากขึ้น

คู่มือนี้ใช้ประกอบการอบรม เพื่อให้ผู้สนใจด้าน IoT มีโอกาส “ได้เรียนรู้และสัมผัสถึงพลังของ IoT ที่จะเชื่อมโยงอุปกรณ์ ของเรา

นำข้อมูลที่ไม่เคยสนใจในอดีตมาใช้ประโยชน์ เพื่อสะท้อนให้เราเห็นภาพของธุรกิจของเราอย่าง

ถ่องแท้ และเพิ่มมูลค่าให้กับธุรกิจ

Connect Raspberry Pi to Azure IoT Hub (Node.js)

In this tutorial, you begin by learning the basics of working with Raspberry Pi that's running Raspbian. You then learn how to seamlessly connect your devices to the cloud by using [Azure IoT Hub](#). For Windows 10 IoT Core samples, go to the [Windows Dev Center](#).

Don't have a kit yet? Try [Raspberry Pi online simulator](#). Or buy a new kit [here](#).

What you do

- Setup Raspberry Pi.
- Create an IoT hub.
- Register a device for Pi in your IoT hub.
- Run a sample application on Pi to send sensor data to your IoT hub.

Connect Raspberry Pi to an IoT hub that you create. Then you run a sample application on Pi to collect temperature and humidity data from a BME280 sensor. Finally, you send the sensor data to your IoT hub.

What you learn

- How to create an Azure IoT hub and get your new device connection string.
- How to connect Pi with a BME280 sensor.
- How to collect sensor data by running a sample application on Pi.
- How to send sensor data to your IoT hub.

What you need



- The Raspberry Pi 2 or Raspberry Pi 3 board.
- An active Azure subscription. If you don't have an Azure account, [create a free Azure trial account](#) in just a few minutes.
- A monitor, a USB keyboard, and mouse that connect to Pi.
- A Mac or a PC that is running Windows or Linux.
- An Internet connection.
- A 16 GB or above microSD card.
- A USB-SD adapter or microSD card to burn the operating system image onto the microSD card.
- A 5-volt 2-amp power supply with the 6-foot micro USB cable.

The following items are optional:

- An assembled Adafruit BME280 temperature, pressure, and humidity sensor.
- A breadboard.
- 6 F/M jumper wires.
- A diffused 10-mm LED.

[!NOTE] These items are optional because the code sample support simulated sensor data.

[!INCLUDE [iot-hub-get-started-create-hub-and-device](#)]

Setup Raspberry Pi

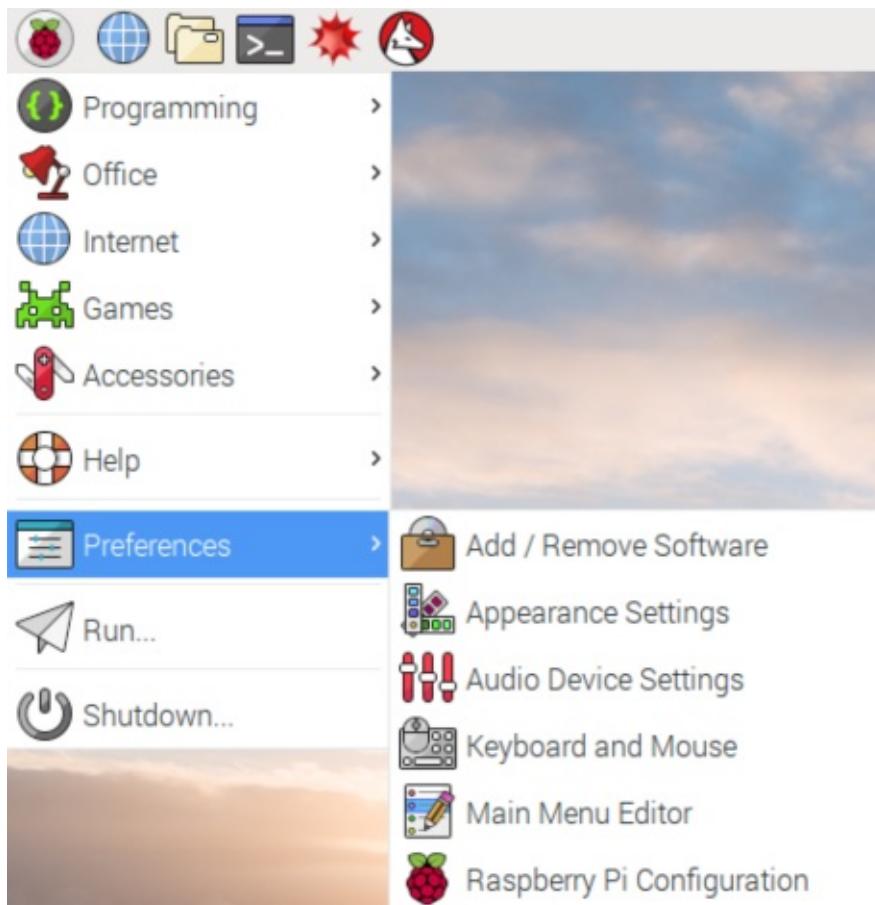
Install the Raspbian operating system for Pi

Prepare the microSD card for installation of the Raspbian image.

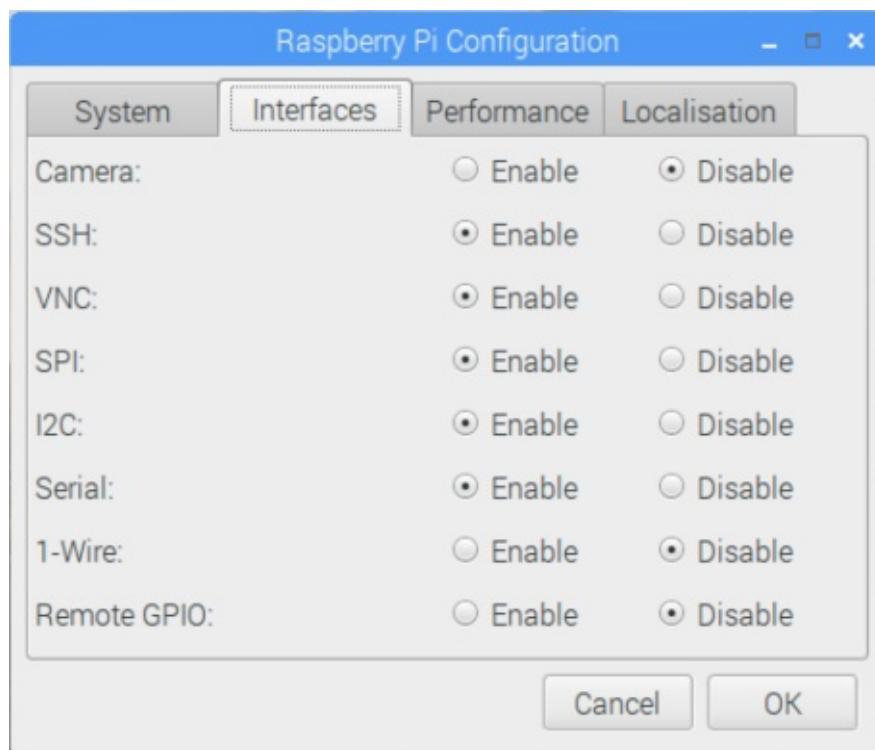
1. Download Raspbian.
 - i. [Download Raspbian Jessie with Pixel](#) (the .zip file).
 - ii. Extract the Raspbian image to a folder on your computer.
2. Install Raspbian to the microSD card.
 - i. [Download and install the Etcher SD card burner utility](#).
 - ii. Run Etcher and select the Raspbian image that you extracted in step 1.
 - iii. Select the microSD card drive. Note that Etcher may have already selected the correct drive.
 - iv. Click Flash to install Raspbian to the microSD card.
 - v. Remove the microSD card from your computer when installation is complete. It's safe to remove the microSD card directly because Etcher automatically ejects or unmounts the microSD card upon completion.
 - vi. Insert the microSD card into Pi.

Enable SSH and I2C

1. Connect Pi to the monitor, keyboard and mouse, start Pi and then log in Raspbian by using `pi` as the user name and `raspberry` as the password.
2. Click the Raspberry icon > **Preferences** > **Raspberry Pi Configuration**.



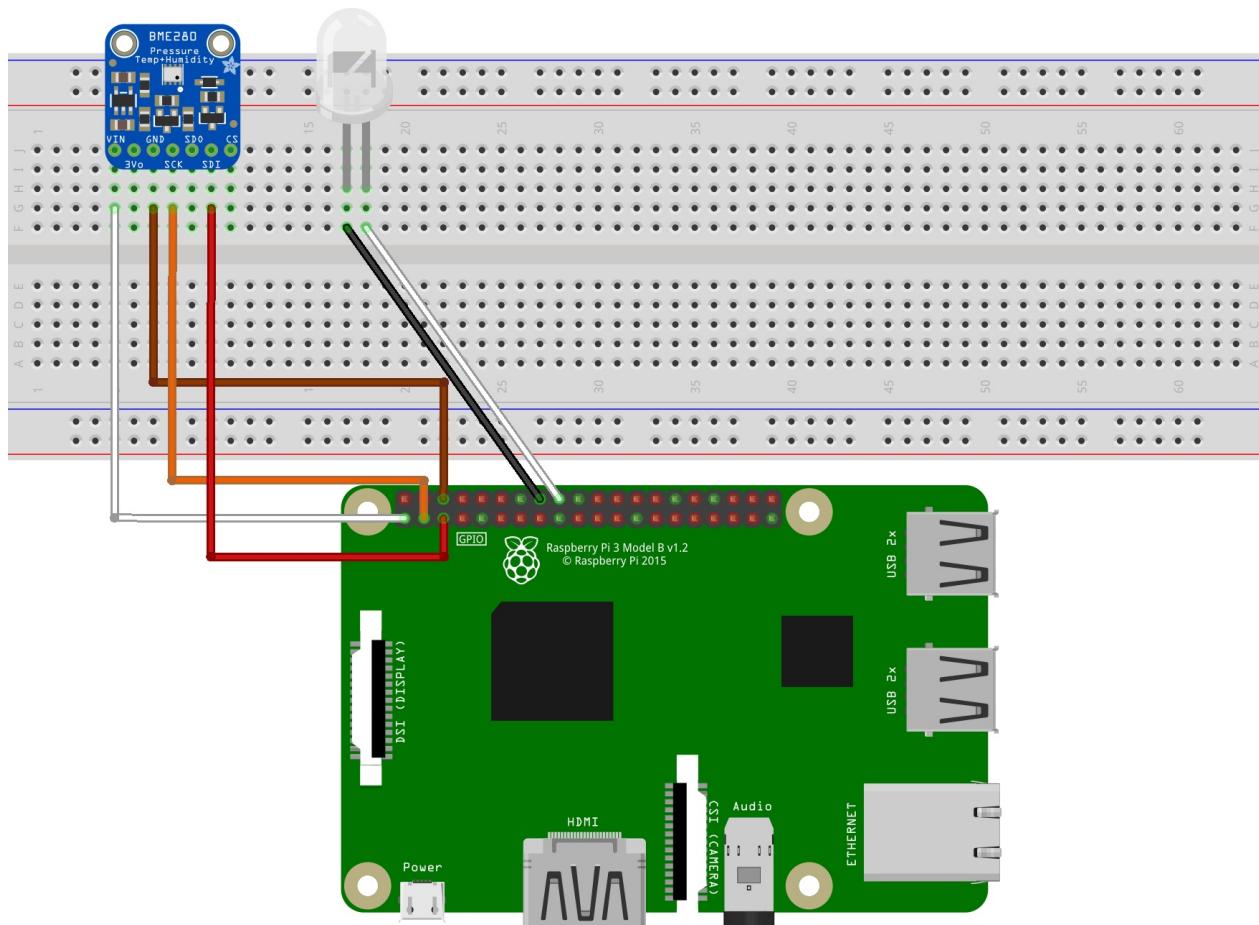
3. On the **Interfaces** tab, set **I2C** and **SSH** to **Enable**, and then click **OK**. If you don't have physical sensors and want to use simulated sensor data, this step is optional.



[!NOTE] To enable SSH and I2C, you can find more reference documents on raspberrypi.org and Adafruit.com.

Connect the sensor to Pi

Use the breadboard and jumper wires to connect an LED and a BME280 to Pi as follows. If you don't have the sensor, skip this section.



fritzing

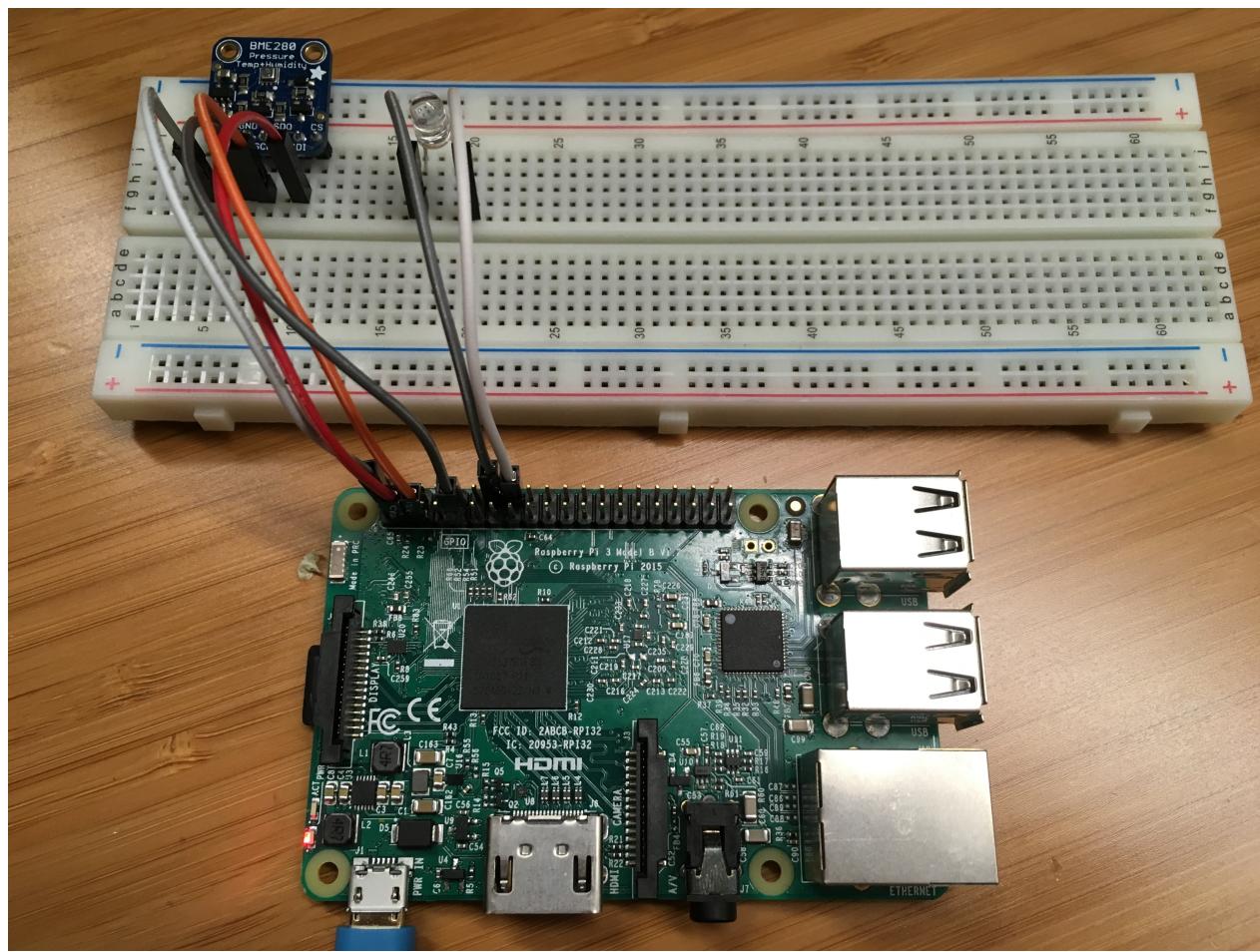
The BME280 sensor can collect temperature and humidity data. And the LED will blink if there is a communication between device and the cloud.

For sensor pins, use the following wiring:

Start (Sensor & LED)	End (Board)	Cable Color
VDD (Pin 5G)	3.3V PWR (Pin 1)	White cable
GND (Pin 7G)	GND (Pin 6)	Brown cable
SCK (Pin 8G)	I2C1 SDA (Pin 3)	Orange cable
SDI (Pin 10G)	I2C1 SCL (Pin 5)	Red cable
LED VDD (Pin 18F)	GPIO 24 (Pin 18)	White cable
LED GND (Pin 17F)	GND (Pin 20)	Black cable

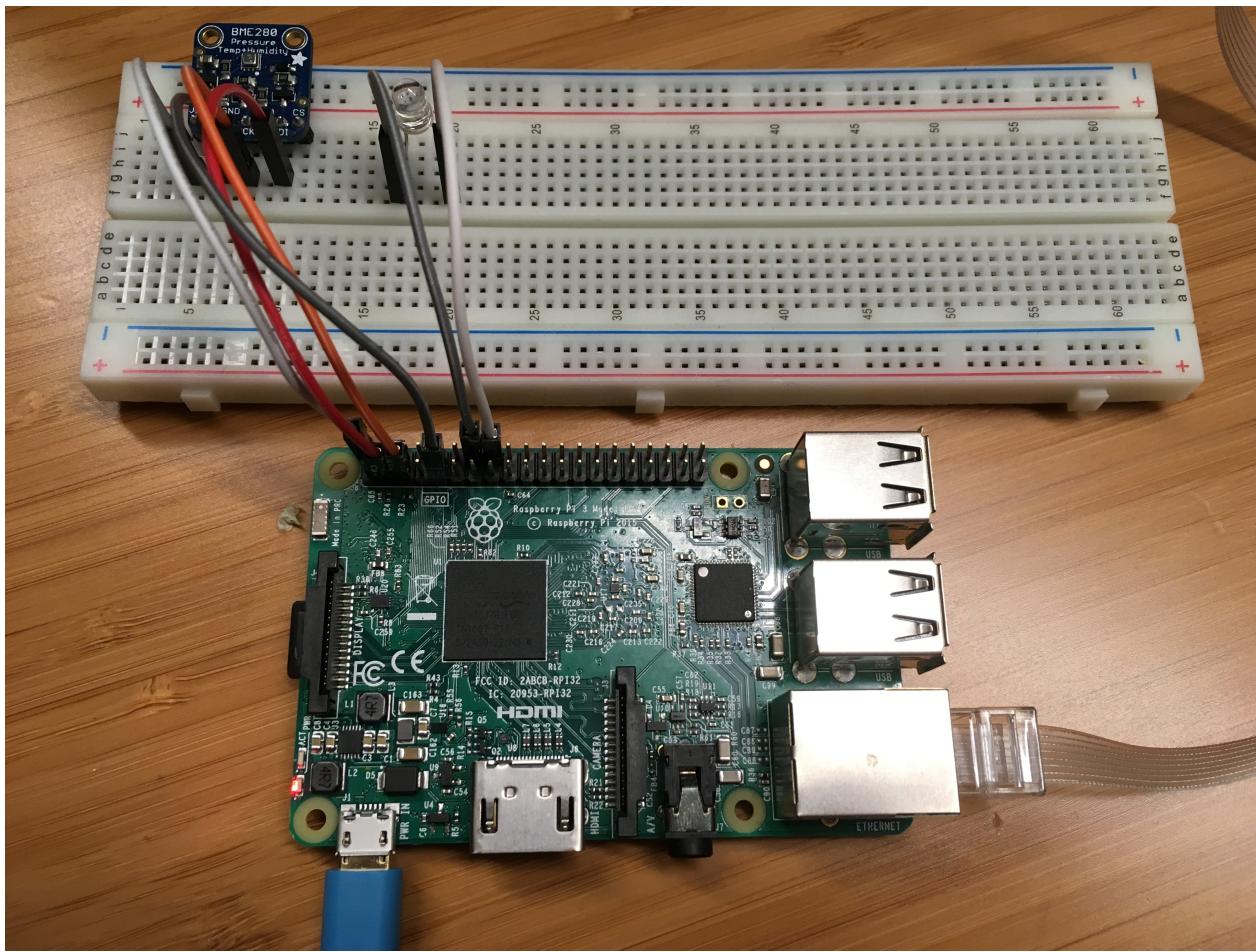
Click to view [Raspberry Pi 2 & 3 Pin mappings](#) for your reference.

After you've successfully connected BME280 to your Raspberry Pi, it should be like below image.



Connect Pi to the network

Turn on Pi by using the micro USB cable and the power supply. Use the Ethernet cable to connect Pi to your wired network or follow the [instructions from the Raspberry Pi Foundation](#) to connect Pi to your wireless network. After your Pi has been successfully connected to the network, you need to take a note of the [IP address of your Pi](#).



[!NOTE] Make sure that Pi is connected to the same network as your computer. For example, if your computer is connected to a wireless network while Pi is connected to a wired network, you might not see the IP address in the devdisco output.

Run a sample application on Pi

Clone sample application and install the prerequisite packages

1. Use one of the following SSH clients from your host computer to connect to your Raspberry Pi.
 - [PuTTY](#) for Windows. You need the IP address of your Pi to connect it via SSH.
 - The built-in SSH client on Ubuntu or macOS. You might need run `ssh pi@<ip address of pi>` to connect Pi via SSH.

[!NOTE] The default username is `pi`, and the password is `raspberry`.

2. Install Node.js and NPM to your Pi.

First you should check your Node.js version with the following command.

```
node -v
```

If the version is lower than 4.x or there is no Node.js on your Pi, then run the following command to install or update Node.js.

```
curl -sL http://deb.nodesource.com/setup_4.x | sudo -E bash  
sudo apt-get -y install nodejs
```

3. Clone the sample application by running the following command:

```
git clone https://github.com/Azure-Samples/iot-hub-node-raspberrypi-client-app
```

4. Install all packages by the following command. It includes Azure IoT device SDK, BME280 Sensor library and Wiring Pi library.

```
cd iot-hub-node-raspberrypi-client-app  
sudo npm install
```

[!NOTE] It might take several minutes to finish this installation process depending on your network connection.

Configure the sample application

1. Open the config file by running the following commands:

```
nano config.json
```

```

1. pi@raspberrypi: ~/xshi/iot-hub-node-raspberrypi-client-app (ssh)
GNU nano 2.2.6          File: config.json

{
    "simulatedData": false,
    "interval": 2000,
    "deviceId": "Raspberry Pi Node",
    "LEDPin": 7,
    "messageMax": 256,
    "credentialPath": "~/.iot-hub",
    "temperatureAlert": 30,
    "i2c0Option": {
        "pin": 9,
        "i2cBusNo": 1,
        "i2cAddress": 119
    }
}

[ Read 14 lines (Converted from DOS format) ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U Uncut Text ^T To Spell

```

There are two items in this file you can configure. The first one is `interval`, which defines the time interval between two messages that send to cloud. The second one `simulatedData`, which is a Boolean value for whether to use simulated sensor data or not.

If you **don't have the sensor**, set the `simulatedData` value to `true` to make the sample application create and use simulated sensor data.

2. Save and exit by pressing Control-O > Enter > Control-X.

Run the sample application

1. Run the sample application by running the following command:

```
sudo node index.js '<your Azure IoT hub device connection string>'
```

[!NOTE] Make sure you copy-paste the device connection string into the single quotes.

You should see the following output that shows the sensor data and the messages that are sent to your IoT hub.

```
● ● ● 1. pi@raspberrypi: ~/xshi/iot-hub-node-raspberrypi-client-app (ssh)
pi@raspberrypi:~/xshi/iot-hub-node-raspberrypi-client-app $ sudo node index.js 'HostName=IoTGetStarted.azure-devices.net;DeviceId=new-device;SharedAccessKey=d0q1tgHj6U8Wb+3PX5I9ism5eIGtJLRTb89M7C3eUQ0='
Sending message: {"messageId":1,"deviceId":"Raspberry Pi Node","temperature":22.09817088597284,"humidity":79.44195810046365}
Message sent to Azure IoT Hub
Sending message: {"messageId":2,"deviceId":"Raspberry Pi Node","temperature":26.183512024547063,"humidity":61.42521225412357}
Message sent to Azure IoT Hub
Sending message: {"messageId":3,"deviceId":"Raspberry Pi Node","temperature":29.520917564174873,"humidity":62.00662798413029}
Message sent to Azure IoT Hub
Sending message: {"messageId":4,"deviceId":"Raspberry Pi Node","temperature":22.591091037492344,"humidity":70.1062754469173}
Message sent to Azure IoT Hub
Sending message: {"messageId":5,"deviceId":"Raspberry Pi Node","temperature":26.451696863853265,"humidity":72.71690012385488}
Message sent to Azure IoT Hub
Sending message: {"messageId":6,"deviceId":"Raspberry Pi Node","temperature":25.
```

Next steps

You've run a sample application to collect sensor data and send it to your IoT hub.

Connect Raspberry Pi online simulator to Azure IoT Hub (Node.js)

In this tutorial, you begin by learning the basics of working with Raspberry Pi online simulator. You then learn how to seamlessly connect the Pi simulator to the cloud by using [Azure IoT Hub](#).

If you have physical devices, visit [Connect Raspberry Pi to Azure IoT Hub](#) to get started.

What you do

- Learn the basics of Raspberry Pi online simulator.
- Create an IoT hub.
- Register a device for Pi in your IoT hub.
- Run a sample application on Pi to send simulated sensor data to your IoT hub.

Connect simulated Raspberry Pi to an IoT hub that you create. Then you run a sample application with the simulator to generate sensor data. Finally, you send the sensor data to your IoT hub.

What you learn

- How to create an Azure IoT hub and get your new device connection string.
- How to work with Raspberry Pi online simulator.
- How to send sensor data to your IoT hub.

Overview of Raspberry Pi web simulator

Click the button to launch Raspberry Pi online simulator.

[!div class="button"] [Start Raspberry Pi simulator](#)

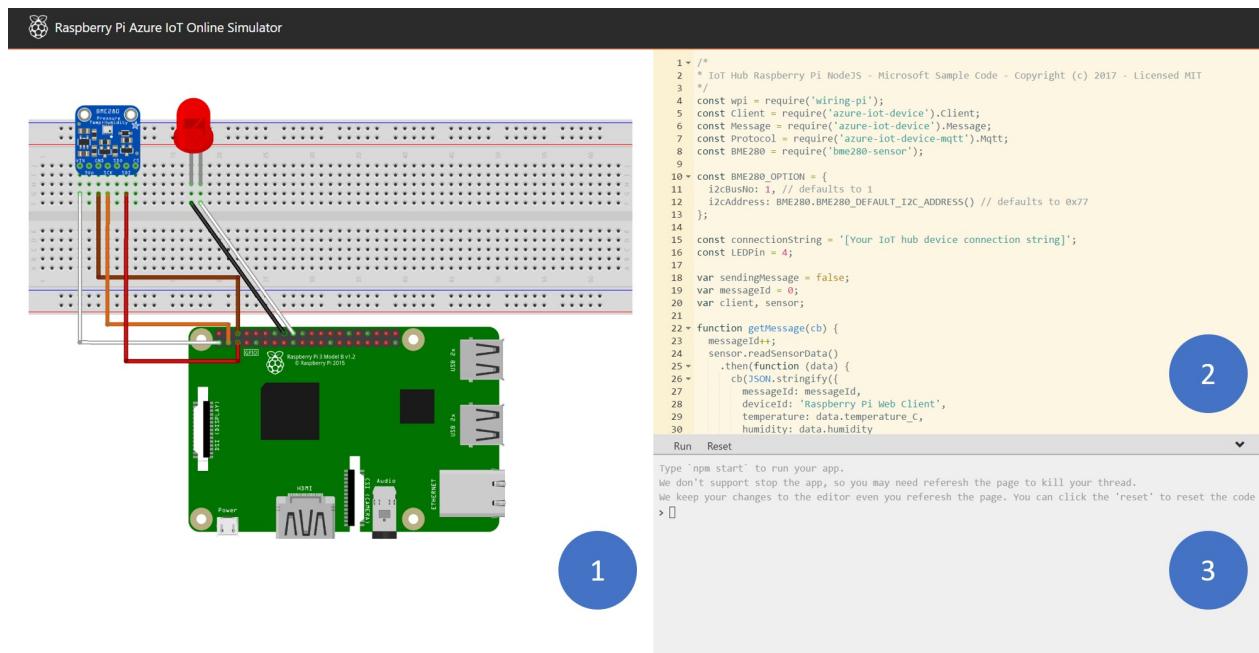
There are three areas in the web simulator.

- Assembly area - The default circuit is that a Pi connects with a BME280 sensor and an LED. The area is locked in preview version so currently you cannot do customization.
- Coding area - An online code editor for you to code with Raspberry Pi. The default sample application helps to collect sensor data from BME280 sensor and sends to your

Azure IoT Hub. The application is fully compatible with real Pi devices.

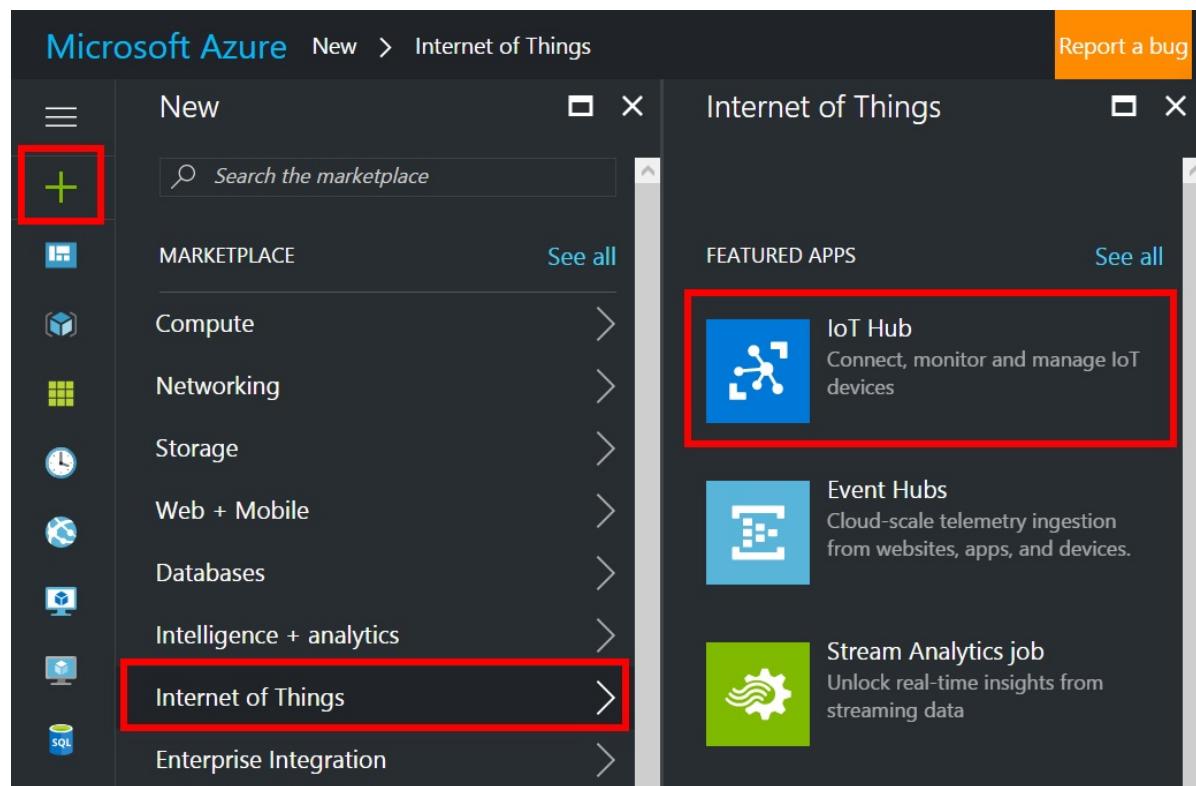
- Integrated console window - It shows the output of your code. At the top of this window, there are three buttons.
 - **Run** - Run the application in the coding area.
 - **Reset** - Reset the coding area to the default sample application.
 - **Fold/Expand** - On the right side there is a button for you to fold/expand the console window.

[!NOTE] The Raspberry Pi web simulator is now available in preview version. We'd like to hear your voice in the [Gitter Chatroom](#). The source code is public on [Github](#).



Create an IoT hub

1. In the [Azure portal](#), click **New > Internet of Things > IoT Hub**.



2. In the **IoT hub** pane, enter the following information for your IoT hub:

Name: It is the name for your IoT hub. If the name you enter is valid, a green check mark appears.

Pricing and scale tier: Select the free F1 tier. This option is sufficient for this demo.
See [pricing and scale tier](#).

Resource group: Create a resource group to host the IoT hub or use an existing one.
See [Using resource groups to manage your Azure resources](#).

Location: Select the closest location to you where the IoT hub is created.

Pin the dashboard: Check this option for easy access to your IoT hub from the dashboard.

IoT hub Microsoft

* Name ✓

* Pricing and scale tier >
F1 - Free

* IoT Hub units ⓘ

* Device-to-cloud partitions ⓘ

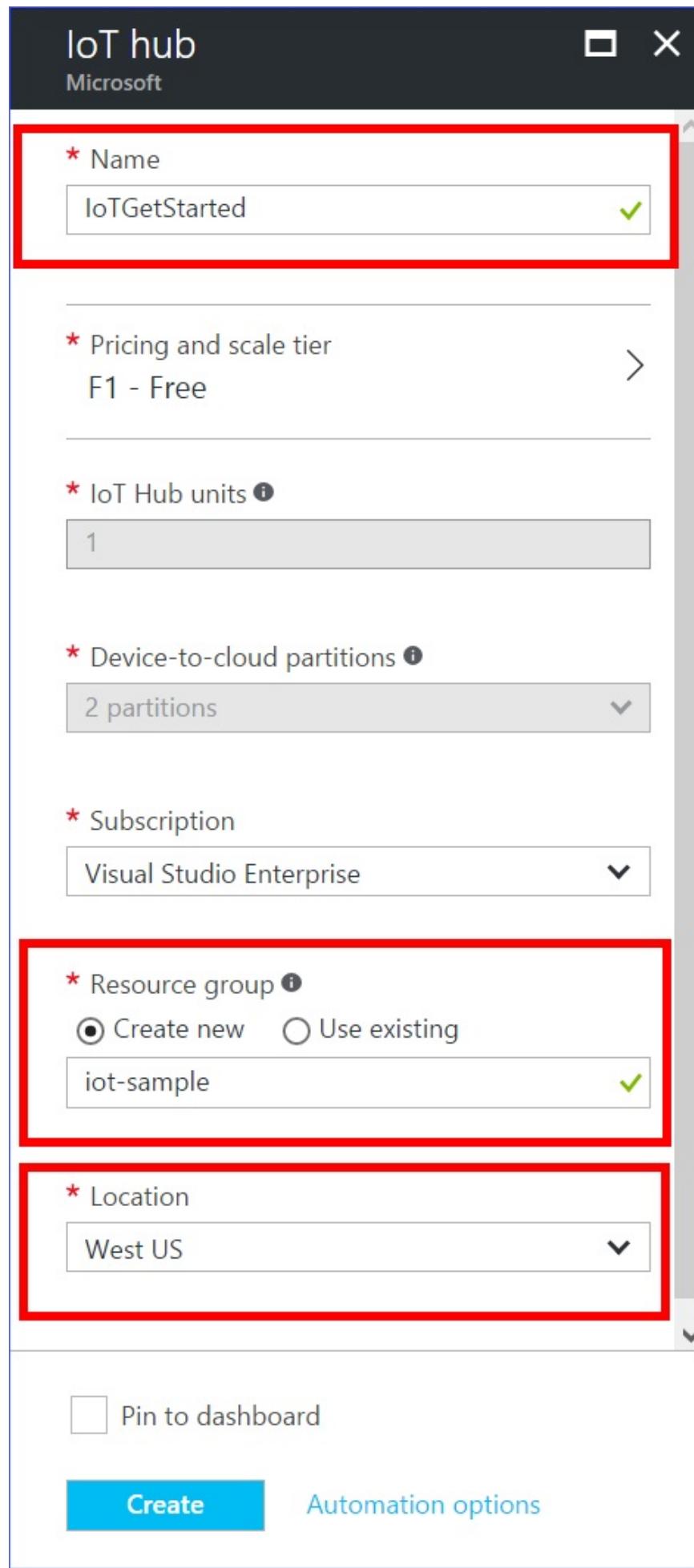
* Subscription

* Resource group ⓘ Create new Use existing
 ✓

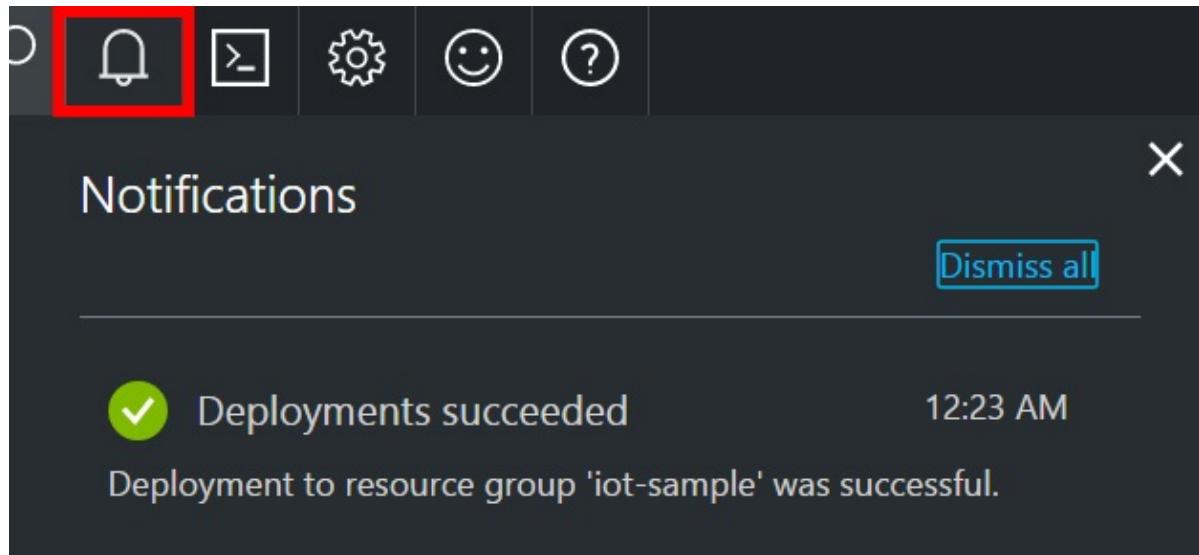
* Location >
West US

Pin to dashboard

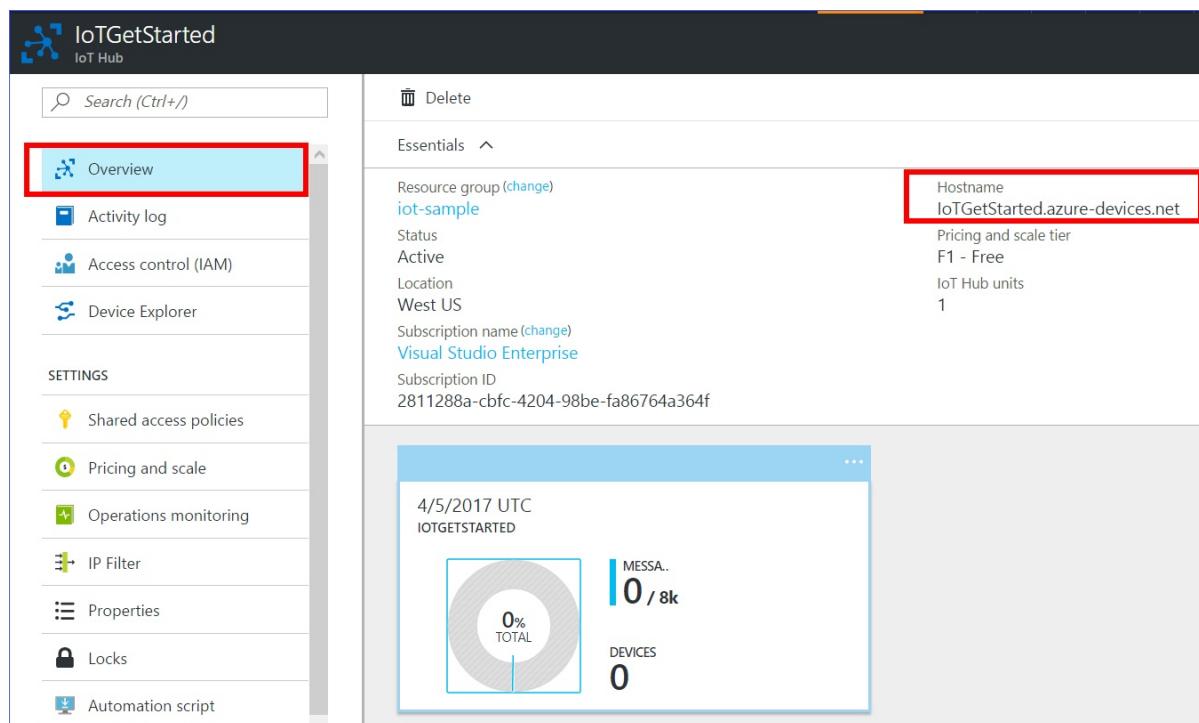
Create Automation options



3. Click **Create**. It could take a few minutes for your IoT hub to be created. You can see progress in the **Notifications** pane.



4. Once your IoT hub is created, click it from the dashboard. Make a note of the **Hostname**, and then click **Shared access policies**.



5. In the **Shared access policies** pane, click the **iothubowner** policy, and then copy and make a note of the **Connection string** of your IoT hub. For more information, see [Control access to IoT Hub](#).

The screenshot shows the 'Shared access policies' section of the Azure IoT Hub. On the left, there's a sidebar with 'Overview', 'Activity log', 'Access control (IAM)', 'Device Explorer', 'Pricing and scale', 'Operations monitoring', 'IP Filter', and 'Properties'. The 'Shared access policies' item is highlighted with a red box. The main area shows a table with one row for the 'iothubowner' policy. The table has two columns: 'POLICY' and 'PERMISSIONS'. The 'iothubowner' row is highlighted with a red box. The 'PERMISSIONS' column lists 'registry write, service connect', 'service connect', 'device connect', 'registry read', and 'registryReadWrite'. To the right of the table, there's a modal window titled 'iothubowner' with tabs for 'IoTGetStarted' and 'More'. It shows the 'Access policy name' as 'iothubowner', 'Permissions' (Registry read, Registry write, Service connect, Device connect all checked), and 'Shared access keys' (Primary key and Secondary key both shown with copy icons). A red box highlights the 'Connection string—primary key' field, which contains 'HostName=IoTGetStarted.azure-devices.r'.

Register a device in the IoT hub for the your device

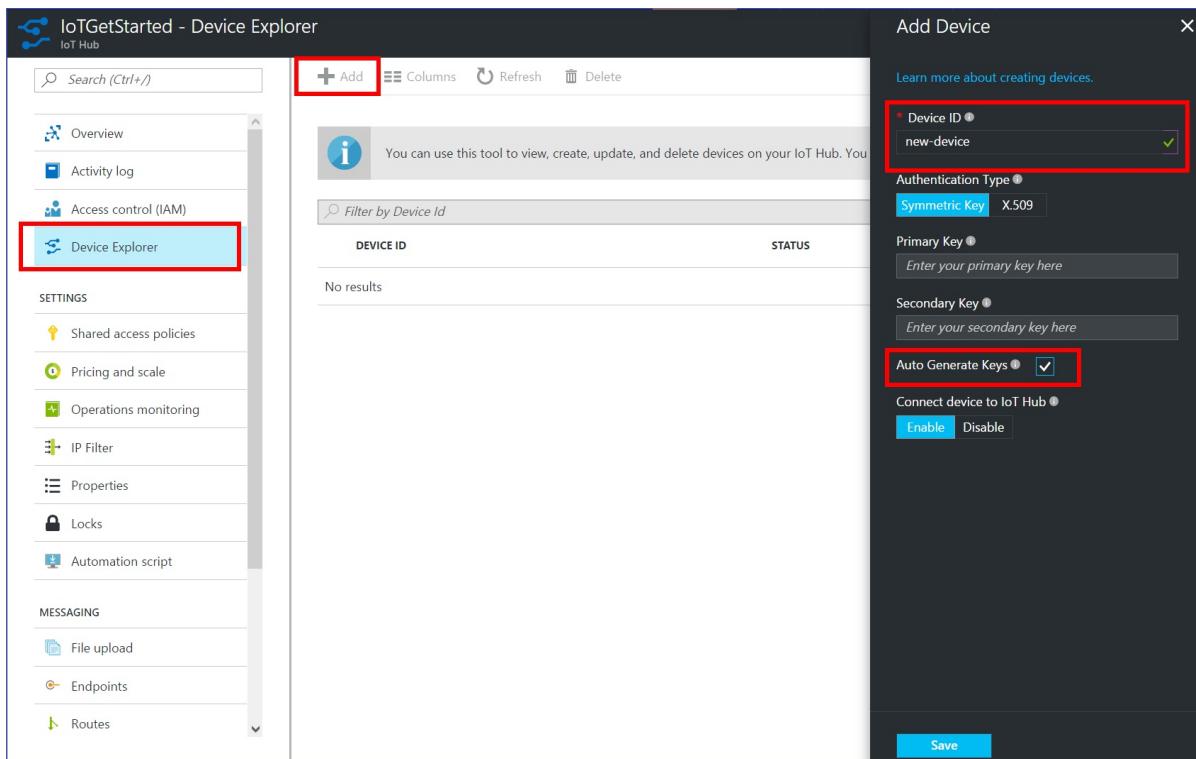
1. In the [Azure portal](#), open your IoT hub.
2. Click **Device Explorer**.
3. In the Device Explorer pane, click **Add** to add a device to your IoT hub.

Device ID: The ID of the new device. Device IDs are case sensitive.

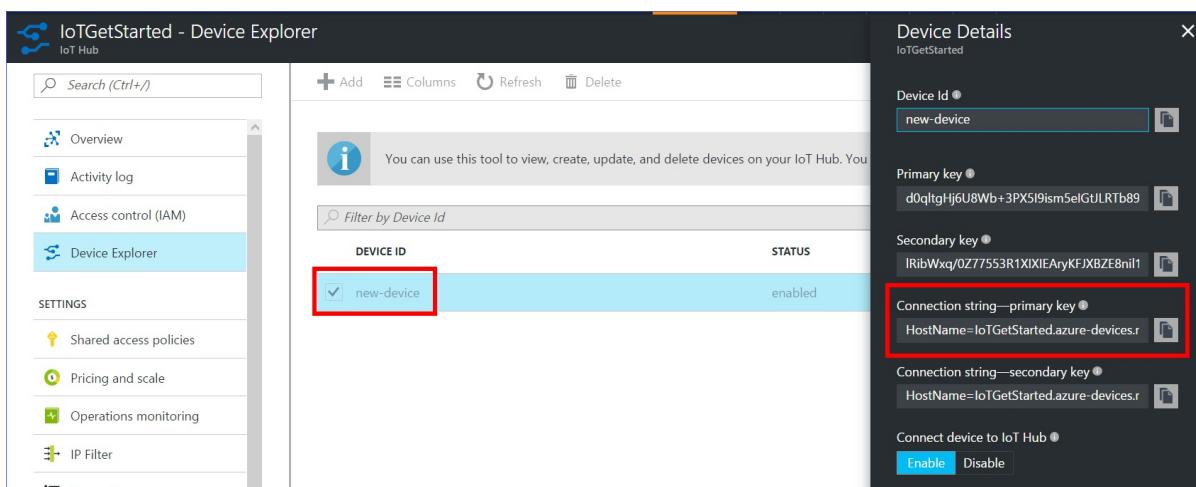
Authentication Type: Select **Symmetric Key**.

Auto Generate Keys: Check this field.

Connect device to IoT Hub: Click **Enable**.



4. Click **Save**.
5. After the device is created, open the device in the **Device Explorer** pane.
6. Make a note of the primary key of the connection string.



Run a sample application on Pi web simulator

1. In coding area, make sure you are working on the default sample application. Replace the placeholder in Line 15 with the Azure IoT hub device connection string.

```
14
15 const connectionString = '[Your IoT hub device connection string]';
16 const LEDPin = 4;
17
```

2. Click **Run** or type `npm start` to run the application.

You should see the following output that shows the sensor data and the messages that are sent to your IoT hub

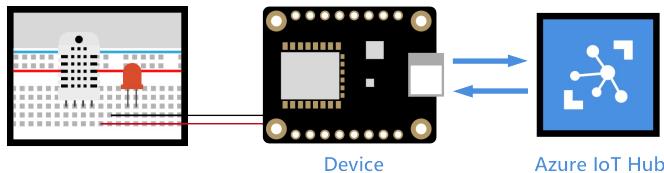
```
Running Reset

Type `npm start` to run your app.
We don't support stop the app, so you may need refresh the page to kill your thread.
We keep your changes to the editor even you refresh the page. You can click the 'reset' to reset the code
>
Sending message: {"messageId":1,"deviceId":"Raspberry Pi Web Client","temperature":25.584710773750324,"humidity"
>
Message sent to Azure IoT Hub
```

Next steps

You've run a sample application to collect sensor data and send it to your IoT hub.

Use iothub-explorer to send and receive messages between your device and IoT Hub



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

[iothub-explorer](#) has a handful of commands that makes IoT Hub management easier. This tutorial focuses on how to use iothub-explorer to send and receive messages between your device and your IoT hub.

What you will learn

You learn how to use iothub-explorer to monitor device-to-cloud messages and to send cloud-to-device messages. Device-to-cloud messages could be sensor data that your device collects and then sends to your IoT hub. Cloud-to-device messages could be commands that your IoT hub sends to your device to blink an LED that is connected to your device.

What you will do

- Use iothub-explorer to monitor device-to-cloud messages.
- Use iothub-explorer to send cloud-to-device messages.

What you need

- Tutorial [Setup your device](#) completed which covers the following requirements:

- An active Azure subscription.
- An Azure IoT hub under your subscription.
- A client application that sends messages to your Azure IoT hub.
- iothub-explorer. ([Install iothub-explorer](#))

Monitor device-to-cloud messages

To monitor messages that are sent from your device to your IoT hub, follow these steps:

1. Open a console window.
2. Run the following command:

```
iothub-explorer monitor-events <device-id> --login "<IoTHubConnectionString>"
```

[!Note] Get `<device-id>` and `<IoTHubConnectionString>` from your IoT hub. Make sure you've finished the previous tutorial. Or you can try to use `iothub-explorer monitor-events <device-id> --login "HostName=<my-hub>.azure-devices.net;SharedAccessKeyName=<my-policy>;SharedAccessKey=<my-policy-key>"` if you have `HostName`, `SharedAccessKeyName` and `SharedAccessKey`.

Send cloud-to-device messages

To send a message from your IoT hub to your device, follow these steps:

1. Open a console window.
2. Start a session on your IoT hub by running the following command:

```
iothub-explorer login <IoTHubConnectionString>
```

3. Send a message to your device by running the following command:

```
iothub-explorer send <device-id> <message>
```

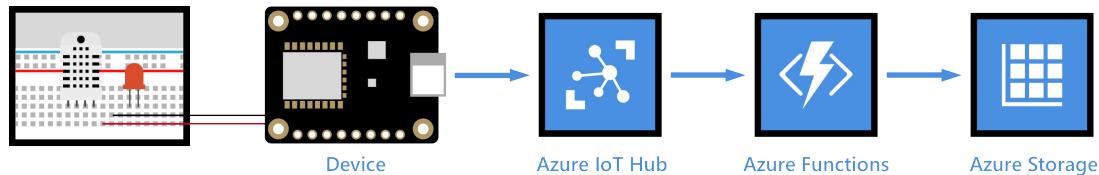
The command blinks the LED that is connected to your device and sends the message to your device.

[!Note] There is no need for the device to send a separate ack command back to your IoT hub upon receiving the message.

Next steps

You've learned how to monitor device-to-cloud messages and send cloud-to-device messages between your IoT device and Azure IoT Hub.

Save IoT Hub messages that contain information like sensor data to Azure table storage



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

What you will learn

You learn how to create an Azure storage account and an Azure Function App to store IoT Hub messages in Azure table storage.

What you will do

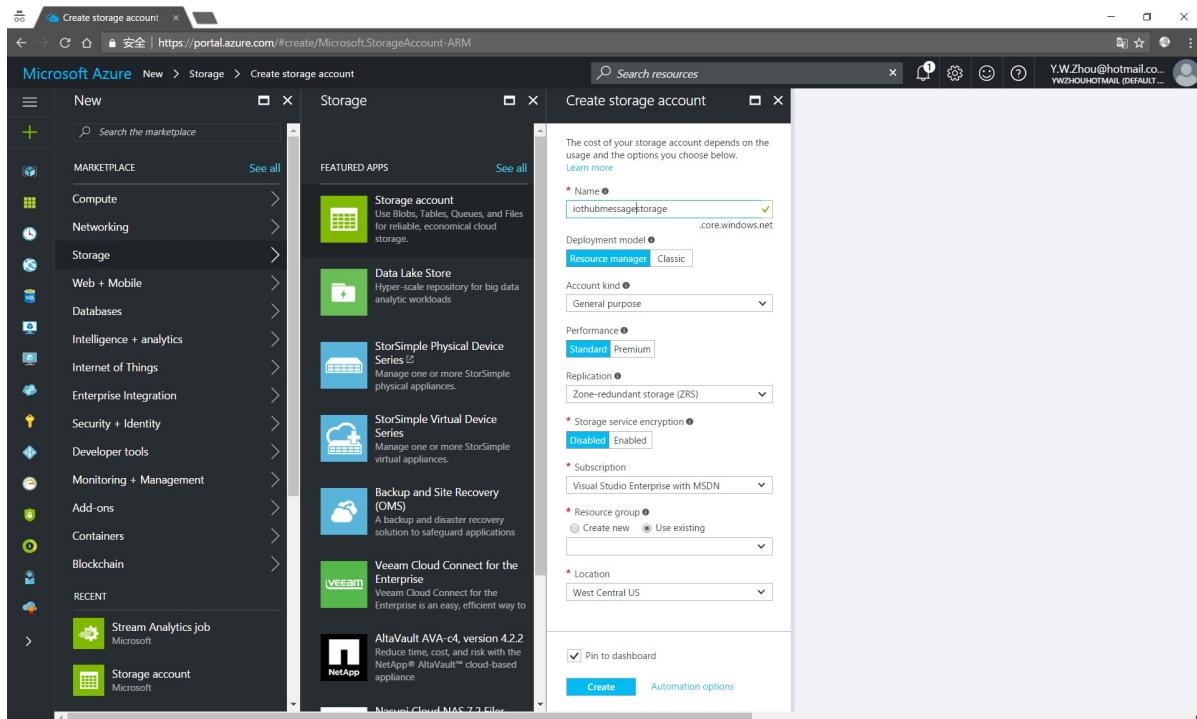
- Create an Azure storage account.
- Prepare for IoT Hub connection to read messages.
- Create and deploy an Azure Function App.

What you will need

- Tutorial [Setup your device](#) completed which covers the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.
 - A running application that sends messages to your Azure IoT hub.

Create an Azure storage account

1. In the Azure portal, click **New > Storage > Storage account**.
2. Enter the necessary information for the storage account:



Name: The name of the storage account. The name must be globally unique.

Resource group: Use the same resource group that your IoT hub uses.

Pin to dashboard: Check this option for easy access to your IoT hub from the dashboard.

3. Click **Create**.

Prepare for IoT Hub connection to read messages

IoT Hub exposes a built-in Event Hub-compatible endpoint to enable applications to read IoT Hub messages. Meanwhile, applications use consumer groups to read data from IoT Hub. Before creating an Azure Function App to read data from your IoT hub, you need to:

- Get the connection string of your IoT hub endpoint.
- Create a consumer group for your IoT hub.

Get the connection string of your IoT hub endpoint

1. Open your IoT hub.
2. On the **IoT Hub** pane, click **Endpoints** under **MESSAGING**.
3. On the right pane, click **Events** under **Built-in endpoints**.
4. In the **Properties** pane, make a note of the following values:
 - Event Hub-compatible endpoint
 - Event Hub-compatible name

5. On the **IoT Hub** pane, click **Shared access policies** under **SETTINGS**.
6. Click **iothubowner**.
7. Make a note of the **Primary key** value.
8. Make up the connection string of your IoT hub endpoint as follows:

```
Endpoint=<Event Hub-compatible endpoint>
endpoint>;SharedAccessKeyName=iothubowner;SharedAccessKey=<Primary key>
```

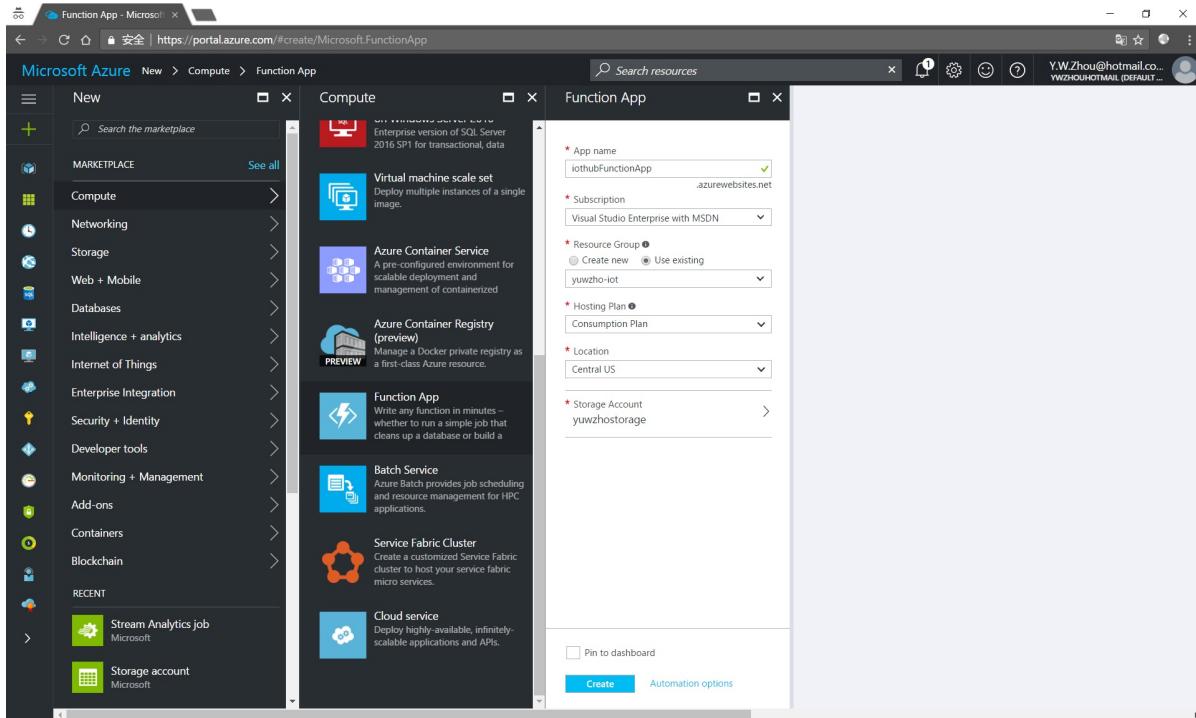
[!Note] Replace `<Event Hub-compatible endpoint>` and `<Primary key>` with the values you noted down.

Create a consumer group for your IoT hub

1. Open your IoT hub.
2. On the **IoT Hub** pane, click **Endpoints** under **MESSAGING**.
3. On the right pane, click **Events** under **Built-in endpoints**.
4. In the **Properties** pane, enter a name under **Consumer groups** and make a note of the name.
5. Click **Save**.

Create and deploy an Azure Function App

1. In the [Azure portal](#), click **New > Compute > Function App**.
2. Enter the necessary information for the Function App.



App name: The name of the Function App. The name must be globally unique.

Resource group: Use the same resource group that your IoT Hub uses.

Storage Account: The storage account that you created.

Pin to dashboard: Check this option for easy access to the Function App from the dashboard.

3. Click **Create**.
4. Open the Function App once it is created.
5. Create a new function in the Function App.
 - i. Click **New Function**.
 - ii. Select **JavaScript** for **Language**, and **Data Processing** for **Scenario**.
 - iii. Click the **EventHubTrigger-JavaScript** template.
 - iv. Enter the necessary information for the template.

Name your function: The name of the function.

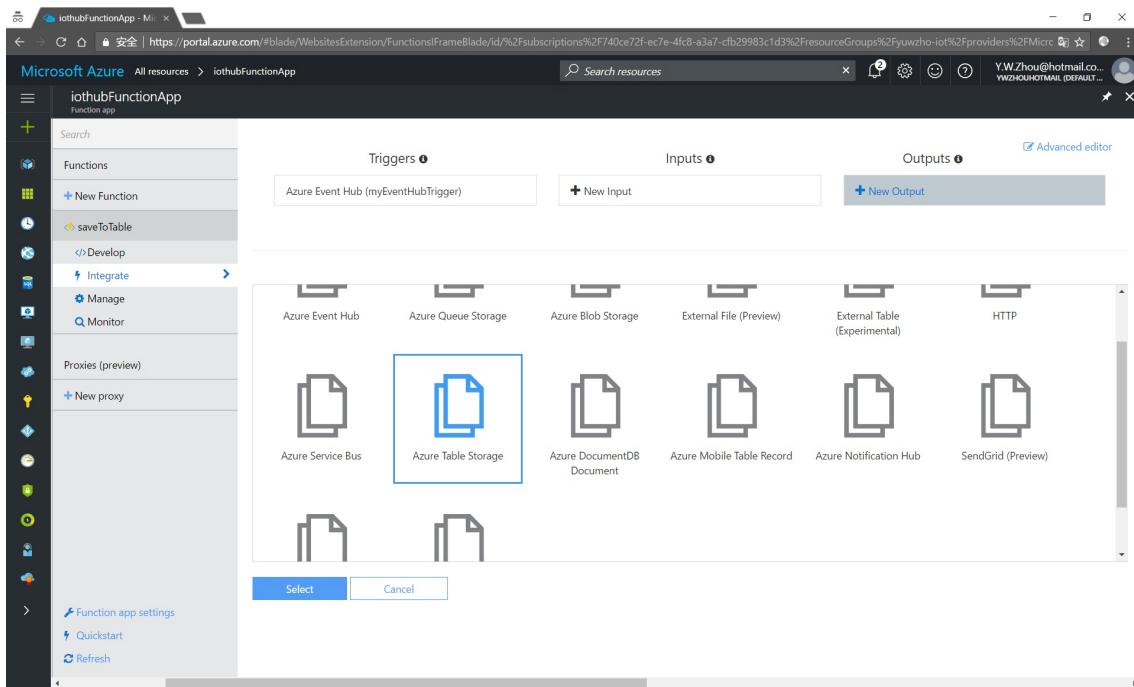
Event Hub name: The Event Hub-compatible name you noted down.

Event Hub connection: Click new to add the connection string of your IoT hub endpoint that you made up.

v. Click **Create**.

6. Configure an output of the function.

i. Click **Integrate > New Output > Azure Table Storage > Select**.



ii. Enter the necessary information.

Table parameter name: Use `outputTable` for the name, which will be used in the Azure Functions' code.

Table name: Use `deviceData` for the name.

Storage account connection: Click **new** and select or input your storage account.

iii. Click **Save**.

7. Under **Triggers**, click **Azure Event Hub (myEventHubTrigger)**.
8. Under **Event Hub consumer group**, enter the name of the consumer group that you created, and then click **Save**.
9. Click **Develop**, and then click **View files**.
10. Replace the code in `index.js` with the following, and then click **Save**.

```
'use strict';

// This function is triggered each time a message is received in the IoTHub.
// The message payload is persisted in an Azure Storage Table

module.exports = function (context, iotHubMessage) {
    context.log('Message received: ' + JSON.stringify(iotHubMessage));
    var date = Date.now();
    var partitionKey = Math.floor(date / (24 * 60 * 60 * 1000)) + '';
    var rowKey = date + '';
    context.bindings.outputTable = {
        "partitionKey": partitionKey,
        "rowKey": rowKey,
        "message": JSON.stringify(iotHubMessage)
    };
    context.done();
};
```

By now, you have created the Function App. It stores messages that your IoT hub receives in your Azure table storage.

[!Note] You can use the **Run** button to test the Function App. When you click **Run**, the test message is sent to your IoT hub. The arrival of the message should trigger the Function App to start and then save the message to your Azure table storage. The **Logs** pane records the details of the process.

Verify your message in your table storage

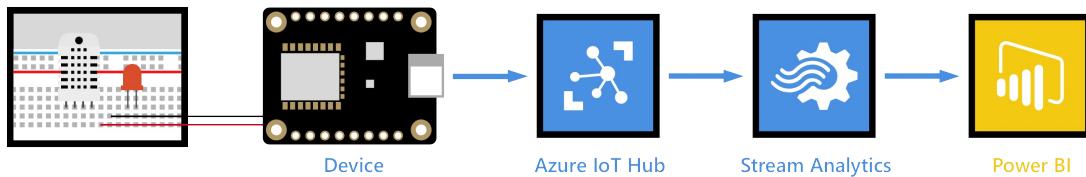
1. Run the sample application on your device to send messages to your IoT hub.
2. [Download and install Microsoft Azure Storage Explorer](#).
3. Open Microsoft Azure Storage Explorer, click **Add an Azure Account > Sign in**, and then sign in to your Azure account.
4. Click your Azure subscription > **Storage Accounts** > your storage account > **Tables** > **deviceData**.

You should see messages sent from your device to your IoT hub logged in the `deviceData` table.

Next steps

You've successfully created your Azure storage account and Azure Function App to store messages that your IoT hub receives in your Azure table storage.

Visualize real-time sensor data from Azure IoT Hub using Power BI



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

What you learn

You learn how to visualize real-time sensor data that your Azure IoT hub receives by Power BI. If you want to try visualize the data in your IoT hub with Web Apps, please see [Use Azure Web Apps to visualize real-time sensor data from Azure IoT Hub](#).

What you do

- Get your IoT hub ready for data access by adding a consumer group.
- Create, configure and run a Stream Analytics job for data transfer from your IoT hub to your Power BI account.
- Create and publish a Power BI report to visualize the data.

What you need

- Tutorial [Setup your device](#) completed which covers the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.
 - A client application that sends messages to your Azure IoT hub.
- A Power BI account. ([Try Power BI for free](#))

Add a consumer group to your IoT hub

Consumer groups are used by applications to pull data from Azure IoT Hub. In this lesson, you create a consumer group to be used by a coming Azure service to read data from your IoT hub.

To add a consumer group to your IoT hub, follow these steps:

1. In the [Azure portal](#), open your IoT hub.
2. Click **Endpoints** on the left pane, select **Events** on the middle pane, enter a name under **Consumer groups** on the right pane, and then click **Save**.

NAME	ENDPOINT
Cloud to device feedback	messages/servicebound/feedback
Events	messages/events

NAME	ENDPOINT TYPE
No results	

Create, configure, and run a Stream Analytics job

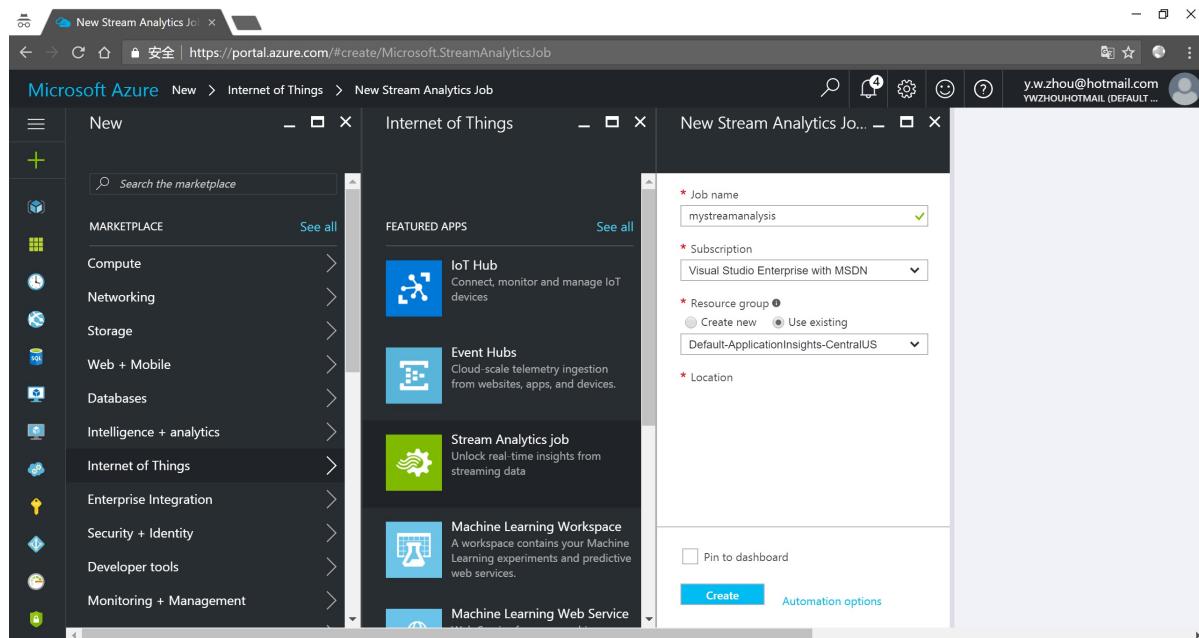
- ## Create a Stream Analytics job
1. In the Azure portal, click New > Internet of Things > Stream Analytics job.
 2. Enter the following information for the job.

Job name: The name of the job. The name must be globally unique.

Resource group: Use the same resource group that your IoT hub uses.

Location: Use the same location as your resource group.

Pin to dashboard: Check this option for easy access to your IoT hub from the dashboard.



3. Click **Create**.

Add an input to the Stream Analytics job

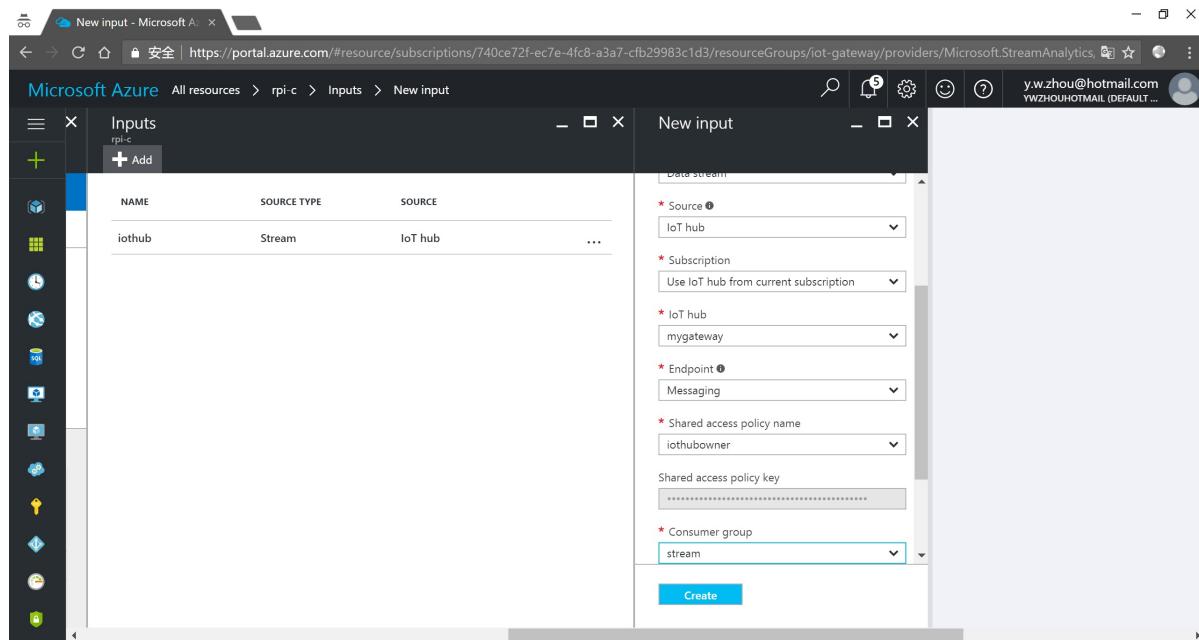
1. Open the Stream Analytics job.
2. Under **Job Topology**, click **Inputs**.
3. In the **Inputs** pane, click **Add**, and then enter the following information:

Input alias: The unique alias for the input.

Source: Select **IoT hub**.

Consumer group: Select the consumer group you just created.

4. Click **Create**.



Add an output to the Stream Analytics job

1. Under **Job Topology**, click **Outputs**.
2. In the **Outputs** pane, click **Add**, and then enter the following information:

Output alias: The unique alias for the output.

Sink: Select **Power BI**.

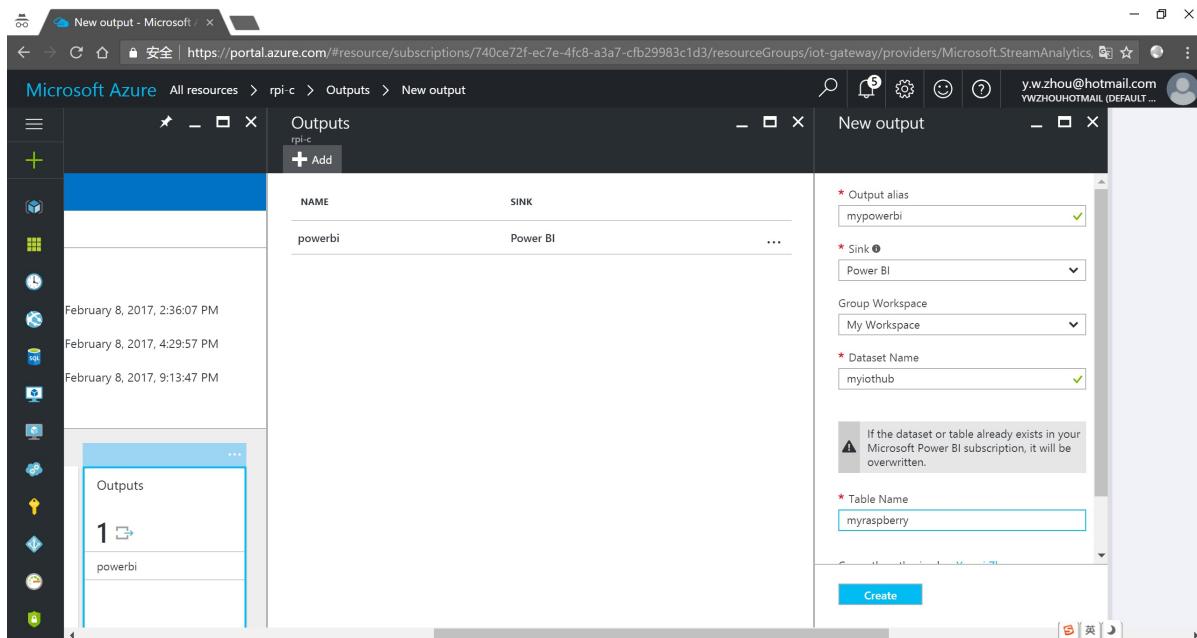
3. Click **Authorize**, and then sign into your Power BI account.
4. Once authorized, enter the following information:

Group Workspace: Select your target group workspace.

Dataset Name: Enter a dataset name.

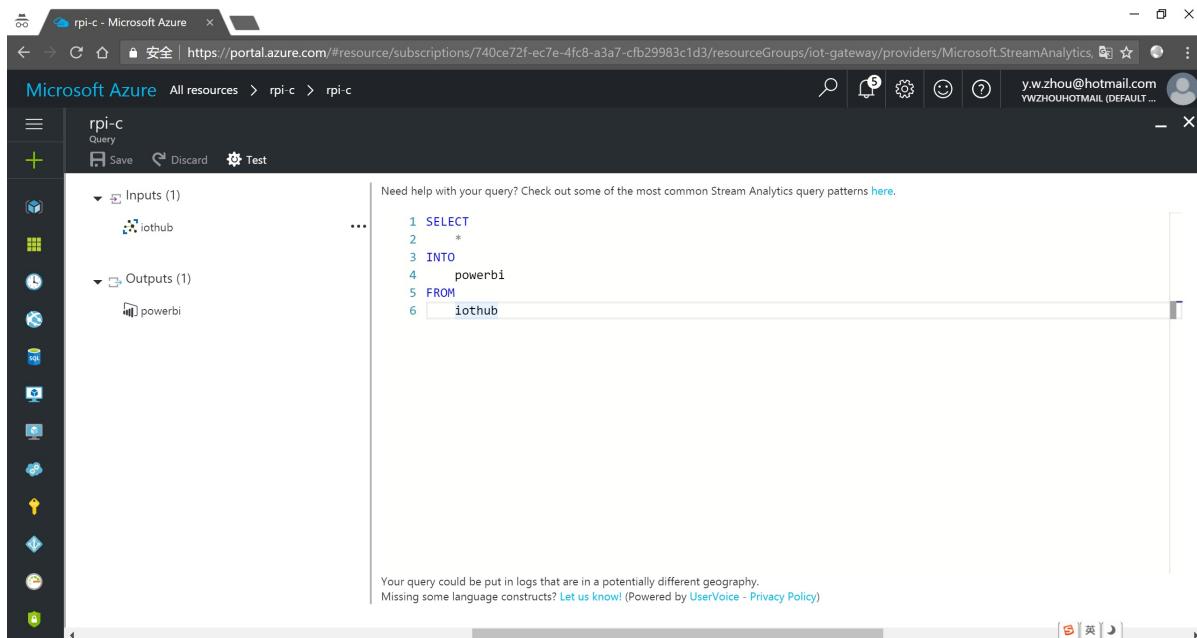
Table Name: Enter a table name.

5. Click **Create**.



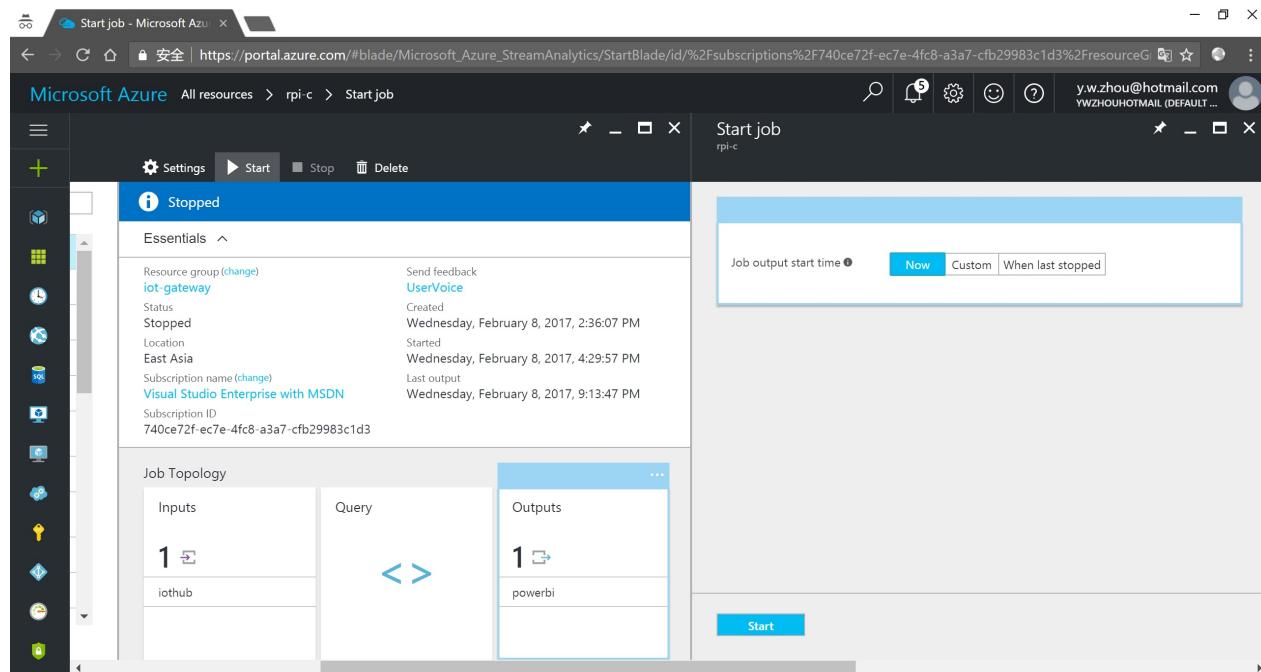
Configure the query of the Stream Analytics job

1. Under **Job Topology**, click **Query**.
2. Replace `[YourInputAlias]` with the input alias of the job.
3. Replace `[YourOutputAlias]` with the output alias of the job.
4. Click **Save**.



Run the Stream Analytics job

In the Stream Analytics job, click **Start > Now > Start**. Once the job successfully starts, the job status changes from **Stopped** to **Running**.



Create and publish a Power BI report to visualize the data

1. Ensure the sample application is running on your device. If not, you can refer to the tutorials under [Setup your device](#).
2. Sign in to your [Power BI](#) account.
3. Go to the group workspace that you set when you created the output for the Stream Analytics job.
4. Click **Streaming datasets**.

You should see the listed dataset that you specified when you created the output for the Stream Analytics job.

5. Under **ACTIONS**, click the first icon to create a report.

The screenshot shows the Power BI Datasets page. On the left, there's a sidebar with navigation links like Work Smart Details, Reports, and Datasets. Under Datasets, it says 'Streaming datasets' and 'No datasets found'. Below that is a yellow button labeled 'Get Data'. The main area is titled 'Streaming data' and contains a search bar. A table lists one dataset:

NAME	TYPE	USED IN DASHBOARDS	HISTORICAL	ACTIONS
yuwzho	API		Enabled	

6. Create a line chart to show real-time temperature over time.

- On the report creation page, add a line chart.
- On the **Fields** pane, expand the table that you specified when you created the output for the Stream Analytics job.
- Drag **EventEnqueuedUtcTime** to **Axis** on the **Visualizations** pane.
- Drag **temperature** to **Values**.

Now a line chart is created. The x-axis of chart displays date and time in the UTC time zone. The y-axis displays temperature from the sensor.

The screenshot shows the Power BI report creation page. The left sidebar has the same structure as before. The main area shows a line chart titled 'Temperature by EventEnqueuedUtcTime'. The chart has a single data series showing temperature values over time, with a sharp peak around 11:00 AM. To the right of the chart are the 'Visualizations' and 'Fields' panes. In the 'Fields' pane, under the 'yuwzho' table, the 'temperature' field is selected and placed in the 'Values' section of the visualizations pane. The 'EventEnqueuedUtcTime' field is also listed in the 'Axis' section.

7. Create another line chart to show real-time humidity over time. To do this, follow the same steps above and place **EventEnqueuedUtcTime** on the x-axis and **humidity** on the y-axis.

8. Click **Save** to save the report.
9. Click **File > Publish to web**.
10. Click **Create embed code**, and then click **Publish**.

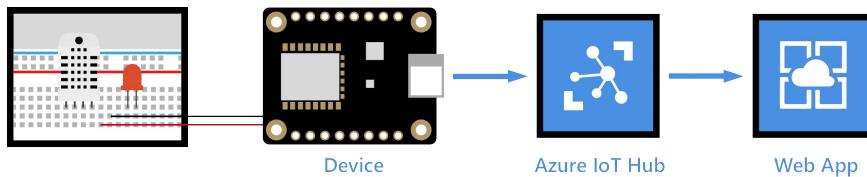
You're provided the report link that you can share with anyone for report access and a code snippet to integrate the report into your blog or website.

Microsoft also offers the [Power BI mobile apps](#) for viewing and interacting with your Power BI dashboards and reports on your mobile device.

Next steps

You've successfully used Power BI to visualize real-time sensor data from your Azure IoT hub. There is an alternate way to visualize data from Azure IoT Hub. See [Use Azure Web Apps to visualize real-time sensor data from Azure IoT Hub](#).

Visualize real-time sensor data from Azure IoT Hub using Azure Web Apps



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

What you learn

In this lesson, you learn how to visualize real-time sensor data that your Azure IoT hub receives by running a web application that is hosted on an Azure web app. If you want to try visualize the data in your IoT hub with Power BI, please see [Use Power BI to visualize real-time sensor data from Azure IoT Hub](#).

What you do

- Create an Azure web app in the Azure Portal.
- Get your IoT hub ready for data access by adding a consumer group.
- Configure the web app to read sensor data from your IoT hub.
- Upload a web application to be hosted by the web app.
- Open the web app to see real-time temperature and humidity data from your IoT hub.

What you need

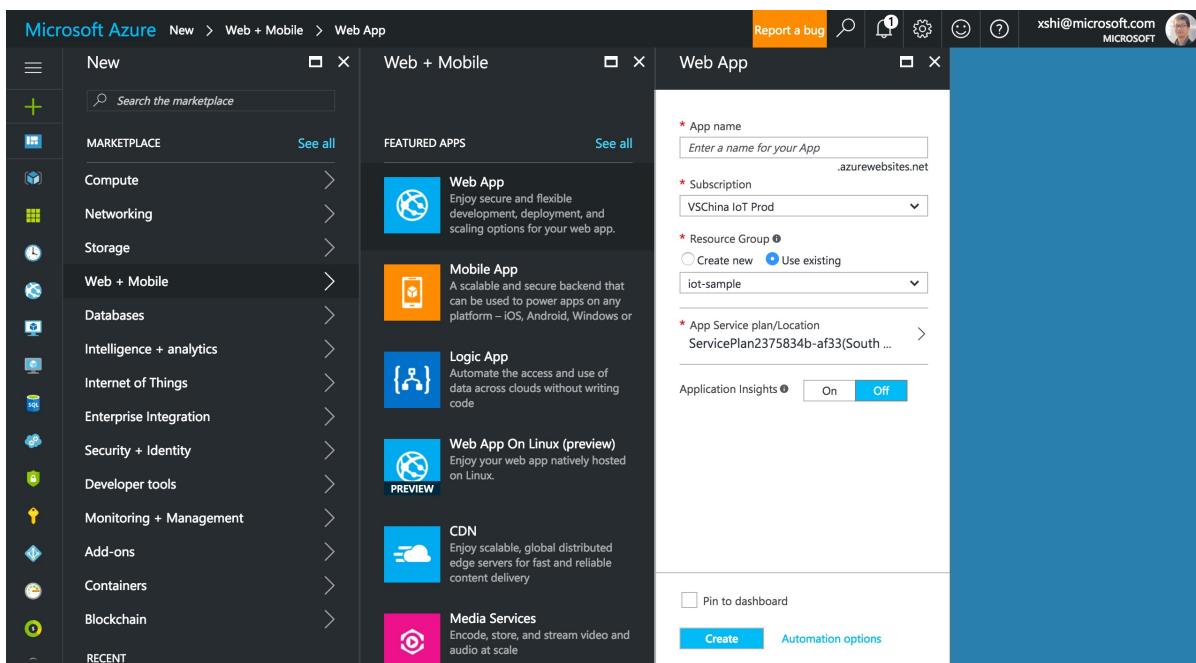
- Tutorial [Setup your device](#) completed which covers the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.

- A client application that sends messages to your Azure IoT hub.
- Git. ([Download Git](#)).

Create an Azure web app

1. In the [Azure portal](#), click **New > Web + Mobile > Web App**.
2. Enter a unique job name, verify the subscription, specify a resource group and a location, select **Pin to dashboard**, and then click **Create**.

We recommend you select the same location where your resource group is located. This assists with processing speed and reduced costs for data transfer.



Add a consumer group to your IoT hub

Consumer groups are used by applications to pull data from Azure IoT Hub. In this lesson, you create a consumer group to be used by a coming Azure service to read data from your IoT hub.

To add a consumer group to your IoT hub, follow these steps:

1. In the [Azure portal](#), open your IoT hub.
2. Click **Endpoints** on the left pane, select **Events** on the middle pane, enter a name under **Consumer groups** on the right pane, and then click **Save**.

The screenshot shows the Microsoft Azure IoT Hub - Endpoints Properties page. On the left, there's a navigation sidebar with various icons and links like 'Enterprise...', 'File upload', 'Endpoints', and 'Routes'. The main area has two sections: 'Built-in endpoints' and 'Custom endpoints'. Under 'Built-in endpoints', there are two entries: 'Cloud to device feedback' (endpoint: messages/servicebound/feedback) and 'Events' (endpoint: messages/events). Under 'Custom endpoints', it says 'You may have up to 1 endpoint on this IoT hub.' and shows a table with one entry: '\$Default' (endpoint type: sb://ihsuprodbyres030dednamespace.send). A right-hand sidebar titled 'Device-to-cloud settings' lists several items, including 'iothub-ehub-mygateway-92679-259c382' and 'sb://ihsuprodbyres030dednamespace.send'.

Configure the web app to read data from your IoT hub

1. Open the Web app you've just provisioned.
2. Click **Application settings**, and then add the following key/value pairs under **App settings**: see the table at the end of the steps.

The screenshot shows the Microsoft Azure xshi - Application settings page. The left sidebar includes 'Overview', 'SETTINGS' (with 'Application settings' selected), 'Authentication / Authorization', 'Backups', 'Custom domains', 'SSL certificates', 'Networking', 'Scale up (App Service plan)', 'Scale out (App Service plan)', 'Security scanning', 'Webjobs', 'Push', 'MySQL In App (preview)', 'Properties', and 'Locks'. The main area has sections for 'Debugging' (Remote debugging: Off/On, Remote Visual Studio version: 2014/2013/2015), 'App settings' (WEBSITE_NODE_DEFAULT_VERSION: 6.9.1, Azure.IoT.IoTHub.ConnectionString, Azure.IoT.IoTHub.ConsumerGroup), 'Connection strings' (No results), 'Default documents' (Default.htm, Default.html), and a 'Slot setting' section. The 'Azure.IoT.IoTHub.ConnectionString' and 'Azure.IoT.IoTHub.ConsumerGroup' settings are highlighted in blue.

3. In **Application settings**, toggle the Web sockets option under General settings.

The screenshot shows the 'Application settings' section of the Azure portal. It includes fields for .NET Framework version (v4.6), PHP version (5.6), Java version (Off), Python version (Off), Platform (32-bit selected), Web sockets (On selected), Always On (On selected), and Managed Pipeline Version (Integrated selected).

Key	Value
Azure.IoT.IoTHub.ConnectionString	Obtained from iothub-explorer
Azure.IoT.IoTHub.DeviceId	Obtained from iothub-explorer
Azure.IoT.IoTHub.ConsumerGroup	The name of the consumer group that you add to your IoT hub

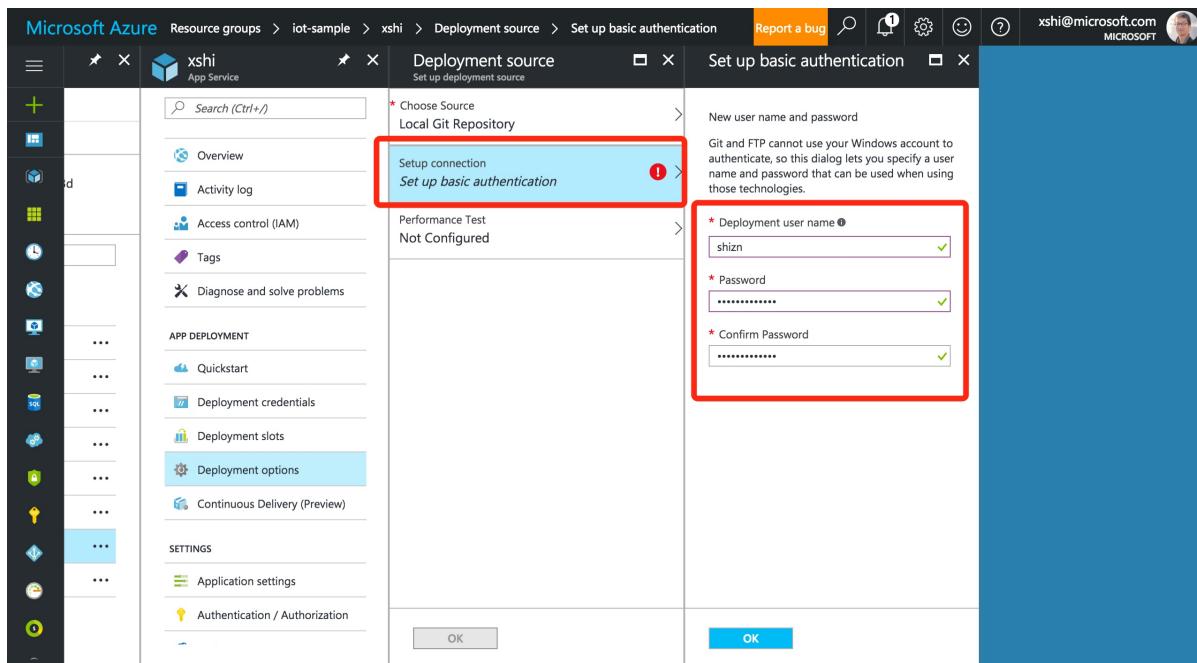
Upload a web application to be hosted by the web app

We made available a web application on GitHub which displays real-time sensor data from your IoT hub. All you need to do is to configure the web app to work with a Git repository, download the web application from GitHub and upload it to Azure for the web app to host.

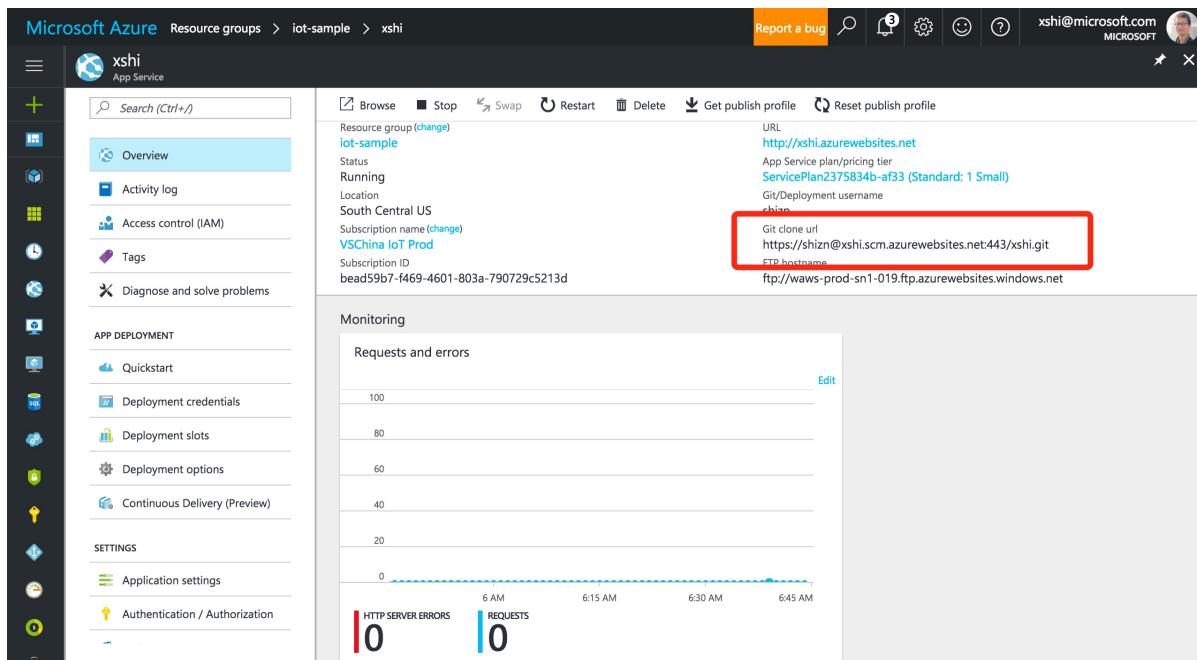
1. In the web app, click **Deployment Options > Choose Source > Local Git Repository**.

The screenshot shows the 'Deployment source' configuration screen. It highlights the 'Choose source' dialog where 'Local Git Repository' is selected. The main pane shows deployment options like 'Setup connection' (Set up basic authentication) and 'Performance Test' (Not Configured). The left sidebar shows 'Deployment options' and 'Continuous Delivery (Preview)' under 'APP DEPLOYMENT'.

- Click **Setup connection**, create a user name and password that will be used to connect to the Git repository in Azure, and then click **OK**.



- Click **OK** to finish the configuration.
- Click **Overview** and make a note of the value of **Git clone url**.



- Open a command or terminal window on your local computer.
- Download the web application from GitHub and upload it to Azure for the web app to host. To do this, run the following commands:

```
git clone https://github.com/Azure-Samples/web-apps-node-iot-hub-data-visualization.git
cd web-apps-node-iot-hub-data-visualization
git remote add webapp <Git clone URL>
git push webapp master:master
```

[!Note] \ is the URL of the Git repository found on the **Overview** page of the web app.

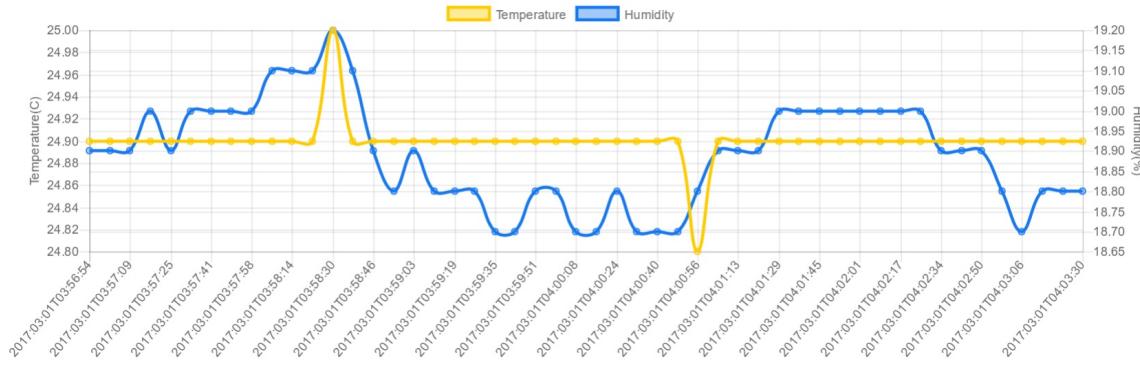
Open the web app to see real-time temperature and humidity data from your IoT hub

On the **Overview** page of your web app, click the URL to open the web app.

The screenshot shows the Azure portal interface for an App Service named 'xshi'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, APP DEPLOYMENT, SETTINGS, and Custom domains. The main panel displays the 'Overview' tab, which includes sections for Resource group (iot-sample), Status (Running), Location (South Central US), Subscription name (VSChina IoT Prod), and Subscription ID (bead59b7-f469-4601-803a-790729c5213d). A prominent red box highlights the 'URL' field, which contains the value 'http://xshi.azurewebsites.net'. Below this, a chart titled 'Requests and errors' shows two data series: 'HTTP SERVER ERRORS' and 'REQUESTS', both of which are currently at 0. The chart has time markers for 11:15 AM, 11:30 AM, 11:45 AM, and 12 PM.

You should see the real-time temperature and humidity data from your IoT hub.

Temperature & Humidity Real-time Data



Next steps

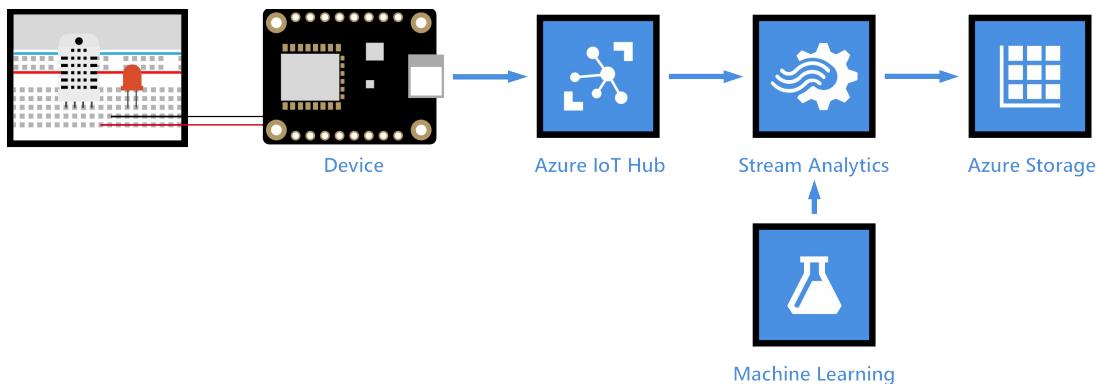
You've successfully used an Azure web app to visualize real-time sensor data from your Azure IoT hub.

There is an alternate way to visualize data from Azure IoT Hub. See [Use Power BI to visualize real-time sensor data from Azure IoT Hub](#).

Next steps

You've successfully used Power BI to visualize real-time sensor data from your Azure IoT hub. There is an alternate way to visualize data from Azure IoT Hub.

Weather forecast using the sensor data from your IoT hub in Azure Machine Learning



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

Machine learning is a technique of data science that helps computers learn from existing data to forecast future behaviors, outcomes, and trends. Azure Machine Learning is a cloud predictive analytics service that makes it possible to quickly create and deploy predictive models as analytics solutions.

What you learn

You learn how to use Azure Machine Learning to do weather forecast (chance of rain) using the temperature and humidity data from your Azure IoT hub. The chance of rain is the output of a prepared weather prediction model. The model is built upon historic data to forecast chance of rain based on temperature and humidity.

What you do

- Deploy the weather prediction model as a web service.
- Get your IoT hub ready for data access by adding a consumer group.
- Create a Stream Analytics job and configure the job to:
 - Read temperature and humidity data from your IoT hub.
 - Call the web service to get the rain chance.

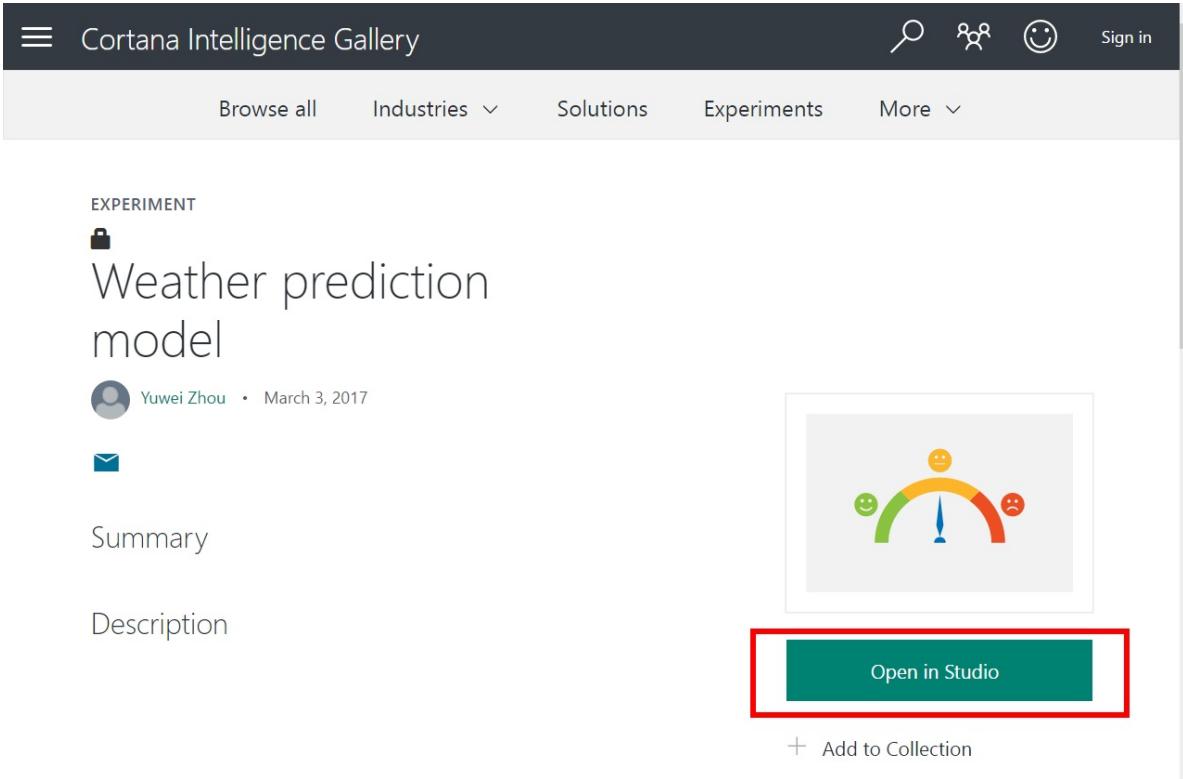
- Save the result to an Azure blob storage.
- Use Microsoft Azure Storage Explorer to view the weather forecast.

What you need

- Tutorial [Setup your device](#) completed which covers the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.
 - A client application that sends messages to your Azure IoT hub.
- An Azure Machine Learning Studio account. ([Try Machine Learning Studio for free](#)).

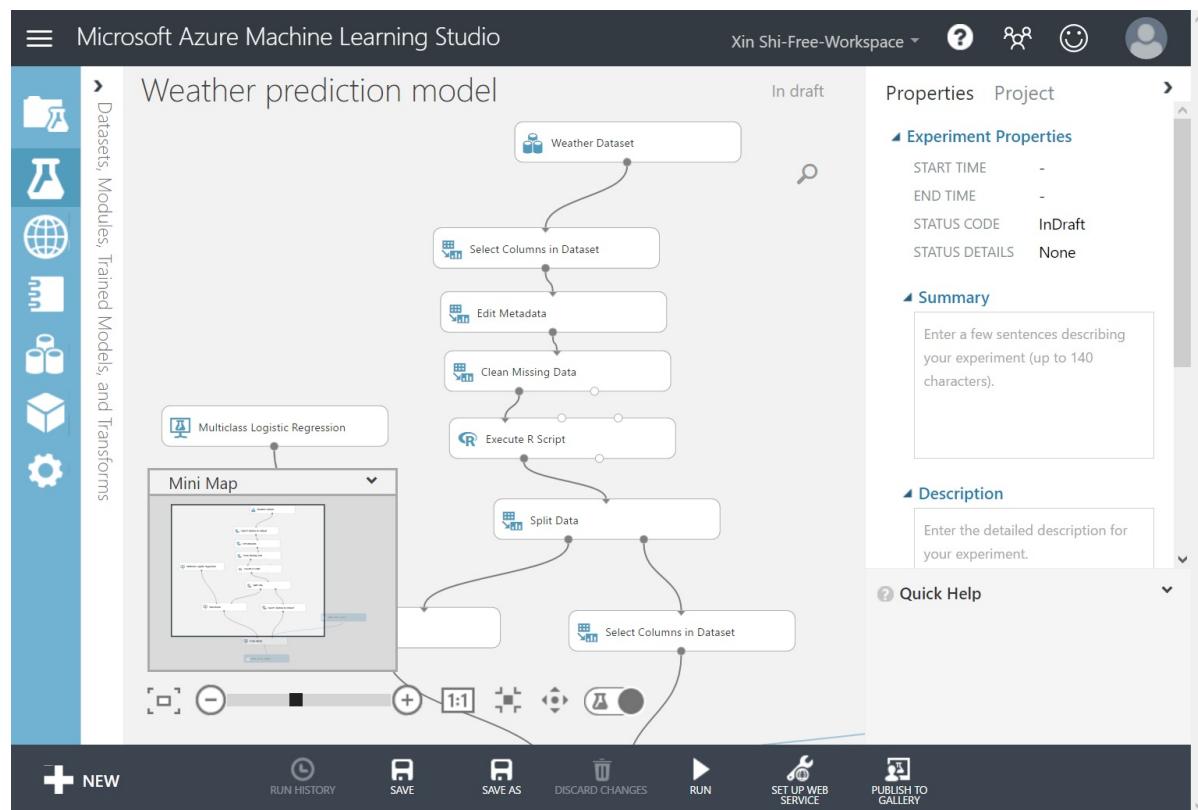
Deploy the weather prediction model as a web service

1. Go to the [weather prediction model page](#).
2. Click **Open in Studio** in Microsoft Azure Machine Learning Studio.

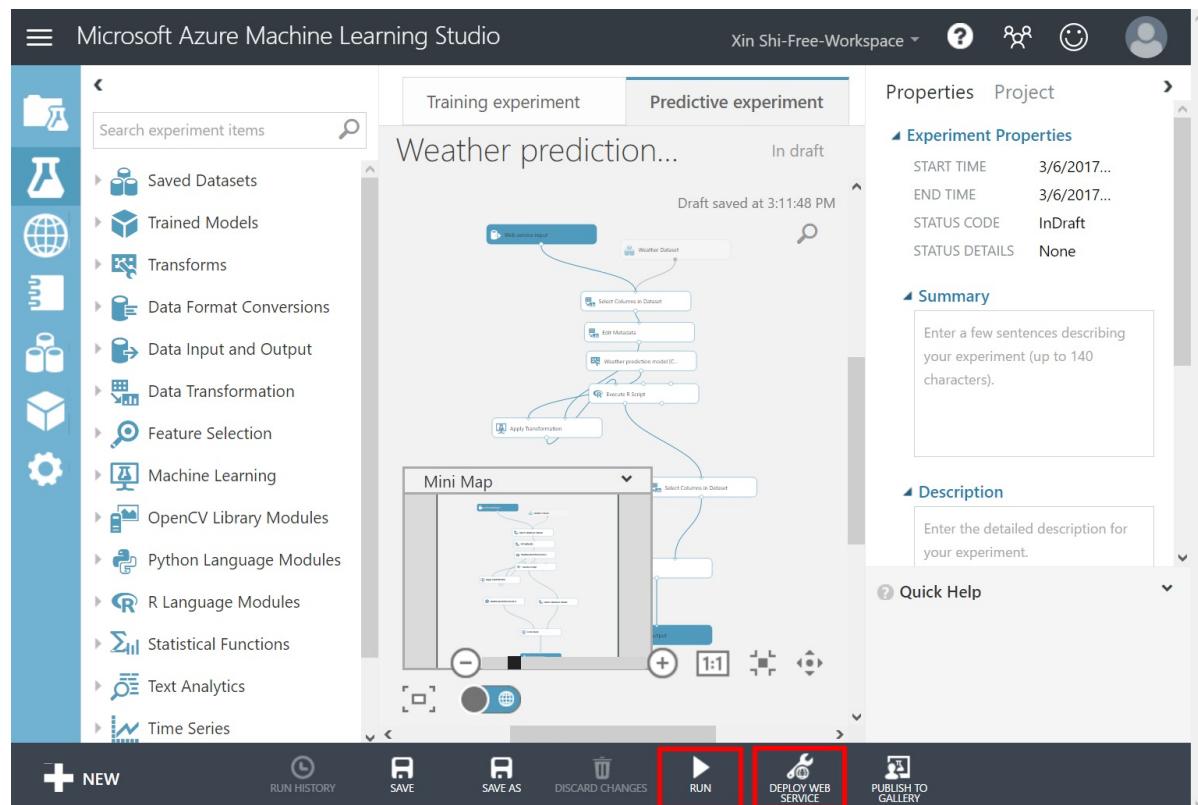


The screenshot shows the Cortana Intelligence Gallery interface. At the top, there's a navigation bar with a menu icon, the text "Cortana Intelligence Gallery", search, user, and sign-in options. Below the navigation bar, there are filters for "Browse all", "Industries", "Solutions", "Experiments", and "More". The main content area is titled "EXPERIMENT" and features a "Weather prediction model" card. The card includes a lock icon, the author's profile picture and name ("Yuwei Zhou • March 3, 2017"), a summary section with a mail icon, and a description section. To the right of the card is a small graphic of a smiley face on a meter scale. At the bottom of the card is a green "Open in Studio" button, which is highlighted with a red rectangular border. Below the card, there are "Add to Collection" and other interaction buttons.

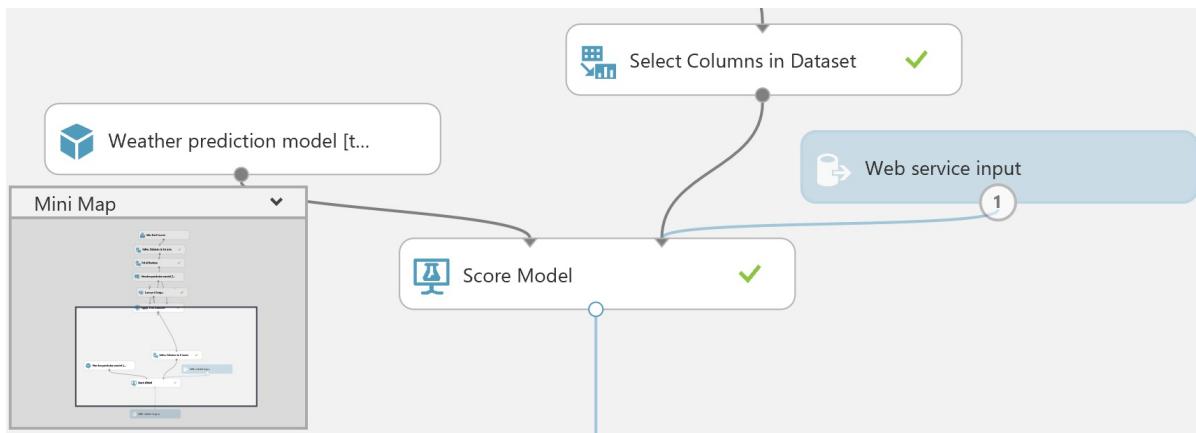
3. Click **Run** to validate the steps in the model. This step might take 2 minutes to complete.



4. Click **SET UP WEB SERVICE > Predictive Web Service**.



5. In the diagram, drag the **Web service input** module somewhere near the **Score Model** module.
6. Connect the **Web service input** module to the **Score Model** module.



7. Click **RUN** to validate the steps in the model.
8. Click **DEPLOY WEB SERVICE** to deploy the model as a web service.
9. On the dashboard of the model, download the **Excel 2010 or earlier workbook** for **REQUEST/RESPONSE**.

[!Note] Ensure that you download the **Excel 2010 or earlier workbook** even if you are running a later version of Excel on your computer.

REQUEST/RESPONSE	TEST	Test preview	Excel 2013 or later Excel 2010 or earlier	LAST UPDATED
BATCH EXECUTION	Test preview		Excel 2013 or later workbook	3/6/2017 3:14:25 PM

10. Open the Excel workbook, make a note of the **WEB SERVICE URL** and **ACCESS KEY**.

Add a consumer group to your IoT hub

Consumer groups are used by applications to pull data from Azure IoT Hub. In this lesson, you create a consumer group to be used by a coming Azure service to read data from your IoT hub.

To add a consumer group to your IoT hub, follow these steps:

1. In the [Azure portal](#), open your IoT hub.
2. Click **Endpoints** on the left pane, select **Events** on the middle pane, enter a name under **Consumer groups** on the right pane, and then click **Save**.

Create, configure, and run a Stream Analytics job

Create a Stream Analytics job

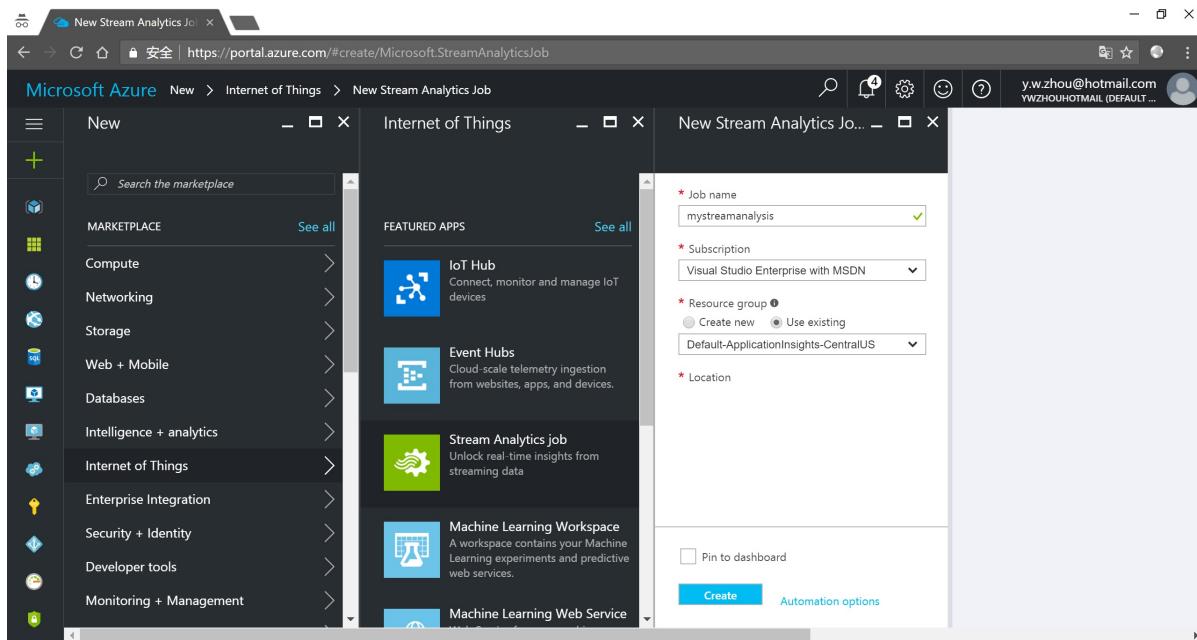
1. In the [Azure portal](#), click **New > Internet of Things > Stream Analytics job**.
2. Enter the following information for the job.

Job name: The name of the job. The name must be globally unique.

Resource group: Use the same resource group that your IoT hub uses.

Location: Use the same location as your resource group.

Pin to dashboard: Check this option for easy access to your IoT hub from the dashboard.



3. Click **Create**.

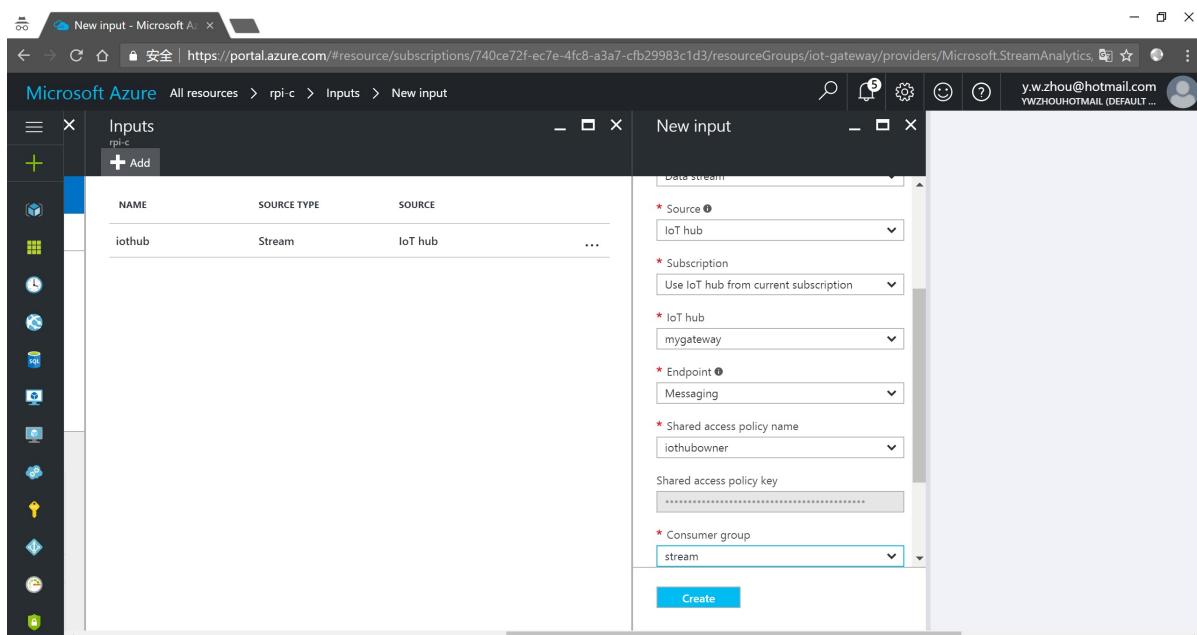
Add an input to the Stream Analytics job

1. Open the Stream Analytics job.
2. Under **Job Topology**, click **Inputs**.
3. In the **Inputs** pane, click **Add**, and then enter the following information:

Input alias: The unique alias for the input.

Source: Select **IoT hub**.

Consumer group: Select the consumer group you created.



4. Click **Create**.

Add an output to the Stream Analytics job

- Under **Job Topology**, click **Outputs**.
- In the **Outputs** pane, click **Add**, and then enter the following information:

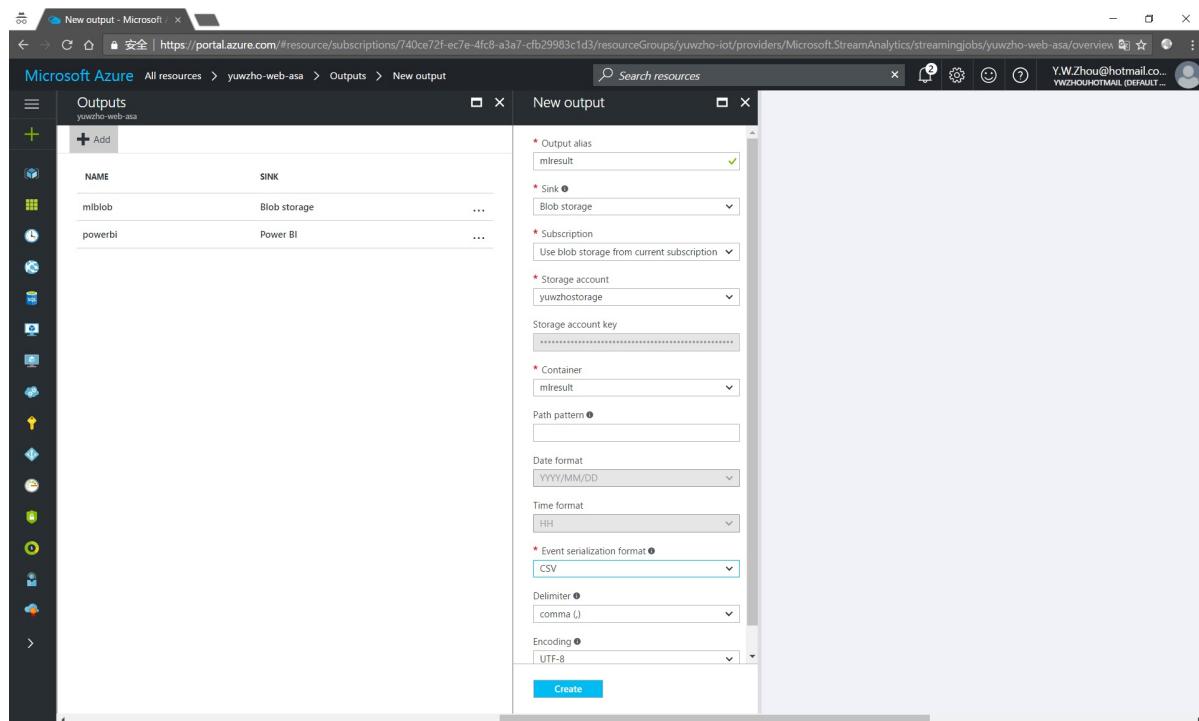
Output alias: The unique alias for the output.

Sink: Select **Blob Storage**.

Storage account: The storage account for your blob storage. You can create a storage account or use an existing one.

Container: The container where the blob is saved. You can create a container or use an existing one.

Event serialization format: Select **CSV**.



- Click **Create**.

Add a function to the Stream Analytics job to call the web service you deployed

- Under **Job Topology**, click **Functions > Add**.
- Enter the following information:

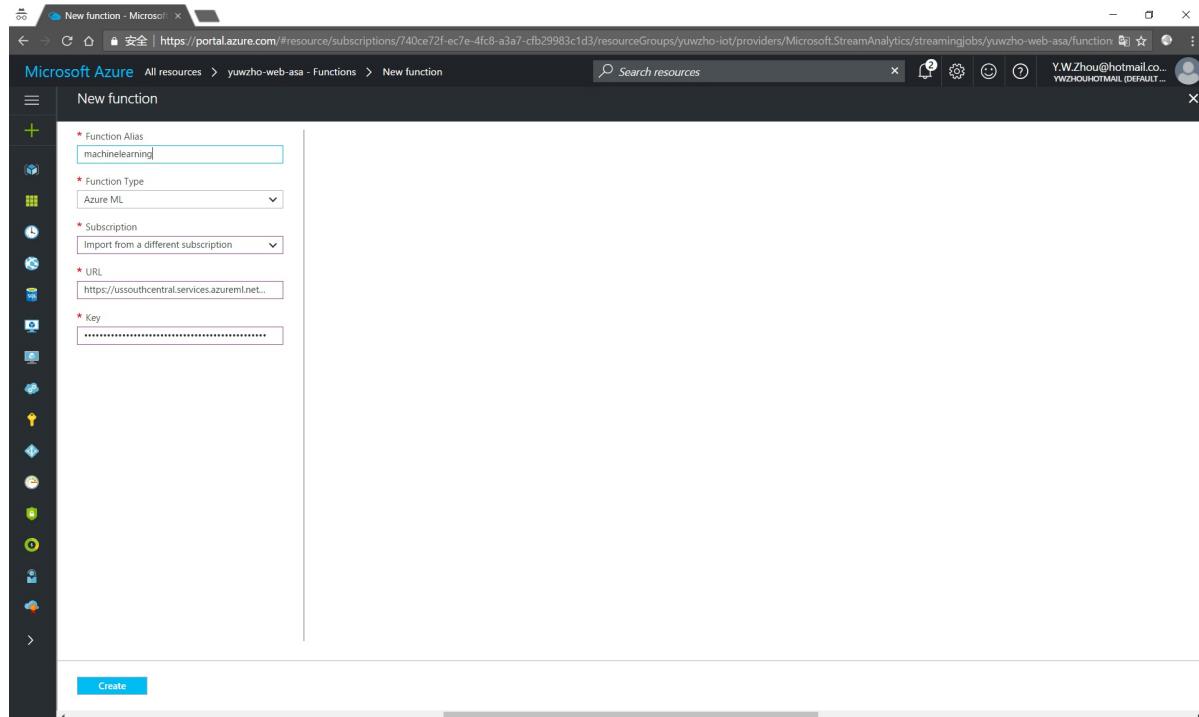
Function Alias: Enter `machinelearning`.

Function Type: Select **Azure ML**.

Import option: Select **Import from a different subscription**.

URL: Enter the WEB SERVICE URL that you noted down from the Excel workbook.

Key: Enter the ACCESS KEY that you noted down from the Excel workbook.



3. Click **Create**.

Configure the query of the Stream Analytics job

1. Under **Job Topology**, click **Query**.
2. Replace the existing code with the following code:

```
WITH machinelearning AS (
    SELECT EventEnqueuedUtcTime, temperature, humidity, machinelearning(temperature
    , humidity) as result from [YourInputAlias]
)
Select System.Timestamp time, CAST (result.[temperature] AS FLOAT) AS temperature,
CAST (result.[humidity] AS FLOAT) AS humidity, CAST (result.[Scored Probabilities]
AS FLOAT ) AS 'probabalities of rain'
Into [YourOutputAlias]
From machinelearning
```

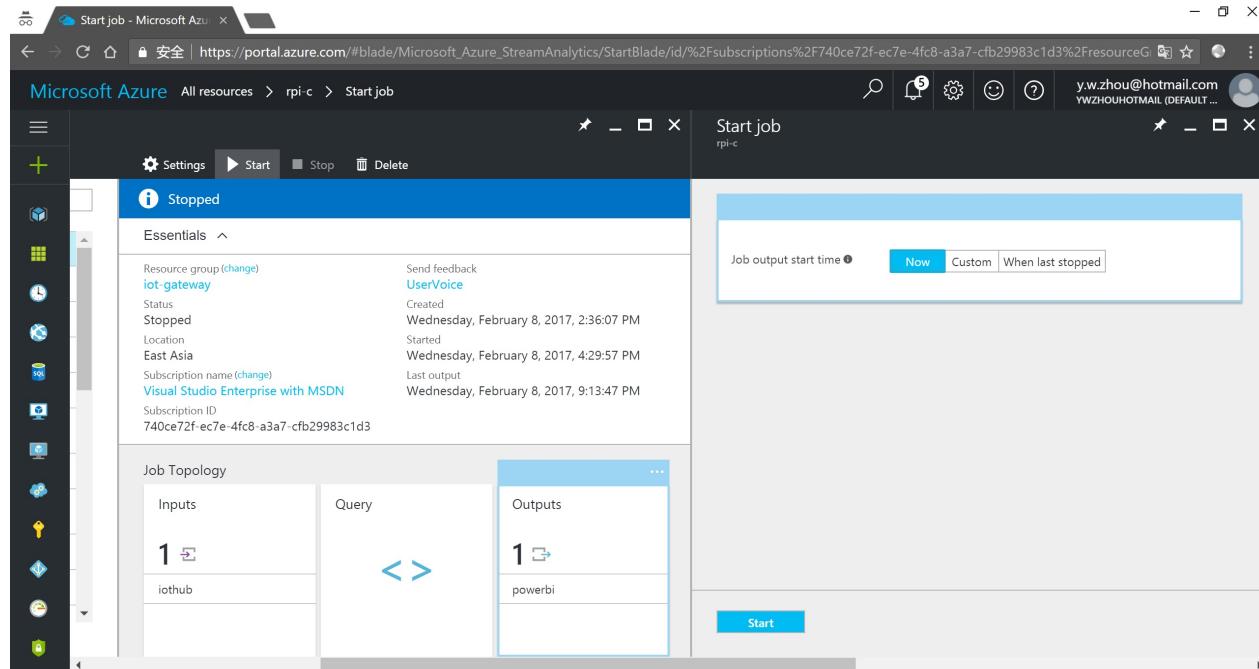
Replace `[YourInputAlias]` with the input alias of the job.

Replace `[YourOutputAlias]` with the output alias of the job.

3. Click **Save**.

Run the Stream Analytics job

In the Stream Analytics job, click **Start > Now > Start**. Once the job successfully starts, the job status changes from **Stopped** to **Running**.



Use Microsoft Azure Storage Explorer to view the weather forecast

Run the client application to start collecting and sending temperature and humidity data to your IoT hub. For each message that your IoT hub receives, the Stream Analytics job calls the weather forecast web service to produce the chance of rain. The result is then saved to your Azure blob storage. Azure Storage Explorer is a tool that you can use to view the result.

1. [Download and install Microsoft Azure Storage Explorer](#).
2. Open Azure Storage Explorer.
3. Sign in to your Azure account.
4. Select your subscription.
5. Click your subscription > **Storage Accounts** > your storage account > **Blob Containers** > your container.
6. Open a .csv file to see the result. The last column records the chance of rain.

2119909009_9e9dee549b224c1e8866be1f5b31dd92_1.csv - Notepad

File Edit Format View Help

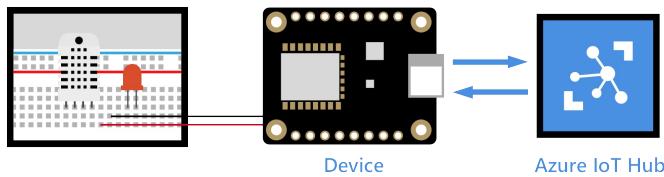
time,temperature,humidity,probabalities of rain

2017-03-07T02:16:35.30000Z,24.6,20.2,0.506996631622314
2017-03-07T02:16:56.8740000Z,25.3,13.3,0.506996631622314
2017-03-07T02:17:04.8420000Z,25.2,13.2,0.506996631622314
2017-03-07T02:17:13.1540000Z,25.2,13.4,0.506996631622314
2017-03-07T02:17:22.6300000Z,25.1,13.3,0.506996631622314
2017-03-07T02:17:30.8280000Z,25.1,13.6,0.506996631622314
2017-03-07T02:17:38.8890000Z,25,13.9,0.506996631622314
2017-03-07T02:17:54.1470000Z,25,13.7,0.506996631622314
2017-03-07T02:18:03.2240000Z,24.9,13.3,0.506996631622314
2017-03-07T02:18:10.7570000Z,24.9,15.4,0.506996631622314
2017-03-07T02:18:20.3880000Z,24.9,14.2,0.506996631622314
2017-03-07T02:18:29.2600000Z,24.9,14.6,0.506996631622314
2017-03-07T02:18:38.9710000Z,24.8,14.9,0.506996631622314
2017-03-07T02:18:48.0020000Z,24.8,14.2,0.506996631622314
2017-03-07T02:18:56.2520000Z,24.8,14.7,0.506996631622314
2017-03-07T02:19:04.1800000Z,24.8,15.3,0.506996631622314
2017-03-07T02:19:12.9650000Z,24.8,14.3,0.506996631622314
2017-03-07T02:19:21.8240000Z,24.8,13.8,0.506996631622314
2017-03-07T02:19:31.2290000Z,24.8,13.6,0.506996631622314
2017-03-07T02:19:39.3300000Z,24.8,14.1,0.506996631622314
2017-03-07T02:19:47.2860000Z,24.8,13.6,0.506996631622314
2017-03-07T02:20:13.7590000Z,24.8,12.7,0.506996631622314
2017-03-07T02:20:21.3560000Z,24.7,13.4,0.517819762229919
2017-03-07T02:20:29.1680000Z,24,7,12,1,0.517819762229919

Summary

You've successfully used Azure Machine Learning to produce the chance of rain based on the temperature and humidity data that your IoT hub receives.

Use iothub-explorer for Azure IoT Hub device management



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

[iothub-explorer](#) is a CLI tool that you run on a host computer to manage device identities in your IoT hub registry. It comes with management options that you can use to perform various tasks.

Management option	Task
Direct methods	Make a device act such as starting or stopping sending messages or rebooting the device.
Twin desired properties	Put a device into certain states, such as setting an LED to green or setting the telemetry send interval to 30 minutes.
Twin reported properties	Get the reported state of a device. For example, the device reports the LED is blinking now.
Twin tags	Store device-specific metadata in the cloud. For example, the deployment location of a vending machine.
Cloud-to-device messages	Send notifications to a device. For example, "It is very likely to rain today. Don't forget to bring an umbrella."
Device twin queries	Query all device twins to retrieve those with arbitrary conditions, such as identifying the devices that are available for use.

For more detailed explanation on the differences and guidance on using these options, see [Device-to-cloud communication guidance](#) and [Cloud-to-device communication guidance](#).

[!NOTE] Device twins are JSON documents that store device state information (metadata, configurations, and conditions). IoT Hub persists a device twin for each device that connects to it. For more information about device twins, see [Get started with device twins](#).

What you learn

You learn using iothub-explorer with various management options on your development machine.

What you do

Run iothub-explorer with various management options.

What you need

- Tutorial [Setup your device](#) completed which covers the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.
 - A client application that sends messages to your Azure IoT hub.
- iothub-explorer. ([Install iothub-explorer](#) on your development machine)

Connect to your IoT hub

Connect to your IoT hub by running the following command:

```
iothub-explorer login <your IoT hub connection string>
```

Use iothub-explorer with direct methods

Invoke the `start` method in the device app to send messages to your IoT hub by running the following command:

```
iothub-explorer device-method <your device Id> start
```

Invoke the `stop` method in the device app to stop sending messages to your IoT hub by running the following command:

```
iothub-explorer device-method <your device Id> stop
```

Use iothub-explorer with twin's desired properties

Set a desired property interval = 3000 by running the following command:

```
iothub-explorer update-twin <your device id> {"properties": {"desired": {"interval": 3000}}}
```

This property can be read by your device.

Use iothub-explorer with twin's reported properties

Get the reported properties of the device by running the following command:

```
iothub-explorer get-twin <your device id>
```

One of the properties is `$metadata.$lastUpdated` which shows the last time this device sends or receives a message.

Use iothub-explorer with twin's tags

Display the tags and properties of the device by running the following command:

```
iothub-explorer get-twin <your device id>
```

Add a field role = temperature&humidity to the device by running the following command:

```
iothub-explorer update-twin <your device id> {"tags": {"role": "temperature&humidity"}}
```

Use iothub-explorer with Cloud-to-device messages

Send a "Hello World" message to the device by running the following command:

```
iothub-explorer send <device-id> "Hello World"
```

See [Use iothub-explorer to send and receive messages between your device and IoT Hub](#) for a real scenario of using this command.

Use iothub-explorer with device twins queries

Query devices with a tag of role = 'temperature&humidity' by running the following command:

```
iothub-explorer query-twin "SELECT * FROM devices WHERE tags.role = 'temperature&humidity'"
```

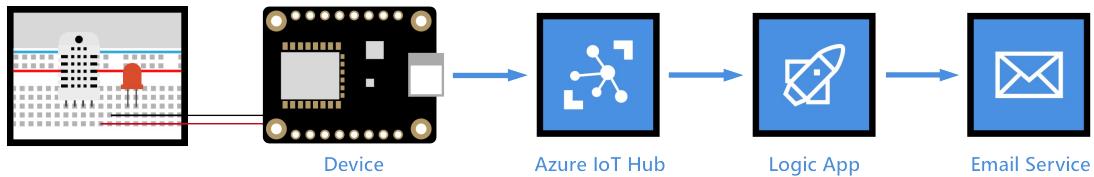
Query all devices except those with a tag of role = 'temperature&humidity' by running the following command:

```
iothub-explorer query-twin "SELECT * FROM devices WHERE tags.role != 'temperature&humidity'"
```

Next steps

You've learned how to use iothub-explorer with various management options.

IoT remote monitoring and notifications with Azure Logic Apps connecting your IoT hub and mailbox



[!NOTE] Before you start this tutorial, make sure you've completed Setup your device. By Setup your device, you set up your IoT device and IoT hub, and deploy a sample application to run on your device. The application sends collected sensor data to your IoT hub.

Azure Logic Apps provides a way to automate processes as a series of steps. A logic app can connect across various services and protocols. It begins with a trigger such as 'When an account is added', and followed by a combination of actions, one like 'sending a push notification'. This feature makes Logic Apps a perfect IoT solution for IoT monitoring, such as staying alert for anomalies, among other usage scenarios.

What you learn

You learn how to create a logic app that connects your IoT hub and your mailbox for temperature monitoring and notifications. When the temperature is above 30 C, the client application marks `temperatureAlert = "true"` in the message it sends to your IoT hub. The message triggers the logic app to send you an email notification.

What you do

- Create a service bus namespace and add a queue to it.
- Add an endpoint and a routing rule to your IoT hub.
- Create, configure, and test a logic app.

What you need

- Tutorial [Setup your device](#) completed which covers the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.
 - A client application that sends messages to your Azure IoT hub.

Create service bus namespace and add a queue to it

Create a service bus namespace

1. On the [Azure portal](#), click **New > Enterprise Integration > Service Bus**.
2. Provide the following information:

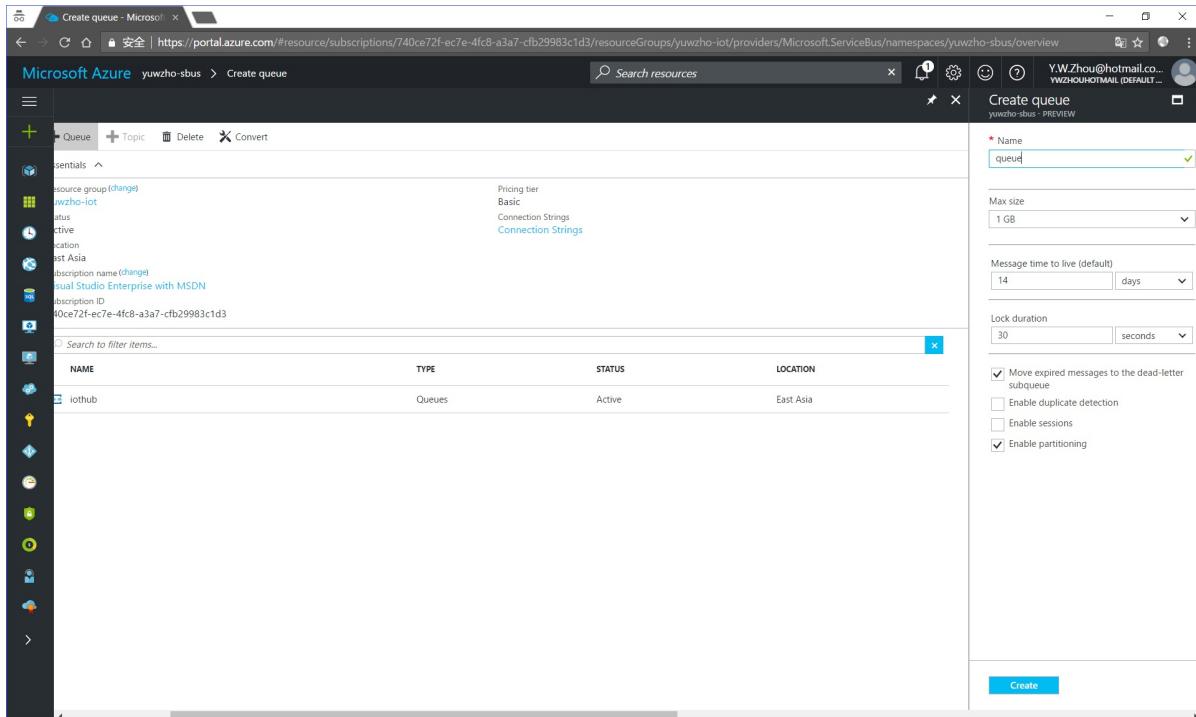
Name: The name of the service bus.

Pricing tier: Click **Basic** > **Select**. The Basic tier is sufficient for this tutorial.

Resource group: Use the same resource group that your IoT hub uses.

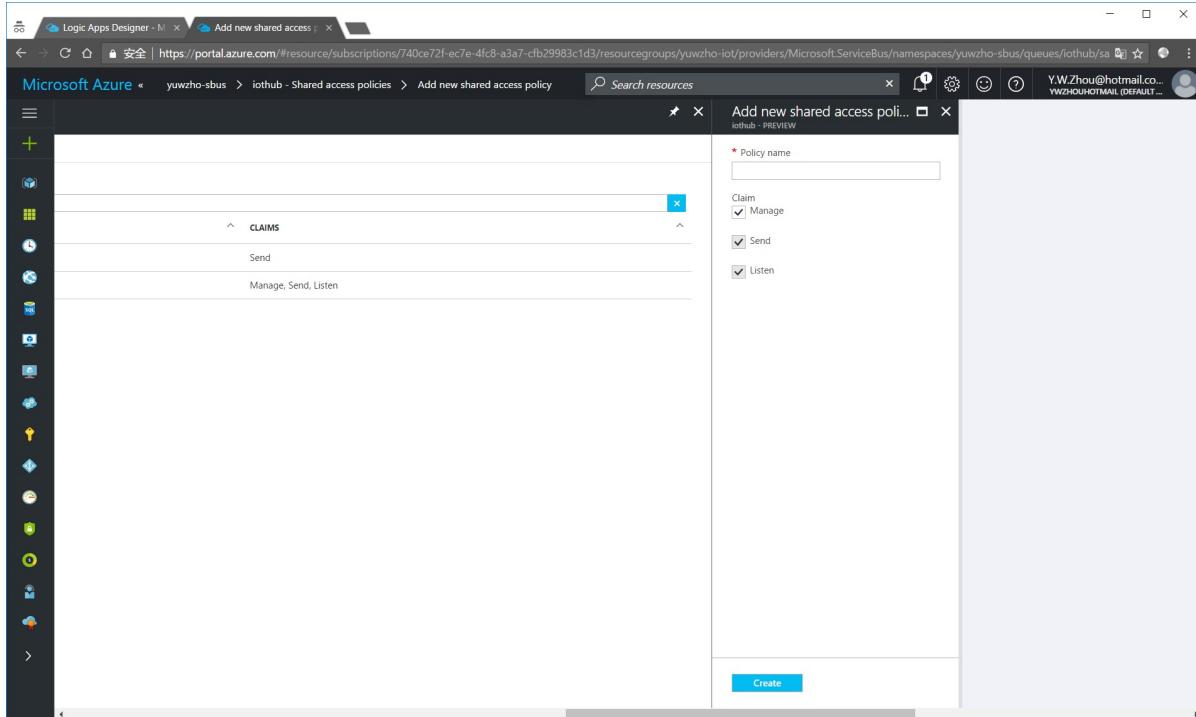
Location: Use the same location that your IoT hub uses.

3. Click **Create**.



Add a service bus queue

1. Open the service bus namespace, and then click **+ Queue**.
2. Enter a name for the queue and then click **Create**.
3. Open the service bus queue, and then click **Shared access policies > + Add**.
4. Enter a name for the policy, check **Manage**, and then click **Create**.



Add an endpoint and a routing rule to your IoT hub

Add an endpoint

1. Open your IoT hub, click Endpoints > + Add.
2. Enter the following information:

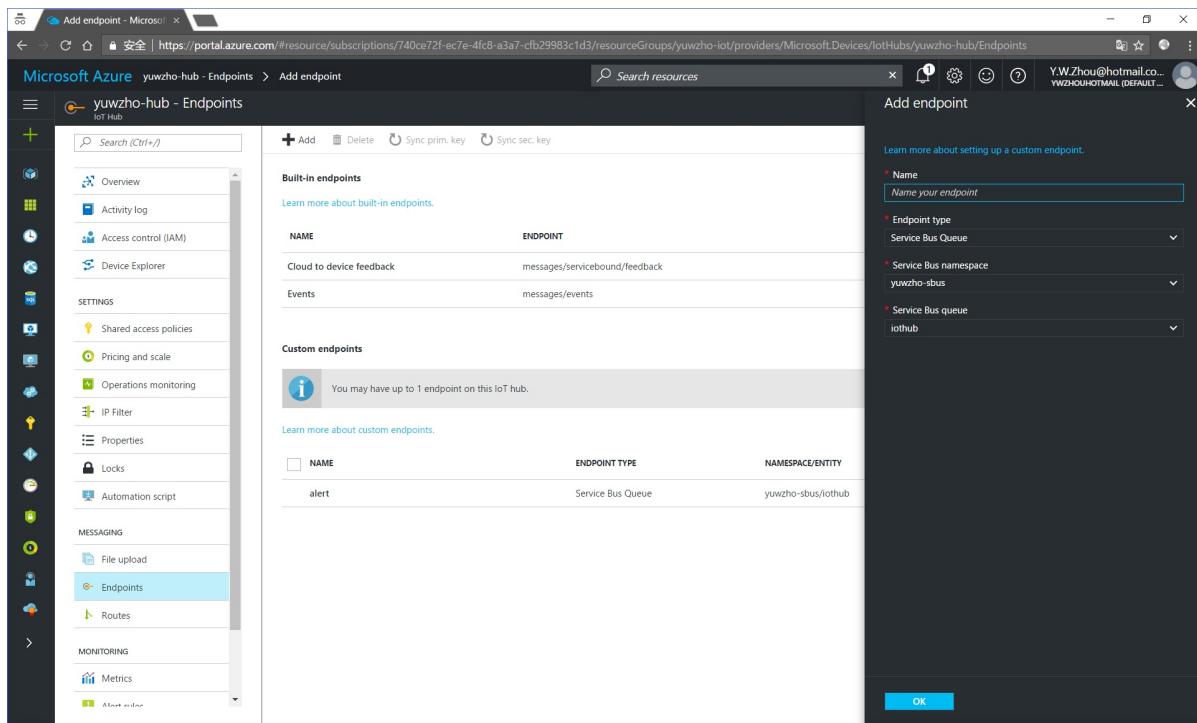
Name: The name of the endpoint.

Endpoint type: Select **Service Bus Queue**.

Service Bus namespace: Select the namespace you created.

Service Bus queue: Select the queue you created.

3. Click **OK**.



Add a routing rule

1. In your IoT hub, click **Routes** > **+ Add**.
2. Enter the following information:

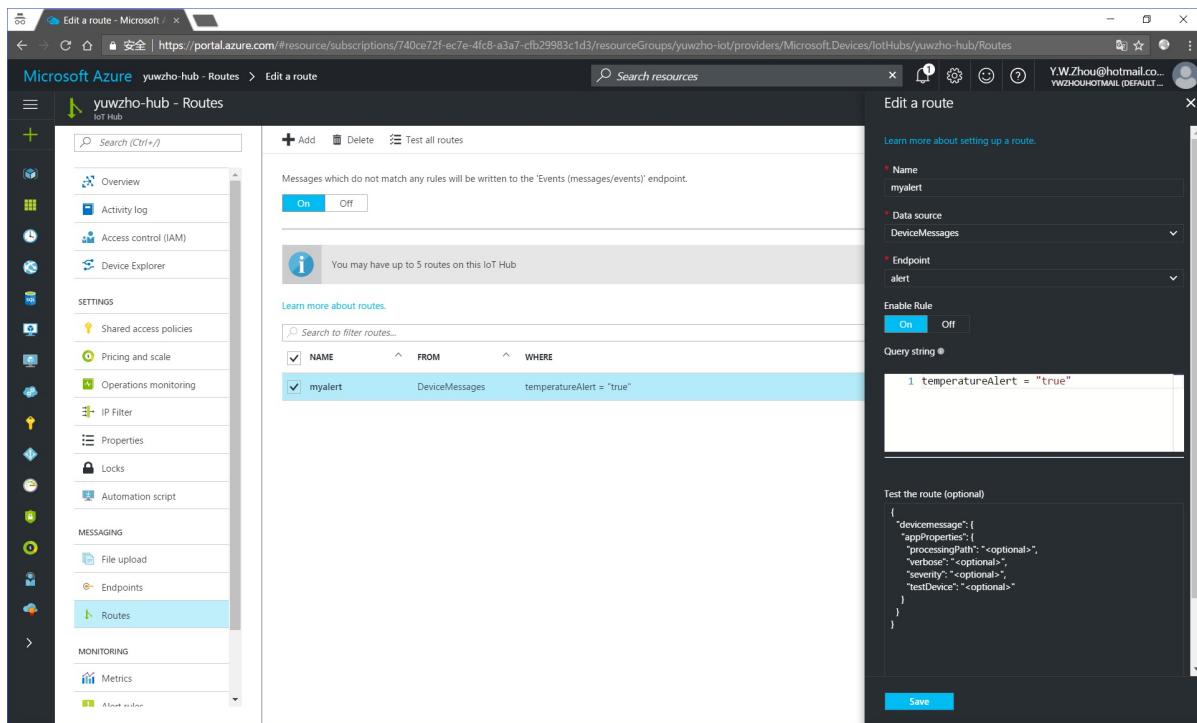
Name: The name of the routing rule.

Data source: Select **DeviceMessages**.

Endpoint: Select the endpoint you created.

Query string: Enter `temperatureAlert = "true"`.

3. Click **Save**.



Create and configure a logic app

Create a logic app

1. In the [Azure portal](#), click **New > Enterprise Integration > Logic App**.
2. Enter the following information:

Name: The name of the logic app.

Resource group: Use the same resource group that your IoT hub uses.

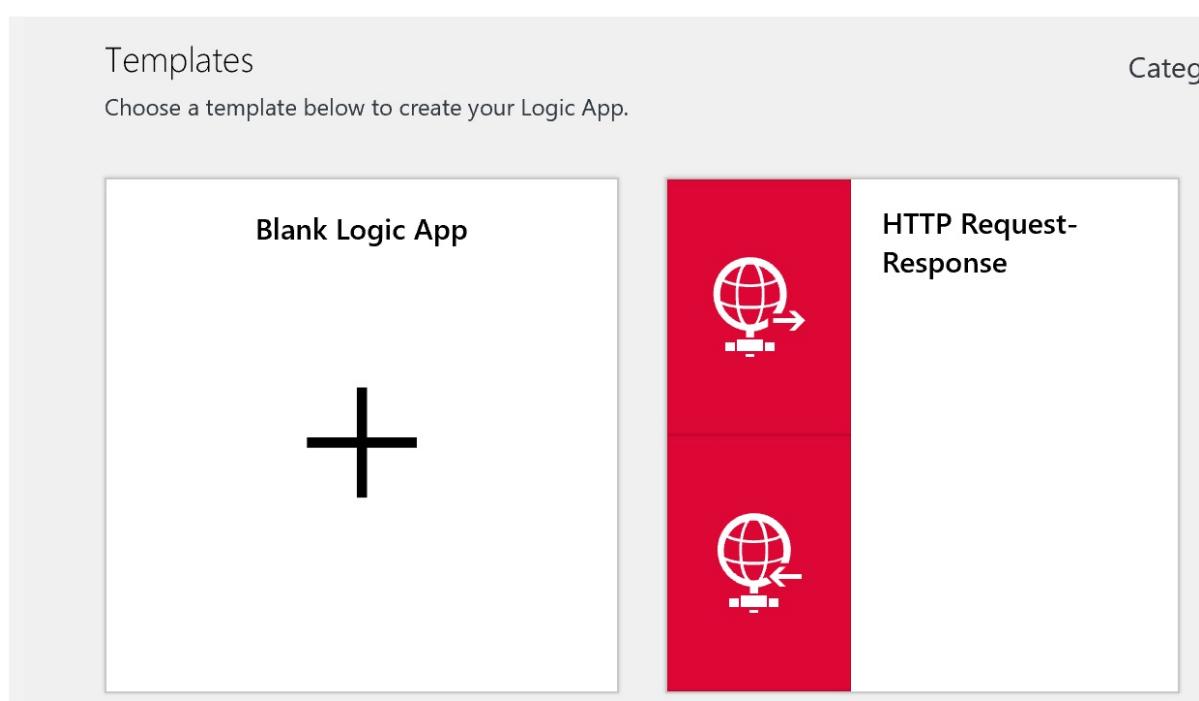
Location: Use the same location that your IoT hub uses.

3. Click **Create**.

Configure the logic app

1. Open the logic app that opens into the Logic Apps Designer.
2. In the Logic Apps Designer, click **Blank Logic App**.

Logic Apps Designer



3. Click **Service Bus**.

This screenshot shows the 'Services' section of the Logic Apps Designer. At the top, there's a search bar with the placeholder 'Search all services and triggers'. Below the search bar, the word 'SERVICES' is centered above a grid of ten service icons. To the right, there's a 'SEE MORE' link. The services listed are: Request / Response (red icon), Schedule (orange icon), Service Bus (green icon), Twitter (blue icon), OneDrive for Business (blue icon); Dynamics 365 (dark blue icon), SharePoint (blue icon), FTP (orange icon), SFTP (yellow icon), and Office 365 / Outlook (blue icon).

4. Click **Service Bus – When one or more messages arrive in a queue (auto-complete)**.

5. Create a service bus connection.

- i. Enter a connection name.

- ii. Click the service bus namespace > the service bus policy > **Create**.

* CONNECTION NAME
ChenServiceBusConnection

* SERVICE BUS POLICY

Name	Rights
RootManageSharedAccessKey	Listen, Manage, Send

Create

Manually enter connection information

- iii. Click **Continue** after the service bus connection is created.
- iv. Select the queue that you created and enter 175 for **Maximum message count**

* Queue name
chenservicebusqueue

Maximum message count
175

Show advanced options ▾

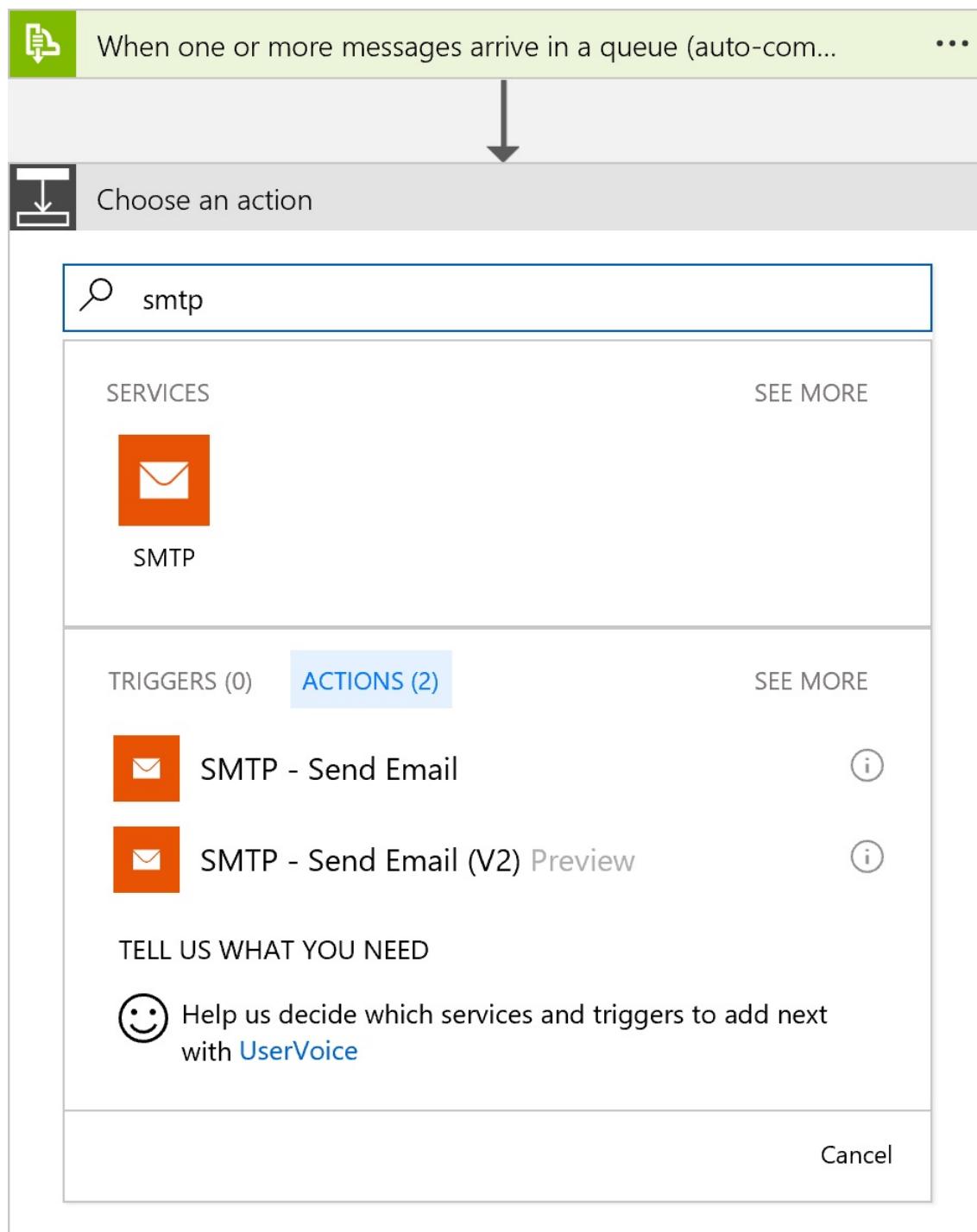
How often do you want to check for items?

* Frequency
Minute

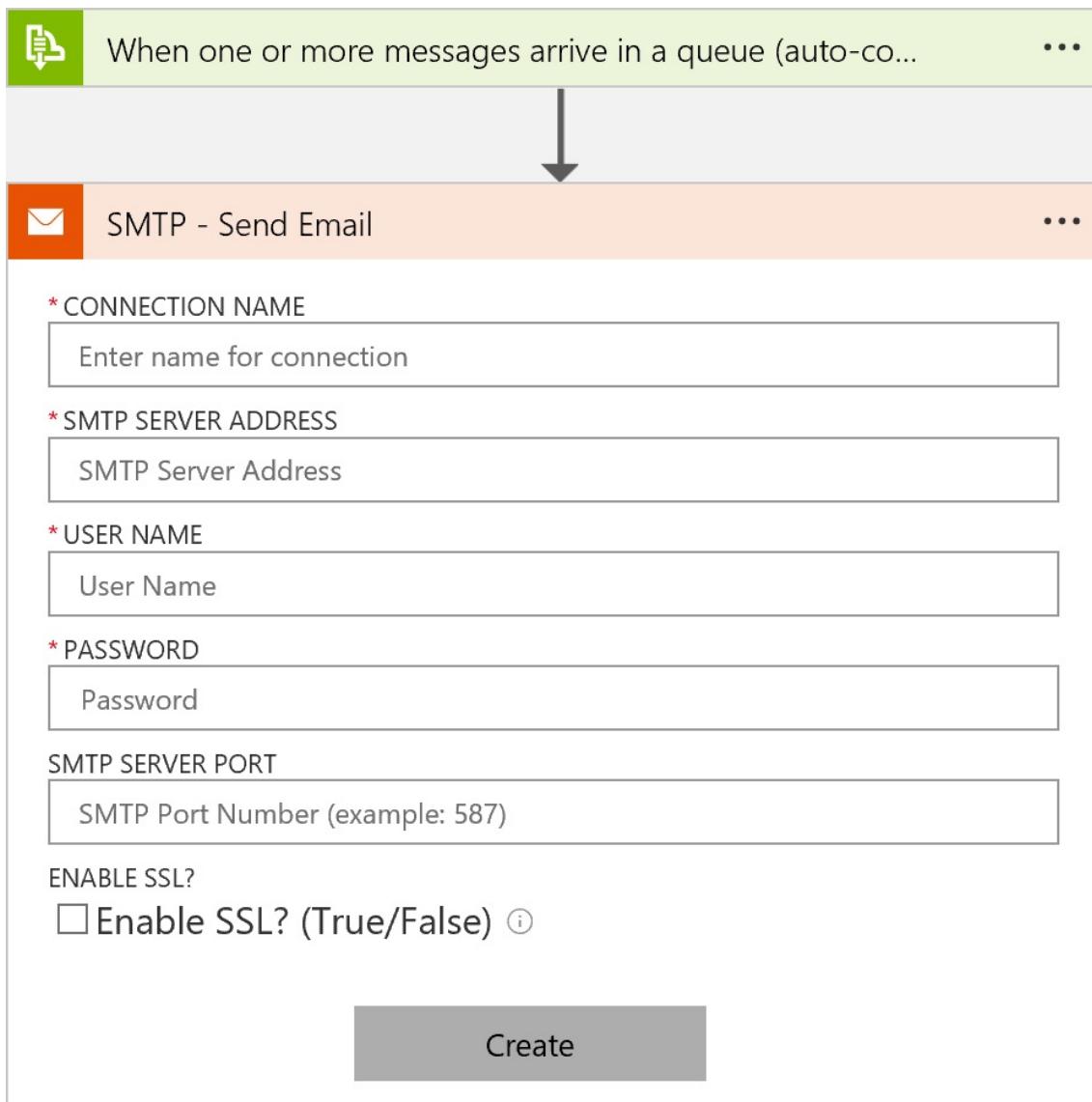
* Interval
3

Connected to ChenServiceBusConnection. [Change connection](#).

- v. Click "Save" button to save the changes.
6. Create an SMTP service connection.
- Click **New step > Add an action**.
 - Type **SMTP**, click the **SMTP** service in the search result, and then click **SMTP - Send Email**.



- iii. Enter the SMTP information of your mailbox, and then click **Create**.



Get the SMTP information for [Hotmail/Outlook.com](#), [Gmail](#), and [Yahoo Mail](#).

- iv. Enter your email address for **From** and **To**, and **High temperature detected** for **Subject** and **Body**.
- v. Click **Save**.

The logic app is in working order when you save it.

Test the logic app

1. Start the client application that you deploy to your device in [Connect ESP8266 to Azure IoT Hub](#).
2. Increase the environment temperature around the SensorTag to be above 30 C. For example, light a candle around your SensorTag.
3. You should receive an email notification sent by the logic app.

[!NOTE] Your email service provider may need to verify the sender identity to make sure it is you who sends the email.

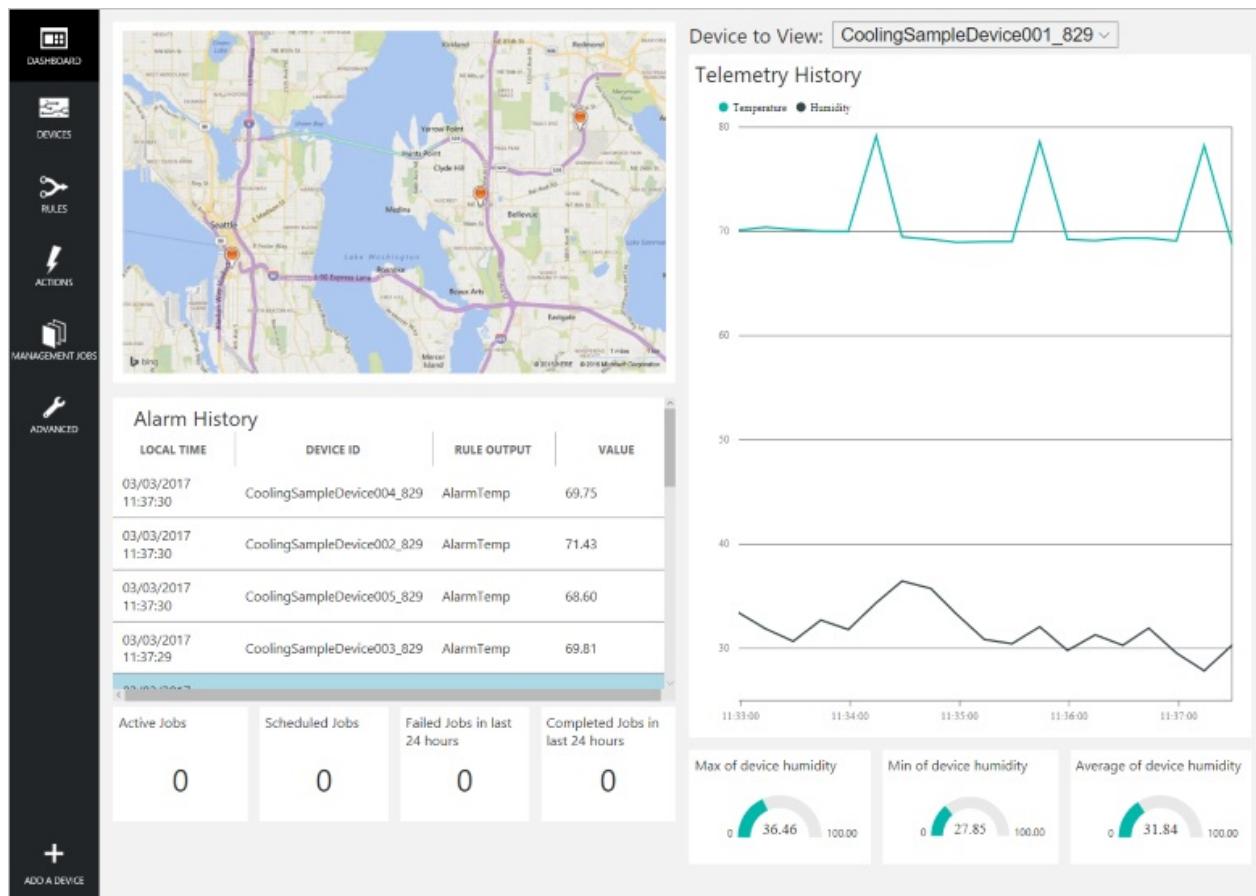
Next steps

You have successfully created a logic app that connects your IoT hub and your mailbox for temperature monitoring and notifications.

Get started with the preconfigured solutions

Azure IoT Suite [preconfigured solutions](#) combine multiple Azure IoT services to deliver end-to-end solutions that implement common IoT business scenarios. The *remote monitoring* preconfigured solution connects to and monitors your devices. You can use the solution to analyze the stream of data from your devices and to improve business outcomes by making processes respond automatically to that stream of data.

This tutorial shows you how to provision the remote monitoring preconfigured solution. It also walks you through the basic features of the preconfigured solution. You can access many of these features from the solution *dashboard* that deploys as part of the preconfigured solution:



To complete this tutorial, you need an active Azure subscription.

[!NOTE] If you don't have an account, you can create a free trial account in just a couple of minutes. For details, see [Azure Free Trial](#).

Provision the solution

If you haven't already provisioned the remote monitoring preconfigured solution in your account:

1. Log on to azureiotsuite.com using your Azure account credentials, and click **+** to create a solution.
2. Click **Select** on the **Remote monitoring** tile.
3. Enter a **Solution name** for your remote monitoring preconfigured solution.
4. Select the **Region** and **Subscription** you want to use to provision the solution.
5. Click **Create Solution** to begin the provisioning process. This process typically takes several minutes to run.

Wait for the provisioning process to complete

1. Click the tile for your solution with **Provisioning** status.
2. Notice the **Provisioning states** as Azure services are deployed in your Azure subscription.
3. Once provisioning completes, the status changes to **Ready**.
4. Click the tile to see the details of your solution in the right-hand pane.

[!NOTE] If you are encountering issues deploying the pre-configured solution, review [Permissions on the azureiotsuite.com site](#) and the [FAQ](#). If the issues persist, create a service ticket on the [portal](#).

Are there details you'd expect to see that aren't listed for your solution? Give us feature suggestions on [User Voice](#).

Scenario overview

When you deploy the remote monitoring preconfigured solution, it is prepopulated with resources that enable you to step through a common remote monitoring scenario. In this scenario, several devices connected to the solution are reporting unexpected temperature values. The following sections show you how to:

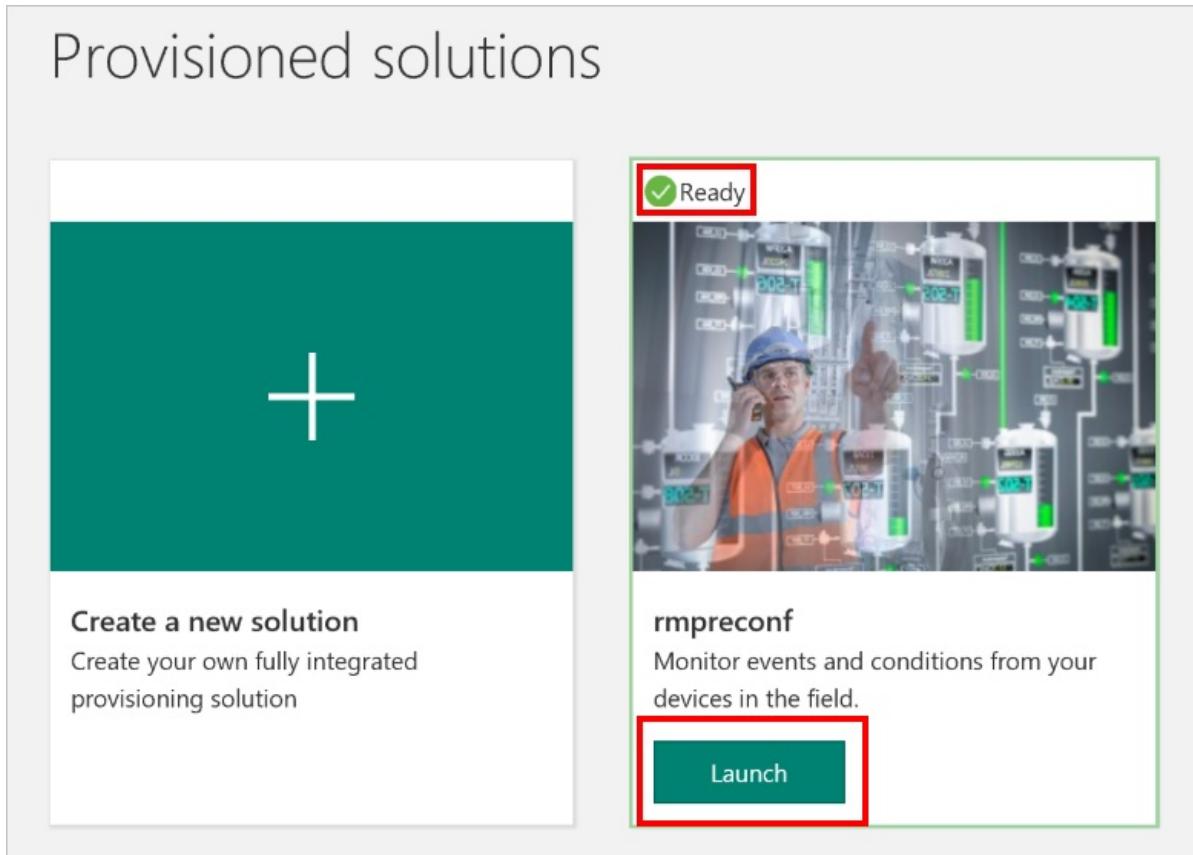
- Identify the devices sending unexpected temperature values.
- Configure these devices to send more detailed telemetry.
- Fix the problem by updating the firmware on these devices.
- Verify that your action has resolved the issue.

A key feature of this scenario is that you can perform all these actions remotely from the solution dashboard. You do not need physical access to the devices.

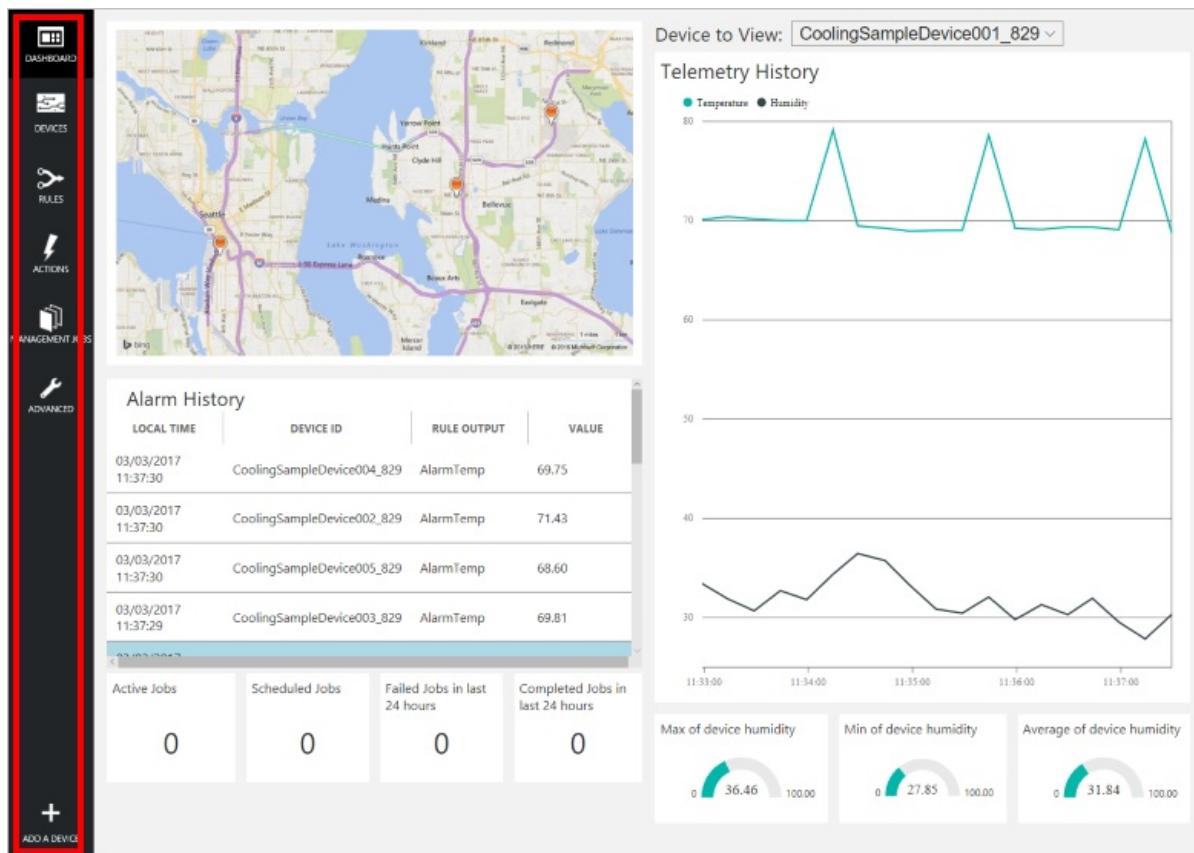
View the solution dashboard

The solution dashboard enables you to manage the deployed solution. For example, you can view telemetry, add devices, and configure rules.

1. When the provisioning is complete and the tile for your preconfigured solution indicates **Ready**, choose **Launch** to open your remote monitoring solution portal in a new tab.



2. By default, the solution portal shows the *dashboard*. You can navigate to other areas of the solution portal using the menu on the left-hand side of the page.

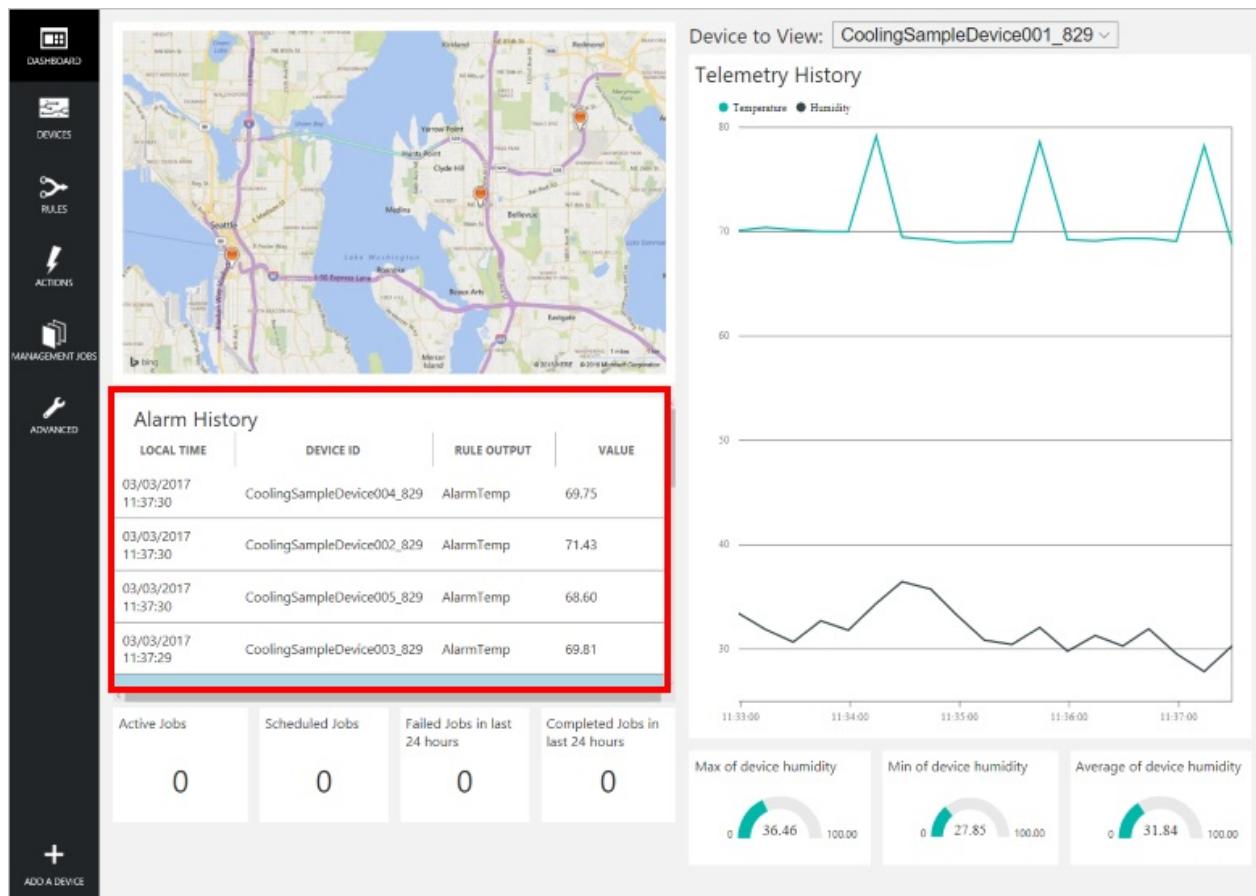


The dashboard displays the following information:

- A map that displays the location of each device connected to the solution. When you first run the solution, there are 25 simulated devices. The simulated devices are implemented as Azure WebJobs, and the solution uses the Bing Maps API to plot information on the map. See the [FAQ](#) to learn how to make the map dynamic.
- A **Telemetry History** panel that plots humidity and temperature telemetry from a selected device in near real time and displays aggregate data such as maximum, minimum, and average humidity.
- An **Alarm History** panel that shows recent alarm events when a telemetry value has exceeded a threshold. You can define your own alarms in addition to the examples created by the preconfigured solution.
- A **Jobs** panel that displays information about scheduled jobs. You can schedule your own jobs on **Management jobs** page.

View alarms

The alarm history panel shows you that five devices are reporting higher than expected telemetry values.



[!NOTE] These alarms are generated by a rule that is included in the preconfigured solution. This rule generates an alert when the temperature value sent by a device exceeds 60. You can define your own rules and actions by choosing [Rules](#) and [Actions](#) in the left-hand menu.

View devices

The devices list shows all the registered devices in the solution. From the device list you can view and edit device metadata, add or remove devices, and invoke methods on devices. You can filter and sort the list of devices in the device list. You can also customize the columns shown in the device list.

1. Choose **Devices** to show the device list for this solution.

The screenshot shows the Azure IoT Suite interface. On the left, there's a dark sidebar with icons for Dashboard, Devices (which is selected and highlighted with a red box), Rules, Actions, Management Jobs, Advanced, and Add a Device. The main content area is titled 'All Devices (26)' and contains a table with the following data:

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERATURE	FWST
	● Running	CoolingSampleDevice001_979	Contoso Inc.	1.0	2	42	
	● Running	CoolingSampleDevice002_979	Contoso Inc.	1.0	2	40	
	● Running	CoolingSampleDevice003_979	Contoso Inc.	2.0	Building 40	34.5	Compl
	● Running	CoolingSampleDevice004_979	Contoso Inc.	1.0	Building 40	34.5	
	● Running	CoolingSampleDevice005_979	Contoso Inc.	1.1	Building 40	34.5	
	● Running	CoolingSampleDevice006_979	Contoso Inc.	1.10	Building 43	34.5	
	● Running	CoolingSampleDevice007_979	Contoso Inc.	1.3	Building 43	34.5	
	● Running	CoolingSampleDevice008_979	Contoso Inc.	1.5	Building 43	34.5	
	● Running	CoolingSampleDevice009_979	Contoso Inc.	1.5	Building 40	34.5	
	● Running	CoolingSampleDevice010_979	Contoso Inc.	1.11	Building 43	34.5	
	● Running	CoolingSampleDevice011_979	Contoso Inc.	1.7	Building 43	34.5	
	● Running	CoolingSampleDevice012_979	Contoso Inc.	1.11	Building 40	34.5	
	● Running	CoolingSampleDevice013_979	Contoso Inc.	1.1	Building 43	34.5	
	● Running	CoolingSampleDevice014_979	Contoso Inc.	1.7	Building 40	34.5	
	● Running	CoolingSampleDevice015_979	Contoso Inc.	1.13	Building 43	34.5	
	● Running	CoolingSampleDevice016_979	Contoso Inc.	1.8	Building 40	34.5	
	● Running	CoolingSampleDevice017_979	Contoso Inc.	1.6	Building 43	34.5	
	● Running	CoolingSampleDevice018_979	Contoso Inc.	1.13	Building 40	34.5	

2. The device list initially shows 25 simulated devices created by the provisioning process. You can add additional simulated and physical devices to the solution.
3. To view the details of a device, choose a device in the device list.

The screenshot shows the Azure IoT Suite dashboard with a sidebar on the left containing icons for Dashboard, Devices, Rules, Actions, Management Jobs, and Advanced. The main area displays a table titled 'All Devices (26)' with columns for Icon, Status, Device ID, Manufacturer, and Firmware Version. One device row is highlighted with a red box. To the right is the 'DEVICE DETAILS' panel, which includes a circular icon with a thermometer, links for Disable Device, Add Rule..., Commands, and Methods, and sections for Device Twin (Tags, Desired Properties, Reported Properties), and Recent Jobs.

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRM
	Running	CoolingSampleDevice001_979	Contoso Inc.	1.0
	Running	CoolingSampleDevice002_979	Contoso Inc.	1.0
	Running	CoolingSampleDevice003_979	Contoso Inc.	2.0
	Running	CoolingSampleDevice004_979	Contoso Inc.	1.0
	Running	CoolingSampleDevice005_979	Contoso Inc.	1.1
	Running	CoolingSampleDevice006_979	Contoso Inc.	1.10
	Running	CoolingSampleDevice007_979	Contoso Inc.	1.3
	Running	CoolingSampleDevice008_979	Contoso Inc.	1.5
	Running	CoolingSampleDevice009_979	Contoso Inc.	1.5
	Running	CoolingSampleDevice010_979	Contoso Inc.	1.11
	Running	CoolingSampleDevice011_979	Contoso Inc.	1.7
	Running	CoolingSampleDevice012_979	Contoso Inc.	1.11
	Running	CoolingSampleDevice013_979	Contoso Inc.	1.1
	Running	CoolingSampleDevice014_979	Contoso Inc.	1.7
	Running	CoolingSampleDevice015_979	Contoso Inc.	1.13
	Running	CoolingSampleDevice016_979	Contoso Inc.	1.8
	Running	CoolingSampleDevice017_979	Contoso Inc.	1.6
	Running	CoolingSampleDevice018_979	Contoso Inc.	1.13

The **Device Details** panel contains six sections:

- A collection of links that enable you to customize the device icon, disable the device, add a rule, invoke a method, or send a command. For a comparison of commands (device-to-cloud messages) and methods (direct methods), see [Cloud-to-device communications guidance](#).
- The **Device Twin - Tags** section enables you to edit tag values for the device. You can display tag values in the device list and use tag values to filter the device list.
- The **Device Twin - Desired Properties** section enables you to set property values to be sent to the device.
- The **Device Twin - Reported Properties** section shows property values sent from the device.
- The **Device Properties** section shows information from the identity registry such as the device id and authentication keys.
- The **Recent Jobs** section shows information about any jobs that have recently targeted this device.

Filter the device list

You can use a filter to display only those devices that are sending unexpected temperature values. The remote monitoring preconfigured solution includes the **Unhealthy devices** filter to show devices with a mean temperature value greater than 60. You can also [create your own filters](#).

1. Choose **Open saved filter** to display a list of available filters. Then choose **Unhealthy devices** to apply the filter:

The screenshot shows the Azure IoT Suite interface. On the left, there's a sidebar with icons for DASHBOARD, DEVICES (selected), RULES, ACTIONS, and MANAGEMENT JOBS. The main area is titled 'All Devices (25)'. A modal window titled 'OPEN SAVED FILTER' is open, showing a list of filters: 'Saved Filters', 'All Devices', 'Old firmware devices', and 'Unhealthy devices'. The 'Unhealthy devices' option is highlighted with a red box. Below the modal, there's a status bar with 'Running' and 'CoolingSampleDevice004_829'.

DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERAT
ColeDevice001_829	Contoso Inc.	1.4	Building 43	70
ColeDevice002_829	Contoso Inc.	1.12	Building 40	70
ColeDevice003_829	Contoso Inc.	1.4	Building 40	70
CoolingSampleDevice004_829	Contoso Inc.	1.12	Building 43	70

2. The device list now shows only devices with a mean temperature value greater than 60.

This screenshot shows the same interface after applying the 'Unhealthy devices' filter. The title bar now says 'Unhealthy devices (5 filtered from 25)'. The status bar at the bottom still shows 'Running' and 'CoolingSampleDevice004_829'. The main table now only lists five devices, all of which have a mean temperature value greater than 60.

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERAT
	Running	CoolingSampleDevice001_829	Contoso Inc.	1.4	Building 43	70
	Running	CoolingSampleDevice002_829	Contoso Inc.	1.12	Building 40	70
	Running	CoolingSampleDevice003_829	Contoso Inc.	1.4	Building 40	70
	Running	CoolingSampleDevice004_829	Contoso Inc.	1.12	Building 43	70
	Running	CoolingSampleDevice005_829	Contoso Inc.	1.12	Building 40	70

Update desired properties

You have now identified a set of devices that may need remediation. However, you decide that the data frequency of 15 seconds is not sufficient for a clear diagnosis of the issue. Changing the telemetry frequency to five seconds to provide you with more data points to better diagnose the issue. You can push this configuration change to your remote devices from the solution portal. You can make the change once, evaluate the impact, and then act on the results.

Follow these steps to run a job that changes the **TelemetryInterval** desired property for the affected devices. When the devices receive the new **TelemetryInterval** property value, they change their configuration to send telemetry every five seconds instead of every 15 seconds:

1. While you are showing the list of unhealthy devices in the device list, choose **Job Scheduler**, then **Edit Device Twin**.
2. Call the job **Change telemetry interval**.
3. Change the value of the **Desired Property** name **desired.Config.TelemetryInterval** to five seconds.
4. Choose **Schedule**.

← Edit Device Twin Unhealthy devices (5)

JOB NAME	
Change telemetry interval	

DESIRED PROPERTIES			
DESIRED PROPERTY NAME	VALUE	DATA TYPE	DELETE
desired.Config.TelemetryInterval	5	Number	<input type="checkbox"/> Clear
desired.sampleprop		String	<input type="checkbox"/> Clear

TAGS			
TAG NAME	VALUE	DATA TYPE	DELETE
tags.sampletag		String	<input type="checkbox"/>

JOB TIME	
START TIME	MAXIMUM EXECUTION TIME
3 March 2017 11:57	0 Mins

Cancel	Schedule
--------	----------

5. You can monitor the progress of the job on the **Management Jobs** page in the portal.

[!NOTE] If you want to change a desired property value for an individual device, use the **Desired Properties** section in the **Device Details** panel instead of running a job.

This job sets the value of the **TelemetryInterval** desired property in the device twin for all the devices selected by the filter. The devices retrieve this value from the device twin and update their behavior. When a device retrieves and processes a desired property from a

device twin, it sets the corresponding reported value property.

Invoke methods

While the job runs, you notice in the list of unhealthy devices that all these devices have old (less than version 1.6) firmware versions.

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERAT
	Running	CoolingSampleDevice001_829	Contoso Inc.	1.4	Building 43	70
	Running	CoolingSampleDevice002_829	Contoso Inc.	1.12	Building 40	70
	Running	CoolingSampleDevice003_829	Contoso Inc.	1.4	Building 40	70
	Running	CoolingSampleDevice004_829	Contoso Inc.	1.12	Building 43	70
	Running	CoolingSampleDevice005_829	Contoso Inc.	1.12	Building 40	70

This firmware version may be the root cause of the unexpected temperature values because you know that other healthy devices were recently updated to version 2.0. You can use the built-in **Old firmware devices** filter to identify any devices with old firmware versions. From the portal, you can then remotely update all the devices still running old firmware versions:

1. Choose **Open saved filter** to display a list of available filters. Then choose **Old firmware devices** to apply the filter:

DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERAT
oleDevice001_829	Contoso Inc.	1.4	Building 43	70
oleDevice002_829	Contoso Inc.	1.12	Building 40	70
oleDevice003_829	Contoso Inc.	1.4	Building 40	70
oleDevice004_829	Contoso Inc.	1.12	Building 43	70

2. The device list now shows only devices with old firmware versions. This list includes the five devices identified by the **Unhealthy devices** filter and three additional devices:

Old firmware devices (8 filtered from 25)							
	ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERATURE
		Running	CoolingSampleDevice001_829	Contoso Inc.	1.4	Building 43	70
		Running	CoolingSampleDevice002_829	Contoso Inc.	1.12	Building 40	70
		Running	CoolingSampleDevice003_829	Contoso Inc.	1.4	Building 40	70
		Running	CoolingSampleDevice004_829	Contoso Inc.	1.12	Building 43	70
		Running	CoolingSampleDevice005_829	Contoso Inc.	1.12	Building 40	70
		Running	CoolingSampleDevice006_829	Contoso Inc.	1.13	Building 43	34.5
		Running	CoolingSampleDevice007_829	Contoso Inc.	1.2	Building 43	34.5
		Running	CoolingSampleDevice008_829	Contoso Inc.	1.2	Building 40	34.5

3. Choose **Job Scheduler**, then **Invoke Method**.
4. Set **Job Name** to **Firmware update to version 2.0**.
5. Choose **InitiateFirmwareUpdate** as the **Method**.
6. Set the **FwPackageUri** parameter to
<https://iotrmassets.blob.core.windows.net/firmwares/FW20.bin>.
7. Choose **Schedule**. The default is for the job to run now.

← Invoke Method Old firmware devices (8)

Job Name

Firmware update to version 2.0

Method

InitiateFirmwareUpdate(string)

8 devices applicable
0 devices inapplicable

Parameters

PARAMETERS	VALUE
FwPackageUri	blob.core.windows.net/firmwares/FW20.bin

Job Time

START TIME

3 March 2017 12:17

MAXIMUM EXECUTION TIME ⓘ Mins

Schedule

[!NOTE] If you want to invoke a method on an individual device, choose **Methods** in the **Device Details** panel instead of running a job.

This job invokes the **InitiateFirmwareUpdate** direct method on all the devices selected by the filter. Devices respond immediately to IoT Hub and then initiate the firmware update process asynchronously. The devices provide status information about the firmware update process through reported property values, as shown in the following screenshots. Choose the **Refresh** icon to update the information in the device and job lists:

The screenshot shows the Azure IoT Suite interface with three main tabs:

- JOBS**: Shows a table of jobs with columns: STATUS, JOB NAME, FILTER, OPERATIONS, START TIME, END TIME, NO. OF DEVICES, and SUCCEEDED COUNT. One job is Running (Firmware update to version 2.0) and two are Queued (Firmware update to version 2.0 and Change telemetry interval).
- DEVICES**: Shows a table of devices with columns: ICON, STATUS, DEVICE ID, MANUFACTURER, FIRMWARE, and BUILDING. All devices are Running and belong to Contoso Inc. with Firmware 1.4.
- MANAGEMENT JOBS**: Shows a table of management jobs with columns: STATUS, JOB NAME, FILTER, OPERATIONS, START TIME, END TIME, NO. OF DEVICES, and SUCCEEDED COUNT. Two jobs are Completed (Firmware update to version 2.0 and Change telemetry interval).

Red boxes highlight specific areas of interest:

- A red box surrounds the **JOBS** tab icon.
- A red box surrounds the **JOBS** table header and the first job row.
- A red box surrounds the **DEVICES** tab icon.
- A red box surrounds the **DEVICES** table header and the first device row.
- A red box surrounds the **MANAGEMENT JOBS** tab icon.
- A red box surrounds the **MANAGEMENT JOBS** table header and the first two job rows.
- A red box surrounds the **DEVICE DETAILS** pane on the right, which displays device location and method update logs.

[!NOTE] In a production environment, you can schedule jobs to run during a designated maintenance window.

Scenario review

In this scenario, you identified a potential issue with some of your remote devices using the alarm history on the dashboard and a filter. You then used the filter and a job to remotely configure the devices to provide more information to help diagnose the issue. Finally, you used a filter and a job to schedule maintenance on the affected devices. If you return to the dashboard, you can check that there are no longer any alarms coming from devices in your solution. You can use a filter to verify that the firmware is up-to-date on all the devices in your solution and that there are no more unhealthy devices:

The image contains two screenshots of the Azure IoT Suite interface, specifically the Device list page. Both screenshots show a search bar at the top with a red box around it. The first screenshot has a search term 'Old firmware devices (0 filtered from 25)' and a filter 'FirmwareVersion < '2.0''. The second screenshot has a search term 'Unhealthy devices (0 filtered from 25)' and a filter 'TemperatureMeanValue > 60'. Below the search bar is a table header with columns: ICON, STATUS, DEVICE ID, MANUFACTURER, FIRMWARE, BUILDING, TEMPERATURE, and FWSTATUS. A message 'No matching records found' is displayed in a red box in the center of each table.

Other features

The following sections describe some additional features of the remote monitoring preconfigured solution that are not described as part of the previous scenario.

Customize columns

You can customize the information shown in the device list by choosing **Column editor**. You can add and remove columns that display reported property and tag values. You can also reorder and rename columns:

The screenshot shows the 'All Devices' list with 26 entries. The columns are: ICON, STATUS, DEVICE ID, MANUFACTURER, FIRMWARE, and BUILDING. A red box highlights the 'COLUMN EDITOR' button at the top of the list. To the right, a modal window titled 'EDIT COLUMNS' is open, showing a list of properties: STATUS (tags.HubEnabledState), DEVICE ID (deviceId), MANUFACTURER (reported.S), FIRMWARE (reported.System.FirmwareVersion), BUILDING (tags.Building), TEMPERATURE (reported.Config.Temperature), and FWSTATUS (reported.Method.UpdateFirmware). A red box highlights the 'MANUFACTURER (reported.S)' entry.

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING
	● Running	CoolingSampleDevice001_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice002_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice003_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice004_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice005_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice006_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice007_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice008_979	Contoso Inc.	1.0	
	● Running	CoolingSampleDevice009_979	Contoso Inc.	1.0	

Customize the device icon

You can customize the device icon displayed in the device list from the **Device Details** panel as follows:

1. Choose the pencil icon to open the **Edit image** panel for a device:

The screenshot shows the 'All Devices' list with 26 entries. The third device in the list has its 'ICON' column highlighted with a blue background. To the right, the 'DEVICE DETAILS' panel is open for this device. In the 'Device Twin' section, there is a thumbnail image of a building with a blue pencil icon overlaid. A red box highlights this pencil icon. Below the image, there are buttons for 'Disable Device', 'Add Rule...', 'Commands', and 'Methods'. The 'Tags' section lists 'Building' and 'Floor' with values '2' and '2F' respectively. There is also an 'Edit' button.

2. Either upload a new image or use one of the existing images and then choose **Save**:

← Edit Icon for CoolingSampleDevice003_979

Upload new icon

Please choose an image file.

 CHOOSE FILE

Please use an image file with 150*150 pixels and less than 4MB.

PREVIEW



Apply existing icon



Cancel

Save

3. The image you selected now displays in the **Icon** column for the device.

[!NOTE] The image is stored in blob storage. A tag in the device twin contains a link to the image in blob storage.

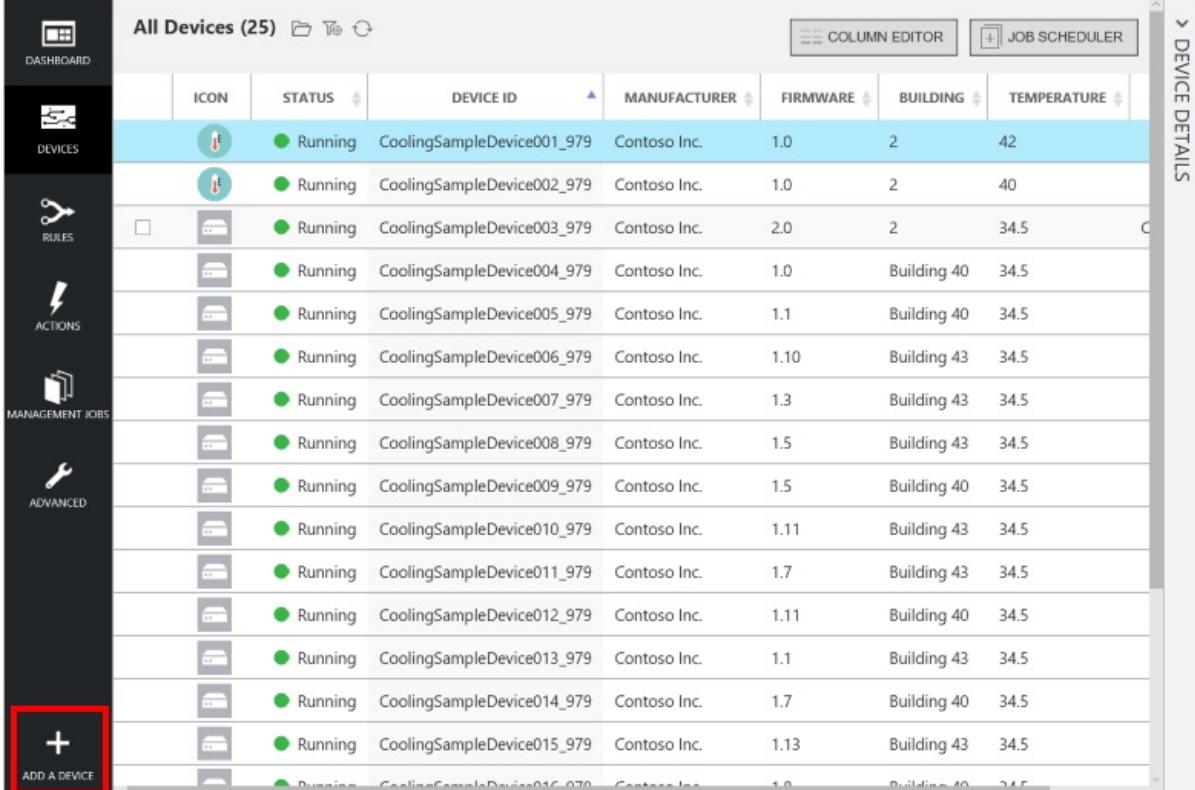
Add a device

When you deploy the preconfigured solution, you automatically provision 25 sample devices that you can see in the device list. These devices are *simulated devices* running in an Azure WebJob. Simulated devices make it easy for you to experiment with the preconfigured solution without the need to deploy real, physical devices. If you do want to connect a real device to the solution, see the [Connect your device to the remote monitoring preconfigured solution](#) tutorial.

The following steps show you how to add a simulated device to the solution:

1. Navigate back to the device list.

2. To add a device, choose **+ Add A Device** in the bottom left corner.



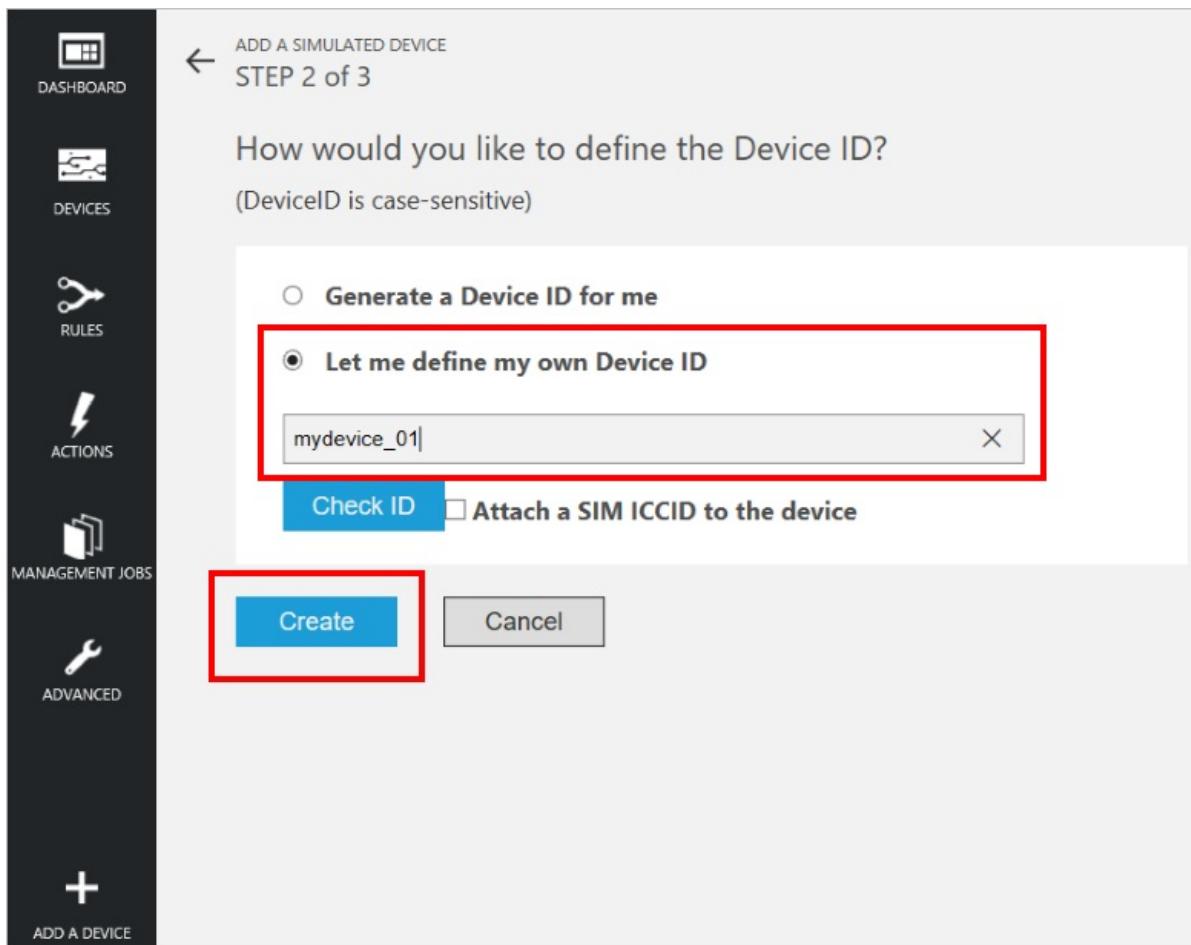
All Devices (25)							
	ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE	BUILDING	TEMPERATURE
		● Running	CoolingSampleDevice001_979	Contoso Inc.	1.0	2	42
		● Running	CoolingSampleDevice002_979	Contoso Inc.	1.0	2	40
		● Running	CoolingSampleDevice003_979	Contoso Inc.	2.0	2	34.5
		● Running	CoolingSampleDevice004_979	Contoso Inc.	1.0	Building 40	34.5
		● Running	CoolingSampleDevice005_979	Contoso Inc.	1.1	Building 40	34.5
		● Running	CoolingSampleDevice006_979	Contoso Inc.	1.10	Building 43	34.5
		● Running	CoolingSampleDevice007_979	Contoso Inc.	1.3	Building 43	34.5
		● Running	CoolingSampleDevice008_979	Contoso Inc.	1.5	Building 43	34.5
		● Running	CoolingSampleDevice010_979	Contoso Inc.	1.11	Building 43	34.5
		● Running	CoolingSampleDevice011_979	Contoso Inc.	1.7	Building 43	34.5
		● Running	CoolingSampleDevice012_979	Contoso Inc.	1.11	Building 40	34.5
		● Running	CoolingSampleDevice013_979	Contoso Inc.	1.1	Building 43	34.5
		● Running	CoolingSampleDevice014_979	Contoso Inc.	1.7	Building 40	34.5
		● Running	CoolingSampleDevice015_979	Contoso Inc.	1.13	Building 43	34.5
		● Running	CoolingSampleDevice016_979	Contoso Inc.	1.0	Building 40	34.5

3. Choose **Add New** on the **Simulated Device** tile.

The screenshot shows the Azure IoT Suite interface. On the left is a dark sidebar with icons for Dashboard, Devices, Rules, Actions, Management Jobs, Advanced, and a plus sign for adding a device. The main area has a header 'Microsoft Azure IoT Suite - Remote Monitoring Solution' and a subtitle 'ADD A DEVICE STEP 1 of 3'. It contains two sections: 'Simulated Device' (highlighted with a red border) and 'Custom Device'. The 'Simulated Device' section describes software for simulating devices and includes a blue 'Add New' button. The 'Custom Device' section describes a physical hardware device and also includes a blue 'Add New' button.

In addition to creating a new simulated device, you can also add a physical device if you choose to create a **Custom Device**. To learn more about connecting physical devices to the solution, see [Connect your device to the IoT Suite remote monitoring preconfigured solution](#).

4. Select **Let me define my own Device ID**, and enter a unique device ID name such as **mydevice_01**.
5. Choose **Create**.

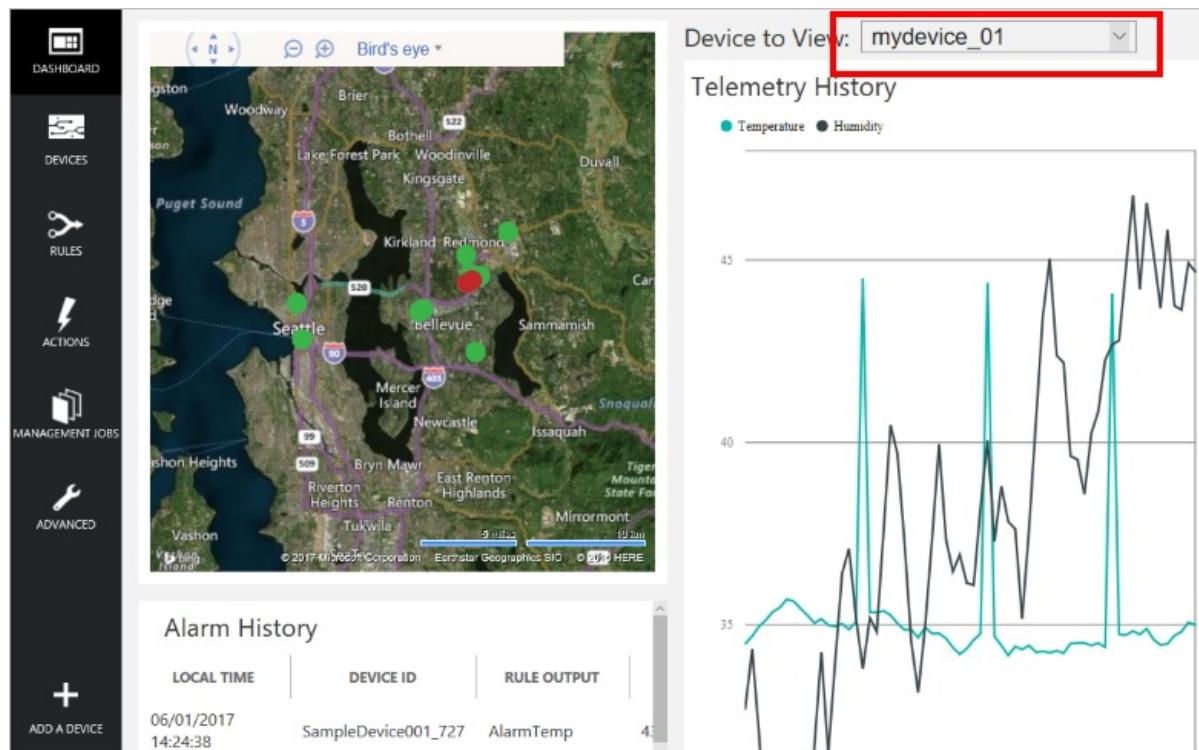


6. In step 3 of **Add a simulated device**, choose **Done** to return to the device list.
7. You can view your device **Running** in the device list.

The screenshot shows the 'All Devices' list in the Azure IoT Suite. The left sidebar includes options like DASHBOARD, DEVICES, RULES, ACTIONS, MANAGEMENT JOBS, and ADD A DEVICE. The main area displays a table titled 'All Devices (26)' with columns: ICON, STATUS, DEVICE ID, MANUFACTURER, and FIRMWARE. The 'my_device01' row is highlighted with a red box. The right panel shows 'DEVICE DETAILS' for this device, including a thumbnail, 'Disable Device', 'Add Rule...', 'Commands', and 'Methods' buttons. Below this is the 'Device Twin' section, which is collapsed. The table footer shows navigation buttons: < Previous, 1, 2, Next.

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMWARE
[Icon]	● Running	CoolingSampleDevice021_979	Contoso Inc.	1.3
[Icon]	● Running	CoolingSampleDevice022_979	Contoso Inc.	1.10
[Icon]	● Running	CoolingSampleDevice023_979	Contoso Inc.	1.0
[Icon]	● Running	CoolingSampleDevice024_979	Contoso Inc.	1.5
[Icon]	● Running	CoolingSampleDevice025_979	Contoso Inc.	1.10
[Icon]	● Running	my_device01	Contoso Inc.	1.14

8. You can also view the simulated telemetry from your new device on the dashboard:



Disable and delete a device

You can disable a device, and after it is disabled you can remove it:

The device list shows 26 devices. A red box highlights the row for 'my_device01', which is marked as 'Disabled'. The right panel displays 'DEVICE DETAILS' with options like Enable Device, Add Rule..., Commands, Methods, and Remove Device... . It also shows the 'Device Twin' section with tags for Building, Building 43, Floor, and 2F.

ICON	STATUS	DEVICE ID	MANUFACTURER	FIRMV
	Running	CoolingSampleDevice021_979	Contoso Inc.	1.3
	Running	CoolingSampleDevice022_979	Contoso Inc.	1.10
	Running	CoolingSampleDevice023_979	Contoso Inc.	1.0
	Running	CoolingSampleDevice024_979	Contoso Inc.	1.5
	Running	CoolingSampleDevice025_979	Contoso Inc.	1.10
	Disabled	my_device01	Contoso Inc.	1.14

Add a rule

There are no rules for the new device you just added. In this section, you add a rule that triggers an alarm when the temperature reported by the new device exceeds 47 degrees. Before you start, notice that the telemetry history for the new device on the dashboard shows the device temperature never exceeds 45 degrees.

1. Navigate back to the device list.

2. To add a rule for the device, select your new device in the **Devices List**, and then choose **Add rule**.
3. Create a rule that uses **Temperature** as the data field and uses **AlarmTemp** as the output when the temperature exceeds 47 degrees:

← Create Rule for Device my_device01

RULE STATUS
Enabled

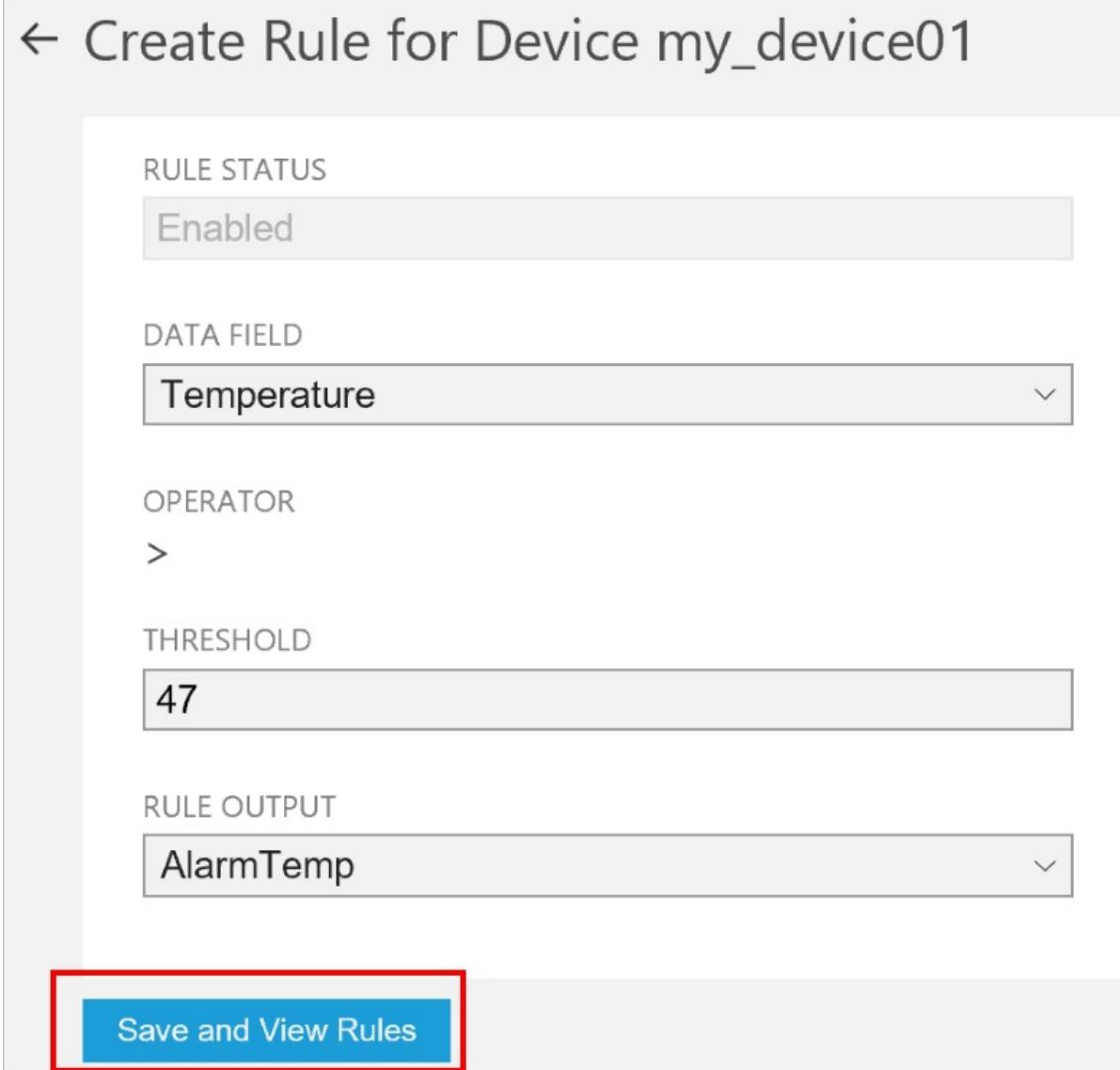
DATA FIELD
Temperature

OPERATOR
>

THRESHOLD
47

RULE OUTPUT
AlarmTemp

Save and View Rules



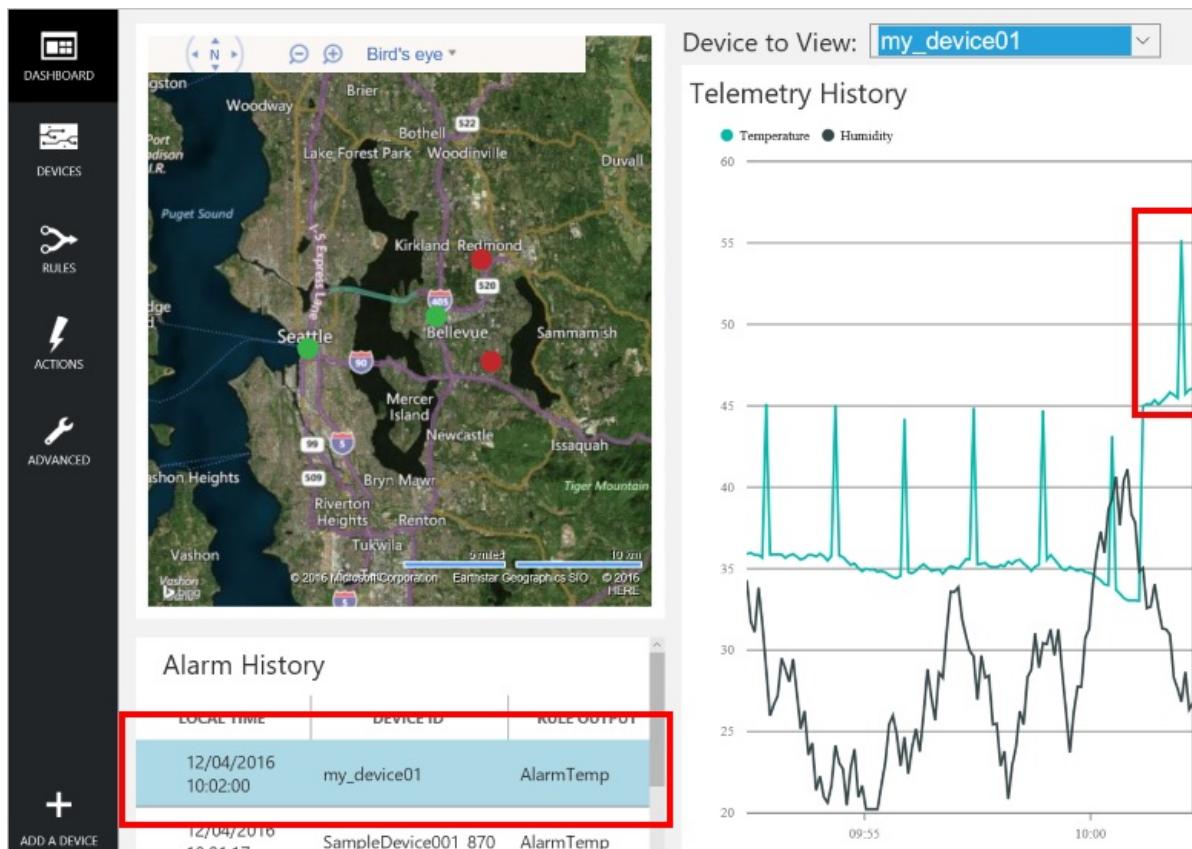
4. To save your changes, choose **Save and View Rules**.
5. Choose **Commands** in the device details pane for the new device.

The screenshot shows the Azure IoT Suite interface. On the left, there's a sidebar with icons for Dashboard, Devices, Rules, Actions, and Management Jobs. The main area is titled 'All Devices (25)' and lists 25 devices. One device, 'my_device01', is selected and highlighted with a red box. The 'Device Details' pane on the right shows a house icon, options to 'Disable Device' or 'Add Rule...', and a 'Commands' button which is also highlighted with a red box. Below that is a 'Device Twin' section with a 'Tags' dropdown containing 'Building' and 'Building 43'.

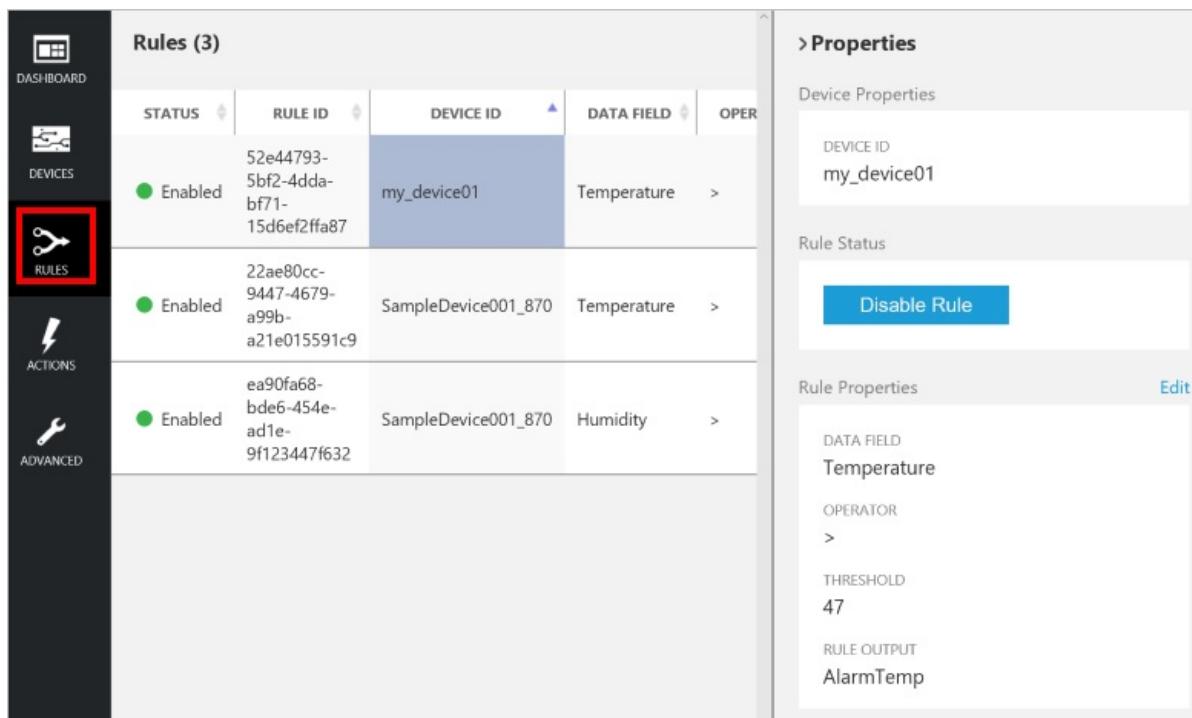
6. Select **ChangeSetPointTemp** from the command list and set **SetPointTemp** to 45. Then choose **Send Command**:

This screenshot shows the 'Commands for my_device01' page. It has a header with a back arrow and the title. Below it is a form with a red border. The form contains a 'COMMAND' section with a dropdown menu set to 'ChangeSetPointTemp'. The 'SETPOINTTEMP' section below it has an input field containing '45'. At the bottom is a blue 'Send Command' button.

7. Navigate back to the dashboard. After a short time, you will see a new entry in the **Alarm History** pane when the temperature reported by your new device exceeds the 47-degree threshold:



8. You can review and edit all your rules on the **Rules** page of the dashboard:



9. You can review and edit all the actions that can be taken in response to a rule on the **Actions** page of the dashboard:

The screenshot shows the 'Actions' section of the Azure IoT Suite interface. On the left is a sidebar with 'DASHBOARD', 'DEVICES', 'RULES', and 'ACTIONS' (which is highlighted with a red box). The main area displays 'Actions (2)' with columns for 'RULE OUTPUT', 'ACTION ID', and 'NO. OF DEVICES'. The first row shows 'AlarmHumidity' with 'Raise Alarm' and '1' device. The second row shows 'AlarmTemp' with 'Send Message' and '2' devices. To the right is a 'Properties' panel titled 'Action ID Properties' with a dropdown 'AVAILABLE ACTIONS' set to 'Send Message' and a blue 'Update' button.

[!NOTE] It is possible to define actions that can send an email message or SMS in response to a rule or integrate with a line-of-business system through a [Logic App](#). For more information, see the [Connect Logic App to your Azure IoT Suite Remote Monitoring preconfigured solution](#).

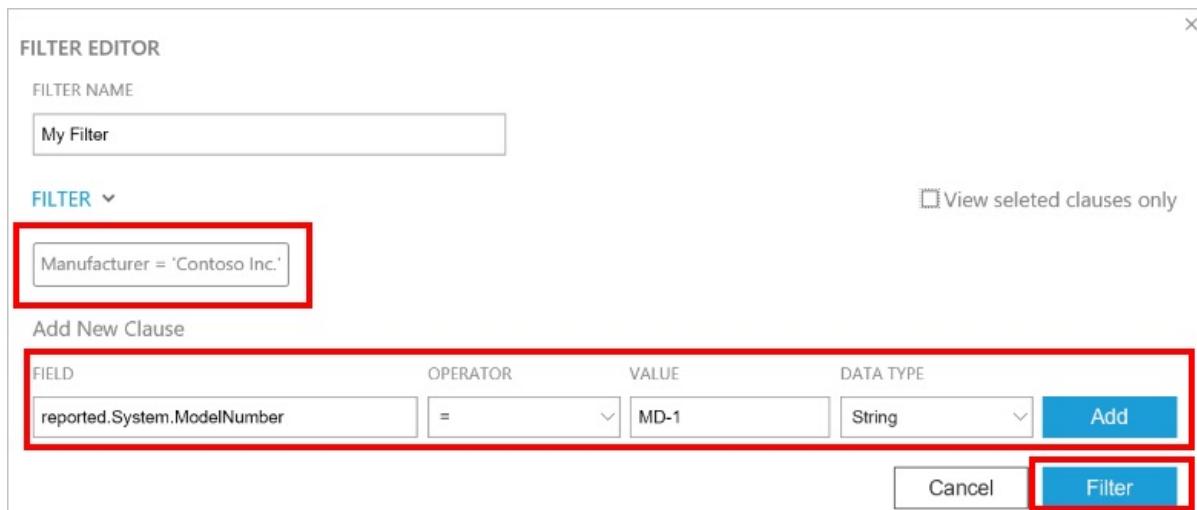
Manage filters

In the device list, you can create, save, and reload filters to display a customized list of devices connected to your hub. To create a filter:

1. Choose the edit filter icon above the list of devices:

The screenshot shows the 'All Devices (26)' list in the Azure IoT Suite. The top bar has a 'Edit Filter' button with a red box around it. The main area shows a table with columns: 'ICON', 'STATUS', 'DEVICE ID', and 'MANUFACTURER'. Three rows of data are visible: CoolingSampleDevice001_979 (Contoso Inc.), CoolingSampleDevice002_979 (Contoso Inc.), and CoolingSampleDevice003_979 (Contoso Inc.). Each row has a green circle icon and the status 'Running'.

2. In the **Filter editor**, add the fields, operators, and values to filter the device list. You can add multiple clauses to refine your filter. Choose **Filter** to apply the filter:



3. In this example, the list is filtered by manufacturer and model number:

ICON	STATUS	DEVICE ID	MANUFACTURER	MODELNUMBER
	Running	CoolingSampleDevice005_979	Contoso Inc.	MD-1
	Running	CoolingSampleDevice013_979	Contoso Inc.	MD-1
	Running	CoolingSampleDevice020_979	Contoso Inc.	MD-1

4. To save your filter with a custom name, choose the **Save as** icon:

5. To reapply a filter you saved previously, choose the **Open saved filter** icon:

The screenshot shows the 'All Devices' page with a sidebar on the left containing 'DASHBOARD', 'DEVICES' (selected), 'RULES', and 'ACTIONS'. The main area displays a table titled 'All Devices (26)' with columns: 'ICON', 'STATUS', 'DEVICE ID', and 'MANUFACTURER'. The table lists three devices, all marked as 'Running': CoolingSampleDevice001_979, CoolingSampleDevice002_979, and CoolingSampleDevice003_979, all from Contoso Inc.

You can create filters based on device id, device state, desired properties, reported properties, and tags. You add your own custom tags to a device in the **Tags** section of the **Device Details** panel, or run a job to update tags on multiple devices.

[!NOTE] In the **Filter editor**, you can use the **Advanced view** to edit the query text directly.

Commands

From the **Device Details** panel, you can send commands to the device. When a device first starts, it sends information about the commands it supports to the solution. For a discussion of the differences between commands and methods, see [Azure IoT Hub cloud-to-device options](#).

1. Choose **Commands** in the **Device Details** panel for the selected device:

The screenshot shows the 'All Devices' page with a sidebar on the left containing 'DASHBOARD', 'DEVICES' (selected), 'RULES', 'ACTIONS', 'MANAGEMENT JOBS', and 'ADVANCED'. The main area displays a table titled 'All Devices (26)' with columns: 'ICON', 'STATUS', 'DEVICE ID', and 'MANUFACTURER'. The table lists 11 devices, all marked as 'Running'. On the right, the 'DEVICE DETAILS' panel is open for 'CoolingSampleDevice001_979'. It shows a circular icon with a thermometer and a gear, and a list of actions: 'Disable Device', 'Add Rule...', 'Commands' (which is highlighted with a red box), and 'Methods'. Below this is the 'Device Twin' section, which includes a 'Tags' table with entries: Building 2, Floor 1F, and HubEnabledState Running.

2. Select **PingDevice** from the command list.
3. Choose **Send Command**.
4. You can see the status of the command in the command history.

← Commands for SampleDevice002_170

COMMAND
Select A Command

Command History

COMMAND NAME	RESULT	VALUES SENT	LOCAL TIME CREATED	LOCAL TIME UPDATED
PingDevice	Success	{}	03/12/2015, 14:50:09	03/12/2015, 14:50:10

Resend

The screenshot shows the 'Commands' section for a device named 'SampleDevice002_170'. At the top, there's a dropdown menu labeled 'COMMAND' with the option 'Select A Command'. Below it is a table titled 'Command History' with columns: COMMAND NAME, RESULT, VALUES SENT, LOCAL TIME CREATED, and LOCAL TIME UPDATED. A single row is present in the table, representing a command named 'PingDevice' that was successful ('Success') and sent an empty JSON object ('{}'). The local time for creation and update is '03/12/2015, 14:50:09'. A blue 'Resend' button is located at the bottom right of the table.

The solution tracks the status of each command it sends. Initially the result is **Pending**. When the device reports that it has executed the command, the result is set to **Success**.

Behind the scenes

When you deploy a preconfigured solution, the deployment process creates multiple resources in the Azure subscription you selected. You can view these resources in the [Azure portal](#). The deployment process creates a **resource group** with a name based on the name you choose for your preconfigured solution:

The screenshot shows the Azure IoT Suite portal interface. At the top, the resource group name 'rmpreconf' is highlighted with a red box. Below it, the 'Overview' tab is selected, indicated by a blue background. On the left sidebar, under 'SETTINGS', there are several options: Quickstart, Resource costs, Deployments, Properties, Locks, and Automation script. Under 'SUPPORT + TROUBLESHOOTING', there is a New support request link. The main content area displays the 'Essentials' section with subscription details: Subscription name 'Visual Studio Ultimate with MSDN', Last deployment '8/17/2016 (Succeeded)', Subscription ID '<your subscription id>', and Location 'East US'. Below this is a table listing resources, with a red box highlighting the entire list. The table columns are NAME, TYPE, and LOCATION. The listed resources are:

NAME	TYPE	LOCATION
rmpreconf-map	Bing Maps AP...	West US
rmpreconf	IoT Hub	East US
rmpreconf	DocumentDB...	East US
rmpreconf	Service Bus	East US
rmpreconf	Storage accou...	East US
rmpreconf-DeviceInfo	Stream Analyt...	East US
rmpreconf-Rules	Stream Analyt...	East US
rmpreconf-Telemetry	Stream Analyt...	East US
rmpreconf-jobsplan	App Service pl...	East US
rmpreconf-plan	App Service pl...	East US
rmpreconf	App Service	East US
rmpreconf-jobhost	App Service	East US

You can view the settings of each resource by selecting it in the list of resources in the resource group.

You can also view the source code for the preconfigured solution. The remote monitoring preconfigured solution source code is in the [azure-iot-remote-monitoring](#) GitHub repository:

- The **DeviceAdministration** folder contains the source code for the dashboard.
- The **Simulator** folder contains the source code for the simulated device.
- The **EventProcessor** folder contains the source code for the back-end process that handles the incoming telemetry.

When you are done, you can delete the preconfigured solution from your Azure subscription on the [azureiotsuite.com](#) site. This site enables you to easily delete all the resources that were provisioned when you created the preconfigured solution.

[!NOTE] To ensure that you delete everything related to the preconfigured solution, delete it on the [azureiotsuite.com](#) site and do not delete the resource group in the portal.

Next Steps

Now that you've deployed a working preconfigured solution, you can continue getting started with IoT Suite by reading the following articles:

- [Remote monitoring preconfigured solution walkthrough](#)
- [Connect your device to the remote monitoring preconfigured solution](#)
- [Permissions on the azureiotsuite.com site](#)