

Microcontrollers and Azure

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-arduino-huzzah-esp8266-get-started>

Step 1 – Install dev environment

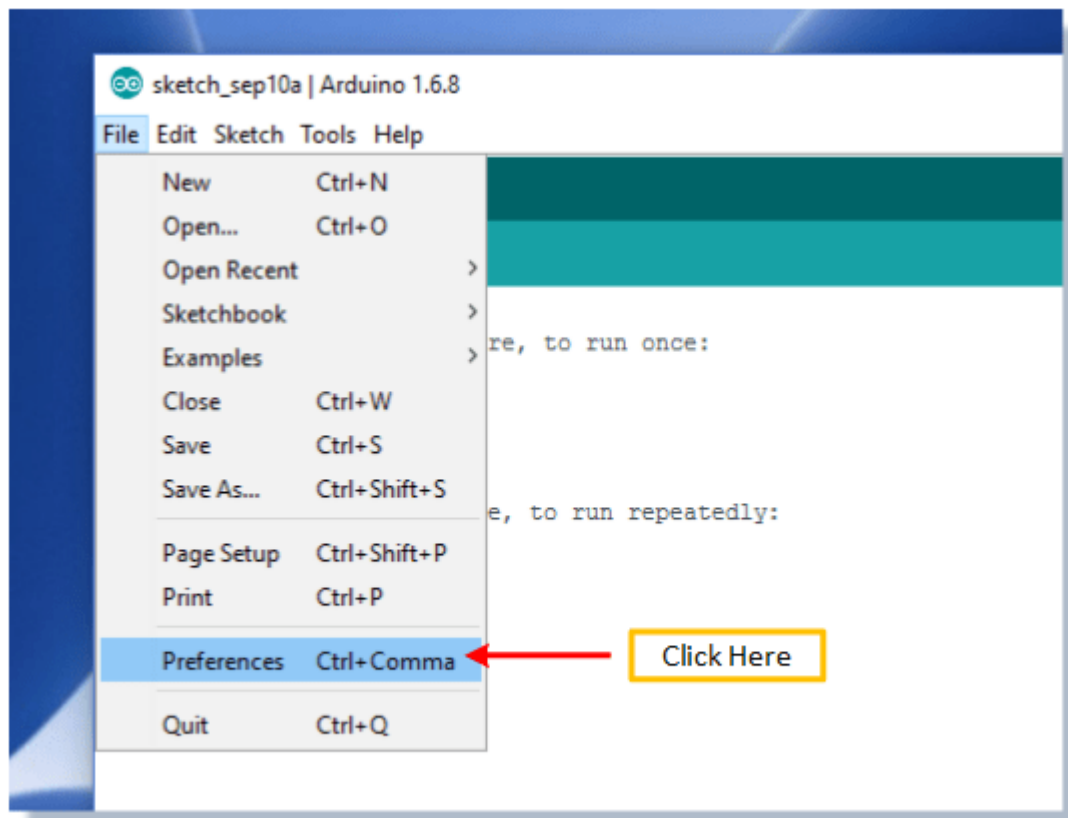
Install Arduino IDE: https://www.arduino.cc/download_handler.php

(From blog: <http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips-and-more/arduino-esp8266-lolin-nodemcu-getting-started/>)

Preparing the Arduino IDE to Work with the NodeMCU Module

Go To 'Preferences'

Select "Preferences" via the File Menu.

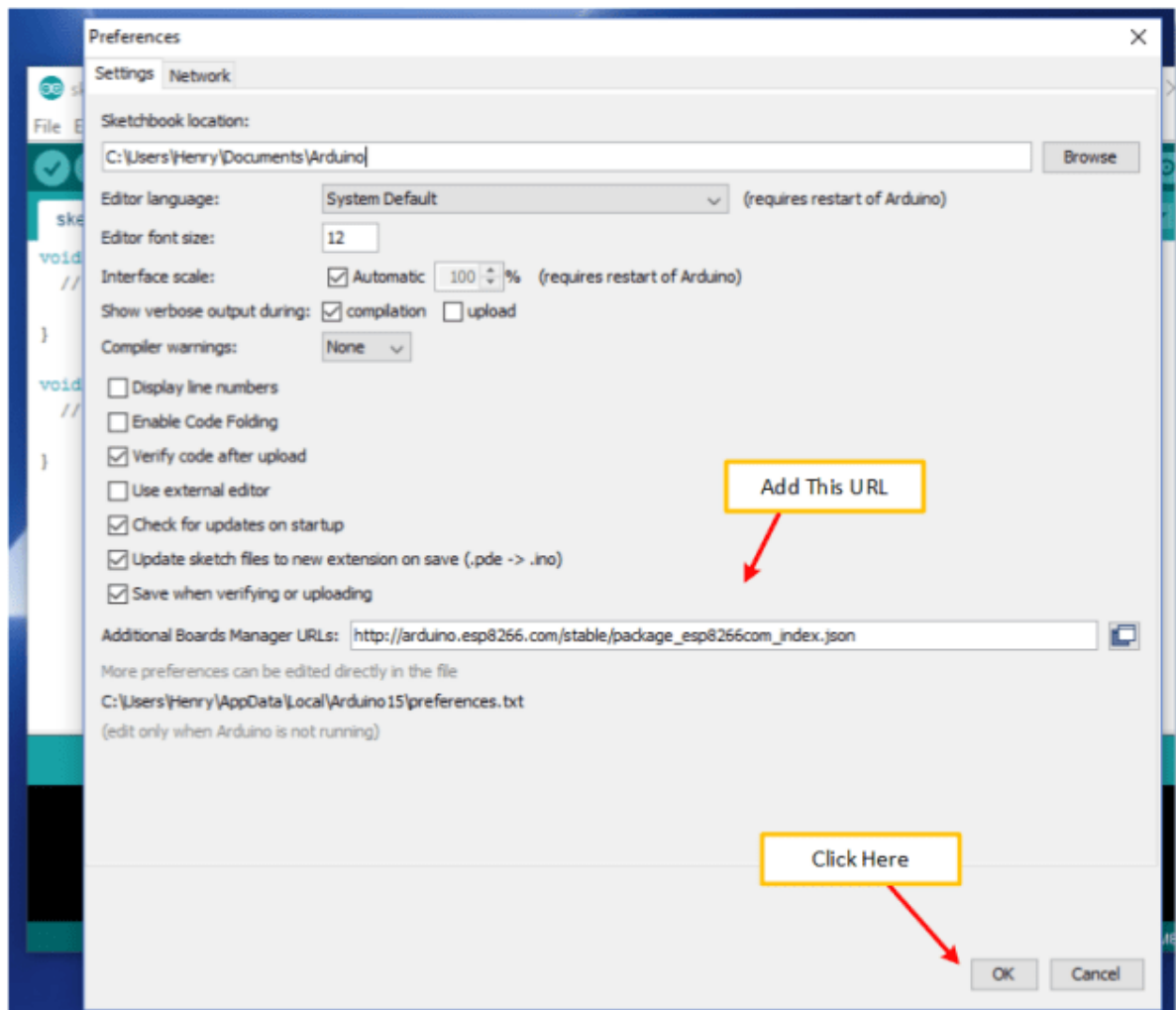


Add A URL

Type

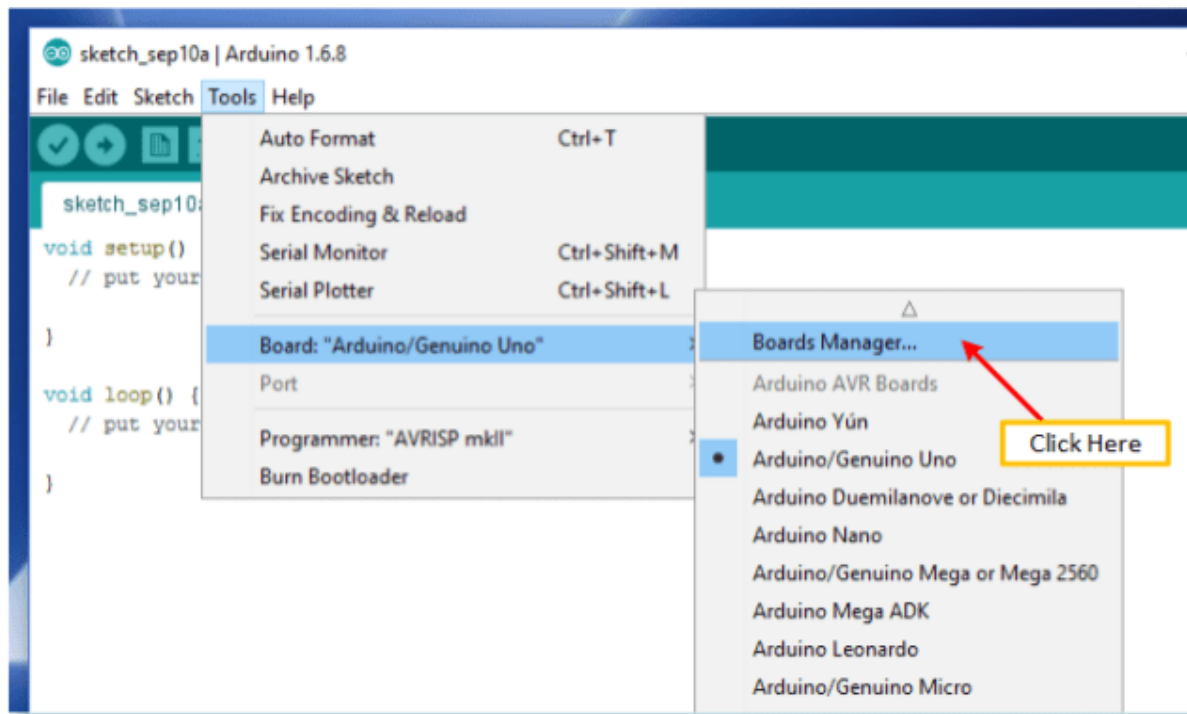
`"http://arduino.esp8266.com/stable/package_esp8266com_index.json"`

into the field for 'Additional Boards Manager URL'.



Access 'Board Manager'

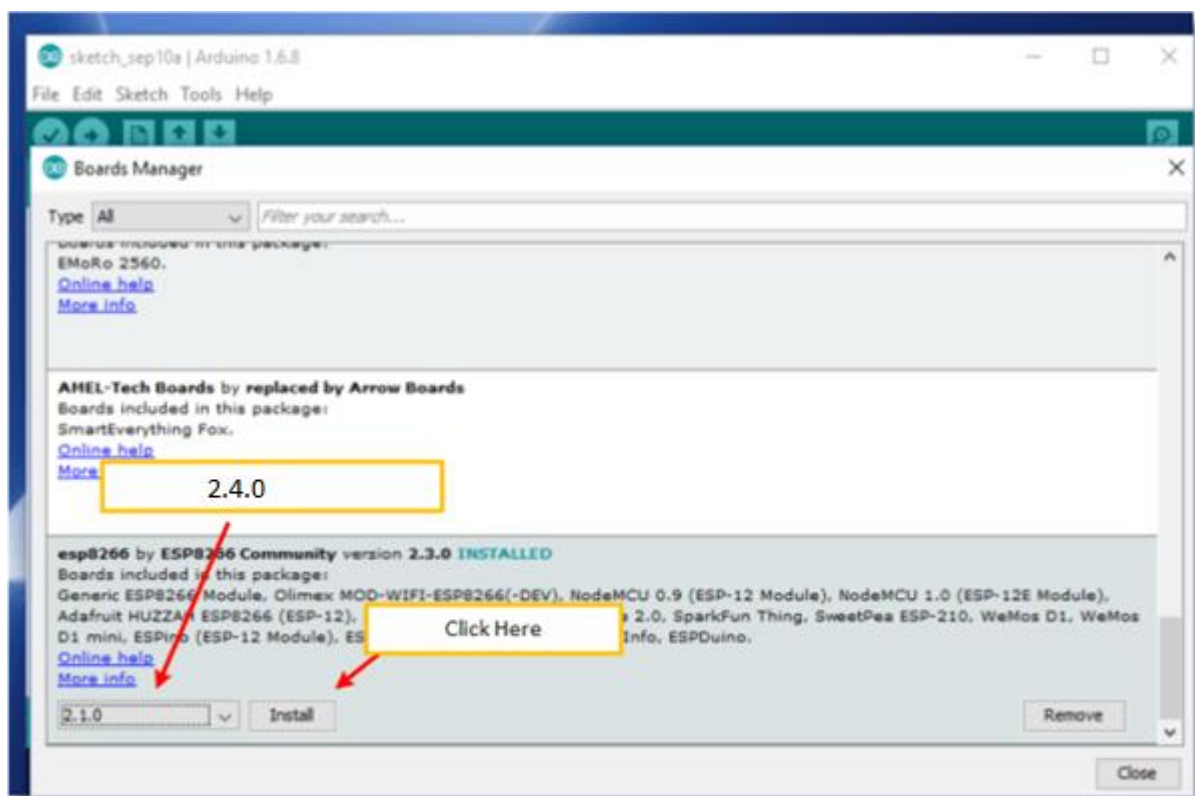
Select "Boards Manager" via the Tool Menu.



Install the ESP8266 Files

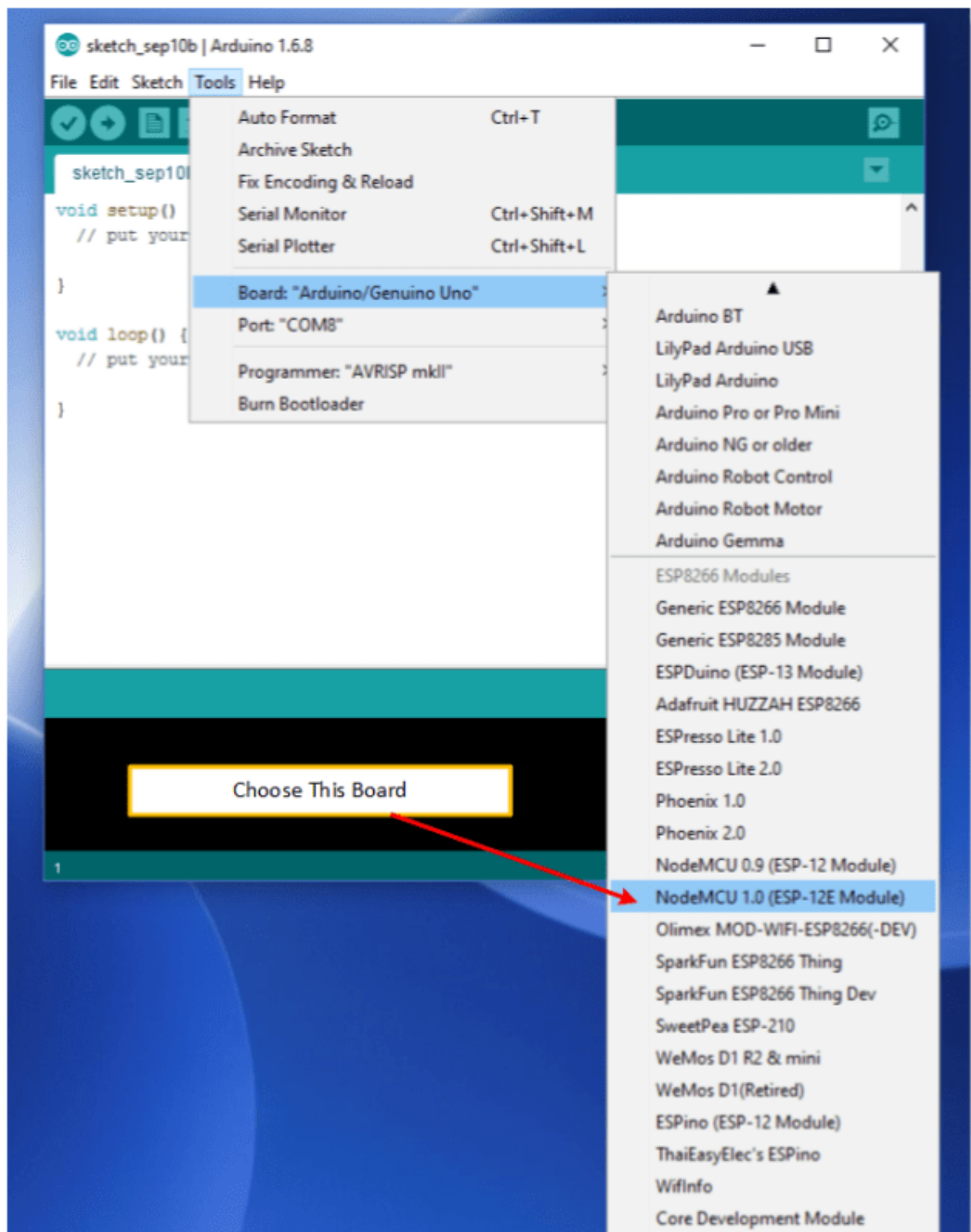
From the boards manager pop up, scroll until you find the esp8266 board. Select the latest version and install.

IMPORTANT SELECT VERSION 2.4.0



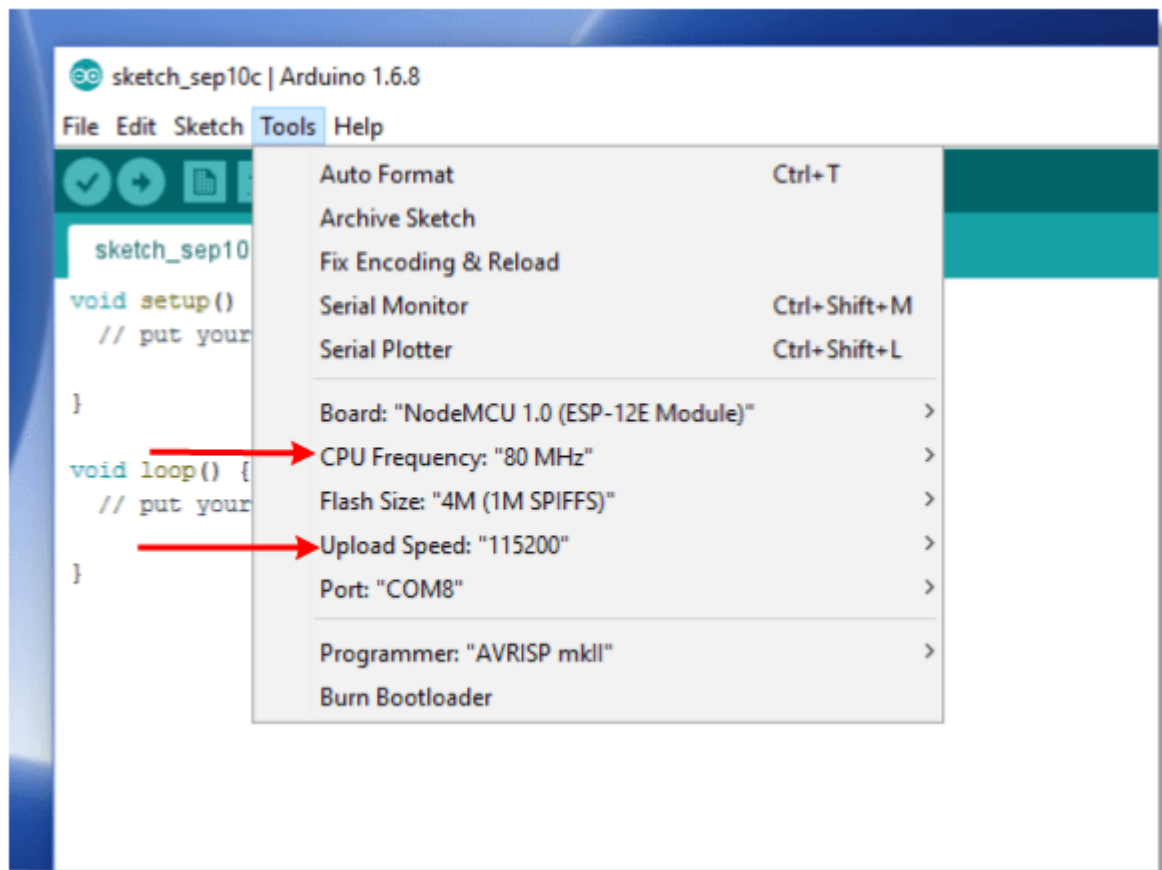
Select the NodeMCU V1.0 ESP8266-12E Board

The board is selected via the 'Tools' menu.



Select the CPU Frequency and Upload Speed

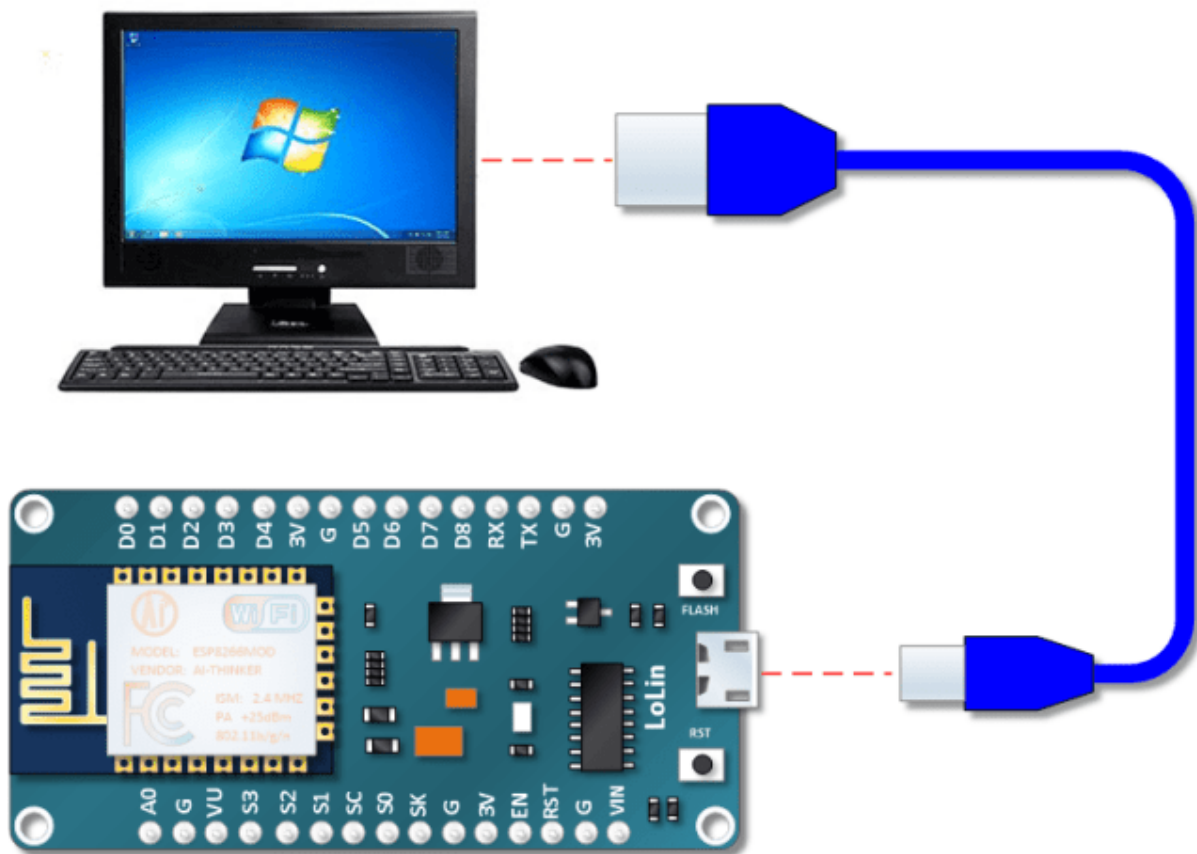
Match the picture below.



Connect the LoLin NodeMCU Module to your PC

First Time Connecting

If this is your first time connecting your development board, you **may** simply need to connect the ESP8266-12E module to your computer via a micro USB cable. Otherwise, there may be a program running that consumes more current than the USB can provide. In that case, you may want to use connect your own supply.



Hello World

Blink the onboard Led

```
#define LED_BUILTIN 2

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);    // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW is the
  voltage level                      // but actually the LED is on; this is because
                                     // it is active low on the ESP-01)
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the
  active low LED)
}
```


Scan Wifi

See the available wifi networks

```
#include "ESP8266WiFi.h"

void setup() {
  Serial.begin(115200);
  // Set WiFi to station mode and disconnect from an AP if it was previously
  connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(2000);
  Serial.println("Setup done");
}

void loop() {
  Serial.println("scan start");

  int n = WiFi.scanNetworks();// WiFi.scanNetworks will return the number of
  networks found
  Serial.println("scan done");
  if (n == 0)
    Serial.println("no networks found");
  else
  {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i)
    {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)? " ":"*");
      delay(10);
    }
  }
  Serial.println("");

  // Wait a bit before scanning again
  delay(5000);
}
```

Azure Setup

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-arduino-huzzah-esp8266-get-started>

Create an IoT hub

This section describes how to create an IoT hub using the [Azure portal](#).

1. Log in to the [Azure portal](#).
2. Choose **+Create a resource**, then *Search the Marketplace* for the **IoT Hub**.
3. Select **IoT Hub** and click the **Create** button. You see the first screen for creating an IoT hub.

The screenshot shows the Azure portal interface for creating an IoT hub. The breadcrumb navigation at the top reads 'Home > New > IoT hub'. The main heading is 'IoT hub' with 'Microsoft' underneath. There are three tabs: 'Basics' (selected), 'Size and scale', and 'Review + create'. Below the tabs, there is a description: 'Create an IoT Hub to help you connect, monitor, and manage billions of your IoT assets. [Learn More](#)'. The section is titled 'PROJECT DETAILS' with a sub-instruction: 'Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.' There are four fields, each with a red asterisk and an information icon: 'Subscription' (dropdown menu showing 'Microsoft Azure Internal Consumption'), 'Resource Group' (radio buttons for 'Create new' (selected) and 'Use existing', with a text input field containing 'contoso-hub-rg' and a green checkmark), 'Region' (dropdown menu showing 'West US'), and 'IoT Hub Name' (text input field containing 'contoso-test-hub' and a green checkmark). At the bottom, there are three buttons: 'Review + create' (blue), 'Next: Size and scale »' (white with a red border and red highlight), and 'Automation options' (blue link).

Fill in the fields.

Subscription: Select the subscription to use for your IoT hub.

Resource Group: You can create a new resource group or use an existing one. To create a new one, click **Create new** and fill in the name you want to use. To use an existing resource group, click **Use existing** and select the resource

group from the dropdown list. For more information, see [Manage Azure Resource Manager resource groups](#).

Region: This is the region in which you want your hub to be located. Select the location closest to you from the dropdown list.

IoT Hub Name: Put in the name for your IoT Hub. This name must be globally unique. If the name you enter is available, a green check mark appears.

Important

The IoT hub will be publicly discoverable as a DNS endpoint, so make sure to avoid any sensitive information while naming it.

4. Click **Next: Size and scale** to continue creating your IoT hub.

The screenshot shows the 'Size and scale' configuration page for an IoT Hub in the Azure portal. The breadcrumb navigation at the top reads 'Home > New > IoT hub'. The page title is 'IoT hub' with the Microsoft logo. Below the title are three tabs: 'Basics', 'Size and scale' (which is selected), and 'Review + create'. A descriptive text states: 'Each IoT Hub is provisioned with a certain number of units in a specific tier. The tier and number of units determine the maximum daily quota of messages that you can send. [Learn more](#)'. Under the heading 'SCALE TIER AND UNITS', there is a section for 'Pricing and scale tier' with a dropdown menu set to 'S1: Standard tier' and a link to 'Learn how to choose the right IoT Hub tier for your solution'. Below this is a slider for 'Number of S1 IoT Hub units' set to 1, with a note: 'This determines your IoT Hub scale capability and can be changed as your need increases.' A list of features is shown, all set to 'Enabled': 'Device-to-cloud-messages', 'Message routing', 'Cloud-to-device commands', 'IoT Edge', and 'Device management'. An 'Advanced Settings' section is expanded, showing a slider for 'Device-to-cloud partitions' set to 4. At the bottom, there are three buttons: 'Review + create' (in blue), '« Previous: Basics', and 'Automation options'.

On this screen, you can take the defaults and just click **Review + create** at the bottom.

Pricing and scale tier: You can choose from several tiers depending on how many features you want and how many messages you send through your solution per day. The free tier is intended for testing and evaluation. It allows 500 devices to be connected to the IoT hub and up to 8,000 messages per day. Each Azure subscription can create one IoT Hub in the free tier.

IoT Hub units: The number of messages allowed per unit per day depends on your hub's pricing tier. For example, if you want the IoT hub to support ingress of 700,000 messages, you choose two S1 tier units.

For details about the other tier options, see [Choosing the right IoT Hub tier](#).

Advanced / Device-to-cloud partitions: This property relates the device-to-cloud messages to the number of simultaneous readers of the messages. Most IoT hubs only need four partitions.

5. Click **Review + create** to review your choices. You see something similar to this screen.

Home > New > IoT hub

IoT hub

Microsoft

Basics Size and scale **Review + create**

BASICS

Subscription ⓘ	Microsoft Azure Internal Consumption
Resource Group ⓘ	contoso-hub-rgrp
Region ⓘ	West US
IoT Hub Name ⓘ	contoso-test-hub

SIZE AND SCALE

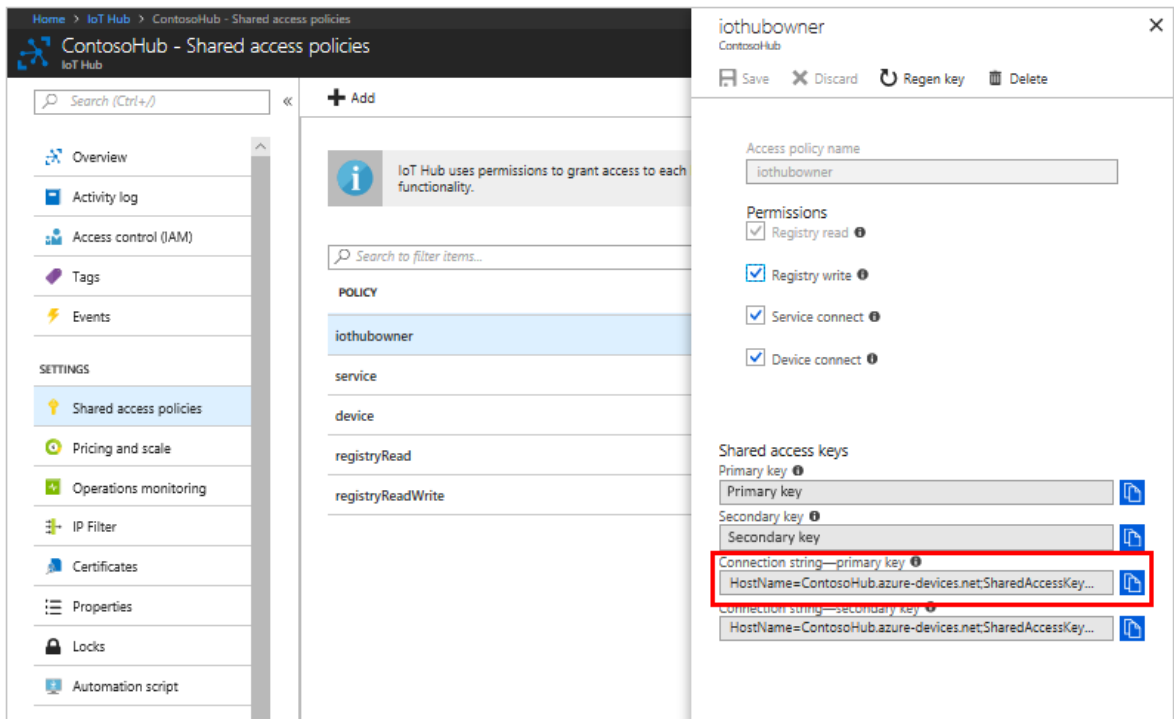
Pricing and scale tier ⓘ	S1
Number of S1 IoT Hub units ⓘ	1
Messages per day ⓘ	400,000
Cost per month	25.00 USD

Create « Previous: Size and scale Automation options

6. Click **Create** to create your new IoT hub. Creating the hub takes a few minutes. Retrieve connection string for IoT hub

After your hub has been created, retrieve the connection string for the hub. This is used to connect devices and applications to your hub.

1. Click on your hub to see the IoT Hub pane with Settings, and so on. Click **Shared access policies**.
2. In **Shared access policies**, select the **iothubowner** policy.
3. Under **Shared access keys**, copy the **Connection string -- primary key** to be used later.

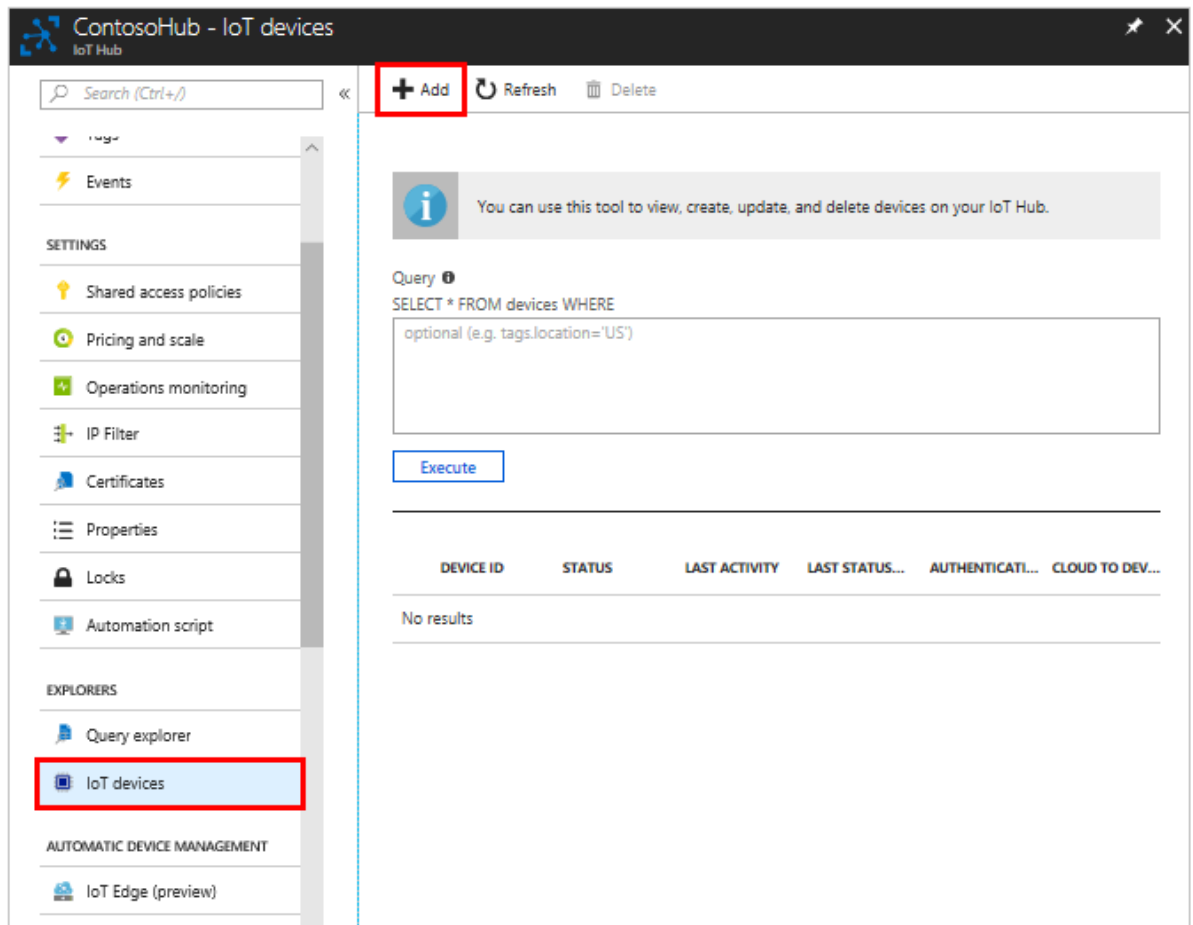


For more information, see [Access control](#) in the "IoT Hub developer guide."


Register a new device in the IoT hub


In this section, you create a device identity in the identity registry in your IoT hub. A device cannot connect to IoT hub unless it has an entry in the identity registry. For more information, see the "Identity registry" section of the [IoT Hub developer guide](#)


1. In your IoT hub navigation menu, open **IoT Devices**, then select **Add** to register a new device in your IoT hub.





2. Provide a name for your new device, such as **myDeviceId**, and select **Save**. This action creates a new device identity for your IoT hub.


 **Create a device** □ ×




[Learn more about creating devices](#) 


*** Device ID** 


 

Authentication type 


Symmetric key X.509 Self-Signed X.509 CA Signed

*** Primary key** 


*** Secondary key** 

Auto-generate keys 

☒

Connect this device to an IoT hub 

Enable Disable

Parent device (Preview) 

No parent device

[Set a parent device](#)

Save

Important

The device ID may be visible in the logs collected for customer support and troubleshooting, so make sure to avoid any sensitive information while naming it.

- After the device is created, open the device from the list in the **IoT devices** pane. Copy the **Connection string---primary key** to use later.

The screenshot shows the 'Device details' page for a device named 'myDeviceId'. The page includes a toolbar with options like 'Save', 'Message to device', 'Direct method', 'Device twin', 'Add module identity', 'Regenerate keys', and 'Refresh'. The main content area displays the following information:

- Device Id:** myDeviceId
- Primary key:** <Primary Key>
- Secondary key:** <Secondary Key>
- Connection string (primary key):** HostName=ContosoHub.azure-devices.net;DeviceId=myDeviceId;SharedAccessKey=<Primary Key> (This row is highlighted with a red box in the original image)
- Connection string (secondary key):** HostName=ContosoHub.azure-devices.net;DeviceId=myDeviceId;SharedAccessKey=<Secondary Key>
- Connect this device to an IoT hub:** Buttons for 'Enable' and 'Disable'.
- Parent device (Preview):** No parent device, with a 'Set a parent device' button.

Note

The IoT Hub identity registry only stores device identities to enable secure access to the IoT hub. It stores device IDs and keys to use as security credentials, and an enabled/disabled flag that you can use to disable access for an individual device. If your application needs to store other device-specific metadata, it should use an application-specific store. For more information, see [IoT Hub developer guide](#).

Setup Sample code for device

Get the sample application from GitHub

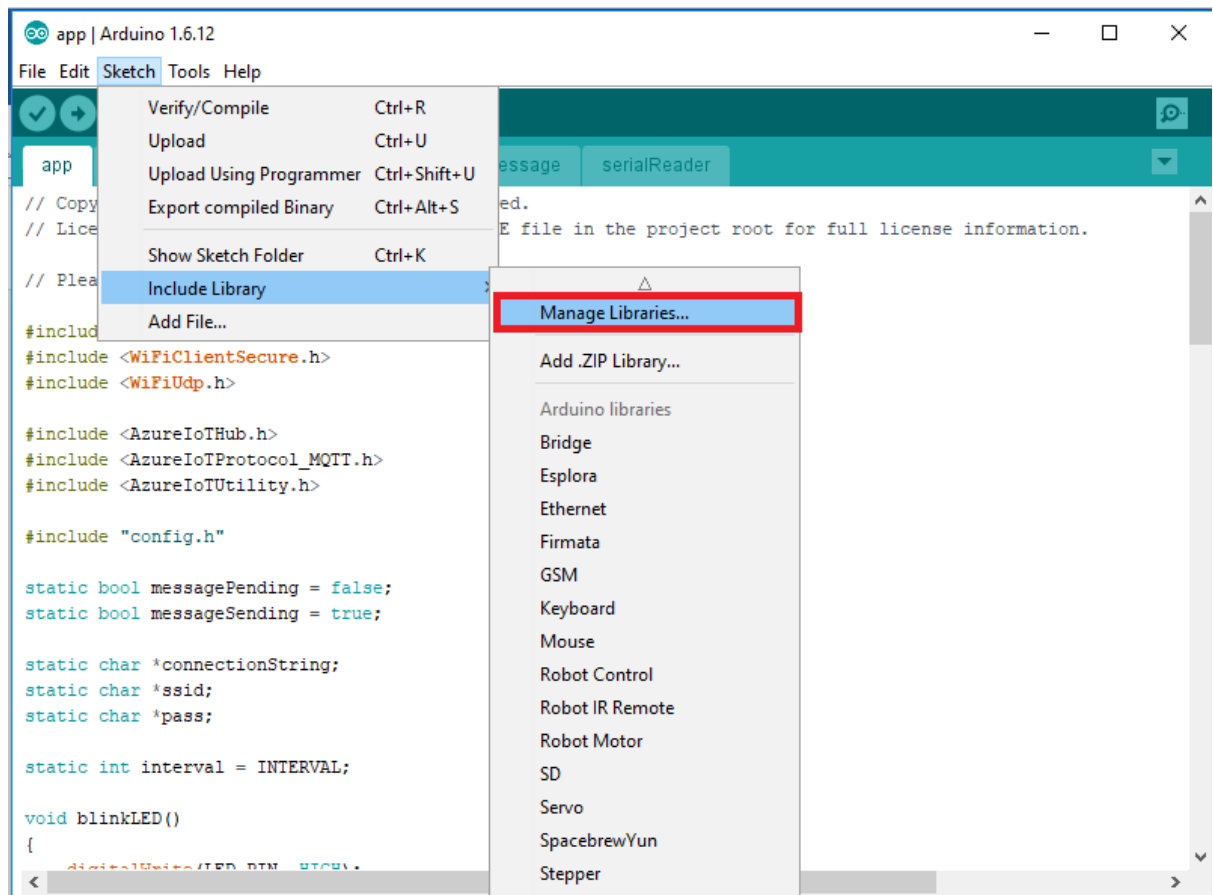
The sample application is hosted on GitHub. Clone the sample repository that contains the sample application from GitHub. To clone the sample repository, follow these steps:

1. Open a command prompt or a terminal window.
2. Go to a folder where you want the sample application to be stored.
3. Run the following command:

```
git clone https://github.com/Azure-Samples/iot-hub-feather-huzzah-client-app.git
```
4. Open the folder where the sample application is stored.
5. Open the app.ino file in the app folder in the Arduino IDE.

Install necessary libraries

1. In the Arduino IDE select : **Sketch → Include libraries → Manage libraries.**



2. Search for the following library names one by one. For each library that you find, click **Install**.

- AzureIoTHub
- AzureIoTUtility
- AzureIoTProtocol_MQTT
- ArduinoJson ← Select version 5.13.5
- DHT sensor library
- Adafruit Unified Sensor

3. Open the `config.h` file in the `app` folder.

// Physical device information for board and sensor

#define DEVICE_ID "**myDeviceId**" ← Change to match setting in Azure

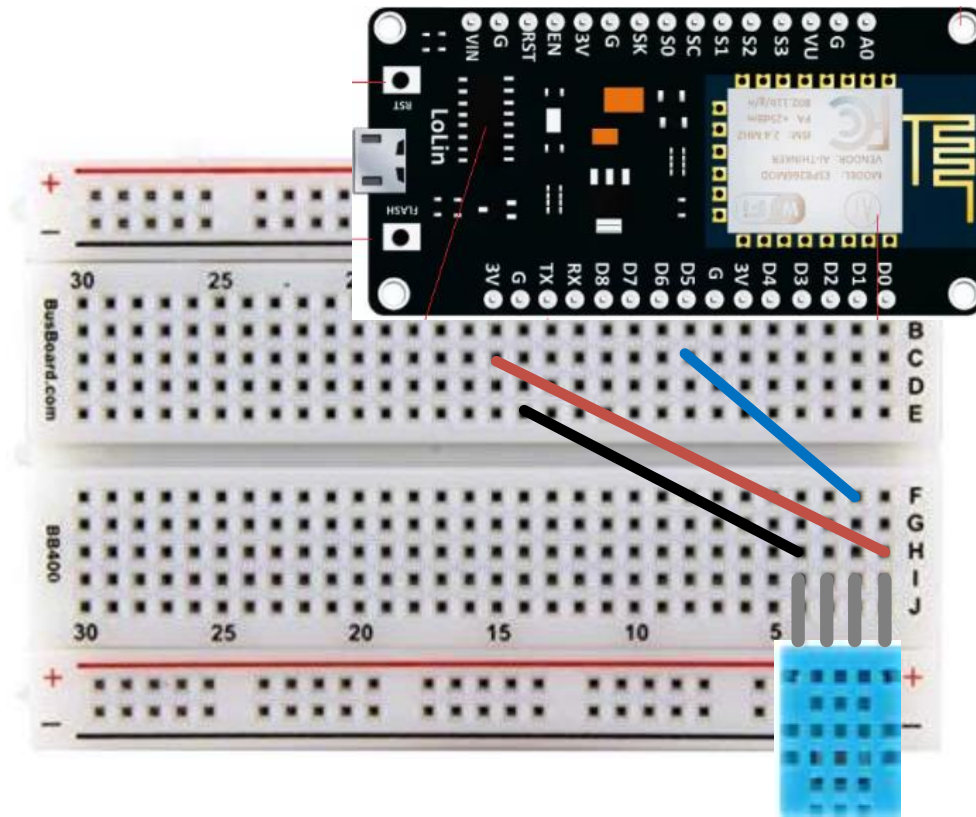
#define DHT_TYPE DHT11 ← Change DHT22 to DHT11

// Pin layout configuration

#define LED_PIN 2

#define DHT_PIN 14

4. Save the file
5. Deploy code to device
6. Attach DHT11 to device



Deploy the sample application to ESP8266

After the upload completes successfully, follow these steps to enter your credentials:

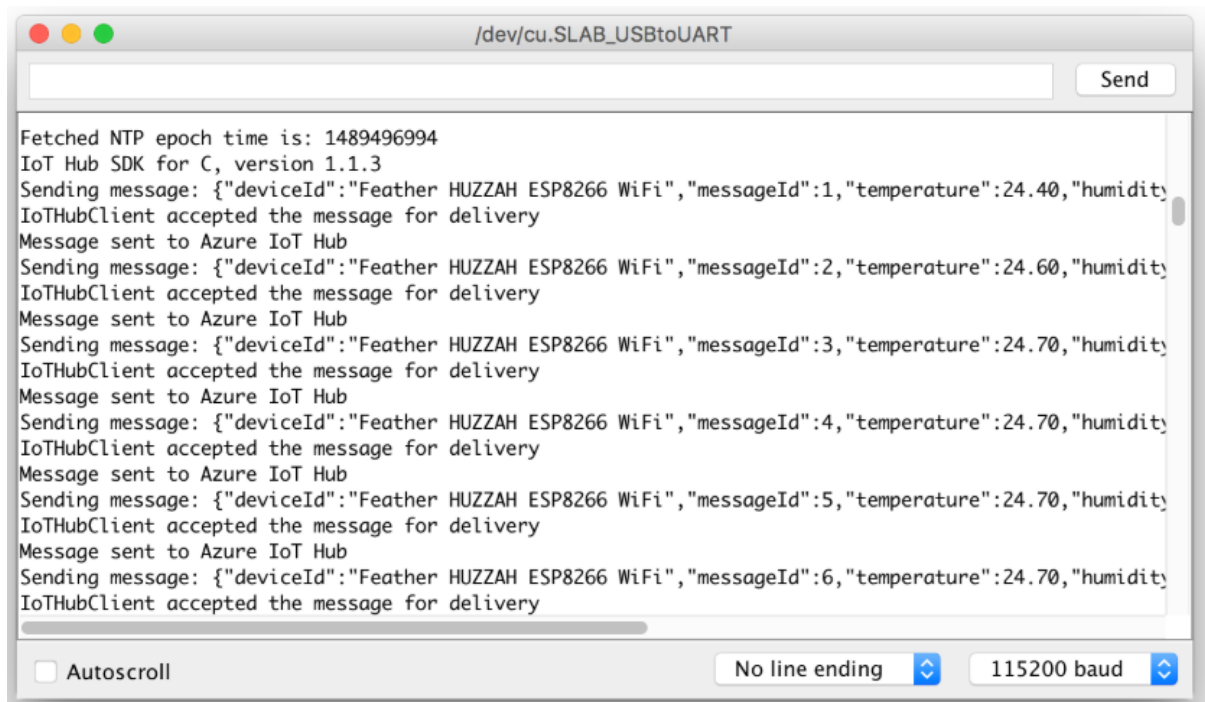
1. Open Arduino IDE, click **Tools > Serial Monitor**.
2. In the serial monitor window, notice the two drop-down lists in the lower-right corner.
3. Select **No line ending** for the left drop-down list.
4. Select **115200 baud** for the right drop-down list.
5. In the input box located at the top of the serial monitor window, enter the following information if you are asked to provide them, and then click **Send**.
 - Wi-Fi SSID
 - Wi-Fi password
 - Device connection string

Note

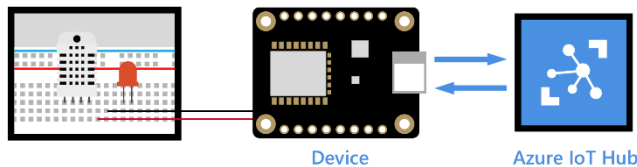
The credential information is stored in the EEPROM of ESP8266. If you click the reset button on the ESP8266 board, the sample application asks if you want to erase the information. Enter **Y** to have the information erased. You are asked to provide the information a second time.

Verify the sample application is running successfully

If you see the following output from the serial monitor window and the blinking LED on ESP8266, the sample application is running successfully.



Use Azure IoT Tools for Visual Studio Code to send and receive messages between your device and IoT Hub



[Azure IoT Tools](#) is a useful Visual Studio Code extension that makes IoT Hub management and IoT application development easier. This article focuses on how to use Azure IoT Tools for Visual Studio Code to send and receive messages between your device and your IoT hub.

Note

Some of the features mentioned in this article, like cloud-to-device messaging, device twins, and device management, are only available in the standard tier of IoT hub. For more information about the basic and standard IoT Hub tiers, see [How to choose the right IoT Hub tier](#).

What you will learn

You learn how to use Azure IoT Tools for Visual Studio Code 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 Azure IoT Tools for Visual Studio Code to monitor device-to-cloud messages.
- Use Azure IoT Tools for Visual Studio Code to send cloud-to-device messages.

What you need

- An active Azure subscription.
- An Azure IoT hub under your subscription.
- [Visual Studio Code](#)

- [Azure IoT Tools for VS Code](#) or [open this link in Visual Studio Code](#).

Sign in to access your IoT hub

1. In **Explorer** view of VS Code, expand **Azure IoT Hub Devices** section in the bottom left corner.
2. Click **Select IoT Hub** in context menu.
3. A pop-up will show in the bottom right corner to let you sign in to Azure for the first time.
4. After you sign in, your Azure Subscription list will be shown, then select Azure Subscription and IoT Hub.
5. The device list will be shown in **Azure IoT Hub Devices** tab in a few seconds.

Note

You can also complete the set up by choosing **Set IoT Hub Connection String**. Enter the connection string for the IoT hub that your IoT device connects to in the pop-up window.

Monitor device-to-cloud messages

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

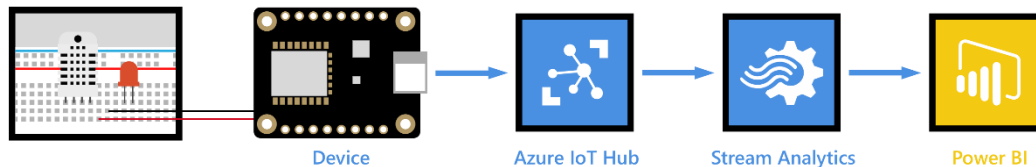
1. Right-click your device and select **Start Monitoring D2C Message**.
2. The monitored messages will be shown in **OUTPUT > Azure IoT Hub Toolkit** view.
3. To stop monitoring, right-click the **OUTPUT** view and select **Stop Monitoring D2C Message**.

Send cloud-to-device messages

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

1. Right-click your device and select **Send C2D Message to Device**.
2. Enter the message in input box. ("start" or "stop")
3. Results will be shown in **OUTPUT > Azure IoT Hub Toolkit** view.

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



What you learn

You learn how to visualize real-time sensor data that your Azure IoT hub receives by using Power BI. If you want to try to 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

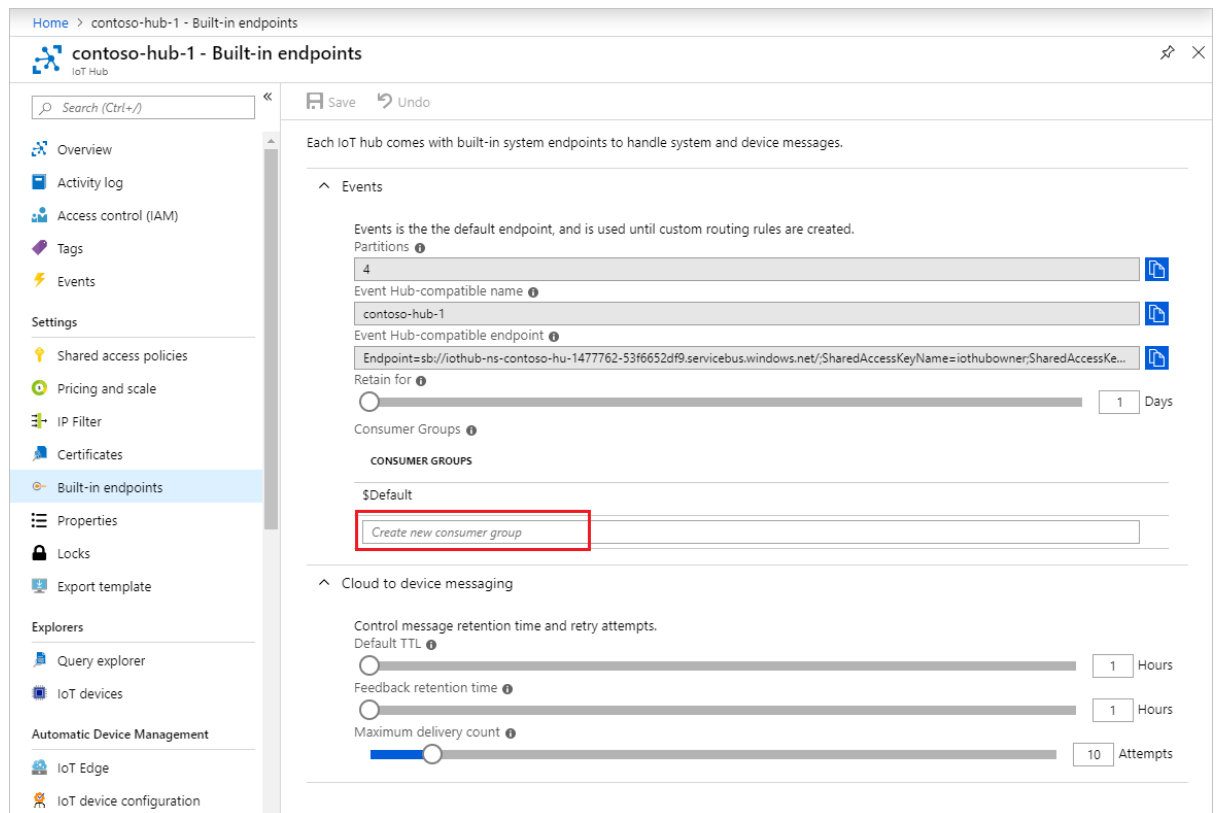
- Complete the [Raspberry Pi online simulator](#) tutorial or one of the device tutorials; for example, [Raspberry Pi with node.js](#). These cover the following requirements:
 - An active Azure subscription.
 - An Azure IoT hub under your subscription.
 - A client application that sends messages to your Azure IoT hub.
- A Power BI account. ([Try Power BI for free](#))

Add a consumer group to your IoT hub

[Consumer groups](#) provide independent views into the event stream that enable apps and Azure services to independently consume data from the same Event Hub endpoint. In this section, you add a consumer group to your IoT hub's built-in endpoint that is used later in this tutorial to pull data from the endpoint.

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

1. In the [Azure portal](#), open your IoT hub.
2. On the left pane, select **Built-in endpoints**, select **Events** on the right pane, and enter a name under **Consumer groups**. Select **Save**.



Create, configure, and run a Stream Analytics job

Let's start by creating a Stream Analytics job. After you create the job, you define the inputs, outputs, and the query used to retrieve the data.

Create a Stream Analytics job

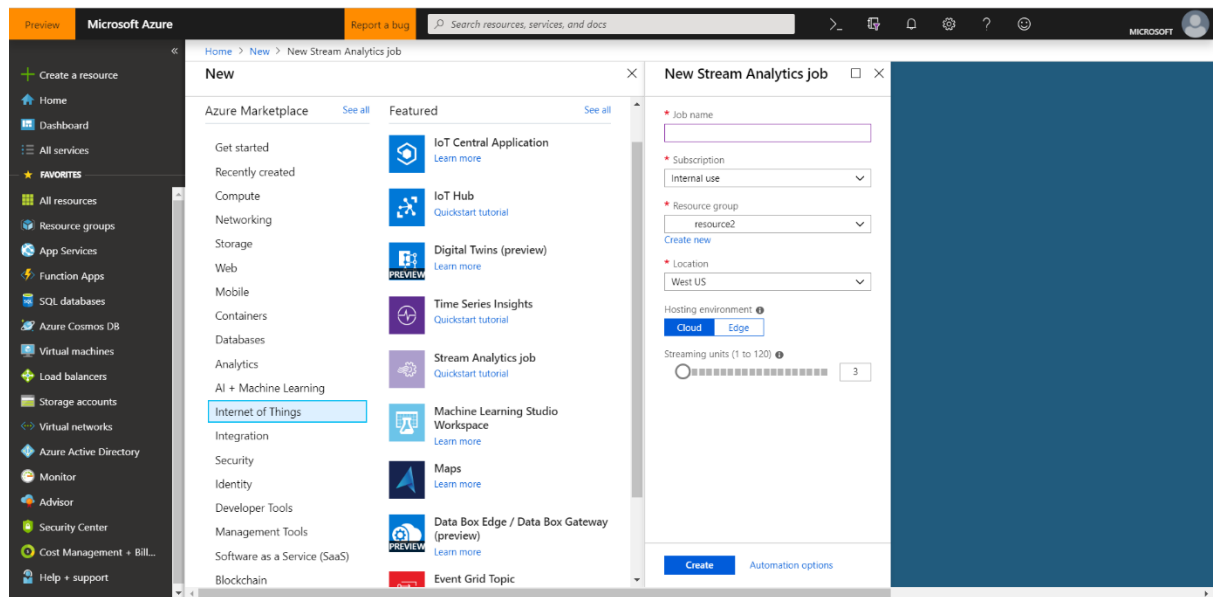
1. In the [Azure portal](#), click **Create a resource** > **Internet of Things** > **Stream Analytics job**.
2. Enter the following information for the job.

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 stream input**, and then enter the following information:

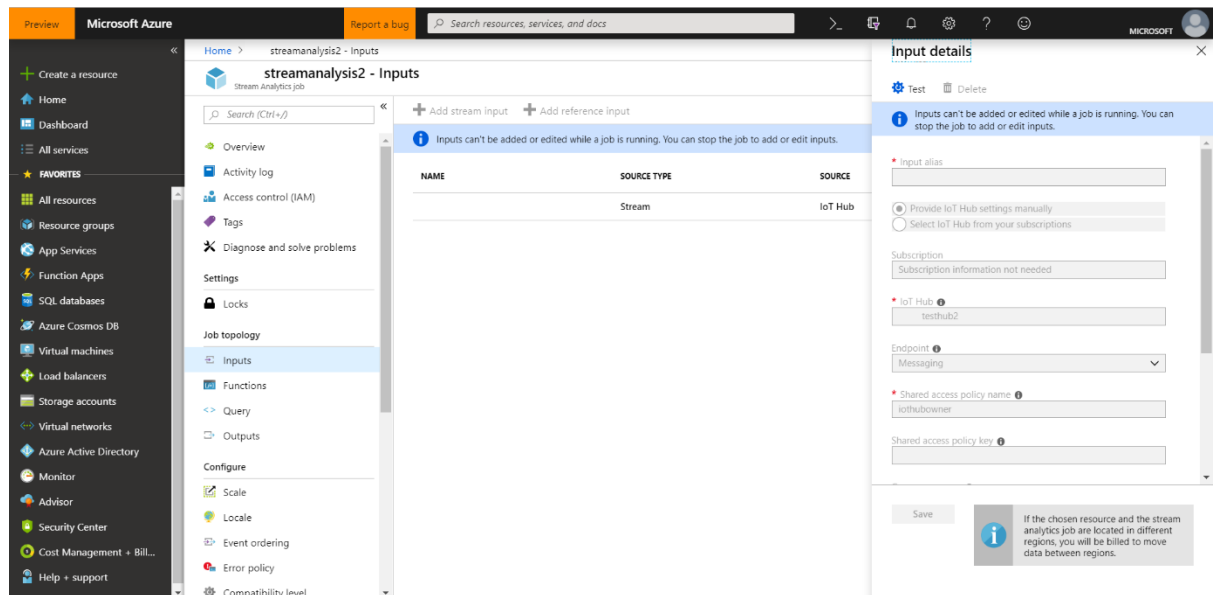
Input alias: The unique alias for the input and select **Provide IoT Hub settings manually** below.

Source: Select **IoT hub**.

Endpoint: Click **Messaging**.

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 **Power BI**, and then enter the following information:

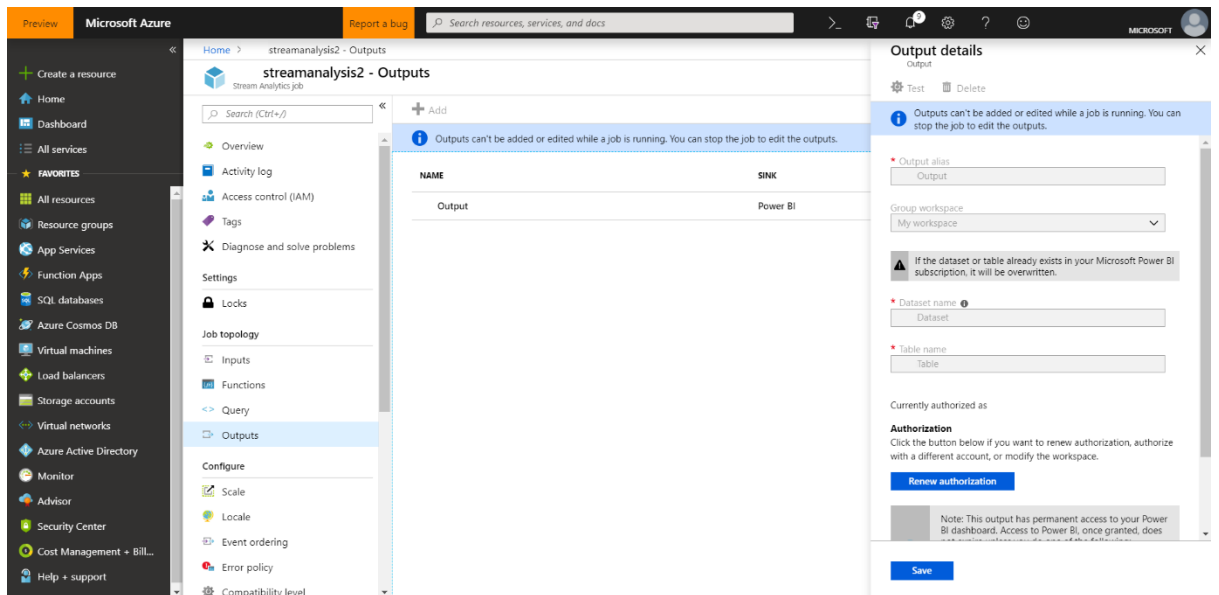
Output alias: The unique alias for the output.

Group Workspace: Select your target group workspace.

Dataset Name: Enter a dataset name.

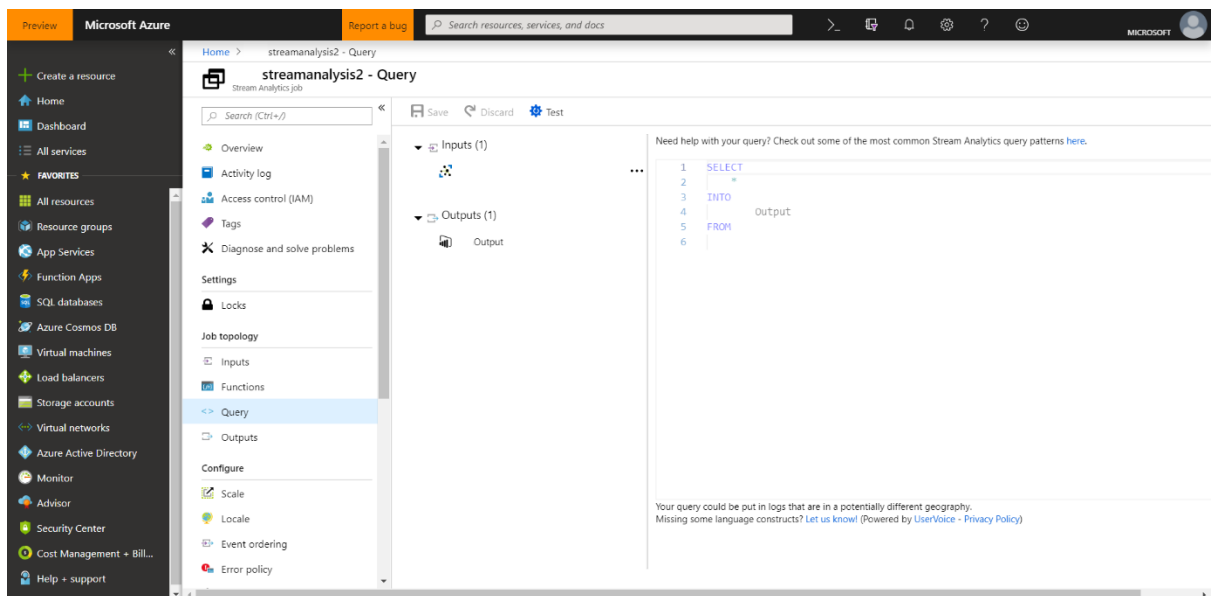
Table Name: Enter a table name.

3. Click **Authorize**, and then sign into your Power BI account.
4. Click **Create**.



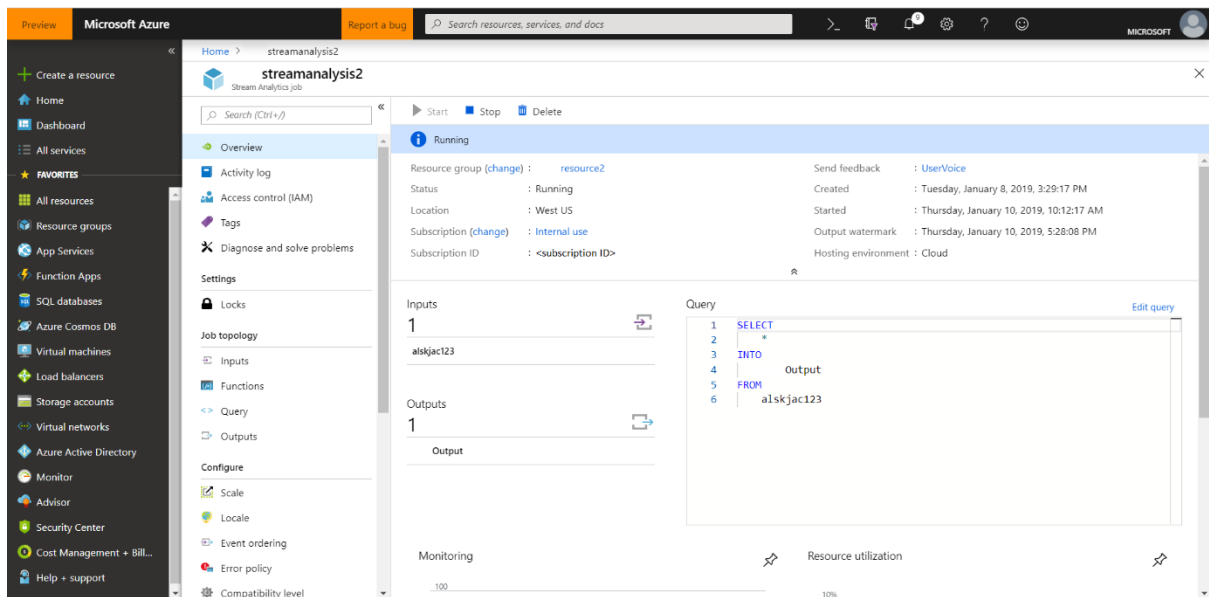
Configure the query of the Stream Analytics job

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**.

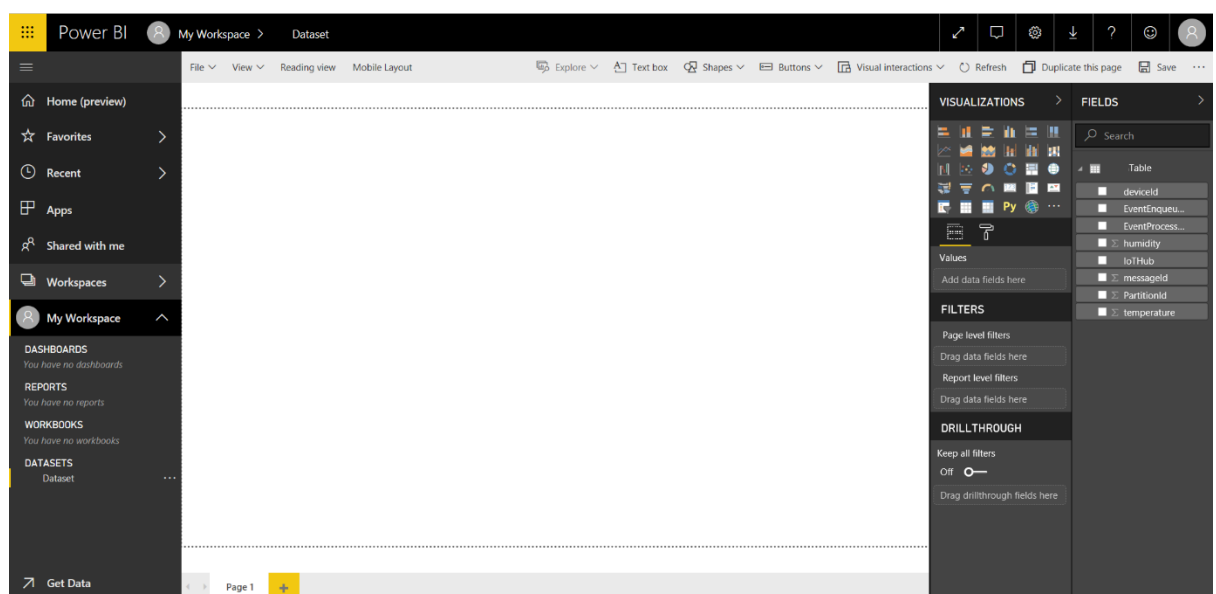


Create and publish a Power BI report to visualize the data

1. Ensure the sample application is running on your device.
2. Sign in to your [Power BI](#) account.
3. Click the workspace you used, **My Workspace**.
4. Click **Datasets**.

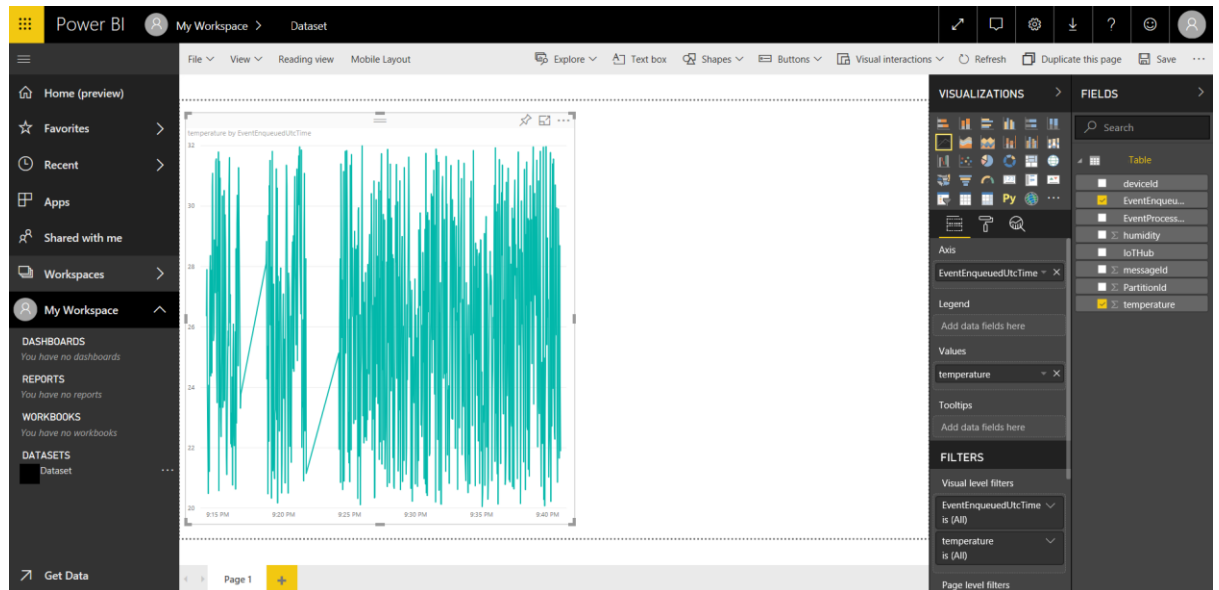
You should see the dataset that you specified when you created the output for the Stream Analytics job.

5. For the dataset you created, click **Add Report** (the first icon to the right of the dataset name).

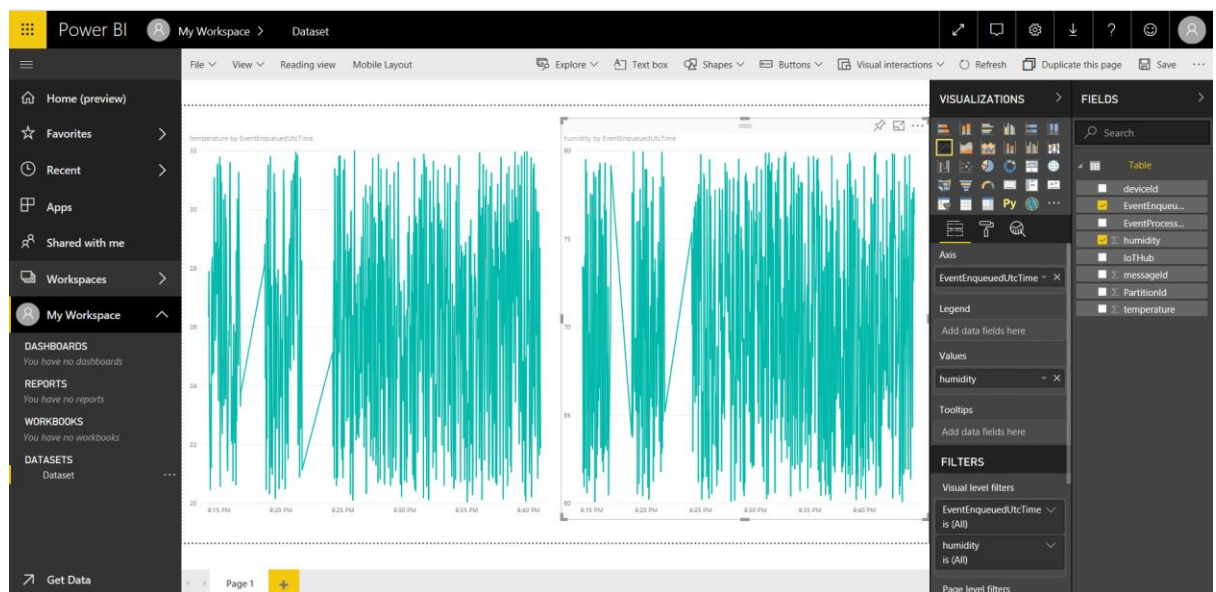


6. Create a line chart to show real-time temperature over time.
 - a. On the report creation page, add a line chart.
 - b. On the **Fields** pane, expand the table that you specified when you created the output for the Stream Analytics job.
 - c. Drag **EventEnqueuedUtcTime** to **Axis** on the **Visualizations** pane.
 - d. Drag **temperature** to **Values**.

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

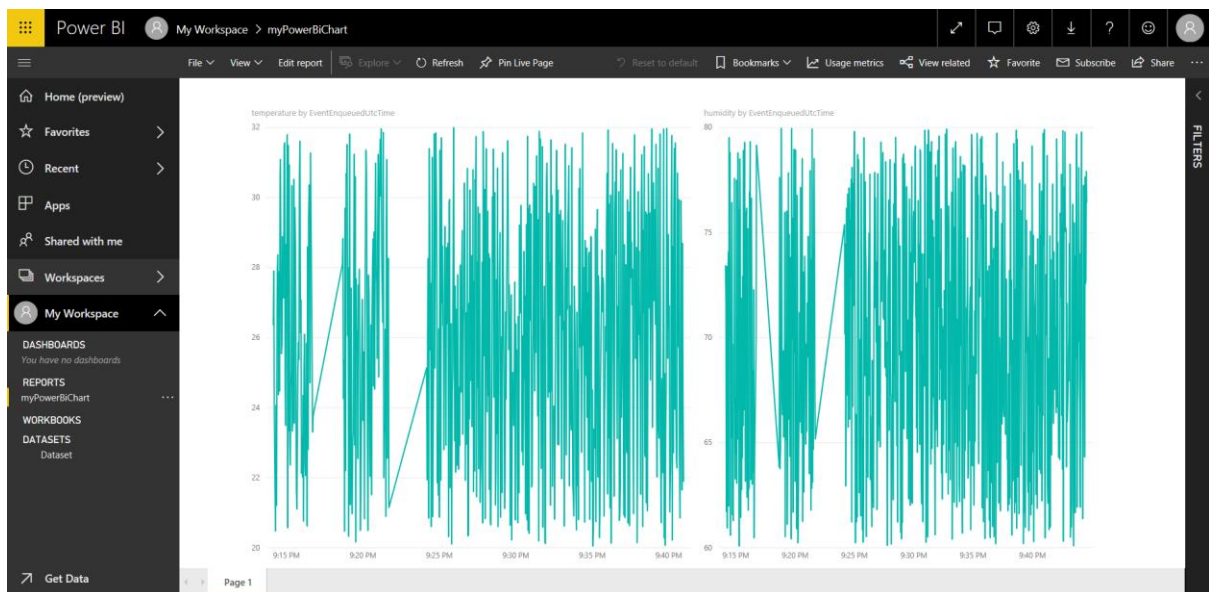


7. Create another line chart to show real-time humidity over time. To do this, follow the same steps above and place **EventEnqueuedUtcTime** on the x-axis and **humidity** on the y-axis.



8. Click **Save** to save the report.
9. Click **Reports** on the left pane, and then click the report that you just created.
10. Click **File > Publish to web**.
11. 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.