

Azure IoT Hands-on workshop

Azure IoT

Agenda

- 09:30-10:00 Dev environment setup, Azure Resource creation (IoT Hub, DPS, Cosmos DB, ASA, Storage, etc)
- 10:00-10:30 Set-up Raspberry Pi
- 10:40-11:00 Run D2C message application on Pi
- 11:00-11:50 Provision a device using Azure IoT DPS (X.509 Individual Enrollment)
- 13:00-13:50 D2C message, Azure Stream Analytics, Data to Storage/DB
- 14:00-17:50 Custom Vision Edge module deployment

1. Create Azure resources

- Create an IoT Hub
- Create Azure Stream Analytics
- Create Blob Storage
- Create Cosmos DB

2. Set-up Raspberry Pi

Install the Raspbian operating system for Pi

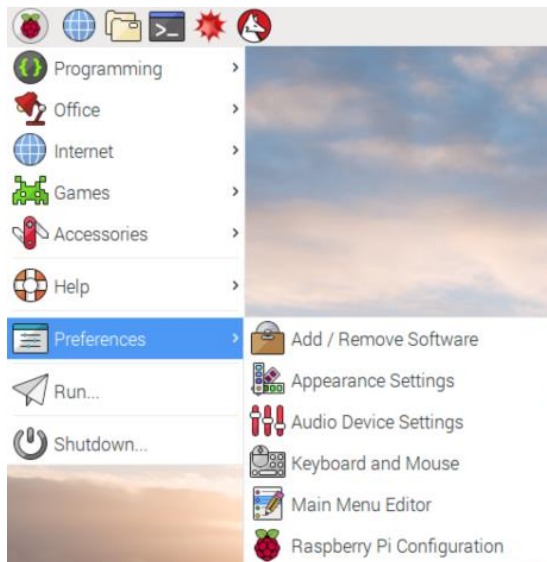
1. Download Raspbian.
 - a. [Download Raspbian Stretch](#) (the .zip file).
 - b. Extract the Raspbian image to a folder on your computer.
2. Install Raspbian to the microSD card.
 - a. [Download and install the Etcher SD card burner utility](#).
 - b. Run Etcher and select the Raspbian image that you extracted in step 1.
 - c. Select the microSD card drive. Etcher may have already selected the correct drive.
 - d. Click Flash to install Raspbian to the microSD card.
 - e. 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.
 - f. Insert the microSD card into Pi.

Enable SSH and I2C

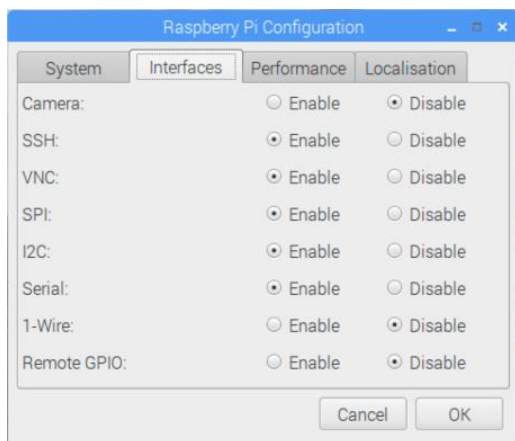
1. Connect Pi to the monitor, keyboard, and mouse.

2. Start Pi and then sign into Raspbian by using pi as the user name and raspberry as the password.

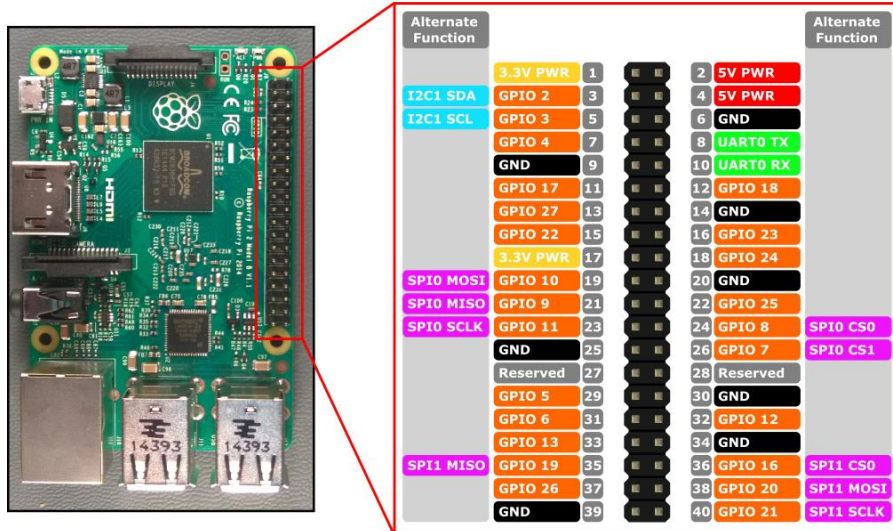
3. Click the Raspberry icon > **Preferences** > **Raspberry Pi Configuration**.



4. 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.



Connect the sensor to Pi



(+) 방향 : 3.3V PWR

중간 : GPIO 4

(-) 방향 : GND

Connect Pi to the network

Note IP address of your Pi.

3. Run D2C message application on Pi

Clone sample application and install the prerequisite packages

1. Connect to your Raspberry Pi with one of the following SSH clients from your host computer:

- [Download and install PuTTY for Windows.](#)

- b. Copy the IP address of your Pi into the Host name (or IP address) section and select SSH as the connection type.
2. Install Node.js and NPM to your Pi.

Install the latest Node.js

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

Clone the sample application

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

Install all packages for the sample

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

Configure the sample application

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

```
nano config.json (or using vi)
```

2. Copy and paste below config

```
{  
  "simulatedData": false,  
  "interval": 2000,  
  "deviceId": "raspberrypi",  
  "LEDPin": 5,  
  "messageMax": 256,  
  "credentialPath": "~/iot-hub",  
  "temperatureAlert": 30,  
  "i2cOption": {  
    "pin": 2,  
    "i2cBusNo": 1,  
    "i2cAddress": 119  
  }  
}
```

3. Save and exit by typing Control-O > Enter > Control-X.
4. Create AM2302Sensor.js

```
/*  
 * IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - Copyright (c) 2017  
 * - Licensed MIT  
 */  
'use strict';  
  
const rpiDhtSensor = require('rpi-dht-sensor');
```

```

function Sensor(/* options */) {
  this.rpidhtsensor = new rpiDhtSensor.DHT22(4);
  // nothing todo
}

Sensor.prototype.init = function (callback) {
  // nothing todo
  callback();
}

Sensor.prototype.read = function (callback) {
  callback(null, {

    temperature: this.rpidhtsensor.read().temperature,
    humidity: this.rpidhtsensor.read().humidity
  });
}

function random(min, max) {
  return Math.random() * (max - min) + min;
}

module.exports = Sensor;

```

5. Modify messageProcessor.js

```

/*
 * IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - Copyright (c) 2017
 * - Licensed MIT
 */
'use strict';

const Bme280Sensor = require('./bme280Sensor.js');
const SimulatedSensor = require('./simulatedSensor.js');
const AM2302Sensor = require('./AM2302Sensor.js');

function MessageProcessor(option) {
  option = Object.assign({
    deviceId: '[Unknown device] node',
    temperatureAlert: 30
  }, option);
  //this.sensor = option.simulatedData ? new SimulatedSensor() : new
Bme280Sensor(option.i2cOption);
  this.sensor = new AM2302Sensor();
  this.deviceId = option.deviceId;
  this.temperatureAlert = option.temperatureAlert
  this.sensor.init(() => {
    this.inited = true;
  });
}

MessageProcessor.prototype.getMessage = function (messageId, cb) {
  if (!this.inited) { return; }

```

```

this.sensor.read((err, data) => {
  if (err) {
    console.log('[Sensor] Read data failed: ' + err.message);
    return;
  }

  cb(JSON.stringify({
    messageId: messageId,
    deviceId: this.deviceId,
    temperature: data.temperature,
    humidity: data.humidity
  }), data.temperature > this.temperatureAlert);
});
}

module.exports = MessageProcessor;

```

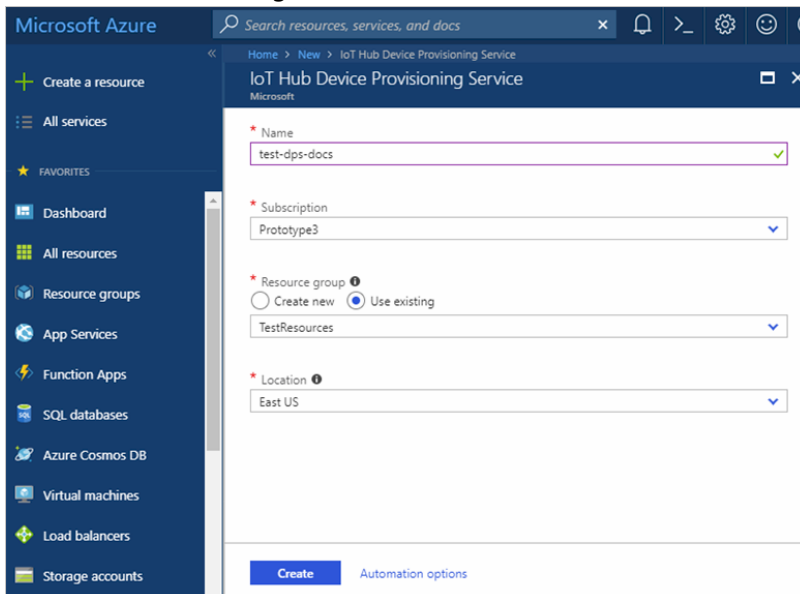
Run the sample application

```
sudo node index.js '<YOUR AZURE IOT HUB DEVICE CONNECTION STRING>'
```

4. Provision a device using Azure IoT DPS

Prepare the environment

1. Create Device Provisioning service

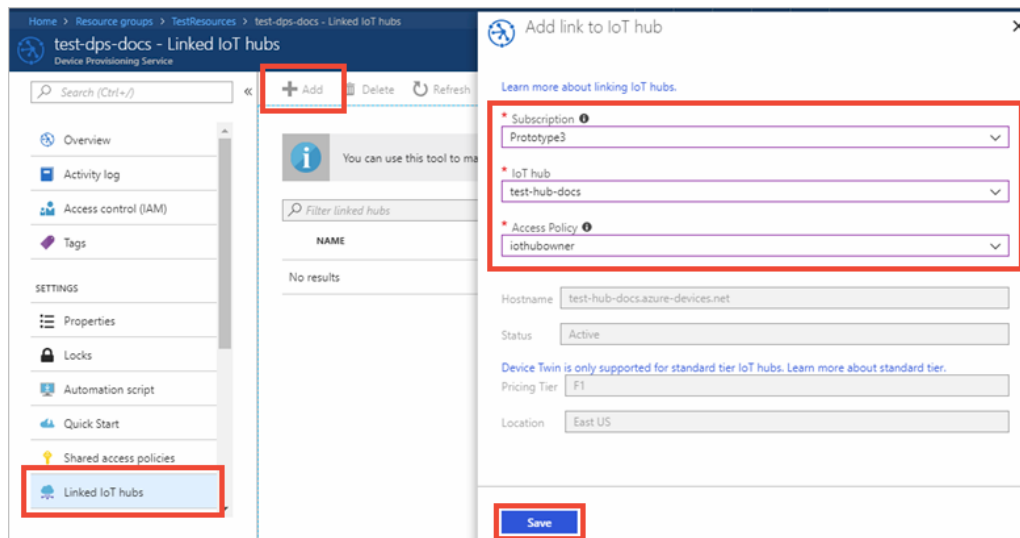


The screenshot shows the Microsoft Azure portal interface for creating a new IoT Hub Device Provisioning Service. The left sidebar contains navigation links for various Azure services. The main panel displays the 'IoT Hub Device Provisioning Service' creation form with the following fields:

- Name:** test-dps-docs (with a green checkmark indicating it is valid)
- Subscription:** Prototype3 (selected from a dropdown menu)
- Resource group:** TestResources (selected from a dropdown menu, with the 'Use existing' radio button selected)
- Location:** East US (selected from a dropdown menu)

At the bottom of the form, there is a blue 'Create' button and a link for 'Automation options'.

2. Link the IoT Hub and DPS



Create a self-signed X.509 device certificate and individual enrollment entry

1. Clone the GitHub repo for the code samples:

```
git clone https://github.com/Azure/azure-iot-sdk-node.git --recursive
```

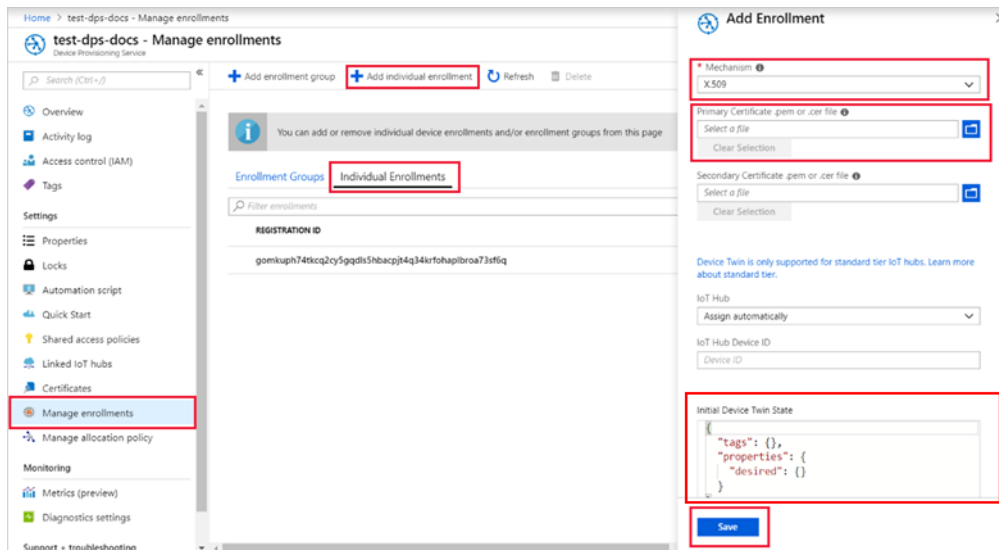
2. Navigate to the certificate generator script and build the project

```
cd azure-iot-sdk-node/provisioning/tools  
npm install
```

3. Create a leaf X.509 certificate by running the script using your own certificate-name. The leaf certificate's common name becomes the Registration ID so be sure to only use lower-case alphanumeric and hyphens.

```
node create_test_cert.js device raspberrypidps  
  
-rw-r--r--  1 pi pi  1678 May 12 13:09 raspberrypidps_key.pem  
-rw-r--r--  1 pi pi  1038 May 12 13:09 raspberrypidps_fullchain.pem  
-rw-r--r--  1 pi pi  1033 May 12 13:09 raspberrypidps_cert.pem
```

4. Add individual enrollment in the Azure Portal

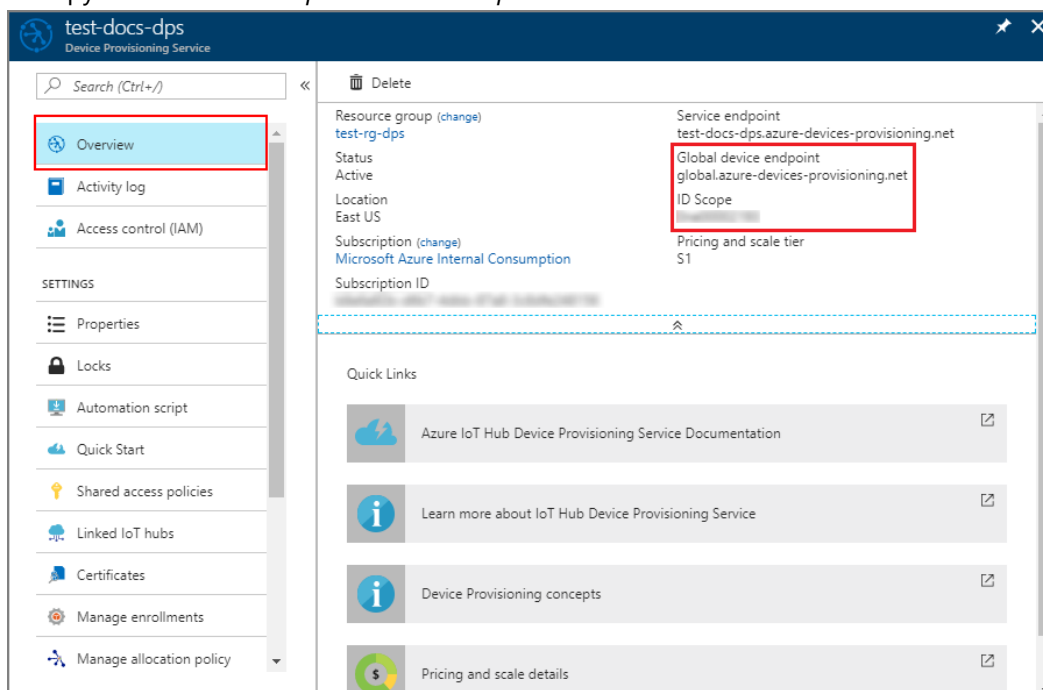


5. Configure "Initial Device Twin State"

```
{
  "tags": {
    "location": "camera-group01"
  },
  "properties": {
    "desired": {
      "mode": "reconnect"
    }
  }
}
```

Run provisioning application on Pi

1. Copy Global device endpoint and ID Scope



2. Copy your certificate and key to the sample folder.

```
cp raspberrypidps_cert.pem ../device/samples/  
cp raspberrypidps_key.pem ../device/samples/
```

3. Navigate to the device sample script and build the project

```
cp ../device/samples/  
npm install
```

4. Edit the `register_x509.js` file

```
var provisioningHost = '[Global Device Endpoint]';  
var idScope = '[Id Scope]';  
var registrationId = '[registrationId]';  
var deviceCert = {  
  cert: fs.readFileSync('raspberrypidps_cert.pem').toString(),  
  key: fs.readFileSync('raspberrypidps_key.pem').toString()  
};
```

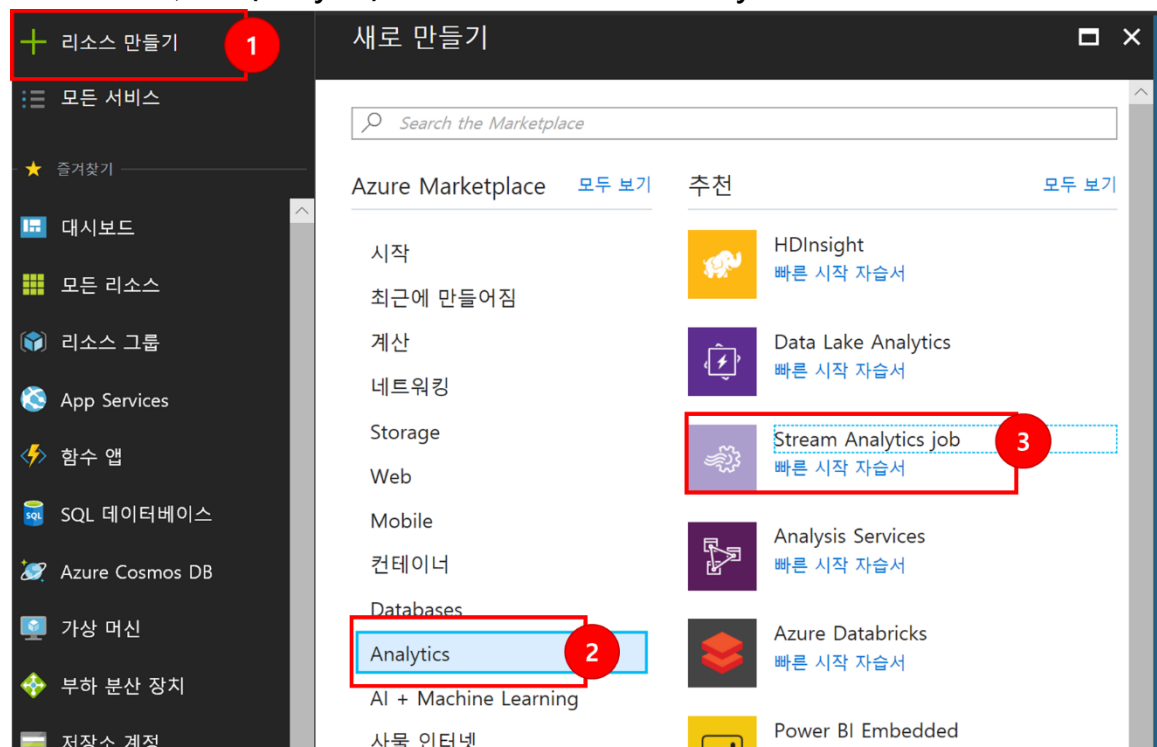
5. Execute the script and verify IoT Hub Device details. Check

```
node register_x509.js
```

5. Process message using Azure Stream Analytics

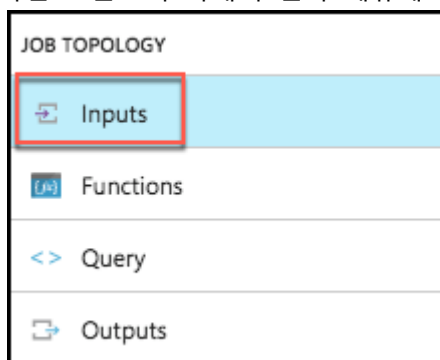
Create Stream Analytics

1. 브라우저에서 **Azure 포털** (<https://portal.azure.com>)을 탐색합니다.
2. **+새로 만들기, 분석(Analytics)**을 선택한 다음 **Stream Analytics Job**을 선택합니다.

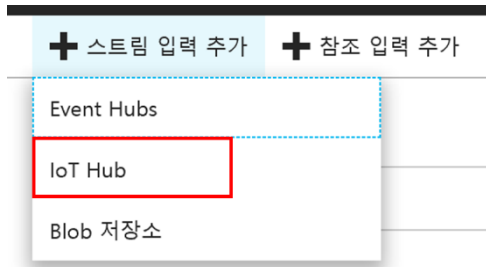


3. 새 Stream Analytics 작업 블레이드에서 다음을 입력합니다.
 - a. 작업 이름: **hot-stream** 을 입력합니다.
 - b. 구독: 지금까지 사용해 온 동일한 구독을 선택합니다.
 - c. 리소스 그룹: **iot-hol** 리소스 그룹을 선택합니다.
 - d. 위치: 다른 리소스와 동일한 위치를 선택합니다.
 - e. 호스팅 환경: **클라우드**를 선택합니다.

- f. 새 Stream Analytics 작업을 프로비저닝하려면 **생성**을 선택합니다.
4. 프로비저닝이 완료되면 포털에서 새 Stream Analytics 작업을 탐색합니다.
5. 작업 토폴로지 아래의 왼쪽 메뉴에서 **입력**을 선택합니다.



6. 입력 블레이드에서 **+스트림 입력 추가**를 선택하여 IoT Hub에 연결된 입력을 추가합니다.



7. 새 입력 블레이드에서 다음을 입력합니다.
- 입력 별칭: 이 값을 **sensorinput** 로 설정합니다.
 - 원본 유형: **데이터 스트림**을 선택합니다.
 - 원본: **IoT Hub**를 선택합니다.
 - 가져오기 옵션: **구독에서 IoT Hub 선택**을 선택합니다.
 - IoT Hub: 기존 IoT Hub 인 **smartmeter-hub**를 선택합니다.
 - 끝점: **메시지**을 선택합니다.
 - 공유 액세스 정책 이름: **서비스**로 설정합니다.
 - 소비자 그룹: **\$Default**로 남겨둡니다.
 - 이벤트 직렬화 형식: **JSON**을 선택합니다.
 - 인코딩: **UTF-8**을 선택합니다.
 - 이벤트 압축 유형: **없음**으로 설정을 둡니다.
 - 생성**을 선택합니다.

IoT Hub
New input

* Input alias

☐ Provide IoT Hub settings manually
☒ Select IoT Hub from your subscriptions

Subscription

IoT Hub ⓘ

Endpoint ⓘ

Shared access policy name ⓘ

Shared access policy key ⓘ

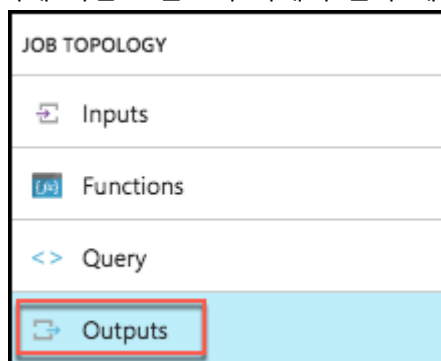
Consumer group ⓘ

* Event serialization format ⓘ

Encoding ⓘ

Event compression type ⓘ

8. 이제 작업 토폴로지 아래의 왼쪽 메뉴에서 **출력**을 선택합니다.



9. 출력 블레이드에서 **+추가**를 선택하여 쿼리에 대한 출력 대상을 Power BI 로 추가합니다.

10. 새 출력 블레이드에서 다음을 입력합니다.

- 출력 별칭: **powerbioutput** 으로 설정합니다
- 데이터 집합 이름: **telemetry** 로 설정합니다.
- 테이블 이름: **telemetry** 로 설정합니다.
- Power BI 계정에 대한 연결에 권한을 부여하려면 **권한 부여**를 선택합니다. 팝업 창에 메시지가 표시되면 핸드온 랩 실습 전, 작업 2 에서 Power BI 계정을 만드는 데 사용한 계정 자격 증명을 입력합니다.

Power BI

New output

* Output alias

powerbioutput



Group workspace

Authorize connection to load workspaces



* Dataset name ⓘ

telemetry



* Table name

telemetry



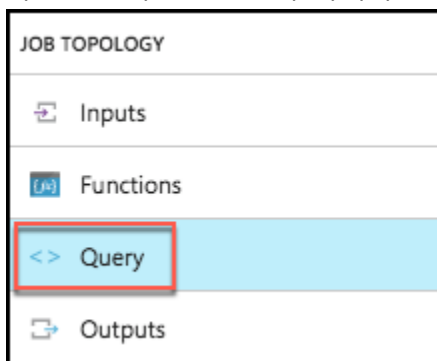
Authorize connection

You'll need to authorize with Power BI to configure your output settings.

Authorize

- 저장을 클릭합니다.

11. 다음으로 작업 토폴로지 아래의 왼쪽 메뉴에서 **쿼리**를 선택합니다.



12. 쿼리 텍스트 상자에서 다음 쿼리를 붙여넣습니다.

```
--평균 온도, 습도 PowerBI Presentation
SELECT AVG(temperature) AS temp, AVG(humidity) AS humidity, deviceId,
EventEnqueuedUtcTime
INTO powerbioutput
FROM sensorinput
GROUP BY TumblingWindow(minute, 5), deviceId, EventEnqueuedUtcTime
```

```
--전체 데이터 CosmosDB 저장
SELECT *
INTO cosmosdboutput
FROM sensorinput
WHERE temperature > 25
```

13. **저장**을 선택하고 확인 메시지가 표시되면 **예**를 선택합니다.



14. Stream Analytics 작업의 개요 블레이드로 돌아가서 **시작**을 선택합니다.



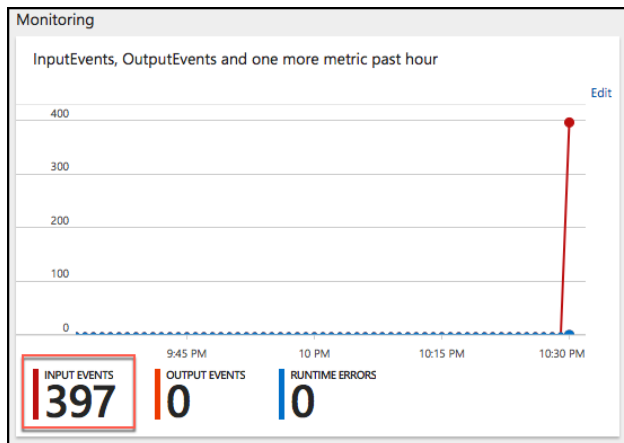
15. 작업 시작 블레이드에서 **지금**을 선택합니다(현재 시점부터 작업이 진행되면서 메시지 처리를 시작).



16. **시작**을 선택합니다.

17. 몇 분 동안 Stream Analytics 작업을 시작합니다.

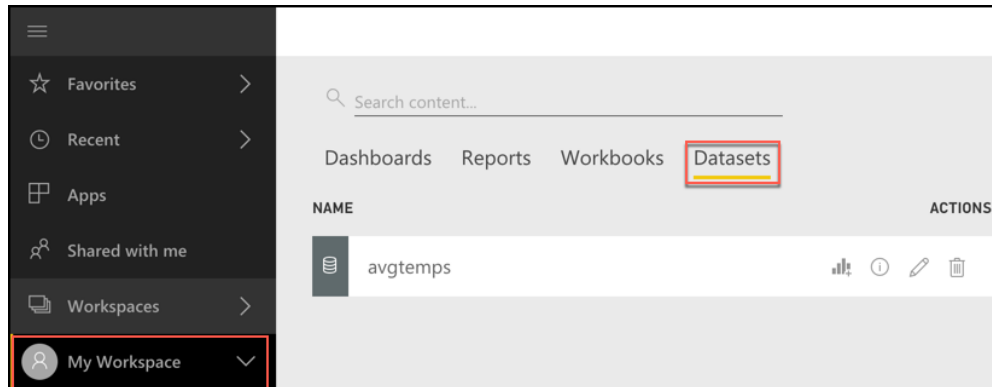
18. Stream Analytics 작업이 성공적으로 시작되면 개요 블레이드의 **모니터링** 차트에 **입력 이벤트**가 하나라도 나타나는지 확인합니다. 라즈베리파이 시뮬레이터에서 이벤트를 확인하는 동안 잠시 "Run"을 눌러서 데이터가 시뮬레이션 되도록 해야 할 수도 있습니다.



Power BI 를 통한 핫 데이터 시각화

1. Power BI 구독(<https://app.powerbi.com>)에 로그인하여 데이터가 수집 중인지 확인합니다.

2. 왼쪽 메뉴에서 내 작업 영역을 선택한 다음, 데이터 집합 탭을 선택합니다.

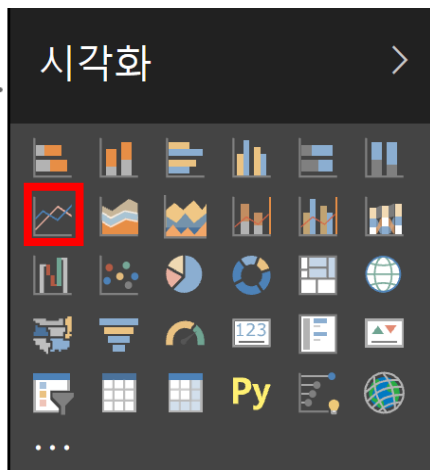


3. 데이터 집합 목록에서 telemetry 데이터 집합을 선택합니다. 데이터 집합 목록에 너무 많은 항목들이 있는 경우에는 telemetry 데이터 집합을 검색합니다. 작업 열 아래의 보고서 생성

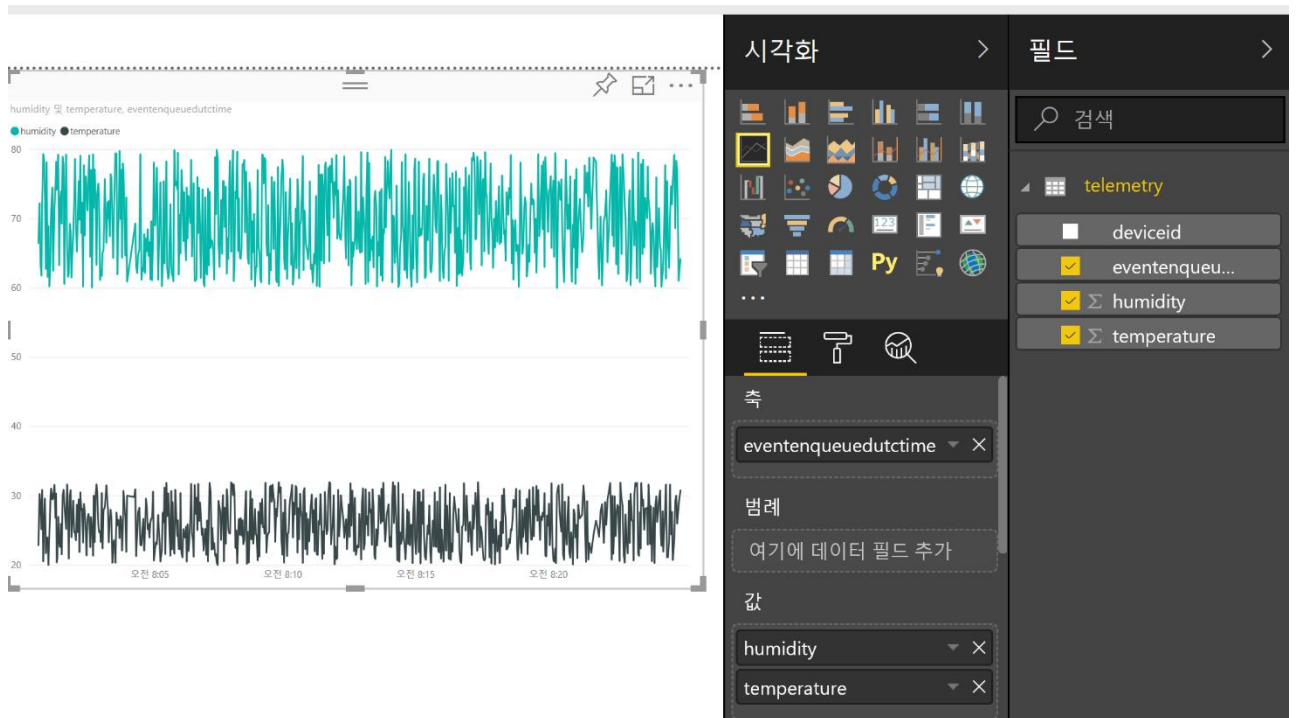
Dashboards Reports Workbooks Datasets Showing 1 item					
NAME ↑	ACTIONS	REFRESHED	NEXT REFRESH	API ACCESS	
telemetry	ⓘ ✎ 🗑️ ⋮	2/20/2019, 5:05:07 PM	N/A	Hybrid	

버튼을 선택합니다.

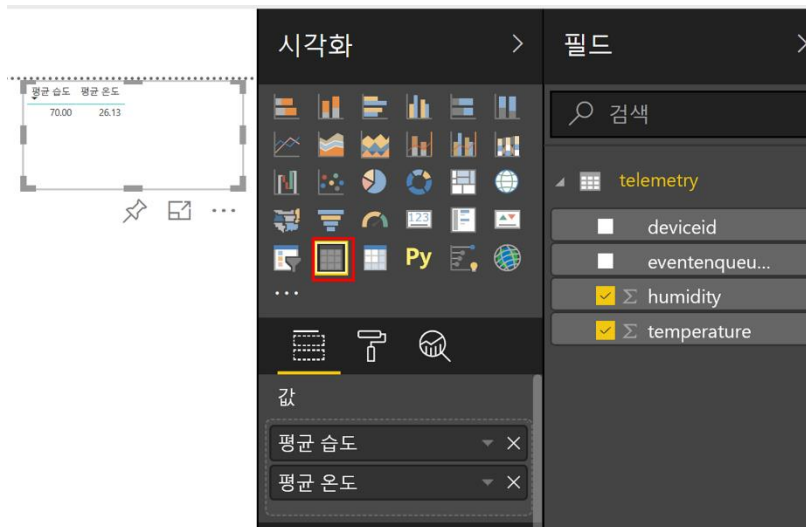
4. 차트 시각화를 위해서는 시각화 팔레트에서 꺾은선형차트를 선택합니다.



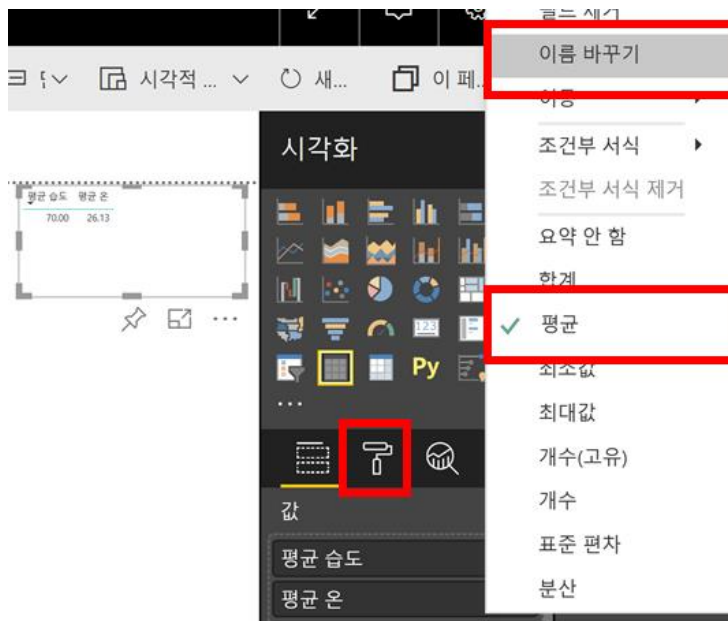
5. 필드 목록에서 **eventenqueuedutctime** 필드를 선택하고 이를 축 항목으로 끌어다 놓습니다. **humidity**와 **temperature** 필드는 값 항목에 끌어 놓습니다.



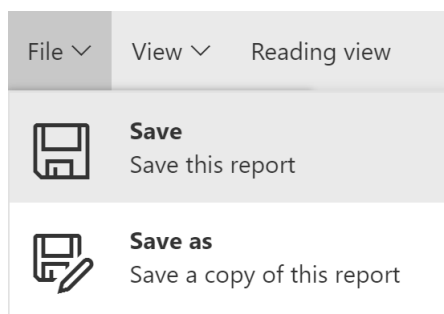
6. 시각화 메뉴에서 **테이블**을 선택하고 **humidity**와 **temperature** 필드를 값에 넣습니다.



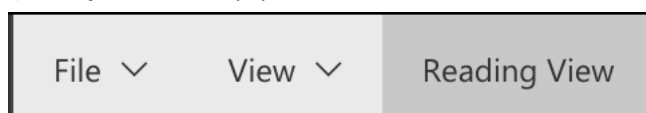
7. 값의 옵션 중 평균을 선택합니다. 또한, 이름 바꾸기를 통하여 이름을 변경할 수 있습니다. 서식 메뉴를 통하여 제목의 폰트나 다른 서식 요소들을 변경할 수 있습니다.



8. 보고서를 저장합니다.



9. 읽기용 보기로 전환합니다.



10. 보고서 내의 열 중 하나를 클릭하여 해당 장치에 대한 데이터를 확인합니다.

평균 습도 평균 온도
70.00 26.13

humidity 및 temperature, eventenqueuedulctime

● humidity ● temperature

