

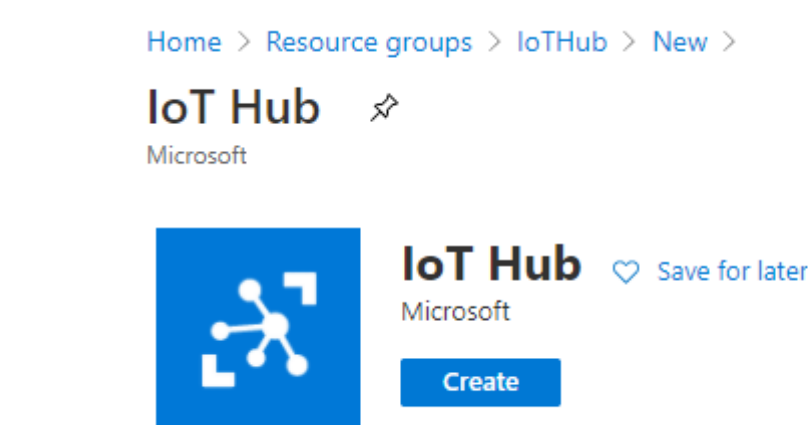
Labo 06 - IoT Hub

Doelstelling

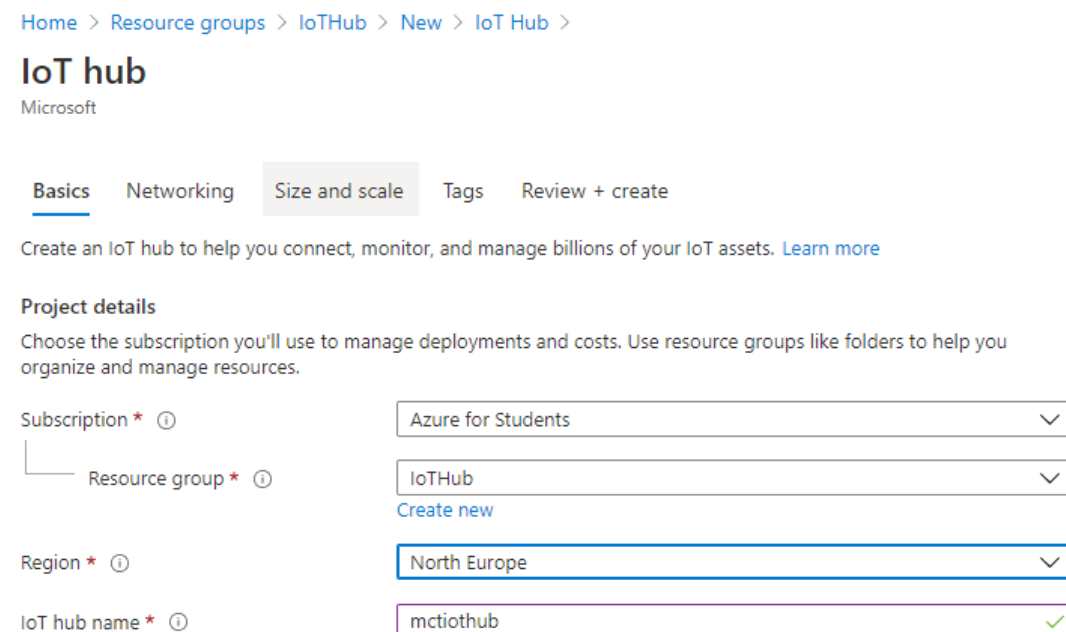
Het doel van deze opgave is leren werken met Azure IoT Hub. We leren communicatie opzetten tussen IoT hub en Python maar ook tussen Azure Functions en IoT Hub.

Aanmaken IoT Hub

Voor we gebruik kunnen maken van de IoT Hub moeten we deze eerst aanmaken. U moet inloggen in de Azure Portal en via de normale weg kan u een IoT hub toevoegen (zoeken op IoT hub).



Daarna vult u de reeds gekende parameters in met als **nieuwe parameter de naam van de IoT hub**, deze moet **uniek** zijn.



Bij Size and scale plaatse we de tier op Free Tier

IoT hub

Microsoft

Basics Networking **Size and scale** Tags Review + create

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](#)

Scale tier and units

Pricing and scale tier * ⓘ

F1: Free tier ▼

[Learn how to choose the right IoT hub tier for your solution](#)

Number of F1 IoT hub units ⓘ

1

Determines how your IoT hub can scale. You can change this later if your needs increase.

Azure Security Center

☐ Off

Turn on Azure Security Center for IoT and add an extra layer of threat protection to IoT Hub, IoT Edge, and your devices. [Learn more](#)

Pricing and scale tier ⓘ	F1	Device-to-cloud-messages ⓘ	Enabled
Messages per day ⓘ	8,000	Message routing ⓘ	Enabled
Cost per month	0.00 EUR	Cloud-to-device commands ⓘ	Enabled
Azure Security Center ⓘ	Feature not enabled	IoT Edge ⓘ	Enabled
		Device management ⓘ	Enabled

▼ Advanced settings

Review + create

< Previous: Networking

Next: Tags >

[Automation options](#)

Daarna volg je verder de wizard en maak je alles aan.

Device Registreren

Als alles is aangemaakt openen we het **IoT Hub startscherm**.

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events

Settings

Shared access policies

Identity

Pricing and scale

Networking

Certificates

Built-in endpoints

Failover

Properties

Locks

Explorers

Query explorer

IoT devices

⚠ Azure IoT Hub and the Azure Device Provisioning Service are updating their TLS certificates starting October 5, 2020 with a new Microsoft Certificate Authority (CA) chained under the certificates, you may need to take action to ensure your devices can continue to connect. [Learn more](#)

Essentials

Resource group (change) : IoTHub

Status : Active

Current location : North Europe

Subscription (change) : Azure for Students

Subscription ID : 7114ddee-9bab-42f4-989f-3aa24bfd5ab9

Tags (change) : [Click here to add tags](#)

Hostname : mctiothub.azure-devices.net

Pricing and scale tier : F1 - Free

Number of IoT Hub units : 1

Device streams (preview) : <https://db-002.northeurope->
[Device streams documentat](#)



Need a way to provision millions of devices?

IoT Hub Device Provisioning Service enables zero-touch, just-in-time provisioning to the right IoT hub without requiring human intervention.



Need a way to monitor and secure your IoT solution?

Azure Security for IoT (ASC for IoT) is a unified security management service. It provides end-to-end threat analysis and protection across hybrid cloud workloads and your Azure IoT solution.



Want to learn more about IoT Hub?

Check out IoT Hub documentation. Learn how to use IoT Hub to connect, monitor, and control billions of Internet of Things assets.






We'd love your feedback!


Your valuable feedback will help us to better understand your requirements in order to improve IoT Hub.


Links in het menu kiezen we voor **IoT devices**. In dit scherm registreren we de toestellen die moeten communiceren met de IoT Hub. We kiezen bovenaan **+New**. In het volgende scherm moeten we een **unieke** naam opgeven. De rest van de opties laten we staan. Daarna kiezen we **Save** en het toestel zal worden aangemaakt.

Create a device


 Find Certified for Azure IoT devices in the Device Catalog 


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 

No parent device
[Set a parent device](#)

Save

Eens het toestel is aangemaakt verschijnt het in de lijst.

 mctiothub | IoT devices 

IoT Hub

< + New Refresh Delete

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events

Settings

Shared access policies

Identity

Pricing and scale

Networking

View, create, delete, and update devices in your IoT Hub.

+ ×

Field

Operator

Value

select or enter a property name

⌵

=

⌵

specify constraint value

+ Add a new clause

Query devices

Switch to query editor

DEVICE ID	STATUS	LAST STATUS UPDATE (UTC)	AUTHENTICATION TYPE	CLOUD TO DEVICE MESSAGE COUNT
pcdieter	Enabled	--	Sas	0

Klikken we op de naam van een toestel dan zien we de **connectiestring van dat toestel**.

Versturen van berichten van het device naar de Cloud

Opmerking: Gelieve Python 3.7 of hoger te gebruiken !!!

Het versturen van berichten naar IoT Hub vanuit Python is niet zo moeilijk. We moeten gewoon de juiste package installeren op onze toestel (PC of Pi). De package die we via **pip** moeten installeren is **pip install azure-iot-device**. U maakt best terug gebruik van een **Python Virtual Environment**. Hoe u dat doet kan u zien in een apart document op Canvas. Daarna voegt u een **Python** file toe aan uw virtual environment. In deze file plaats je volgende code. Dit is een voorbeeld uit de **Azure IoT Hub Device client SDK**.

```
import os
import json
import time
import random
import asyncio
from azure.iot.device import IoTHubDeviceClient

def main():
    # Fetch the connection string from an environment variable
    conn_str = ""

    # Create instance of the device client using the authentication provider
    device_client = IoTHubDeviceClient.create_from_connection_string(conn_str)

    # Connect the device client.
    device_client.connect()

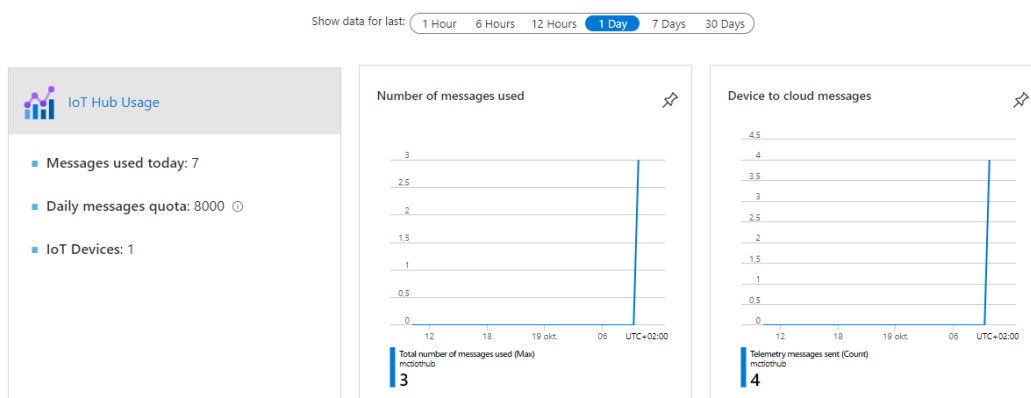
    # Send a single message

    print("Sending message...")
    device_client.send_message("This is a message that is being sent")
    print("Message successfully sent!")

    # finally, disconnect
    device_client.disconnect()

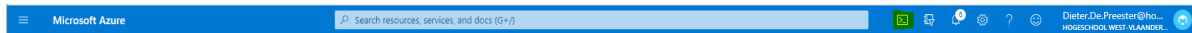
if __name__ == "__main__":
    main()
```

Bij de variable **conn_str** moeten we de connectiestring opgeven van het toestel. Zie bij vorige stuk hierboven over **Device Registreren** waar je deze kan vinden. Daarna alles opslaan en testen door het Python script te runnen. Als je nu gaat kijken in het **IoT Hub** overzicht scherm zou er beweging in de grafieken moeten zitten.

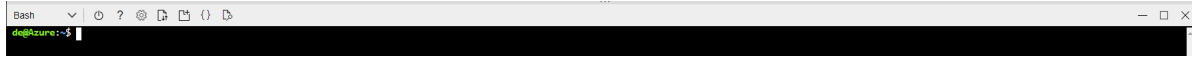


Monitoring & Debugging in de Cloud

Soms is het handig om te weten of een toestel nog berichten aan het versturen is en vooral welke berichten. We kunnen dit debuggen via **Cloud Monitoring**. Hiervoor moeten we de **Bash Shell** op azure activeren. Kies bovenaan in de portal voor de shell (zie screenshot hieronder geel icoon).



Onderaan zal de **Bash shell** verschijnen:



Tik nu het volgende commando om de IoT hub extension te installeren op de command line

```
az extension add -n azure-iot
```

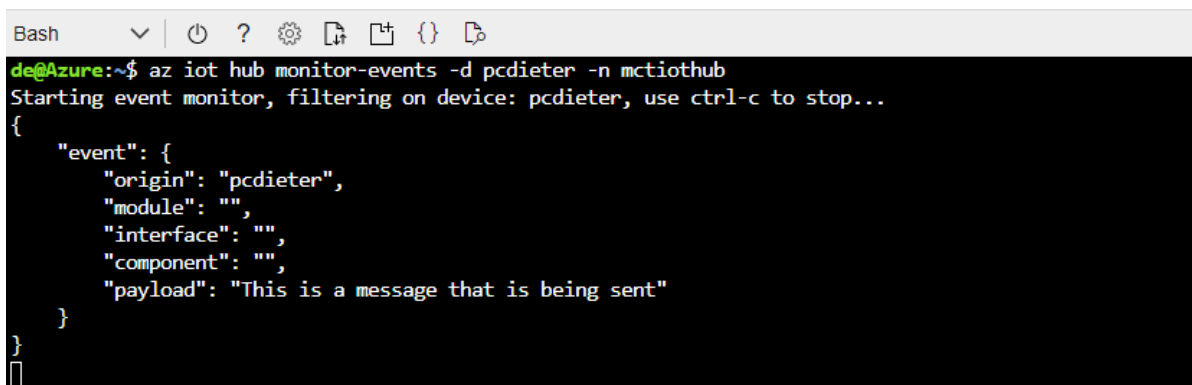
Daarna drukken we op enter. Nu kunnen we de monitor activeren via

```
az iot hub monitor-events -d <device naam> -n <uw iothub naam>
```

Bij mij is dit

```
az iot hub monitor-events -d pcdieter -n mctiothub
```

Wanneer je dit start en je stuurt dan een bericht door je **Python** script te starten dan zie je de berichten binnenkomen in de monitor



Indien je problemen zou hebben met een niet gevonden subscription dan kan je deze instellen. Kijk hiervoor op volgende url:

[Manage Azure subscriptions with the Azure CLI | Microsoft Docs](#)

Versturen van berichten uit de Cloud naar Device

Opmerking voor mensen met een Mac

Bij een Mac moeten er certificaten geïnstalleerd worden, meer info op deze site [Error while sending device to cloud Messages · Issue #332 · Azure/azure-iot-sdk-python · GitHub](#)

Dit is het commando (wijzig naar uw Python versie)

```
open /Applications/Python\ 3.7/Install\ Certificates.command
```

In dit onderdeel gaan we berichten **vanuit de Cloud sturen naar ons toestel**. We noemen dit C2D. Onderstaande code is een voorbeeld uit de Azure IoT Hub Client SDK voor het ontvangen van berichten vanuit de Cloud. Vergeet terug niet de connectionstring toe te voegen op de lijn **conn_str = ""**. We hebben een **main** function. Deze zal via de connectiestring verbinden met de IoT Hub.

```
import os
import asyncio
from six.moves import input
import threading
from azure.iot.device import IoTHubDeviceClient

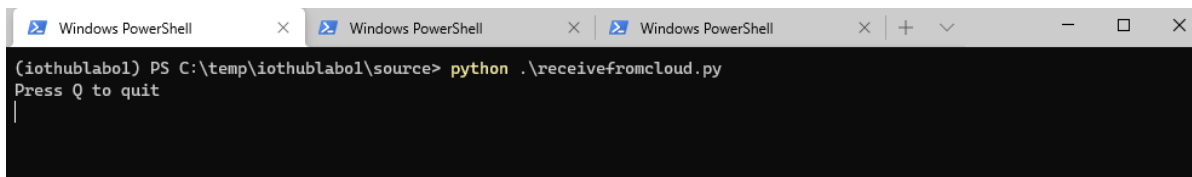
conn_str = ""
device_client = IoTHubDeviceClient.create_from_connection_string(conn_str)

def message_received_handler(message):
    print("the data in the message received was ")
    print(message.data)
    print("custom properties are")
    print(message.custom_properties)
    print("content Type: {0}".format(message.content_type))
    print("")

def run():
    while True:
        selection = input("Press Q to quit\n")
        if selection == "Q" or selection == "q":
            print("Quitting...")
            device_client.disconnect()
            break

if __name__ == "__main__":
    device_client.connect()
    print("Connected")
    device_client.on_message_received = message_received_handler
    run()
```

Als we dit nu uitvoeren krijgen we onderstaande:

A screenshot of a Windows PowerShell terminal window. The title bar shows three open windows, all named 'Windows PowerShell'. The terminal content shows the command 'python .\receivefromcloud.py' being executed in a directory 'C:\temp\iothublab01\source'. The output of the script is 'Press Q to quit', followed by a vertical cursor line indicating the prompt is ready for input.

Nu moeten we een bericht sturen vanuit de **Cloud naar het device**. We gaan naar de **Azure Portal** en gaan naar onze IoT Hub?

IoTHub Resource group

Search (Ctrl+/) << + Add Edit columns Delete resource group Refresh Export to CSV Open query Assign tags Move Delete Export template Fee

Overview

Activity log
Access control (IAM)
Tags
Settings
Quickstart
Deployments
Policies
Properties

Essentials

Subscription (change) : Azure for Students
Subscription ID : 7114dde-9bab-42f4-989f-3aa24bfd5ab9
Tags (change) : Click here to add tags

Filter by name... Type == all Location == all Add filter

Showing 1 to 1 of 1 records. Show hidden types

Name	Type	Location
mctiothub	IoT Hub	North Europe

Deployments : 1 Succeeded

Daarna kiezen we voor **IoT Devices**.

mctiothub IoT Hub

Search (Ctrl+/) << Move Delete

Overview

Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events
Settings
Shared access policies
Identity
Pricing and scale
Networking
Certificates
Built-in endpoints
Failover
Properties
Locks
Explorers
Query explorer
IoT devices

Essentials

Resource group (change) : IoT
Status : Ac
Current location : No
Subscription (change) : Az
Subscription ID : 71
Tags (change) : Cli

Azure IoT Hub and the Azure [continue to connect. Learn mc

Daarna kiezen we ons toestel uit de lijst:

Search (Ctrl+J) < + New Refresh Delete

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Events

Settings

- Shared access policies
- Identity
- Pricing and scale

View, create, delete, and update devices in your IoT Hub.

Field	Operator	Value
+ × select or enter a property name	=	specify constraint value

+ Add a new clause

Query devices


DEVICE ID	STATUS	LAST STATUS UPDATE (UTC)	AUTHENTICATION TYPE
pcdieter	Enabled	--	Sas


Dan kiezen we voor **Message to Device**


pcdieter


mctiothub


Save Message to Device Direct Method Add Module Identity Device Twin M:


Device ID  pcdieter

Primary Key 

Secondary Key 

Primary Connection String 

Secondary Connection String 

Enable connection to IoT Hub  ☒ Enable ☐ Disable

We vullen een bericht in, in JSON en kiezen **Send Message**



Message to device



pcdieter



Send Message



You can use this tool to send messages to a device in your IoT Hub. Message

Device Id ⓘ

pcdieter

Message Body ⓘ

```
{  
  "Command": "On"  
}
```

In ons Python programma zou dit moeten verschijnen.

```
Windows PowerShell x Windows PowerShell x Windows PowerShell >  
(iothublab01) PS C:\temp\iothublab01\source> python .\receivefromcloud.py  
Press Q to quit  
the data in the message received was  
b'{"Command": "On"}'  
custom properties are  
{}  
content Type: None
```

Opvangen van berichten in Azure Functions

In dit stuk gaan we berichten afkomstig van ons Python Script opvangen in Azure Functions. Maak een nieuwe Azure Functions project aan en kies als **Trigger** voor **IoT Hub Trigger**. Bij **Connection string setting name** vullen we **IoT Hub** in. Dit is de naam die we straks ook zullen opgeven in de **local.settings.json** file. Maak deze functie nu aan.

mctiothub | Built-in endpoints

IoT Hub

Search (Ctrl+/) « Save Undo

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events

Settings

Shared access policies

Identity

Pricing and scale

Networking

Certificates

Built-in endpoints

Failover

Properties

Locks

Explorers

Query explorer

IoT devices

Automatic Device Management

IoT Edge

Each IoT hub comes with built-in system endpoints to handle system and device messages. When you create new endpoints and routes, messages stop flowing to the built-in endpoints.

Events

Event Hub Details

Partitions ①

2

Event Hub-compatible name ①

iothub-ehub-mctiothub-5358141-d044840914

Retain for ①

0 1 Days

Consumer Groups ①

Consumer Groups

\$Default

Create new consumer group

Event Hub compatible endpoint

Only policies that allow service connect permissions can be selected. These permissions are applied to the built-in endpoint.

Shared access policy ①

iothubowner

Event Hub-compatible endpoint ①

Endpoint=sb://mctiothub019dednamespace.servicebus.windows.net/SharedAccessKeyName=iothubowner/SharedAccessKey=h...

Start nu eerste de Azure Functions op en start daarna uw Python script om berichten versturen. Kijk of de berichten binnenkomen in uw Azure Functions project. Dit ziet er als volgt uit:

```

messageNumber: 22, Count: 1
[2020-10-19T13:22:26.626] C# IoT Hub trigger function processed a message: This is a message that is being sent
[2020-10-19T13:22:26.628] Executed 'Function1' (Succeeded, Id=366c00e3-bf9b-4499-bbeb-5ba9a08c02fb, Duration=4ms)

```

Opdracht 1

In het Python script gaan we **random** temperaturen genereren. Verstuur deze naar **IoT hub** en vang de berichten op in **Azure Functions**. De opgevangen berichten schrijf je weg naar **Azure Table Storage**. Het Python script moet ook de ganse tijd temperaturen versturen. Je zal dus soort oneindige lus nodig hebben. Voorzie ook wat tijd tussen twee berichten te versturen, kijk hiervoor naar de functie `time.sleep(..)` in Python. Als dit werkt bouwen we nog een extra beveiliging in. We willen enkel temperaturen boven de 30 graden versturen !!.

Vertrek van onderstaande basiscript.

```

import os
import json
import time
import random
from azure.iot.device import IoTHubDeviceClient

conn_str = ""

# Create instance of the device client using the authentication provider
device_client = IoTHubDeviceClient.create_from_connection_string(conn_str)

def main():
    global device_client

    while True:
        #temperatuur genereren
        #versturen
        time.sleep(3) #even wachten
        print("Message successfully sent!")

```

```
# finally, disconnect
device_client.disconnect()

if __name__ == "__main__":
    device_client.connect()
    main()
```

IoT Hub Device Twin

Een **IoT Hub device twin** is een vorm van **configuratie (desired properties)** die we kunnen inlezen op ons toestel. We kunnen hier bv. de snelheid van inlezen in opslaan, of de kleur van een led, enz... Het is dus vooral configuratie informatie. Deze info slaan we op in de Cloud in een **twin document**. Dit document is verbonden met ons toestel in the field. Vanop ons toestel kunnen we dit document inlezen. In het volgende voorbeeld gaan we de drempelwaarde opslaan in zo een twin document en dit inlezen bij het opstarten van ons Python script. Uit dat document zullen we dan ophalen vanaf welke waarde we de temperatuur moeten doorsturen.

Het instellen van de parameters in het **twin document** doen we als volgt. Ga naar de **IoT Hub** en kies uw toestel.

In het onderstaande scherm kiezen we voor **Device Twin**:

Daarna krijgen we het onderstaande scherm:



The device twin for 'pcdieter' is shown below. You can add tags and desired properties to your device twin here. To remove a tag or desired property, set the value of th

```
{
  "deviceId": "pcdieter",
  "etag": "AAAAAAAAAAI=",
  "deviceEtag": "OTI5MDAxOTQ0",
  "status": "enabled",
  "statusUpdateTime": "0001-01-01T00:00:00Z",
  "connectionState": "Disconnected",
  "lastActivityTime": "2020-10-20T06:13:13.0498844Z",
  "cloudToDeviceMessageCount": 0,
  "authenticationType": "sas",
  "x509Thumbprint": {
    "primaryThumbprint": null,
    "secondaryThumbprint": null
  },
  "version": 3,
  "properties": {
    "desired": {
      "speed": 100,
      "$metadata": {
        "$lastUpdated": "2020-10-20T06:10:36.4062523Z",
        "$lastUpdatedVersion": 2,
        "speed": {
          "$lastUpdated": "2020-10-20T06:10:36.4062523Z",
          "$lastUpdatedVersion": 2
        }
      },
      "$version": 2
    },
    "reported": {
```

Voeg onderstaande toe in het json twin document en druk op **Save** bovenaan:

```
    "properties": {
      "desired": {
        "threshold" : 30,
```

Nu moeten we deze waarde inlezen in ons Python script. Wijzig de code waar we temperaturen versturen als volgt: na de connect gaan we het twin document inlezen via **device_client.get_twin()**. We krijgen een **Python Dictionary** terug en kunnen dan op de reeds gekende manier de properties uitlezen.

```
if __name__ == "__main__":
    device_client.connect()
    print("Connected")
    twin = device_client.get_twin()
    threshold = twin["desired"]["threshold"]
```

Pas nu zelf de code aan zodat we enkel berichten versturen voor temperaturen die **boven** de threshold uitkomen.

Naast het inlezen van het **twin document** kunnen we ook een **event handler** gebruiken om veranderingen in het **twin document** op te vangen. Voeg onderstaande code toe **NA** de **connect()**. Deze code bevat een **eventhandler** die actief zal worden wanneer er een update is van het **twin document**.

```
def twin_patch_handler(patch):
    print("the data in the desired properties patch was: {}".format(patch))

# set the twin patch handler on the client
device_client.on_twin_desired_properties_patch_received = twin_patch_handler
```

Wijzig nu zelf uw code zodat we bij een verandering van het **twin** document de nieuwe **threshold** uitlezen en gaat instellen als de nieuwe threshold in de applicatie. **Dit moet gebeuren zonder dat we de applicatie opnieuw starten.**

IoT Hub Direct Methods


Met behulp van **Direct Methods** kunnen we Python methodes activeren op ons toestel. Een voorbeeld hiervan is reboot van een device. We hebben terug een **eventhandler** die we moeten koppelen in onze code. Deze zal **Direct Methods** uitvoeren. Voeg onderstaande code toe **NA** de **connect()**. Deze code zal het event **on_method_request_received** koppelen aan de functie **method_request_handler**. Deze functie vangt de **Direct Methods** op en zal een antwoord terugsturen. Voeg ook bovenaan volgende toe: **from azure.iot.device import MethodResponse**








```
def method_request_handler(method_request):
    # Determine how to respond to the method request based on the method
    name
    if method_request.name == "reboot":
        payload = {"result": True, "data": "some data"} # set response
        payload
        status = 200 # set return status code
        print("executed reboot")
    else:
        payload = {"result": False, "data": "unknown method"} # set
        response payload
        status = 400 # set return status code
        print("executed unknown method: " + method_request.name)






    # Send the response
    method_response =
    MethodResponse.create_from_method_request(method_request, status, payload)
    device_client.send_method_response(method_response)

# set the twin patch handler on the client
device_client.on_method_request_received = method_request_handler
```

Het aanroepen van direct methods doen we via de Azure Portal. Ga naar je **device** in de Azure Portal en kies **Direct Method**.

pcdieter 
mctiothub

 Save  Message to Device  **Direct Method**  Add Module Identity  Device Twin  Manage keys  Refresh

Device ID 	pcdieter
Primary Key 
Secondary Key 
Primary Connection String 
Secondary Connection String 

Daarna vullen we de naam in van de methode vb: **reboot** en eventueel een **payload**.



Direct method

pcdieter



Invoke Method



You can use this tool to send direct methods to a device. Direct methods ha

Device Id ⓘ

pcdieter


Method Name ⓘ

reboot

Payload ⓘ

```
{  
  "sec" : 10  
}
```

Druk nu op **Invoke Method** en de methode zal actief worden op het toestel (ons Python script).
We krijgen ook de response terug.

 Invoke Method



You can use this tool to send direct methods to a dev

Device Id ⓘ

pcdieter

Method Name ⓘ

Payload ⓘ

Connection Timeout ⓘ

Method Timeout ⓘ

Result ⓘ

`("result":true,"data":"some data")`

IoT Hub Reporting Properties

Naast de **desired properties** hebben we ook **reporting properties**. Deze laatste zijn rapporteringen van ons toestel (ons Python script) aan de IoT Hub. Een voorbeeld kan zijn, boot, of battery level, etc.... Dit is zeer eenvoudige te implementeren. Voeg onderstaande code toe **NA** de **connect()**.

```
reported_properties = {"bootstatus": "ok"}  
device_client.patch_twin_reported_properties(reported_properties)
```

Wanneer we nu gaan kijken in de IoT Hub bij **Device** twin dan zien we deze staan.



The device twin for 'pcdieter' is shown below. You can add tags and desired properties to your device

```
{
  "deviceId": "pcdieter",
  "etag": "AAAAAAAAABs=",
  "deviceEtag": "OTI5MDAxOTQ0",
  "status": "enabled",
  "statusUpdateTime": "0001-01-01T00:00:00Z",
  "connectionState": "Connected",
  "lastActivityTime": "2020-10-20T07:01:57.2252365Z",
  "cloudToDeviceMessageCount": 0,
  "authenticationType": "sas",
  "x509Thumbprint": {
    "primaryThumbprint": null,
    "secondaryThumbprint": null
  },
  "version": 29,
  "properties": {
    "desired": {
      "speed": 100,
      "threshold": 500,
      "$metadata": {
        "$lastUpdated": "2020-10-20T07:01:57.1800266Z",
        "$lastUpdatedVersion": 27,
        "speed": {
          "$lastUpdated": "2020-10-20T07:01:57.1800266Z",
          "$lastUpdatedVersion": 27
        },
        "threshold": {
          "$lastUpdated": "2020-10-20T07:01:57.1800266Z",
          "$lastUpdatedVersion": 27
        }
      },
      "$version": 27
    },
    "reported": {
      "bootstatus": "done",

```

Queries op toestellen

Nog een voordeel van IoT hub is dat je via een SQL taal kan filteren op toestellen. Voor één toestel is dit niet zo belangrijk maar heb je duizenden toestellen dan is dit zeer handig. Ga naar het overzicht scherm van je toestellen in de **IoT Hub** en kies daar **Switch to query editor**.

mctiohub | IoT devices

Search (Ctrl+/) << + New Refresh Delete

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events
Settings
Shared access policies
Identity
Pricing and scale
Networking

View, create, delete, and update devices in your IoT Hub.

Field Operator Value
select or enter a property name = specify constraint value
+ Add a new clause
Query devices </> Switch to query editor

DEVICE ID	STATUS	LAST STATUS UPDATE (UTC)	AUTHENTICATION TYPE	CLOUD TO DEVICE MESSAGE COUNT
pcdieter	Enabled	--	Sas	0

Kies nu **Enable query editor** en antwoord **Yes**.

mctiohub | IoT devices

Search (Ctrl+/) << + New Refresh Delete

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events
Settings
Shared access policies
Identity
Pricing and scale
Networking

View, create, delete, and update devices in your IoT Hub.

SELECT * FROM devices WHERE
|
Query devices Enable query editor Switch to query builder

DEVICE ID	STATUS	LAST STATUS UPDATE (UTC)	AUTHENTICATION TYPE	CLOUD TO DEVICE MESSAGE COUNT
pcdieter	Enabled	--	Sas	0

Nu kan je **queries** schrijven vb: toon alle toestellen waar de bootstatus done is. Dit doen we door in de WHERE te schrijven **properties.reported.bootstatus = 'done'**. Je kan dit doen voor alle properties.

mctiohub | IoT devices

Search (Ctrl+/) << + New Refresh Delete

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events
Settings
Shared access policies
Identity
Pricing and scale
Networking

View, create, delete, and update devices in your IoT Hub.

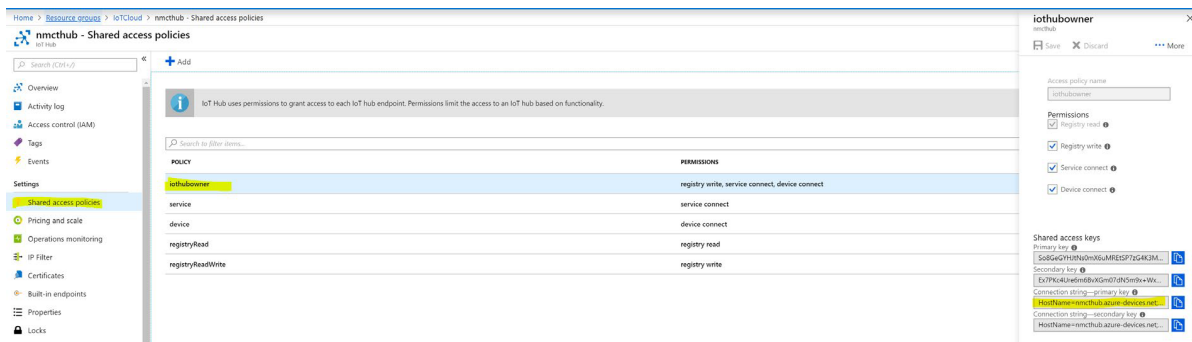
SELECT * FROM devices WHERE
properties.reported.bootstatus = 'done'
Query devices Enable query editor Switch to query builder

DEVICE ID	STATUS	LAST STATUS UPDATE (UTC)	AUTHENTICATION TYPE	CLOUD TO DEVICE MESSAGE COUNT
pcdieter	Enabled	--	Sas	0

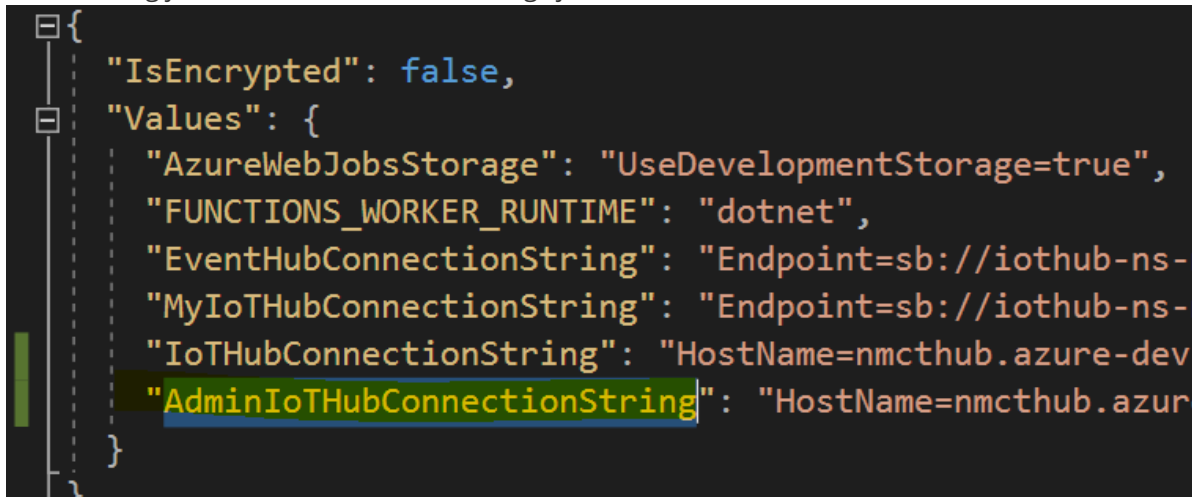
Probeer nu zelf de query te schrijven om te filteren op de **threshold desired property**. Haal de toestellen op waar die **bvb > 100**.

IoT Hub en .NET

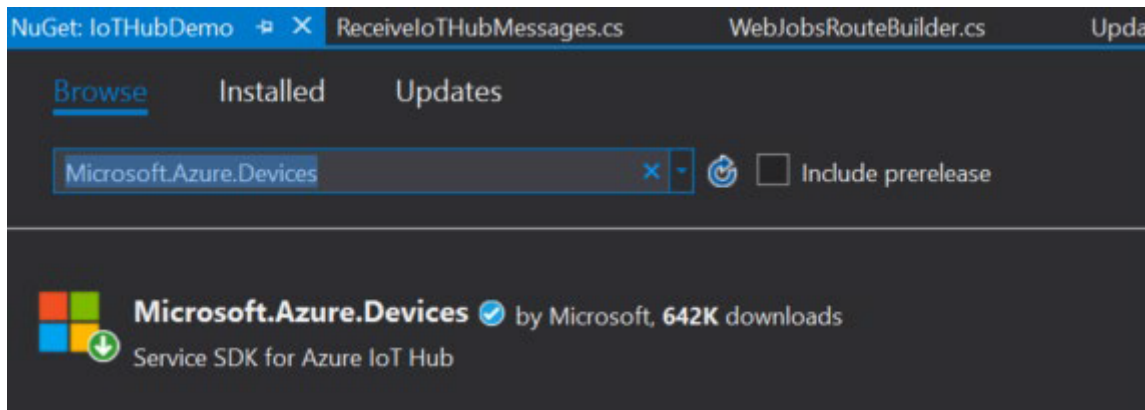
Tot nu toe hebben we het aansturen van de device gedaan vanuit de Azure IoT Hub portal. We kunnen direct methodes starten, properties wijzigen of queries uitvoeren. Toch is het zeer handig om dit ook vanuit C# te kunnen. Ideaal is terug Azure Functions. Voor we starten moeten we een nieuwe connectionstring toevoegen aan ons project. Deze connectiestring heeft meer rechten om devices te beheren. U kan deze vinden op onderstaande locatie:



Daarna voeg je deze toe aan de local.settings.json.



Daarna moeten we volgende Nuget package toevoegen: Microsoft.Azure.Devices



Voeg nu een nieuwe Azure Function toe met een HTTP Trigger. We maken er een **HTTP PUT** met 2 parameters in de URL: het **Deviceid** en de **threshold**. Dit ziet er ongeveer als volgt uit:

```
[FunctionName("UpdateTemperatureThreshold")]
public static async Task<ActionResult> UpdateTemperatureThreshold([HttpTrigger(AuthorizationLevel.Anonymous, "put", Route = "devices/{deviceid}/{threshold}")] HttpRequest req,
    string deviceid,
    int threshold,
    ILogger log)
```

Nu kunnen we via de **RegistryManager** klasse connecteren naar de **IoT Hub**. Daarna vragen we het twin document op via **GetTwinAsync**. Dan kunnen we de property wijzigen met de nieuwe threshold. Als laatste moeten we de update doorsturen via **UpdateTwinAsync**. Wat in de vorige stukken via de portal deden, doen we nu via code.

```
RegistryManager manager = RegistryManager.CreateFromConnectionString(Environment.GetEnvironmentVariable("AdminIoTHubConnectionString"));
var twin = await manager.GetTwinAsync(deviceid);
twin.Properties.Desired["threshold"] = threshold;
await manager.UpdateTwinAsync(deviceid, twin, twin.ETag);

return new OkObjectResult("");
```

Naast het wijzigen van properties is het ook mogelijk om queries uit te voeren via C#. Voeg een nieuwe Azure Function toe GetDevicesFunction(). Dit is terug een HTTP Trigger. De code in de functie ziet er als volgt uit:

Via de RegistryManager maken we terug connectie naar de IoT Hub. We voeren een query uit om alle devices op te vragen. Daarna overlopen we de resultaten en steken we het twin document in een lijst. Deze lijst keren we op het einde terug naar de client.

```
[FunctionName("GetDevices")]
0 references
public static async Task<IActionResult> GetDevices([HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = "devices")] HttpRequest req, ILogger log)
{
    RegistryManager manager = RegistryManager.CreateFromConnectionString(Environment.GetEnvironmentVariable("AdminIoTHubConnectionString"));
    var devices = manager.CreateQuery("SELECT * FROM Devices");
    List<Twin> twins = new List<Twin>();
    while (devices.HasMoreResults)
    {
        var page = await devices.GetNextAsTwinAsync();
        foreach (var twin in page)
        {
            twins.Add(twin);
        }
    }
    return new OkObjectResult(twins);
}
```

De laatste actie die we wensen uit te voeren via .NET en Azure Functions is het activeren van een reboot. We voegen een nieuwe Azure Function toe RebootDevice, terug een HTTP Trigger. Nu gaan we via ServiceClient communiceren met IoT Hub. We maken een CloudToDeviceMethod aan en vullen de payload in. Deze sturen we dan naar de device.

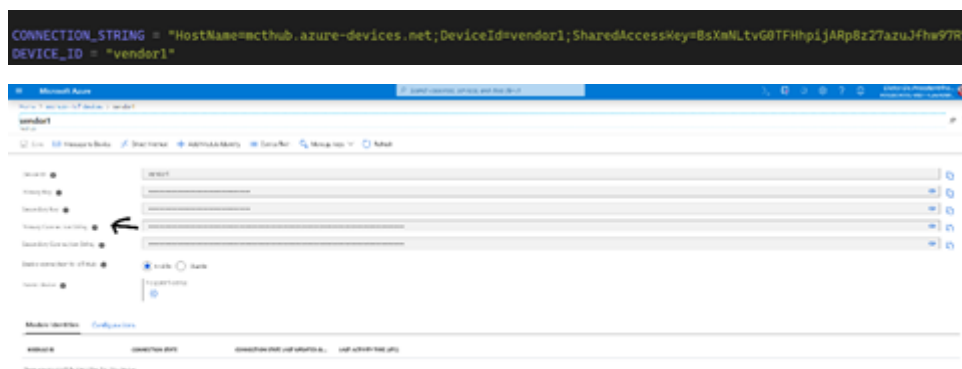
```
[FunctionName("RebootDevice")]
0 references
public static async Task<IActionResult> RebootDevice([HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = "devices/reboot/{deviceid}")] HttpRequest req, string deviceid, ILogger log)
{
    try
    {
        ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(Environment.GetEnvironmentVariable("AdminIoTHubConnectionString"));
        CloudToDeviceMethod method = new CloudToDeviceMethod("reboot");
        method.SetPayloadJson("{\"seconds\" : 15}");
        await serviceClient.InvokeDeviceMethodAsync(deviceid, method);

        return new OkObjectResult("");
    }
    catch (Exception ex)
    {
        throw;
    }
}
```

Overzicht IoT Hub Connectionstrings (ter info)

Bij het werken met IoT Hub zijn er 3 mogelijke connectionstrings weet goed het verschil en wanneer wat te gebruiken:

ConnectionString voor het toestel (=RaspberryPi,Python script)



ConnectionString voor de IoT Hub Trigger (Azure Functions message opvangen)

```
[FunctionName("ItemGetFunctionsTableStorage")]
0 references
public static async Task Run[AzureTableStorage](IoTHubTrigger("messages/events", Connection = "EventHubConnectionString"))EventData message, ILogger log)
{
    try
    {

```



Connectionstring om te praten met de IoT Hub voor management (Twin Update, Reboot,etc)

```
string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
LocationMessage locationMessage = JsonConvert.DeserializeObject<LocationMessage>(requestBody);

RegistryManager manager = RegistryManager.CreateFromConnectionString(Environment.GetEnvironmentVariable("IoTHubConnectionString"));
var twin = await manager.GetTwinAsync(LocationMessage.DeviceId);
```

```
ServiceClient serviceClient;
string connectionString = Environment.GetEnvironmentVariable("IoTHubConnectionString");
serviceClient = ServiceClient.CreateFromConnectionString(connectionString);
```

