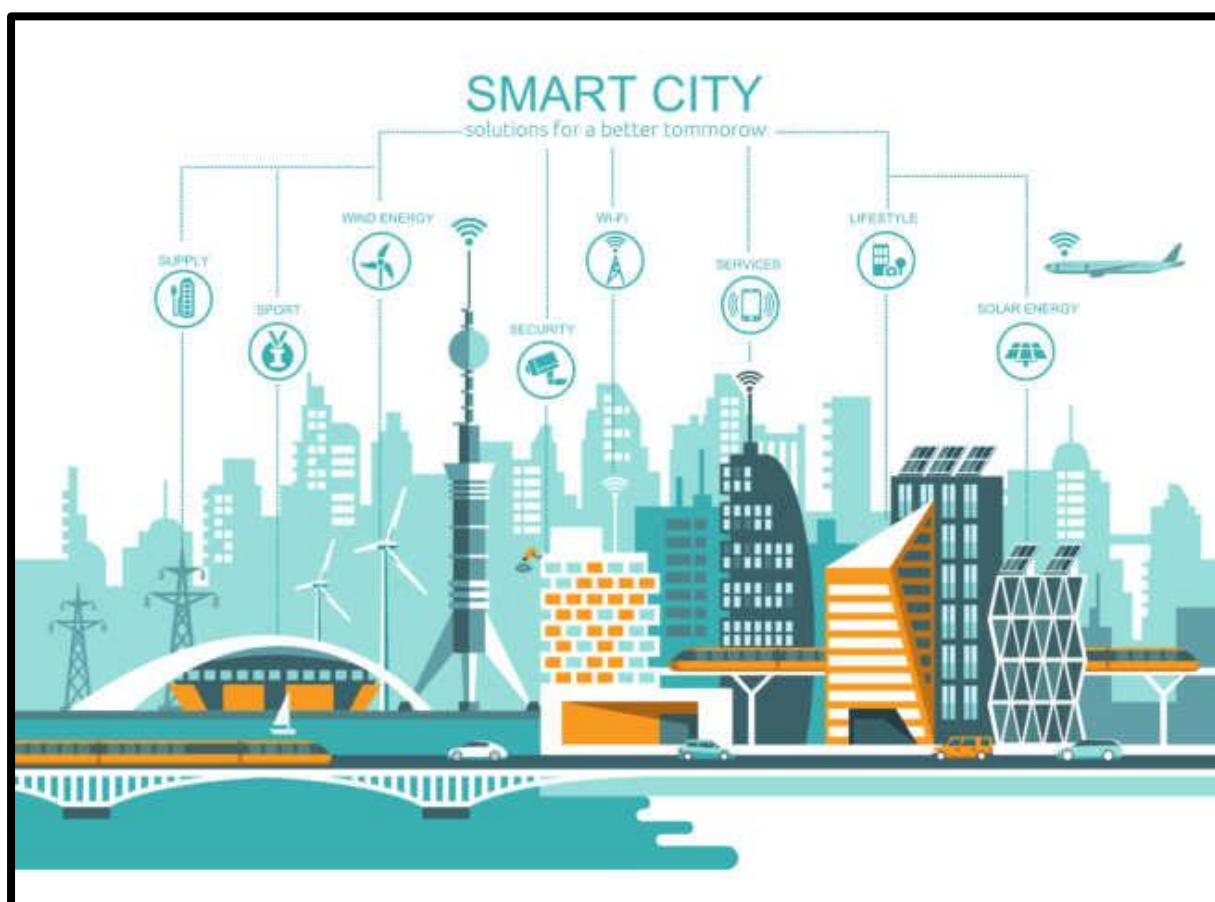


Project 4.0

Smart City Herentals

Installatie handleiding



Connect IT

Jens Lambrechts

Bert Moelans

Thomas Dergent

Yori Verbist

Jannes Manneart

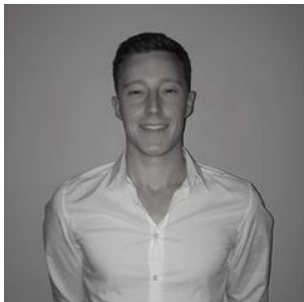
Cedric Meeuwis

Inleiding

Ons team

Als 3de jaar studenten bij de IT Factory van Thomas More hogeschool te Geel kregen we bij het vak 'Project 4.0' de opdracht om van de stad Herentals een smart city te maken.

We zijn een samenstelling van alle richtingen van de IT Factory. We hebben één iemand die de IoT heeft opgesteld, Jannes Mannaert. De verantwoordelijke voor de front-end is Thomas Dergent en voor de backend is Cedric Meeuwis. Daarnaast zorgt Yori Verbist voor AI-integratie door Qlik en zorgen Bert Moelans en Jens Lambrechts voor de hosting van dit project zodat alles mooi aan elkaar hangt.



Bert Moelans



Jannes Mannaert



Yori Verbist



Jens Lambrechts



Thomas Dergent



Cedric Meeuwis

Onze opdracht

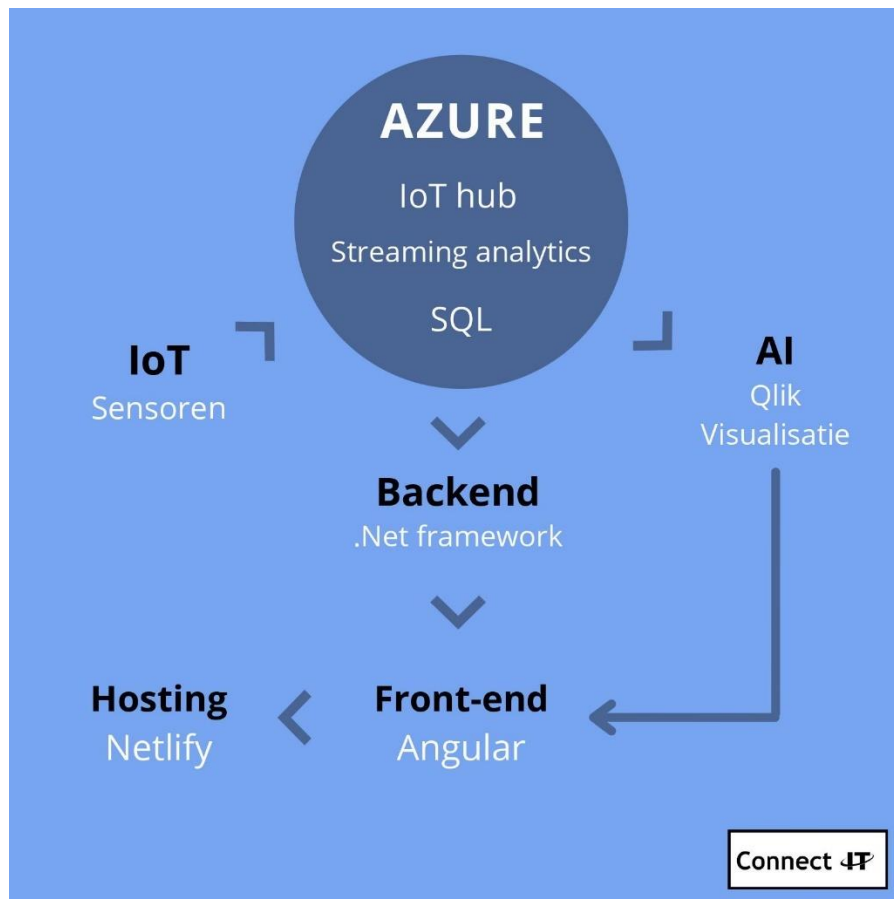
We kwamen op het idee het vuilnis ophaalsysteem te automatiseren. Hierbij hebben we een IoT opstelling gemaakt die we in de vuilbakken kunnen plaatsen. Dit zorgt voor een monitoring van verschillende factoren van de vuilbak.

Zo hebben we een sensor voor beweging en een drukknop, die een melding stuurt dat de vuilbak vol is aan de buitenkant. Aan de binnenkant hebben we sensoren die het volume, het gewicht en brand detecteren. Dit alles wordt verwezen naar een Azure omgeving waarop de centrale SQL Database staat. Onze website staat hier ook mee in connectie zodat we altijd de laatste nieuwe data op de site tonen.

We hebben deze handleiding opgesteld zodat het project opnieuw kan worden nagemaakt en in elkaar gezet voor eventuele infodagen.

Schematische voorstelling

We beginnen bij onze IoT. Die data wordt doorgestuurd naar Azure IoT Hub. Daarna wordt die data doorgestuurd naar de SQL Database door de streaming analytics job. De backend en Qlik visualisatie staan in verbinding met de SQL Database en dan wordt die data verstuurd naar de front-end, zodat het op de site te zien is. De site zelf wordt gehost door Netlify.



Inhoudstafel

Inleiding	2
Ons team	2
Onze opdracht	2
Schematische voorstelling	3
Stappenplan	5
Stap 1 - GitHub code	5
Stap 2 - Azure omgeving opstellen	6
IoT Hub	6
Streaming Analytics job	7
SQL Server & Database.....	8
CORS	10
Runbook	10
Stap 3 - Visual studio naar Azure	13
Stap 4 IoT.....	15
SAS token	15
PCB.....	17
Code	18
Stap 5 Qlik	24
Visualisaties inladen	24
Embedding	25
Jaaroverzicht	26
Map.....	26
Stap 6 Netlify	27

Stappenplan

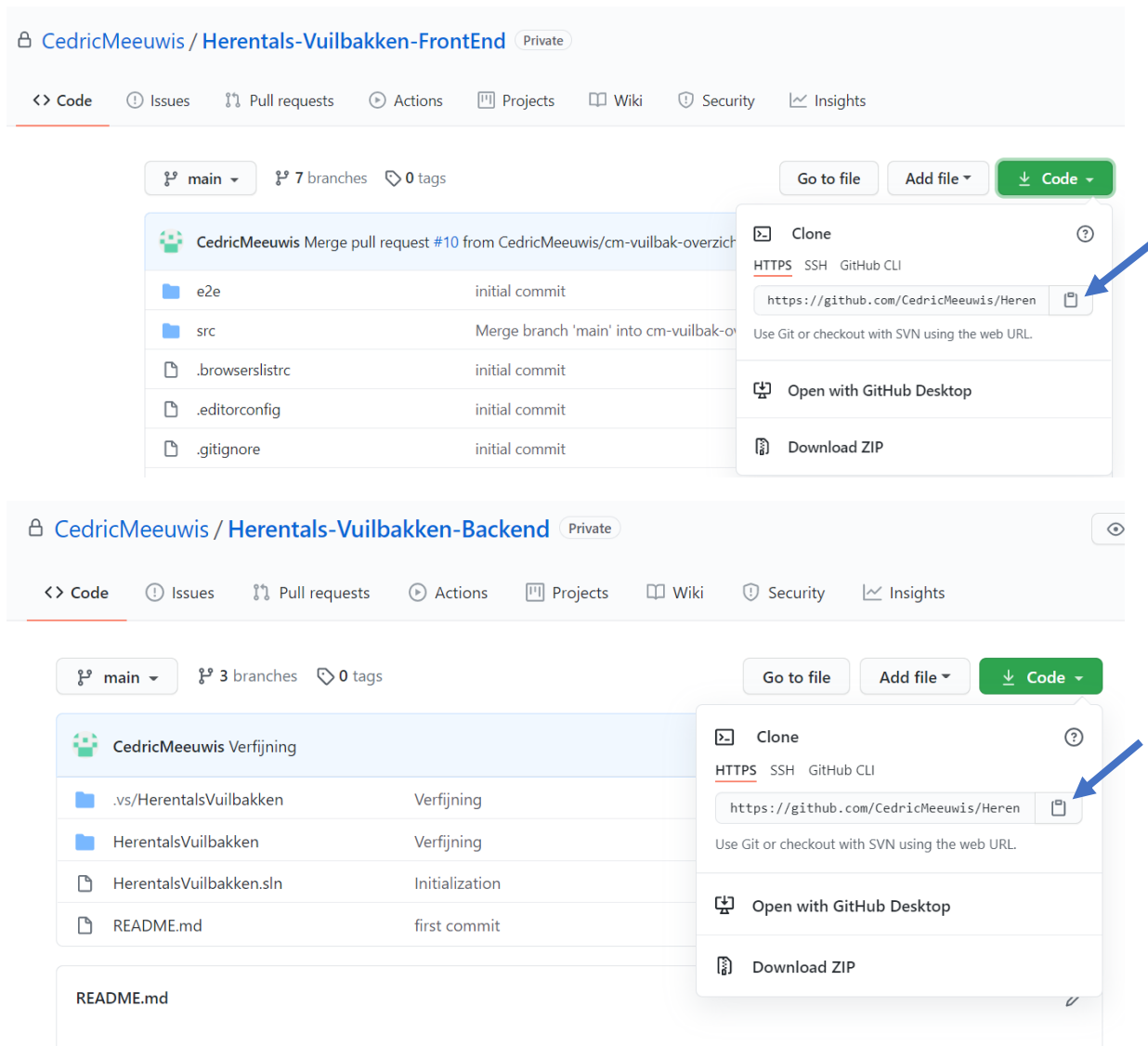
Stap 1 - GitHub code

Ga naar onderstaande links.

<https://github.com/CedricMeeuwis/Herentals-Vuilbakken-FrontEnd>

<https://github.com/CedricMeeuwis/Herentals-Vuilbakken-Backend>

U drukt op de volgende knop en kopieert de link.



U gaat naar een lokale map op uw computer en klikt rechtermuisklik "open git bash".

Voer het volgende commando uit:

Git clone <https://github.com/CedricMeeuwis/Herentals-Vuilbakken-Backend.git>

Git clone <https://github.com/CedricMeeuwis/Herentals-Vuilbakken-FrontEnd.git>

Nu staat de backend en front-end tot uw beschikking.

Stap 2 - Azure omgeving opstellen

Als u de Azure omgeving opstelt is het enorm belangrijk altijd dezelfde resource group aan te maken. Hieronder vindt u een overzicht over alle resources die gekoppeld zijn aan onze resource group genaamd connect-it-rg. Deze resources zijn allemaal nodig voor ons project werkend te krijgen. De voor de hand liggende resources zal ik niet bespreken zoals de app service, hosting plan, ... Wel de belangrijkste zullen dus uitgelegd worden, in deze stap, hoe die precies in elkaar zitten.

All resources ✕

Thomas More

+ Add Manage view Refresh Export to CSV Open query Assign tags Delete Feedback

Filter for any field... Subscription == all Resource group == all Type == all Location == all Add filter

Showing 1 to 100 of 101 records. Show hidden types Group by resource group List view

Name	Type	Resource group	Location	Subscription
connect-it-rg				
cit-app-service	App Service	connect-it-rg	West Europe	Azure for Students
cit-hosting-plan	App Service plan	connect-it-rg	West Europe	Azure for Students
cit-iot-hub	IoT Hub	connect-it-rg	West Europe	Azure for Students
cit-sql-db (cit-sql-server/cit-sql-db)	SQL database	connect-it-rg	West Europe	Azure for Students
cit-sql-server	SQL server	connect-it-rg	West Europe	Azure for Students
cit-update-sql	Automation Account	connect-it-rg	West Europe	Azure for Students
citstorageaccount123	Storage account	connect-it-rg	West Europe	Azure for Students
IoTDataToSQL-Live	Stream Analytics job	connect-it-rg	West Europe	Azure for Students
SaveDataToSQL	Stream Analytics job	connect-it-rg	West Europe	Azure for Students
SQL-update (cit-update-sql/SQL-update)	Runbook	connect-it-rg	West Europe	Azure for Students

IoT Hub

Bij onze IoT opstelling zijn er 2 scripts voor 2 apparaten. Eén apparaat representeert de binnenkant van de vuilbak (gewicht, volume en brand sensor) en het ander apparaat gaat over de buitenkant van de vuilbak (drukknop en bewegingssensor). BK staat voor binnenkant en BU staat voor buitenkant.

cit-iot-hub | IoT devices ✕

IoT Hub

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

Built-in endpoints

Failover

Properties

Locks

Explorers

Query explorer

IoT devices

Automatic Device Management

IoT Edge

IoT device configuration

Messaging

File upload

Message routing

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

Field Operator Value

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

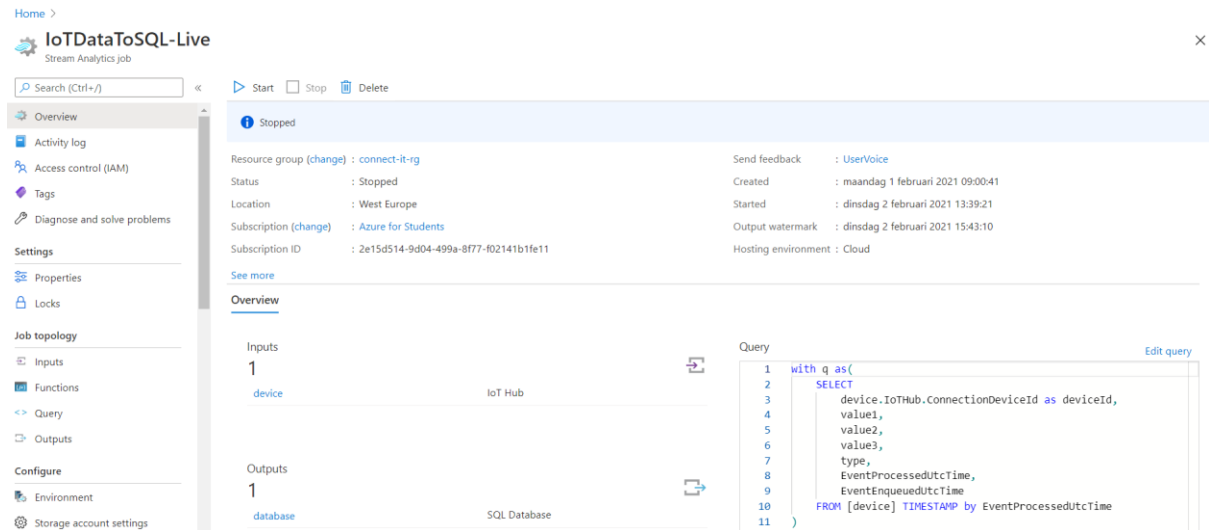
+ Add a new clause

Query devices </> Switch to query editor

Device ID	Status	Last Status Update (UTC)	Authentication Type	Cloud to Device Messa...
cit-iot-device-bk	Enabled	--	Sas	0
cit-iot-device-bu	Enabled	--	Sas	0

Streaming Analytics job

Hier ziet u het overzicht van onze streaming analytics job. Dit zorgt ervoor dat de data van IoT Hub naar de SQL Database wordt herleid. IoT Hub wordt hier altijd gezien als 1 input stream en staat voor 'device'.



De code die bij de query wordt gebruikt is de volgende:

```
with q as(
    SELECT
        device.IoTHub.ConnectionDeviceId as deviceId,
        value1,
        value2,
        value3,
        type,
        EventProcessedUtcTime,
        EventEnqueuedUtcTime
    FROM [device] TIMESTAMP by EventProcessedUtcTime
)
SELECT
    deviceId,
    value1,
    avg(value2) as value2,
    value3,
    type,
    EventProcessedUtcTime,
    EventEnqueuedUtcTime
INTO [database]
FROM q
partition by value1
group by deviceId, value1, value2, value3, type, EventProcessedUtcTime, EventEnqueuedUtcTime, TumblingWindow(second, 10)
```

Bij job topologie moet je de input en output instellen. Bij input moet je 'add stream input' klikken en checken of de gegevens kloppen. Bij output voeg je de SQL database toe en verwijst je naar de iotlive tabel.

Output details

database

Test Delete

Server name *
cit-sql-server.database.windows.net

Table *
iotlive

Username *
connect-it

Password *
.....

☒ Merge all input partitions into a single writer
☐ Inherit partition scheme of previous query step or input

Max batch count ⓘ
10000

Input details

device

Test Delete

IoT Hub * ⓘ
cit-iot-hub

Consumer group ⓘ
\$Default

Shared access policy name * ⓘ
iothubowner

Shared access policy key ⓘ
.....

Endpoint ⓘ
Messaging



Partition key ⓘ

Event serialization format * ⓘ
JSON

Wanneer alles is ingesteld kan je bij het overzicht de job starten en zal de data verstuurd worden naar je iotlive tabel.

SQL Server & Database

Het is belangrijk de SQL server en database juist in te stellen. Hier wordt uiteindelijk al je data in bewaard. Dit is het overzicht van onze SQL database.

**cit-sql-db (cit-sql-server/cit-sql-db)** 
SQL database

Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview

Activity log

Tags

Diagnose and solve problems

Quick start

Query editor (preview)

Power Platform

Power BI (preview)

Power Apps (preview)

Essentials

Resource group (change)
connect-it-rg

Status
Online

Location
West Europe

Subscription (change)
Azure for Students

Subscription ID
2e15d514-9d04-499a-8f77-f02141b1fe11

Tags (change)
Click here to add tags

Server name
cit-sql-server.database.windows.net


Elastic pool
No elastic pool

Connection strings
Show database connection strings

Pricing tier
Basic


Earliest restore point
2021-01-25 08:17 UTC

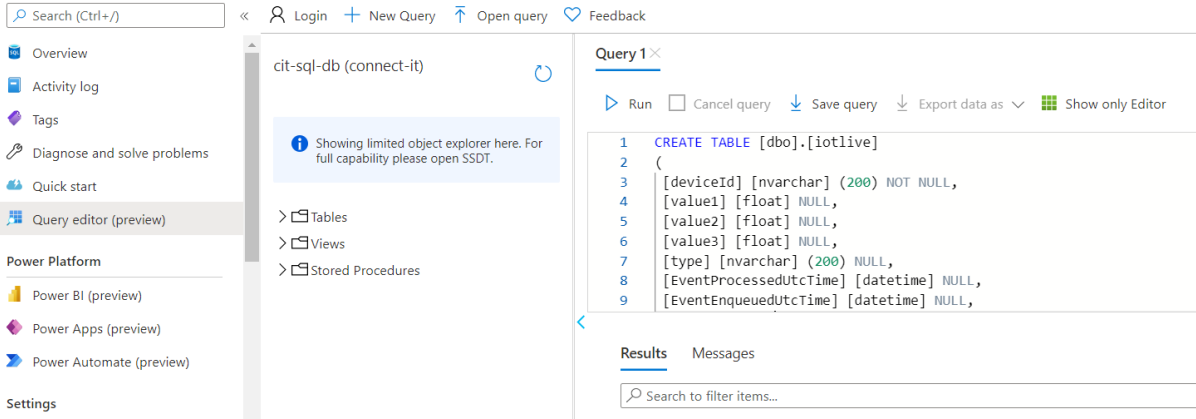
Telkens als je verbinding maakt met Azure via een andere client ip moet je dit toelaten op de server firewall.

Connect it 

Door de backend te publishen komen alle tabellen die je nodig hebt voor de site op de SQL database maar dan moet je de iotlive tabel nog toevoegen en dat doe je bij de query editor op de SQL database.

[Home](#) > [cit-sql-db \(cit-sql-server/cit-sql-db\)](#)

 **cit-sql-db (cit-sql-server/cit-sql-db)** | Query editor (preview)
SQL database



Search (Ctrl+/) « Login + New Query ↑ Open query Feedback

Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
Power Platform
Power BI (preview)
Power Apps (preview)
Power Automate (preview)
Settings

cit-sql-db (connect-it)

Showing limited object explorer here. For full capability please open SSDT.

Tables
Views
Stored Procedures

Query 1 ×

Run Cancel query Save query Export data as Show only Editor

```
1 CREATE TABLE [dbo].[iotlive]
2 (
3     [deviceId] [nvarchar] (200) NOT NULL,
4     [value1] [float] NULL,
5     [value2] [float] NULL,
6     [value3] [float] NULL,
7     [type] [nvarchar] (200) NULL,
8     [EventProcessedUtcTime] [datetime] NULL,
9     [EventEnqueuedUtcTime] [datetime] NULL,
```

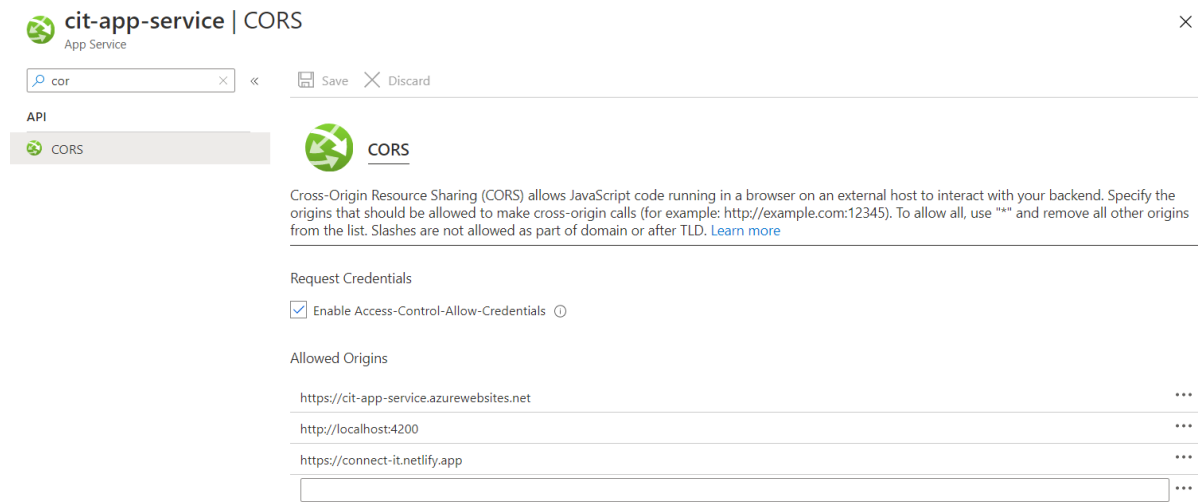
Results Messages

Search to filter items...

```
CREATE TABLE [dbo].[iotlive]
(
    [deviceId] [nvarchar] (200) NOT NULL,
    [value1] [float] NULL,
    [value2] [float] NULL,
    [value3] [float] NULL,
    [type] [nvarchar] (200) NULL,
    [EventProcessedUtcTime] [datetime] NULL,
    [EventEnqueuedUtcTime] [datetime] NULL,
) ON [PRIMARY]
```

CORS

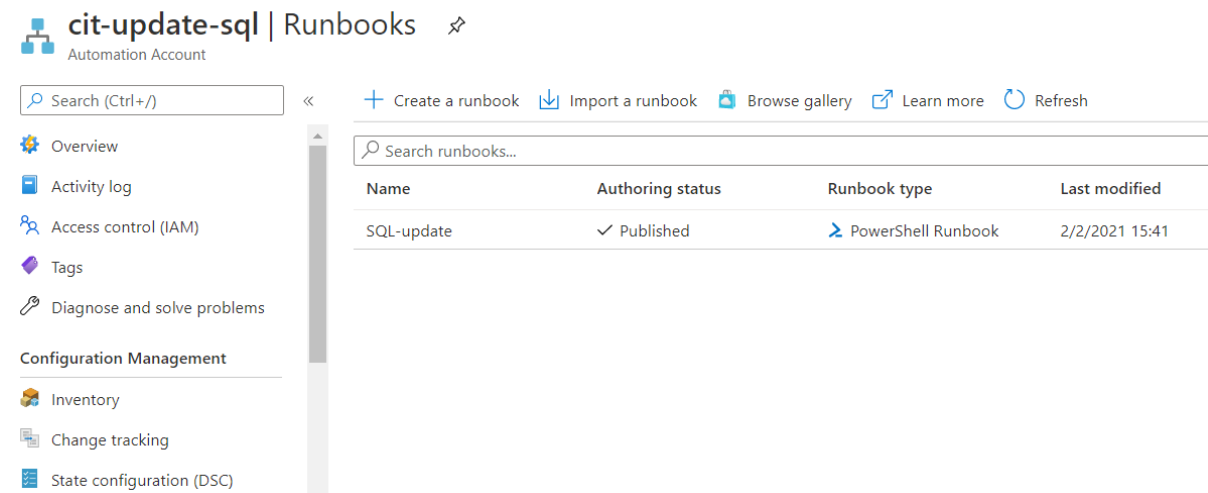
Dit zorgt ervoor dat API calls worden toegelaten van een extern domein.



The screenshot shows the 'cit-app-service | CORS' configuration page in the Azure portal. The left sidebar has 'API' and 'CORS' tabs, with 'CORS' selected. The main area contains a search bar with 'cor', 'Save' and 'Discard' buttons, and a description of CORS. Below this, the 'Request Credentials' section has a checked box for 'Enable Access-Control-Allow-Credentials'. The 'Allowed Origins' section lists three origins: 'https://cit-app-service.azurewebsites.net', 'http://localhost:4200', and 'https://connect-it.netlify.app', each with a three-dot menu icon to its right.

Runbook

Voor de IoT data live op de site te krijgen maken we gebruik van een runbook. We hebben hier een automation account voor nodig. Ons automation account noemt cit-update-SQL en ons runbook dat hieraan gekoppeld is hebben we SQL-update genoemd.



The screenshot shows the 'cit-update-sql | Runbooks' page in the Azure portal. The left sidebar has a search bar and a list of navigation items: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration Management, Inventory, Change tracking, and State configuration (DSC). The main area has a search bar, buttons for 'Create a runbook', 'Import a runbook', 'Browse gallery', 'Learn more', and 'Refresh'. Below this is a table with the following data:

Name	Authoring status	Runbook type	Last modified
SQL-update	✓ Published	PowerShell Runbook	2/2/2021 15:41

Hieronder ziet u het overzicht van ons runbook en zien we dat we deze kunnen starten en linken aan een schedule. Wij hebben dit op herhalend om het uur gezet. Je kan dit ook manueel update door op 'Start' te klikken. Dit is perfect voor bij een demo.

SQL-update (cit-update-sql/SQL-update) ✨

Runbook

Start View Edit Link to schedule Add webhook Delete Export Refresh

Overview

- Activity log
- Tags
- Diagnose and solve problems

Resources

- Jobs
- Schedules
- Webhooks

Runbook settings

- Properties
- Description
- Logging and tracing

Settings

- Locks

Essentials

Resource group : [connect-it-rg](#) Subscription ID : 2e15d514-9d04-499a-8f77-f02141b1fe11
Account : cit-update-sql Status : Published
Location : West Europe Runbook type : PowerShell Runbook
Subscription : [Azure for Students](#) Last modified : 2/2/2021 15:41
Tags (change) : [Click here to add tags](#)

Recent Jobs

Status	Created	Last updated
✓ Completed	2/2/2021 15:38:38	2/2/2021 15:39:03
✓ Completed	2/2/2021 15:38:28	2/2/2021 15:38:57
✓ Completed	2/2/2021 15:35:53	2/2/2021 15:36:18
✓ Completed	2/2/2021 15:33:19	2/2/2021 15:33:42
✓ Completed	2/2/2021 15:26:09	2/2/2021 15:26:38
✓ Completed	2/2/2021 15:22:24	2/2/2021 15:23:03

Onderstaand venster ziet u wanneer u op Start heeft gedrukt en kan u checken of er errors zijn.

[Home](#) > [Automation Accounts](#) > [cit-update-sql](#) > [SQL-update \(cit-update-sql/SQL-update\)](#) >

SQL-update 3/2/2021 09:00

Job

 Resume  Stop  Suspend  Refresh

^ Essentials

Id : 1fd49ace-f74a-42db-beba-e42f4df0610f

Created : 3/2/2021 09:00:18

Status : Queued

Last Update : 3/2/2021 09:00:18

Ran ... : Azure

Runbook : [SQL-update](#)

Ran ... : User

Source snaps... : [View source snapshot](#)

[Input](#) [Output](#) [Errors](#) [Warnings](#) [All Logs](#) [Exception](#)

Errors

0 

Time	Type	Details
------	------	---------

No errors found for the job

Hieronder vind u het script dat we gebruikt hebben voor de SQL-update job. Belangrijk hierbij is dat het een Powershell script moet zijn.

```
$AzureSQLServerName = "cit-sql-server.database.windows.net"
```

```
$AzureSQLDatabaseName = "cit-sql-db"
```

```
$Cred = Get-AutomationPSCredential -Name "cit-sql-credential"
```

```
$SQLOutput = $(Invoke-Sqlcmd -ServerInstance $AzureSQLServerName -Username  
$Cred.UserName -Password $Cred.GetNetworkCredential().Password -Database  
$AzureSQLDatabaseName -Query "UPDATE dbo.Vuilbak SET Gewicht=(select top 1  
value1 from dbo.iotlive WHERE type='binnenkant' order by EventProcessedUtcTime  
desc), Volheid=(select top 1 value2 from dbo.iotlive WHERE type='binnenkant' order by  
EventProcessedUtcTime desc), Brand=(select top 1 value3 from dbo.iotlive WHERE  
type='binnenkant' order by EventProcessedUtcTime desc) WHERE VuilbakID=1" -  
QueryTimeout 65535 -ConnectionTimeout 10 -Verbose) 4>&1
```

```
Write-Output $SQLOutput
```

Onze code zit zo in elkaar dat we een verwijzing moeten doen van de iotlive tabel naar 3 kolommen in de Vuilbak tabel namelijk 'Volheid', 'Gewicht' en 'Brand'. Hierbij hebben we gekozen dat Vuilbak met ID 1 onze vuilbak voor live data wordt.

Stap 3 - Visual studio naar Azure

U kan de backend publishen naar Azure. Zorg ervoor dat uw client ip adres wordt toegelaten op de database firewall.

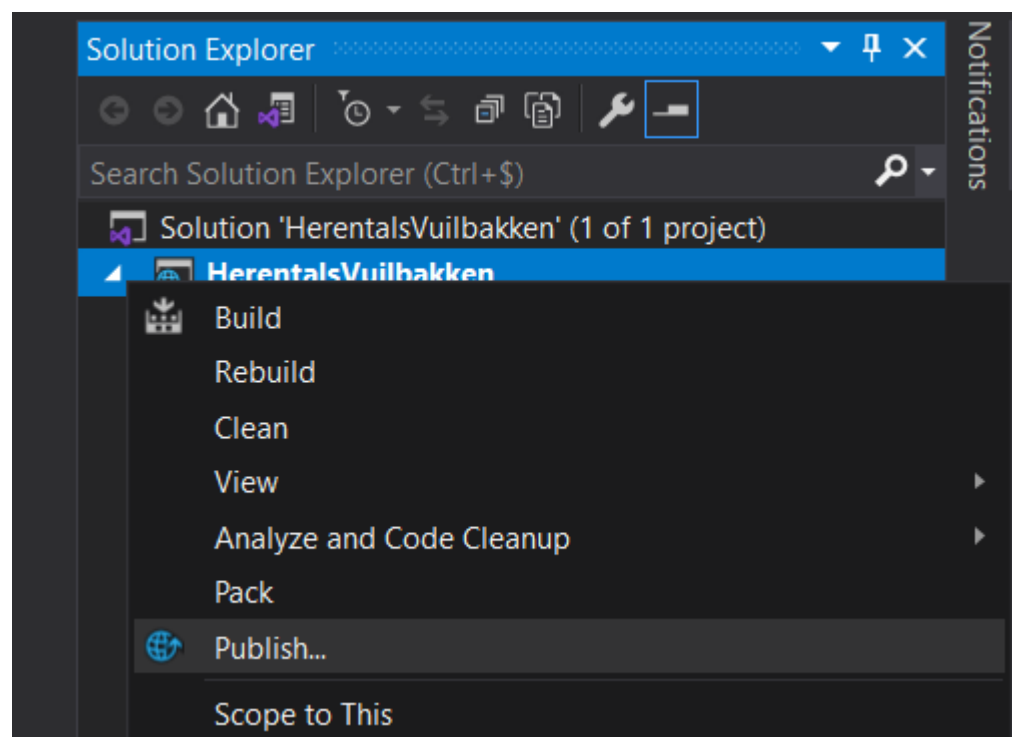
Doorloop de volgende stappen om de backend te publishen.

Zorg dat het pad waarin u de backend heeft gezet geen spaties bevat in de namen van de mappen.

> This PC > Data (D:) > Project4.0 > Herentals-Vuilbakken-Backend >		
	Name	Date modified
Files	.vs	25/01/2021 9:42
	HerentalsVuilbakken	25/01/2021 9:42
	HerentalsVuilbakken.sln	25/01/2021 9:42
	README.md	25/01/2021 9:42

Klik dan rechtermuisklik op "HerentalsVuilbakken.sln" en open dit met Visual studio.

Daarna kan u rechtermuisklikken op de titel van het project en "publish" klikken.



Druk 2x 'next' en voor dit te kunnen instellen zal u moeten inloggen met de persoonlijke email van onze hosting student. Contacteer ons hier ook voor.

Publish

Select existing or create a new Azure App Service

Target

Subscription

Azure for Students

Specific target

View

Resource group

App Service

Search

API Management

App Service instances

connect-it-rg

Back

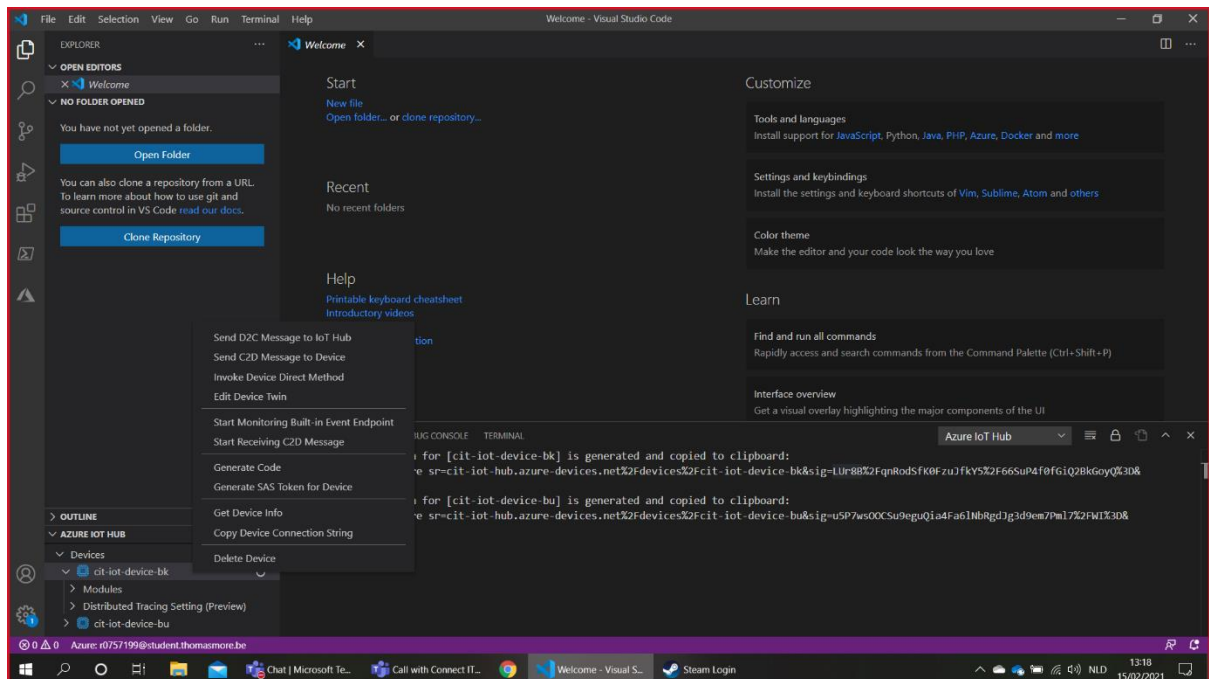
Next

Finish

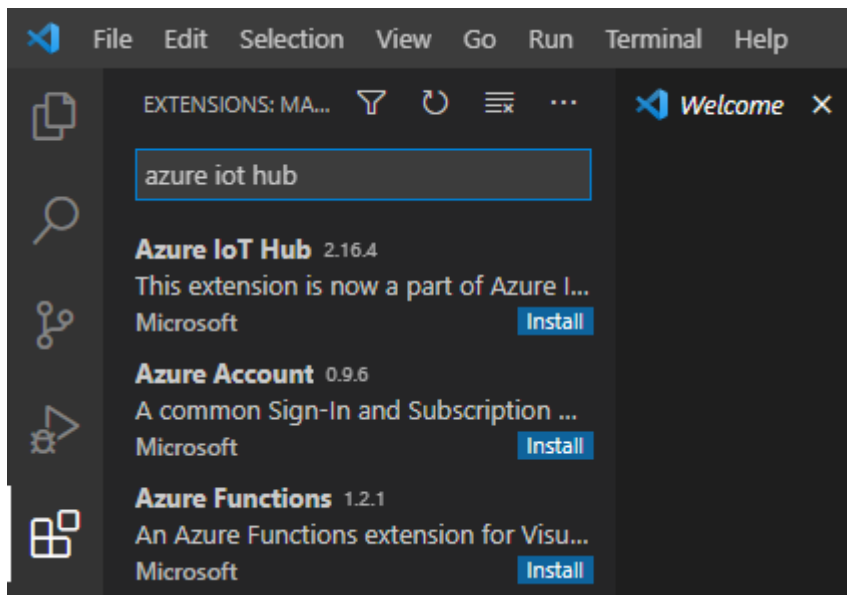
Cancel

Stap 4 IoT

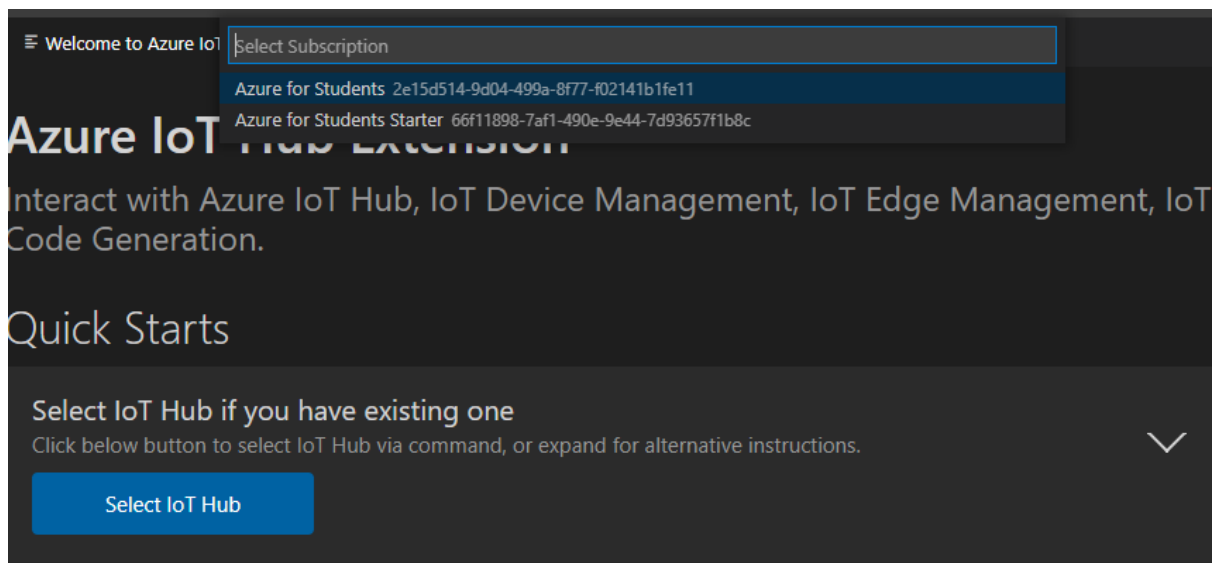
SAS token



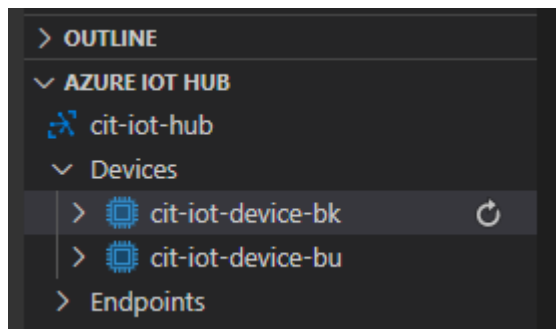
Maak verbinding met Visual Studio code en Azure IoT Hub. Dit kan je doen door links op extensies te drukken en vervolgens Azure IoT hub te installeren. Daarna zal je moeten inloggen op je Azure en wordt je herleid naar een pagina dat je zegt dat je bent ingelogd.



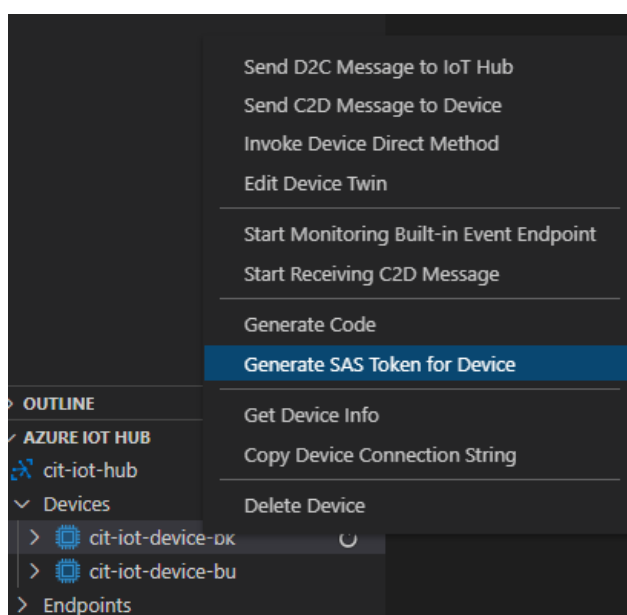
Vervolgens druk je op "Select IoT Hub" en kies je jouw subscriptie en IoT hub.



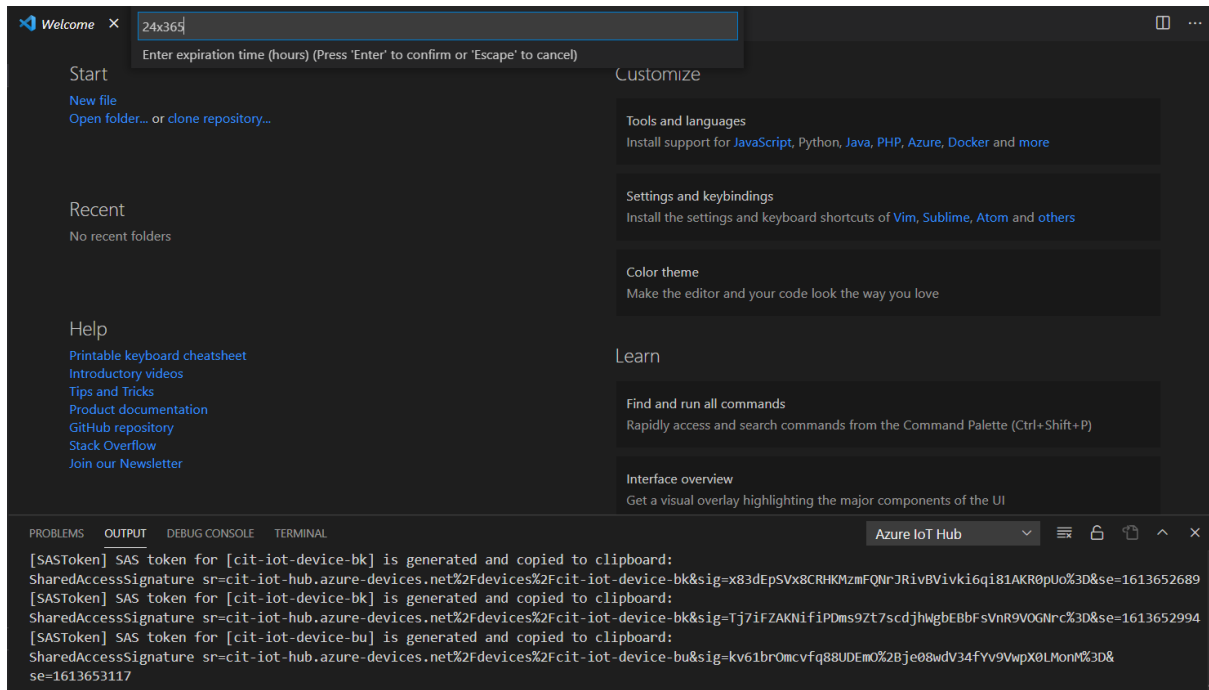
Je zal dan zien dat er links onderaan connectie is met je IoT hub.



Hierna gaan we een SAS token genereren, dit zorgt voor de verbinding tussen het IoT device en Azure. Dit doe je door rechtermuisklik op je device te klikken (binnenkant of buitenkant). Je kiest dan voor "Generate SAS token for device".

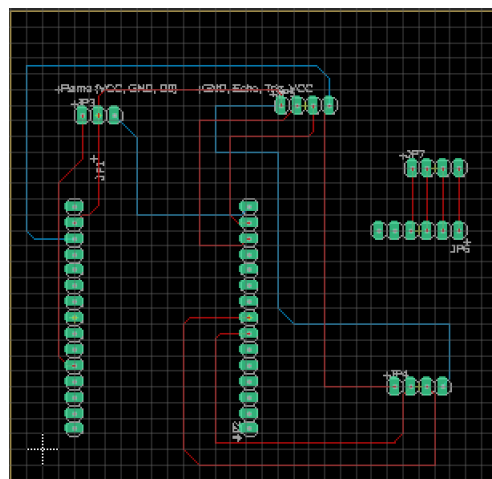
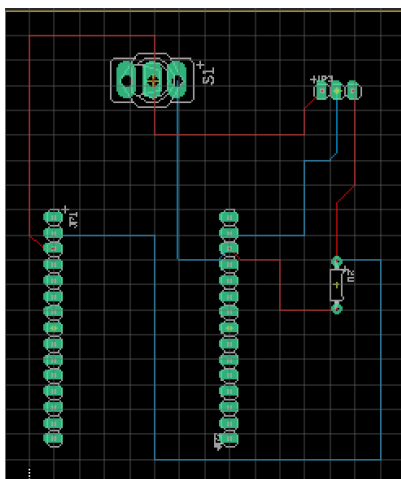


Nu komt er vanboven een tekst vak te staan. Hier moet je 24x365 (24 uur maal 365 dagen = 1 jaar) intikken, dan is je SAS token 1 jaar geldig. Hierna moet je ENTER klikken en je krijgt je SAS token. Nu moet je enkel nog je SAS token op de juiste plek in je code zetten.



PCB

Hierna neem je alle componenten en soldeer je deze op de PCB's. De PCB tekeningen zien er als volgt uit:



Code

Wanneer de componenten gesoldeerd zijn moet je de Arduino code op de ESP8266 zetten. Hieronder kan u de code van de binnenkant en die van de buitenkant terugvinden.

Binnenkant:

```
#include <ESP8266WiFi.h>           // Include the Wi-Fi library
#include <ESP8266WiFiMulti.h>
#include <WiFiClientSecure.h>
#include <WiFiClient.h>
ESP8266WiFiMulti wifiMulti;       // Create an instance of the
ESP8266WiFiMulti class, called 'wifiMulti'
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include "HX711.h"
#include <EmailSender.h>

//EmailSender emailSend("connectitla@gmail.com", "Cit123456!");
uint8_t connection_state = 0;
uint16_t reconnect_interval = 10000;

#define calibration_factor 1380 //This value is obtained using the
SparkFun_HX711_Calibration sketch
#define LOADCELL_DOUT_PIN  D5
#define LOADCELL_SCK_PIN  D6
HX711 scale;

const int flame = D0;
int trigPin = D1;           // HC-SR04 trigger pin
int echoPin = D2;           // HC-SR04 echo pin
//int potmeter = A0;
float duration, distance;
float gewicht;

const char* mqtt_server = "cit-iot-hub.azure-devices.net"; // hostname IoT
hub
const char* clientId = "cit-iot-device-bk"; // device ID
const char* clientUser = "cit-iot-hub.azure-devices.net/cit-iot-device-bk";
// hub/deviceID/?api-version=2018-06-30
const char* clientPass = "SharedAccessSignature sr=cit-iot-hub.azure-
devices.net%2Fdevices%2Fcit-iot-device-
bk&sig=LUr8B%2FqnRodSfK0FzuJfkY5%2F66SuP4f0fGiQ2BkGoyQ%3D&se=1613477773";
// SASS token from deviceId
const char* topic = "devices/cit-iot-device-bk/messages/events/";
int port = 8883; // secured connection
const char* fingerprint =
"C7:06:8B:B0:31:2C:35:9B:B4:1B:79:5E:8D:D8:C9:78:77:E1:71:99";

BearSSL::WiFiClientSecure espClient;
//BearSSL::X509List cert(server_cert);

PubSubClient client(espClient);
long lastMsg = 0;
int value = 0;

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
```

```

    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect(clientId, clientUser, clientPass)) {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

String getSensorValuesJSON(float value1, float value2, float value3, char*
type ){
    DynamicJsonDocument doc(84);

    doc["value1"] = value1;
    doc["value2"] = value2;
    doc["value3"] = value3;
    doc["type"] = type;

    String output;
    serializeJson(doc, output);
    return output;
}

void setup() {
    Serial.begin(115200);
    delay(200);
    wifiMulti.addAP("Linksys00111", "luf866krpp"); // add Wi-Fi networks
you want to connect to
    wifiMulti.addAP("Linksys00111_2GEXT", "luf866krpp");
    wifiMulti.addAP("AndroidAP", "iotislife");
    Serial.println("---");
    Serial.println("Connecting ...");
    int i = 0;
    while (wifiMulti.run() != WL_CONNECTED) { // Wait for the Wi-Fi to
connect: scan for Wi-Fi networks, and connect to the strongest of the
networks above
        delay(1000);
        Serial.print('.');
    }
    Serial.println('\n');
    Serial.print("Connected to ");
    Serial.println(WiFi.SSID()); // Tell us what network we're
connected to
    Serial.print("IP address:\t");
    Serial.println(WiFi.localIP()); // Send the IP address of the
ESP8266 to the computer

```

```

    scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
    scale.set_scale(calibration_factor); //This value is obtained by using
the SparkFun_HX711_Calibration sketch
    scale.tare(); //Assuming there is no weight on the scale at start up,
reset the scale to 0

    pinMode(trigPin, OUTPUT); // define trigger pin as output
    pinMode(flame, INPUT);

    espClient.setFingerprint(fingerprint);
    client.setServer(mqtt_server, 8883);
    client.setCallback(callback);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    long now = millis();
    if (now - lastMsg > 10000) {
        lastMsg = now;

        digitalWrite(echoPin, LOW); // set the echo pin LOW
        digitalWrite(trigPin, LOW); // set the trigger pin LOW
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH); // set the trigger pin HIGH for 10µs
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH); // measure the echo time (µs)
        distance = (duration/2.0)*0.0343; // convert echo time to distance
(cm)
        delay(800);
        Serial.println("-----");
        if(distance>400 || distance<2) Serial.println("Buiten bereik");
        else
        {
            Serial.print("Afstand = ");
            Serial.print(distance, 1); Serial.println(" cm");
        }
        int t = digitalRead(flame);
        if (t==0) {
            t = 1;
        }
        else if (t==1){
            t = 0;
        }
        Serial.print("Gewicht: ");
        gewicht = scale.get_units();
        Serial.print(gewicht); //scale.get_units() returns a float
        Serial.println(" gr"); //You can change this to kg but you'll need to
refactor the calibration_factor

        Serial.println("Verzonden bericht: ");

        String Payload = getSensorValuesJSON(gewicht, distance, t,
"binnenkant");
        Serial.println(Payload);
        client.publish(topic, Payload.c_str());
    }
}

```

```

        Serial.println("-----");
    }
}

```

Buitenkant:

```

#include <ESP8266WiFi.h>           // Include the Wi-Fi library
#include <ESP8266WiFiMulti.h>
#include <WiFiClientSecure.h>
#include <WiFiClient.h>
ESP8266WiFiMulti wifiMulti;       // Create an instance of the
ESP8266WiFiMulti class, called 'wifiMulti'
#include <PubSubClient.h>
#include <ArduinoJson.h>

int switch_pin = D2;
int sensor = D1;  // Digital pin D7
int stort = 0;

const char* mqtt_server = "cit-iot-hub.azure-devices.net"; // hostname IoT
hub
const char* clientId = "cit-iot-device-bu"; // device ID
const char* clientUser = "cit-iot-hub.azure-devices.net/cit-iot-device-bu";
// hub/deviceID/?api-version=2018-06-30
const char* clientPass = "SharedAccessSignature sr=cit-iot-hub.azure-
devices.net%2Fdevices%2Fcit-iot-device-
bu&sig=u5P7wsOOCsu9eguQia4Fa6lNbRgdJg3d9em7Pml7%2FWI%3D&se=1613477893"; //
SASS token from deviceId
const char* topic = "devices/cit-iot-device-bu/messages/events/";
int port = 8883; // secured connection
const char* fingerprint =
"C7:06:8B:B0:31:2C:35:9B:B4:1B:79:5E:8D:D8:C9:78:77:E1:71:99";

BearSSL::WiFiClientSecure espClient;
//BearSSL::X509List cert(server_cert);

PubSubClient client(espClient);
long lastMsg = 0;
int value = 0;

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect(clientId, clientUser, clientPass)) {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());

```

```

        Serial.println(" try again in 5 seconds");
        // Wait 5 seconds before retrying
        delay(5000);
    }
}
}
String getSensorValuesJSON(float value1, float value2, float value3, char*
type ){
    DynamicJsonDocument doc(84);
    doc["value1"] = value1;
    doc["value2"] = value2;
    doc["value3"] = value3;
    doc["type"] = type;
    String output;
    serializeJson(doc, output);
    return output;
}

void setup() {
    Serial.begin(115200);
    delay(100);
    wifiMulti.addAP("Linksys00111", "luf866krpp");    // add Wi-Fi networks
you want to connect to
    wifiMulti.addAP("Linksys00111_2GEXT", "luf866krpp");
    wifiMulti.addAP("AndroidAP", "iotislife");
    Serial.println("---");
    Serial.println("Connecting ...");
    int i = 0;
    while (wifiMulti.run() != WL_CONNECTED) { // Wait for the Wi-Fi to
connect: scan for Wi-Fi networks, and connect to the strongest of the
networks above
        delay(1000);
        Serial.print('.');
    }
    Serial.println('\n');
    Serial.print("Connected to ");
    Serial.println(WiFi.SSID());                    // Tell us what network we're
connected to
    Serial.print("IP address:\t");
    Serial.println(WiFi.localIP());                // Send the IP address of the
ESP8266 to the computer

    pinMode(switch_pin, INPUT);
    pinMode(sensor, INPUT);    // declare sensor as input

    espClient.setFingerprint(fingerprint);
    client.setServer(mqtt_server, 8883);
    client.setCallback(callback);
}
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    long now = millis();
    if (now - lastMsg > 10000) {
        lastMsg = now;

        if(digitalRead(switch_pin) == HIGH){
            Serial.println("Er is gesluikstort");

```

```

    stort = 1;
}
if(digitalRead(switch_pin) == LOW){
    Serial.println("Geen meldingen over sluikstorten ontvangen");
    stort = 0;
}
long state = digitalRead(sensor);
Serial.println(state);
delay(600);

Serial.println("Verzonden bericht: ");

String buitenPayload = getSensorValuesJSON(state, stort, 0,
"buitenkant");
Serial.println(buitenPayload);
client.publish(topic, buitenPayload.c_str());

Serial.println("-----");
}
}

```

Wanneer de code op de ESP staat werkt het allemaal zeer simpel. Je moet het device gewoon voeden aan de netstroom, met een powerbank of een batterij en het werkt. De sensoren zullen alles opmeten en doorsturen naar Azure.

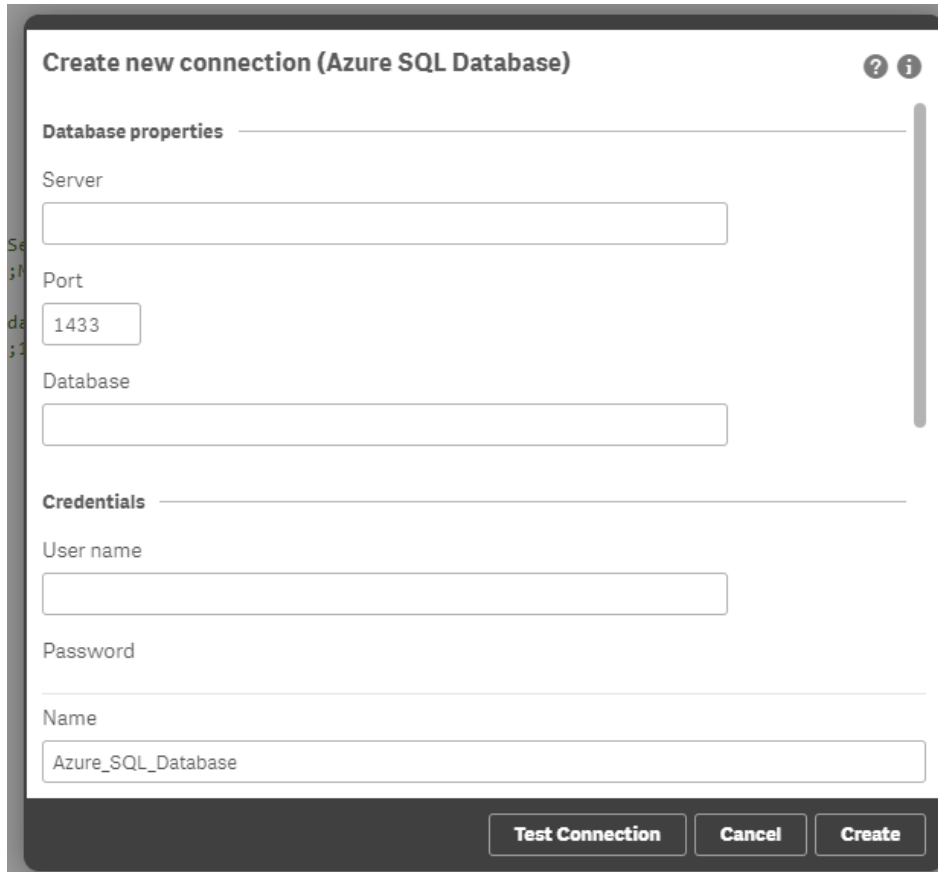
Stap 5 Qlik

Visualisaties inladen

Om de Qlik visualisatie op de site te laten zien, moet je Qlik pagina ook openstaan. Dit kopieert elkaar en staat in verbinding.

Eerst moet je inloggen op Qlik om de visualisaties in te laden. Deze kan je inladen door links vanboven op 'App toevoegen.'

Hierna moet je bij de app die **ConnectIT-map** noemt openen. Klik nu op 'datamodel -> data load editor'. Klik hier rechts vanboven op 'Create new connection.'



Create new connection (Azure SQL Database)

Database properties

Server

Port

Database

Credentials

User name

Password

Name

Test Connection **Cancel** **Create**

Vul hier in:

Server: cit-sql-server.database.windows.net

database: cit-sql-db

user name: connect-it

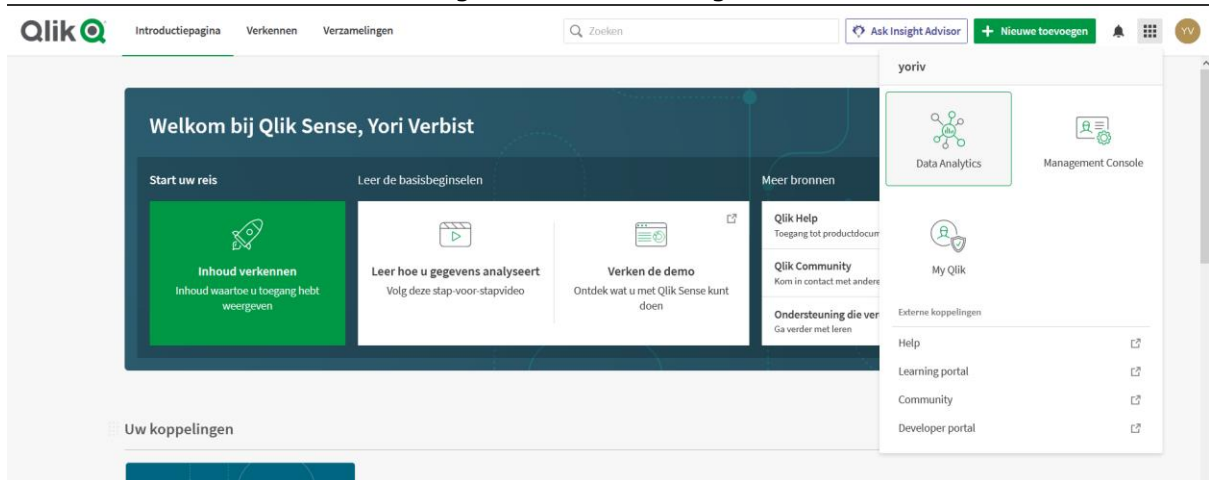
password: Cit123456

Als je nu op 'test connection' klikken krijg je een foutmelding met een IP-adres in. Voeg dit IP-adres toe aan de firewall van de SQL-database. Nu kan je wel verbinding maken.

Upload nu ook de file 'ConnectIT-jaaroverzicht' in Qlik. Hier moet niets meer aan aangepast worden.

Embedding

Klik linksboven om naar de management console te gaan:



Klik nu links in de balk op web.

Geef nu de volgende gegevens in (de link is de link die je later in NNetlify moet ingeven, deze kan je ook anders noemen als 'visualisaties' al bezet is. Pas dit hier dan ook aan):

Webintegratie maken

Naam

embedding

Oorsprongen

Een oorsprong toevoegen

https://visualisaties.netlify.app

Toevoegen

Toegestane oorsprongen

De lijst is leeg.

Er zijn geen toegestane oorsprongen voor deze webintegratie.

Annuleren

Maken

Clone nu de volgende GitHub repo: <https://github.com/YoriVerbist/Herentals-Vuilbakken-Visualisaties>.

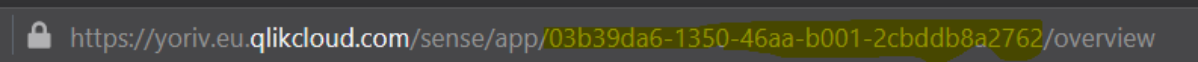
Jaaroverzicht

Open deze files in een editor en ga naar het index.js bestand.

Pas hier lijn 13 en 16 aan. 13 is de naam van je eigen qlik connectie. In qlik nadat je bij web een link naar de netlify site hebt gemaakt komt er een ID te staan. Kopieer dit ID naar lijn 16. Herhaal dit ook bij lijn 42 en 46.

13	<code>const urlQlikServer = "https://yoriv.eu.qlikcloud.com";</code>
14	<code>const urlLoggedIn = "/api/v1/audits"; //Use GET request to see if you are authenticated</code>
15	<code>const urlLogin = "/login";</code>
16	<code>const webIntegrationId = 'EE0oDIgP234lUdXymAF9JJ0Bd3t4r9f0';</code>
embedding	23Uq5SVyji3Yy4X2dC2rx1g4yY2_gfWm

Pas nu op lijn 65 de id aan. Dit ID vindt je als je het project opendoet in Qlik in de URL.

64	<code>//open apps -- inserted here --</code>
65	<code>var app = qlik.openApp("03b39da6-1350-46aa-b001-2cbddb8a2762", config1);</code>
66	<code>//get objects -- inserted here --</code>
67	<code>app.getObject('QV01', 'NMsuj'); // map garbagebins</code>
68	<code>app.getObject('QV02', 'kDLMZuP'); // average weight vs volume</code>
69	<code>app.getObject('QV03', 'FJKhKb'); // average volume</code>
70	<code>app.getObject('QV04', 'zPxpPuJ'); // totale volume/day</code>
71	<code>app.getObject('QV05', 'gpeeZ'); // avarage volume/weekday</code>
72	<code>app.getObject('QV06', 'tDbNbLY'); // average volume/month</code>
73	<code>app.getObject('QV07', 'Ljrtg'); // KPI totale volume</code>
74	<code>app.getObject('QV08', 'aFtXFcR'); // Drill-down</code>
75	
	

Pas in index.html op lijn 9 en 14 het deel aan als dezelfde manier als je in index.js op lijn 13.

7	<code><link</code>
8	<code>rel="stylesheet"</code>
9	<code>href="https://r0743694.eu.qlikcloud.com/resources/autogenerated/qlik-styles.css"</code>
10	<code>></code>
11	<code><link rel="stylesheet" href="style.css" /></code>
12	<code><script</code>
13	<code>type="text/javascript"</code>
14	<code>src="https://r0743694.eu.qlikcloud.com/resources/assets/external/requirejs/require.js"</code>
15	<code>></script></code>

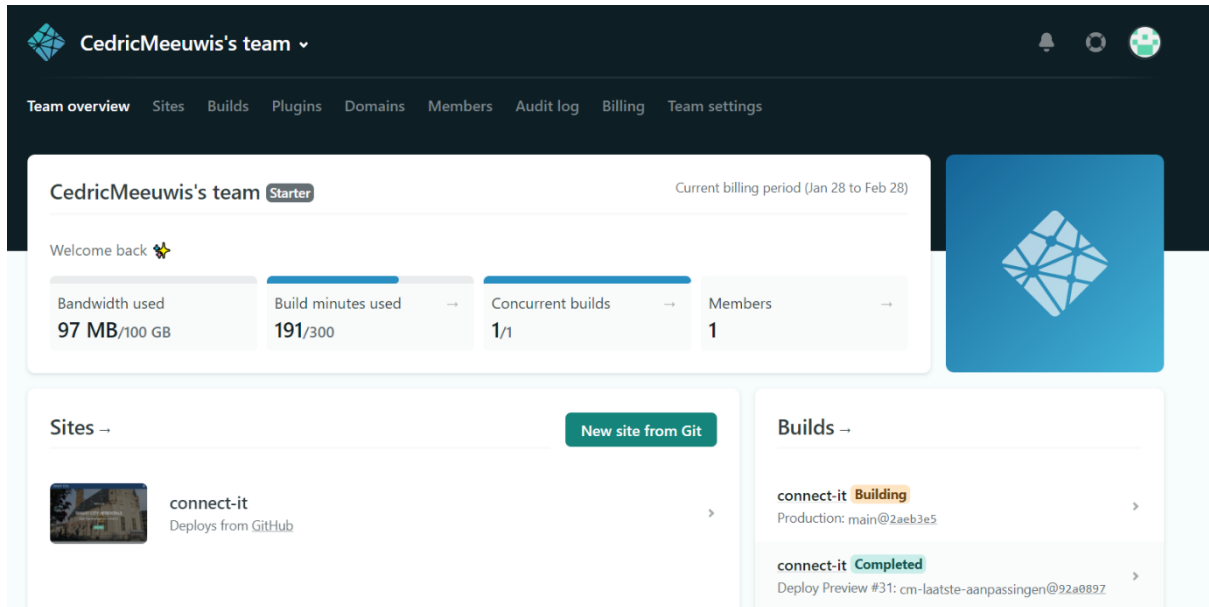
Map

Herhaal dezelfde stappen met het ander project (ConnectIT-map) en pas de gegevens in de file map.js en map.html aan.

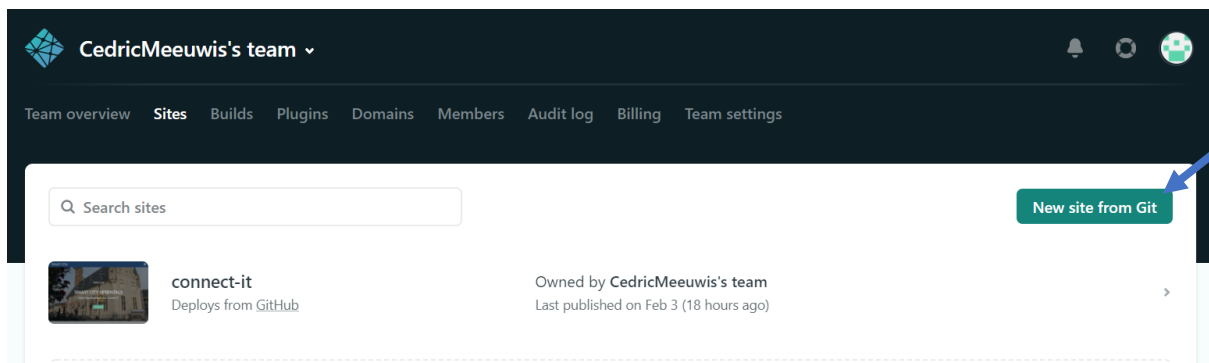
Stap 6 Netlify

We gebruiken Netlify om onze site te hosten. Het handige hier is dat telkens als er een git push wordt gedaan de site automatisch kan gepubliceerd worden. Het handigste is om je aan te melden met je GitHub account op Netlify, dan worden je bestanden op GitHub meteen beschikbaar voor Netlify.

Eerst maak je hier een team aan. Daarna kan je een overzicht zien van jouw team.



Daarna kan je een site aanmaken van GitHub.



Hieronder vindt u onze build settings. Zorg ervoor dat je het build command juist overneemt en de dist folder kiest met je projectnaam achter de /.

Build settings

Repository:	github.com/CedricMeeuwis/Herentals-Vuilbakken-FrontEnd
-------------	-------------------------------------------------------------------------------------------------------------------------------------

Base directory:	Not set
-----------------	---------

Build command:	ng build --prod
----------------	-----------------

Publish directory:	dist/SmartcityHerentals
--------------------	-------------------------

Deploy log visibility:	Logs are public
------------------------	-----------------

Builds:	Active
---------	--------

<https://connect-it.netlify.app/>