Thingslogic Docker Container

Description

Docker

Docker is a software container platform. With Docker you can create Containers, where your whole development environment is installed. That is what we have done for you. This Docker Image contains NodeRed, CrateDB and Grafana.

The things network

The things network is a community based Internet of Things Network, which allows IOT devices to connect to the internet without 3G or WIFI. The Used Technology is called LoRaWAN, a long range and low power radio frequency protocol.

NodeRed

NodeRed is a visual tool for wiring the Internet of Things. It can also be used for other applications to quickly assemble flows of services. NodeRed has multiple plugins for many different services, e.g. Database connection or HTTP Requests.

CrateDB

CrateDB is a distributed SQL database built on top of a NoSQL foundation. It combines the familiarity of SQL with the scalability and data flexibility of NoSQL.

Grafana

Grafana is an open source metric analytics & visualization suite. It is most commonly used for visualizing time series data for infrastructure and application analytics.

Software Requirements

Kitematik:

Windows:	https://download.docker.com/win/stable/DockerToolbox.exe
Mac OSX:	https://download.docker.com/mac/stable/DockerToolbox.pkg

Installation Steps

Steps only needed for Windows:

1. Open Kitematik
2. Open VirtualBox on Windows
3. You should see a Linux machine named "default"
4. Open the machine default by clicking twice on it, now a window should open
5. In the window you can insert some commands. Run the following command:

```
sudo sysctl -w vm.max_map_count=262144
```

General Steps for Windows and MacOSX:

Run the following steps in order to get the thingslogic stack running.

1. Open Kitematik and choose to create a new container (top left corner)
2. Then type "thingslogic-grafana" in the search bar and choose to create a new container of "thingslogic-grafana". It should automatically start to download the image.
3. Then type "thingslogic-nodered" in the search bar and choose to create a new container of "thingslogic-nodered". It should automatically start to download the image.
4. Then type "thingslogic-crate" in the search bar and choose to create a new container of "thingslogic-crate". It should automatically start to download the image.
5. If the containers are running, you have to make some settings in every container.
6. thinglogic-grafana
    a. choose the thingslogic-grafana container
    b. in the right column click on every volume and select enable volume, to allow the container to save some data on your drive.
    c. go to the settings tab and choose network
    d. set the port forward to port number 3000 and save your settings
7. thinglogic-nodered
    a. choose the thingslogic-nodered container

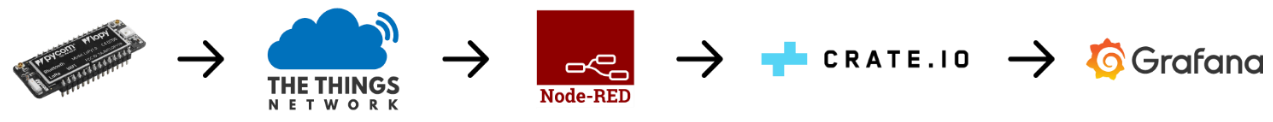
    b. in the right column click on every volume and select enable volume, to allow the container to save some data on your drive.
    c. go to the settings tab and choose network
    d. set the port forward to port number 1880 and save your settings

8. thinglogic-crate
    a. choose the thingslogic-crate container
    b. in the right column click on every volume and select enable volume, to allow the container to save some data on your drive.
    c. go to the settings tab and choose network
    d. set the port forward to port number 4200 and save your settings
9. Now you should be able to access the services via the given urls.

Access URLs

- NodeRed: http://{{ip}}:1880
- Grafana: http://{{ip}}:3000
- CrateDB: http://{{ip}}:4200

Sample Config



Sample code for the Pycom module

1. boot.py (Example Code)

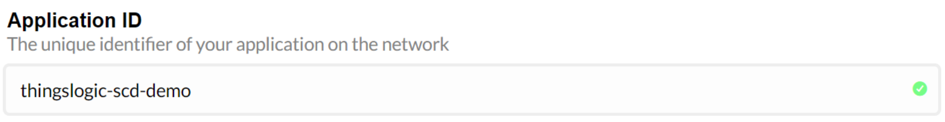
```python
from machine import UART
import os
uart = UART(0, 115200)
os.dupterm(uart)
```

2. main.py (Example Code)

```python
from network import LoRa
import socket
import time
import binascii
import machine
from machine import Pin
import ultrasonic
import struct

sensor1_trigPin = Pin("P23")
sensor1_echoPin = Pin("P22")

sensor1 = ultrasonic.Ultrasonic(sensor1_trigPin, sensor1_echoPin)

# Initialize LoRa in LORAWAN mode.
lora = LoRa(mode=LoRa.LORAWAN)

# create an OTAA authentication parameters
app_eui = binascii.unhexlify('XX XX XX XX XX XX XX XX'.replace(' ',''))
app_key = binascii.unhexlify('XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX'.replace(' ',''))

# join a network using OTAA (Over the Air Activation)
lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout=0)

# wait until the module has joined the network
while not lora.has_joined():
    time.sleep(2.5)
    print('Not joined yet ...')

# create a LoRa socket
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)

# set the LoRaWAN data rate
s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)

# Things to do every 30 seconds
while True:
    s.setblocking(True)

    # send distance
    distance1 = sensor1.distance_in_cm()
    print("Distance (Metric System)", distance1, "cm")
    distanceString = "d" + str(distance1)
    s.send(b'' + distanceString)

    s.setblocking(False)
    time.sleep(30)
```

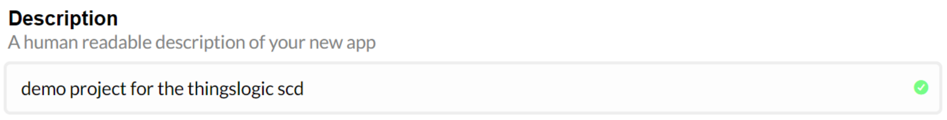
The things network

1. go to https://console.thethingsnetwork.org/applications and create a new application
   a. add an application ID (can be everything, but must be unique inside the things network



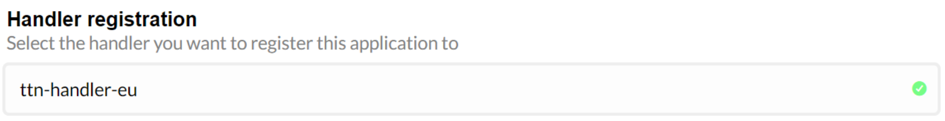
Application ID
The unique identifier of your application on the network
thingslogic-scd-demo

   b. add a description for the application



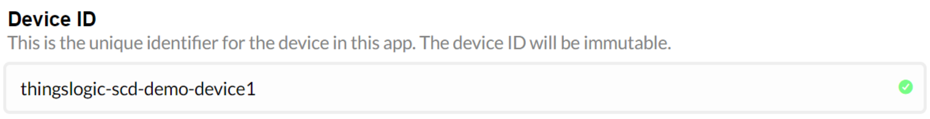
Description
A human readable description of your new app
demo project for the thingslogic scd

   c. choose an TTN handler



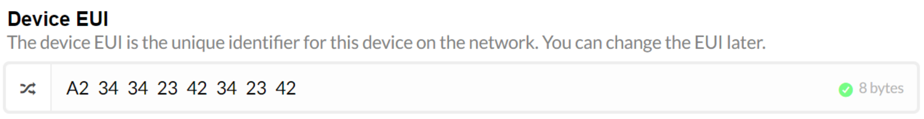
Handler registration
Select the handler you want to register this application to
ttn-handler-eu

2. go to devices and register a new device to the application
   a. set the device id



Device ID
This is the unique identifier for the device in this app. The device ID will be immutable.
thingslogic-scd-demo-device1

   b. set the device EUI -> every device has its own unique EUI (like a MAC address)

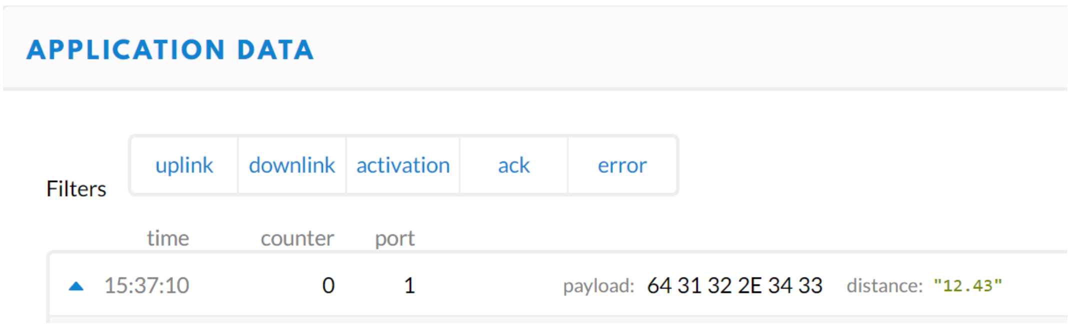


Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.
A2 34 34 23 42 34 23 42

3. go back to the application you have created just before and choose the tab payload formats. Here you can define decode, convert, validate and encode functions you want to use within your application.
    a. Sample decoder function:

```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var decoded = {};
  var result = "";

  for (var i = 0; i < bytes.length; i++){
    result += String.fromCharCode(parseInt(bytes[i]));
  }

  var first = result.substring(0,1);

  if (first === 'd'){
    if (result.substring(1,result.length) < 60){
      decoded.distance = result.substring(1,result.length);
    }
  }
}
```

4. to check if you receive data from the Pycom module, go to the tab devices > data and there should be something like this. Here you can see that your decoder works. Your decoder transforms a binary string ("64 31 32 2E 34 33") into an JSON Object ("{distance: "12.43"}")

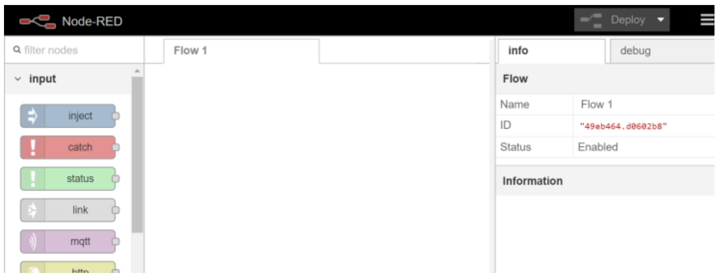

APPLICATION DATA

Filters: uplink | downlink | activation | ack | error

	time	counter	port	
▲	15:37:10	0	1	payload: 64 31 32 2E 34 33 distance: "12.43"

NodeRed

1. go to http://{{ip}}:1880 and you should see something like this

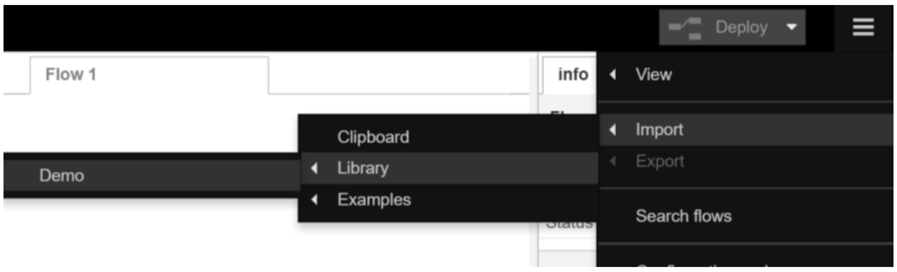


Flow

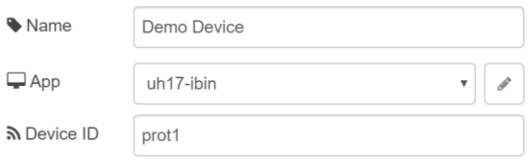
Name	Flow 1
ID	"49eb464.d0602b8"
Status	Enabled

Information

2. Open the menu on the right top corner and go to Import > Library > Demo. Then you see a sample which you can be added to your board.

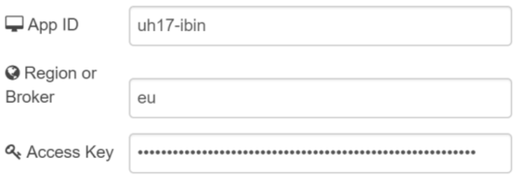


3. Now configure the TTN (the things network) connection settings inside all TNN nodes. Click on the TTN nodes, then a window pops up. Here you can insert your Device ID (thingslogic-scd-demo-device1).



Name: Demo Device
App: uh17-ibin
Device ID: prot1

4. Edit the TTN App setting using the edit button. Then you can insert your AppID (thingslogic-scd-demo), Region (eu) and Access Key. The whole data can be found at your the things network console.



App ID: uh17-ibin
Region or Broker: eu
Access Key: ••

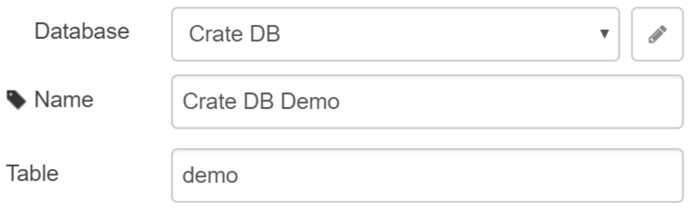
5. You can add every component you want and do something with your data. For example, use some function nodes.

6. There are two components you have to use if you want to save and visualize the received data. The first one is the CrateDB node and the second is a function before the CrateDB node. The two nodes have some requirements:

   a. Function node (before the CrateDB node)
      needs a return statement with the following structure. For every key inside the data object you need to create a table in the database (explanation later).

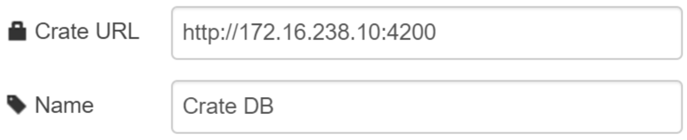
```
return {
  data: {
    ...
  }
}
```

b. CrateDB node
contains the basic database connection configuration, here I have to set the name of the table (you have to create the table in the database – shown in the next section) where the data should be stored.



Database	Crate DB
Name	Crate DB Demo
Table	demo

The Crate URL must be set to http://{{ip}}:4200. This IP is only a local virtual network inside the docker environment.

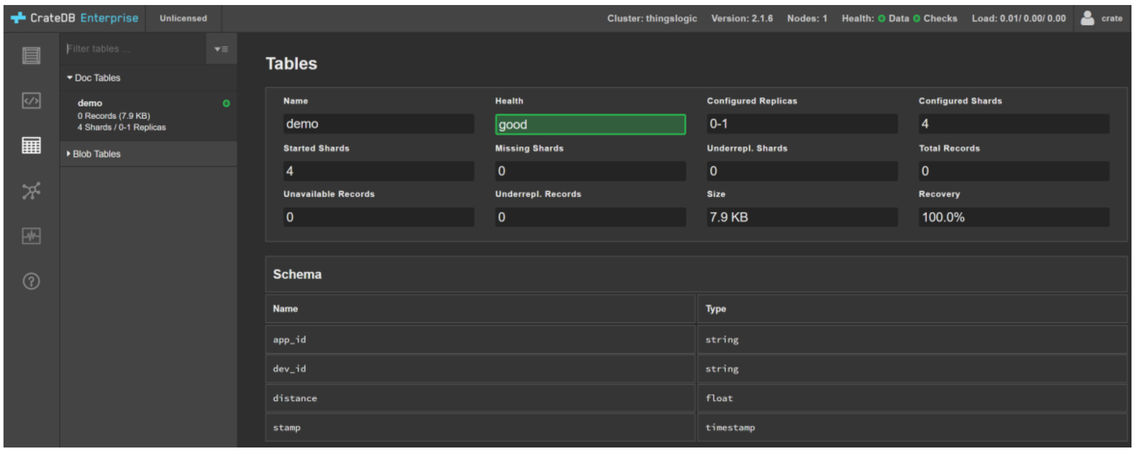


Crate URL	http://172.16.238.10:4200
Name	Crate DB

7. In order to start the nodes, need to press on the deploy button on the right top corner.

CrateDB

1. go to http://{{ip}}:4200 and you should see the CrateDB Dashboard
2. If you want to show all available tables, then you have to go to the table tab in the menu on the left side. By default, the demo table should be there.



Cluster: thingslogic Version: 2.1.6 Nodes: 1 Health: Data Checks Load: 0.01/ 0.00/ 0.00

demo
0 Records (7.9 KB)
4 Shards / 0-1 Replicas

Tables

Name	Health	Configured Replicas	Configured Shards
demo	good	0-1	4
Started Shards	**Missing Shards**	**Underrepl. Shards**	**Total Records**
4	0	0	0
Unavailable Records	**Underrepl. Records**	**Size**	**Recovery**
0	0	7.9 KB	100.0%

Schema

Name	Type
app_id	string
dev_id	string
distance	float
stamp	timestamp

3. If you want to create a new table for your application you can navigate to the console tab. Then you can run can insert a query (all queries can be seen here: https://crate.io/docs/crate/reference/sql/index.html)

You can create the demo table using the following command:

```
create table demo (
        distance float,
        stamp timestamp,
        dev_id string,
        app_id string
);
```

4. To show the data stored inside the demo table use the following command.

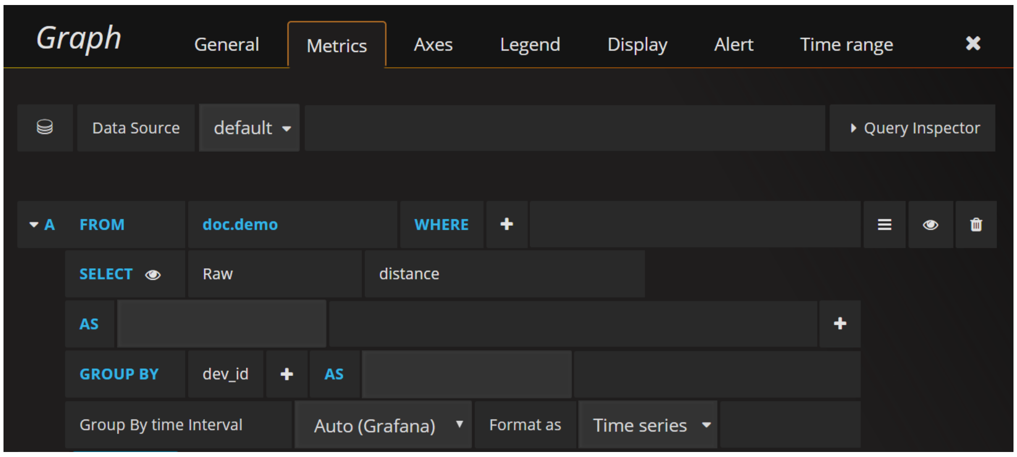
```
select * from demo
```

With this command, you can delete all data, which was stored in the demo table.

```
delete from demo where true
```

Grafana

1. go to http://{{ip}}:3000
2. The default login credentials are
   - Username: admin
   - Password: admin
3. You have to create a new datasource (Data from the CrateDB) the following way:
   a. Open the menu on the top left corner and navigate to datasource
   b. Choose to add a new datasource
   c. For the new datasource choose the following settings:
      - Type: Crate
      - URL: http://{{ip}}:4200 This IP is only a local virtual network inside the local docker environment
      - Access: proxy
      - Schema: doc
      - Table: <the name of your table> = demo
      - Timecolumn: <name of the column which contains the time> = stamp
4. Finally, you want to create a chart, to visualize your data.
   a. Open the menu on the top left corner and navigate to dashboard > new
   b. Choose the type of a new dashboard. In this case a graph
   c. To select the column, which you want to visualize click on the panel title and then select edit. There should popup a new window section.
   d. In the new window section go to the tab metrics. Here you can edit the select statement. For example:

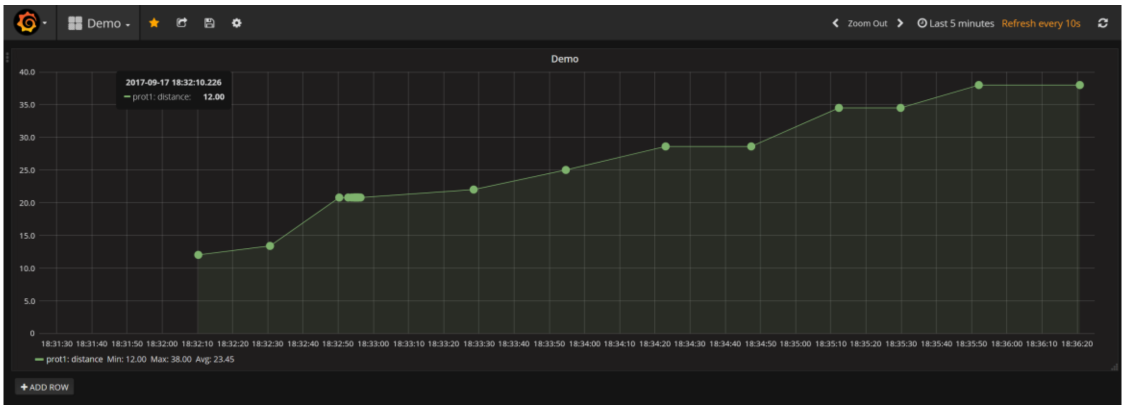

Graph | General | Metrics | Axes | Legend | Display | Alert | Time range

Data Source: default | Query Inspector

A
FROM doc.demo WHERE +
SELECT Raw distance
AS +
GROUP BY dev_id + AS
Group By time Interval: Auto (Grafana) | Format as: Time series

   e. Save the recently created Dashboard

5. Now you should see the Grafana Dashboard including a sample Chart. (there might be no data at the beginning – this is only a sample how your chart could look like)



Start building amazing IoT Applications