4 IoT Monitoring System

Table of Contents

4.1 IoT Monitoring System

We are going to implement the following or a similar system to fig 1 with your choice of solutions for TTN Bridge (subscriber) and InfluxDB (database).

We do this by subscribing to sensor data from schools ELSYS sensors (LoRa Nodes) via TTN and MQTT.

Then our TTN Bridge will save data to an IoT database (InfluxDB) for further visualization in Grafana (if needed). InfluxDB v2.x can take care of visualization as well.

Lastly we are going to dockerize our whole IoT solution stack so it is easy to develop, distribute, maintain and run it on any system. Preferable we support custom configuration of the stack so we do not need to rebuild one or more images for one variable change.

Report for the whole lab
Show that you have a working implementation of the full system in figure 1. Hand in your source code, some simple description of your solution and a screen recording over mission accomplished.
Put your docker images on your www-account or in a repository and attach the link to them.

The whole solution you produce should be able to run with docker-compose by the teacher with a minimal effort to configure it. At least the sensor data collecting part.

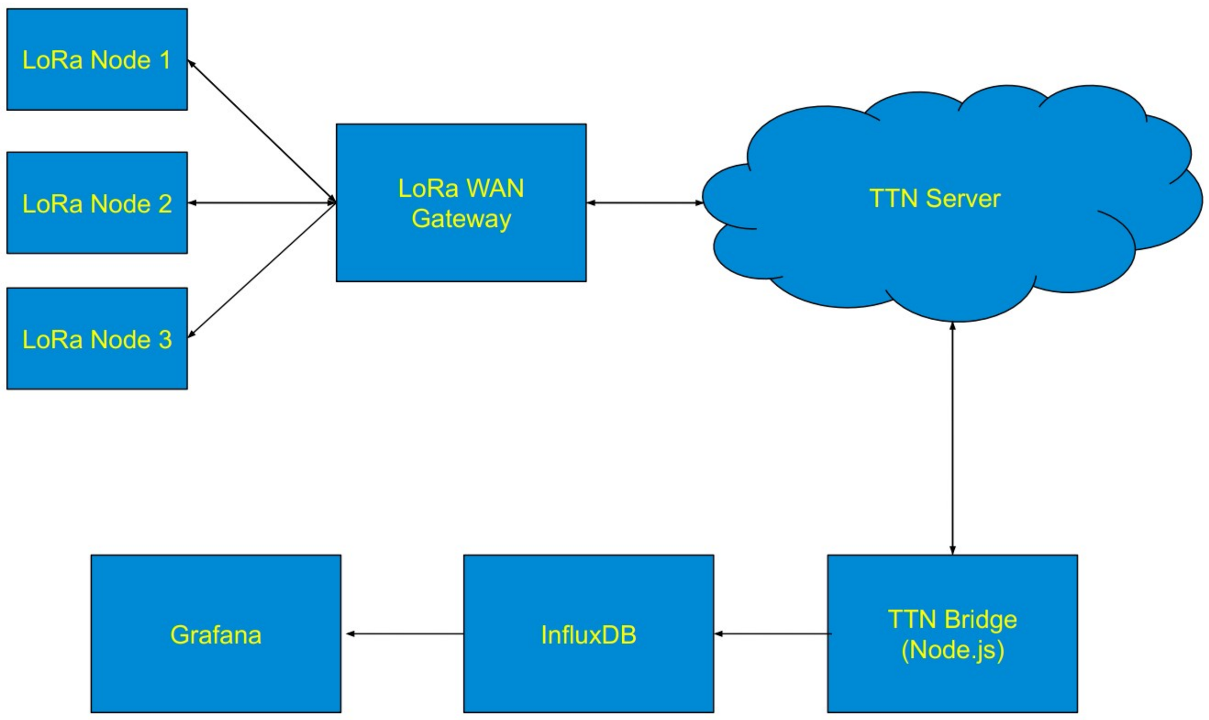


Fig 1. IoT monitoring system

The task is pretty free to implement in the way you want. It is quite large but when you have done it you have a template you can use for almost any monitoring solution. One way to solve the lab is to follow the IoT Monitoring - The Things Network article at: https://github.com/ucwlabs/iot-monitoring-ttn, which is bundled with the lab. One can run everything on a PC or on a Raspberry Pie (may be available at school), or in a virtual machine. Everything should run in one or more Docker containers which is the approach for us in the lab.

Inspiration from Andreas Spiess - #352 Raspberry Pi4 Home Automation Server (incl. Docker, OpenHAB, HASSIO, NextCloud): https://www.youtube.com/watch?v=KJRMjUzlHI8

My own example solutions with Python and C# and the latest version of InfluxDB is Dockerized. See the Docker section for how to install and use Docker with these containers. Also see Learn for more links to books, resources etc. about Docker.

4.1.1 TTN Bridge Task 1 (Subscribe to LoRa Nodes)

For the TTN Bridge you have a number of solutions to consider:

- You can extend your C# MQTT program from lab 3.2.2 (this is what I choose for my solution).
- Or you can use an another MQTT programming language solution supported by the TTN Data API at:
  - https://www.thethingsnetwork.org/docs/applications/apis.html > MQTT
- Or you can use a TTN SDK & Library for the TTN Data API at: https://www.thethingsnetwork.org/docs/applications/sdks.html. For example:

  - Node-RED
  - Python
  - Node.js (discontinued)
- If you administrate the LoRa node sensors in TTN one can also use the HTTP integration in TTN and send data to a web service (receiver) you create by yourself for later posting to a database. But this is not possible now.

The above (and probably some more) ways can be used in the task. eg. subscribe to the LoRa nodes publishing of sensor data for further writing to your storage solution.

Node-RED
Node-RED is a programming tool for wiring together hardware devices, APIs and online services. It provides a browser-based editor that wire together flows using nodes in the palette.
It require some additional NPM-packages if you want to use it with The Things Network. It also needs some learning to understand it. Refer to the appendix for how to install and run Node-RED in a Docker container. Note that Node-RED does not support InfluxDB v2.x yet (at the time of this is written).

Make sure you use the latest docker Node-RED!: https://github.com//node-red/node-red-docker and the latest-minimal version works ok.

Python
I have tested Python with good results in solutions functioning as a TTN Bridge.

Node.js
Node.js support in TTN is however discontinued according to:
https://github.com/TheThingsNetwork/node-app-sdk

ELSYS
The LoRa nodes sensor documentation is found at ELSYS website: https://www.elsys.se/
We got 2 pieces of ERS CO2 and 2 pieces of ELT-2 sensors. They are publishing their sensor data on the following topics at TTN.

ERS CO2
- campusborlangeelsys/devices/a81758fffe045fa4/up
- campusborlangeelsys/devices/a81758fffe045fa5/up

ELT-2
- campusborlangeelsys/devices/a81758fffe0459fd/up
- campusborlangeelsys/devices/a81758fffe0459fe/up

Javascript code to decode the ELSYS payload (for example in the TTN Console or in Node.js) is found here: https://www.elsys.se/en/elsys-payload/

Some C# libraries to decode the ELSYS payload if using C#:
- https://github.com/sjkp/Elsys.Decoder

    - This is what I used. The library got a non-critical bug which I posted an issue about: https://github.com/sjkp/Elsys.Decoder/issues/1
- https://github.com/SeppPenner/SensorsPayloadDecoder

The LoRa nodes Application ID is: **campusborlangeelsys**. The **Access Key** which also is needed to subscribe to the IoT data at TTN **is given in Learn**. Please keep it secret.

When you have subscribed to the TTN ELSYS topics, can receive data and parse the Elsys sensor data this task is done.

4.1.2 TTN Bridge Task 2 (Write to DB and Visualization)

What we need to do now is to set up the storage system in the way you want according the solution or guide you have selected to follow, i.e. write sensor measurements to a data storage solution as InfluxDB (you are free to use any other storage solution).

The next step after this is to set up visualization and monitoring of your nodes data in real-time and to be able to get historical data dumps from the database.
InfluxDB
Get InfluxDB in any distribution form from: https://portal.influxdata.com/downloads/

In case of using C# for storing data into InfluxDB there are several resources available on Github: https://github.com/search?q=influxdb&type=Repositories. The most recently updated one seems to be the reference InfluxDB.Client: https://github.com/influxdata/influxdb-client-csharp.

I modified the attached platform example code from here: https://github.com/influxdata/influxdb-client-csharp/blob/master/Examples/PlatformExample.cs to suit my needs for putting data into InfluxDB 2.x.

One important thing to note about the influxdata git library above is that you must use the C# code version that matches your InfluxDB version. Also if you want to run InfluxDB v2 in Windows you currently need to run it in a Docker container: https://hub.docker.com/editions/community/docker-ce-desktop-windows and with the InfluxDB 2.x alpha version.
Refer to the appendix and other Learn material for how to install and run InfluxDB in a Docker container.

The good thing with using InfluxDB v2.x is that it have a HTTP GUI web application built into it itself which the InfluxDB v1.x does not have. This simplifies administration and installation. Also it has builtin graphs and dashboards, so the need for Grafana is very questionable in this case. Reference: https://www.influxdata.com/products/influxdb-overview/influxdb-2-0/

In both cases however you need to use login or organization id, bucket names or Tokens etc. The Token is found in the (Load) Data > Tokens page for InfluxDB v2.

Grafana

Get Grafana from: https://grafana.com/get. Refer to the appendix for how to install and run Grafana in a Docker container.
If you got networking problems in Docker you can also use a standalone/native installer at: https://grafana.com/grafana/download.

Note that if you have installed InfluxDB v2.x and you by some reason want to install and run Grafana you need the Grafana Flux (InfluxDB) [BETA] plugin at: https://grafana.com/grafana/plugins/grafana-influxdb-flux-datasource/installation which connects to: http://localhost:9999 or port 8086. Otherwise it will not work.

Also note that InfluxDB v2.x use a new query language named Flux: https://www.influxdata.com/products/flux/. Some Flux query examples: https://thenewstack.io/how-influxdb-and-flux-gather-meaningful-insights-from-time-series-data/

Docker compose
To get networking to work between Docker containers that depends on each other may be a problem. The easiest solution for this is to use the docker compose command https://docs.docker.com/compose/ together with a docker-compose.yml configuration file.
A sample YML-file for this scenario which you can edit is attached.
Visual Code have a Docker MS extension that is helpful for Docker YML editing and other common Docker tasks.

4.1.3 Dockerize your whole IoT monitoring solution

Regardless of what you have chosen regarding programming languages and software you should be able to Dockerize your solution and run it with docker-compose.

Example from my C# app and InfluxDB 2.0 RC.

1. Dockerfile file located in VS project folder for a C# app (no build container)

```
# https://docs.microsoft.com/en-us/dotnet/core/docker/build-container
FROM mcr.microsoft.com/dotnet/core/runtime:3.1
# build app before locally since RUN dotnet ... publish executables in image
# https://stackoverflow.com/questions/42346498/dotnet-aspnetcore-docker-build-
fails-with-a-145-error-code
# RUN dotnet publish ConsoleAppLoRa.csproj --configuration Release
# DO NOT FORGET ABOVE BUILD AND PUBLISH STEP BETWEEN CHANGES!
COPY bin/Release/netcoreapp3.1/publish/ App/
WORKDIR /App
ENTRYPOINT ["dotnet", "ConsoleAppLoRa.dll"]
```

2. Dockerfile file located in VS solution folder for a C# solution (with build container)

```
# https://hub.docker.com/_/microsoft-dotnet-core
# https://hub.docker.com/_/microsoft-dotnet-core-samples
FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
WORKDIR /source
# copy all project folders and sub-files from .SLN root folder
COPY . .
# and restore as distinct layers
RUN dotnet restore
```

```
# publish app and libraries, point out ENTRYPOINT
RUN dotnet publish ConsoleAppLoRa -c release -o /app --no-restore

# final stage/image
FROM mcr.microsoft.com/dotnet/core/runtime:3.1
WORKDIR /app
COPY --from=build /app .
ENTRYPOINT ["dotnet", "ConsoleAppLoRa.dll"]
```

Build and Run

```
# Build image
# docker build --no-cache -t lora-ttn-image -f Dockerfile .

# Run interactive and remove container after stop
# docker run -it --rm --name lora-ttn-app lora-ttn-image

# As above run but also use a volume: https://docs.docker.com/storage/volumes/
# docker run -v ////c/vm/conf:/root/conf -it --rm --name lora-ttn-app lora-ttn-
image
```

Repositories – local

```
# Publish on other host - How to copy Docker images from one host to another
without using a repository
# https://stackoverflow.com/questions/23935141/how-to-copy-docker-images-from-one-
host-to-another-without-using-a-repository
# docker save -o c:/tmp/lora-ttn.tar lora-ttn:latest
# docker load -i c:/tmp/lora-ttn.tar
```

Repositories – Docker Hub

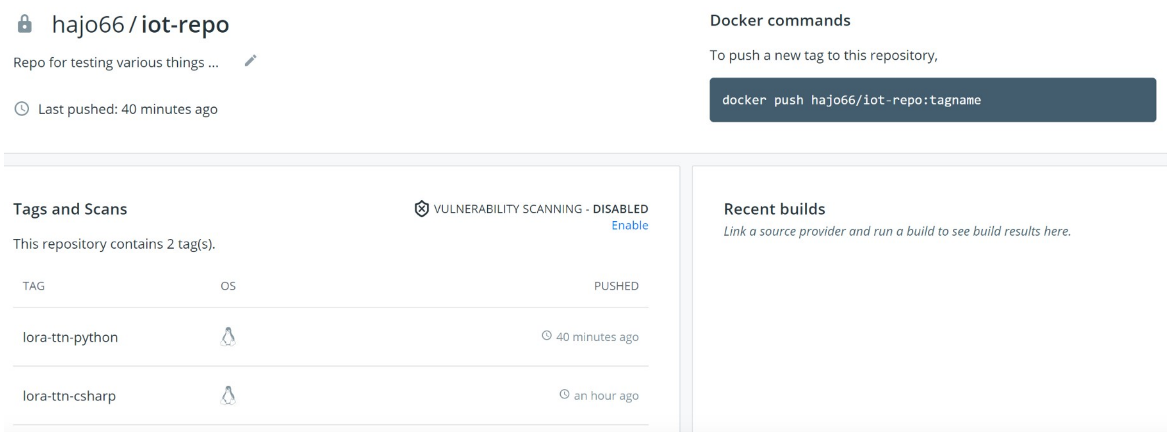
You get one private repository for free with your Docker user account:

- https://docs.docker.com/docker-hub/repos/
- https://hub.docker.com/repository/create

When you have a repository you can push built images from Docker Desktop and automatic build pipelines as Github, Bitbucket and possibly Azure DevOps to your image repo.

Example build, push and pull an image:

```
docker build -t hajo66/iot-repo:lora-ttn-csharp -f Dockerfile .
docker push hajo66/iot-repo:lora-ttn-csharp
docker pull hajo66/iot-repo:lora-ttn-csharp
```


hajo66 / iot-repo
Repo for testing various things ...
Last pushed: 40 minutes ago

Docker commands
To push a new tag to this repository,

```
docker push hajo66/iot-repo:tagname
```

Tags and Scans
VULNERABILITY SCANNING - DISABLED
Enable
This repository contains 2 tag(s).

TAG	OS	PUSHED
lora-ttn-python		40 minutes ago
lora-ttn-csharp		an hour ago

Recent builds
Link a source provider and run a build to see build results here.

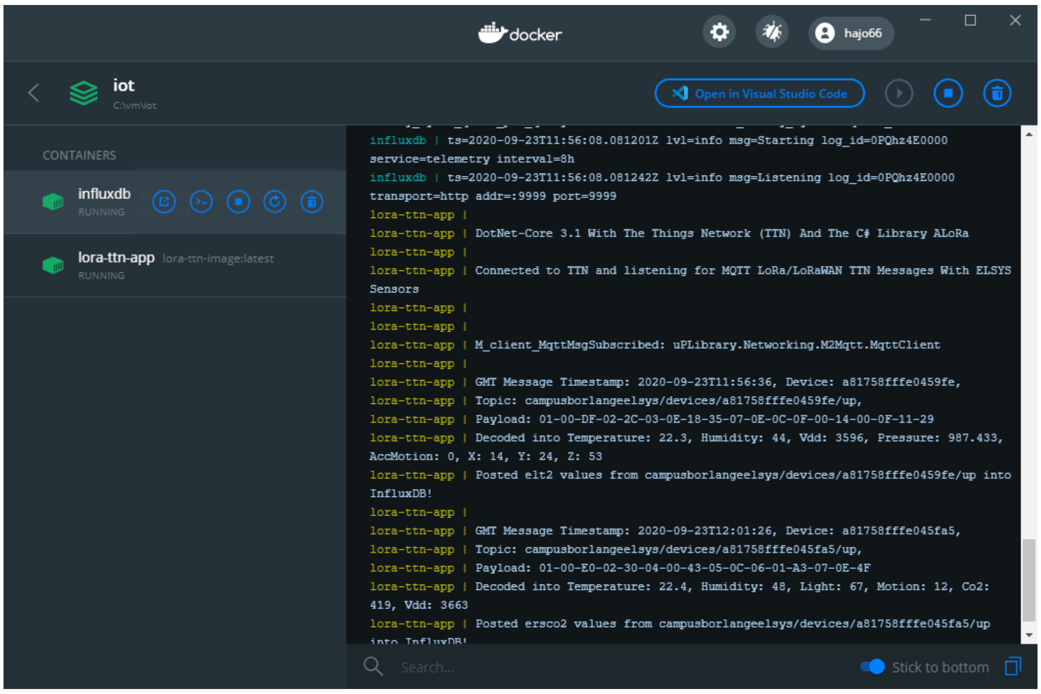
Fig 2. Docker Hub repository.

InfluxDB 2.0 RC or final?

Already built by InfluxDB developers so it will be downloaded when running docker-compose. Check the image tag '`2.0.x-yz`' so it is the latest InfluxDB image.

Docker-compose.yml file for the docker-compose command

```
# docker-compose -p "IoTMonitoringSystem" up
version: '3.8'
services:
  influxdb:
    image: 'quay.io/influxdb/influxdb:2.0.0-rc'
    container_name: influxdb
    networks:
      - iot_default
    ports:
      - '8086:8086'
    volumes:
      - ////c/vm/iot/influxdb:/root/.influxdbv2/
    restart: unless-stopped
  # C# app which get IoT-data from TTN and store it in influxdb
  lora-ttn:
    image: lora-ttn-image:latest
    container_name: lora-ttn-app
    networks:
      - iot_default
    volumes:
      - ////c/vm/conf:/vm/conf
    depends_on:
      - influxdb
    restart: unless-stopped
networks:
  iot_default:
    external: true
```



```
influxdb | ts=2020-09-23T11:56:08.081201Z lvl=info msg=Starting log_id=0PQhz4E0000
service=telemetry interval=8h
influxdb | ts=2020-09-23T11:56:08.081242Z lvl=info msg=Listening log_id=0PQhz4E0000
transport=http addr=:9999 port=9999
lora-ttn-app |
lora-ttn-app | DotNet-Core 3.1 With The Things Network (TTN) And The C# Library ALoRa
lora-ttn-app |
lora-ttn-app | Connected to TTN and listening for MQTT LoRa/LoRaWAN TTN Messages With ELSYS
Sensors
lora-ttn-app |
lora-ttn-app |
lora-ttn-app | M_client_MqttMsgSubscribed: uPLibrary.Networking.M2Mqtt.MqttClient
lora-ttn-app |
lora-ttn-app | GMT Message Timestamp: 2020-09-23T11:56:36, Device: a81758fffe0459fe,
lora-ttn-app | Topic: campusborlangeelsys/devices/a81758fffe0459fe/up,
lora-ttn-app | Payload: 01-00-DF-02-2C-03-0E-18-35-07-0E-0C-0F-00-14-00-0F-11-29
lora-ttn-app | Decoded into Temperature: 22.3, Humidity: 44, Vdd: 3596, Pressure: 987.433,
AccMotion: 0, X: 14, Y: 24, Z: 53
lora-ttn-app | Posted elt2 values from campusborlangeelsys/devices/a81758fffe0459fe/up into
InfluxDB!
lora-ttn-app |
lora-ttn-app | GMT Message Timestamp: 2020-09-23T12:01:26, Device: a81758fffe045fa5,
lora-ttn-app | Topic: campusborlangeelsys/devices/a81758fffe045fa5/up,
lora-ttn-app | Payload: 01-00-E0-02-30-04-00-43-05-0C-06-01-A3-07-0E-4F
lora-ttn-app | Decoded into Temperature: 22.4, Humidity: 48, Light: 67, Motion: 12, Co2:
419, Vdd: 3663
lora-ttn-app | Posted ersco2 values from campusborlangeelsys/devices/a81758fffe045fa5/up
into InfluxDB!
```

Fig 3. Docker Desktop > Dashboard. Version 2.4.0.0 got much more UI controls.

Adding App config or Environment variables

Get your custom Application Configuration into your Docker Containers. In other words you should not be forced to recompile your solution for a simple change of one variable in your app. References:

- https://dantehranian.wordpress.com/2015/03/25/how-should-i-get-application-configuration-into-my-docker-containers/
- https://vsupalov.com/docker-arg-env-variable-guide/

Variables in C#, Python or other language you choosen which may change is:

TTN

- string TTN_APP_ID = "camp...";
- string TTN_ACCESS_KEY = "ttn-account-...;

InfluxDB

- private static readonly char[] TOKEN = "SmxY...".ToCharArray();
- private const string INFLUXDB_URL = "http://localhost:8086";
- private const string BUCKET_NAME = "elsys";
- private const string ORG_ID = "campus_borlange";

C# Nuget package:
https://www.nuget.org/packages/System.Configuration.ConfigurationManager/

To get your custom app configuration in C# from the Docker Volume:
https://docs.docker.com/storage/volumes/ you need a method like below. It will return the setting value from the App.config file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="ConnString" value="Server=@DB_SERVER@;Initial Catalog=@DB_NAME@;User
Id=@DB_USER_NAME@;Password=@DB_USER_PWD@"/>
    <add key="OracleConnString" value="data source=ORACLE;user
id=scott;password=tiger"/>
  </appSettings>
</configuration>
```

when called like this:

```
string INFLUXDB_URL = GetAppSettingValue("INFLUXDB_URL");

/// <summary>
/// Use this method for custom App.config files:
https://stackoverflow.com/questions/10656077/what-is-wrong-with-my-app-config-file
/// </summary>
/// <param name="appSettingKey"></param>
/// <returns>Appsetting value</returns>
public static string GetAppSettingValue(string appSettingKey)
{
    try
    {
        ExeConfigurationFileMap fileMap = new ExeConfigurationFileMap();
        fileMap.ExeConfigFilename = "/vm/conf/App.config";
        var configuration =
ConfigurationManager.OpenMappedExeConfiguration(fileMap,
ConfigurationUserLevel.None);
```

```
        string value = configuration.AppSettings.Settings[appSettingKey].Value;

        //string value = ConfigurationManager.AppSettings[appSettingKey];
        if (string.IsNullOrEmpty(value))
        {
            string message = $"Can not find value for appSetting key:
'{appSettingKey}'.";
            throw new ConfigurationErrorsException(message);
        }
        return value;
    }
    catch (Exception e)
    {
        Console.WriteLine($"The appSettingKey: {appSettingKey} could not be
read:");
        Console.WriteLine(e.Message);
        return "";
    }
}
```

For Python it is easier with environment variables. In your Python script set:

```
TTN_APP_ID = os.environ.get('TTN_APP_ID')
TTN_ACCESS_KEY = os.environ.get('TTN_ACCESS_KEY')
and so on …
```

When running the container add the -e option to set the env variable -e.

```
-e, --env list                    Set environment variables
--env-file list                   Read in a file of environment variables

# docker run -e TTN_APP_ID=campusborlangeelsys TTN_ACCESS_KEY=ttn-account-
v2.seLxyz… -it --name lora-ttn-app-python lora-ttn-python

Using a env-file it MUST NOT contain any white space at all. Fileformat:
TTN_APP_ID=campus...
TTN_ACCESS_KEY=ttn-account-…
and so on …
```

Run the image with:

```
docker run -it --env-file=env_ttn_influxdb.txt --name lora-ttn-app-python lora-
ttn-python
```

With docker-compose you can add the list of environment variables file by adding the **env_file** or the **environment** option into the docker-compose.yml file.

```
# docker-compose -p "IoTMonitoringSystem" up
version: '3.8'
services:
  lora-ttn:
    image: lora-ttn-python:latest
    container_name: lora-ttn-python
    networks:
        - iot_default
    env_file: env_ttn_influxdb.txt
    # environment:
    #     TTN_APP_ID=campus…
    #     TTN_ACCESS_KEY=ttn-account-…
    # if docker cannot attach to the console when using docker-compose
    # add stdin_open and tty
    stdin_open: true
    tty: true
    restart: unless-stopped
networks:
    iot_default:
        external: true
```

NOTE!
When using App config or Environment variables I strongly suggest that you write out the parsed or used variables in the app as soon as possible to the console.
Otherwise you do not know if the container is using the custom configuration variables or are using default values as variables or are using any values as variables at all.

4.1.4 Help, references and resources

- InfluxDB 2.0 alpha - Get Started!: https://www.youtube.com/watch?v=pweNlzHrEO4
- #255 Node-Red, InfluxDB, and Grafana Tutorial on a Raspberry Pi: https://www.youtube.com/watch?v=JdV4x925au0
- Update the RPi distro to Buster: https://pimylifeup.com/upgrade-raspbian-stretch-to-raspbian-buster/
- #295 RaspberryPi Server based on Docker, with VPN remote access, Dropbox backup, Influx, Grafana, etc.: https://www.youtube.com/watch?v=a6mjt8tWUws
- InfluxDb with Grafana Tutorial: https://www.youtube.com/watch?v=DmIWgkawcw4&list=PLoVvAgF6geYMb029jpxqMuz5dRDtO0ydM
- To play with queries in Grafana goto: https://play.grafana.org. One can also setup a test database in settings.
- Node-RED: https://www.automatiserar.se/guide-hemautomation-med-node-red/

Docker

- Docker Cheat Sheet: https://github.com/wsargent/docker-cheat-sheet
- Top 15 Docker Commands You Should Know: https://medium.com/edureka/docker-commands-29f7551498a8
- Understanding and Managing Docker Container Volumes: https://www.ionos.com/community/server-cloud-infrastructure/docker/understanding-and-managing-docker-container-volumes/
- Docker GUI comparisons
    - https://medium.com/@dockstation/a-brief-comparison-of-gui-docker-tools-46bd6a24ae31
    - https://www.starwindsoftware.com/blog/managing-windows-containers-with-graphical-user-interfaces

4.2 Docker and Grafana appendix (som parts are outdated)

Short manual for how to install Docker and run various containers.

4.2.1 Docker-Compose, DockStation and Kitematic

When you have installed Docker Desktop for your OS: https://docs.docker.com/docker-for-windows/, continue to install a GUI Docker tool.

First I used **Kitematic**, and it have a link in the Docker Desktop menu.
When installed create/use the attached file docker-compose.yml which I modified from: https://community.influxdata.com/t/where-is-the-docker-data-stored/10108

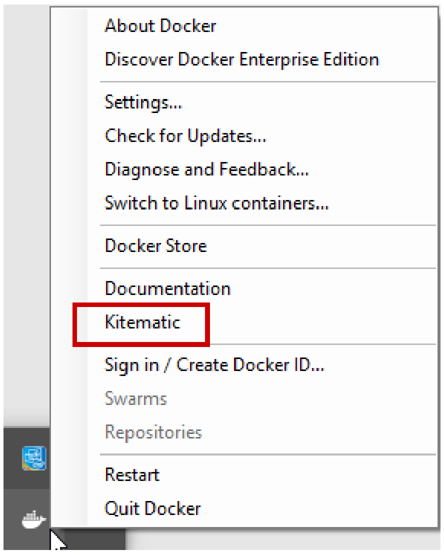


Fig 4. Kitematic.

DockStation https://dockstation.io/ is another GUI Docker tool which manage compose YML-files and projects much better than Kitematic.

Click in the DOCKER CLI button in lower left corner in Kitematic. In the same folder as you have the docker-compose.yml run the command:

```
docker-compose up -d
```

With DockStation you just have to create a new Project and point to the YML-file.

Now the container or containers should be downloaded and started. Here after you should be able to use Kitematic or DockStation to start and stop the containers, or run the CLI command: `docker-compose start or stop`

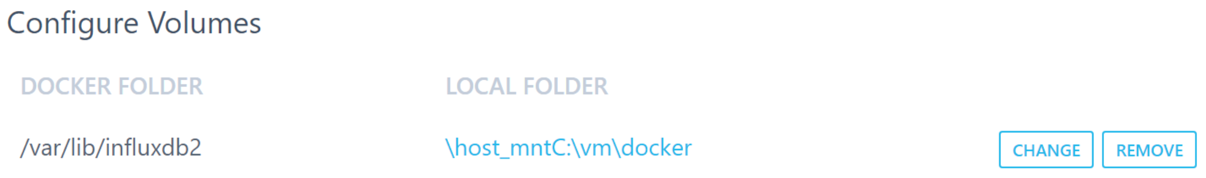
Only run the docker-compose down command if you want to remove the containers.

To update the containers if the publisher have released new ones run the CLI comand:

```
docker-compose pull
```

Refer to the documentation: https://docs.docker.com/compose/ for more commands.

At the moment I could not get volume sharing to work with Grafana, but with InfluxDB you should be able to do something like this in the Volumes tab.

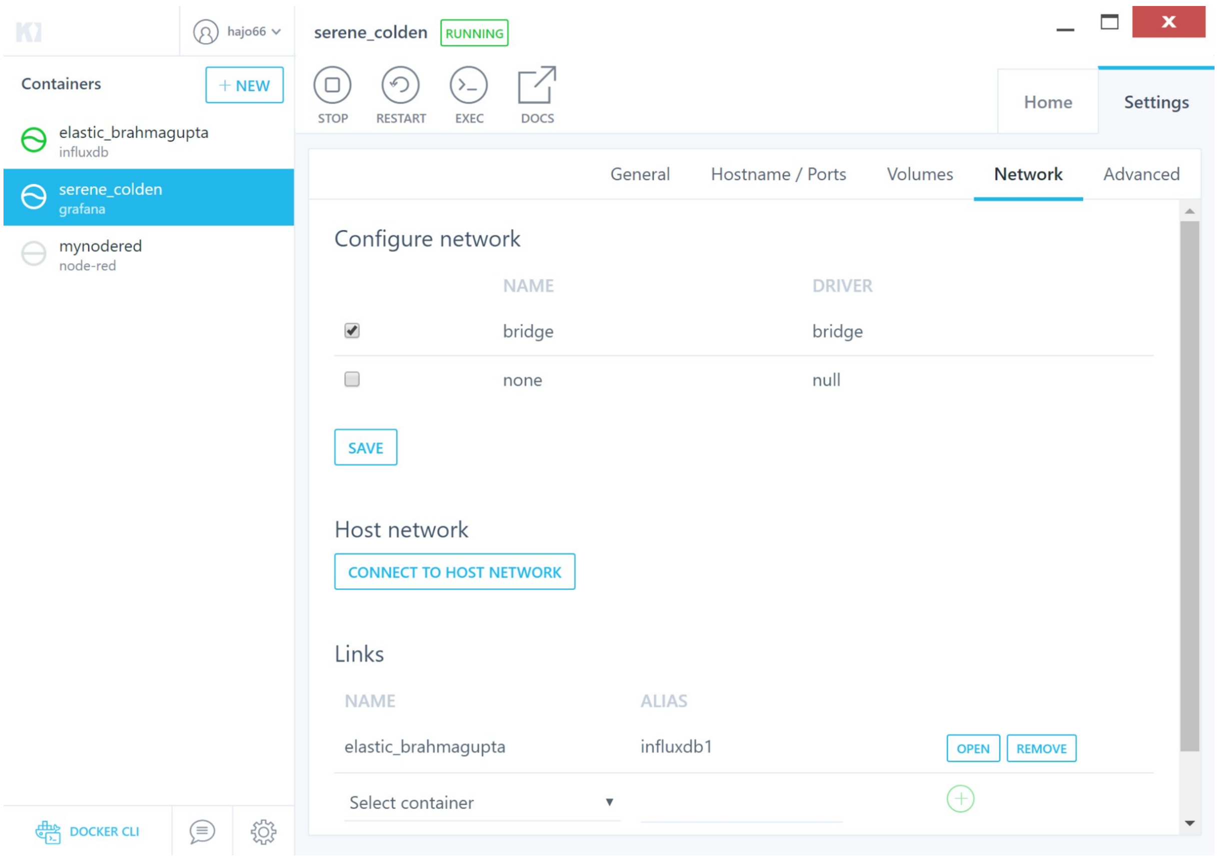

Configure Volumes

DOCKER FOLDER	LOCAL FOLDER	
/var/lib/influxdb2	\host_mntC:\vm\docker	CHANGE REMOVE

Fig 4. Persistent storage.

Using volumes the data you create should be persistent if you create a new volume later on. Otherwise you have to export and import data.

Another way of running containers together is with linking. Refer to the Network tab in Kitematic for this. See fig 4.



	NAME	DRIVER
✓	bridge	bridge
	none	null

NAME	ALIAS
elastic_brahmagupta	influxdb1

Fig 5. Kitematic Network tab.

4.2.2 Containers and more CLI commands etc.

If you got problems with docker-compose or the GUI, need to log into a shell in the container or just want to run one container and browse the container I have documented some of it here.

InfluxDB
https://v2.docs.influxdata.com/v2.0/get-started/
Get/dwl the container with the DOCKER CLI shell, run it and create a volume on local drive:
docker run -p 9999:9999 quay.io/influxdb/influxdb:2.0.0-alpha --reporting-disabled
Access InfluxDB via browser: http://localhost:9999

Get a shell in the container id / container name:
PS C:\> docker exec -it influxdb /bin/bash
root@d57ba271133b:/#

Grafana

https://grafana.com/docs/installation/docker/
Get/dwl the container with the DOCKER CLI shell:
docker run -d -p 3000:3000 grafana/grafana
Access Grafana via browser: http://localhost:3000

If using **old** v1.x InfluxDB.
https://community.influxdata.com/t/cannot-connect-to-influx-datasource-from-grafana/8048
As Grafana runs as a container inside of the Docker networking stack, Grafana should be configured to speak to InfluxDB using the: http://influx:8086 (when using “Access: Server”) or http://localhost:8086 when using “Access: Client”.

Node-RED (does not support InfluxDB v2.x yet)
https://nodered.org/docs/getting-started/docker
Get/dwl the container in a shell:
docker run -it -p 1880:1880 --name mynodered nodered/node-red
Access Node-RED via browser: http://localhost:1880

Needs some NPM packages to work, see configure Node-RED for TTN:
https://www.thethingsnetwork.org/docs/applications/nodered/quick-start.html

Installing NPMs: https://github.com//node-red/node-red-docker

Get a shell in the container id / container name:
PS C:\> docker exec -it mynodered /bin/bash

Install the TTN npm and InfluxDB packages. Use sudo if needed.

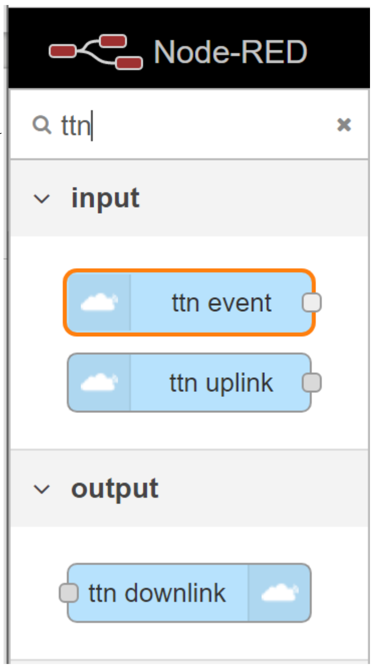


Fig 6. Node-RED TTN components.

bash-4.4$ npm install node-red-contrib-ttn
https://flows.nodered.org/node/node-red-contrib-influxdb
bash-4.4$ npm install node-red-contrib-influxdb

Howto get stopped and running containers in a shell
PS C:\> docker ps --filter "status=exited"
PS C:\> docker ps

```
CONTAINER ID        IMAGE                                COMMAND                  CREATED
STATUS              PORTS                  NAMES
5d6099dcb1a8        nodered/node-red                     "npm start -- --user…"   12 minutes ago
Up 28 seconds       0.0.0.0:1880->1880/tcp   mynodered
b1a3c4beed2b        grafana/grafana                      "/run.sh"                15 minutes ago
Up 29 seconds       0.0.0.0:3000->3000/tcp   serene_colden
ed944147032c        quay.io/influxdb/influxdb:2.0.0-alpha  "/entrypoint.sh infl…"   21 minutes ago
Up 29 seconds       0.0.0.0:9999->9999/tcp   elastic_brahmagupta
```

Start all containers with their IDs or names (mynodered, serene_colden, elastic_brahmagupta)

```
PS C:\> docker start ed944147032c b1a3c4beed2b 5d6099dcb1a8
```

Stop all containers with their IDs or names

```
PS C:\>docker stop mynodered serene_colden elastic_brahmagupta
```

Using the default bridge network with linking: https://docs.docker.com/network/network-tutorial-standalone/

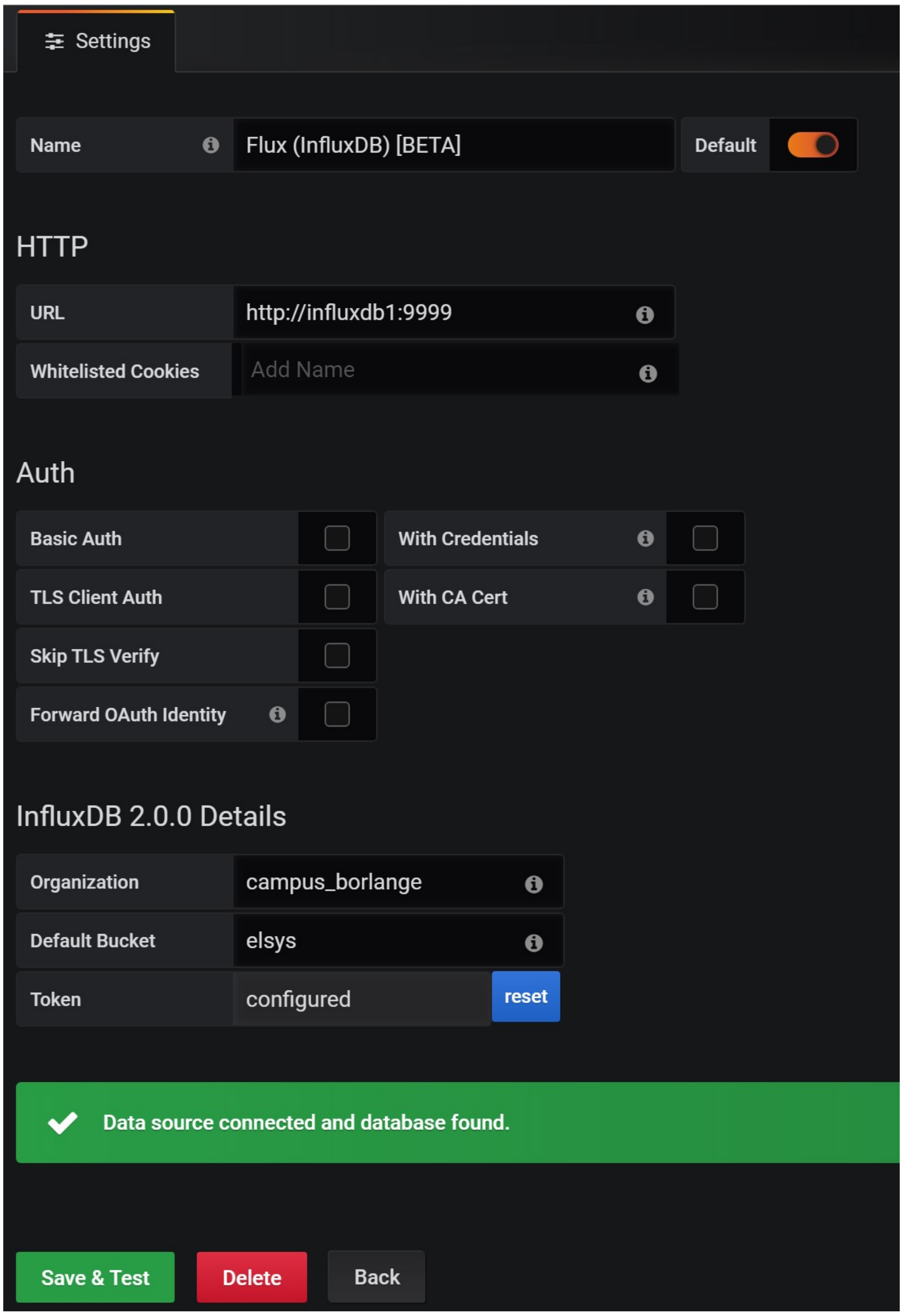
4.2.3 Working settings in Grafana against InfluxDB v2.x

Get a shell in the container id / container name:

```
PS C:\> docker exec -it grafana /bin/bash
```

Install the Grafana Flux (InfluxDB) [BETA] plugin at:
https://grafana.com/grafana/plugins/grafana-influxdb-flux-datasource/installation

```
bash-5.0$ grafana-cli plugins install grafana-influxdb-flux-datasource
installing grafana-influxdb-flux-datasource @ 5.4.0
from: https://grafana.com/api/plugins/grafana-influxdb-flux-datasource/versions/5.4.0/
download
into: /var/lib/grafana/plugins
✔ Installed grafana-influxdb-flux-datasource successfully
Restart grafana after installing plugins . <service grafana-server restart>
```



Settings

Name: Flux (InfluxDB) [BETA] — Default: on

HTTP

URL: http://influxdb1:9999

Whitelisted Cookies: Add Name

Auth

Basic Auth	☐	With Credentials	☐
TLS Client Auth	☐	With CA Cert	☐
Skip TLS Verify	☐		
Forward OAuth Identity	☐		

InfluxDB 2.0.0 Details

Organization: campus_borlange

Default Bucket: elsys

Token: configured

Data source connected and database found.

Fig 7. Grafana settings.

Note. Depending on how you run Grafana (native vs. container) the HTTP URL will be different.
Since I used Docker compose (the same applies to linking) the hostname is the container name. In this case it was influxdb1. If you run Grafana native it should be localhost.

The Organization, Default Bucket and Token is the same strings used to insert data into InfluxDB v2.x.

Remember that InfluxDB v2.x use the new query language Flux (as earlier stated). The easiest way of getting the correct Flux query is to use the Query Builder in InfluxDB and when finished press Script Editor to view the query. Then copy paste it into Grafana as:

```
from(bucket: "elsys")
  |> range(start:2019-10-29T00:00:00Z, stop:2019-10-30T00:00:00Z)
  |> filter(fn: (r) => r._measurement == "elt2" or r._measurement == "ersco2")
```

Example query for the co2 value and the ersco2 sensors

```
from(bucket: "elsys")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "ersco2")
  |> filter(fn: (r) => r._field == "co2")
```

Example query for he temperature value and the elt2 sensors

```
from(bucket: "elsys")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "elt2")
  |> filter(fn: (r) => r._field == "temperature")
```

4.3 Lab feedback

a) Were the labs relevant and appropriate and what about length etc?
b) What corrections and/or improvements do you suggest for these labs?