Prototype and Deployment of a Big Data Analytics Platform

CTT2.0 Carbon Track and Trace Deliverable D3.1

Atle Vesterkjær, Patrick Merlot (Numascale)

Oslo, Norway| 14 December 2016

climate-kic.org

Climate-KIC is supported by the
EIT, a body of the European Union

Contents

Preface

About LoCaL

This report was written through support from Low Carbon City Lab (LoCaL). LoCaL aims to reduce 1Gt of CO_2 and mobilize €25 billion of climate finance for cities annually by 2050. It is an innovation platform aiming to provide cities with better tools for assessing greenhouse gas emissions, planning, investing and evaluating progress. Started in 2015, LoCaL is a growing community of more than 20 organisations dedicated to unlocking climate finance for cities. This report was realized as part of the project Closing the Gap through Transformative LoCaL Action (CGTLA) under LoCaL.. LoCaL is a Climate-KIC flagship programme.

http://local.climate-kic.org. Contact: victor.gancel@climate-kic.org

About Climate KIC

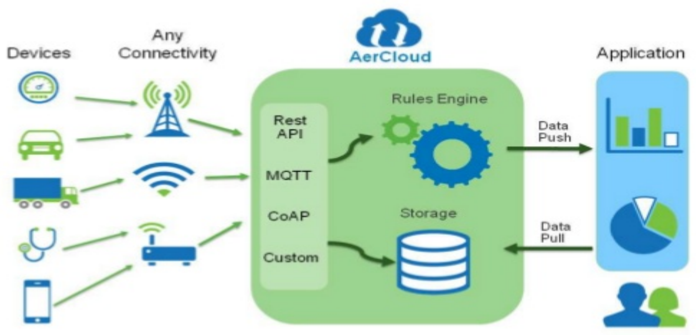
Climate-KIC is the EU's largest public private partnership addressing climate change through innovation to build a zero carbon economy. We address climate change across four priority themes: urban areas, land use, production systems, climate metrics and finance. Education is at the heart of these themes to inspire and empower the next generation of climate leaders. We run programmes for students, start-ups and innovators across Europe via centres in major cities, convening a community of the best people and organisations. Our approach starts with improving the way people live in cities. Our focus on industry creates the products required for a better living environment, and we look to optimise land use to produce the food people need. Climate-KIC is supported by the European Institute of Innovation and Technology (EIT), a body of the European Union.

About Carbon Track and Trace

The Carbon Track and Trace (CTT) project is intended to provide cities with real-time greenhouse gas (GHG) measurement capability. Traditional methods of building and maintaining municipal GHG emission inventories are expensive, time-consuming, and are of questionable utility for mitigation decision and planning support processes. CTT couples low-cost, open source sensors to a Big Data analytics platform that provides cities and regions with a unique capacity to directly measure the impacts of their policy and planning decisions and to develop a semi-autonomous system for building, maintaining, and reporting their annual GHG emissions.

Requirement Analysis

The requirements analysis for the Big Data Analytics Platform applies to the CTT project in terms of software and infrastructure for calibrating the emissions in a city. This involves gathering data from existing sources and finding a way to complement these datasets. CTT was based on the need to deploy own CO_2 sensors on the ground and produce its own data. These ideas are very similar in design to Numascale's end-to-end (E2E) Analytics platform.



Typical IoT end-to-end solution

Numascale's technology lowers the barrier for building high performance scale-up systems from commodity servers. Numascale has through working on multiple international E2E Analytics projects matured the software stack in their E2E Analytics Appliances. In these projects Numascale has optimized lower level libraries preserving scalability when the problem sizes grow.

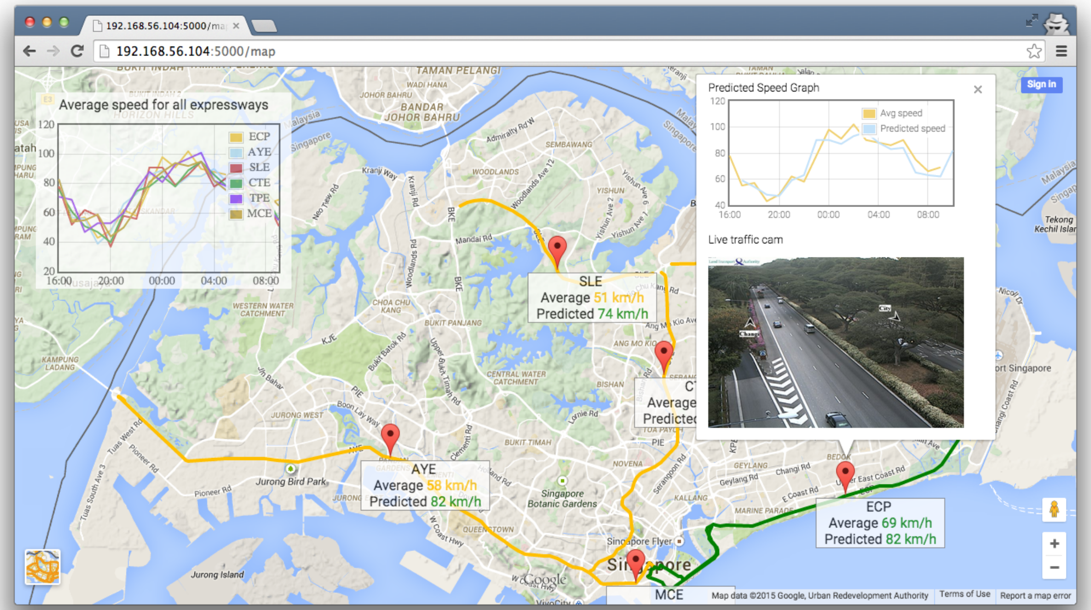
Previous E2E Analytics IoT platform

We present in this section two previous large IoT projects developed by Numascale to demonstrate the capability of the end-to-end IoT platform and Numascale's services.

Traffic Flow Prediction – Singapore

Numascale's traffic analysis demo predicts traffic patterns over the next hour. Accurate prediction models were obtained by combining:

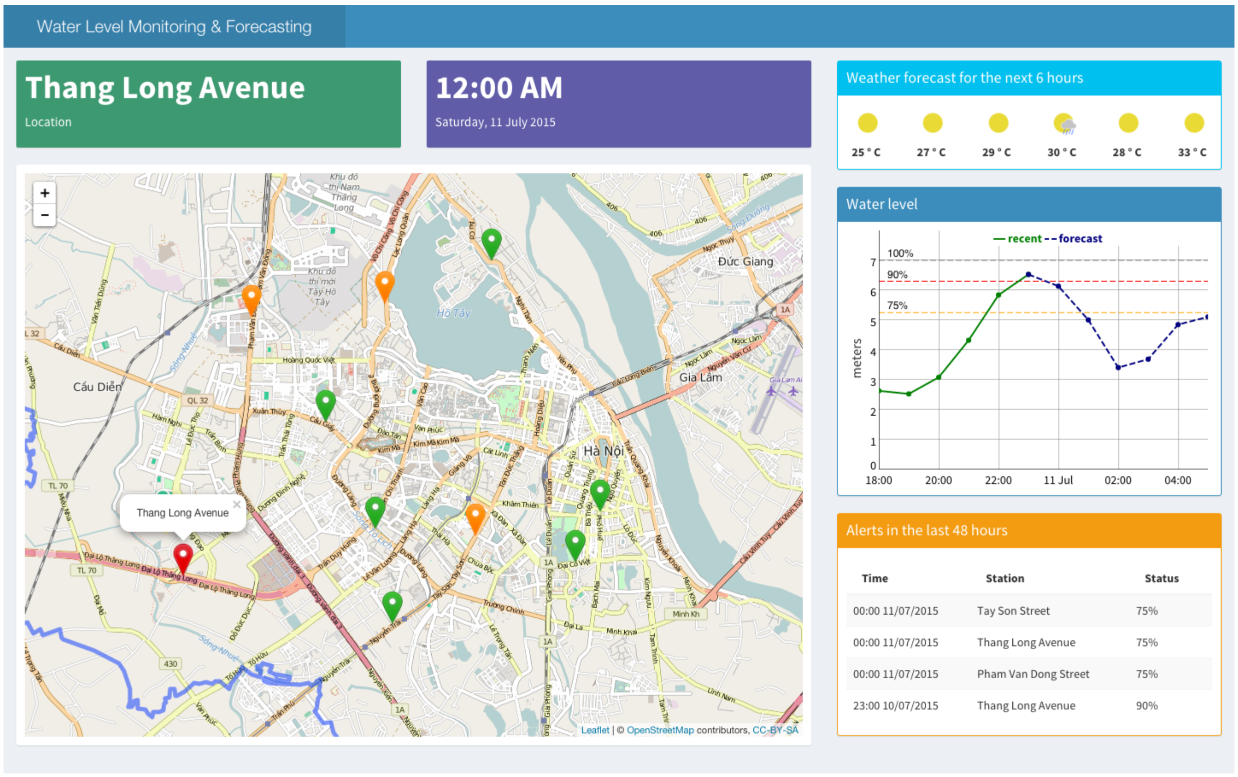
- real-time inputs using data from GPS tracking devices embedded in cars and aggregated in real-time,
- historical data based on 3-years of Traffic data stored in-memory.



Dashboard for the Singapore's traffic flow prediction demo.

Flood Prediction – Hanoi

Numascale's flood prediction demo uses real-time water level sensors and weather data to predict floods within the next 6 hours and to issue real-time alerts to residents.



Time	Station	Status
00:00 11/07/2015	Tay Son Street	75%
00:00 11/07/2015	Thang Long Avenue	75%
00:00 11/07/2015	Pham Van Dong Street	75%
23:00 10/07/2015	Thang Long Avenue	90%

Dashboard for the Hanoi’s flood prediction demo

CTT Analytics IoT platform

In addition to the unique benefits provided by its powerful machines adapted to large in-memory analytics, Numascale is involved in all steps of an end-to-end analytics solution. The consortium uses a 4-node Numascale Analytic Appliance as the platform for the project. Numascale also provides services for the following requirements that are part of the Analytics Platform:

- providing a platform for analytics adapted to the project’s needs
- retrieving data from the sensors,
- cleaning/processing the raw data,
- storing the data in databases,
- comparing historical data with real-time data stream,
- finding patterns in data,
- presenting/visualizing the data,
- building predictive models to provide actionable insights to other domains of expertise (municipality planning & any decision supports systems).

The deployment of the system of sensors is described in WP2 and implemented by other partners (CTT, Wireless Trondheim, TheThingsNetwork).

State of the art technology overview

What is a big data platform?

A big data platform is a type of IT solution that combines the features and capabilities of several big data applications and utilities within a single solution. It is an enterprise class IT platform that enables organizations (e.g. CTT, CTT partners) in developing, deploying, operating and managing a big data infrastructure/environment. Big data platforms generally consist of large data storage, servers, databases systems, business intelligence and other data management utilities. It also supports custom development, querying and integration with other systems. The primary benefit behind a big data platform is to reduce the complexity of multiple vendors/solutions into a one cohesive solution and to scale in a good way when the size of the data grows large. Big data platforms can also delivered through cloud where the provider provides an all inclusive big data solutions and services.

The Numascale Big Data platform

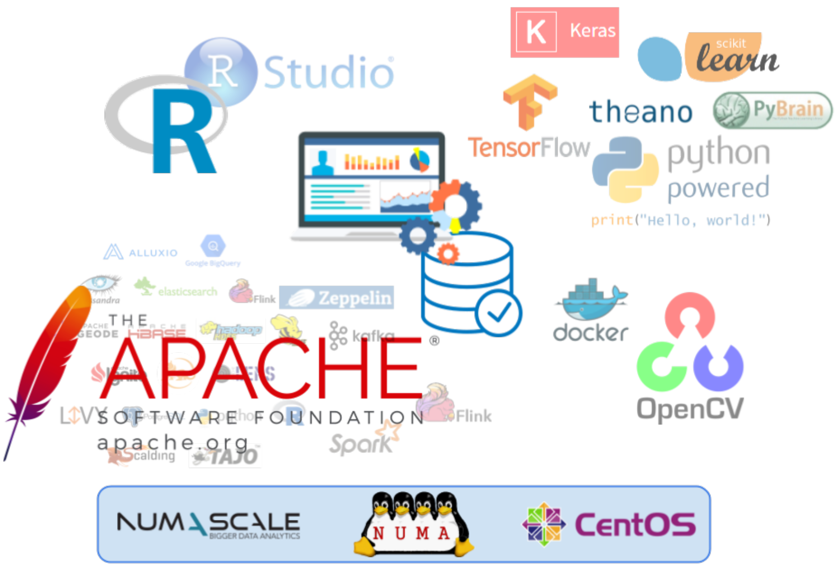
NUMA: Non-Uniform Memory Access

The Numascale technology enables the realization of large scale shared memory systems from commodity servers enabling large shared memory systems to be built at the price of clusters. This is achieved through the use of Numascale's adapter card connected to the HyperTransport processor bus of standard AMD based servers. The technology uses a low latency torus fabric based on the SCI standard. The resulting system is a cache coherent Non-Uniform Memory Access (ccNUMA) machine. Shared memory is attractive to developers, as any processor can access any data in the system through direct load and store operations, thus making the programming of the system easier and the code less error-prone. The Numascale system presents itself as a single system with many cores and large memory for the programmer. It runs a single image standard Linux operating system, which simplifies the operation of the system.

The Numascale Analytics Appliance in Trondheim contains 4 IBM servers x3755-M3, 128 cores, 512 GB of Memory, and 2TB of storage.

Software Stack & Analytic Appliance

R & Python are becoming the world's most popular statistical programming languages and environments. The Numascale Analytic Appliance provides R with all the memory it needs for massive data set computations. Numascale goal is to enable Data scientists to use R for any size problem, and avoid the need to recode in Python or C++ once the dataset and complexity grows. Similarly any application developed in Python can run in parallel with Apache Spark using the in-memory capabilities of a Numascale server.



Numascale's Big Data platform: a Linux-based platform with more available components

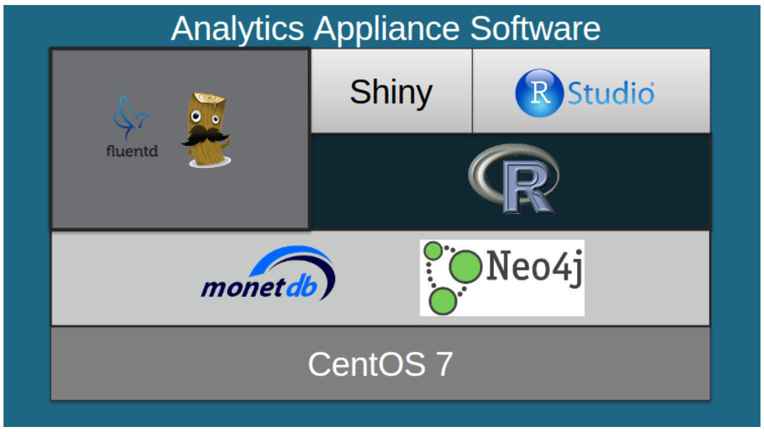
Adaption of standard and development of custom components

Standard components

The platform will be delivered with standard components (hardware and software) to deal with big data, external big data sources, a range of analytics to use upon and with tools to deliver applications/reports. The Numascale Analytic Appliance comes with CentOS 7.1, an optimized Linux kernel and a pre-installed sets of components that have proven to work well on similar demanding backend analytics projects. These include:

- the Numascale R appliance: an optimized version of the Open Source R from CRAN, which comes with Numascale preloaded LAPACK module for this hardware architecture, NC-LAPACK)
- MonetDB: an up-to-date pioneer among the column-store database management systems.
- R & RStudio: an open source statistical language and tools, to make sense of data.
- Shiny: An intuitive web application framework for R turning analyses into interactive web applications

It can be summarized with the following stack:



Components for Big Data

Providing standard components means taking into account various specifications of data and data sources: their degree of structure, their volume, their method of acquisition, their historical significance, their quality, their value, their relationship to other forms of data. These specifications will determine how data is managed, processed, used and integrated.

Components for Analytics

There are many types of analysis that can be performed, by different types of users (or systems), using many different tools, and though a variety of channels. Some types of analysis require current information and others work mostly with historical information. Some are performed proactively and others are reactive. The architecture design must be universal and extensible to support a full range of analytics, and should contain mostly free and open-source software suites in order to promote sharing of knowledge between Cities (e.g. Apache Hadoop family, R, Python, multiple open-source libraries and modules, ...).

Components for Applications and Reporting

The added-value of analytics resides in the way insights are delivered. These insights must be embedded in the applications that users/workers use to perform their decisions/jobs. Real-time analytics is a crucial part of an IoT platform. It enables the business to leverage information and analysis as events are unfolding. It includes:

- Speed of Thought Analysis – Analysis is often a journey of discovery, where the results of one query determine the input to the next. The system must support this journey in an expeditious manner. System performance must keep pace with the users' thought process.
- Interactive Dashboards – Dashboards provide a heads-up display of information and analysis that is most pertinent to the user. Interactive dashboards allow the user to immediately react to information being displayed, providing the ability to drill down and perform root cause analysis of situations at hand.
- Advanced Analytics – Advanced forms of analytics, including data mining, machine learning, and statistical analysis enable businesses to better understand past activities and identify trends that can carry forward into the future. Applied in real-time, advanced

analytics can enhance customer interactions and buying decisions, detect fraud and waste, and enable the business to make adjustments according to current conditions.

- Event Processing – Real-time processing of events enables immediate responses to existing problems and opportunities. It filters through large quantities of streaming data, triggering predefined responses to known data patterns.

This platform should aim at simplifying ways for building such applications or reports.

Custom components

Standard components may not always be sufficient for a developer/user of the platform. Numascale is offering support for installing, developing and maintaining custom components either for a specific user or as a new component of the platform. In the case where a specific software or libraries can't be found on the platform, Numascale permits this flexibility for the users to install extra tools on their own. In other cases, if no software either fit the platform nor exists to perform a specific task, Numascale might develop a specific solution to fulfil that task. (i.e. collecting and storing the data coming from CTT sensors).

Deployment of the platform

The system as described above was set up. Specific guides were made available to get started: How to access it using SSH/putty from Linux/Windows, the Numascale Analytic Appliance handbook.

In addition a set of public datasets for climate research were provided.

The system set-up then included:

- sensor data being collected in real-time in our database through any type of communication network (2G/3G/4G mobile network, WiFi, LoRaWan/SigFox, ...) connected to internet.
- external data sets (historic or real-time data) and data sources being stored on the server.
- an analytics engine working optimally with the database system to provide easy and fast data aggregation and predictive analytics.
- visualization tools to help generate reports and dashboards from the analytics engine.
- using the insights provided by the visualization to automate actions

Usage of the Big Data analytics platform

CTT sensor data collection

In the CTT project we started with a few pre-configured IoT devices sending MQTT frames on a LoRaWAN network in two cities: Trondheim (Norway) and Vejle (Denmark). The devices were IoT open-source nodes from Libelium equipped with various sets of air quality sensors (one or several of CO2, NO2, PM1, PM2.5, PM10 sensors) connected to an Arduino-based board, with always a basic set of sensors reporting Temperature, Pressure, Humidity and Battery level.

The first custom components added to the Numascale Analytics appliance were:

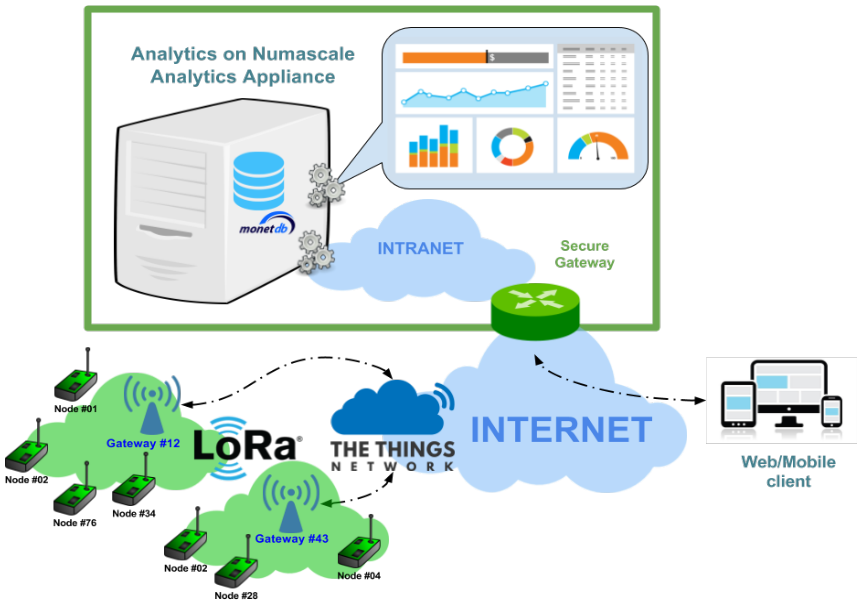
- Mosquito: an Open Source MQTT Broker to collect the messages in real-time.
- Paho (MQTT library) to actually interpret MQTT messages programmatically.
- R & Python typical data analytics packages (e.g. Numpy, Pandas, Lubridate, ...)
- simple open linux commands tools to automate processes.

Numascale also developed and maintained scripts to:

- collect the MQTT messages
- extract the compressed payload into meaningful values and measurements
- store the relevant data and metadata into CTT's database systems (MonetDB)

Any data sent over the same LoRaWAN network and application ID by any new or upgraded device will be automatically stored in CTT's databases.

The flow of data is summarized in the schema below.



The following other software components were added:

- Leaflet: an open-source JavaScript library to build interactive maps with layers
- Apache Spark: a fast and general engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing. Spark is at the core of Numascale IoT demos that were used to demonstrate the capabilities of the platform.

The CTT data collection solution software is open source and is available at https://github.com/Carbon-Track-and-Trace/ctt_data_collection. It will automate the download/fetching of MQTT messages sent by all the devices in both Trondheim and Vejle, extract the payload and store sensors' measurements and metadata on a MonetDB database.

Various datasets about Traffic and Air Quality

In addition to the Numascale Analytics Appliance and custom built software, a set of public datasets for climate research was also provided. The datasets are collected from:

- the CO2 SATELLITE DATA ENVIRONMENT provided by NASA
  - Methane and CO2
    ACOS data focuses on the transport mechanisms of Carbon Dioxide (CO2) and Methane (CH4). GOSAT seeks to observe sources and sinks of greenhouse gases, particularly CO2, over time in an effort to make future reductions in greenhouse gas emissions.
  - High-Resolution CO2
    OCO-2 (Lite) is the only mission focusing solely on the behavior of atmospheric carbon dioxide. Data from OCO-2 will provide a complete picture of CO2 sources and sinks. OCO-2 strives to map the global geographic distribution of CO2, and study the CO2 changes over time.
  - Validation on the Ground
    TCCON data structures are ground truths. Often times, researchers use TCCON

establishments for validation purposes with their satellite models. There are several TCCON sites to choose from globally.

   - Tropospheric CO2
     TES data focuses on the Earth's troposphere. Measurements from the TES instrument improve further understanding of long-term variations of minor gases and the resulting effects on climate and the biosphere.

- Emissions of greenhouse gases by SSB
  (Statistisk sentralbyrå i.e.Statistics Norway)
- Luftkvalitet.info, utviklet og driftes av NILU (Norsk institutt for luftforskning)

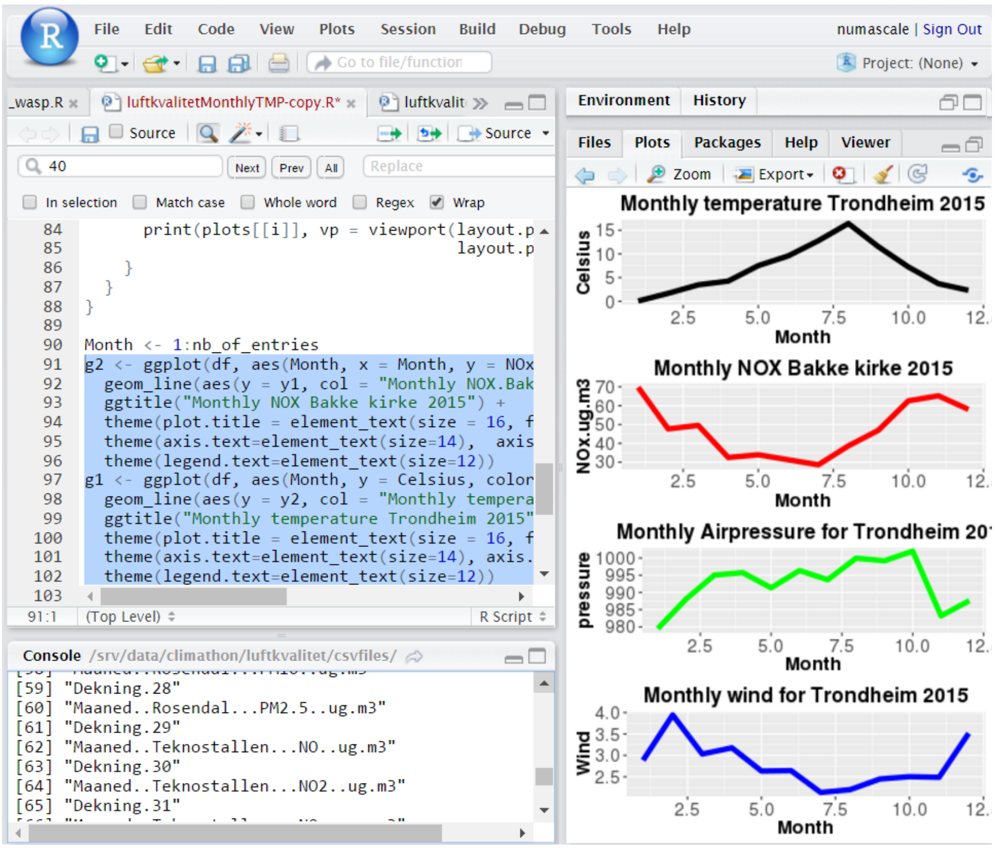
By providing the proper tools and code in R, Numascale has been able to demonstrate how one could combine, present and find patterns in data from several sources like NASA ACOS satellite, NILUs ground stations in Trondheim, eklima.no's data for Trondheim and streaming data from CO_2 sensors deployed by CTT.

Analytics Engine & Use Case examples

In addition to deploying the platform and collecting data from CTT sensors, Numascale also presented a set of proof-of-concept of analytics studies.

Air quality data from NILU

NILU NO_X Data is combined with weather data in RStudio. This is the working environment that Numascale hosts on the Numascale analytics appliance.

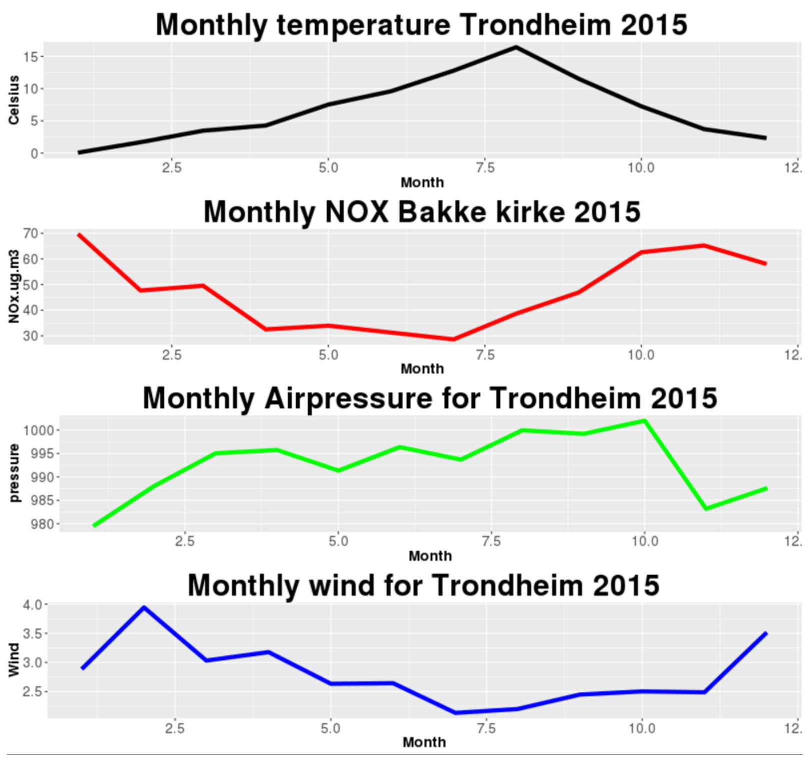


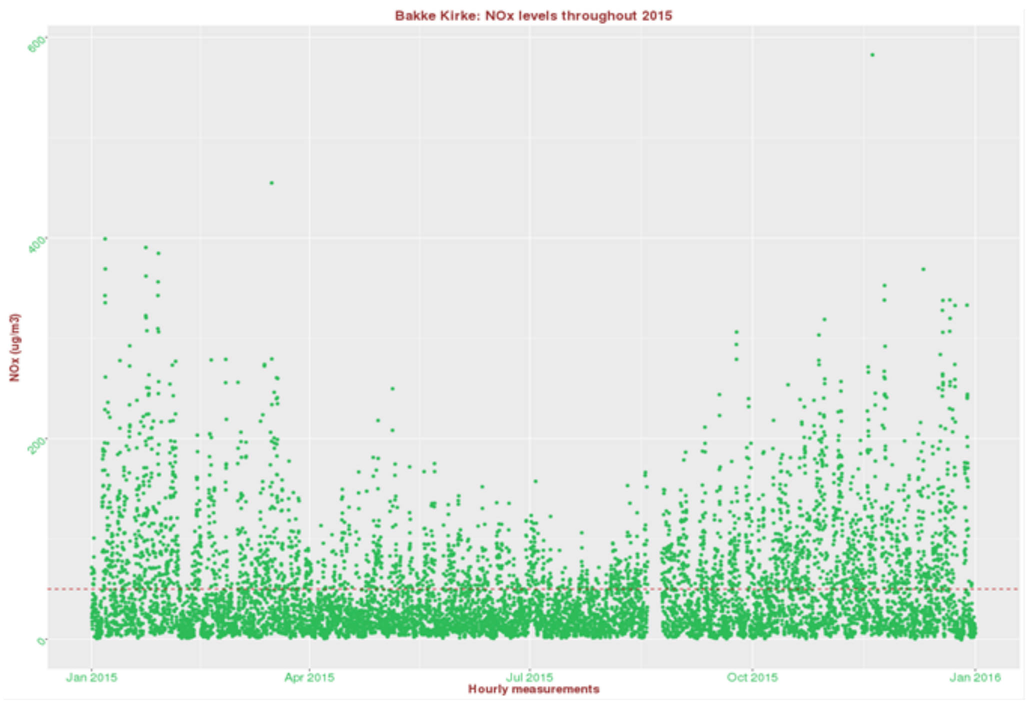
```
        print(plots[[i]], vp = viewport(layout.p
                                        layout.p
      }
    }
}

Month <- 1:nb_of_entries
g2 <- ggplot(df, aes(Month, x = Month, y = NOx
  geom_line(aes(y = y1, col = "Monthly NOX.Bak
  ggtitle("Monthly NOX Bakke kirke 2015") +
  theme(plot.title = element_text(size = 16, f
  theme(axis.text=element_text(size=14),  axis
  theme(legend.text=element_text(size=12))
g1 <- ggplot(df, aes(Month, y = Celsius, color
  geom_line(aes(y = y2, col = "Monthly tempera
  ggtitle("Monthly temperature Trondheim 2015"
  theme(plot.title = element_text(size = 16, f
  theme(axis.text=element_text(size=14), axis.
  theme(legend.text=element_text(size=12))
```

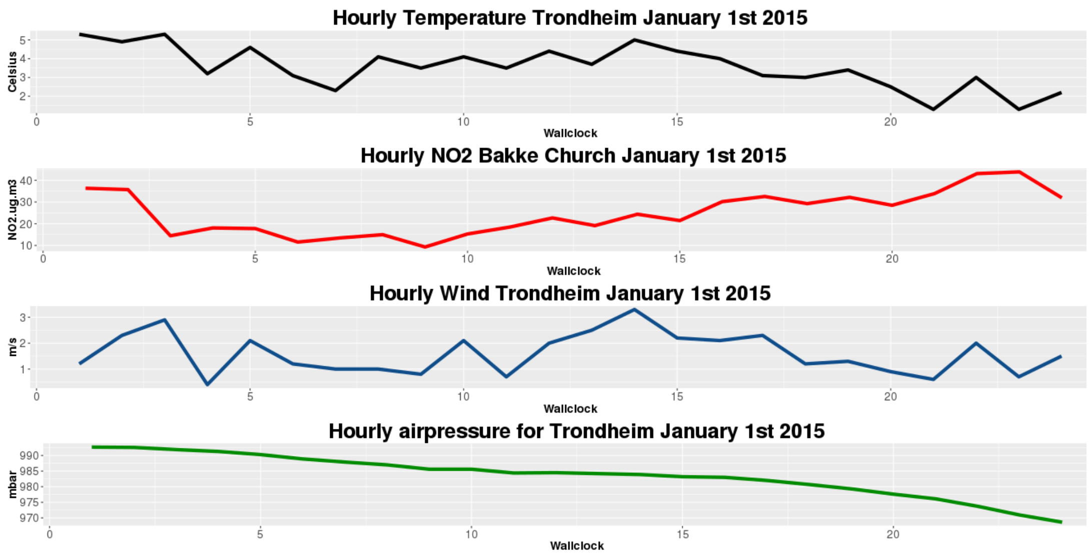
Console /srv/data/climathon/luftkvalitet/csvfiles/

```
[59] "Dekning.28"
[60] "Maaned..Rosendal...PM2.5..ug.m3"
[61] "Dekning.29"
[62] "Maaned..Teknostallen...NO..ug.m3"
[63] "Dekning.30"
[64] "Maaned..Teknostallen...NO2..ug.m3"
[65] "Dekning.31"
```

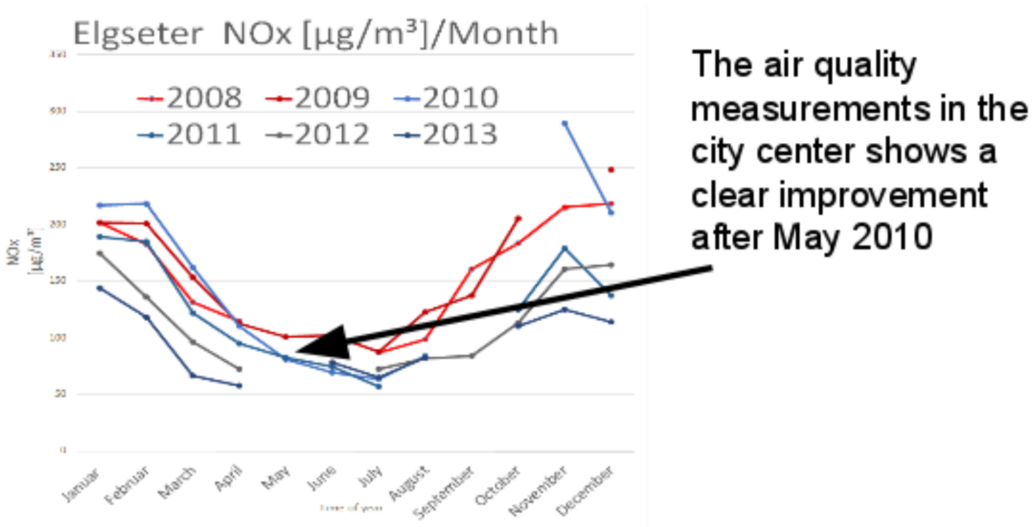




NILU NO_x raw data measurements over time.

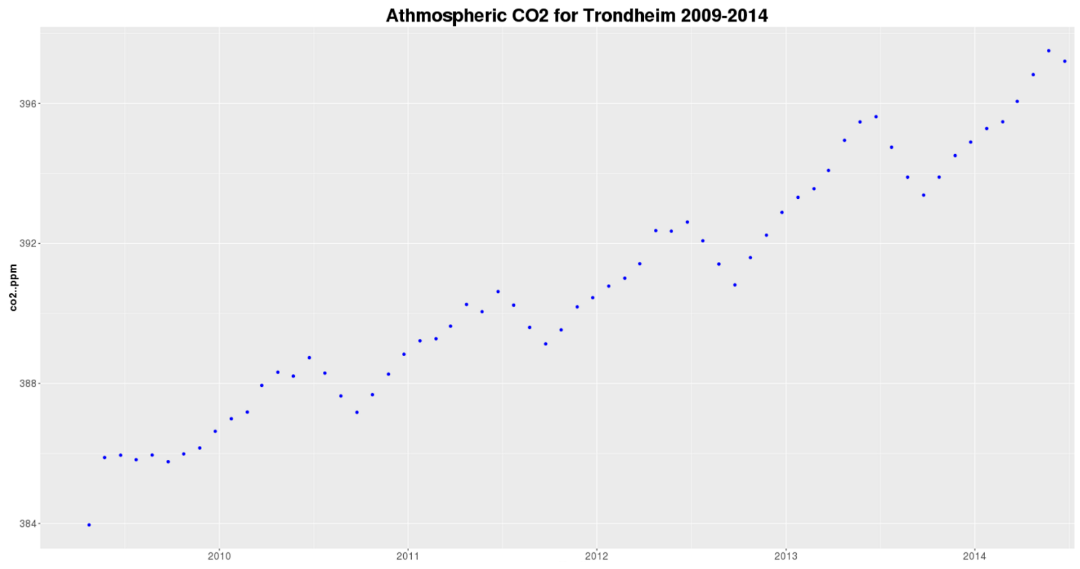


NILU hourly-averaged NO_x measurements in Bakke Church. You can see that low the combination of low air pressure, little wind and low temperature correlates with High NO_2 values.



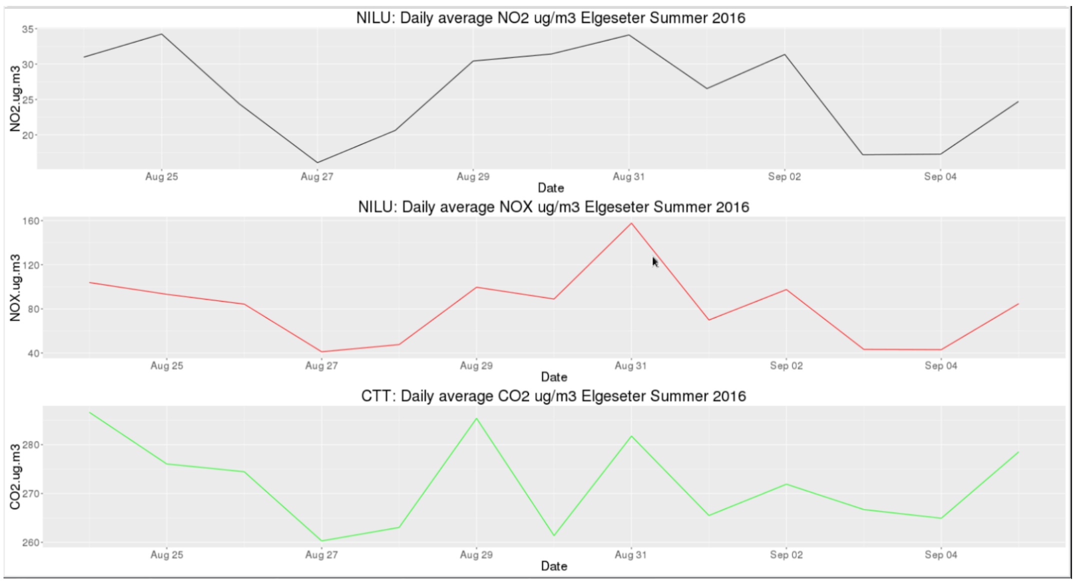
Using NILU data on NO_x in Elgeseter (Trondheim) to show impact of urban measures. In May 2010 a new road opened that took some traffic pressure away from Elgeseter. with CTT we can document the effect using open data that were not originally used by Trondheim Municipality.

Satellite data from NASA

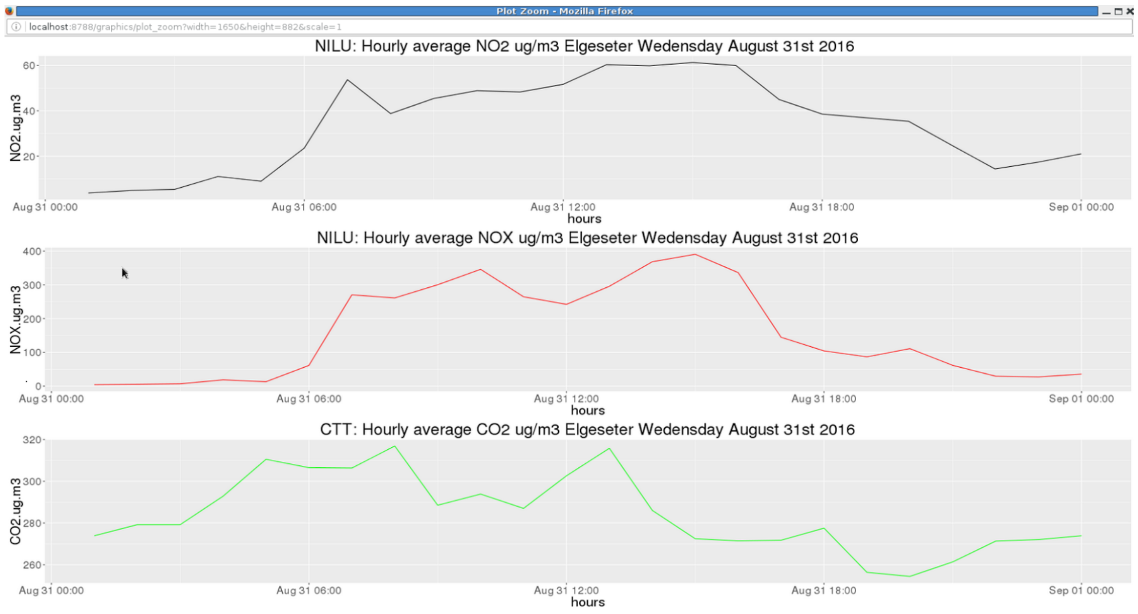


5-years window observing sources and sinks of greenhouse gases around Trondheim. The trends show that the CO_2 emission increases. This correlates with other data for Trondheim.

Data from CTT sensors



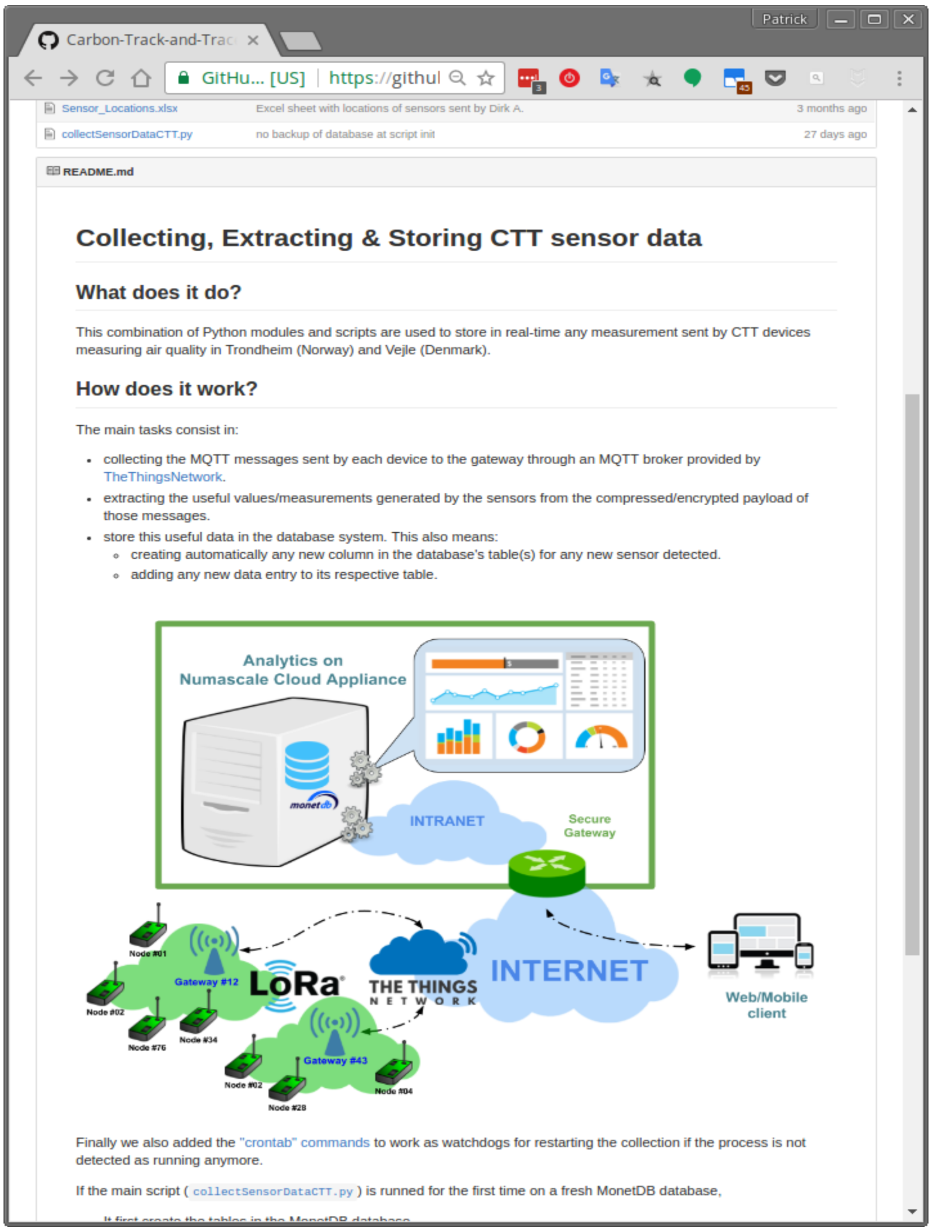
Example comparing data from CTT's owns sensors to NILUs "gold standard" sensors in Elgeseter (Trondheim).



Comparison of CTT to NILU data: One day hourly NILU measurements of NO_2, NO_x & CO_2 in Elgeseter (Trondheim). This is a very trafficked road; values are highest during the day and there is correlation between sensor sources at a low to medium level.

Documentation & Reproducibility

For reproducibility in any other city, the scripts and modules created are all open-source and available with a technical installation guide on the NumaScale CTT Github account (https://github.com/Carbon-Track-and-Trace/ctt_data_collection). Details can be found in the Annex.



Sensor_Locations.xlsx	Excel sheet with locations of sensors sent by Dirk A.	3 months ago
collectSensorDataCTT.py	no backup of database at script init	27 days ago

README.md

Collecting, Extracting & Storing CTT sensor data

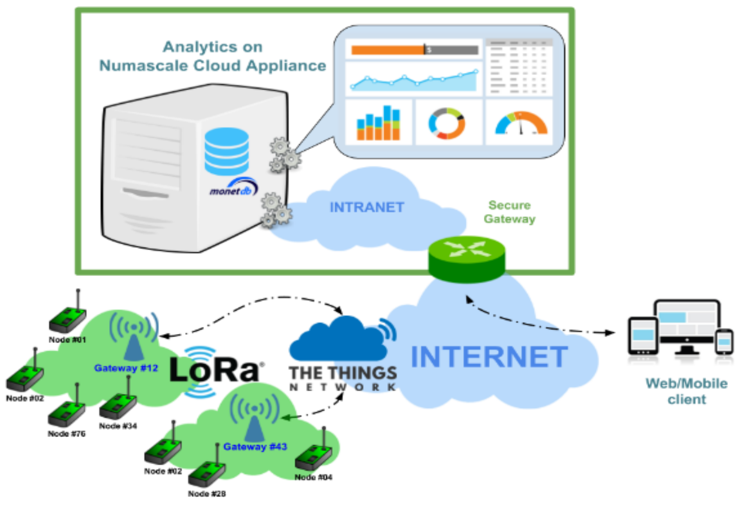
What does it do?

This combination of Python modules and scripts are used to store in real-time any measurement sent by CTT devices measuring air quality in Trondheim (Norway) and Vejle (Denmark).

How does it work?

The main tasks consist in:

- collecting the MQTT messages sent by each device to the gateway through an MQTT broker provided by TheThingsNetwork.
- extracting the useful values/measurements generated by the sensors from the compressed/encrypted payload of those messages.
- store this useful data in the database system. This also means:
  - creating automatically any new column in the database's table(s) for any new sensor detected.
  - adding any new data entry to its respective table.



Finally we also added the "crontab" commands to work as watchdogs for restarting the collection if the process is not detected as running anymore.

If the main script (`collectSensorDataCTT.py`) is runned for the first time on a fresh MonetDB database,

Annex

Annex: Collecting, Extracting & Storing CTT sensor data

Guide and software code to collect, extract, and store sensor data. Code written in Python.

Code reproduced here for reference only, latest version available from
https://github.com/Carbon-Track-and-Trace/ctt_data_collection/blob/master/README.md

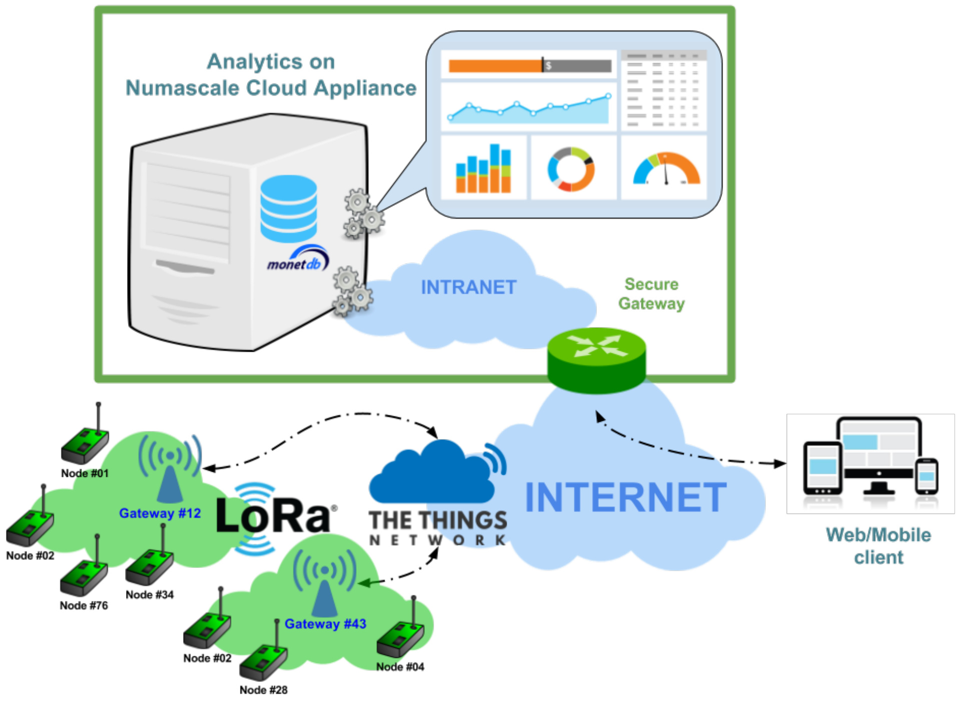
What does it do?

This combination of Python modules and scripts are used to store in real-time any measurement sent by CTT devices measuring air quality in Trondheim (Norway) and Vejle (Denmark).

How does it work?

The main tasks consist in:

- collecting the MQTT messages sent by each device to the gateway through an MQTT broker provided by TheThingsNetwork.
- extracting the useful values/measurements generated by the sensors from the compressed/encrypted payload of those messages.
- store this useful data in the database system. This also means:
  - creating automatically any new column in the database's table(s) for any new sensor detected.
  - adding any new data entry to its respective table.



Finally we also added the "crontab" commands to work as watchdogs for restarting the collection if the process is not detected as running anymore.

If the main script ([`collectSensorDataCTT.py`]) is runned for the first time on a fresh MonetDB database,

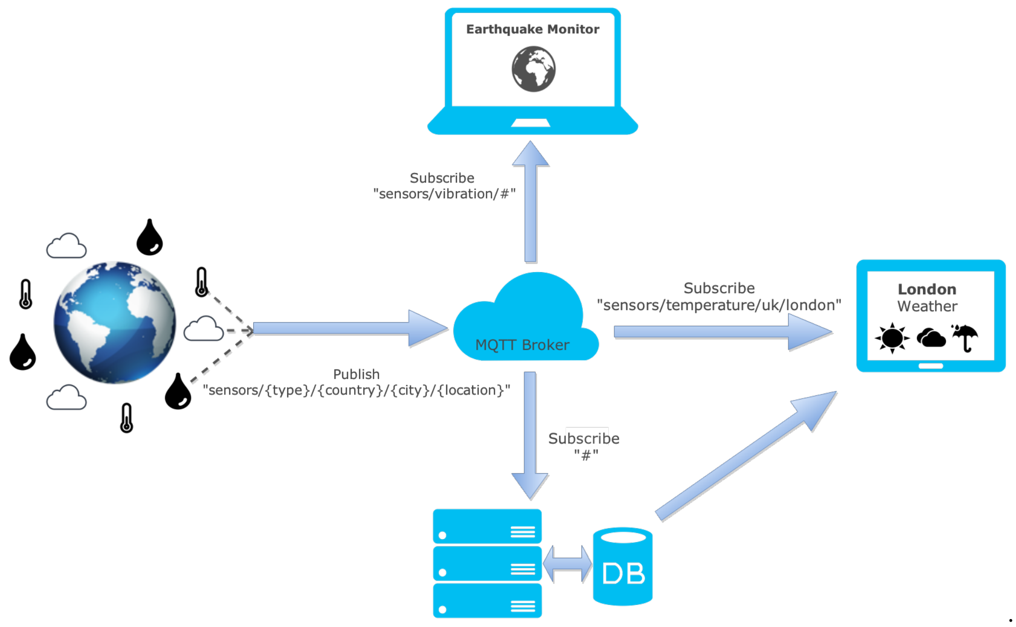
- It first create the tables in the MonetDB database,
- then it used to read the historical sensor data from TheThingsNetwork REST API which was deprecated in August 2016, and store them in MonetDB,
- for every new type of sensor appearing in the sensor's payload, a new column in MonetDB is automatically created. This allows for a highly dynamic intake of any new sensor.
- finally an infinite loop waits for any single new MQTT message sent by the sensors and stores each one in the database as well.

Building your own air quality data collection

Getting started with MQTT and listening to MQTT messages

In *CTT 2.0* the MQTT messages are captured by gateways connected to *The Things Network*.

The MQTT protocol allows any device to send MQTT messages to a MQTT Broker (in this case *The Things Network*) whose task is to redistribute copies of each message to the subscribers of any topic that includes that device (as shown on the sketch below).


Earthquake Monitor
Subscribe "sensors/vibration/#"
MQTT Broker
Subscribe "sensors/temperature/uk/london"
London Weather
Publish "sensors/{type}/{country}/{city}/{location}"
Subscribe "#"
DB

This source code contains ways to collect the MQTT message using the python package "Paho", but you can test your MQTT broker by using Mosquitto to fetch ("to subscribe to") incoming messages as they arrive:

```
mosquitto_sub -h staging.thethingsnetwork.org -p 1883 -u
70B3D57ED0000AD8 -P LJtFqN8NSqHQzDaaZkHVQ+G+KCDJ+fZbptl94NyUXGg= -t
70B3D57ED0000AD8/devices/+/up
```

In the command line example above, we connect to the MQTT broker hosted by "**staging.thethingsnetwork.org**", on port "**1883**" with user/password identified by the **-u** and **-P** options with topic(s) on the form "**70B3D57ED0000AD8/devices/+/up**".

This topic actually matches any topic which starts by "**70B3D57ED0000AD8/devices/**" and finishes by "**/up**".

Understanding the Libelium payload format and extracting useful measurements

Libelium provides *arduino*-based devices equipped with communication capabilities and various types of sensors. In this project, the MQTT messages include a payload with the measurements of the sensors.

The order of the measurements in the payload is chosen by CTT when configuring the device. Here is an example of *arduino*-based code showing it:

```
frame.createFrame(BINARY); frame.addSensor(SENSOR_GP_CO2,
co2concentration); frame.addSensor(SENSOR_GP_NO2, no2concentration);
frame.addSensor(SENSOR_GP_TC, temperature);
frame.addSensor(SENSOR_GP_HUM, humidity);
frame.addSensor(SENSOR_GP_PRES, pressure);
if(PMX == true){
      frame.addSensor(SENSOR_OPC_PM1, OPC_N2._PM1);
      frame.addSensor(SENSOR_OPC_PM2_5, OPC_N2._PM2_5);
      frame.addSensor(SENSOR_OPC_PM10, OPC_N2._PM10);
} else {
      frame.addSensor(SENSOR_OPC_PM1, -1);
      frame.addSensor(SENSOR_OPC_PM2_5, -1);
      frame.addSensor(SENSOR_OPC_PM10, -1);
}
frame.addSensor(SENSOR_BAT, battery); frame.showFrame(); char
data[frame.length*2 + 1]; Utils.hex2str(frame.buffer, data,
frame.length);
```

Most configurations of the sensors were also shared here on Github.

Once the measurements concatenated in the binary frame, the payload is compressed in hexadecimal, then in base 64 to optimize the size of the MQTT message.

The Python module **CTT_Nodes.py** extracts the Binary Frame from Libelium and returns the measurements in a dictionary.

It can extract data from the payload compressed in base 64, e.g.:

```
ctt_data_collection$ python ./CTT_Nodes.py -h usage: CTT_Nodes.py [-
h] [-hex BASE16 | -b64 BASE64] [-f FILE_PATH] [-v]  optional
arguments:   -h, --help           show this help message and exit
-hex BASE16, --base16 BASE16                    Read and
Extract payload from a message in
hexadecimal, such as 3c3d3e002866155818544b43545430322
3778ff596d1438900000000903d0a03419200b6b142935a6fc8473
428   -b64 BASE64, --base64 BASE64                    Read and
Extract payload from a message in base64,
such as PD0+ADdk4lcYVkpDVFQwMSNgj0hmtUOJAAAAAJD2KJpBko
DlkUKTsATGR5fJMoU/mKRU8j+Z4BE5QDRa   -f FILE_PATH, --file_path
FILE_PATH                        Input file to read data from to be
store in DB,                       formatted as dictionaries on
each line.   -v, --verbose         increase output verbosity
ctt_data_collection$ python ./CTT_Nodes.py -b64
PD0+ADdk4lcYVkpDVFQwMSNgj0hmtUOJAAAAAJD2KJpBkoDlkUKTsATGR5fJMoU/mKRU
8j+Z4BE5QDRa {'SENSOR_BAT': 90,  'SENSOR_GP_CO2': 362.799072265625,
'SENSOR_GP_HUM': 72.9482421875,  'SENSOR_GP_NO2': 0.0,
'SENSOR_GP_PRES': 101385.375,  'SENSOR_GP_TC': 19.270000457763672,
'SENSOR_OPC_PM1': 1.0406123399734497,  'SENSOR_OPC_PM10':
2.8917160034179688,  'SENSOR_OPC_PM2_5': 1.8932080268859863}
ctt_data_collection$
```

or extract data from the payload compressed in hexadecimal (base16), e.g.:

```
ctt_data_collection$ python ./CTT_Nodes.py -hex
3c3d3e002866155818544b435454303223778ff596d1438900000000903d0a034192
00b6b142935a6fc8473428 {'SENSOR_BAT': 40,  'SENSOR_GP_CO2':
419.1793518066406,  'SENSOR_GP_HUM': 88.85546875,  'SENSOR_GP_NO2':
0.0,  'SENSOR_GP_PRES': 102622.703125,  'SENSOR_GP_TC':
8.1899995803833} ctt_data_collection$
```

Creating your *MonetDB* database

Once you can:

- check that you receive MQTT message using Mosquitto
- check that you can extract sensors' measurements

it may become interesting to store the data.

We choose **MonetDB** to do so.

Please refer to their documentation to install MonetDB.

Create one MonetDB instance

Login as root to create the instance, e.g. *mydbfarm* under */home/* directory:

```
# monetdbd create /home/mydbfarm/ # monetdbd start /home/mydbfarm/ #
monetdb create ctt # monetdb release ctt
```

Connect to your *MonetDB* instance using mclient

Note: the initial password for the "monetdb" username is also "monetdb".

```
# mclient -u monetdb -d ctt `password: Welcome to mclient, the
MonetDB/SQL interactive terminal (Jun2016) Database: MonetDB
v11.23.3 (Jun2016), 'mapi:monetdb://Jarvis:50000/ctt' Type \q to
quit, \? for a list of available commands auto commit mode: on sql>
```

Create a new user/password for your database

The new user is given his own schema for the MonetDB database. First connect the instance using **mclient** and the default "monetdb" username, then create a new schema and by specifying your new username and password.

```
sql>CREATE USER "co2" WITH PASSWORD 'ctt' NAME 'My new database'
SCHEMA "sys"; sql>CREATE SCHEMA "co2" AUTHORIZATION "co2"; sql>ALTER
USER "co2" SET SCHEMA "co2"; sql>\q
```

Then you should be able to connect to the MonetDB instance using your new username.

```
$ mclient -u co2 -d ctt password: Welcome to mclient, the
MonetDB/SQL interactive terminal (Jul2015-SP1) Database: MonetDB
v11.21.11 (Jul2015-SP1), 'mapi:monetdb://numascale-r:50000/ctt' Type
\q to quit, \? for a list of available commands auto commit mode: on
sql>
```

Running the automated data collection

Now everything is ready to create the necessary tables in the database and launch the data collection by listening continuously to any incoming MQTT message.

Simply run the following command:

```
$ ./collectSensorDataCTT.py
```