www.opennms.co.uk

Final Report

*Dr Craig Gall*en, Director Entimoss Ltd (OpenNMS UK)

Email : craig.gallen@entimoss.com
: cgallen@opennms.org

Mobile: +44 (0) 7789 938012

entimOSS limited Company registered in England and Wales No. 06402040

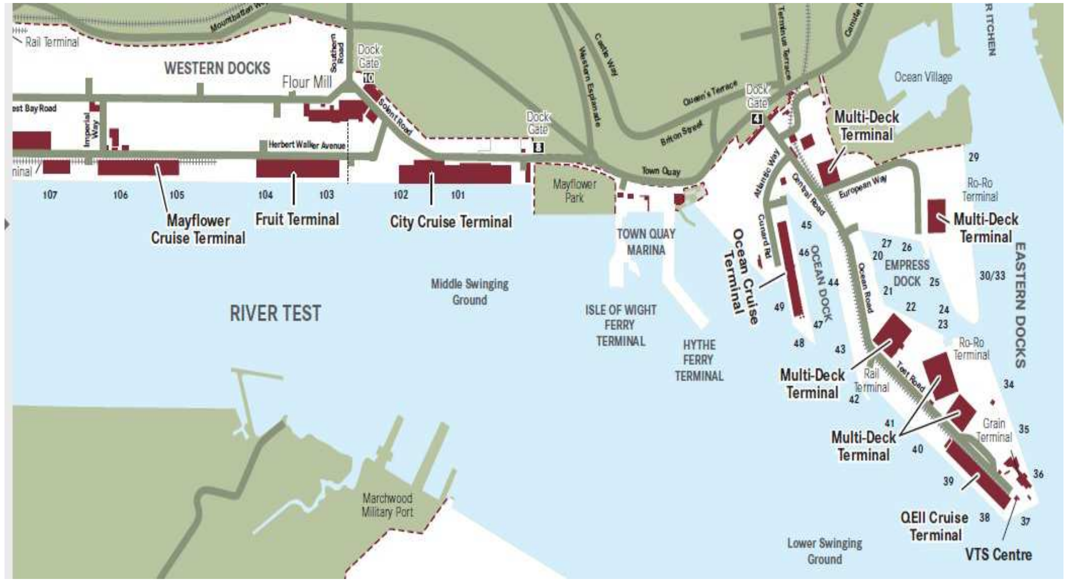
Contents

- **Background**
- **Solent Airwatch**
- **Why Use OpenNMS for IoT ?**
- **Architecture for IoT measurements**
- **Grafana Dashboard and OpenNMS Measurements API**
- **OpenNMS MQTT Plugin**
- **Where is the code**
- **Next Steps**

Southampton's (Smart) Port

www.opennms.co.uk

- The UK's number one cruise port, which welcomes 1.7m passengers
- Each ship up to 6000 passenger and crew
- Ship turn around 1-2 days
- Increasing problem of air pollution due to generators running while ships in port
- Each container ship arriving is like 50 trucks driving through town



Solent Airwatch

- **http://www.solentairwatch.co.uk/**
- **Community air monitoring project**
- **Not for profit charity run by Southampton volunteers**
- **With University of Southampton and Solent University**
- **Sniffy – version 1**
  - https://github.com/SolentAirWatch/sniffy
  - Raspberry Pi Zero
  - Low cost laser Smoke detector
  - Ruggedized container

SOLENT AIR WATCH

The Solent Airwatch Challenge

- **The problem**
  - Currently impossible to accurately monitor city wide pollution
  - Legal enforcement of air quality requires calibrated sensors
  - Commercial calibrated air quality sensors very expensive
  - Cheap air quality sensors have wide variation in sensitivity and cant be used for enforcement
- **The solution**
  - Community sourced low cost air sensors
  - Gamification – web site to show your sensor data and city wide view
  - Engage schools, maker space groups , private citizens and city council
- **The Differentiator - Data Science**
  - Data Science makes low cost sensors viable for accurate measurements
  - Peer reviewed experiment plan to validate the science behind data processing
  - Big data processing of city wide sensor data
  - Neural Network based calibration against commercial sensors

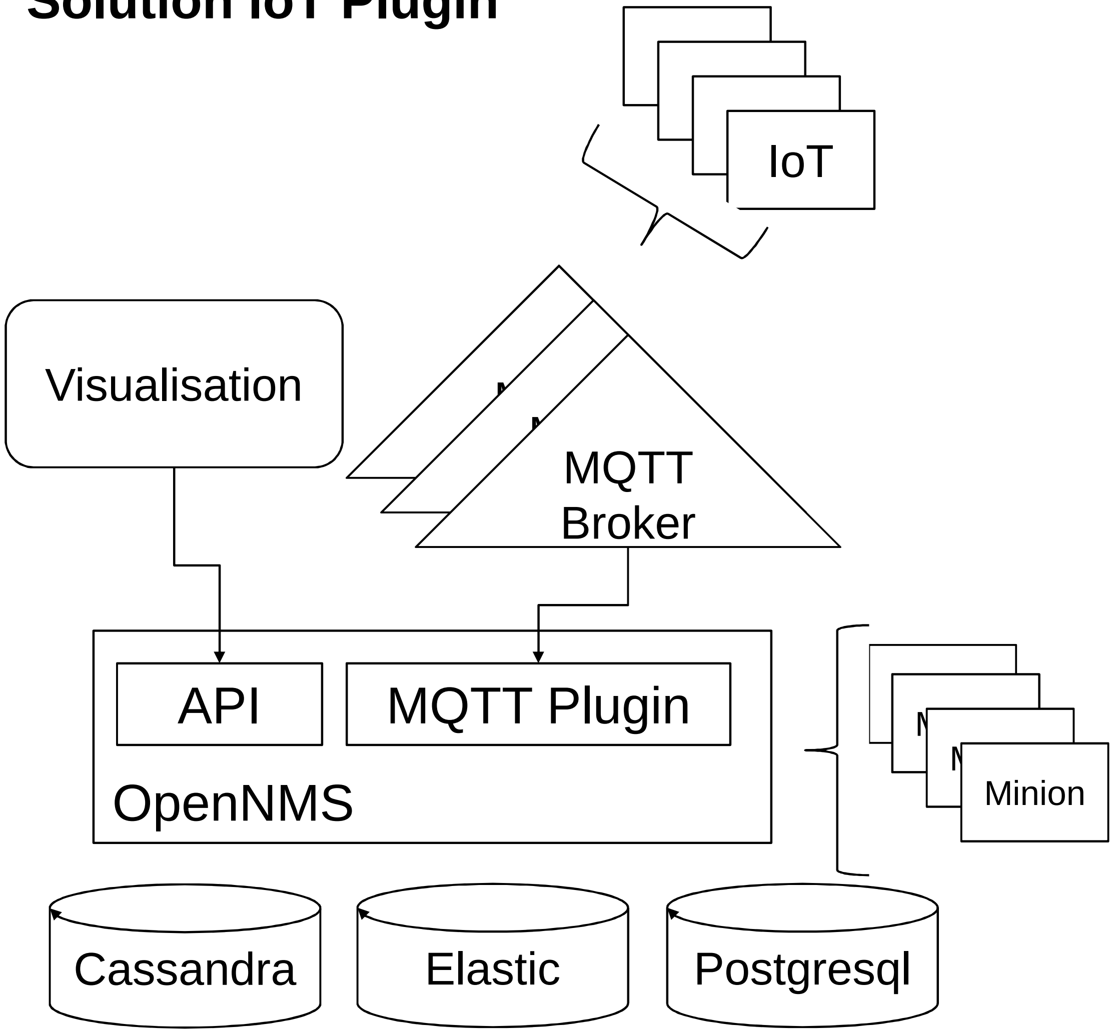
Why Use OpenNMS for IoT?

- **OpenNMS is**
  - Open Source
  - Proven in production
  - Highly Scalable
  - Can manage the infrastructure as well as collect data
  - Plugin Architecture based on OSGi

- **Cloud ready**
  - Grafana
  - Sophisticated Measurements API
  - Cassandra
  - Elastic Search
  - OpenNMS Minions – distributed OSGi based data collectors
  - OpenNMS Drift collects Streaming measurements Netflow 5,9, IPFIX and sFlow packages and parses them accordingly

- **OpenNMS users investigating IoT**

- **Solution IoT Plugin**

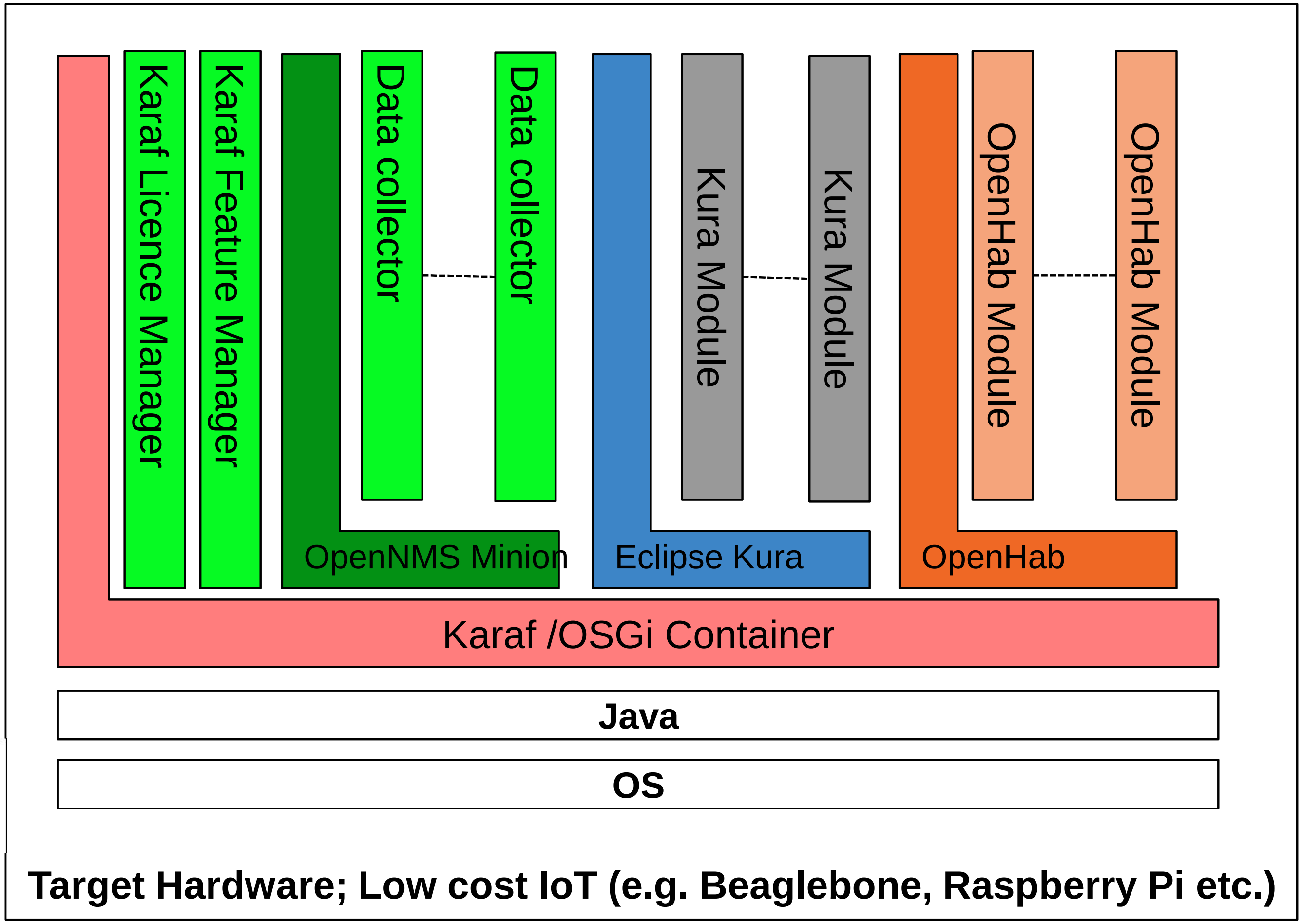


Data can be persisted in the cloud

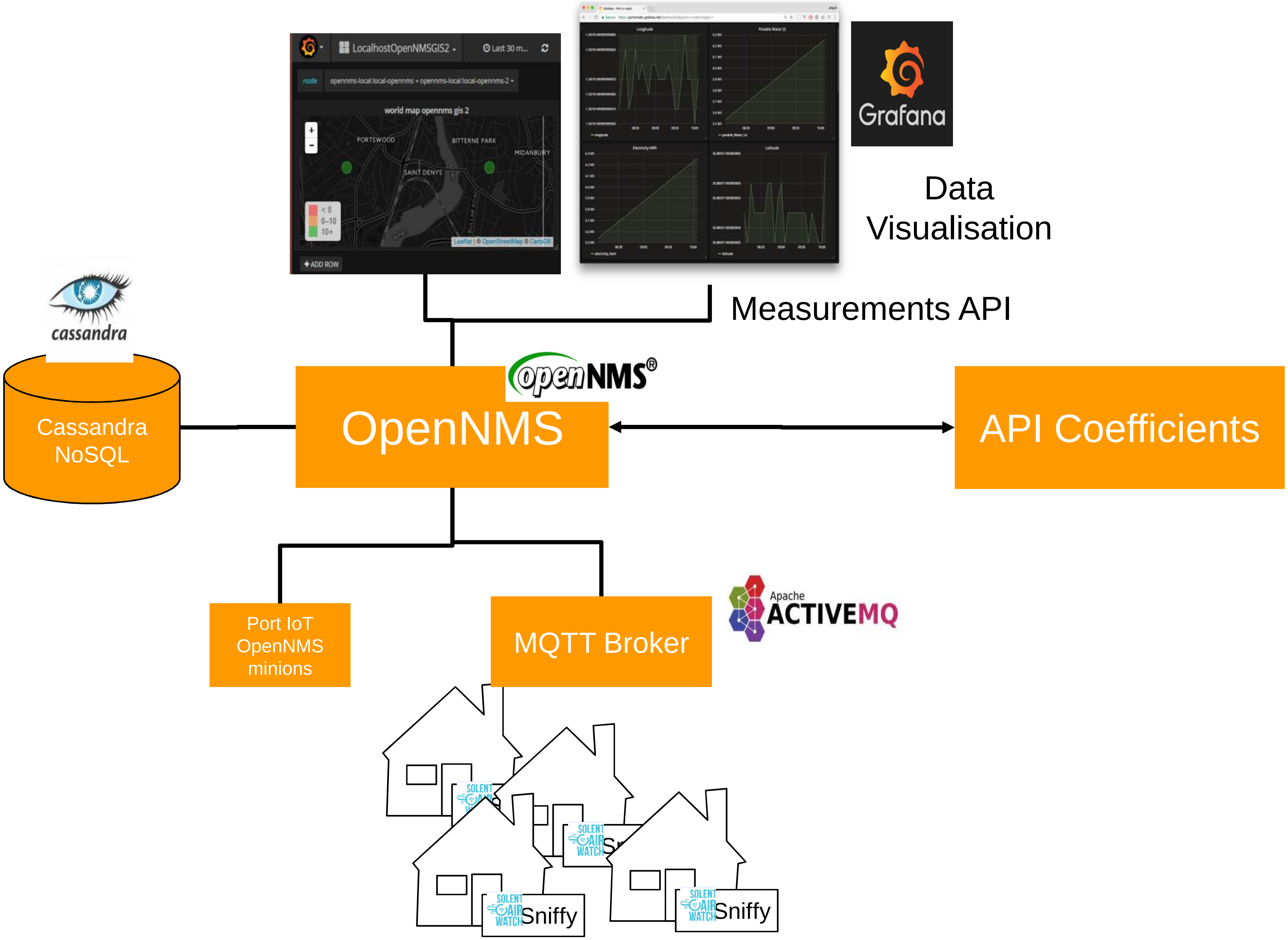
Java IoT Ecosystem

www.opennms.co.uk

- **Java IoT Ecosystem**
  - Apache Karaf
  - Eclipse Kura
  - OpenHab
- **OpenNMS Karaf Management**
  - Minions – remote data collectors
  - Karaf Feature manager
  - Karaf Licence manager
  - Karaf Plugin Manager
  - JMX Data collection



Architecture for IoT measurements



Data Visualisation

Measurements API

Cassandra NoSQL

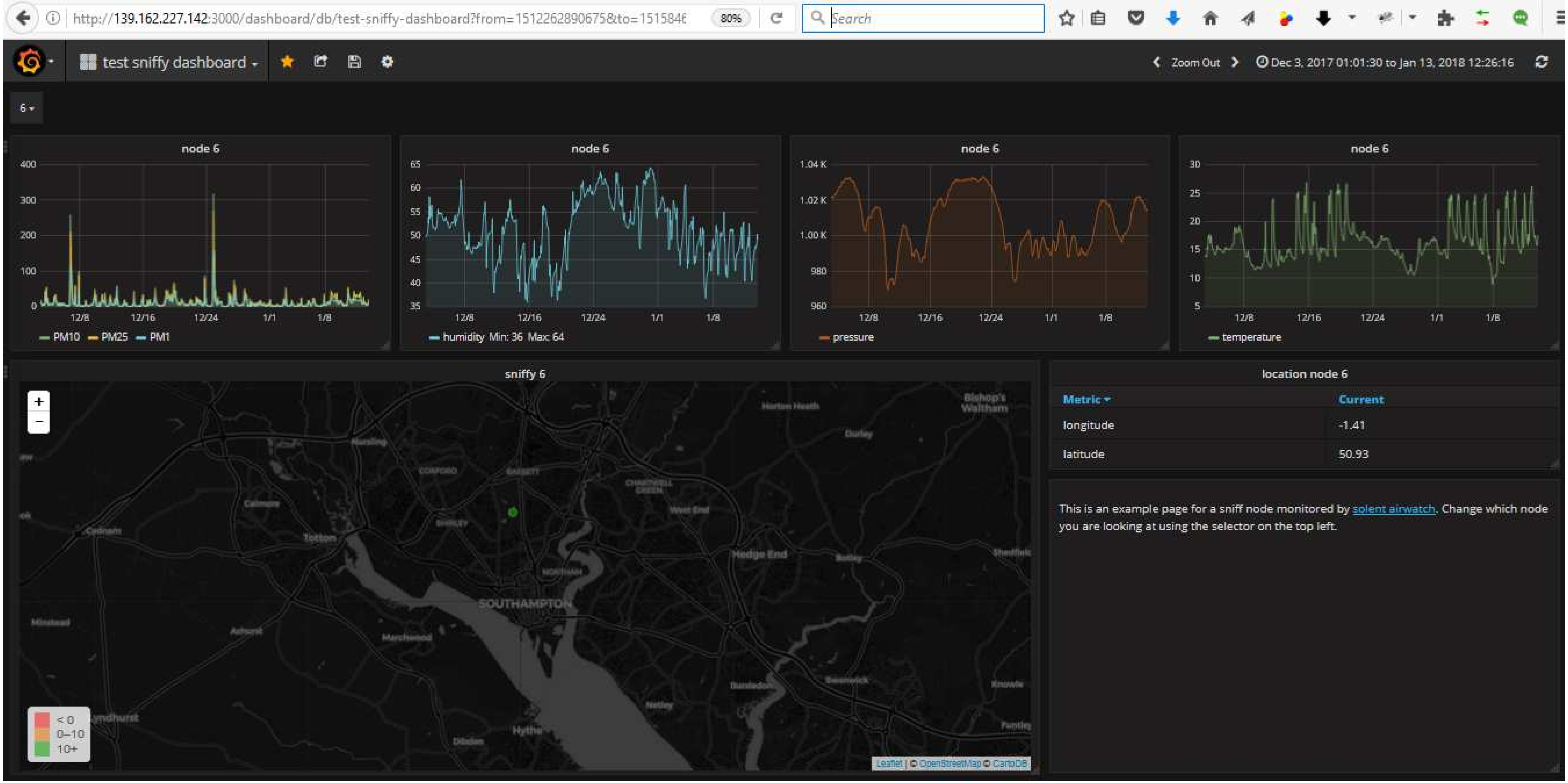
OpenNMS

API Coefficients

Port IoT OpenNMS minions

MQTT Broker

Pollution Measurements Dashboard

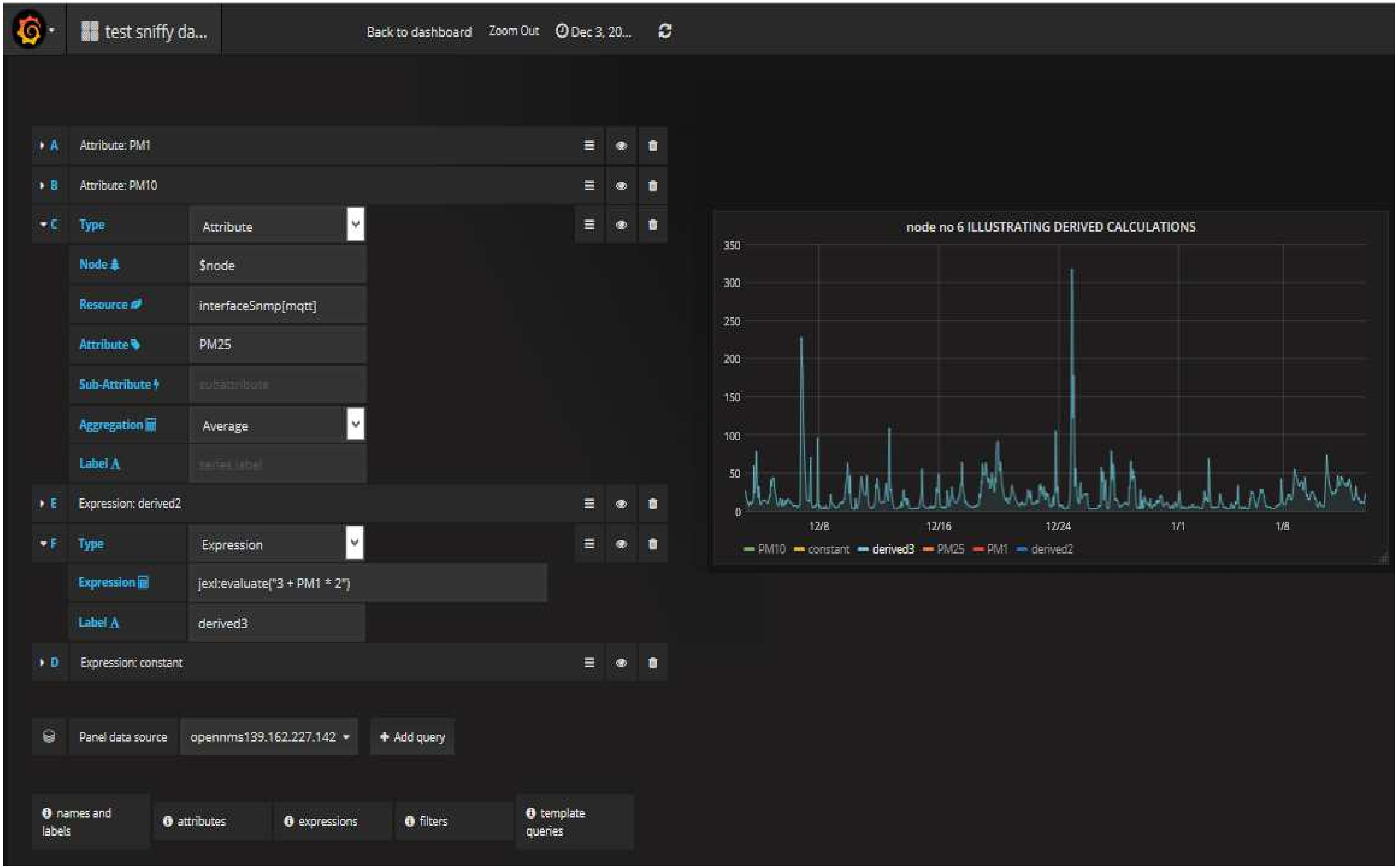


Metric	Current
longitude	-1.41
latitude	50.93

This is an example page for a sniff node monitored by solent airwatch. Change which node you are looking at using the selector on the top left.

Derived Calculations

www.opennms.co.uk



test sniffy da...

Back to dashboard | Zoom Out | Dec 3, 20...

- A Attribute: PM1
- B Attribute: PM10
- C
  - Type: Attribute
  - Node: $node
  - Resource: interfaceSnmp[mqtt]
  - Attribute: PM25
  - Sub-Attribute: subattribute
  - Aggregation: Average
  - Label: series label
- E Expression: derived2
- F
  - Type: Expression
  - Expression: jexl:evaluate("3 + PM1 * 2")
  - Label: derived3
- D Expression: constant

Panel data source: opennms139.162.227.142

OpenNMS Measurements API

Allows

- **Derived Values**
- **Constants**
- **Injected Formulae**
- **See**
  - https://docs.opennms.org/opennms/releases/latest/guide-development/guide-development.html#_measurements_api

Request

```
{
  "start": 1512262890675,
  "end": 1515846376323,
  "step": 2491992,
  "maxrows": 1438,
  "source": [
      {
        "aggregation": "AVERAGE",
        "attribute": "PM1",
        "label": "PM1",
        "resourceId": "nodeSource[mqtt:6].interfaceSnmp[mqtt]",
        "transient": "false"
      },
      {
        "aggregation": "AVERAGE",
        "attribute": "PM10",
        "label": "PM10",
        "resourceId": "nodeSource[mqtt:6].interfaceSnmp[mqtt]",
        "transient": "false"
      },
      {
        "aggregation": "AVERAGE",
        "attribute": "PM25",
        "label": "PM25",
        "resourceId": "nodeSource[mqtt:6].interfaceSnmp[mqtt]",
        "transient": "false"
      }
  ],
  "expression": [
      {
        "label": "derived3",
        "value": "jexl:evaluate(\"3 + PM1 * 2\")",
        "transient": "false"
      }
  ]
}
```

Reply

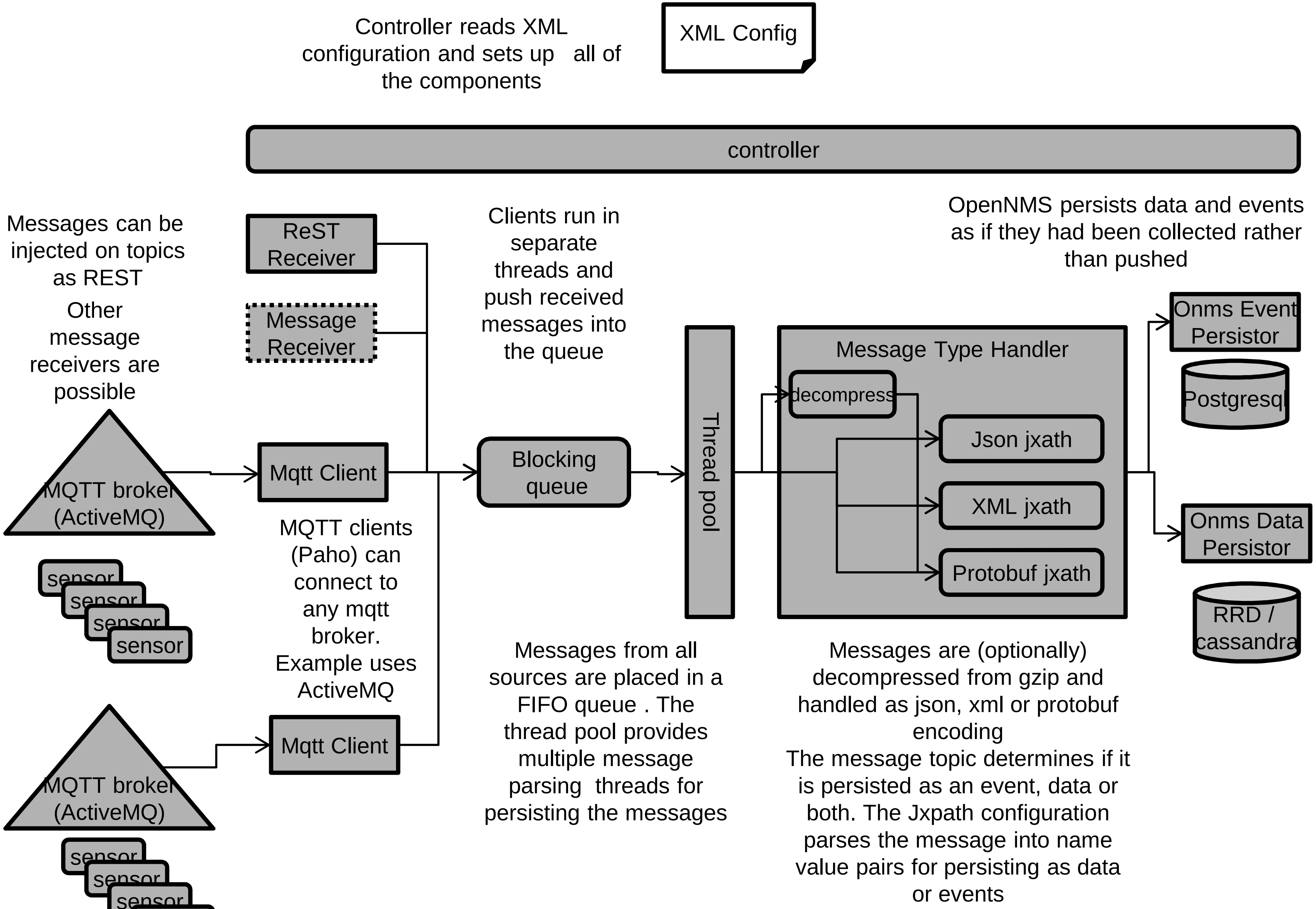
```
{
      "step": 3600000,
      "start": 1512262890675,
      "end": 1515846376323,
      "timestamps": [
          1512262800000,
          1512266400000,
               ...
          1515848400000
      ],
      "labels": [
          "PM10",
          "constant",
          "derived3",
          "PM25",
          "PM1",
          "derived2"
      ],
      "columns": [
          {
                 "values": [
                         30.582777777777775,
                         19.8075,
                         ...
                         12.172777777777773
                 ]
          },
          {
                 "values": [
                         1234.0,
                         1234.0,
                         ...
                         1234.0
                 ]
          },

         ........

          {      "values": [
                         18.196388888888887,
                         12.221944444444443,
                         ...
                         9.638888888888891
                 ]
          }
      ],
      "constants": [
          {
                 "key": "PM25.constant1",
                 "value": "1200.0"
          },
          {
                 "key": "PM10.formula1",
                 "value": "PM25.constant1+PM10"
          }
      ]
}
```

MQTT Plugin Architecture

- https://github.com/gallenc/opennms-mqtt-plugin



Controller reads XML configuration and sets up all of the components

XML Config

controller

Messages can be injected on topics as REST

Other message receivers are possible

ReST Receiver

Message Receiver

Clients run in separate threads and push received messages into the queue

OpenNMS persists data and events as if they had been collected rather than pushed

MQTT broker (ActiveMQ)

sensor
sensor
sensor
sensor

Mqtt Client

Blocking queue

Thread pool

Message Type Handler

decompress

Json jxath

XML jxath

Protobuf jxath

Onms Event Persistor

Postgresql

Onms Data Persistor

RRD / cassandra

MQTT clients (Paho) can connect to any mqtt broker. Example uses ActiveMQ

MQTT broker (ActiveMQ)

sensor
sensor
sensor
sensor

Mqtt Client

Messages from all sources are placed in a FIFO queue . The thread pool provides multiple message parsing threads for persisting the messages

Messages are (optionally) decompressed from gzip and handled as json, xml or protobuf encoding
The message topic determines if it is persisted as an event, data or both. The Jxpath configuration parses the message into name value pairs for persisting as data or events

OpenNMS MQTT plugin

MQTT Message payloads

- **Payloads Currently Supported**
  - Fully configurable for
    - Json
    - XML
    - Line separated CSV
    - Compressed (gzip) and uncompressed
    - Configurable parser based on j-xpath
  - Parser configurations for
    - Eclipse Kura / Kapura K-Payload
      - https://github.com/eclipse/kapua/wiki/K-Payload-JSON-Format
    - Sniffy Json protocol
  - Google Protobuf being worked on

- **Payloads can be parsed into**
  - OpenNMS events
  - OpenNMS Performance Measurements

Simple Sniffy format

```
{
        "time": "2017-10-19 10:15:02.854888",
        "id": "monitorID",
        "cityName": "Southampton",
        "stationName": "Common#1",
        "latitude": 0,
        "longitude": 0,
        "averaging": 0,
        "PM1": 10,
        "PM25": 100,
        "PM10": 1000
}
```

Kura Format (Simple)

```
{
 "sentOn" : 1491298822,
 "position" : {
  "latitude" : 45.234,
  "longitude" : -7.3456,
  "altitude" : 1.0,
  "heading" : 5.4,
  "precision" : 0.1,
  "speed" : 23.5,
  "timestamp" : 1191292288,
  "satellites" : 3,
  "status" : 2
 },
 "metrics": {
  "code" : "A23D44567Q",
  "distance" : 0.26456E+4,
  "temperature" : 27.5,
  "count" : 12354,
  "timestamp" : 23412334545,
  "enable" : true,
  "rawBuffer" : "cGlwcG8gcGx1dG8gcGFwZXJpbm8="
 },
 "body": "UGlwcG8sIHBsdXRvLCBwYXBlcmlubywgcXVpLCBxdW8gZSBxdWEu"
}
```

Where is the code?

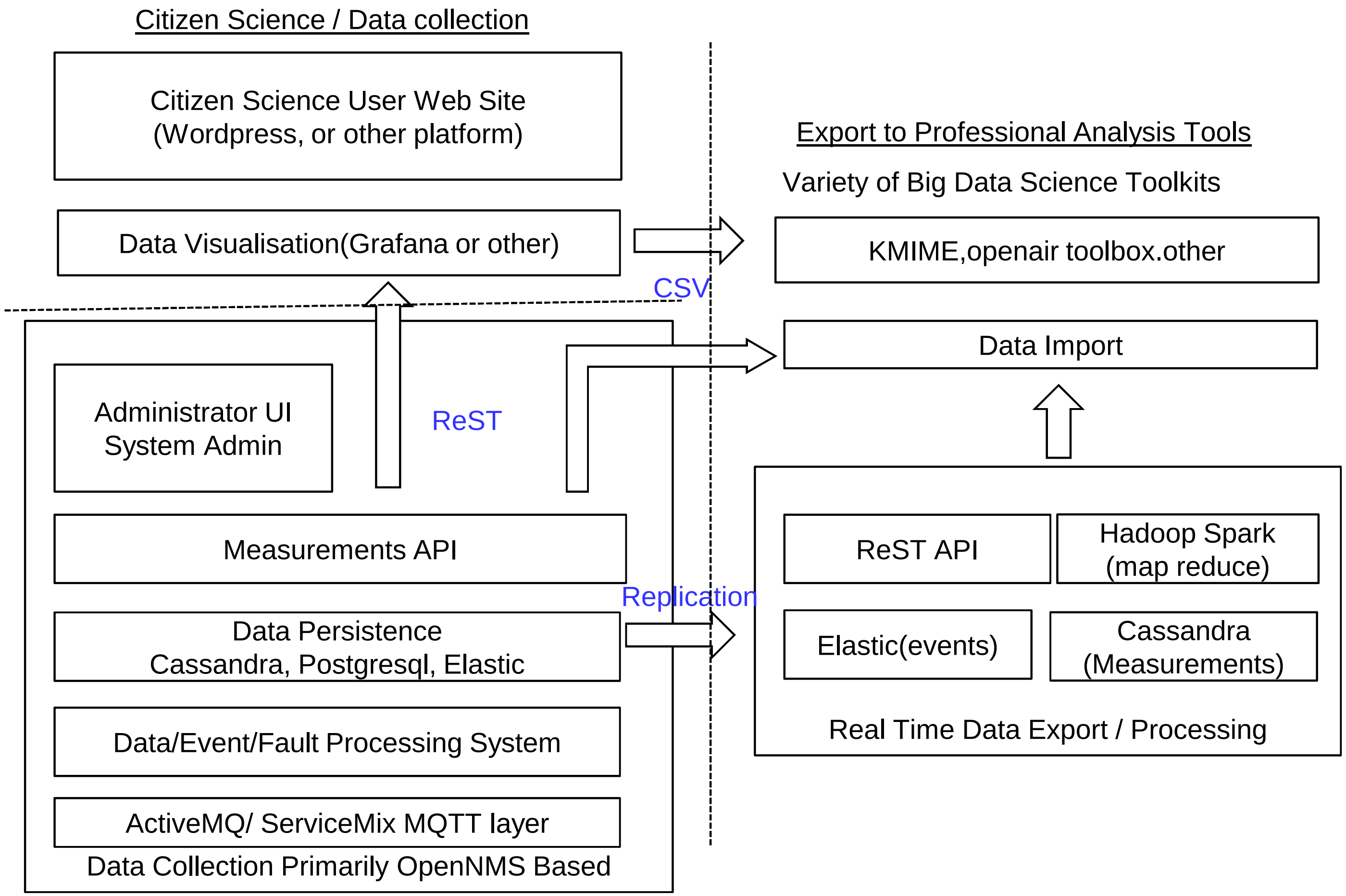
- **OpenNMS**
  - available at www.opennms.org

- **MQTT adaptor Project**
  - https://github.com/gallenc/opennms-mqtt-plugin
  - Plugin code
  - Docker-Compose
    - Configuration for whole solution with Cassandra, OpenNMS, Postgresql, Grafana and an ActiveMQ MQTT broker

- **Sniffy Version 1**
  - https://github.com/SolentAirWatch/sniffy

Next Steps

- **Migrate Sniffy to Eclipse Kura**
- **Build supporting 'citizen science' web site infrastructure**
- **Build supporting 'professional science' infrastructure for analysing measurements at scale**
- **Build separate MQTT plugin into OpenNMS**



Citizen Science / Data collection

Citizen Science User Web Site (Wordpress, or other platform)

Data Visualisation(Grafana or other)

Administrator UI System Admin

ReST

Measurements API

Data Persistence Cassandra, Postgresql, Elastic

Data/Event/Fault Processing System

ActiveMQ/ ServiceMix MQTT layer

Data Collection Primarily OpenNMS Based

CSV

Replication

Export to Professional Analysis Tools

Variety of Big Data Science Toolkits

KMIME,openair toolbox.other

Data Import

ReST API

Hadoop Spark (map reduce)

Elastic(events)

Cassandra (Measurements)

Real Time Data Export / Processing