# API Days Nordic 2016 Tampere

Mika Karaila

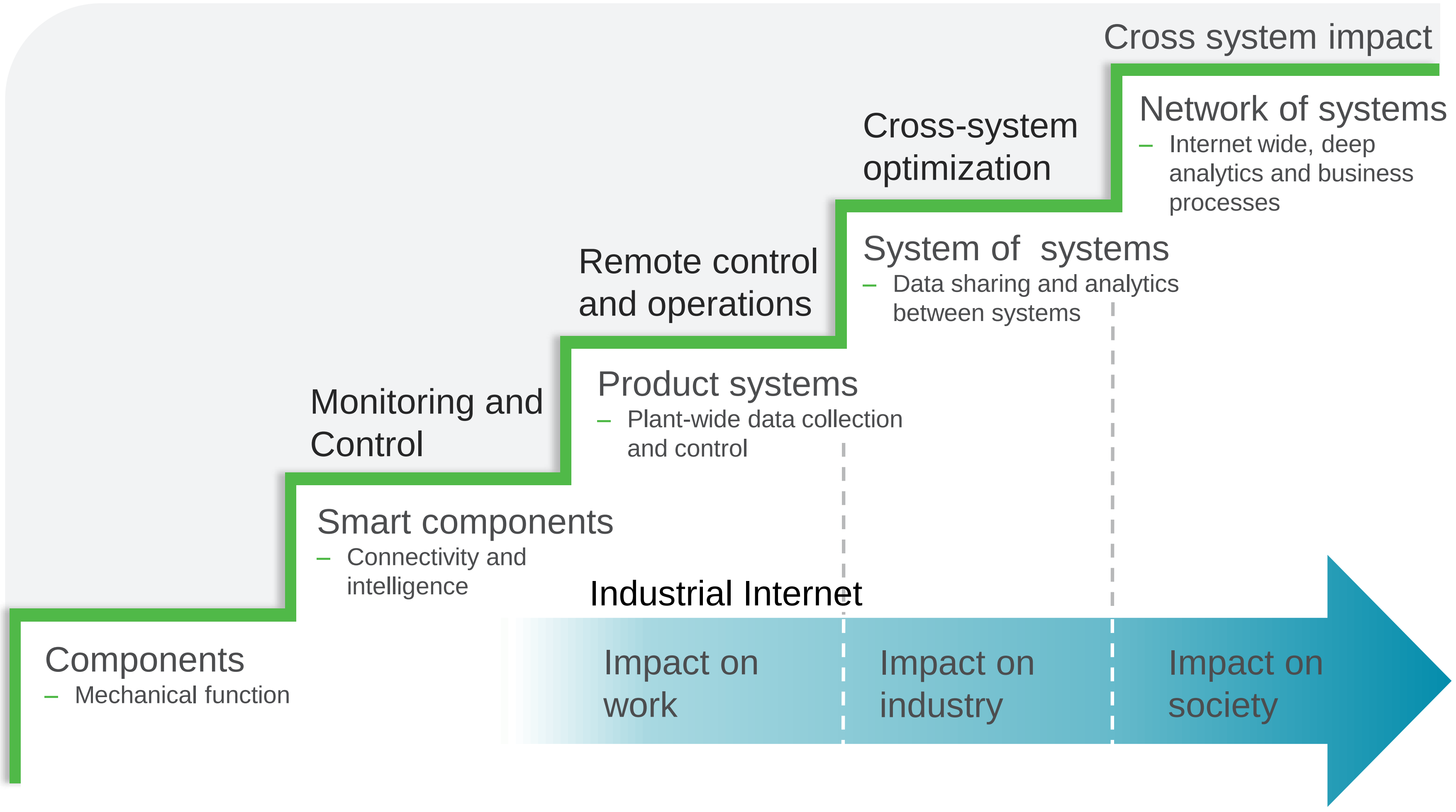
Research manager

Valmet Automation

## Contents

# Importance of APIs for Industrial Internet

# Evolution to Industrial Internet

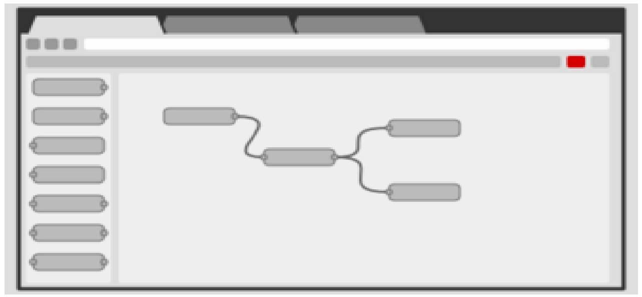

Cross system impact
Network of systems
– Internet wide, deep analytics and business processes
Cross-system optimization
System of systems
– Data sharing and analytics between systems
Remote control and operations
Product systems
– Plant-wide data collection and control
Monitoring and Control
Smart components
– Connectivity and intelligence
Components
– Mechanical function
Industrial Internet
Impact on work
Impact on industry
Impact on society

Valmet

## Visual flow programming

# Node-red

## Visual programming



### Browser-based flow editing

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range nodes in the palette. Flows can be then deployed to the runtime in a single-click.

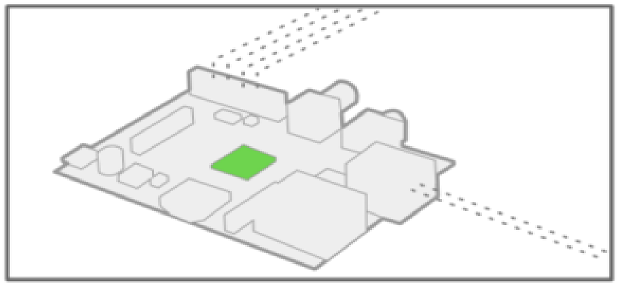
JavaScript functions can be created within the editor using a rich text editor.

A built-in library allows you to save useful functions, templates or flows for re-use.

### Built on Node.js

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

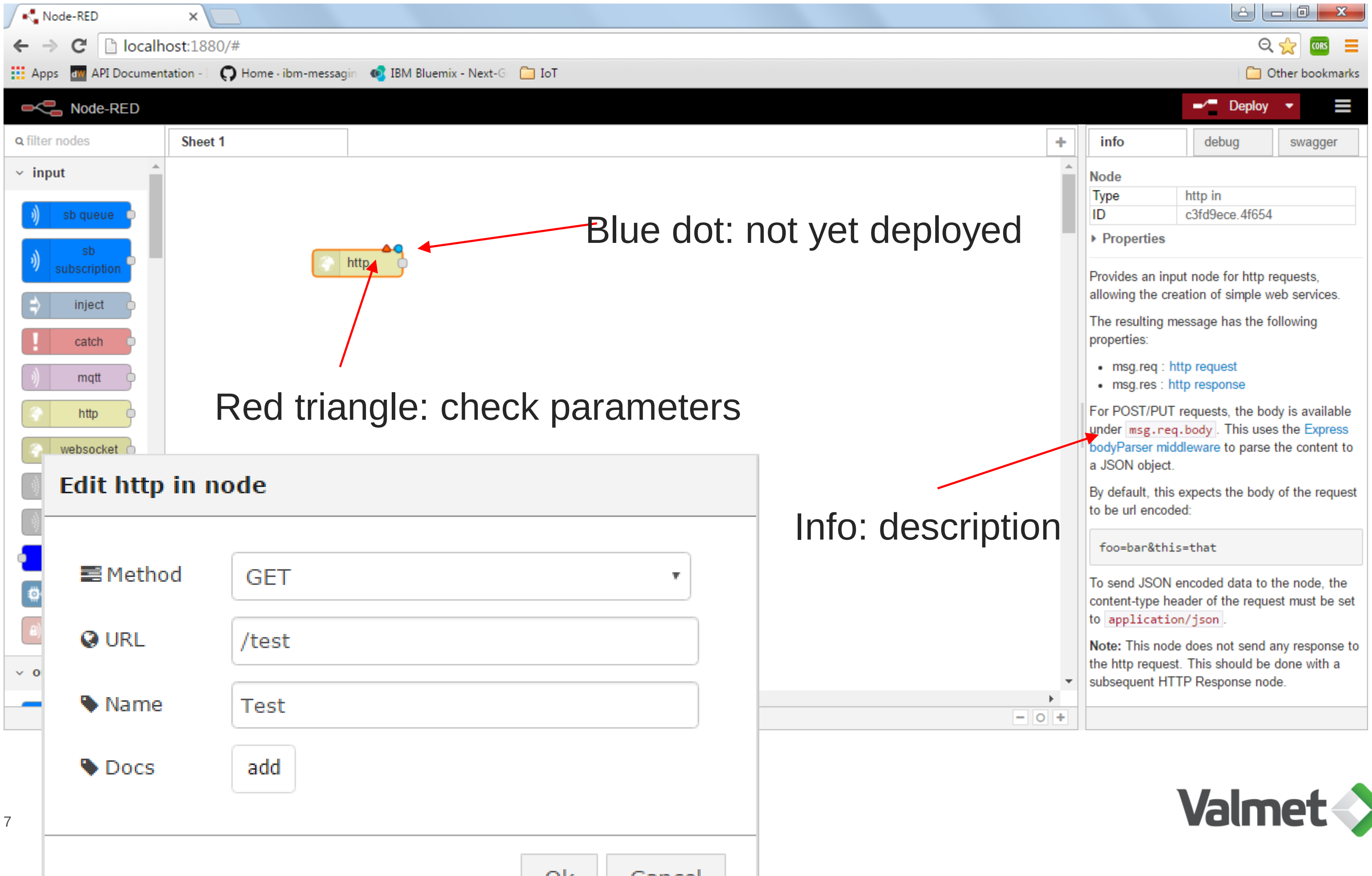
With over 225,000 modules in Node's package repository, it is easy to extend the range of palette nodes to add new capabilities.



http://nodered.org/

# Node-red principles

## Add node, edit parameters



**Node**

| | |
|---|---|
| Type | http in |
| ID | c3fd9ece.4f654 |

▸ Properties

Provides an input node for http requests, allowing the creation of simple web services.

The resulting message has the following properties:

- msg.req : http request
- msg.res : http response

For POST/PUT requests, the body is available under `msg.req.body`. This uses the Express bodyParser middleware to parse the content to a JSON object.

By default, this expects the body of the request to be url encoded:

```
foo=bar&this=that
```

To send JSON encoded data to the node, the content-type header of the request must be set to `application/json`.

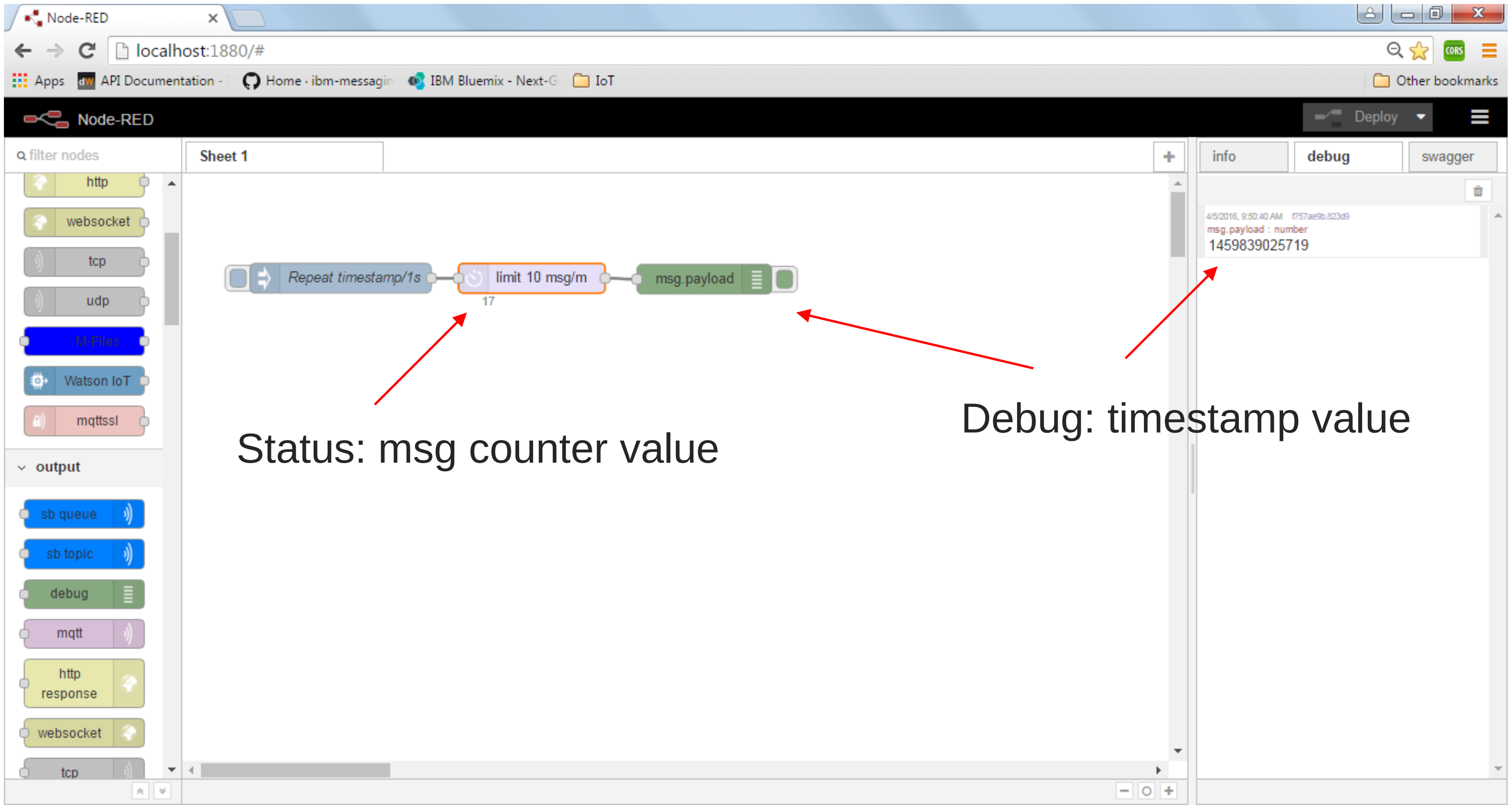
**Note:** This node does not send any response to the http request. This should be done with a subsequent HTTP Response node.

**Edit http in node**

| | |
|---|---|
| Method | GET |
| URL | /test |
| Name | Test |
| Docs | add |

# Node-red example

## Nodes: Inject, limit, debug

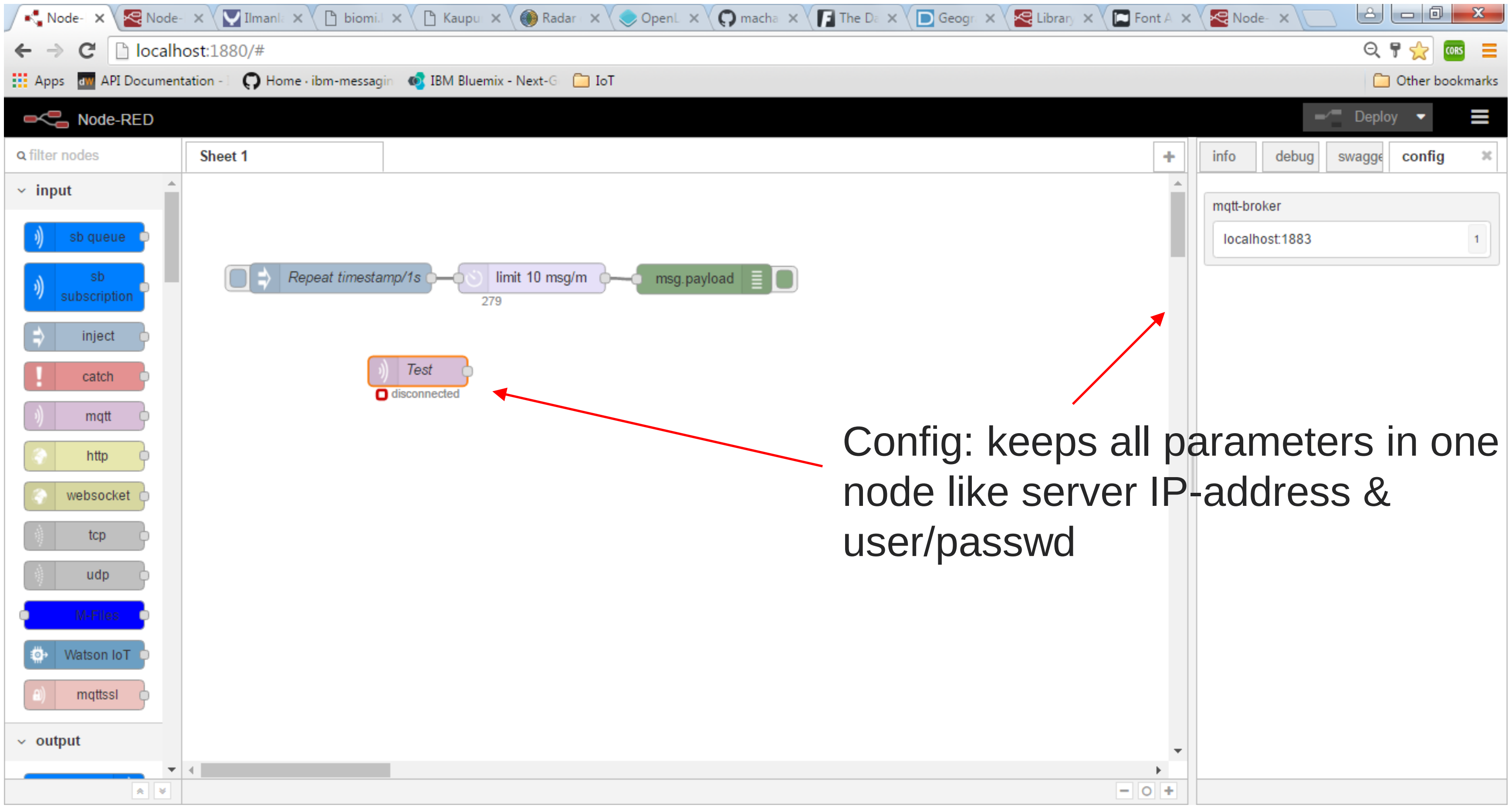


```
4/5/2016, 9:50:40 AM   f757ae9b.823d9
msg.payload : number
1459839025719
```

Status: msg counter value

Debug: timestamp value

# Server parameters in one node



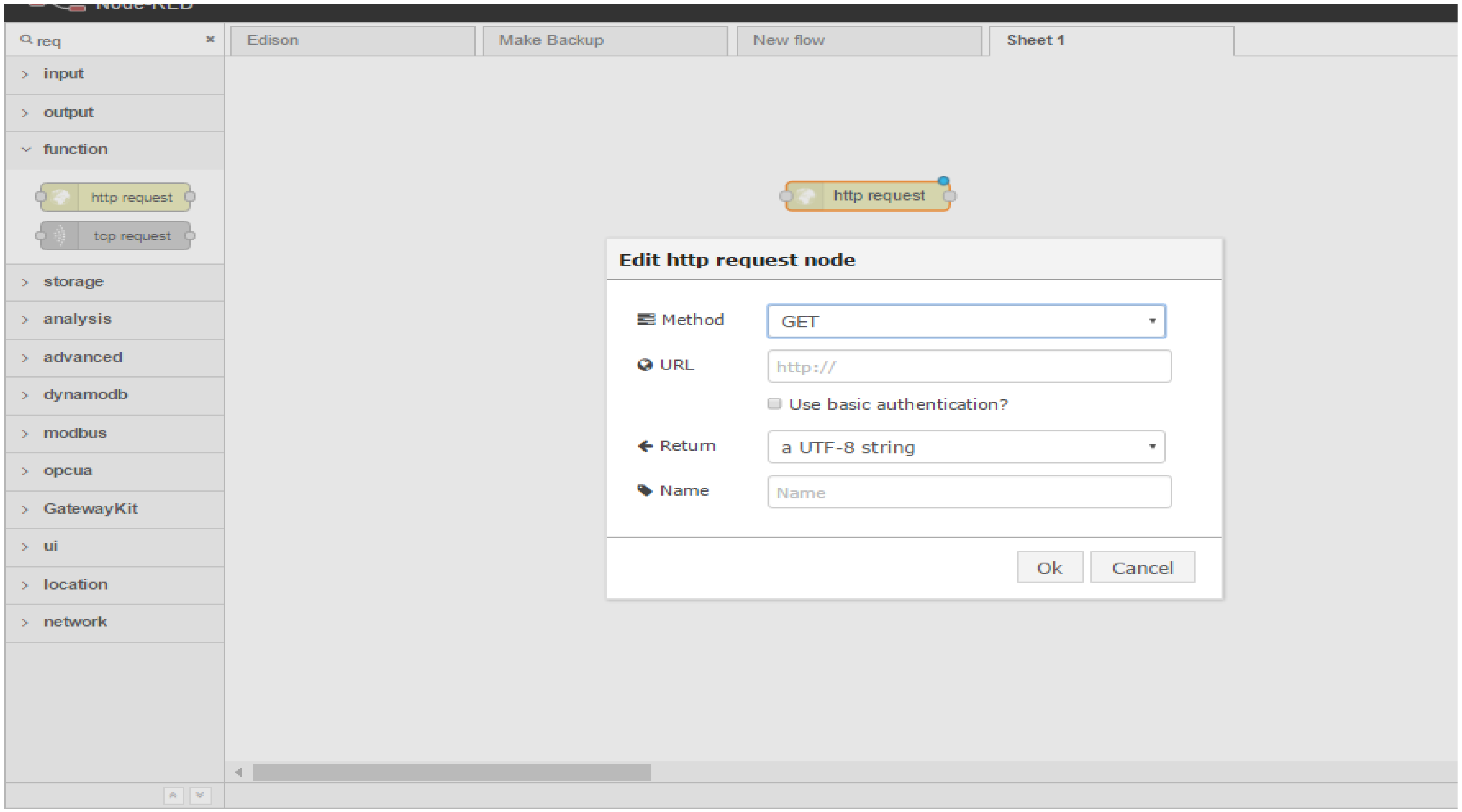
Config: keeps all parameters in one node like server IP-address & user/passwd

## Using REST APIs and Swagger

INTERNAL

## Creating REST API

### http request node



Edit http request node

Method: GET

URL: http://

Use basic authentication?

Return: a UTF-8 string

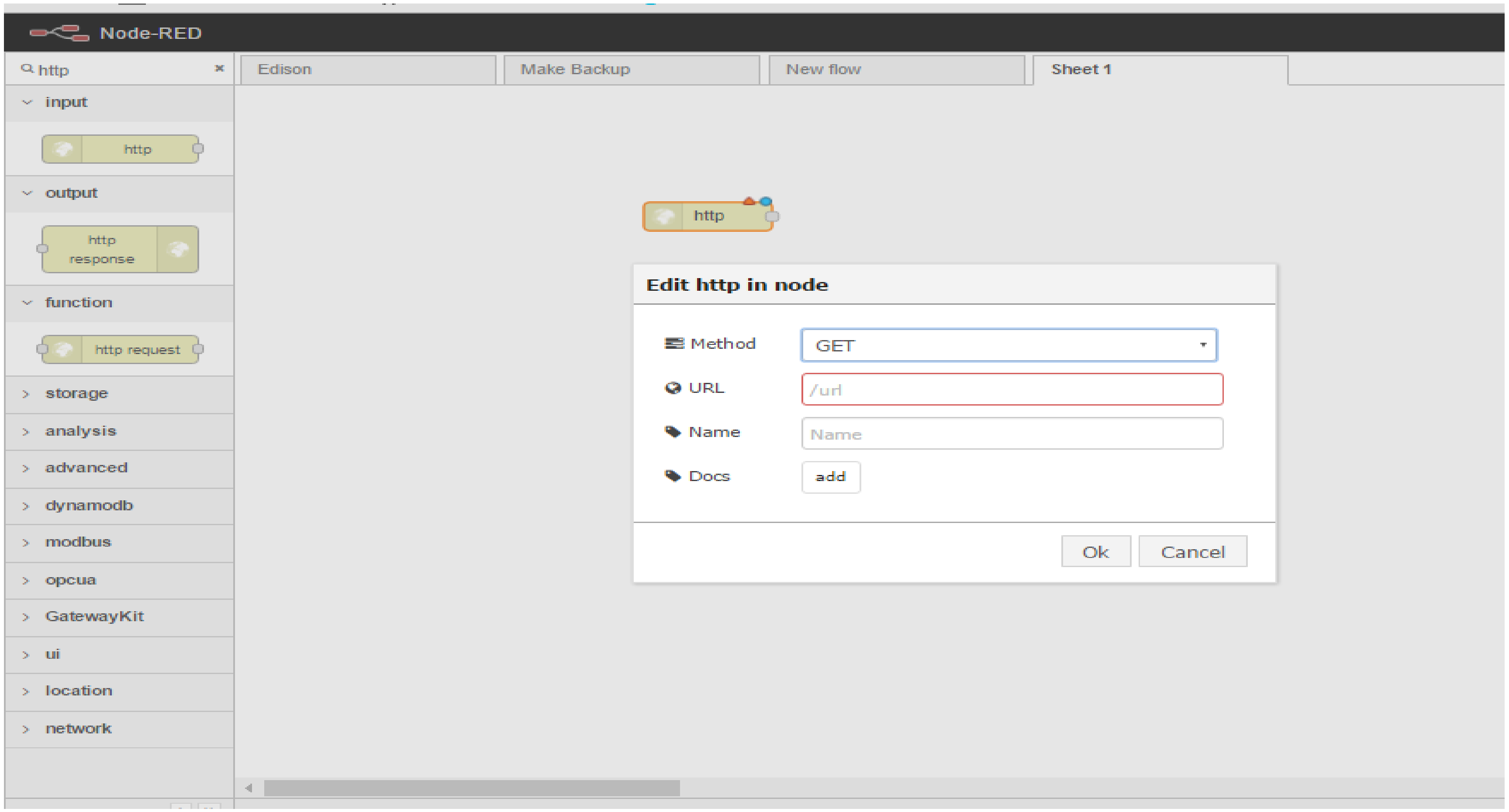
Name: Name

Ok Cancel

Valmet

# HTTP GET

## Defining endpoint

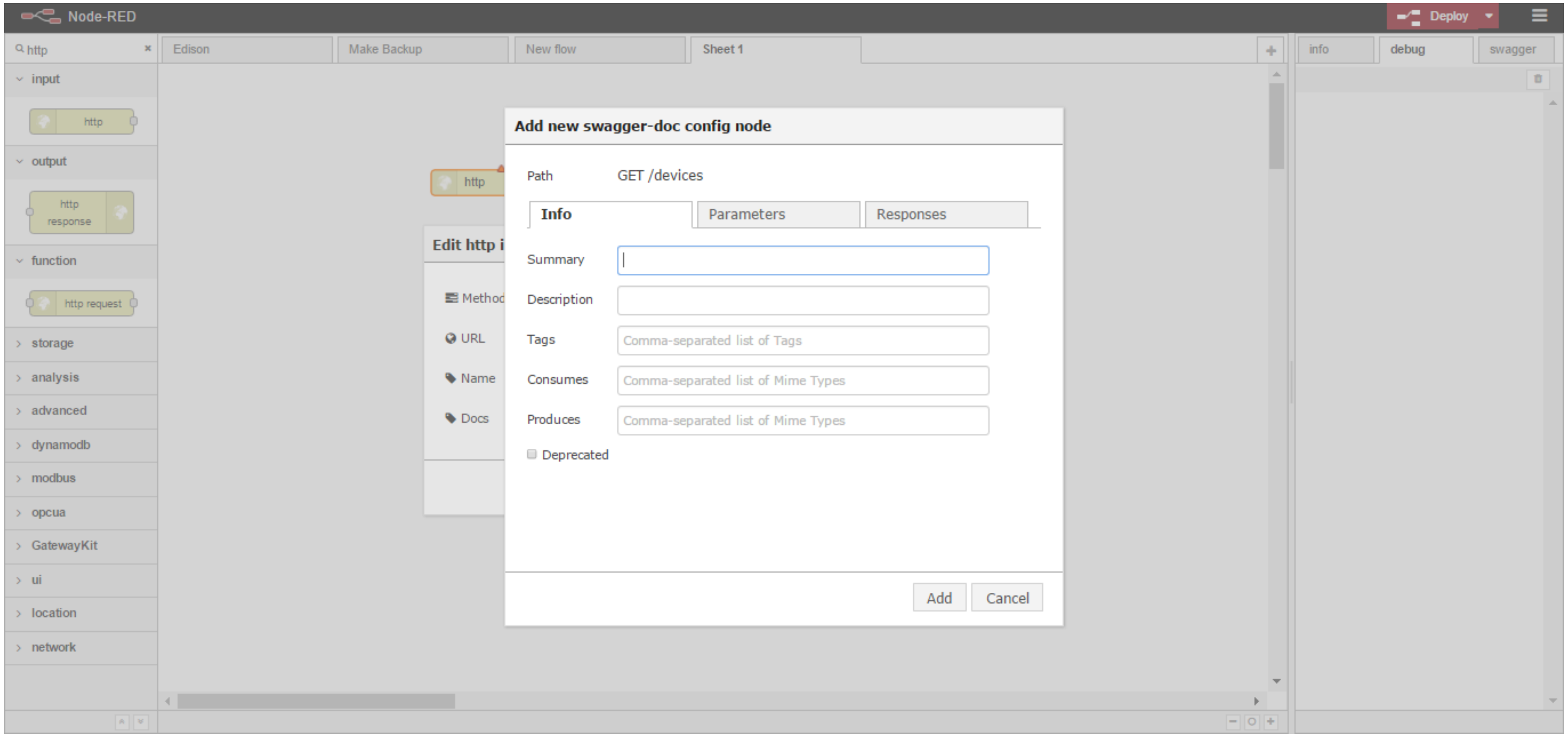


**Edit http in node**

| | |
|---|---|
| Method | GET |
| URL | /url |
| Name | Name |
| Docs | add |

# Documenting endpoint

## Description



Add new swagger-doc config node

Path GET /devices

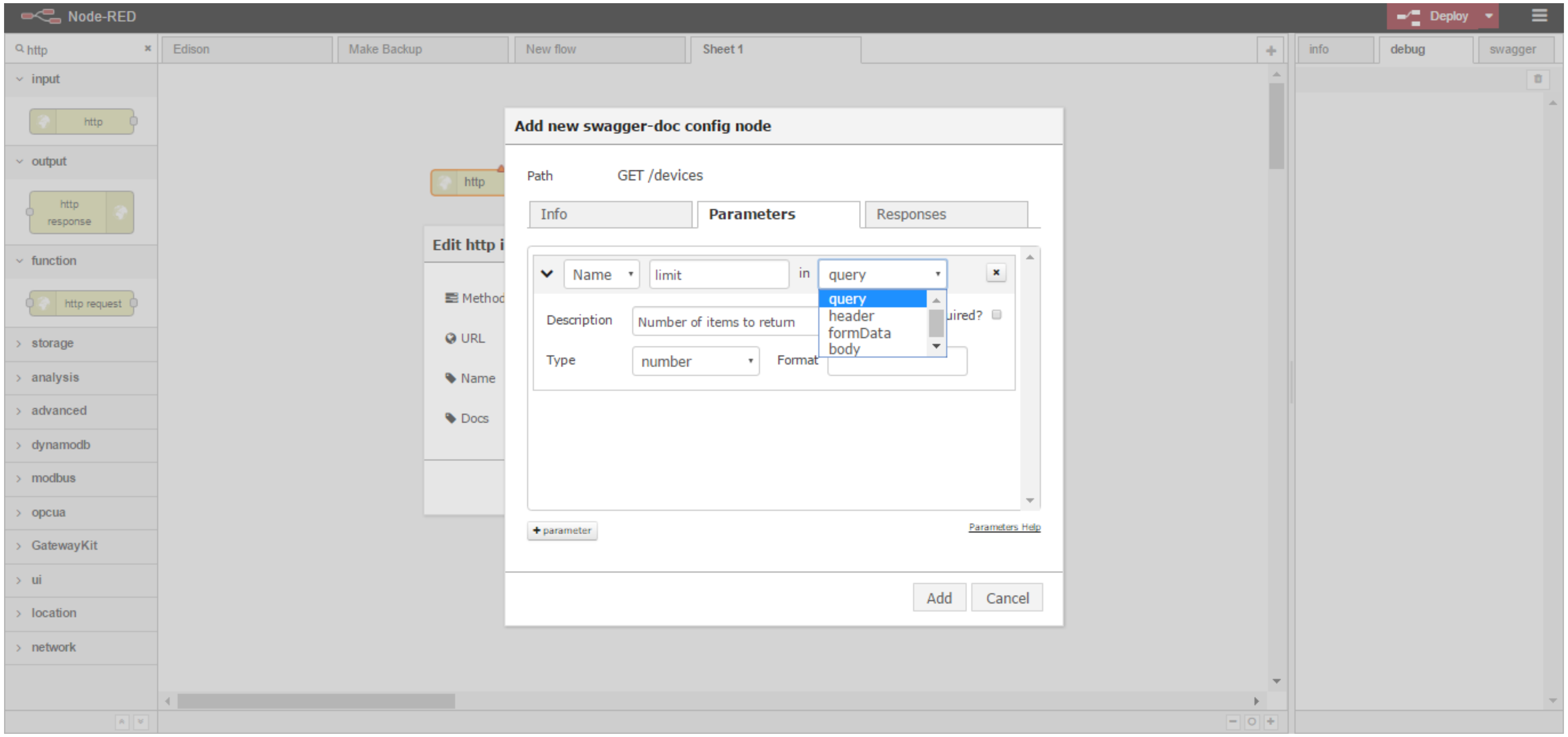
Info | Parameters | Responses

| | |
|---|---|
| Summary | |
| Description | |
| Tags | Comma-separated list of Tags |
| Consumes | Comma-separated list of Mime Types |
| Produces | Comma-separated list of Mime Types |

☐ Deprecated

Add Cancel

# Parameters for endpoint



Add new swagger-doc config node

Path GET /devices

Info | Parameters | Responses

Name: limit in query

query
header
formData
body

Description: Number of items to return

Type: number

Format

Required?

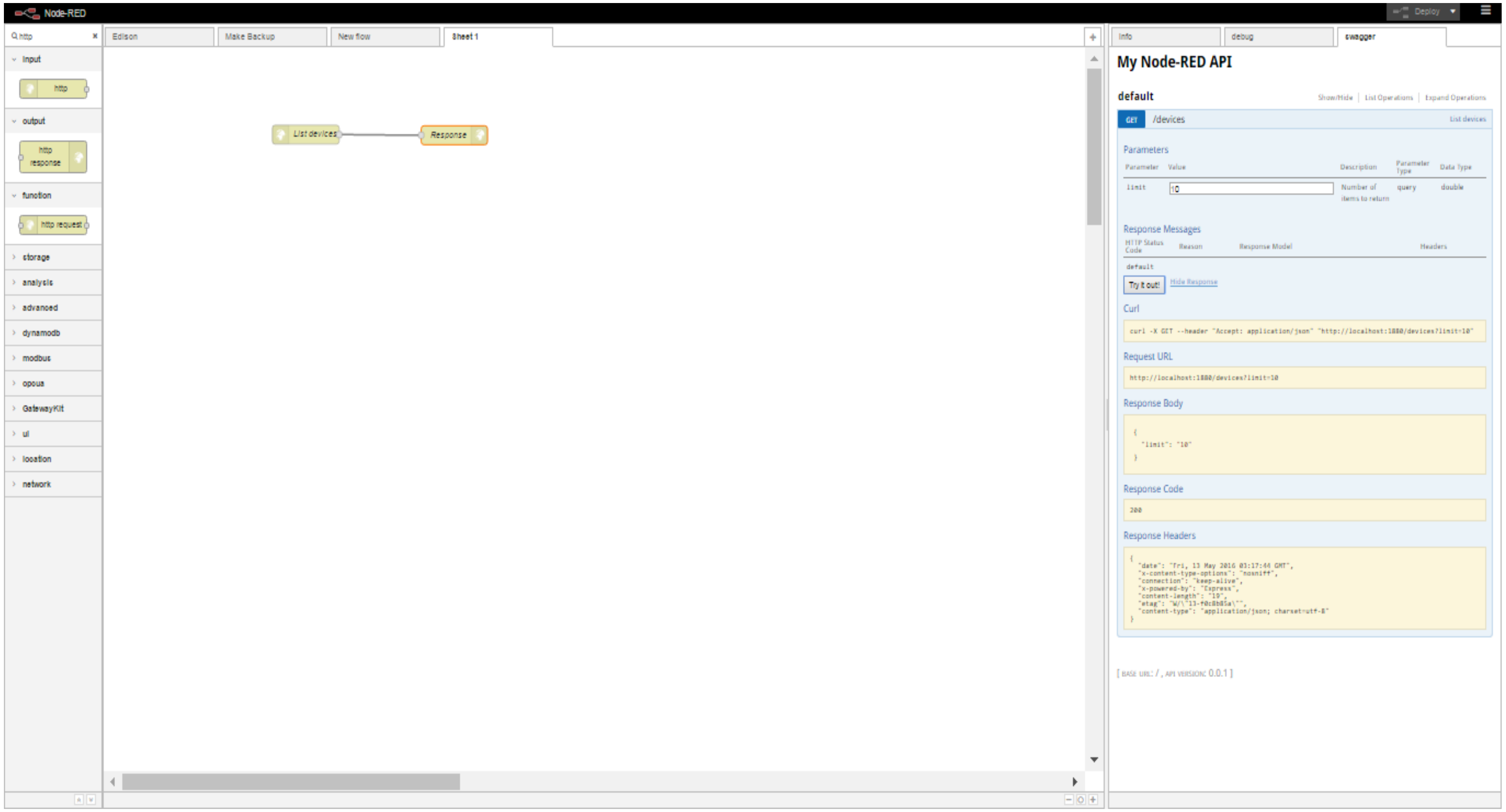
+ parameter

Parameters Help

Add Cancel

# Deploy & use Swagger

## Swagger integrated



My Node-RED API

default

GET /devices — List devices

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|---|---|---|---|---|
| limit | 10 | Number of items to return | query | double |

Response Messages

| HTTP Status Code | Reason | Response Model | Headers |
|---|---|---|---|
| default | | | |

Curl

```
curl -X GET --header "Accept: application/json" "http://localhost:1880/devices?limit=10"
```

Request URL

```
http://localhost:1880/devices?limit=10
```

Response Body

```
{
  "limit": "10"
}
```

Response Code

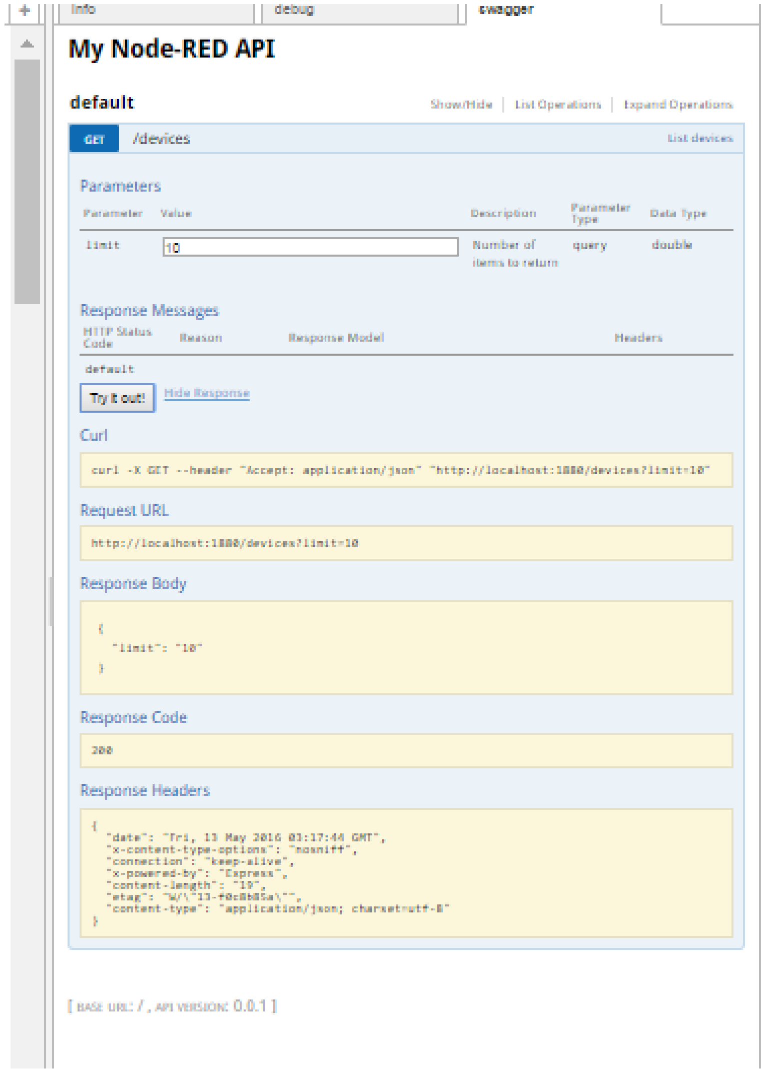
```
200
```

Response Headers

```
{
  "date": "Fri, 13 May 2016 03:17:44 GMT",
  "x-content-type-options": "nosniff",
  "connection": "keep-alive",
  "x-powered-by": "Express",
  "content-length": "19",
  "etag": "W/\"13-f0c8b85a\"",
  "content-type": "application/json; charset=utf-8"
}
```

[ BASE URL: / , API VERSION: 0.0.1 ]

# Ready to use & test



My Node-RED API

default

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|---|---|---|---|---|
| limit | 10 | Number of items to return | query | double |

Response Messages

| HTTP Status Code | Reason | Response Model | Headers |
|---|---|---|---|
| default | | | |

Curl

```
curl -X GET --header "Accept: application/json" "http://localhost:1880/devices?limit=10"
```

Request URL

```
http://localhost:1880/devices?limit=10
```

Response Body

```
{
  "limit": "10"
}
```

Response Code

```
200
```

Response Headers

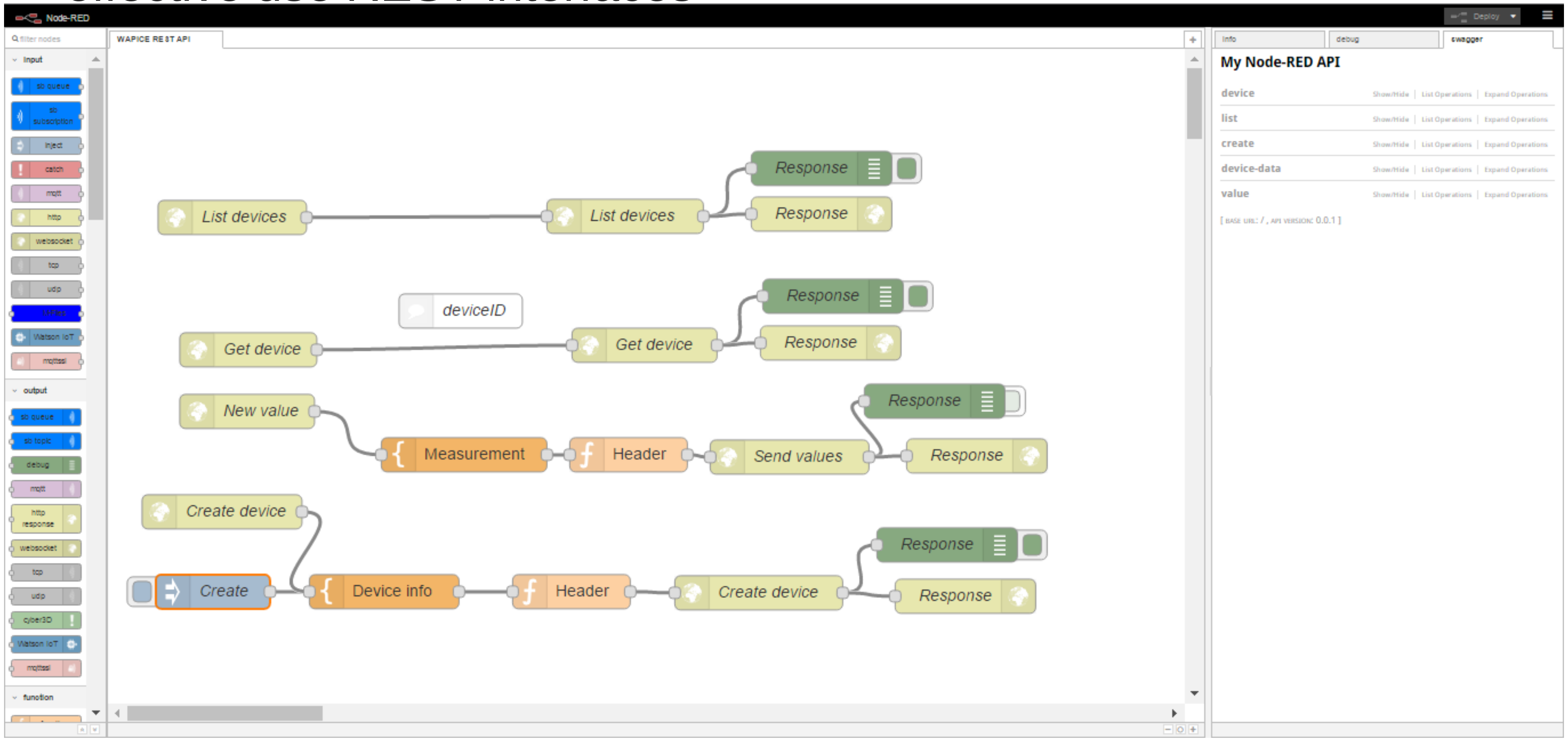
```
{
  "date": "Fri, 13 May 2016 03:17:44 GMT",
  "x-content-type-options": "nosniff",
  "connection": "keep-alive",
  "x-powered-by": "Express",
  "content-length": "19",
  "etag": "W/\"13-f0c8b85a\"",
  "content-type": "application/json; charset=utf-8"
}
```

[ BASE URL: / , API VERSION: 0.0.1 ]

# DEMO

## Live demo with Wapice IoT-Ticket: swagger.json

- Demonstrates how to use node-red with swagger to simplify and effective use REST interfaces



## Node-red meta programming

## Meta-programming

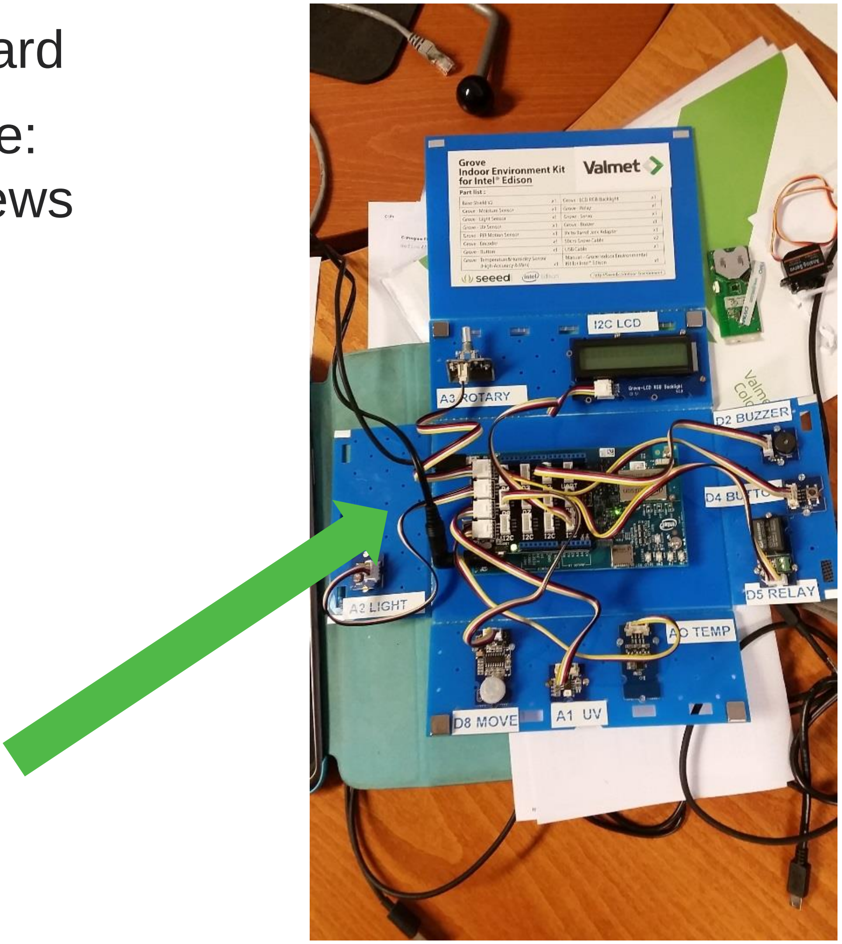
Program that creates a program

- Node-red supports admin level REST interface
  - Endpoint to GET / POST to access flows

- Ideal for managing & testing larger amount of nodes

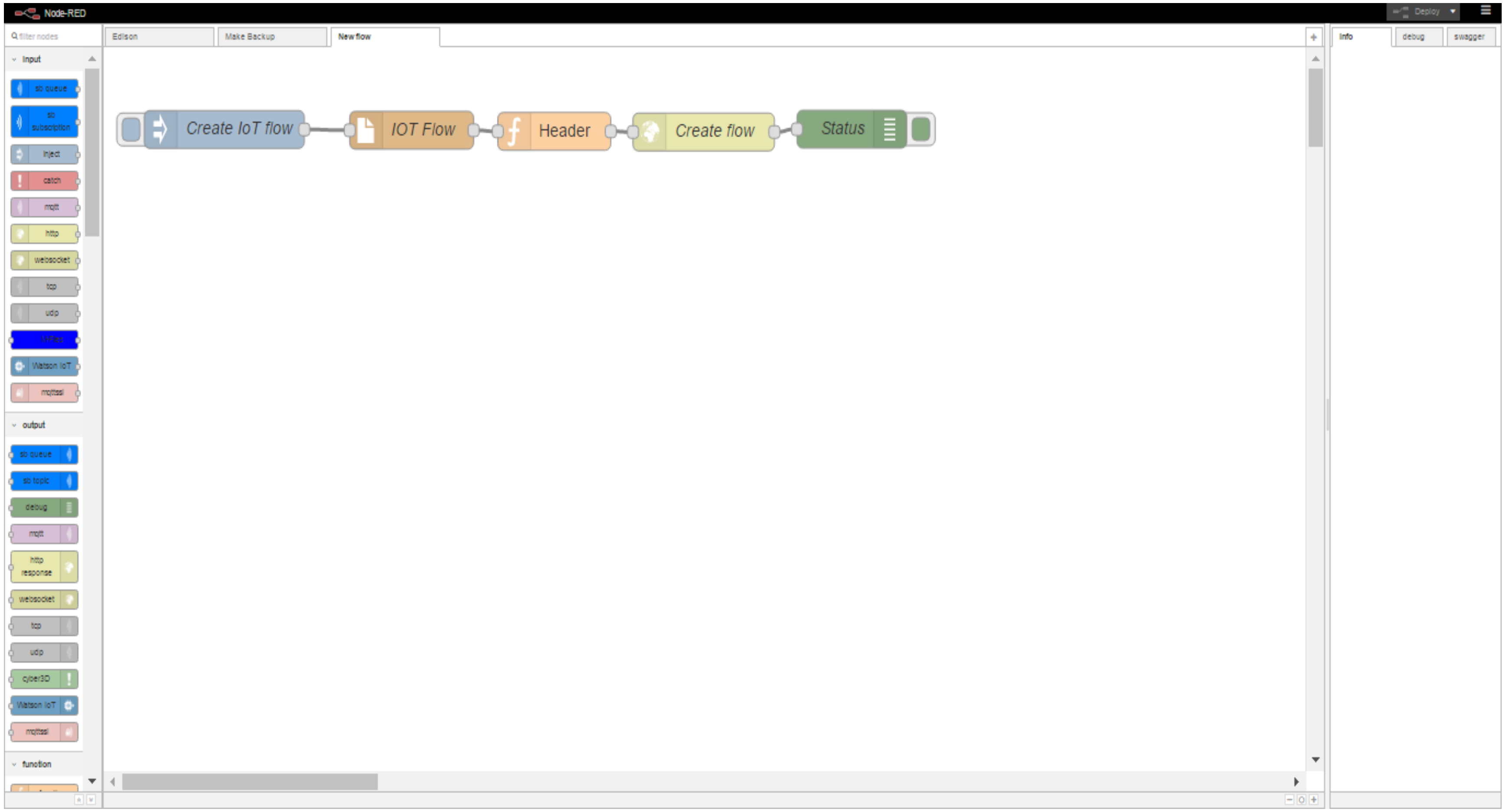
# Demo hardware: Intel Edison + Groove kit

1. Intel Edison into base board
2. Base board into enclosure:
   4 x metal lifters + 8 x screws
3. Arduino sensor adapter
4. Sensors:
   2 x screws + 2 x nuts
   4 x washers



Leave space for connectors:
Power line + USB-debug

## Simple flow that will send “program” to Edison



## Demo summary

- Same programming environment
- Simple REST APIs
- Effective and very easy to reuse => high productivity

Valmet

## Summary

INTERNAL

## Highlights

- Document as you program
- Test immediately
- Easiness & productivity
- Have fun !

**Questions & discussion**