IBM Watson IoT – T5 – Cloudant

Commissioning task T5

1. Introduction

IBM Cloudant is a fully managed JSON document database. Cloudant is compatible with Apache CouchDB and accessible through a simple to use HTTPS API for web, mobile, and IoT applications. See https://ibm.com/cloud/cloudant for more details.

In this commissioning task we will establish a cloudant database on IBM Cloud Watson IoT Platform service.

The aim of the task is to load IoT sensor data in Cloudant DB in JSON format.

2. Cloudant in IBM Cloud

For the Watson IoT NodeRED commissioning of Cloudant you need to create a new Cloudant database ONLY IF you have not earlier worked with the NodeRED editor in your Watson IoT platform! The NodeRED flows are actually stored in a Cloudant database! The Cloudant NoSQLDB is already up and running!

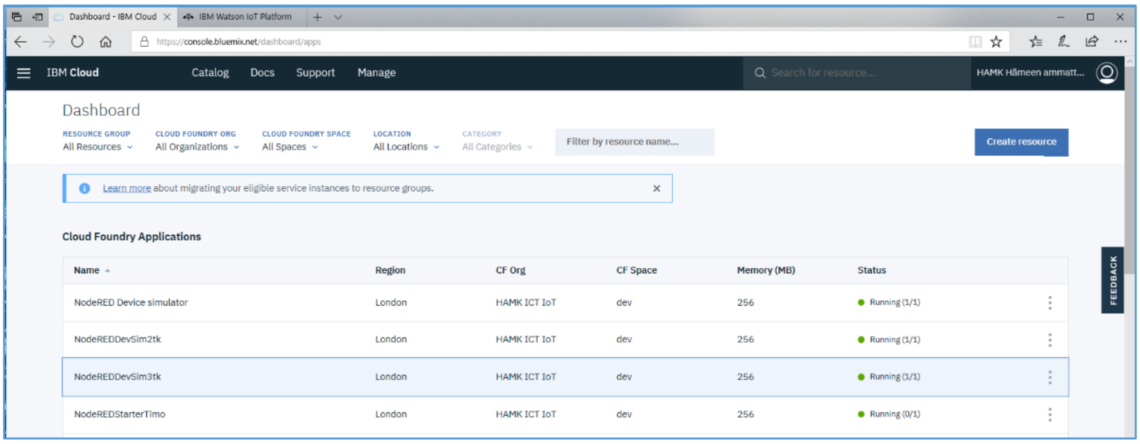
For other experiments with the Cloudant NoSQLDB you might want to create a new Cloudant.

The IBM recipe instructing this action:

https://console.bluemix.net/docs/services/IoT/cloudant_connector.html#cloudant_main

The version “Last Updated: 2018-09-17” was used for this document.

Please follow the original recipe instructions. It is devided in numbered tasks. Some screenshots are provided here showing the starting point and the most important actions.



Learn more about migrating your eligible service instances to resource groups.

Cloud Foundry Applications

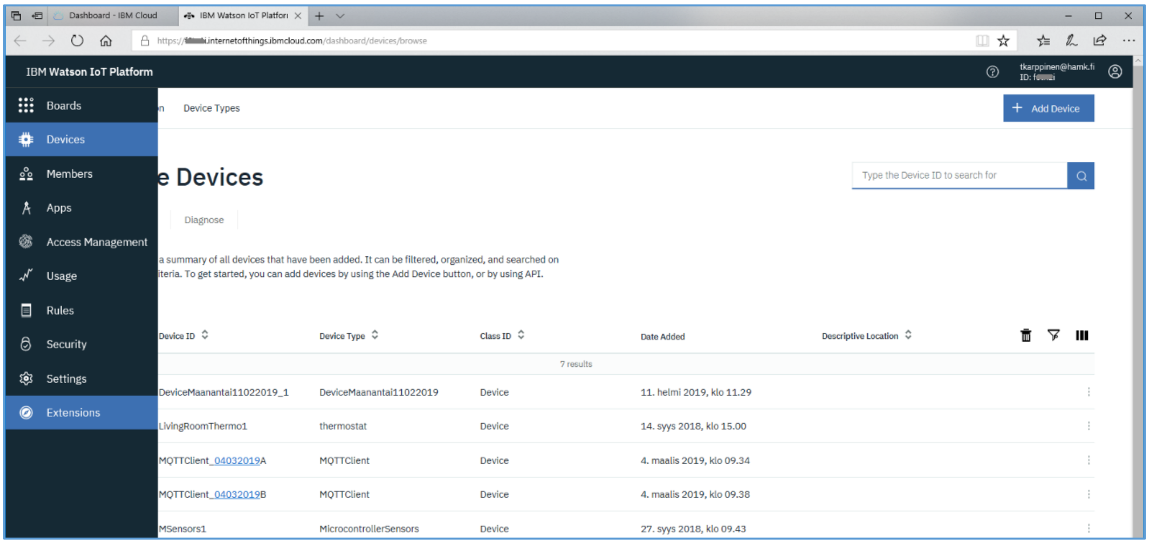
Name	Region	CF Org	CF Space	Memory (MB)	Status
NodeRED Device simulator	London	HAMK ICT IoT	dev	256	● Running (1/1)
NodeREDDevSim2tk	London	HAMK ICT IoT	dev	256	● Running (1/1)
NodeREDDevSim3tk	London	HAMK ICT IoT	dev	256	● Running (1/1)
NodeREDStarterTimo	London	HAMK ICT IoT	dev	256	● Running (0/1)

Fig 2.1 IBM Cloud Dashboard

If you already have an existing Cloud Foundry Application for your IoT Platform and NodeRED flow please open the corresponding Cloud Foundry Application in your Dashboard.

Open and launch the IoT Platform.

Select the Extensions on the left menu

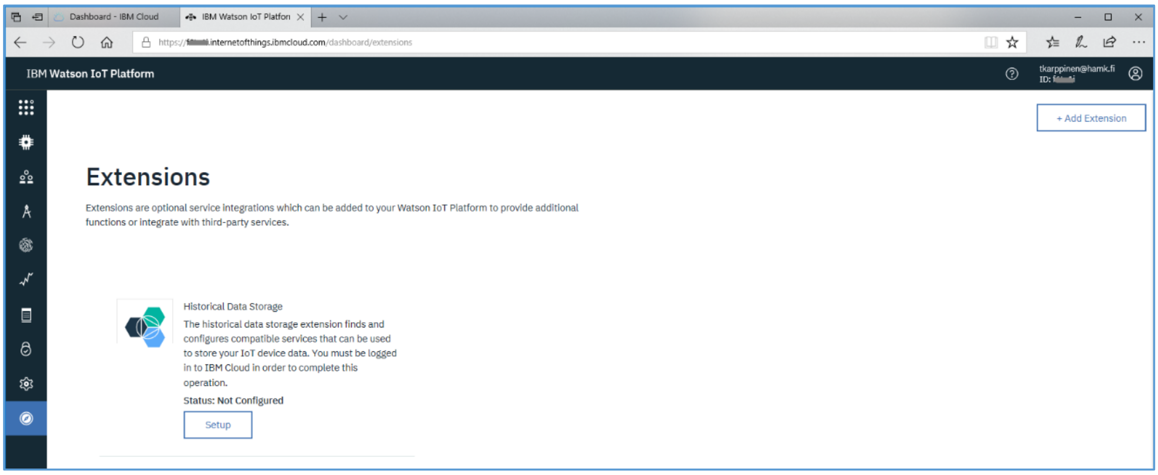


a summary of all devices that have been added. It can be filtered, organized, and searched on iteria. To get started, you can add devices by using the Add Device button, or by using API.

Device ID	Device Type	Class ID	Date Added	Descriptive Location
7 results				
DeviceMaanantai11022019_1	DeviceMaanantai11022019	Device	11. helmi 2019, klo 11.29	
LivingRoomThermo1	thermostat	Device	14. syys 2018, klo 15.00	
MQTTClient_04032019A	MQTTClient	Device	4. maalis 2019, klo 09.34	
MQTTClient_04032019B	MQTTClient	Device	4. maalis 2019, klo 09.38	
MSensors1	MicrocontrollerSensors	Device	27. syys 2018, klo 09.43	

Fig. 2.2 IoT Platform Dashboard and Extensions

In the Extensions select the Historical Data Storage and Setup.



Extensions

Extensions are optional service integrations which can be added to your Watson IoT Platform to provide additional functions or integrate with third-party services.

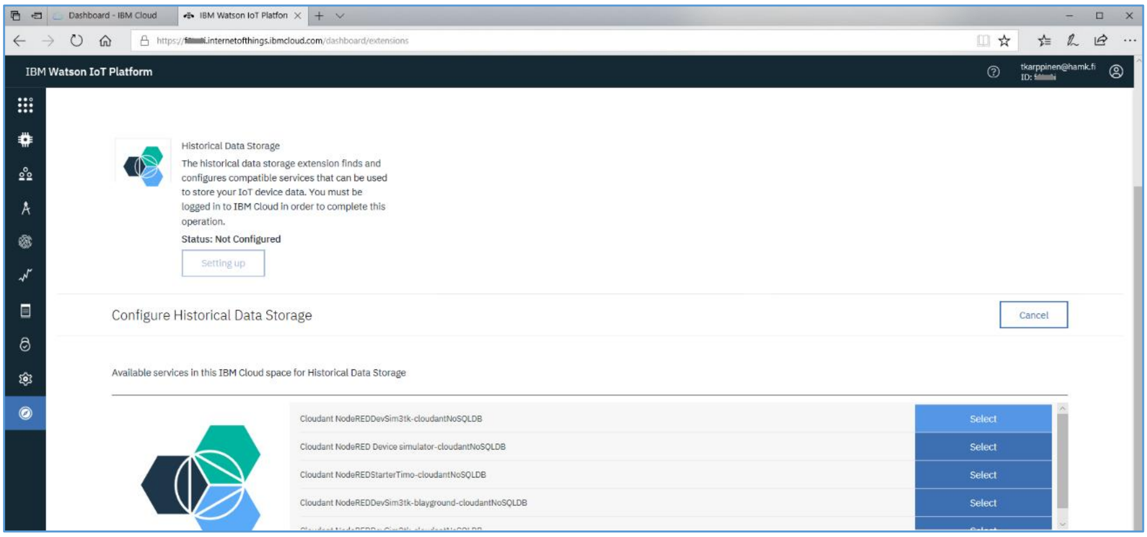
Historical Data Storage

The historical data storage extension finds and configures compatible services that can be used to store your IoT device data. You must be logged in to IBM Cloud in order to complete this operation.

Status: Not Configured

Fig. 2.3 Extensions, Historical Data Storage, Setup

Please select one of the services you have available in your IBM Cloud space. It is of course best to select the service where you can send sensor data from a sensor device, from mqtt client application or from simulated device.



Historical Data Storage

The historical data storage extension finds and configures compatible services that can be used to store your IoT device data. You must be logged in to IBM Cloud in order to complete this operation.

Status: Not Configured

Configure Historical Data Storage

Available services in this IBM Cloud space for Historical Data Storage

Cloudant NodeREDDevSim3tk-cloudantNoSQLDB	Select
Cloudant NodeRED Device simulator-cloudantNoSQLDB	Select
Cloudant NodeREDStarterTimo-cloudantNoSQLDB	Select
Cloudant NodeREDDevSim3tk-blayground-cloudantNoSQLDB	Select

Fig. 2.4 Selecting one of the services.

As instructed in the original IBM recipe:

Select your IBM Cloudant configuration options:

“”

a. Select a bucket interval. The bucket interval controls how frequently new databases are created to store device data. New buckets are created at midnight in the selected timezone using your selected bucket interval.

b. Select a time zone. The time in the selected timezone is used to determine which bucket device data is placed in. Timestamps on device data that is sent to IBM Cloudant are converted into the timezone that is associated with the database to which the data is sent.

c. The database name is created in the format iotp_<orgID>_<dbname>_<bucket_name>, where:

<orgID> is your organization ID.

<dbname> is the name of the database.

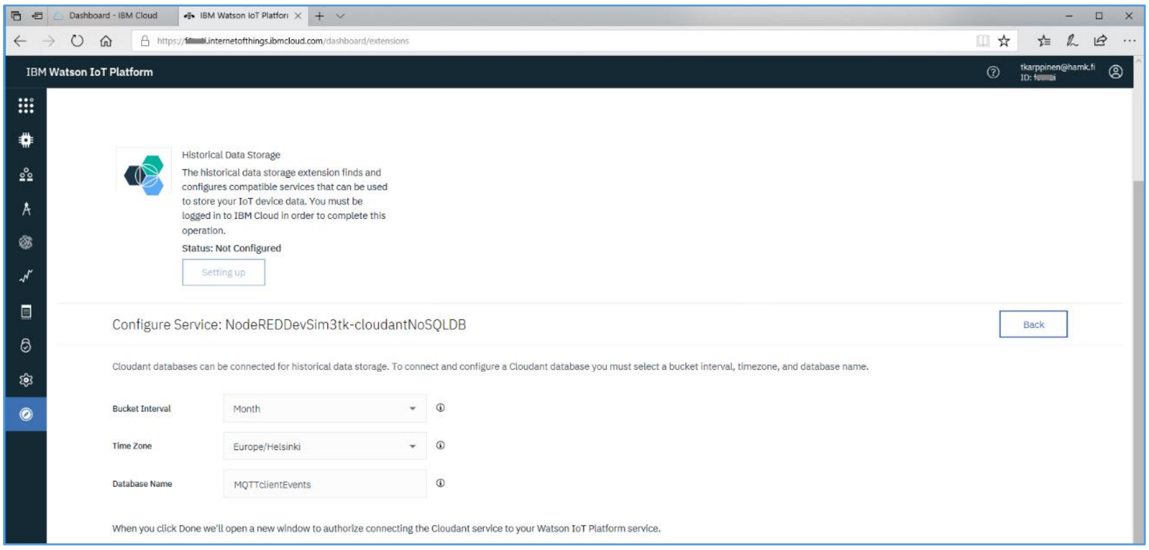
<bucket_name> is the name of the bucket.

For day bucket intervals, <bucket_name> is in the format yyyy-mm-dd. For example, 2016-07-06 for events on July 6th 2016.

For week bucket intervals <bucket_name> is in the format yyyy-'w'ww where 'w'ww indicates a week number. For example, 2016-w03 for events in the 3rd week of 2016.

For month bucket intervals <bucket_name> is in the format yyyy-mm. For example, 2016-07 for events in July 2016.

“”



Historical Data Storage

The historical data storage extension finds and configures compatible services that can be used to store your IoT device data. You must be logged in to IBM Cloud in order to complete this operation.

Status: Not Configured

Configure Service: NodeREDDevSim3tk-cloudantNoSQLDB

Cloudant databases can be connected for historical data storage. To connect and configure a Cloudant database you must select a bucket interval, timezone, and database name.

Bucket Interval: Month

Time Zone: Europe/Helsinki

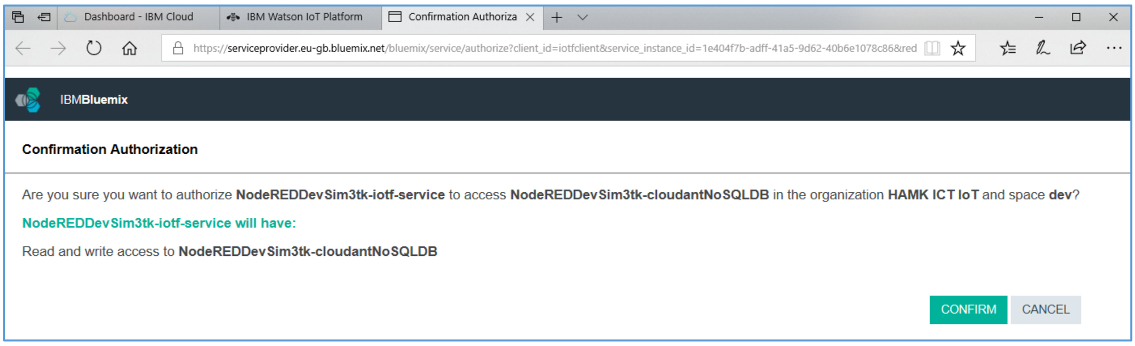
Database Name: MQTTclientEvents

When you click Done we'll open a new window to authorize connecting the Cloudant service to your Watson IoT Platform service.

Fig. 2.5 Cloudant DB configuration.

You can set configuration data as seen on picture but only with name ”default” the creation was successful !

The IBM Cloud tries to open a pop up window. Please allow it! You will see an authorization window. Confirm the authorization



Dashboard - IBM Cloud
IBM Watson IoT Platform
Confirmation Authoriza
https://serviceprovider.eu-gb.bluemix.net/bluemix/service/authorize?client_id=iotfclient&service_instance_id=1e404f7b-adff-41a5-9d62-40b6e1078c86&red
IBMBluemix

Confirmation Authorization

Are you sure you want to authorize **NodeREDDevSim3tk-iotf-service** to access **NodeREDDevSim3tk-cloudantNoSQLDB** in the organization **HAMK ICT IoT** and space **dev**?

NodeREDDevSim3tk-iotf-service will have:

Read and write access to **NodeREDDevSim3tk-cloudantNoSQLDB**

Fig. 2.6 Confirm the authorization

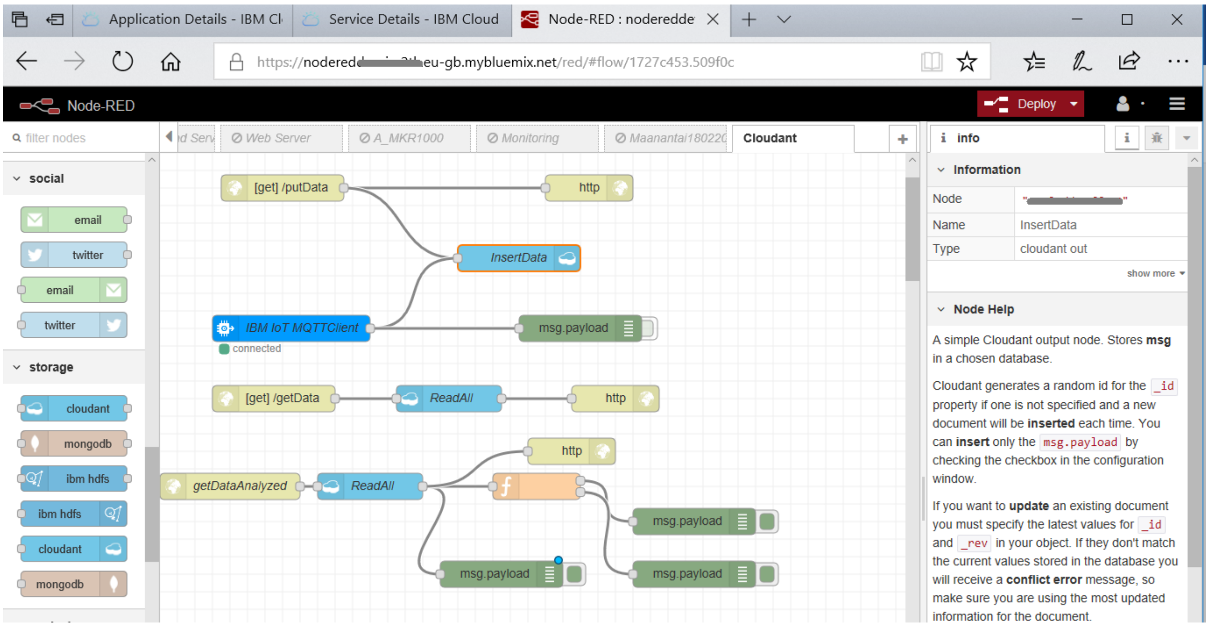
You will get a message “Authorization successful”.

Now your Cloudant DB is ready for IoT data.

3. How to upload data to and query data from Cloudant NoSQLDB using the Watson NodeRED

Open the IBM Cloud Dashboard. Select the Cloud Foundry Application where you have your NodeRED flows from earlier configuration tasks.

Create the following flow.



Information

Node	"[redacted]"
Name	InsertData
Type	cloudant out

show more

Node Help

A simple Cloudant output node. Stores **msg** in a chosen database.

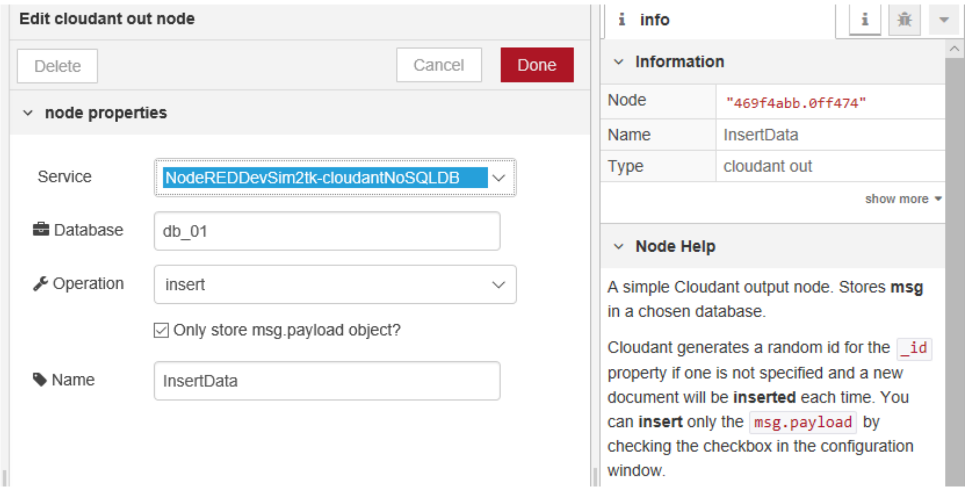
Cloudant generates a random id for the `_id` property if one is not specified and a new document will be **inserted** each time. You can **insert** only the `msg.payload` by checking the checkbox in the configuration window.

If you want to **update** an existing document you must specify the latest values for `_id` and `_rev` in your object. If they don't match the current values stored in the database you will receive a **conflict error** message, so make sure you are using the most updated information for the document.

Fig. 3.1 NodeRED flow including connections to a Cloudant NoSQLDB

The Cloudant nodes can be found in the storage section of the available nodes. All the other nodes are those common http, function and debug nodes.

The output node for Cloudant in the top center of the flow should be configured with following parameters.



Edit cloudant out node

Delete | Cancel | Done

node properties

Service: NodeREDDevSim2tk-cloudantNoSQLDB

Database: db_01

Operation: insert

☑ Only store msg.payload object?

Name: InsertData

info

Information

Node	"469f4abb.0ff474"
Name	InsertData
Type	cloudant out

show more

Node Help

A simple Cloudant output node. Stores **msg** in a chosen database.

Cloudant generates a random id for the `_id` property if one is not specified and a new document will be **inserted** each time. You can **insert** only the `msg.payload` by checking the checkbox in the configuration window.

Fig 3.2 output node Cloudant

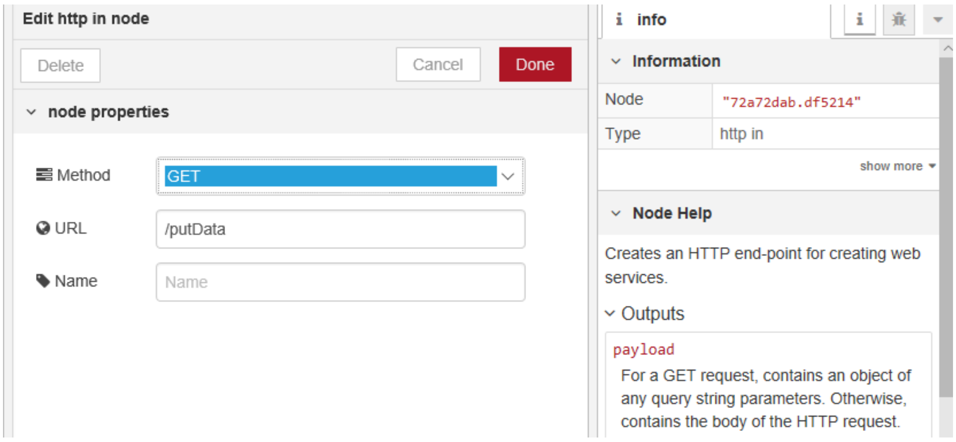
The Service is the name of the cloudant instance. Your instance was created when you created the NodeRED environment. All node red flows are actually stored in JSON format in the Cloudant NoSQLDB. The the name from the list.

Database is the name for the new database. The database name must begin with a letter and can include only lowercase characters (a-z), numerals (0-9), and any of the following characters _, $, (,), +, -, and /. More information can be found at https://console.bluemix.net/docs/services/Cloudant/getting-started.html#getting-started-with-cloudant .

Operation – select the insert from the list.

Tick the Only store msg payload object. You can experiment without this as well. The difference is in the amount data it stores from your mqtt publishing message. The simple http put method work only with this ticked.

Name is just a name for the node for the flow view.



Edit http in node

Delete | Cancel | Done

node properties

Method: GET

URL: /putData

Name: Name

info

Information

Node	"72a72dab.df5214"
Type	http in

show more

Node Help

Creates an HTTP end-point for creating web services.

Outputs

`payload`
For a GET request, contains an object of any query string parameters. Otherwise, contains the body of the HTTP request.

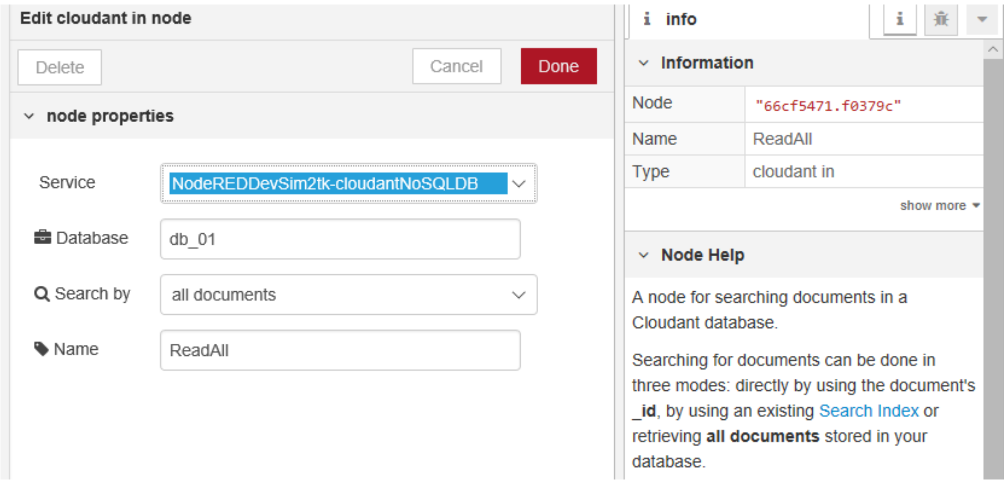
Fig 3.3 input node http

The http input node in top left is configured with parameters:

Method – GET

URL - /putData

The http request output node on top right of the flow is inserted in flow with default setting.



Service: NodeREDDevSim2tk-cloudantNoSQLDB

Database: db_01

Search by: all documents

Name: ReadAll

Node	"66cf5471.f0379c"
Name	ReadAll
Type	cloudant in

A node for searching documents in a Cloudant database.

Searching for documents can be done in three modes: directly by using the document's **_id**, by using an existing Search Index or retrieving **all documents** stored in your database.

Fig 3.4 Storage node Cloudant with an input and an output

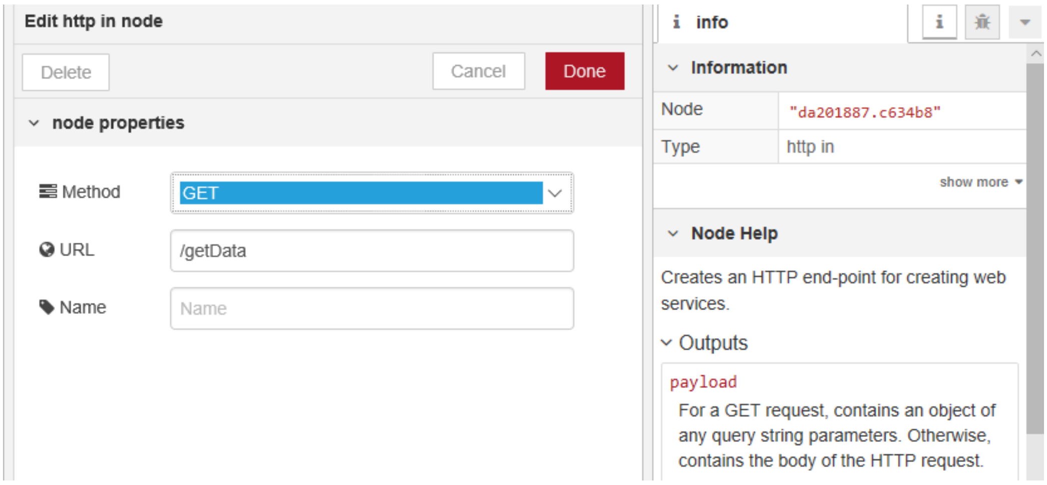
The Cloudant node in the middle of the flow is configured with parameters:

Service – the same as earlier

Database – the same as earlier

Search by – all document

Name – is just a name for the node for the flow view



Method: GET

URL: /getData

Name: Name

Node	"da201887.c634b8"
Type	http in

Creates an HTTP end-point for creating web services.

payload

For a GET request, contains an object of any query string parameters. Otherwise, contains the body of the HTTP request.

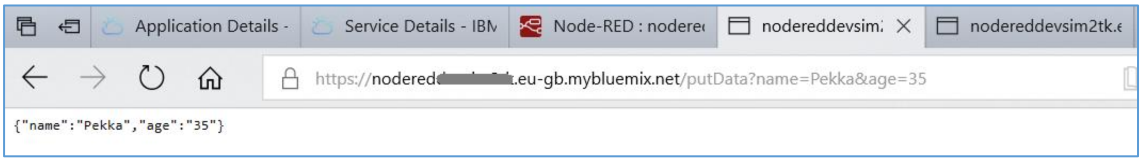
Fig 3.5 Input node http for reading data

The http input node for reading data out should be configured:

Method – GET

URL - /getData

Now we should be ready for the first test.



https://nodered[redacted].eu-gb.mybluemix.net/putData?name=Pekka&age=35

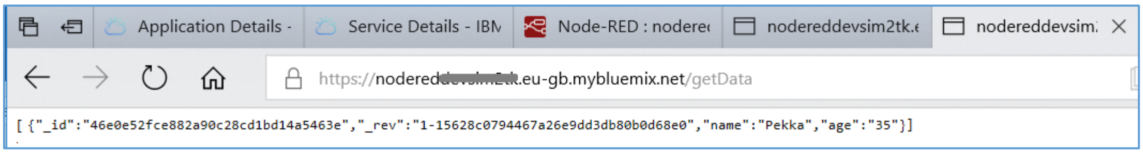
```
{"name":"Pekka","age":"35"}
```

Fig 3.6 Testing with http put

Please note the syntax of the put method:

/putData?name=Pekka&age=35

The host address is copied from your NodeRED address.

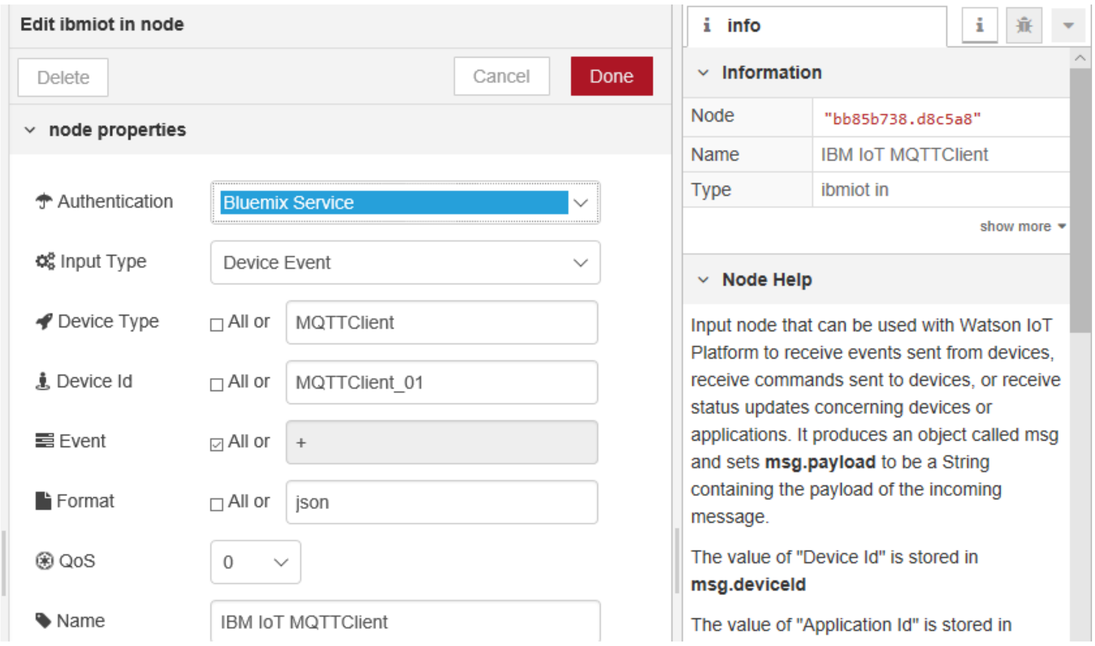


https://nodered[redacted].eu-gb.mybluemix.net/getData

```
[ {"_id":"46e0e52fce882a90c28cd1bd14a5463e","_rev":"1-15628c0794467a26e9dd3db80b0d68e0","name":"Pekka","age":"35"}]
```

Fig 3.7 Testing with Cloudant query

In the Watson IoT node in the left you can specify what data you would upload to the Cloudant NoSQLDB from your Watson IoT Devices.



Authentication		Bluemix Service
Input Type		Device Event
Device Type	□ All or	MQTTClient
Device Id	□ All or	MQTTClient_01
Event	☑ All or	+
Format	□ All or	json
QoS		0
Name		IBM IoT MQTTClient

Node	"bb85b738.d8c5a8"
Name	IBM IoT MQTTClient
Type	ibmiot in

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

The value of "Device Id" is stored in **msg.deviceId**

The value of "Application Id" is stored in

Fig 3.8 The Watson IoT mqtt node is set for reading dat from one device only.

Please test with a device sending mqtt events.

4. Analyzing data retrieved from Cloudant NoSQLDB

In the NodeRED flow seen already in the previous chapter there is a function node.

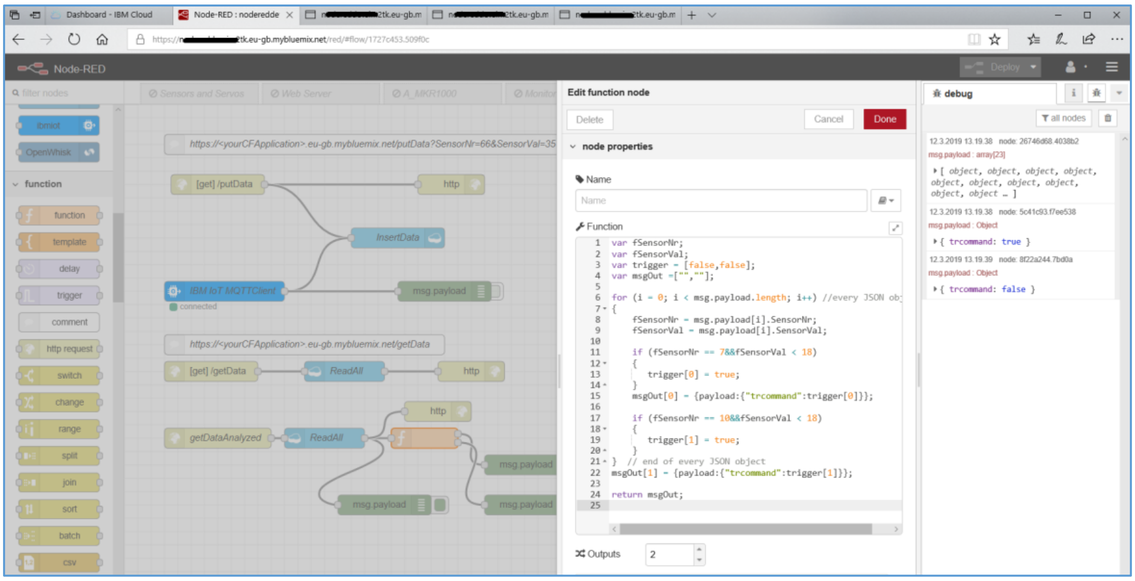
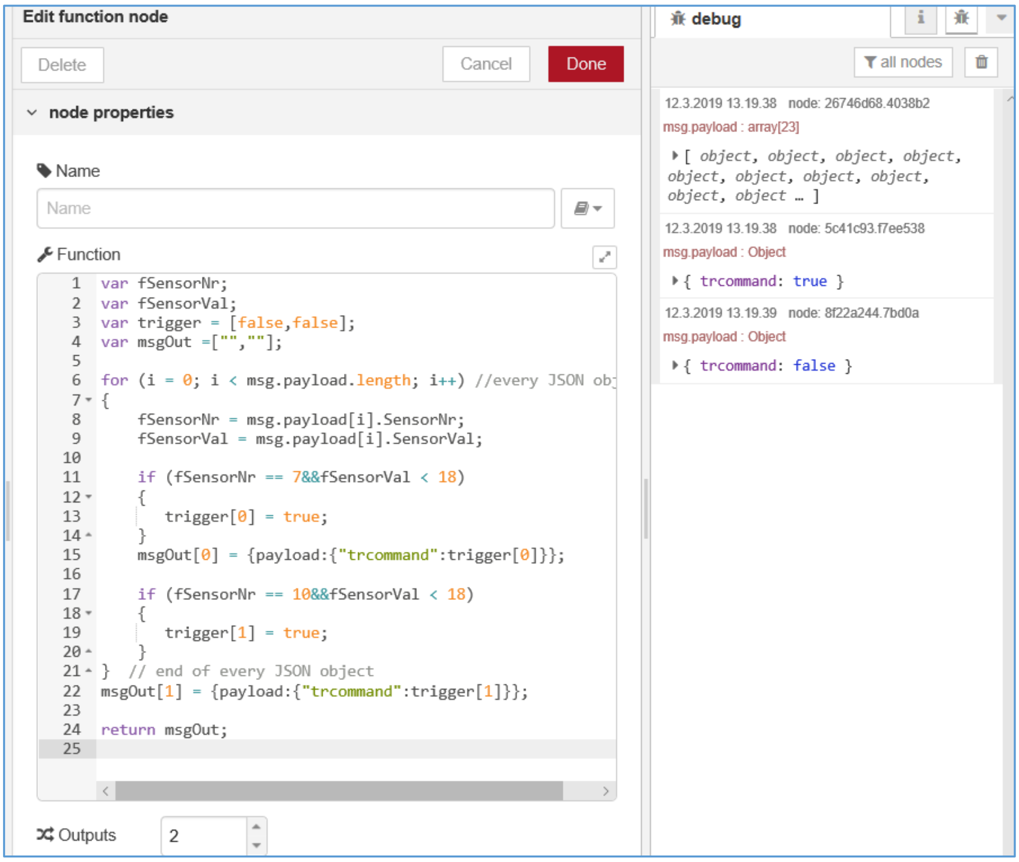


Fig. 4.1 NodeRED flow with Cloudant NoSQLDB

The function is programmed with javascript.



```
var fSensorNr;
var fSensorVal;
var trigger = [false,false];
var msgOut =["",""];

for (i = 0; i < msg.payload.length; i++) //every JSON ob
{
    fSensorNr = msg.payload[i].SensorNr;
    fSensorVal = msg.payload[i].SensorVal;

    if (fSensorNr == 7&&fSensorVal < 18)
    {
        trigger[0] = true;
    }
    msgOut[0] = {payload:{"trcommand":trigger[0]}};

    if (fSensorNr == 10&&fSensorVal < 18)
    {
        trigger[1] = true;
    }
} // end of every JSON object
msgOut[1] = {payload:{"trcommand":trigger[1]}};

return msgOut;

```

```
12.3.2019 13.19.38  node: 26746d68.4038b2
msg.payload : array[23]
▸[ object, object, object, object,
object, object, object, object,
object, object … ]
12.3.2019 13.19.38  node: 5c41c93.f7ee538
msg.payload : Object
▸{ trcommand: true }
12.3.2019 13.19.39  node: 8f22a244.7bd0a
msg.payload : Object
▸{ trcommand: false }
```

Fig. 4.2 The java script function

Please write a similar function. Test how it works.

Otherwise the nodes in the flow are with the same parameters as in the earlier chapter.

Your task is now to develop there some kind of function for analyzing the data in the Cloudant NoSQLDB.

Commissioning Task A.

Read all the JSON objects in the database. You probably have same kinds of object properties as in the example; there is some sensor name and a value associated to it.

Find out the lowest sensor value in the database for each sensor name.

Find out the highest sensor value in the database for each sensor name.

Trigger the output one if the highest value is outside set limit value.

Trigger the output two if the lowest value is outside set limit value.

Commissioning Task B.

Make a quick review on earlier commissioning task documents. Publish the trigger message as a command type of event back to IBM IoT platform and back to the MQTT client.