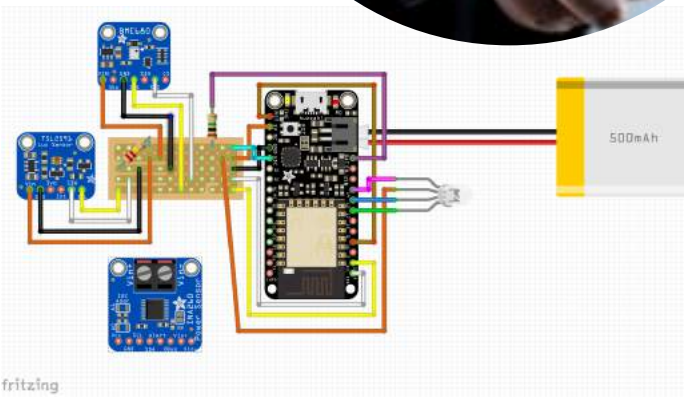
{ Lab 3 }

IoT application in IBM Cloud with Node-RED and IBM Watson



April 2020 – Version 2.4

Authors: B. Marolleau, C. Bacle, C. Lalevée
March 2020 - v2.4

Agenda

Before Starting 4

1. Hands-on presentation 5
Section 1. Overview 5
Section 2. Prerequisites 7

2. Create your Node-RED application 8

3. Create sensor and a new flow 15
Section 1. Sensors & IoT 15
Section 2. Node-RED flow: creation & importation 16
Section 3. Insert IoT Data in Cloudant DB 19
Section 4. Process IoT Data with Watson 21

4. Create a dashboard application in Node-RED 23
Section 1. Import Node-RED Dashboarding capability 23
Add extra nodes to your Node-RED palette 25
Section 2. Create a simple Node-RED Dashboard 31
Section 3. Add voice alert on dashboard (Optional) 35

5. Create a dashboard in Watson IoT Platform (optional) 38
Section 1. Create new device in Watson IoT Platform 38
Section 2. Node-RED: redirect sensor data to IBM IoT Platform 42
Section 3. IBM Watson IoT Platform: create dashboard 45

6. Physical Device – MQTT Connection 52
Section 1. Simulate your device 57
Section 2. Create a MQTT Device on a micro-controller 61
Section 3. Data Visualization from MQTT Subscribers Apps 64

Before Starting

Information

This hands-on required to have an IBM Cloud account. If you don't, you can create one here: http://bluemix.net/.

- Open a browser and access to IBM Cloud: https://console.bluemix.net.
- If you have an IBM Cloud account, click **Log in**, and enter your IBM ID credentials.
  If you don't have an IBM Cloud account, click **Create a free account**. Enter your email address, and additional information required. You will receive an email with activation link. Once activated, you could use your new free IBM cloud account: log in.

- Select organization, location and space to use during this lab.

- If needed, free resources (GB / #Services) in your IBM Cloud Organization & Spaces to run the lab exercises.
  If you encounter a resource contention (error message saying you are out of resources), clean up your spaces by deleting existing Apps or Services.

1. Hands-on presentation

Section 1. Overview

In this hands-on session, you will create a Node-RED application in IBM Cloud to collect, store and display virtual sensor data.

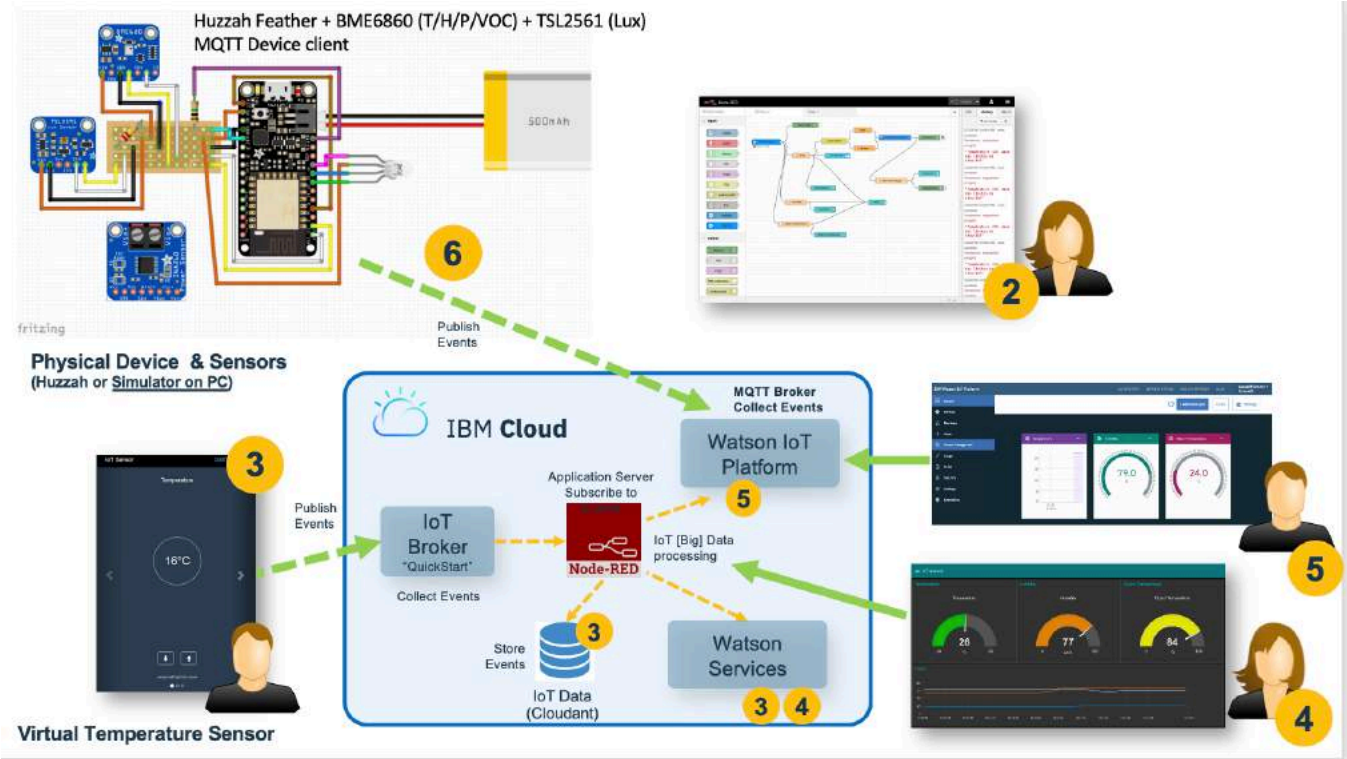
Node-RED (https://nodered.org/) is a flow-based programming tool, originally developed by the IBM Emerging Technology Services team (in early 2013) and now a part of JS Foundation. Traditional development can be very technical, but Node-RED enables you to concentrate on the logic of your workflow and allows fast prototyping.
Node-RED consists of a Node.js-based runtime with a flow editor accessed through a web browser. Within the browser, you create your application by dragging nodes from a customizable palette into a workspace and start to wire them together. With a single click, the application is deployed back to its runtime.

Session objectives are:

- Create & modify an application using Node-RED in IBM Cloud
- Discover new services & Node-RED to consume or create services (IoT / database…)
- Discover Watson services
- Discover Watson IoT Platform

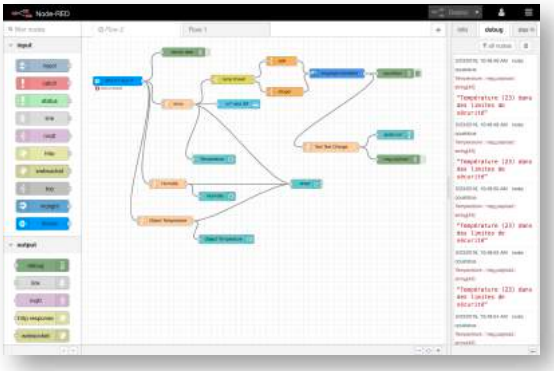
Find below lab overview (with exercise numbers).

Exercise 6 (optional) requires a micro-controller (Huzzah, Arduino, etc) or a Windows / Linux / Mac desktop with node.js installed.

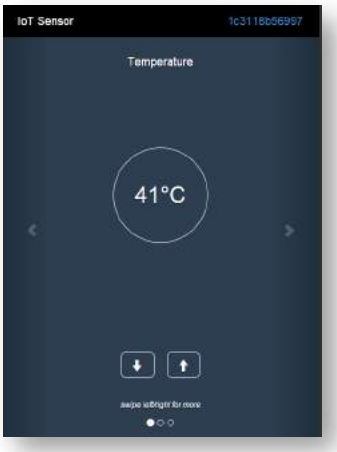


Expected results are :

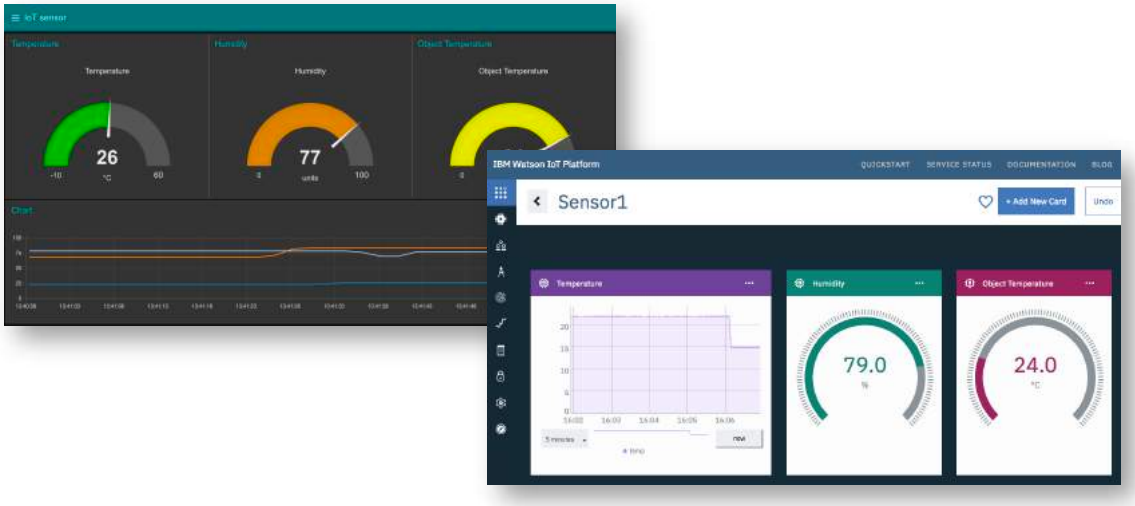
- Your Node-RED application is operational (using Node.js runtime), accessing Cloudant & IoT platform (QuickStart)



- Your Node-RED app is online (reachable from the Internet), & will be connected to a temperature simulator (sensor)



- Optionally, you are able to provide 2 dashboards: one with voice alert implemented in Node-RED, the second designed and hosted in Watson IoT Platform



Section 2. Prerequisites

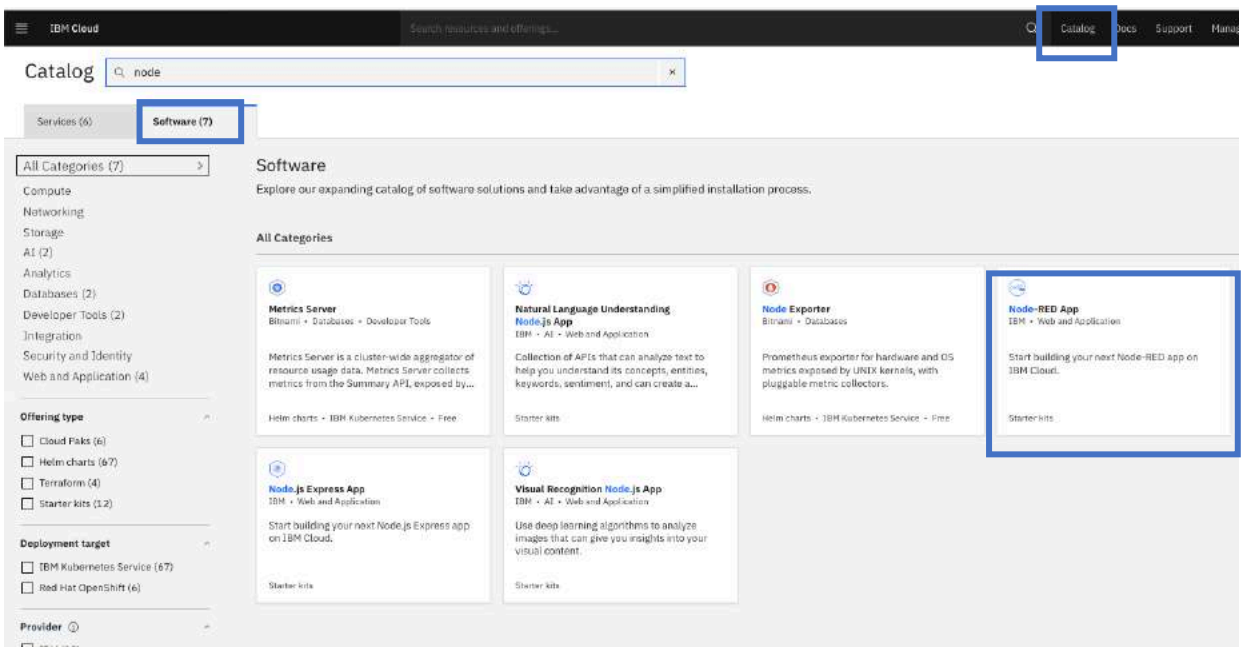
The required file used during this lab can be downloaded the JSON files from

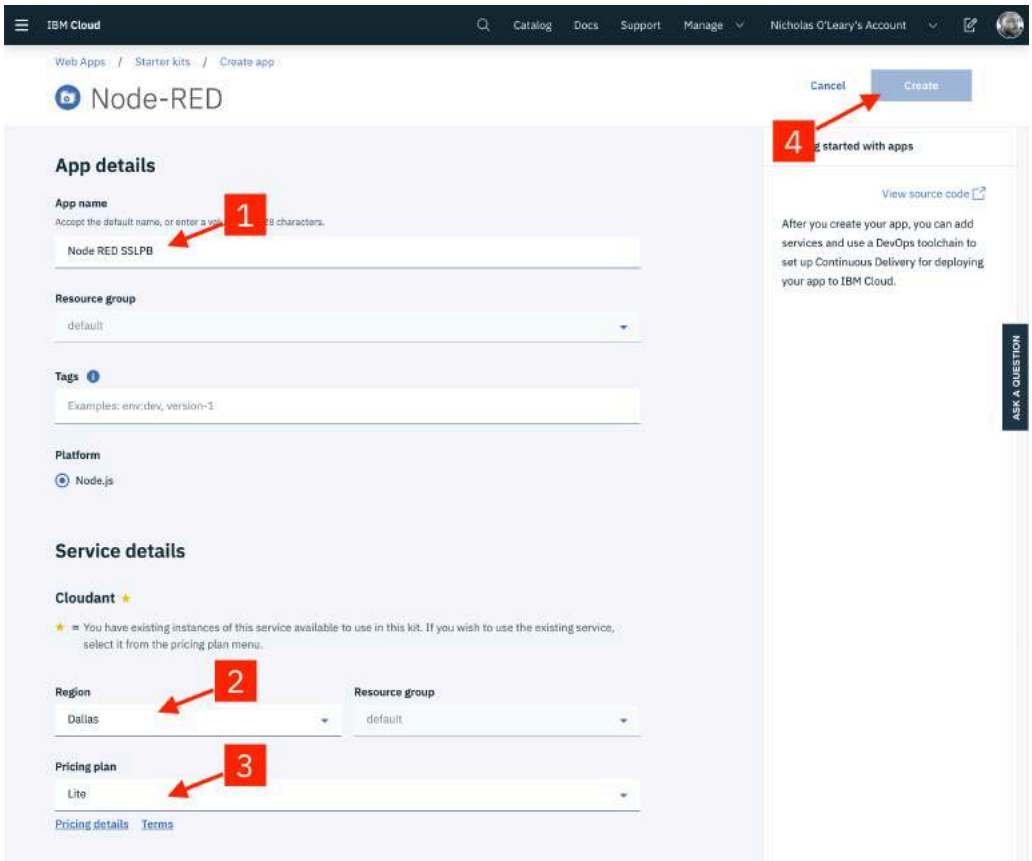
- http://ibmcloud-watson-day.mybluemix.net/files/Lab3_IoT_Flow.v2.1.json
- http://ibmcloud-watson-day.mybluemix.net/files/Lab3_IoT_Dashboard.v2.1.json

2. Create your Node-RED application

__ 1. In IBM Cloud **Catalog**, choose **Software** category

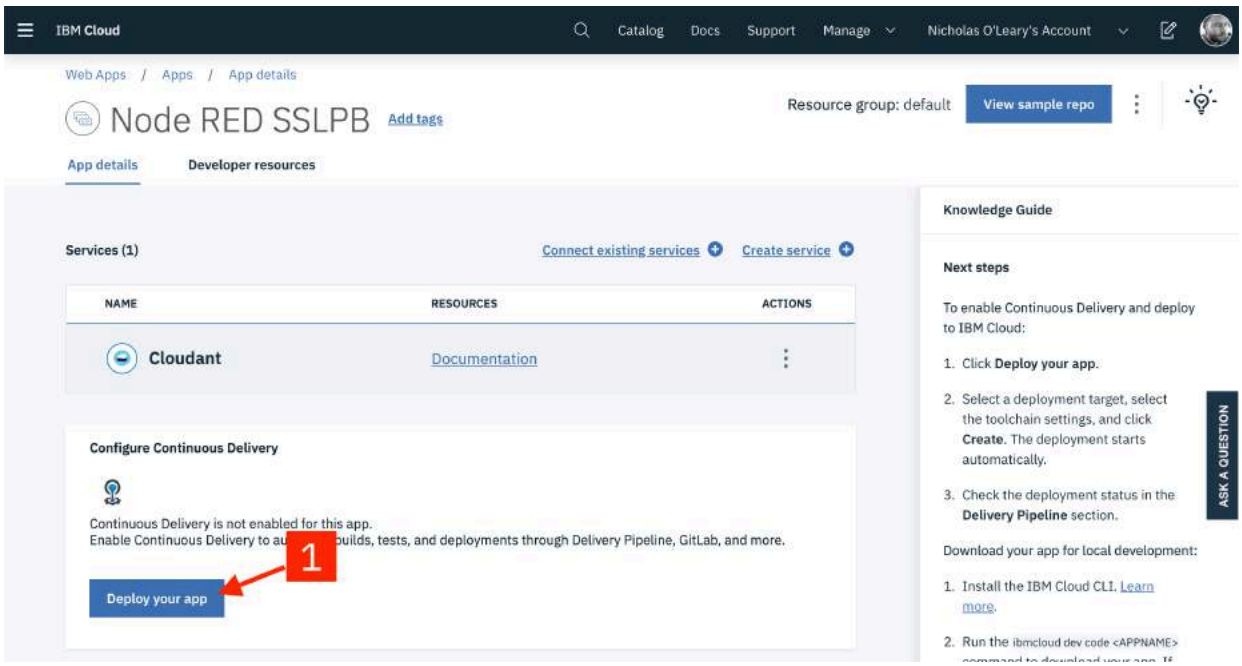
Search for “node” , select Node-Red App , fill in the form. Click on **Create**



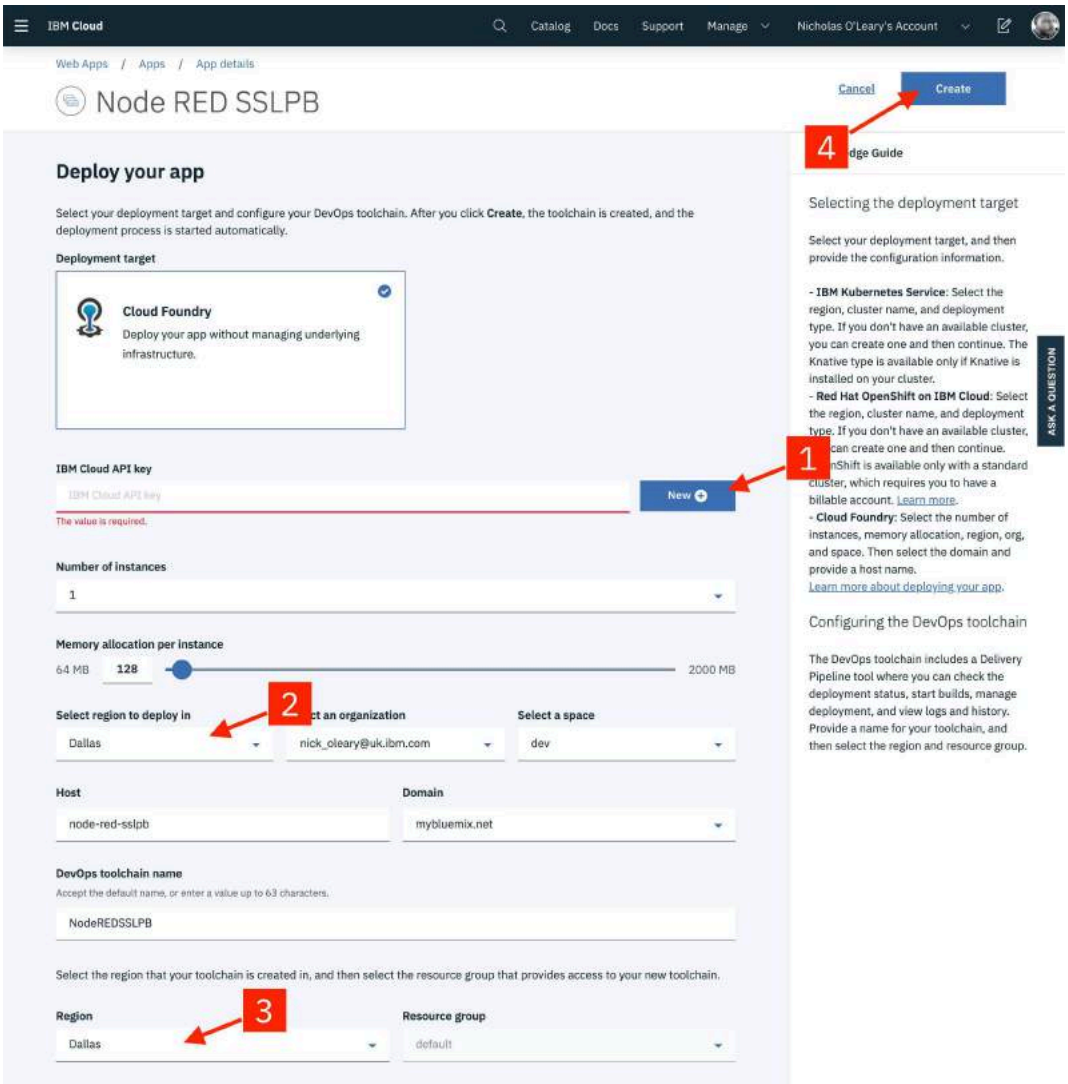


At this point, you have created the application and the resources it requires, but you have not deployed it anywhere to run. This step shows how to setup the Continuous Delivery feature that will deploy your application into the **Cloud Foundry** space of IBM Cloud.

__ 2. On the next screen, click the **Deploy your app** button (1) to enable the *Continuous Delivery* feature for your application.



__ 3. You will need to create an **IBM Cloud API** key to allow the deployment process to access your resources. Click the **New** button (1) to create the key. A message dialog will appear. Read what it says and then confirm and close the dialog.

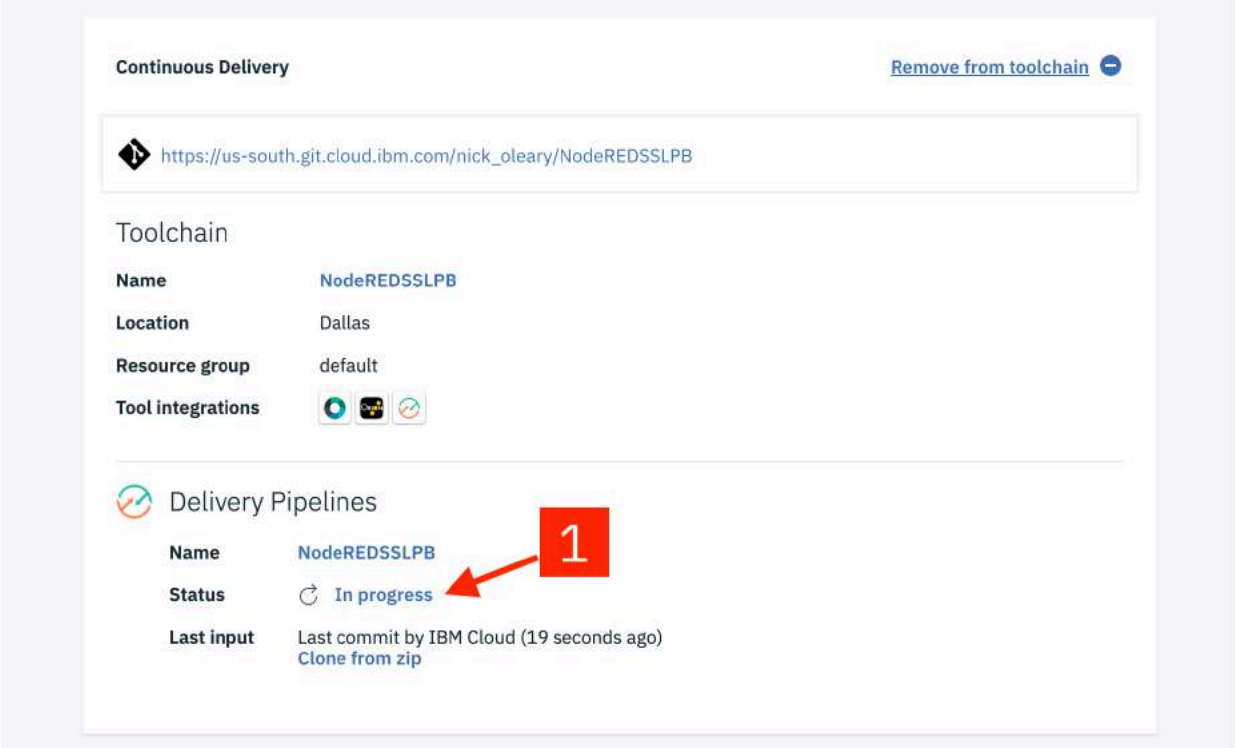


__ 4. The Node-RED Starter kit only supports deployment to the **Cloud Foundry** space of IBM Cloud. Select the **region** (2) to **deploy your application to**. This should match the region you created your Cloudant instance in.

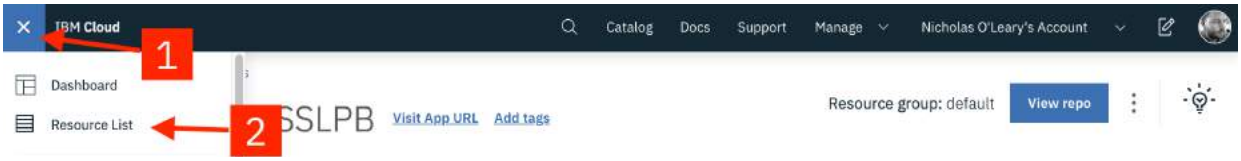
__ 5. Select the region (3) to create the DevOps toolchain.

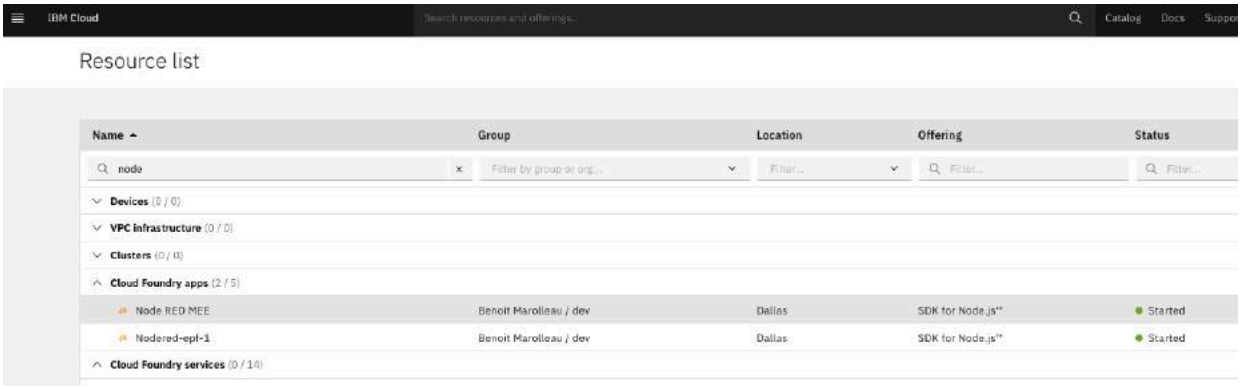
__ 6. Click **Create** (4). This will take you back to the application details page

__ 7. After a few moments, the Continuous Delivery section will refresh with the details of your newly created Toolchain. The Status field of the Delivery Pipeline will show **In progress**. That means your application is still being built and deployed.

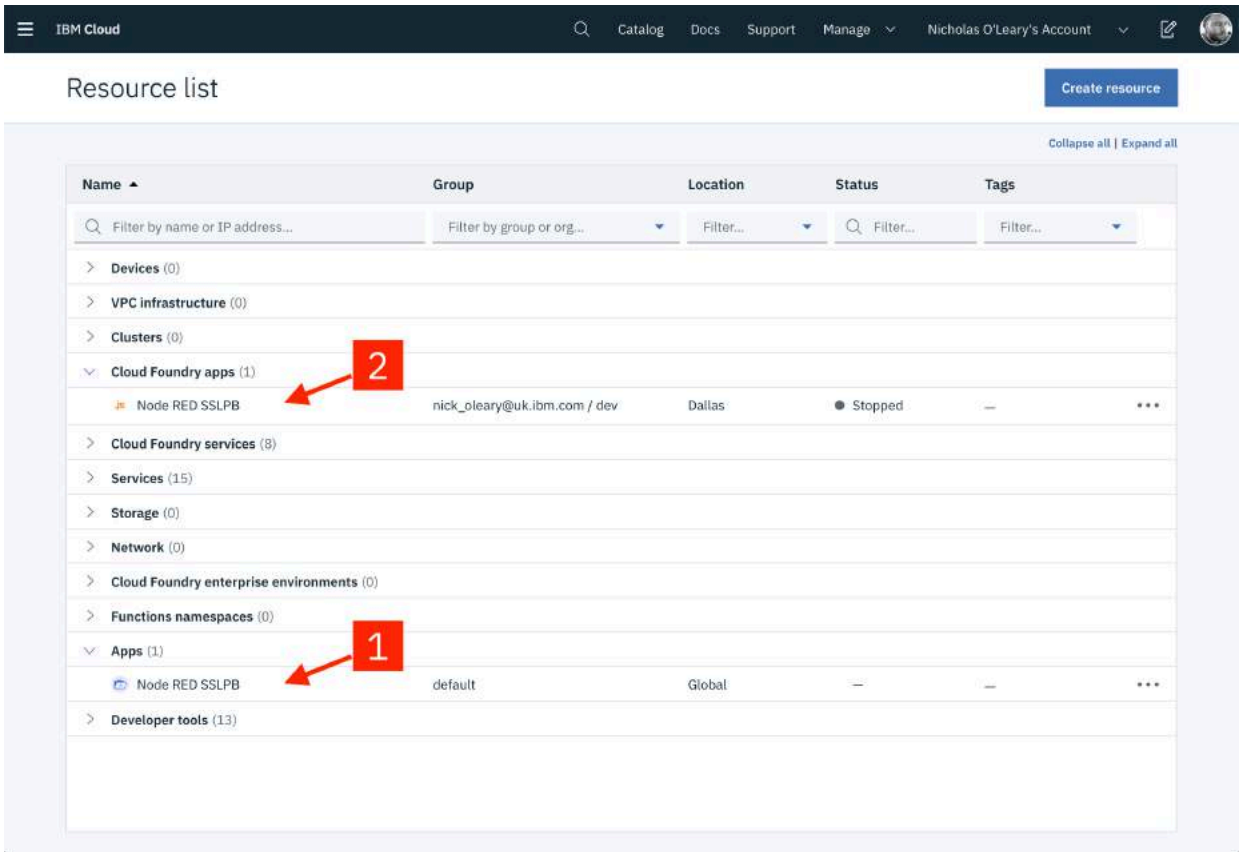


__ 8. Open your IBM Cloud Resource list by selecting the sidebar menu (1) and then selecting **Resource List** (2)





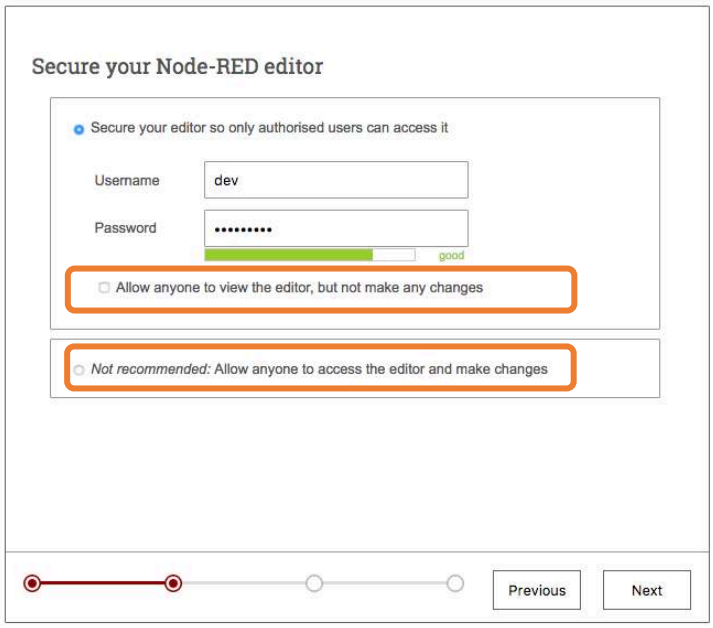
__ 9. You will see your newly created Node-RED Application listed under the **Apps** section (1). You will also see a corresponding entry under the **Cloud Foundry apps** section (2). Click on this Cloud Foundry app entry to go to your deployed application's details page.



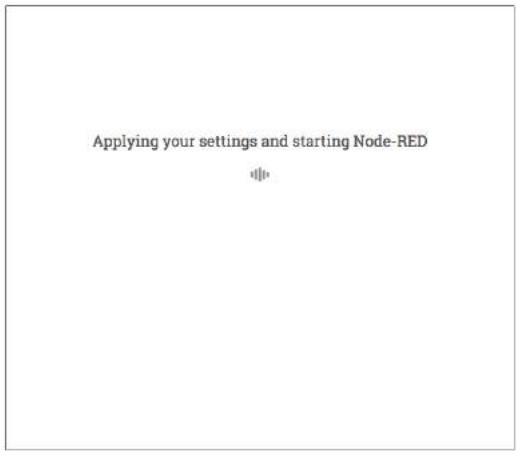
__ 10. From the details page, click the **Visit App URL** link to access your Node-RED Starter application.

__ 11. Run the wizard to configure authentication: secure your editor with your own credentials so only authorized users can access it (Node-RED has its own authentication system).

Don't check **Allow anyone to view the editor, but not make any changes** and **Allow anyone to view the editor**.

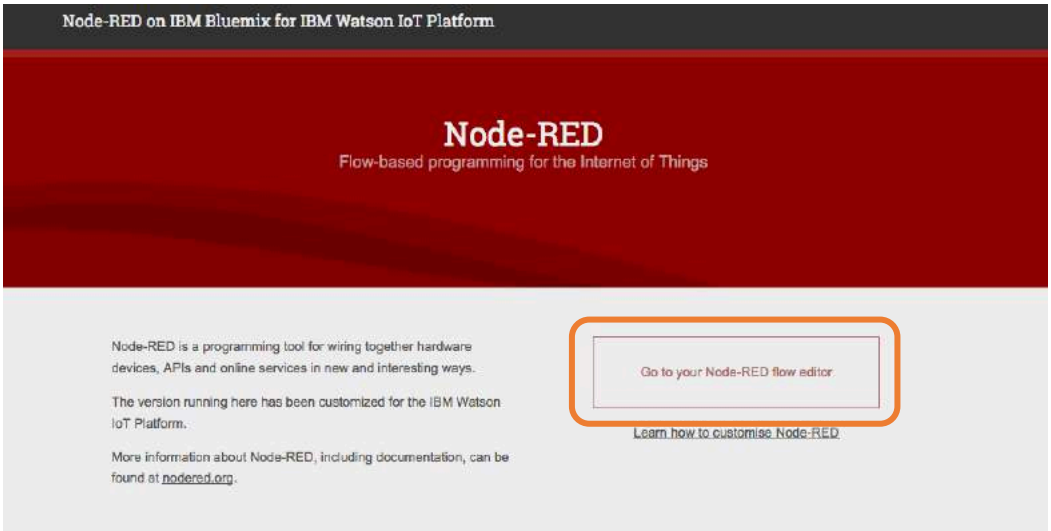


__ 12. On wizard last step screen, click on **Finish** to start Node-RED.

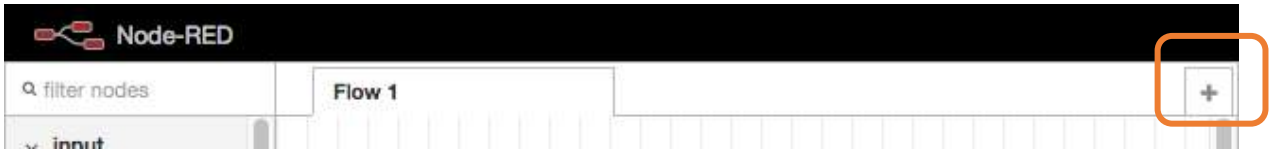


__ 13. Node-RED is a browser-based editor that makes it easy to wire together flows that can be deployed to the runtime. In the case of IoT, Node-RED is really powerful to quickly test all the possibilities that IBM Cloud offers with different kind of services.

Your Node-RED app has a public URL like any web app (you defined it in step 2).
Click on **Go to your Node-RED flow editor** and use the credentials provided before.



__ 14. You now have access to the Node-RED UI.
Keep the existing default flow and create a new flow: click **+**. You will use it in next exercise.



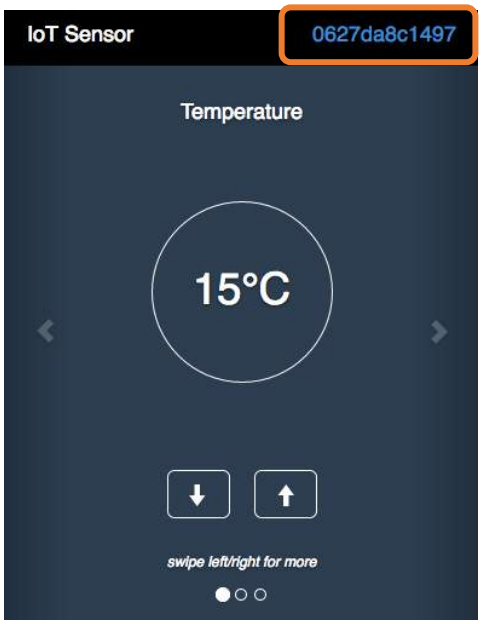
3. Create sensor and a new flow

Section 1. Sensors & IoT

__ 1. Open a new window or tab in your browser.

__ 2. To create a sensor simulator, connect to http://ibm.biz/iotsensor
There are 3 simulated sensors:
- Object temperature
- Temperature
- Humidity

The simulator (from IBM Cloud IoT Quickstart) connects automatically and starts publishing data.
It must remain connected to visualize the data.
Use the simulator buttons to change the simulated sensor readings. Data is published periodically.



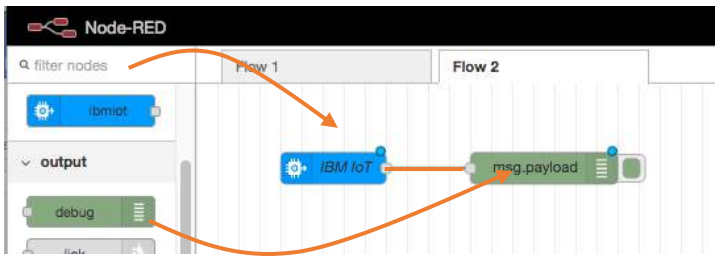
Note: Instead of using your desktop browser, you can use your smartphone.

__ 3. Identify your virtual device ID (top right corner) : copy it. You will use it in next section.
Warning: if you reload this page, the device ID changes.

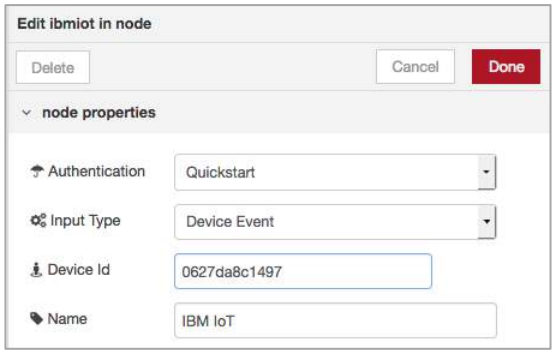
Section 2. Node-RED flow: creation & importation

__ 1. Go back to Node-RED window

__ 2. From left panel, drag and drop nodes to the workspace
- Chose the Input node **ibmiot**
- Add an output **debug** node
- Link them

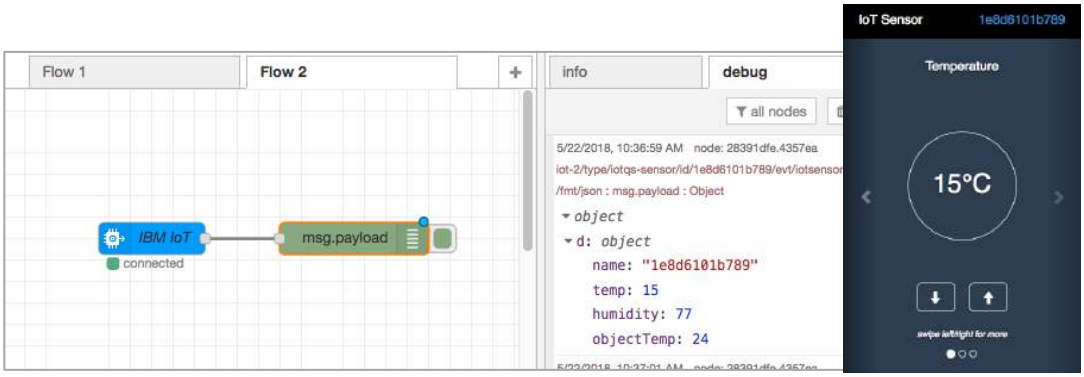


__ 3. Configure IBM IoT by double clicking on it :
- Authentication: Quickstart (means it is a simple authentication – for demo purposes)
- Device ID : <The value from Section 1, step 3 - Generated by the Simulator>

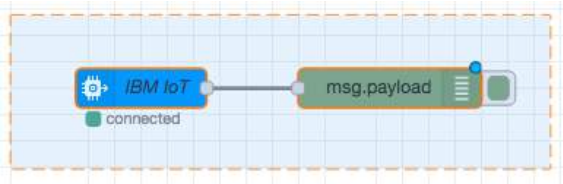


__ 4. Click **Done** & deploy your flow by clicking the **Deploy** button (top right).

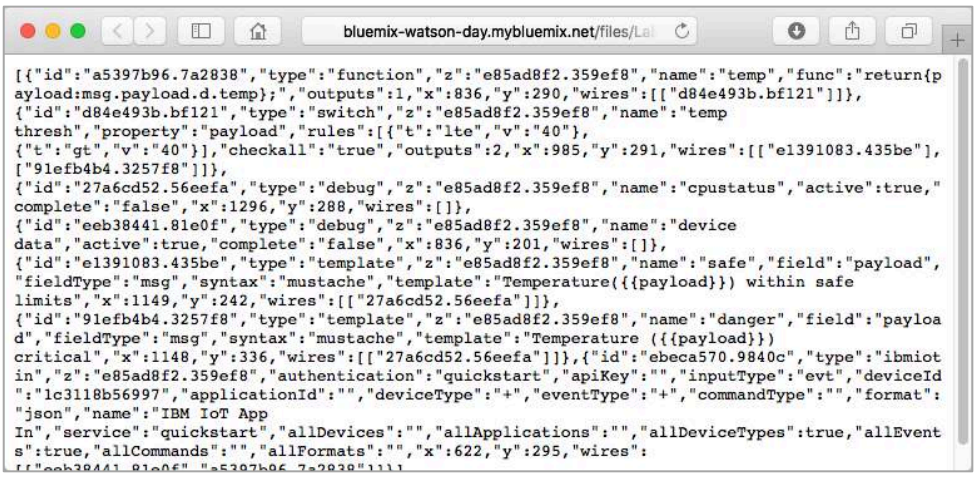
__ 5. Check the **Debug** Panel on the right side while you are playing with the sensor simulator. You should receive Device (sensor = web app. you opened in other window) data as the IBM IoT Node subscribed to this particular Device topic.



__ 6. Delete the whole flow by selecting all the nodes & pressing the 'Delete' key.

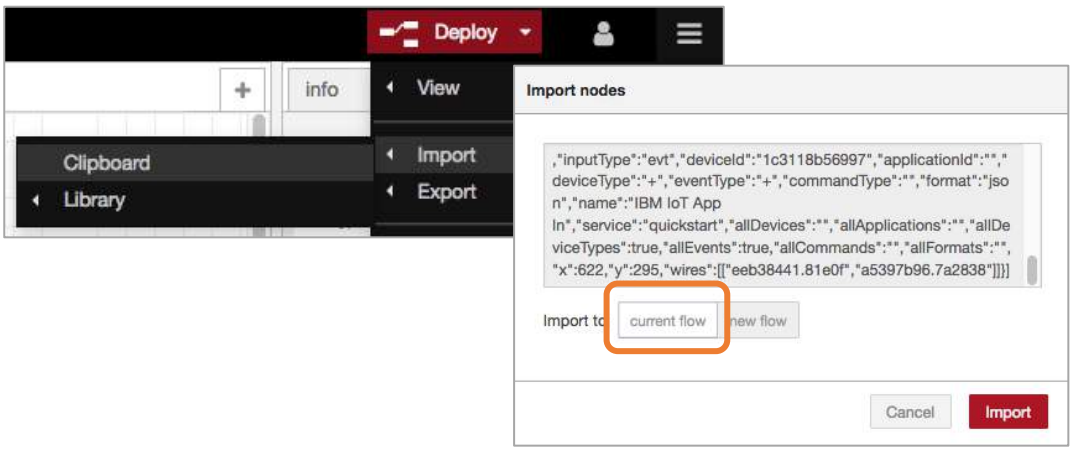


__ 7. Now import a new flow. A flow can be exported and imported using JSON file.
Open link http://ibmcloud-watson-day.mybluemix.net/files/Lab3_IoT_Flow.v2.1.json to display code in JSON format. You can also open the file you downloaded previously.



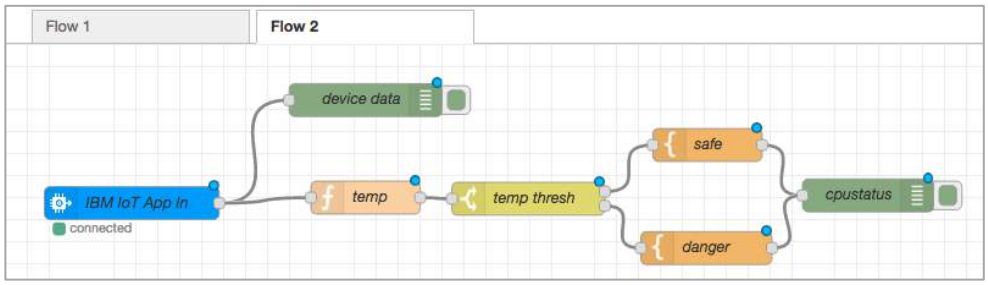
__ 8. Select all and copy it

__ 9. Click on the top right button near **Deploy**.
Select **Import**, **Clipboard** & copy/paste the content of the JSON file in **Current flow**.

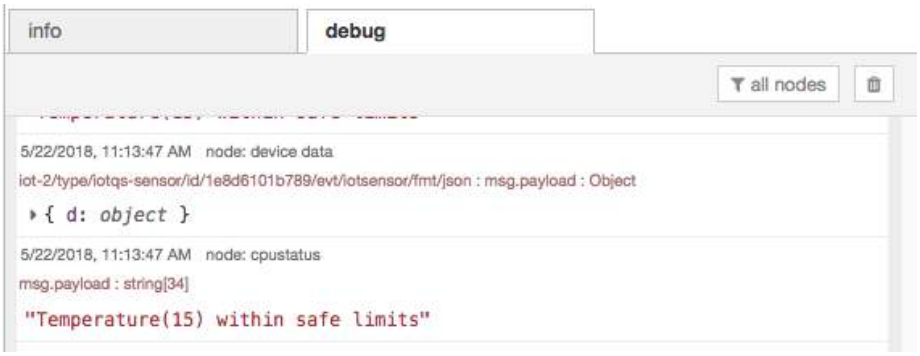


__ 10. Click on workspace to paste imported nodes

__ 11. Fill in the **Device ID** field in the **IBM IoT App In** node.



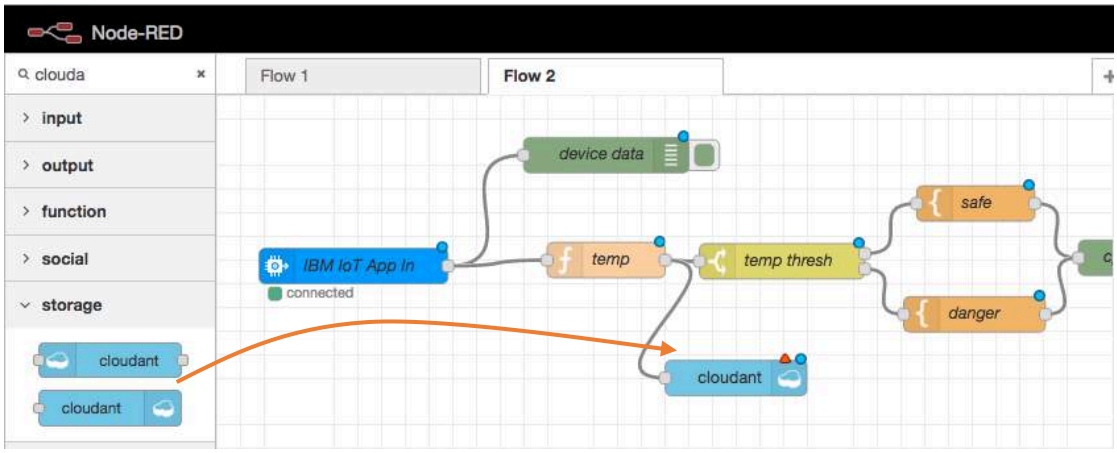
__ 12. Click **Deploy** to deploy the new Flow.
Modify the Device Temperature & check the **Debug** logs.



Section 3. Insert IoT Data in Cloudant DB

Let's insert the event data coming from the Device sensors in a Cloudant database! Remember that you already have a Cloudant service deployed for Node-RED. You will use it to store your data.

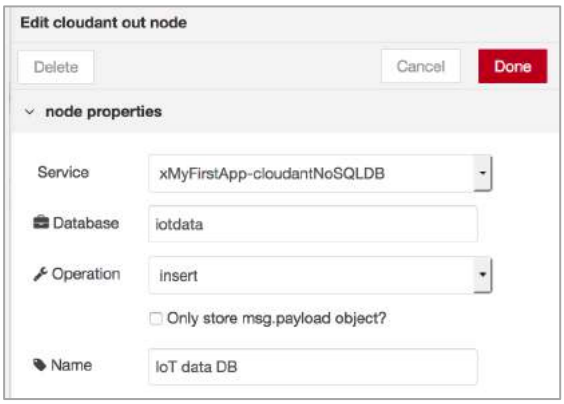
__ 1. Add a Cloudant Node (**Cloudant OUT** node in the **Storage** Category) & link it to the **temp** function node



__ 2. Configure it:

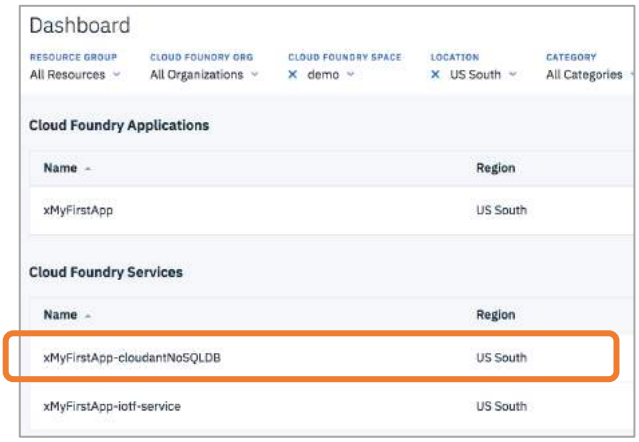
- Service : Cloudant service name bound to your Node.js runtime.
  As Node.js is already bound to a Cloudant Service, the service name should appear in the Drop-down list.
- Database: name of your choice (lower case)
- Name (node): name of your choice

Click **Done**.



__ 3. Deploy your new flow (**Deploy** button)

__ 4. From your IBM Cloud Dashboard (IBM Cloud window of your browser), start the Cloudant dashboard by clicking on the line of Cloudant service



__ 5. Click **Launch** button to start Cloudant console

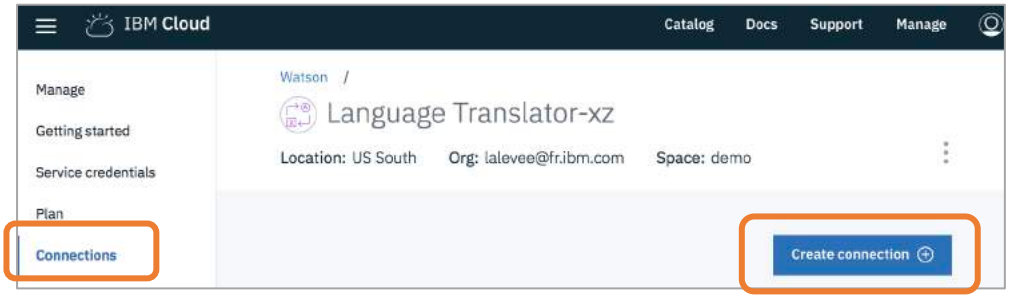
__ 6. Select **Database** icon in left panel, then your database name (defined in step 2).

__ 7. Have a look to the inserted data in the database. Click on record to see content.

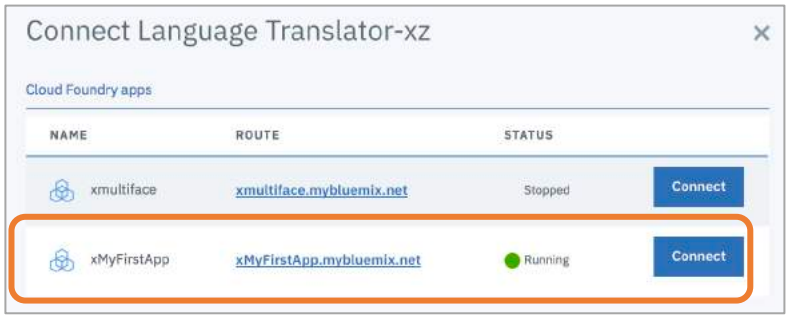
Section 4. Process IoT Data with Watson

__ 1. In IBM Cloud windows, add a « Watson Language Translator » service to your existing Node-RED application. From **Catalog**, click on **Language Translator** in Watson category.
Select **Lite** plan, then click **Create**. Service is deployed.

__ 2. From Language Translator dashboard, **Connections** menu, click on **Create connection**.



__ 3. Connect service to your Node-RED application

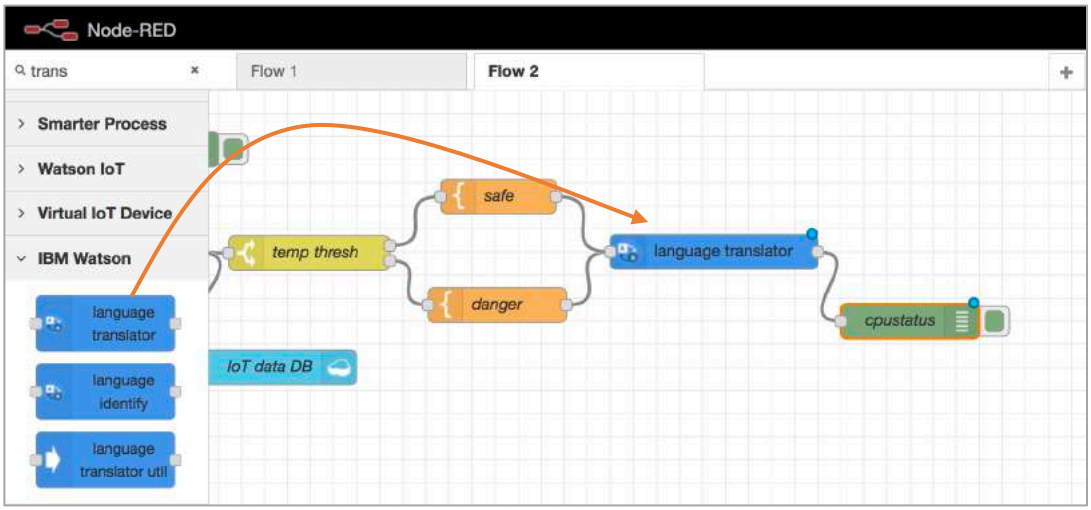


__ 4. Accept the restage step to actually bind the service to the app.

__ 5. While it is restaging (~3 minutes), take a look at **Service Credentials**. This information is useful if you want to invoke your Watson Service from any program (running in IBM Cloud or outside IBM Cloud)

__ 6. Go back to Node-RED window.
If you get a connection error message, your application restaging is not finished. Wait. Try to refresh page.

__ 7. Go back to the Node-RED environment.
Add **Language translator** node and link it between the template (*safe & danger) &* debug nodes (*cpustatus*).



__ 8. Configure the Watson language translator node:

- Name (of your choice)
- Mode: **Translate**
- Domains: **Conversational**
- Source: **English**
- Target: **French** (or Spanish, Portuguese & Arabic)

Note: The user/password fields are not necessary & do not appear in the node settings if a Watson Language Translator service is properly bound to Node-RED application.

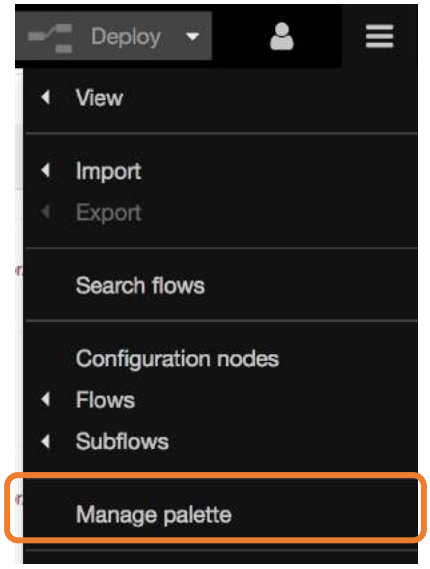
__ 9. **Deploy** your flow & check the logs (**debug** tab).

4. Create a dashboard application in Node-RED

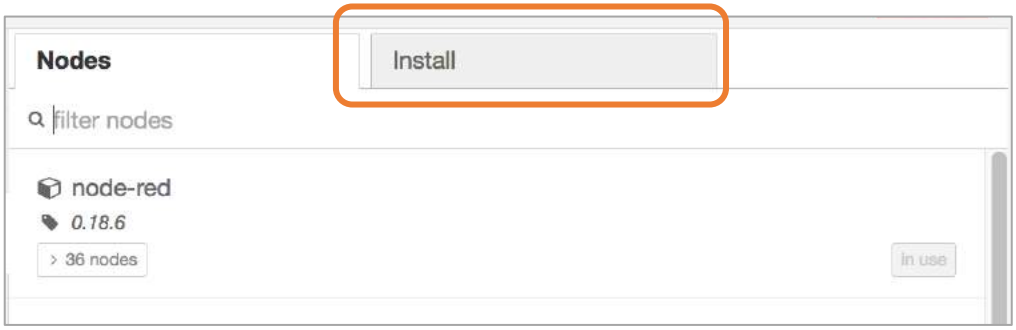
Section 1. Import Node-RED Dashboarding capability

__ 1. At the top right-hand side of the page, click the 'burger' menu:

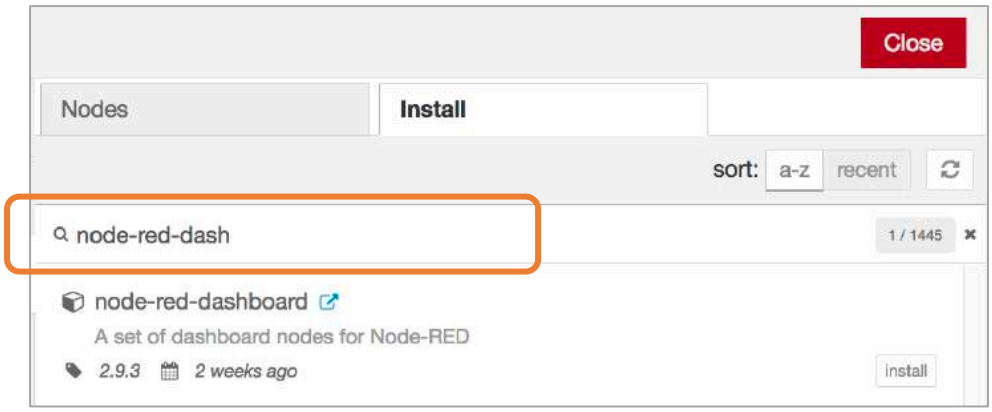
__ 2. Click **Manage palette**:



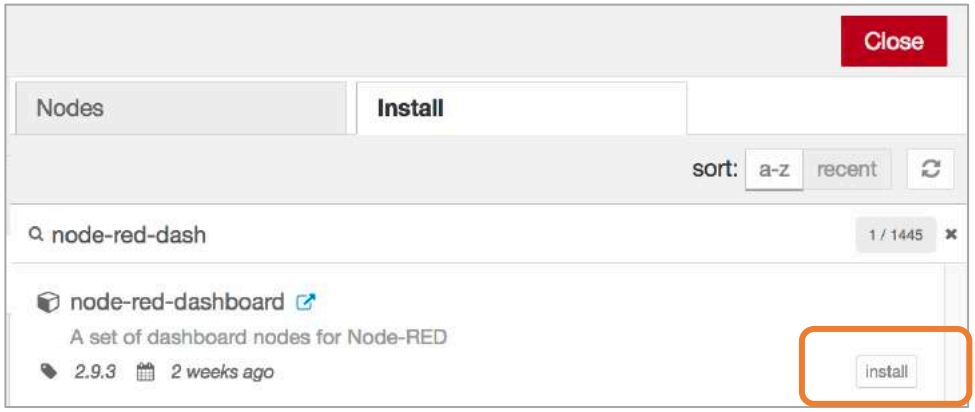
__ 3. In the sidebar that appears on the left-hand side of the page, click the Install tab:



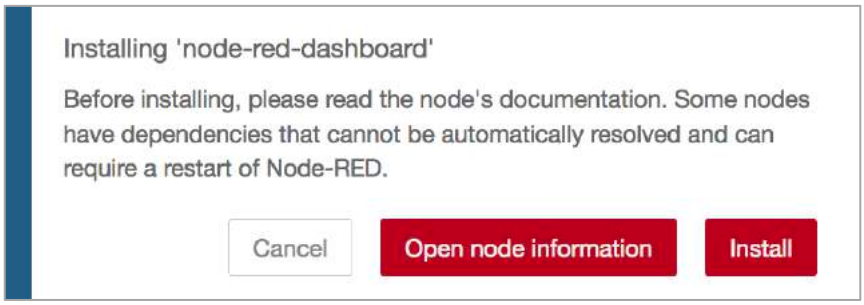
__ 4. In the search field, type **node-red-dash**:



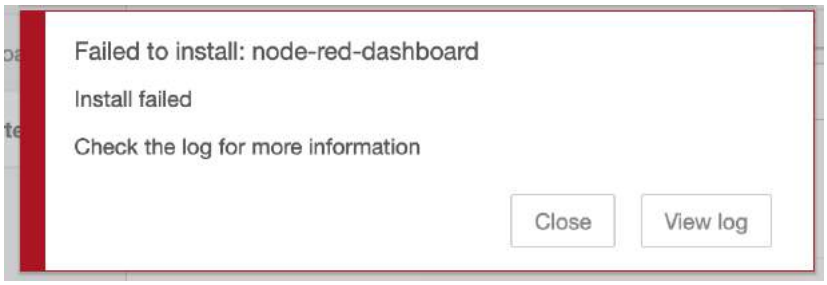
__ 5. Next to the nod-red-dashboard result, click **install**



__ 6. Click **Install**



Note: In the current configuration of your Node-RED application, you probably get an error message



In that case, please use the following procedure to add additional nodes (like node-red-dashboard) to your Node-Red App:

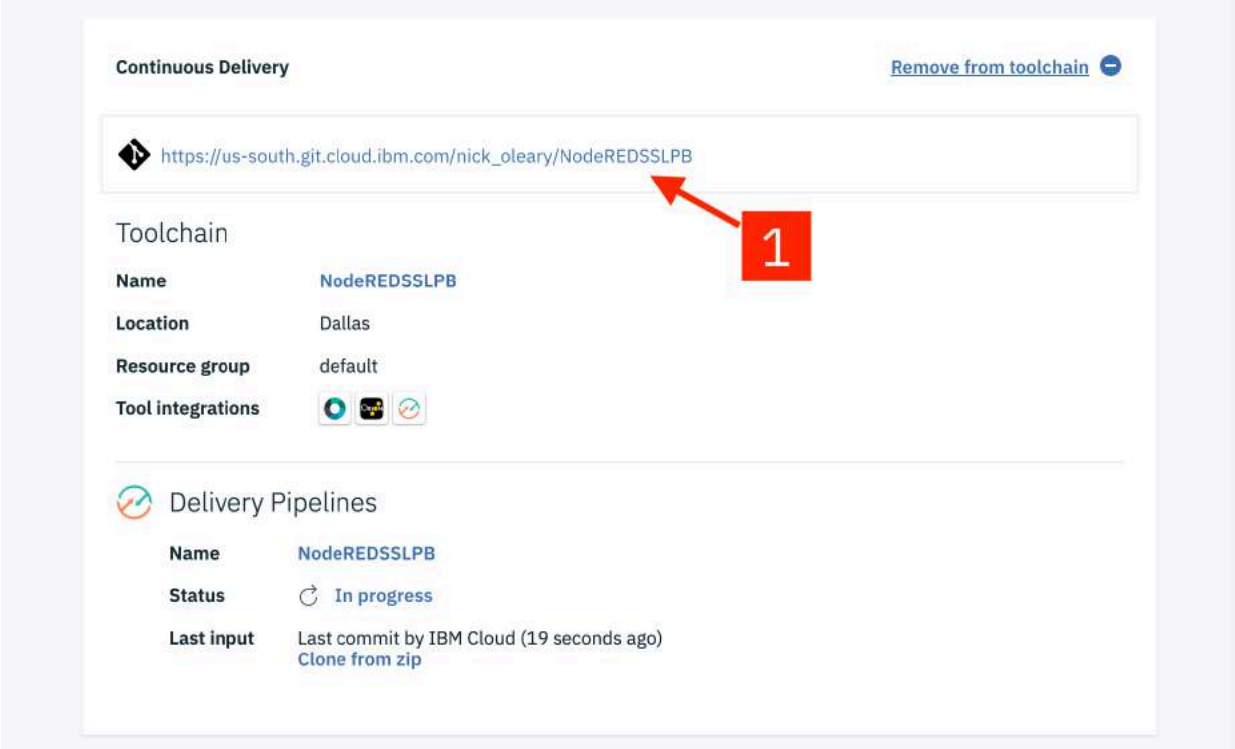
Add extra nodes to your Node-RED palette

Node-RED provides the palette manager feature that allows you to install additional nodes directly from the browser-based editor. This is convenient for trying nodes out, but it can cause issues due to the limited memory of the default Node-RED starter application.

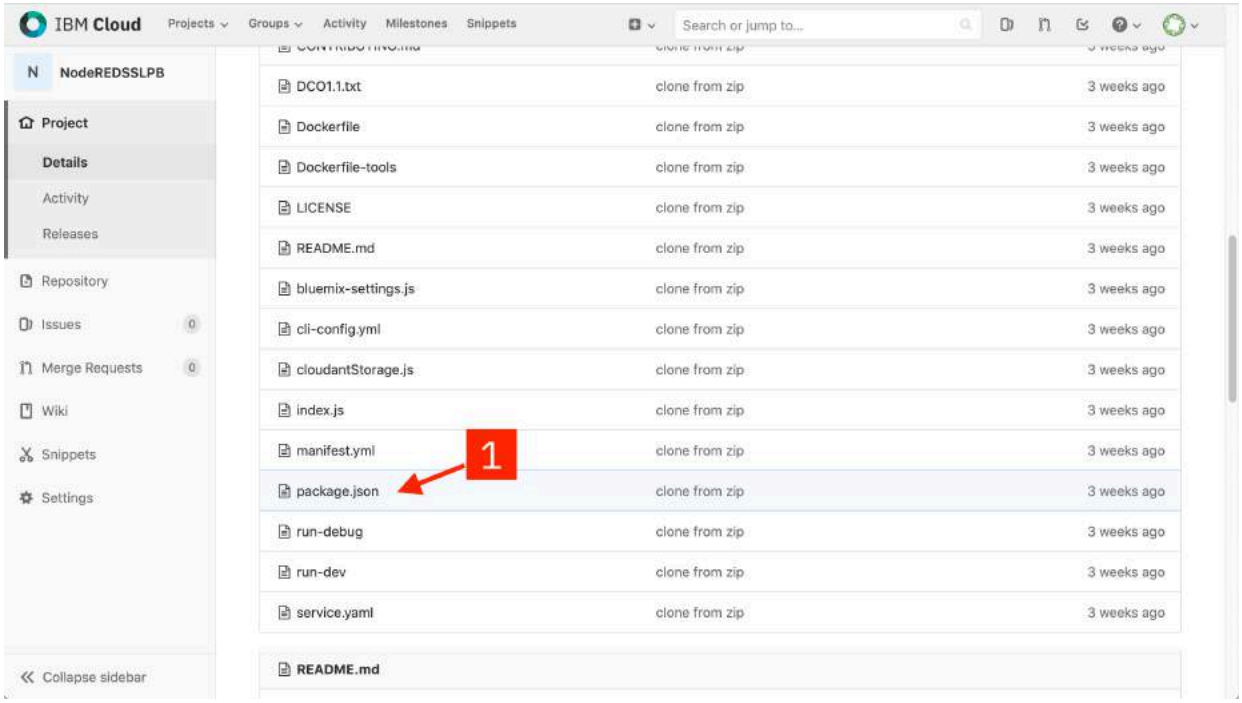
The recommended approach is to edit your application's package.json file to include the additional node modules and then redeploy the application.

This step shows how to do that in order to add the **node-red-dashboard** module.

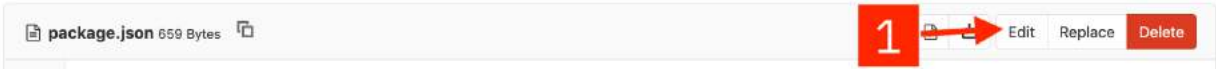
1. On your application's details page, click the url in the Continuous Delivery box. This will take you to a git repository where you can edit the application source code from your browser.



2. Scroll down the list of files and click on **package.json**. This file lists the module dependencies of your application.



3. Click the **Edit** button



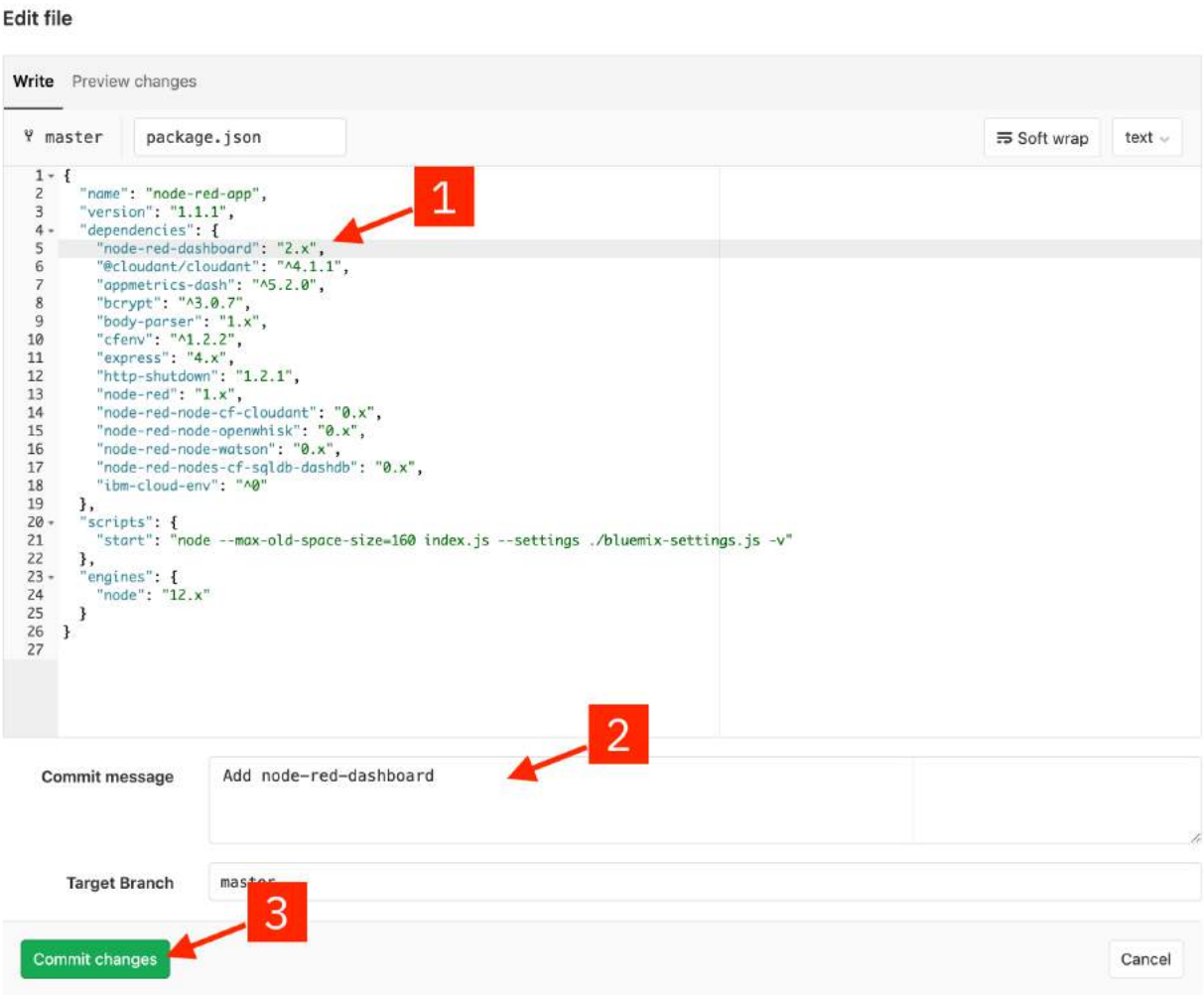
4. Add the following entry to the top of the dependencies section (1):

   "node-red-dashboard": "2.x",

   "node-red-contrib-scx-ibmiotapp" : "0.x"

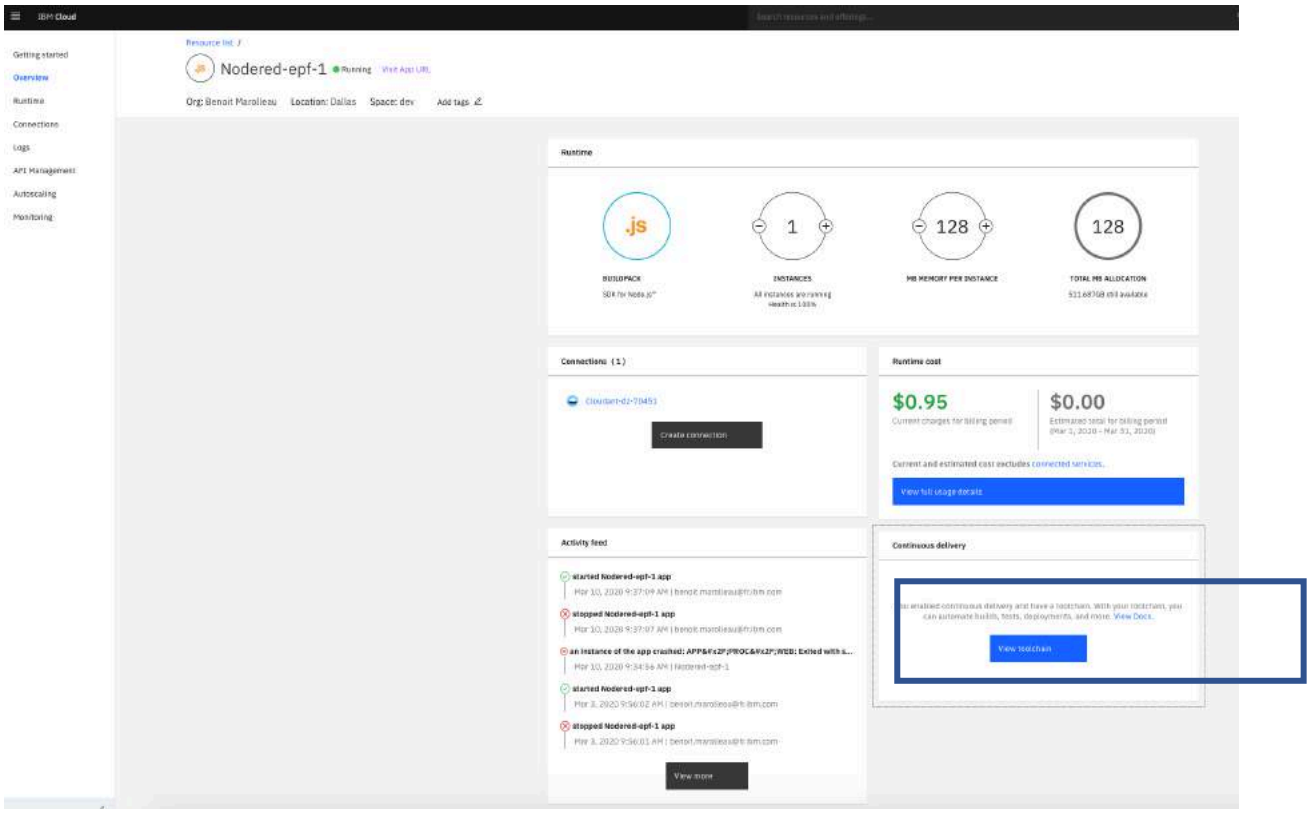
Note: Do not forget the comma (,) at the end of the line to separate it from the next entry. This is an example, you have to copy the list of nodes you want to add. Above, we are adding node-red-dashboard and ibmiotapp nodes.

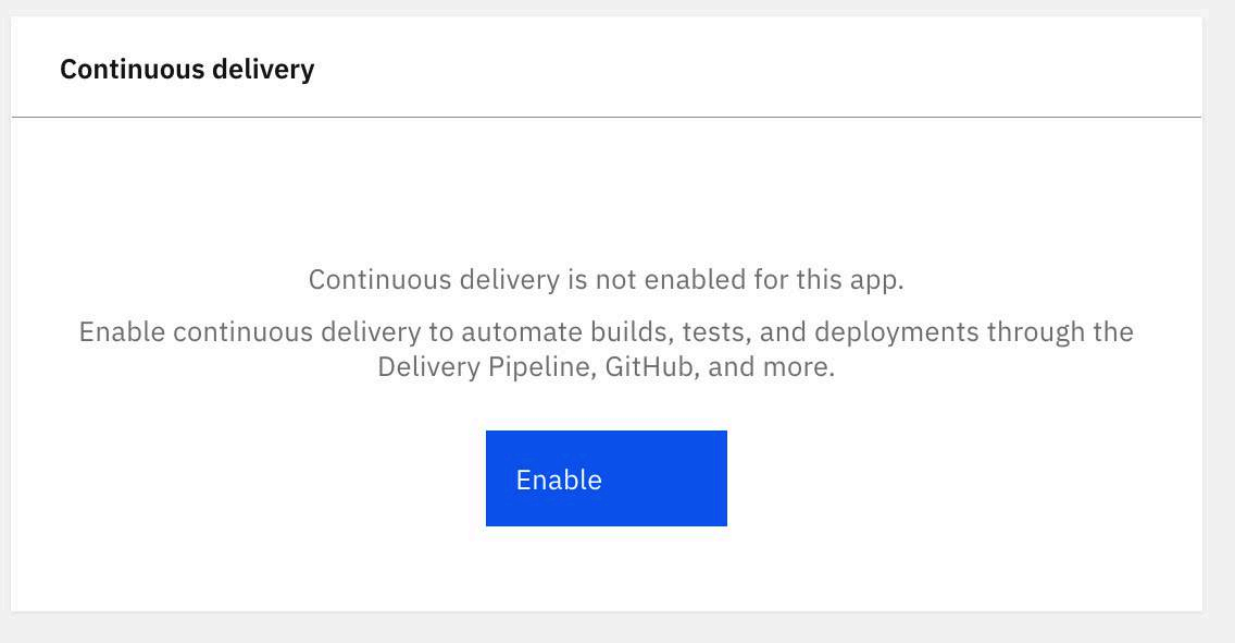
Add a **Commit message** (2) and click **Commit changes** (3)



At this point, the Continuous Delivery pipeline will automatically run to build and deploy that change into your application. If you view the Delivery Pipeline you can watch its progress. The Build section shows you the last commit made (1) and the Deploy section shows the progress of redeploying the application (2).

Note: If your Continuous Delivery Service expired, you'll get a Warning Message. In that case, go to your Resources List, choose your Node-red Cloud Foundry App, delete and recreate the Continuous Delivery Service attached to your app as shown below (Delivery Pipeline part) – specifying Existing Git Repository, unselecting the Issue Tracking check box.





Resource list / Cloud Foundry App /

Continuous Delivery Toolchain

Cancel

Tools:

Dallas

default

Select a CF Organization (deprecated)

Tool Integrations

Git Repos and Issue Tracking
Required

Delivery Pipeline

More tools

The Delivery Pipeline automates continuous deployment.

IBM Cloud API key:

New +

Description:

Pipeline for Node RED MEE

Resource list / Cloud Foundry App /

Continuous Delivery Toolchain

Cancel Create

Tools:

Tool Integrations

Git Repos and Issue Tracking

Delivery Pipeline

More tools

Git repos and issue tracking hosted by IBM and built on GitLab Community Edition.

Server:

US South (https://us-south.git.cloud.ibm.com)

Authorized as benoit.marolleau with access granted to zero US South group(s)

Repository type:

Existing

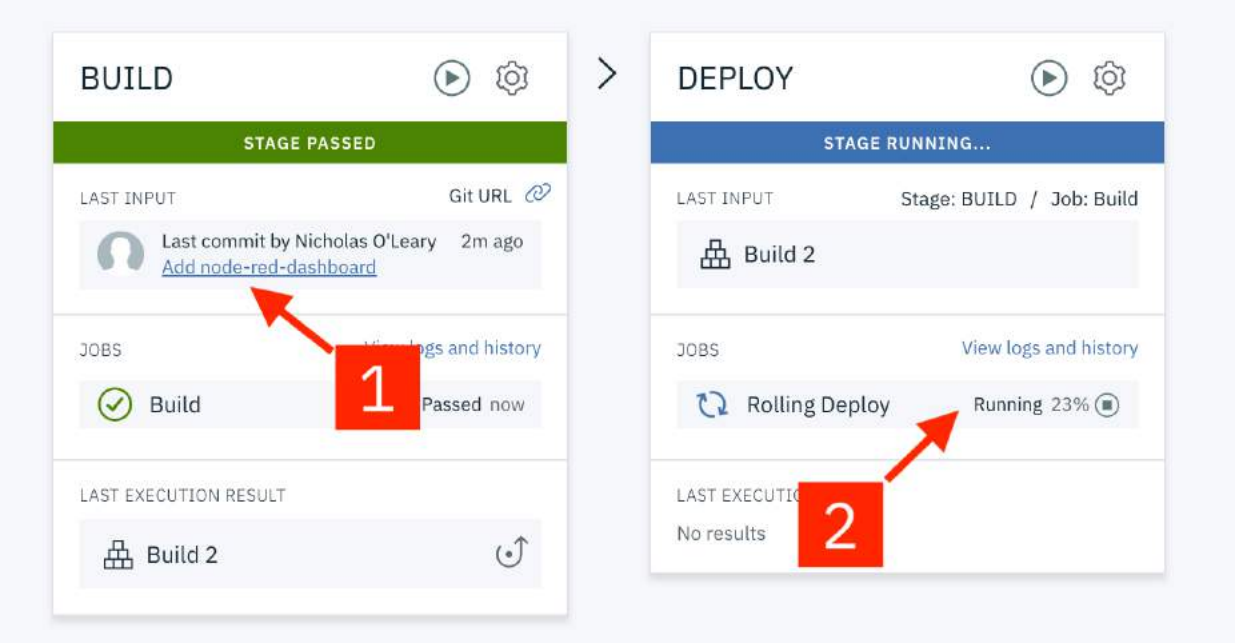
Link to the repository that is specified in the Repository URL field.

Repository URL:

https://us-south.git.cloud.ibm.com/benoit.marolleau/Nodered-epf-1

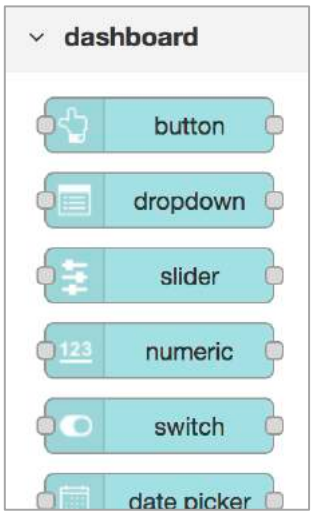
Enable Issues

Track deployment of code changes



Once the Deploy stage completes, your application will have restarted and now have the node-red-dashboard nodes preinstalled.

__ 7. Note the additional dashboard nodes on the palette : **Dashboard** category.

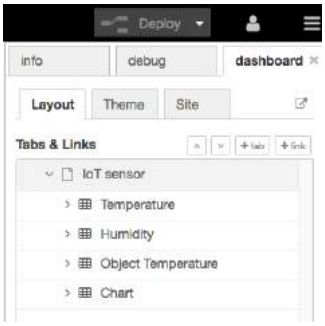


__ 8. Note also that there is a new dashboard tab in the right-hand sidebar:

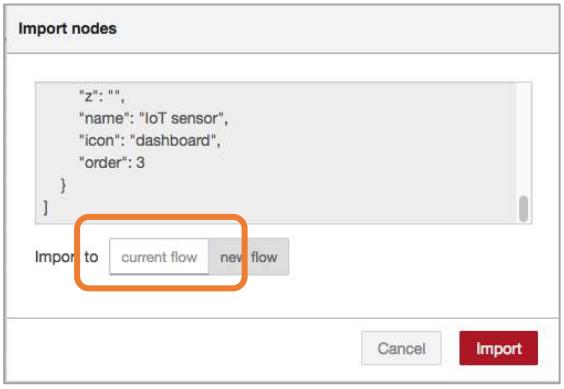
TIP: This dashboard tab may be used to add new tabs, menus etc. to the visualization dashboard. There are also two available themes – light and dark.

Section 2. Create a simple Node-RED Dashboard

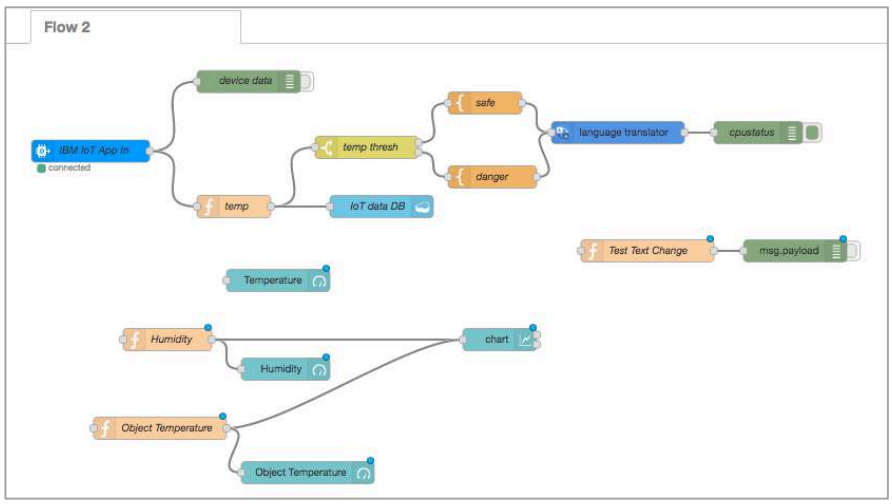
In this section, you will create a simple dashboard for sensor data using new dashboard nodes installed.



__ 1. In the current Node-RED tab, import the file named **Lab3_IoT_Dashboard.v2.0.txt** (previously downloaded): **Menu** > **Import** > **Clipboard**.
Click **current flow** then **Import**.

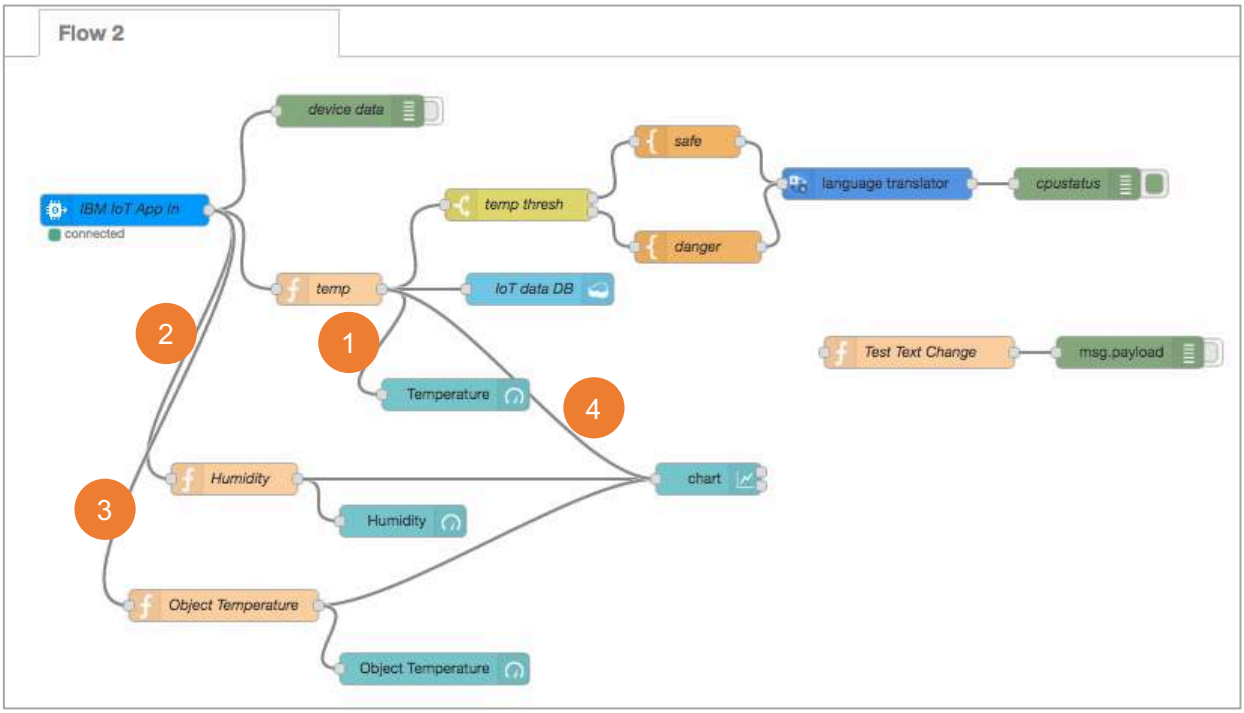


__ 2. Click on workspace to paste new nodes.



__ 3. Connect new nodes to existing…

1. **Temperature** to **temp**
2. **Humidity** to **IBM Iot App In**
3. **Object Temperature** to **IBM Iot App In**
4. **chart** to **temp**

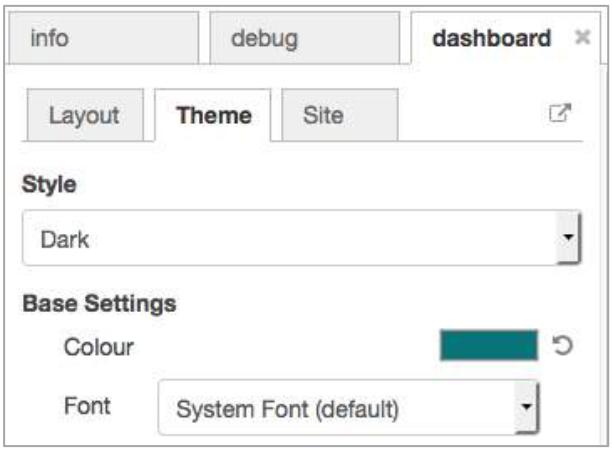


__ 4. Deploy the flow: **Deploy**

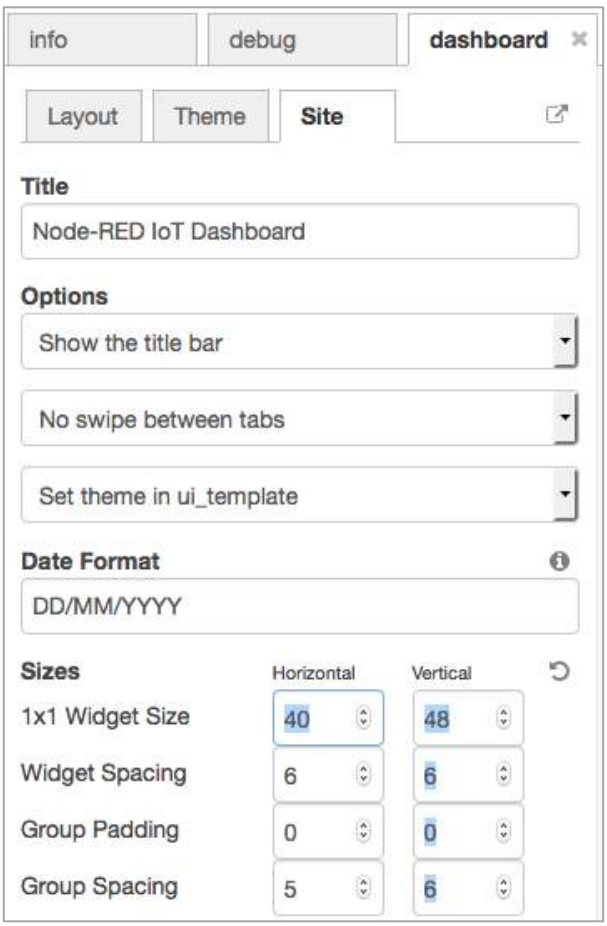
__ 5. Connect to **http://<YOUR_APP_HOSTNAME>/ui** (URL generated by Node-red-dashboard node) to see your new dashboard. Change value on virtual sensor app. to see impact on gauges and lines.

__ 6. Customize your dashboard using **Dashboard** tab

- **Theme**: use Dark style



- **Site**: as below, change **Title**, **1x1 Widget Size**, **Group spacing**

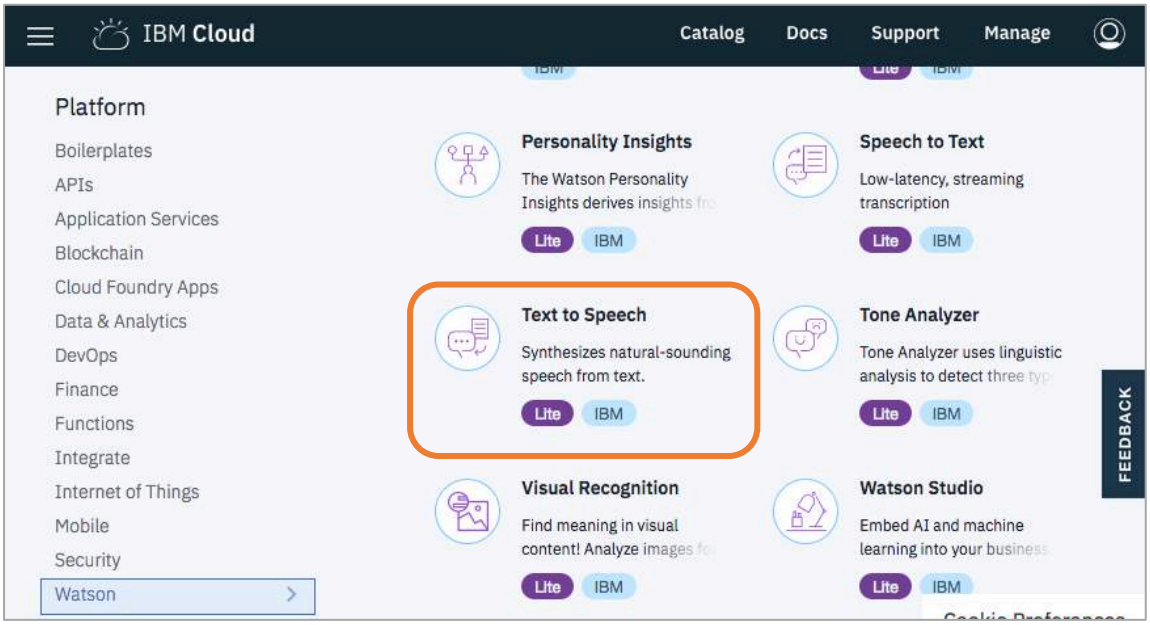


__ 7. Deploy and check your dashboard.

Section 3. Add voice alert on dashboard (Optional)

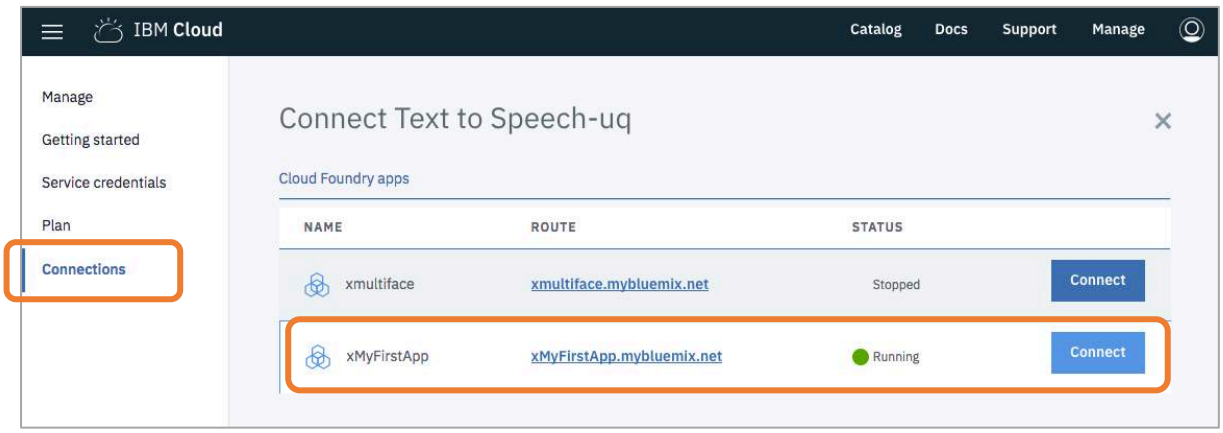
In this section, you will add a voice node allowing your app to tell say message when temperature change. To do that, you will deploy a new Watson service: Text to Speech.

__ 1. In the IBM Cloud window, click **Catalog** then **Watson** category. Click on **Text to Speech** service to create your own instance.

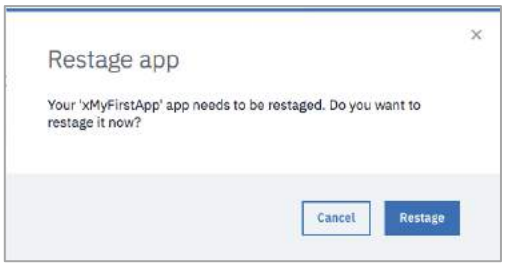


__ 2. Enter a name for your service and click **Create**.

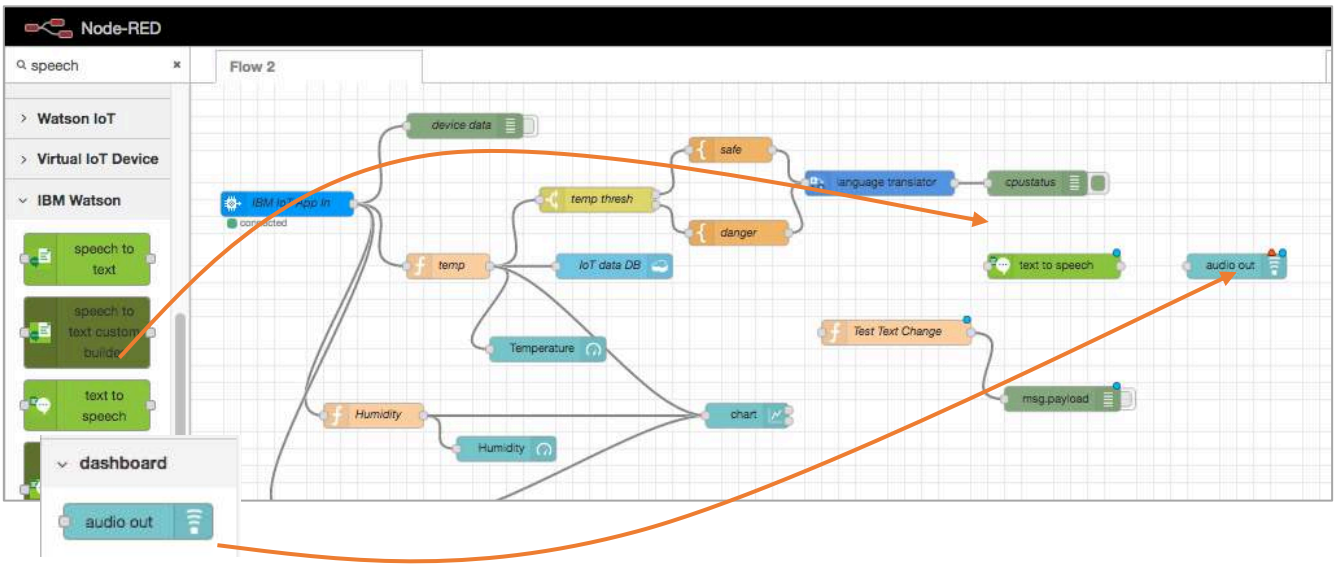
__ 3. On new service dashboard, click **Connection**, and connect Text to speech service to your Node-RED application: **Connect**.



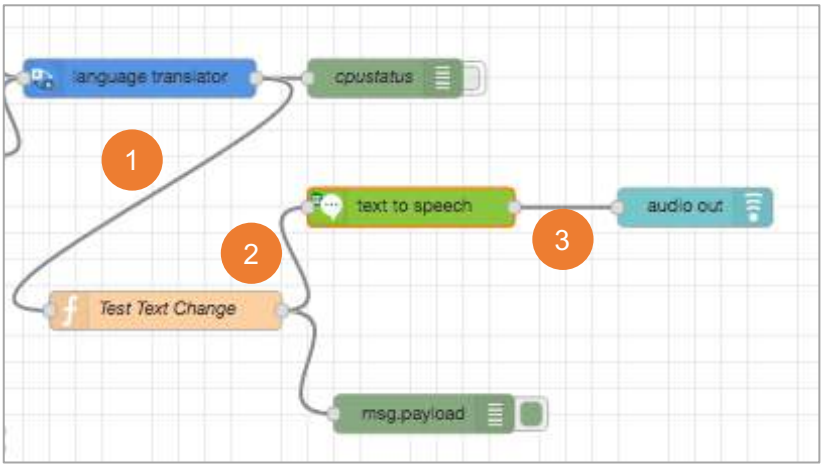
__ 4. Accept to restage application and wait for your Node-RED application to restart.



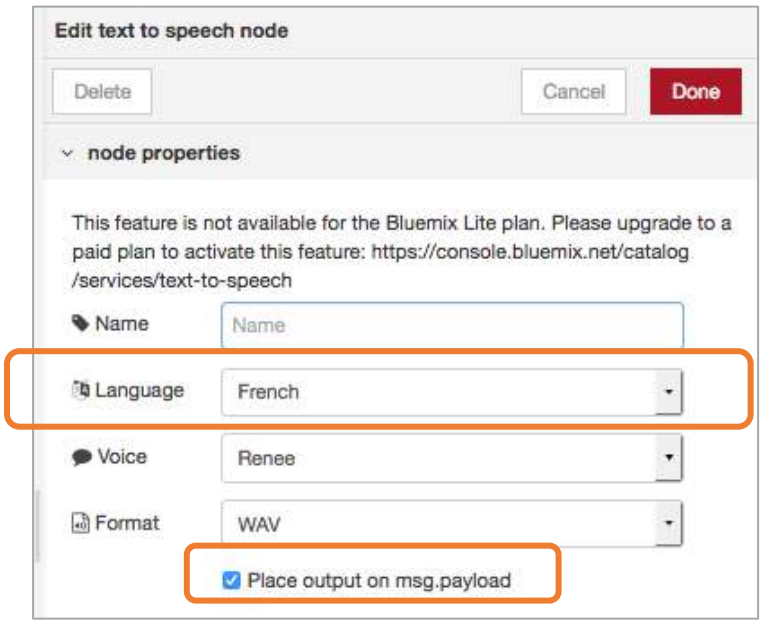
__ 5. When restarted, in Node-RED environment, add an **Audio out** node (category **Dashboard**) and a **text to speech** node (category **Watson**) to your flow



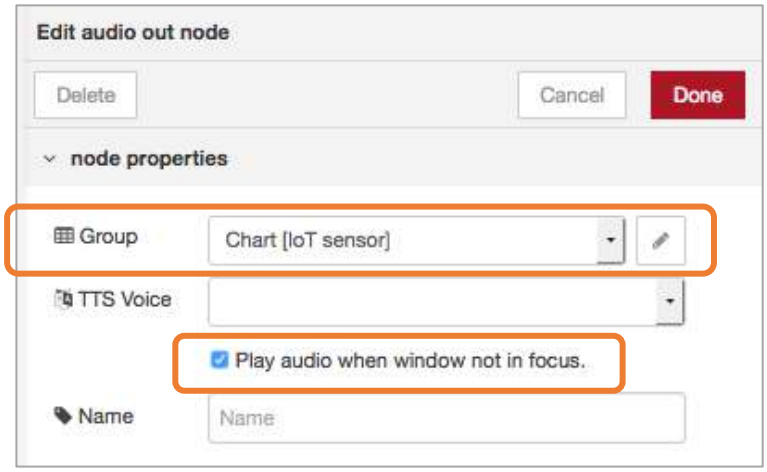
__ 8. Connect nodes as below
1. **Test text change** to **language translator**
2. **Test text change** to **text to speech**
3. **text to speech** to **audio out**



__ 9. Configure **text to speech** node to
- use the language your translated to (as configured before in **language translator** node)
- place the output on msg.payload (check box)



__ 10. Configure **audio out** node
- Set **group** to **chart [IoT sensor]**
- Check **Play audio when window not in focus**



__ 11. Deploy the flow: **Deploy**

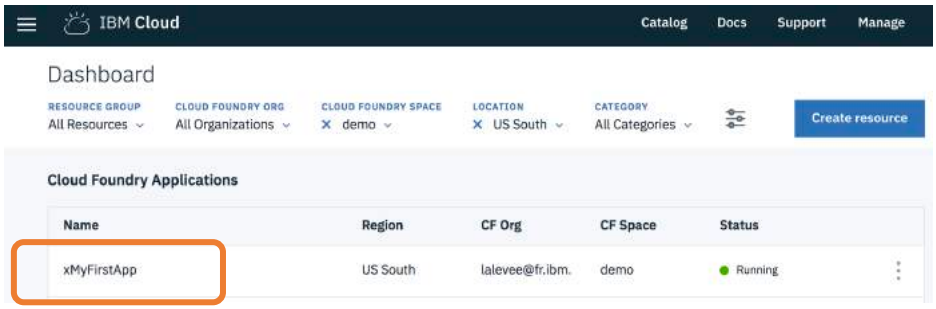
__ 12. Do you hear something ?
Try to change temperature in virtual sensor app.

5. Create a dashboard in Watson IoT Platform (optional)

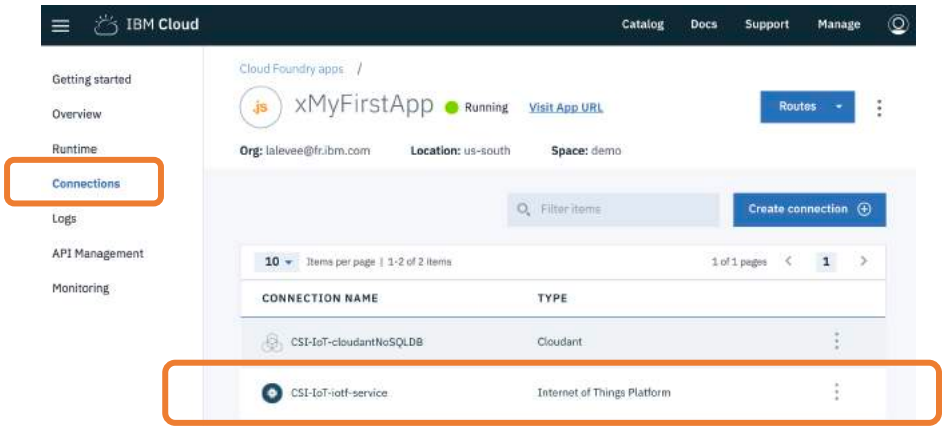
Section 1. Create new device in Watson IoT Platform

__ 1. Switch back to your browser window open on IBM Cloud environment.

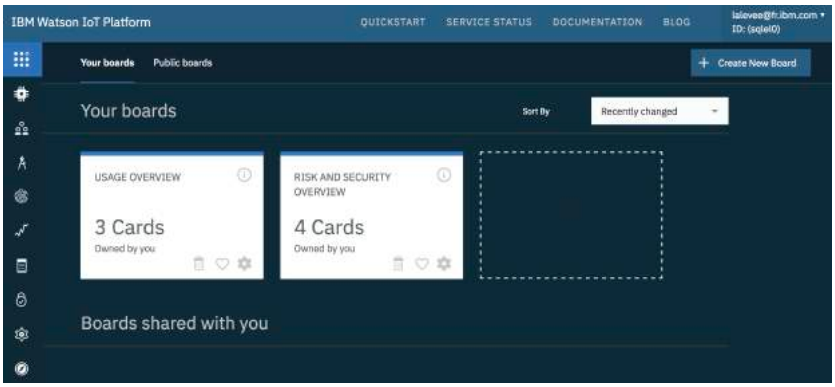
__ 2. From your dashboard, click on name of application created in exercise 2, to open application dashboard.



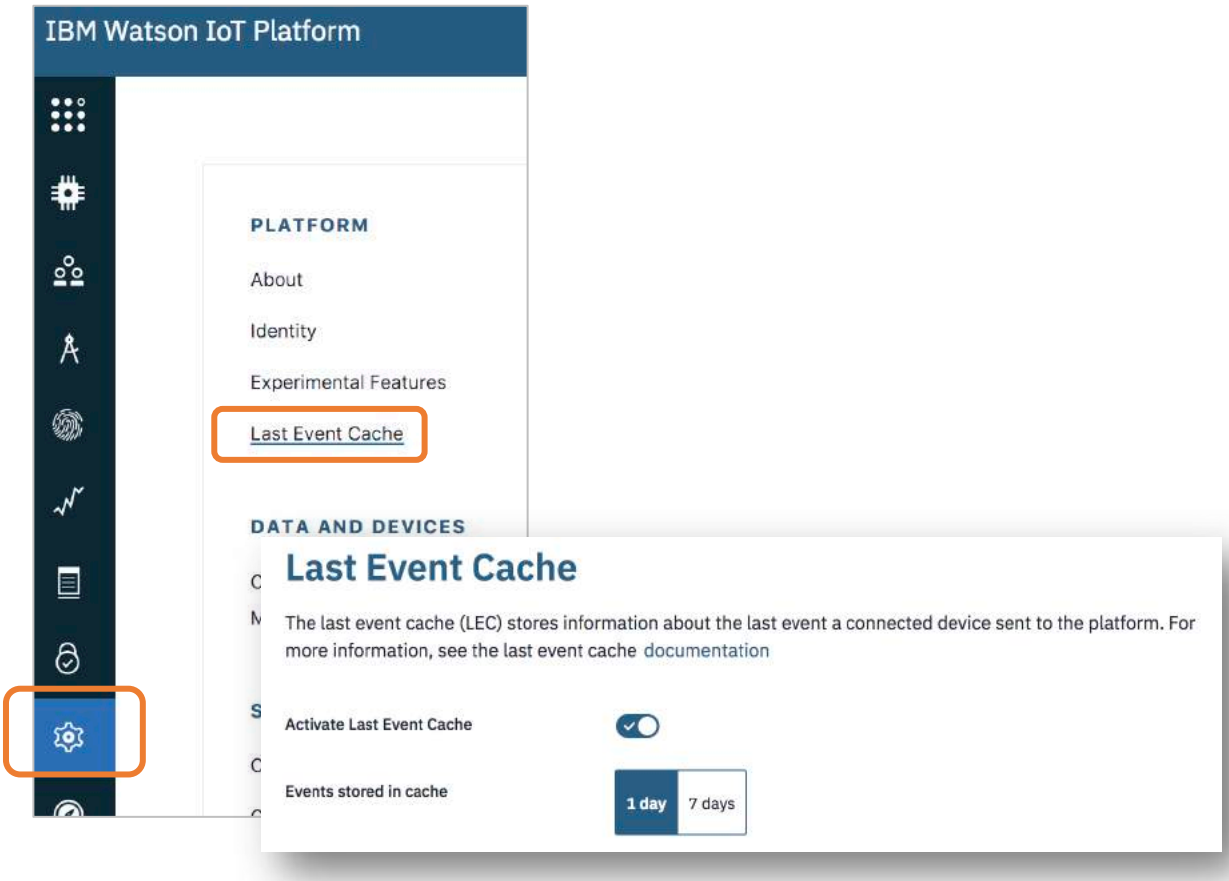
__ 3. From left panel, click on **Connections** to see bound services.
Click **Internet of Things Platform** service.



__ 4. Click **Launch** button to open your Watson IoT organization dashboard in a new browser tab. You are now connected to the IBM Watson IoT Platform dashboard. With the platform you can manage your devices, store and access your data.

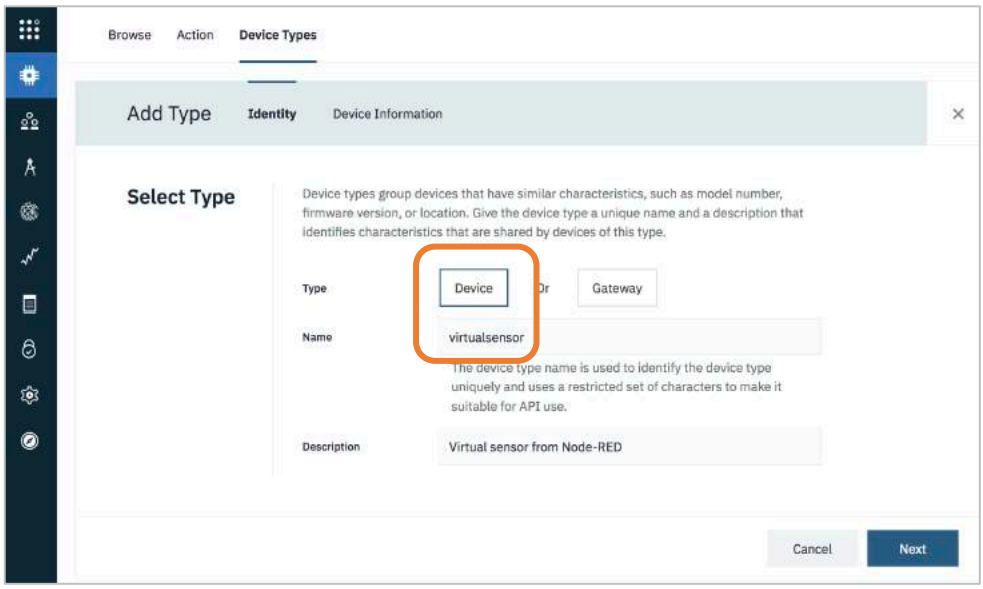


__ 5. From left panel **Settings** menu, activate the **Last Event Cache** feature :
By using the Watson IoT Platform Last Event Cache API, you can retrieve the last event that was sent by a device. This works whether the device is online or offline, which allows you to retrieve device status regardless of the device's physical location or use status. Last event data of a device can be retrieved for any specific event that occurred up to 7/45 days ago.



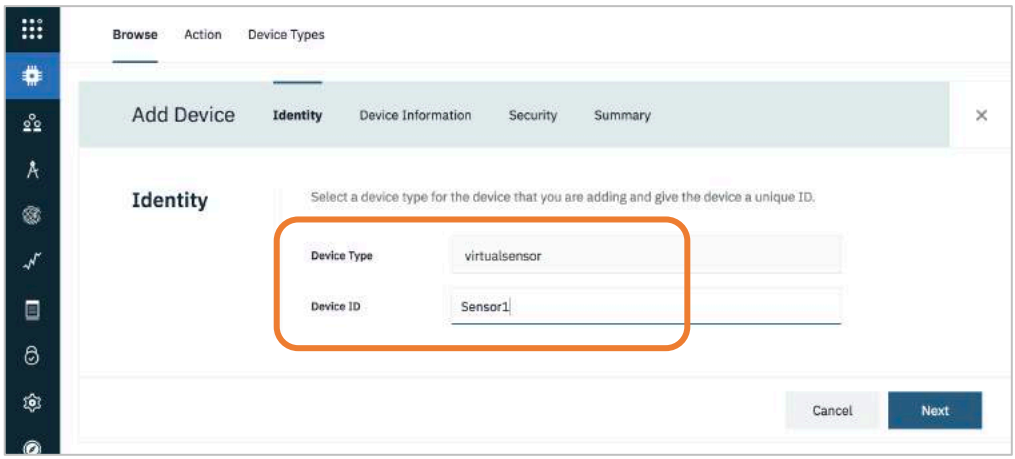
__ 6. You are now going to register a new device in your organization: first adding a device type, then the device. Click on **Devices** menu from left panel.
Click on **Device Type**, then **Add device Type** button

__ 7. Choose **Device** and put a name (case sensitive) for your device : *virtualsensor*
Click **Next**.
You don't need to add Device Information.
Click **Done**.



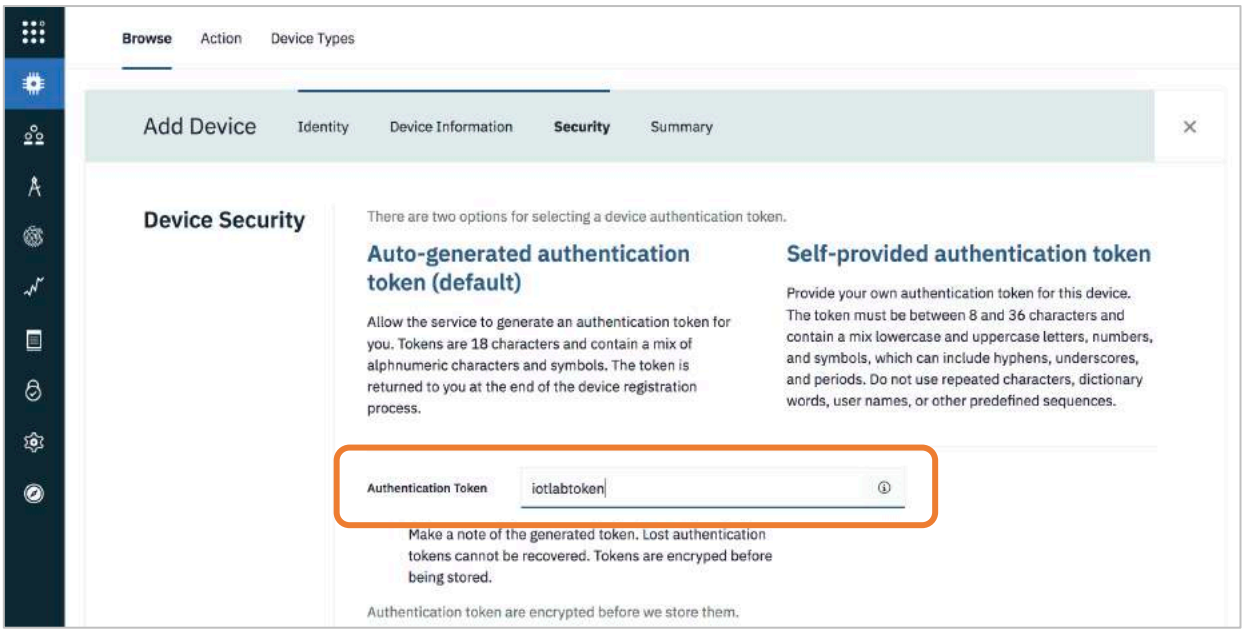
__ 8. Click **Browse** and click on **Add Device** button on the right

__ 9. Select **Device Type** you created before.
Give **Sensor1** as device' name (it will be your **Device ID**), click **Next**.



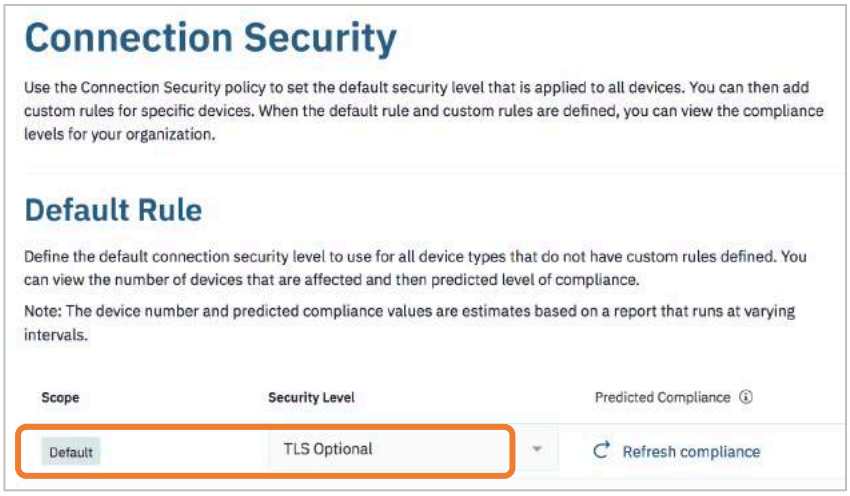
__ 10. Click **Next** again: you don't need any metadata.

__ 11. For the security part, it is recommended to you to provide a simple token (between 8 and 36 characters long and should contain a mix of lower and upper-case letters, numbers and symbols). If you skip this, a token will be automatically generated but this one won't be easy to use for the next steps of this hands-on.
Click **Next**.



__ 12. A summary of your device details appears. Copy all these information in a text editor or as a screenshot. The token is unrecoverable.
Click **Done**.

__ **13.** In the **Security** menu on the left panel, click **Connection Security** and change security settings to accept non-SSL connections : **TLS Optional**

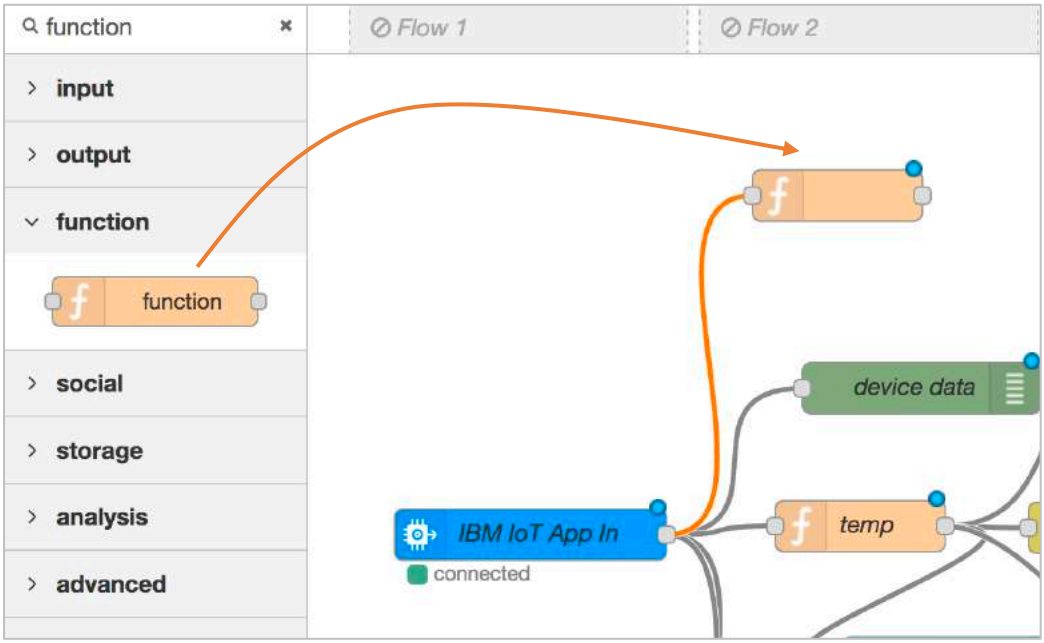


__ 14. Your device is created. You are now going to update your Node-RED application.

Section 2. Node-RED: redirect sensor data to IBM IoT Platform

__ 1. Switch back to your browser window opened on Node-RED development interface.

__ 2. Add a **Function** node and connect it to **IBM IoT App In** existing node

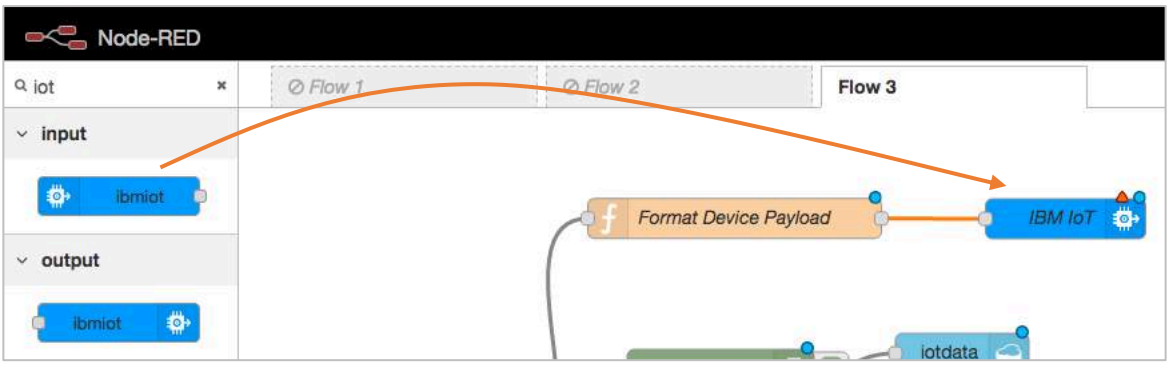


__ 3. Open new node. Name it **Format Device Payload** and insert following code

```
// Create MQTT message in JSON
msg = {
  payload: JSON.stringify(
    {
      d:{
        "temp" : msg.payload.d.temp,
        "humidity" : msg.payload.d.humidity,
        "objectTemp": msg.payload.d.objectTemp
      }
    }
  )
};
return msg;
```

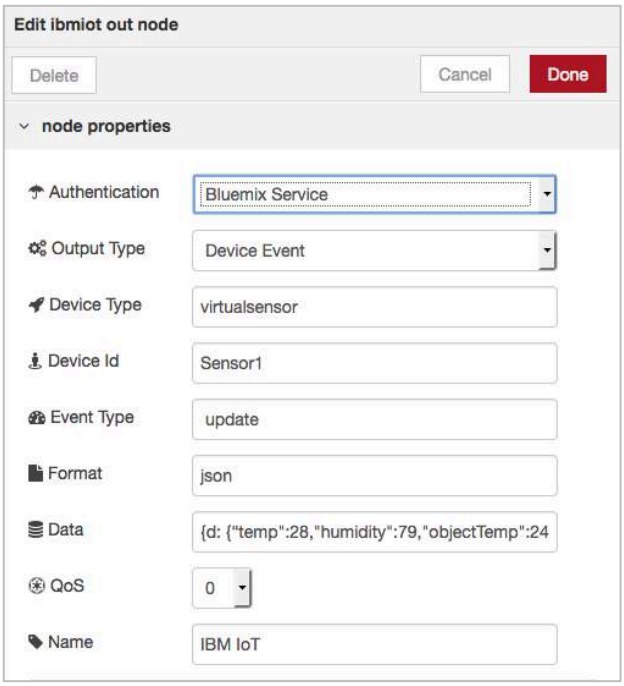
Click **Done.**

__ 4. Add an **ibmiot out** node and connect it to **Format Device Payload** node**.**



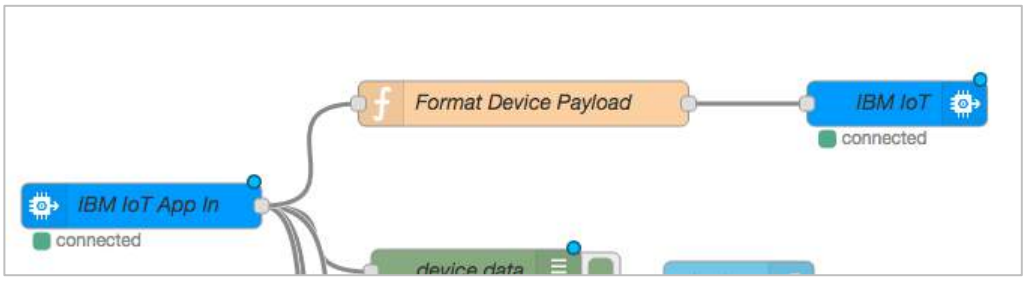
__ 5. Open **ibmiot out** node and configure it as below**.**

- Authentication : **Bluemix Service**
- Output Type: **Device Event**
- Device Type: virtualsensor (cf. Section 1, step 7)
- Device Id: Sensor1 (cf. Section 1, step 9)
- Event Type: update
- Format: JSON
- Data: *<json sample>*
  (Enter simple JSON sample like {d:{“temp":28,"humidity":79,"objectTemp":24}}
- Name: Send to IoT Platform



Click **Done.**

__ 6. Click **Deploy** to deploy your updated Node-RED flow **: ibmiot** nodes should now be connected.

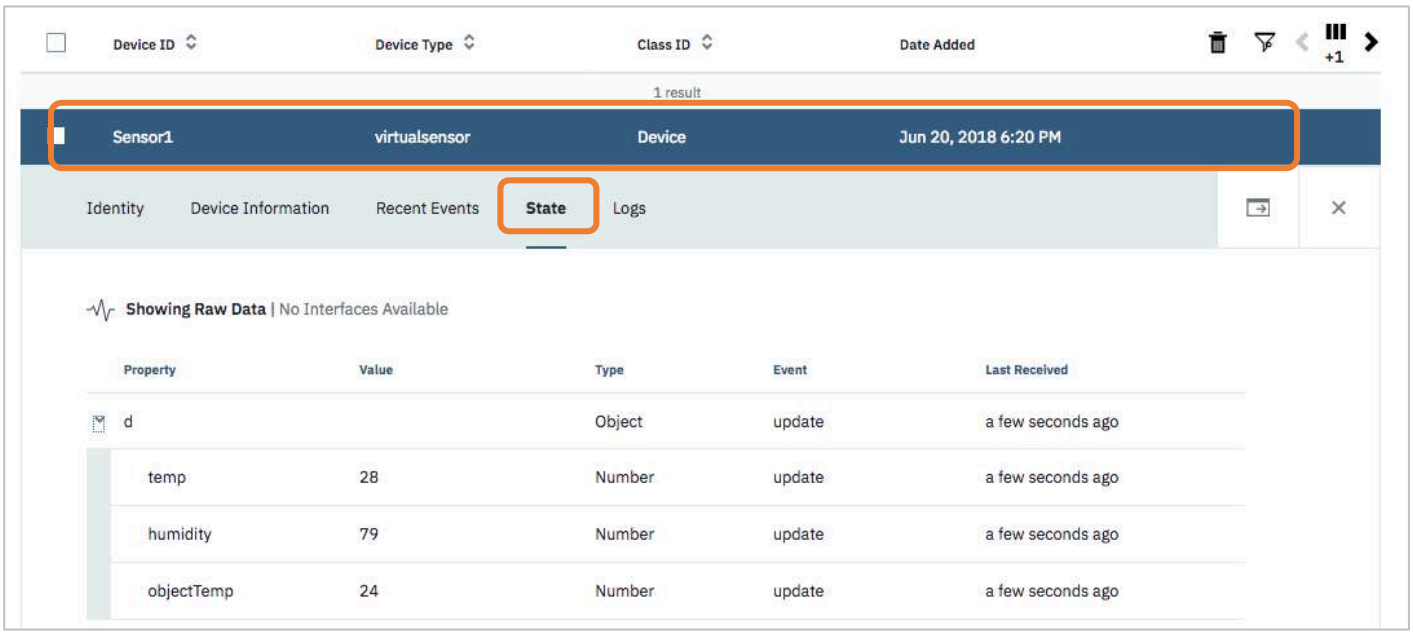


Section 3. IBM Watson IoT Platform: create dashboard

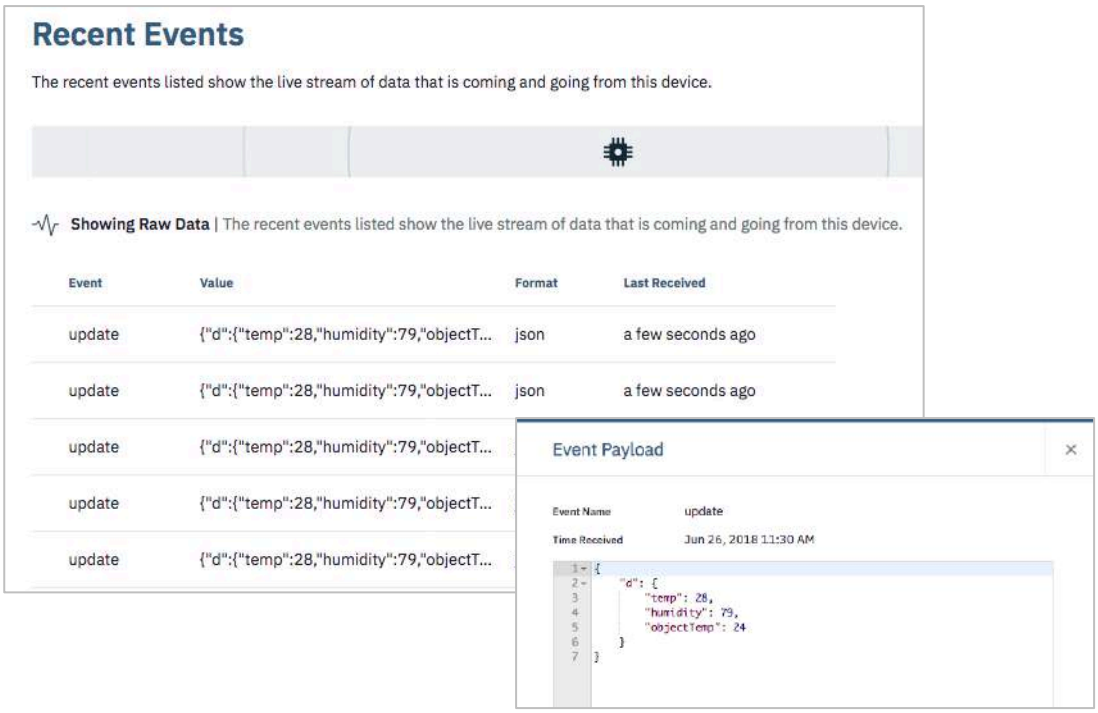
__ 1. Switch back to your browser window opened on IBM Watson IoT Platform interface.

__ 2. In left panel, click **Device** menu

__ 3. Click on **Sensor1**, your previously created device. Click on **State** to see the last device status, and last values of your Sensor1 device (remember that Sensor1 receives data from Node-RED, Node-RED receives data from your IoT sensor Web app).

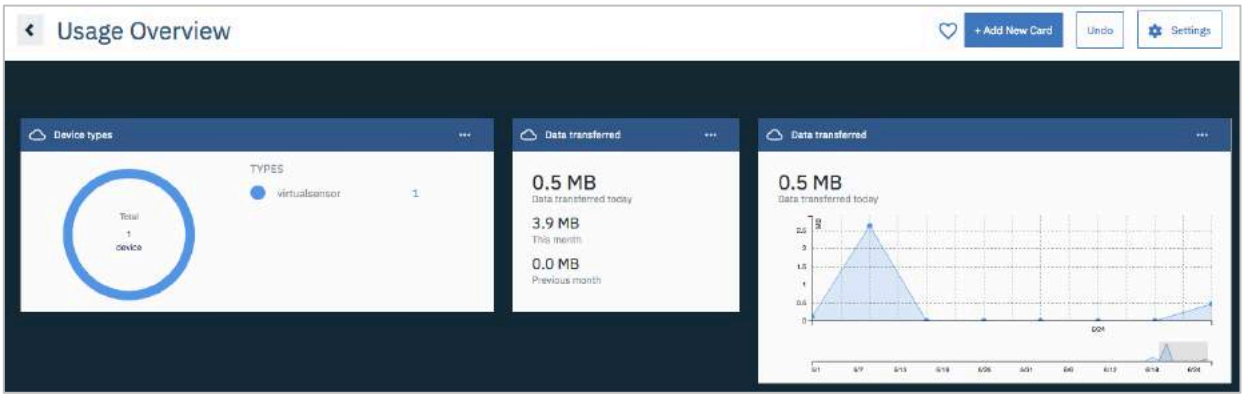


__ 4. You can also click on **Recent events** to see the last 5 payloads



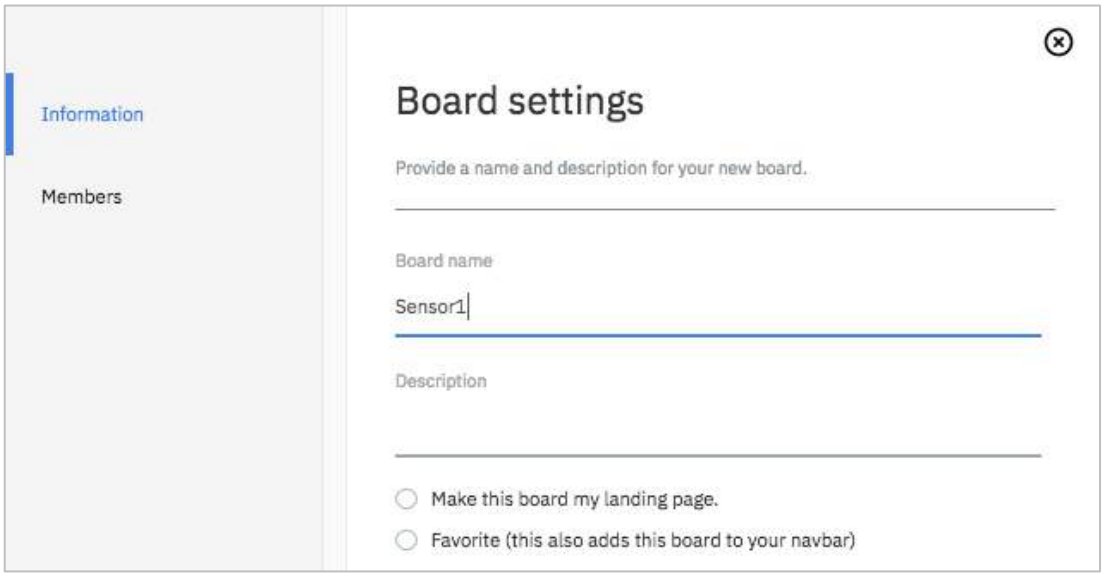
__ 5. We are now going to create a simple dashboard using IBM Watson IoT Platform. From left panel, click on **Board**.

__ 6. You can see that you already have some card. Have a look at **Usage Overview**. You can see cards about devices connected and data transferred.



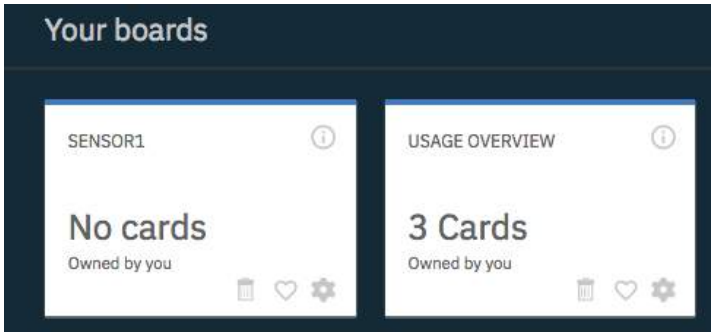
__ 7. Click **<** (top left corner).
Click on **Create New Board.**

__ 8. Click **<** (top left corner).
Click on **Create New Board.**
Enter a name to your new board: *Sensor1*

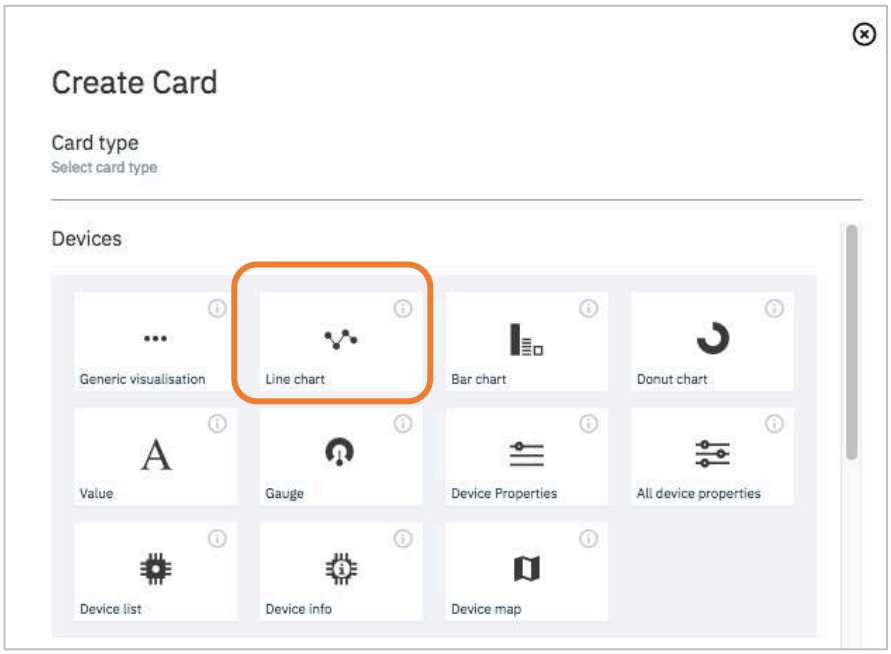


__ 9. Click **Next**, then **Summit.**

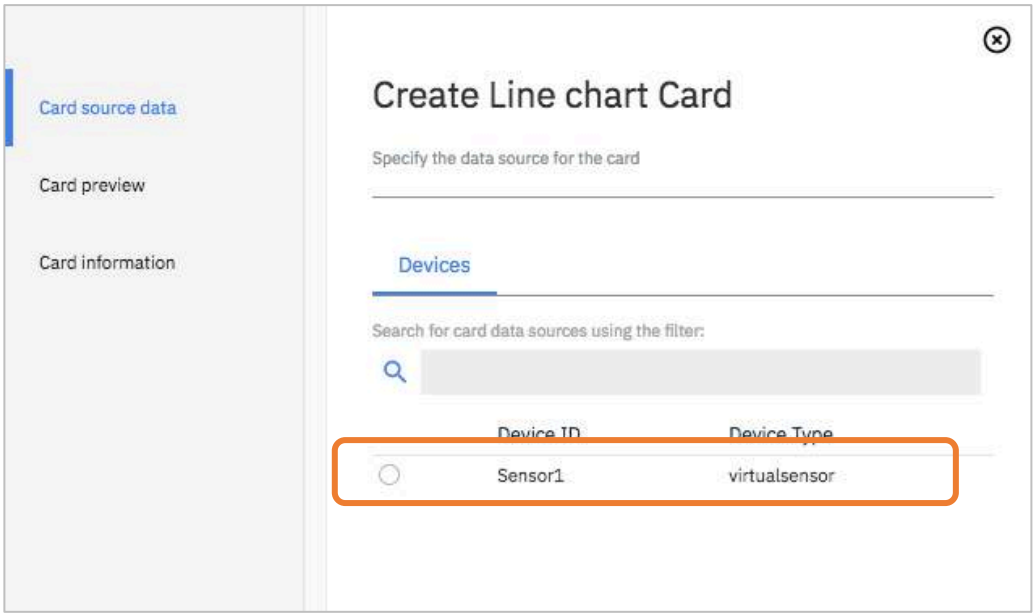
__ 10. Click on your new board



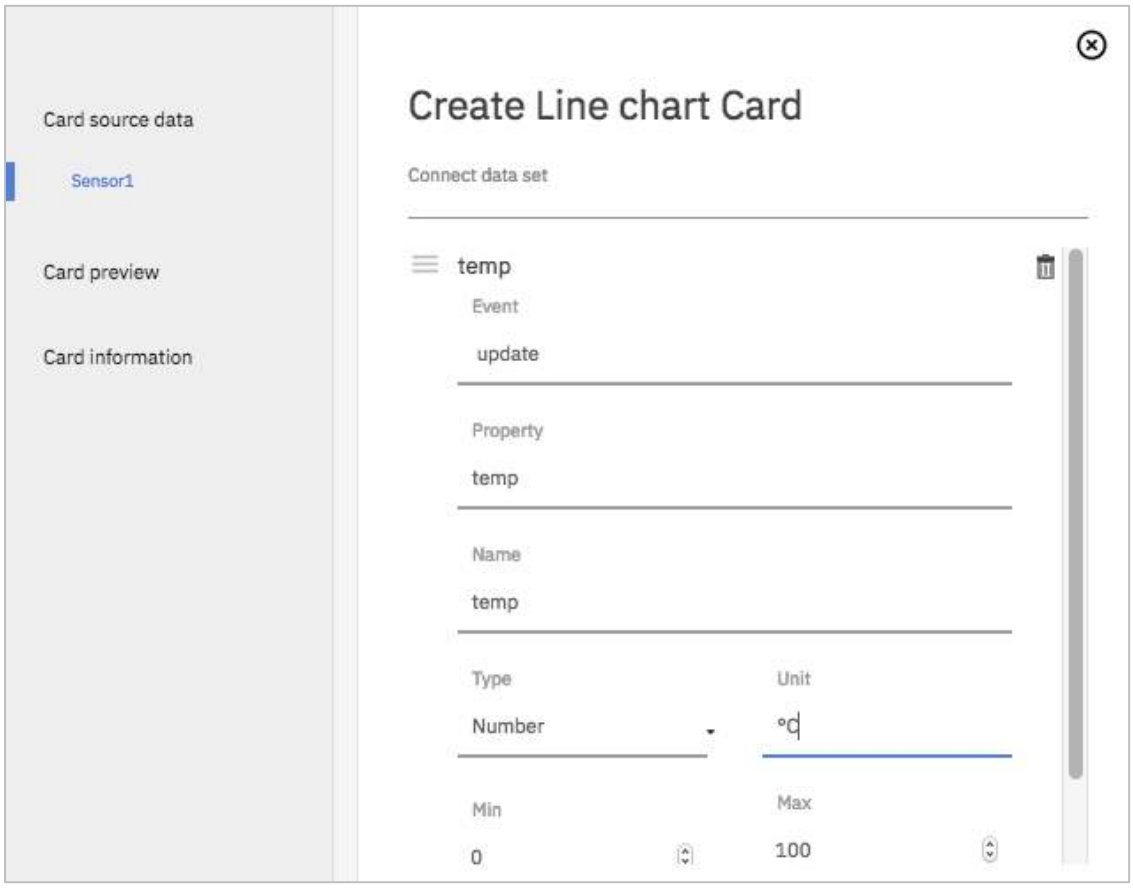
__ 11. Click on **Add new card** button
Select **Line Chart**. Click **Next**.



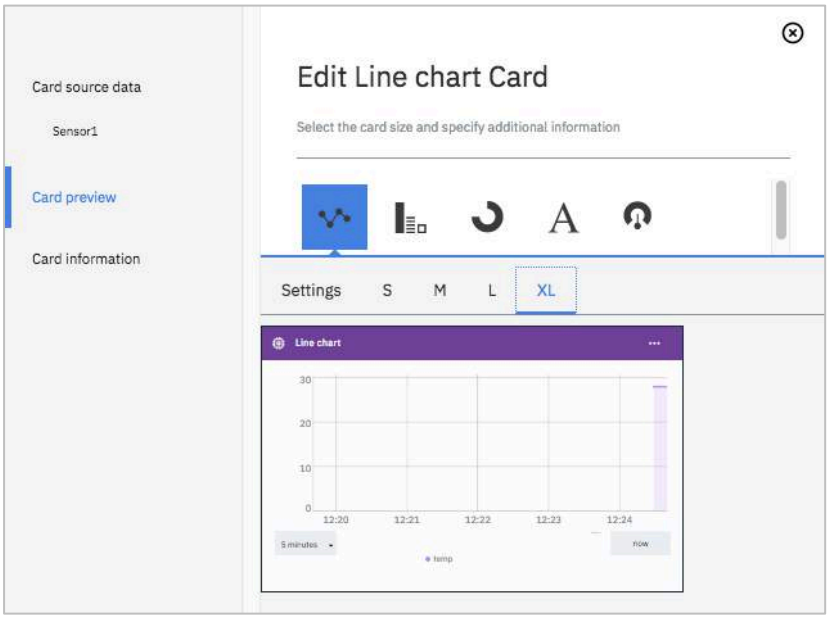
__ 12. To collect data from your device, select Sensor1. Click **Next**.



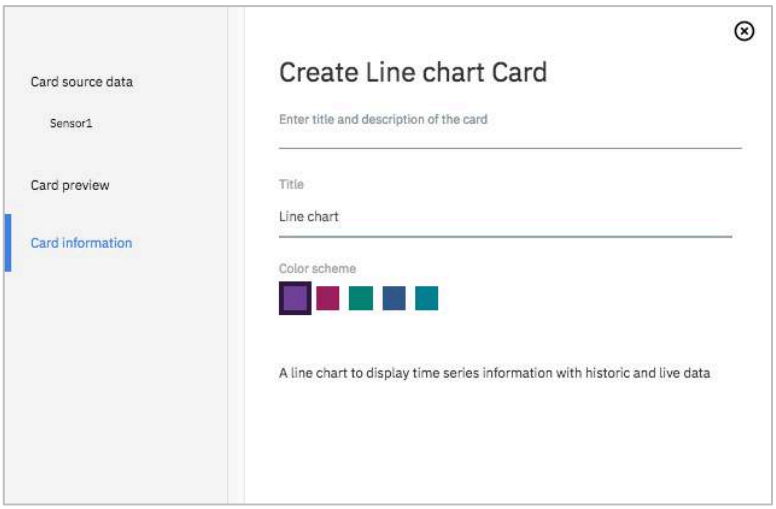
__ 13. To select value to display, click on Connect new data set, and fill form as below. Click **Next**.



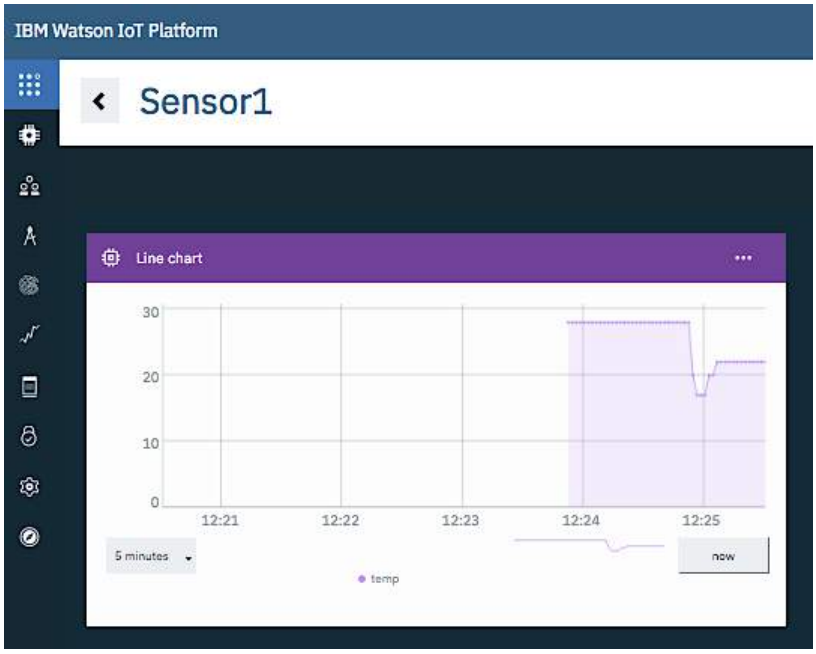
__ 14. Change the card size to XL (you can change others parameters, if you want). Click **Next**.



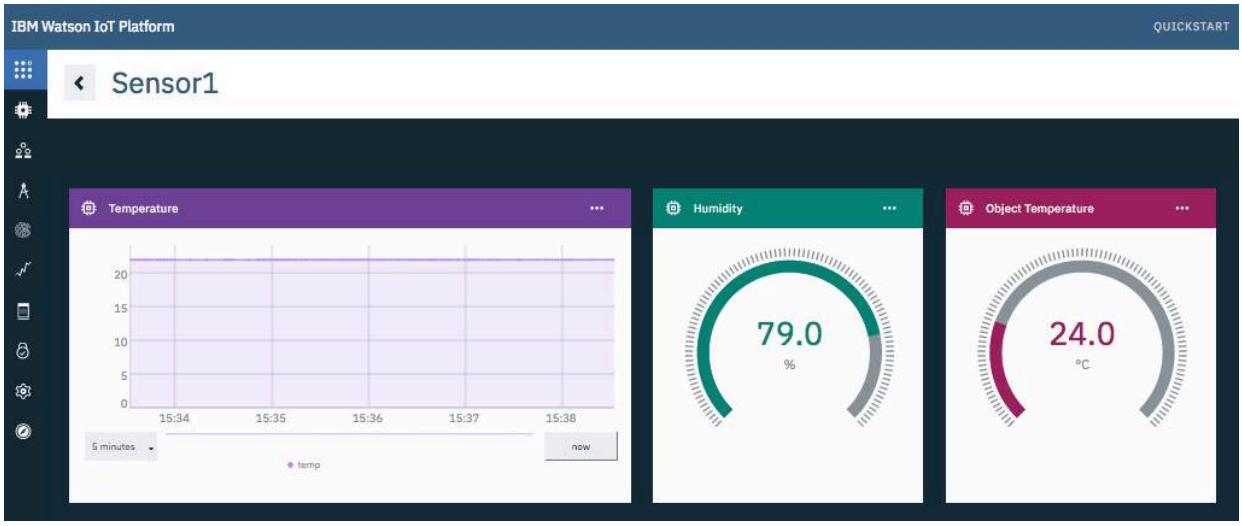
__ 15. Enter a card title. Click **Summit**.



__ 16. Your new card is available and displays live temperature values from Sensor1



__ 17. Following same steps (from 11 to 16), create 2 cards (gauges) to display **Humidity** and **Object temperature**.



6. Physical Device – MQTT Connection

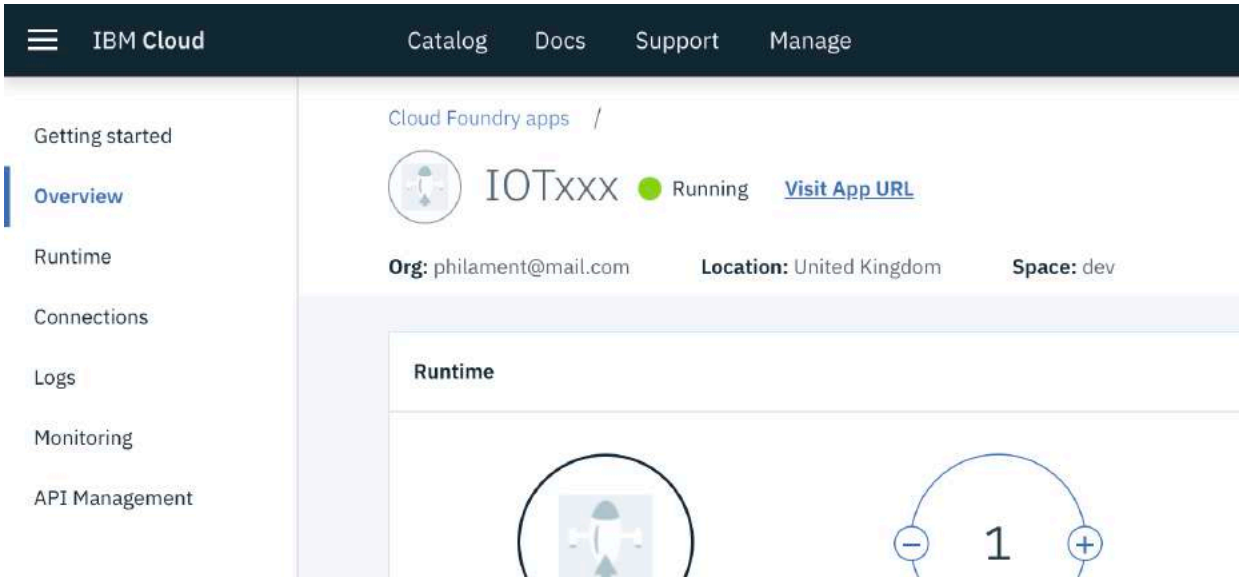
Credit : https://github.com/phthom/PracticalIBMCloud/

Section 1. Register your Device

Create and Register your device with IoT Platform service. Follow these steps to get access to this service.

Task 1. Go back to the Dashboard

Click on IBM Cloud on the top left part of the screen and Click on your application IoTxxx:



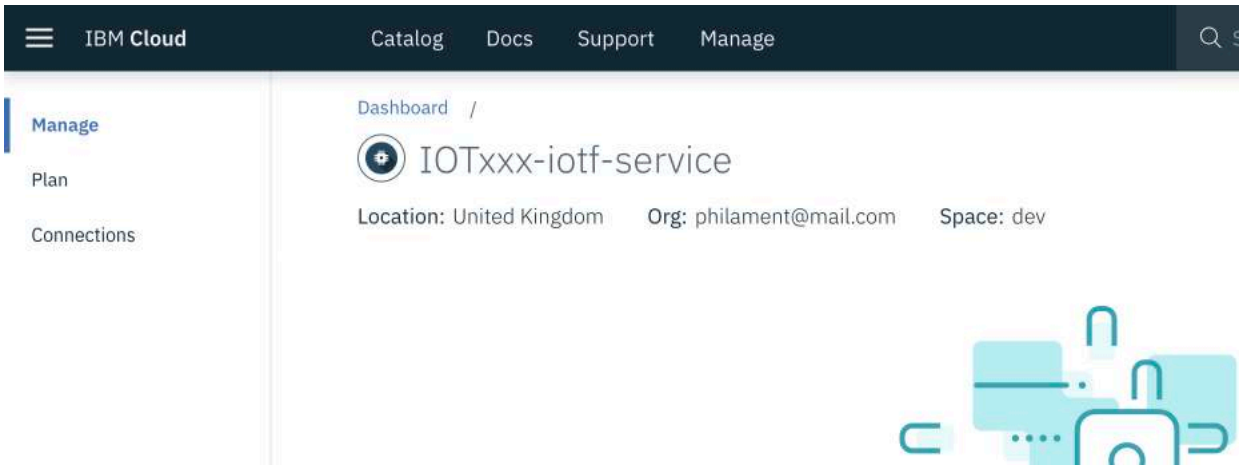
Task 2. IoT platform access

At the bottom of the screen, you should see 2 services (one is the **Cloudant** database and the other one is the **Internet of Things Platform service**). The IBM Internet of Things Platform service lets your apps communicate with and consume data collected by your connected devices, sensors, and gateways.

Our recipes make it super easy to get devices connected to our Internet of Things cloud.

Your apps can then use our real-time and REST APIs to communicate with your devices and consume the data you've set them up to collect.

Click on the IoT Platform service. This is launching the IoT platform console.

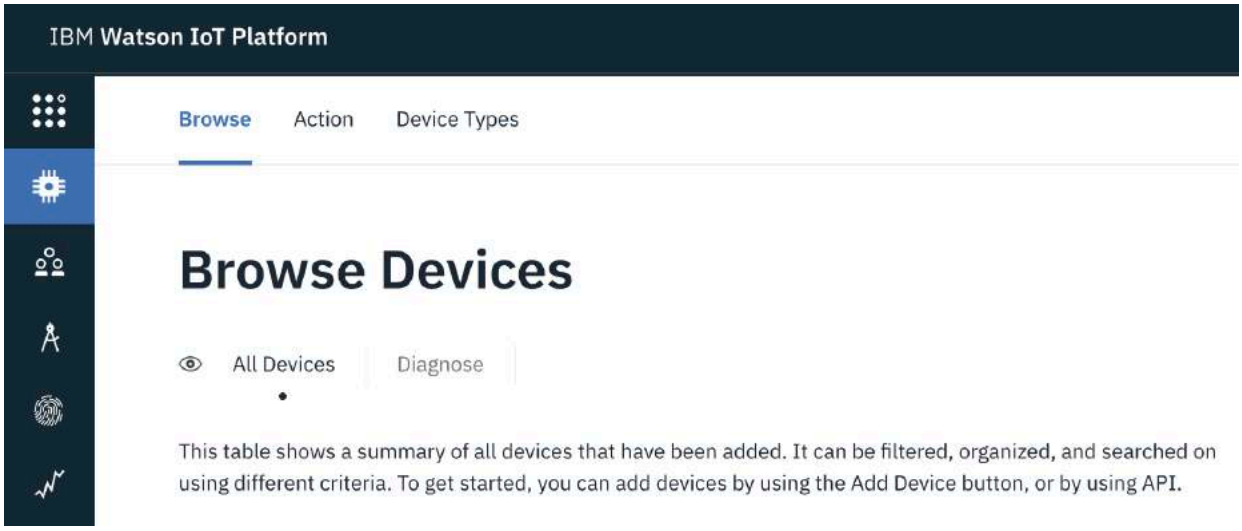


Task 3. LAUNCH IoT Dashboard

Click the LAUNCH green button. it can take a few minutes.

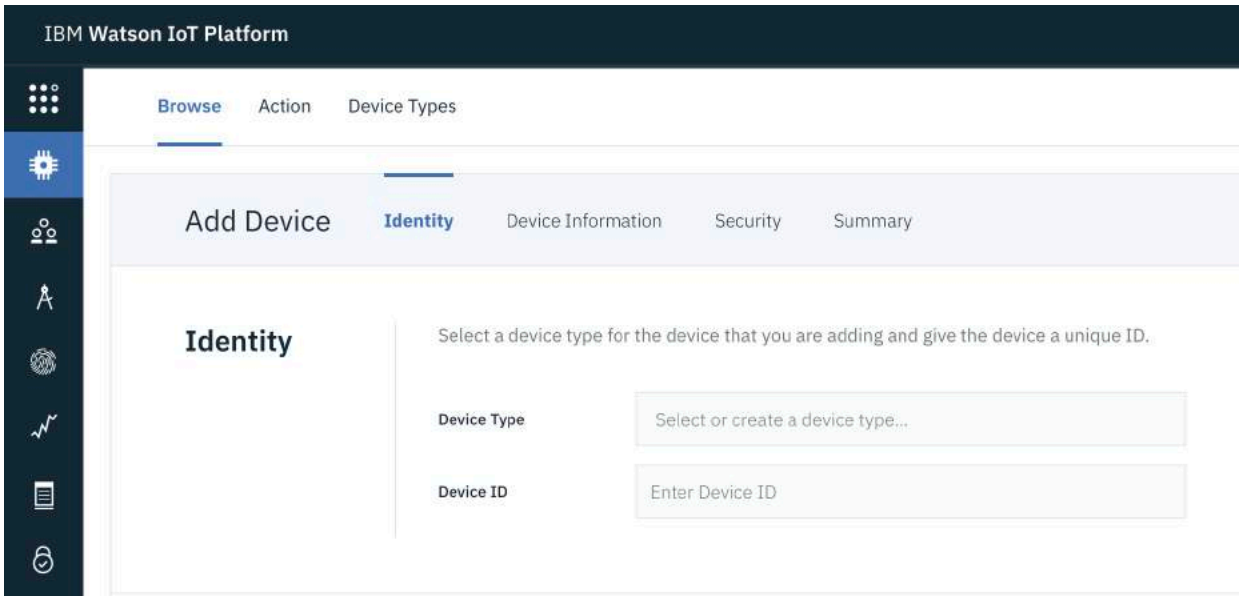
Task 4. Select Devices

On the left (the little circuit):



Task 5. Add a device

Click Add Device button

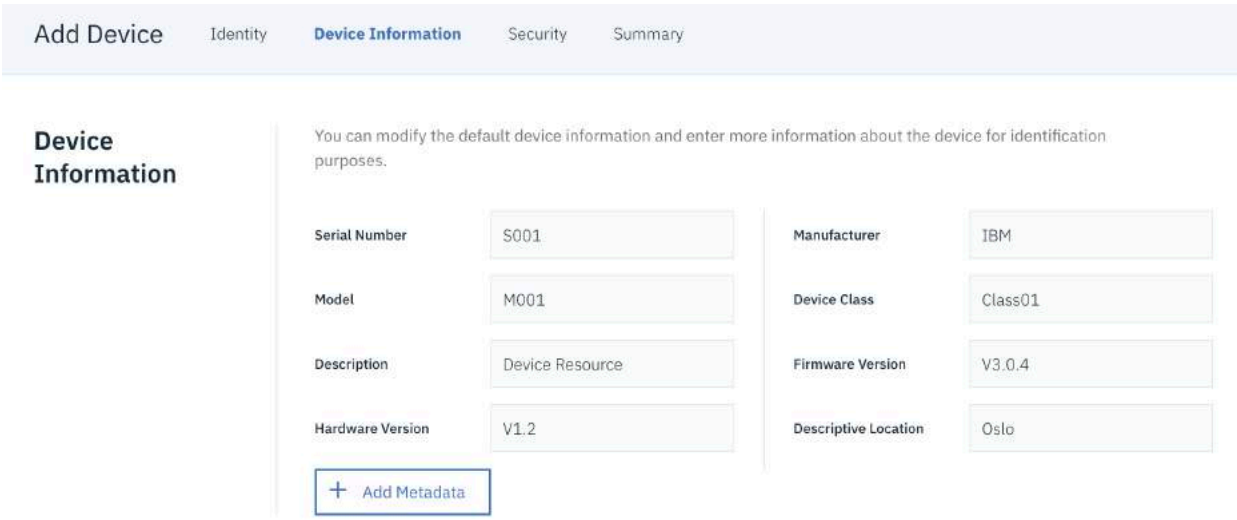


Type a name for the Device type : **DeviceType1** and a device ID : **100i700**

Click **Next**

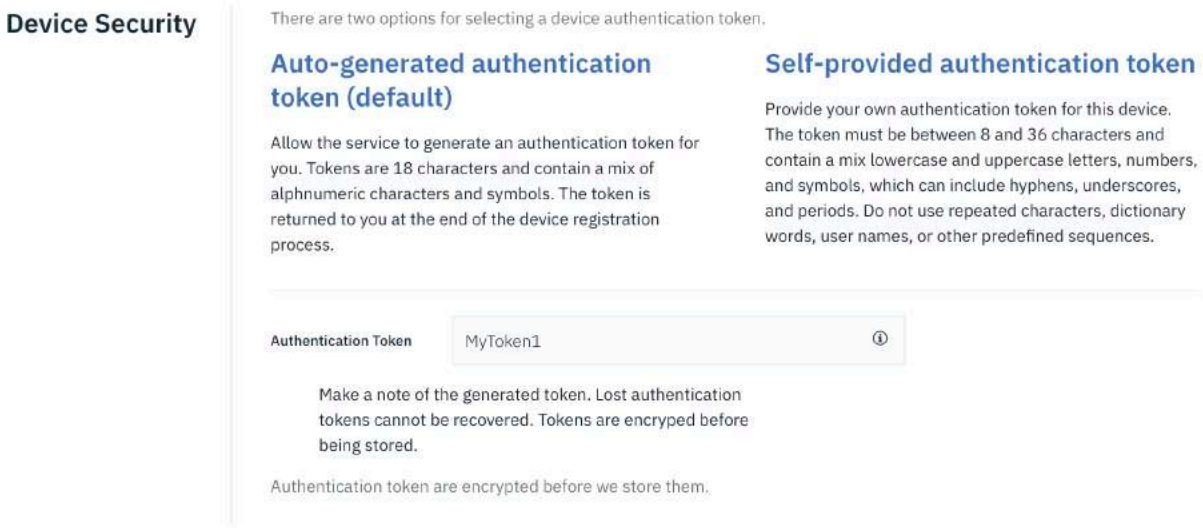
Task 6. Fill the device information

Enter Device information :

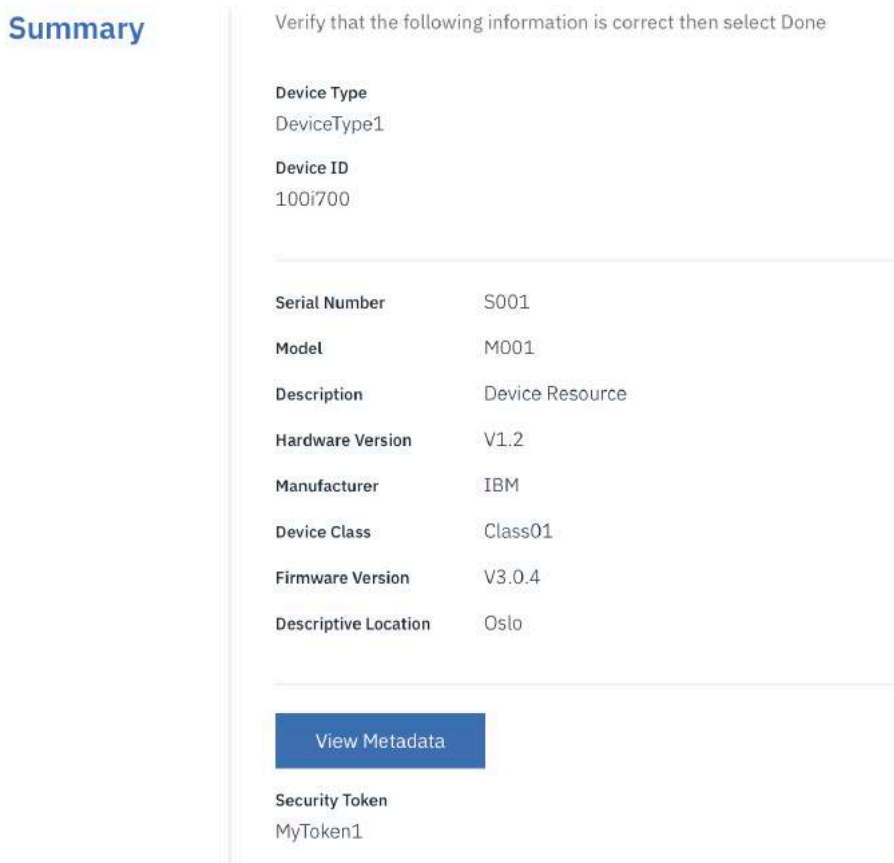


Task 7. Security - Provide a token

Type a token : **MyToken1** Click **Next** and **Add** to add a device.



Task 8. Summary



Review all the information and type **Done**

Device 100i700

Device Credentials

You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details.

Organization ID	djuoj2
Device Type	DeviceType1
Device ID	100i700
Authentication Method	use-token-auth
Authentication Token	MyToken1

Take a note of all the device security information.

Organization ID

```
1  Organization ID
2  djuoj2
3  Device Type
4  DeviceType1
5  Device ID
6  100i700
7  Authentication Method
8  use-token-auth
9  Authentication Token
10 MyToken1
```

Section 1. Simulate your device

We are going to simulate a device on your laptop by using a node application and connect this device to IoT Platform.

Note that every language provides MQTT Libraries for publishing and subscribing to MQTT events – C , C++, C# , Python, Node.js & javascript, Java. etc.

In this exercise, we will use a Node.js library “ibmiotf” for connecting to our IoT Broker (Watson IoT service).

We will also use a more generic and lower level library ‘mqtt.js” for publishing events.

Task 1. Device Simulator

On your machine, open a terminal window or a command line window, create a new directory for DeviceSimulator and get the git clone of the simulator app :

```
mkdir DeviceSimulator

cd DeviceSimulator
```

Task 2. Clone the simulator application:

```
git clone https://github.com/gmagie/device-simulator-for-ibm-iot

cd device-simulator-for-ibm-iot
```

Task 3. Configure your simulator application

Depending of your operating system, duplicate and rename a file called .env.example

MacOS or Linux `cp .env.example .env`

Windows `copy .env.example .env`

Edit the .env file with a text editor of your choice (nano, vi or notepad) and Replace all the values with the one you saved earlier.

Double check all the fields and Save the file.

Task 4. Review and Start the simulator

Review first app.js using your favorite editor (notepad, vscode, vi etc.)

```
vi app.js
```

Start node with the following command :

Note: Node.js has to be installed on your Desktop first.
Refer to https://nodejs.org/en/download/

```
npm install
```

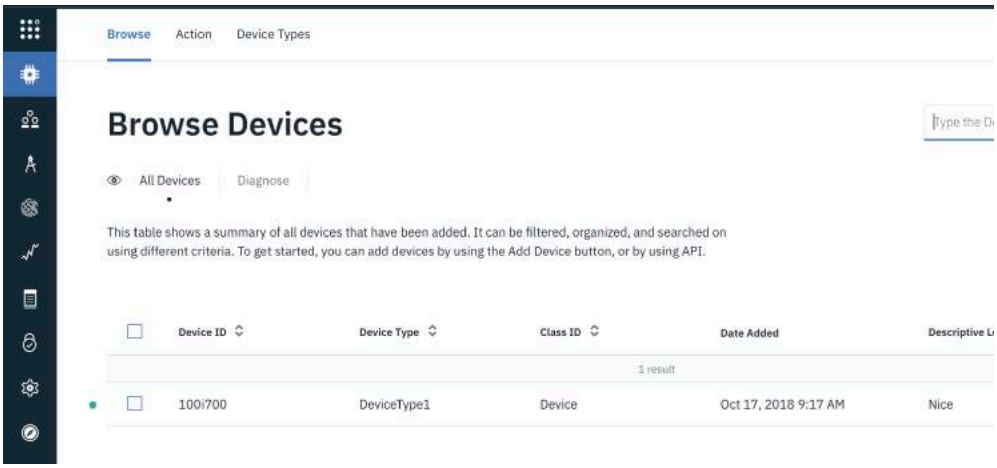
```
node app.js
```

```
--- DEBUG appENV: ---
AppEnv {
  isLocal: true,
  app: {},
  services: {},
  name: 'device-simulator-for-ibm-iot-7777',
  port: 6001,
  bind: 'localhost',
  urls: [ 'http://localhost:6001' ],
  url: 'http://localhost:6001' }


--- DEBUG iotConfig: ---
{ org: 'djuoj2',
  id: '100i700',
  'auth-token': 'MyToken1',
  type: 'DeviceType1',
  'auth-method': 'token' }

Server started on http://localhost:6001
Device simulator is connected to the IoT Foundation service
QoS level set to: 0
{"d":{"temperature":0,"pressure":50,"humidity":10,"luminosity":5},"ts":"2018-10-17T07:44:10.718Z"}
{"d":{"temperature":20,"pressure":52,"humidity":12,"luminosity":5},"ts":"2018-10-17T07:44:12.724Z"}
{"d":{"temperature":40,"pressure":54,"humidity":14,"luminosity":5},"ts":"2018-10-17T07:44:14.725Z"}
{"d":{"temperature":60,"pressure":56,"humidity":16,"luminosity":5},"ts":"2018-10-17T07:44:16.729Z"}
{"d":{"temperature":80,"pressure":58,"humidity":18,"luminosity":5},"ts":"2018-10-17T07:44:18.731Z"}
```

At this point you should see this running application (simulating a device) is connected to the IoT platform service and some data (temperature, pressure and humidity) is being sent to the platform.

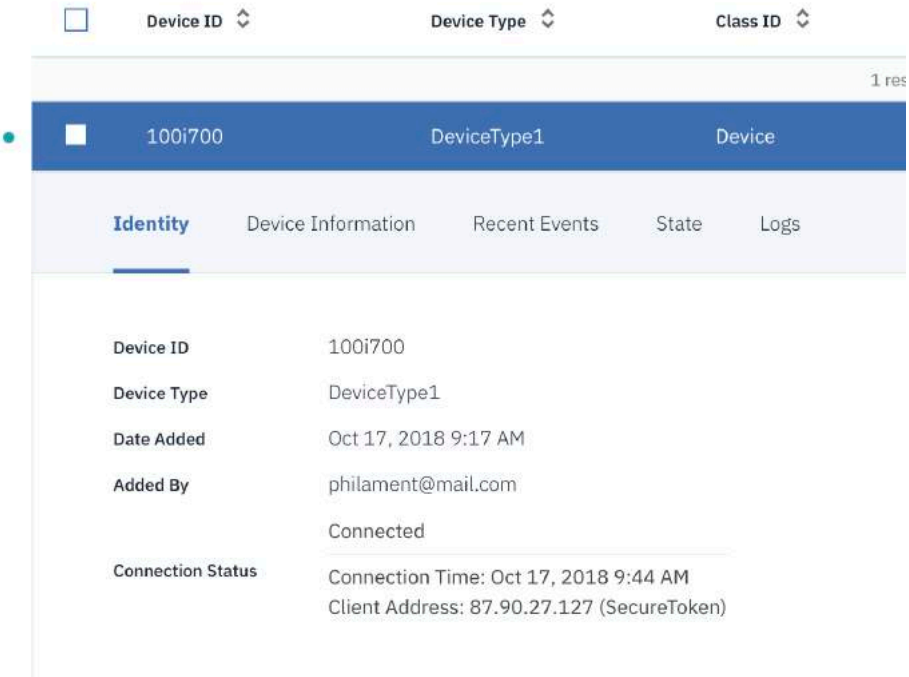
Task 5. Check the received messages on your IoT Broker

At this point, go back the IoT platform dashboard:

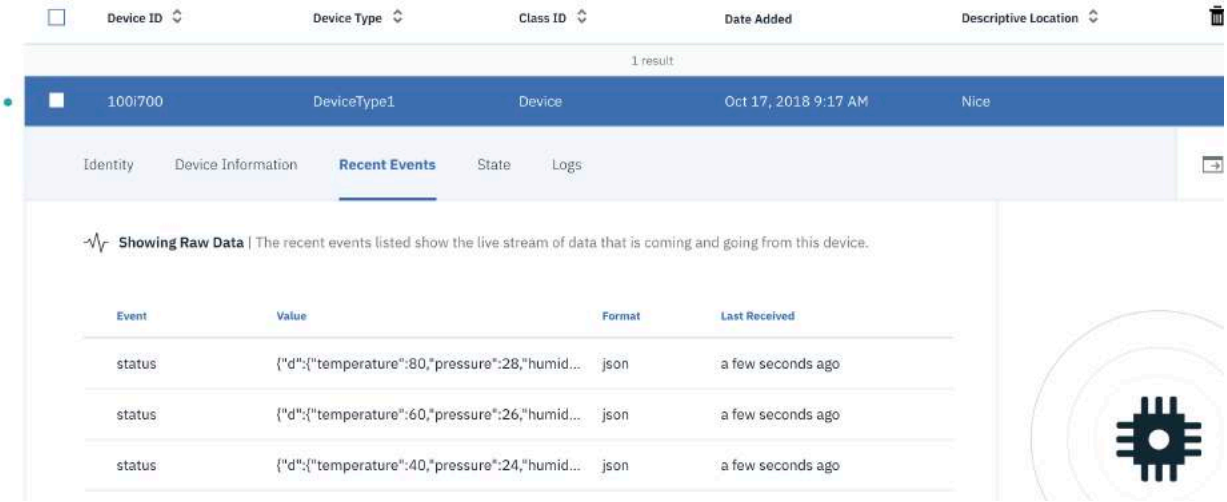


Our device should be connected and you should see some date coming.

Click on the device and scroll the page to the sensor information:



Click on the **recent events:** to see all the events coming from the simulated device (our application) :



Section 2. Create a MQTT Device on a micro-controller

In the previous section, we used the 'ibmiotf' library for subscribing events to a MQTT Broker. Let's use a generic library 'mqtt.js' , that allows us to directly use the MQTT Broker and the MQTT protocol with our Watson IoT Broker – an alternative to the ibmiotf library.

Task 1. Install MQTT Libraries on your Micro-controller *OPTIONAL

On a Micro-controller (ESP8266 Adafruit Huzzah, etc), we can use a similar library like explained below:

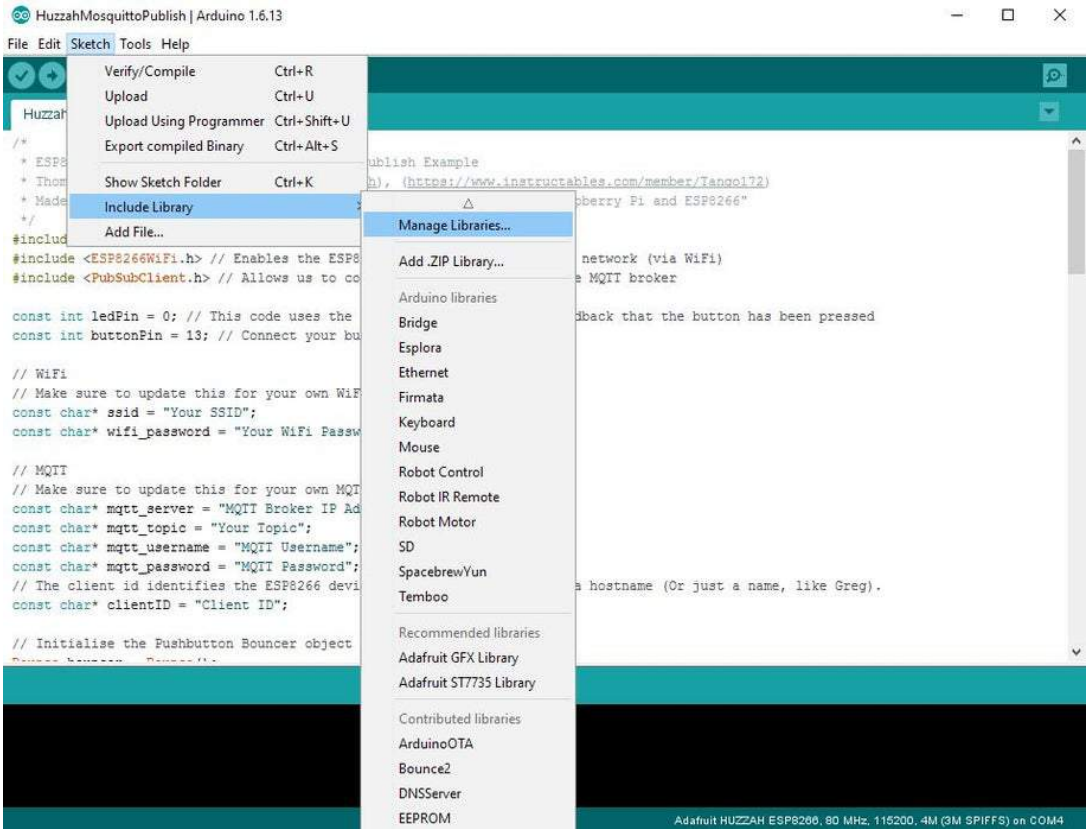


Fig. Soure & complete Lab using a ESP8266:
https://www.instructables.com/id/How-to-Use-MQTT-With-the-Raspberry-Pi-and-ESP8266/

Task 2. Install mqtt.js in Node.js - Physical device Simulation

From your current directory:

```
npm install mqtt -g
```

The previous command install mqtt node.js packages, and the mqtt command line which allows you to use mqtt.js without coding in Node.js

Task 3. Use mqtt.js using mqtt CLI

Use your **OrgID** , **Device Type**, **Device ID**, **Token** used in the previous section:

```
#
# Note that you need to rep
# parameters that you regis
# registration can either b
# via its http API
#

iotf_org=djuoj2
iotf_type=DeviceType1
iotf_id=100i700
iotf_authtoken=MyToken
```

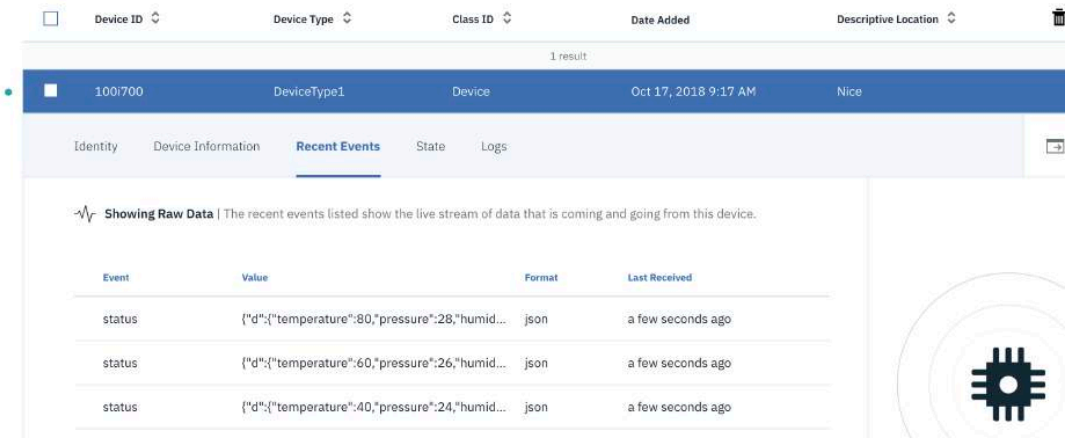
Replace the variables by your own values from your .env file. Below a sample script (bash for Linux or Mac) . On Windows, remove the backslashes and use a single line command with your variable set:

```
. .env

mqtt pub -h ${iotf_org}.messaging.internetofthings.ibmcloud.com  \
-p '8883'  \
-i d:${iotf_org}:${iotf_type}:${iotf_id}  \
-C mqtts -u use-token-auth \
-P ${iotf_authtoken} \
--qos 0  \
-t 'iot-2/evt/update/fmt/json' \
-m '{"d":{"temp":28,"humidity":79,"objectTemp":24}}'
```

Note 1 : You can see all the MQTT protocol options by using mqtt –help
-i : DeviceID
-C mqtt | mqtts
-t : topic
-P: password or token

Note 2: the Watson IoT Broker respects naming conventions and format , documented here https://www.ibm.com/support/knowledgecenter/SSQP8H/iot/platform/devices/mqtt.html

Check the received messages on your IoT Broker, like in the previous section:



Section 3. Data Visualization from MQTT Subscribers Apps

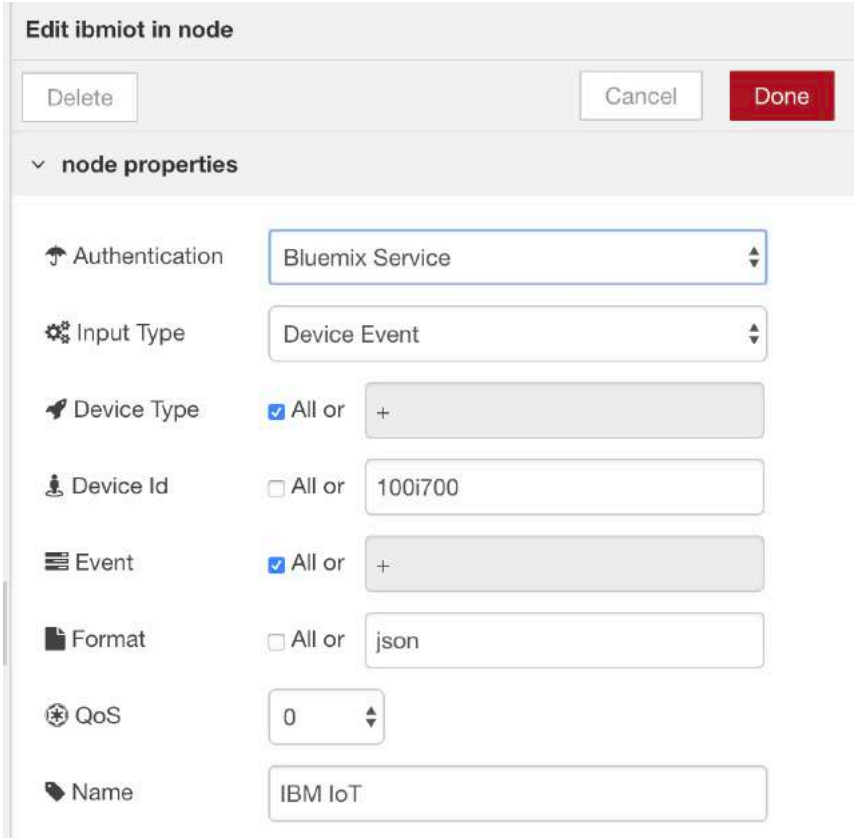
Validate the device connection and visualize the data in Node-RED.

Task 1. Suscription from Node-Red

Then go back to the **Node-RED flow editor**. With your mouse, select all the flows and boxes in the Flow 1 and **clear the Flow 1**. The screen should looks like that after the clear:

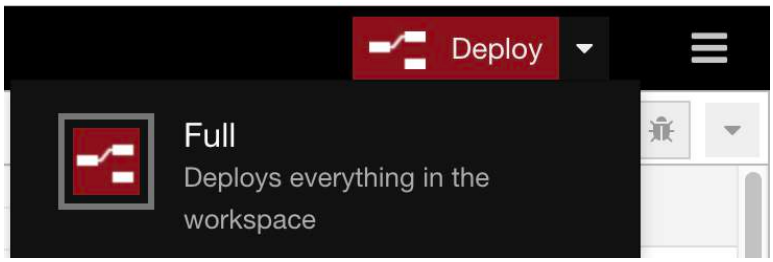
Drag the **ibmiot input node** from the left section and drop it on to the center app development environment. **Drag the debug output node** from the left section and drop it on to the center app development environment to the right of the ibmiot node. **Connect the 2 nodes**. It should look like this:

Double click the IBM IoT node. Update the Authentication section to **Bluemix Service** and update the Device id section to the DeviceID of your simulated device i.e. **100i700**



Click **Done**

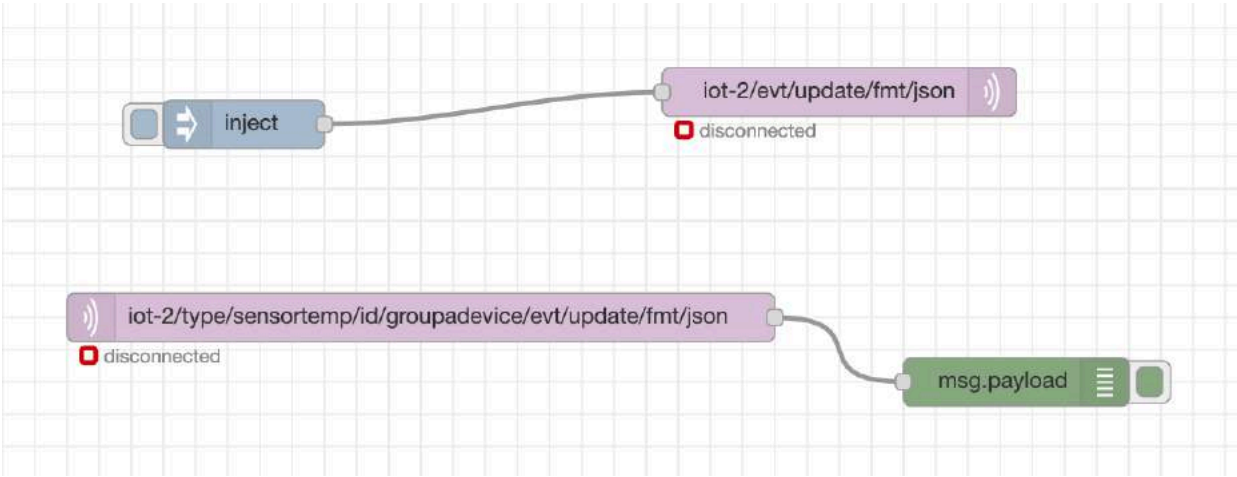
Click **Deploy** on the top tight section (Note, anytime there is a change in the app, the Deploy button turns red and it needs to be deployed for the change to take effect). The green icon and connected should appear below the IBM IoT flow.

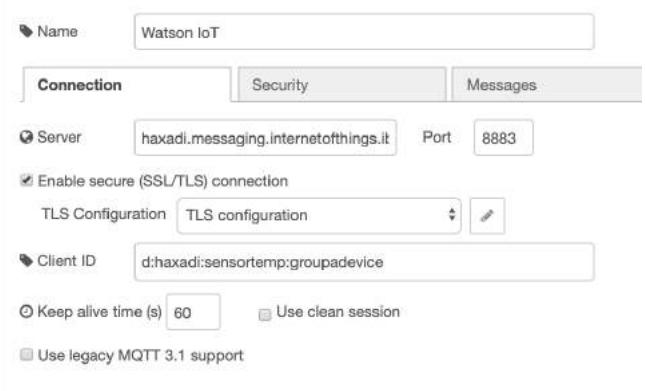


Go to the **debug tab **on the right section of the node-red and you will see the MQTT data being sent from the device to the node-red. If you click on the cursor in front of the object, you will see the temperature, pressure, humidity …

You can enhance this flow with a lot of features (logic, database …) and play with Node-RED.

An alternative way is to use the MQTT Node as shown below:





Successful exercise !

You learned how to use the IBM Internet of Things, Node-Red and a Node application to implement MQTT messages to send and collect metrics.

_____ END OF LAB _____