



## { Lab 3 }

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



March 2020 – Version 2.3

Authors: B. Marolleau, C. Bacle, C. Lalevée  
March 2020 - v2.3

© Copyright IBM Corp. 2020  
Materials may not be reproduced in whole or in part without the prior written permission of IBM

---

© Copyright IBM Corp. 2018  
Materials may not be reproduced in whole or in part  
without the prior written permission of IBM.

# Agenda

<b>Before Starting</b>	<b>4</b>
<b>1. Hands-on presentation</b>	<b>5</b>
Section 1. Overview	5
Section 2. Prerequisites	7
<b>2. Create your Node-RED application</b>	<b>8</b>
<b>3. Create sensor and a new flow</b>	<b>15</b>
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
<b>4. Create a dashboard application in Node-RED (optional #1)</b>	<b>23</b>
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	35
<b>5. Create a dashboard in Watson IoT Platform (optional #2)</b>	<b>38</b>
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

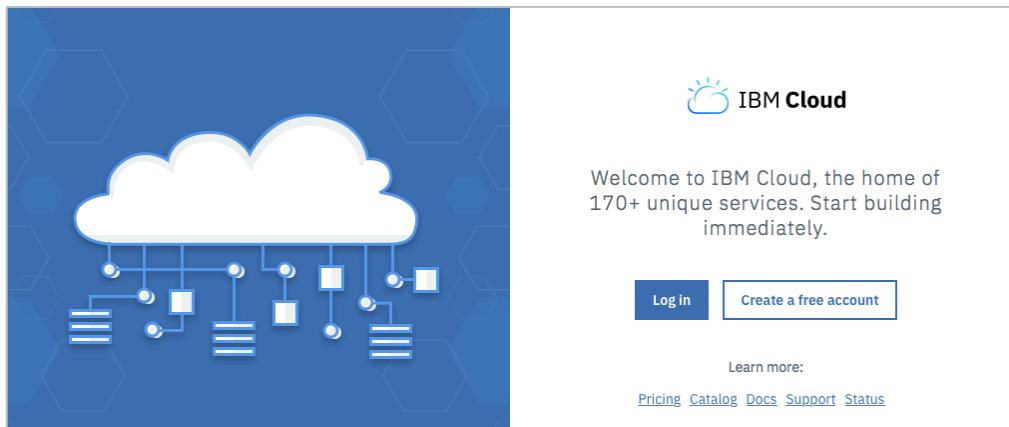
## Before Starting



### 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.

Dashboard

RESOURCE GROUP CLOUD FOUNDRY ORG CLOUD FOUNDRY SPACE LOCATION CATEGORY

All Resources All Organizations All Spaces US South All Categories

- 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.

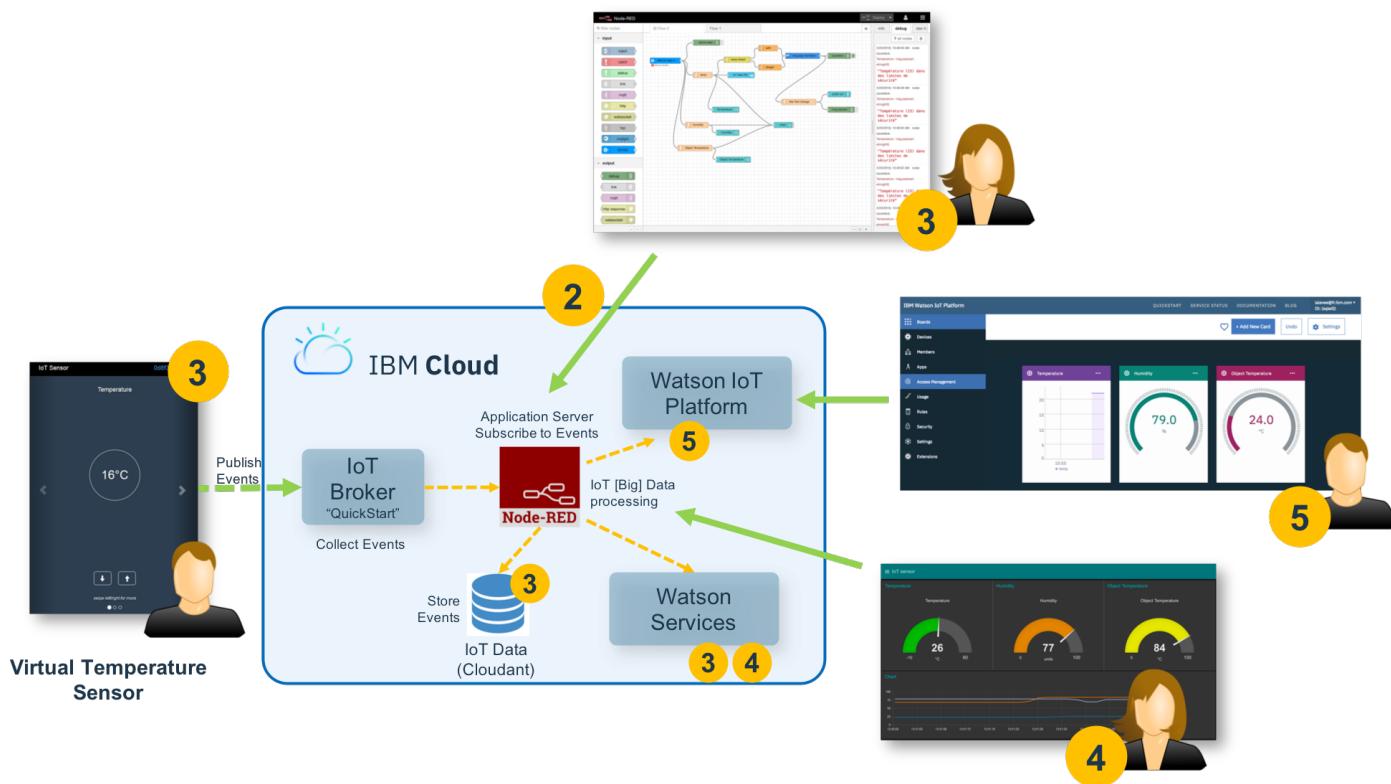
Node-RED (<https://nodered.org/>) is a flow-based programming tool, originally developed by the IBM Emerging Technology Services team (in early 2013) and now a part of JS Foundation. Traditional development can be very technical, but Node-RED enables you to concentrate on the logic of your workflow and 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:

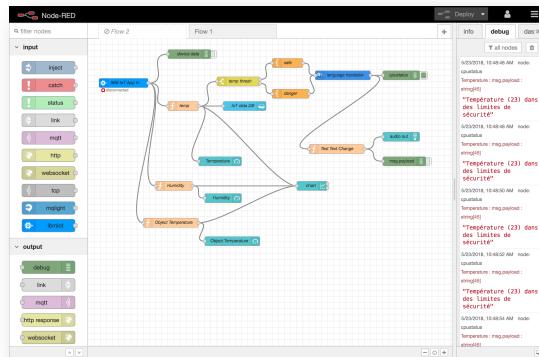
- Create & modify an application using Node-RED in IBM Cloud
- Discover new services & Node-RED to consume or create services (IoT / database...)
- Discover Watson services
- Discover Watson IoT Platform

Find below lab overview (with exercise numbers).



Expected results are :

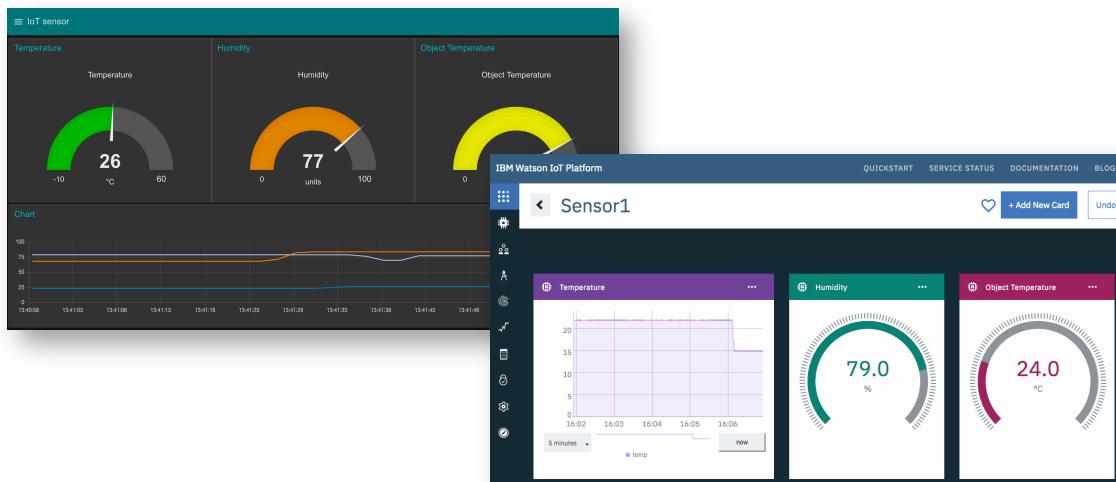
- Your Node-RED application is operational (using Node.js runtime), accessing Cloudant & IoT platform (QuickStart)



- 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_Flow.v2.1.json)
- [http://ibmcloud-watson-day.mybluemix.net/files/Lab3\\_IoT\\_Dashboard.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**

The screenshot shows the IBM Cloud Catalog interface. The search bar at the top contains "node". The "Software" category is selected, indicated by a blue box around the "Software (?)" tab. Below the search bar, there are filters for "Services (6)" and "Software (7)". The main content area is titled "Software" with a sub-section "All Categories". It lists several software solutions: Metrics Server, Natural Language Understanding Node.js App, Node Exporter, Node-RED App, Node.js Express App, and Visual Recognition Node.js App. Each listing includes a brief description and a "Starter kits" link. The "Node-RED App" listing is highlighted with a blue box.

The screenshot shows the "Create app" form for a Node-RED application. The "App details" section includes fields for "App name" (set to "Node RED SSLPB"), "Resource group" (set to "default"), and "Tags" (set to "env:dev, version-1"). The "Service details" section includes fields for "Region" (set to "Dallas"), "Resource group" (set to "default"), and "Pricing plan" (set to "Lite"). A red number "1" points to the "App name" field. A red number "2" points to the "Region" dropdown. A red number "3" points to the "Pricing plan" dropdown. A red number "4" points to the "Create" button. To the right of the form, there is a sidebar with a "View source code" link and a message about setting up Continuous Delivery for the app.

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.

The screenshot shows the IBM Cloud interface for an application named "Node RED SSLPB". The top navigation bar includes links for Catalog, Docs, Support, Manage, and Account information. Below the navigation is a breadcrumb trail: Web Apps / Apps / App details. The main content area displays the "App details" tab, which lists a single service named "Cloudant" with a "Documentation" link. To the right of the main content, there's a sidebar titled "Knowledge Guide" and a "Next steps" section. The "Next steps" section contains three numbered steps: 1. Click Deploy your app, 2. Select a deployment target, 3. Check the deployment status. A red arrow points to the "Deploy your app" button in the sidebar, which is labeled with the number 1. A vertical "ASK A QUESTION" button is also visible on the right side of the sidebar.

- 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.

**Deploy your app**

Select your deployment target and configure your DevOps toolchain. After you click **Create**, the toolchain is created, and the deployment process is started automatically.

**Deployment target**

**Cloud Foundry**  
Deploy your app without managing underlying infrastructure.

**IBM Cloud API key**  
IBM Cloud API key  
The value is required.

**New +**

**Number of instances**  
1

**Memory allocation per instance**  
64 MB 128 2000 MB

**Select region to deploy in** Dallas **Select an organization** nick\_oleary@uk.ibm.com **Select a space** dev

**Host** node-red-sslpb **Domain** mybluemix.net

**DevOps toolchain name**  
Accept the default name, or enter a value up to 63 characters.  
NodeREDSSLPB

Select the region that your toolchain is created in, and then select the resource group that provides access to your new toolchain.

**Region** Dallas **Resource group** default

**Create**

**Delivery Guide**

1 Selecting the deployment target

2 Select your deployment target, and then provide the configuration information.

- **IBM Kubernetes Service:** Select the region, cluster name, and deployment type. If you don't have an available cluster, you can create one and then continue. The Knative type is available only if Knative is installed on your cluster.
- **Red Hat OpenShift on IBM Cloud:** Select the region, cluster name, and deployment type. If you don't have an available cluster, you can create one and then continue. OpenShift is available only with a standard cluster, which requires you to have a billable account. [Learn more](#).
- **Cloud Foundry:** Select the number of instances, memory allocation, region, org, and space. Then select the domain and provide a host name. [Learn more about deploying your app](#).

**Configuring the DevOps toolchain**

The DevOps toolchain includes a Delivery Pipeline tool where you can check the deployment status, start builds, manage deployment, and view logs and history. Provide a name for your toolchain, and then select the region and resource group.

**ASK A QUESTION**

- 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.

The screenshot shows the 'Continuous Delivery' section of the IBM Cloud interface. At the top, there is a link to the GitHub repository: [https://us-south.git.cloud.ibm.com/nick\\_oleary/NodeREDSSLPB](https://us-south.git.cloud.ibm.com/nick_oleary/NodeREDSSLPB). To the right is a 'Remove from toolchain' button. Below this, the 'Toolchain' section displays the following details:

- Name:** NodeREDSSLPB
- Location:** Dallas
- Resource group:** default
- Tool integrations:** Icons for GitHub, Docker, and Jenkins.

Below the toolchain section is the 'Delivery Pipelines' section, which contains one pipeline entry:

- Name:** NodeREDSSLPB
- Status:** In progress (indicated by a red arrow pointing to a red box labeled '1').
- Last input:** Last commit by IBM Cloud (19 seconds ago)
- Clone from zip:** A blue link.

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

The screenshot shows the IBM Cloud sidebar. A red box labeled '1' points to the 'IBM Cloud' logo. A red box labeled '2' points to the 'Resource List' option in the sidebar menu.

The screenshot shows the 'Resource list' page. The search bar at the top contains the text 'node'. The table below lists resources, with two entries under the 'Cloud Foundry apps' section:

Name	Group	Location	Offering	Status
Node RED MEE	Benoit Marolleau / dev	Dallas	SDK for Node.js™	Started
Nodered-epf-1	Benoit Marolleau / dev	Dallas	SDK for Node.js™	Started

- 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.

Resource list

Create resource

Collapse all | Expand all

Name	Group	Location	Status	Tags
<input type="text"/> Filter by name or IP address...	<input type="text"/> Filter by group or org...	<input type="text"/> Filter...	<input type="text"/> Filter...	<input type="text"/> Filter...
> Devices (0)				
> VPC infrastructure (0)				
> Clusters (0)				
< Cloud Foundry apps (1)				
Node RED SSLPB	nick_oleary@uk.ibm.com / dev	Dallas	● Stopped	—
> Cloud Foundry services (8)				
> Services (15)				
> Storage (0)				
> Network (0)				
> Cloud Foundry enterprise environments (0)				
> Functions namespaces (0)				
< Apps (1)				
Node RED SSLPB	default	Global	—	—
> Developer tools (13)				

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

Resource list / App details

Node RED SSLPB

Visit App URL Add tags

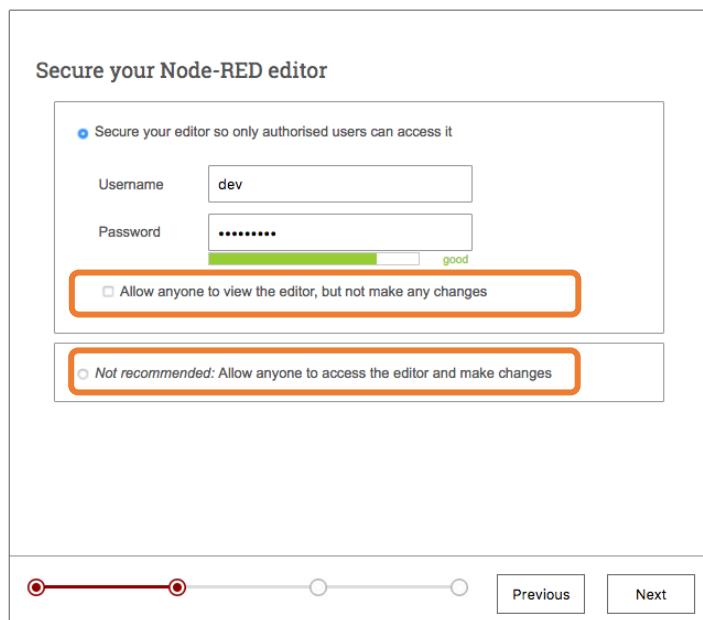
Resource group: default

View repo

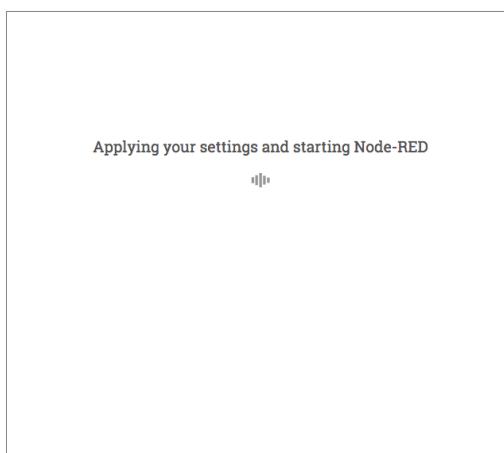
App details Developer resources

- 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**.

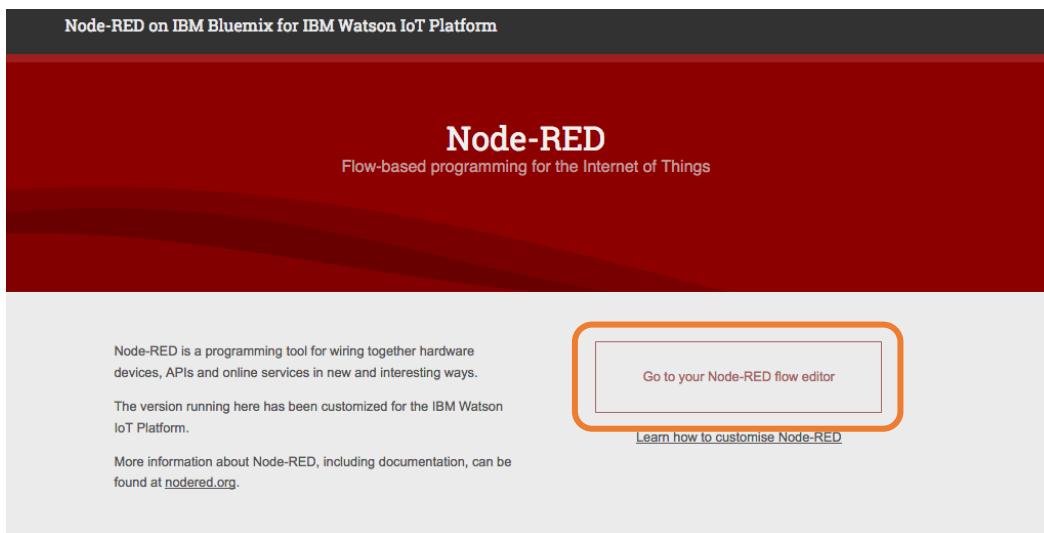


- 12. On wizard last step screen, click on **Finish** to start Node-RED.



- 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

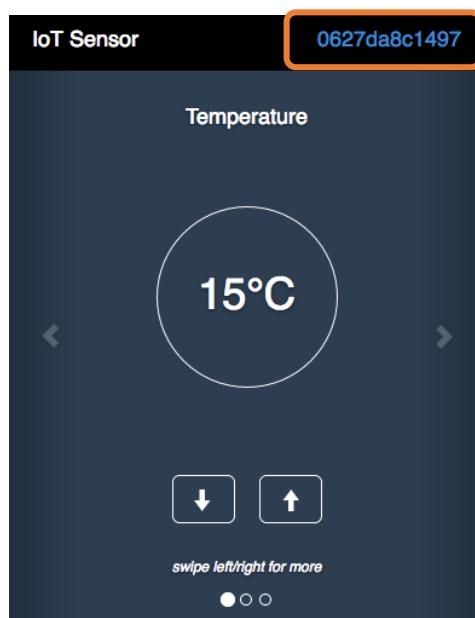
#### Section 1. Sensors & IoT

- 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.

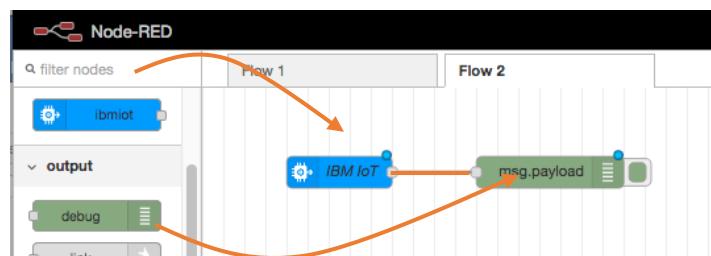


Note: Instead of using your desktop browser, you can use your smartphone.

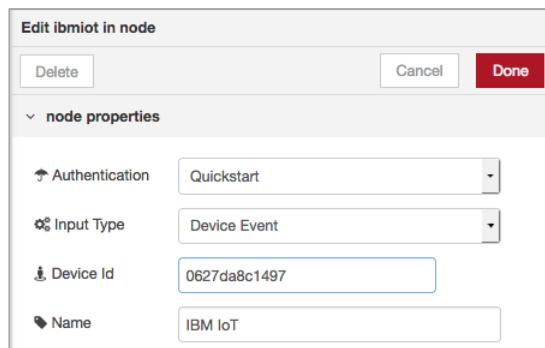
- 3. Identify your virtual device ID (top right corner) : copy it. You will use it in next section.  
Warning: if you reload this page, the device ID changes.

## 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



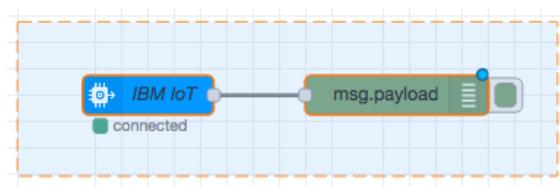
- 3. Configure IBM IoT by double clicking on it :
  - Authentication: Quickstart (means it is a simple authentication – for demo purposes)
  - Device ID : <The value from Section 1, step 3 - Generated by the Simulator>



- 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.

The Node-RED workspace displays a flow consisting of an 'IBM IoT' node and a 'msg.payload' output node. The 'IBM IoT' node has a green 'connected' status indicator. To the right, the 'info' tab of the Node-RED interface shows a log entry for a device event. The IoT Sensor application window shows a temperature reading of 15°C.

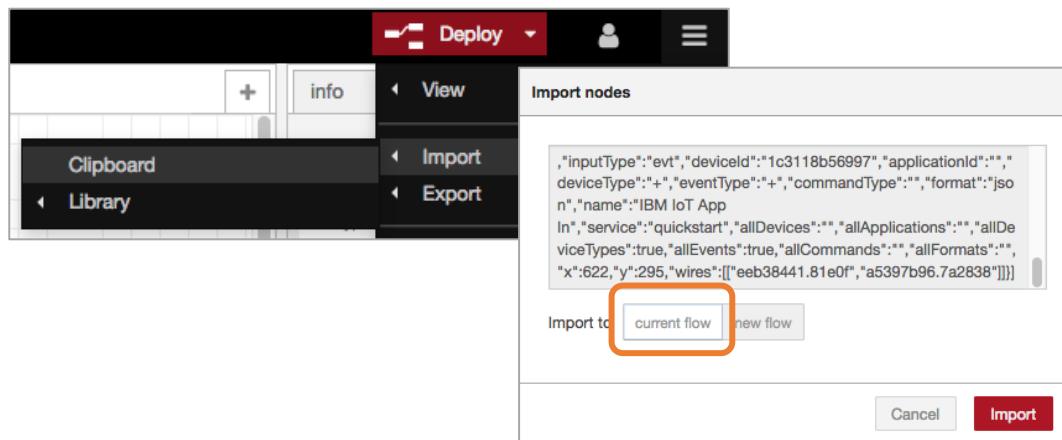
- \_\_\_ 6. Delete the whole flow by selecting all the nodes & pressing the 'Delete' key.



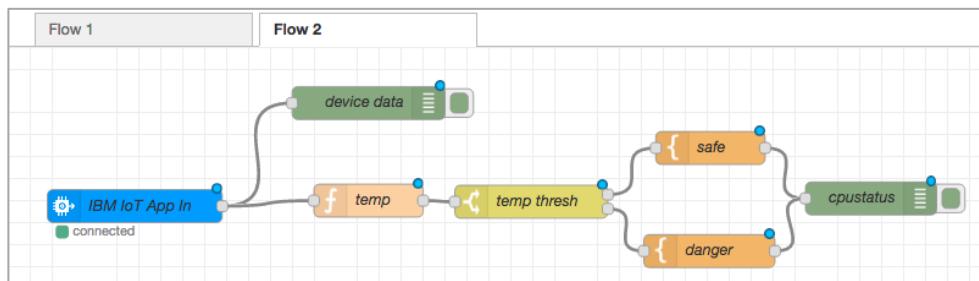
- \_\_\_ 7. Now import a new flow. A flow can be exported and imported using JSON file. Open link [http://ibmcloud-watson-day.mybluemix.net/files/Lab3\\_IoT\\_Flow.v2.1.json](http://ibmcloud-watson-day.mybluemix.net/files/Lab3_IoT_Flow.v2.1.json) to display code in JSON format. You can also open the file you downloaded previously.

```
[{"id": "a5397b96.7a2838", "type": "function", "z": "e85ad8f2.359ef8", "name": "temp", "func": "return{payload:msg.payload.d.temp};", "outputs": 1, "x": 836, "y": 290, "wires": [{"d84e493b.bf121"]}], {"id": "d84e493b.bf121", "type": "switch", "z": "e85ad8f2.359ef8", "name": "temp thresh", "property": "payload", "rules": [{"t": "lte", "v": "40"}, {"t": "gt", "v": "40"}], "checkall": "true", "outputs": 2, "x": 985, "y": 291, "wires": [{"e1391083.435be"}, {"91efb4b4.3257f8"}]}, {"id": "27a6cd52.56eefa", "type": "debug", "z": "e85ad8f2.359ef8", "name": "cpustatus", "active": true, "complete": "false", "x": 1296, "y": 288, "wires": []}, {"id": "eeb38441.81e0f", "type": "debug", "z": "e85ad8f2.359ef8", "name": "device data", "active": true, "complete": "false", "x": 836, "y": 201, "wires": []}, {"id": "e1391083.435be", "type": "template", "z": "e85ad8f2.359ef8", "name": "safe", "field": "payload", "fieldType": "msg", "syntax": "mustache", "template": "Temperature({{payload}}) within safe limits", "x": 1149, "y": 242, "wires": [{"27a6cd52.56eefa"}]}, {"id": "91efb4b4.3257f8", "type": "template", "z": "e85ad8f2.359ef8", "name": "danger", "field": "payload", "fieldType": "msg", "syntax": "mustache", "template": "Temperature {{(payload)}} critical", "x": 1148, "y": 336, "wires": [{"27a6cd52.56eefa"}]}, {"id": "ebeca570.9840c", "type": "ibmiot in", "z": "e85ad8f2.359ef8", "authentication": "quickstart", "apiKey": "", "inputType": "evt", "deviceId": "1c3118b56997", "applicationId": "", "deviceType": "+", "eventType": "+", "commandType": "", "format": "json", "name": "IBM IoT App"}, {"id": "a5397b96.7a2838", "type": "service", "z": "a5397b96.7a2838", "name": "quickstart", "allDevices": "", "allApplications": "", "allDeviceTypes": true, "allEvents": true, "allCommands": "", "allFormats": "", "x": 622, "y": 295, "wires": [{"eeb38441.81e0f"}, {"a5397b96.7a2838"}]}]
```

- \_\_\_ 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.

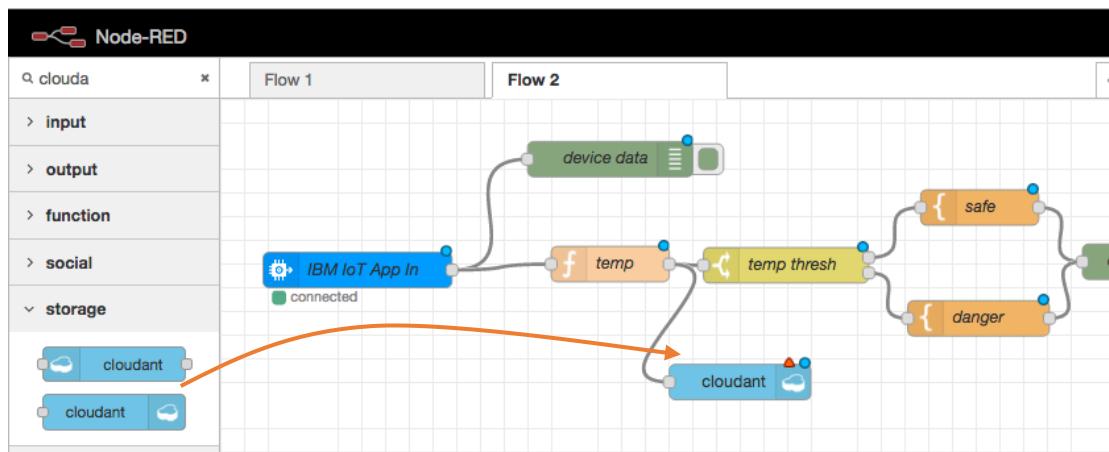
```
5/22/2018, 11:13:47 AM node: device data
iot-2/type/iotqs-sensor/id/1e8d6101b789/evt/iotsensor/fmt/json : msg.payload : Object
  ↪ { d: object }

5/22/2018, 11:13:47 AM node: cpustatus
msg.payload : string[34]
"Temperature(15) within safe limits"
```

### Section 3. Insert IoT Data in Cloudant DB

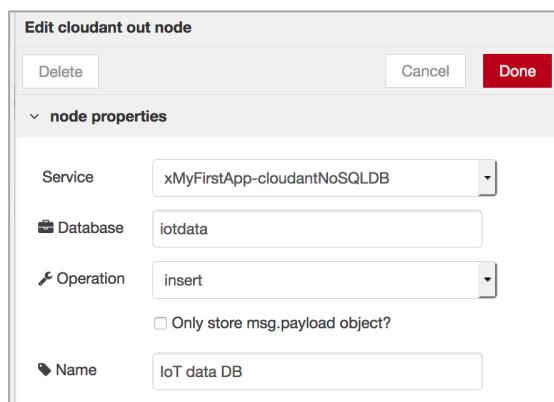
Let's insert the event data coming from the Device sensors in a Cloudant database! Remember that you already have a Cloudant service deployed for Node-RED. You will use it to store your data.

- 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

The screenshot shows the IBM Cloud Dashboard with the following filters: RESOURCE GROUP (All Resources), CLOUD FOUNDRY ORG (All Organizations), CLOUD FOUNDRY SPACE (demo), LOCATION (US South), and CATEGORY (All Categories). The 'Cloud Foundry Applications' section lists 'xMyFirstApp' with 'Region' US South. The 'Cloud Foundry Services' section lists three services: 'xMyFirstApp-cloudantNoSQLDB' (Region US South), 'xMyFirstApp-iotf-service' (Region US South), and another unnamed service (Region US South). The service 'xMyFirstApp-cloudantNoSQLDB' is highlighted with an orange box.

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

The screenshot shows the 'Data & Analytics / xMyFirstApp-cloudantNoSQLDB' service details page. It displays the location (US South), organization (lalevee@fr.ibm.com), and space (demo). Below this, it says 'Cloudant NoSQL DB'. On the right side, there is a 'LAUNCH' button, which is highlighted with an orange box.

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

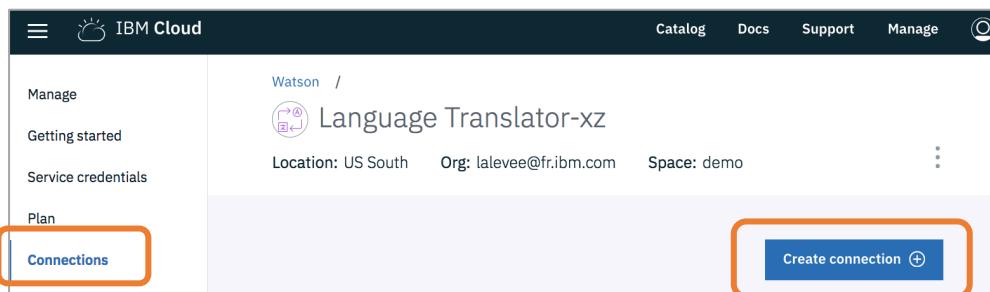
The screenshot shows the 'Databases' list page. The left sidebar has a 'Database' icon highlighted with an orange box. The main table lists databases: 'iotdata' (selected and highlighted with an orange box) and 'nodered'. Each row has an 'Actions' column with icons for edit, lock, and delete.

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

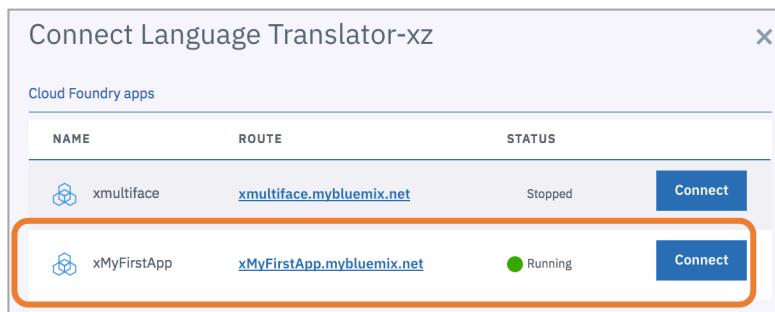
The screenshot shows the 'iotdata' database details page. The left sidebar has 'All Documents' selected. The main area shows a table of documents with columns: id, key, and value. One document is selected and highlighted with an orange box. The table includes a header row and several data rows.

## 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**.



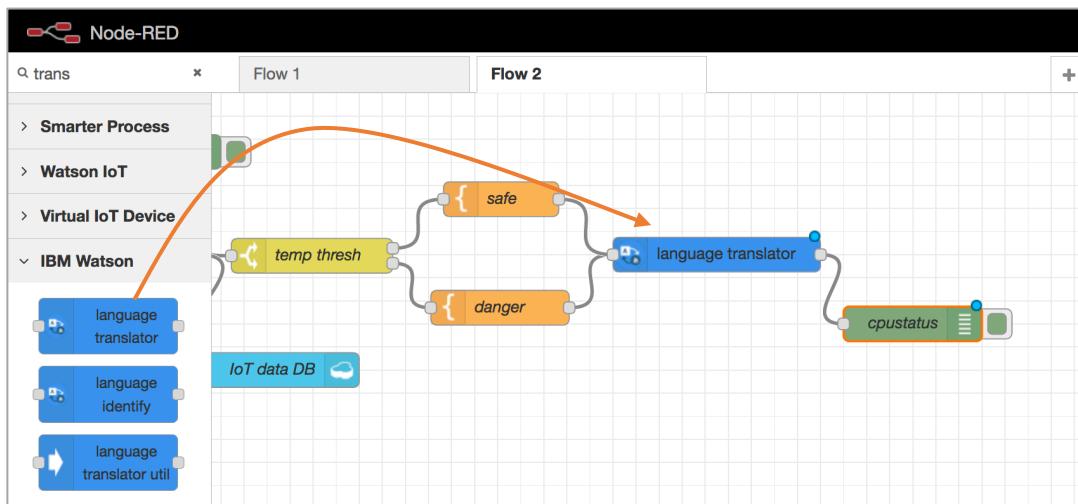
- 3. Connect service to your Node-RED application



- 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.

Lost connection to server, reconnecting in 48s. [Try now](#)

- 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).

info	debug
	<p>5/22/2018, 4:25:54 PM node: cpustatus    msg.payload : string[45]    "Température (17) dans les limites de sécurité"</p> <p>5/22/2018, 4:25:56 PM node: device data    iot-2/type/iotqs-sensor/id:229e50b82f91/evt/otsensor/fmt/json : msg.payload : Object    ↪ { d: object }</p> <p>5/22/2018, 4:25:56 PM node: cpustatus    msg.payload : string[45]    "Température (17) dans les limites de sécurité"</p> <p>5/22/2018, 4:25:58 PM node: device data</p>

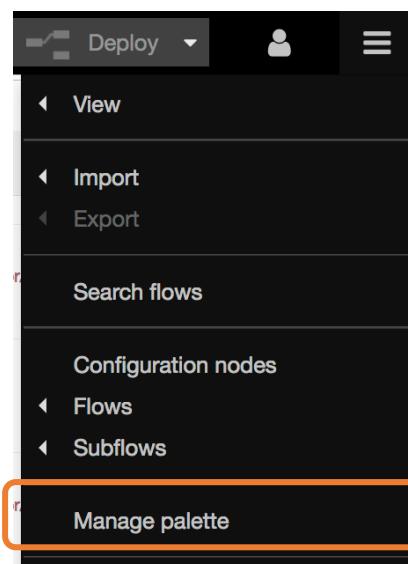
## 4. Create a dashboard application in Node-RED (optional #1)

### Section 1. Import Node-RED Dashboarding capability

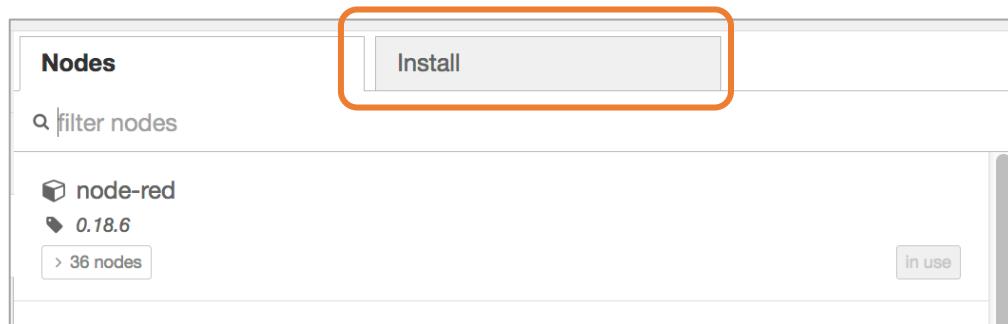
- 1. At the top right-hand side of the page, click the ‘burger’ menu:



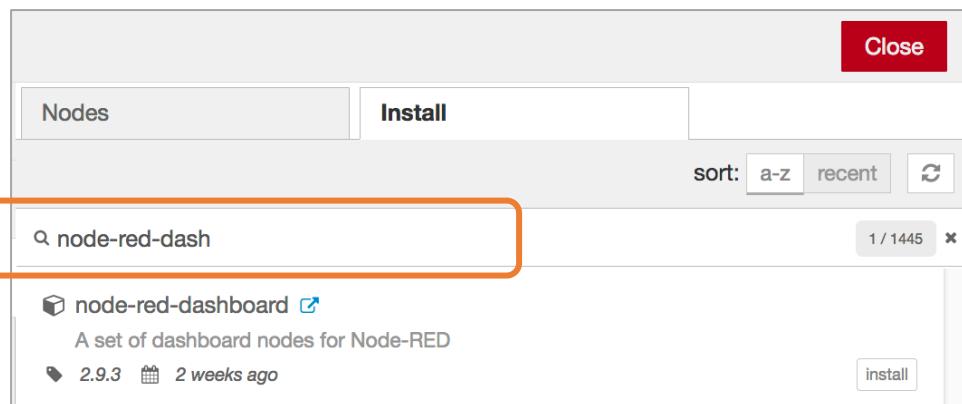
- 2. Click **Manage palette**:



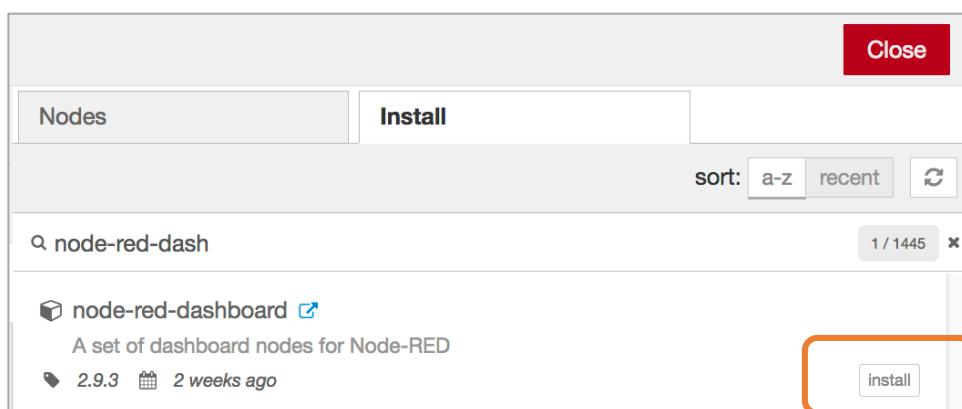
- 3. In the sidebar that appears on the left-hand side of the page, click the **Install** tab:



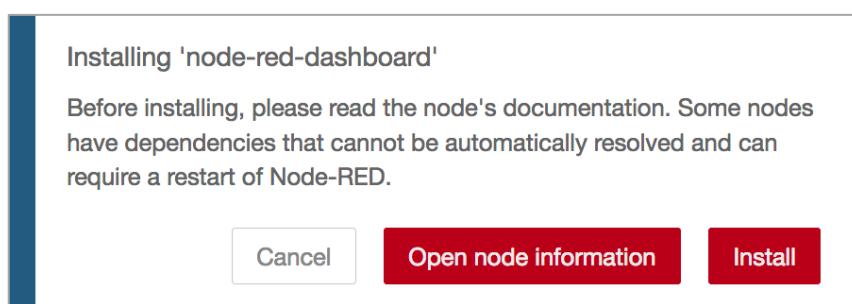
4. In the search field, type **node-red-dash**:



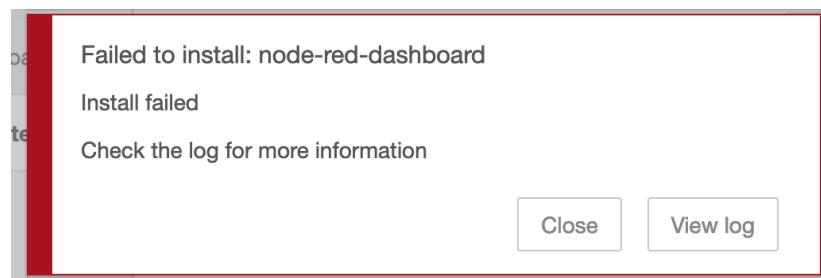
5. Next to the nod-red-dashboard result, click **install**



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.

Continuous Delivery

Remove from toolchain [-](#)

Toolchain

Name	NodeREDSSLPB
Location	Dallas
Resource group	default
Tool integrations	

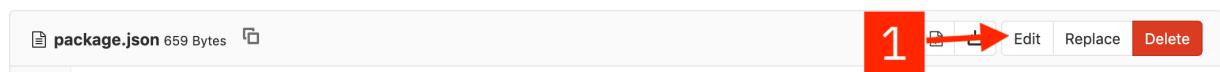
Delivery Pipelines

Name	NodeREDSSLPB
Status	In progress
Last input	Last commit by IBM Cloud (19 seconds ago) <a href="#">Clone from zip</a>

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

The screenshot shows the IBM Cloud interface for a project named "NodeREDSSLPB". The left sidebar includes options like Project, Details, Activity, Releases, Repository, Issues (0), Merge Requests (0), Wiki, Snippets, and Settings. The main area displays a list of files under the heading "CONTRIBUTING.md". The "package.json" file is visible in the list, with a red arrow pointing to it and a red box labeled "1" highlighting it. Other files listed include DCO1.1.txt, Dockerfile, Dockerfile-tools, LICENSE, README.md, bluemix-settings.js, cli-config.yml, cloudantStorage.js, index.js, manifest.yml, run-debug, run-dev, service.yaml, and README.md. Each file entry includes a "clone from zip" link and a timestamp indicating when it was last updated (3 weeks ago).

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)

Edit file

The screenshot shows the 'Edit file' interface in the IBM Client Center. At the top, there are tabs for 'Write' and 'Preview changes'. Below that, a code editor window titled 'package.json' is open, showing a JSON configuration file. A red arrow labeled '1' points to the 'dependencies' section of the code. The code editor has a 'Soft wrap' and 'text' dropdown at the top right.

Below the code editor is a commit dialog. It includes fields for 'Commit message' (containing 'Add node-red-dashboard') and 'Target Branch' (set to 'master'). A red arrow labeled '2' points to the 'Commit message' field. A green button labeled 'Commit changes' is at the bottom left, and a 'Cancel' button is at the bottom right.

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.

The screenshot shows the IBM Cloud Client Center interface. On the left, a sidebar lists navigation options: Getting started, Overview, Runtime, Connections, Logs, API Management, Autoscaling, and Monitoring. The main content area displays the details for the 'Nodered-epf-1' application.

**Application Details:**

- Icon: Node.js (js)
- Name: Nodered-epf-1
- Status: Running
- Org: Benoit Marolleau
- Location: Dallas
- Space: dev
- Add tags

**Runtime Metrics:**

Category	Value
BUILDPACK	SDK for Node.js™
INSTANCES	1
MB MEMORY PER INSTANCE	128
TOTAL MB ALLOCATION	511.687GB still available

**Connections:** 1 (Cloudant-dz-70451) | Create connection

**Runtime cost:** \$0.95 (Current charges for billing period) | \$0.00 (Estimated total for billing period Mar 1, 2020 - Mar 31, 2020). Current and estimated cost excludes connected services. View full usage details

**Activity feed:**

- Mar 10, 2020 9:37:09 AM | benoit.marolleau@fr.ibm.com: started Nodered-epf-1 app
- Mar 10, 2020 9:37:09 AM | benoit.marolleau@fr.ibm.com: stopped Nodered-epf-1 app
- Mar 10, 2020 9:34:56 AM | Nodered-epf-1: an instance of the app crashed: APP&#x2F;PROC&#x2F;WEB: Exited with s...
- Mar 3, 2020 9:36:02 AM | benoit.marolleau@fr.ibm.com: started Nodered-epf-1 app
- Mar 3, 2020 9:56:01 AM | benoit.marolleau@fr.ibm.com: stopped Nodered-epf-1 app

**Continuous delivery:** You enabled continuous delivery and have a toolchain. With your toolchain, you can automate builds, tests, deployments, and more. View Docs. View toolchain

**Continuous delivery section:**

### Continuous delivery

Continuous delivery is not enabled for this app. Enable continuous delivery to automate builds, tests, and deployments through the Delivery Pipeline, GitHub, and more.

**Enable**

[Resource list](#) / [Cloud Foundry App](#) /

## Continuous Delivery Toolchain

[Cancel](#)[Cr](#)

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)

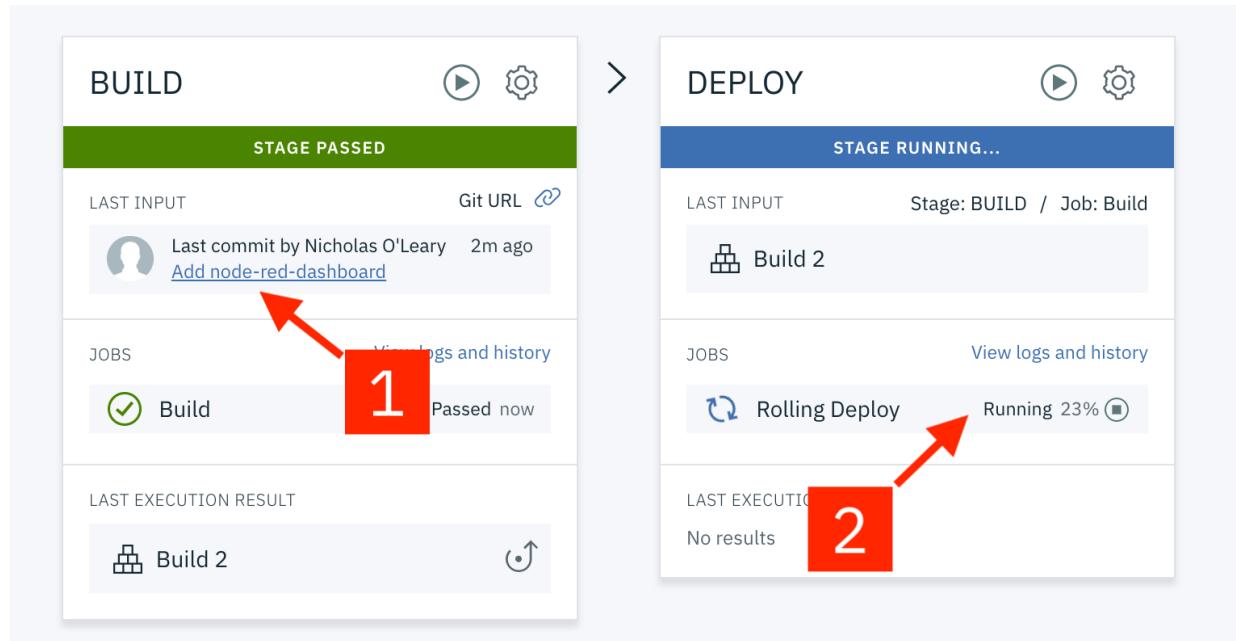
Repository type:

Existing

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

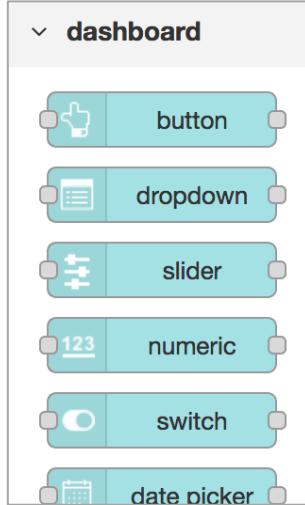
Repository URL:

<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.



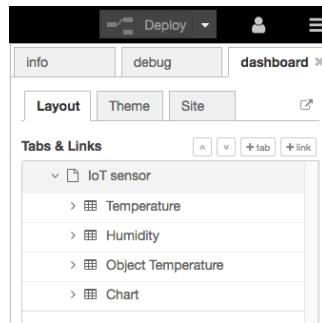
\_\_ 8. Note also that there is a new dashboard tab in the right-hand sidebar:



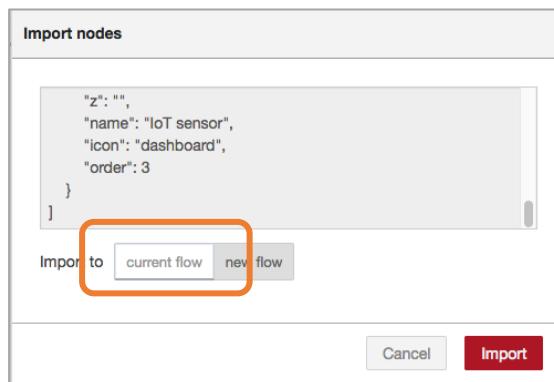
**TIP:** This dashboard tab may be used to add new tabs, menus etc. to the visualization dashboard. There are also two available themes – light and dark.

## Section 2. Create a simple Node-RED Dashboard

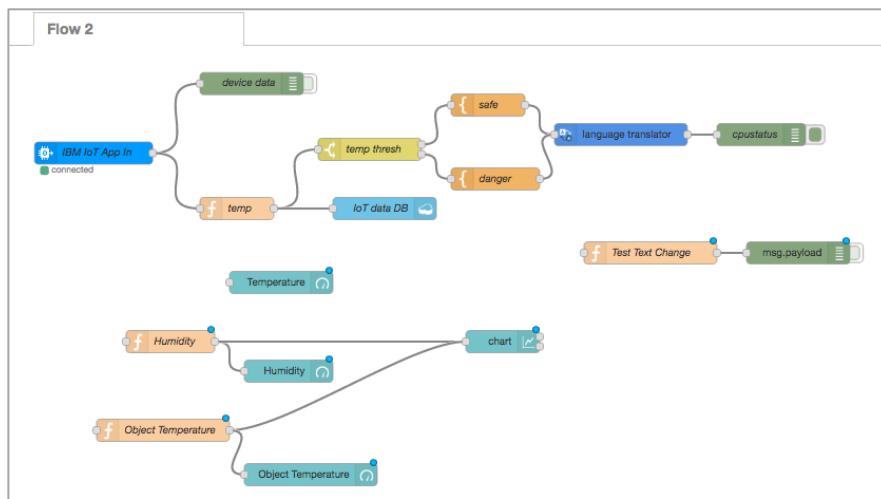
In this section, you will create a simple dashboard for sensor data using new dashboard nodes installed.



- 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**.

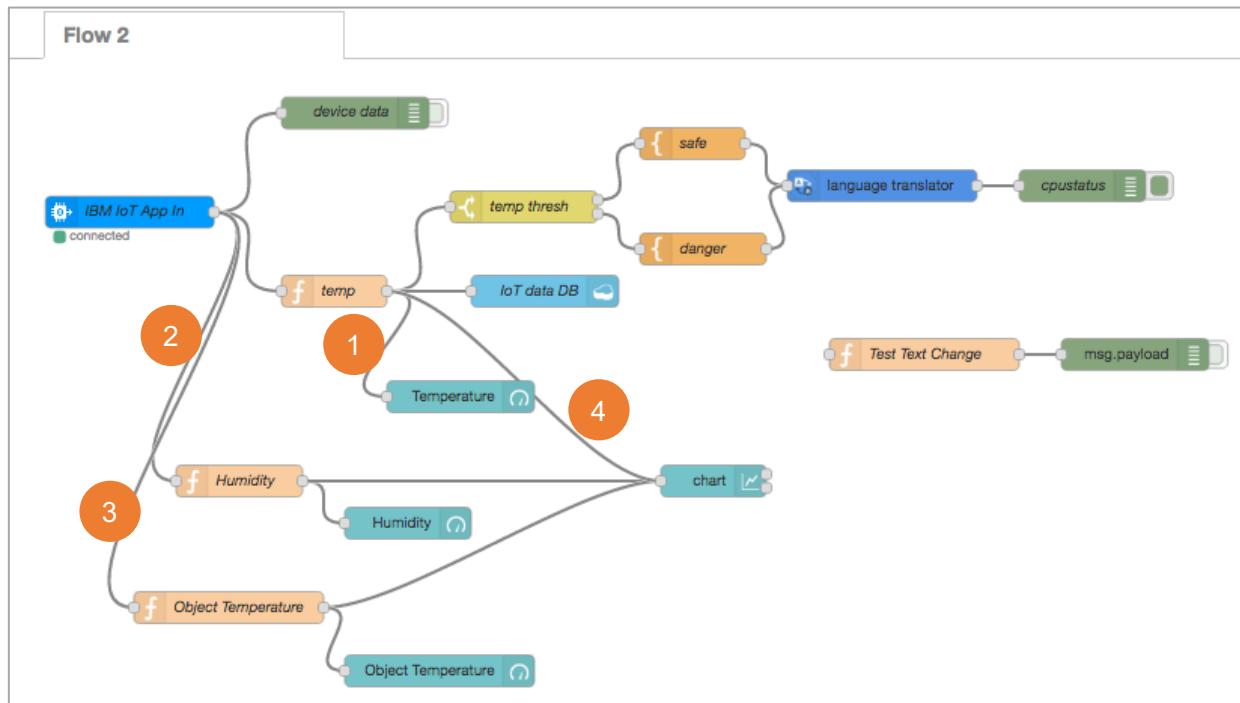


- 2. Click on workspace to paste new nodes.

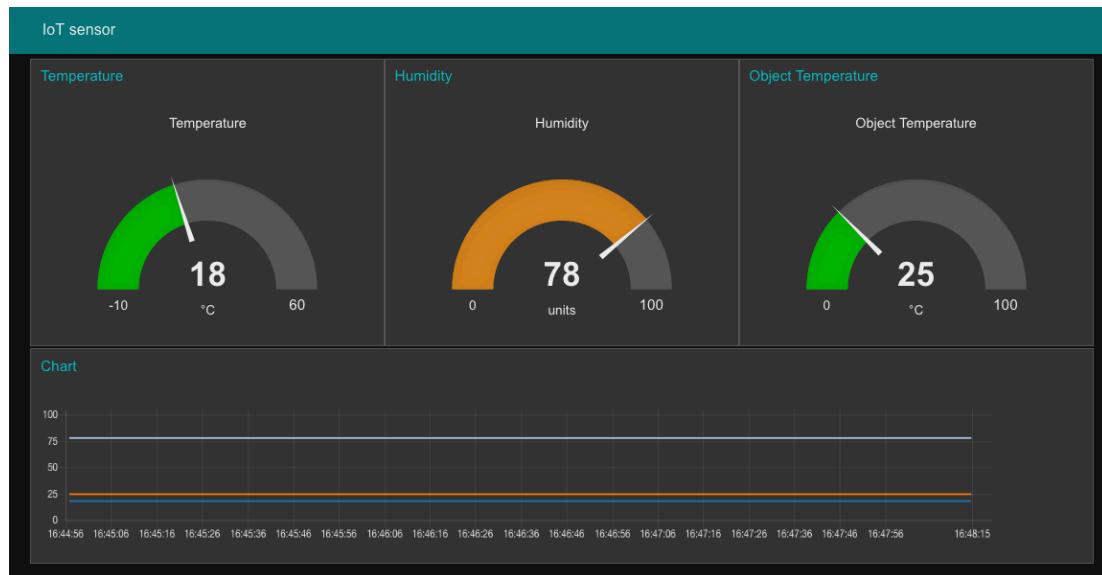


- 3. Connect new nodes to existing...

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

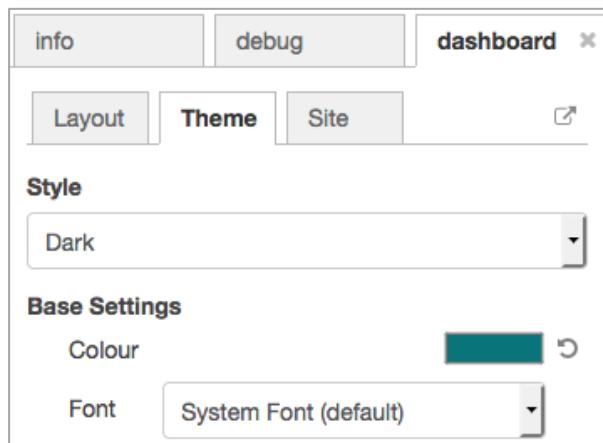


- 4. Deploy the flow: **Deploy**
- 5. Connect to **[http://<YOUR\\_APP\\_HOSTNAME>/ui](http://<YOUR_APP_HOSTNAME>/ui)** (URL generated by Node-red-dashboard node) to see your new dashboard. Change value on virtual sensor app. to see impact on gauges and lines.

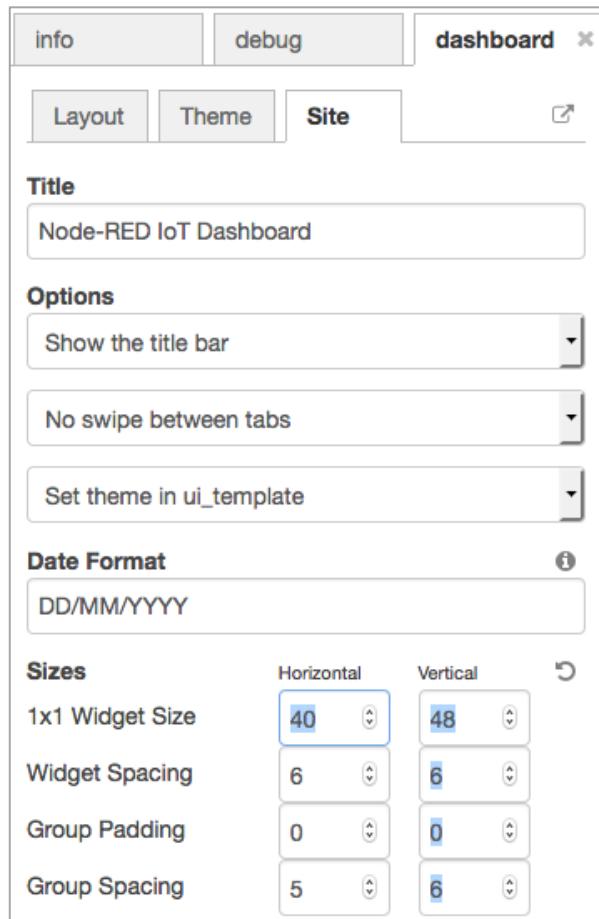


## 6. Customize your dashboard using **Dashboard** tab

- **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

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.

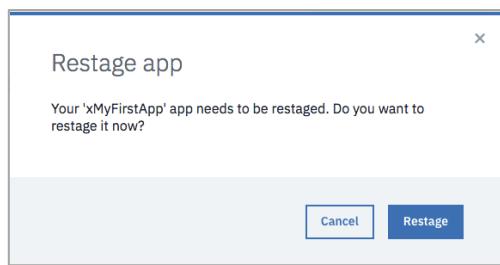
- 1. In the IBM Cloud window, click **Catalog** then **Watson** category.  
Click on **Text to Speech** service to create your own instance.

The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with a 'Platform' section containing links like Boilerplates, APIs, Application Services, etc., and a 'Watson' section which is currently selected and highlighted with a blue border. The main area displays several Watson services: Personality Insights, Speech to Text, Text to Speech (which is highlighted with an orange box), Tone Analyzer, Visual Recognition, and Watson Studio. Each service has a brief description and 'Lite' and 'IBM' buttons below it. A vertical 'FEEDBACK' button is on the right side of the catalog area.

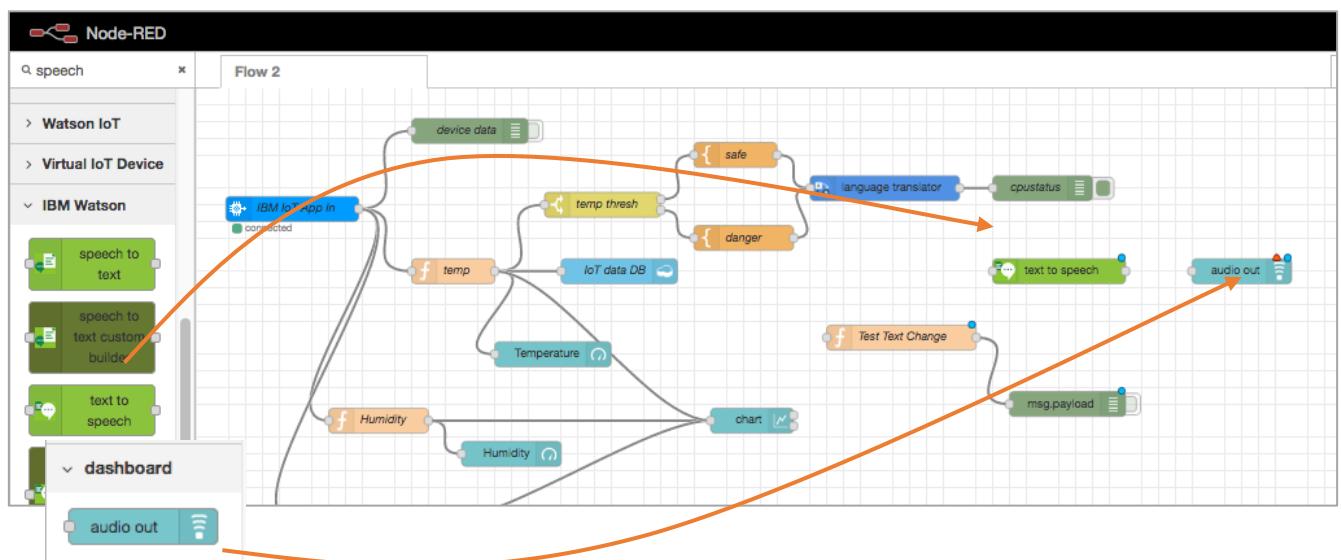
- 2. Enter a name for your service and click **Create**.
- 3. On new service dashboard, click **Connection**, and connect Text to speech service to your Node-RED application: **Connect**.

The screenshot shows the 'Connections' page for the 'Text to Speech' service. On the left, there's a sidebar with 'Connections' selected and highlighted with an orange box. The main area shows a table titled 'Cloud Foundry apps' with two entries: 'xmultipiface' and 'xMyFirstApp'. The 'xMyFirstApp' row is highlighted with an orange box. Each entry has a 'ROUTE' column showing the URL, a 'STATUS' column (with 'Stopped' or 'Running' status), and a 'Connect' button.

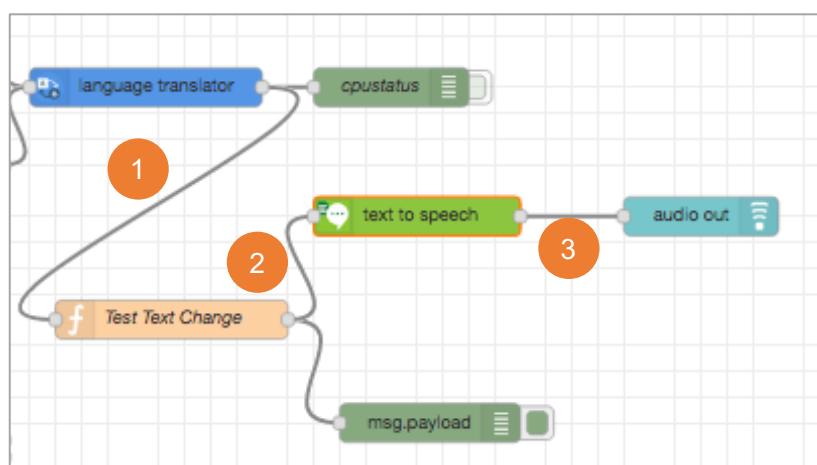
- 4. Accept to restage application and wait for your Node-RED application to restart.



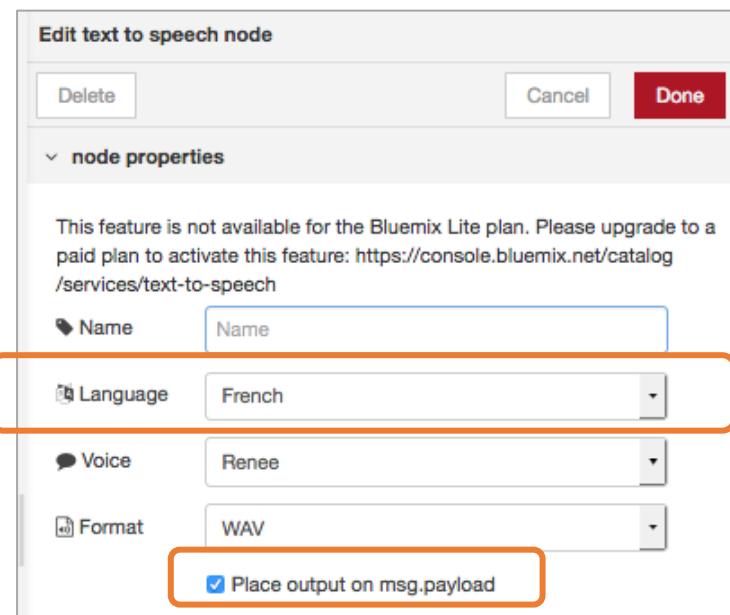
- 5. When restarted, in Node-RED environment, add an **Audio out** node (category **Dashboard**) and a **text to speech** node (category **Watson**) to your flow



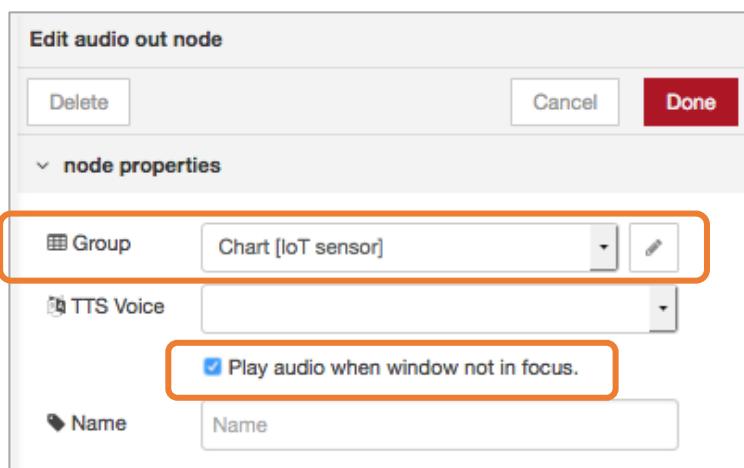
- 8. Connect nodes as below  
 1. **Test text change** to **language translator**  
 2. **Test text change** to **text to speech**  
 3. **text to speech** to **audio out**



- \_\_\_ 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 #2)

### 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.

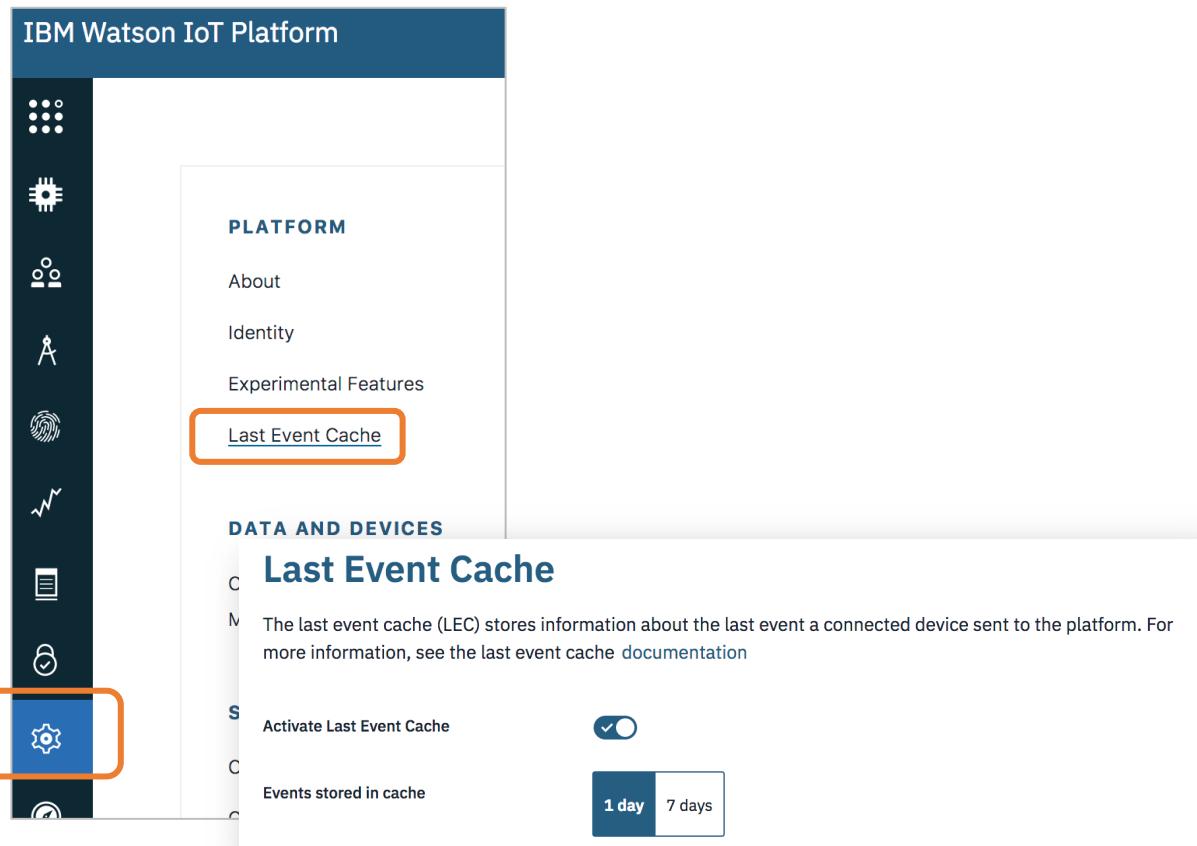
Name	Region	CF Org	CF Space	Status
xMyFirstApp	US South	lalevee@fr.ibm.com	demo	<span style="color: green;">Running</span>

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

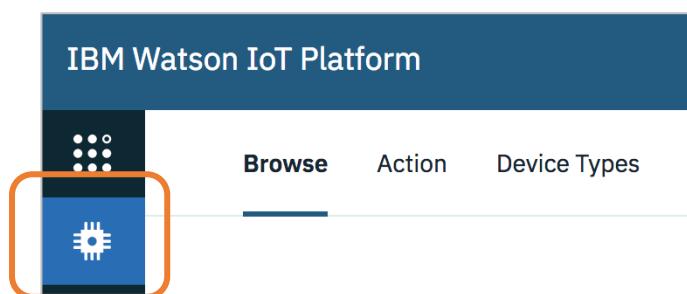
CONNECTION NAME	TYPE
CSI-IoT-cloudantNoSQLDB	Cloudant
CSI-IoT-iotf-service	Internet of Things Platform

- 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.

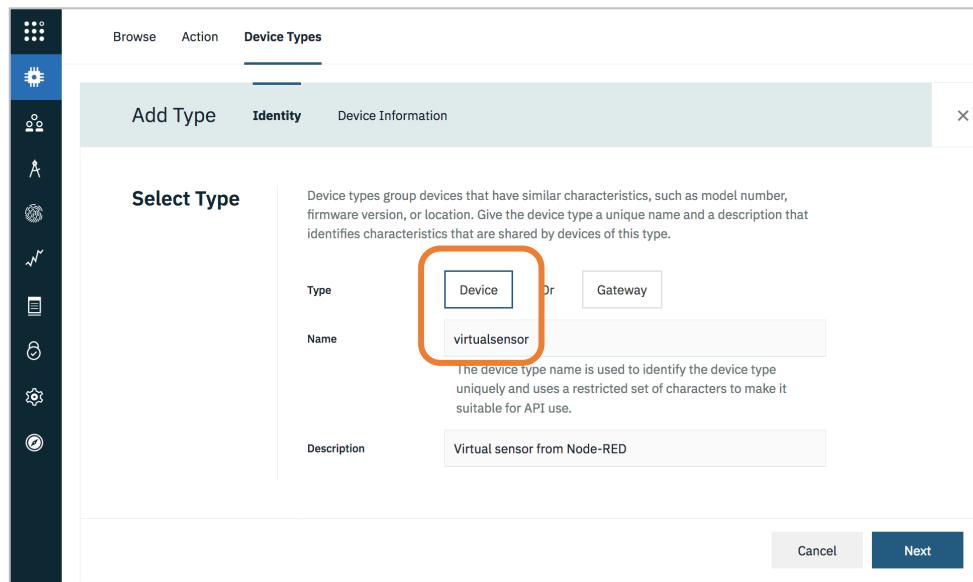
- 5. From left panel **Settings** menu, activate the **Last Event Cache** feature :  
By using the Watson IoT Platform Last Event Cache API, you can retrieve the last event that was sent by a device. This works whether the device is online or offline, which allows you to retrieve device status regardless 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



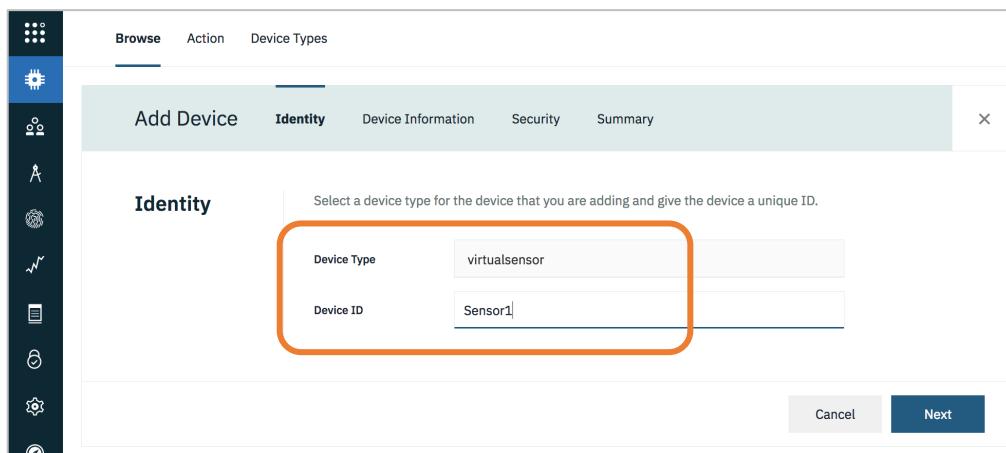
- 7. Choose **Device** and put a name (case sensitive) for your device : *virtualsensor*  
 Click **Next**.  
 You don't need to add Device Information.  
 Click **Done**.



- 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**.

The screenshot shows the 'Device Security' step of the device creation process. The 'Security' tab is selected. A text input field contains 'iotlabtoken'. A note below the field says: 'Make a note of the generated token. Lost authentication tokens cannot be recovered. Tokens are encrypted before being stored.'

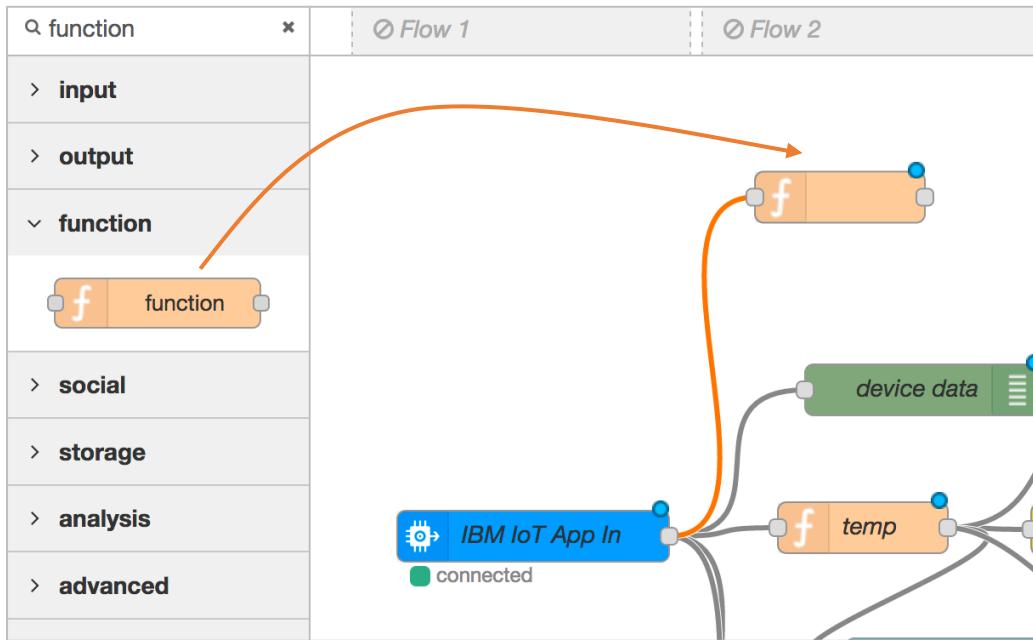
- 12. A summary of your device details appears. Copy all these information in a text editor or as a screenshot. The token is unrecoverable.  
Click **Done**.
- 13. In the **Security** menu on the left panel, click **Connection Security** and change security settings to accept non-SSL connections : **TLS Optional**

The screenshot shows the 'Connection Security' page. The 'Default Rule' section shows 'Scope' as 'Default' and 'Security Level' as 'TLS Optional'. A 'Predicted Compliance' button is also visible.

- 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

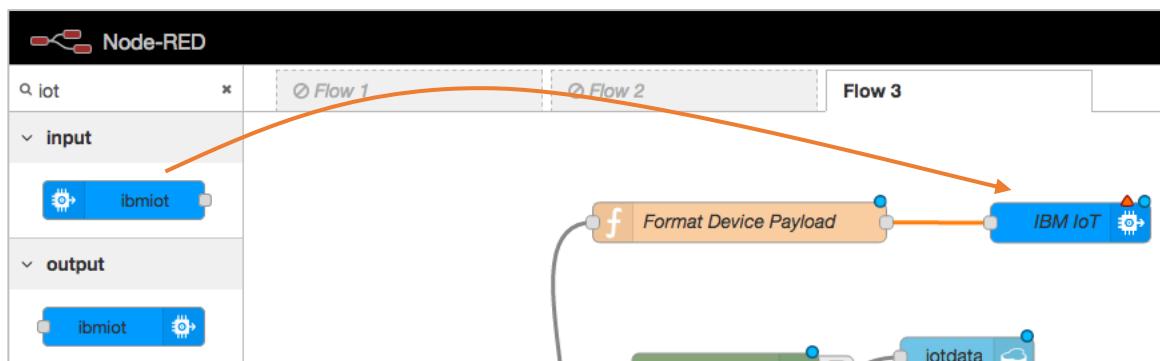


- \_\_\_ 3. Open new node. Name it **Format Device Payload** and insert following code

```
// 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;
```

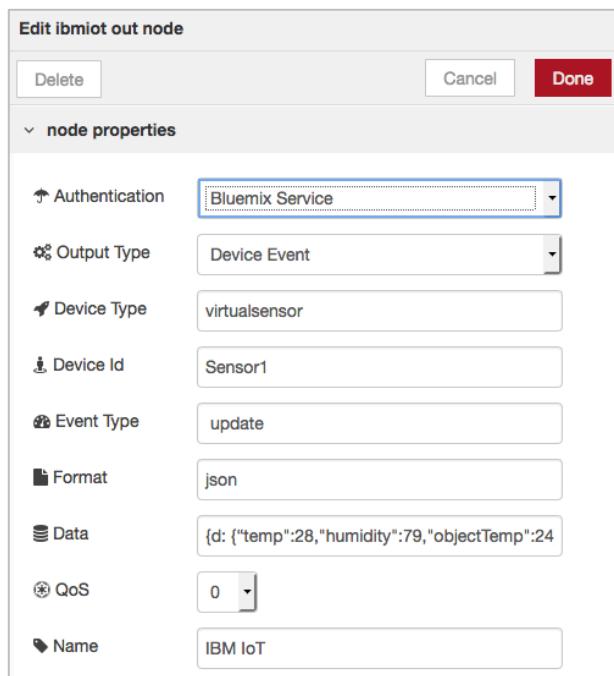
Click **Done**.

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



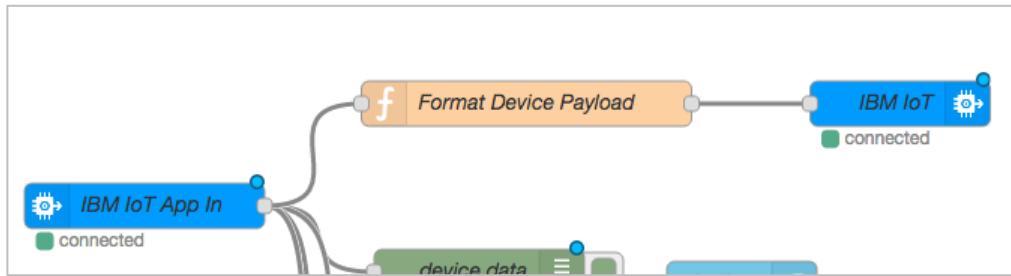
5. Open **ibmiot out** node and configure it as below.

- 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).

A screenshot of the IBM Watson IoT Platform's Device Details page. At the top, there are four filter dropdowns: "Device ID", "Device Type", "Class ID", and "Date Added". To the right are icons for delete, edit, and add. Below these is a table header with columns: "Device ID", "Device Type", "Device", and "Date Added". A single result is shown: "Sensor1" (Device ID), "virtualsensor" (Device Type), "Device" (Device), and "Jun 20, 2018 6:20 PM" (Date Added). This row is highlighted with an orange border. Below the table are tabs: "Identity", "Device Information", "Recent Events", "State" (which is highlighted with an orange border), and "Logs". Under the "State" tab, there is a section titled "Showing Raw Data | No Interfaces Available". Below this is a table with columns: "Property", "Value", "Type", "Event", and "Last Received". The table contains four rows of data:

Property	Value	Type	Event	Last Received
d		Object	update	a few seconds ago
temp	28	Number	update	a few seconds ago
humidity	79	Number	update	a few seconds ago
objectTemp	24	Number	update	a few seconds ago

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

**Recent Events**

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
update	{"d": {"temp": 28, "humidity": 79, "objectT...}}	json	a few seconds ago
update	{"d": {"temp": 28, "humidity": 79, "objectT...}}	json	a few seconds ago
update	{"d": {"temp": 28, "humidity": 79, "objectT...}}	json	a few seconds ago
update	{"d": {"temp": 28, "humidity": 79, "objectT...}}	json	a few seconds ago
update	{"d": {"temp": 28, "humidity": 79, "objectT...}}	json	a few seconds ago

**Event Payload**

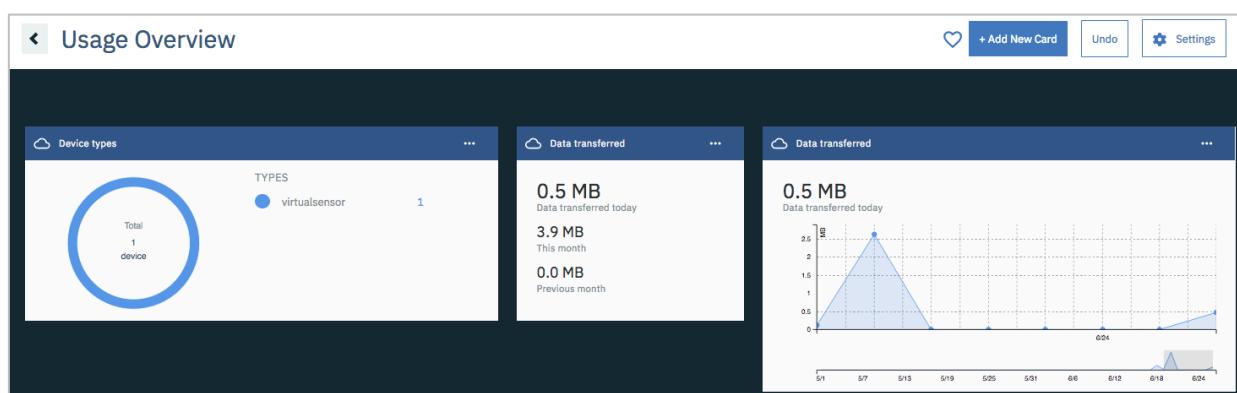
```

Event Name           update
Time Received       Jun 26, 2018 11:30 AM
1 <[
2   "d": {
3     "temp": 28,
4     "humidity": 79,
5     "objectTemp": 24
6   }
7 ]
```

- 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.



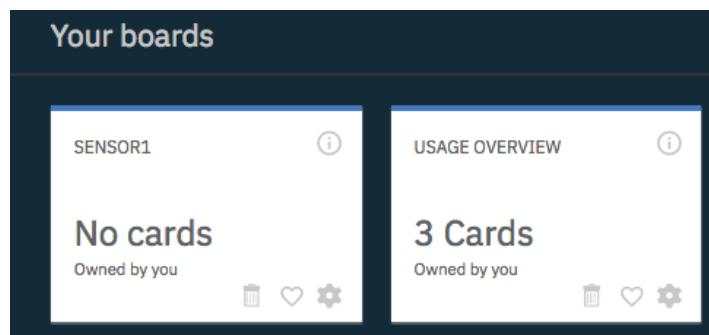
- 7. Click < (top left corner).  
Click on **Create New Board**.



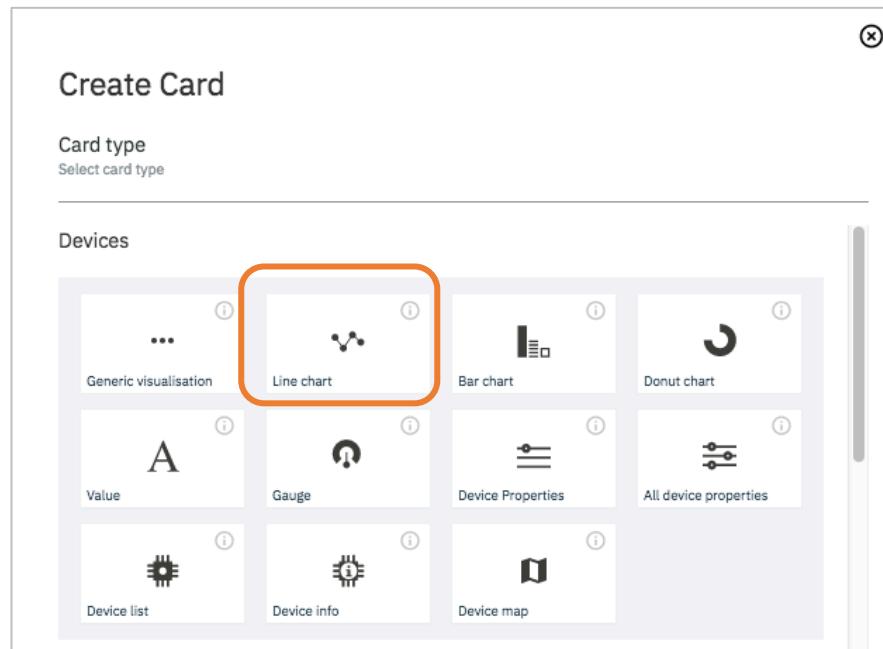
- 8. Click < (top left corner).  
Click on **Create New Board**.  
Enter a name to your new board: *Sensor1*

A screenshot of a "Board settings" dialog box. On the left is a sidebar with "Information" and "Members" sections. The main area has a title "Board settings" with a close button. It asks for a name and description. The "Board name" field contains "Sensor1". Below it is a "Description" field. At the bottom are two radio buttons: "Make this board my landing page." and "Favorite (this also adds this board to your navbar)".

- 9. Click **Next**, then **Summit**.  
— 10. Click on your new board



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

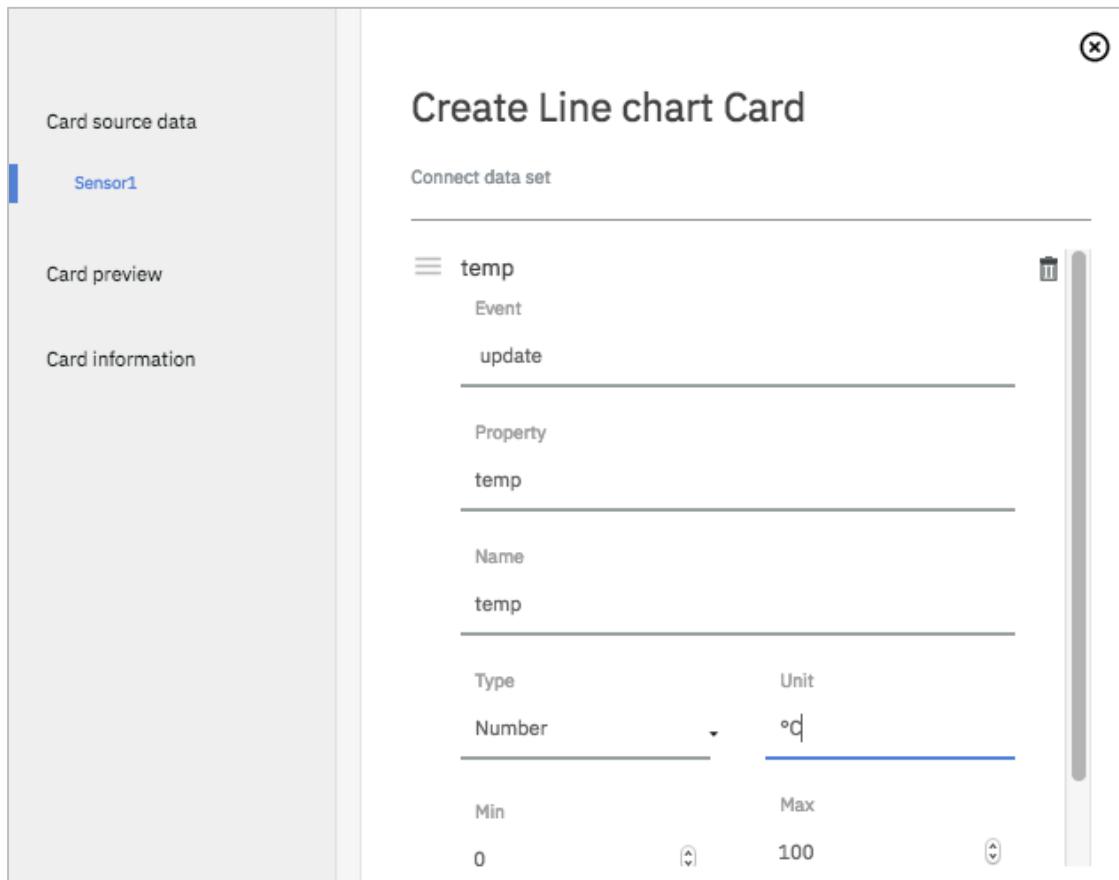


- 12. To collect data from your device, select Sensor1. Click **Next**.

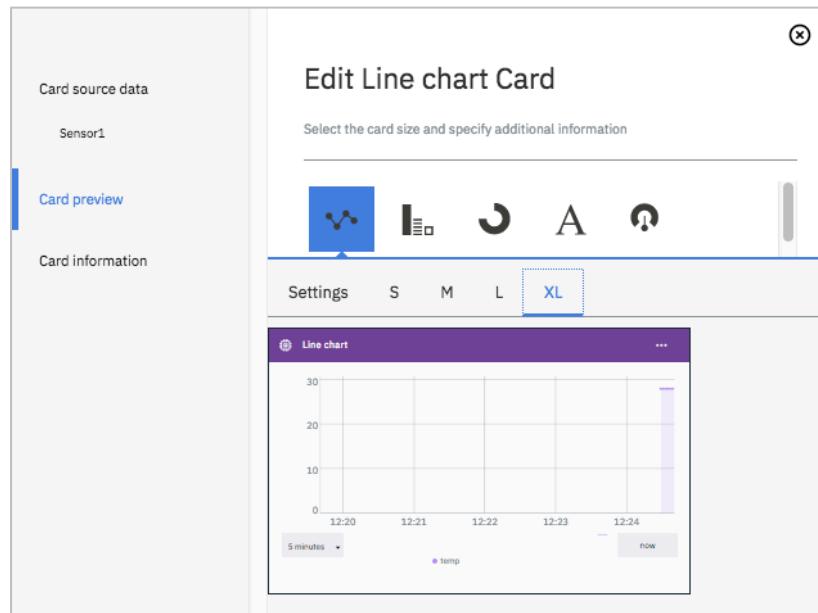
The screenshot shows the 'Create Line chart Card' interface. On the left, there are three tabs: 'Card source data' (selected), 'Card preview', and 'Card information'. The main area is titled 'Create Line chart Card' and has a sub-section 'Devices'. It says 'Specify the data source for the card'. Below that is a search bar with the placeholder 'Search for card data sources using the filter:' and a magnifying glass icon. A table lists data sources. The first row, which contains 'Sensor1' under 'Device ID' and 'virtualsensor' under 'Device Type', is highlighted with an orange rectangular box.

Device ID	Device Type
Sensor1	virtualsensor

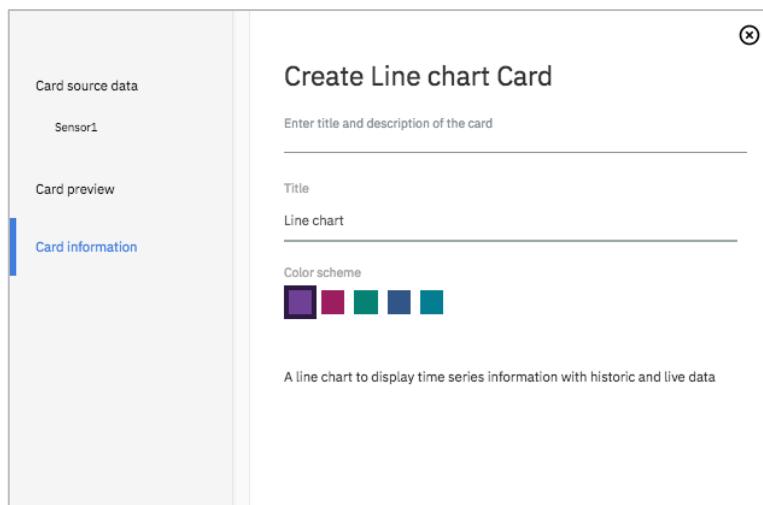
- 13. To select value to display, click on Connect new data set, and fill form as below.  
Click **Next**.



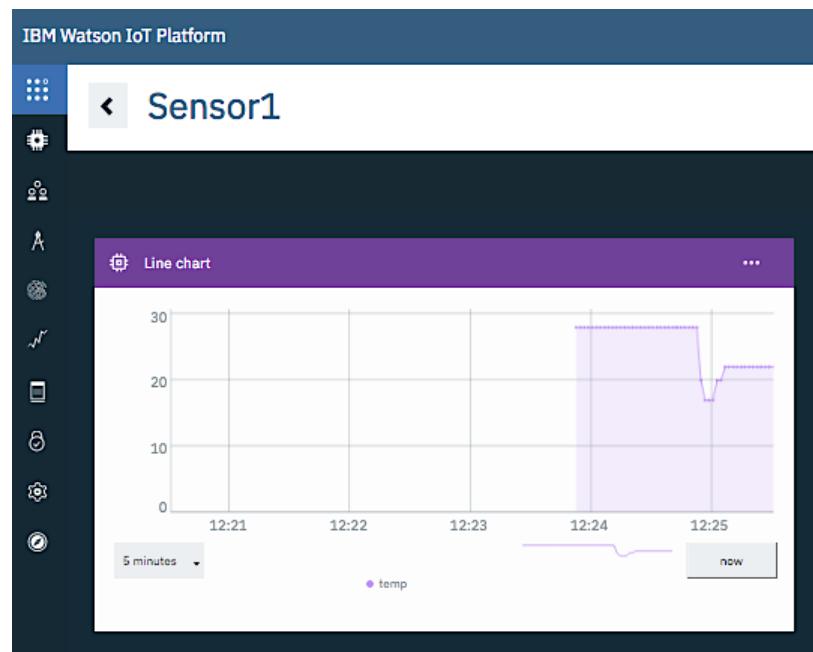
- 14. Change the card size to XL (you can change others parameters, if you want).  
Click **Next**.



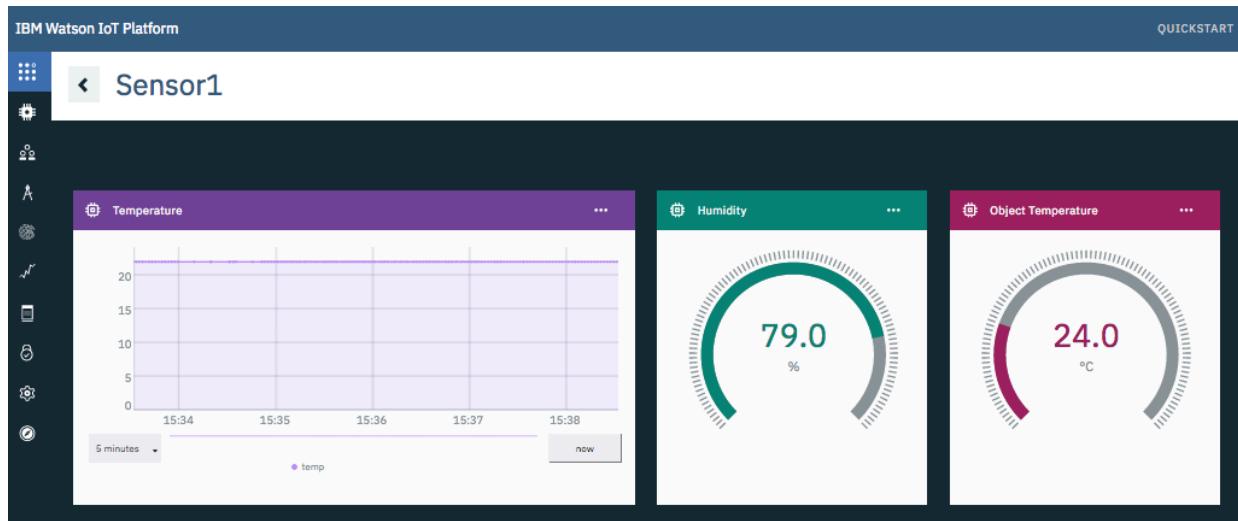
\_\_\_ 15. Enter a card title. Click **Summit**.



\_\_\_ 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**.



---

END OF LAB

---