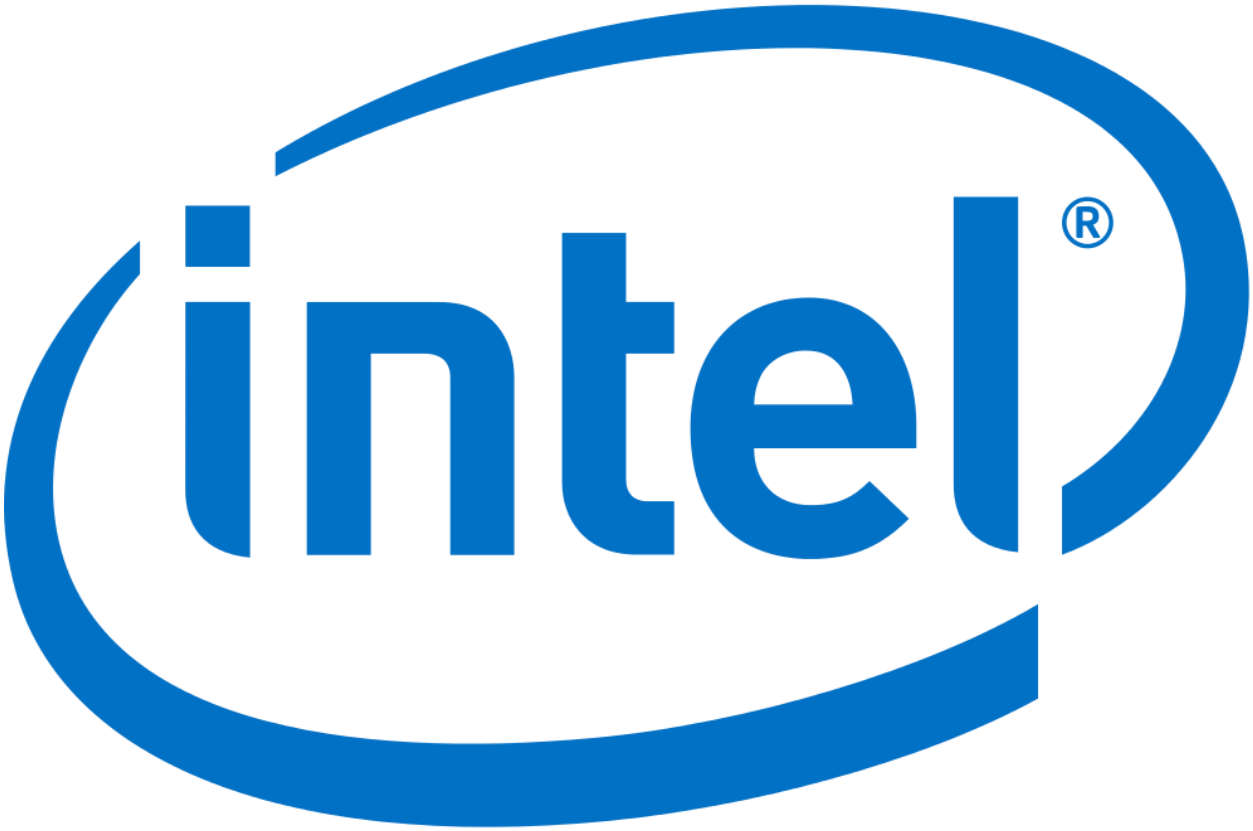
Intel IoT Gateway / Arduino 101 with IBM Watson IoT

Hands-On Workshop

Revision 8



Topics covered: Intel IoT Gateway, Grove sensors, Node-RED, IBM Cloud, Watson Internet of Things Platform, Cloudant NoSQL Database and Twitter.

In this lab, we will unbox and set up an Intel IoT Gateway and the Arduino 101 board (with a Grove Starter kit) along with several services available in IBM Cloud to monitor the temperature and alert maintenance of a high temperature. Using Node-RED, running on the Intel NUC Gateway, the application will read the temperature value from a Grove temperature sensor. If the temperature rises, a LED light is turned on. The temperature will be sent via Watson's Internet of Things Platform service to a Node-RED application hosted on IBM Cloud. If the temperature reaches 30°C, we'll send a tweet. We will send a command to the Arduino 101 to control the background color of the LCD screen connected to the Arduino 101. We'll store the temperature in a Cloudant NoSQL database so we can track patterns. Finally, we'll create an HTTP endpoint that exposes the historical temperatures that third-party applications could consume and perform analysis or other fun stuff.

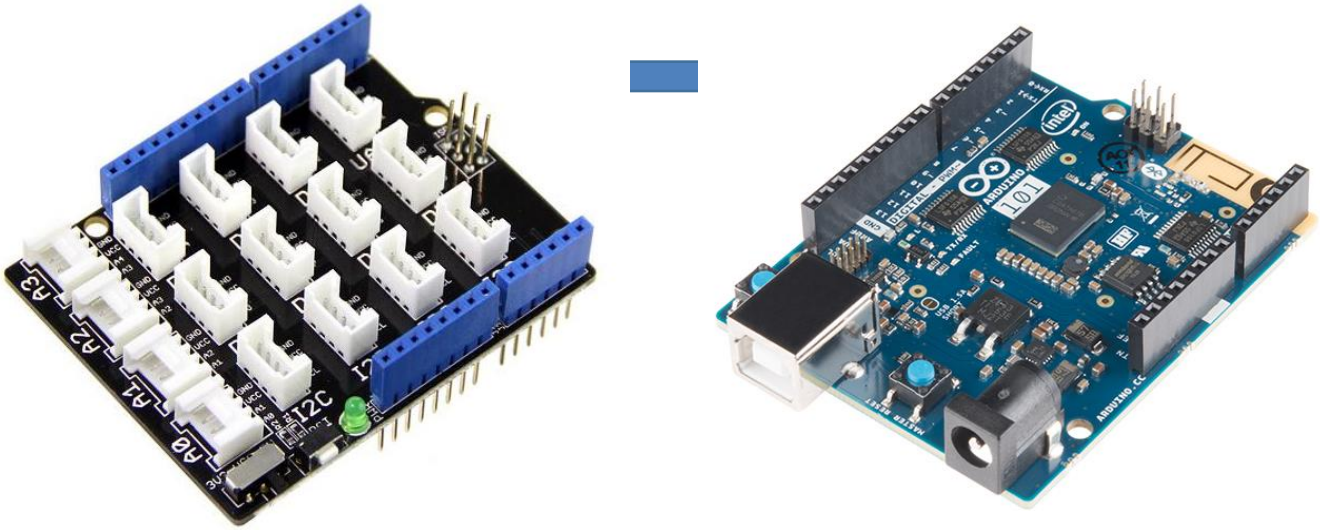
Getting Started with Grove IoT Commercial Developer Kit

Step 1 - Unboxing Grove IoT Commercial Developer Kit and Arduino 101

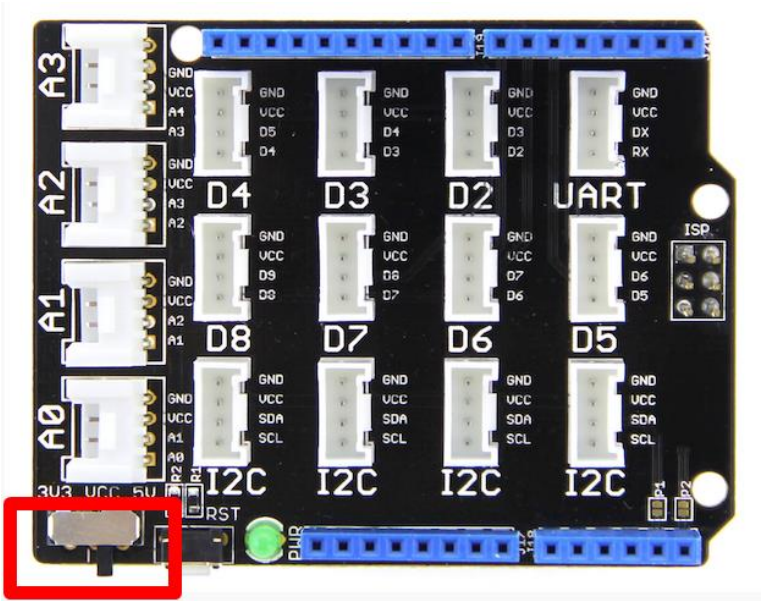
Arduino 101 is not included in Grove IoT Commercial Development Kit and will be given separately.

Step 2 – Connect Base shield to Arduino 101

Base shield will be included in the Grove starter kit. Attach the Grove base shield to the Arduino 101.

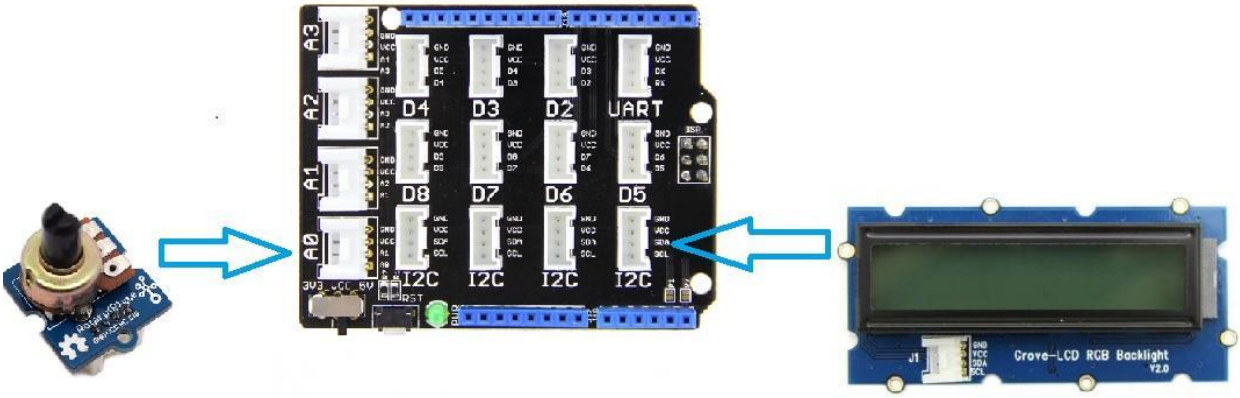


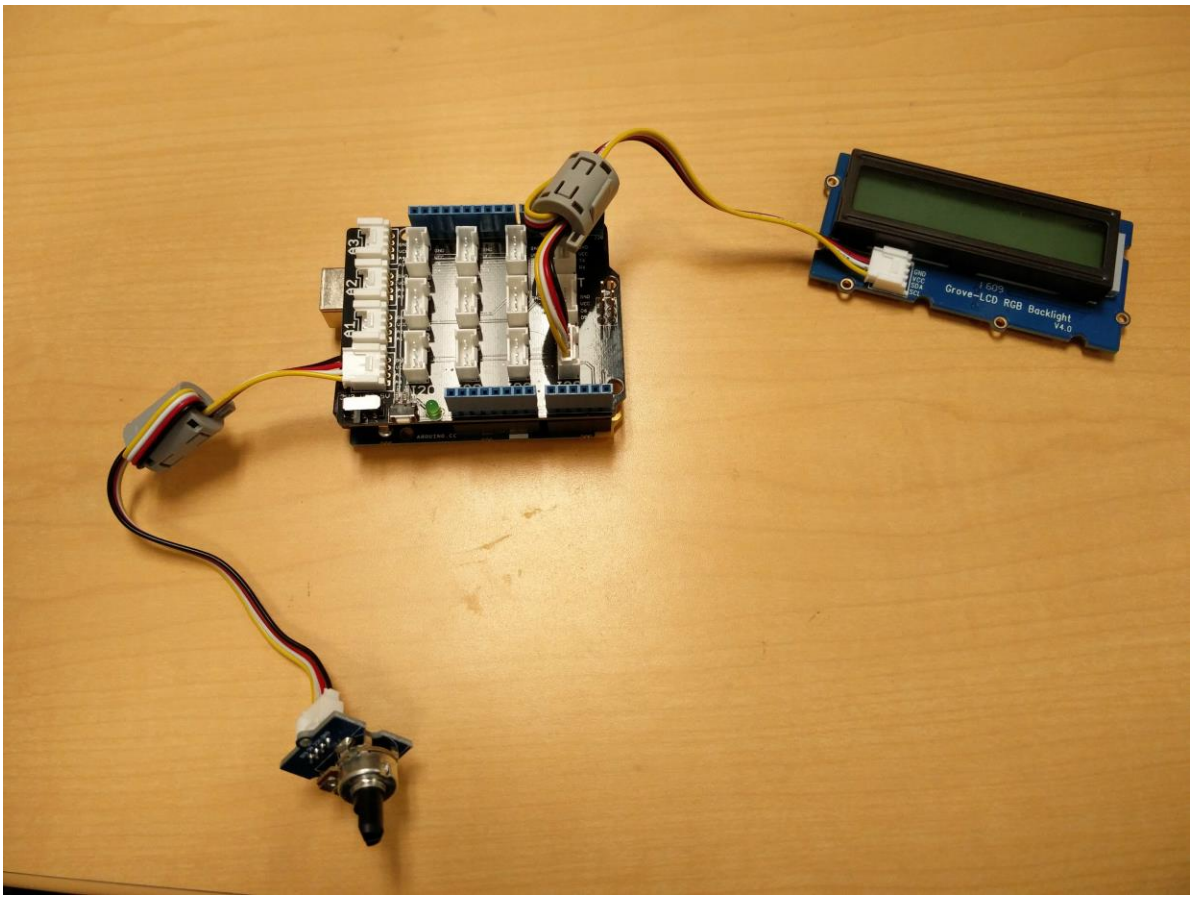
Step 3 – Move Base Shield switch towards 5 Volt



Step 4 – Connect LCD and Rotary Angle to Arduino 101

Connect LCD to any I2C socket and rotary angle to A0 socket with a grove connector cable as shown below, the cables are in the green box.

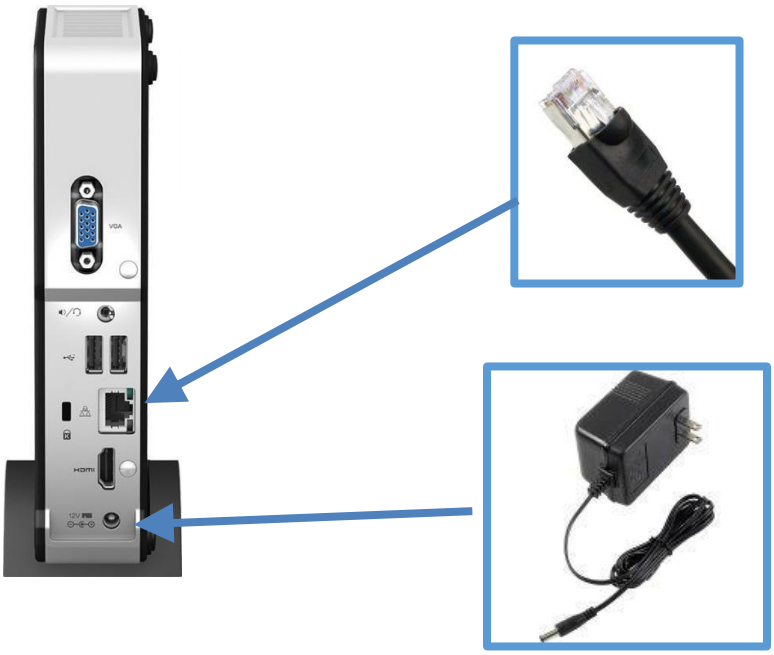




Step 5 – Connect Arduino 101 and Intel IoT Gateway via USB A-B Cable

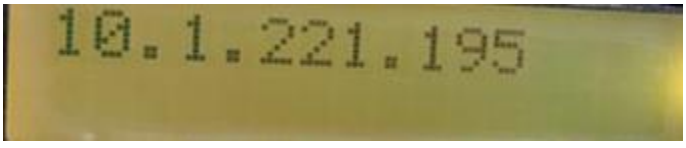
Now connect Arduino 101 with Intel IoT Gateway via USB A-B cable.

Step 6 – Connect Intel IoT gateway with Ethernet cable and Power adapter



Step 7 – Press Power button to start Gateway

Press power button to start gateway. After around 2 minutes, it will display the IP address of Gateway on the LCD. **Note Down the IP address.**



Step 8 – Open your browser and go to IP address to access the IoT Developer Hub

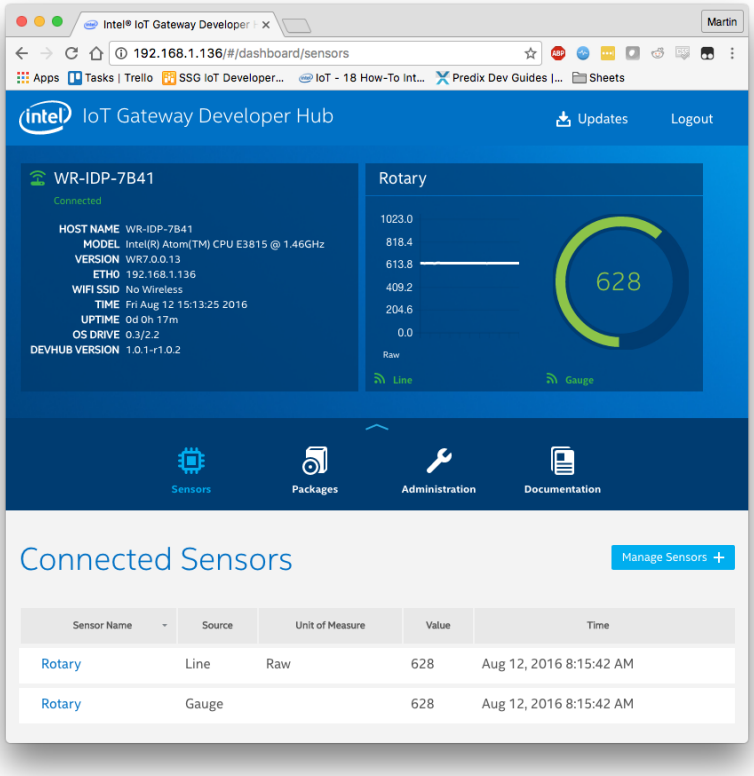
- Login to the Intel® IoT Gateway Developer Hub using **root** for both the username and the password.
- Use the Intel® IoT Gateway Developer Hub to explore and extend the basic capabilities of the Gateway.

Intel NUC Developer Hub Overview

The Developer Hub is a front-end interface for the Gateway. It has 5 main functions:

- Display sensor data and basic Gateway information on a configurable dashboard
- Access the Node-RED development environment
- Add repositories, install and upgrade packages
- Administer the Gateway – update OS, configure network setting
- Access documentation

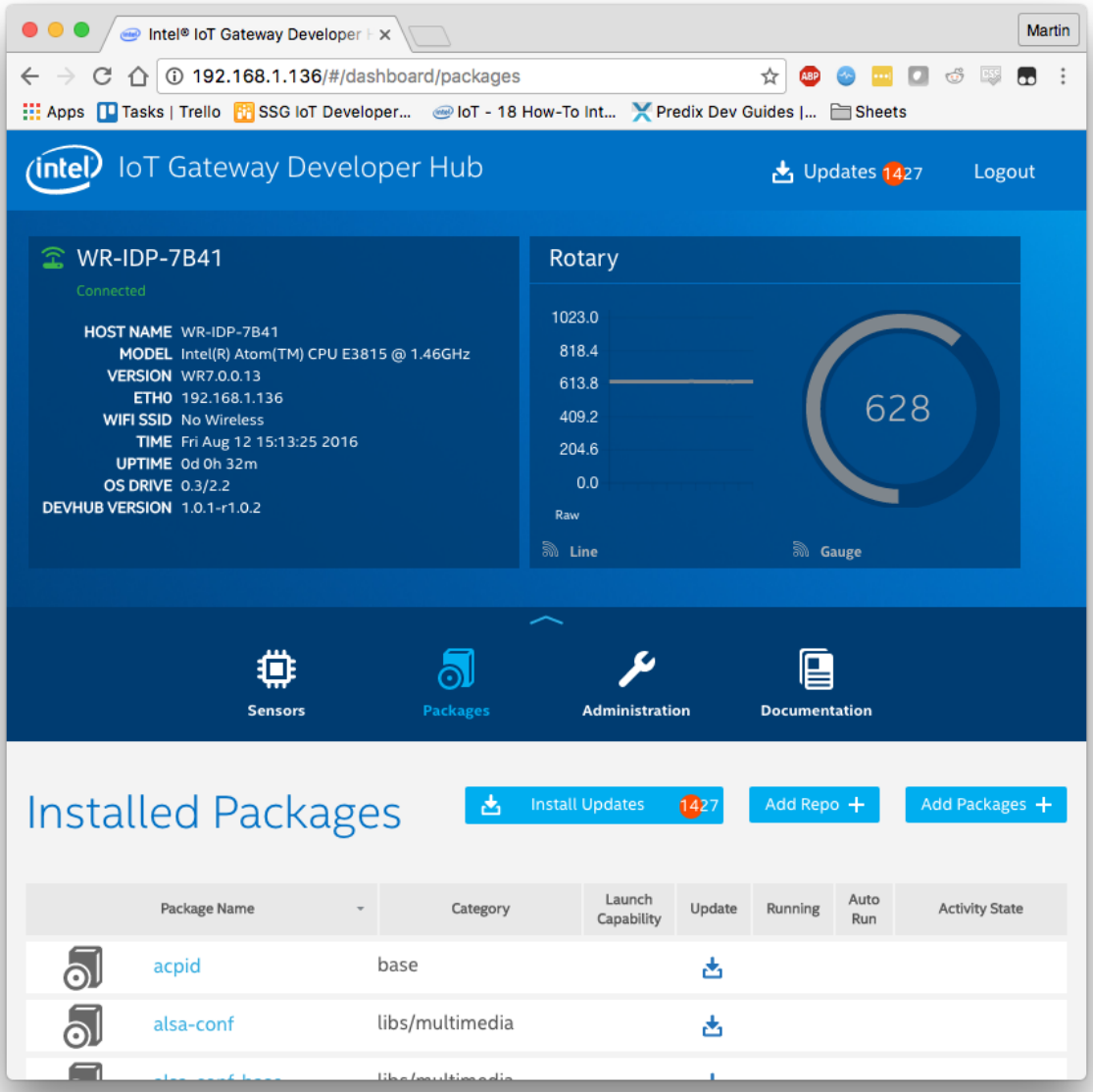
Dashboard



Dashboard with Rotary Angle Sensor Displayed

By default, the Gateway is programmed to display the value of the rotary angle sensor (knob) on the dashboard. Twist the knob to see the gauge move.

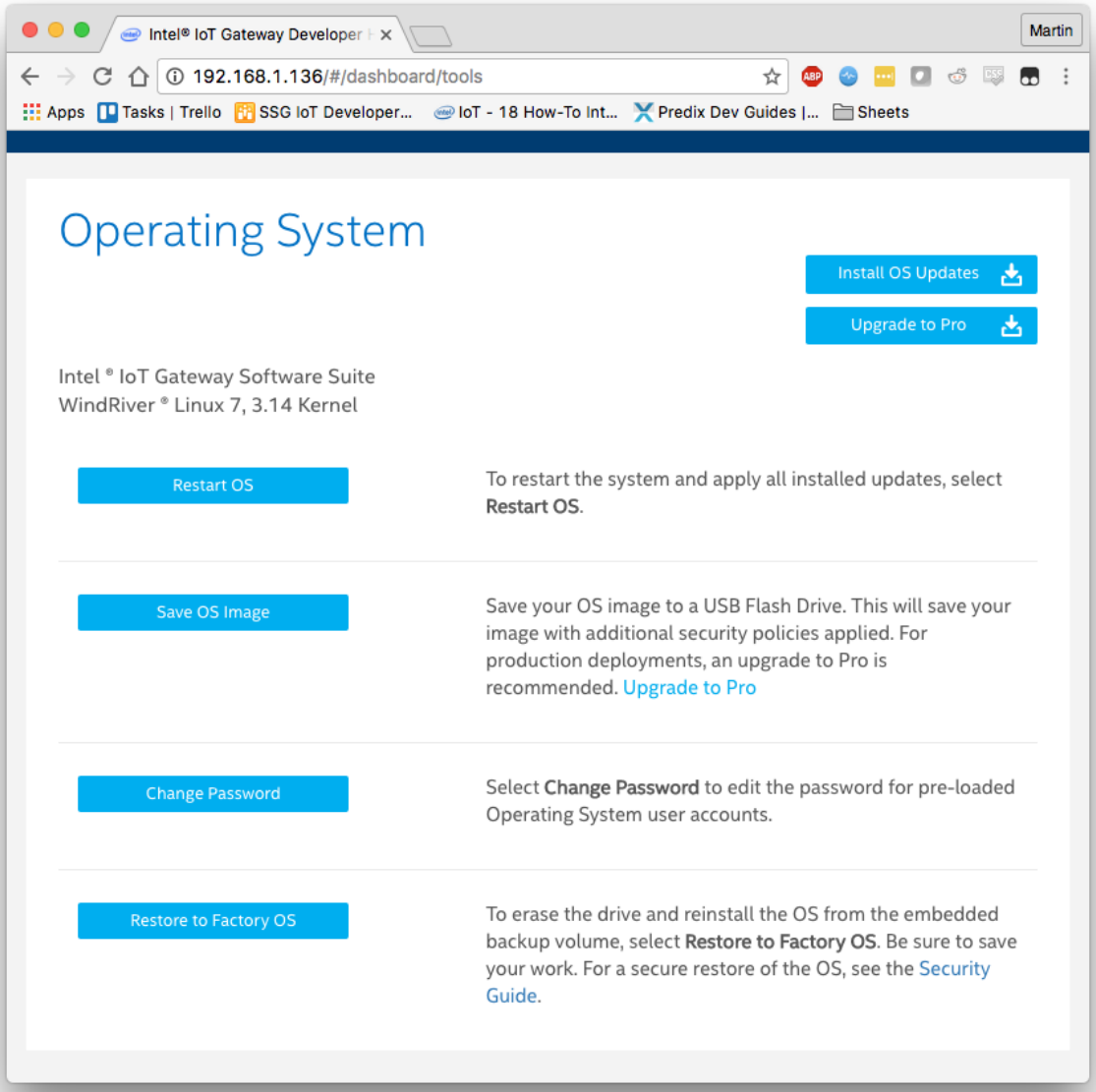
Package Manager



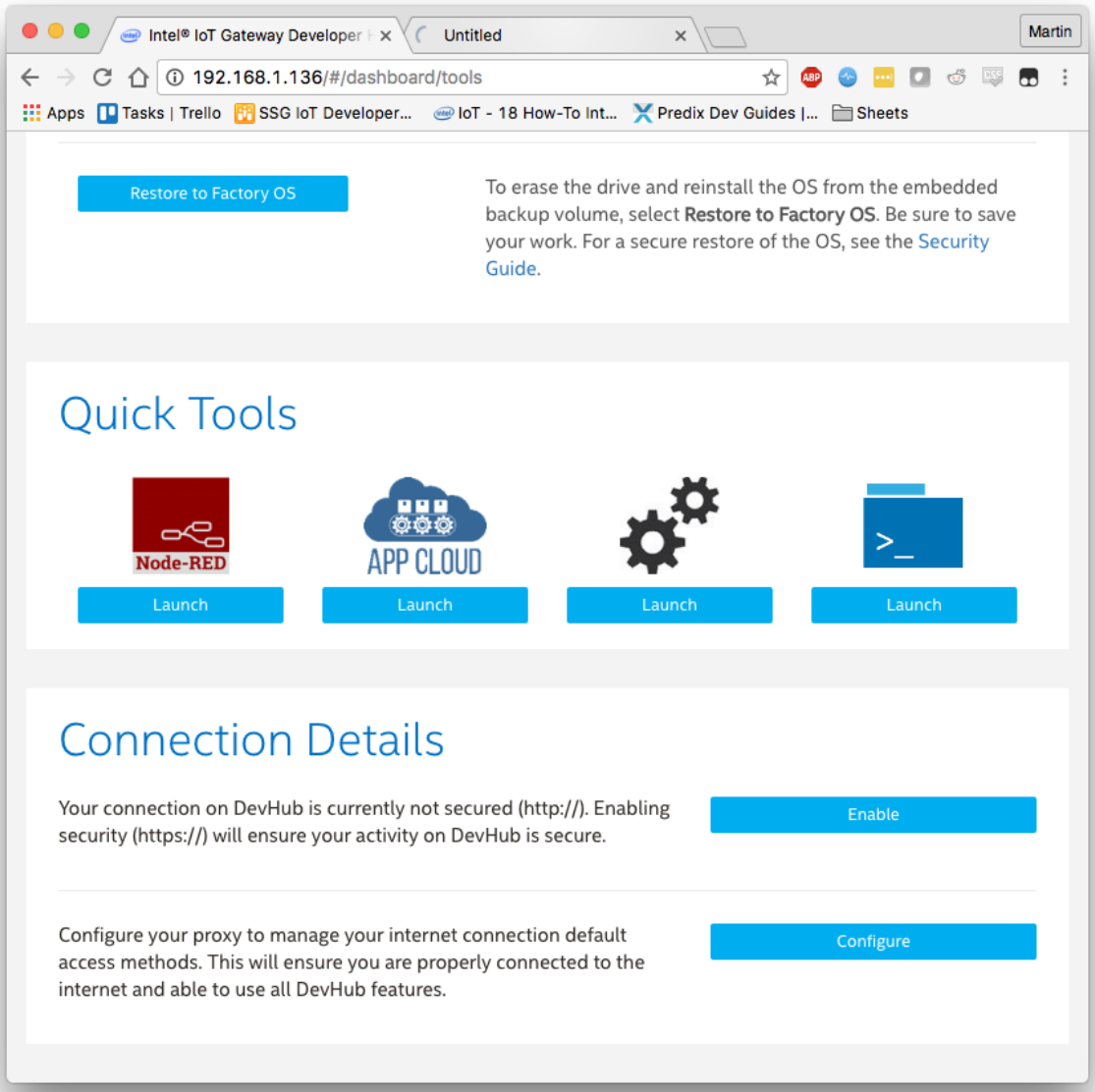
View of Packages screen

On the package manager interface you will most likely see an option to “Install Updates”.
Do Not Install Updates!
This process can take a significant amount of time and will use up a lot of bandwidth.

Administration

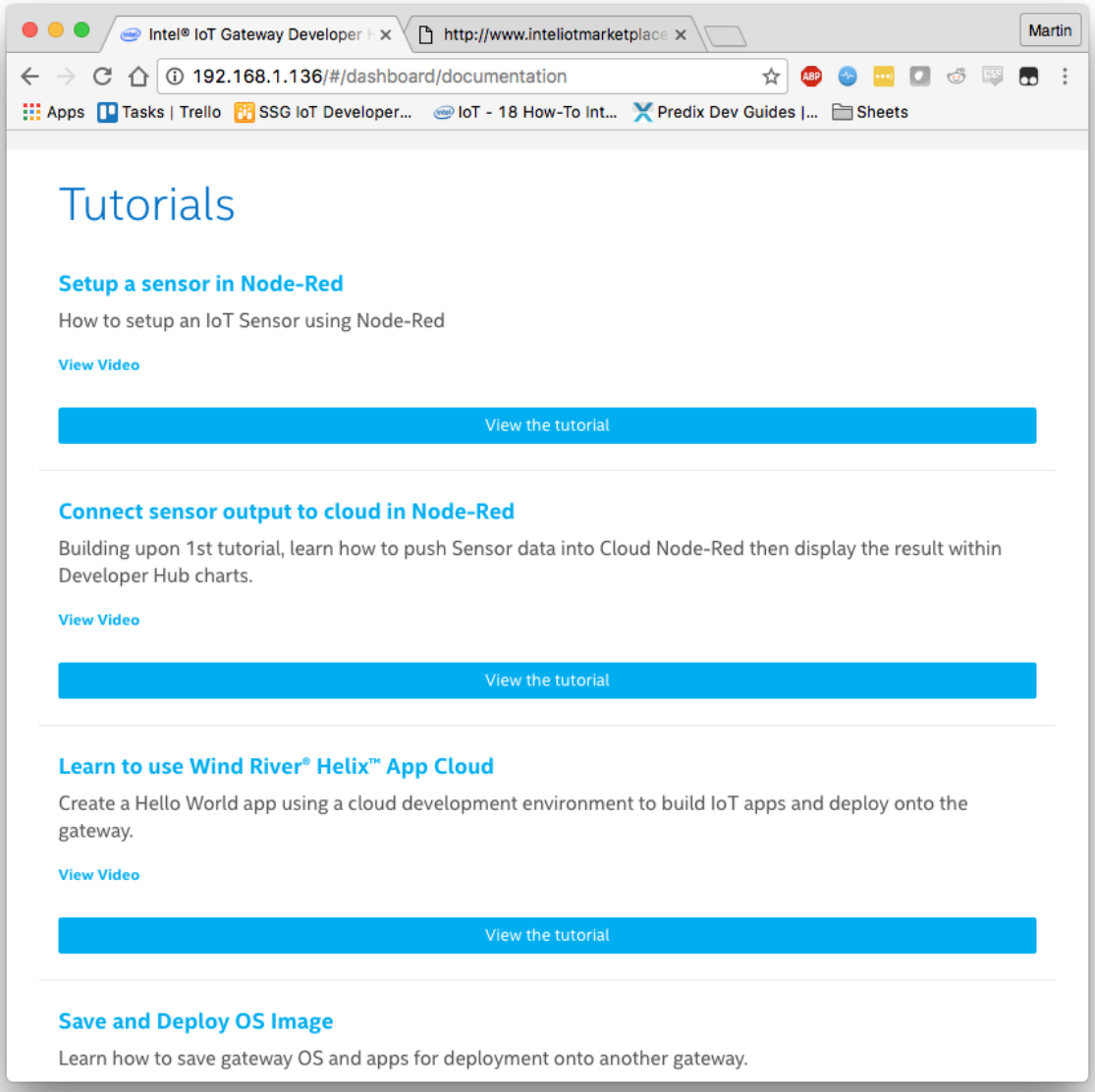


In the Administration interface you will see options to Update the OS, Upgrade to Pro, as well as others. **Please do not select any of these options!**



Further down on the page you will see links to some quick tools. Feel free to explore these, however: **Please do not alter any settings in the configuration tool (two gears)!**

Documentation



The Documentation tab will give you links to some of the documentation for using the Gateway.

Connect to your Gateway with a terminal client

In order to perform advanced configuration of the Gateway either a monitor and keyboard, or a Secure Shell (SSH) connection is required. On Mac and Linux there are default programs that can do this - Screen and SSH respectively. However, on Windows no default exists, however Putty is light weight and easy to install and can be used to SSH into the Gateway.

For Windows Users:

- Visit the PuTTY download page.
- Under the "For Windows on Intel x86" heading, click on the "putty.exe" link to download the latest release version to your computer.

- Double-click putty.exe on your computer to launch PuTTY.
- Enter IP address of the Gateway
- You can login with the username **root** and the password **root**.

For Mac and Linux Users:

- Open a Terminal
- Type $ *ssh root@<<IP Address>>. Replace <<IP Address>> with the IP address of your gateway.*
- Enter ***root*** as password

Update Node-RED and UPM Nodes

Before continuing we are going to ensure we have the latest available Node-RED version (0.14.6) and UPM nodes installed on the gateway.

Run the following commands in your SSH terminal window to update Node-RED:

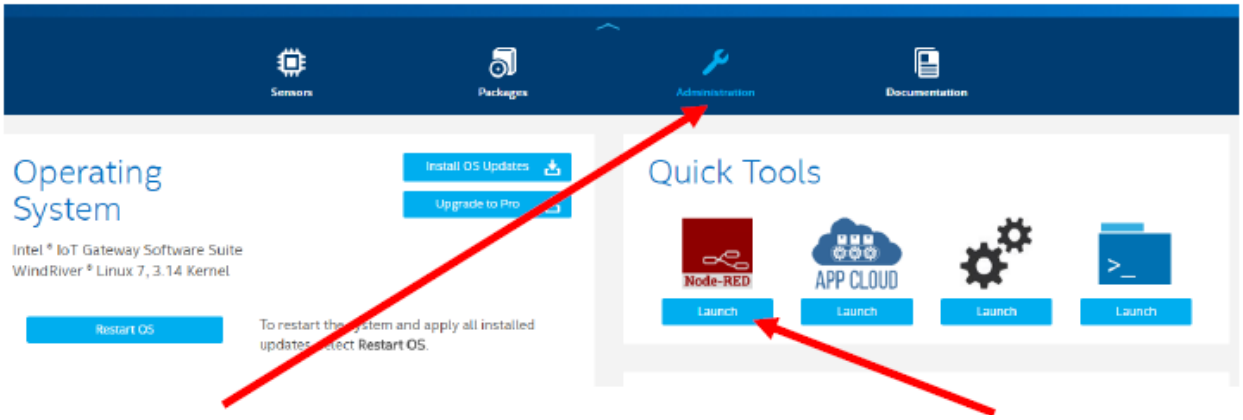
```
smart update
smart upgrade node-red node-red-experience-devkit
```

Finally restart Node-RED by running the following command:

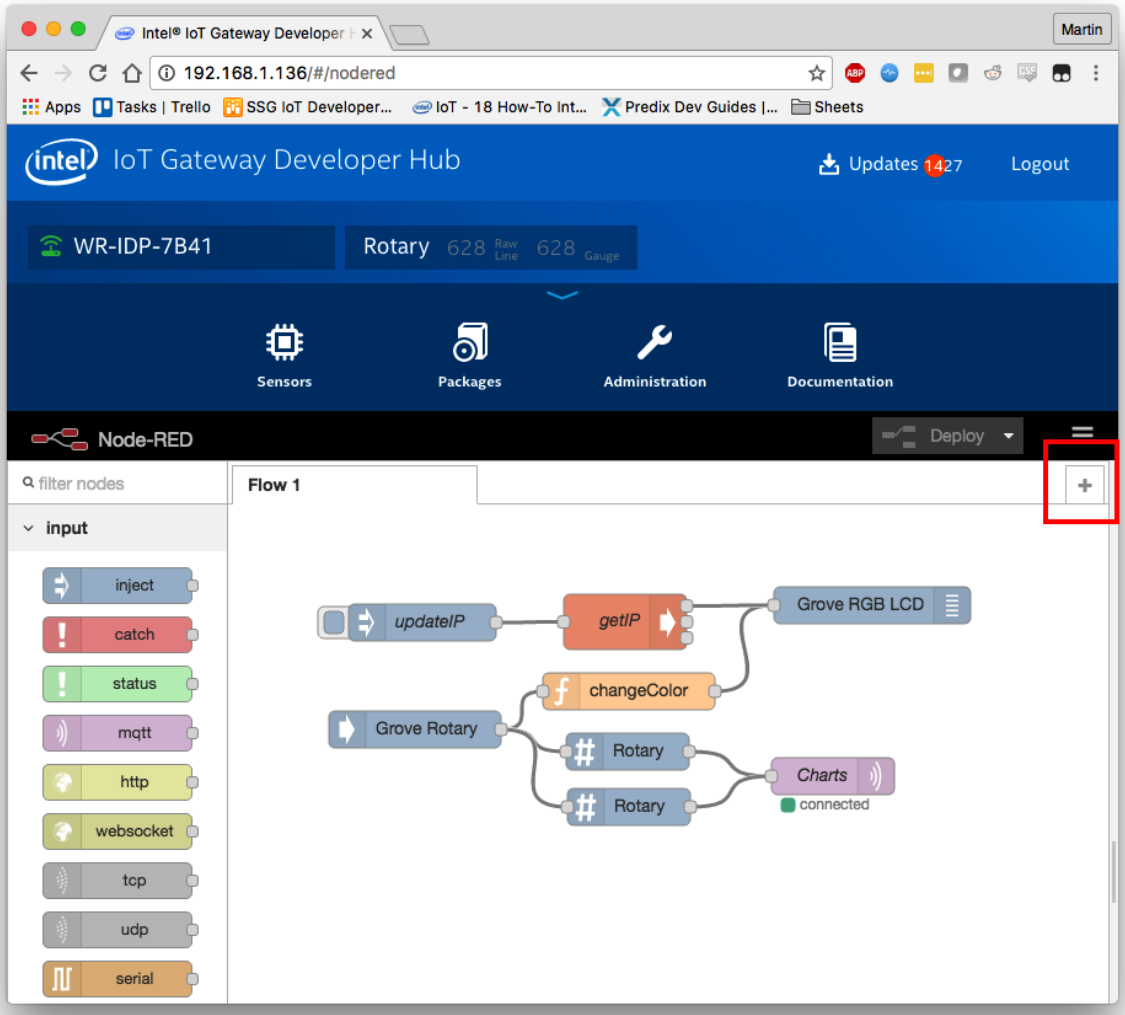
```
systemctl restart node-red-experience
```

Starting Node-RED for Blinky LED

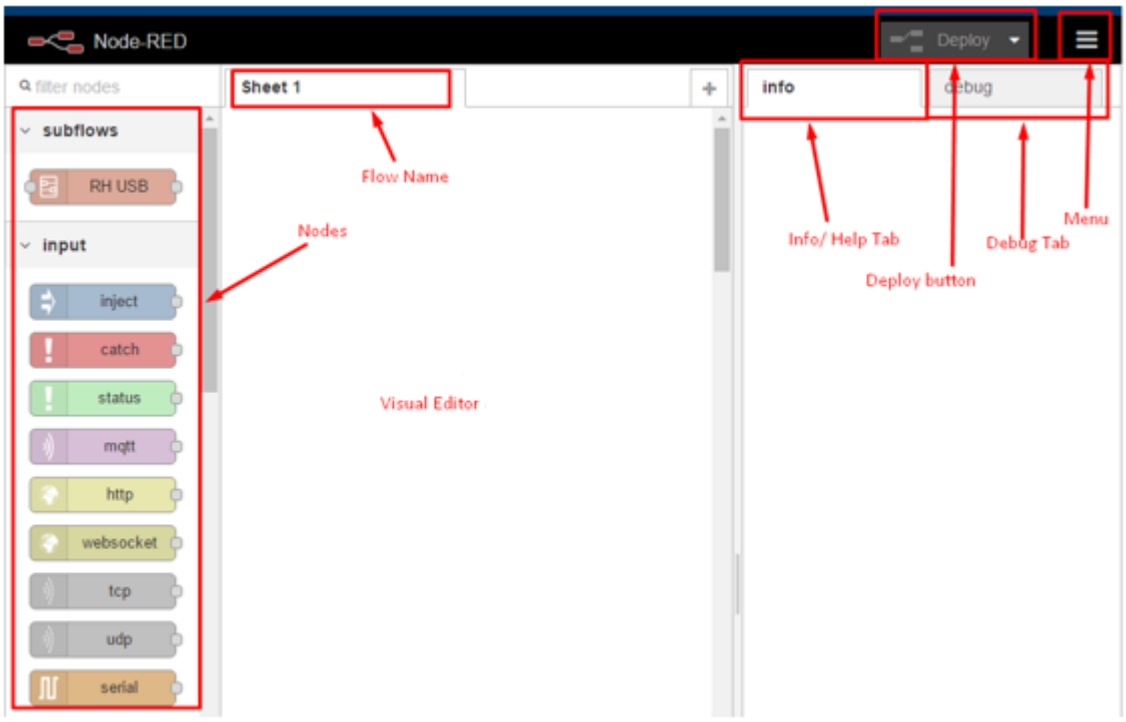
- Go to Development Hub by entering the IP Address assigned to your Gateway (E.g. http://192.168.1.1) in your browser.
- To load the Node-RED interface, click **Administration** on the navigation ribbon. Click **Launch** under the Node-RED icon.



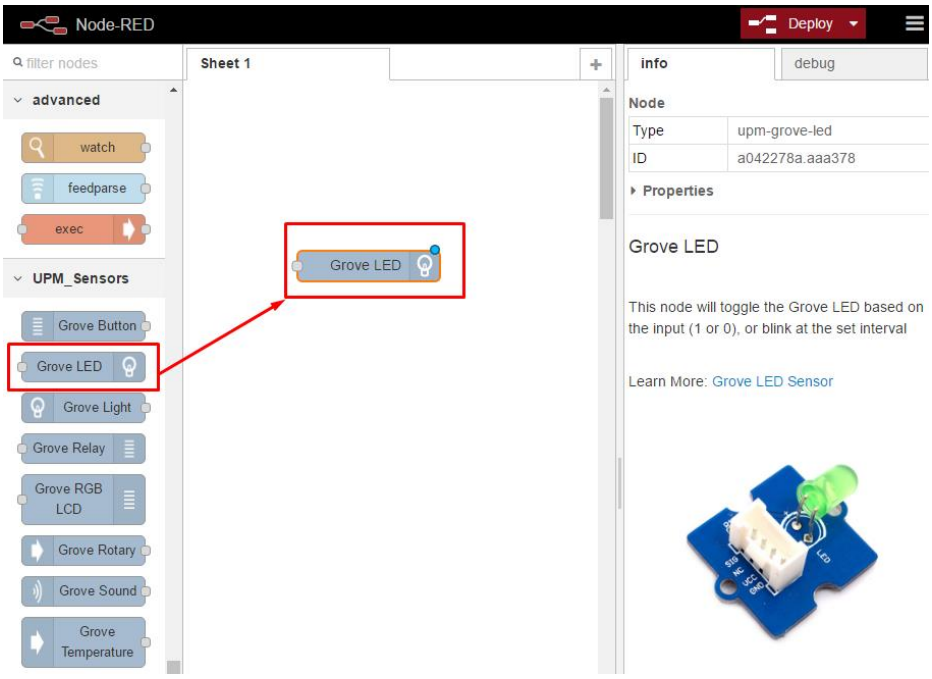
- Node-Red is a visual tool for writing the Internet of Things (IoT).
- Enter username and password as **root** and **root** respectively.



You will most likely have an empty workspace but if the default flow which shows the IP address on the LCD screen is still present (as in the screenshot above) create a new flow to program the rest of the exercises. Click the **+** button in the upper right corner of the Node-RED interface if necessary.

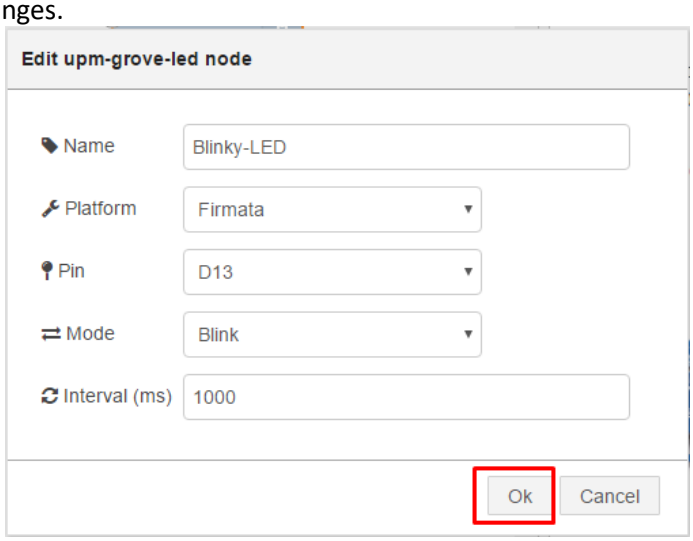


- Deploy Blinky LED Node. Go to Nodes, at the bottom of page there will be group on Nodes called UPM Sensors. Drag Grove LED from Nodes to Visual Editor

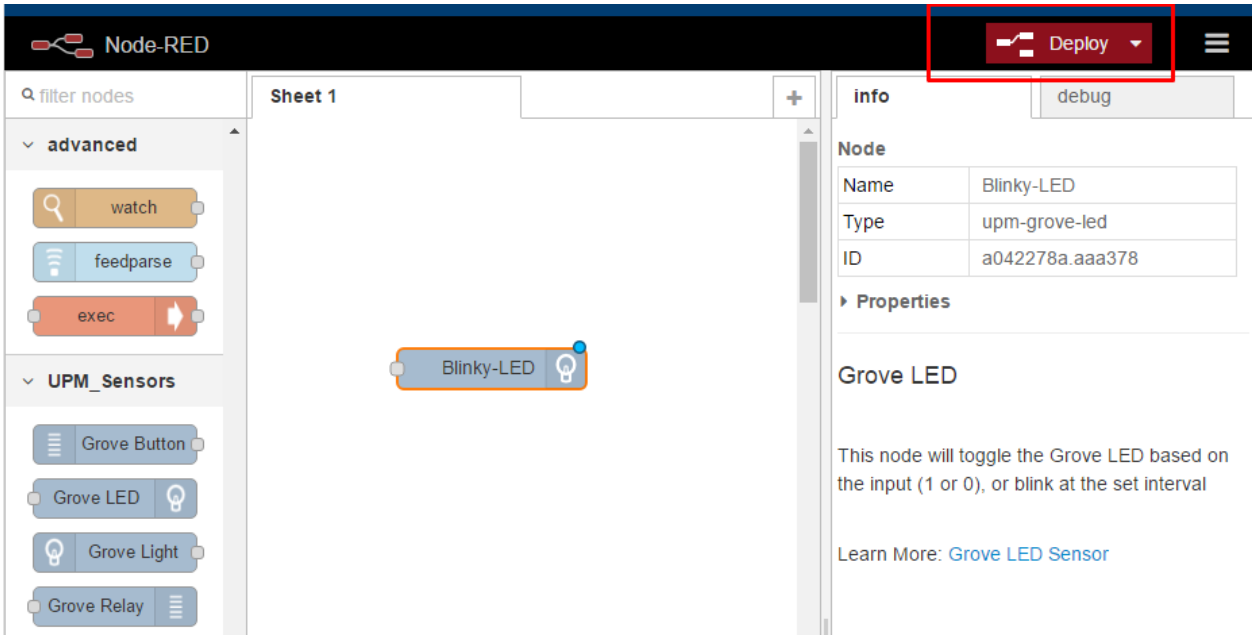


- Double Click on Grove LED node on visual Editor. It will open a dialog box to edit properties of node.
  - ➢ Name – Blinky LED
  - ➢ Platform – Fermata
  - ➢ Pin – D13
  - ➢ Mode- Blink
  - ➢ Interval(ms) – 1000

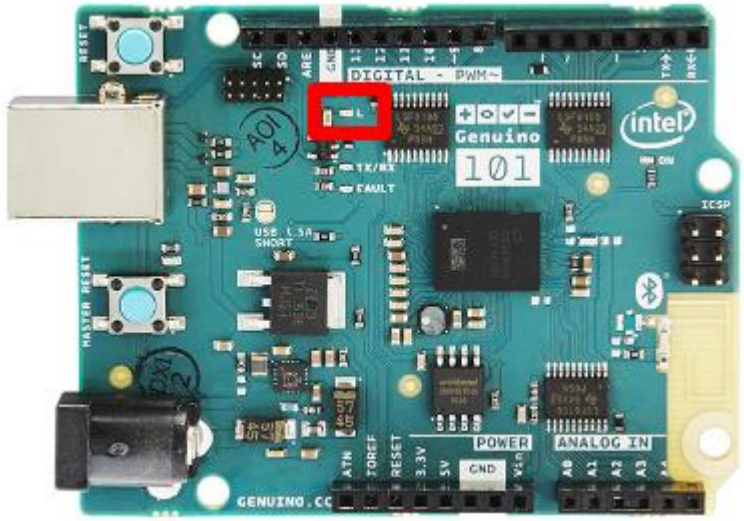
  Then Click OK to save changes.



- Click the Deploy button on the top of menu bar to deploy the Node-RED flow.

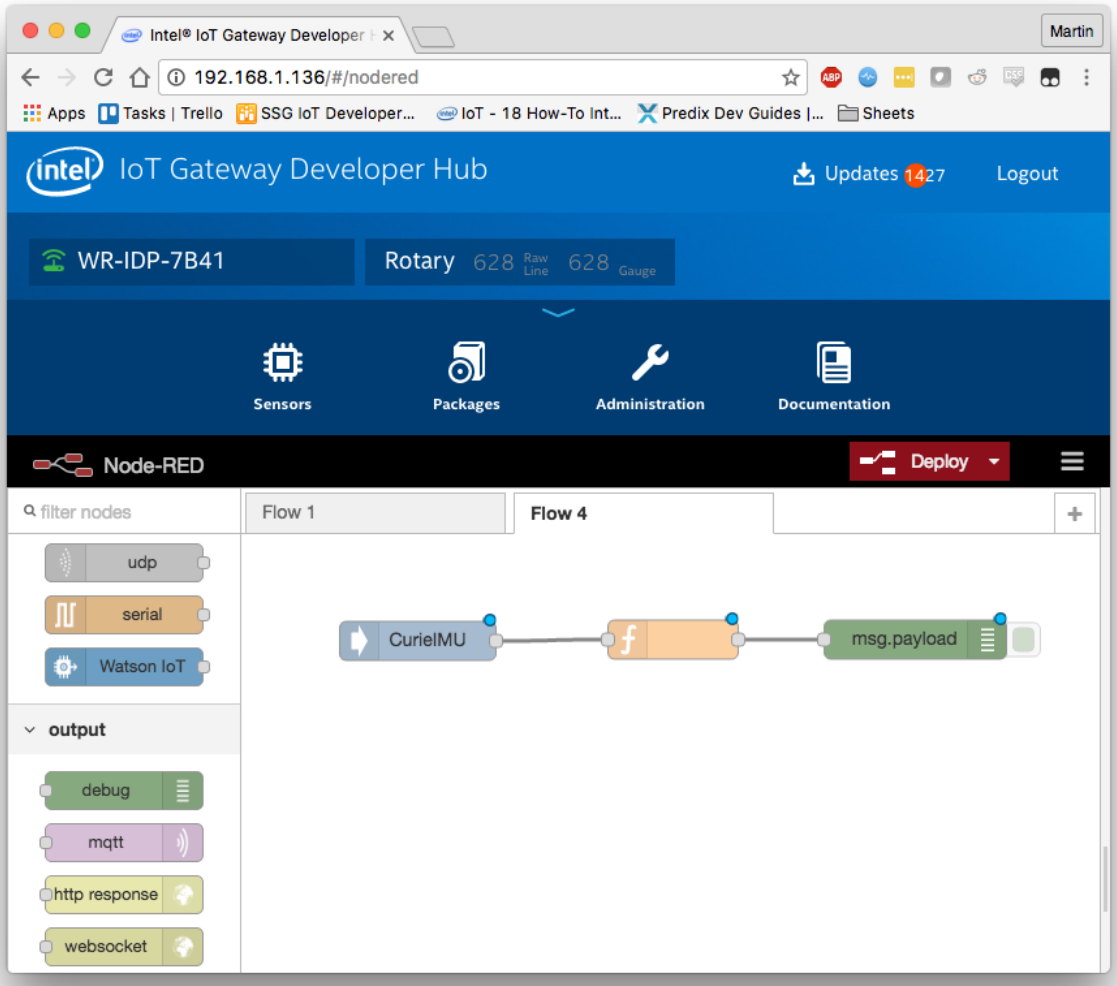


- The Arduino 101 LED on board LED will start blinking

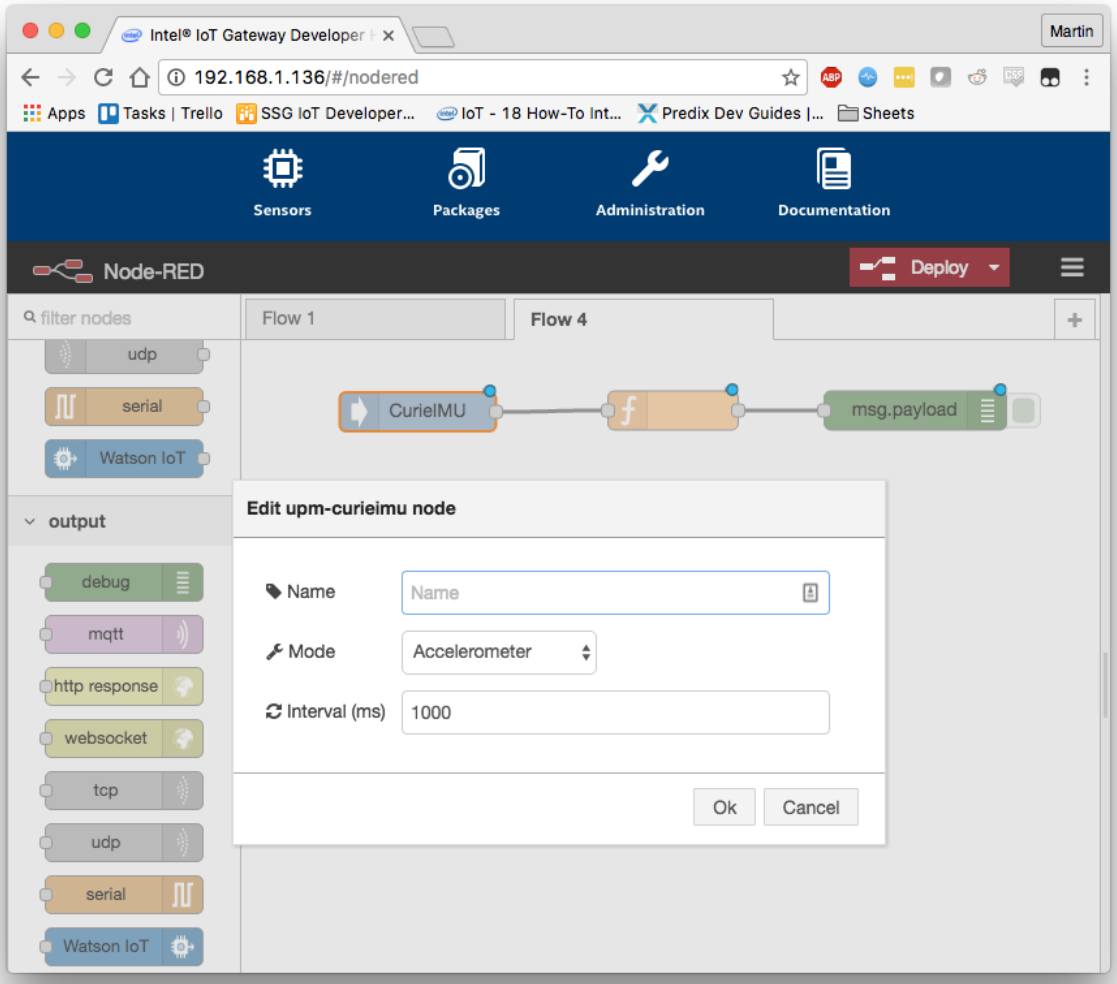


Adding Sensors and Processing to the Gateway

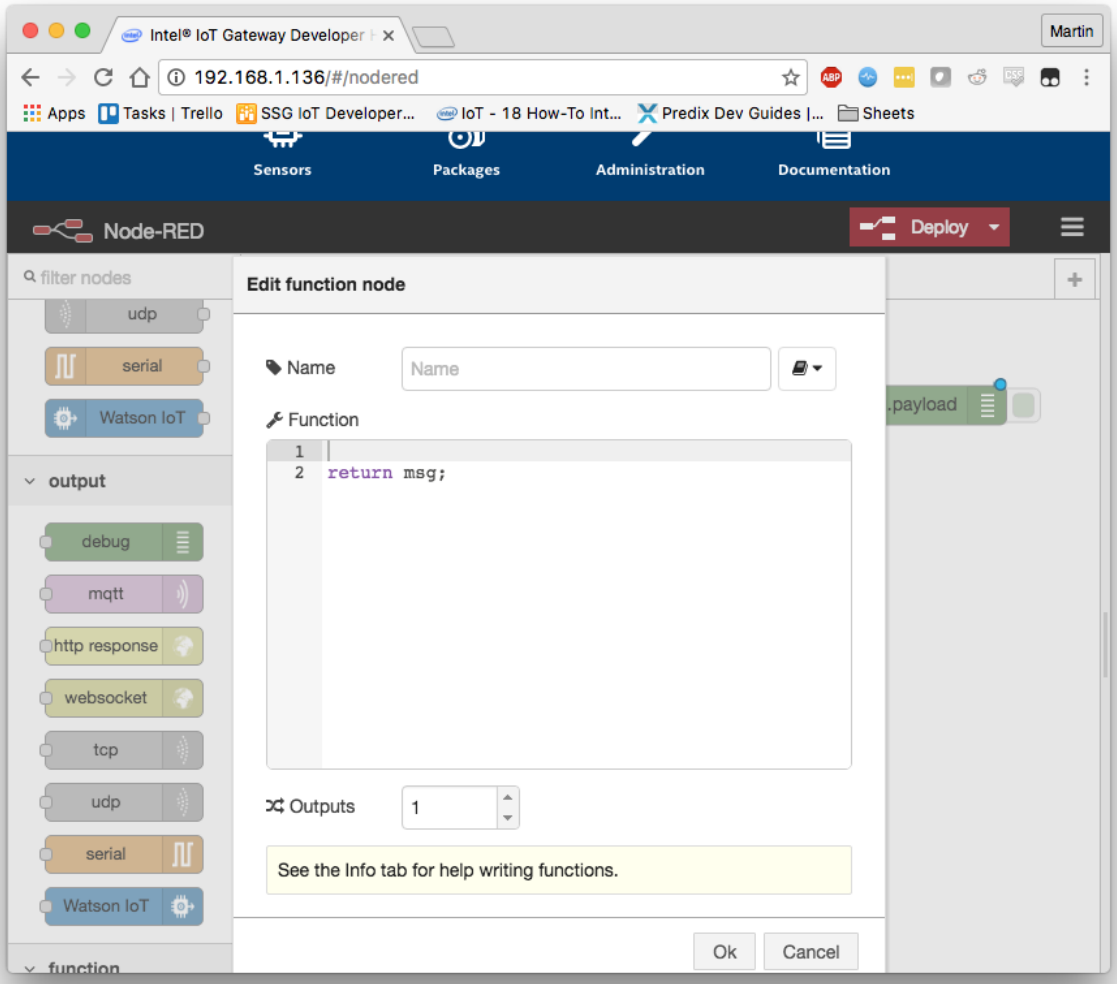
Now that you have a basic understanding of how to interact with the Gateway let's move on to adding some more sensor data and processing to the gateway.



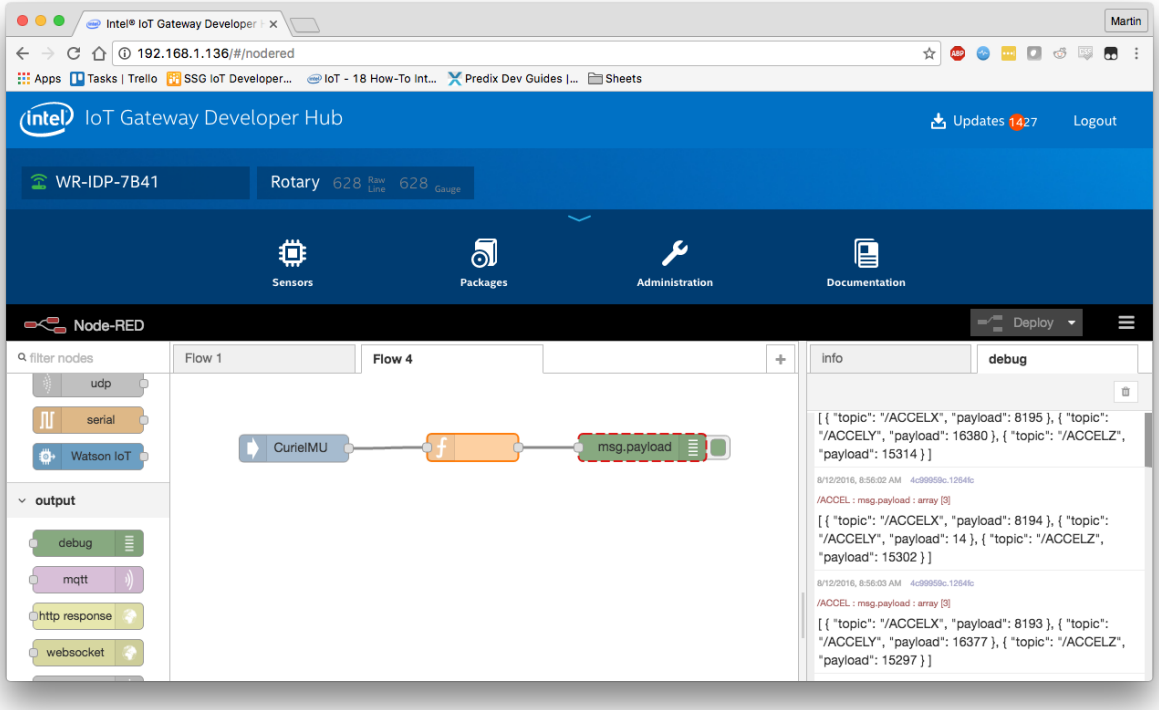
- From **UPM Sensors** add **CurieIMU** node
- From **function** add a **function** node
- From **output** add a **debug** node
- Wire the nodes together



If you double click the **CurieIMU** node you can bring up the configuration interface. Here you can name the node, change the mode from Accelerometer to Gyroscope and set the intervals at which the node will send data. Leave these as defaults for now.



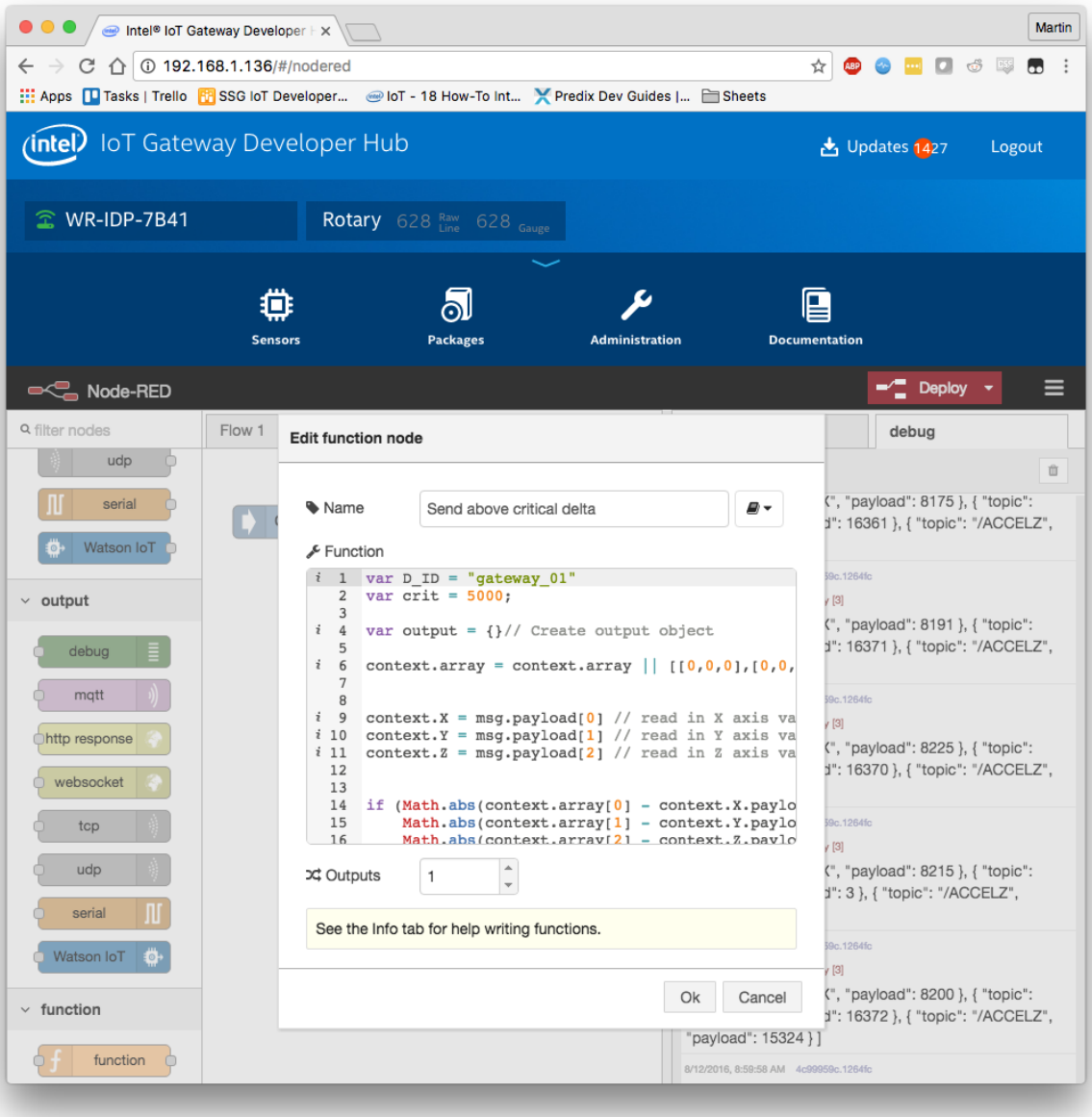
If you double click the function node you will bring up the function interface. Here you can enter JavaScript code that will be executed when the node gets input. By default, it only has “return msg;” which will simply pass the incoming message to the output. Let’s leave this for now.



Hit deploy to run the flow. Open the debug tab to see the output from the function node.
Note: you may have to drag from the right to open the debug tab.

As you can see the CurieIMU reads out the accelerometer data once a second and passes it to the debug output.

If we are sending data up to a cloud service, we probably want to limit it to significant data. Let's add a filter to the function node so that it only passes the accelerometer data to the debug tab if the values between reading exceed some critical value – i.e. the accelerometer is bumped.



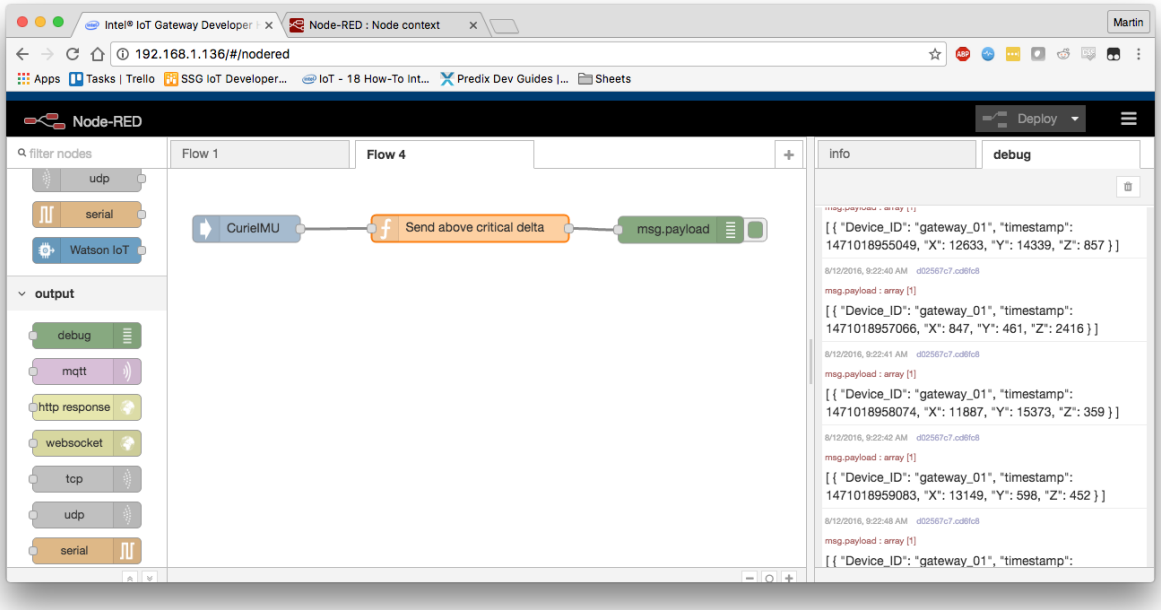
Open the function node again and enter / paste the following code:

```
var D_ID = "gateway_01";
var crit = 5000;

var output = {};
context.array = context.array || [0,0,0];

context.X = msg.payload[0];
context.Y = msg.payload[1];
context.Z = msg.payload[2];

if (Math.abs(context.array[0] - context.X.payload) > crit |
    Math.abs(context.array[1] - context.Y.payload) > crit |
    Math.abs(context.array[2] - context.Z.payload) > crit)
    {
        output.payload = [
            {
                d: {
                    "Device_ID": D_ID,
                    "timestamp" : Date.now(),
                    "X" : context.X.payload,
                    "Y" : context.Y.payload,
                    "Z" : context.Z.payload
                }
            }
        ];
        context.array = [context.X.payload, context.Y.payload, contex
t.Z.payload];
        return output;
    }
    else {
```

```
        context.array = [context.X.payload, context.Y.payload, contex
t.Z.payload];

    }
```

Note: The context object is specific to Node-RED. Any variables inside the node are lost every time the code is run (whenever the node gets a message input). In order to preserve the variables across iterations the context object is used. There are also contexts for the entire flow and even across flows. Think of them as global variables. For more info see: http://nodered.org/docs/creating-nodes/context

The code does a few things. First it parses the incoming message into X, Y, and Z accelerometer values. Then it compares the incoming values against the last measured values which are stored in an array ([0,0,0] at the start). If the delta between values is above the critical value, 5000 in our case, it sends the accelerometer data to the output. The code then overwrites the array with the incoming values to prepare for the next iteration. The code also adds a Device Id and timestamp to the output.



Hit deploy to run the code. The debug output should only display the accelerometer data when the Arduino 101 board registers movement.

Share Node-RED flows across workspaces

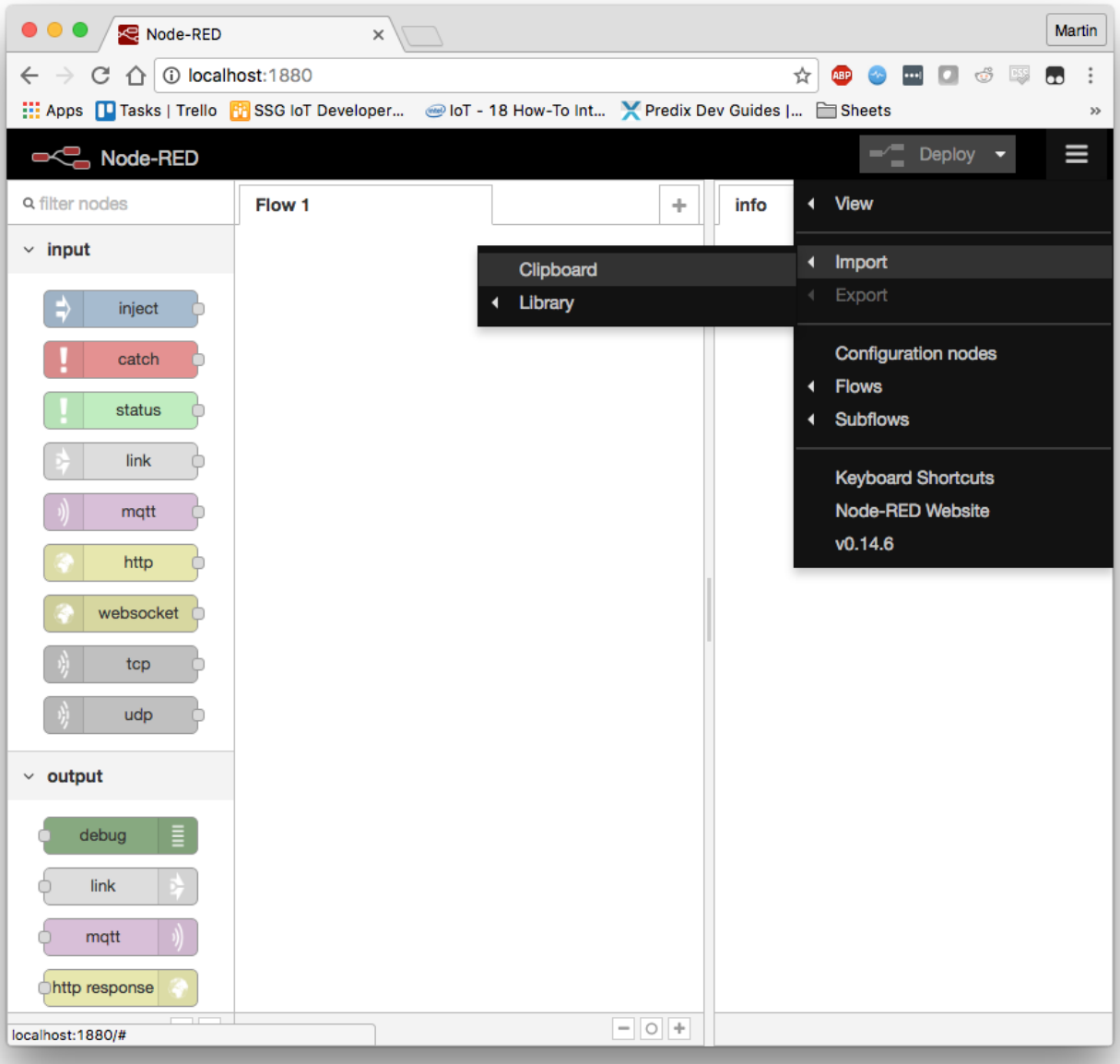
You can export and import entire flows (nodes and connections) out of and into your workspace.

Importing Flows:

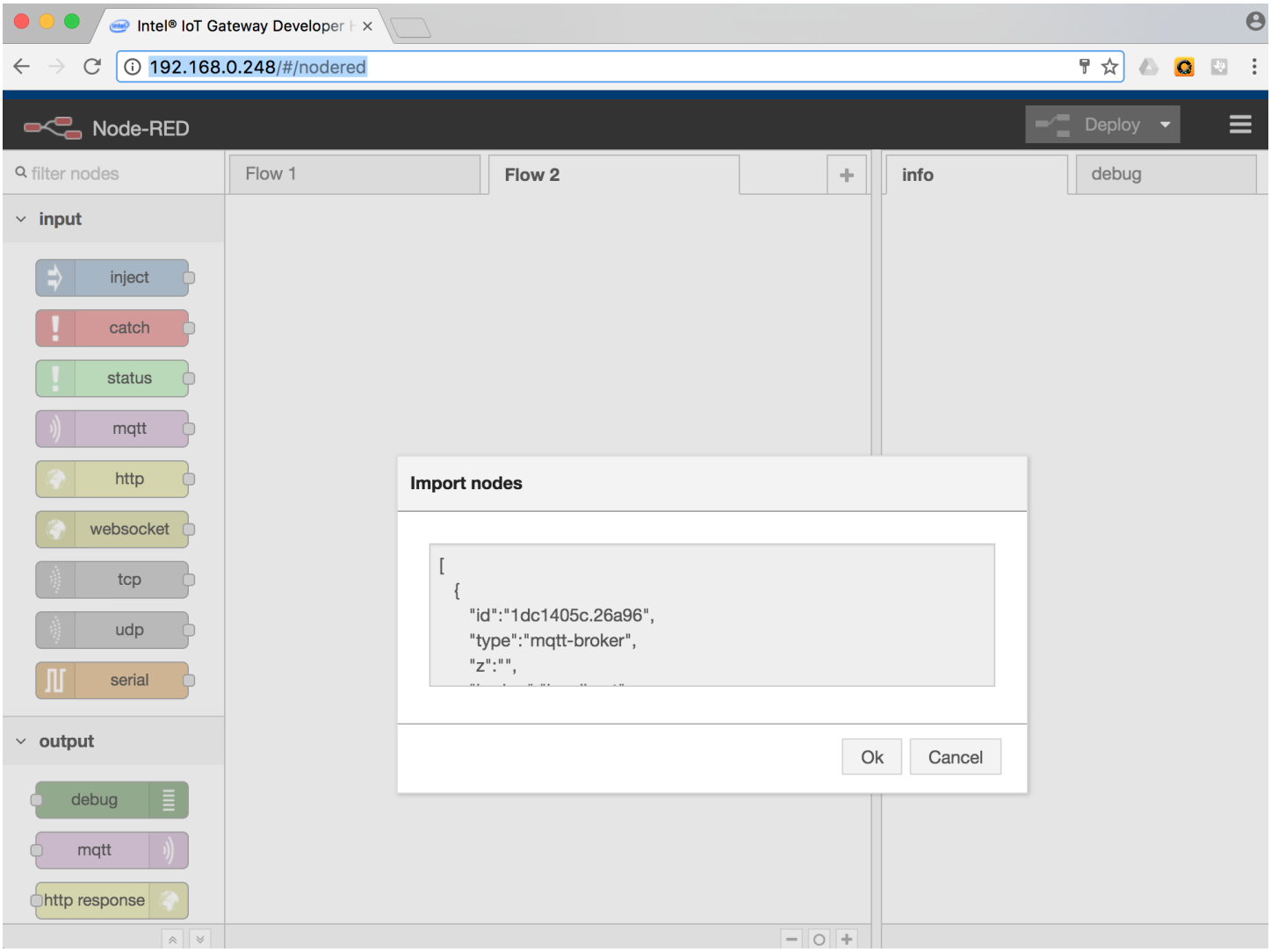
To import the above flow, go to:

https://github.com/martinkronberg/Intel-IBM-IoT-Workshop/blob/master/Default-Flow.json
Copy the JSON code to your clipboard (ctrl-c)

Navigate to the Import -> Clipboard from the upper right menu:



Paste the JSON from your clipboard (ctrl-v) and click OK:



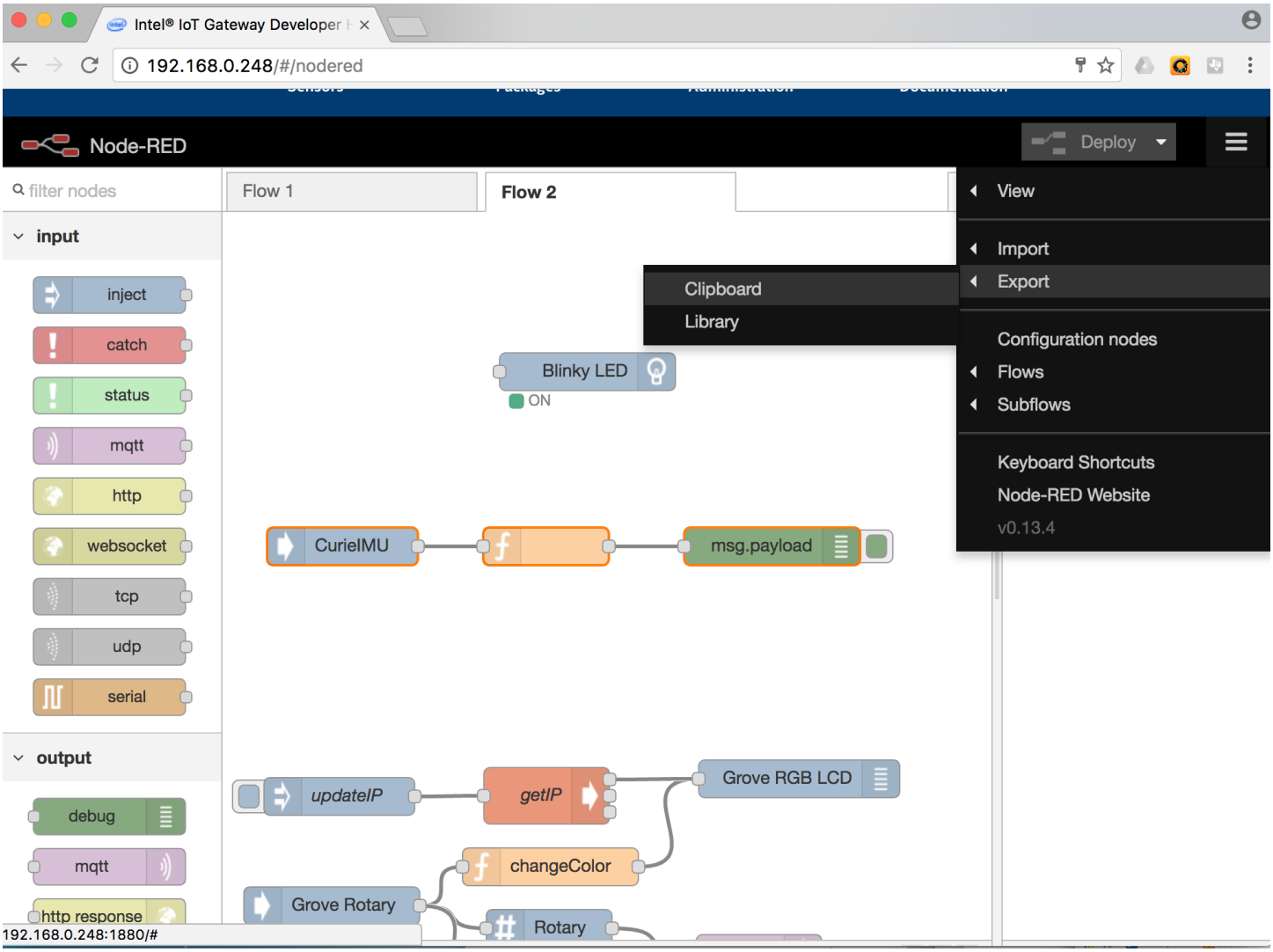
The imported flow can now be positioned using your mouse pointer. Click to place on the sheet

You will now have the entire flow inside your workspace, but remember you need to hit deploy to make the code live.

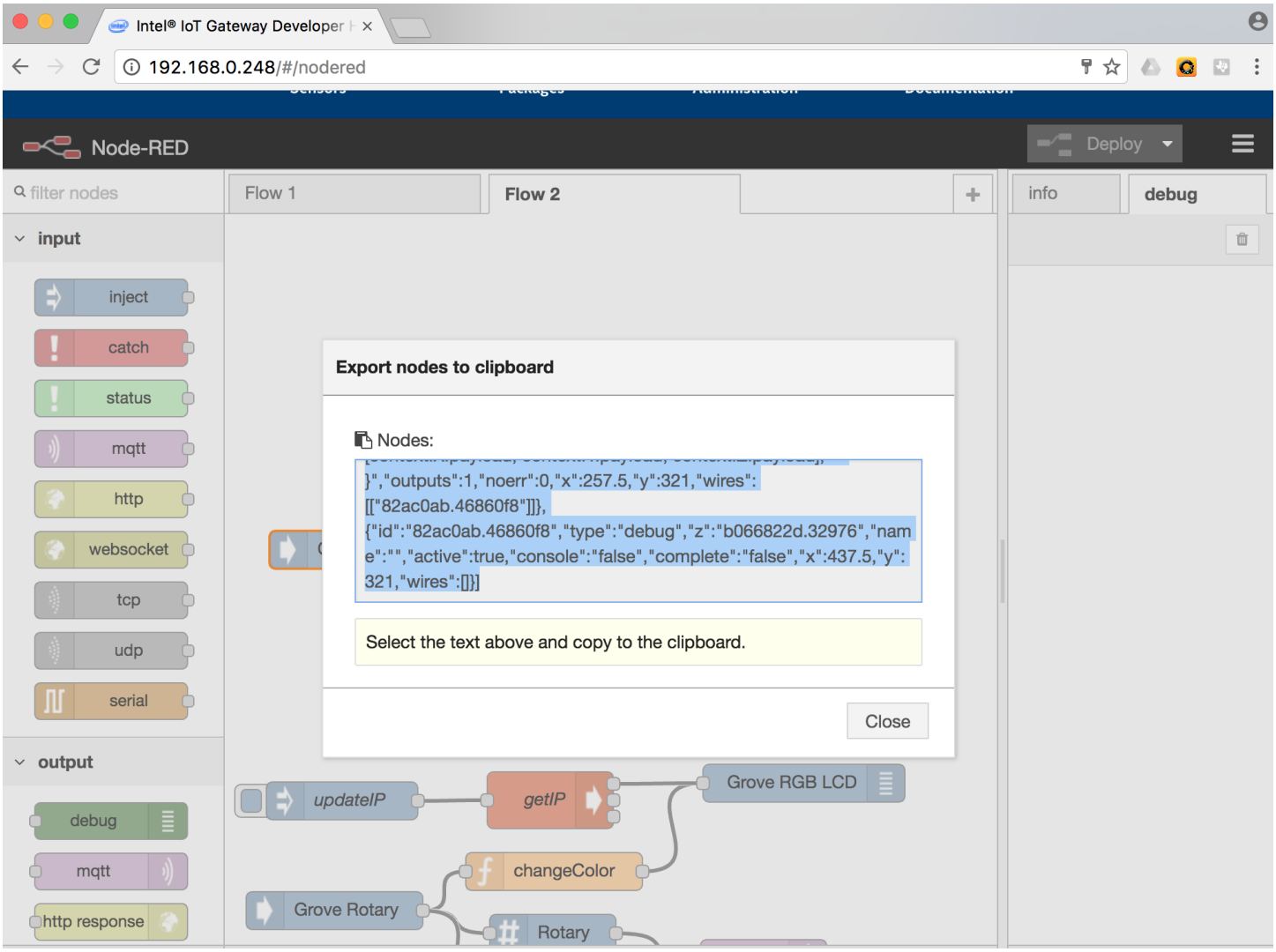
Follow the same procedure for importing anymore code you come across in this workshop.

Exporting Flows

To export a flow, select the nodes you wish to export. When selected, the border will turn orange. Once selected, navigate to Export -> Clipboard from the menu

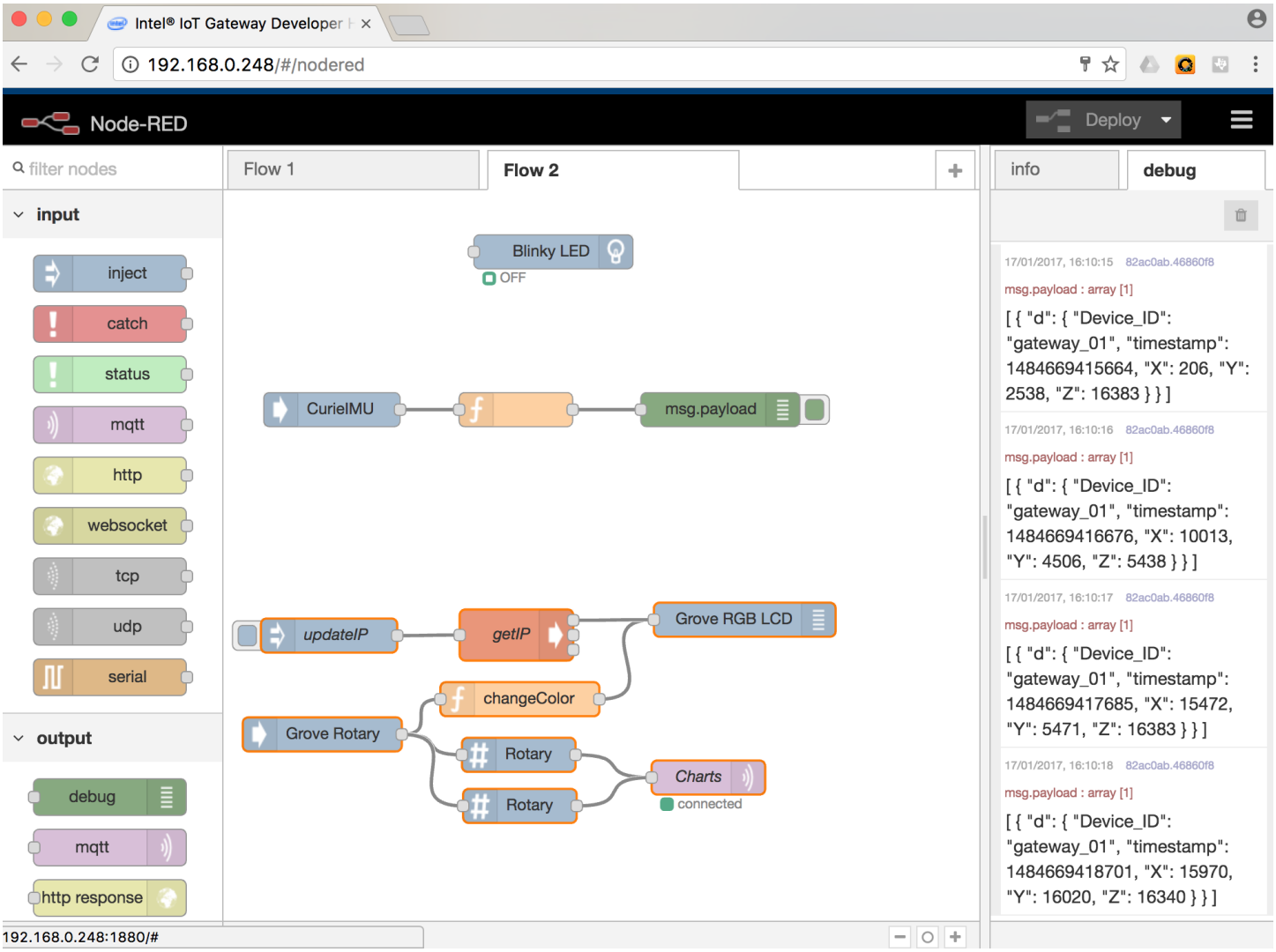


Copy the text in the window that opens. You can now paste this somewhere and share it with others. I would recommend running it through a JSON beautifier like: https://jsonformatter.curiousconcept.com/ first if you want it to be readable.



Deleting Nodes

Nodes can be deleted from the sheet by selecting them, then hitting the delete or backspace key on your keyboard. Delete the flow you imported, which is the same as the default flow on Flow 1 sheet.


Intel® IoT Gateway Developer
192.168.0.248/#/nodered
Node-RED
Deploy
filter nodes
Flow 1
Flow 2
info
debug
input
inject
catch
status
mqtt
http
websocket
tcp
udp
serial
output
debug
mqtt
http response
Blinky LED
OFF
CurieIMU
msg.payload
updateIP
getIP
Grove RGB LCD
changeColor
Grove Rotary
Rotary
Rotary
Charts
connected
17/01/2017, 16:10:15 82ac0ab.46860f8
msg.payload : array [1]
[{ "d": { "Device_ID": "gateway_01", "timestamp": 1484669415664, "X": 206, "Y": 2538, "Z": 16383 } }]
17/01/2017, 16:10:16 82ac0ab.46860f8
msg.payload : array [1]
[{ "d": { "Device_ID": "gateway_01", "timestamp": 1484669416676, "X": 10013, "Y": 4506, "Z": 5438 } }]
17/01/2017, 16:10:17 82ac0ab.46860f8
msg.payload : array [1]
[{ "d": { "Device_ID": "gateway_01", "timestamp": 1484669417685, "X": 15472, "Y": 5471, "Z": 16383 } }]
17/01/2017, 16:10:18 82ac0ab.46860f8
msg.payload : array [1]
[{ "d": { "Device_ID": "gateway_01", "timestamp": 1484669418701, "X": 15970, "Y": 16020, "Z": 16340 } }]
192.168.0.248:1880/#

Add Watson IoT Nodes and OS Monitoring

Next we will add some custom nodes in order to connect to Watson IoT as well as monitor memory load.

Open a command line interface to the Gateway. You can do this using the SSH interface of your choice or with the built-in command line on the Gateway. **Note:** The built in command line interface is functional, however not recommended for extended use as it is missing many useful features.

SSH into the IP address displayed on the LCD as **root**. The password is **root**.

Once you are connected to the command line run the following command to install the IBM Watson IoT nodes:

```
npm install -g node-red-contrib-ibm-watson-iot
```

Also install the Node-RED OS monitoring nodes with:

```
npm install node-red-contrib-os -g
```

The install process will throw some warnings; you can ignore these.

Finally restart Node-RED for the changes to take effect:

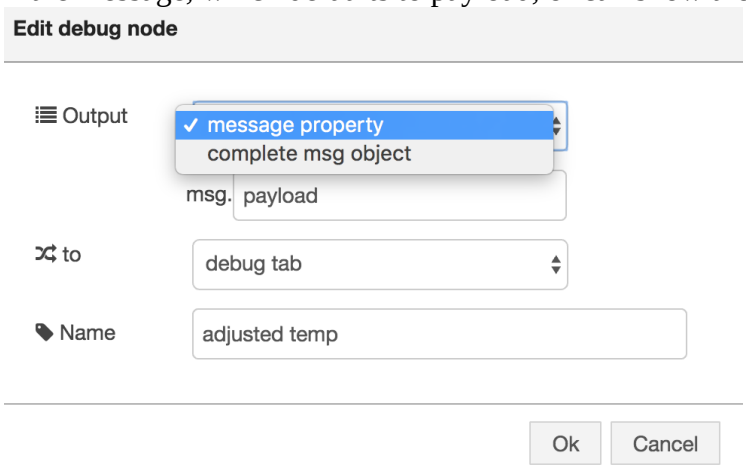
```
systemctl restart node-red-experience
```

Debugging techniques

When working on Node-RED sometimes you want to find out what is going on in a flow and sometimes you may want to simulate events to help test code without having to get a sensor to fire a live event. The Debug and Inject nodes are your friend here. You can use these techniques through the workshop if you encounter errors.

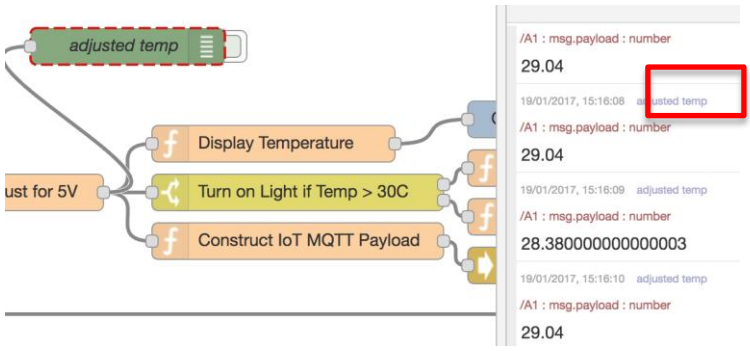
Debug node

The Debug node allows you to see what data is flowing in a node. The debug node can be configured to show a specific property in the message, which defaults to payload, or can show the entire message content.

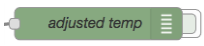


You can also name a node. When messages are shown in the debug panel if the node is named then the name will show against the message, so you can see which node is outputting the message to the debug panel

You can also click or hover over the name in the debug panel and the node that created the content will be highlighted on the sheet



The button at the right end of the node can be used to enable 

and disable 

the debug node.

Update alert

Recently a new version of Node-RED was released. The new release is available on IBM Cloud, but not yet in the NUC image. The behavior of the debug panel changed in the new release. To help display large objects in the debug panel the objects are now collapsed and a twisty is shown to allow you to expand the object as required.

Notice that the node name is no longer shown in the debug panel, but clicking or hovering over the Node ID will highlight the node that generated the entry.

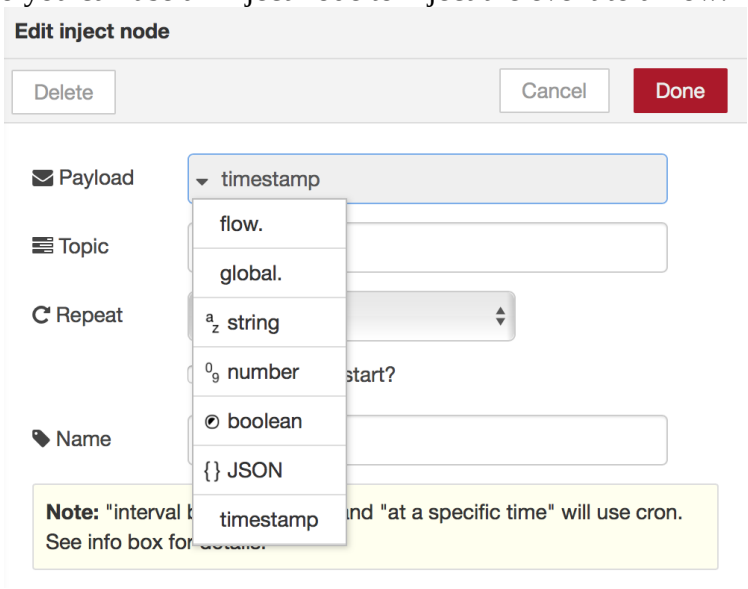
In a function node you can use the logging functions to add messages to the console and debug panel. See http://nodered.org/docs/writing-functions - logging-events for more detail.

```
node.log("Debug entry");
node.warn("Warning entry");
node.error("Error entry");
```

Log entries only go to the console (do not show in the debug panel). Warning and errors go to the console and debug panel. Warning entries have a yellow band at the side and errors have a red band.

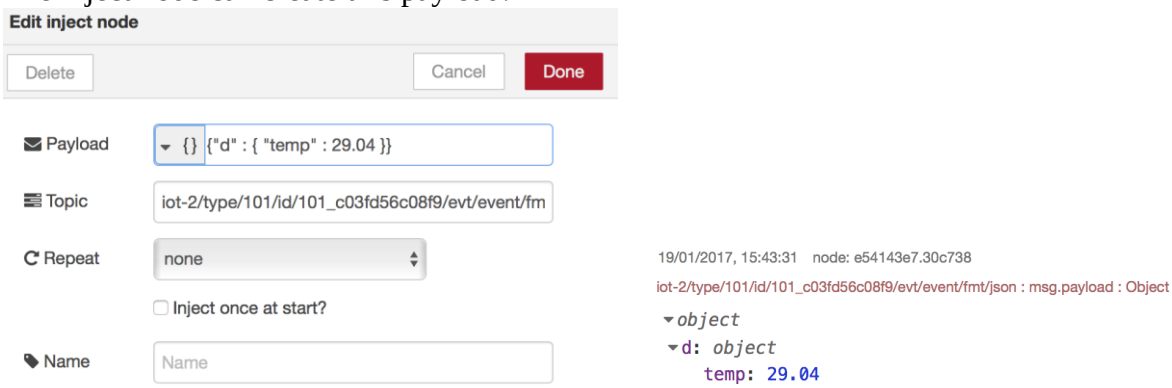
Inject node

The inject node allow you to manually inject data into a flow. It is often difficult to get a real device to generate certain events, so you can use an inject node to inject the event to a flow.



So if a device is sending data into the platform:

The inject node can create this payload:

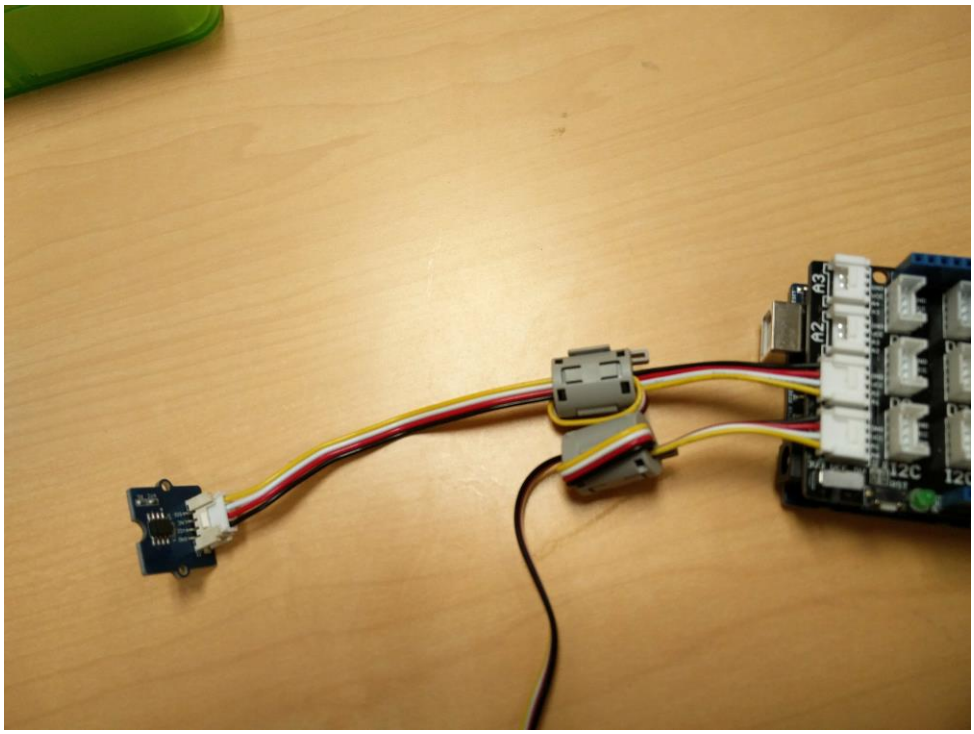


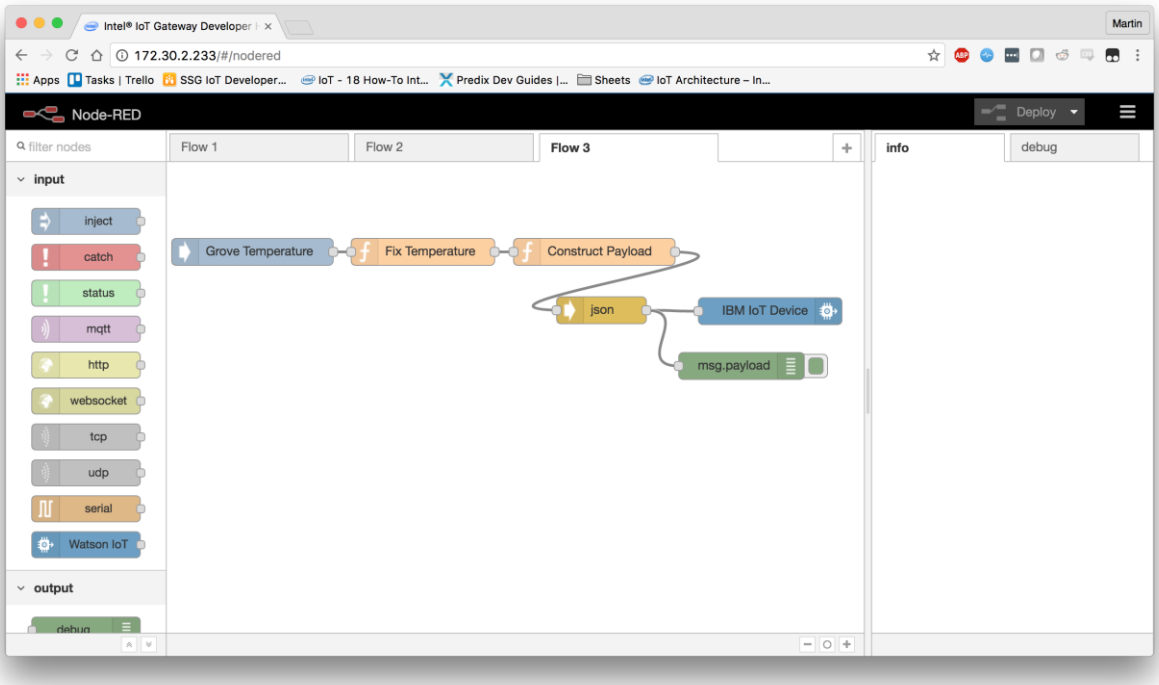
A function node can be used to add missing properties if they are outside the payload property.

The button on the left end of the node can be used to manually inject a message to the flow, or the node can be configured to automatically send messages.

Create Temperature Flow

We will now create a flow to measure the local temperature and send the data to IBM Watson. First Plug in a Grove Temperature sensor into A1 using a grove cable:





The above flow is available at http://bit.ly/2bONsqO

Once the Gateway reboots go back to your Node-RED, create a new flow and add the following nodes:

- From **UPM_Sensors** add a **Grove Temperature Sensor** node
- Double click the **Grove Temperature Sensor** and change the **platform** to **firmata** and the pin to **A1**, you can select if you prefer Fahrenheit or Celsius
- From **function** add 2 **function** nodes
- From **function** add a **JSON** node
- From **output** add a **Watson IoT** node
- From **output** add a **debug** node
- Wire the nodes as shown above

In the “Fix Temperature” function add:

msg.payload = msg.payload * 3.3 / 5;
return msg;

This will allow the correct temperature to be read at a 5V board voltage (the sensor functions at 3.3 by default)

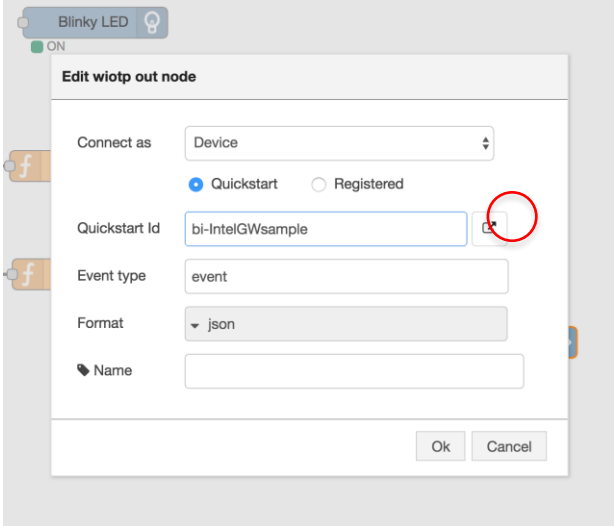
In the “Construct Payload” function add:

```
return {
    payload: {
            d: {
                temp: msg.payload
            }
    }
};
```

This will format the temperature to allow Watson IoT to ingest the data easier.

Open the configuration for the Watson IoT node and configure it:

- Connect as : Device and Quickstart
- Quickstart Id : enter a unique ID for yourself



Once you have configured the IBM IoT Device nodes your flow will send local temperature data to IBM Watson IoT cloud services. Use the link in the configuration dialog to launch a browser where you will see your data.

Remember to deploy the flow to make it live.

What Next?

To explore some advanced projects that you can build with the gateway make sure you have the Intel XDK IoT Edition up and running and check out:

Smart Home Path to Product:

https://software.intel.com/en-us/articles/iot-path-to-product-how-to-build-the-smart-home-prototype

Smart Vending Machine Path to Product:

https://software.intel.com/en-us/articles/iot-path-to-product-how-to-build-an-intelligent-vending-machine

Connected Transportation Path to Product:

https://software.intel.com/en-us/articles/iot-path-to-product-how-to-build-a-connected-transportation-solution

A collection of How-To IoT Projects:

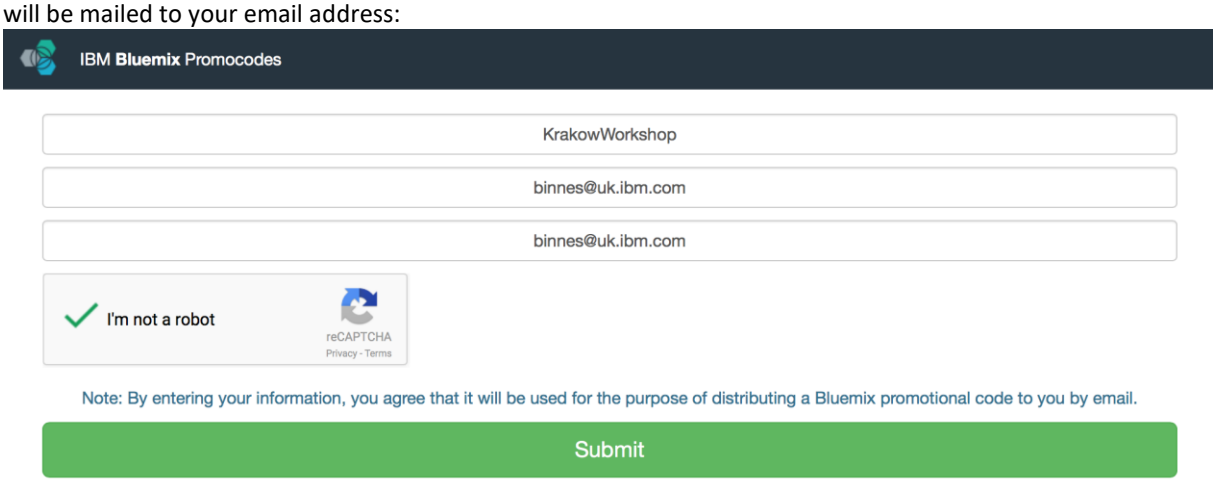
https://software.intel.com/en-us/blogs/2015/11/04/announcing-18-new-how-to-intel-iot-code-samples

Please wait for IBM to finish their presentation before moving on!

Creating an IoT application in IBM Cloud

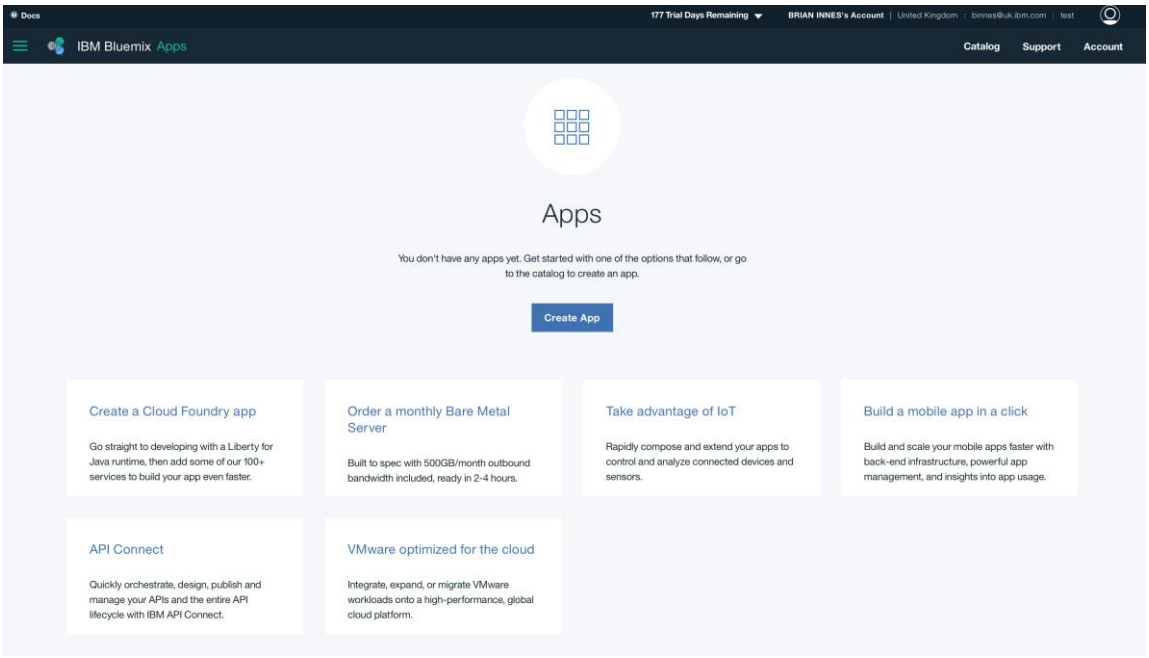
In this section, we will create an Internet of Things boilerplate application in IBM Cloud and use it to receive temperature values from a simulated IoT temperature sensor.

1. Ensure you have an active IBM Cloud account. For this workshop we are making a promo code available to give you additional cloud resources. Go to the site http://promocodes2.mybluemix.net and enter **KrakowWorkshop** for the event name. Provide a valid email address that you can access, as the promo code will be mailed to your email address:

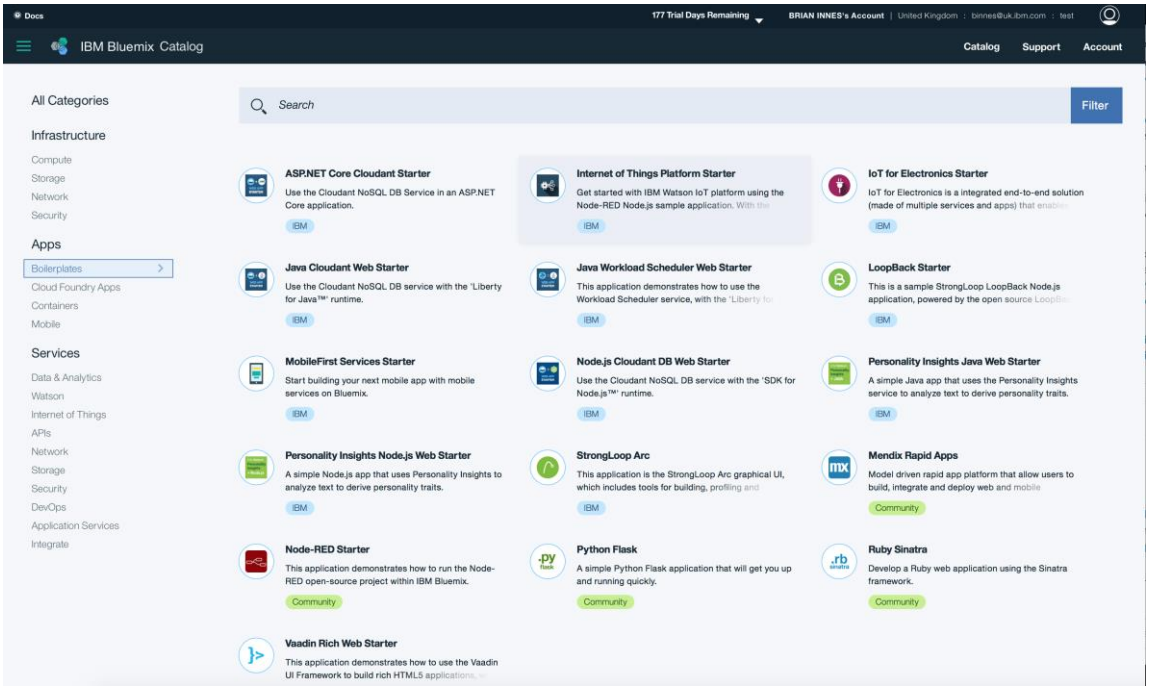


Follow the instructions in the mail to apply the promo code to your account.

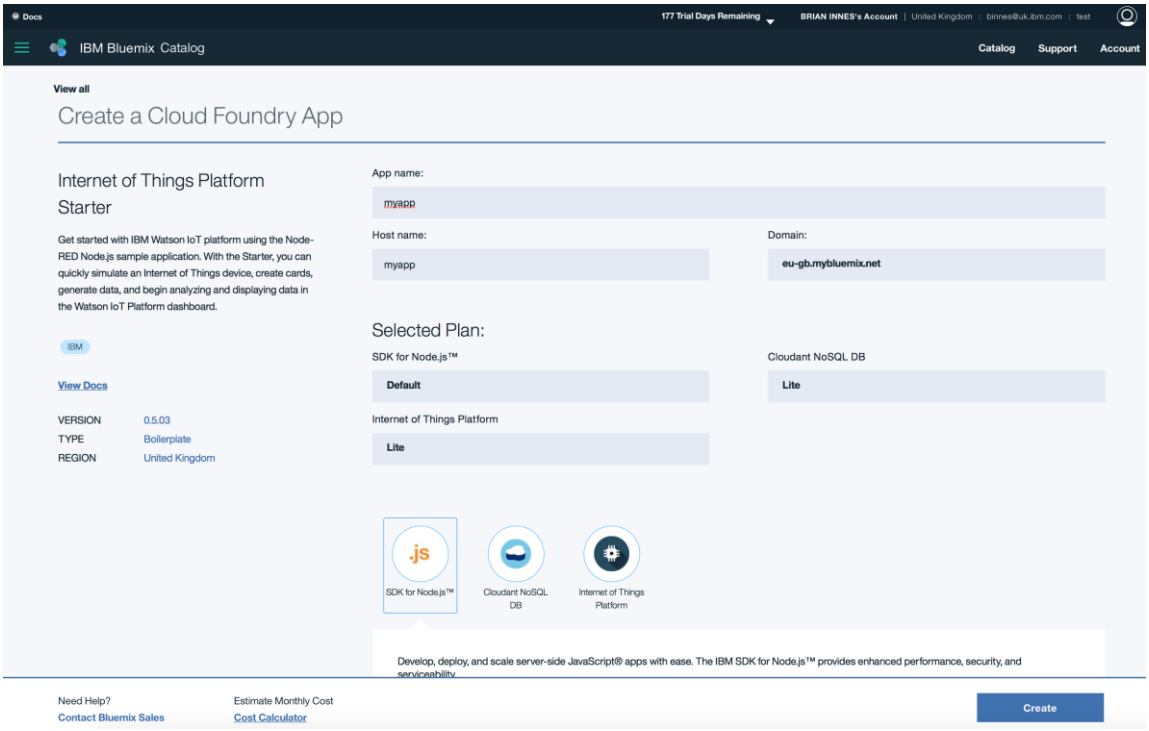
2. Open a web browser and go to your IBM Cloud dashboard, http://bluemix.net. This is the IBM Cloud dashboard where you have access to creating Cloud Foundry apps, IBM Containers, Virtual Machines, and a variety of Services. We are going to create a Cloud Foundry application today. To access the catalog, click on the **All Items tab** and then the blue **+** hexagon on the top right.



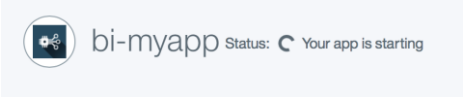
3. In the Catalog, IBM Cloud offers a handful of boilerplate applications that you can create and get started quickly. The Internet of Things Starter boilerplate includes Node-RED, a Cloudant NoSQL database, and the SDK for Node.js. Under Boilerplates, select the **Internet of Things Platform Starter** boilerplate tile.



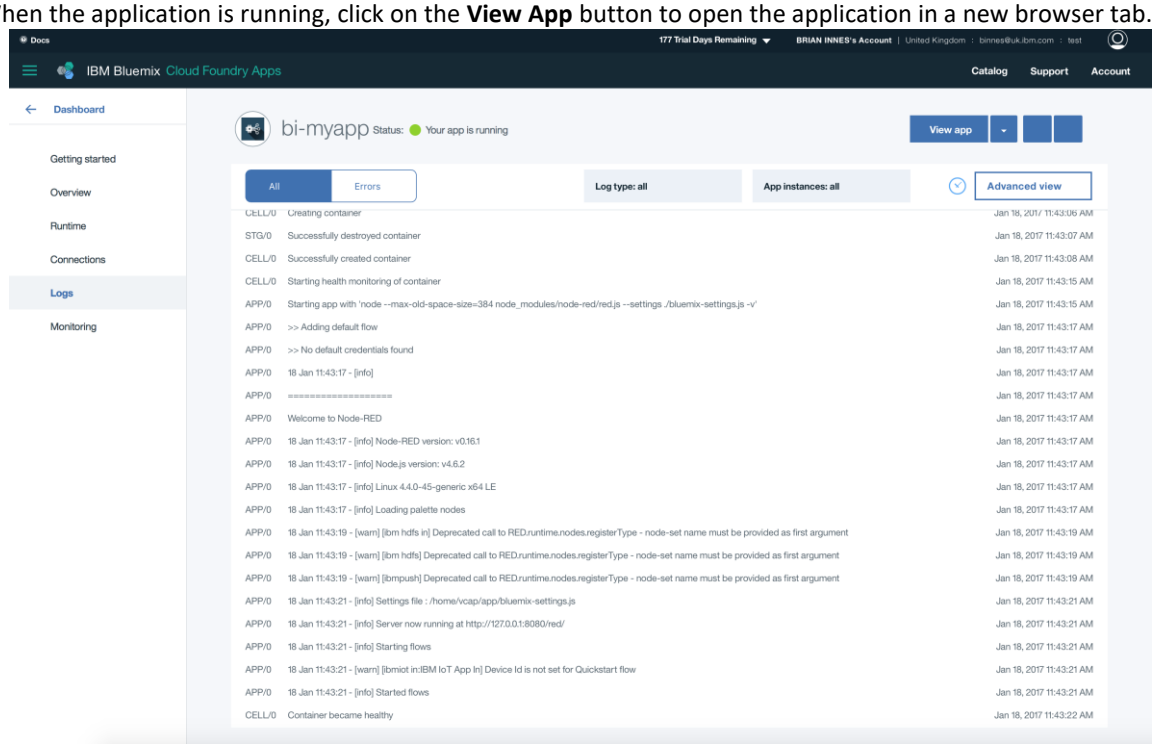
- Pick a unique name for your application. Your application will be located at the URL created by joining the app name (Host name) and the Domain, which will default to **mybluemix.net** if using the US South region, **eu-de.mybluemix.net** if using the Germany region or **eu-gb.mybluemix.net** if using the United Kingdom region. There can only be a single app using a URL, so ensure your app name creates a unique URL for your application. You can try adding your initials in front of the host if the host you choose is already taken by someone else. Click on **Create** to create the application instance.



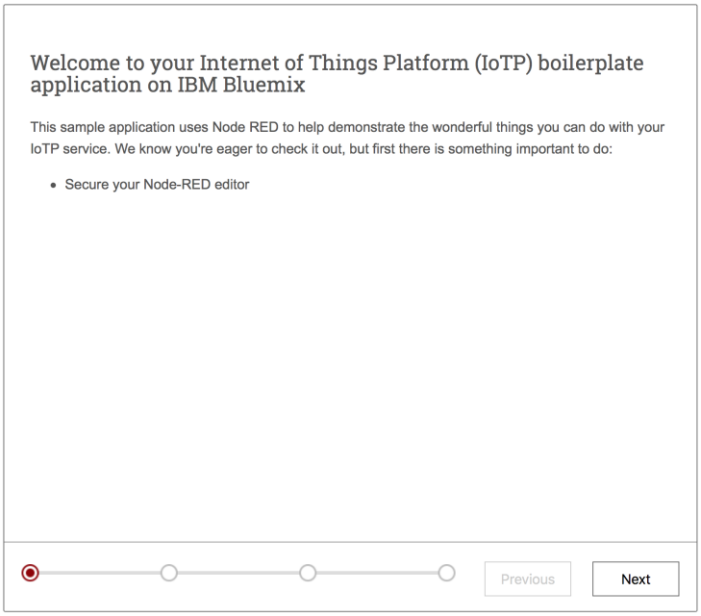
4. IBM Cloud will create an application in your account based on the services in the boilerplate. This is called staging an application. It can take a few minutes for this process to complete. While you wait, you can click on the **Logs** tab and see activity logs from the platform and Node.js runtime.



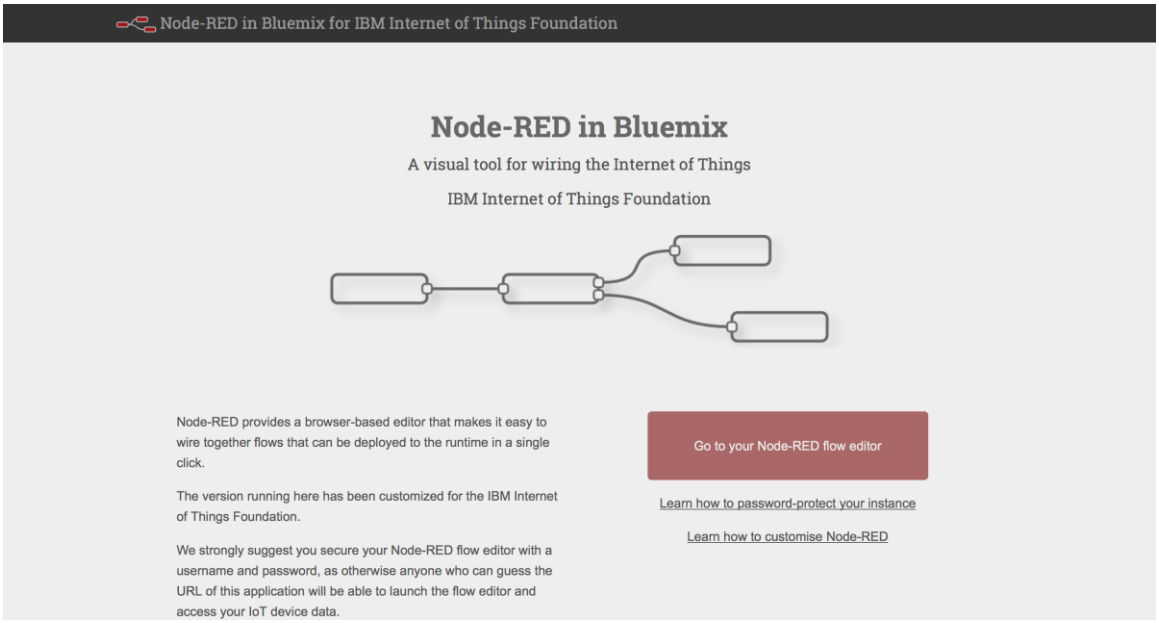
5. When the application is running, click on the **View App** button to open the application in a new browser tab.



6. The first time you access your cloud based Node-RED environment you are required to enter a username and password to protect the editor, as this is running on the open internet. Work through the wizard using the next button to create your credentials – PLEASE DO NOT FORGET THE DETAILS, as you will need them to access the Node-RED editor to complete the rest of the workshop:

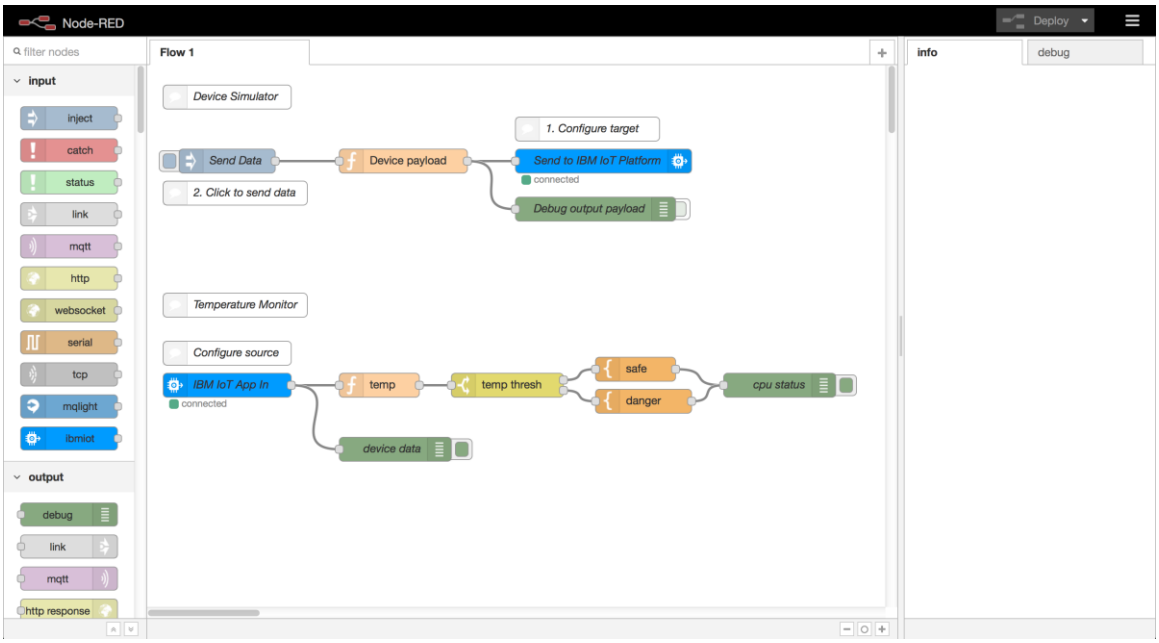


7. When you have created the credentials, this is the Node-RED start page. Node-RED is an open-sourced Node.js application that provides a visual editor that makes it easy to wire together flows. Click on the red button **Go to your Node-RED flow editor** to launch the editor. You should be presented with a username and password entry dialog. Enter the details you created in the wizard.

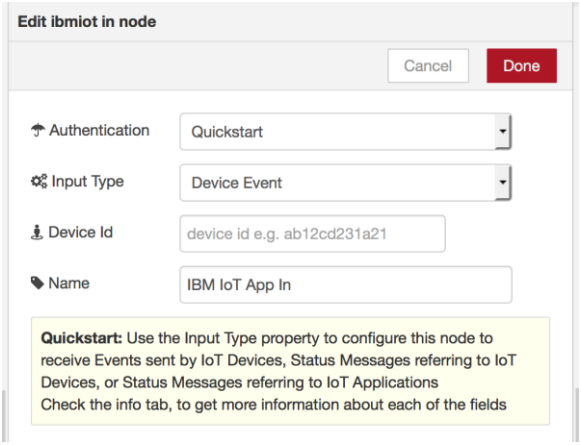


8. On the left side is a palette of nodes. Each node performs a defined function, configurable by dialog windows or by modifying the msg input. You can click on a node in the palette and find out what it does in the info tab on the right side of the screen. Drag and drop nodes and connect them together in the middle pane, called a canvas. Double click on nodes in the canvas to customize settings used by the node. Connect two or more nodes via the grey knobs on the left and right sides of the node, constructing what is called a flow. A flow is executed from left to right and is completed when the last node is reached. A flow can split off into two or more flows, for example, with a switch node. Two flows can also share a node or a flow, reusing the same logic in multiple scenarios by connecting their grey output knobs to the shared node's left grey input knob.

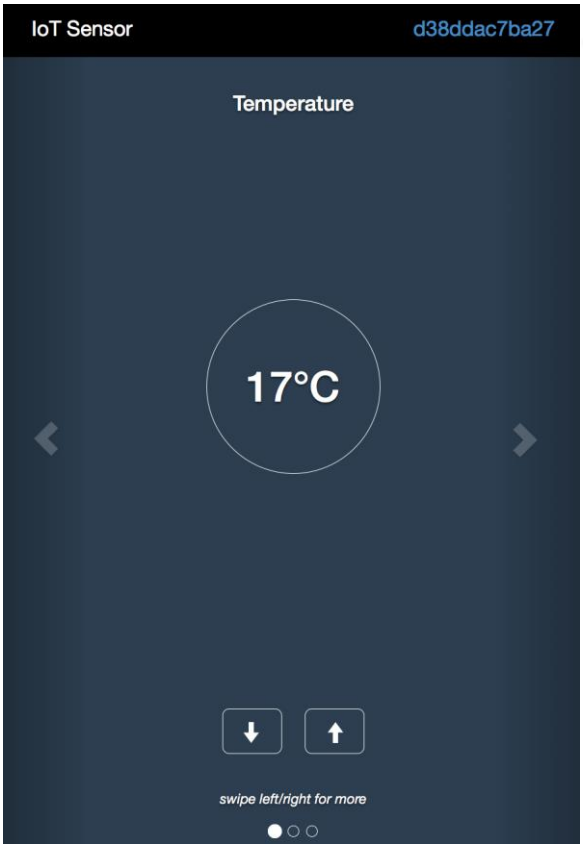
   Let's begin by configuring the Watson IoT App In node. Double click on the Watson IoT App In node.



   Nodes can be customized in different ways depending on what they do. The IBM IoT node begins a flow when a message is received from a specific device via the Watson IoT Platform. We need a Device Id of a device that is connected to the IoT Platform. We could use a real device, or we can simulate devices.

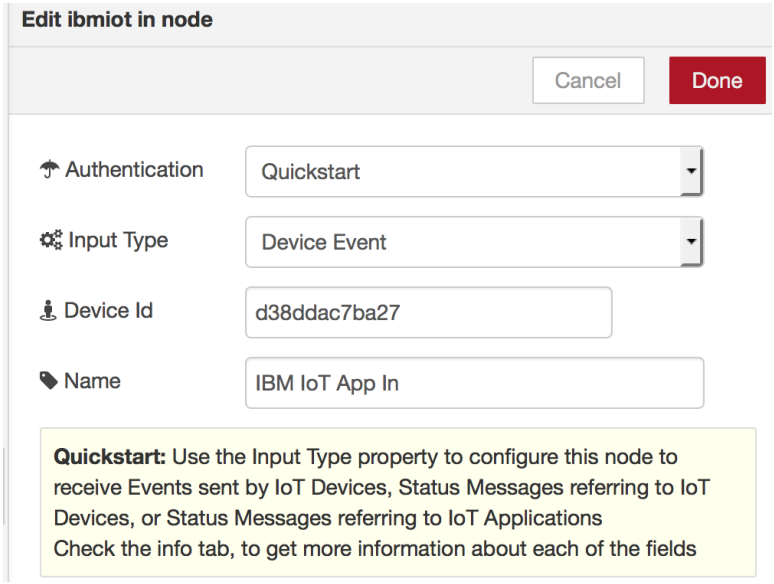


9. To get a device ID from a simulated device, visit http://ibm.biz/iotsensor. In the upper right is an alphanumeric value. This simulated temperature sensor sends the temperature to the Watson Internet of Things Platform service using this device ID.



Write down this Device ID

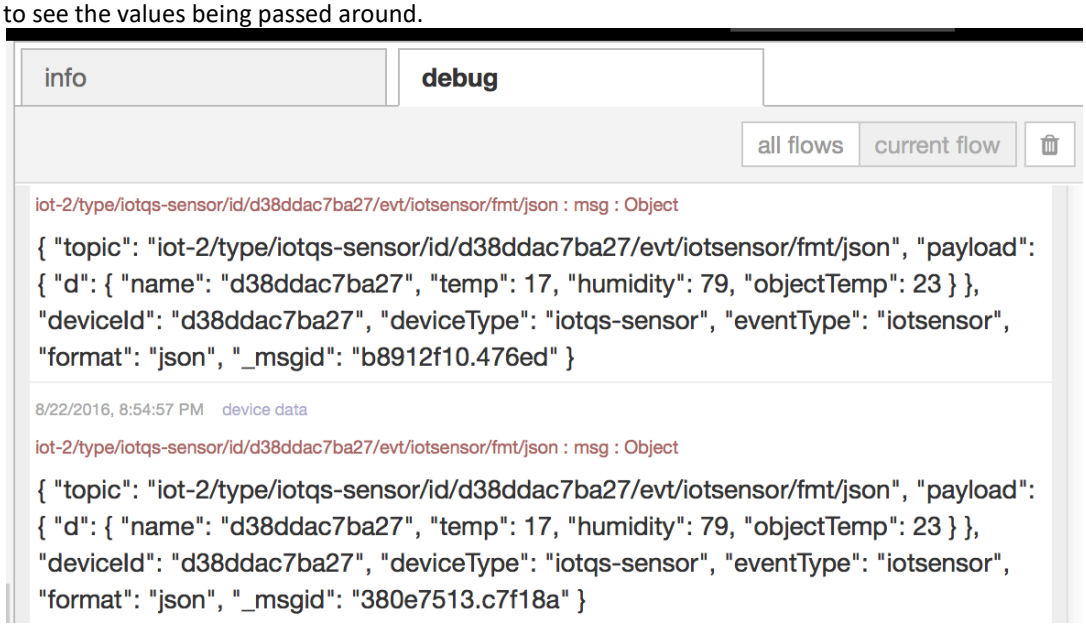
10. Copy this alphanumeric device ID into the Device ID field in the Node-RED application as shown below. Click **Done**.

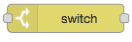


11. Click on the 

button in the top right of the screen to save and deploy your changes.

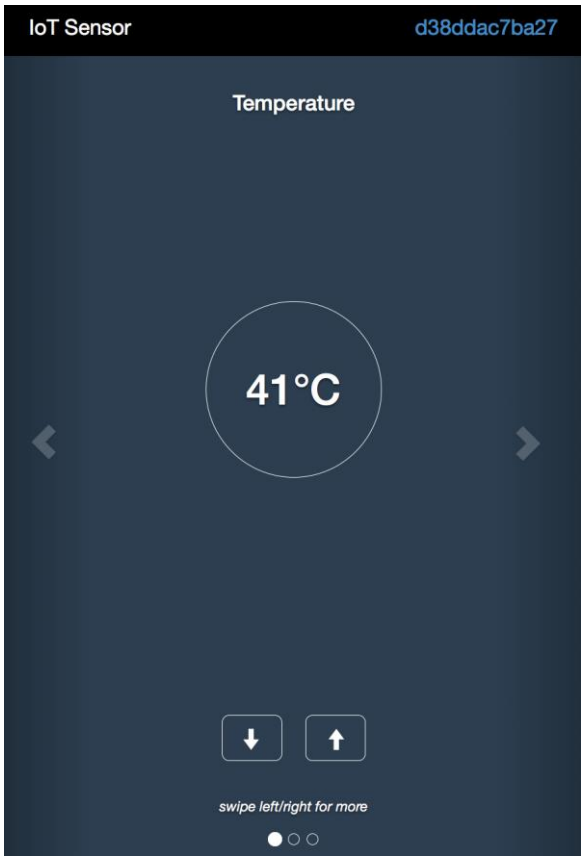
12. Click on the debug tab in the right-hand pane. Every second, the simulator emits a device event to the Watson IoT platform with temperature and other data. The Node-RED application subscribes to these events and the IBM IoT in node triggers the flow with the temperature data in the message. When the debug nodes are processed, contents of the message object are in the debug tab. Adding debug nodes can be helpful when something doesn't work right and you want to see the values being passed around.



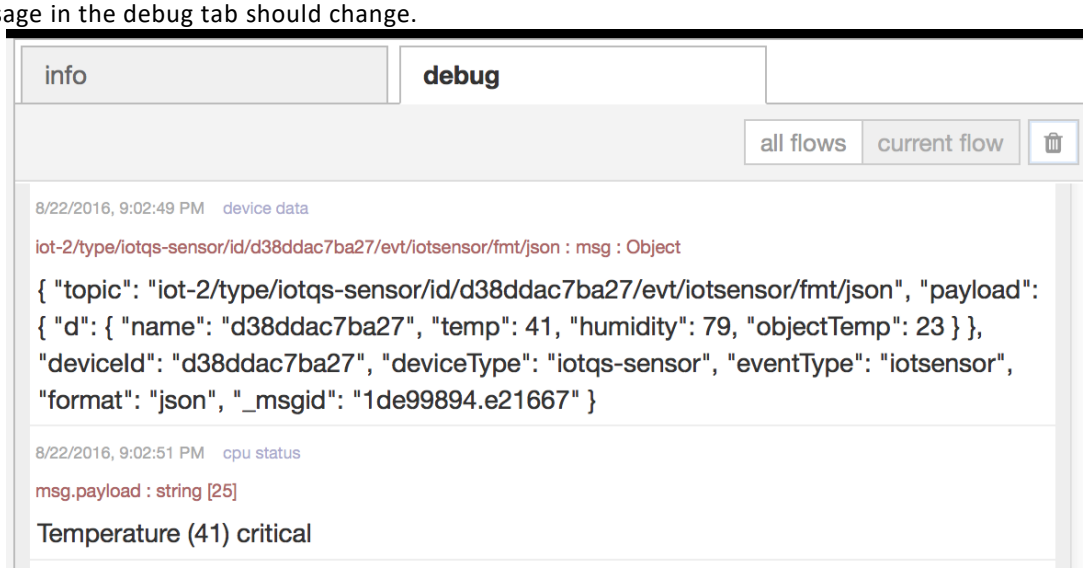
13. The yellow node is called a 

node. You can program logic using a switch node and split a flow into two or more flows based on a property's value. In this example, if the temperature is less than or equal to 40°C, it is considered "safe" and continues with the flow to the template labeled safe. If the temperature is greater than 40°C, it is considered "danger[oust]" and continues with the flow to the template labeled danger.

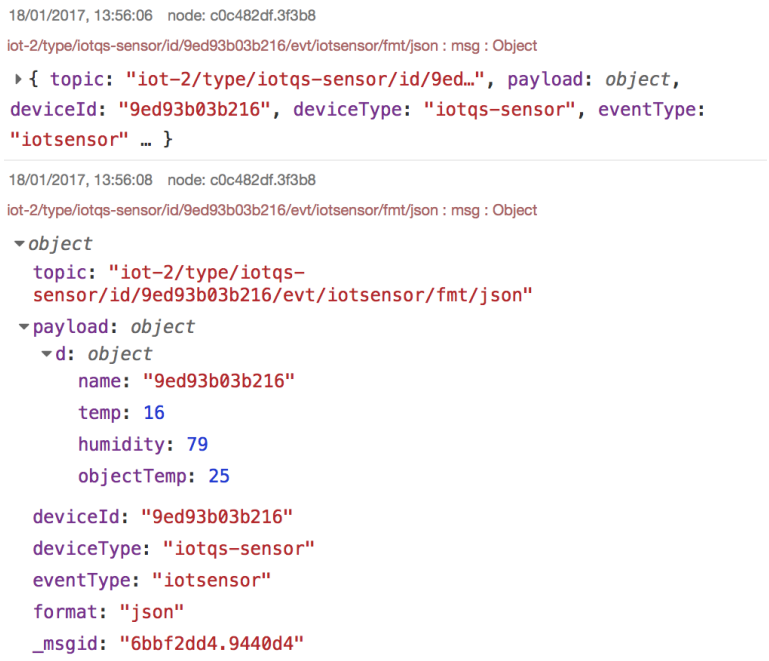
14. To trigger the flow labeled danger, increment the temperature using the up arrow on the simulated temperature gauge.



The message in the debug tab should change.



Note: There has been a recent update to Node-RED and with the new version you need to expand the objects in the debug panel to see values:



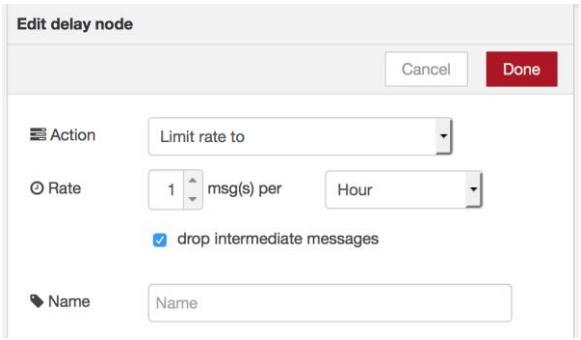
Connect to Twitter and Tweet High Temperature

In this section, we will connect a Twitter account and use the Twitter account to tweet when the temperature from the temperature sensor is “dangerous”. This section is optional and may be skipped.

1. Sign up for a Twitter account at http://twitter.com . If you already have a Twitter account, proceed to step 2.

2. Add a 

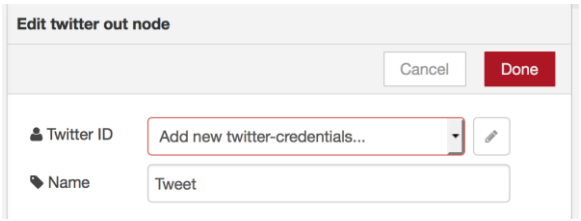
node as shown below.

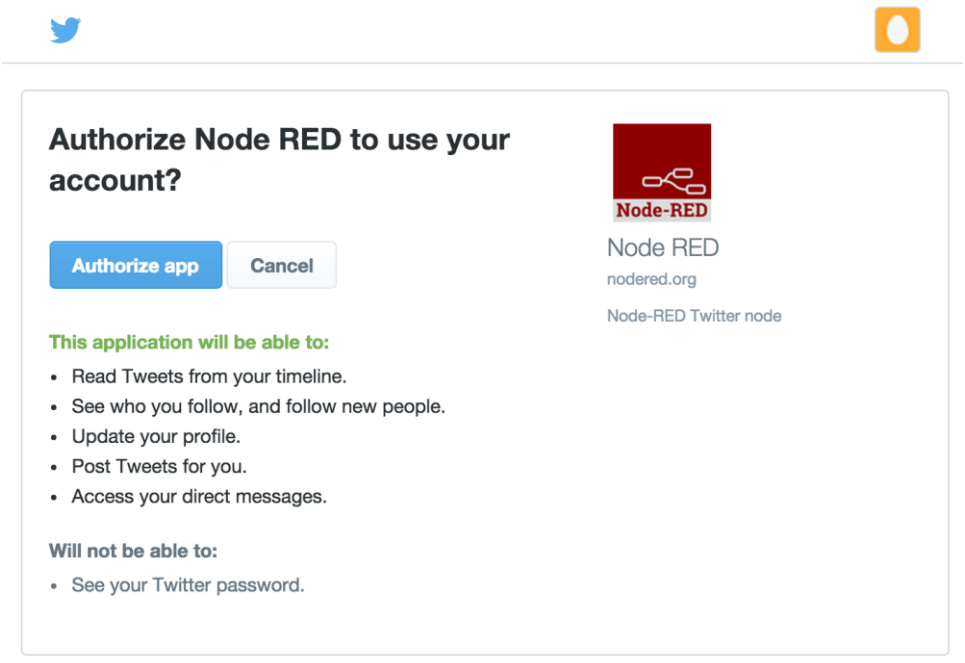


The delay node limits how often the flow is run. Since the temperature is dangerous every second, without this node, a tweet would be sent every second. With this node limiting messages to once an hour, the Twitter node will send a tweet once an hour, dropping any additional messages during the hour timeframe.

3. Add a 

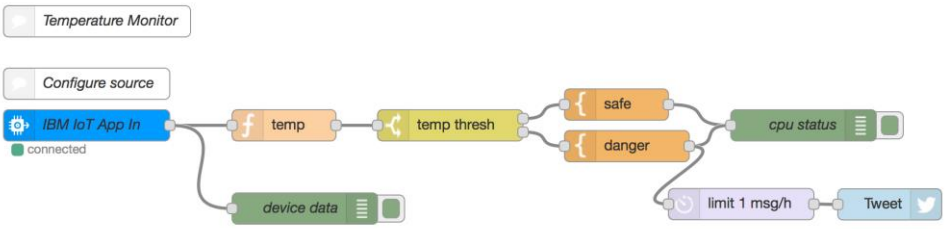
node. Click on the pencil button and authenticate with Twitter. The account you sign in with will be used to send tweets.





Return back to the node configuration. The settings for the Twitter node should have your username set. Click **Add**.

4. Connect the nodes as shown below.



5. Click on  to save and deploy the changes.

6. Since the temperature is above 30°C, the switch statement will continue to the “danger[ous]” function, compose a message, and pass it to the Twitter node. The Twitter node uses this message as the content for the tweet.

7. Visit the Twitter timeline for the user you authenticated with and verify the message has been tweeted.

There are other ways to send notification to users, such as e mail and SMS (via twilio). The nodes for these services are pre-installed, so feel free to try these instead of sending a tweet.

Create a Node-RED flow on Intel NUC

In the first section, we used a simulated device to get started with the Internet of Things Platform. When you have simulated what your device will do, then you can invest in making the hardware and hopefully reduce the cost incurred along the way. The Intel NUC board has made prototyping IoT devices easy. The Grove Starter Kit comes with a variety of sensors that can be connected to the Intel NUC and Arduino 101. With Node-RED nodes and some custom logic it enables developing hardware solutions down to a level where anyone can build quickly and easily.

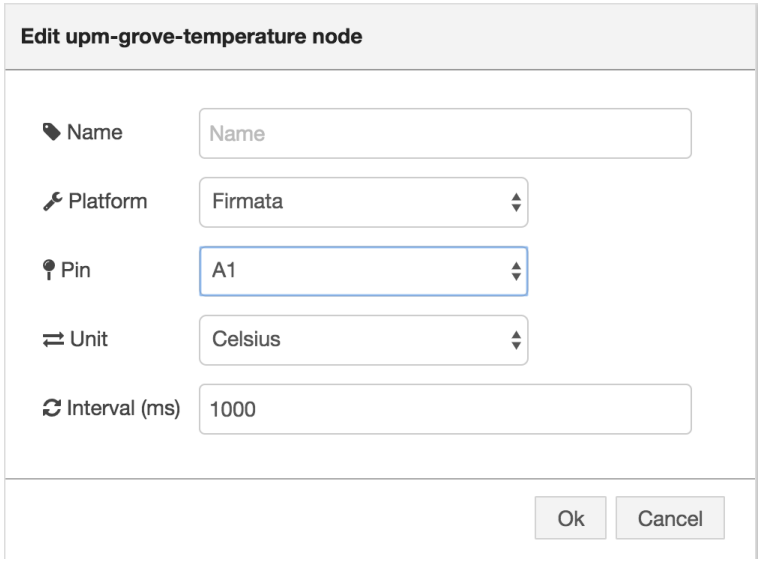
In this section, we will connect to the Node-RED application running on the Intel Edison board, capture the value of a real temperature sensor, control an LCD screen, control a LED light, and send the temperature to the Watson Internet of Things Platform.

Your setup from the previous section should be:

- Rotary switch connected to A0
- Temperature sensor connected to A1
- Grove LCD RGB Backlight connected to I2C connector

1. Clear up all sheets from by deleting default application and previous work. Select all nodes on the sheet then hit the delete or backspace key on your keyboard. When the nodes have been delete hit Deploy to stop the running flows that were deleted.
2. To get started, drag a 

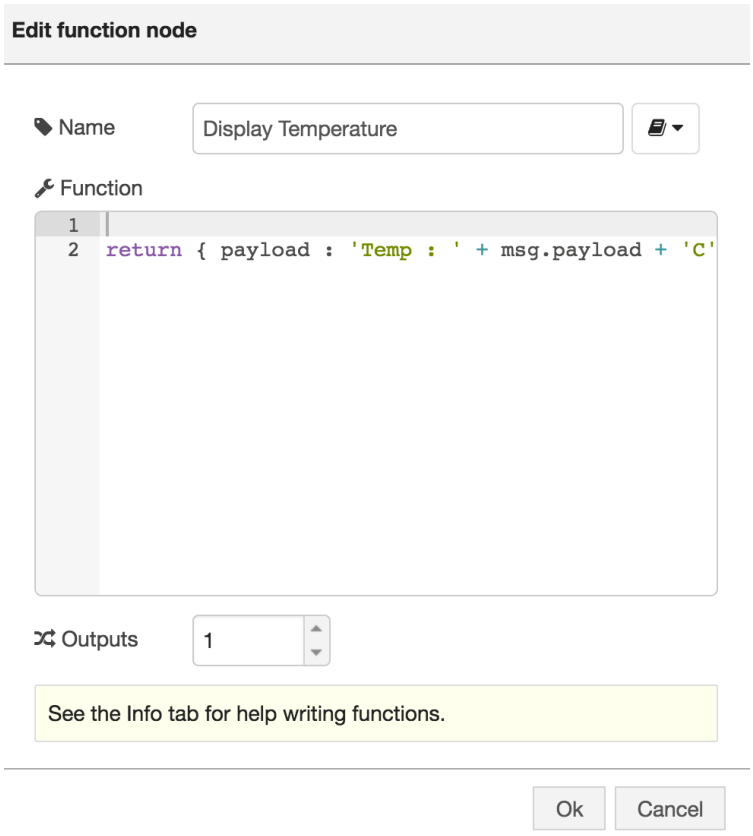
Grove Temperature node onto the canvas. Double click on the Grove Temperature node and customize as shown below.



This will configure the temperature node to listen on Port A1, and return Celsius temperature value.

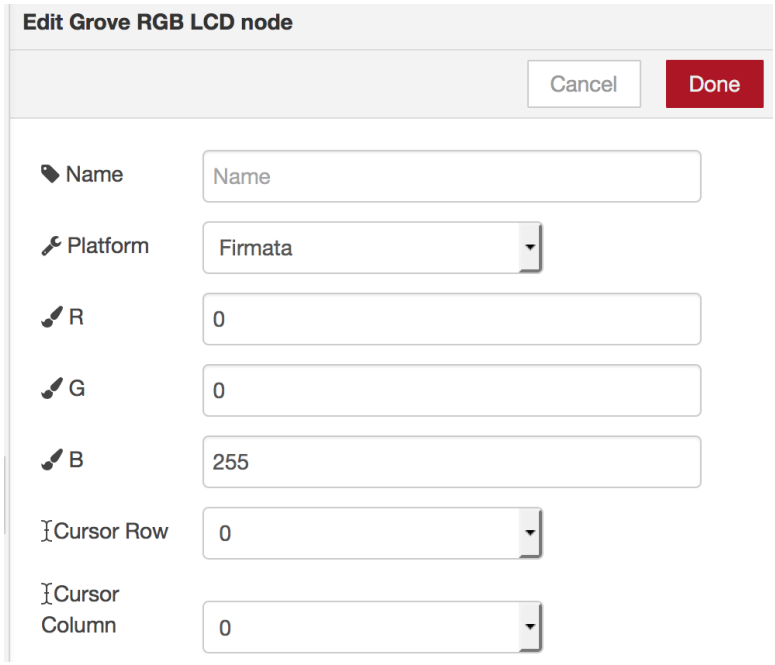
3. Add a function node as shown below:

4. Add a function node as shown below:




```
return { payload : 'Temp : ' + msg.payload + 'C'};
```

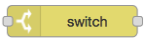
5. Add a  Grove RGB LCD node as shown below.



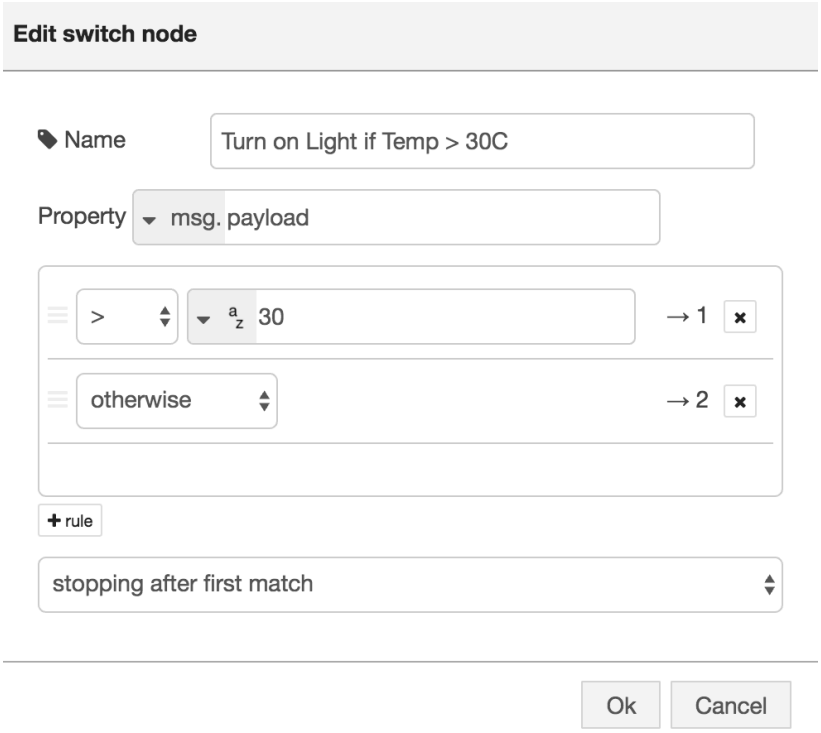
This will display the temperature value on the LCD connected to the I2C port.

6. Connect the nodes together as shown below.

7. Click on the  button in the top right of the screen to save and deploy your changes. The LCD screen should display the temperature in Fahrenheit. The temperature value screen will update once every second when the flow is activated by the inject node.

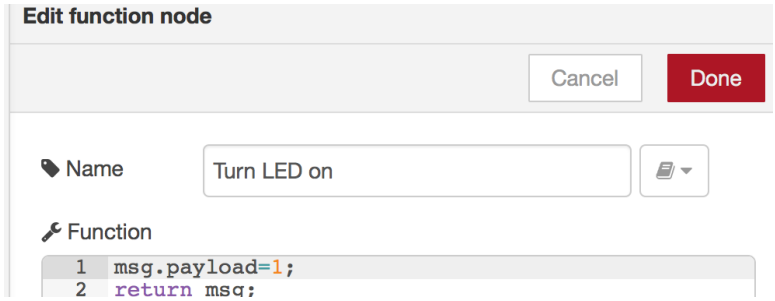
8. Next, we will activate a LED light when the temperature exceeds 80°F. Add a 

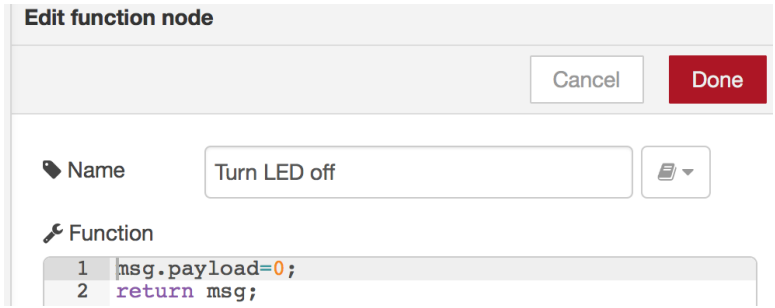
node and connect it to the temperature node as shown below. (Adjust the trigger temperature, considering the current room temperature)



9. Add two 

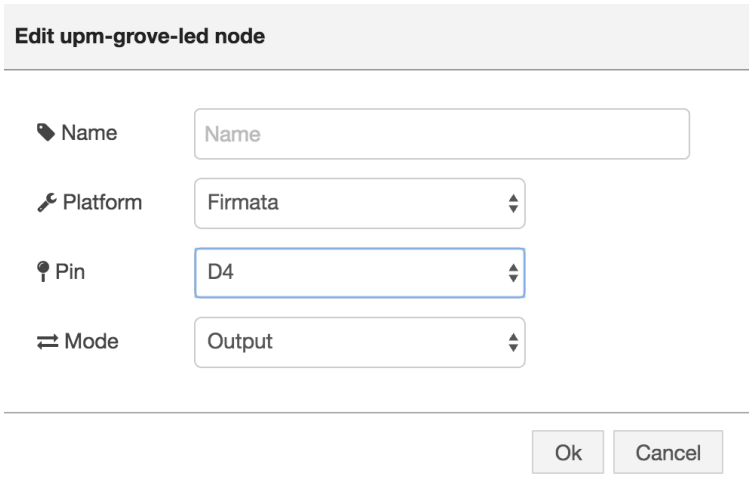
nodes, one to turn the LED on, and one to turn the LED off as shown below.



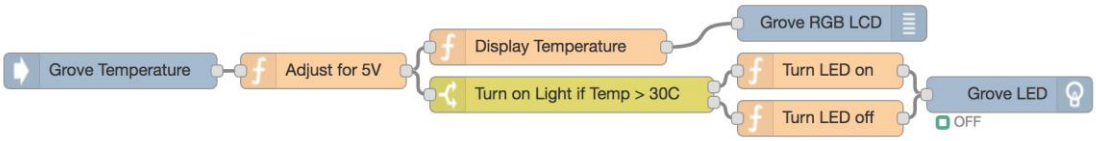


10. Add a 

Grove LED node as shown below. This will control the LED connected to port D4.



11. Connect the nodes together as shown below



12. Connect a LED light to the D4 port on the Grove base shield connected to the Intel Edison.

13. Click on the 

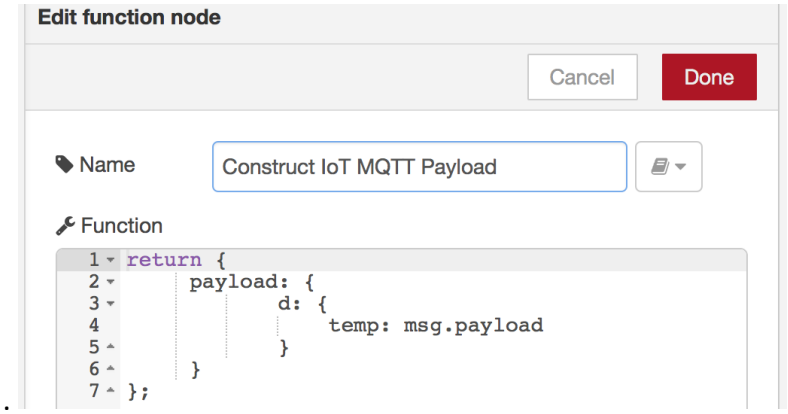
button in the top right of the screen to save and deploy your changes. When the temperature reaches above 80°F, the LED light will turn on. Otherwise, the LED light will turn off.

Sending Temperature Sensor to IoT Platform

In this section, we will connect the Node-RED application running on the Intel NUC, Arduino 101 and Grove Shield to the Watson Internet of Things Platform service. We will send the temperature value from the Grove temperature sensor as device events so that the Node-RED application in IBM Cloud can react to high temperatures.

1. Add a 

node as shown below




```
return {
    payload : {
        d : {
            temp : msg.payload
        }
    }
};
```

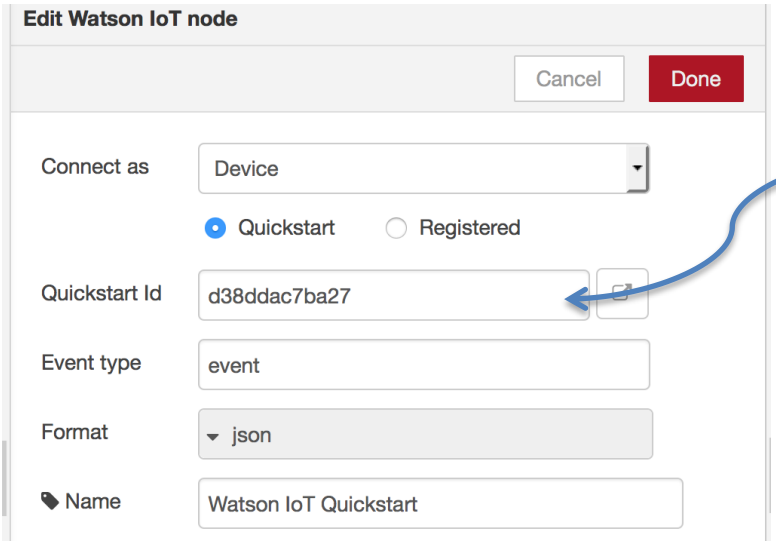
2. This JavaScript function constructs and returns a JavaScript object. The JavaScript object has a property named **payload**, which has property named **d**, which contains a property named **temp**. We use the Celsius value from the temperature sensor and assign it to the property **temp**.

3. Add a 

node. This node does one of two things. When passed a JSON string, it will attempt to parse it into a JavaScript object. When passed a JavaScript object, it returns a JSON string. Since we're passing in a JavaScript object, it will return a JSON string.

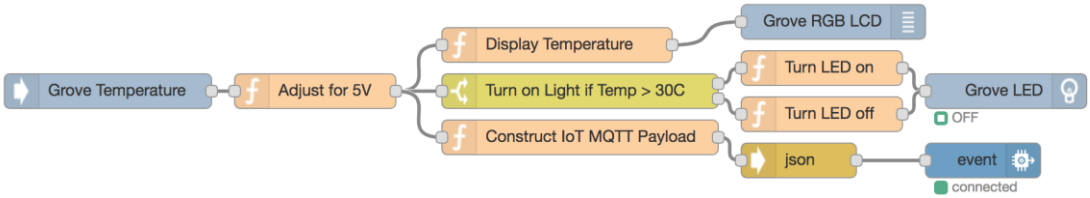
4. Add a 

node as shown below. You can reuse the device Id generated by the simulated temperature sensor, or, change the device ID here and in the Watson IoT in node of the Node-RED application in the cloud. **If you reuse the device ID, close the simulated temperature sensor window to stop the simulated temperature values from conflicting with the real temperature data.**



Reuse Device ID from simulated IoT sensor

5. Connect the nodes together as shown below.



6. Click on 

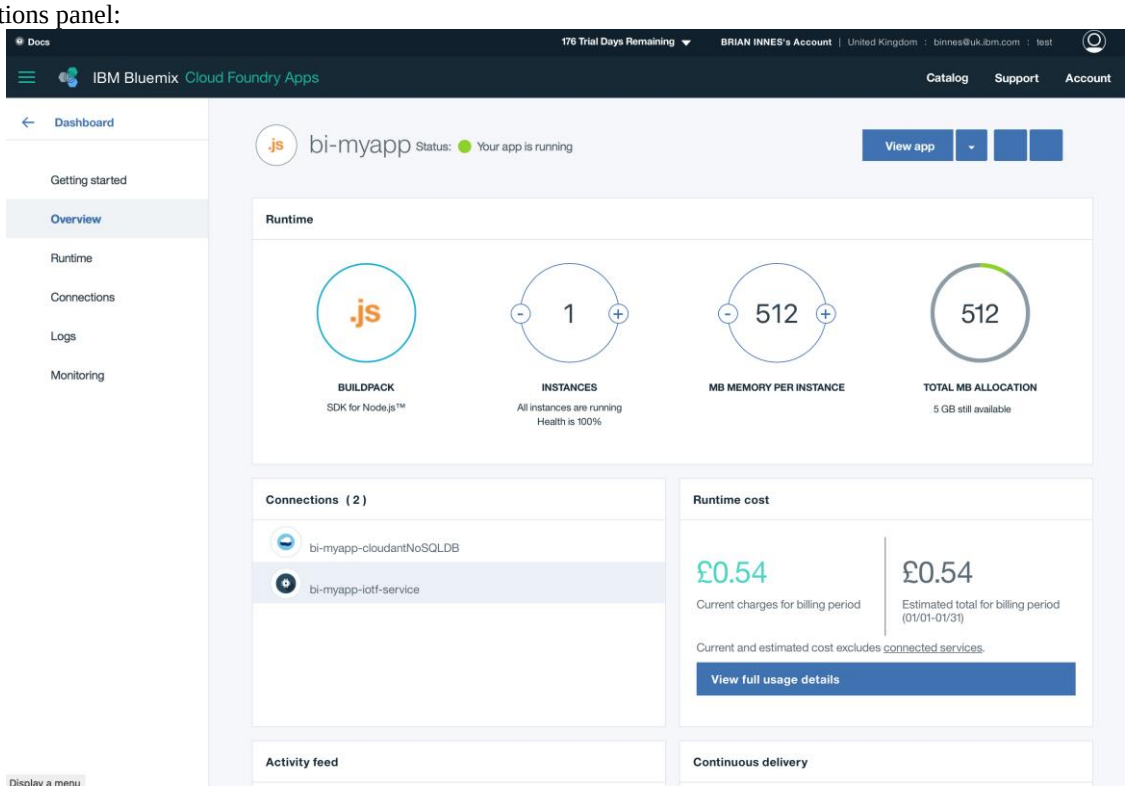
to save and deploy your changes.

7. Return to the Node-RED application in IBM Cloud. You should see the real temperature being reported.

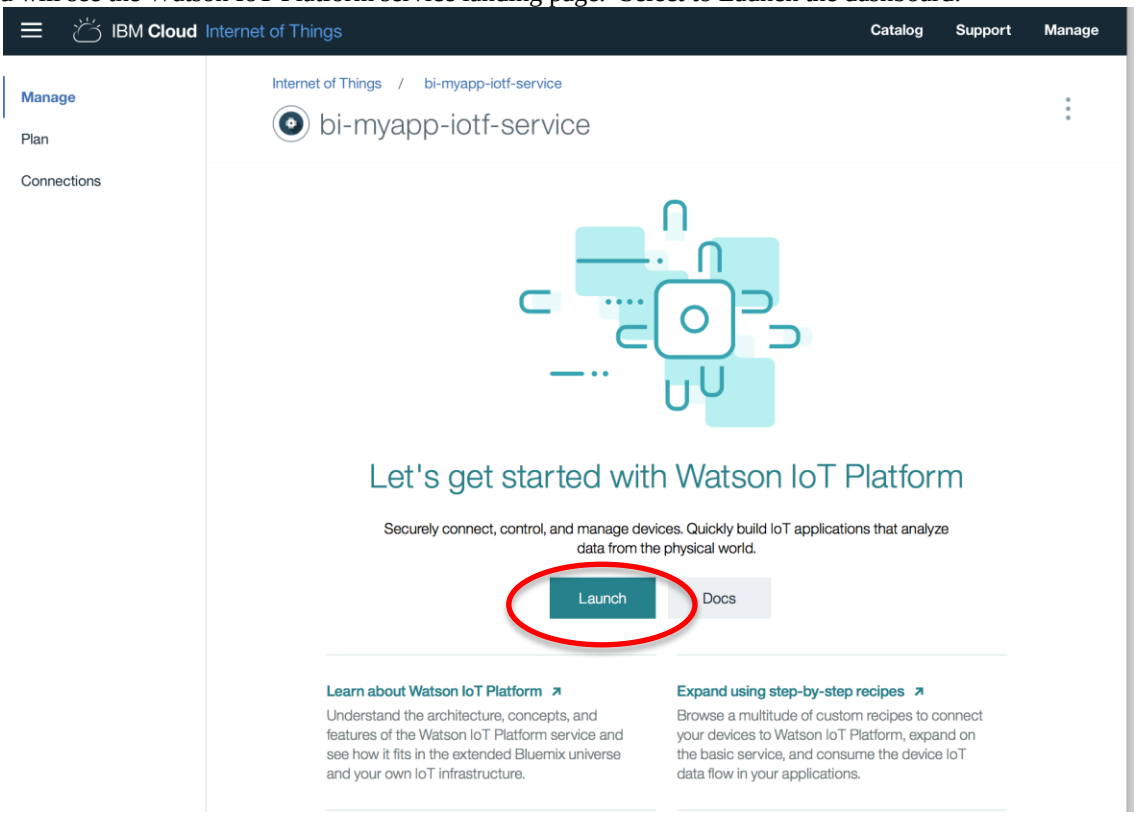
Moving to secured communications with registered devices

Up to this point you have use the Quickstart service of the Watson IoT platform. The Quickstart platform is good to get something working quickly, but is an open, unsecure environment. In this exercise, you will move to using registered devices against your own organization within Watson IoT platform.

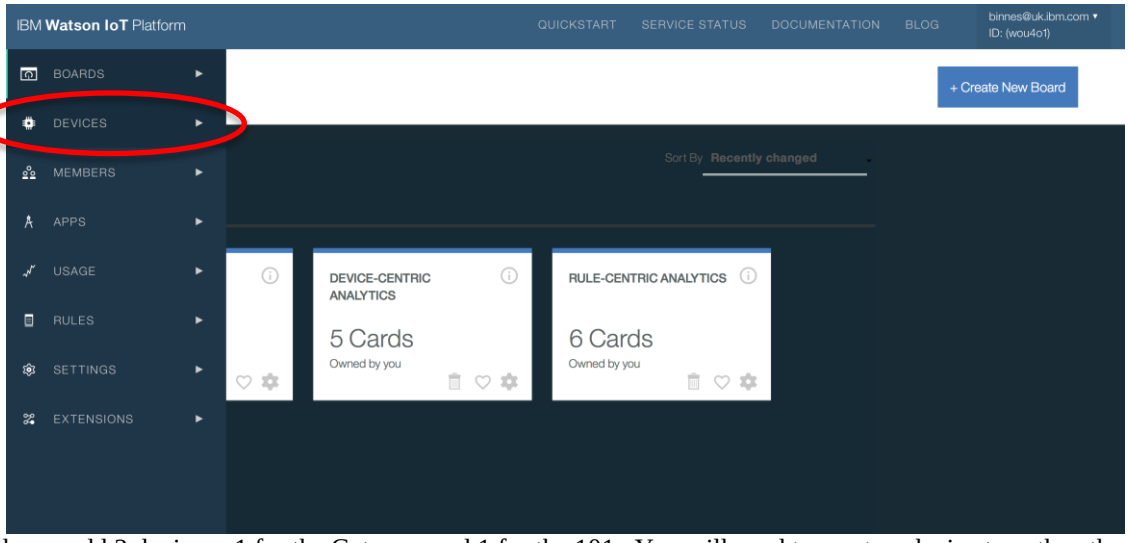
1. In IBM Cloud go to the Overview page of your boilerplate application and double click the iotf service in the Connections panel:



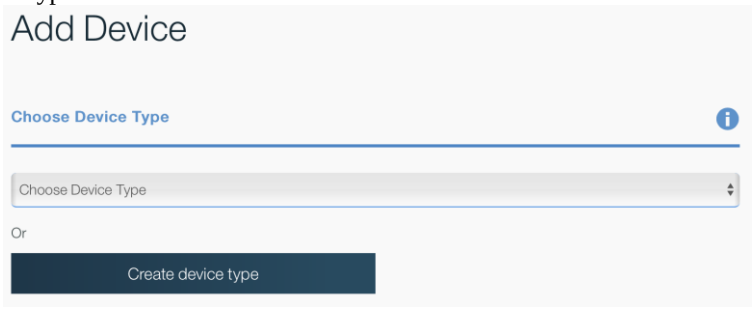
2. You will see the Watson IoT Platform service landing page. Select to Launch the dashboard:



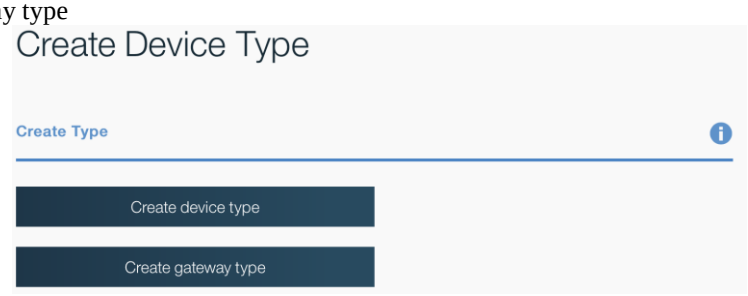
3. From the side menu select the devices section:



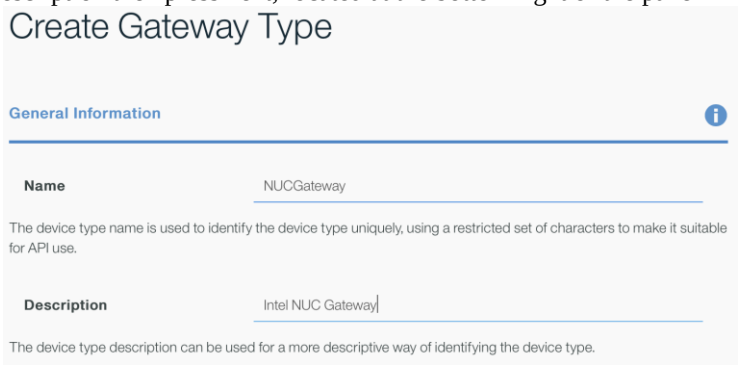
4. You will now add 2 devices. 1 for the Gateway and 1 for the 101. You will need to create a device type then the device:
    a. Select the "+ Add Device button"
    b. Then the Create device type button



    c. Select Create gateway type



    d. Provide a name and description then press next, located at the bottom right of the panel



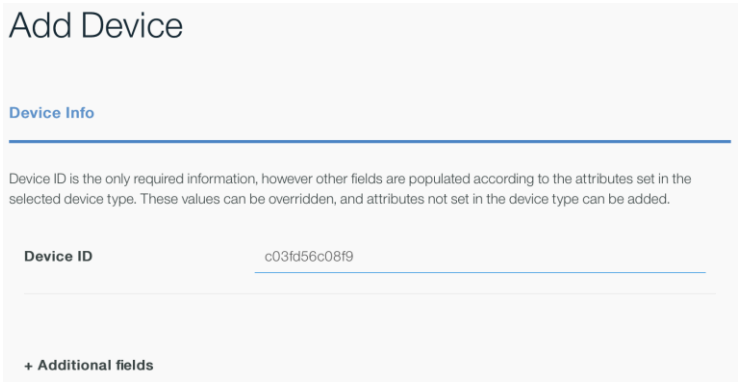
    e. You can leave the next panel with default, so press next and next again. You can enter metadata for the type, but we will leave this blanc, so press create. You have completed defining the Gateway type for the NUC

f. To add your NUC device leave the device type selected and press next

g. Enter a device ID. This must be unique for the device you are registering within the type. A common practice is to use the physical network address (MAC Address) of the device without the colons. You can find this from the NUC by opening an ssh session and entering the command ifconfig:

```
root@bi-intelGW:~# ifconfig
eth0      Link encap:Ethernet  HWaddr c0:3f:d5:6c:08:f9
          inet addr:192.168.0.248  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c23f:d5ff:fe6c:8f9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:59401 errors:0 dropped:2 overruns:0 frame:0
          TX packets:48185 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14508403 (13.8 MiB)  TX bytes:6603211 (6.2 MiB)
          Interrupt:106 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:15134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15134 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1928147 (1.8 MiB)  TX bytes:1928147 (1.8 MiB)
```

h. In this example the device ID will be c03fd56c08f9. Press next when Device ID has been entered

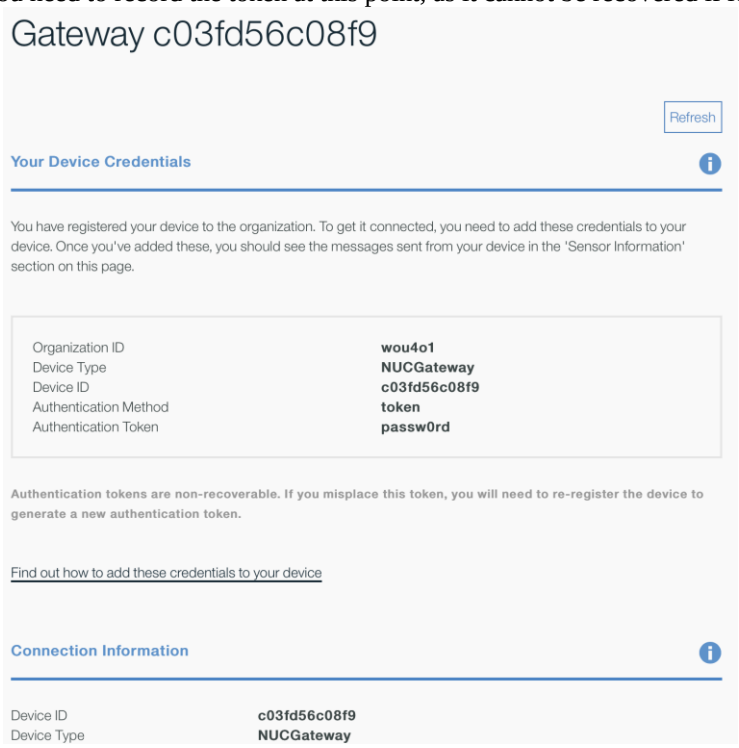


i. We will not add any additional metadata for press next

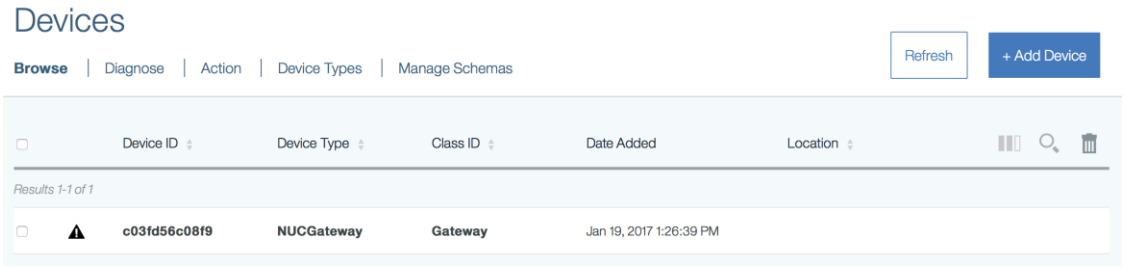
j. You have the option of entering a token (password). If you leave it blank a token will be auto generated. However, when prototyping you may need to register devices multiple times and enter the token application configuration, so you may want to set the token to something you can remember. Press next.

k. You are now shown a summary of the device. If all is correct press Add to add the device.

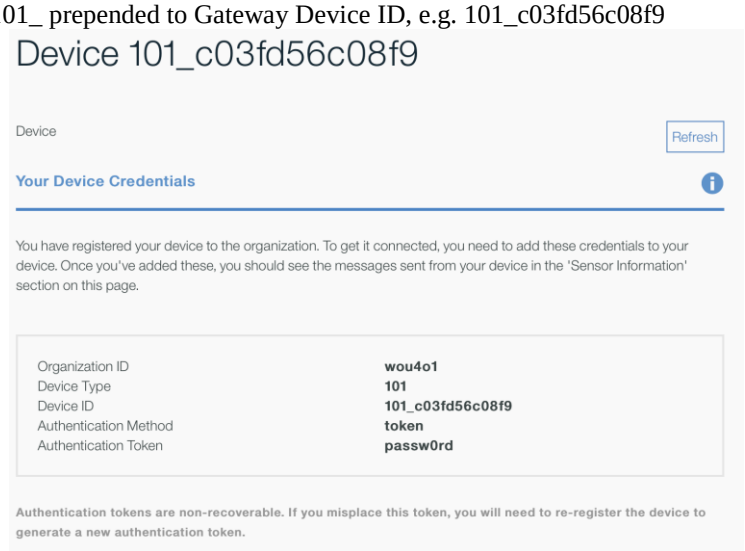
l. Once added you see the device definition - this is the last time you can see the Authentication Token - if one was auto-generated you need to record the token at this point, as it cannot be recovered if lost.



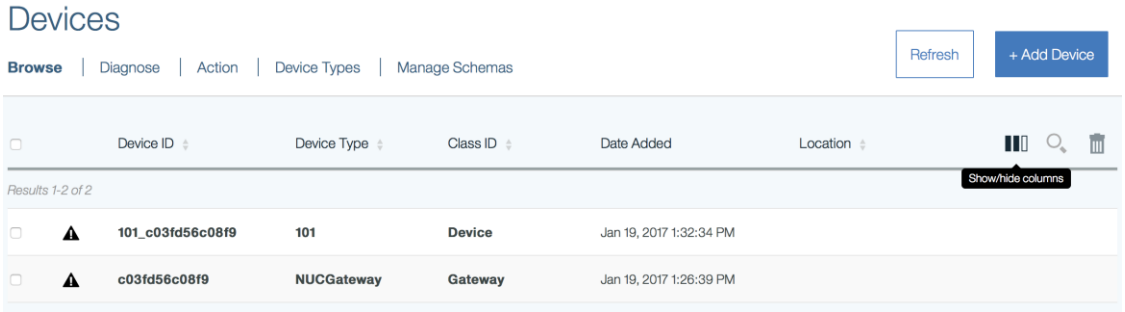
m. Press the ⊗ to the top right of the device panel to close it. You will now see the registered device:



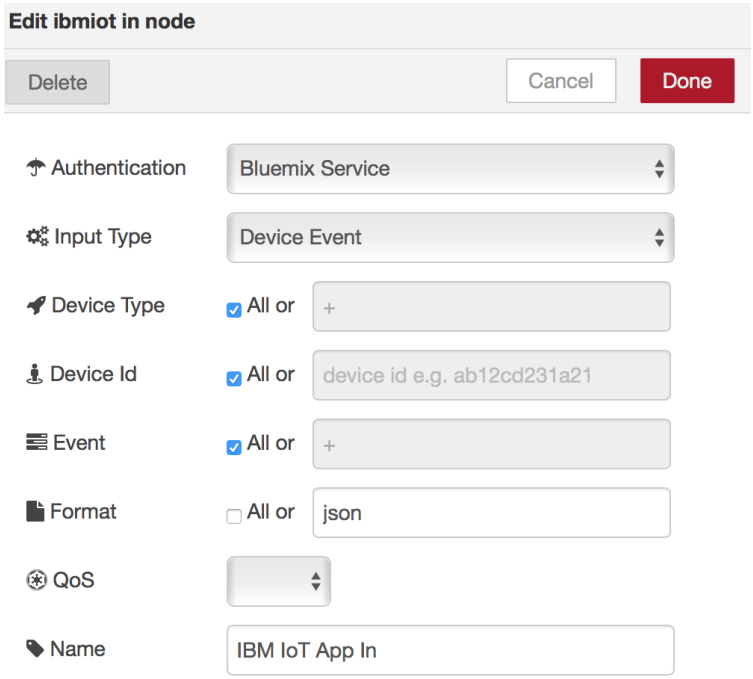
n. Repeat the process for the 101
   i. Create a device type rather than a gateway type
   ii. Device Type : 101
   iii. Device ID 101_ prepended to Gateway Device ID, e.g. 101_c03fd56c08f9



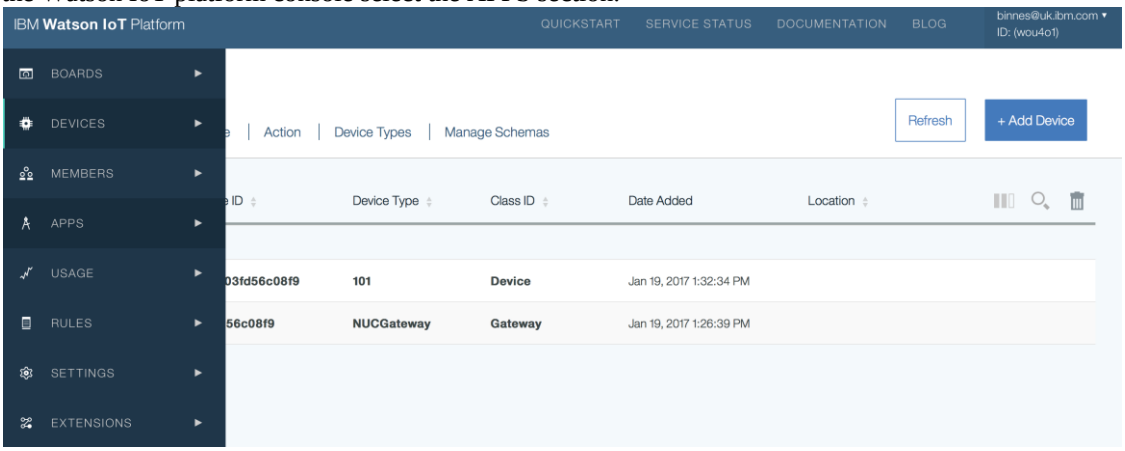
o. You will now have 2 devices registered to your platform:



5. Moving to Node-RED running on IBM Cloud, you need to update the node to receive data from the registered device rather than from the quickstart sandbox. Open the Node-RED editor and then the configuration for the IBM IoT App In node. Update the configuration to Authenticate to the IBM Cloud Service and receive Device Events from All Divice Types and Device Ids:

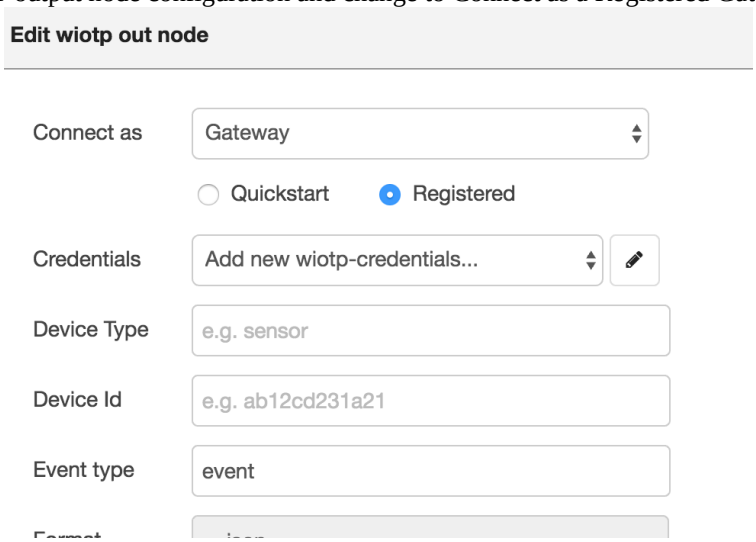


6. Hit deploy to make the updates live. Node-RED running on IBM Cloud can take advantage of bound services to the application and several nodes make use of this and do not require you to enter credentials. On IBM Cloud the application is automatically registered on the Watson IoT platform and nodes can access credentials using platform standard features.
7. Back in the Watson IoT platform console select the APPS section:

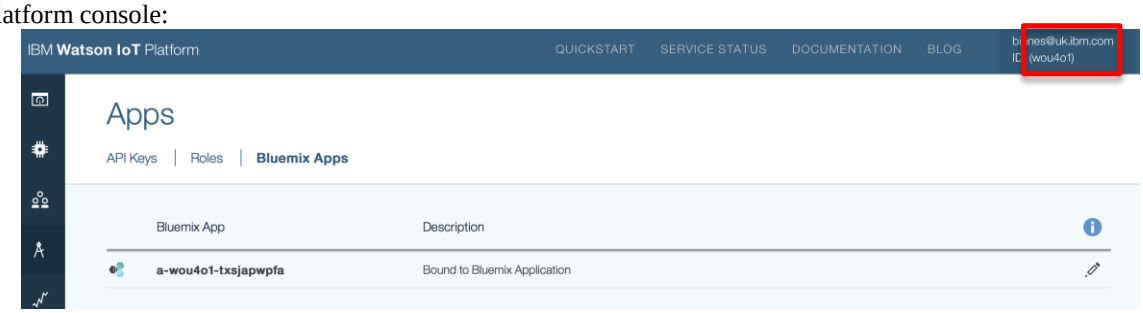


8. Then select the IBM Cloud Apps. You see the Node-RED application is automatically registered here. If you were not on the IBM Cloud platform you would need to create an API Key and use that to connect to the platform as an application.
9. Now switch to Node-RED running on the NUC Gateway. You need to get this application to publish as a registered device. You will connect to the platform as the Gateway device, but publish data from the 101 device.

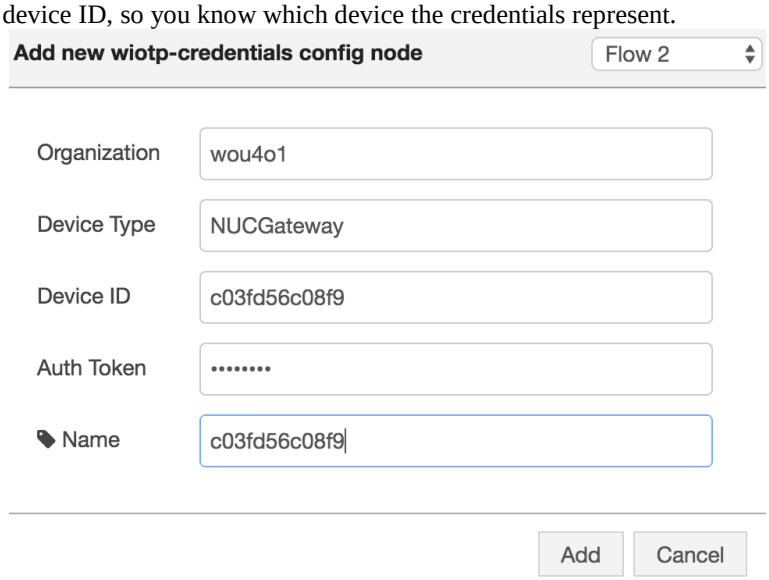
a. Open the Watson IoT output node configuration and change to Connect as a Registered Gateway:



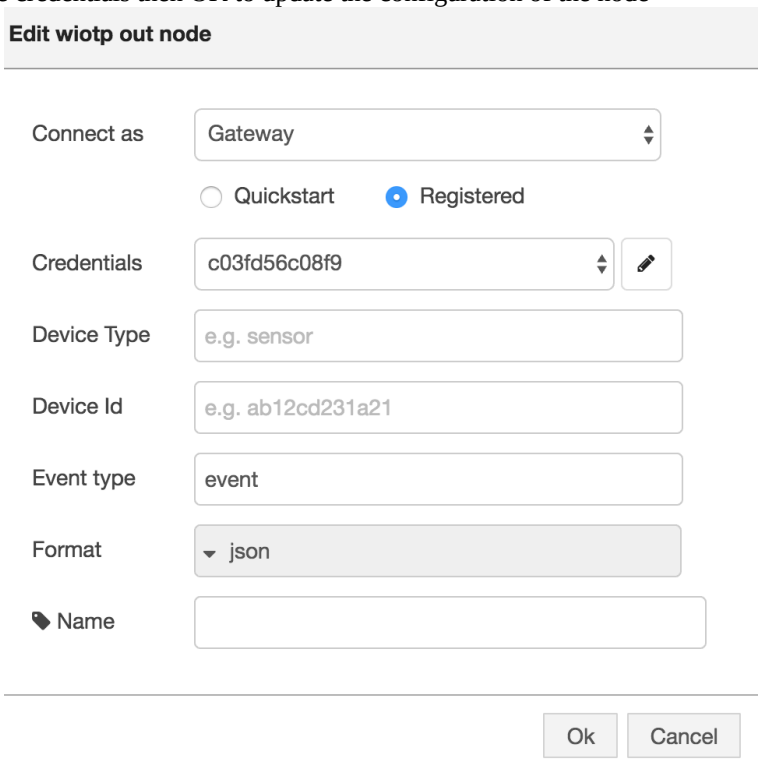
b. Press the pencil button to the right of the Credentials box to Add new wiotp credentials. You need to enter the details for your registered NUC Gateway device. The Organization can be found at the top of your Watson IoT Platform console:



Enter the details, referring to the Devices section of the Watson IoT Platform console if required. Enter the Auth Token for the NUC Gateway. (If you didn't record it when registering the device you will need to delete the device from the Watson IoT Platform and reregister it to get a new Auth Token). You may want to name the credentials with the device ID, so you know which device the credentials represent.



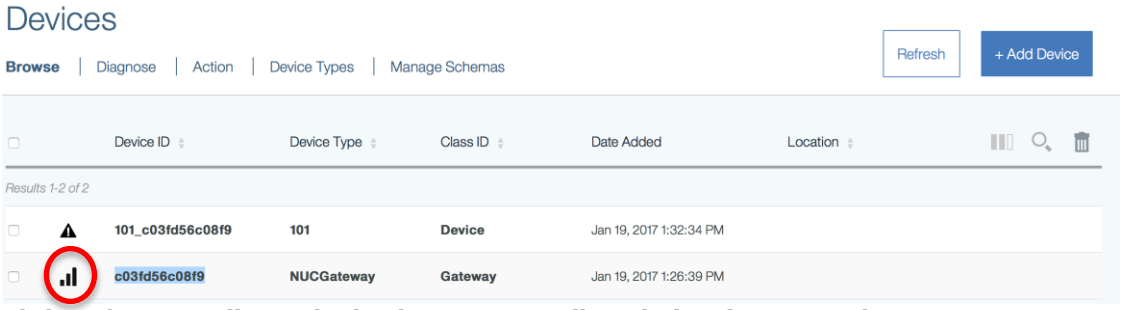
c. Press Add to save the credentials then OK to update the configuration of the node



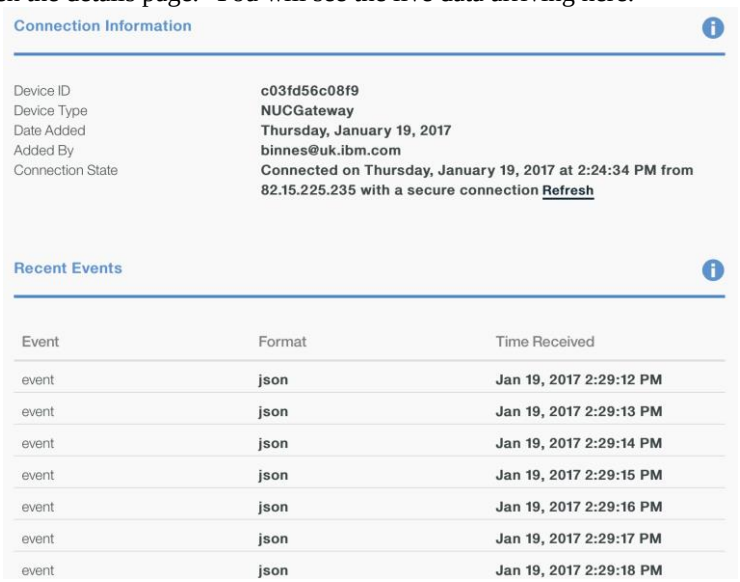
Note: If you want the node to publish data on behalf of a device you can enter the device type and device ID in the configuration. If you leave it blank the data will be published as the gateway device - leave the fields blank.

d. Press Deploy to make the changes live.

10. Switch to the Watson IoT Platform console and to the devices section. You should see the NUCGateway is connected.

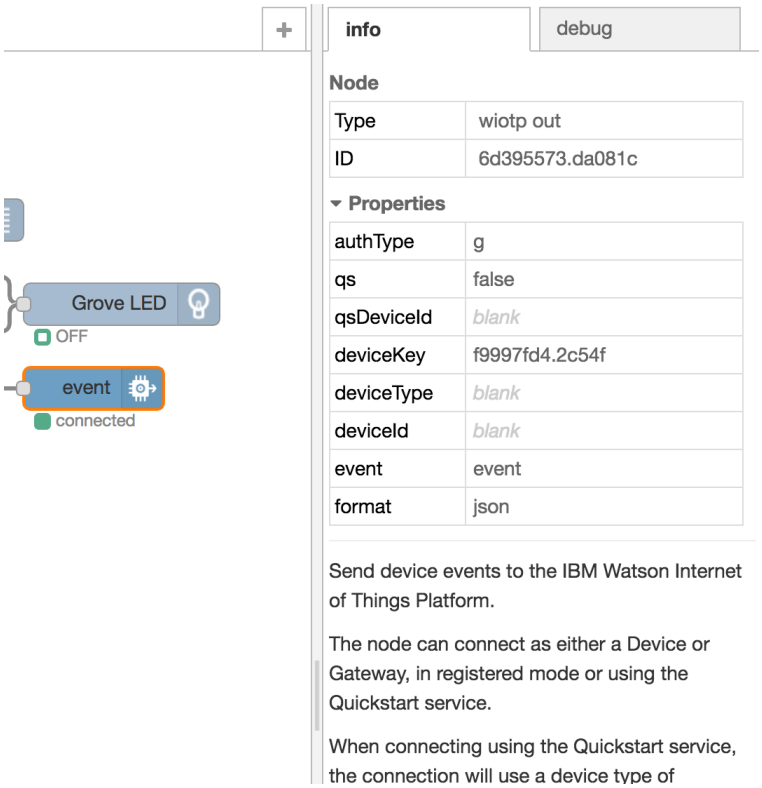


Clicking the entry will open the details page. You will see the live data arriving here.



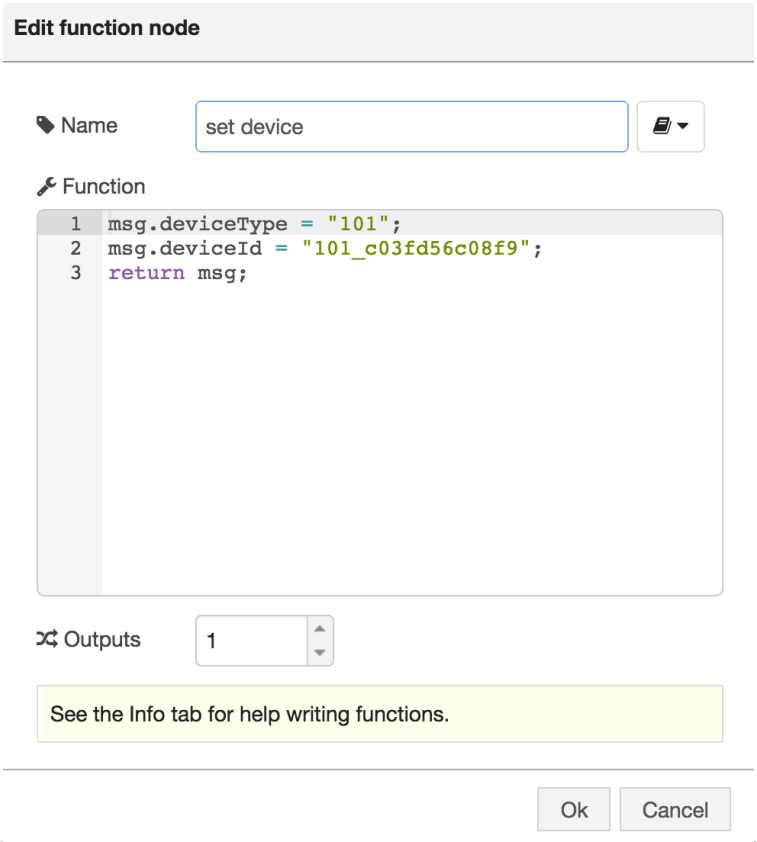
11. You now have the data flowing securely to your private organization on the Watson IoT Platform, but it is arriving as the wrong device. It should be arriving from the 101 device. To fix that we can alter the flow and use the functionality to send node configuration data in the incoming message. Click the Watson IoT node in the Node-RED editor to select it (orange border) and then switch to the Info panel. Expand the Properties section. Here you can see the properties that the node uses. Some properties are populated from the configuration panel, but you will notice that the deviceType and deviceId properties are blank, as you left them blank in the configuration. We can send these properties into the node

with the temperature data.



12. Add a 

node to the sheet and configure it as shown. Enter your device ID and device Type to match what you registered in the Watson IoT Platform:




```
msg.deviceType = "101";
msg.deviceId = "101_c03fd56c08f9";
return msg;
```

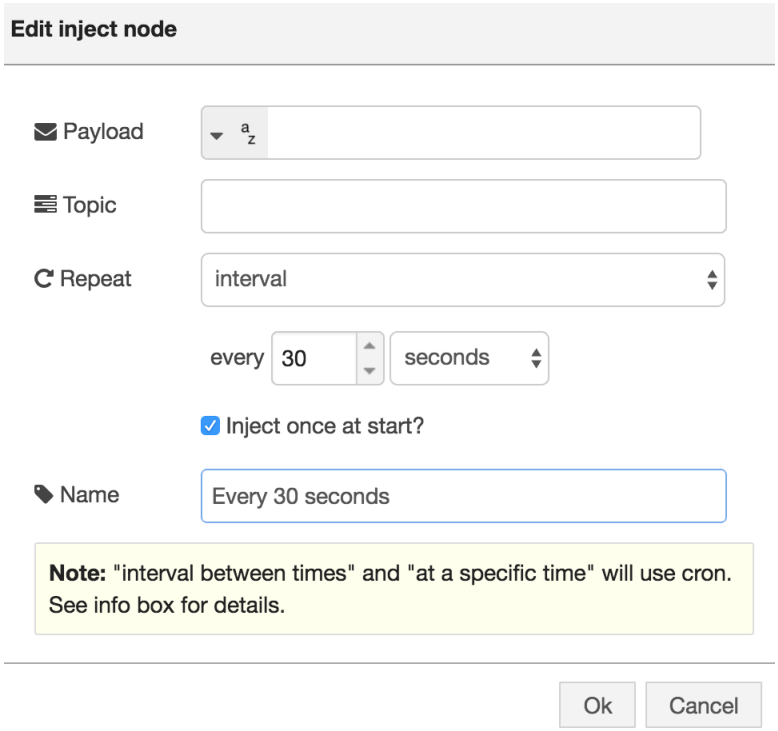
13. Hit deploy and then go look back in the Watson IoT Platform console. You should see that the Gateway is still connected and the device is not showing as connected, but clicking into the Gateway device you should not be seeing any

data arriving on the platform. Clicking into the device you should now be seeing the device data arriving, as the Gateway is now publishing data on behalf of the device.

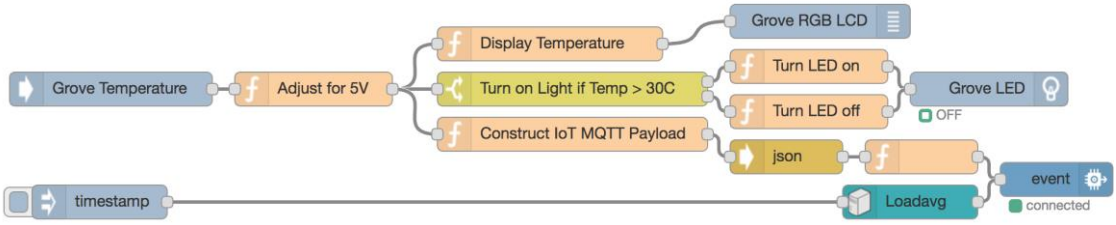
14. Add a 

node and a 

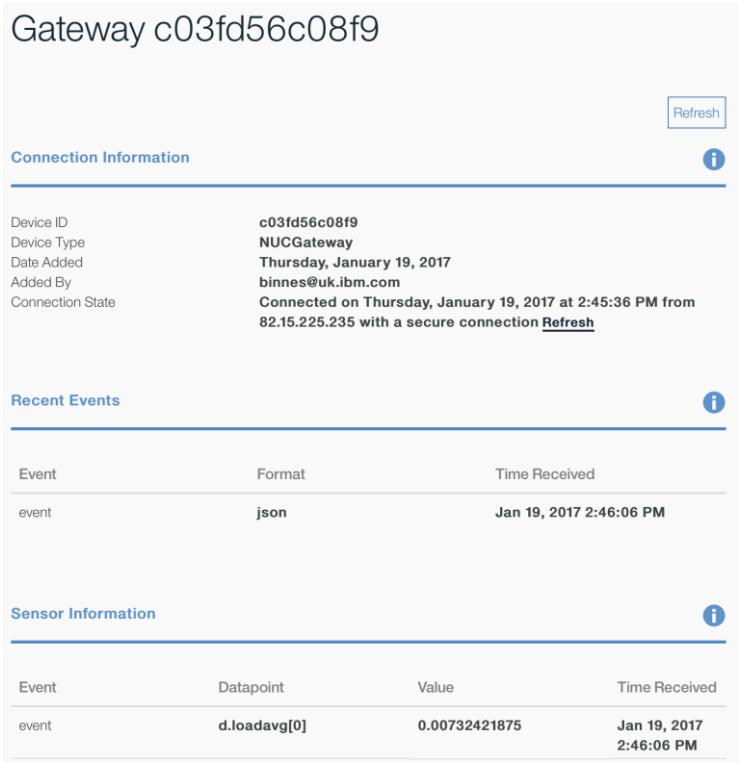
node. Configure the inject node as shown. The Loadavg node requires no configuration:



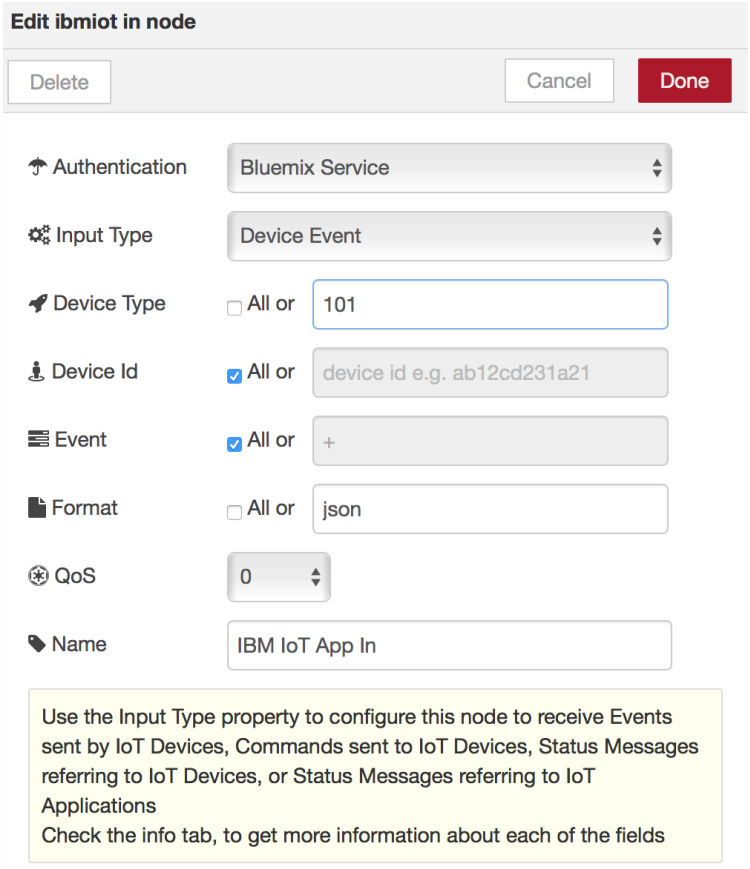
Connect the nodes as shown:



15. Now you have the CPU load average being published by the gateway every 30 seconds, and the temperature from the 101 device being published. Verify the correct data is being published from the correct device using the Watson IoT Platform console.



16. Now you are publishing the CPU Load Average data, you need to go back to the Node-RED application running on IBM Cloud, as this is only expecting the temperature data, but the node is configured to receive data from all device types and all devices. Edit the node configuration to only receive data from the 101 device type (modify to match the device Type you created if not 101). Hit deploy and verify that you are still receiving data from the 101 and not from the NUC Gateway device.

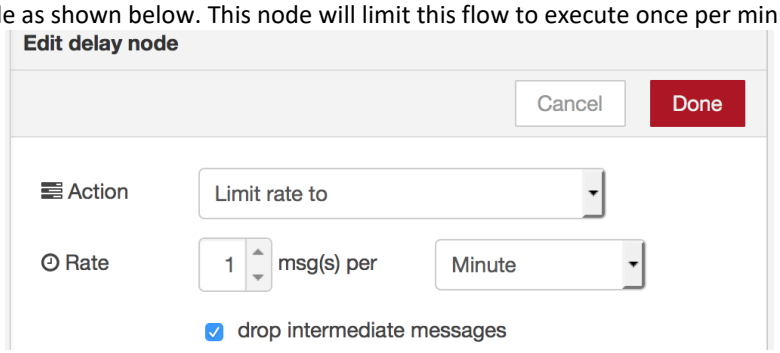


Store Temperature in to a Cloudant NoSQL Database

This section will show how to add a Cloudant NoSQL database to the Node-RED application and store temperatures reported (one per minute). This functionality can be useful to run historical analysis (outside the scope of this lab) or find patterns over time. In this section you will work in the Node-RED application running on IBM Cloud to extend the sample application deployed with the boilerplate.

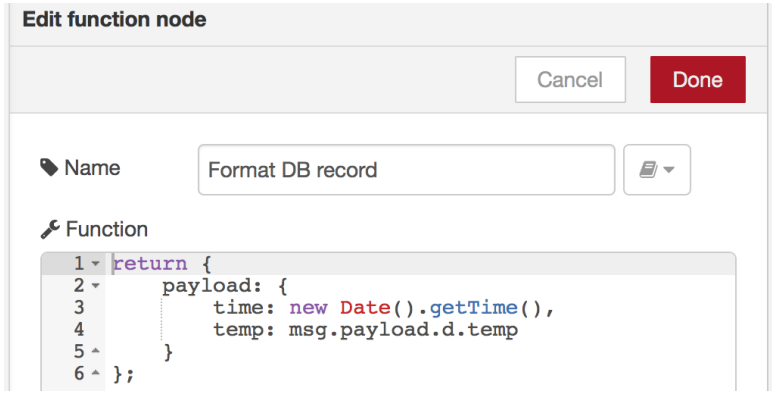
1. Add a 

node as shown below. This node will limit this flow to execute once per minute.



2. Add a 

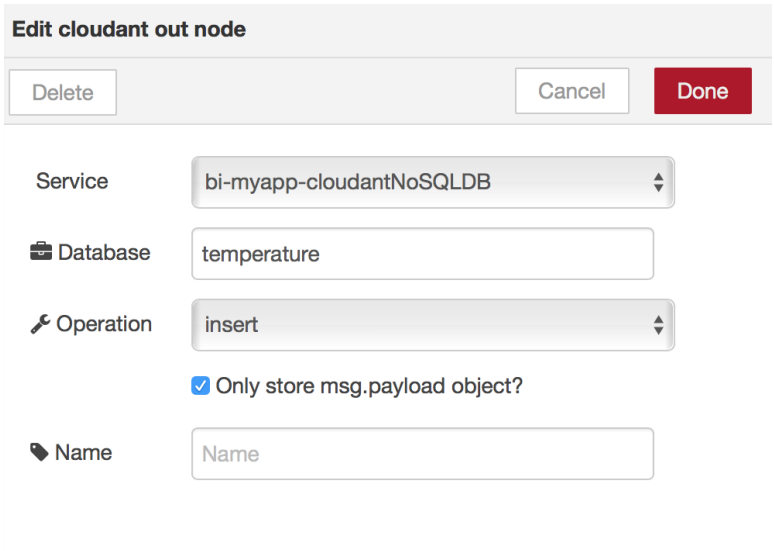
node as shown below.



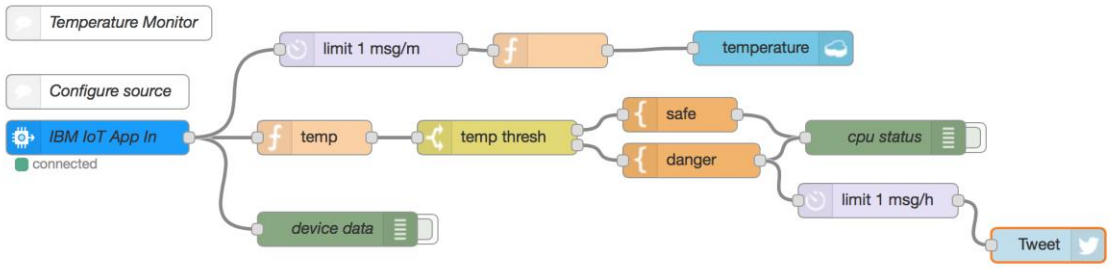

```
return {
    payload : {
        time : new Date().getTime(),
        temp: msg.payload.d.temp
    }
};
```

3. Add a 

node as shown below.



4. Connect the nodes together as shown below.

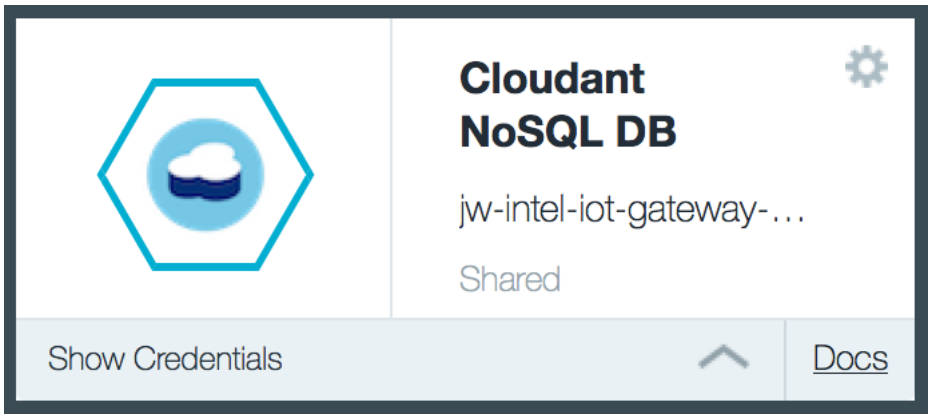


This flow is available at https://ibm.biz/BdrgQS

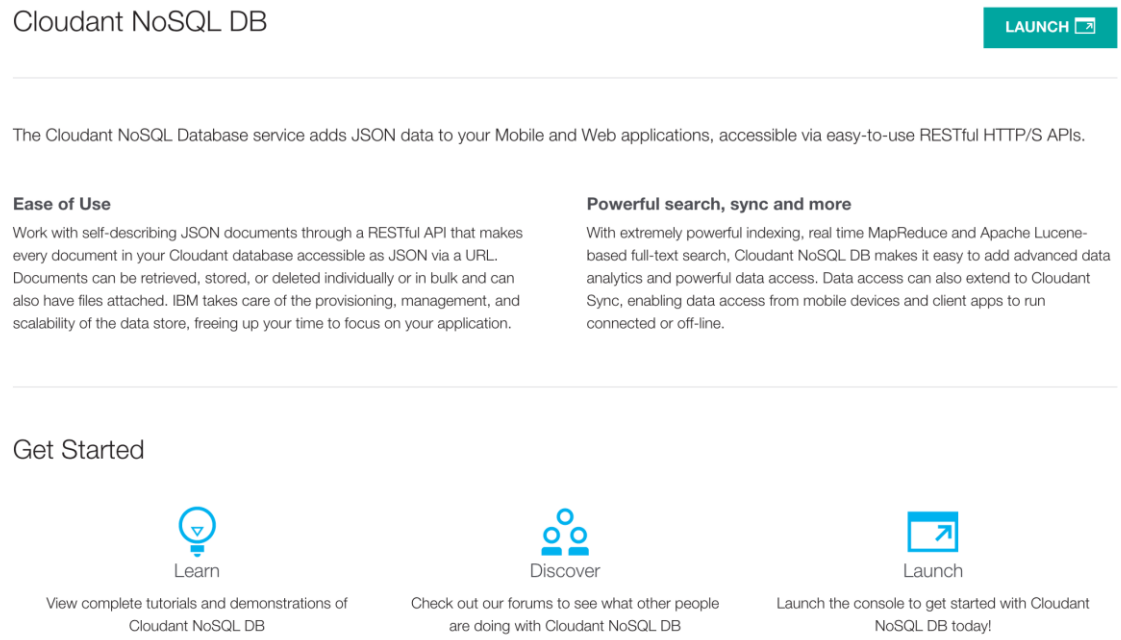
5. Click on 

to save and deploy your changes.

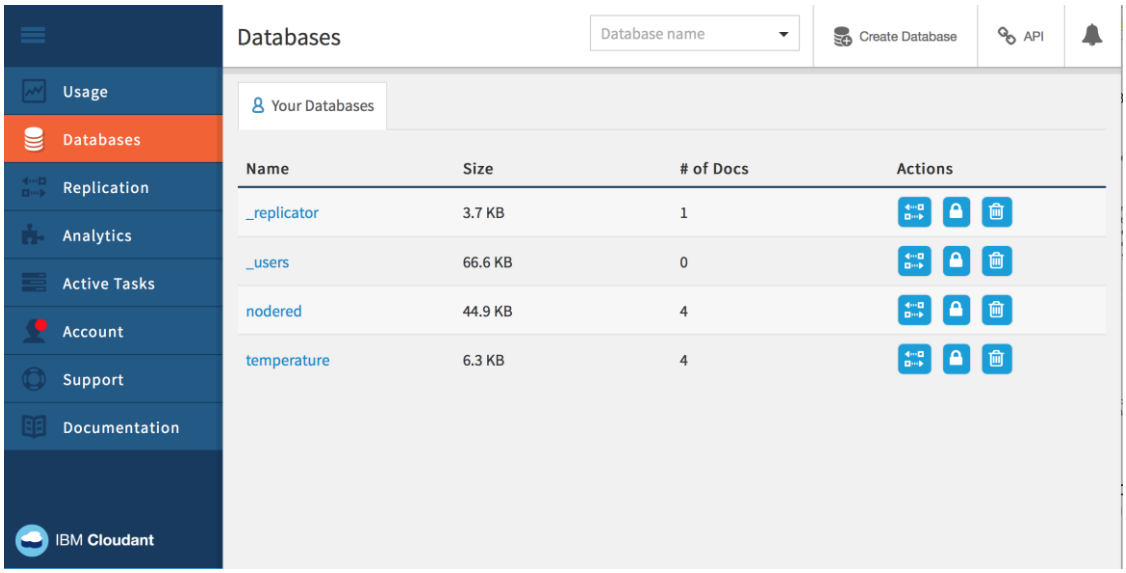
6. Go back to the IBM Cloud dashboard and the **Services** tab. Click on the **Cloudant NoSQL DB** service tile.



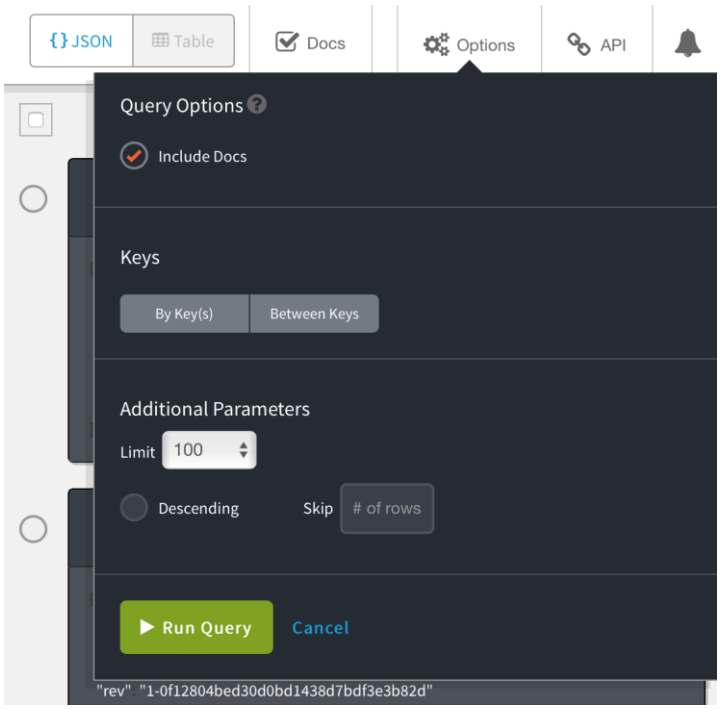
7. Click on the green **Launch** button.



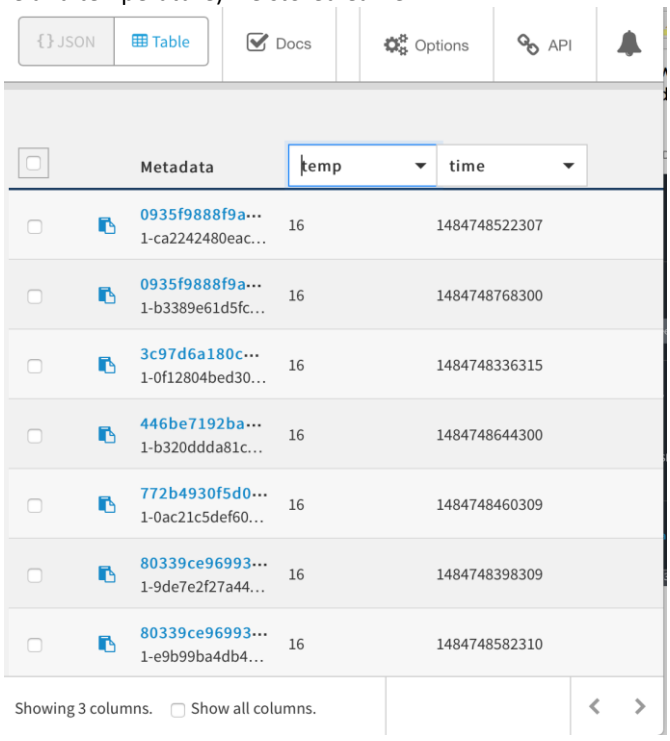
8. This is the Cloudant NoSQL database dashboard. Select the Databases section. A list of databases is displayed. The database named temperature contains documents representing each temperature event that has been stored by the Node-RED application. Click on the database named **temperature**



9. To see the expanded view of the documents, click on **Options** then on **Query Options**, check the box next to **Include Docs**, and click on the green **Run Query** button.



10. Select to show Table rather than JSON then each box represents one document (in our case one temperature event) that contains the payload (time and temperature) we stored earlier.



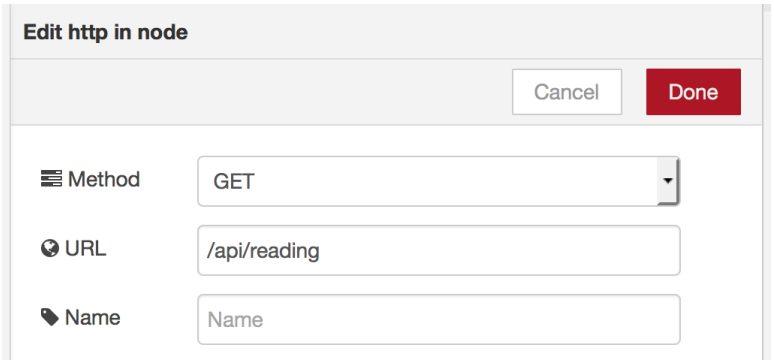
Retrieve Temperatures from Cloudant NoSQL DB

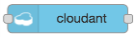
This section shows how to retrieve temperature data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time.

Now that we have a Cloudant NoSQL database containing reported temperatures, let's expose the data as an HTTP endpoint.

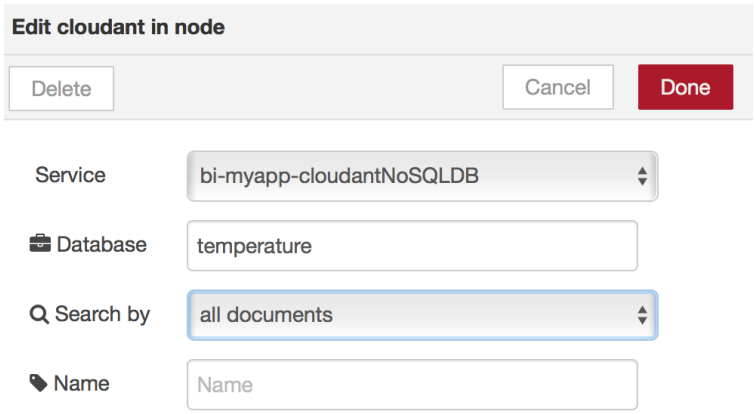
1. Add a 

node as shown below.



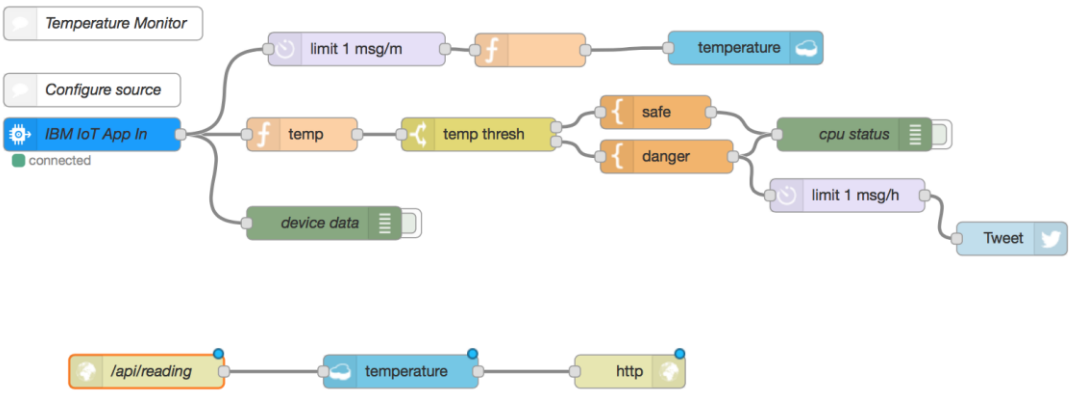
2. Add a 

node as show below.



3. Finally, add a 

node. Connect the nodes together as shown below.



4. Click on 

to save and deploy your changes.

5. Open a browser tab and visit your application's URL, appended by /api/reading. If you chose ***myapp*** when setting up your application, the URL would be:

   https://***myapp***.mybluemix.net/api/reading

   You should see data returned similar to the following:

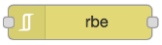

```
[{"_id":"0935f9888f9aa663f1b1cdd33debb8aa","_rev":"1-ca2242480eac26944ce814fa07ec5915","time":1484748522307,"temp":16},
{"_id":"0935f9888f9aa663f1b1cdd33df19e39","_rev":"1-b3389e61d5fc606bb3131d61e2053169","time":1484748768300,"temp":16},
{"_id":"0935f9888f9aa663f1b1cdd33df49669","_rev":"1-4b54f5939673ada224df354d7ce2ceba","time":1484748950299,"temp":16},
{"_id":"33f7647931b6975f75fd13df0514c8db","_rev":"1-dda39d7b9c676db26d3683aa4460fefa","time":1484749074287,"temp":16},
{"_id":"3c97d6a180c5bd3fdb67b24720066c83","_rev":"1-0f12804bed30d0bd1438d7bdf3e3b82d","time":1484748336315,"temp":16},
{"_id":"446be7192ba3827cdd2354371bc8aec4","_rev":"1-b320ddda81c5b33bbd1ae54a69dd0278","time":1484748644300,"temp":16},
{"_id":"446be7192ba3827cdd2354371bd1b5ea","_rev":"1-23db478edea013ee871c42dc4fc47c09","time":1484749012294,"temp":16},
{"_id":"772b4930f5d0d6e145e98e8701c3f339","_rev":"1-0ac21c5def60e15bbe3bf61e9d67c114","time":1484748460309,"temp":16},
{"_id":"80339ce96993b43c641236f8034e5b27","_rev":"1-9de7e2f27a4412576cb08d431703559a","time":1484748398309,"temp":16},
{"_id":"80339ce96993b43c641236f80352a31c","_rev":"1-e9b99ba4db44204dcb5f985657298ef9","time":1484748582310,"temp":16},
{"_id":"80339ce96993b43c641236f80355efac","_rev":"1-9afb1e80f59078267cde843af6d1fe2e","time":1484748706349,"temp":16},
{"_id":"80339ce96993b43c641236f80358c026","_rev":"1-57c92ea9bbcc04490f1705604b54644d","time":1484748830294,"temp":16},
{"_id":"80339ce96993b43c641236f8035f91e3","_rev":"1-7100b88805967828f724504ae0367c02","time":1484749136288,"temp":16},
{"_id":"a9580c6666a1b6b1a8f9c5a81b9281cd","_rev":"1-f990640ac31b9b1e2bb178a027e7cc63","time":1484748890297,"temp":16},
{"_id":"fcf69d4b229c009cf2581a6f43851223","_rev":"1-16482d6c9751eb4f9c900e6e55f82839","time":1484749190282,"temp":16}]
```

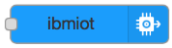

   The temperature values are returned in a JSON array. You can use this data in web applications or analytical tools. As you inject more data in the IoT example in Node-RED, this dataset will expand to include that data.

Send Alerts from Watson IoT to the Intel Gateway

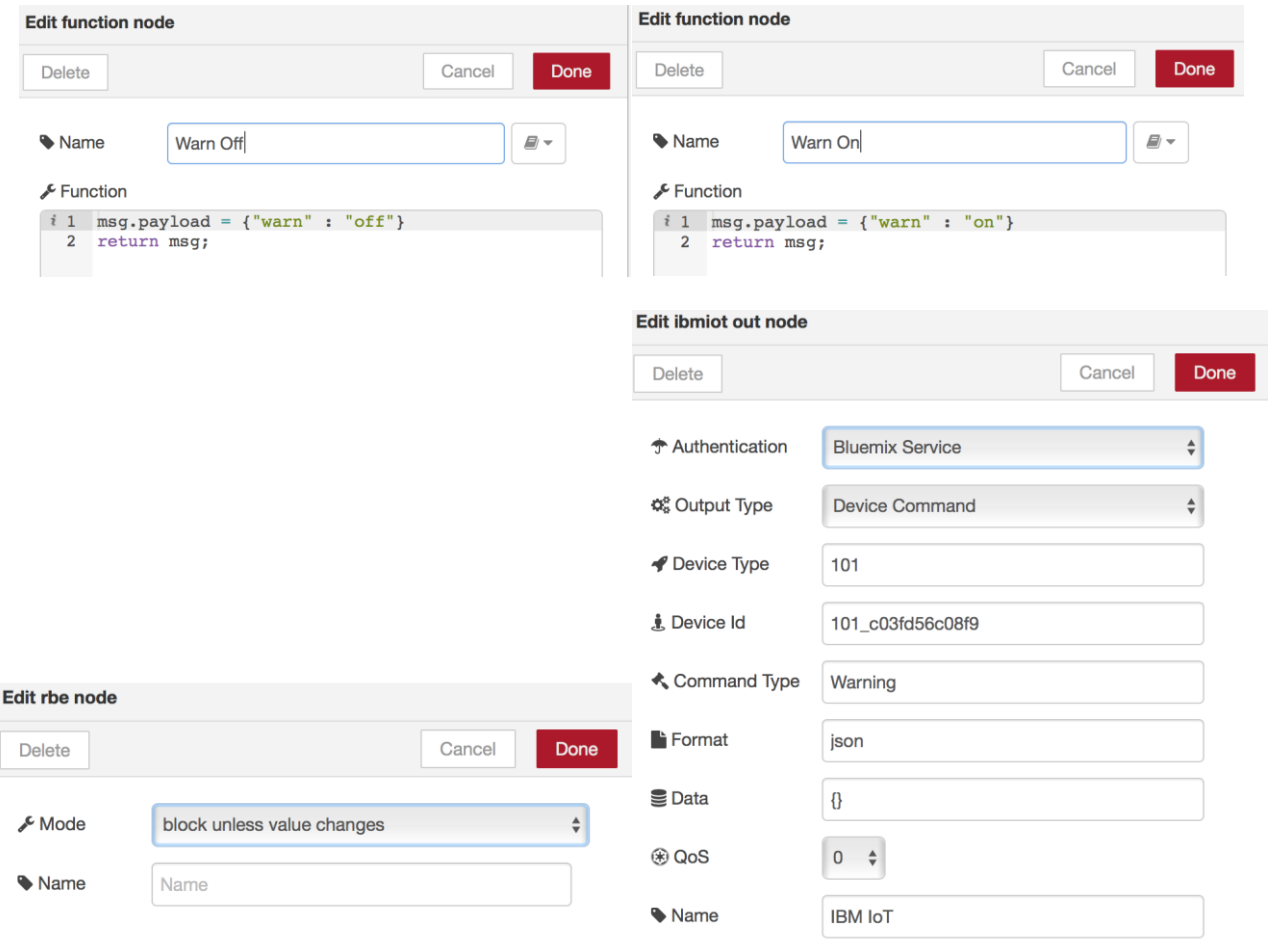
In this section, we'll extend the temperature sensor LCD to show a server generated alert. For this example we'll use the temperature data being sent from the Gateway.

1. In Node-RED running on IBM Cloud add 2 

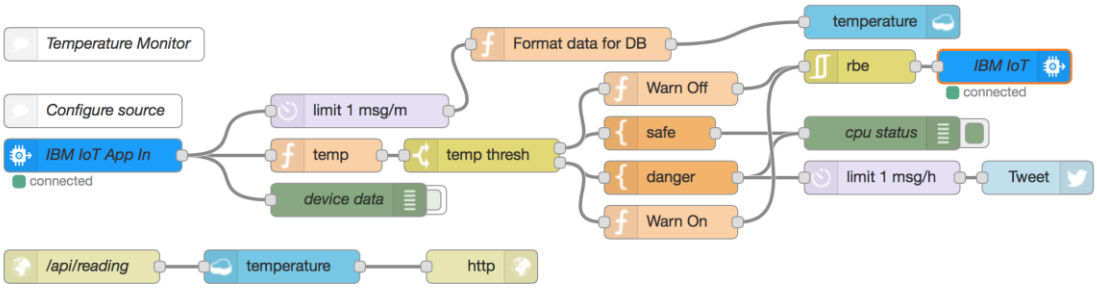
nodes, a 

node and a 

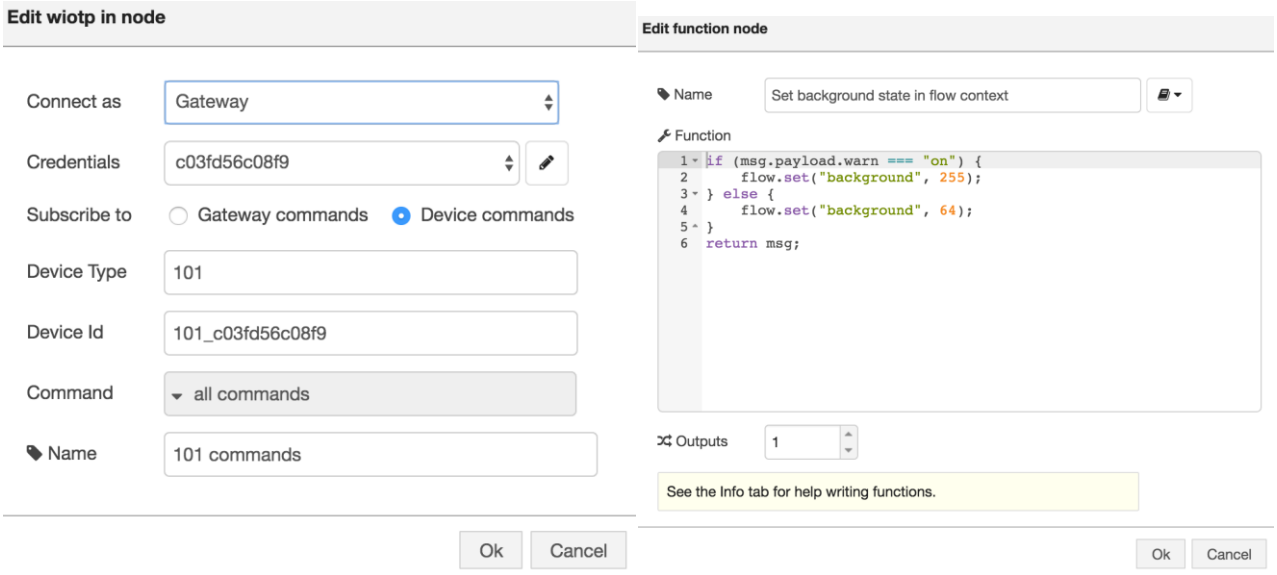
and configure them as follows:



replacing the Device Type and Device Id to match your configuration. Then connect as shown

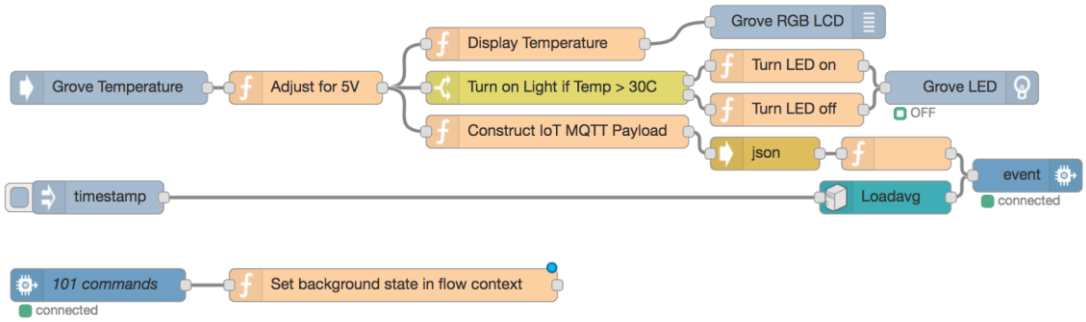


2. In Node-RED running on the Gateway add a  node and a  node and configure as shown:




```
if (msg.payload.warn === "on") {
    flow.set("background", 255);
} else {
    flow.set("background", 64);
}
return msg;
```

ensure the Device Type and Device Id match your 101 device configuration registered in the Watson IoT Platform. Connect the nodes as shown



3. Modify the code in the Display Temperature node to pick up the background color:

```
var r = flow.get("background") || 64;
return { payload : 'Temp : ' + msg.payload + 'C',
        "lcdColor" : {r : r, g : 64, b : 64}};
```

Now when the temperature rises above the trigger temperature set in the Node-RED flow running on IBM Cloud the background color of the LCD display will turn red. Modify the trigger temperature and test by holding or breathing on the temperature sensor to make it rise above the trigger temperature.

Useful links

IBM Cloud : http://bluemix.net/

Node-RED : http://nodered.org
MQTT : http://mqtt.org

Watson IoT Platform : http://www.ibm.com/internet-of-things/roles/iot-developer/
Watson IoT Platform developer docs : https://console.eu-gb.bluemix.net/docs/services/IoT/developer_doc_overview.html

IoT Recipes : https://developer.ibm.com/recipes/tutorials/category/internet-of-things-iot/