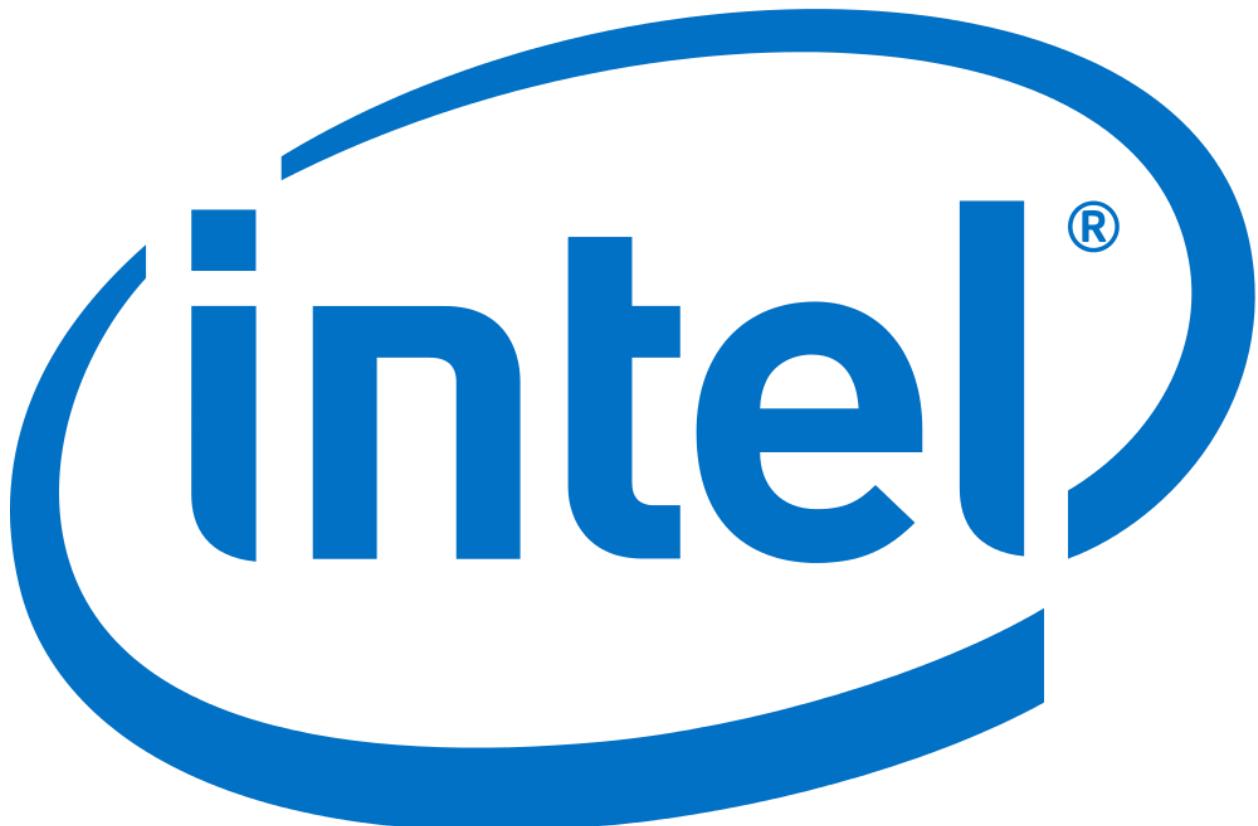


Intel IoT Gateway / Arduino 101 with IBM Watson IoT

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	3
Intel NUC Developer Hub Overview	8
Starting Node-RED for Blinky LED.....	14
Adding Sensors and Processing to the Gateway	17
Share Node-RED flows across workspaces.....	24
Add Watson IoT Nodes and OS Monitoring	29
Debugging techniques	30
Create Temperature Flow	33
What Next?	36
Creating an IoT application in IBM Cloud.....	37
Connect to Twitter and Tweet High Temperature	45
Create a Node-RED flow on Intel NUC	47
Sending Temperature Sensor to IoT Platform	52
Moving to secured communications with registered devices	54
Store Temperature in to a Cloudant NoSQL Database.....	64
Retrieve Temperatures from Cloudant NoSQL DB	68
Send Alerts from Watson IoT to the Intel Gateway.....	70
Useful links.....	71

Getting Started with Grove IoT Commercial Developer Kit

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

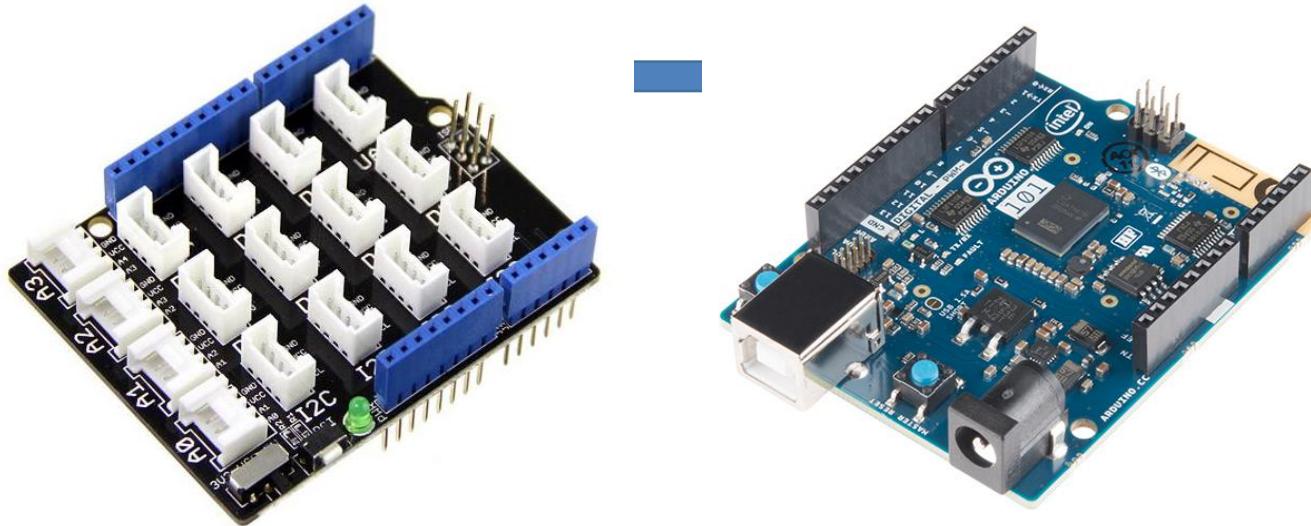


Arduino 101 is not included in Grove IoT Commercial Development Kit and will be given separately.

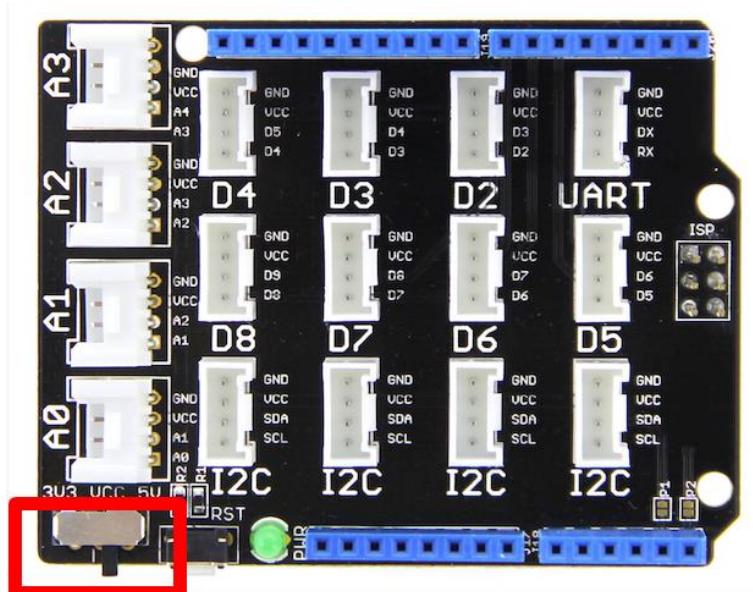


Step 2 – Connect Base shield to Arduino 101

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

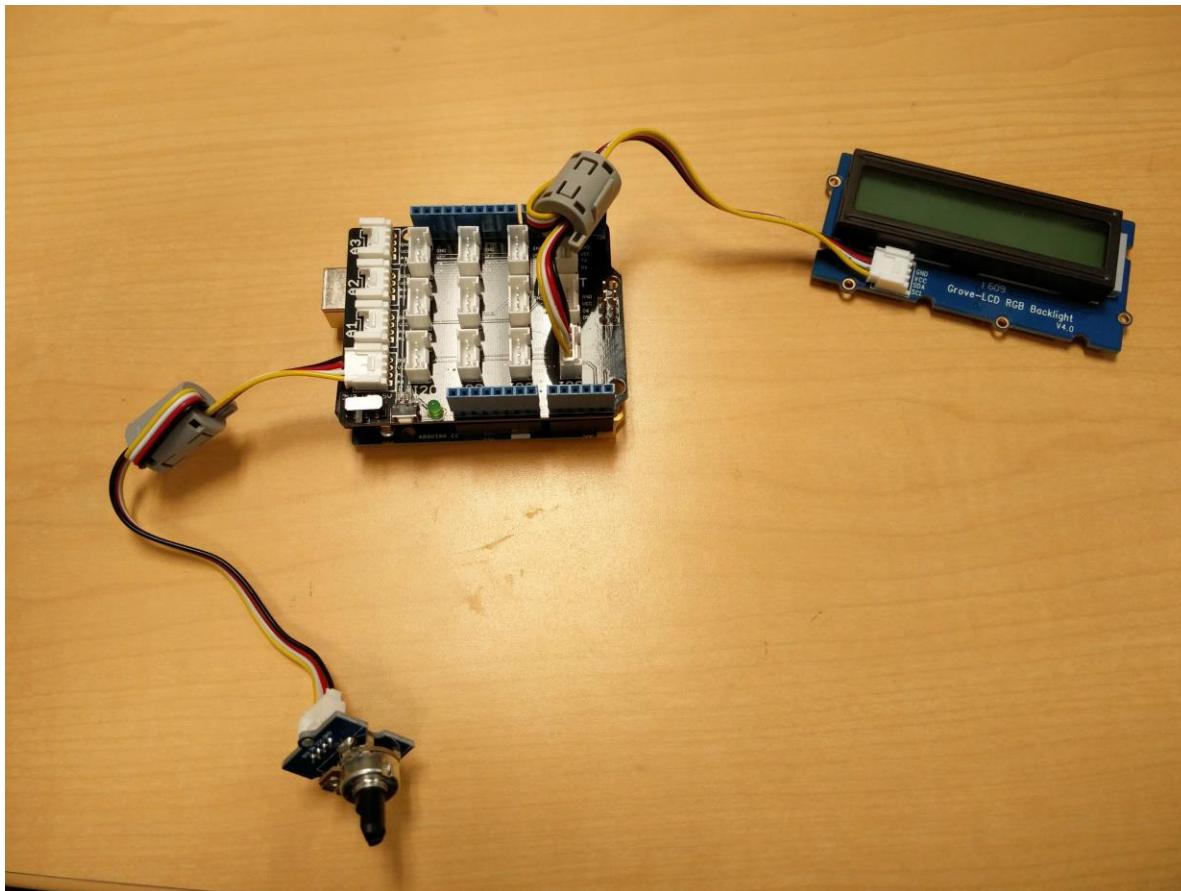
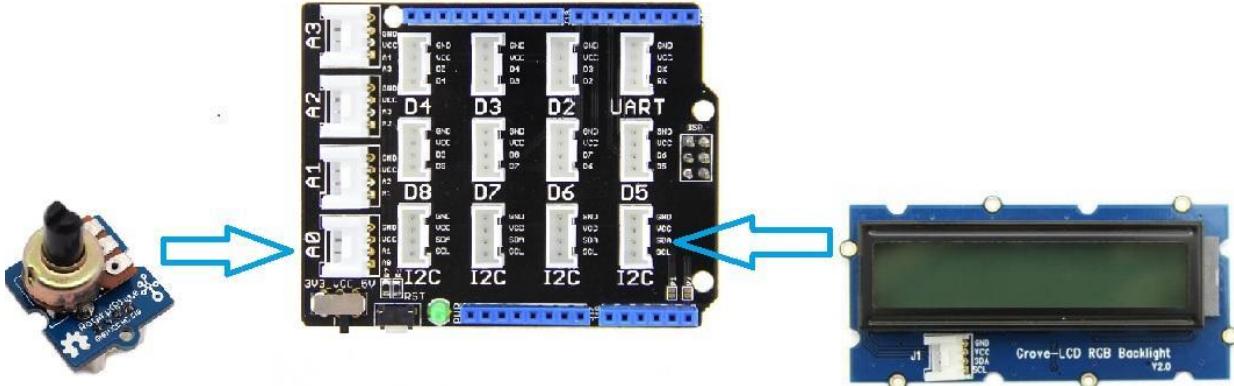


Step 3 – Move Base Shield switch towards 5 Volt



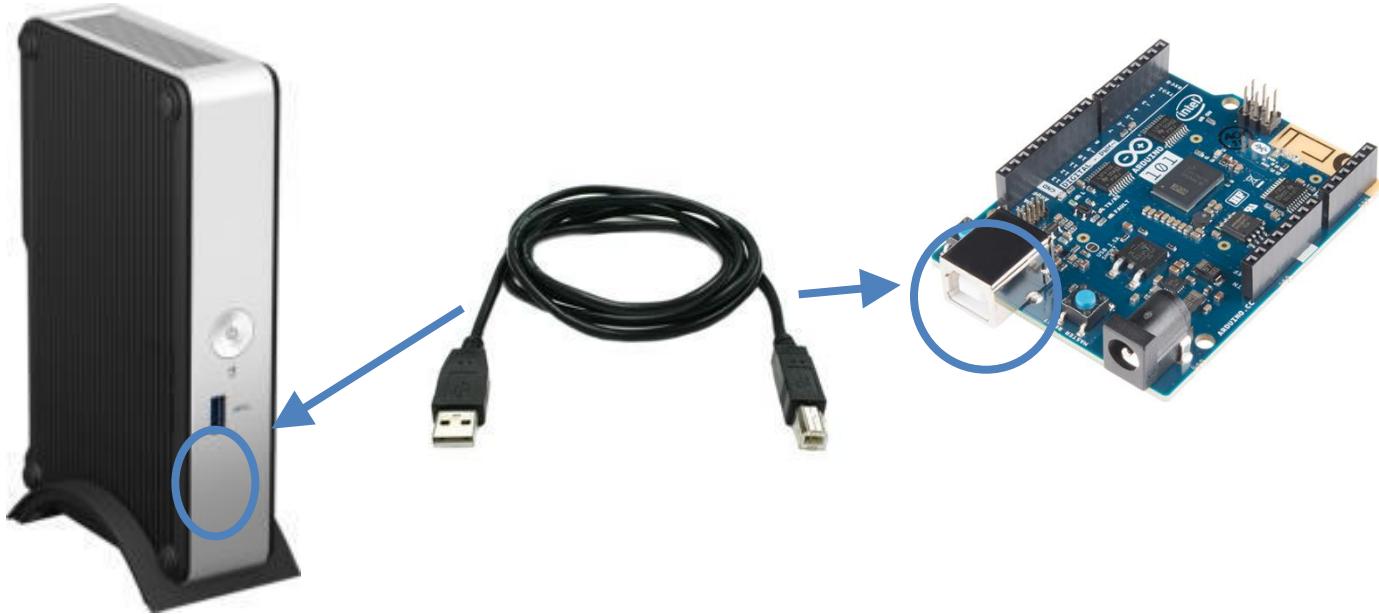
Step 4 – Connect LCD and Rotary Angle to Arduino 101

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.

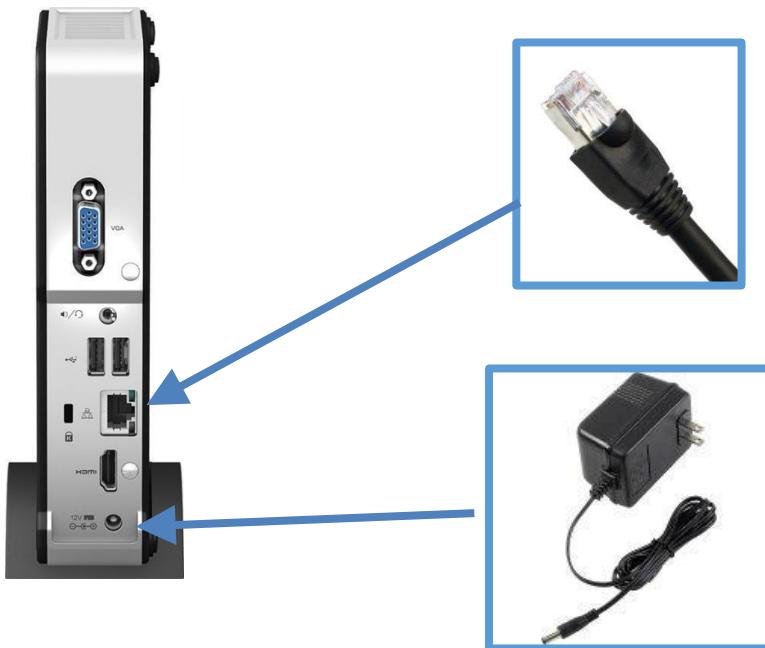


Step 5 – Connect Arduino 101 and Intel IoT Gateway via USB A-B Cable

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

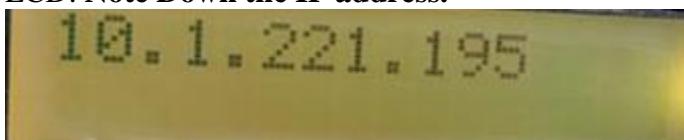


Step 6 – Connect Intel IoT gateway with Ethernet cable and Power adapter



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

The screenshot shows the Intel IoT Gateway Developer Hub dashboard. At the top left, it displays the host name "WR-IDP-7B41" and its connection status ("Connected"). Below this, there's a detailed system status section with the following information:

HOST NAME	WR-IDP-7B41
MODEL	Intel(R) Atom(TM) CPU E3815 @ 1.46GHz
VERSION	WR7.0.0.13
ETH0	192.168.1.136
WIFI SSID	No Wireless
TIME	Fri Aug 12 15:13:25 2016
UPTIME	0d 0h 17m
OS DRIVE	0.3j2.2
DEVHUB VERSION	1.0.1-1.0.2

To the right of the status section is a circular gauge chart titled "Rotary" with a value of 628. The chart has a green arc indicating the current value. Below the chart are two buttons: "Line" and "Gauge".

At the bottom of the dashboard, there are four navigation links: Sensors, Packages, Administration, and Documentation. The "Sensors" link is currently active.

Below the dashboard, there is a section titled "Connected Sensors" with a table showing the following data:

Sensor Name	Source	Unit of Measure	Value	Time
Rotary	Line	Raw	628	Aug 12, 2016 8:15:42 AM
Rotary	Gauge		628	Aug 12, 2016 8:15:42 AM

A "Manage Sensors +" button is located at the top right of this table.

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

The screenshot shows the Intel IoT Gateway Developer Hub dashboard. At the top right, there is a user profile for "Martin". Below the header, a navigation bar includes links for "Apps", "Tasks | Trello", "SSG IoT Developer...", "IoT - 18 How-To Int...", "Predix Dev Guides |...", and "Sheets". The main content area features a blue header with the Intel logo and the text "IoT Gateway Developer Hub". On the left, a card displays device information for "WR-IDP-7B41":
HOST NAME WR-IDP-7B41
MODEL Intel(R) Atom(TM) CPU E3815 @ 1.46GHz
VERSION WR7.0.0.13
ETH0 192.168.1.136
WIFI SSID No Wireless
TIME Fri Aug 12 15:13:25 2016
UPTIME 0d 0h 32m
OS DRIVE 0.3/2.2
DEVHUB VERSION 1.0.1-r1.0.2
The card also indicates the device is "Connected". To the right of this card is a circular gauge chart titled "Rotary" with a value of 628. The chart has a scale from 0.0 to 1023.0. Below the chart are two buttons: "Line" and "Gauge". Further down the page, there are four navigation icons: "Sensors" (gear icon), "Packages" (disk icon), "Administration" (wrench icon), and "Documentation" (book icon). The "Packages" section is currently selected. At the top of this section, there are three buttons: "Install Updates" (with a download icon and the number 1427), "Add Repo +", and "Add Packages +". Below these buttons is a table titled "Installed Packages" with the following data:

Package Name	Category	Launch Capability	Update	Running	Auto Run	Activity State
acpid	base					
alsa-conf	libs/multimedia					
alsa-lib	libs/multimedia					

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

The screenshot shows a web browser window titled "Intel® IoT Gateway Developer" with the URL "192.168.1.136/#/dashboard/tools". The page is titled "Operating System" and displays information about the Intel® IoT Gateway Software Suite, specifically WindRiver® Linux 7, 3.14 Kernel. It features four main buttons: "Install OS Updates" (with a download icon), "Upgrade to Pro" (with a download icon), "Restart OS" (blue button), and "Save OS Image" (blue button). Below these are two more sections: "Change Password" (blue button) and "Restore to Factory OS" (blue button). To the right of each button is a descriptive text block.

Intel® IoT Gateway Software Suite
WindRiver® Linux 7, 3.14 Kernel

Install OS Updates

Upgrade to Pro

Restart OS

To restart the system and apply all installed updates, select **Restart OS**.

Save OS Image

Save your OS image to a USB Flash Drive. This will save your image with additional security policies applied. For production deployments, an upgrade to Pro is recommended. [Upgrade to Pro](#)

Change Password

Select **Change Password** to edit the password for pre-loaded Operating System user accounts.

Restore to Factory OS

To erase the drive and reinstall the OS from the embedded backup volume, select **Restore to Factory OS**. Be sure to save your work. For a secure restore of the OS, see the [Security Guide](#).

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!

Screenshot of the Intel IoT Gateway Developer web interface showing the 'Tools' section.

The page title is "Intel® IoT Gateway Developer" and the tab is "Untitled". The URL is "192.168.1.136/#/dashboard/tools".

A blue button labeled "Restore to Factory OS" is present. A text block explains: "To erase the drive and reinstall the OS from the embedded backup volume, select **Restore to Factory OS**. Be sure to save your work. For a secure restore of the OS, see the [Security Guide](#)".

Quick Tools

Four quick tools are listed:

- Node-RED**: Represented by a red square icon with a white gear and plug. A "Launch" button is below it.
- APP CLOUD**: Represented by a blue cloud icon with three white gears inside. A "Launch" button is below it.
- Gears**: Represented by two interlocking black gears. A "Launch" button is below it.
- Terminal**: Represented by a blue terminal window icon with a white greater-than sign. A "Launch" button is below it.

Connection Details

Your connection on DevHub is currently not secured (<http://>). Enabling security (<https://>) will ensure your activity on DevHub is secure.

A blue "Enable" button is available to enable HTTPS.

Configure your proxy to manage your internet connection default access methods. This will ensure you are properly connected to the internet and able to use all DevHub features.

A blue "Configure" button is available to manage proxy settings.

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 screenshot shows a web browser window with the following details:

- Title Bar:** Intel® IoT Gateway Developer | http://www.inteliotmarketplace.com
- Address Bar:** 192.168.1.136/#/dashboard/documentation
- Toolbar:** Back, Forward, Home, Stop, Refresh, etc.
- Tab Bar:** Apps, Tasks | Trello, SSG IoT Developer..., IoT - 18 How-To Int..., Predix Dev Guides |..., Sheets
- User Profile:** Martin

The main content area displays the following sections:

- ## Tutorials

 - Setup a sensor in Node-Red**

How to setup an IoT Sensor using Node-Red

[View Video](#)

[View the tutorial](#)
 - Connect sensor output to cloud in Node-Red**

Building upon 1st tutorial, learn how to push Sensor data into Cloud Node-Red then display the result within Developer Hub charts.

[View Video](#)

[View the tutorial](#)
 - Learn to use Wind River® Helix™ App Cloud**

Create a Hello World app using a cloud development environment to build IoT apps and deploy onto the gateway.

[View Video](#)

[View the tutorial](#)
 - Save and Deploy OS Image**

Learn how to save gateway OS and apps for deployment onto another gateway.

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:

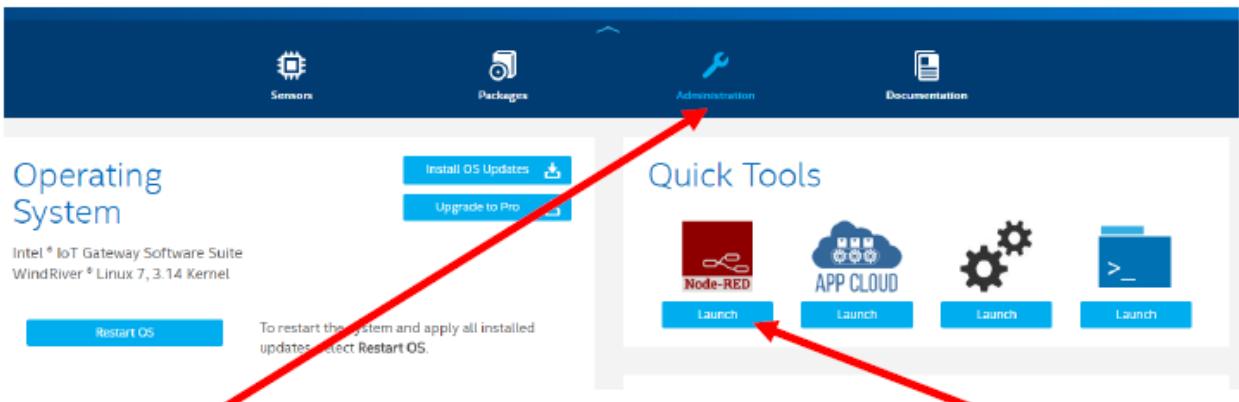
```
smart update  
smart upgrade node-red node-red-experience-devkit
```

Finally restart Node-RED by running the following command:

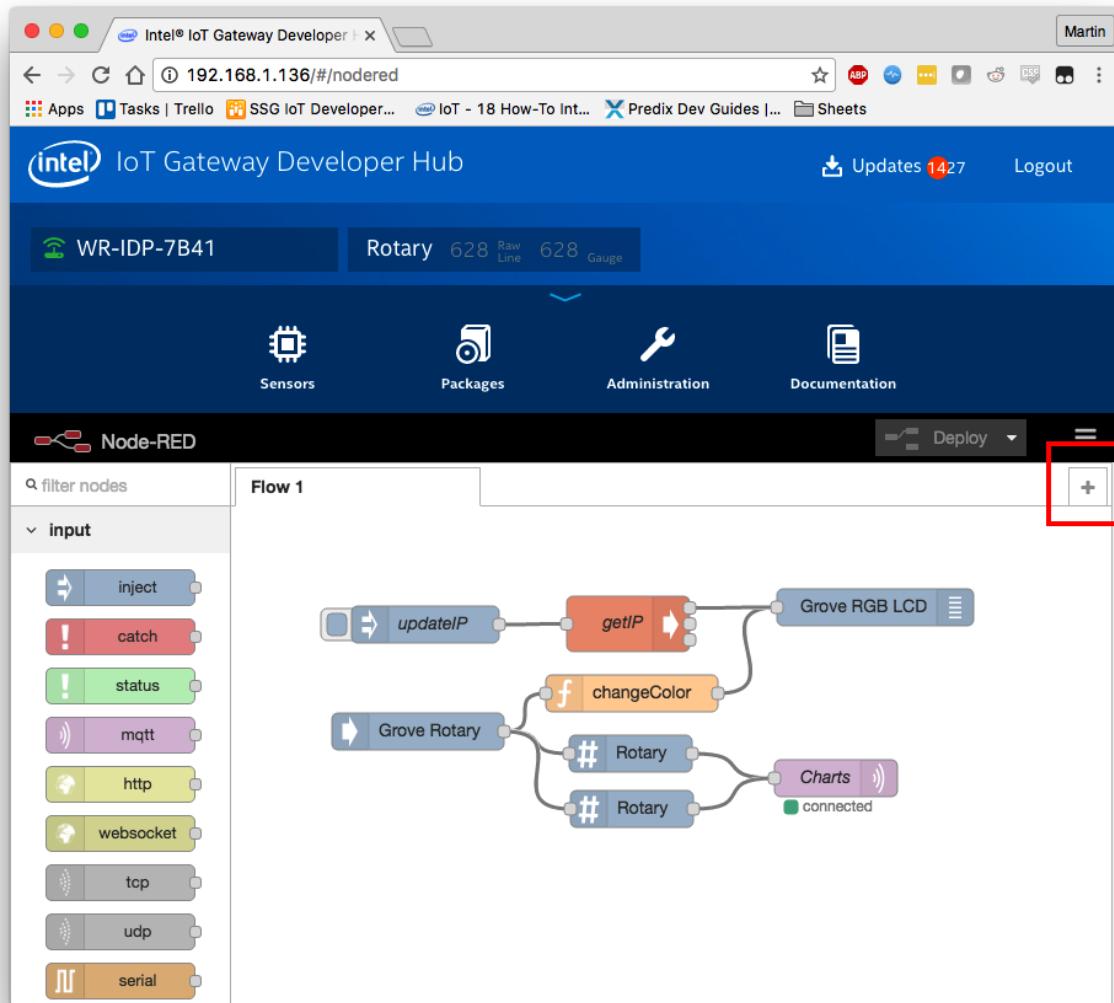
```
systemctl restart node-red-experience
```

Starting Node-RED for Blinky LED

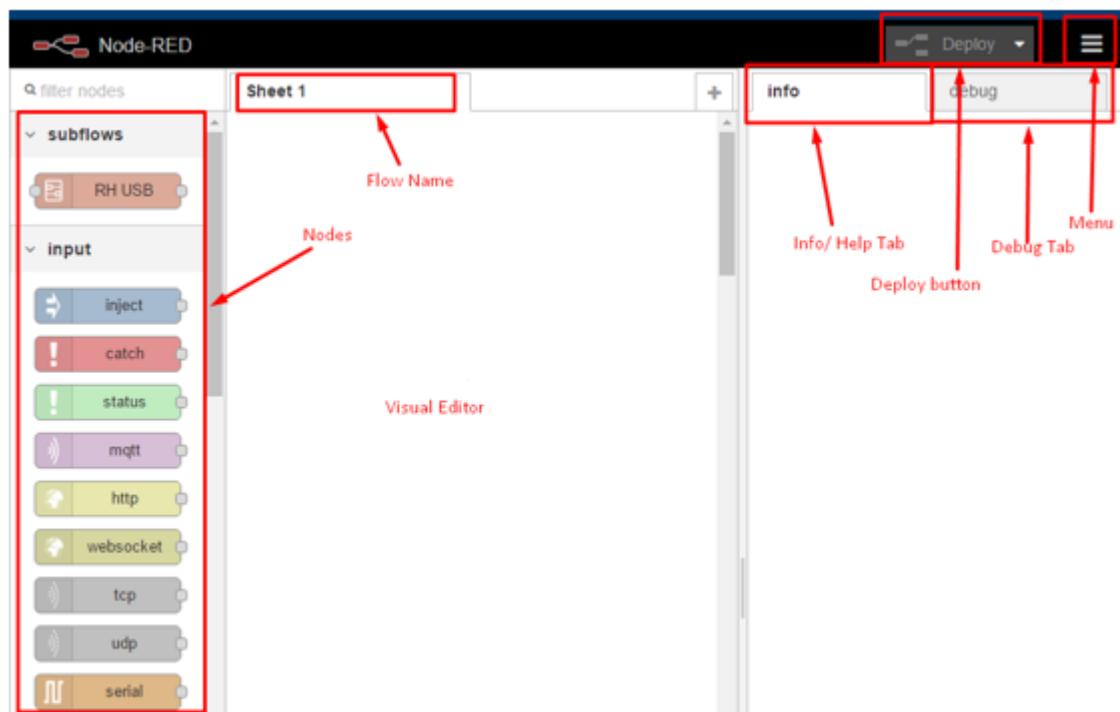
- Go to Development Hub by entering the IP Address assigned to your Gateway (E.g. <http://192.168.1.1>) in your browser.
- To load the Node-RED interface, click **Administration** on the navigation ribbon. Click **Launch** under the Node-RED icon.



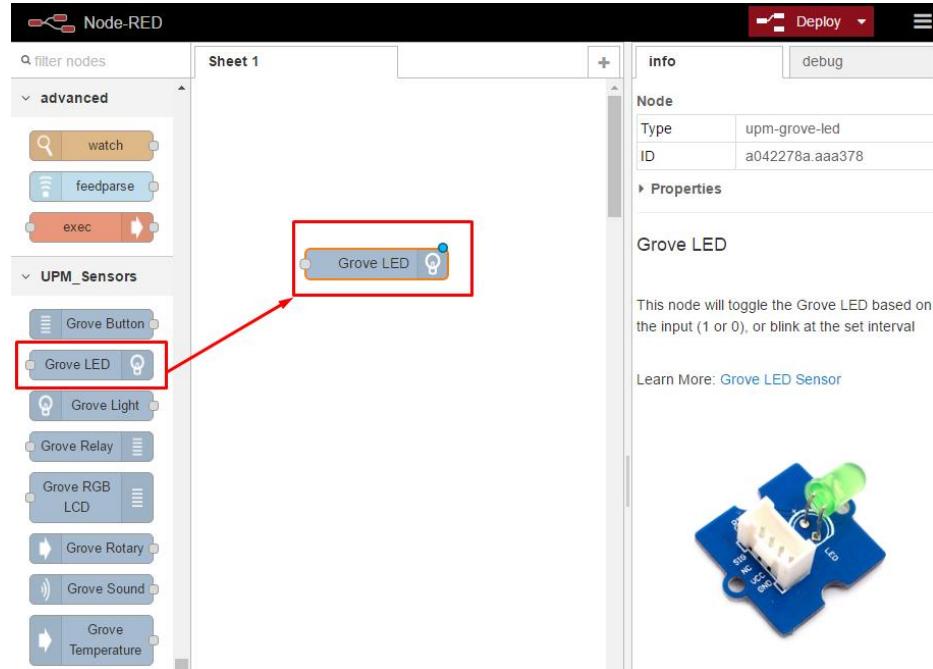
- Node-Red is a visual tool for writing the Internet of Things (IoT).
- Enter username and password as **root** and **root** respectively.



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.

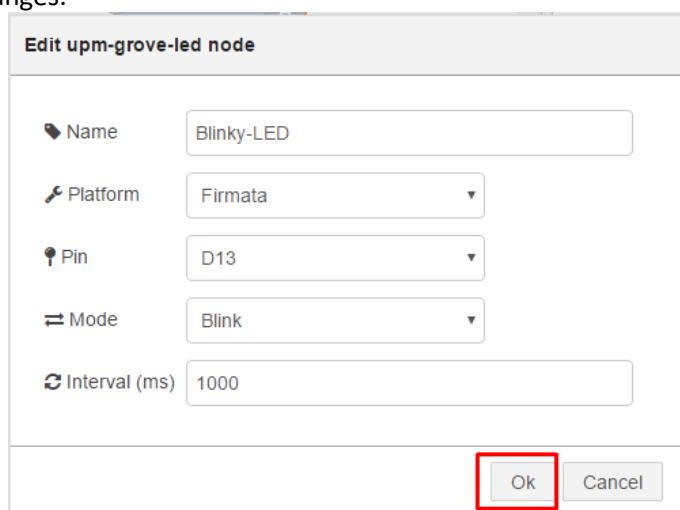


- Deploy Blinky LED Node. Go to Nodes, at the bottom of page there will be group on Nodes called UPM Sensors. Drag Grove LED from Nodes to Visual Editor



- 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

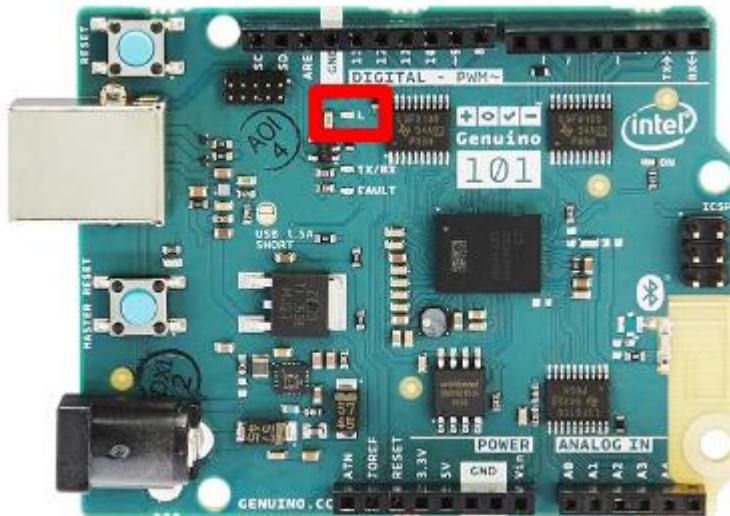
Then Click OK to save changes.



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

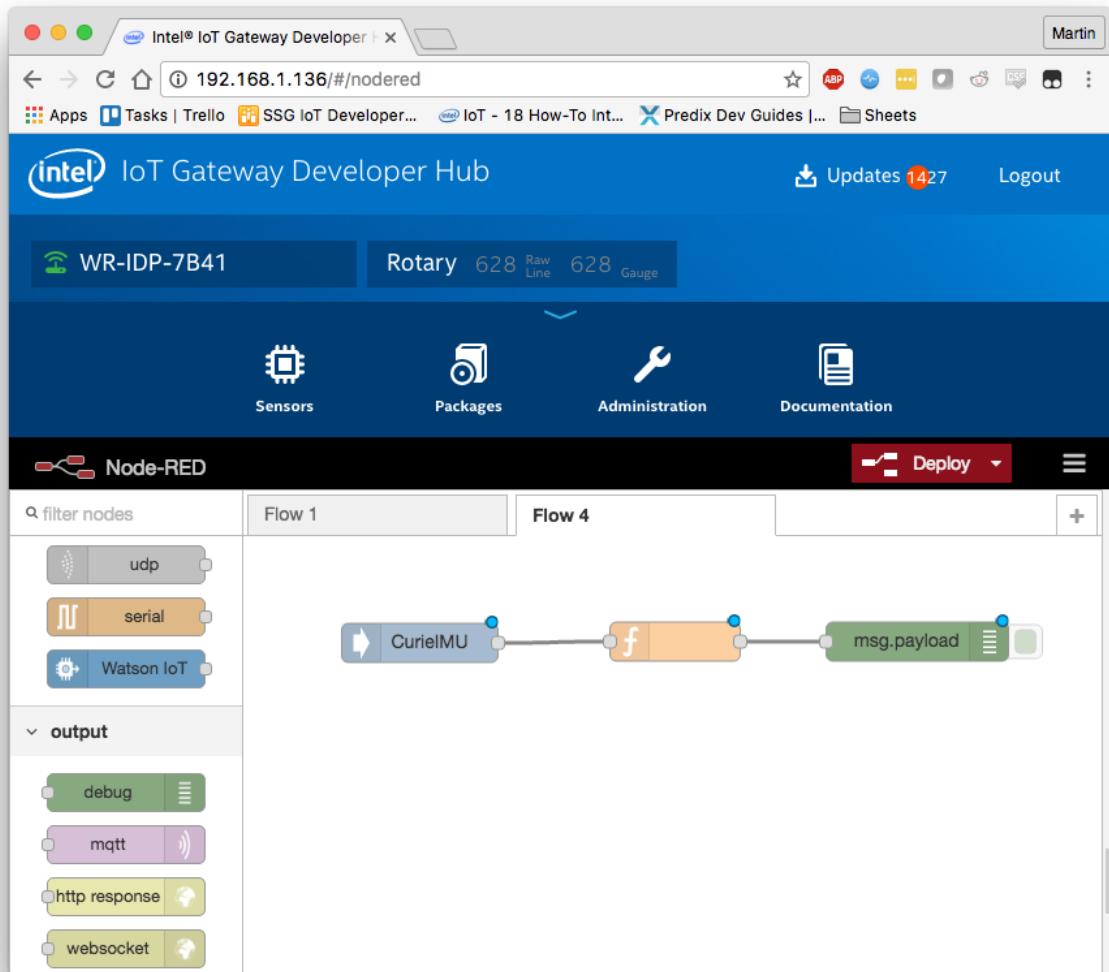
The screenshot shows the Node-RED interface. On the left, there's a palette with nodes categorized under 'advanced' and 'UPM_Sensors'. A 'Blinky-LED' node from the 'UPM_Sensors' category is selected and highlighted with an orange border. On the right, the 'info' tab displays the node's properties: Name is 'Blinky-LED', Type is 'upm-grove-led', and ID is 'a042278a.aaa378'. Below the properties, a description states: 'This node will toggle the Grove LED based on the input (1 or 0), or blink at the set interval'. A link to 'Learn More: Grove LED Sensor' is also present.

- The Arduino 101 LED on board LED will start blinking

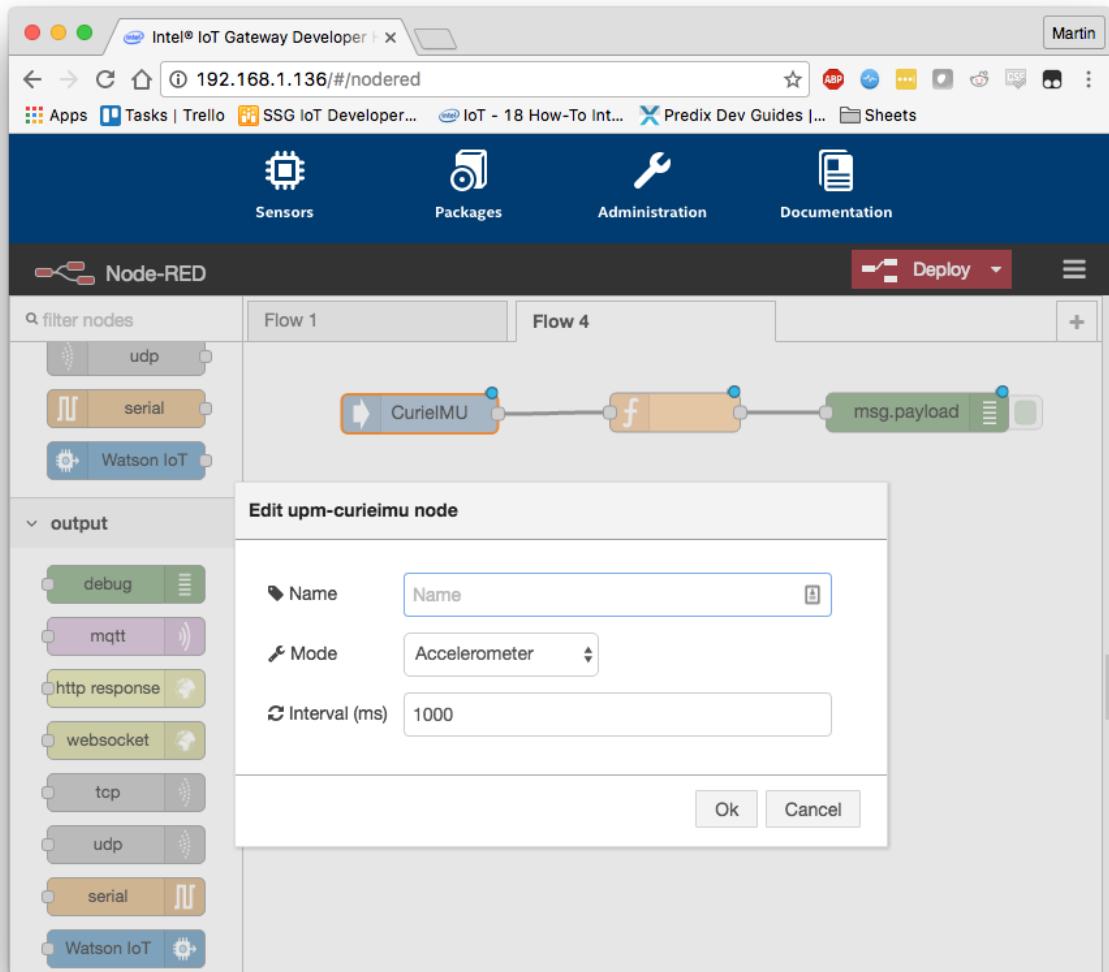


Adding Sensors and Processing to the Gateway

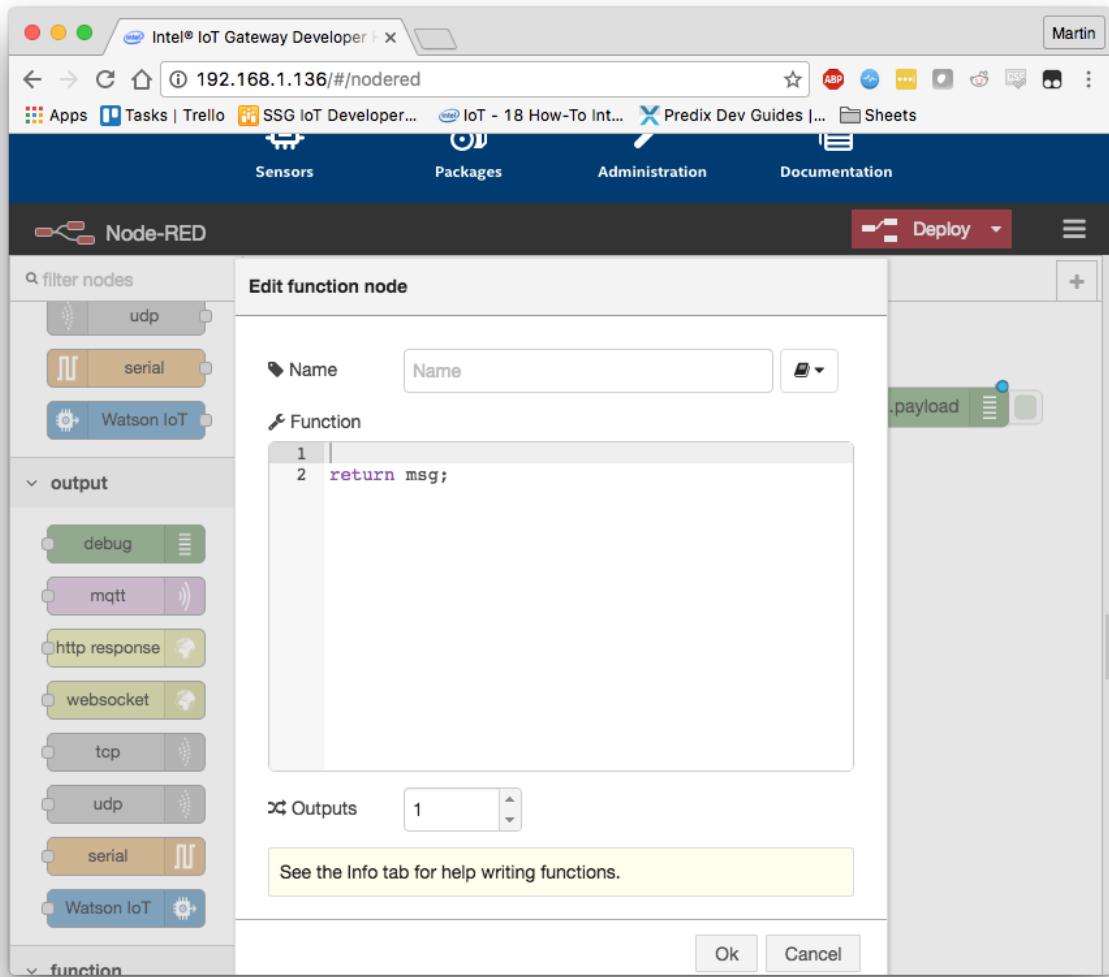
Now that you have a basic understanding of how to interact with the Gateway let's move on to adding some more sensor data and processing to the gateway.



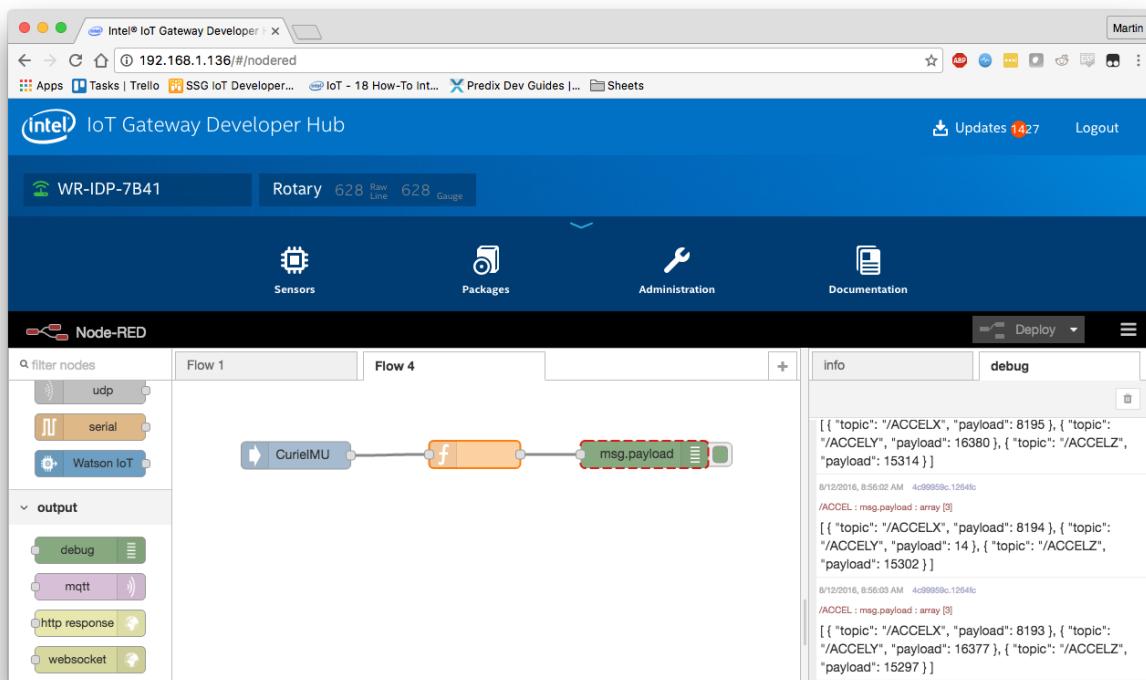
- From **UPM Sensors** add **CurieIMU** node
- From **function** add a **function** node
- From **output** add a **debug** node
- Wire the nodes together



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.

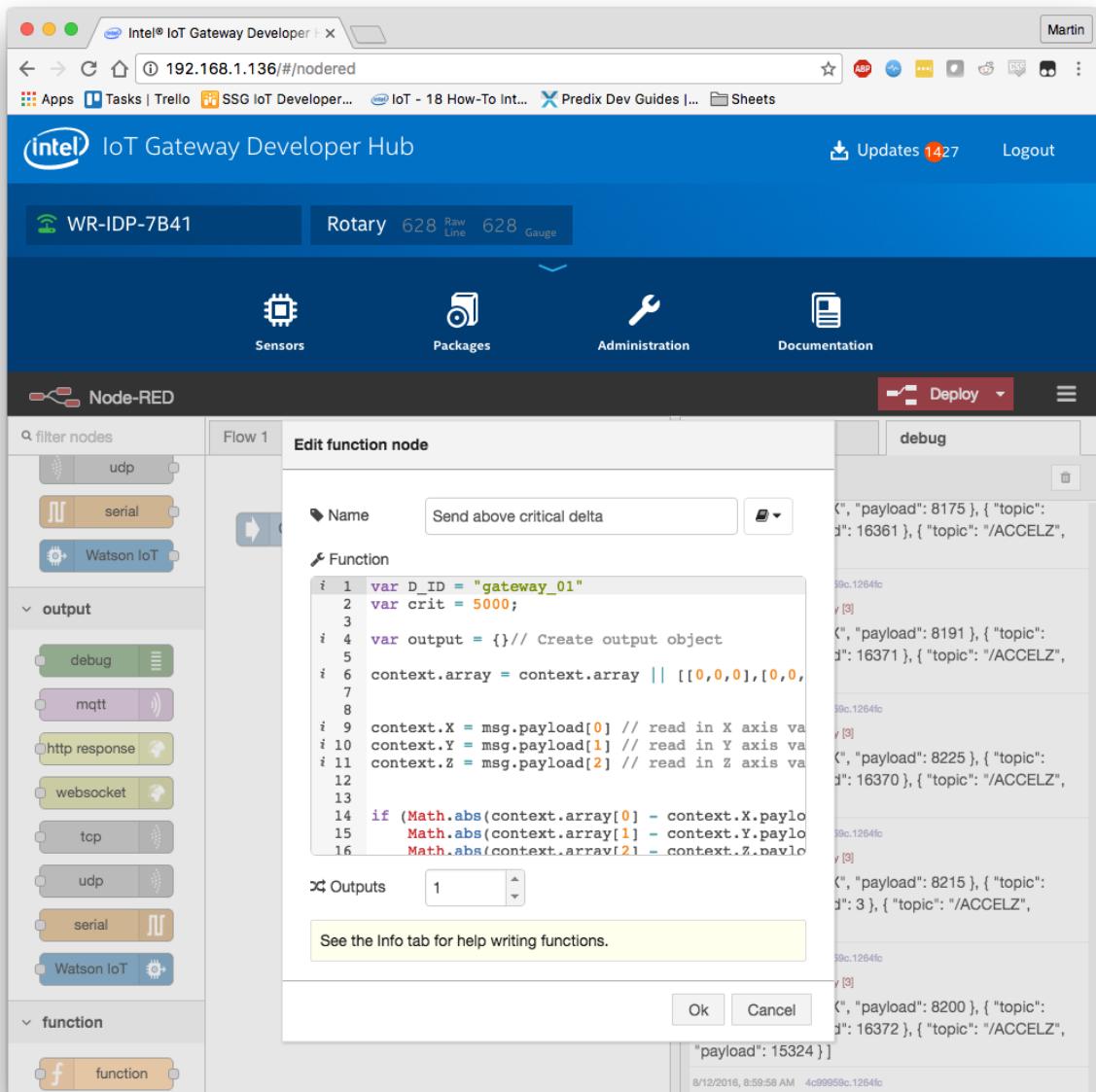


Hit deploy to run the flow. Open the debug tab to see the output from the function node.

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, context.Z.payload];
    return output;
}
else {

```

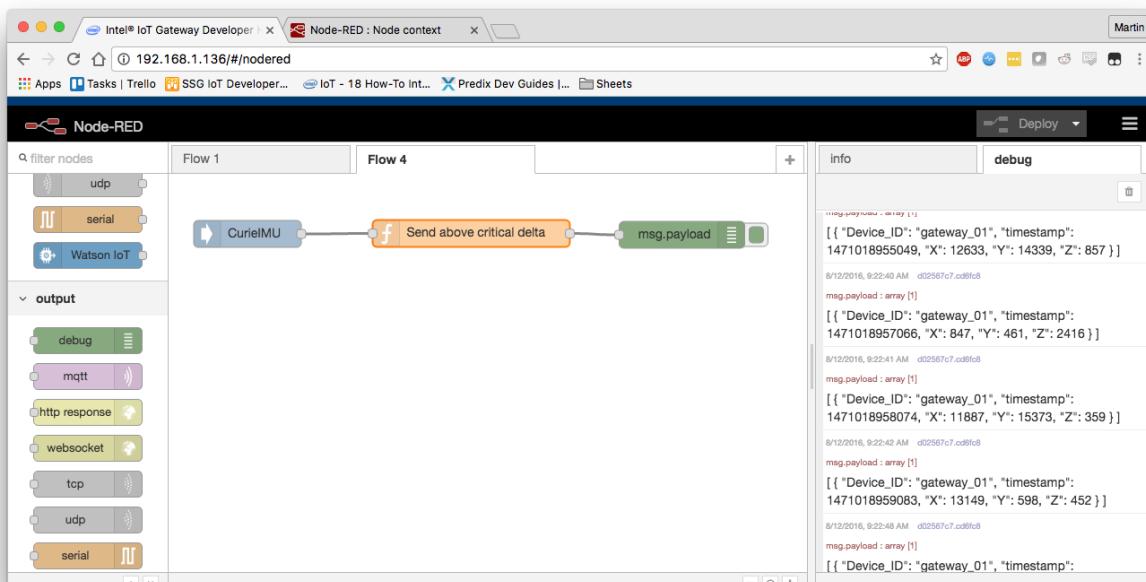
```

        context.array = [context.X.payload, context.Y.payload, context.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

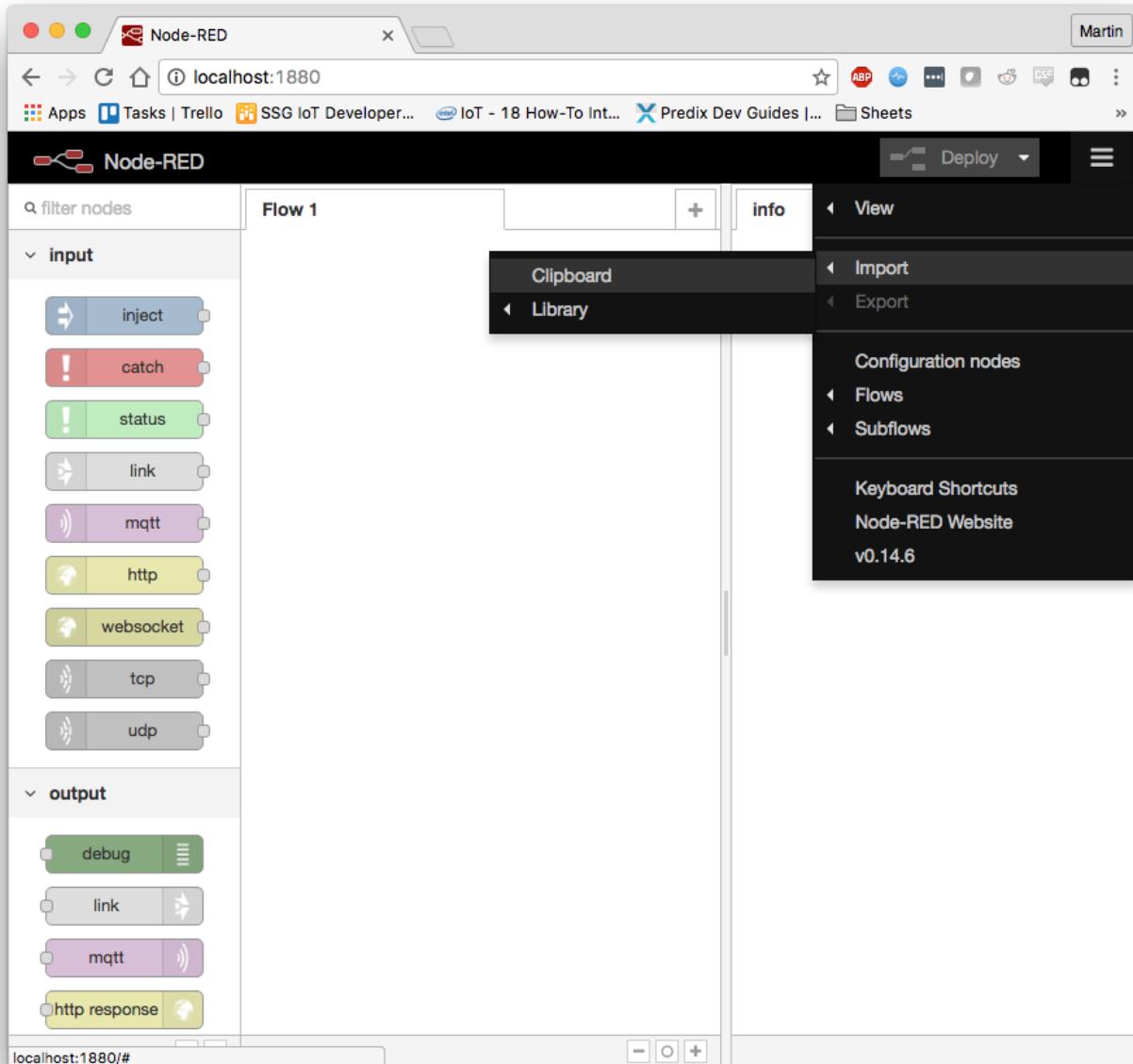
You can export and import entire flows (nodes and connections) out of and into your workspace.

Importing Flows:

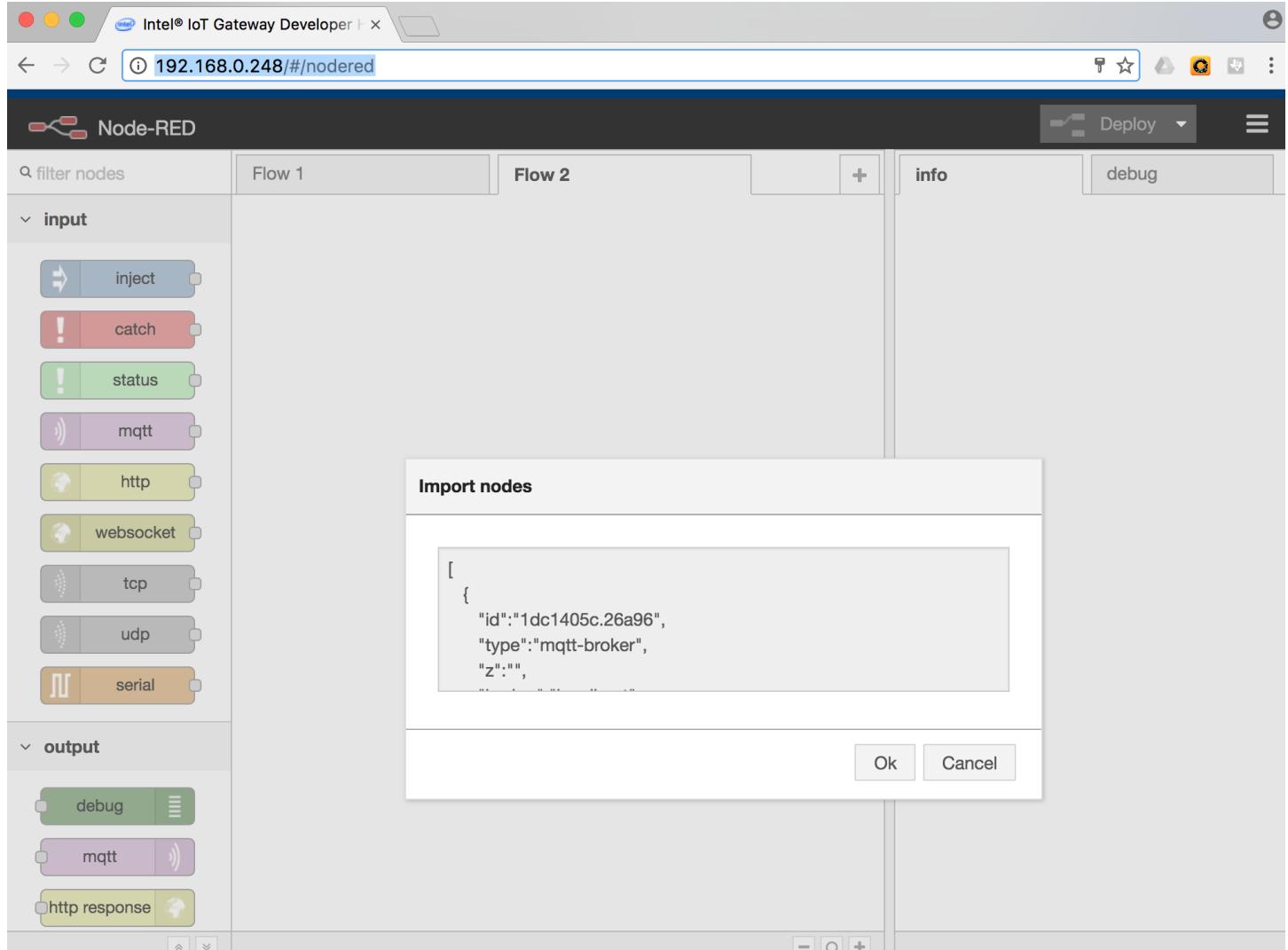
To import the above flow, go to:

<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:



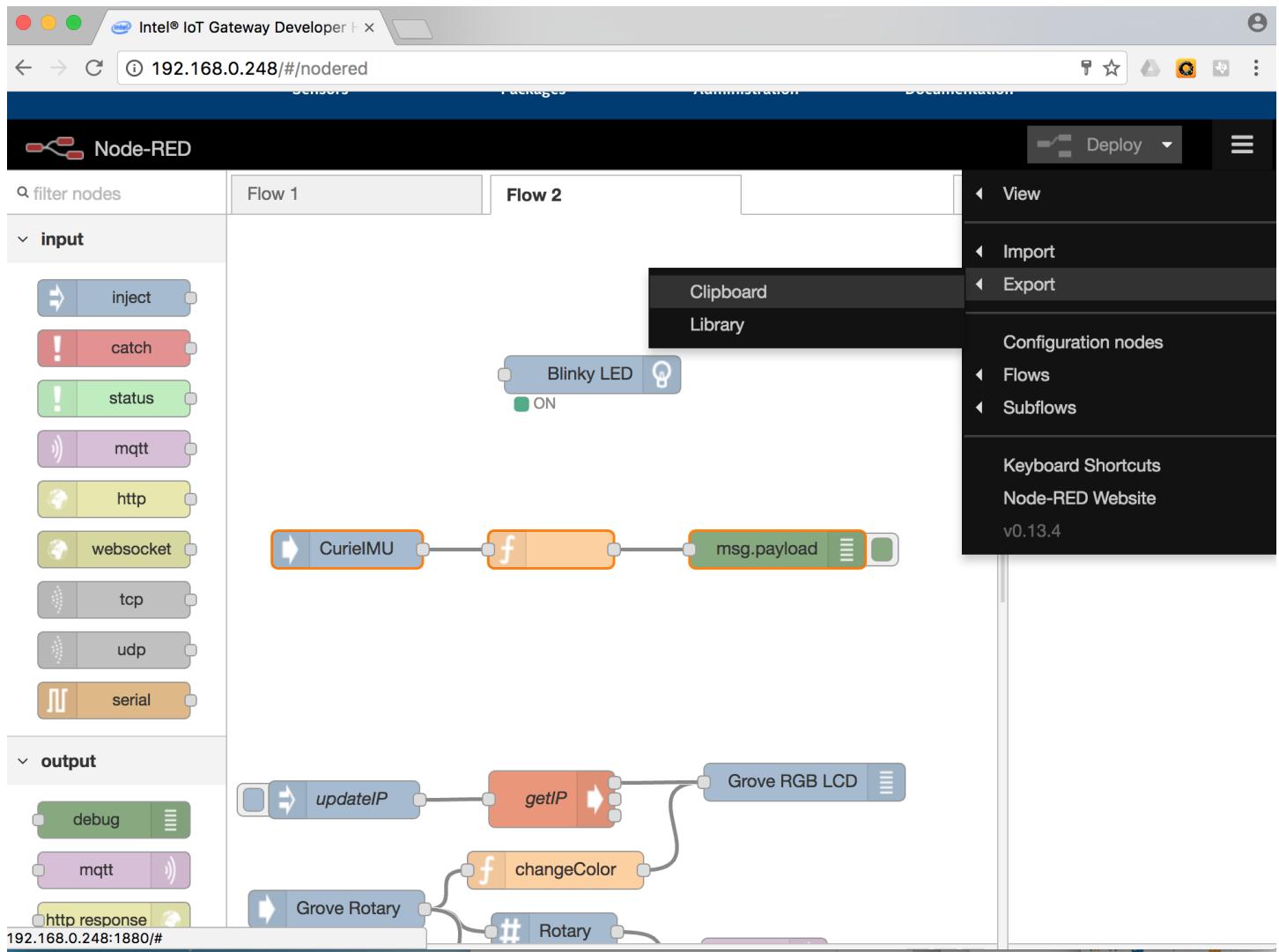
The imported flow can now be positioned using your mouse pointer. Click to place on the sheet

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

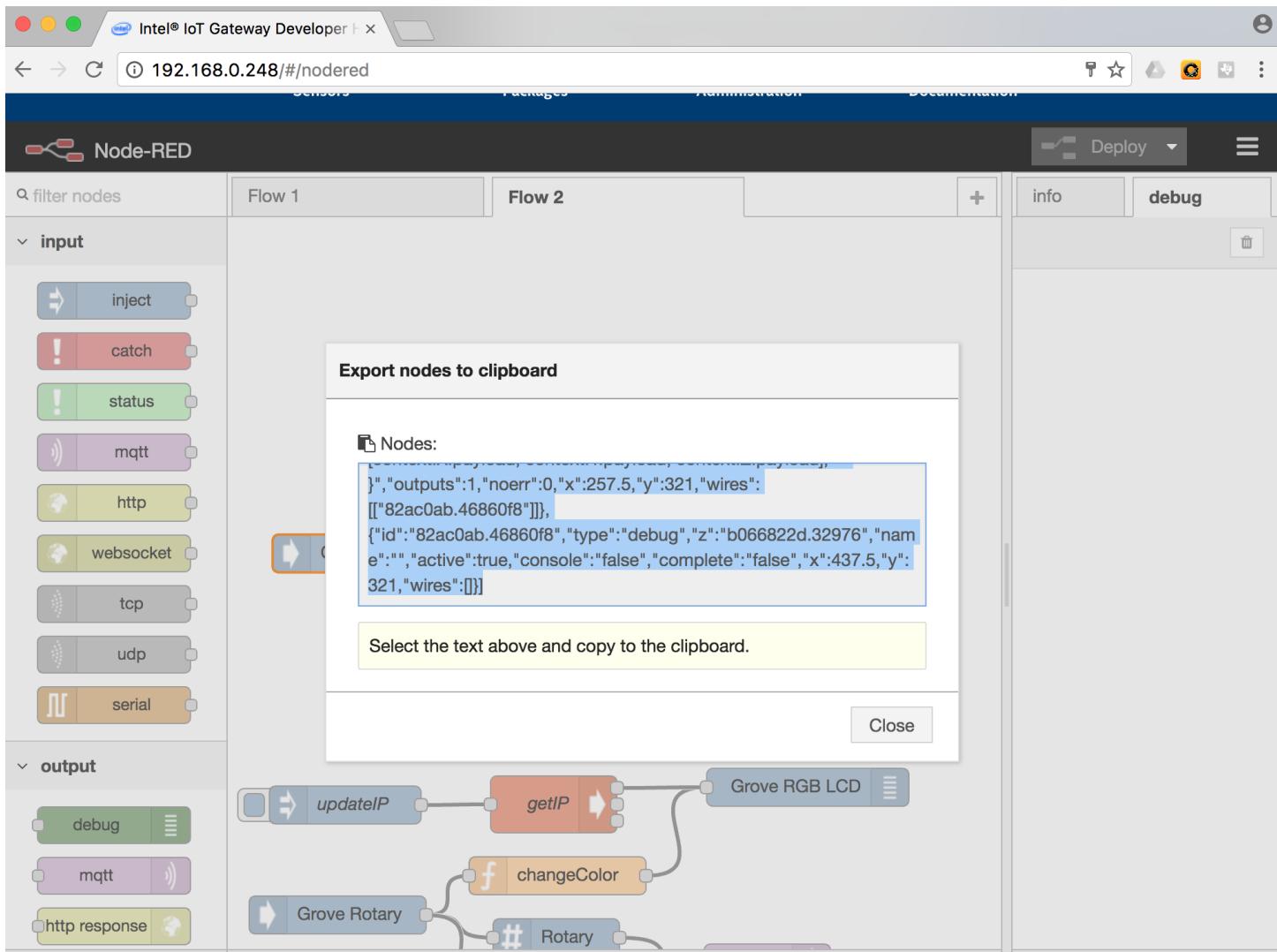
Follow the same procedure for importing anymore code you come across in this workshop.

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

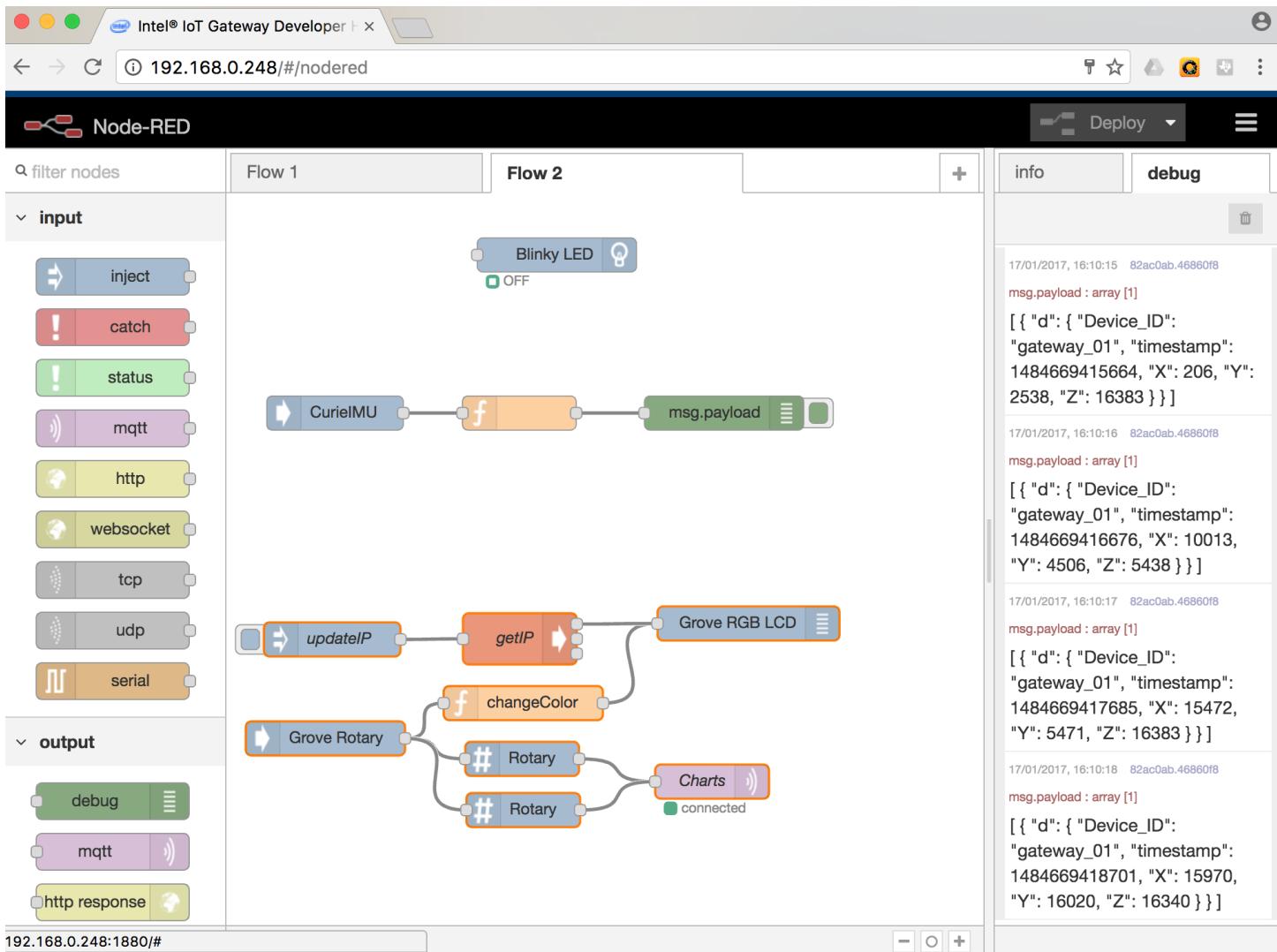


Copy the text in the window that opens. You can now paste this somewhere and share it with others. I would recommend running it through a JSON beautifier like: <https://jsonformatter.curiousconcept.com/> first if you want it to be readable.



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.



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:

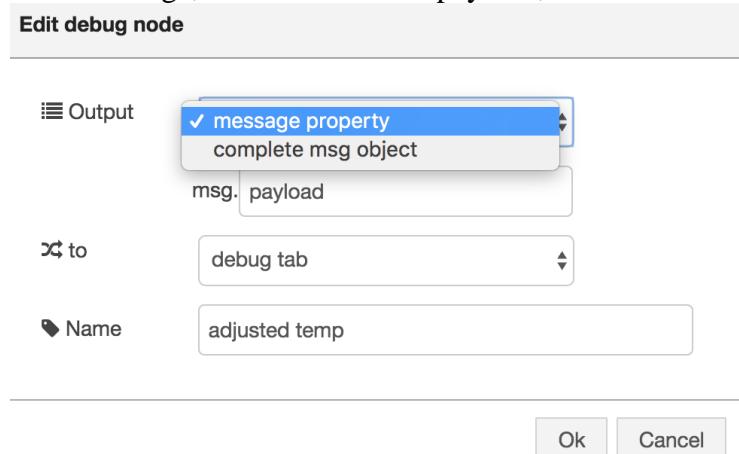
```
systemctl restart node-red-experience
```

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

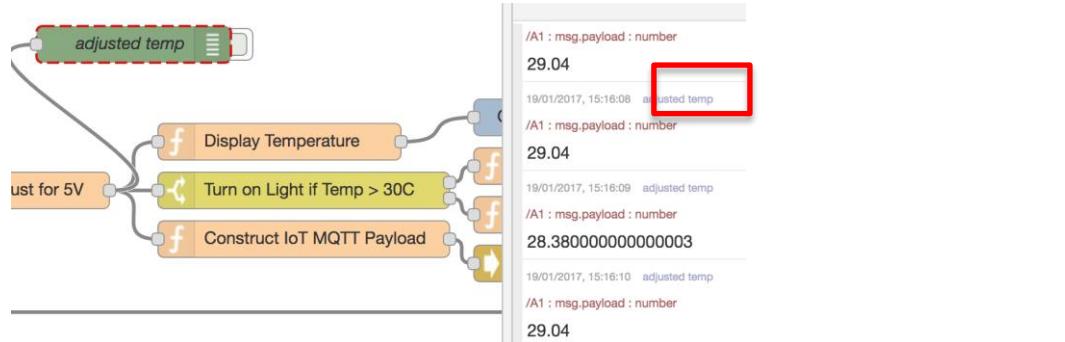
The Debug node allows you to see what data is flowing in a node. The debug node can be configured to show a specific property in the message, which defaults to payload, or can show the entire message content.



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

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.

```
19/01/2017, 15:22:56  node: c0c482df.3f3b8
iot-2/type/101/id/101_c03fd56c08f9/evt/event/fmt/json : msg :
Object
  ▾ object
    topic: "iot-
2/type/101/id/101_c03fd56c08f9/evt/even"
  ▾ payload: object
    ▾ d: object
      temp: 29.04
      deviceId: "101_c03fd56c08f9"
      deviceType: "101"
      eventType: "event"
      format: "json"
      _msgid: "85cd553a.7a32a8"
}

19/01/2017, 15:22:57  node: c0c482df.3f3b8
iot-2/type/101/id/101_c03fd56c08f9/evt/event/fmt/json : msg :
Object
  ▶ { topic: "iot-
2/type/101/id/101_c03fd56c08f9", payload:
object, deviceId: "101_c03fd56c08f9",
deviceType: "101", eventType: "event" ...
}
```

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");
```

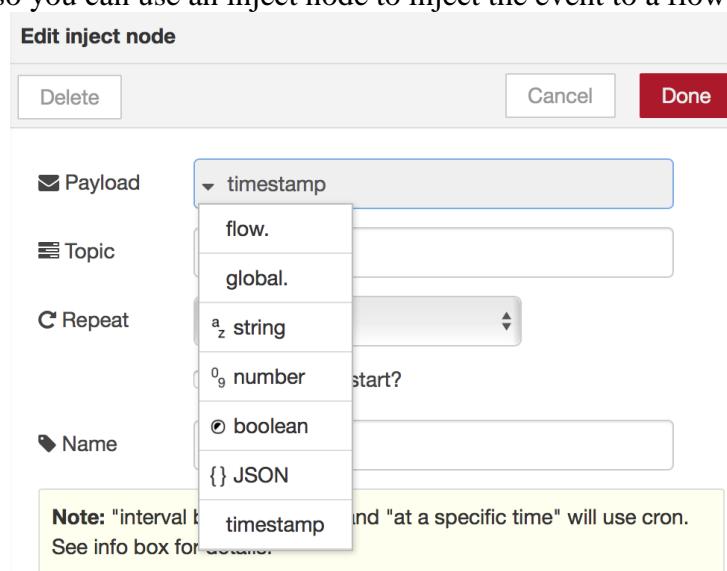
Log entries only go to the console (do not show in the debug panel). Warning and errors go to the console and debug panel. Warning entries have a yellow band at the side and errors have a red band.

```
19/01/2017, 15:30:31  node: 2fec9c4d.9cc164
function : (warn)
"Warning entry"

19/01/2017, 15:30:31  node: 2fec9c4d.9cc164
function : (error)
"Error entry"
```

Inject node

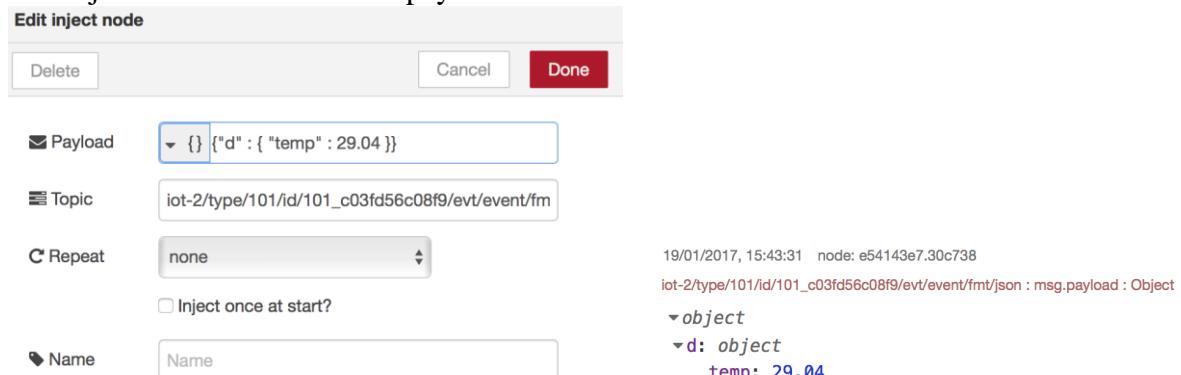
The inject node allow you to manually inject data into a flow. It is often difficult to get a real device to generate certain events, so you can use an inject node to inject the event to a flow.



So if a device is sending data into the platform:

```
▼ object
  topic: "iot-2/type/101/id/101_c03fd56c08f9/evt/event"
  ▼ payload: object
    ▼ d: object
      temp: 29.04
```

The inject node can create this payload:



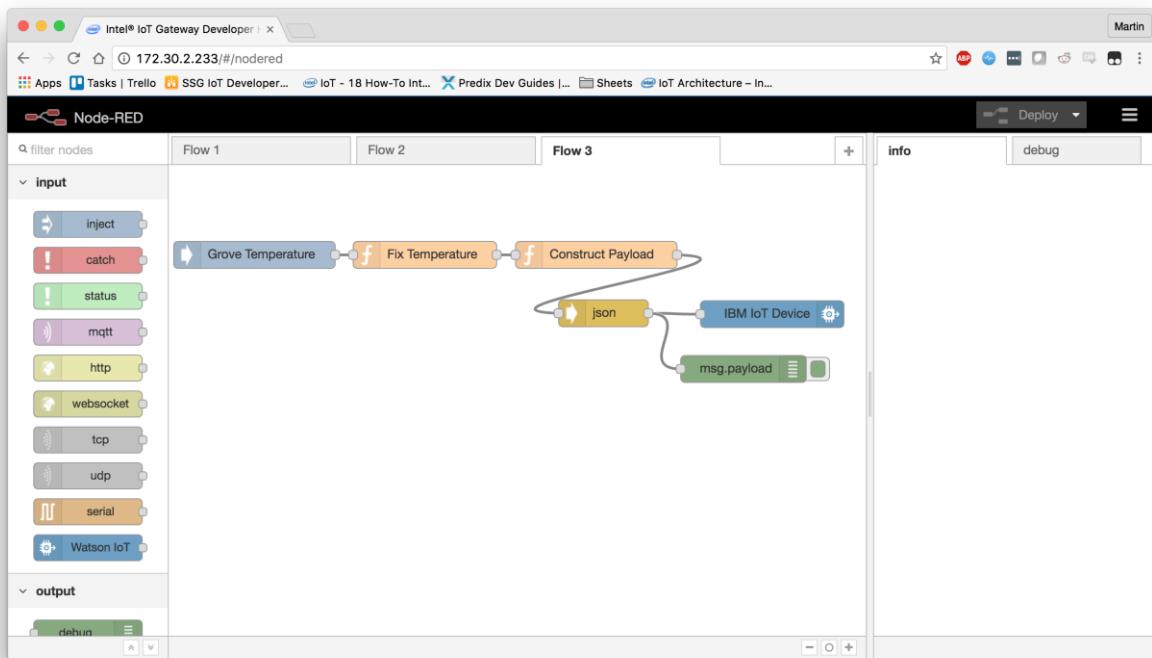
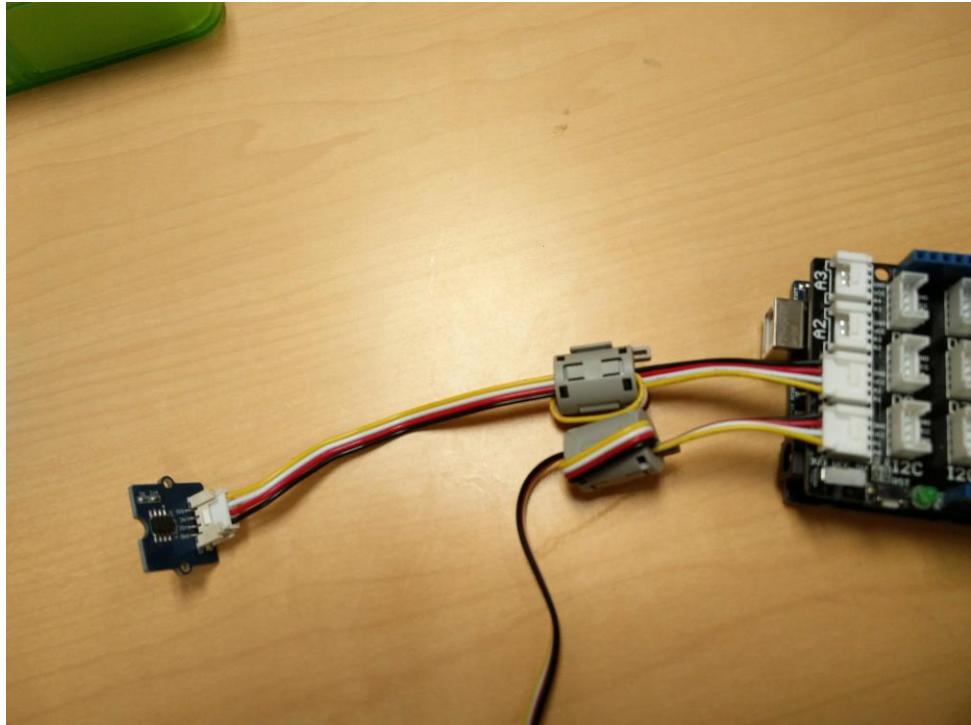
A function node can be used to add missing properties if they are outside the payload property.

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)

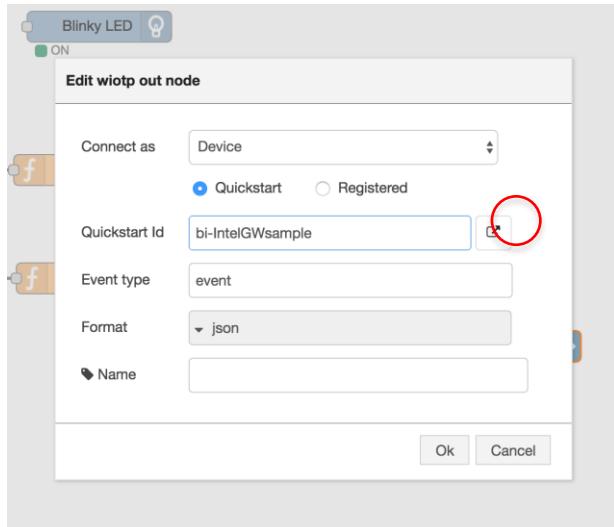
In the “Construct Payload” function add:

```
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:

The screenshot shows a web form titled "IBM Bluemix Promocodes". It has three input fields: the first contains "KrakowWorkshop", the second contains "binnes@uk.ibm.com", and the third also contains "binnes@uk.ibm.com". Below the fields is a reCAPTCHA section with a checked checkbox "I'm not a robot" and a "reCAPTCHA" logo. A note at the bottom states: "Note: By entering your information, you agree that it will be used for the purpose of distributing a Bluemix promotional code to you by email." A green "Submit" button is at the bottom right.

- 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.

The screenshot shows the IBM Cloud dashboard. At the top, there's a navigation bar with "Docs", "IBM Bluemix Apps" (selected), "Catalog", "Support", and "Account". The main area is titled "Apps" and shows a message: "You don't have any apps yet. Get started with one of the options that follow, or go to the catalog to create an app." A "Create App" button is below this. There are four cards: "Create a Cloud Foundry app", "Order a monthly Bare Metal Server", "Take advantage of IoT", and "Build a mobile app in a click". At the bottom, there are two more cards: "API Connect" and "VMware optimized for the cloud".

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.

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a sidebar with categories like All Categories, Infrastructure, Apps (Boilerplates, Cloud Foundry Apps, Containers, Mobile), Services (Data & Analytics, Watson, Internet of Things, APIs, Network, Storage, Security, DevOps, Application Services, Integrate), and a search bar. The main area displays a grid of application starters:

- ASP.NET Core Cloudant Starter**: Use the Cloudant NoSQL DB Service in an ASP.NET Core application.
- Internet of Things Platform Starter**: Get started with IBM Watson IoT platform using the Node-RED Node.js sample application. With the Watson IoT Platform dashboard.
- IoT for Electronics Starter**: IoT for Electronics is an integrated end-to-end solution (made of multiple services and apps) that enables...
- Java Cloudant Web Starter**: Use the Cloudant NoSQL DB service with the 'Liberty' for Java™ runtime.
- Java Workload Scheduler Web Starter**: This application demonstrates how to use the Workload Scheduler service, with the 'Liberty' for...
- LoopBack Starter**: This is a sample StrongLoop LoopBack Node.js application, powered by the open source LoopBack...
- MobileFirst Services Starter**: Start building your next mobile app with mobile services on Bluemix.
- Node.js Cloudant DB Web Starter**: Use the Cloudant NoSQL DB service with the 'SDK for Node.js™' runtime.
- Personality Insights Java Web Starter**: A simple Java app that uses the Personality Insights service to analyze text to derive personality traits.
- Node-RED Starter**: This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.
- Node.js Cloudant DB Web Starter**: Use the Cloudant NoSQL DB service with the 'SDK for Node.js™' runtime.
- StrongLoop Arc**: This application is the StrongLoop Arc graphical UI, which includes tools for building, profiling and...
- Mendix Rapid Apps**: Model driven rapid app platform that allows users to build, integrate and deploy web and mobile...
- Python Flask**: A simple Python Flask application that will get you up and running quickly.
- Ruby Sinatra**: Develop a Ruby web application using the Sinatra framework.
- Vaadin Rich Web Starter**: This application demonstrates how to use the Vaadin UI Framework to build rich HTML5 applications.

- 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.

The screenshot shows the 'Create a Cloud Foundry App' dialog box. It has fields for App name (myapp), Host name (myapp), and Domain (eu-gb.mybluemix.net). Under 'Selected Plan:', it shows the SDK for Node.js™ (Default) and Cloudant NoSQL DB (Lite). Below this, it lists the Internet of Things Platform (Lite). At the bottom, it says 'Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.' and includes links for 'Need Help? Contact Bluemix Sales' and 'Estimate Monthly Cost Cost Calculator'. A 'Create' button is at the bottom right.

- 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.

The screenshot shows the IBM Bluemix Cloud Foundry Apps dashboard. The application 'bi-myapp' is listed with a status of 'Your app is running'. The 'Logs' tab is selected. The log output shows the following:

```

Logs
All Errors Log type: all App instances: all Advanced view
Creating container
Successfully destroyed container
Successfully created container
Starting health monitoring of container
Starting app with 'node -max-old-space-size=384 node_modules/node-red/red.js --settings ./bluemix-settings.js -v'
>> Adding default flow
>> No default credentials found
18 Jan 11:43:17 - [info]
=====
Welcome to Node-RED
18 Jan 11:43:17 - [info] Node-RED version: v0.16.1
18 Jan 11:43:17 - [info] Node.js version: v4.6.2
18 Jan 11:43:17 - [info] Linux 4.4.0-45-generic x64 LE
18 Jan 11:43:17 - [info] Loading palette nodes
18 Jan 11:43:19 - [warn] [bmhdts] Deprecated call to RED.runtime.nodes.registerType - node-set name must be provided as first argument
18 Jan 11:43:19 - [warn] [bmhdts] Deprecated call to RED.runtime.nodes.registerType - node-set name must be provided as first argument
18 Jan 11:43:19 - [warn] [bmrpush] Deprecated call to RED.runtime.nodes.registerType - node-set name must be provided as first argument
18 Jan 11:43:21 - [info] Settings file: /home/vcap/app/bluemix-settings.js
18 Jan 11:43:21 - [info] Server now running at http://127.0.0.1:8080/red/
18 Jan 11:43:21 - [info] Starting flows
18 Jan 11:43:21 - [warn] [bmiot] IoT App In| Device Id is not set for Quickstart flow
18 Jan 11:43:21 - [info] Started flows
Container became healthy

```

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:

Welcome to your Internet of Things Platform (IoTP) boilerplate application on IBM Bluemix

This sample application uses Node RED to help demonstrate the wonderful things you can do with your IoTP service. We know you're eager to check it out, but first there is something important to do:

- Secure your Node-RED editor

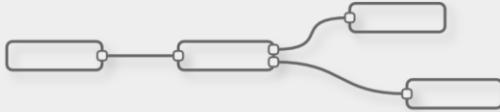
Previous Next

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.

Node-RED in Bluemix

A visual tool for wiring the Internet of Things

IBM Internet of Things Foundation



Node-RED provides a browser-based editor that makes it easy to wire together flows that can be deployed to the runtime in a single click.

The version running here has been customized for the IBM Internet of Things Foundation.

We strongly suggest you secure your Node-RED flow editor with a username and password, as otherwise anyone who can guess the URL of this application will be able to launch the flow editor and access your IoT device data.

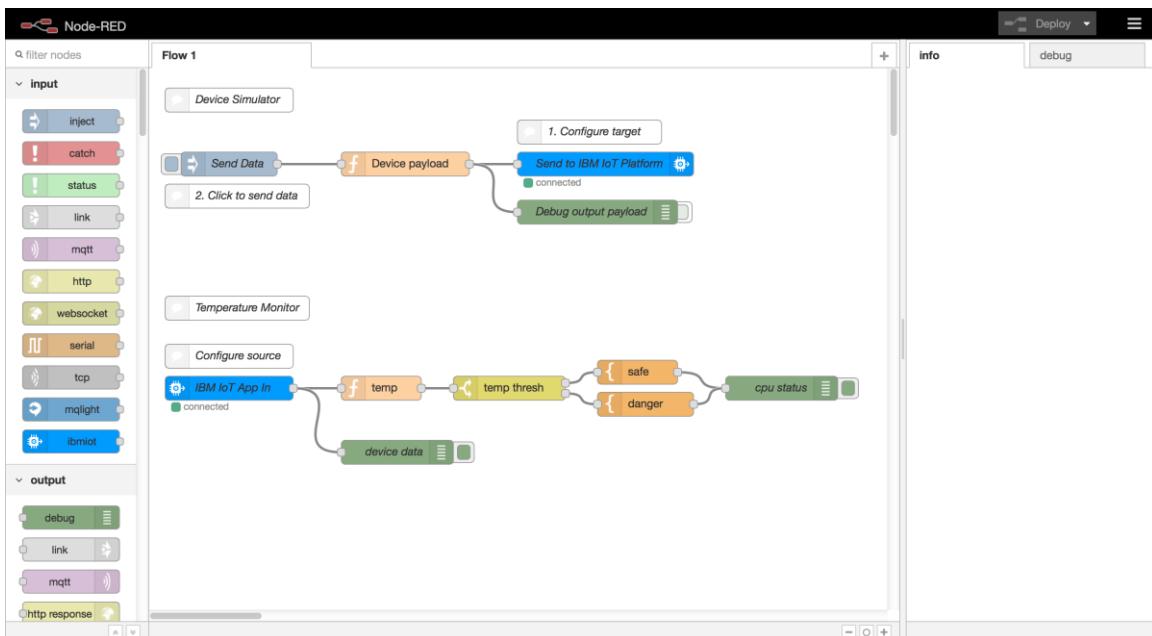
[Go to your Node-RED flow editor](#)

[Learn how to password-protect your instance](#)

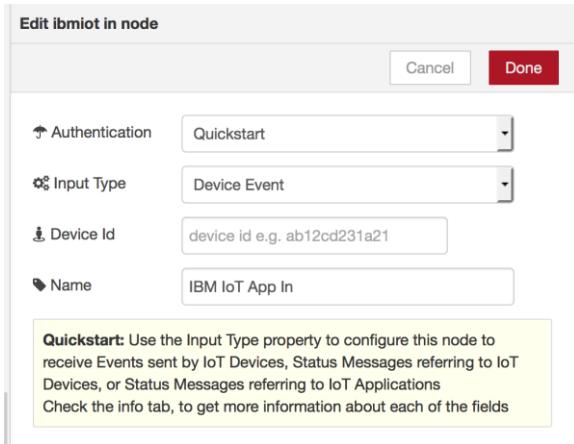
[Learn how to customise Node-RED](#)

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.

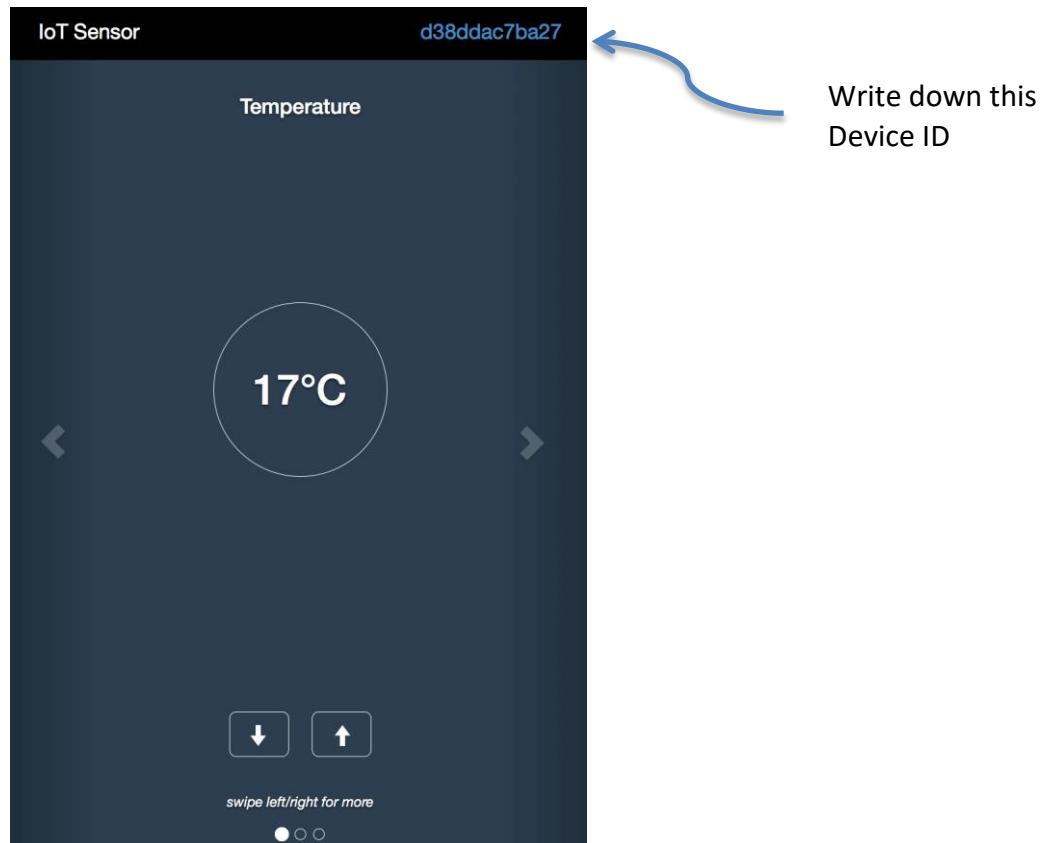
Let's begin by configuring the Watson IoT App In node. Double click on the Watson IoT App In node.



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.



- 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.



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

Edit ibmiot in node

Cancel	Done
Authentication	Quickstart
Input Type	Device Event
Device Id	d38ddac7ba27
Name	IBM IoT App In

Quickstart: Use the Input Type property to configure this node to receive Events sent by IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
 Check the info tab, to get more information about each of the fields

11. Click on the Deploy 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.

info debug

all flows current flow

```

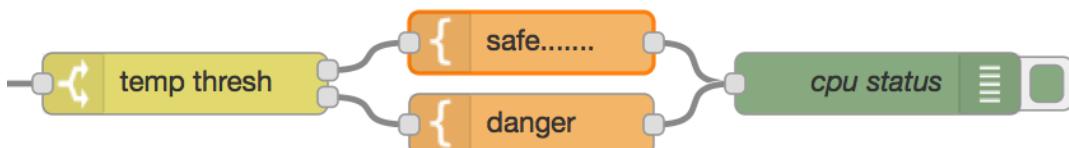
iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/iotsensor/fmt/json : msg : Object
{
  "topic": "iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/iotsensor/fmt/json",
  "payload": {
    "d": {
      "name": "d38ddac7ba27",
      "temp": 17,
      "humidity": 79,
      "objectTemp": 23
    }
  },
  "deviceId": "d38ddac7ba27",
  "deviceType": "iotqs-sensor",
  "eventType": "iotsensor",
  "format": "json",
  "_msgid": "b8912f10.476ed"
}

8/22/2016, 8:54:57 PM device data

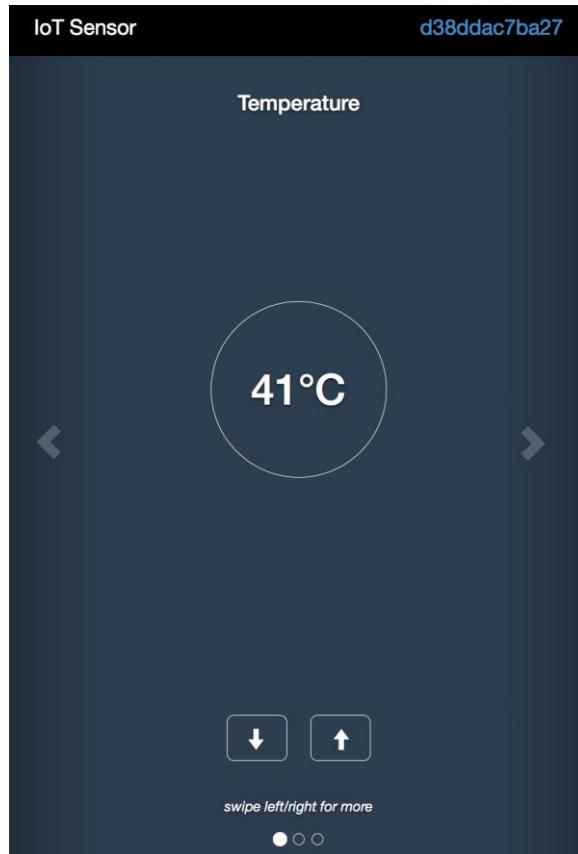
iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/iotsensor/fmt/json : msg : Object
{
  "topic": "iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/iotsensor/fmt/json",
  "payload": {
    "d": {
      "name": "d38ddac7ba27",
      "temp": 17,
      "humidity": 79,
      "objectTemp": 23
    }
  },
  "deviceId": "d38ddac7ba27",
  "deviceType": "iotqs-sensor",
  "eventType": "iotsensor",
  "format": "json",
  "_msgid": "380e7513.c7f18a"
}

```

13. The yellow node is called a switch 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.

A screenshot of the "debug" tab in a log viewer. The tab has tabs for "info" and "debug", with "debug" selected. There are buttons for "all flows", "current flow", and a trash bin icon. The log entries are:

```
8/22/2016, 9:02:49 PM device data
iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/ioticsensor/fmt/json : msg : Object
{
  "topic": "iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/ioticsensor/fmt/json",
  "payload": {
    "d": {
      "name": "d38ddac7ba27",
      "temp": 41,
      "humidity": 79,
      "objectTemp": 23
    }
  },
  "deviceId": "d38ddac7ba27",
  "deviceType": "iotqs-sensor",
  "eventType": "ioticsensor",
  "format": "json",
  "_msgid": "1de99894.e21667"
}
8/22/2016, 9:02:51 PM cpu status
msg.payload : string [25]
Temperature (41) critical
```

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:

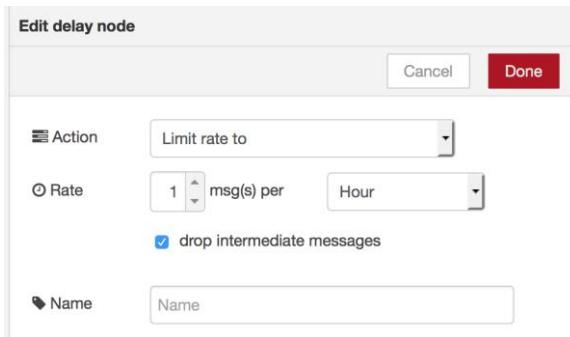
```
18/01/2017, 13:56:06 node: c0c482df.3f3b8
iot-2/type/iotqs-sensor/id/9ed93b03b216/evt/otsensor/fmt/json : msg : Object
▶ { topic: "iot-2/type/iotqs-sensor/id/9ed...", payload: object,
deviceId: "9ed93b03b216", deviceType: "iotqs-sensor", eventType:
"otsensor" ... }

18/01/2017, 13:56:08 node: c0c482df.3f3b8
iot-2/type/iotqs-sensor/id/9ed93b03b216/evt/otsensor/fmt/json : msg : Object
▼ object
  topic: "iot-2/type/iotqs-
  sensor/id/9ed93b03b216/evt/otsensor/fmt/json"
▼ payload: object
  ▼ d: object
    name: "9ed93b03b216"
    temp: 16
    humidity: 79
    objectTemp: 25
  deviceId: "9ed93b03b216"
  deviceType: "iotqs-sensor"
  eventType: "otsensor"
  format: "json"
  _msgid: "6bbf2dd4.9440d4"
```

Connect to Twitter and Tweet High Temperature

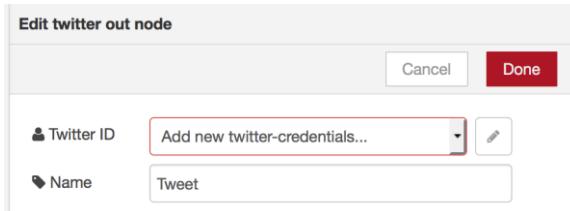
In this section, we will connect a Twitter account and use the Twitter account to tweet when the temperature from the temperature sensor is “dangerous”. This section is optional and may be skipped.

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.



Authorize Node RED to use your account?

Authorize app **Cancel**

This application will be able to:

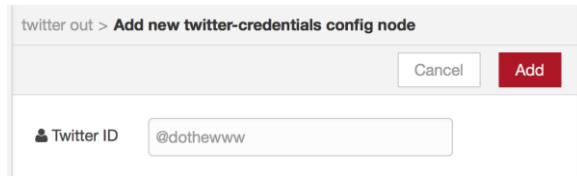
- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.

Will not be able to:

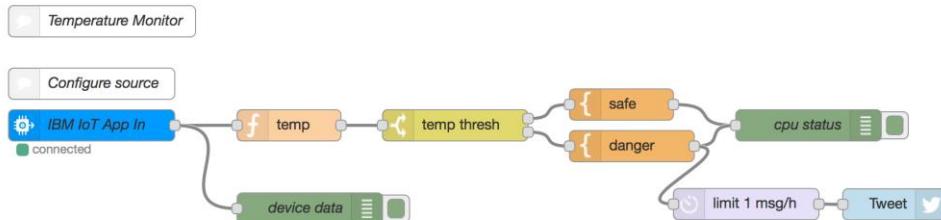
- See your Twitter password.

Node RED
nodered.org
Node-RED Twitter node

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 **Deploy** 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

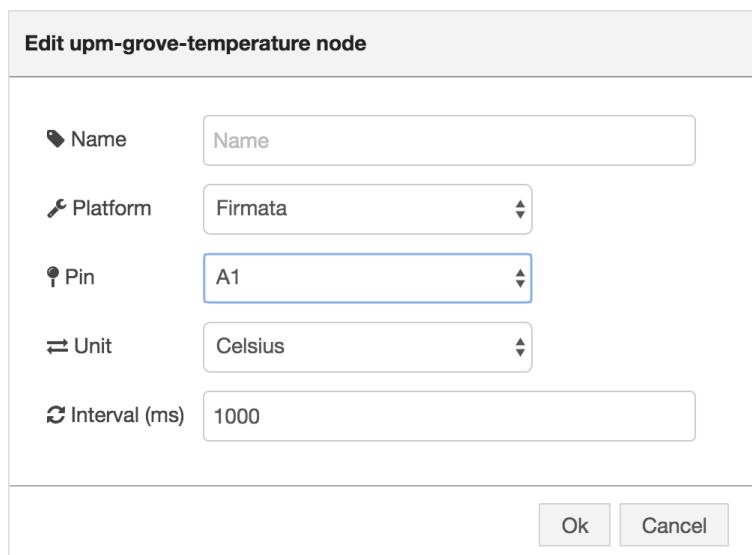
In the first section, we used a simulated device to get started with the Internet of Things Platform. When you have simulated what your device will do, then you can invest in making the hardware and hopefully reduce the cost incurred along the way. The Intel NUC board has made prototyping IoT devices 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 deleted 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.

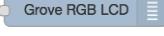
3. Add a  function node as shown below:



4. Add a  function node as shown below:



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

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

Edit Grove RGB LCD node

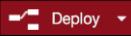
Cancel Done

Name	Name
Platform	Firmata
R	0
G	0
B	255
Cursor Row	0
Cursor Column	0

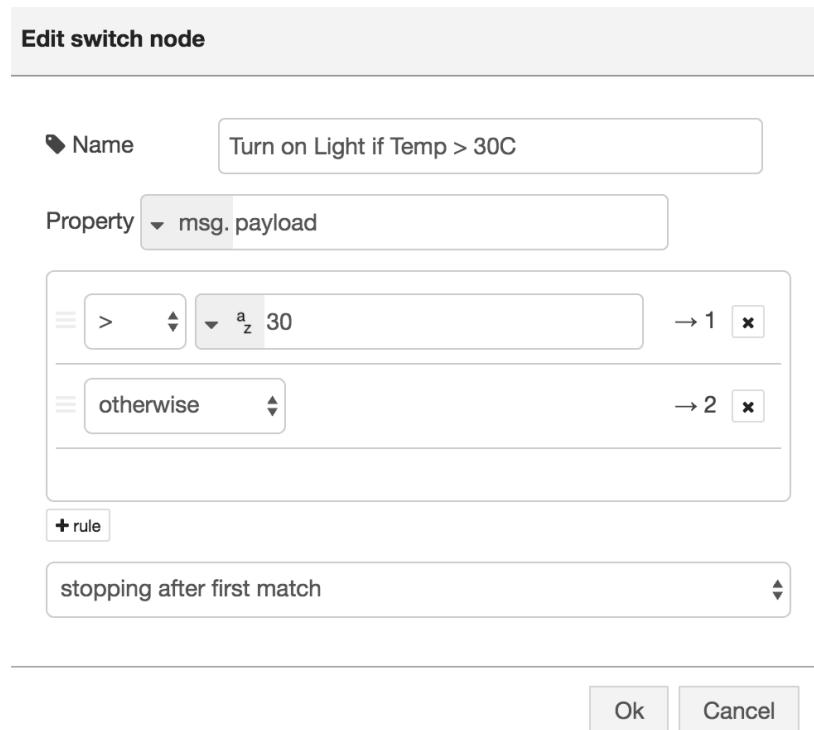
This will display the temperature value on the LCD connected to the I2C port.

6. Connect the nodes together as shown below.



7. Click on the  Deploy 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.

Edit function node

Name: Turn LED on

Function:

```
1 msg.payload=1;
2 return msg;
```

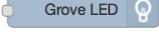
Edit function node

Name: Turn LED off

Function:

```
1 msg.payload=0;
2 return msg;
```

Done Done

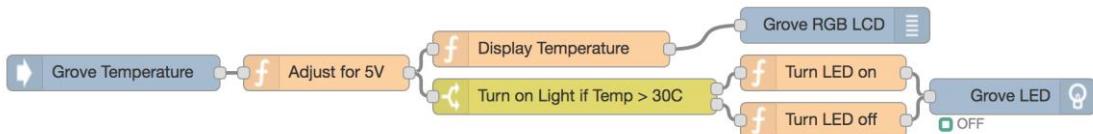
10. Add a  Grove LED node as shown below. This will control the LED connected to port D4.

Edit upm-grove-led node

Name	Name
Platform	Firmata
Pin	D4
Mode	Output

Ok Cancel

11. Connect the nodes together as shown below



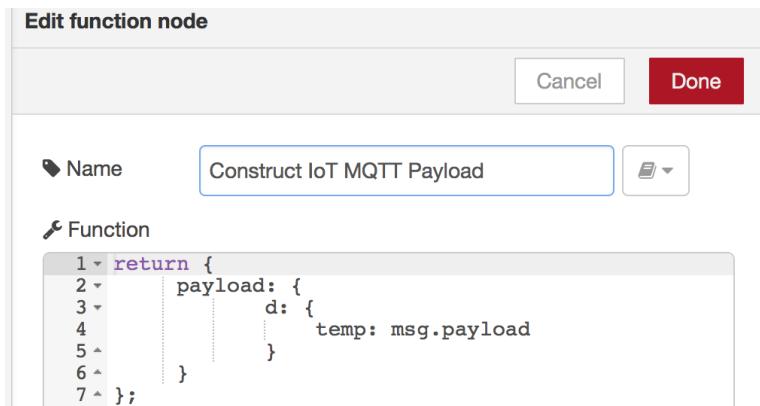
12. Connect a LED light to the D4 port on the Grove base shield connected to the Intel Edison.

13. Click on the  Deploy 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.**

Edit Watson IoT node

Cancel Done

Connect as Device Quickstart Registered

Quickstart Id d38ddac7ba27

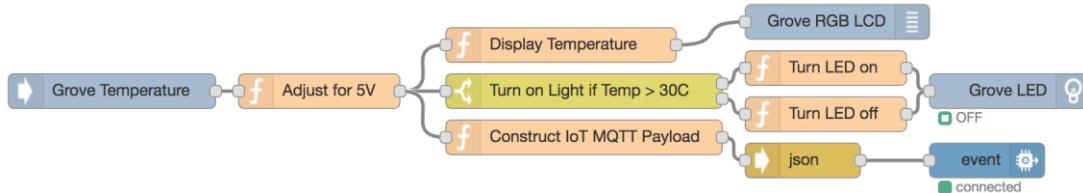
Event type event

Format json

Name Watson IoT Quickstart

Reuse Device ID
from simulated IoT
sensor

5. Connect the nodes together as shown below.



6. Click on Deploy 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 used 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.

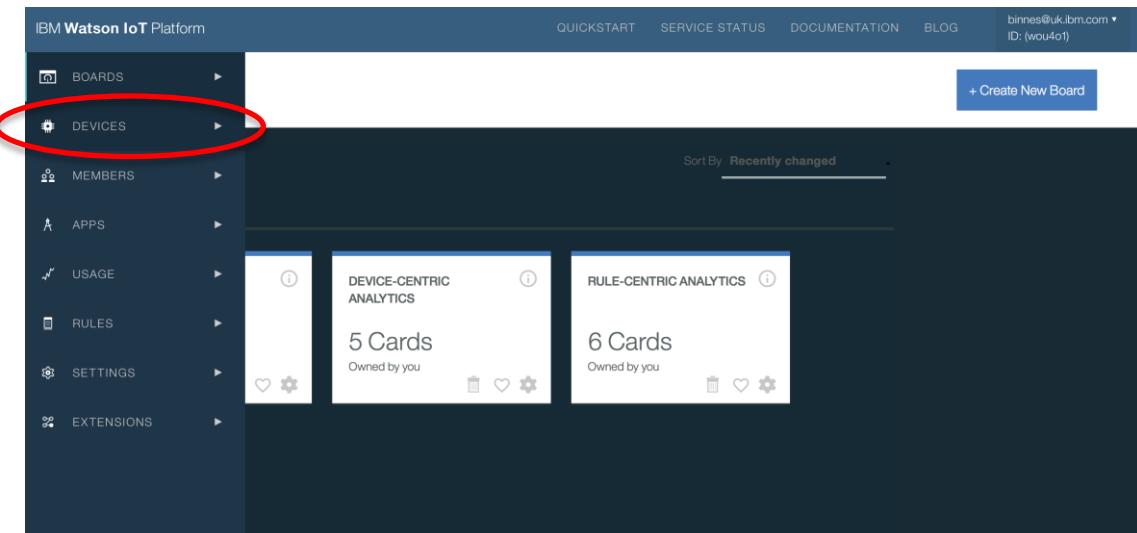
1. In IBM Cloud go to the Overview page of your boilerplate application and double click the iotf service in the Connections panel:

The screenshot shows the IBM Bluemix Cloud Foundry Apps Overview page for an application named 'bi-myapp'. The page displays runtime statistics: Buildpack (Node.js), Instances (1), MB Memory per Instance (512), and Total MB Allocation (512). It also shows two connections: 'bi-myapp-cloudantNoSQLDB' and 'bi-myapp-iotf-service'. A 'Runtime cost' section indicates charges of £0.54 for the current period and an estimated total of £0.54 for the billing period (01/01-01/31). A 'View full usage details' button is present. The sidebar on the left includes links for Dashboard, Getting started, Overview (selected), Runtime, Connections, Logs, and Monitoring.

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

The screenshot shows the Watson IoT Platform service landing page for 'bi-myapp-iotf-service'. The page features a central graphic of a lock and a key, symbolizing security. Below it, a call-to-action button labeled 'Launch' is circled in red. The page also includes sections for 'Learn about Watson IoT Platform' and 'Expand using step-by-step recipes', both with expandable arrows. The left sidebar has a 'Manage' section with 'Plan' and 'Connections' options.

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

Add Device

Choose Device Type



Choose Device Type

Or

Create device type

- c. Select Create gateway type

Create Device Type

Create Type



Create device type

Create gateway type

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

Create Gateway Type

General Information



Name

NUCGateway

The device type name is used to identify the device type uniquely, using a restricted set of characters to make it suitable for API use.

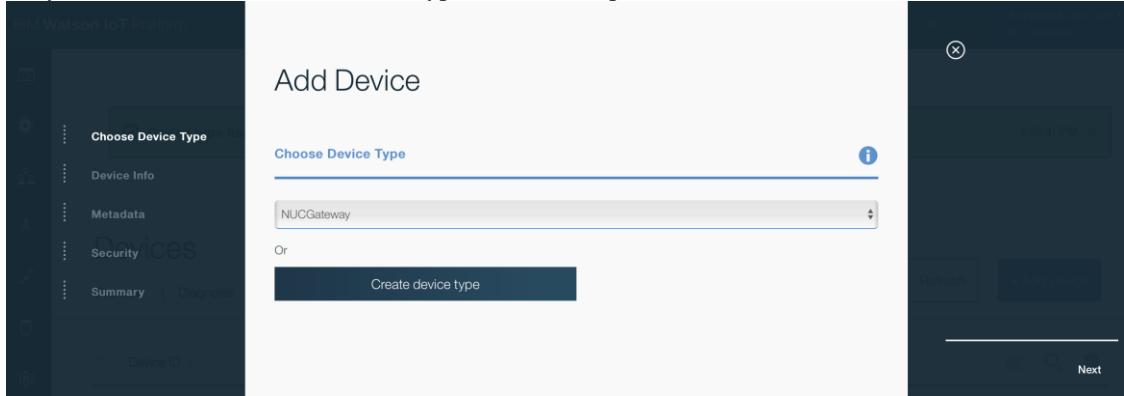
Description

Intel NUC Gateway|

The device type description can be used for a more descriptive way of identifying the device type.

- 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 blank, 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

Add Device

Device Info

Device ID is the only required information, however other fields are populated according to the attributes set in the selected device type. These values can be overridden, and attributes not set in the device type can be added.

Device ID	c03fd56c08f9
-----------	--------------

+ Additional fields

- 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.

- 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.

Gateway c03fd56c08f9

Your Device Credentials

You have registered your device to the organization. To get it connected, you need to add these credentials to your device. Once you've added these, you should see the messages sent from your device in the 'Sensor Information' section on this page.

Organization ID	wou4o1
Device Type	NUCGateway
Device ID	c03fd56c08f9
Authentication Method	token
Authentication Token	passw0rd

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

[Find out how to add these credentials to your device](#)

Connection Information

Device ID: c03fd56c08f9
Device Type: NUCGateway

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

Devices

Devices

Browse | Diagnose | Action | Device Types | Manage Schemas Refresh + Add Device

	Device ID	Device Type	Class ID	Date Added	Location	Actions
Results 1-1 of 1	c03fd56c08f9	NUCGateway	Gateway	Jan 19, 2017 1:26:39 PM		

- Repeat the process for the 101

- Create a device type rather than a gateway type
- Device Type : 101
- Device ID 101_ prepended to Gateway Device ID, e.g. 101_c03fd56c08f9

Device 101_c03fd56c08f9

Your Device Credentials

You have registered your device to the organization. To get it connected, you need to add these credentials to your device. Once you've added these, you should see the messages sent from your device in the 'Sensor Information' section on this page.

Organization ID	wou4o1
Device Type	101
Device ID	101_c03fd56c08f9
Authentication Method	token
Authentication Token	passw0rd

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

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

Devices

Browse | Diagnose | Action | Device Types | Manage Schemas Refresh + Add Device

Device ID	Device Type	Class ID	Date Added	Location
101_c03fd56c08f9	101	Device	Jan 19, 2017 1:32:34 PM	
c03fd56c08f9	NUCGateway	Gateway	Jan 19, 2017 1:26:39 PM	

Results 1-2 of 2 Show/hide columns

- 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 Device Types and Device Ids:

Edit ibmiot in node

Delete Cancel Done

Authentication	Bluemix Service
Input Type	Device Event
Device Type	All or +
Device Id	All or device id e.g. ab12cd231a21
Event	All or +
Format	All or json
QoS	
Name	IBM IoT App In

- 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:

IBM Watson IoT Platform binnes@uk.ibm.com ▾ ID: (wou4o1)

QUICKSTART SERVICE STATUS DOCUMENTATION BLOG

BOARDS DEVICES MEMBERS APPS USAGE RULES SETTINGS EXTENSIONS

Action | Device Types | Manage Schemas Refresh + Add Device

Device ID	Device Type	Class ID	Date Added	Location
101_c03fd56c08f9	101	Device	Jan 19, 2017 1:32:34 PM	
c03fd56c08f9	NUCGateway	Gateway	Jan 19, 2017 1:26:39 PM	

- 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:

Edit wiotp out node

Connect as	Gateway
	<input type="radio"/> Quickstart <input checked="" type="radio"/> Registered
Credentials	Add new wiotp-credentials... 
Device Type	e.g. sensor
Device Id	e.g. ab12cd231a21
Event type	event

- 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.

Add new wiotp-credentials config node Flow 2

Organization	wou4o1
Device Type	NUCGateway
Device ID	c03fd56c08f9
Auth Token
Name	<input type="text" value="c03fd56c08f9"/>

Add **Cancel**

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

Edit wiotp out node

Connect as	Gateway
	<input type="radio"/> Quickstart <input checked="" type="radio"/> Registered
Credentials	c03fd56c08f9
Device Type	e.g. sensor
Device Id	e.g. ab12cd231a21
Event type	event
Format	json
Name	

Ok **Cancel**

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.

Devices

	Device ID	Device Type	Class ID	Date Added	Location	Refresh	+ Add Device
Results 1-2 of 2							
<input type="checkbox"/>	101_c03fd56c08f9	101	Device	Jan 19, 2017 1:32:34 PM			
<input type="checkbox"/>	c03fd56c08f9	NUCGateway	Gateway	Jan 19, 2017 1:26:39 PM			

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

Connection Information

Device ID	c03fd56c08f9
Device Type	NUCGateway
Date Added	Thursday, January 19, 2017
Added By	binnes@uk.ibm.com
Connection State	Connected on Thursday, January 19, 2017 at 2:24:34 PM from 82.15.225.235 with a secure connection Refresh

Recent Events

Event	Format	Time Received
event	json	Jan 19, 2017 2:29:12 PM
event	json	Jan 19, 2017 2:29:13 PM
event	json	Jan 19, 2017 2:29:14 PM
event	json	Jan 19, 2017 2:29:15 PM
event	json	Jan 19, 2017 2:29:16 PM
event	json	Jan 19, 2017 2:29:17 PM
event	json	Jan 19, 2017 2:29:18 PM

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.

The screenshot shows a Node-RED flow with two nodes: a Grove LED node and an event node. The event node has a status indicator showing it is connected. To the right of the flow is a properties panel for a 'wiotp out' node. The properties panel includes fields for authType (g), qs (false), qsDeviceId (blank), deviceKey (f9997fd4.2c54f), deviceType (blank), deviceld (blank), event (event), and format (json). Below the properties panel is a block of text explaining the node's function: "Send device events to the IBM Watson Internet of Things Platform. The node can connect as either a Device or Gateway, in registered mode or using the Quickstart service. When connecting using the Quickstart service, the connection will use a device type of".

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:

The screenshot shows the 'Edit function node' dialog. The node is named 'set device'. The function code is defined as follows:

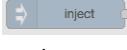
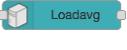
```
1 msg.deviceType = "101";
2 msg.deviceId = "101_c03fd56c08f9";
3 return msg;
```

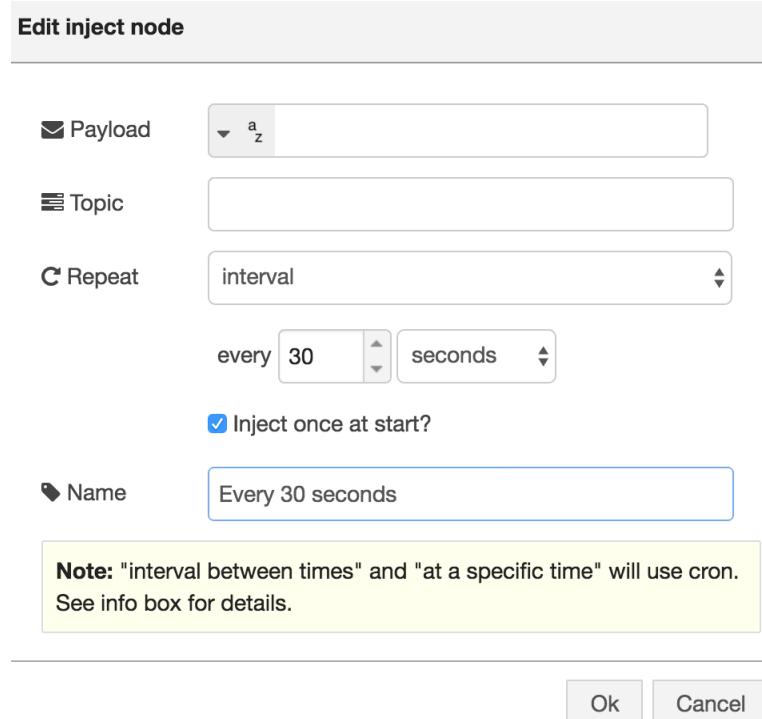
The 'Outputs' dropdown indicates there is 1 output. A note at the bottom says 'See the Info tab for help writing functions.' At the bottom right are 'Ok' and 'Cancel' buttons. A preview window at the bottom shows the generated JavaScript code:

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

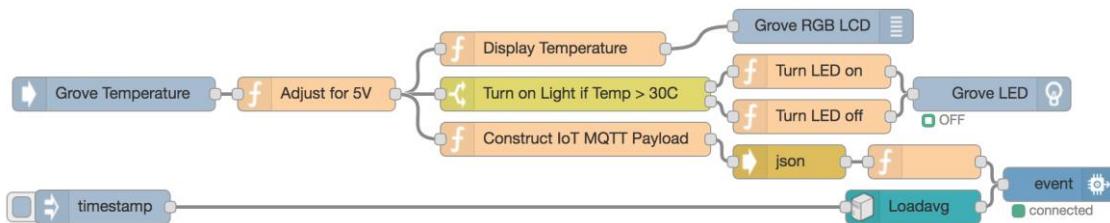
13. Hit deploy and then go look back in the Watson IoT Platform console. You should see that the Gateway is still connected and the device is not showing as connected, but clicking into the Gateway device you should not be seeing any

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.

Gateway c03fd56c08f9

Connection Information

Device ID	c03fd56c08f9
Device Type	NUCGateway
Date Added	Thursday, January 19, 2017
Added By	binnest@uk.ibm.com
Connection State	Connected on Thursday, January 19, 2017 at 2:45:36 PM from 82.15.225.235 with a secure connection Refresh

Recent Events

Event	Format	Time Received
event	json	Jan 19, 2017 2:46:06 PM

Sensor Information

Event	Datapoint	Value	Time Received
event	d.loadavg[0]	0.00732421875	Jan 19, 2017 2:46:06 PM

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.

Edit ibmiot in node

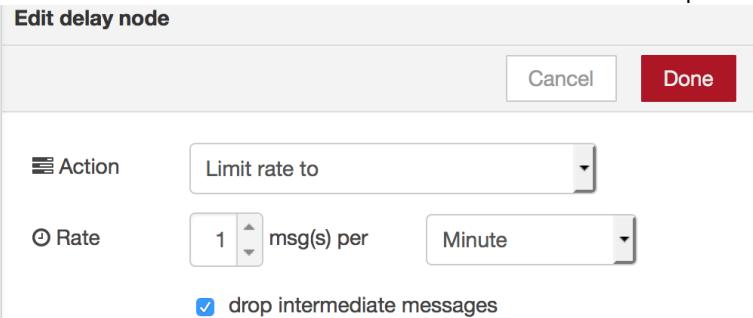
Delete	Cancel	Done
Authentication	Bluemix Service	
Input Type	Device Event	
Device Type	<input type="checkbox"/> All or <input checked="" type="checkbox"/> 101	
Device Id	<input checked="" type="checkbox"/> All or <input type="text" value="device id e.g. ab12cd231a21"/>	
Event	<input checked="" type="checkbox"/> All or <input type="text" value="+"/>	
Format	<input type="checkbox"/> All or <input type="text" value="json"/>	
QoS	0	
Name	IBM IoT App In	

Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields

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.

Edit cloudant out node

Service: bi-myapp-cloudantNoSQLDB

Database: temperature

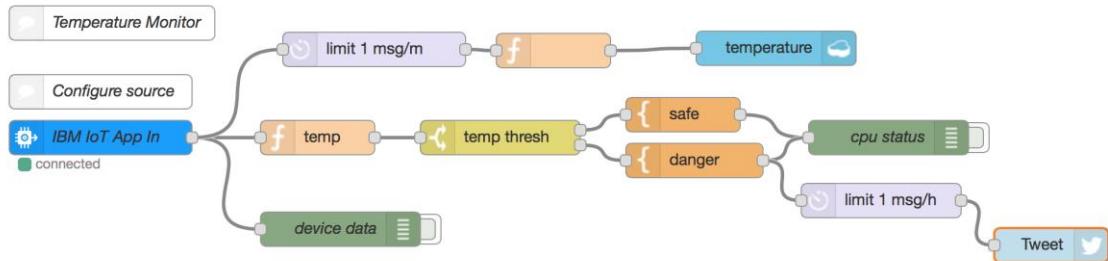
Operation: insert

Only store msg.payload object?

Name: Name

Cancel Done

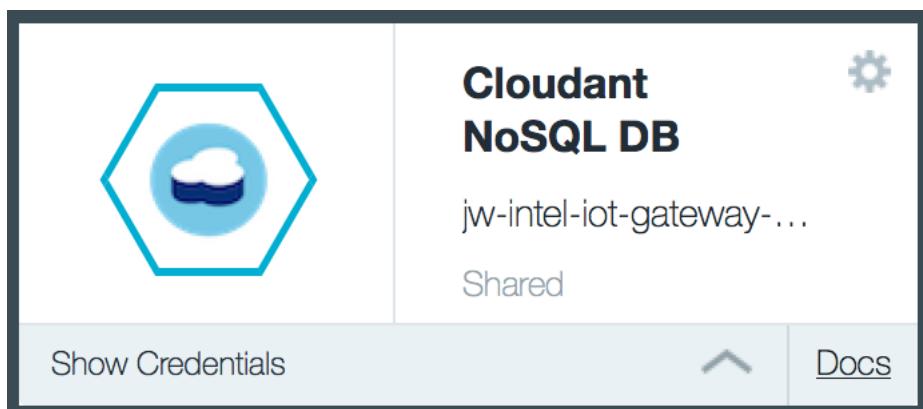
4. Connect the nodes together as shown below.



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.

The Cloudant NoSQL Database service adds JSON data to your Mobile and Web applications, accessible via easy-to-use RESTful HTTP/S APIs.

Ease of Use

Work with self-describing JSON documents through a RESTful API that makes every document in your Cloudant database accessible as JSON via a URL. Documents can be retrieved, stored, or deleted individually or in bulk and can also have files attached. IBM takes care of the provisioning, management, and scalability of the data store, freeing up your time to focus on your application.

Powerful search, sync and more

With extremely powerful indexing, real time MapReduce and Apache Lucene-based full-text search, Cloudant NoSQL DB makes it easy to add advanced data analytics and powerful data access. Data access can also extend to Cloudant Sync, enabling data access from mobile devices and client apps to run connected or off-line.

Get Started



Learn

View complete tutorials and demonstrations of Cloudant NoSQL DB



Discover

Check out our forums to see what other people are doing with Cloudant NoSQL DB



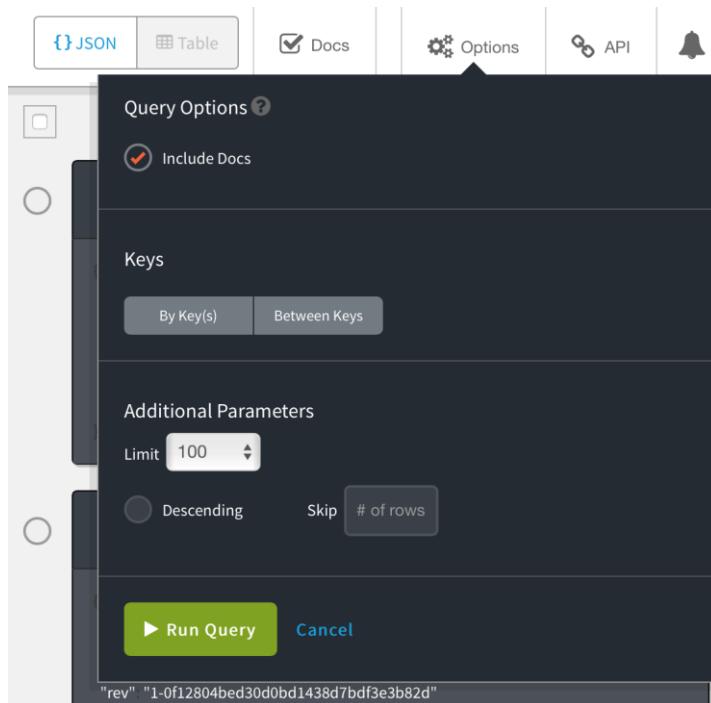
Launch

Launch the console to get started with Cloudant NoSQL DB today!

- 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**

Name	Size	# of Docs	Actions
_replicator	3.7 KB	1	
_users	66.6 KB	0	
nodered	44.9 KB	4	
temperature	6.3 KB	4	

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.

	Metadata	temp	time
<input type="checkbox"/>	0935f9888f9a... 1-ca2242480eac...	16	1484748522307
<input type="checkbox"/>	0935f9888f9a... 1-b3389e61d5fc...	16	1484748768300
<input type="checkbox"/>	3c97d6a180c... 1-0f12804bed30...	16	1484748336315
<input type="checkbox"/>	446be7192ba... 1-b320ddda81c...	16	1484748644300
<input type="checkbox"/>	772b4930f5d0... 1-0ac21c5def60...	16	1484748460309
<input type="checkbox"/>	80339ce96993... 1-9de7e2f27a44...	16	1484748398309
<input type="checkbox"/>	80339ce96993... 1-e9b99ba4db4...	16	1484748582310

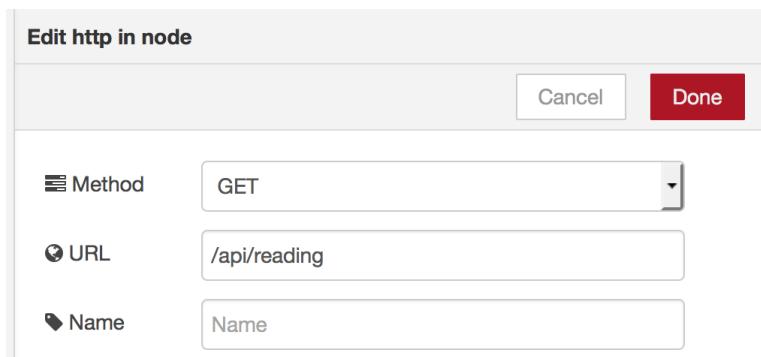
Showing 3 columns. Show all columns. < >

Retrieve Temperatures from Cloudant NoSQL DB

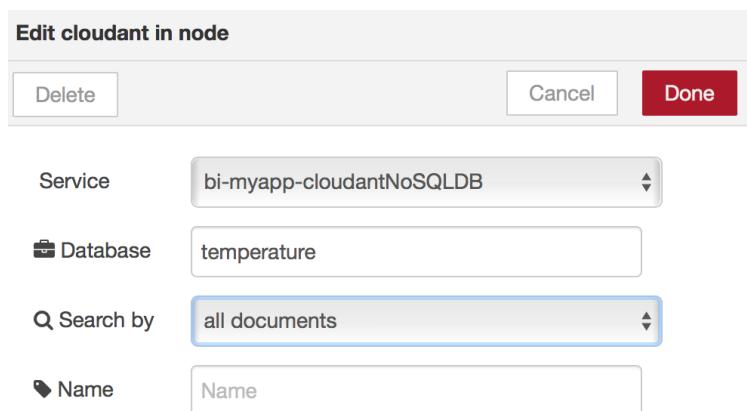
This section shows how to retrieve temperature data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time.

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

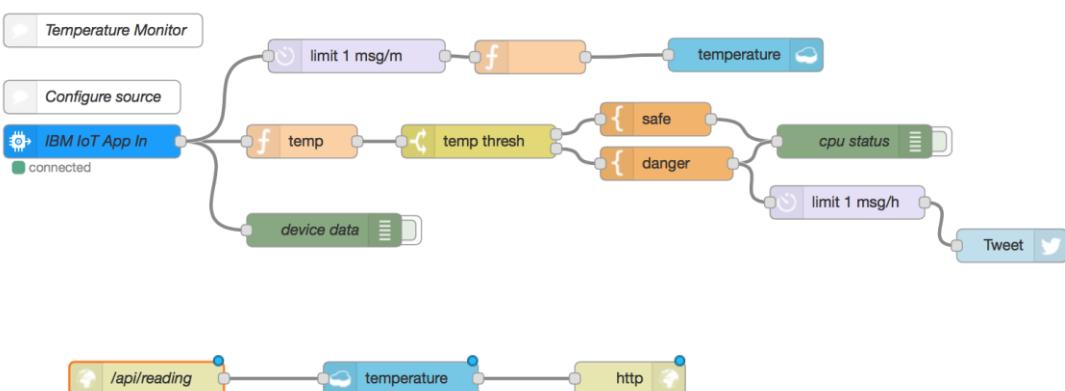
1. Add a  node as shown below.

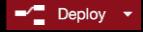


2. Add a  node as show below.



3. Finally, add a  node. Connect the nodes together as shown below.



- Click on  Deploy to save and deploy your changes.
- 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>

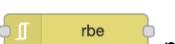
You should see data returned similar to the following:

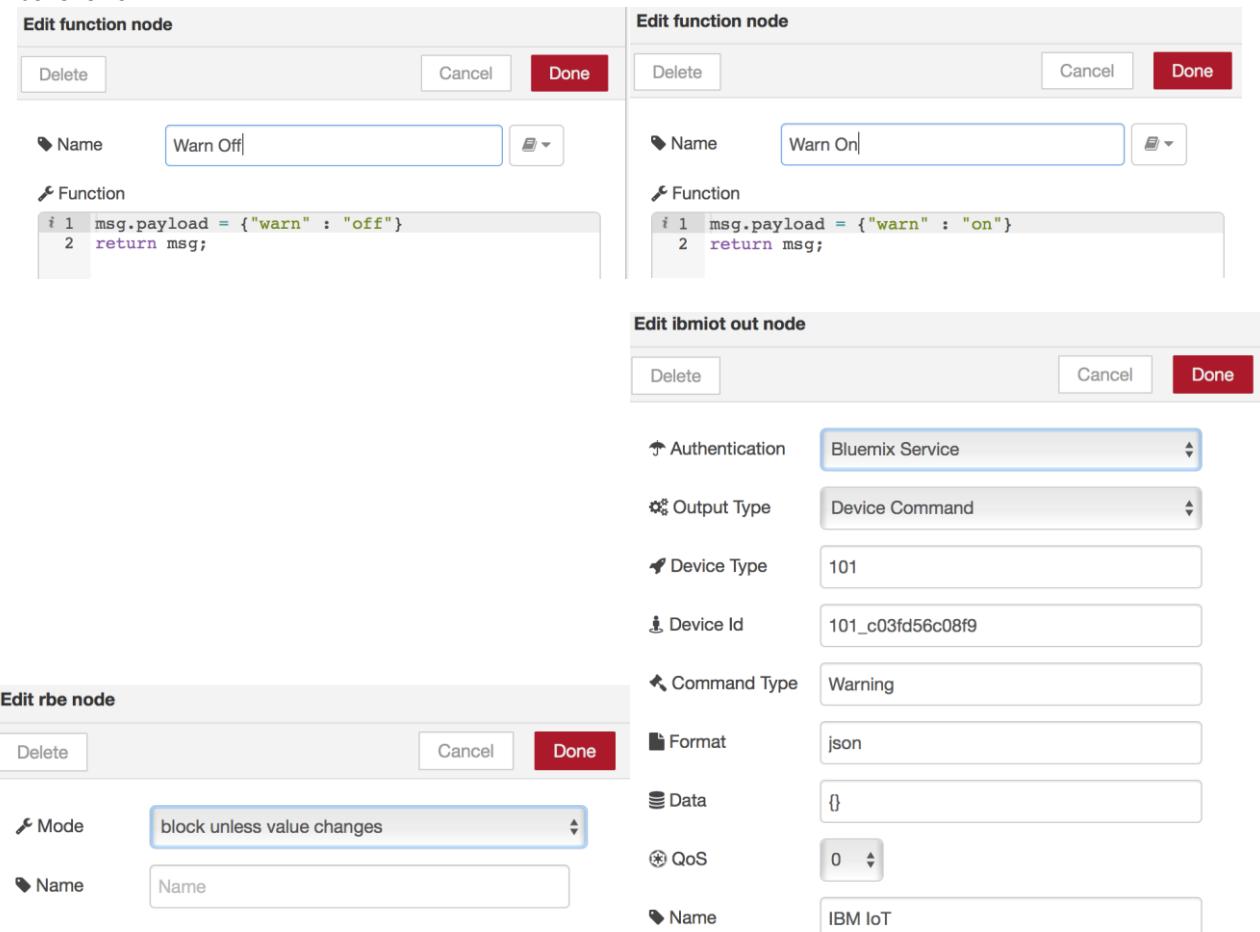
```
[{"_id": "0935f9888f9aa663f1b1cdd33debb8aa", "_rev": "1-ca2242480eac26944ce814fa07ec5915", "time": 1484748522307, "temp": 16}, {"_id": "0935f9888f9aa663f1b1cdd33df19e39", "_rev": "1-b3389e61d5fc606bb3131d61e2053169", "time": 1484748768300, "temp": 16}, {"_id": "0935f9888f9aa663f1b1cdd33df49669", "_rev": "1-4b54f5939673ada224df354d7ce2ceba", "time": 1484748950299, "temp": 16}, {"_id": "33f7647931b6975f75fd13df0514c8db", "_rev": "1-dda39d7b9c676db26d3683aa4460fefa", "time": 1484749074287, "temp": 16}, {"_id": "3c97d6a180c5bd3fd67b24720066c83", "_rev": "1-0f12804bed30d0bd1438d7bdf3e3b82d", "time": 1484748336315, "temp": 16}, {"_id": "446be7192ba3827cdd2354371bc8aec4", "_rev": "1-b320ddda81c5b33bbd1ae54a69dd0278", "time": 1484748644300, "temp": 16}, {"_id": "446be7192ba3827cdd2354371bdb5ea", "_rev": "1-23db478dea013ee871c42dc4fc47c09", "time": 1484749012294, "temp": 16}, {"_id": "772b4930f5d0d6e145e98e8701c3f339", "_rev": "1-0ac21c5def60e15bbe3bf61e9d67c114", "time": 1484748460309, "temp": 16}, {"_id": "80339ce96993b43c641236f8034e5b27", "_rev": "1-9de7e2f27a412576cb08d43170359a", "time": 1484748398309, "temp": 16}, {"_id": "80339ce96993b43c641236f80352a31c", "_rev": "1-e9b99ba4db4204dcbb5f985657298ef9", "time": 1484748522310, "temp": 16}, {"_id": "80339ce96993b43c641236f80355efac", "_rev": "1-9afble80f59078267cd843af6d1fe2e", "time": 1484748706349, "temp": 16}, {"_id": "80339ce96993b43c641236f80358c026", "_rev": "1-57c92ea9bcc04490f1705604b54644d", "time": 1484748830294, "temp": 16}, {"_id": "80339ce96993b43c641236f8035f91e3", "_rev": "1-7100b888059678287f24504ae0367c02", "time": 1484749136288, "temp": 16}, {"_id": "a9580c6666a1b6bla8f9c5a81b9281cd", "_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:



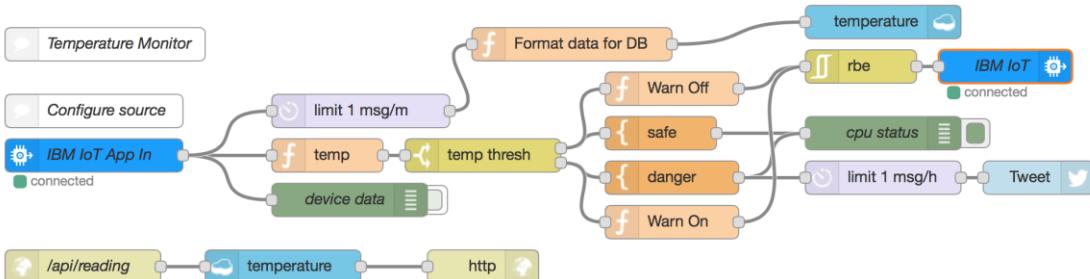
The image shows four configuration dialogs for Node-RED nodes:

- Edit function node (Warn Off):** Name: Warn Off. Function code:

```
i 1 msg.payload = {"warn" : "off"}  
2 return msg;
```
- Edit function node (Warn On):** Name: Warn On. Function code:

```
i 1 msg.payload = {"warn" : "on"}  
2 return msg;
```
- Edit rbe node:** Mode: block unless value changes. Name: Name.
- Edit ibmiot out node:** Authentication: Bluemix Service. Output Type: Device Command. Device Type: 101. Device Id: 101_c03fd56c08f9. Command Type: Warning. Format: json. Data: {}. QoS: 0. Name: IBM IoT.

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:

Edit wiotp in node

Connect as

Credentials

Subscribe to Gateway commands Device commands

Device Type

Device Id

Command

Name

Edit function node

Name Set background state in flow context

Function

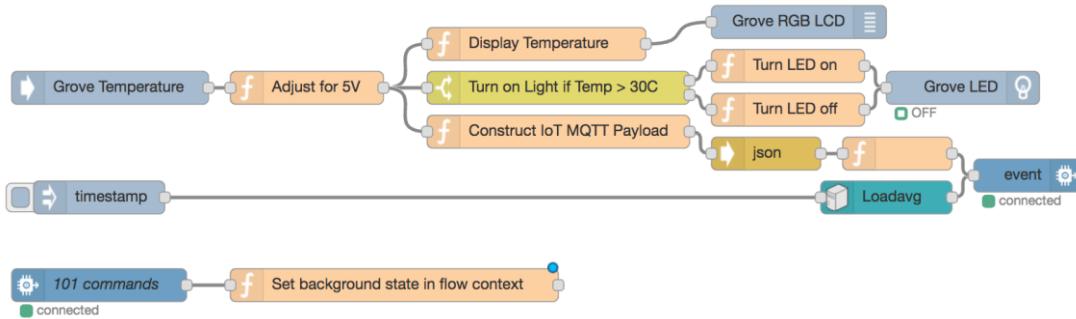
```
1 if (msg.payload.warn === "on") {
2   flow.set("background", 255);
3 } else {
4   flow.set("background", 64);
5 }
6 return msg;
```

Outputs 1

See the Info tab for help writing functions.

```
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/>