

Intel IoT Gateway / Arduino 101 with Watson IoT

Hands-On Workshop

Authors:

Martin Kronberg | martin.kronberg@intel.com | [@martinkronberg](https://twitter.com/martinkronberg)
Vaghesh Patel | vaghesh.patel@intel.com | [@vaghesh
JeanCarl Bisson | \[jbisson@us.ibm.com\]\(mailto:jbisson@us.ibm.com\) | \[@dothewww
John Walicki | \\[walicki@us.ibm.com\\]\\(mailto:walicki@us.ibm.com\\) | \\[@johnwalicki\\]\\(https://twitter.com/johnwalicki\\)\]\(https://twitter.com/dothewww\)](https://twitter.com/vaghesh)

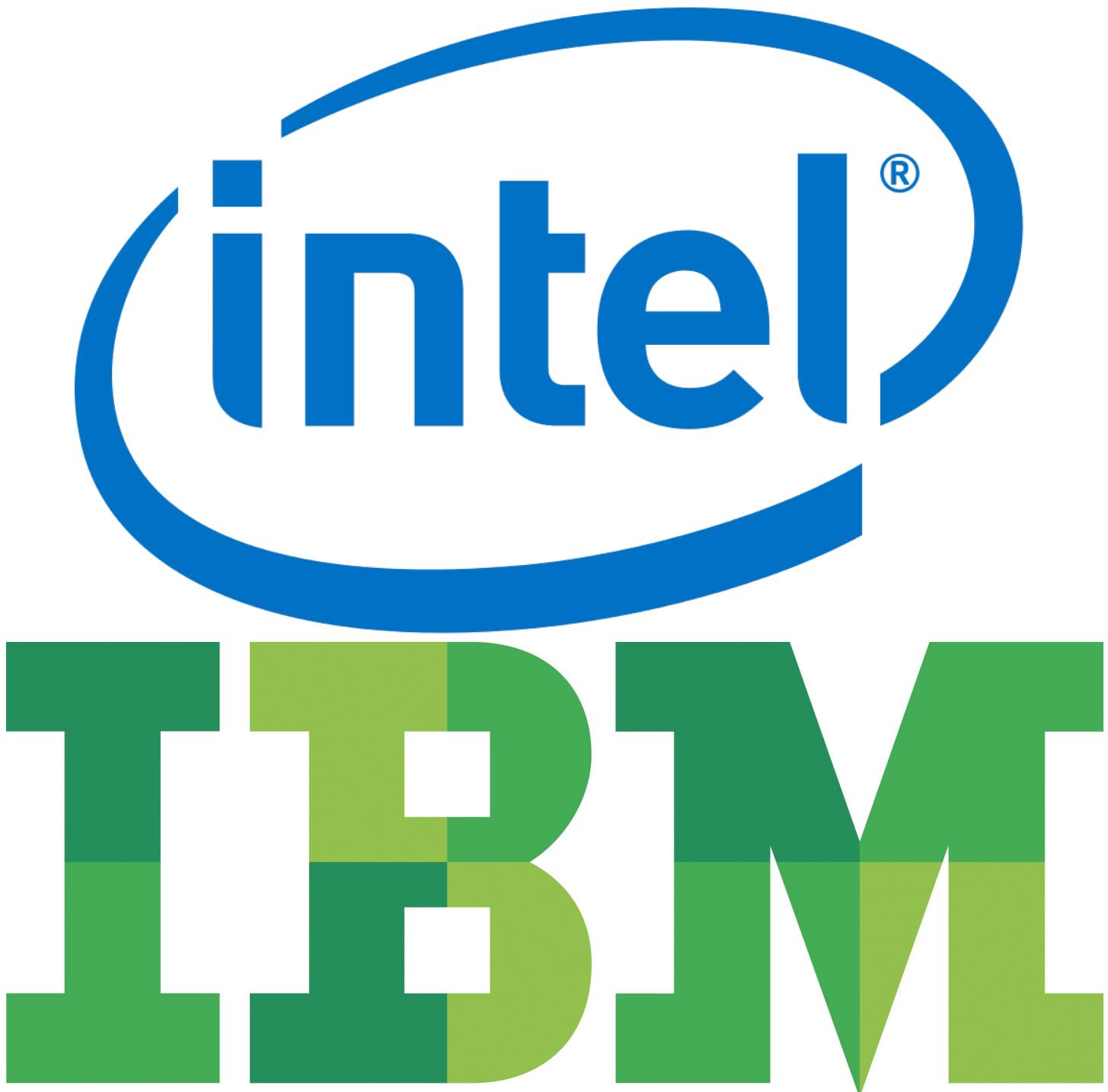
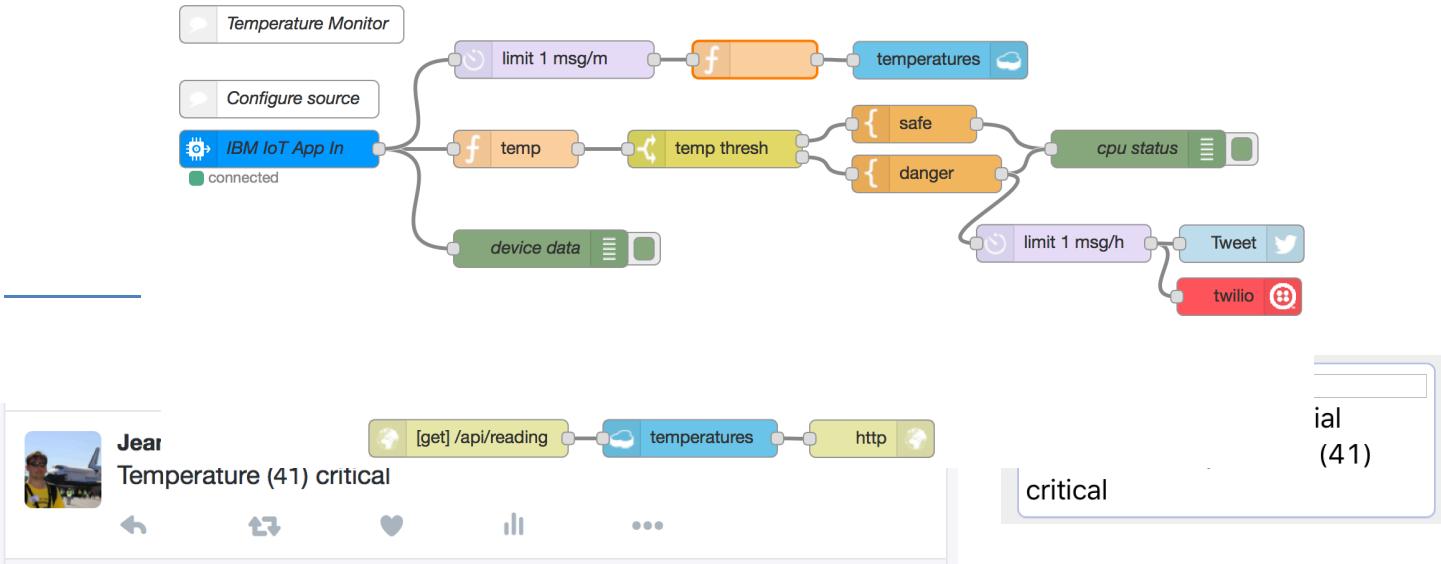


Table of Contents

Topics covered: Intel IoT Gateway, Grove sensors, Node-RED, Watson Internet of Things Platform, Cloudant NoSQL Database, Twitter, Twilio, and Weather Company Data for IBM Bluemix.

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 Bluemix 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 Bluemix. If the temperature reaches 40°C, we'll send a tweet via Twitter and a text message via Twilio. We will retrieve outside weather data and display it on a 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	41
Intel NUC Developer Hub Overview	53
Starting Node-RED for Blinky LED	41
Adding Sensors and Processing to the Gateway	53
Add Watson IoT nodes and OS Monitoring	41
Creating an IoT application in IBM Bluemix	28
Connect to Twitter and Tweet High Temperature	34
Connect to Twilio and Text High Temperature	36
Create a Node-RED flow on Intel NUC	37
Connecting Temperature Sensor to IoT Platform	41
Store Temperature Into Cloudant NoSQL Database	43
Retrieve Temperatures From Cloudant NoSQL DB	47
Send Weather Alerts from Watson IoT to the Intel Gateway	49



```
[{"_id": "bf263d883e2946bccf426eff410d76cb", "_rev": "1-b91cdf81c2ea48efaf8bdeff95213691", "time": 1470947934516, "temp": 29}, {"_id": "bf263d883e2946bccf426eff41258098", "_rev": "1-a8a28b2abd28dfc8e1d50ceef12fa0cc6", "time": 147094795007, "temp": 29}, {"_id": "d346612f6fabfd5359759c7d889098e4", "_rev": "1-d15dbcef08e6f098664e85248c493054", "time": 1470948055512, "temp": 29}, {"_id": "d346612f6fabfd5359759c7d88a71d25", "_rev": "1-0358ac7df1770c79c6acaf102bcbb24e", "time": 1470948116011, "temp": 29}]
```

Getting Started with Grove IoT Commercial Developer Kit

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

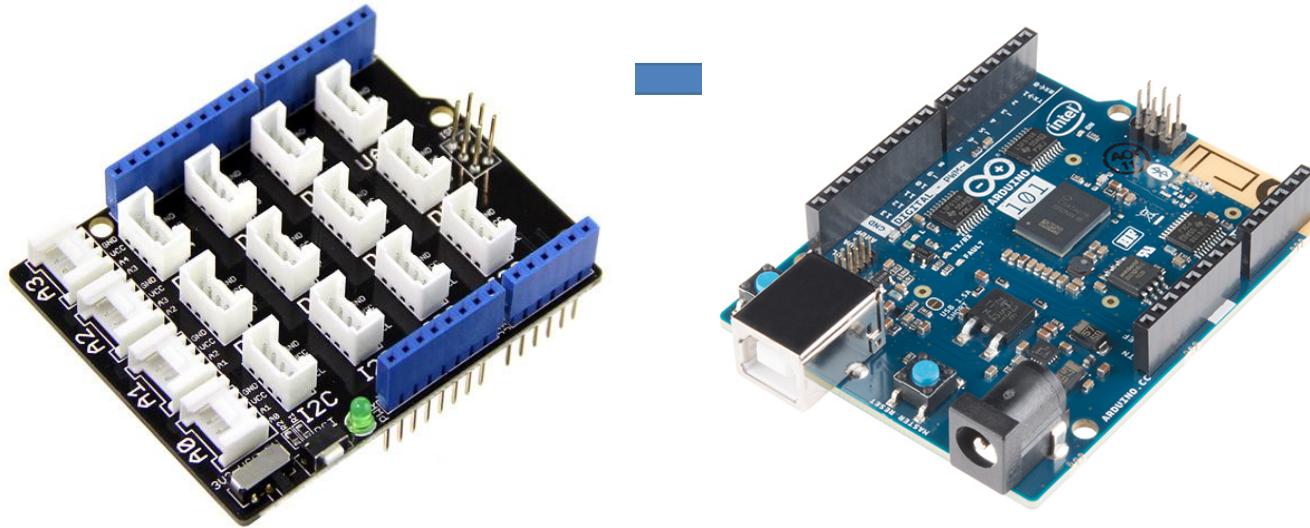


Arduino 101 is not included in Grove IoT Commercial Development Kit and will be given separately.

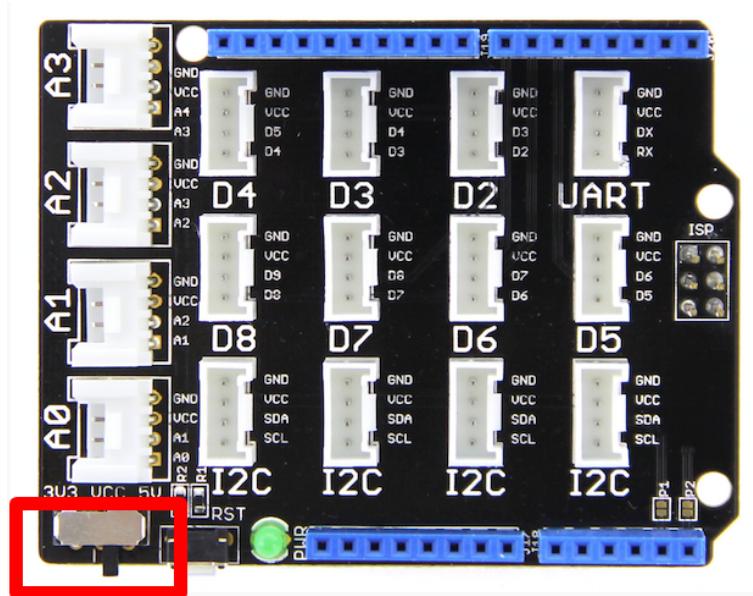


Step 2 – Connect Base shield to Arduino 101

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

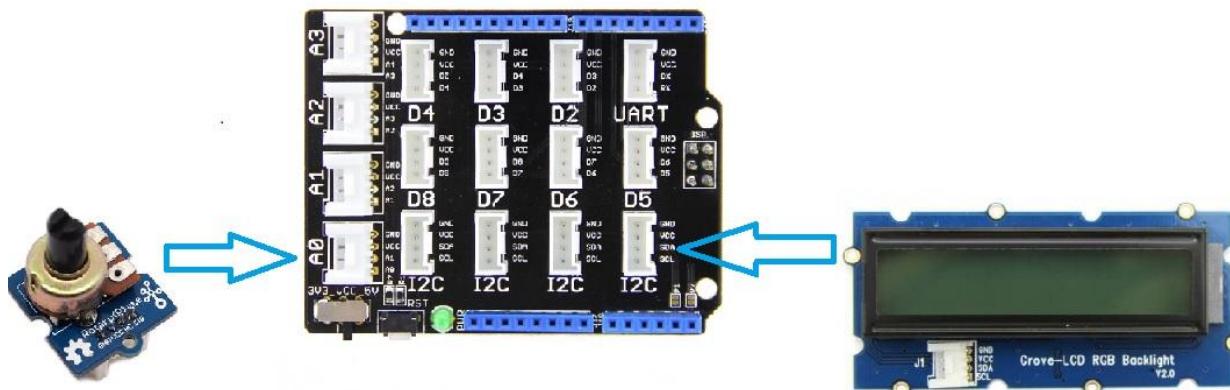


Step 3 – Move Base Shield switch towards 5 Volt



Step 4 – Connect LCD and Rotary Angle to Arduino 101

Connect LCD to any I2C pin of base shield and rotary angle to A0 pin of base shield with the help of a USB cable as shown in below image.

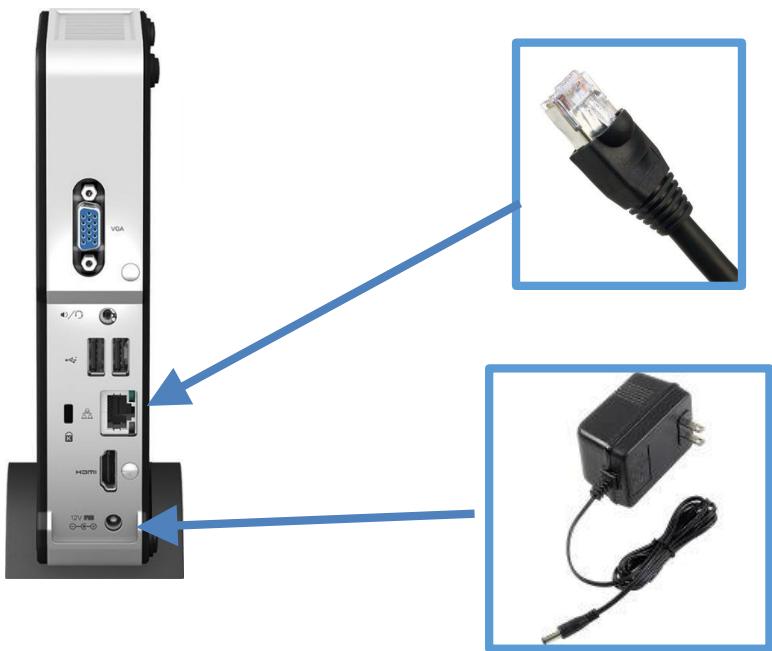


Step 5 – Connect Arduino 101 and Intel IoT Gateway via USB A-B Cable

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

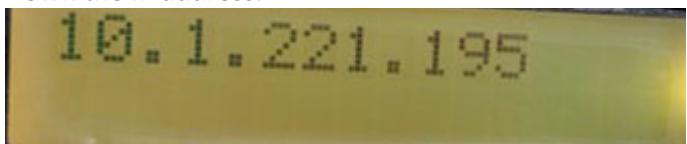


Step 6 – Connect Intel IoT gateway with Ethernet cable and Power adapter



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

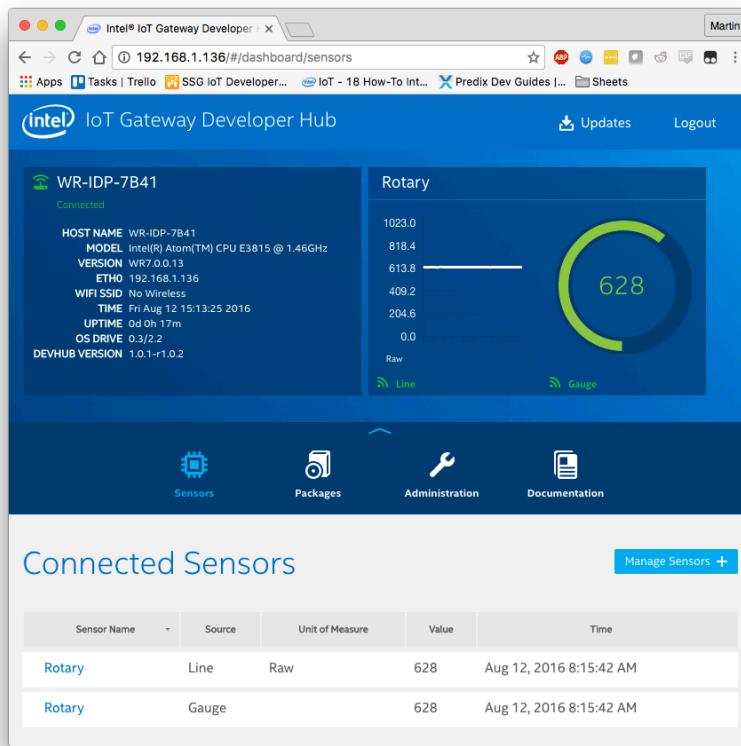
- Login to the Intel® IoT Gateway Developer Hub using root for both the username and the password.
- Use the Intel® IoT Gateway Developer Hub to explore and extend the basic capabilities of the Gateway.

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

The screenshot shows the Intel IoT Gateway Developer Hub dashboard. At the top, there's a header with the Intel logo, user name 'Martin', and navigation links for 'Updates 1427' and 'Logout'. Below the header, on the left, is a section for device information: 'WR-IDP-7B41' (Connected), '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', and 'DEVHUB VERSION 1.0.1-r1.0.2'. To the right of this is a circular gauge chart titled 'Rotary' with a value of 628, ranging from 0.0 to 1023.0. Below the gauge are two buttons: 'Line' and 'Gauge'. At the bottom of the dashboard are four navigation icons: Sensors (a gear icon), Packages (a speaker icon), Administration (a wrench icon), and Documentation (a book icon). The main content area is titled 'Installed Packages' and contains a table with three rows of data. The columns are: Package Name, Category, Launch Capability, Update, Running, Auto Run, and Activity State. The packages listed are acpid, alsamixer, and alsamixer-base.

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

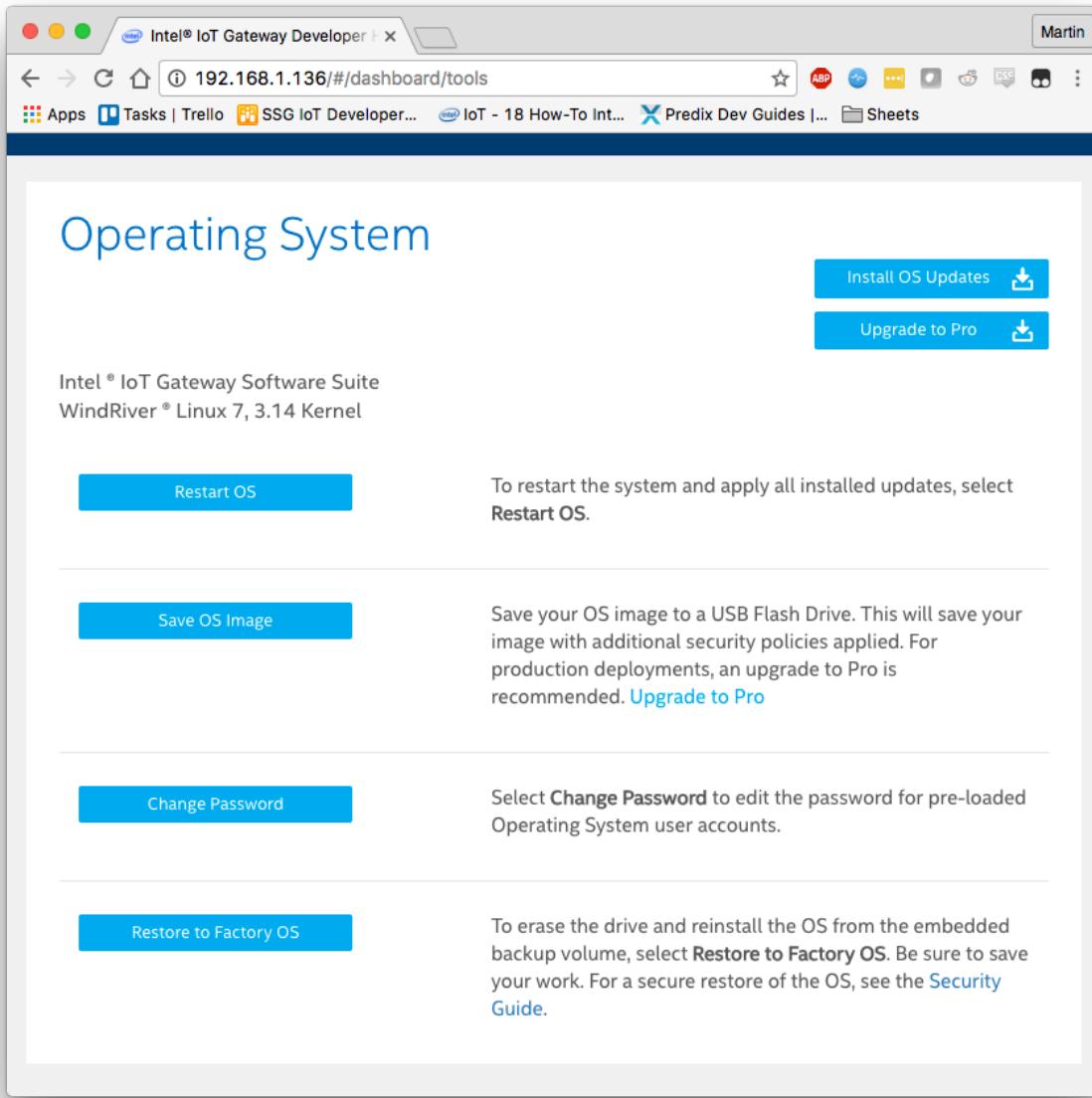
View of Packages screen

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



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 DevHub interface:

The page title is "Intel® IoT Gateway Developer". The URL in the address bar is "192.168.1.136/#/dashboard/tools". The browser tab is "Untitled". The user is logged in as "Martin".

Top navigation bar items include: Apps, Tasks | Trello, SSG IoT Developer..., IoT - 18 How-To Int..., Predix Dev Guides |..., Sheets.

A "Restore to Factory OS" button is present, with a note: "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

- Node-RED**: Icon shows a red square with a white gear and a wrench. A "Launch" button is below it.
- APP CLOUD**: Icon shows a blue cloud with three white gears inside. A "Launch" button is below it.
- Gears**: Icon shows two interlocking black gears. A "Launch" button is below it.
- File with arrow**: Icon shows a blue folder with a white right-pointing arrow. 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.

Enable button (blue)

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.

Configure button (blue)

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 "Tutorials" section of the documentation:

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 OSX 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.



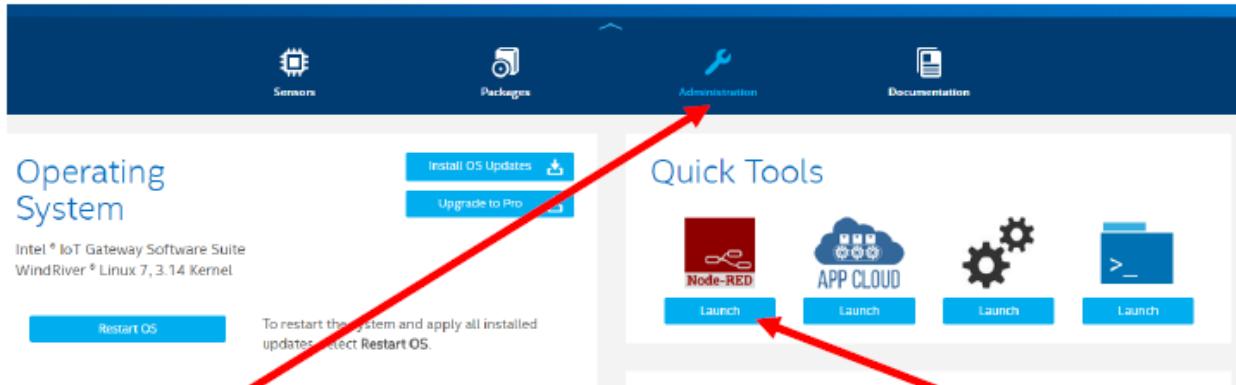
- Double-click putty.exe on your computer to launch PuTTY.
- Enter IP address of the Gateway
- You can login with the username **root** and the password **root**.

For Mac and Linux Users:

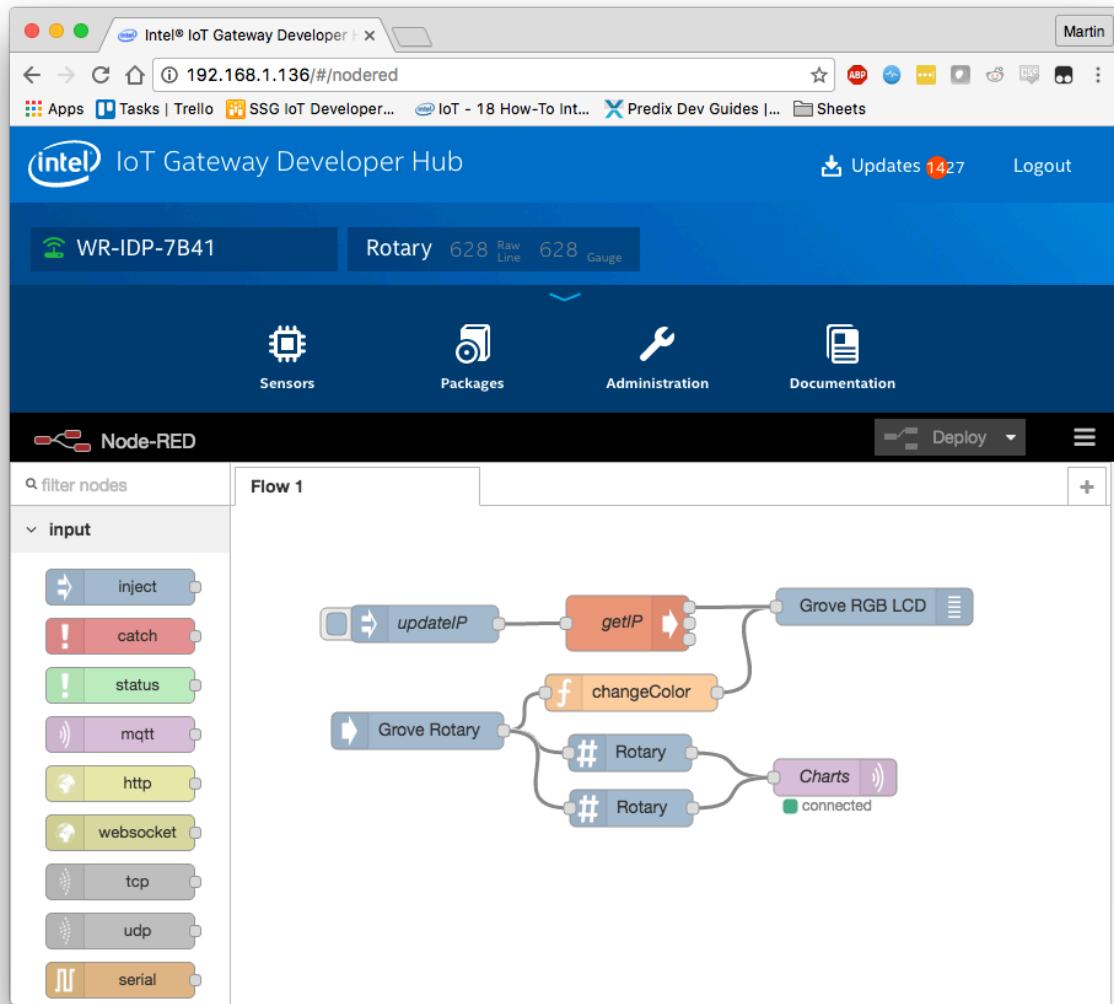
- Open a Terminal
- Type `$ ssh root@<<IP Address>>`. Replace <<IP Address>> with the IP address of your gateway.
- Enter **root** as password

Starting Node-RED for Blinky LED

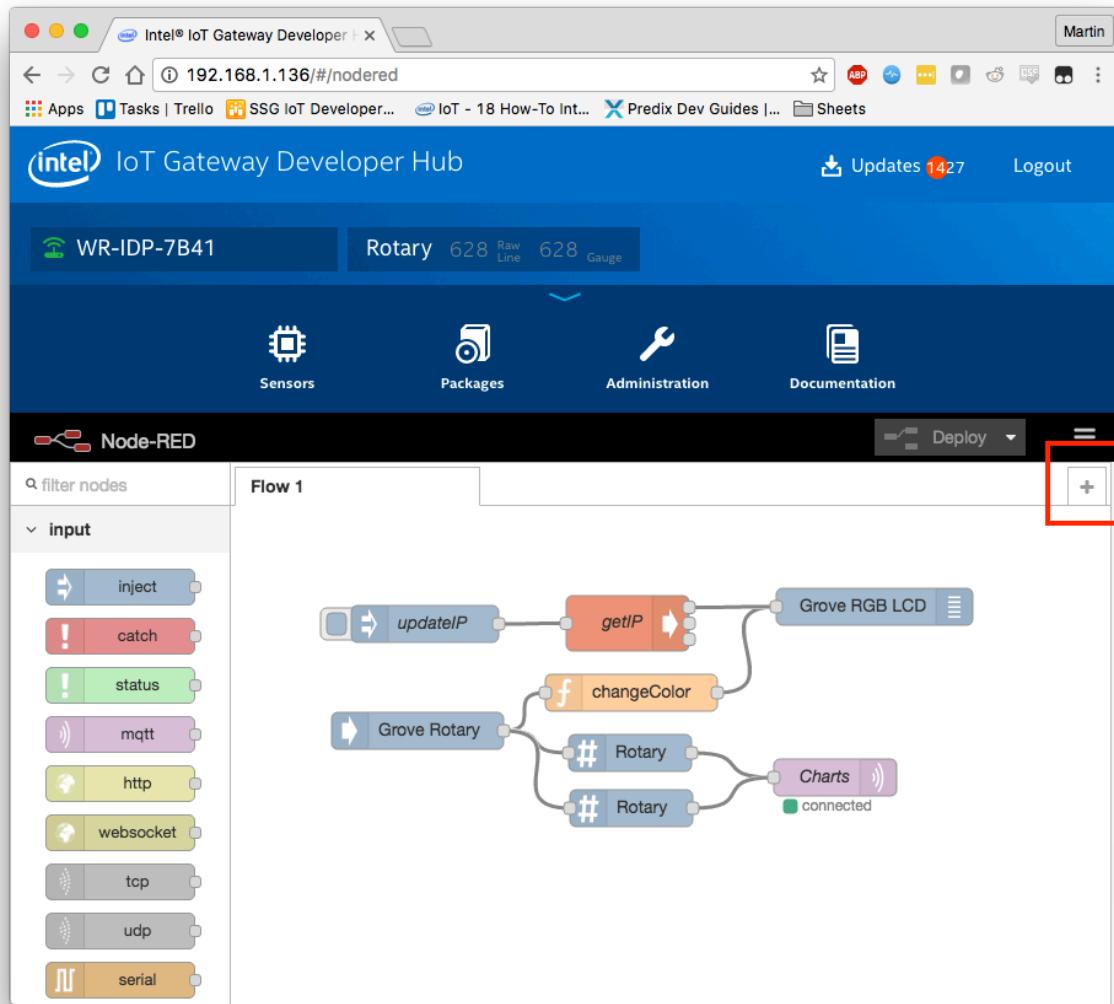
- Go to Development Hub by entering the IP Address assigned to your Gateway (Eg. <http://192.168.1.1>) in your browser.
- To load the Node-RED interface, click **Administration** on the navigation ribbon. Click **Launch** under the Node-RED icon.



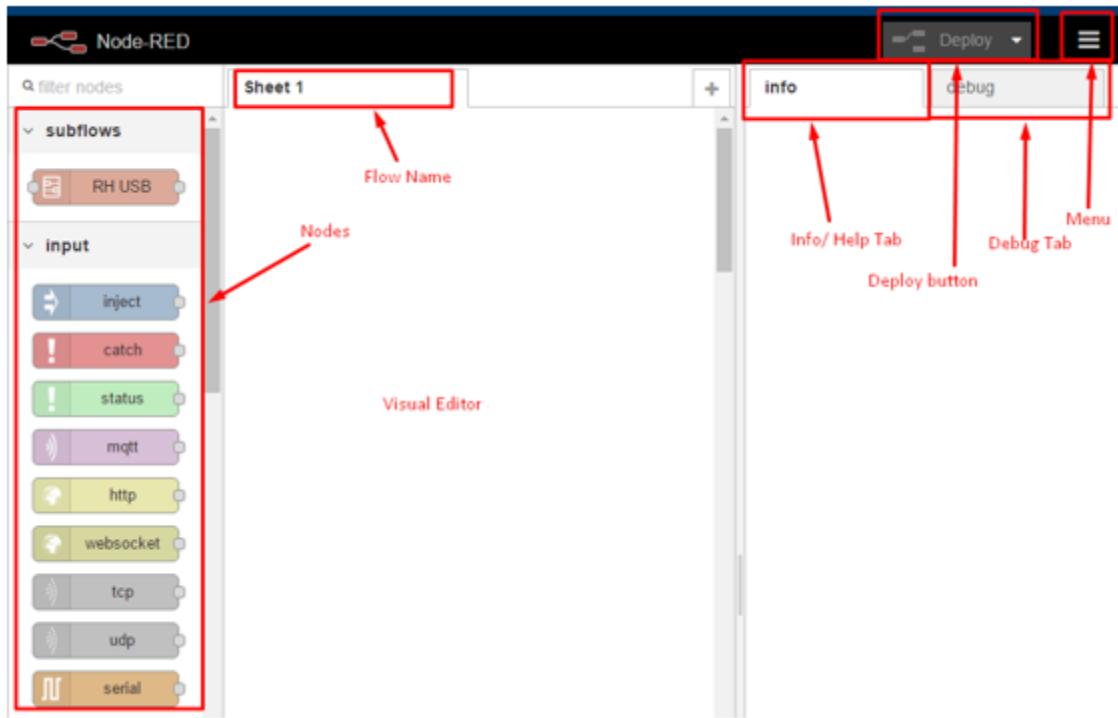
- Node-Red is a visual tool for writing the Internet of Things (IoT).
- Enter username and password as **root** and **root** respectively.



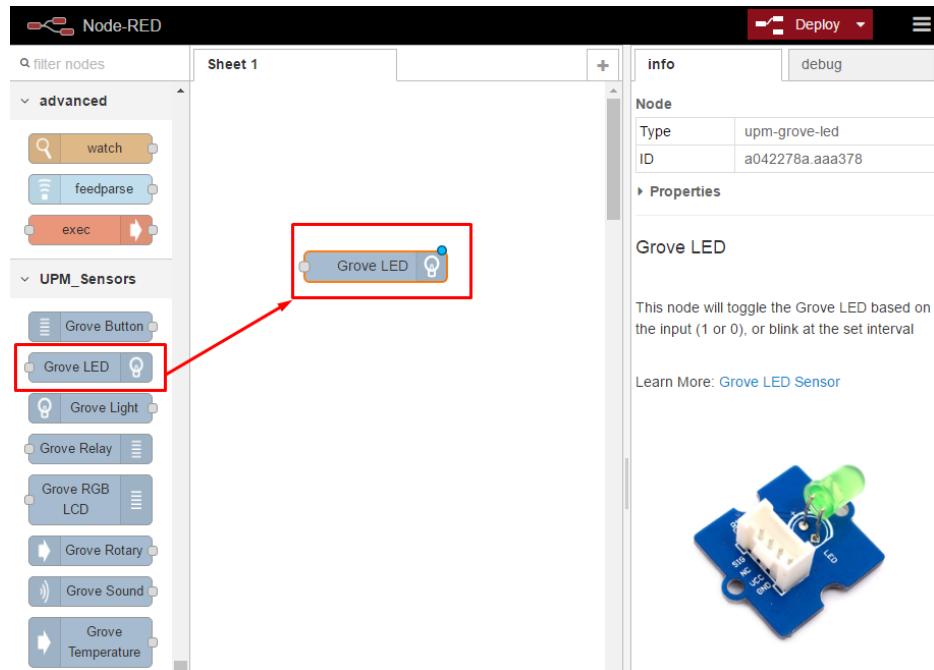
You will see the default application that sends the IP address to the LCD screen and the rotary angle sensor data to the chart on the dashboard. Leave this flow as is for now.



Create a new flow to program the rest of the exercises. Click the + button in the upper right corner of the Node-RED interface.



- 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 – Blinks LED
 - Platform – Firmata
 - Pin – D13
 - Mode- Blink
 - Interval(ms) – 1000

Then Click OK to save changes.

Edit upm-grove-led node

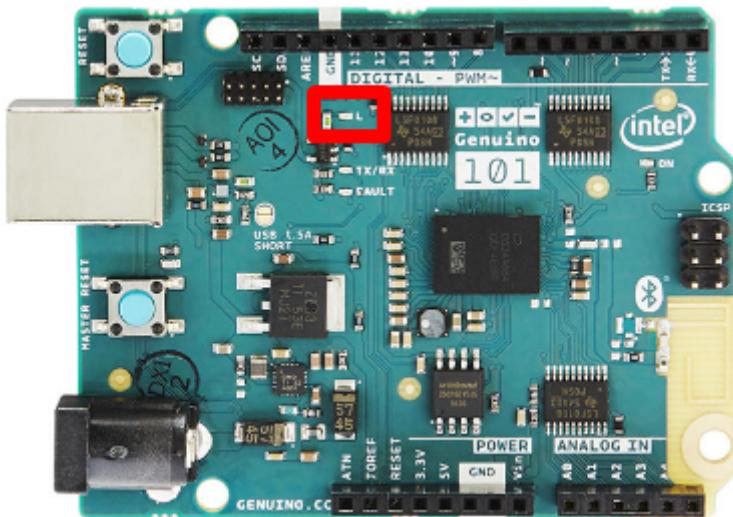
Name	Blinky-LED
Platform	Firmata
Pin	D13
Mode	Blink
Interval (ms)	1000

Ok **Cancel**

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

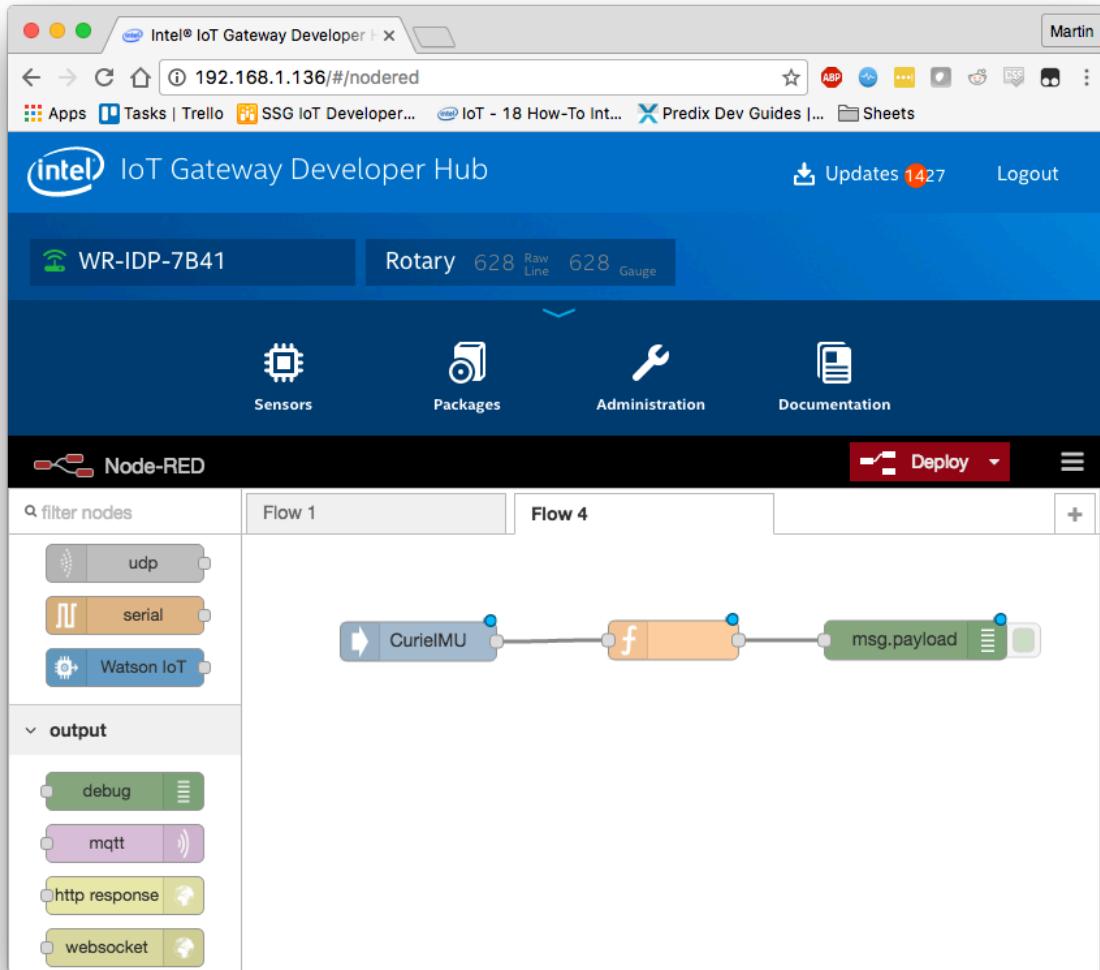
The screenshot shows the Node-RED interface. On the left, there's a sidebar with categories like 'advanced' and 'UPM_Sensors'. In the center, a flow is displayed with a single node labeled 'Blinky-LED'. On the right, there's a properties panel for this node, showing details such as Name: Blinky-LED, Type: upm-grove-led, and ID: a042278a.aa378. At the top right of the interface, a red box highlights the 'Deploy' button.

- The Arduino 101 LED on board LED will start blinking

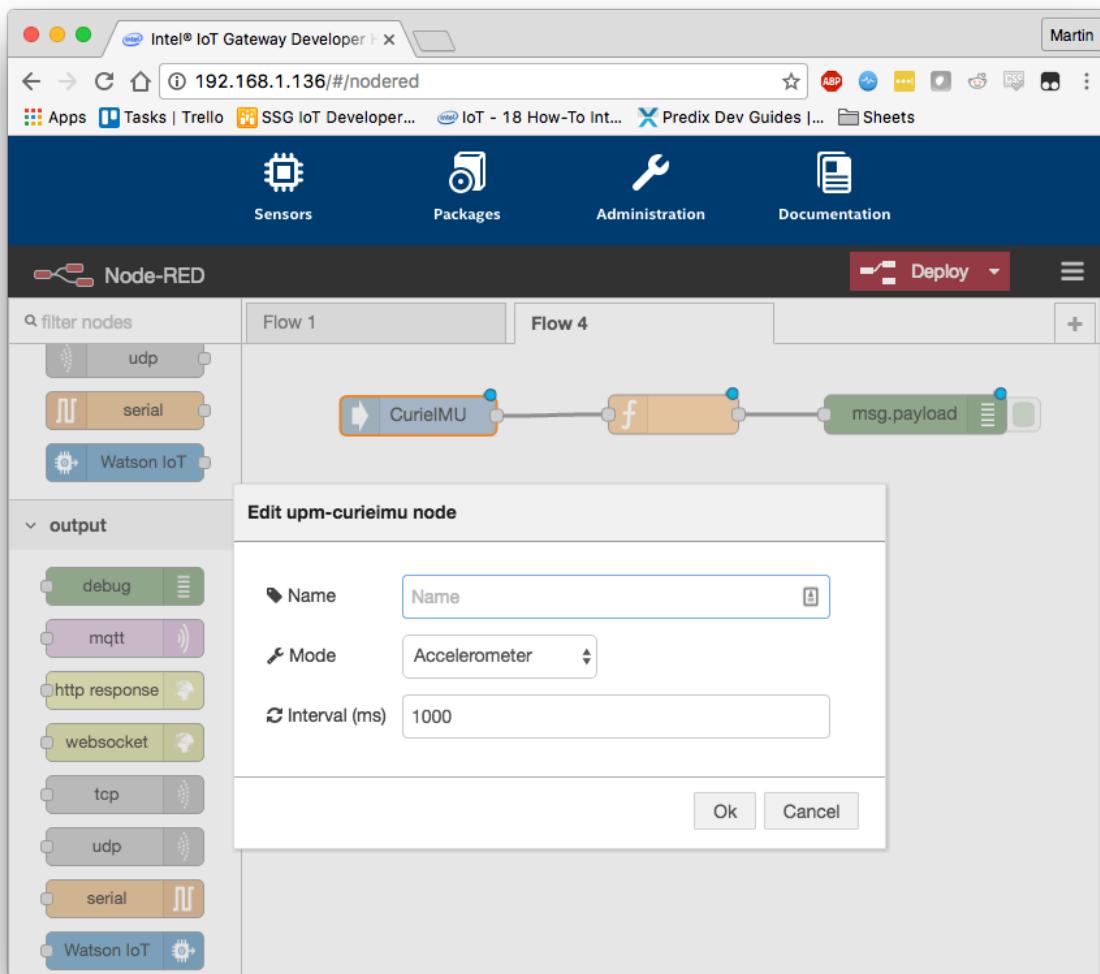


Adding Sensors and Processing to the Gateway

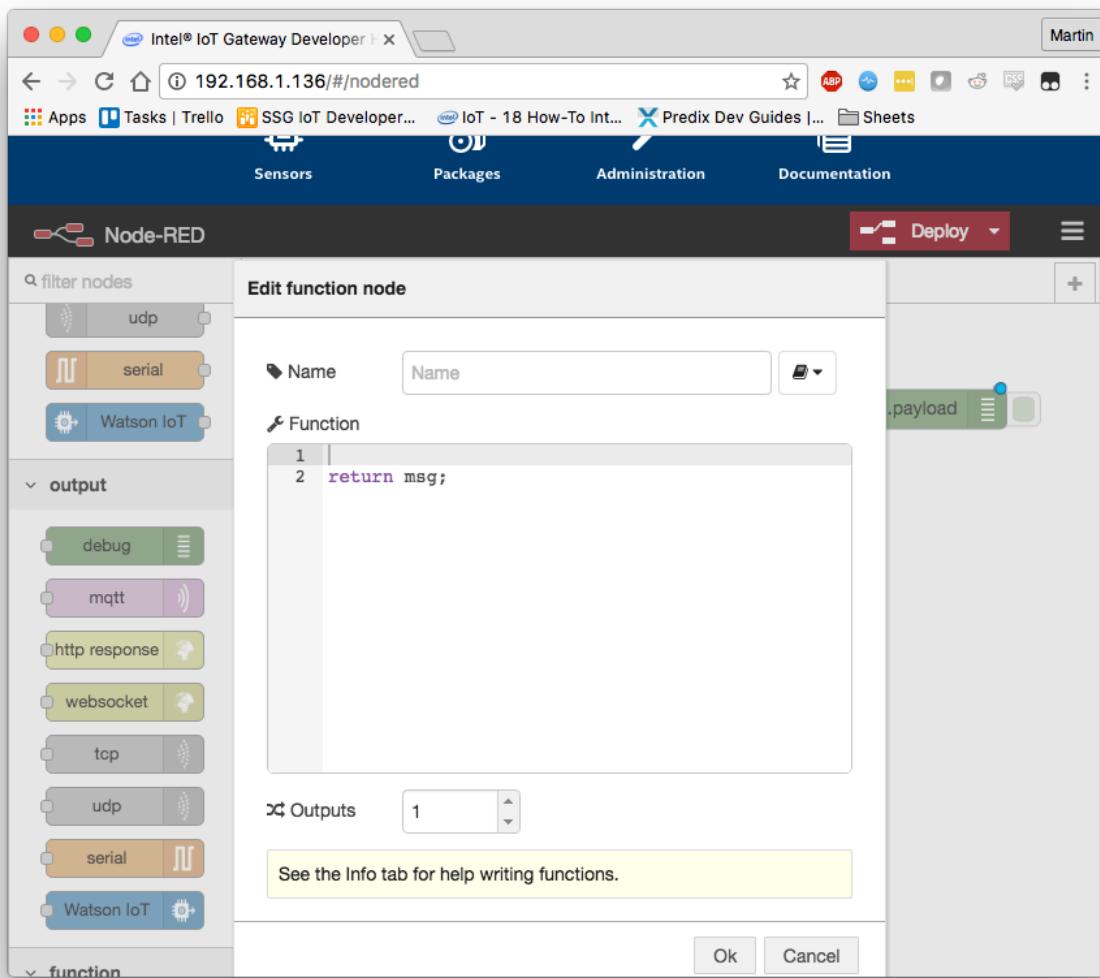
Now that you have a basic understanding of how to interact with the Gateway let's move on to adding some more sensor data and processing to the gateway.



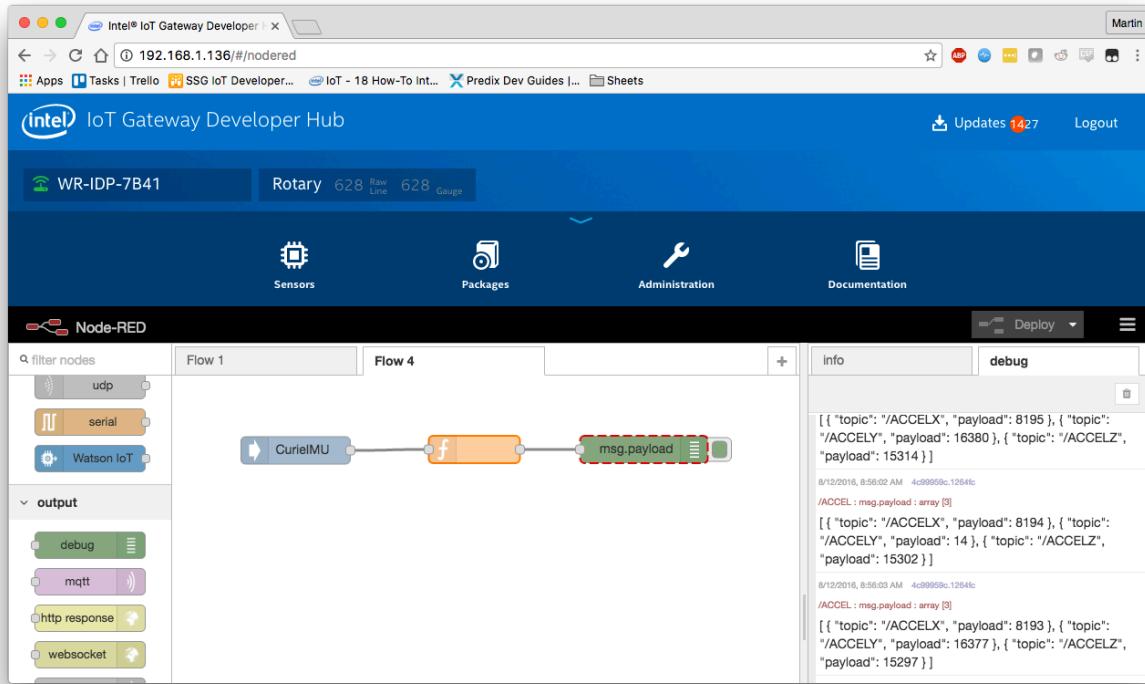
- From **UPM Sensors** add **CurielMU** node
- From **function** add a **function** node
- From **output** add a **debug** node
- Wire the nodes together



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.

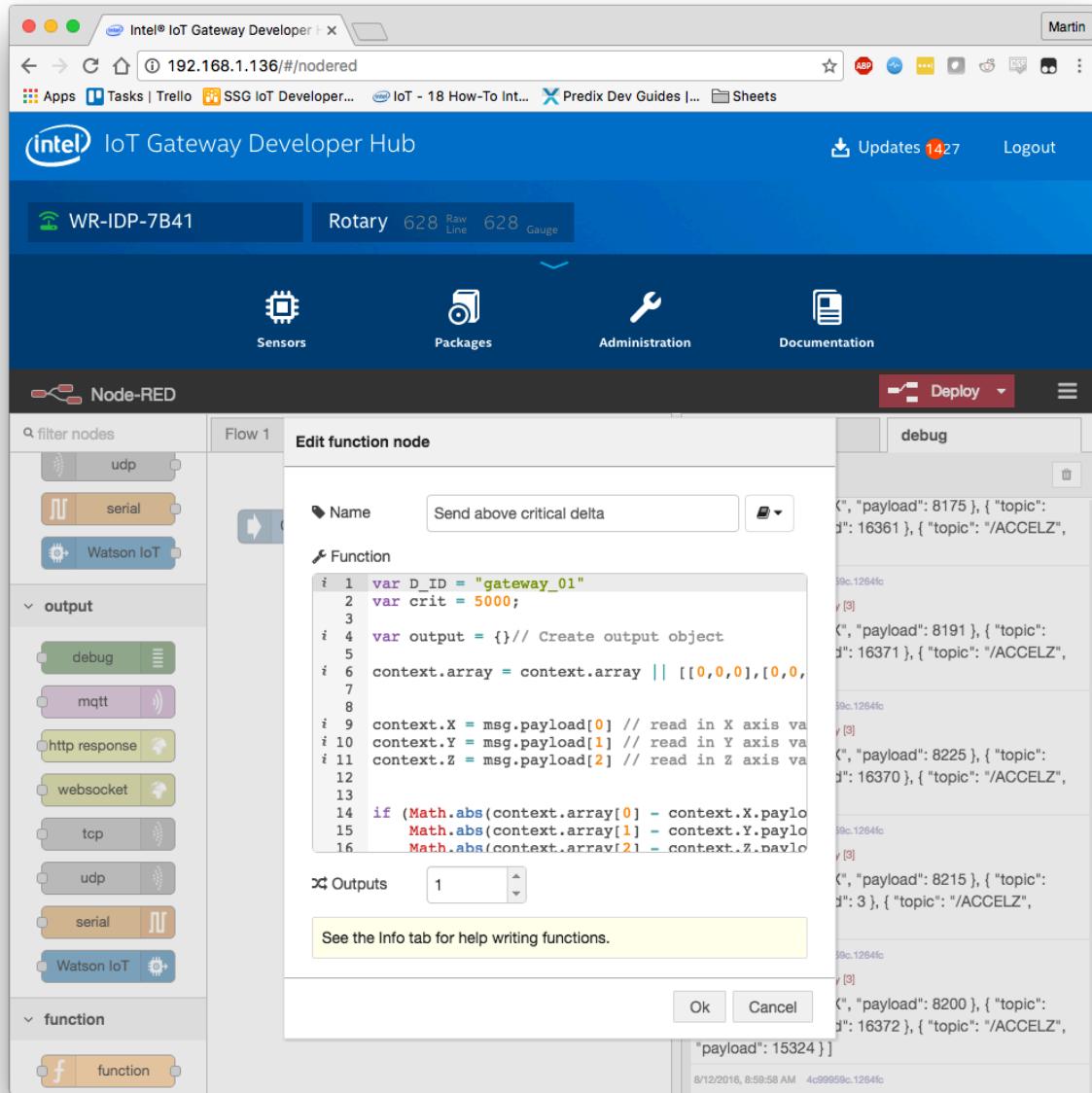


Hit deploy to run the flow. Open the debug tab to see the output from the function node.

Note: you may have to drag from the right to open the debug tab.

As you can see the CurielMU 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 - which is also available at :
<https://ibm.biz/BdrgCd>

```

var D_ID = "gateway_01"
var crit = 5000;

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

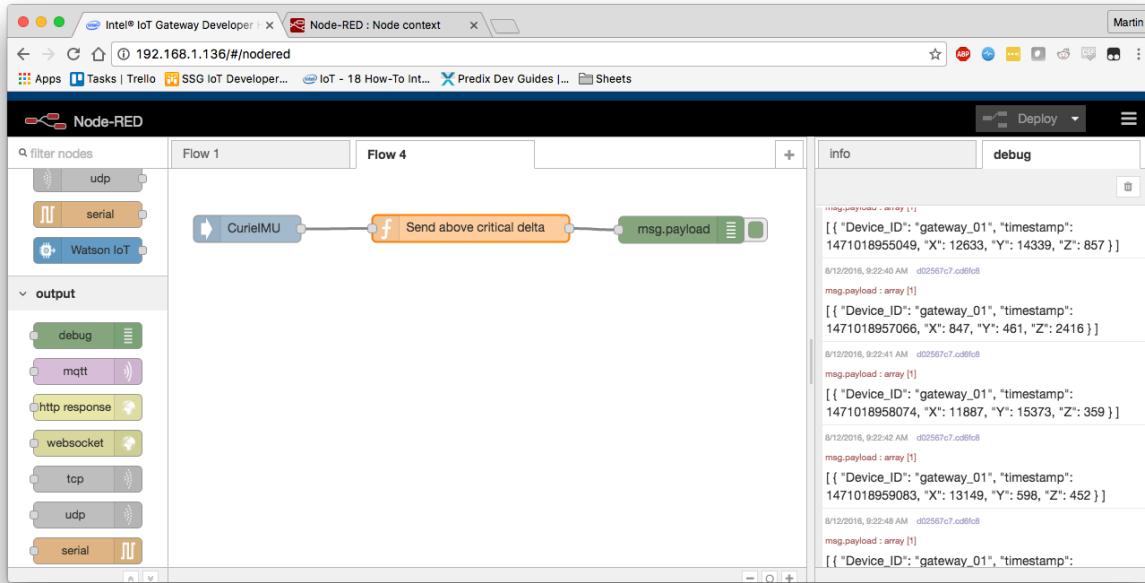
context.X = msg.payload[0]
context.Y = msg.payload[1]
context.Z = msg.payload[2]

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 a 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.

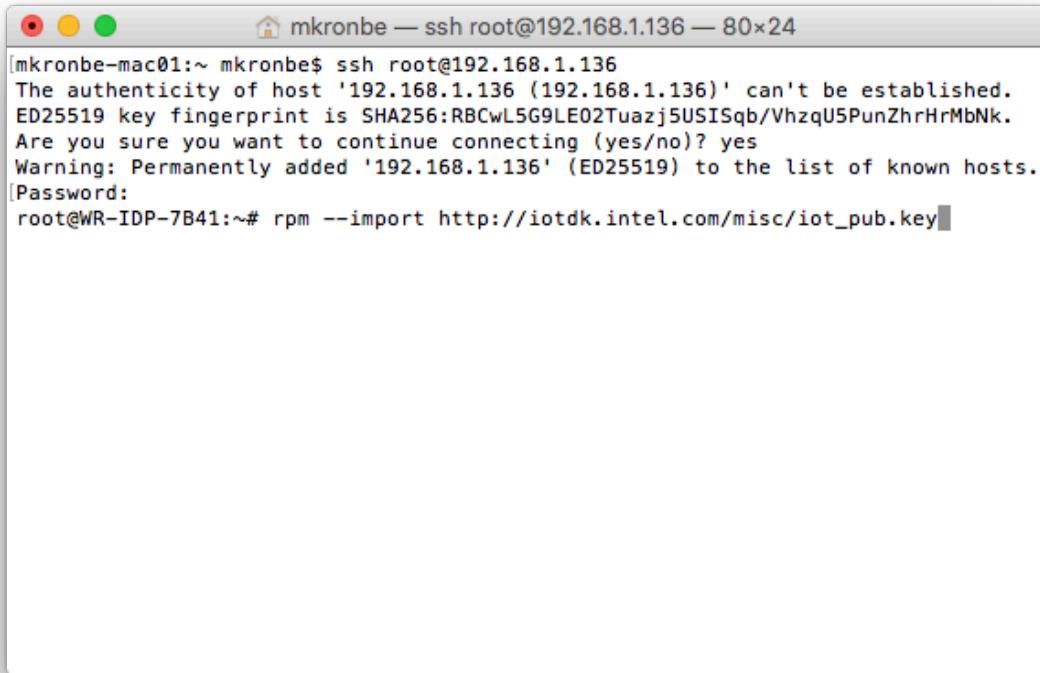


Hit deploy to run the code. The debug output should only display the accelerometer data when the Arduino 101 board registers movement.

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 ease of life features.



The screenshot shows a terminal window titled "mkronbe — ssh root@192.168.1.136 — 80x24". The window contains the following text:

```
[mkronbe-mac01:~ mkronbe$ ssh root@192.168.1.136
The authenticity of host '192.168.1.136 (192.168.1.136)' can't be established.
ED25519 key fingerprint is SHA256:RBCwL5G9LE02Tuazj5USISqb/VhzqU5PunZhrHrMbNk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.136' (ED25519) to the list of known hosts.
[Password: ]root@WR-IDP-7B41:~# rpm --import http://iotdk.intel.com/misc/iot_pub.key]
```

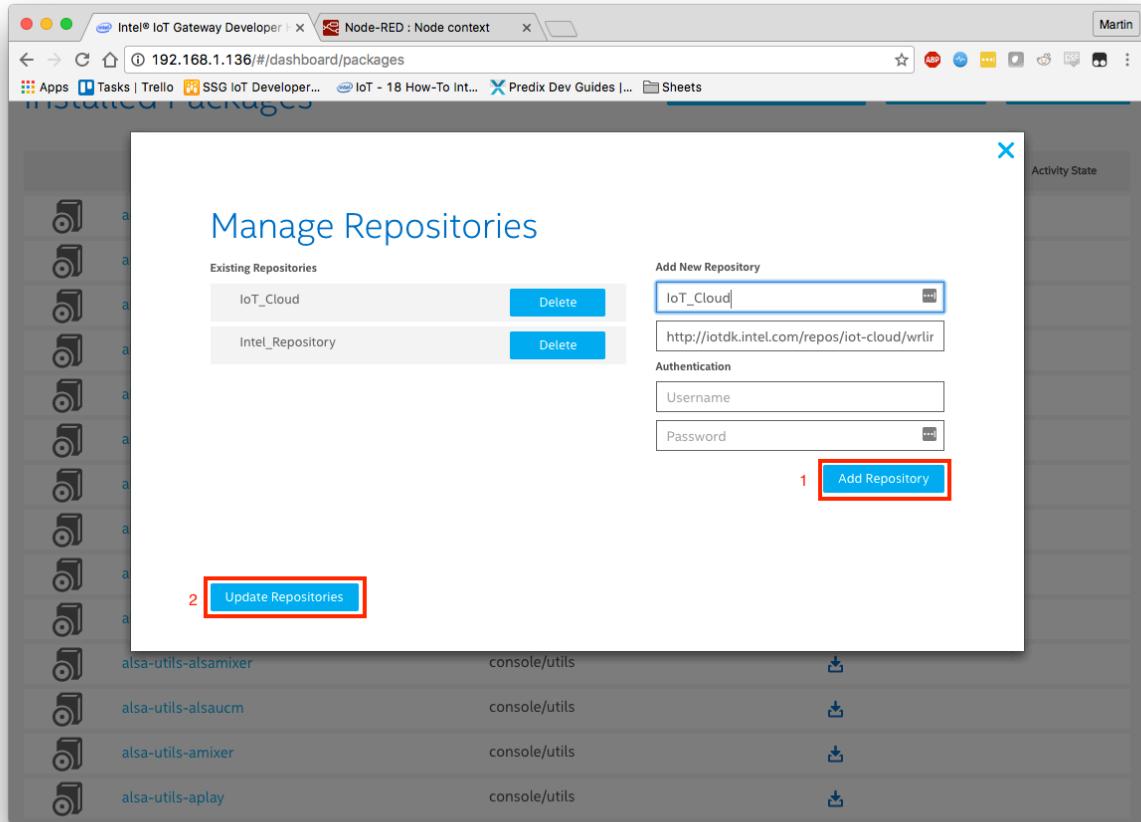
SSH into the ip address displayed on the LCD as root. The password is root. Once you are connected to the command line enter the following to add the repository key to rpm:

```
# rpm --import http://iotdk.intel.com/misc/iot_pub.key
```

Also install the Node-RED OS monitoring nodes with:

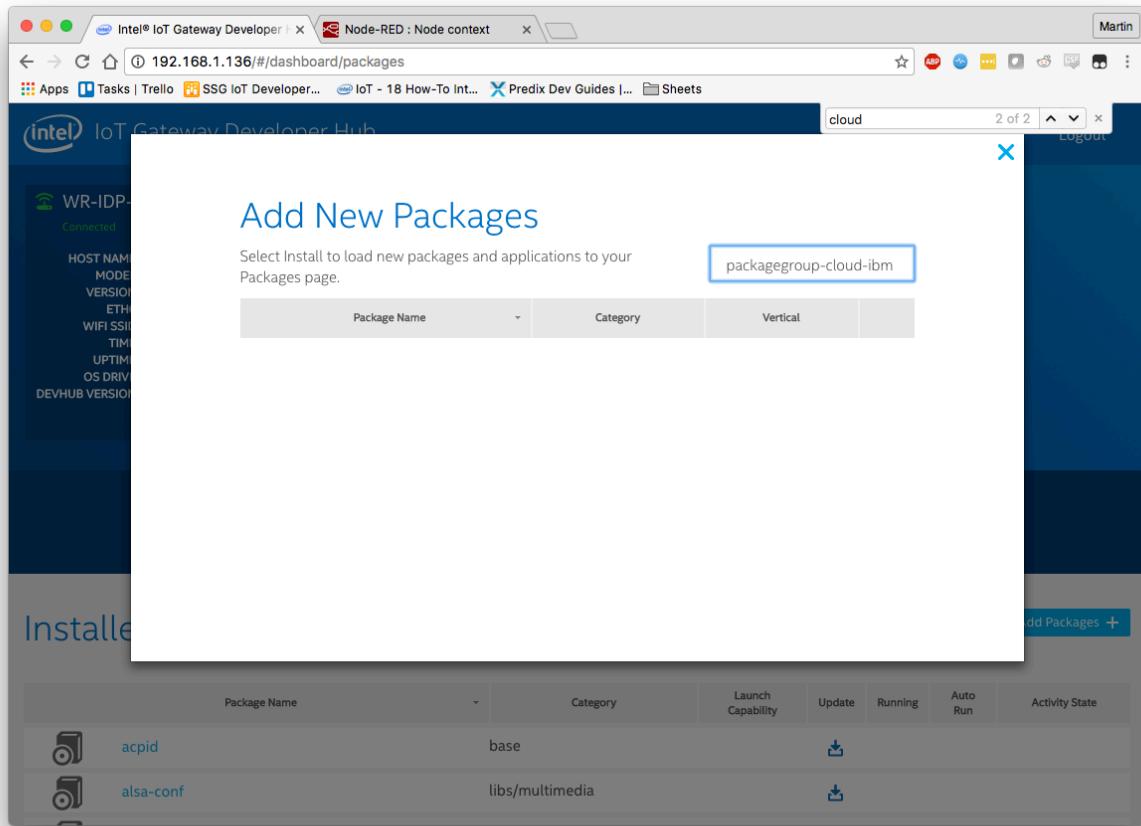
```
# npm install node-red-contrib-os -g
```

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



Navigate back to the repository manager on the Gateway.

- Select “Add Repo”
- In the Name Field enter “IoT_Cloud”
- In the URL Field enter <http://iotdk.intel.com/repos/iot-cloud/wrlinux7/rpcl13/>
- Click “Add Repository”
- After it is finished adding click “Update Repositories”
- Close the manage repository window

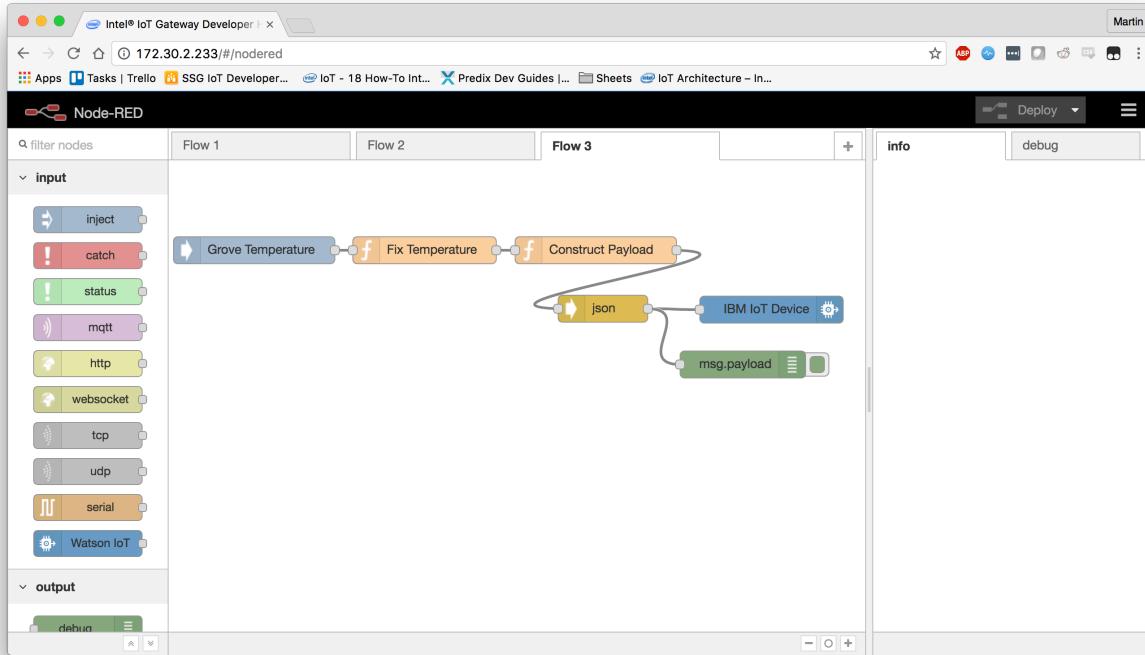


- Open “Add Packages”
- Search for “packagegroup-cloud-ibm”
- Click “Install” next to the package name
- Close the Add New Package window

The screenshot shows the Intel IoT Gateway Developer dashboard at 192.168.1.136/#/dashboard/tools. The top left displays system information for the device WR-IDP-7B41, including its host name, model (Intel Atom E3815 @ 1.46GHz), version (WR7.0.0.13), and uptime (0d 0h 30m). To the right is a circular gauge chart titled "Rotary" with a value of 383, showing raw data from 0 to 1023. Below the gauge are two tabs: "Line" and "Gauge". The bottom navigation bar includes links for Sensors, Packages, Administration, and Documentation. On the left, under the "Operating System" section, there are buttons for "Install OS Updates" and "Upgrade to Pro". A red box highlights the "Restart OS" button. To the right, under "Quick Tools", there are four items: Node-RED (Launch), APP CLOUD (Launch), Gears (Launch), and a terminal icon (Launch).

Restart OS from the Administration tab

Create Temperature Flow



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
- From **function** add 2 **function** nodes
- From **function** add a **json** node
- From **output** add a **Watson IoT** 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  
    }  
  }  
};
```

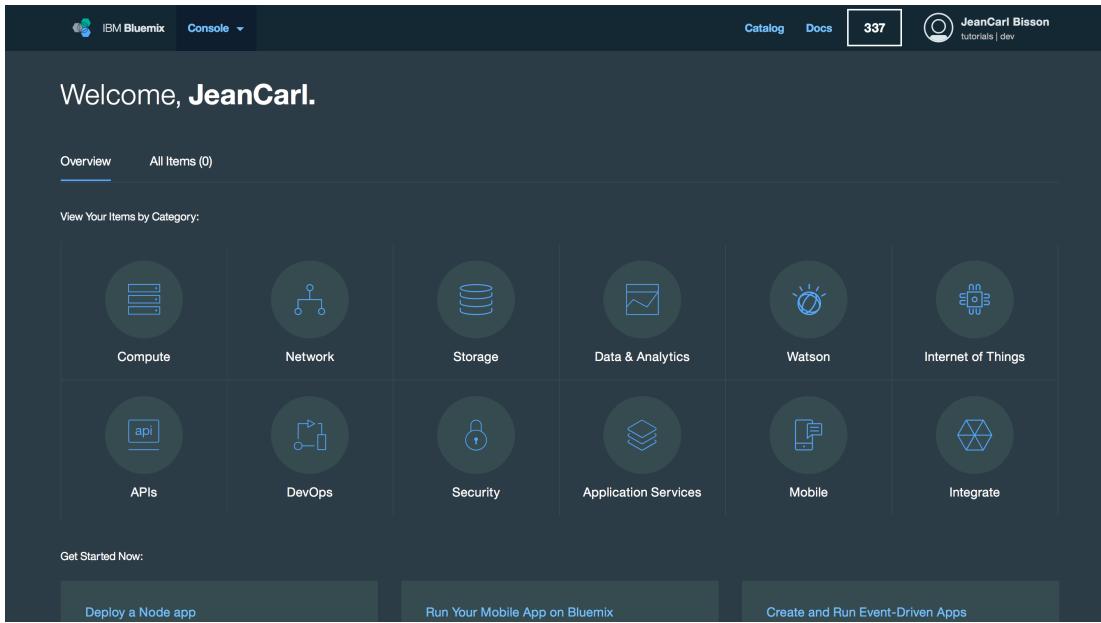
This will format the temperature to allow Watson IoT to ingest the data easier.

Once you have configured the IBM IoT Device nodes your flow will send local temperature data to IBM Watson IoT cloud services.

Creating an IoT application in IBM Bluemix

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

1. Open a web browser and go to your IBM Bluemix dashboard, <http://bluemix.net>. This is the IBM Bluemix 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.

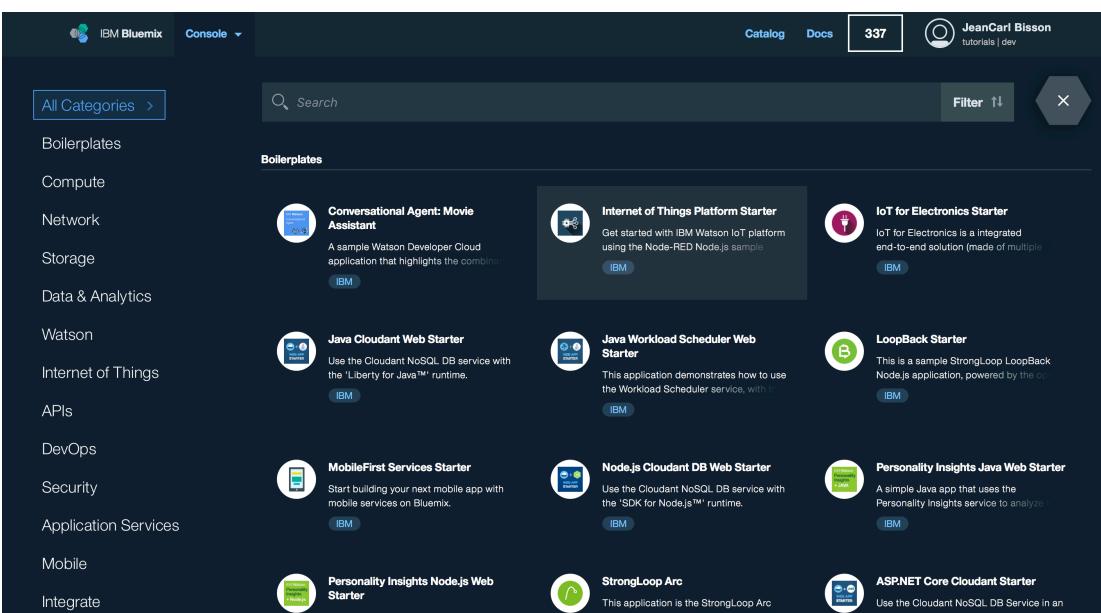


The screenshot shows the IBM Bluemix dashboard. At the top, there's a navigation bar with 'IBM Bluemix' and 'Console'. On the right, there are links for 'Catalog', 'Docs', and a user profile with '337' notifications. Below the navigation is a welcome message 'Welcome, JeanCarl.' and a 'Overview' section with a link to 'All Items (0)'. A grid of service categories is displayed in two rows:

Compute	Network	Storage	Data & Analytics	Watson	Internet of Things
APIs	DevOps	Security	Application Services	Mobile	Integrate

Below the grid, there's a 'Get Started Now:' section with three buttons: 'Deploy a Node app', 'Run Your Mobile App on Bluemix', and 'Create and Run Event-Driven Apps'.

2. IBM Bluemix 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. At the top, there's a navigation bar with 'IBM Bluemix' and 'Console'. On the right, there are links for 'Catalog', 'Docs', and a user profile with '337' notifications. Below the navigation is a search bar and a filter button. The left sidebar lists categories: All Categories, Boilerplates, Compute, Network, Storage, Data & Analytics, Watson, Internet of Things, APIs, DevOps, Security, Application Services, Mobile, and Integrate. The 'Boilerplates' section displays several tiles, with the 'Internet of Things Platform Starter' tile highlighted. Other visible tiles include 'Conversational Agent: Movie Assistant', 'Java Cloudant Web Starter', 'MobileFirst Services Starter', 'Personality Insights Node.js Web Starter', 'StrongLoop Arc', and 'ASP.NET Core Cloudant Starter'.

- Pick a unique name for your application. If you choose **myapp**, your application will be located at `http://myapp.mybluemix.net`. There can only be one “**myapp**” application registered in IBM Bluemix. 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.

IBM Bluemix Console

Create a Cloud Foundry Application

Internet of Things Platform Starter

App name: myapp

Host name: myapp Domain: mybluemix.net

Selected Plan:

SDK for Node.js™: Default Cloudant NoSQL DB: Shared

Internet of Things Platform: Free

VERSION: 0.5.02 TYPE: Boilerplate REGION: US South

- IBM Bluemix 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.

Cloud Foundry Applications

myapp Status: Your app is starting

Getting Started Overview Runtime Connections Logs Monitoring

View App

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

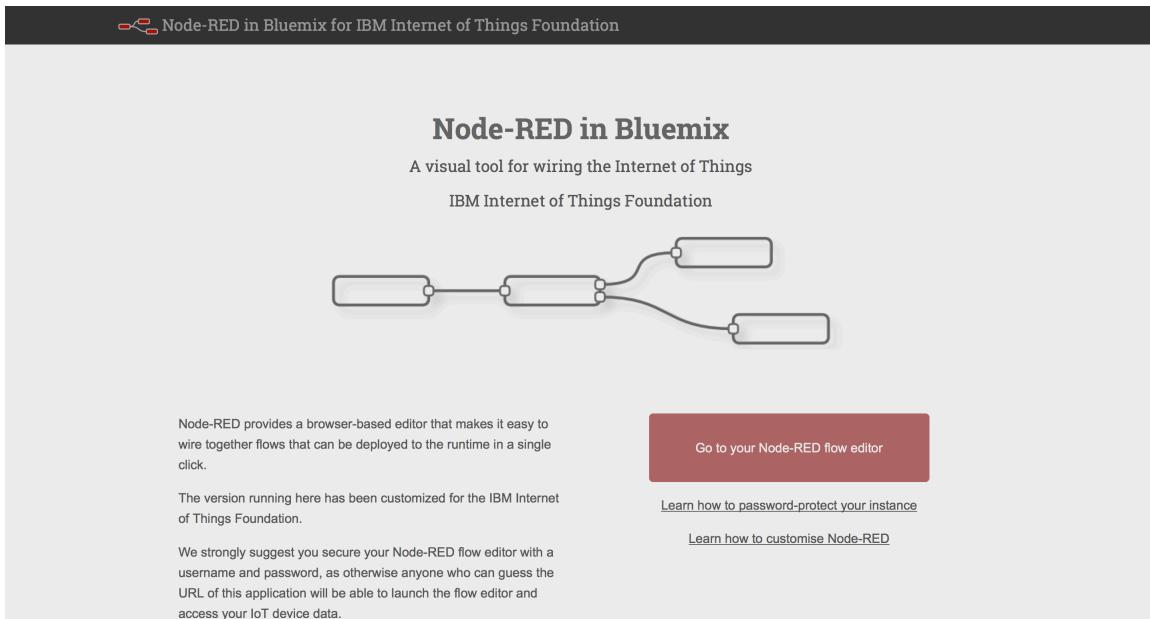
Cloud Foundry Applications

myapp Status: Your app is running

Getting Started Overview Runtime Connections Logs Monitoring

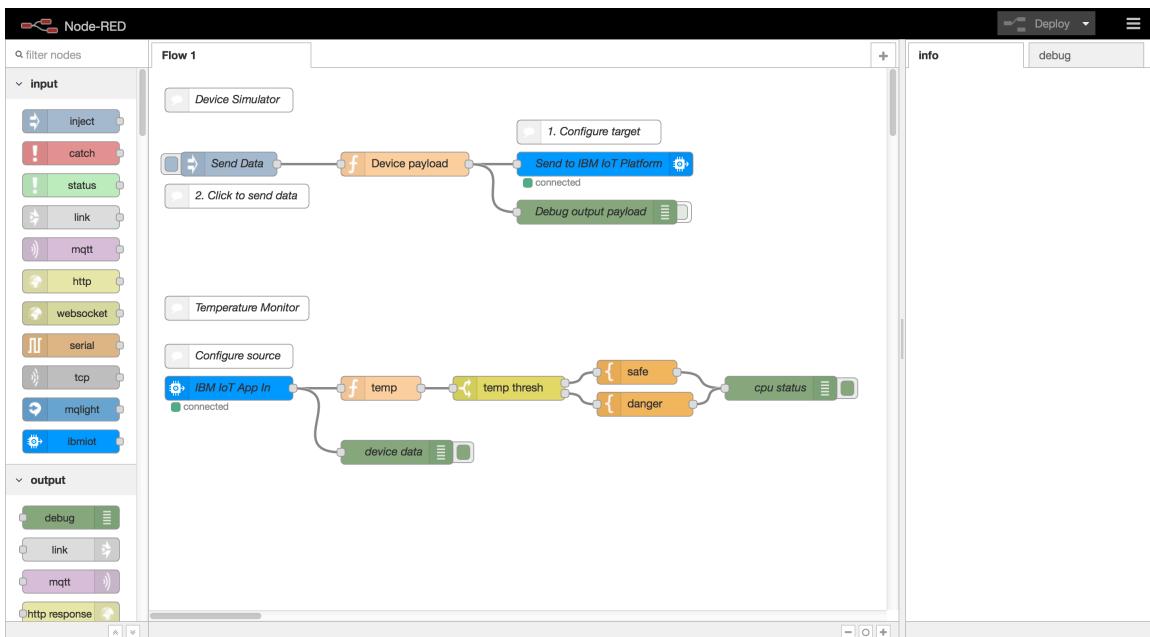
View App

6. This is 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.

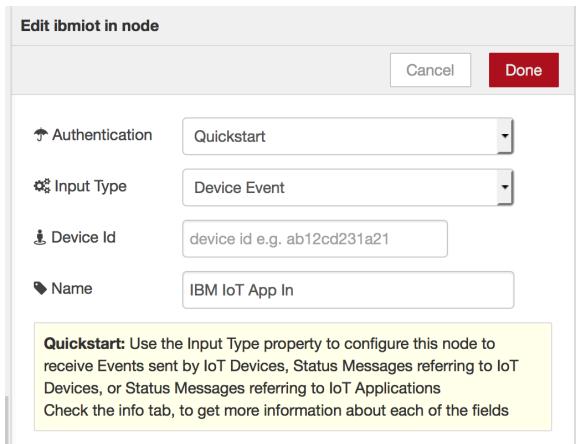


7. On the left side is a palette of nodes. Each node performs a defined function, configurable by dialog windows or by modifying the msg input. You can click on a node in the palette and find out what it does in the info tab on the right side of the screen. Drag and drop nodes and connect them together 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.

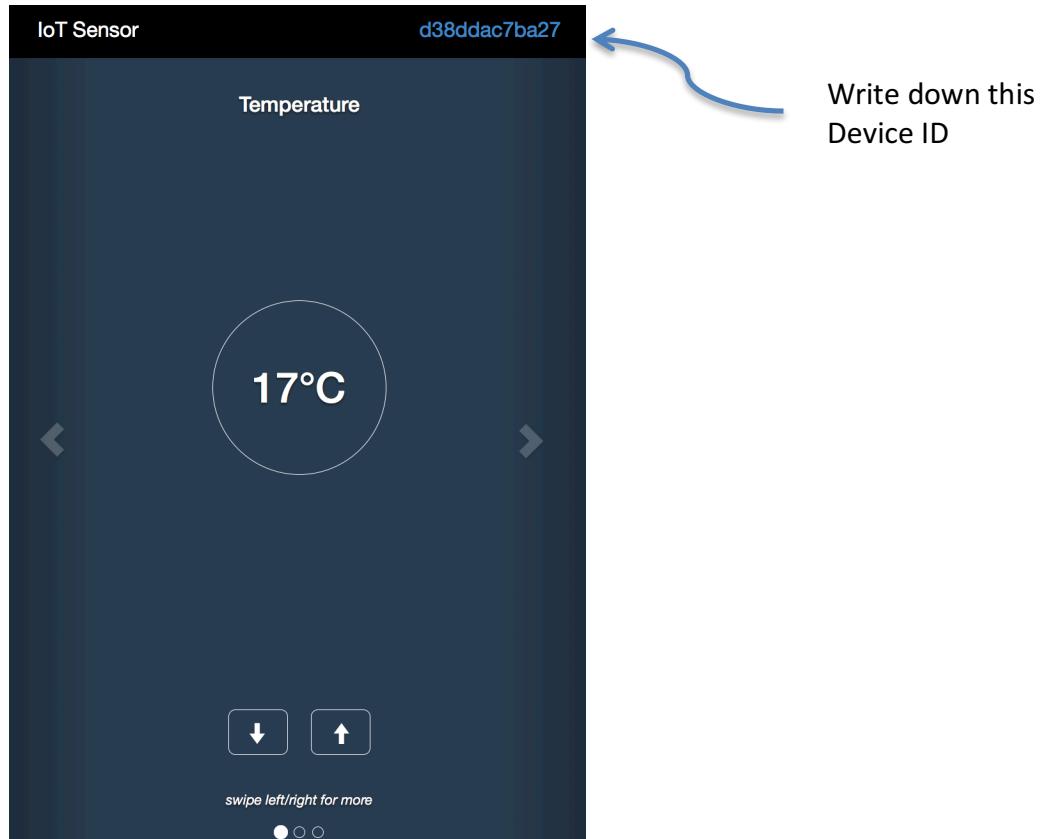
Let's begin by configuring the Watson IoT App In node. Double click on the Watson IoT App In node.



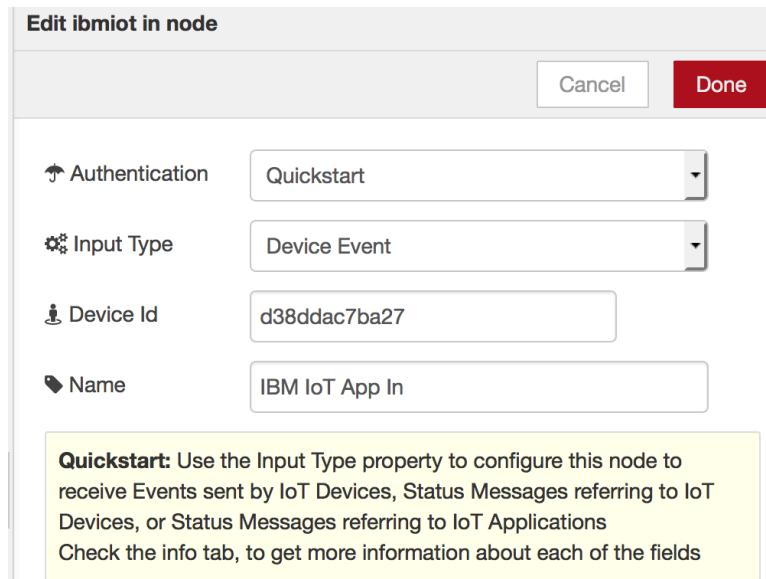
Nodes can be customized in different ways depending on what they do. The IBM IoT node begins a flow when a message is received from a specific device via the Watson IoT Platform. We need a Device Id of a device that is connected to the IoT Platform. We could use a real device, or we can simulate devices.



8. 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.



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



10. Click on the Deploy button in the top right of the screen to save and deploy your changes.

11. 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.

```

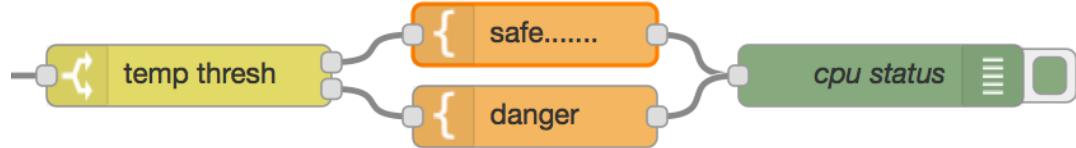
iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/otsensor/fmt/json : msg : Object
{
  "topic": "iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/otsensor/fmt/json",
  "payload": {
    "d": {
      "name": "d38ddac7ba27",
      "temp": 17,
      "humidity": 79,
      "objectTemp": 23
    }
  },
  "deviceId": "d38ddac7ba27",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

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

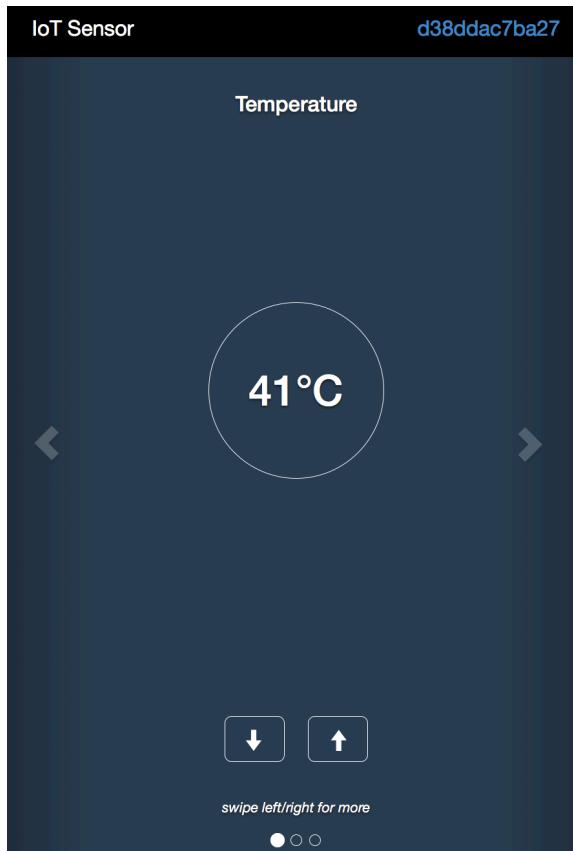
iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/otsensor/fmt/json : msg : Object
{
  "topic": "iot-2/type/iotqs-sensor/id/d38ddac7ba27/evt/otsensor/fmt/json",
  "payload": {
    "d": {
      "name": "d38ddac7ba27",
      "temp": 17,
      "humidity": 79,
      "objectTemp": 23
    }
  },
  "deviceId": "d38ddac7ba27",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

```

12. 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[ous]" and continues with the flow to the template labeled danger.



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



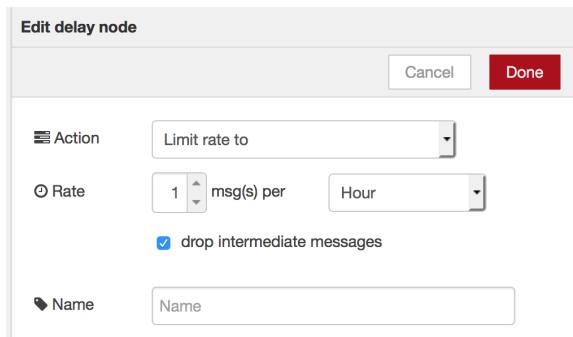
The message in the debug tab should change.

info	debug
	all flows current flow
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	

Connect to Twitter and Tweet High Temperature

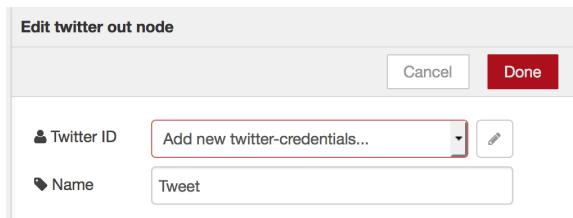
In this section, we will connect a Twitter account and use the Twitter account to tweet when the temperature from the temperature sensor is “dangerous”. This section is optional and may be skipped.

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](#)



Node RED

nodered.org

Node-RED Twitter node

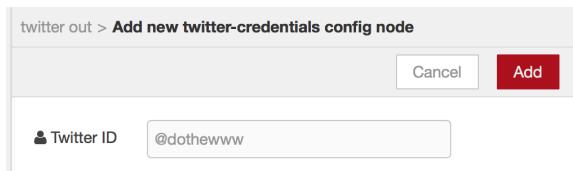
This application will be able to:

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

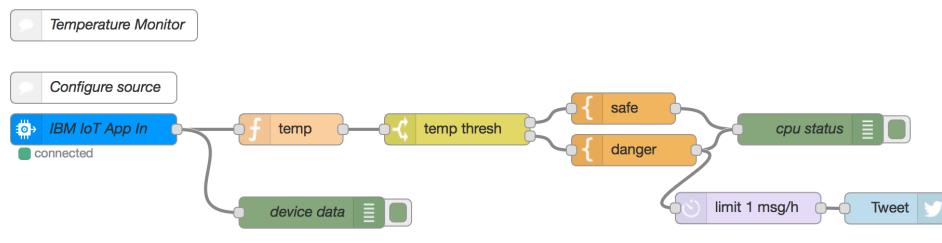
Will not be able to:

- See your Twitter password.

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.



Get the code:
ibm.biz/BdresH

5. Click on to save and deploy the changes.

6. Since the temperature is above 40°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.

Connect to Twilio and Text High Temperature

In this section, we will connect a Twilio phone number to the application and send a text message notification when the temperature is at least 40°C. This section is optional and may be skipped.

1. Sign up for a Twilio account at <http://twilio.com>. If you already have a Twilio account, sign in. Create a Twilio phone number that will be used in step #3.
2. In the Console Dashboard, click on the lock icon next to Auth Token. Copy the Account SID and Auth Token.

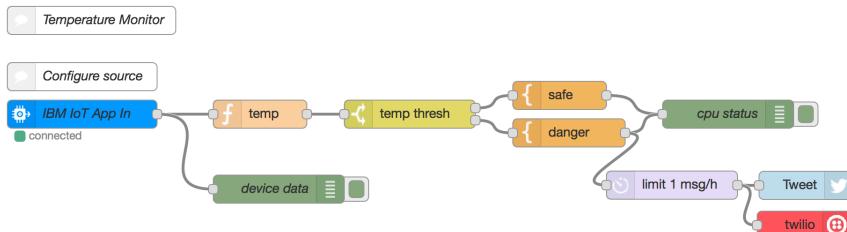
The screenshot shows the Twilio Console Beta dashboard. At the top, it displays the Account SID (AC12345ab67890cd12efg34567890hi1234) and Balance (+ \$6.632). Below this, there's a 'Recently Used Products' section with links to Programmable SMS, Phone Numbers, Programmable Voice, and Add-ons. The 'Programmable Voice' section is expanded, showing its description and a link to the Programmable Video beta.

3. Add a node. Click on the Pencil to provide your **Account SID** and **Auth Token** from step #2, and **From** phone number using the Twilio phone number from step #1. Fill in the **SMS to** textbox with a phone number that will be texted to.

The screenshot shows the Node-RED editor with a node titled 'twilio out > Edit twilio-api node'. The configuration fields are as follows:

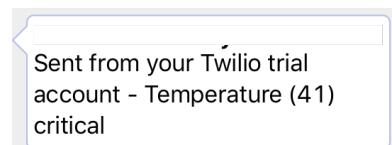
- Account SID: AC12345ab67890cd12efg34567890hi1234
- From: 15555555555
- Token: (redacted)
- Name: Name

4. Connect the nodes together as shown.



Get the code:
ibm.biz/Bdres

5. Click on to save and deploy the changes. You should receive a text message shortly.

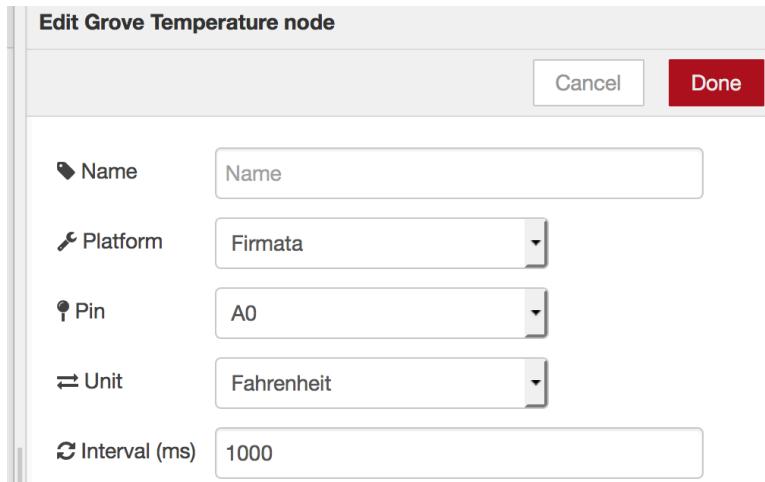


Create a Node-RED flow on Intel NUC

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 and, along with custom logic, can bring developing hardware solutions down to a level anyone can build quickly and easily.

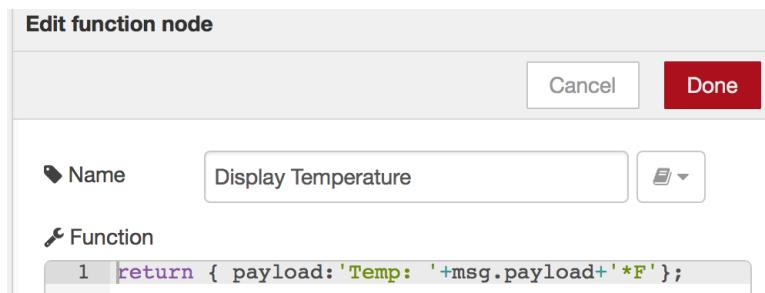
In this section, we will connect to the Node-RED application running on the Intel Edison board, capture the value of a real temperature sensor, control an LCD screen, control a LED light, and send the temperature to the Watson Internet of Things Platform.

1. 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 A0, and return Fahrenheit temperature value.

2. Add a  function node as shown below.



3. 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

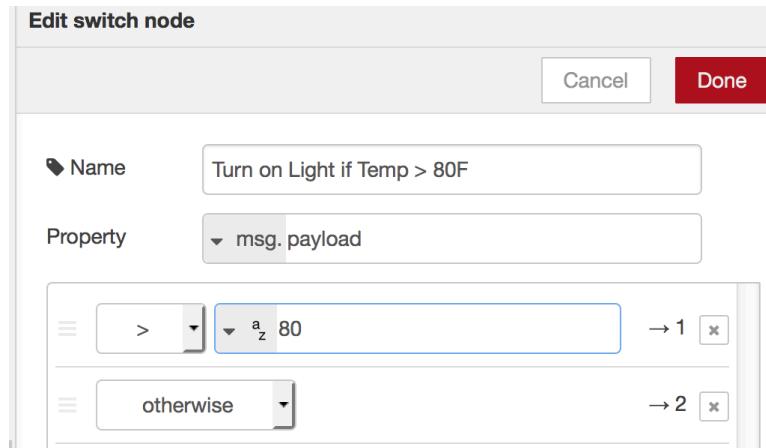
This will display the temperature value on the LCD connected to the I2C port.

4. Connect the nodes together as shown below.



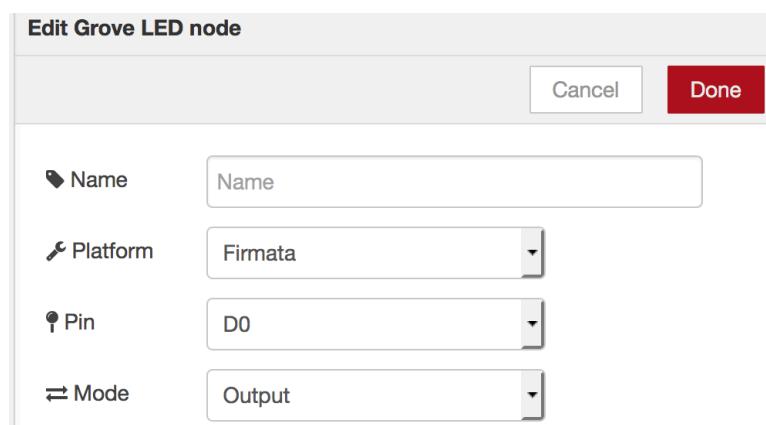
5. Connect a Grove LCD screen to the I2C port on the Grove shield connected to Arduino.
6. Click on the  Deploy button in the top right of the screen to save and deploy your changes. The LCD screen should display the temperature in Fahrenheit. The temperature value screen will update once every second when the flow is activated by the inject node.

7. 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.

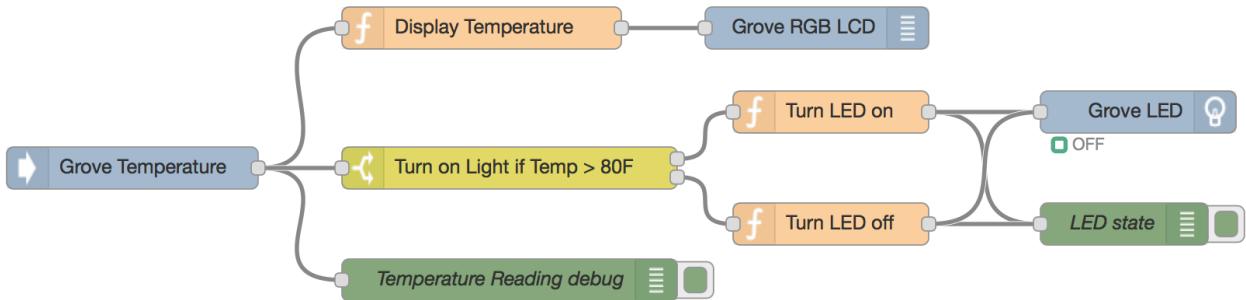


8. Add two  nodes, one to turn the LED on, and one to turn the LED off as shown below.

9. Add a  Grove LED node as shown below. This will control the LED connected to port D0.



10. Connect the nodes together as shown below.



This flow is available at <https://ibm.biz/BdrgCw>

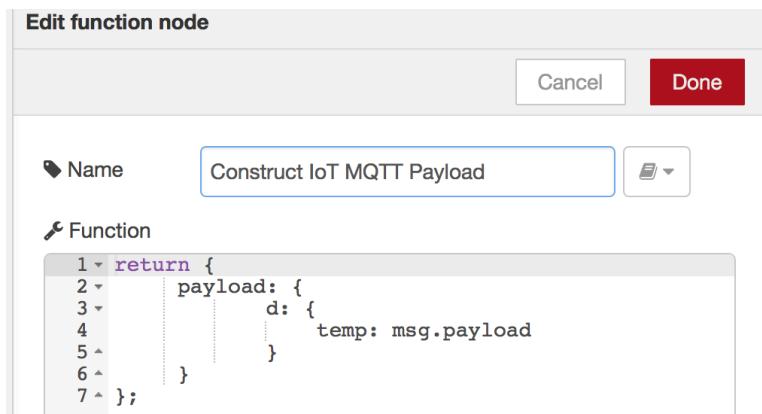
11. Connect a LED light to the D6 port on the Grove base shield connected to the Intel Edison.

12. 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 Bluemix can react to high temperatures.

1. Add a  node as shown below



2. This JavaScript function constructs and returns a JSON 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 JSON 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

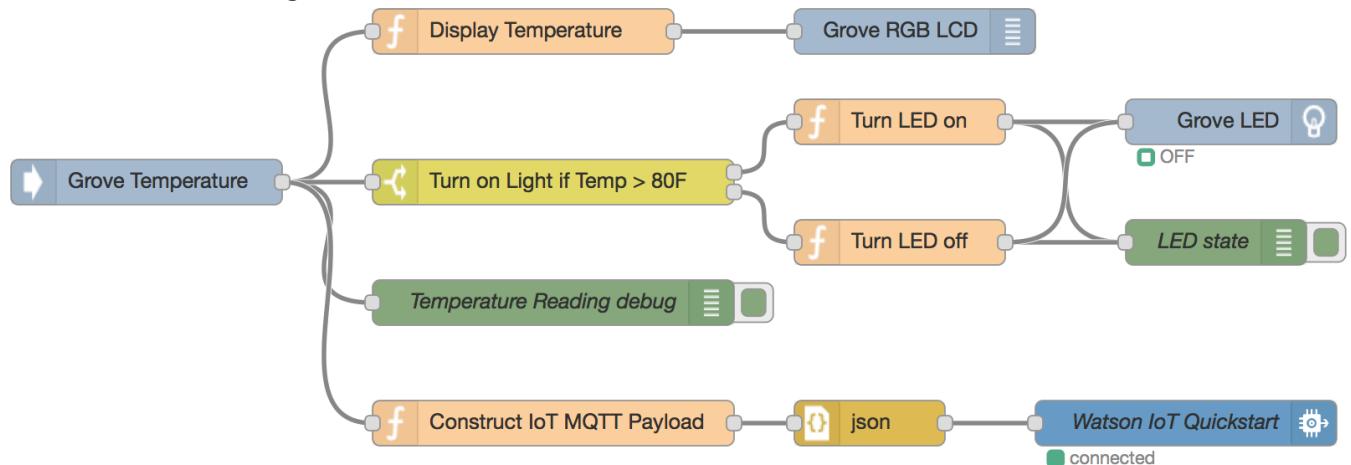
Event type event

Format json

Name Watson IoT Quickstart

Reuse Device ID
from simulated IoT
sensor

5. Connect the nodes together as shown below.



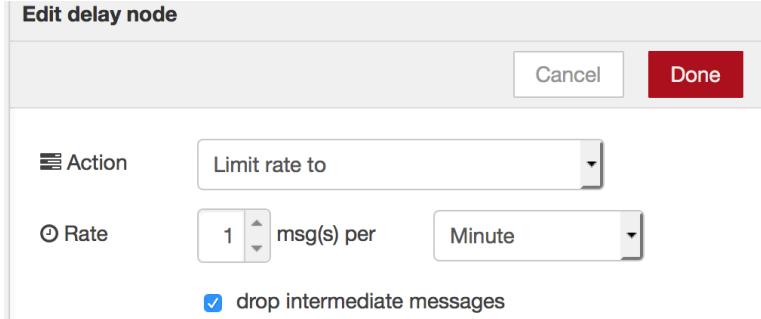
This flow is available at <https://ibm.biz/BdrgCw>

6. Click on Deploy to save and deploy your changes.
7. Return to the Node-RED application in IBM Bluemix. You should see the real temperature being reported.

Store Temperature Into 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. This section is optional and can be skipped. However, it is a prerequisite for the **Retrieve Temperatures From Cloudant NoSQL Database** section.

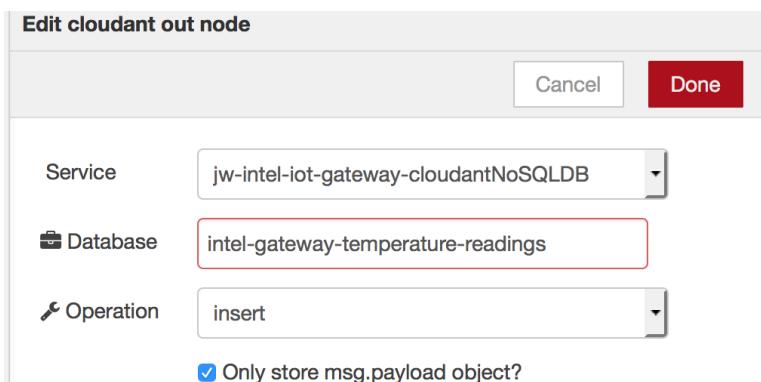
1. Add a  node as shown below. This node will limit this flow to execute once per minute.



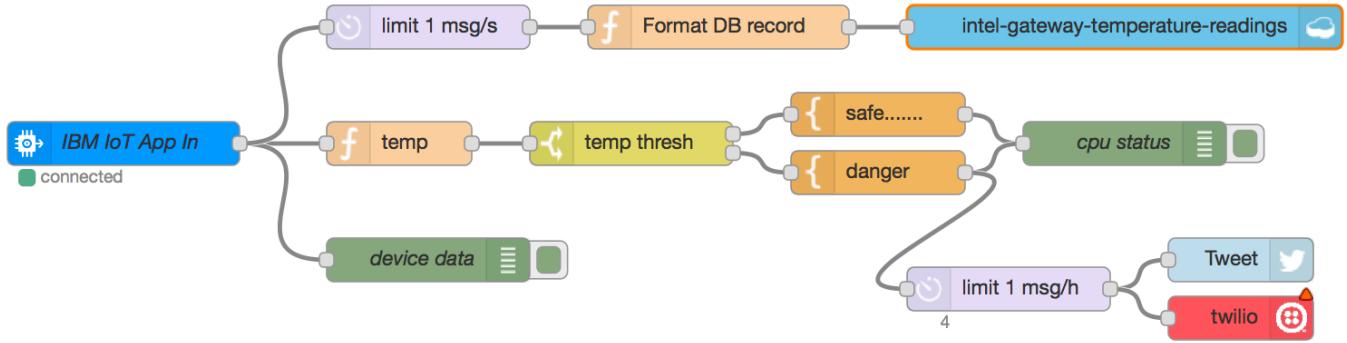
2. Add a  node as shown below.



3. Add a  node as shown below.

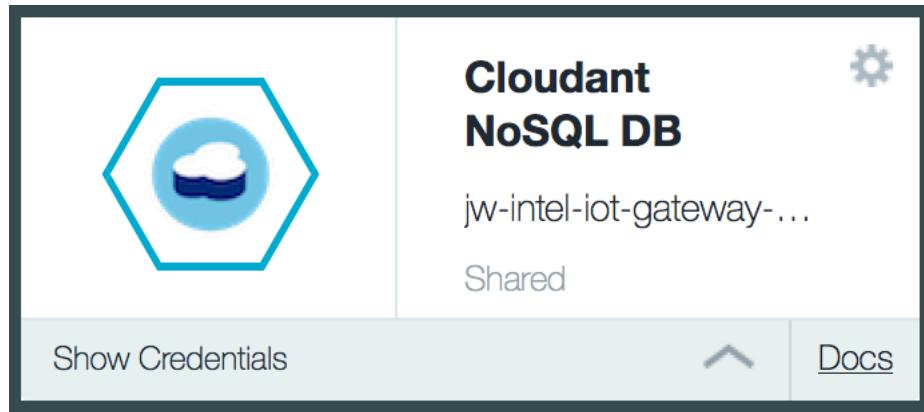


4. Connect the nodes together as shown below.



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

5. Click on Deploy to save and deploy your changes.
6. Go back to the IBM Bluemix 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. A list of databases is displayed. The database named **intel-gateway-temperature-readings** contains documents representing each temperature event that has been stored by the Node-RED application. Click on the database named **intel-gateway-temperature-readings**.

Name	Size	# of Docs	Update Seq	Actions
intel-gateway-temperature-readings	2.9 KB	57	5	
nodered	38.7 KB	4	1	

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

The screenshot shows the IBM Cloudant interface. On the left, there's a sidebar with various navigation links: Databases, Replication, Integrations, Active Tasks, Account, Support, Documentation, and IBM Cloudant. Below the sidebar is a 'Log Out' link. The main area is titled 'intel-gateway-temp...' and contains sections for All Documents, Query, Permissions, Changes, and Design Documents. A modal window titled 'Query Options' is open, showing settings for a query. The 'Include Docs' checkbox is checked. Under 'Keys', there are two tabs: 'By Key(s)' and 'Between Keys'. Below that is a section for 'Additional Parameters' with a 'Limit' dropdown set to 'None', a 'Descending' radio button, and a 'Skip' field with '# of rows'. At the bottom of the modal are 'Run Query' and 'Cancel' buttons. The main interface below the modal shows a list of documents with their keys and some truncated JSON data.

10. Each box represents one document (in our case one temperature event) that contains the payload (time and temperature) we stored earlier.

The screenshot shows the IBM Cloudant interface with a table view. The top bar includes 'JSON' and 'Table' buttons, a 'Docs' checkbox, 'Options' and 'API' buttons, and a bell icon. The table has columns for 'Metadata', 'temp', and 'time'. There are four rows of data:

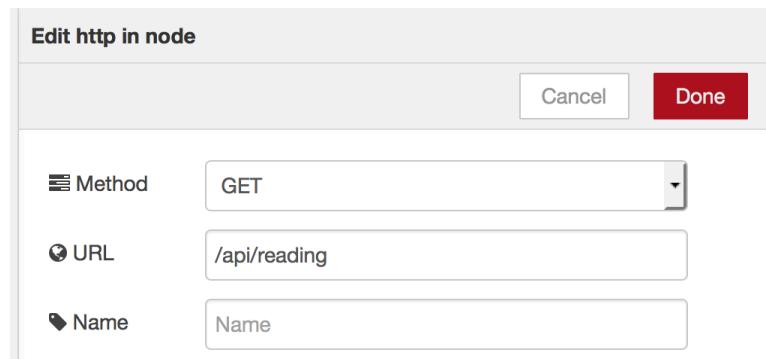
Metadata	temp	time
37da0b7e91e3517... 1-30e5dbcb0681406...	126	1471917603243
37da0b7e91e3517... 1-a7565eb1a52335...	120	1471917613311
37da0b7e91e3517... 1-35405b8de2c9f28...	126	1471917648537
37da0b7e91e3517... 1-63e0435e3f29be2...	126	1471917653569

Retrieve Temperatures From Cloudant NoSQL DB

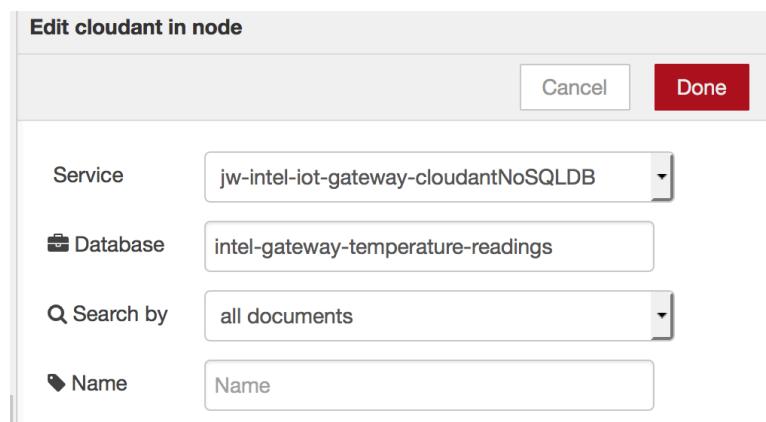
This section shows how to retrieve temperature data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time. This section is optional and can be skipped. Completion of the section titled **Store Temperature Into Cloudant NoSQL Database** is required before beginning this section.

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

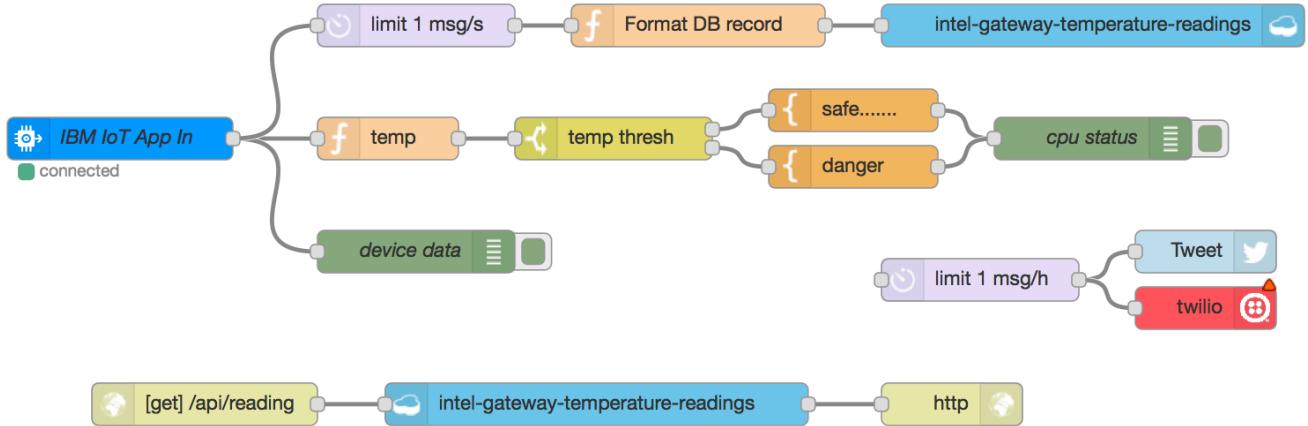
1. Add a  node as shown below.



2. Add a  node as show below.



3. Finally, add a  node. Connect the nodes together as shown below. Download this flow: <https://ibm.biz/BdrgQS>



4. Click on Deploy 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": "0bf60630164bd19c0bc74bef6e037da6", "_rev": "1-cad3644fdbbe75376b9998c6be5280fe2", "time": 1471922146114, "temp": 129}, {"_id": "0bf60630164bd19c0bc74bef6e070b6a", "_rev": "1-ae2aa69dd2fb9d4ab8ae48582f206a66", "time": 1471922161179, "temp": 129}, {"_id": "0bf60630164bd19c0bc74bef6e0d72df", "_rev": "1-c909094db72089d0da08f9eb97e2fe87", "time": 1471922186297, "temp": 129}, {"_id": "0bf60630164bd19c0bc74bef6e11d7c9", "_rev": "1-4da325037071066aae783cf27f99920a", "time": 1471922201380, "temp": 129}, {"_id": "0bf60630164bd19c0bc74bef6e12e917", "_rev": "1-f1b4e045bbb1d895beb146c50fba000d", "time": 1471922206405, "temp": 124}, {"_id": "0bf60630164bd19c0bc74bef6e15a4dc", "_rev": "1-532d6200c5c4c5be3fc3d0a6b1367ec", "time": 1471922216456, "temp": 129},
```

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.

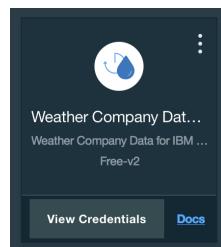
Send Alerts from Watson IoT to the Intel Gateway

In this section, we'll extend the temperature sensor LCD to include the outside temperature. The outside temperature will be retrieved via the Weather Company Data for IBM Bluemix service.

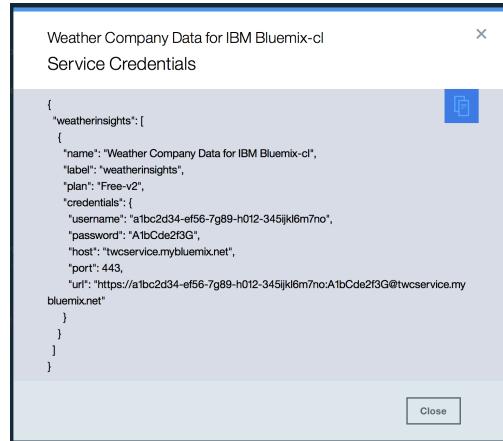
1. Click on the **+ Add a Service or API** card in the application overview for your Node-RED application in IBM Bluemix.
2. Select the **Weather Company Data for IBM Bluemix** under the Data & Analytics category.

The screenshot shows the IBM Bluemix interface. At the top, it says "IBM Bluemix Ready? Try the new Bluemix | New! Try OpenWhisk" and "DASHBOARD". Below that, the "ORG: walicki@us.ibm...." dropdown is shown, along with a search bar containing "weather". On the left, there's a sidebar titled "Services" with a minus sign icon. It lists several categories: Watson, Mobile, DevOps, Web and Application, Network, Integration, and Data and Analytics. The "Data and Analytics" option has a checked checkbox. The main content area is titled "Services // The building blocks of any great app". It features a section for "Data and Analytics" with a sub-section titled "Essential data services; limitless possibilities". A large green button labeled "HELP ME PICK" is visible. To the right, there's a hexagonal icon with a blue water drop and a circular arrow, representing the Weather Company Data service. Below the icon, the text "Weather Company Data for IBM Bluemix" and the IBM logo are displayed.

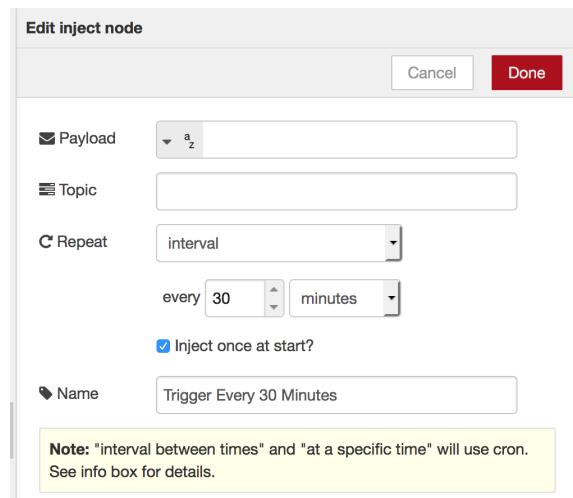
3. Click **Create** to add the service to your application.
4. When prompted to restage the application, click **Restage** to restart the application and update the environment with the credentials to the Weather service.
5. When the application has restarted, a third service tile will appear in the Connections tab. Click on **View Credentials** in the Weather Company Data service tile to show the service credentials.



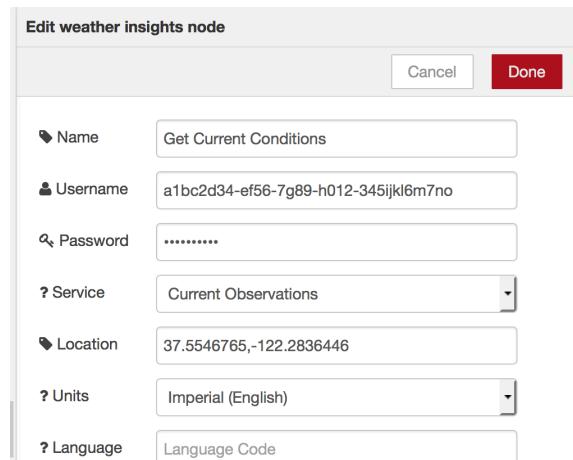
6. Copy the username and password values and save them for use in step #8.



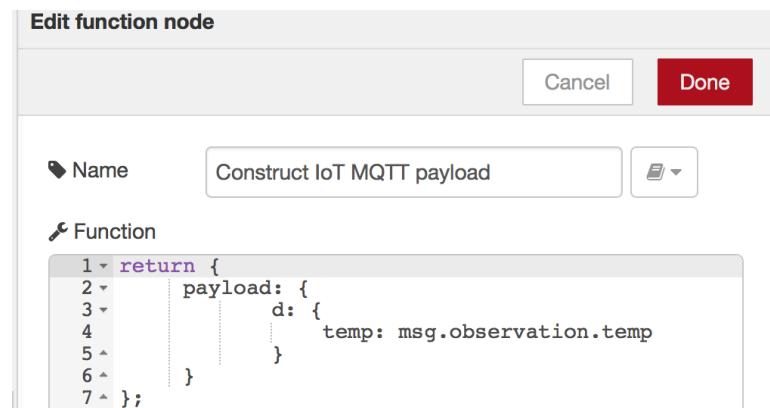
7. Add a node as shown below. This will trigger the flow once at startup, and then every 30 minutes.



8. Add a node as shown below. Use the Weather Company Data API credentials from step #6.

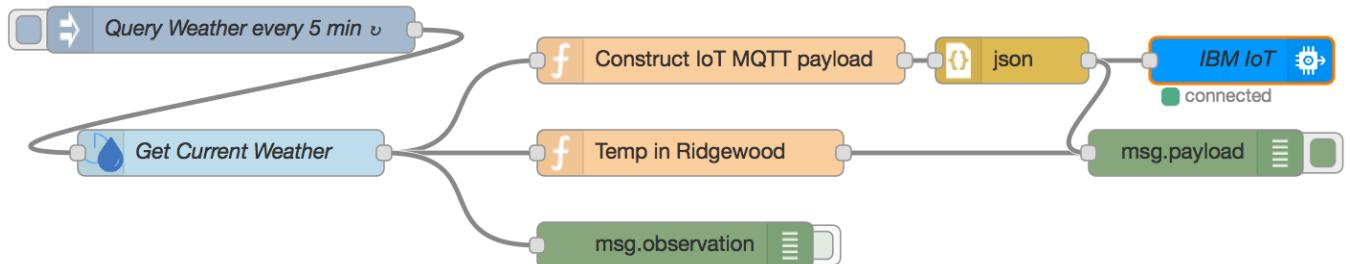


9. Add a  node as shown below.



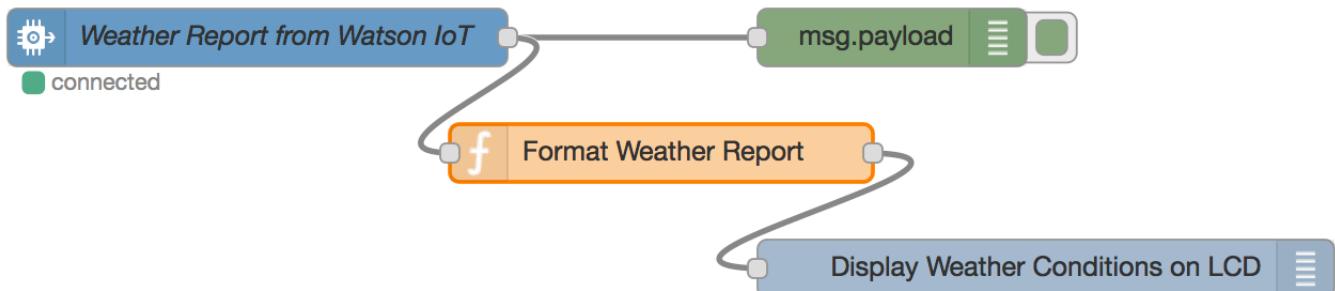
This will format the temperature in a MQTT message payload, for use in step #11. If the weather condition changes every 5 minutes, the value will be sent to the Intel NUC for display on the LCD.

10. On IBM Bluemix, Connect the nodes together as shown below.



This flow is available at <https://ibm.biz/BdrgQL>

11. On the Intel NUC, Connect the nodes together as shown.



This flow is available at <https://ibm.biz/BdrgQg>

12. Click on  to save and deploy your changes. The LCD should display the outside temperature and will update the temperature from the Weather service every 5 minutes.