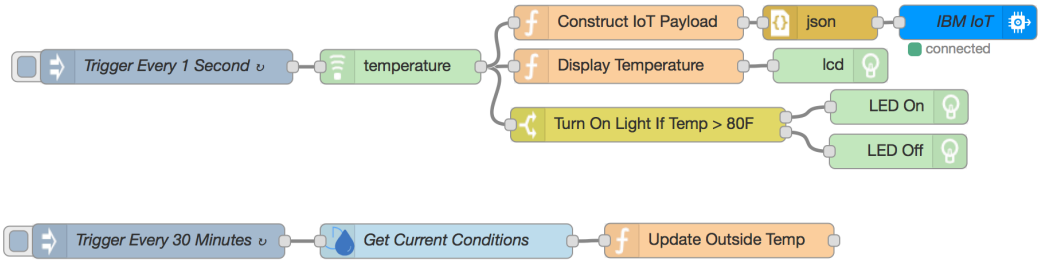
Tracking Temperature with Intel Edison and Grove Sensors

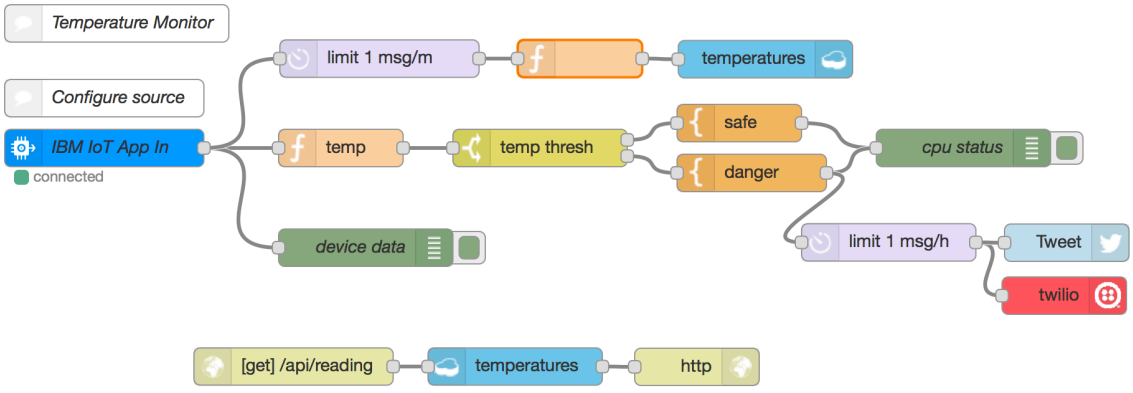
Hands-On Lab

Author: JeanCarl Bisson | jbisson@us.ibm.com | @dothewww



Node-RED application running on Intel Edison captures temperature from Grove temperature sensor (pg. 13), displays temperature on Grove LCD (pg. 14), triggers Grove LED light (pg. 15) on high temperature, retrieves outside weather condition from the Weather Company Data API (pg. 17), and sends temperature data to Watson IoT Platform (pg. 20).

Node-RED application running on IBM Bluemix responds to incoming high temperature (pg. 3) by tweeting (pg. 9) and texting (pg. 11) alerts. Temperature is also stored for historical analysis in a Cloudant NoSQL database (pg 22 & 24).

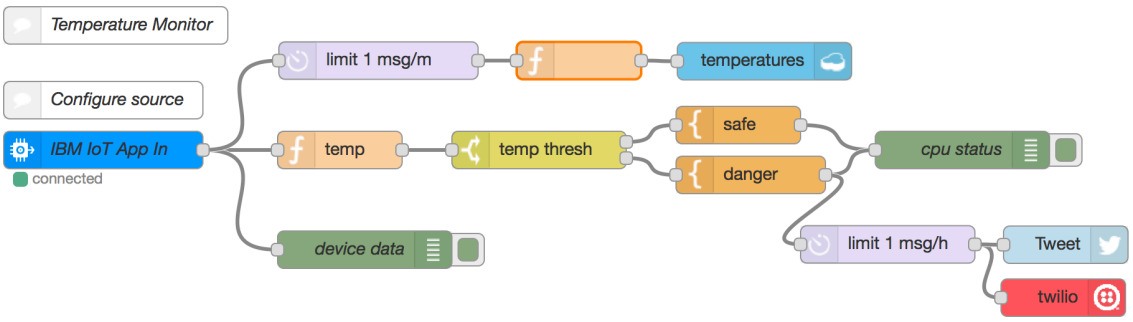


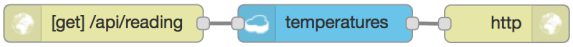
A digital copy of this lab and completed flows can be found at:
`http://ibm.biz/lab-temperature-sensor-with-intel-edison`

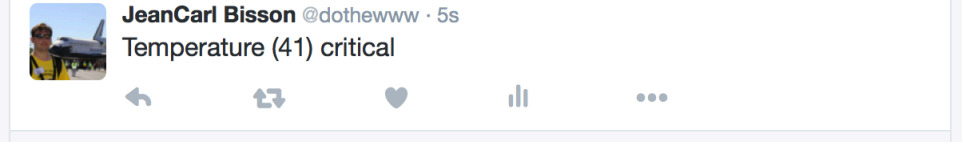
Table of Contents

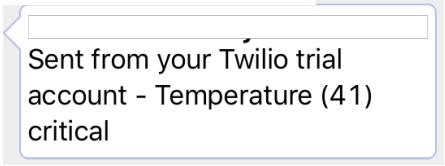
Topics covered: Intel Edison, Grove sensors, Node-RED, Watson Internet of Things Platform, Cloudant NoSQL Database, Twitter, Twilio, and Weather Company Data for IBM Bluemix.

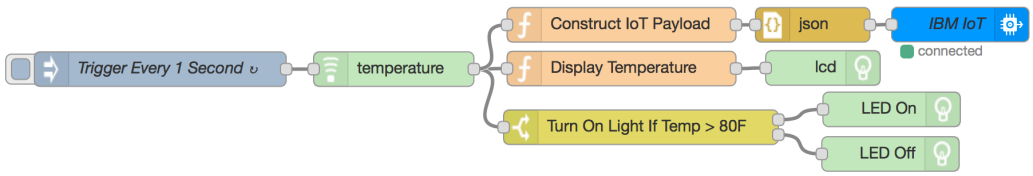
In this lab, we will use Node-RED on an Intel Edison board (with a Grove Starter kit) along with several services available in IBM Bluemix to monitor the temperature and alert maintenance of a high temperature. Using Node-RED, running on the Intel Edison board, the application will read the temperature 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 Intel Edison. 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.







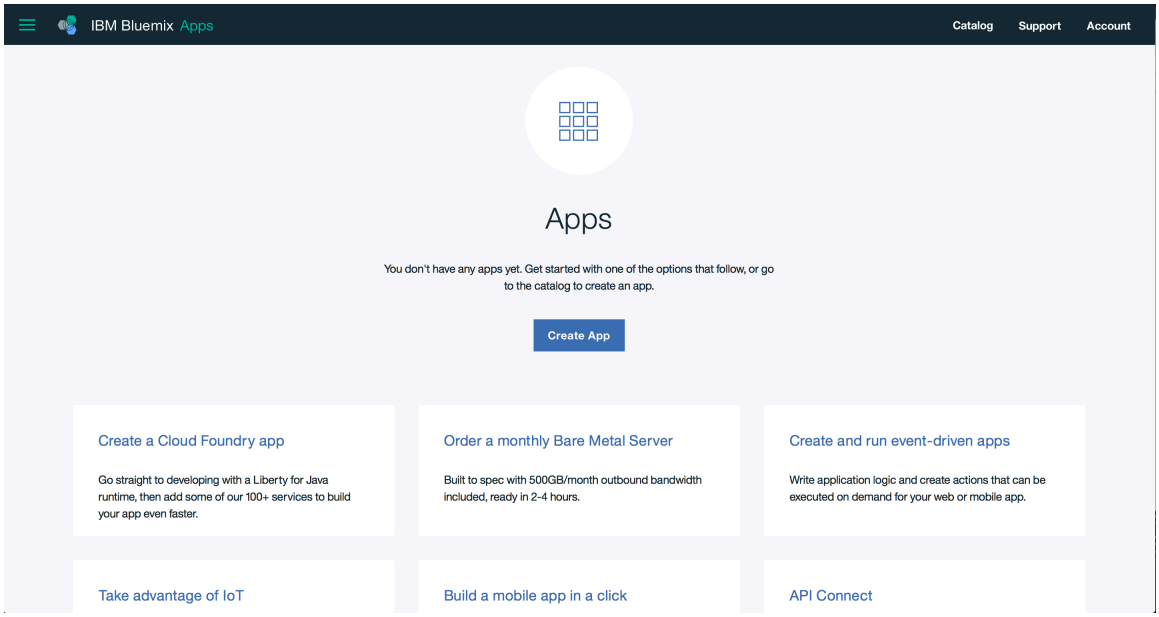




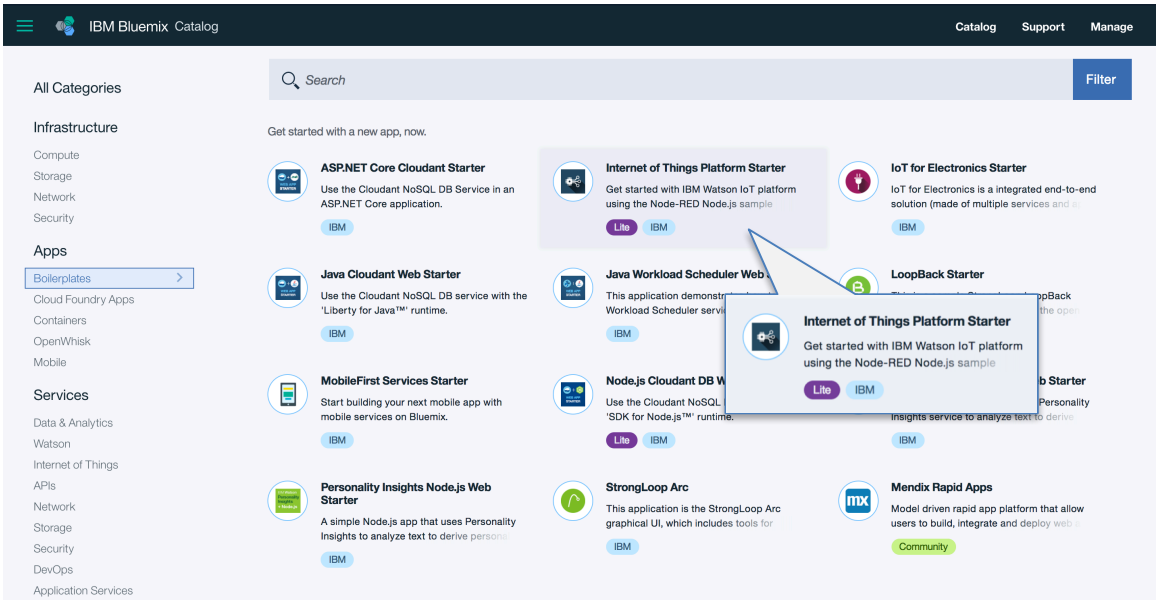
Creating an IoT application in IBM Bluemix

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.

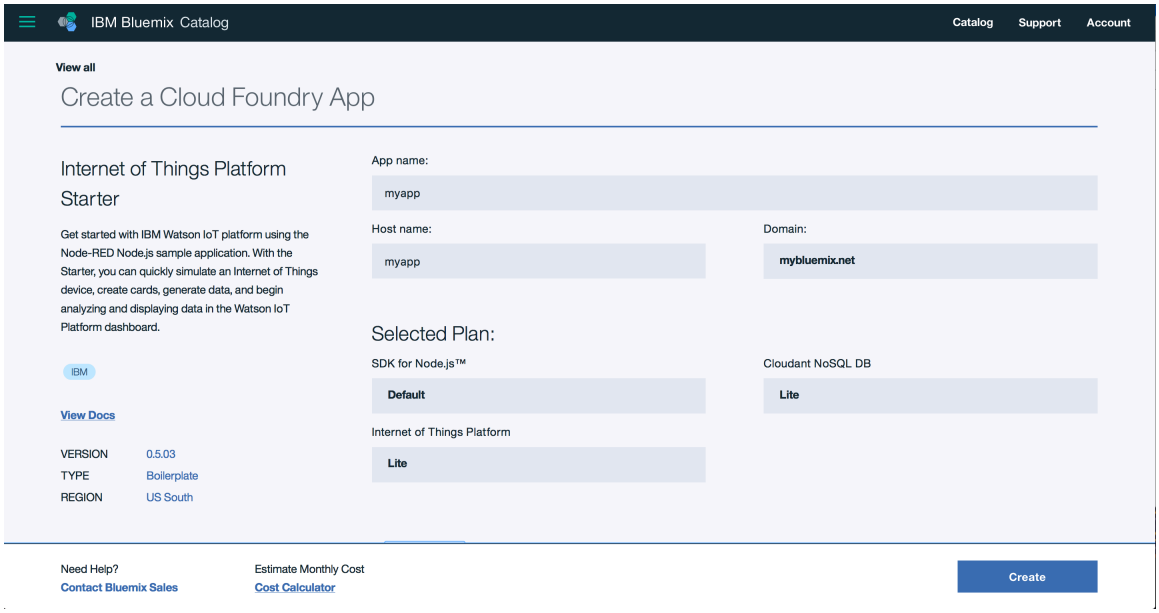
1. Open a web browser and go to your IBM Bluemix dashboard, `http://bluemix.net`. This is the IBM Bluemix dashboard where you have access to creating Cloud Foundry apps, IBM Containers, Virtual Machines, and a variety of Services. We are going to create a Cloud Foundry application today. To access the catalog, click on the **Catalog** link in the top right corner.



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, Internet of Things Platform service, and the SDK for Node.js. Under Boilerplates, select the **Internet of Things Platform Starter** boilerplate tile.

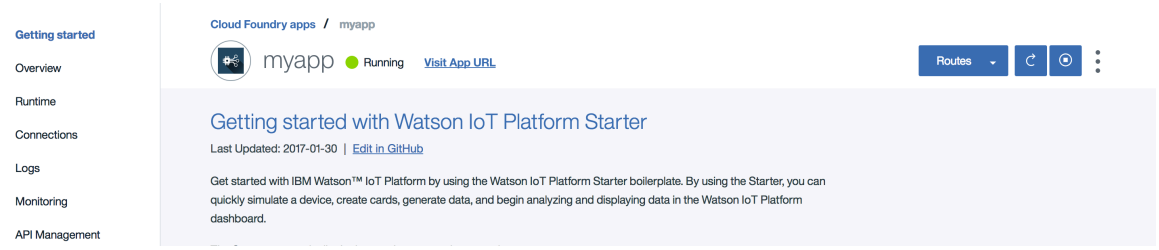


3. Pick an unique name for your application. If you choose ***myapp***, your application will be located at `http://`***`myapp`***`.mybluemix.net`. There can only be one "***myapp***" application registered in IBM Bluemix. You can try adding your initials in front of the host if the host you choose is already taken by someone else. Click on **Create** to create the application instance.

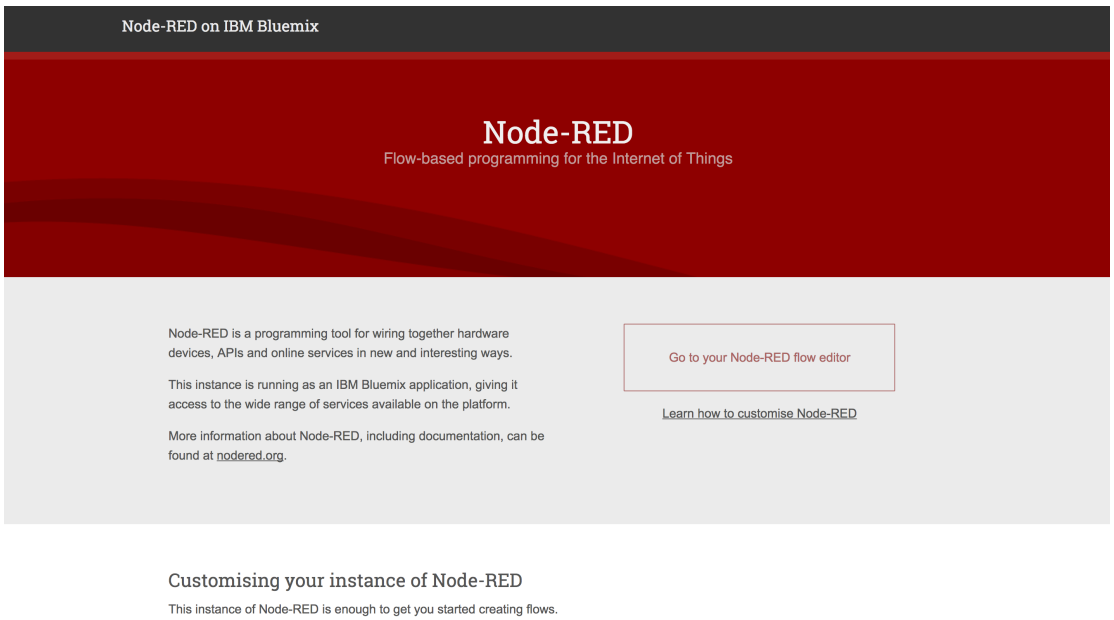


4. IBM Bluemix will create an application in your account based on the services in the boilerplate. This is called staging an application. It can take a few minutes for this process to complete. While you wait, you can click on the **Logs** tab and see activity logs from the platform and Node.js runtime.

5. When the application is running, click on the **Visit App URL** link to open the application in a new browser tab.

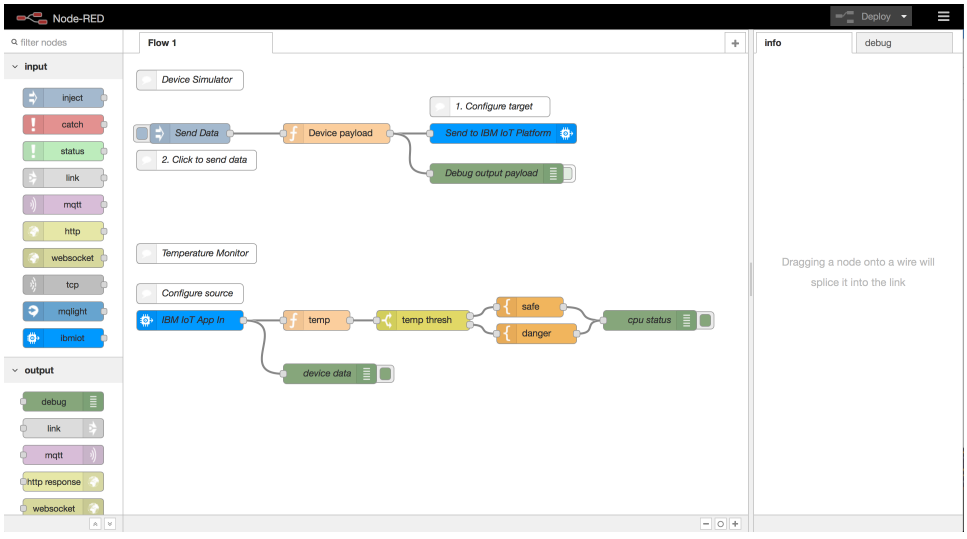


6. This is Node-RED start page. Node-RED is an open-sourced Node.js application that provides a visual editor that makes it easy to wire together flows. Click on the red button **Go to your Node-RED flow editor** to launch the editor.

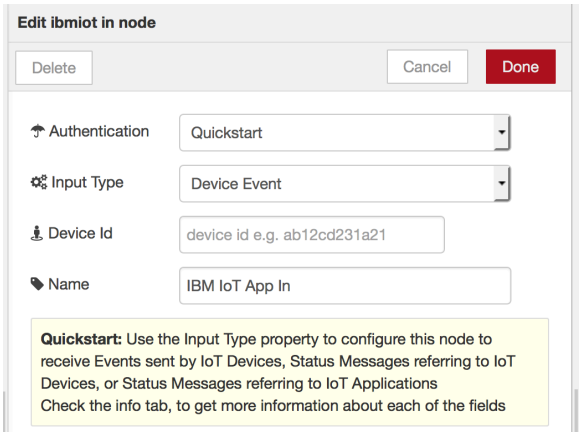


7. On the left side is a palette of nodes. Each node performs a defined function, configurable by dialog windows or by modifying the msg input. You can click on a node in the palette and find out what it does in the info tab on the right side of the screen. Drag and drop nodes and connect them together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

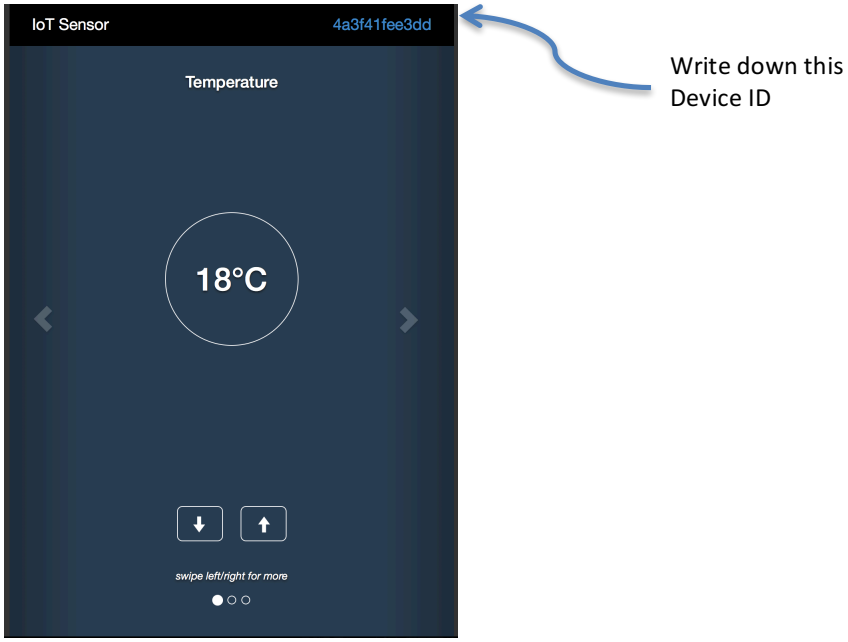
   Let's begin by configuring the IBM IoT App In node. Double click on the IBM IoT App In node.



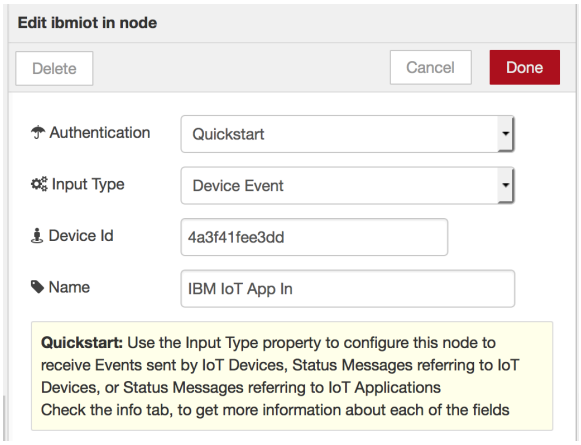
   Nodes can be customized in different ways depending on what they do. The IBM IoT node begins a flow when a message is received from a specific device via the Watson IoT Platform. We need a Device Id of a device that is connected to the IoT Platform. We could use a real device, or we can simulate devices.



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



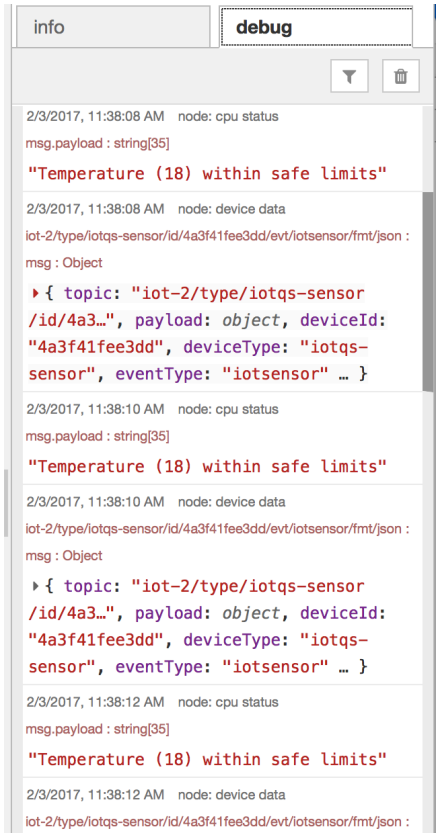
9. Copy this alphanumeric device ID into the **Device ID** field in the Node-RED application as shown below. Click **Done**.



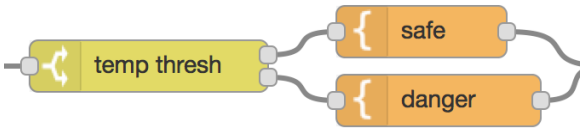
10. Click on the 

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

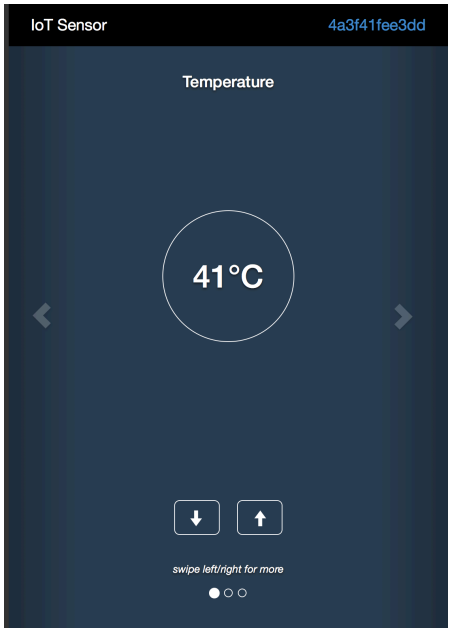
11. Click on the debug tab in the right-hand pane. Every second, the simulator emits a device event to the Watson IoT platform with temperature and other sensor data. The Node-RED application subscribes to these events and the IBM IoT in node triggers the flow with the sensor data in the message. When 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. You can see the sensor data as a JSON object in the debug pane.

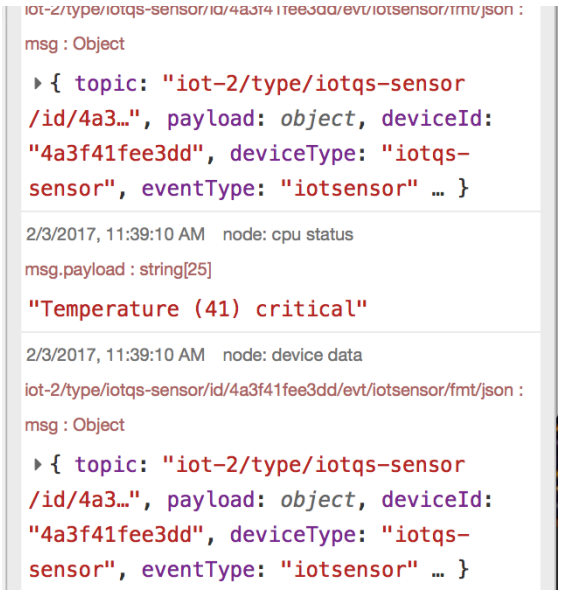


12. The yellow node below is called a **switch** node. You can program logic using a switch node and split a flow into one or more flows based on a property's value. In this example, if the temperature is less than or equal to 40°C, it is considered "safe" and continues with the flow to the template 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.



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.


IoT Sensor
4a3f41fee3dd
Temperature
41°C
swipe left/right for more


iot-2/type/iotqs-sensor/id/4a3f41fee3dd/evt/iotsensor/fmt/json :
msg : Object
{ topic: "iot-2/type/iotqs-sensor /id/4a3…", payload: object, deviceId: "4a3f41fee3dd", deviceType: "iotqs-sensor", eventType: "iotsensor" … }
2/3/2017, 11:39:10 AM node: cpu status
msg.payload : string[25]
"Temperature (41) critical"
2/3/2017, 11:39:10 AM node: device data
iot-2/type/iotqs-sensor/id/4a3f41fee3dd/evt/iotsensor/fmt/json :
msg : Object
{ topic: "iot-2/type/iotqs-sensor /id/4a3…", payload: object, deviceId: "4a3f41fee3dd", deviceType: "iotqs-sensor", eventType: "iotsensor" … }

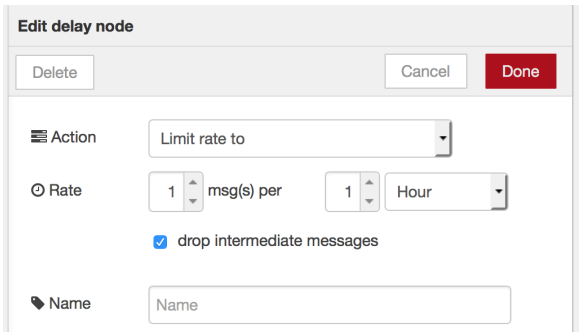
Connect to Twitter and Tweet High Temperature

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

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

2. Add a 

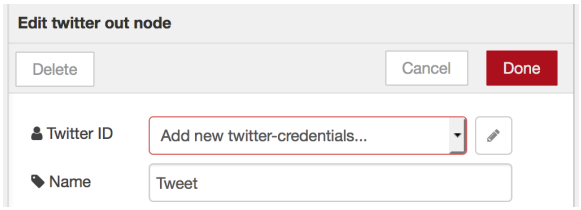
node as shown below.

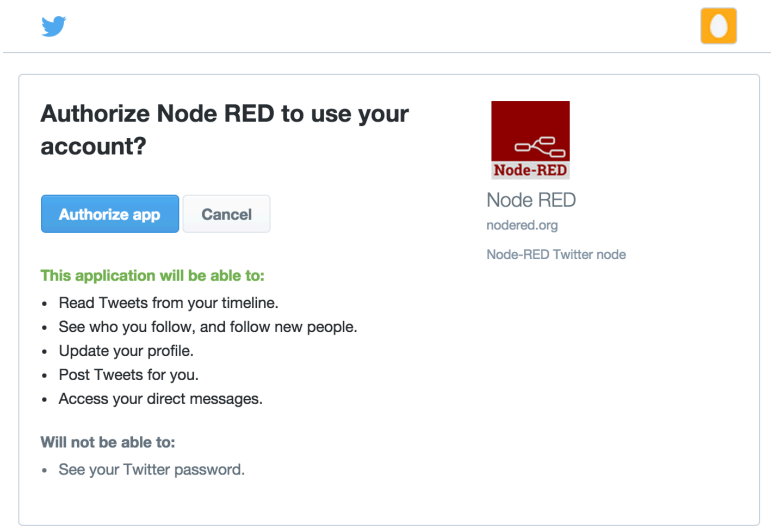


   The delay node limits how often the flow is run. Since the temperature is dangerous every second, without this node, a tweet would be sent every second. With this node limiting messages to once an hour, the Twitter node will send a tweet once an hour, dropping any additional messages during the hour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

3. Add a 

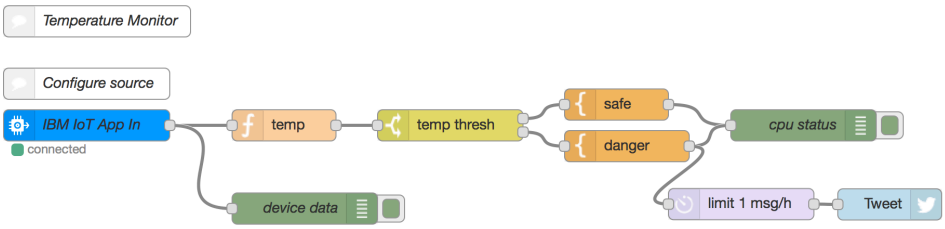
node. Click on the pencil button and authenticate with Twitter. The account you sign in with will be used to send tweets.





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

4. Connect the nodes as shown below.



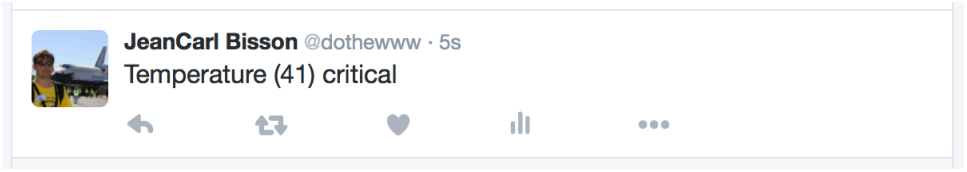
Get the code:
`ibm.biz/BdsCN2`

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]” template, 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.

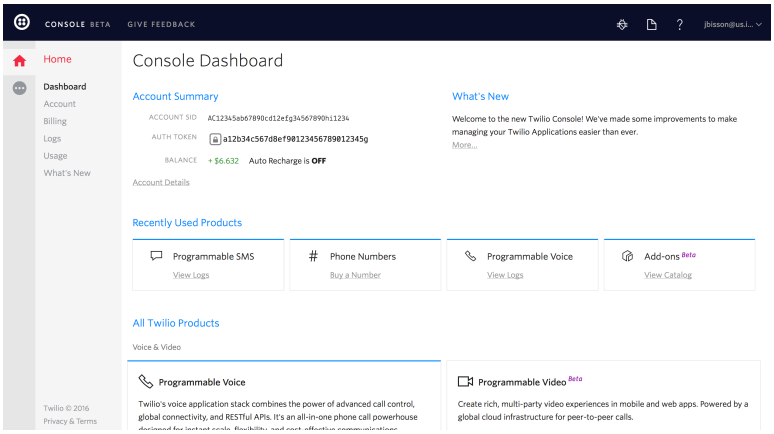


Connect to Twilio and Text High Temperature

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.

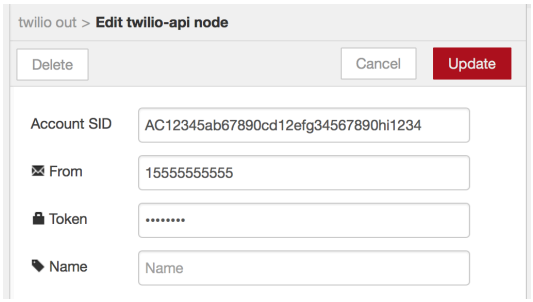
1. Sign up for a Twilio account at `http://twilio.com`. If you already have a Twilio account, sign in. Create a Twilio phone number that will be used in step #3.

2. In the Console Dashboard, click on the lock icon next to Auth Token. Copy the **Account SID** and **Auth Token**.

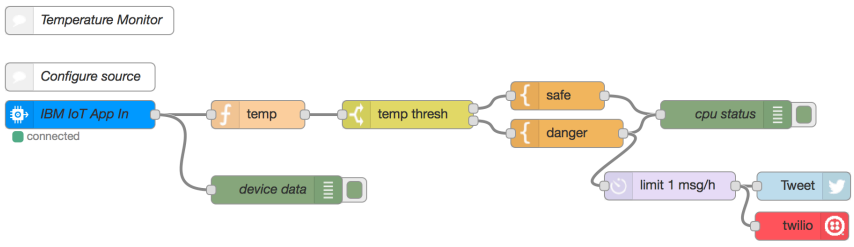


3. Add a 

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.



4. Connect the nodes together as shown.



Get the code:
`ibm.biz/BdsCNP`

5. Click on 

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

Install Node-RED on Intel Edison

In the first section we created Node-RED application in the Cloud. Node-RED can also be installed wherever Node.js can be run. In this section, we'll install Node-RED on an Intel Edison board. Access the command line on the Intel Edison by SSHing or connecting via USB.

1. To install Node-RED, run the following command in the command line on the Intel Edison:

   ```
   npm install -g node-red
   ```

2. Start Node-RED with the following command in the command line on the Intel Edison:

   ```
   node-red
   ```

3. Open a web browser and visit `http://<IP address of Edison>:1880`

4. Since Node-RED is an open-source project, you can also use community-contributed nodes to expand the capabilities of flows you compose. Install additional nodes (`nodered.org`) by installing npm packages from the command line.

   ```
   cd ~/.node-red
   npm install <package name>
   ```

   Replace `<package name>` with a Node-RED package name shown in the listing below. Install the packages listed below with an asterisk for use in the remainder of this lab.

5. Restart Node-RED by following steps #2 and #3.

NPM Package	**Nodes**
node-red-bluemix-nodes*	MongoDB, Twilio, and Weather nodes.
node-red-node-watson	Watson nodes: Conversation, Dialog, Document Conversion, Feature Extract, Image Analysis, Language Identification, Language Translation, Natural Language Classifier, News, Personality Insights, Relationship Extraction, Retrieve and Rank, Speech To Text, Text To Speech, Tone Analyzer, Tradeoff Analytics, Visual Recognition
node-red-contrib-scx-ibmiotapp*	Connect to both quickstart and registered versions of IBM Watson Internet of Things Platform.
node-red-contrib-grove-edison*	Work with Grove sensors connected to an Intel Edison board.
node-red-nodes-cf-sqldb-dashdb	Work with a database in a SQLDB or dashDB service that is integrated with IBM Bluemix.
node-red-node-cf-cloudant	Access a Cloudant database to insert, update, delete and search for documents.

* NPM packages required and used in the remainder of this lab

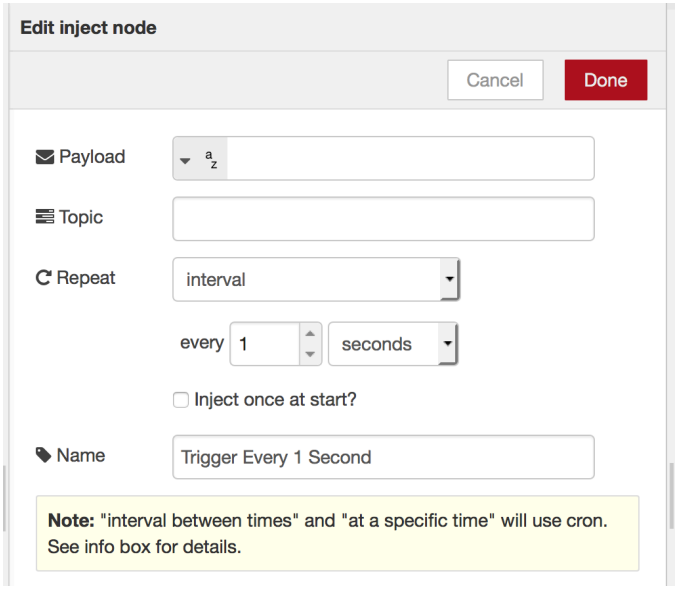
Create a Node-RED flow on Intel Edison

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 Edison board has made prototyping IoT devices easy. The Grove Starter Kit comes with a variety of sensors that can be connected to the Intel Edison 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 

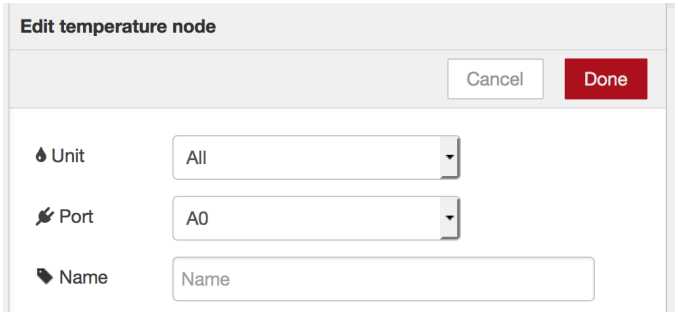
node onto the canvas. Double click on the inject node and customize as shown below.



The inject node will trigger the flow once every second.

2. Add a 

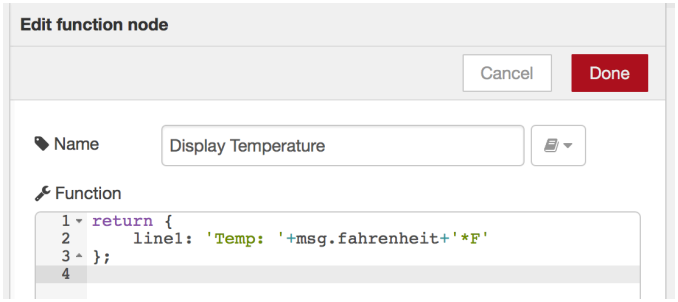
node to the right of the inject node as shown below.

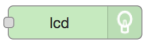


This will configure the temperature node to listen on Port A0, and return both Celsius and Fahrenheit temperature values.

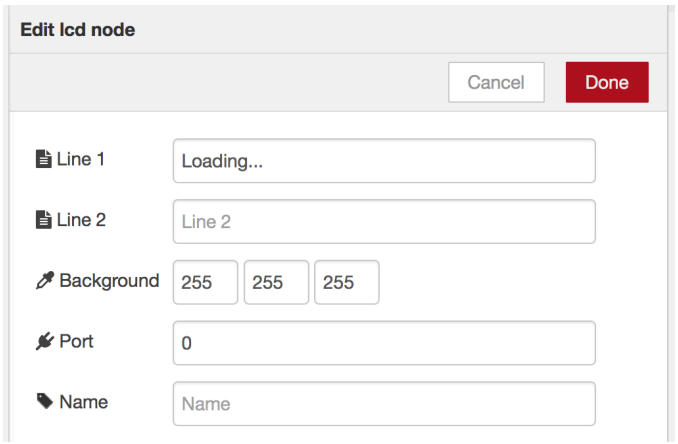
3. Add a 

node as shown below.



4. Add a 

node as shown below.

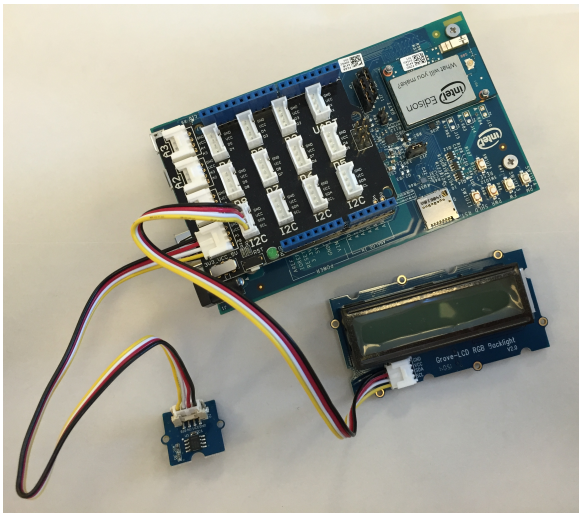


This will display the temperature value on the LCD connected to the I2C port.

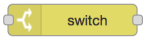
5. Connect the nodes together as shown below.

Get the code:
`ibm.biz/Bdresz`

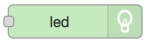
6. Connect a Grove Temperature sensor to the A0 port and a Grove LCD screen to the I2C port on the base shield.



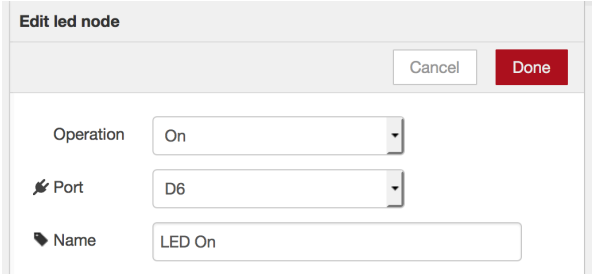
7. Click on the 

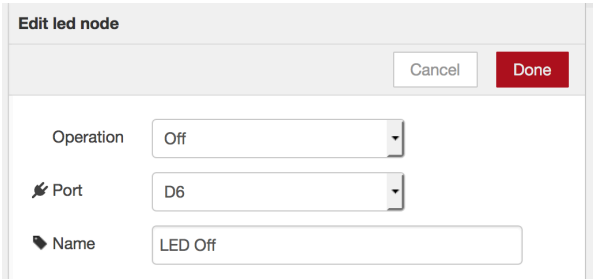
button in the top right of the screen to save and deploy your changes. The LCD screen should display the temperature in Fahrenheit. The temperature value screen will update once every second when the flow is activated by the inject node.
8. Next, we will activate a LED light when the temperature exceeds 80°F. Add a 

node and connect it to the temperature node as shown below.

9. Add two 

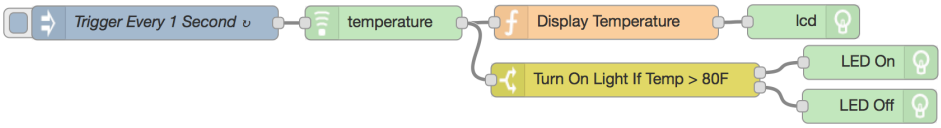
nodes, one to turn the LED on, and one to turn the LED off as shown below.





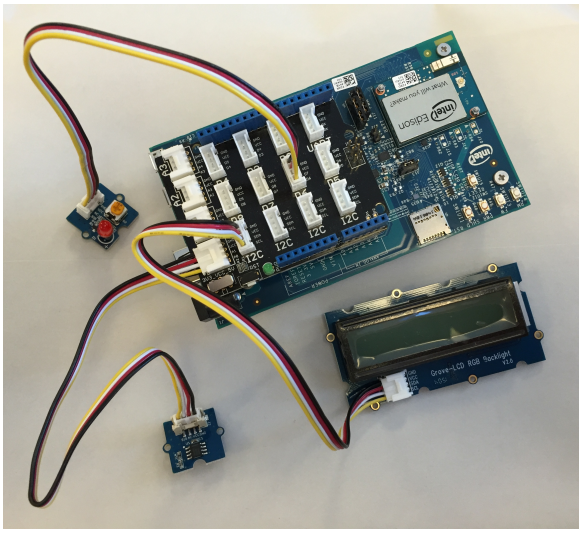
These will control an LED light connected to port D6.

10. Connect the nodes together as shown below.



Get the code:
`ibm.biz/Bdresv`

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



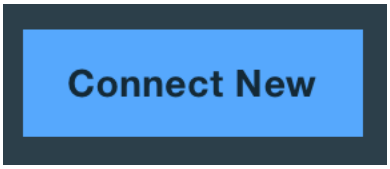
12. Click on the 

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

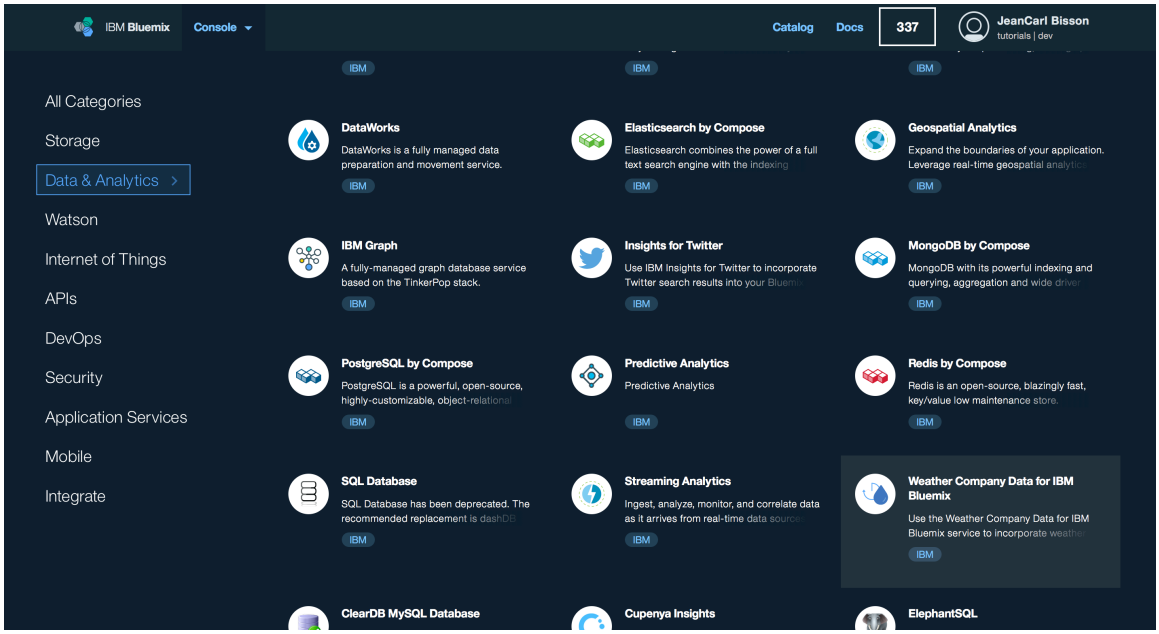
Add Weather in Node-RED

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. In the **Connections** tab in the application overview for your Node-RED application in IBM Bluemix, click on **Connect New**.



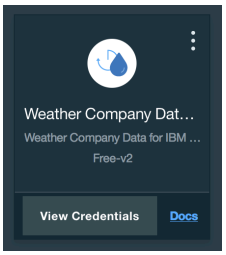
2. Select the **Weather Company Data for IBM Bluemix** under the Data & Analytics category.



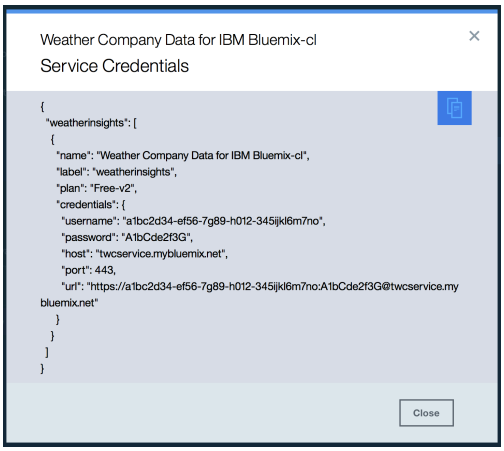
3. Click **Create** to add the service to your application.

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.

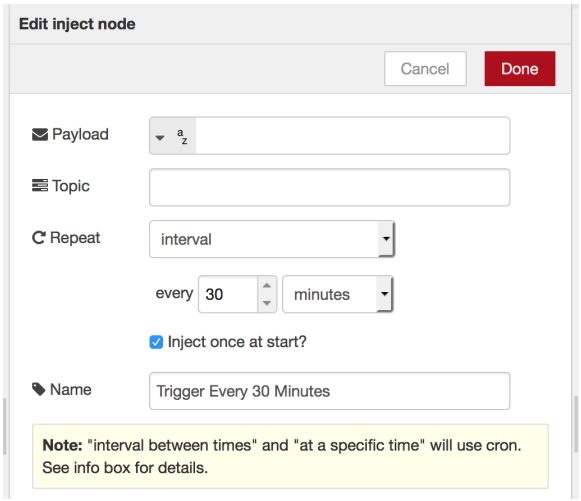


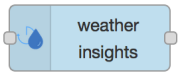
6. Copy the username and password values and save them for use in step #8.



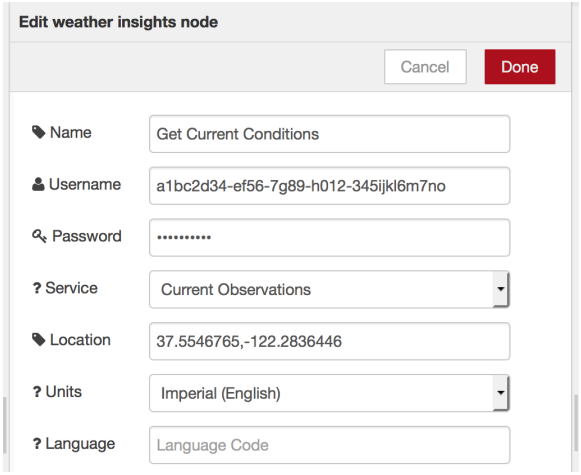
7. Add a 

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



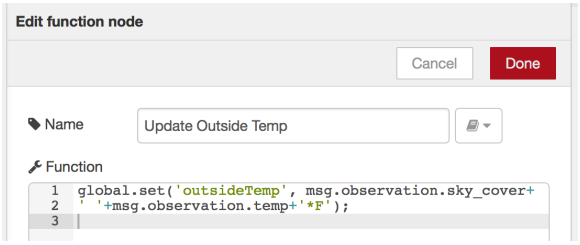
8. Add a 

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



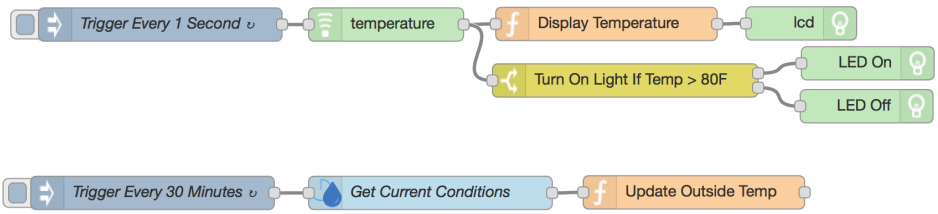
9. Add a 

node as shown below.



This will concatenate the sky cover and temperature and store it in a global variable, for use in step #11. If the weather condition changes every half hour, the value in the outsideTemp global variable will be updated.

10. Connect the nodes together as shown below.



Get the code:
ibm.biz/BdresG

11. Modify the Display Temperature function node, adding the line2 property of the message to read the global variable created in step #9. This will add a second line to the LCD, reading the value saved from the last API call to the Weather Company Data service.

12. Click on 

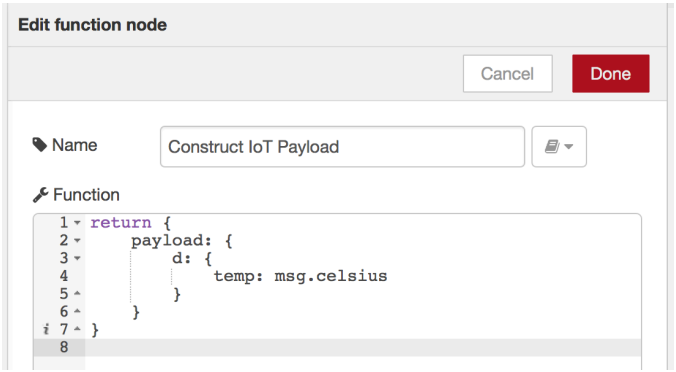
to save and deploy your changes. The LCD should display the outside temperature and will update the temperature from the Weather service every 30 minutes.

Connecting Temperature Sensor to IoT Platform

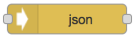
In this section, we will connect the Node-RED application running on the Intel Edison to the Watson Internet of Things Platform service and emit the temperature value from the Grove temperature sensor as device events so that the Node-RED application in IBM Bluemix can react to high temperatures.

1. Add a 

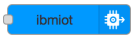
node as shown below.



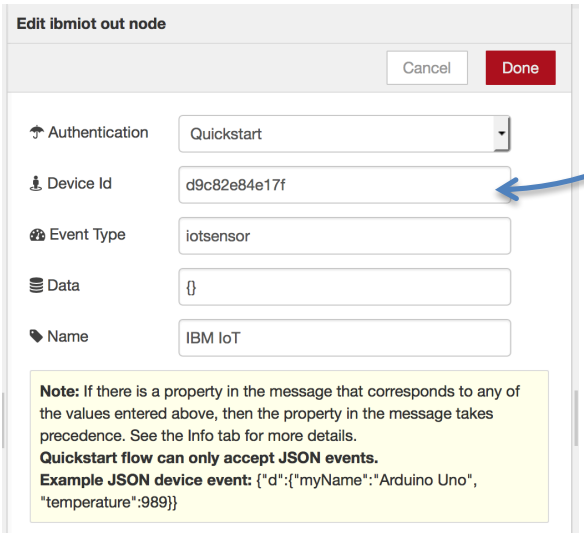
This JavaScript function constructs and returns a JSON object. The JavaScript object has a property named **payload**, which has property named **d**, which contains a property named **temp**. We use the Celsius value from the temperature sensor and assign it to the property **temp**.

2. 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.

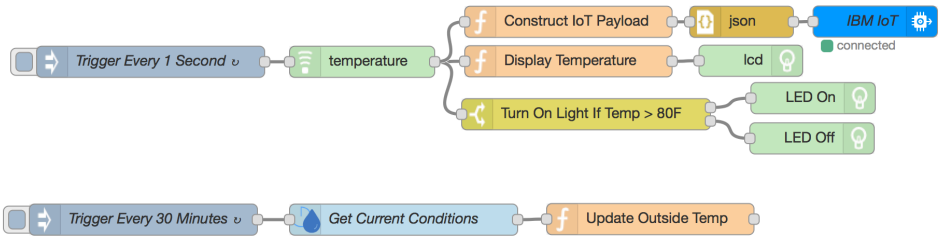
3. Add a 

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 IBM IoT in node of the Node-RED application in the cloud. If you reuse the device ID, close the simulated temperature sensor window to stop the simulated temperature values from conflicting with the real temperature data.



Reuse Device ID from simulated IoT sensor

4. Connect the nodes together as shown below.



Get the code:
`ibm.biz/Bdrese`

5. Click on 

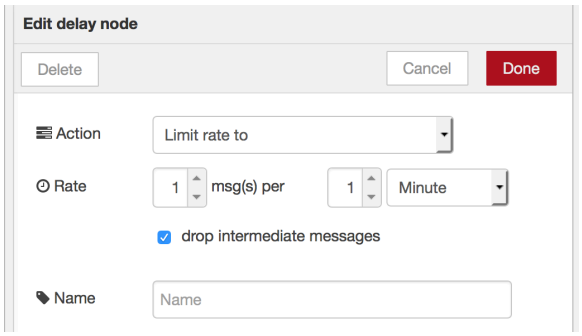
to save and deploy your changes. Return to the Node-RED application in IBM Bluemix. You should see the real temperature being reported.

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 (once 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.

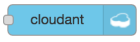
1. Add a 

node as shown below. Without this node, the application would store each temperature event (once a second) into the database. This node will limit this flow to execute once per minute.

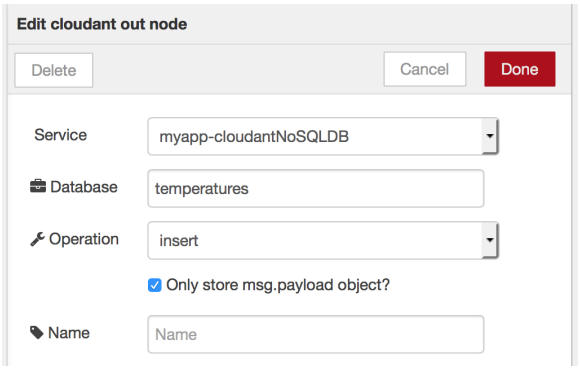


2. Add a 

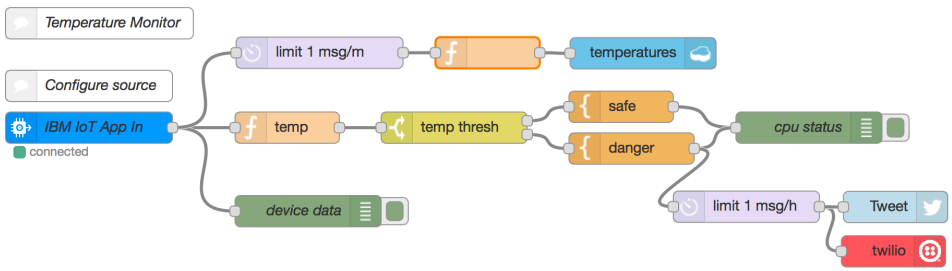
node as shown below. This constructs formats the contents that will be inserted into the database.

3. Add a 

node as shown below. This configures which database to store the data in.



4. Connect the nodes together as shown below.

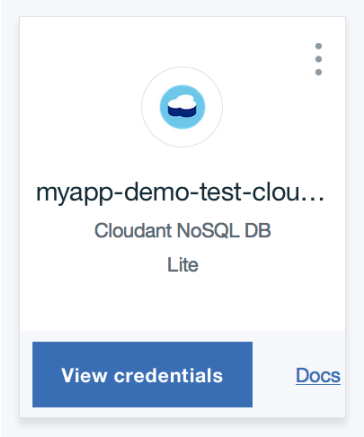


Get the code:
`ibm.biz/BdsCNS`

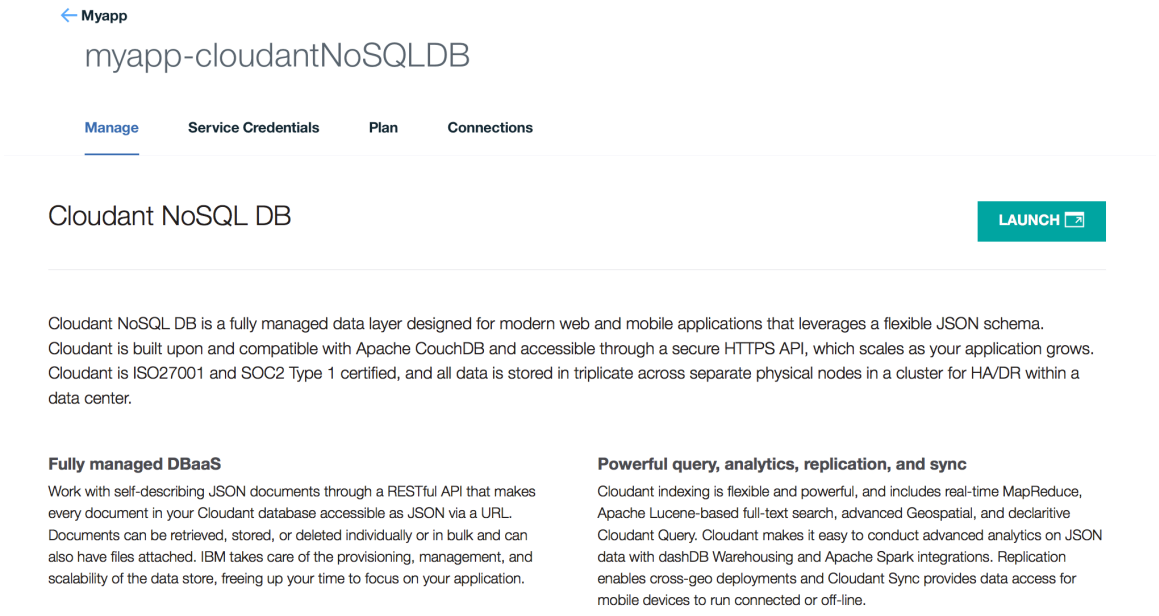
5. Click on 

to save and deploy your changes.

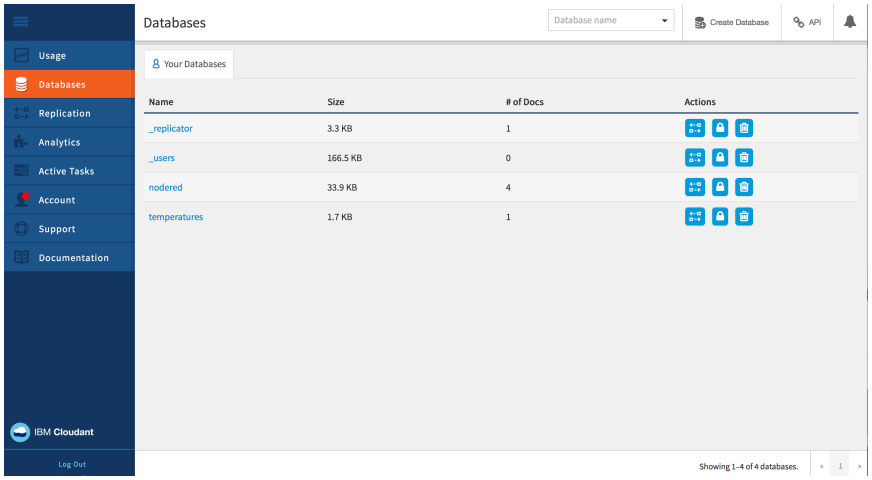
6. Go back to the IBM Bluemix dashboard and the **Connections** tab. Click on the **Cloudant NoSQL DB** service tile.



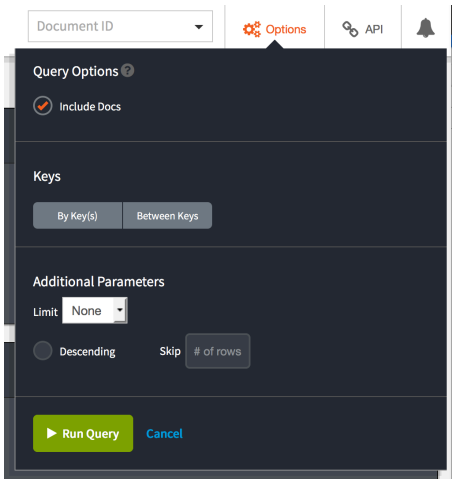
7. Click on the green **Launch** button to access the Cloudant NoSQL database dashboard.



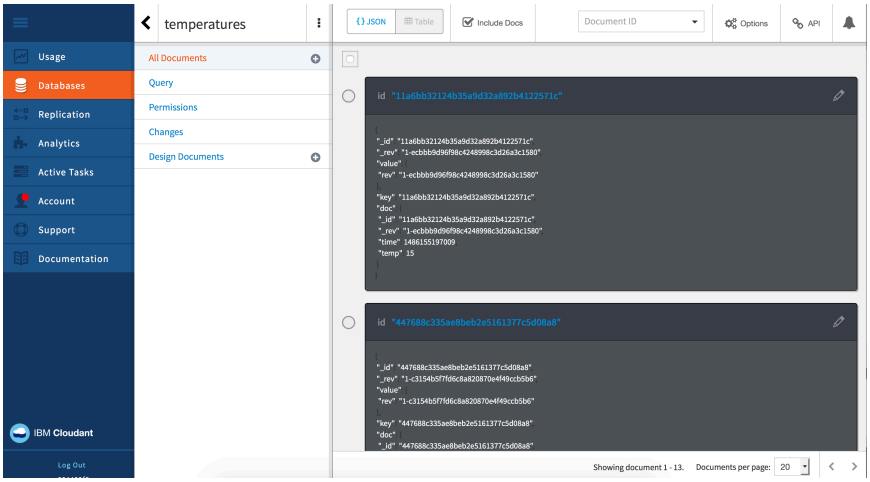
8. This is the Cloudant NoSQL database dashboard. A list of databases is displayed under the **Databases** tab on the left. The database named temperatures contains documents representing each temperature event that has been stored by the Node-RED application. Click on the database named **temperatures.**

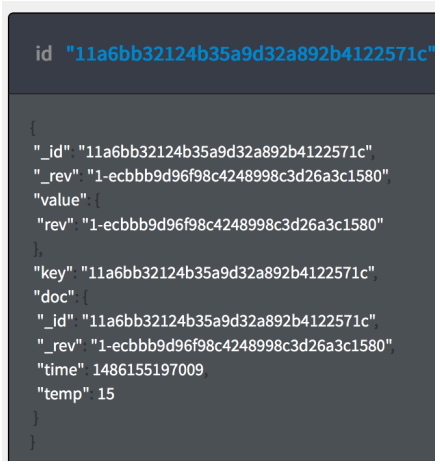


9. To see the expanded view of the documents, click on **Options** link in the top right corner, check the box next to **Include Docs**, and click on the green **Run Query** button.



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





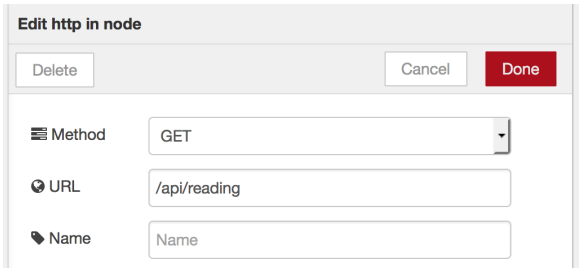
Retrieve Temperatures From Cloudant NoSQL DB

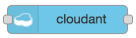
This section shows how to retrieve temperature data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time. This section is optional and can be skipped. Completion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

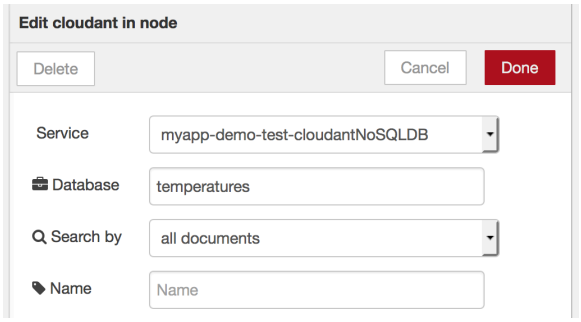
1. Add a 

node as shown below. This sets the endpoint to `/api/reading`.



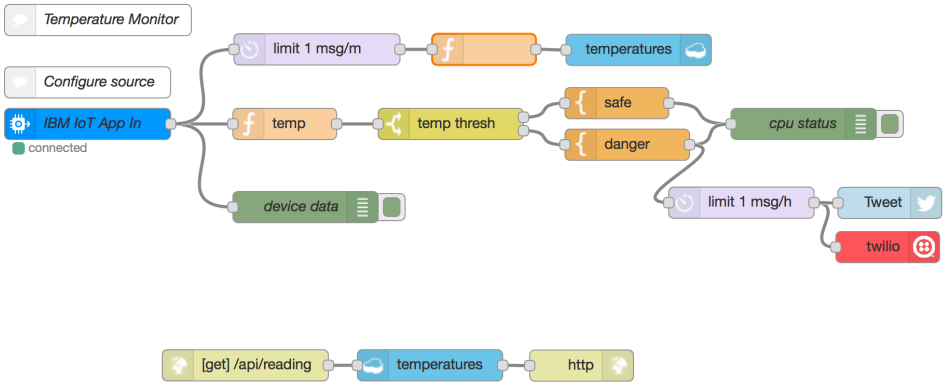
2. Add a 

node as shown below.



3. Finally, add a 

node. Connect the nodes together as shown below.



Get the code:
`ibm.biz/BdsCNv`

4. Click on 

to save and deploy your changes.

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

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

   You should see data returned similar to the following:


```
[{"_id":"11a6bb32124b35a9d32a892b4122571c","_rev":"1-ecbbb9d96f98c4248998c3d26a3c1580","time":1486155197009,"temp":15},
{"_id":"447688c335ae8beb2e5161377c5d08a8","_rev":"1-c3154b5f7fd6c8a820870e4f49ccb5b6","time":1486155136924,"temp":15},
{"_id":"882dba86d6998d0c235ff300f85a85ef","_rev":"1-064ef1bf95af88f6f16fdb23eccf1f57","time":1486154582161,"temp":15},
{"_id":"882dba86d6998d0c235ff300f887cef0","_rev":"1-bc465c323da7ad309e91c7abf48aa919","time":1486155327100,"temp":15},
{"_id":"882dba86d6998d0c235ff300f88bc343","_rev":"1-9b6bd76aa16ca5e185ec4f622962748e","time":1486155395104,"temp":15},
{"_id":"b35c4054bcb1a9ba2b4c7e5a64bc9a3f","_rev":"1-7d78d5c5393b697dc6d40ba4302d0ba5","time":1486154644158,"temp":15},
{"_id":"b35c4054bcb1a9ba2b4c7e5a64bfcc78","_rev":"1-d3e2ba96acdbdf331a4e13e904a91194","time":1486154704679,"temp":15},
{"_id":"b35c4054bcb1a9ba2b4c7e5a64ce9bd9","_rev":"1-98d98c81f578d97ae3f6d910d69cb10b","time":1486154947876,"temp":15},
{"_id":"c6f6f38546325ccb4be45f634d3cd463","_rev":"1-cfc64d79f1e07dbfa5ccbf387aeeb30a","time":1486154765575,"temp":15},
{"_id":"c6f6f38546325ccb4be45f634d43d8fb","_rev":"1-aa3fcd04359a5e06dee46512c2c1fa47","time":1486154887868,"temp":15},
{"_id":"c6f6f38546325ccb4be45f634d5c3a9e","_rev":"1-60a48899bfaf3d7a214f38963d39fddd","time":1486155265598,"temp":15},
{"_id":"dfd1a36d4ad4b7eb8acd9f47987daf29","_rev":"1-d1f49550a1361c9844d07ed7b1b316b6","time":1486155010420,"temp":15},
{"_id":"e5d8cd63363bece3182ec5c492097577","_rev":"1-7f3ee772550d4c67c6ae2c1306669bd2","time":1486155074620,"temp":15},
{"_id":"e5d8cd63363bece3182ec5c4921e73ed","_rev":"1-bc1355f2a4ee46b9210e6337815c6437","time":1486155387100,"temp":15},
{"_id":"f5fe86659597c38185715790d7c9edc1","_rev":"1-67d3e8c092887f7ae7e5a6e17df2286b","time":1486154827500,"temp":15}]
```


   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.