

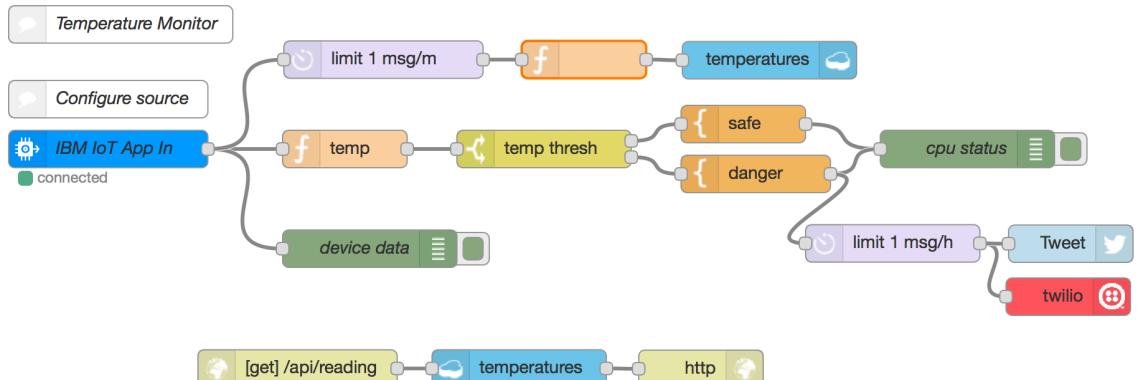
# Tracking Temperature with Watson IoT Starter Boilerplate

## Hands-On Lab

Author: JeanCarl Bisson | [jbisson@us.ibm.com](mailto:jbisson@us.ibm.com) | [@dothewww](https://twitter.com/dothewww)



*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 12 & 15).*



A digital copy of this lab and completed flows can be found at:  
<http://ibm.biz/lab-temperature-sensor-with-iot-starter>

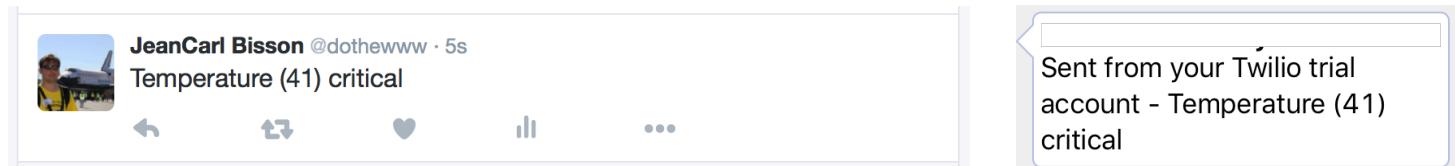
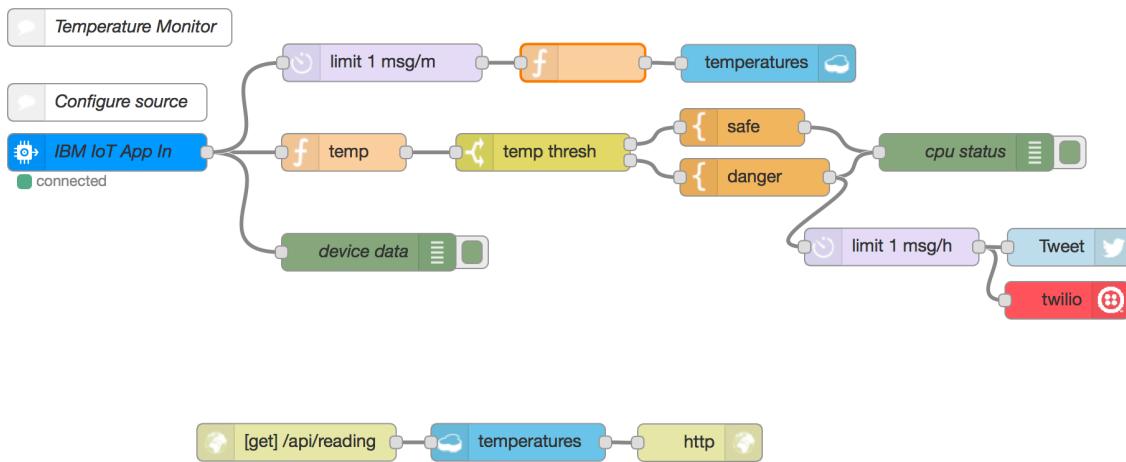


## Table of Contents

Topics covered: Node-RED, Watson Internet of Things Platform, Cloudant NoSQL Database, Twitter, and Twilio.

In this lab, we will use the Internet of Things Boilerplate in the IBM Bluemix Catalog to listen for IoT events from a simulated Quickstart IoT device. The temperature will be sent via Watson's Internet of Things Platform service to the Node-RED application hosted on IBM Bluemix. If the temperature reaches 40°C, we'll send a tweet via Twitter and/or a text message via Twilio. 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 .....	3
Connect to Twitter and Tweet High Temperature .....	9
Connect to Twilio and Text High Temperature .....	11
Store Temperature Into Cloudant NoSQL Database .....	12
Retrieve Temperatures From Cloudant NoSQL DB .....	15
Additional Resources.....	16



```
[{"_id": "11a6bb32124b35a9d32a892b4122571c", "_rev": "1-ecbbb9d96f98c4248998c3d26a3c1580", "time": 1486155197009, "temp": 15}, {"_id": "447688c335ae8beb2e5161377c5d08ab", "_rev": "1-c3154b5f7fd6c8a20870e4f49ccb5b6", "time": 1486155136924, "temp": 15}, {"_id": "882dba86d6d998d0c235ff300f85a85ef", "_rev": "1-064ef1bf95af88f0ff16fd23eccf1f57", "time": 1486154582161, "temp": 15}, {"_id": "882dba86d6d998d0c235ff300f887cef0", "_rev": "1-bc465c323da7ad309e91c7abf48aa919", "time": 1486155327100, "temp": 15}, {"_id": "882dba86d6d998d0c235ff300f88bc343", "_rev": "1-9b6bd76a16ca5e185ec4f622962748e", "time": 1486155395104, "temp": 15}, {"_id": "b35c4054bcbb1a9ba2b4c7e5a64bc9a3c", "_rev": "1-7d78d5c5393b697dc6d40ba4302d0ba5", "time": 1486154644158, "temp": 15}, {"_id": "b35c4054bcbb1a9ba2b4c7e5a64bfcc78", "_rev": "1-d3e2ba96acdbdf331a4e13e904a91194", "time": 1486154704679, "temp": 15}, {"_id": "b35c4054bcbb1a9ba2b4c7e5a64ce9bd9", "_rev": "1-98d98c81f578d97a3f6d910d69cb10b", "time": 1486154947876, "temp": 15}, {"_id": "c6f6f38546325cb4be45f634d3cd463", "_rev": "1-cfc6d79f1e07dbfa5ccbfb387aeeb30a", "time": 1486154765575, "temp": 15}, {"_id": "c6f6f38546325cb4be45f634d43d8fb", "_rev": "1-a3fc0d4359a5e06dee46512c2c1fa47", "time": 1486154887868, "temp": 15}, {"_id": "c6f6f38546325cb4be45f634d5c3a9e", "_rev": "1-60a48899bfaf3d7a214f38963d399fddd", "time": 1486155265598, "temp": 15}, {"_id": "dd1la36d4ad47eb8acd9f47987daf29", "_rev": "1-d1f49550a1361c9844d07ed7b1b316b6", "time": 1486155010420, "temp": 15}, {"_id": "e5d8cd63363bece3182ec5c492097577", "_rev": "1-7f3ee772550d4c67c6ae2c130669db2", "time": 1486155074620, "temp": 15}, {"_id": "e5d8cd63363bece3182ec5c4921e73ed", "_rev": "1-bc1355f2a4ee46b9210e6337815c6437", "time": 1486155387100, "temp": 15}, {"_id": "f5fe86659597c38185715790d7c9edc1", "_rev": "1-67d3e8c092887f7ae7e5a6e17df2286b", "time": 1486154827500, "temp": 15}]
```

# 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.

The screenshot shows the IBM Bluemix Apps dashboard. At the top, there's a navigation bar with 'IBM Bluemix Apps', 'Catalog', 'Support', and 'Account'. Below the navigation is a large circular icon with a grid of squares. The word 'Apps' is centered below the icon. A message says 'You don't have any apps yet. Get started with one of the options that follow, or go to the catalog to create an app.' Below this is a 'Create App' button. There are six quick-start options arranged in two rows of three:

- Create a Cloud Foundry app: Go straight to developing with a Liberty for Java runtime, then add some of our 100+ services to build your app even faster.
- Order a monthly Bare Metal Server: Built to spec with 500GB/month outbound bandwidth included, ready in 2-4 hours.
- Create and run event-driven apps: Write application logic and create actions that can be executed on demand for your web or mobile app.
- Take advantage of IoT
- Build a mobile app in a click
- API Connect

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** tile.

The screenshot shows the IBM Bluemix Catalog. At the top, there's a navigation bar with 'IBM Bluemix Catalog', 'Catalog', 'Support', and 'Manage'. On the left, there's a sidebar with categories like All Categories, Infrastructure, Apps (with a 'Boilerplates' dropdown), and Services. The main area has a search bar and a 'Filter' button. It lists several boilerplate applications:

- ASP.NET Core Cloudant Starter
- Internet of Things Platform Starter (highlighted with a blue box and a callout)
- IoT for Electronics Starter
- Java Cloudant Web Starter
- Java Workload Scheduler Web Starter
- LoopBack Starter
- MobileFirst Services Starter
- Node.js Cloudant DB View Starter
- Personality Insights Node.js Web Starter
- StrongLoop Arc
- Mendix Rapid Apps

- 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.

IBM Bluemix Catalog

Create a Cloud Foundry App

Internet of Things Platform Starter

App name: myapp

Host name: myapp

Domain: mybluemix.net

Selected Plan:

SDK for Node.js™: Default

Cloudant NoSQL DB: Lite

Internet of Things Platform: Lite

VERSION: 0.5.03

TYPE: Boilerplate

REGION: US South

Need Help? Contact Bluemix Sales

Estimate Monthly Cost Cost Calculator

Create

- 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 apps / myapp

myapp Starting Visit App URL Routes

Getting started with Watson IoT Platform Starter

Last Updated: 2017-01-30 | Edit in GitHub

Get started with IBM Watson™ IoT Platform by using the Watson IoT Platform Starter boilerplate. By using the Starter, you can quickly simulate a device, create cards, generate data, and begin analyzing and displaying data in the Watson IoT Platform dashboard.

Getting started Overview Runtime Connections Logs Monitoring API Management

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

Cloud Foundry apps / myapp

myapp Running Visit App URL Routes

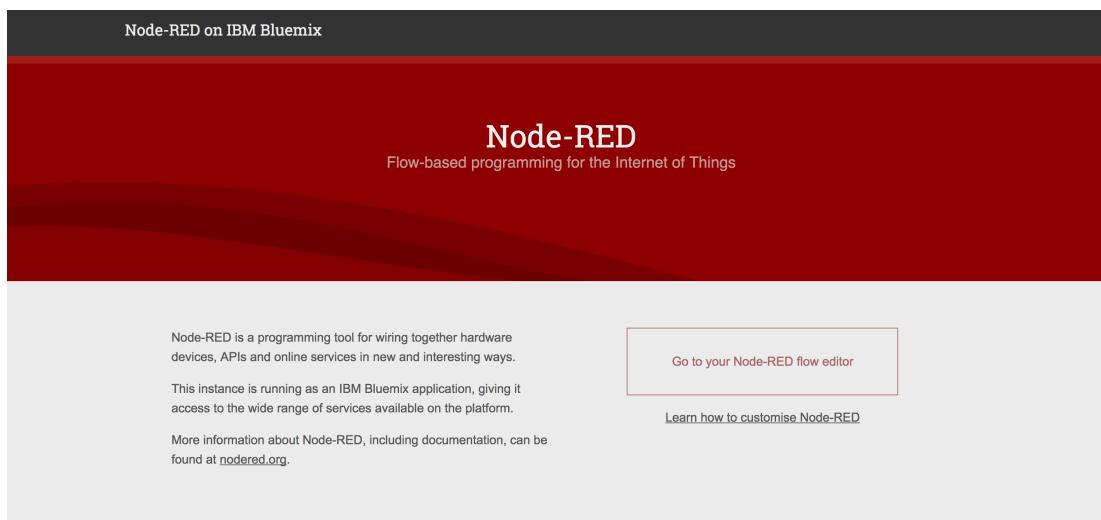
Getting started with Watson IoT Platform Starter

Last Updated: 2017-01-30 | Edit in GitHub

Get started with IBM Watson™ IoT Platform by using the Watson IoT Platform Starter boilerplate. By using the Starter, you can quickly simulate a device, create cards, generate data, and begin analyzing and displaying data in the Watson IoT Platform dashboard.

Getting started Overview Runtime Connections Logs Monitoring API Management

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.

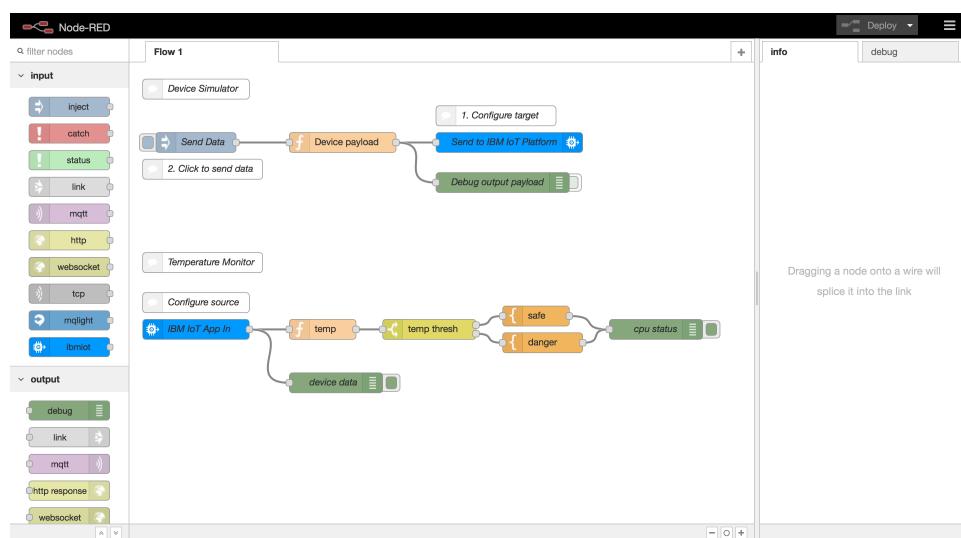


#### Customising your instance of Node-RED

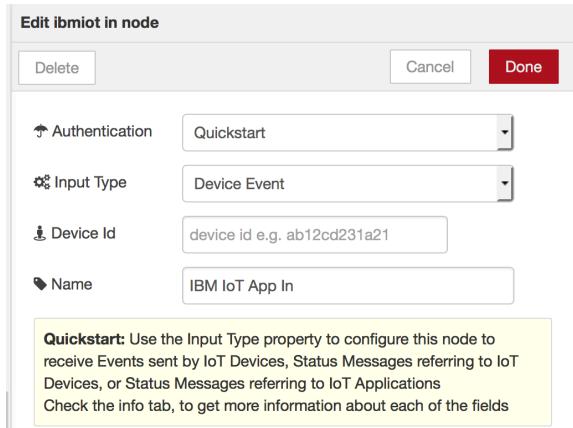
This instance of Node-RED is enough to get you started creating flows.

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.

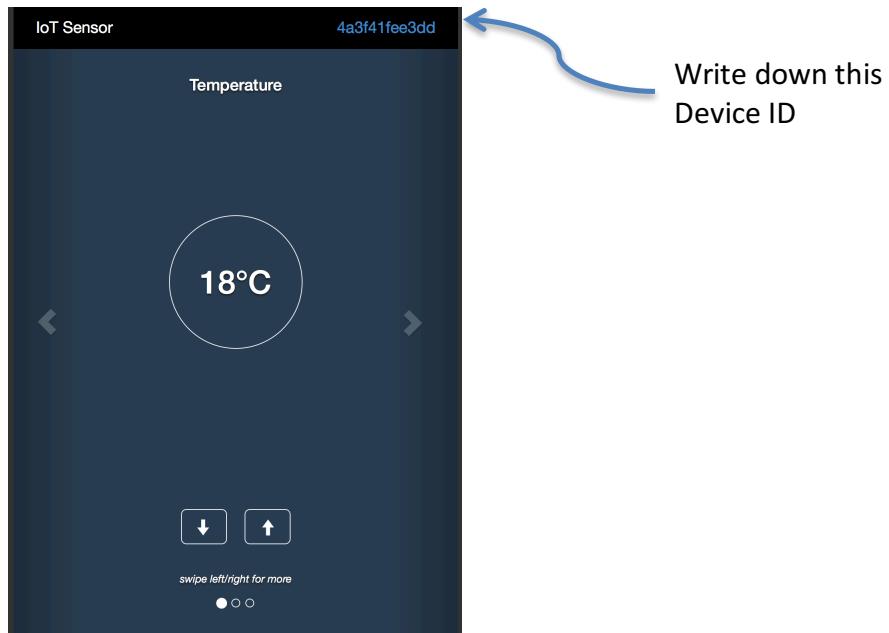
Let's begin by configuring the IBM IoT App In node. Double click on the IBM IoT App In node.



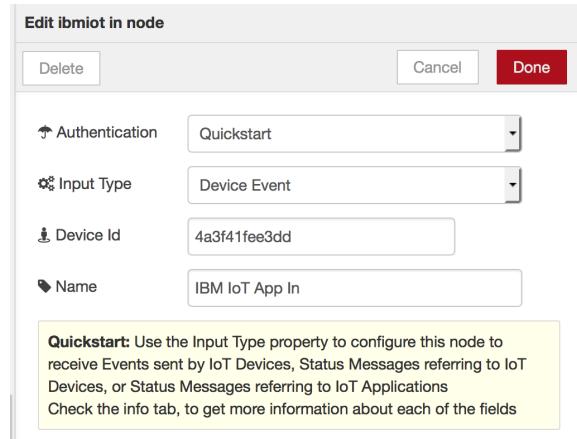
Nodes can be customized in different ways depending on what they do. The IBM IoT node begins a flow when a message is received from a specific device via the Watson IoT Platform. We need a Device Id of a device that is connected to the IoT Platform. We could use a real device, or we can simulate devices.



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.

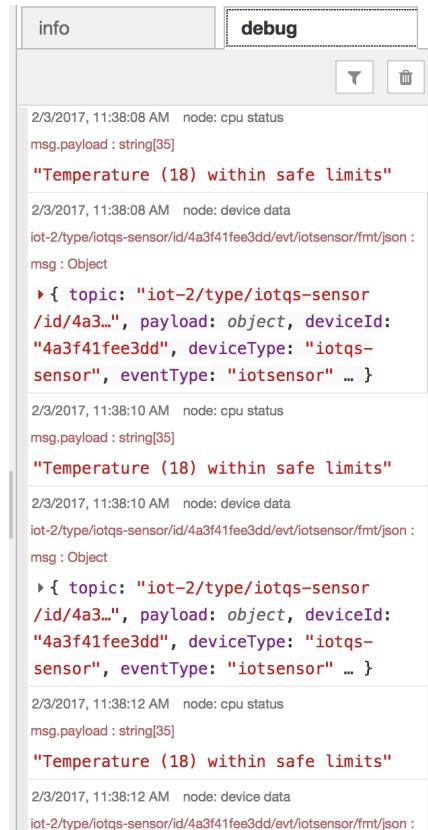


9. Copy this alphanumeric device ID into the **Device ID** field in the Node-RED application as shown below. Click **Done**.

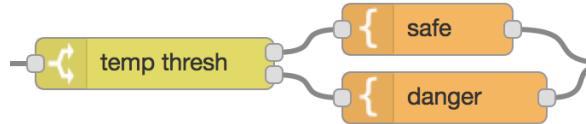


10. Click on the Deploy button in the top right of the screen to save and deploy your changes.

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 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.



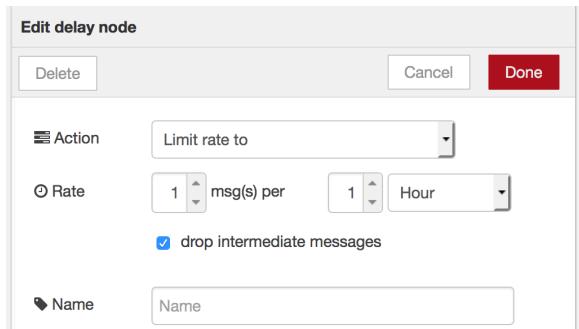
To trigger the flow labeled danger, increment the temperature using the up arrow on the simulated temperature gauge. The message in the debug tab should change.

```
iot-2/type/iotqs-sensor/id/4a3f41fee3dd/evt/otsensor/fmt/json :  
msg : Object  
▶ { topic: "iot-2/type/iotqs-sensor /id/4a3...", payload: object, deviceId: "4a3f41fee3dd", deviceType: "iotqs-sensor", eventType: "otsensor" ... }  
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/otsensor/fmt/json :  
msg : Object  
▶ { topic: "iot-2/type/iotqs-sensor /id/4a3...", payload: object, deviceId: "4a3f41fee3dd", deviceType: "iotqs-sensor", eventType: "otsensor" ... }
```

# Connect to Twitter and Tweet High Temperature

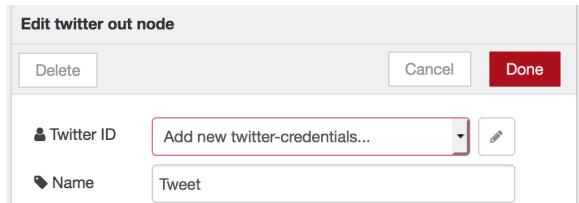
In this section, we will connect a Twitter account to Node-RED and use the Twitter account to tweet when the temperature from the temperature sensor is “dangerous”. This section is optional and may be skipped.

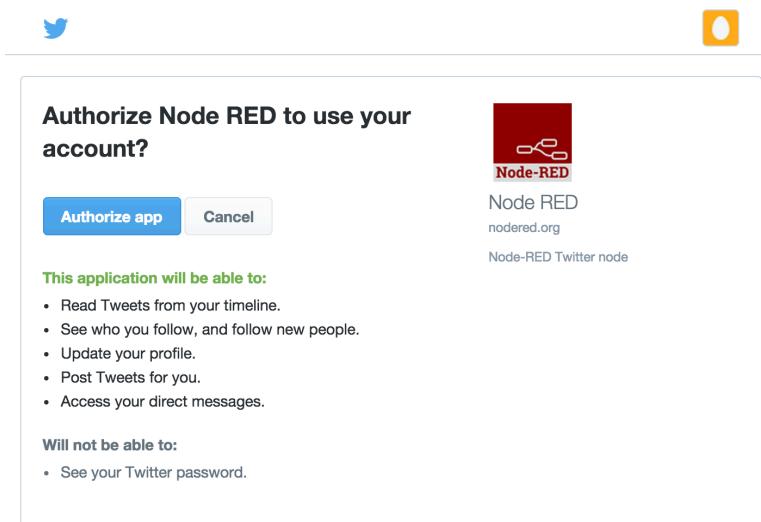
1. Sign up for a Twitter account at <http://twitter.com>. If you already have a Twitter account, proceed to step 2.
2. Add a  node as shown below.



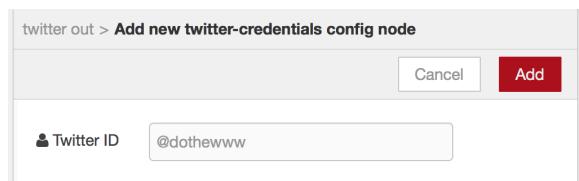
The delay node limits how often the flow is run. Since the temperature is dangerous every second, without this node, a tweet would be sent every second. With this node limiting messages to once an hour, the Twitter node will send a tweet once an hour, dropping any additional messages during the hour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

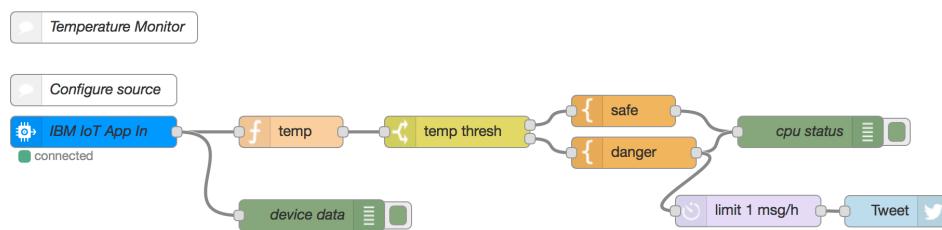




Return back to the node configuration. The settings for the Twitter node should have your username set. Click **Add**.



#### 4. Connect the nodes as shown below.



Get the code:  
[ibm.biz/BdsCN2](http://ibm.biz/BdsCN2)

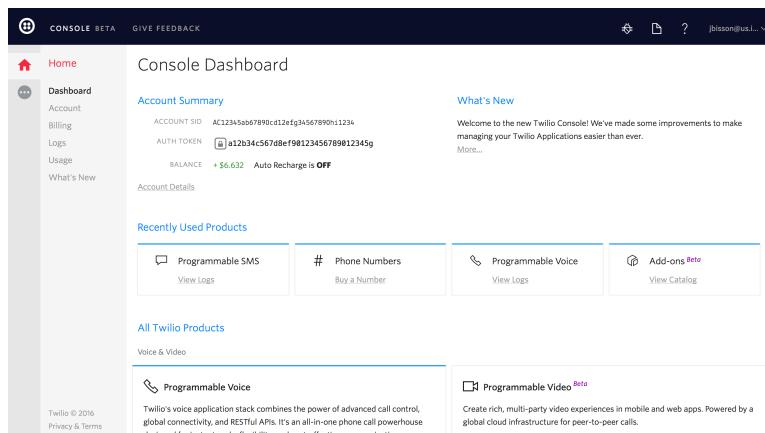
5. Click on to save and deploy the changes.
6. Since the temperature is above 40°C, the switch statement will continue to the “danger[ous]” template, compose a message, and pass it to the Twitter node. The Twitter node uses this message as the content for the tweet.
7. Visit the Twitter timeline for the user you authenticated with and verify the message has been tweeted.



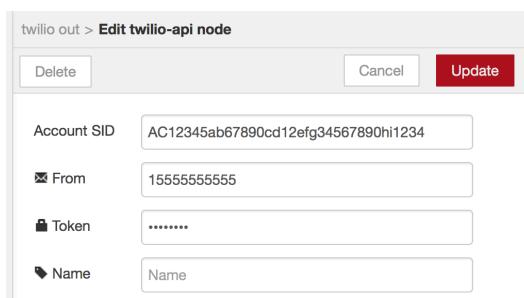
# Connect to Twilio and Text High Temperature

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.

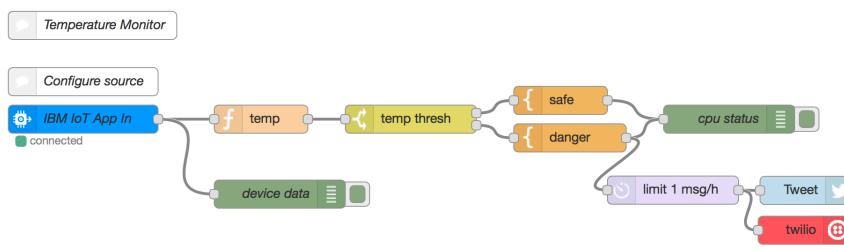
1. Sign up for a Twilio account at <http://twilio.com>. If you already have a Twilio account, sign in. Create a Twilio phone number that will be used in step #3.
2. In the Console Dashboard, click on the lock icon next to Auth Token. Copy the **Account SID** and **Auth Token**.



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.



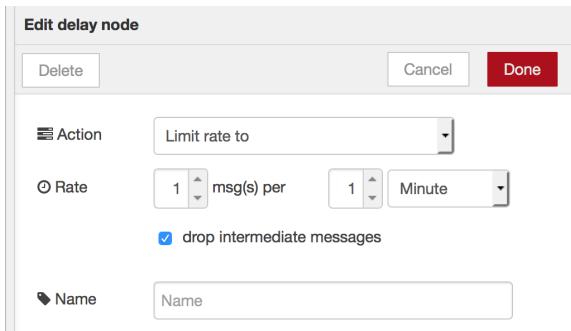
Get the code:  
[ibm.biz/BdsCNP](http://ibm.biz/BdsCNP)

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

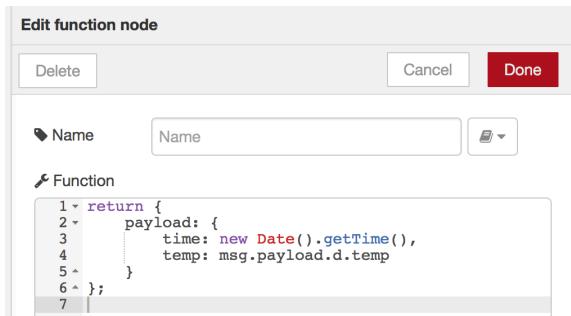
# Store Temperature Into Cloudant NoSQL Database

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.

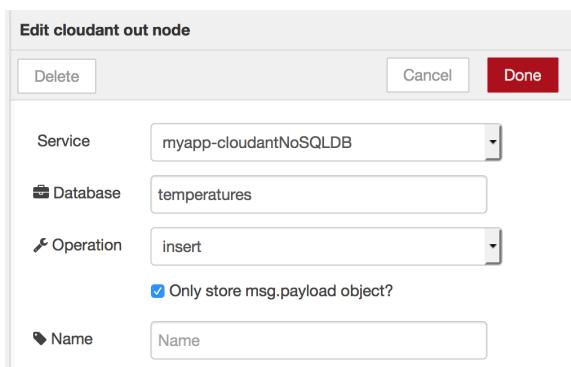
1. Add a  node as shown below. Without this node, the application would store each temperature event (once a second) into the database. This node will limit this flow to execute once per minute.



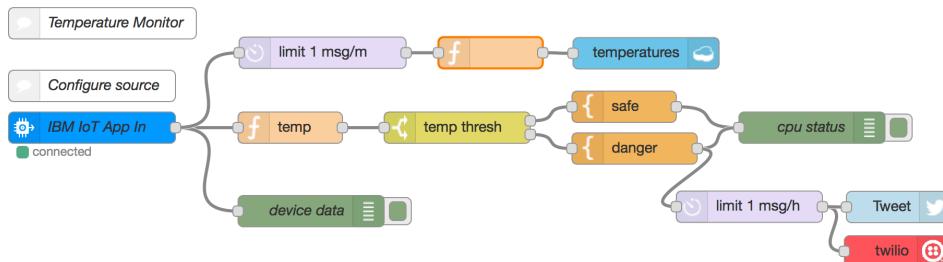
2. Add a  node as shown below. This constructs formats the contents that will be inserted into the database.



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



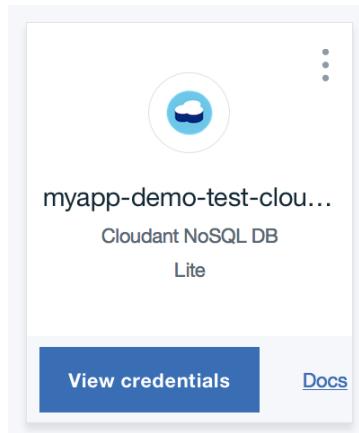
4. Connect the nodes together as shown below.



Get the code:  
ibm.biz/BdsCNS

5. Click on Deploy 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.

Cloudant NoSQL DB is a fully managed data layer designed for modern web and mobile applications that leverages a flexible JSON schema. Cloudant is built upon and compatible with Apache CouchDB and accessible through a secure HTTPS API, which scales as your application grows. Cloudant is ISO27001 and SOC2 Type 1 certified, and all data is stored in triplicate across separate physical nodes in a cluster for HA/DR within a data center.

#### Fully managed DBaaS

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 query, analytics, replication, and sync

Cloudant indexing is flexible and powerful, and includes real-time MapReduce, Apache Lucene-based full-text search, advanced Geospatial, and declarative Cloudant Query. Cloudant makes it easy to conduct advanced analytics on JSON data with dashDB Warehousing and Apache Spark integrations. Replication enables cross-geo deployments and Cloudant Sync provides data access for mobile devices to run connected or off-line.

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**.

Name	Size	# of Docs	Actions
_replicator	3.3 KB	1	
_users	166.5 KB	0	
nodered	33.9 KB	4	
temperatures	1.7 KB	1	

Showing 1-4 of 4 databases. < 1 >

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.

Document ID  Options API Bell

Query Options

Include Docs

Keys

Additional Parameters

Limit: None  Descending Skip # of rows

Cancel

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

temperatures

All Documents

Query Permissions Changes Design Documents

Document ID Options API Bell

id: "11a6bb32124b35a9d32a892b4122571c"

```
{
  "_id": "11a6bb32124b35a9d32a892b4122571c",
  "_rev": "1-ecbbbd9d9f98c4248998c3d26a3c1580",
  "value": {
    "key": "11a6bb32124b35a9d32a892b4122571c",
    "doc": {
      "id": "11a6bb32124b35a9d32a892b4122571c",
      "rev": "1-ecbbbd9d9f98c4248998c3d26a3c1580",
      "time": 148615197009,
      "temp": 15
    }
  }
}
```

id: "447688c335ae8beb2e5161377cd08a8"

```
{
  "_id": "447688c335ae8beb2e5161377cd08a8",
  "_rev": "1-c154b5f7f66ca820870e449cc5b56",
  "value": {
    "key": "447688c335ae8beb2e5161377cd08a8",
    "doc": {
      "id": "447688c335ae8beb2e5161377cd08a8",
      "rev": "1-c154b5f7f66ca820870e449cc5b56",
      "time": 1447688c335ae8beb2e5161377cd08a8,
      "temp": 15
    }
  }
}
```

Showing document 1 - 13. Documents per page: 20 < >

```
id: "11a6bb32124b35a9d32a892b4122571c"

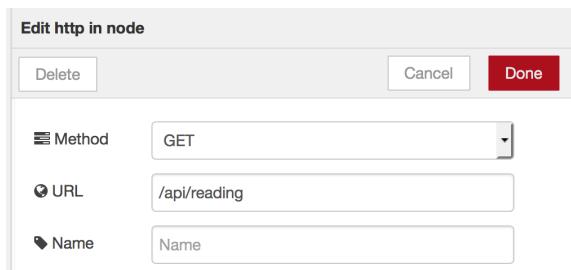
{
  "_id": "11a6bb32124b35a9d32a892b4122571c",
  "_rev": "1-ecbbbd9d9f98c4248998c3d26a3c1580",
  "value": {
    "key": "11a6bb32124b35a9d32a892b4122571c",
    "rev": "1-ecbbbd9d9f98c4248998c3d26a3c1580"
  },
  "key": "11a6bb32124b35a9d32a892b4122571c",
  "doc": {
    "id": "11a6bb32124b35a9d32a892b4122571c",
    "_rev": "1-ecbbbd9d9f98c4248998c3d26a3c1580",
    "time": 148615197009,
    "temp": 15
  }
}
```

# Retrieve Temperatures From Cloudant NoSQL DB

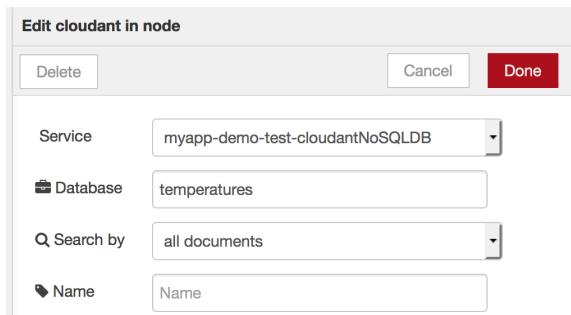
This section shows how to retrieve temperature data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time. This section is optional and can be skipped. Completion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

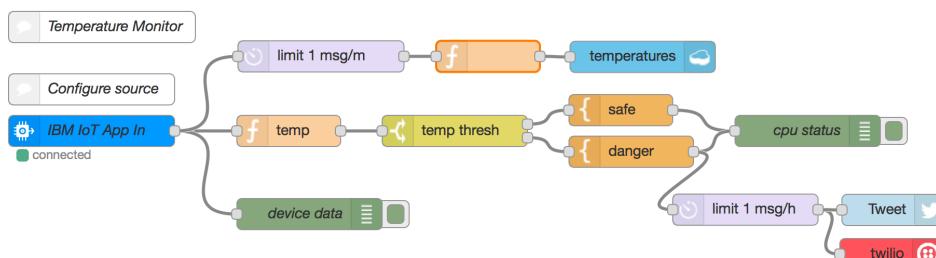
1. Add a  node as shown below. This sets the endpoint to /api/reading.



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](http://ibm.biz/BdsCNv)



- Click on  Deploy to save and deploy your changes.
- Open a browser tab and visit your application's URL, appended by /api/reading. If you chose **myapp** when setting up your application, the URL would be:

<https://myapp.mybluemix.net/api/reading>

You should see data returned similar to the following:

```
[{"_id": "11a6bb32124b35a9d32a892b4122571c", "_rev": "1-ecbbb9d96f98c4248998c3d26a3c1580", "time": 1486155197009, "temp": 15}, {"_id": "447688c335ae8beb2e5161377c5d08a8", "_rev": "1-c3154b5f7fd6c8a820870e4f49cc5b6", "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": "1-9b6bd76aal16ca5e185ec4f622962748e", "time": 1486155395104, "temp": 15}, {"_id": "b35c4054bc1a9ba2b4c7e5a64bc9a3f", "_rev": "1-7d78d5c5393b697dc6d40ba4302d0ba5", "time": 1486154644158, "temp": 15}, {"_id": "b35c4054bc1a9ba2b4c7e5a64bfcc78", "_rev": "1-d3e2ba96acdbdf331a4e13e904a91194", "time": 1486154704679, "temp": 15}, {"_id": "b35c4054bc1a9ba2b4c7e5a64ce9bd9", "_rev": "1-98d98c81f578d97ae3f6d910d9cb10b", "time": 1486154947876, "temp": 15}, {"_id": "c6ff638546325ccb4be45f634d3cd463", "_rev": "1-cfc64d79f1e07dbfa5ccbf387aee30a", "time": 1486154765575, "temp": 15}, {"_id": "c6ff638546325ccb4be45f634d43d8fb", "_rev": "1-aa3fcd04359a5e06dee46512c2c1fa47", "time": 1486154887868, "temp": 15}, {"_id": "c6ff638546325ccb4be45f634d5c3a9e", "_rev": "1-60a48899bfaf3d7a214f38963d39ffdd", "time": 1486155265598, "temp": 15}, {"_id": "df1d1a36d4ad4b7eb8acd9f47987daf29", "_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.

## Additional Resources

Ready to connect real hardware to the Watson Internet of Things Platform, such as the Intel Edison or the Raspberry Pi. Visit [ibm.biz/node-red-labs](http://ibm.biz/node-red-labs) for additional labs.