Interop ITX 2018

April 30th – May 4th

Using IoT and cognitive services to provide a personalized experience

Authors:
Cesar Ivan Orozco Cervantes, Software Engineer, Cesar.Cervantes@hcl.com
Heath Thomann, Software Engineer, Heath.Thomann@hcl.com

Table of Contents

1. Introduction

Every day the world becomes more and more connected. Everything from the cars we drive, the phones we use, devices we wear, and to the factories producing our products. These devices can be linked together and to a huge network of data. It is estimated that by 2020 there will be 50 billion Internet of Things (IoT) devices and an estimated 90% of cars will be connected.

Taking advantage of IoT we can build applications that provide a more personalized experience for each individual user, not only by displaying personalized content, but also by changing the look and feel of the application itself to adapt it to how people "feel" about certain topics, this can be achieved by using the cognitive capabilities of Watson to analyze social networks, in combination with physical sensors we can personalize the experience even more by using information from the environment.

Large companies and independent developers can implement applications that use this combination of technologies to create a new generation of applications, making both, the products offered by these applications and the applications themselves more attractive to users.

By implementing this system, users will have the possibility to enter a set of keywords, Watson can analyze social networks (user's own posts or the social network in general) using this keywords to determine how to change the look and feel of the application, for example, if Watson detects that users are upset, using color psychology, it can

automatically change the colors of the application in an attempt to influence the mood of the users, combining this system with light sensors, the application can adapt the colors chosen by Watson to the current light conditions in the user's environment.
SimpleLink SensorTag by Texas Instruments is a relatively low cost IoT-ready device designed for early adopters and IoT enthusiasts to start experimenting. It has sensors for different purposes: Infrared thermopile temperature, 9-axis motion, altimeter/pressure, ambient light, magnet and humidity sensors. It works under Bluetooth Low Energy (BLE) technology supported by Bluetooth 4.0 and is compatible with Beacons and Bluetooth Smart. The Texas Instruments BLE mobile application allows you to quickly put the data gathered by these sensors right onto the IBM IoT Foundation (IBM IoTF) cloud, either on the Quick Start boilerplate or well in a custom IoT application powered by IBM IoTF.
While IoT is still a buzz word, it is matured enough that most people have a general idea what it is, and what a 'thing' is. But what might not be as clear is how to connect to an IoT 'thing' to do something useful. That is, the concept of IoT might still be somewhat nebulous, and using it to one's advantage is still a mystery for most people.
Given the proliferation of small, low consumption sensors, there is a need for software that can be used to create IoT applications that can tie together devices and its data to users. Enter Node-RED. Node-RED is used to create IoT applications. It is a Flow-based Programming tool, meaning it is a network of connected "nodes" where each node has a specific purpose. For example, a node can be created to listen to a sensor on a device, such as a SensorTag, and pass the sensor data to a temperature function node which might act upon the data at a certain threshold by sending the data to an email, or twitter, node. The network of connected (wired) node is consider a flow.
In this workshop, the attendee will learn how to connect an IoT device to the IBM IoT Foundation (running on IBM Bluemix) to retrieve its data and combining it with Watson services to personalize the experience in the application. A mobile / web application, that displays relevant information to the user, and a Bluemix application, running the backend and gathering the data from IoT Foundation, will be developed using Rational Application Developer.

2. Before you go...

2.1 Prerequisites

You will need:

- An IBM Bluemix account. If you do not have one yet, go to https://ibm.biz/BluemixSignUp and sign up for one.
- A Twitter account. This can be your personal account, or one you created just for this event.

- Very basic familiarity with Node-RED (although an intro/overview will be given in the lab): nodered.org
- Very basic familiarity with the Texas Instruments SensorTag: http://www.ti.com/ww/en/wireless_connectivity/sensortag/

2.2 Lab overview and software used

This lab will use Rational Application Developer (RAD) which is installed on the provided Virtual Machine. It will also use the Node-RED server shipped with RAD and the Node-RED Editor view contained in RAD. Using the RAD Node-RED Editor view (which is an embedded instance of the Node-RED editor) we will create and deploy two Node-RED flows. In general, the first flow will consist of the following:

1. A Twitter node that will connect to a user's Twitter account and retrieve tweets from the user account.
2. A Watson node which connects to the Watson Tone Analyzer. The tweet text will be sent to the analyzer to analyze the tone of the tweet. The analyzer will return a score for 5 different types of tones (Anger, Disgust, Fear, Joy, or Sadness).
3. A node to process the results from the Watson Tone Analyzer to find the highest score, and corresponding tone.
4. A node to save the highest tone.
5. An http request/response node to return the tone as a JSON string.

The second flow will basically consist of the following:

1. An IBM IoT node that will connect to either a Texas Instrument SensorTag IoT sensor, or a simulated sensor provided by BlueMix.
2. A node to extract and save light data from the SensorTag.
3. A node to save the light data.
4. An http request/response node to return the light data as a JSON string.

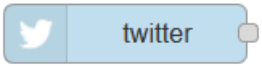
The consumer of the tone and light data from the Node-RED flows described above will be a mobile application. This application will use the tone and light data to change the color of the app text/background.

3. Instructions

3.1 Create the Twitter and Watson Tone Analyzer Node-RED application

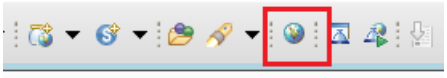
We will start the lab work by create the Twitter and Watson Tone Analyzer flow.

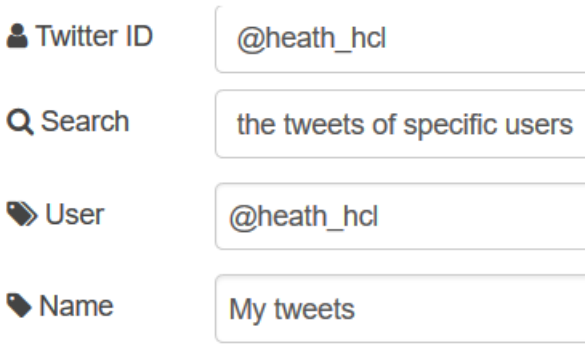
3.1.1 Configure the Twitter node

1. Under the **social** section within the palette (the palette is on the left-hand side of the Node-RED editor), find the twitter node: 

2. Drag-and-drop the twitter node into the flow workspace.
3. Double-click the twitter node.
4. Click the pencil icon: 

   NOTE: The next few steps are a slight work around to an issue where clicking the **Click here to authenticate with Twitter** button is not working as expected. If the following steps still do not allow you to authenticate, please see Appendix B for a alternate steps.
5. Right-click the button labeled **Click here to authenticate with Twitter**.
6. Select **Copy Link Location**.
7. In the RAD tool bar, click the following icon (see red box) to open a web browser:

   
8. In the web browser, right click the address bar and select paste and hit enter.
9. Log in to Twitter and click the **Authorize app** button. A message should be displayed indicating that authorization was successful and directing you to close the browser windows. Close the browser window to return to the Node-RED Editor view.
10. If your twitter handle is not filled in where the **Click here to authenticate with Twitter** buttons is, click the button again and your twitter id should show up.
11. Click the **Add** button.
12. In the **Search** pull down, select **the tweets of specific users**.
13. In the **User** text field enter your twitter handle
14. In the **Name** text field, enter a name of your choice (e.g. My tweets). The configured node should look as follows (my twitter handle is @heath_hcl):



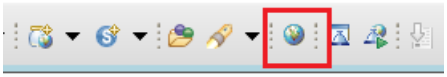
15. Click the **Done** button.

16. Click the **Deploy** button to deploy the flow to the server.

3.1.2 Create the Watson Tone Analyzer service

We will now create the **Watson Tone Analyzer** service in BlueMix. We will use this service to analyze the tone of the tweets. The **Watson Tone Analyzer** service can recognize 5 tones as follows: anger, disgust, fear, joy and sadness.

1. In the RAD tool bar, click the following icon (see red box) to open a web browser:



2. In the web browser, enter the following URL:

   https://console.bluemix.net/dashboard/apps

3. Log in to your BlueMix account.
4. Click the **Create Resource** button.
5. In the search bar enter **tone analyzer** and click **filter**.
6. In the search results, click the **Tone Analyzer**.
7. The **Lite** plan should be selected by default, if not, select it and click the **Create** button. NOTE: at this time, BlueMix shows an 'error' in the resultant page when creating the service. It appears to be harmless.
8. In the resultant page, click **Service Credentials** on the left-hand side menu.
9. If there are pre-defined credentials skip to the next step. If there are no credentials, click the **New Credentials** button, then **Add**.
10. Under the **Actions** column click **View credentials**. This gives the URL to use to connect to the analyzer, as well as the username and password.
11. Take note of the credentials. They should look something like this (the username and password will be different of course):

```
{
  "url": "https://gateway.watsonplatform.net/tone-analyzer/api",
  "username": "cb38dd76-9690-4e38-a723-0f9fd950106f",
  "password": "mruMaC6rPrA4"
}
```

12. Leave open the browser to the credentials page, as we will come back to it in the steps below.

13. Return to the Node-RED Editor view.

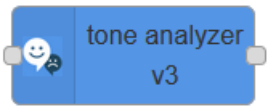
3.1.3 Configure the Watson Tone Analyzer node

Now that we have created the tone analyzer service, we can continue with the Node-RED flow and configure the tone analyzer node. Node-RED does not have the Watson nodes pre-installed. To get the Watson nodes, we need to install a Node-RED package. Information on the Watson nodes can be found here:
https://flows.nodered.org/node/node-red-node-watson

To install these nodes, we need to do the following:

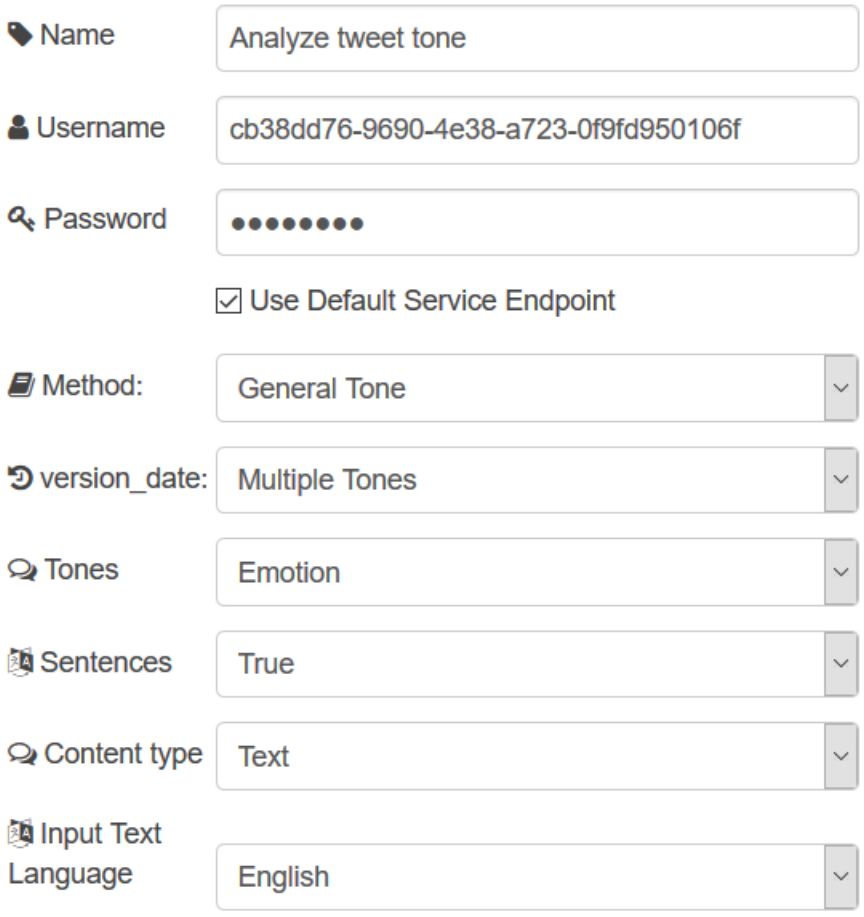
1. In the Node-RED editor in RAD, click on the menu button: 
2. Click **Manage Palette**.
3. Click the **Install** tab.
4. In the **search modules** text box enter ***node-red-node-watson***.
5. Find the ***node-red-node-watson*** in the results and click the **install** button next to it.
6. In the resultant window (titled **Install nodes**) click **Install**.
7. The install might take 30 seconds. When it is finished, click **close**.

With the Watson nodes installed, we can continue creating the flow.

8. In the Node-RED editor in RAD, look under the **IBM Watson** section in the palette and find the tone analyzer node: 

9. Drag-and-drop the tone analyzer node into the flow workspace (drop it next to the recently configured twitter node).
10. Double-click the analyzer node.
11. In the **Name** field, enter a name, for example, enter My tone analyzer.

12. For the **Username** field, enter the username provided by the Watson Tone Analyzer service from the steps above (e.g. cb38dd76-9690-4e38-a723-0f9fd950106f from above).
13. For the **Password** field, enter the password provided by the Watson Tone Analyzer service from the steps above.
14. Ensure that the **Use Default Service Endpoint** check box is checked.
15. In the **Method** box, select **General Tone**.
16. In the **version_date** enter **Multiple Tones.**
17. In the **Tones** box, select **Emotion**.
18. In the **Sentences** box, select **true**.
19. In the **Content type** box, select **Text**.
20. The configured node should look as follows:



21. Hit the **Done** button.

22. Click the **Deploy** button to deploy the flow to the server.

3.1.4 Connect (wire together) the Twitter and Watson nodes

Now that the two nodes are fully configured, we need to connect them.

1. From the twitter node output (the small grey circle on the right-hand side of the node), left mouse click and hold.

2. Drag a line from the twitter output to the tone analyzer input. The entire flow to this point should look as follows:

3. Click the **Deploy** button to deploy the flow to the server.

3.1.5 Process the tone analyzer output

The tone analyzer will return a score for 5 different types of tones (Anger, Disgust, Fear, Joy, or Sadness). We want to find the highest score, and corresponding tone.

1. Find the **function** node in the palette: 

2. Drag-and-drop the function node into the flow workspace (drop it next to the recently configured tone analyzer node).

3. Double-click the function node.

4. In the **Name** text field, enter **Get highest tone**.

5. We need to add some code to extract the scores and tones. Copy/paste the following code into the **Function** text box:

```
var ar = msg.response.document_tone.tone_categories[0].tones;
var score = ar[0].score;
var tone = ar[0].tone_name;

for (i = 1; i < ar.length; i++) {
```

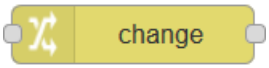
```
        if (score < ar[i].score){
            score = ar[i].score;
            tone = ar[i].tone_name;
        }
    }

    msg.payload="The highest tone is: " + tone;
    msg.tone=tone;
    return msg;
```

6. Click the **Done** button.
7. Connect the output of the tone analyzer to the input of the function node.
8. Click the **Deploy** button to deploy the flow to the server.
9. At this point in time, the flow should look as follows:

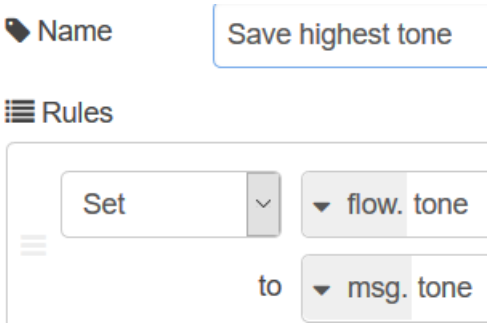
3.1.6 Save the highest tone

After finding the highest tone, the value needs to be saved for future use.

1. Find the **change** node under the **function** section in the palette:



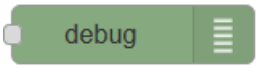
2. Drag-and-drop the change node into the flow workspace (drop it next to the recently configured function node).
3. Double-click the change node.
4. In the **Name** text field enter **Save highest tone**.
5. Make sure **Set** is selected in the **Rules** section.
6. Define the rule as follows:



7. Click the **Done** button.
8. Connect the output of the function node to the input of the change node.
9. Click the **Deploy** button to deploy the flow to the server.
10. At this point in time, the flow should look as follows:

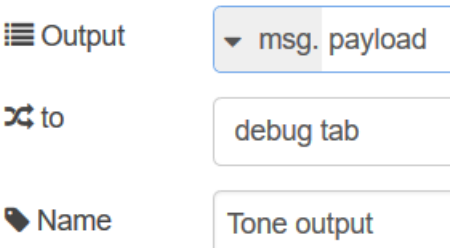
3.1.7 Add debug nodes to view tweet and tone data

If everything is configured correctly, the tweet node should be receiving the texts you send, and the analyzer should process them. It would be nice to see some output from this process. Let's add a debug node to see the tone data.

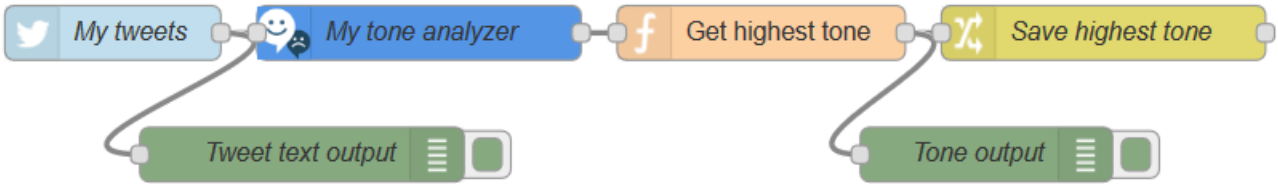
1. Find the **debug** node in the palette: 

2. Drag-and-drop the debug node into the flow workspace (drop it next to the twitter node).
3. Double-click the debug node.
4. Add **tweet.text** to the **Output** text box, and add a name of **Tweet text output** to the **Name** text field, as follows:

5. Click the **Done** button.
6. Connect the output of the twitter node to the input of the debug node.
7. Drag-and-drop another debug node into the flow workspace (drop it next to the function node).
8. Double-click the debug node, and this time simply add a **Name** as **Tone output**, as follows:



9. Click the **Done** button.
10. Connect the output of the function node to the input of the second debug node.
11. At this point in time, the flow should look as follows:

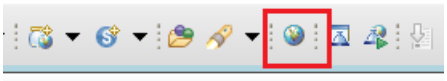


12. Click the **Deploy** button to deploy the flow to the server.

3.1.8 Let's test our work so far

Let's now have some fun and test our flow.

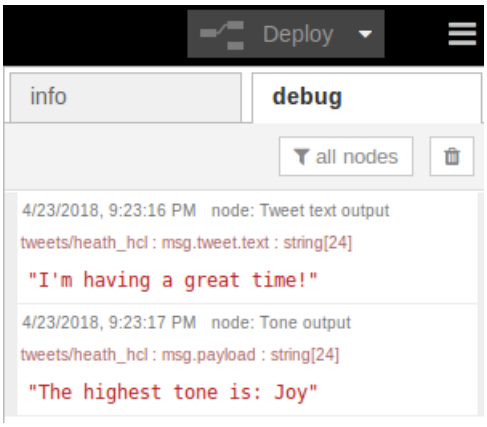
1. In the Node-RED editor find the **Debug** tab on the right-hand side of the editor and click it.
2. In the RAD tool bar, click the following icon (see red box) to open a web browser:



3. Go to your twitter account and enter the following tweet: **I'm having a great time!!**

4. Go back to the RAD Node-RED Editor view, and watch the Debug tab in the Node-RED editor. In about 30-60 seconds your tweet should arrive.

5. Once the tweet arrives and is processed, the debug output will look as follows:

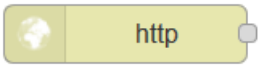


6. Let's try another one. In Twitter enter the following tweet: **I'm going to be very sad if it rains.**

7. After 30-60 seconds, the debug output should contain your tweet, along with the tone **Sadness**.

3.1.9 Create an HTTP request/response for external application

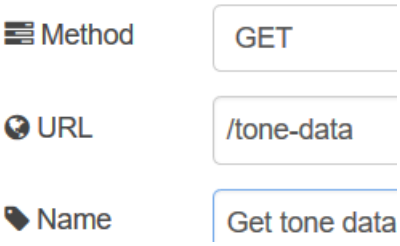
At this point we have saved the tone. To allow external applications access to the tone, we need to create an http request/response to allow external access. Later in the demo, our mobile application will access the tone via the http request.

1. Find the **http** node in the palette under the input section (don't use the http request node, there is a distinction): 

2. Drag-and-drop the http node into the flow workspace (drop it under the existing flow).

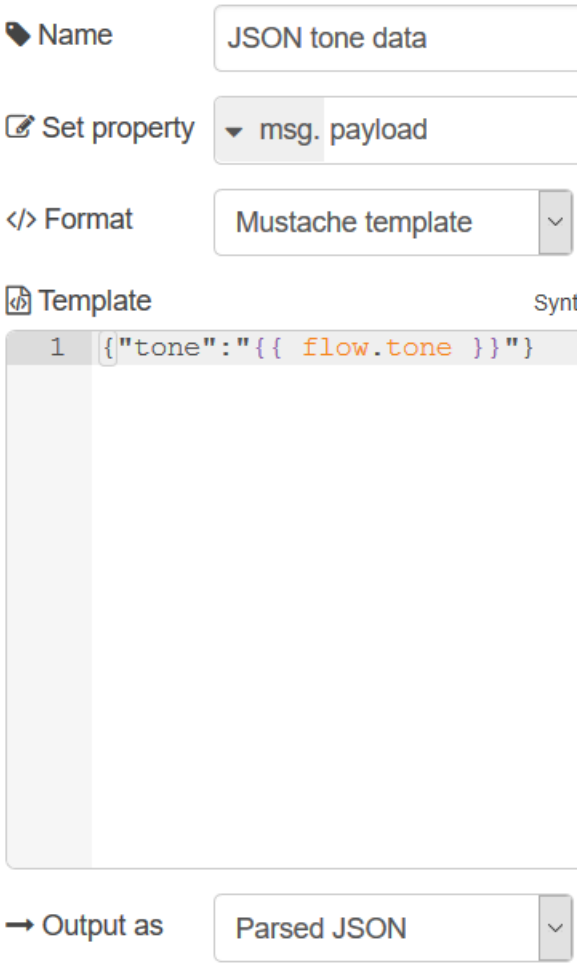
3. Double-click the http node.

4. Configure the node as follows:

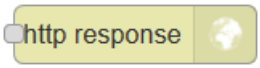


5. Click the **Done** button.
6. Find the **template** node in the palette under the **function** section:

7. Drag-and-drop the template node into the flow workspace next to the http node.
8. Double-click the template node.
9. In the **Name** text box enter a name, e.g. **JSON tone data**.
10. In the **Template** body enter **{"tone":"{{ flow.tone }}"}**
11. Scroll down (this is easy to miss!!) and find the **Output as** pull down, select **Parsed JSON**.
12. The configured node should look as follows:

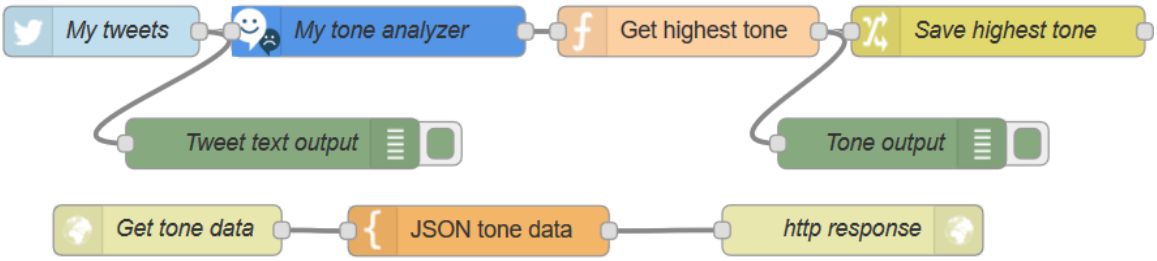


13. Click the **Done** button.
14. Connect the output of the http node to the input of the template node.
15. Find the **http response** node in the palette under the **output** section:



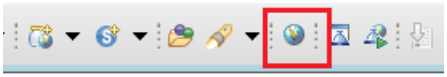
16. Drag-and-drop the http response node into the flow workspace next to the template node.
17. Double-click the http response node.
18. Simply add **http response** in the **Name** field.
19. Click the **Done** button.

20. Connect the output of the template node to the input of the http response node.
21. Click the **Deploy** button to deploy the flow to the server.
22. The entire flow should look as follows:



3.1.10 Test the HTTP request/response

1. In the RAD tool bar, click the following icon (see red box) to open a web browser:



2. Enter the URL: http://localhost:1880/tone-data
3. The most current tone data should be returned in JSON format.

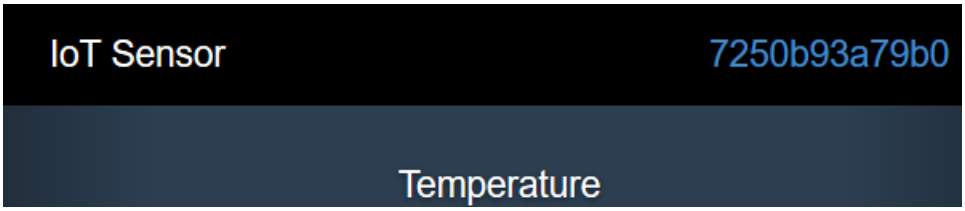
3.2 Connect to an IoT device and create the IoT Node-RED flow

The last Node-RED flow we will create is one that will receive data from either an IoT sensor simulator, or a Texas Instruments SensorTag IoT device. There will be a few SensorTag IoT devices provided by the instructors which will be used to collect light levels (lux) in the room. The instructors will provide the device id of the SensorTag devices (one such device id will be 247189cc0501). However, some participants might choose to simulate an IoT device. IBM offers an IoT device simulator that similarly as the SensorTag. The IoT device simulator provides temperature simulation rather than lux. However, that is not important, the important part is having a simulator to provide a varying number, be it lux or temperature. The following steps will use the IoT simulator. However, as will be seen below, it is simple to switch from the IoT simulator to the SensorTag devices in the lab just by changing the device id.

3.2.1 Connect to an IBM IoT sensor simulator

1. Go to the following site which is an IBM IoT sensor simulator: https://quickstart.internetofthings.ibmcloud.com/iotsensor/

2. At the top of the sensor you should see the following:



3. The number (i.e. 7250b93a79b0) in the upper right-hand corner is the sensor id. Copy this number as it will be used below when creating the Node-Red flow.

   - Note that each time you refresh the sensor web page, a new number will be given. Furthermore, this number will time-out eventually. Keep this in mind when using the sensor id.
   - As was mentioned above, a SensorTag device will be available on the day of the lab. The SensorTag will have a sensor id just like the above simulated sensor. This makes it easy to use one or the other. This will become more obvious in the steps below.

3.2.2 Create the IoT Node-RED flow

To connect to the IoT sensor from Node-RED, we will use the IBM IoT QuickStart nodes. As was performed above for the Watson nodes, we will need to install some additional nodes. These nodes are described here:
https://flows.nodered.org/node/node-red-contrib-scx-ibmiotapp

To install these nodes, we need to do the following:

1. In the Node-RED editor in RAD, click on the menu button: 
2. Click **Manage Palette**.
3. Click the **Install** tab.
4. In the **search modules** text box enter ***node-red-contrib-scx-ibmiotapp***
5. Find the ***node-red-contrib-scx-ibmiotapp*** in the results and click the **install** button next to it.
6. In the resultant window (titled **Install nodes**) click **Install**.
7. The install might take 30 seconds. When it is finished, click **close**.

With the IBM IoT nodes installed, we can continue creating the flow.

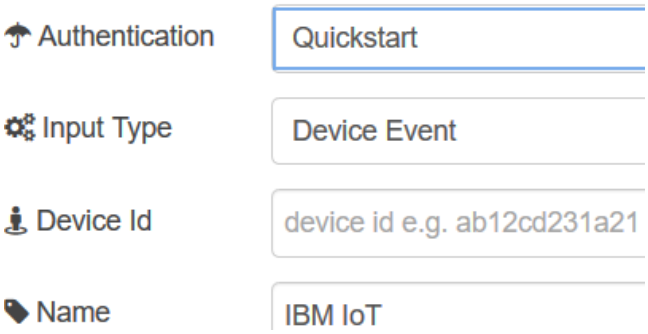
8. In the Node-RED editor, click the  button to create a new flow.

9. Find the **ibmiot** node in the palette under the **input** section: 

10. Drag-and-drop the ibmiot node into the flow workspace.

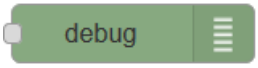
11. Double-click the ibmiot node.

12. The ibmiot configurations will look as follows:



13. In the Device Id text field, enter the id from the IoT device (e.g. from the above steps the id was 7250b93a79b0). This can be the device id from the IoT sensor simulator, or from the SensorTag provided the day of the lab.

14. Click the **Done** button.

15. Find the **Debug** node under the **output** section in the palette:


16. Drag-and-drop the debug node into the flow workspace.

17. Double-click the debug node.

18. In the **Output** text field, enter **payload.d.temp** if you are using the IoT sensor simulator. If you are using the SensorTag on the day of the lab, enter **payload.d.light**. Ensure that the **msg.** is selected.

19. In the **Name** field enter **IoT sensor output**.

20. Click the **Done** button.

21. Connect the output of the ibmiot node to the input of the debug node.

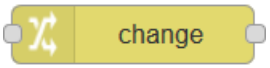
22. Click the **Deploy** button to deploy the flow to the server.

23. The flow should look as follows:

24. Find the **Debug** tab on the right-hand side of the editor and click it.

25. The data from the IoT sensor should be printed in the debug section about every second. If you connected to the IoT sensor simulator, the value set should be printed. You can go back to your browser where you connected to the sensor and change the temperature. The change should be reflected in the Node-RED debug output. If you are connected to the SensorTag, the instructor will manipulate the sensor to show how the data changes.

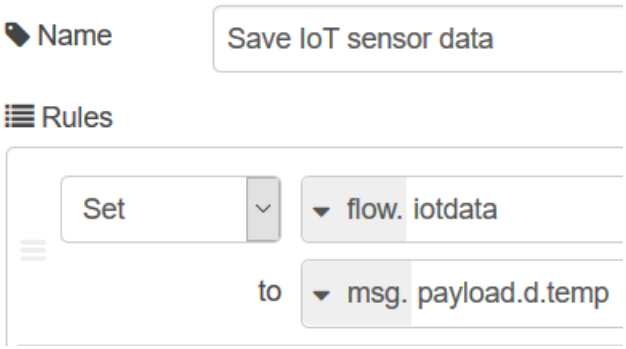
26. Find the **change** node under the **function** section in the palette:



27. Drag-and-drop the change node into the flow workspace, dropping it under and near to the ibmiot node.

28. Double-click the change node.

29. In the **Name** text field enter **Save IoT sensor data**. Make the **Rules** look as follows:



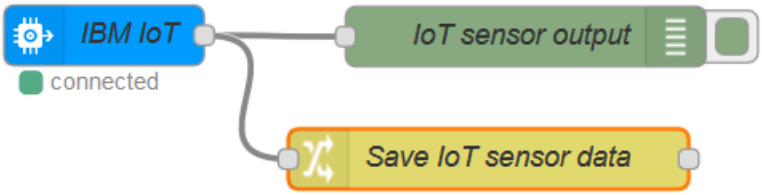
Note: if you are using the SensorTag data, use **payload.d.light** rather than payload.d.temp.

30. Click the **Done** button.

31. Connect the output of the ibmiot node to the input of the change node.

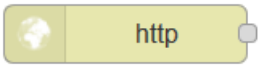
32. Click the **Deploy** button to deploy the flow to the server.

33. The flow should look as follows:



3.2.3 Create an HTTP request/response for external application

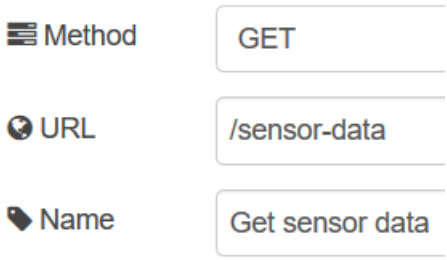
At this point we have saved the sensor data. To allow external applications access to the data, we need to create an http request/response to allow external access. Later in the demo, our mobile application will access the data via the http request.

1. Find the **http** node in the palette under the input section (don't use the http request node, there is a distinction): 

2. Drag-and-drop the http node into the flow workspace (drop it under the existing flow).

3. Double-click the http node.

4. Configure the node as follows:



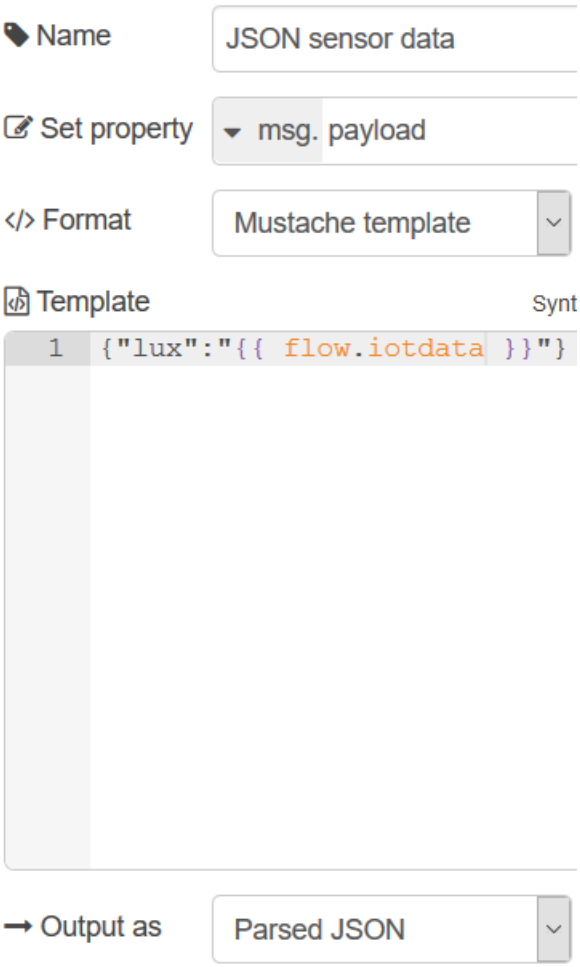
5. Click the **Done** button.

6. **Find** the **template** node in the palette under the **function** section:


7. Drag-and-drop the template node into the flow workspace next to the http node.

8. Double-click the template node.

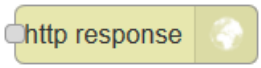
9. Configure the node as follows (copy/paste **{"lux":"{{ flow.iotdata }}"}** in to the template body), and note that we will use 'lux' regardless of whether or not you use the SensorTag light/lux data, or the simulated IoT data:



10. Make sure that the 'Output as' is listed as **Parsed JSON**, and not plain text. Click the **Done** button.

11. Connect the output of the http node to the input of the template node.

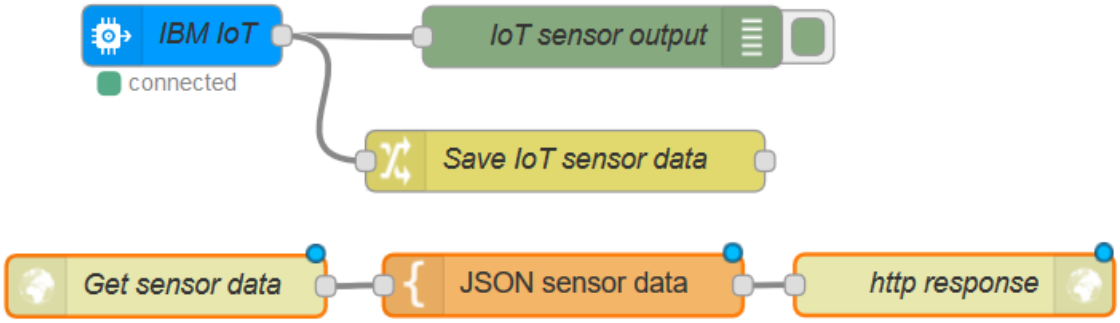
12. Find the **http response** node in the palette under the **output** section:



13. Drag-and-drop the http response node into the flow workspace next to the template node.

14. Double-click the http response node.

15. Simply add **http response** in the **Name** field.
16. Click the **Done** button.
17. Connect the output of the template node to the input of the http response node.
18. Click the **Deploy** button to deploy the flow to the server.
19. The entire flow should look as follows:



3.2.4 Test the HTTP request/response

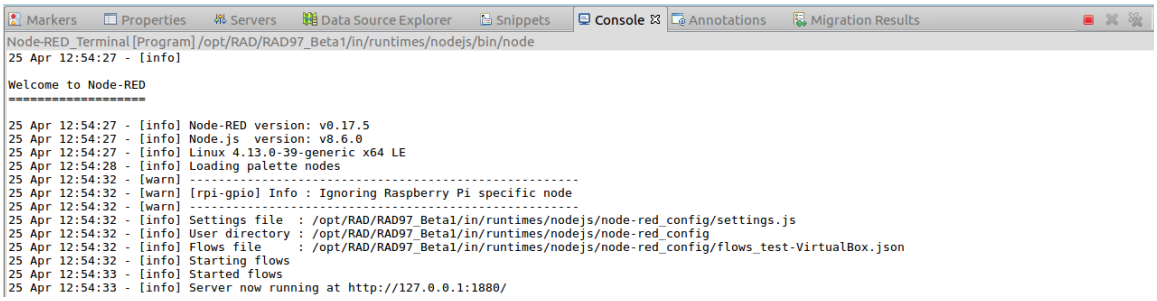
1. Go to a browser and enter the URL: http://localhost:1880/sensor-data
2. The most current sensor data should be returned in JSON format.

3.2.5 An aside: Open the RAD Node-RED Explorer.

RAD offers a unique view of your Node-RED flows and nodes in the form of an explorer. For those familiar with Eclipse or RAD, there are many different types of explorers which allow you to view files in your application. While we won't make much use of the Node-RED Explorer for this lab, we would like to quick point out this feature. To open the Node-RED Explorer, go to **Window > Show View > Other… > Internet of Things > Node-RED Explorer**. This should open the Node-RED Explorer in the upper left-hand corner of RAD (relative to the Enterprise Explorer). Notice that it shows the two flows you created (you might need to select the refresh button: , and you'll need to refresh when you deploy changes). You can expand a flow and see the nodes within it. You can double click a flow or node and be taken to that flow in the Node-RED Editor view. While our number of flows and nodes are relatively small in this lab, the explorer comes in handy when dealing with lots of flows and nodes.

3.2.6 Another aside: Node-RED terminal and RAD Node-RED Preferences.

You might be wondering how you can start and stop the Node-RED server. You might have noticed that the Node-RED console is below the Node-RED editor, as can be seen here:



RAD has integrated a terminal which starts the Node-RED server and shows its output. To stop the Node-RED server you can push the red button in the upper right-hand corner as seen in the screen cap above.
To start the Node-RED server, as well as perform some other Node-RED configuration, open the Node-RED Preferences. To do so go **Window > Preferences > Node-RED Platforms.** You will also notice that you can add URLs to any other Node-RED server. Those URLs will be populated in the Node-RED Editor view, allowing you to easily change to any number of Node-RED servers and edit them.

3.3 Building the mobile application

In this section, you will build a mobile application from a Git repository. This application will receive data from Node-RED and adjust the colors and brightness of the application based on that.

3.3.1 Clone the mobile project

1. Go to **File** > **Import...**
2. Expand **Git**.
3. Select **Projects from Git**.
4. Click **Next**.
5. Select **Clone URI**, click **Next**.
6. Enter the URL: https://github.com/cesarcer/projects, click **Next**.
7. In the **Branch Selection** page, select **master**, click **Next**.

8. In **Local Destination** page keep the defaults, click **Next**.
9. In **Select a wizard to use for importing projects** page select **Import existing Eclipse projects**.
10. Click **Next**.
11. In **Import Projects** page, select **CognitiveApp**.
12. Click **Finish**.

3.3.2 View the Source Code

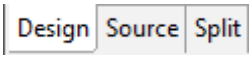
The application consists of 2 web pages, the main page and a configuration page, as well as a JavaScript file to dynamically change the color and brightness of the application. This application was developed using the **JQuery Mobile** framework. Let's take a brief and quick peek so you are familiar with the portion of the code which calls to your Node-RED flow.

1. In the **Enterprise Explorer** view, expand the **CognitiveApp** project (the view can be found in the upper left section of RAD, or can be opened via: **Window > Show View > Enterprise Explorer)**.
2. Expand the **WebContent** folder to find the **index.html** file, this is the entry point of the application.
3. Double click the **index.html** file, this will open the web page in the **Rich Page Editor**.
4. At the bottom left of the editor select **Source**, this will switch to source mode to have a better view of the application code:

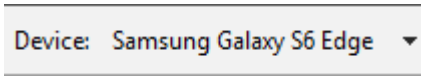
5. In line **89**, this is where the code calls to get the tone data.
6. Next, find line 111. This is where the code calls to get the sensor data.
7. Now double click the **settings.html** file to open it in the editor and switch to the **Source** tab.
8. Notice that at line **76** the code gets the tone data, and at line **93** it gets the sensor data.

Now let's switch to Design mode to have a quick view of how the application looks:

9. Open the **index.html** file.
10. At the bottom left of the editor select **Design**, this will switch to design mode to have a better view of the web page.



11. In the top right corner of the editor, next to **Device**, is a pull-down list (initially set to **Standard**). This will render the preview as if it was running on the selected device, e.g:



12. Now double click the **settings.html** file to open it in the editor, and repeat the previous two steps

3.4 Testing

3.4.1 Test the application locally

Now that we have the project in the workspace we can first test it locally to make sure it is working as expected, testing locally helps to reduce deployment times. We will deploy the application to the **Web Preview Server** (this server is part of Rational Application Developer and you can use it to quickly deploy and test your applications):

1. Right click the **index.html** file > **Run As** > **Run on Server**.
2. Expand the **127.0.0.1** folder and select **Web Preview Server**, click **Next**.
3. Keep the default on this page and click **Finish**.

This will launch the application in the **Internal Web Browser**. Using the local **Web Preview Server,** you can easily make changes in the source code and just refresh the browser to quickly test it without waiting for the application to be deployed to a remote server.

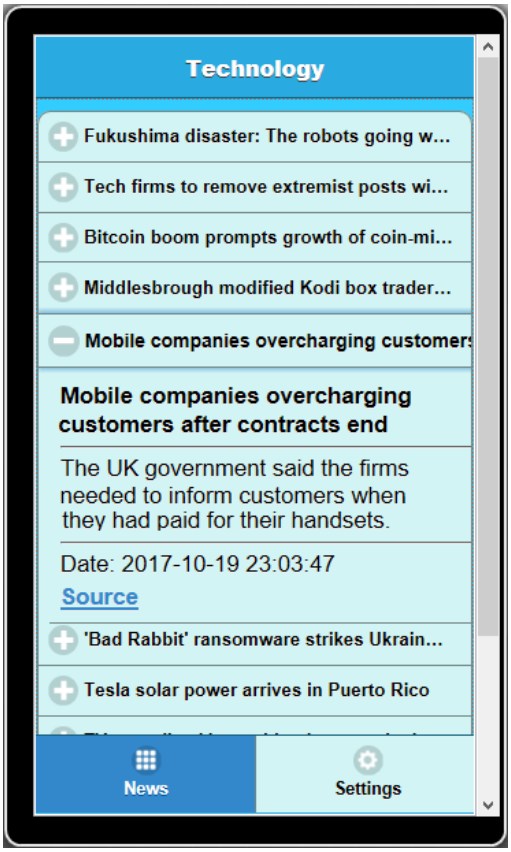
4. Use the buttons at the bottom of the web page to switch to the **Setting** page:

5. Use the drop-down menu to select a category of news:

Notice that in the **Settings** page, you can change how often the application is refreshed, default is 10 seconds.

6. Use the buttons at the bottom of the web page to switch to the **News** page, now it will load news based on the selection you made:

7. Click on the **+** icon on any news to expand it to see more details as well as a link to the source:



8. Access your Twitter account and post a tweet, the colors of the application will be automatically adjusted to match the tone of your post. The **Watson Tone Analyzer** service can recognize 5 tones as follows: anger, disgust, fear, joy and sadness; depending on the tone of your tweets. One of the following colors will be automatically used:

Tone	Color
Anger	Blue
Disgust	Brown
Fear	Orange
Joy	Yellow
Sadness	Red

Next, we can test the IoT integration. If you are using the SensorTag provided by the instructors, the instructor should cover the device to prevent light from reaching the sensor. Doing so, you will see how the brightness of the application is automatically adjusted. Do this gradually to allow more or less light from reaching the sensor and the

brightness will be adjusted accordingly. If you are using the IoT device simulator, adjust the data up and down to see the brightness change accordingly.

3.4.2 Test the application remotely (TIME PERMITTING!!)

At this point in the lab, we have done everything locally. We have used the Node-RED server integrated in RAD, and the Node-RED Editor view, to create flows. We have used RAD's git integration to clone a mobile app. The app is coded to access Node-RED locally. Finally, we have used a locally running server to test our mobile app (and in turn the mobile app can access the Node-RED tone and sensor data). This greatly aids development. However, obviously we would like to deploy our mobile app remotely in order to make available to the world and make millions…..or maybe just impress your friends or family with the results of the lab. In the last portion of the lab we will do just that (well, I can't promise you'll make millions….or impress anyone). We will deploy our app to BlueMix. In so doing, you can access your app via your cell phone or web browser. However, the difficult part in a lab like this is allowing remote access to the Node-RED tone and sensor data running on your VM given we don't know the details of the lab network or the details of each student's machine. That is, if the mobile app is running on BlueMix, calls to localhost to get the tone and sensor data is not going to work. We could update the mobile app code to call the IP of your machine, and set up 'port forwarding' to the VM. However, this is far too complex and time consuming for this lab. Therefore, we are going to cheat. In the next steps we are going to update the mobile app to call to a Node-RED server on BlueMix which is running the flows we have all created in this lab. You can find it here:

http://node-red-bluemix-starter-20170711135923629.mybluemix.net/red

The instructor owns this account. We will update the mobile app code to get /tone-data and /sensor-data from this Node-RED server. The instructor will add new tweets and manipulate the SensorTag to provide changes to the app. Note that in 'Appendix A' below the steps are given to create your own Node-RED server on BlueMix, this is an exercise you can perform after the lab.

3.4.2.1 Edit the application source code

In section 3.3.2, we described where the URLs/GETs are located to the /tone-data and /sensor-data. The URLs contain local host. Visit that section again, and replace **localhost:1880** with **node-red-bluemix-starter-20170711135923629.mybluemix.net**. An easy way to do this is to open index.html, then do a Ctrl+F and enter the two strings, and hit **Replace All**. Do the same for settings.html. After saving the two files, the changes should be deployed to the local server where you can test again.

3.4.3 Create a Bluemix server to deploy the mobile application

1. In the RAD view section (lower portion of RAD) find the Servers tab and select it:


2. In the Servers section, right click the **Servers** view > **New** > **Server**.
3. From the list of servers, select **IBM Bluemix**, click **Next**.
4. In the **Email** and **Password** field, enter your Bluemix credentials, click **Next**.
5. Select the **dev** space and click **Next**.
6. Keep the defaults on this page and click **Finish**.

3.4.3.1 Deploy the application to Bluemix

With the application working as expected when running locally, we can now deploy to the cloud.

1. Right click the **index.html** file > **Run As** > **Run on Server**.
2. Expand the **Cloud** folder and select **IBM Bluemix**, click **Next**.
3. Keep the default on this page and click **Finish**.
4. This will launch a wizard used to deploy applications to Bluemix.
5. In the **Name** field, you can change the name of the application, or take the default.
6. The **Subdomain** will be part of the URL of the application so it must be unique, in the **Subdomain** field add your name or a random number at the end, ie: CognitiveApp**79372**, we need to add this to avoid possible conflicts with other existing applications, click **Next**.
7. For this specific mobile application, we don't need any Bluemix services, so keep the defaults and click **Next**.
8. For this specific mobile application, we don't need any environment variable, so keep the defaults and click **Finish**.
9. Wait for the application to be deployed, the deployment time depends on the internet speed.

Once the application is deployed, the **Internal Web Browser** will be opened, but now the application is running in the cloud, meaning that it can be accessed from any device with a web browser. To test your application, you can open a web browser in your smartphone and enter the URL. You will likely need to use **http**, not https in the URL.

This concludes the lab!!!!

4. Appendix A – Creating a Node-RED boilerplate

4.1 Create a Node-RED boilerplate in BlueMix

This section will detail how to create a Node-RED boilerplace in BlueMix.

1. Click the following link to go to open BlueMix: https://console.bluemix.net/dashboard
2. Log into your BlueMix account.
3. Once logged in, select the **Catalog** link in the upper right-hand corner.
4. In the **Search** bar enter **Node-RED**.
5. In the results, select **Node-RED Starter**.
6. In the **Create a Cloud Foundry App**, enter an app name of **interopitx-nodered**.
7. In the **Host name field**, edit the name to create a unique host name. I would suggest we all use a host name in the form interopitx-nodered-<unique id>; where <unique-id> is something unique to your application's hostname.
8. Take the defaults for the rest of the page, and click the **Create** button.
9. Be patient, after hitting create the browser might spin for a minute while BlueMix creates the app.
10. Eventually a new screen should open indicating the app is starting. Have more patience.
11. Eventually the app will start and a "**Visit App URL**" will be available.
12. When available, click "**Visit App URL**".
13. In the welcome page, click **Next**.
14. In the **Secure your Node-RED editor**, select the **Not Recommended** option. For demo purposes, we don't care if our editor is unsecure.
15. Tick the box to confirm you want your editor to be insecure.
16. Click **Next**.
17. Click **Next** in the next page.
18. Click **Finish** in the next page.
19. After the settings have been applied, click **Go to your Node-RED editor.**
20. Take note of the URL of your Node-RED editor. Copy the portion up to #flow. In other words, the URL will look something like: https://interopitx-nodered-<unique id>.mybluemix.net/red/#flow/<flow number>

    Copy the portion https://interopitx-nodered-<uniqu id>.mybluemix.net/red
21. We will use this URL in the next steps.

4.2 Configure RAD to use the BlueMix Node-RED server.

At this time, the Node-RED Editor view is connected to the locally running Node-RED server. To add the BlueMix Node-RED server to RAD, do the following:

1. Go to **Window > Preferences > Node-RED Platform**.
2. Click the **Add…** button.
3. Enter **BlueMix** in the **Platform name** text box.
4. Paste the URL to your BlueMix Node-RED app (from the steps above) **URL** text box.
5. Click **OK**.
6. Select the **BlueMix** platform and click the **Set as Default** button.
7. Click the **Test Connection** to verify the connection.
8. Click the **Apply and Close** button.
9. In the Node-RED Editor view's pull-down box (currently populated with localhost:1880), you should be able to select the box and find the URL to the BlueMix Node-RED server. Select the URL and it should open the BlueMix Node-RED editor.

5. Appendix B – Possible Twitter Authentication issue on Ubuntu

This section offers an alternate way to authenticate with Twitter given that the **Click here to authenticate with Twitter** button doesn't work as expected from within the Node-RED editor in RAD on Ubuntu. After dragging and dropping the Twitter node into the editor workspace, do the following:

1. Click the **Deploy** button to deploy the flow to the server.
2. Select **Confirm deploy**.
3. Open FireFox and enter http://localhost:1880
4. Double click the Twitter node.
5. Click the pencil icon: 
6. Click the **Click here to authenticate with Twitter** button.
7. In the resultant page, log in to Twitter and click the **Authorize app** button. A message should be displayed indicating that authorization was successful and directing you to close the browser windows. Close the browser window to return to the Node-RED editor.
8. Your Twitter id should be filled in. Click **Add**.

9. Click **Done**.
10. Click **Deploy** and close FireFox.
11. Return to the RAD Node-RED Editor view.
12. Select **Review Changes**, then **Merge.**
13. Select **Deploy**.
14. Return to the section Configure the Twitter node and the steps immediately after authenticating with the Twitter node (i.e. return to the steps detailing the **Search** terms in the Twitter node).