Water Usage Sensor

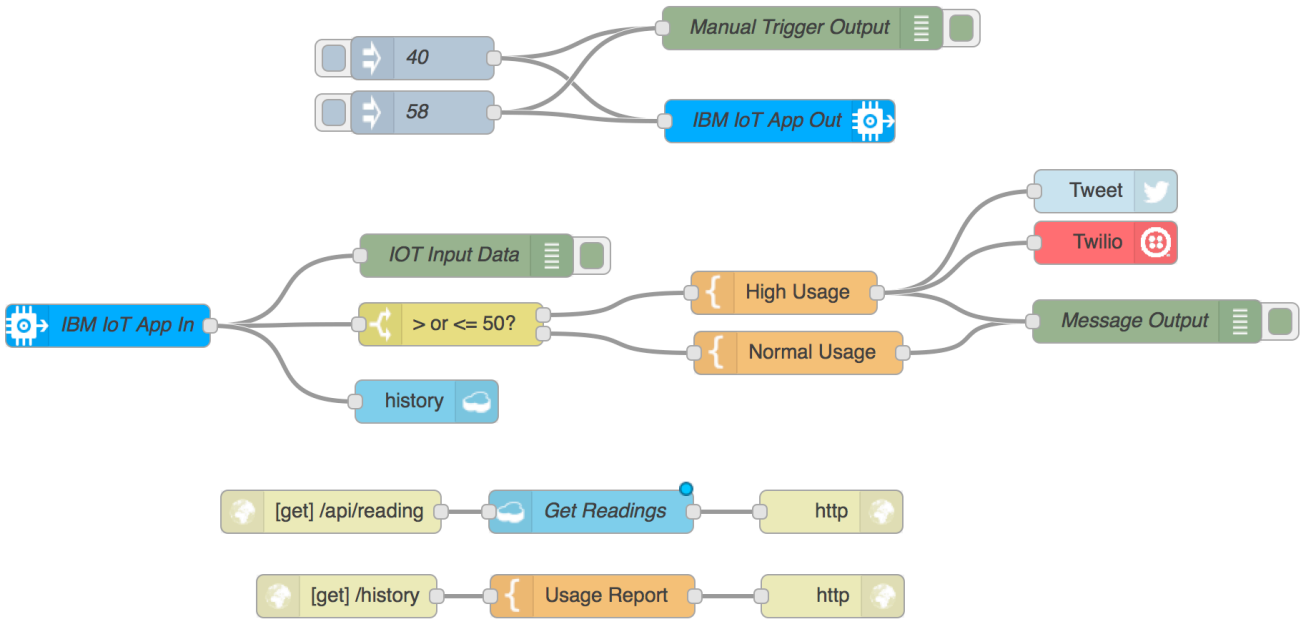
Hands-On Lab

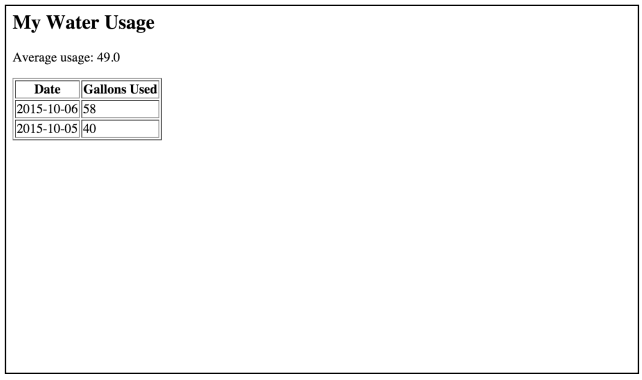
Author: JeanCarl Bisson | jbisson@us.ibm.com | @dothewww

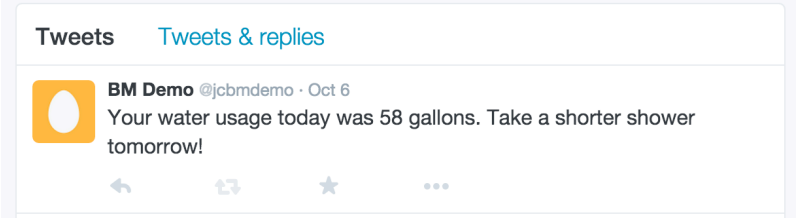
Table of Contents

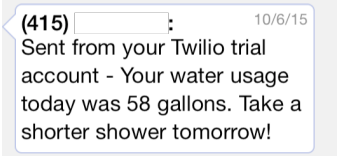
Topics covered: Node-RED, Internet of Things, Twitter, Twilio, Cloudant NoSQL database, AngularJS

This lab shows how to set up a Node-RED starter application and add the Internet of Things Service to simulate a water usage sensor. Using the simulated water usage data, the Node-RED application will responds to values that exceed a specified threshold and send a tweet using Twitter and a SMS message using Twilio. The usage values are stored in a Cloudant NoSQL database. These values can then be retrieved from the Cloudant NoSQL database for use in a webpage. Finally, a webpage is created displaying historical water usage reported by the sensor, written in HTML, JavaScript and AngularJS.







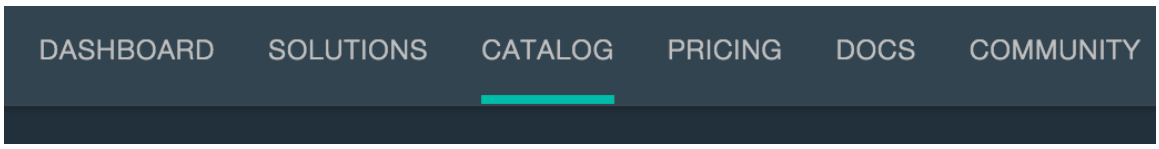


Create a Node-RED application

Node-RED is an open-source project that makes it easy to program control logic using graphical blocks called nodes. Each node has a defined functionality, optionally with input and outputs connectors. In this section, we'll create a Node-RED application in IBM Bluemix.

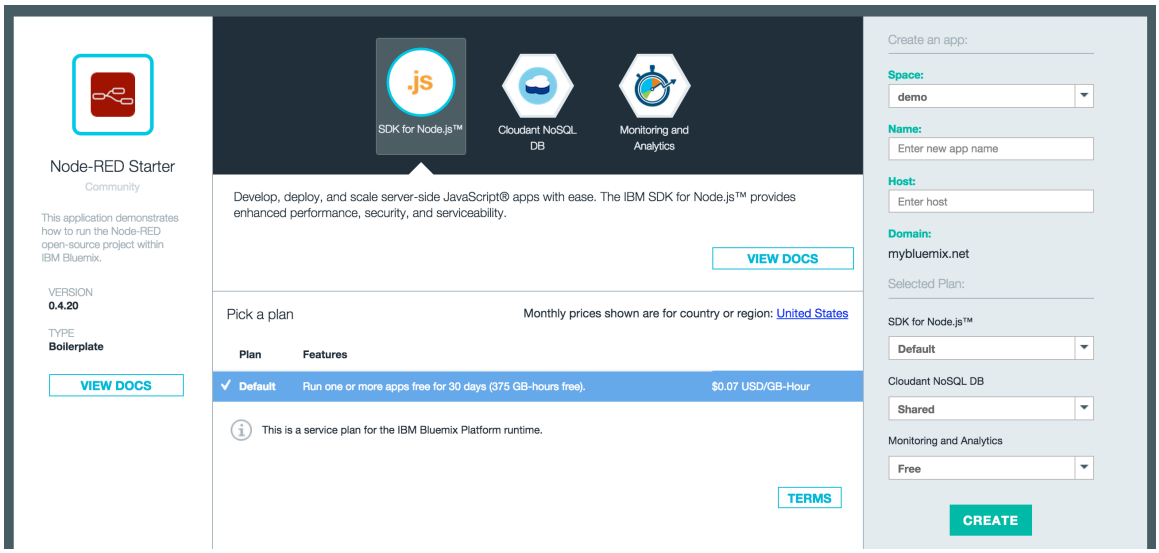
1. To get started, sign up for a new account or log into your IBM Bluemix dashboard at `https://bluemix.net`.

2. Once logged in, click on the **Catalog** link at the top of the page.

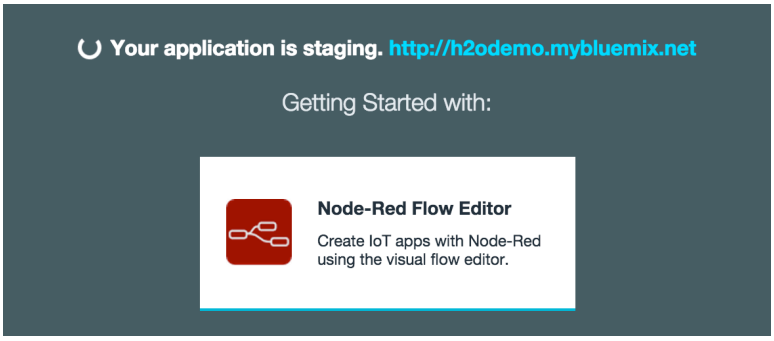


3. Click on the **Node-RED Starter** tile.

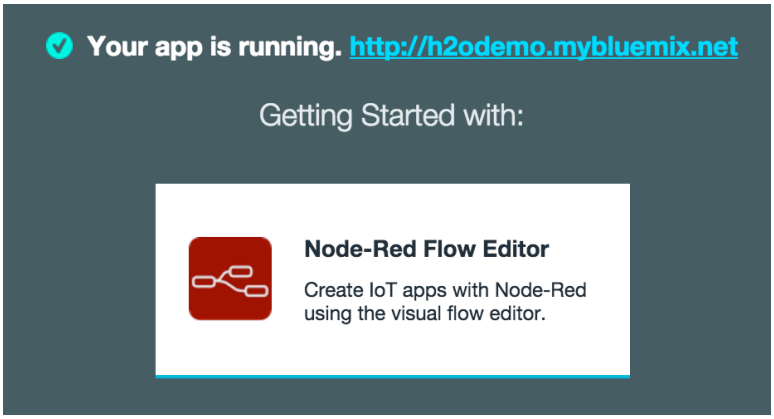
4. This is the configuration screen for this boilerplate. Enter an application name and a host name for your application. The name is used in the IBM Bluemix dashboard for your convenience. The host is used in the application's URL. If you choose *`something`* as your host, your application will be accessible at `http://something.mybluemix.net`. If another user has already registered an application with the host *`something`*, you will be prompted to try another name. Click on **Create**.



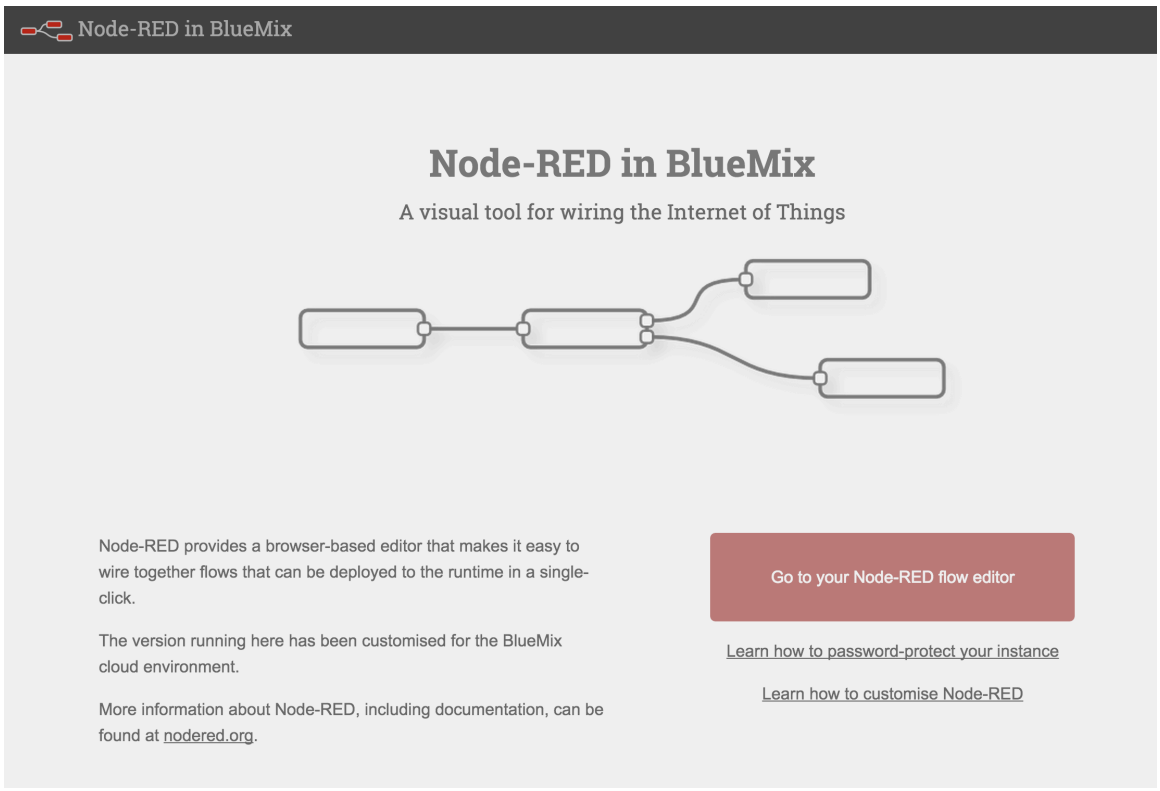
IBM Bluemix will start up a Node.js instance and attach a Cloudant NoSQL Database Service to this instance. It can take up to a several minutes for this process to complete.



5. When the setup is complete, the URL of your application should become hyperlinked and underlined. Click on the URL.



6. This is the default homepage for the Node-RED application. Click on the red button labeled **Go to your Node-RED flow editor**.



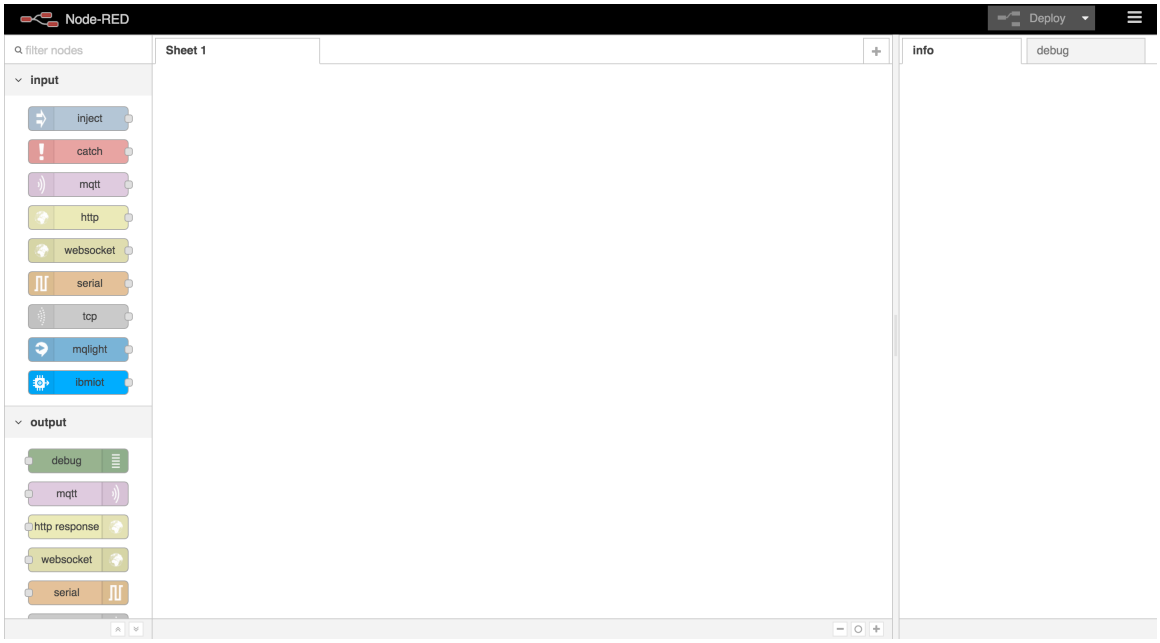
7. You will spend most of your time on this screen building out logic that runs when events occur, either triggered manually or automatically. There are several pieces to note about this interface. On the left-hand sidebar is a list of nodes. I like to think of this as a bin of parts. They are categorized with labels. Input nodes usually began a flow. Output nodes usually end a flow. Other nodes may have both input and output connectors, receiving input, doing something, and then outputting a result. Other categories include Function, Storage, Social and IBM Watson. With this set of nodes, you can build complex flows. You can find more nodes at `http://nodered.org` or write your own (outside of the scope of this lab).

   The middle portion of this screen is called a sheet. A sheet is a space that contains nodes, connected via a line. You can have multiple sheets to organize all your flows. Each flow progresses from left to right, input on the left, output on the right side. You can add as many nodes and flows to this sheet as you please. You can also connect separate flows together, reusing a part or all of a flow.

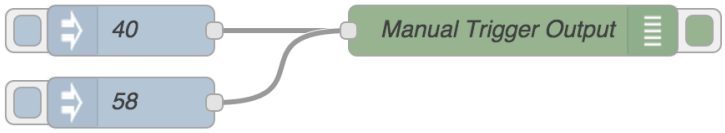
   On the right-hand sidebar are two tabs. When a node is selected, the **info** tab provides documentation about how to use the node. You can find out what a node expects as an input, what it does, and what it will output. The **debug** tab is useful when you need to inspect the output of a node. Connect a 

node to display the output of the node it is connected to.

   Finally, each time you make a change, make sure to click on the **Deploy** button in the top right corner. It is common to forget to deploy a change.

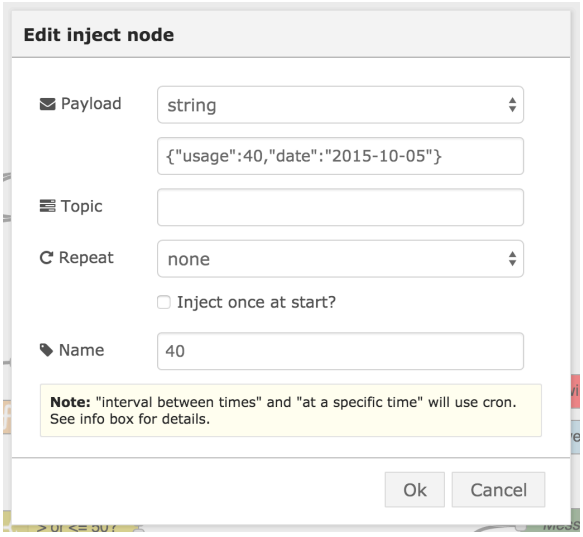


8. Let's create three nodes, which will look like this:



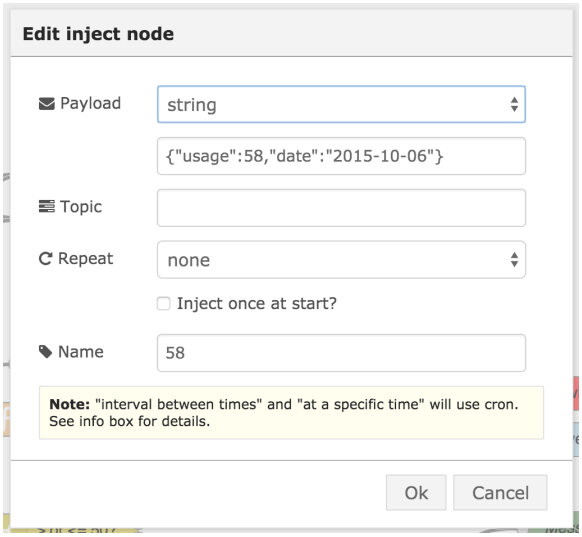
Add a 

node. Double click and use the following settings:



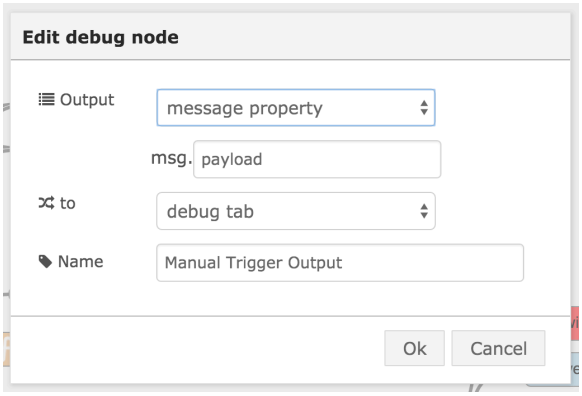
9. Add another 

node. Double click and use the following settings:



10. Add a 

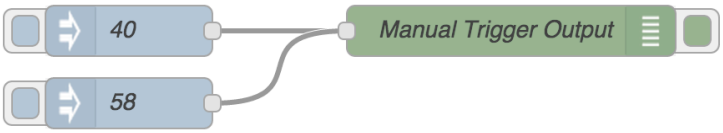
node. Double click and use the following settings:



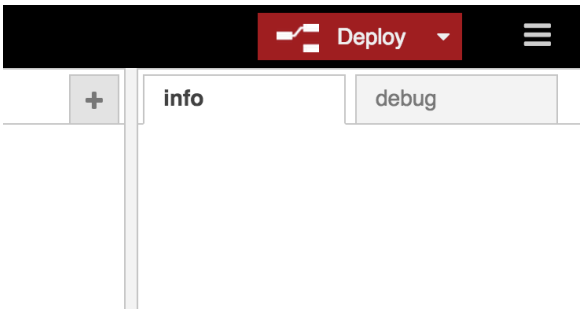
11. Connect the two 

nodes into the 

node as follows:



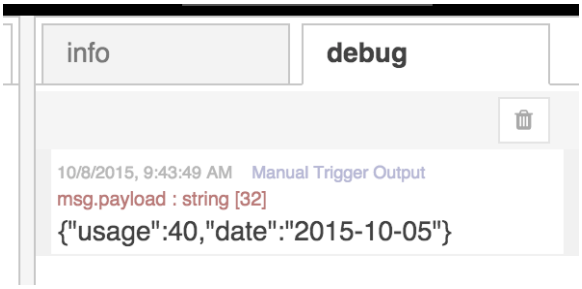
12. Click on the **Deploy** button at the top of the screen to save the changes.



This deploys your new flow. Each time you make a change, click on the **Deploy** button to make them active.

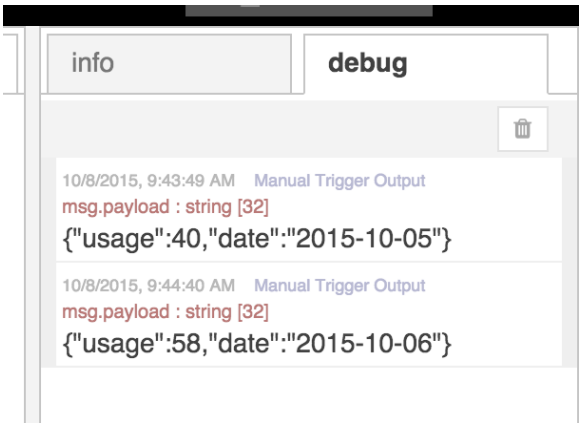
13. Click on the **debug** tab. This is where the debug node outputs data. Click on the nob on the left side of the 

node. The message payload is sent to the debug node, which outputs it in the debug tab.



14. Click on the nob on the left side of the 

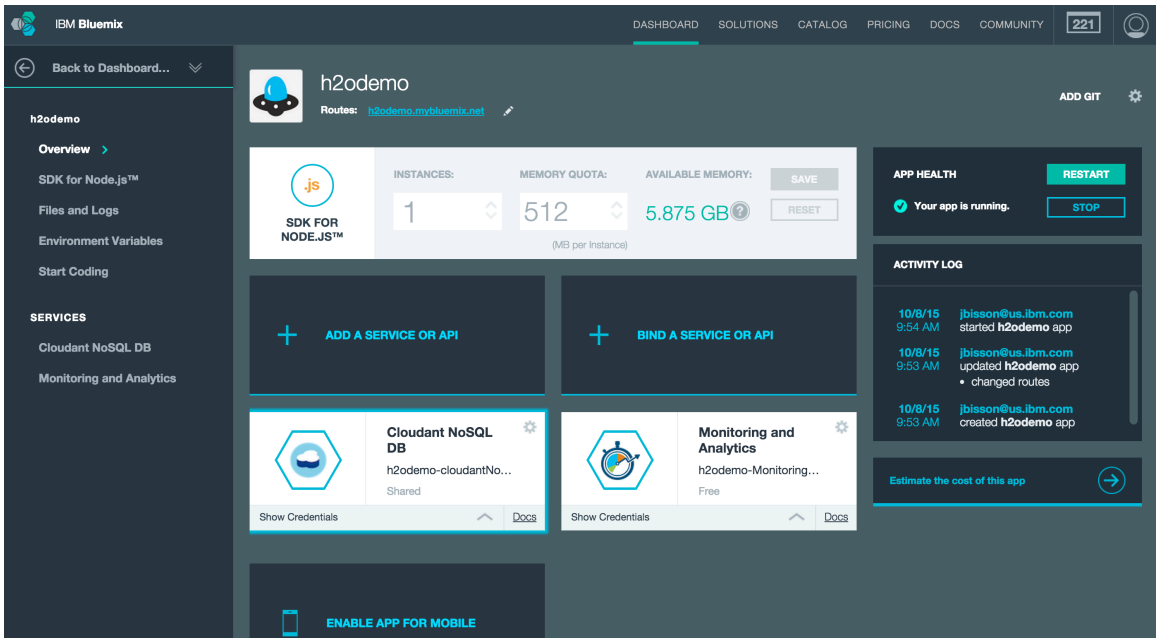
node. The message payload for that node is sent to the debug node, which outputs it in the debug tab.



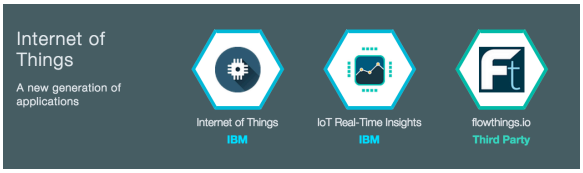
Add Internet of Things Service

The IBM Internet of Things service makes it easy for devices to connect to the service and submit device events with optional event data. A device can be anything that you want to connect to the internet. All the device needs is internet connectivity and a way to send the IoT service the event data. For this lab, we're going to create a water usage sensor. This sensor would theoretically submit water usage each day. The application could monitor how much water is used in a day and perform actions if the usage exceeds a threshold.

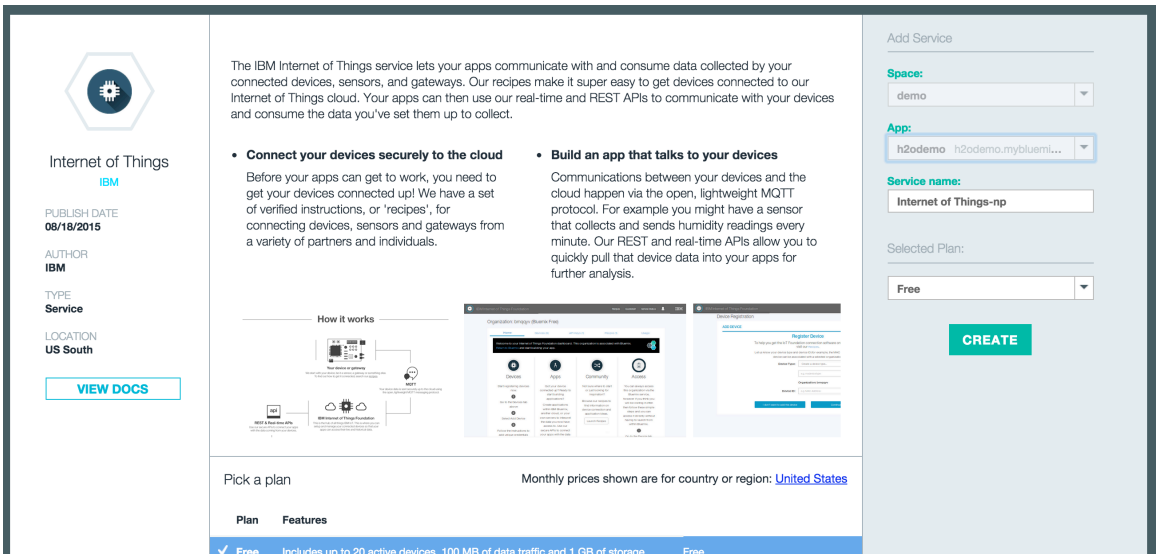
1. Return to application overview in the IBM Bluemix dashboard. Click on **Add a Service or API**.



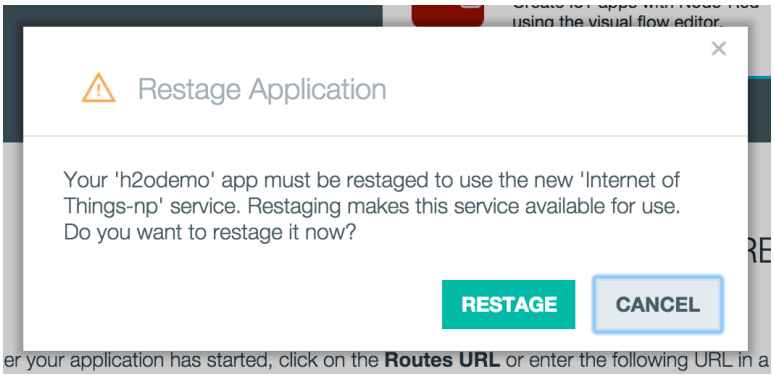
2. Select the **Internet of Things** service tile.



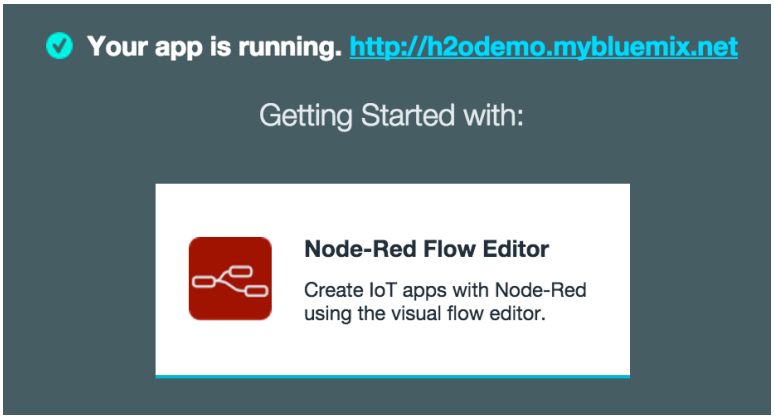
3. You can change the service name or leave it as the default. Click on **Create** to add the new service.



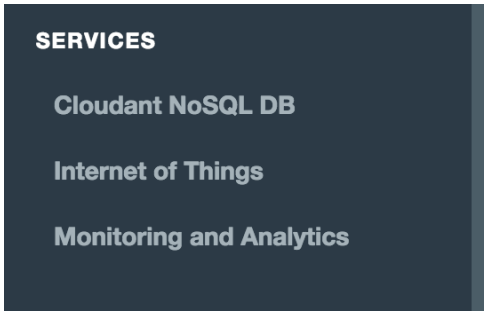
4. IBM Bluemix will ask to restage the application. Click on **Restage**.



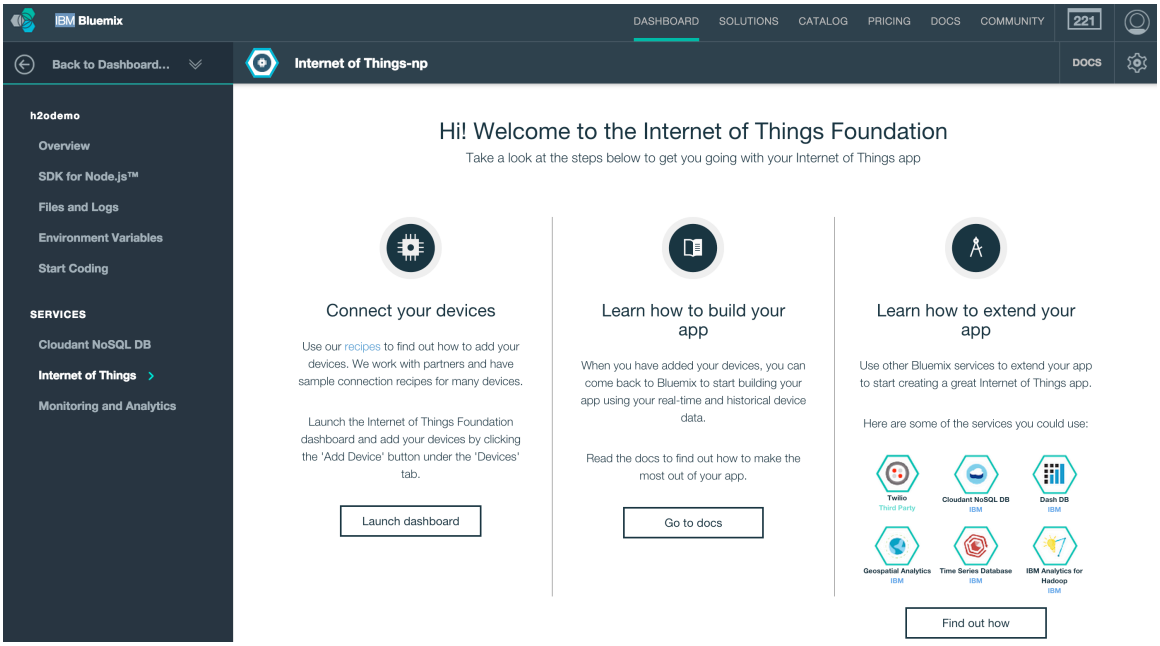
Restaging may take up to several minutes. When it is finished, the status will change to "Your app is running."



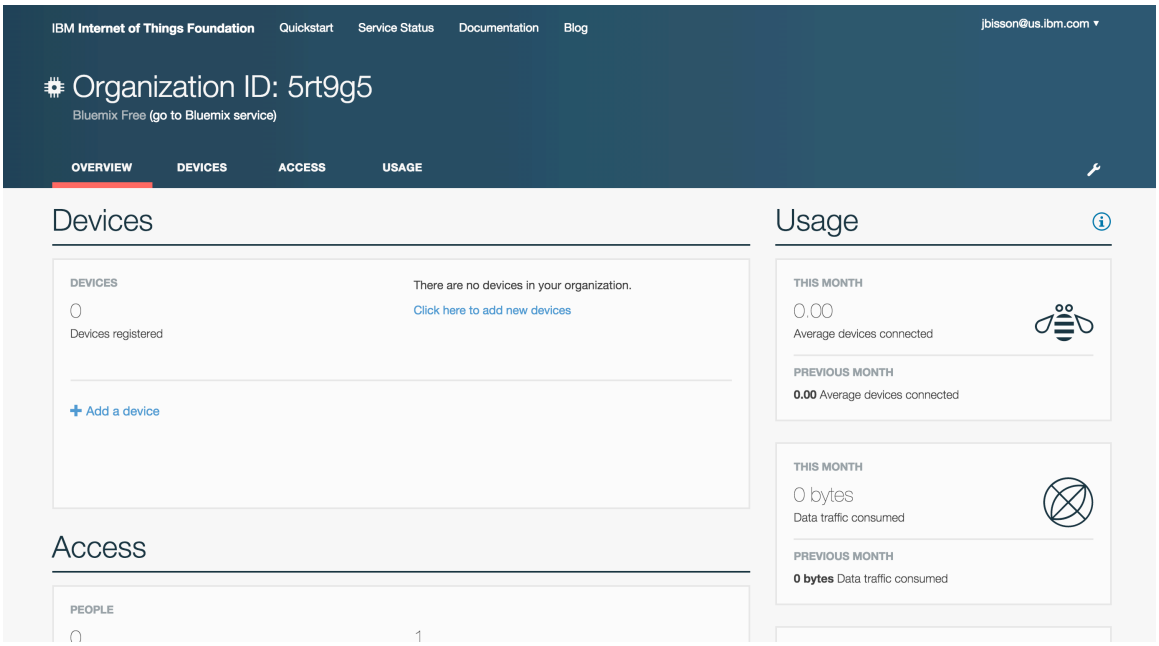
5. Click on **Internet of Things** in the left sidebar under **Services**.



6. Click on **Launch Dashboard** in the left column.

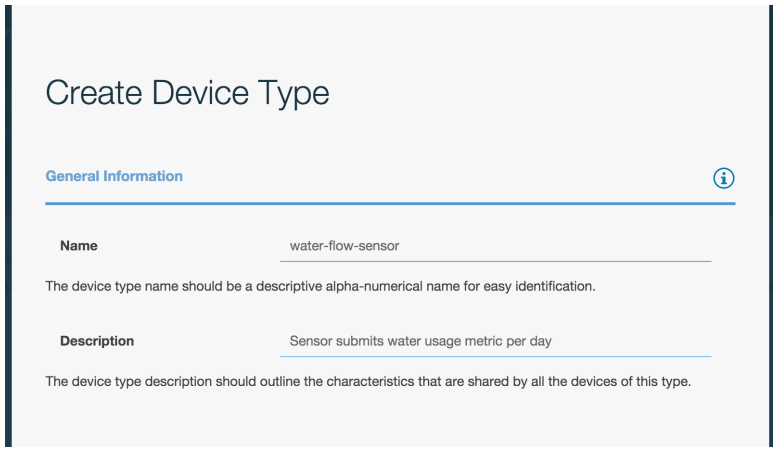


7. We need to create a device type for our simulated water usage sensor. Click on **Add a device**.

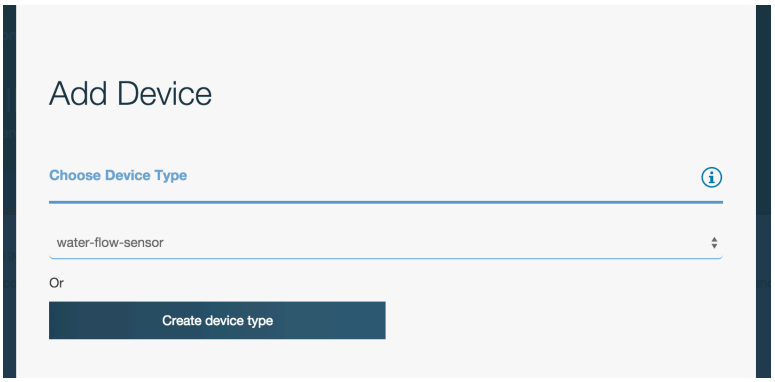


8. Since we don’t have a device type yet, the drop-down menu is empty. Click on **Create device type**.

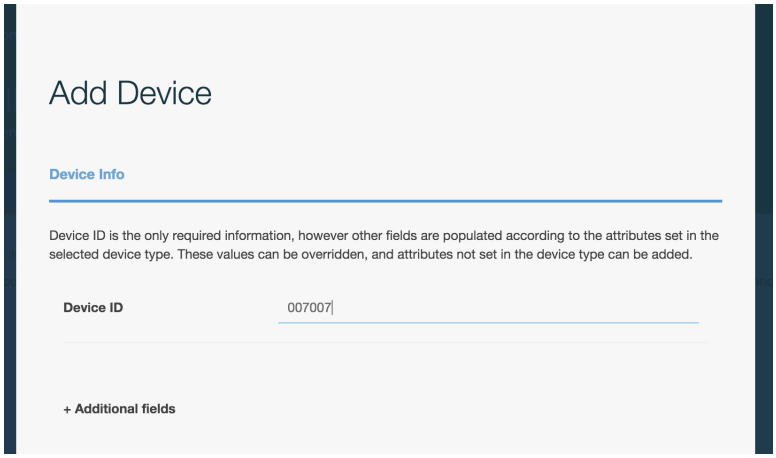
9. Fill out the next screen with the following:



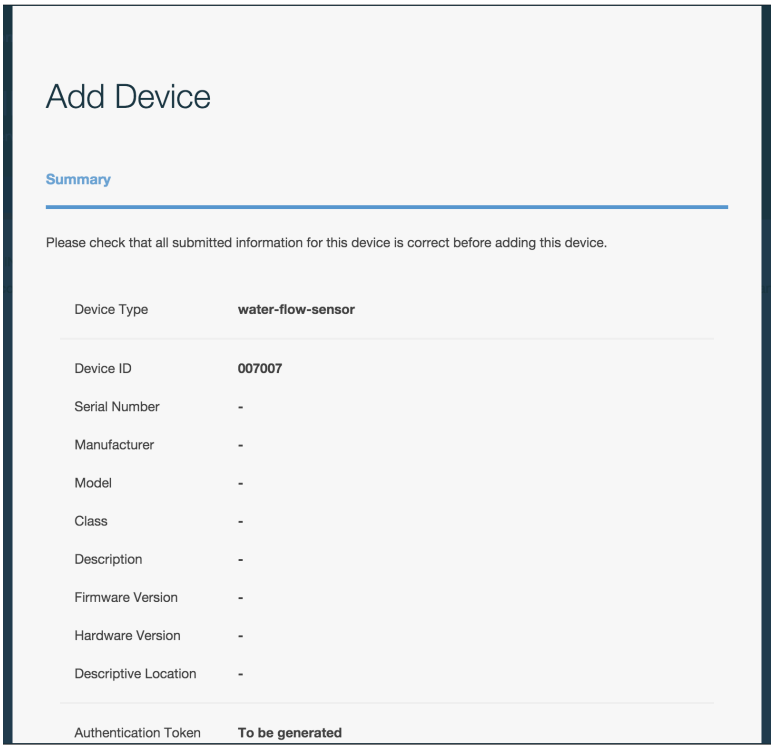
10. Skip through the next couple of screens, and then click on **Create**. Our new device has been added to the drop-down menu. Click on **Next**.



11. Enter a Device ID. This can be any combination of letter and numbers. This will help us identify a particular instance of this sensor. We'll choose **007007**.



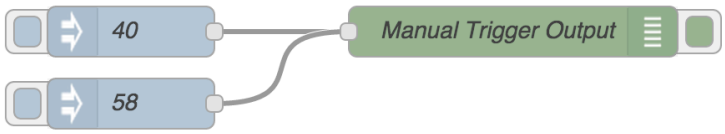
12. Click **Next** to skip through the screens until you see the **Summary**. Click on **Add**.

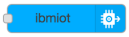


13. Take note of the device credentials. You will need them later.

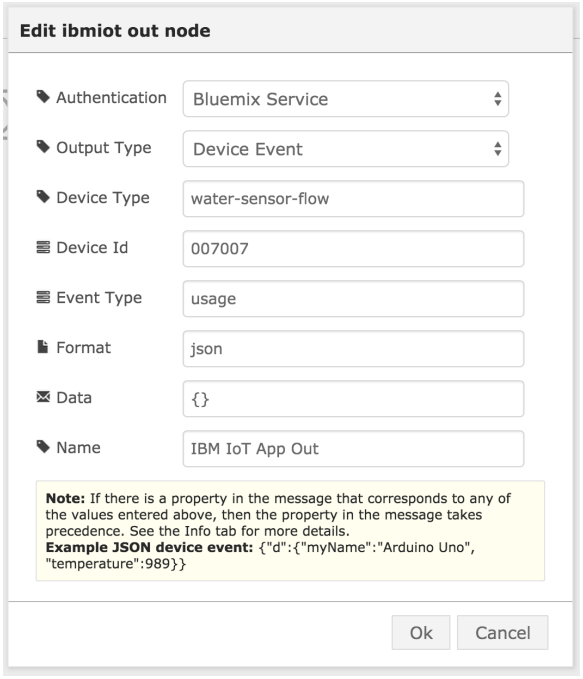
Connect to Internet of Things Service

Now that we have a device registered to the Internet of Things service, let's modify the flow we created earlier (shown below) to submit this water usage metric to the cloud.

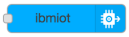


1. Add a 

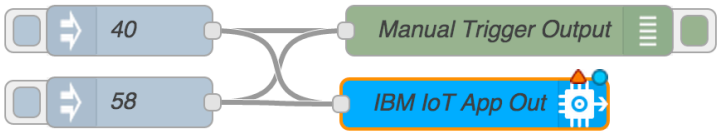
node under the debug node with the following settings:



2. Connect the two 

nodes to the 

node as follows. Click **Deploy** to save the changes.



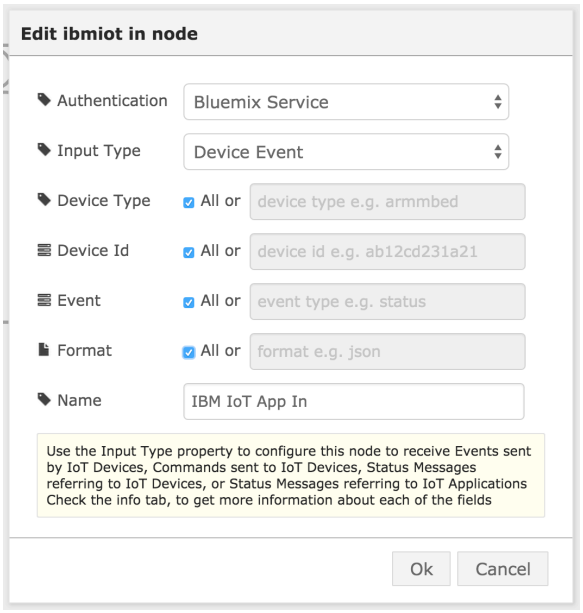
Each time either inject node is triggered, the IBM Internet of Things service will be sent the usage data as an event.

Receive Internet of Things Events

In the last step we were able to emit events to our Internet of Things service. Now that we have a simulated water usage sensor emitting our water usage at the end of the day, let's build another flow to handle this data.

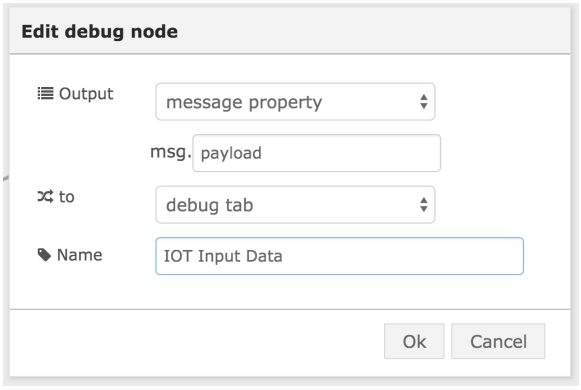
1. Add a 

node with the following settings:



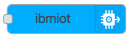
2. Add a 

node with the following settings:



3. Connect the two nodes together and click **Deploy** to save the changes.

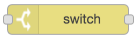
4. Click on the nob on the 

node. In the debug window, we see the first message is from the first debug node. The second debug node comes from our new 

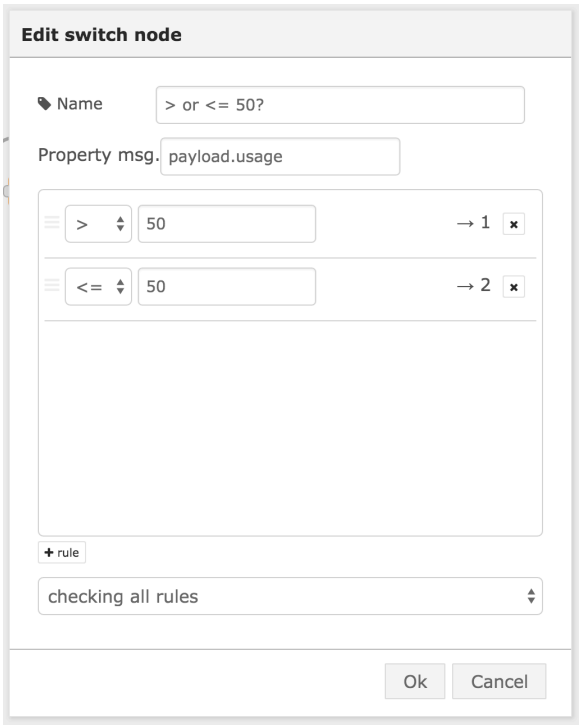
input node.

5. Let's add a node that makes a decision of which flow to continue processing based on the input:

- If the water usage is greater than 50 gallons per day, the first flow will be processed, representing the case when too much water has been used.
- If the water usage is less than or equal to 50 gallons per day, the second flow will be processed, representing the case when no more than the average amount of water has been used.

6. Add a 

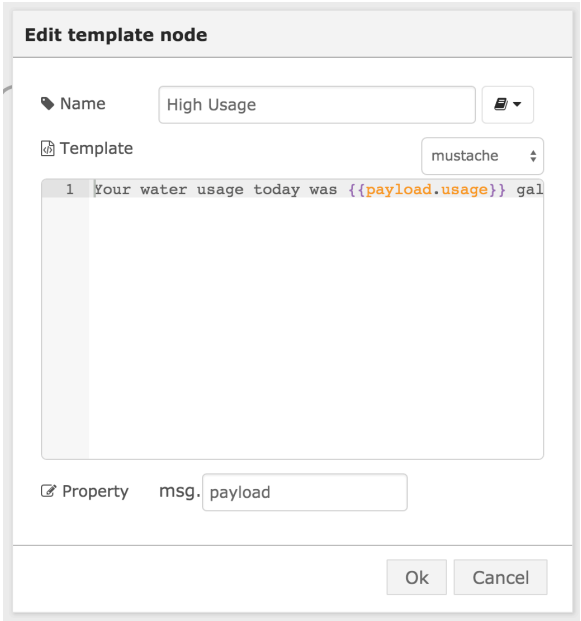
node with the following settings:



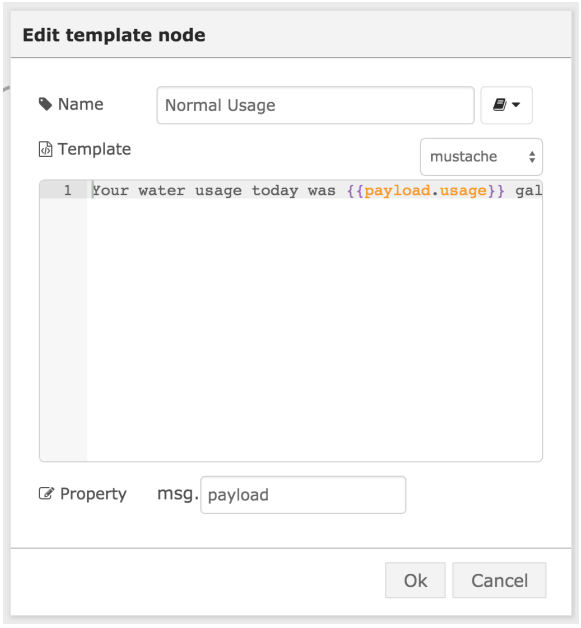
7. Create two 

nodes with the following settings:

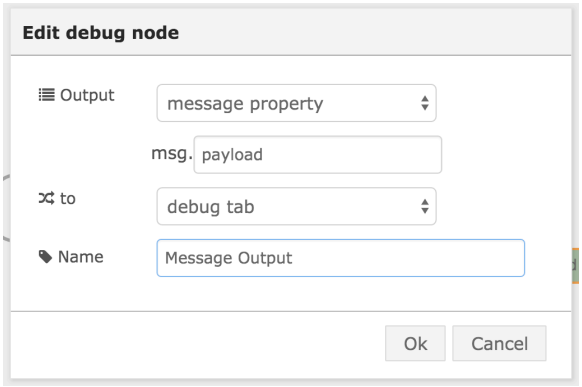
```
Your water usage today was {{payload.usage}} gallons. Take a shorter shower tomorrow!
```



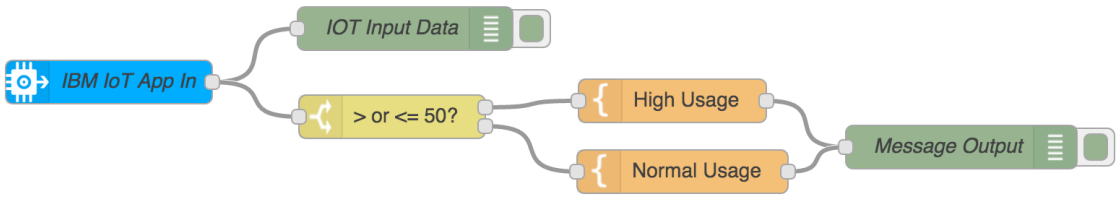

```
Your water usage today was {{payload.usage}} gallons. Keep it up!
```



8. Add a  node with the following settings:

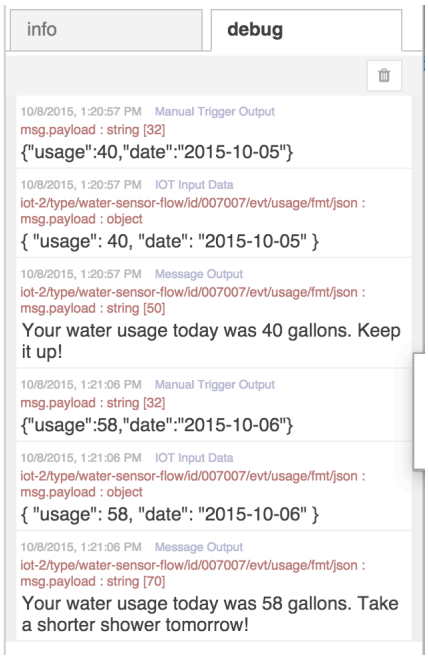


9. Connect the nodes together as follows:



10. Click on **Deploy** to save the changes.

11. Click on the nob next to the  and  nodes to trigger those flows. The template messages are displayed in the debug window.



The application is now connected to the Internet of Things service. It can emit device events with water usage, and can receive device events and process different messages based on the amount of water being used.

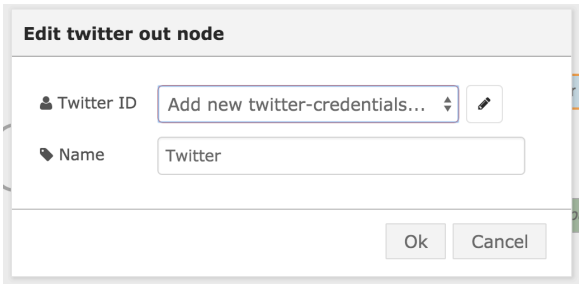
Connect to Twitter and Tweet Usage

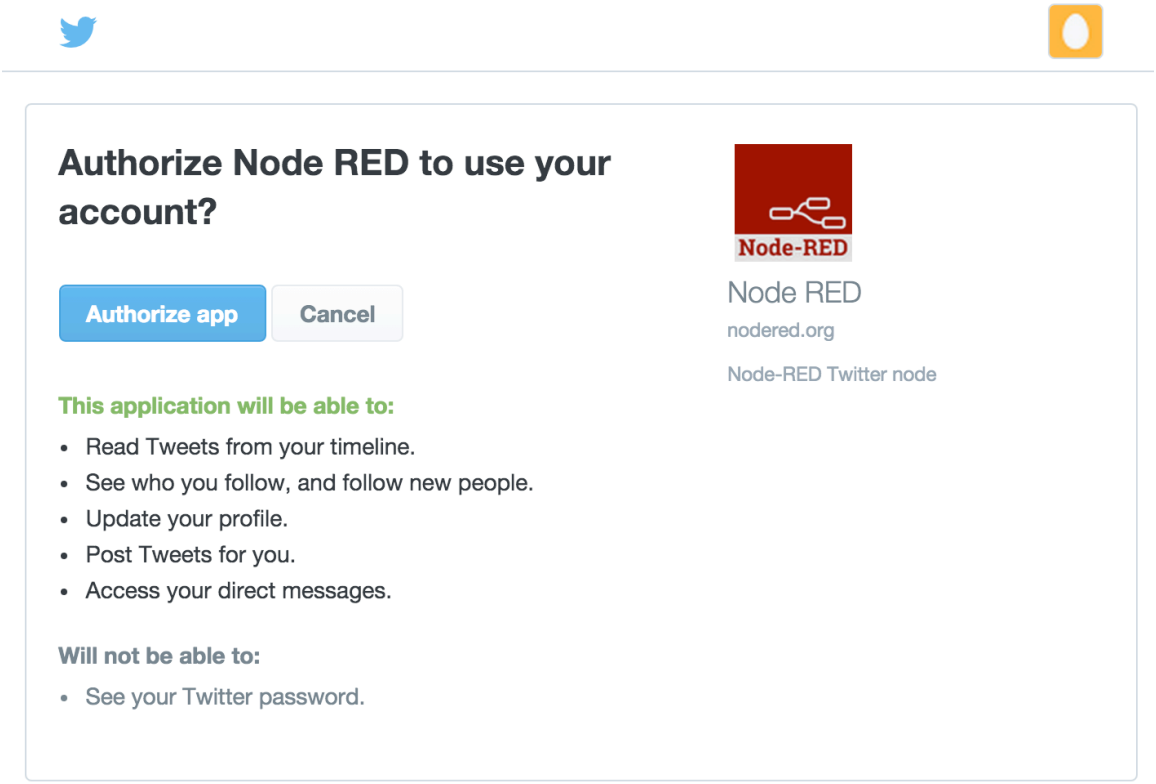
This section connects to a Twitter account and uses the Twitter account to tweet a notification when the water usage exceeds a specified amount (in this lab, 50 gallons of water). This section is optional and may be skipped.

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

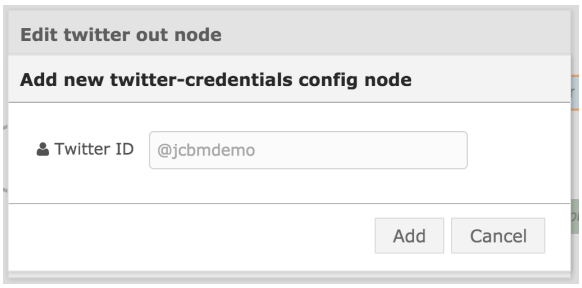
2. Add a 

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

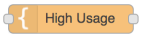




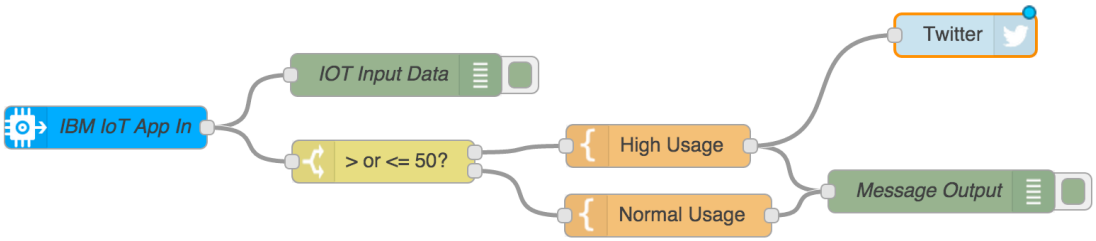
Return back to the node configuration. The settings for the Twitter node should have your username set.



3. Connect the 

node to the 

node as follows:

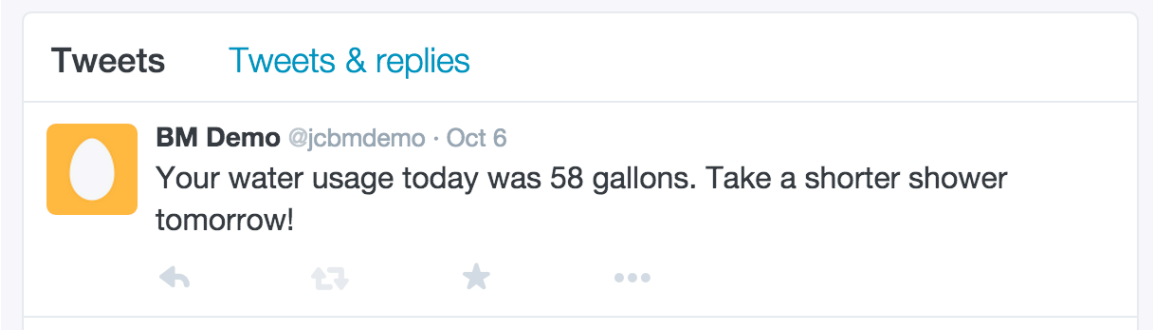


4. Click on **Deploy** to save the changes.

5. Click on the nob on the 

node. Since the water usage exceeds the 50 gallons, the first flow of the switch statement is triggered. The High Usage node constructs the message and is passed to the Twitter node. The Twitter node uses this message as the content for the tweet.

6. Visit the Twitter timeline for the user and verify the message has been tweeted.



Note: clicking on the 

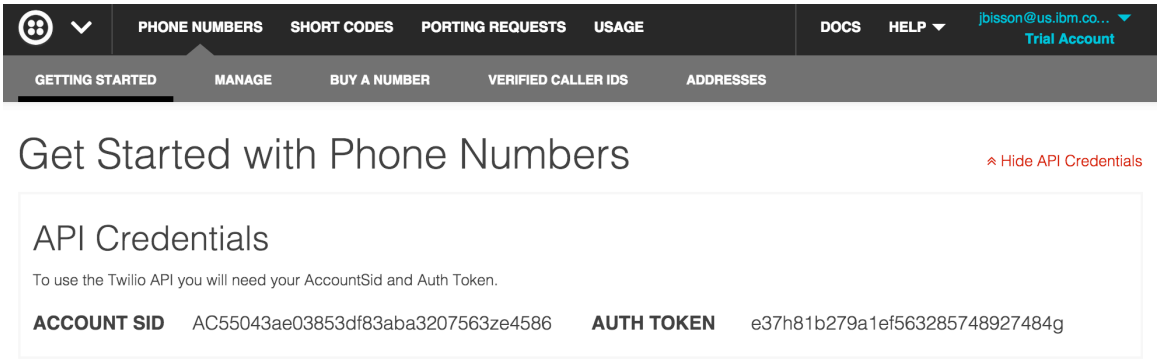
again may result in an error being shown in the debug tab and the tweet not being added to the timeline. This is due to a constraint where Twitter may reject successive tweets that are the same. Delete the previous tweet and try again.

Connect to Twilio and Text Usage

This section connects a Twilio phone number to our application and sends a text message notification when the water usage exceeds a specified amount (in this lab, 50 gallons of water). This section is optional and may be skipped.

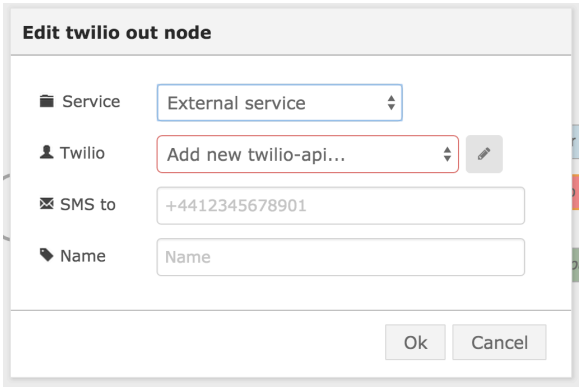
1. Sign up for a Twilio account at `http://twilio.com`. If you already have a Twilio account, sign in.

2. Go to the **Getting Started** tab, and click on **Show Credentials**.

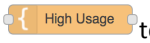


3. Add a 

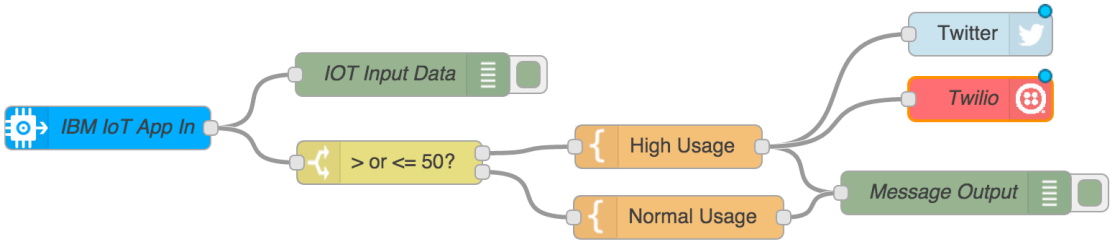
node. Double click on the node to edit the configuration. Click on the Pencil to provide your **Account SID** and **Auth Token**. Fill in the **SMS to** textbox with your phone number that will be texted to.



4. Connect the 

node to the 

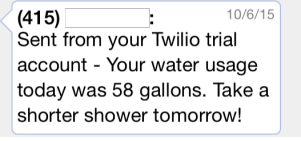
template node. The flow should look like this:



5. Click **Deploy** to save the changes.

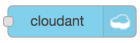
6. Click on the nob for the 

node to active this flow. You should receive a text message.

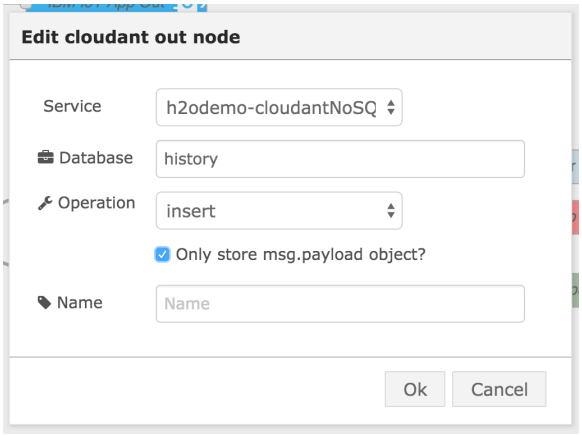


Store Usage Into Cloudant NoSQL Database

This section adds a Cloudant NoSQL database to the application and stores each usage metric reported. 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. However, it is a prerequisite for the **Retrieve Usage From Cloudant NoSQL Database** section.

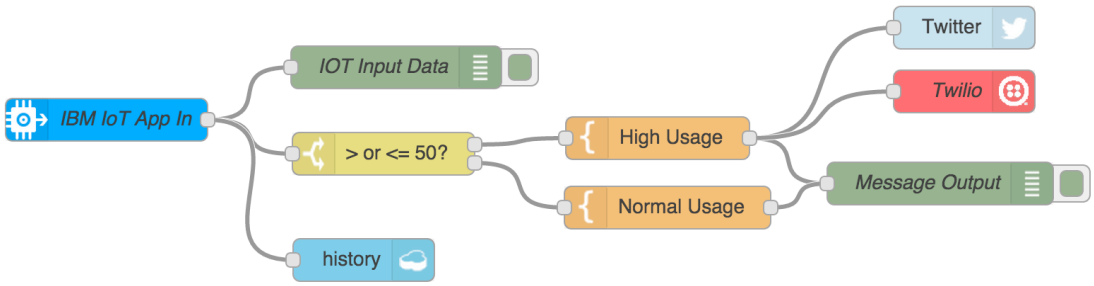
1. Add a 

node with the following settings:



2. Connect this node to the 

node. The flow should look like this:

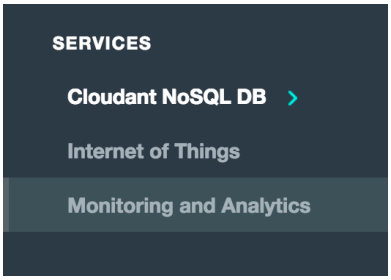


3. Click **Deploy** to save the changes.

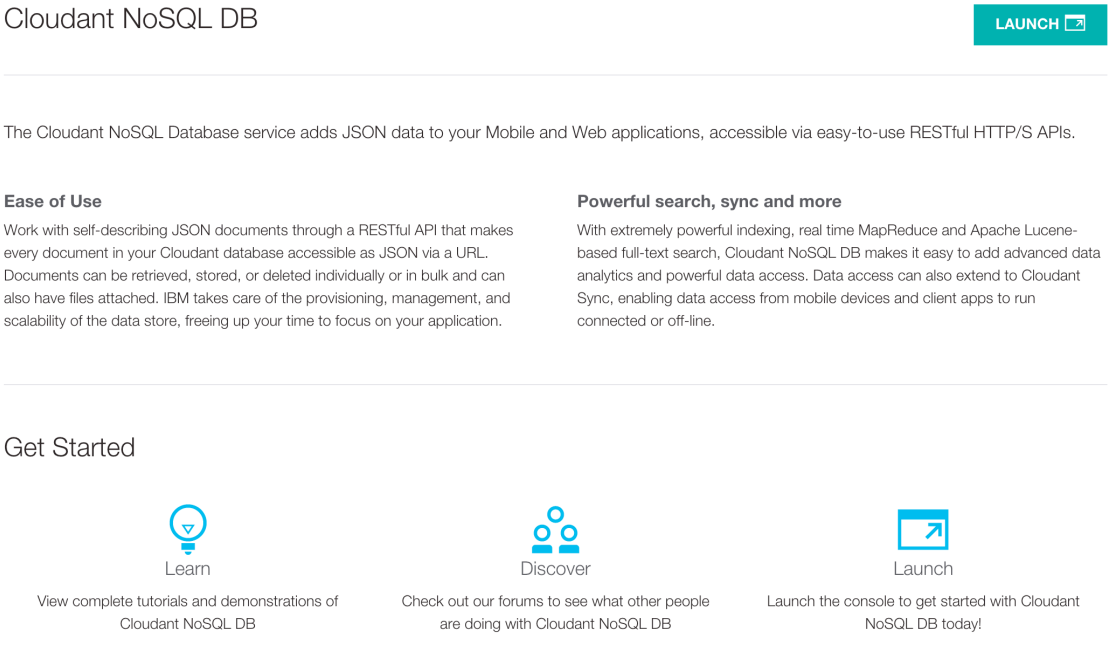
4. Click on the nob on the 

node.

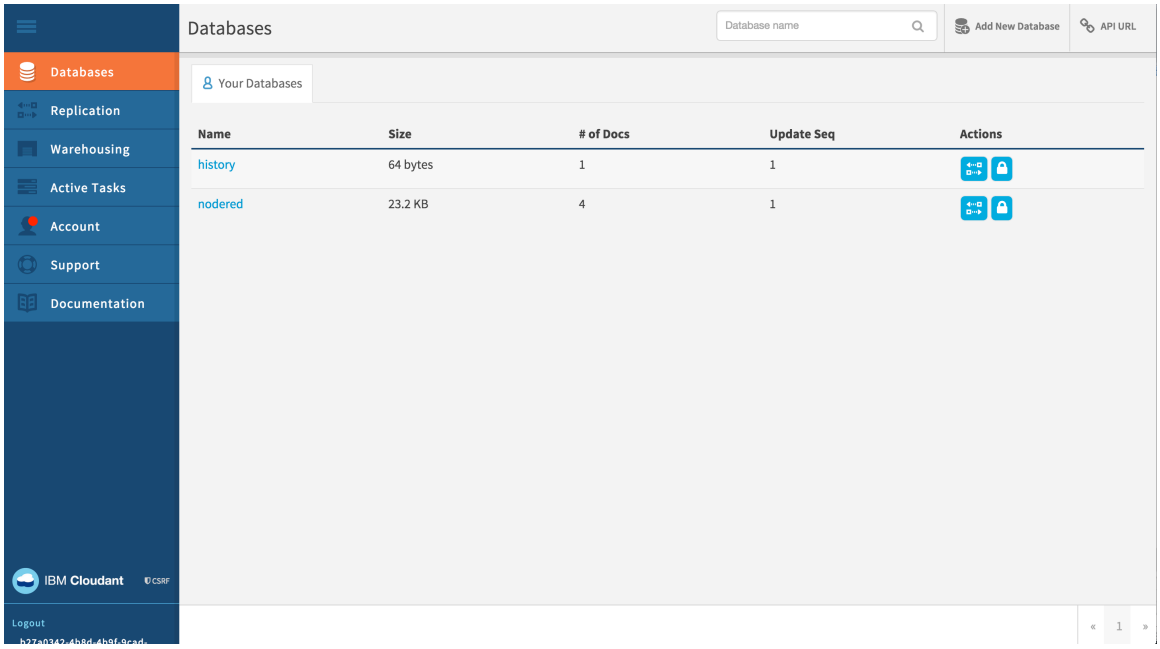
5. Go back to the IBM Bluemix dashboard and the app overview. Click on the **Cloudant NoSQL DB** link in the sidebar.



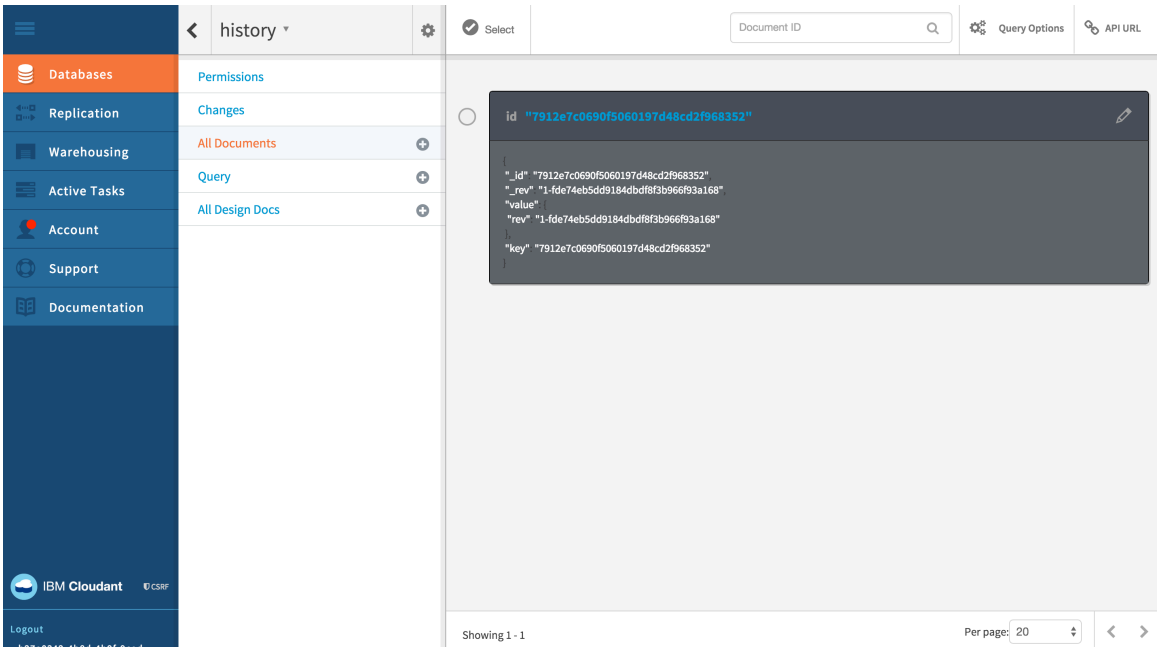
6. Click on the **Launch** button.



7. This is the Cloudant NoSQL database dashboard. A list of databases is displayed. The database named history contains documents representing the water usage device events that have been received by the application. Click on the database named **history.**



8. You can view the documents that are stored each time the IoT application receives data reported by the sensor.


history
Databases
Replication
Warehousing
Active Tasks
Account
Support
Documentation
IBM Cloudant
CSRF
Logout
Permissions
Changes
All Documents
Query
All Design Docs
Select
Document ID
Query Options
API URL
id "7912e7c0690f5060197d48cd2f968352"
{
"_id" "7912e7c0690f5060197d48cd2f968352"
"_rev" "1-fde74eb5dd9184dbdf8f3b966f93a168"
"value"
"rev" "1-fde74eb5dd9184dbdf8f3b966f93a168"
}
"key" "7912e7c0690f5060197d48cd2f968352"
}
Showing 1 - 1
Per page: 20

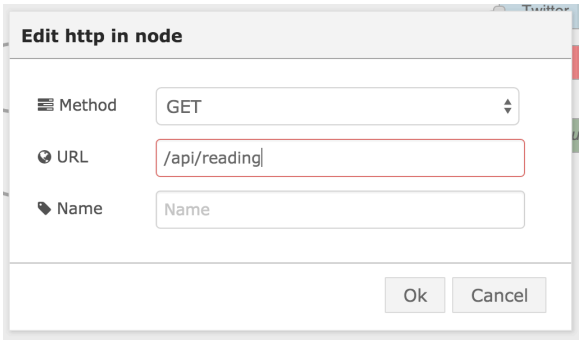
Retrieve Usage From Cloudant NoSQL Database

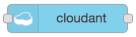
This section retrieves data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time. This section is optional and can be skipped. Completion of the section titled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

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

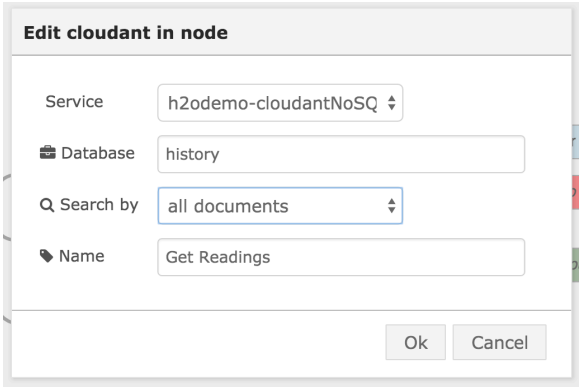
1. Add a 

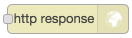
node with the following settings:



2. Add a 

node with the following settings:



3. Finally, add a 

node. Connect all three nodes together as follows:

4. Click **Deploy** to save the changes.

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

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

You should see data returned similar to the following:


```
[
  {
    "_id": "3845635942cd86ae8d6737a51b5db2e9",
    "_rev": "1-fde74eb5dd9184dbdf8f3b966f93a168",
    "usage": 40,
    "date": "2015-10-05"
  },
  {
    "_id": "76dc00d7bd155453d42da7603dcefdec",
    "_rev": "1-b904bc9b2bbc9234e64dfffbe339f870",
    "usage": 58,
    "date": "2015-10-06"
  }
]
```

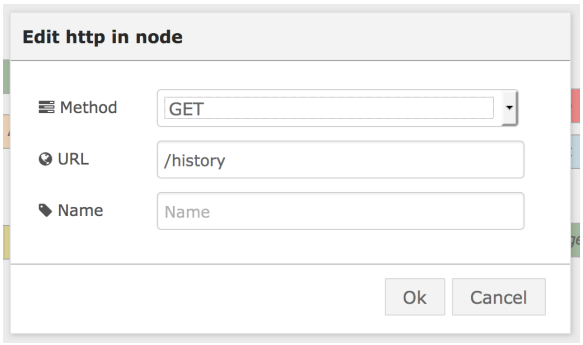

This data is in a format called JSON, and you can use this data in web applications. As you inject more data in the IoT example in Node-RED, this dataset will expand to include that data.

Create a Report Webpage

Now that we have data, let's create a webpage where you can write HTML and JavaScript to display the usage in a table. We'll use AngularJS to make a request to the HTTP endpoint that we previously created to output the water usage readings in the JSON format. Completion of the section titled **Retrieve Usage From Cloudant NoSQL Database** is required before beginning.

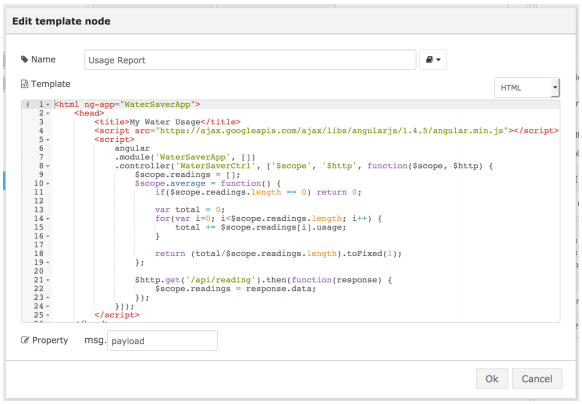
1. Add a 

node with the following settings:



2. Add a 

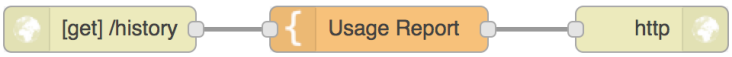
node with the following settings:



```
<html ng-app="WaterSaverApp">
    <head>
        <title>My Water Usage</title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.min.js"></script>
        <script>
            angular
            .module('WaterSaverApp', [])
            .controller('WaterSaverCtrl', ['$scope', '$http', function($scope, $http) {
                $scope.readings = [];

                $scope.average = function() {
                    if($scope.readings.length == 0) return 0;

                    var total = 0;
                    for(var i=0; i<$scope.readings.length; i++) {
                        total += $scope.readings[i].usage;
                    }

                    return (total/$scope.readings.length).toFixed(1);
                };

                $http.get('/api/reading').then(function(response) {
                    $scope.readings = response.data;
                });
            }]);
        </script>
    </head>
    <body ng-controller="WaterSaverCtrl">
        <h2>My Water Usage</h2>
        <p>Average usage: <span ng-bind="average()"></span></p>

        <table border="1">
            <tr>
                <th>Date</th>
                <th>Gallons Used</th>
            </tr>
            <tr ng-repeat="reading in readings | orderBy:'-date'">
                <td ng-bind="reading.date"></td>
                <td ng-bind="reading.usage"></td>
            </tr>
      </table>
  </body>
</html>
```

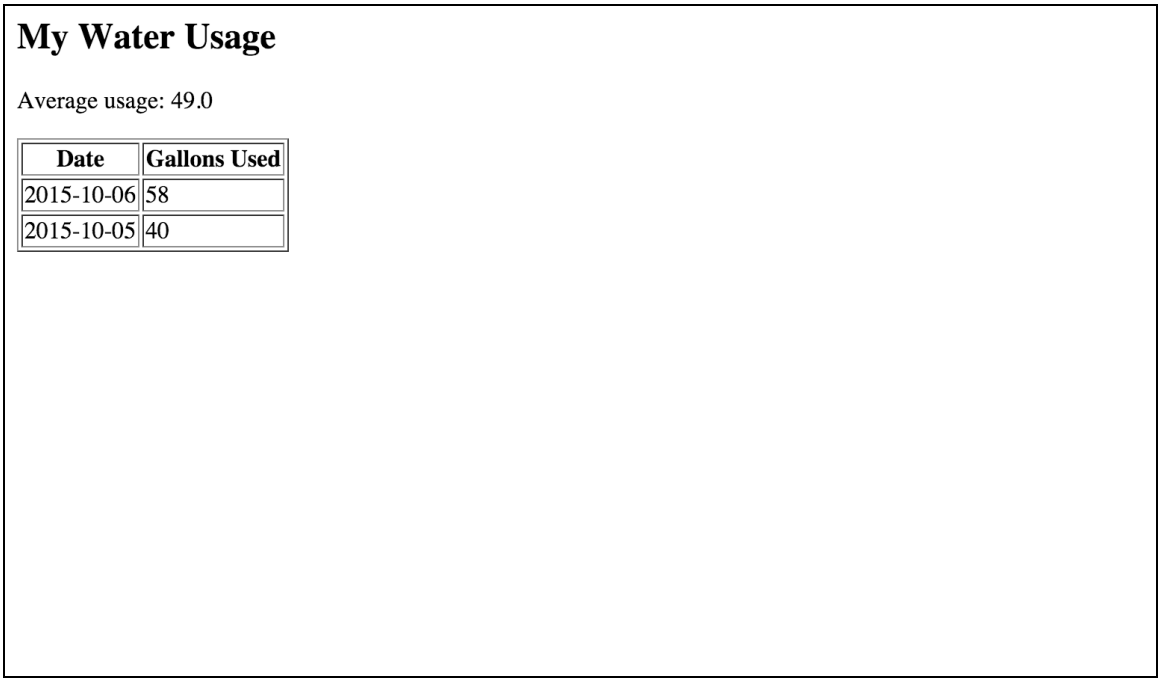
3. Add a 

node. Connect the three nodes together as follows:



4. Open your browser and visit your application's URL, appended with `/history`. If your application host was *something*, the URL would be:

`https://something.mybluemix.net/history`



This table will grow as more usage values are reported. The average will also adjust as new values are reported. You can modify this template and include additional HTML, JavaScript, and CSS. Find out more about AngularJS at `https://angularjs.org`.