Getting Started with the Watson IoT Java SDK

Java SIG Hands-On Lab

Download this PDF and Node-RED flows at
https://github.com/johnwalicki/watson-iot-java-workshop

Authors:
John Walicki | walicki@us.ibm.com | @johnwalicki

Table of Contents

Topics covered: Watson Internet of Things Platform, Java IoT SDK, IBM Bluemix.

In this lab, we will set up some sample Java IoT applications and connect to the IBM Bluemix and Watson IoT Platform.   We will send data via Java apps to the Watson IoT platform and display it on QuickStart and in the Watson IoT Platform dashboards. We will create a Registered Device and publish MQTT events using the IBM IoT Java Library. We will generate random temperature readings and, using the Watson IoT Real Time Insights, set alerts on high temperature readings.

Section 1 – Watson IoT Java SDK / Sample App Set Up

Step 1 – Review and Install Watson IoT Java SDK / Sample Applications

Instructions on how to install the Watson IoT Java Sample Applications can be found at
https://github.com/ibm-watson-iot/iot-application-samples/tree/master/java/standalone-samples

Building the sample - Required if you want to run the samples outside of Eclipse

- Clone the iot-application-samples project using git clone as follows:

```
$ git clone https://github.com/ibm-messaging/iot-application-samples.git
```

- Navigate to the standalone-samples project:

```
$ cd iot-application-samples\java\standalone-samples
```

- Run the maven build as follows:

```
$ mvn clean package
```

This will download the Java Client library for Watson IoT Platform and all required dependencies and starts the building process. Once built, the sample can be located in the target directory, for example, target\ibmiot-standalone-samples-0.0.1.jar.

Step 2 – Maven install of Watson IoT Java SDK

Optionally, if you want to install the Watson IoT Java SDK library using Maven:

```
$ mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes
-DarchetypeArtifactId=maven-archetype-quickstart
-DgroupId=com.ibm.messaging -DartifactId=watson-iot
-DinteractiveMode=false

$ cd watson-iot

$ mvn clean package
```

Likewise, the Watson IoT Java SDK library is also available in github:

```
$ git clone https://github.com/ibm-watson-iot/iot-java.git
```

Section 2 - Send data to Watson IoT Quickstart

The first sample application we will explore sends data to Watson IoT Quickstart using the MQTT Application Device event publish sample. This is an application sample that publishes an event on behalf of a device, to Watson IoT Quickstart service. The EPLv1.0 licensed source code is available at:

> https://github.com/ibm-watson-iot/iot-application-samples/blob/master/java/standalone-samples/src/main/java/com/ibm/iotf/sample/client/application/QuickstartMQTTApplicationDeviceEventPublish.java

Step 2 – Run QuickstartMQTTApplicationDeviceEventPublish sample outside Eclipse

- Navigate to the iot-application-samples/java/standalone-samples/src/main/java/com/ibm/iotf/sample/client/application directory
- Edit the QuickstartMQTTApplicationDeviceEventPublish.java file.
- Modify Line 50 to a unique string

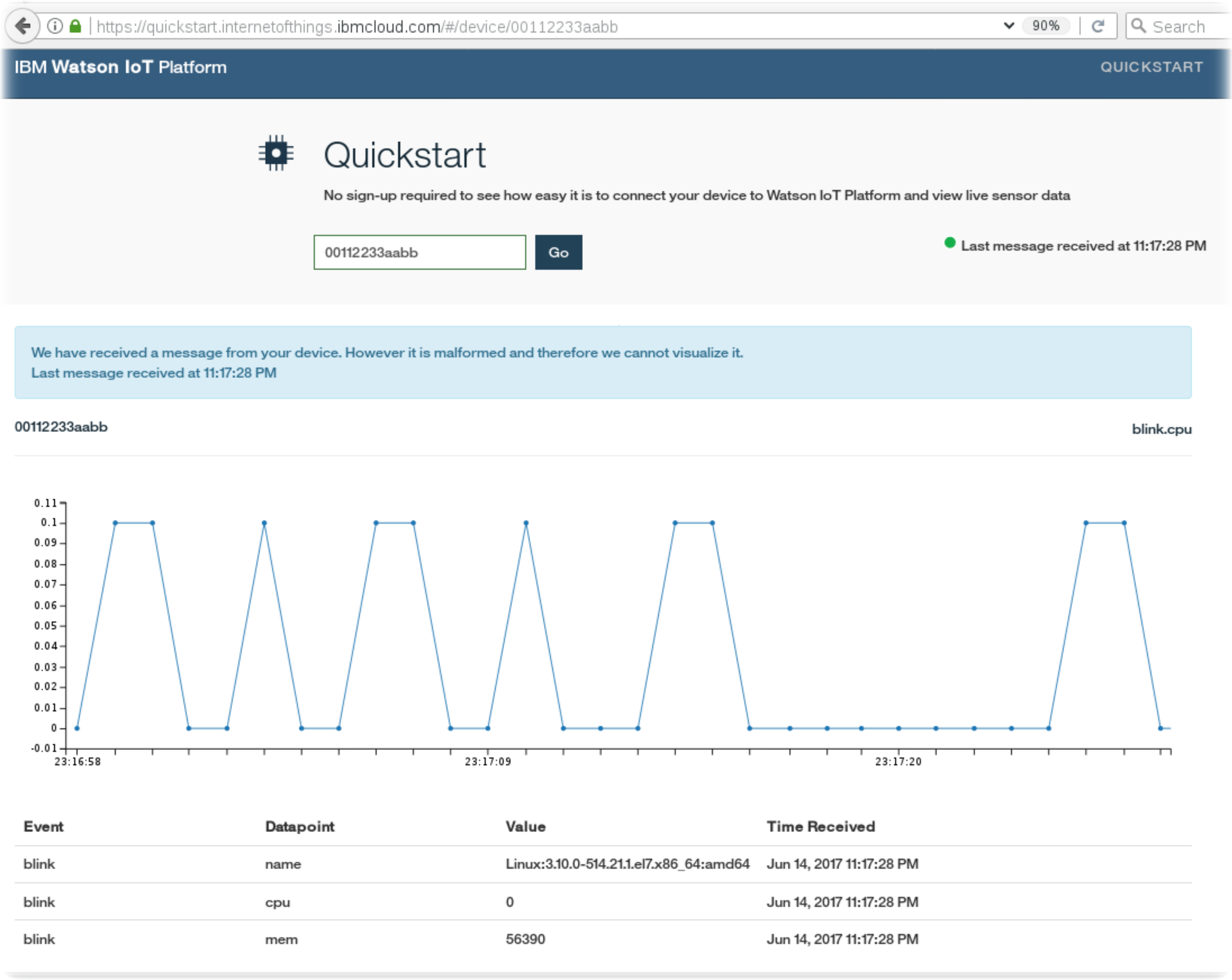
```
String deviceId = “00112233aabb”
```

- Save this source file.
- Go back to the root project directory where the POM.xml file is present and start the sample using the following command:

```
mvn exec:java
-Dexec.mainClass="com.ibm.iotf.sample.client.application.QuickstartMQTTAppli
cationDeviceEventPublish"
```

Observe that the application connects to Watson IoT Platform and publishes a device event named “blink” that automatically sending the system Name, CPU load and Memory used to the IBM Watson IoT Platform QuickStart.

Open a browser tab to **http://quickstart.internetofthings.ibmcloud.com/#/deviceId/<device id>**


https://quickstart.internetofthings.ibmcloud.com/#/device/00112233aabb
90%
Search
IBM Watson IoT Platform
QUICKSTART

Quickstart

No sign-up required to see how easy it is to connect your device to Watson IoT Platform and view live sensor data

● Last message received at 11:17:28 PM

We have received a message from your device. However it is malformed and therefore we cannot visualize it.
Last message received at 11:17:28 PM

00112233aabb

blink.cpu

Event	Datapoint	Value	Time Received
blink	name	Linux:3.10.0-514.21.1.el7.x86_64:amd64	Jun 14, 2017 11:17:28 PM
blink	cpu	0	Jun 14, 2017 11:17:28 PM
blink	mem	56390	Jun 14, 2017 11:17:28 PM

Section 3 Send sensor data to a Bluemix Application

Quickstart is a fast way to send and see sensor data on the Watson IoT Platform. The next step is to create a Bluemix Watson IoT application that receives the Java application sensor data via the Quickstart service and takes actions on it.

- If you do not yet have a Bluemix account, visit **http://bluemix.net/registration** and enter the requested information. Check your email to confirm the Bluemix account creation.

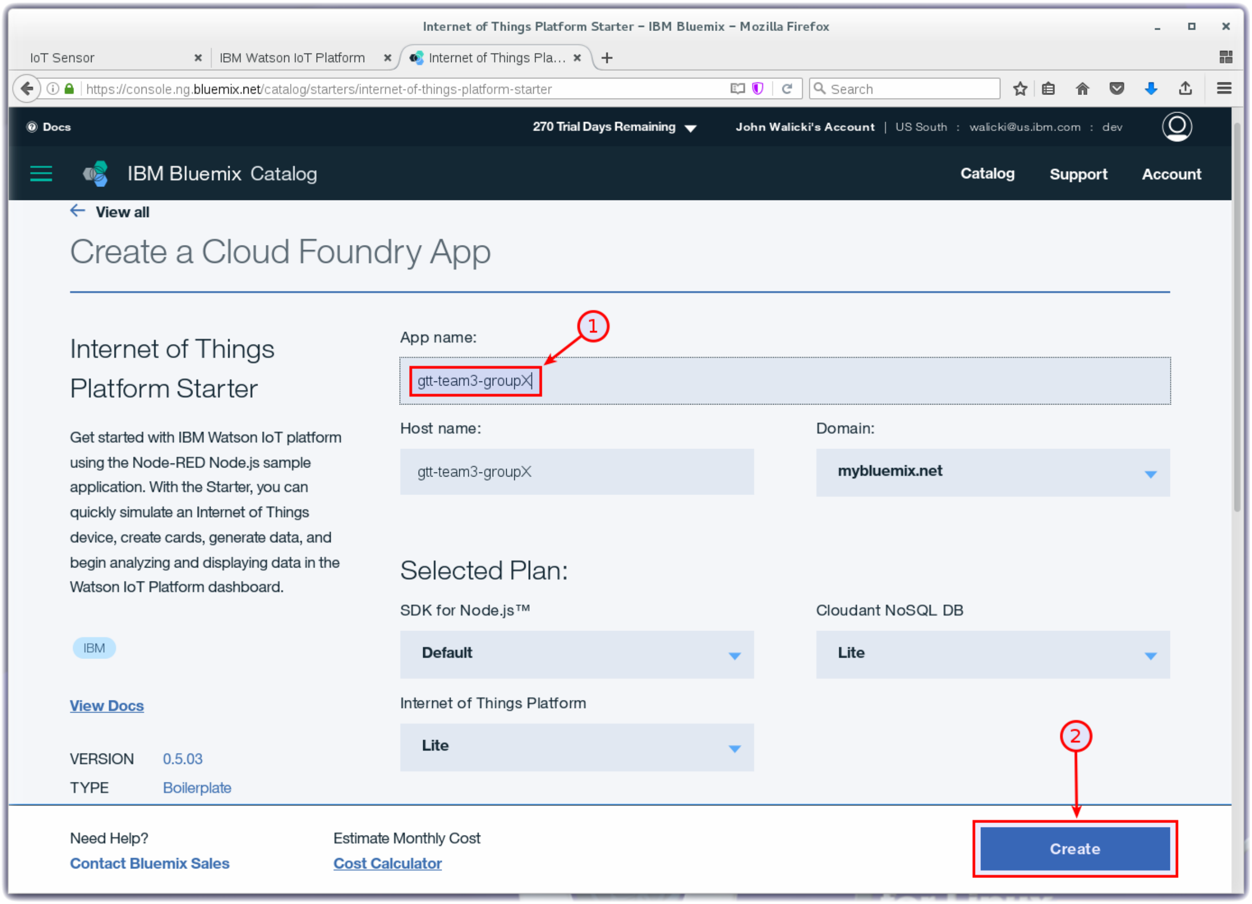
Section 4 – Create an Internet of Things Starter App

Step 1 – Create an IoT Starter Application

Now that we have sent the Java sample data readings to Watson IoT Quickstart, in this Section we will create an IoT Starter Application to ingest and analyze the Quickstart data.

1. The **Internet of Things Platform Starter** boilerplate is designed with pre-assembled services that work together. The Internet of Things Platform Starter includes a Node-RED Node.js web server, Cloudant database to store the sensor data, and the IoT platform service so you can connect devices.
2. Name your application something unique (1). If you choose ***myapp***, your application will be located at http://***myapp***.mybluemix.net There can only be one "***myapp***" application and URL registered in IBM Bluemix.
3. Give the application a unique name (1) - eg. **java-sig-workshop-teamX**

4. Press the Create button (2).


Internet of Things Platform Starter – IBM Bluemix – Mozilla Firefox
IoT Sensor
IBM Watson IoT Platform
Internet of Things Pla…
https://console.ng.bluemix.net/catalog/starters/internet-of-things-platform-starter
Search
Docs
270 Trial Days Remaining
John Walicki's Account | US South : walicki@us.ibm.com : dev
IBM Bluemix Catalog
Catalog
Support
Account
View all
1
2
Create

Create a Cloud Foundry App

Internet of Things Platform Starter

Get started with IBM Watson IoT platform using the Node-RED Node.js sample application. With the Starter, you can quickly simulate an Internet of Things device, create cards, generate data, and begin analyzing and displaying data in the Watson IoT Platform dashboard.

IBM

View Docs

VERSION	0.5.03
TYPE	Boilerplate

App name: gtt-team3-groupX

Host name: gtt-team3-groupX

Domain: mybluemix.net

Selected Plan:

SDK for Node.js™: Default

Cloudant NoSQL DB: Lite

Internet of Things Platform: Lite

Need Help?
Contact Bluemix Sales

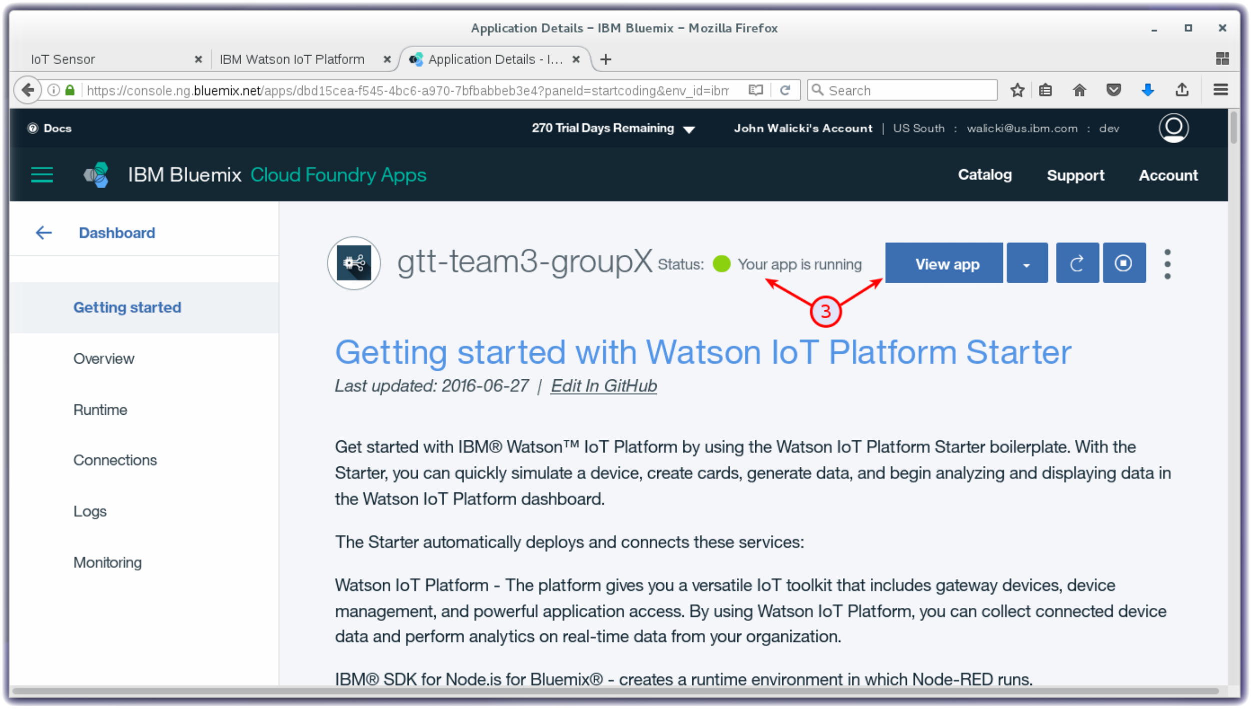
Estimate Monthly Cost
Cost Calculator

5. 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.

Step 2 - Launch the IoT Starter Application

- Once the Green “Your app is running” appears, Click the View app View App button (3).

Step 3 – Open the Node-RED visual programming editor


Application Details – IBM Bluemix – Mozilla Firefox
IoT Sensor
IBM Watson IoT Platform
Application Details - I…
https://console.ng.bluemix.net/apps/dbd15cea-f545-4bc6-a970-7bfbabbeb3e4?paneId=startcoding&env_id=ibm
Search
Docs
270 Trial Days Remaining
John Walicki's Account | US South : walicki@us.ibm.com : dev
IBM Bluemix Cloud Foundry Apps
Catalog
Support
Account
Dashboard
Getting started
Overview
Runtime
Connections
Logs
Monitoring
gtt-team3-groupX Status: Your app is running
View app
3

Getting started with Watson IoT Platform Starter

Last updated: 2016-06-27 | Edit In GitHub

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

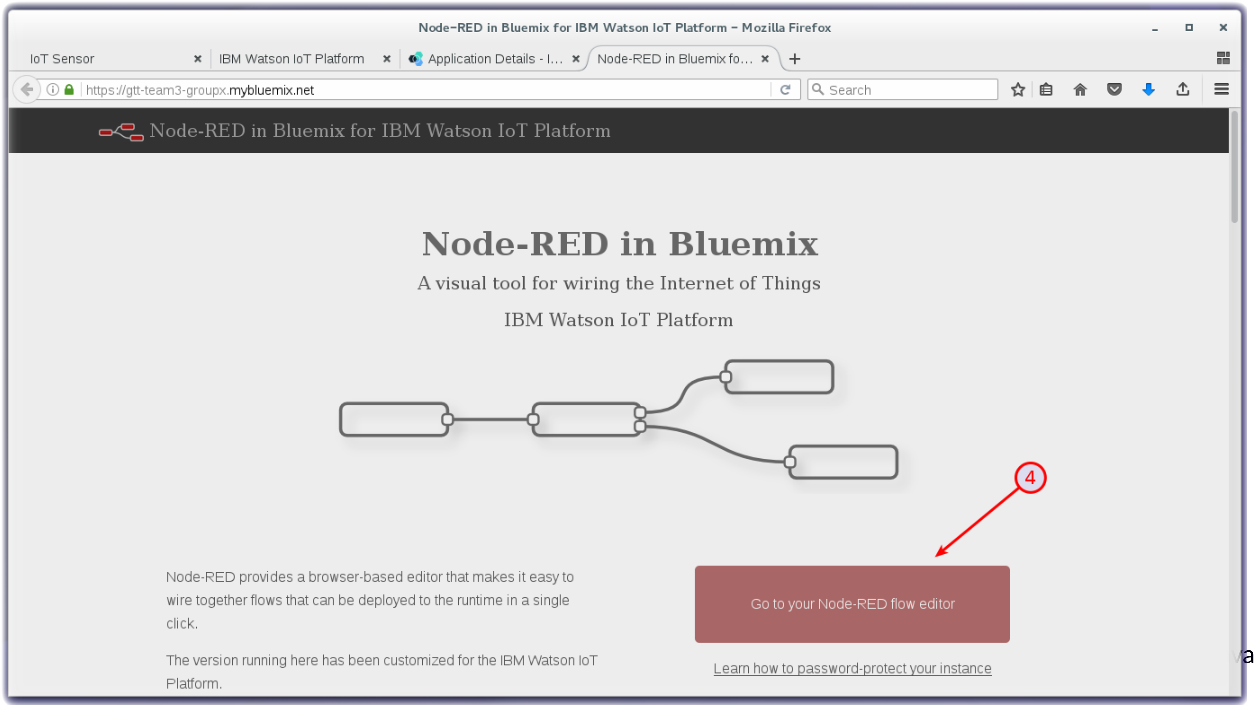
The Starter automatically deploys and connects these services:

Watson IoT Platform - The platform gives you a versatile IoT toolkit that includes gateway devices, device management, and powerful application access. By using Watson IoT Platform, you can collect connected device data and perform analytics on real-time data from your organization.

IBM® SDK for Node.js for Bluemix® - creates a runtime environment in which Node-RED runs.

1. A new browser tab will open to the Node-RED start page. Node-RED is an open-source Node.js application that provides a visual programming editor that makes it easy to wire together flows. Click the red button **Go to your Node-RED flow editor** (4) to launch the editor.

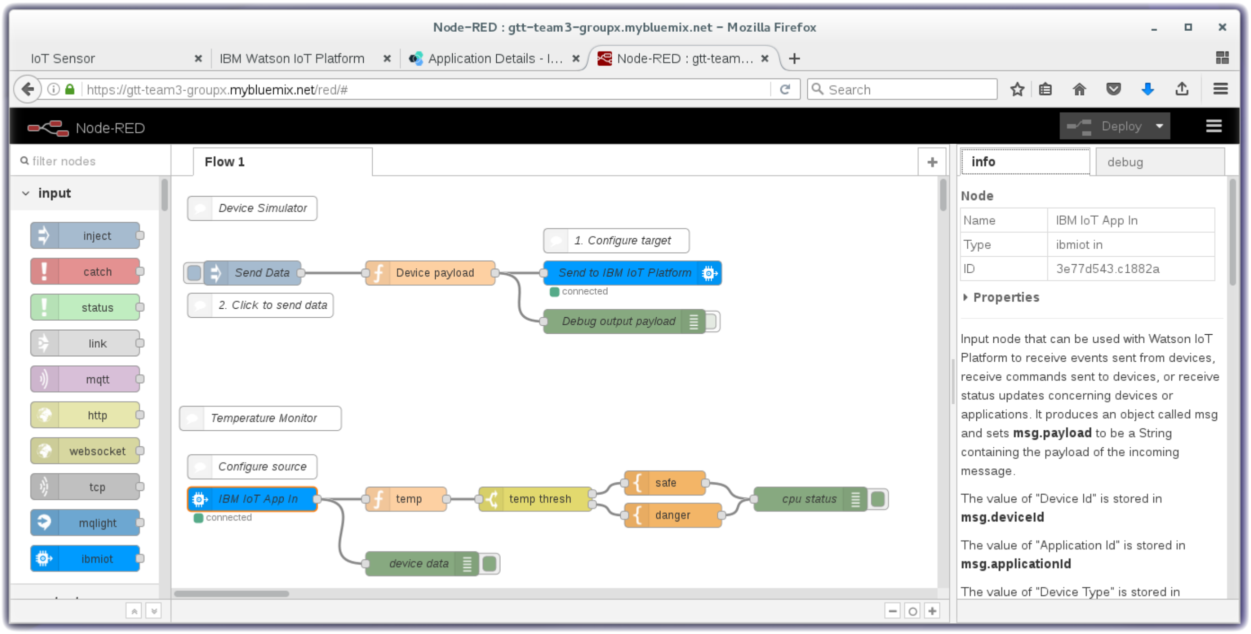
2. The Node-RED Visual Programming Editor will open with a default flow.
3. On the left side is a palette of nodes that you can drag onto the flow.


Node-RED in Bluemix for IBM Watson IoT Platform – Mozilla Firefox
IoT Sensor
IBM Watson IoT Platform
Application Details - I…
Node-RED in Bluemix fo…
https://gtt-team3-groupx.mybluemix.net
Search
Node-RED in Bluemix for IBM Watson IoT Platform
Node-RED in Bluemix
A visual tool for wiring the Internet of Things
IBM Watson IoT Platform
4
Go to your Node-RED flow editor
Learn how to password-protect your instance

Node-RED provides a browser-based editor that makes it easy to wire together flows that can be deployed to the runtime in a single click.

The version running here has been customized for the IBM Watson IoT Platform.

4. You can wire nodes together to create a program.



Node

Name	IBM IoT App In
Type	ibmiot in
ID	3e77d543.c1882a

▸ Properties

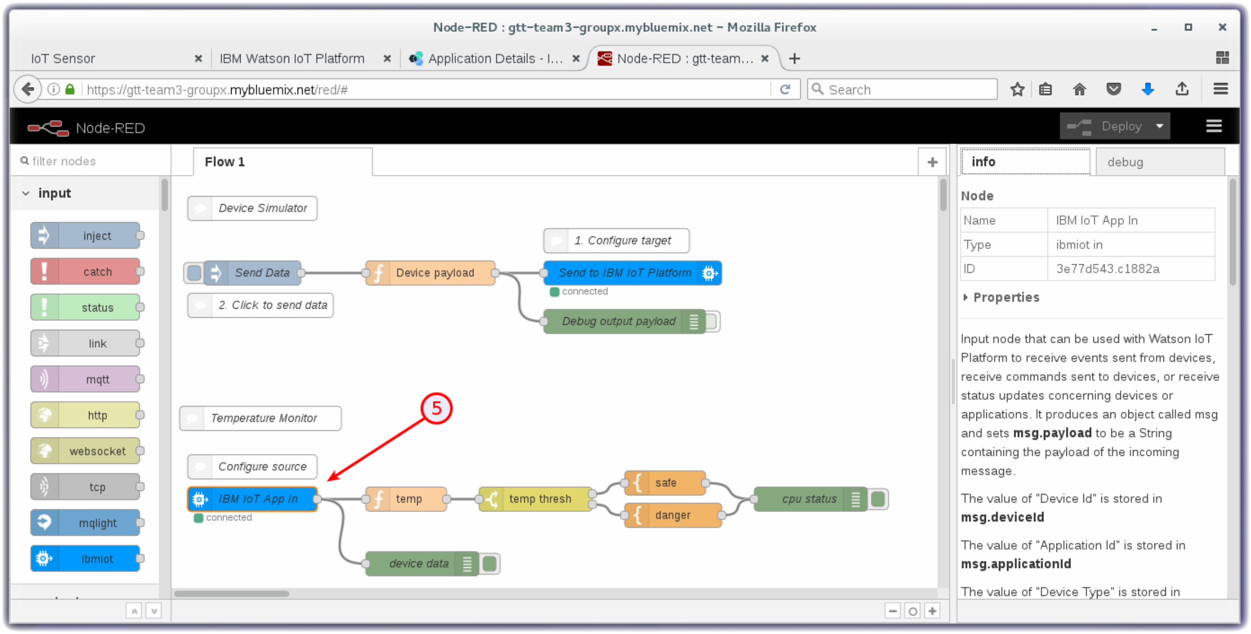
Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

The value of "Device Id" is stored in **msg.deviceId**

The value of "Application Id" is stored in **msg.applicationId**

The value of "Device Type" is stored in

5. The top half of the sample IoT Starter flow is not applicable to this workshop.
6. The bottom half of the sample will be modified to send email alerts.
7. Double Click on the IBM IoT App In (5) node.



Node

Name	IBM IoT App In
Type	ibmiot in
ID	3e77d543.c1882a

▸ Properties

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

The value of "Device Id" is stored in **msg.deviceId**

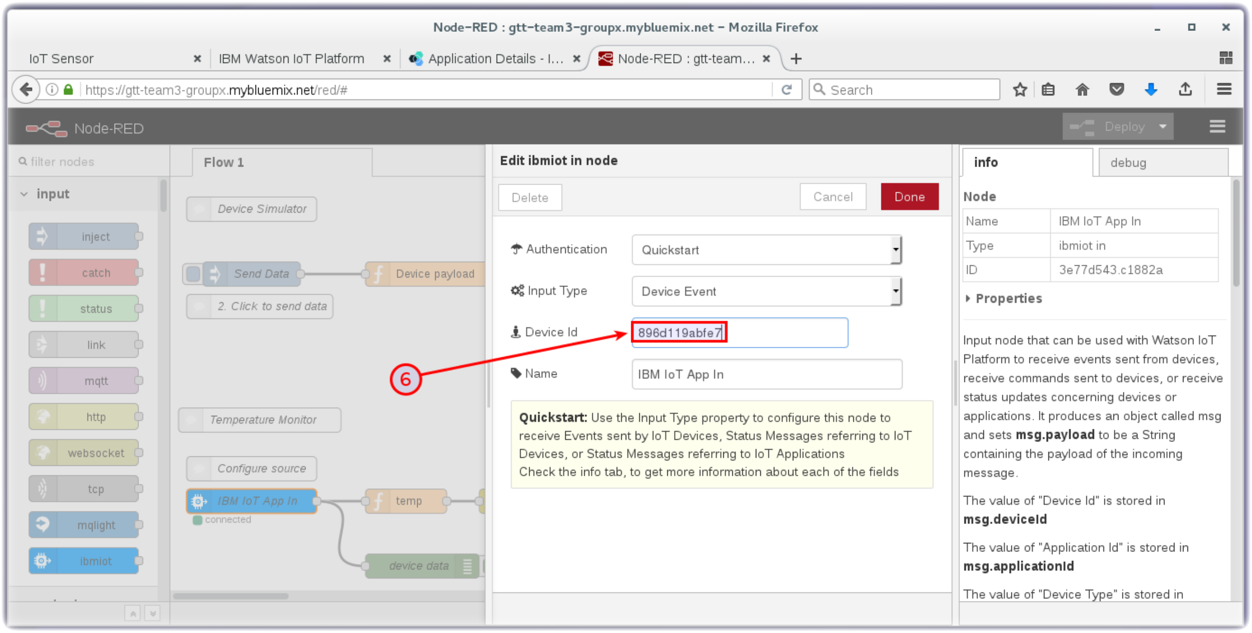
The value of "Application Id" is stored in **msg.applicationId**

The value of "Device Type" is stored in

Observations:

Step 4 – Connect the IBM IoT Node to your Java IoT device application

- An ibmiot in node configuration panel will open. Paste the Java Quickstart Device ID from Section 2 Step 2 into the “Device ID” field (6) and click on the Done button.



Edit ibmiot in node

Delete | Cancel | Done

Authentication: Quickstart

Input Type: Device Event

Device Id: 896d119abfe7

Name: IBM IoT App In

Quickstart: Use the Input Type property to configure this node to receive Events sent by IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields

Node

Name	IBM IoT App In
Type	ibmiot in
ID	3e77d543.c1882a

Properties

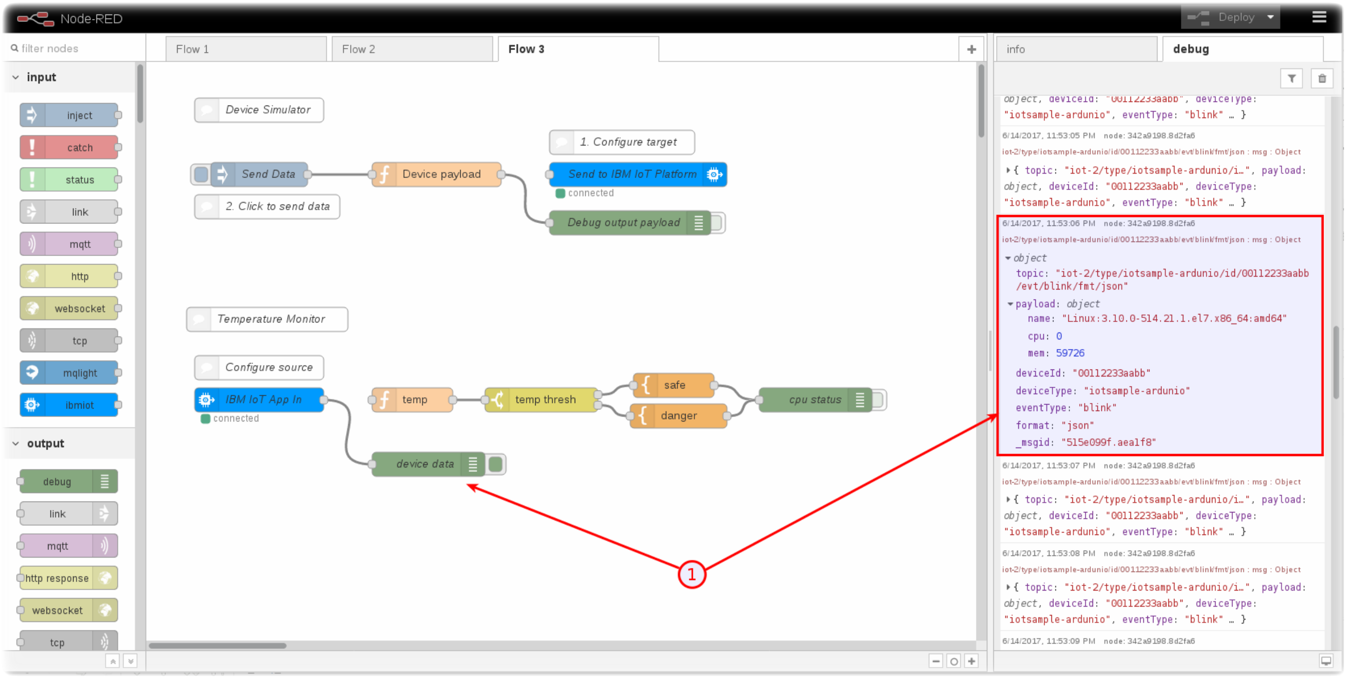
Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

The value of "Device Id" is stored in **msg.deviceId**

The value of "Application Id" is stored in **msg.applicationId**

The value of "Device Type" is stored in

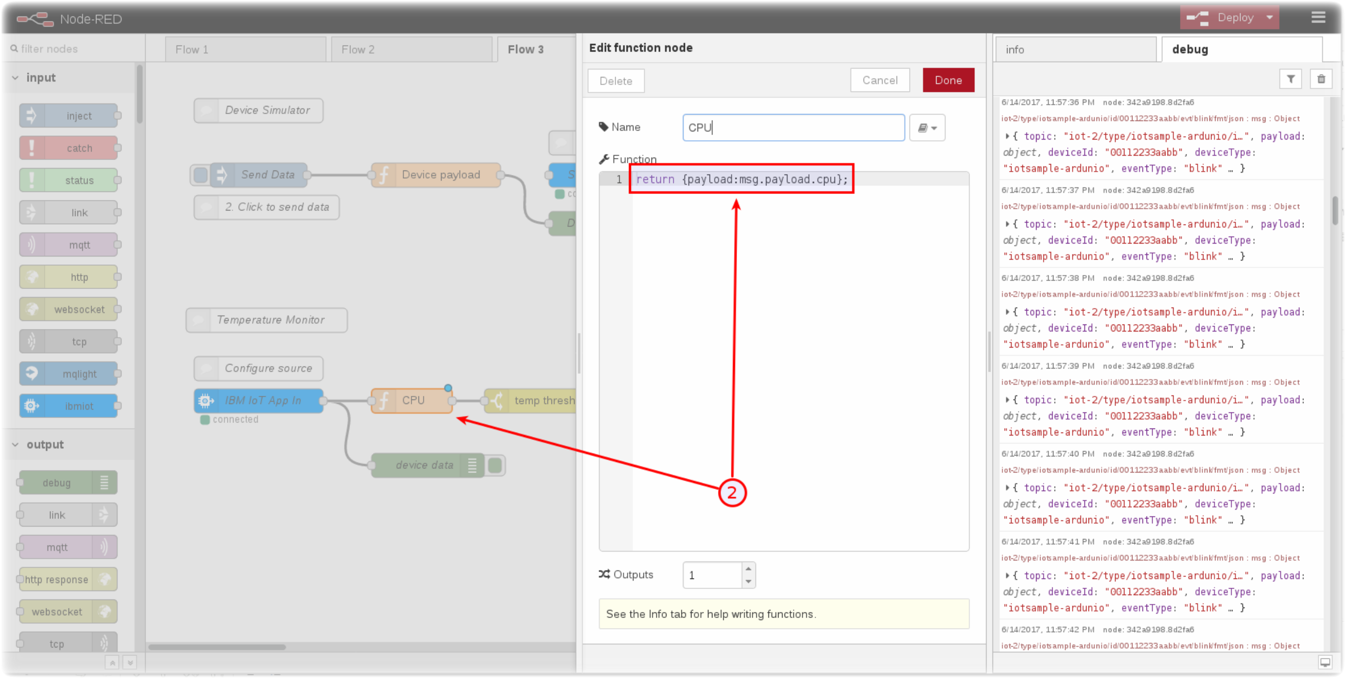
- Click the red Deploy Deploy button on the top of menu bar to deploy the Node-RED flow.
- Turn to the “debug” tab to see the sensor data flowing through your Node-RED application.



```
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:53:05 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:53:06 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▾object
  topic: "iot-2/type/iotsample-ardunio/id/00112233aabb
  /evt/blink/fmt/json"
 ▾payload: object
    name: "Linux:3.10.0-514.21.1.el7.x86_64:amd64"
    cpu: 0
    mem: 59726
  deviceId: "00112233aabb"
  deviceType: "iotsample-ardunio"
  eventType: "blink"
  format: "json"
  _msgid: "515e099f.aea1f8"
6/14/2017, 11:53:07 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:53:08 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:53:09 PM   node: 342a9198.8d2fa6
```

- Expand the twisties in the JSON object to see the payload values.

- Since the QuickStartMQTTApplicationDeviceEventPublish program is not sending d.temp, we need to modify this to set an alert on the cpu value.



Edit function node

Delete Cancel Done

Name: CPU

Function

```
return {payload:msg.payload.cpu};
```

Outputs: 1

See the Info tab for help writing functions.

```
6/14/2017, 11:57:36 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:57:37 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:57:38 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:57:39 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:57:40 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:57:41 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
▸{ topic: "iot-2/type/iotsample-ardunio/i…", payload:
object, deviceId: "00112233aabb", deviceType:
"iotsample-ardunio", eventType: "blink" … }
6/14/2017, 11:57:42 PM   node: 342a9198.8d2fa6
iot-2/type/iotsample-ardunio/id/00112233aabb/evt/blink/fmt/json : msg : Object
```

- Double click on the temp node (2)
- Change the Name of the node to CPU and modify the msg.payload.d.temp to msg.payload.cpu
- Click the red Deploy Deploy button on the top of menu bar to deploy the Node-RED flow.