IBM i Meets IoT

Michael Dawson
IBM Community Lead for Node.js

Jesse Gorzinski
jgorzins@us.ibm.com

Agenda Key:

© 2019 IBM Corporation

About Michael Dawson
IBM Community Lead for Node.js

- Active Node.js community member
  - Collaborator
  - Node.js Technical Steering Committee TSC Chair
  - Community Committee member
  - Working group(s) member/leadership

- Twitter: @mhdawson1
- GitHub: @mhdawson
- Linkedin: https://www.linkedin.com/in/michael-dawson-6051282

© 2019 IBM Corporation

About **Jesse**

- Business Architect of Open Source on i
  - Leader of development teams
  - Owns IBM i OSS strategy

- Twitter: @IBMJesseG
- GitHub: @ThePrez
- Linkedin: https://www.linkedin.com/in/ibmjesseg

© 2019 IBM Corporation

Agenda

- Intro to IoT and MQTT
- Using MQTT with Node.js and IBM i
- Lets Look at some devices
- Anatomy of a simple MQTT Light and Temperature Sensor
- Consuming MQTT data on IBM i
- Leveraging the Cloud

© 2019 IBM Corporation

IoT Introduction

- Internet of Things (IoT)
  - network of physically connected devices (things)
  - devices provide data
  - devices can be controlled
  - https://en.wikipedia.org/wiki/Internet_of_Things

© 2019 IBM Corporation

MQTT Introduction

- MQTT (MQ Telemetry Transport)
  - lightweight publish/subscribe
  - small footprint
  - low bandwidth (minimum size is 2 bytes)
  - From http://mqtt.org/

> "MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol"

© 2019 IBM Corporation

MQTT - History

- Invented in 1999
  - Andy Standford-Clark(IBM)
  - Arlen Nipper (Eurotech)

- IBM and Eurotech Donated MQTT to Eclipse Paho project in 2013 - https://www.eclipse.org/paho/
  - open-source client implementations

- Mosquitto broker also moved into the Eclipse foundation in 2013 - https://projects.eclipse.org/projects/technology.mosquitto
  - open-source broker

- Version 3.11 is an OASIS standard
- ISO standard as of 2016 (IOS/IC 20922)

- Version v5.0 official as OASIS standard as of April 2019

© 2019 IBM Corporation

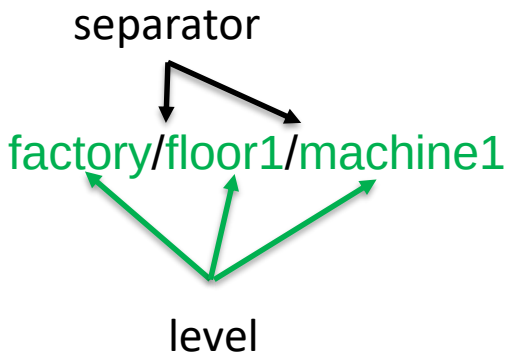
MQTT – Key terminology

- Client
  - Publishers and subscribers
  - Paho
  - mqtt.js
- Broker
  - Mosquitto
  - ActiveMQ
  - Mosca (Node.js)
- Topic
  - Shared id to subscribe or publish on
- Message
  - Free form text
- QoS
  - Quality of Service (0-2)

© 2019 IBM Corporation

MQTT Topics

- **Topics** are one or more **levels** separated by the topic level **separator**



- **Restrictions**
  - Must be at least one character
  - Case sensitive

- **Wildcards**

+ Matches one level

factory/+/machine1 factory/floor1/machine1 (yes) factory/floor1/room1/machine1 (no)

matches multiple levels

only allowed at end

factory/# factory/floor1/machine1 (yes) factory/floor1/room1/machine1 (yes)

© 2019 IBM Corporation

MQTT QoS

- 3 Levels
  - 0 – At most once
  - 1 – At least once
  - 2 – Exactly once
- Downgrade of QoS
  - Uses QoS of receiver, so downgrade may occur if sending used higher level
- More overhead for each level
- 0 is generally the default

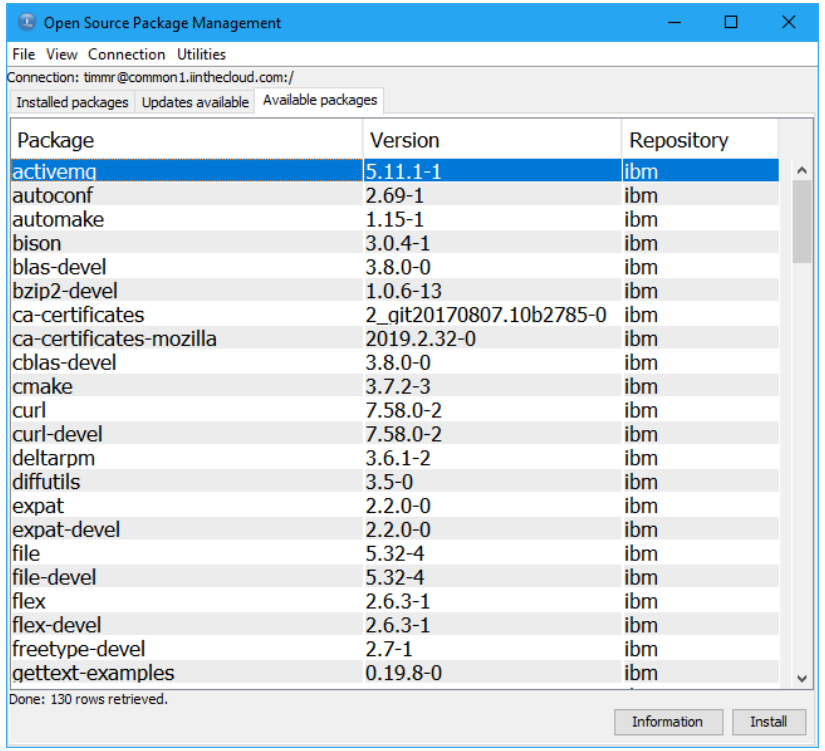
© 2019 IBM Corporation

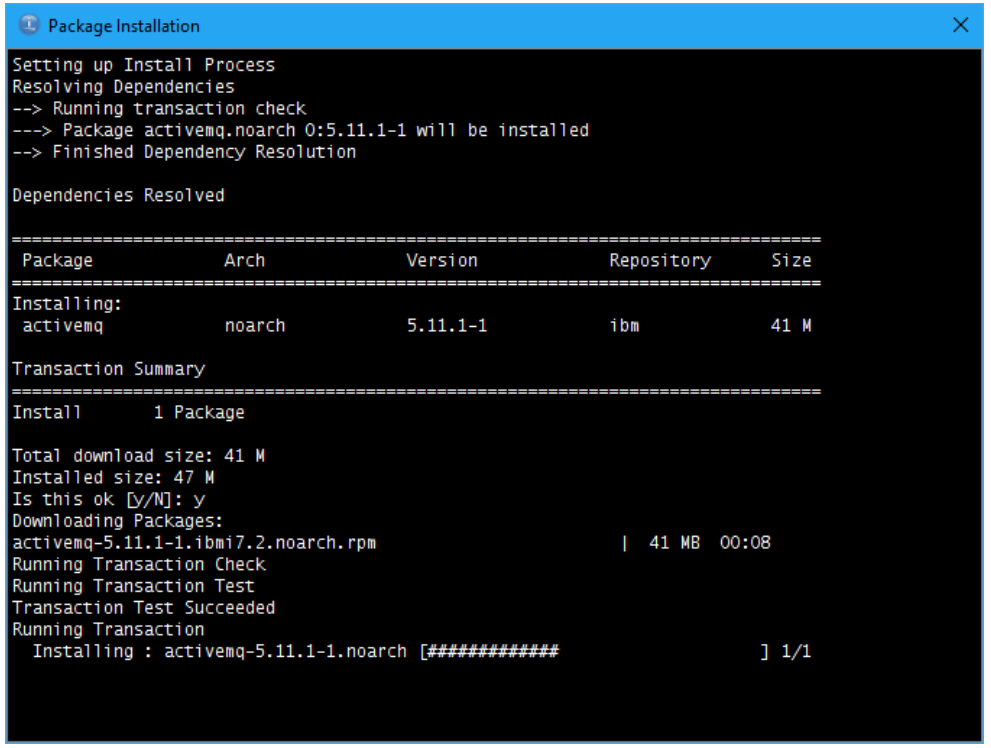
Why IoT with IBM i?

© 2019 IBM Corporation

Using MQTT with IBM i

- ActiveMQ Install





© 2019 IBM Corporation

Using MQTT with IBM i

- Start ActiveMQ

```
activemq start
```

- Configure
  - http://activemq.apache.org/mqtt.html
  - Default config in - /QOpenSys/pkgs/lib/activemq/conf/

- Support
  - http://ibm.biz/ibmi-oss-support

© 2019 IBM Corporation

MQTT install

npm install mqtt

© 2019 IBM Corporation

Simple Client (client-local.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// setup mqtt
let  mqttOptions;
mqttOptions = {
                key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
                cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
                ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
                clientId: 'simple-client',
                checkServerIdentity: function() { return undefined },
                rejectUnauthorized: false,
                username: '',
                password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
  console.log('connected');
  mqttClient.subscribe('onibmi/topic');
  mqttClient.on('message', (topic, message) => {
    console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
  });
});
```


© 2019 IBM Corporation

Simple Publisher (publisher-local.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// setup mqtt
let  mqttOptions;
mqttOptions = {
                key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
                cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
                ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
                clientId: 'simple publish',
                checkServerIdentity: function() { return undefined },
                rejectUnauthorized: false,
                username: '',
                password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
  console.log('connected');
  setInterval(() => {
    console.log('publishing');
    mqttClient.publish('onibmi/topic', 'hello world');
  }, 10000 );
});
```


© 2019 IBM Corporation

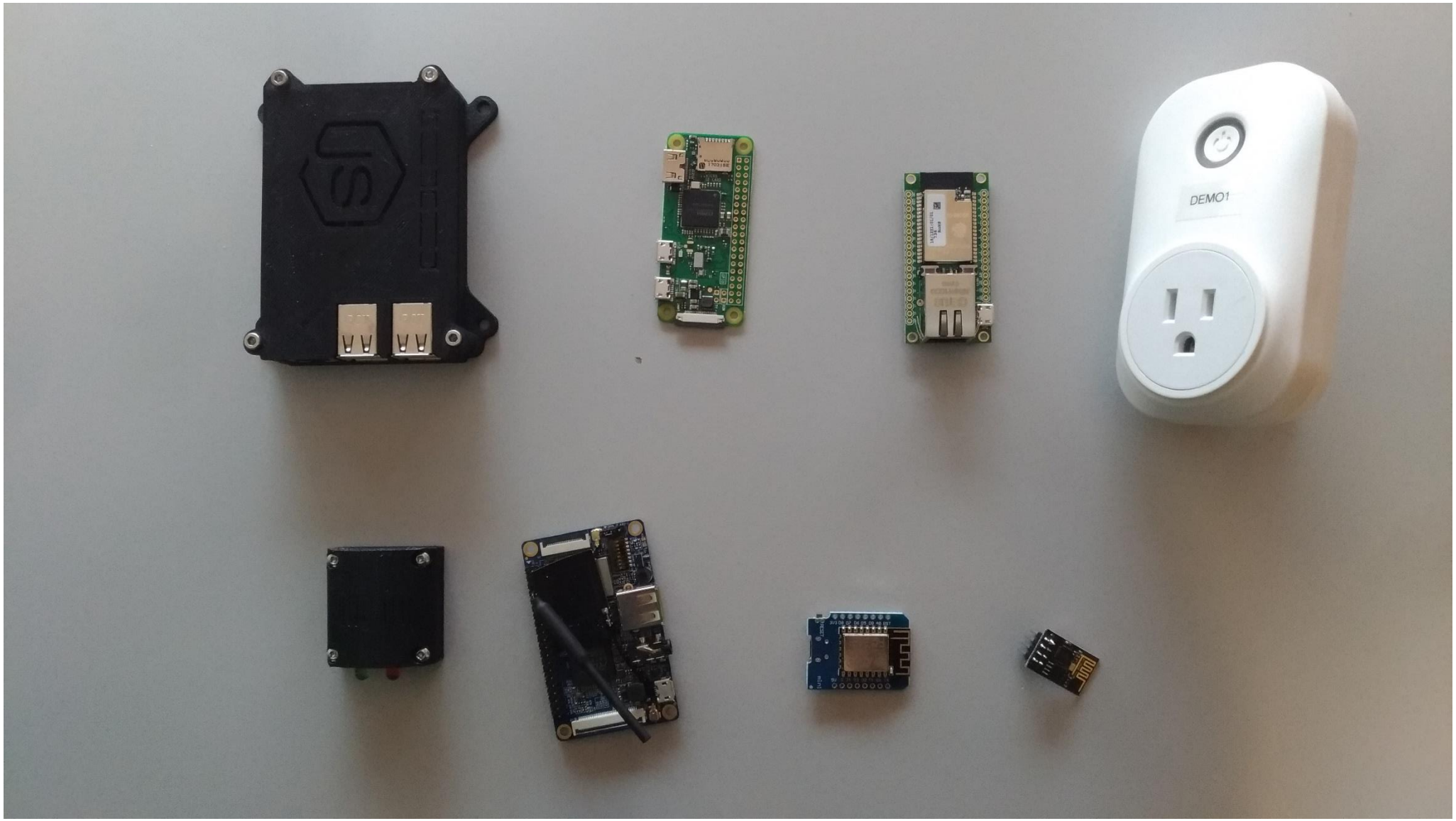
Demo

```
driveway@Common1:~/michael/mqttsamples$ node
publish-local.js
connected
publishing
publishing
publishing
publishing
publishing
publishing
publishing
publishing
```

```
driveway@Common1:~/michael/mqttsamples$ node client-local.js
connected
message received topic (onibmi/topic) message (hello world)
message received topic (onibmi/topic) message (hello world)
message received topic (onibmi/topic) message (hello world)
message received topic (onibmi/topic) message (hello world)
message received topic (onibmi/topic) message (hello world)
```


© 2019 IBM Corporation

Lets look at some Devices

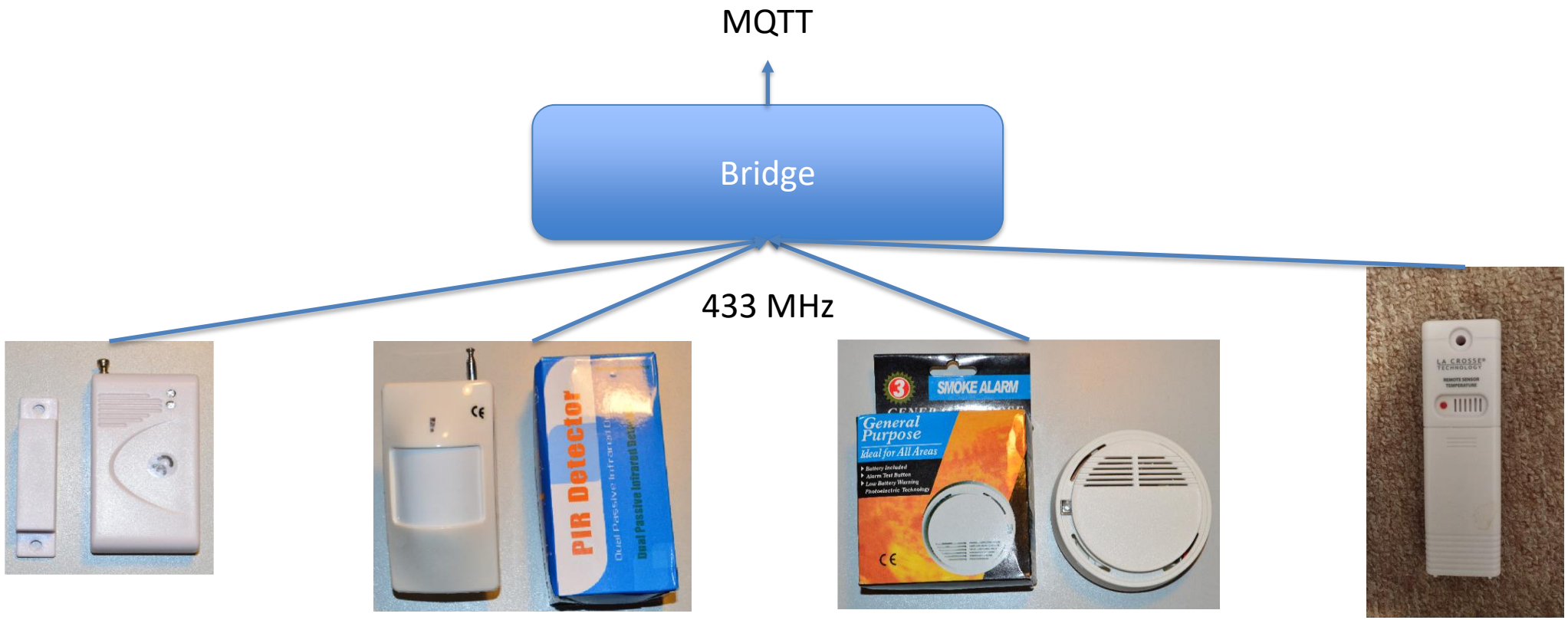


© 2019 IBM Corporation

Other/Existing Devices

- Common approach is gateway or bridge

- As an example 433MHz to MQTT bridge
  https://github.com/mhdawson/arduino-esp8266/tree/master/Mqtt433Bridge



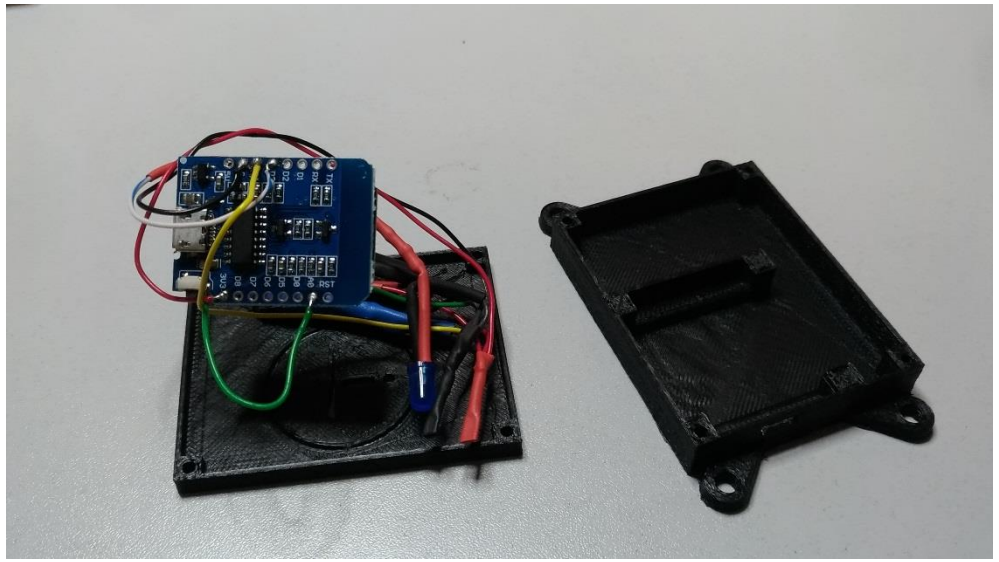
© 2019 IBM Corporation

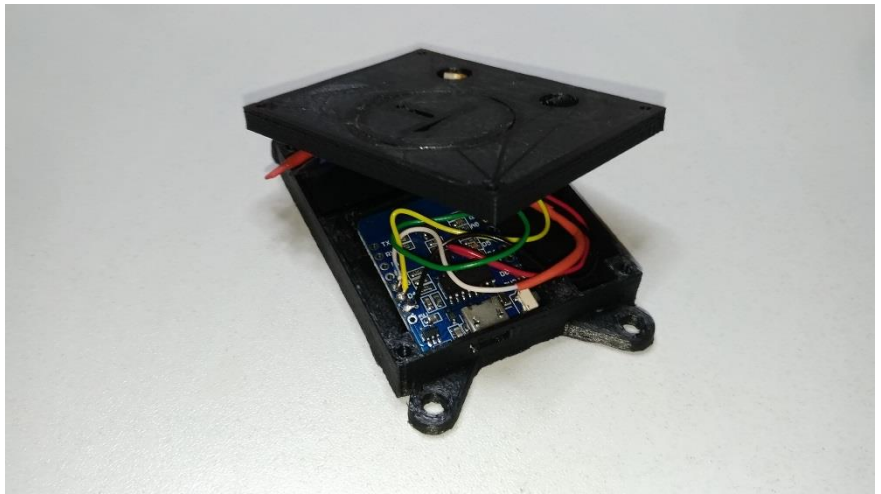
Anatomy of a Simple Device

- Maybe your business grows plants?
- **Temperature** and **Light** intensity through the day might be interesting?

© 2019 IBM Corporation

Anatomy of a Simple Device – Temp and Light Sensor

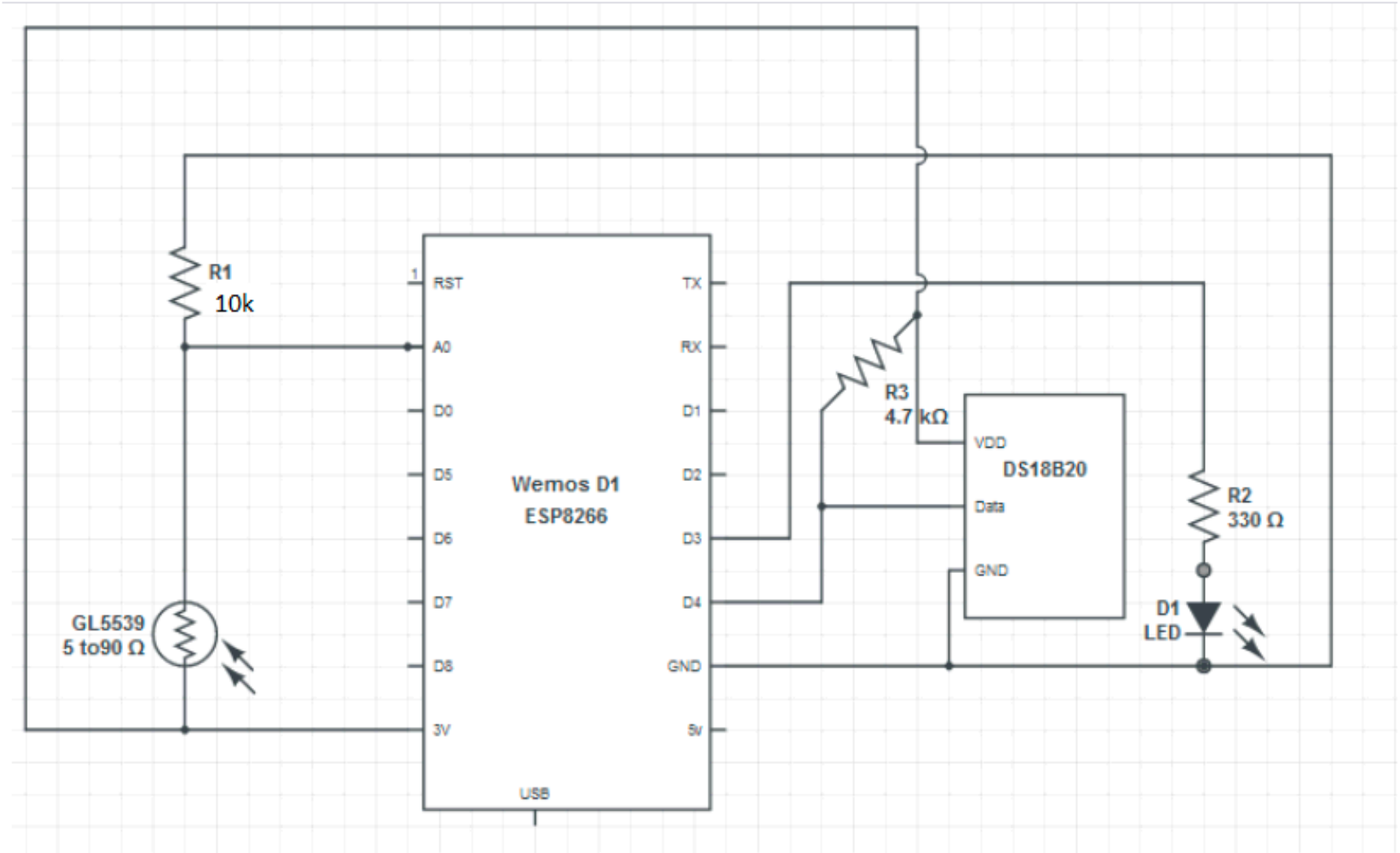




https://github.com/mhdawson/arduino-esp8266/tree/master/TempAndLightSensor

© 2019 IBM Corporation

Anatomy of a Simple Device



© 2019 IBM Corporation

Anatomy of a Simple Device

```
#ifndef __SENSOR_CONFIG_H__
#define __SENSOR_CONFIG_H__
#define LIGHT_TOPIC "factory/1/light"
#define TEMP_TOPIC  "factory/1/temp"
#define LED_TOPIC   "factory/1/led"
#endif
```

```
// wireless setup
#ifndef __WIRELESS_H__
#define __WIRELESS_H__
const char *ssid =  "XXXXXXXXXX";   // cannot be longer than 32 characters!
const char *pass =  "XXXXXXXXX";   //
const char* mqttServerString = "XX.XX.XX.XX";
const uint16_t mqttServerPort = 8883;

#define USE_CERTS
unsigned char client_cert[] PROGMEM = {
  0x30, 0x82, 0x03, 0x92, 0x30, 0x82, 0x02, 0x7a, 0x02, 0x09, 0x00, 0xcb,
.............
};
unsigned int client_cert_len = YYYY;


unsigned char client_key[] PROGMEM = {
  0x30, 0x82, 0x04, 0xa5, 0x02, 0x01, 0x00, 0x02, 0x82, 0x01, 0x01, 0x00,
 .......
};
unsigned int client_key_len = ZZZZ;
#endif
```


© 2019 IBM Corporation

Anatomy of a Simple Device

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// device specifics
#include "WirelessConfig.h"
#include "SensorConfig.h"

#define TRANSMIT_INTERVAL_SECONDS 60
#define MILLIS_IN_SECOND 1000
#define LOOP_DELAY 100

#define LED_PIN D3
#define LED_BLINK_TIME_SECONDS 2

#define MAX_MESSAGE_SIZE 100

#define LIGHT_PIN A0

#define DS18B20_PIN D4  // don't use D0 or D2 as can interfere with boot
OneWire ds(DS18B20_PIN);
DallasTemperature tempSensors(&ds);
```


© 2019 IBM Corporation

Anatomy of a Simple Device

```
31  bool ledOn = true;
32  void toggleLED() {
33    if (ledOn) {
34      ledOn = false;
35      digitalWrite(LED_PIN, LOW);
36    } else {
37      ledOn  = true;
38      digitalWrite(LED_PIN, HIGH);
39    }
40  }
41
42  void callback(char* topic, uint8_t* message, unsigned int length) {
43    if (strncmp((const char*)message,"on", strlen("on")) == 0) {
44      digitalWrite(LED_PIN, HIGH);
45      ledOn = true;
46    } else {
47      digitalWrite(LED_PIN, LOW);
48      ledOn = false;
49    }
50  };
```


© 2019 IBM Corporation

Anatomy of a Simple Device

```
52  ESP8266WiFiGenericClass wifi;
53
54  #ifdef USE_CERTS
55  // if certs are used the following must be defined in WirelessConfig.h
56  //    unsigned char client_cert[] PROGMEM = {bytes in DER format};
57  //    unsigned int client_cert_len = 918;
58  //    unsigned char client_key[] PROGMEM = {bytes in DER format};
59  //    unsigned int client_key_len = 1193;
60  //
61  //    conversion can be done using
62  //    openssl x509 -in cert -out client.cert -outform DER
63  //    openssl rsa -in key -out client.key -outform DER
64  //    and then using xxd to generate the required array and lengths
65  //    see https://nofurtherquestions.wordpress.com/2016/03/14/making-an-esp8266-web-accessible/
66  //    for more detailed info
67  WiFiClientSecure wclient;
68  #else
69  WiFiClient wclient;
70  #endif
71
72  PubSubClient client(mqttServerString, mqttServerPort, callback, wclient);
73
74  int counter = 0;
75  char macAddress[] = "00:00:00:00:00:00";
```

Runs when MQTT message received

© 2019 IBM Corporation

Anatomy of a Simple Device

```
77  void setup() {
78    delay(1000);
79
80    // Setup console
81    Serial.begin(115200);
82    delay(10);
83    Serial.println();
84    Serial.println("Started");
85
86    pinMode(LED_PIN, OUTPUT);
87    digitalWrite(LED_PIN, HIGH);
88
89  #ifdef USE_CERTS
90    wclient.setCertificate_P(client_cert, client_cert_len);
91    wclient.setPrivateKey_P(client_key, client_key_len);
92  #endif
93
94    // turn of the Access Point as we are not using it
95    wifi.mode(WIFI_STA);
96    WiFi.begin(ssid, pass);
97
98    // first reading always seems to be wrong, read it early and
99    // throw it away
100   tempSensors.requestTemperatures();
101
102    // get the mac address to be used as a unique id for connecting to the mqtt server
103   byte macRaw[6];
104   WiFi.macAddress(macRaw);
105   sprintf(macAddress,
106           "%02.2X:%02.2X:%02.2X:%02.2X:%02.2X:%02.2X",
107           macRaw[0],
108           macRaw[1],
109           macRaw[2],
110           macRaw[3],
111           macRaw[4],
112           macRaw[5]);
113 }
```


© 2019 IBM Corporation

Anatomy of a Simple Device

```
115  void loop() {
116    client.loop();
117    delay(LOOP_DELAY);
118
119    // make sure we are good for wifi
120    if (WiFi.status() != WL_CONNECTED) {
121      Serial.print("Connecting to ");
122      Serial.println(ssid);
123      WiFi.reconnect();
124
125      if (WiFi.waitForConnectResult() != WL_CONNECTED) {
126        Serial.println("Failed to reconnect WIFI");
127        Serial.println(WiFi.waitForConnectResult());
128        delay(1000);
129        return;
130      }
131    }
132
133
134    if (!client.connected()) {
135      if (client.connect(macAddress)) {
136        Serial.println("mqtt connected:");
137        Serial.println(macAddress);
138        Serial.println("\n");
139        client.subscribe(LED_TOPIC);
140      }
141    }
```

Make sure ID is Unique !

Subscribe to topics of interest

© 2019 IBM Corporation

Anatomy of a Simple Device

```
143     counter++;
144     if (counter == (TRANSMIT_INTERVAL_SECONDS * (MILLIS_IN_SECOND/LOOP_DELAY))) {
145       Serial.println("Sending");
146
147       // don't send out temperature too often as we'll get
148       // incorrect values if we sample too often
149       char tempMessage[MAX_MESSAGE_SIZE];
150       char floatBuffer[10];
151       tempSensors.requestTemperatures();
152       float currentTemp = tempSensors.getTempCByIndex(0);
153       snprintf(tempMessage, MAX_MESSAGE_SIZE, "0, 0 - temp: %s",
154                dtostrf(currentTemp, 4, 2, floatBuffer));
155       client.publish(TEMP_TOPIC, tempMessage);
156
157       char lightMessage[MAX_MESSAGE_SIZE];
158       int lightValue = analogRead(LIGHT_PIN);
159       snprintf(lightMessage, MAX_MESSAGE_SIZE, "0, 0 - light: %d", lightValue);
160       client.publish(LIGHT_TOPIC, lightMessage);
161
162       toggleLED();
163       counter = 0;
164     } else if (counter == (LED_BLINK_TIME_SECONDS * (MILLIS_IN_SECOND/LOOP_DELAY))) {
165       toggleLED();
166     }
167   }
```

© 2019 IBM Corporation

Anatomy of Simple Device

- Don’t like C++? Can use JavaScript as well
  - https://github.com/mhdawson/espruino-stuff/blob/master/SmartPlug.js

```
client.on('publish', function(message) {
  console.log(message);
  if (message.topic === (devicePrefix + '/power')) {
    if (message.message === 'on') {
      powerState = 1;
    } else if (message.message === 'off') {
      powerState = 0;
    }
    digitalWrite(powerPin, powerState);
    console.log('Power state:' + powerState);
  } else if (message.topic === (devicePrefix + '/led')) {
    clearLedFlashTimer();
    if (message.message === 'on') {
      ledState = 1;
    } else if (message.message === 'off') {
      ledState = 0;
    } else if (message.message.substr(0, 'flash'.length) === 'flash') {
      try {
        timeout = message.message.split(':')[1];
        startFlashTimer(timeout);
      } catch (err) {
        console.log(err);
      }
    }
    digitalWrite(ledPin, (ledState + 1) % 2);
    console.log('Led state:' + ledState);
  } else if (message.topic === (devicePrefix + '/query_state')) {
    client.publish(devicePrefix + '/state/power', powerState);
    client.publish(devicePrefix + '/state/led', ledState);
  }
});
```

© 2019 IBM Corporation

Consuming Sensor Data on IBM i (client-sensor.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// setup mqtt
let  mqttOptions;
mqttOptions = {
                key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
                cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
                ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
                clientId: 'simple-client',
                checkServerIdentity: function() { return undefined },
                rejectUnauthorized: false,
                username: '',
                password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
  mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.subscribe('factory/1/light');
    mqttClient.subscribe('factory/1/temp');
    mqttClient.on('message', (topic, message) => {
      console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
    });
});
```


© 2019 IBM Corporation

Controlling Sensor on IBMi (control.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');


// setup mqtt
let  mqttOptions;
mqttOptions = {
                key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
                cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
                ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
                clientId: 'control client',
                checkServerIdentity: function() { return undefined },
                rejectUnauthorized: false,
                username: '',
                password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
  console.log('connected');
  mqttClient.publish('factory/1/led', process.argv[2], () => {
    setTimeout( () => {
      process.exit(0);
    }, 2000);
  });
});
```


© 2019 IBM Corporation

Consuming Sensor Data on IBM i – Store to DB

```
const { DBPool } = require('idb-pconnector');
const pool = new DBPool();

async function setupDb() {
  try {
    await pool.prepareExecute('CREATE SCHEMA JESSEGIOT');
  }catch(err) {
    if(err.stack.includes('SQLSTATE=42710')) {
        console.log('schema already exists');
    } else {
      console.log('error: '+err.stack);
    }
  }
  try {
    await pool.prepareExecute(`CREATE OR REPLACE TABLE JESSEGIOT.IOT_RECORDS (
                                    DEVICE VARCHAR(80) ALLOCATE (10) CCSID 1208 NOT NORMALIZED NOT NULL NOT HIDDEN,
                                    SENSORVALUE DECIMAL(7, 2) NOT NULL NOT HIDDEN,
                                    SENSORTIME TIMESTAMP(6) GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
                                    NOT NULL NOT HIDDEN
                                )
                                NOT VOLATILE UNIT ANY KEEP IN MEMORY NO`);
  }catch(err) {
      console.log('error: '+err.stack);
  }
  console.log('Database setup complete!');
}
```

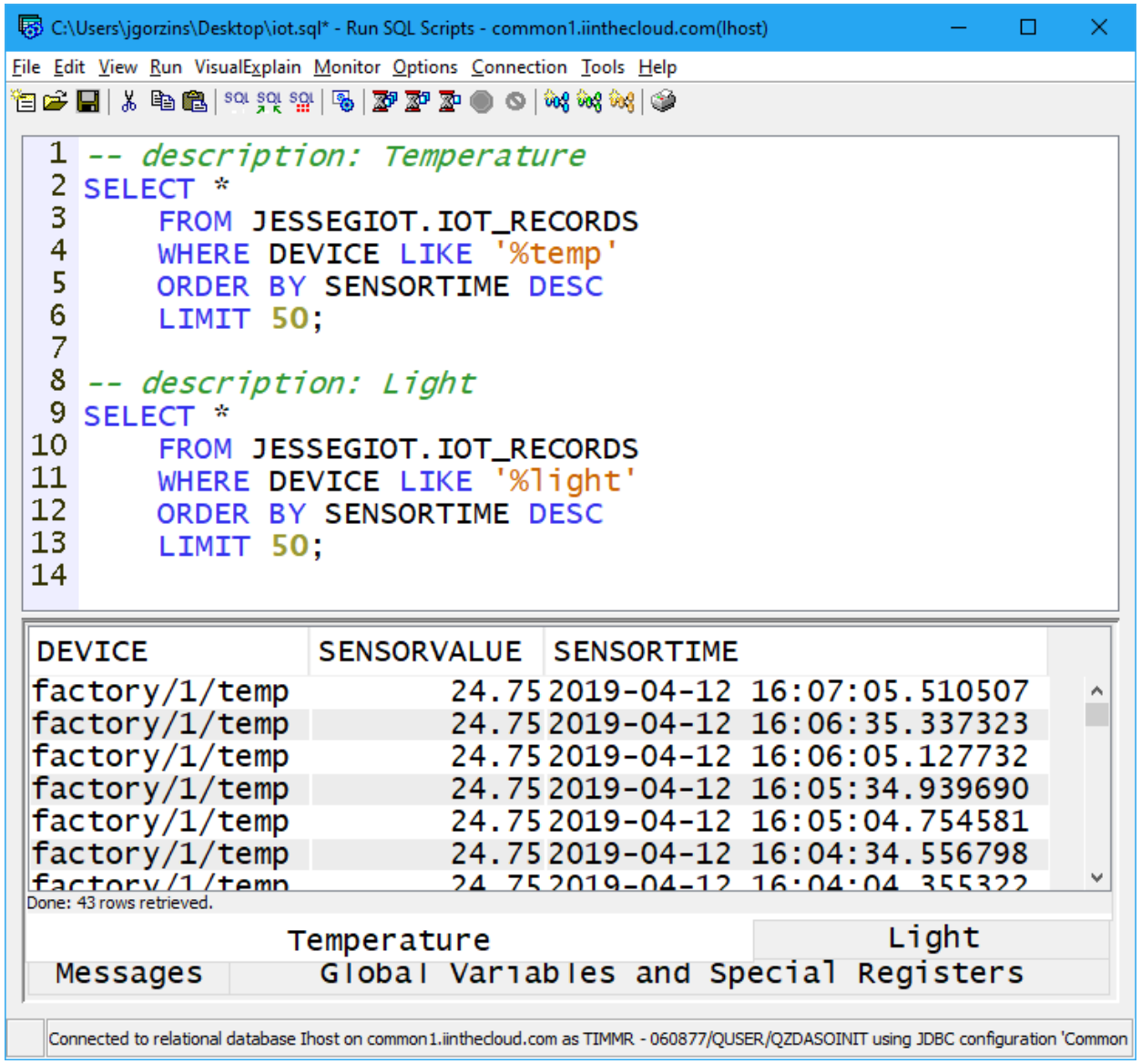

© 2019 IBM Corporation

Consuming Sensor Data on IBM i – Store to DB

```
const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions); mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.subscribe('factory/1/light');
    mqttClient.subscribe('factory/1/temp');
    mqttClient.on('message', (topic, message) => {
      let value = message.toString().replace(/.*:/g,'').replace(/[^0-9.]+/g,'');
      pool.prepareExecute('insert into JESSEGIOT.IOT_RECORDS(device, sensorvalue) values(?, ?)', [topic, value]);
      console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
    });
});
```


© 2019 IBM Corporation

Controlling Sensor on IBM i (control.js)



© 2019 IBM Corporation

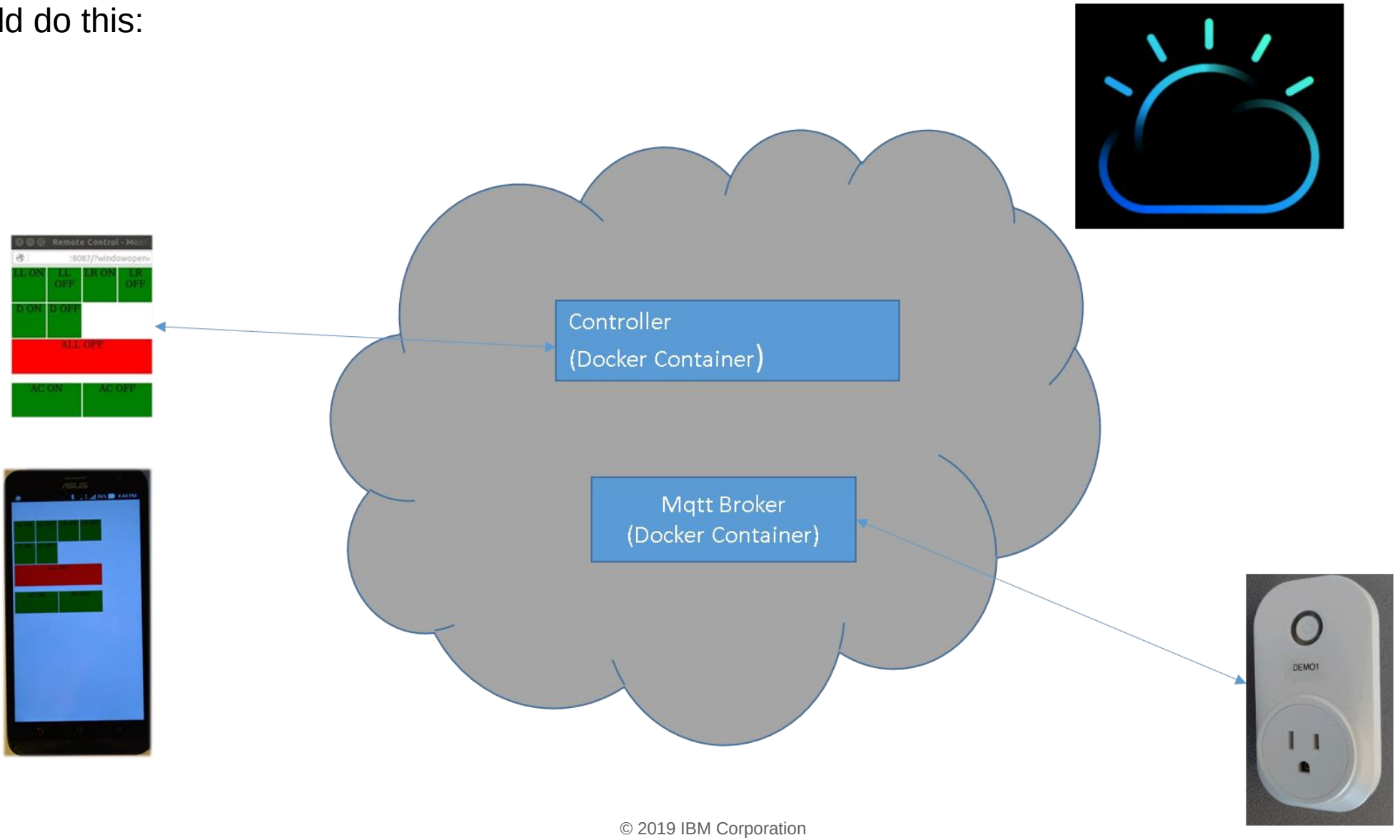
Demo

- Live Device data flow

© 2019 IBM Corporation

Leveraging the Cloud

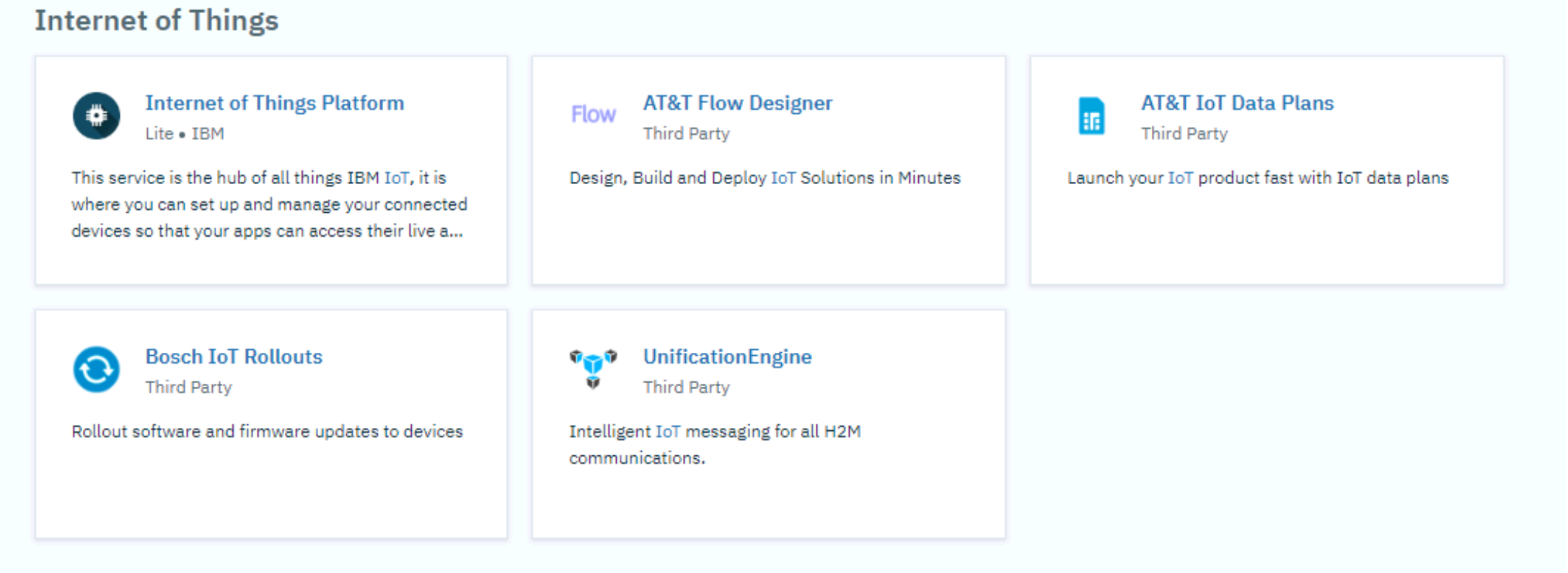
- Could do this:



© 2019 IBM Corporation

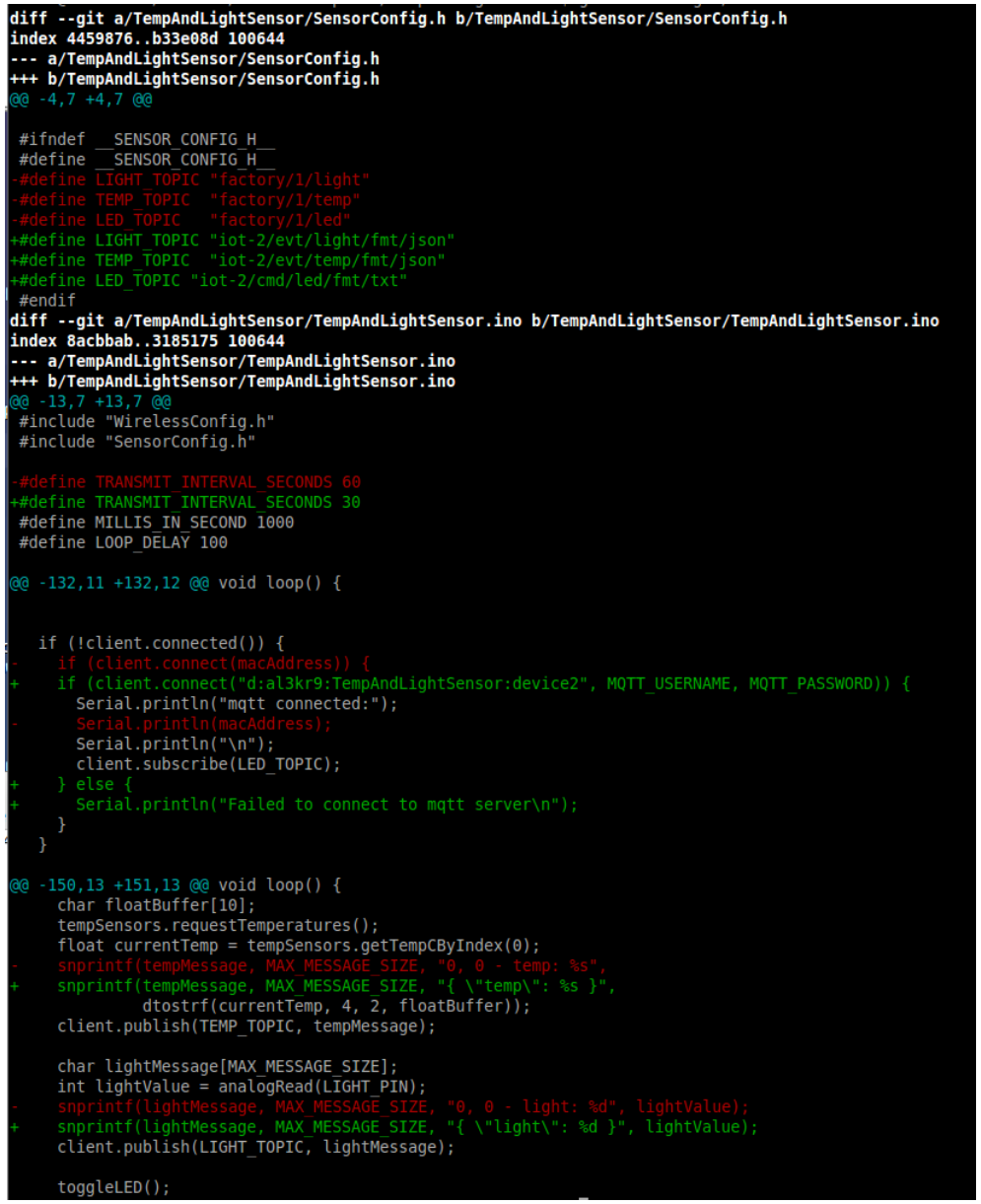
Leveraging the Cloud – Even Better

- Cloud Based Service
  - Don’t worry about infrastructure
  - Get started fast
  - Easy visualization



© 2019 IBM Corporation

Leveraging the Cloud



```
diff --git a/TempAndLightSensor/SensorConfig.h b/TempAndLightSensor/SensorConfig.h
index 4459876..b33e08d 100644
--- a/TempAndLightSensor/SensorConfig.h
+++ b/TempAndLightSensor/SensorConfig.h
@@ -4,7 +4,7 @@

 #ifndef __SENSOR_CONFIG_H__
 #define __SENSOR_CONFIG_H__
-#define LIGHT_TOPIC "factory/1/light"
-#define TEMP_TOPIC  "factory/1/temp"
-#define LED_TOPIC   "factory/1/led"
+#define LIGHT_TOPIC "iot-2/evt/light/fmt/json"
+#define TEMP_TOPIC  "iot-2/evt/temp/fmt/json"
+#define LED_TOPIC "iot-2/cmd/led/fmt/txt"
 #endif
diff --git a/TempAndLightSensor/TempAndLightSensor.ino b/TempAndLightSensor/TempAndLightSensor.ino
index 8acbbab..3185175 100644
--- a/TempAndLightSensor/TempAndLightSensor.ino
+++ b/TempAndLightSensor/TempAndLightSensor.ino
@@ -13,7 +13,7 @@
 #include "WirelessConfig.h"
 #include "SensorConfig.h"

-#define TRANSMIT_INTERVAL_SECONDS 60
+#define TRANSMIT_INTERVAL_SECONDS 30
 #define MILLIS_IN_SECOND 1000
 #define LOOP_DELAY 100

@@ -132,11 +132,12 @@ void loop() {


   if (!client.connected()) {
-    if (client.connect(macAddress)) {
+    if (client.connect("d:al3kr9:TempAndLightSensor:device2", MQTT_USERNAME, MQTT_PASSWORD)) {
       Serial.println("mqtt connected:");
-      Serial.println(macAddress);
       Serial.println("\n");
       client.subscribe(LED_TOPIC);
+    } else {
+      Serial.println("Failed to connect to mqtt server\n");
     }
   }

@@ -150,13 +151,13 @@ void loop() {
     char floatBuffer[10];
     tempSensors.requestTemperatures();
     float currentTemp = tempSensors.getTempCByIndex(0);
-    snprintf(tempMessage, MAX_MESSAGE_SIZE, "0, 0 - temp: %s",
+    snprintf(tempMessage, MAX_MESSAGE_SIZE, "{ \"temp\": %s }",
             dtostrf(currentTemp, 4, 2, floatBuffer));
     client.publish(TEMP_TOPIC, tempMessage);

     char lightMessage[MAX_MESSAGE_SIZE];
     int lightValue = analogRead(LIGHT_PIN);
-    snprintf(lightMessage, MAX_MESSAGE_SIZE, "0, 0 - light: %d", lightValue);
+    snprintf(lightMessage, MAX_MESSAGE_SIZE, "{ \"light\": %d }", lightValue);
     client.publish(LIGHT_TOPIC, lightMessage);

     toggleLED();
```

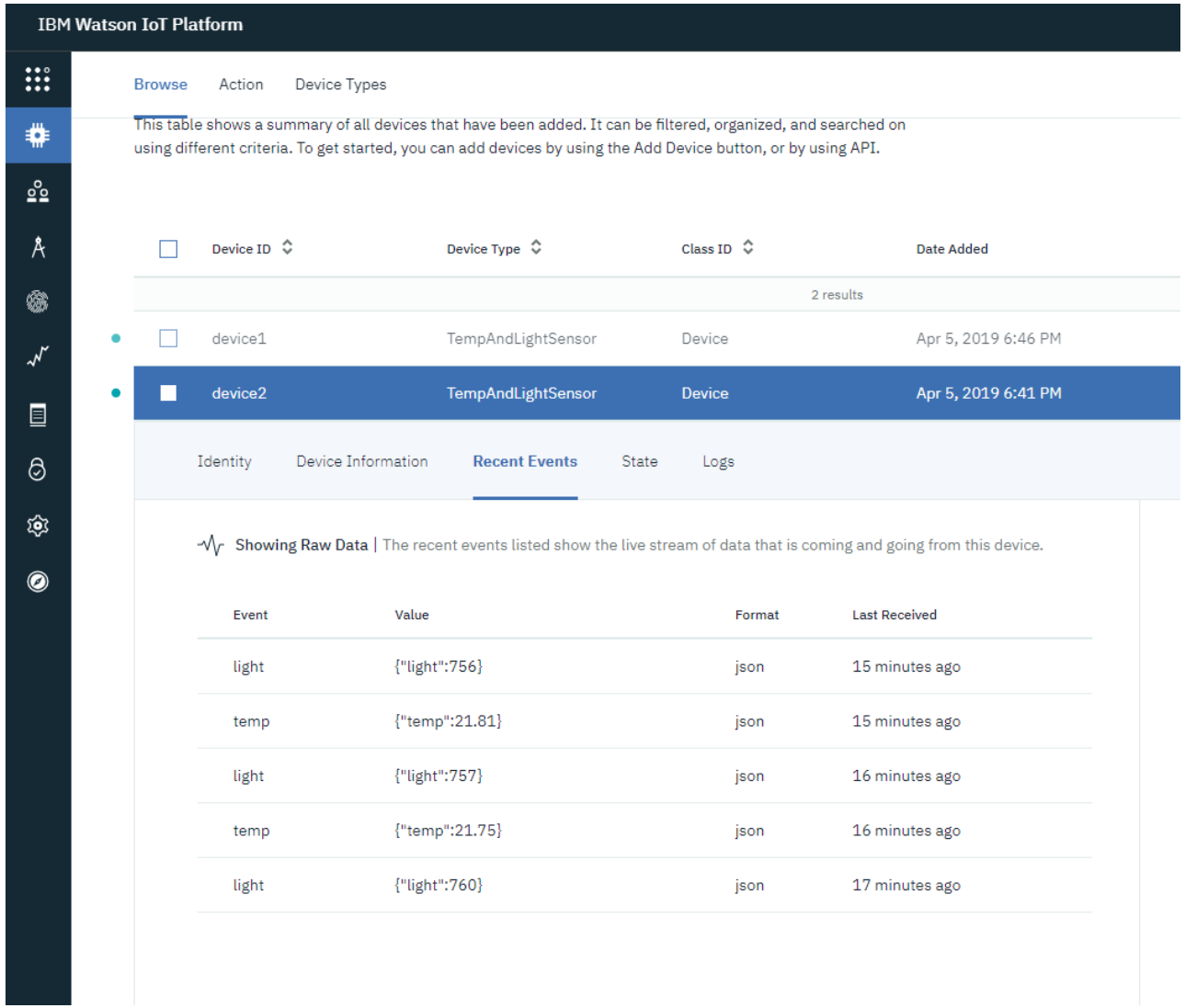
https://cloud.ibm.com/docs/services/IoT/reference/security?topic=iot-platform-connect_devices_apps_gw#connect_devices_apps_gw

```
-#define LIGHT_TOPIC "factory/1/light"
-#define TEMP_TOPIC  "factory/1/temp"
-#define LED_TOPIC   "factory/1/led"
+#define LIGHT_TOPIC "iot-2/evt/light/fmt/json"
+#define TEMP_TOPIC  "iot-2/evt/temp/fmt/json"
+#define LED_TOPIC "iot-2/cmd/led/fmt/txt"
```

```
-    snprintf(lightMessage, MAX_MESSAGE_SIZE, "0, 0 - light: %d", lightValue);
+    snprintf(lightMessage, MAX_MESSAGE_SIZE, "{ \"light\": %d }", lightValue);
```

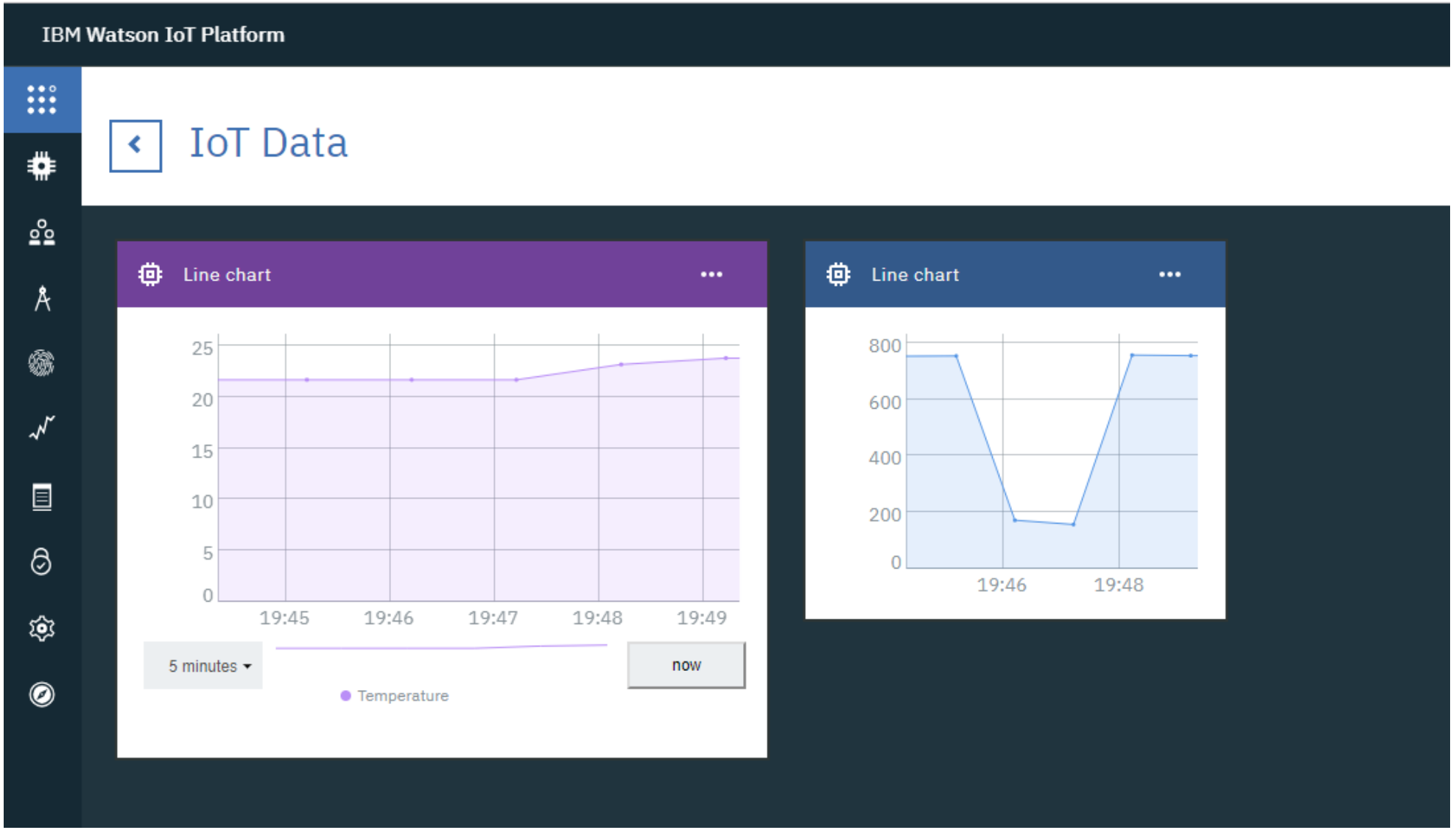

© 2019 IBM Corporation

Leveraging the Cloud



© 2019 IBM Corporation

Leveraging the Cloud



© 2019 IBM Corporation

Consuming Data on IBMi (client-ibmcloud.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');


// setup mqtt
let  mqttOptions;
mqttOptions = {
                key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
                cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
                ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
                clientId: 'A:XXXXXX:XXXXXXXXXX',
                checkServerIdentity: function() { return undefined },
                rejectUnauthorized: false,
                username: 'a-XXXXXX-XXXXXXXXXX',
                password: 'XXXXXXXXXXXXXXXXX'
}

const mqttClient = mqtt.connect('mqtts:XXXXXX.messaging.internetofthings.ibmcloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
  console.log('connected');
  mqttClient.subscribe('iot-2/type/+/id/+/evt/+/fmt/+');
  mqttClient.on('message', (topic, message) => {
    console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
  });
});
```

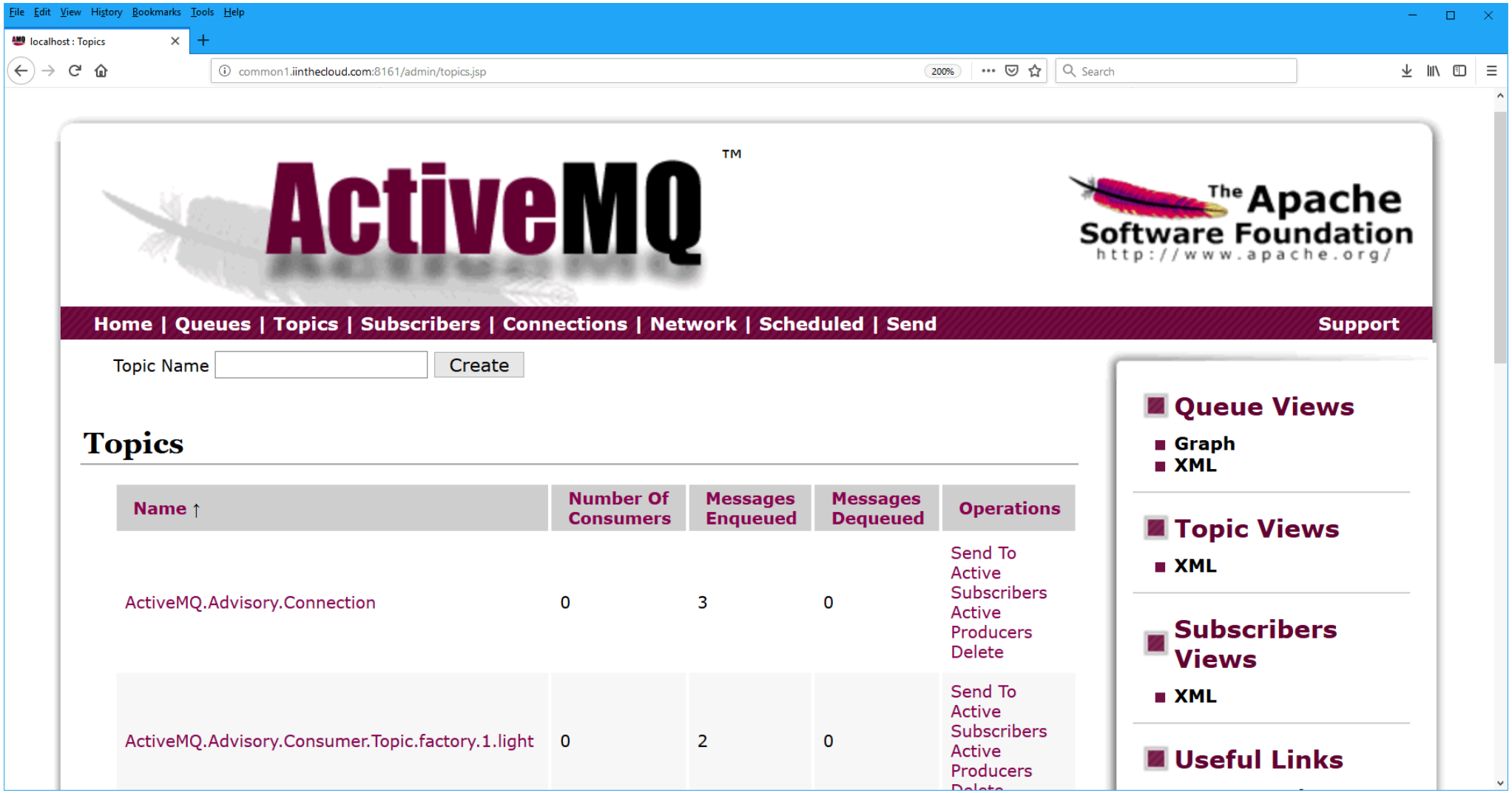

© 2019 IBM Corporation

Summary

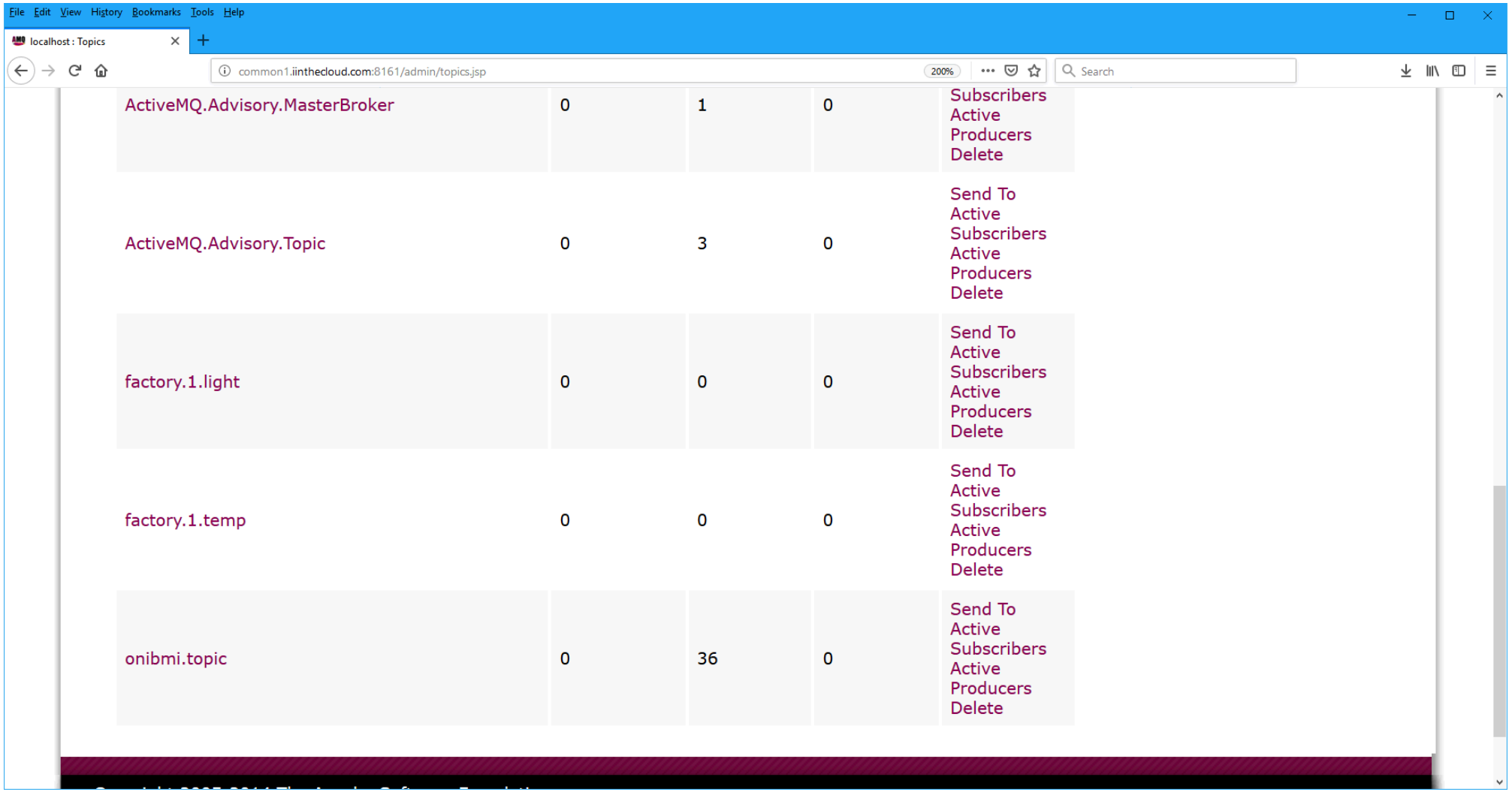
- Intro to IoT and MQTT
- Using MQTT with Node.js and IBM i
- Lets Look at some devices
- Anatomy of a simple MQTT Light and Temperature Sensor
- Consuming MQTT data on IBMi
- Leveraging the Cloud

© 2019 IBM Corporation

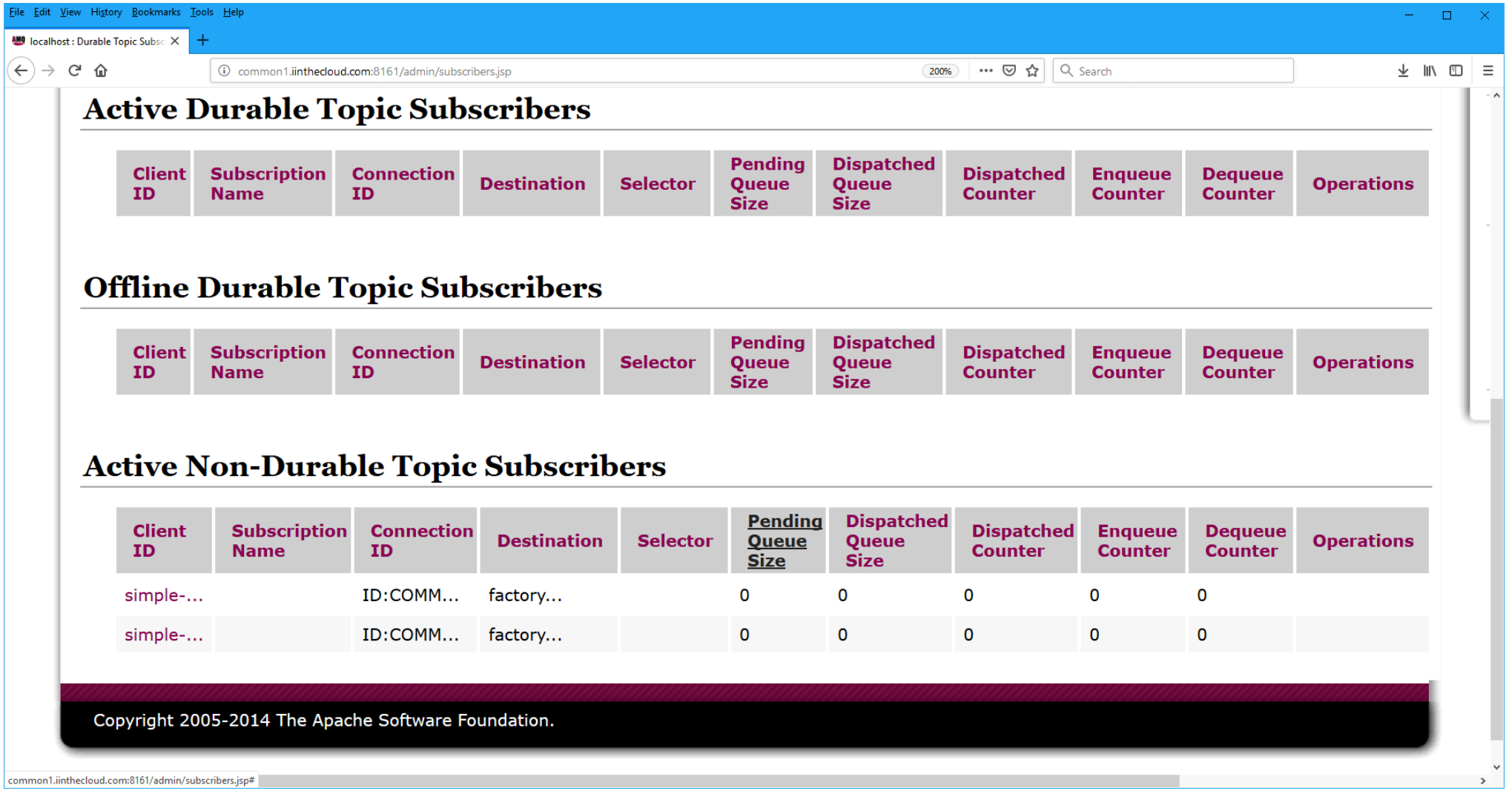
© 2019 IBM Corporation



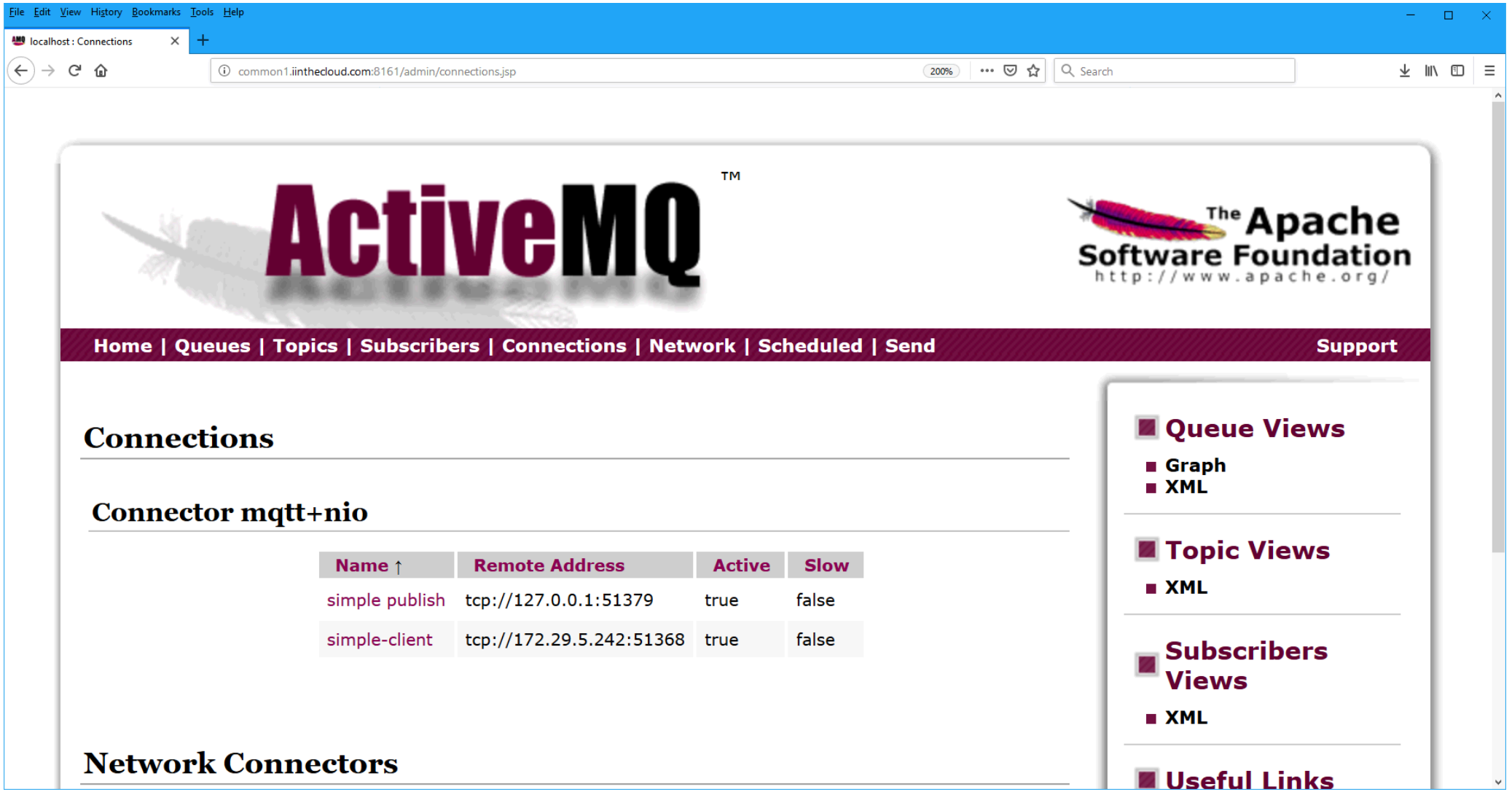
© 2019 IBM Corporation



© 2019 IBM Corporation



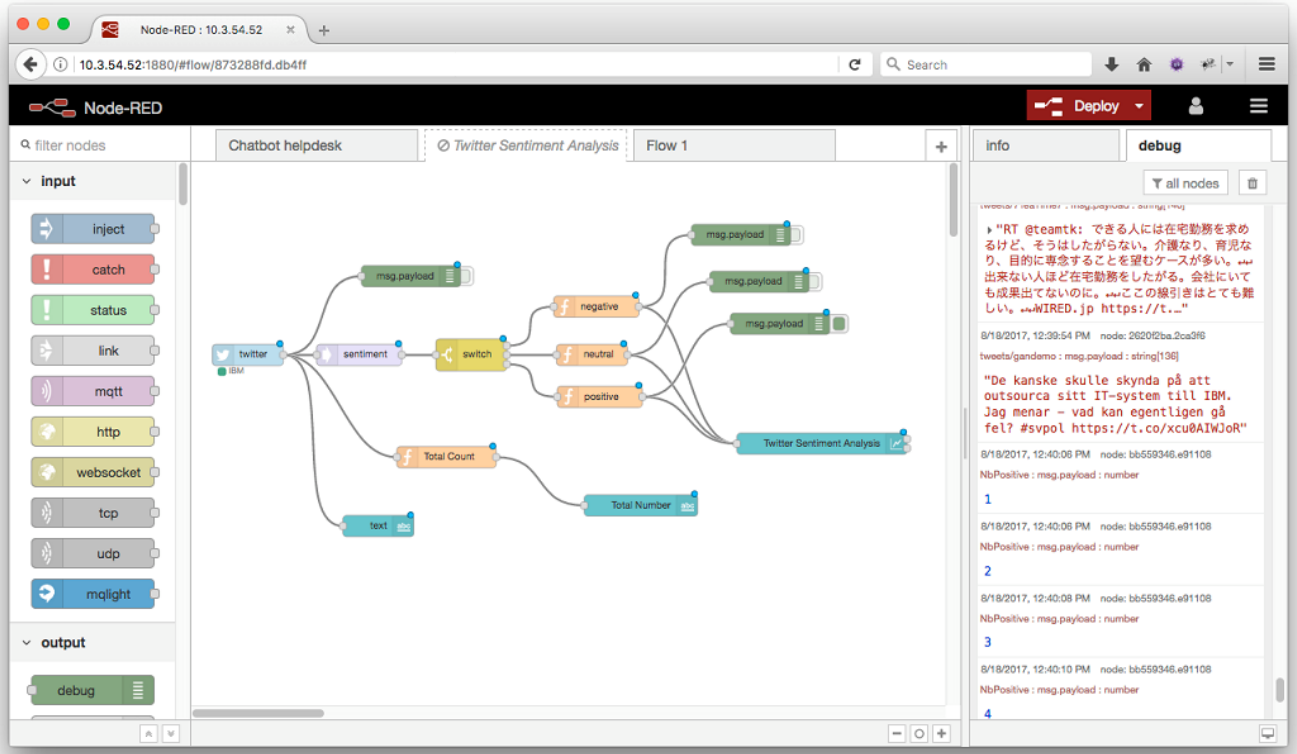
© 2019 IBM Corporation



© 2019 IBM Corporation

Node-RED

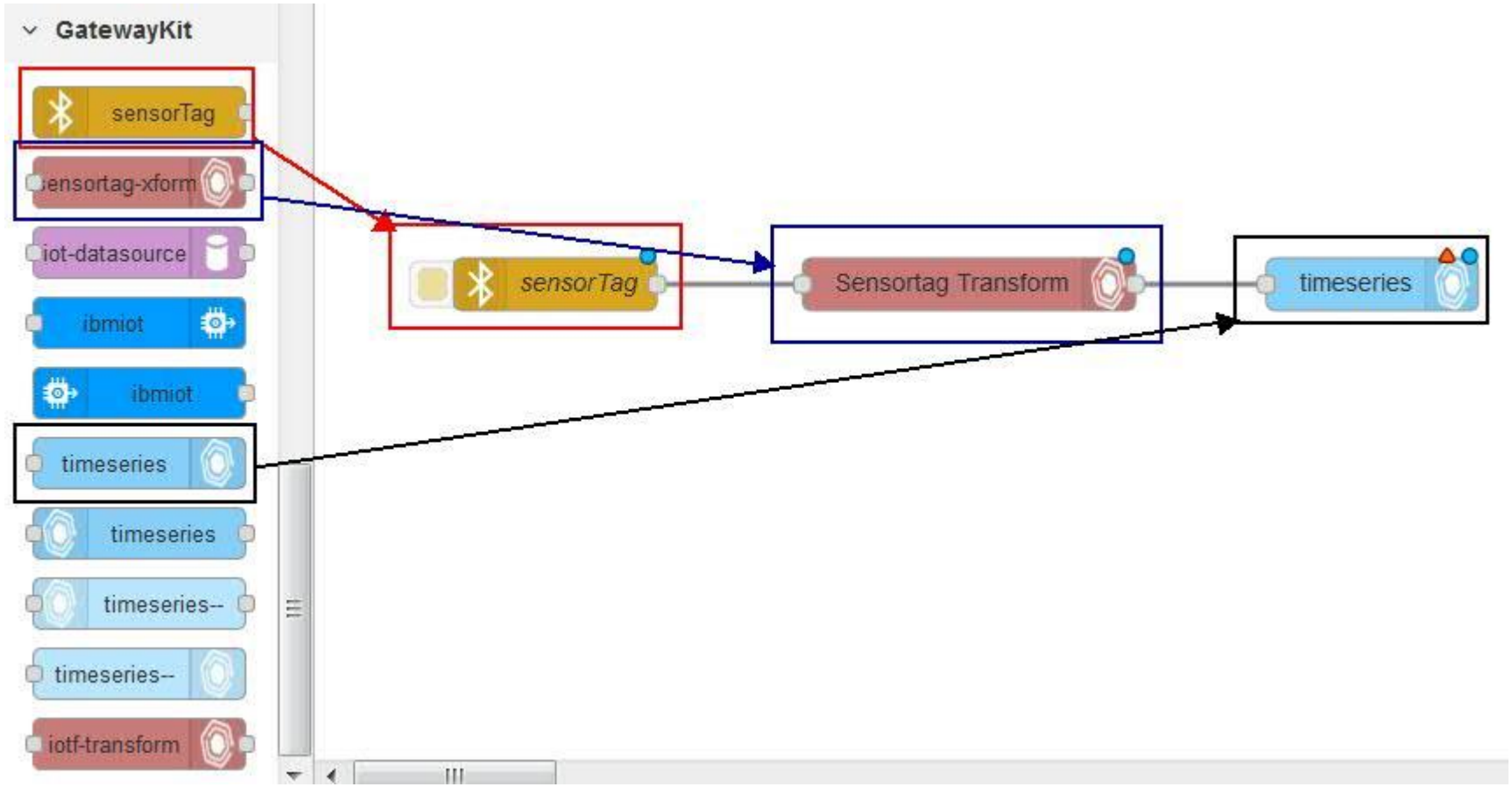
- Created by IBM
- Low-code graphical way to write flow-based programs
- https://developer.ibm.com/tutorials/i-running-node-red/
- Nodes can be any building block that can receive and send messages. Examples include:
  - Db2 for i queries
  - IoT devices
  - Web pages
  - Web APIs
  - Cloud services
  - Dashboards



© 2019 IBM Corporation

Node-RED for IoT

- https://www.ibm.com/developerworks/community/blogs/cee6c09c-a315-4b04-ad14-57d6a60fa8bb/entry/Creating_A_Node_red_Flow_for_IoT?lang=en



© 2019 IBM Corporation

Q&A

The END!

© 2019 IBM Corporation

Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Revised September 26, 2006

© 2019 IBM Corporation

Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, Db2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, Db2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: http://www.**ibm.com**/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
AltiVec is a trademark of Freescale Semiconductor, Inc.
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.
Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.
NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
UNIX is a registered trademark of The Open Group in the United States, other countries or both.
•Node.js is an official trademark of Joyent. IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project..
•“TWITTER, TWEET, RETWEET and the Twitter logo are trademarks of Twitter, Inc. or its affiliates.”

Revised December 2, 2010

Other company, product and service names may be trademarks or service marks of others.

© 2019 IBM Corporation