



Discover. Collaborate. Deploy.

# Talk to your Code

Michael Dawson

# Please Note

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

# About **Michael Dawson**

**IBM Runtimes/IBM Node.js Community Lead**

**Node.js collaborator**

**Chair of Technical Steering Committee (TSC)**

**Community Committee member**

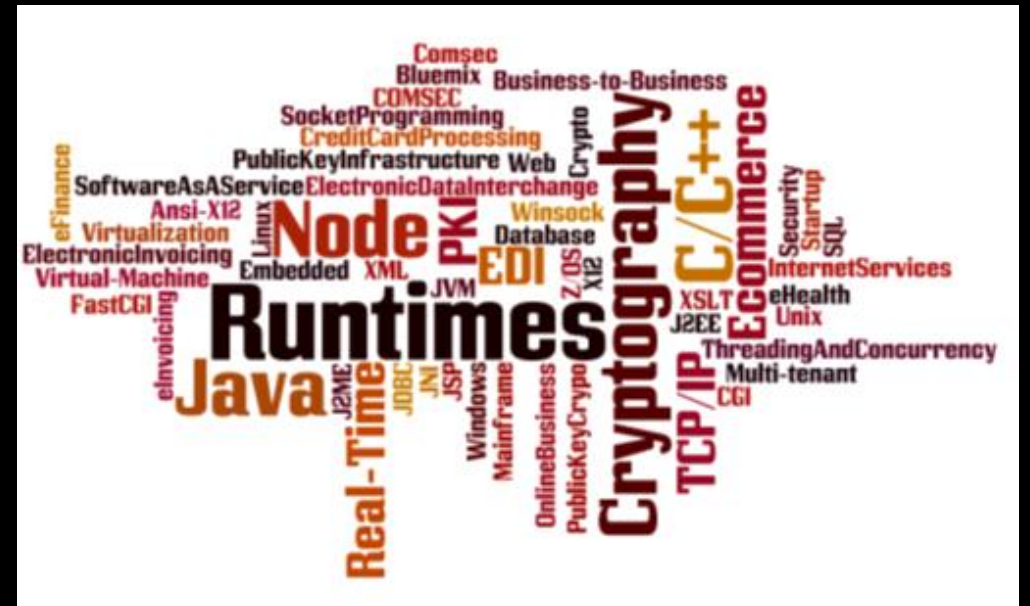
**Active in many working groups/teams**

- Benchmarking
- Build
- Release
- N-API
- Diagnostics
- Security-wg
- User Feedback


**Twitter: @mhdawson1**

**GitHub: @mhdawson**

**Linkedin: <https://www.linkedin.com/in/michael-dawson-6051282>**



# Overview

-  ❤️ IoT
- Interaction V1
- Voice Service on the Rise
- Alexa to Mqtt Bridge
- Interaction V2
- Building an Alexa Skill
- Real world experience



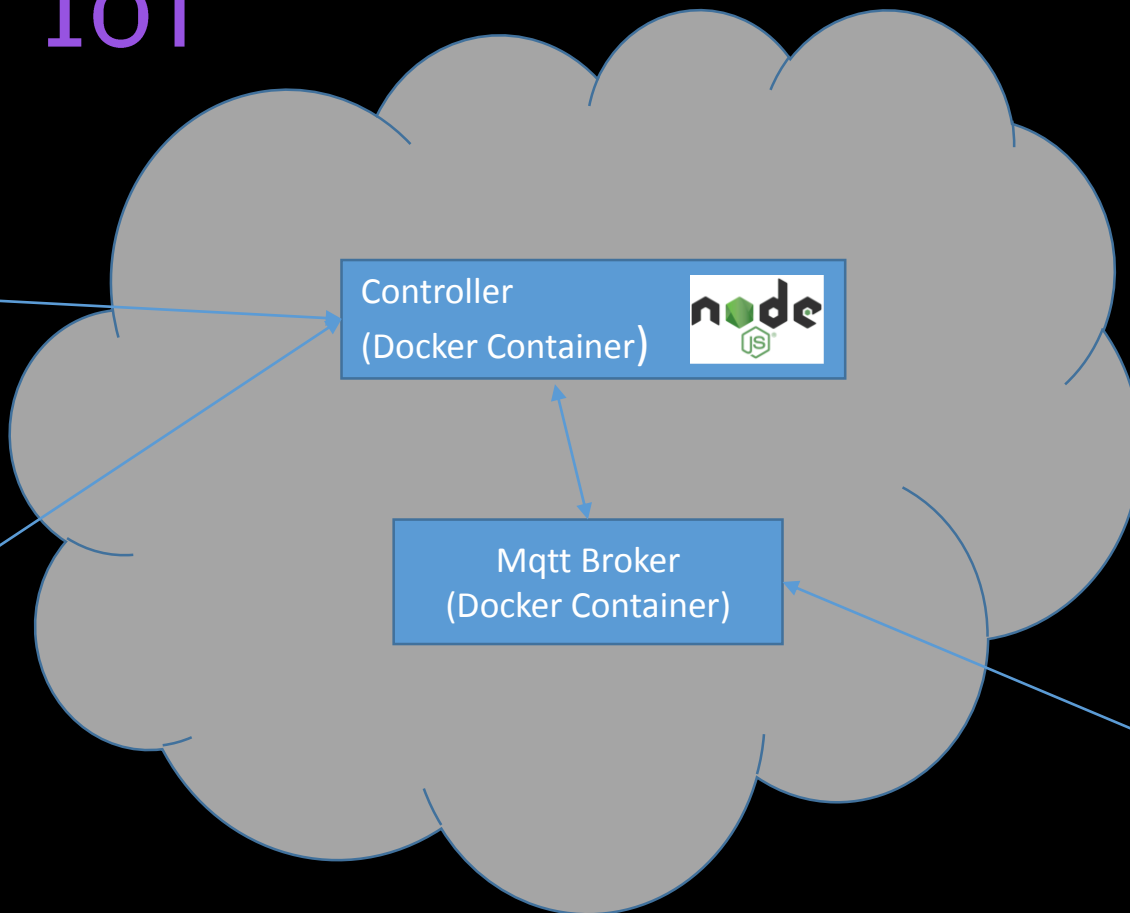
# IoT

- 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](https://en.wikipedia.org/wiki/Internet_of_Things)
- 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”





IoT





# IoT

```
289 var client = mqtt.connect(mqttServerUrl, mqttOptions);
290
291 /* each time we connect register on all topics we are interested
292  * in. This must be done after a reconnect as well as the
293  * initial connect
294  */
295 client.on('connect',function() {
296   client.subscribe(alarmStatusTopic);
297   client.subscribe(zoneTopicPrefix + '+/+');
298   client.subscribe(newPictureTopic);
299   for(topic in zoneMapping) {
300     client.subscribe(topic);
301   }
302 });
303
304 client.on('message', function(topic, message) {
305   latestData[topic] = message;
```

## Receive

```
client.publish(cameraCaptureTopic, 'take');
```

## Send

# JavaScript ♥ IoT

```
54 client.on('publish', function(message) {
55   console.log(message);
56   if (message.topic === (devicePrefix + '/power')) {
57     if (message.message === 'on') {
58       powerState = 1;
59     } else if (message.message === 'off') {
60       powerState = 0;
61     }
62     digitalWrite(powerPin, powerState);
63     console.log('Power state:' + powerState);
64   } else if (message.topic === (devicePrefix + '/led')) {
65     clearLedFlashTimer();
66     if (message.message === 'on') {
67       ledState = 1;
68     } else if (message.message === 'off') {
69       ledState = 0;
70     } else if (message.message.substr(0, 'flash'.length) === 'flash') {
71       try {
72         timeout = message.message.split(':')[1];
73         startFlashTimer(timeout);
74       } catch (err) {
75         console.log(err);
76       }
77     }
78     digitalWrite(ledPin, (ledState + 1) % 2);
79     console.log('Led state:' + ledState);
80   } else if (message.topic === (devicePrefix + '/query_state')) {
81     client.publish(devicePrefix + '/state/power', powerState);
82     client.publish(devicePrefix + '/state/led', ledState);
83   }
84 });
```

<https://github.com/mhdawson/espruino-stuff/blob/master/SmartPlug.js>

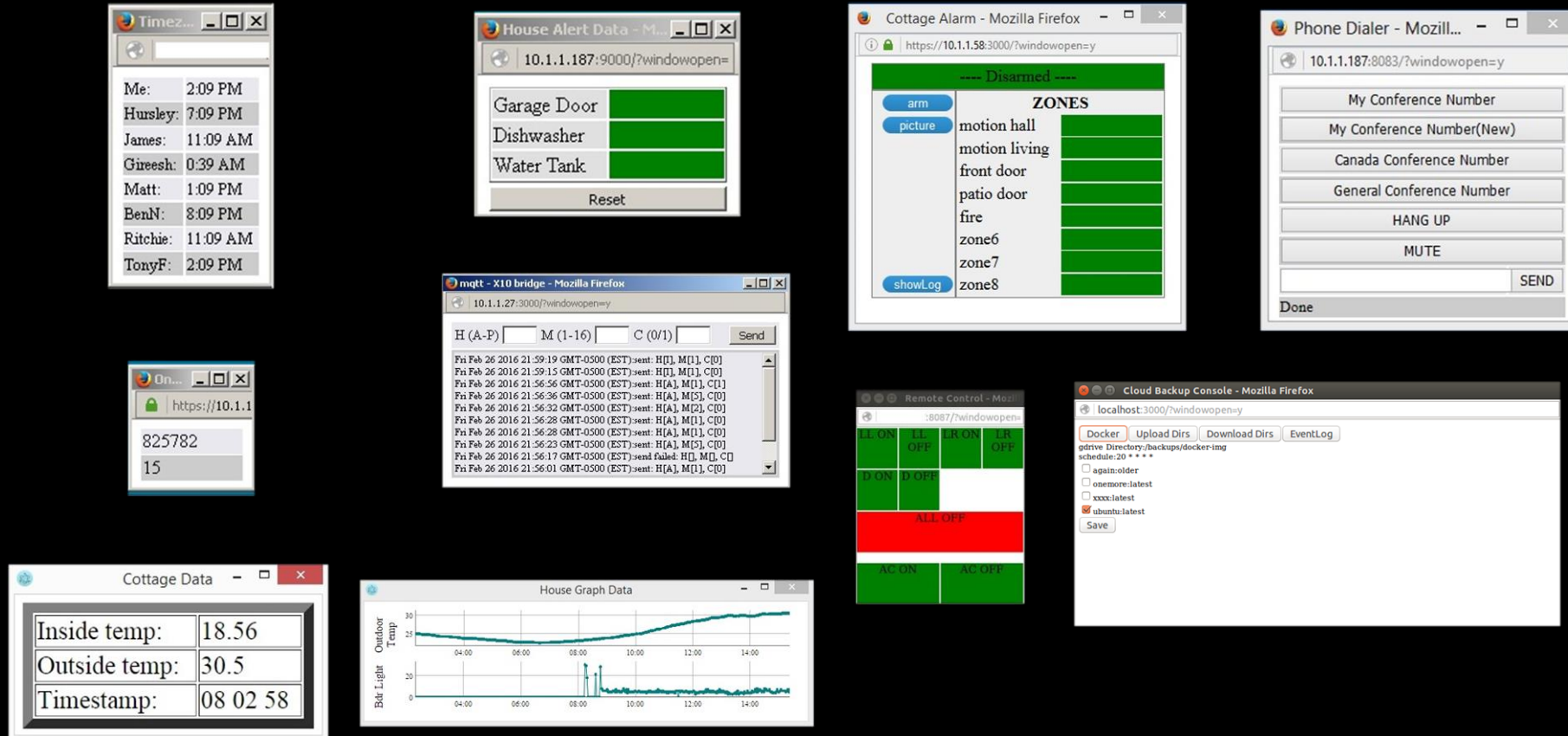




# Interaction – V1

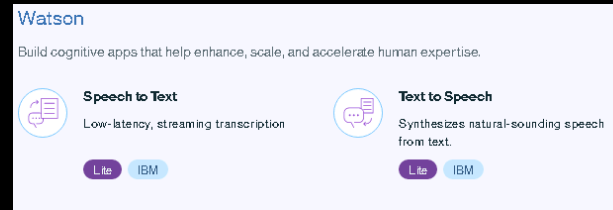
- Demo

# Good enough, create bunch of Apps



# Voice services on the Rise

- IBM Watson



<https://www.ibm.com/watson/services/speech-to-text/>

- Google Home

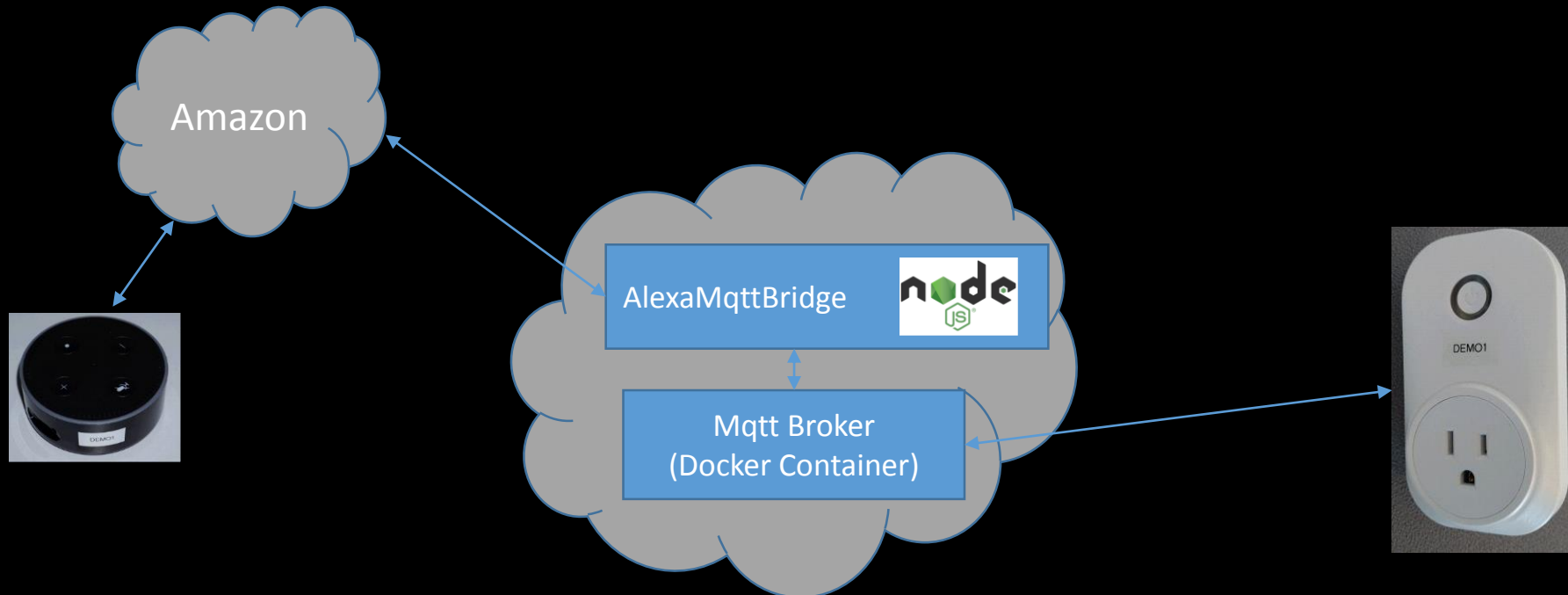
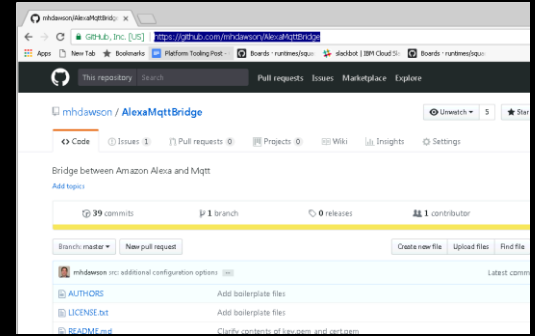


- Amazon Alexa



# Alexa to Mqtt Bridge

- Convert Voice Request to Mqtt Message (and vice-versa)
- <https://github.com/mhdawson/AlexaMqttBridge>



# Interaction V2

- Demo

# Alexa to Mqtt Bridge

- Server in 

- Alexa Skill

# Writing an Alexa Skill

- Skill Information
- Interaction Model
  - Intent
  - Slots/Slot Types
  - Sample Utterances
- Endpoint

# Skill Information

- Name
- Invocation Name

Ask **Michael** to ...

- Language (new, at least to me 😊)



# Intents

```
{
  "intents": [
    {
      "slots": [
        {
          "name": "Device",
          "type": "List_of_devices"
        }
      ],
      "intent": "TurnOff"
    },
    {
      "slots": [
        {
          "name": "Device",
          "type": "List_of_devices"
        }
      ],
      "intent": "TurnOn"
    },
    ...
  ]
}
```

## Intent Schema

The schema of user intents in JSON format. For more information, see [Intent Schema](#).  
Also see [built-in slots](#) and [built-in intents](#).

```
1 {
2   "intents": [
3     {
4       "slots": [
5         {
6           "name": "Device",
7           "type": "List_of_devices"
8         }
9       ],
10      "intent": "TurnOff"
11    }
12  ]
13 }
```

# Slot Types

- List of options for Slot
  - Does not have to be exhaustive
  - Helps recognition
- Built in types as well
  - AMAZON.TVSeries, AMAZON.NUMBER
- Custom example:
  - List\_of\_devices

```
living room lights
living room right
living room left
dining room
power monitor
TV
TV 1
alarm
andrew
office light
fan
fan low
fan hi
fan medium
media1
Tv1
Socket
socketlight
```

# Sample Utterances

- Samples of expected speech patterns

TurnOff Turn off {Device}  
TurnOff to Turn off {Device}  
TurnOn Turn on {Device}  
TurnOn to Turn on {Device}  
Tune {Device} to channel {Channel}  
Tune {Device} channel {Channel}  
Tune set {Device} channel {Channel}  
Tune set {Device} to channel {Channel}  
Tune to set {Device} channel {Channel}  
Tune to set {Device} to channel {Channel}  
VolumeUp {Device} volume up {Repeat}  
VolumeUp to turn {Device} volume up {Repeat}  
VolumeUp turn {Device} volume up {Repeat}  
VolumeDown {Device} volume down {Repeat}  
VolumeDown to turn {Device} volume down {Repeat}  
VolumeDown turn {Device} volume down {Repeat}  
Mute mute {Device}  
Pause pause {Device}  
UnPause unpause {Device}  
Stop stop {Device}  
Seek seek {Device} to {Time}

# Endpoint

- [AWS Lamda](#)
- [Microservice](#)
  - <https://alexademo.devrus.com/alexa?XXXXXXXX>
  - Must be SSL
    - Trusted cert authority
      - Let's Encrypt™ certificate - <https://letsencrypt.org/>
      - CloudFlare
      - Cloud provider (ex IBM Cloud)
    - Upload self-signed

# Server Config

```
{
  "logging": false,
  "port": 5000,
  "terminateSessionDefault": true,
  "url": "/alexa?XXXXXXXXXXXXXXXXXXXX",
  "intents": { "TurnOn": { "alarm": { "topic": "house/alarm/control", "message": "arm" },
    "livingroomright": { "topic": "house/x10", "message": "A,1,1" },
    "diningroom": { "topic": "house/x10", "message": "A,5,1" },
    "officelight": { "topic": "home/2272/200", "message": "0F0FFFFFF0101" },
    "ac": { "topic": "home/2272", "message": "0F0FFFFFF0110" },
    "socket": { "topic": "house/esp2/power", "message": "on" },
    "socketlight": { "topic": "house/esp2/led", "message": "on" }
  },
    "TurnOff": { "livingroomlights": [ { "topic": "house/x10", "message": "A,5,0" },
    { "topic": "house/x10", "message": "A,1,0:1000" },
    { "topic": "house/x10", "message": "A,2,0:2000" } ],
    "livingroomright": { "topic": "house/x10", "message": "A,1,0" },
    "diningroom": { "topic": "house/x10", "message": "A,5,0" },
    "officelight": { "topic": "home/2272/200", "message": "0F0FFFFFF0110" },
    "socket": { "topic": "house/esp2/power", "message": "off" },
    "socketlight": { "topic": "house/esp2/led", "message": "off" }
  },

  "WhatsNew": { "tv": { "topic": "house/dlnaplay/control",
    "message": "whatsnew",
    "responseTopic": "house/dlnaplay/response" } },

  "Recipe": { "default": { "topic": "pdfViewer/request",
    "message": "${slots.RecipeNames.value}" } },

  },
  "mqtt": { "serverUrl": "mqtt:XXXX:1883"
  },
  "mqttExternal": { "serverUrl": "mqttts:XXXXX:8883"
  }
}
```

# Server

```
56  const requestHandler = (request, response) => {
57    var respondImmediately = true;
58    var responseData = { "version": "1.0",
59                        "response": {
60                          "outputSpeech": {
61                            "type": "PlainText",
62                            "text": "ok"
63                          },
64                          "shouldEndSession": true
65                        }
66    };
67
68    if (config.terminateSessionDefault === false ) {
69      responseData.response.shouldEndSession = false;
70    }
71
72    var requestData = '';
73    request.on('data', function(chunk) {
74      requestData = requestData + chunk.toString();
75    });
76
```

# Server

```
77     request.on('end', function(chunk) {
78         if (request.url !== config.url) {
79             return;
80         }
81
82         const jsonObject = JSON.parse(requestData);
83
84         // Handle Launch request
85         if (jsonObject.request.type === 'LaunchRequest') {
86             responseData.response.outputSpeech.text = "Hi, I'm Michael";
87             responseData.response.shouldEndSession = false;
88             response.writeHead(200, {'Content-Type': 'application/json;charset=UTF-8'});
89             response.end(JSON.stringify(responseData));
90             return;
91         }
92
93         // Handle SessionEndedRequest
94         if (jsonObject.request.type === 'SessionEndedRequest') {
95             response.writeHead(200, {'Content-Type': 'application/json;charset=UTF-8'});
96             response.end(JSON.stringify(responseData));
97             return;
98         }
99     });
```

# Server

```
100    // Handle IntentRequest
101    consoleWrapper.log(jsonObject);
102    consoleWrapper.log(jsonObject.request.intent);
103
104    const intent = jsonObject.request.intent;
105
106    // get the device associated with the request some intents do not
107    // have a device slot at all. In this case we expect there to be
108    // a default device entry
109    var device = 'default';
110    if ((intent.slots.Device) && (intent.slots.Device.value)) {
111        device = intent.slots.Device.value.toString().toLowerCase().replace(/'/g, '').replace(/ /g, '');
112    } else if ((intent.slots.SnapTarget) && (intent.slots.SnapTarget.value)) {
113        device = intent.slots.SnapTarget.value.toString().toLowerCase().replace(/'/g, '').replace(/ /g, '');
114    }
```



# Server

```
116     if (intent && intent.name && config.intents[intent.name]) {
117         const intentObject = config.intents[intent.name];
118         var key = intentObject[device];
119         if (key === undefined) {
120             key = intentObject['default'];
121         }
122         consoleWrapper.log(key);
123         if (key) {
124             if (Object.prototype.toString.call(key) !== '[object Array]' ) {
125                 key = [ key ];
126             }
127             try {
128                 const slots = intent.slots;
129                 for (let i = 0; i < key.length; i++) {
130                     const topic = eval('`' + key[i].topic + '`');
131                     let mqttClientHandle = mqttClient;
132                     if (key[i].server === 'external') {
133                         mqttClientHandle = mqttClientExternal;
134                     }
135                     consoleWrapper.log('topic:' + topic);
136                     var message = key[i].message;
137                     if (message) {
138                         message = eval('`' + message + '`');
139                     } else {
140                         message = '';
141                     }
142                     consoleWrapper.log('message:' + message);
```

# Server

```
144      // if there is a response topic setup to receive a response
145      let listener;
146      let timer;
147      if (key[i].responseTopic) {
148          listener = function(topic, message) {
149              if (topic === key[i].responseTopic) {
150                  responseData.response.outputSpeech.text = message.toString();
151                  response.writeHead(200, {'Content-Type': 'application/json;charset=UTF-8'});
152                  response.end(JSON.stringify(responseData));
153                  mqttClientHandle.removeListener('message', listener);
154                  if (timer) {
155                      clearTimeout(timer);
156                  }
157              }
158          };
159
160          mqttClientHandle.on('message', listener);
161          mqttClientHandle.subscribe(key[i].responseTopic);
162      }
163
164      // send out the message
165      mqttClientHandle.publish(topic, message);
```

# Demo

- Show debug/console output

# What Works?

- Search
  - Play next episode
  - Recipes
- Hands free

# What Doesn't

- Repeated actions
  - Volume up/down
- Noisy environments
- When you demo to your friends/family 😊

# Summary and Questions

# Notices and disclaimers

- © 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.
- **U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**
- Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
- IBM products are manufactured from new parts or new and used parts.  
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”
- **Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**
- Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those
- customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
- References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
- Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
- It is the customer’s responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

# Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).
- 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.
- Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Linux is a registered trademark of Linus Torvalds in the United States
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- "TWITTER, TWEET, RETWEET and the Twitter logo are trademarks of Twitter, Inc. or its affiliates."
- Let's Encrypt is a trademark of the Internet Security Research Group. All rights reserved.
- Alexa, AWS, and Amazon are trademarks of Amazon. All rights reserved.





Discover. Collaborate. Deploy.

# Talk to your Code

Michael Dawson