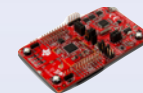


# A day in the life with speech recognition, machine learning, & IOT



#IBMBluemix



**David Boloker**

CTO

Emerging Technologies, IBM

[boloker@us.ibm.com](mailto:boloker@us.ibm.com)

**Mark VanderWiele**

Distinguished Engineer

Emerging Technologies, IBM

@MarkVanderWiele

**Ben Sechrist**

Emerging Technologies, IBM

@BenDSechrist

# The user experience is changing

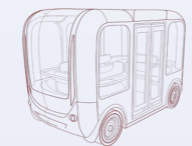
Search and Apps



Voice as the New Command Line



Ask and Receive



Analyze and Notify

ChatBOTS

Teach and Learn

Personal Assistants



## Mean time to know or act

# Looking forward

*We will no longer have to learn to use the machine,  
the machine will learn from listening to us.*

## **We will converse naturally within our own digital world to:**

- Ask questions
- Control devices
- Collaborate more naturally
- Purchase goods and services using “Conversational Commerce”
- Carry out our daily tasks - a personal assistant
- Learn, adapt, and extend out digital world



Re-imagining Enterprises - IT & LoB - where a majority of B2C interactions with new user experiences are initiated thru voice and text

- Conversational apps will eclipse the totality of application created to-date

# Speech Scenarios:

---

## Command and Control:

- “Make my drone fly” “turn on my oven”

## Question and Answer:

- “What can you do”

## Ask and Receive a Variable Response:

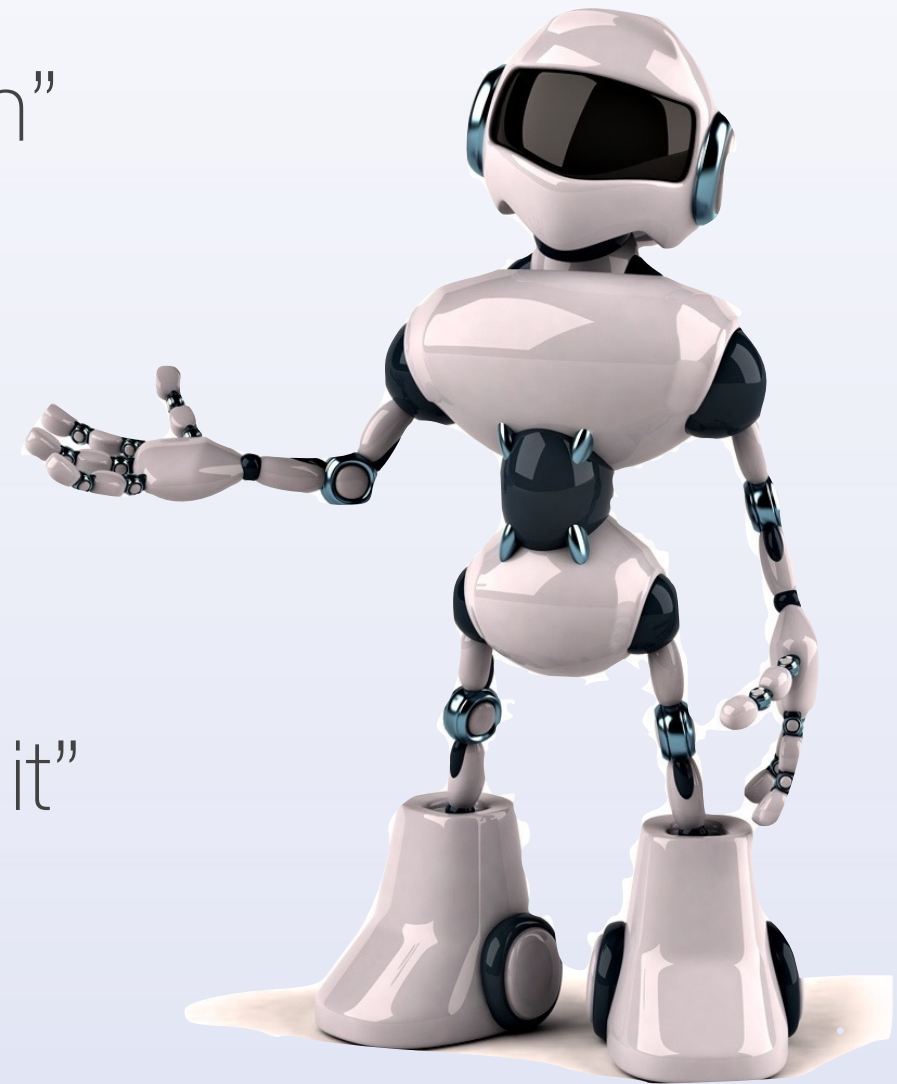
- “send” Response: “done” “sure” “got it”

## Conversations:

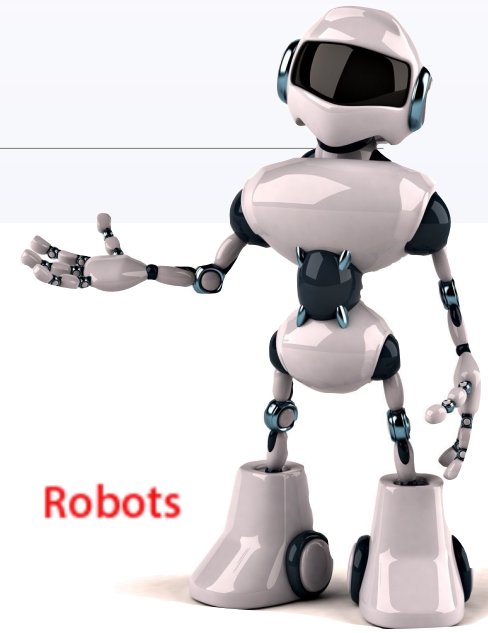
- “I would like to remodel my kitchen”

## Conversation with Reasoning:

- “Diagnose my problem”



# Conversation Use Cases & New Jobs



## Uses for the Conversation Service

### Customer Service



Add a chatbot to your website that automatically responds to your customers' most frequently asked questions

### Mobile Apps



Allow your customers to control your mobile app using natural language virtual agents

### Messaging Channels



Build Twitter, Slack, Facebook Messenger, and other messaging platform chatbots that interact instantly with channel users

### Internet-of-Things



Power connected devices to understand natural language and respond to your users' commands

### Robots

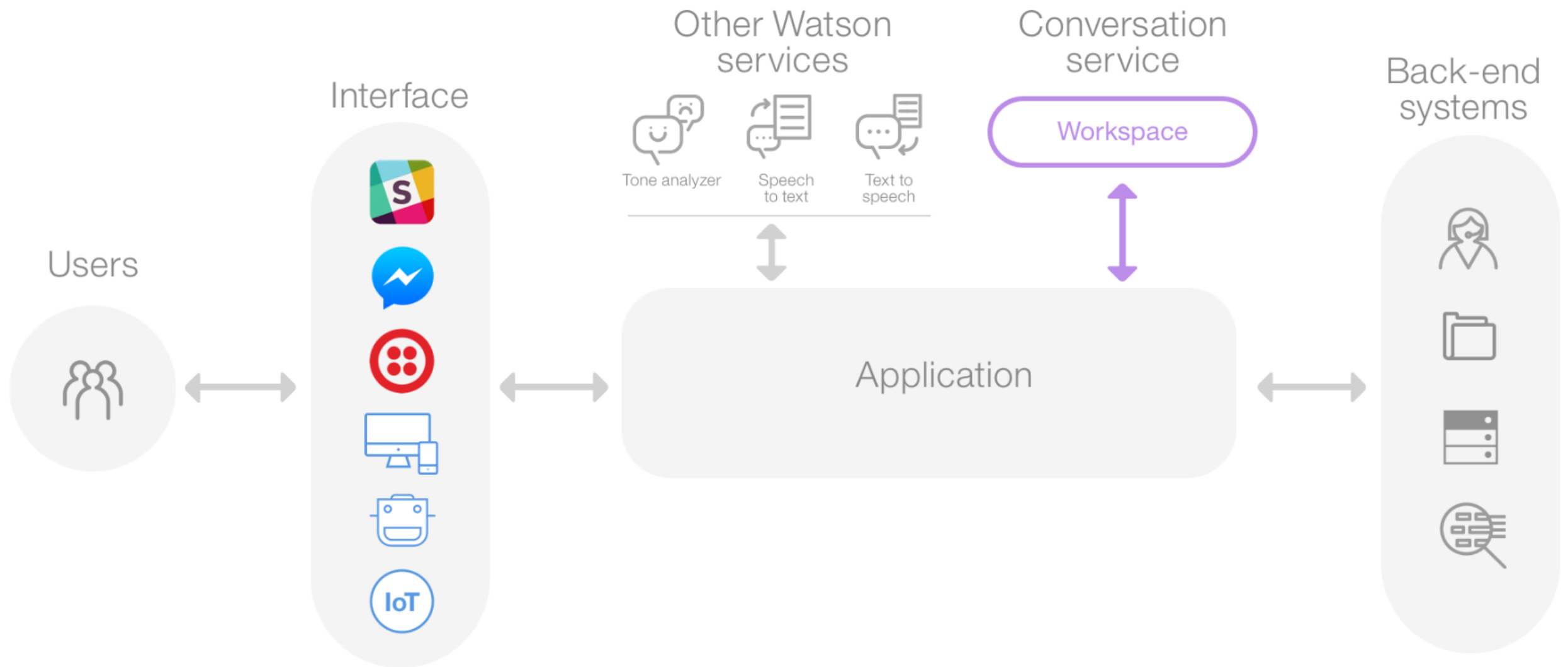
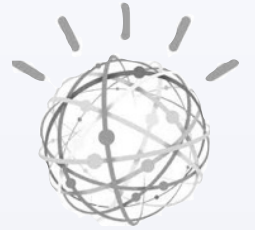


Power your robots with natural language understanding and conversational capabilities

New Job Roles - Interaction Designer, Conversation and Dialog, The Next Phase Of Designing Chatbot Personalities, Conversation Techniques For Designers...



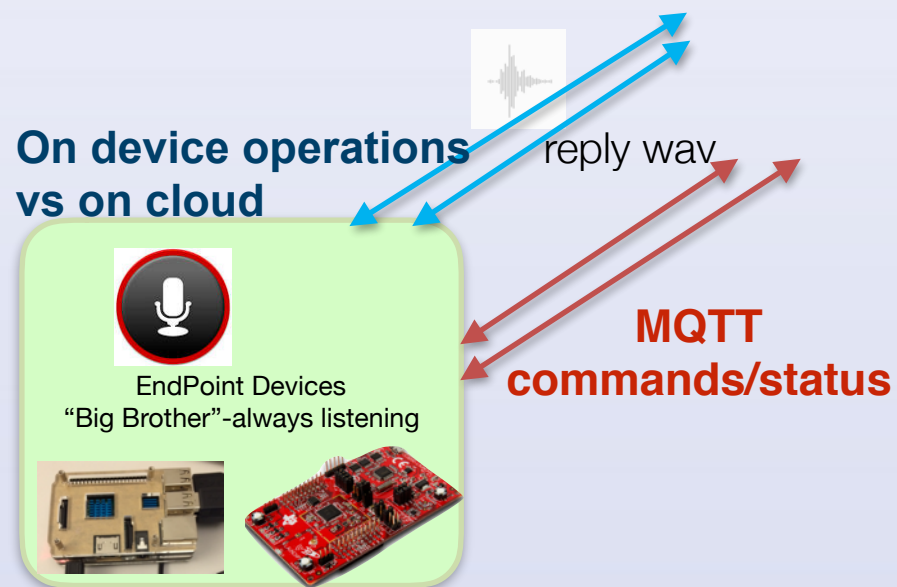
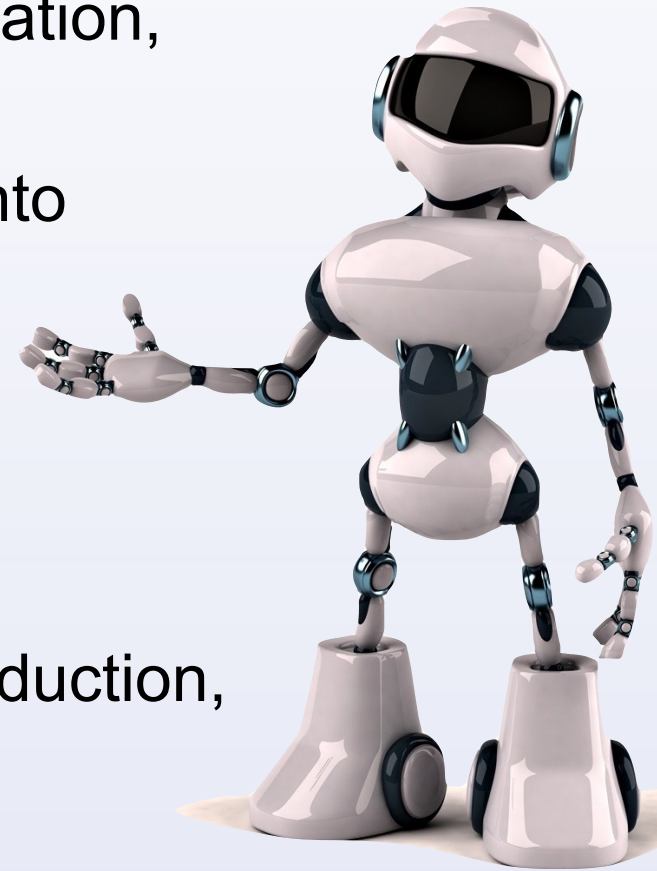
# Conversational Application - architectural flow



[https://www.ibm.com/watson/developercloud/doc/conversation/apps\\_overview.shtml](https://www.ibm.com/watson/developercloud/doc/conversation/apps_overview.shtml)

# Device Characteristics?

- Always on listening for key/wakeup words, immediate voice activation, and voice as a command line.
  - some push to talk, others wakeup and command, others go into conversational mode
- Respond in < 2 seconds
- Balance between on device operations vs cloud
  - based on device capabilities
- mic config varies - Single directional mic to mems array, noise reduction, beam forming, voice detection, biometrics...



*“The last “next” mile of device interface and analytics”*

# Aways listening (for wakeup words) low cost open device

<https://developer.ibm.com/recipes/tutorials/connect-a-simplelink-wi-fi-cc3200-launchpad-to-iotf->



<http://www.ti.com/ww/en/launchpad/launchpads-connected-cc3200-launchxl.html>

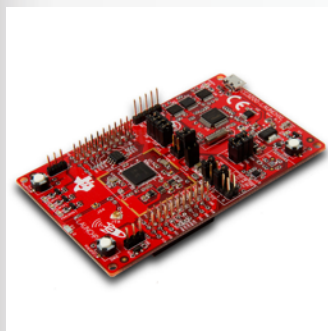
## cc3200 + audio booster pack ~ \$60

### Audio booster pack

- mic, line in, DSP, codec, \$30

### CC3200 launchpad

- Built in Wifi
- Simple Link Wifi Config via iPhone app
- on board Accelerometer
- 2 temp sensors (board+object)
- 2 buttons
- < \$30
- Arduino like dev environment & TI CodeComposer
  - <http://www.energia.nu>
  - 3.3V VCC, GPIOs, 1.5v analog read



@energiaproject  
#TILaunchPad  
#IBMBlueMix

Optional HW for Wakeup words and Voice Biometrics

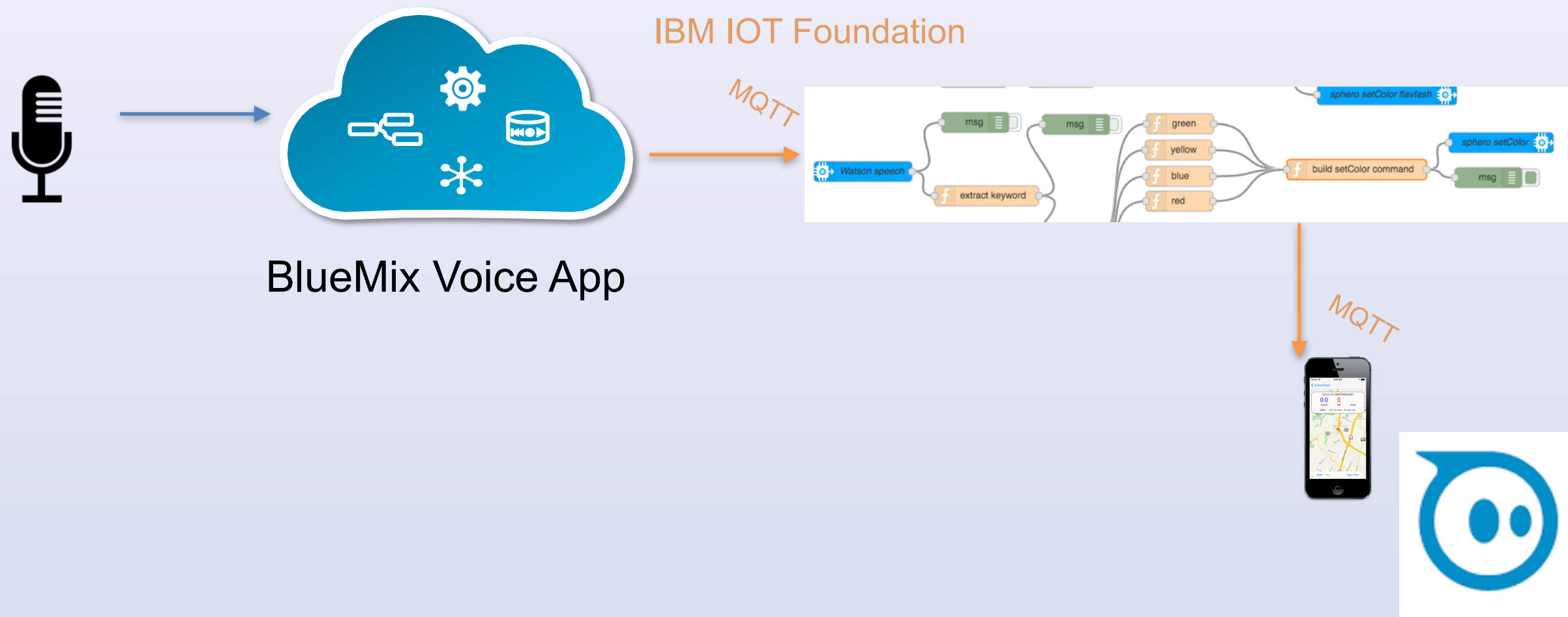
Additional work on far field beam forming & noise reduction



# Demo 1: Voice Command/Control

## Simple voice to text

- Match command to known list of actions (look for key words, nouns, verbs)
- Respond with text
- Generate response
- Generate MQTT command to control device



# Demonstration 2: A day in the Life with Speech, ML, and IOT

Using speech recognition and cloud technologies to:

Demo:

- **Monitor** and help plan my day's activities, including waking me up.
- **Summarize** my portfolio and News
  - Send to my phone or car.
- **Analyze** business data, IOT Data
- **Search and follow** hot topics
- **Plan and track** my projects, **shop**, **get help**
- **Display, Analyze & Visualize** my devices around the world
- **Securely control** devices around the world (with voice)
- **Create reusable conversations** from my interactions, **creating a "verbal mashup"**

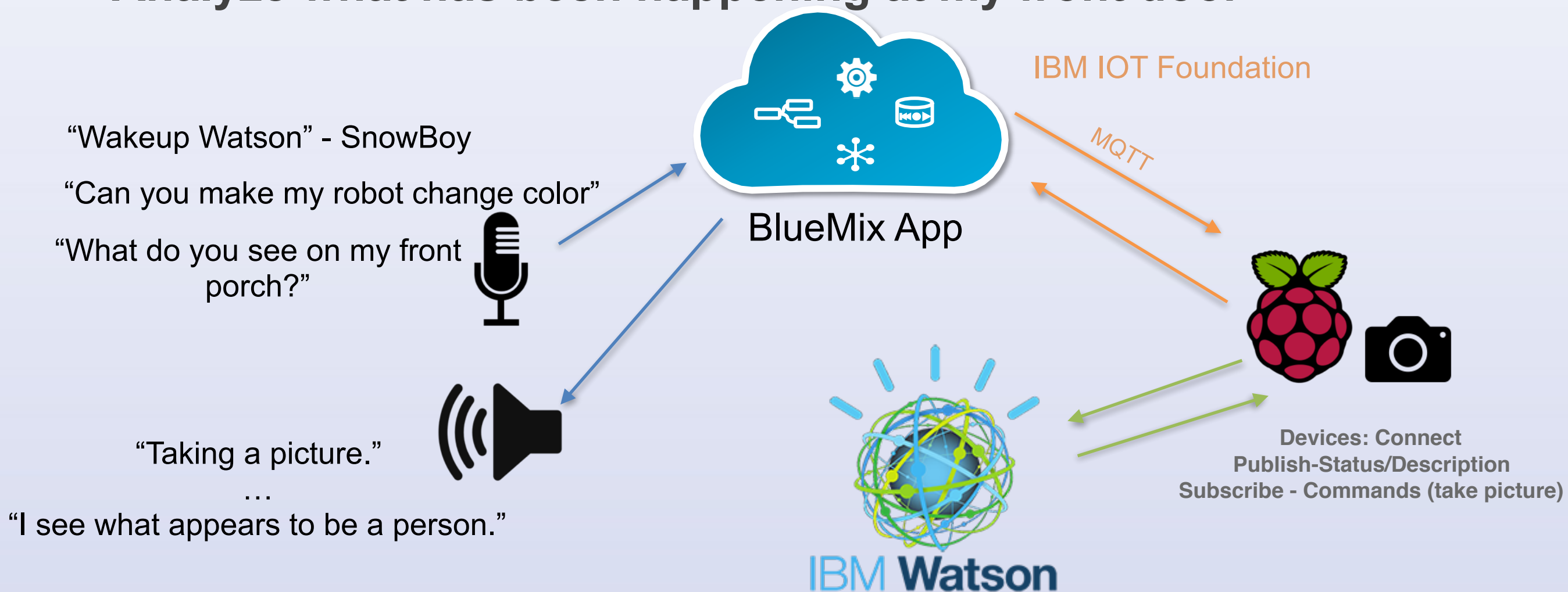


# Demo 3: Use Voice to command a device and analyze output

## You can do this at home!

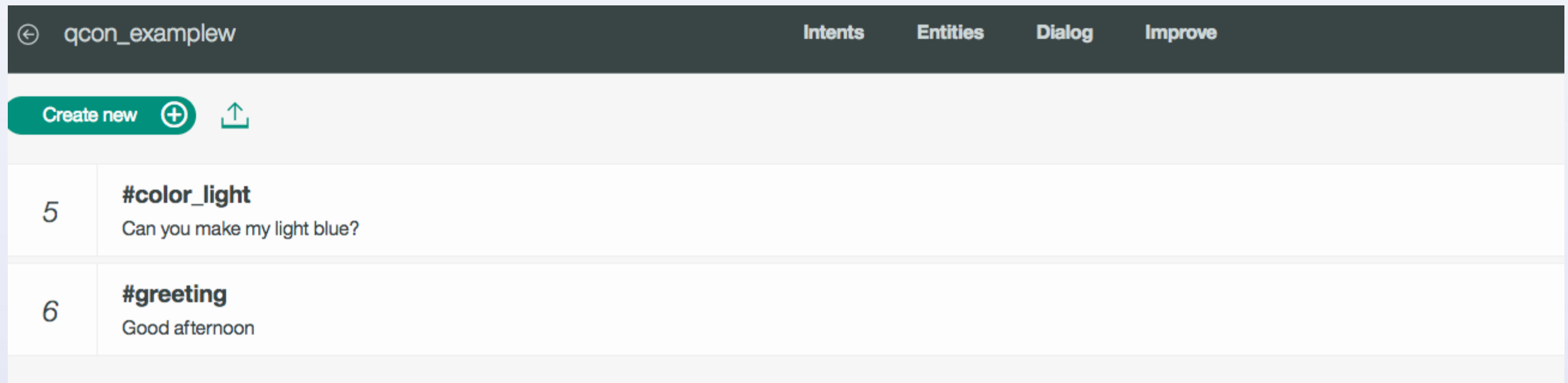
Live conversation with a device

- Control other devices in the room - BLE - ex BB8
- Take a picture
- Analyze the picture and tell me about it
- Analyze what has been happening at my front door



# Demo 4: Voice Command/Control

Build a conversation live



The screenshot shows the IBM Watson Assistant interface for a project named 'qcon\_examplew'. The top navigation bar includes 'Intents', 'Entities', 'Dialog', and 'Improve'. Below the navigation bar, there is a 'Create new' button with a plus icon and an upload icon. The main content area displays a list of intents:

5	<b>#color_light</b> Can you make my light blue?
6	<b>#greeting</b> Good afternoon

## Behind The Scenes

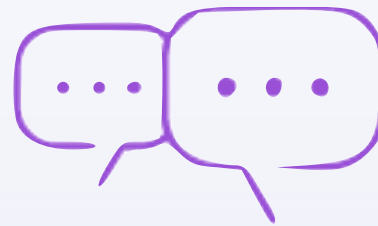
Using Customization based on conversations



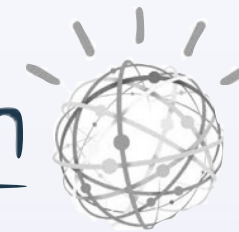
Speech to Text

Streamed voice wav

Text to Speech



### Conversation



text

JSON

text

Custom Node App -  
Deployed in Bluemix -  
Orchestrate between  
services  
"Hal9000"

### Tone Analyzer

Discover, understand, and revise the language tones in text.



### Visual Recognition



### AlchemyData News



### Language Translator



### Weather Company Data

Use the Weather Company Data for IBM Bluemix service to incorporate weat



Domain Services

Standard Services

Health

input voice wav



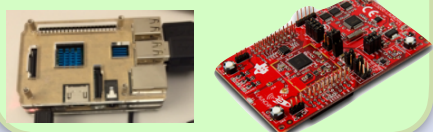
reply wav

MQTT  
commands/status

On device operations vs on cloud



EndPoint Devices  
"Big Brother"-always listening





# Watson service pipeline



What did you say?

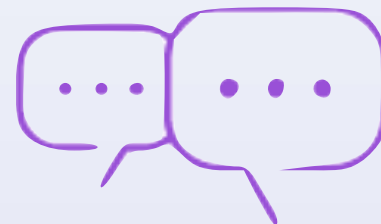


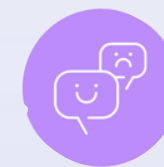
Speech to Text

What do you really want?  
What is your current **context**?  
What was your **Intent**?  
What **Object/Entity**?

Are you upset?

Using Customization  
based on  
conversations

 Conversation



Tone Analyzer

Discover, understand, and revise the language tones in text

- Use response (Q/A)
- Randomize response
- Personalize
- Use context
- Generate response

- Add empathy
- Escalate

# Understanding Speech to Text



Press to start speaking

## Bolt on simple speech front end - direct command/control

<http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/speech-to-text/#sampleApp>

```
speechToText.recognize({audio: audio, content_type: 'audio/l16; rate=44100'},  
function(err, transcript){  
  if (err) return res.status(500).json({error: err});  
  else return res.json(transcript);  
});
```

```
{ "results": [ { "alternatives": [  
  {  
    "confidence": 0.8691191673278809,  
    "transcript": "make my drone fly",  
    word_confidence    "make" 0.95    "my" 0.56    "drone" 0.86    "fly" 0.8  
  }  
], "final": true } ], "result_index": 0 }
```

Ouch, match text returned to cmd != strcmp

# Speech code - evolution to custom model

## 1. Speech to Text API

2. Match text to CMD  
Guess vs Final?  
Word confidence score  
Conversations & Context?  
Phonetic matching?  
Acronyms?  
Nouns n verbs?  
Utterances?

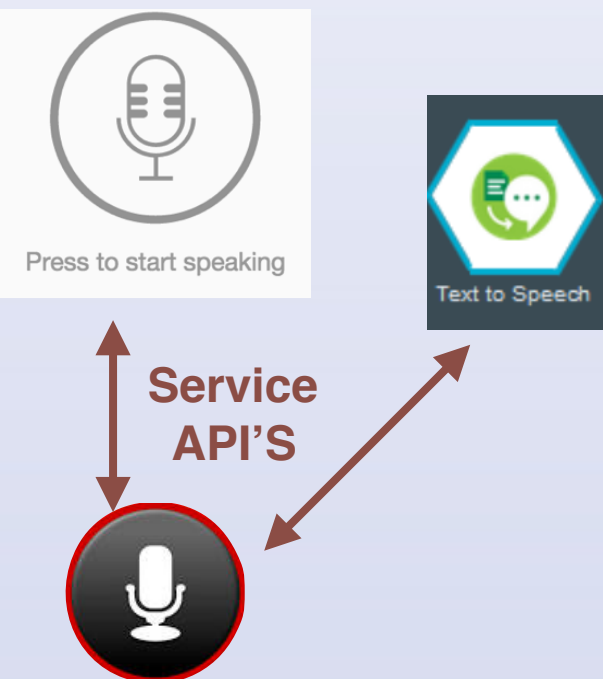
I said “my drone is Bryan's bebop”  
I got “my German is Brian's Bieber”

“Can you ask my drone to fly please”  
“Can you ask my drum the fly place”

## Retrain the speech to text service

To increase the probability within context of matching a command we use a custom speech model

You can use multiple custom models within a conversation



# Creating a Custom Speech Model

## 1. Create custom model

- POST /api/v1/customizations
  - {
    - “name”: “Custom Model”,
    - “base\_model\_name”: “en-US\_BroadbandModel”,
    - “description”: “Custom Model for QCON”
- Response 201
  - {
    - “customization\_id”: “abcdefghijklmnopqrstuvwxy”

## 2. Add corpus

- POST /api/v1/customizations/{id}/corpora/{name}
  - Corpus file
- Response 201

## 3. Wait for corpus to be analyzed

- GET /api/v1/customizations/{id}/corpora
- Response 200
  - {
    - “corpora”: [
      - ...
      - {
        - “name”: {name},
        - ...
        - “status”: “analyzed”
    - ...

## 4. Train model

- POST /api/v1/customizations/{id}/train
- Response 200

## 5. Wait for model to be trained

- GET /api/v1/customizations/{id}
- Response 200
  - {
    - ...
    - “status”: “ready”
    - ...

```
train.txt
1 Start my day
2 Yes
3 No
4 Why did you get me up so early?
5 Where should I be right now?
6 Check my portfolio
7 What went on since we last spoke
8 Positive
9 Ask news
10 We can discuss the summaries on the way to work
11 Tell Twitter
12 Ask business
13 Tell business
14 Get sales data
15 Summarize it
16 Show table
17 Show graph
18 Show bar chart
19 Exit
20 Ask drones
21 What drones are available?
22 Tell us about the future
23 Can you fly my drone?
24 Ask internet of things
25 Turn Dave's lamp on
26 Turn right
27 Take selfie
28 Take a picture
29 Land my drone
30 Can you locate my robot
31 Color red
32 Color blue
```

<http://www.ibm.com/watson/developercloud/speech-to-text/api/v1/>

# Example Code

## Create Custom Model

```
1 |const request = require('request');
2
3 |const WATSON_API = 'https://stream.watsonplatform.net/speech-to-text/api/v1/';
4
5 |request({
6 |  baseUrl: WATSON_API,
7 |  uri: '/customizations',
8 |  method: 'POST',
9 |  json: true,
10 |  auth: {
11 |    user: '',
12 |    pass: ''
13 |  }
14 |}, (err, res, body) => {
15 |  if (err || res.statusCode !== 201) {
16 |    // Handle error
17 |  } else {
18 |    // Grab customization id from json body
19 |  }
20 |});
```



## Add Corpus to Model

```
1  const request = require('request'),
2    fs = require('fs');
3
4  const WATSON_API = 'https://stream.watsonplatform.net/speech-to-text/api/v1/';
5  const customization_id = 'abcd1234';
6  const corpus_name = 'corpus1';
7
8  fs.createReadStream('corpus1.txt').pipe(request({
9    baseUrl: WATSON_API,
10   uri: '/customizations/' + customization_id + '/corpora/' + corpus_name,
11   method: 'POST',
12   json: true,
13   auth: {
14     user: '',
15     pass: ''
16   }
17 }, (err, res, body) => {
18   if (err || res.statusCode !== 201) {
19     // Handle error
20   }
21 }));
```

## Train Custom Model

```
1  const request = require('request');
2
3  const WATSON_API = 'https://stream.watsonplatform.net/speech-to-text/api/v1/';
4  const customization_id = 'abcd1234';
5
6  request({
7    baseUrl: WATSON_API,
8    uri: '/customizations/' + customization_id + '/train',
9    method: 'POST',
10   json: true,
11   auth: {
12     user: '',
13     pass: ''
14   }
15 }, (err, res, body) => {
16   if (err || res.statusCode !== 201) {
17     // Handle error
18   }
19 });
```

# Using Custom Model in Speech-to-Text

```

1  const request = require('request'),
2     fs = require('fs');
3
4  const WATSON_API = 'https://stream.watsonplatform.net/speech-to-text/api/v1/';
5  const customization_id = 'abcd1234';
6
7  fs.createReadStream('audio.wav').pipe(request({
8     baseUrl: WATSON_API,
9     uri: '/recognize',
10    method: 'POST',
11    headers: {
12      'Content-type': 'audio/wav'
13    },
14    json: true,
15    auth: {
16      user: '',
17      pass: ''
18    },
19    qs: {
20      customization_id: customization_id
21    }
22  }, (err, res, body) => {
23    if (err || res.statusCode !== 200) {
24      // Handle error
25    } else {
26      // Interpret transcript
27    }
28  }));

```

	Watson	Watson Custom Model
<b>Returned</b>	<b>why did you get me out so early</b>	<b>why did you get me up so early</b>
Confidence	80.30%	87.20%
Accuracy	61.29%	100.00%
Latency	3132 ms	3046 ms
<b>Returned</b>	<b>ask nerds</b>	<b>ask news</b>
Confidence	36.80%	66.50%
Accuracy	66.67%	100.00%
Latency	3069 ms	2044 ms
<b>Returned</b>	<b>asking there's</b>	<b>ask news</b>
Confidence	41.00%	70.40%
Accuracy	28.57%	100.00%
Latency	2417 ms	2408 ms
<b>Returned</b>	<b>other</b>	<b>positive</b>
Confidence	4.00%	98.10%
Accuracy	0.00%	100.00%
Latency	2919 ms	2064 ms
<b>Returned</b>	<b>ask news</b>	<b>ask news</b>
Confidence	36.50%	99.00%
Accuracy	100.00%	100.00%
Latency	2227 ms	1664 ms

# Building a conversation with Watson Conversations

mv-test Intents

Create new +

9	<b>#check-temperature</b> get my wine temperature
---	--

**#welcome**

+ Add a new user example...

- good afternoon
- good morning
- hello
- hello my friend
- hi

# Building a conversation with Watson Conversations

## Entities

← mv-test

Improve

My entities System entities

Create new +



1 entity

Sort by: Newest ▾

@Places



+ Add a new value

<input type="checkbox"/>	Home	<i>Air</i>	<i>house</i>	<i>inside</i>				(3 Synonyms)
<input type="checkbox"/>	Wine	<i>Cabernet</i>	<i>Cabernet Sauvig...</i>	<i>Chardonnay</i>	<i>Fermented Grapes</i>	<i>Grapes</i>		(10 Synonyms)



# Building a conversation with Watson Conversations

The image displays the IBM Watson Conversations interface. At the top left, a navigation bar shows a back arrow and the text "mv-test". A large speech bubble labeled "Dialog" points to a central dialog flow diagram. The diagram starts with a node labeled "#welcome" with a dropdown arrow and a plus icon. Below it is a "Watson Response" box containing a JSON snippet: 

```
"output": {  
  "text": {  
    "values": [  
      "hello",  
      "hi",  
      "..."  
    ]  
  }  
}
```

 Below the response box are two nodes: "#check-temperature" and "Anything else". To the right, a chat window titled "Try it out" with a "Clear" button shows a sequence of user inputs and bot responses. The inputs are "aloha", "hi", "howdy", "hi", and "aloha". Each input is followed by a dropdown menu showing "#welcome" as the selected response. At the bottom of the chat window is a text input field with the placeholder "Enter something to test your bot".

# Pick Your Device

## IBM IoT Foundation

# Recipes

[developer.ibm.com/recipes/](http://developer.ibm.com/recipes/)



ARM mbed



BeagleBone with SensorTag



SimpleLink™ Wi-Fi®  
CC3200 LaunchPad



Intel Galileo



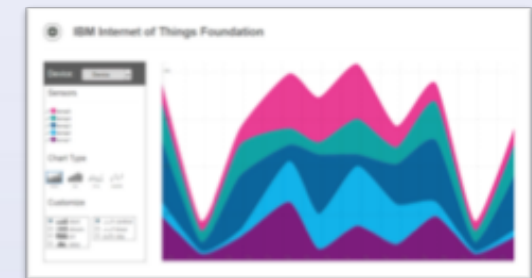
Raspberry Pi



Arduino Uno  
with Wi-Fi Shield



Device Simulator

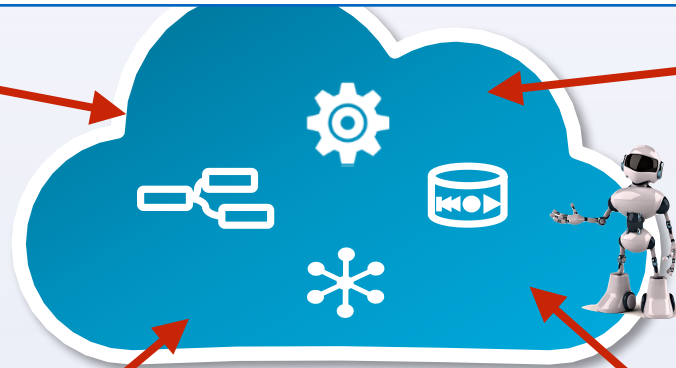
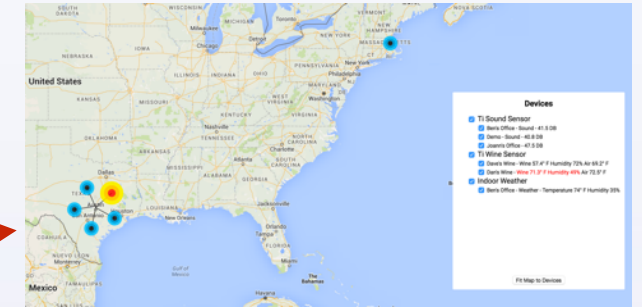


# Device Connect & Control - Journey/Experience



**Voice  
Activation/Control**

Cloud services: Analytics, Voice, Storage,  
Security, Message Storage/Delivery,  
Wiring/Logic, Views...



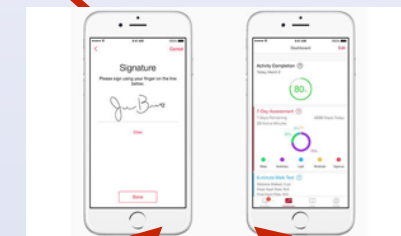
**MQTT**

**MQTT**

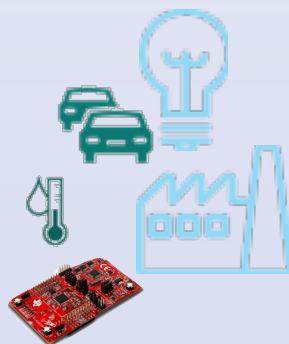
**JSON**

**Devices: Connect  
Publish-Status/Description  
Subscribe - Commands**

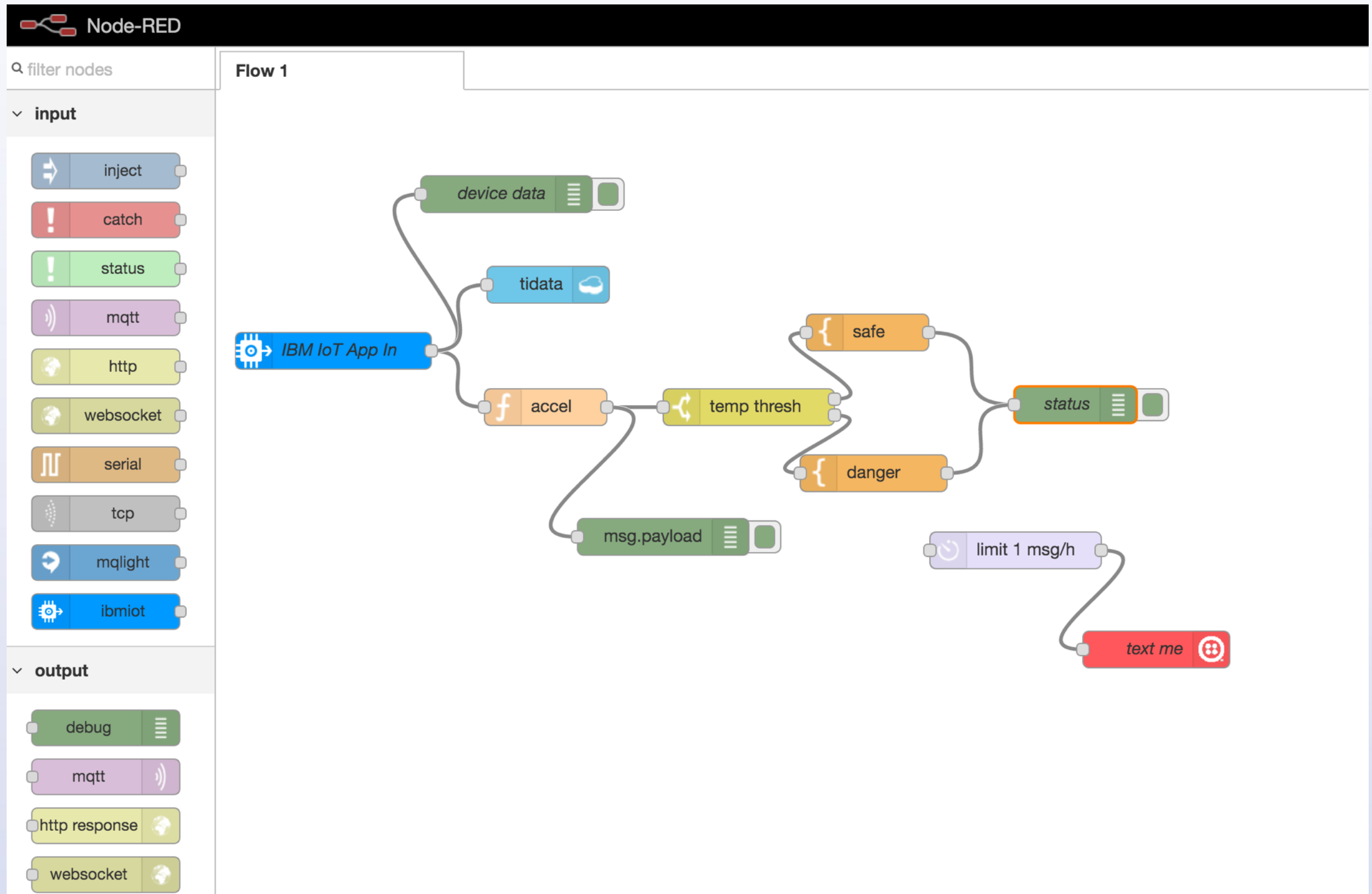
**Device Brokers**



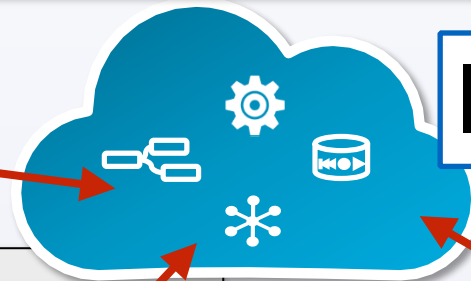
**HomeKit  
HealthKit**



# Wire new flows for your device



# Device control - from speech to command



IBM IOT Foundation

Subscribe  
cmd/set

MQTT

## Publish

```
{
  "d" : {
    "id" : "5B937D56-2E75-5293-BE2B-CB17C2EA539B",
    "name" : "David's Home : iDevicesNightLight",
    "data" : {
      "hue" : {
        "writable" : true,
        "step" : 1,
        "max" : 360,
        "value" : 220,
        "format" : "number",
        "min" : 0
      },
      "on" : {
        "value" : true,
        "writable" : true,
        "format" : "bool"
      }
    },
    "location" : {
      "lng" : -71.15152086101887,
      "lat" : 42.29974632421209
    },
    "iso" : "2016-04-11T10:54:29.317-0400"
  }
}
```

JSON  
Self  
Describe  
Writable  
Attributes

```
{
  "d" : {
    "id" : "5B937D56-2E75-5293-
    BE2B-CB17C2EA539B",
    "on" : true,
    "hue" : 160
  }
}
```





# MQTT

MQTT is **simple** to implement

**Connect**

**Subscribe**

**Publish**

**Unsubscribe**

**Disconnect**

```
client = new Messaging.Client(hostname, port, clientId)
client.onMessageArrived = messageArrived;
client.onConnectionLost = connectionLost;
client.connect({ onSuccess: connectionSuccess });

function connectionSuccess() {
    client.subscribe("planets/earth");
    var msg = new Messaging.Message("Hello world!");
    msg.destinationName = "planets/earth";
    client.publish(msg);
}

function messageArrived(msg) {
    console.log(msg.payloadString);
    client.unsubscribe("planets/earth");
    client.disconnect();
}
```

Eclipse Paho JavaScript MQTT client

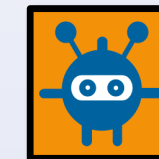
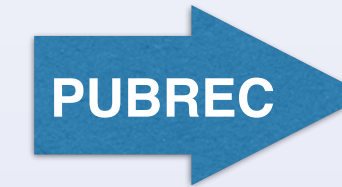
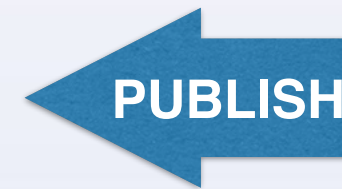
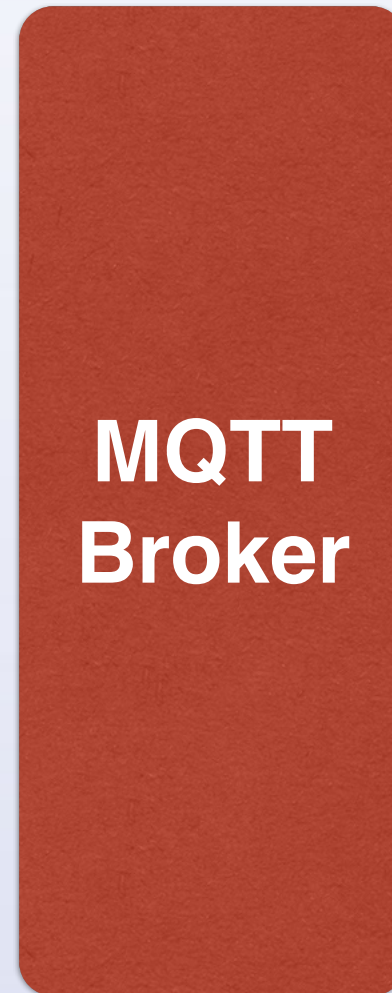
# MQTT

## Quality of Service for **reliable messaging**

Publish to topic `iot-2/evt/<event-type-id>/fmt/json`

Subscribe to topic `iot-2/cmd/<event_id>/fmt/json`

**QoS 0**  
at most once

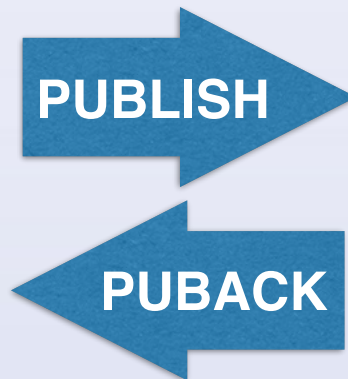


**QoS 2**  
exactly once

- doesn't survive failures
- never duplicated

- survives connection loss
- never duplicated

**QoS 1**  
at least once



- survives connection loss
- can be duplicated

---

# The Future



IBM Bluemix™  
[www.bluemix.net](http://www.bluemix.net)

---

# What if...

Personal assistants or other **devices could learn**

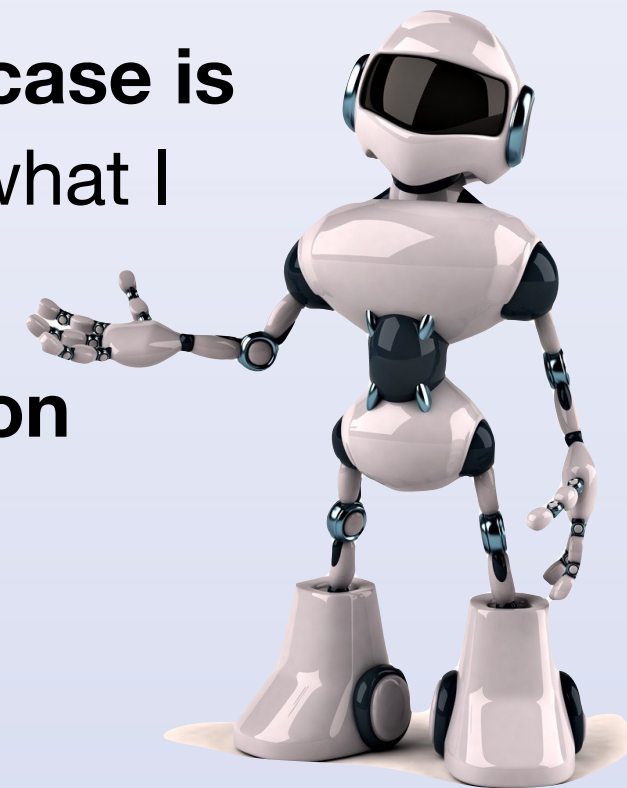
The **conversational style of interaction** with devices is more than just asking the device to perform a set of static tasks

We could **teach the devices new things through conversation**, combining tasks from an endless set of rich content components

We are **programming by example, the example in this case is in the form of conversation** - do what I say! AND learn what I do!

In many ways, the **zero UI of a conversational interaction pattern is much easier**

What better way to prescribe is there than to describe



# Why PaaS? What is Bluemix? - Sign up for a free trial



**(PaaS)** - for rapidly building, managing, and running cloud based applications and services of all types without worrying about the underlying infrastructure. Program in your choice of language.

**(IBM's Bluemix)** - Built on open-standards and open source technologies: Cloud Foundry, OpenStack, MQTT, docker,...

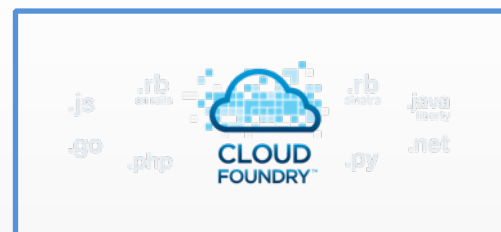
State of the Art User Interface

**Services Catalog** containing Services/APIs for Mobile, Data, Enterprise data connectors, Cognitive, Analytics, Social and any callable Rest based service

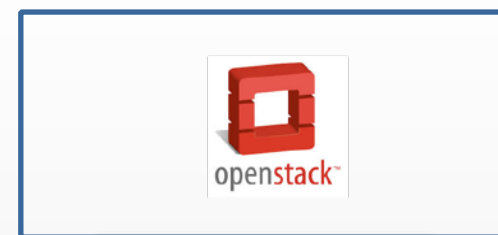
Multiple flavors - **public, dedicated, on-premise, hybrid**



## Instant Runtimes

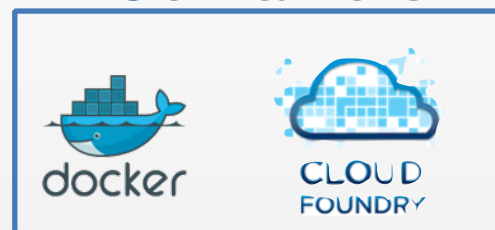


## Virtual Machines



IaaS

## Containers



## Build Apps Using Services



## NEW: Serverless





Thank you for your time



[www.bluemix.net](http://www.bluemix.net)

Tweet #IBMBluemix @qconsf @MarkVanderwiele



# for more info see the following blogs



Jon Kaufman: [jkaufman.io](http://jkaufman.io)

<https://github.com/watson-developer-cloud/company-insights>

Steve Atkiri: [stevenatkin.com](http://stevenatkin.com)

Niklas Heidlof <http://heidloff.net>

James Thomas <http://jamesthom.as/blog/categories/bluemix/>

#IBMBlueMix

