

# Workshop: Connecting IoT devices to the cloud with IBM and mbed Connector

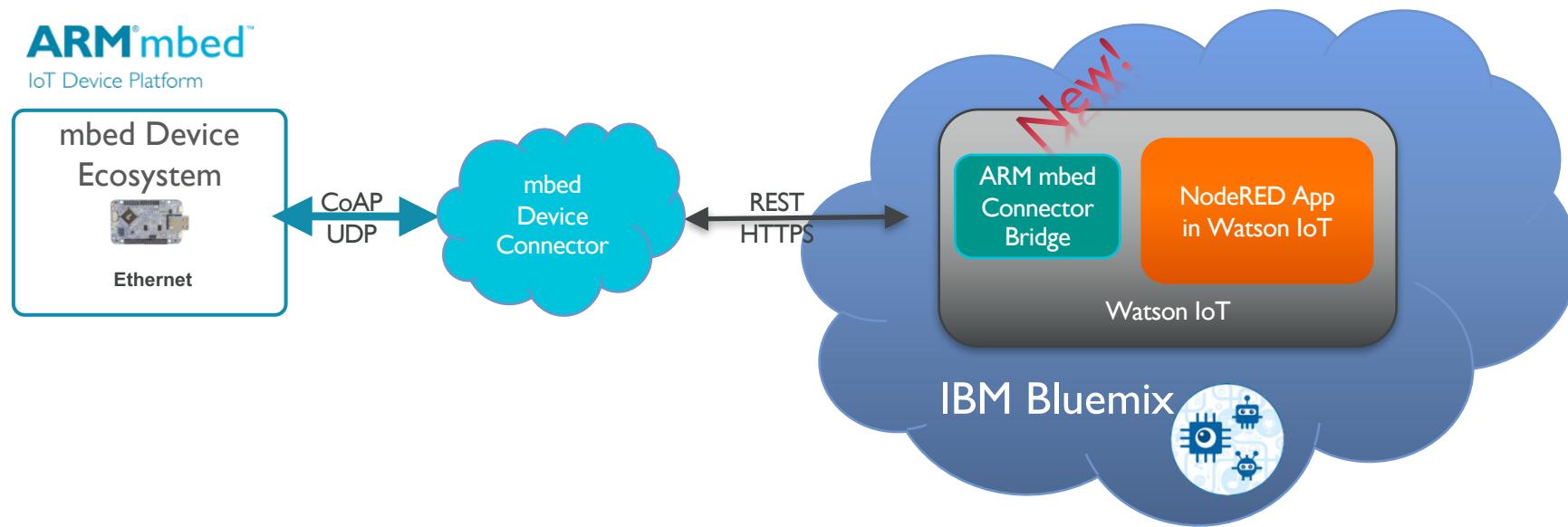
**ARM**

Doug Anson  
Solutions Architect / IoT BU / ARM

Pratul Sharma  
Product Marketing Manager - Ecosystem / IoT BU / ARM

Santa Clara, CA / mbed Connect USA  
October / 24 / 2016

# What will we build in this Workshop?



- Connect your mbed device into Watson IoT through mbed Connector and Watson's Connector Bridge
  - Create a simple mbed device and connect it to mbed Device Connector
  - Watson IoT now has a fully integrated mbed Device Connector Bridge that links mbed Device Connector to Watson!!
  - Exploration of the device data telemetry using NodeRED flows within Watson IoT

# Workshop: Let's get started!

- Create and setup your Bluemix account (should be completed prior to workshop)
- Install the necessary tools/drivers into your Windows/Mac/Linux PC (should be completed prior to workshop)
- Create mbed developer and Connector accounts (should be completed prior to workshop)
- Retrieve a set of provisioning credentials (security.h)
- Import our mbed sample project into the online IDE
- Update the sample project with the provisioning credentials (security.h)
- Compile, Install, Download, and Copy into the mbed device
- Connect a Serial Terminal (115200,8N1, proper mbed COM port chosen for Windows users...)
- Send the “Break” command to reset the mbed device
- See the device output on our Serial Terminal (PTSOI method...)

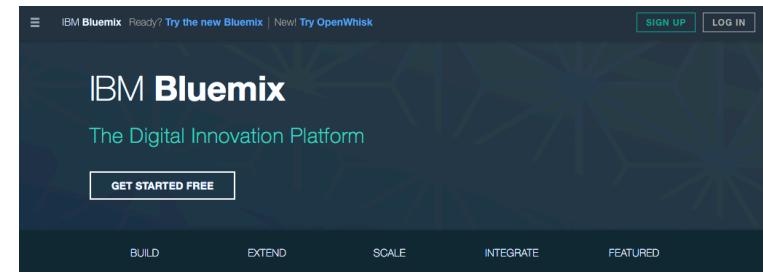
NOTE: During the workshop, be sure to double-check your copy/paste operations...

Quick Links: Visit [https://github.com/ARMmbed/anson\\_workshop\\_mbed\\_connect\\_2016](https://github.com/ARMmbed/anson_workshop_mbed_connect_2016)

- README.md has most of the links in the workshop...

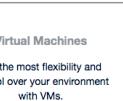
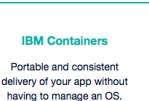
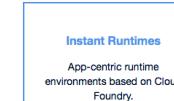
# Create our Bluemix Account

- Navigate to your Bluemix Dashboard:  
<https://console.ng.bluemix.net/>
  - Press “Sign Up”
  - Complete the sign-up process
  - A confirmation email will be sent that must be acknowledged
  - Log into your Bluemix account and establish your default organization and space
- Prior to  
Workshop



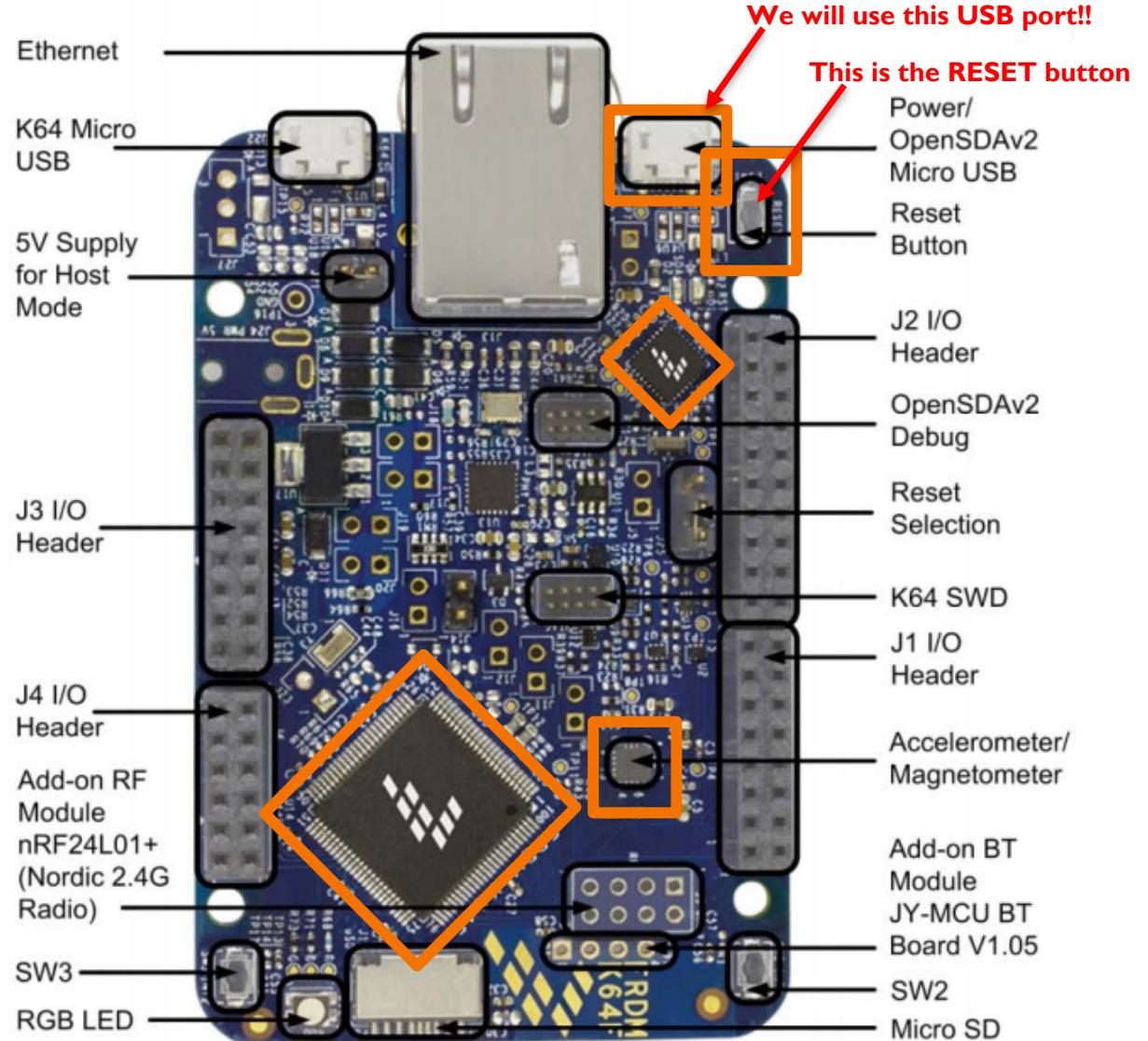
Build your apps, your way.

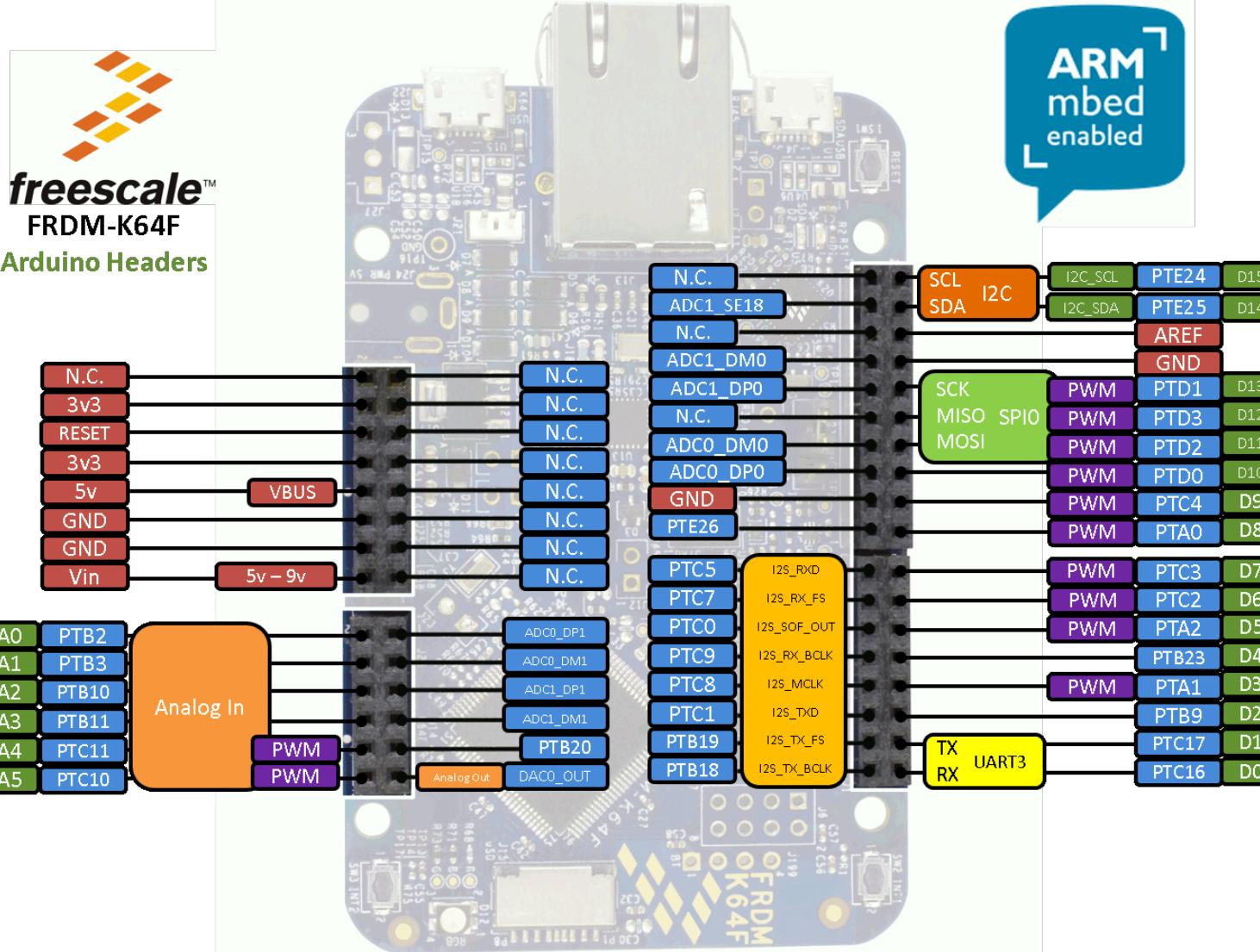
Use a combination of the most prominent open-source compute technologies to power your apps. Then, let Bluemix handle the rest.

The screenshot shows the 'Sign up for IBM Bluemix' form. It includes fields for 'First Name\*', 'Email Address\*', 'Last Name\*', 'Password\*', 'Re-enter Password\*', 'Company', 'Security Question\*', 'Select your country or region' (with 'UNITED KINGDOM' selected), and 'Security Answer'. There are also checkboxes for 'Keep me informed of products, services, and offerings from IBM companies worldwide: By email' and 'By telephone'. A 'CREATE ACCOUNT' button is at the bottom right.

# NXP- FRDM-K64F Overview

- **Freedom Development Platform**
  - Quick, simple development experience with rich features
    - Cortex-M4, 120MHz, 1MB Flash, 256KB SRAM
    - Easy access to MCU I/O
    - 3-axis **accelerometer**/3-axis **magnetometer**
    - RGB LED
    - Add-on **Bluetooth** Module
    - Built-in Ethernet/Add-on **Wireless** Module
    - Micro SD
- **Arduino shield compatible**
- Flash programming functionality
- Enabled by OpenSDA debug interface





# Install the Necessary Tools

Prior to  
Workshop

- **Windows**

- **IMPORTANT:** Install the mbed USB Serial driver -  
[https://developer.mbed.org/media/downloads/drivers/mbedWinSerial\\_16466.exe](https://developer.mbed.org/media/downloads/drivers/mbedWinSerial_16466.exe)
  - Insert the USB cable and mbed device BEFORE running the Serial Driver installation...
  - the installer MUST see the device first
- Serial Terminal: Putty - <http://www.putty.org/>
- Option: Install CoolTerm - [http://freeware.the-meiers.org/CoolTerm\\_Win.zip](http://freeware.the-meiers.org/CoolTerm_Win.zip)
- Chrome and Firefox Browsers installed

- **Mac**

- Serial Terminal: CoolTerm - [http://freeware.the-meiers.org/CoolTerm\\_Mac.zip](http://freeware.the-meiers.org/CoolTerm_Mac.zip)
- Chrome and Firefox Browsers installed

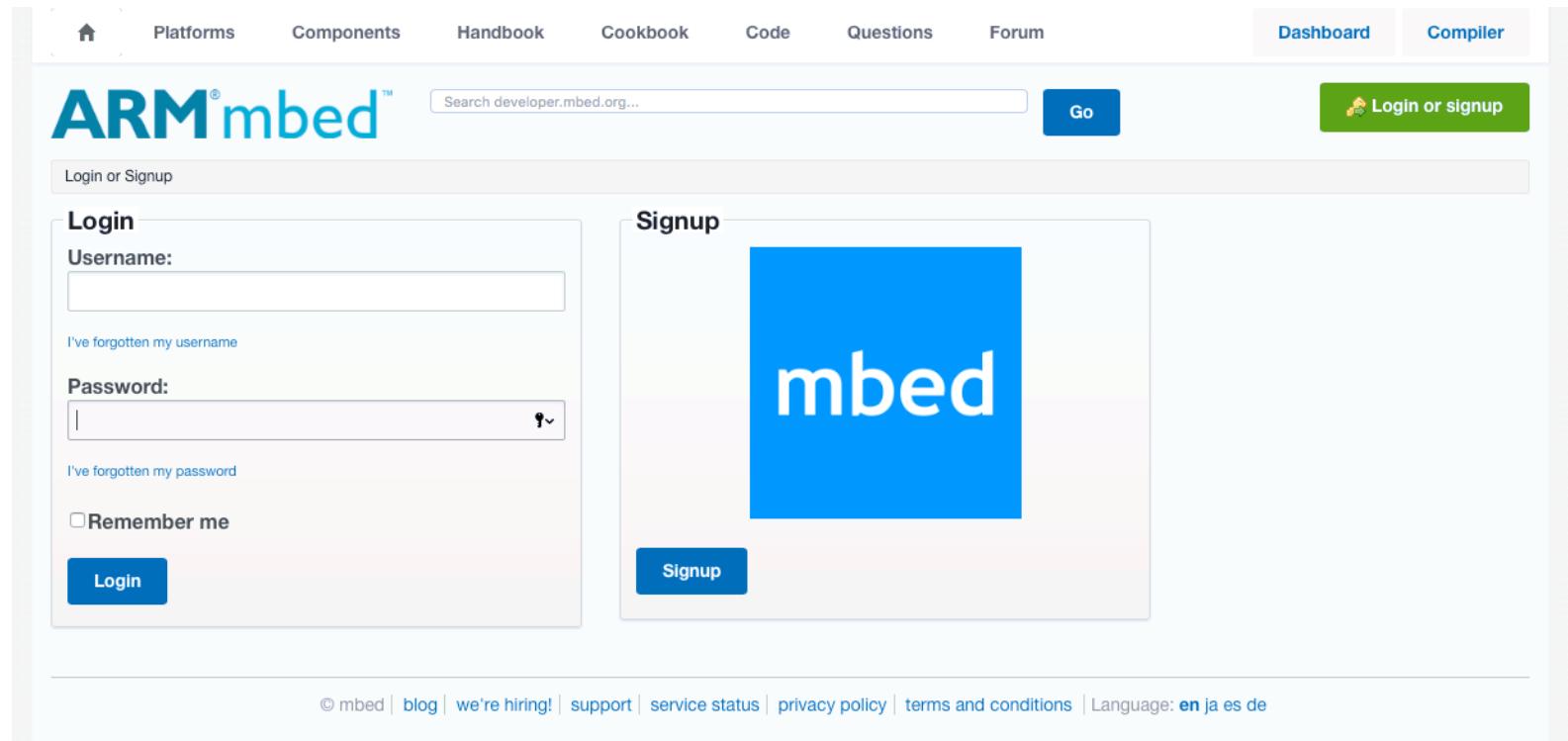
- **Linux**

- Serial Terminal: CoolTerm - [http://freeware.the-meiers.org/CoolTerm\\_Linux.zip](http://freeware.the-meiers.org/CoolTerm_Linux.zip)
- Chrome and Firefox Browsers installed

Connect both your USB cable and Ethernet cable to the K64F and leave it there for now

# Create Your mbed Account

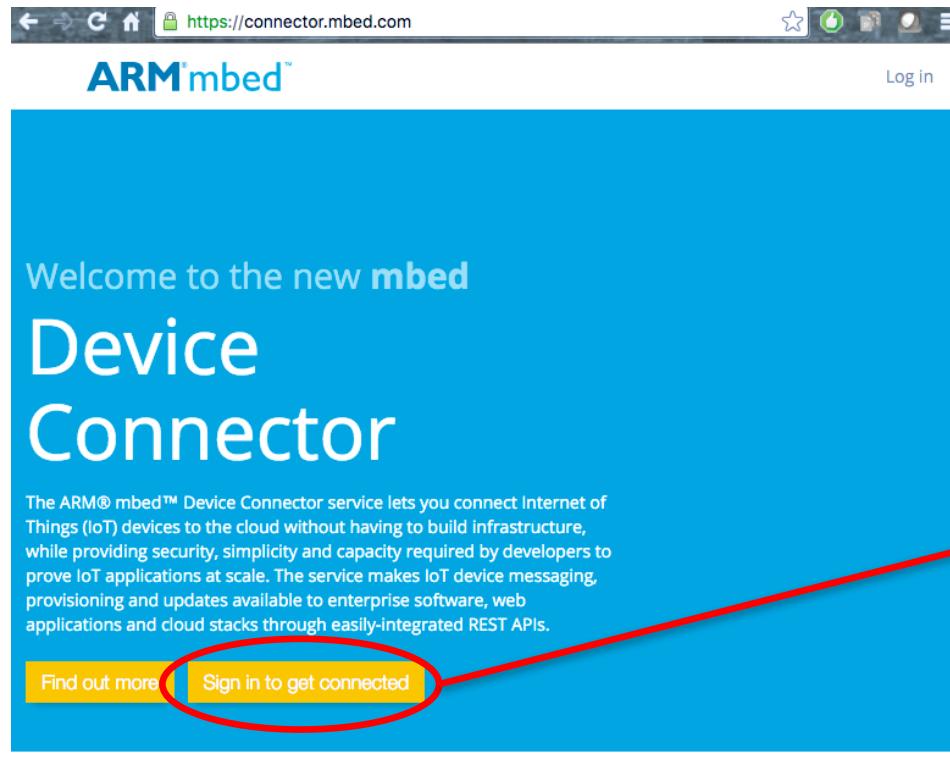
- Navigate to: <https://developer.mbed.org>
- Create an Account



# Create Your mbed Connector Account...

Prior to  
Workshop

- Navigate to the mbed Connector Dashboard: <https://connector.mbed.com>
- Confirm that you can log into your mbed device Connector dashboard



The screenshot shows the mbed Device Connector Dashboard. At the top, it says "mbed Device Connector (Beta)" and "Dashboard". The dashboard is divided into several sections: "My environment" (Dashboard), "My devices" (Connected devices: 0 of 100, Security credentials), "Device Connector" (API Console: 7 of 10000 per hour transactions), and "My applications" (Access keys: 5 of 2). There are also links to "Learn how to develop my device application", "Access REST API documentation", and "Learn how to develop my web application".

# Log into the Online IDE - Import our K64F Project

- Go to <https://developer.mbed.org>

- Select the “Compiler” page

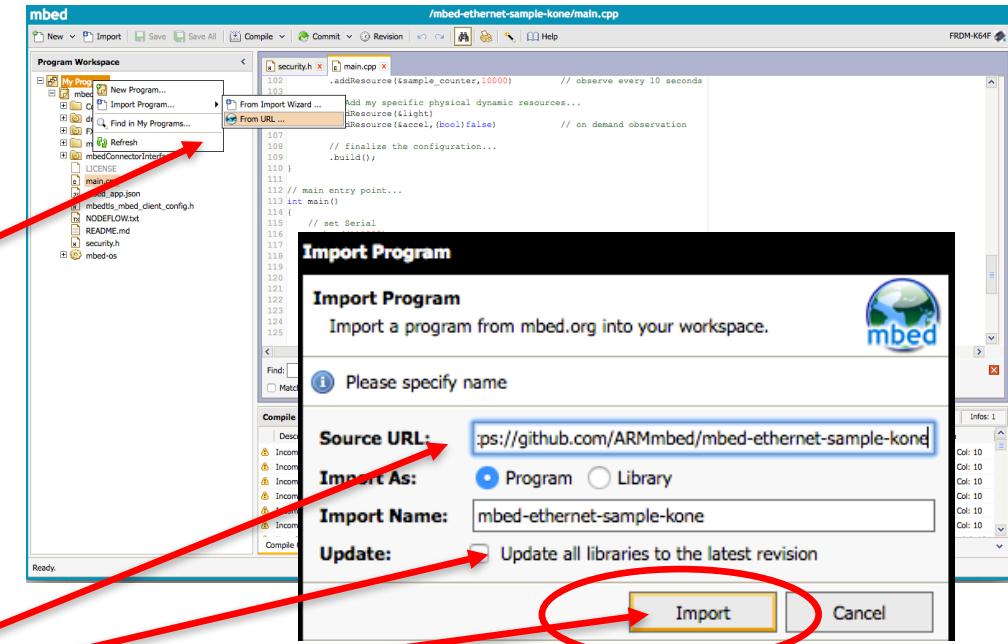
- Right-click on “My Programs” → “Import Program”

- Select “from URL...”

- Enter this URL (Leave the “Update all libraries...” **unchecked**)

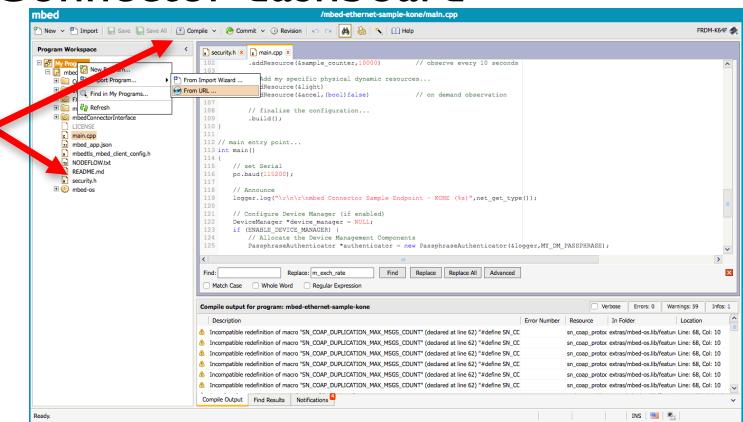
<https://github.com/ARMmbed/mbed-ethernet-sample-techcon2016/>

- Press “Import”



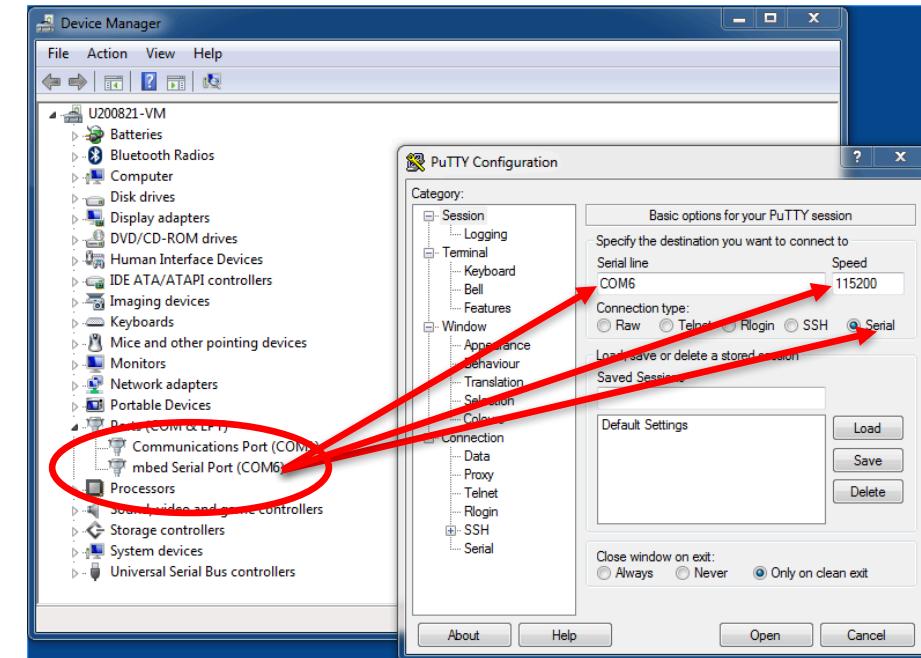
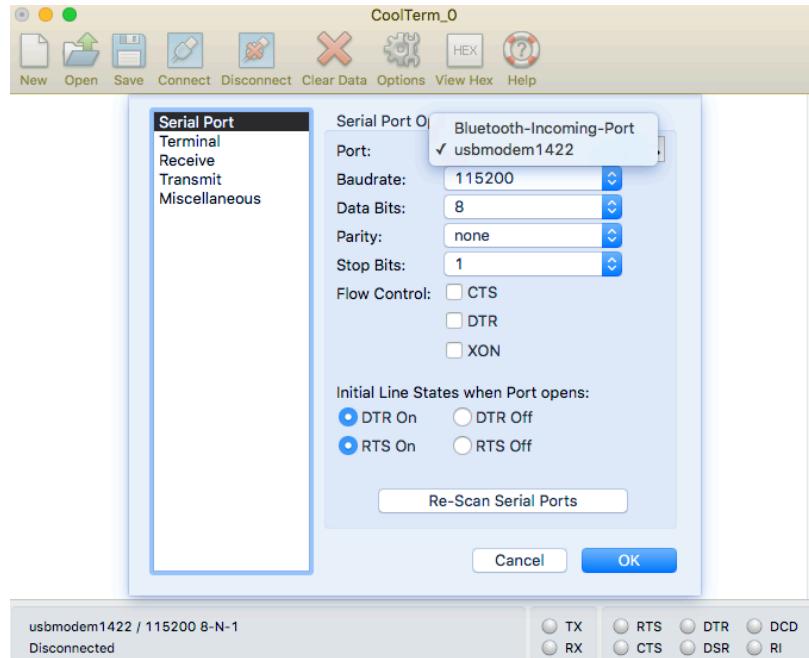
# Set Provisioning Credentials in Your Endpoint Code

- Go back to the mbed Device Connector dashboard: <https://connector.mbed.com>
  - In the left sidebar, select “Security Credentials” → “Get My Security Credentials”
  - Copy all contents of security.h (**Record the MBED\_ENDPOINT\_NAME & MBED\_DOMAIN values... used later!**)
- Now, go back to the Compiler page of your online IDE
- Replace security.h with the the new security.h that you copied from your Connector dashboard
- Save
  - Glance at main.cpp... a clean and simple mbed endpoint example
  - Exposes two CoAP resources: accelerometer and LED
- Select the project name and press the “Compile” button
- The endpoint code should compile up successfully
- The online IDE will deposit a “bin” file into your downloads directory
- Drag-n-Drop this bin file to your “MBED” flash drive (may also be called “DAPLINK”)
- K64F green LED will flicker for a bit, then stop, and dismount/remount...



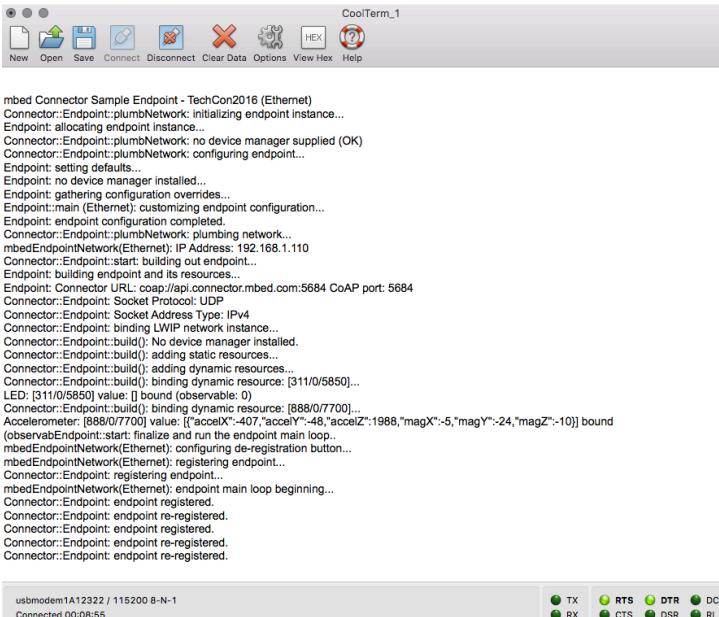
# Running your Endpoint Code

- Bring up your Serial Terminal
  - CoolTerm for Windows, Mac, Linux: Select: “Options→”Re-scan serial ports”. Select the mbed one found.
    - For MAC, you may need to “authorize” the CoolTerm Application under your “System Preferences”-> “Security & Privacy”... you may have to allow apps to run from “any developer”, then authorize the launch of CoolTerm.
  - PuTTY for Windows : You must determine what COM port your mbed device is. Look in the Windows Device Manager FYI
- For the endpoint serial configuration, set the baud rate to 115200 baud, defaults for everything else: (8, N, one)
  - PuTTY for Windows: Ensure that you have “Serial” radio button selected...



# Running your Endpoint Code...

- Connect your Serial Terminal to the K64F:
  - CoolTerm for Windows, Mac, Linux: Simply press the “Connect” button on the top part of the CoolTerm GUI
  - PuTTY for Windows: Press the “Open” button
- Send a “Break” command from the serial terminal (or press the RESET button on the K64F)
  - CoolTerm for Windows, Mac, Linux: “Connection” → “Send Break”
  - PuTTY for Windows: Right-click on top of Window, Select “Special Command” → “Break”
- Look at the output – you need to confirm that you see “endpoint registered” in the output:



CoolTerm\_1

mbed Connector Sample Endpoint - TechCon2016 (Ethernet)

Connector::Endpoint::plumbNetwork: initializing endpoint instance...

Endpoint: allocating endpoint instance...

Connector::Endpoint::plumbNetwork: no device manager supplied (OK)

Connector::Endpoint::plumbNetwork: configuring endpoint...

Endpoint: setting defaults...

Endpoint: no device manager installed...

Endpoint: gathering configuration services...

Endpoint::main (Ethernet): customizing endpoint configuration...

Endpoint: endpoint configuration completed.

Connector::Endpoint::plumbNetwork: plumbing network...

mbedEndpointNetwork(Ethernet): IP Address: 192.168.1.110

Connector::Endpoint::start: building out endpoint...

Endpoint: building endpoint and its resources...

Endpoint: Connector URL: coap://api.connector.mbed.com:5684 CoAP port: 5684

Connector::Endpoint::Socket Protocol: UDP

Connector::Endpoint::Socket Address Type: IPv4

Connector::Endpoint::binding LWIP network instance...

Connector::Endpoint::build(): No device manager installed.

Connector::Endpoint::build(): adding static resources...

Connector::Endpoint::build(): adding dynamic resources...

Connector::Endpoint::build(): binding dynamic resource: [311/0/5850]...

LED: [311/0/5850] value: [] bound (observable: 0)

Connector::Endpoint::build(): binding dynamic resource: [888/0/7700]...

Accelerometer: [888/0/7700] value: [{"accelX": -407, "accelY": -48, "accelZ": 1988, "magX": -5, "magY": -24, "magZ": -10}] bound (observeEndpoint::start: finalize and run the endpoint main loop...

mbedEndpointNetwork(Ethernet): configuring de-registration button...

mbedEndpointNetwork(Ethernet): registering endpoint...

Connector::Endpoint::re-register...

mbedEndpointNetwork(Ethernet): endpoint main loop beginning...

Connector::Endpoint::endpoints registered.

Connector::Endpoint::endpoints registered.

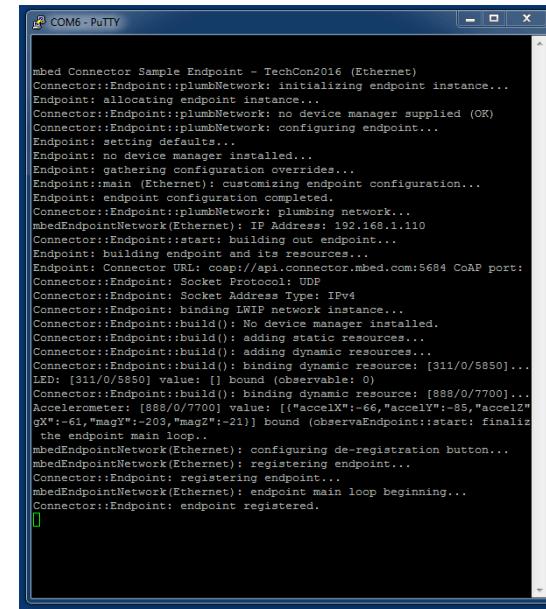
Connector::Endpoint::endpoints registered.

Connector::Endpoint::endpoints registered.

Connector::Endpoint::endpoints registered.

usbmodem1A12322 / 115200 8-N-1  
Connected 00:08:55

TX RTS DTR DCD  
RX CTS DSR RI



COM6 - PUTTY

mbed Connector Sample Endpoint - TechCon2016 (Ethernet)

Connector::Endpoint::plumbNetwork: initializing endpoint instance...

Endpoint: allocating endpoint instance...

Connector::Endpoint::plumbNetwork: no device manager supplied (OK)

Connector::Endpoint::plumbNetwork: configuring endpoint...

Endpoint: setting defaults...

Endpoint: no device manager installed...

Endpoint: gathering configuration overrides...

Endpoint::main (Ethernet): customizing endpoint configuration...

Endpoint: endpoint configuration completed.

Connector::Endpoint::plumbNetwork: plumbing network...

mbedEndpointNetwork(Ethernet): IP Address: 192.168.1.110

Connector::Endpoint::start: building out endpoint...

Endpoint: building endpoint and its resources...

Endpoint: Connector URL: coap://api.connector.mbed.com:5684 CoAP port: 5684

Connector::Endpoint::Socket Protocol: UDP

Connector::Endpoint::Socket Address Type: IPv4

Connector::Endpoint::binding LWIP network instance...

Connector::Endpoint::build(): No device manager installed.

Connector::Endpoint::build(): adding static resources...

Connector::Endpoint::build(): adding dynamic resources...

Connector::Endpoint::build(): binding dynamic resource: [311/0/5850]...

LED: [311/0/5850] value: [] bound (observable: 0)

Connector::Endpoint::build(): binding dynamic resource: [888/0/7700]...

Accelerometer: [888/0/7700] value: [{"accelX": -61, "accelY": -203, "accelZ": 1988, "magX": -5, "magY": -24, "magZ": -10}] bound (observeEndpoint::start: finalize and run the endpoint main loop...

mbedEndpointNetwork(Ethernet): configuring de-registration button...

mbedEndpointNetwork(Ethernet): registering endpoint...

Connector::Endpoint::registering endpoint...

mbedEndpointNetwork(Ethernet): endpoint main loop beginning...

Connector::Endpoint::endpoints registered.

# Status Check

## So far we've completed the following

- Setup our accounts and PC with appropriate tools (prior to workshop)...
- Retrieved a set of provisioning credentials (security.h)
- Imported our mbed sample project into the online IDE
- Updated the sample project with the provisioning credentials (security.h)
- Compiled, Installed, Downloaded, and Copied into the mbed device
- Connected a Serial Terminal (115200,8NI, proper mbed COM port chosen for Windows users...)
- Sent the “Break” command to reset the mbed device
- Saw the device output on our Serial Terminal (PTSOOI method...)

Next, we will import and configure the Watson IoT mbed Connector Bridge...

# Watson IoT mbed Connector Bridge Configuration

# What we will do next...

- Create our own Watson IoT Instance within our Bluemix account
- Create our Watson IoT Application Credentials (used in the NodeRED application...)
- Create our sample Watson IoT NodeRED Application
- Bind the Watson NodeRED Application to our Watson IoT instance
- Create a mbed Connector API Token
- Acquire our mbed Connector MBED\_DOMAIN value
- Configure the Watson IoT ARM mbed Connector Bridge

# Create our Watson IoT Instance

- Go to the Bluemix dashboard:  
<https://console.ng.bluemix.net>
- Select “Internet of Things”
- Select “Get Started”
- Configure your Watson IoT Instance
  - Leave “unbound”
  - Select the “Free” plan  
(scroll down...)
- Select “Create”

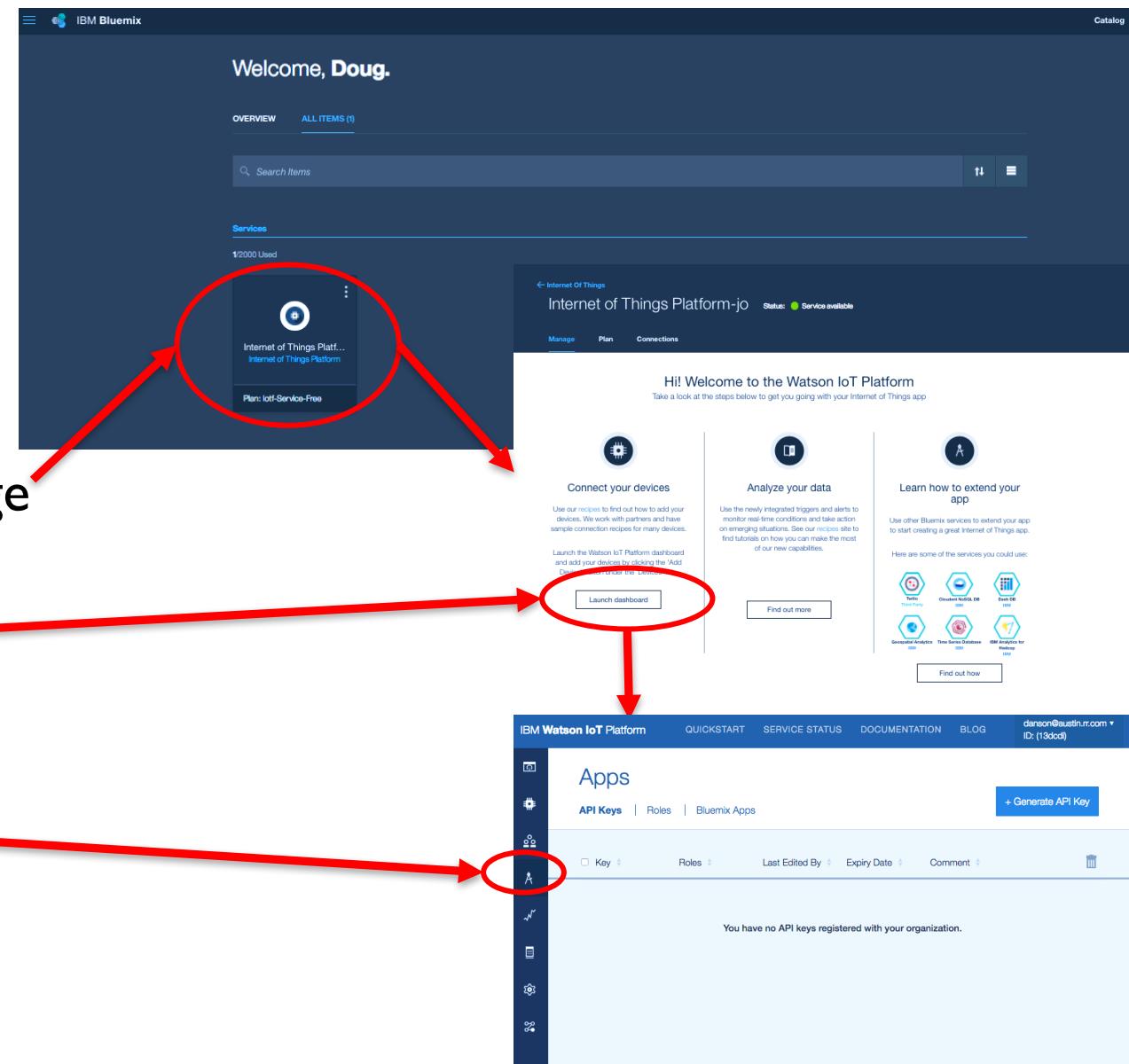
The image consists of three screenshots from the Bluemix dashboard illustrating the steps to create a Watson IoT instance.

- Screenshot 1:** Shows the left sidebar menu with "Internet of Things" highlighted. A red arrow points from the "Get Started" button on the main dashboard to this menu item.
- Screenshot 2:** Shows the "Internet of Things Platform" service page. It displays features like connecting devices to the cloud and building apps that talk to devices. A red arrow points from the "Leave unbound" dropdown menu to the "Pricing Plans" section.
- Screenshot 3:** Shows the "Pricing Plans" section for the Watson IoT instance. It lists a "Standard" plan with details: "The Standard service plan for Internet of Things Platform includes your free tier of 100 MB each of data exchanged, data analyzed and edge data analyzed per month at no cost." Below this, it says "Charge per MB of data exchanged (billed by usage in MB)" and "Charge per MB of edge data analyzed". A red arrow points from the "Standard" plan to the "Create" button at the bottom right.

This will create your Watson IoT instance

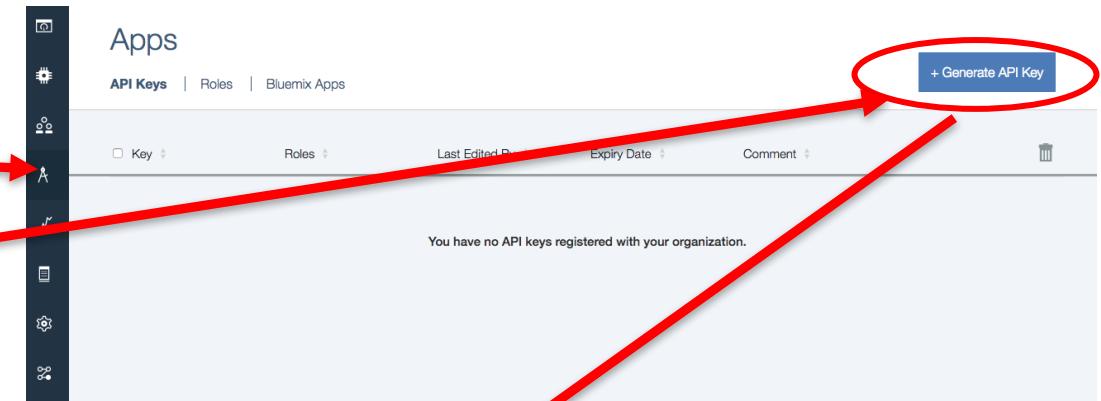
# Create Watson IoT Application Credentials

- Navigate to the Bluemix dashboard:  
<https://console.ng.bluemix.net>



# Create Watson IoT Application Credentials...

- Press "Apps"



- Press “Generate API Key”

- Record the API Key

Generate API Key

API Key

Authentication Token

a-bo522z-smaoymkkzn

7Pkf\_tv@?K@cpO3(1k

- Record the Authentication Token

- Press “Generate” after adding a comment

Comment

My NodeRED Application Key

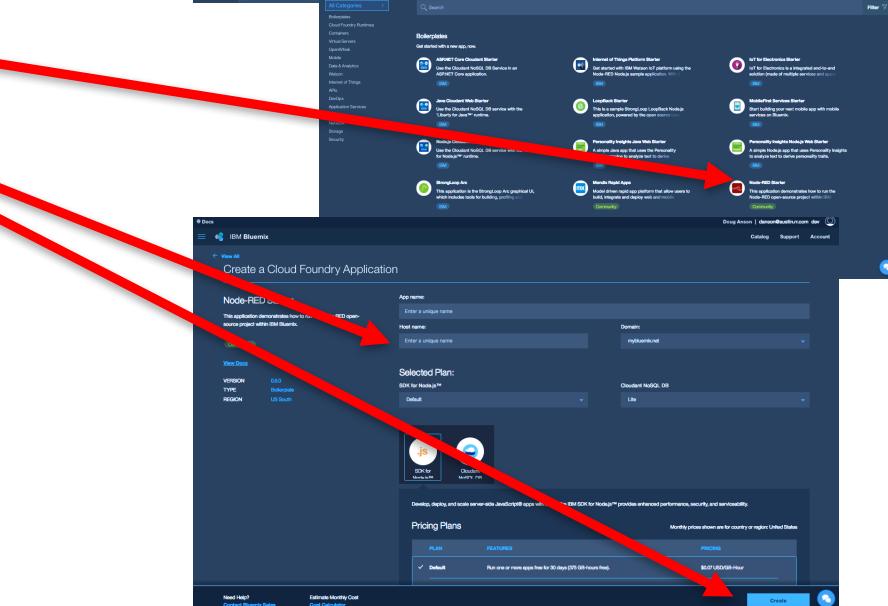
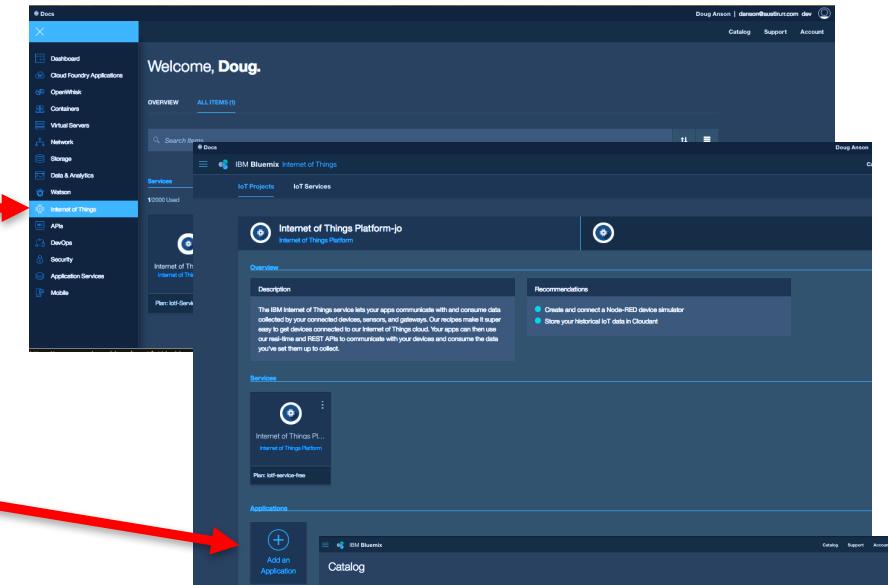
Set API key expiry

10/13/2016

- Save these two values! You will need them later...

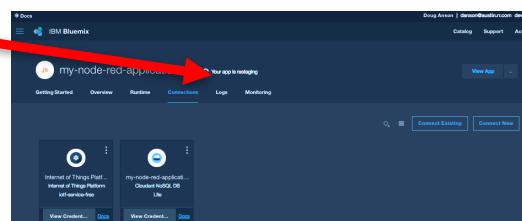
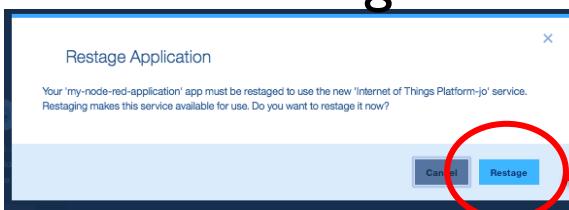
# Creating our Sample Watson IoT NodeRED Application

- Navigate to our Bluemix Dashboard: <https://console.ng.bluemix.net>
- Select “Internet of Things”
- Click on “Add an Application”
- Select “NodeRED Starter”
- Enter an “App Name” (must be unique), Press “Create”
  - Application will “Stage”... this will take a few minutes...
  - Application will report “Running” when the staging has completed
- Go back to the dashboard and Save the new applications’ URL:



# Bind the Watson NodeRED Application to our Watson IoT instance

- Navigate to the Bluemix dashboard: <https://console.ng.bluemix.net>
- Select your NodeRED application badge
- Press “Connections”, then “Connect Existing”
- Check your Watson IoT instance
- Press “Connect”
- Your NodeRED application will “restage”... this will take awhile.  
Wait for the “Running” state:

A series of three screenshots illustrating the connection process:

- The first screenshot shows the "my-node-red-application" badge with a red arrow pointing to its badge.
- The second screenshot shows the "Connections" tab selected, with a red arrow pointing to the "Connect Existing" button.
- The third screenshot shows the "Connect Existing Service" dialog with the "Internet of Things Platform IBM" service selected, and a red arrow pointing to the "Connect" button.

# Create our mbed Connector Access/API Token

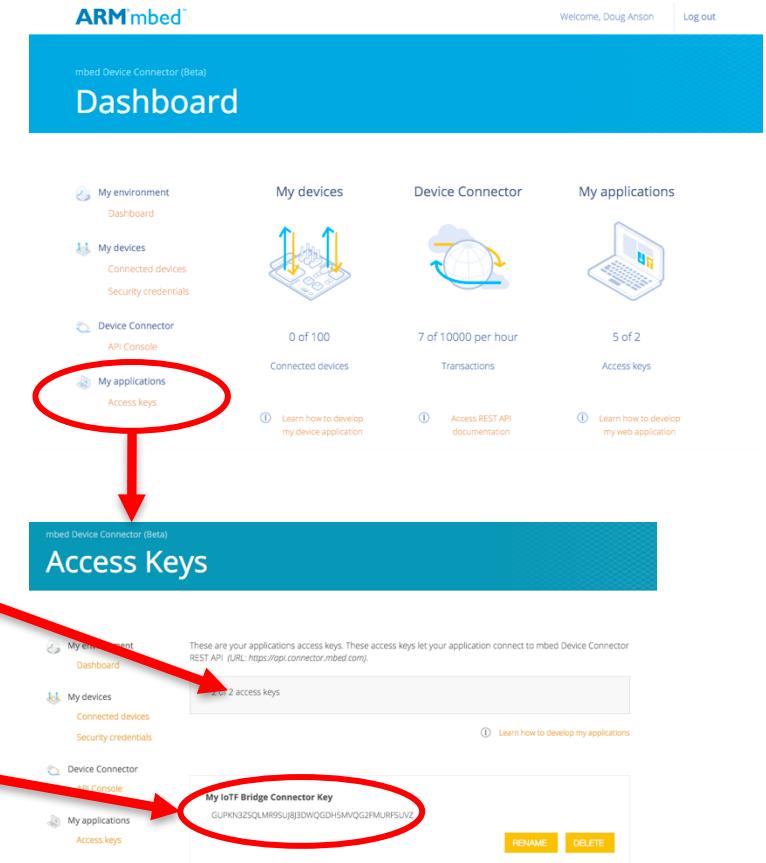
- Navigate to the mbed Connector Dashboard: <https://connector.mbed.com>

- Log in

- Select “Access Keys”... you will create a new token

- Generate an API Token, give it a name

- Save the API Token



# Acquire our mbed Connector "DOMAIN" value...

- Navigate to the mbed Connector Dashboard: <https://connector.mbed.com>

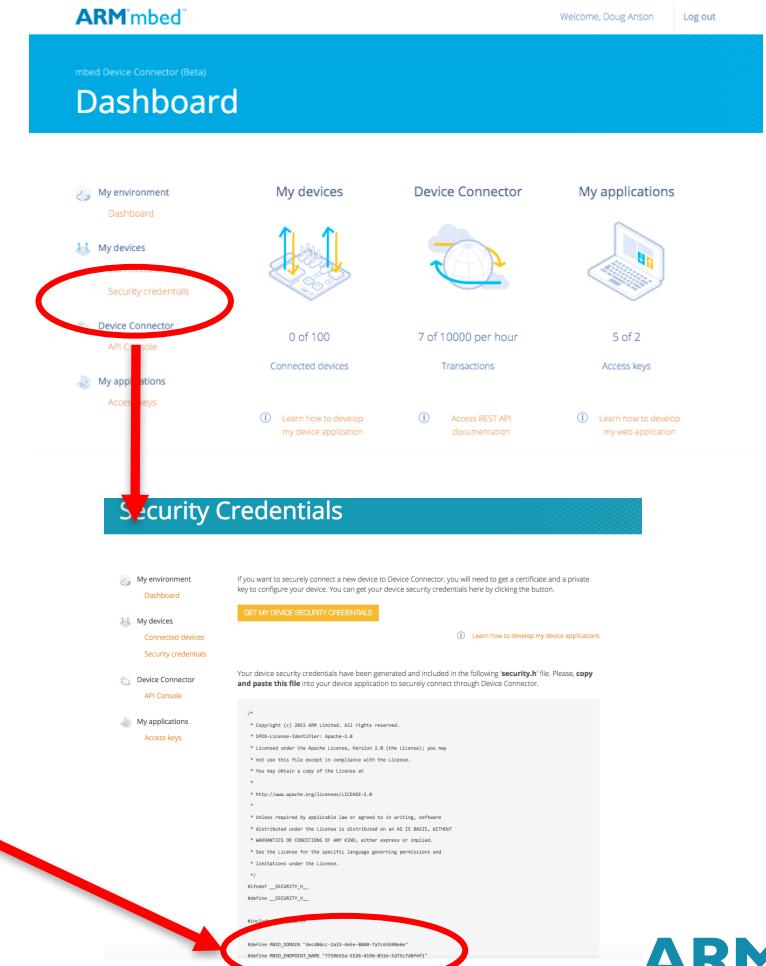
- Log in

- Select “Security Credentials”

- Select “Get my device security credentials”

- Record the **MBED\_DOMAIN** value

- Alternatively, you can use the saved **MBED\_DOMAIN** value you saved previously, it will be the same



# Configure the Watson IoT ARM mbed Connector Bridge

- Go to Watson IoT Dashboard via Bluemix: <https://console.ng.bluemix.net>

The image consists of three main screenshots of the IBM Watson IoT Platform interface:

- Screenshot 1:** Shows the Watson IoT Platform dashboard with the "Extensions" icon highlighted by a red circle. A red arrow points from the "Add Extension" button in the top right corner of the dashboard to the "Add Extension" button in the "Add A New Extension" dialog.
- Screenshot 2:** Shows the "Add A New Extension" dialog. The "ARM mbed Connector" option is selected and highlighted with a red arrow pointing to its "Add" button.
- Screenshot 3:** Shows the "Configure ARM mbed Connector" dialog. It displays fields for "Access Key" and "Domain Id", both of which are annotated with red arrows pointing to the "MBED\_DOMAIN" step in the list below. At the bottom, a "Check Connection" button is also annotated with a red arrow pointing to the "Check Connection" step in the list.

**Configuration Steps:**

- Paste your Connector API Token
- Paste your **MBED\_DOMAIN**
- Press “Check Connection”
- If OK, press “Done”

# Status Check

## **So far we've completed the following**

- Created our own Watson IoT Instance within our Bluemix account
- Created our Watson IoT Application Credentials (used in the NodeRED application...)
- Created our sample Watson IoT NodeRED Application
- Bound the Watson IoT Application to our Watson IoT instance/service
- Created a mbed Connector API Token
- Acquired our mbed Connector MBED\_DOMAIN value
- Configured the Watson IoT ARM mbed Connector Bridge

Next, we will Import our NodeRED flow sample and restart the Endpoint... We should then see device telemetry flowing from the device, through Connector, through the Bridge, and into Watson IoT!

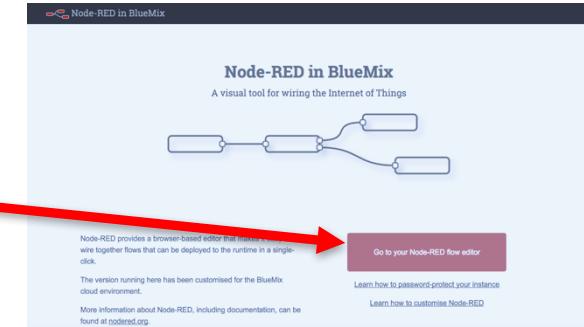
# Importing the NodeRED Flow Example

# Copy the Sample NodeRED Flow JSON

- Go back to your Online IDE workspace
- Go to your “mbed-ethernet-sample-techcon2016” project
- Double-click on NODEFLOW.txt
- Copy all of that file...

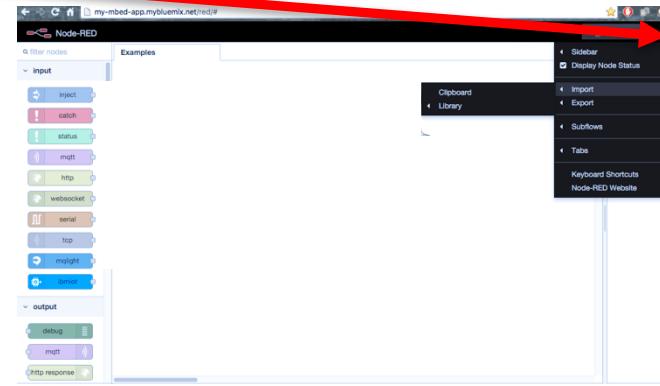
# Import the NodeRED Flow

- Navigate to the URL that we recorded earlier for your Watson IoT NodeRED Application



- Select “go to your NodeRED flow editor”

- Select the menu (far right)



- Select “Import”

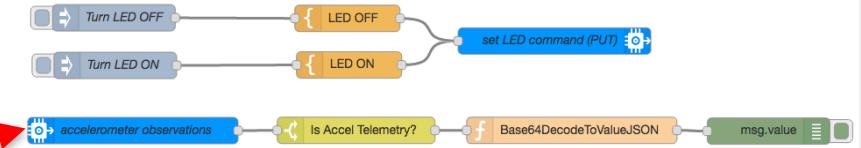
- Select “Clipboard”

- Paste the entire contents of the JSON code you copied in the previous slide into the “paste nodes here” window... then press “OK”.

- If you have trouble copy/pasting the NODEFLOW.txt file contents, try copying it from <https://github.com/ARMmbed/mbed-ethernet-sample-techcon2016> (NODEFLOW.txt is listed there...click & copy...)

# Your new Watson IoT Application Node Flow: Configure it

- Configure your NodeRED flow input and output nodes (Blue) and link them to your Watson IoT instance



- Click on the observation node

accelerometer observations

Is Accel Telemetry?

Base64DecodeToValue.JSON

msg.value

Edit ibmiot in node

Authentication API Key MyWatsonIoT

Input Type Device Event

Device Type mbed-endpoint

Device Id cc69e7c5-c24f-43cf-8365-8d23bb01c

Event observation

Format json

Name connector-bridge source (observations)

Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to 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

Ok Cancel

- Click on the “edit” button

- Provide a name

- Insert your Watson API Key from slide 17

- Insert your Watson Auth Token from slide 17

- Press “Update” to save

MyWatsonIoT

a-uzttt4-28lb0hgffy

.....

5 nodes use this config

Delete Update Cancel

Edit ibmiot config node

\* Ethernet

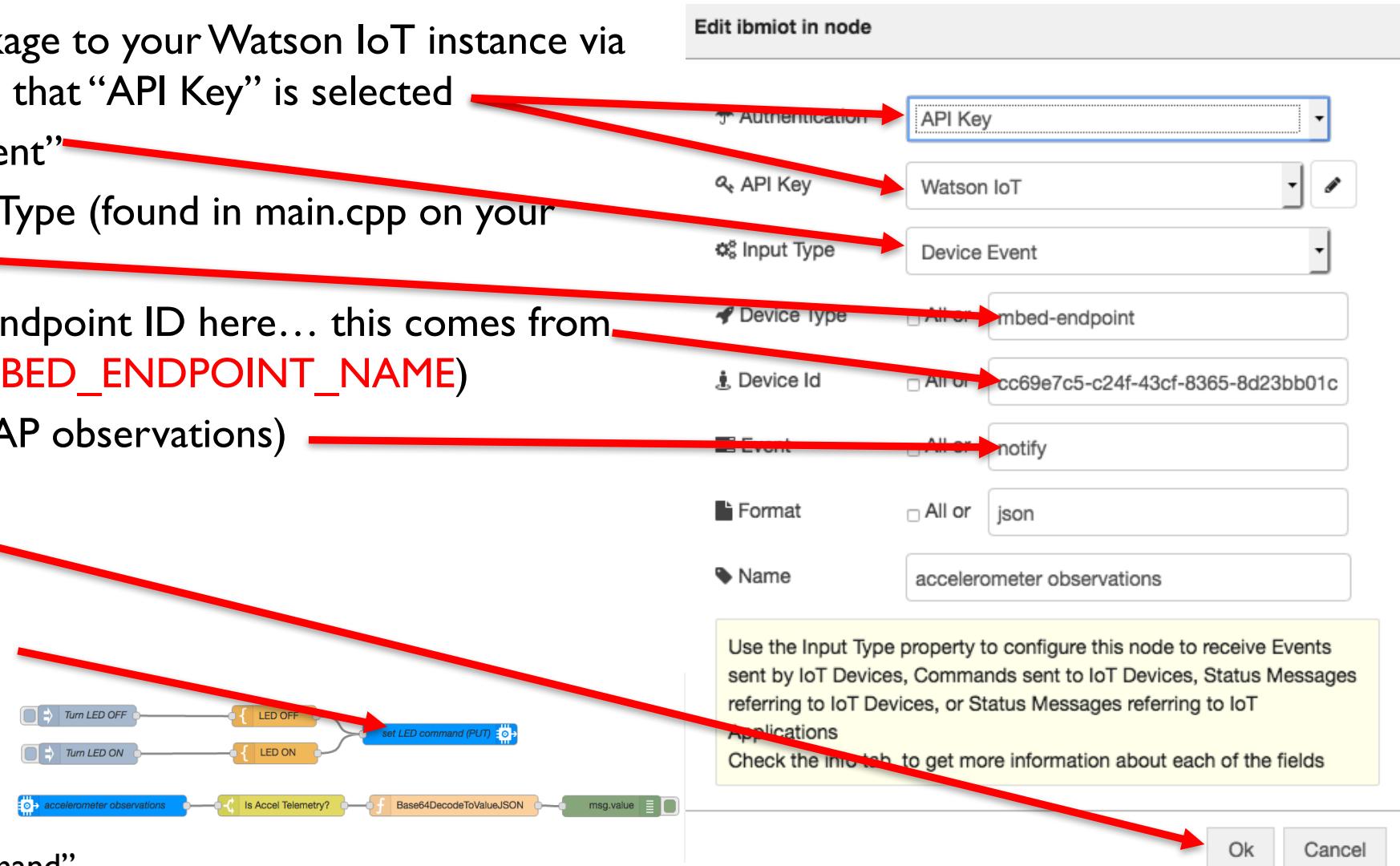
Name MyWatsonIoT

API Key a-uzttt4-28lb0hgffy

API Token .....

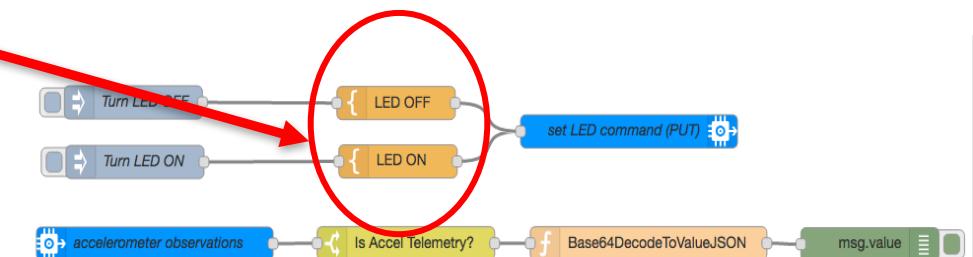
# Your new Watson IoT Application Node Flow: Configure it...

- Now, select the new linkage to your Watson IoT instance via the drop down here and that “API Key” is selected
- Input Type is “Device Event”
- You can filter by Device Type (found in main.cpp on your endpoint) or “all”
- Ensure you have your endpoint ID here... this comes from the security.h header (**MBED\_ENDPOINT\_NAME**)
- Event is “notify” (i.e. CoAP observations)
- Press “OK”
- Edit the blue PUT node
  - API Key Auth
  - Select same API Key
  - Device Type, Device ID
  - Event is “PUT” (all caps!)
  - Input Type is “Device Command”



# Your new Watson IoT Application Node Flow: Configure it...

- Lastly, we have to look at the remaining “Orange” command builder nodes and update them as well
- For both LED OFF/ON nodes, select and note the JSON payload created. “deviceld” needs to be the value you have for your Endpoint (**MBED\_ENDPOINT\_NAME**)
  - “deviceld” will be on the right-hand side in the JSON structure... you have to scroll to the right to see it... You can also enlarge the edit window...
- Locate **MBED\_ENDPOINT\_NAME\_Goes\_Here** and replace it with your actual **MBED\_ENDPOINT\_NAME** value ... press “Done” when complete.
- FYI, Each LED ON/OFF has a Base64 encoded value
  - It’s a “1” for ON, “0” for OFF... each must be Base64 encoded
- Press the red “Deploy” button in the upper right hand side of your NodeRED console



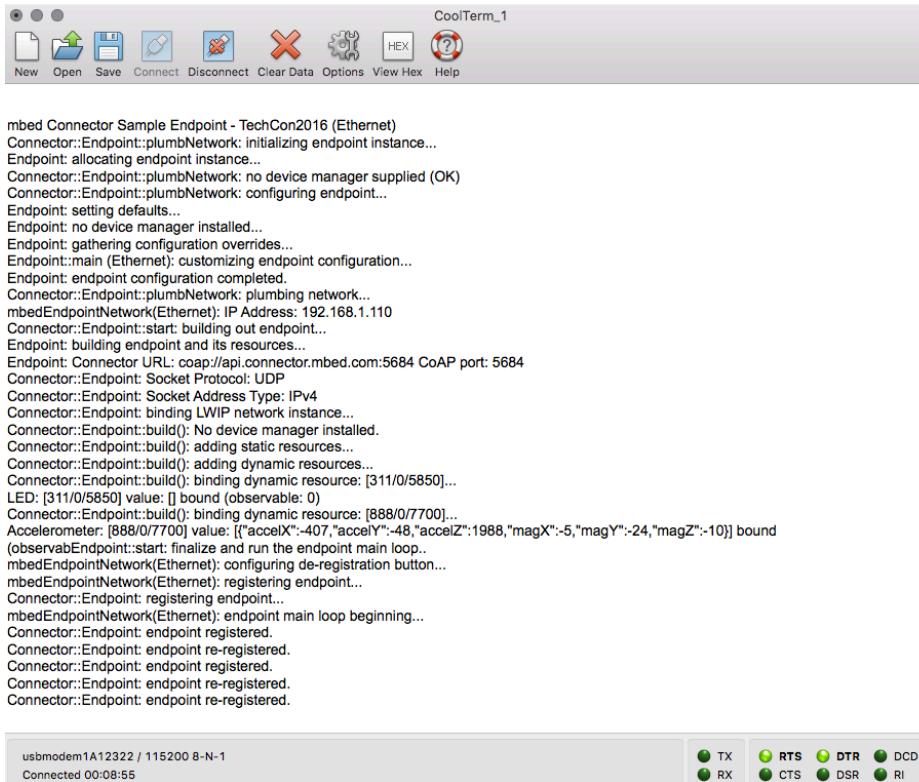
# Putting it all Together

- Press the “reset” button on your mbed device (next to the USB port)
- On your NodeRED Editor, select the “debug” window

Lets check how things are working...

# Putting it all Together: Check the Serial Terminal

- You should see output that looks something like this:
  - Make sure that you see a message like “endpoint registered”



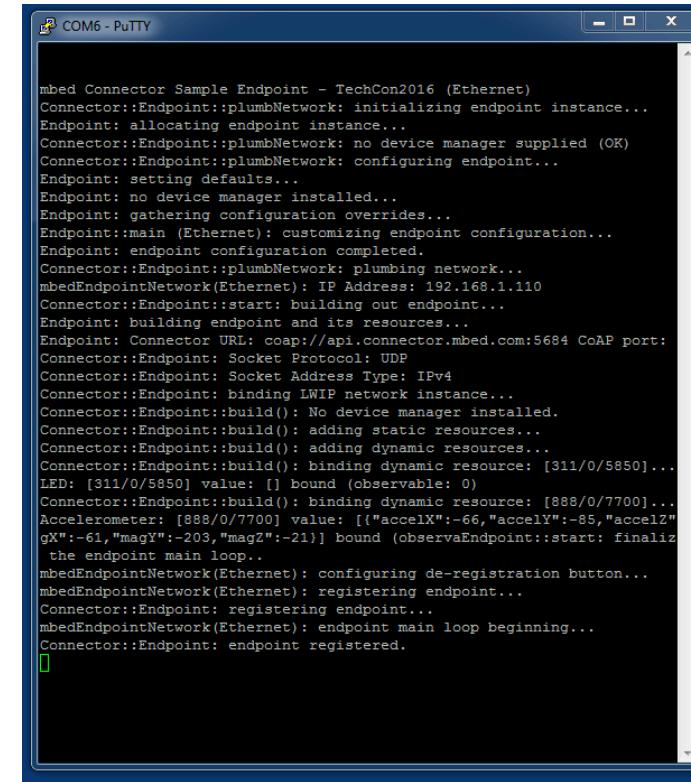
CoolTerm\_1

New Open Save Connect Disconnect Clear Data Options View Hex Help

```
mbed Connector Sample Endpoint - TechCon2016 (Ethernet)
Connector::Endpoint::plumbNetwork: initializing endpoint instance...
Endpoint: allocating endpoint instance...
Connector::Endpoint::plumbNetwork: no device manager supplied (OK)
Connector::Endpoint::plumbNetwork: configuring endpoint...
Endpoint: setting defaults...
Endpoint: no device manager installed...
Endpoint: gathering configuration overrides...
Endpoint::main (Ethernet): customizing endpoint configuration...
Endpoint: endpoint configuration completed.
Connector::Endpoint::plumbNetwork: plumbing network...
mbedEndpointNetwork(Ethernet): IP Address: 192.168.1.110
Connector::Endpoint::start: building out endpoint...
Endpoint: building endpoint and its resources...
Endpoint: Connector URL: coap://api.connector.mbed.com:5684 CoAP port: 5684
Connector::Endpoint: Socket Protocol: UDP
Connector::Endpoint: Socket Address Type: IPv4
Connector::Endpoint: binding LWIP network instance...
Connector::Endpoint::build(): No device manager installed.
Connector::Endpoint::build(): adding static resources...
Connector::Endpoint::build(): adding dynamic resources...
Connector::Endpoint::build(): binding dynamic resource: [311/0/5850]...
LED: [311/0/5850] value: [] bound (observable: 0)
Connector::Endpoint::build(): binding dynamic resource: [888/0/7700]...
Accelerometer: [888/0/7700] value: [{"accelX": -407, "accelY": -48, "accelZ": 1988, "magX": -5, "magY": -24, "magZ": -10}] bound
(observaEndpoint::start: finalize and run the endpoint main loop..
mbedEndpointNetwork(Ethernet): configuring de-registration button...
mbedEndpointNetwork(Ethernet): registering endpoint...
Connector::Endpoint: registering endpoint...
mbedEndpointNetwork(Ethernet): endpoint main loop beginning...
Connector::Endpoint: endpoint registered.
Connector::Endpoint: endpoint re-registered.
Connector::Endpoint: endpoint registered.
Connector::Endpoint: endpoint re-registered.
Connector::Endpoint: endpoint re-registered.
```

usbmodem1A12322 / 115200 8-N-1  
Connected 00:08:55

TX RTS DTR DCD  
RX CTS DSR RI

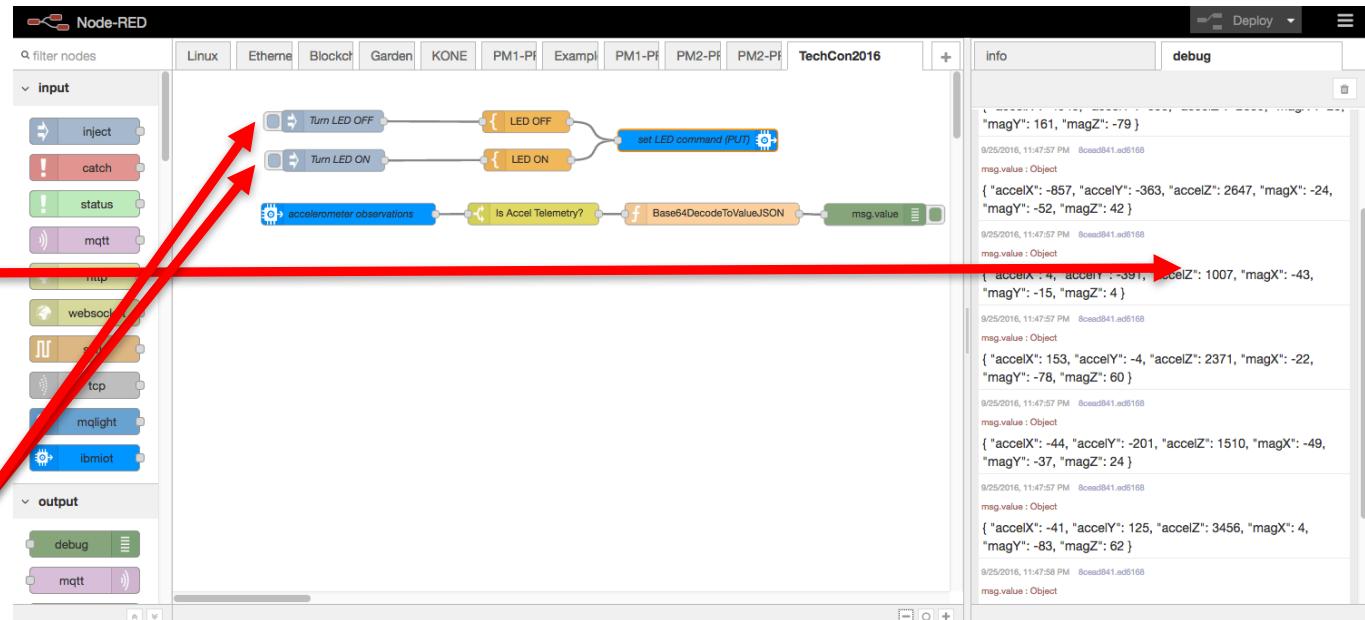


COM6 - PuTTY

```
mbed Connector Sample Endpoint - TechCon2016 (Ethernet)
Connector::Endpoint::plumbNetwork: initializing endpoint instance...
Endpoint: allocating endpoint instance...
Connector::Endpoint::plumbNetwork: no device manager supplied (OK)
Connector::Endpoint::plumbNetwork: configuring endpoint...
Endpoint: setting defaults...
Endpoint: no device manager installed...
Endpoint: gathering configuration overrides...
Endpoint::main (Ethernet): customizing endpoint configuration...
Endpoint: endpoint configuration completed.
Connector::Endpoint::plumbNetwork: plumbing network...
mbedEndpointNetwork(Ethernet): IP Address: 192.168.1.110
Connector::Endpoint::start: building out endpoint...
Endpoint: building endpoint and its resources...
Endpoint: Connector URL: coap://api.connector.mbed.com:5684 CoAP port: 5684
Connector::Endpoint: Socket Protocol: UDP
Connector::Endpoint: Socket Address Type: IPv4
Connector::Endpoint: binding LWIP network instance...
Connector::Endpoint::build(): No device manager installed.
Connector::Endpoint::build(): adding static resources...
Connector::Endpoint::build(): adding dynamic resources...
Connector::Endpoint::build(): binding dynamic resource: [311/0/5850]...
LED: [311/0/5850] value: [] bound (observable: 0)
Connector::Endpoint::build(): binding dynamic resource: [888/0/7700]...
Accelerometer: [888/0/7700] value: [{"accelX": -66, "accelY": -85, "accelZ": -61, "magX": -203, "magY": -21}]] bound
(observaEndpoint::start: finalize and run the endpoint main loop..
mbedEndpointNetwork(Ethernet): configuring de-registration button...
mbedEndpointNetwork(Ethernet): registering endpoint...
Connector::Endpoint: registering endpoint...
mbedEndpointNetwork(Ethernet): endpoint main loop beginning...
Connector::Endpoint: endpoint registered.
```

# Putting it all Together: NodeRED debug info...

- Navigate to your Watson IoT application and NodeRED flow editor
- Examine the debug window
- You should see output similar to this... telemetry in Watson IoT
- You can toggle your LED On and off by clicking each of these... look in the serial terminal for output from the endpoint

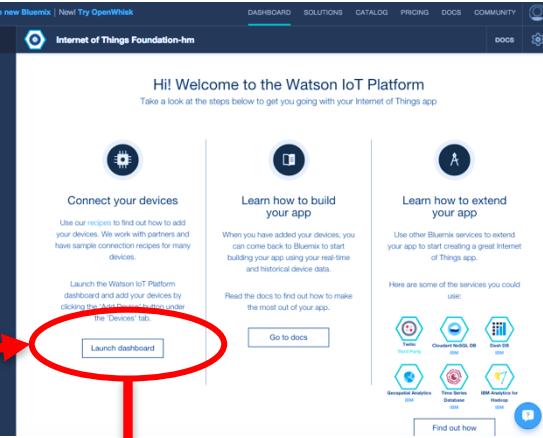
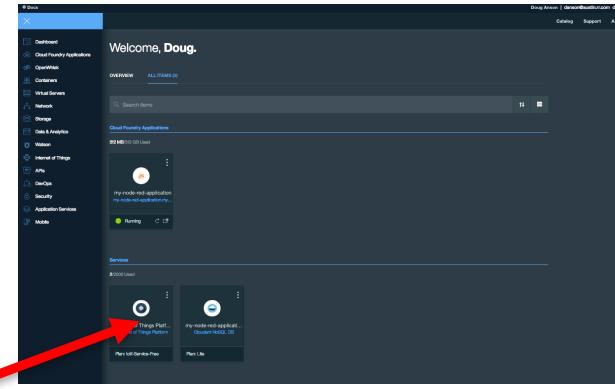


# Putting it all Together: Check Watson IoT Devices

- Navigate to the Bluemix dashboard:

<https://console.ng.bluemix.net>

- Select your Watson IoT Service badge



# Putting it all Together: Check Watson IoT Devices

- You should see something like this page

The screenshot shows the 'Devices' page of the IBM Watson IoT Platform. At the top, there's a navigation bar with links for 'QUICKSTART', 'SERVICE STATUS', 'DOCUMENTATION', 'BLOG', and user information ('danson@austin.rr.com'). Below the navigation is a search bar and a 'Refresh' button. On the left, there's a sidebar with icons for 'Devices', 'Diagnose', 'Actions', 'Device Types', and 'Manage Schemas'. The main area is titled 'Devices' and contains a table with two rows of data. The columns are 'Device ID', 'Device Type', 'Class ID', 'Date Added', and 'Location'. The first device listed is 'cc69e7c5-c24f-43cf-8365-8d23bb01c707' (mbed-endpoint, Device) added on Oct 12, 2016 at 1:08:39 PM. The second device is 'e11cccdde-33e8-4428-960a-4d7994dad082' (parking-meter, Device) added on Oct 12, 2016 at 1:54:07 PM.

Device ID	Device Type	Class ID	Date Added	Location
cc69e7c5-c24f-43cf-8365-8d23bb01c707	mbed-endpoint	Device	Oct 12, 2016 1:08:39 PM	
e11cccdde-33e8-4428-960a-4d7994dad082	parking-meter	Device	Oct 12, 2016 1:54:07 PM	

- Select your device

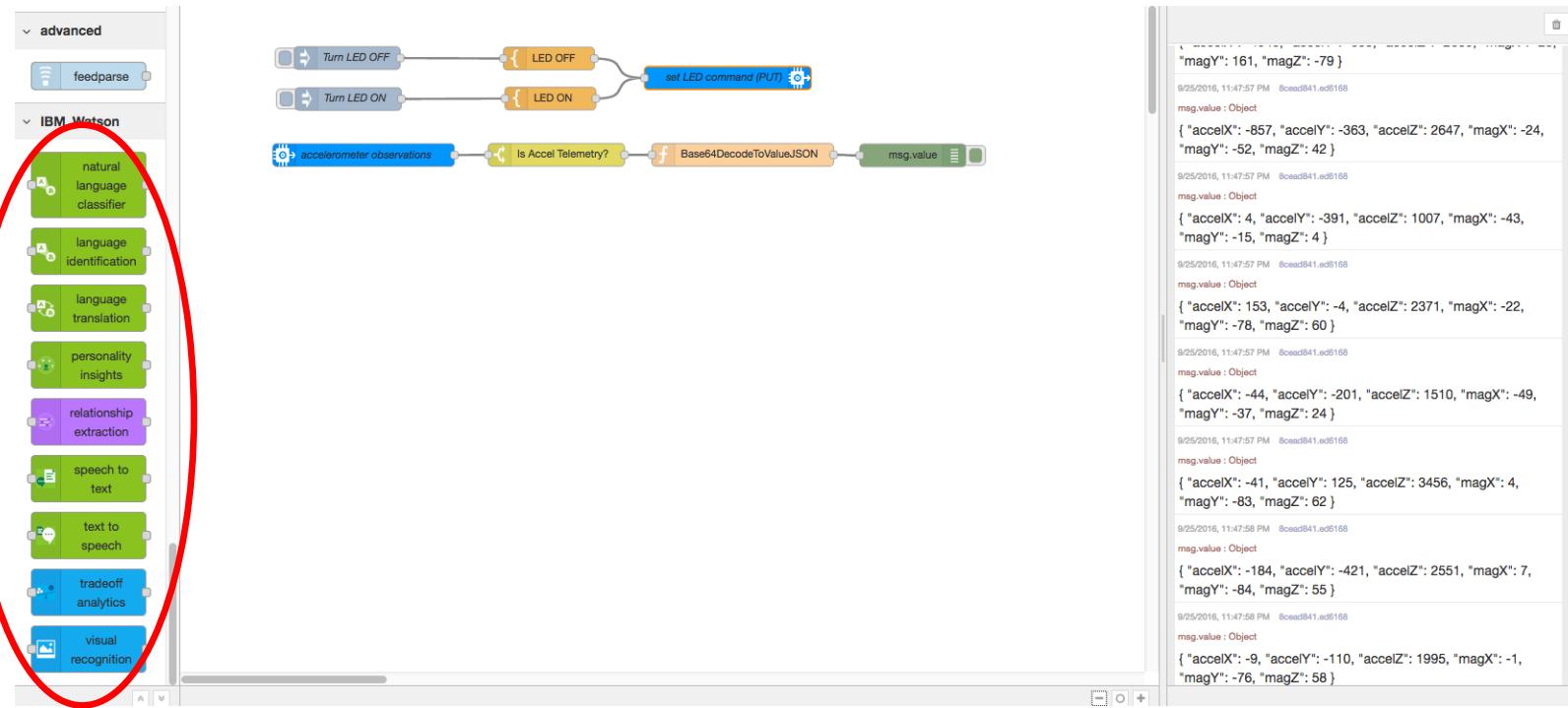
- It's in Watson IoT now!

- You can watch CoAP notify events occurring via the bridge

The screenshot shows a detailed view of a specific device. The top part of the screen displays the device's ID and type: 'Device cc69e7c5-c24f-43cf-8365-8d23bb01c707' (mbed-endpoint). Below this, there are sections for 'Connection Information', 'Recent Events', and 'Device Information'. The 'Recent Events' section lists four 'notify' events received from the device, each with a timestamp: 'Oct 12, 2016 1:19:51 PM', 'Oct 12, 2016 1:19:52 PM', 'Oct 12, 2016 1:19:53 PM', and 'Oct 12, 2016 1:19:53 PM'. The 'Device Information' section provides details like 'Device ID: cc69e7c5-c24f-43cf-8365-8d23bb01c707', 'Device Type: mbed-endpoint', 'Data Added: Wednesday, October 12, 2016', 'Added By: a-kz2dd-ekdkeyDwB0', and 'Connection State: Registered Refresh'.

# Ah Ha!

This is SUPER COOL



- CONGRATS! You have now finished getting mbed device data into IBM Watson IoT.
- We can deliver mbed device data into Watson IoT analytics via our NodeRED flow!
- The bigger challenge remains...What can/do you do with your data telemetry?

# Summary

We have completed the following – congratulations!

- Created our Bluemix mbed environments and PC tool setup
- Imported our mbed endpoint, customized, installed, and ran it
- Created our own Watson IoT Service Instance and NodeRED application
- Configured our Watson mbed Connector Bridge
- Imported a NodeRED flow and customized it
- Examined live telemetry and some bi-directional capabilities of our bridge

Great JOB! Thanks for your time.