

Workshop: Connecting IoT devices to the cloud with IBM Watson IoT and mbed Cloud

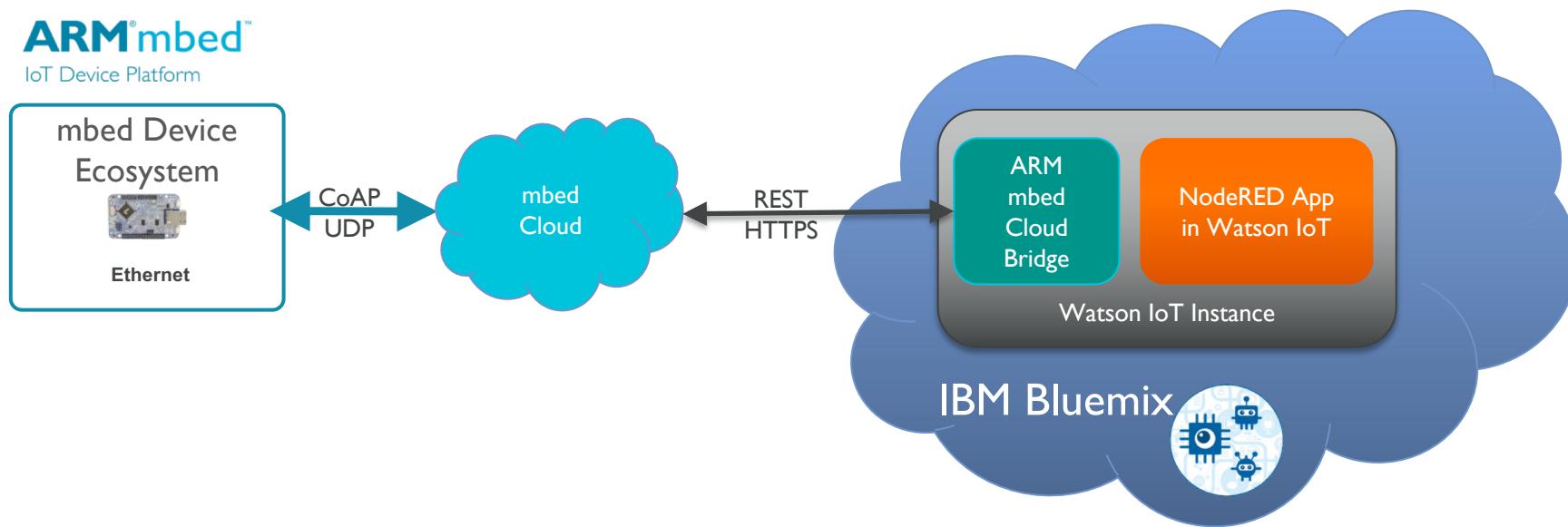


Doug Anson
Solutions Architect / IoT BU / ARM

Brian Daniels
Applications Engineer / IoT BU / ARM

April 26, 2018 – v2.5

What will we build in this Workshop?



- Connect your mbed device into Watson IoT through mbed Cloud and Watson's Cloud Bridge
 - Create a simple mbed device and connect it to mbed Cloud
 - Watson IoT now has a fully integrated mbed Device Cloud Bridge that links mbed Cloud 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 mbed Cloud accounts (should be completed prior to workshop)
- Retrieve a set of provisioning credentials (mbed_cloud_dev_credentials.c)
- Import our mbed sample project into the online IDE
- Update the sample project with the provisioning credentials (mbed_cloud_dev_credentials.c)
- 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 (PTSOOI method...)

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

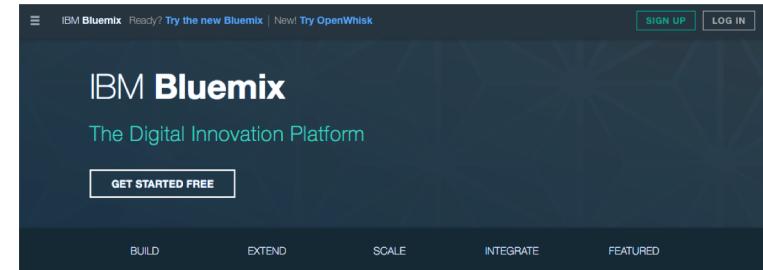
Quick Links: Visit https://github.com/ARMmbed/bridge_workshop_ibm_watson_cloud

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

Create our Bluemix Account

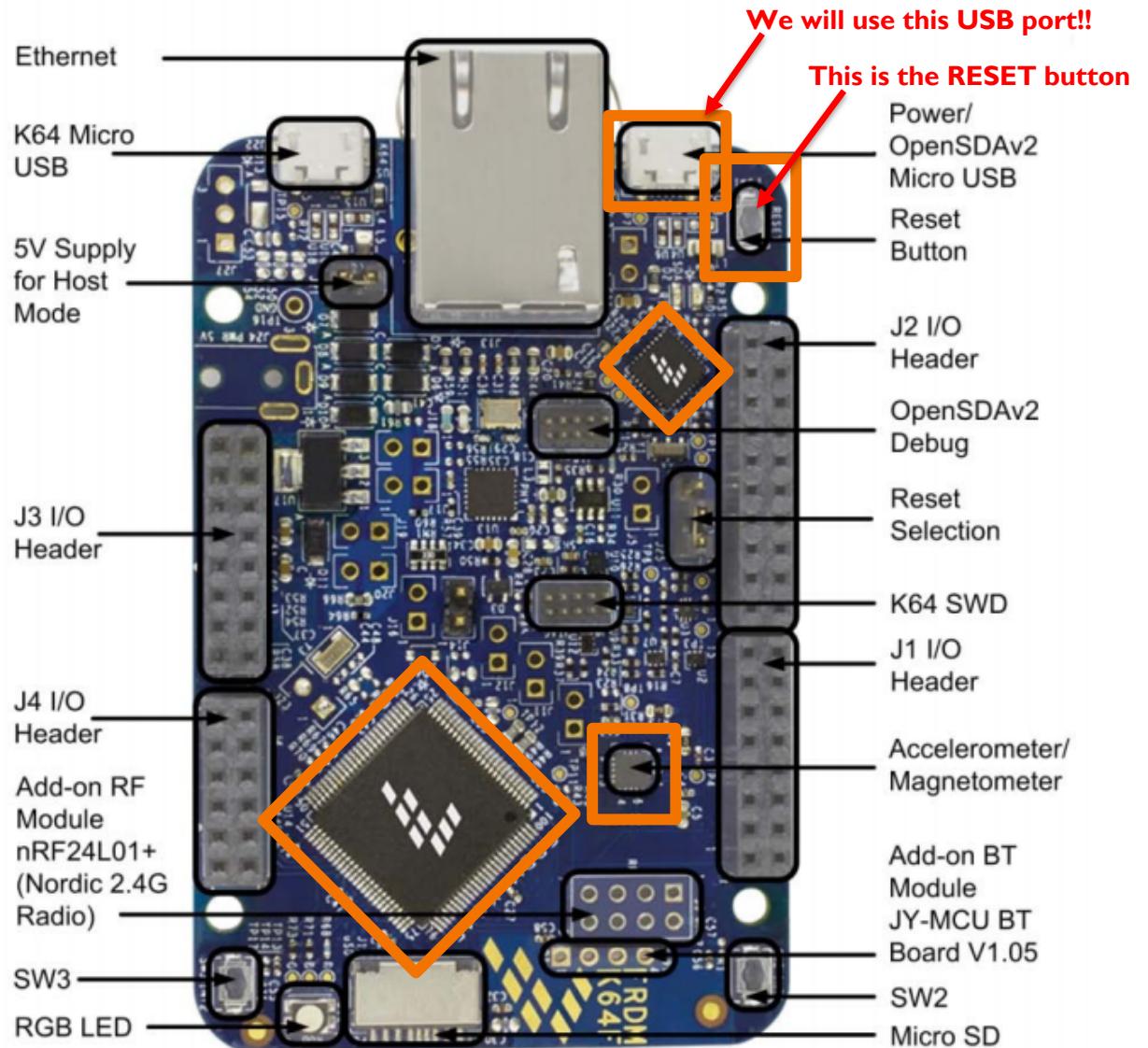
- Navigate to the Bluemix Dashboard:
<https://console.bluemix.net/>
- Press “Sign Up”
- Complete the sign-up process
- A confirmation email will be sent that must be acknowledged
- Log into your Bluemix account and establish your default organization and space

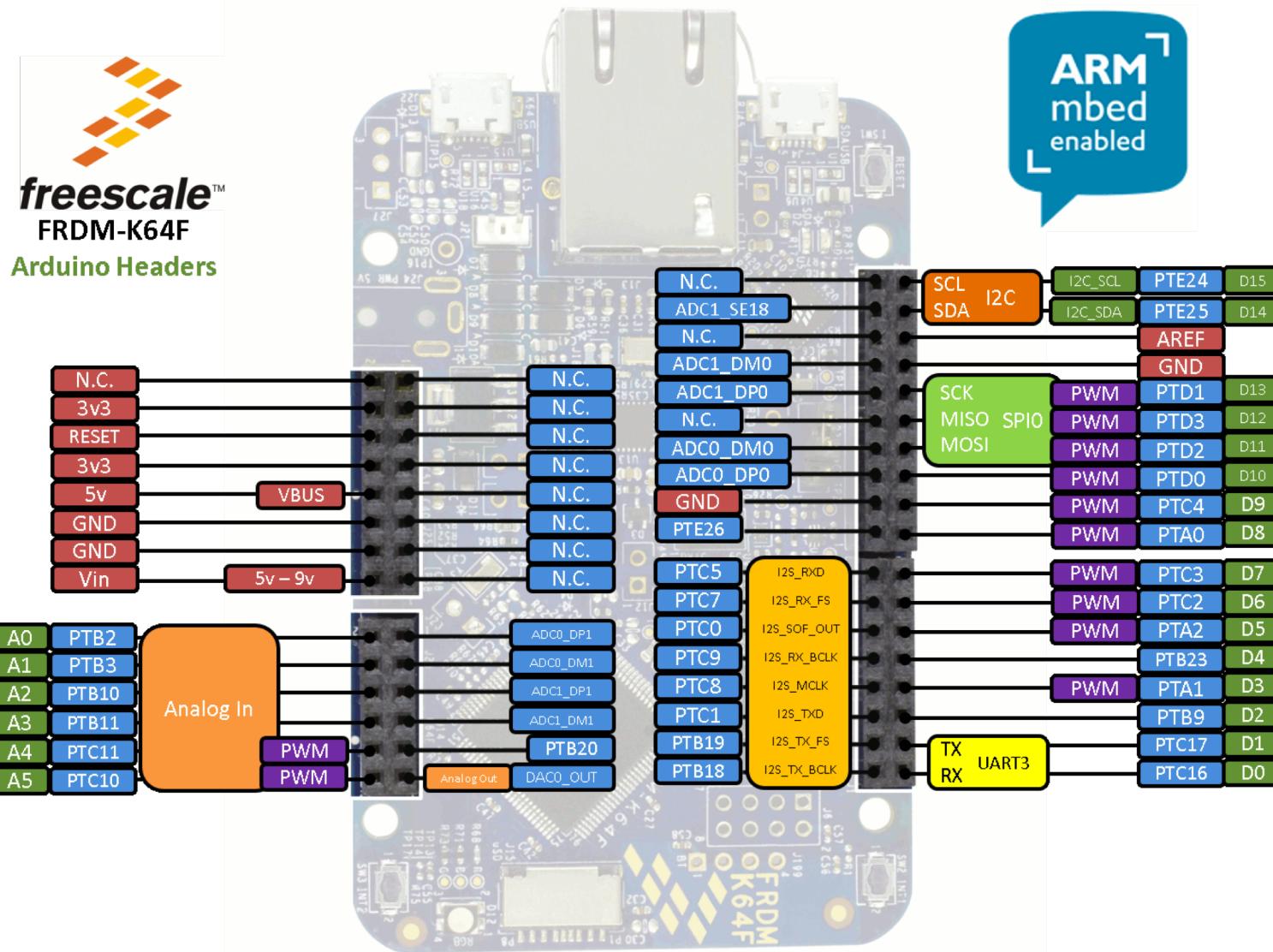
Prior to
Workshop

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

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

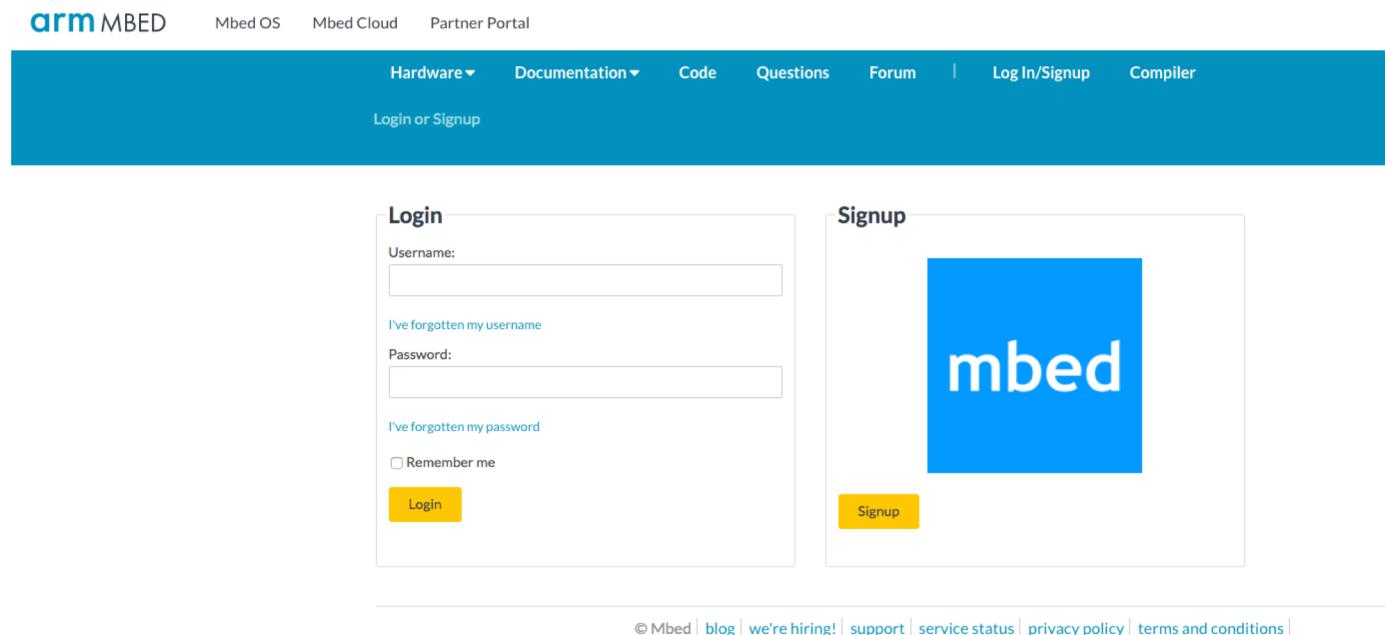
Prior to
Workshop

- **Windows**
 - **IMPORTANT:** Install the mbed USB Serial driver -
https://os.mbed.com/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
 - Chrome and Firefox Browsers installed – **NOTE: IE will NOT WORK... Please use Chrome and/or Firefox**
- **Mac**
 - Serial Terminal: CoolTerm - 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
 - 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

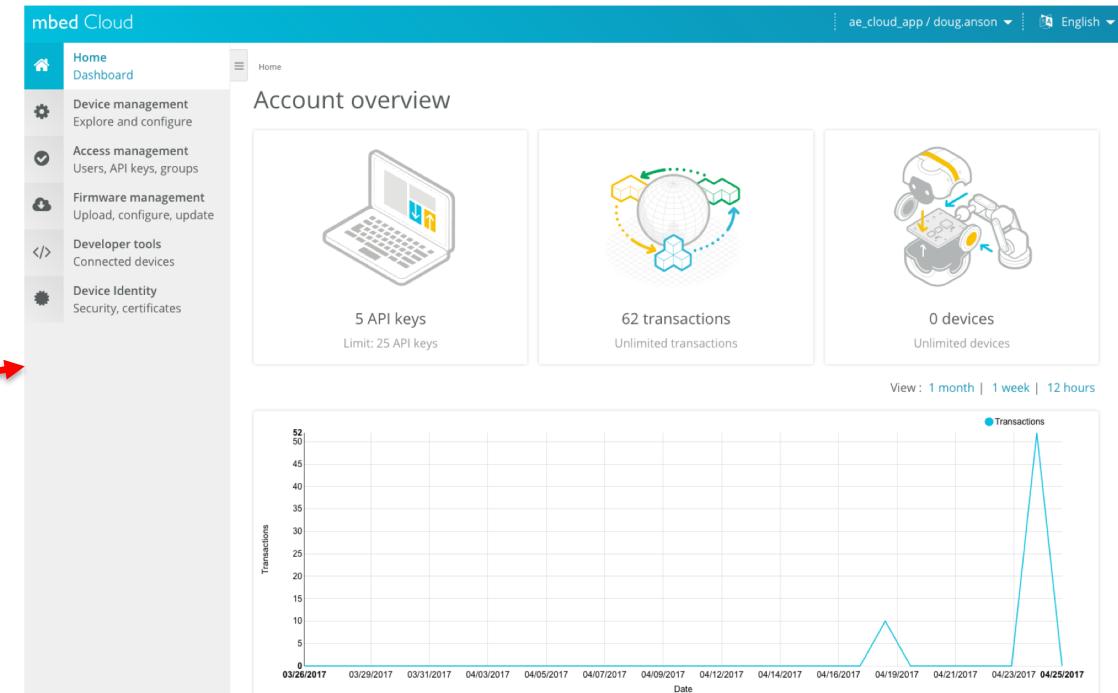
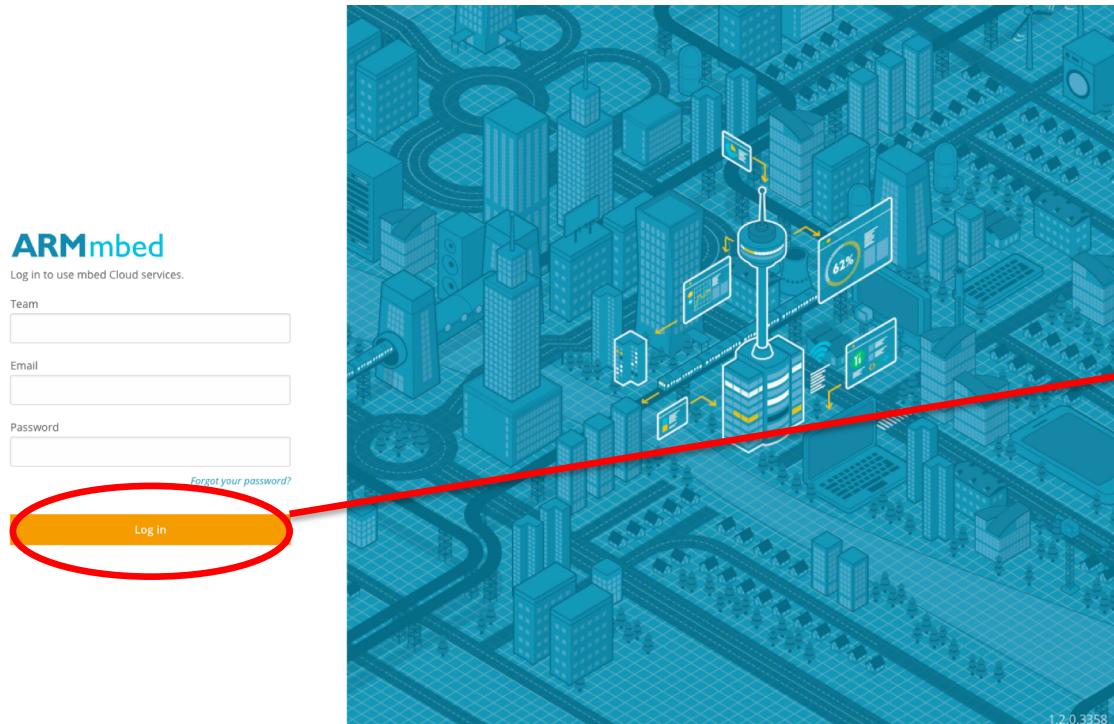
- Navigate to: <https://os.mbed.com/account/login/?next=/>
- Create an Account



Create Your mbed Cloud Account...

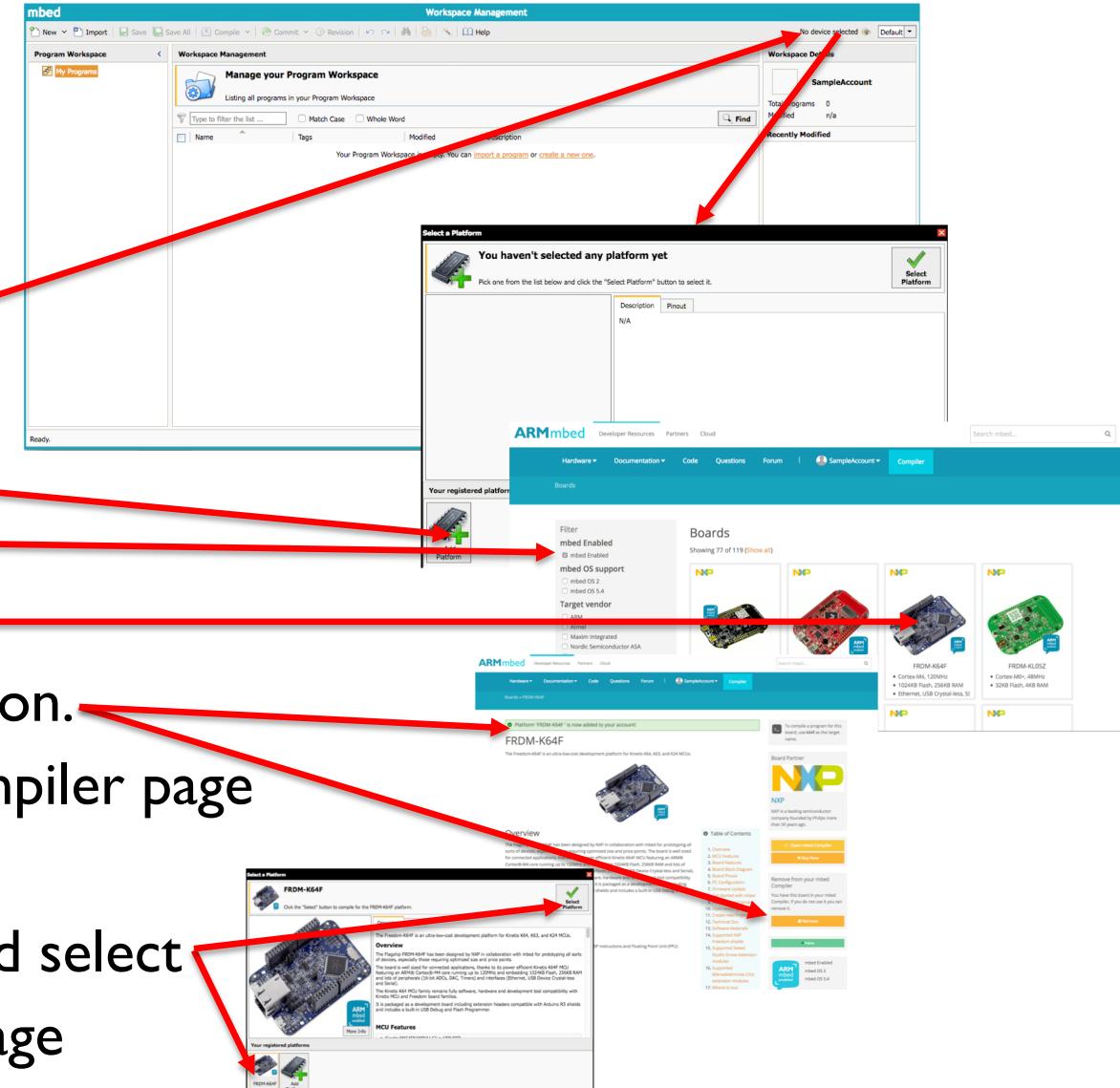
Prior to
Workshop

- Navigate to the mbed Cloud Dashboard: <https://portal.us-east-1.mbedcloud.com>
- Confirm that you can log into your mbed device mbed Cloud dashboard



Log into the Online IDE – Add the K64F Compile Target

- Go to <https://os.mbed.com/>



Import our K64F Endpoint Project

- Go to <https://os.mbed.com/>

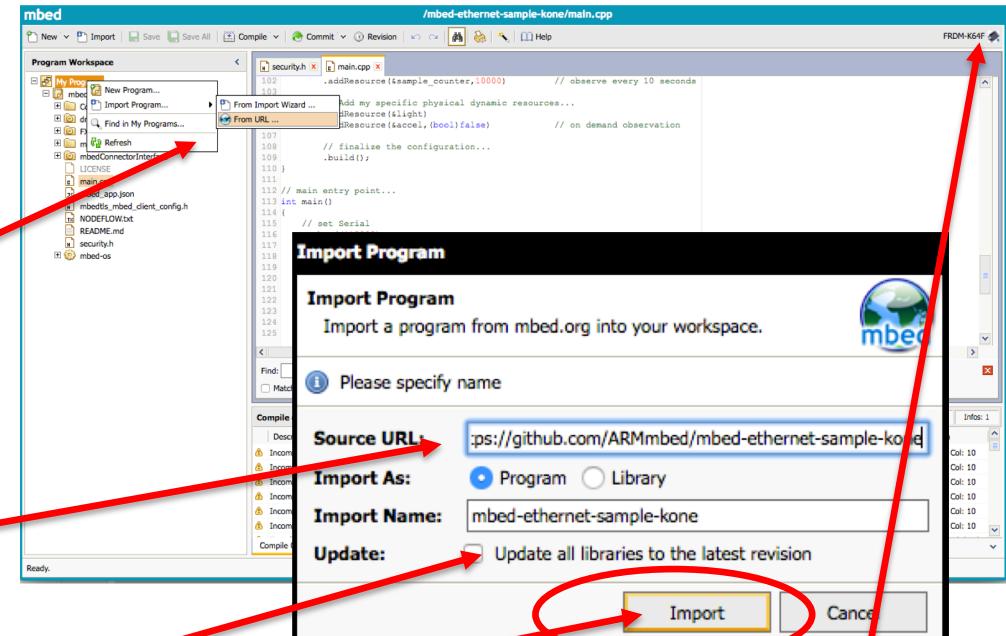
- Select the “Compiler” page

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

- Choose Select “from URL...”

- Enter this URL (Leave the “Update all libraries...” **unchecked**)
<https://github.com/ARMmbed/mbed-cloud-sample/>

- Press “Import”, the ensure that “FRDM-K64F” is selected



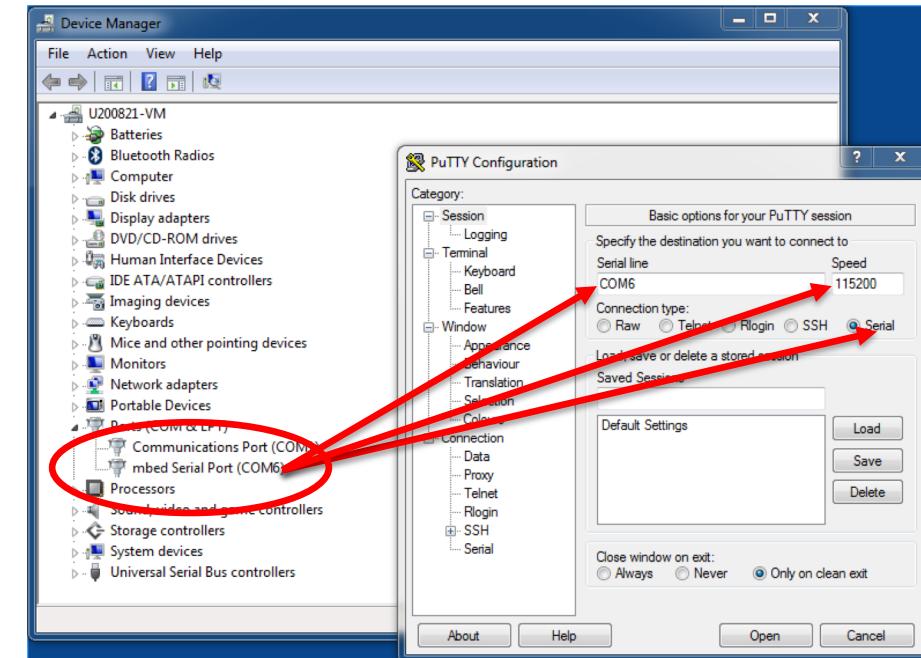
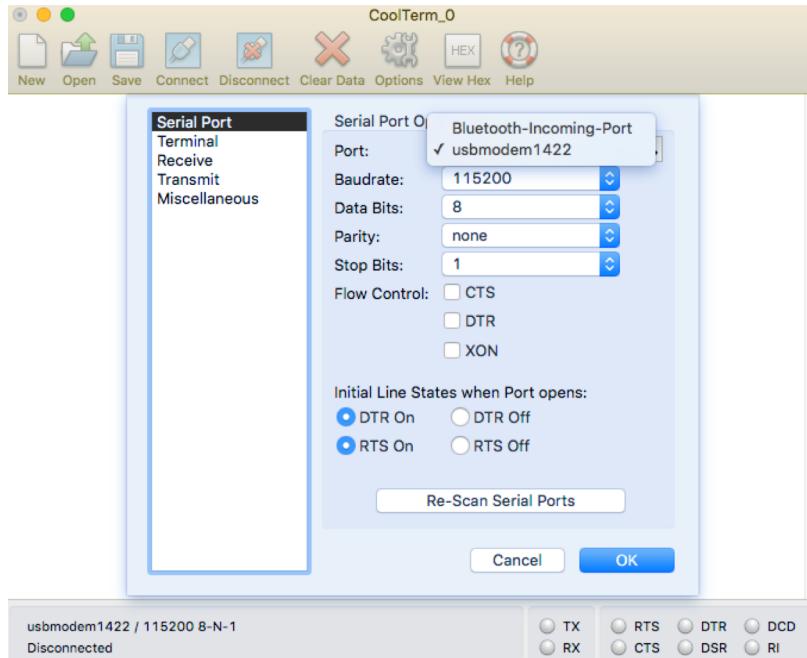
Set Provisioning Credentials in Your Endpoint Code

- Go back to the mbed Device mbed Cloud dashboard: <https://portal.us-east-1.mbedcloud.com>
 - In the left sidebar, select “Device Identity” → “Certificates”
 - Under “Actions”, select “create a developer certificate”
 - Give your developer certificate a name and description... press “create certificate”
 - Press “Download Developer C file” – this will deposit “mbed_cloud_dev_credentials.c” in your downloads directory
- Now, go back to the Compiler page of your online IDE
- Replace `mbed_cloud_dev_credentials.c` with newly downloaded one from the 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 “L
- K64F green LED will flicker for a bit, then stop, and dismount/remount...

The screenshot shows the mbed online IDE interface. The 'Program Workspace' tab is active, displaying a list of files including `main.cpp`, `security.h`, and `security.c`. The 'Code Editor' tab is also visible, showing the contents of `main.cpp`. The code in `main.cpp` includes comments about observing a counter every 10 seconds, initializing configuration, and setting up a main entry point. It also defines a serial port and a device manager. The 'Code Editor' tab has several search and replace tools at the bottom.

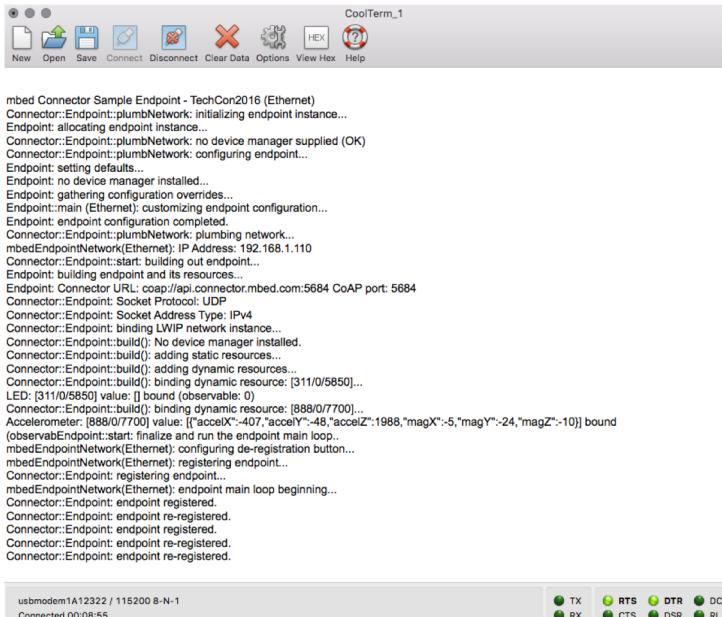
Running your Endpoint Code

- 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 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::bind dynamic resource: [311/0/5850]...

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

Connector::Endpoint::bind dynamic resource: [888/0/7700]...

Accelerometer: [888/0/7700] value: [{"accelX": -61, "accelY": -48, "accelZ": 1988, "magX": -5, "magY": -24, "magZ": -10}] bound (observable: 1)

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

mbedEndpointNetwork(Ethernet): registering endpoint...

Connector::Endpoint::endPointMainLoop...

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

Connector::Endpoint::endpoint registered.

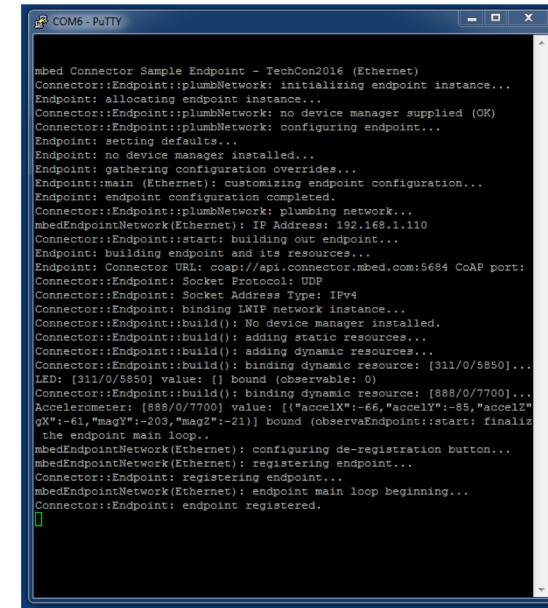
Connector::Endpoint::endpoint re-registered.

Connector::Endpoint::endpoint registered.

Connector::Endpoint::endpoint re-registered.

Connector::Endpoint::endpoint registered.

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: 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: coapt://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::bind dynamic resource: [311/0/5850]...

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

Connector::Endpoint::bind dynamic resource: [888/0/7700]...

Accelerometer: [888/0/7700] value: [{"accelX": -61, "accelY": -48, "accelZ": 1988, "magX": -5, "magY": -24, "magZ": -10}] bound (observable: 1)

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

mbedEndpointNetwork(Ethernet): registering endpoint...

Connector::Endpoint::registering endpoint...

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

Connector::Endpoint::endpoint 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 (mbed_cloud_dev_credentials.c)
- Imported our mbed sample project into the online IDE
- Updated the sample project with the provisioning credentials (mbed_cloud_dev_credentials.c)
- 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 Cloud Bridge...

Watson IoT mbed Cloud 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 Cloud API Key/Token
- Configure the Watson IoT ARM mbed Cloud Bridge

Create our Watson IoT Instance

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

This will create your Watson IoT instance

The screenshots illustrate the process of creating a Watson IoT instance:

- Step 1: Bluemix Dashboard**
The first screenshot shows the Bluemix dashboard with the URL <https://console.bluemix.net>. A red arrow points to the "IBM Bluemix Apps" button in the top left corner.
- Step 2: Services Catalog**
The second screenshot shows the "Services" catalog. A red arrow points to the "Internet of Things" service category in the sidebar.
- Step 3: Internet of Things Service Overview**
The third screenshot shows the "Internet of Things" service overview page. A red arrow points to the "Get Started" button.
- Step 4: Catalog Item Details**
The fourth screenshot shows the "Internet of Things Platform" catalog item details. A red arrow points to the "Create" button at the bottom right.

Create Watson IoT Application Credentials

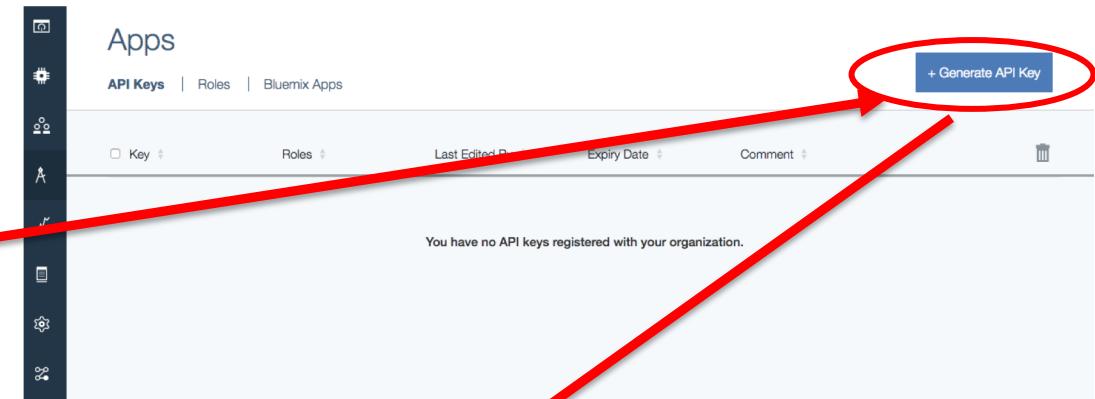
- Once your Watson IoT Instance is created, you should see the Watson IoT dashboard
- Launch the “Launch”
- Press “Apps”

The screenshot shows the IBM Bluemix Internet of Things dashboard for a instance named "my-watson-iot-instance". The dashboard features a central icon of a device connected to a network, with the text "Welcome to Watson IoT Platform" and a brief description: "Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world." Below this is a "Launch" button, which is highlighted with a large red arrow. To the right of the launch button are two links: "Learn about Watson IoT Platform" and "Expand using step-by-step recipes".

The screenshot shows the Watson IoT Platform dashboard with a sidebar menu on the left. The menu items include BOARDS, DEVICES, MEMBERS, APPS, USAGE, RULES, SECURITY, SETTINGS, and EXTENSIONS. The "APPS" item is highlighted with a red arrow. The main area of the dashboard displays four cards: "DEVICE-CENTRIC ANALYTICS" (5 Cards), "USAGE OVERVIEW" (3 Cards), and "RULE-CENTRIC ANALYTICS" (6 Cards). A blue button "+ Create New Board" is located in the top right corner of the main area.

Create Watson IoT Application Credentials...

- 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

Select API Role(s)

Standard Application

+ Add another role

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

Please DO NOT USE Internet Explorer for this task!

- Go back to our Bluemix Dashboard: <https://console.bluemix.net>

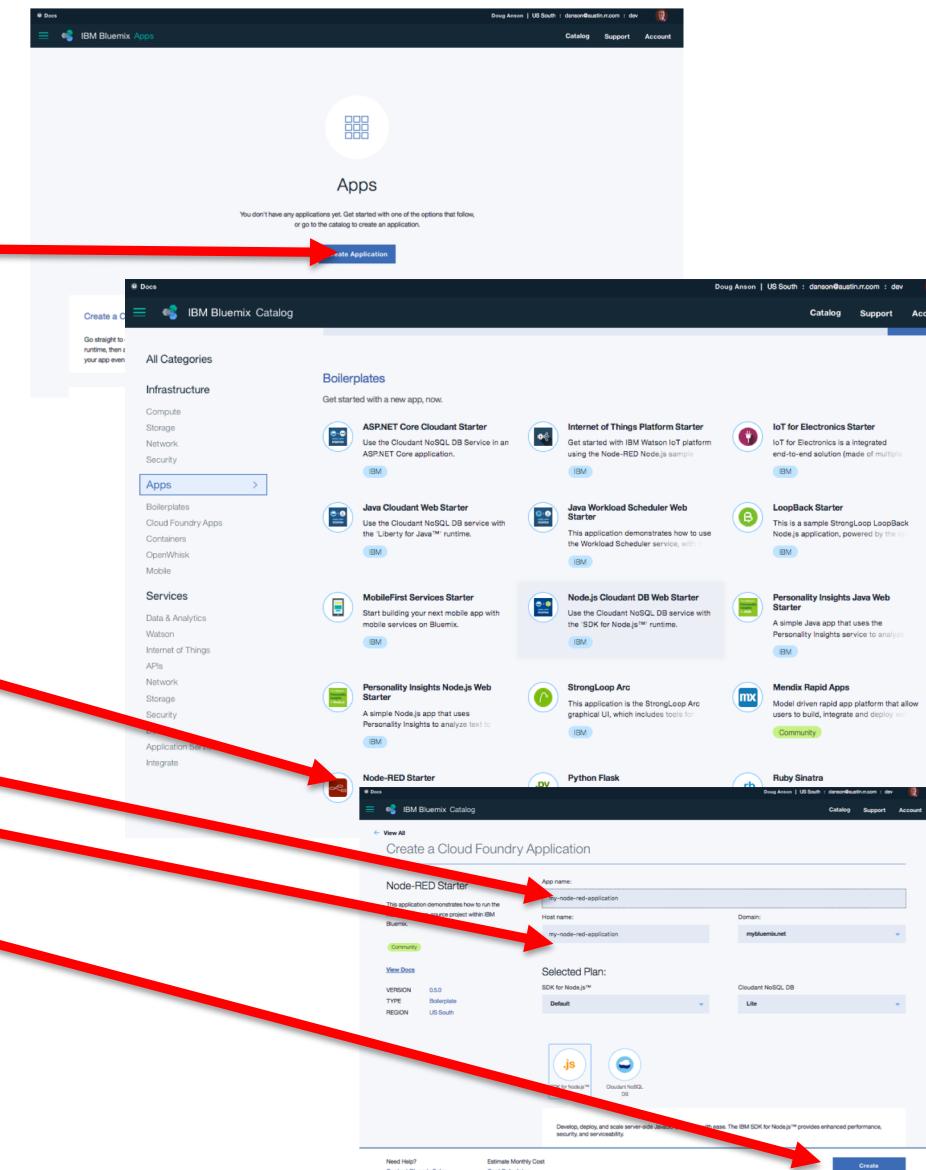
- Click on “Create Application”

- Select “NodeRED Starter”

- Enter an “App Name” (must be unique)
 - Make a note that your app URL is <app name>.mybluemix.net
 - Press “Create” to finalize creating the application

- Application will “Stage”... this will take a few minutes...

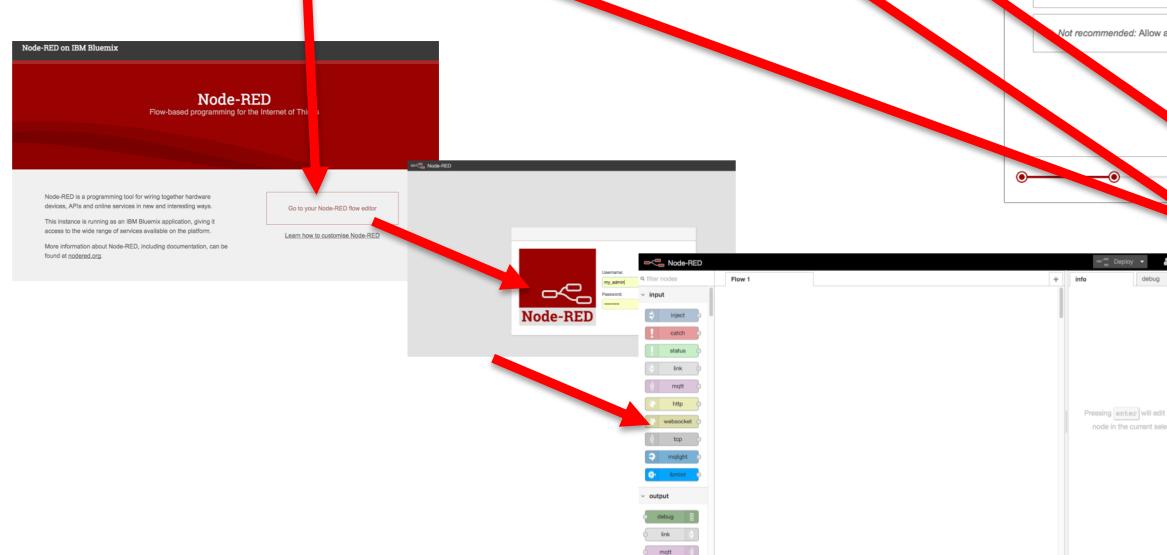
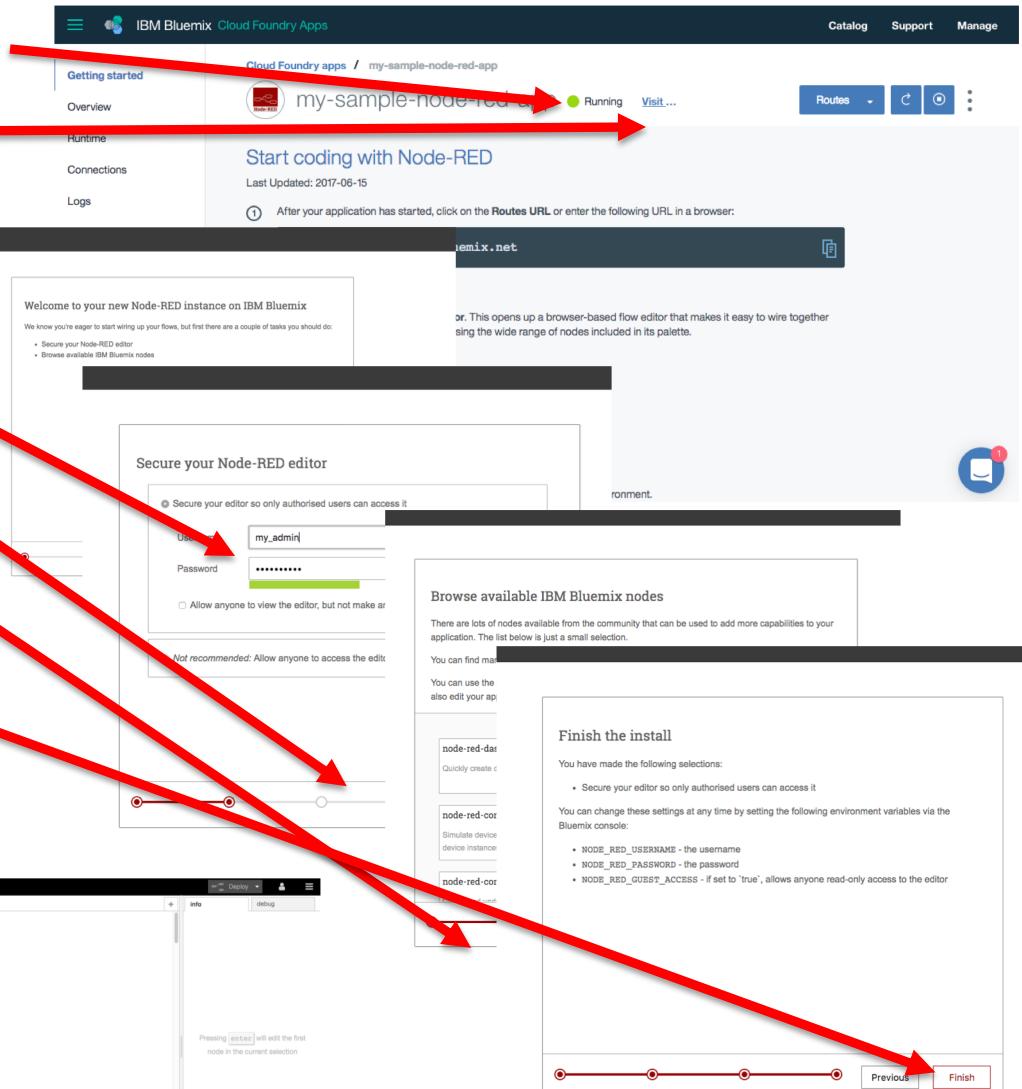
- Next we will configure the NodeRED application (next slide)



Configure Watson IoT NodeRED Application

Please DO NOT USE Internet Explorer for this task!

- Your NodeRED app should fully "stage" and report "running"
- Click on "Visit..."
- Next, you must configure your NodeRED app
 - Create an admin username/password for the dashboard
 - Press "Next"
 - Press "Next" (no additional nodes need to be added)
 - Press "Finish"
- Now, login to the dashboard



Bind the Watson NodeRED Application to our Watson IoT instance

Please DO NOT USE Internet Explorer for this task!

- Dashboard -> Apps, select your NodeRED app

IBM Bluemix Cloud Foundry Apps

Cloud Foundry apps / my-sample-node-red-app

my-sample-node-red-app • Running [Visit...](#)

Runtime

Node-RED

BUILDPACK Node-RED Starter

INSTANCES 1

MB MEMORY PER INSTANCE 512

TOTAL MB ALLOCATION 512 MB still available

Connections (1) my-sample-node-red-app-cloudantNoSQLDB

Connect new Connect existing

Routes \$0.00 \$0.00

- Press “Connect Existing”
- Click on your Watson IoT Instance
- Press “Connect”
- Your NodeRED application will “restage”... this will take awhile.
Wait for the “running” state:

IBM Bluemix Cloud Foundry Apps

Connect existing service

Services

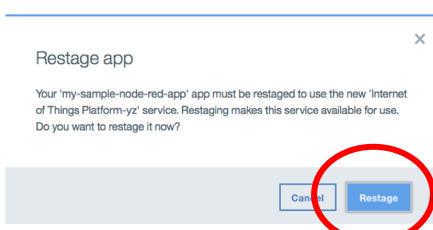
Internet of Things Platform-yz

my-watson-node-red-app-cloud

Visual Recognition-Id

Select compatible service to connect to my-sample-node-red-app

Connect



Create our mbed Cloud Access/API Token

- Navigate to the mbed Cloud Dashboard: <https://portal.us-east-1.mbedcloud.com>

- Log in, Select “Access Management”, then “API keys”

- Press “Create new API Key ”... you will create a key

- Give the new API Key a name

- Select “Developers” group

- Create the API Key

Create new API key

API key name: My New API Key

Groups: Developers

Create API key Cancel

Key name	Groups	Date last connected	Date created
AWS IoT	1	-	April 19, 2017 4:51 PM
Home	1	-	April 19, 2017 4:51 PM
IBM Watson	1	-	April 19, 2017 4:49 PM
MS IoTHub	1	-	April 19, 2017 4:50 PM
Test	1	-	April 19, 2017 4:51 PM

The API key has been created

This will be the last time the API key is available to you, but you can generate a new one from the API Key Details page.

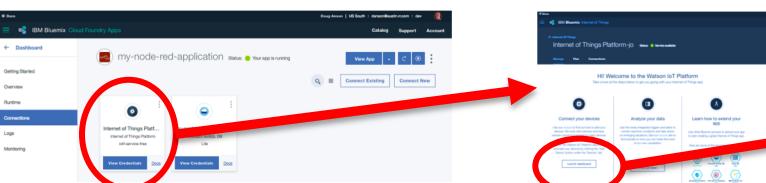
Key name:	My New API Key
API key:	ak_1HDEIYj051x3zG2n0010918m00w4fJ1D00nRQnWDA015bab3e32d02420a012311 00000000key12T2tp12C4P3Lx#Htq0g13a444xz

Copy to clipboard Add another key

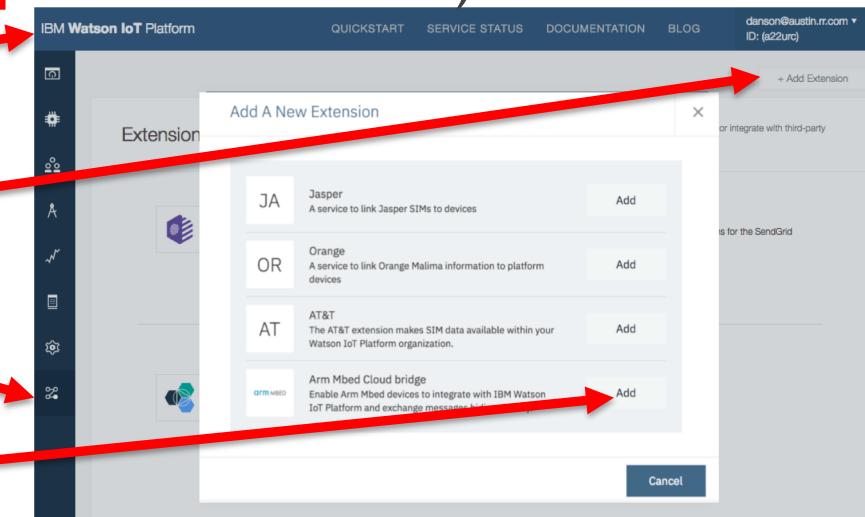
- Press “copy to clipboard” – save to a file for later use

Configure the Watson IoT ARM mbed Cloud Bridge

- Go back to the Watson IoT Dashboard (last window open on slide 21...)

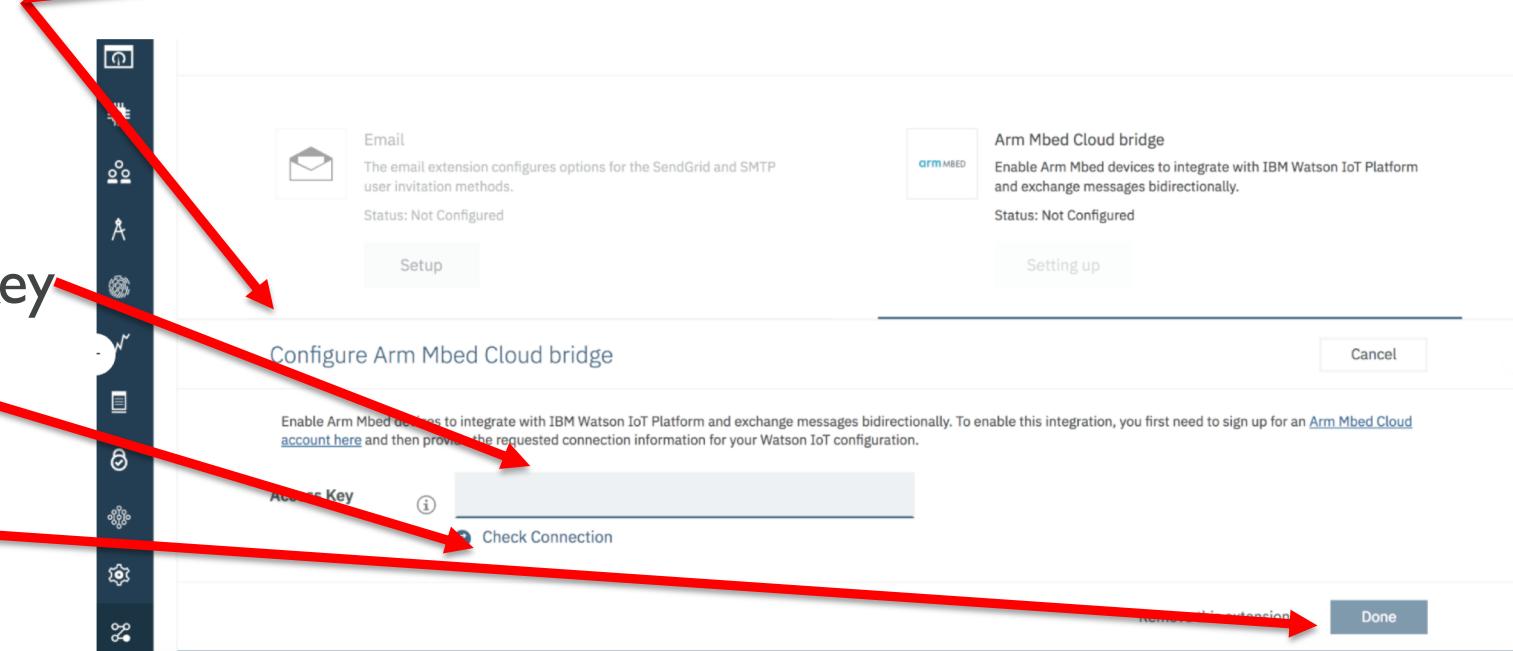


- Select “Extensions” (bottom icon)
Select “Add Extension”



- Select “ARM mbed Cloud”
(Press “add”, then “setup”)
- scroll down a bit...

- Paste your mbed Cloud API Key
- 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 Cloud API Token
- Configured the Watson IoT ARM mbed Cloud Bridge

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

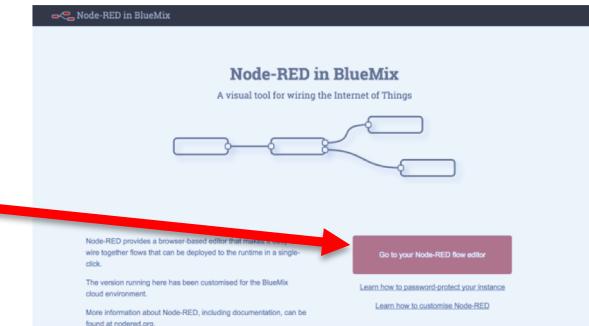
Importing the NodeRED Flow Example

Copy the Sample NodeRED Flow JSON

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

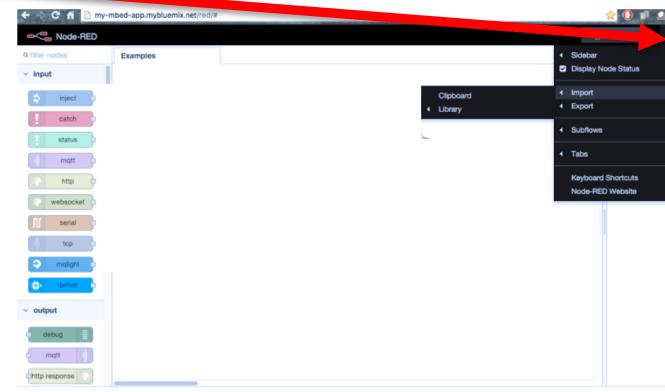
Import the NodeRED Flow

- Navigate to the URL that we recorded earlier for your Watson IoT NodeRED Application (i.e. `http://<app name>.mybluemix.net`)



- Select “go to your NodeRED flow editor”

- Select the menu (far right)



- Select “Import”

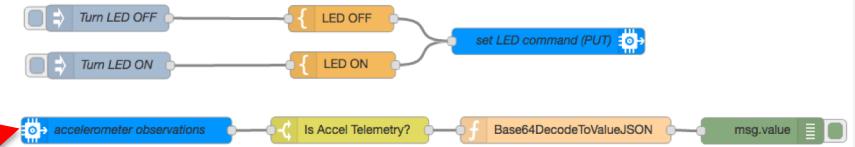
- Select “Clipboard”

- 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-cloud-sample> (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

Edit ibmiot in node

Authentication: API Key
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 20

- Insert your Watson Auth Token from slide 20

- Press “Update” to save

Edit ibmiot config node

* Ethernet

Name: MyWatsonIoT

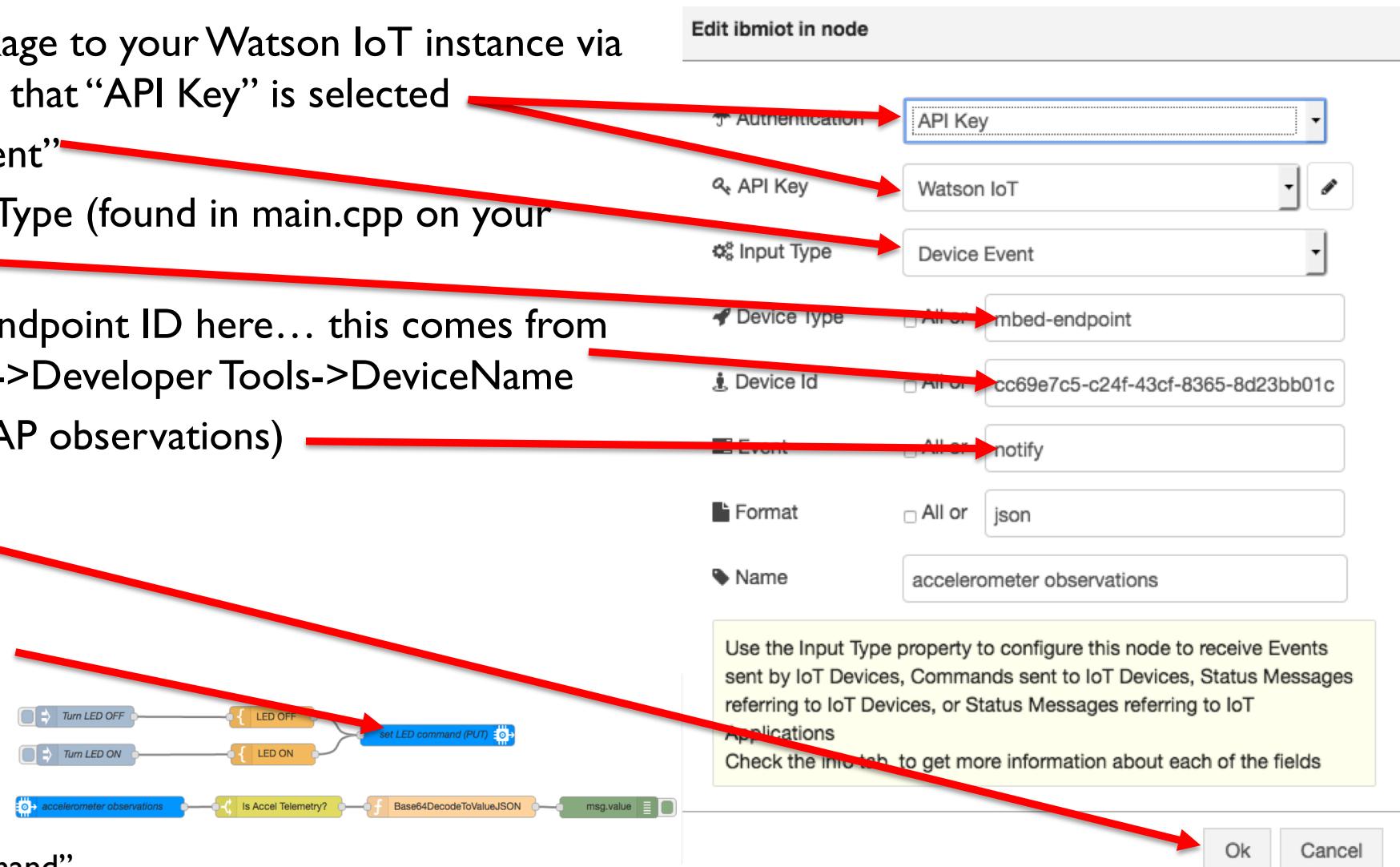
API Key: a-uzttt4-28lb0hgffy

API Token:
5 nodes use this config

Delete Update Cancel

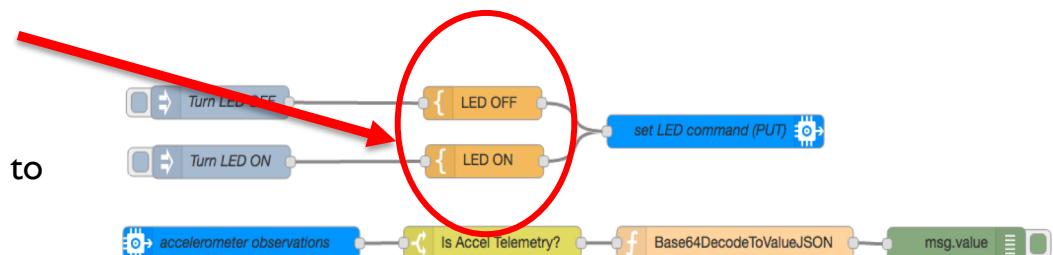
Your new Watson IoT Application Node Flow: Configure it...

- 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 mbed Cloud Dashboard->Developer Tools->DeviceName
- 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 **Device Name** (from mbed Cloud Dashboard→ Developer Tools)
 - “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 **Device 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



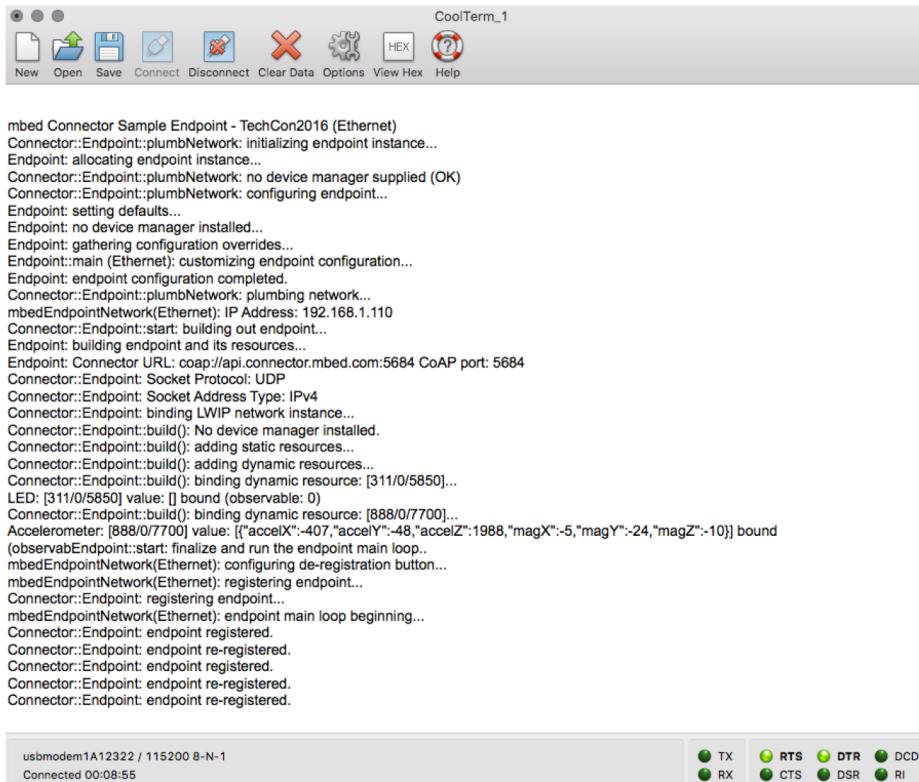
Putting it all Together

- 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”



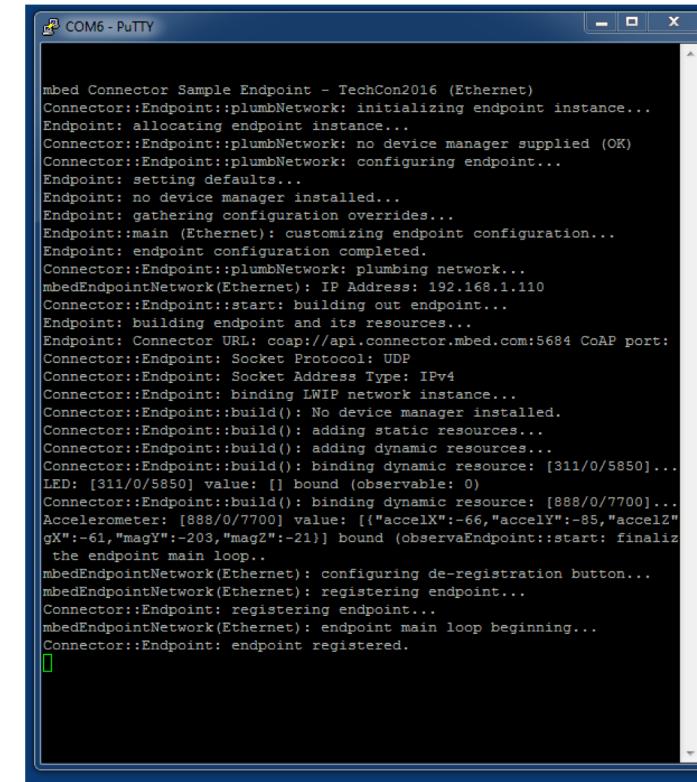
CoolTerm_1

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": 1988, "magX": -61, "magY": -203, "magZ": -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.
```

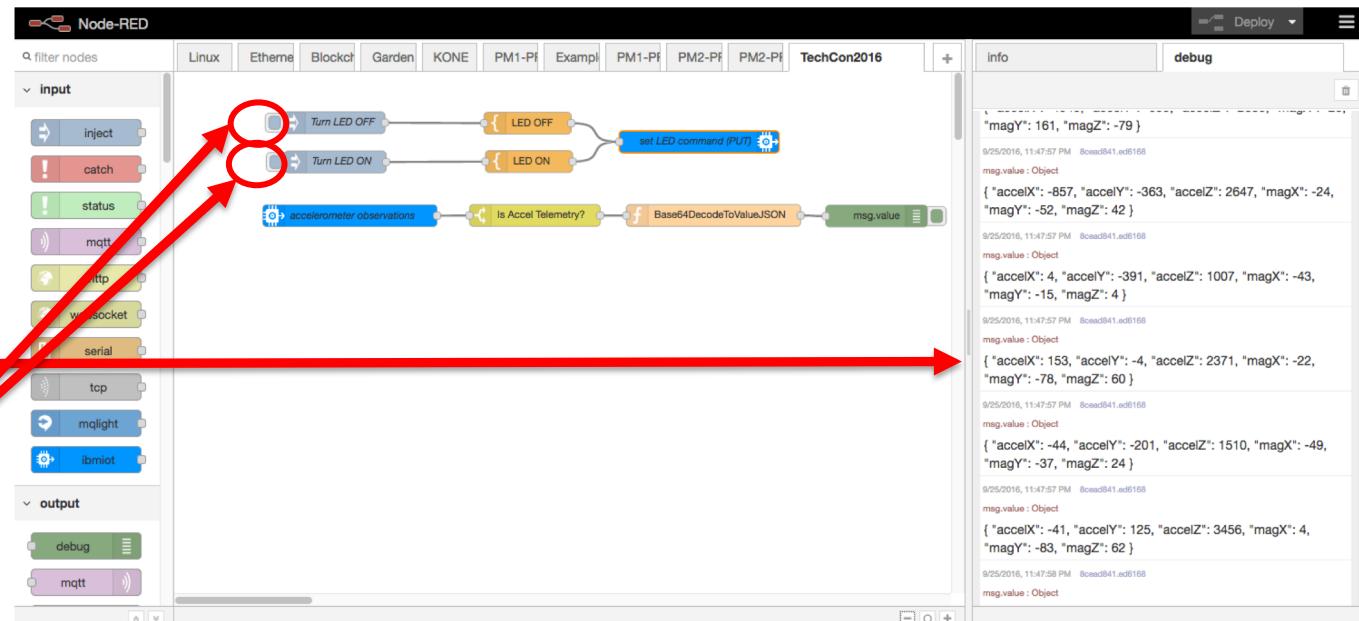
Putting it all Together: NodeRED debug info...

- Navigate to your Watson IoT application and NodeRED flow editor

- Examine the debug window
 - Shake your K64F!

- You should see output similar to this... telemetry in Watson IoT

- You can toggle your LED On and off (left-most block) by clicking each of these... look in the serial terminal for output from the endpoint



Putting it all Together: Check Watson IoT Devices

- Go to the last window you had open from slide 19
- Select "Devices"

IBM Watson IoT Platform

QUICKSTART SERVICE STATUS DOCUMENTATION BLOG

deneon@watson.com • ID: 8d22ed

+ Add Extension

Extensions

Single Sign On The Single Sign On (SSO) extension allows additional authentication options to be enabled. Status: Not Configured

Email The email extension configures options for the SendGrid and Amazon SES user invitation methods. Status: Not Configured

ARM mbed Connector This integration enables ARM mbed Connector devices to integrate with IBM's Watson IoT Platform and exchange messages bi-directionally. Status: Configured

Historical Data Storage The historical data storage extension holds and configures complex services that can be used to store your IoT device data. You must be logged in to Bluemix in order to complete this operation. Status: Not Configured

- The bridge automatically creates Watson IoT devices for you

Devices

Browse | Diagnose | Action | Device Types | Manage Schemas Refresh + Add Device

Device ID	Device Type	Class ID	Date Added	Location	Action
cc69e7c5-c24f-43cf-8365-8d23bb01c707	mbed-endpoint	Device	Oct 31, 2016 3:58:23 PM		

Putting it all Together: Check Watson IoT Devices

- Select your device

The screenshot shows the Watson IoT Platform's Devices dashboard. A red arrow points from the 'Select your device' bullet point to the device list. The list displays one result: 'cc69e7c5-c24f-43cf-8365-8d23bb01c707'. This row includes columns for Device ID, Device Type (mbed-endpoint), Class ID, Date Added (Oct 31, 2016 3:58:23 PM), and Location.

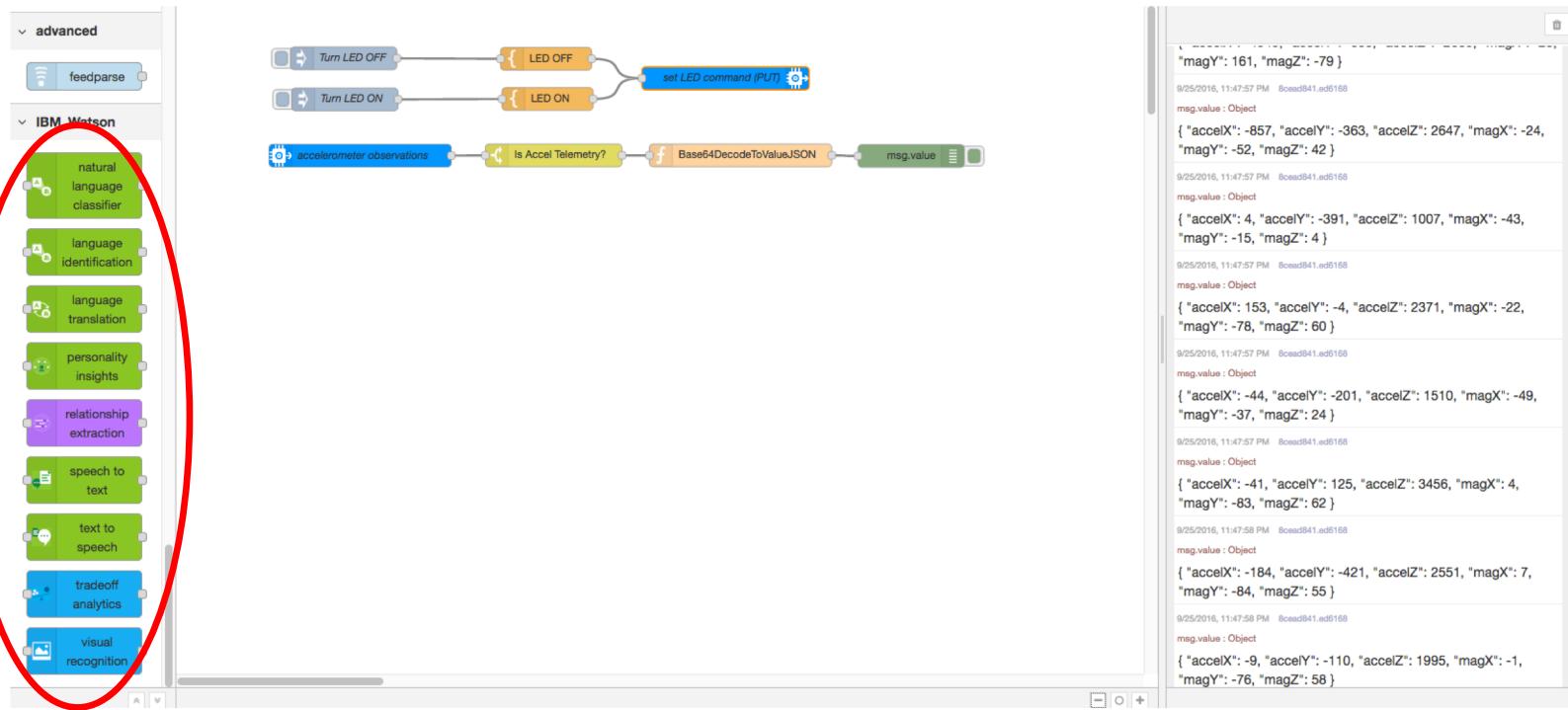
- You can watch CoAP
notify events occurring
via your bridge
(observations)

The screenshot shows the 'Device cc69e7c5-c24f-43cf-8365-8d23bb01c707' details page. A red arrow points from the 'Observations' bullet point to the 'Recent Events' section. This section lists three notifications received on Oct 12, 2016, at 1:19:51 PM, 1:19:52 PM, and 1:19:53 PM, all in JSON format.

Time Received	Format	Message
Oct 12, 2016 1:19:51 PM	json	notify
Oct 12, 2016 1:19:52 PM	json	notify
Oct 12, 2016 1:19:53 PM	json	notify

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 Cloud 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.