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

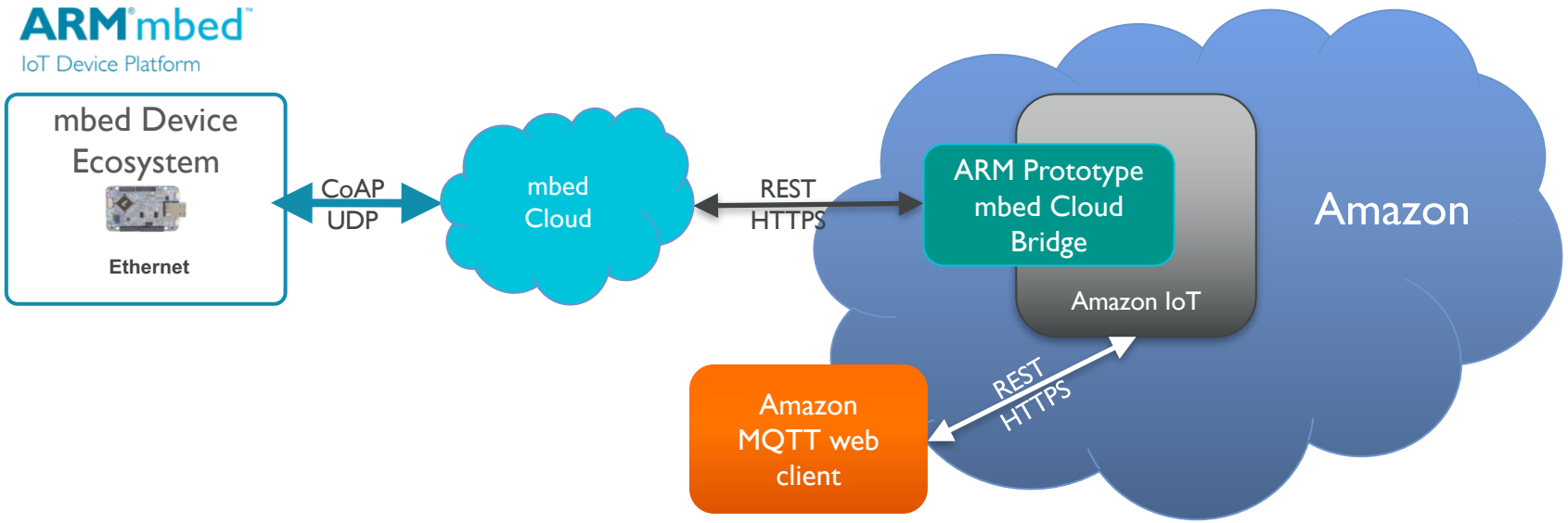
ARM

Doug Anson
Solutions Architect / IoT BU / ARM

Brian Daniels
Applications Engineer / IoT BU / ARM

Sept 25, 2017 – v2.2

What will we build in this Workshop?



- Connect your mbed device into Amazon IoT through mbed Cloud and ARMs Prototype AWS IoT Cloud Bridge
  - Create a simple mbed device and connect it to mbed Cloud
  - Create an instance of ARM's Prototype AWS IoT Cloud bridge and bind it to your Amazon and mbed Cloud accounts
  - Exploration of the device data telemetry using Amazon's web-based MQTT client

Workshop: Let's get started!

- Create and setup your Amazon account (should be completed prior to workshop)
- Install the necessary tools/drivers PC/Mac/Linux (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/anson_workshop_aws_iot_2016

- README.md has all of the links in the workshop + this presentation in PDF format

Create our Amazon Account

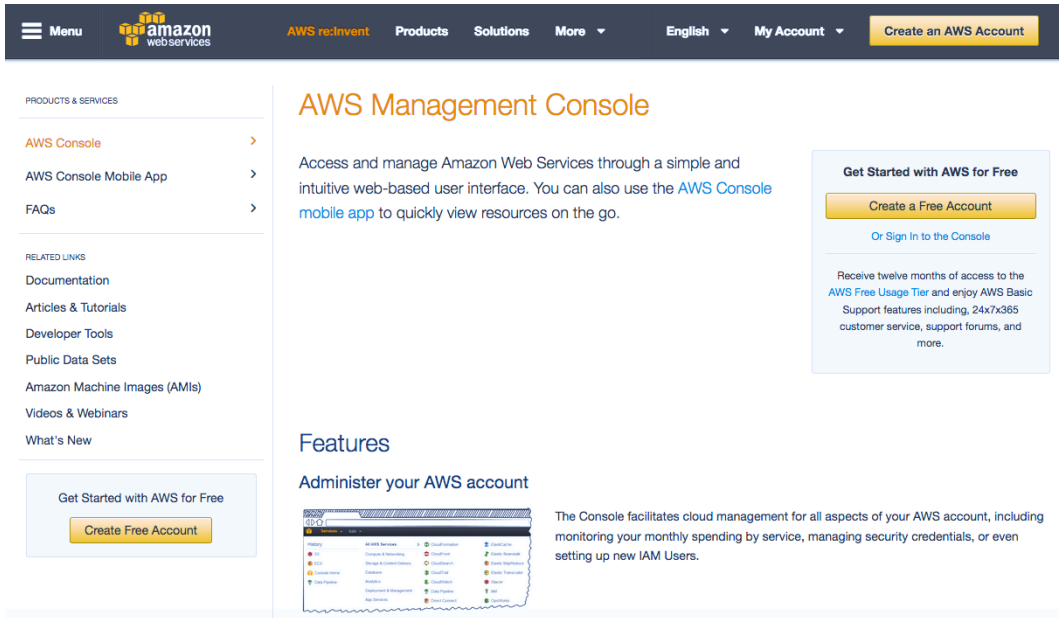
- Navigate to the Amazon Management Console: https://aws.amazon.com/console/

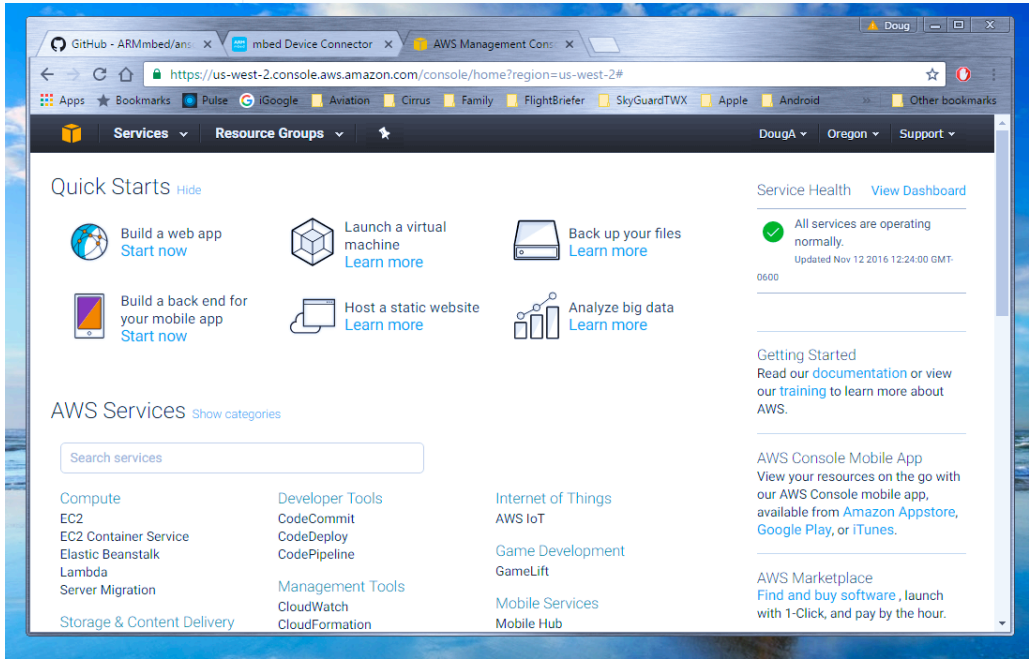
- Press "Create a Free Account"
  - Enter your email address that you can access
  - Select "I am a new user"
  - Press "Sign in using our secure server"
  - You will need to provide a Credit Card
  - You will be called on your cell phone to verify identity

- Complete the Account signup and validation
  - You can choose the "Basic" support

- Press "Sign into the Console"

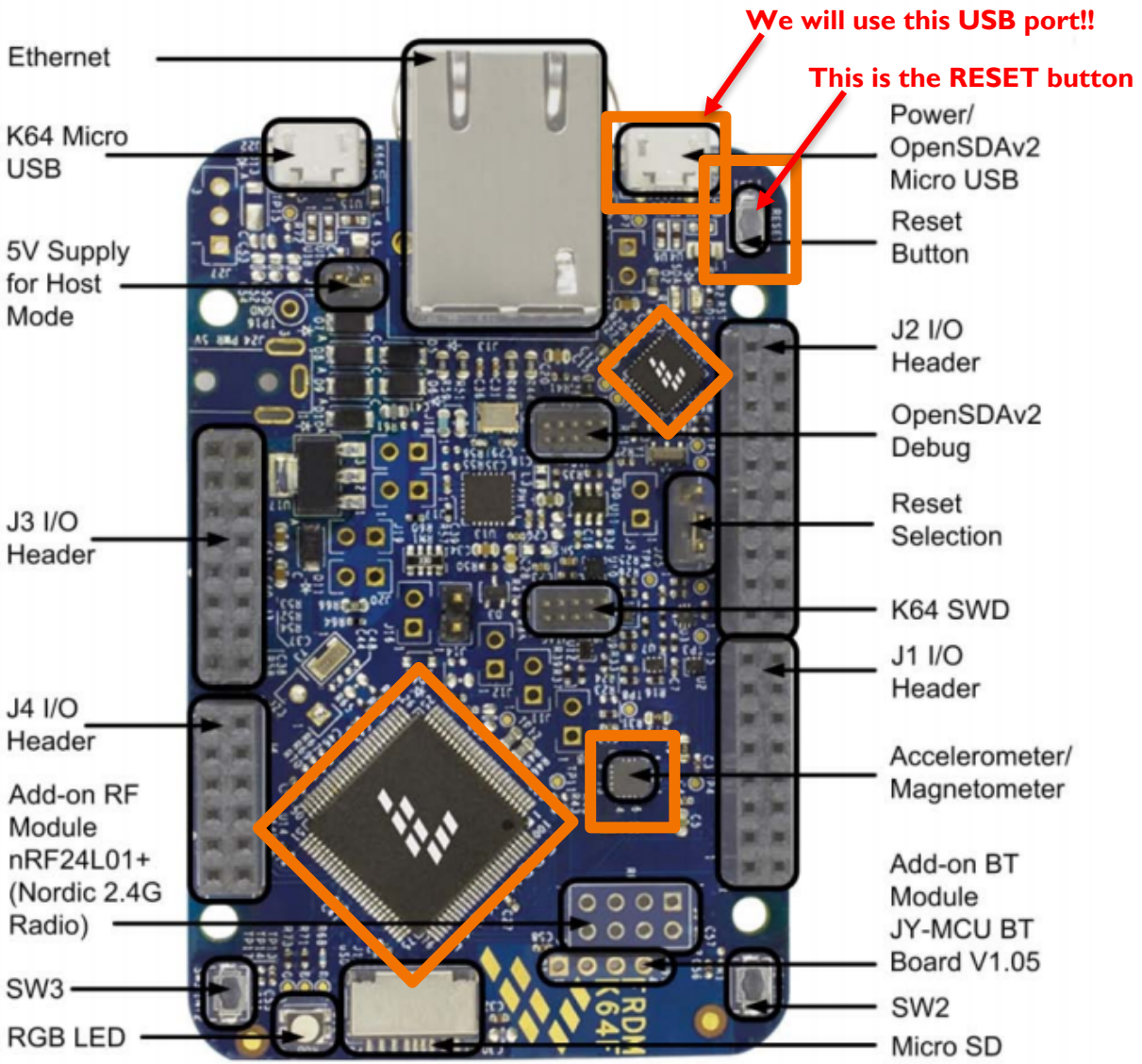
Prior to Workshop





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

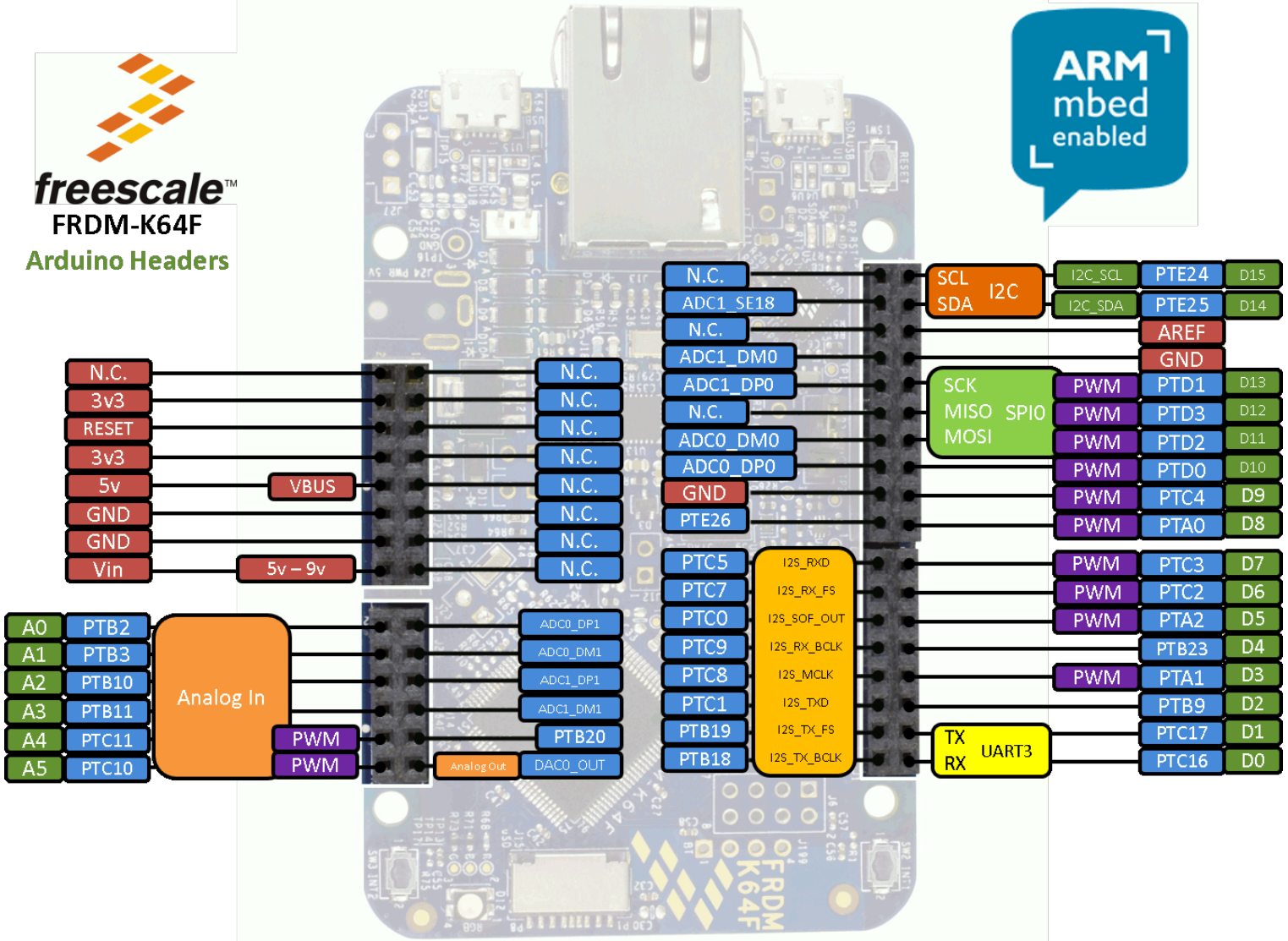


ARM

freescale™

FRDM-K64F

Arduino Headers



ARM

Install the Necessary Tools

- **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*
  - *You will need administrator access to your Windows PC to install this driver*

Prior to Workshop

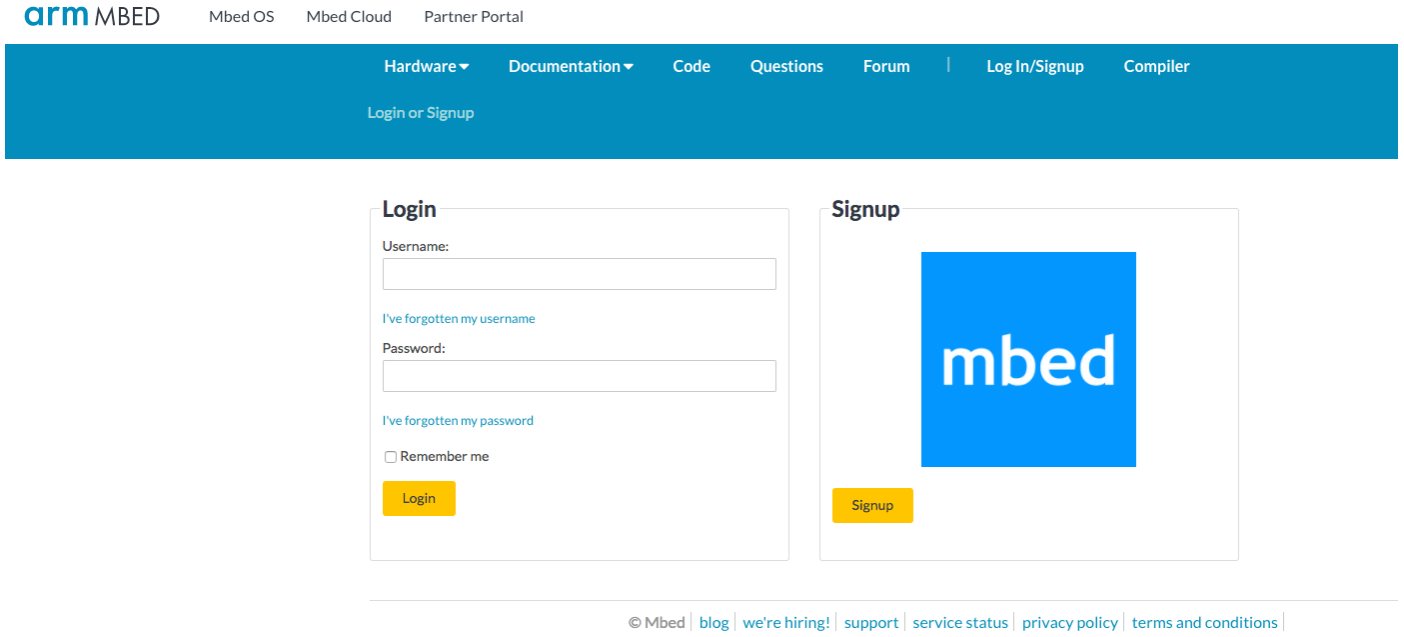
- Serial Terminal: Putty - http://www.putty.org/
- Chrome and Firefox Browsers may also be used
- Docker *Toolbox* installed on your Windows PC: https://github.com/docker/toolbox/releases/tag/v1.12.2
- Docker Engine installed on your Mac - https://docs.docker.com/engine/installation/mac/ (*not* Toolbox)
- Docker Engine installed on your Linux - https://docs.docker.com/engine/installation/linux/
- Mac/Linux: install "git" (mac: https://git-scm.com/download/mac, Ubuntu: use "apt-get install git"
- Mac Serial Terminal - http://freeware.the-meiers.org/CoolTerm_Mac.zip
- Linux Serial Terminal - http://freeware.the-meiers.org/CoolTerm_Linux.zip

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

Create Your mbed Account

Prior to Workshop

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

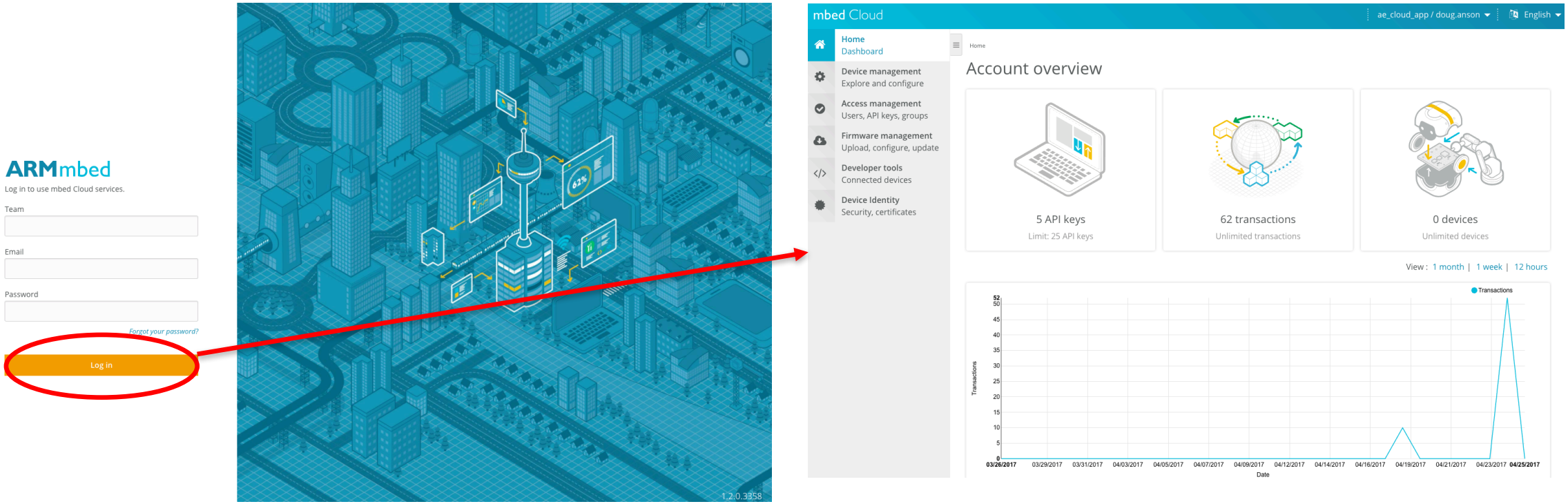


ARM

Create Your mbed Cloud Account… Prior to Workshop

- Navigate to the mbed Cloud Dashboard: https://portal.mbedcloud.com

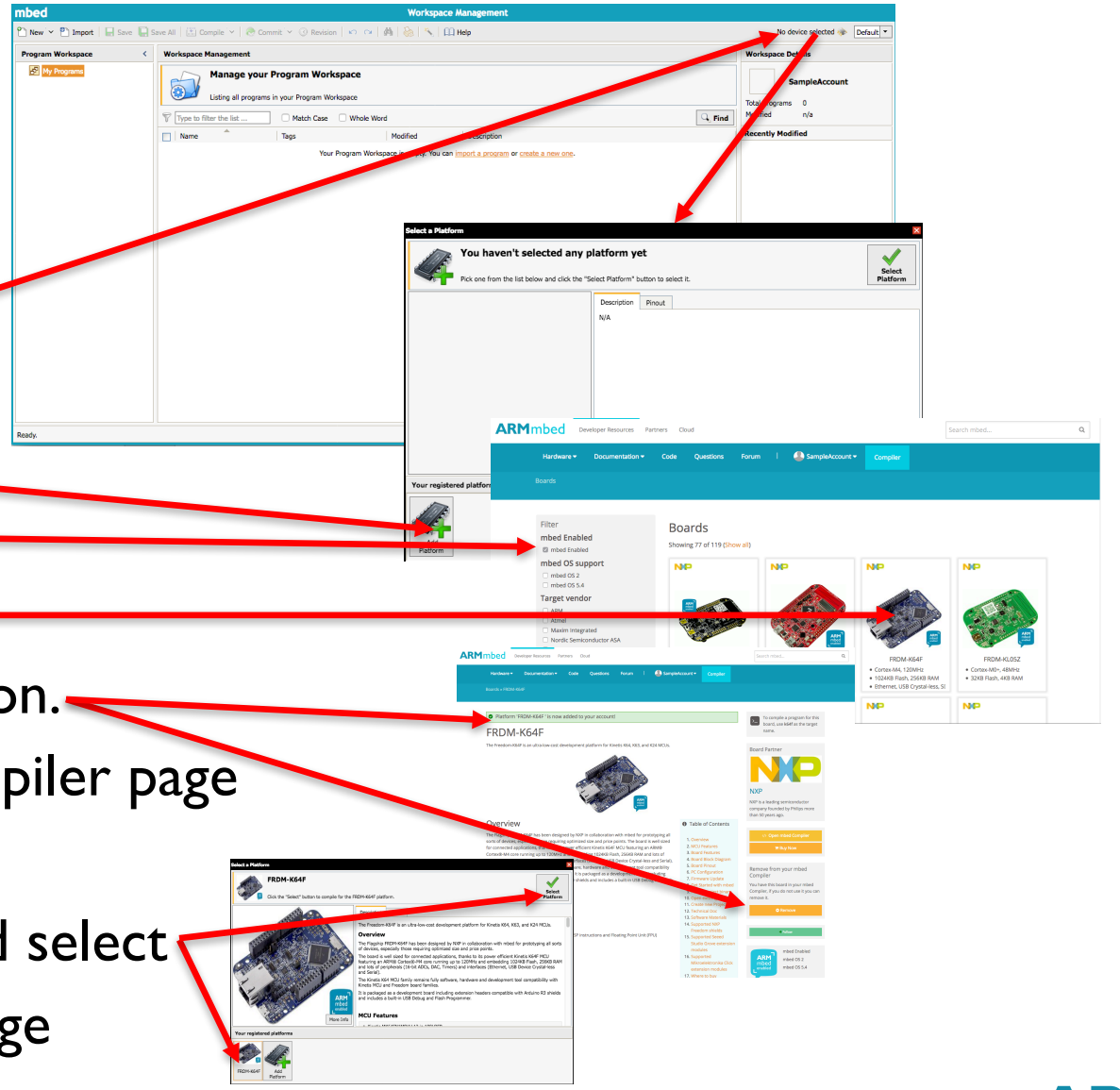
- Confirm that you can log into your mbed device mbed Cloud dashboard



Log into the Online IDE – Add the K64F Compile Target

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

- Select the "Compiler" page
- Click on "No device selected"
- Click on "Add Device"
- Check "mbed Enabled"
- Click on the "FRDM-K64F"
- Click on "Add to Compiler"... note addition.
- Dismiss all windows... go back to the compiler page
- Click on "No device selected"
- Now, select the "FRDM-K64F" to add and select
- Dismiss and confirm on your compiler page

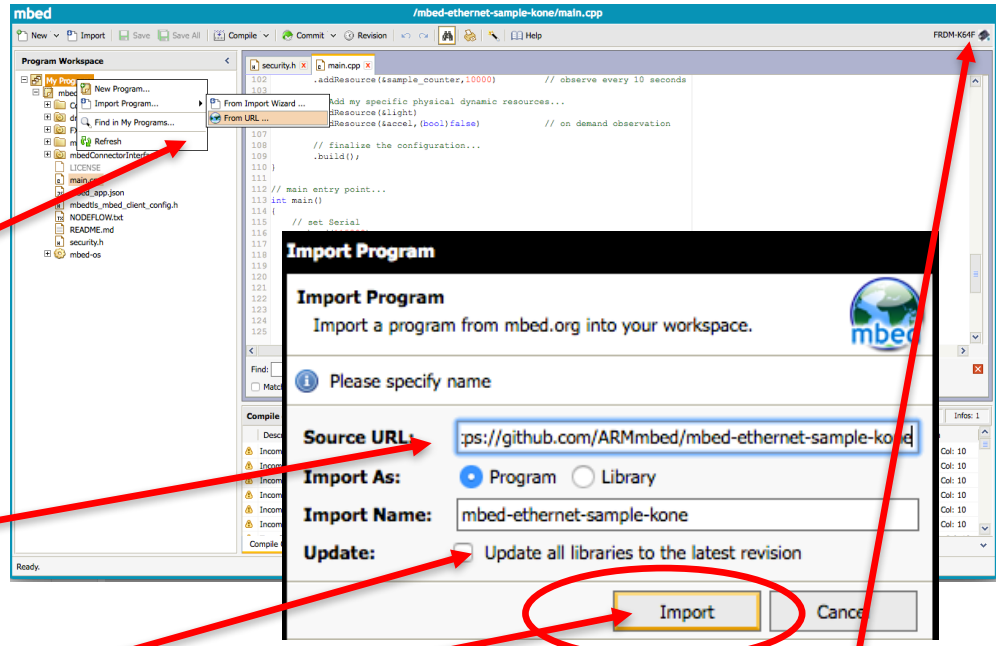


ARM

Import our K64F Endpoint Project

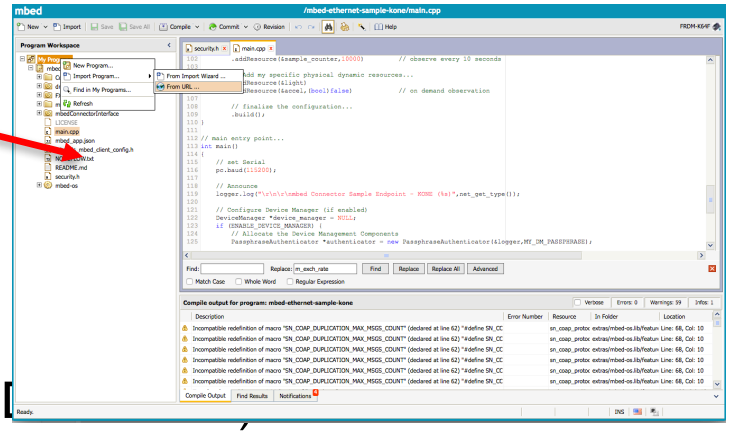
- Go to https://os.mbed.com/
- Select the "Compiler" page
- Right-click on "My Programs"→"Import Program"
- Choose Select "from URL…"
- Enter this URL (Leave the "Update all libraries…" **unchecked**)

  https://github.com/ARMmbed/mbed-cloud-sample/
- Press "Import", the ensure that "FRDM-K64F" is selected



Set Provisioning Credentials in Your Endpoint Code

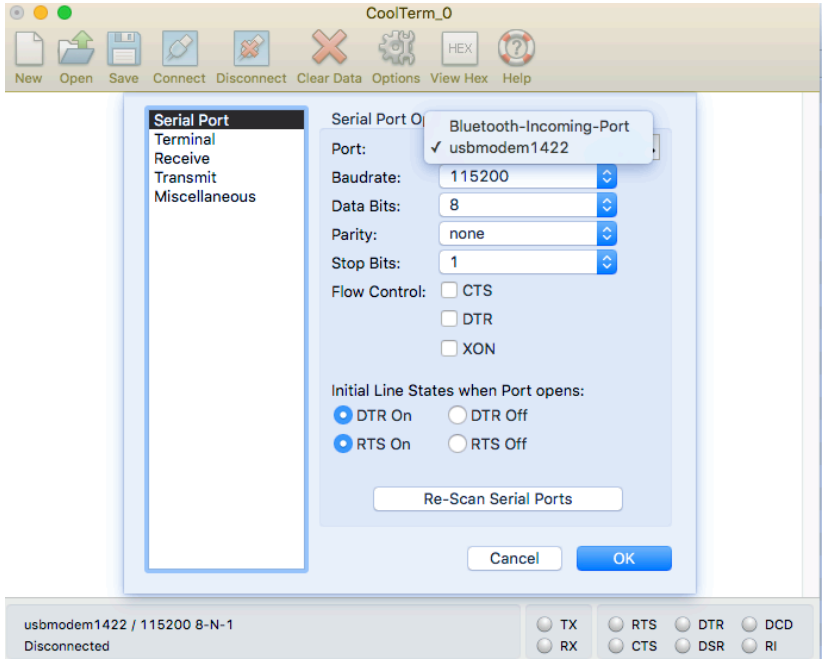
- Go back to the mbed Device mbed Cloud dashboard: https://portal.mbedcloud.com
  - In the left sidebar, select "Device Identity"→ "Certificates"
  - Under "Actions", select "create a developer certificate"
  - Give your developer certificate a name and description… press "create certificate"
  - Press "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 "
- K64F green LED will flicker for a bit, then stop, and dismount/remount…

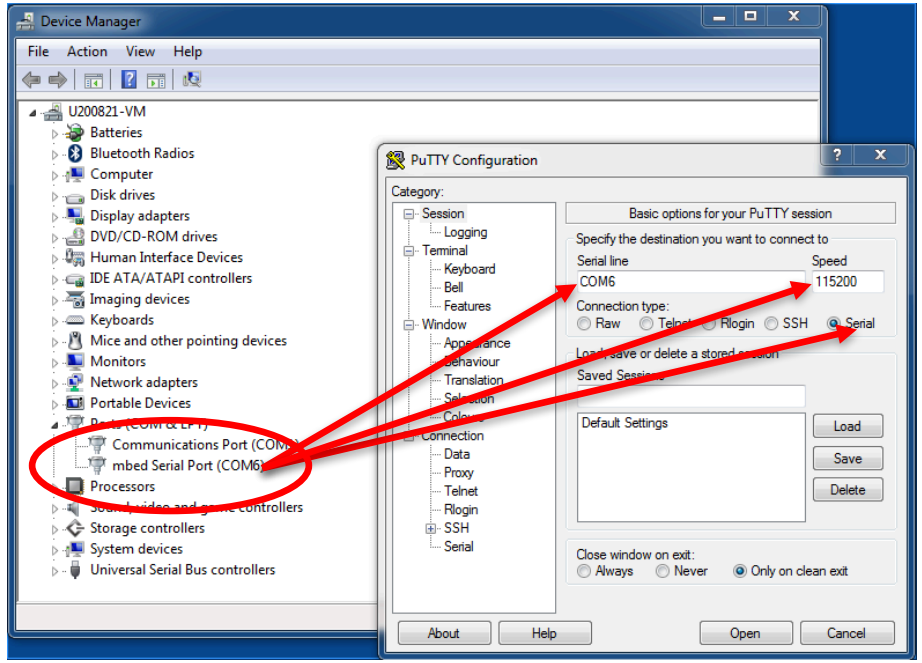


Running your Endpoint Code

- 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…



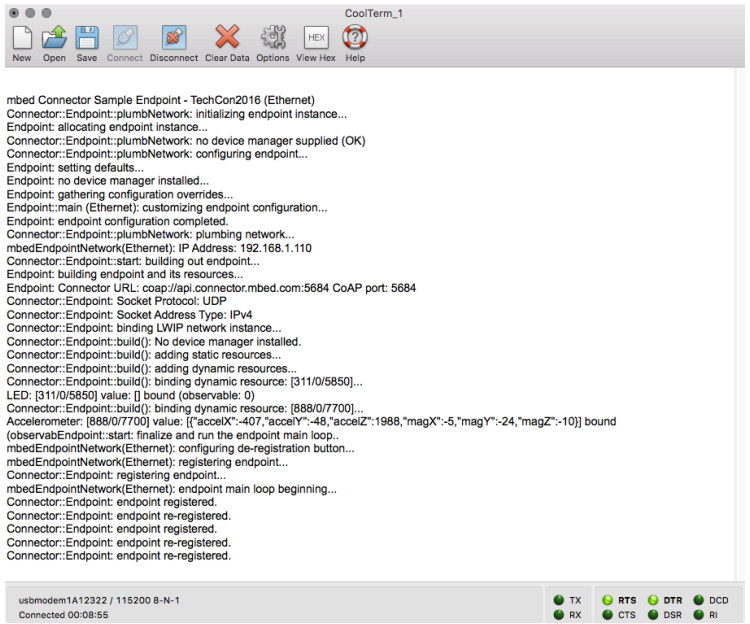


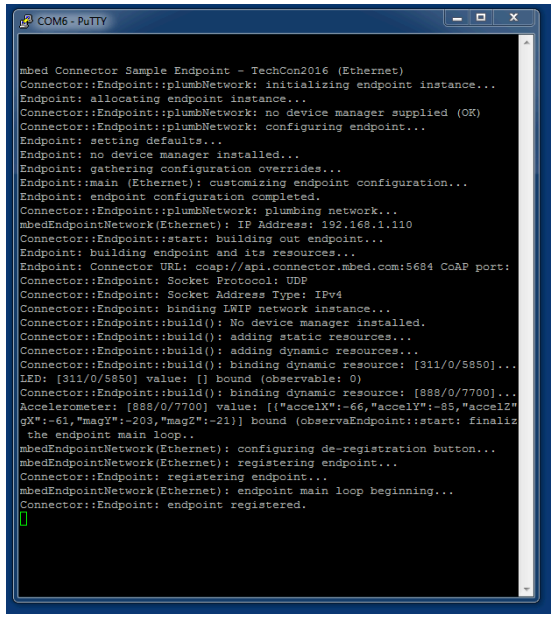
Running your Endpoint Code…

- 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:





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 (115200,8N1, 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 ARM AWS IoT Prototype mbed Cloud Bridge...

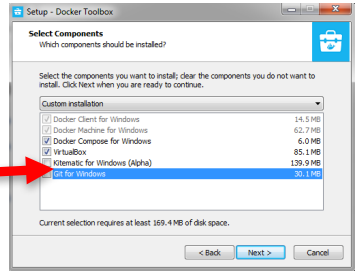
ARM AWS IoT mbed Cloud Bridge Setup

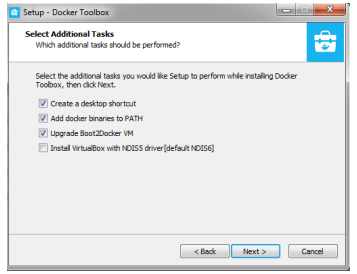
What we will do next…

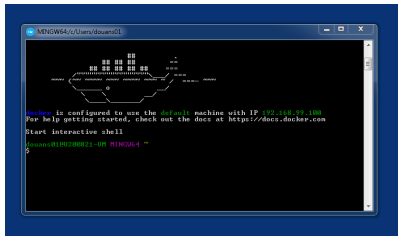
- Ensure that we have the necessary pre-requisites setup on our PC
- Create our mbed Cloud API Key/Token
- Create our AWS IoT Instance
- Create our AWS Access Key and ID
- Import our Prototype mbed Cloud Bridge instance as a Docker image into our Docker runtime
- Configure our Prototype mbed Cloud Bridge for AWS IoT

Double check (Windows) : All our needed tools are installed

- Latest Docker Toolbox runtime installed
  - Ensure that "Git for Windows" is checked!
  - https://github.com/docker/toolbox/releases/tag/v1.12.2







- Windows mbed USB driver installed and functioning properly.
  - "DAPLINK" flash drive present
  - "mbed Serial Port" seen in Windows Device Manager when K64F USB is connected
- Putty installed and operational
- Chrome and/or Firefox installed

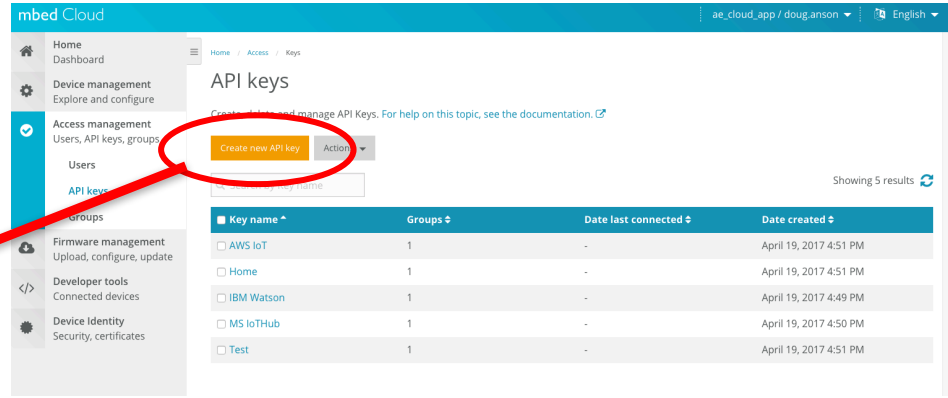
ARM

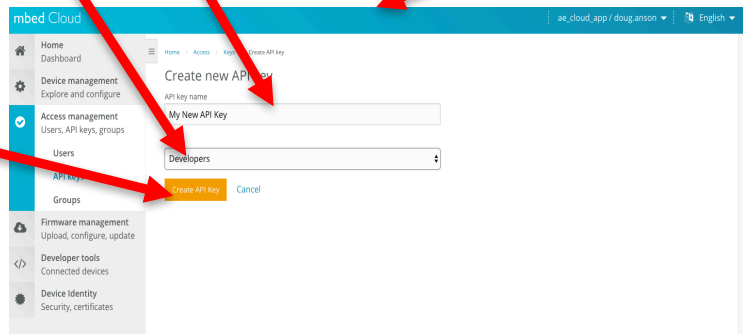
Double check (Mac/Linux) : All our needed tools are installed

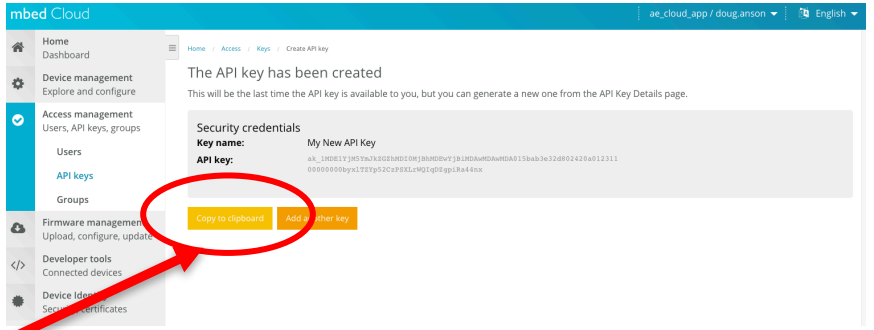
- Docker Engine installed (do not install the "Toolbox" version on Mac... use the native engine)
  - Docker Engine for Mac (do not install the "Toolbox" version on the mac):
    - https://docs.docker.com/engine/installation/mac/
  - Docker Engine for Linux:
    - https://docs.docker.com/engine/installation/linux/
- Suitable serial terminal installed and operational
- Chrome and/or Firefox installed
- "git" installed
- Access to a command line terminal

Create our mbed Cloud Access/API Token

- Navigate to the mbed Cloud Dashboard: https://portal.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
- Press "copy to clipboard" – save to a file for later use

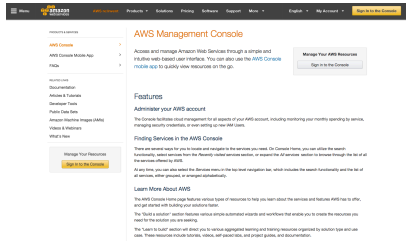


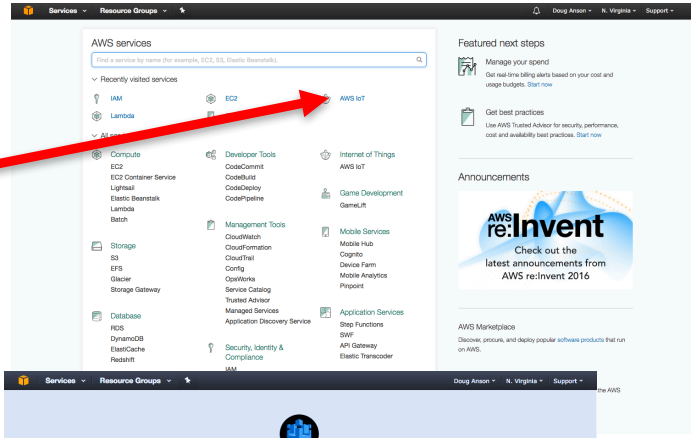


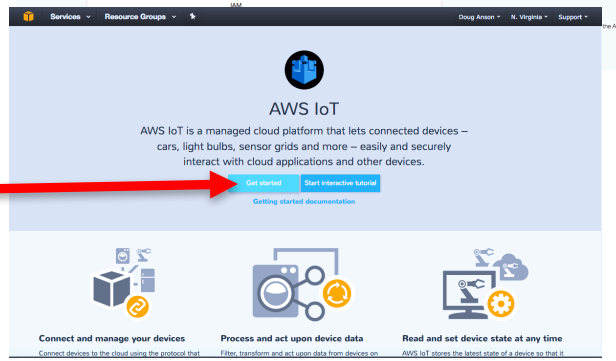


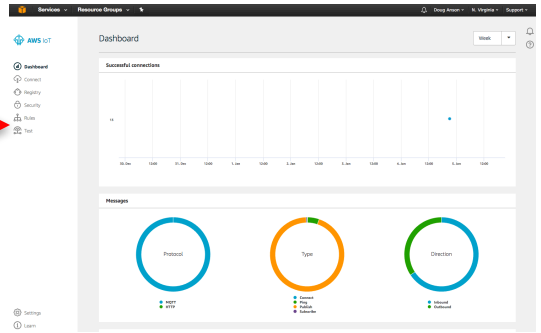
Create our Amazon AWS IoT Instance

- log into your AWS Management Console
  - https://aws.amazon.com/console/
  - Sign In...
- Select "AWS IoT"
- Create an instance if you don't have one
  - Complete steps to create your instance
- Once created, go to your AWS IoT
  - Dashboard... Select "Test"



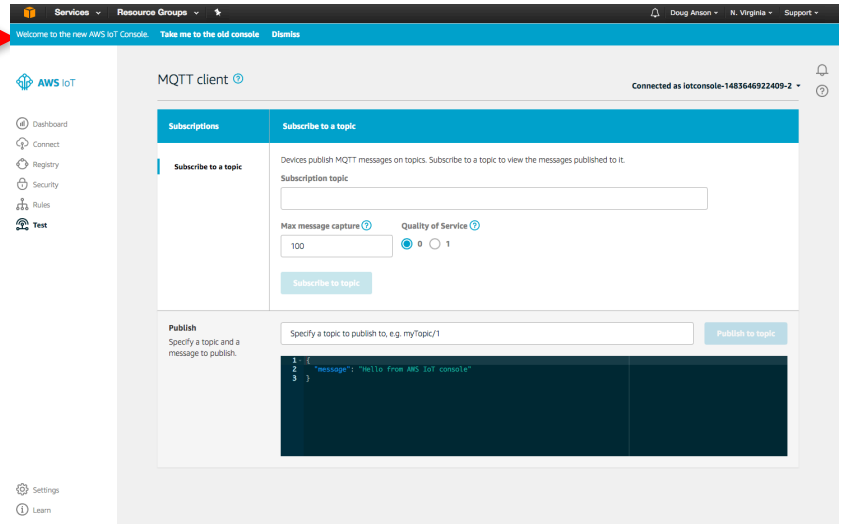






Determine our AWS Region

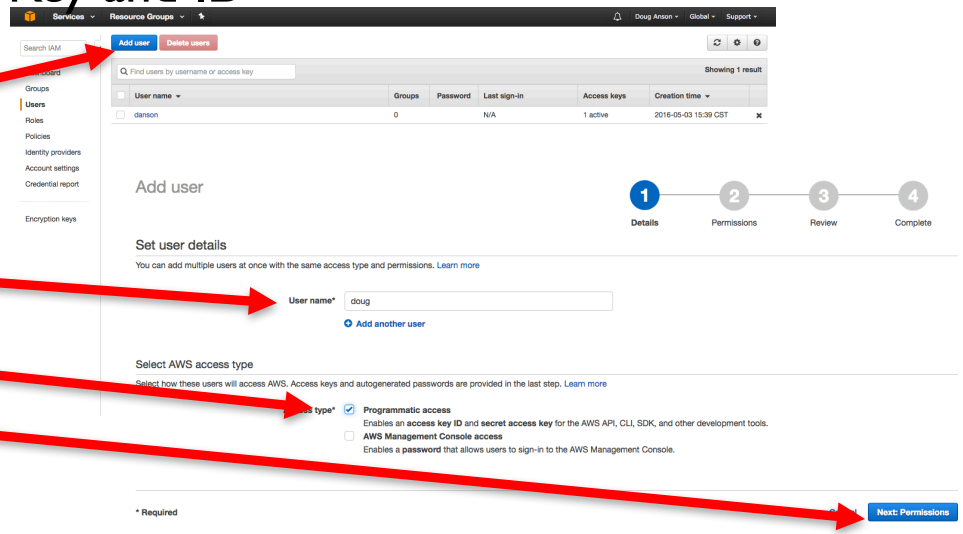
- Continuing with the "Test" AWS IoT Console

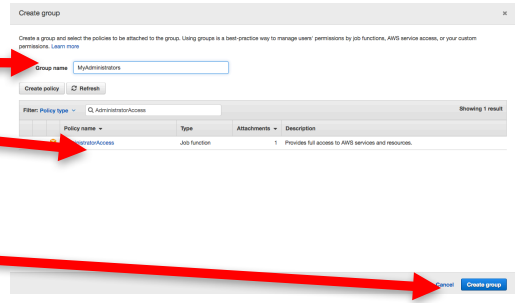


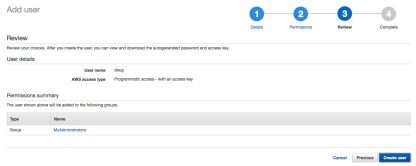
- Look at the URL for your AWS IoT Test Console
- You should see something like this:
  - https://console.aws.amazon.com/iotv2/home?region=us-east-1#/test
- Make a note of your "region" parameter value
- In the above example, my AWS Region is "us-east-1". Yours may be different.
- Note what yours is.. we will use that value later

Create our AWS Access Key and ID (New IAM User)

- Using Chrome, navigate to:
  - http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/AWSCredentials.html
  - Complete the steps in Amazon's HOWTO to create the Access Key and ID
- If you do not have an IAM User... create one
  - Create a new user
  - Select "Programmatic Access"
  - Press "Next Permissions"
- Create a Permissions Group
  - Name the Group
  - Search for and select "AdministratorAccess" permission...
  - Create the Group
  - Complete creating the user!

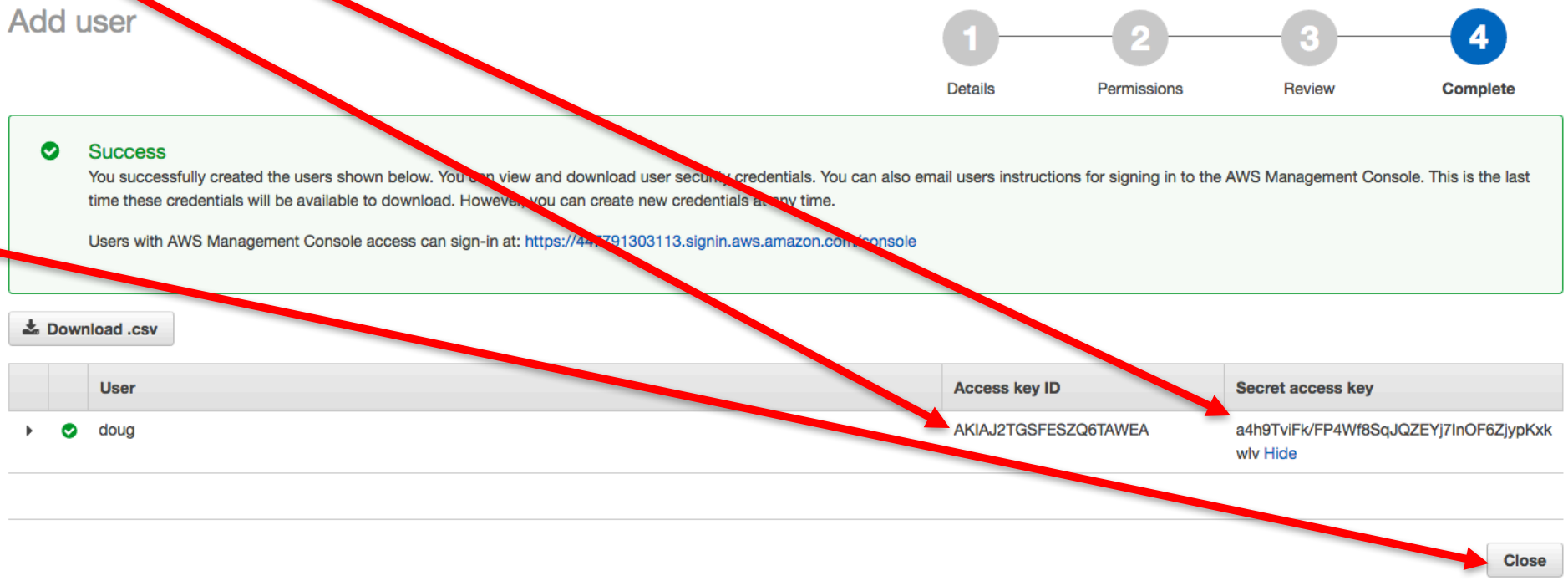






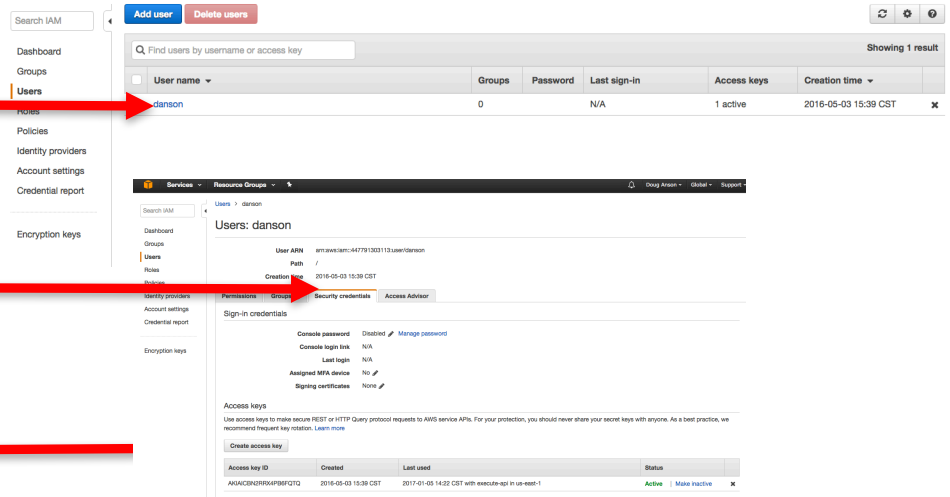
Create our AWS Access Key and ID(New IAM User)...

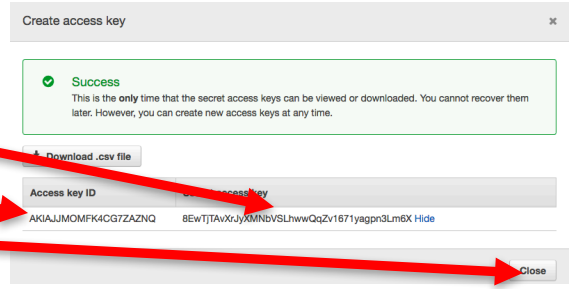
- Finally, after the new IAM User is created, you will be shown your keys
  - "Show" your Secret Access Key – save off both for later
  - AWS IoT Secret Access Key
  - AWS IoT Access Key ID
  - Press Close



Create our AWS Access Key and ID (Existing IAM user)...

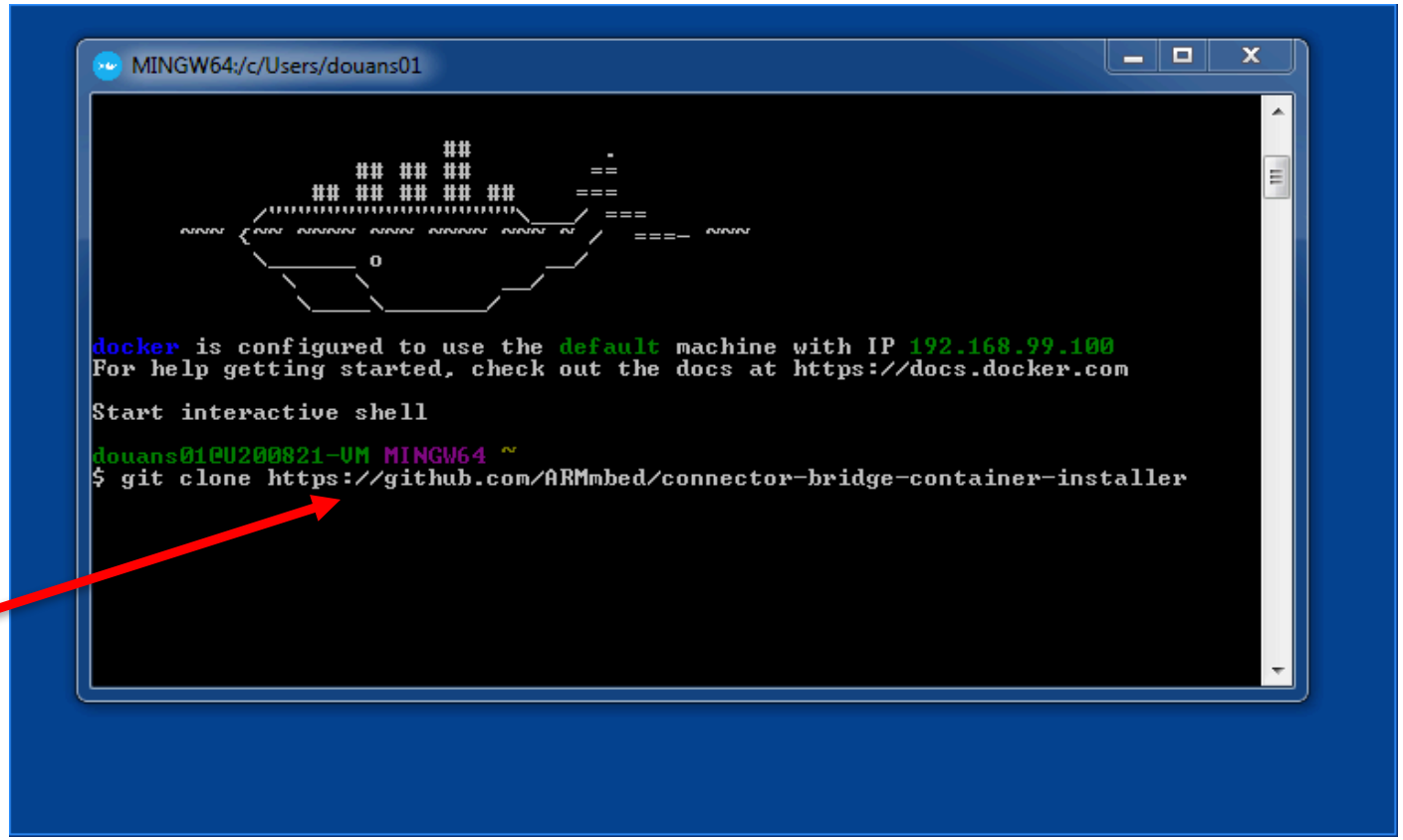
- If you have an existing IAM User, then you can simply create a new AccessKey for that user
- Select your IAM User
- Select "Security Credentials"
- Press "Create Access Key"
- Show & Save off the values you acquired ("Show" the secret access key)
  - AWS IoT Secret Access Key
  - AWS IoT Access Key ID
  - Press Close





Import our mbed Cloud AWS IoT Bridge Installer

- Windows: Launch the Docker QuickStart Terminal
  - It may take a few minutes to initialize if launched the first time... it will have to download additional stuff
  - Allow VirtualBox to make changes to your network interface
- Mac/Linux: Open a terminal
- Type the following "git" command (below)

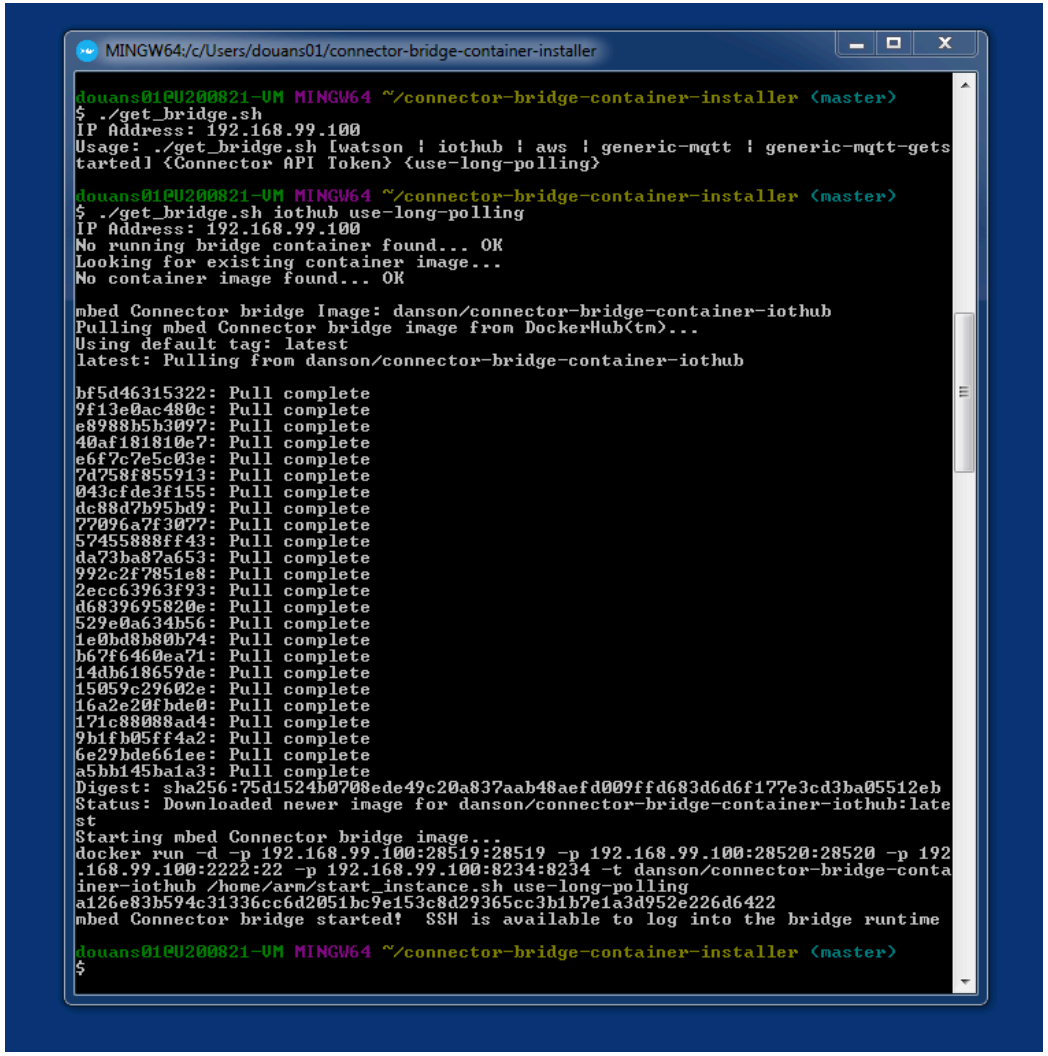


- For all platforms (Windows/Mac/Linux), this is the "git" command to run:
  $ git clone https://github.com/ARMmbed/connector-bridge-container-installer

ARM

Import our mbed Cloud AWS IoT Bridge Container

- cd into the installer repo
  % cd connector-bridge-container-installer
- Run the installer script (look at options)
  % ./get_bridge.sh
- Run the installer script with options
  % ./get_bridge.sh aws use-long-polling
- Bridge container will import from DockerHub



ARM

Configure our AWS IoT Bridge (Windows)

- Your Container IP address will be:

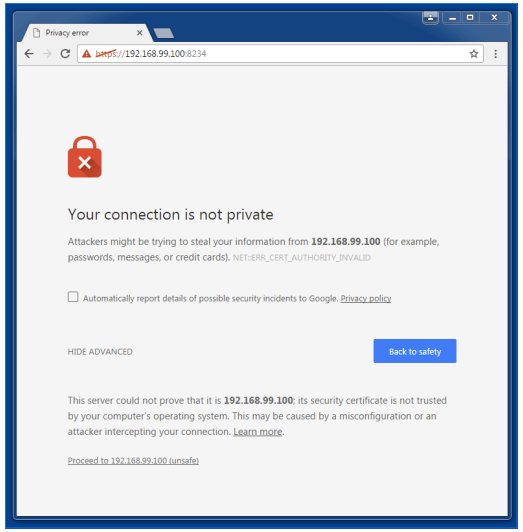
  192.168.99.100

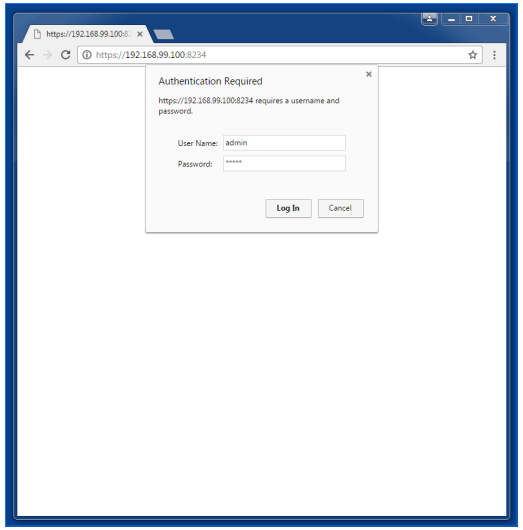
- Bring up Firefox/Chrome and go to:

  https://192.168.99.100:8234

- Accept the self signed certificate

- Username: admin PW: admin





Configure our AWS IoT Bridge (Mac/Linux)

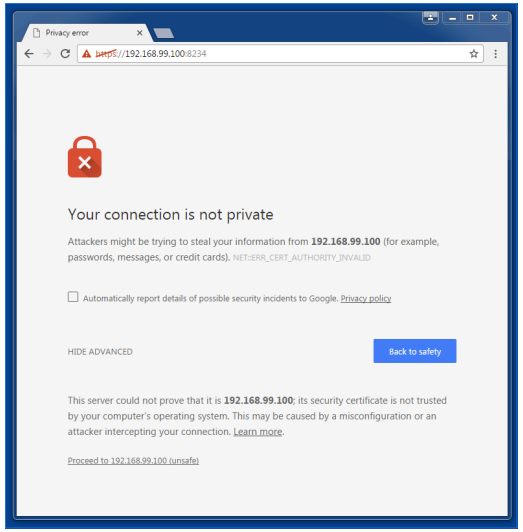
- Your Container IP address be ”localhost” if installed on your local box, otherwise, make note of it if installed remotely

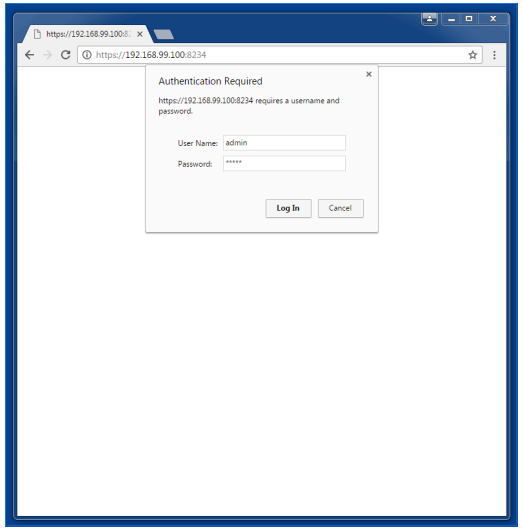
- Bring up Firefox/Chrome and go to:

  https://<your container IP address>:8234
  (typically: https://localhost:8234)

- Accept the self signed certificate

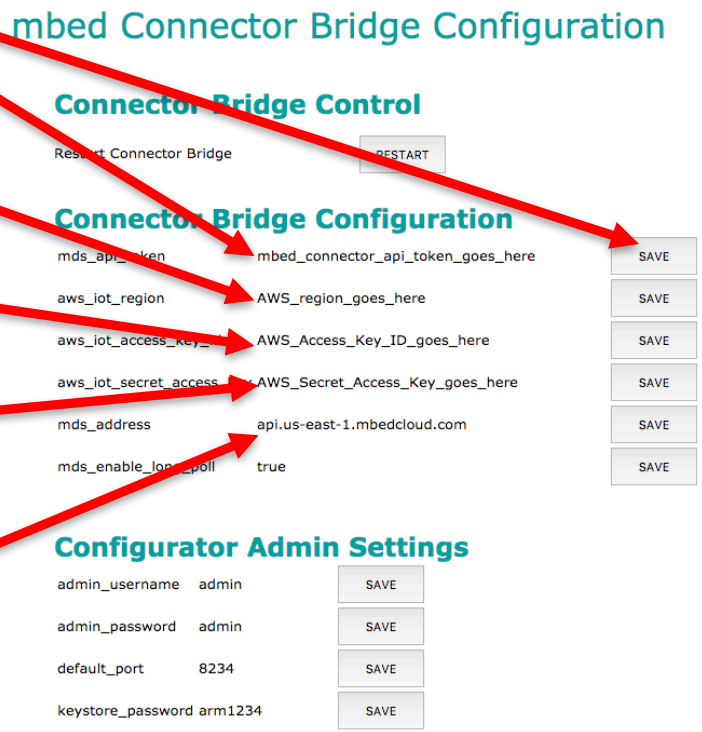
- Username: admin PW: admin





Configure the mbed Cloud Bridge for AWS IoT…

- Enter the mbed Cloud API Key/Token, *THEN* Press "Save"
- Enter your AWS Region, Press "Save"
- Enter your AWS Access Key ID, Press "Save"
- Enter your AWS Secret Key, Press "Save"
- Ensure you have the right API endpoint
  - Cloud: api.us-east-1.mbedcloud.com
  - Connector: api.connector.mbed.com
- Press "RESTART"
- Your mbed Cloud Bridge is now configured for AWSIoT



Update for GA

ARM

Status Check

So far we've completed the following

- Ensure that we have the necessary pre-requisites setup on our PC
- Create our mbed Cloud API Key/Token
- Create our AWS IoT Instance
- Create our AWS Access Key and ID
- Import our bridge instance: a Docker image into our Docker runtime
- Configure our mbed Cloud Bridge for AWS IoT

With our bridge now operational, we should be able to restart our endpoint and see messages coming into our AWS IoT via Amazon's MQTT Client/explorer

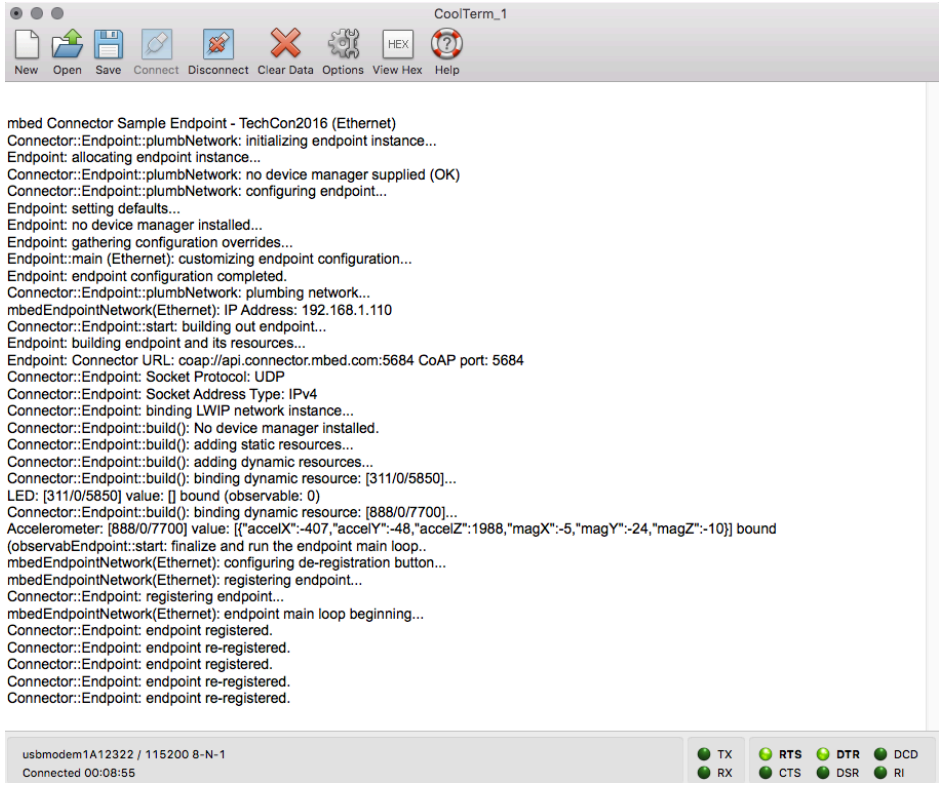
Putting it all Together

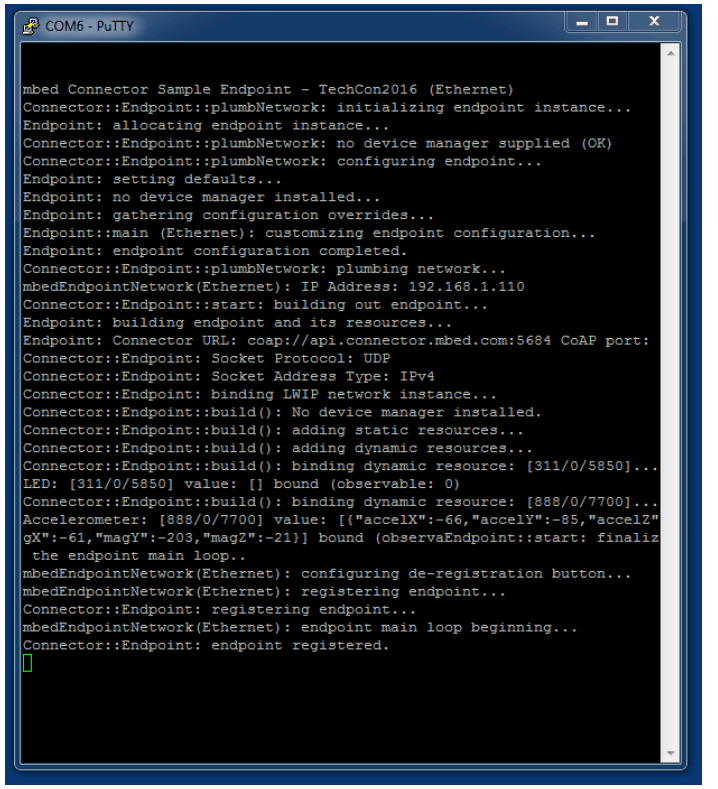
- Press the "reset" button on your mbed device (next to the USB port)
- Ensure that you see "endpoint registered" in the serial terminal
- Go to your AWS IoT Dashboard
  - URL should look someting like this:
    - https://console.aws.amazon.com/iotv2/home?region=us-east-1#/dashboard
    - Your AWS Region ("us-east-1") may be different though...

Lets check how things are working…

Putting it all Together: Check the Serial Terminal

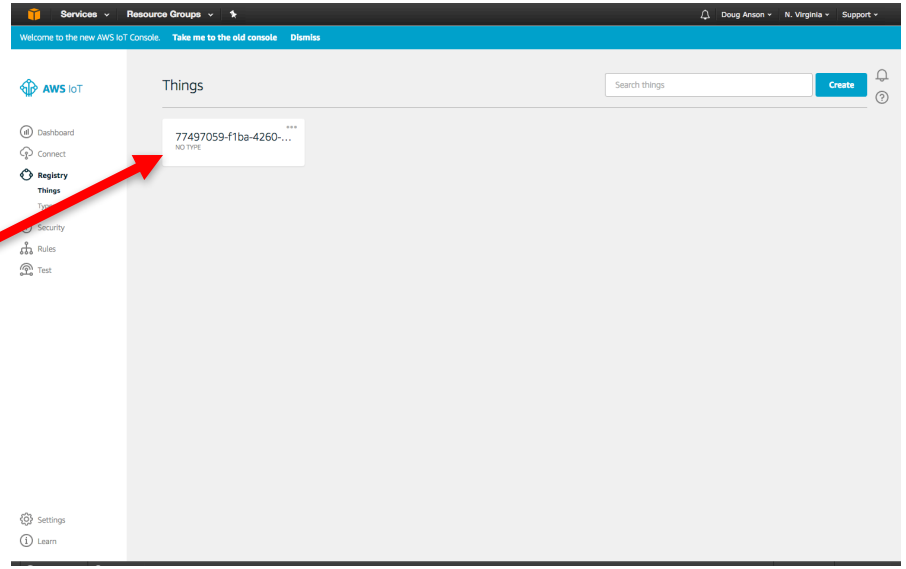
- You should see output that looks something like this:
  - Make sure that you see a message like "endpoint registered"

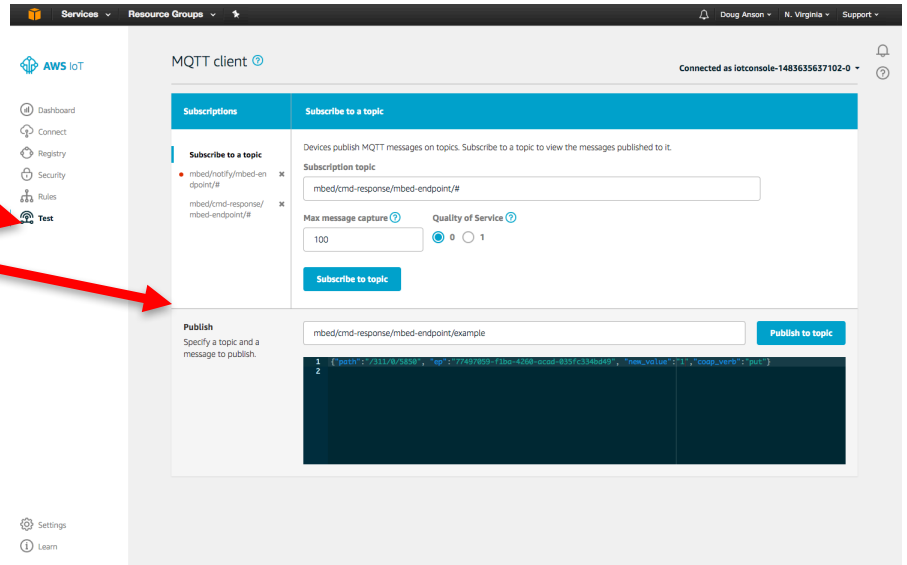




View your device telemetry in AWS IoT MQTT Client

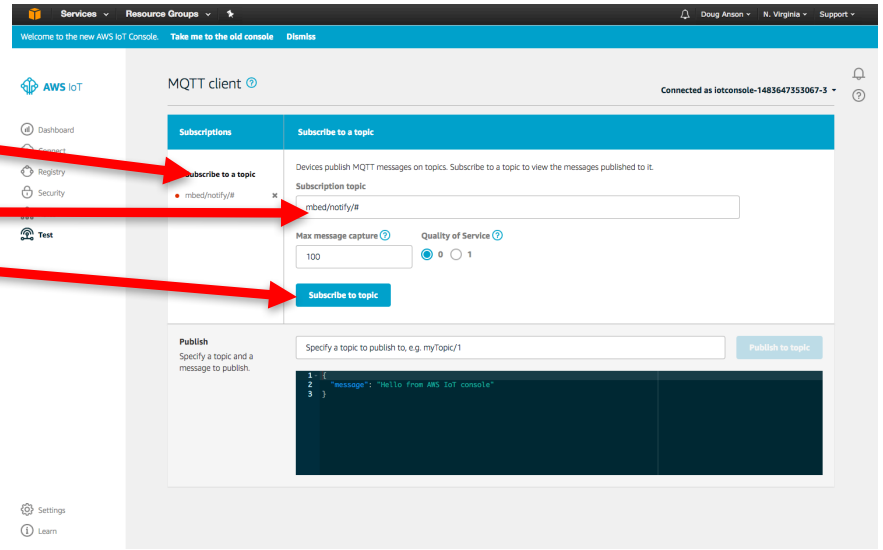
- Look at your AWS IoT Dashboard
- Under the “Registry/Things" tab, you should see your device listed as a "Thing"
- Next lets bring up the "Test" dashboard to interact with the device

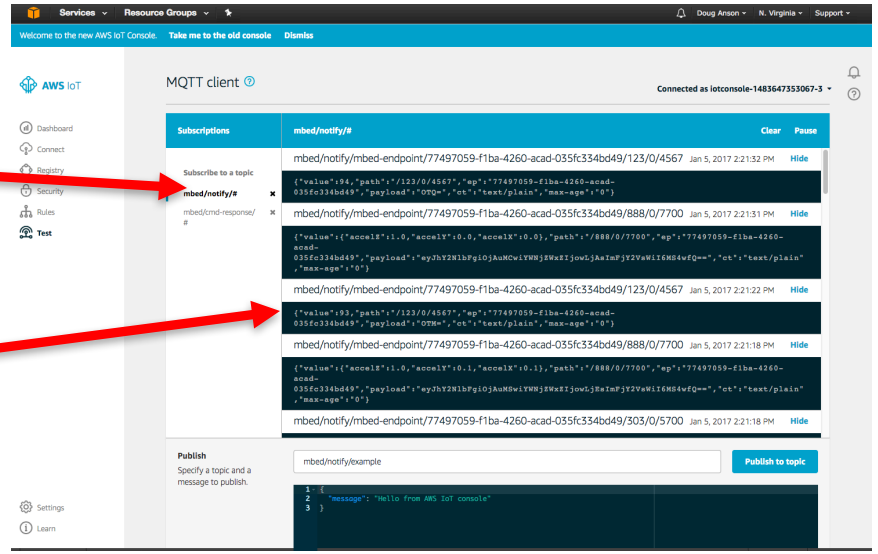




View your device telemetry in AWS IoT MQTT Client...

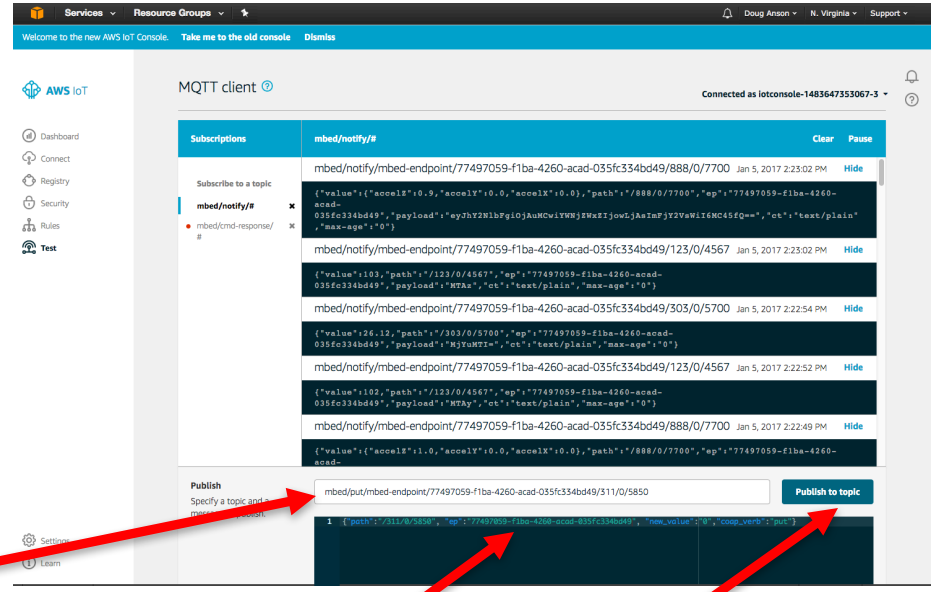
- Now, "subscribe" to the observation channel:
  - Client ID: "mbed/notify/#"
- Also, subscribe to the command response channel:
  - Client ID: "mbed/cmd-response/#"
- Click on one of these subscriptions...
- You should start seeing output from your devices' observations





Interacting with your device in AWS MQTT Client

- Next, we will send a CoAP "put" command to the device to toggle the LED on ("1") or off ("0")



- Enter the topic:
  mbed/put/<endpoint_type>/<endpoint_name>/311/0/5850
  (<endpoint_type> and <endpoint_name> can be see in the observations...)

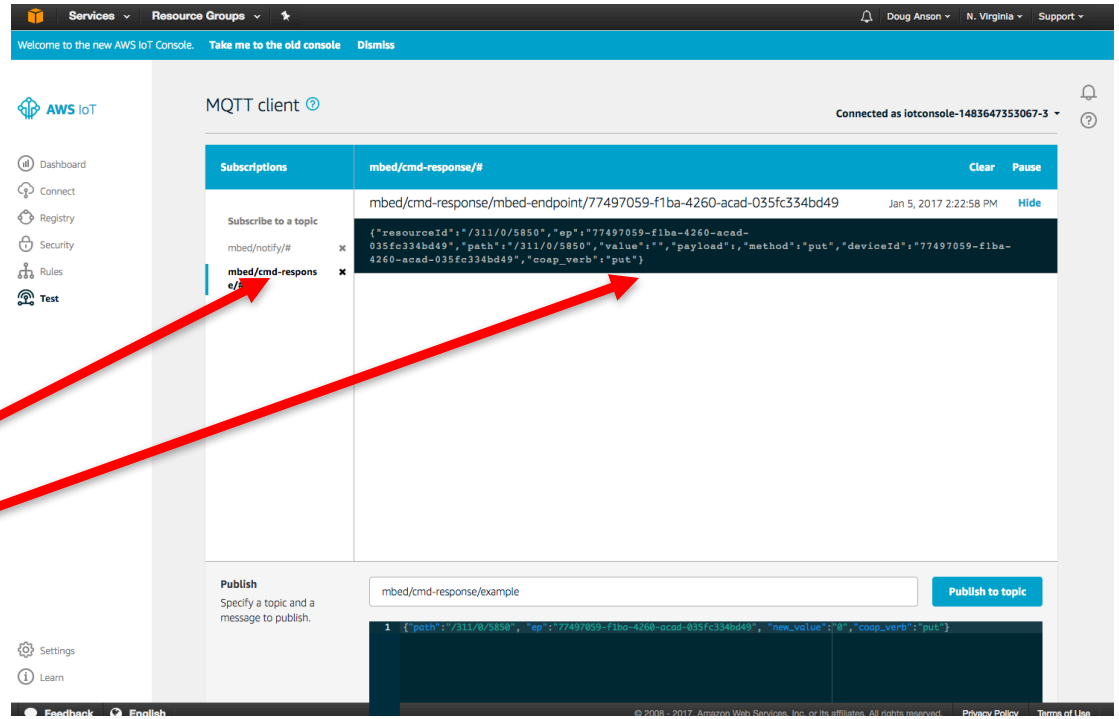
- Enter the Payload (make "new_value":"0" – this will turn the blue LED off):
  {"path":"/311/0/5850", "ep":"ENDPOINT_NAME_GOES_HERE", "new_value":"0","coap_verb":"put"}
  NOTE: Make ENDPOINT_NAME_GOES_HERE your actual endpoint name

- Press "Publish to Topic"

ARM

View your device telemetry in AWS MQTT Client...

- Your blue LED should turn off (or on if you used "1")



- You will get a notification response in the "cmd-response" window – that indicates that the CoAP "put" completed successfully!

ARM

Summary

We have completed the following – congratulations!

- Created our Amazon AWS IoT and mbed environments and PC tool setup
- Imported our mbed endpoint, customized, installed, and ran it
- Created our own Amazon AWS IoT instance
- Imported and configured our ARM AWS IoT Prototype mbed Cloud Bridge
- Examined live telemetry from within Amazon's MQTT Client
- Interacted with our device through Amazon's MQTT Client

Great JOB! Thanks for your time.