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

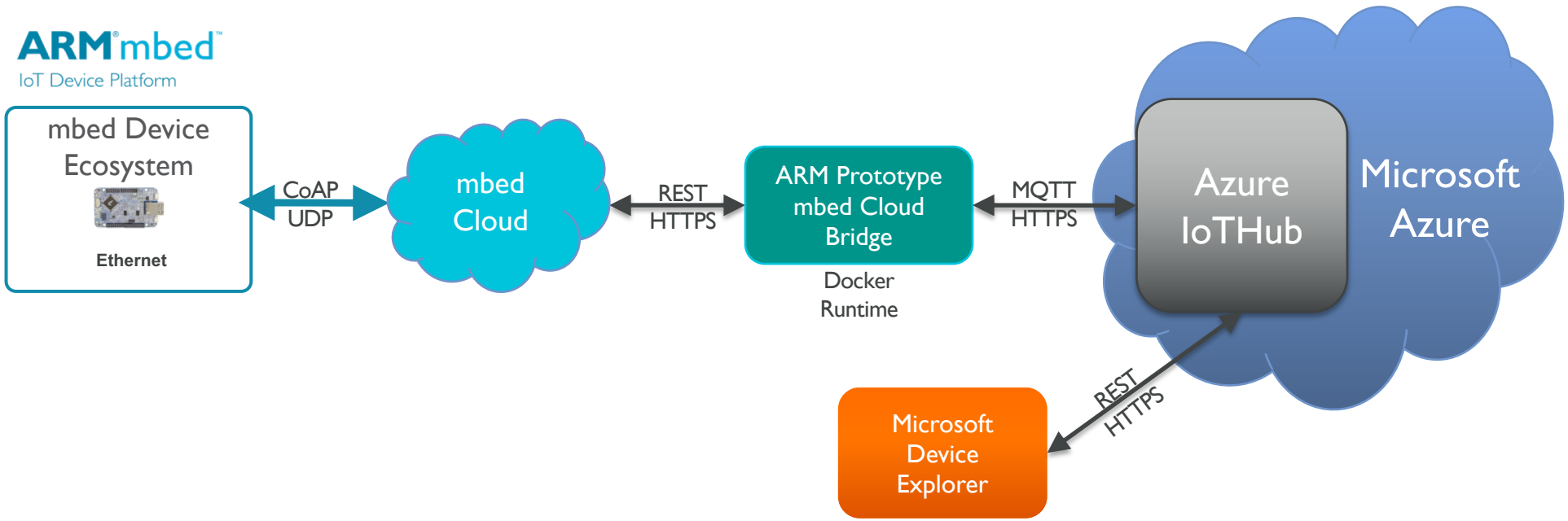
ARM

Doug Anson
Solutions Architect / IoT BU / ARM

Brian Daniels
Applications Engineer / IoT BU / ARM

August 22, 2018 - v2.5

What will we build in this Workshop?



- Connect your mbed device into Azure IoTHub through mbed Cloud and ARMs Prototype IoTHub Cloud Bridge
  - Create a simple mbed device and connect it to mbed Cloud
  - Create an instance of ARM's Prototype Azure IoTHub Cloud bridge and bind it to your Azure and mbed Cloud accounts
  - Exploration of the device data telemetry using Microsoft's Device Explorer

Workshop: Let's get started!

- Create and setup your Azure account (should be completed prior to workshop)
- Install the necessary tools/drivers into your Windows 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...)

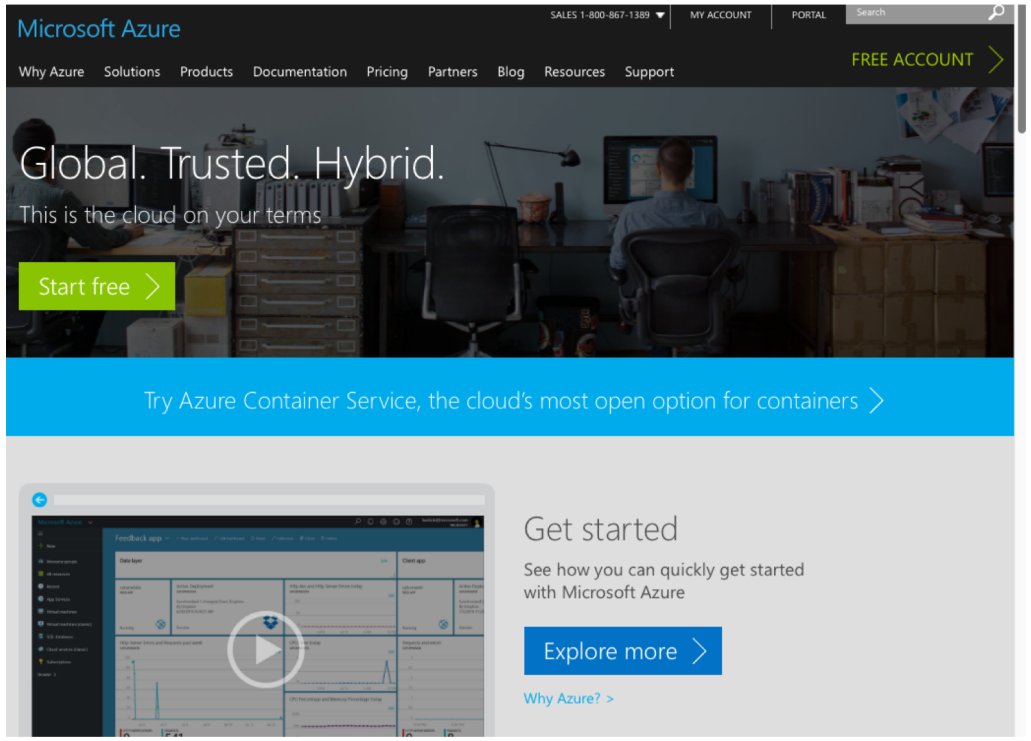
NOTE: During the workshop, be sure to double-check your copy/paste operations...

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

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

Create our Azure Account

- Navigate to the Azure Portal: https://www.azure.com
- Press “Free Account”, “Start Free”
- Complete the sign-up process
- A confirmation email will be sent that must be acknowledged
- Log into your Azure account

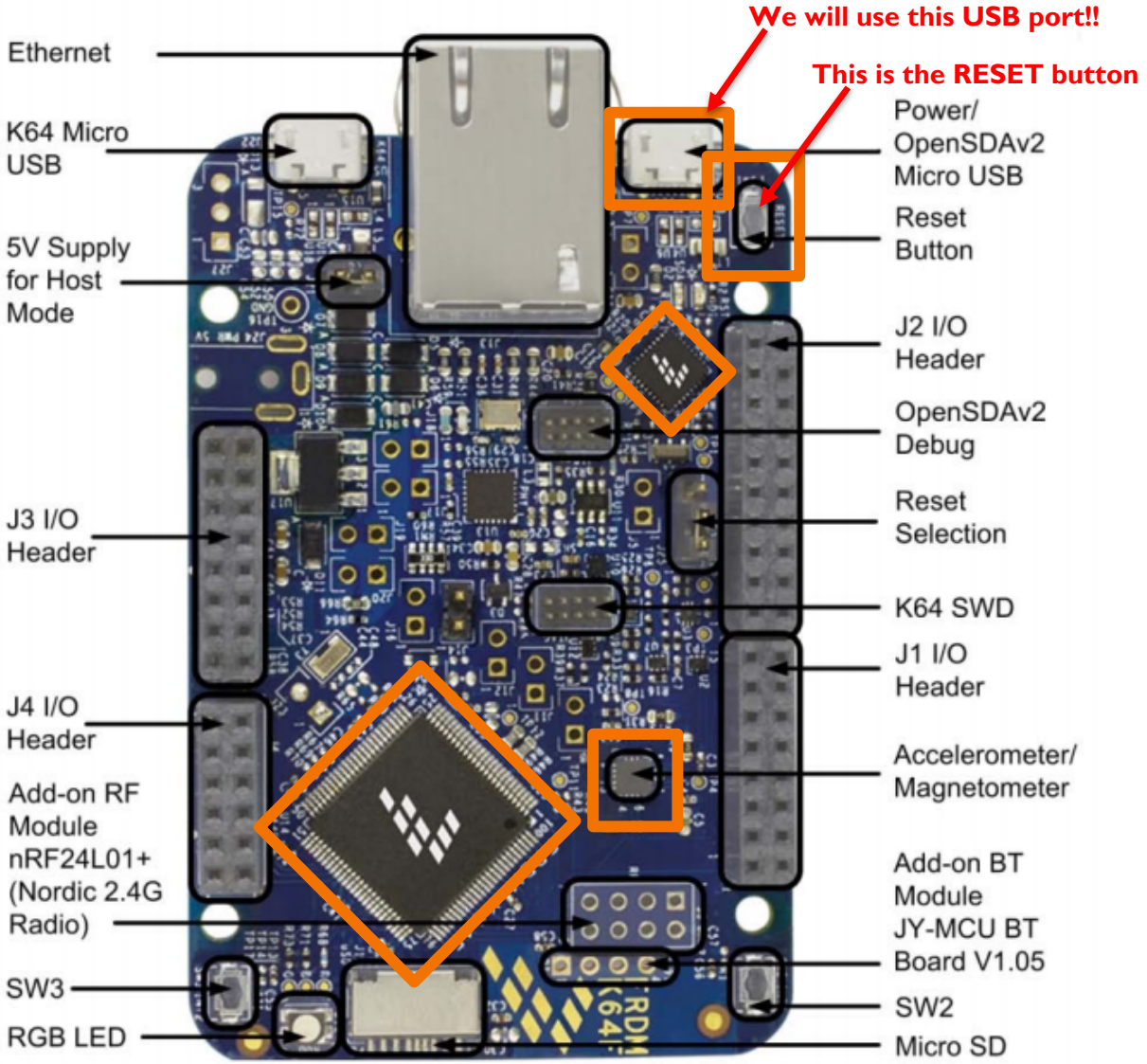


Prior to Workshop

ARM

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

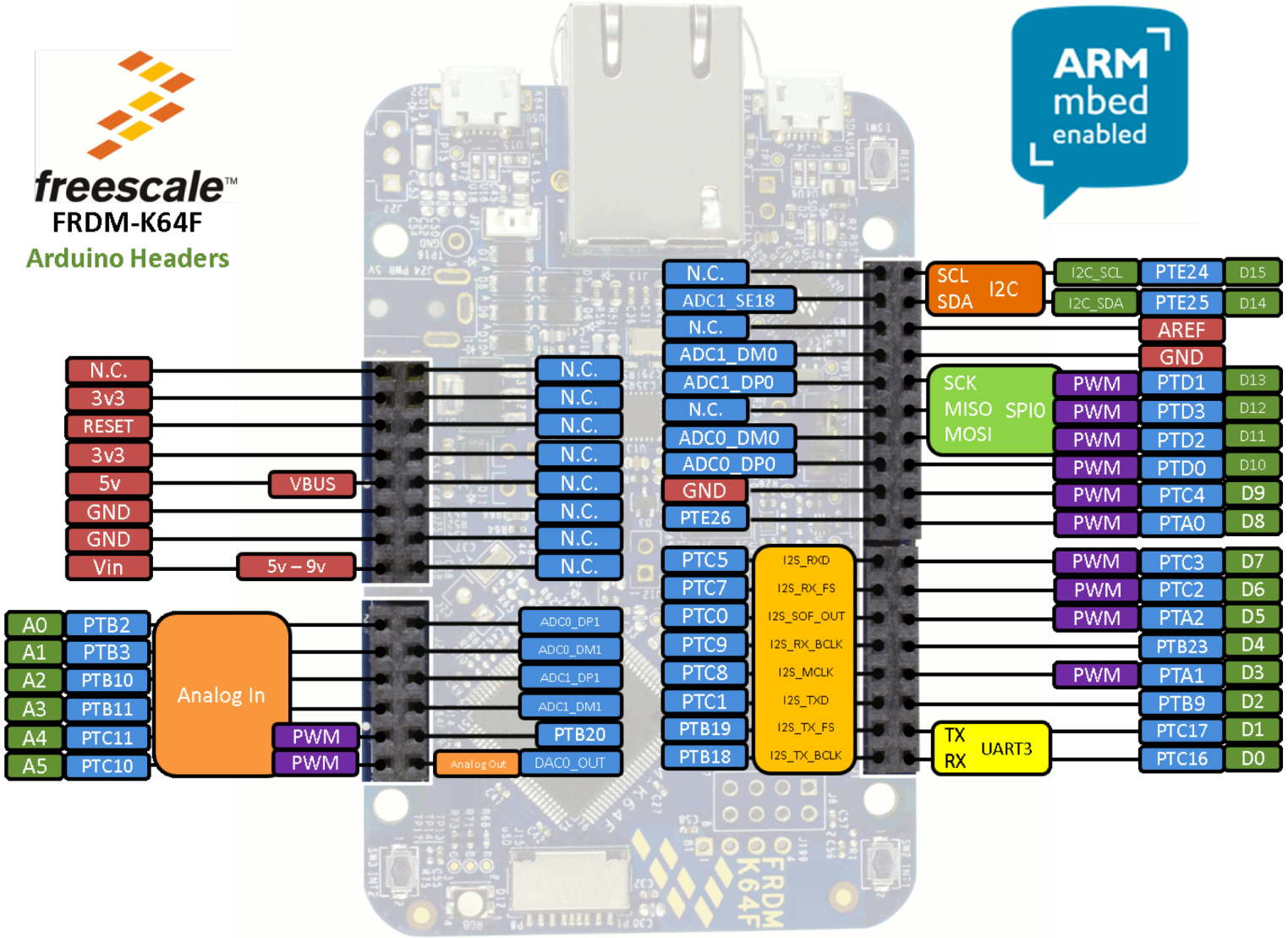


ARM

freescale™

FRDM-K64F

Arduino Headers



ARM

Install the Necessary Tools

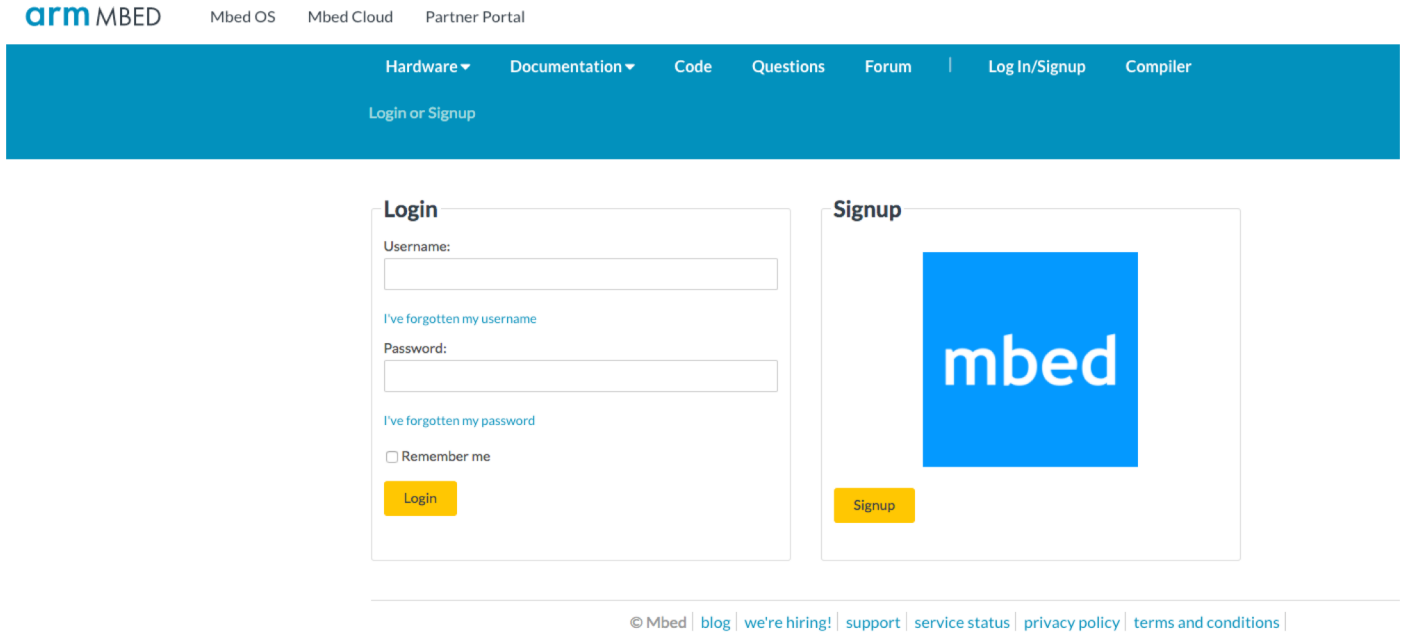
- **This workshop requires a Windows PC (for MS IoTHub Device Explorer)**
  - **IMPORTANT**: Install the mbed USB Serial driver - https://os.mbed.com/media/downloads/drivers/mbedWinSerial_16466.exe
    - Insert the USB cable and mbed device BEFORE running the Serial Driver installation…
    - the installer MUST see the device *first*
    - *You will need administrator access to your Windows PC to install this driver*
  - Serial Terminal: Putty - http://www.putty.org/
  - Chrome and Firefox Browsers may also be used
  - Docker Toolbox installed on your Windows PC (v1.12.2 or later):
    - https://github.com/docker/toolbox/releases/tag/v1.12.2
  - Microsoft Device Explorer (latest version) installed:
    - https://github.com/Azure/azure-iot-sdks/blob/master/tools/DeviceExplorer/doc/how_to_use_device_explorer.md
    - Device Explorer may require additional .NET redistributable downloads... the setup installer will guide you.

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



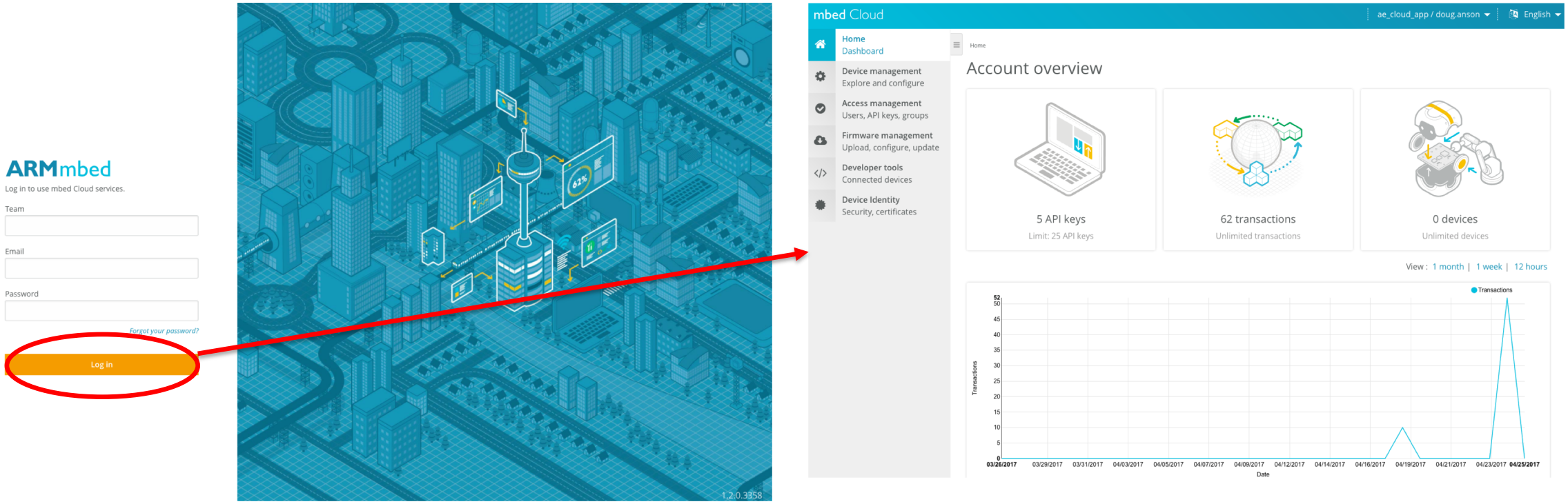
ARM

Create Your mbed Cloud Account…

Prior to Workshop

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

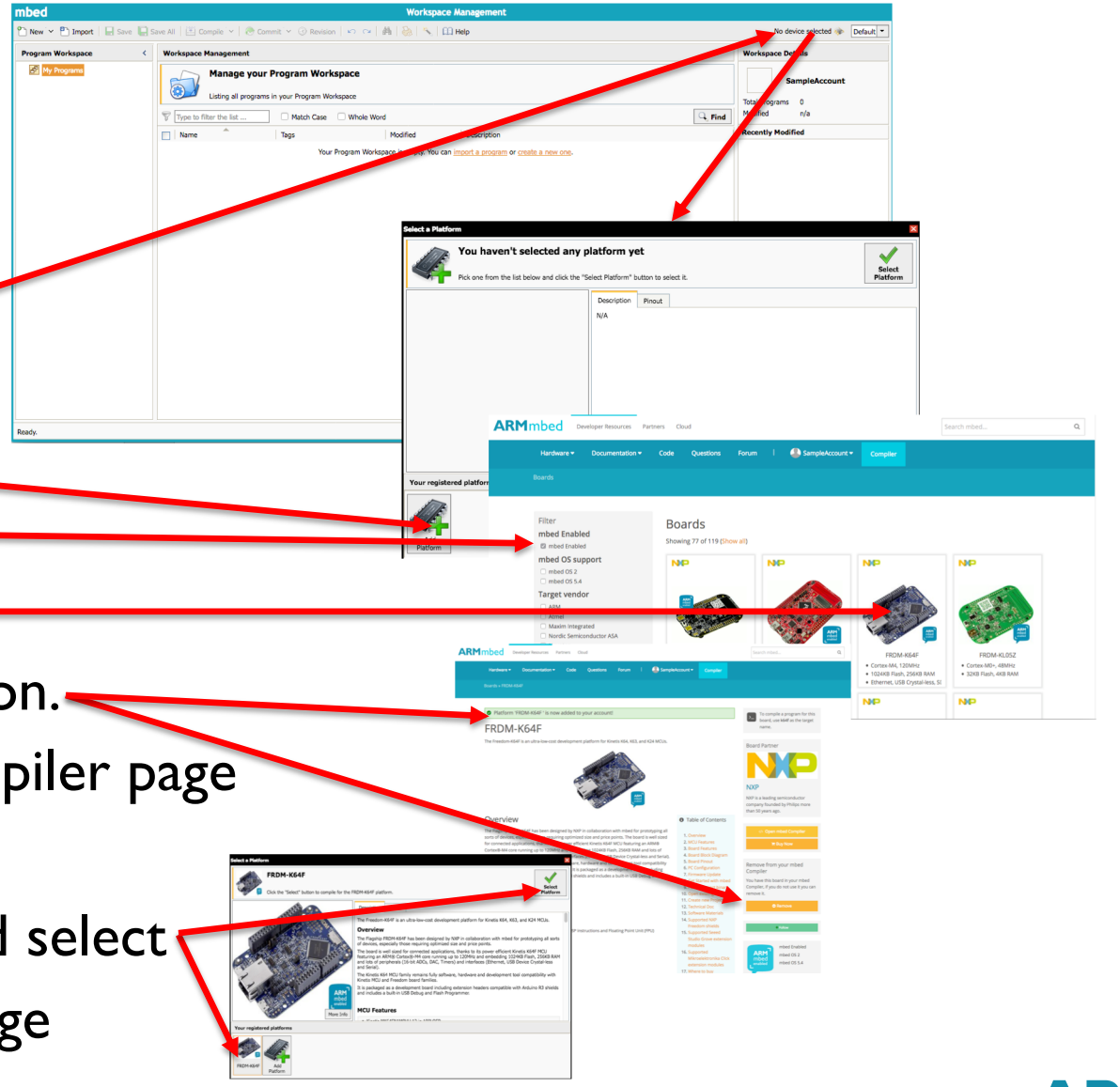
- Confirm that you can log into your mbed device mbed Cloud dashboard



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

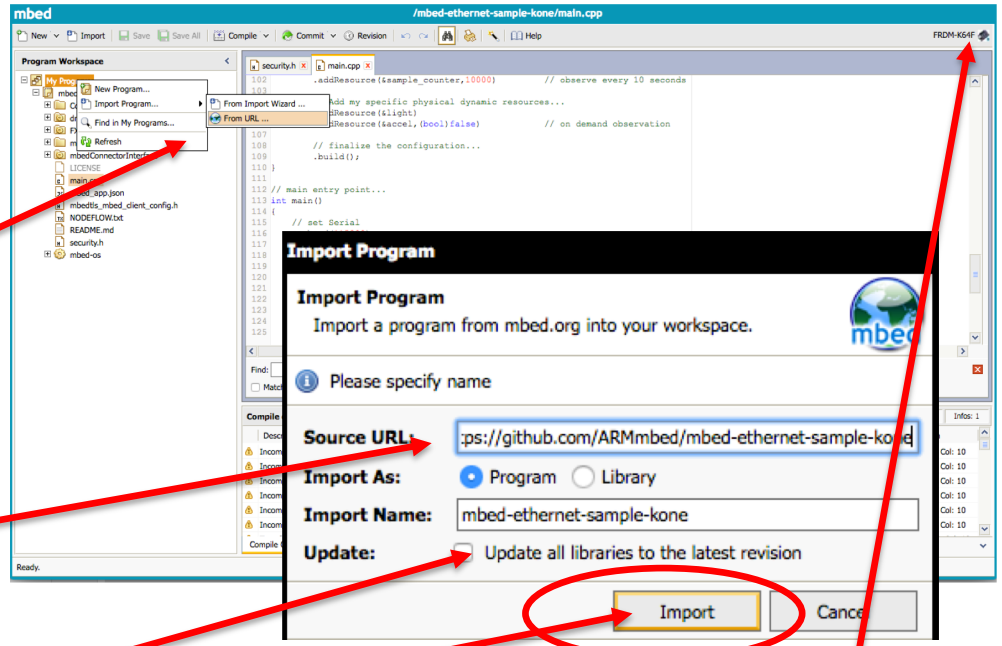


ARM

Import our K64F Endpoint Project

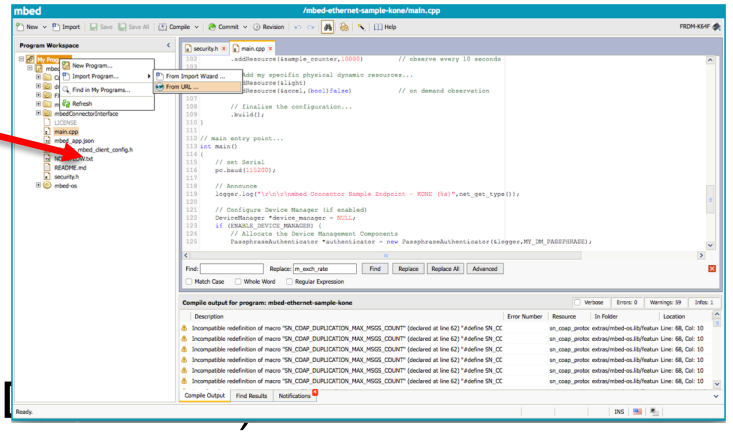
- Go to https://os.mbed.com/
- Select the "Compiler" page
- Right-click on "My Programs"→"Import Program"
- Choose Select "from URL…"
- Enter this URL (Leave the "Update all libraries…" **unchecked**)

  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…

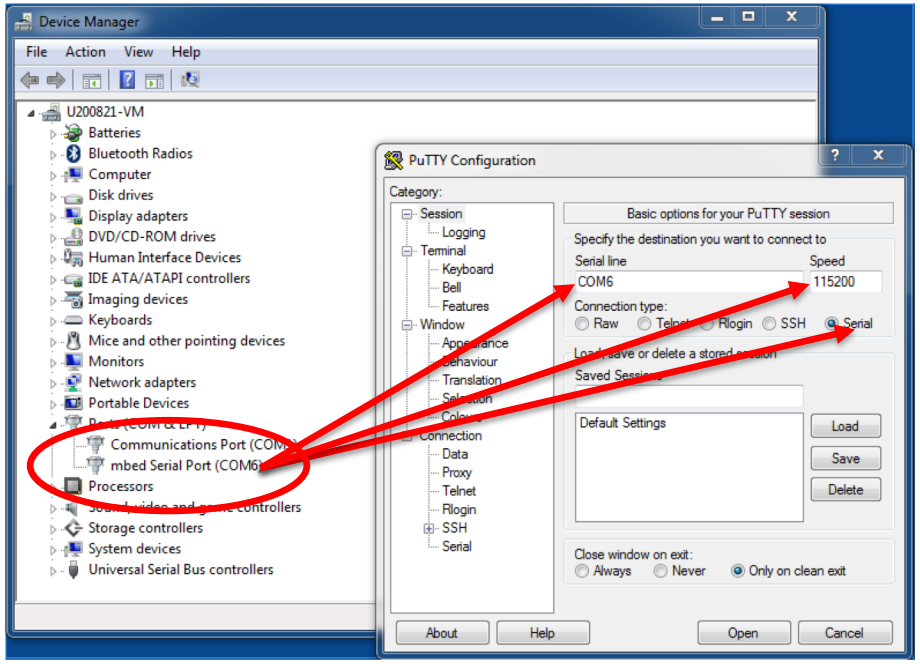


ARM

Running your Endpoint Code

- Bring up your Serial Terminal
  - PuTTY for Windows : You must determine what COM port your mbed device is. Look in the Windows Device Manager FYI

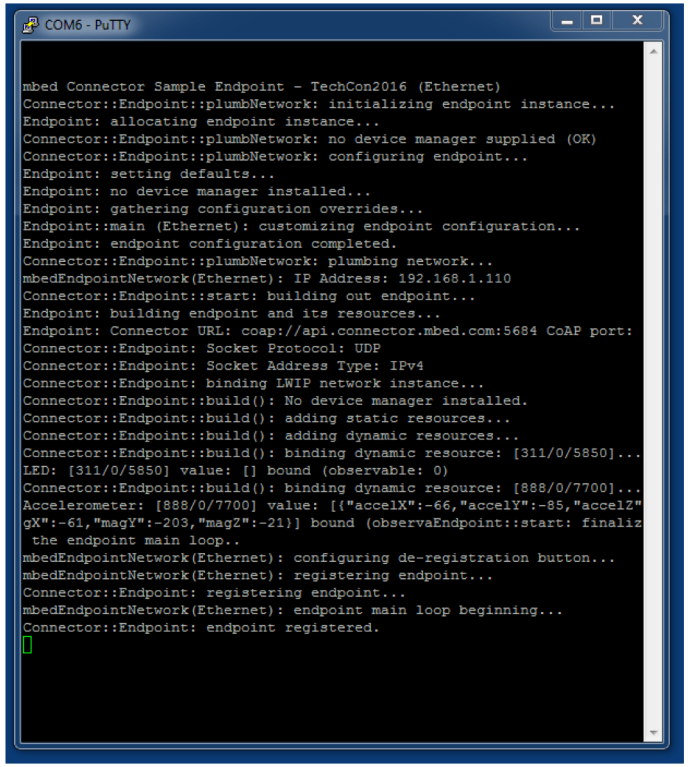
- For the endpoint serial configuration, set the baud rate to 115200 baud, defaults for everything else: (8, N, one)
  - PuTTY for Windows: Ensure that you have "Serial" radio button selected…



ARM

Running your Endpoint Code…

- Connect your Serial Terminal to the K64F:
  - PuTTY for Windows: Press the "Open" button
- Send a "Break" command from the serial terminal (or press the RESET button on the K64F)
  - 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 IoTHub Prototype mbed Cloud Bridge...

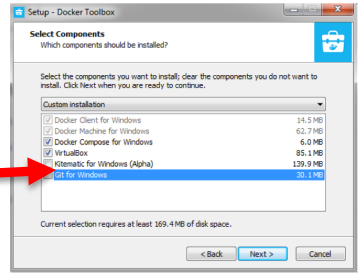
ARM IoTHub mbed Cloud Bridge Setup

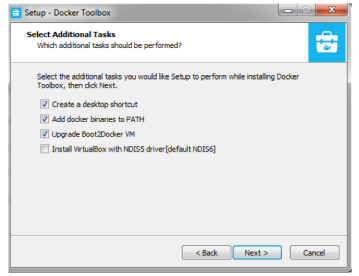
What we will do next…

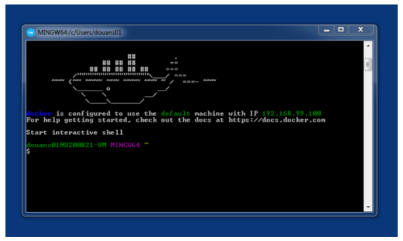
- Ensure that we have the necessary pre-requisites setup on our Windows PC
- Create our mbed Cloud API Key/Token
- Create our IoTHub instance in Azure
- Using DeviceExplorer, we create an IoTHubOwner SAS Token
- Import our Prototype mbed Cloud Bridge instance as a Docker image into our Docker runtime
- Configure our Prototype mbed Cloud Bridge for IoTHub

Double check: all our needed Tools Installed

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







- MS DeviceExplorer installed and available.
  - https://github.com/Azure/azure-iot-sdks/blob/master/tools/DeviceExplorer/doc/how_to_use_device_explorer.md

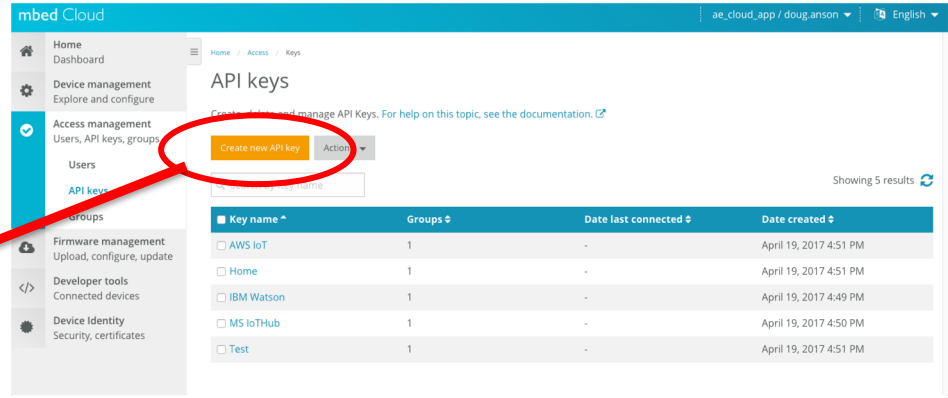
- Windows mbed USB driver installed and functioning properly.
  - "DAPLINK" flash drive present
  - "mbed Serial Port" seen in Windows Device Manager when K64F USB is connected

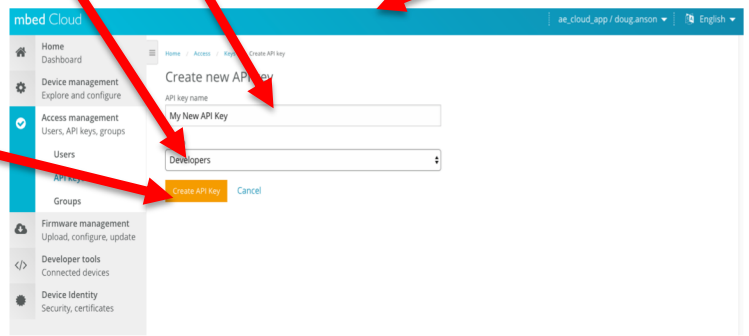
- Putty installed and operational

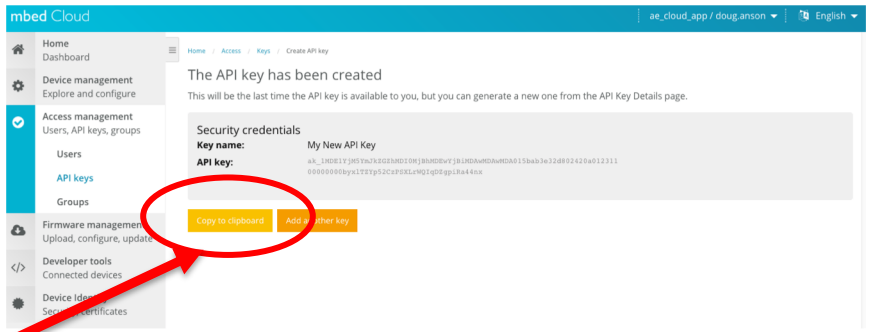
ARM

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

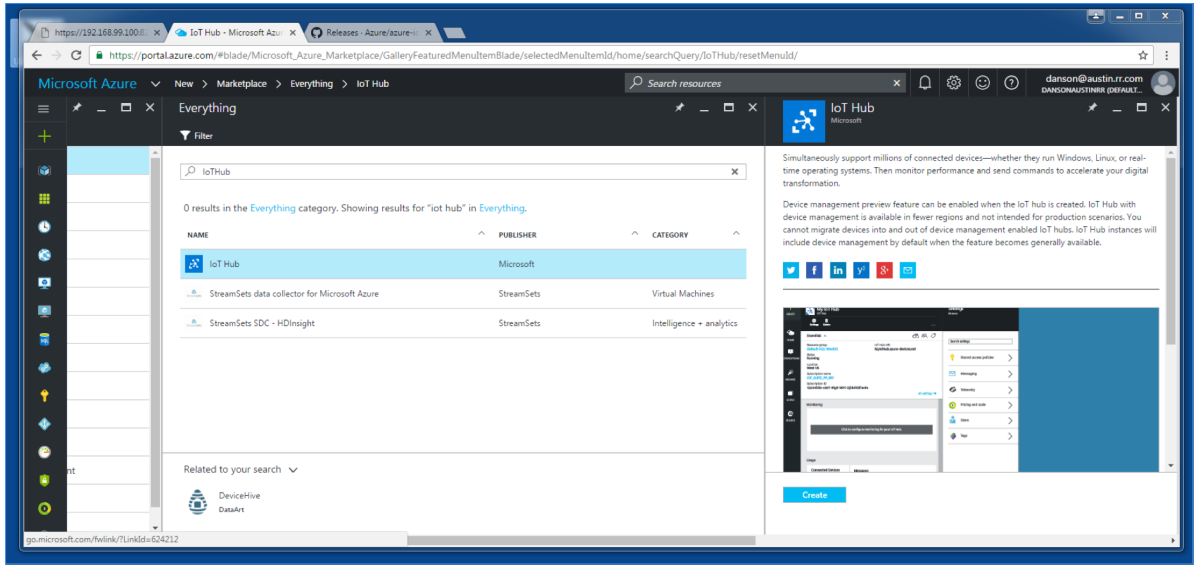


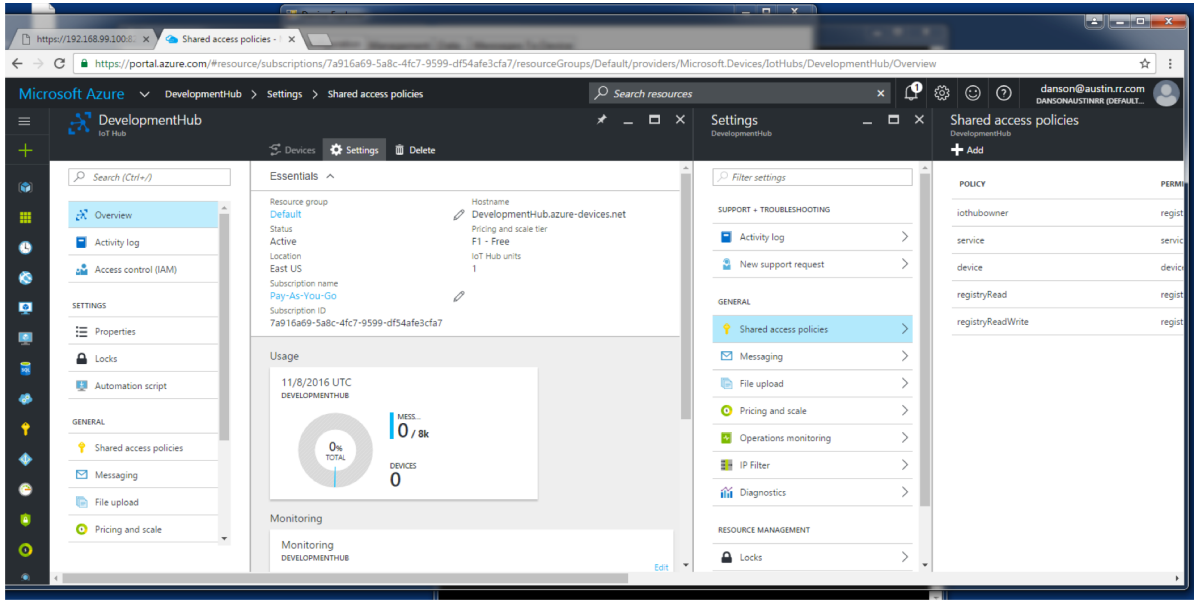




Create our Azure IoTHub Instance

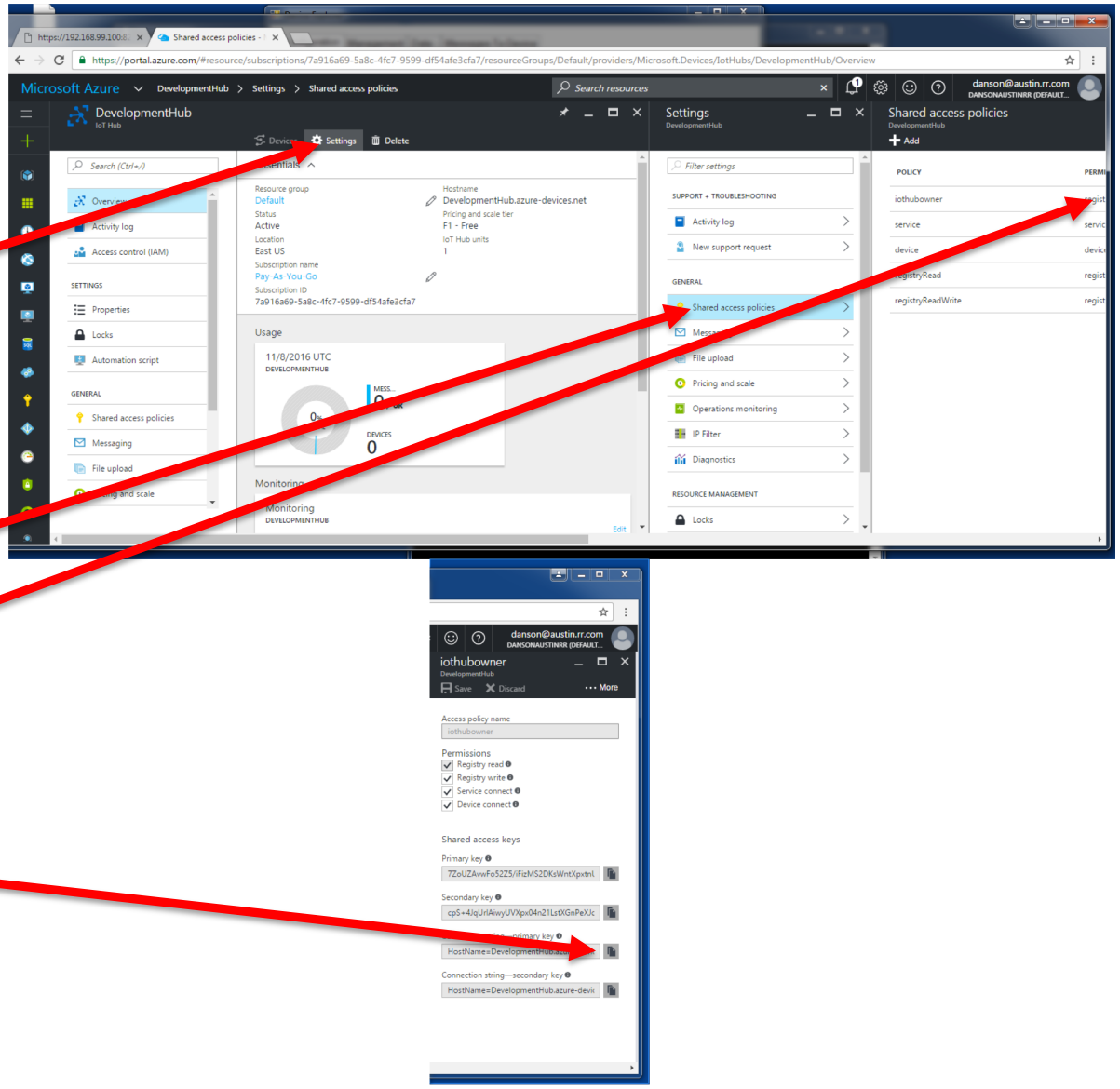
- Log into your Azure Portal
- Search for IoTHub
- Select and “Create”
- Give your IoTHub a name
- Be sure to choose “Free” Tier
- Azure will create the IoTHub





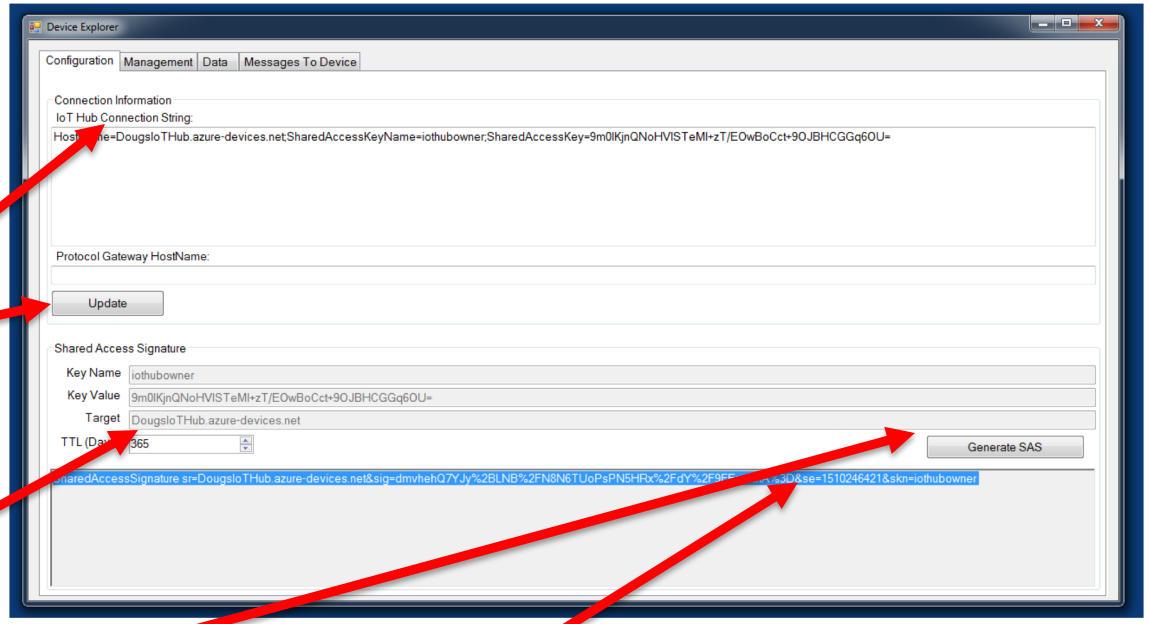
Copy our IoTHub Connection String

- In the IoTHub dashboard
- Select your IoTHub Resource
- Press “Settings”
- Select “Shared Access Policies”
- Select “iothubowner”
- Select Copy to Clipboard
  - Primary Connection String



Create our SAS Token with DeviceExplorer

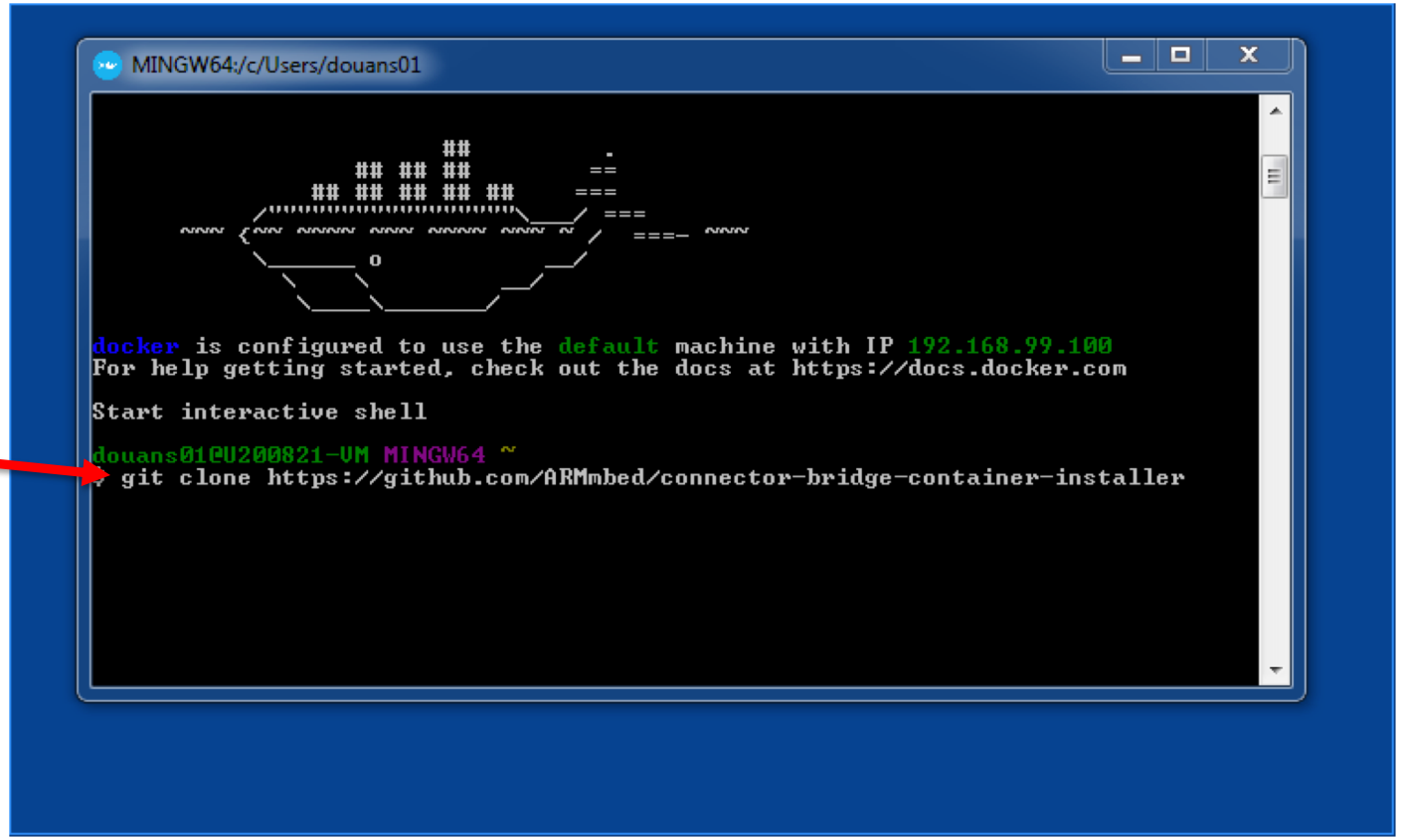
- Launch DeviceExplorer
- Paste your connection string here
- Press “Update”
- Ensure “365” is entered here
- Press “Generate SAS”
- Copy ALL contents of your SAS Token
  - Save this for later!!



ARM

Import our mbed Cloud IoTHub Bridge Installer

- Launch the Docker QuickStart Terminal
- Type the following "git" command

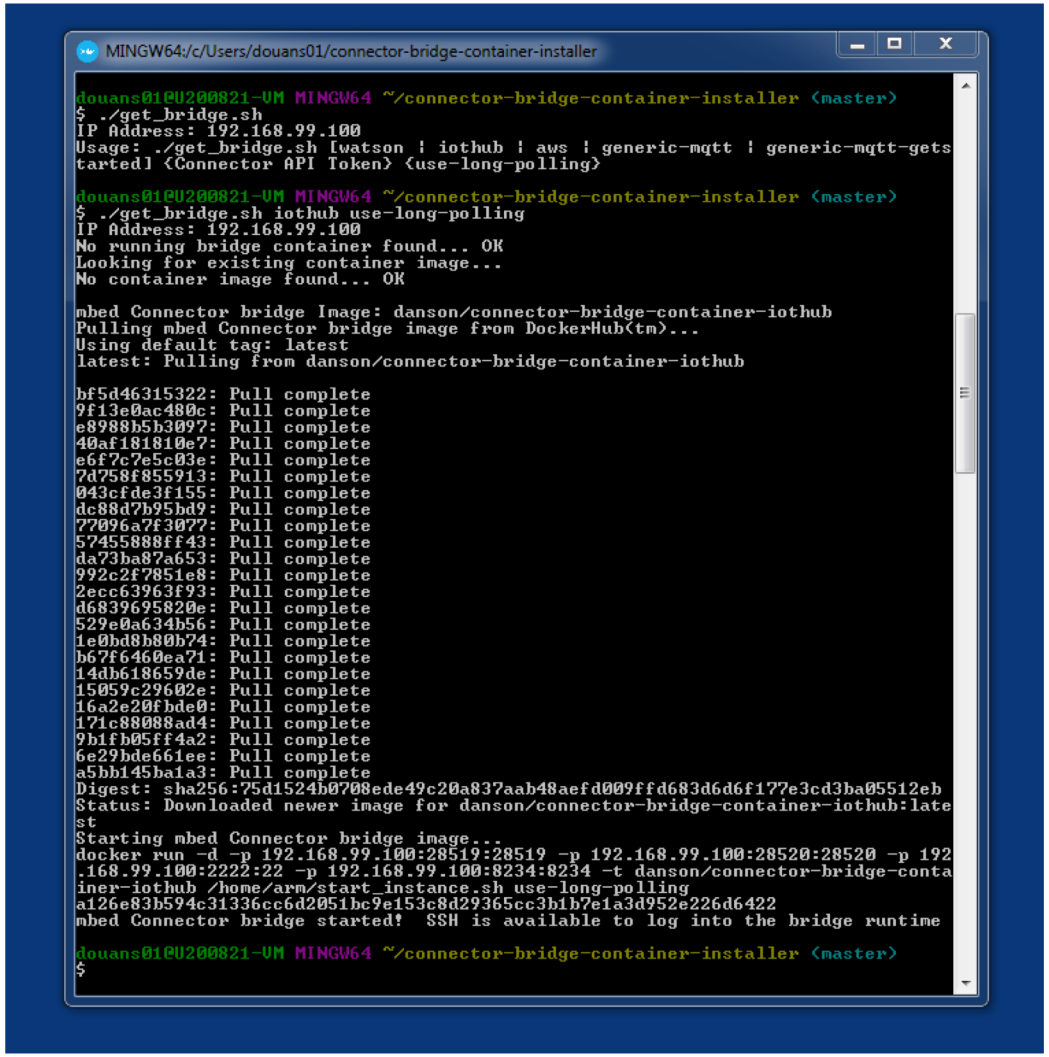


% git clone https://github.com/ARMmbed/connector-bridge-container-installer

ARM

Import our mbed Cloud IoTHub 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 iothub use-long-polling
- Bridge container will import from DockerHub



ARM

Configure our IoTHub Bridge (Windows)

- Your Container IP address will be:

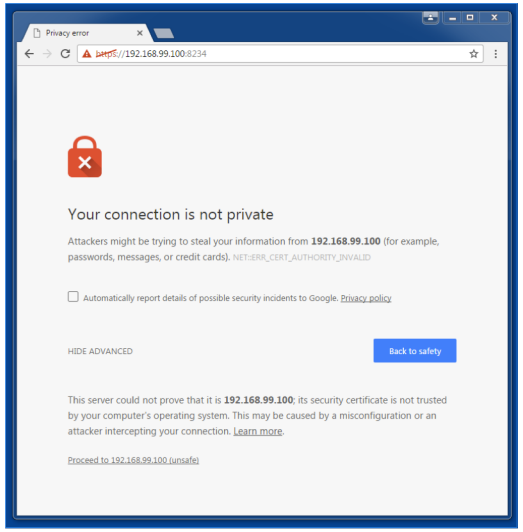
  192.168.99.100

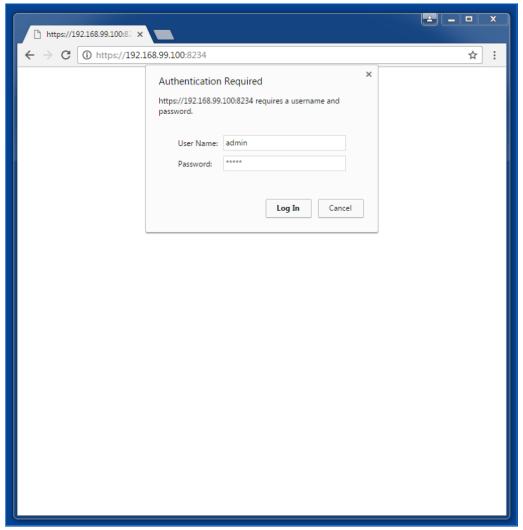
- Bring up Firefox/Chrome and go to:

  https://192.168.99.100:8234

- Accept the self signed certificate

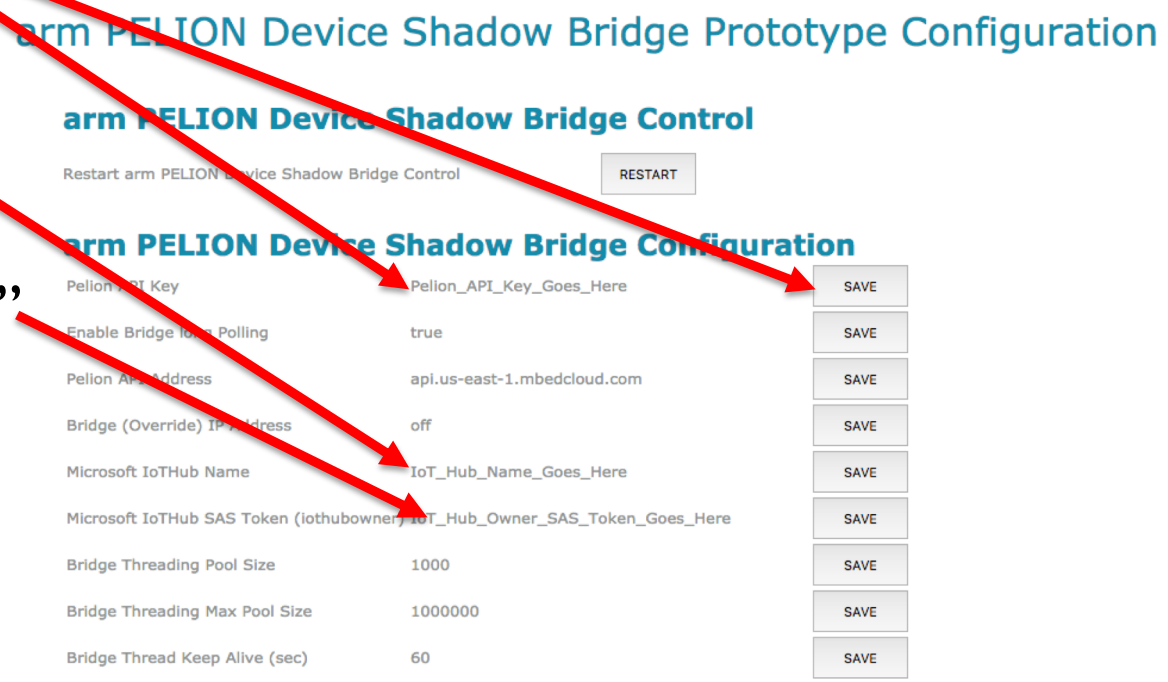
- Username: admin PW: admin





Configure the mbed Cloud Bridge for IoTHub…

- Enter the mbed Cloud API Key/Token, *then* Press "Save"
- Enter the name of your IoTHub, Press "Save"
- Enter your iothubowner SAS Token, Press "Save"
- Now, Press "RESTART"



- Your mbed Cloud Bridge is now configured for IoTHub

ARM

Status Check

So far we've completed the following

- Ensure that we have the necessary pre-requisites setup on our Windows PC
- Create our mbed Cloud API Key/Token
- Create our IoTHub instance in Azure
- Using DeviceExplorer, we create an IoTHubOwner SAS Token
- Import our bridge instance: a Docker image into our Docker Toolbox runtime
- Configure our Prototype mbed Cloud Bridge for IoTHub

With our bridge now operational, we should be able to restart our endpoint and see messages coming into our IoTHub via DeviceExplorer

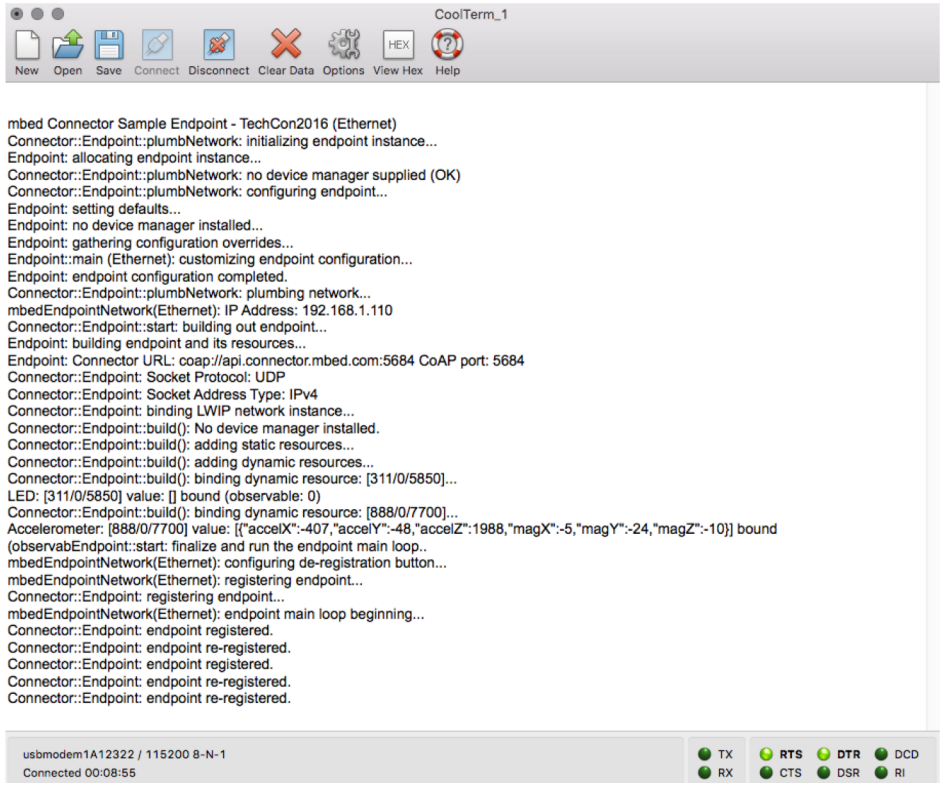
Putting it all Together

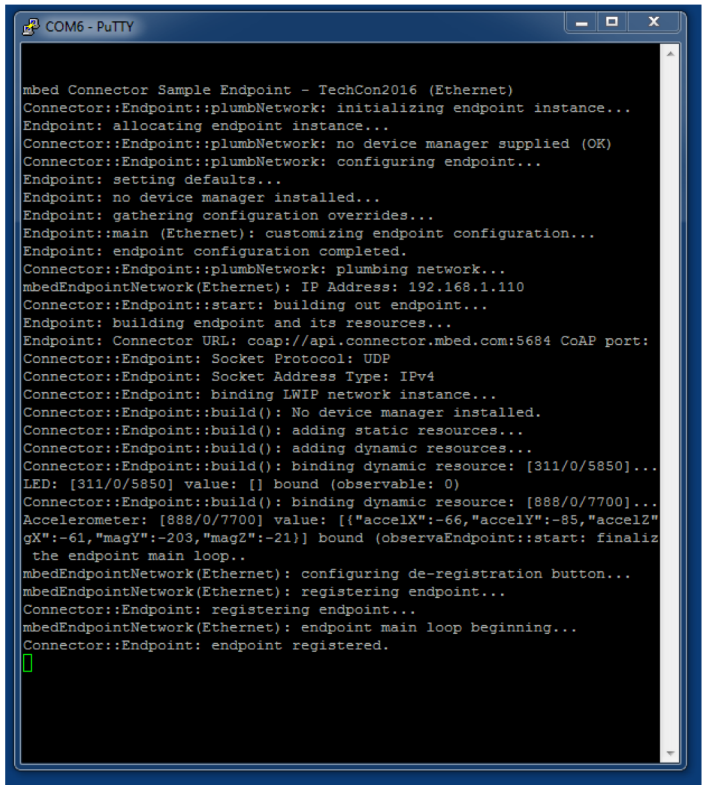
- Press the "reset" button on your mbed device (next to the USB port)
- Ensure that you see "endpoint registered" in the serial terminal
- Bring up DeviceExplorer

Lets check how things are working…

Putting it all Together: Check the Serial Terminal

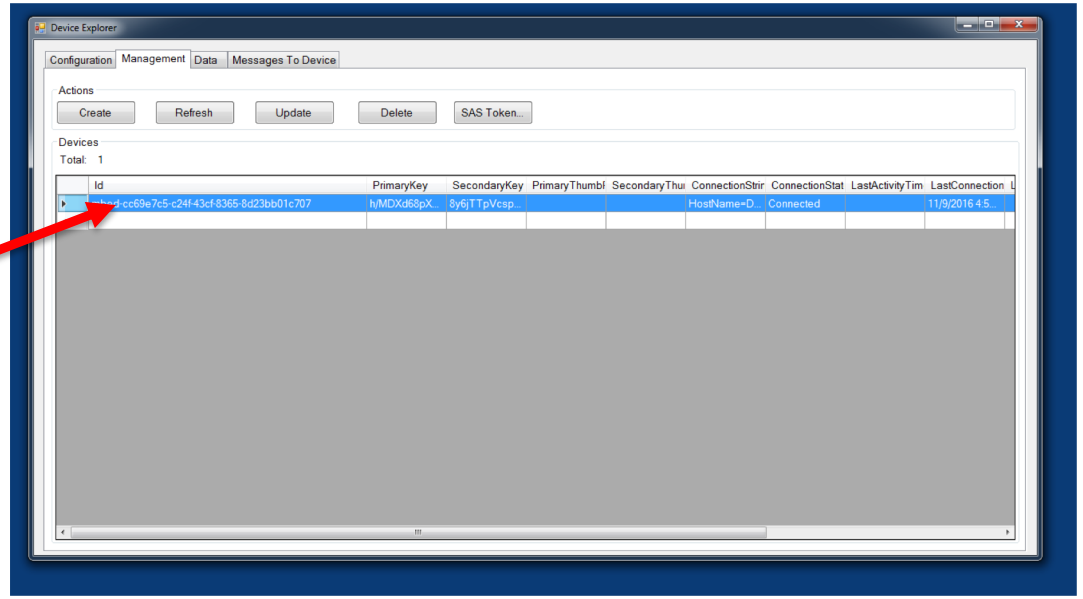
- You should see output that looks something like this:
  - Make sure that you see a message like "endpoint registered"

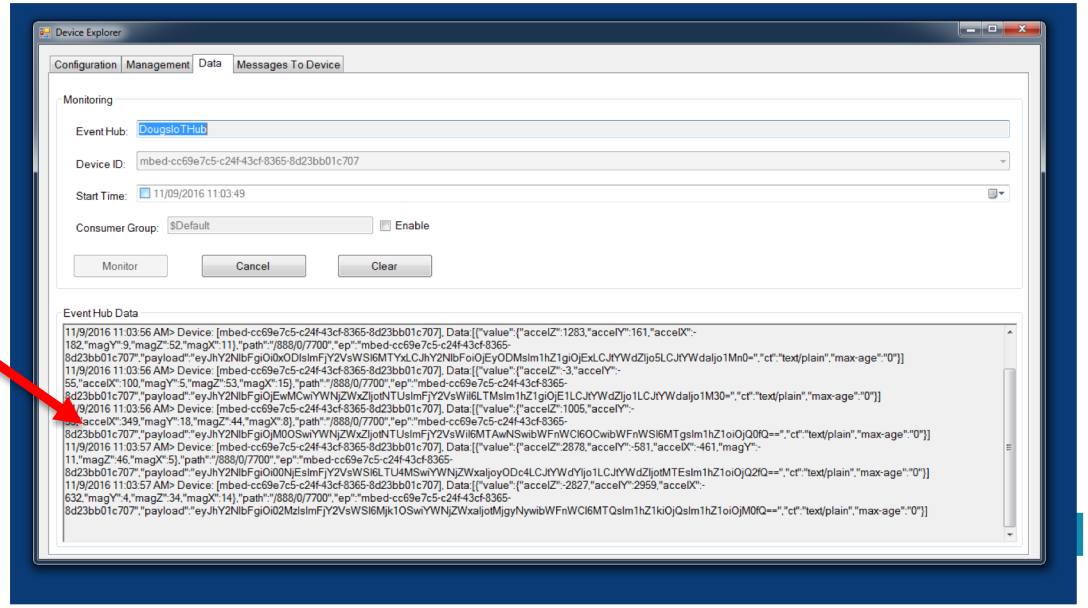




View your device telemetry in DeviceExplorer

- Bring up your Device Explorer
- Under the "Management" tab you should see your device listed
- Under "Data" you should see output (shake your device…)
  - Press "monitor" to start monitoring



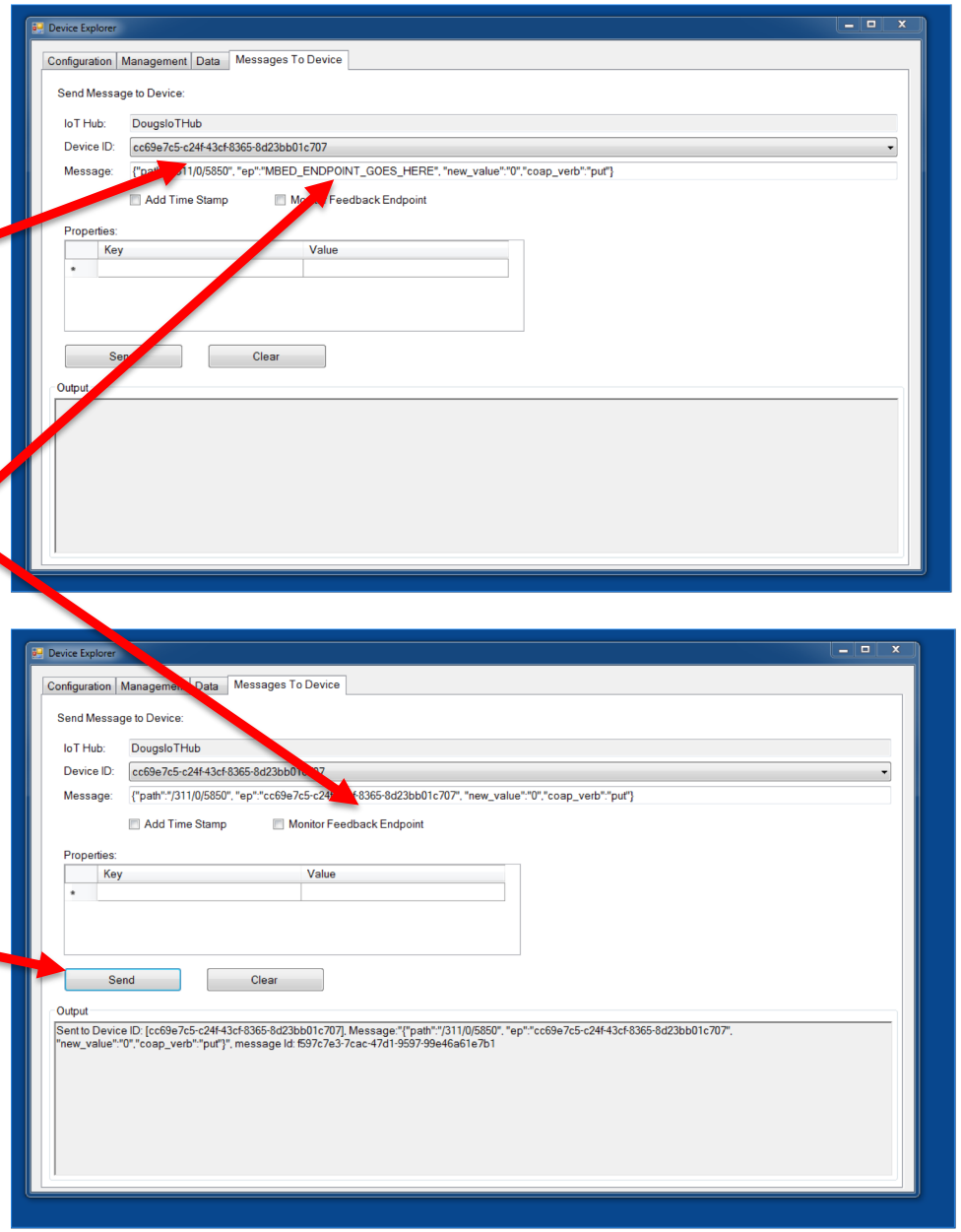


Interact with your device in DeviceExplorer

- Select “Messages to Device”
- Enter the following JSON here

  {"path":"/311/0/5850", "ep":"MBED_ENDPOINT_GOES_HERE", "new_value":"0","coap_verb":"put"}

- Replace MBED_ENDPOINT_GOES_HERE with your actual device name (look in the Data you've received for quick copy...)
- Press “Send”
-
- Blue LED should turn ON or OFF depending on “new_value” being “0” ,“1”



Summary

We have completed the following – congratulations!

- Created our Azure IoTHub and mbed environments and PC tool setup
- Imported our mbed endpoint, customized, installed, and ran it
- Created our own Azure IoTHub instance
- Imported and configured our ARM IoTHub Prototype mbed Cloud Bridge
- Examined live telemetry from within MS's DeviceExplorer
- Interacted with our device through MS's DeviceExplorer

Great JOB! Thanks for your time.