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

ARM

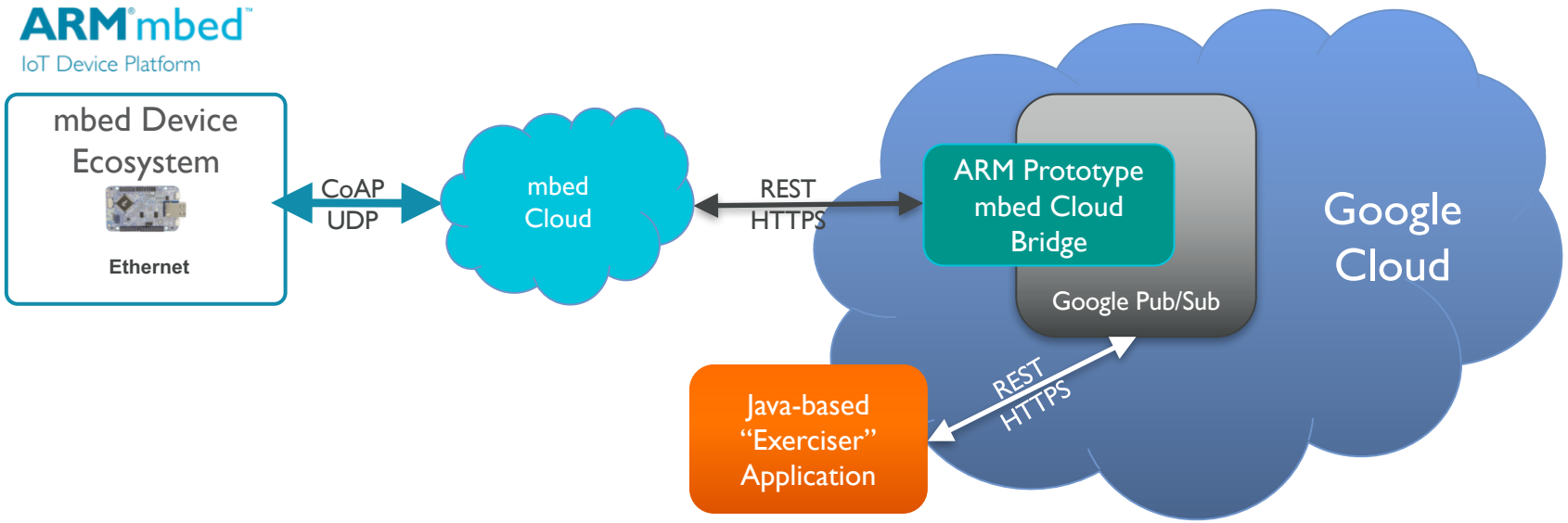
Doug Anson
Solutions Architect / IoT BU / ARM

Brian Daniels
Applications Engineer / IoT BU / ARM

Dec / 03 / 2017 – v2.1

©ARM 2017

What will we build in this Workshop?



- Connect your mbed device into Google Cloud through mbed Cloud and ARMs Prototype Google Cloud Bridge
  - Create a simple mbed device and connect it to mbed Cloud
  - Create an instance of ARM's Prototype Google Cloud bridge and bind it to your Google and mbed Cloud accounts
  - Exploration of the device data telemetry using a java-based "exerciser" application utilizing Google's Pub/Sub APIs

©ARM 2017

Workshop: Let's get started!

- Create and setup your Google 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...

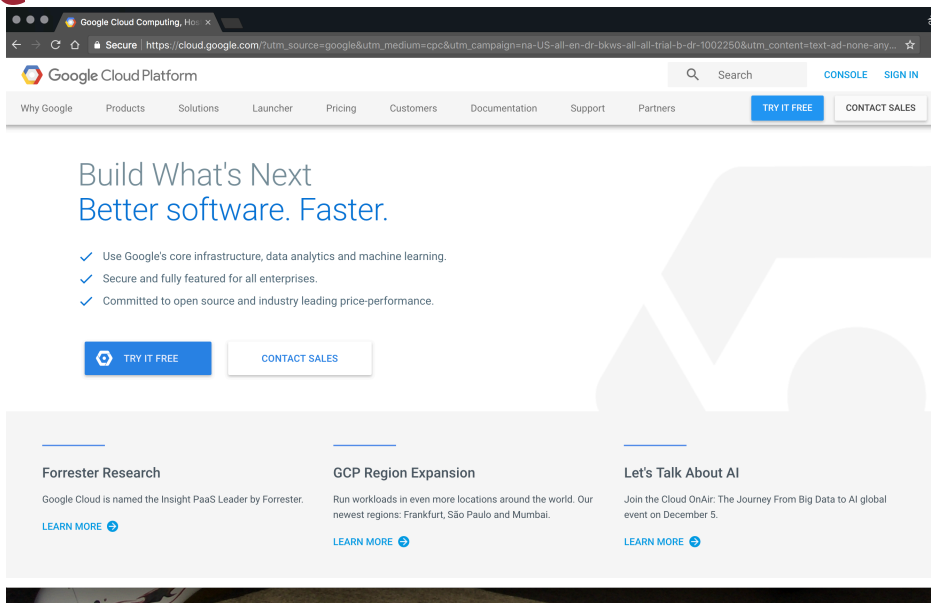
Quick Links: Visit https://github.com/ARMmbed/anson_workshop_google_cloud_2017

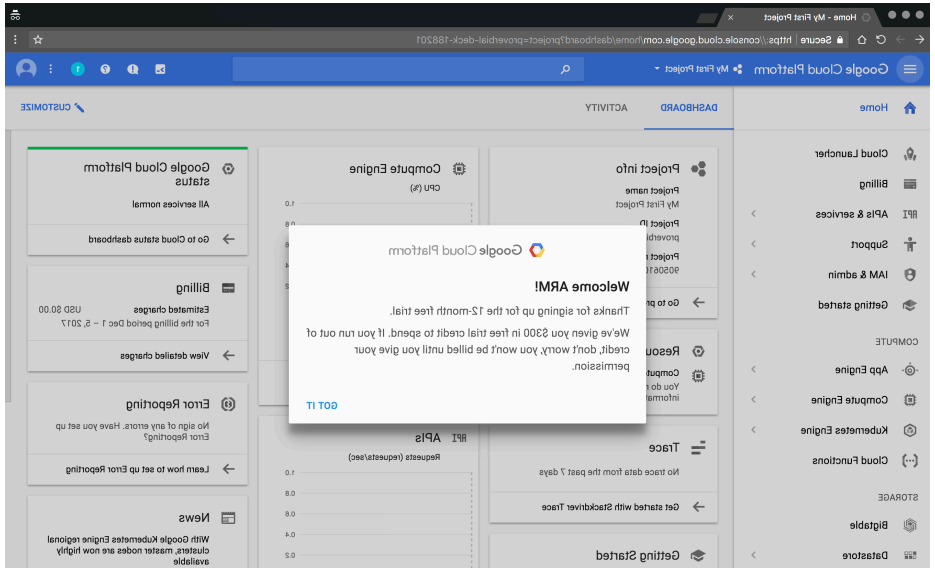
- README.md has all of the links in the workshop + this presentation in PDF format

©ARM 2017

Create our Google Cloud Account

- Navigate to the Google Console: https://cloud.google.com
- Press ”Try It Free”
- Complete the signup process (you’ll need to supply a credit card – but the trial is for 1 year)
- You’ll then be at the main dashboard



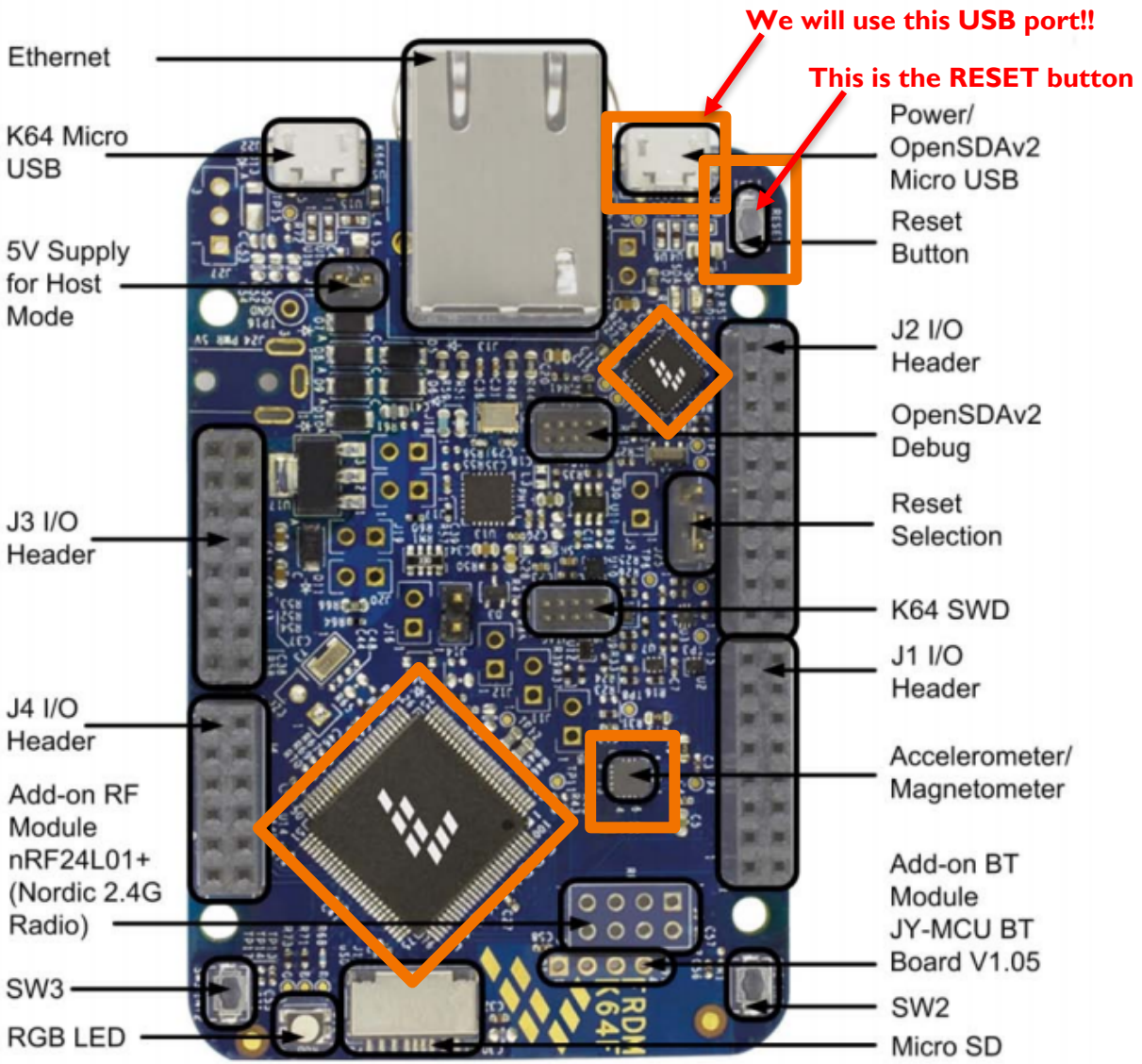


Prior to Workshop

©ARM 2017

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

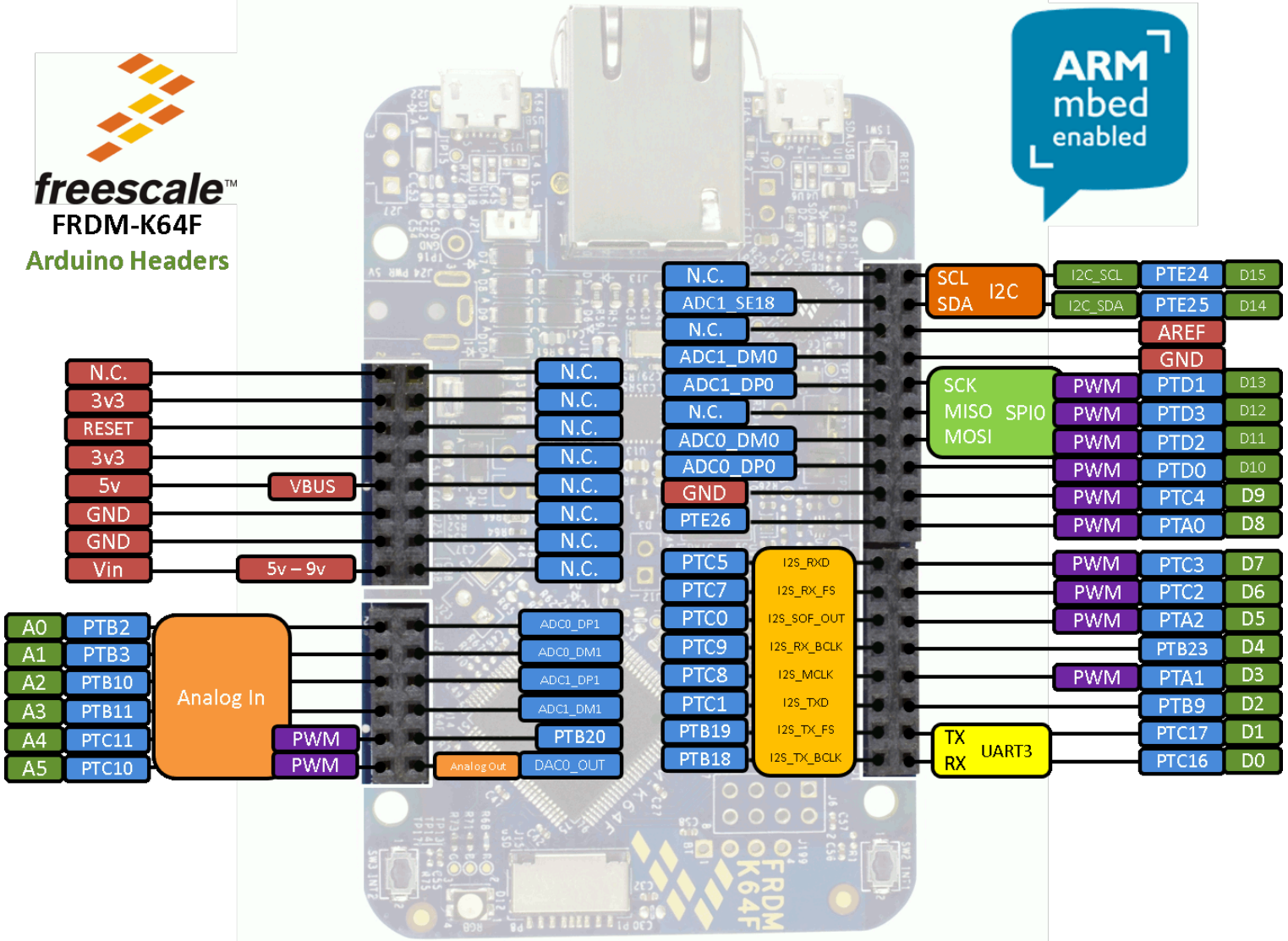


©ARM 2017
ARM

freescale™

FRDM-K64F

Arduino Headers



©ARM 2017

ARM

Install the Necessary Tools

- **Windows IMPORTANT**: Install the mbed USB Serial driver -

  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*
  - *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

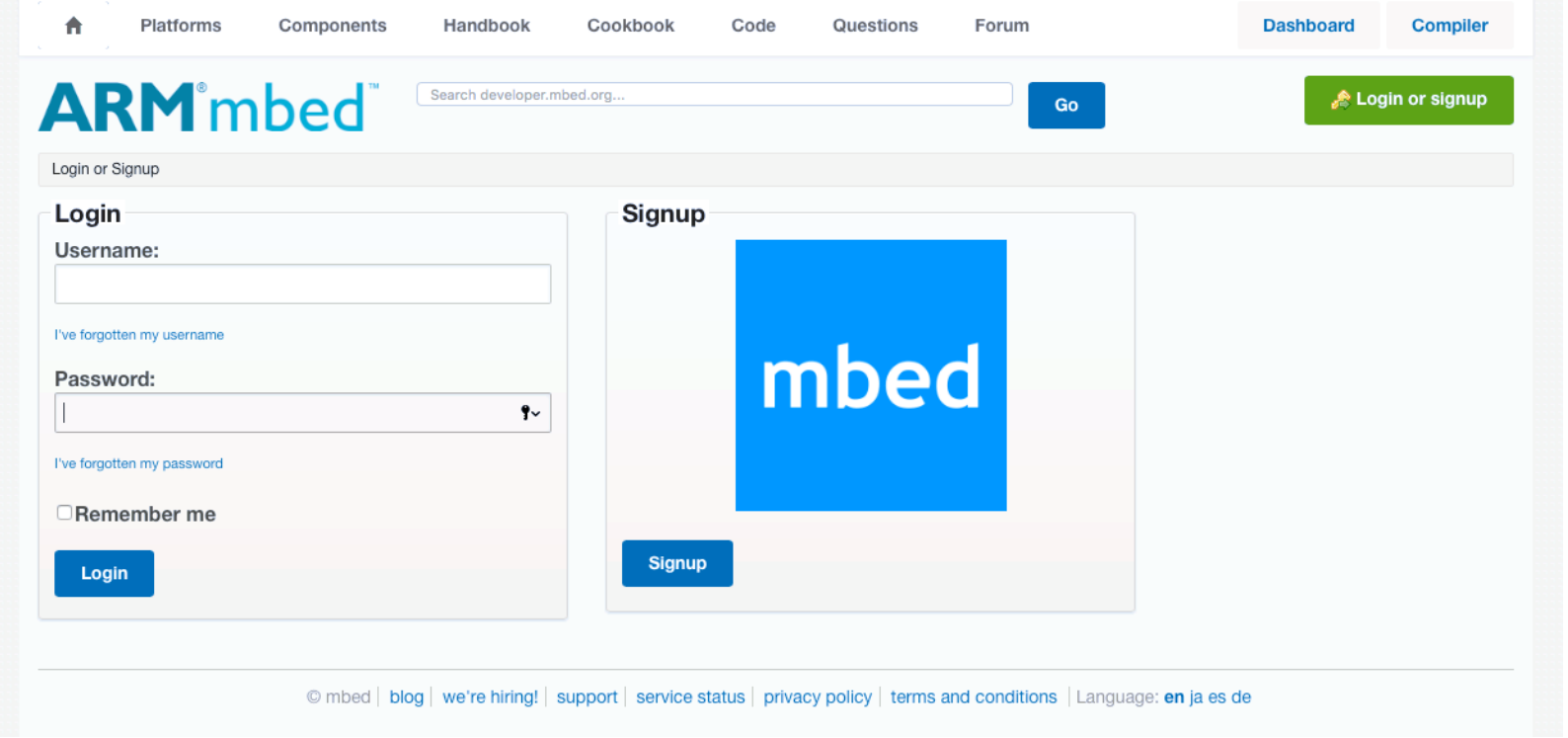
Connect both your USB cable and Ethernet cable to the K64F and leave it there for now

©ARM 2017

Create Your mbed Account

Prior to Workshop

- Navigate to: https://os.mbed.com/
- Create an Account

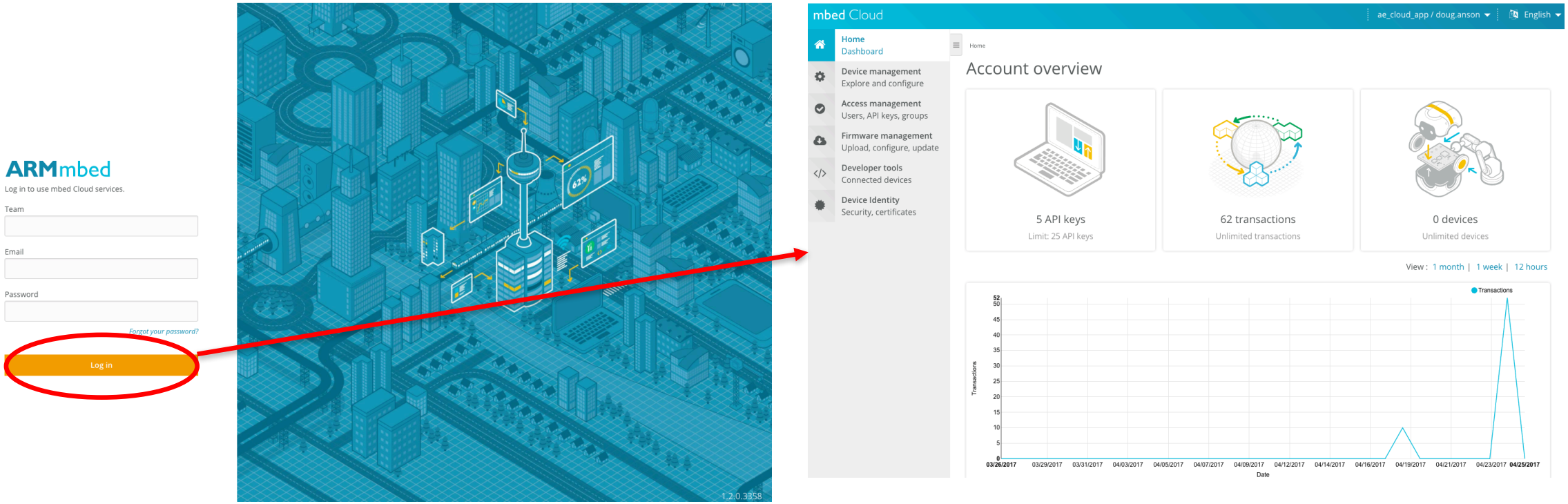


©ARM 2017
ARM

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

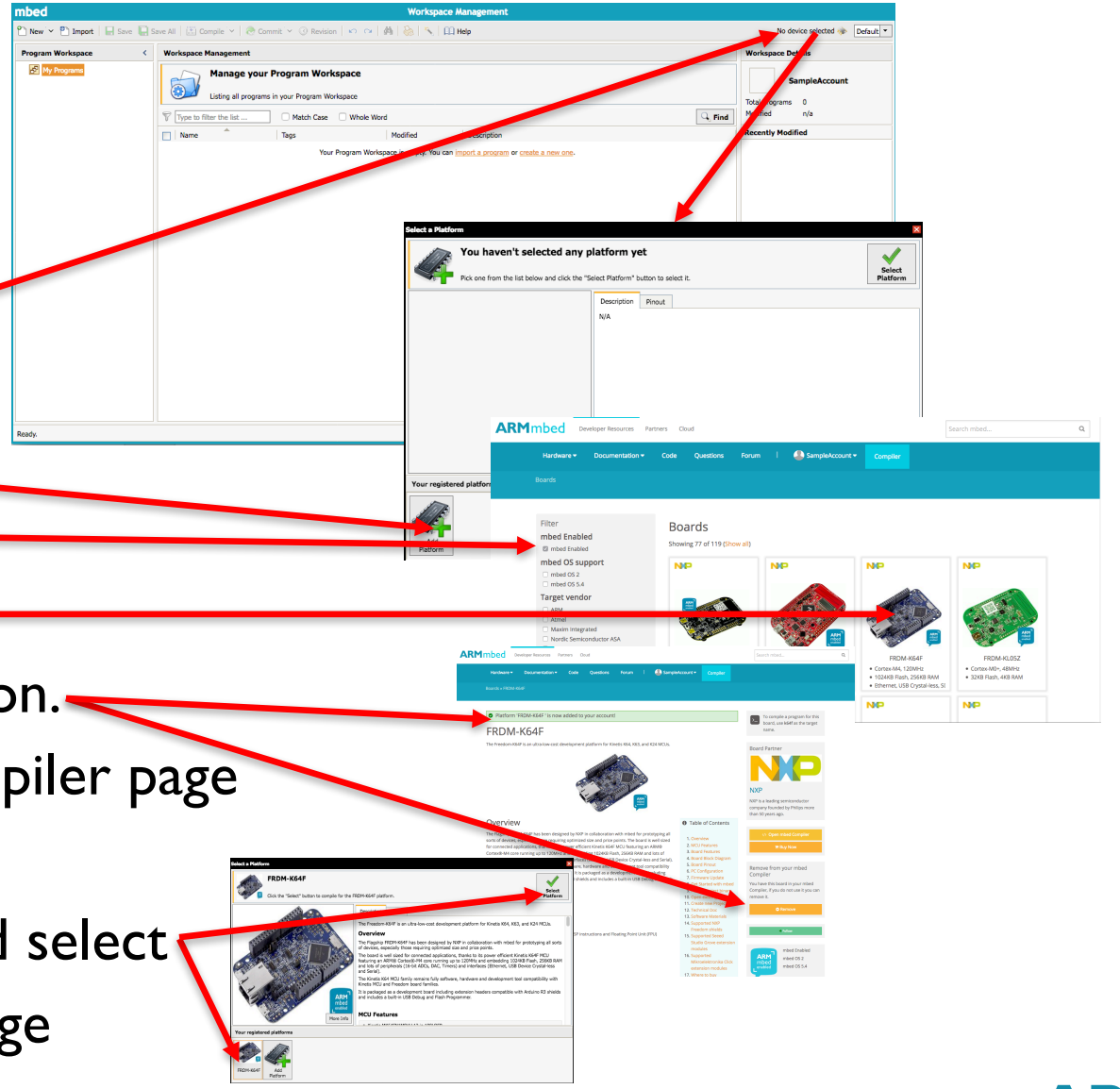


©ARM 2017

Log into the Online IDE – Add the K64F Compile Target

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

- 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

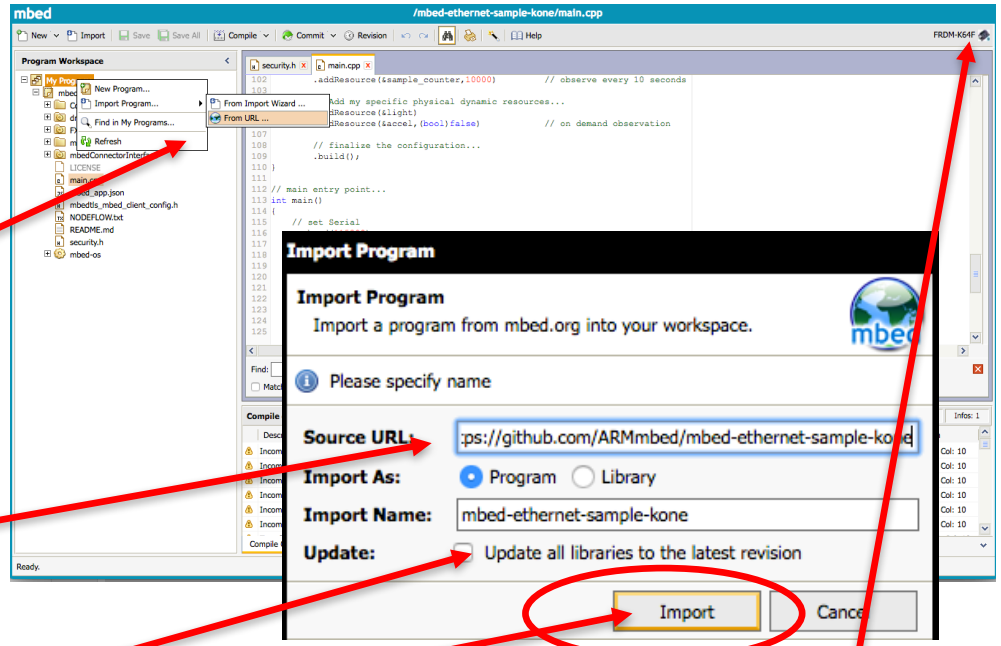


©ARM 2017
ARM

Import our K64F Endpoint Project

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

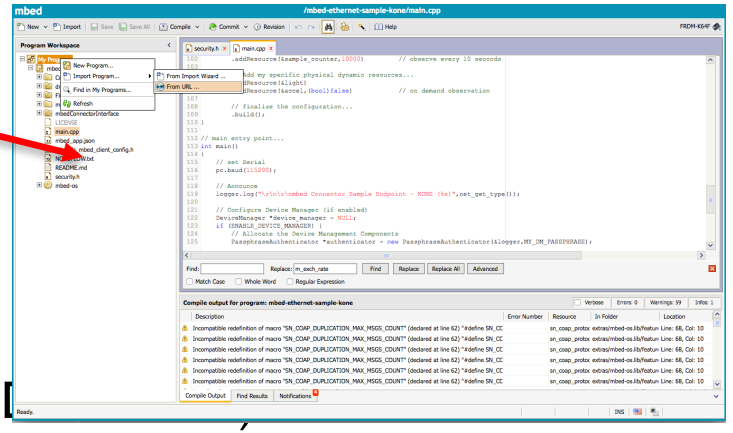
  https://github.com/ARMmbed/mbed-cloud-sample/
- Press "Import", the ensure that "FRDM-K64F" is selected



©ARM 2017

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…

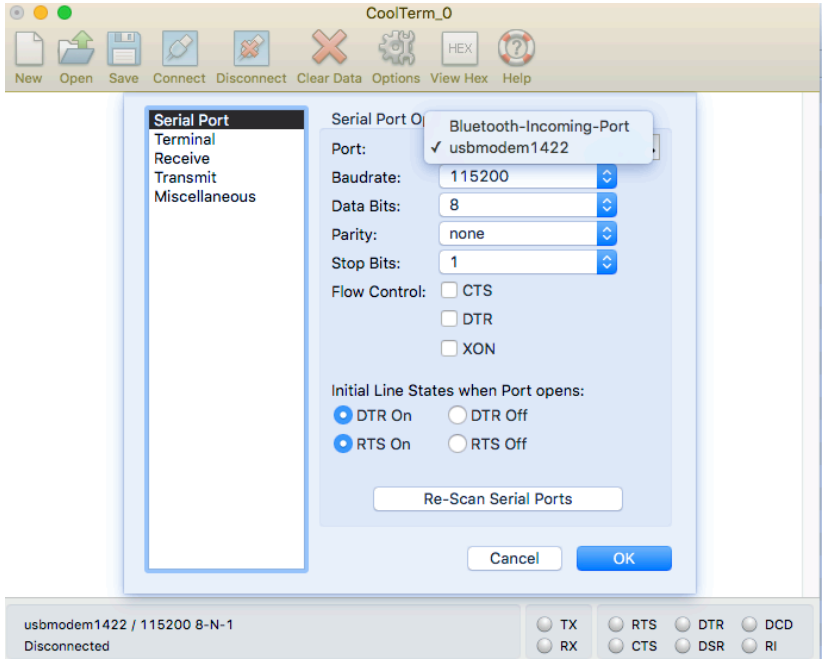


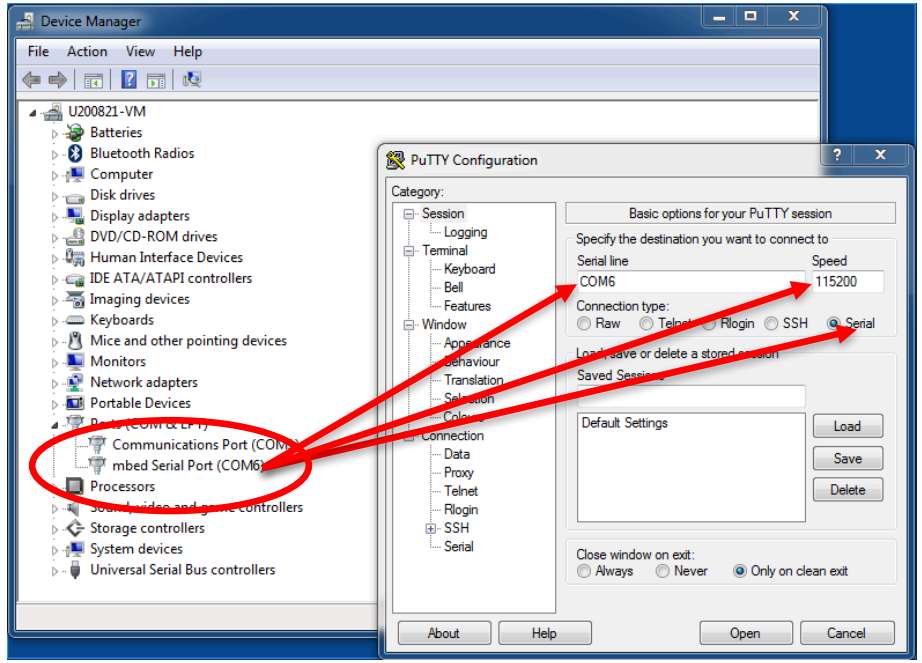
©ARM 2017

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…





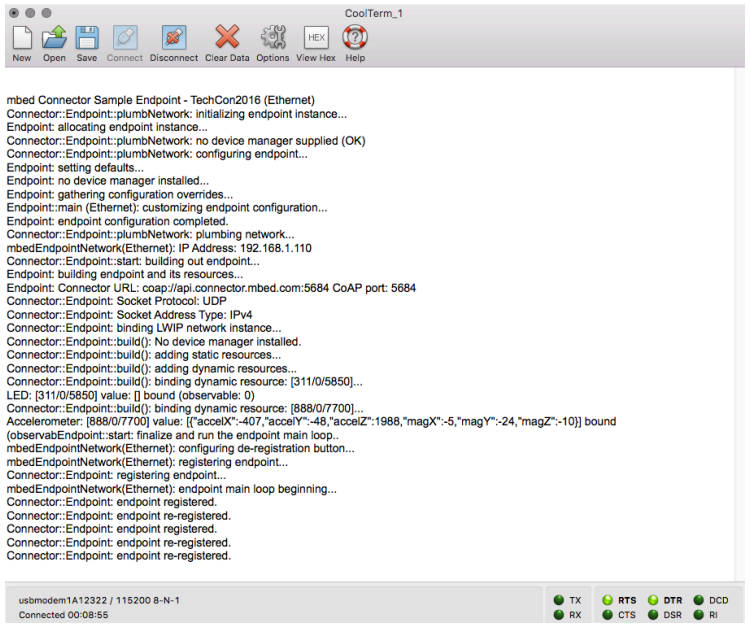
©ARM 2017

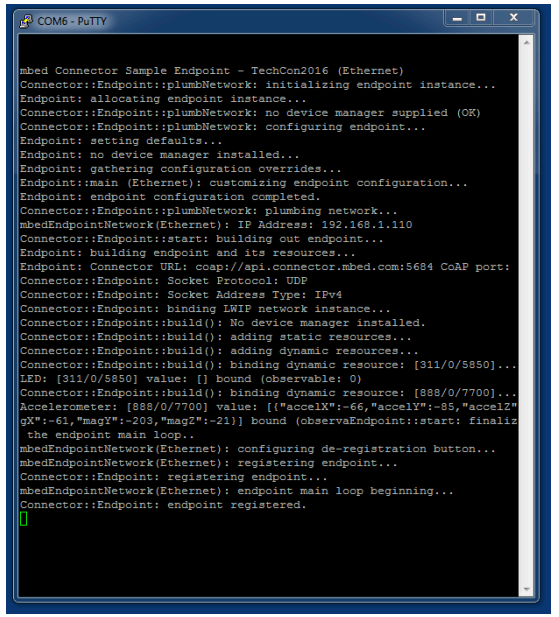
Running your Endpoint Code…

- 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:





©ARM 2017

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 Google Prototype mbed Cloud Bridge...

©ARM 2017

ARM Google mbed Cloud Bridge Setup

©ARM 2017

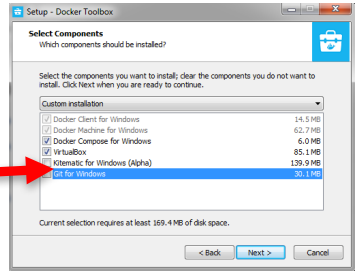
What we will do next…

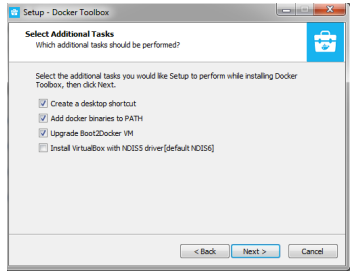
- Ensure that we have the necessary pre-requisites setup on our PC
- Create our mbed Cloud API Key/Token
- Create our Google Service Account & Authentication JSON
- Enable the Pub/Sub API
- Import our Prototype mbed Cloud Bridge instance as a Docker image into our Docker runtime
- Configure our Prototype mbed Cloud Bridge for Google Cloud

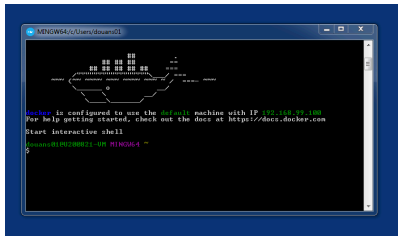
©ARM 2017

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 2017
ARM

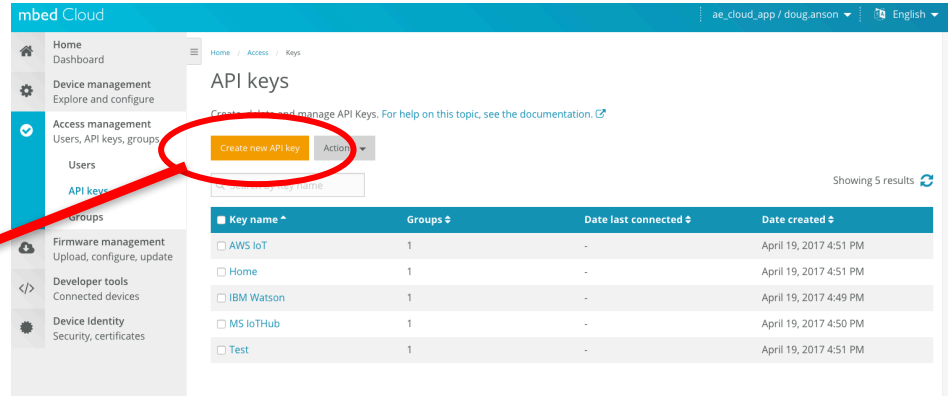
Double check (Mac/Linux) : All our needed tools are installed

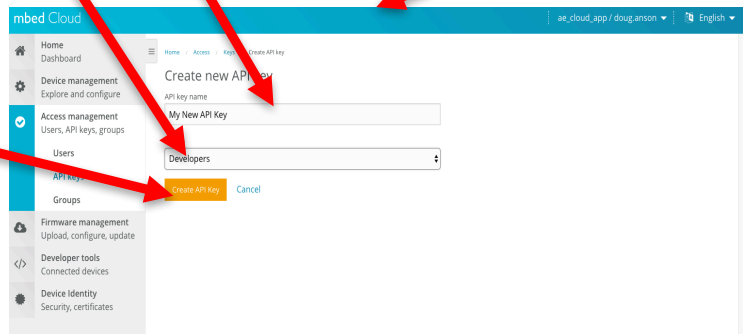
- Docker Engine installed (do not install the "Toolbox" version on Mac... use the native engine)
  - Docker Engine for Mac (do not install the "Toolbox" version on the mac):
    - https://docs.docker.com/engine/installation/mac/
  - Docker Engine for Linux:
    - https://docs.docker.com/engine/installation/linux/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

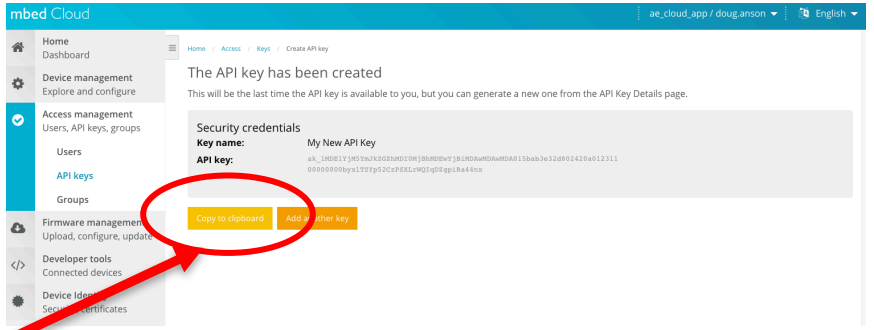
©ARM 2017

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



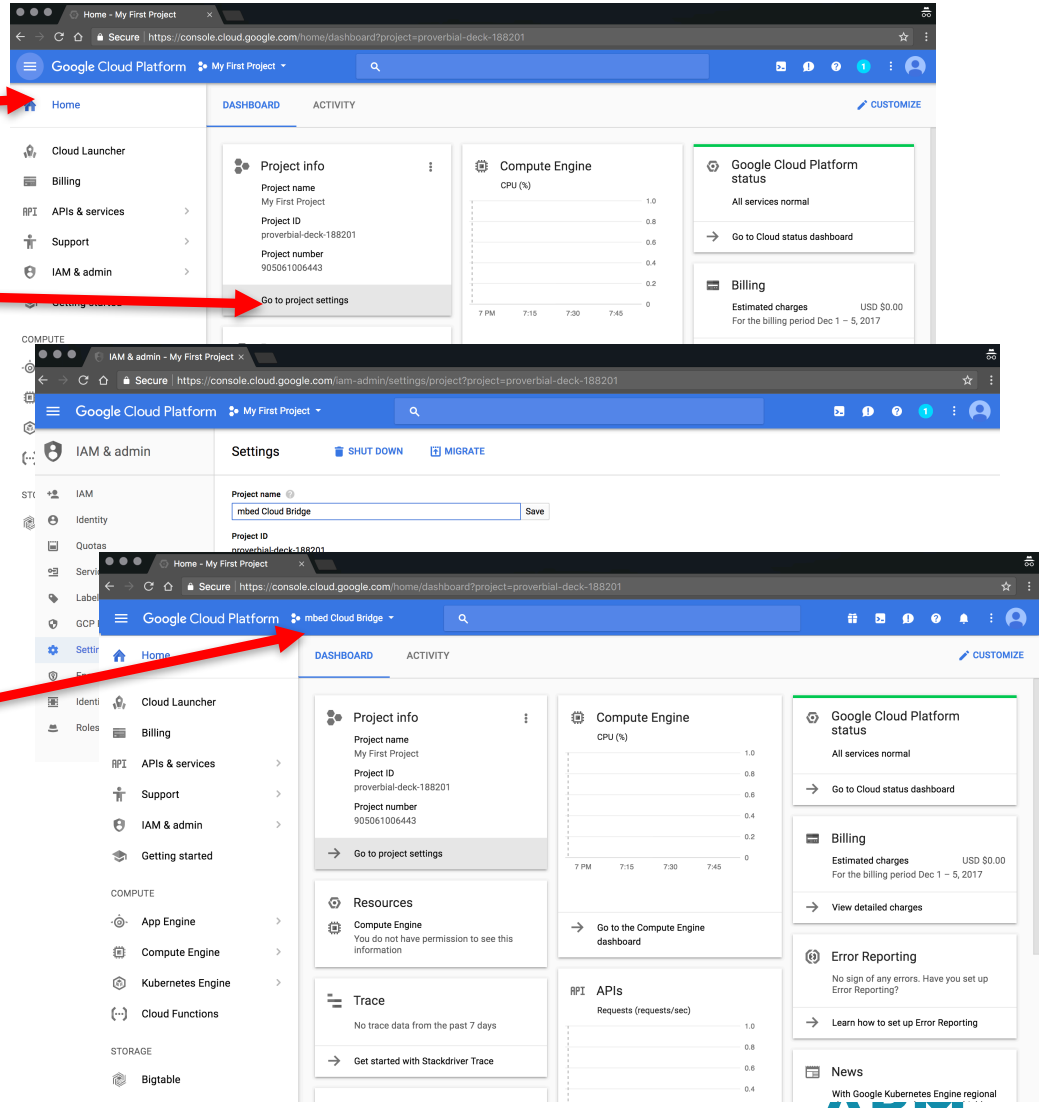




©ARM 2017

Create/Name our Google Cloud Default Project

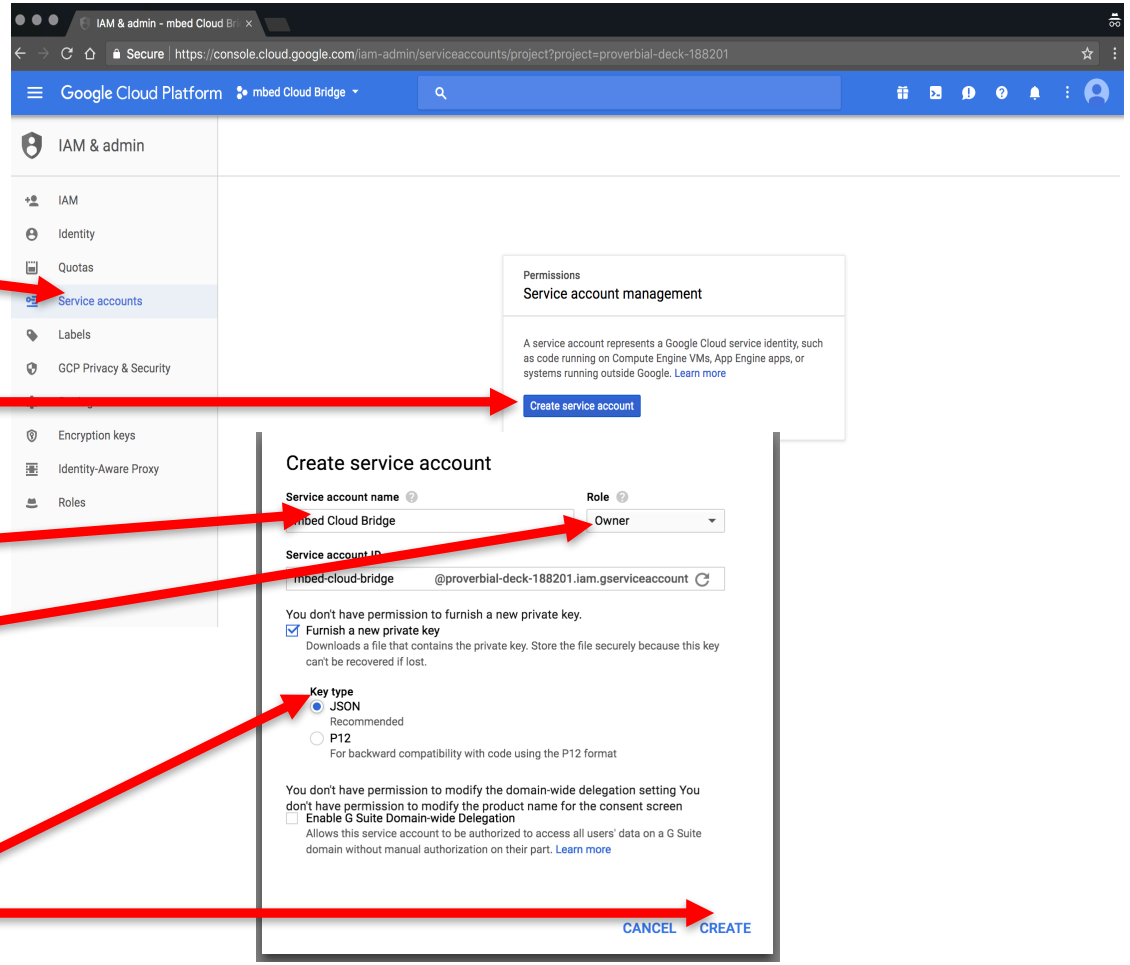
- Go to your Google Cloud Console
- Select "Project Settings"
- New Project name: "mbed Cloud Bridge"
  - Press "Save"
  - Refresh your browser!
- Confirm that the default project name has changed



©ARM 2017

Create a Service Account

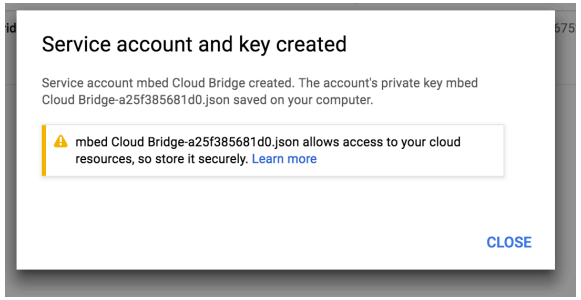
- Go to your Google Cloud Dashboard
- Select "Service Account"
- Press "Create a Service Account"
- Name the service account
- Select "Owner"
- Confirm "JSON" selected, press Create



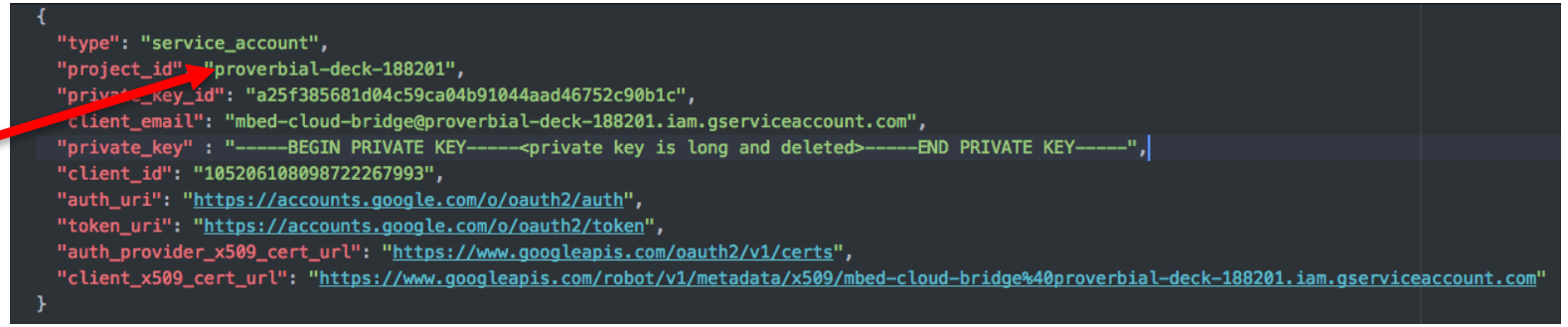
©ARM 2017

Create a Service Account... "project_id" value

- A JSON file will be downloaded as a result of creating the service account



- View the JSON file
- Save the "project_id" value

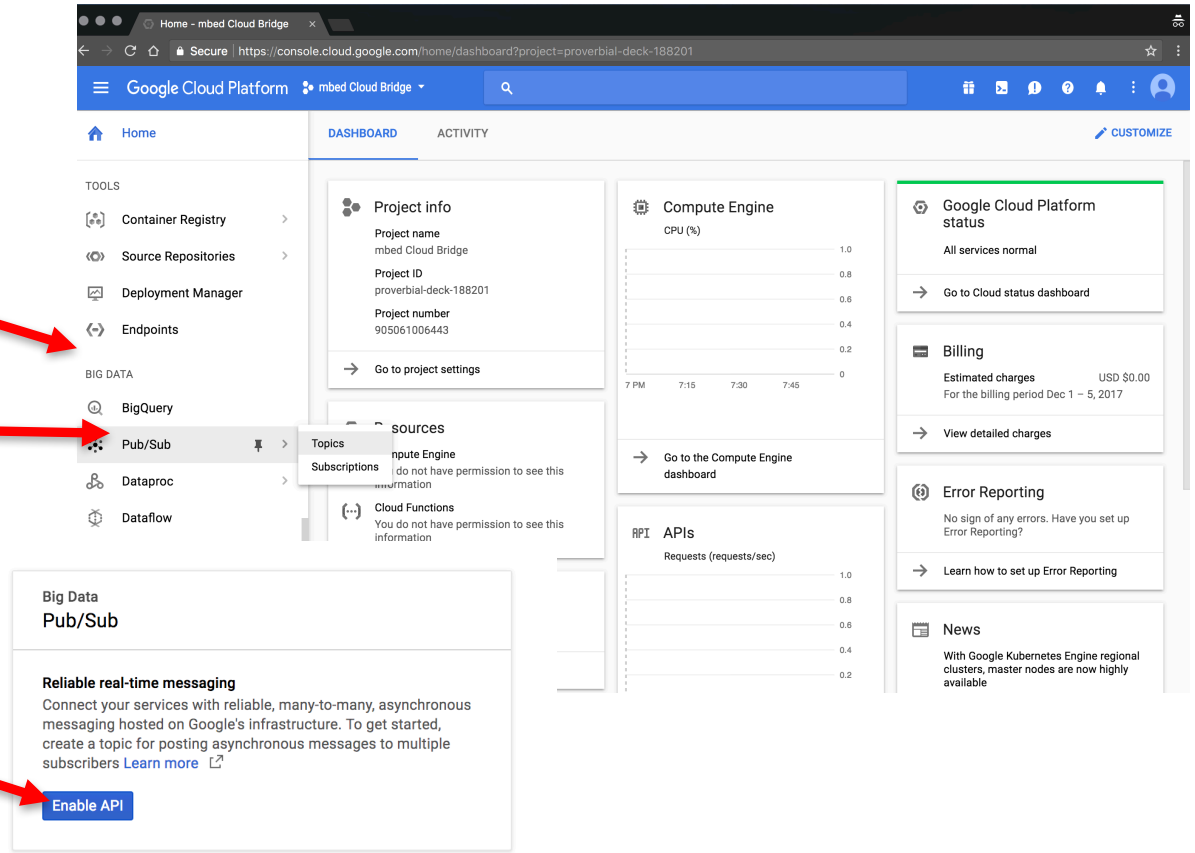


- The "project_id" value is the "google_cloud_app_name" value for the bridge config
  - Save it off for now... we'll use it later.

©ARM 2017
ARM

Create a Pub/Sub Service Instance

- Go to the Google Cloud Dashboard
- Scroll down to the "Big Data" section
- Select "Pub/Sub"
- Press "Enable API"

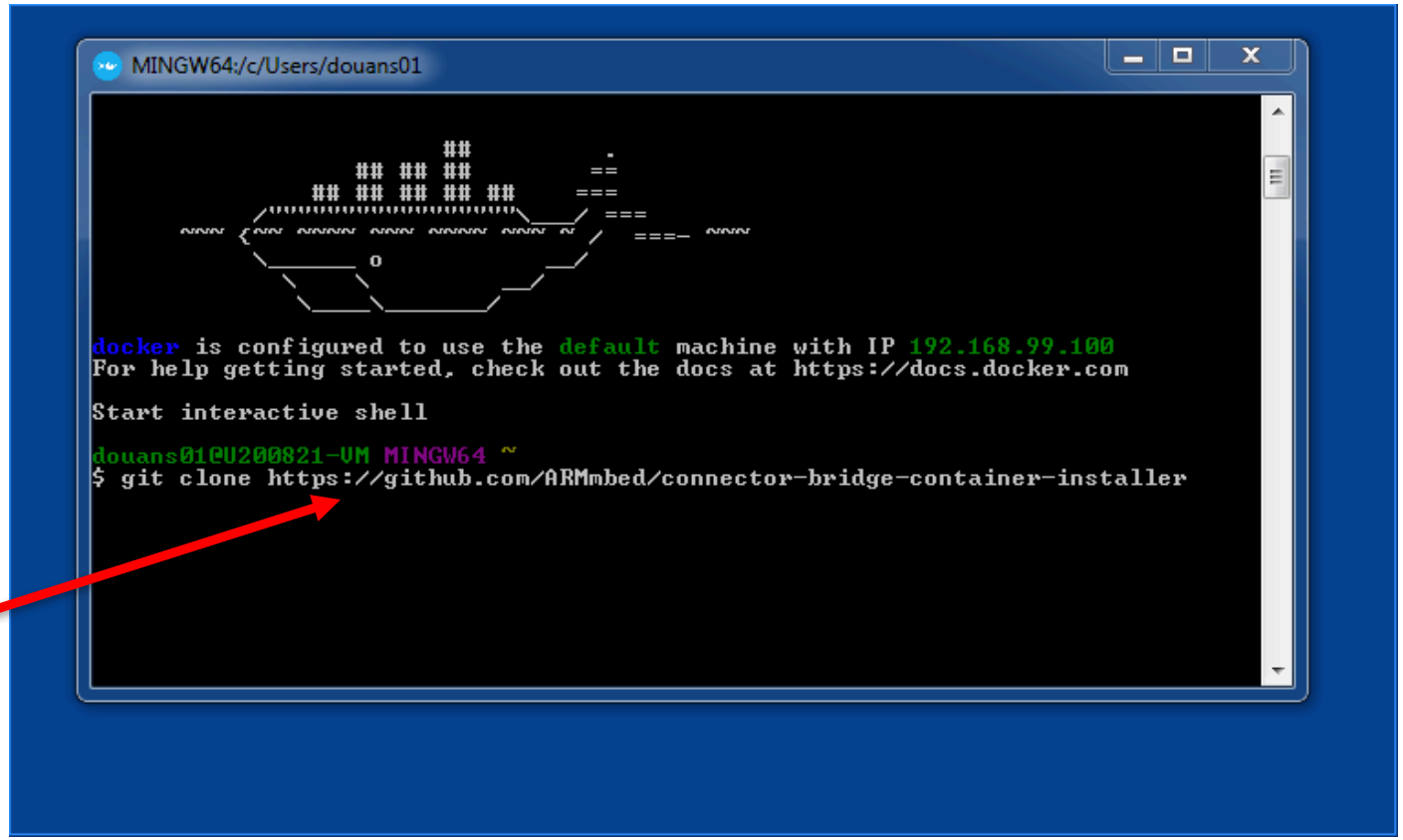


Once created, our Google Cloud Environment is ready for our bridge!

©ARM 2017

Import our mbed Cloud Google 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)

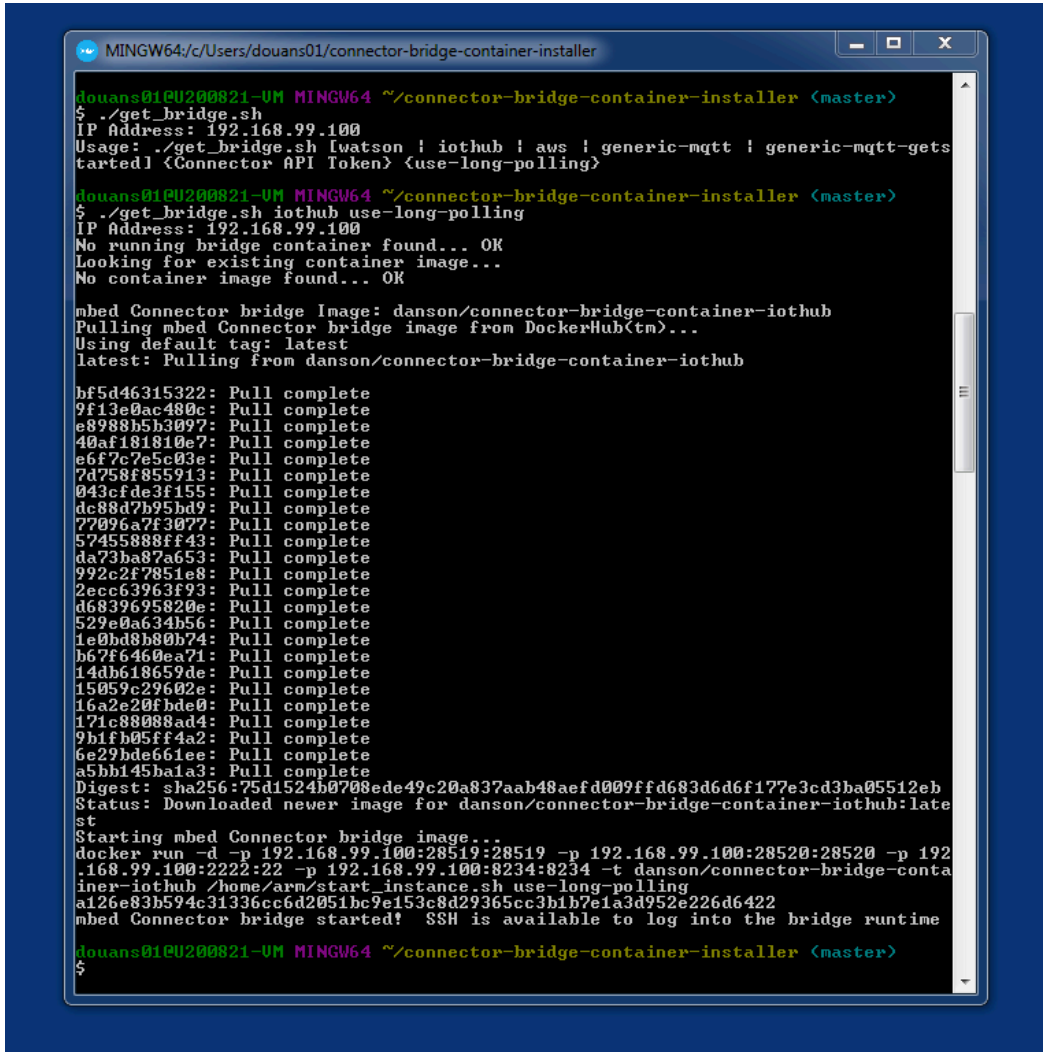


- For all platforms (Windows/Mac/Linux), this is the "git" command to run:
  $ git clone https://github.com/ARMmbed/connector-bridge-container-installer

©ARM 2017
ARM

Import our mbed Cloud Google 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 google use-long-polling
- Bridge container will import from DockerHub



©ARM 2017
ARM

Configure our Google Bridge (Windows)

- Your Container IP address will be:

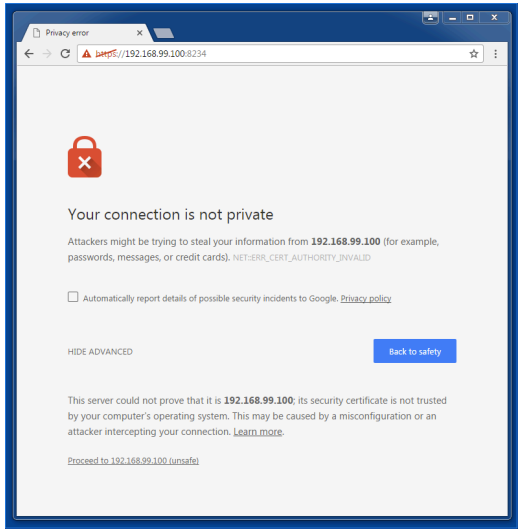
  192.168.99.100

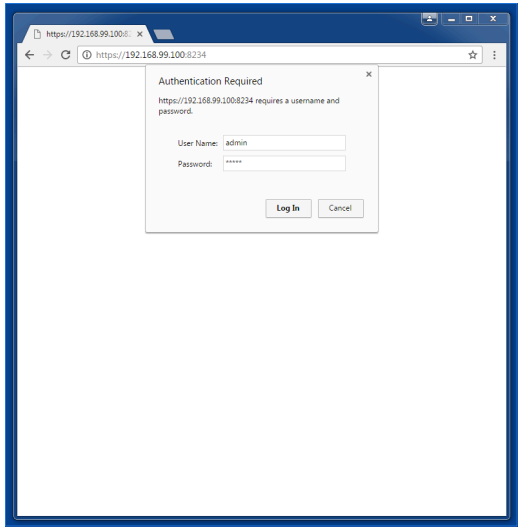
- Bring up Firefox/Chrome and go to:

  https://192.168.99.100:8234

- Accept the self signed certificate

- Username: admin PW: admin





©ARM 2017

Configure our Google Bridge (Mac/Linux)

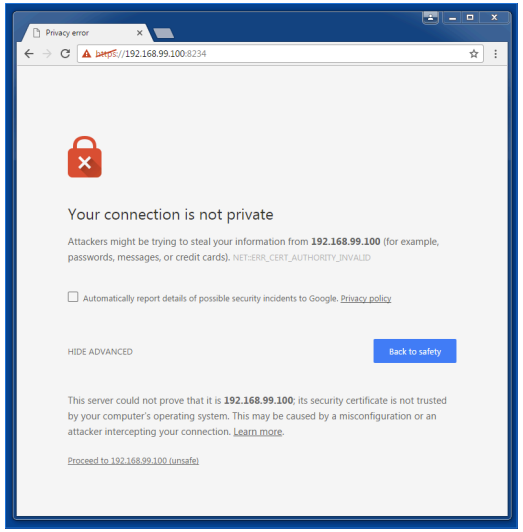
- Your Container IP address be ”localhost” if installed on your local box, otherwise, make note of it if installed remotely

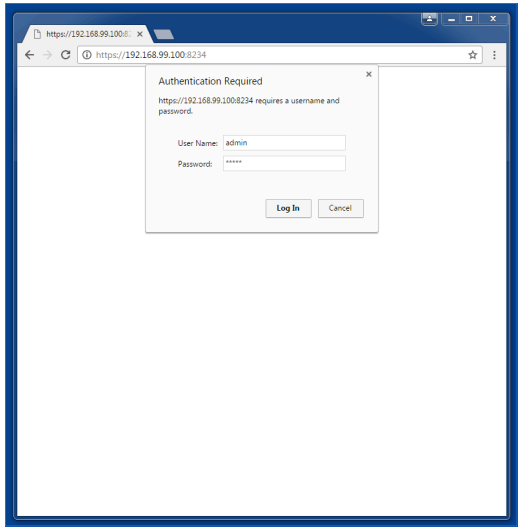
- Bring up Firefox/Chrome and go to:

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

- Accept the self signed certificate

- Username: admin PW: admin

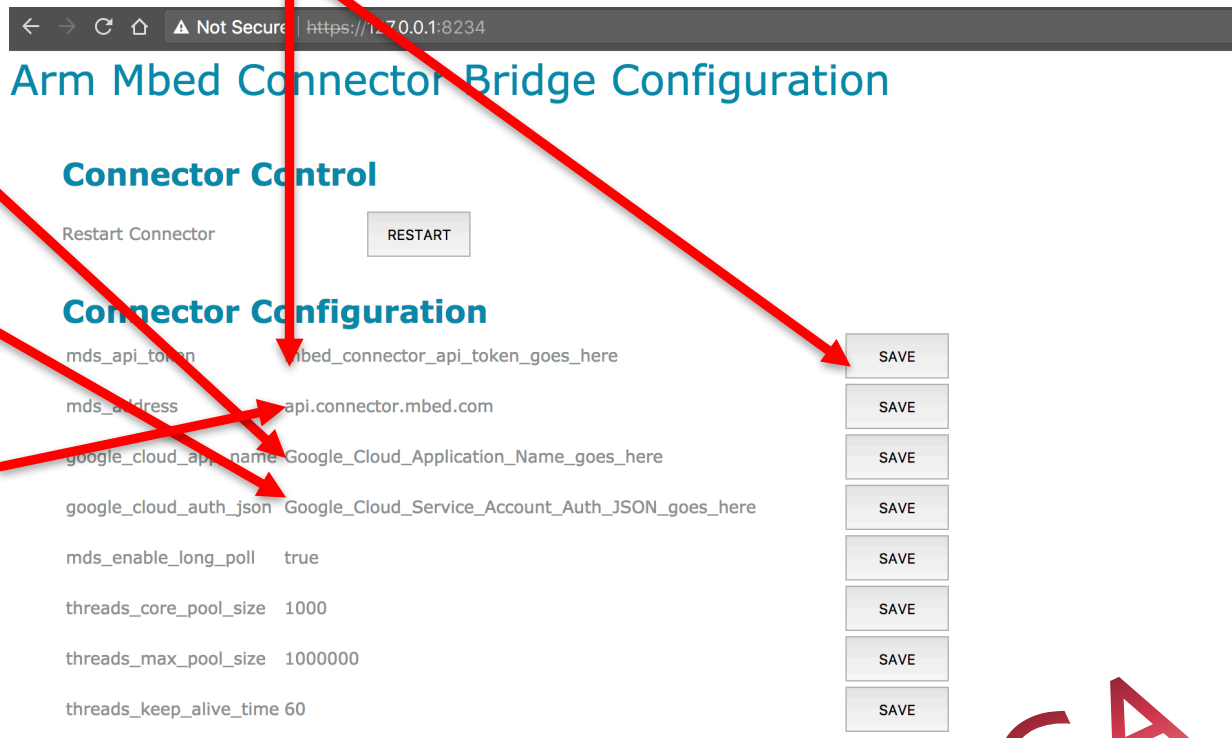




©ARM 2017

Configure the mbed Cloud Bridge for Google…

- Enter the mbed Cloud API Key/Token, *THEN* Press "Save"
- Enter your "project_id" value, Press "Save"
- Enter your Google Auth JSON, Press "Save"
- Ensure you have the right mbed API endpoint
  - Cloud: api.us-east-1.mbedcloud.com
  - Connector: api.connector.mbed.com
  - If changed… be sure to press "Save"
- Press "RESTART"
- Your mbed Cloud Bridge is now configured for Google Cloud



Update for GA

©ARM 2017
ARM

Status Check

So far we've completed the following

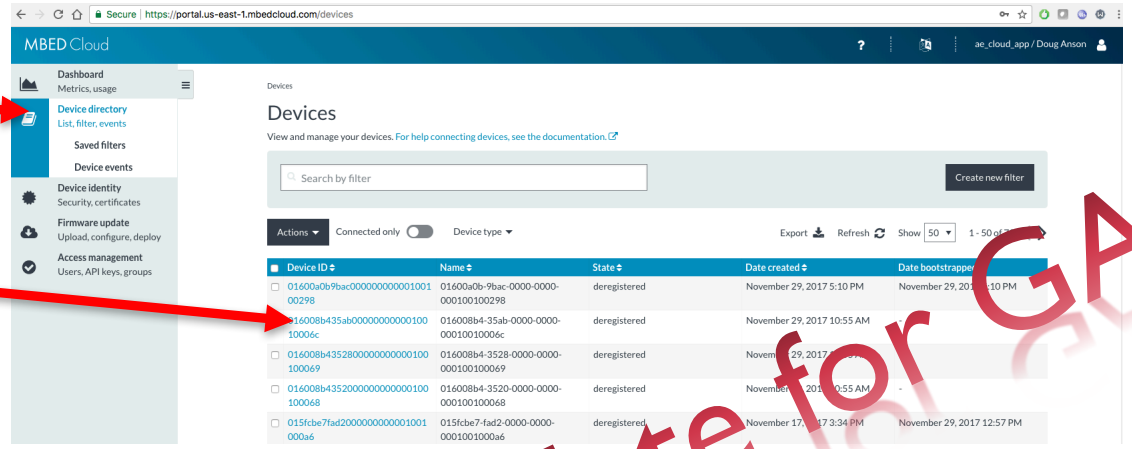
- Ensure that we have the necessary pre-requisites setup on our PC
- Create our mbed Cloud API Key/Token
- Create our Google Service Account & Authentication JSON
- Enable the Pub/Sub API
- Import our bridge instance: a Docker image into our Docker runtime
- Configure our mbed Cloud Bridge for Google

With our bridge now operational, we should be able to restart our endpoint and see messages coming into our Google via a java-based "Exerciser" Application

©ARM 2017

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 the mbed Cloud Portal dashboard: https://portal.us-east-1.mbed.cloud.com
- Look for your endpoint in the list
- Record your endpoint's Device ID
  You'll need this in the next steps...



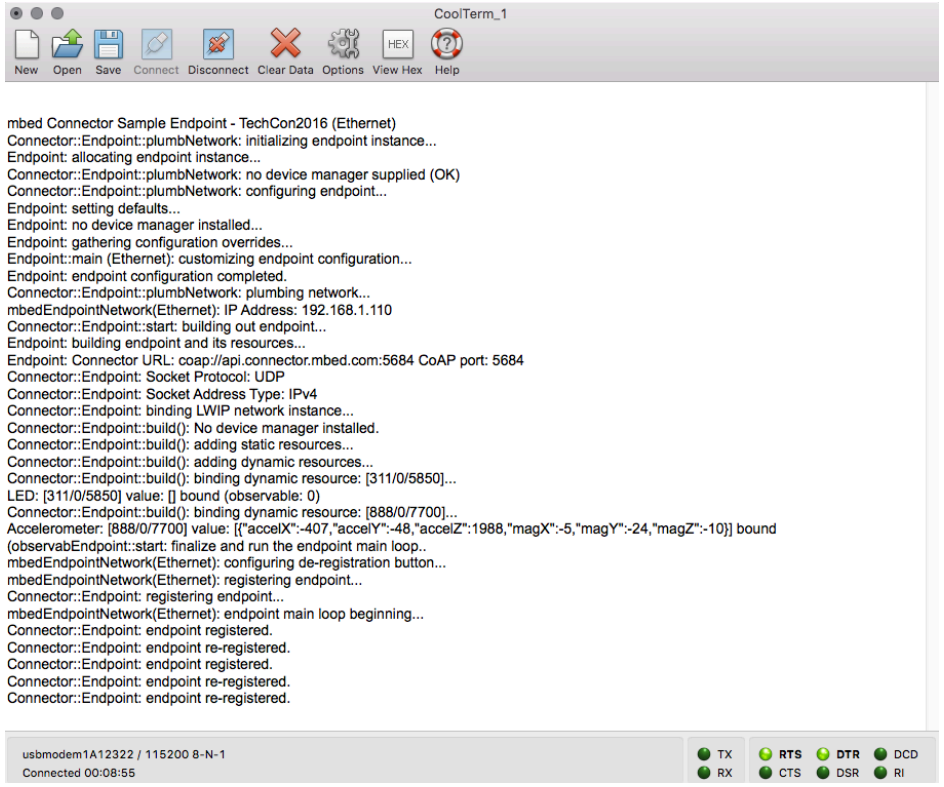
Update for GA

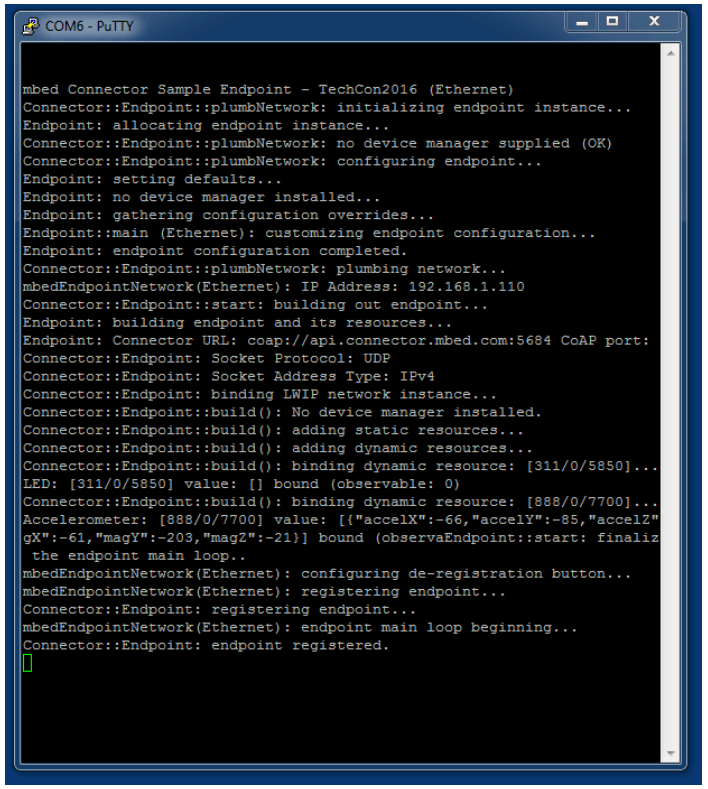
Lets check how things are working…

©ARM 2017
ARM

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"

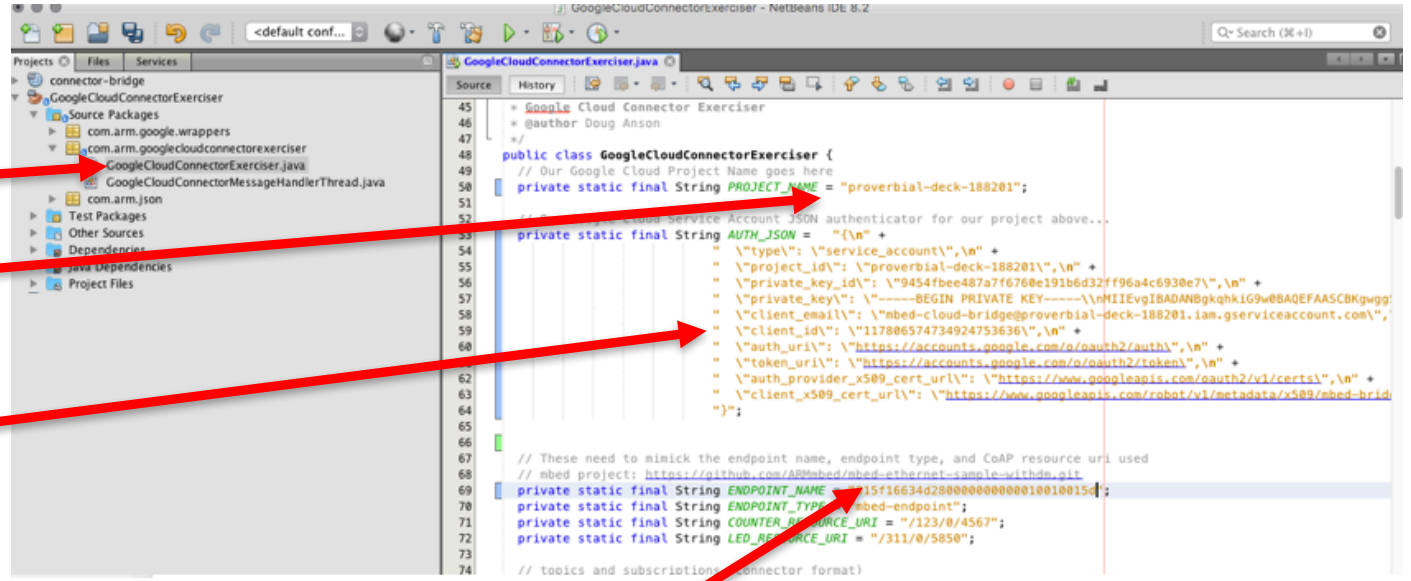




©ARM 2017

Install the Java Exerciser Application

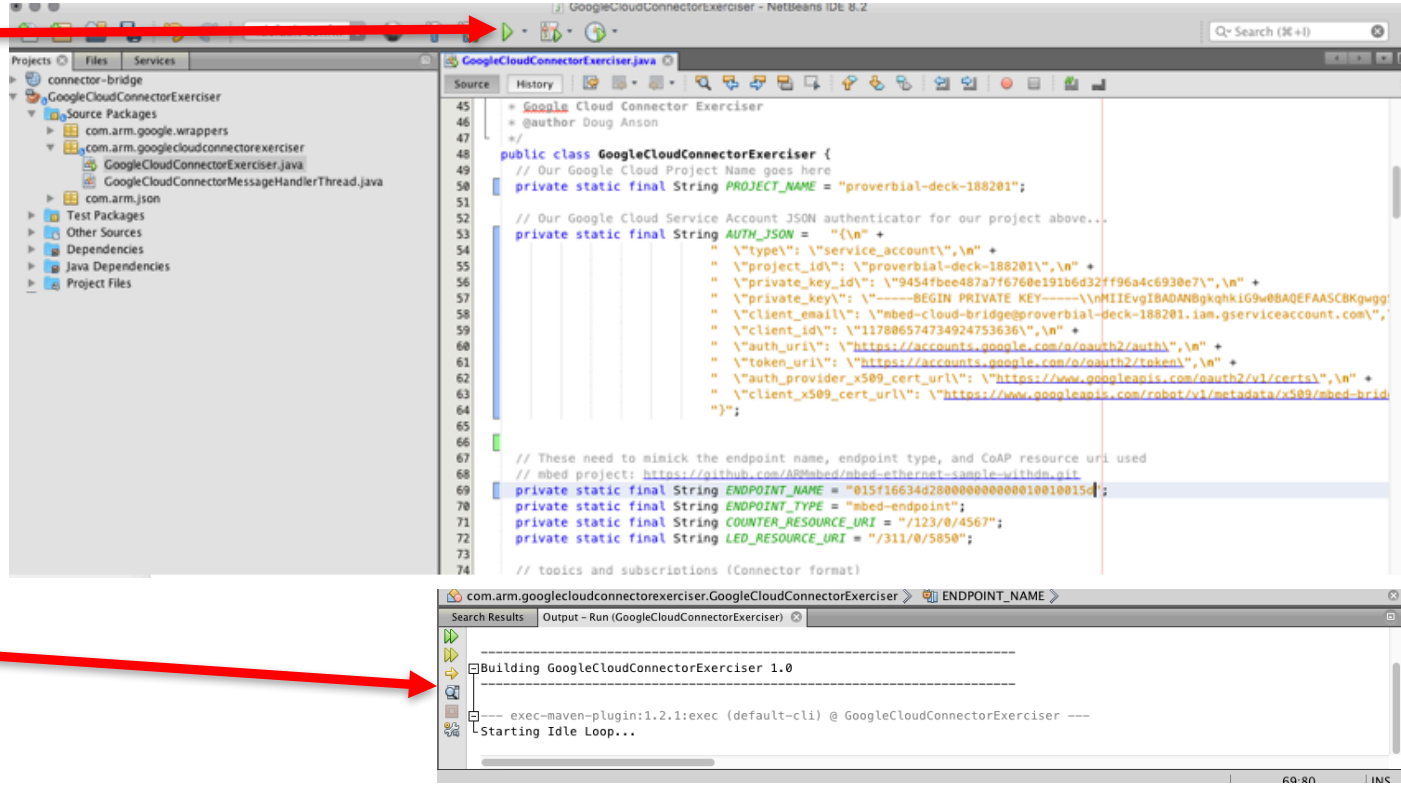
- Install Java and the NetBeans IDE http://www.netbeans.org
- Clone the following GitHub repo: https://github.com/ARMmbed/GoogleCloudConnectorExerciser
- Open the project in NetBeans
  Select this source file
- Enter "project_id" value
- Enter your JSON string
  (note "string" formatting!)
- Enter your mbed Cloud Endpoint Device ID you saved!



©ARM 2017
ARM

Running the Java Exerciser Application

- Press the "Run" button
- The project will compile
- If the compile succeeds without error, the project will run...
- You should see output down here
- The Exerciser will turn the LED resource on and off depending on whether the monotonic resource observation value is even or odd... this demonstrates the bi-directional nature of the bridge with your endpoint! CONGRATS!! You are finished!



©ARM 2017
ARM

Summary

We have completed the following – congratulations!

- Created our Google Cloud and mbed environments and PC tool setup
- Imported our mbed endpoint, customized, installed, and ran it
- Created our own Google Pub/Sub instance
- Imported and configured our ARM Google Prototype mbed Cloud Bridge
- Examined live telemetry with the java-based ”Exerciser” application
- Interacted with our device through the java-based “Exerciser” application

Great JOB! Thanks for your time.

©ARM 2017