Workshop

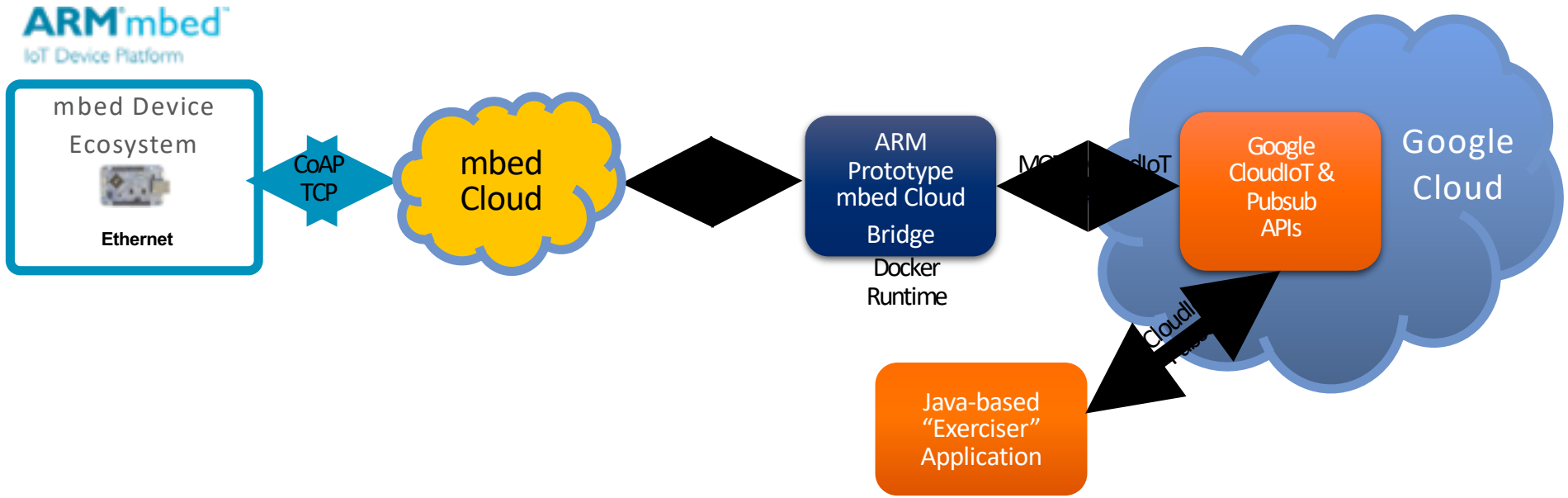
Connecting IoT devices to the cloud with Google CloudIoT & mbed Cloud

Doug Anson

v2.4 (GCR Edition)

April 3, 2018

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 CloudIoT & Pubsub APIs

arm

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/bridge_workshop_google_cloud

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

arm

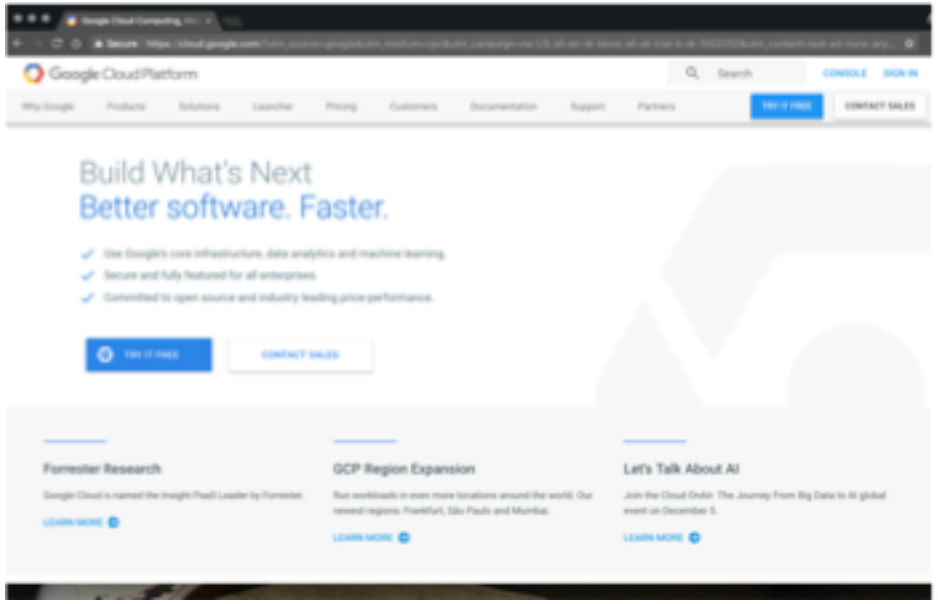
Create our Google Cloud Account

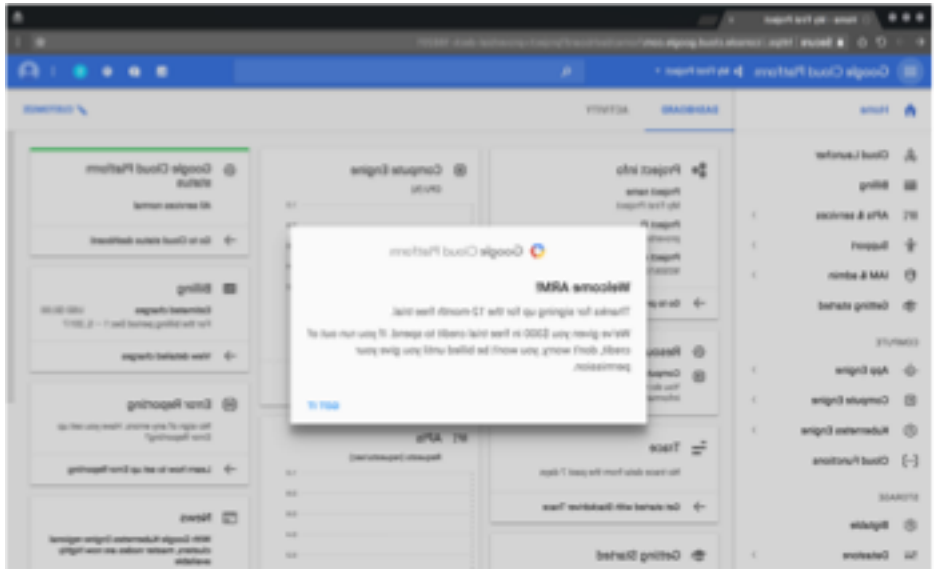
Navigate to the Google Console:
https://cloud.google.com

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

NXP- FRDM-K64F Overview

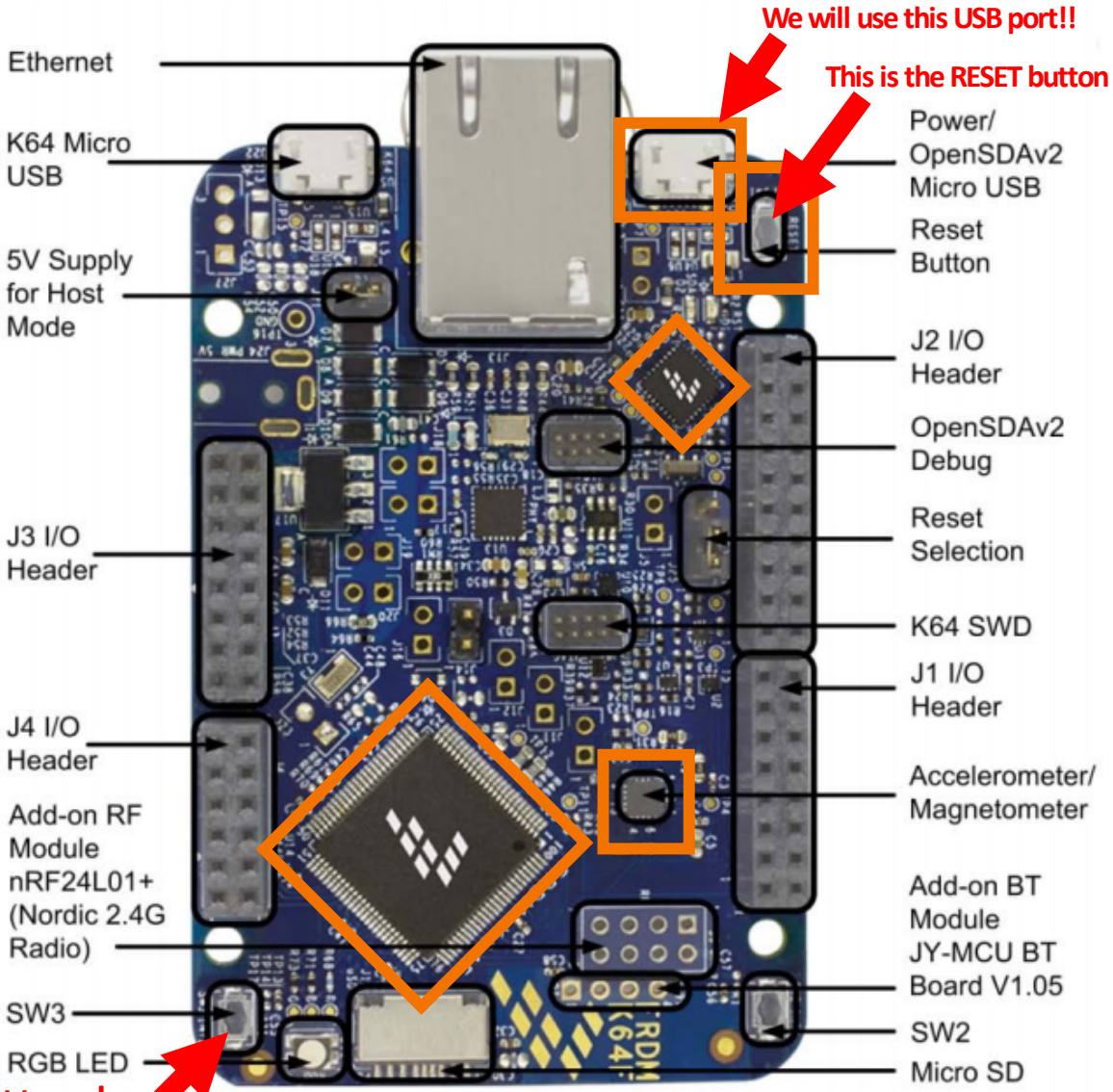
Freedom Development Platform - Quick, simple development experience with rich features

- 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

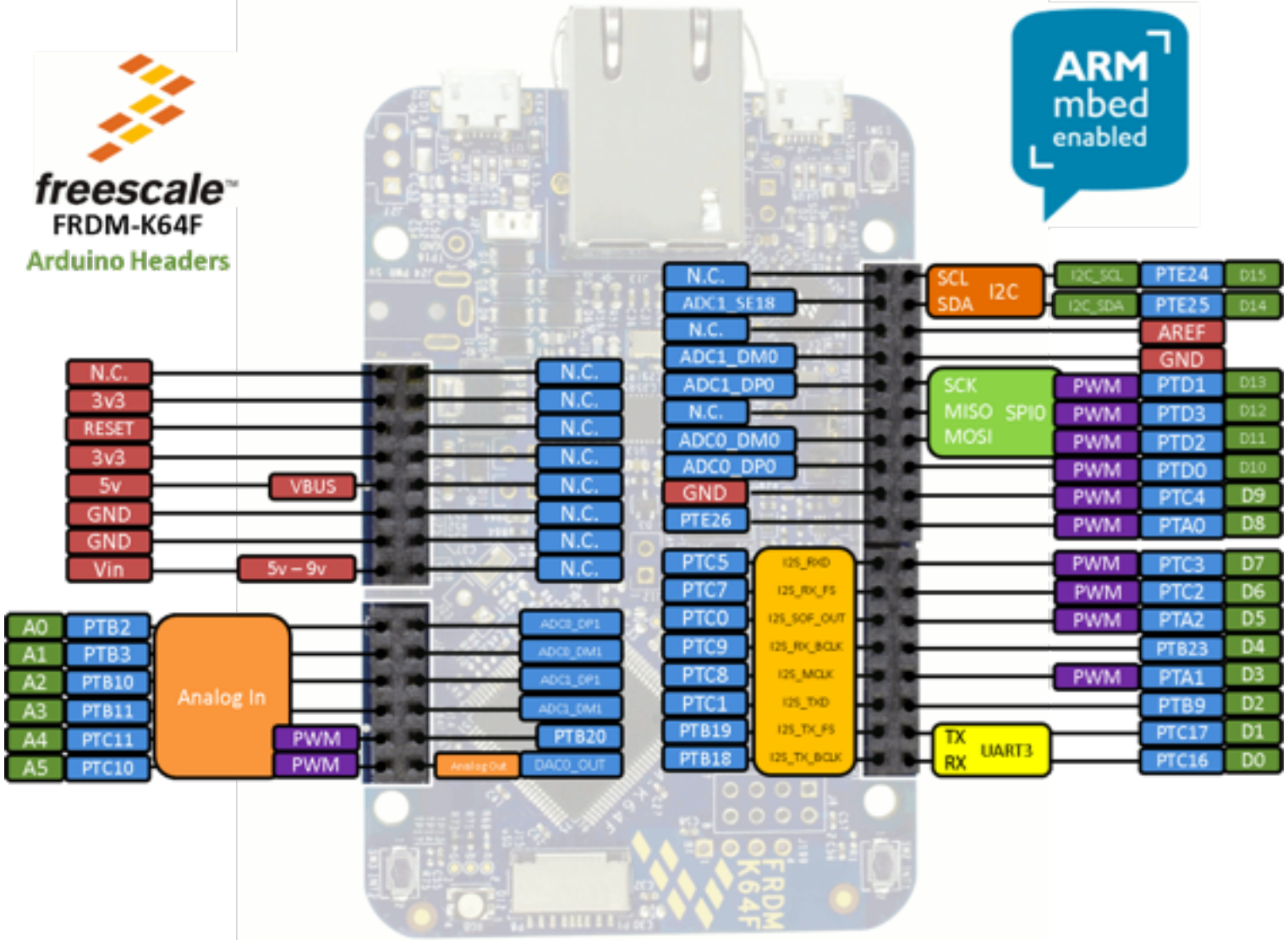


De-Registration Button!

(SW3)

arm

freescale™
FRDM-K64F
Arduino Headers



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*
- 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

Prior to Workshop

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

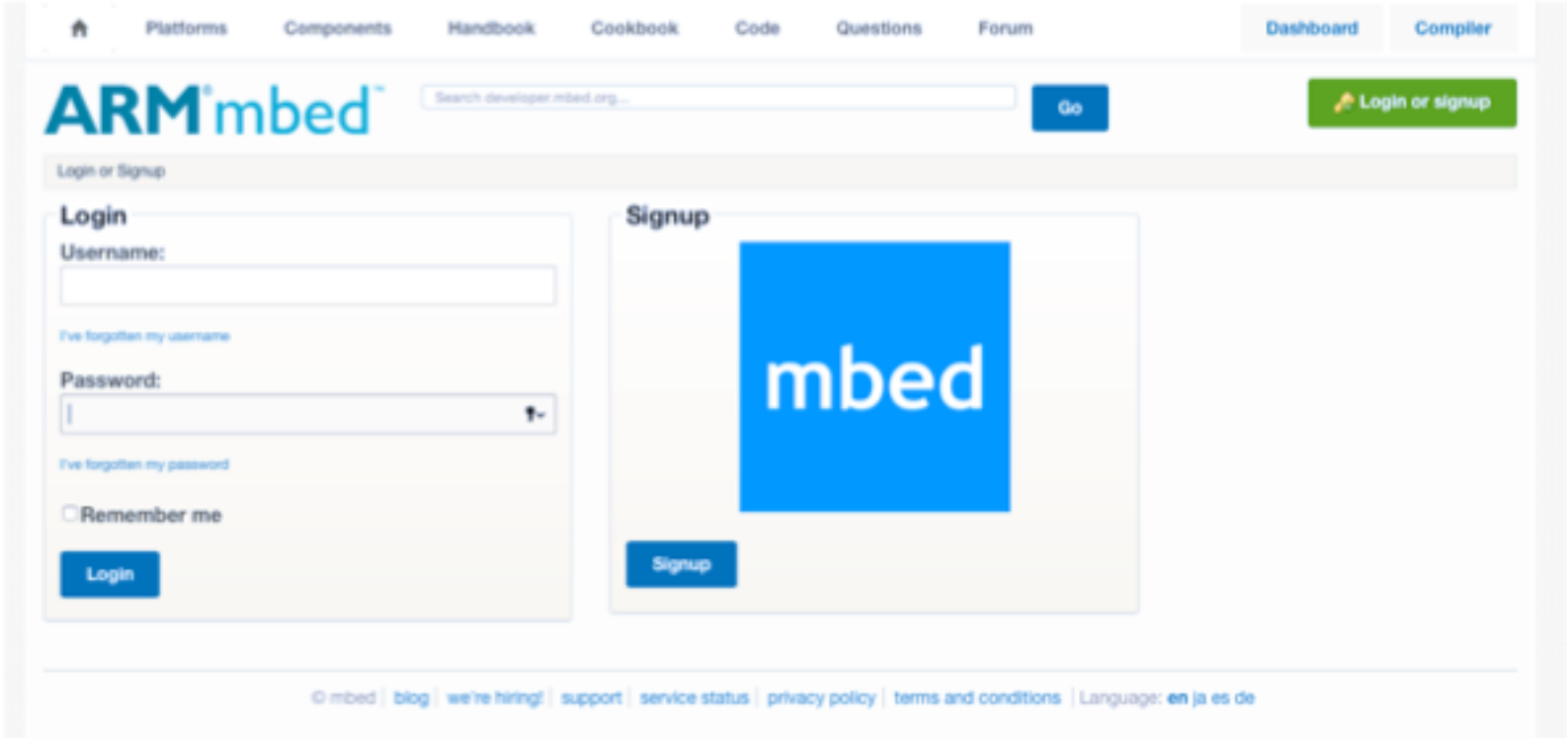
arm

Create Your mbed Account

Prior to Workshop

Navigate to: https://os.mbed.com/

Create an Account

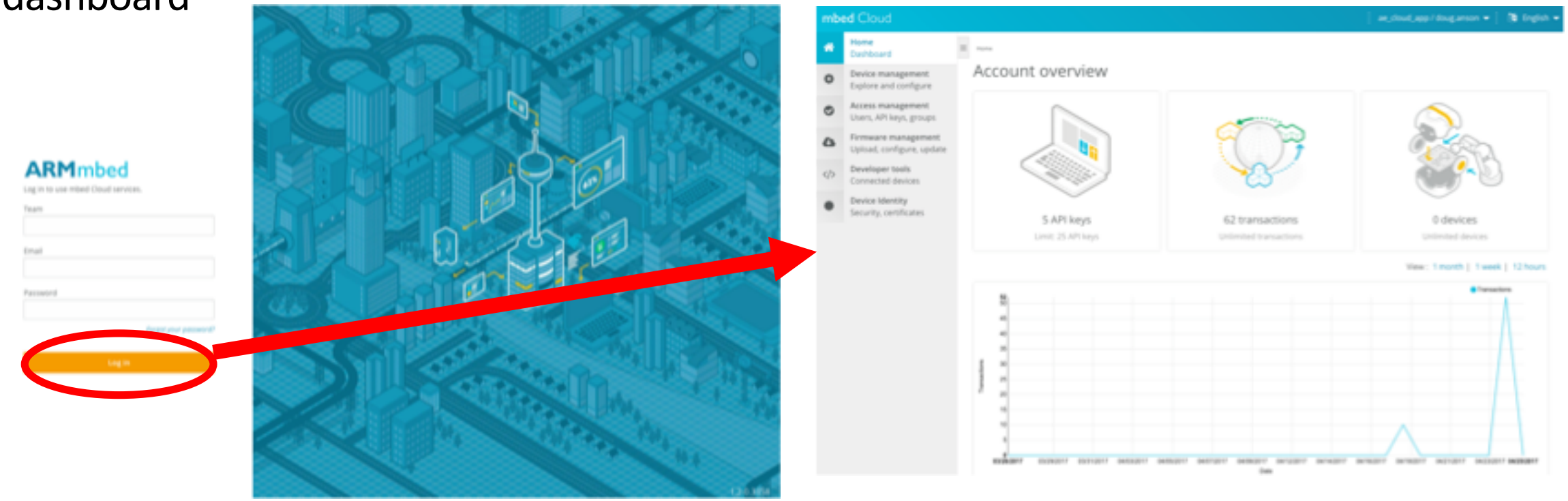


Create Your mbed Cloud Account…

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

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”

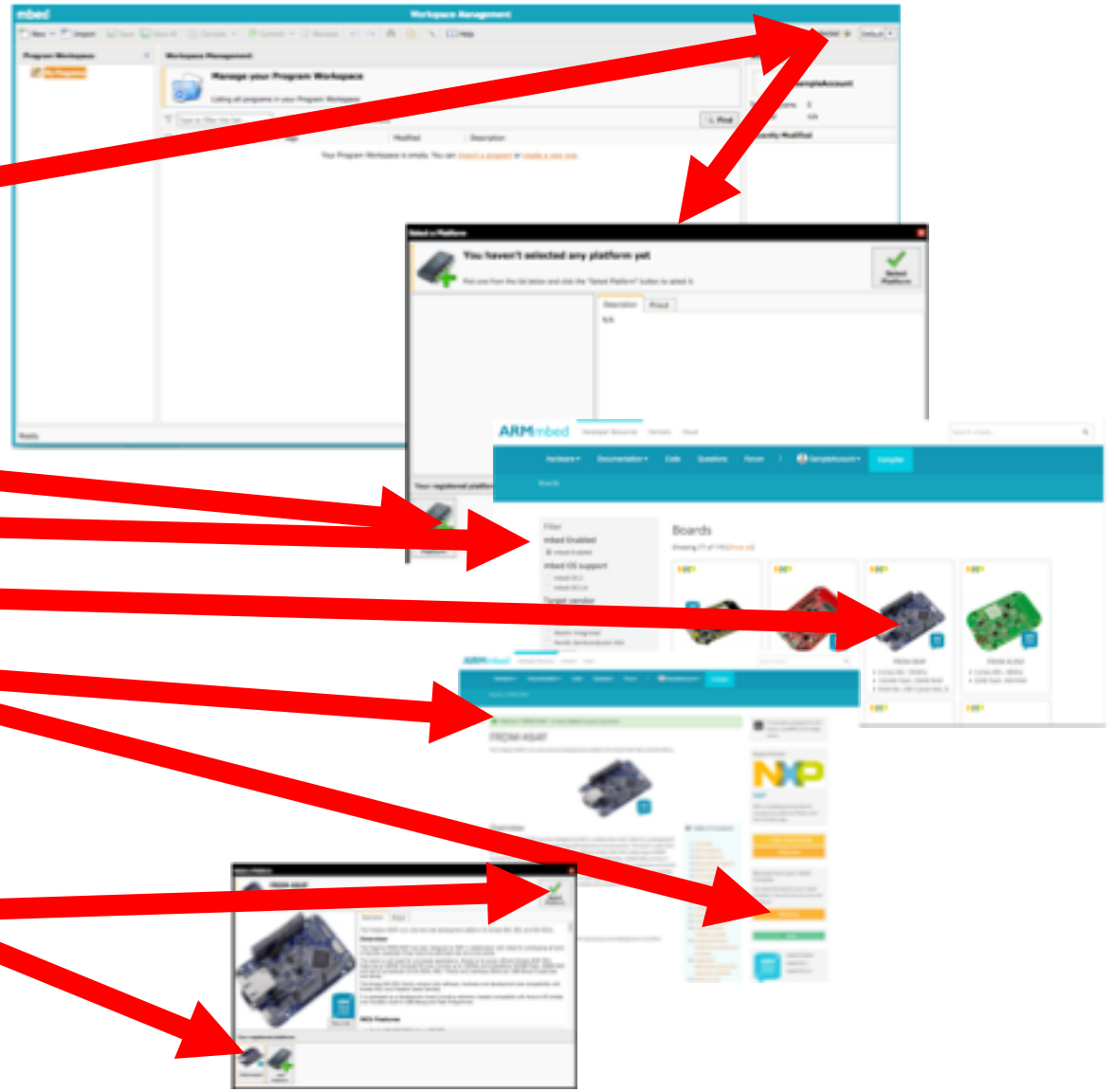
Click on “Add to Compiler”… note addition.

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



Import our K64F Endpoint Project

Go to https://developer.mbed.org

Select the “Compiler” page

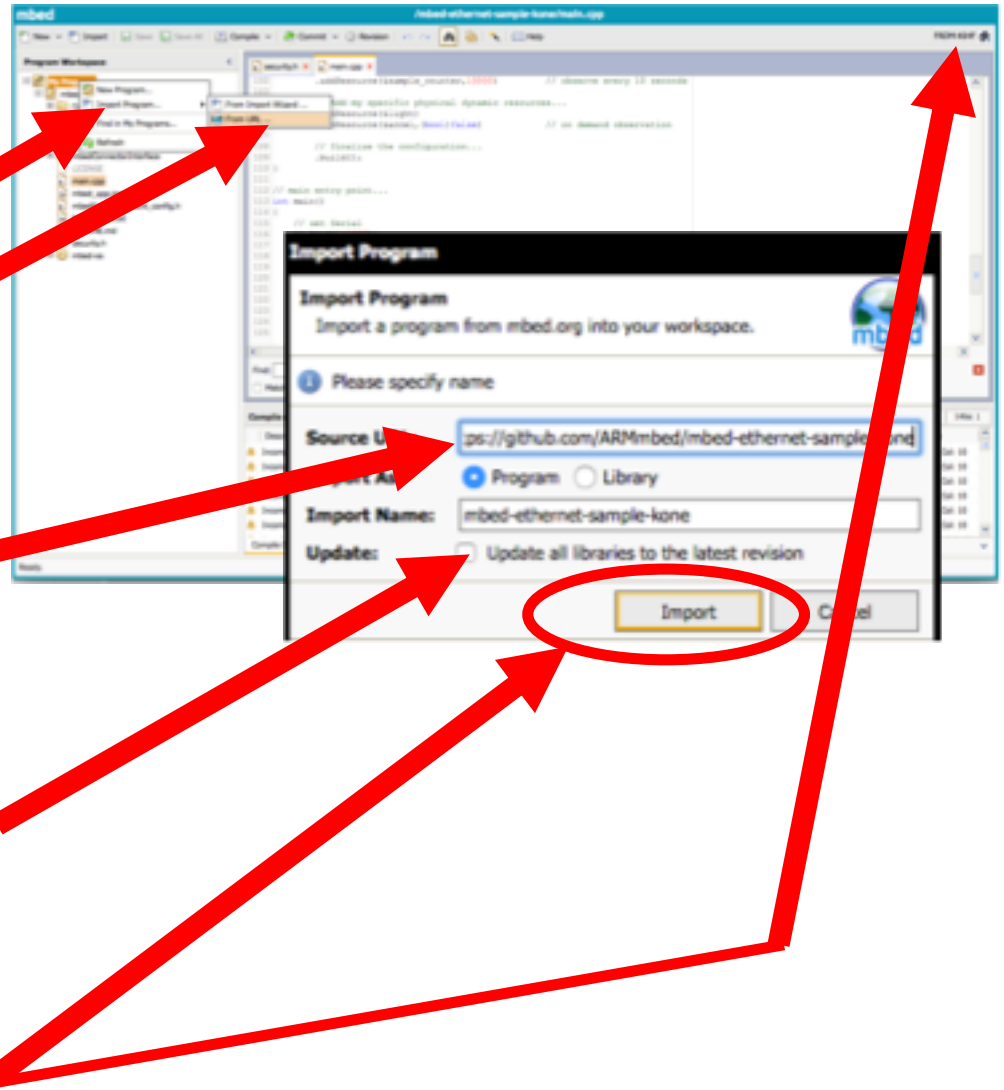
Right-click on “My Programs”→”Import Program”

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



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 three CoAP resources: accelerometer , monotonic counter, and LED

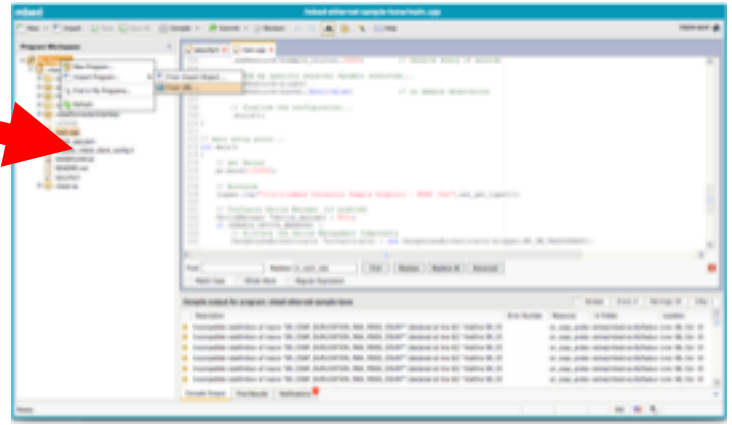
Select the project name and press the “Compile” button

The endpoint code should compile up successfully

The online IDE will deposit a “bin” file into your downloads directory

Drag-n-Drop this bin file to your “MBED” flash drive (may also be called “DAPLINK”)

K64F green LED will flicker for a bit, then stop, and dismount/remount…



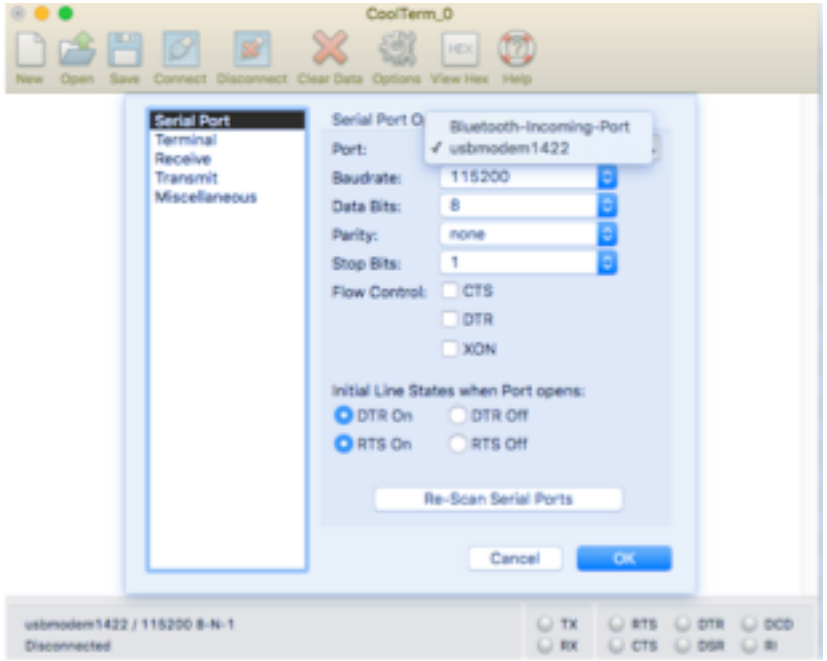
Running your Endpoint Code

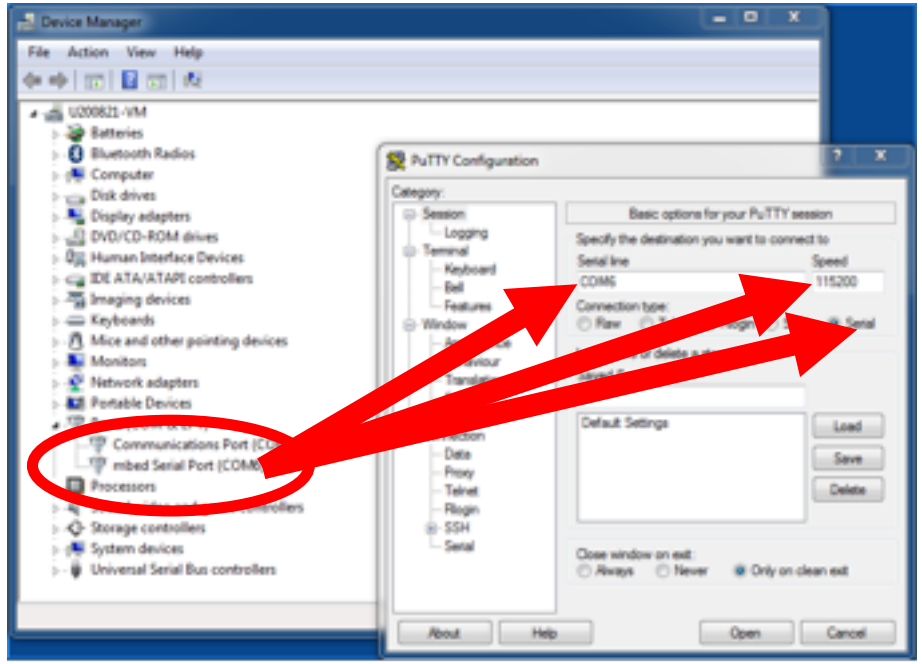
Bring up your Serial Terminal

- CoolTerm for Windows, Mac, Linux: Select: “Options→”Re-scan serial ports”. Select the mbed one found.
  - For MAC, you may need to “authorize” the CoolTerm Application under your “System Preferences”-> “Security & Privacy”… you may have to allow apps to run from “any developer”, then authorize the launch of CoolTerm.
- PuTTY for Windows : You must determine what COM port your mbed device is. Look in the Windows Device Manager FYI

For the endpoint serial configuration, set the baud rate to 115200 baud, defaults for everything else: (8, N, one)

- PuTTY for Windows: Ensure that you have “Serial” radio button selected…





Running your Endpoint Code…

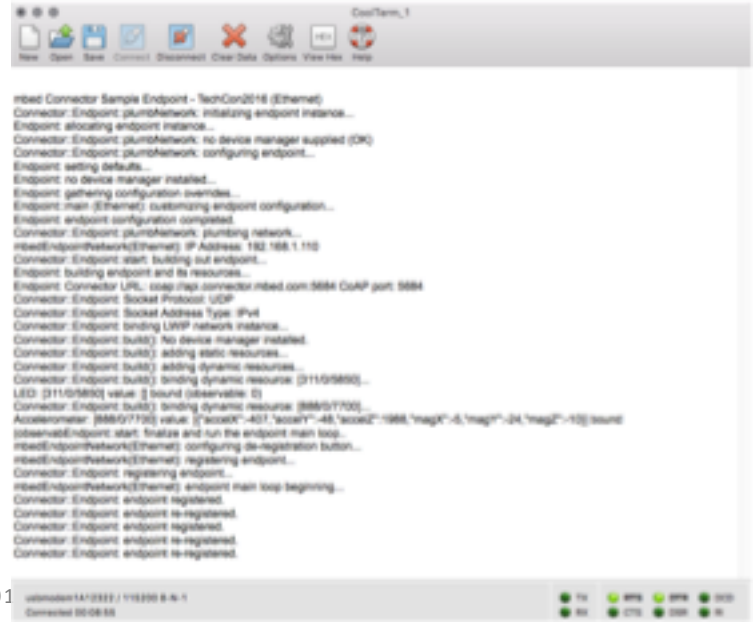
Connect your Serial Terminal to the K64F:

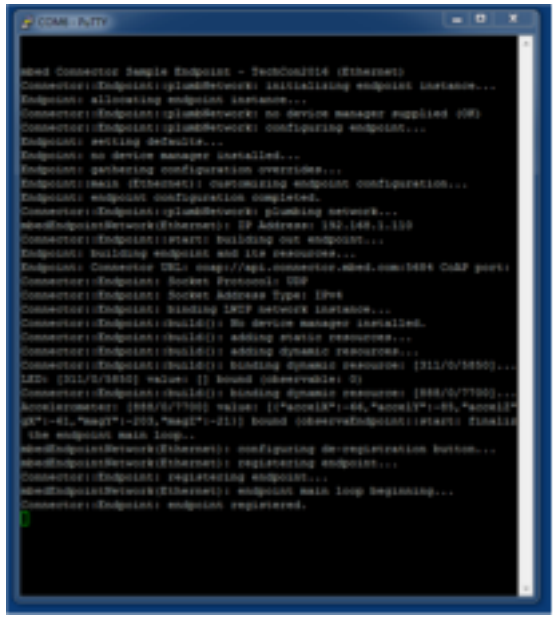
- CoolTerm for Windows, Mac, Linux: Simply press the “Connect” button on the top part of the CoolTerm GUI
- PuTTY for Windows: Press the “Open” button

Send a “Break” command from the serial terminal (or press the RESET button on the K64F)

- CoolTerm for Windows, Mac, Linux: “Connection” → “Send Break”
- PuTTY for Windows: Right-click on top of Window, Select “Special Command” → ”Break”

Look at the output – you need to confirm that you see “endpoint registered” in the output:





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 Google mbed Cloud Bridge Setup

What we will do next…

Confirm all of our needed tools are installed
Create our mbed Cloud API Key/Token
Create our Google Service Account & Authentication JSON
Enable the Google Pubsub and CloudIoT API Instances
Import our prototype mbed Cloud Bridge instance as a Docker runtime from GCR
Configure our Prototype mbed Cloud Bridge for Google Cloud

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

Latest “git” installed

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

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”

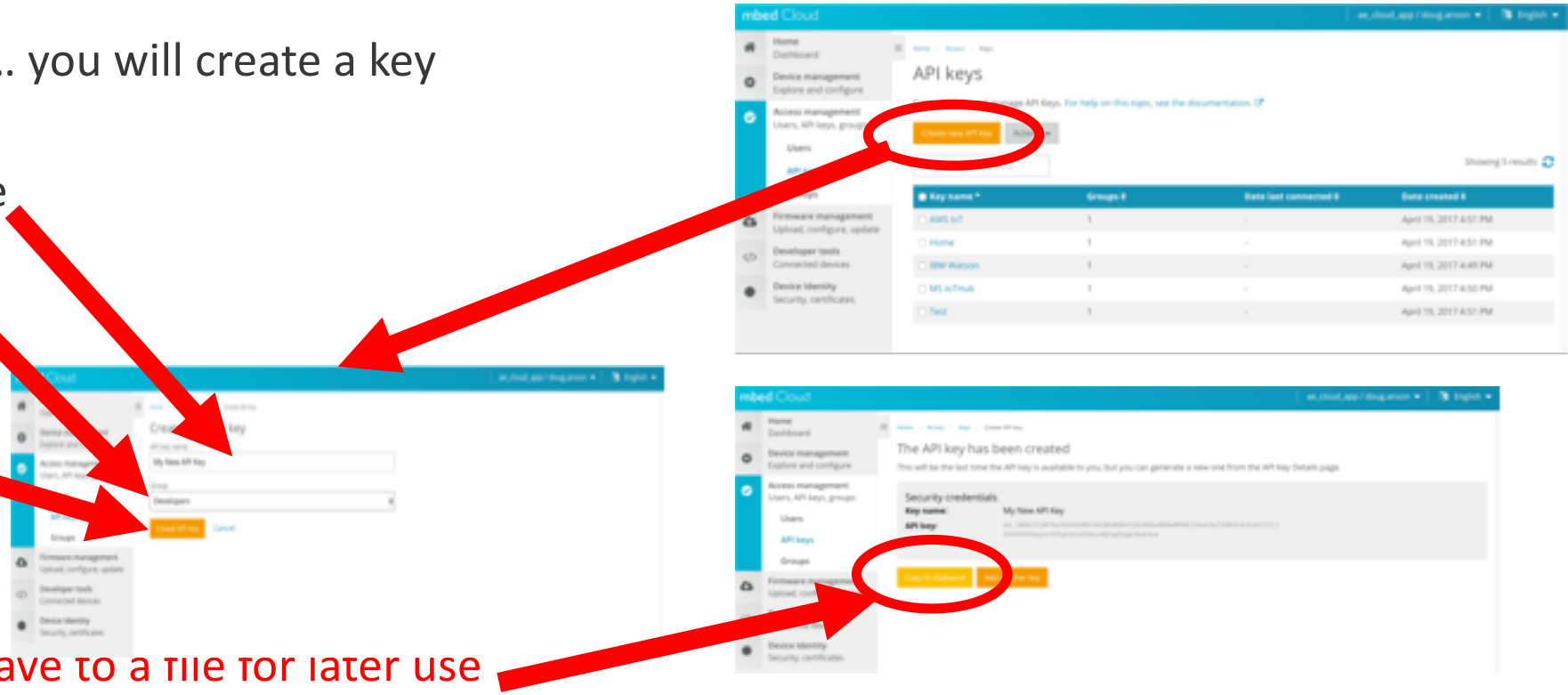
Press “Create new API Key ”… you will create a key

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/Name our Google Cloud Default Project

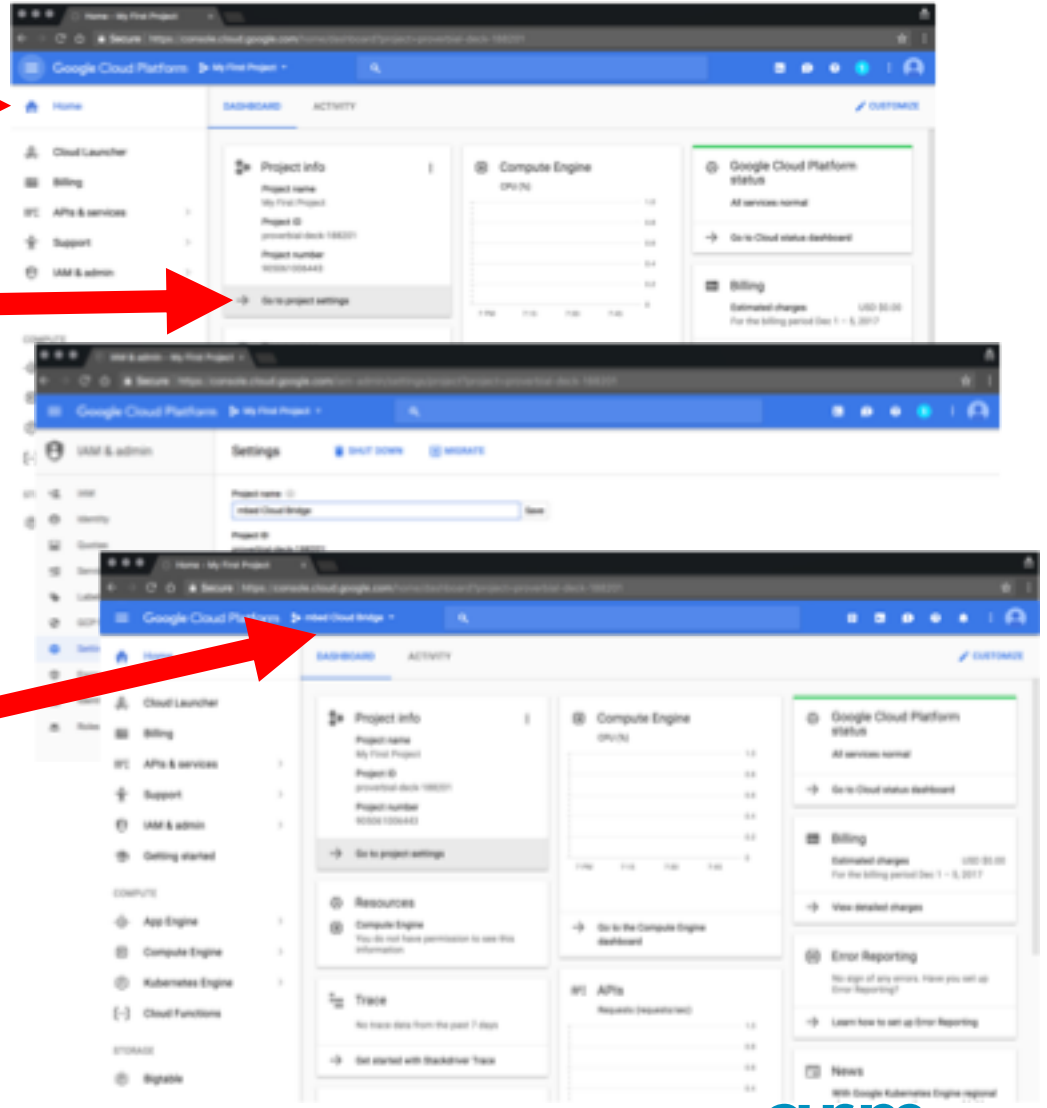
Go to your Google Cloud Console

Select “Project Settings”

New Project name: “mbed Cloud Bridge”

- Press ”Save”
- Refresh your browser!

Confirm that the default project name has changed



Create a Service Account

Go to your Google Cloud Dashboard

Select “Service Account”

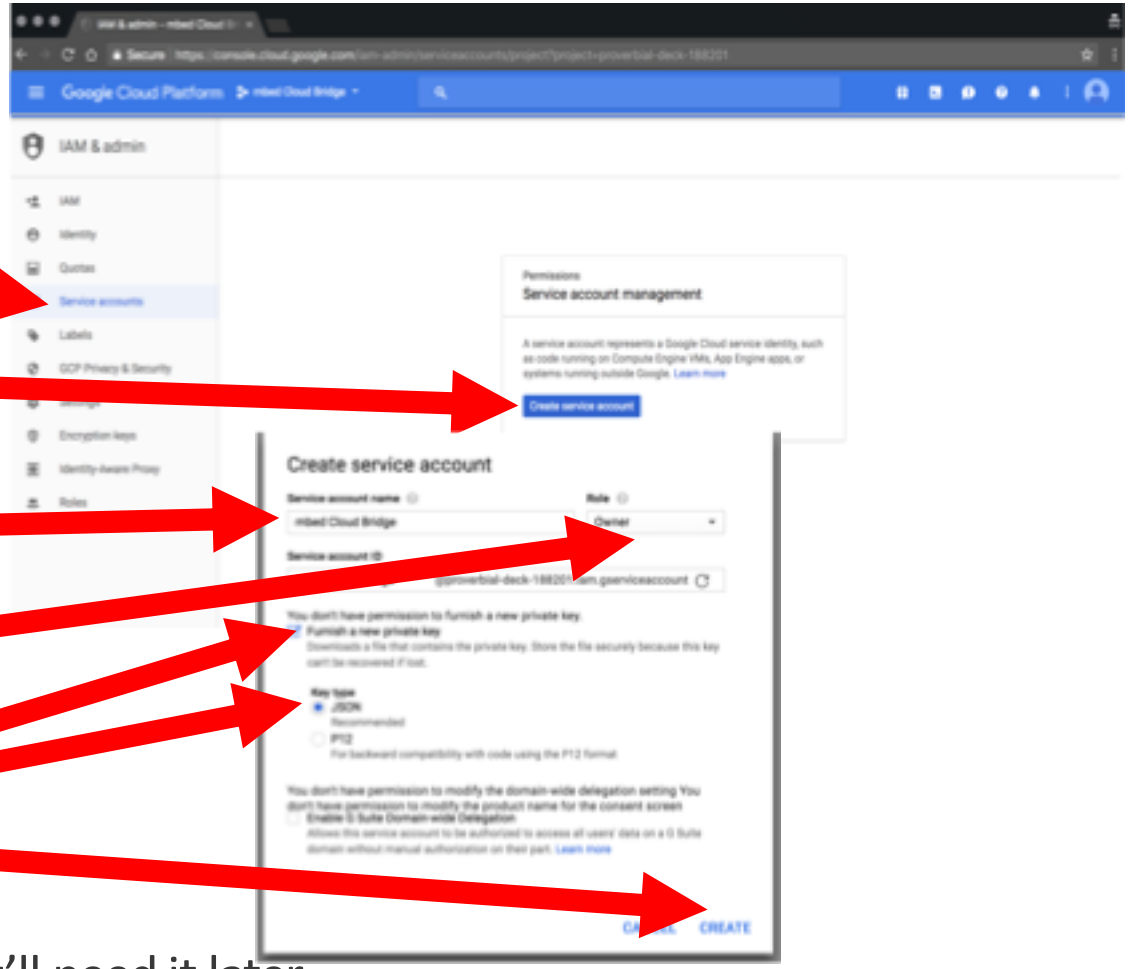
Press “Create a Service Account”

Name the service account

Select Project “Owner”

Confirm “JSON” selected, press Create

The JSON file down auto download - Save off this file – you’ll need it later.



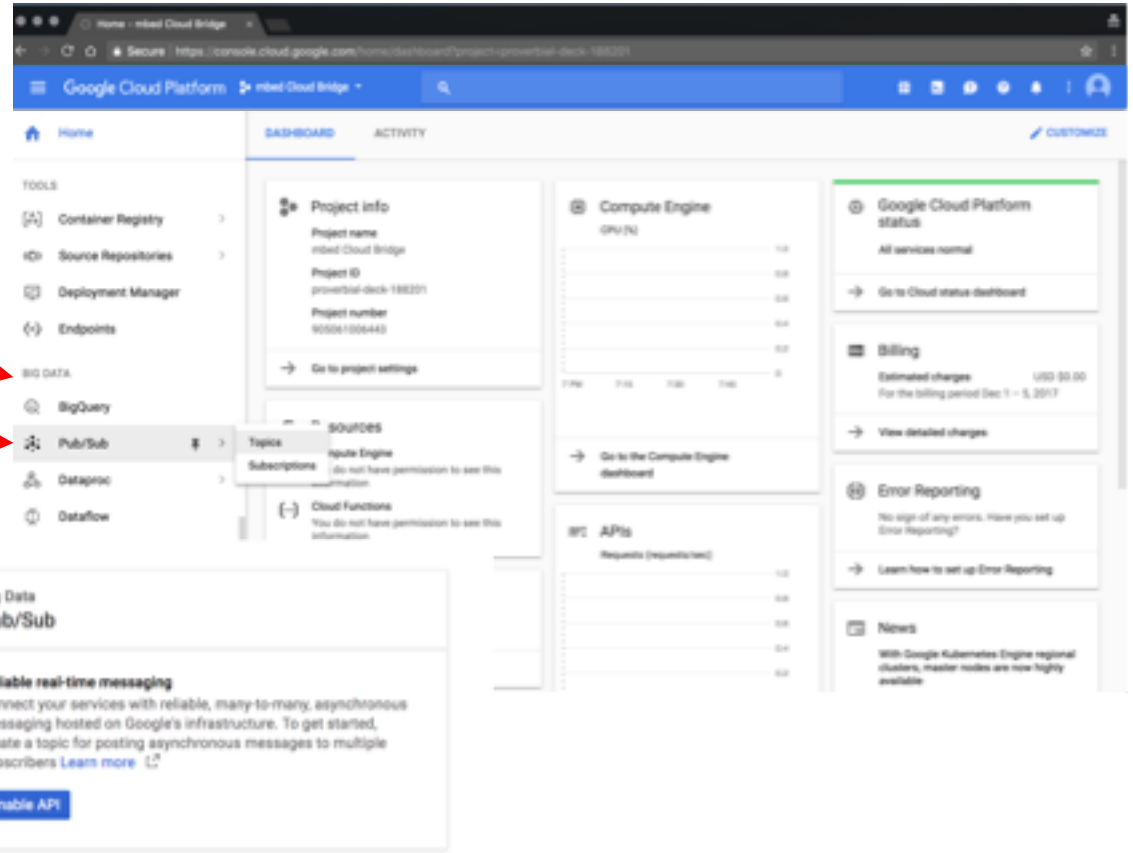
Create a Google Pubsub API Instance

Go to the Google Cloud Dashboard

Scroll down to the “Big Data” section

Select “Pub/Sub”

Press “Enable API”



Next, we have to create our Google CloudIoT API instance…

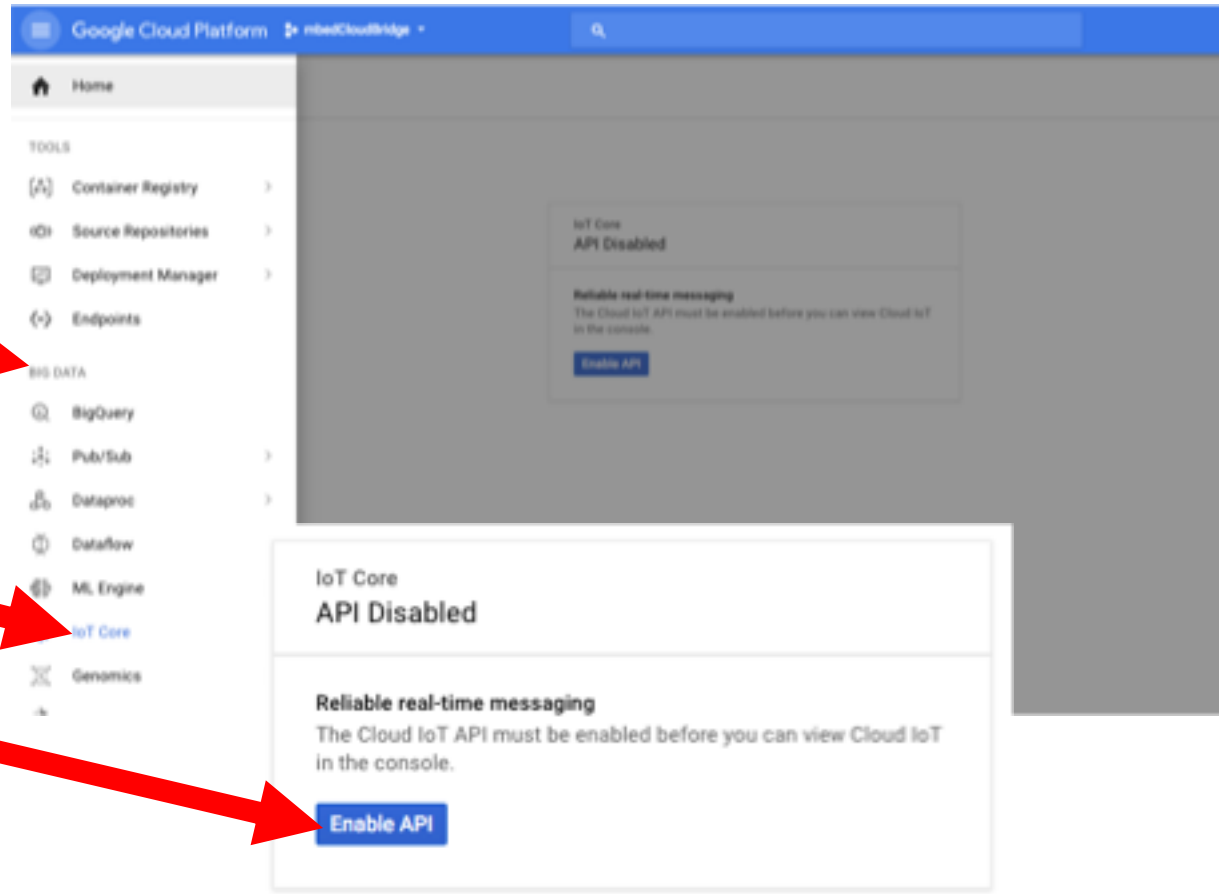
Create a Google CloudIoT API Instance

Go to the Google Cloud Dashboard

Scroll down to the “Big Data” section

Select “IoT Core”

Press “Enable API”

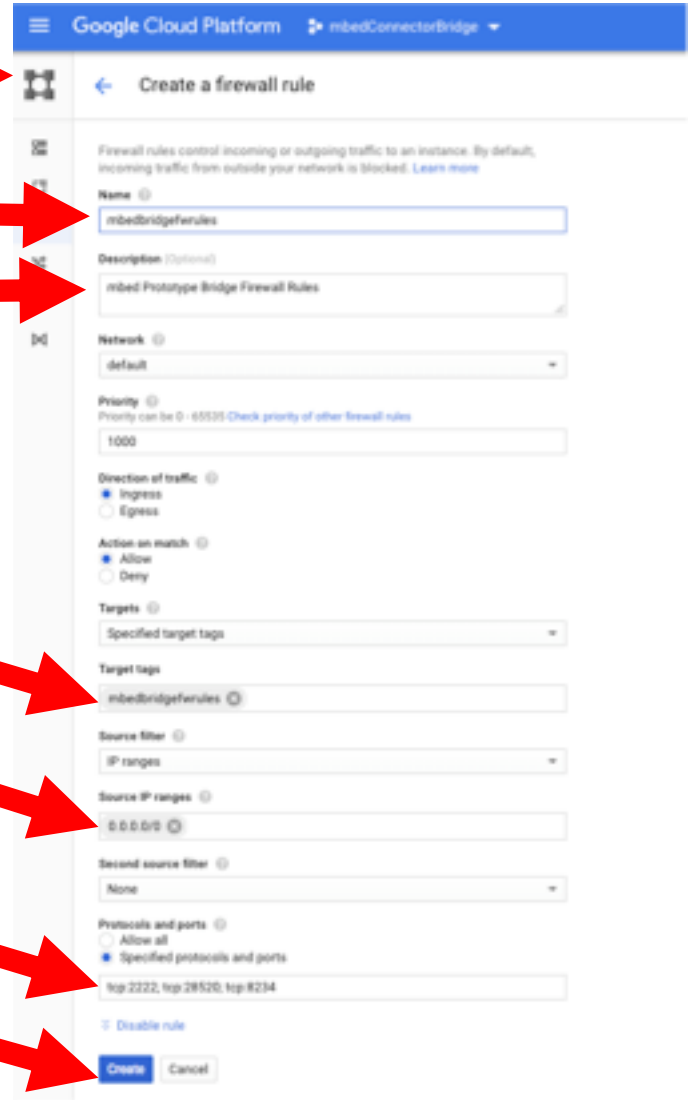


Once created, our Google Cloud Environment is ready for our bridge!

Import the mbed Cloud prototype Bridge for Google from GCR

First we have to create a special firewall rule via VPC Networks

- Name the firewall rule
- Provide a description
- Create a tag (remember this!)
- 0.0.0.0/0 for the port range
- Specify these ports
- Press “Create”



Import the mbed Cloud prototype Bridge for Google from GCR...

With “Compute Engine” select “VM Instances”

Select “Create”

Provide a name for your bridge

Choose “micro” machine type

Check the “deploy container image”

Container image entered here

gcr.io/mbedconnectorbridge-155920/github-douganson-connector-bridge-container-google:latest

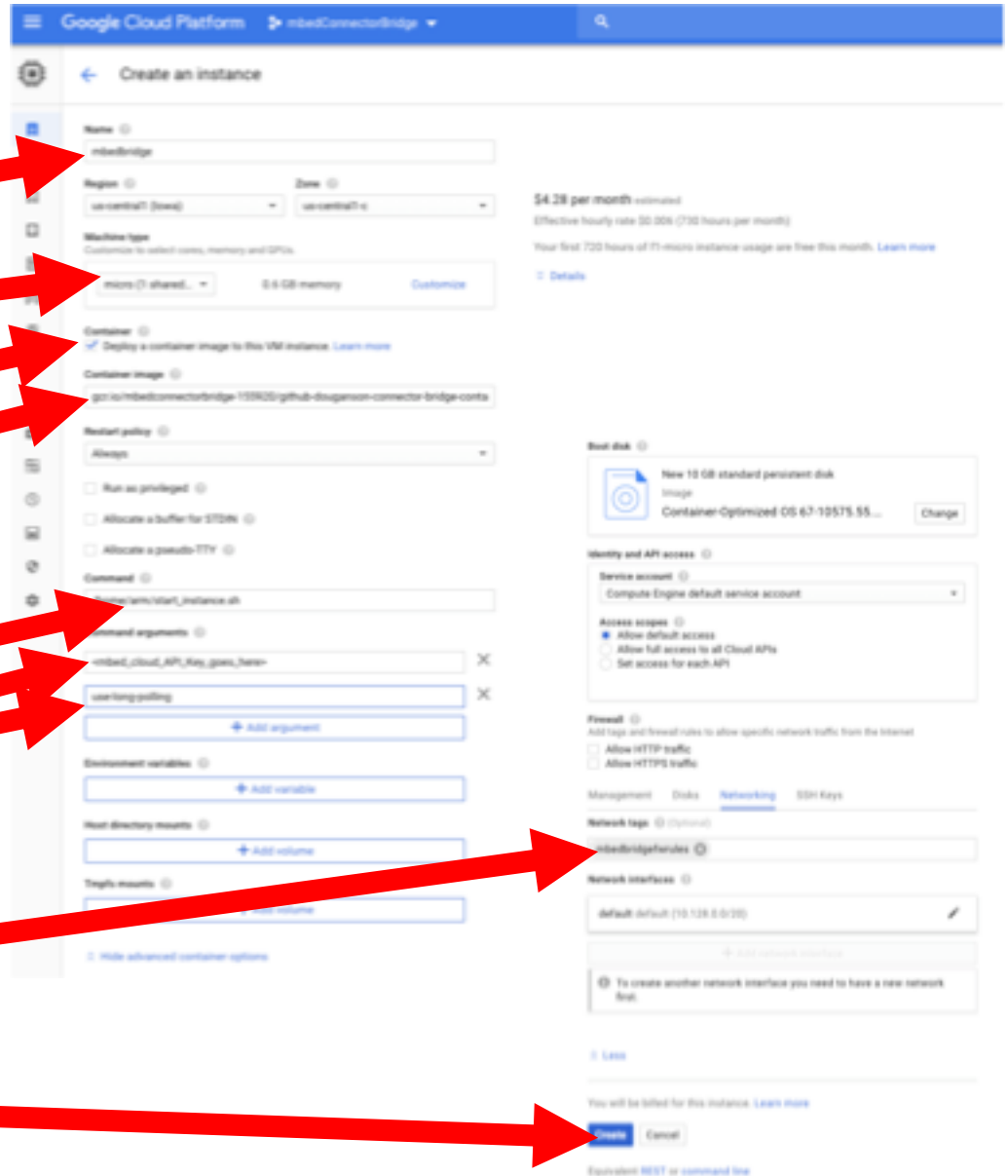
Command entered here

/home/arm/start_instance.sh

Command arguments

Network tag created from firewall rule

Press ”Create”

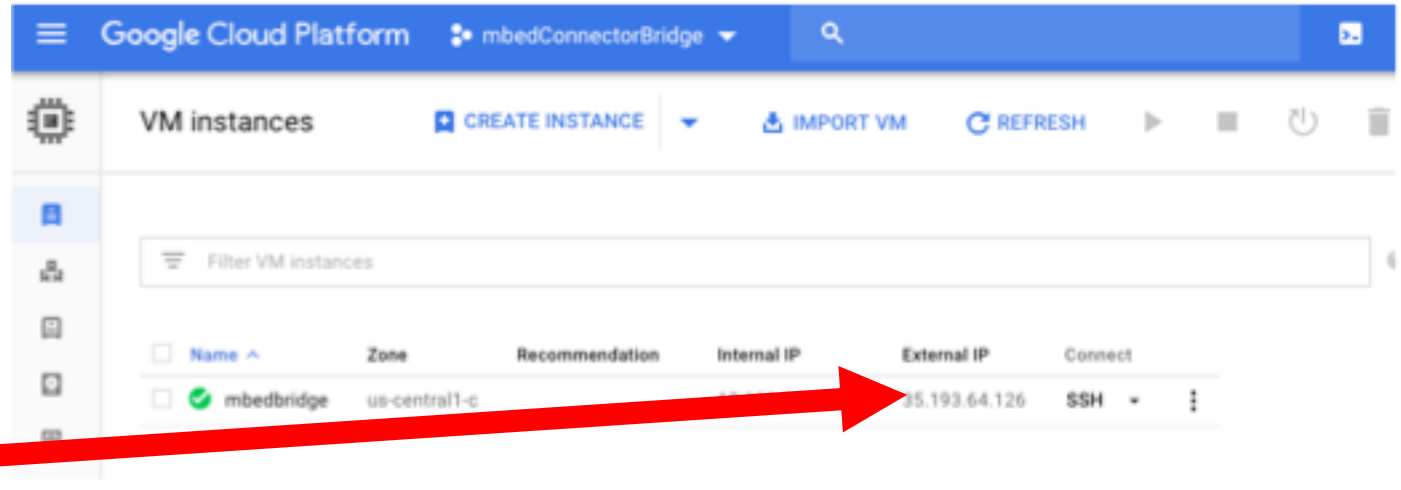


Import the mbed Cloud prototype Bridge for Google from GCR...

Once Created, you should see this:

Give this instance about 5 minutes to spin up and ready itself

Next, record this IP address



This IP address is your public-facing container IP address... You'll need it in the next step...

arm

Configure the mbed Cloud prototype Bridge for Google

Using your containers public-facing IP address:

Bring up Firefox/Chrome and go to:

https://<your public container IP address>:8234

Accept the self signed certificate

Username: admin PW: admin

Configure the mbed Cloud prototype Bridge for Google…

If not already entered, enter the mbed Cloud API Key/Token, *THEN* Press “Save”

Enter your Google Auth JSON, Press "Save”

NOTE: make sure you are pasting raw text!

-html markup will botch the config file…you then have to “./remove_bridge.sh” and “./get_bridge.sh” to retry…(slide 26)

-if, after saving part of the configuration page disappears, you’ll have to re-pull per above)

Ensure you have the right mbed API endpoint

- Mbed Cloud: api.us-east-1.mbedcloud.com
- Change to above and press “Save”

If your region is different, change, then “Save”

Press “RESTART”

Your mbed Cloud Bridge is now configured for Google Cloud

Putting it all Together

Press the “reset” button on your mbed device (next to the USB port)

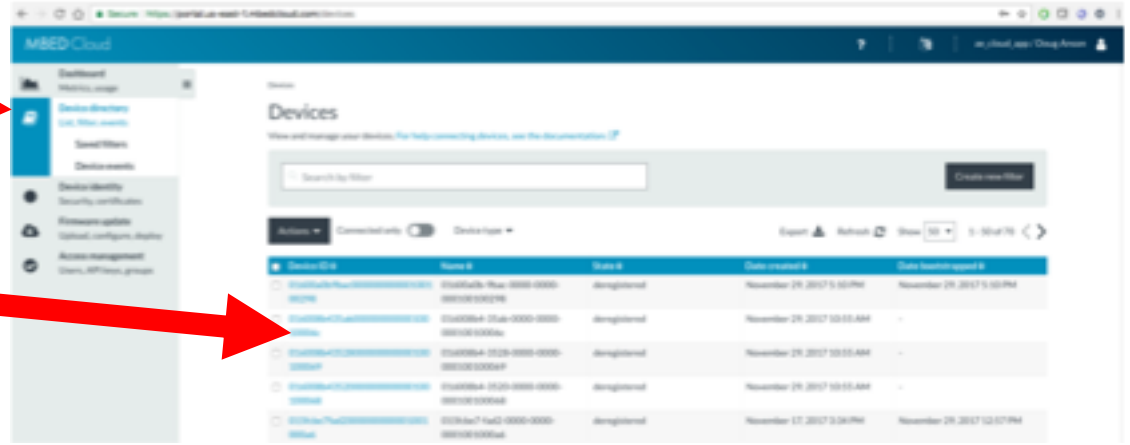
Ensure that you see “endpoint registered” in the serial terminal

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…

Lets check how things are working…



Install the Google Exerciser Application

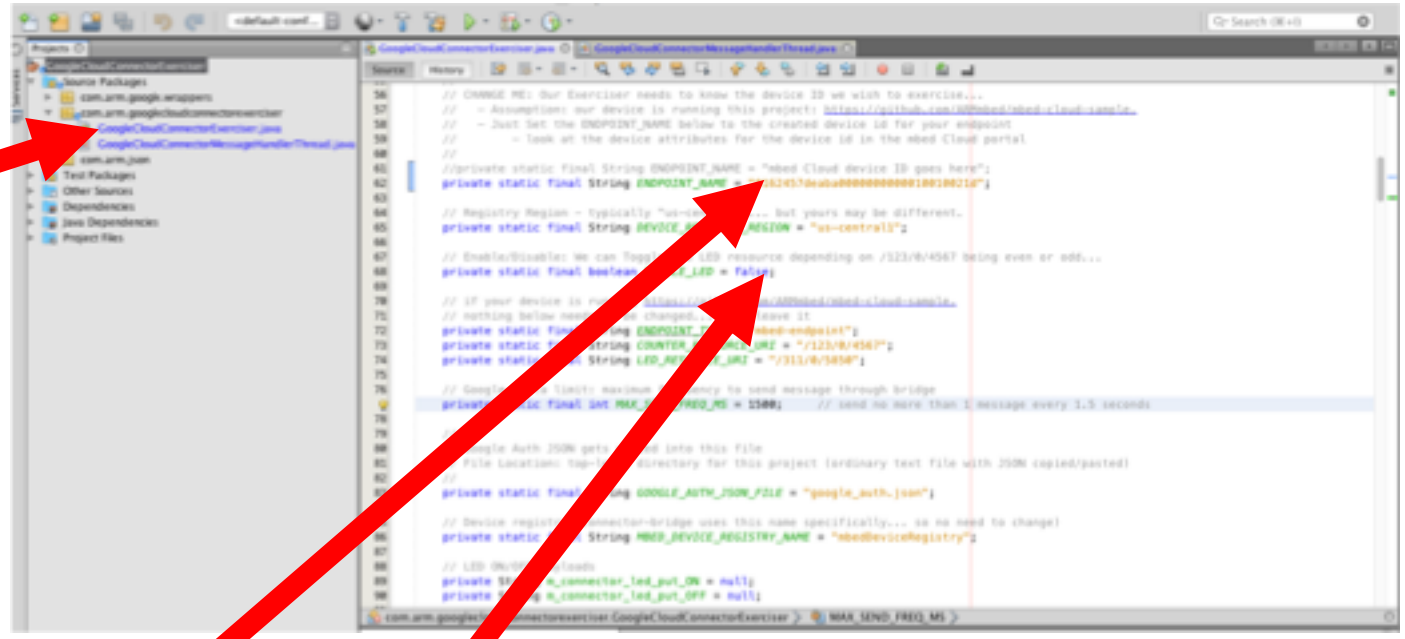
Install Java 8 JDK: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Install Java and the NetBeans IDE http://www.netbeans.org (Java SE support or “All”… either should be OK)

Clone the following GitHub repo: https://github.com/ARMmbed/GoogleCloudConnectorExerciser

Import into NetBeans as a Maven Project

Select this source file



Enter your mbed Cloud Endpoint Device ID you saved!

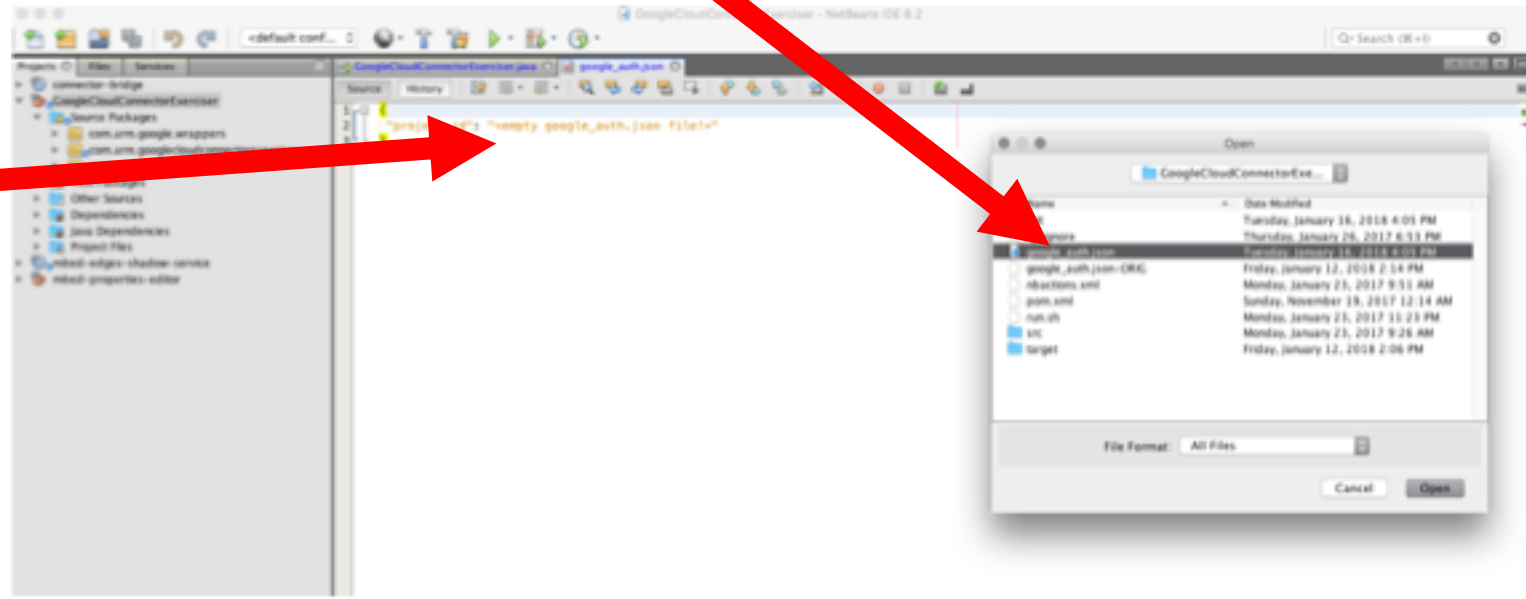
You can enable/disable the LED toggle (default is “false” or OFF)

Configuring the Google Exerciser Application (Auth)

In NetBeans, select File->Open.

Navigate to the project root directory and open “google_auth.json”

Paste the contents of your Google auth json and save.



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 and CloudIoT API Instances

Import our prototype mbed Cloud Bridge docker runtime for Google via GCR

Configure our prototype mbed Cloud Bridge for Google

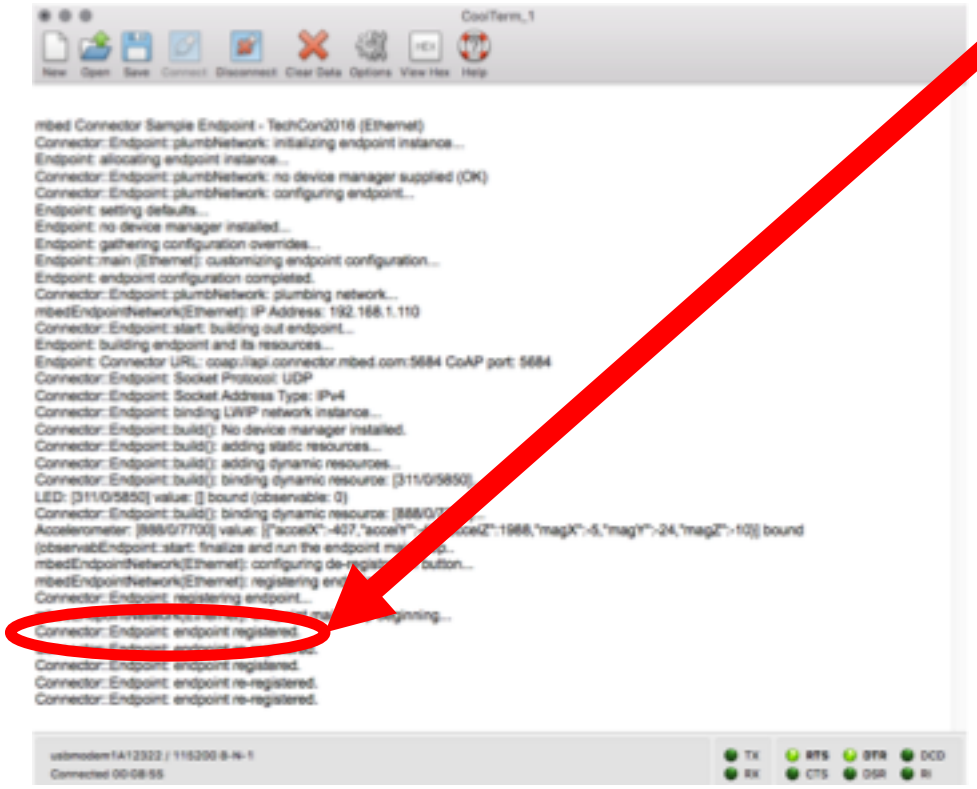
Installed and configured the "Exerciser" Application

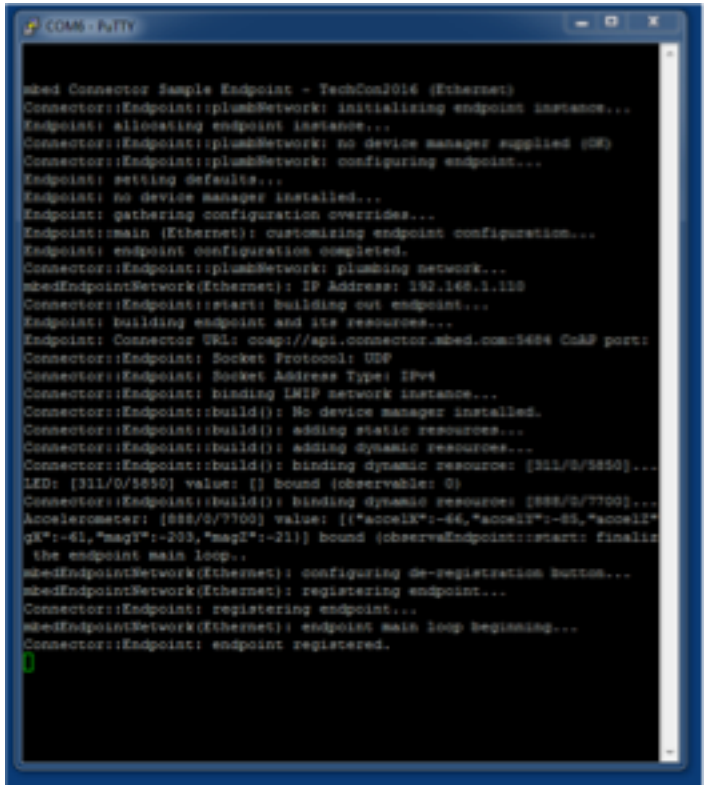
With our bridge now operational, we should be able to restart our endpoint and see messages coming into our Google via a java-based "Exerciser" Application

Putting it all Together: Check the Serial Terminal

Press the deregister button (SW3, slide 6)... wait about a minute or so... now Reboot your endpoint.... You should see output that looks something like this:

Make sure that you see a message like “endpoint registered”





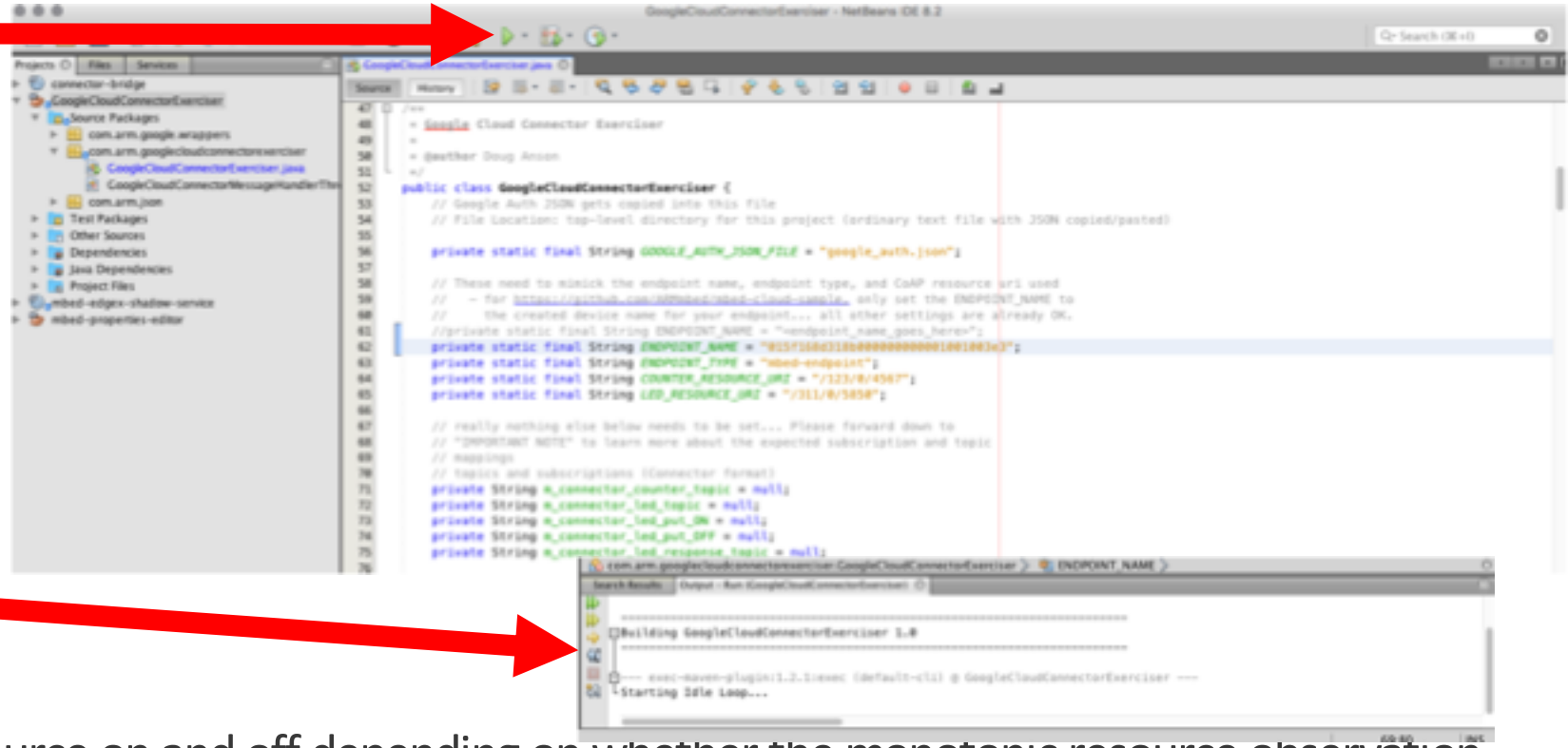
Running the Google Exerciser Application

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 (resource /123/0/4567) is even or odd… this demonstrates the bi-directional nature of the bridge with your endpoint!

CONGRATS!! You are finished!

Thank You!!