Hands-On Workshop: Develop Managed IoT Deployments with Arm Mbed OS and NXP Platforms

Mac Lobdell & Jim Carver

Arm, Inc

September 2018 | AMF-ENT-T3315

SECURE CONNECTIONS FOR A SMARTER WORLD

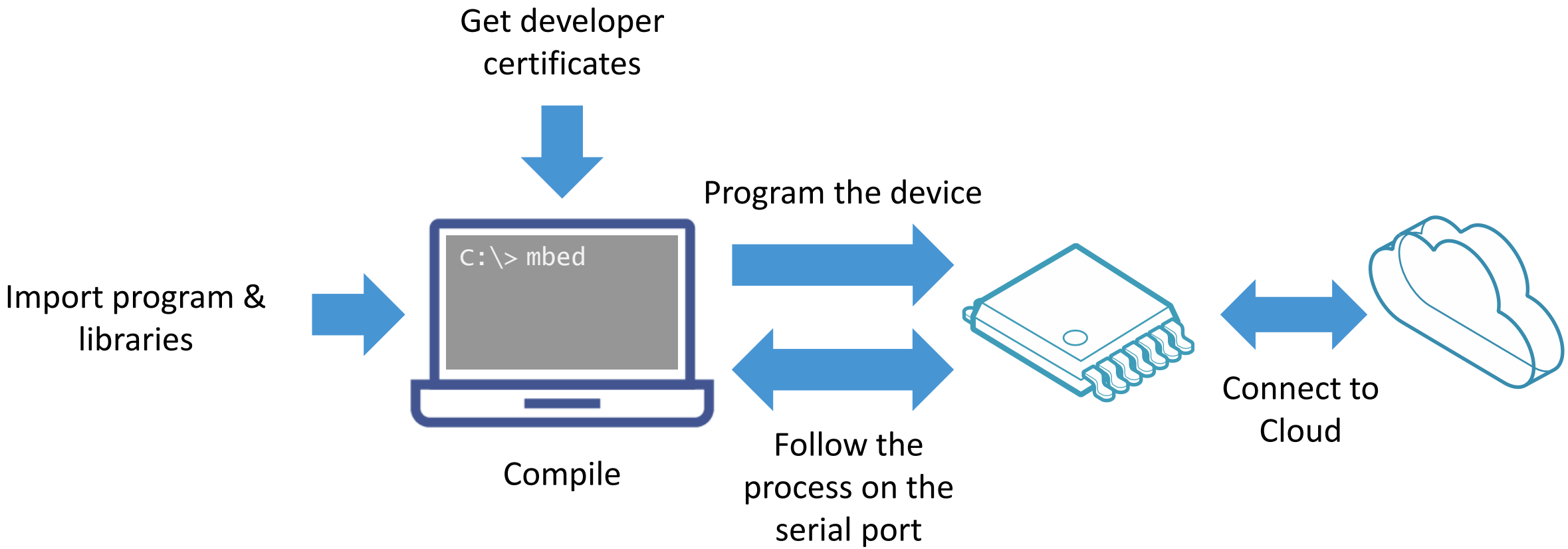
Abstract

Mbed OS & Pelion IoT Platform* Overview, Building device apps with NXP MCUXpresso IDE, Building device apps that connect to Pelion IoT Platform

* Formerly Mbed Cloud

Aims of the workshop

Connect a device to Pelion IoT Platform



Presenters

IoT Services Group
www.mbed.com

Mac Lobdell
Technical Account Manager
IoT Services Group
Arm, Inc

Jim Carver
Sr Mgr, Americas Business Development
IoT Services Group
Arm, Inc

Agenda

- Introduction
- Overview of NXP MCUs for Mbed OS
- Overview of Mbed OS & Pelion IoT Platform
- Hands on: Mbed OS app development on LPC54608
- Demos: Real World Applications, Firmware Updates
- Q/A
- Summary

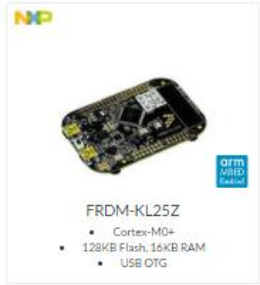
Mbed Enabled NXP MCUs

NXP platforms for Mbed OS app development

NXP platforms for Mbed OS app development

mbed LPC1768
- Cortex-M3, 96MHz
- 512KB Flash, 32KB RAM

mbed LPC11U24
- Cortex-M0, 48MHz
- 32KB Flash, 8KB RAM



FRDM-KL25Z
- Cortex-M0+
- 128KB Flash, 16KB RAM
- USB OTG

LPCXpresso4337
- Cortex-M4, up to 204MHz
- 1MB Flash, 136kB RAM
- Arduino Formfactor headers

FRDM-K20D50M
- Cortex-M4, 48MHz
- 128KB Flash, 16KB RAM, 32KB
- USB OTG

LPCXpresso824-MAX
- Cortex™-M0+, 30MHz
- 32KB Flash, 8KB RAM
- Arduino Formfactor headers

FRDM-KL43Z
- Cortex-M0+, 48MHz
- 256KB Flash, 32KB RAM
- segment LCD, USB crystal-less

FRDM-K66F
- Cortex-M4, 180MHz
- 2MB Flash, 256KB RAM
- Ethernet, microSD Card, Audio

FRDM-K82F
- Cortex-M4, 150MHz
- 256KB Flash, 256KB RAM
- USB, QuadSPI, Bootloader

NXP LPC800-MAX
- Cortex-M0+
- 16KB Flash, 4KB RAM

FRDM-KL46Z
- Cortex-M0+, 48MHz
- 256KB Flash, 32KB RAM
- USB OTG

FRDM-K64F
- Cortex-M4, 120MHz
- 1024KB Flash, 256KB RAM
- Ethernet, USB Crystal-less, SD

FRDM-K22F
- Cortex-M4, 120MHz
- 512KB Flash, 128KB RAM
- USB OTG Crystal-less

Ethernet IoT Starter Kit
- NXP K64F Processor
- mbed application shield
- IBM IoT Client pre-loaded

FRDM-KW24D512
- Cortex-M4, 50MHz
- 512KB Flash, 64KB RAM
- IEEE 802.15.4, ZigBee, Thread

FRDM-KL82Z
- Cortex®-M0+, up to 96MHz
- 128KB Flash, 96KB RAM
- USB OTG, QSPI Flash

NXP LPCXpresso54114
- 100MHz Cortex-M4F and M0+
- 256KB flash, 192KB SRAM

NXP LPCXpresso54608
- Cortex-M4F, 180 MHz
- 512 KB Flash, 200 KB SRAM

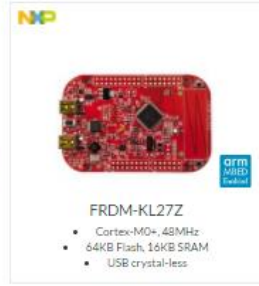
FRDM-KL05Z
- Cortex-M0+, 48MHz
- 32KB Flash, 4KB RAM

LPCXpresso1549
- Cortex-M3, 72MHz
- 256KB Flash, 36KB RAM
- Arduino Formfactor headers

LPCXpresso11U68
- Cortex-M0+, 50MHz
- 256KB Flash, 36kB RAM
- Arduino Formfactor headers

FRDM-KW41Z
- Cortex-M0+, 48MHz
- 512KB Flash, 128KB RAM
- BLE, IEEE® 802.15.4, Thread

Hexiwear
- Cortex-M4, 120MHz
- 1024KB Flash, 256KB RAM
- Color OLED display, 8x Sensors



FRDM-KL27Z
- Cortex-M0+, 48MHz
- 64KB Flash, 16KB SRAM
- USB crystal-less

NXP LPCXpresso54628
- Cortex-M4F, 220MHz
- 512 KB Flash, 200 KB SRAM

IMXRT1050-EVKB
- Cortex-M7, 600MHz
- Up to 512KB RAM
- Various Memory Interfaces

NXP platforms for Mbed OS by Core

Cortex-M0/M0+

mbed LPC11U24

- Cortex-M0, 48MHz
- 32KB Flash, 8KB RAM

FRDM-KL25Z

- Cortex-M0+
- 128KB Flash, 16KB RAM
- USB OTG

NXP LPC800-MAX

- Cortex-M0+
- 16KB Flash, 4KB RAM

FRDM-KL46Z

- Cortex-M0+, 48MHz
- 256KB Flash, 32KB RAM
- USB OTG

FRDM-KL05Z

- Cortex-M0+, 48MHz
- 32KB Flash, 4KB RAM

LPCXpresso11U68

- Cortex-M0+, 50MHz
- 256KB Flash, 36KB RAM
- Arduino Formfactor headers

LPCXpresso824-MAX

- Cortex ™-M0+, 30MHz
- 32KB Flash, 8KB RAM
- Arduino Formfactor headers

FRDM-KL27Z

- Cortex-M0+, 48MHz
- 64KB Flash, 16KB SRAM
- USB crystal-less

FRDM-KW41Z

- Cortex-M0+, 48MHz
- 512KB Flash, 128KB RAM
- BLE, IEEE® 802.15.4, Thread

FRDM-KL82Z

- Cortex®-M0+, up to 96MHz
- 128KB Flash, 96KB RAM
- USB OTG, QSPI Flash

Cortex-M3/M4

mbed LPC1768

- Cortex-M3, 96MHz
- 512KB Flash, 32KB RAM

FRDM-K64F

- Cortex-M4, 120MHz
- 1024KB Flash, 256KB RAM
- Ethernet, USB Crystal-less, SD

Ethernet IoT Starter Kit

- NXP K64F Processor
- mbed application shield
- IBM IoT Client pre-loaded

LPCXpresso1549

- Cortex-M3, 72MHz
- 256KB Flash, 36KB RAM
- Arduino Formfactor headers

FRDM-K20D50M

- Cortex-M4, 48MHz
- 128KB Flash, 16KB RAM, 32KI
- USB OTG

FRDM-K22F

- Cortex-M4, 120MHz
- 512KB Flash, 128KB RAM
- USB OTG Crystal-less

Hexiwear

- Cortex-M4, 120MHz
- 1024KB Flash, 256KB RAM
- Color OLED display, 8x Sensor

FRDM-K66F

- Cortex-M4, 180MHz
- 2MB Flash, 256KB RAM
- Ethernet, microSD Card, Audio

FRDM-KW24D512

- Cortex-M4, 50MHz
- 512KB Flash, 64KB RAM
- IEEE 802.15.4, ZigBee, Thread

FRDM-K82F

- Cortex-M4, 150MHz
- 256KB Flash, 256KB RAM
- USB, QuadSPI, Bootloader

NXP LPCXpresso54114

- 100MHz Cortex-M4F and M0+
- 256KB flash, 192KB SRAM

NXP LPCXpresso54608

Cortex-M7

IMXRT1050-EVKB

- Cortex-M7, 600MHz
- Up to 512KB RAM
- Various Memory Interfaces

NXP platforms for Pelion Device Management

Requirements

- RAM: 128K or greater
- Flash: 512K or greater
- True Random Number Generator (TRNG)
- Flash In-Application-Programming (IAP) driver
- Real Time Clock (RTC)
- Storage (Internal Flash, SPI Flash, SD card)
- IP connectivity (Eth, Wi-Fi, Cellular, 6lowpan, Thread)
- Mbed OS *

Tested w/ Pelion

Get to know your Mbed hardware – LPCXpresso54608

During this workshop, we are going to use the LPCXpresso54608

https://os.mbed.com/platforms/LPCXpresso54608

Micro USB Cable
(Power, programming, debugging interface, serial)

Reset

LPCXpresso54608 Highlights

- LPC54608ET512 Arm Cortex-M4 @180MHz
- 512 KB Flash, 200 KB SRAM
- 10/100 Mbps Ethernet
- On-board debug interface
- Expansion (Arduino UNO and Pmod™)
- 272x480 color LCD w/ touch screen
- 128 Mb Micron MT25QL128 Quad-SPI flash

For compatibility with Mbed os, the debug interface firmware should be updated to the latest DAPLink version

https://os.mbed.com/teams/NXP/wiki/Updating-LPCXpresso-firmware

Tools for Development

During the first part of the workshop, we are going to use the Mbed online compiler for a fast getting-started experience

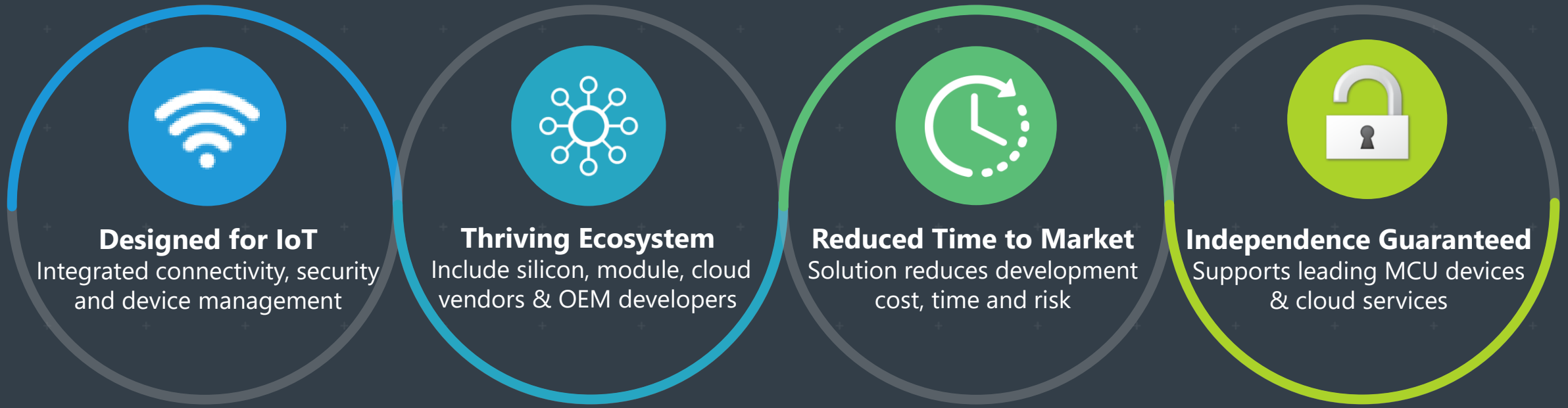
You can also use MCUXpresso IDE (see last section)

Mbed OS Overview

Arm Mbed OS is a free, open-source embedded operating system designed specifically for the "things" in the Internet of Things.

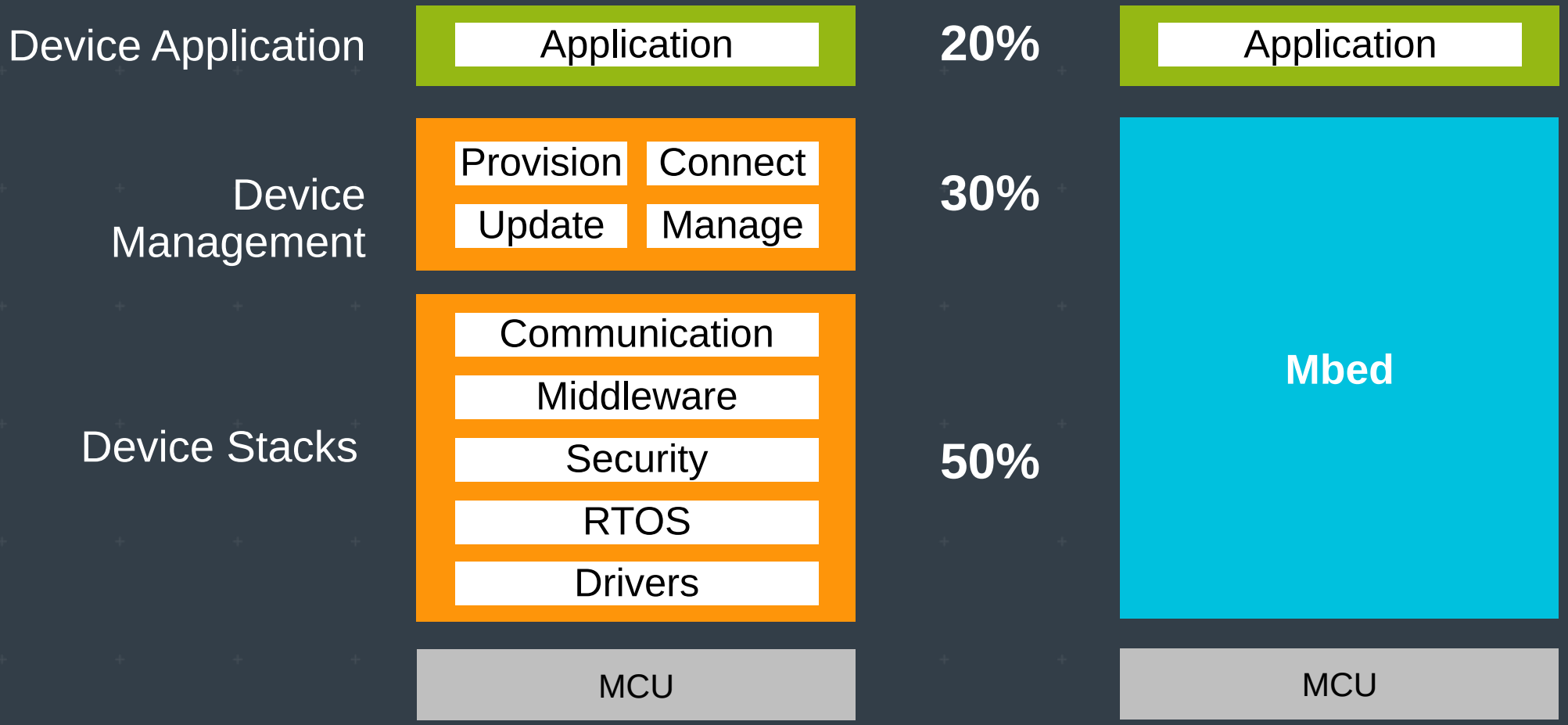
What is Mbed?

Operating system solutions for meeting the complex requirements of IoT devices



Developed in collaboration with silicon partners
and provided free to silicon customers
to accelerate creation of managed IoT devices

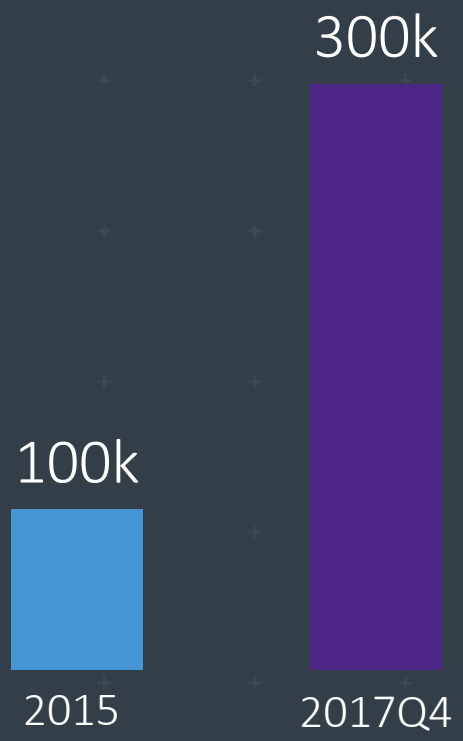
Where is the development complexity?



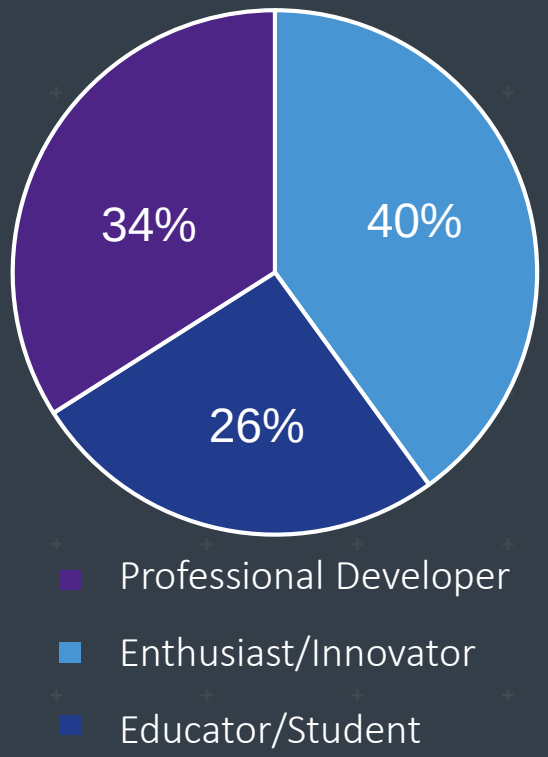
Mbed developers

Growing a powerful channel to IoT product developers

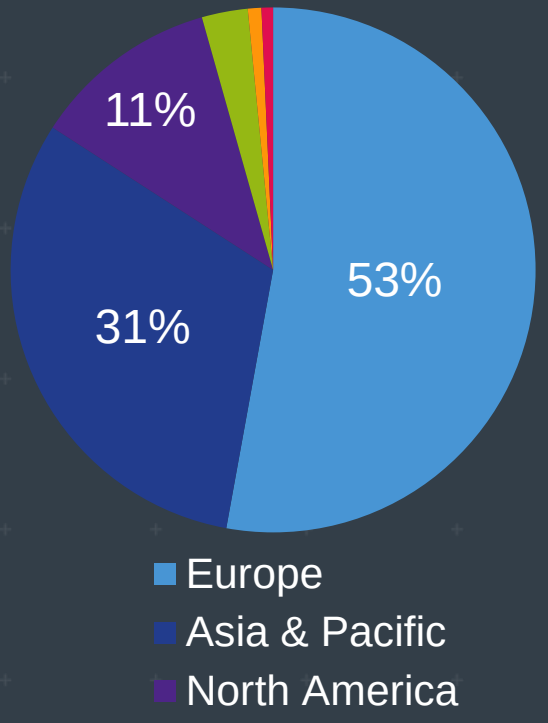
Accelerating developer ecosystem growth



A third of developers are professionals



A global footprint



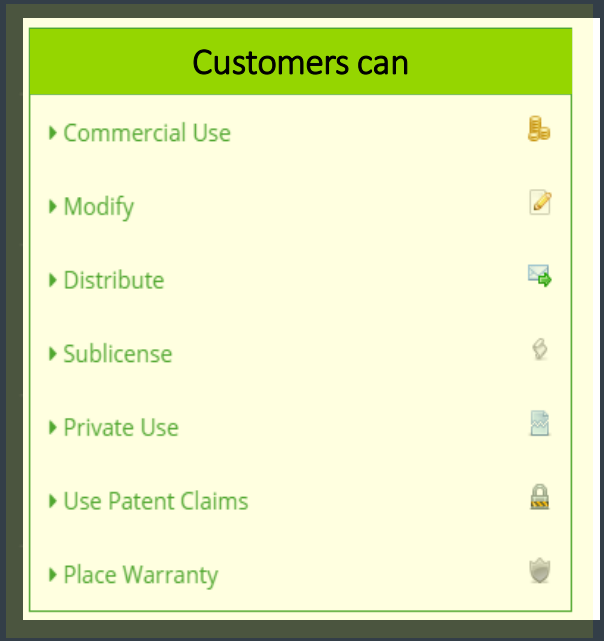
source: mbed.com, Mbed developer survey 2017

Mbed OS designed for reuse

Mbed OS is based on Apache 2.0 License

- **Designed for reuse**, customers can do whatever they want with the software, as long as they include the required notices

Customers can do almost everything with Mbed OS



How Mbed OS Integration Works

Mbed OS Reference implementation → Testing of Reference Implementation → Adoption from Partners → Testing of Partner Adoption → Release

In last one year, 300+ engineers contributed 4500+ commits

- 5 Million+ line of tested code available
- Majority of the contribution are by community and partners

Code testing is completely automated

- Testing is done on partner boards using test farm and continuous integration systems
- The results are publicly available linked to Github status

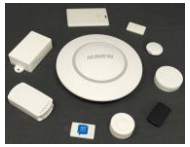
Built with Mbed

LightGrid

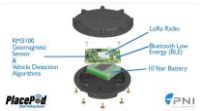
Smart Speakers

Tilt monitor

Beacons



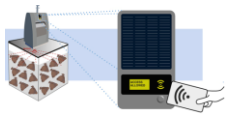
Parking sensor



Gateway

Asset tracker

Smart city bins



Factory humidity

Indoor positioning

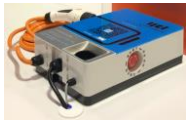
Street light

Smart City

Industrial sensor

Submarine

EV Charger



Agriculture

Smart shelf

Patient tracker

Many More…

Office lighting

Luminaire

Gateway

Electric meter

Parking barrier

City mapping

Pedometer

Water sensor

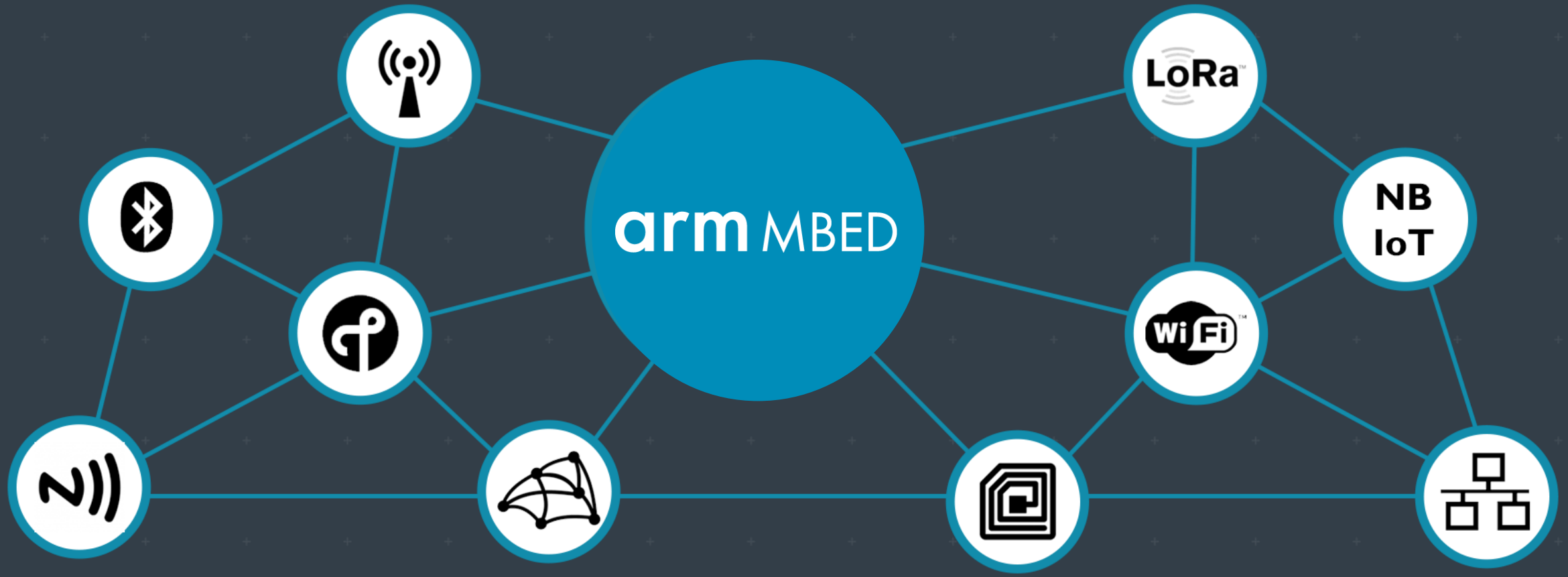
Air vent

Food allergy

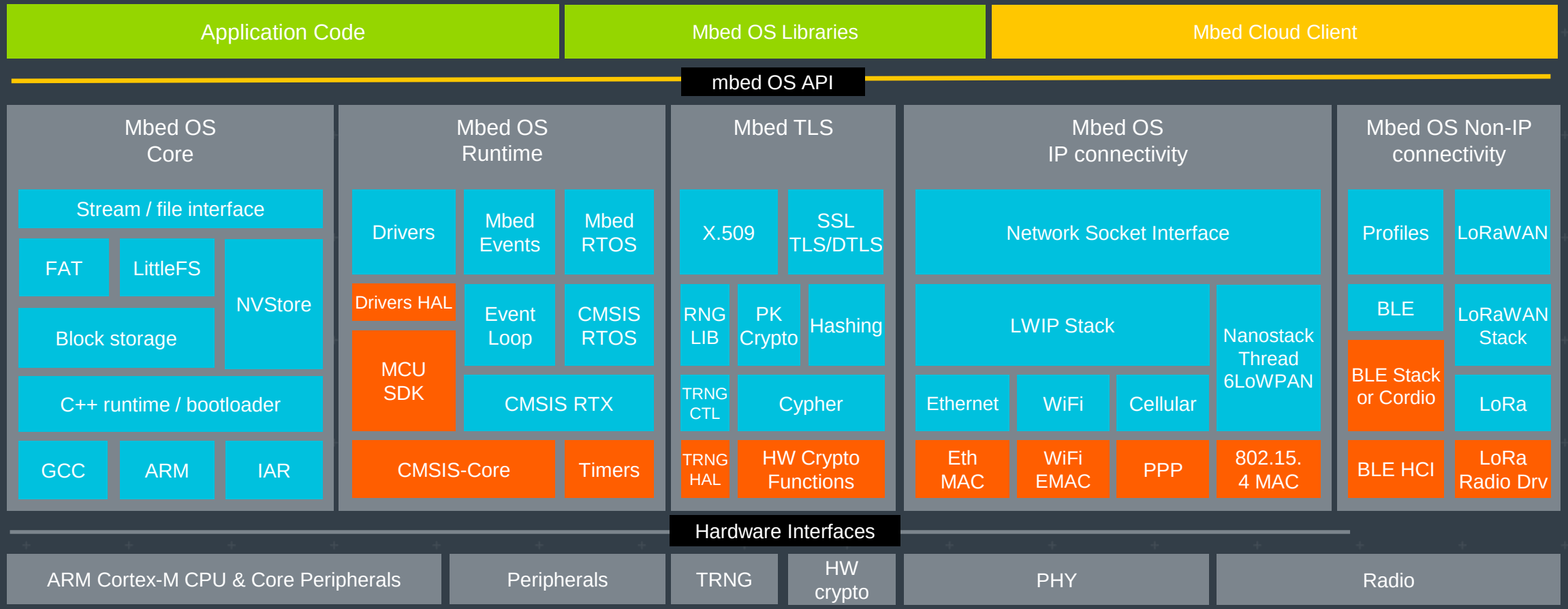
Door lock

Solving connectivity challenges

Mbed OS supports a diverse connectivity portfolio for a diversity of IoT applications

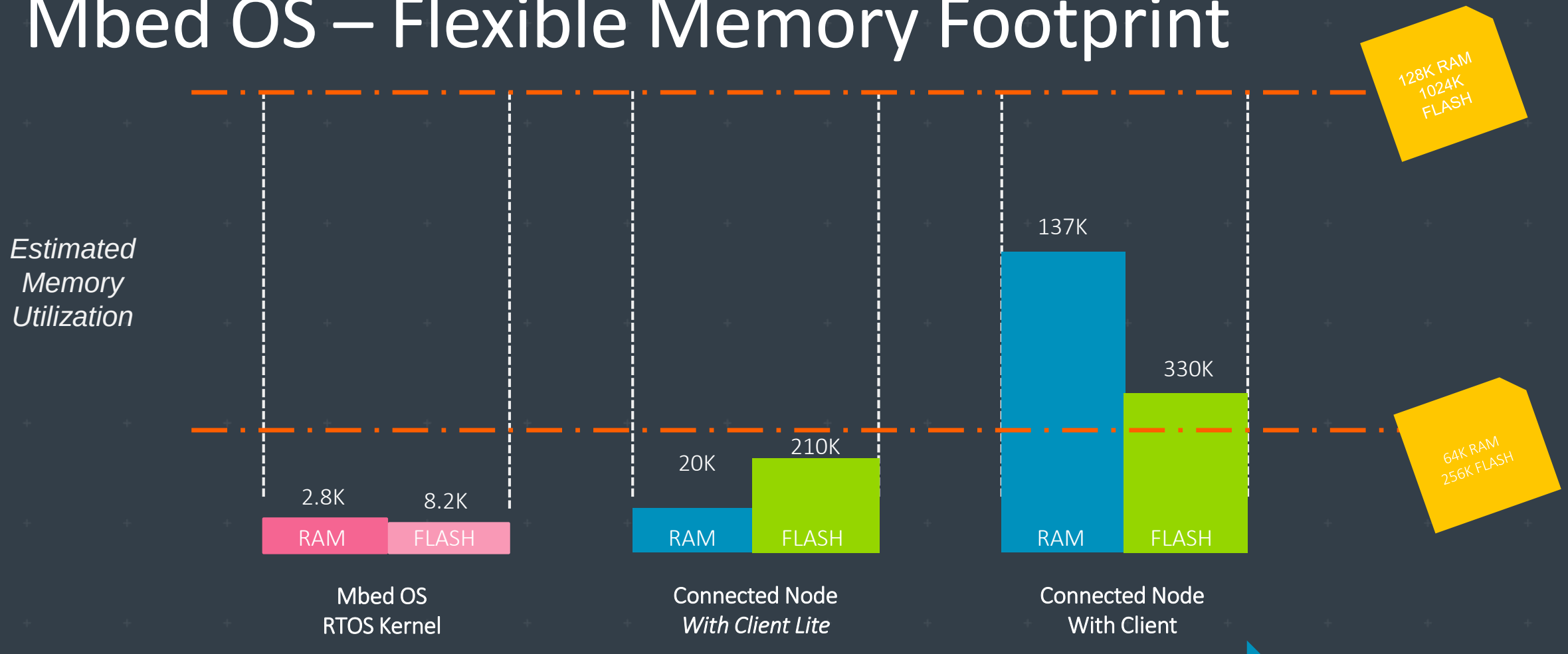


Mbed OS



Mbed OS partners and community | Mbed Client components | Partner components/ports | Mbed OS components

Mbed OS – Flexible Memory Footprint



Mbed OS tools

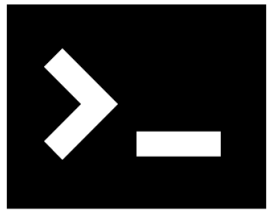
Free web-based tools for building, debugging,

Rich third-party tool support for most popular tools

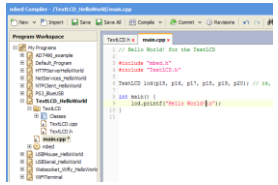
Mbed OS IDEs and toolchains

Arm Compiler 5

Mbed OS core tools

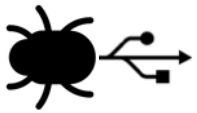


Mbed CLI
Command Line Interface



Mbed Compiler
Free Online IDE

Mbed Greentea
Porting Testsuite and CI



Mbed pyOCD
CMSIS-DAP Debug Library

Mbed DAPLink
CMSIS-DAP Debug Firmware

Mbed OS DVCS support

git

mercurial

GitHub

Atlassian Bitbucket

GitLab

mbed.org/code

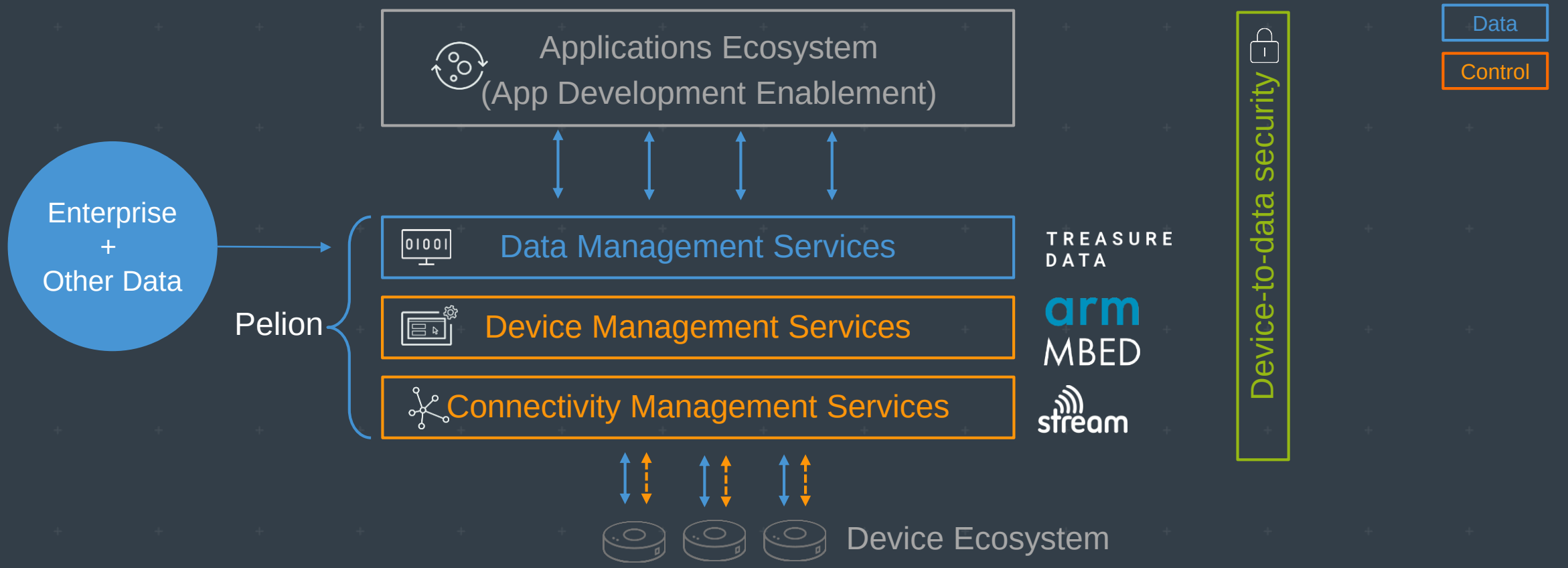
Pelion IoT Platform

Pelion Device Management provides flexible, secure and simple IoT device management for any device, any network and any cloud.

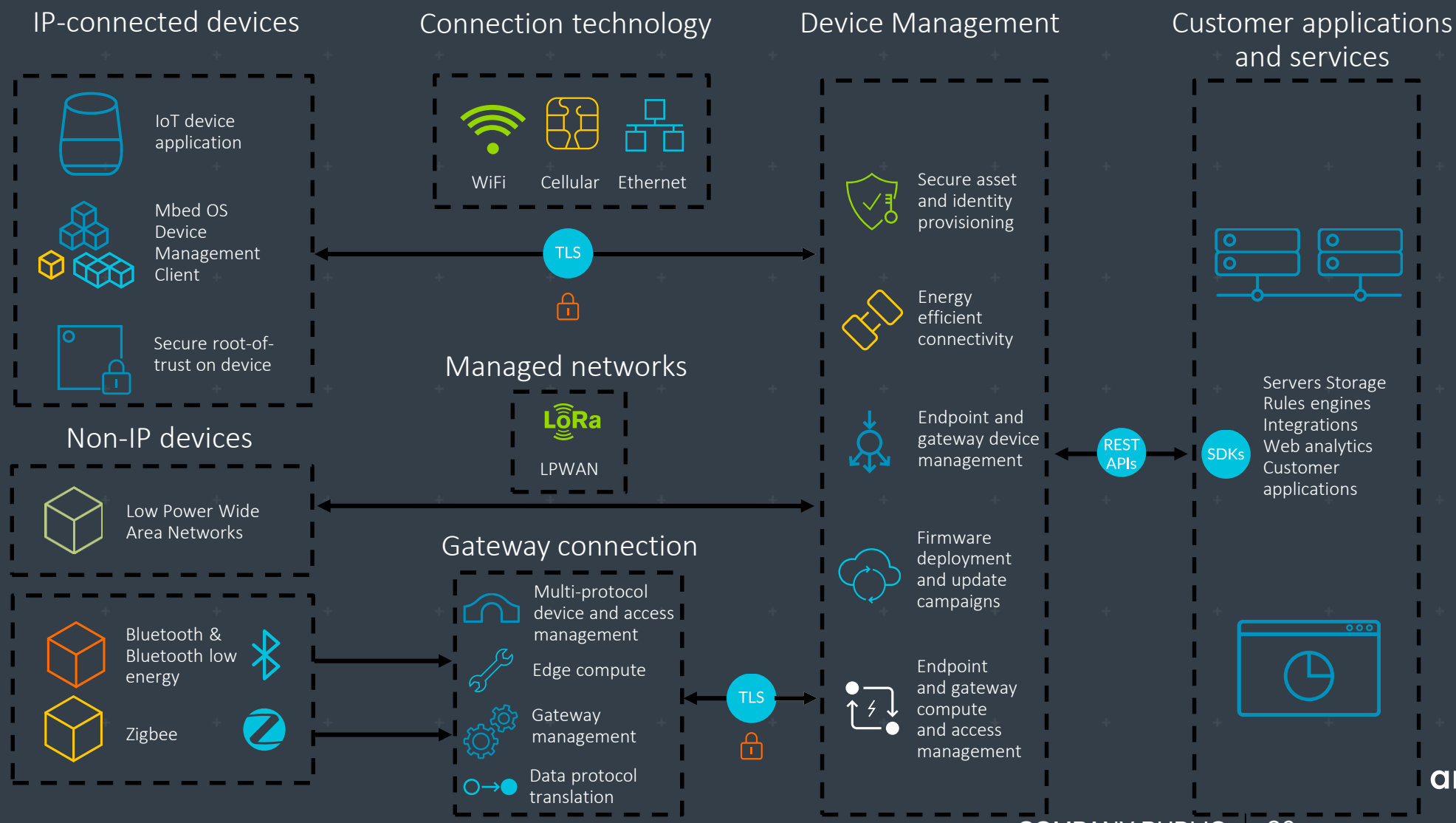
(Formerly branded as Mbed Cloud)

Introducing the Pelion IoT Platform

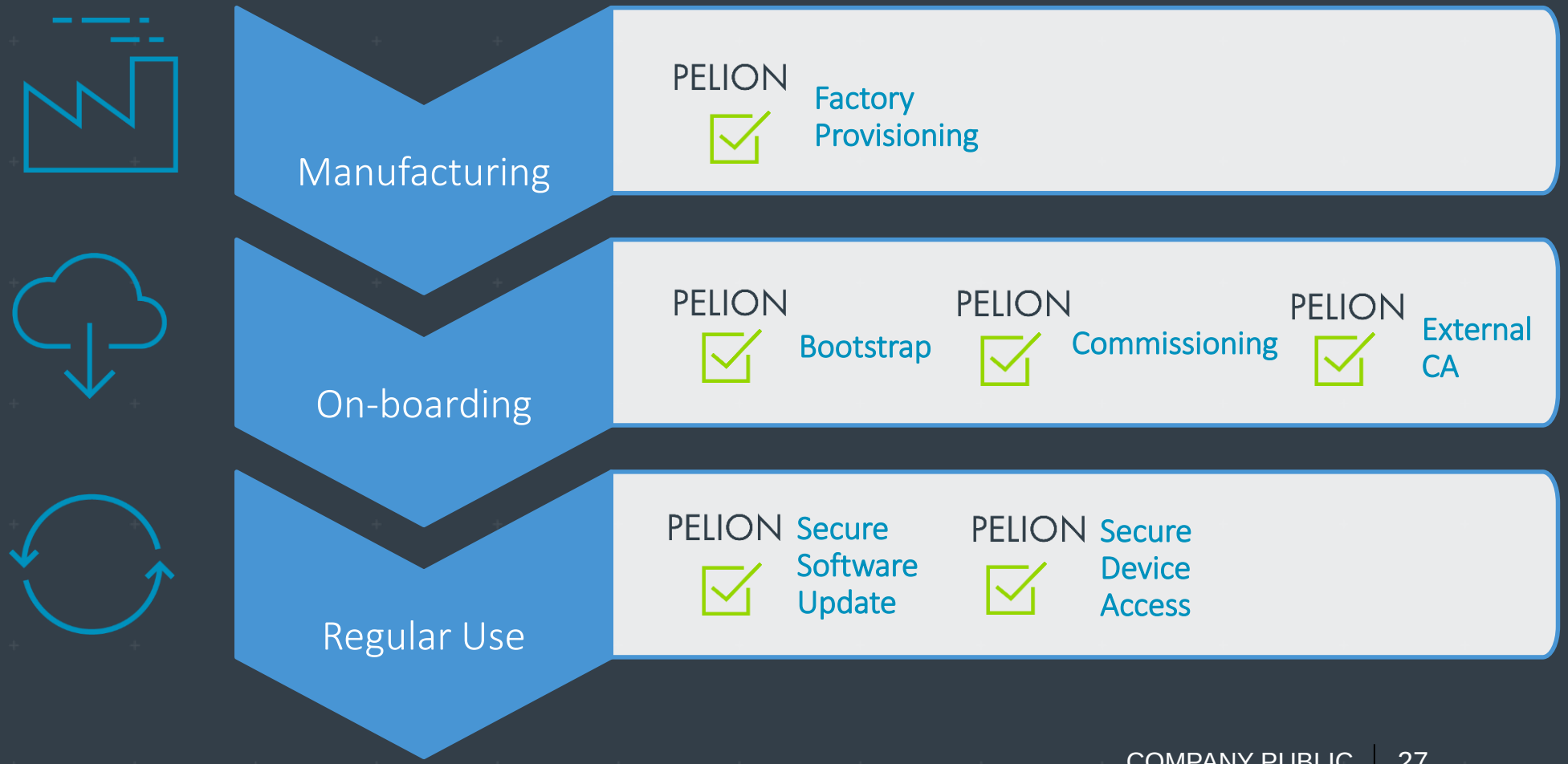
End-to-end services built on Arm IPG+ISG security framework



Flexible deployment and management



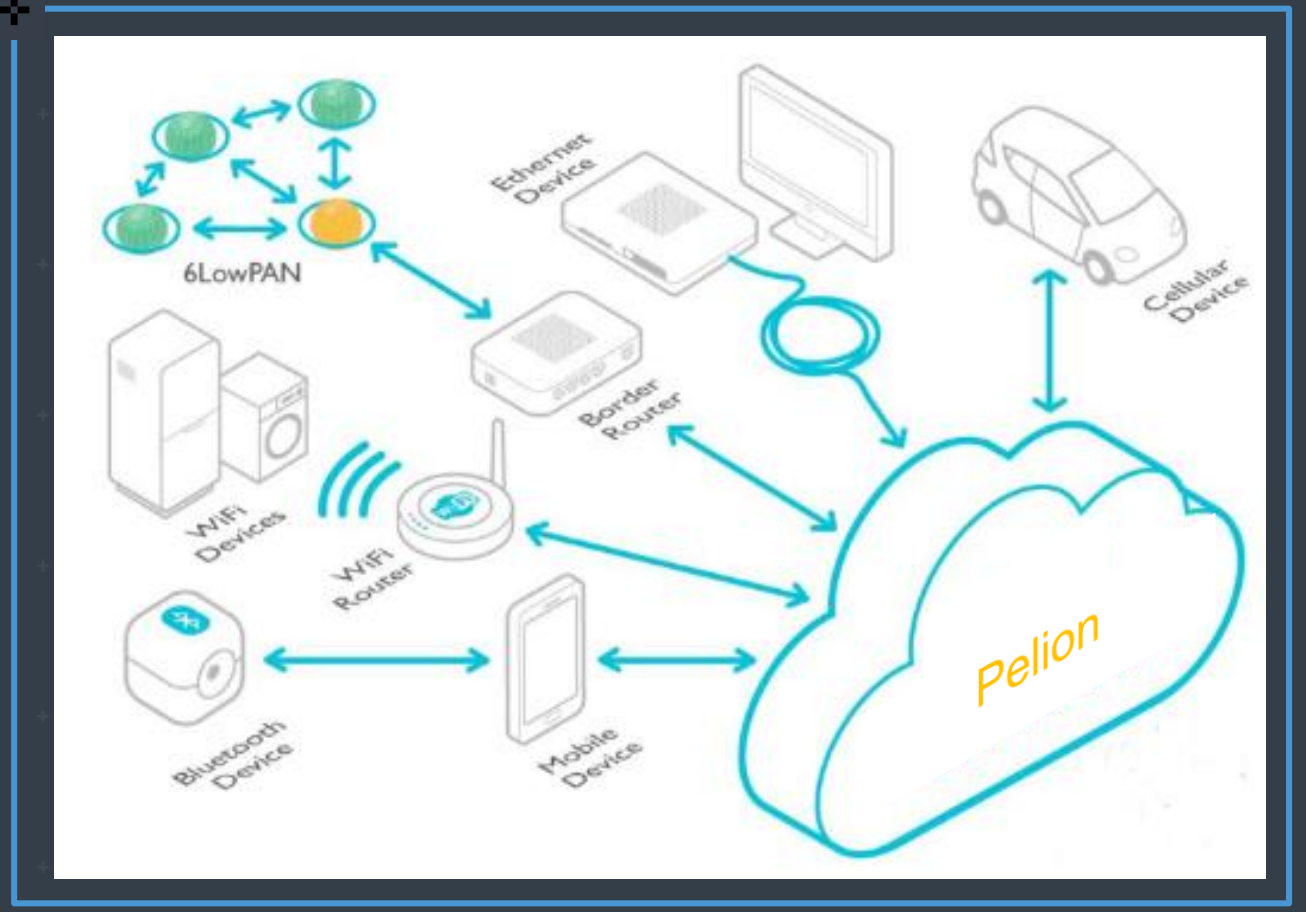
Pelion Device Management secures all stages of device life-cycle



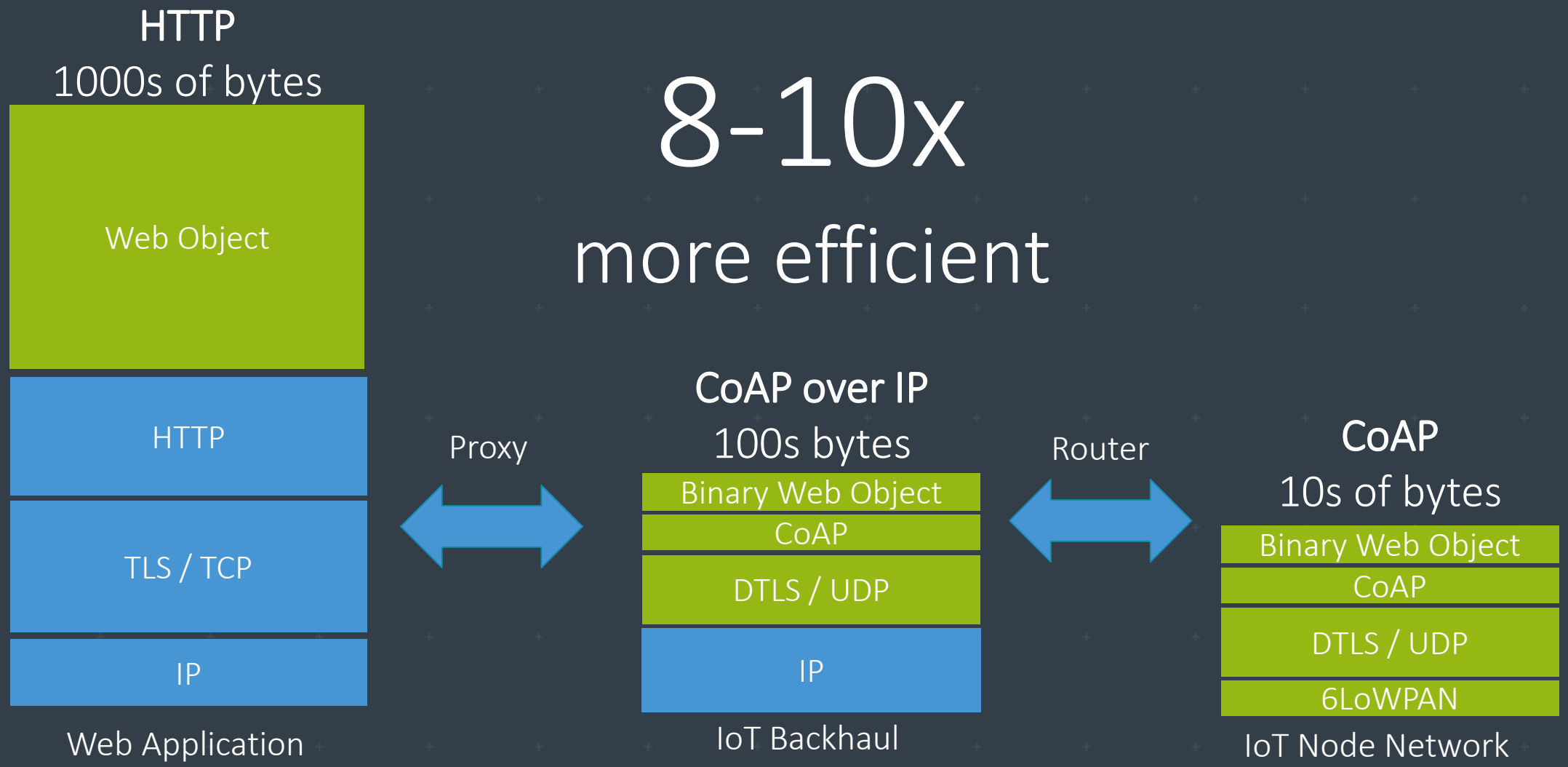
Pelion Device Management Connect

Simple, secure and energy efficient IoT connectivity solution for wide range of devices and enabling unified connectivity from cloud applications

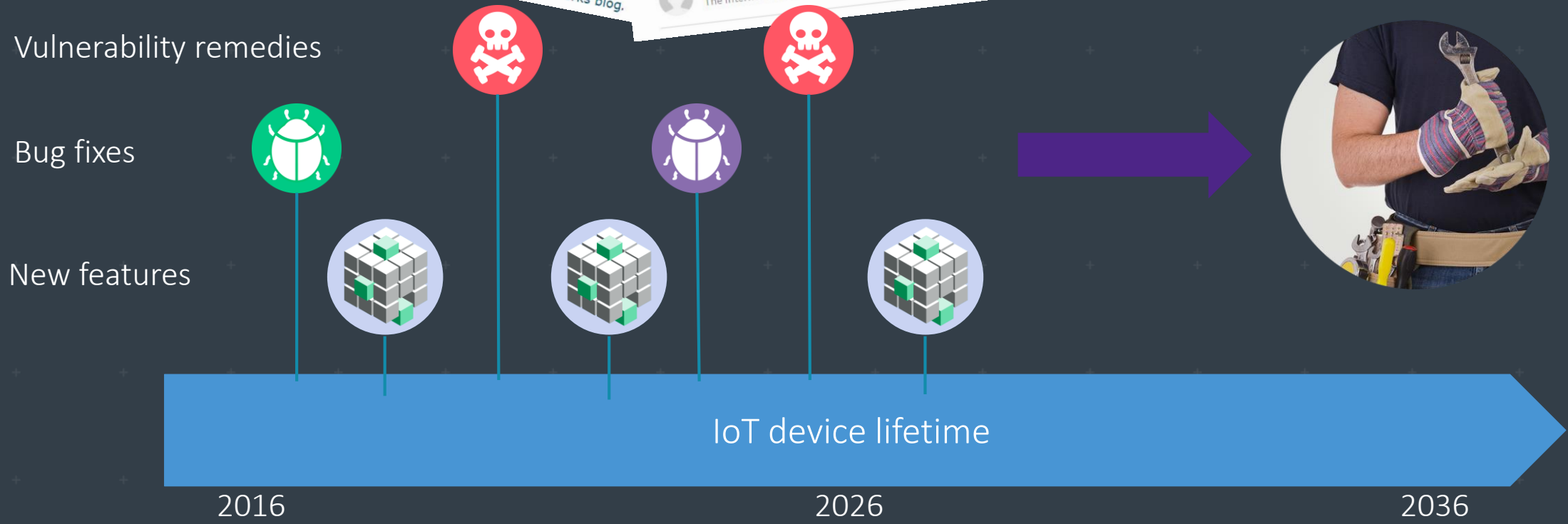
- Standard: OMA LWM2M, CoAP and TLS/DTLS
- Secure device bootstrap and onboarding
- Optimized for constrained and battery-operated devices
- IoT device communication via REST APIs to enterprise software and web apps



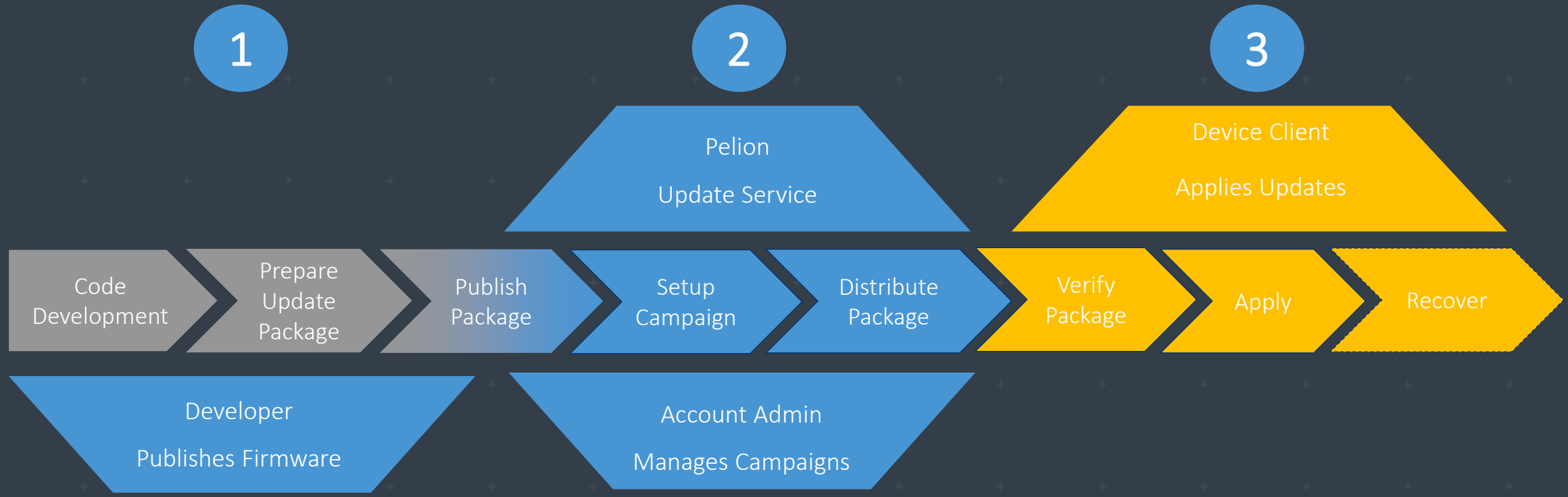
Network efficient - from web app to IoT nodes



In-field secure software update is crucial for IoT success



Pelion supports secure software update at IoT scale



Software Developer

PELION

Device Admin/ Service Manager

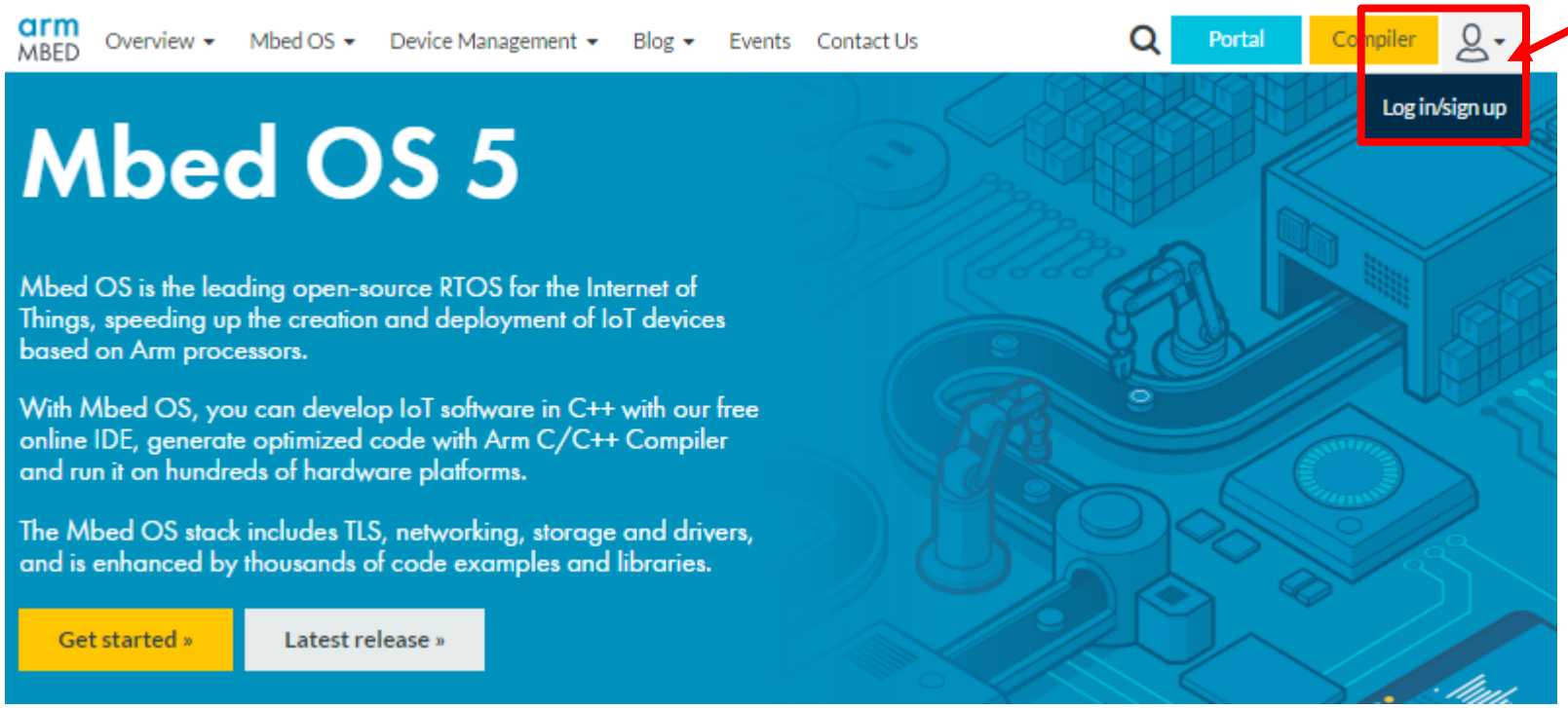
Connected Device

Hands On!

Use the Mbed Online compiler to build and download an application

Blinky (1)

- Point your browser to: https://os.mbed.com
- Click log in/Sign up – if you have credentials login, if you don't sign up



Blinky (2)

- Navigate to https://os.mbed.com/platforms/
- Search for the board you are going to use – in this case “LPCXpresso54608”
- Click on it

Blinky (3)

- Click on “Add to your Mbed Compiler”

LPCXpresso54608

LPCXpressoV3 development board for NXP LPC5460x processors

MCU Features

The LPC546xx MCU family combines the power efficiency of the Arm Cortex®-M4 core running at up to 220 MHz with multiple high-speed connectivity options, advanced timers, and analog features. DSP capabilities enable LPC546xx MCU devices to support complex algorithms in data-intensive application. Providing flexibility with up to 512 KB Flash and external memory interfaces, this family provides the ability to adapt as requirements change. Flash options support large, flexible internal and external memory configurations. Compatibility within the LPC54000 series enables the LPC546xx MCU family to provide a

Table of Contents

Board Partner

NXP

NXP is a leading semiconductor company founded by Philips more than 50 years ago.

Add to your Mbed Compiler

Buy Now

Follow

Blinky (4)

- Click ”Open Mbed Compiler”

LPCXpresso54608

LPCXpressoV3 development board for NXP LPC5460x processors

MCU Features

The LPC546xx MCU family combines the power efficiency of the Arm Cortex®-M4 core running at up to 220 MHz with multiple high-speed connectivity options, advanced timers, and analog features. DSP capabilities enable LPC546xx MCU devices to support complex algorithms in data-intensive application. Providing flexibility with up to 512 KB Flash and external memory interfaces, this family provides the ability to adapt as requirements change. Flash options support large, flexible internal and external memory configurations. Compatibility within the LPC54000 series enables the LPC546xx MCU family to provide a

Table of Contents

Board Partner

NXP

NXP is a leading semiconductor company founded by Philips more than 50 years ago.

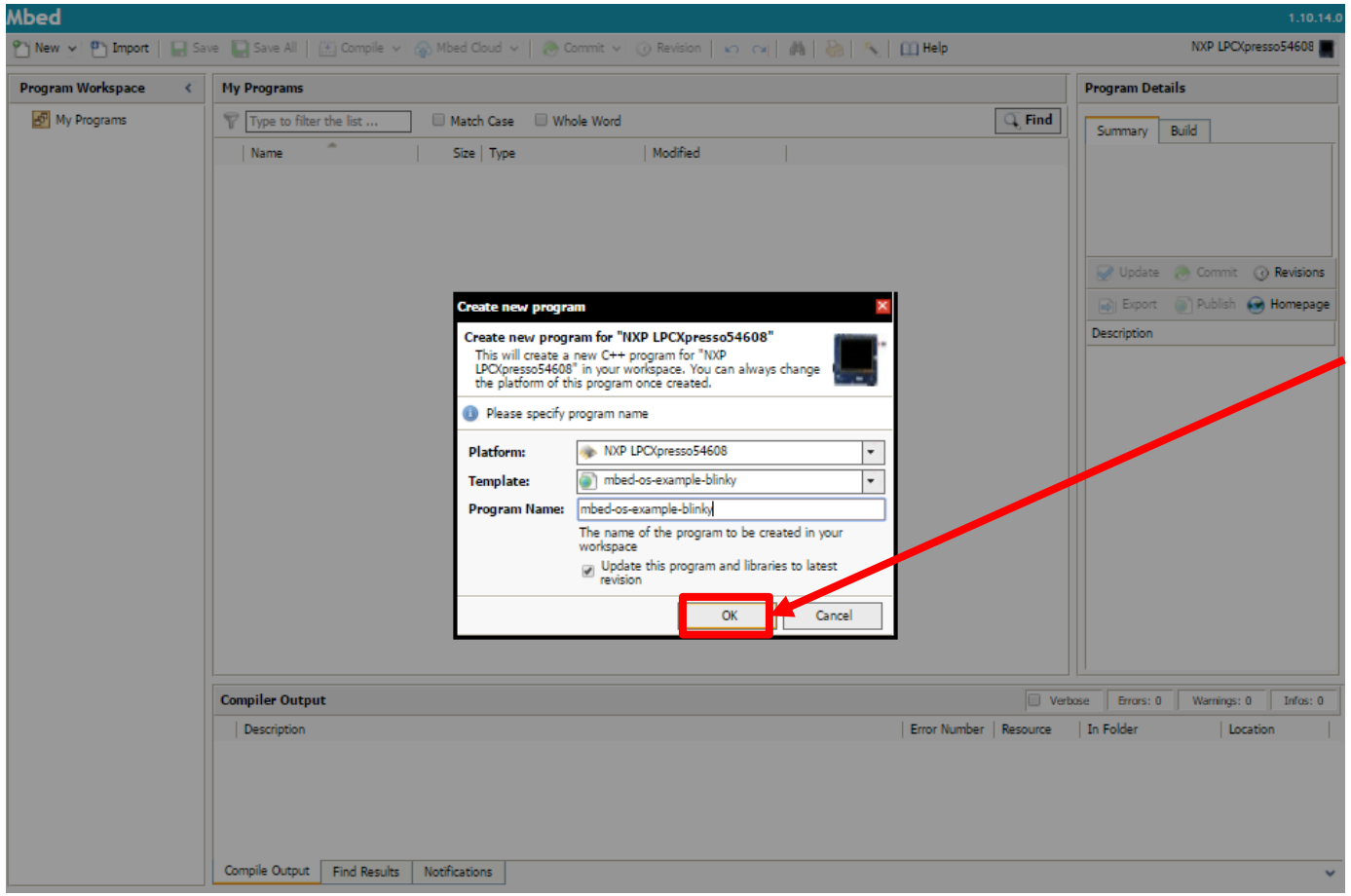
</> Open Mbed Compiler

Buy Now

+ Follow

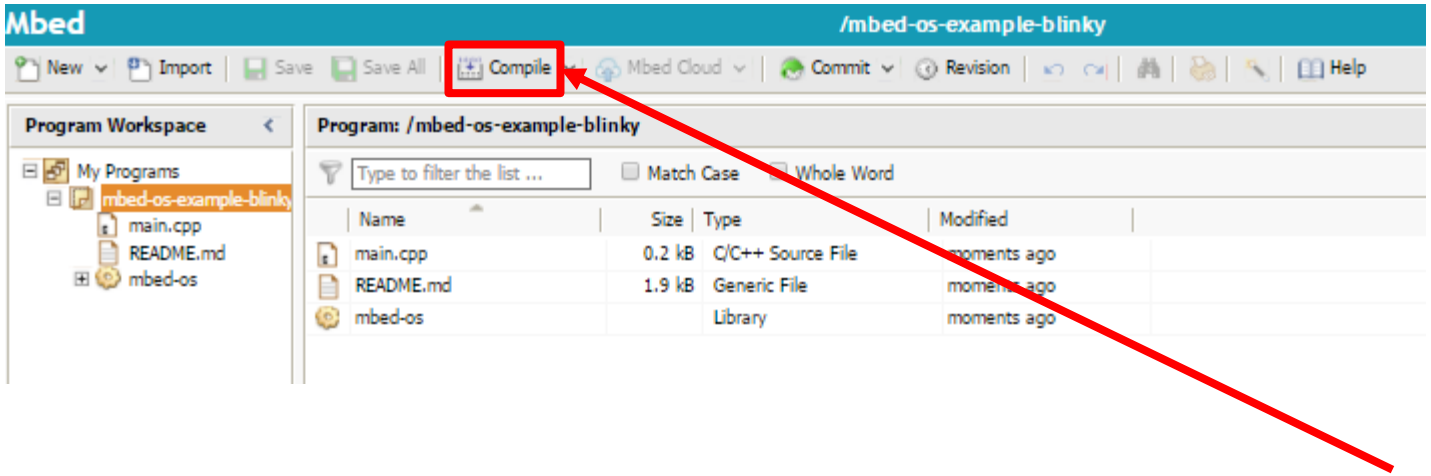
Blinky (5)

- The site will automatically offer to create a new program.
- Click ”OK” to create the Blinky program



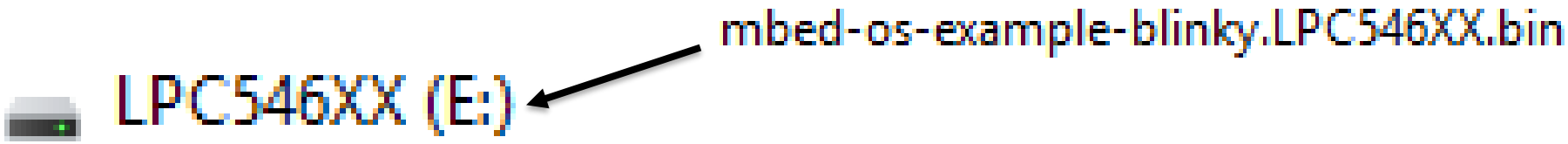
Blinky (6)

- Click “Compile”
- This will create and download a bin file to your downloads folder

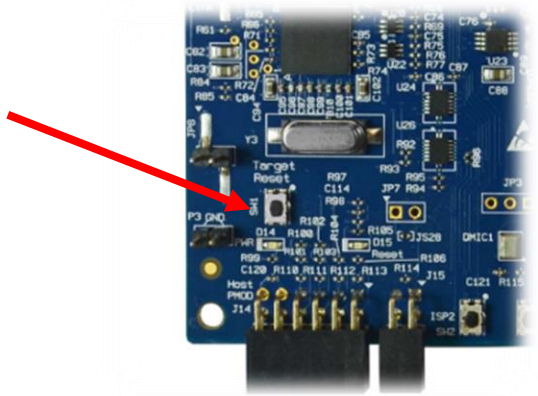


Blinky (7)

- Drag the downloaded file onto the LPC546XX “drive”



- Reset the board by clicking the reset button – [SW1]



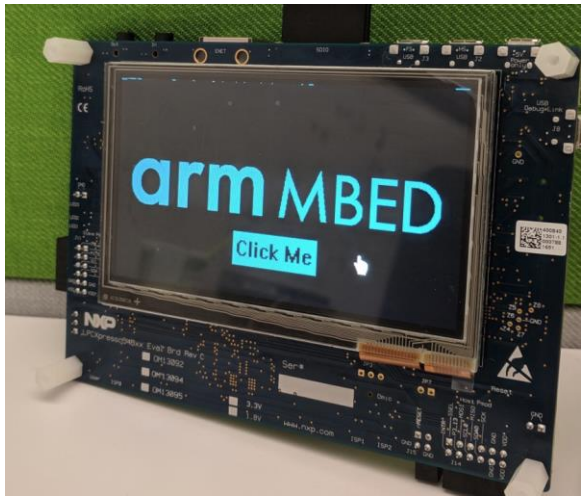
- Congrats – Blinky is working

Want to go further?

- Import an example that uses the LCD

  https://os.mbed.com/users/jplunkett/code/LPCXpresso54608-Touch-Cursor-Example/

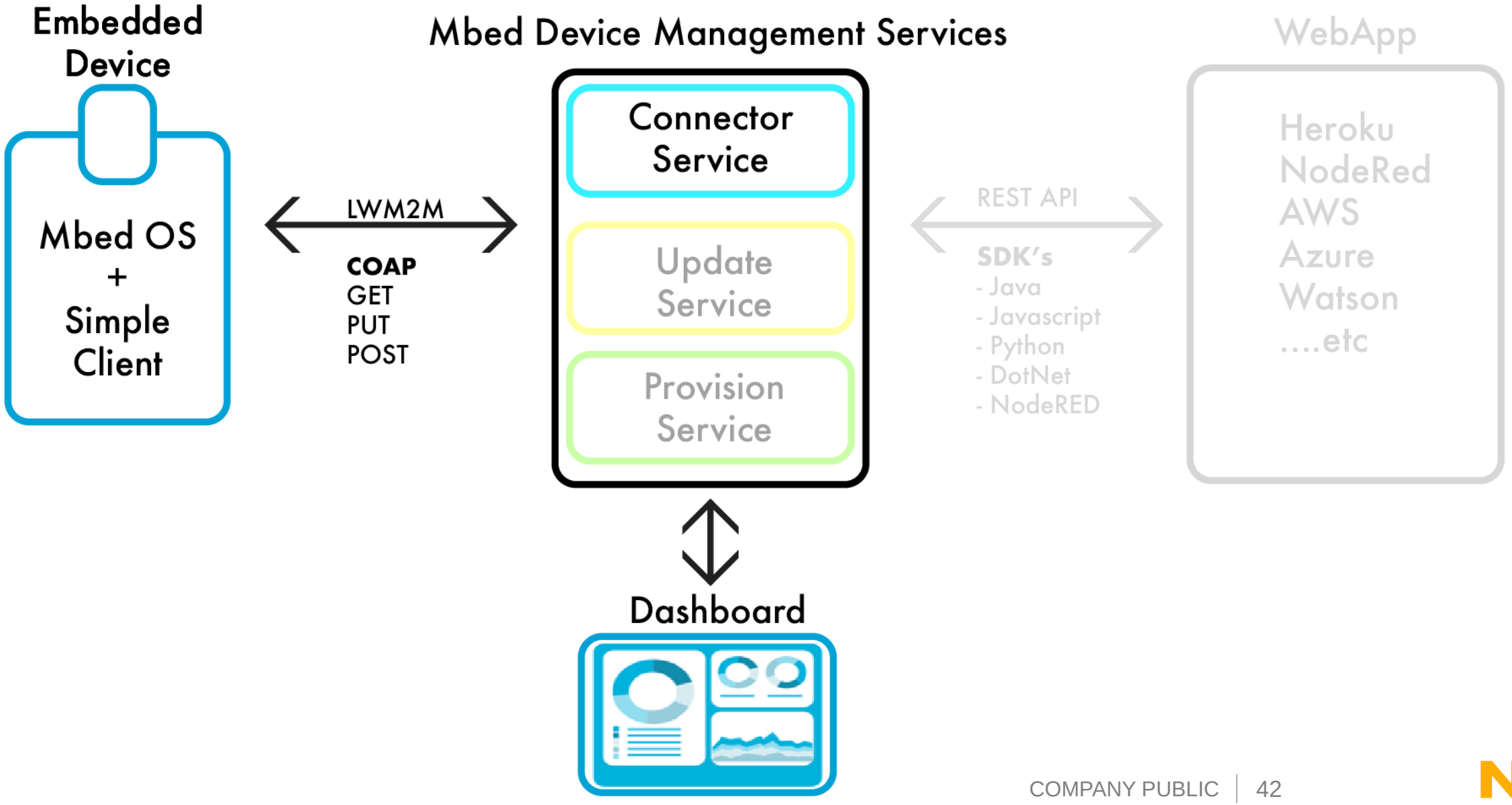
  (Skip ahead to see import instructions on the next few slides)
- Compile
- Download



Pelion Hands On

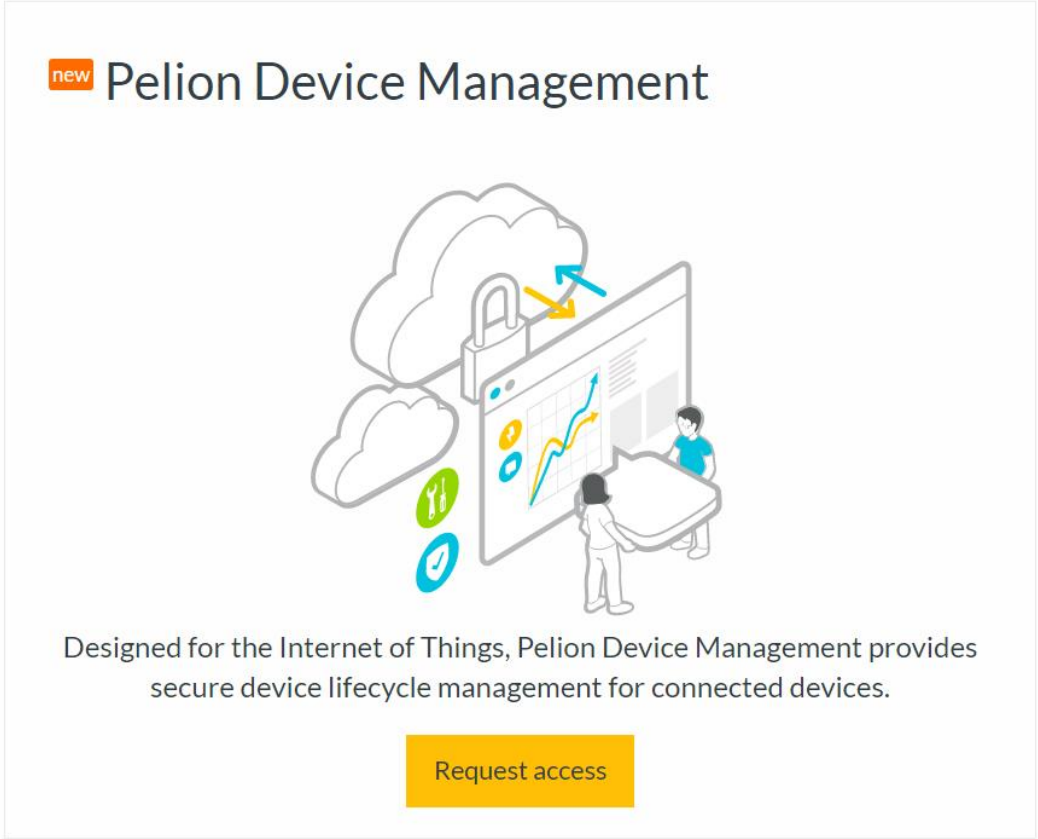
Use the Mbed Online compiler to build and download an application

Today's Focus - The Connector service



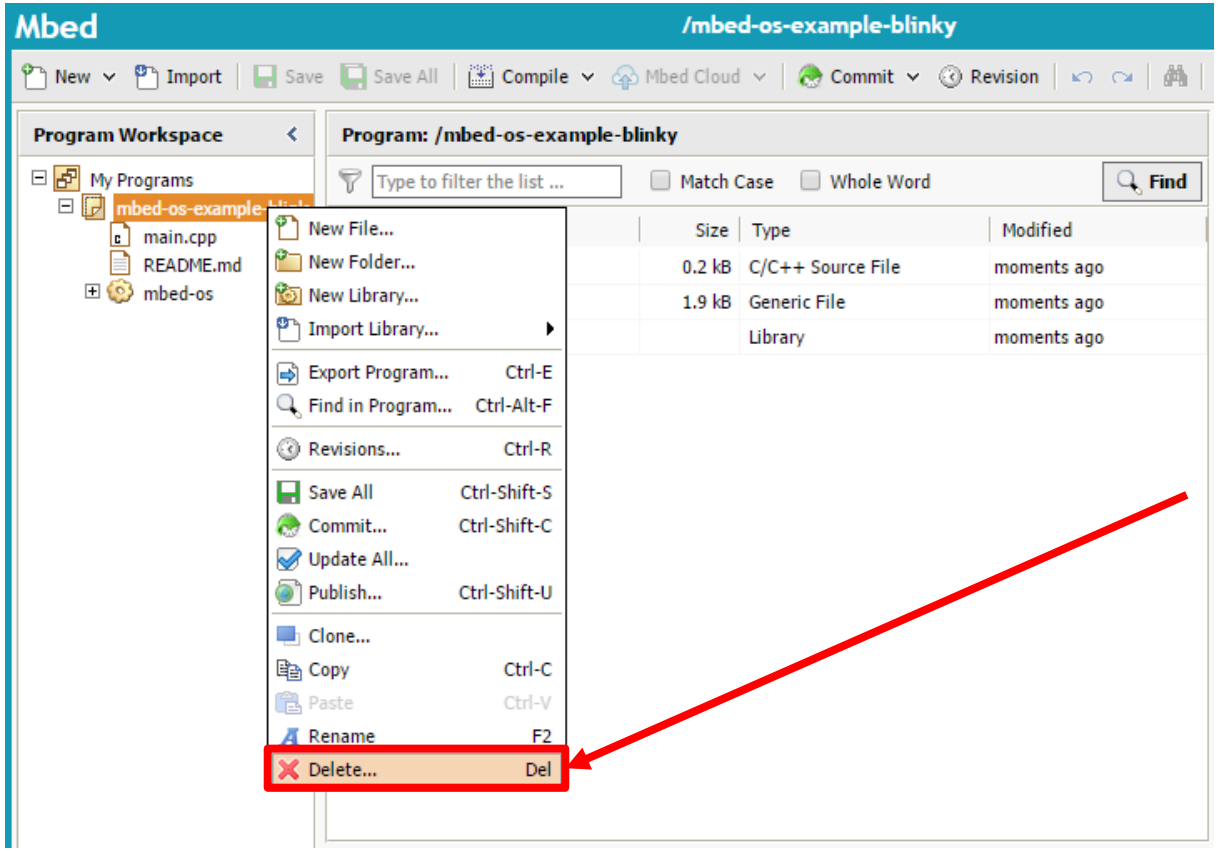
Request access to Pelion

- Go to https://console.mbed.com/
- Click Request access
- Give the workshop instructor your user name for expedited approval



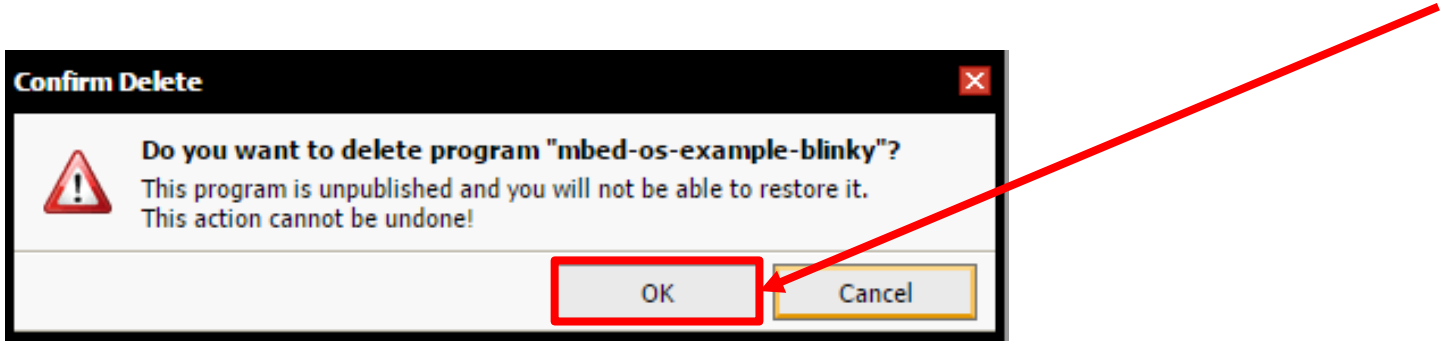
Mbed Cloud Client (1)

- Right click on the Blinky project folder
- Click delete



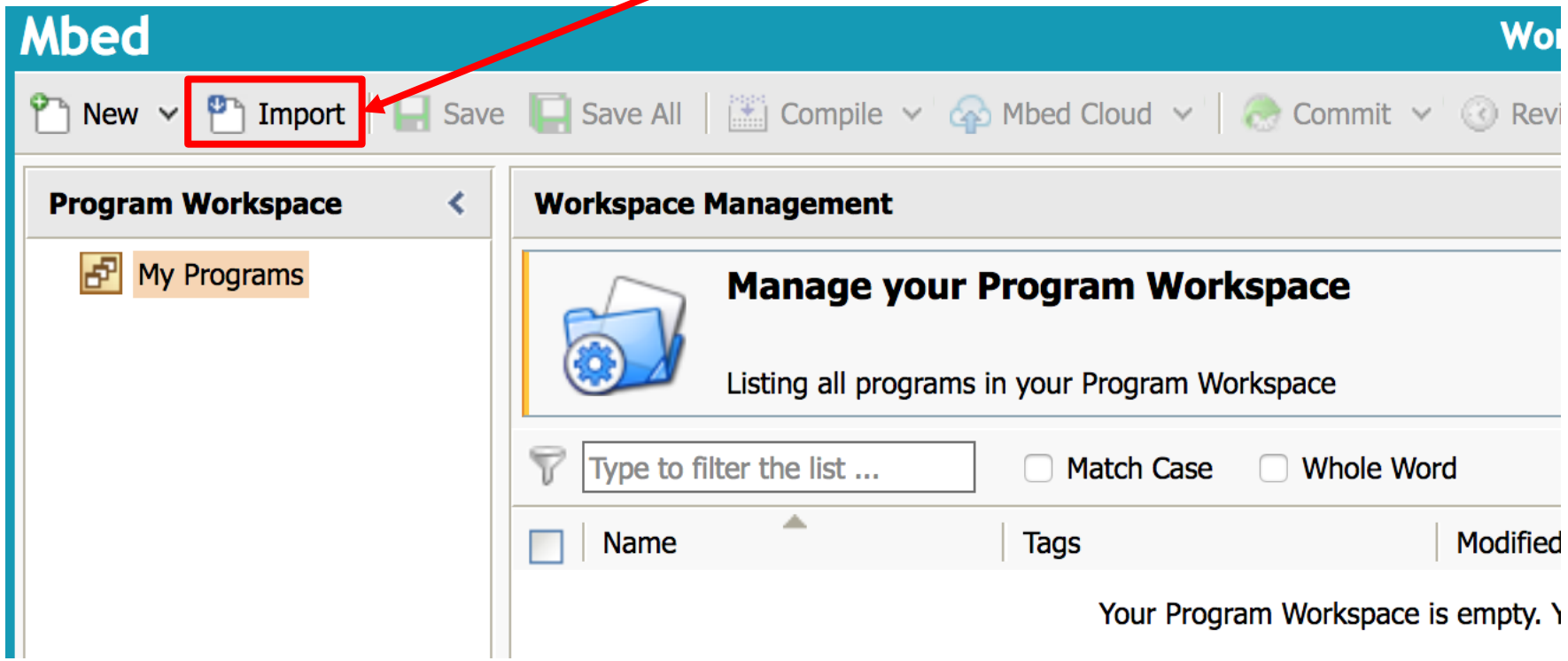
Mbed Cloud Client (2)

- Click “OK”



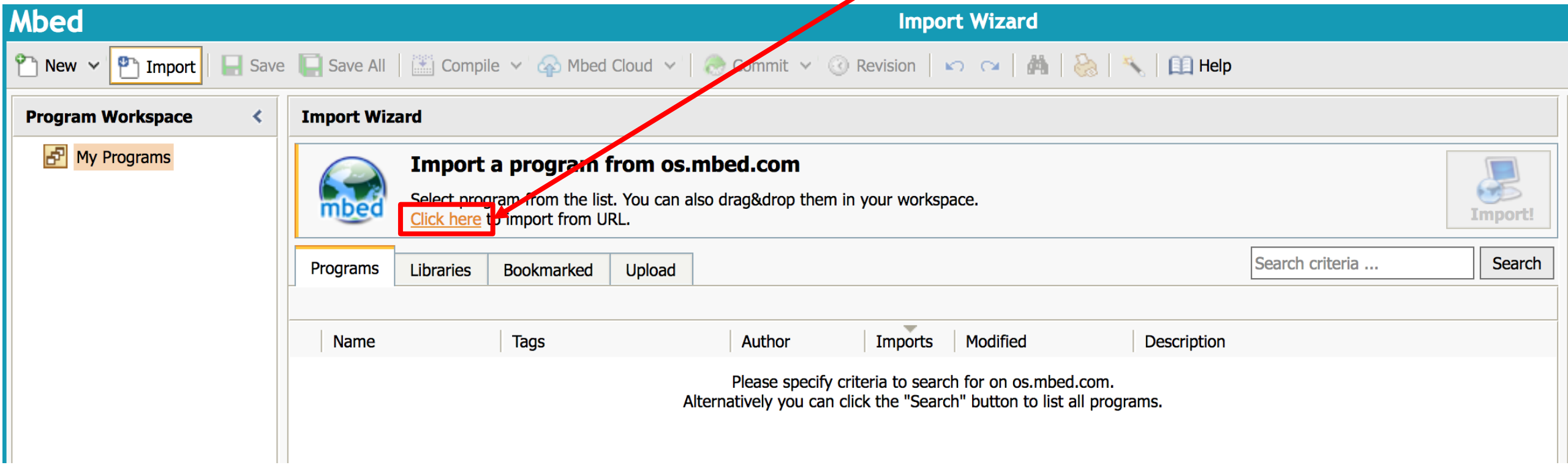
Mbed Cloud Client (3)

- Click “Import”



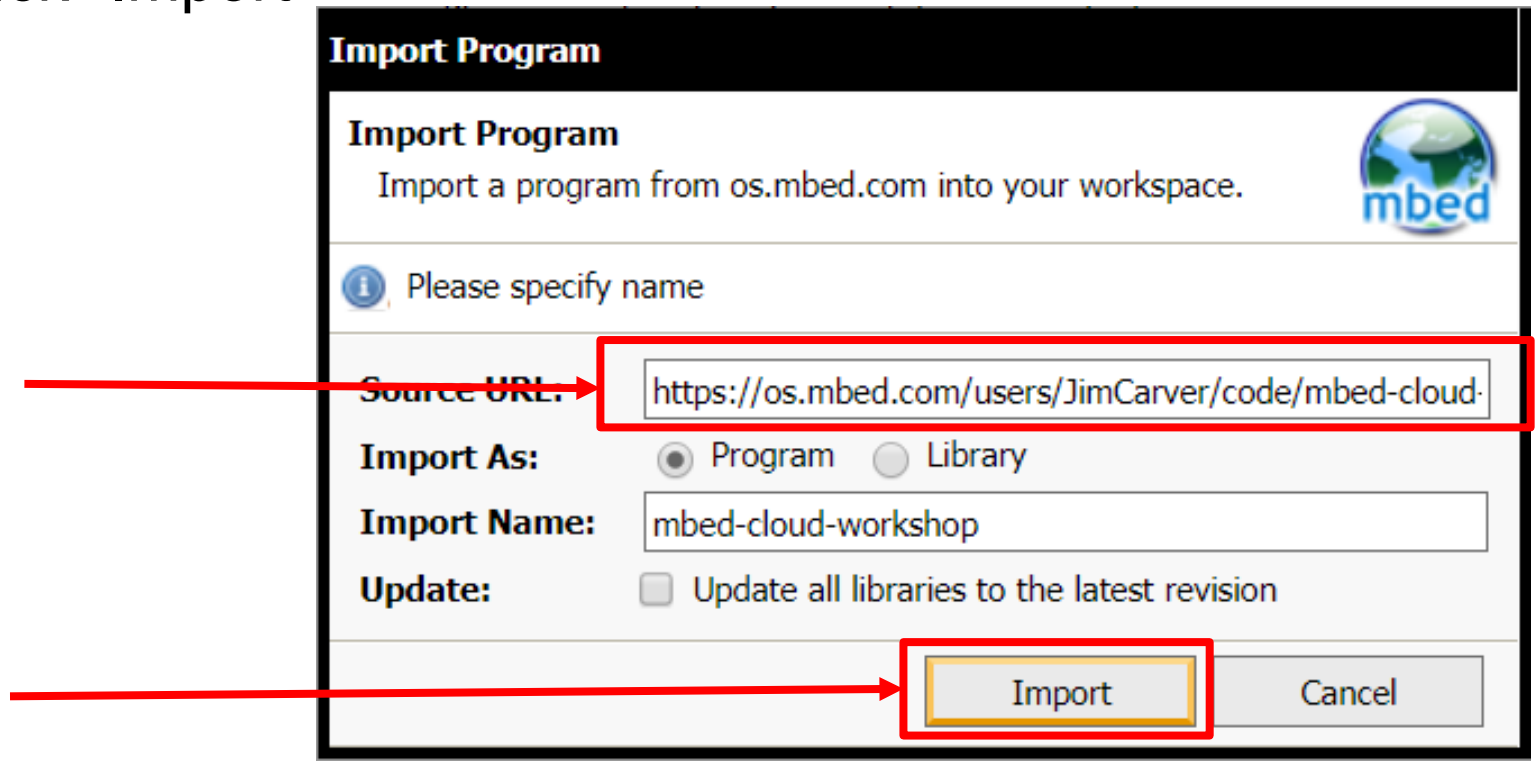
Mbed Cloud Client (4)

- Click the little orange link “Click here”



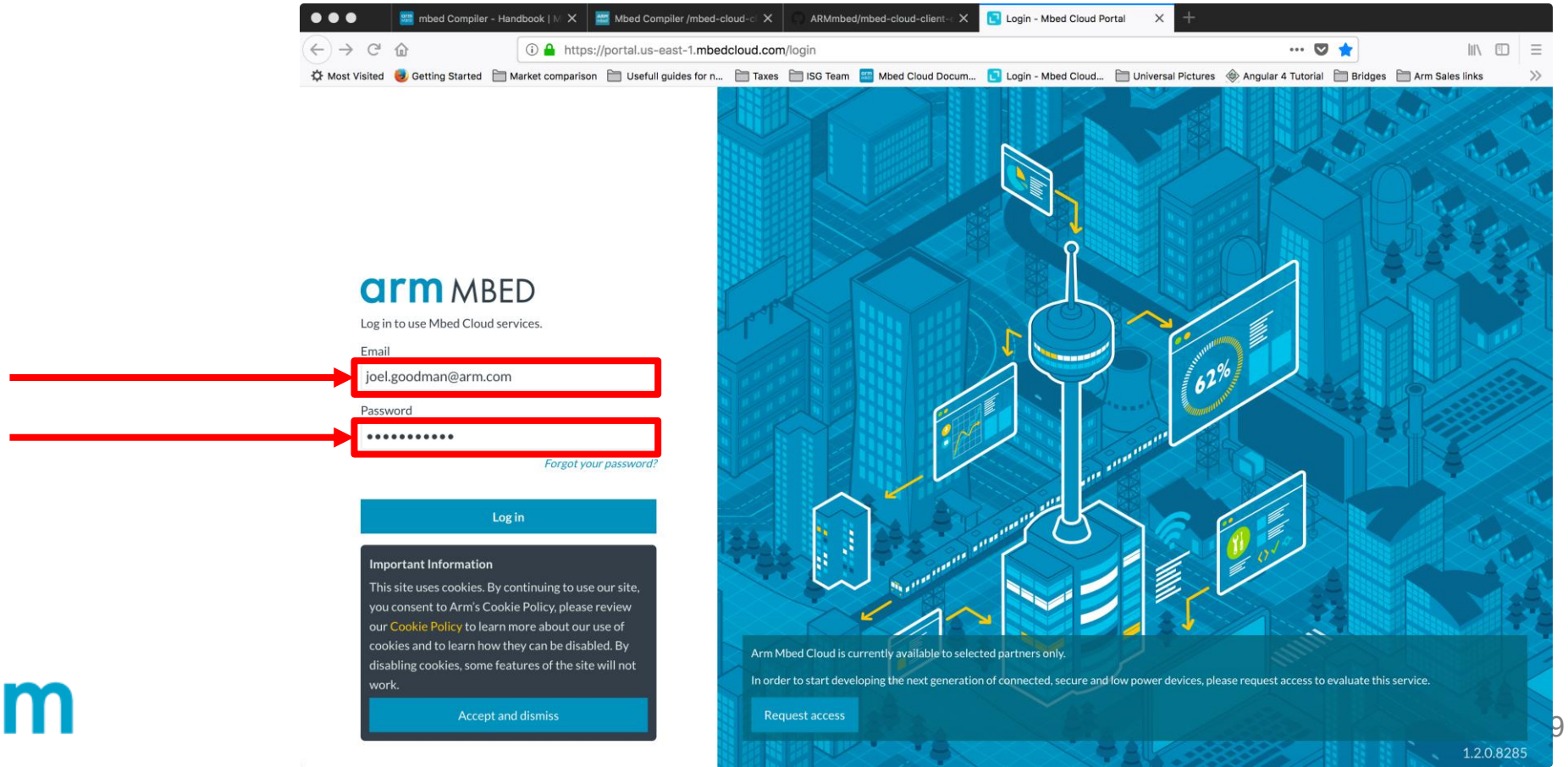
Mbed Cloud Client (5)

- In the ”Source URL” box paste the following link: https://os.mbed.com/users/maclobdell/code/mbed-cloud-example-lpc546xx/
- Click “Import”



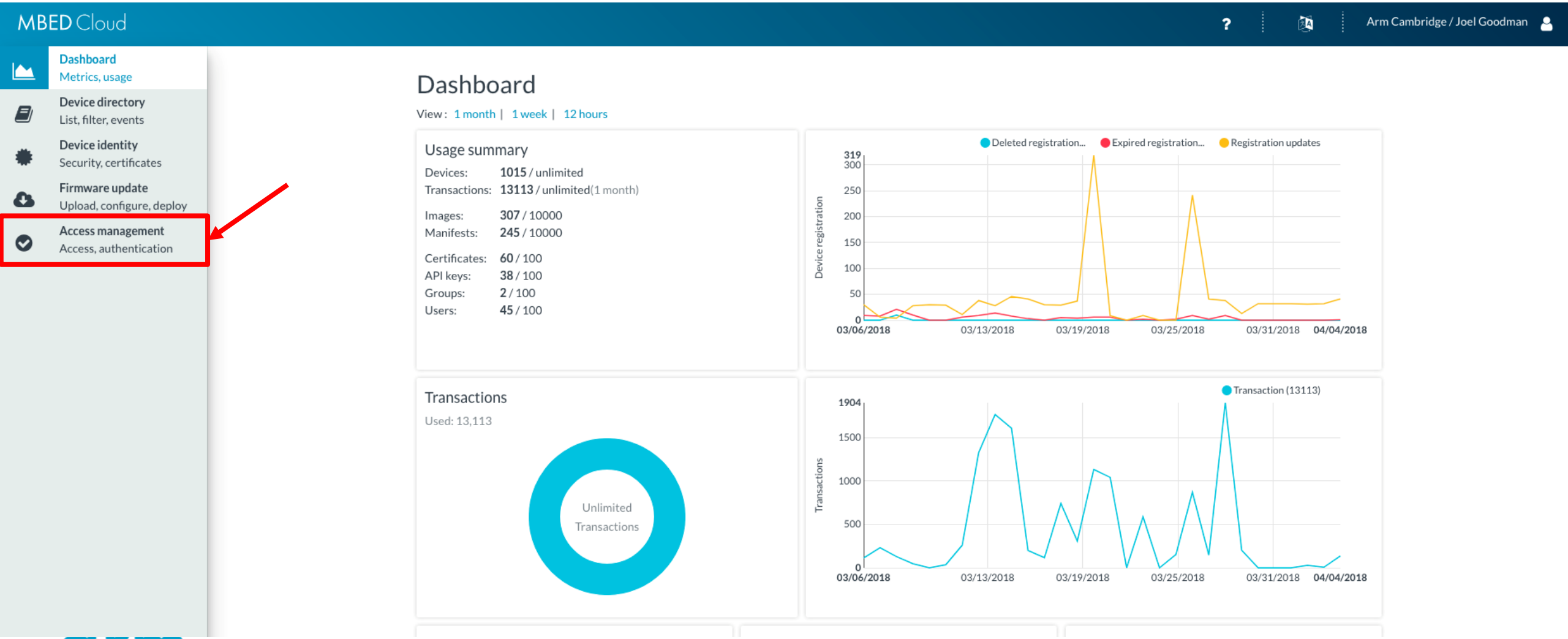
Mbed Cloud Client (6)

- Open a new tab
- Navigate to: https://portal.mbedcloud.com
- *If requested*, fill in your credentials and login



Mbed Cloud Client – Create and API Key (7)

- Click “Access management"

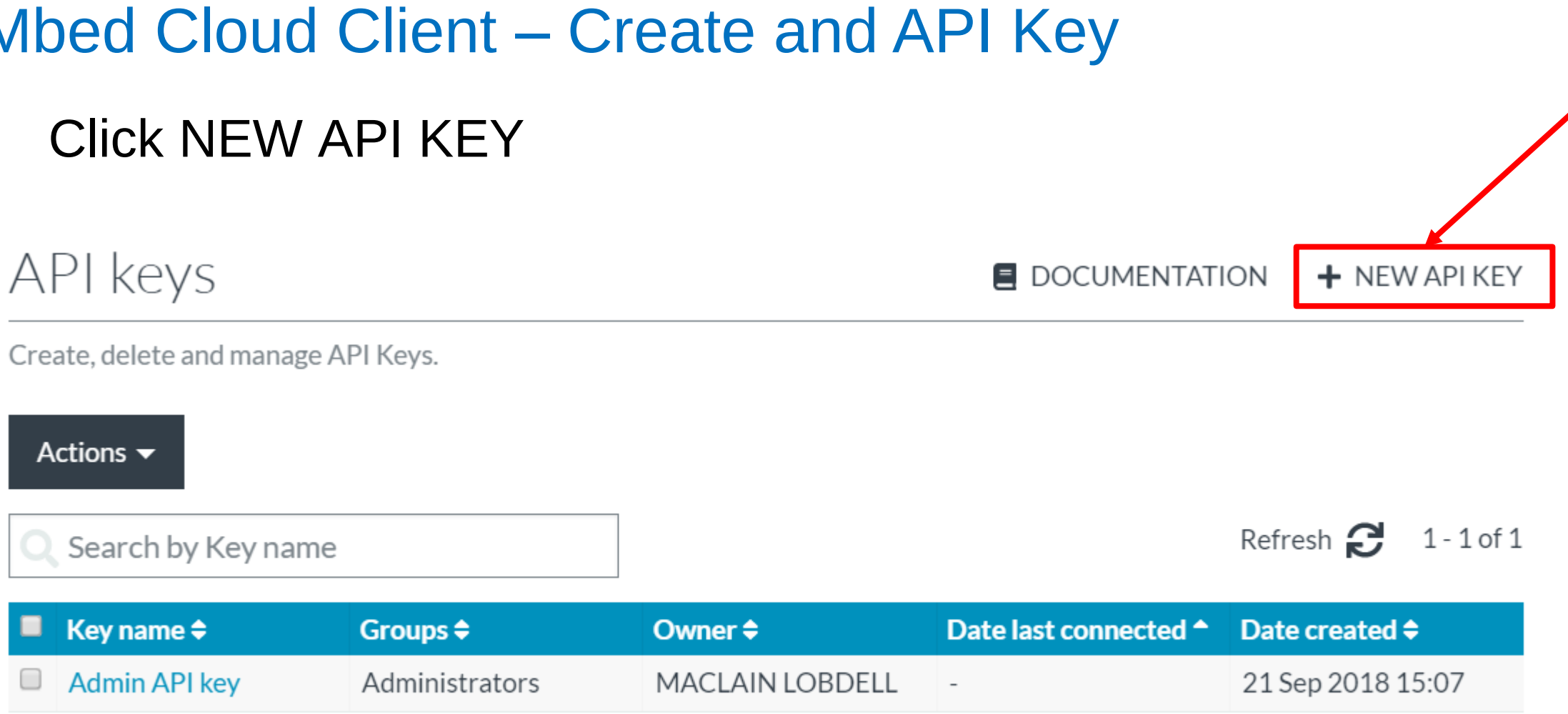


Mbed Cloud Client – Create and API Key (8)

- Click “API Keys"

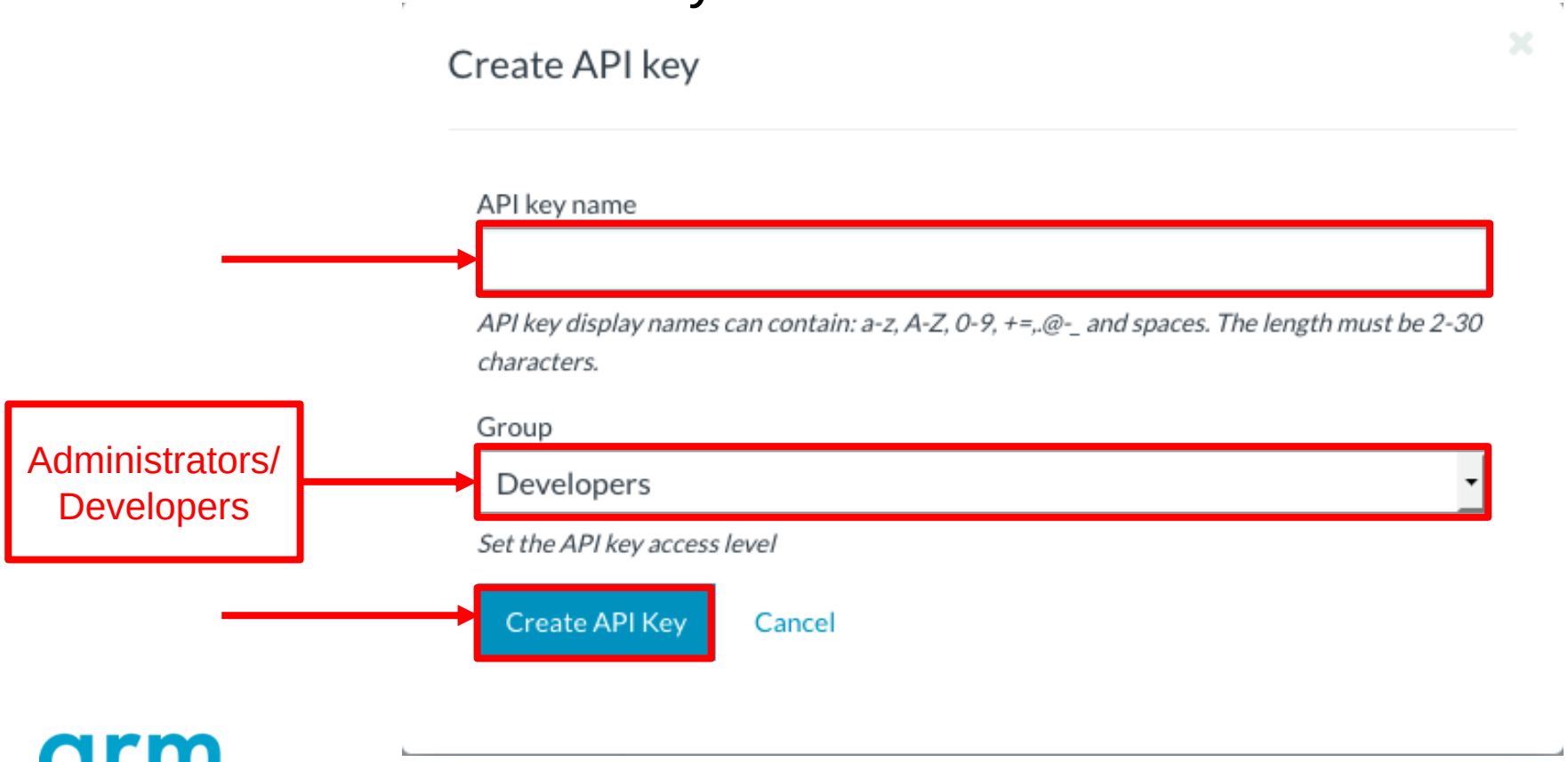
Mbed Cloud Client – Create and API Key

- Click NEW API KEY



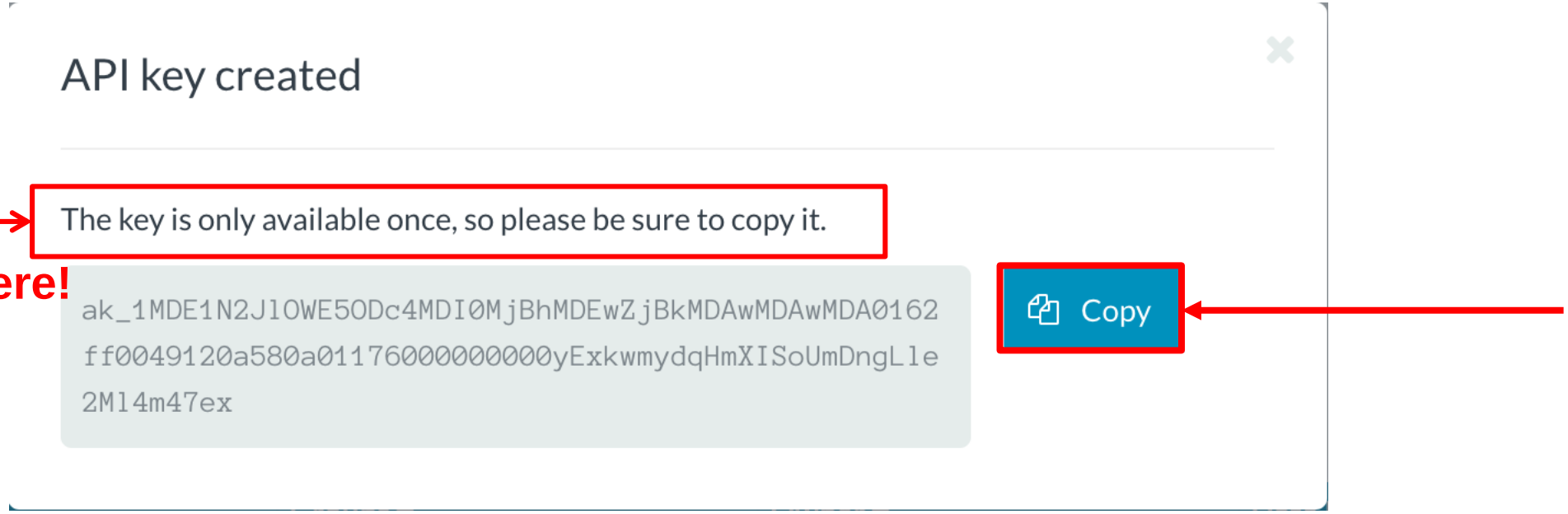
Mbed Cloud Client – Create and API Key (9)

- Give your API key a name you will recognize
- In the Group tab select ”Developers.
- Click Create API Key



Mbed Cloud Client – Create and API Key (10)

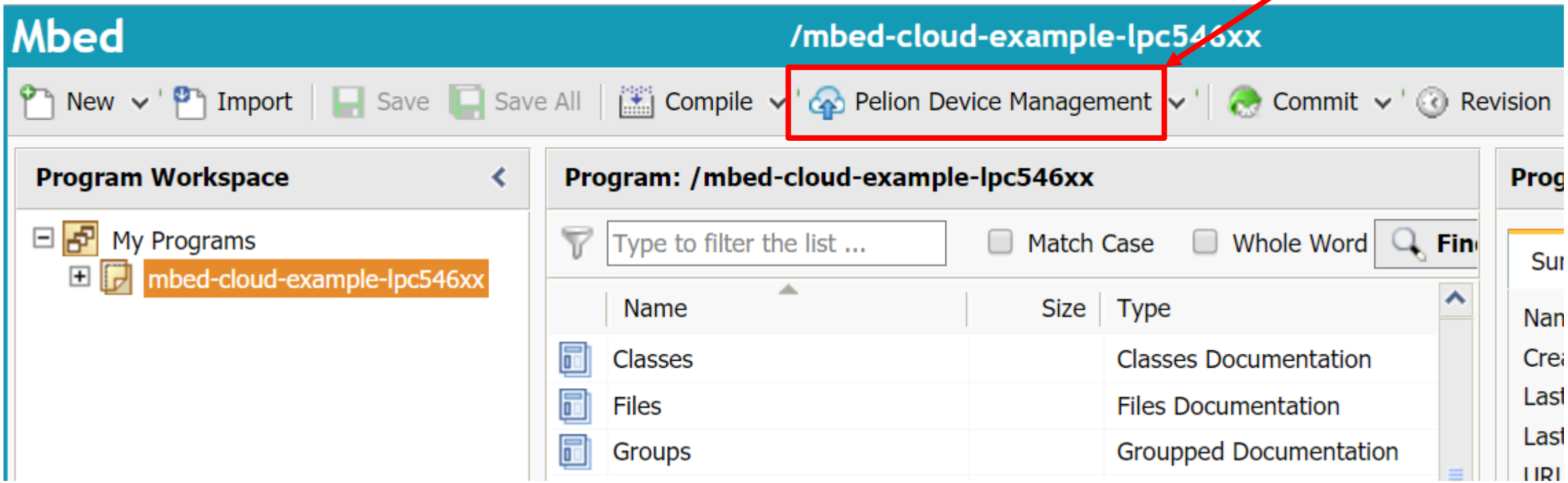
- Click “Copy”
- Save your API Key some where safe



Please Note
Save somewhere!

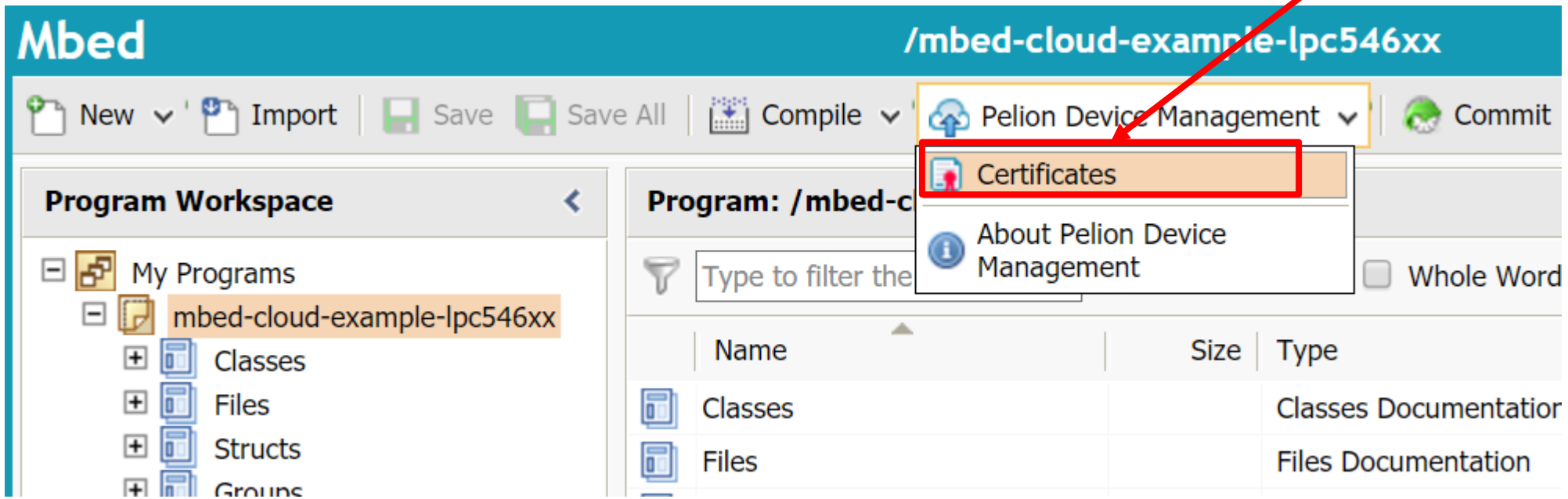
Mbed Cloud Client – Creating Device Certificate (11)

- Return to your online compiler tab
- Click on the ”Pelion Device Management” Button



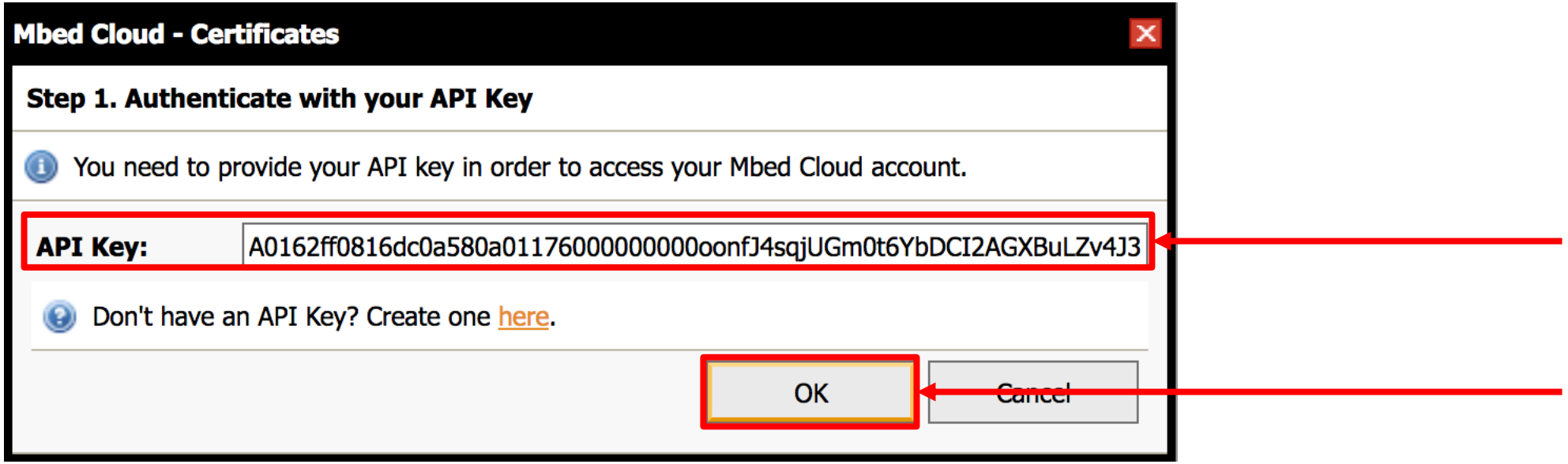
Mbed Cloud Client - Creating Device Certificate (12)

- Clock on “Certificates”



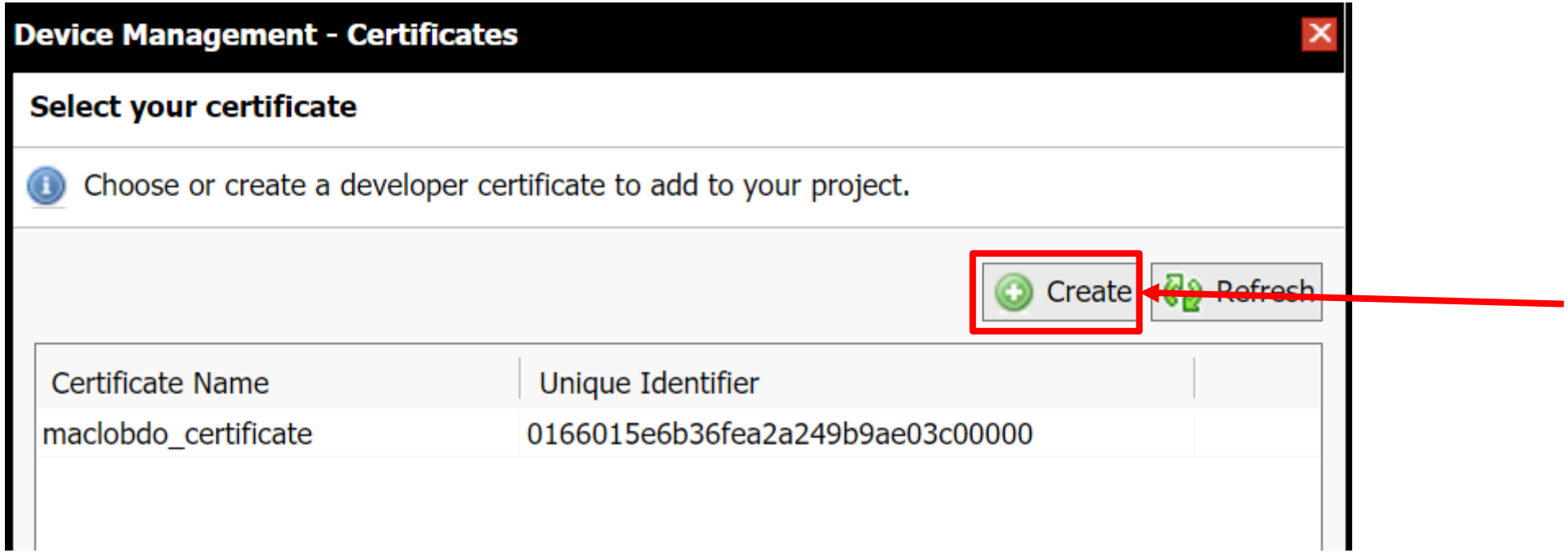
Mbed Cloud Client - Creating Device Certificate (13)

- Paste your API Key
- Click “OK”



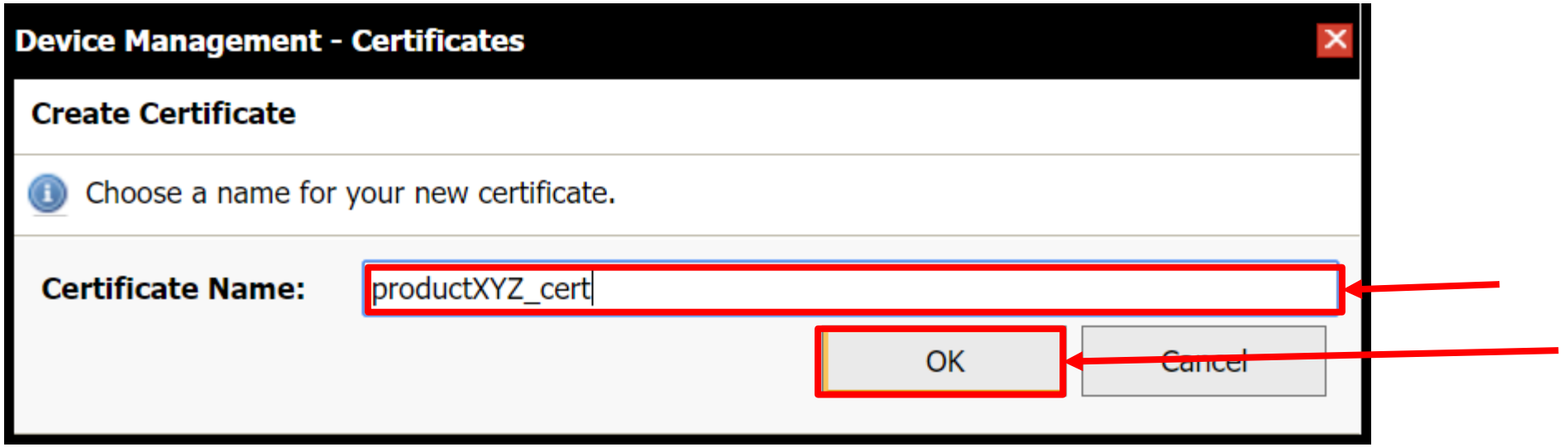
Mbed Cloud Client - Creating Device Certificate (14)

- Click “Create”



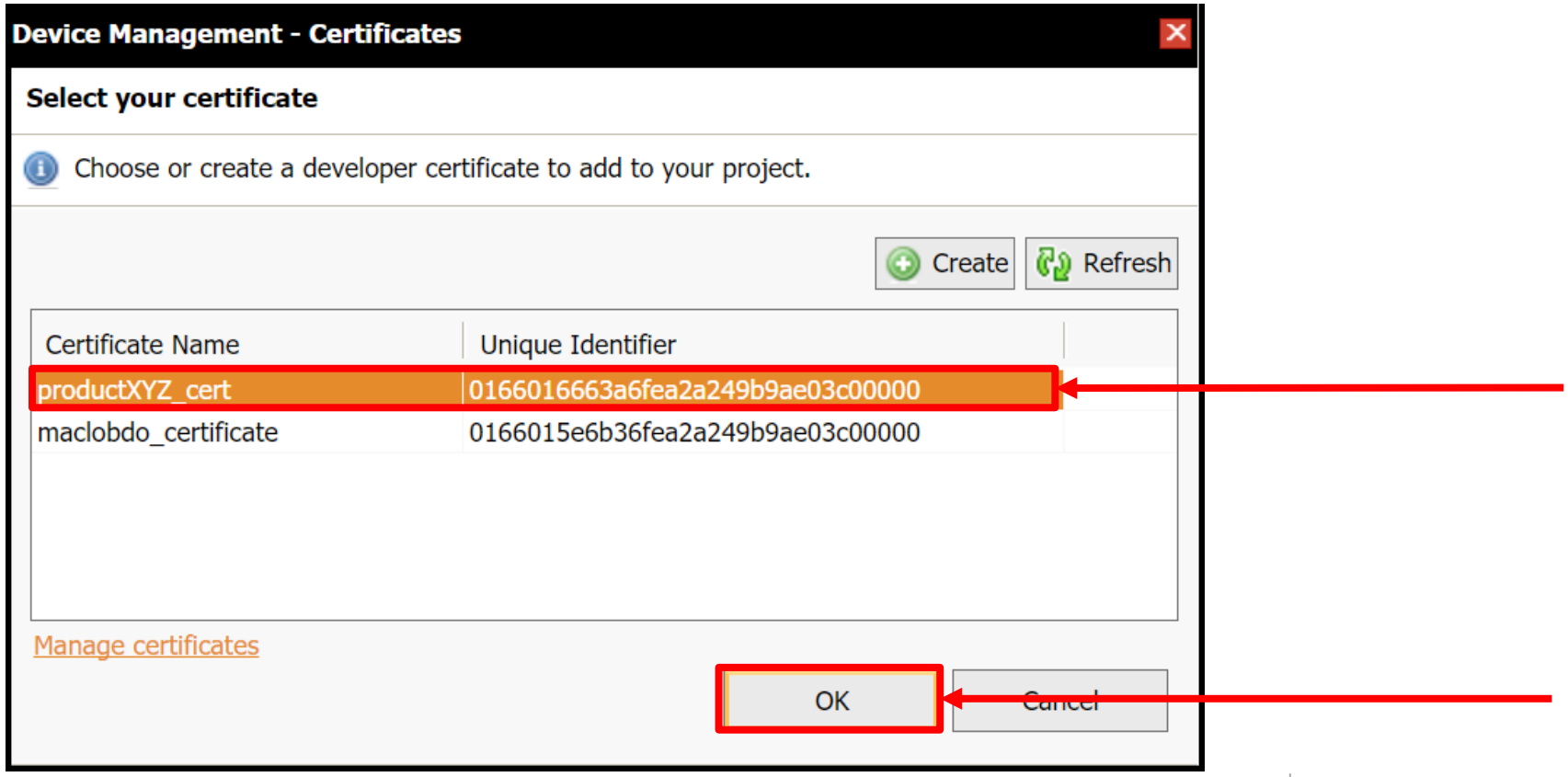
Mbed Cloud Client - Creating Device Certificate (15)

- Give your device certificate a name you will recognize
- Click ”OK”



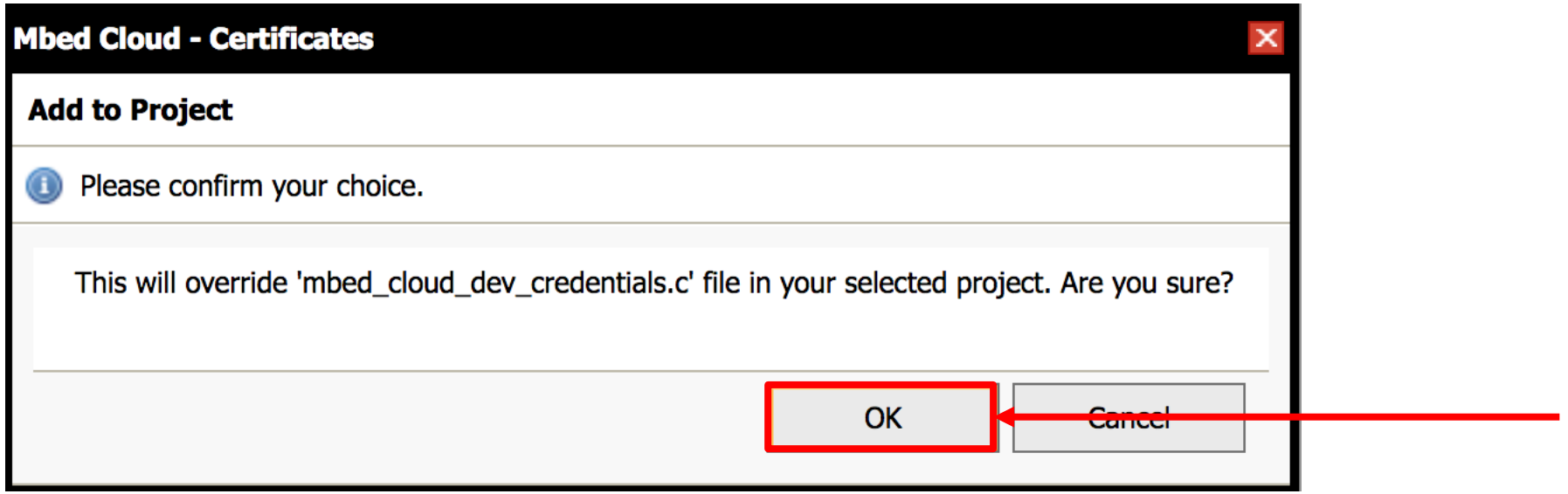
Mbed Cloud Client - Creating Device Certificate (16)

- Select your device certificate
- Click ”OK”



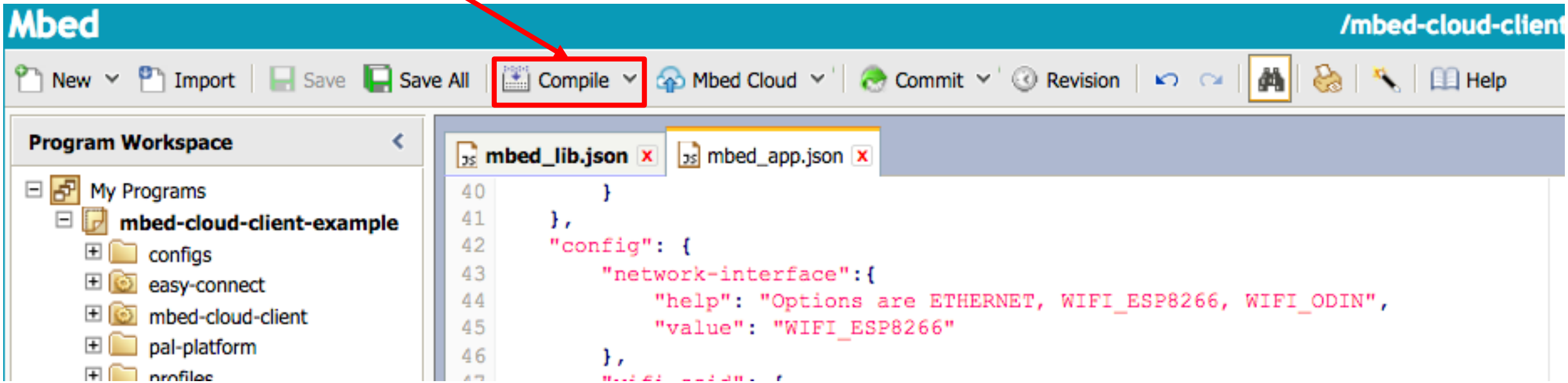
Mbed Cloud Client - Creating Device Certificate (17)

- Click “Ok” to override the ‘mbed_cloud_dev_credentials.c”



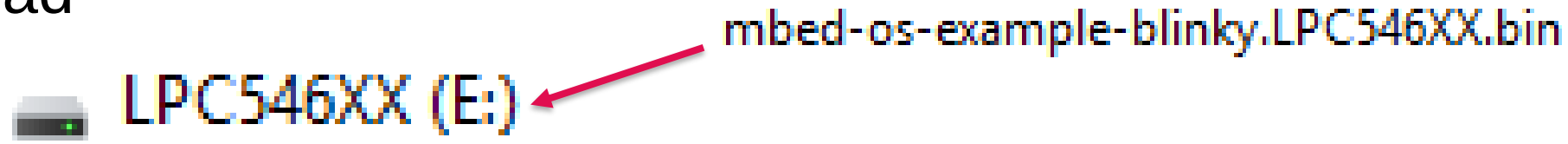
Mbed Cloud Client – Configuring the client to your device (21)

- Click “**Compile**”

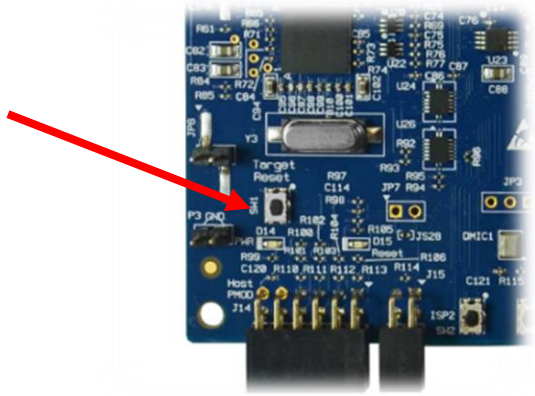


Mbed Cloud Client

1. Download

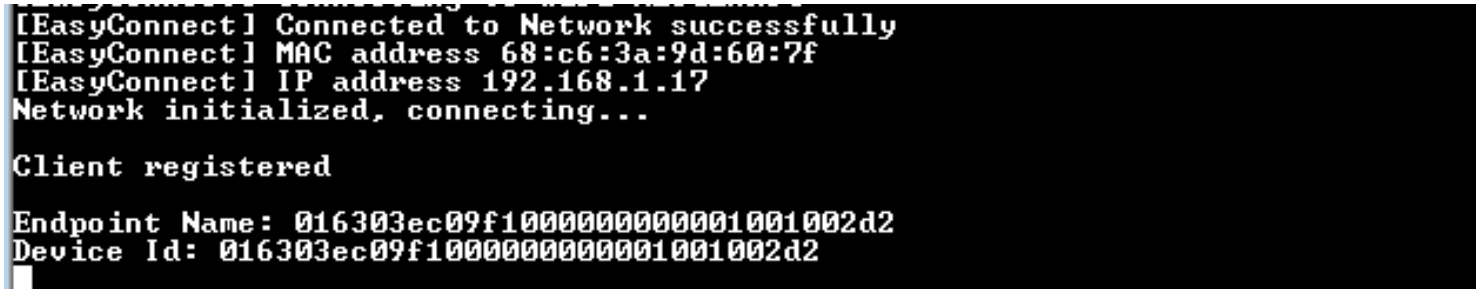


2. Reset



Mbed Cloud Client – Configuring the client to your device (23)

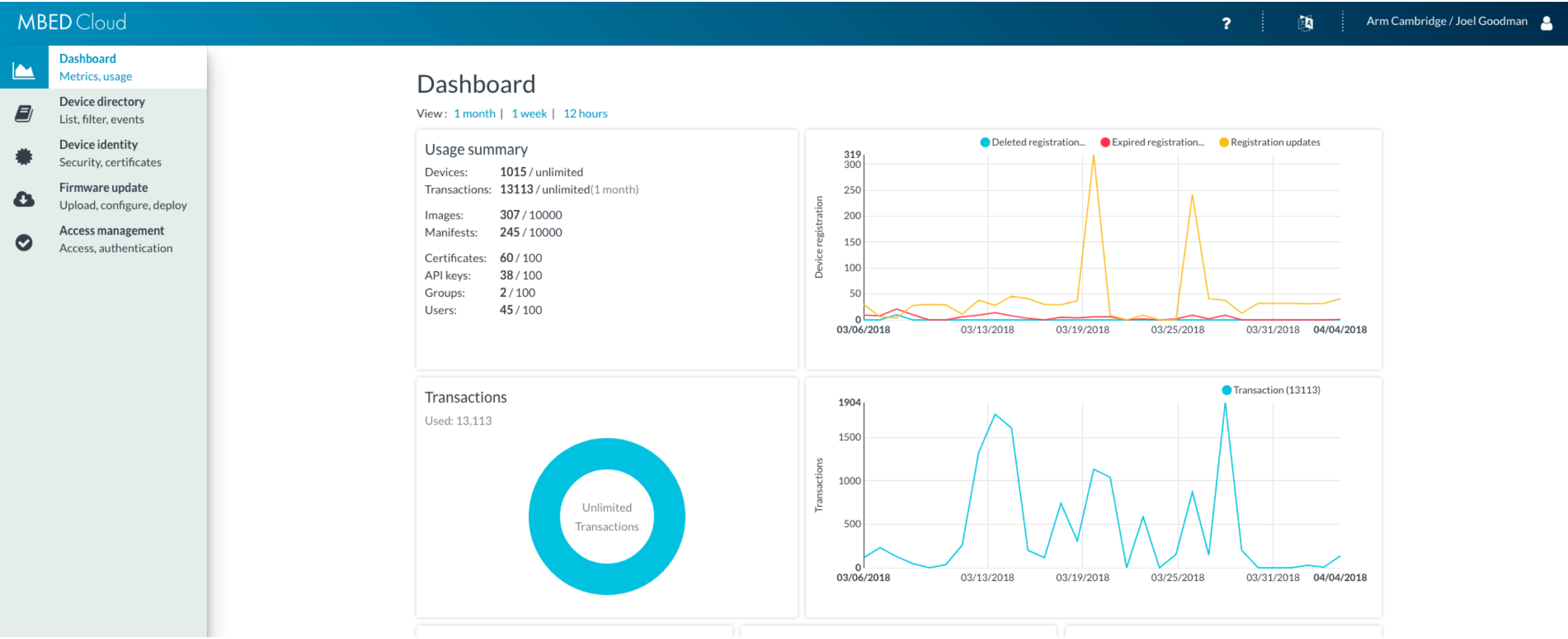
- Open your serial terminal (e.g. Tera Term) - > set the COM port to the port your board is connected to -> set baud rate to 115200 -> click “Connect”
- Once the device has connected to the network and successfully registered to Mbed Cloud you will see your device ID and device name printed to the serial terminal.



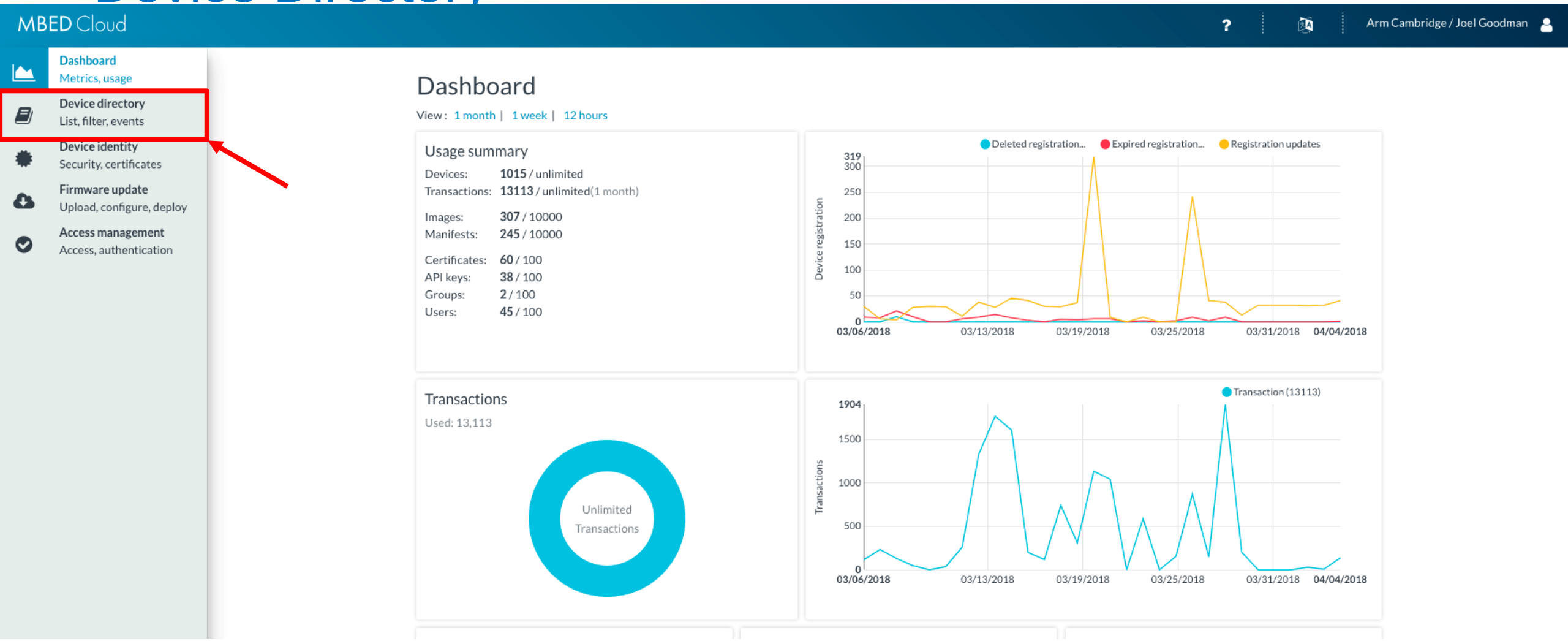
Pelion Portal

View your device data, control your device

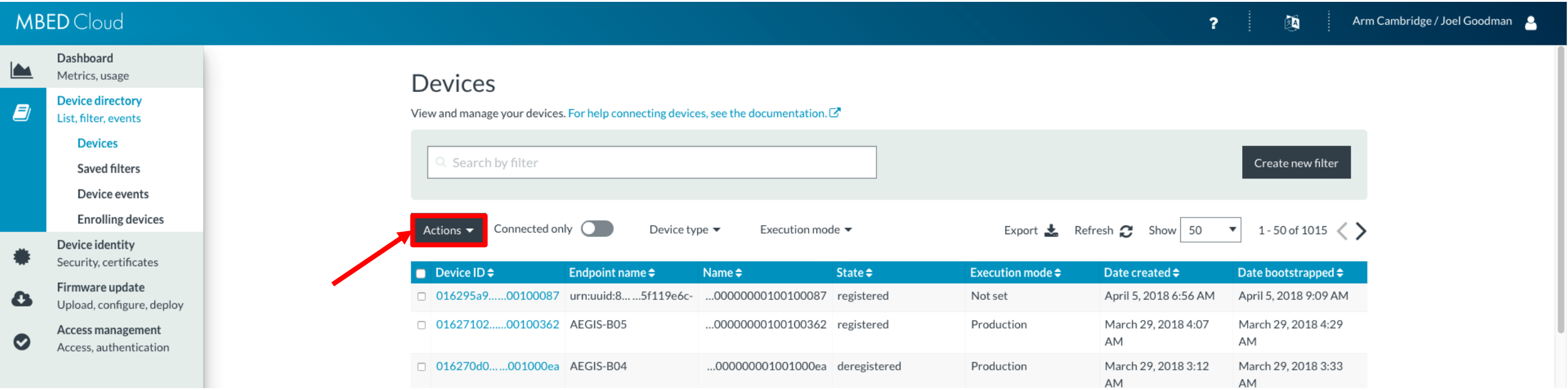
Back to the portal https://portal.mbedcloud.com



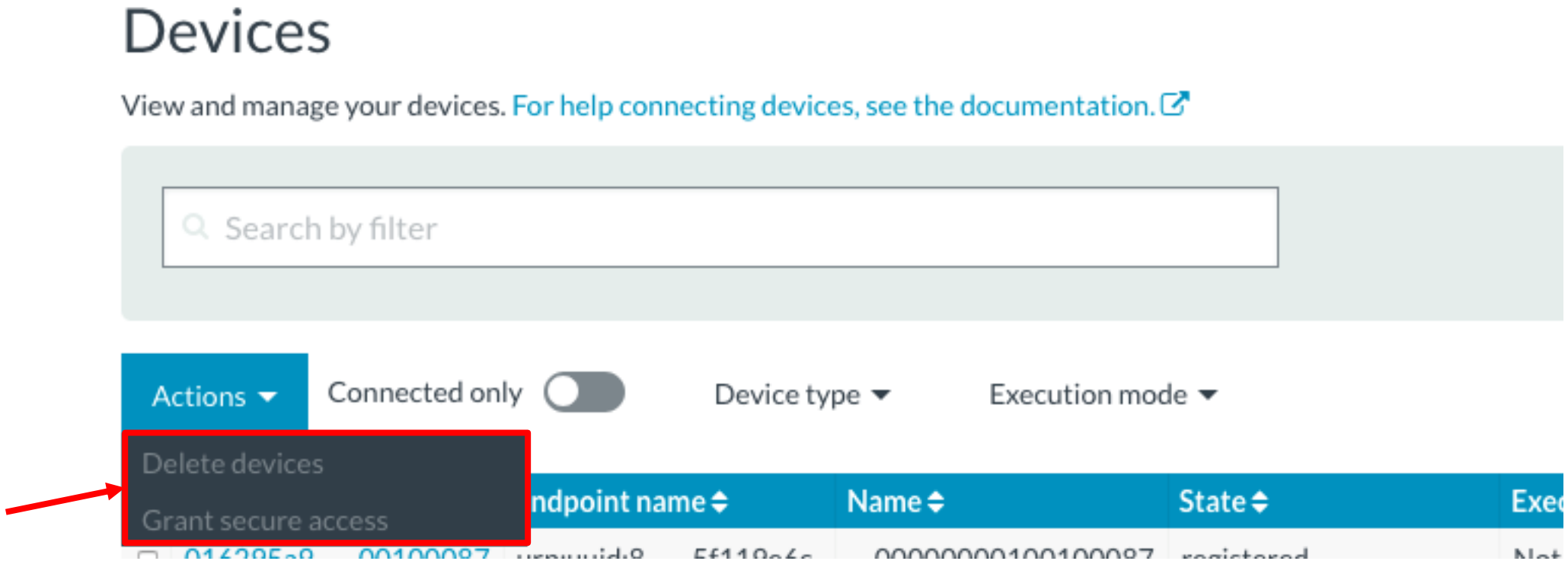
Device Directory



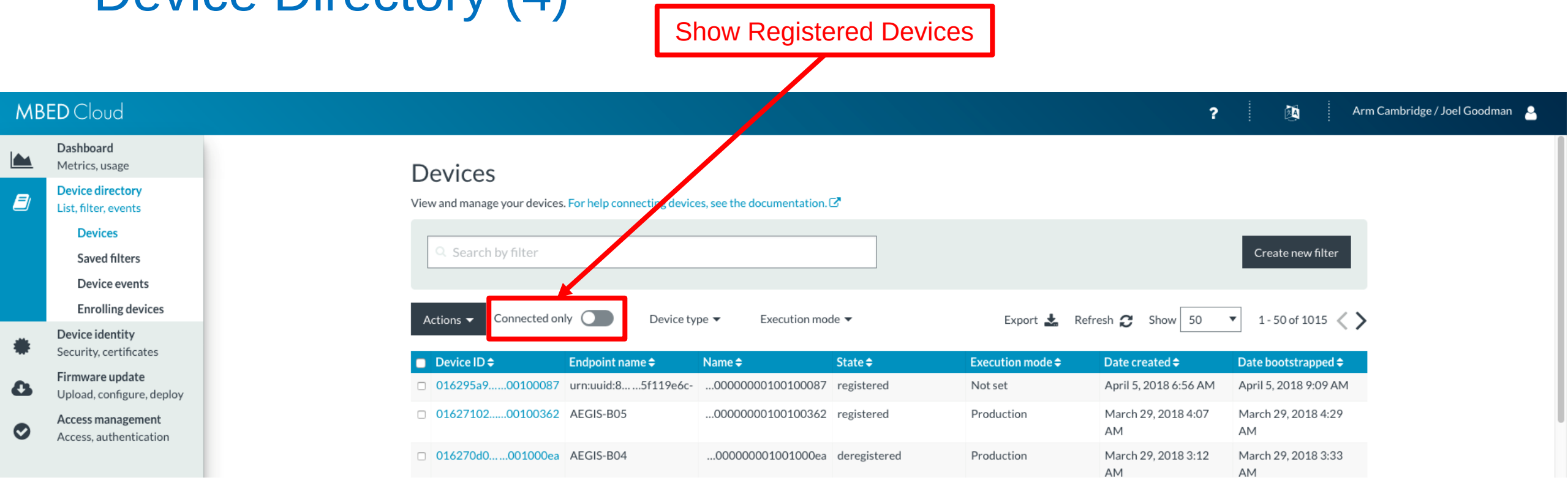
Device Directory (2)



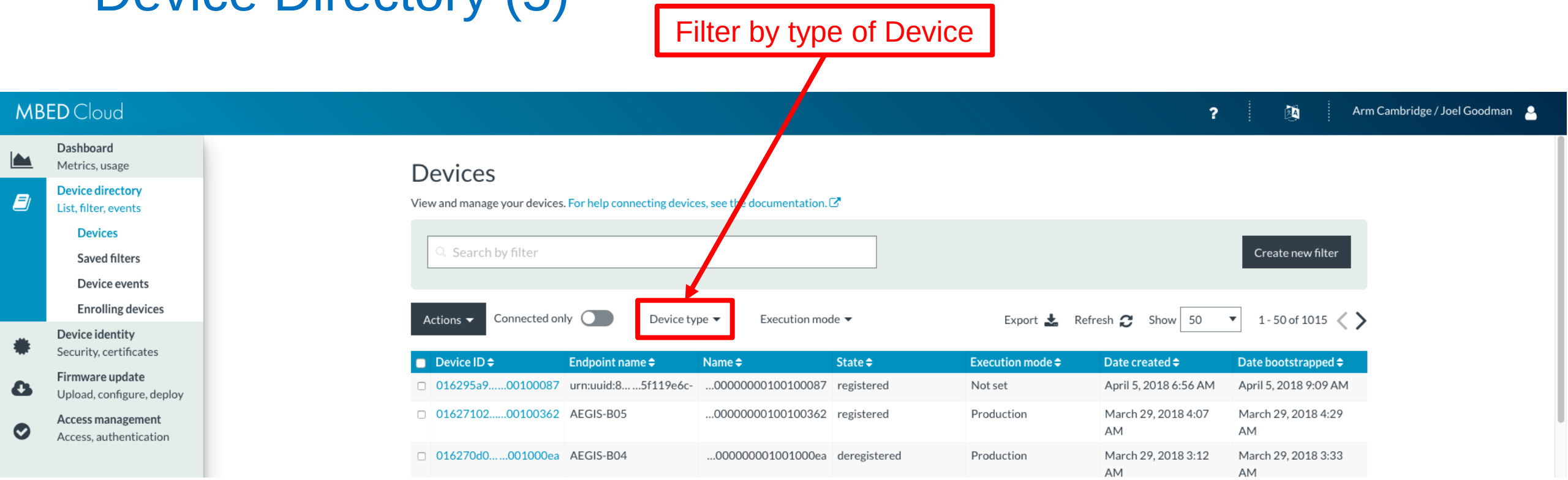
Device Directory (3)



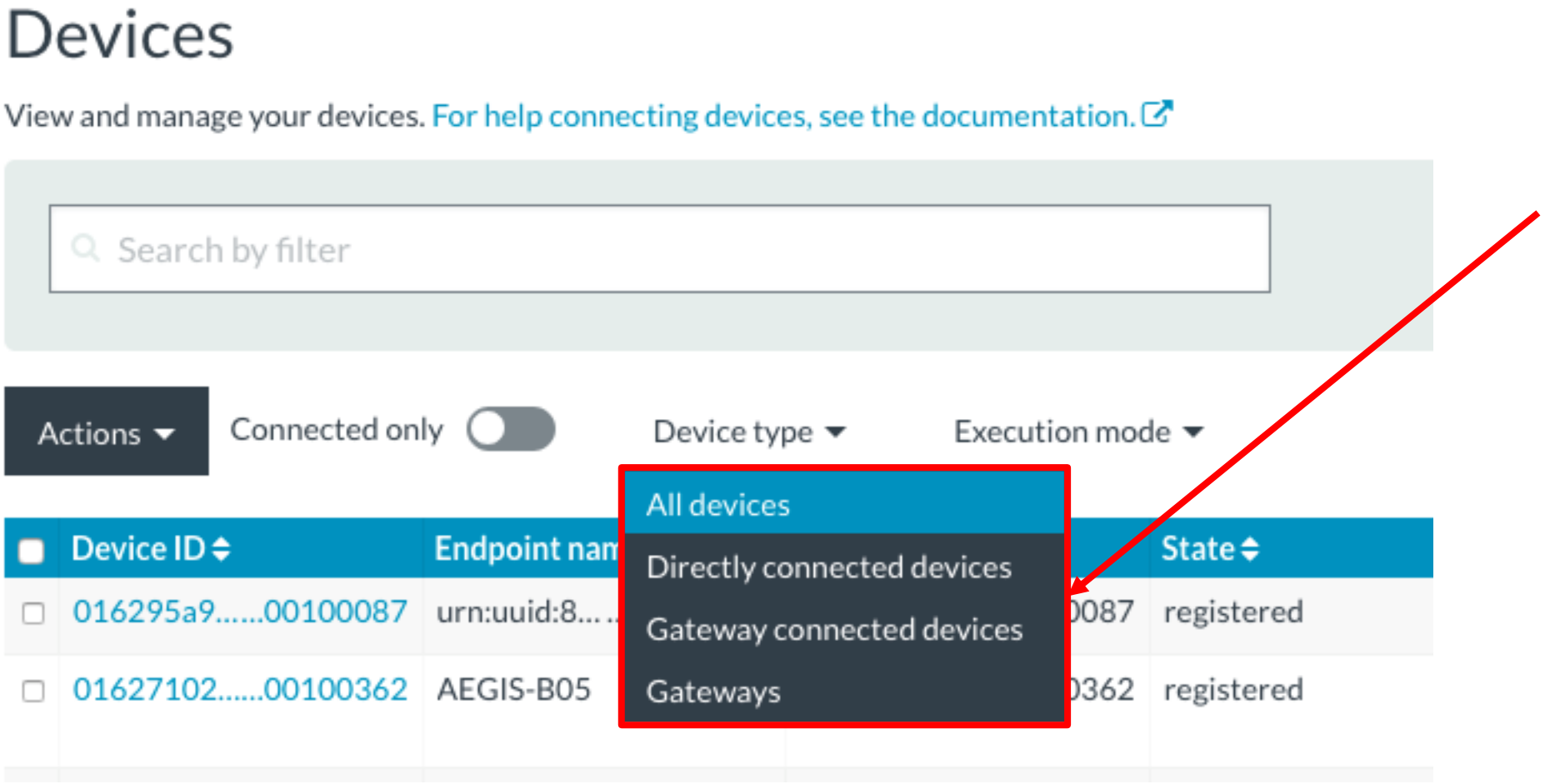
Device Directory (4)



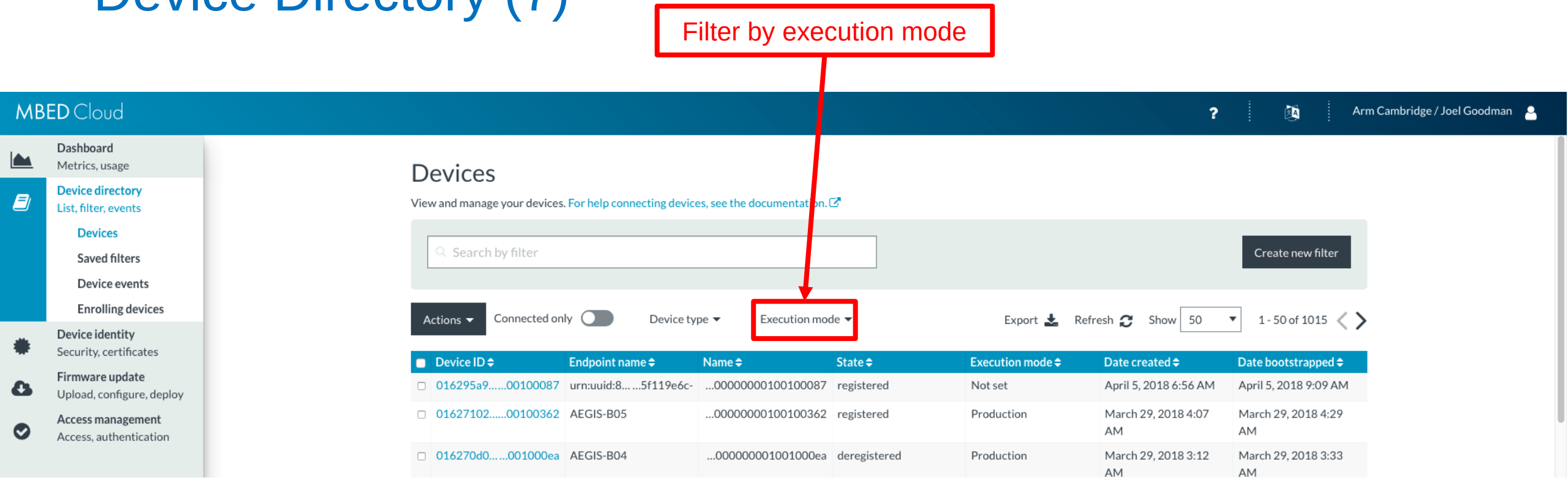
Device Directory (5)



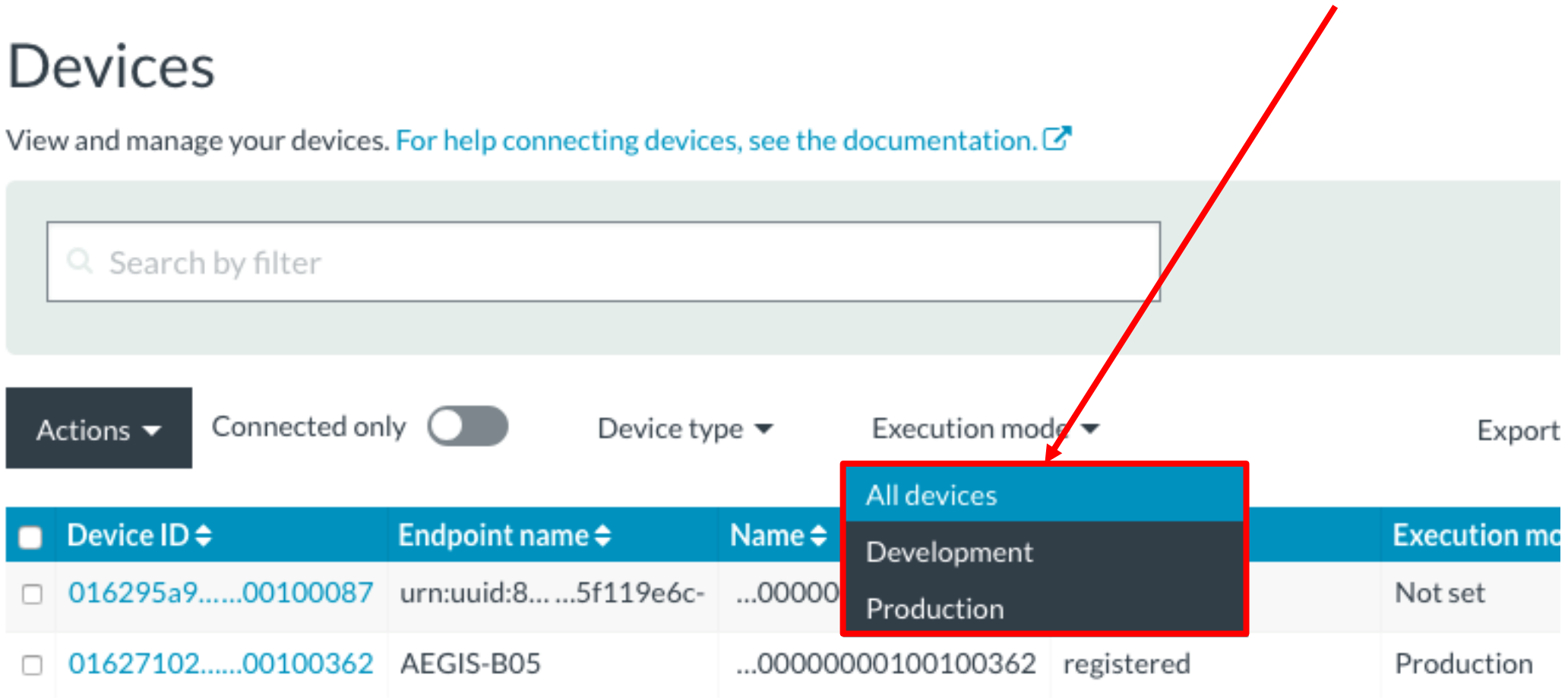
Device Directory (6)



Device Directory (7)

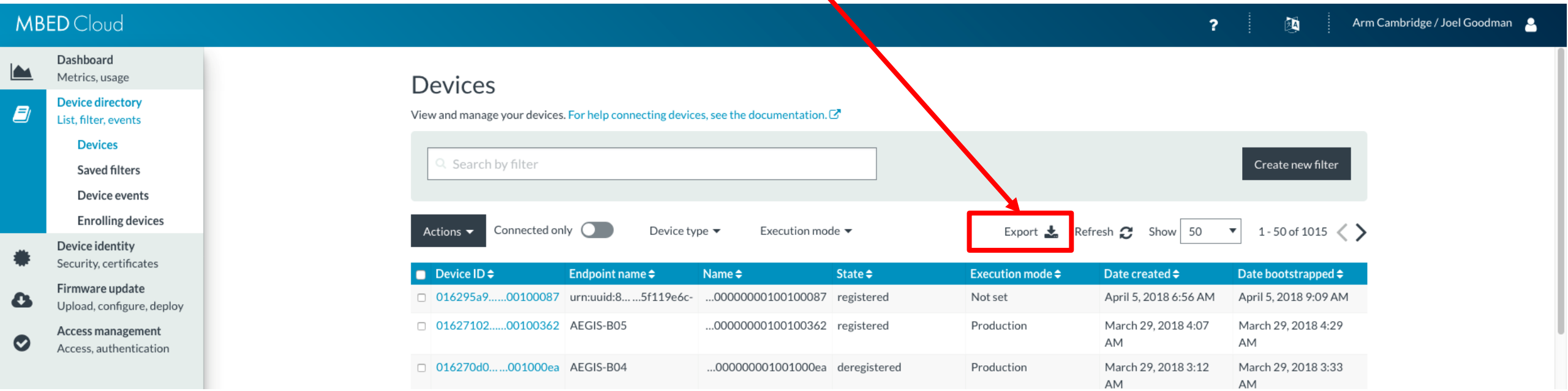


Device Directory (8)



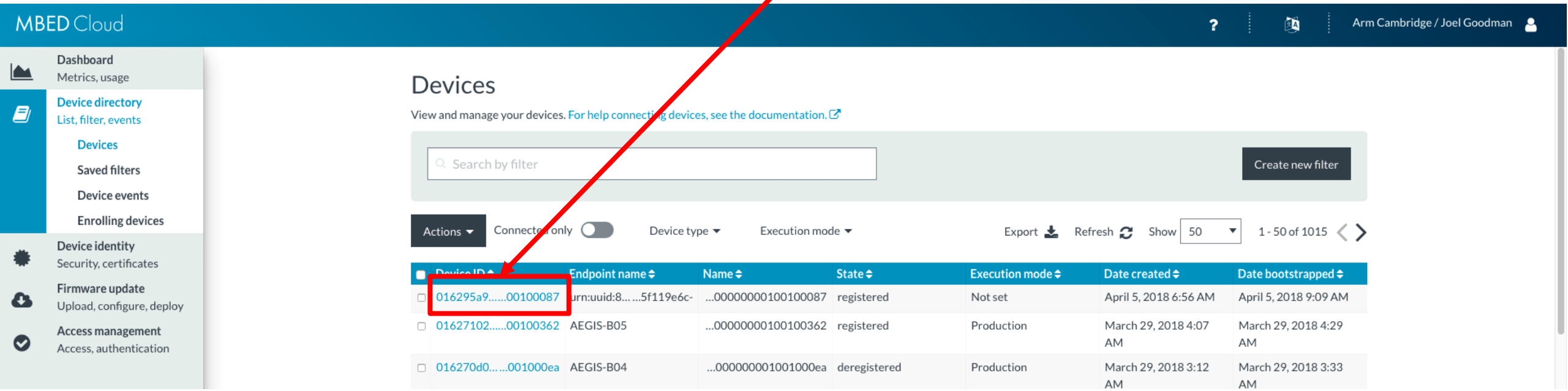
Device Directory (9)

Export list of devices to CSV

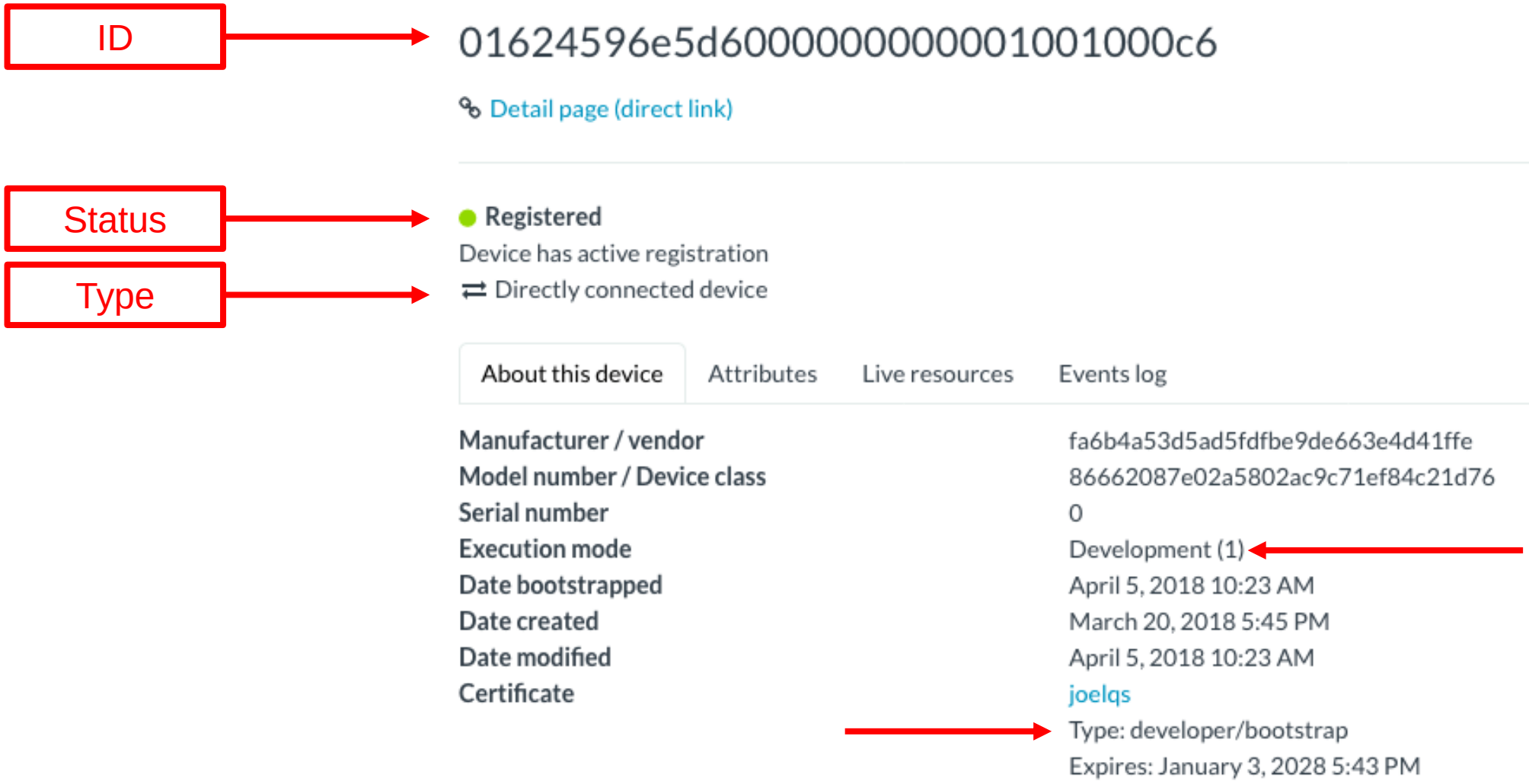


Device Directory (10)

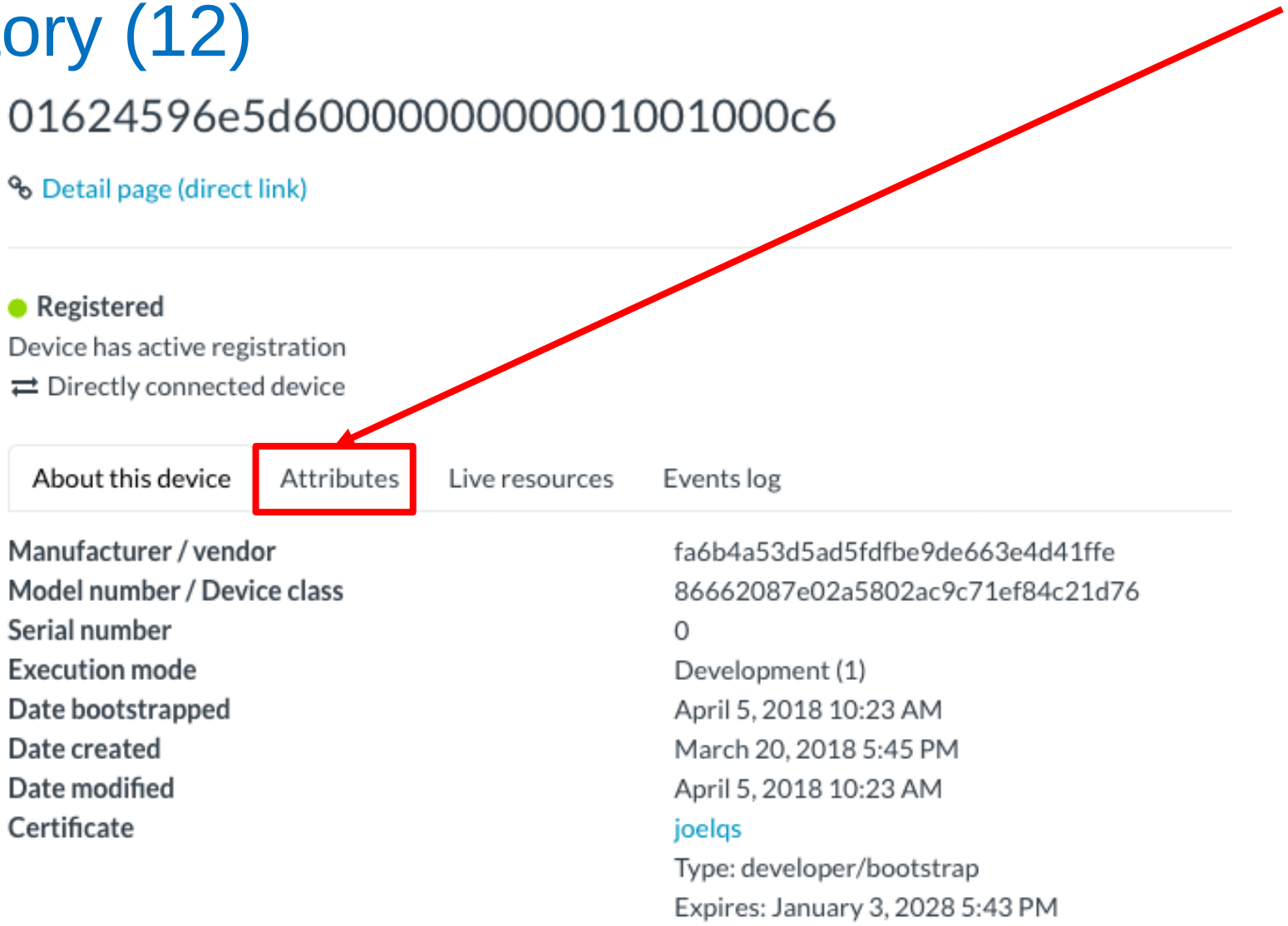
Deep dive to device



Device Directory (11)



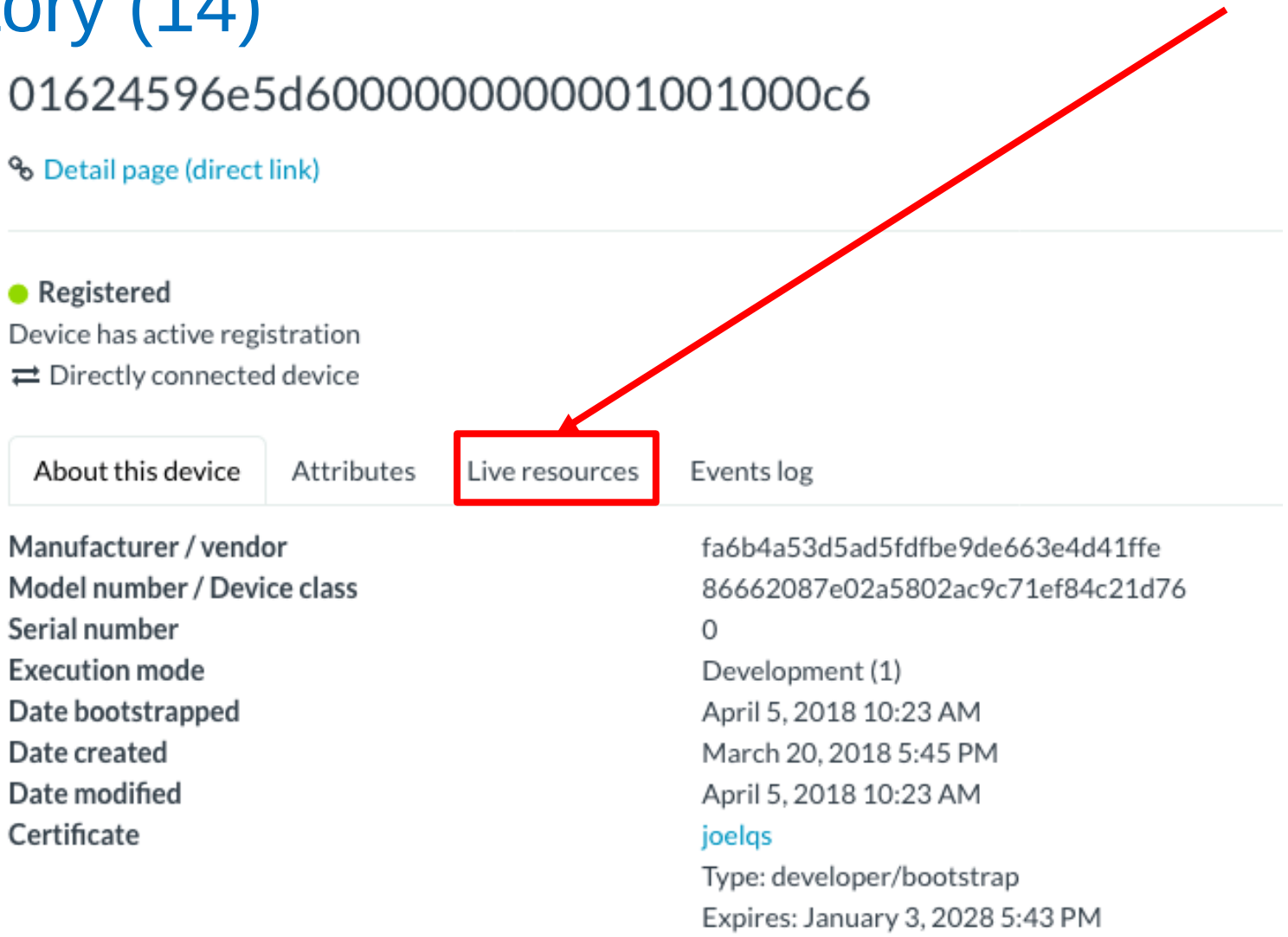
Device Directory (12)



Device Directory (13)

Name	Value
account_id	0157be9a987802420a010f0d00000000
auto_update	false
bootstrap_expiration_date	
bootstrapped_timestamp	2018-04-05T15:23:24.922835Z
ca_id	0160be68e7e42af07b2a4b1c03c00000
connector_expiration_date	
created_at	2018-03-20T22:45:39.943736Z
deployed_state	development
deployment	
description	
device_class	86662087e02a5802ac9c71ef84c21d76
device_execution_mode	1
device_key	F8:2E:84:04:2B:BD:84:B4:95:C9:A3:F3:5A:F2:41:58:2C:48:E2:5F:79:42:1A:A6:B9:F1:9D:99:8E:1F:2D:42
endpoint_name	01624596e5d6000000000001001000c6
endpoint_type	default
enrolment_list_timestamp	
etag	2018-04-05T15:23:35.393919Z
firmware_checksum	0000000000000000000000000000000000000000000000000000000000000000
host_gateway	
id	01624596e5d6000000000001001000c6
manifest	
manifest_timestamp	1970-01-01T00:00:00Z
mechanism	connector
mechanism_url	
name	01624596e5d6000000000001001000c6
object	device
serial_number	0
state	registered
updated_at	2018-04-05T15:23:35.393919Z
vendor_id	fa6b4a53d5ad5fdfbe9de663e4d41ffe

Device Directory (14)



Device Directory (15)

Device /3

Name	Path	Observable
Device - 0	/3/0	No
Manufacturer	/3/0/0	Yes
Model Number	/3/0/1	Yes
Serial Number	/3/0/2	Yes
Reboot	/3/0/4	No
Error Code	/3/0/11	No
Current Time	/3/0/13	Yes
Supported Binding and Modes	/3/0/16	Yes
Device Type	/3/0/17	Yes
Hardware Version	/3/0/18	Yes
Memory Total	/3/0/21	Yes

Firmware Update /5

Name	Path	Observable
Firmware Update	/5	No
Firmware Update - 0	/5/0	Yes
Package	/5/0/0	No
PackageURI	/5/0/1	No
Update	/5/0/2	No
State	/5/0/3	Yes
UpdateResult	/5/0/5	Yes
PkgName	/5/0/6	Yes
PkgVersion	/5/0/7	Yes

Illuminance /3301

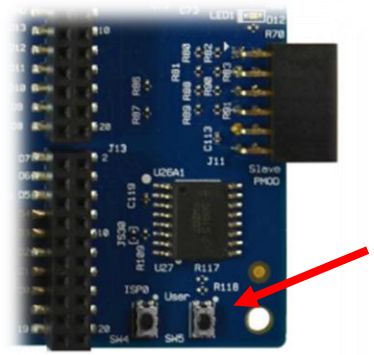
Name	Path	Observable
Illuminance	/3301	No
Illuminance - 0	/3301/0	No
light_value	/3301/0/5700	Yes

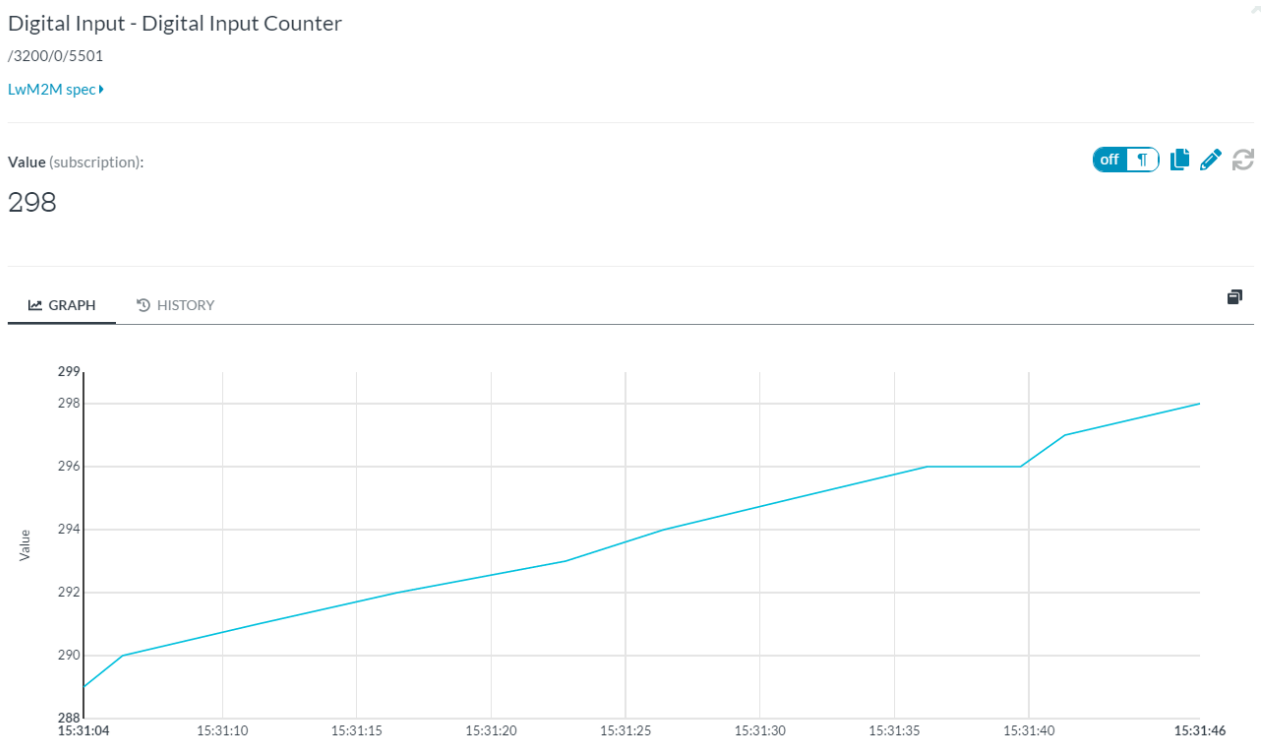
View the button resource

Digital Input	/3200
Digital Input - 0	/3200/0
button_resource	/3200/0/5501
3201	/3201
3201 -0	/3201/0
blink_resource	/3201/0/5850
pattern_resource	/3201/0/5853

Press the button, see the resource updated

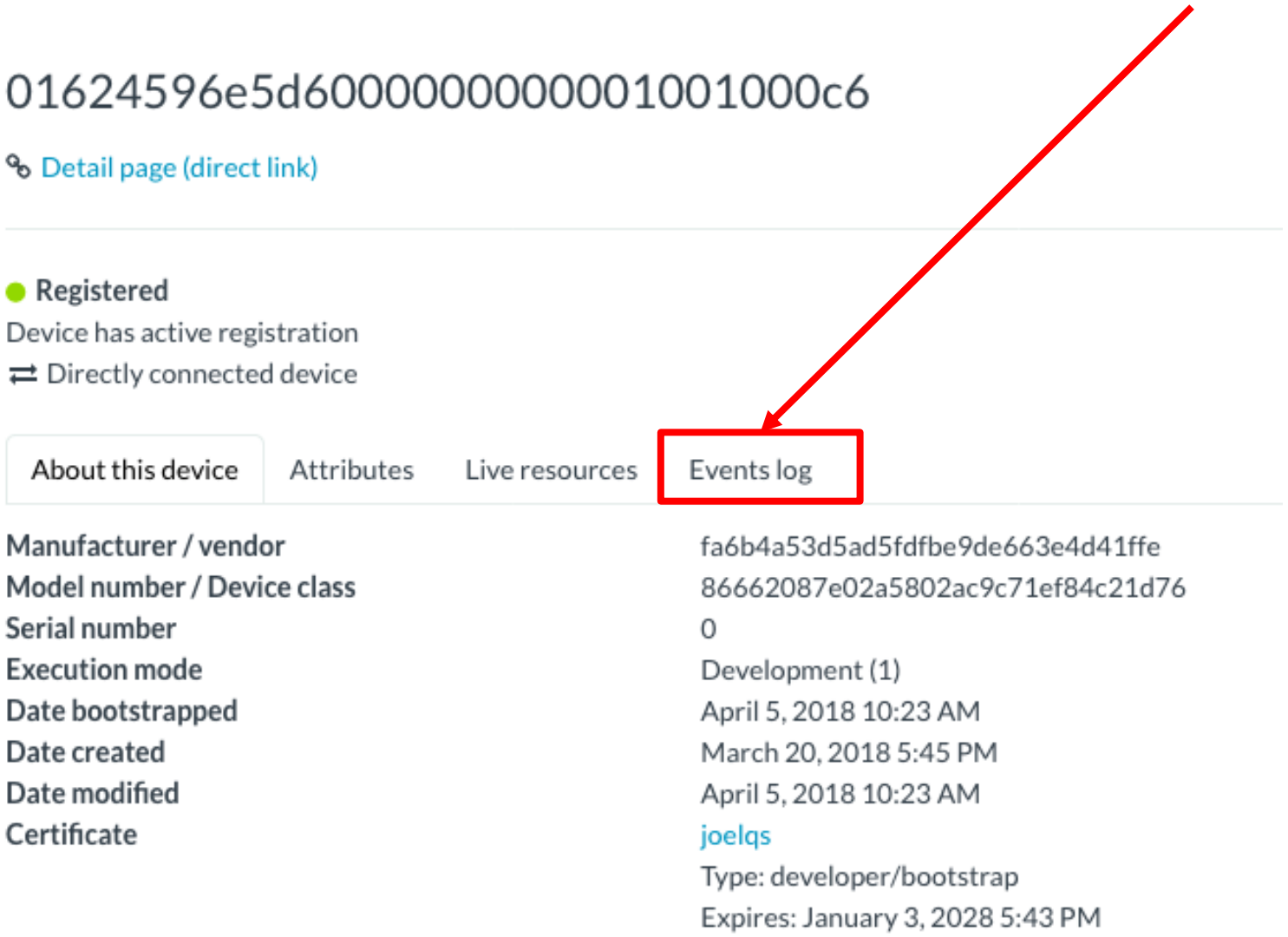
- Press SW5





This feature is not implemented yet in the example.
Currently it uses a simulated button press.

Device Directory (16)



Device Directory (17)

Event at	Description
April 5, 2018 10:23 AM	Firmware update state set to: Idle (before downloading or after successful updating) (0)
April 5, 2018 10:23 AM	Firmware update state set to: Idle (before downloading or after successful updating) (0)
April 5, 2018 10:23 AM	Device firmware package checksum notification received
April 5, 2018 10:23 AM	Device firmware package checksum notification received
April 5, 2018 10:23 AM	Device firmware package version notification received
April 5, 2018 10:23 AM	Device firmware package version notification received
April 5, 2018 10:23 AM	Device record updated, changed 'state' from 'bootstrapped' to 'registered' and

What Next?

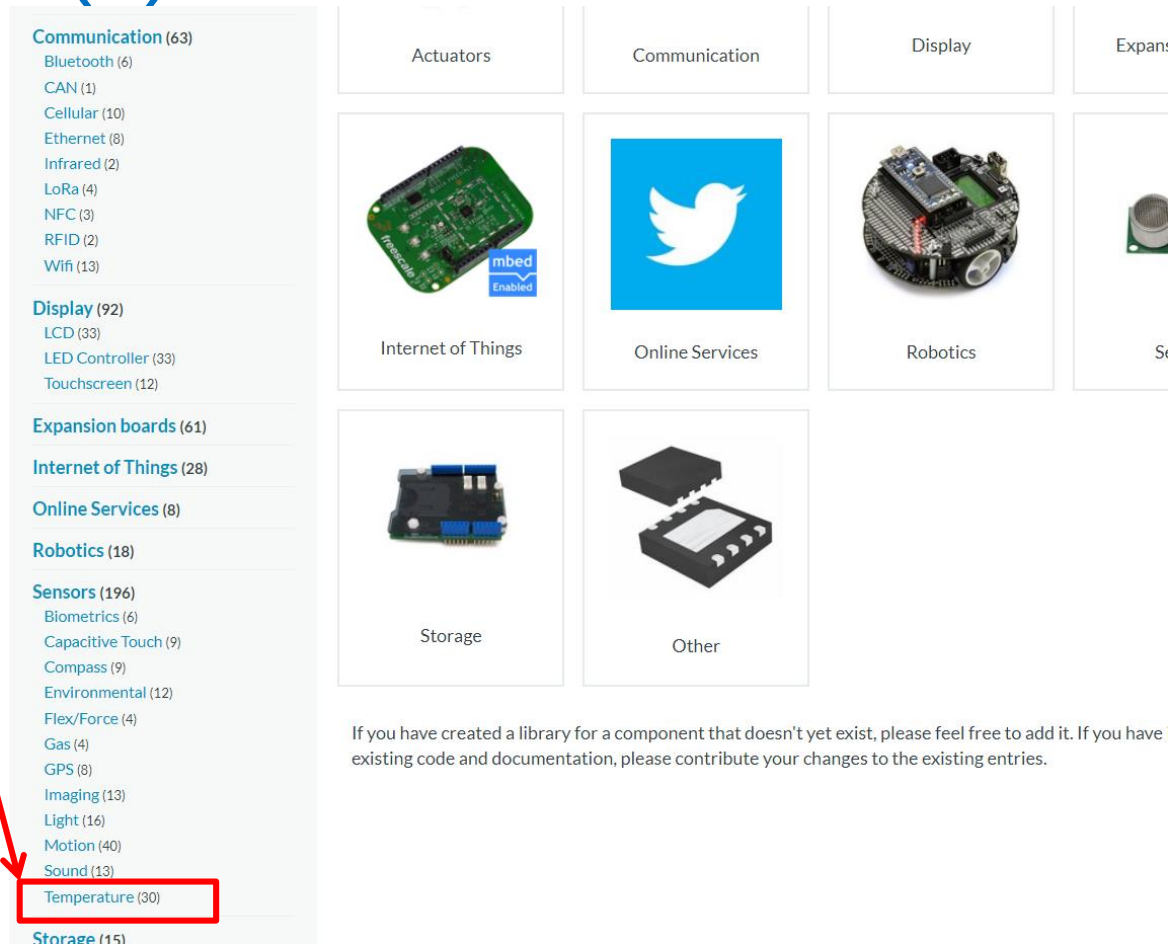
Add a sensor to your design

Let’s Add a Temperature Sensor to our Design

- Mbed CI Test Shield with SD card also has a LM75B temperature sensor
- Things we will discover:
  - How to use the component libraries available on mbed
  - How to configure the peripheral driver libraries for our hardware
  - How to add new resources to our cloud project and send the data to the cloud

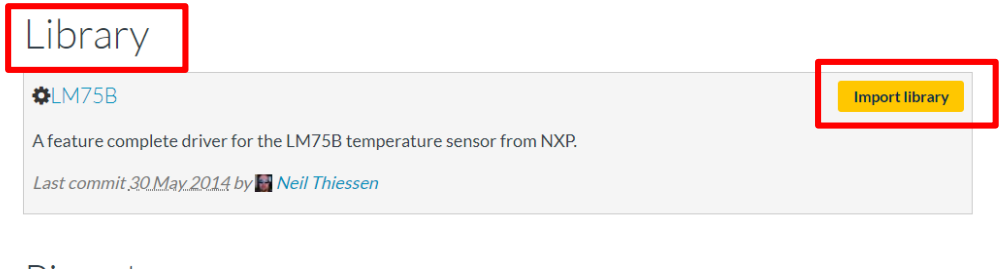
Adding a Temperature Sensor (1)

- Navigate to
- https://os.mbed.com/components/
- Select “Temperature” under Sensors:
- Select the LM75B

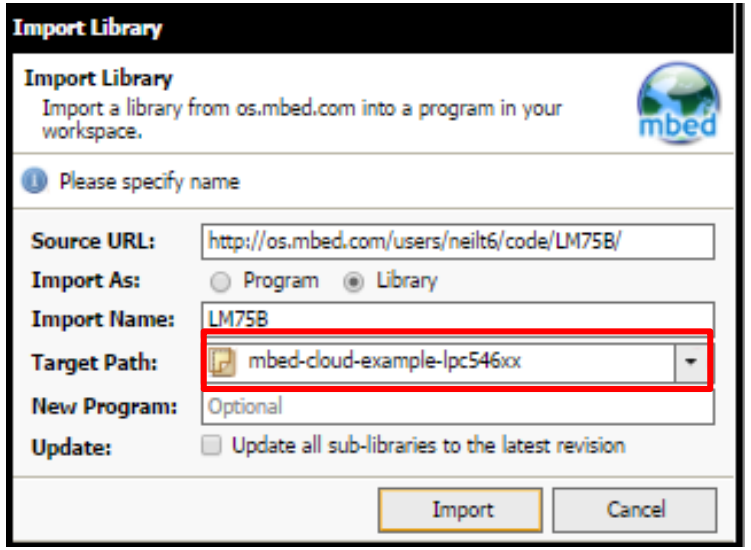


Adding a Temperature Sensor (2)

- Find the LM75B Library
  - Select “Import Library”



- The Online compiler will open in a new tab
  - You can close the old tab
- Select “mbed-cloud-example…” as the Target Path



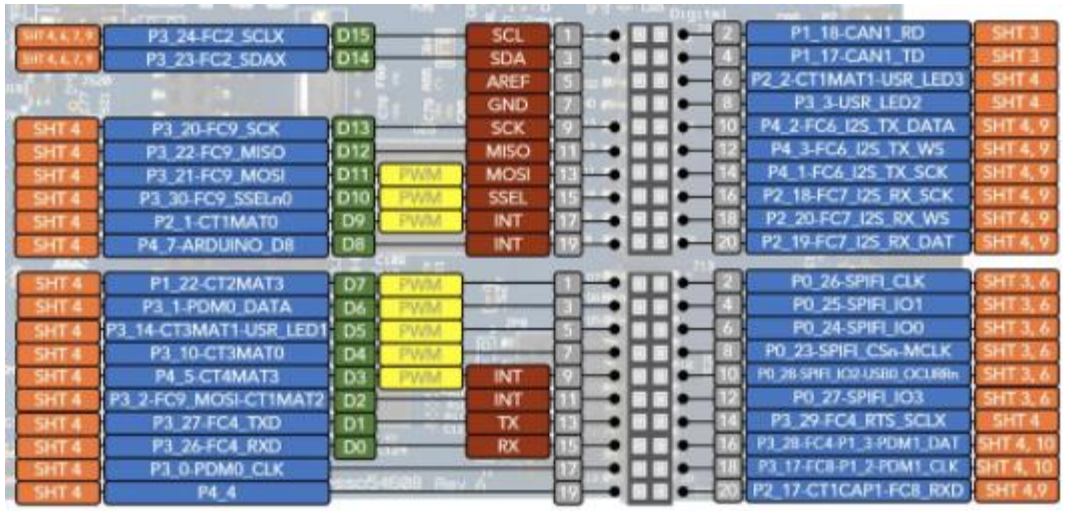
- We have now added the library to our project

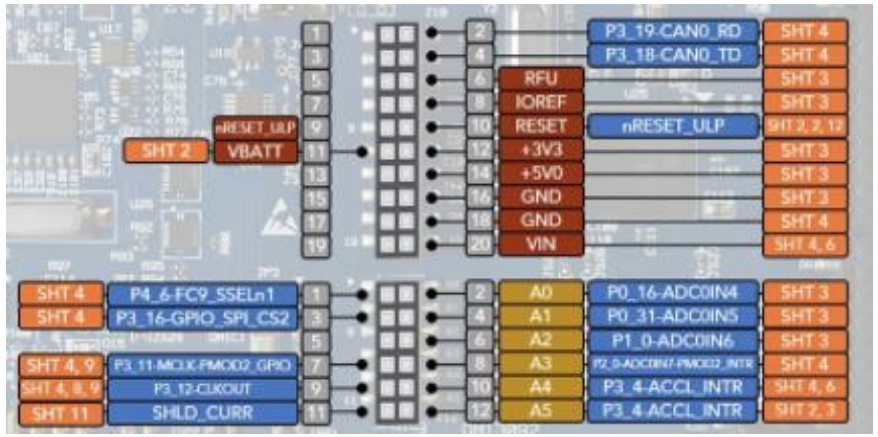
Adding a Temperature Sensor (3)

- Navigate to https://os.mbed.com/platforms/
- Search for the board you are going to use – in this case “LPCXpresso54608”
- Click on it

Adding a Temperature Sensor (4)

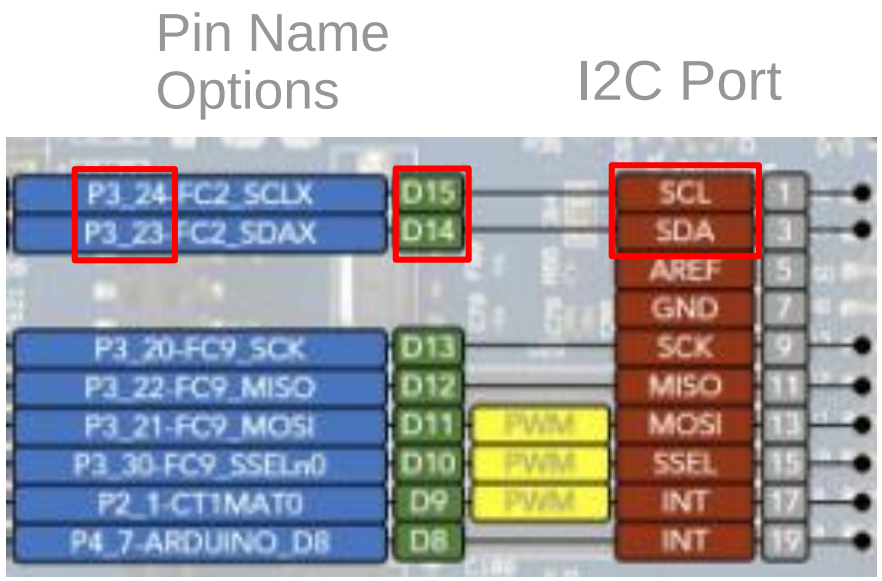
- Scroll down to the Board Pinout section





- These diagrams show us the available peripherals and the CPU ports they are connected to

Adding a Temperature Sensor (5)

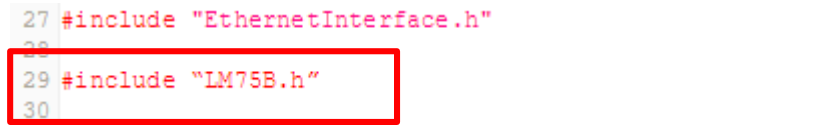


- We need the I2C port that is connected to the LB75B
- When we create the LM75B class in our program we need to assign the SDA and SCL pins to the class
- These assignments can use either the Arduino pin name or the Port name
- Examples:
  - LM75B sensor(D14, D15); // or
  - LM75B sensor(P3_23, P3_24); // both are valid
- This creates an object named “sensor” of the LM75B class

Adding a Temperature Sensor (6)

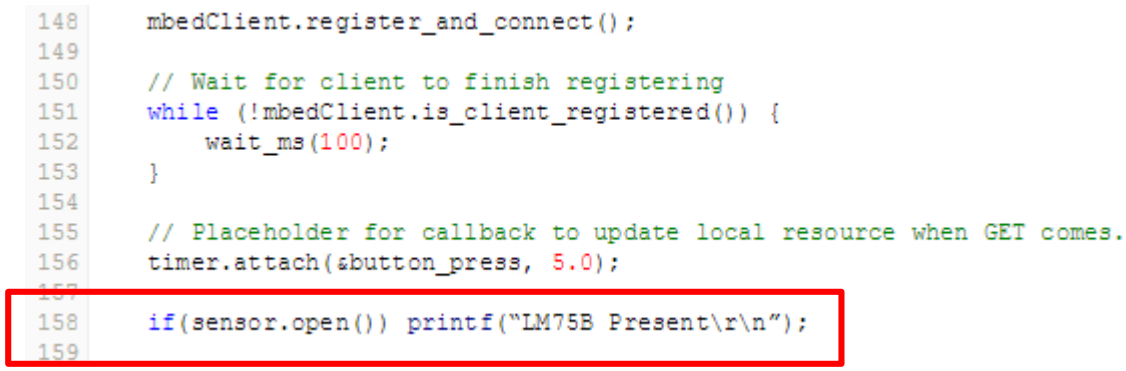
- Return to the online compiler and open main.cpp of project “mbed-cloud workshop”
- At the top of the file, in the includes section add:

```
#include "LM75B.h"
LM75B sensor(D14, D15);
```



- In the main function, add the following:

```
if(sensor.open()) printf("LM75B Present\r\n");
```



- At this point you could compile, download, and run the code to verify that the sensor is present. Next we will add the code to send the data to the cloud

Adding a Temperature Sensor (7)

- To add a new resource requires three things:
  - A pointer to a new MbedCloudClientResource entry
  - A definition of the resource
  - A routine to update that resource
- Easy parts first!
  - In main.cpp add the new pointer, let's call it "temperature_ptr"

```
42 // Pointers to the resources that will be created in main_
43 static MbedCloudClientResource* pattern_ptr;
44 static MbedCloudClientResource* temperature_ptr;
```

  - In the final loop in the "main" function add a call to update_temperature();

```
167         if (button_pressed) {
168             button_pressed = false;
169             printf("Simulated button clicked %d times\r\n", ++button_count);
170             button->set_value(button_count);
171         }
172
173         update_temperature();
174
```

Adding a Temperature Sensor (8)

- Next lets define the resource, in main.cpp find the definition of the button resource.
  - We will use this as a template

```
// Mbed Cloud Client resource setup
MbedCloudClientResource *button = mbedClient.create_resource("3200/0/5501", "button_resource");
button->set_value("0");
button->methods(M2MMethod::GET);
button->observable(true);
button->attach_notification_callback(button_callback);
```

  - Cut and paste to make a duplicate, modify as noted below

```
MbedCloudClientResource *temperature = mbedClient.create_resource("3303/0/5700", "temperature_resource");
temperature->set_value("0");
temperature->methods(M2MMethod::GET);
temperature->observable(true);
temperature_ptr = temperature;
```

  - Change the resource identifier, update the URI, remove the callback, set the pointer to this resource
  - The resource URI and data type are part of LWM2M, let's understand why we picked these values

Adding a Temperature Sensor (9)

- This is how LWM2M defines the different elements of a temperature resource:

- 3303/0/5700
  - ObjectID = 3303
  - First Object = 0
  - Sensor Value = 5700
  - Data Type = FLOAT

Temperature

Description

Description: This IPSO object should be used with a temperature sensor to report a temperature measurement. It also provides resources for minimum/maximum measured values and the minimum/maximum range that can be measured by the temperature sensor. An example measurement unit is degrees Celsius (ucum:Cel).

Object definition

Name	Object ID	Object Version	LWM2M Version
Temperature	3303		
Object URN		Instances	Mandatory
urn:oma:lwm2m:ext:3303		Multiple	Optional

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
5700	Sensor Value	R	Single	Mandatory	Float		Defined by â€œUnitsâ€ resource.	Last or Current Measured Value from the Sensor
5601	Min Measured Value	R	Single	Optional	Float		Defined by â€œUnitsâ€ resource.	The minimum value measured by the sensor since power ON or reset
5602	Max Measured Value	R	Single	Optional	Float		Defined by â€œUnitsâ€ resource.	The maximum value measured by the sensor since power ON or reset
5603	Min Range Value	R	Single	Optional	Float		Defined by â€œUnitsâ€ resource.	The minimum value that can be measured by the sensor

- Learn More Here:
- OMA LwM2M Registry
- http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html

Adding a Temperature Sensor (10)

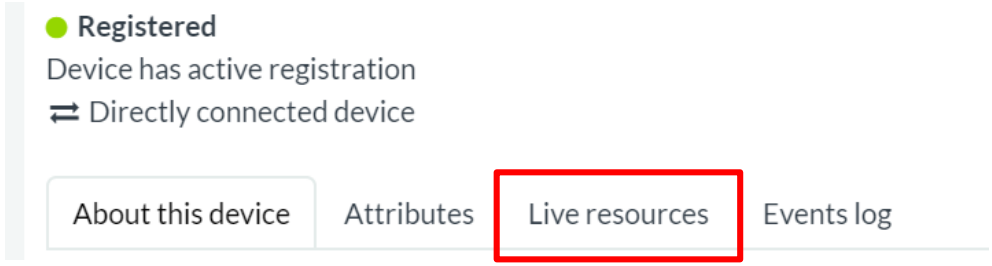
- Final step is to create a function to update the temperature resource
- It should look like this:

```
80 void update_temperature(void)
81 {
82     char tempbuffer[16];
83     // copy the temperature as a float to the buffer
84     snprintf(tempbuffer, 16, "%.3f", sensor.temp());
85     //copy the value to the set value element of the resource
86     temperature_ptr->set_value(tempbuffer);
87 }
```

- At this point you should be able to compile, load the binary onto your board, & connect your device to the cloud.
- Once the device is connected, return to the cloud portal and select your connected device.

Adding a Temperature Sensor (11)

- Once you have selected your device select “Live Resources”



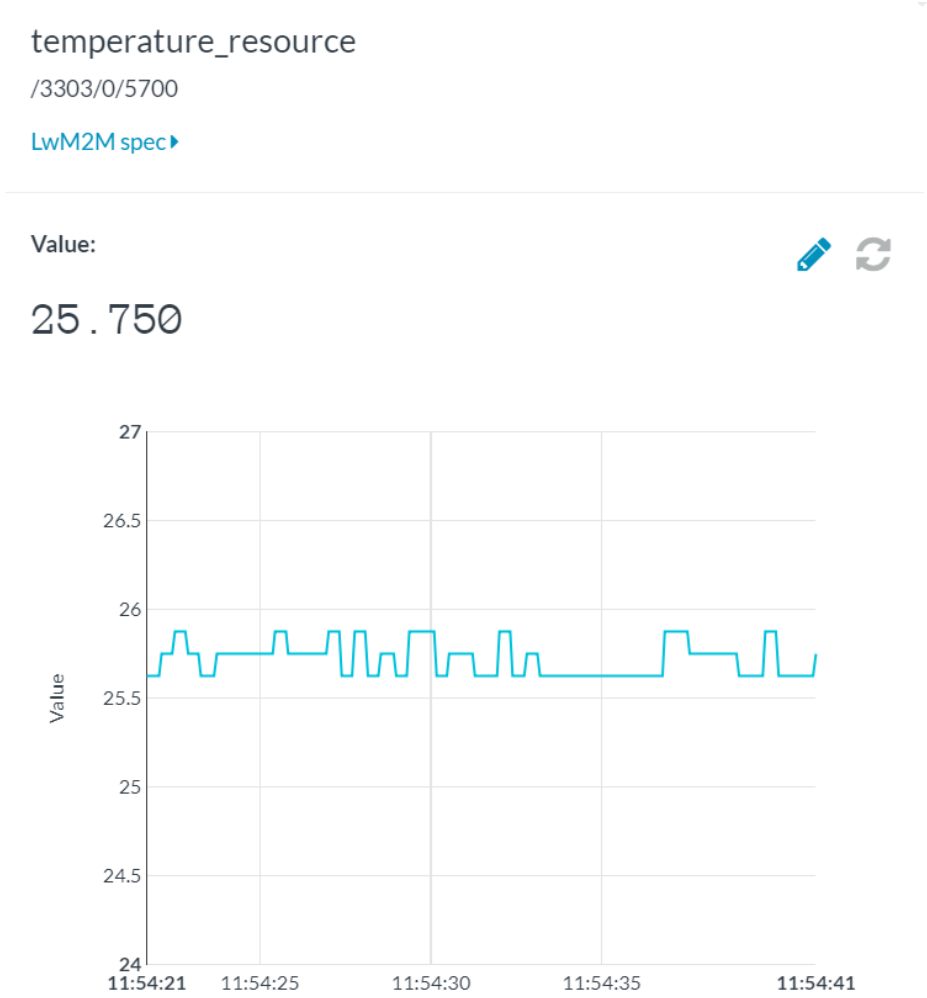
- Scroll down the resources and find your Temperature Resource

Temperature /3303

Name	Path	Observable
Temperature	/3303	No
Temperature - 0	/3303/0	No
temperature_resource	/3303/0/5700	Yes

- Select “temperature_resource and observe the temperature, since the value is not filtered it is normal to have small variations

Adding a Temperature Sensor (12)



Debugging w/ MCUXpresso IDE

Develop and Debug your Mbed OS–based applications with an IDE

Tools for development

Already installed for you today.

https://os.mbed.com/docs/latest/tools/installation-and-setup.html

Windows Installer Available, installs all these for you

- Mbed Command Line Interface (CLI)
  - Comes with Python 2, Git, Mercurial, GCC 6

Import example application

Open Command Prompt 

- Type **cmd** in the Windows search bar

Change directory. Type these commands at the command prompt.

- `cd documents`

Import the example in your PC. Type these commands.

- `mbed import https://github.com/ArmMbed/mbed-os-example-blinky`

Change Directory

- `cd mbed-os-example-blinky`

Open Windows explorer to view the files such as main.cpp

- `explorer .`

Develop & debug with an IDE

Exporters (a.k.a Project File Generators)

- ✓Keil µVision IDE
- ✓IAR Embedded Workbench
- ✓Eclipse IDE
- ✓**MCU**Xpresso IDE 
- ✓Makefiles

Note: Not all platforms are supported by all exporters / IDEs

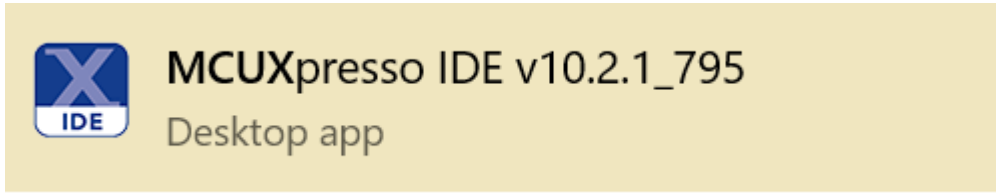
Export (a.k.a generate project files)

Export project to MCUXpresso IDE

- `mbed export -i mcuxpresso -m LPC546XX`

Open MCUXpresso IDE

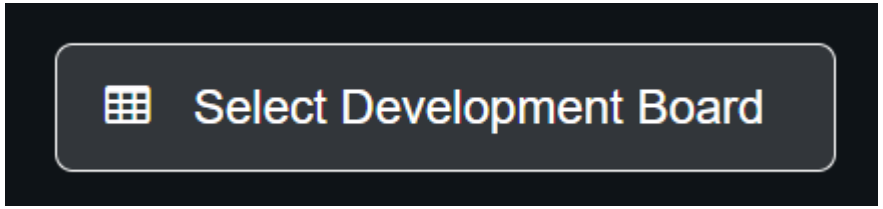
Search for and open MCUXpresso IDE 10.2



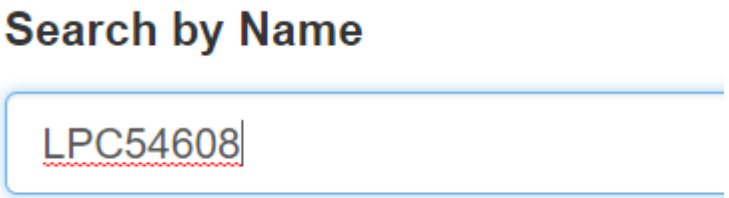
Install SDK for LPC546XX

Visit https://mcuxpresso.nxp.com/en/welcome

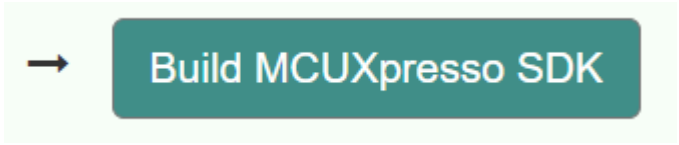
Select Development Board



Search by Name

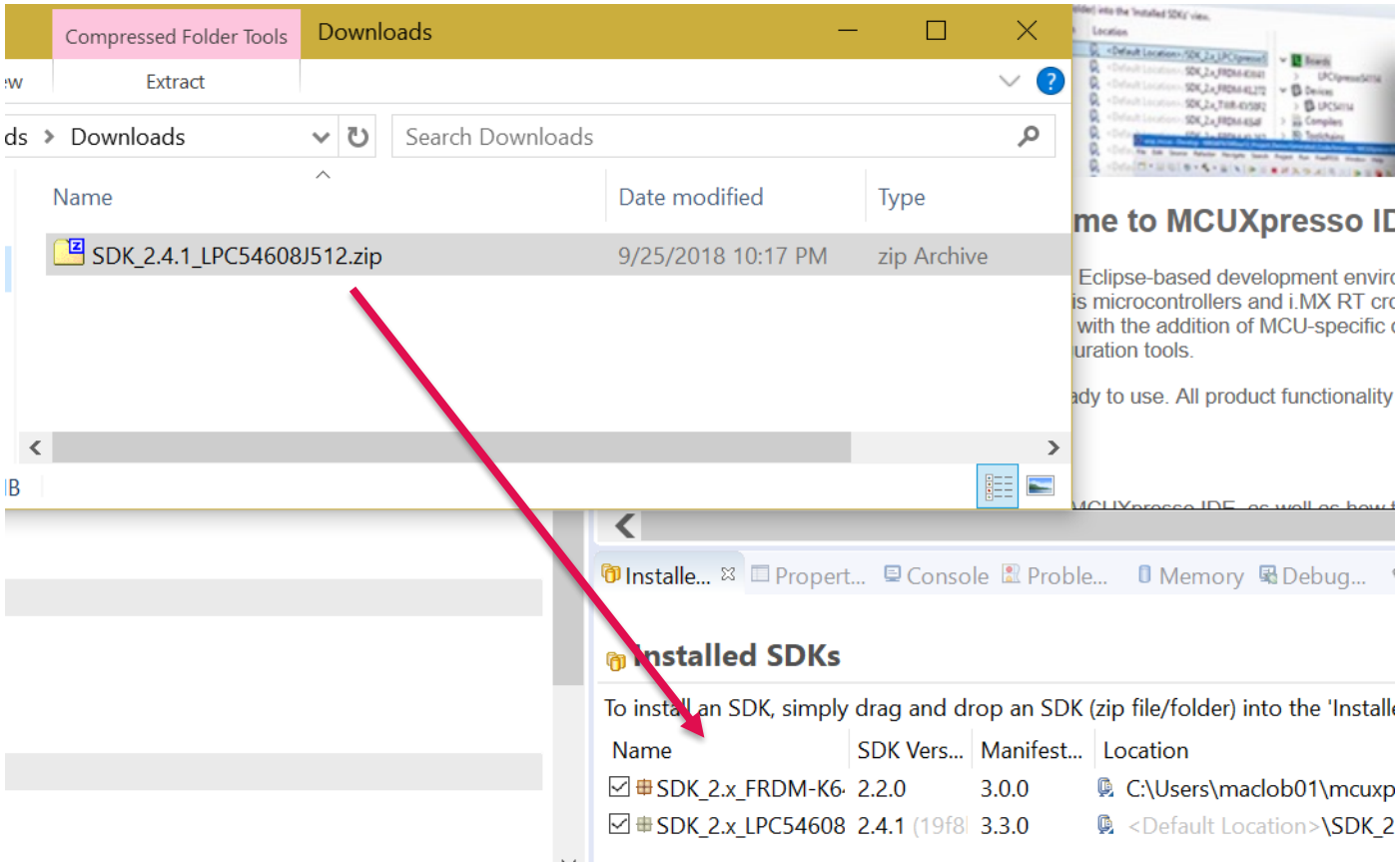


Click Build MCUXpresso SDK



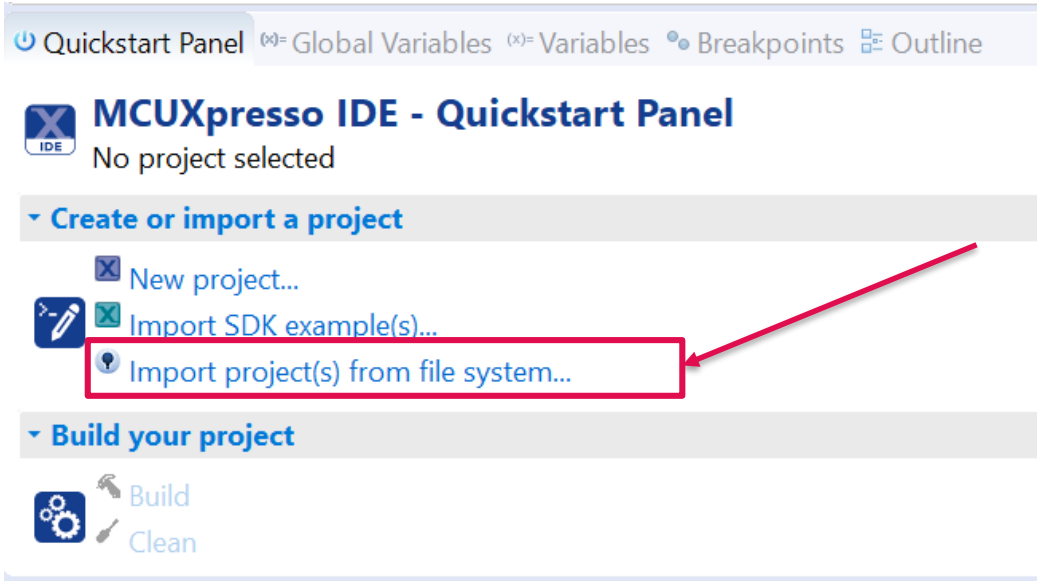
Add SDK to MCUXpresso IDE

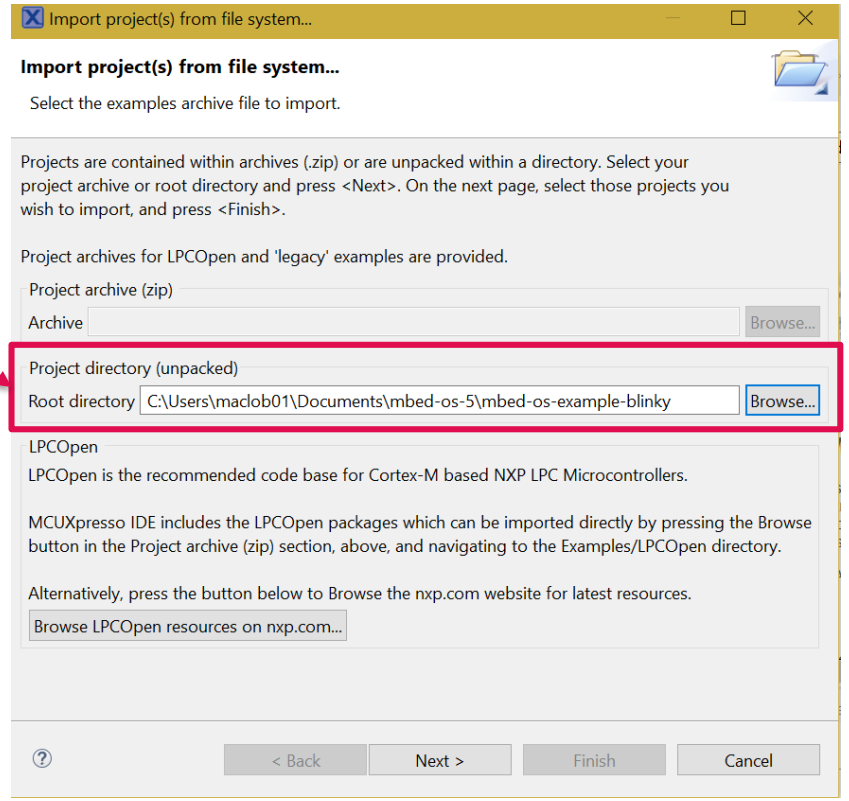
Download the SDK zip file, then drag it into MCUXpresso IDE



Open MCUXpresso IDE Project

1. Choose a workspace location – any will do
2. In the Quickstart Panel, choose Import project(s) from file system…
3. Under Project directory (unpacked) browse to the project



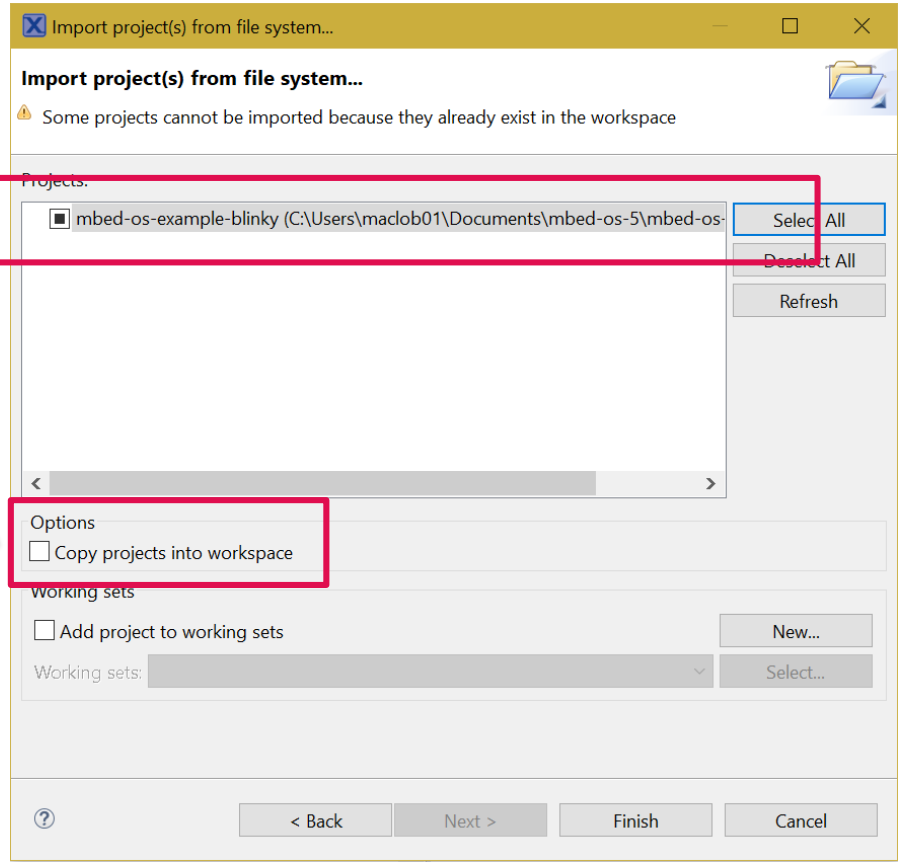


Open MCUXpresso Project

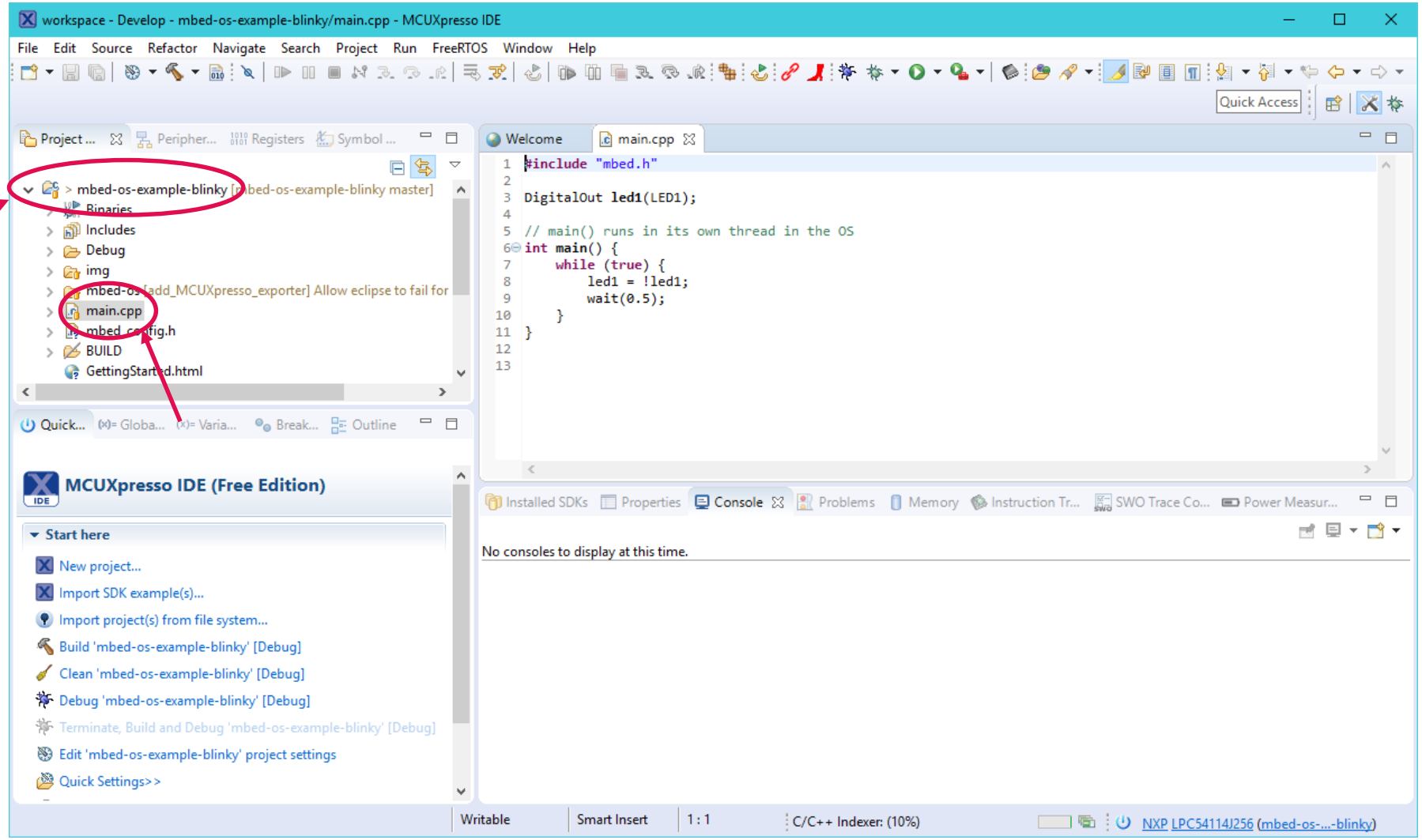
Select the project

Un-check Copy projects into workspace!

Click Finish



Open main.cpp



Build using the command line

Go back to the windows command prompt

Execute the following command within the project directory on the command line.

```
mbed compile -m LPC546XX -t GCC_ARM
```

Important Note
Building Mbed OS apps within MCUXpresso IDE is currently not working due to an Mbed OS problem. The workaround is to build on the command line.

Debugging Options

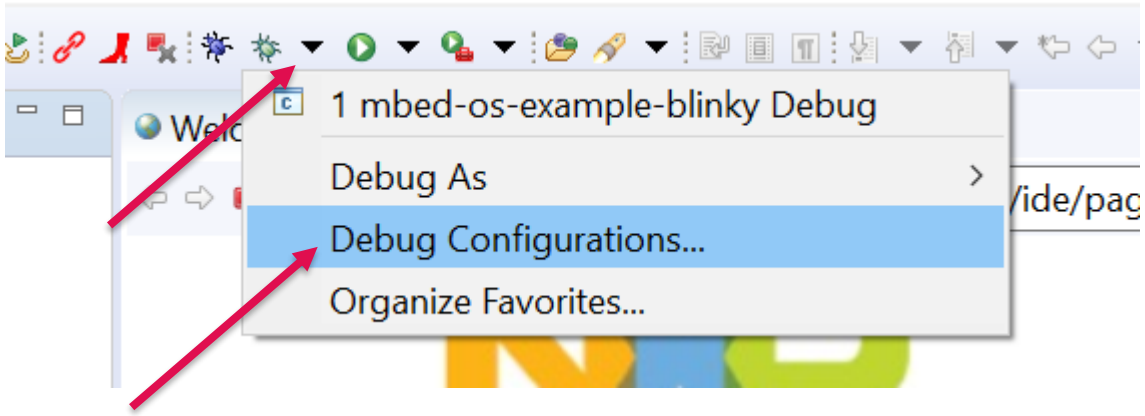
There are a few options for the debug connection driver.

We’re going to use pyOCD.

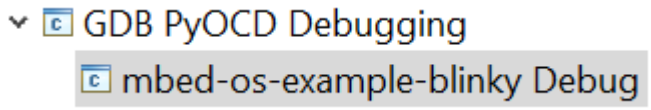
Install pyOCD plugin by following the blog post at

https://os.mbed.com/users/c1728p9/notebook/debugging-with-eclipse-and-pyocd/

Open Debug Configurations

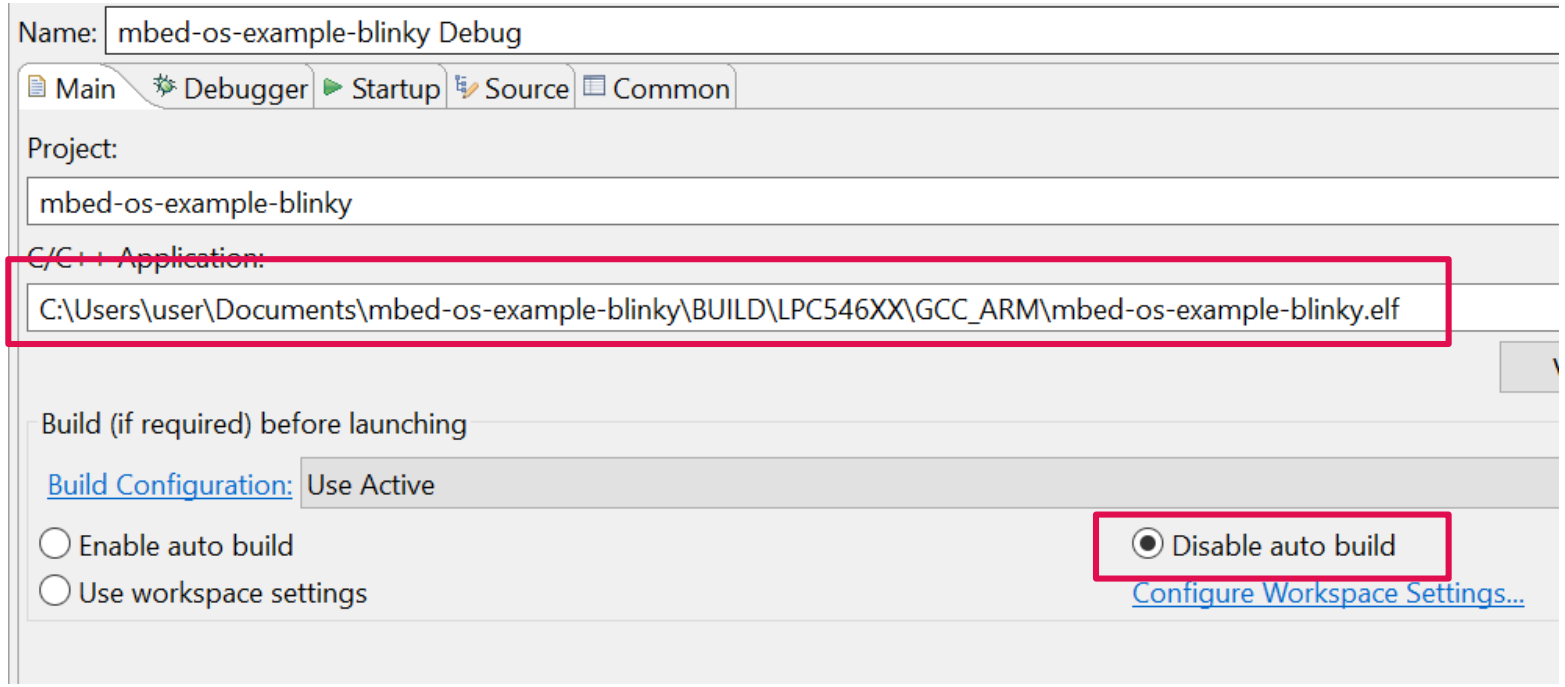


Click on GDB PyOCD Debugging

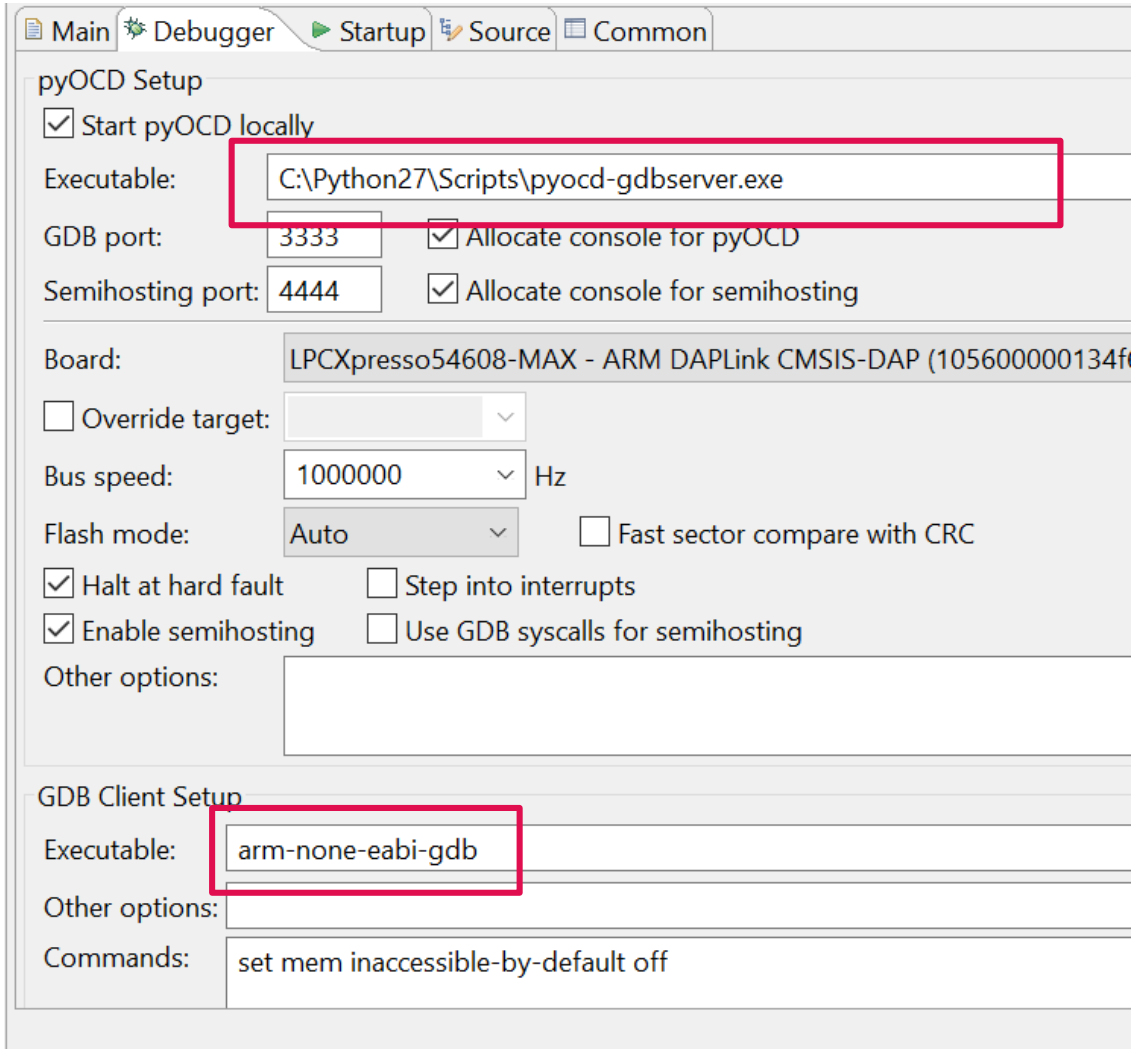


Use the following settings

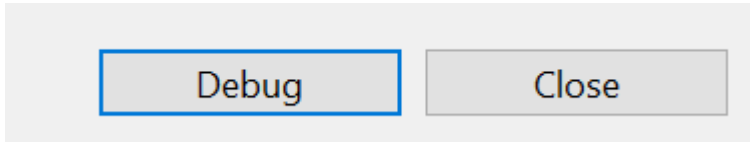
- Adjust the path as necessary to the elf file
- Disable auto build



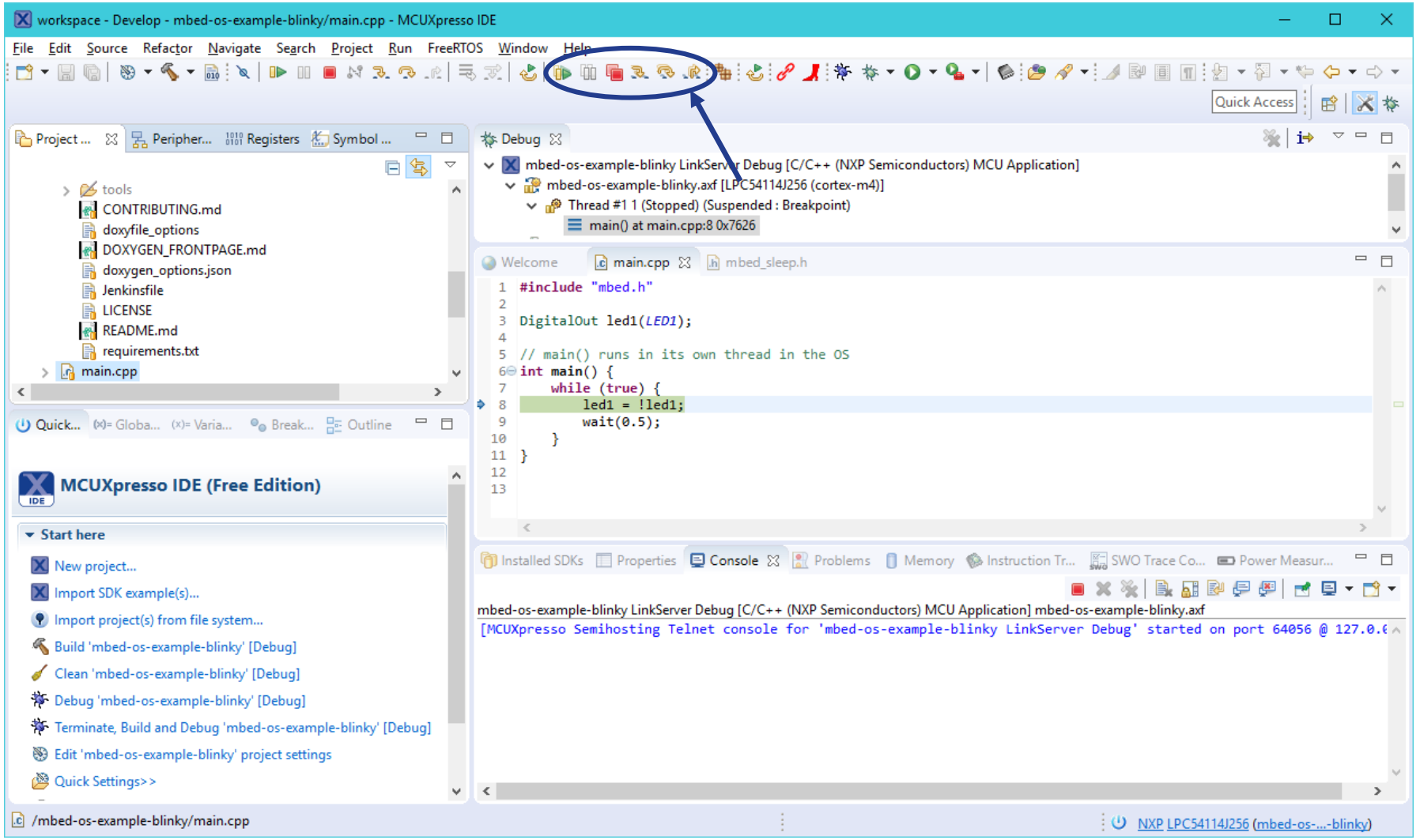
Use the following settings on the Debugger tab



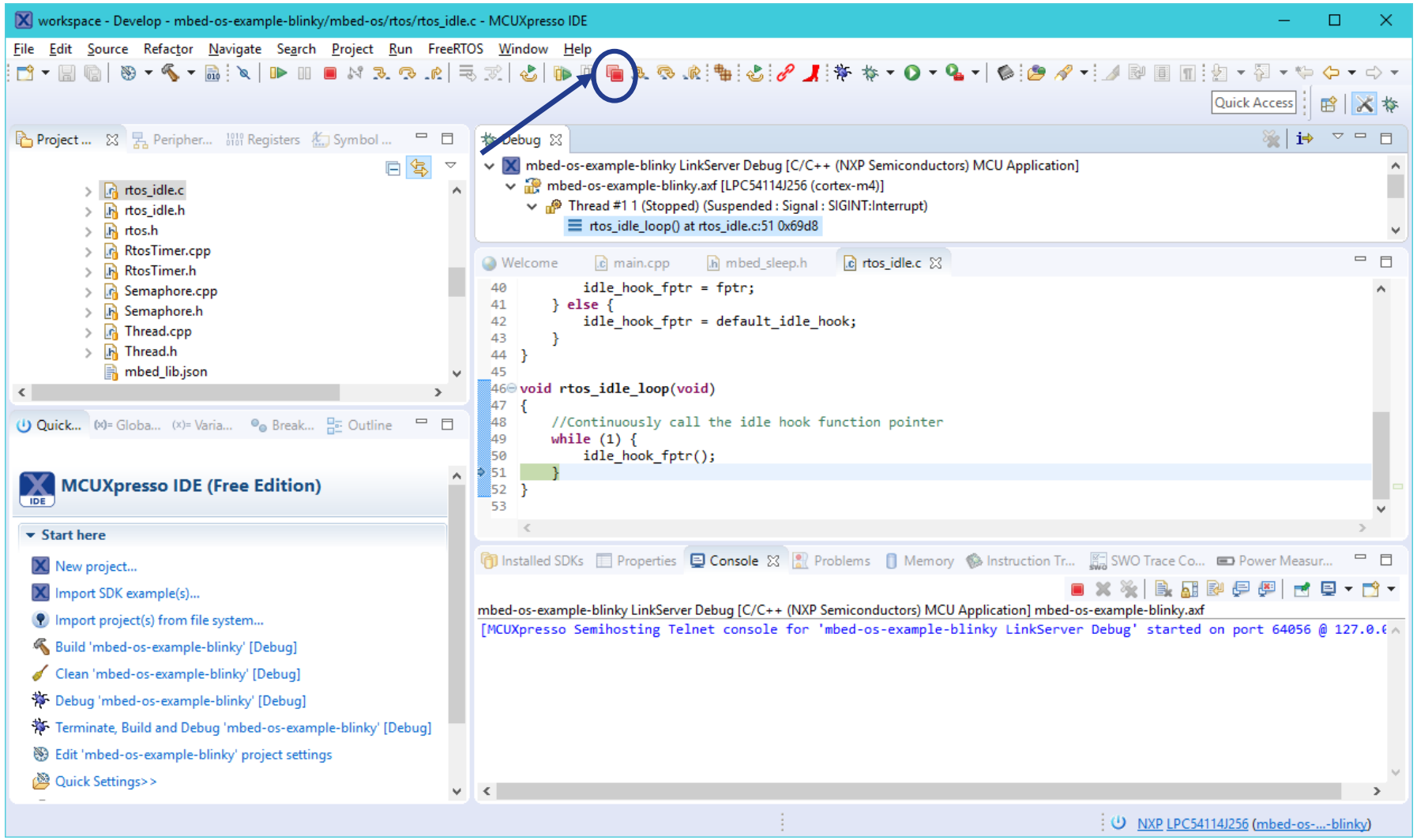
Click Debug



Run, Halt, Step through the application



Terminate the debug session



Add a print statement

Update main.cpp with the following code.

```
#include "mbed.h"

DigitalOut led1(LED1);

// main() runs in its own thread in the OS
int main() {
    while (true) {
        led1 = !led1;
        printf("hello\r\n");
        wait(0.5);
    }
}
```

Edit main.cpp, Build, Debug, Run

Open terminal program

Check the COM port

Run this command in the command prompt 

```
mbed detect
[mbed] Detected LPC546XX, port COMxx, mounted D:
[mbed] Supported toolchains for LPC546XX
```

Open a terminal program such as Tera Term and choose the settings:

- Baud rate: **9600**
- Data: 8 bit
- Parity: None
- Stop: 1 bit
- Flow control: none

When complete, terminate the debug

Create a gpio interrupt

```
#include "mbed.h"

DigitalOut led1(LED1,1);
InterruptIn sw(P1_1);

void rise_handler(void) {
        // Toggle LED
    led1 = !led1;
}
int main() {

        sw.rise(rise_handler);

        while(1);
}
```

Edit main.cpp, Build, Debug, Run

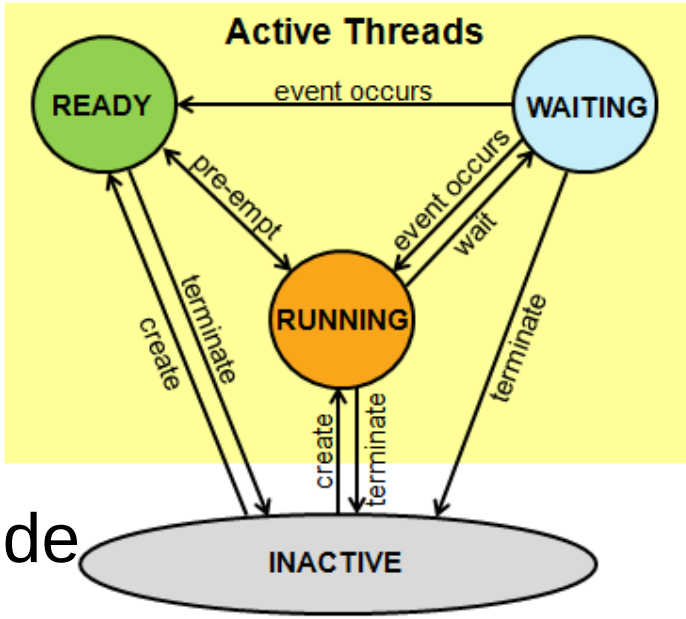
Press **SW5**. The LED should light up.

When complete, terminate the debug session.

Mbed OS 5 - Mbed RTOS

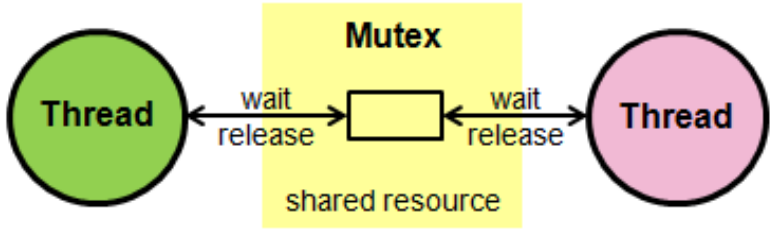
Includes CMSIS-RTOS RTX

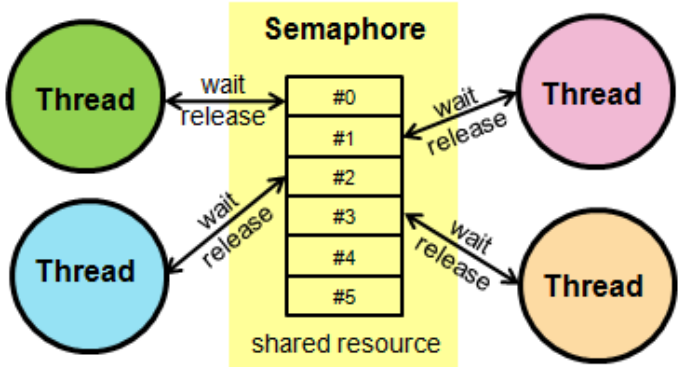
- Based on the Keil RTX Real-Time Operating System
- Multi-Thread & pre-emptive scheduler



Mbed RTOS is a C++ wrapper over the Keil RTX code

- Thread
- Mutex
- Semaphores
- Queue and MemoryPool
- Mail
- RTOS Timer
- ISR





Add an RTOS thread

```
#include "mbed.h"
DigitalOut led1(LED1,1);
DigitalOut led2(LED2,1);
Thread thread;

void led2_thread() {
    while (true) {
        led2 = !led2;
        Thread::wait(1000);
    }
}
int main() {
    thread.start(led2_thread);

    while (true) {
        led1 = !led1;
        Thread::wait(500);
    }
}
```

Edit main.cpp, Build, Debug, Run.

When complete, terminate the debug session.

Mbed OS 5 - Event queue library

The mbed-events library provides a flexible queue for scheduling events.

Initialized within an Mbed RTOS task

Thread & IRQ safe

mbed-events library can

- Act as drop-in scheduler
- Provide synchronization between multiple threads
- Act as a mechanism for moving events out of interrupt contexts.

Targeting power constrained applications

Use event loop

```
#include "mbed.h"
#include "mbed_events.h"

DigitalOut led1(LED1);
InterruptIn sw(P1_1);
EventQueue queue(32 * EVENTS_EVENT_SIZE);
Thread t;

void rise_handler(void) {
    // Toggle LED
    led1 = !led1;
}

void fall_handler(void) {
    printf("fall_handler in context %p\r\n", Thread::gettid());
    // Toggle LED
    led1 = !led1;
}
```

Continued on next slide

Edit main.cpp

Use event loop (Continued)

```
int main() {
    // Start the event queue
    t.start(callback(&queue, &EventQueue::dispatch_forever));
    printf("Starting in context %p\r\n", Thread::gettid());
    // The 'rise' handler will execute in IRQ context
    sw.rise(rise_handler);
    // The 'fall' handler will execute in the context of thread 't'
    sw.fall(queue.event(fall_handler));

    while(1);

}
```

Continue editing main.cpp, Build, Debug, Run.

When complete, terminate the debug session.


Learn More at…
www.mbed.com

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com