STM32L4 Amazon FreeRTOS Hands-On-Training: Simplifying AWS IoT Cloud Platform Connectivity for Embedded Devices

Ernesto Manuel Cantone, STMicroelectronics

Daniel Griffin, STMicroelectronics

Anton Shmagin, Amazon Web Services

Scott Francis, Amazon Web Services

life.augmented

aws partner network

Standard Technology Partner

Agenda

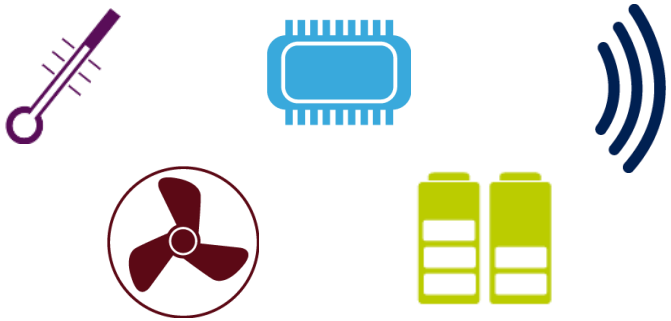
- Introduction
- Training Material and installation needs
- Connect to Amazon Workspaces
- Create your Device (“Thing”) on AWS
- Compile “Hello World” demo project
- Connect to AWS IoT & Send Data
- Compile Sensors demo project
- Send sensors data to the AWS IoT
- Store and transform data in AWS

….

ST Has all the Building Blocks for the IoT

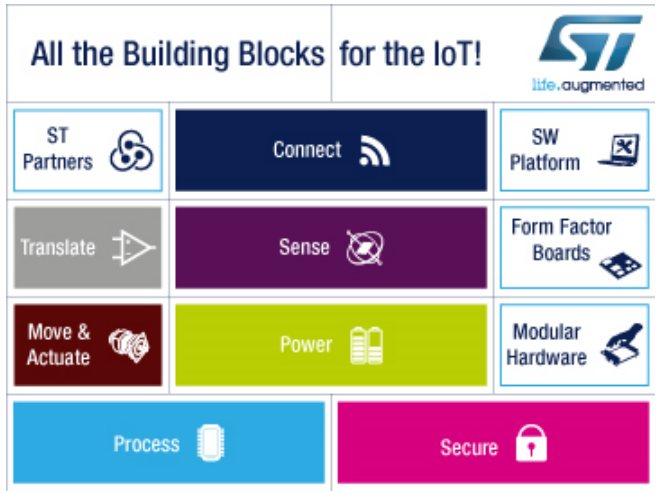
“Thing” you know how to build.

Plus what “Smart”?



Pieces from different sources and and no building instructions…

...or



life.augmented

Supporting the IoT Movement

Connect

Sense

Translate

Move
Actuate

Power

Secure

Process

SensorTile

BlueCoin

SmarTAG

Discovery Kit IoT Node

STM32 Nucleo Development
& Expansion boards

Pre-integrated SW for vertical applications

Smart Things

Smart Home

Smart City

Smart Industry

Development Ecosystem

Code generators

Prototyping software

IDE

Development environments

Debug solutions

Simulation and analysis tools

On-line design tools

ST Community

life.augmented

Partner Program

life.augmented

STM32L475 Discovery Kit IoT Node

B-L475E-IOT01A

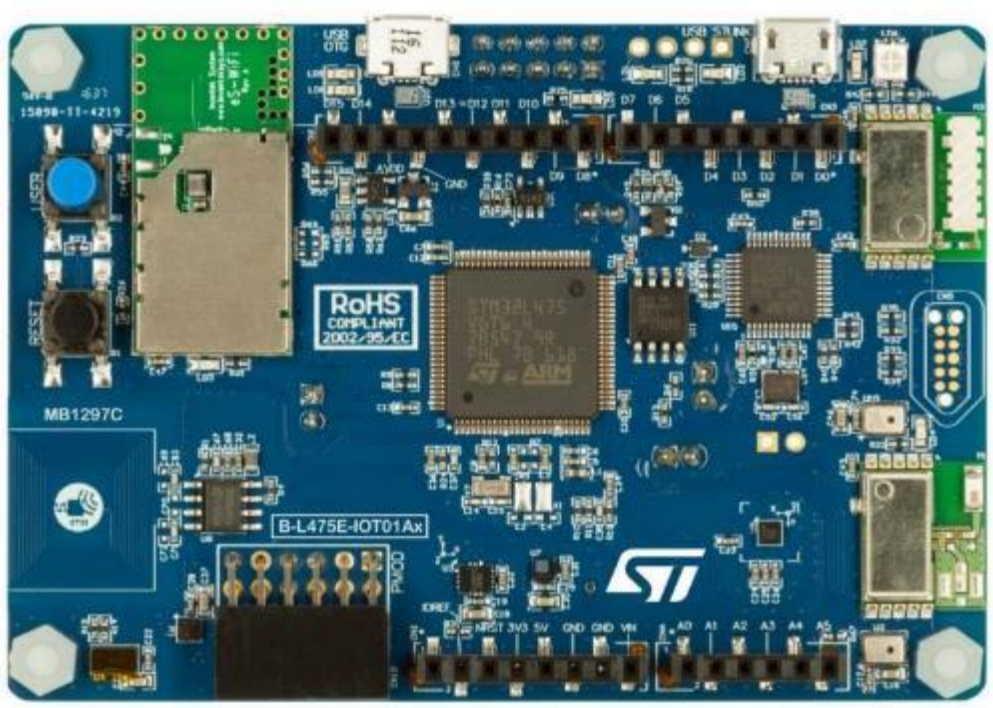
SW Libraries for STM32L4 MCU & Sensors

Low-power long-range communication (SubGHz)

Direct Wi-Fi connection to cloud servers

Environmental awareness: humidity, pressure, temp

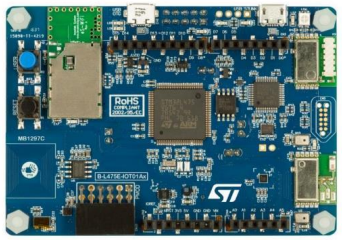
Detection hub: motion, proximity, audio



life.augmented

Workshop Deliverables

- For the workshop ST will provide



Discovery Kit IoT node

https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-discovery-kits/b-l475e-iot01a.html

USB 2.0 A-Male to Micro B Cable

STM32L4 Discovery Kit IoT Node

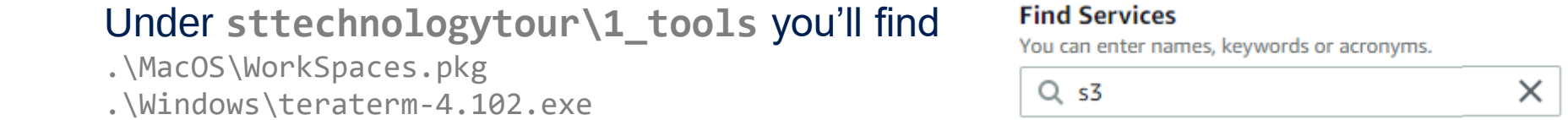
- STM32L4 MCU enables your IoT projects with the combination of ultra-low power and high performance
- STM32L4 Discovery Kit IoT is a compact, yet powerful board to explore various connectivity options
- Next Steps
  - You can add BLE, NFC and sensors features to the Amazon FreeRTOS project to create an application that can connect to the cloud and also has short range connectivity with your smartphone
  - Post your projects or ideas on the ST Community website to gather feedback and get support: https://community.st.com/community/share-your-activities/pages/overview
  - Refer to Amazon FreeRTOS user guide for more examples.

Lab Deliverables

Training Material on AWS S3 Bucket (USB Key Backup)

Navigate to: https://sttt.signin.aws.amazon.com/console with credentials distributed to you
In **Find Services** search box type **“S3”**
Under `sttechnologytour\1_tools` you’ll find

```
.\MacOS\WorkSpaces.pkg
.\Windows\teraterm-4.102.exe
.\Windows\Amazon+WorkSpaces.msi
.\Windows\7\en.stsw-link009.zip
.\en.stm32cubeprog.zip
.\This presentation
```

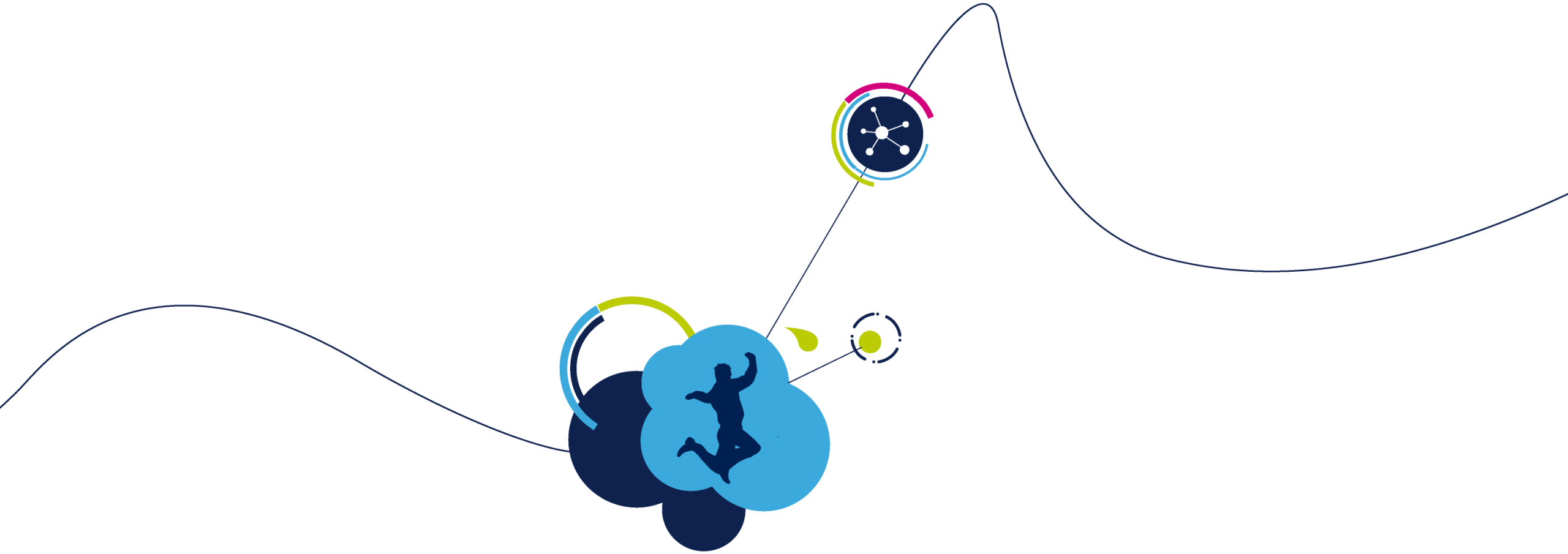


Install `WorkSpaces.pkg`

Install `Amazon+Workspaces.msi`
Unzip `en.stsw-link009.zip` and run `dpinst_amd64.exe` (W7 Only)

No Client available for Linux (Web Browser)

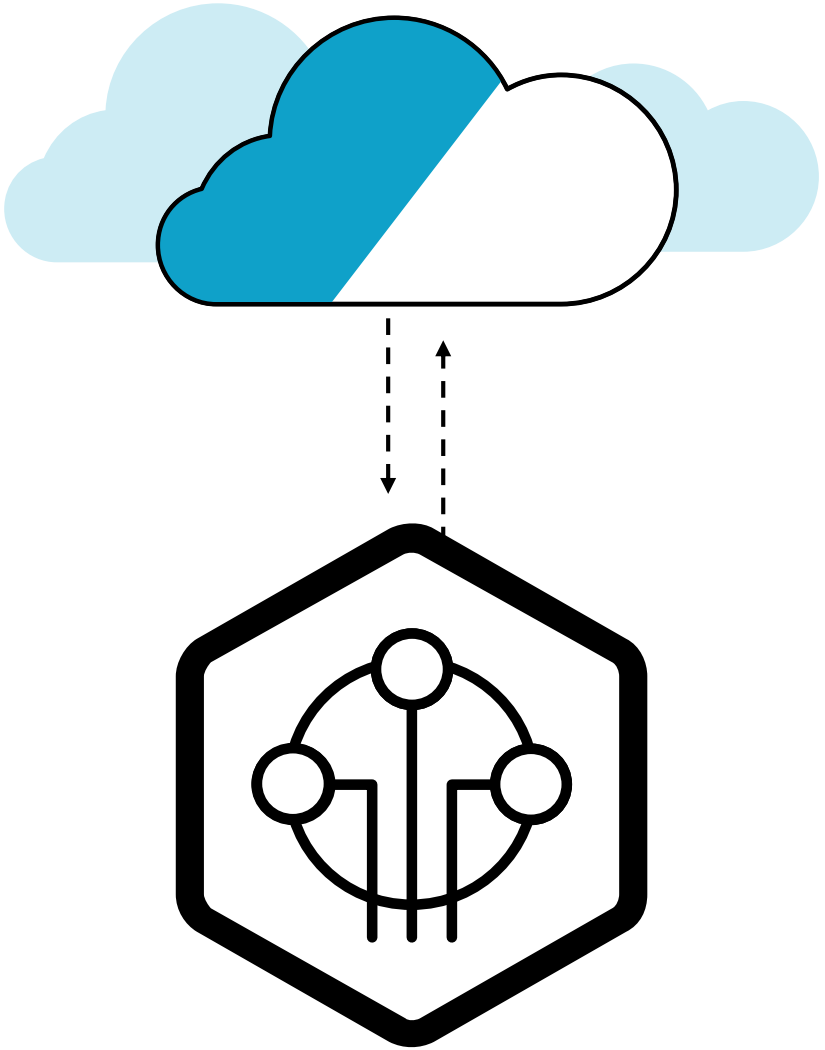
life.augmented



Amazon Web Services

life.augmented

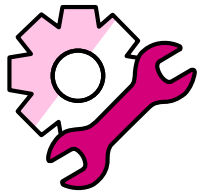
IoT on AWS



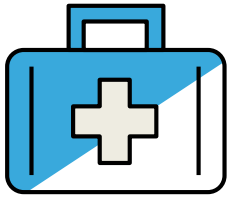
If you knew **the state of everything** and could **reason on top of that data**…

what problems would you solve?

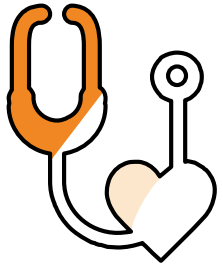
What customers are doing with AWS IoT



Predictive maintenance

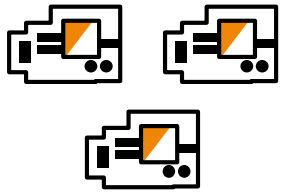


Wellness & health solutions

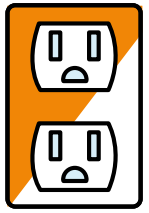


Remote patient monitor

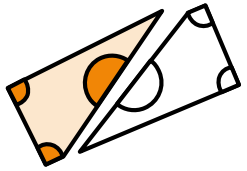
Connected buildings & city systems



Maintain device fleets



Monitor energy efficiency



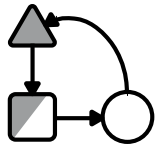
IoT payment & connected commerce

Safeguard manufacturing facilities

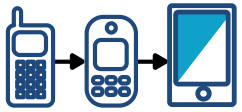
Nobody just buys IoT technology…

they seek business outcomes

Business outcomes with IoT



New services & business models



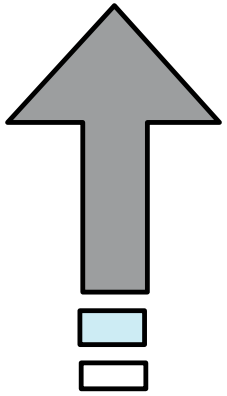
Products that get better with time

Better relationship with customers

Increased efficiency

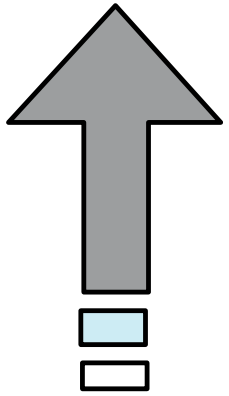
Intelligent decision making

Data driven discipline



Revenue growth

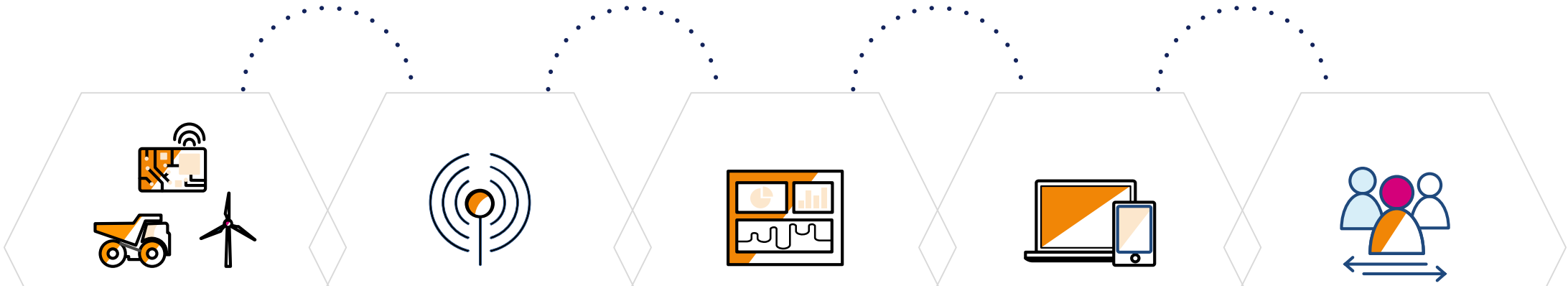
IoT data drives business growth



Operational efficiency

IoT data decreases OpEx

IoT solutions are complex & multidimensional



Devices & sensors

Onboard, provision, manage

Connectivity & infrastructure

Connect, communicate, secure

Analytics & insights

Analyze, visualize, act

Applications & services

Engage, empower, delight

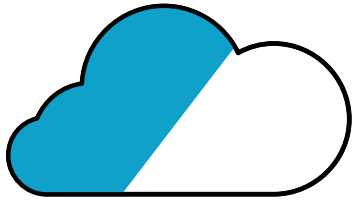
Change management

Transform, shift culture

Our concept of IoT

Things
Sense & Act

Cloud
Storage & Compute

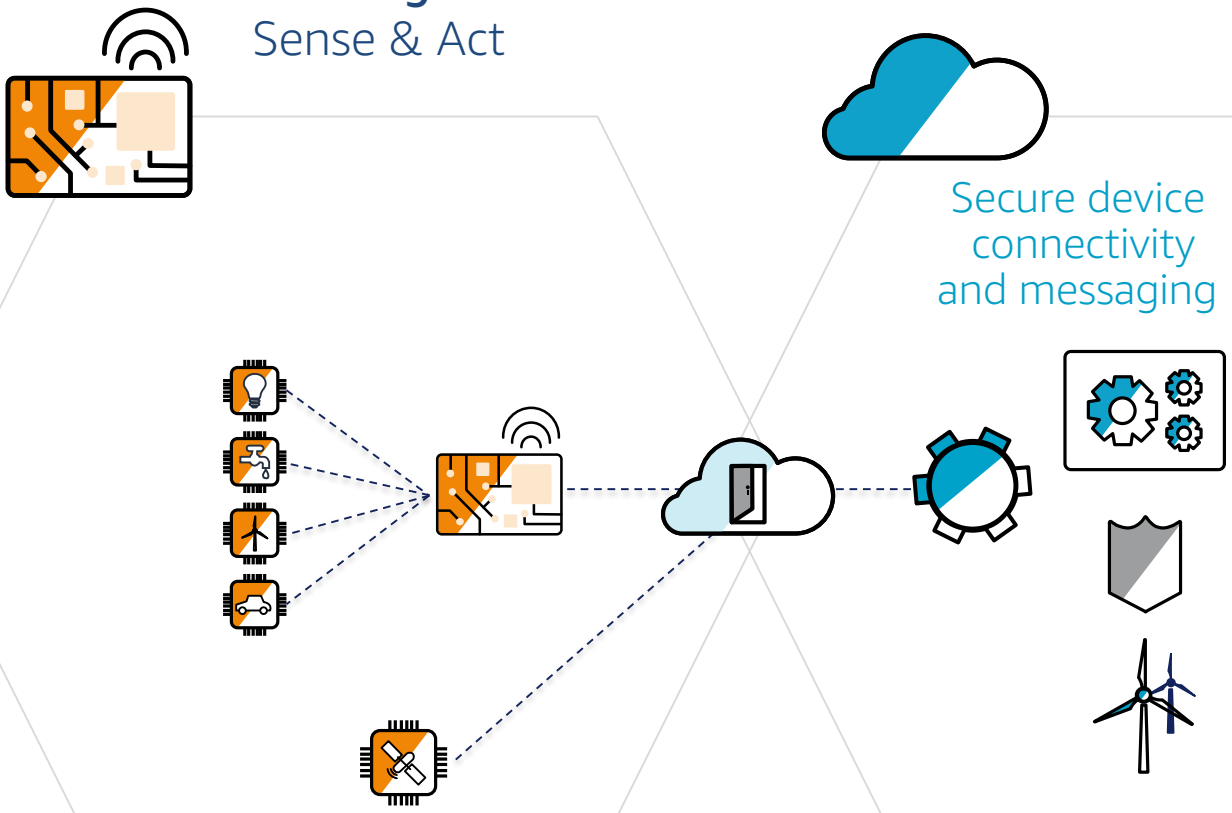


Intelligence
Insights & Logic → Action

AWS IoT Architecture

Things
Sense & Act

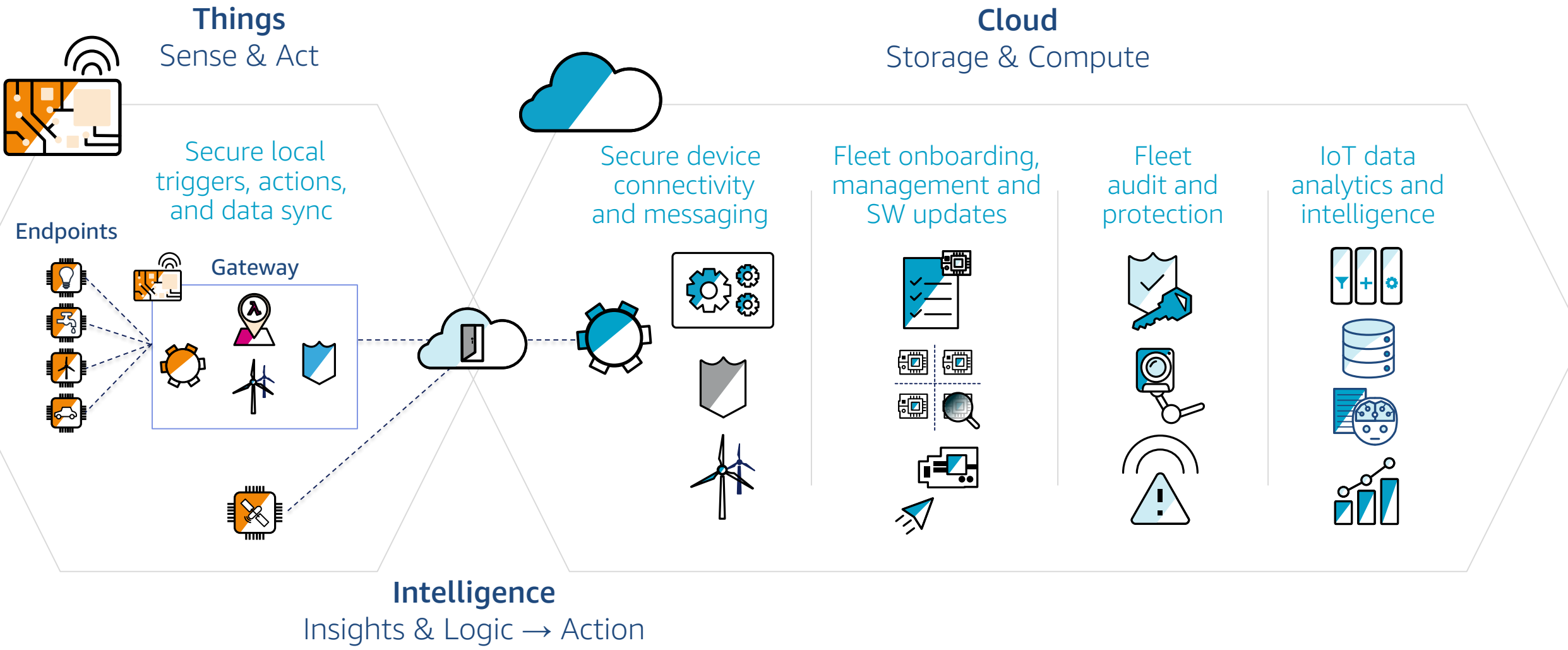
Cloud
Storage & Compute



Secure device connectivity and messaging

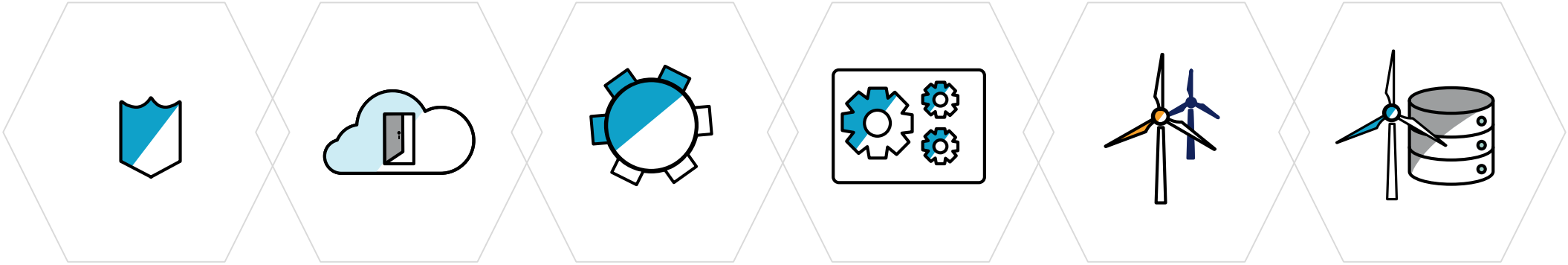
Intelligence
Insights & Logic → Action

AWS IoT Architecture



AWS IoT Core

Secure Device Connectivity and Messaging



Identity Service

Device Gateway

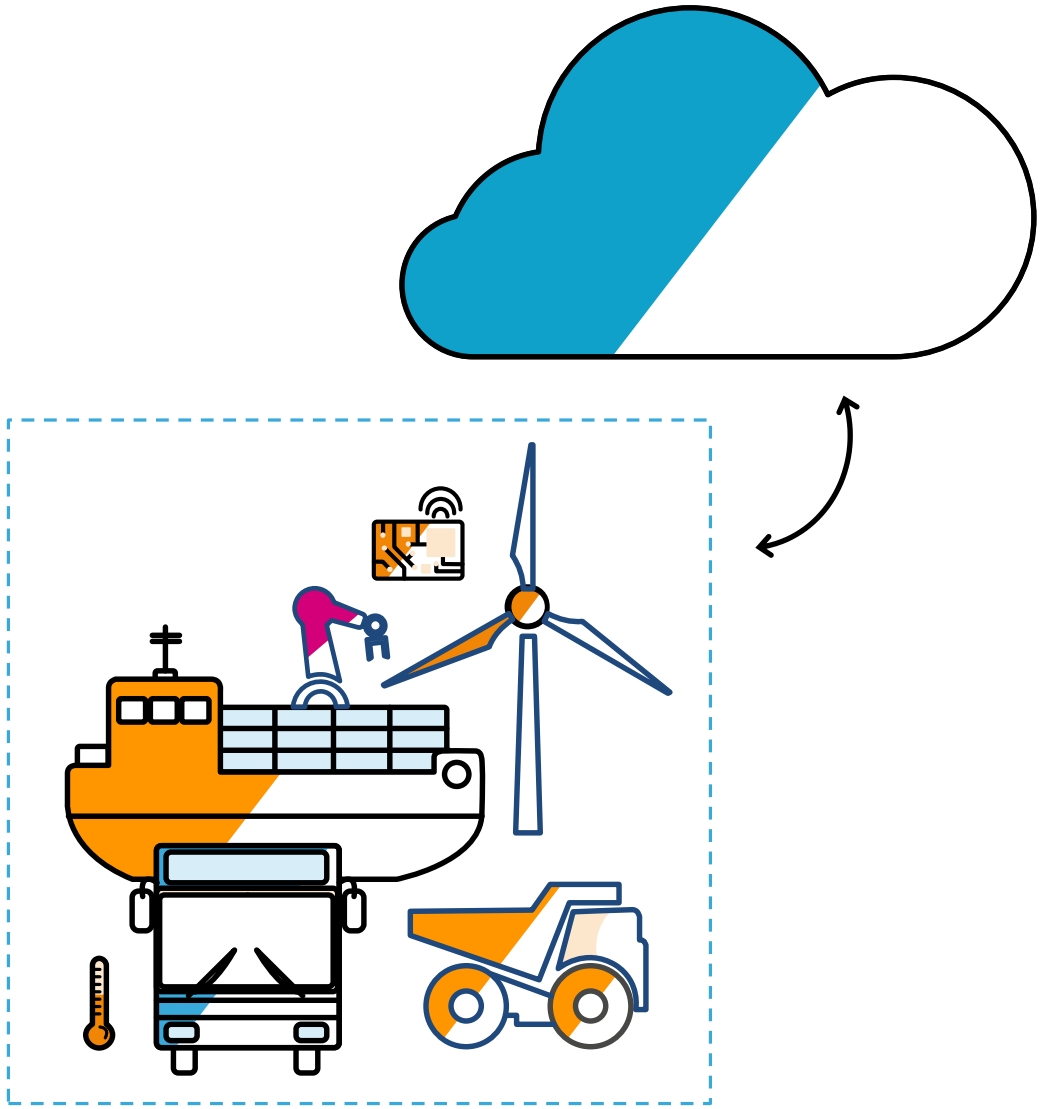
Message Broker

Rules Engine

Device Shadow

Registry

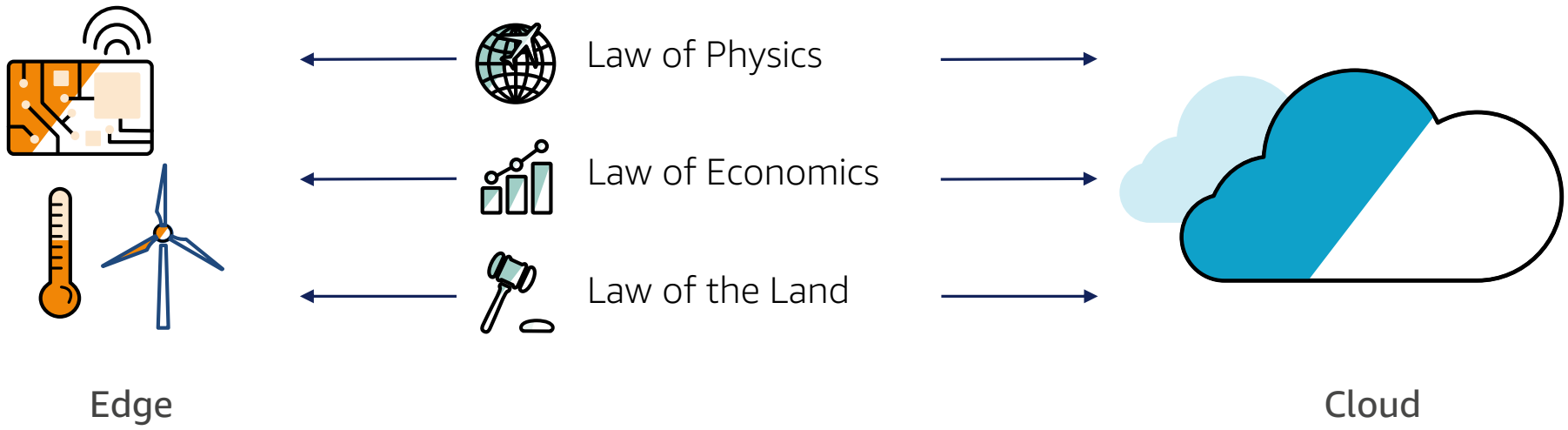
How can I extend AWS cloud capabilities to the edge?



AWS Greengrass

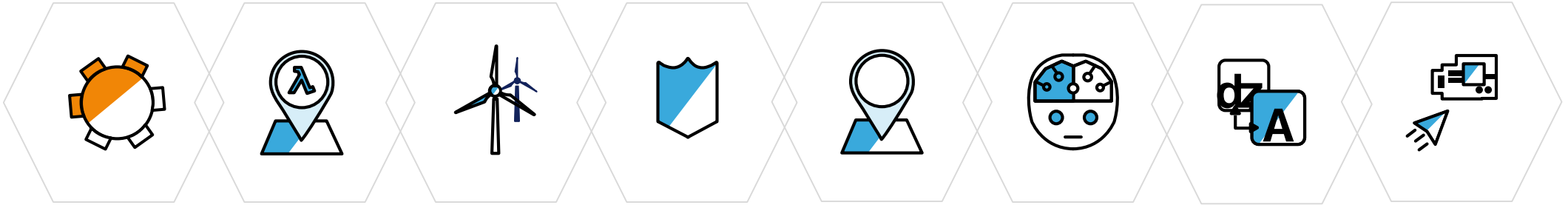
Extend AWS IoT to the Edge

AWS Greengrass extends AWS IoT onto your devices, so that they can act locally on the data they generate, while still taking advantage of the cloud.



AWS Greengrass

Extend AWS IoT to the Edge



Local Messages and Triggers

Local Message Broker

Local Actions

Lambda Functions

Data and State Sync

Local Device Shadows

Security

AWS-grade security

Local Resource Access

Lambdas Interact With Peripherals

Machine Learning Inference

Local Execution of ML Models

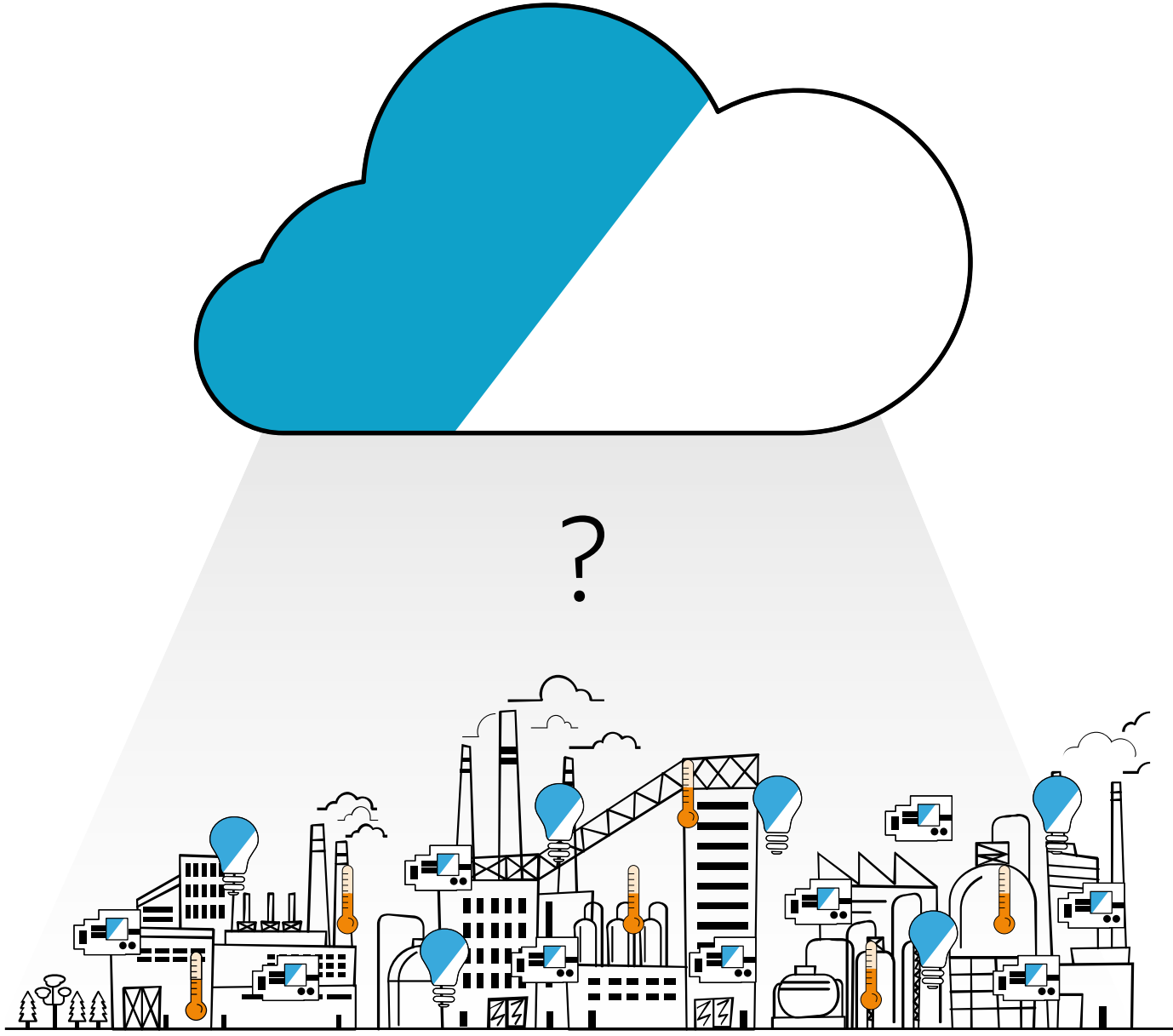
Protocol Adapters

Easy Integrations With Local Protocols

Over the Air Updates

Easily Update Greengrass Core

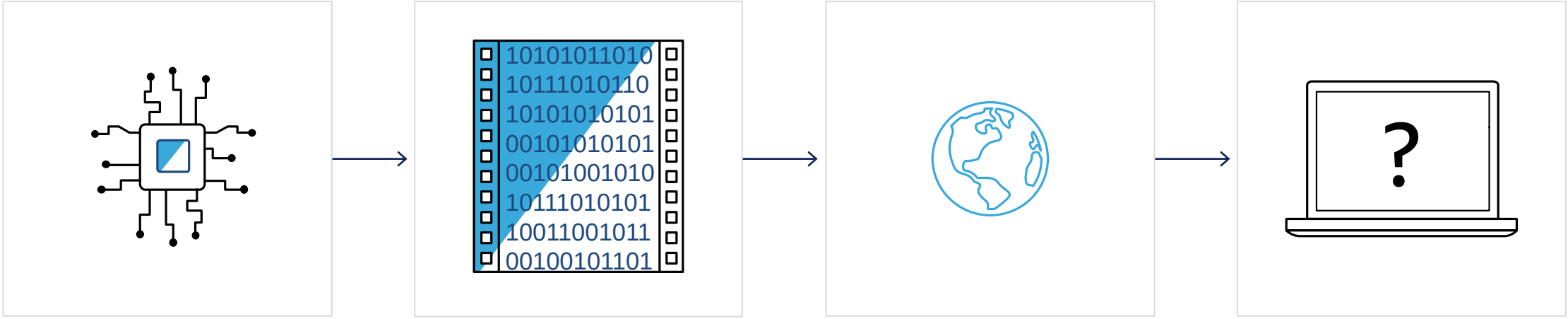
How can I securely connect constrained, microcontroller-based devices?



Amazon FreeRTOS

IoT Operating System for Microcontrollers

Amazon FreeRTOS, based on the popular FreeRTOS, is a microcontroller operating system that makes small, low powered edge devices easy to program, deploy, secure, connect, and maintain.



Will it work on my chip?

Does it have the functionality I need?

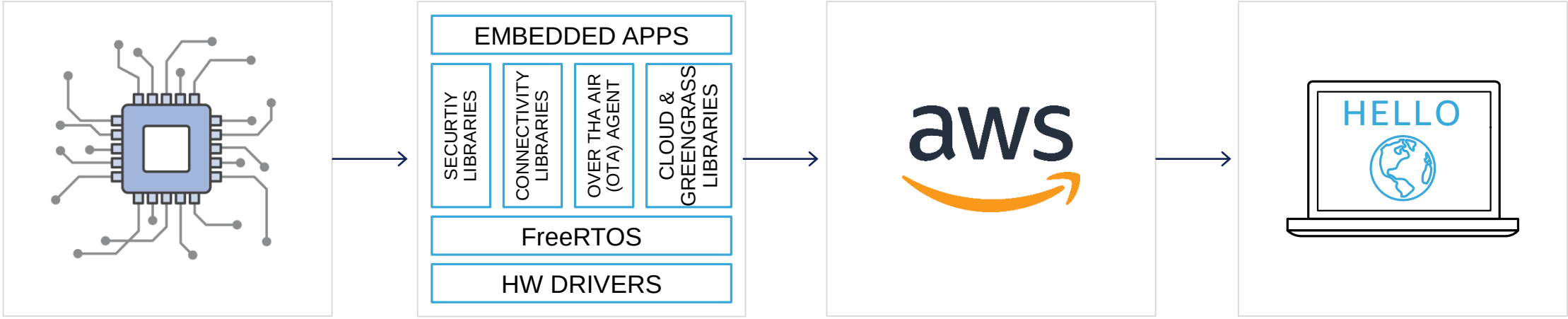
Where do I get it?

How do I start?

Amazon FreeRTOS

IoT Operating System for Microcontrollers

Amazon FreeRTOS, based on the popular FreeRTOS, is a microcontroller operating system that makes small, low powered edge devices easy to program, deploy, secure, connect, and maintain.



Will it work on my chip?

Does it have the functionality I need?

Where do I get it?

How do I start?

Amazon FreeRTOS

IoT Microcontroller OS

Local Connectivity Libraries

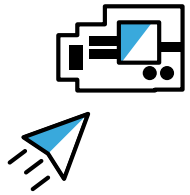
AWS Greengrass

Based on FreeRTOS kernel

Cloud Connectivity Libraries

AWS IoT Core

Security Connectivity Libraries



OTA Beta & Code Signing

Based on #1 Real-Time Operating System for Microcontrollers

- 15 years, trusted, and widely distributed
- 40+ supported architectures
- Broad ecosystem support
- Free and open source
- Introducing version 10
- MIT Open Source License
- Improved Inter-Process Communication (IPC) capabilities with stream and message buffers

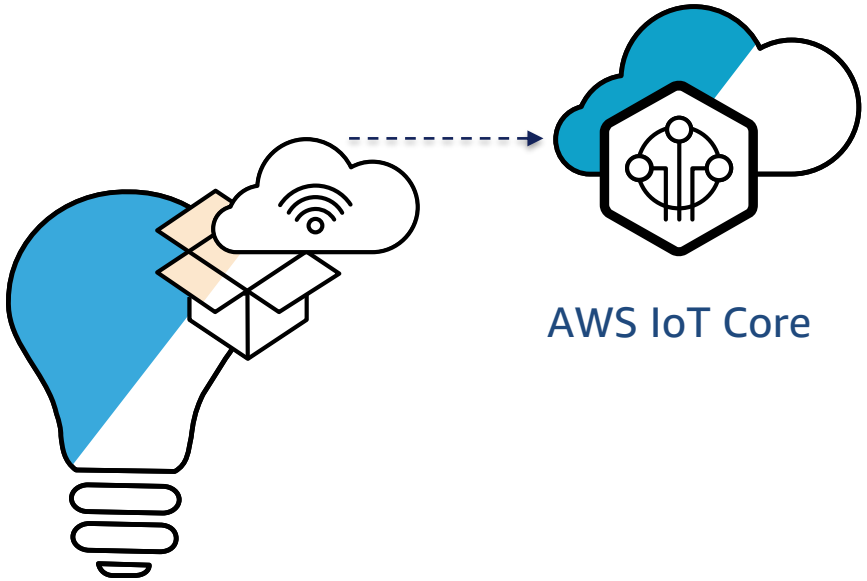
Local Connectivity Libraries

Connect with AWS Greengrass

- Local communication with edge gateways and a Wi-Fi stack, including AWS Greengrass discovery support
- Wi-Fi management library implements an abstraction layer for Wi-Fi features such as setup, configuration, provisioning, security, and power management
- Continue communicating, collecting data, and taking actions without a cloud connection
- Support for many network topologies and use cases

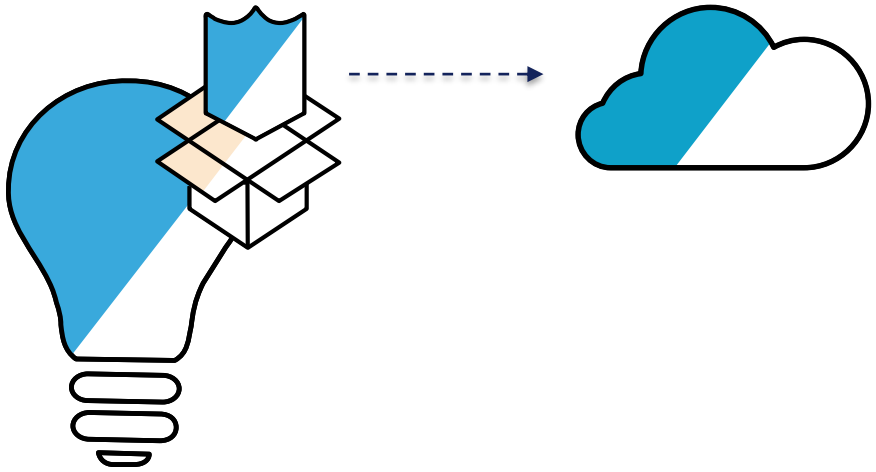
Cloud Connectivity Libraries

- Connectivity to AWS IoT Core
- MQTT Pub/Sub messaging
- Device Shadow support
- Take advantage of IoT Core benefits like IoT Device Management, scalable architecture, and pay as you go pricing
- Fastest way to get started on IoT microcontrollers



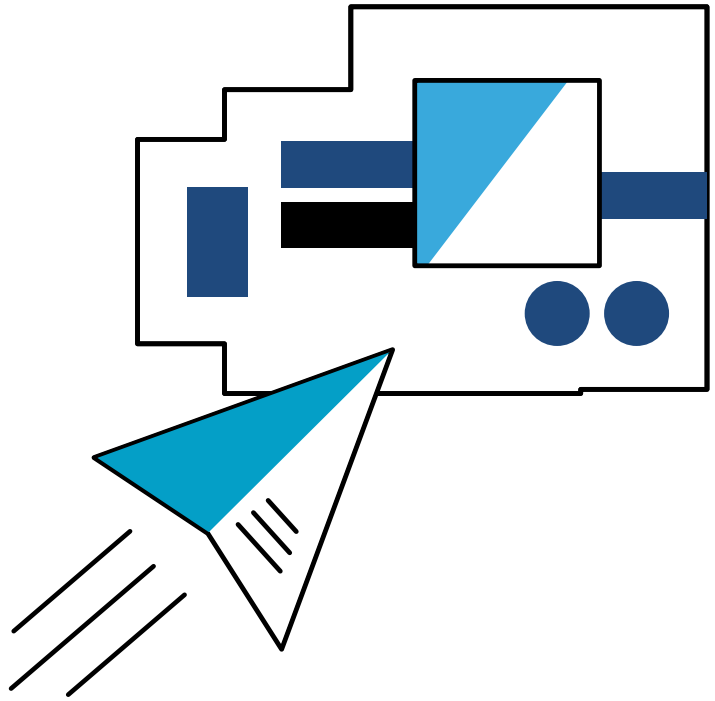
Security Connectivity Libraries

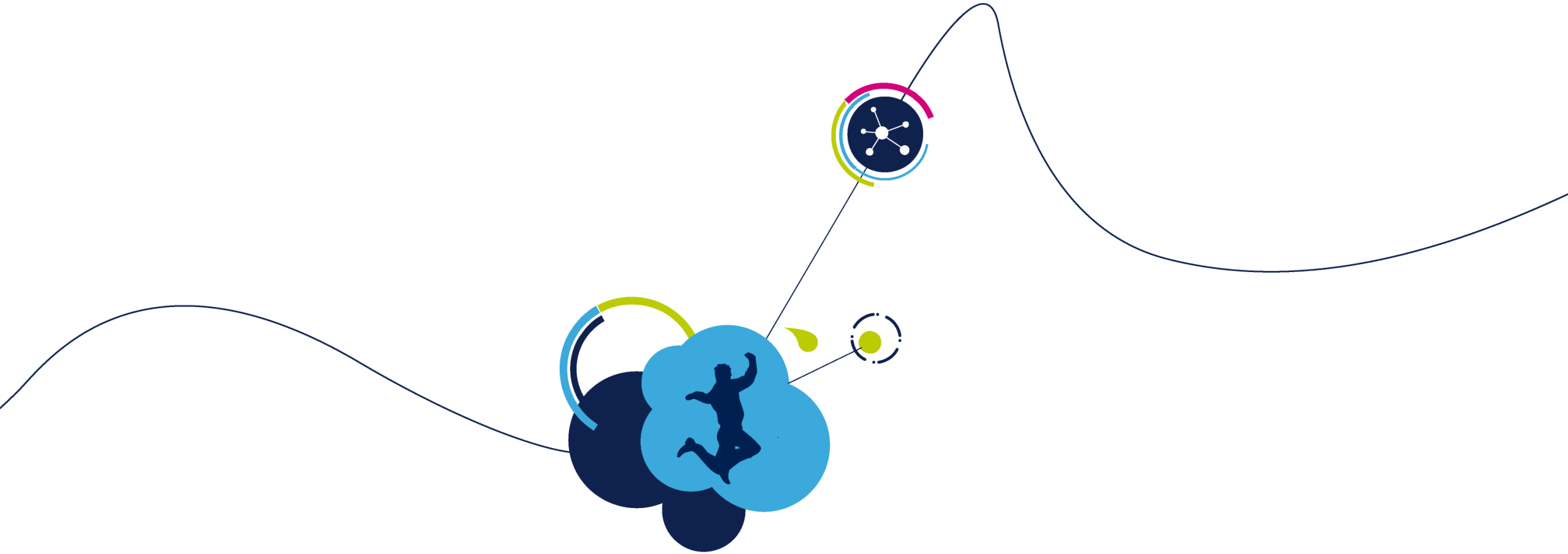
- Secure sockets using TLS
- Certificate-based authentication
- PKCS#11 interface for key management
- Secure by default
- No open network ports
- Only run trusted code
- Clear, modular implementation



Over-the-Air Firmware Updates

- Use AWS IoT Device Management to assign updates to groups
- Code sign new firmware images
- Stream updates to your device over MQTT
- Validate signature on device
- APIs to control installation and reboot logic
- Simple to manage groups
- Control authorship and ensure devices only run trusted code
- Memory efficient updated client



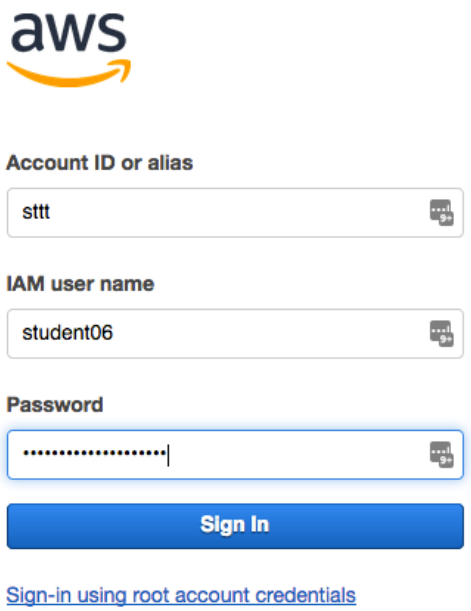


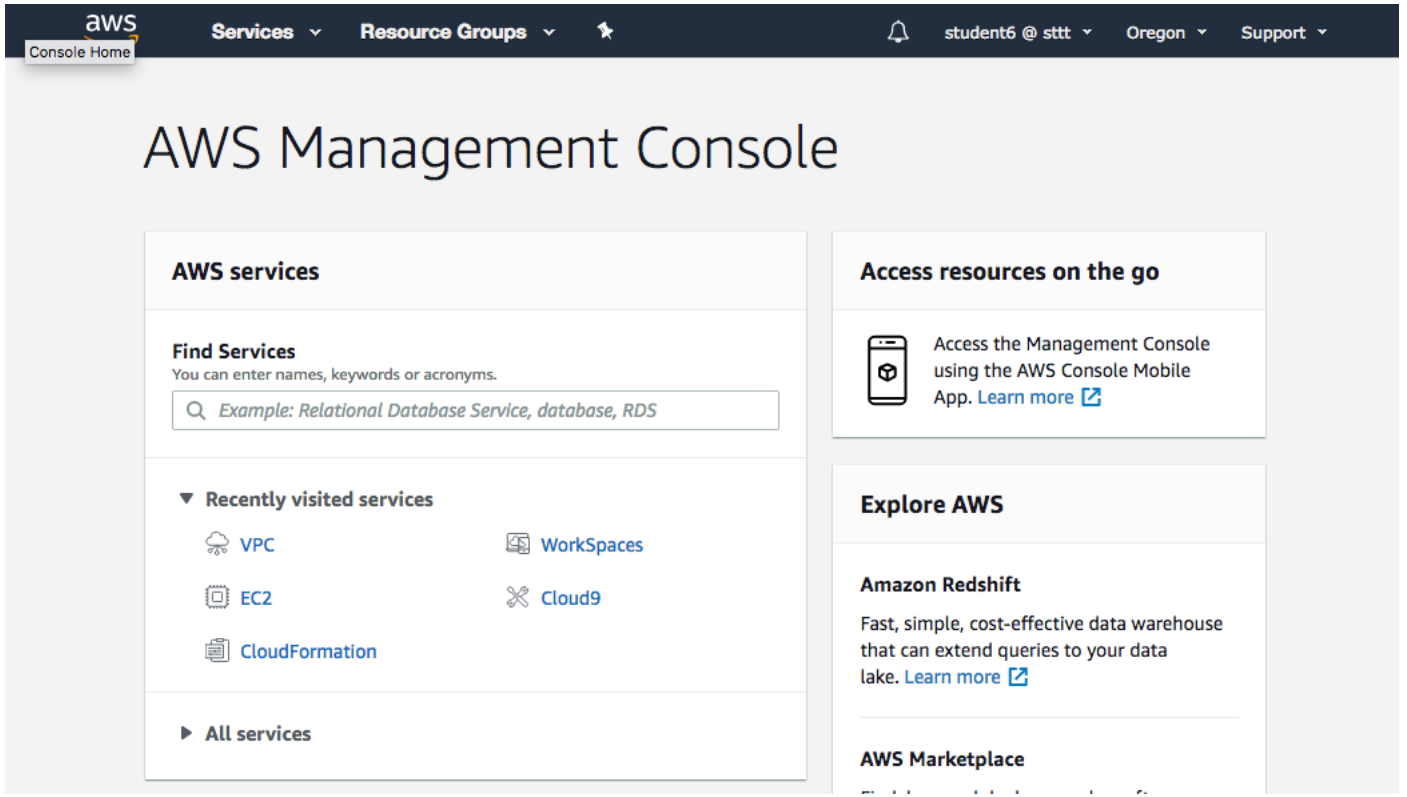
Let’s get started!

life.augmented

AWS Console Login

- Navigate to: https://sttt.signin.aws.amazon.com/console
- Login with credentials distributed to you

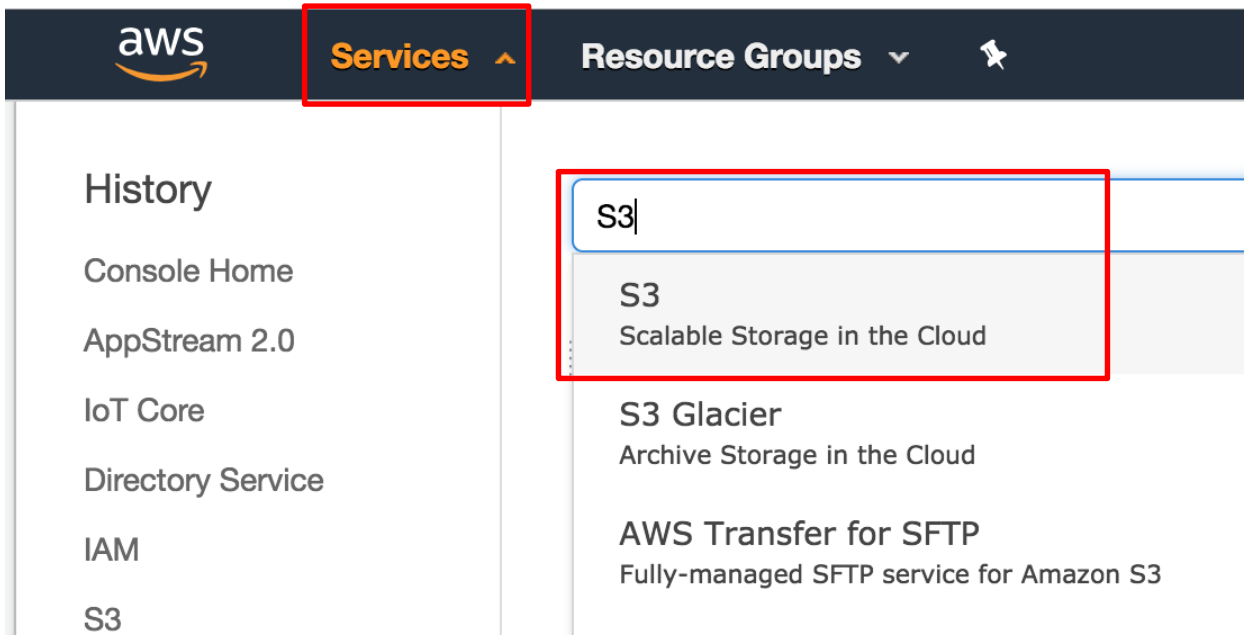




life.augmented

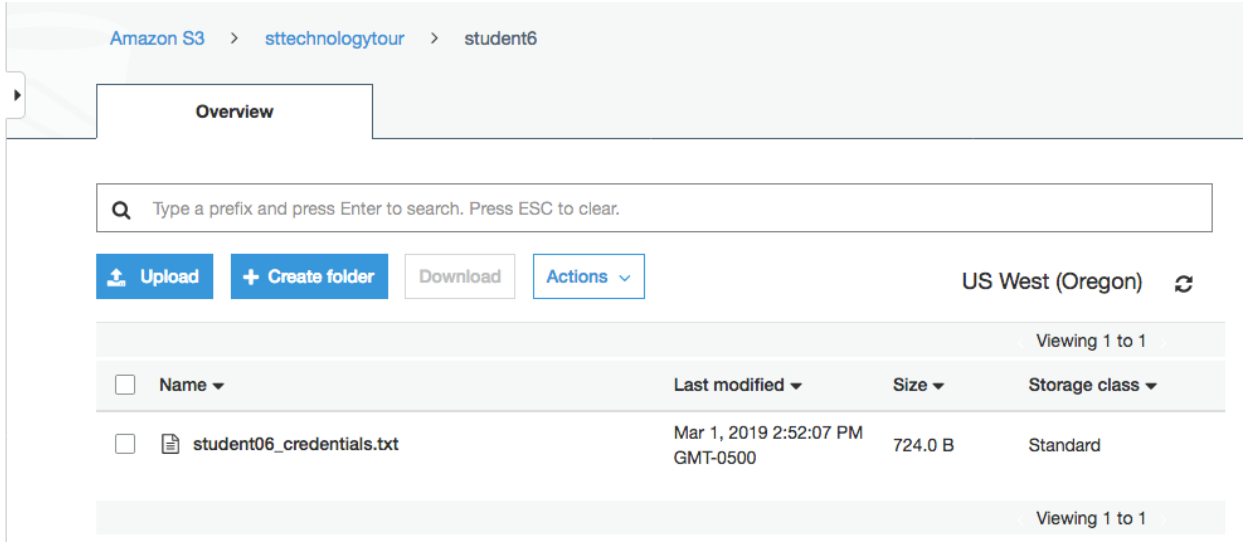
Get the credentials file from S3 bucket

- On your laptop open the browser and login to AWS Console
- Navigate to S3 service (Services -> type S3 -> Select S3)



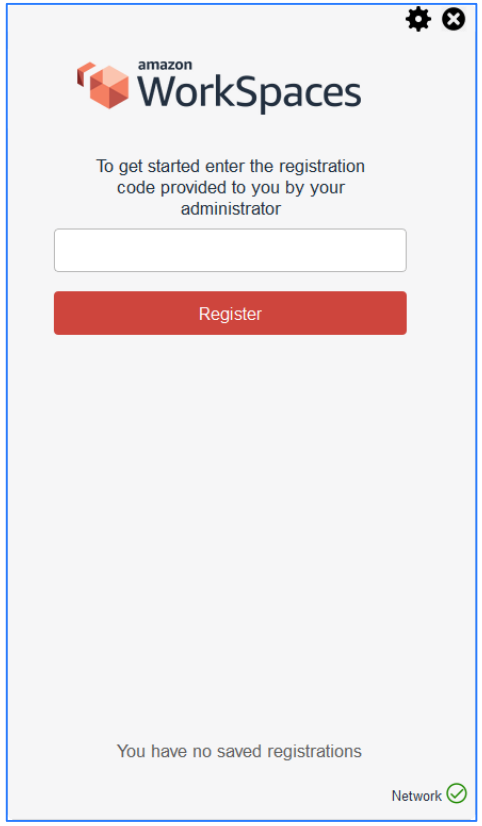
Get the credentials file from S3 bucket

- Open `sttechnologytour/studentXX` folder
- Download `studentXX_credentials.txt` file

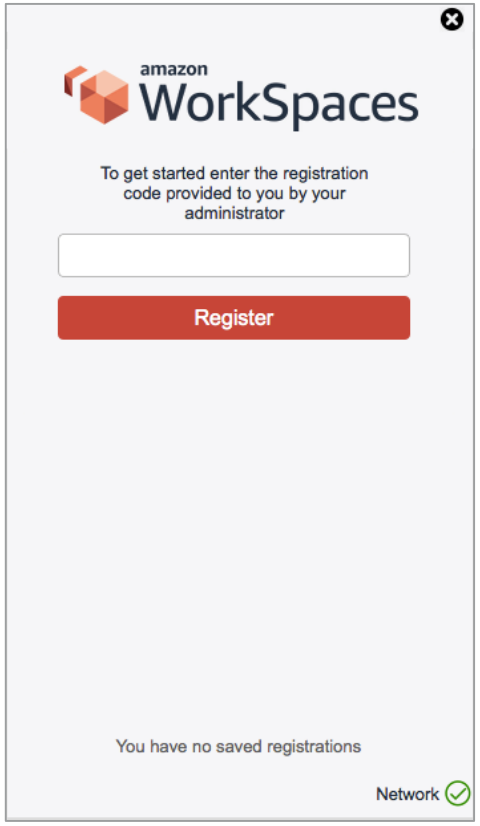



```
Workspaces Login:
Registration: WSpdx+P2D4ZD
student06: [redacted]

AWS Console Login:
https://714604847878.signin.aws.amazon.com/console

Access key: [redacted]
Secret key: [redacted]

GitHub URL: https://github.com/yona75/sttt-workshop
Branches: master, st-sensors-demo

Content for sttt-wooorkshop/tools/ws_config_quick_start/configure.json file:
{
    "afr_source_dir":"../..",
    "thing_name":"student06",
    "wifi_ssid":"staws2019",
    "wifi_password":"staws2019",
    "wifi_security":"eWiFiSecurityWPA2"
}

Line 90 in sttt-workshop/blob/master/demos/common/mqtt/aws_hello_world.c :
#define echoTOPIC_NAME          ( ( const uint8_t * ) "freertos/demos/echo/student06" )
```

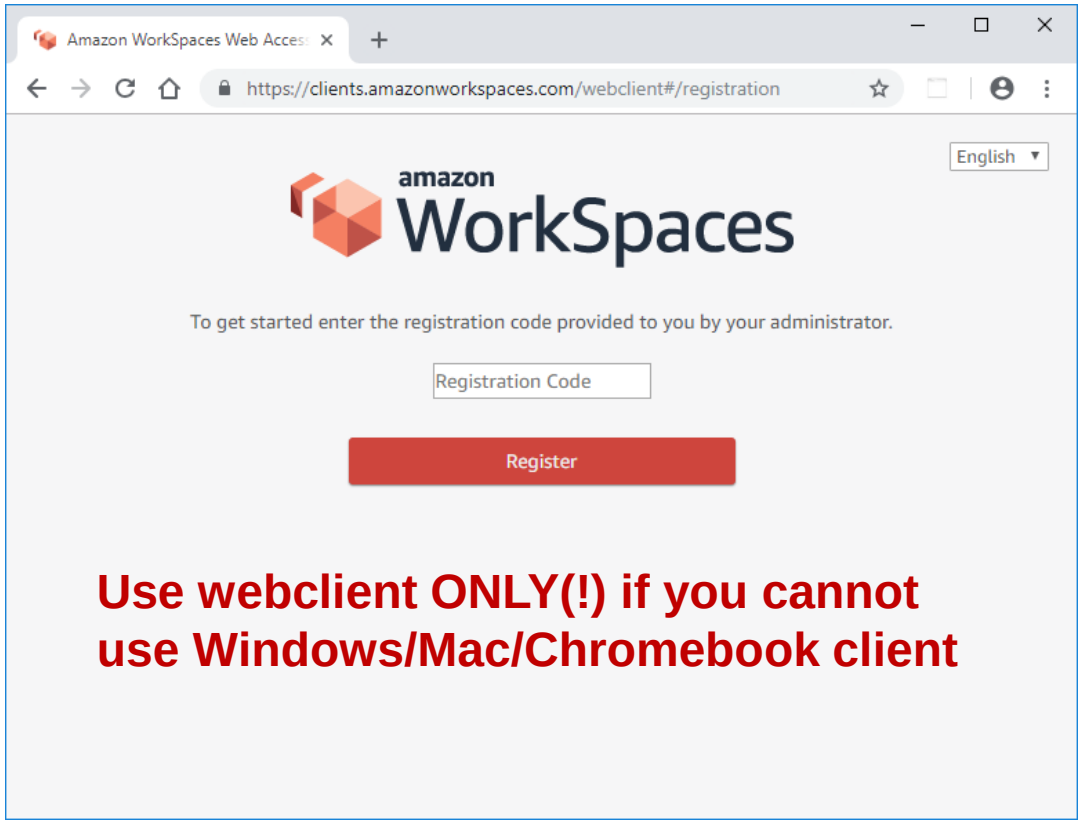
Set up Amazon Workspaces client



Windows
Desktop
Client
(W7, W8, W10)



MacOS
Desktop
Client
(Mac OS X 10.8.1 and above)



Use webclient ONLY(!) if you cannot use Windows/Mac/Chromebook client

Web Client
(Chrome or Firefox v48 or later)

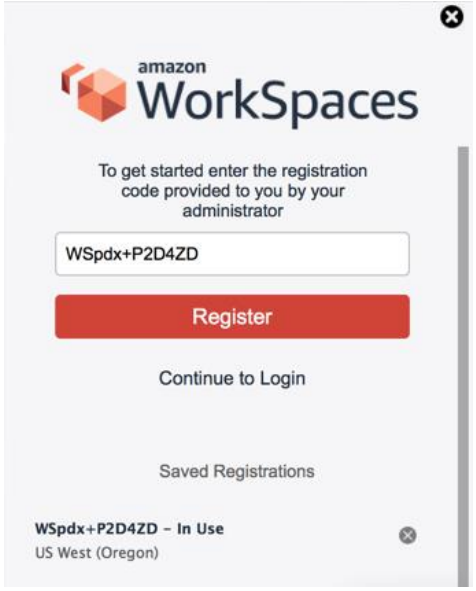
https://clients.amazonworkspaces.com/

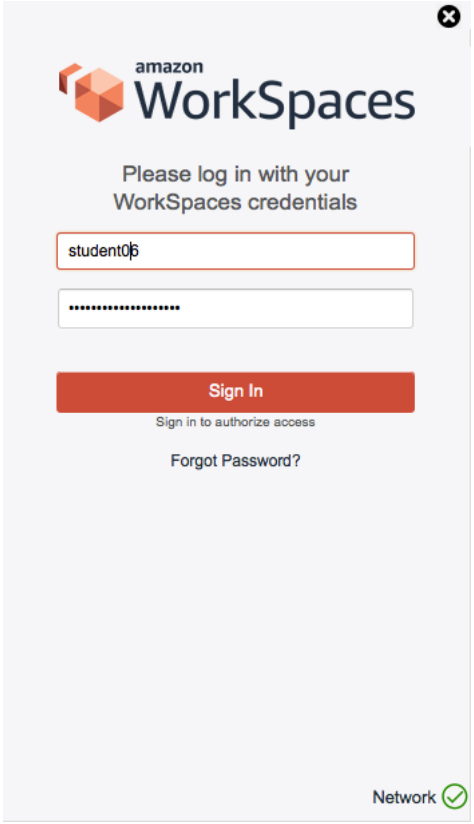
https://clients.amazonworkspaces.com/webclient

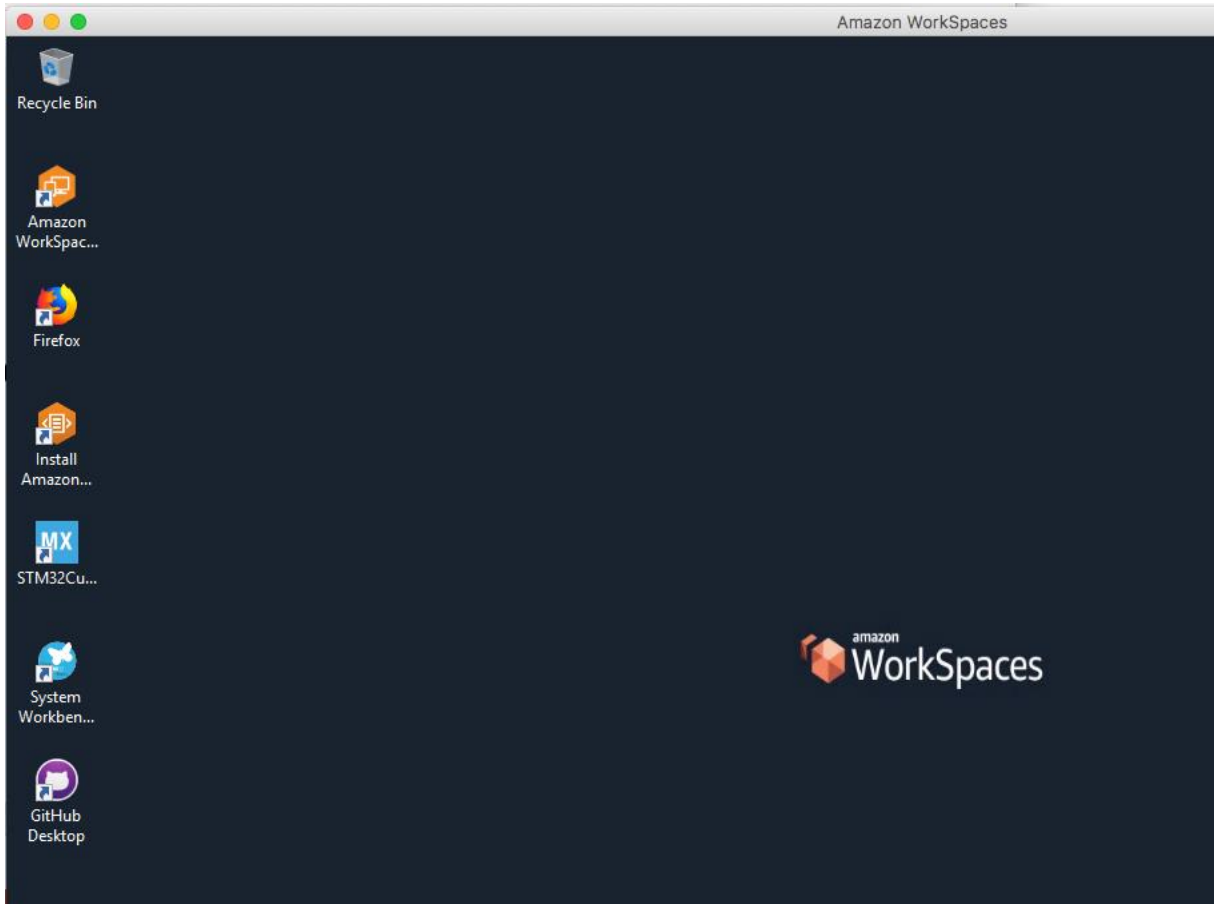
life.augmented

Log in to your Workspace

- Run client, click Options -> Manage Registrations, register with WSpdx+P2D4ZD
- Login with credentials you have in your file



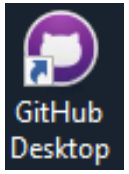




life.augmented

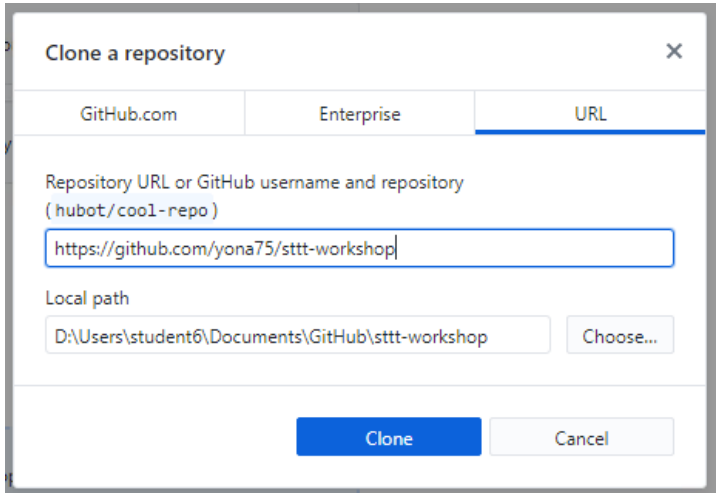
Clone the repository

- Run GitHub Desktop

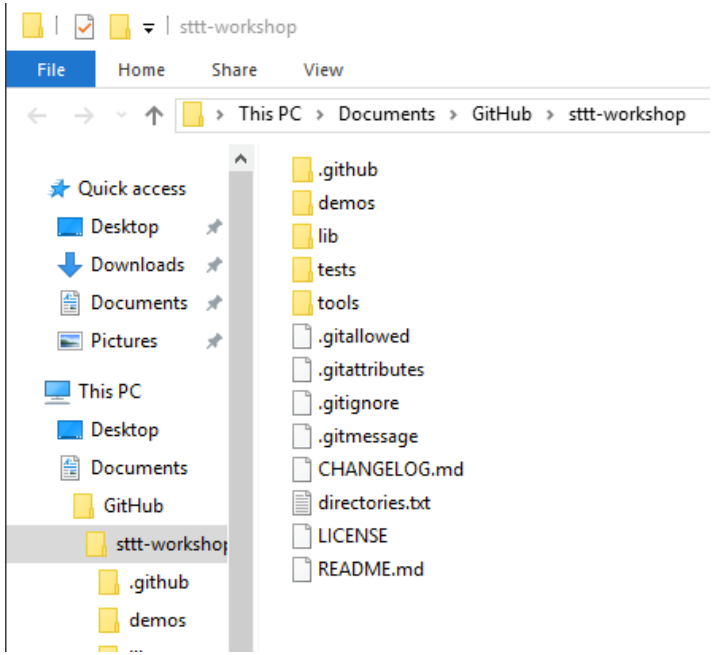


- Press *Skip this step*, then *Continue*, then *Finish*

- Click *Clone a repository from Internet …* and provide URL found in studentXX_credentials.txt file



Click *Show in Explorer*

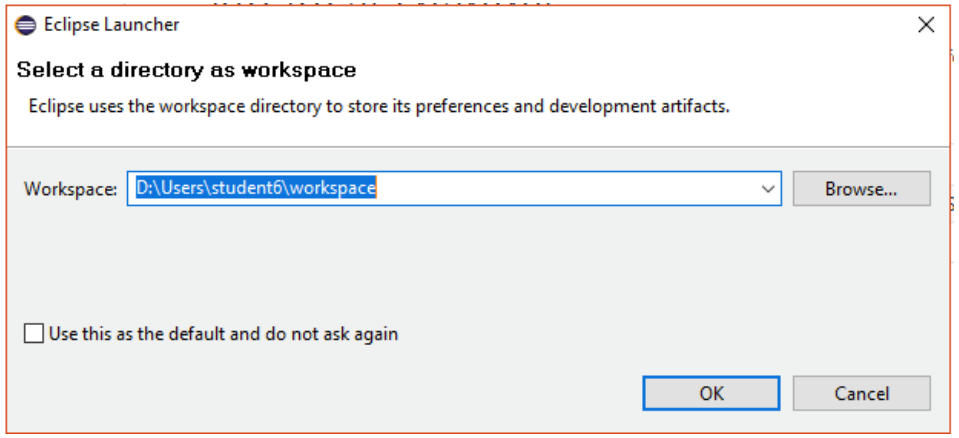


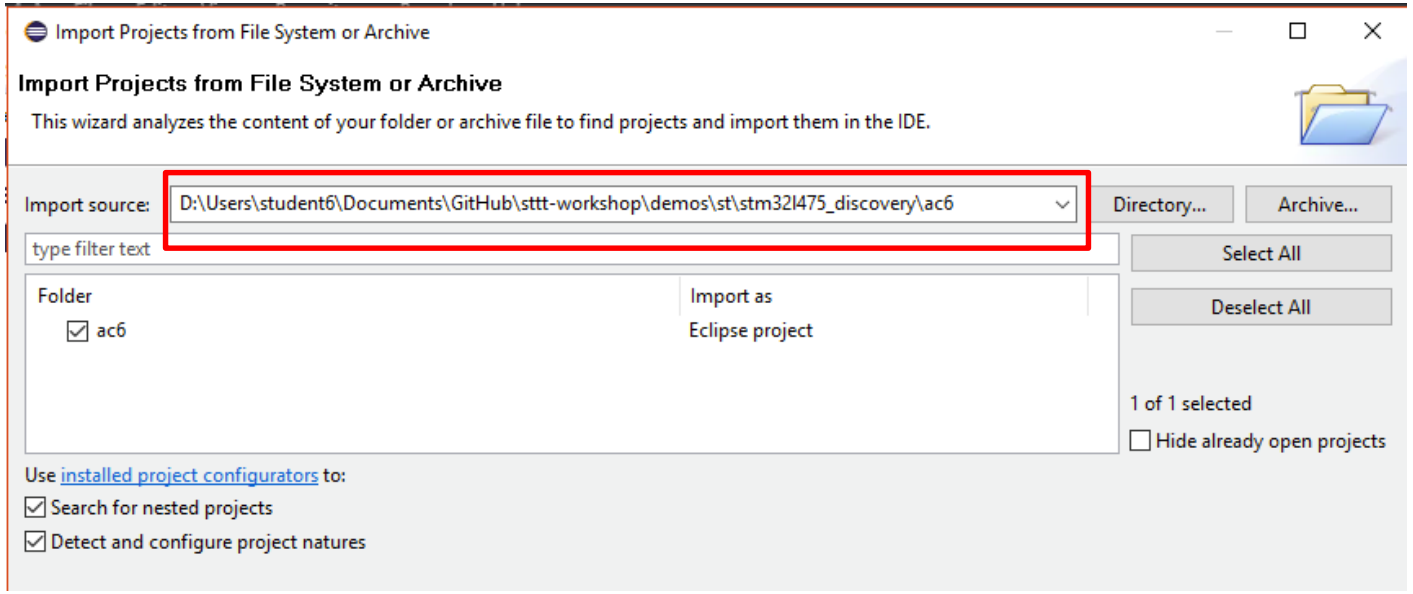
life.augmented

Import the project to System Workbench

- Run System Workbench for STM32

- Accept default location of your Eclipse workspace
- Click *File -> Open Projects from File System …*



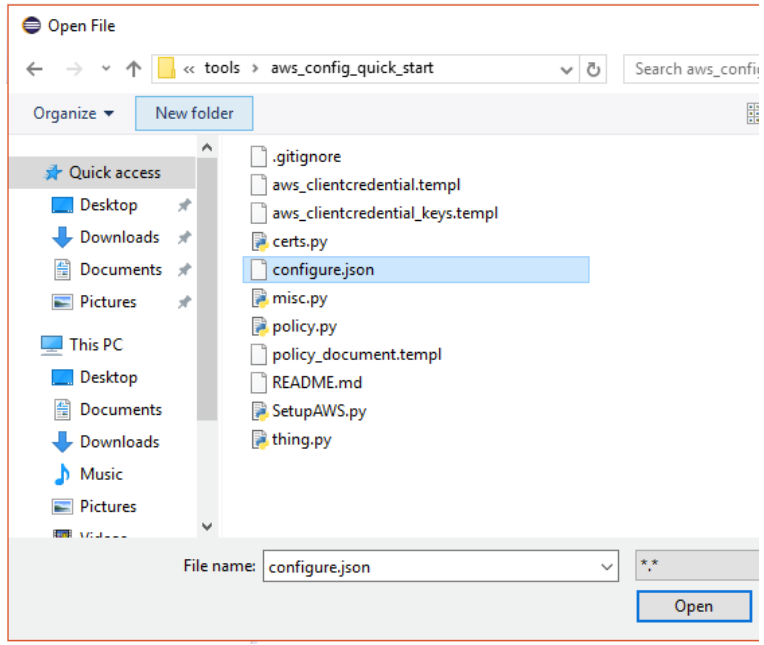


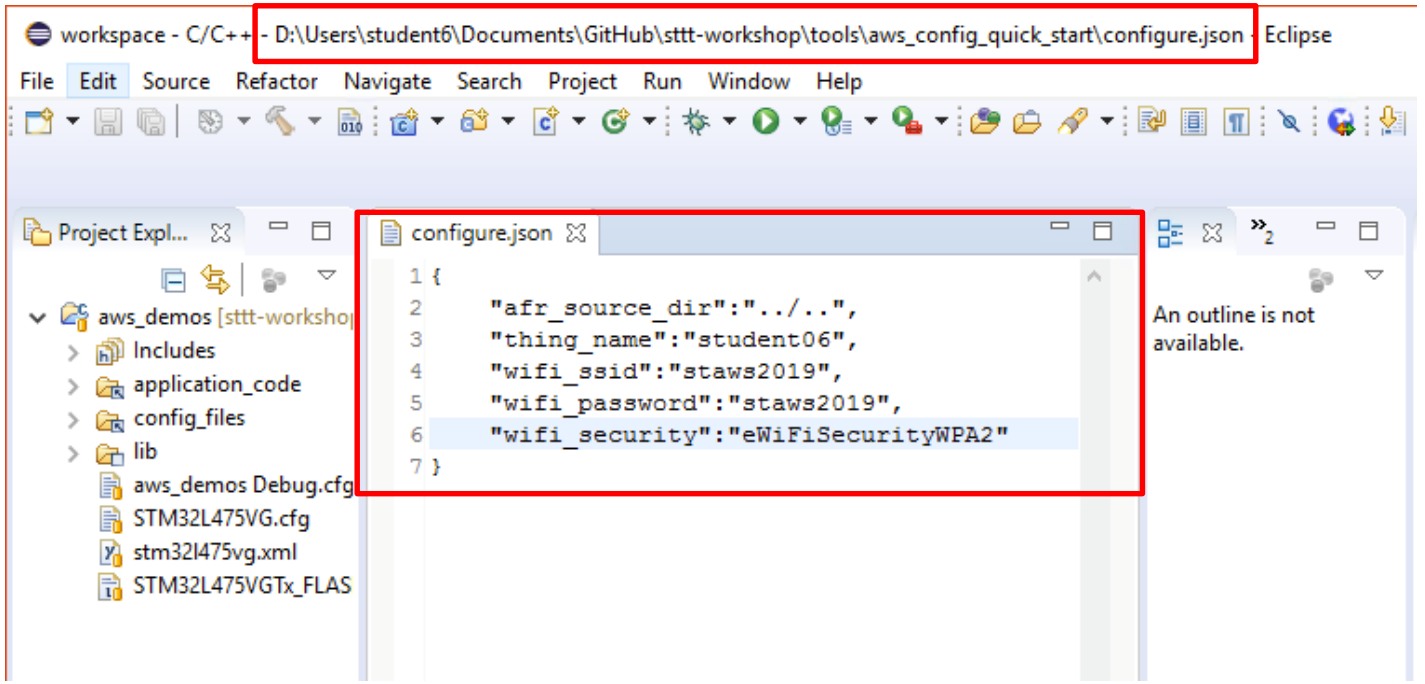
life.augmented

Provision Thing, and device credentials

- In System Workbench click *File -> Open file…* and navigate to *D:\Users\studentXX/Documents\GitHub\sttt-workshop/tools/aws_config_quick_start/configure.json*

- Edit it content to match example in *studentXX_credentials.txt and save the changes*

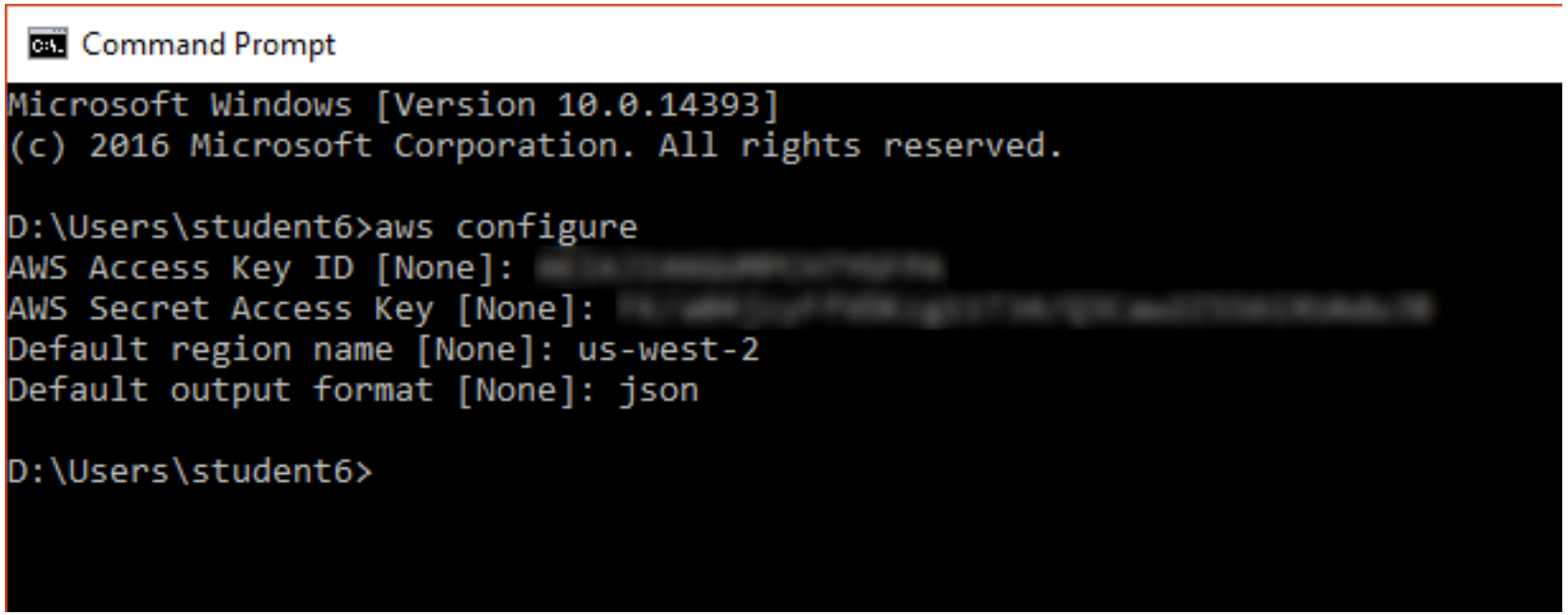




life.augmented

Provision Thing, and device credentials

- Run Windows *Command Prompt* and configure AWS CLI by issuing **aws configure** command (make sure the region is ***us-west-2*** and format is ***json***)
- AWS Access and Secret keys are in the *studentXX_credentials.txt* file



life.augmented

Provision Thing, and device credentials

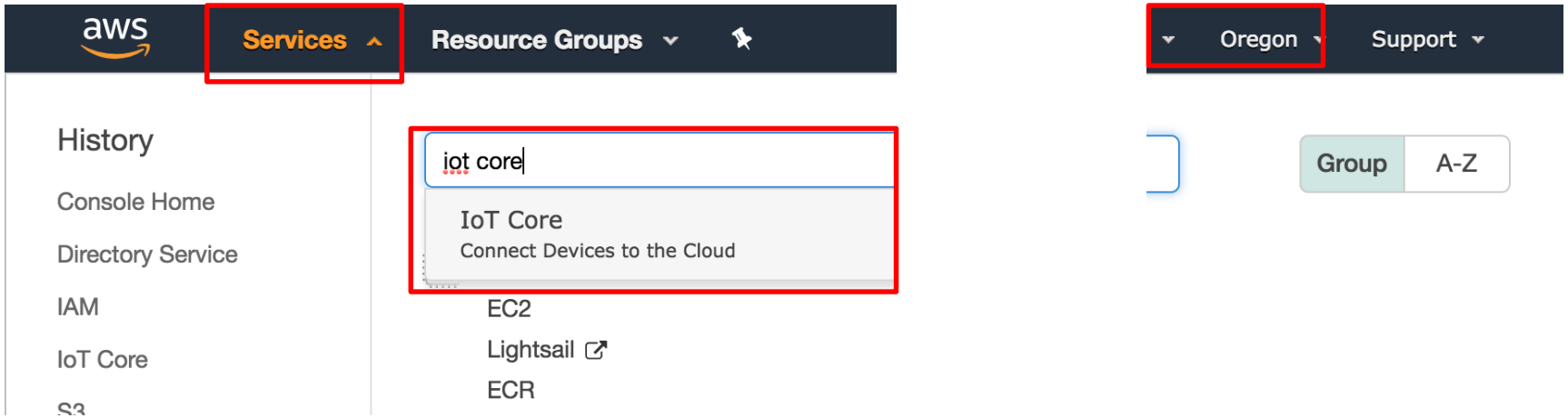
- Change directory to
  D:\Users\studentXX\Documents\GitHub\sttt-workshop\tools\aws_config_quick_start

- Run Provisioning script: *python SetupAWS.py setup*

```
D:\Users\student6>cd Documents\GitHub\sttt-workshop\tools\aws_config_quick_start

D:\Users\student6\Documents\GitHub\sttt-workshop\tools\aws_config_quick_start>python SetupAWS.py setup

D:\Users\student6\Documents\GitHub\sttt-workshop\tools\aws_config_quick_start>
```

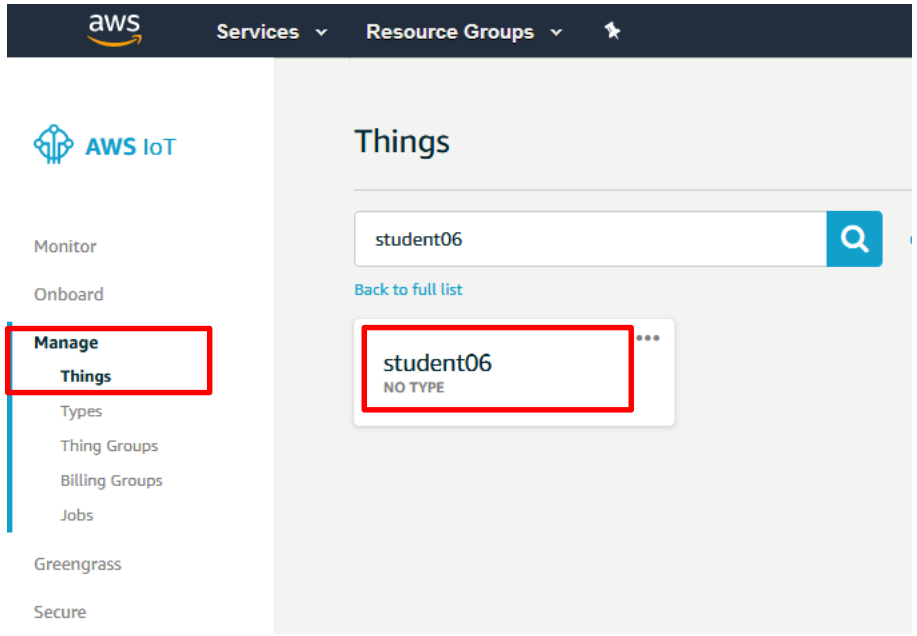
Provision Thing, and device credentials

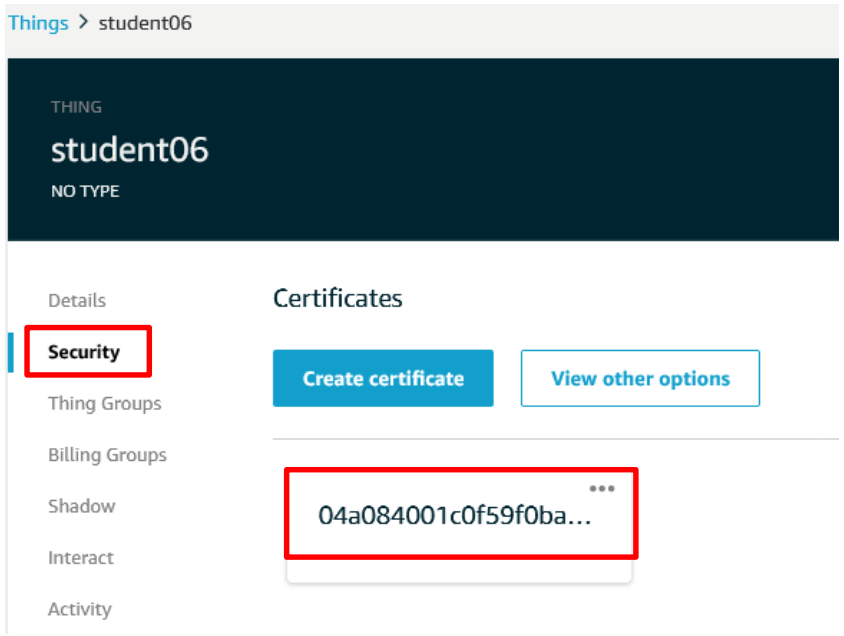
- Start Firefox browser
- In the AWS Console click *Services* and type *iot core*
- Select *IoT Core* service
- Make sure your region is set to Oregon.



Provision Thing, and device credentials

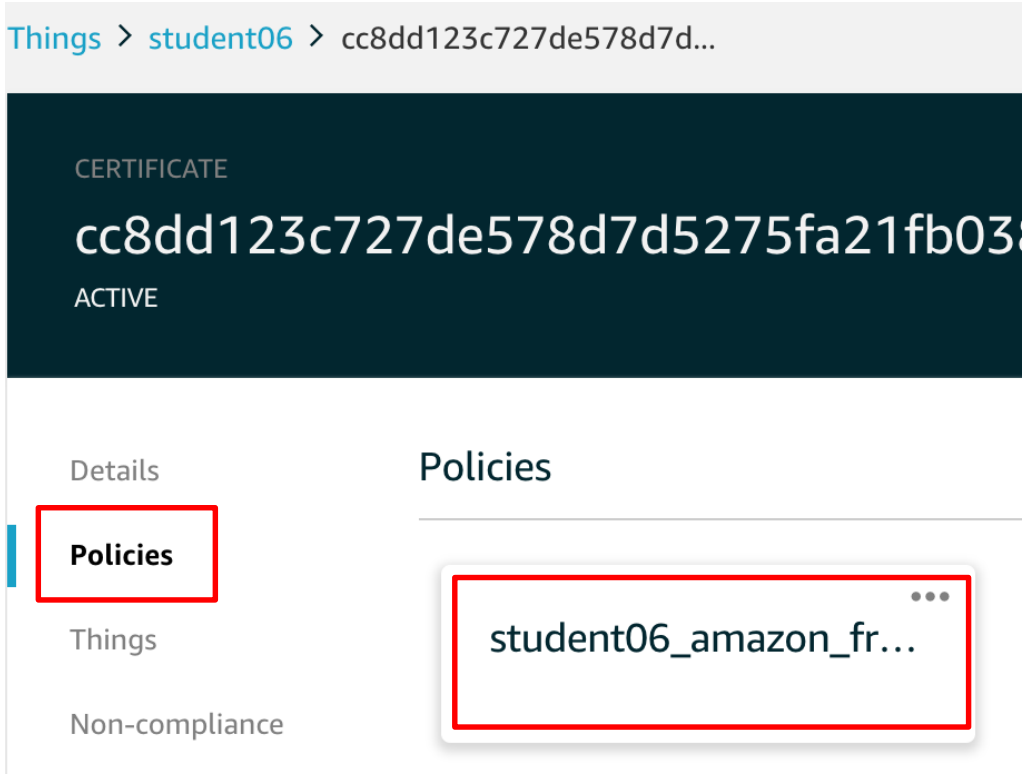
- Examine Thing, Certificate and Policy created in the AWS IoT Core
- Click Manage ->Things and type *studentXX* to search for your thing

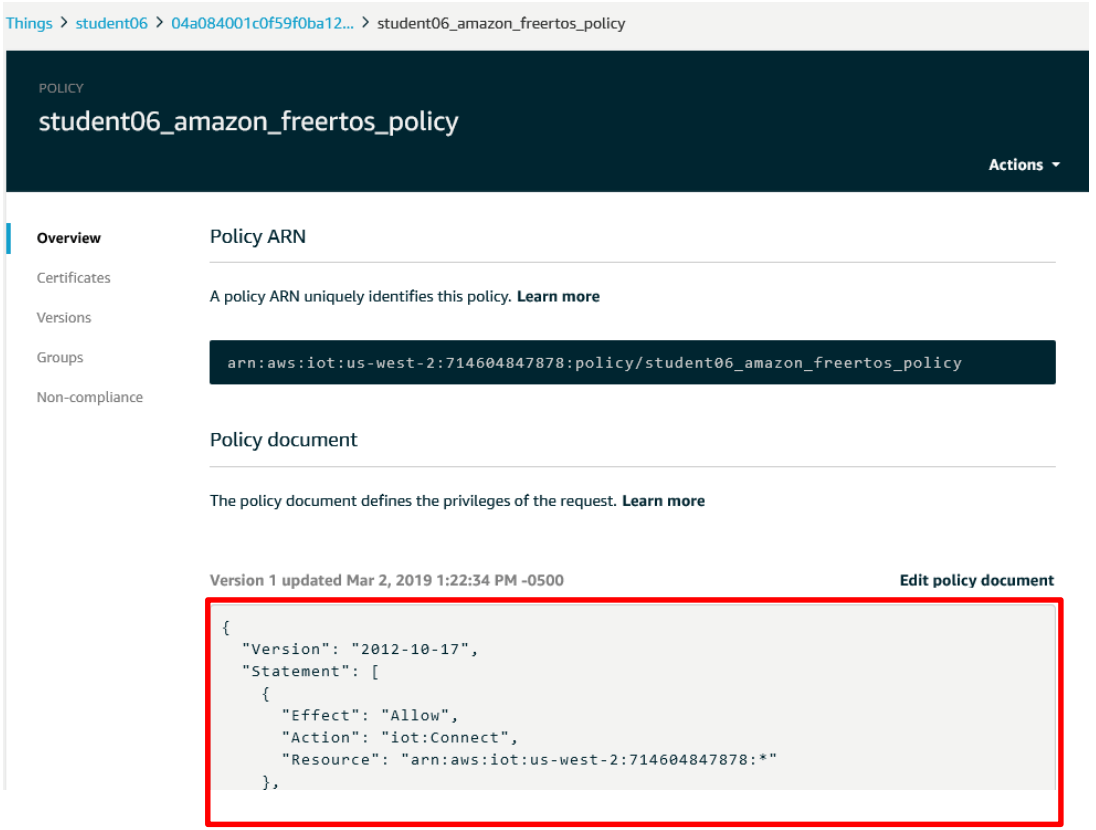




Provision Thing, and device credentials

- Drill down





Provision Thing, and device credentials

- Run *python SetupAWS.py cleanup* in Windows *Command Prompt*
- In AWS Console verify that your Thing, Certificate and Policy are gone
- Run *python SetupAWS.py setup* in Windows *Command Prompt* to provision your device again

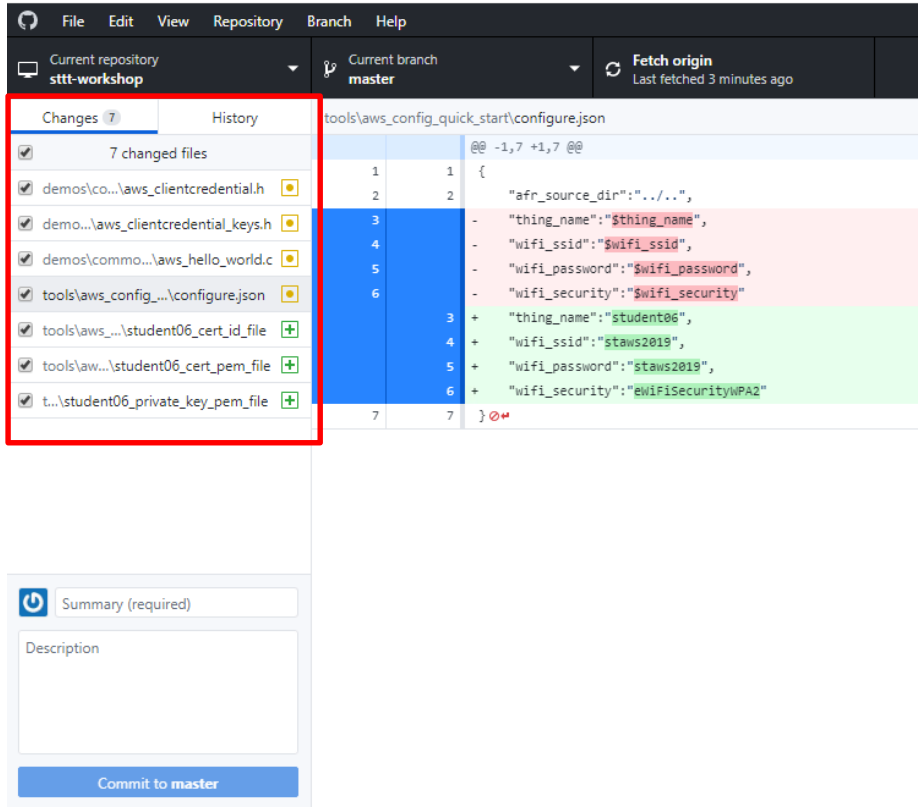
Edit the MQTT topic you will be publishing

- Return to System Workbench
- In the *aws_demos* project, navigate to *application_code/common_demos/source/aws_hello_world.c* file
- On line **90** change *"freertos/demos/echo"* to *"freertos/demos/echo/studentXX"*
- Save the file

life.augmented

Examine all the changes

- Open GitHub Desktop and see all the changes that were made to the project



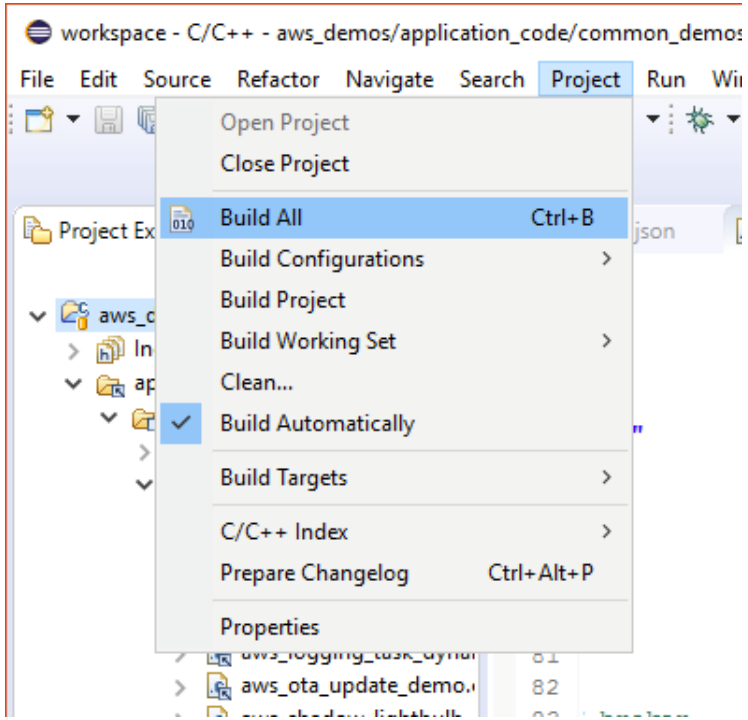
There are 2 files you edited
And 5 files generated/modified by provisioning script

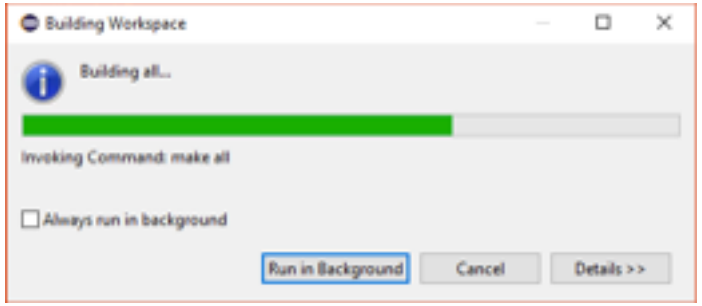
Inspect them and ask the mentors if you do not understand the purpose of each change

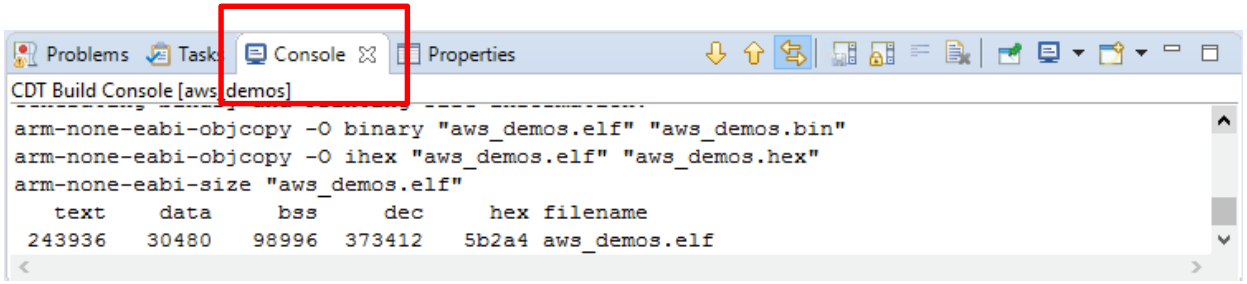
life.augmented

Build the project

- Return to System Workbench. Highlight *aws_demos* project. Press *Ctrl+B*
- Switch to *SW Console* tab and check output files



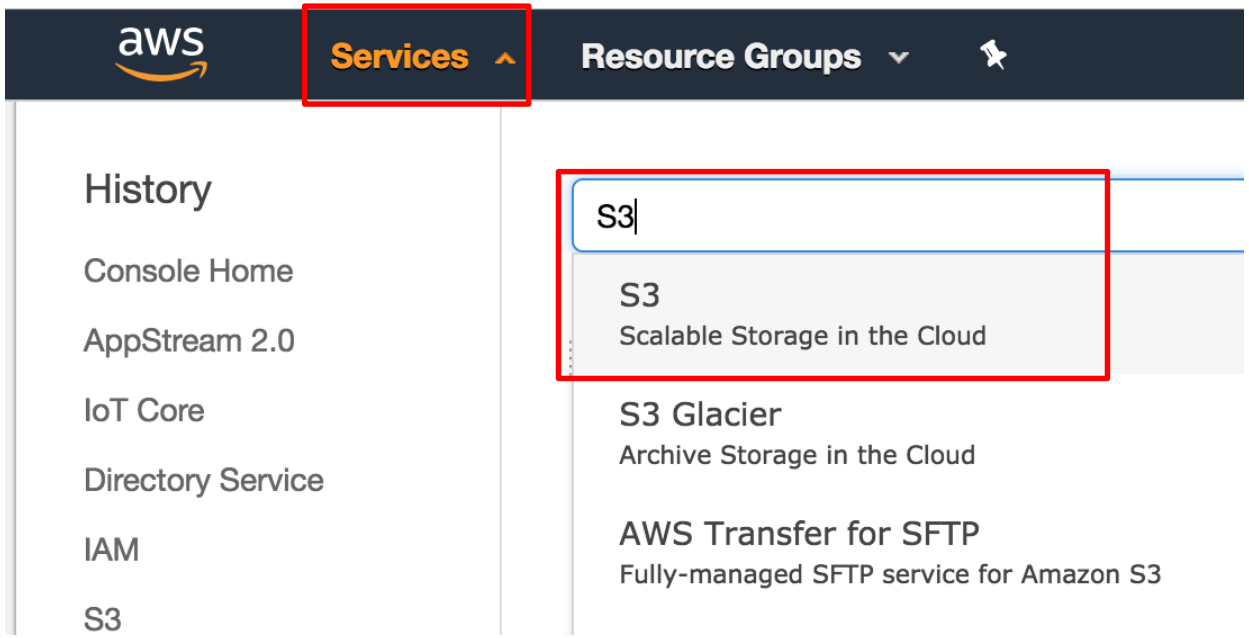




life.augmented

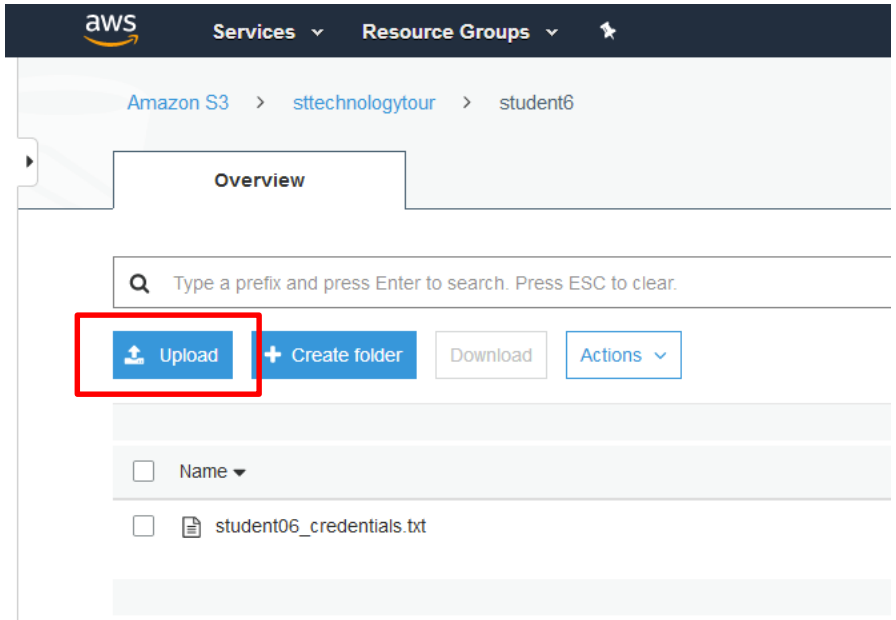
Transfer firmware binary to your laptop

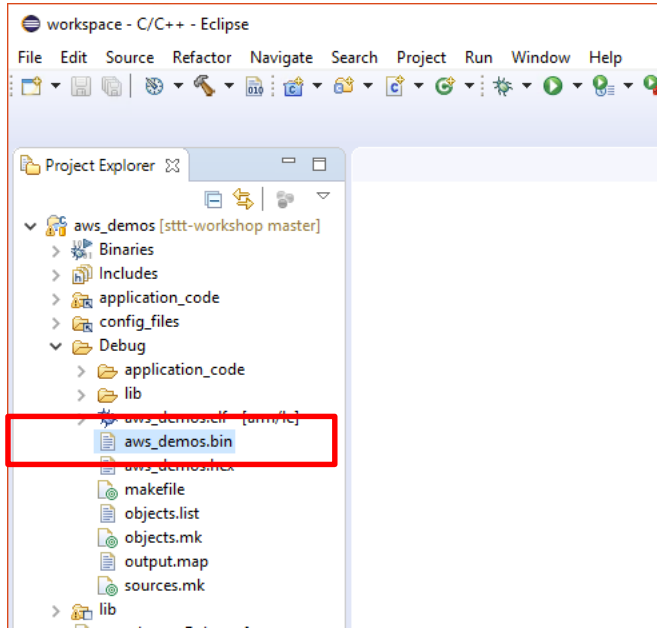
- In the AWS Workspace open the browser and login to AWS Console
- Navigate to S3 service (Services -> type S3 -> Select S3)



Transfer firmware binary to your laptop

- Navigate to *sttechnologytour* bucket -> *studentXX* prefix
- Click on Upload button. Upload aws_demos.bin file by dragging it from SW to S3 upload interface



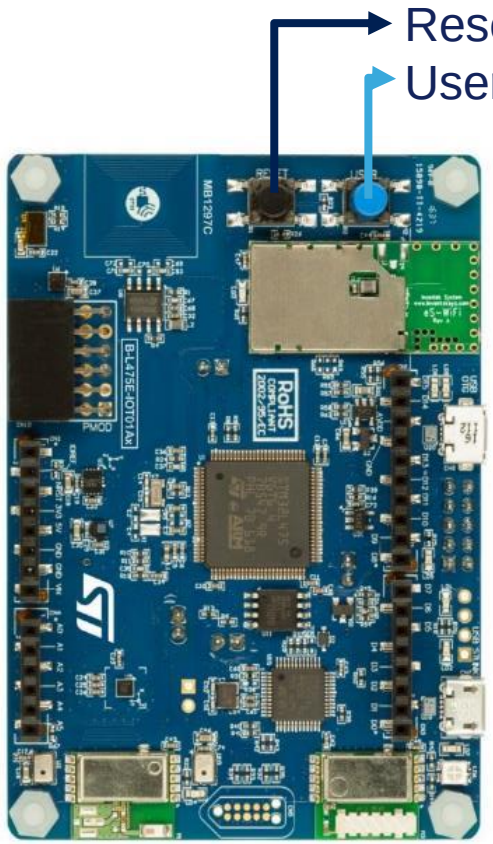


life.augmented

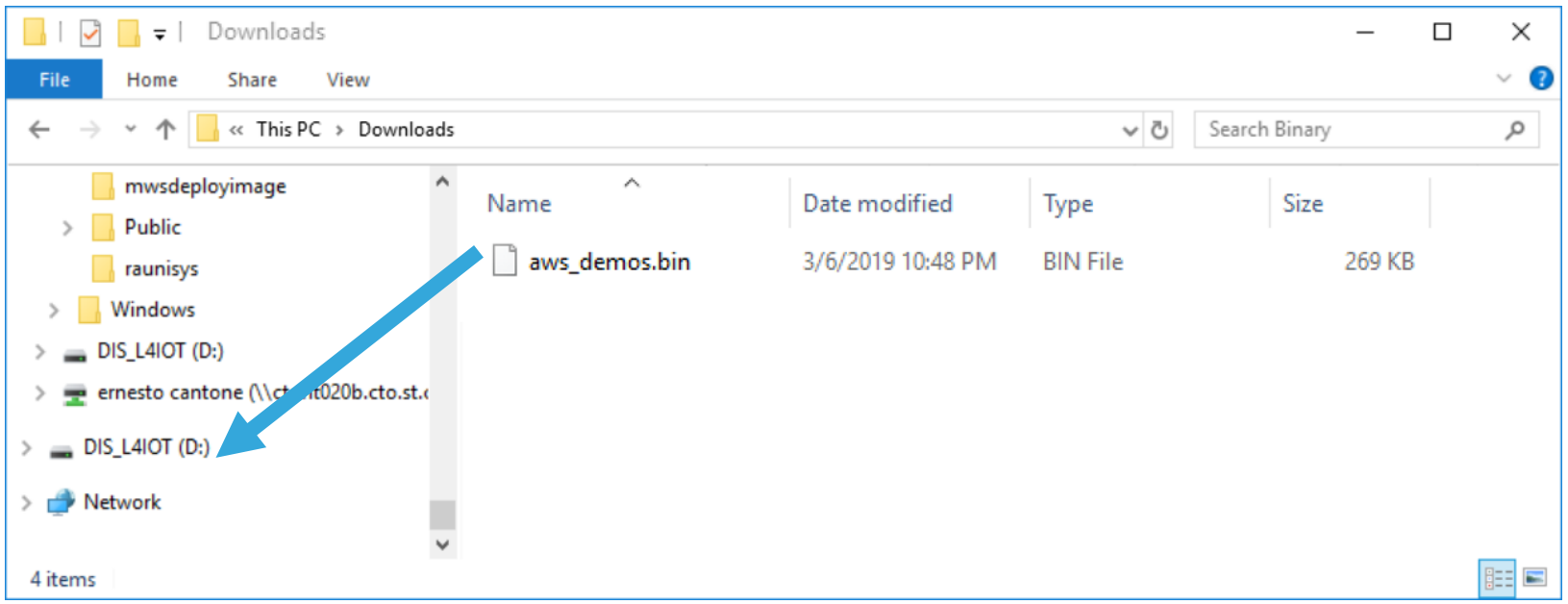
Transfer firmware binary to your laptop

- Login to the AWS Console from your laptop
- Navigate to *S3 service -> sttechnologytour bucket -> studentXX prefix*
- Click on *aws_demos.bin* file and download it

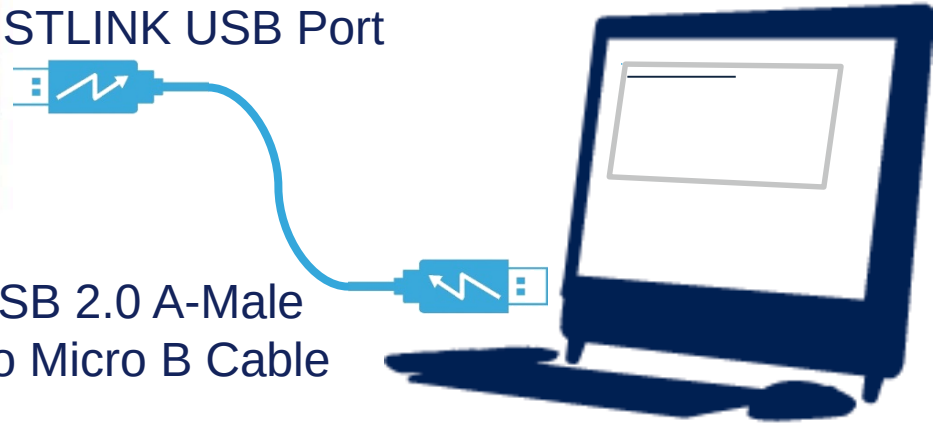
How to Flash the IoT DK



IoT DK



1 Connect to STLINK USB Port



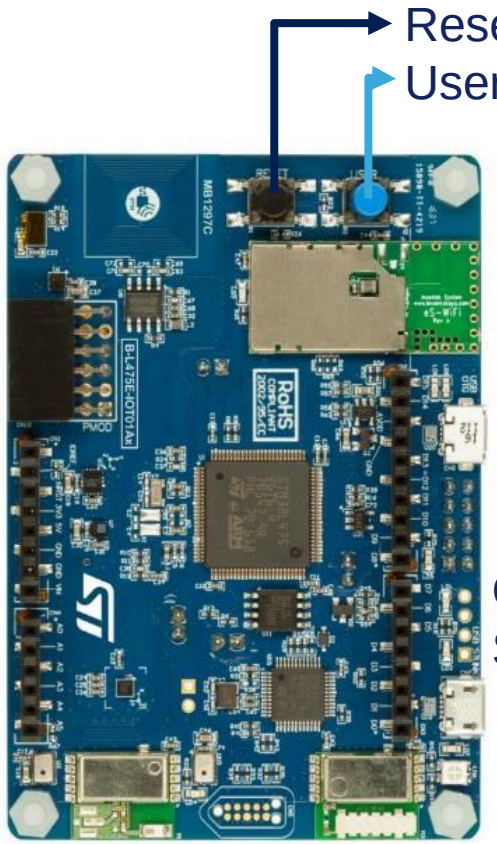
USB 2.0 A-Male to Micro B Cable

2 Drag'n'Drop `aws_demos.bin` into `DIS_L4IOT` (F: in the example)
This action will flash the board with the new FW

3 Press the Reset Button to restart program

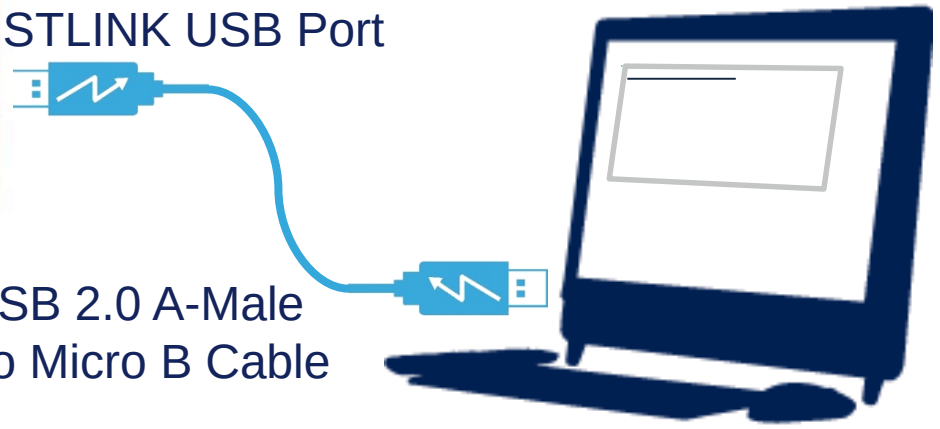
life.augmented

How to Flash the IoT DK (Opt 2)



IoT DK

1 Connect to STLINK USB Port



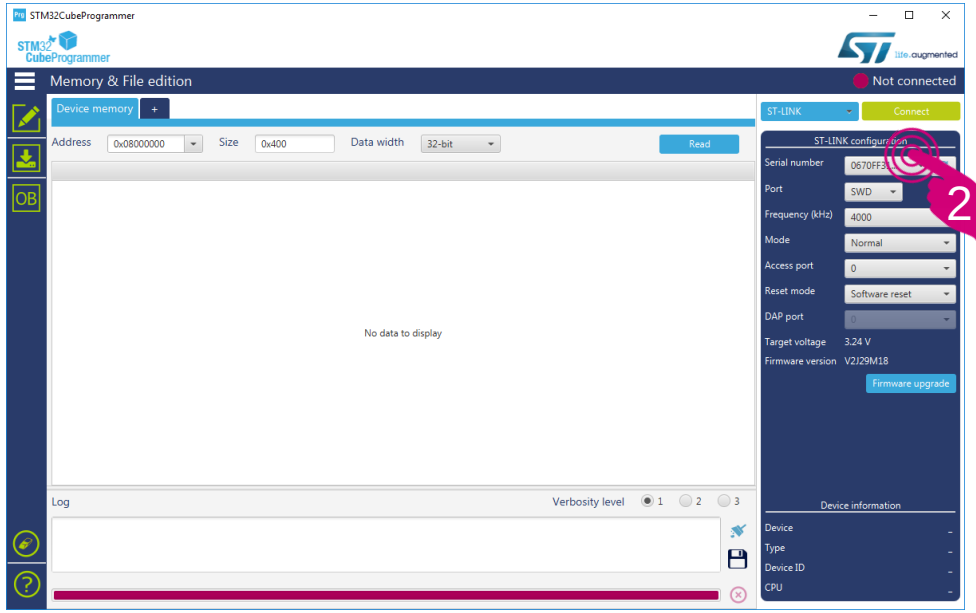
USB 2.0 A-Male to Micro B Cable

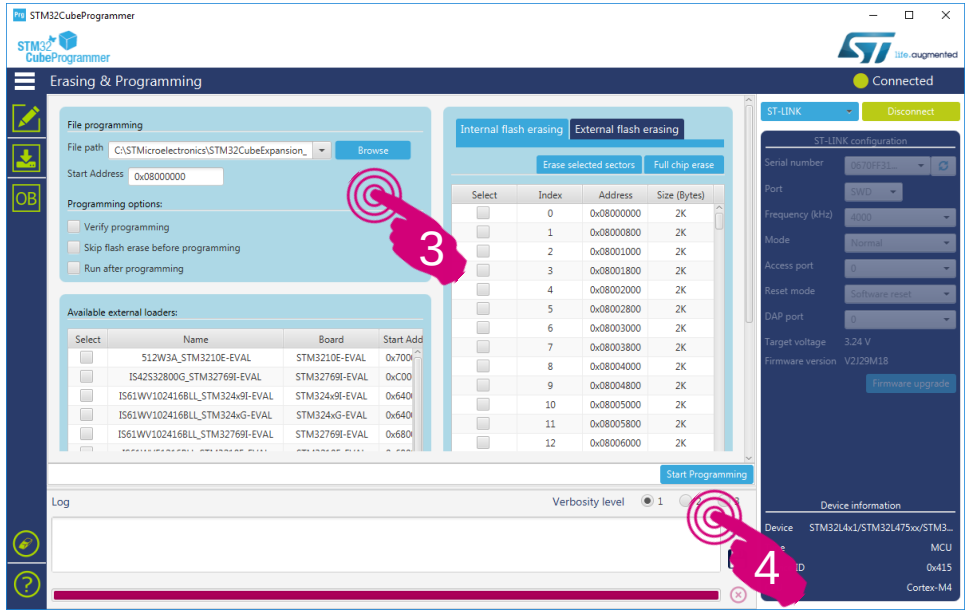
2 Click CONNECT

3 BROWSE to `aws_demos.bin`

4 START PROGRAMMING

5 Press the Reset Button to restart program





life.augmented

Local Machine

MQTT
TLS

IoT DK

STLINK
USB Port

USB 2.0 A-Male
to Micro B Cable

Serial Terminal

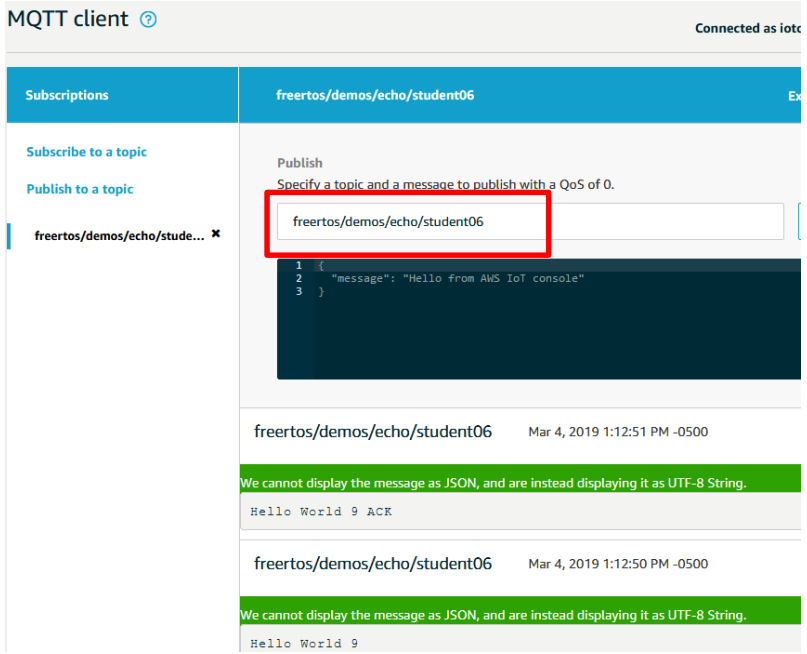
```
~ — -bash ... ~ — -bash ...
210 521 [Tmr Svc] WiFi module initialized.
1 532 [Tmr Svc] Write certificate...
2 553 [Tmr Svc] Write device private key...
3 4487 [Tmr Svc] WiFi connected to AP staws2019.
4 4491 [Tmr Svc] IP Address acquired 192.168.8.203
5 4496 [Tmr Svc] WiFi firmware version is: C3.5.2.5.STM
6 4501 [Tmr Svc] WiFi firmware is up-to-date.
7 4505 [Tmr Svc] Creating MQTT Echo Task...
8 4509 [MQTTEcho] MQTT echo attempting to connect to akavp55t2efxk-ats.iot.us-west-2.amazonaws.com.
9 7141 [MQTTEcho] MQTT echo connected.
10 7144 [MQTTEcho] MQTT echo test echoing task created.
11 8158 [MQTTEcho] MQTT Echo demo subscribed to freertos/demos/echo/student06
12 9178 [MQTTEcho] Echo successfully published 'Hello World 0'
13 10196 [Echoing] Message returned with ACK: 'Hello World 0 ACK'
14 15201 [MQTTEcho] Echo successfully published 'Hello World 1'
15 16220 [Echoing] Message returned with ACK: 'Hello World 1 ACK'
16 21224 [MQTTEcho] Echo successfully published 'Hello World 2'
17 22242 [Echoing] Message returned with ACK: 'Hello World 2 ACK'
18 27246 [MQTTEcho] Echo successfully published 'Hello World 3'
```

life.augmented

Verify that your firmware functions properly

- Open Console and verify your device can connect to WiFi, AWS MQTT broker, and publish/receive messages on *freertos/demos/echo/studentXX* topic
- In AWS Console open IoT Core->Test and subscribe to **freertos/demos/echo/studentXX** topic. Observe MQTT messages

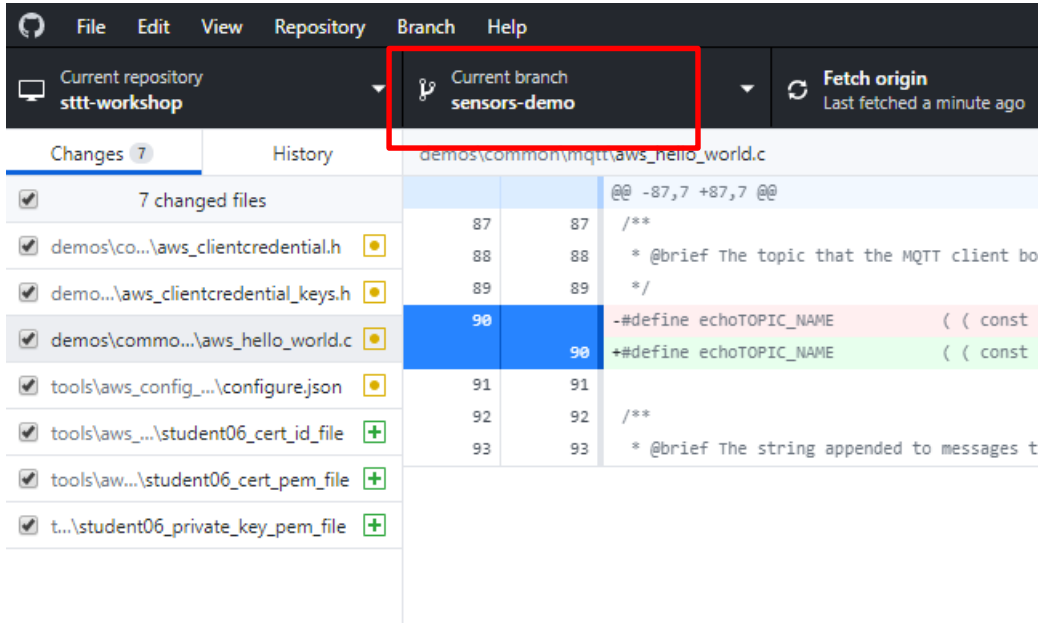
```
~ — -bash ... ~ — -bash ...
210 521 [Tmr Svc] WiFi module initialized.
1 532 [Tmr Svc] Write certificate...
2 553 [Tmr Svc] Write device private key...
3 4487 [Tmr Svc] WiFi connected to AP staws2019.
4 4491 [Tmr Svc] IP Address acquired 192.168.8.203
5 4496 [Tmr Svc] WiFi firmware version is: C3.5.2.5.STM
6 4501 [Tmr Svc] WiFi firmware is up-to-date.
7 4505 [Tmr Svc] Creating MQTT Echo Task...
8 4509 [MQTTEcho] MQTT echo attempting to connect to akavp55t2efxk-ats.iot.us-west-2.amazonaws.com.
9 7141 [MQTTEcho] MQTT echo connected.
10 7144 [MQTTEcho] MQTT echo test echoing task created.
11 8158 [MQTTEcho] MQTT Echo demo subscribed to freertos/demos/echo/student06
12 9178 [MQTTEcho] Echo successfully published 'Hello World 0'
13 10196 [Echoing] Message returned with ACK: 'Hello World 0 ACK'
14 15201 [MQTTEcho] Echo successfully published 'Hello World 1'
15 16220 [Echoing] Message returned with ACK: 'Hello World 1 ACK'
16 21224 [MQTTEcho] Echo successfully published 'Hello World 2'
17 22242 [Echoing] Message returned with ACK: 'Hello World 2 ACK'
18 27246 [MQTTEcho] Echo successfully published 'Hello World 3'
```

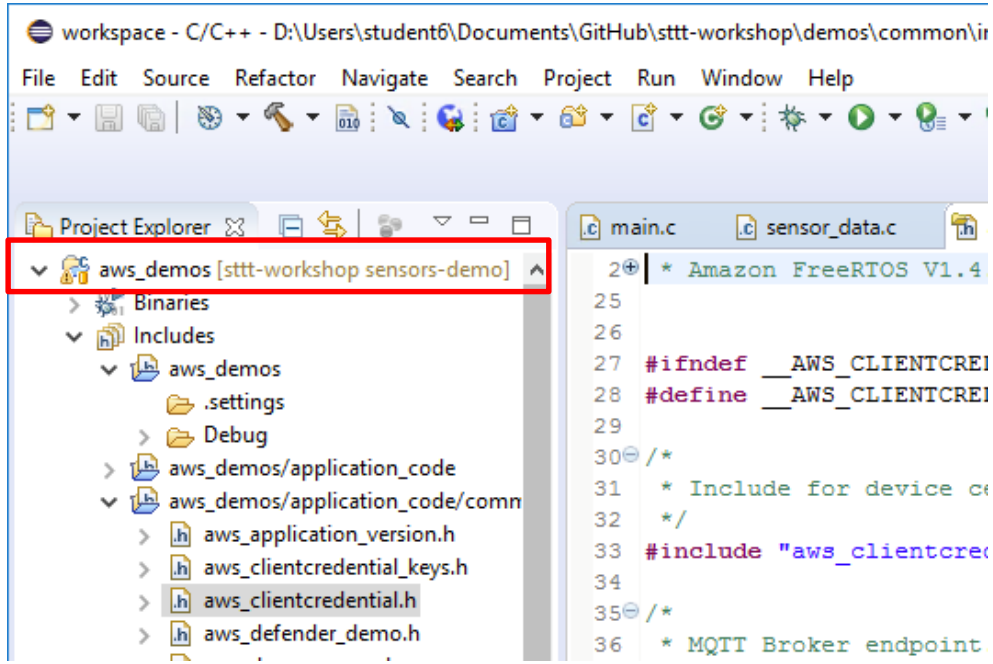


life.augmented

Change Git branch

- Open GitHub desktop and change branch to “*sensors-demo*”
- In the System Workbench **click** on *aws_demos* project and verify that branch is changed from *master* to *sensors-demo*





life.augmented

Compile and run

- Build the project in System Workbench
  (In some cases you might need to clean your previous build first)
- Transfer file using your S3 bucket/folder to the laptop
- Flash firmware to the board

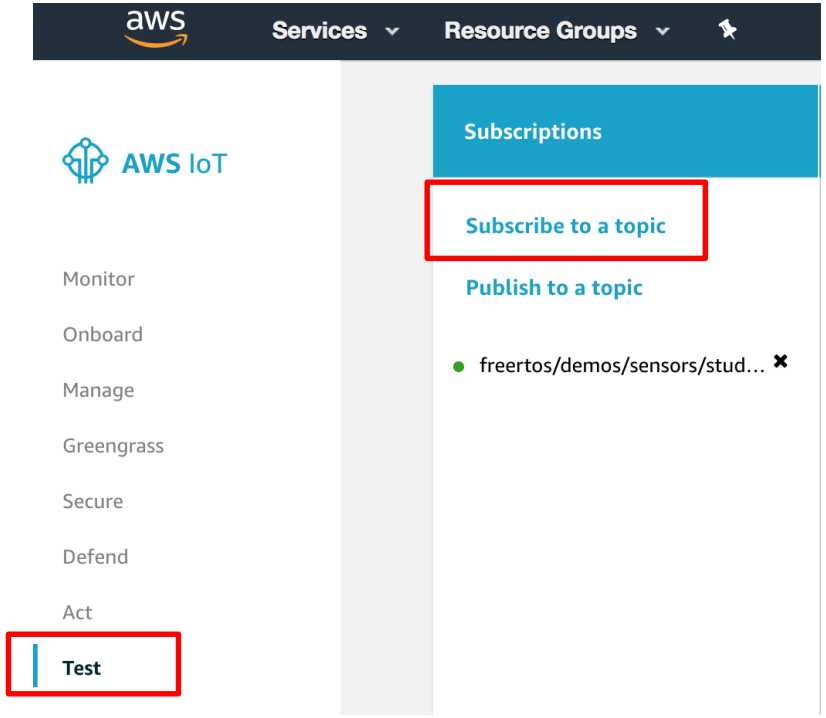
Sensor values in the local console

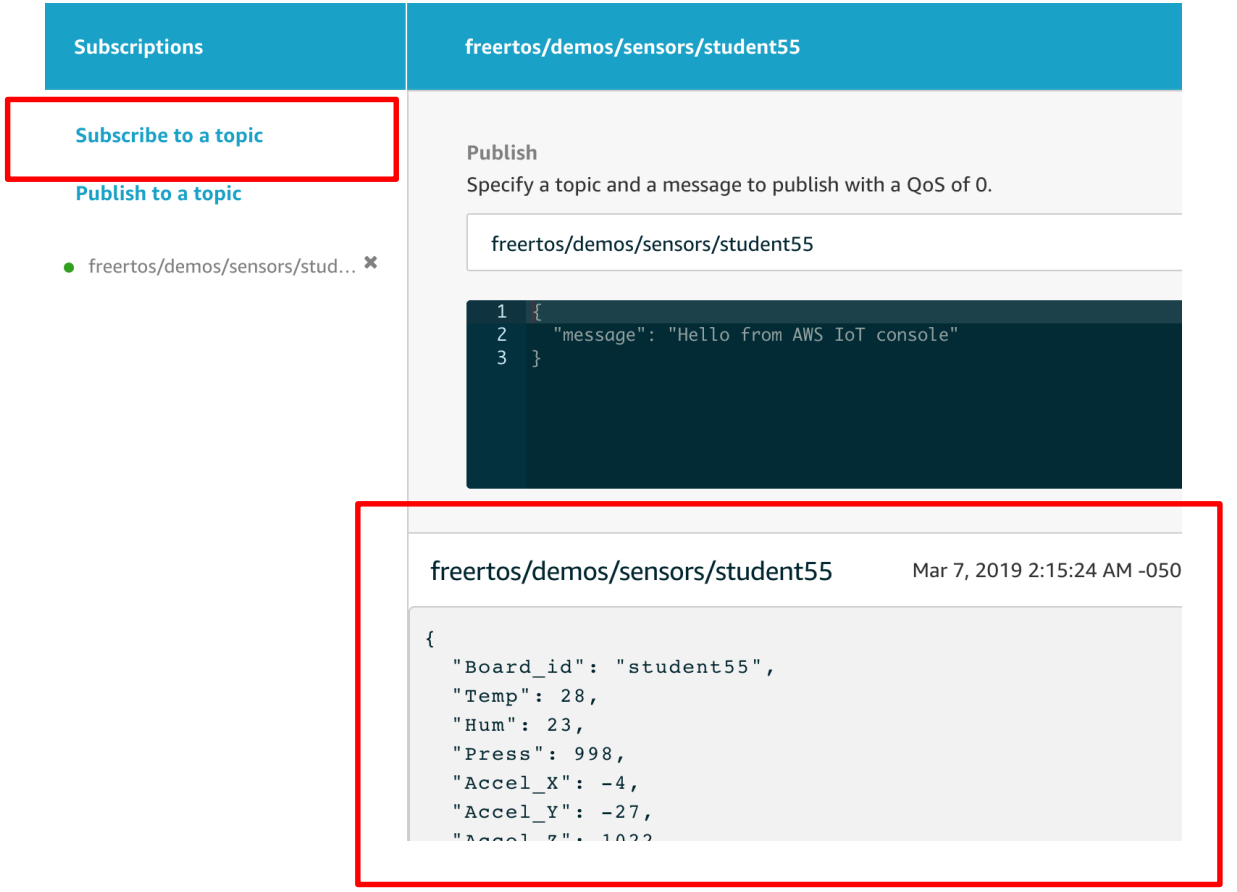
- Open Console and observe sensor values transmitted

```
Accel_Z": 1020, "Gyro_X": -1330, "Gyro_Y": -2380, "Gyro_Z": 700, "Magn_X": -602, "Magn_Y": 138, "Magn_Z": -488}'
171 967300 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -93, "
Accel_Z": 1019, "Gyro_X": -1120, "Gyro_Y": -2310, "Gyro_Z": 700, "Magn_X": -600, "Magn_Y": 131, "Magn_Z": -486}'
172 973333 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -93, "
Accel_Z": 1019, "Gyro_X": -1260, "Gyro_Y": -2310, "Gyro_Z": 700, "Magn_X": -600, "Magn_Y": 133, "Magn_Z": -497}'
173 979367 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -92, "
Accel_Z": 1018, "Gyro_X": -1330, "Gyro_Y": -2380, "Gyro_Z": 700, "Magn_X": -594, "Magn_Y": 137, "Magn_Z": -486}'
174 985401 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -91, "
Accel_Z": 1020, "Gyro_X": -1260, "Gyro_Y": -2310, "Gyro_Z": 630, "Magn_X": -601, "Magn_Y": 133, "Magn_Z": -492}'
175 991435 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -92, "
Accel_Z": 1020, "Gyro_X": -1260, "Gyro_Y": -2310, "Gyro_Z": 630, "Magn_X": -598, "Magn_Y": 137, "Magn_Z": -491}'
176 997468 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -90, "
Accel_Z": 1020, "Gyro_X": -1260, "Gyro_Y": -2310, "Gyro_Z": 700, "Magn_X": -594, "Magn_Y": 132, "Magn_Z": -493}'
177 1003502 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -5, "Accel_Y": -23,
"Accel_Z": 1021, "Gyro_X": -1400, "Gyro_Y": -2380, "Gyro_Z": 700, "Magn_X": -584, "Magn_Y": 121, "Magn_Z": -487}'
178 1009536 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -8, "Accel_Y": -21,
"Accel_Z": 1022, "Gyro_X": -1330, "Gyro_Y": -2310, "Gyro_Z": 700, "Magn_X": -586, "Magn_Y": 126, "Magn_Z": -498}'
179 1015569 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -8, "Accel_Y": -21,
"Accel_Z": 1022, "Gyro_X": -1330, "Gyro_Y": -2310, "Gyro_Z": 700, "Magn_X": -590, "Magn_Y": 123, "Magn_Z": -494}'
180 1021603 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -8, "Accel_Y": -21,
"Accel_Z": 1022, "Gyro_X": -1260, "Gyro_Y": -2310, "Gyro_Z": 700, "Magn_X": -586, "Magn_Y": 119, "Magn_Z": -502}'
181 1027637 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -7, "Accel_Y": -23,
"Accel_Z": 1022, "Gyro_X": -1330, "Gyro_Y": -2450, "Gyro_Z": 630, "Magn_X": -585, "Magn_Y": 128, "Magn_Z": -494}'
182 1033672 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -25,
"Accel_Z": 1021, "Gyro_X": -1330, "Gyro_Y": -2380, "Gyro_Z": 630, "Magn_X": -583, "Magn_Y": 118, "Magn_Z": -491}'
183 1039705 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -25,
"Accel_Z": 1021, "Gyro_X": -1330, "Gyro_Y": -2380, "Gyro_Z": 700, "Magn_X": -586, "Magn_Y": 119, "Magn_Z": -495}'
184 1045739 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -24,
"Accel_Z": 1021, "Gyro_X": -1260, "Gyro_Y": -2310, "Gyro_Z": 630, "Magn_X": -589, "Magn_Y": 121, "Magn_Z": -489}'
185 1051773 [SensorPub] MQTT successfully published '{"Board_id":"student55","Temp": 28, "Hum": 23, "Press": 998, "Accel_X": -6, "Accel_Y": -25,
"Accel_Z": 1022, "Gyro_X": -1260, "Gyro_Y": -2380, "Gyro_Z": 630, "Magn_X": -585, "Magn_Y": 118, "Magn_Z": -485}'
```

Sensor values received by MQTT broker

- Check MQTT messages in AWS Console -> AWS IoT Core
  The topic is *freertos/demos/**sensors**/studentXX*

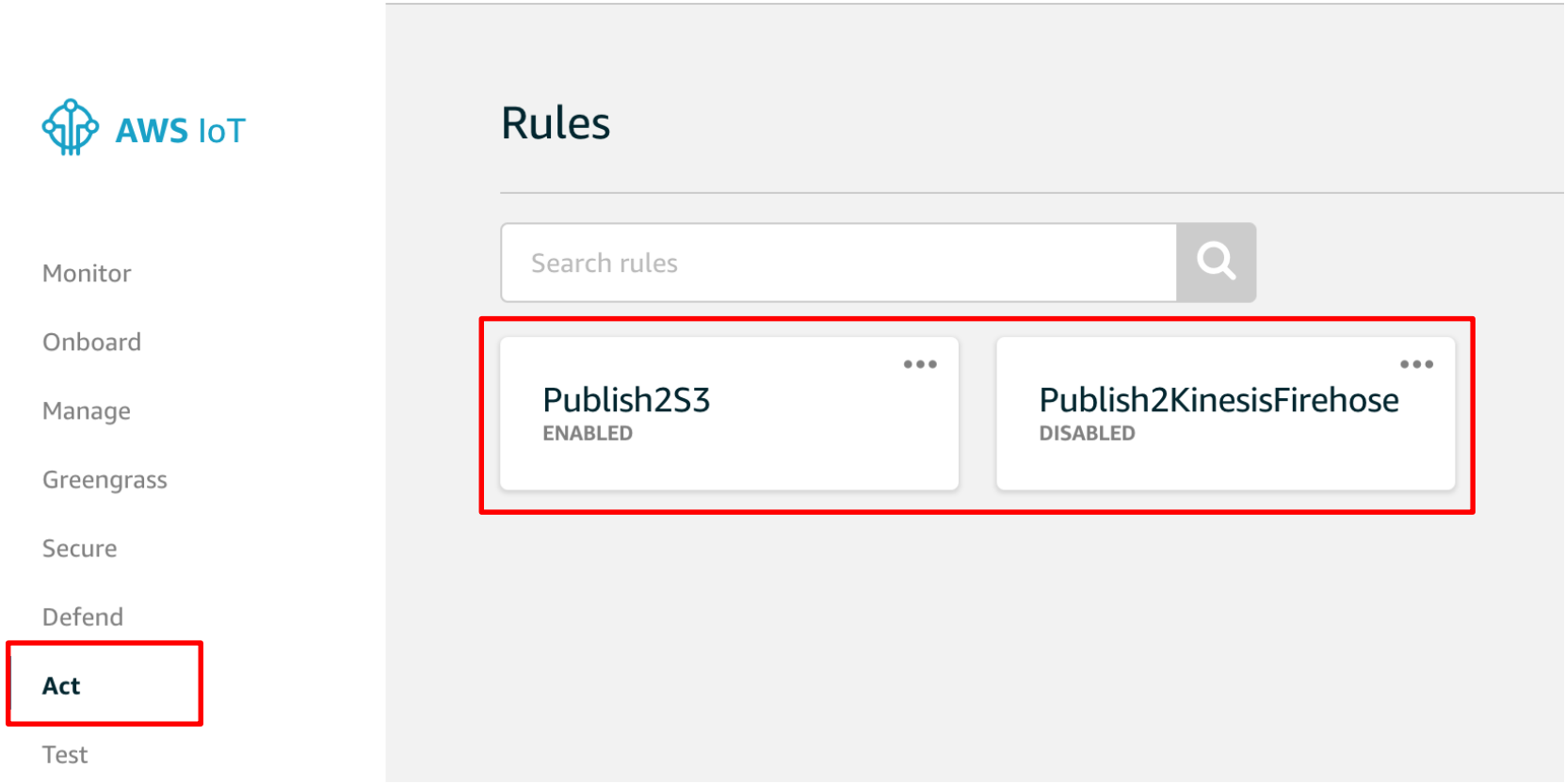




life.augmented

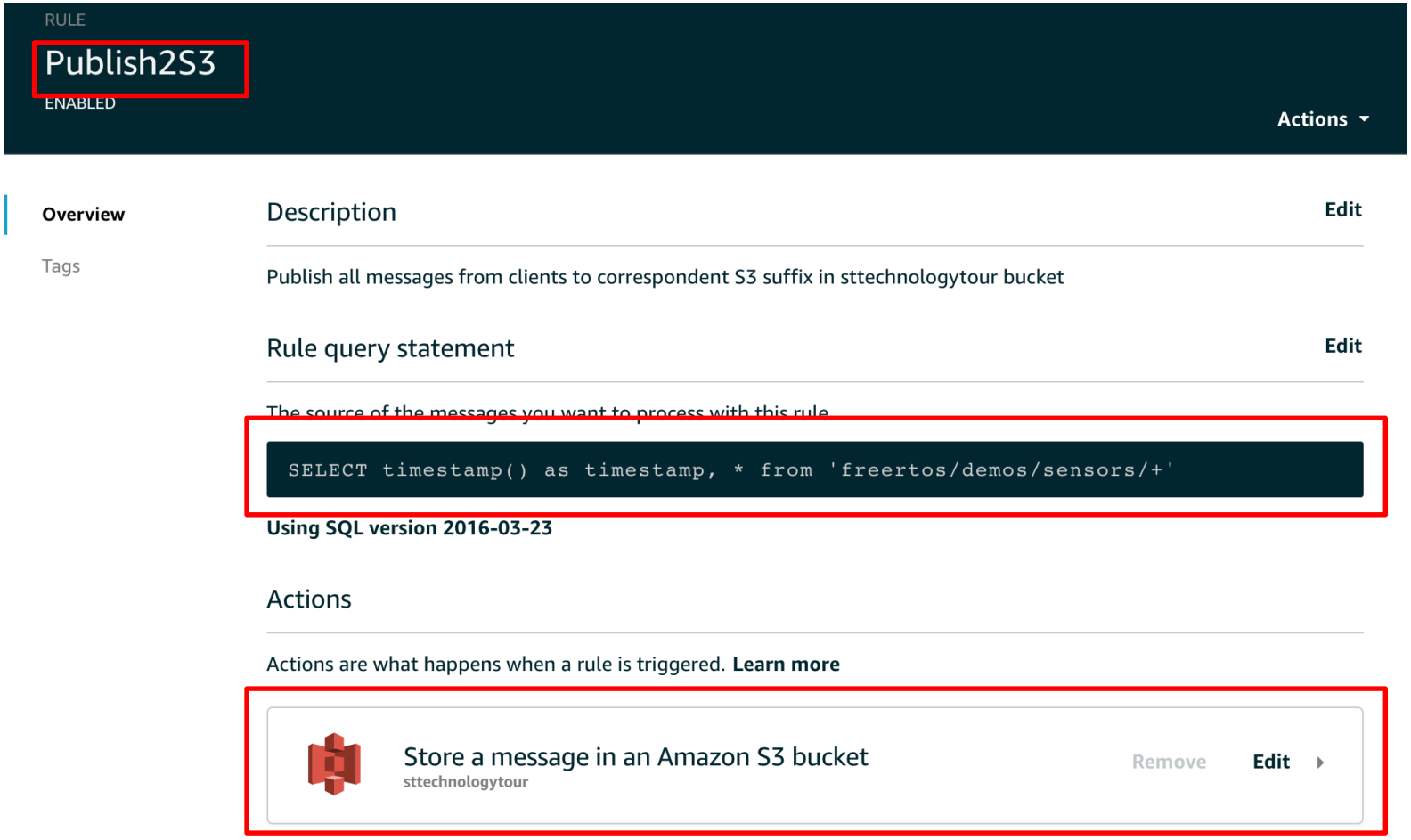
AWS IoT Topic Rules

- In the AWS Console go to Services -> IoT Core
- Click on Act tab and inspect 2 rules currently provisioned



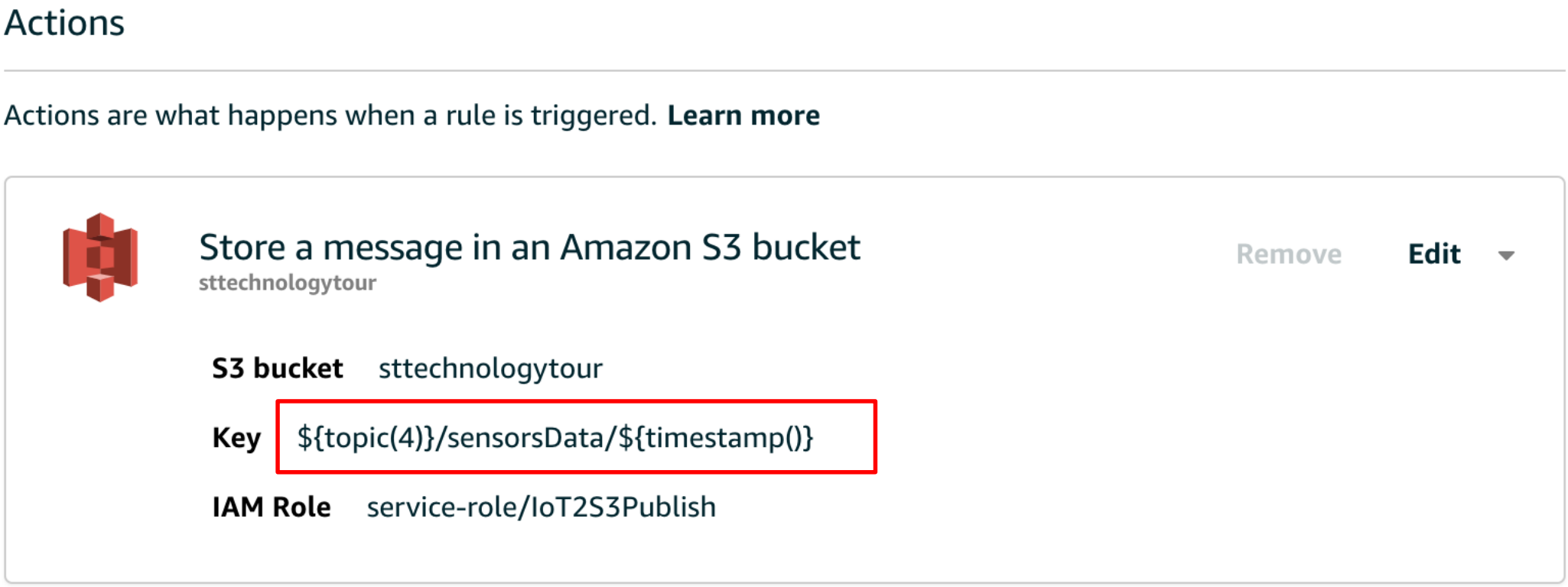
Store every MQTT message as a file

- Publish to S3 bucket rule
  (check the SQL syntax operation on JSON)



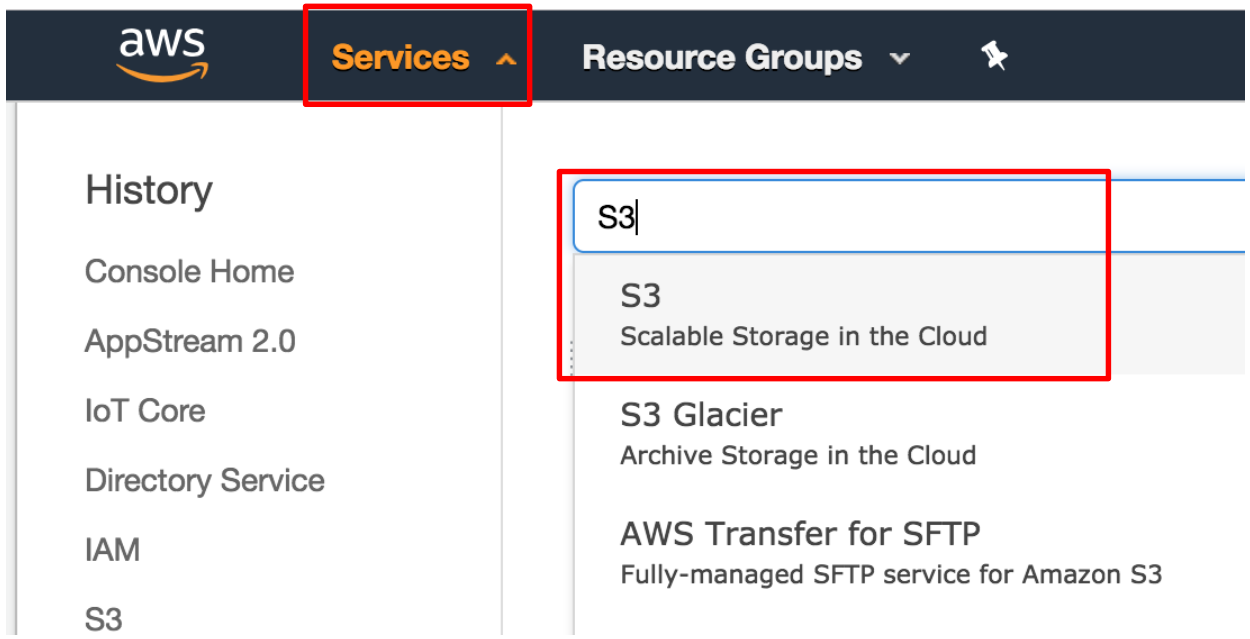
Store every MQTT Message as a file

- Publish to S3 bucket rule action
  (notice the usage of functions to create dynamic S3 key)
  https://docs.aws.amazon.com/iot/latest/developerguide/iot-sql-functions.html



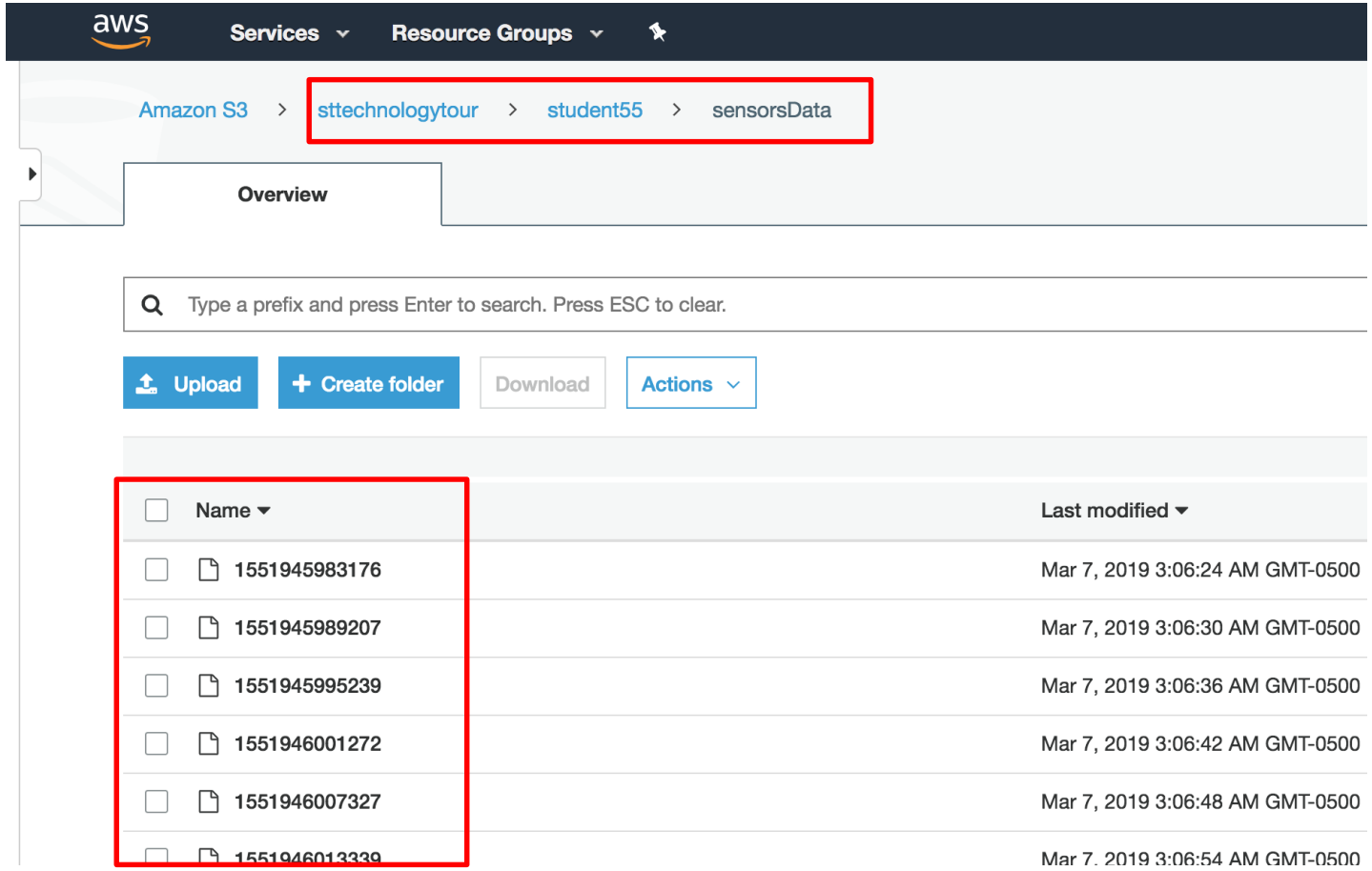
Check the files published to S3

- In the AWS Workspace open the browser and login to AWS Console
- Navigate to S3 service (Services -> type S3 -> Select S3)



Check the files published to S3

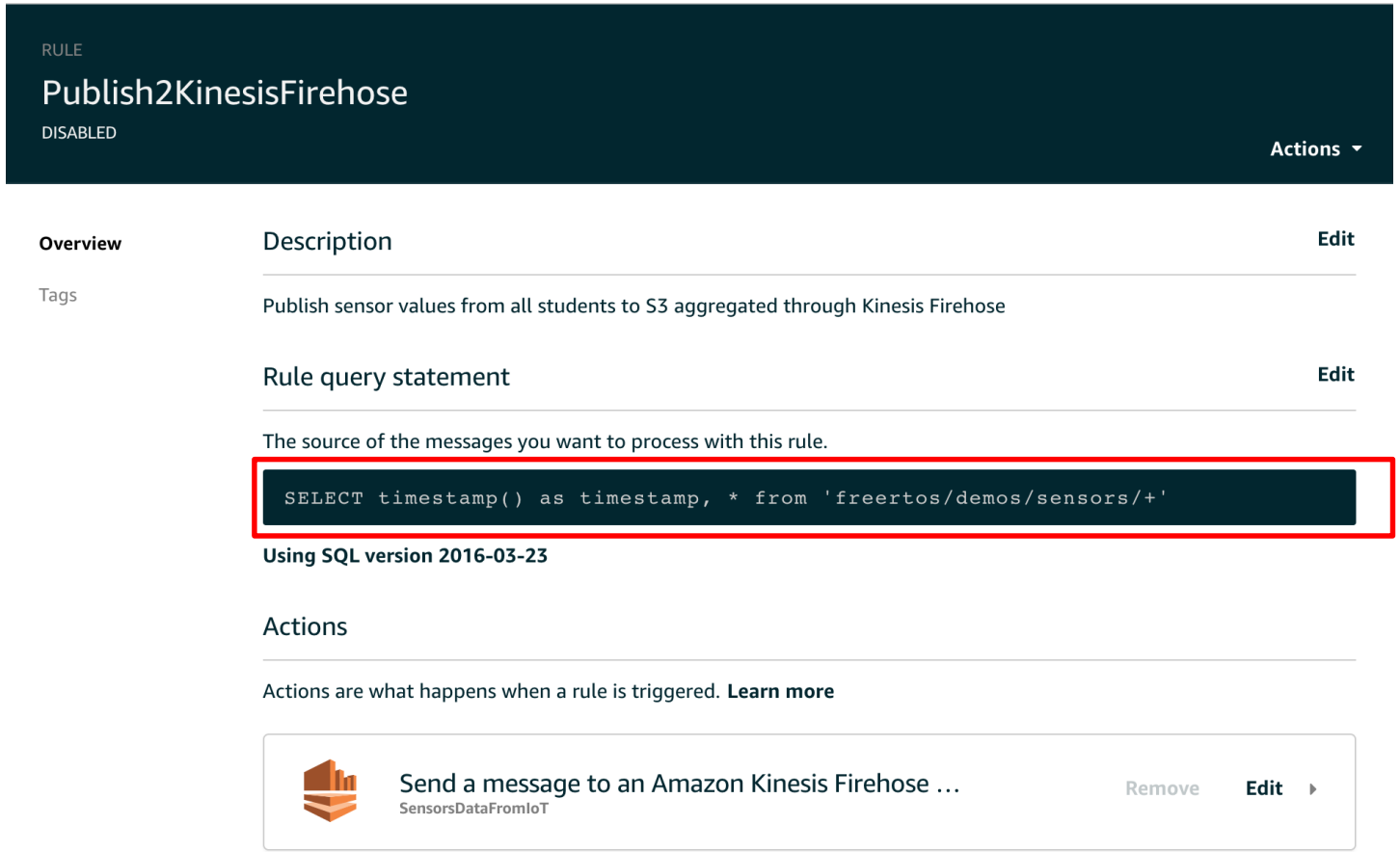
- Navigate to *sttechnologytour* bucket -> *studentXX/sensorsData* prefix
- Inspect the files created and open one of the files



life.augmented

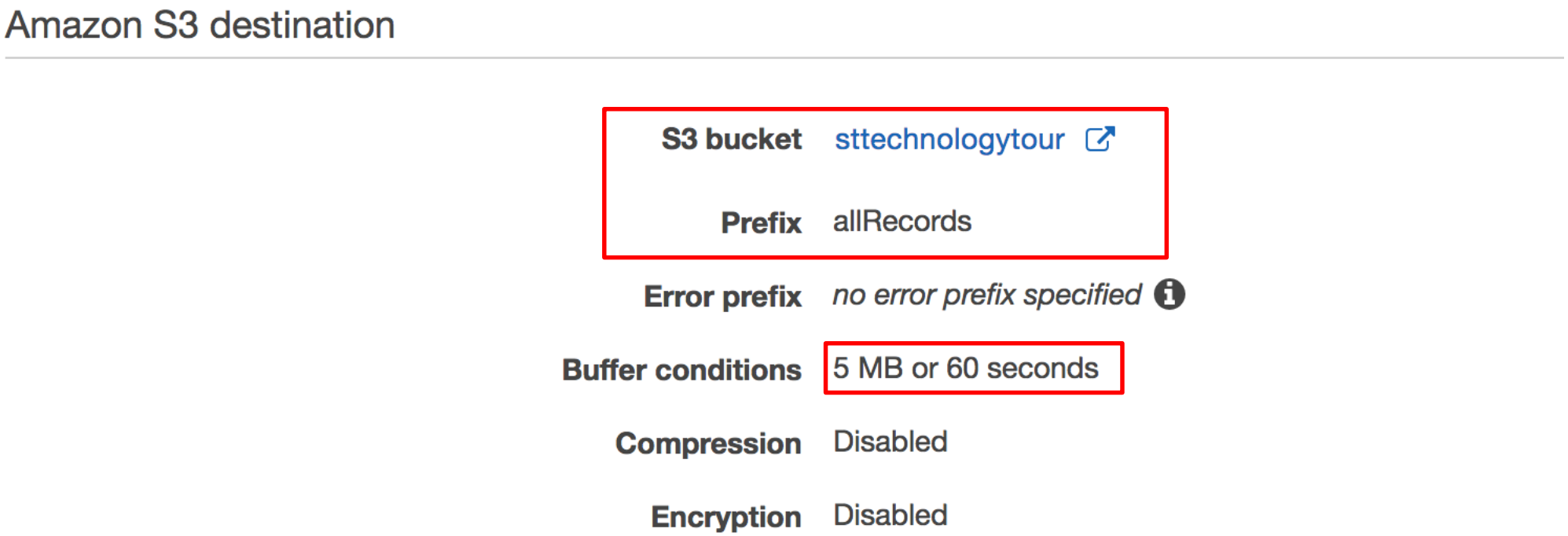
Aggregate messages

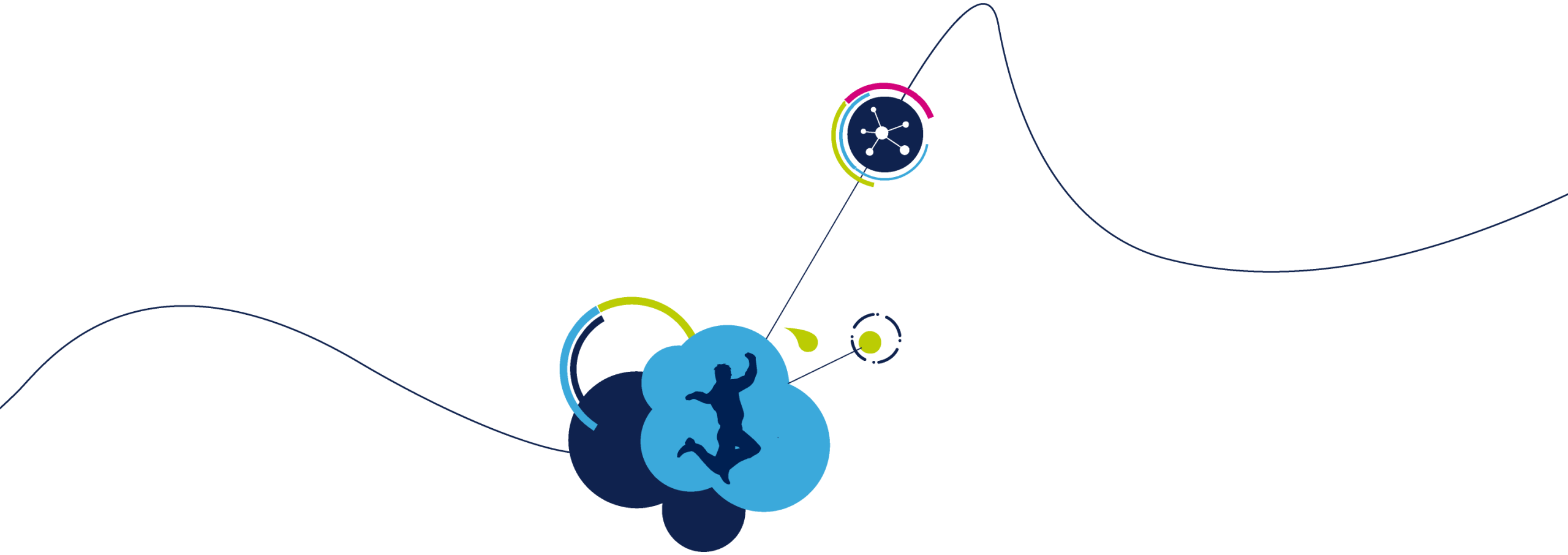
- Check Publish2KinesisFirehose rule
- It aggregates messages from all boards, buffer them for 60 seconds in Kinesis stream and then publish to S3



Aggregate messages

- Check Publish2KinesisFirehose rule
- It aggregates messages from all boards, buffer them for 60 seconds in Kinesis stream and then publish to S3





Appendix: Software and Other Pre-requisites

life.augmented

System Workbench for STM32
(requires registration to openstm32.org)

http://www.openstm32.org/Downloading+the+System+Workbench+for+STM32+installer

- install_sw4stm32_win_64bits-v2.5.exe
- install_sw4stm32_macos_64bits-v2.5.run
  - Warning: To run System Workbench for STM32 on MAC OSX systems, XCode may be required. To download it, please refer to the Apple developer website (registration as Apple Developer is required)
  - The downloaded installer is an executable binary file. Your web browser might have removed the execution right of the file. Please set the execution right to the installer file (chmod 755 install_sw4stm32.run then ./install_sw4stm32.run) OR Launch it with /bin/bash (/bin/bash install_sw4stm32.run)
  - If an error message saying the installer “is damaged and can’t be opened. You should move it to the Trash.”, please modify the installation access right in the Gatekeeper. On latest version of MAC OSX, go in the terminal:

```
#To disable
sudo spctl --master-disable
#To set the Gatekeeper access right back
sudo spctl --master-enable
```

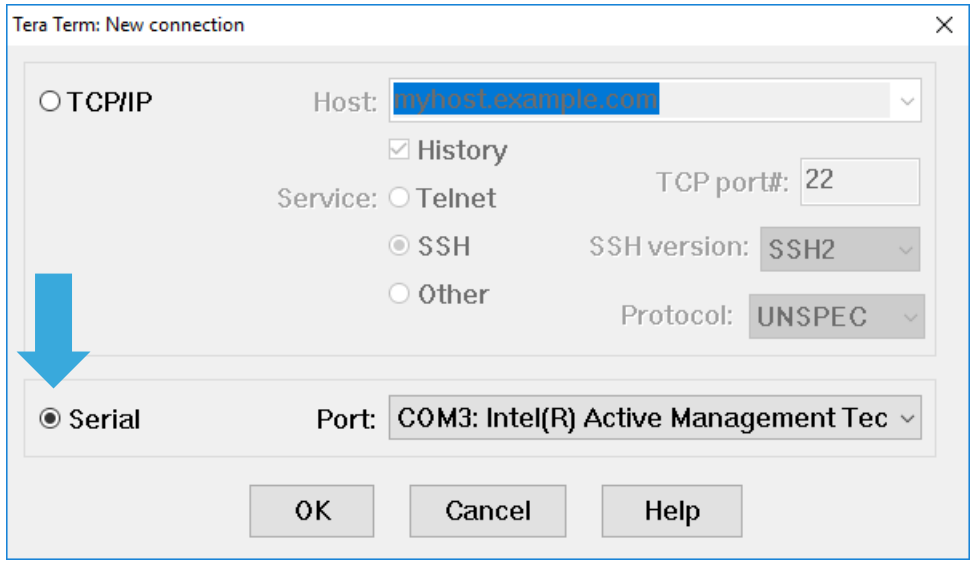
  - On older version of MAC OSX:
  - Go in the “System Preferences” > “Security & Privacy”, then select “Allow downloaded app from :” “Anywhere”. When the installation is done, restore the setting value back at “Mac App Store and identified developers”
- install_sw4stm32_linux_64bits-latest.run
  - The installer in GUI-mode requires gksudo. On Ubuntu, gksudo is in package gksu, that you can install by sudo apt-get install gksu. If gksudo is not installed, the installer can still be launched in command-line.
  - The downloaded installer is an executable binary file. Your web browser might have removed the execution right of the file. Please set the execution right to the installer file (chmod 755 install_sw4stm32.run then ./install_sw4stm32.run) OR launch it with /bin/bash (/bin/bash install_sw4stm32.run)

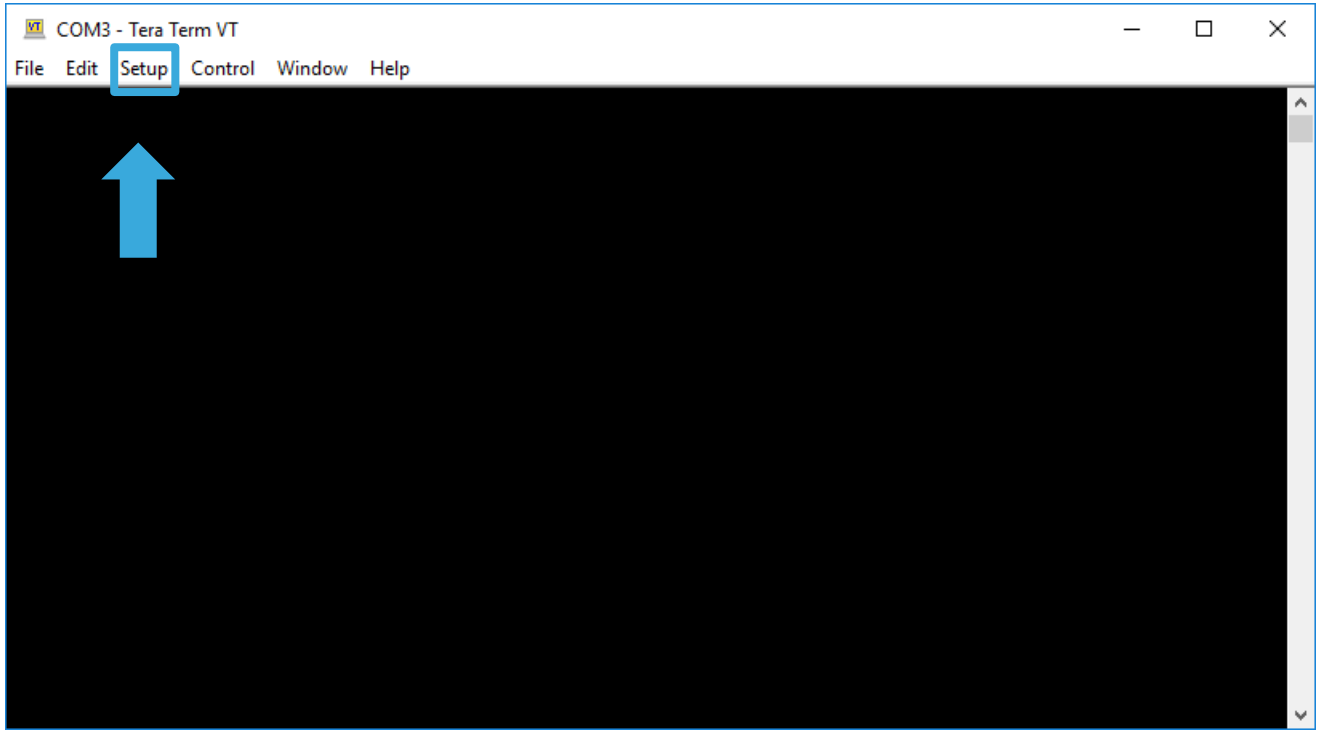
Software and Others Pre-requisites

Windows (Win7, Win8, Win10)

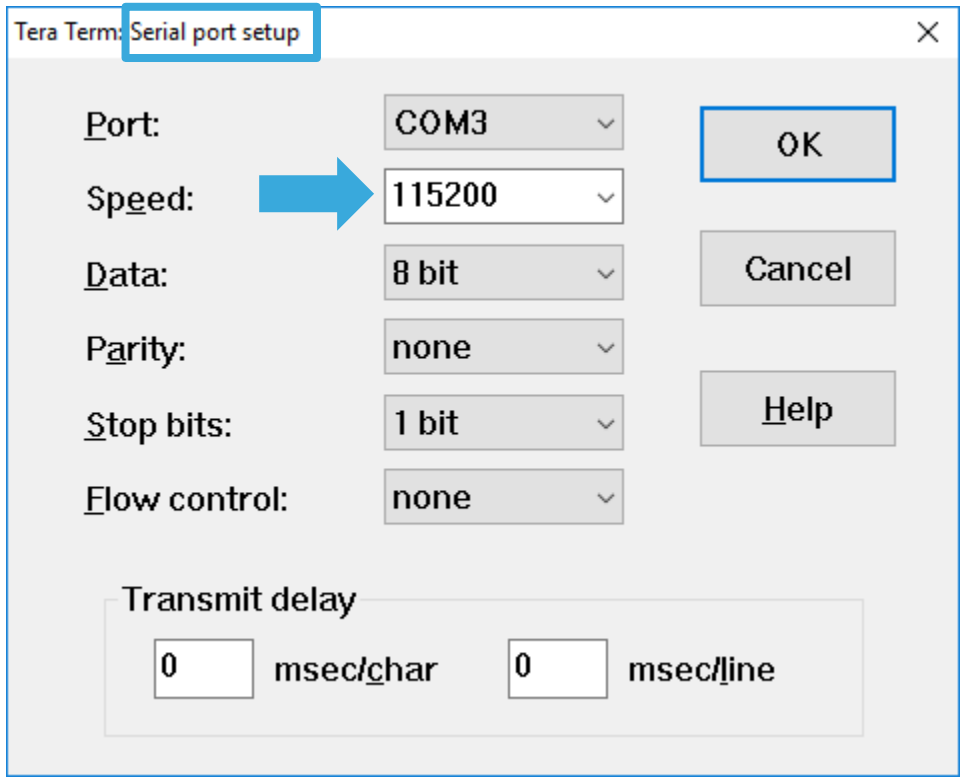
- **STM32 ST-Link Utility:** STSW-LINK004 (requires registration to my.st.com)
  - NOTE: Required for Window 7
- **ST-LINK, ST-LINK/V2, ST-LINK/V2-1 USB driver signed for Windows7, 8, 10:** STSW-LINK009 (requires registration to my.st.com)
  - NOTE: Required for Window 7
- **Serial line monitor**: Tera Term (https://ttssh2.osdn.jp/)
  - Install from USB `.\Windows\teraterm-4.99.exe`

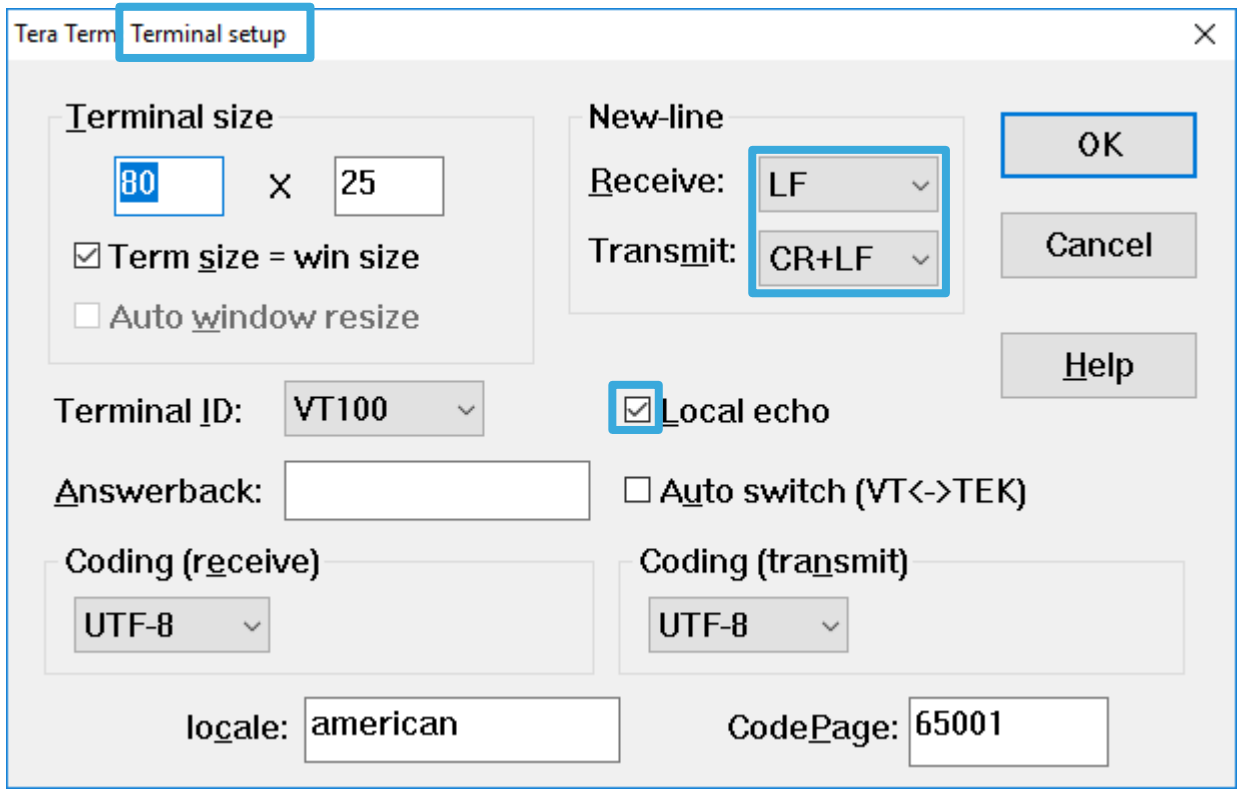
Tera Term Setup





Tera Term Setup





life.augmented

Software and Others Pre-requisites

Mac OS: PicoCom

Serial line monitor: PicoCom or Screen

- Launch Spotlight by pressing Cmd + Space. Type terminal and select the Terminal app.
- In the Terminal window, enter the commands
  - `$ brew install picocom`
  - `$ ls –l /dev/tty*usbmodem*`
    - Example: /dev/tty.usbmodem413
  - `$ picocom --imap lfcrlf -b 115200 -p 1 -d 8 –c <usb device file>`
    - Example: `$ picocom --imap lfcrlf -b 115200 -p 1 -d 8 –c /dev/tty.usbmodem413`

Software and Others Pre-requisites

Mac OS: Screen (native app)

Serial line monitor: PicoCom or Screen

- Launch Spotlight by pressing Cmd + Space. Type terminal and select the Terminal app.
- In the Terminal window, enter the command: `ls /dev/cu.usb*`
- In the list of devices, look for a device that contains `cu.usbserial` or `cu.usbmodem`; in the example below IoT DK is mapped to `/dev/cu.usbmodem1413`
- Launch the from the terminal the `screen` utility by entering the command:

```
cesmosrv03:hack marco$ ls /dev/cu.usb*
/dev/cu.usbmodem1413
cesmosrv03:hack marco$ screen -L /dev/cu.usbmodem1413 115200 -L
```

usb device name

Serial terminal baudrate

- The `screen` command will open a serial terminal connected to the device. Reset the board to see log messages from the device

Software and Others Pre-requisites

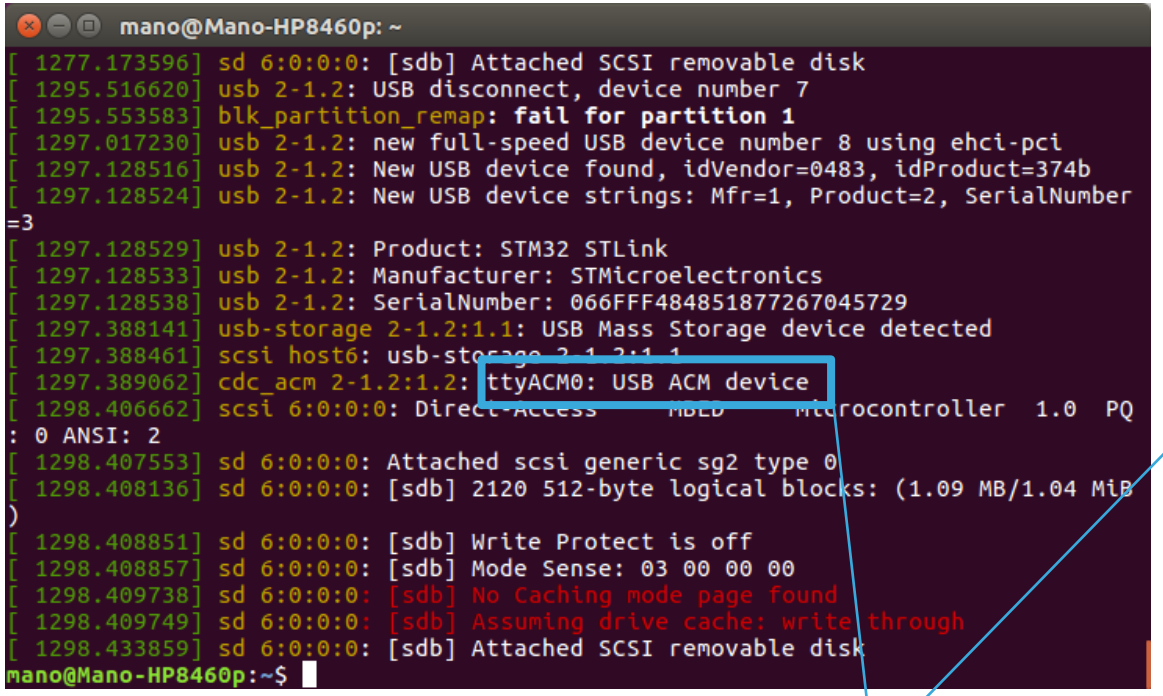
Linux (Ubuntu)

Serial line monitor: Putty or Picocom

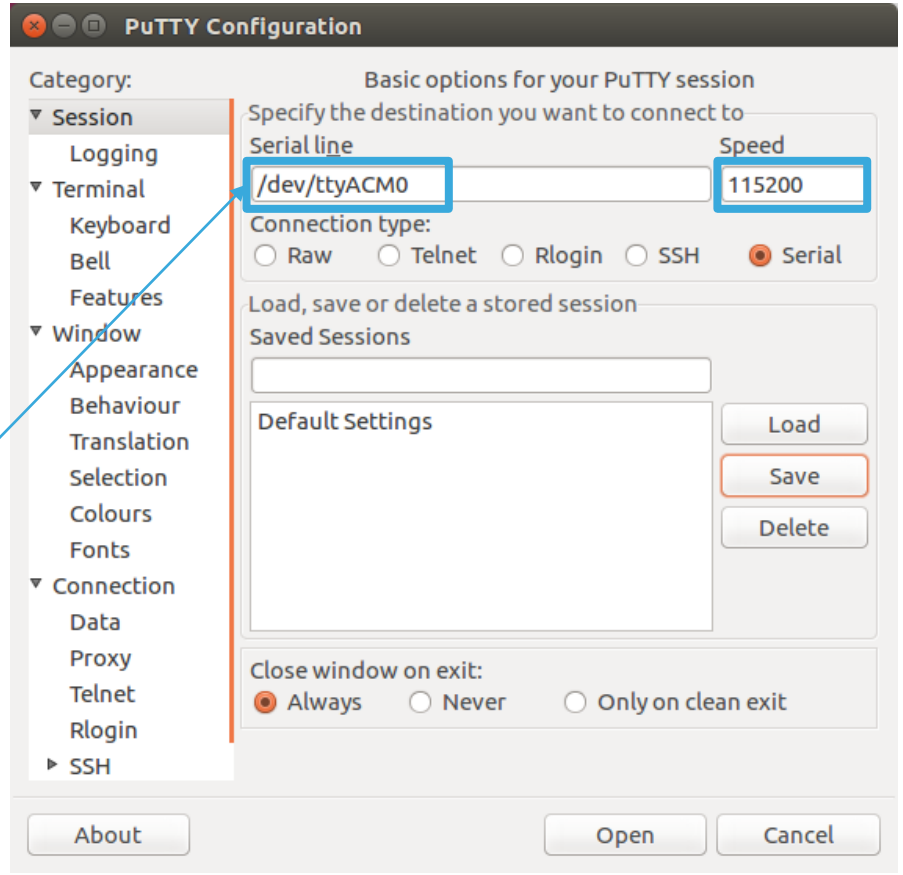
- Putty
  - $ sudo apt-get update (This command updates the Ubuntu package list with latest one)
  - $ sudo apt-get install -y putty
- Picocom
  - $ sudo apt-get install picocom
  - $ ls –l /dev/tty*usbmodem*
    - Example: /dev/tty.usbmodem413
  - $ picocom --imap lfcrlf -b 115200 -p 1 -d 8 -c <usb device file>
    - Example: $ picocom --imap lfcrlf -b 115200 -p 1 -d 8 –c /dev/tty.usbmodem413

Open and Configure Serial Terminal (Putty)

- Open a Linux terminal and enter command: `dmesg`
- Open Putty



Note down device name for Discovery Kit



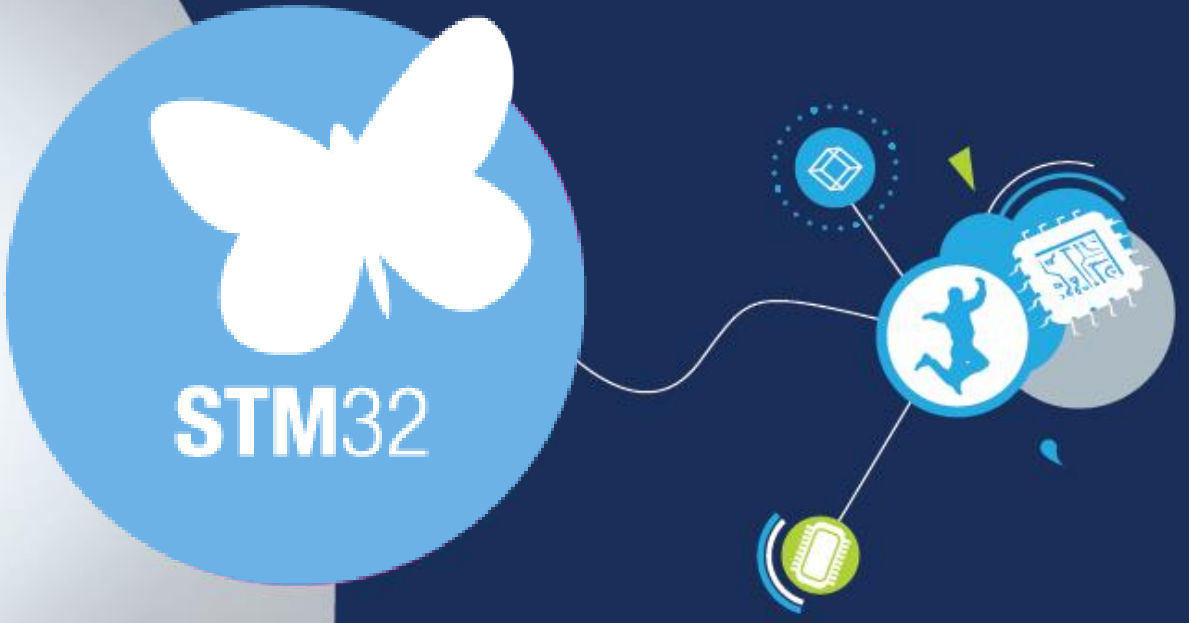
life.augmented

Releasing Your Creativity

/STM32

@ST_World

st.com/e2e



www.st.com/stm32

life.augmented