EDGEX FOUNDRY™

An Introduction

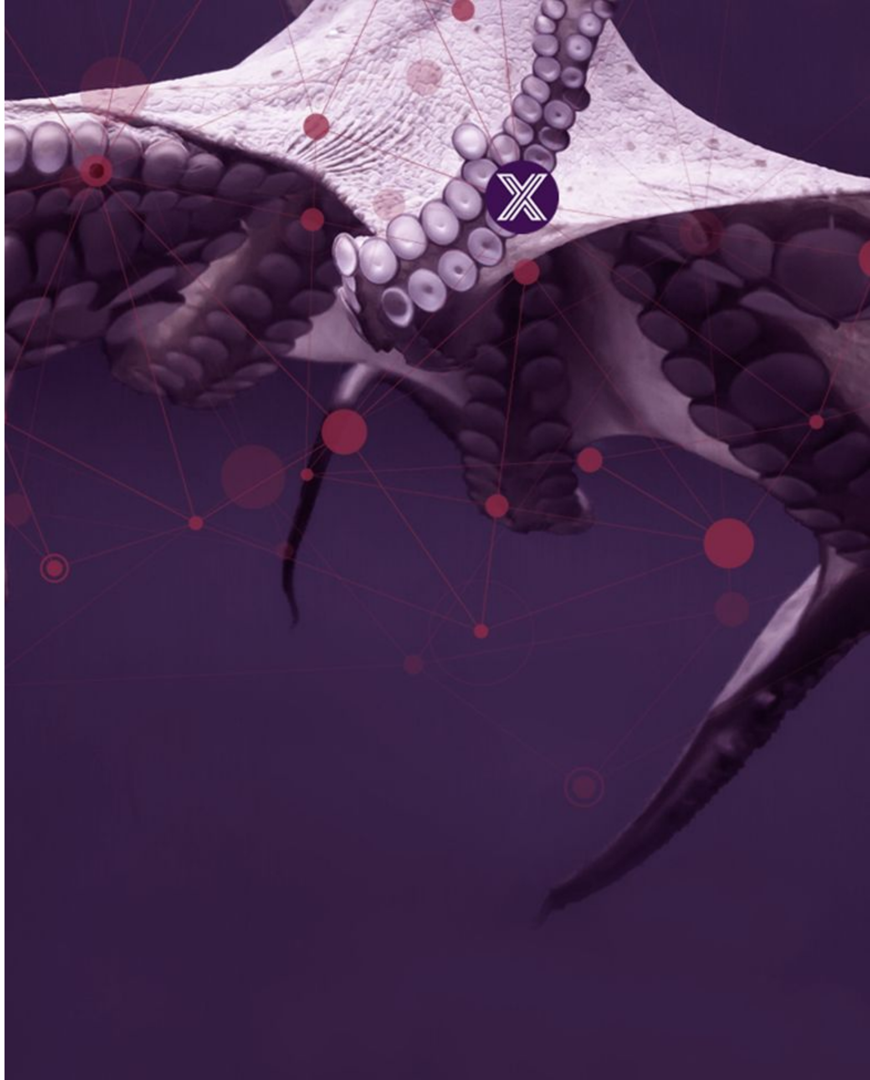
Slides by **Michael Hall**

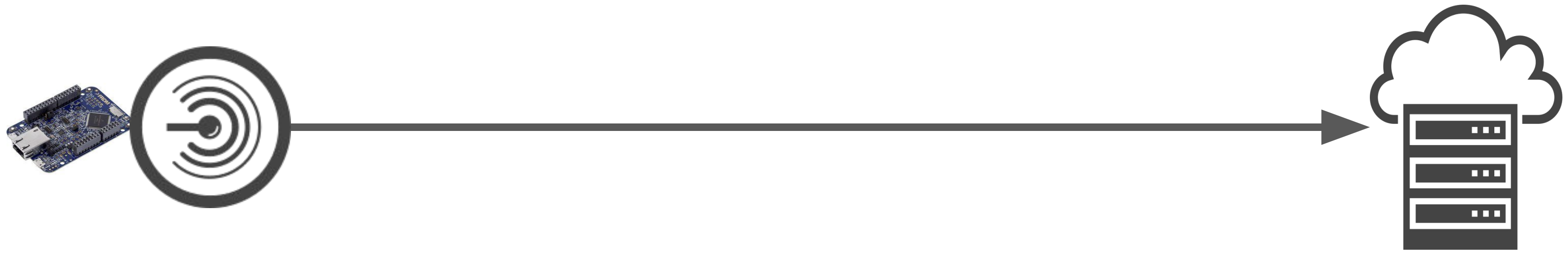
Modified and Presented by **Alex Courouble**

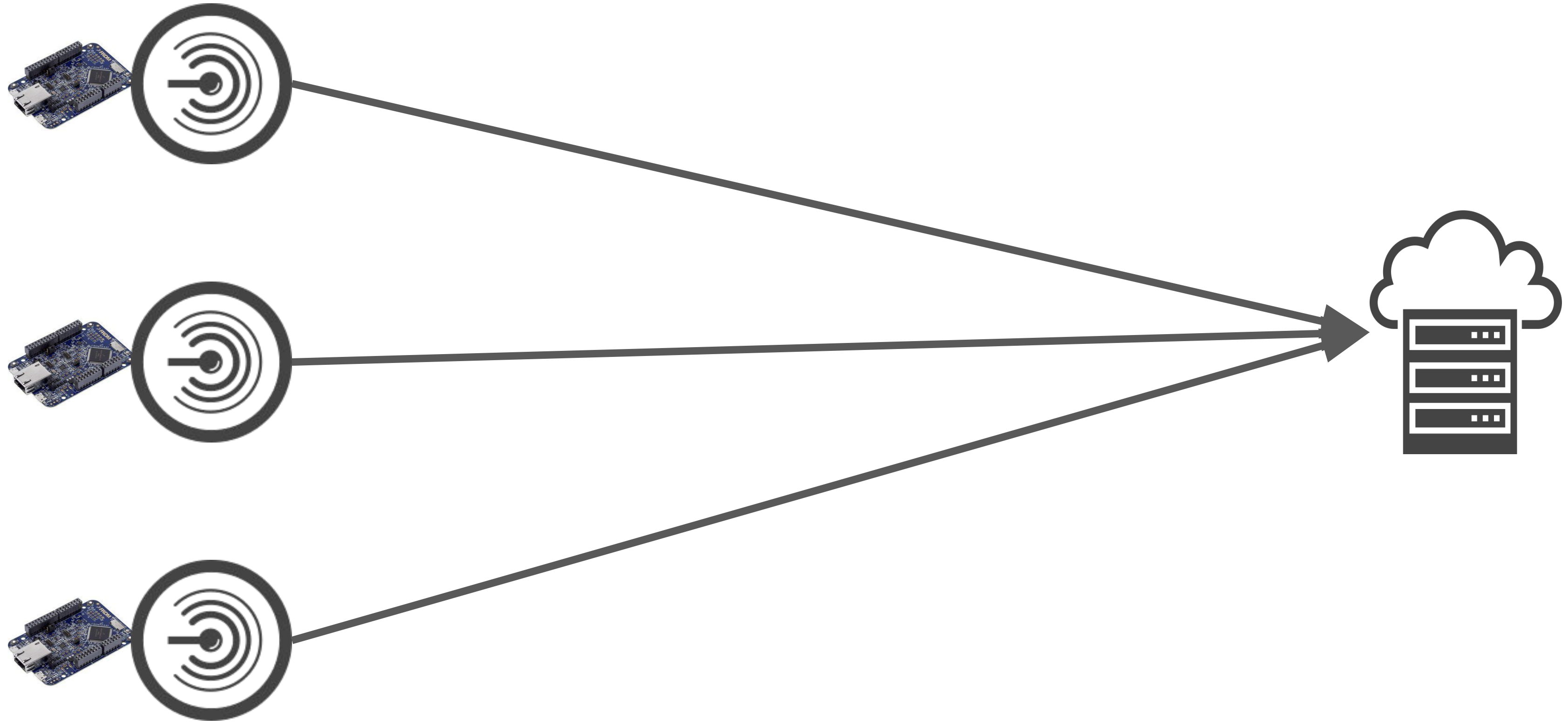
EDGE X FOUNDRY™

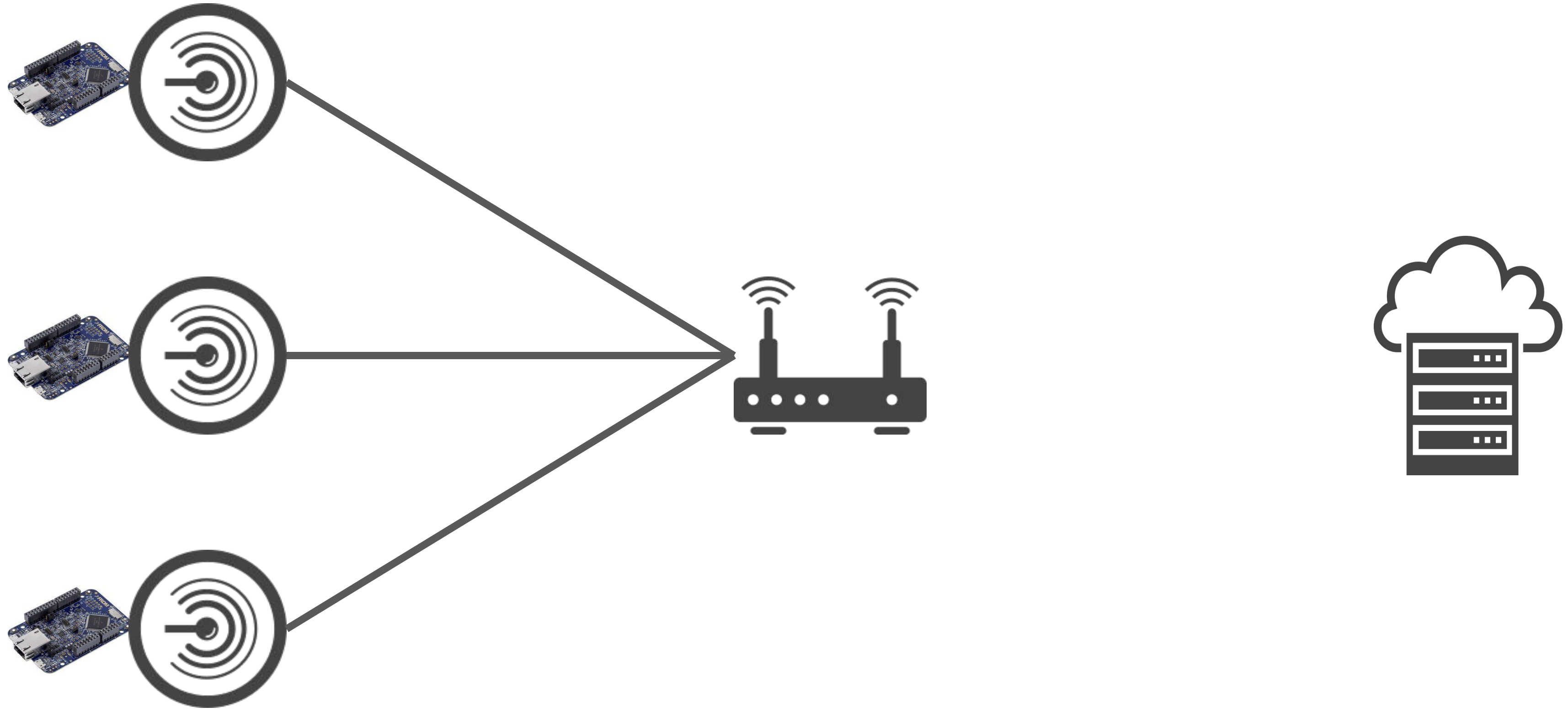
Edge Computing

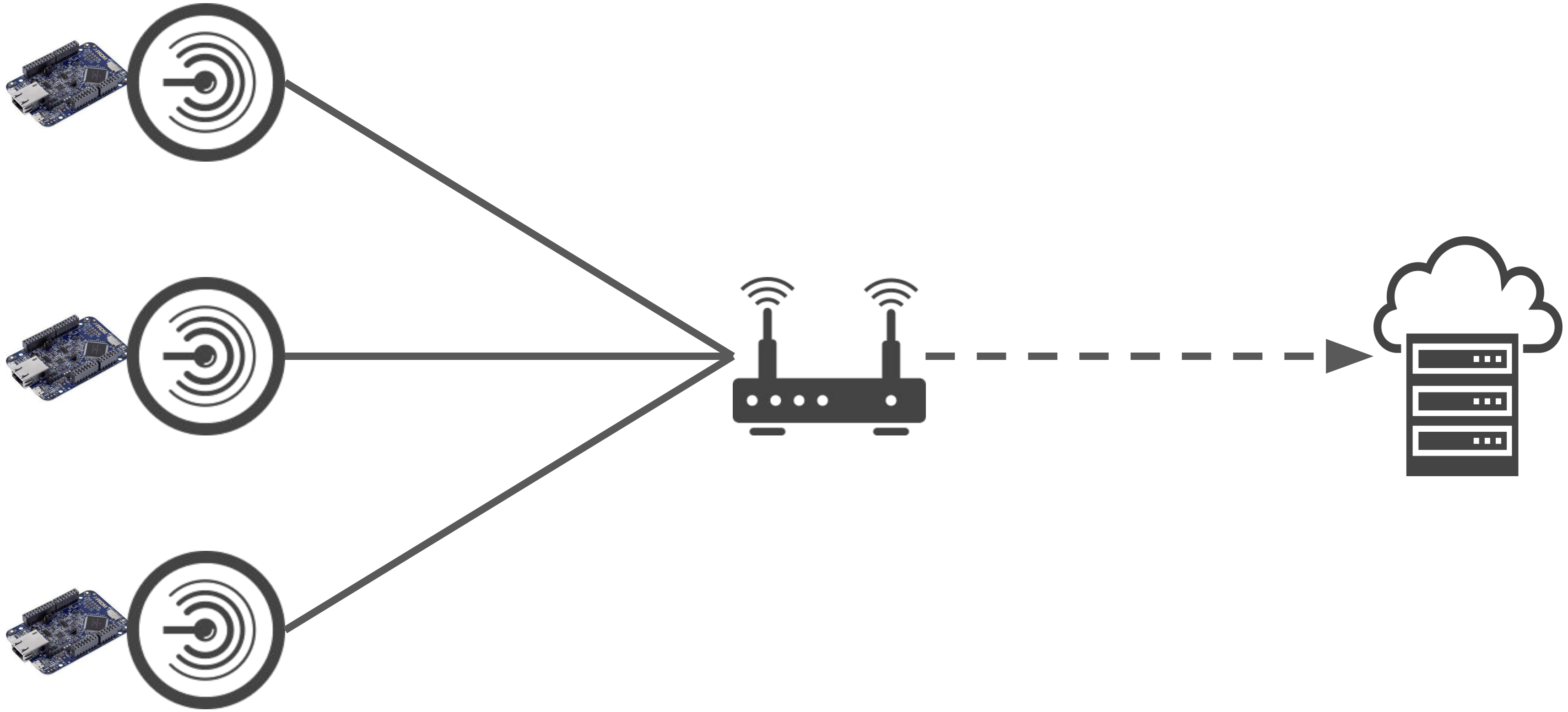
Why do I need it?

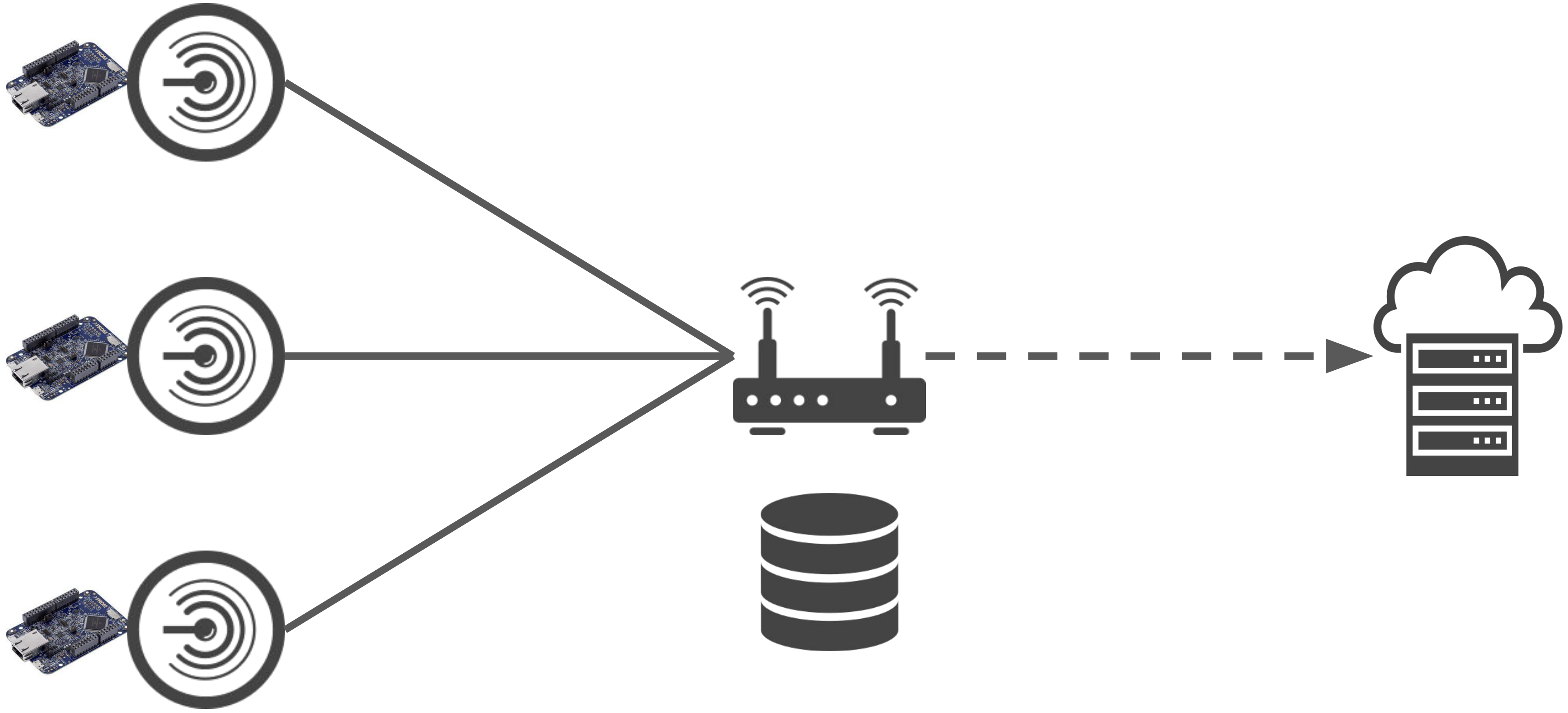


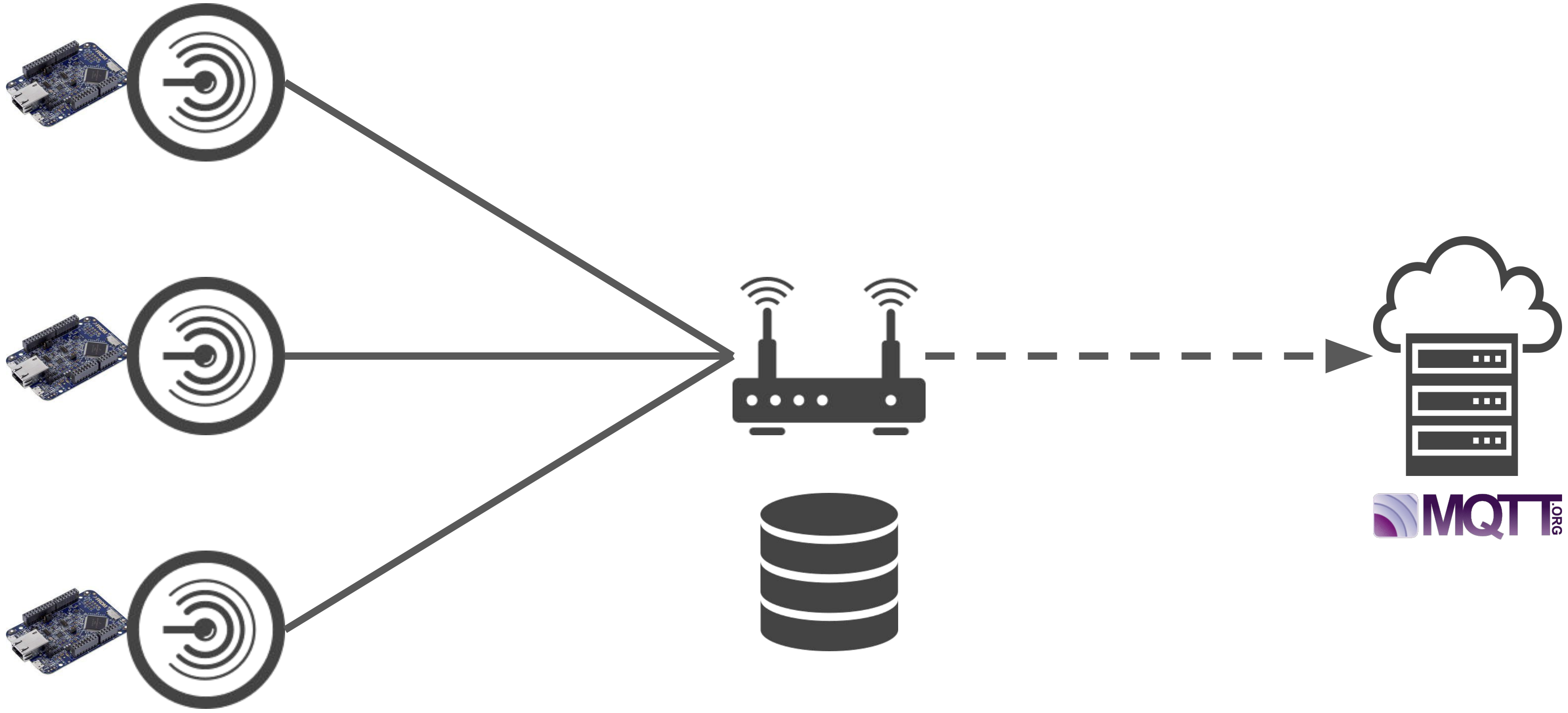


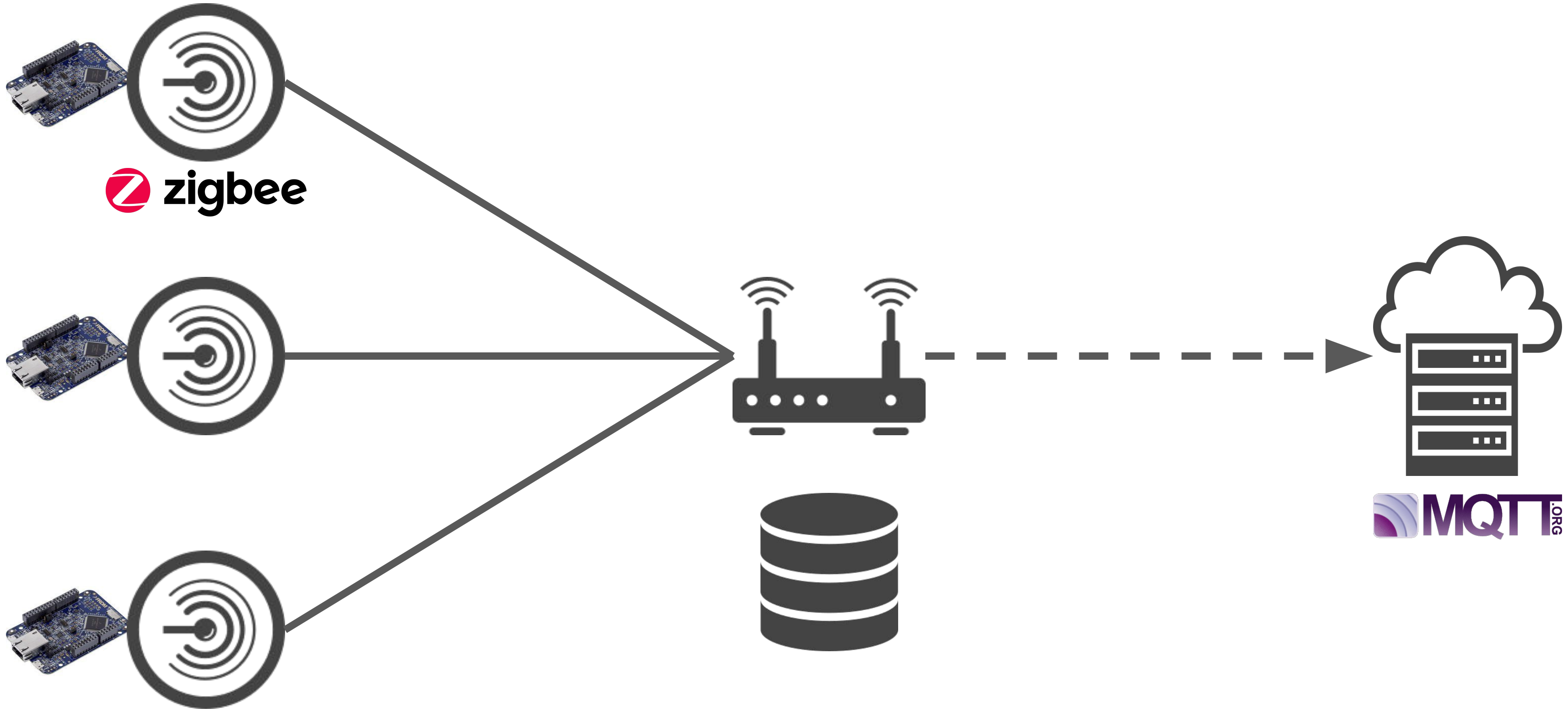


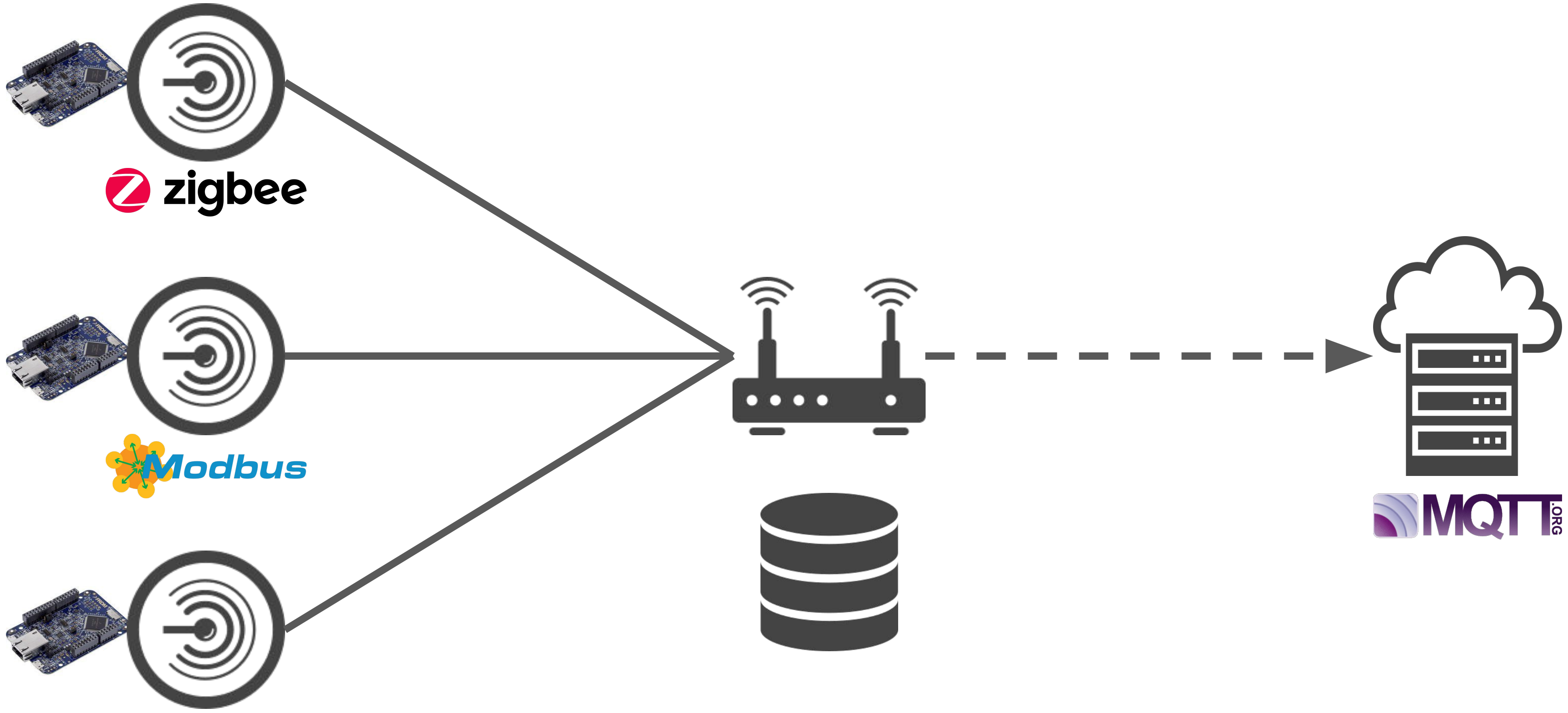


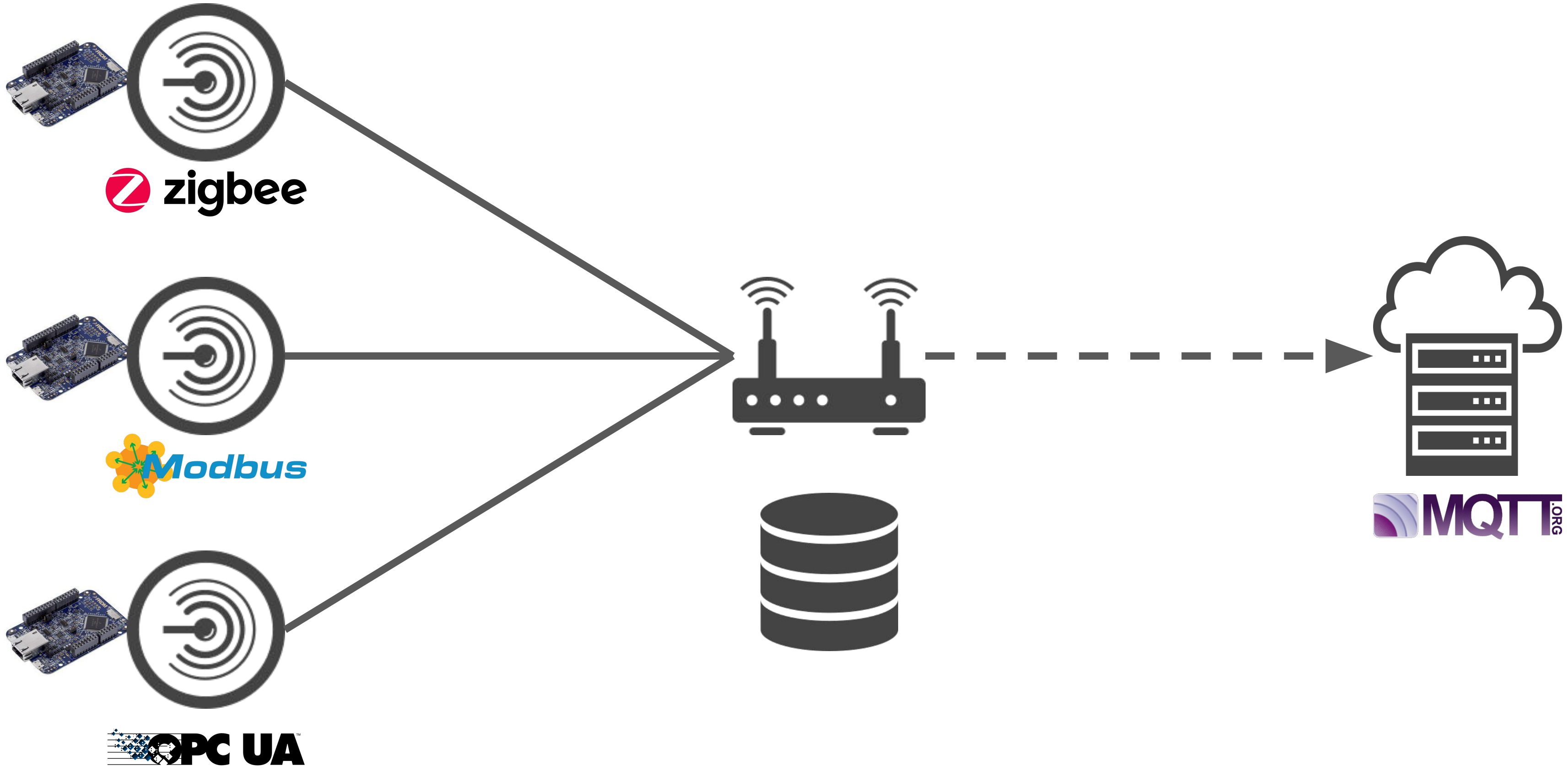


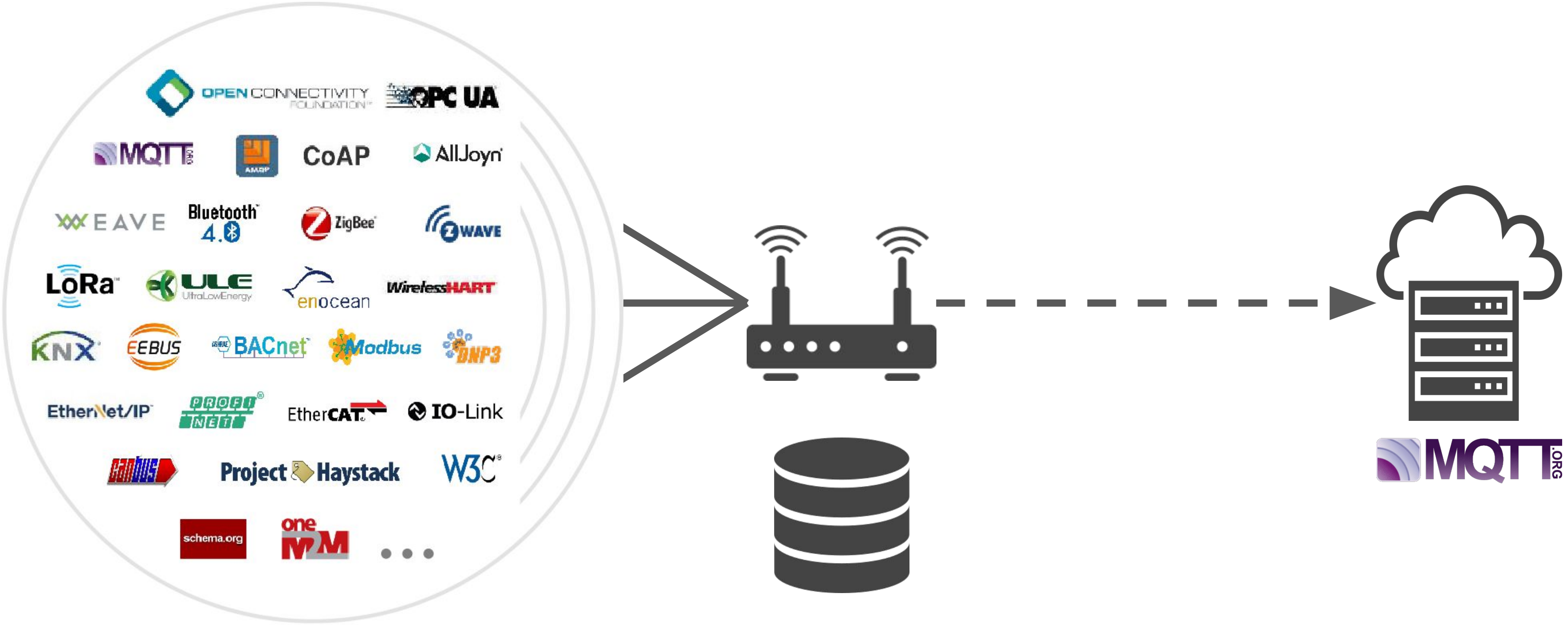


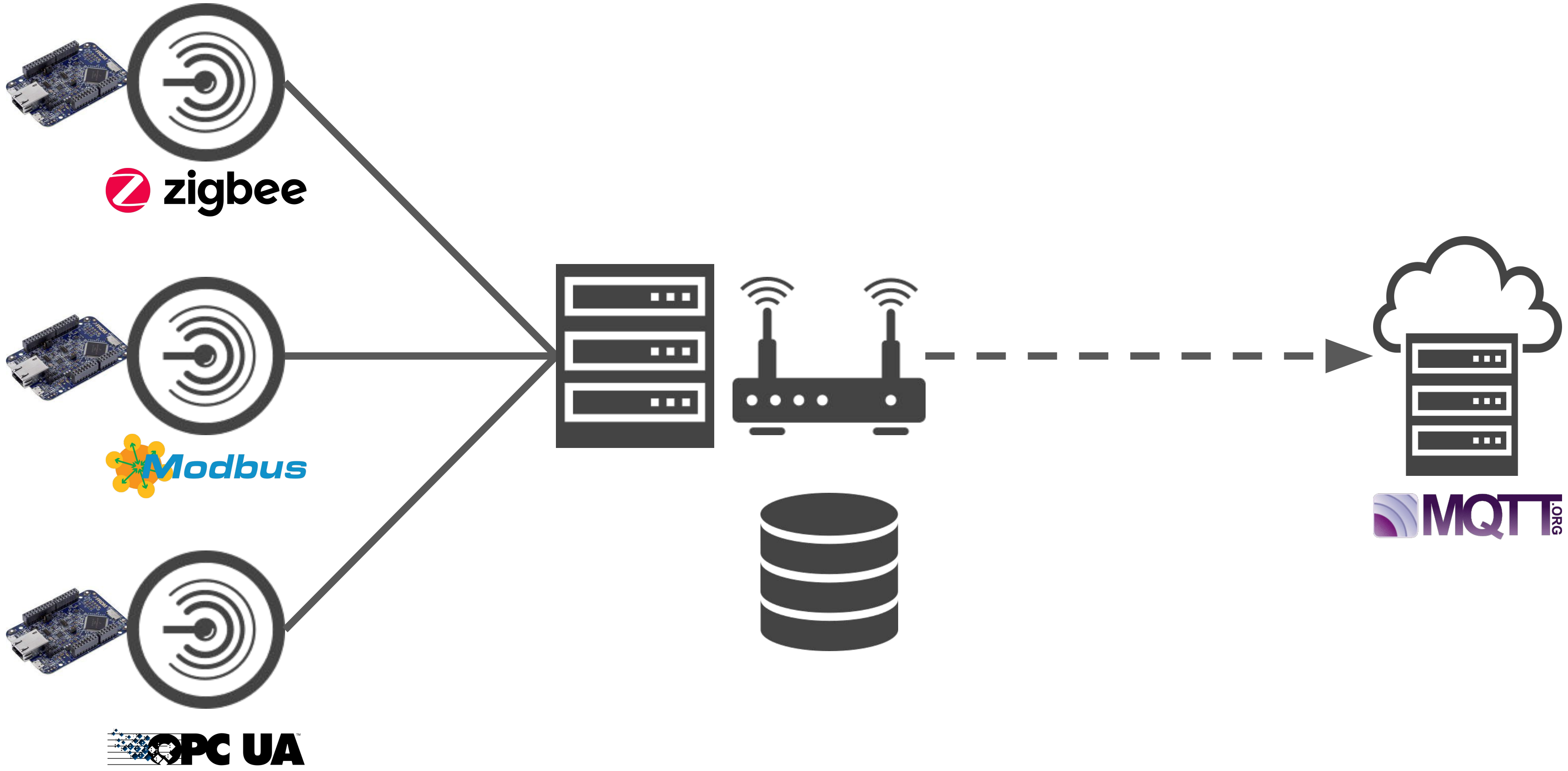

MQTT.ORG

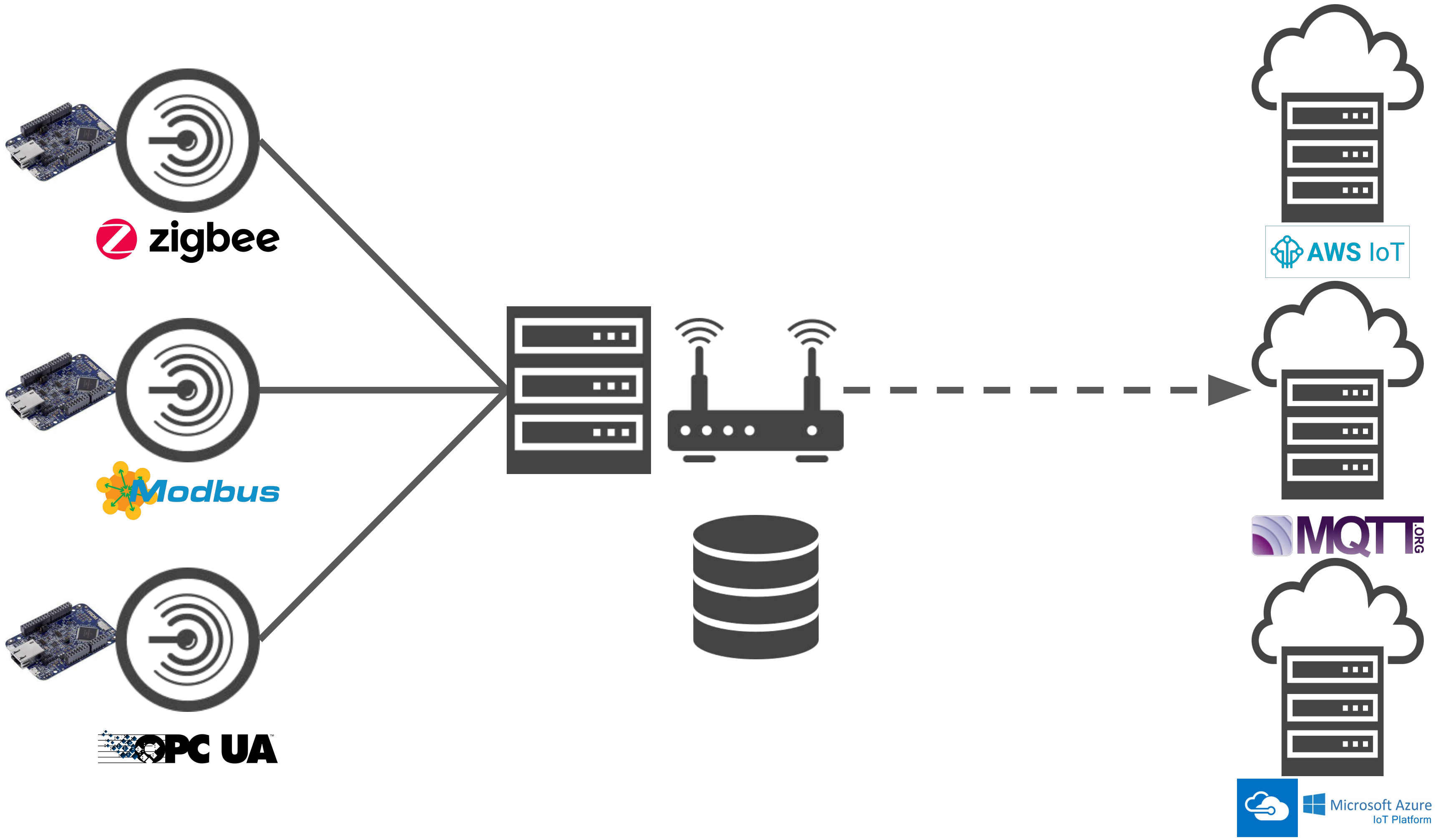

zigbee
MQTT.ORG

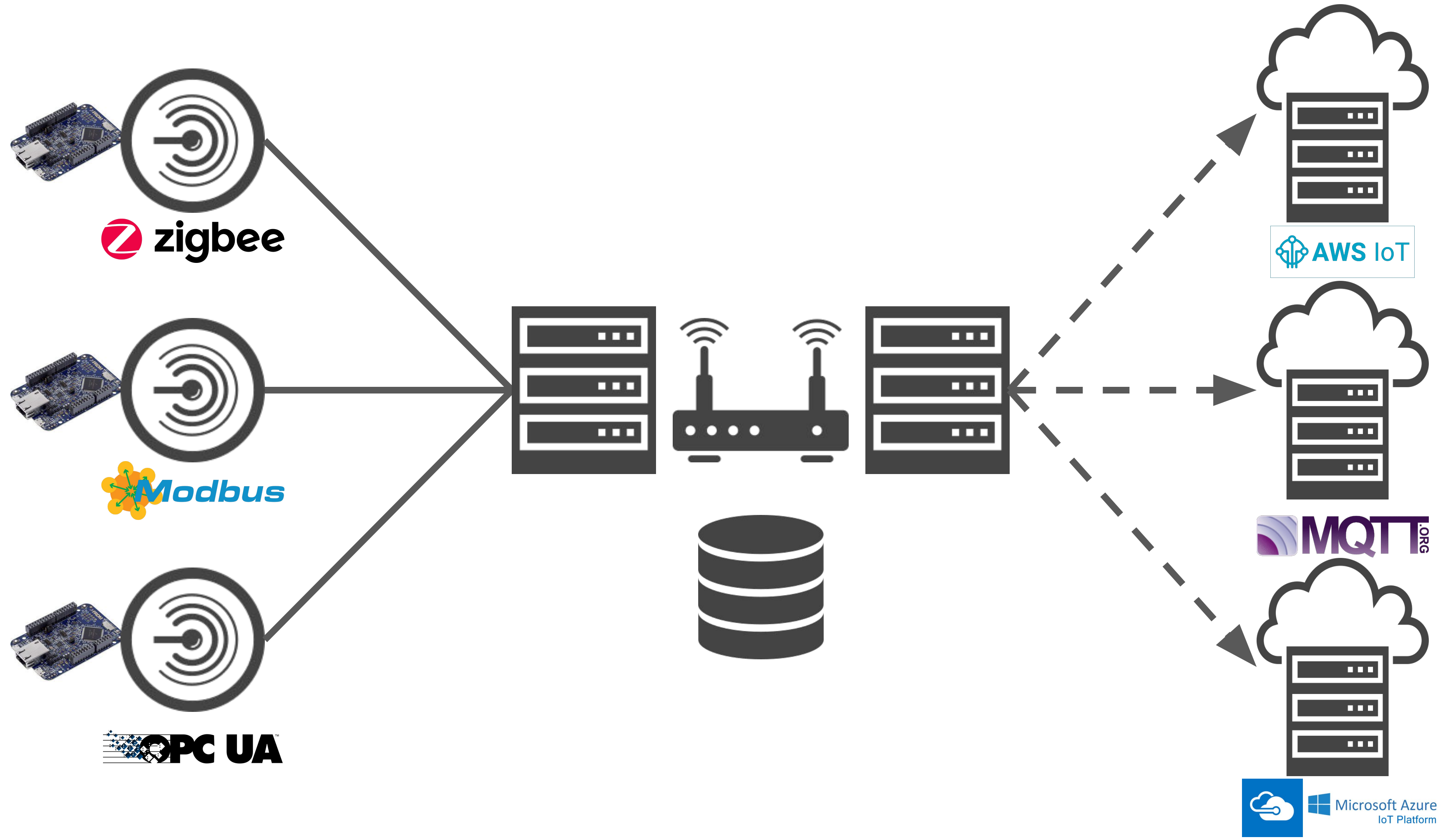

zigbee
Modbus
MQTT.ORG

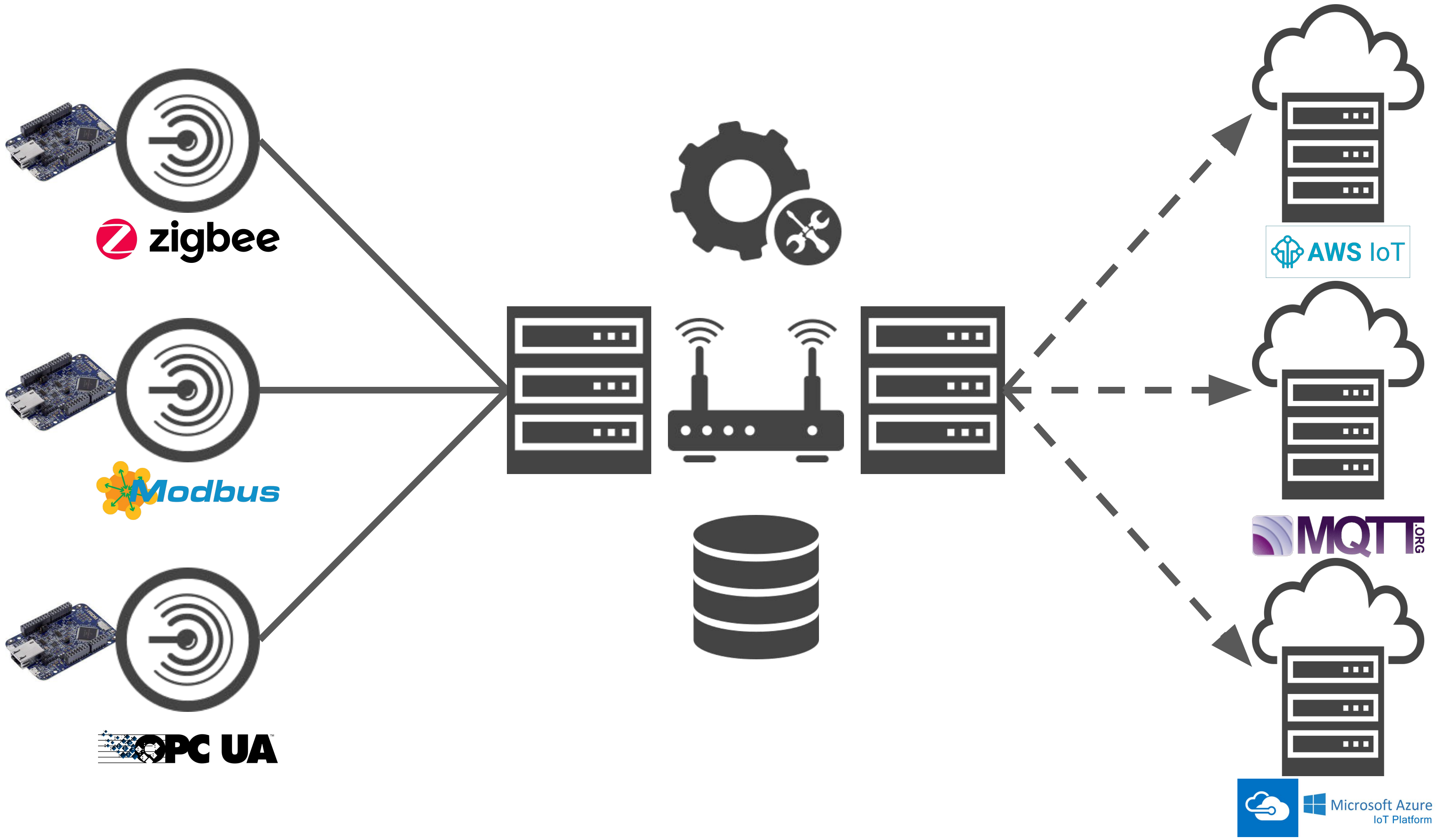

zigbee
Modbus
OPC UA
MQTT.ORG

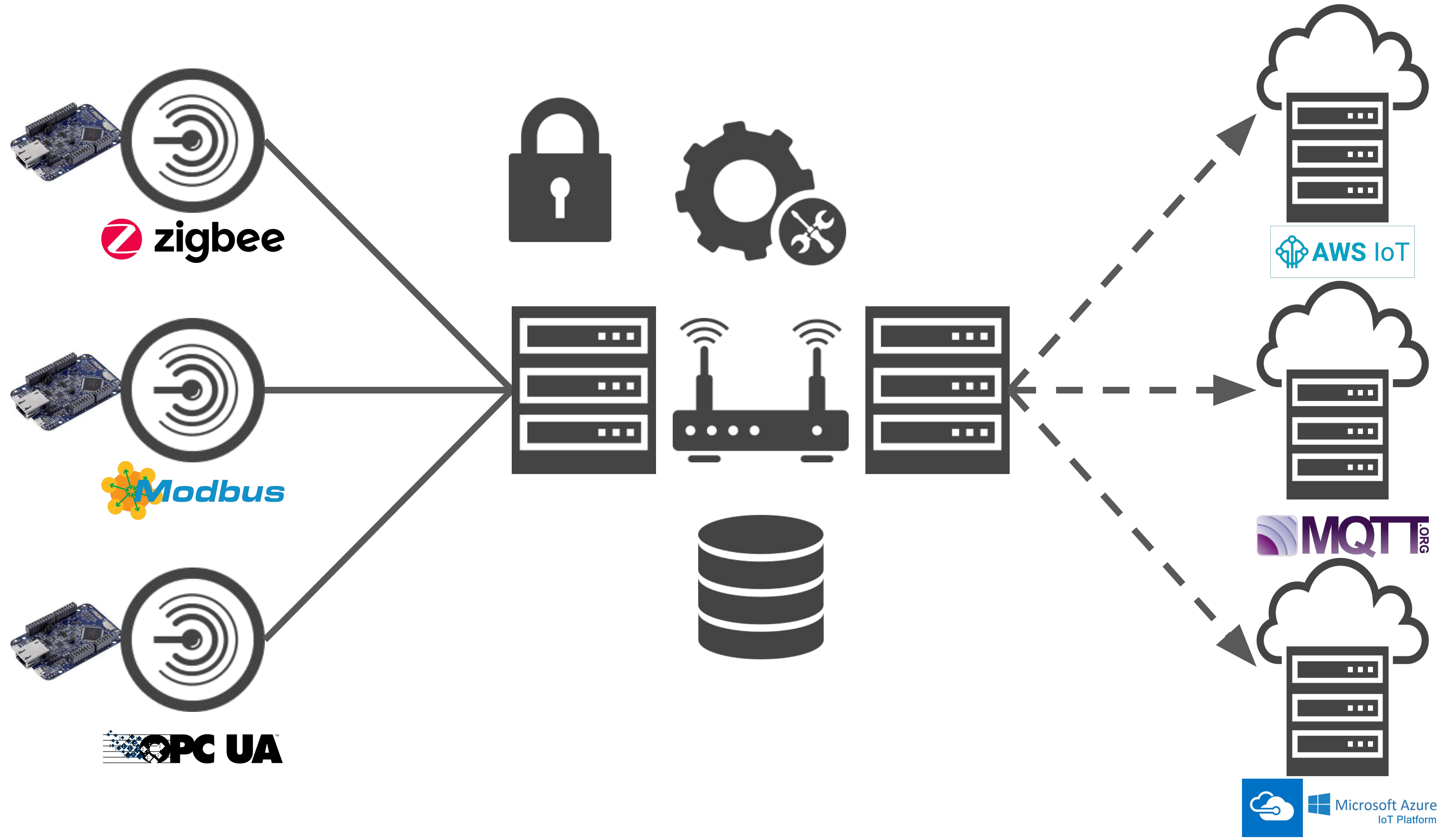

OPEN CONNECTIVITY FOUNDATION
OPC UA
MQTT.ORG
AMQP
CoAP
AllJoyn
WEAVE
Bluetooth 4.0
ZigBee
Z-WAVE
LoRa
ULE UltraLowEnergy
enocean
WirelessHART
KNX
EEBUS
BACnet
Modbus
DNP3
EtherNet/IP
PROFINET
EtherCAT
IO-Link
canbus
Project Haystack
W3C
schema.org
oneM2M
...
MQTT.ORG

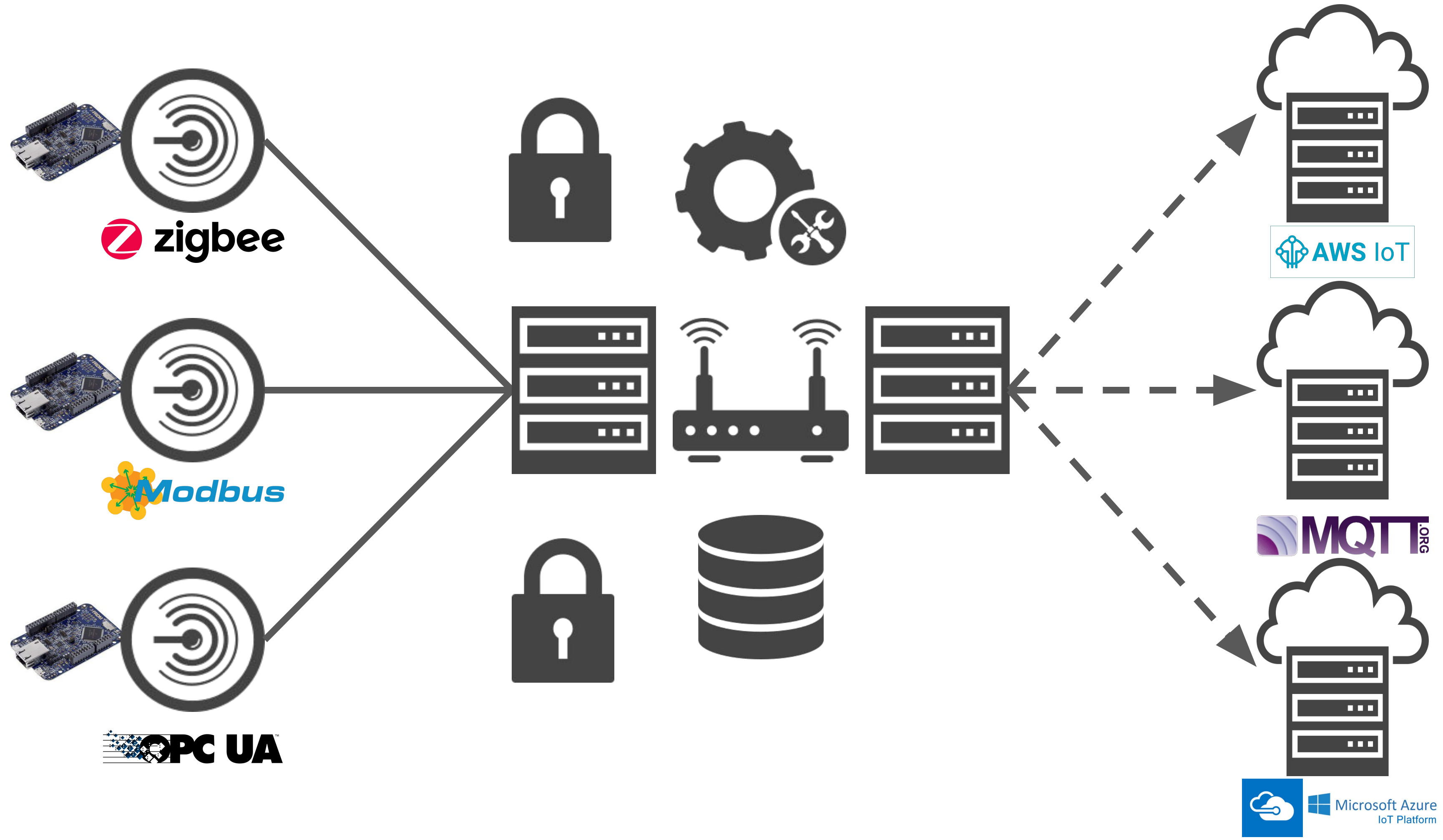

zigbee
Modbus
OPC UA
MQTT.ORG

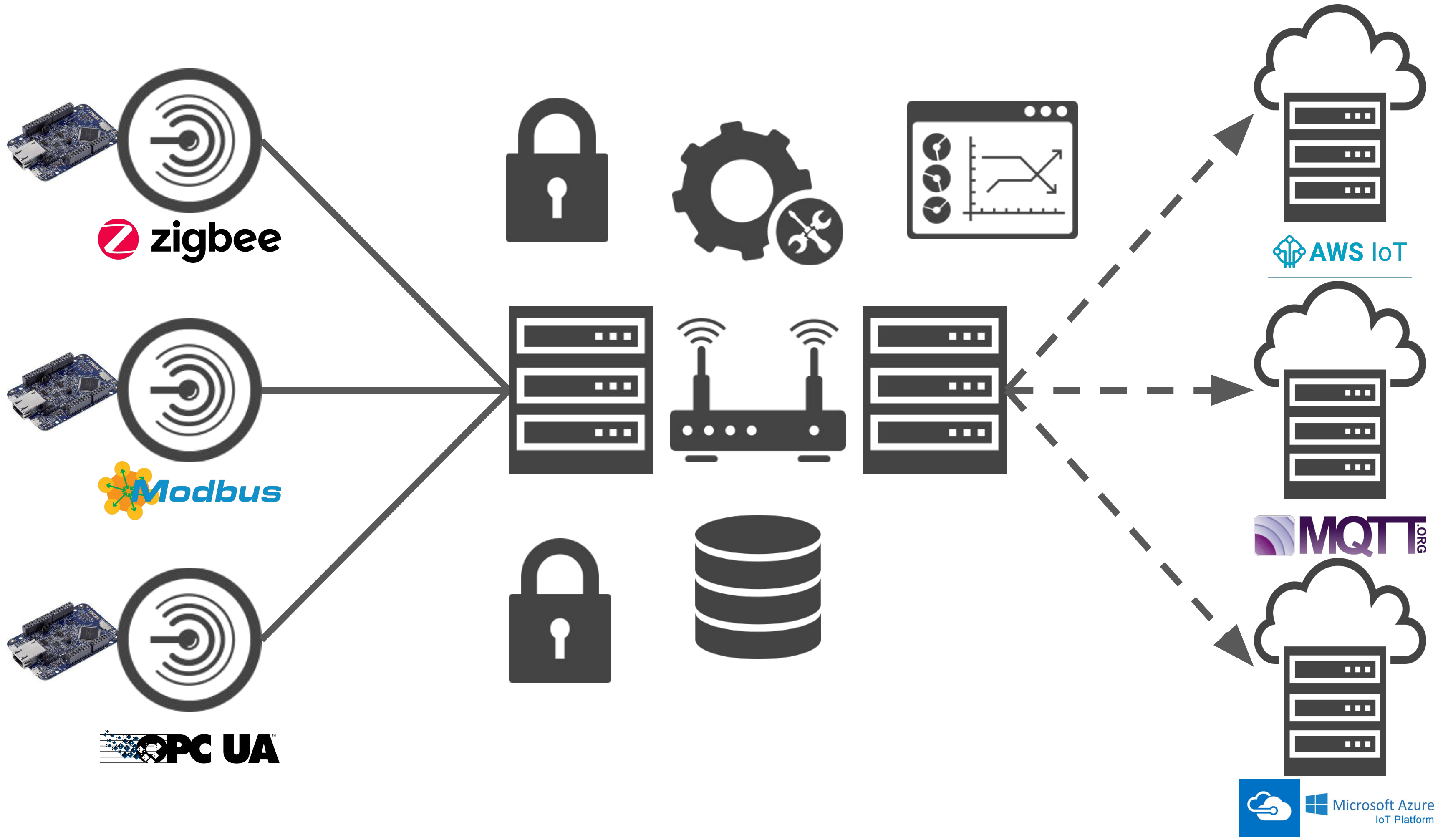

zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform

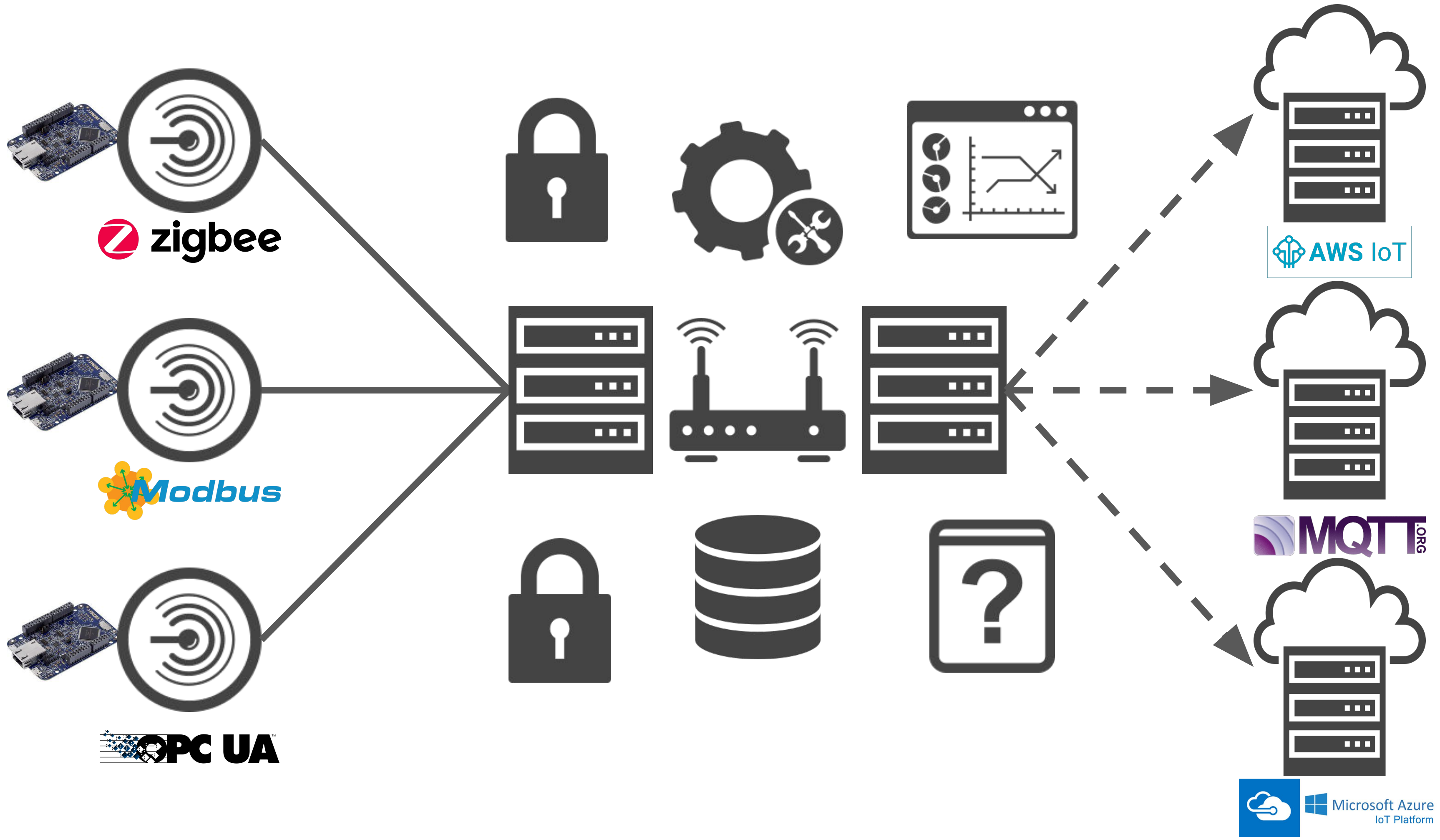

zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform

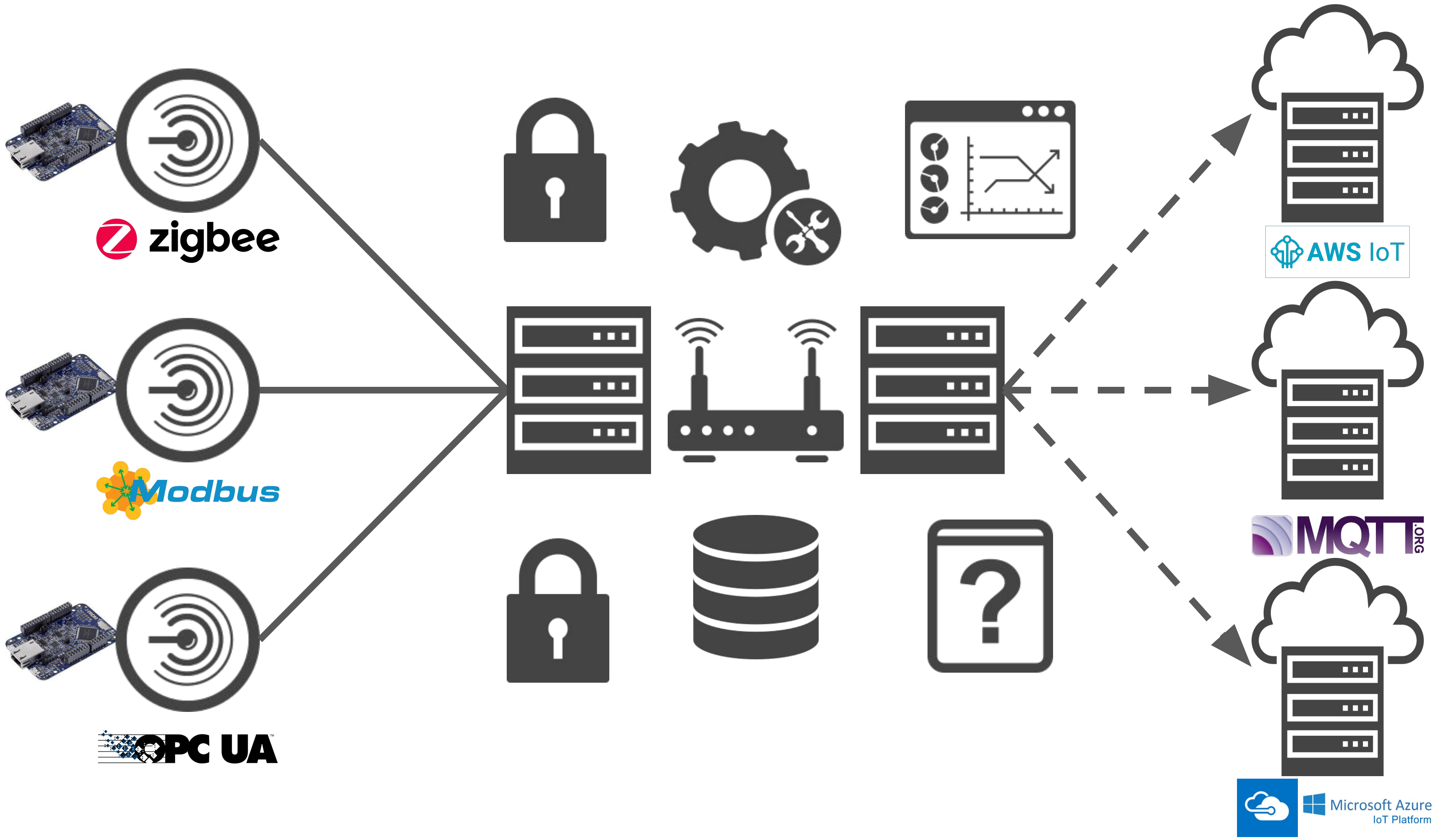

zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform


zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform


zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform

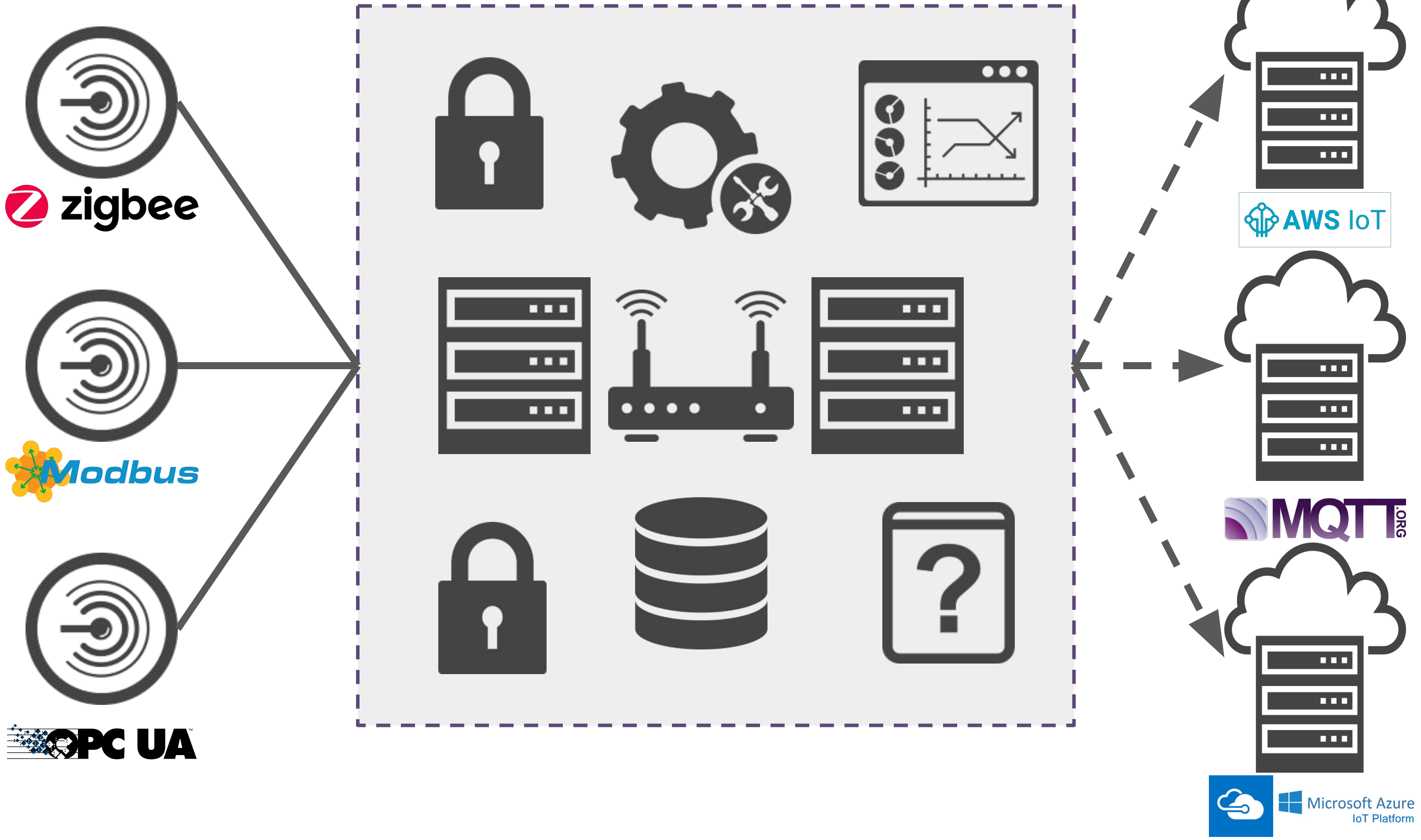

zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform


zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform


zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform

EDGE X FOUNDRY™

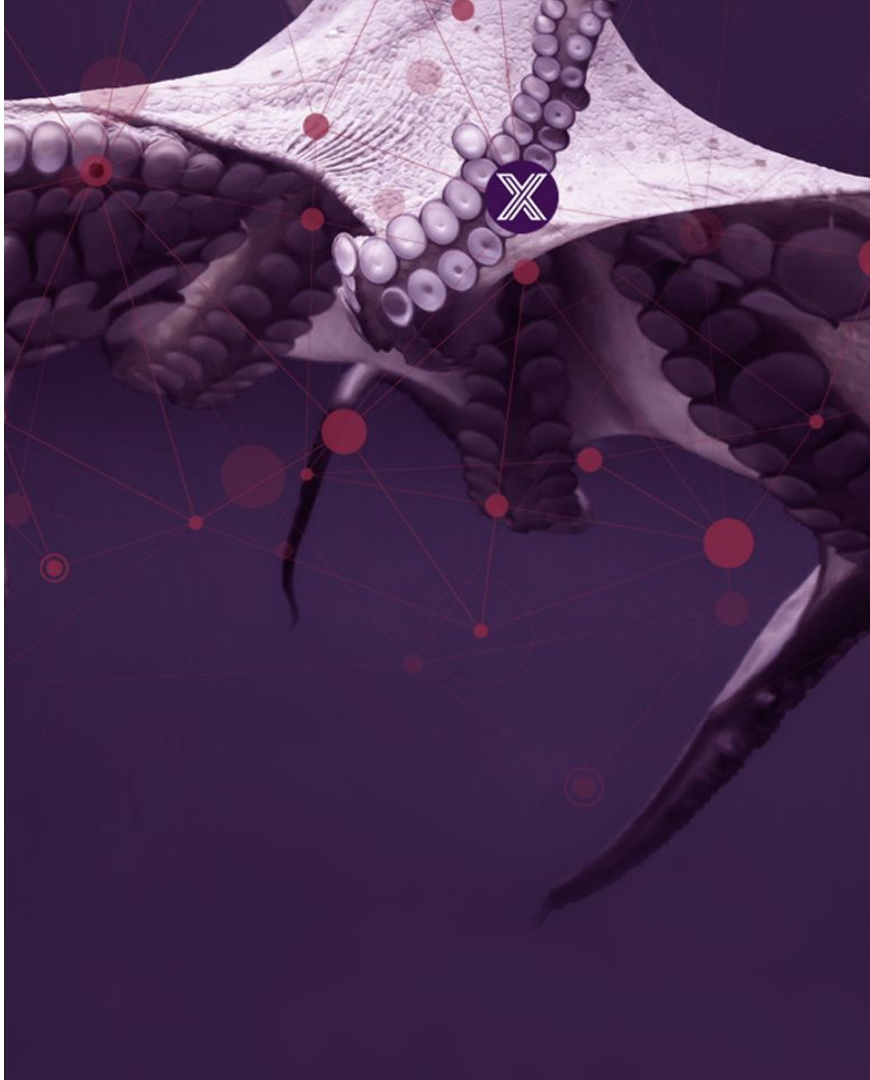
EDGE X FOUNDRY™



EDGE X FOUNDRY™

Who is EdgeX Foundry?

And how to join us



EDGE X FOUNDRY™

Vendor-neutral open source project hosted by The Linux Foundation building a common open framework for IoT edge computing.

Interoperability framework and reference platform to enable an ecosystem of plug-and-play components that unifies the marketplace and accelerates the deployment of IoT solutions.

Architected to be agnostic to protocol, silicon (e.g., x86, ARM), OS (e.g., Linux, Windows, Mac OS), and application environment (e.g., Java, JavaScript, Python, Go Lang, C/C++) to support customer preferences for differentiation

Part of the **LF Edge** project at the Linux Foundation

LF Edge Premium Members

Aricent ALTRAN GROUP
arm
AT&T
Baidu 百度
DELL EMC
DIANOMIC

ERICSSON
hp
Hewlett Packard Enterprise
HUAWEI
IBM
intel

inwinSTACK
JUNIPER NETWORKS
MobiledgeX
NETSIA
NOKIA
NTT

OSIsoft.
Qualcomm Qualcomm Technologies, Inc.
Radisys
redhat
SAMSUNG
SEAGATE

Tencent 腾讯
WIND™
wipro
ZEDEDA

LF Edge General Members

alef
Alleantia
Beechwoods
ubuntu Delivered by Canonical
CertusNet
CloudPlugs Connecting Things
DATA AHEAD
EMQ
enigmedia
EPISENSOR
FOGHORN
FORGEROCK
FOUNDRIES.IO
谐云科技 HARMONY CLOUD
IOTech
IoTium www.iotium.io
KMC CONTROLS
Linaro
MAINFLUX Internet of Things Solutions
MARVELL
MOCANA
NETFOUNDRY SPIN UP YOUR NETWORK
packet
Pluribus NETWORKS
RackN
redislabs home of redis
REPLY
Section
Vapor
vitro
Volterra EDGE SERVICES
萬向集團 WANXIANG GROUP

Associate Members

AECC AUTOMOTIVE EDGE COMPUTING CONSORTIUM
北京邮电大学 Beijing University of Posts and Telecommunications
ETRI Electronics and Telecommunications Research Institute
INFRASTRUCTURE MASONS
Project Haystack

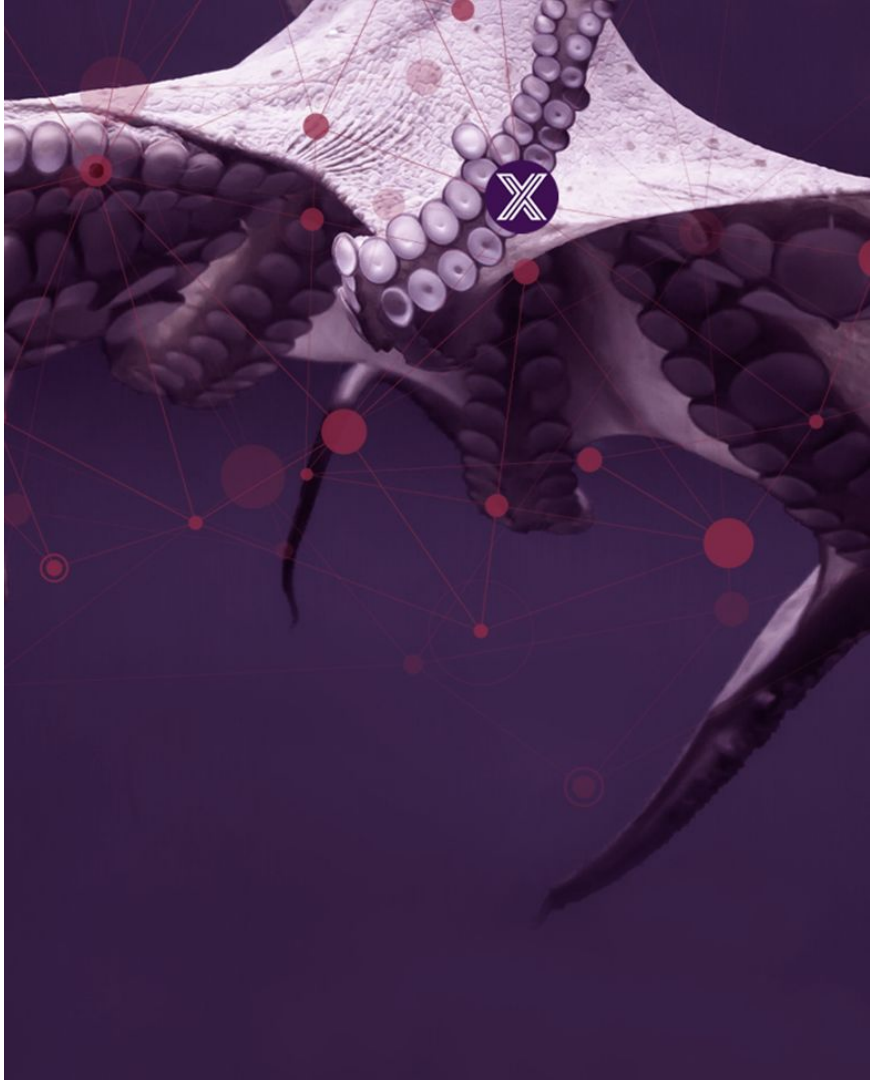
Getting Involved

- Open Source and contributor driven, anybody can participate
- TSC and WG meetings open to public
- Technical leadership (TSC & WG chairs) elected by technical contributors

- GitHub:
  - https://github.com/edgexfoundry
- Documentation
  - https://docs.edgexfoundry.org
- Slack
  - https://slack.edgexfoundry.org
- Mailing Lists
  - https://lists.edgexfoundry.org
  - https://lists.edgexfoundry.org/calendar

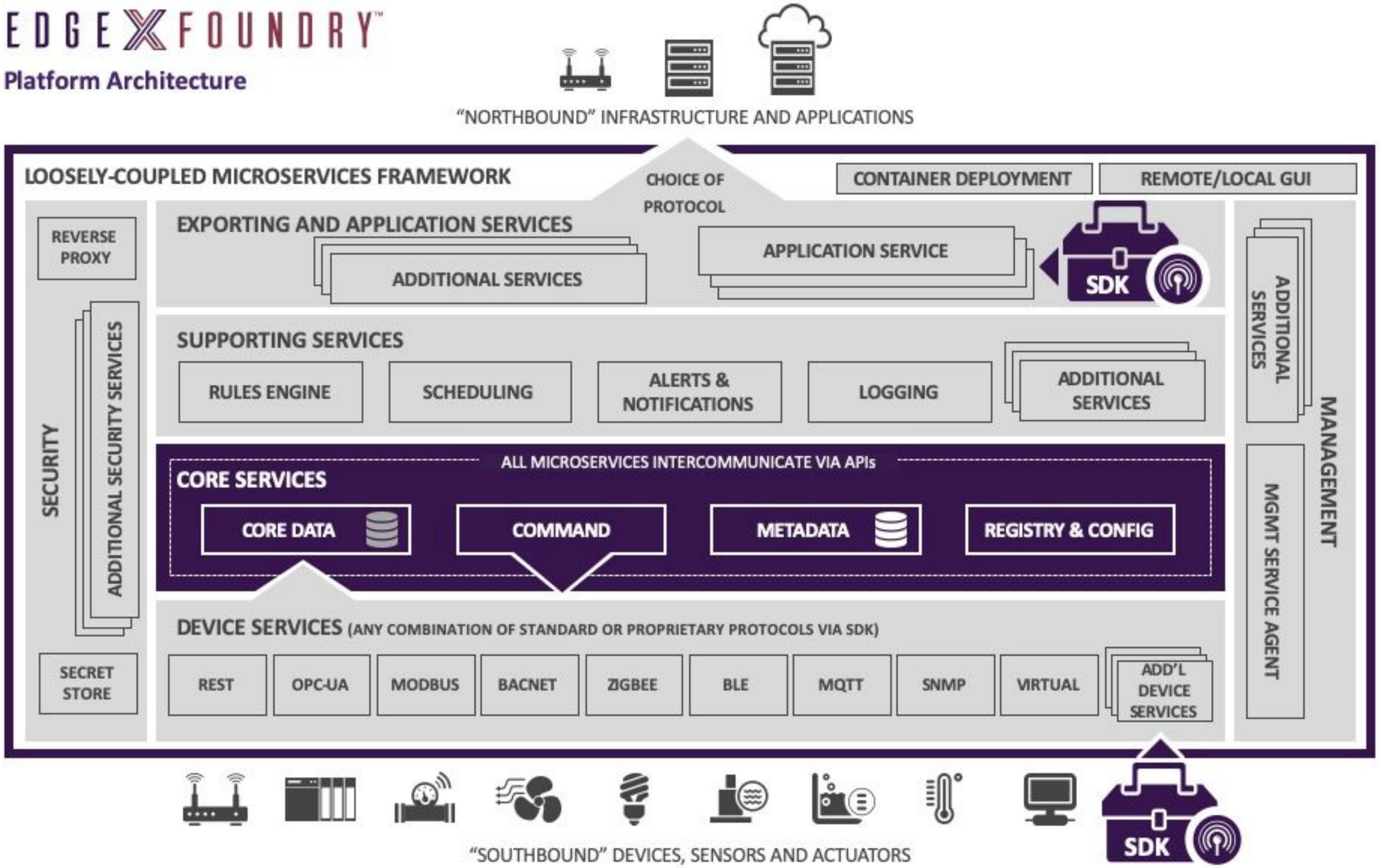
What is EdgeX?

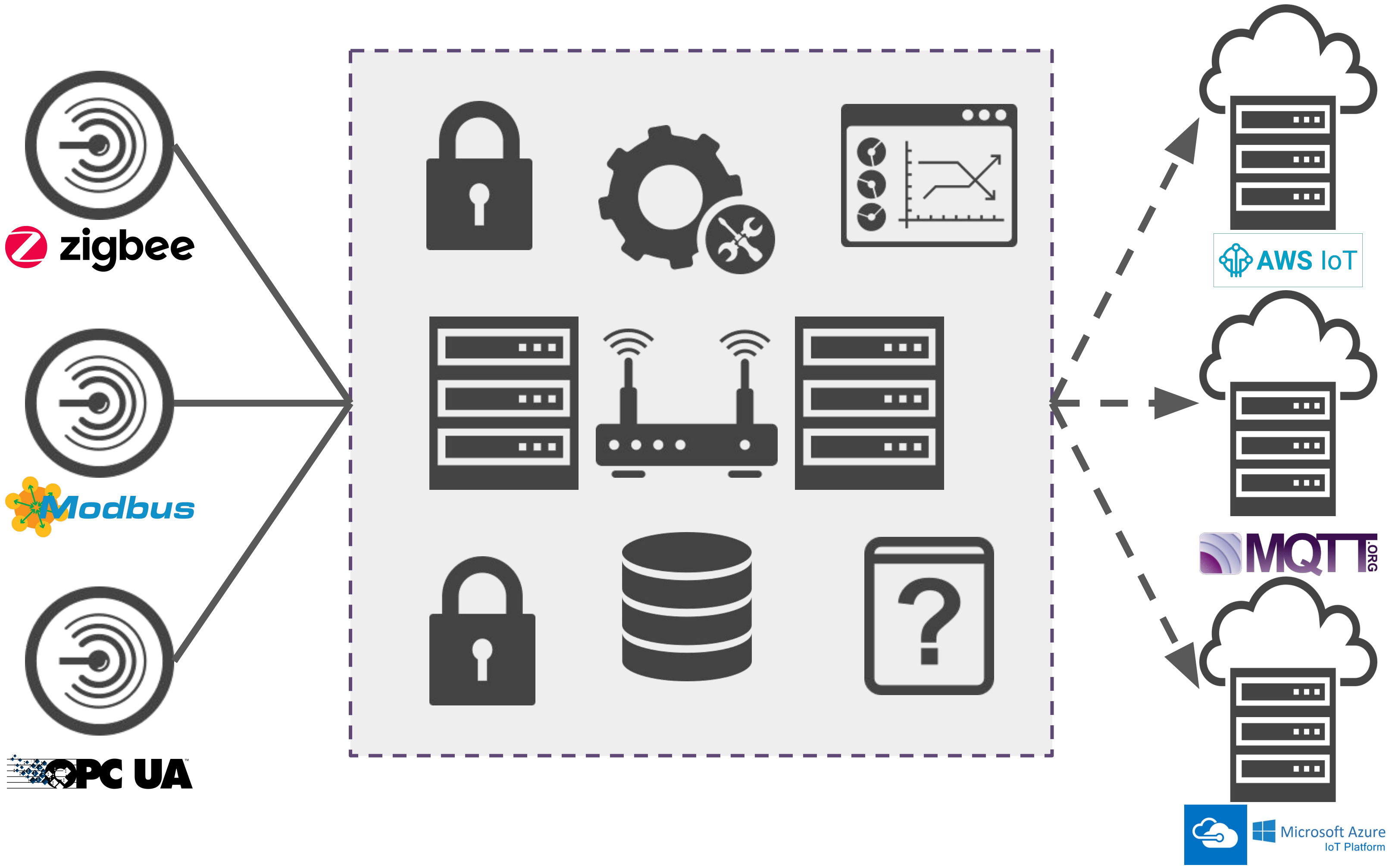
Microservices and Deployments

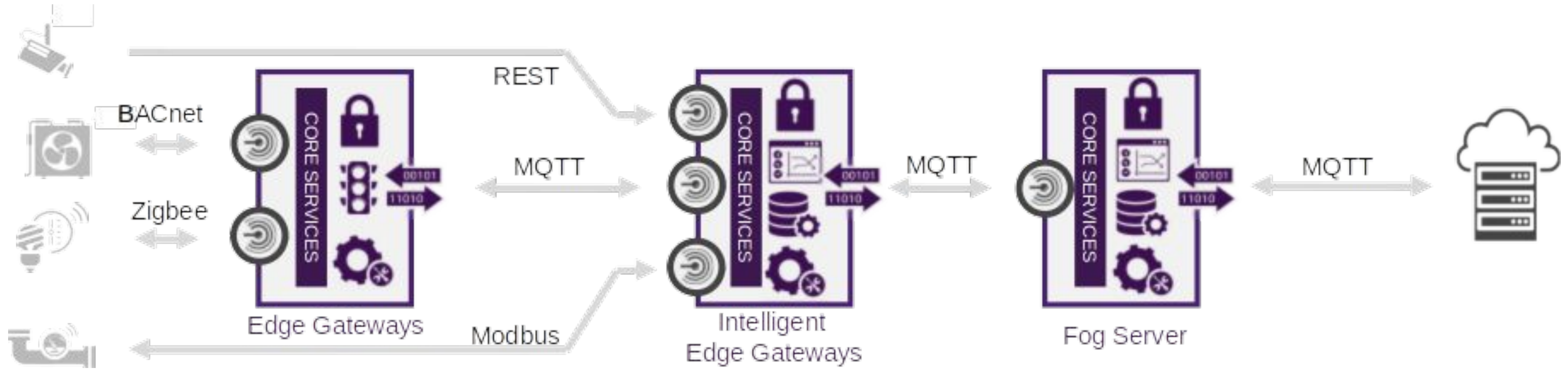


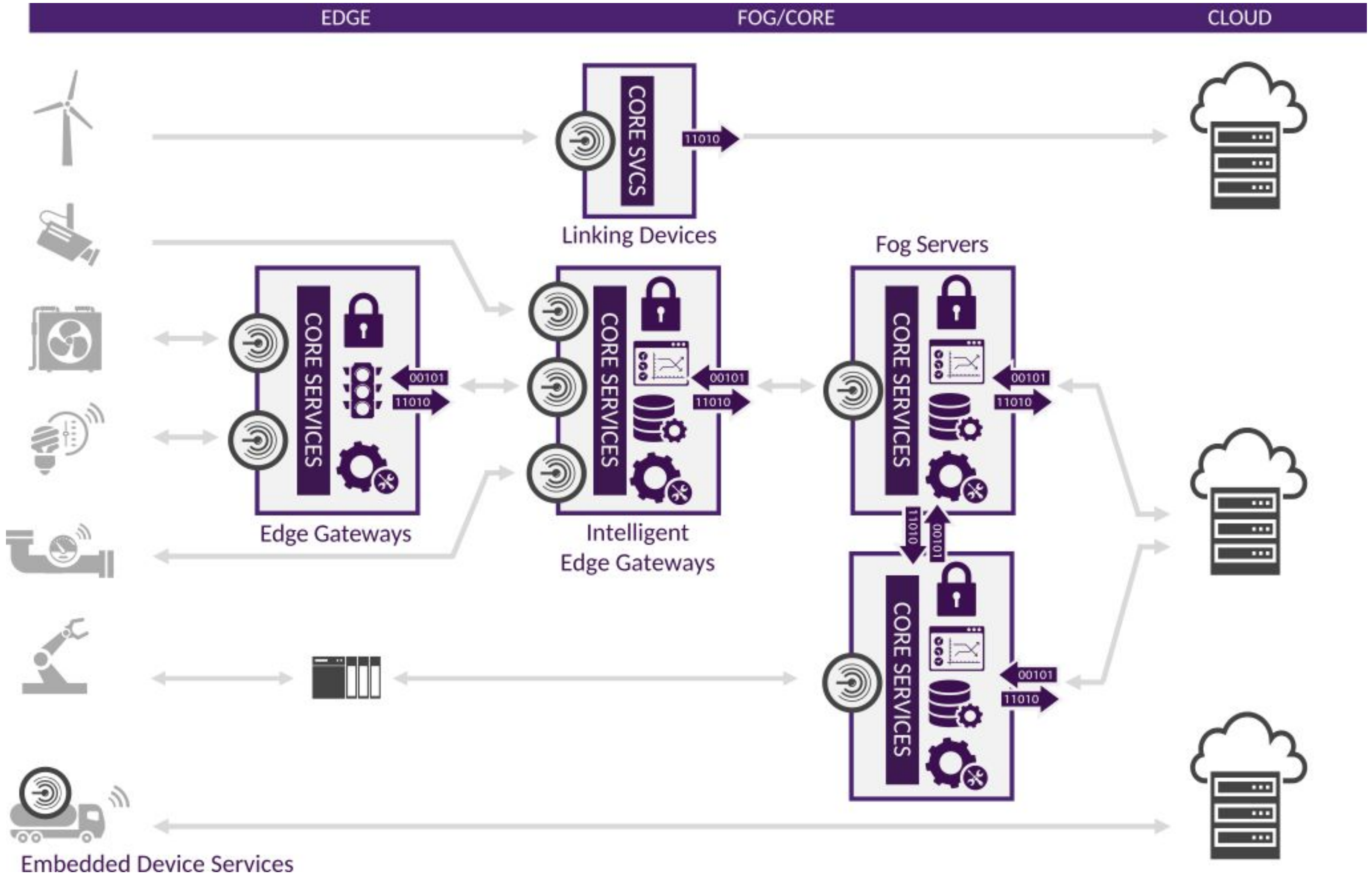
EDGE X FOUNDRY™

Platform Architecture




zigbee
Modbus
OPC UA
AWS IoT
MQTT.ORG
Microsoft Azure
IoT Platform


REST
BACnet
Zigbee
CORE SERVICES
00101
11010
Edge Gateways
MQTT
Modbus
CORE SERVICES
00101
11010
Intelligent
Edge Gateways
MQTT
CORE SERVICES
00101
11010
Fog Server
MQTT


EDGE
FOG/CORE
CLOUD
CORE SVCS
11010
Linking Devices
Fog Servers
CORE SERVICES
00101
11010
Edge Gateways
Intelligent Edge Gateways
Embedded Device Services

Walkthrough

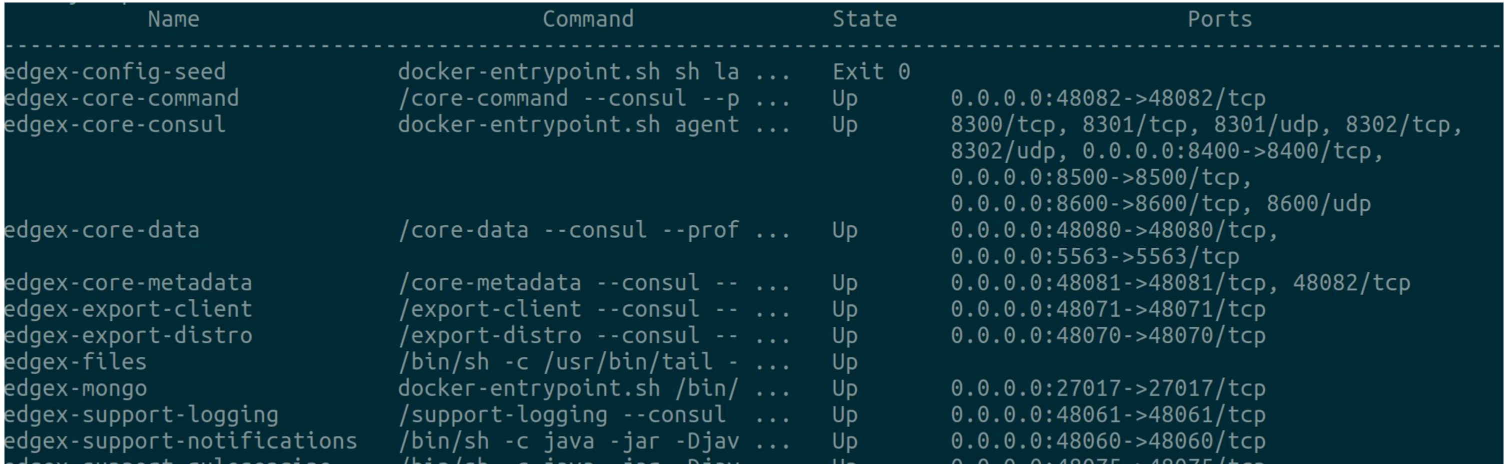
Let's see it in action

Get Started in Three Steps

1. Run the EgdeX Microservices with Docker Compose
2. Create a Device Service with **device-sdk-go**
3. Create an Application Service with **app-function-sdk-go**

Deploying with Docker

- Install docker & docker-compose
- Download the compose file from the developer-scripts repo:
  - https://raw.githubusercontent.com/edgexfoundry/developer-scripts/master/releases/edinburgh/compose-files/docker-compose-edinburgh-no-secty-1.0.1.yml
- docker-compose -f docker-compose-edinburgh-no-secty-1.0.1.yml up -d



```
          Name                         Command               State                      Ports
-------------------------------------------------------------------------------------------------------------------
edgex-config-seed             docker-entrypoint.sh sh la ...   Exit 0
edgex-core-command            /core-command --consul --p ...   Up      0.0.0.0:48082->48082/tcp
edgex-core-consul             docker-entrypoint.sh agent ...   Up      8300/tcp, 8301/tcp, 8301/udp, 8302/tcp,
                                                                       8302/udp, 0.0.0.0:8400->8400/tcp,
                                                                       0.0.0.0:8500->8500/tcp,
                                                                       0.0.0.0:8600->8600/tcp, 8600/udp
edgex-core-data               /core-data --consul --prof ...   Up      0.0.0.0:48080->48080/tcp,
                                                                       0.0.0.0:5563->5563/tcp
edgex-core-metadata           /core-metadata --consul -- ...   Up      0.0.0.0:48081->48081/tcp, 48082/tcp
edgex-export-client           /export-client --consul -- ...   Up      0.0.0.0:48071->48071/tcp
edgex-export-distro           /export-distro --consul -- ...   Up      0.0.0.0:48070->48070/tcp
edgex-files                   /bin/sh -c /usr/bin/tail - ...   Up
edgex-mongo                   docker-entrypoint.sh /bin/ ...   Up      0.0.0.0:27017->27017/tcp
edgex-support-logging         /support-logging --consul  ...   Up      0.0.0.0:48061->48061/tcp
edgex-support-notifications   /bin/sh -c java -jar -Djav ...   Up      0.0.0.0:48060->48060/tcp
```

Creating a Device Service

- Define a **Device Profile**
- Implement the device sdk **functions**
- **Build** and **Run** the service
- Tutorial: https://docs.edgexfoundry.org/Ch-GettingStartedSDK-Go.html

Defining your device - Device Profile

```
name: "camera monitor profile"
manufacturer: "Dell"
model: "Cam12345"
labels:
    - "camera"
description: "Human and canine camera monitor profile"
commands:
  -
    (Next Slide)
```

Defining your device - Device Profile - Commands

```
commands:
  -
    name: People
    get:
        path: "/api/v1/devices/{deviceId}/peoplecount"
        responses:
          -
            code: "200"
            description: "Number of people on camera"
            expectedValues: ["humancount"]
          -
            code: "503"
            description: "service unavailable"
            expectedValues: ["cameraerror"]
```

Defining your device - Device Profile - Commands

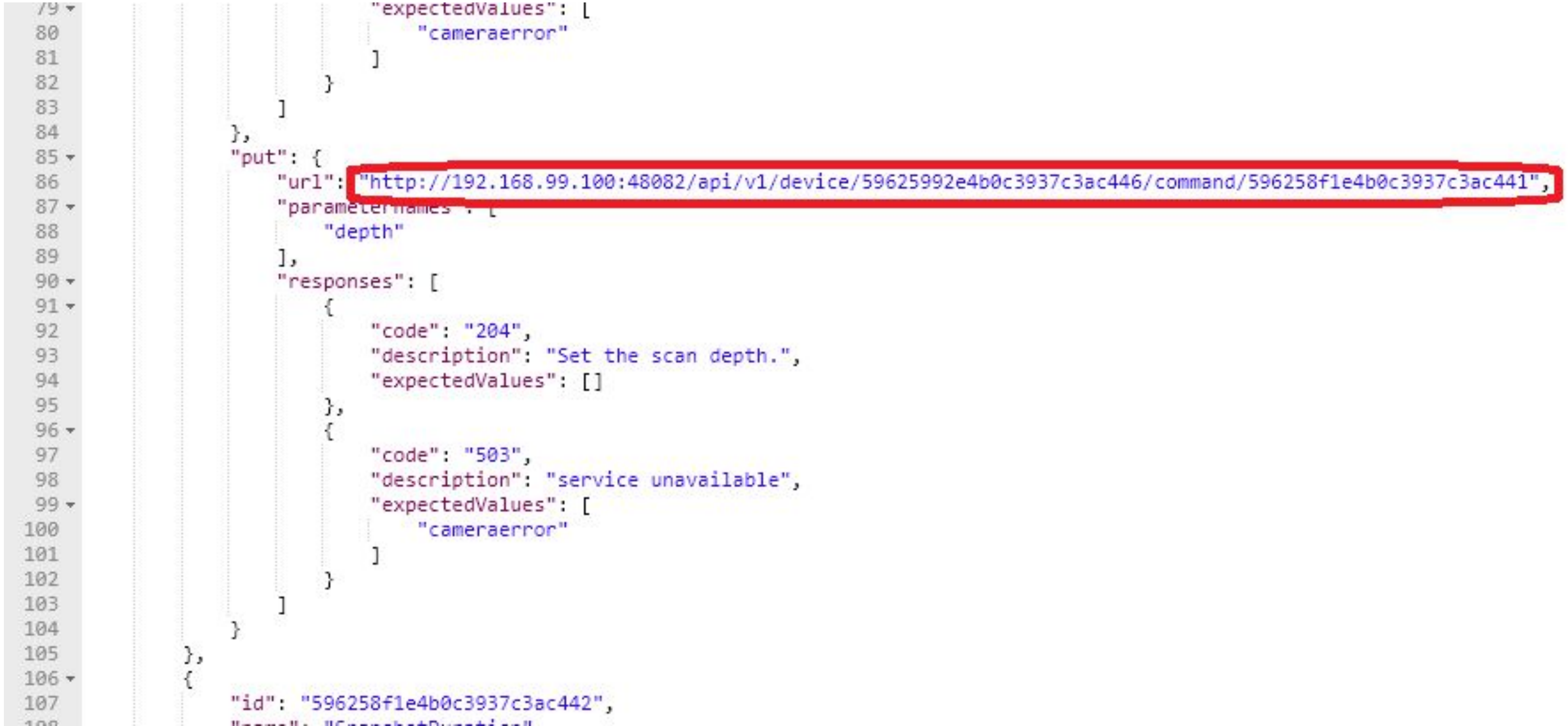
```
name: ScanDepth
get:
    ...
put:
    path: "/api/v1/devices/{deviceId}/scandepth"
    parameterNames: ["depth"]
    responses:
      -
        code: "204"
        description: "Set the scan depth."
        expectedValues: []
      -
        code: "503"
        description: "service unavailable"
        expectedValues: ["cameraerror"]
```

Implementing Device SDk Functions

```
Initialize()                // Device service start
HandleReadCommand()         // Get command called
HandleWriteCommand()        // Put command called
Stop()                      // device stopped
AddDevice()                 // device added
UpdateDevice()              // device updated
RemoveDevice()              // device removed
```

Calling device commands

GET to [http://localhost:48082/api/v1/device/name/**countcamera1**](http://localhost:48082/api/v1/device/name/countcamera1)



```
                "expectedValues": [
                    "cameraerror"
                ]
            }
        ]
    },
    "put": {
        "url": "http://192.168.99.100:48082/api/v1/device/59625992e4b0c3937c3ac446/command/596258f1e4b0c3937c3ac441",
        "parameterNames": [
            "depth"
        ],
        "responses": [
            {
                "code": "204",
                "description": "Set the scan depth.",
                "expectedValues": []
            },
            {
                "code": "503",
                "description": "service unavailable",
                "expectedValues": [
                    "cameraerror"
                ]
            }
        ]
    }
},
{
    "id": "596258f1e4b0c3937c3ac442",
```

Calling device commands

PUT to http://localhost:48082/api/v1/device/**<device id>**/command/**<command id>**

```
{
    "depth":"9"
}
```

Reading events

GET to http://localhost:48080/api/v1/**event/device/countcamera1**/10

GET to http://localhost:48080/api/v1/**reading/name/humancount**/10

Building an Application Service

- app-function-sdk: https://github.com/edgexfoundry/app-functions-sdk-go/
- Build a **function pipeline** with built-in functions or custom functions
  - Pipeline is triggered on each event generated by your device
  - Can be used to **filter** and **export** events or send a **command** to a device
  - Each function in the pipeline receives the value returned by the **previous** function

Building an Application Service

```
edgexSdk.SetFunctionsPipeline(
  transforms.NewFilter(deviceNames).FilterByDeviceName,
  transforms.NewConversion().TransformToXML,
  printXMLToConsole //Custom function
  )


func printXMLToConsole(edgexcontext *appcontext.Context, params ...interface{})
(bool, interface{}) {
     if len(params) < 1 {
          // We didn't receive a result
          return false, nil
     }

     fmt.Println(params[0].(string))

     // Leverage the built in logging service in EdgeX
     edgexcontext.LoggingClient.Debug("XML printed to console")
     edgexcontext.Complete([]byte(params[0].(string)))
     return false, nil
}
```

Sample App on Raspberry Pi

- Tutorial on how to deploy EdgeX on a RPI
- Includes:
  - Detailed instructions on how to setup RPI with **64-bit** OS
  - Custom docker-compose file with **ARM** images
  - Sample **Virtual GPS Device** to get started with gps coordinate data
- https://github.com/vmware-samples/automotive-iot-samples/tree/master/edgex_sample

EDGE X FOUNDRY™

Developing & Contributing

Install Go

Get GoLang 1.11.x:

```
wget https://dl.google.com/go/go1.11.8.linux-amd64.tar.gz

sudo tar -C /usr/local -xvf go1.11.8.linux-amd64.tar.gz
```

Setup your environment

```
cat >> ~/.bashrc << ‘EOF’
export GOPATH=$HOME/go
export PATH=/usr/local/go/bin:$PATH:$GOPATH/bin
EOF

source ~/.bashrc
```

Install MongoDB

- sudo apt install mongodb-server
- systemctl status mongodb
- wget https://github.com/edgexfoundry/docker-edgex-mongo/raw/master/init_mongo.js
- sudo -u mongodb mongo < init_mongo.js

Get the EdgeX source code

- go get **github.com/edgexfoundry/edgex-go**
- cd ~/go/src/github.com/edgexfoundry/edgex-go
- sudo apt install libczmq-dev
- make build
- make run

- cd ./docs
- ./build.sh

Setup your git repository

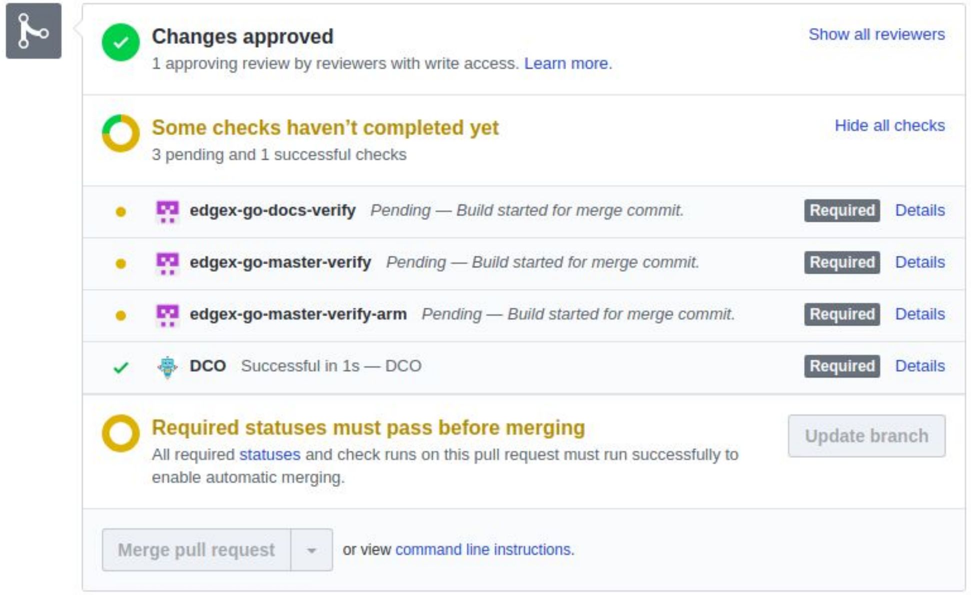
- Fork https://github.com/edgexfoundry/edgex-go
- git remote add **mygithub** https://github.com/**<your_username>**/edgex-go.git
- git config **--global.user.name** “John Doe”
- git config **--global.user.email** johndoe@example.com

Contributing changes

- git checkout **-b your_fix_branch_name**
- git add <files you changed>
- git commit **--signoff** -m “Your commit message”
- git push **mygithub your_fix_branch_name**

PR review and approval

- Pass DCO Signoff
- Pass automated tests
- Have at least one approving review



Changes approved
1 approving review by reviewers with write access. Learn more.

Some checks haven't completed yet
3 pending and 1 successful checks

● **edgex-go-docs-verify** *Pending — Build started for merge commit.*	Required	Details
● **edgex-go-master-verify** *Pending — Build started for merge commit.*	Required	Details
● **edgex-go-master-verify-arm** *Pending — Build started for merge commit.*	Required	Details
✓ **DCO** Successful in 1s — DCO	Required	Details

Required statuses must pass before merging
All required statuses and check runs on this pull request must run successfully to enable automatic merging.

or view command line instructions.