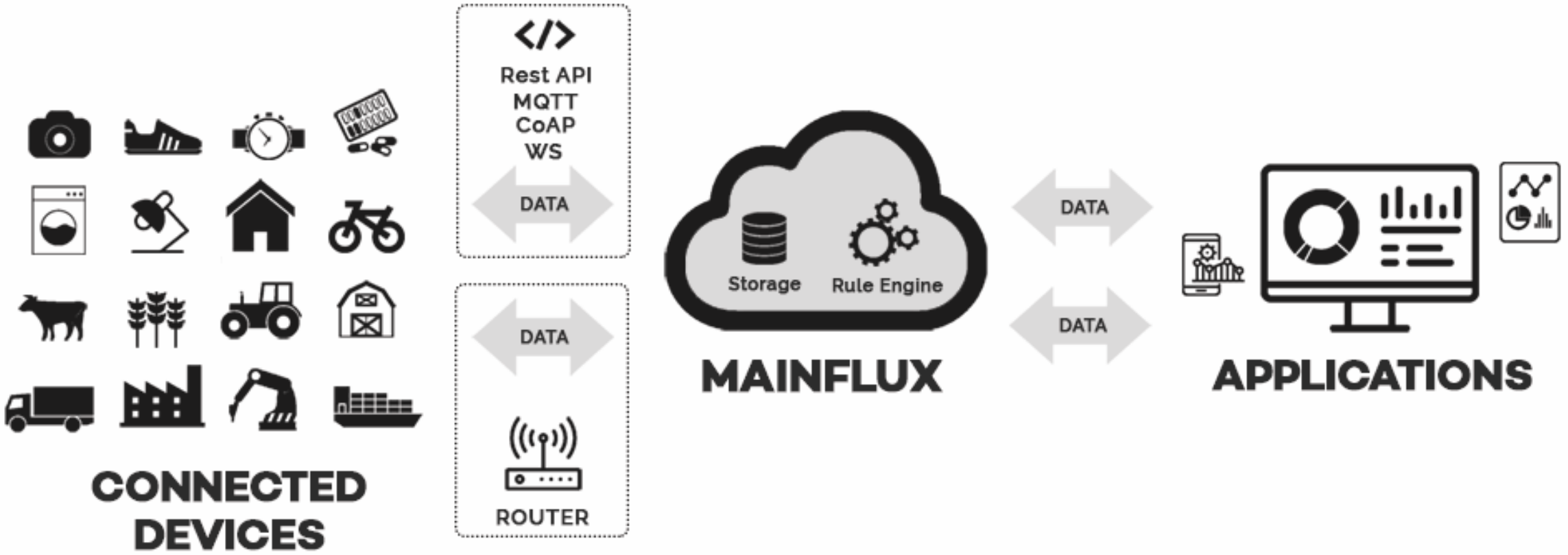
EDGE X FOUNDRY™

Technical Workshop
Mainflux Open Source IoT Platform

Day 2: June 2, 2017

Mainflux – Platform Overview



Mainflux – Development Philosophy

Typical Smart Device Technology Stack

- User App
- Application Management
- User Management
- Access Control
- Device Message Broker
- Data Storage
- Device Provisioning
- Network Security
- Multi Protocol Connectivity
- Smart Device

Mainflux – Smart Device Technology Stack

- User App
- Mainflux – Open Source IoT Platform
  - OAuth 2.0 Identity Provider
  - User Management
  - Policy Based Access Control
  - Device Message Broker
  - Data Storage
  - Device Provisioning
  - Network Security
  - Multi Protocol Connectivity (HTTP, WS, MQTT, CoAP)
- Smart Device

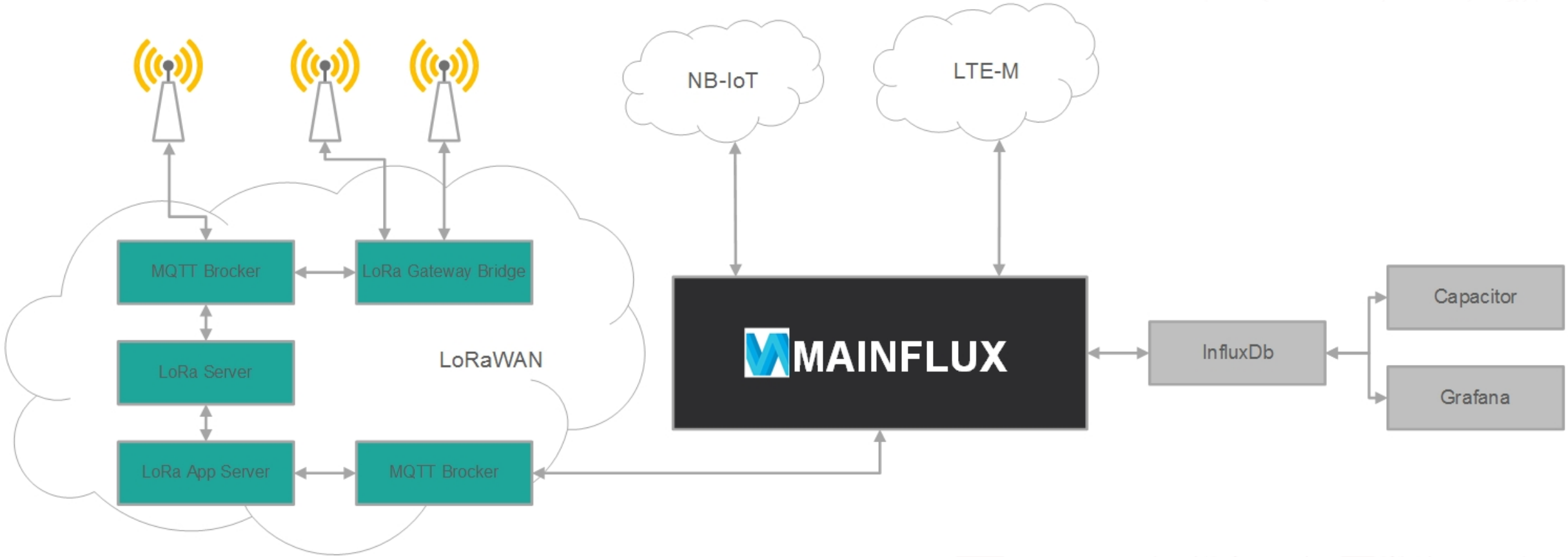
Benefits with Mainflux IoT Platform

- User App
- MAINFLUX

```
>> git clone https://github.com/Mainflux/mainflux.git && cd mainflux
>> docker-compose up
```

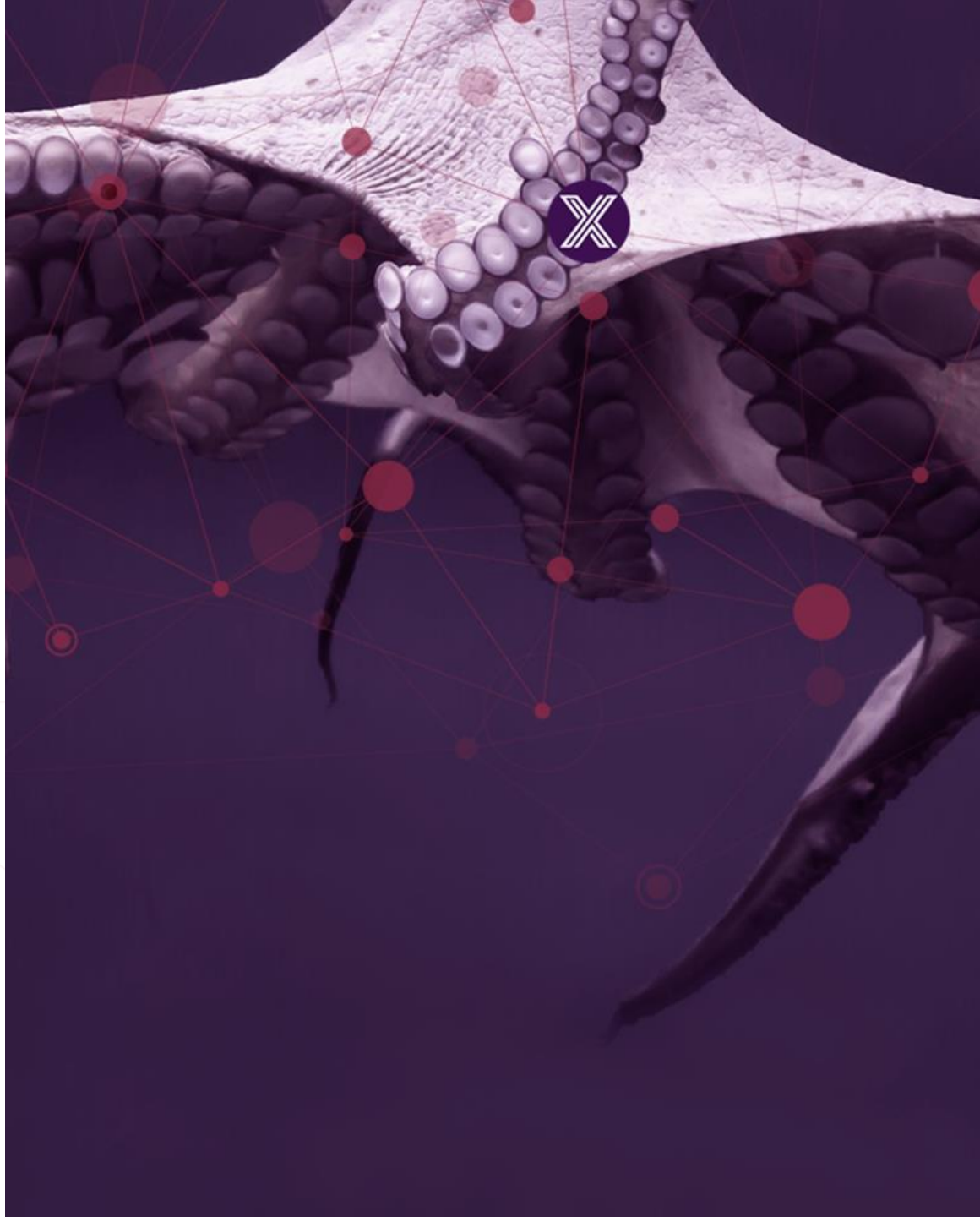
- Smart Device

Mainflux – LoRa End 2 End Solution

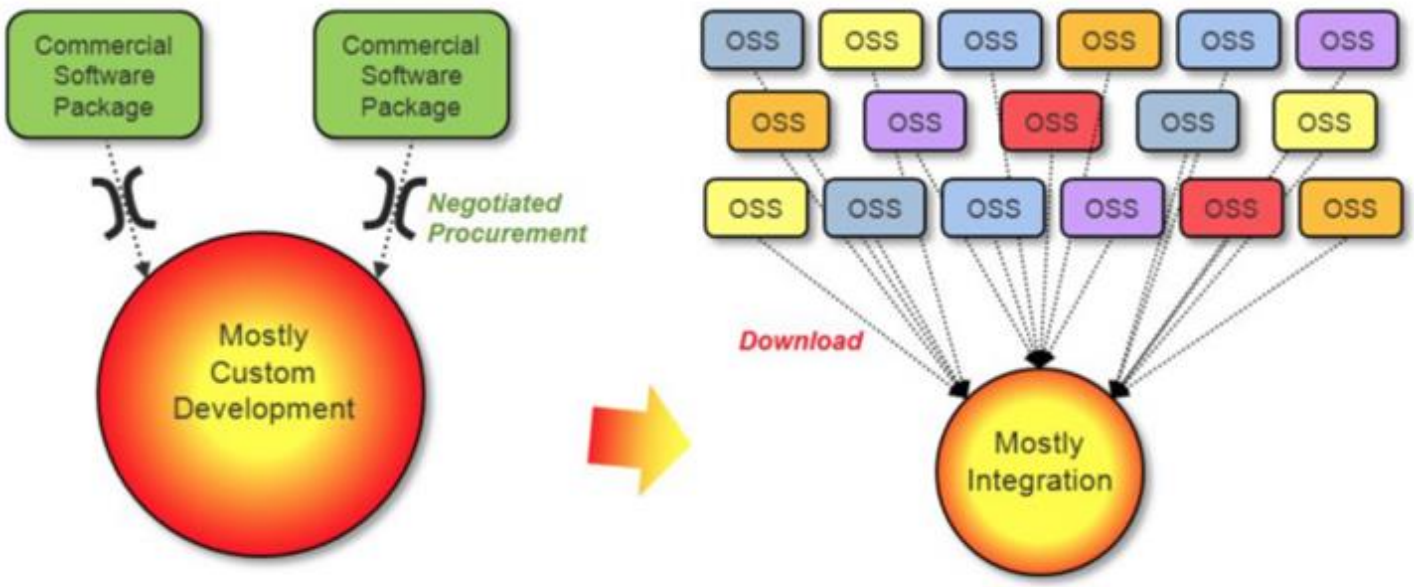


EDGE X FOUNDRY™

Mainflux – System Architecture

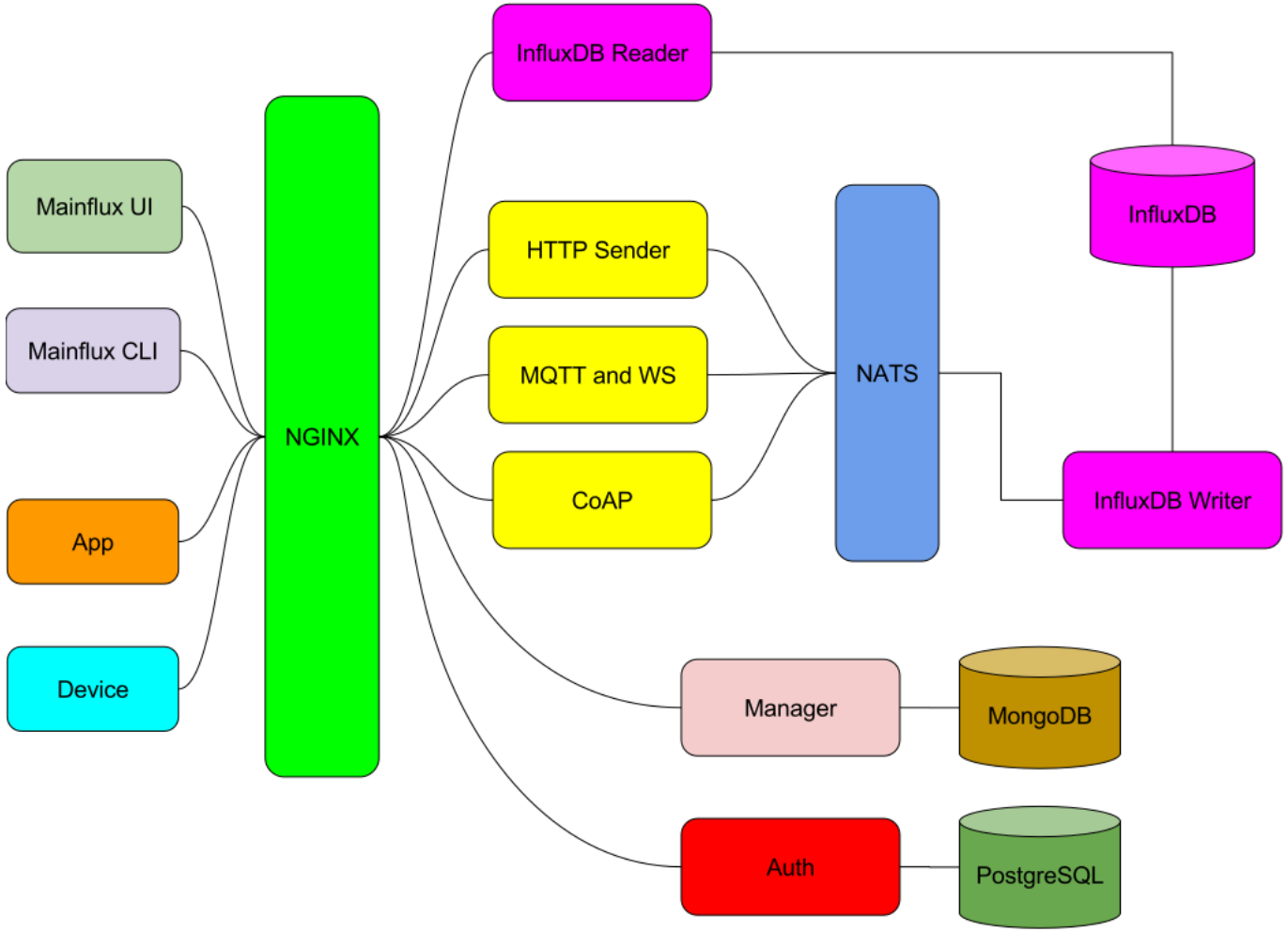


Mainflux Embraces OSS Model



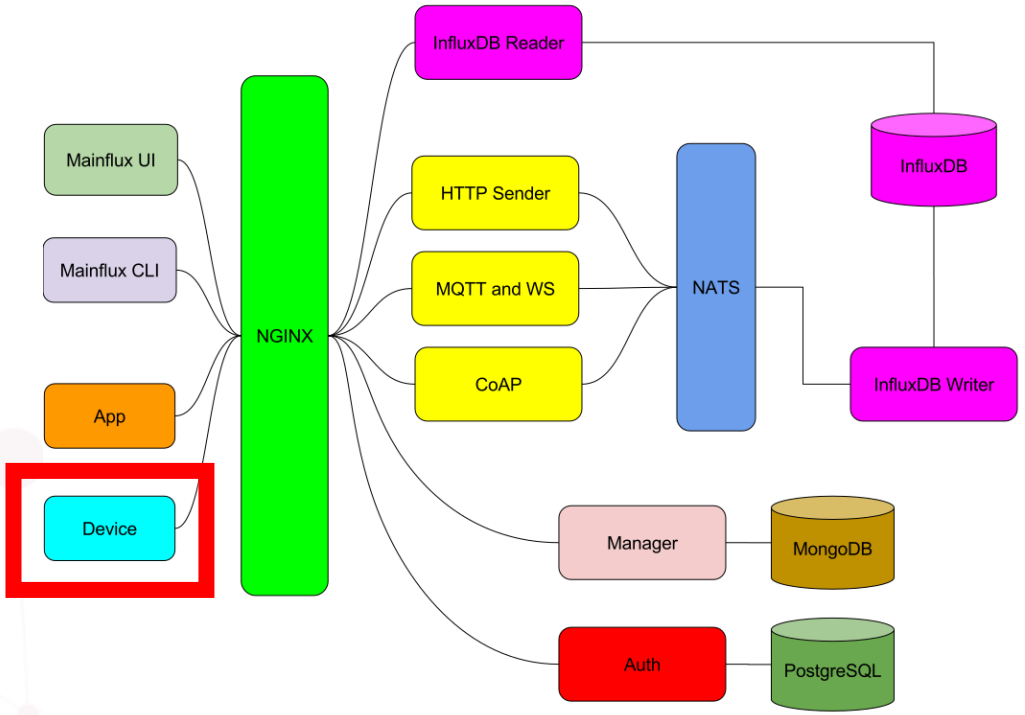
“In the past, software products were largely developed in-house. Now we are repeatedly downloading code from the Internet to evaluate, prototype, and integrate. We only code the parts that are truly unique to our application.” - Linux Foundation

Mainflux – System Architecture



Devices

- MCU + RF front-end
- Bare-metal or small RTOS (for HAL and drivers)
- Constrained CPU clocking (low on kHz)
- Constrained flash and RAM
- Battery powered devices

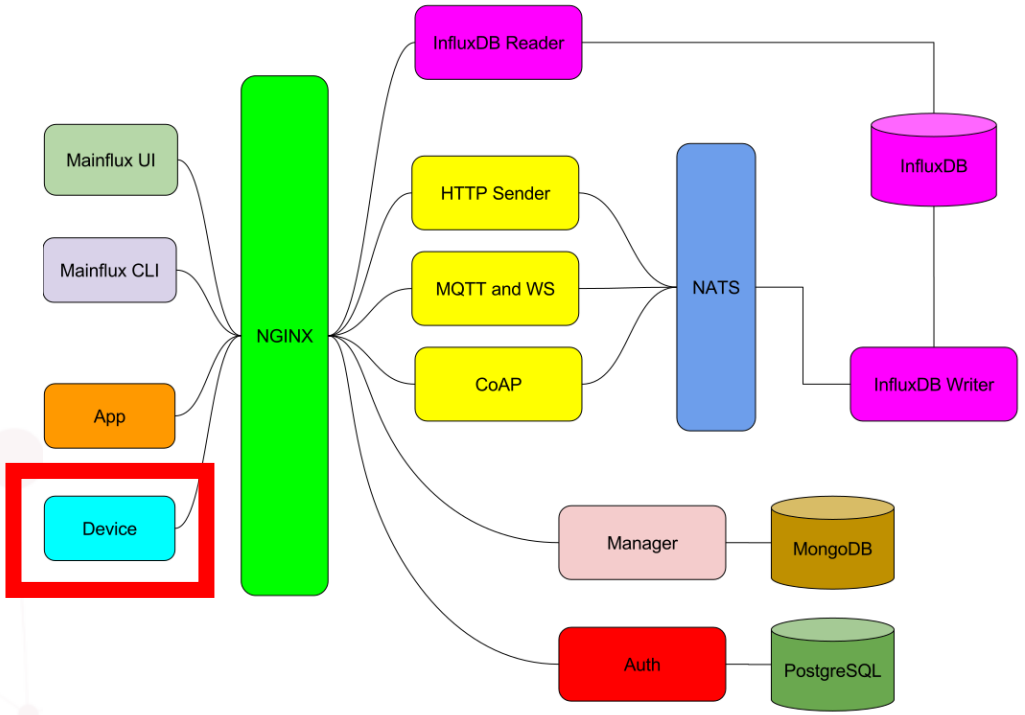



Heavy networking stack just won't fit!


And every radio TX/RX is power hungry!

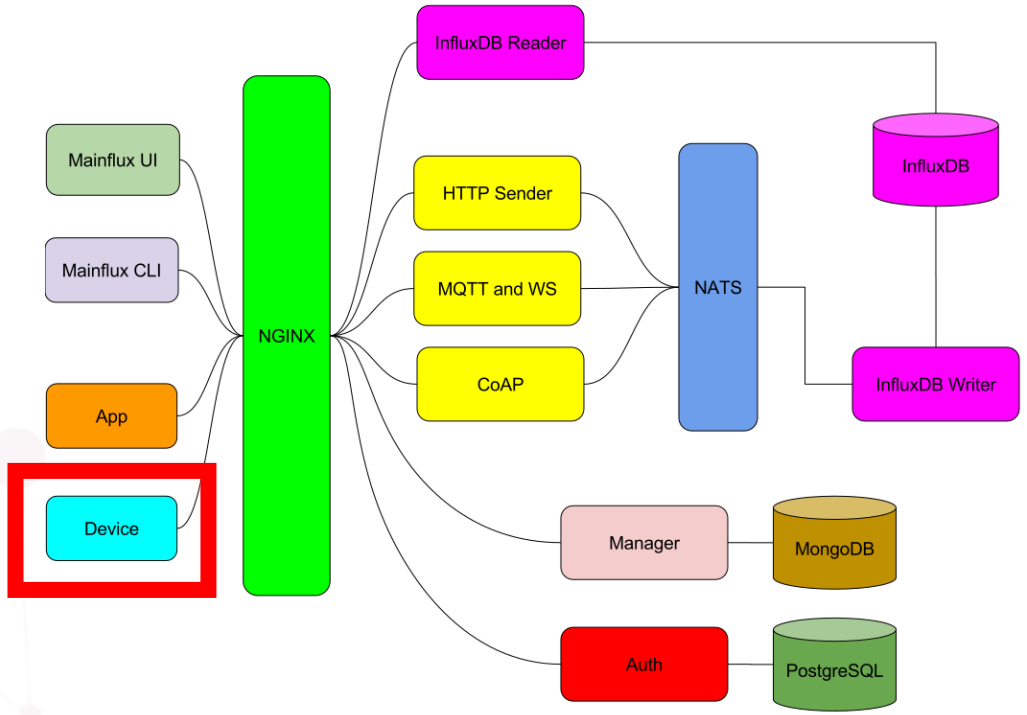
Devices – IoT Node Mode of Operation

- Sleep most of the time - wake up just to send telemetry, then go to sleep again
- Keep the messages short to save power (less protocol overhead, less metadata in the header, etc...)



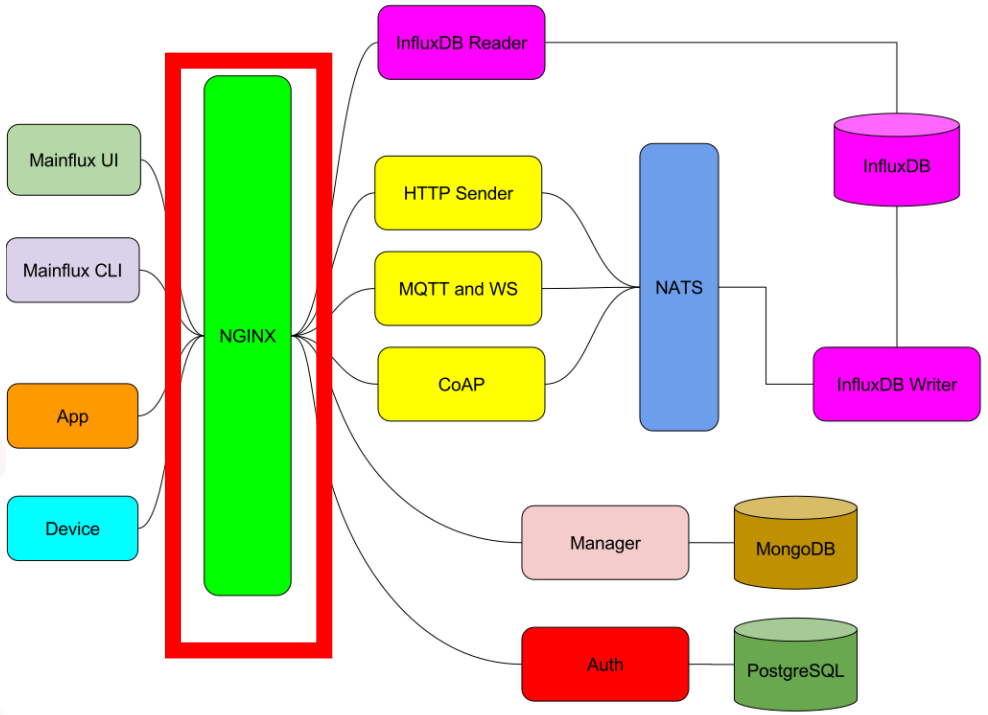
Devices – Mainflux – Supported Protocols

- Devices
  - MQTT
  - CoAP
- Apps
  - HTTP
  - WS



NginX – Reverse Proxy

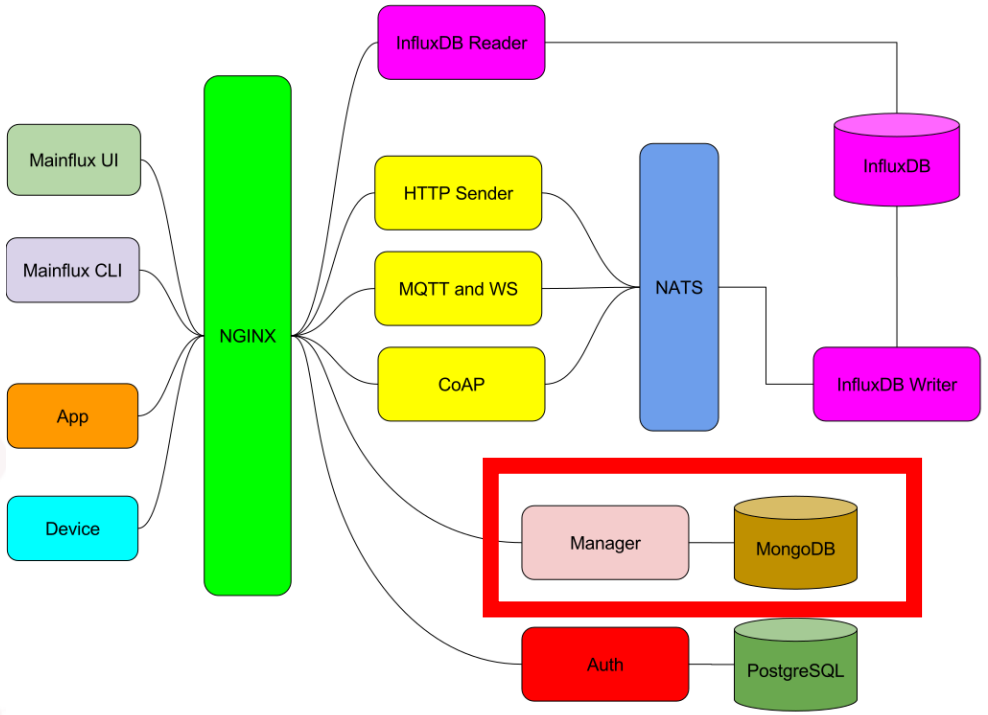
- Load Balance - scaling, HA, Fault Tolerance.
- SSL termination - keep these certs at one place. MQTT is pure TCP, and CoAP is UDP.
- PEP (Policy Enforcement Point) Proxy - forward the req for auth to Auth server (Policy Decision Point).




Intercept the requests and send them for Auth

Mainflux – Device Management

- Mainflux Platform is multi-user, multi-device, multi-app.
- Management subsystem exposes RESTful API for system entities provisioning:
  - Users
  - Devices
  - Channels
  - Organizations
  - Applications



Mainflux – Provision Device Model

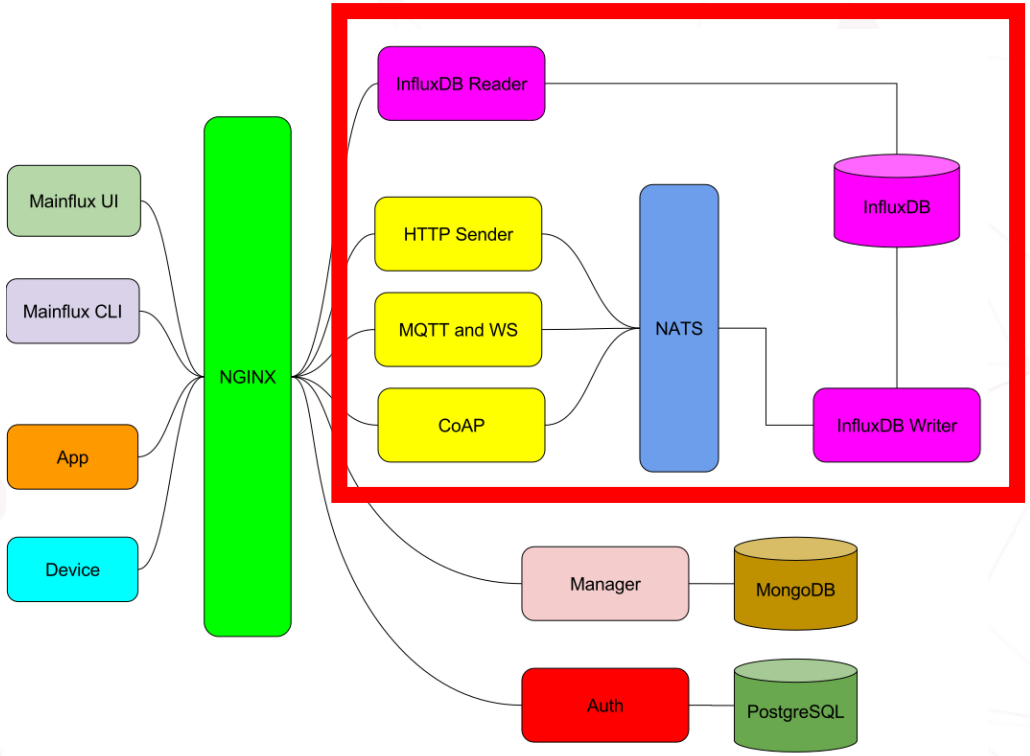
```
curl -s -S -i -X POST -H "Accept: application/json" \
         -H "Content-Type: application/json" http://localhost:7070/devices

HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Nov 2016 21:49:26 GMT
Content-Length: 69

{
  "response": "created",
  "id": "e35b157f-21b8-4adb-ab59-9df21461c815"
}
```

Mainflux – Provision Channel

```
curl -s -S -X POST -H "Accept: application/json" \
         -H "Content-Type: application/json" http://localhost:7070/channels

{
  "response": "created",
  "id": "5c912c4e-e37b-4ba6-8f4b-373c7ecfeaa9"
}
```

Mainflux – Device Messaging Subsystem

- Messaging subsystem is composed of following microservices:
  - HTTP Message Sender
  - MQTT (and WS) Broker
  - CoAP Server
  - NATS Broker
  - InfluxDB Writer
  - InfluxDB Reader
- It's role is to distribute messages between various clients that can connect via various protocols - i.e. it makes a messaging bridge between them.



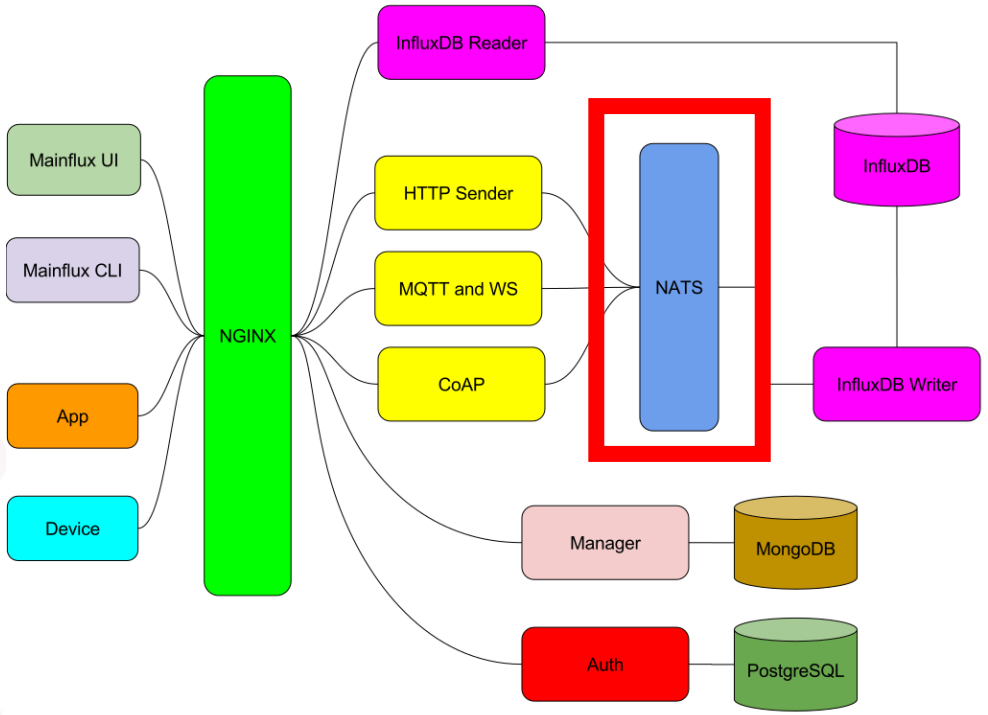

HTTP-to-WS-to-MQTT-to-CoAP?


Yes. Mainflux is bridging protocols.
GIFsFORUM.com

NATS

- Brokers the IoT messages as an events in Mainflux NATS Message Format.
- Mainflux Message Format:

```
message MM {
        string channel;
        string protocol;
        string publisher;
        string content_type;
        Bytes payload;
}
```

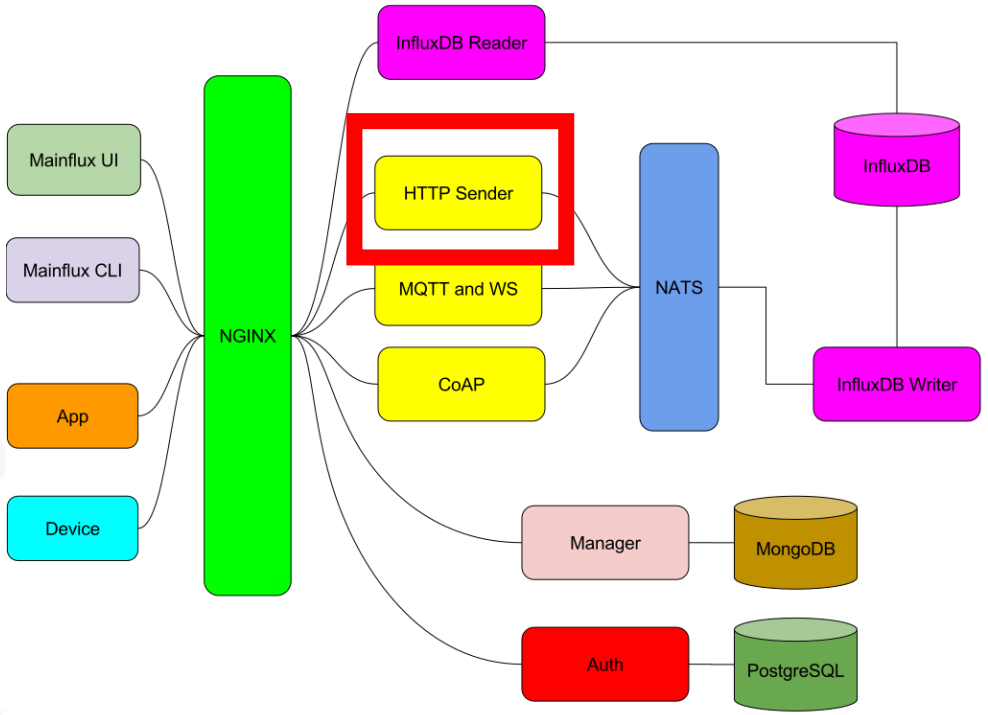


HTTP

HTTP Message Sender is an HTTP Server that exposes RESTful API for sending (and sending only, not receiving) IoT messages received from HTTP clients (devices and applications).

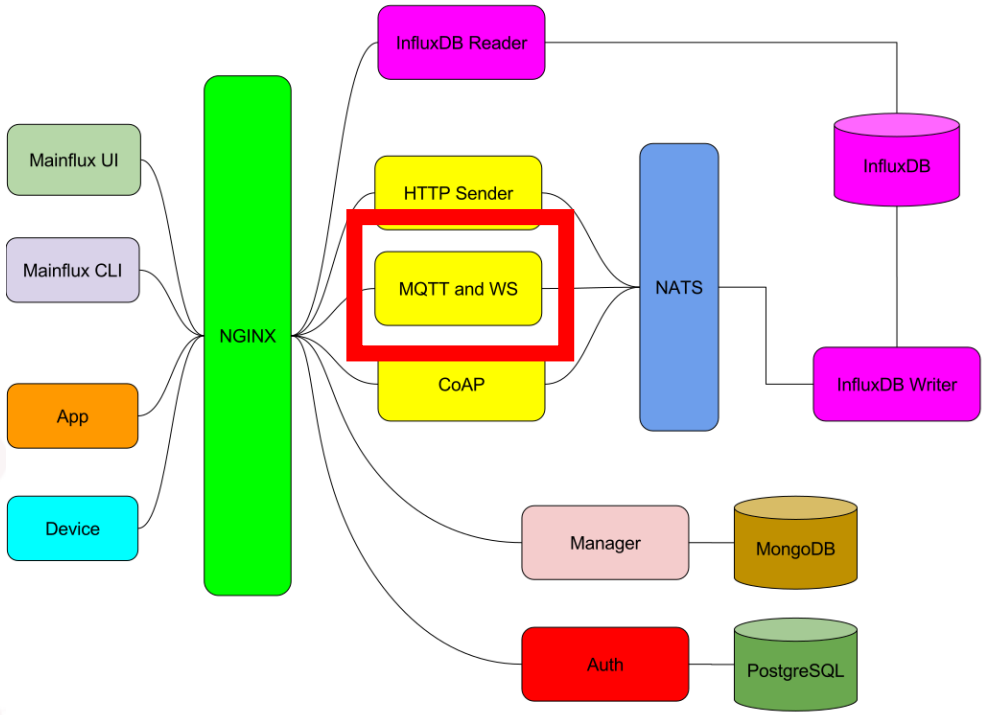
- Message is sent in the form of SenML.
- End Point:

/channels/<channel_id>/messages



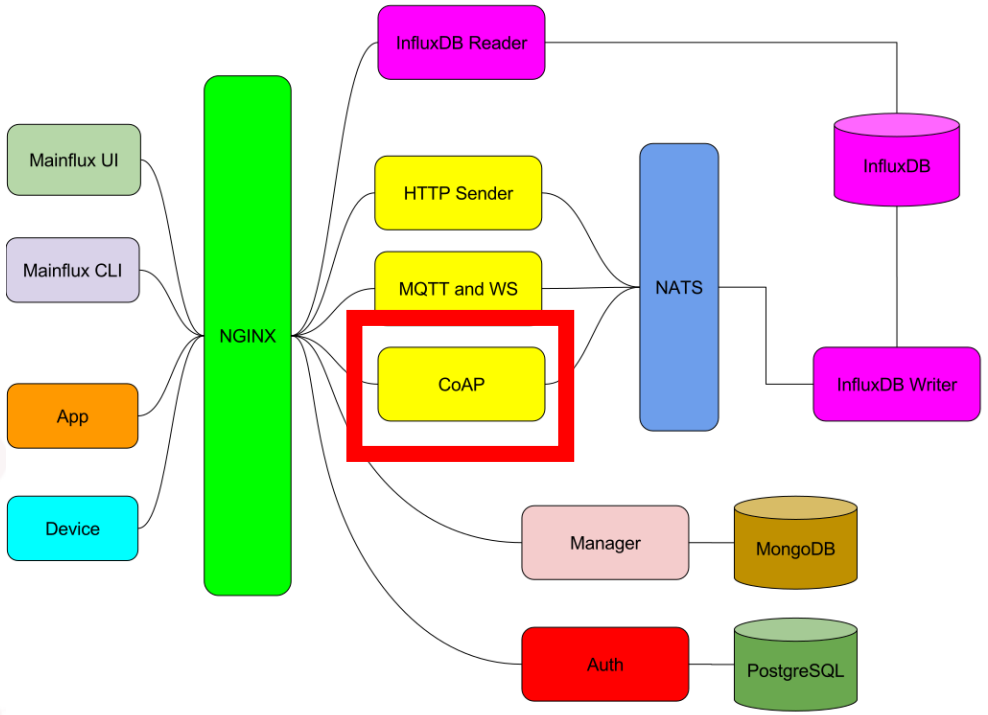
MQTT

- MQTT broker accepts and publishes SenML and binary messages to MQTT clients and at the same time on NATS broker for database persistence and analytics.
- SenML JSON messages are published on channels/<channel_id>/messages/senml-json.
- Binary messages are published on channels/<channel_id>/messages/octet-stream.



CoAP

- CoAP server accepts CoAP (UDP) connections
- RESTful-like API for sending IoT messages received from CoAP clients (devices and applications)
- RESTful-like API for CoAP-observing (similar to MQTT subscribing) of message channels.
- SenML message format.



SenML – Model for IoT Messages

- IETF standard (https://tools.ietf.org/html/draft-ietf-core-senml-05)
- Provides simple model for retrieving data from sensors and controlling actuators
- Provides minimal semantics for the data inline and allows for more metadata with in-line extensions and links

SenML – Message Format

```
[
   {"bn":"urn:dev:ow:10e2073a0108006;","bt":1.276020076001e+09,
    "bu":"A","bver":5,
    "n":"voltage","u":"V","v":120.1},
   {"n":"current","t":-5,"v":1.2},
   {"n":"current","t":-4,"v":1.3},
   {"n":"current","t":-3,"v":1.4},
   {"n":"current","t":-2,"v":1.5},
   {"n":"current","t":-1,"v":1.6},
   {"n":"current","v":1.7}
]
```

Binary Message Format

- Message is sent in the form of binary blob.
- Mainflux doesn’t parse the binary message.
- Binary message is stored in the database.
- 3rd Party App would have to parse the message and use it.

Reporting Values (aka. Send Message)

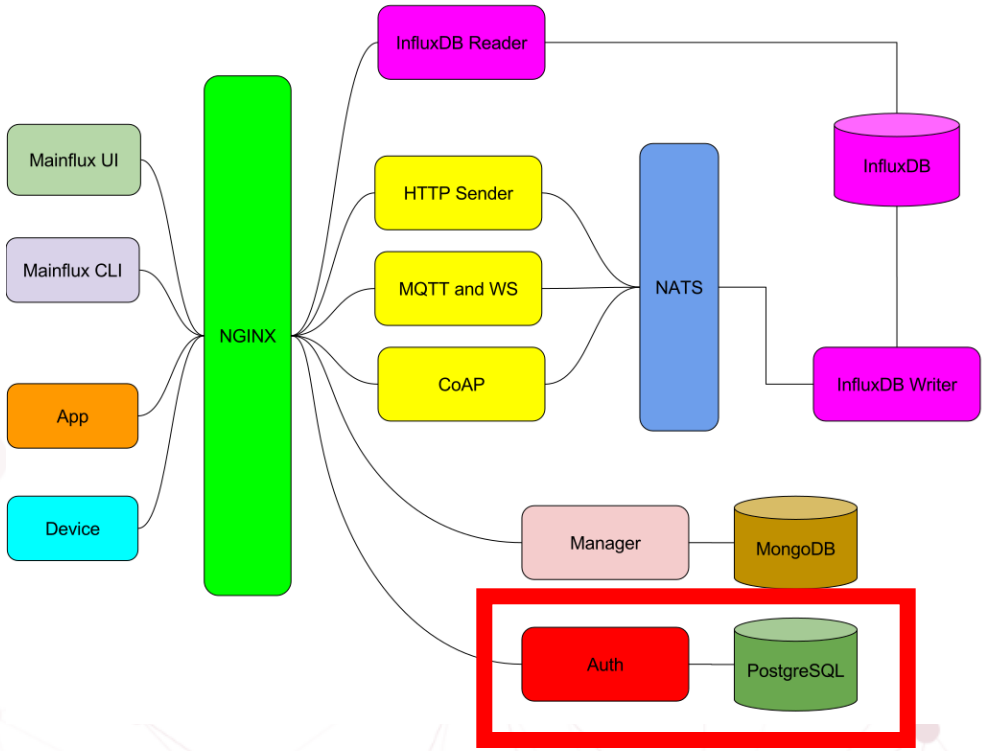
```
curl -s -S -i -X POST -H "Accept: application/json" \
         -H "Content-Type: application/json" \
         http://localhost:7070/channels/78c95058-7ef3-454f-9f60-82569ddec4e2/msg \
         -d '[{"bn":"some-base-name:","bt":1.276020076001e+09, "bu":"A","bver":5,
                "n":"voltage","u":"V","v":120.1}, {"n":"current","t":-5,"v":1.2},
                {"n":"current","t":-4,"v":1.3}]'

HTTP/1.1 202 Accepted
Content-Type: application/json; charset=utf-8
Date: Sun, 18 Dec 2016 18:25:36 GMT
Content-Length: 28

{
  "response": "message sent"
}
```

MQTT PUB

```
mosquitto_pub -i 472dceec-9bc2-4cd4-9f16-bf3b8d1d3c52 \
              -t mainflux/channels/5c912c4e-e37b-4ba6-8f4b-373c7ecfeaa9 \
              -m '[{"bn":"e35b157f-21b8-4adb-ab59-9df21461c815",
                          "bt":1.276020076001e+09, "bu":"A","bver":5,
                          "n":"voltage","u":"V","v":120.1},
                        {"n":"current","t":-5,"v":1.2},
                        {"n":"current","t":-4,"v":1.3}]'
```

Mainflux – Auth Subsystem

- Policy Based Auth.
- Supports OAuth2.0 for Apps.
- JWT and Certs for Devices.
- Acts as Identity Provider.
- Based on Hydra and Ladon.



Mainflux – Device Auth Constraints

- UDP is more lightweight than TCP
- TLS becomes a problem (DTLS for UDP, but implementation missing in many languages)
- Elliptic Curve Cryptography - Diffie-Hellman
- HW encryption engine helps

Mainflux – Device Auth – AuthX

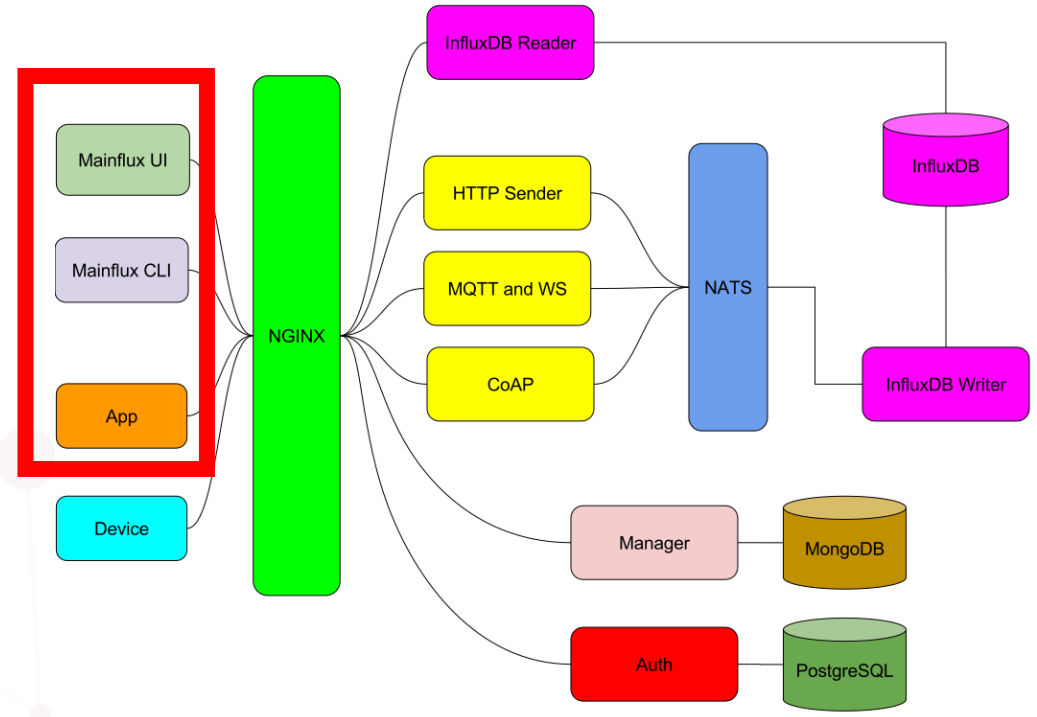
- CA must be burned into device flash for server auth for TLS
- Client-side certificates - TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- JWT
- OAuth2.0 Client Secret - how do we issue temporary token to devices?

Mainflux – Device Auth – AuthZ

- Scoped API keys - JWT
- Drawback - no revoke, and device JWT has infinite TTL
- Revoke destroys stateless approach - so use just for session tokens with low TTL and no revoke
- Mainflux uses Access Control Policies - inspired by AWS IAM Policies

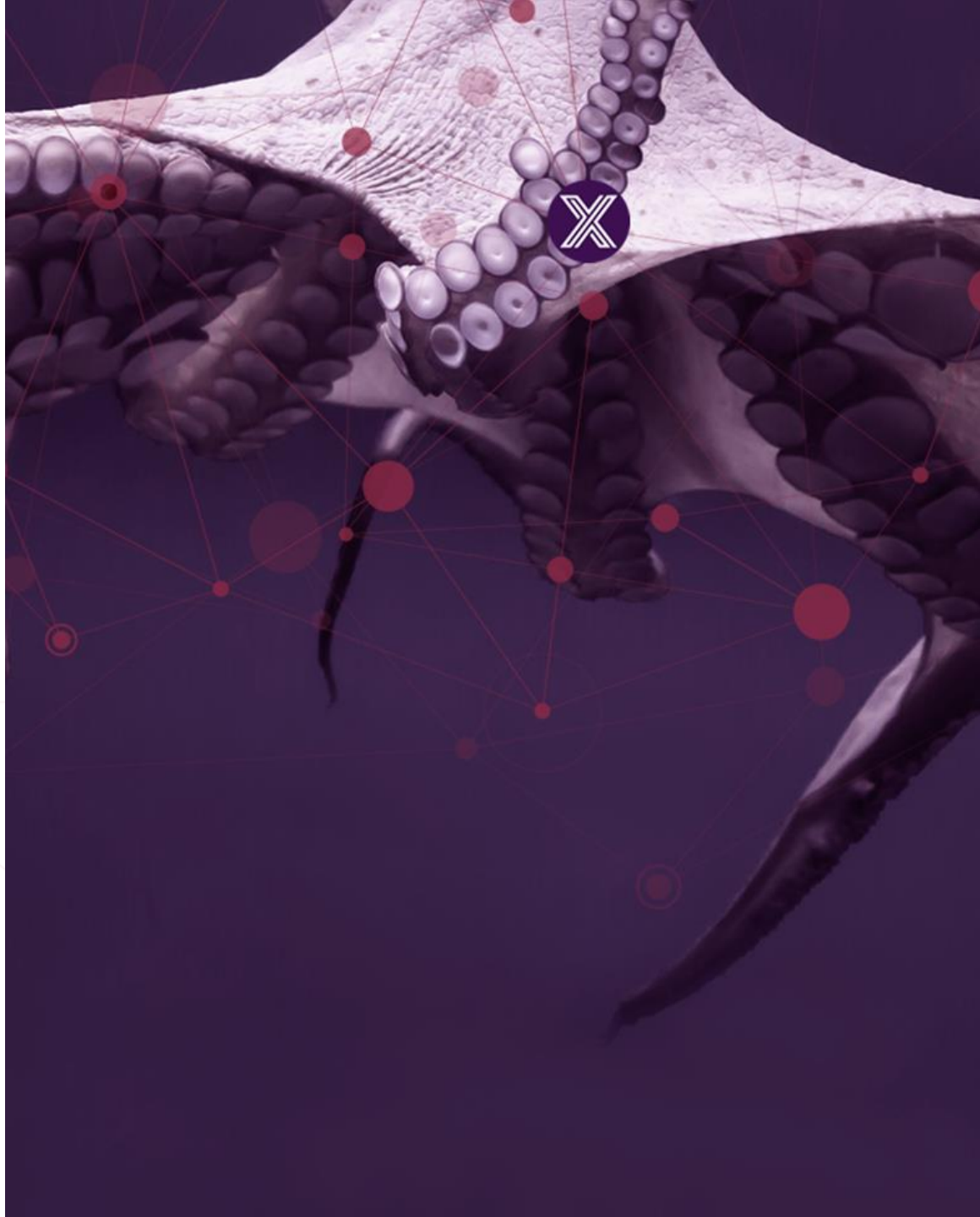
Applications

- Mainflux CLI
- Mainflux UI (WIP)
- 3rd Party Apps



EDGE X FOUNDRY™

Mainflux – Deployment



Mainflux – Install / Deploy Docker Images

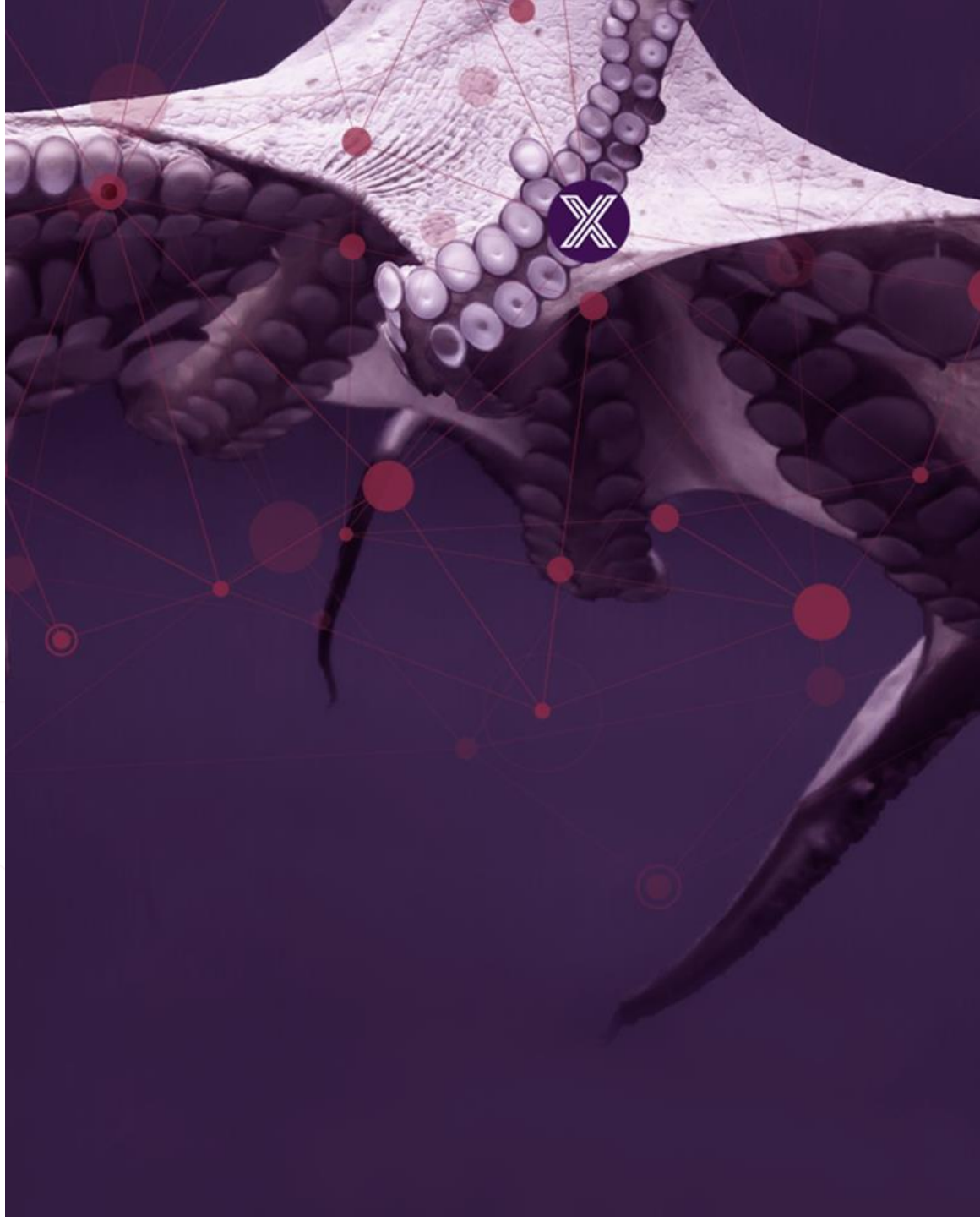
- Clone the repo:
  - git clone https://github.com/Mainflux/mainflux.git && cd mainflux
- Start the Docker composition:
  - docker-compose up
- This will automatically download Docker images from Mainflux Docker Hub and deploy the composition of Mainflux microservices.

Mainflux – Compile the Source Code

- Go compiles to static binary
- Can be compiled to run on:
  - Linux
  - Windows
  - Mac
  - ARM devices
- Can be deployed on RPi or similar systems.

EDGE X FOUNDRY™

Mainflux – Future Plans

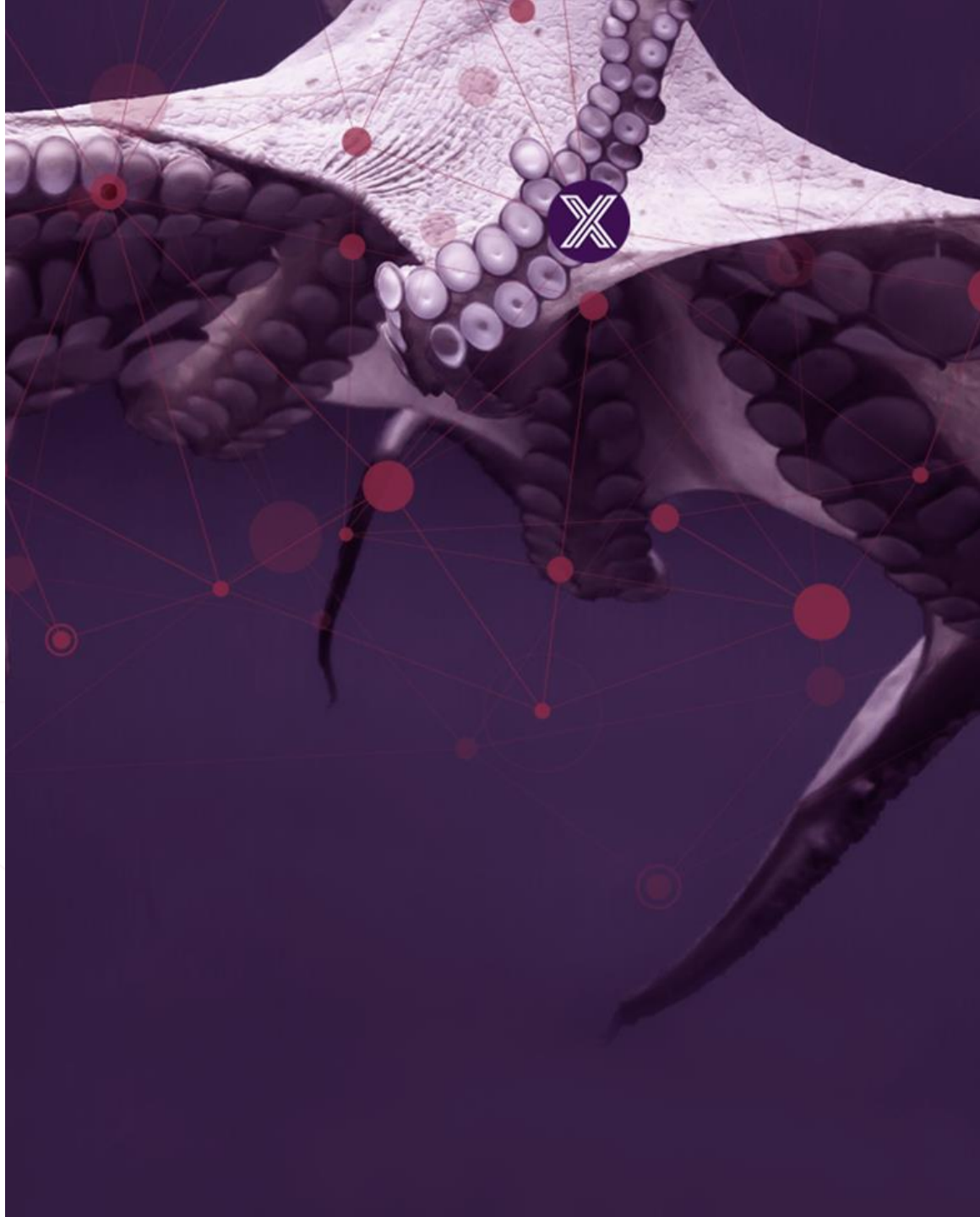


Mainflux – Future Plans

- Evaluate Mainflux as EdgeX Gateway Manager.
- Evaluate Mainflux as Device Manager for devices connected to EdgeX.
- Evaluate Mainflux as Security Manager for EdgeX.
- Push Sensor Data to Hyperledger.
- Add Machine Learning and Data Analysis Layer on top of Mainflux.

EDGE X FOUNDRY™

Mainflux – The Team



MAINFLUX TEAM

Mainflux team has an extensive working experience and knowledge of hardware and intelligent devices and can provide operational expertise for the best cost-quality related solutions and support needed for every IoT project. Our expertise covers every functional aspect of the networking technology required for connecting objects, including its huge market and a large number of manufacturers.

DRASKO DRASKOVIC

MAINFLUX PROJECT CO-FOUNDER
M.Sc. Electronics

Draško has over 15 years of professional experience gained in fortune 500 companies as well as technological start-ups. He worked in telecom and semiconductor giants like NOKIA, Alcatel-Lucent, Texas Instruments and PHILIPS - being engaged in embedded systems, semiconductor, and telecommunication technologies.

NIKOLA MARCETIC

SOFTWARE DEVELOPMENT
M.Sc. Economy

Nikola has experience of more than eight years, covering a wide range of technologies and IT directions, from IT administration over computer networks and security, system architecture to software development and testing.

JANKO ISIDOROVIC

MAINFLUX PROJECT CO-FOUNDER
M.Sc. Telecommunications

Janko gained comprehensive work experience in NELT, South Europe's biggest logistic and distribution company (P&G, Kraft Foods, Wrigley) as Project Manager, System Architect and Application & System Engineer.

He has been managing highly technical projects as company acquisition, MVNO enablement, OTT services setup.

SASA KLOPANOVIC

BUSINESS DEVELOPMENT
M.Sc. In Philosophy

Recently engaged in the biggest real estate project in South-East Europe master-planned by world renowned architects, Saša gained significant working experience in communication with international companies, eminent consultancies, and government institutions.

MAINFLUX TEAM

MANUEL IMPERIALE

HW & EMBEDDED SW DEVELOPMENT
M.Sc. In Industrial Computing and Robotics

A University Pierre and Marie Curie graduate, Manuel has specialized in industrial informatics and both software and hardware technologies. He was working in The Institute for Intelligent Systems and Robotics (ISIR), and companies , 3D Sound Labs and Devialet. In Devialet he was working on the wireless sound system which has the longest positive review in the history of magazine WIRED.

DARKO DRASKOVIC

SOFTWARE DEVELOPMENT
M.Sc. In Philosophy

Darko is software developer. His special fields of interest include web development, graphics programming and data science. He made several frameworks for graphics application development in javascript. Darko is fluent in C, Python, Lua and Go.
Darko is also philosopher with a special interest in contemporary science and cutting edge IT technology. Currently, he is finishing his PhD thesis on AI at UNIL, Switzerland.

DEJAN MIJIC

SOFTWARE DEVELOPMENT
M.Sc. In Computer Science

Dejan is software engineer with significant experience in developing large-scale distributed enterprise systems. Experienced in designing, implementing and delivering networked systems. He was 4 years Teaching Assistant at Faculty of Technical Sciences, University of NS and has working experience in various companies: SmartCat.io Expro I.T. Consulting, NTsystems and Typhoon HIL, Inc.
Dejan is certified in Machine Learning, Parallel programming and Scala Programing.