ARCHITECTURING AND SECURING IOT PLATFORMS

JANKO ISIDOROVIC
CEO @ MAINFLUX

IT NEXT SUMMIT

Outline

Internet of Things (IoT)

- Common IoT Project challenges
  - Networking
  - Power Consumption
  - Computing Power
  - Scalability
  - Security

IoT Server Architecture

- Protocol Choices
- Centralized vs Decentralized
- Typical Architecture

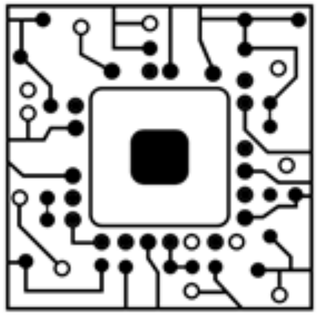
Mainflux IoT Platform – Lessons Learned

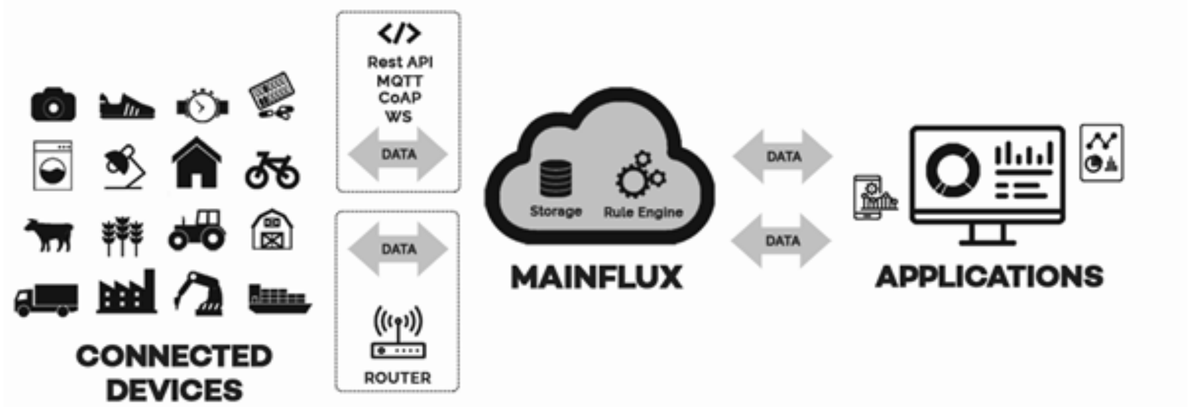
- Architecture
- Security
- Technology Choices

What is Internet of Things

The Internet of Things is the network of dedicated physical objects (things) that contain embedded technology to sense or interact with their internal state or external environment.

The IoT comprises an ecosystem that includes things, communications, applications and data analysis.





Internet of Things - IoT > M2M

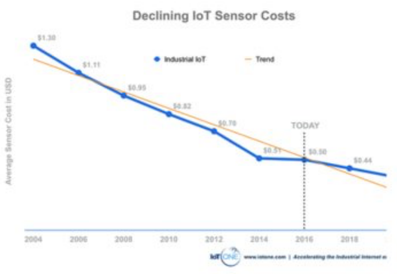
Connectivity and embedded technology

The cost of connectivity and embedded technology is becoming less of a barrier to adoption.

Broadband communication, Wi-Fi, Near Field Communication (NFC), Bluetooth and mobile networks are becoming ubiquitous and able to support large volumes of IoT connectivity at little incremental cost.

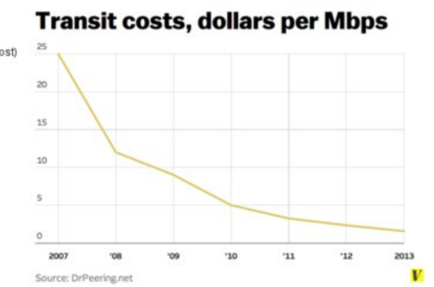
- Low Cost Sensors
- Long Range - Low Power Radio Networks
- Bitcoin like Micro transactions

Sensor Cost



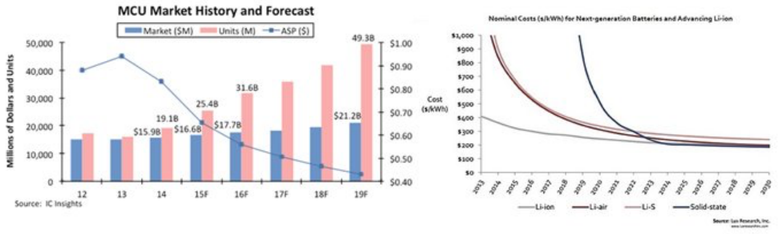
Computing Cost

Networking Cost



Microcontroller Cost

Battery Cost

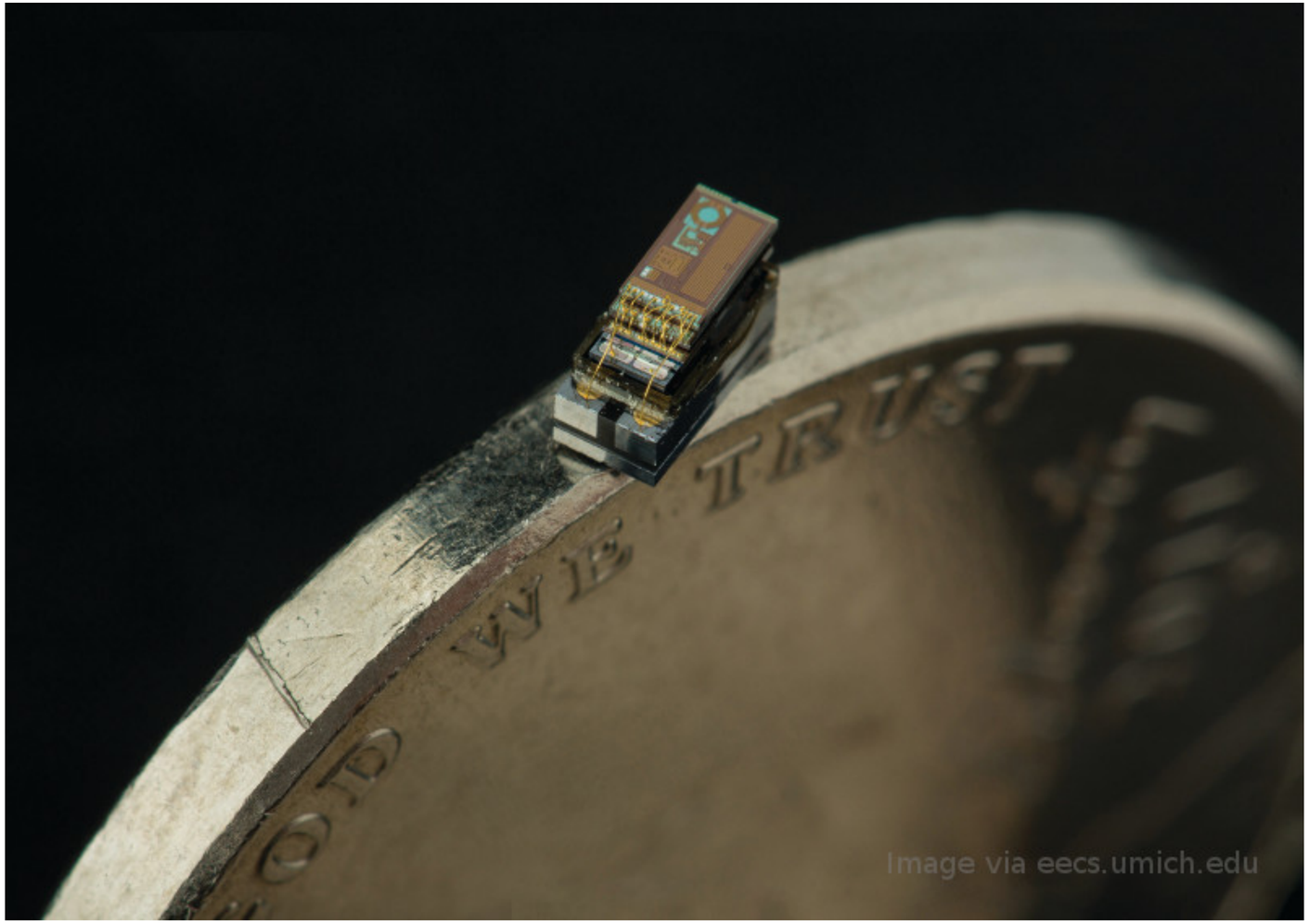


DESIGNING THE IOT SYSTEM

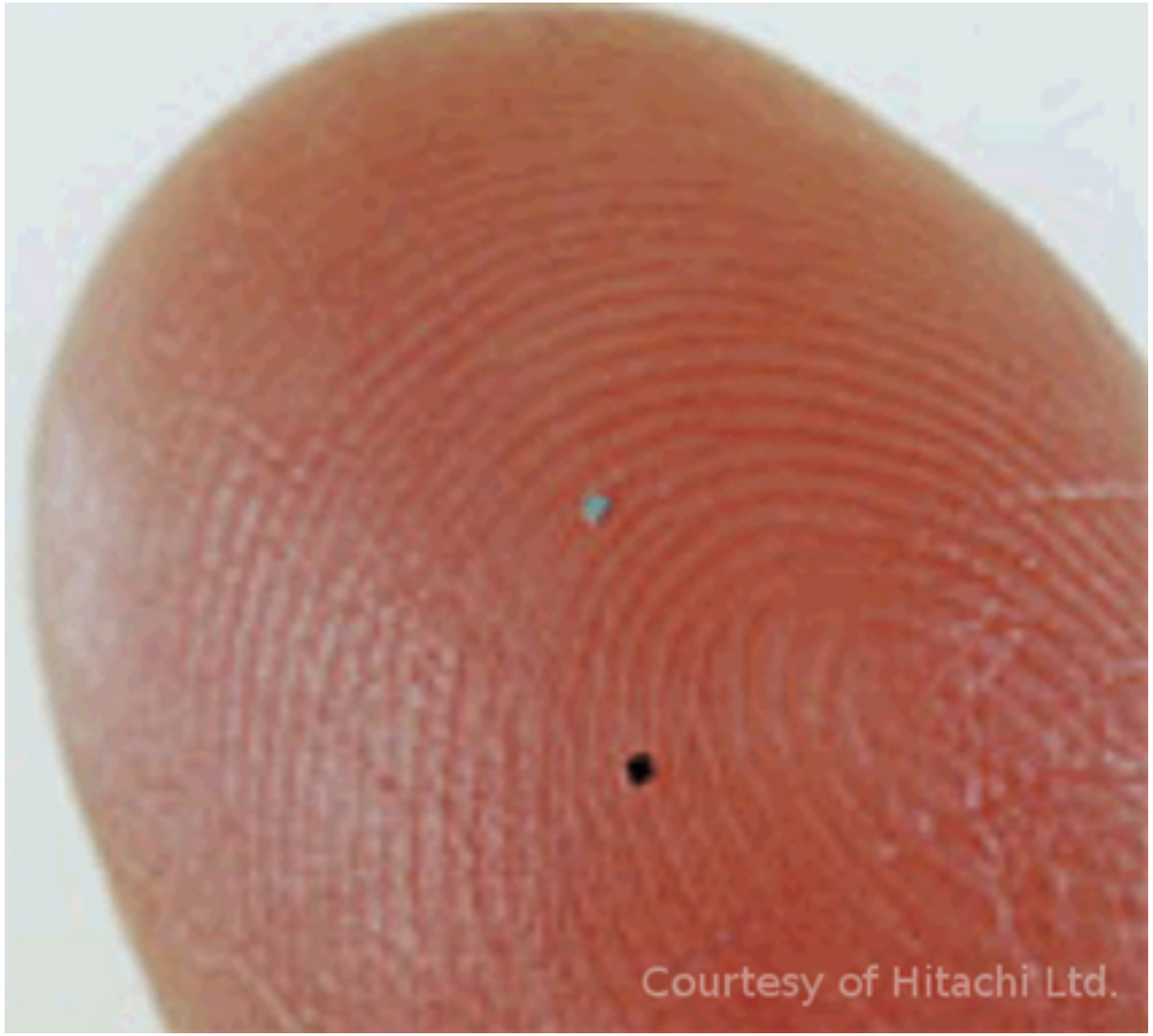

… which means we need to talk about electronics, firmware, RF, network protocols, server architecture, devops, security…

IOT DEVICE NODES

We are talking about something like this:



Or even this:



Common IoT Challenges - Networking

No one size fits all solution

Wireless Network Frequency

- Sub-GHz Solutions
- LoRaWAN
- Sigfox
- 2.4GHz Solutions
- NB-IoT, LTE-M

Wireless Network Range

- Long range
- Short range

Network Latency

- Cloud Systems ~ 200ms
- On Premise Systems ~ 50ms
- Fog Systems ~ sub 5ms

Wireless Network Bandwidth

- High Bandwidth
- Low Bandwidth

IoT Device Nodes - Characteristics

- 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

Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

IETF RFC 7228: ***Terminology for Constrained-Node Networks***


Heavy networking stack just won't fit!

IoT Device Nodes - IoT Protocols

- 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


And every radio TX/RX is power hungry!

IoT Device Nodes - 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...)
- Accumulate (and filter) on the edge

IOT PROTOCOLS

Should I use WS to connect devices?


HTTP then?


WAYNE'S
WORLD
MQTT?


CoAP?
YES. YES.

MAINFLUX - EXAMPLE IOT PLATFORM


Centralized VS Decentralized

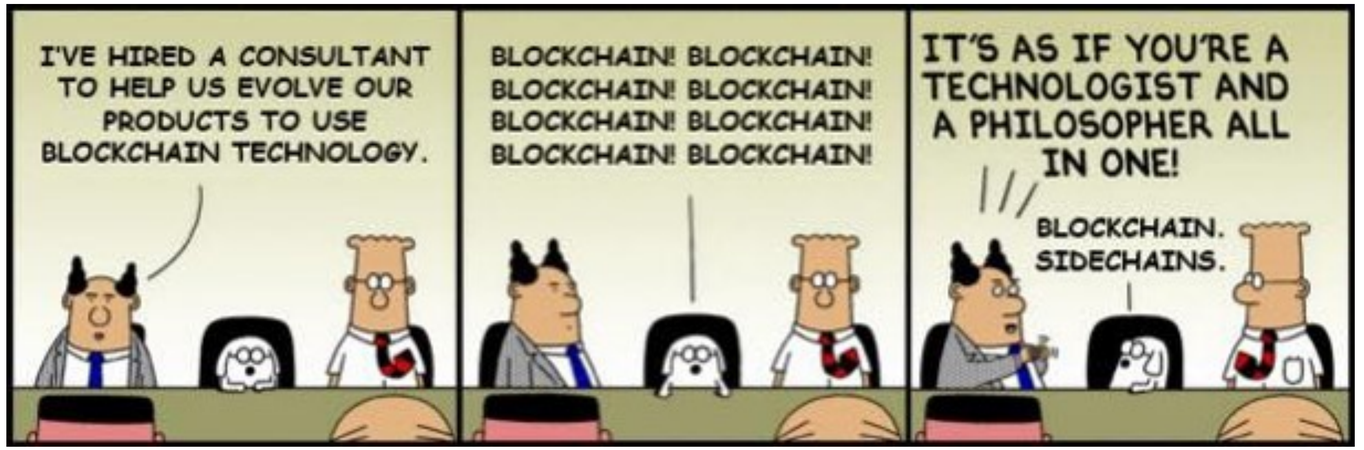
IoT and Cloud - Decentralized vs Centralized

Decentralized:

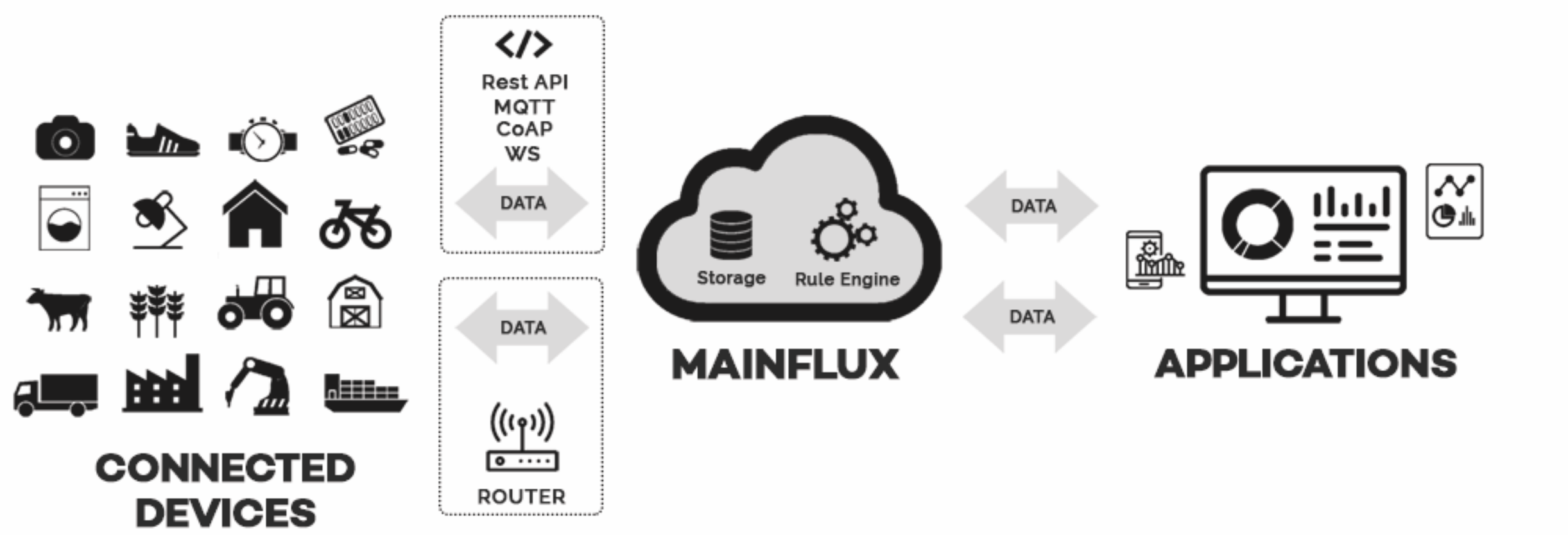
- AllJoyn (Qualcomm)
- IoTivity (Intel)
- Telehash
- Blockchain, Blockchain, Blockchain

Centralized

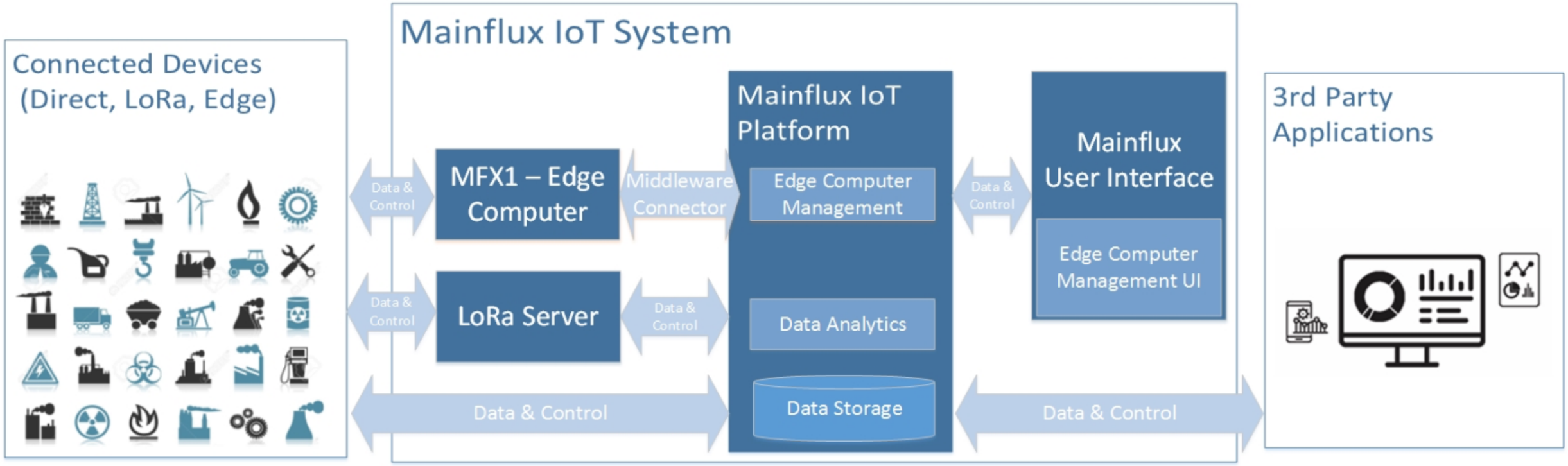
- Everybody else (practically)
- Mainflux included



IoT and Cloud - Typical IoT Architecture

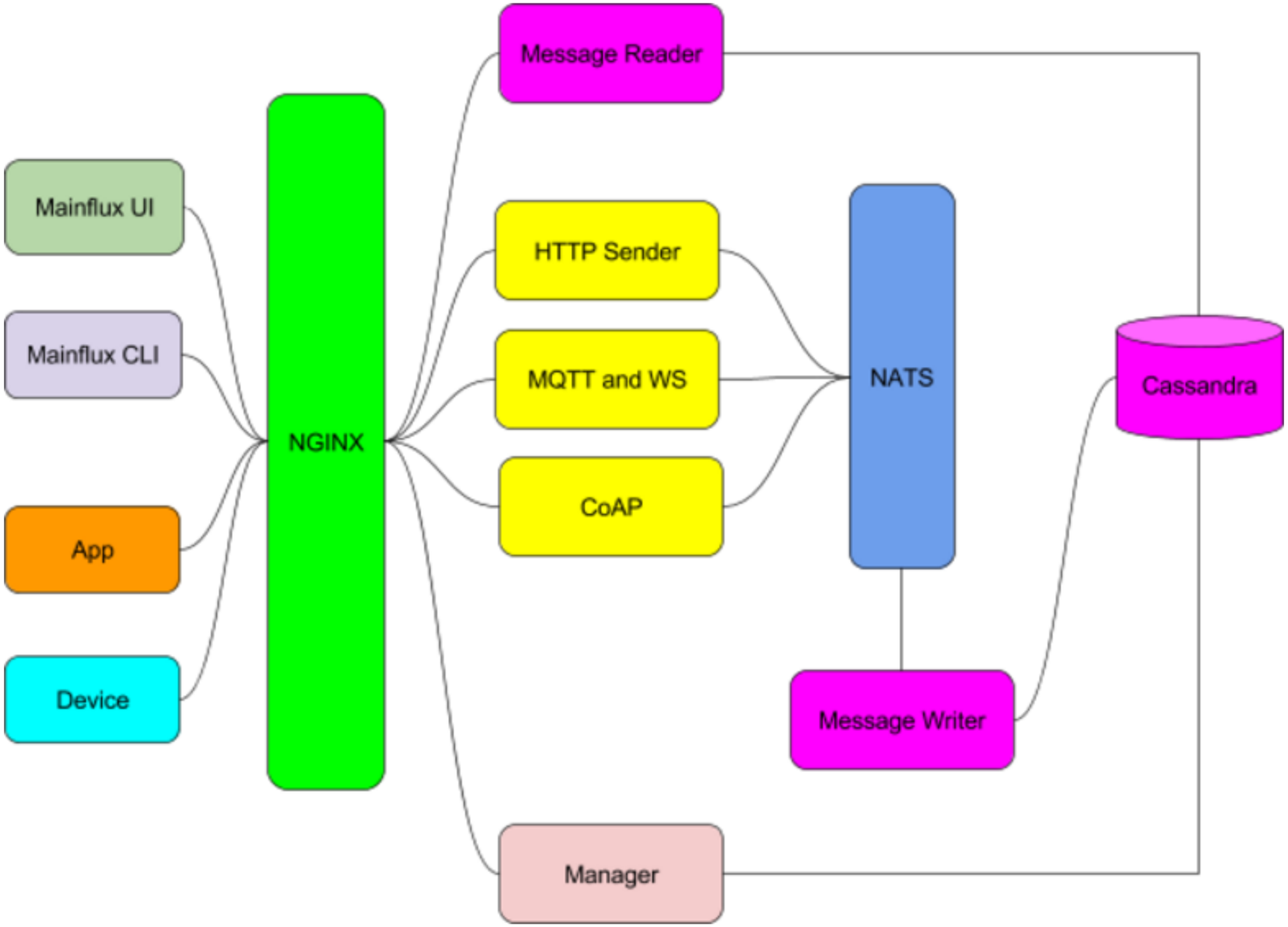


Mainflux - Example IoT Platform




Mainflux architecture is based
on microservices

Mainflux IoT Platform - Microservice Architecture



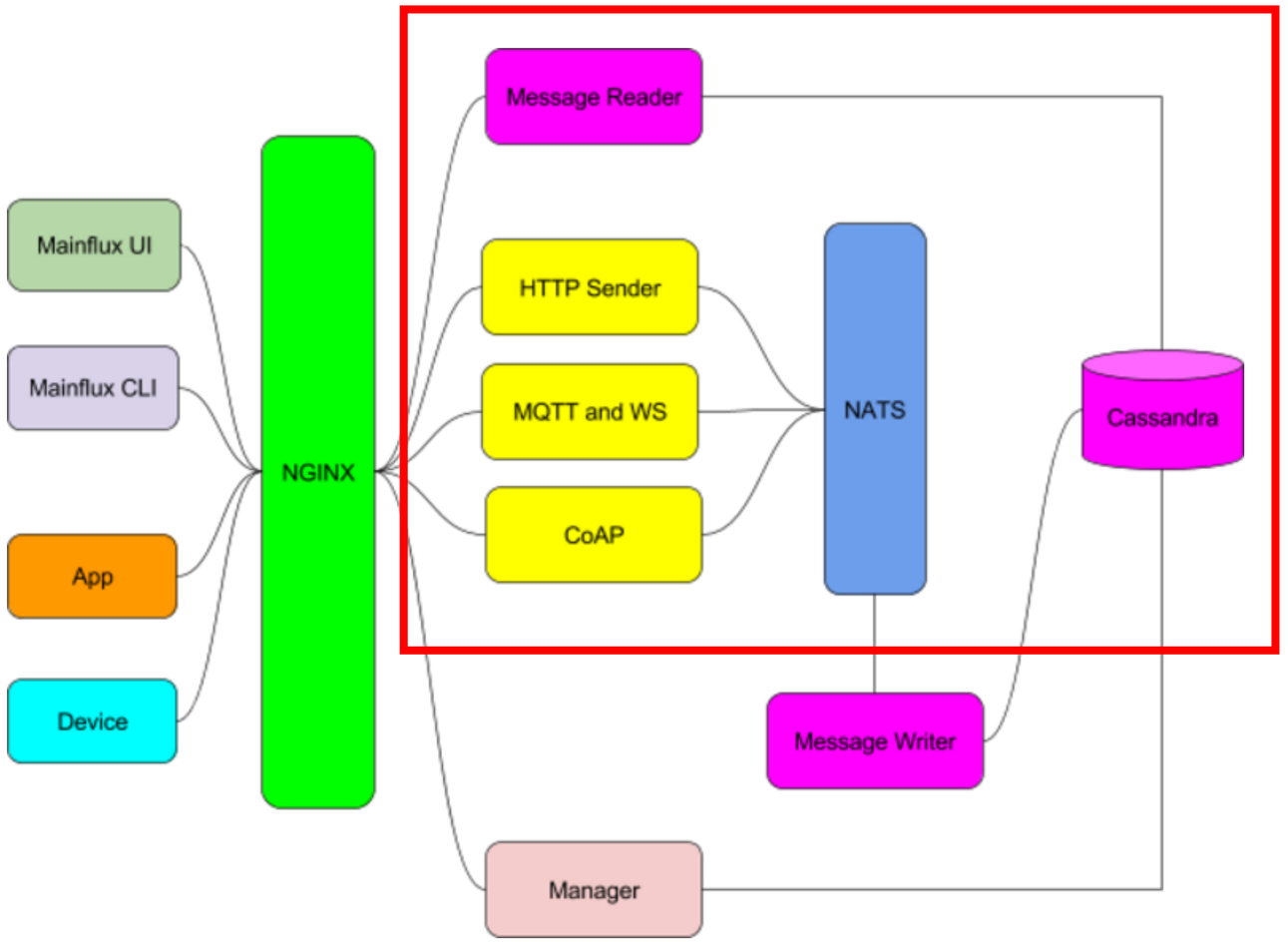
IOT MESSAGE SUBSYSTEM

Messaging Subsystem

Messaging subsystem is composed of following microservices:

- HTTP Server
- MQTT (and WS) Broker
- CoAP Server
- NATS Broker
- Cassandra Adapter and Storage

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.



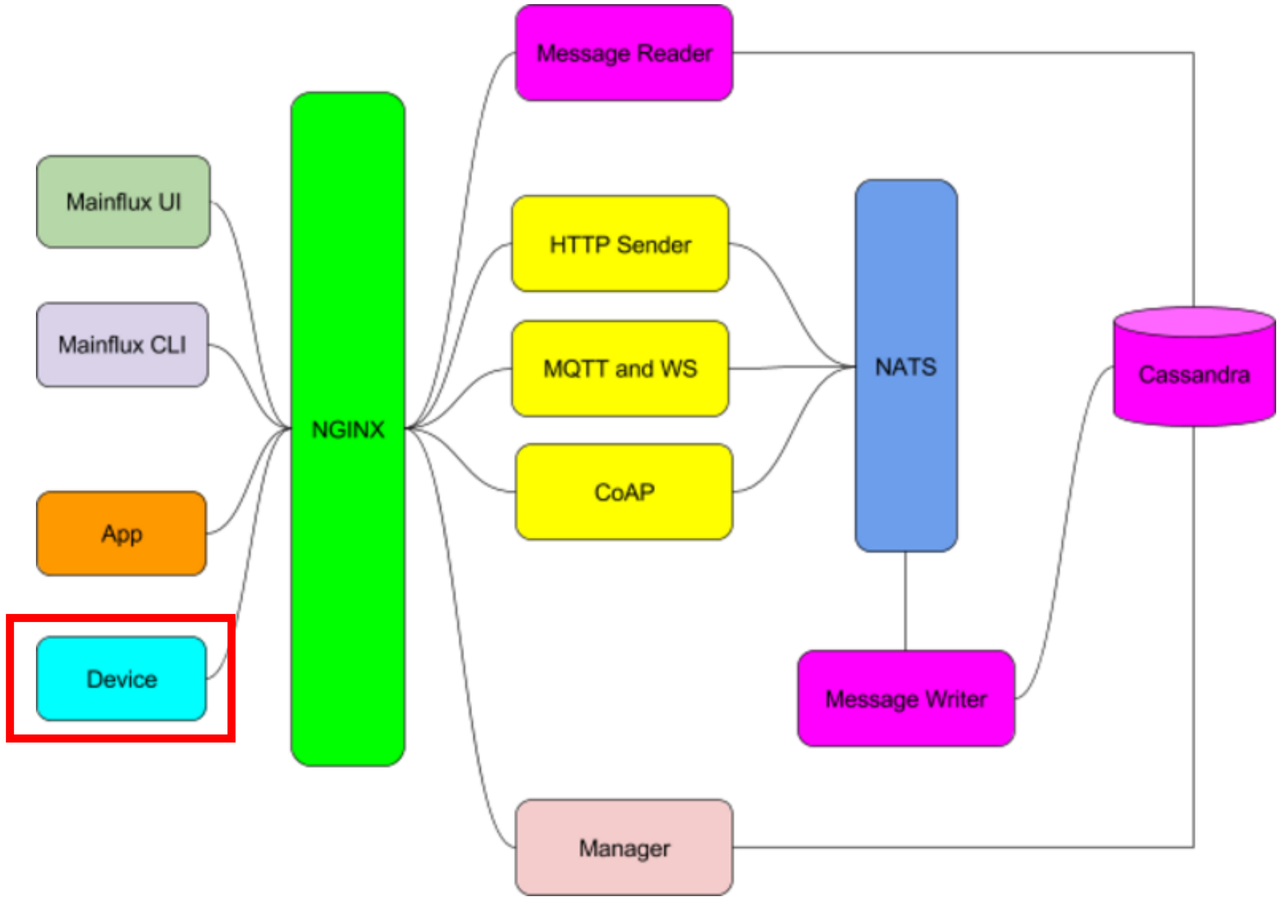
IoT Protocols - Protocol Choices

Devices

- MQTT
- CoAP

Apps

- HTTP
- WS



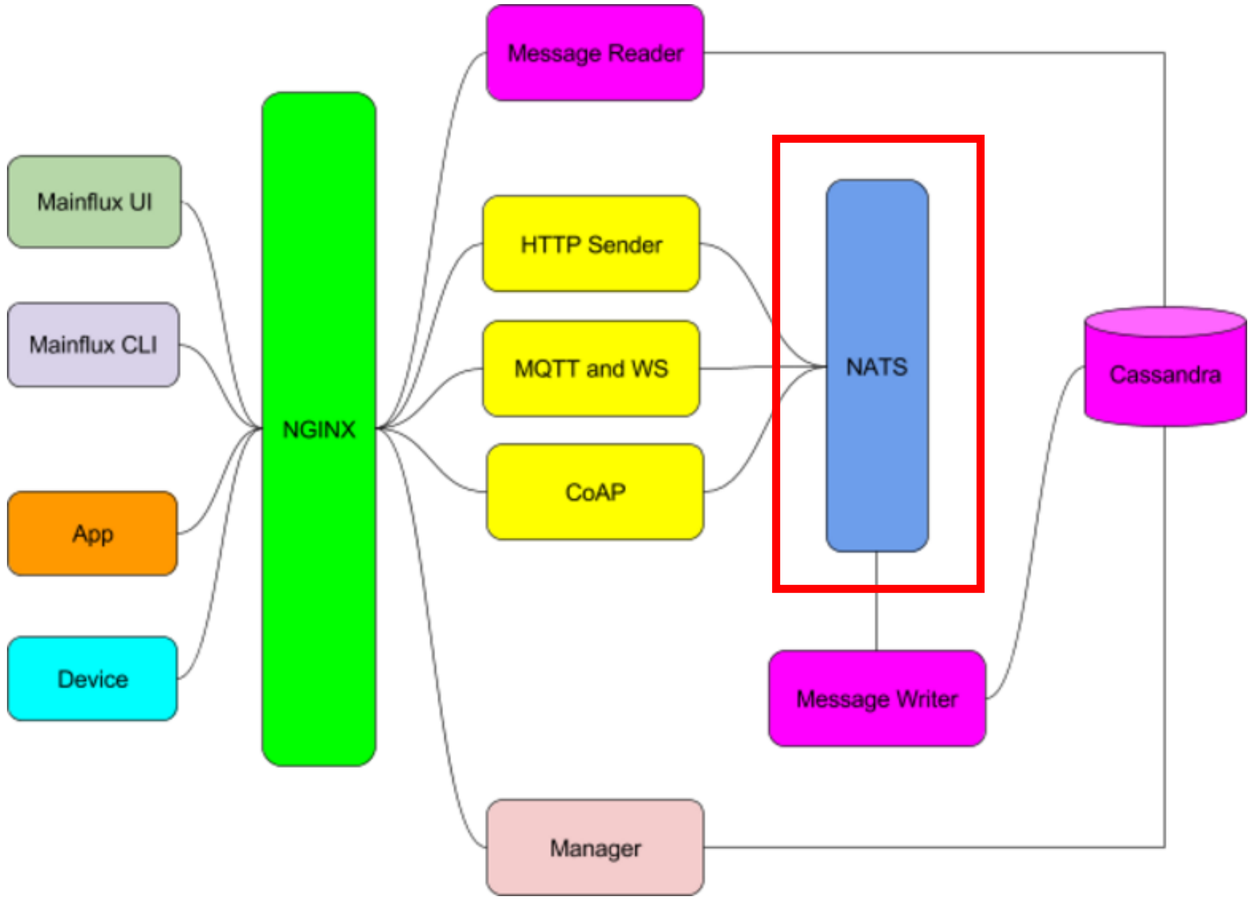

HTTP-to-WS-to-MQTT-to-CoAP?


Yes. Mainflux is bridging protocols.
IWDRM
GIFsFORUM.com

Messaging Subsystem - NATS

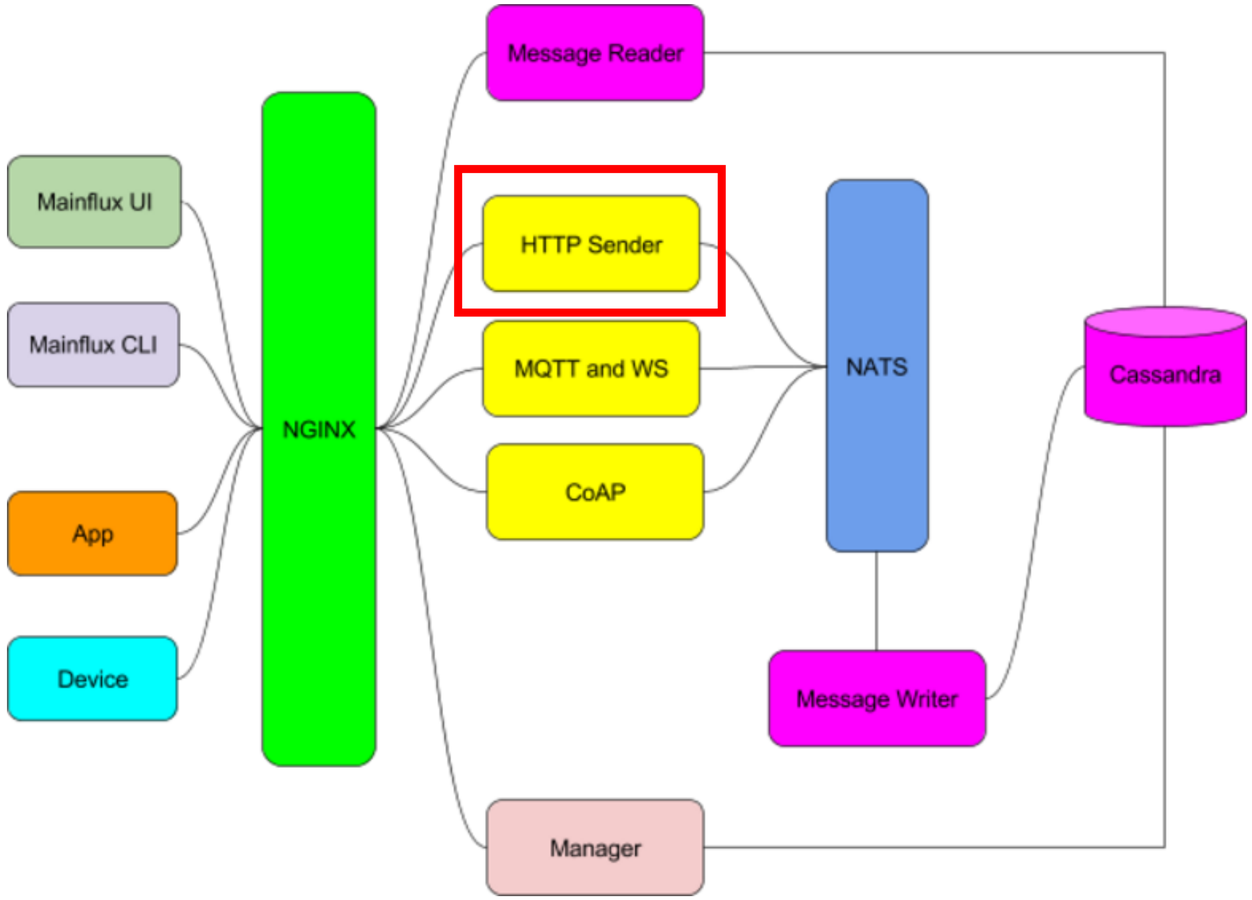
- Brokers the IoT messages as an events in Mainflux NATS Message format

```
// RawMessage represents a message emitted by the
mainflux adapters layer.
type RawMessage struct {
    Channel     string `json:"channel"`
    Publisher   string `json:"publisher"`
    Protocol    string `json:"protocol"`
    ContentType string `json:"content_type"`
    Payload     []byte `json:"payload"`
}
```



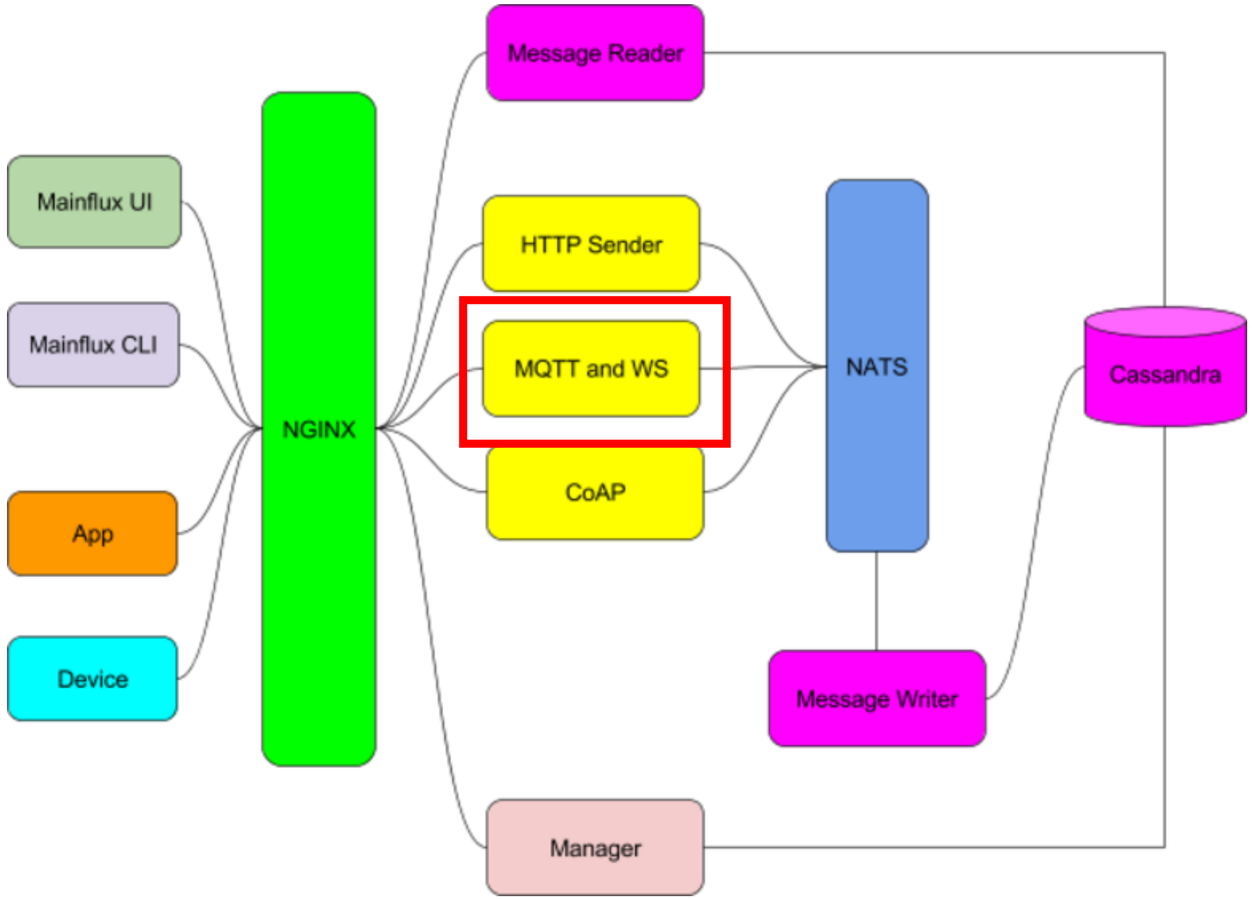
Messaging Subsystem - HTTP

- 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
- Endpoint: `/channels/<channel_id>/messages`



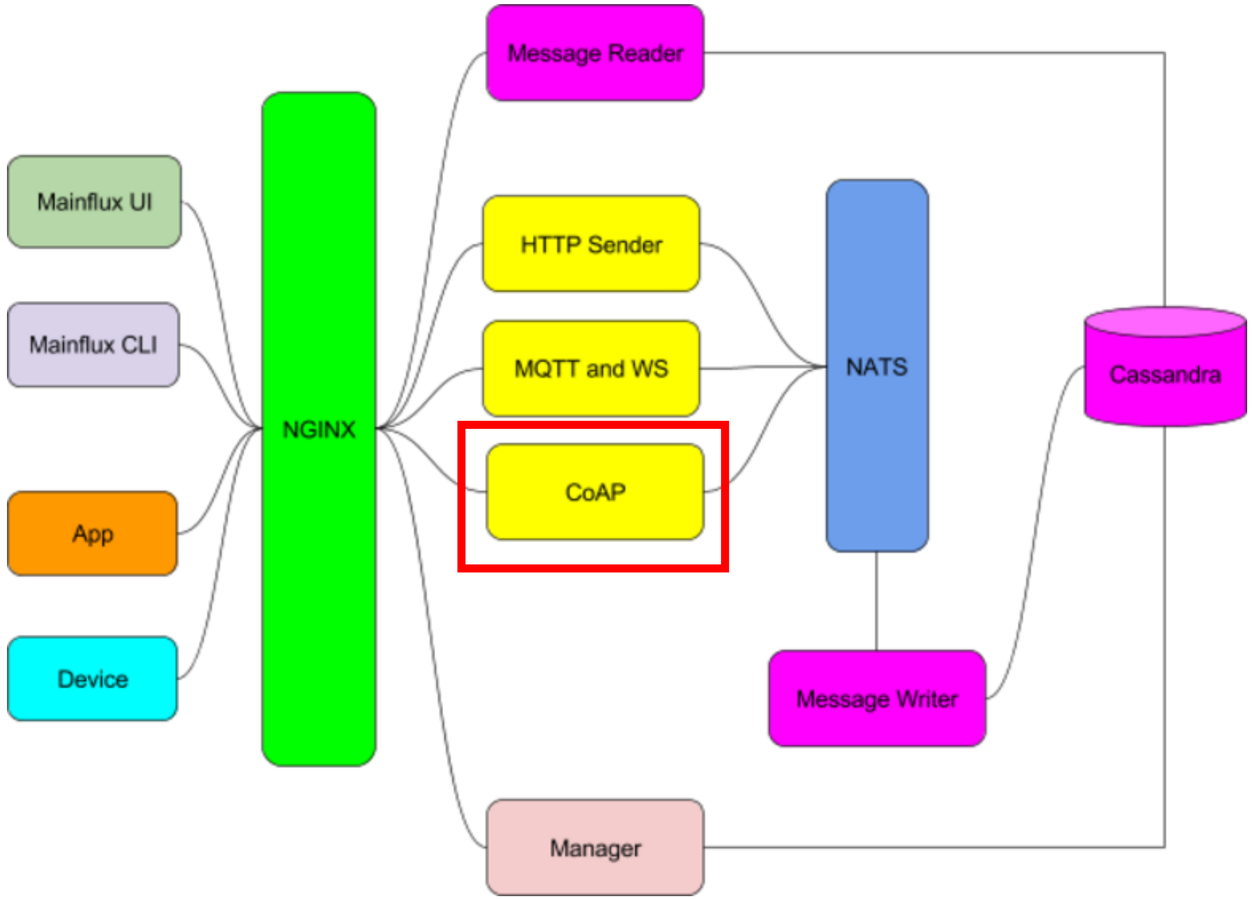
Messaging Subsystem - MQTT

- 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`



Messaging Subsystem - 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
- Supports Binary Message Format – Stored as binary blob
- JSON Message example

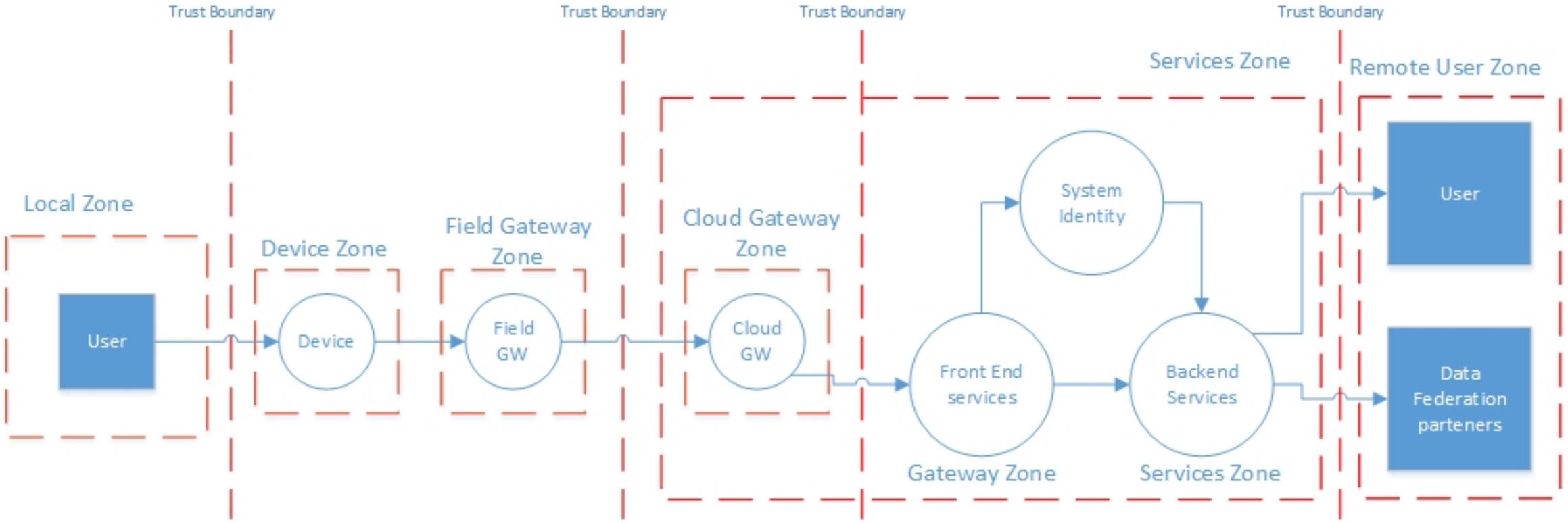
```
[
    {"bn":"urn:dev:ow:10e2073a0108006;",
       "bt":1.276020076001e+09, "bu":"A",
       "bver":5, "n":"voltage","u":"V","v":120.1},
      {"n":"current","t":-5,"v":1.2},
      {"n":"current","t":-4,"v":1.3},
      {"n":"current","t":-3,"v":1.4},
      {"n":"current","t":-2,"v":1.5},
      {"n":"current","t":-1,"v":1.6},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

SECURING IOT SYSTEM

IoT Security

- Secure the Device
- Secure the Communication Channel
- Secure the Field Gateway
- Secure the Server/Cloud Gateway
- Secure the IoT Server/IoT Cloud

- Secure the Microservices
- Secure the Data at Rest
- Secure the Data in Motion
- Secure the Apps on Top of IoT System



Securing the Edge Gateway

- HW TEE (Trusted Execution Environment)
  - Required in platform to protect and isolate security sensitive values
- Key Store/Key Management
  - Required in platform to protect stored keys
  - Containerize Key Vault
  - Later someone would need to connect this to other OS services and hardware RoT
- RNG (Random Number Generator)
  - TRNG (True Random Number Generator)
  - DRNG (Deterministic RNG)
- Secure Boot
  - Signature validation at each boot level
  - Integrity checks at each boot level
  - Connection into chain of trust in EdgeX
  - System will only boot of integrity checks pass
- Digital Signature Algorithm
  - ECDSA
- Access Management
  - OAuth2.0 Roles
  - Resource Owner
  - Client
  - Resource Server
  - Authorization Server

Securing the Server - Auth Server

AuthX - Authenticate devices and users

- CA must be burned into device flash for server auth for TLS
- Client-side certificates for higher security - use small-size encryption (constrained devices), for example TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- JWT (current Mainflux implementation)
- OAuth2.0 Client Secret - how do we issue temporary token to devices?

AuthZ - Authorize devices and users

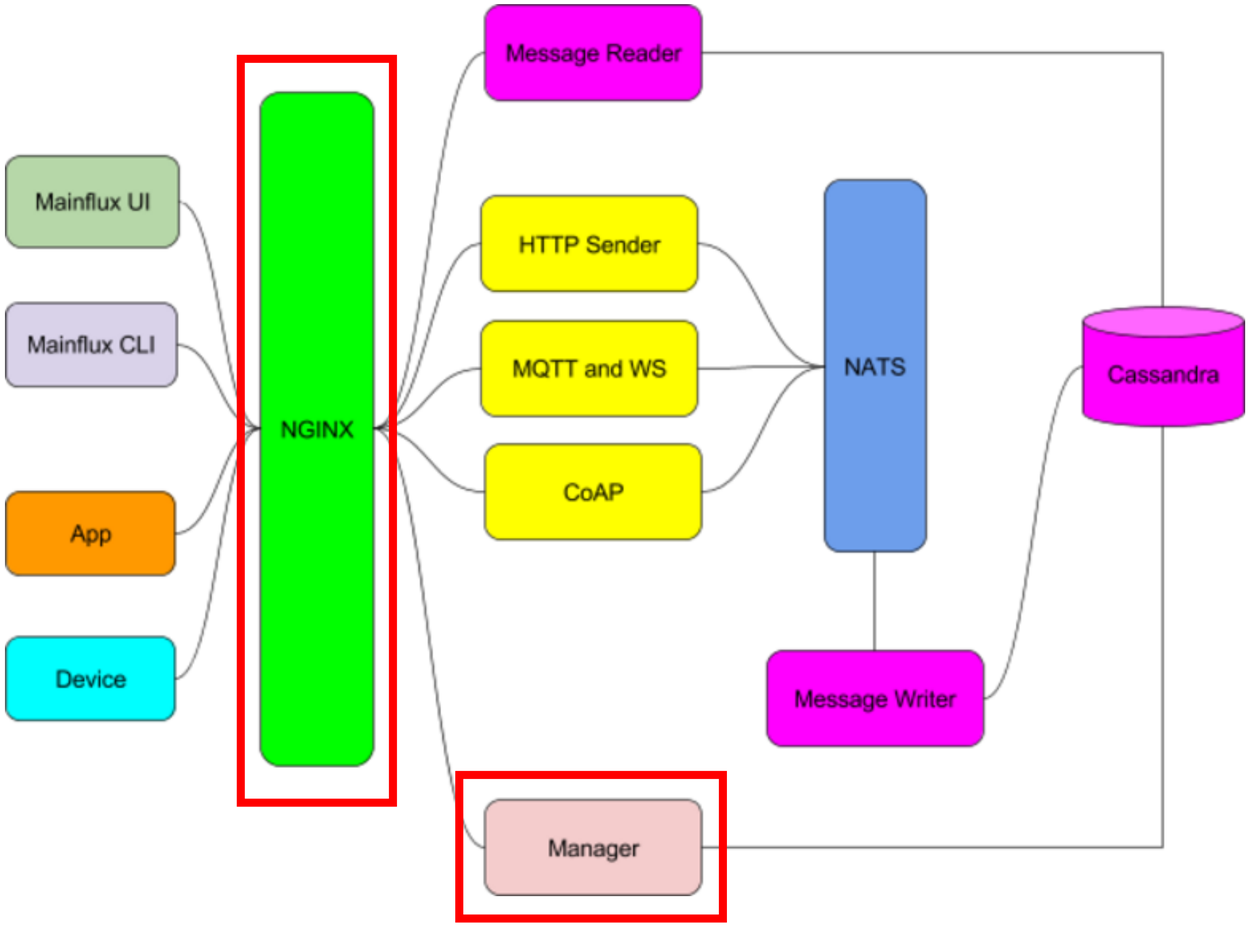
- Scoped API keys - JWT
- Drawback - no revoke, and device JWT has infinite TTL
- Revoke destroys stateless approach - so use just for session tokens with low TTL and no revoke
- Mainflux uses Access Control Policies - inspired by AWS IAM Policies


Intercept the requests and send them for Auth

Mainflux – Auth Subsystem

- Reverse Proxy
- SSL termination - keep these certs at one place.

  MQTT is pure TCP, and CoAP is UDP
- NGINX - MQTT TLS Termination
- NGINX - UDP DTLS Termination
- Policy Based Auth
  - Grouping devices by connection to channels
  - Channel corresponds to MQTT topic
- JWT and Certs for Devices
  - JWT (short TTL)
  - Bearer Token

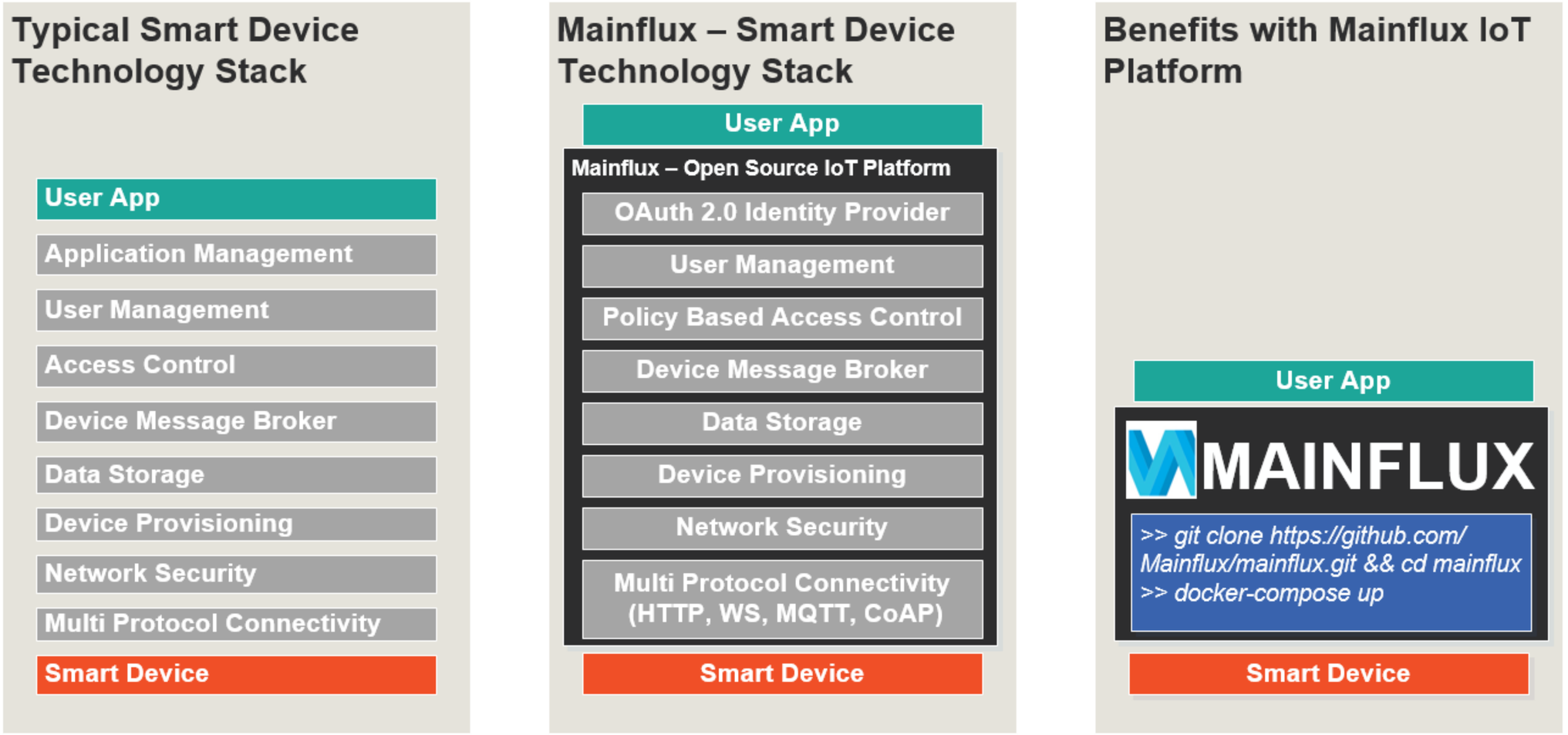


Securing Microservices

- Continuous Monitoring Is Critical. Discover and monitor inter-service communications
- Diversify Security Tactics. Secure data in motion and data at rest
- How do we know that no one has changed the micro service?
  - Securing the containers before they land in production
  - Sign Docker Images

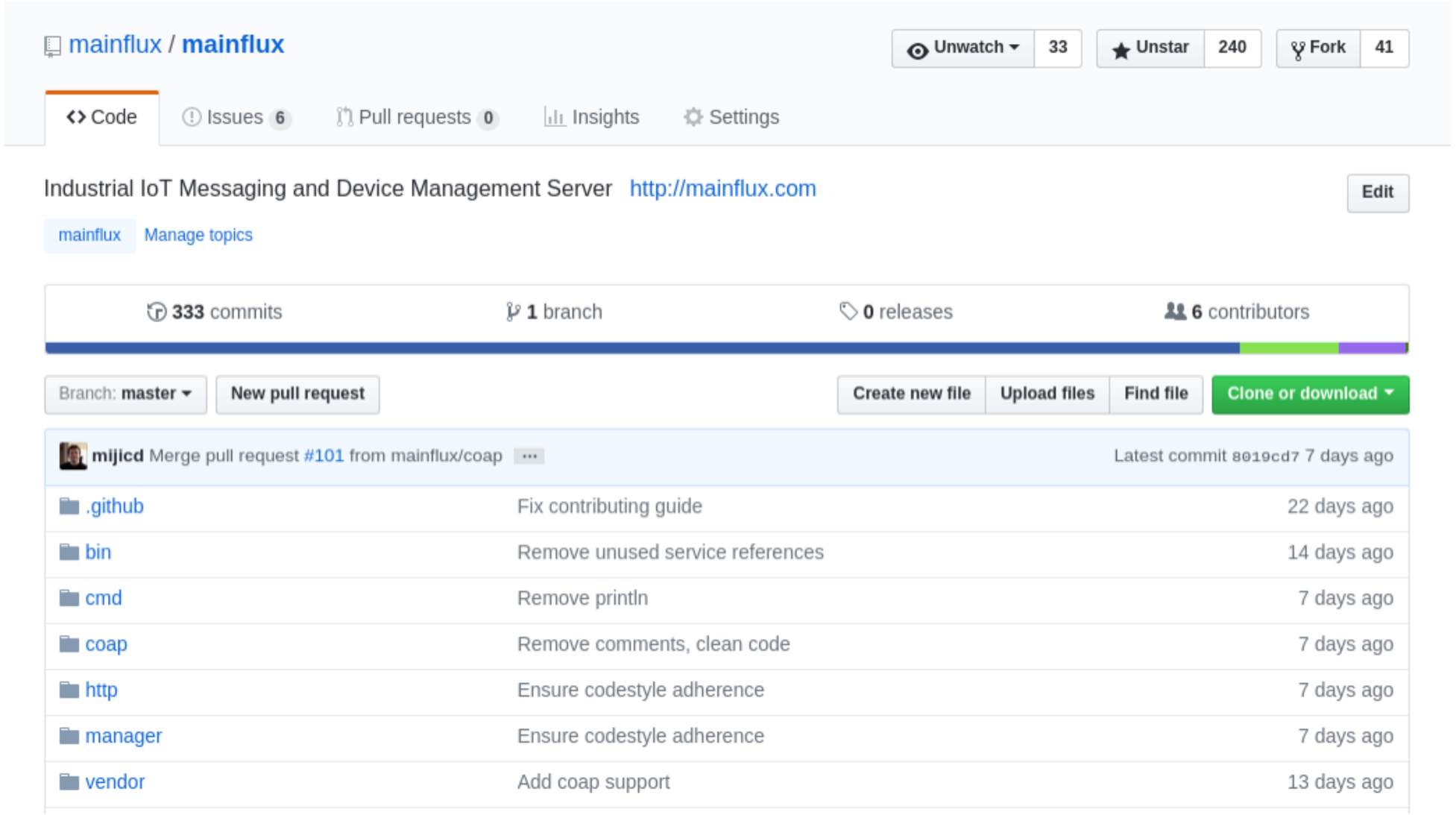
- Each microservice has to be able to stand on its own security-wise
  - Secure the microservice API with authentication or access tokens from centralized server
  - Certificate-based security. Use TLS for communication between microservices
  - Use anomaly detection system to monitor communication between microservices

Mainflux Benefits




`docker-compose up` and you got your Mainflux system running!
I'm the king of the world!

Mainflux - Internet For Robots And Machines



Thank you!

Janko Isidorovic

janko@mainflux.com

@janko-isidorovic

@jankoisidorovic