MAINFLUX

Open Source Internet of Things Technology & Consulting Services

Technology Overview – IoT Fuse 2019
Part 1

Janko Isidorovic

Mainflux COO & Co-founder

Janko is the co-founder of Mainflux IoT open source project.

He is also chair of the Application Work Group of Linux Foundation EdgeX project.

Janko has a 10+ years background in Project Management, IT and Software integrations.
He holds MSc. In Telecommunications.

Agenda

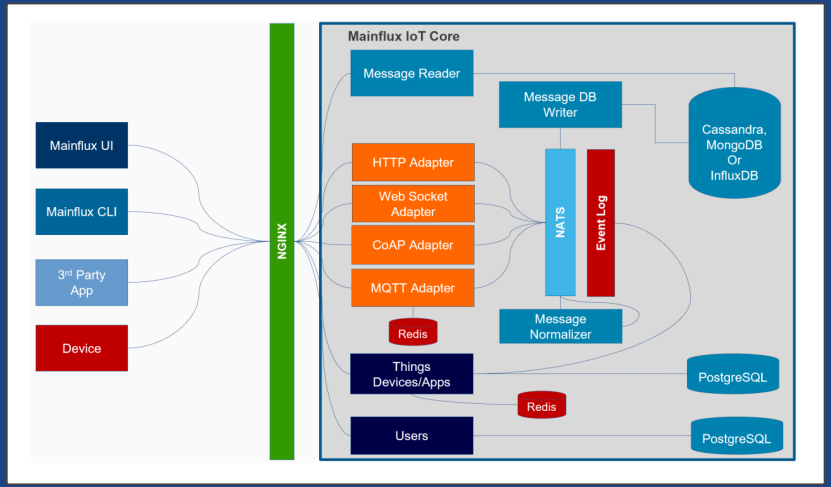
1. Mainflux Technology Overview
2. Lab – Deploy Mainflux using Docker
3. How to deploy Mainflux on Kubernetes
4. Mainflux Add-ons Overview

IoT Fuse 2019 – Slides and scripts:
https://github.com/janko-isidorovic/iotfuse2019

Mainflux GitHub:
https://github.com/mainflux/mainflux

MAINFLUX

What is Mainflux?



MAINFLUX – The Project

Internet of Things Platform

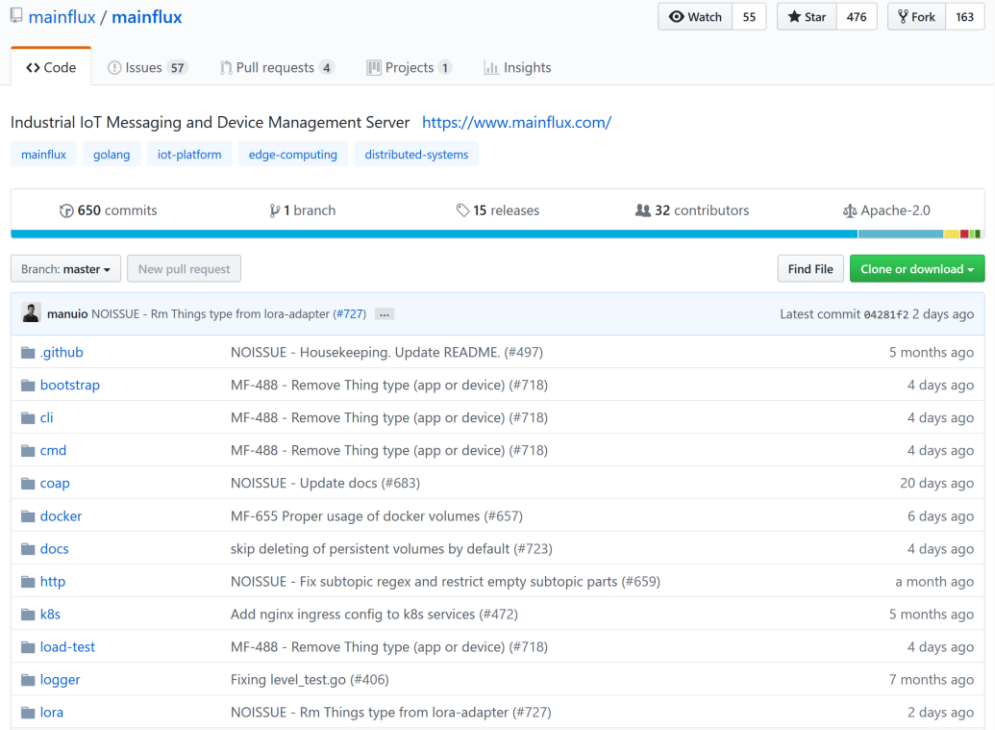
Open source Apache 2.0 license

Production ready

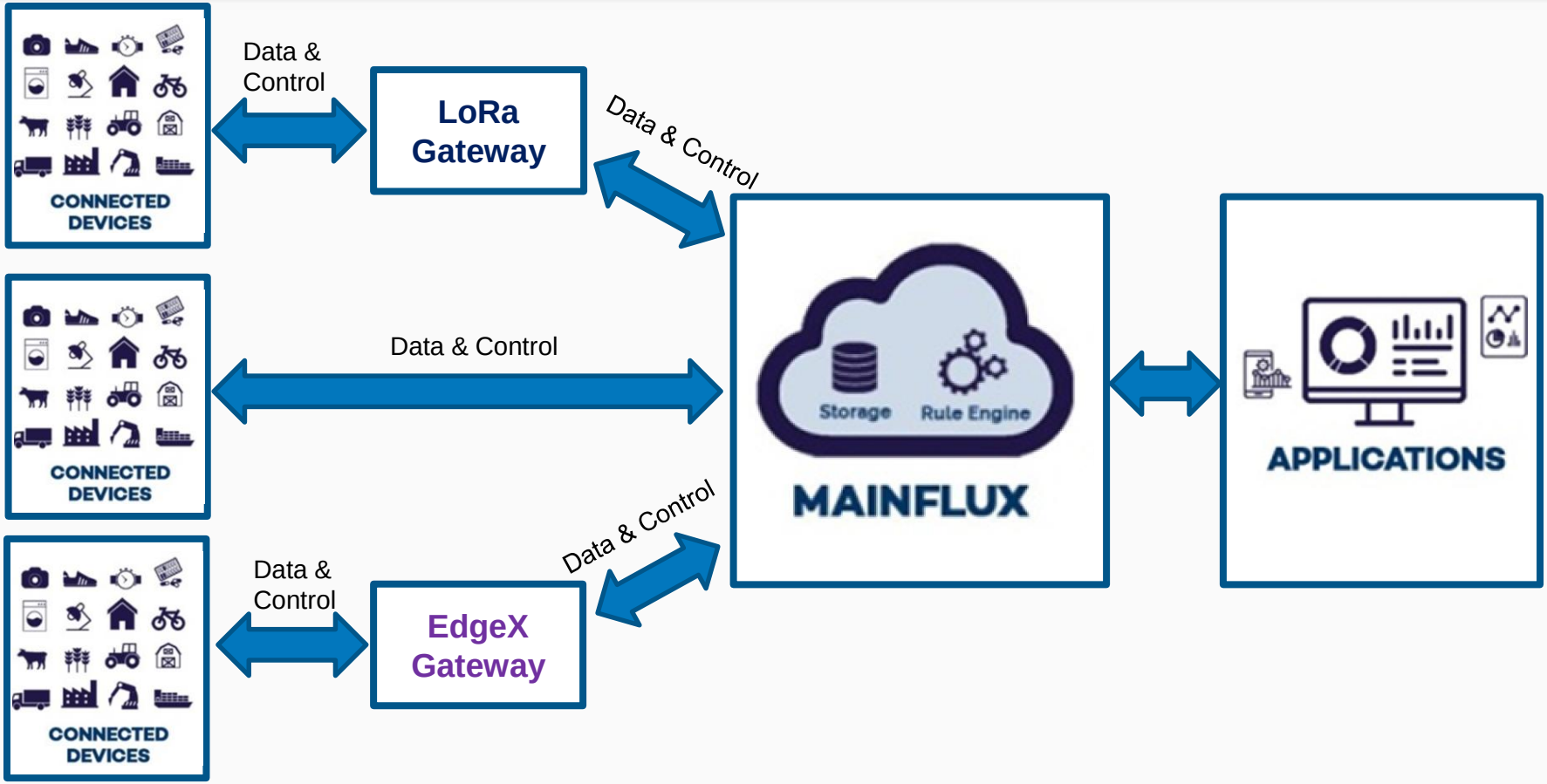
Mainflux GitHub:

https://github.com/mainflux/mainflux

200.000 Docker Hub Downloads



MAINFLUX – The Project

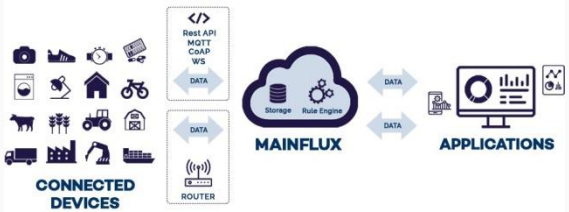


MAINFLUX – The Company

Mainflux is a technology company offering:

- Full-stack, open source, patent-free IoT Platform
- Expert consulting services
- MFX-1 IoT EdgeX Gateway

Open-Source IoT Platform



Full-scale, comprehensive, secure and performant IoT Platform, developed in Golang and deployed in Docker or Kubernetes

Consulting Services

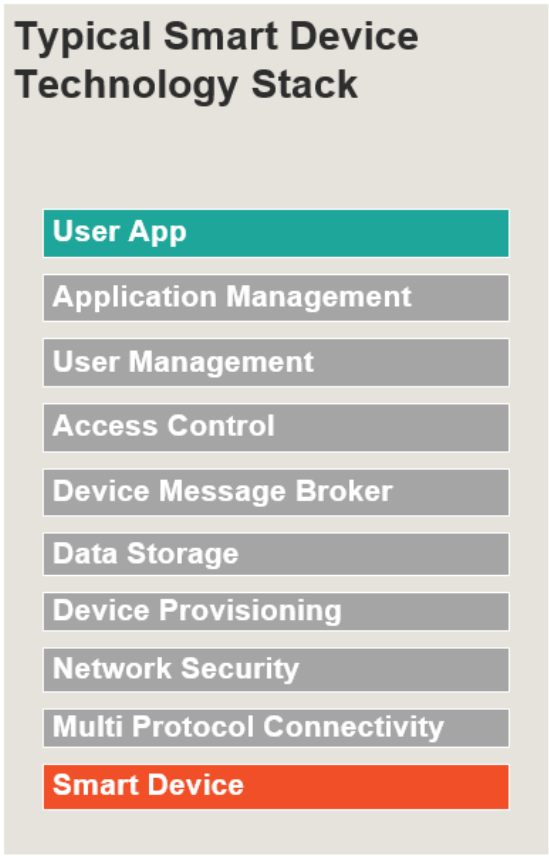
Provided by cross-functional team, covering both software and hardware layers of the IoT technology

MFX-1 IoT EdgeX Gateway

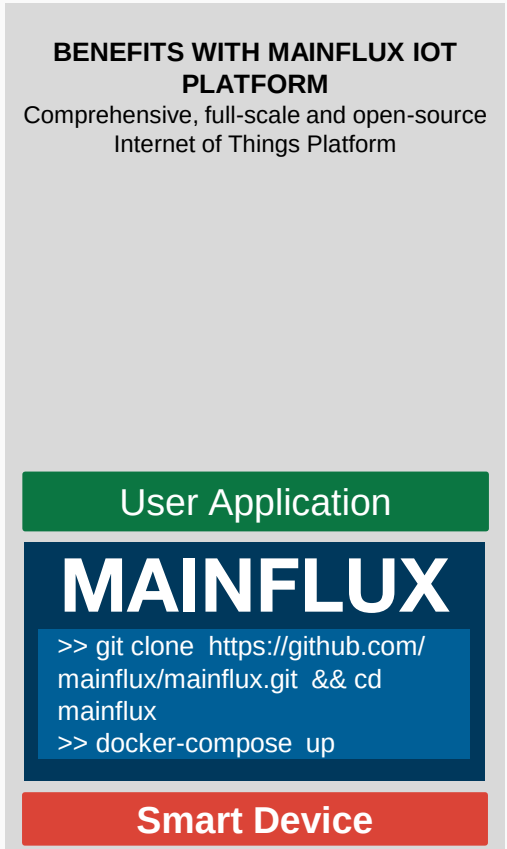
Compliant IoT gateway for Linux Foundation's EdgeX Foundry Project

MAINFLUX – Existing IoT Technologies Problems

- In order to develop an IoT application today, a collection of complex middleware components (server infrastructure) must be assembled and integrated: device management and provisioning, messaging system, data storage, security...
- In order to develop an IoT application today, a collection of complex middleware components (server infrastructure) must be assembled and integrated: device management and provisioning, messaging system, data storage, security...
- Every new application has to re-implement these middleware components, introducing the risk of bugs and failures
- IoT platforms often lack the support of new IoT protocols and support only specific hardware devices thus limiting connectivity choices
- Proprietary IoT platforms generate lack of control, vendor-lock and client access license costs
- Majority of IoT platforms are SaaS-only. Many organizations seek an on-prem deployment option because of security, data-privacy, response latencies and other organizational reasons
- Vertical integration often required an edge gateway in the data path however most platforms do not have an adequate solution for this use case



MAINFLUX – Existing IoT Technologies Problems

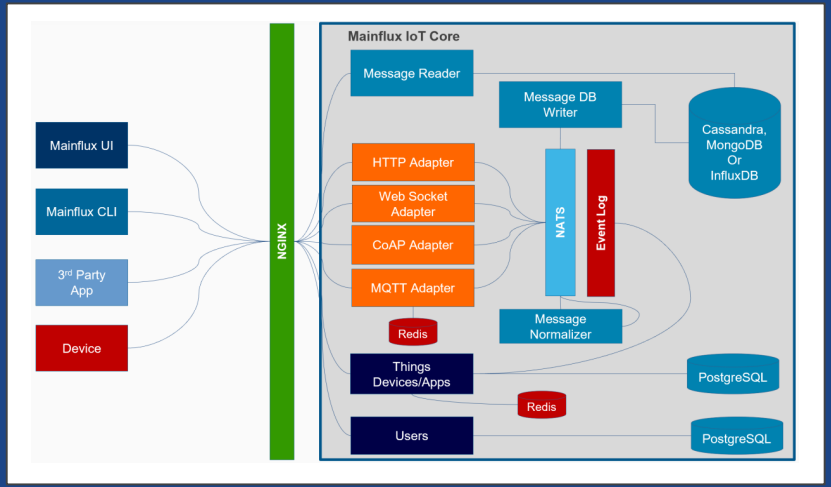


- Mainflux solves these problems by creating a comprehensive reusable middleware software platform for building IoT solutions
- Easy to deploy on-prem or in the cloud
- SaaS option available through Mainflux (Q3 2018)
- Built for extensibility - No need to reconfigure or modify the core platform when adding functionality for vertical solutions and applications.
- Hardware and Network Protocol agnostic - connect any device over any transport
- Open-source for code adaptation, bug fixing, community support and verification
- Supports any EdgeX Foundry compliant gateway

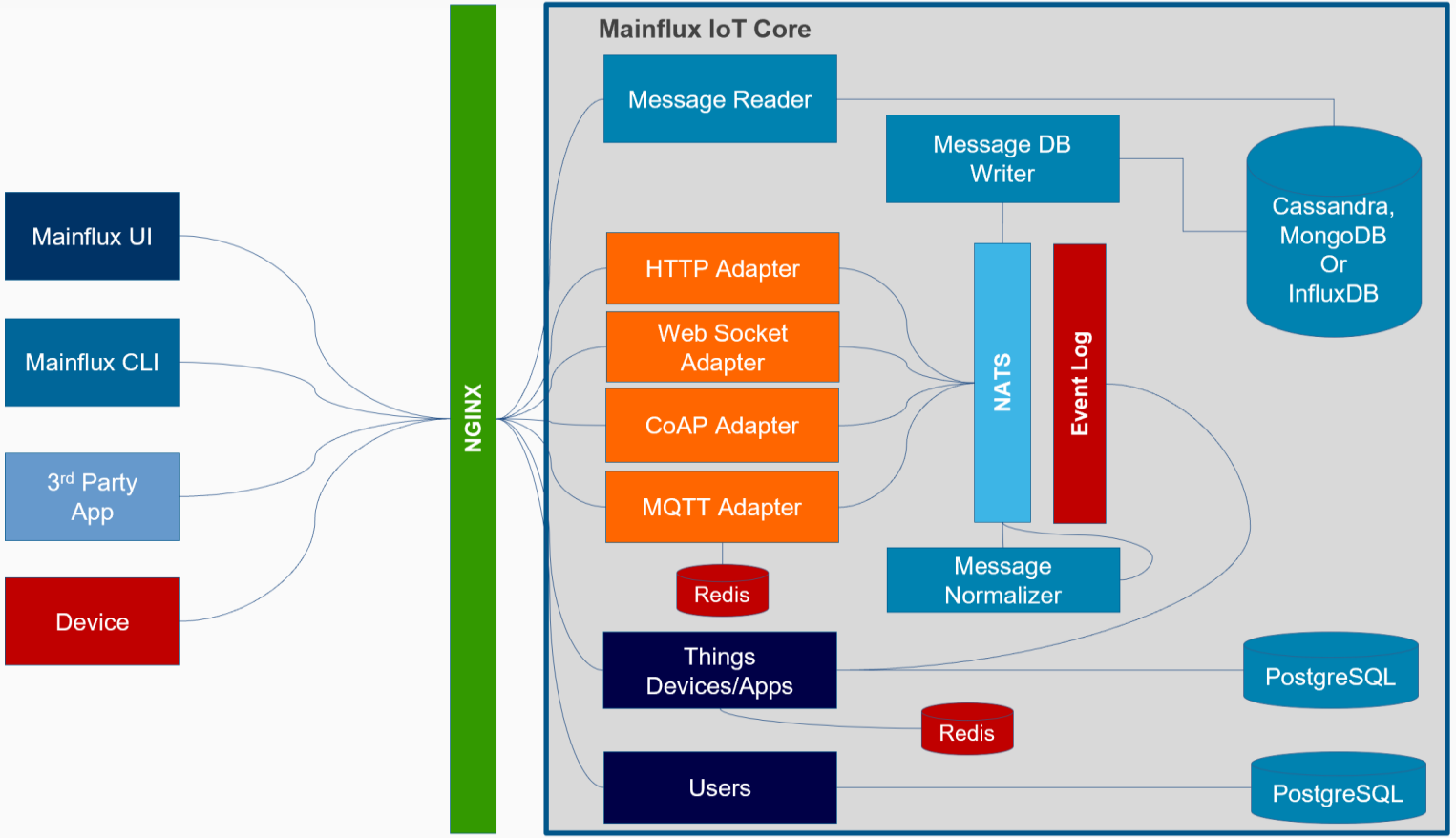
User only connects sensors and starts building the application!

MAINFLUX

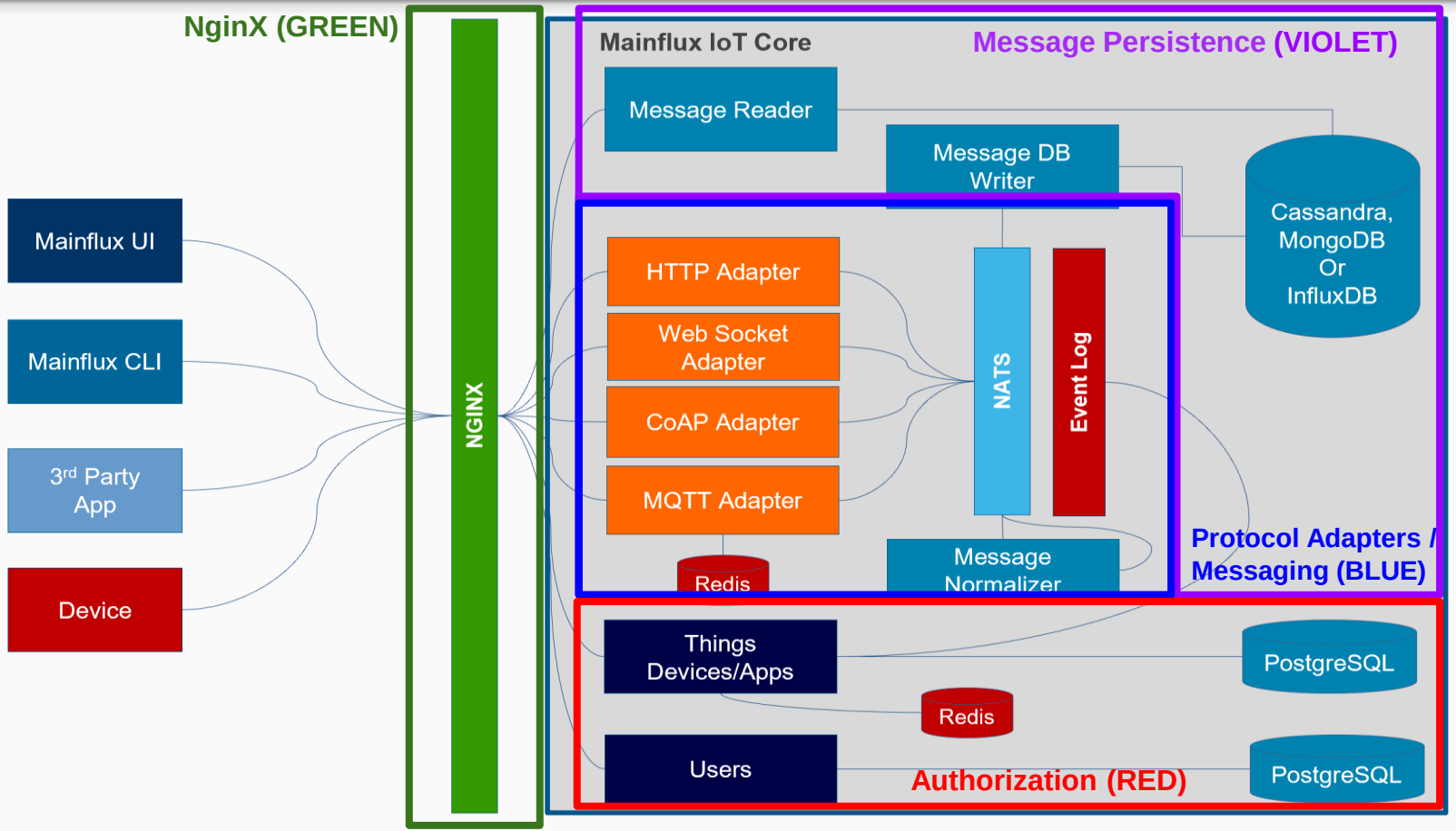
Architecture



MAINFLUX - High Level Design

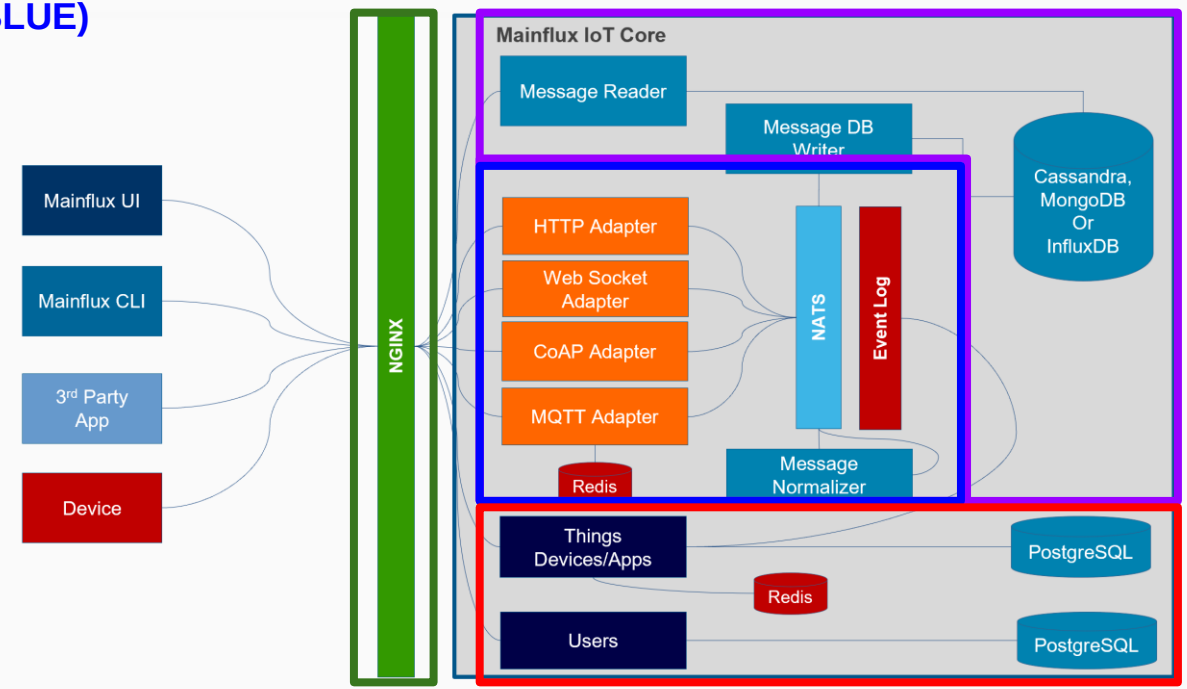


MAINFLUX - High Level Design



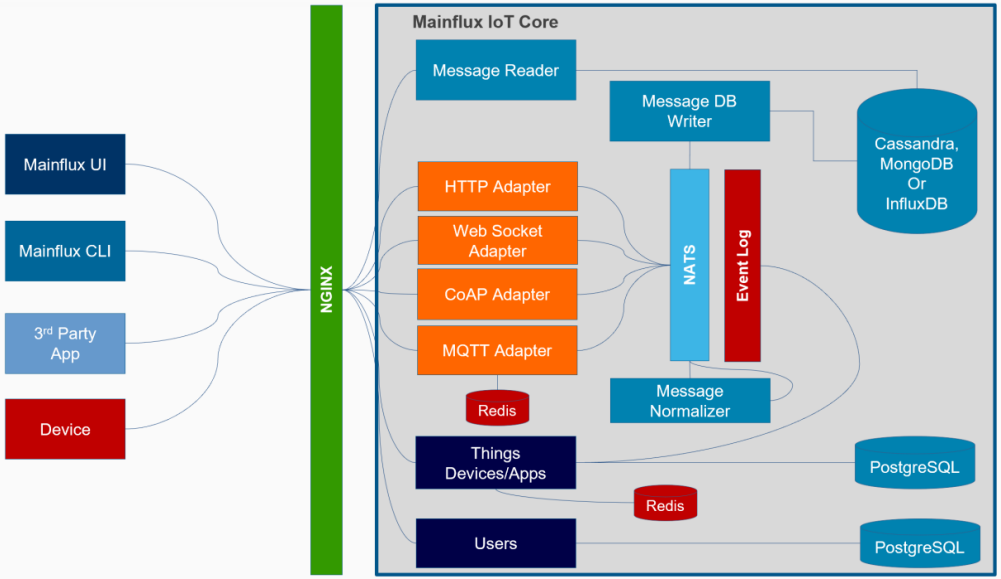
MAINFLUX - Subsystems

- **Protocol Adapters/Messaging (BLUE)**
  - HTTP
  - MQTT
  - WebSocket
  - CoAP (Experimental)
- **Authorization (RED)**
  - Users Management
  - Device Management
  - App Management
- **Message Persistence (VIOLET)**
  - InfluxDB or
  - MongoDB or
  - Cassandra
  - or all of them
- **NginX (GREEN)**
  - TLS/DTLS Termination
  - Reverse Proxy



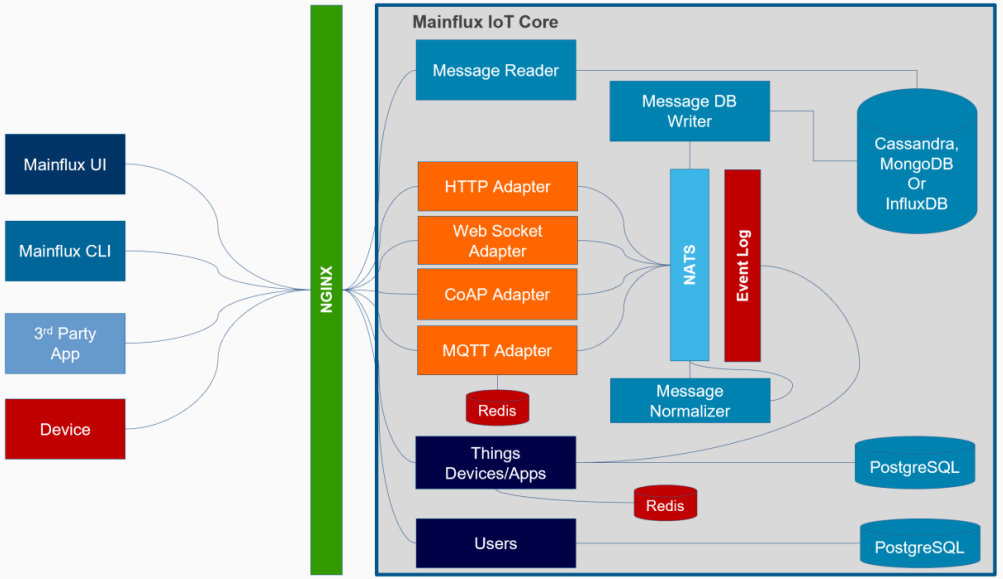
MAINFLUX - Technologies

- Microservice Architecture
- Golang wherever possible - Go Kit
- NATS Message Bus
- NginX
  - TLS/DTLS Termination
  - Reverse Proxy for UI
- SQL database for structured data
- NoSQL database for Telemetry:
  - InfluxDB
  - MongoDB
  - Cassandra
- Mainflux Scales from PRi class devices to multi datacenter with Kubernetes and Cassandra DB
- Deployment:
  - Native
  - Docker containers (compose provided)
  - Kubernetes scripts



MAINFLUX - System Entities

- **Users**
  - User represents a human user of the system and is basic entity in the Mainflux IoT Platform. User is authenticated by email and password.
  - Once authenticated user receives JWT to use for further actions.
  - Each user is Admin within his domain
- **Things**
  - Connected devices and applications are the same entity. We call them Things.
  - Internal representation of every device is saved to database
- **Channels**
  - Channel connects Things (devices and/or applications)
  - Only Things connected to the same Channel can communicate with each other.



MAINFLUX - Protocol Adapters

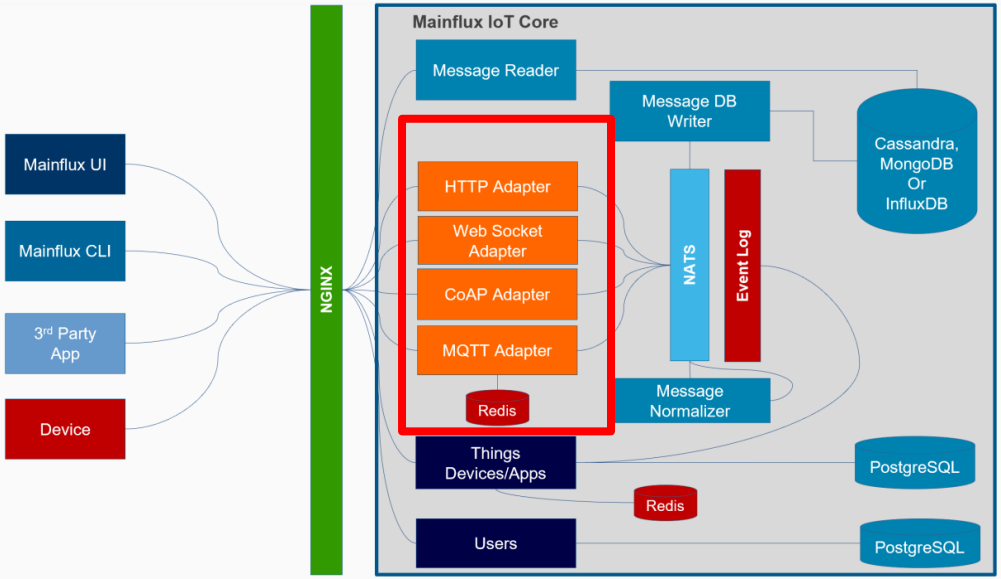
Out if the Box Mainflux Supports:

- HTTP
- MQTT
- WebSocket
- CoAP (Experimental)

Redis Cache is used to preserve the MQTT messages sent to device while the device is offline

Each connection is authorized against Things service

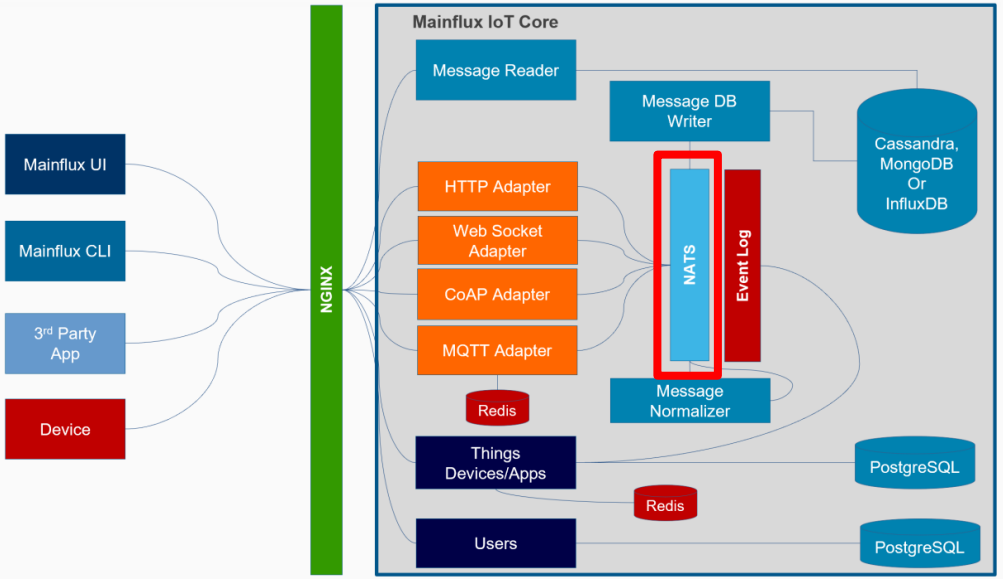
Mainflux uses gRPC protocol to connect Protocol Adapters to Things Service



MAINFLUX - Channels and Device 2 Device Communication

NATS Message Bus

- Highly performant
- Easy to cluster across servers
- Bridging between protocols
- Channels for communication between devices and apps
- Security - Only devices and apps connected to same channel can communicate with each other



MAINFLUX - Message Format

SenML is IETF standard (https://tools.ietf.org/html/draft-ietf-core-senml-05)

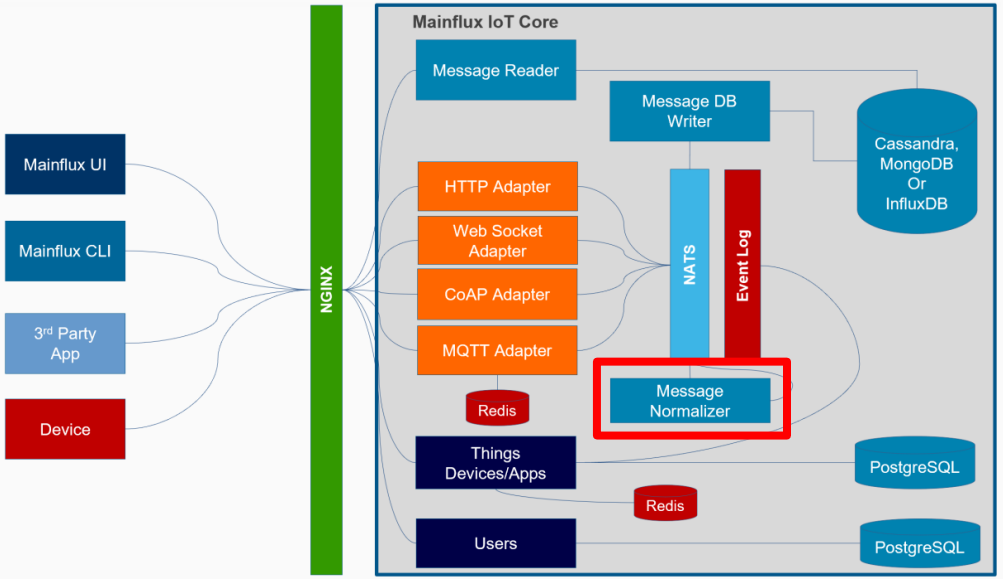
SenML provides simple model for retrieving data from sensors and controlling actuators

Minimal semantics for the data inline

Allows for more metadata with in-line extensions and links

Send Array of Measurements using single Message

Normalizer Microservice is used to Normalize Message containing Message Array into multiple messages with correct timestamp.



MAINFLUX - Message Format

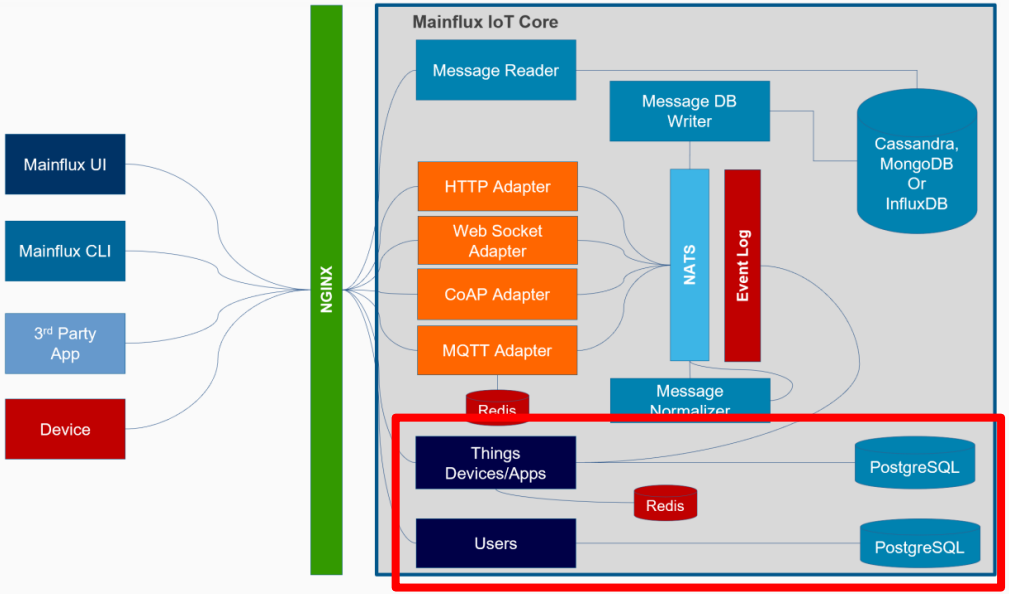
SenML Message Example:


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

MAINFLUX - Authorization

During the process of entity creation (so-called provisioning), each user or client (device or application) is issued an authentication token.

- User Authorization
  - Username and password
  - JWT
- Things Authorization
  - Simple String Token
  - Server's Public Certificate for TLS/DTLS
  - mTLS
- Message Authorization
  - Only the Things plugged into the same channel can communicate over it



MAINFLUX - Security

Authentication with Mainflux keys

- Mainflux key is a secret key that's generated at the Thing creation

docker-compose -f docker/docker-compose.yml up

Mutual TLS Authentication with X.509 Certificates

- Client-to-server authentication using client-side X.509
- This is called two-way or mutual authentication
- Mainflux supports mTLS over HTTP, WS, and MQTT protocols (no CoAP at the moment)
- Thing key will be used to create x.509 certificate
- HTTPS – Authorization header does not have to be present
- MQTTS – Password filed in CONNECT message must match the key from the certificate
- WSS – Authorization header or authorization query parameter must match cert key

AUTH=x509 docker-compose -f docker/docker-compose.yml up -d

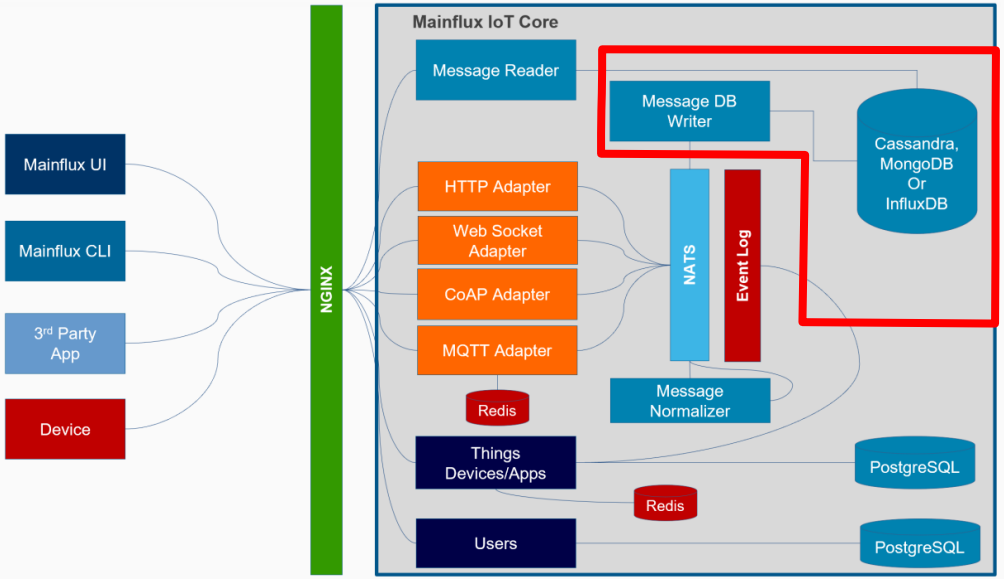
MAINFLUX - Message Persistence

Message DB Writer is reading normalized messages from NATS and writing them to DB

Non SenML messages are persisted as binary blobs

Supported Databases for Telemetry:

- InfluxDB
  - Clustering is not open source
- MongoDB
  - Popular among Web developers
- Cassandra DB
  - Linear scalability
  - For multi datacenter deployments



MAINFLUX - Message Flow

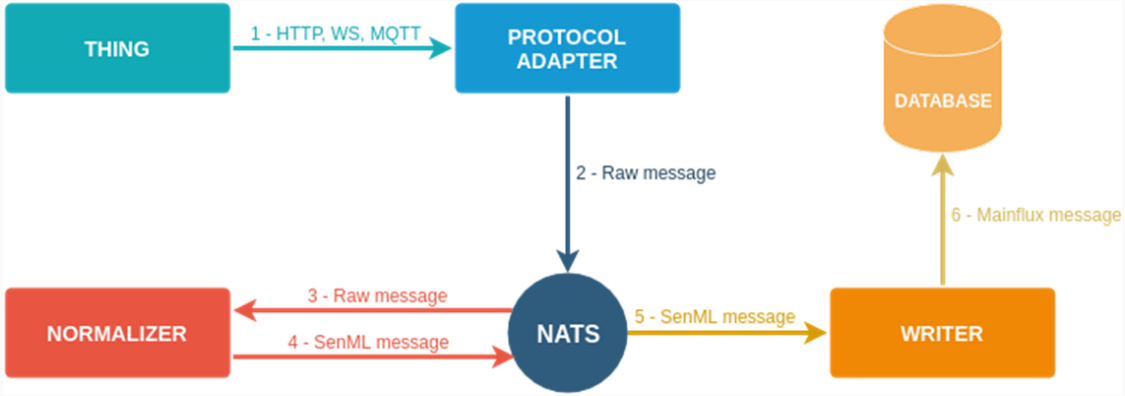
In this example, messages are written to the database.

1. The Thing (device or app) is sending message to the Mainflux IoT Platform.

2. The message is received by the Protocol adapter and forwarded to the NATS Topic as raw message.

3, 4. The Normalizer reads all raw messages, normalizes them and forwards them to normalized topic in the NATS.

5, 6. Database writer microservice, then, reads all the messages and writes them to the database.



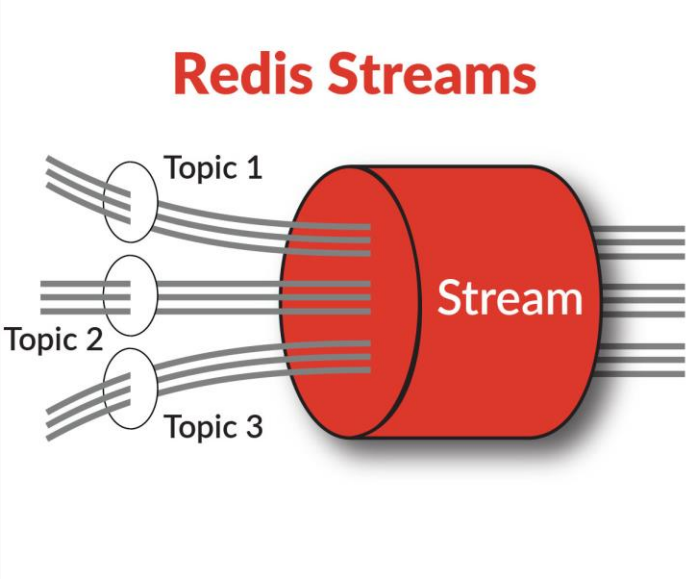
MAINFLUX – Event Sourcing

Event Sourcing

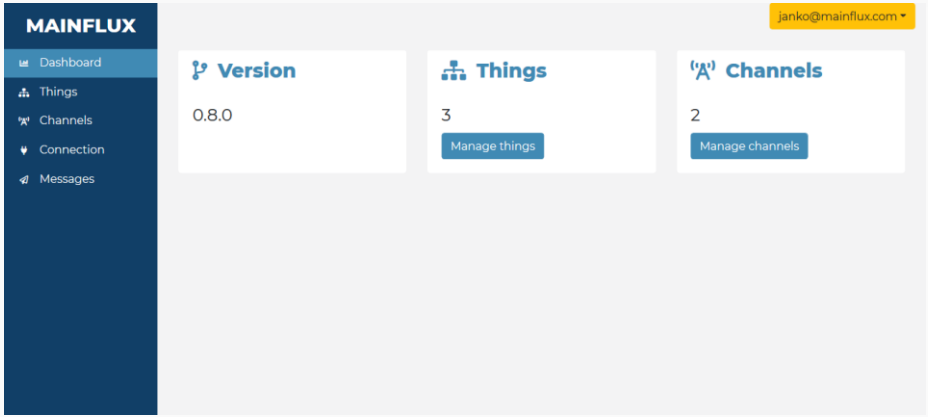
Things Service publishes events to Event Stream
- Create Thing
- Delete Thing
- Etc..

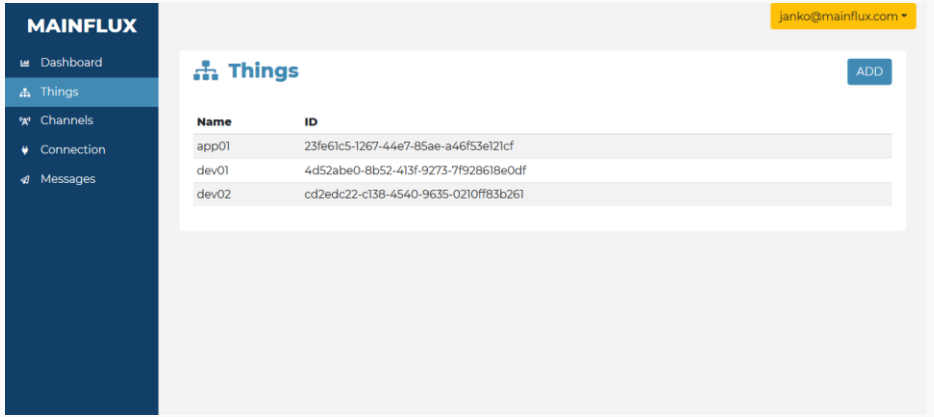
Order of events is preserved
Integration services can use this event stream to do event driven integraton with external systems
Mainflux is using it for:
- LoRa WAN integration
- Bootstraping Service

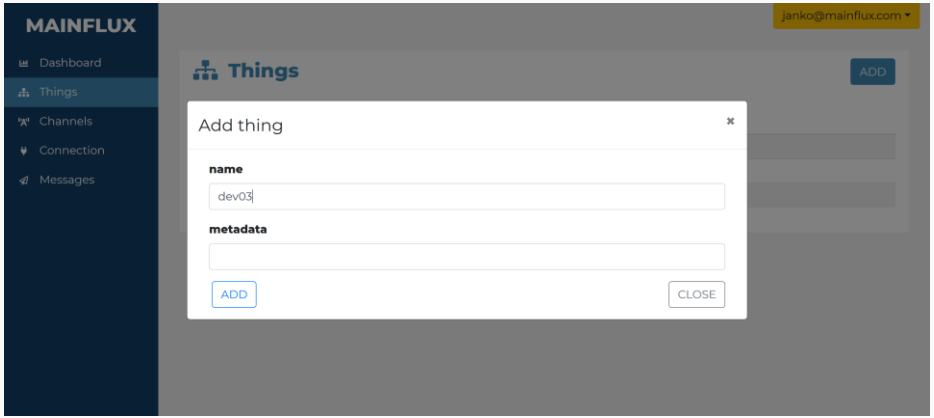
Mainflux is using Redis Streams for Events sourcing

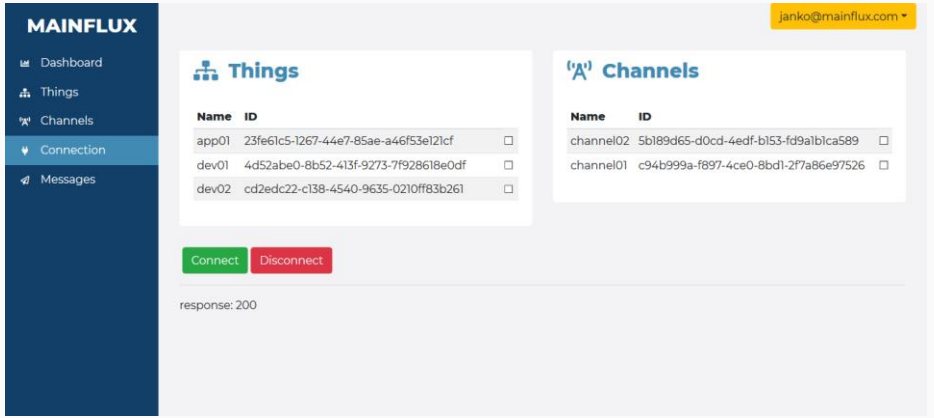


MAINFLUX – UI

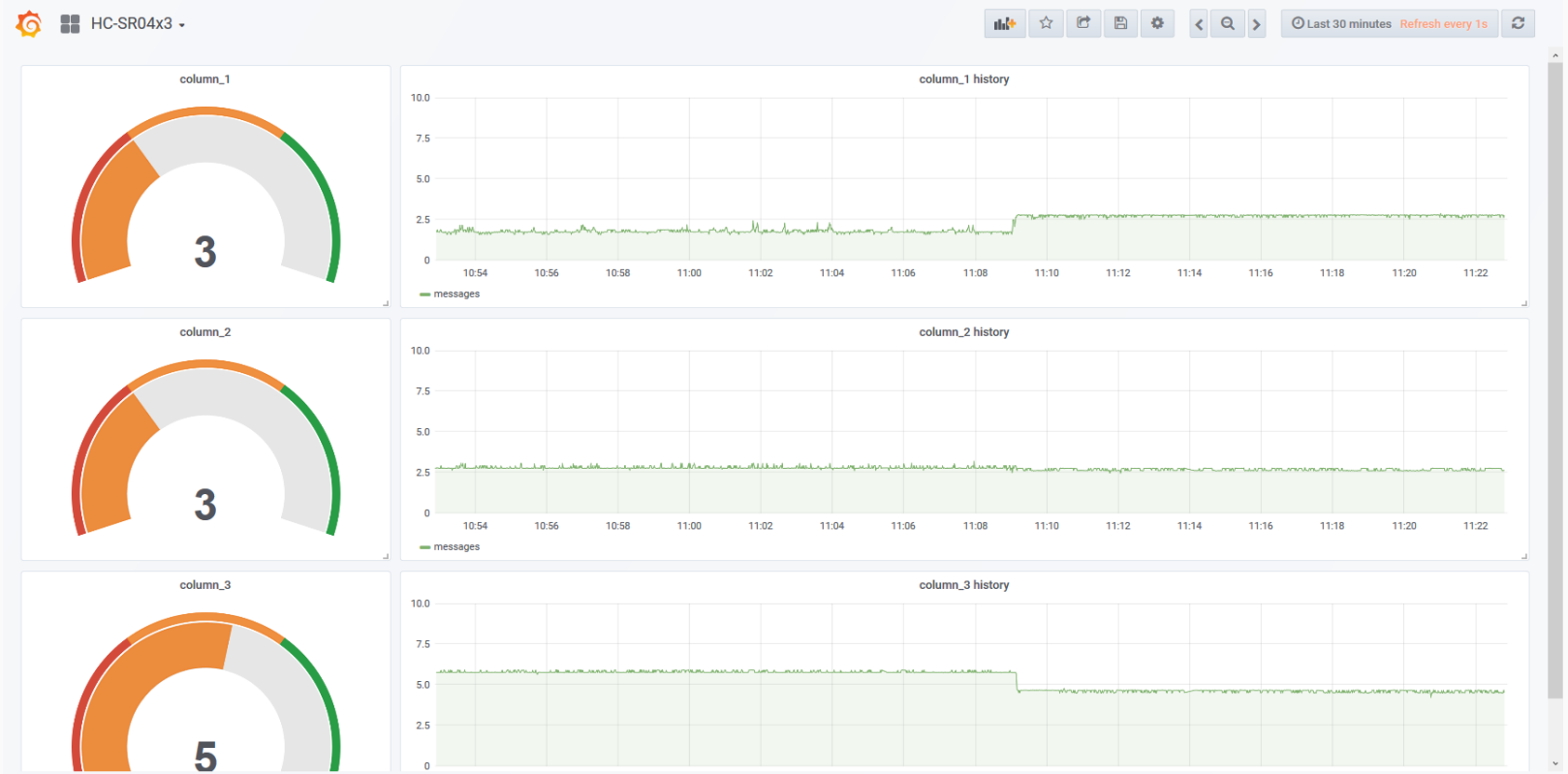

MAINFLUX
Dashboard
Things
Channels
Connection
Messages
janko@mainflux.com
Version
0.8.0
Things
3
Manage things
Channels
2
Manage channels


MAINFLUX
Dashboard
Things
Channels
Connection
Messages
janko@mainflux.com
Things
ADD
Name
ID
app01
23fe61c5-1267-44e7-85ae-a46f53e121cf
dev01
4d52abe0-8b52-413f-9273-7f928618e0df
dev02
cd2edc22-c138-4540-9635-0210ff83b261


MAINFLUX
Dashboard
Things
Channels
Connection
Messages
janko@mainflux.com
Things
ADD
Add thing
name
dev03
metadata
ADD
CLOSE

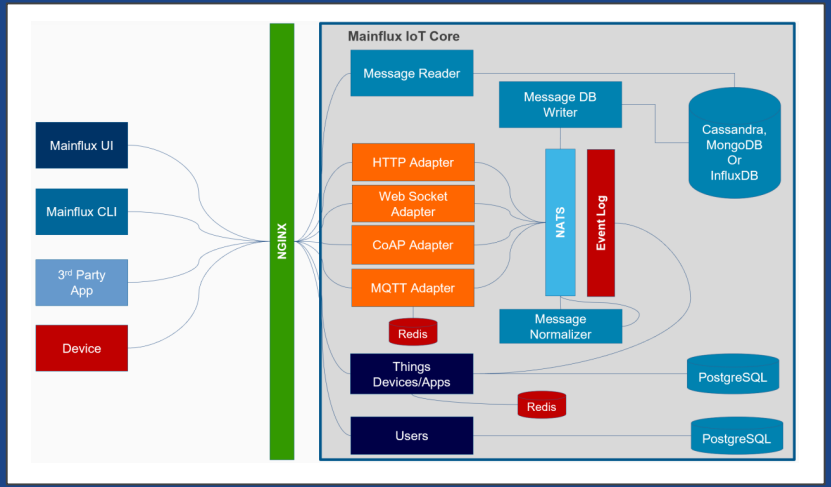

MAINFLUX
Dashboard
Things
Channels
Connection
Messages
janko@mainflux.com
Things
Name
ID
app01
23fe61c5-1267-44e7-85ae-a46f53e121cf
dev01
4d52abe0-8b52-413f-9273-7f928618e0df
dev02
cd2edc22-c138-4540-9635-0210ff83b261
Channels
Name
ID
channel02
5b189d65-d0cd-4edf-b153-fd9a1b1ca589
channel01
c94b999a-f897-4ce0-8bd1-2f7a86e97526
Connect
Disconnect
response: 200

MAINFLUX – Message Visualization Using Grafana



MAINFLUX

Integration Points



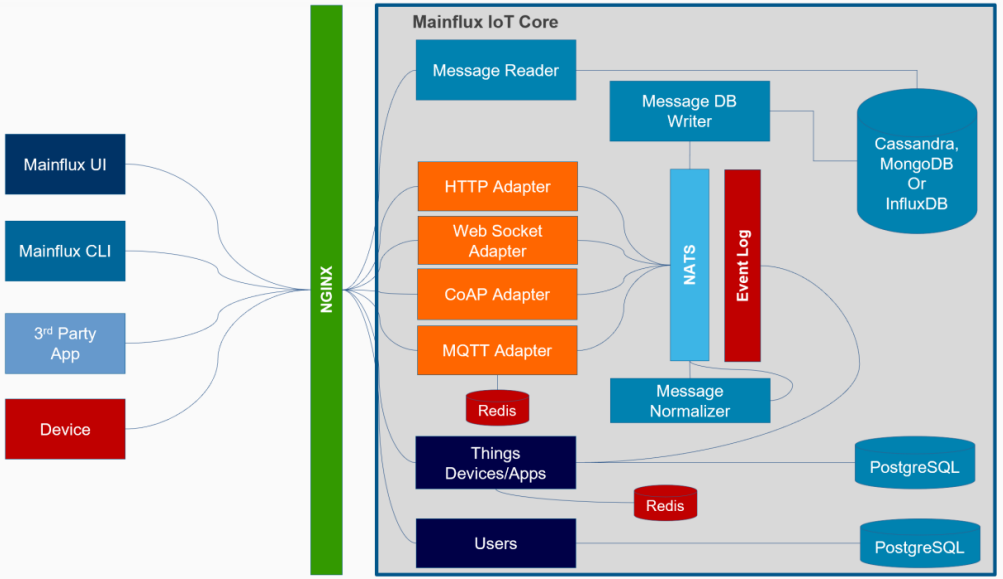
MAINFLUX - Extension/Integration Points

Additional Protocols

- New protocol adapter needs to be developed. It should receive messages, authorize against Things service using gRPC and push authorized messages to channel on NATS

Additional Message Formats

- New microservice needs to be developed. It would replace Message normalizer or work in parallel with it if multiple message formats are required



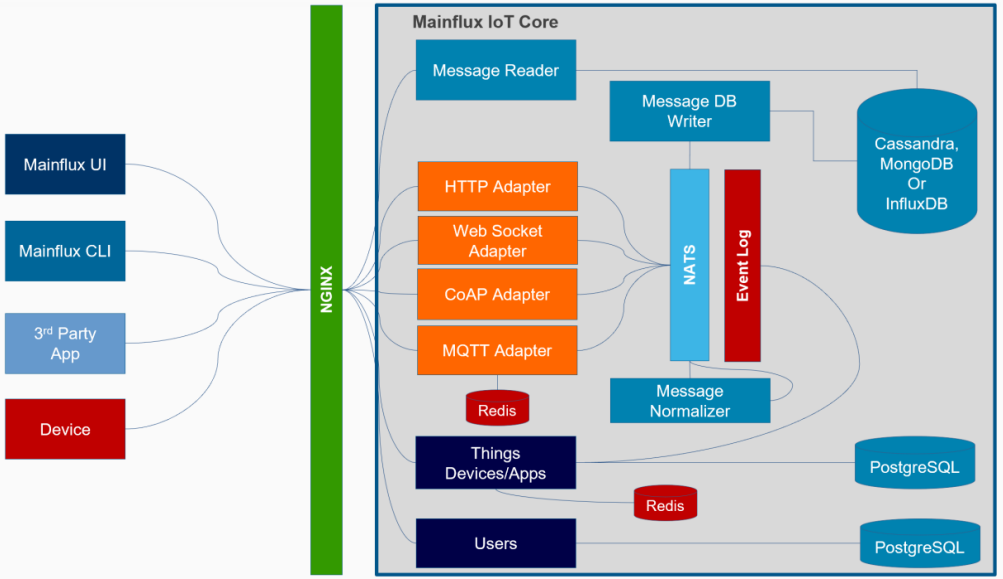
MAINFLUX - Extension/Integration Points

Additional Message persistence Database

- Message DB writer needs to be developed. It would read the messages from NATS and write them to the database.

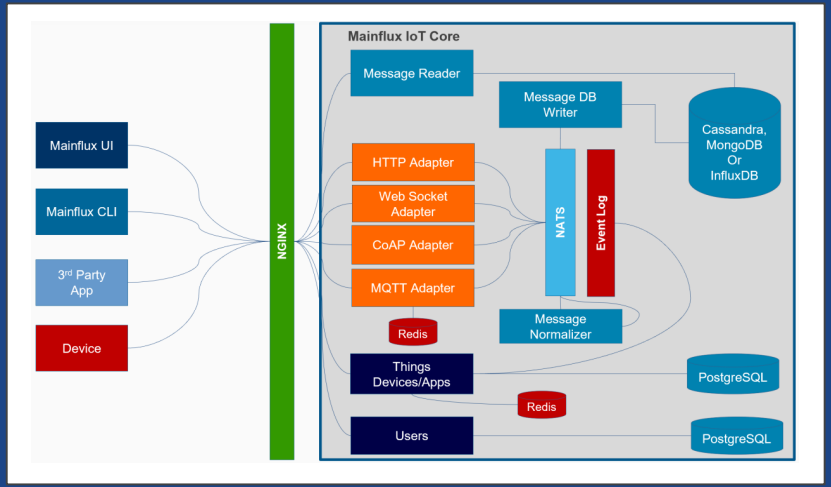
Event Sourcing Stream

- Things Service pushes events like: device created, device deleted, etc. to the Redis Stream. The clients connect to the stream and can read the events from the bus and act on the events.



MAINFLUX

Deployment Options



MAINFLUX - Deployment Strategies

Mainflux can be deployed:

- **On Premise**
  - Single Instance
  - Kubernetes Cluster
- **In the Cloud (Cloud Agnostic)**
  - Any Cloud which supports docker containers and or Kubernetes
- **On the Edge** - Even on RPi Class Device

- **Native Deployment**
  - Compiled for Intel or ARM
  - Compiled for Windows, Linux or Mac
- **Docker Containers**
  - Docker Compose script is in Github
- **Kubernetes Deployment**
  - Kubernetes scripts are in Github

MAINFLUX - Monitoring, Logging and Performance Metrics

Prometheus

- Go Kit for Metrics
- Autoscale Mainflux kubernetes deployments using Prometheus API

```
mainflux-mqtt    | {"level":20,"time":1541712839922,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"$SYS/O6viLrVX7/heartbeat","qos":0,"retain":false},"v":1}
mainflux-things | {"level":"info","message":"Method can_access for channel 1 and thing 2 took 3
37.996µs to complete without errors.","ts":"2018-11-08T21:34:35.165760434Z"}
mainflux-http   | {"level":"info","message":"Method publish took 32.598µs to complete without e
rrors.","ts":"2018-11-08T21:34:35.166118963Z"}
mainflux-normalizer | {"level":"info","message":"Method normalize took 203.862µs to complete wi
thout errors.","ts":"2018-11-08T21:34:35.166874197Z"}
mainflux-nginx   | 172.19.0.1 - - [08/Nov/2018:21:34:35 +0000] "POST /http/channels/1/messages H
TTP/1.1" 202 0 "-" "curl/7.61.0"
mainflux-mqtt    | {"level":20,"time":1541712875169,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"channels/1/messages","qos":2,"retain":false},"v":1}
mainflux-mqtt    | {"level":20,"time":1541712899923,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"$SYS/O6viLrVX7/heartbeat","qos":0,"retain":false},"v":1}
mainflux-mqtt    | {"level":20,"time":1541712959926,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"$SYS/O6viLrVX7/heartbeat","qos":0,"retain":false},"v":1}
mainflux-mqtt    | {"level":20,"time":1541713019928,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"$SYS/O6viLrVX7/heartbeat","qos":0,"retain":false},"v":1}
mainflux-mqtt    | {"level":20,"time":1541713079930,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"$SYS/O6viLrVX7/heartbeat","qos":0,"retain":false},"v":1}
mainflux-mqtt    | {"level":20,"time":1541713139931,"msg":"published","pid":1,"hostname":"692d72
f1de6a","message":{"topic":"$SYS/O6viLrVX7/heartbeat","qos":0,"retain":false},"v":1}
mainflux-mqtt    | {"level":20,"time":1541713199934,"msg":"published","pid":1,"hostname":"692d72
```

MAINFLUX – Grafana Performance Dashboards

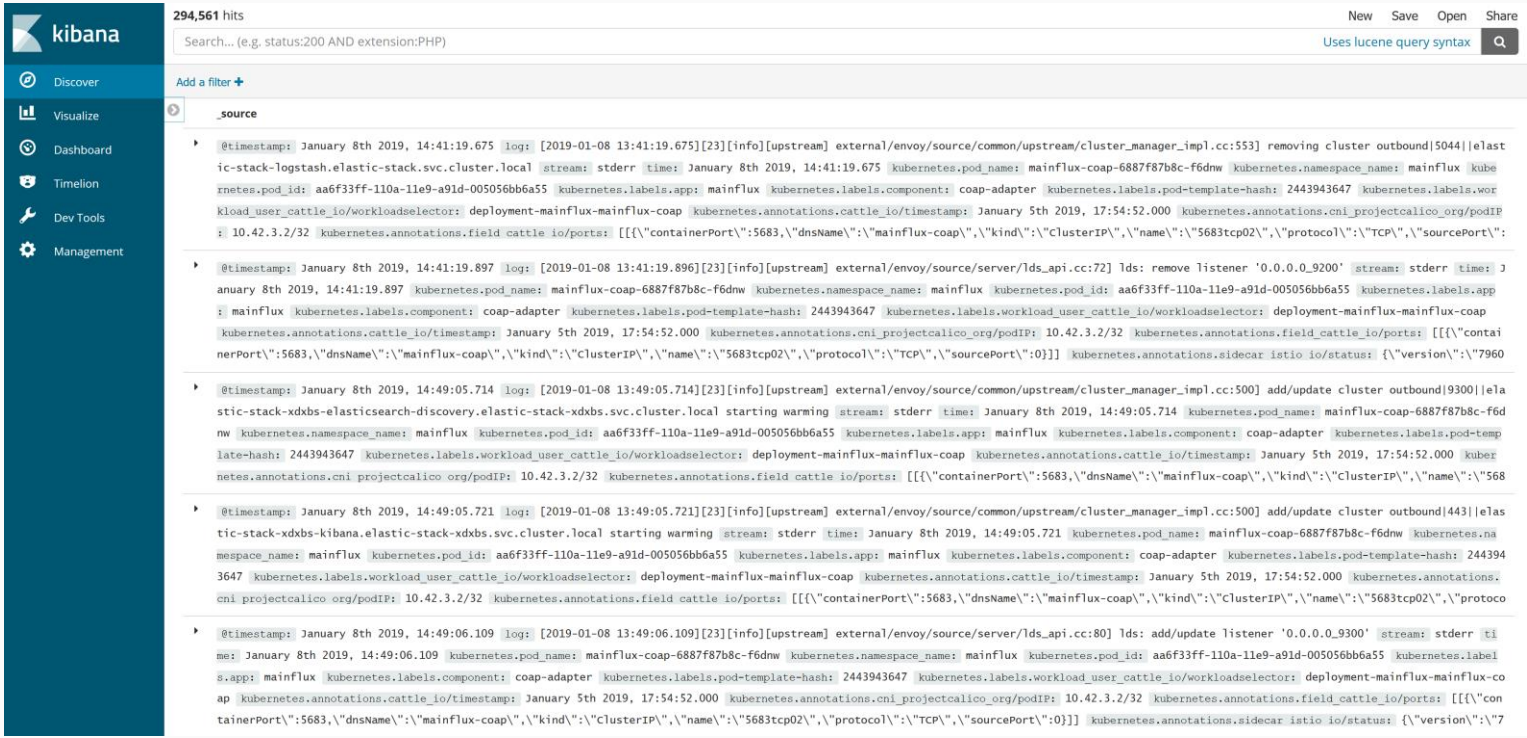
MAINFLUX - Monitoring, Logging and Performance Metrics

EKL Stack - Logs and Troubleshooting

- Elasticsearch
- Kibana
- Logstash

or EFK Stack -

- Elasticsearch
- Fluent bit
- Kibana



THANK YOU!

www.mainflux.com
info@mainflux.com