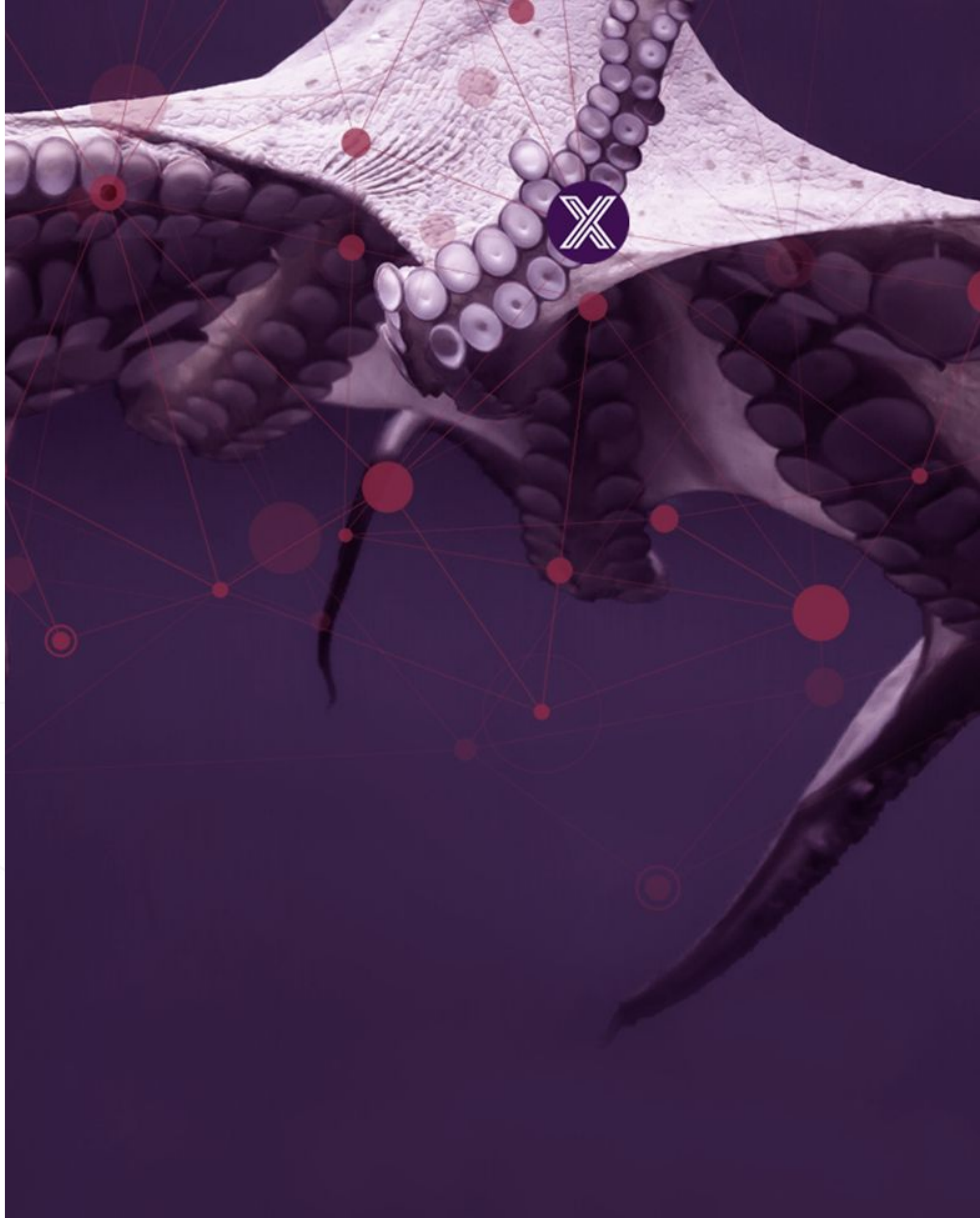
EDGE X FOUNDRY™

Application Services Design

Application Working Group
2018-10-09



LF Antitrust Policy Notice

- EdgeX Foundry meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.
- Examples of types of actions that are prohibited at EdgeX Foundry meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

Meeting Logistics

- Time: October 9, 2018 11am PDT – 12am PDT

Join from PC, Mac, Linux, iOS or Android: https://zoom.us/j/611544838

Or iPhone one-tap :

US: +16465588656,,611544838#

or +16699006833,,611544838#

Or Telephone:

Dial(for higher quality, dial a number based on your current location):

US: +1 646 558 8656 or +1 669 900 6833

or +1 855 880 1246 (Toll Free)

or +1 877 369 0926 (Toll Free)

Meeting ID: 611 544 838

International numbers available: https://zoom.us/u/aoLL4E9yo

Today's Agenda

- Message Bus vs FAAS/Lambda/Serveless Technology

Message Bus

- Generally standardized; avoiding vendor lock in - many provider
- Services publish/consume messages
- Always on technology; single scale
- Scale at the endpoints
- Plenty of language bindings
- Time trusted technology
- Testing/debugging at the endpoints

FAAS/Lambda/Serveless Technology

- 3rd party / vendor provided – lock in concerns
- Language runtimes may be limited
  - Many are built in Go
  - Typically packaged as Docker container
- Micro-microservices are brought into FAAS provider when needed
  - Only on when needed; scale per function need
- Early adopter technology; largely cloud platform based
- No standardized security built in
- Can be hard to test/debug
- Not ready for edge (size, deployment options, etc.)

Message Bus Options - Message Brokers

Name	Licence	Size	Prd. Ready	Lang.	Client	OS Support
Mosquitto	EPL/EDL	very lightweight	Yes	C		Linux, Win, Mac
Paho	EPL-1.0	lightweight	Yes	C	C, Go, Java, ...	Linux, Win, Mac (client)
Apache ActiveMQ	Apache License 2.0		Yes	Java	Java, C, C++, C#, ...	Linux, Win, Mac
Apache Apollo						
RabbitMQ	MPL 1.1.	lightweight	Yes	Erlang	Erlang,Java, .NET, PHP, Python, JavaScript, Ruby, Go	Linux, Win, Mac
Qpid	Apache License 2.0			J, C++		Linux, Win, Mac
NATS	Apache License 2.0	lightweight	Yes	Go		Linux, Win, Mac

EDGE X FOUNDRY™

Message Bus Options - No Message Brokers

Name	Version	Size	Supported Lang
0MQ			
Nanomsg			
GRPC			
Thrift			

FAAS Options

- Need non-Cloud provider
- OpenFaas
- OpenWhish
- Kubeless
- IronFunctions
- nuclio

Recommendation

- FAAS/Serveless probably too early
- Not stable yet
- Not edge ready
- Can we build functions in a way that supports FAAS in the future
- Architect the service functions small and well defined so that some day they could be moved to a FAAS framework with little work

EDGE X FOUNDRY™

Application Service Functions

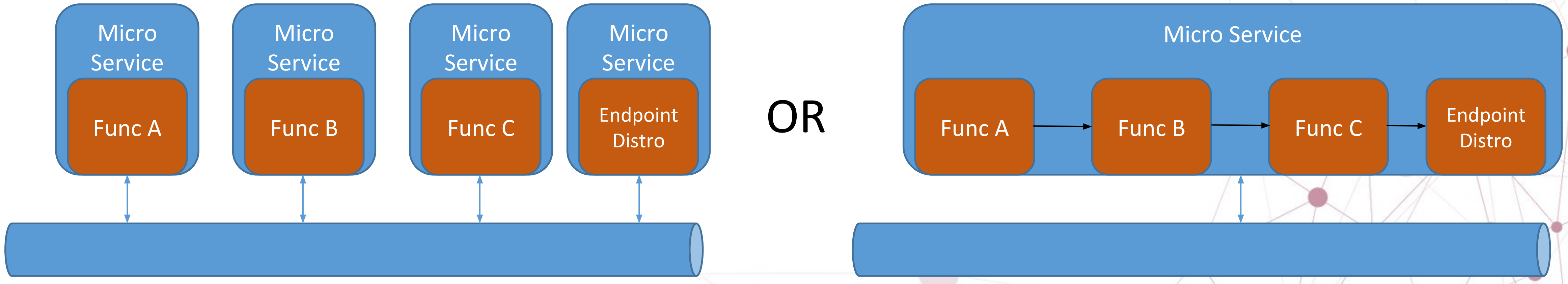
Application Services Functions

- Essentially an EAI engine
- Operations
  - Filter (only give me readings from device A; only give me readings regarding temperature, …)
  - Validator (device ID, reading against value descriptor, …)
  - Transformation (convert C to F values, convert CBOR to Protobuf, …)
  - Enrich (add device metadata to reading, …)
  - Format (JSON, XML, CSV, …)
  - Encrypt (really different kind of transformation)
  - Compress (really different kind of transformation)
  - Custom (black box that you define what you want to happen inside)

Application Services Functions

- Endpoints
  - HTTP(s)
  - MQTT(s)
  - AMQP
  - XMPP
  - WebSockets
  - CoAP
- Cloud providers
  - Azure IoT
  - AWS IoT
  - Google Cloud IoT
  - …

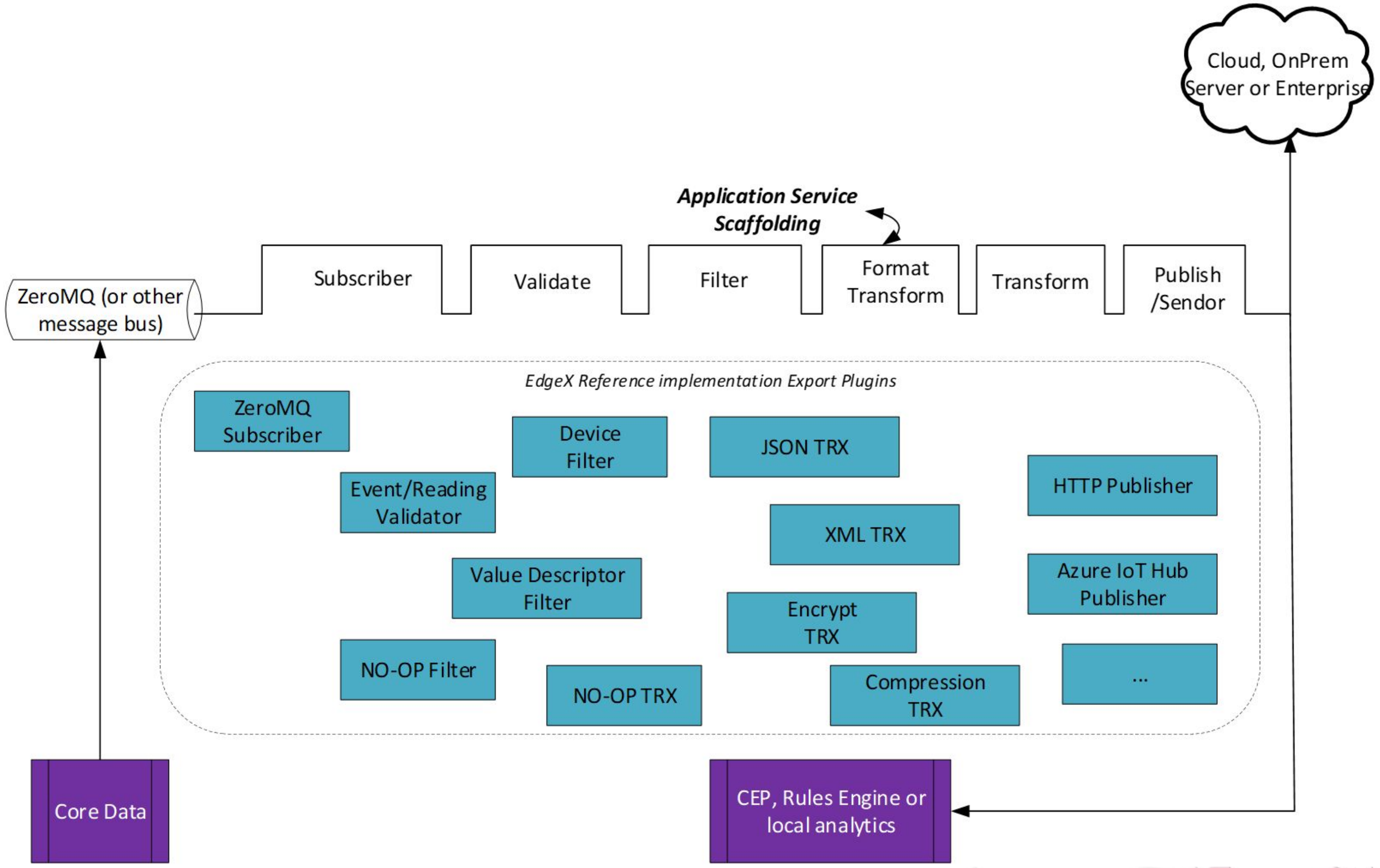
Implement by functions or by service?



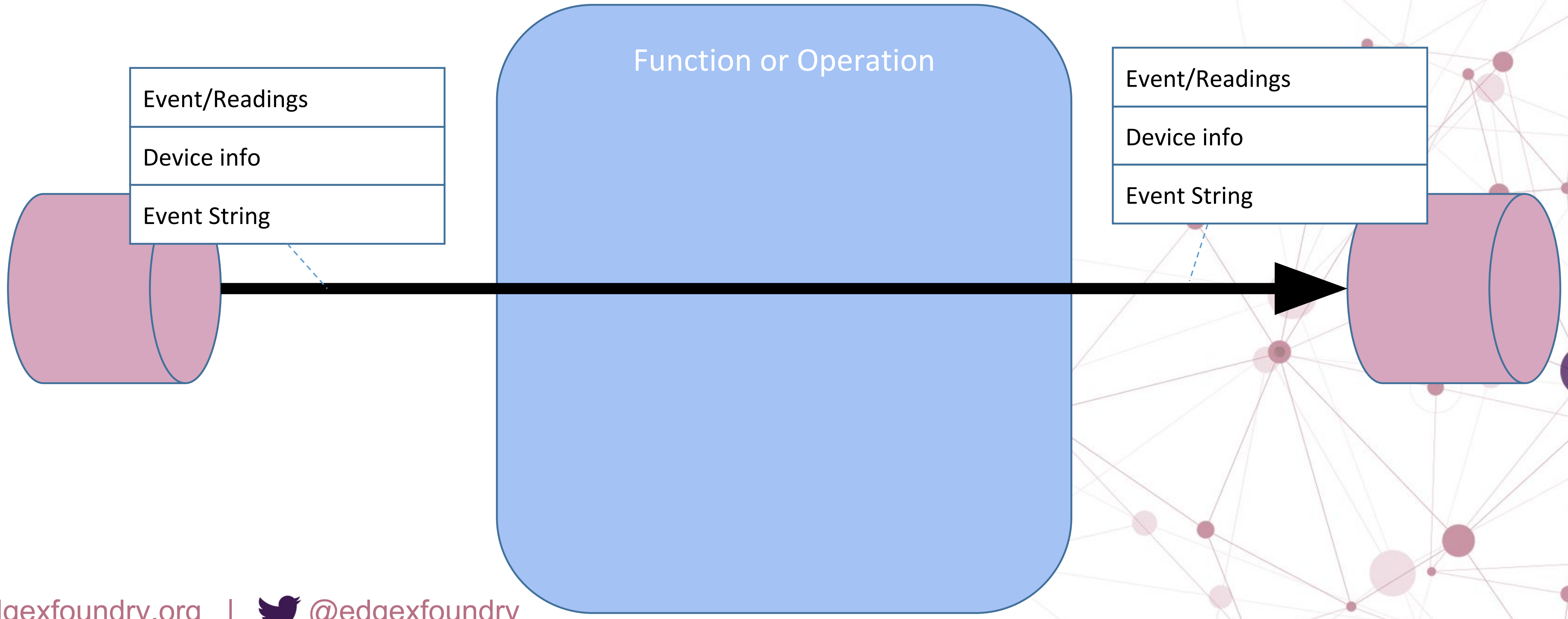
How to orchestrate per client?
How to secure?
Functions can be reused across clients

Internally secure
Internally orchestrated
Per client micro service
Duplicate code in each service

What provides the "scaffolding"?



Application Services Exchange


Event/Readings
Device info
Event String
Function or Operation
Event/Readings
Device info
Event String