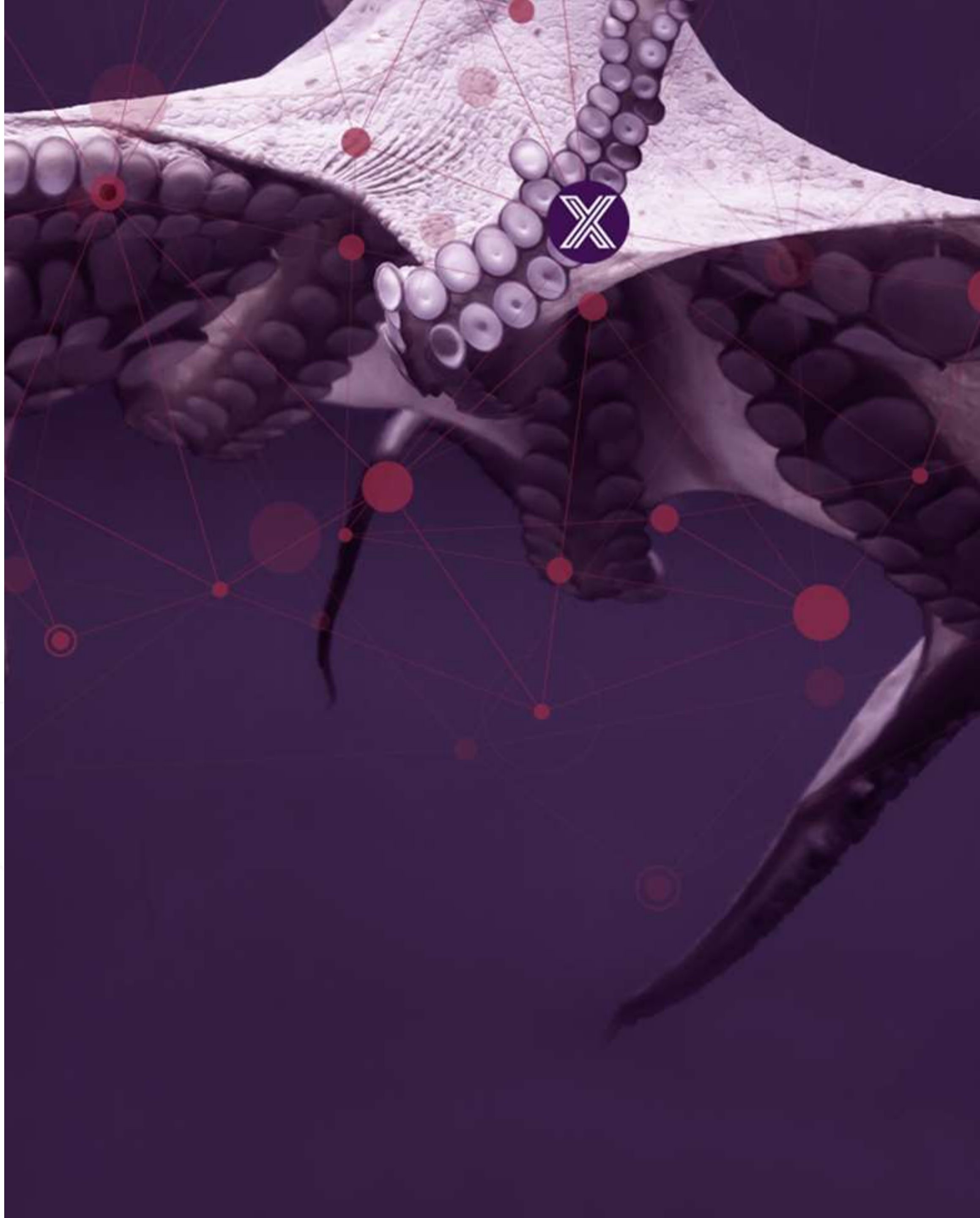
EDGEXFOUNDRY™

Getting Lean and Distributed at the Edge

Jim White

Dell Technologies

October 2018



edgexfoundry.org | @edgexfoundry

Agenda

- 2 minute quick intro to EdgeX Foundry
- EdgeX architecture
- Requirements of an edge platform
- EdgeX performance metrics
  - In the beginning
  - Today
  - Keys to the diet
- Distributing EdgeX micro services
  - Why distribution matters

About Me

- Jim White (james_white2@dell.com)
  - Dell Technologies IoT Solutions Division – Distinguished Engineer
  - Team Lead of the IoT Platform Development Team
  - Chief architect and lead developer of Project Fuse
    - Dell's original IoT platform project that became EdgeX Foundry
    - Yes – I wrote the first line(s) of code for EdgeX (apologies in advance)
  - EdgeX Foundry …
    - Vice Chairman, Technical Steering Committee
    - Systems Management Working Group Chair
    - Ad hoc and unofficial lead architect

EdgeX Foundry

open source™ +

THE LINUX FOUNDATION + THE APACHE™ SOFTWARE FOUNDATION 2.0

UNIX suse

intel ARM

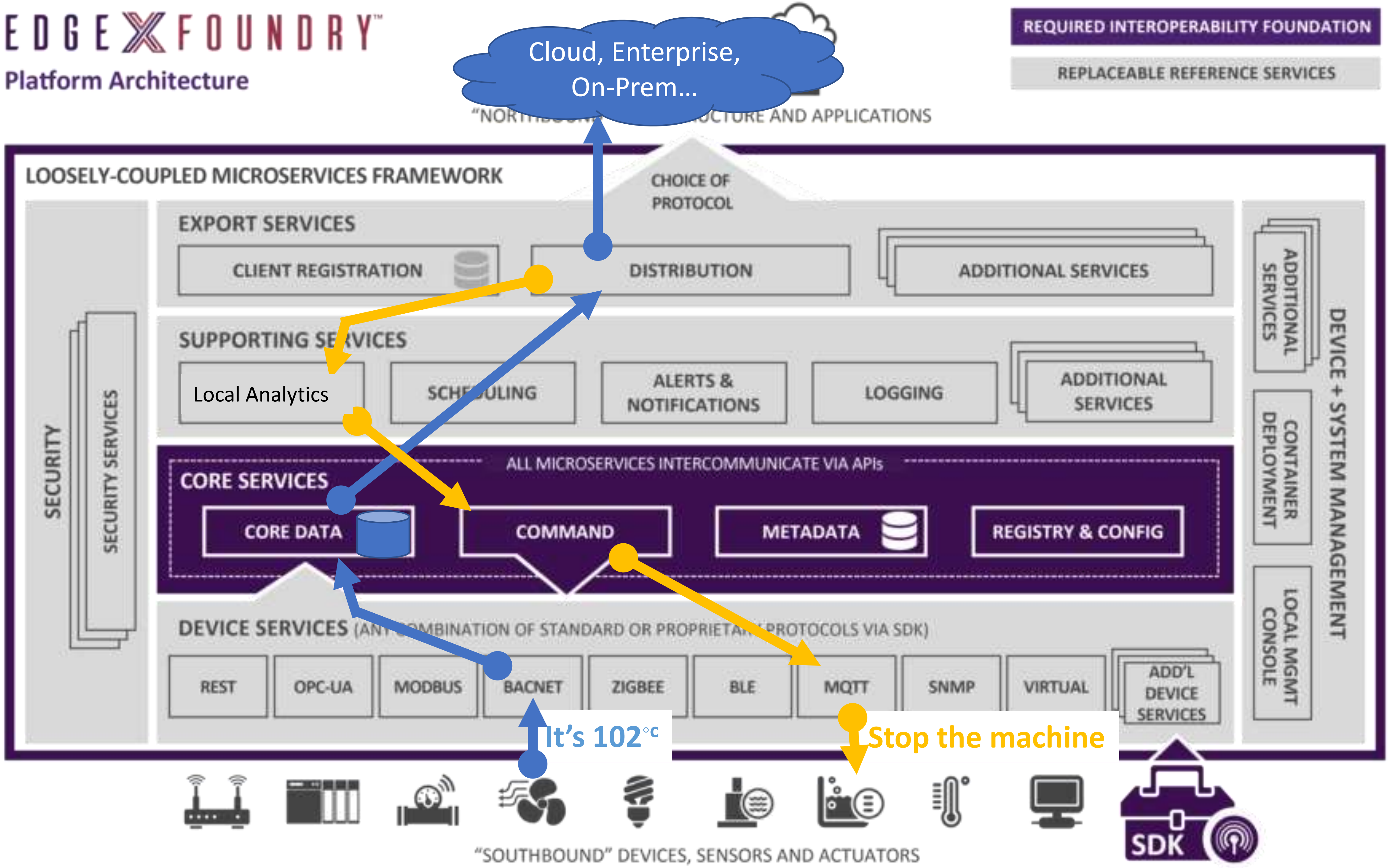
Backed by Industry Leaders

ABSOLUTE
aicas realtime
Alleantia
AMD
ANALOG DEVICES
AXZON
BASKING
Beechwoods
ubuntu Delivered by Canonical
ULL
CAVIUM
CertusNet
CIVIL INFRASTRUCTURE PLATFORM
CLOUD FOUNDRY
CloudPlugs Connecting Things
Cumulocity A SOFTWARE AG COMPANY
DELL EMC
DEVICE AUTHORITY Trust for every Thing
enigmedia
enocean
EPISENSOR
ETRI Electronics and Telecommunications Research Institute
FIWARE FOUNDATION
FOGHORN
FORGEROCK
IMS EVOLVE
industrial internet CONSORTIUM
inwinSTACK
IOTech
IoTium www.iotium.io
ISSAT
Mateur
KMC CONTROLS
Linaro
MAINFLUX Internet of Things Solutions
MOCANA
modius
NETFOUNDRY
neustar
OMG OBJECT MANAGEMENT GROUP
OPTO 22
Project Haystack
Queen's University Belfast
redislabs home of redis
REPLY CONCEPT
RIOT
Rubicon
SAMSUNG
SAMSUNG
SAMSUNG SDS
Sixgill Govern IoE
striim
SWITCH AUTOMATION
THALES
twobulls
ULE ALLIANCE
V5 SYSTEMS
vmware
Volterra EDGE SERVICES
万向集团 WANXIANG GROUP
xage SECURITY
ZEDEDA
Zephyr
ZingBox

With more in process!

EdgeX Foundry Goals

- Common open platform unifying edge computing
- Create an ecosystem of interoperable plug-and-play components
- Certify to ensure interoperability and compatibility
- Provide tools to create IoT edge solutions
- Collaborate across the IoT



EDGE X FOUNDRY™

Platform Architecture

REQUIRED INTEROPERABILITY FOUNDATION

REPLACEABLE REFERENCE SERVICES

Cloud, Enterprise, On-Prem…

"NORTHBOUND" INFRASTRUCTURE AND APPLICATIONS

LOOSELY-COUPLED MICROSERVICES FRAMEWORK

CHOICE OF PROTOCOL

SECURITY

SECURITY SERVICES

EXPORT SERVICES

CLIENT REGISTRATION

DISTRIBUTION

ADDITIONAL SERVICES

SUPPORTING SERVICES

Local Analytics

SCHEDULING

ALERTS & NOTIFICATIONS

LOGGING

ADDITIONAL SERVICES

ALL MICROSERVICES INTERCOMMUNICATE VIA APIs

CORE SERVICES

CORE DATA

COMMAND

METADATA

REGISTRY & CONFIG

DEVICE SERVICES (ANY COMBINATION OF STANDARD OR PROPRIETARY PROTOCOLS VIA SDK)

REST | OPC-UA | MODBUS | BACNET | ZIGBEE | BLE | MQTT | SNMP | VIRTUAL | ADD'L DEVICE SERVICES

ADDITIONAL SERVICES

CONTAINER DEPLOYMENT

LOCAL MGMT CONSOLE

DEVICE + SYSTEM MANAGEMENT

It's 102°C

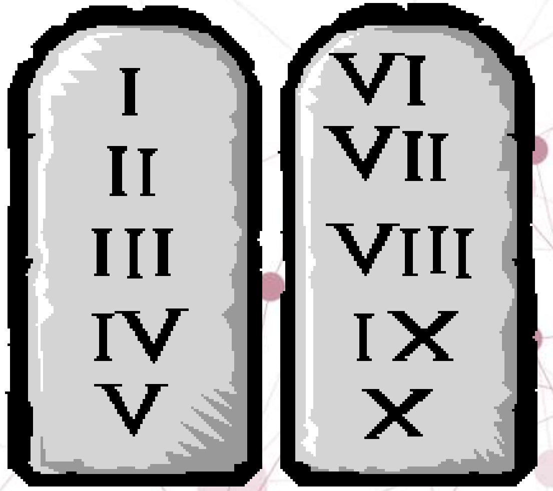
Stop the machine

SDK

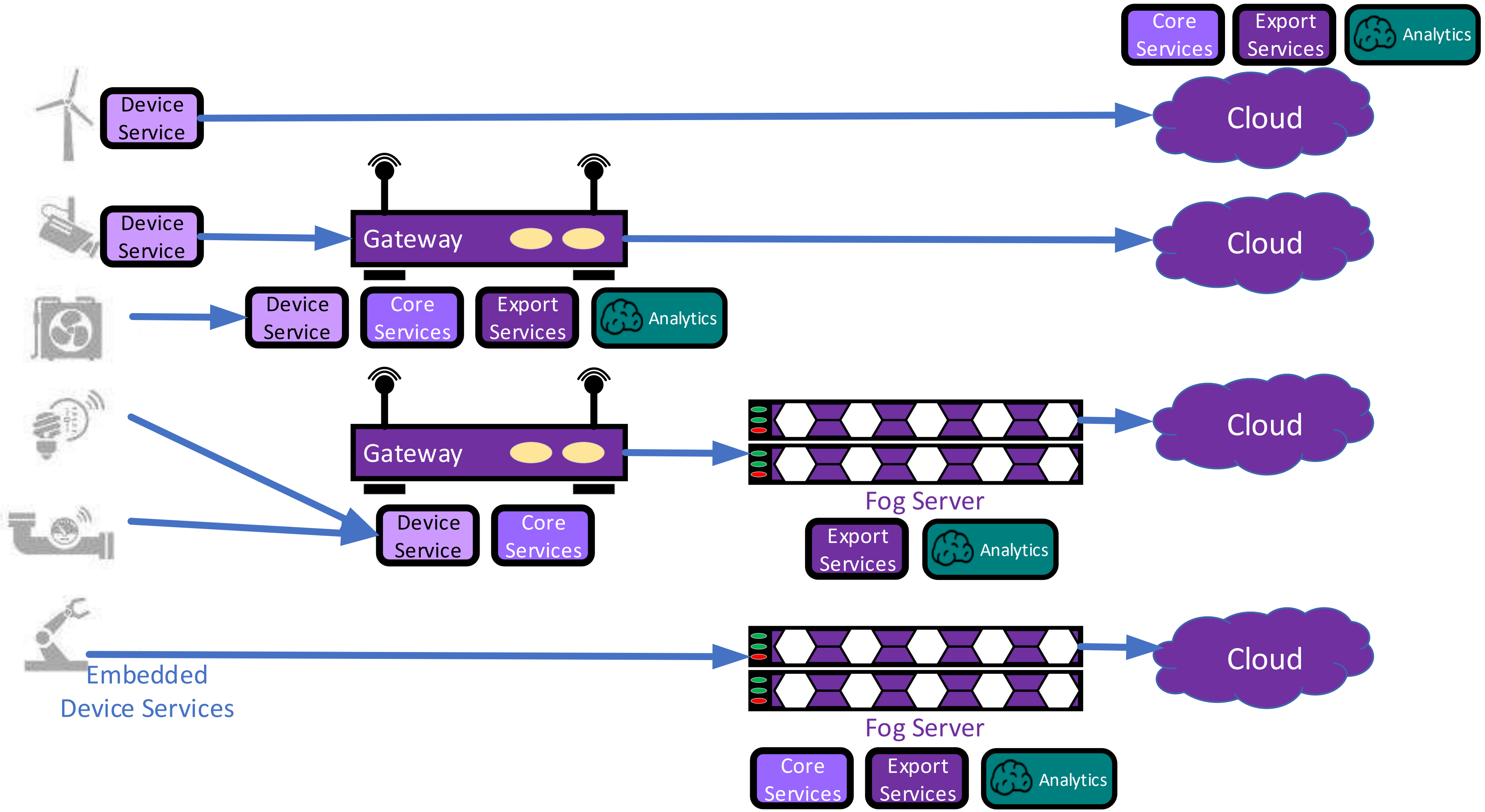
"SOUTHBOUND" DEVICES, SENSORS AND ACTUATORS

EdgeX Architectural Tenets

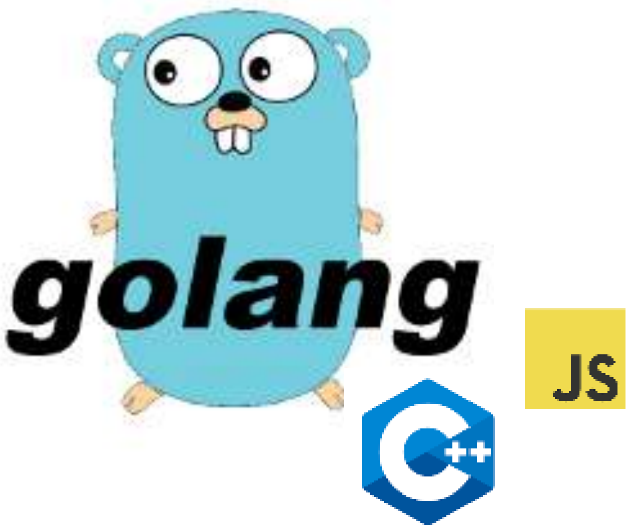
- Platform agnostic
- Extremely flexible
- Encourages best of breed solutions
- Store and forward
- Facilitate “intelligence” moving closer to the edge
- Support brown and green devices/sensors
- Must be secure and easily managed

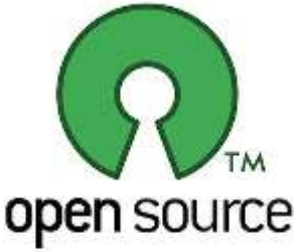


EdgeX Enables Tiered Fog Deployments



EdgeX Technology





Key Project Links

https://github.com/edgexfoundry

https://docs.edgexfoundry.org/

https://wiki.edgexfoundry.org/display/FA/EdgeX+Tech+Talks

https://www.edgexfoundry.org/news/blog/

https://lists.edgexfoundry.org/mailman/listinfo

https://chat.edgexfoundry.org/home

https://www.edgexfoundry.org/about/members/join/

https://www.linkedin.com/company/edgexfoundry/

https://twitter.com/EdgeXFoundry

https://www.youtube.com/edgexfoundry

Requirements of an Edge Platform

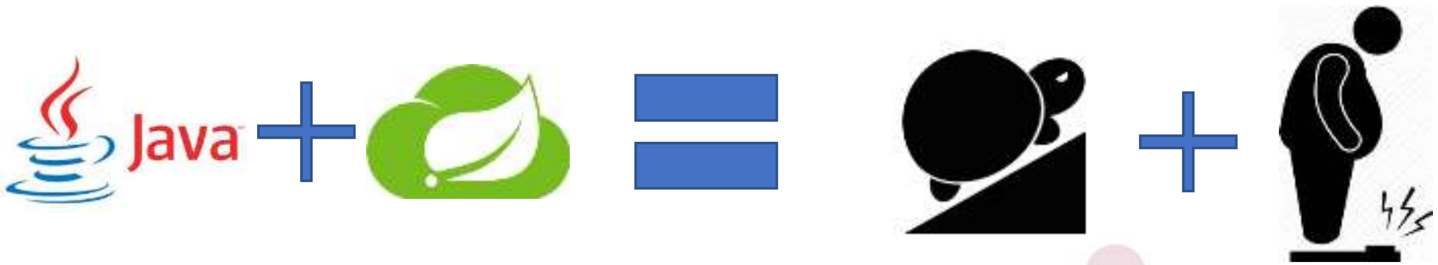
- Near real-time
- Sensor sampling rates of 100-1000/second
- Handling kilobytes per message
- Latency < 1 second from sensor to analytics to actuation

Typical Collection or Sampling Rates

Use Case	Sampling Rate	Order of Volume
Electric Grid Line Fault Detection	Micro second	Bytes, KBs
Vibration Sensor	Micro second to seconds	Bytes, KBs
Pressure Sensor	Micro second to seconds	Bytes, KBs
PLC	Micro second to seconds	Bytes, KBs
CNC Machines	Micro second to seconds	KBs, MBs
CANBus	Micro second to seconds	MB
Building Automation	Micro second to seconds	Bytes, KBs
Locomotive Telemetry	Micro second to seconds	KBs, MBs
RTU in Utility	Micro second to seconds	KBs, MBs
Oil/Gas	Micro second to seconds	KBs
RFID	Micro second to seconds	Bytes, KBs

EdgeX in the beginning - April 2017

- EdgeX started as a Dell POC in 2015



- 7-5 GB RAM
- ~50% CPU utilization (2CPU, 8GB gateway)
  - Pegged at 100% at startup
- Startup > 30 seconds/service
  - System startup ~5 minutes

EdgeX @ the start (Hannover Messe 2017)

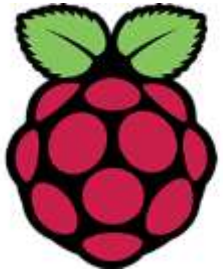
Service	Container size (in MB)	RAM (in MB)
Metadata	165	301
Coredata	170	299
Command	159	244
Logging	158	249
Scheduler	161	215
Notifications	153	222
Export Client	164	240
Export Distro	168	291
Rules Engine	177	270
SNMP DS	169	265
Fishertech DS	169	281
Bacnet DS	437	278
BLE DS	168	301
UI	607	50
Mongo	402	99

CPU % - 34%
Mem: 5.1 of 7.7G used
Swap: 168M of 7.9G used

Performance Targets set Oct 2017

- 2nd EdgeX Technical Steering Committee Face to Face Decision

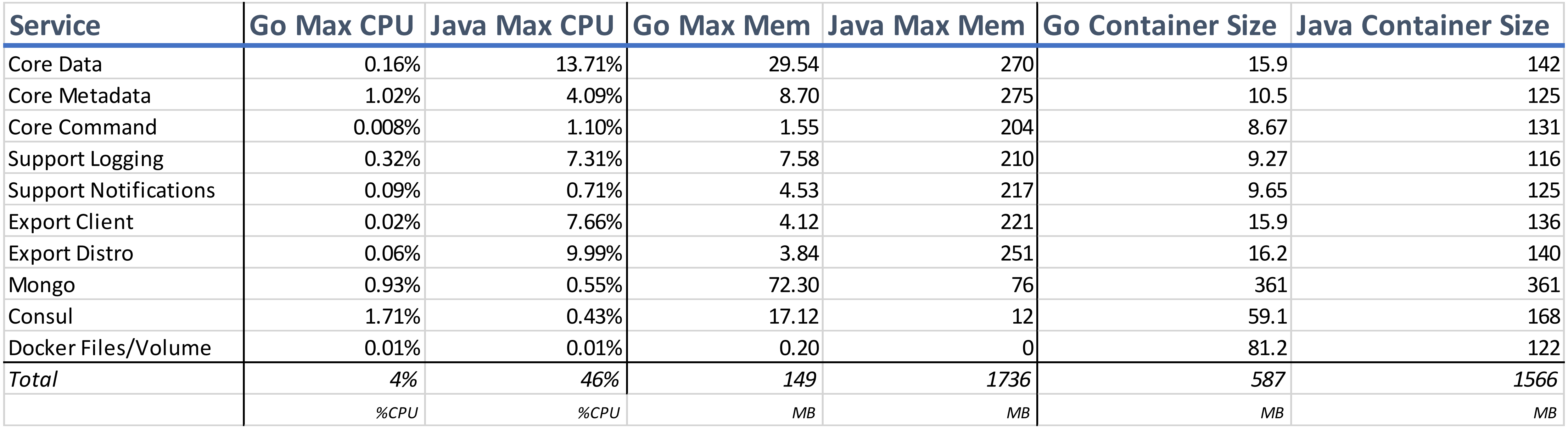
- Run in < 1GB RAM
- Use < 25% CPU
- Use < 32GB Storage
- Startup < 10 seconds
- Latency < 1 second (ingestion to actuation)



Early Go v. Java Experiment (October 2017)

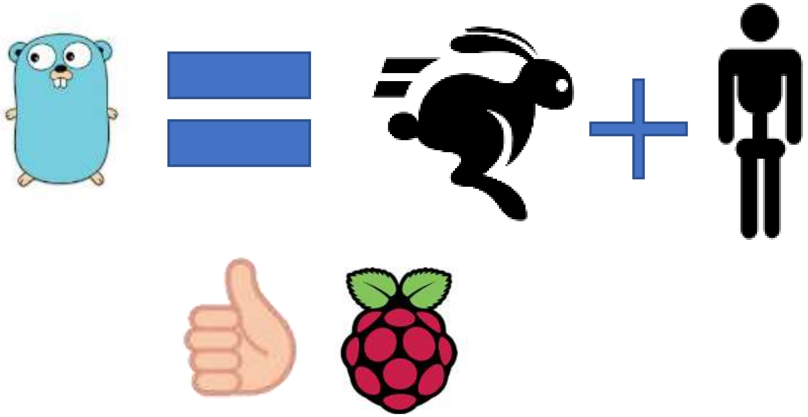
Measure	***Go***	***Java***
Executable/JAR Footprint	11.7 MB	42.4 MB
Container Footprint	**16.2 MB**	**165 MB**
Memory Usage (On Startup)	**4.3 MB**	**221 MB**
Memory Usage (Under Load)	**9.2 MB**	**230 MB**
CPU Usage (Steady State)	0.15%	0.30%
CPU Usage (Under Load)	5.0% - 15.0%	6.0% - 15.0% **(spiked at 90% for heavy load)**
Startup Speed	**0.14 Seconds**	**12.55 Seconds**
Response Speed (Ping)	0.0011 Seconds	0.0022 Seconds
Response Speed (Post)	0.0091 Seconds	0.0137 Seconds
Response Speed (Get)	0.0038 Seconds	0.0062 Seconds

EdgeX Today (Oct 2018 – Delhi release)



Service	Go Max CPU	Java Max CPU	Go Max Mem	Java Max Mem	Go Container Size	Java Container Size
Core Data	0.16%	13.71%	29.54	270	15.9	142
Core Metadata	1.02%	4.09%	8.70	275	10.5	125
Core Command	0.008%	1.10%	1.55	204	8.67	131
Support Logging	0.32%	7.31%	7.58	210	9.27	116
Support Notifications	0.09%	0.71%	4.53	217	9.65	125
Export Client	0.02%	7.66%	4.12	221	15.9	136
Export Distro	0.06%	9.99%	3.84	251	16.2	140
Mongo	0.93%	0.55%	72.30	76	361	361
Consul	1.71%	0.43%	17.12	12	59.1	168
Docker Files/Volume	0.01%	0.01%	0.20	0	81.2	122
Total	*4%*	*46%*	*149*	*1736*	*587*	*1566*
	%CPU	*%CPU*	*MB*	*MB*	*MB*	*MB*

EdgeX Delhi Release

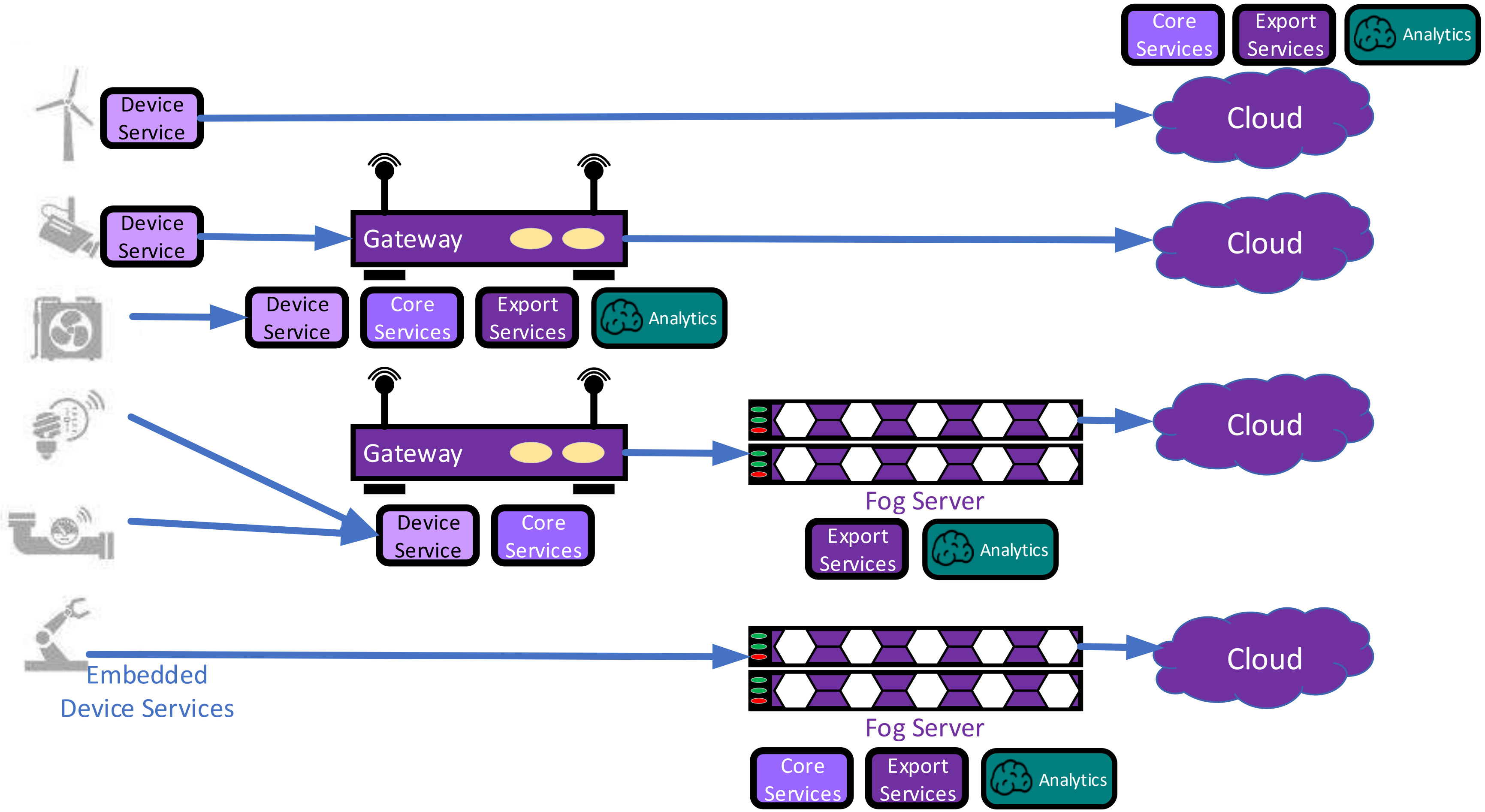


- > 90% memory reduction
- > 90% CPU reduction
- > 80% footprint reduction
- > 95% startup time reduction
  - Startup ~ 5 seconds

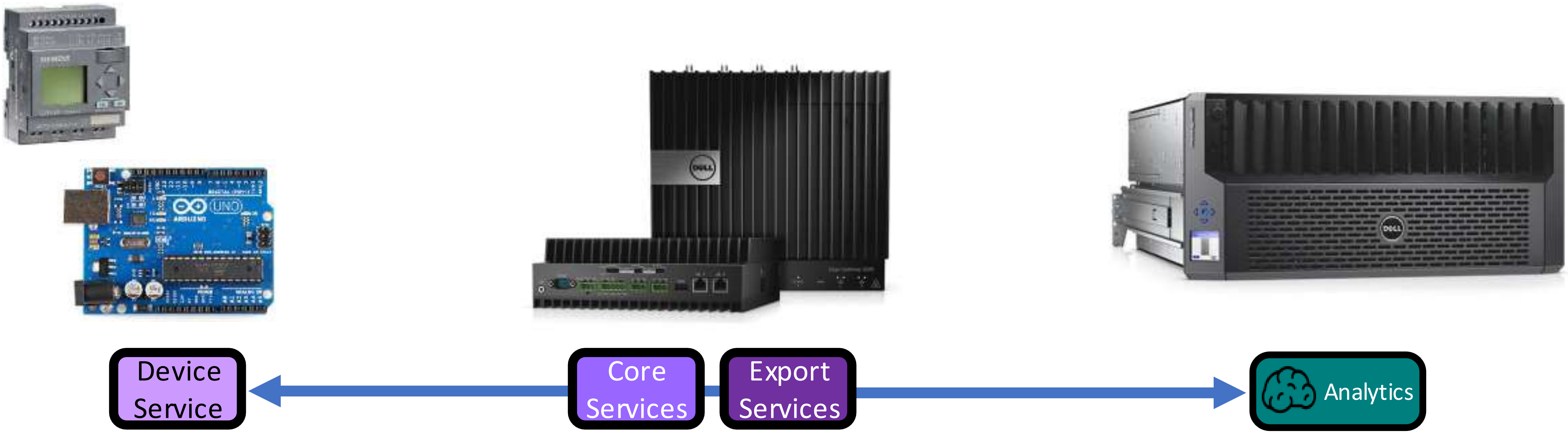
Keys to the diet success

- Go vs. Java
  - Executable vs virtual machine
  - Frameworks are heavy (Spring, EAI engine, etc.)
  - Library bloat (frameworks and Java use a lot of libraries)
- Micro service architecture
  - Replace one service at a time
  - Allowed for exploration and discovery (best practices, tools, etc.)
- Iteration
  - Crawl-walk-run
  - Get a replacement service, make it better, standardize across services

Distribution – the other tool in lean & mean



Distribution – Run where you can



Device Service

Core Services

Export Services

Analytics

EDGEXFOUNDRY™

Question??

Thank you

Jim White: james_white2@dell.com

Agenda

- 2 minute quick intro to EdgeX Foundry
- EdgeX architecture
- Requirements of an edge platform
- EdgeX performance metrics
  - In the beginning
  - Today
  - Keys to the diet
- Distributing EdgeX micro services
  - Why distribution matters

About Me

- Jim White (james_white2@dell.com)
  - Dell Technologies IoT Solutions Division – Distinguished Engineer
  - Team Lead of the IoT Platform Development Team
  - Chief architect and lead developer of Project Fuse
    - Dell's original IoT platform project that became EdgeX Foundry
    - Yes – I wrote the first line(s) of code for EdgeX (apologies in advance)
  - EdgeX Foundry …
    - Vice Chairman, Technical Steering Committee
    - Systems Management Working Group Chair
    - Ad hoc and unofficial lead architect

DELL Technologies

EdgeX Foundry

An open source, vendor neutral project (and ecosystem)

A **micro service**, loosely coupled software framework for IoT edge computing

Hardware and OS agnostic

A Dell incubated project started in 2015; entered into open source in April 2017

Linux Foundation, Apache 2 project

Goal: enable and encourage growth in IoT solutions

- The community builds and maintains common building blocks and APIs
- Plenty of room for adding value and getting a return on investment
- Allowing best-of-breed solutions

Backed by Industry Leaders

ABSOLUTE
aicas realtime
Alleantia
AMD
ANALOG DEVICES
AXZON
BASKING
Beechwoods
ubuntu Delivered by Canonical

CAVIUM
CertusNet
CIVIL INFRASTRUCTURE PLATFORM
CLOUD FOUNDRY
CloudPlugs Connecting Things
Cumulocity A SOFTWARE AG COMPANY
DELL EMC
DEVICE AUTHORITY Trust for every Thing
enigmedia

enocean
EPISENSOR
ETRI Electronics and Telecommunications Research Institute
FIWARE FOUNDATION
FOGHORN
FORGEROCK
IMS EVOLVE
industrial internet CONSORTIUM
inwinSTACK
IOTech

IoTium www.iotium.io
ISSAT Mateur
KMC CONTROLS
Linaro
MAINFLUX Internet of Things Solutions
MOCANA
modius
NETFOUNDRY
neustar

OMG OBJECT MANAGEMENT GROUP
OPTO 22
Project Haystack
Queen's University Belfast
redislabs home of redis
REPLY CONCEPT
RIOT
Rubicon
SAMSUNG
SAMSUNG SAMSUNG SDS

Sixgill Govern IoE
striim
SWITCH AUTOMATION
THALES
twobulls
ULE ALLIANCE
V5 SYSTEMS
vmware
Volterra EDGE SERVICES
万向集团 WANXIANG GROUP

xage SECURITY
ZEDEDA
Zephyr
ZingBox

With more in process!

EdgeX Foundry Goals

- Build and promote EdgeX as the common open platform unifying edge computing
- Enable and encourage the rapidly growing community of IoT solutions providers to create an ecosystem of interoperable plug-and-play components
- Certify EdgeX components to ensure interoperability and compatibility
- Provide tools to quickly create EdgeX-based IoT edge solutions
- Collaborate with relevant open source projects, standards groups, and industry alliances to ensure consistency and interoperability across the IoT

EdgeX Primer - How it works

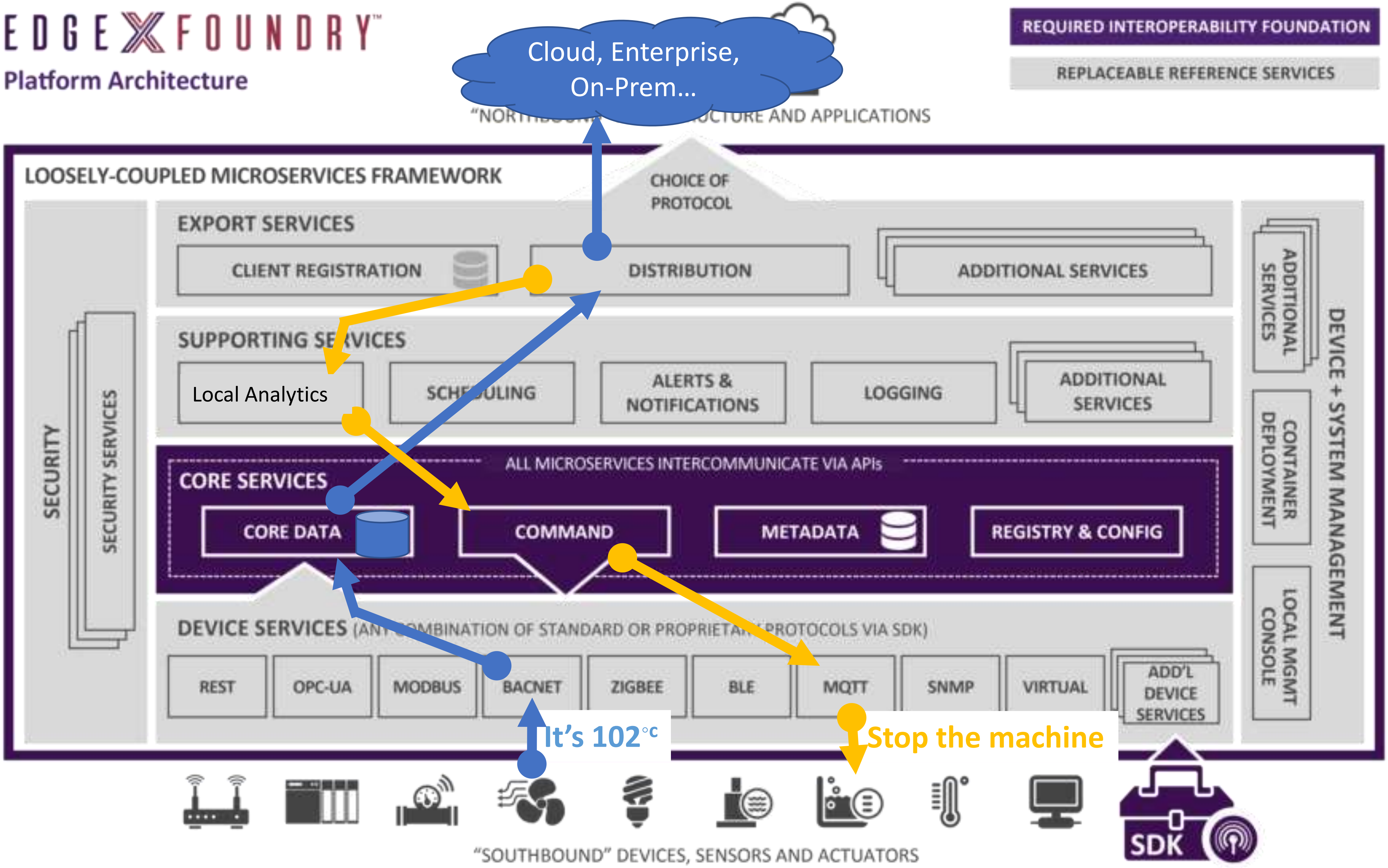
- A collection of a dozen+ micro services
  - Written in multiple languages (Java, Go, C, … we are polyglot believers!!)
- EdgeX data flow:
  - Sensor data is collected by a **Device Service** from a thing
  - Data is passed to the **Core Services** for local persistence
  - Data is then passed to **Export Services** for transformation, formatting, filtering and can then be sent “north” to enterprise/cloud systems
  - Data is then available for edge analysis and can trigger device actuation through Command service
  - Many others services provide the supporting capability that drives this flow
- REST communications between the service
  - Some services exchange data via message bus (core data to export services and rules engine)
- Micro services are deployed via Docker and Docker Compose

EDGEXFOUNDRY™

Platform Architecture

REQUIRED INTEROPERABILITY FOUNDATION

REPLACEABLE REFERENCE SERVICES



Cloud, Enterprise, On-Prem…

“NORTHBOU UCTURE AND APPLICATIONS

LOOSELY-COUPLED MICROSERVICES FRAMEWORK

CHOICE OF PROTOCOL

SECURITY

SECURITY SERVICES

EXPORT SERVICES

CLIENT REGISTRATION

DISTRIBUTION

ADDITIONAL SERVICES

SUPPORTING SERVICES

Local Analytics

SCHEDULING

ALERTS & NOTIFICATIONS

LOGGING

ADDITIONAL SERVICES

ALL MICROSERVICES INTERCOMMUNICATE VIA APIs

CORE SERVICES

CORE DATA

COMMAND

METADATA

REGISTRY & CONFIG

DEVICE SERVICES (ANY COMBINATION OF STANDARD OR PROPRIETARY PROTOCOLS VIA SDK)

REST

OPC-UA

MODBUS

BACNET

ZIGBEE

BLE

MQTT

SNMP

VIRTUAL

ADD'L DEVICE SERVICES

ADDITIONAL SERVICES

CONTAINER DEPLOYMENT

LOCAL MGMT CONSOLE

DEVICE + SYSTEM MANAGEMENT

It’s 102°C

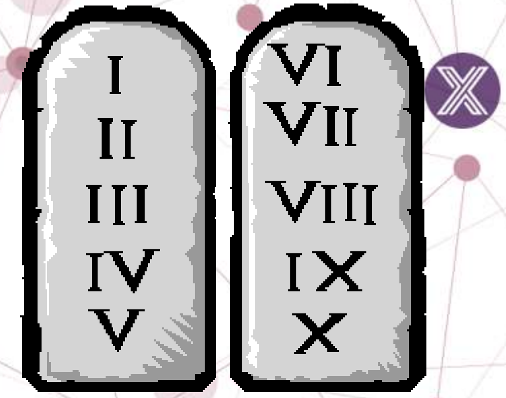
Stop the machine

SDK

“SOUTHBOUND” DEVICES, SENSORS AND ACTUATORS

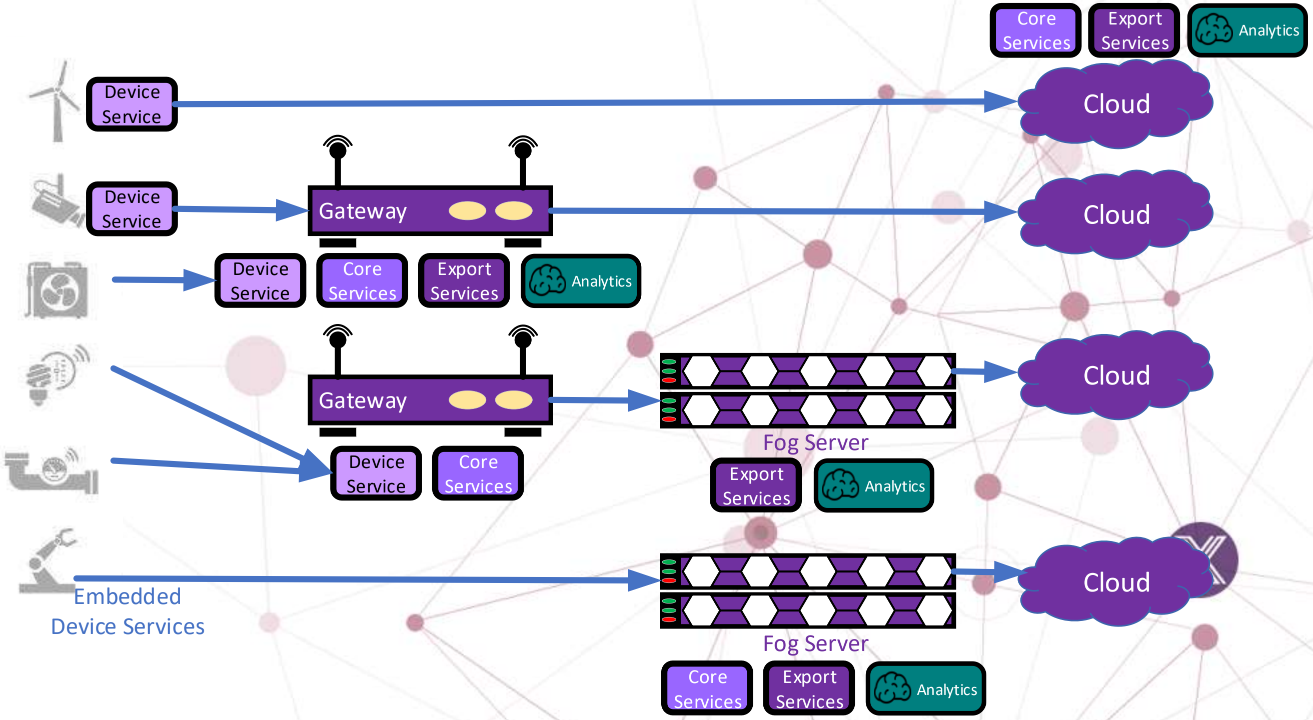
EdgeX Architectural Tenets

- EdgeX Foundry must be **platform agnostic** with regard to hardware, OS, distribution/deployment, protocols/sensors
- EdgeX Foundry must be **extremely flexible**
  - Any part of the platform may be upgraded, replaced or augmented by other micro services or software components
  - Allow services to scale up and down based on device capability and use case
- EdgeX Foundry should provide “reference implementation” services but **encourages best of breed solutions**
- EdgeX Foundry must provide for **store and forward** capability (to support disconnected/remote edge systems)
- EdgeX Foundry must support and **facilitate “intelligence” moving closer to the edge** in order to address
  - Actuation latency concerns
  - Bandwidth and storage concerns
  - Operating remotely concerns
- EdgeX Foundry must **support brown and green device/sensor** field deployments
- EdgeX Foundry **must be secure and easily managed**



EdgeX Enables Tiered Fog Deployments

- In today’s IoT landscape, it is imperative to leverage compute, storage, network resources where every they live
- Loosely-coupled architecture enables distribution across nodes to enable tiered edge/fog computing
- Scope includes embedded sensors to controllers, edge gateways and servers
- Quantity and function of micro services deployed on a given node depends on the use case and capability of hardware



EdgeX Technology

- A majority of the micro services are written in Go Lang
  - Previously written in Java
  - Some Device Services written in C/C++
  - A user interface is provided in JavaScript
  - Polyglot belief – use the language and tools that make sense for each service
- Each service has a REST API for communicating with it
- Uses MongoDB to persist sensor data at the edge
  - Also stores application relevant information
  - Allows for alternate persistence storage (and has been done in the past)
- A message pipe connects Core Data to Export Services and/or Rules Engine
  - Uses ZeroMQ by default
  - Allow use of MQTT as alternate if broker is provided
- Uses open source technology where appropriate
  - Ex: Consul for configuration/registry, Kong for reverse proxy, Drools for rules engine,…

Key Project Links

Access the code:

https://github.com/edgexfoundry

Access the technical documentation:

https://docs.edgexfoundry.org/

Access technical video tutorials:

https://wiki.edgexfoundry.org/display/FA/EdgeX+Tech+Talks

EdgeX Blog:

https://www.edgexfoundry.org/news/blog/

Join an email distribution:

https://lists.edgexfoundry.org/mailman/listinfo

Join the Rocket Chat:

https://chat.edgexfoundry.org/home

Become a project member:

https://www.edgexfoundry.org/about/members/join/

LinkedIn:

https://www.linkedin.com/company/edgexfoundry/

Twitter:

https://twitter.com/EdgeXFoundry

YouTube:

https://www.youtube.com/edgexfoundry

Requirements of an Edge Platform

- Not all platforms are equal
  - Real time vs near real time
- Collection rates
  - Typically dealing with sensor sampling rates of 100-1000/second
- Throughput
  - Typically handling kilobytes per message
  - Surveillance systems much greater
- Latency
  - To the cloud (or application layer)
  - Back to the device
  - < 1 second from sensor to analytics to actuation

Typical Collection or Sampling Rates

Use Case	Sampling Rate	Order of Volume
Electric Grid Line Fault Detection	Micro second	Bytes, KBs
Vibration Sensor	Micro second to seconds	Bytes, KBs
Pressure Sensor	Micro second to seconds	Bytes, KBs
PLC	Micro second to seconds	Bytes, KBs
CNC Machines	Micro second to seconds	KBs, MBs
CANBus	Micro second to seconds	MB
Building Automation	Micro second to seconds	Bytes, KBs
Locomotive Telemetry	Micro second to seconds	KBs, MBs
RTU in Utility	Micro second to seconds	KBs, MBs
Oil/Gas	Micro second to seconds	KBs
RFID	Micro second to seconds	Bytes, KBs

EdgeX in the beginning - April 2017

- EdgeX started as a Dell POC in 2015
  - We needed a platform independent language/framework
  - Enter Java/Spring Framework
  - It proved the concepts but wasn’t edge worthy
- Initial Performance was poor and not able to meet requirements
  - Running all the micro services required 7+ GB RAM
  - Improved to 5.1GB RAM with some tuning
  - CPU utilization around ~50% on a 2CPU, 8GB gateway
  - At startup, CPU pegged at 100%
  - Startup time for each service > 30 seconds
  - System startup ~5 minutes (due to service dependencies)

EdgeX @ the start (Hannover Messe 2017)

Service	Container size (in MB)	RAM (in MB)
Metadata	165	301
Coredata	170	299
Command	159	244
Logging	158	249
Scheduler	161	215
Notifications	153	222
Export Client	164	240
Export Distro	168	291
Rules Engine	177	270
SNMP DS	169	265
Fishertech DS	169	281
Bacnet DS	437	278
BLE DS	168	301
UI	607	50
Mongo	402	99

CPU % - 34%
Mem: 5.1 of 7.7G used
Swap: 168M of 7.9G used

Setting new Performance Targets in Oct 2017

- Clearly, EdgeX was not going to cut it as an edge solution
- 2nd EdgeX Technical Steering Committee Face to Face Decision
  - Move to an executable language (Go Lang) – based on early experimental testing
  - Set reasonable edge performance targets
- The target is to run all of EdgeX on a Raspberry Pi 3 type of device
  - 1 GB RAM, 64bit CPU, at least 32GB storage space
- Additional performance targets
  - Startup in 10 seconds or less (post OS boot)
  - Latency for one piece of data from data ingestion to actuation will be < 1 second
- Remaining OS and Hardware agnostic
  - Windows, Linux, *nix, …
  - Intel/Arm 64/Arm 32

Early Go v. Java Experiment (October 2017)

Measure	***Go***	***Java***
Executable/JAR Footprint	11.7 MB	42.4 MB
Container Footprint	**16.2 MB**	**165 MB**
Memory Usage (On Startup)	**4.3 MB**	**221 MB**
Memory Usage (Under Load)	**9.2 MB**	**230 MB**
CPU Usage (Steady State)	0.15%	0.30%
CPU Usage (Under Load)	5.0% - 15.0%	6.0% - 15.0% **(spiked at 90% for heavy load)**
Startup Speed	**0.14 Seconds**	**12.55 Seconds**
Response Speed (Ping)	0.0011 Seconds	0.0022 Seconds
Response Speed (Post)	0.0091 Seconds	0.0137 Seconds
Response Speed (Get)	0.0038 Seconds	0.0062 Seconds

EdgeX Today (Oct 2018 – Delhi release)

Service	Go Max CPU	Java Max CPU	Go Max Mem	Java Max Mem	Go Container Size	Java Container Size
Core Data	0.16%	13.71%	29.54	270	15.9	142
Core Metadata	1.02%	4.09%	8.70	275	10.5	125
Core Command	0.008%	1.10%	1.55	204	8.67	131
Support Logging	0.32%	7.31%	7.58	210	9.27	116
Support Notifications	0.09%	0.71%	4.53	217	9.65	125
Export Client	0.02%	7.66%	4.12	221	15.9	136
Export Distro	0.06%	9.99%	3.84	251	16.2	140
Mongo	0.93%	0.55%	72.30	76	361	361
Consul	1.71%	0.43%	17.12	12	59.1	168
Docker Files/Volume	0.01%	0.01%	0.20	0	81.2	122
Total	*4%*	*46%*	*149*	*1736*	*587*	*1566*
	%CPU	*%CPU*	*MB*	*MB*	*MB*	*MB*

- Can run on a Rasp Pi 3 level of device
  - < 1GB RAM
- > 90% memory reduction
- > 90% CPU reduction
- > 80% footprint reduction
- > 95% startup time reduction
  - Startup ~ 5 seconds

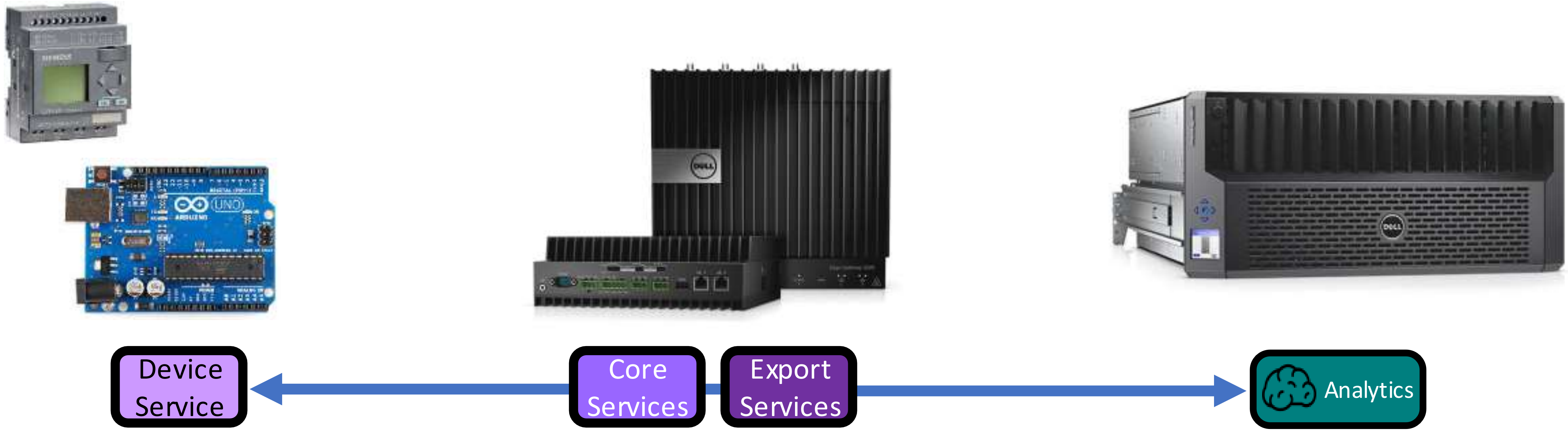
Keys to the diet success

- Go vs. Java
  - Executable vs virtual machine
  - Frameworks are heavy (Spring, EAI engine, etc.)
  - Library bloat (frameworks and Java use a lot of libraries)
- Micro service architecture
  - Replace one service at a time
  - Allowed for exploration and discovery (best practices, tools, etc.)
- Iteration
  - Crawl-walk-run
  - Get a replacement service, make it better, standardize across services

Distribution – the other tool in lean & mean

- EdgeX is a collection of independent and loosely couple micro services
  - Allows for distribution across the available compute resources
- In today’s IoT landscape, it is imperative to leverage compute, storage, network resources where every they live
- Loosely-coupled architecture enables distribution across nodes to enable tiered edge/fog computing
- Scope includes embedded sensors to controllers, edge gateways and servers
- Quantity and function of micro services deployed on a given node depends on the use case and capability of hardware

Distribution – Run where you can



Device Service

Core Services

Export Services

Analytics

EDGEXFOUNDRY™

Question??

Thank you

Jim White: james_white2@dell.com