EDGE X FOUNDRY™

Technical Workshop

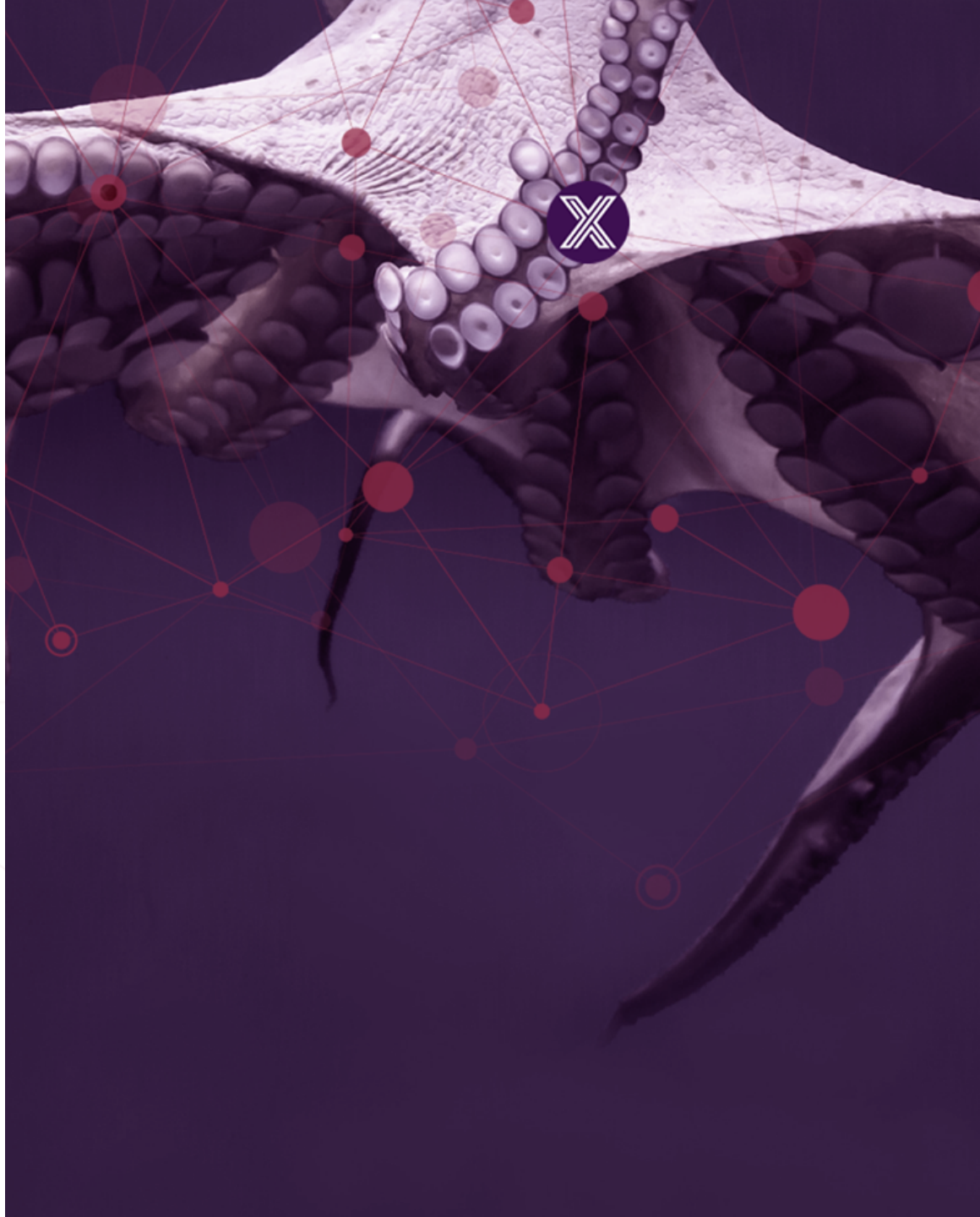
Day 1: June 1, 2017

Agenda

Thursday, June 1	
8-9am	Breakfast
9-9:30am	Introductions
9:30-10:30am	High level project organization, proposed working groups, key additional functional areas
10:30-Noon	Overall alignment on project goals, performance needs, presentation from IOTech on footprint evaluation, group discussion on functional requirements for subsequent working group breakouts
Noon-1	Working lunch. Continued prep for breakouts.
1-5pm	Technical hands-on working group breakouts
7pm	Dinner
Friday, June 2	
8-9am	Breakfast
9-noon	Continued technical hands-on breakouts, summarize output per guidance
Noon-2pm	Working lunch, groups report out status, assimilate output
2-3pm	Group discussion on TSC organization, future certification program, resourcing/contributions, meeting/development structure, next steps

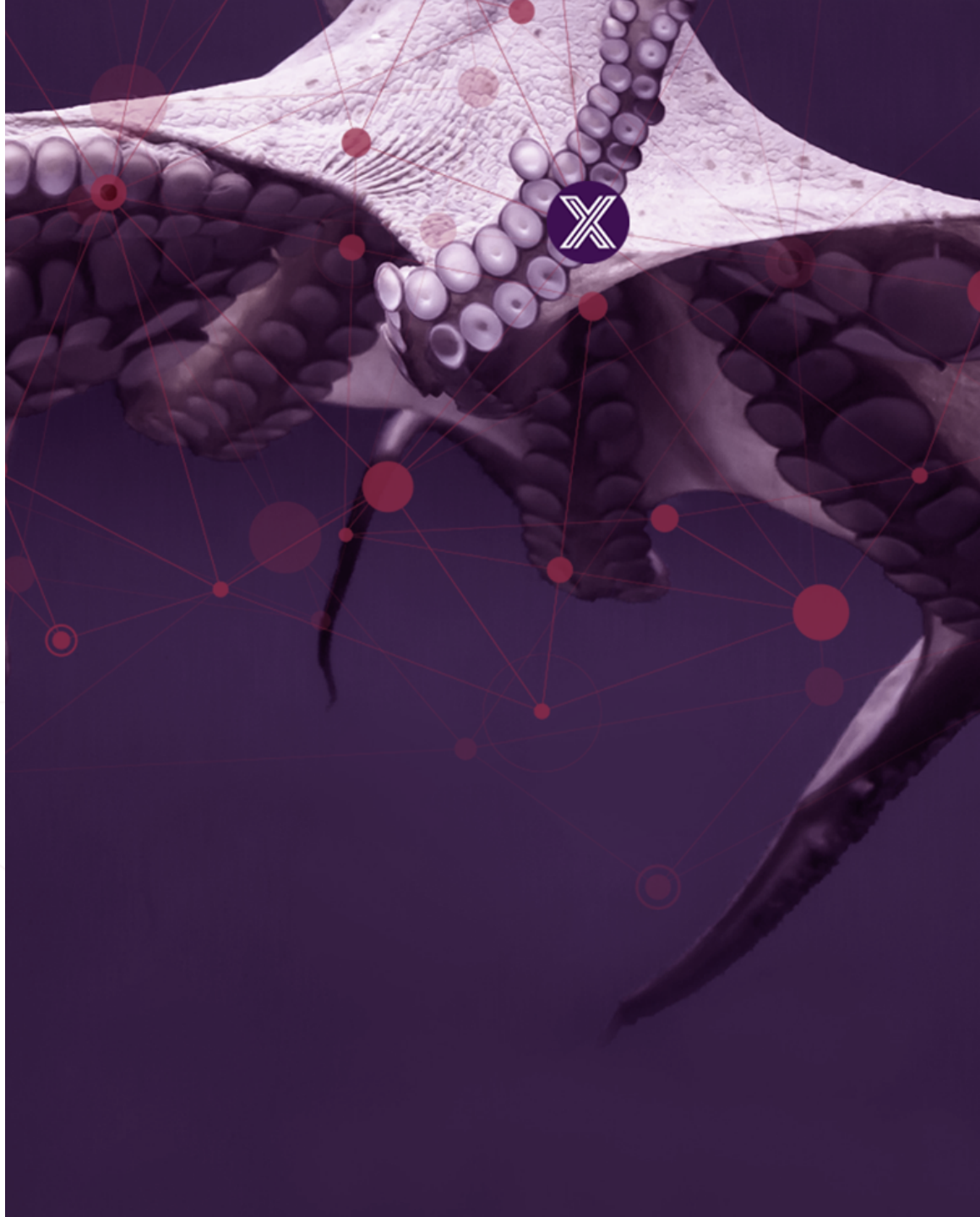
EDGE X FOUNDRY™

Introductions

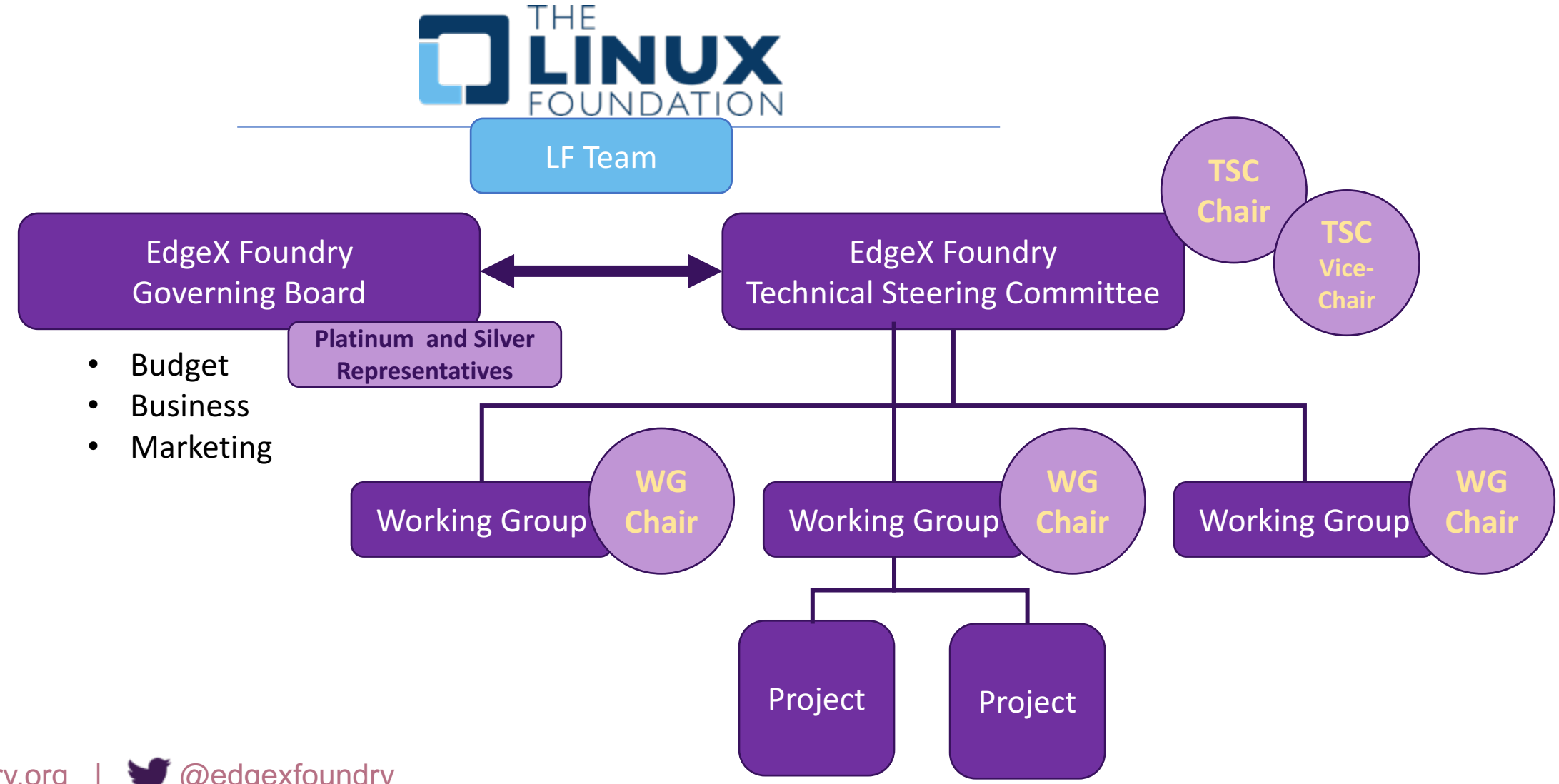


EDGE X FOUNDRY™

High-Level Project Organization



EdgeX Foundry Project Organization

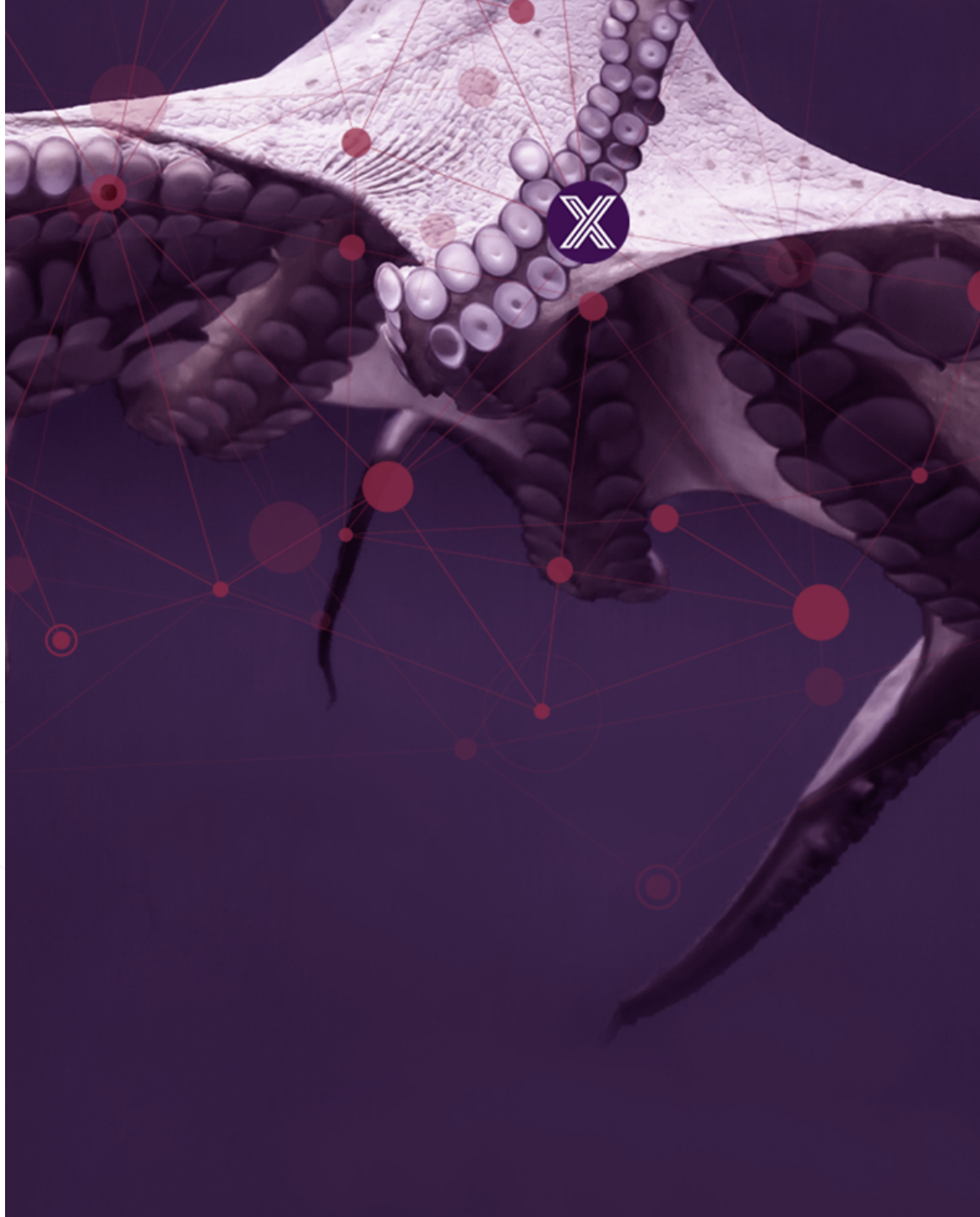


Project Documents

- Approved and Accepted (Changes made by the Governing Board)
  - Project Charter - https://www.edgexfoundry.org/about/project-charter/
  - Code of Conduct - https://www.edgexfoundry.org/about/code-of-conduct/
- Structure and Operations (To be developed by the membership)
  - Technical Work in the EdgeX Foundry Project [Draft]
    - EdgeX Foundry Project Principals [Draft]
    - Working Group Charter Template [Draft]
    - Project Lifecycle [Draft]
    - Proposal Template for an EdgeX Project [Draft]
    - Project Incubation Exit Criteria [Draft]
    - Release Taxonomy [Draft]

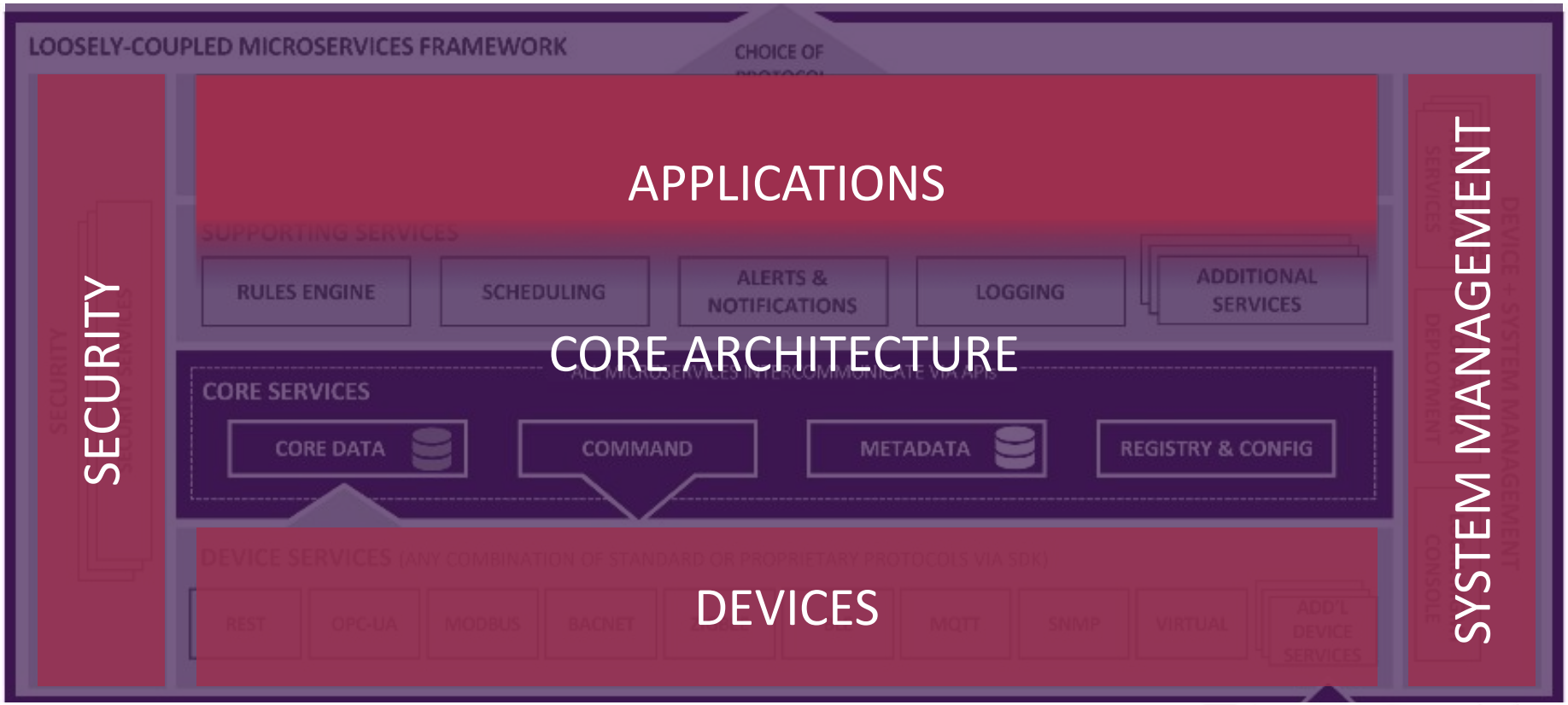
EDGE X FOUNDRY™

Proposed TSC Working Groups



Proposed Baseline Working Groups

- Proposed groups align with key functional areas in platform architecture
- Inherent bleed between groups with alignment maintained via common APIs



DevOps?

QA?

?

Core Architecture Workgroup

Overall Responsibilities

- Curate overall architecture and recommendations for technology implementation
- Foundational APIs and native EdgeX data model
- Microservice deployment and inter-communication
- Optimize performance, reliability, scalability, extensibility and platform independence
- Address needs of other working groups

Security Workgroup

Overall Responsibilities

- Core security architecture and functional requirements
- Features for securing host device and connected devices/sensors
- Threat modeling and penetration testing
- Recommendations on best practices for code implementation and field deployment

System Management Workgroup

Overall Responsibilities

- APIs and reference implementation for the management and health of the host system and connected devices/equipment
- Deployment and maintenance of EdgeX microservices (start, stop, restart, update, provide status, configuration, etc.)
- Test integrations with 3rd party consoles

Device Workgroup

Overall Responsibilities

- Interoperability between “southbound” connectivity protocols and Core Services
- Device microservices that provide this connectivity
- SDK to create new Device Services, including various code implementations
- Working with security and system management workgroups to extend functionality to connected devices

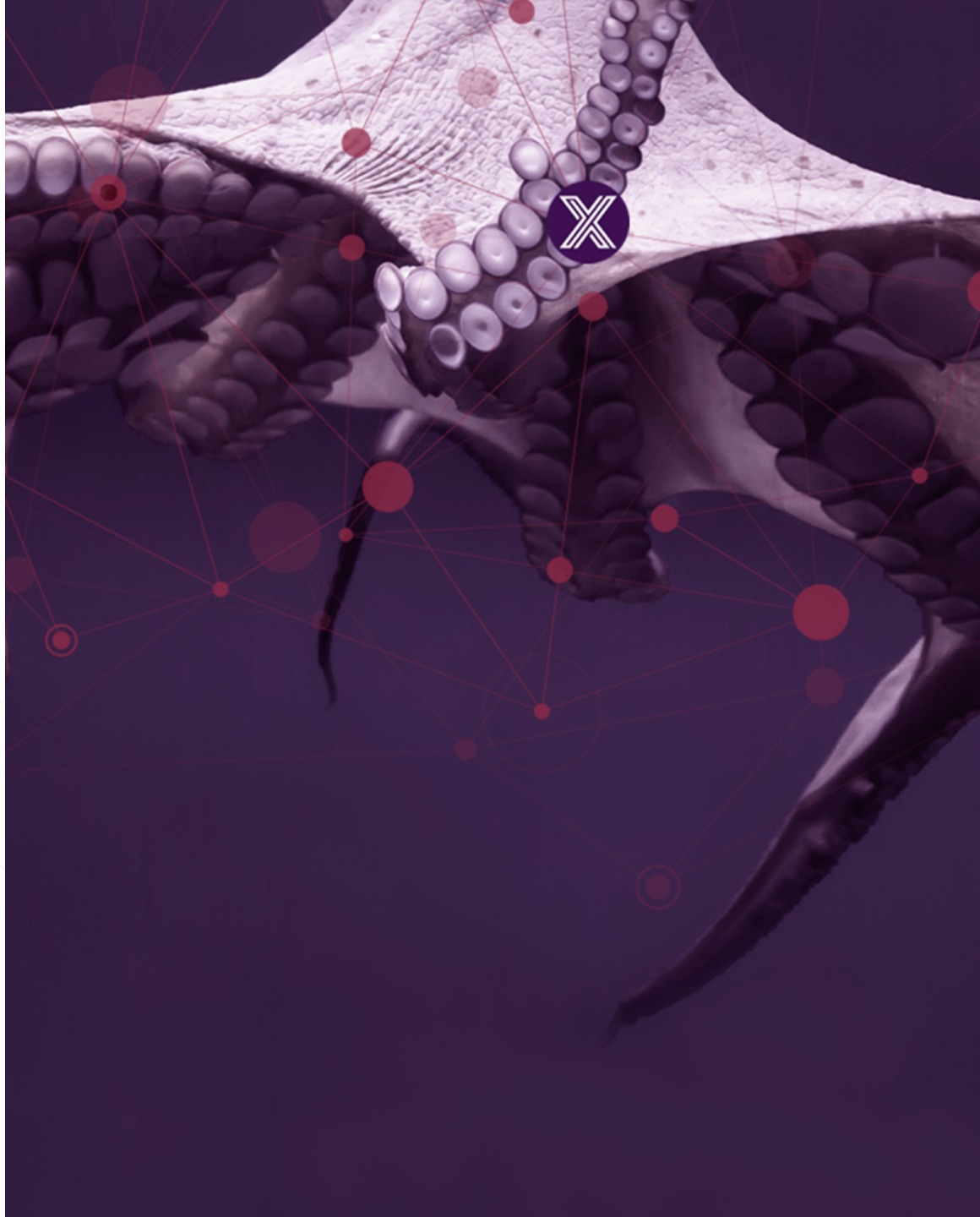
Applications Workgroup

Overall Responsibilities

- Define MVP functionality (e.g. deployed microservices) for usable instance, associated performance and footprint requirements
- "Northbound" Export Services for interoperability and exchange of data (with local applications, other edge/fog nodes and cloud systems)
- Tools and SDKs to add additional integration capability in various formats
- Format/model transformations (Haystack, OPC-UA, etc.)
- API and performance needs of general applications such as edge analytics (CEP, machine learning), database, and other “intelligence” integration that resides in EdgeX or is provided in coordination with backend systems
- “Fog” data orchestration
- Plugin new transformations, filters, encryption, compression, protocol connecters, etc.

EDGE X FOUNDRY™

Additional Key Project Functional Areas



DevOps (1/2)

Proposal: up-level to standalone working group

- Project and Software Development Management
  - LF provides the infrastructure (tools) setup and support. The TSC is responsible for project + software development management
- Source Code Management
  - LF provides and supports the tools -- e.g. GitHub setup, ongoing administration (adding new projects and maintainers). The TSC/project maintainers are responsible for proper source code management.
- Code Review
  - LF provides an inbound code management framework but code reviews are generally performed manually by the code maintainers. Anyone in the community can provide code reviews; maintainers are responsible for final review and accepting or rejecting contributions. The LF has set up an environment that publishes automated code testing results alongside developer review and the community can contribute new automated tests.
- Bug Tracking
  - This has been set up with GitHub. Bugs/Issues will be tracked using GitHub Issues; the TSC and community are responsible for triaging and responding to issues as part of source code management.
- Communications Infrastructure
  - Provided by LF and already set up: Email proxies as necessary (PR, Events, Membership, Info); initial Mailing Lists, Wiki (which will be refreshed within the next 1-1.5 weeks with large content transfer from Dell, Rocket.Chat (developer forum)

DevOps (2/2)

Proposal: up-level to standalone working group

- QA and Testing
  - Test processes, test cases, test frameworks, test platforms (suitable hardware/OS to run the tests on), tests (unit, integration, system, performance, scalability
  - Recommendation is for the project to form a Working Group to address. LF provides administration of virtual machine test platforms as part of CI administration.]
- Continuous Integration
  - Provided by LF: a Jenkins master integrated with GitHub , a dynamic pool of Jenkins worker minions that scales up or down to meet project demand and Nexus Repository Manager. The LF will be taking the lead to implement a better release process than what was previously utilized; one that uses Nexus to hold artifacts and help create and push new Docker containers into private Docker repos (and then Docker Hub). The EdgeX Community is welcome to participate in the scoping/roll-out of the new process.
- Release Engineering and Management
  - Provided by LF
- Systems Administration
  - Provided by LF
- Inbound Code Scanning
  - Initial scanning executed by Dell on seed contribution. Future code scanning will be enabled once the project budget is able to fund.
- Election of Code Maintainers for Initial Contributions
  - Managed by TSC

Quality Assurance

Also up-level to standalone working group?

- QA is a top priority to establish EdgeX as high quality resource as foundation for application development
- Requires processes, procedures and systems to support
  - Product management – requirements definition and management, product roadmap development
  - QA and testing processes – requirements analysis and design, test plan, test cases, unit tests, integration tests, performance testing, scalability testing, soak testing, platform/OS testing, release management
- Also requires strong link with DevOps (automation of the build, test, release and continuous integration processes)

Addressing Business Needs

- Enabling proprietary value-add around open interop core
- Multi-tenancy
- Meeting requirements of various stakeholders (IT, OT, admins, installers, service personnel, etc.)
- Tracking entitlement and usage of proprietary microservice licenses (e.g. enable pay-for-services model)
- ?

Community Engagement

- Consortia / standards bodies
  - EdgeX in consortium test beds (IIC, OpenFog, etc.)
  - Collaboration for better-together with protocol standards
- Developer Resources
  - Discussion: Best place to post dev questions? Rocket.Chat, Mail Lists, Stack Overflow (would people be looking at SO, to answer questions)? Dev questions come through multiple channels. Where does the community "live" > i.e. we don't want to recommend SO if nobody uses it anymore)
- Developer Training Videos
  - Series of Live + Recorded Zoom sessions. Example topics include: Intro and high level architecture; Core Training; Base Service Frameworks Training; Technical Getting Started; Debugging/Testing; Debugging/Testing with third party tools; Contribution Process, git, gerrit, and coding standards.
- University liaisons/research projects
- Event / conference participation
  - Trade shows
  - Plug-fests and hackathons
- Social media, blogs and 3rd party forums

Commercial Enablement

Building on QA and internal DevOp, tasks required to make EdgeX fit to be consumed by commercial users.

- Support system – bug tracking system/support portal
- Product packaging – binary distributions and professional installation
- User Documentation – Installation Guide, Getting Started Guide, User Guide, API Reference, Administration Guide
- Release notes – known issues, bug fixes, change summary, documentation links, tutorial links, software license terms and conditions
- Extensive example code and tutorials
- Supporting training materials for use with customers and partners
- Dev kits

Certification Program

- Certification program to be paid for by EdgeX project membership funding and incremental certification fees
- Requires definition of process, selection of native data model, certification spec and test jigs, etc.
- Project can go one of two paths:
  a) Follow similar process/practice as OCI: https://github.com/opencontainers/certification
  b) Leverage outside resource (i.e. UnderwriterLabs)
- UI: relationship to OpenFog and IIC test beds, others
- Will discuss in a bit more detail on Day 2

Example community offerings to be certified:

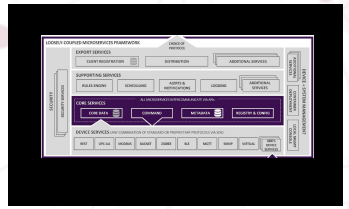
Individual "North-side" Microservices

Export Services

Edge Analytics

Security Services

System Mgmt Agennt



Entire proprietary edge platforms built on Interoperability foundation

Device Services

Connectivity Servers

Device Services to accompany new sensors/devices or to accommodate brownfield connectivity using choice of protocol

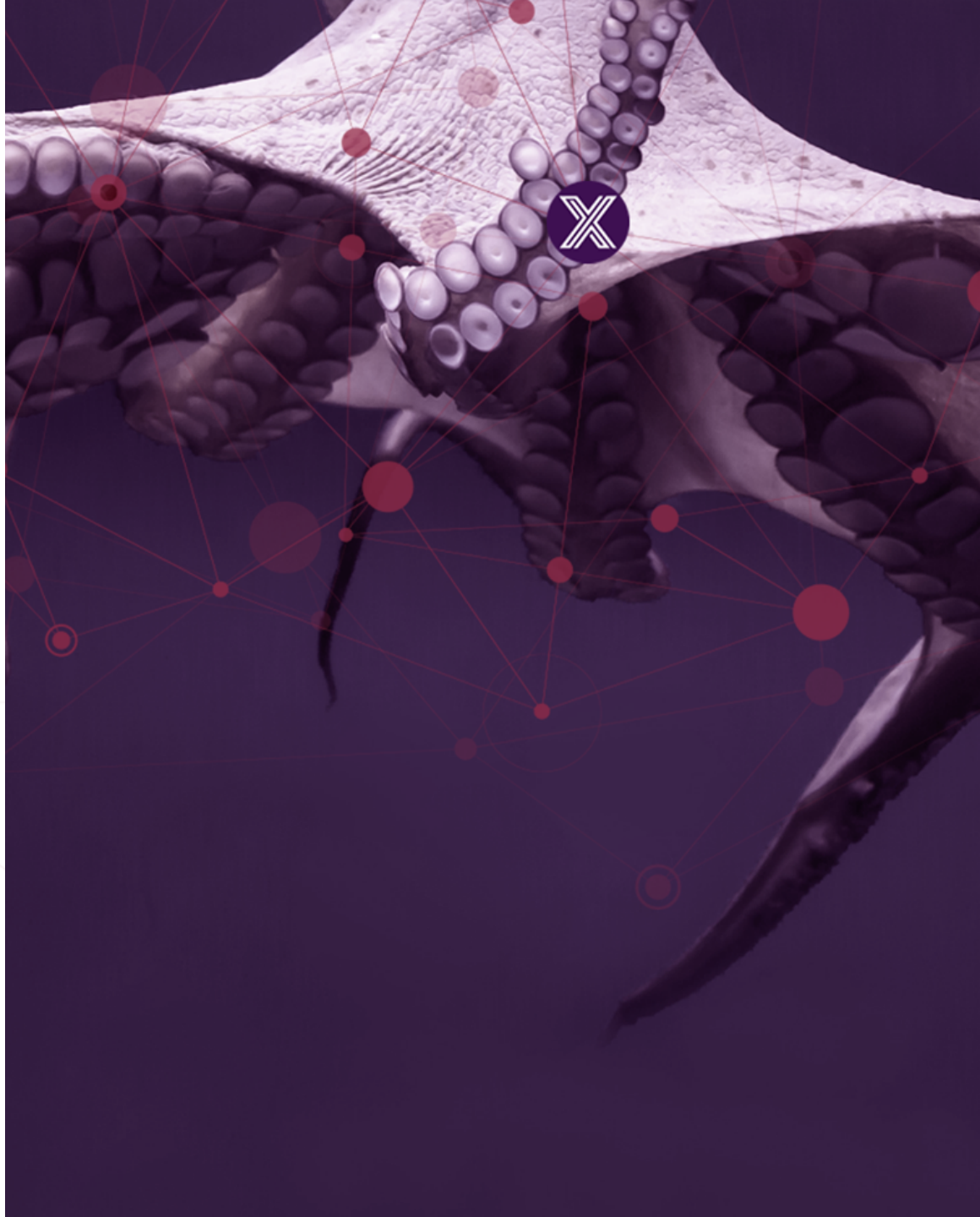
Existing gateway stacks / connectivity applications

For End of Day 2

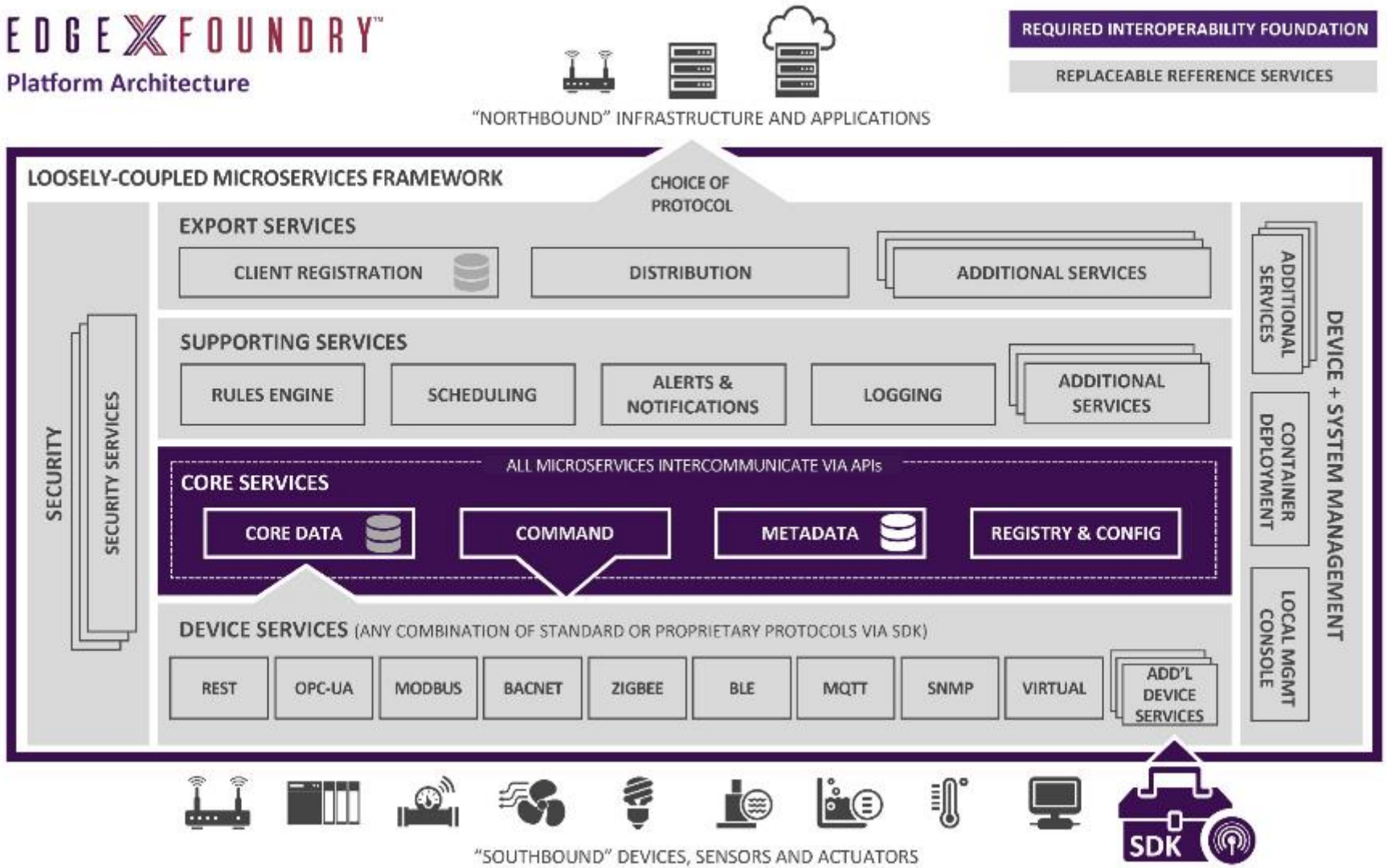
- Discuss integration of key functional areas into Working Groups, up-leveling DevOps and QA to standalone groups
- Create first-pass sub-projects
- Finalize draft TSC structure

EDGE X FOUNDRY™

Overall Alignment on Project Goals



Platform Architecture



Current State of Project Seed

- Solid architecture
- Stable but heavy footprint
- Dockerized for ease of end user delivery
- No security features
- Limited system management features
- Some clean up items to be addressed
- Limited documentation – especially for the “Getting Started” types
- Single node system – doesn’t address east/west or extended fog distributed system

Proposal: Stabilize Current Base Before Creating Alternative Implementations

- Explore alternative technology choices and code implementations in parallel to stabilizing and optimizing current base
- Expedite definition of MVP Security and System/Device Management to influence core to be implemented after key technology choices and programming languages are established
- Stabilize Java-based Device Service SDK before adding more reference device services and exploring alternative code implementations

Proposed Near and Longer-Term Goals

Beta (Fall 2017)

- Clean up current code and optimize footprint/performance
- Define foundational security and system management requirements, implement MVP functionality
- Refine Device Services and Export SDKs
- Create additional reference device and export services
- Test/curate with various platforms (hardware and operating systems)
- Establish and implement base Open Source QA procedures
- Additional productization of Open Source code base to enable community/commercial use

Production Enablement (2018+)

- Enable further footprint reduction via alternative code implementations and embedded flavors
- Improve/expand south and north-side connectivity and partnerships
- Advanced security and system management
- Explore real time and embedded versions
- EdgeX in the fog (east-west distribution, load-balancing, failover, etc.)

Proposed Key Milestones for Next 12 months

Establish Key Technology Choices

- Optimized core
- Alternative language support
- MVP security and system management features
- Native data model

Beta Release for Community Acceleration

(Target: IoT Solutions World Congress October 3, 2017)

- Dev kits through channel
- Ramp events including plug-fests with partner value add, relations with universities, smart city efforts, etc.
- PoCs with end customers for feedback and further hardening

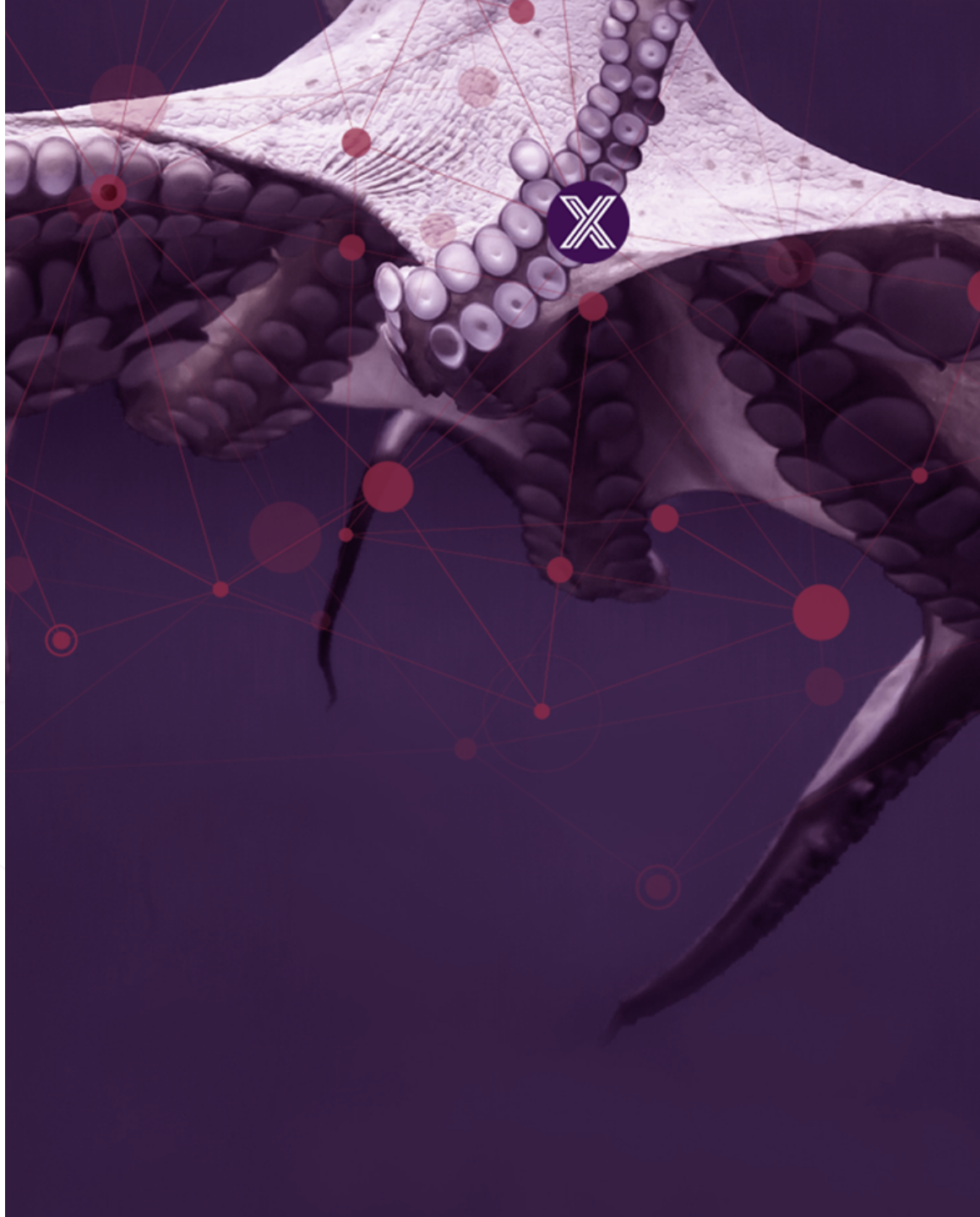
Commercial-ready MVP

(TBD, target Spring CY18)

- Commercial-quality foundation
- Formal certification program in place at the Linux Foundation

EDGE X FOUNDRY™

Alignment on Performance Needs



Summary of Current State on Performance

- To date project priority has been to develop a highly extensible architecture per the key tenets over performance and footprint
- Performance is highly relevant to use case including quantity and function of deployed microservices
- Recent prototyping has improved on these numbers and shown existing architecture can support drastically reduced footprint in a typical use case

Near Term Solutions for Performance

- Remove services (export, central logging, central config/registry, notifications, etc.) in constrained environments
  - Lift: Small (configuration changes and some conditionals to operate with/without supporting services)
  - Benefit: A device service or two, with core services and rules engine would run nominally on 2GB platform
  - Negative consequences: Remove capability
- Combine services (e.g. combine core, metadata, and command) into one service
  - Lift: Medium
  - Benefit: Reduce footprint (POC still under development but estimate half the size for core services) and improved performance given fewer REST calls
  - Consequence: Reduce flexibility
- Use commercial Java Native Code Compiler to create native EXE from Java micro service JAR
  - Lift: Small (long build process in excess of 2 hours, but does not require extensive developer work)
  - Benefit: ~1/2 the start time, ~2x speed, ~3/4 the footprint of a normal Java service
  - Consequence: $, commercial license needed to use the product
- Explore use of Mongo indexes to improve performance – especially around UI
- Combination of these efforts

Longer Term Solutions for Performance

- Rewrite services in Go Lang or other language
  - Lift: Large
  - Benefit: Large reduction of footprint. Promising early prototype of Core Data (only 8MB memory, 8M footprint), near instantaneous start times
  - Consequence:
    - Resources to develop
    - Requires a compile for a specific target platform (can be done as part of a more complex build process)
    - Availability of frameworks/tools hinders development
- Refactor Java code base to optimize for scale / performance
  - Lift: Large
  - Benefit: Retain the architecture and some code base
  - Consequence:
    - Resources to rework code base and tools/expertise to locate pockets of reduction
    - How much smaller / faster could the Java microservices be?

Evaluation on Reducing Footprint

Keith Steele, IOTech

Micro Service Clean Up – Footprint & Performance (1)

- Java is resource heavy – current footprint roughly 7-8gb with full complement of microservices, rules engine and local UI
- Optimum approach to reduce footprint/ optimize performance would to re-implement some/all micro services in another programming language
- Micro service architecture, permits incremental replacement
- Initial research and POC's suggest Go Lang would be a good choice; easy to implement a web application, concurrency model, and fitting Docker container
- Estimated average footprint for each Go Lang based micro service is around 40 ~ 50 MB. Each Java micro service is currently around 250 ~ 300 MB
- Possible exception - Device SDK optimally should be written in C, allowing the integration of resource constrained devices at the edge and RTOSs that do not support Go Lang or Java

IOTech

Micro Service Clean Up – Footprint & Performance (2)

- Reducing the footprint can improve the performance as most CPU resource is handling garbage collection
- The current implementation could be modified to use Undertow as the web container instead of tomcat. It could reduce footprint by about 4MB for each micro service
  - See https://docs.spring.io/spring-boot/docs/current/reference/html/howto-embedded-servlet-containers.html#howto-use-undertow-instead-of-tomcat and
  - https://examples.javacodegeeks.com/enterprise-java/spring/tomcat-vs-jetty-vs-undertow-comparison-of-spring-boot-embedded-servlet-containers/
- Continue analysis of breakdown of startup timings to understand where the time is consumed; identify possible optimizations
- Continue investigating VM's like IBM j9 support multi-tenancy (running multiple applications in on server JVM), this could be a quick way of reducing the JVM overhead – initial result look promising (see next slide)
- Also potentially modify the micro services so can be co-located in a standard JVM

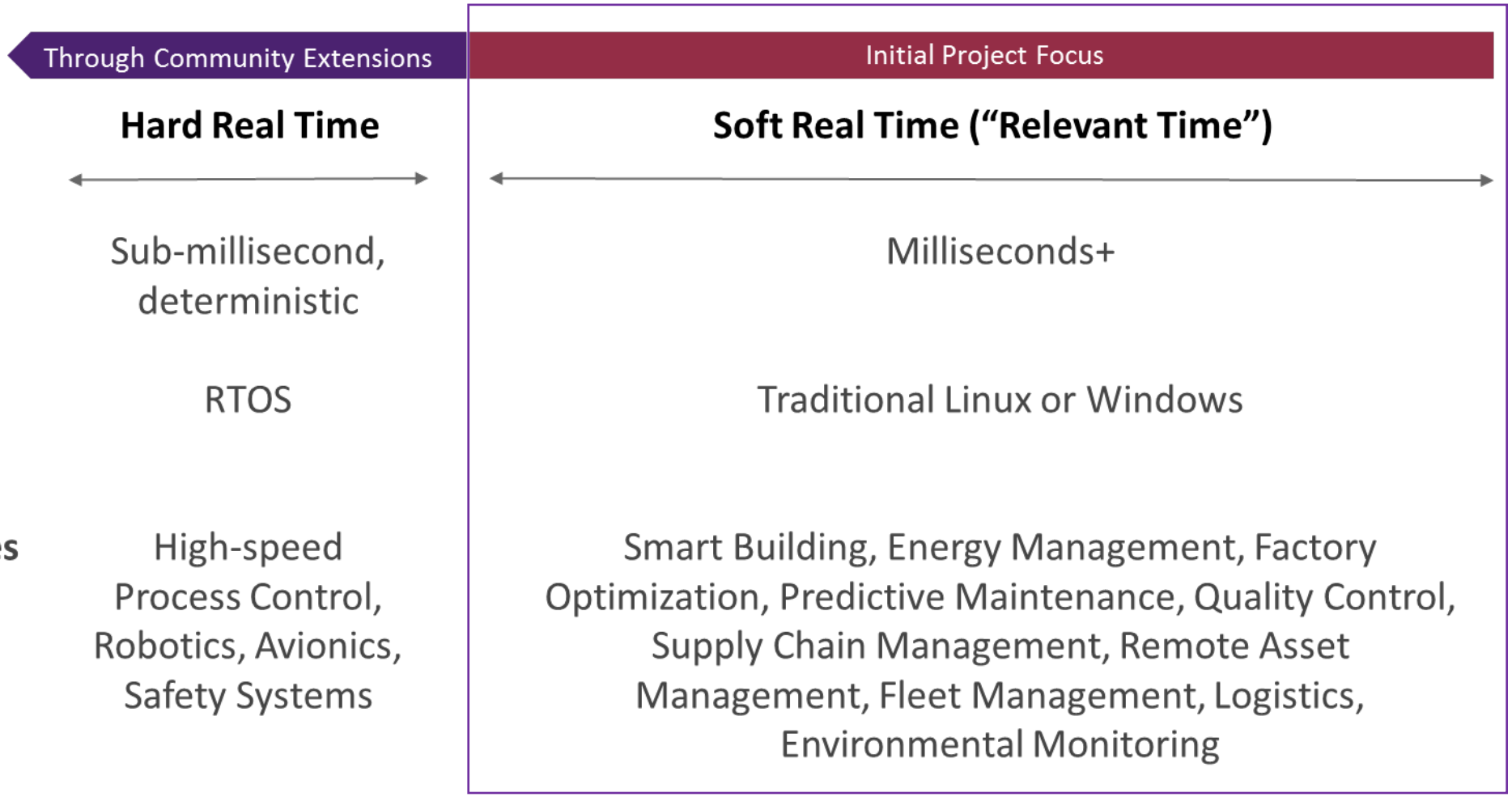
IOTech

Preliminary IBM J9 Performance and Footprint Results

- Test setup - Dell Latitude 3440 laptop, CPU: Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz, RAM: DDR3 8GB, OS: Ubuntu Desktop 14.04
- Test 1 - run EdgeX normally = **start up time: 5 minutes, memory consumption: 6.4 GB**
- Test 2 - re-build all Docker images of Java micro services based on IBM j9 with the Class Data Sharing feature = **start up time: 3 minutes, memory consumption: 1.7 GB**
- Test 3 - re-build the Docker images again with Undertow instead of Tomcat. Result as in Test 2, thus we don't need to replace the web container when using the Class Data Sharing feature because it might only reduce the footprint by 4Mb
- Test 4 - removed logging and rules engine micro service = **start up time: 2 minutes, memory consumption: 1.5 GB**
- Using the IBM J9 VM EdgeX could run easily on the Dell 5000 Gateway smoothly, and could run on 3000 Gateway by e.g. disabling logging and rules engine
- Work ongoing

IOTech

Proposal: Focus on Soft Real Time for First Commercial Offerings



Recommendations

- Use short term fixes (one or combination) to optimize current code base first
- Submit long term solutions as part of the technical roadmap to the TSC
  - Divide work up among community

Performance Targets

- Need to define typical light, medium and heavy workloads (e.g. Light: Core Services and baseline Supporting Services, one Device Service, one Export Service, no database, analytics or local UI?)
- What hardware target for Beta at light workload? Atom-class gateway w/ 2GB RAM? Raspberry Pi (ARM V8, 1GB RAM)?

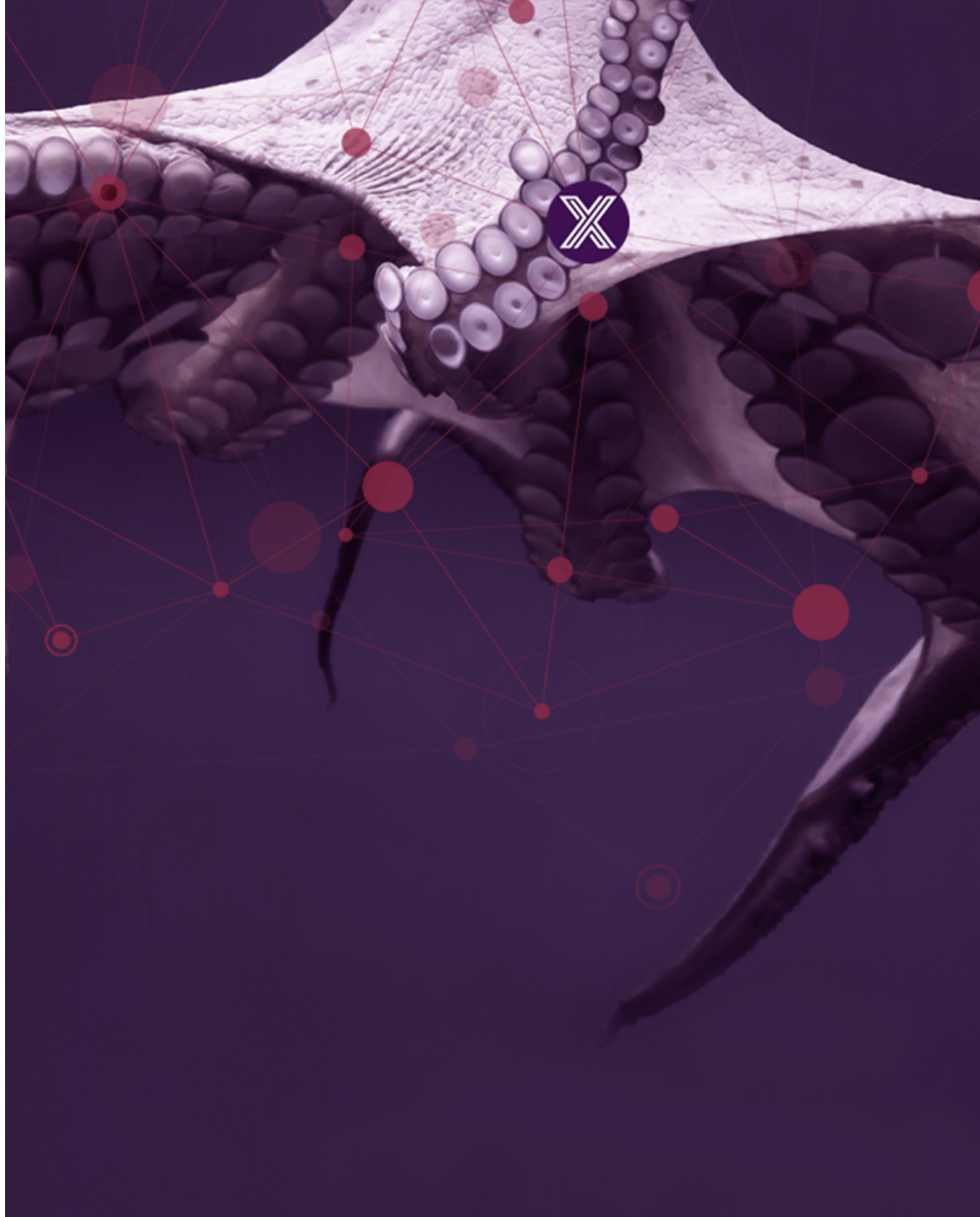
Attribute	Alpha (Today – Hannover Demo)	Beta MVP	Commercially-Viable Base
Min CPU	Dual Core Atom	???	???
Footprint (disk)	3.5GB	???	???
Footprint (mem)	4.5GB	???	???
Boot Time	5-7min	???	???
Latency (acquisition to export)	300ms?	???	???
Latency (acquisition to actuation)	500ms	???	???
???			

Rough alignment here to be expanded on during breakout

EDGE X FOUNDRY™

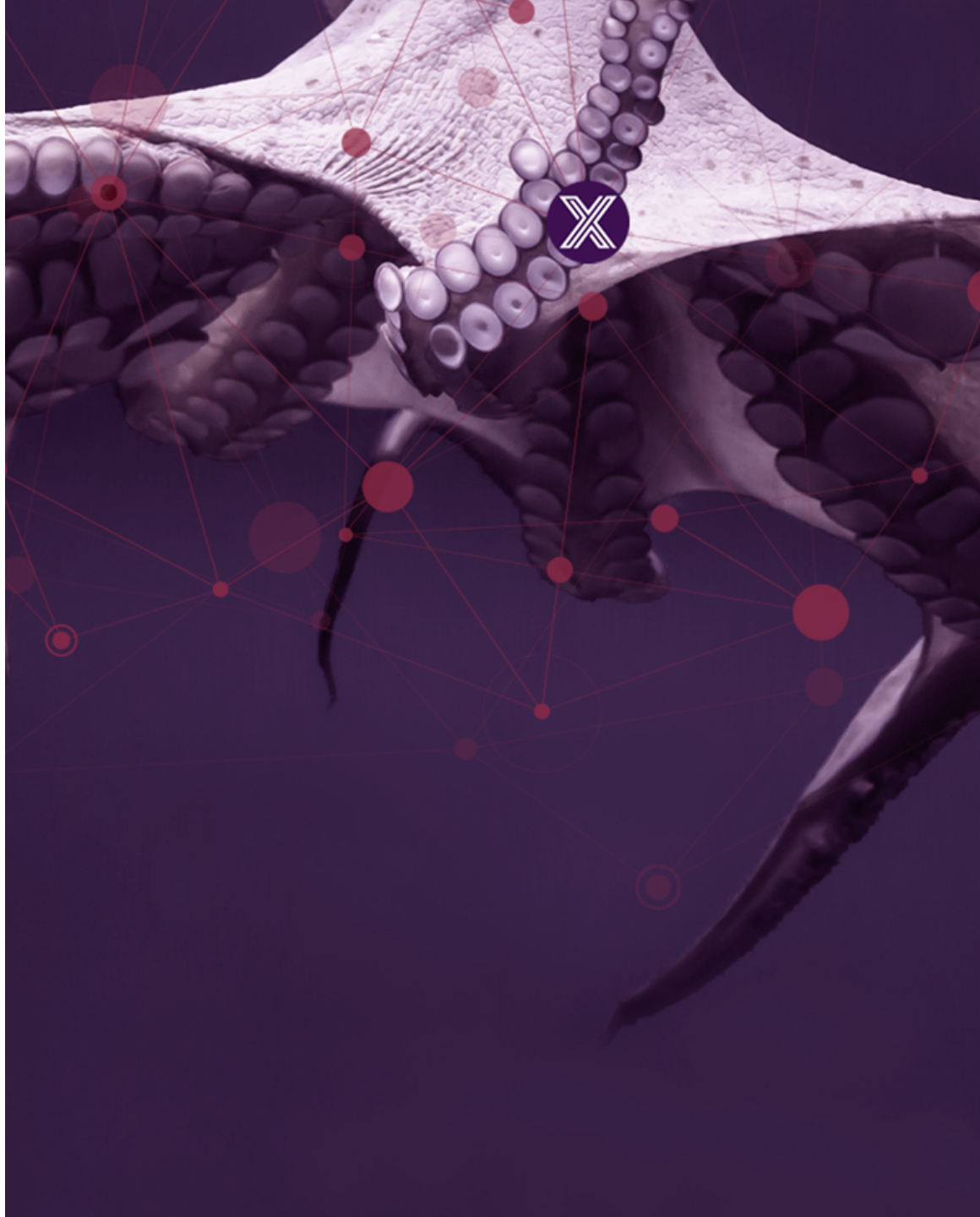
Alignment on Overall Requirements and Design

(For group discussion to seed deep dives in working groups)



EDGE X FOUNDRY™

Core Architecture

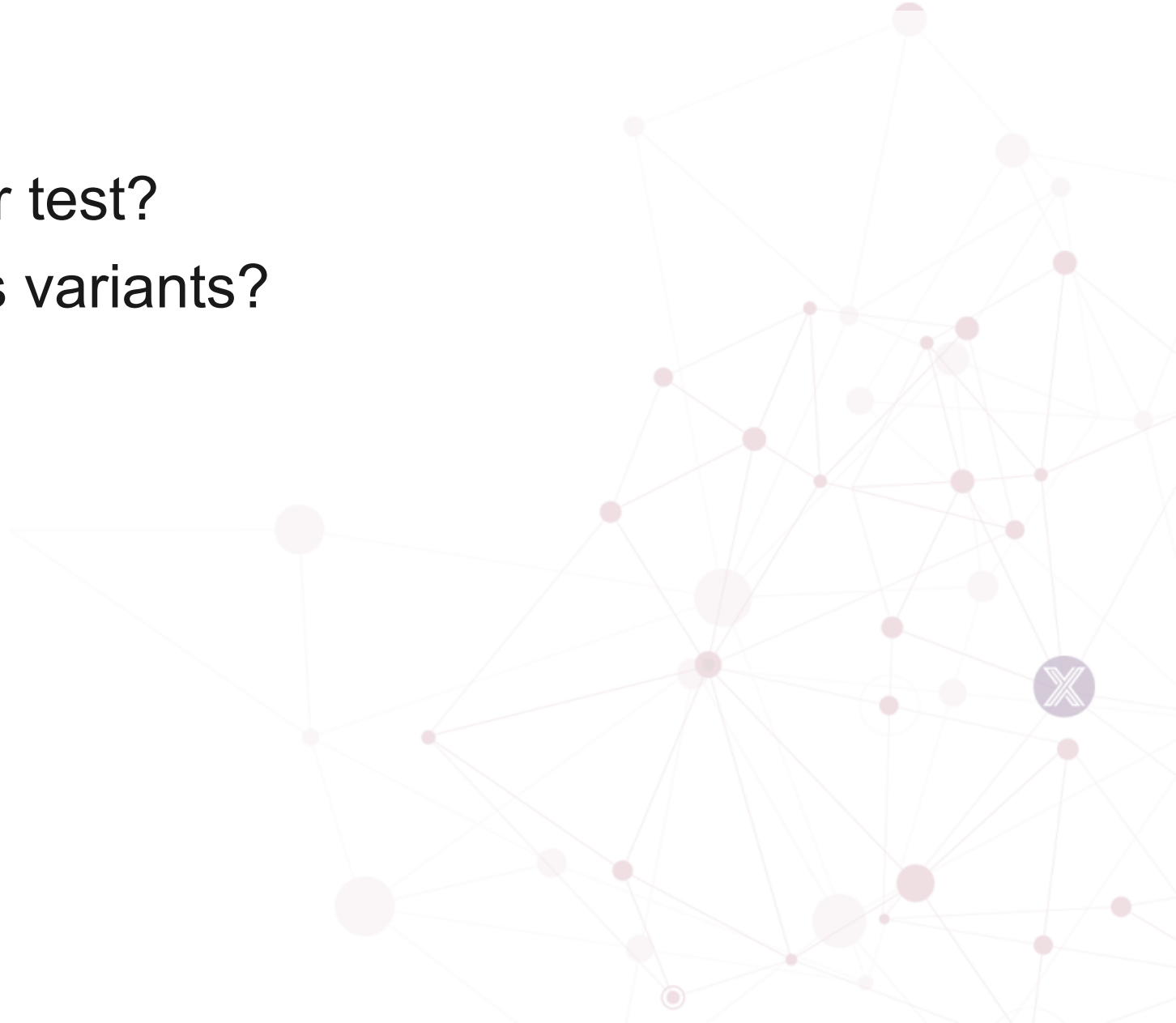


General Clean Up of Core

- Reduce footprint and latency of current code base
- Common baseline functionality for all services (include security and system management APIs)
- Dynamic configuration change callbacks
- Dependency service retries
- More unit, integration and system tests
- ?

Priority Platforms

- Target hardware hosts for test?
- What Linux and Windows variants?



Native Data Model

- The EdgeX architecture can accommodate virtually any data input and output; however, order to establish a certification program in the open source collaboration project there must be a native data model established in Core Services
- During incubation of FUSE seed code Dell created a simple, flat model in order to be agnostic to an eventual community-led selection
- Selection of data model will be an organic community effort, intent is to lock on a baseline by Beta release
- *Recommend we table this discussion until community is more familiar with code base.*

Container Orchestration

- Final reference solution should accomodate various container implementations (e.g. Docker, Snaps, etc.)
- Expand on current Docker Compose reference implementation?
- Should explore Kubernetes
- Does this fit under system management working group?
- ?

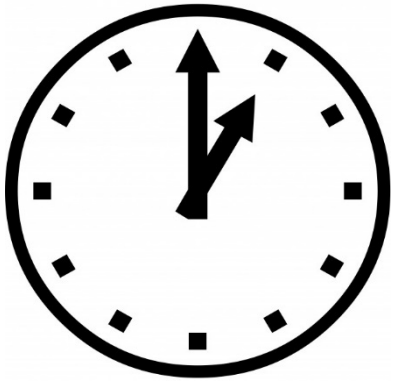
For Future Discussion:

Intersecting Key Tech Trends

- Blockchain
- LBS
- AR
- Voice
- Etc.

Addressing Real Time/Near Real Time (2018+)

- Alternative implementations of microservices (e.g. Go Lang, C)
- Consolidating microservices
- Use of better (QoS) and faster inter-service communications like DDS
- Move to RTOS to support the platform for some use cases

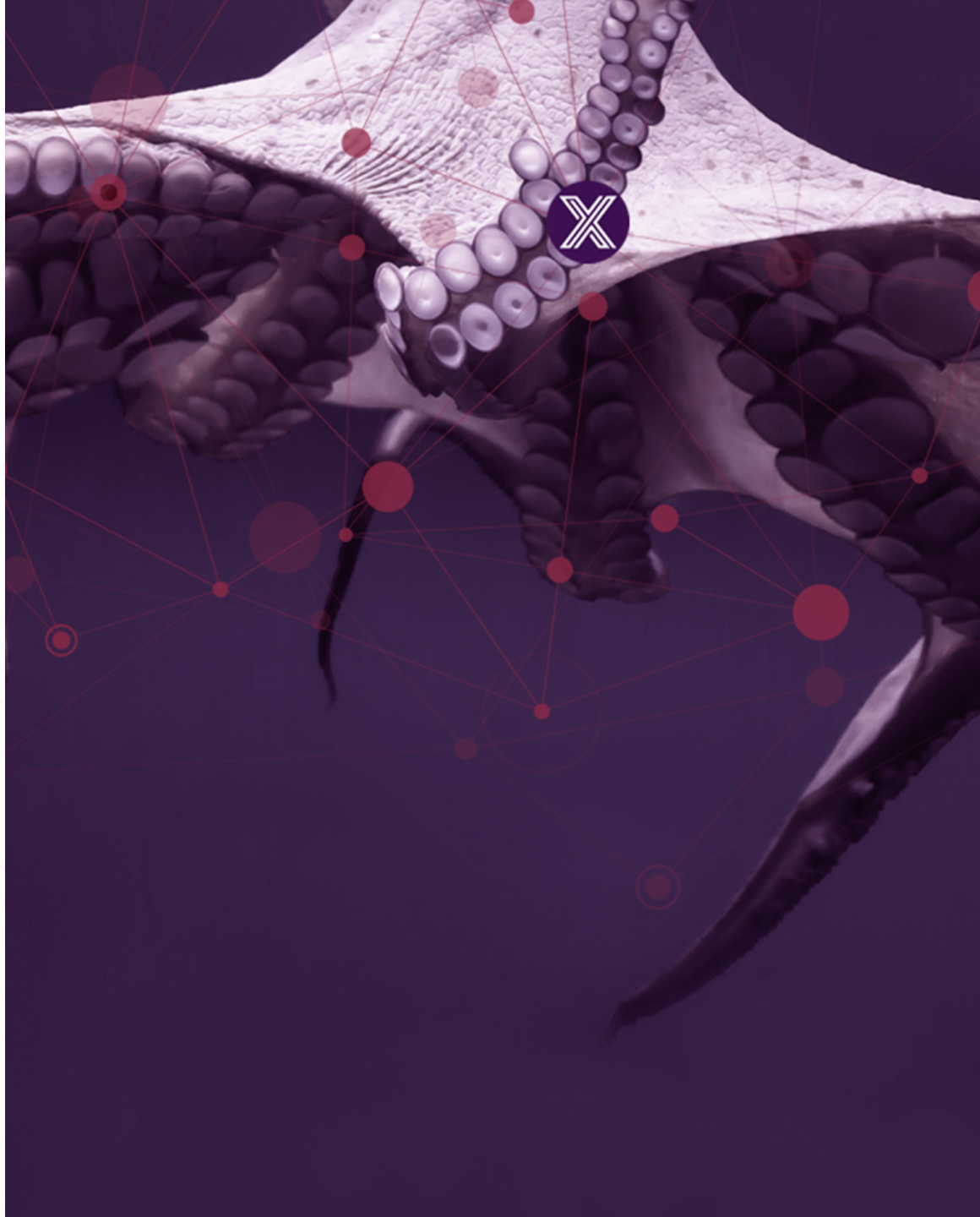


Proposed Goals for Beta Release: Core Architecture

- Finalize technology choices
- Performance optimizations (footprint, latency, stability)
- Implementing Minimum Viable Product (MVP) security and system management features
- Establish native data model
- Alternative code implementations for microservices
- Plan for path to more real time
- Improve documentation
- ?

EDGE X FOUNDRY™

Devices



Discussion: Priority Device Services

- **Current reference implementations**
  - BACnet
  - Modbus
  - MQTT
  - BLE
  - OPC-UA
  - SNMP

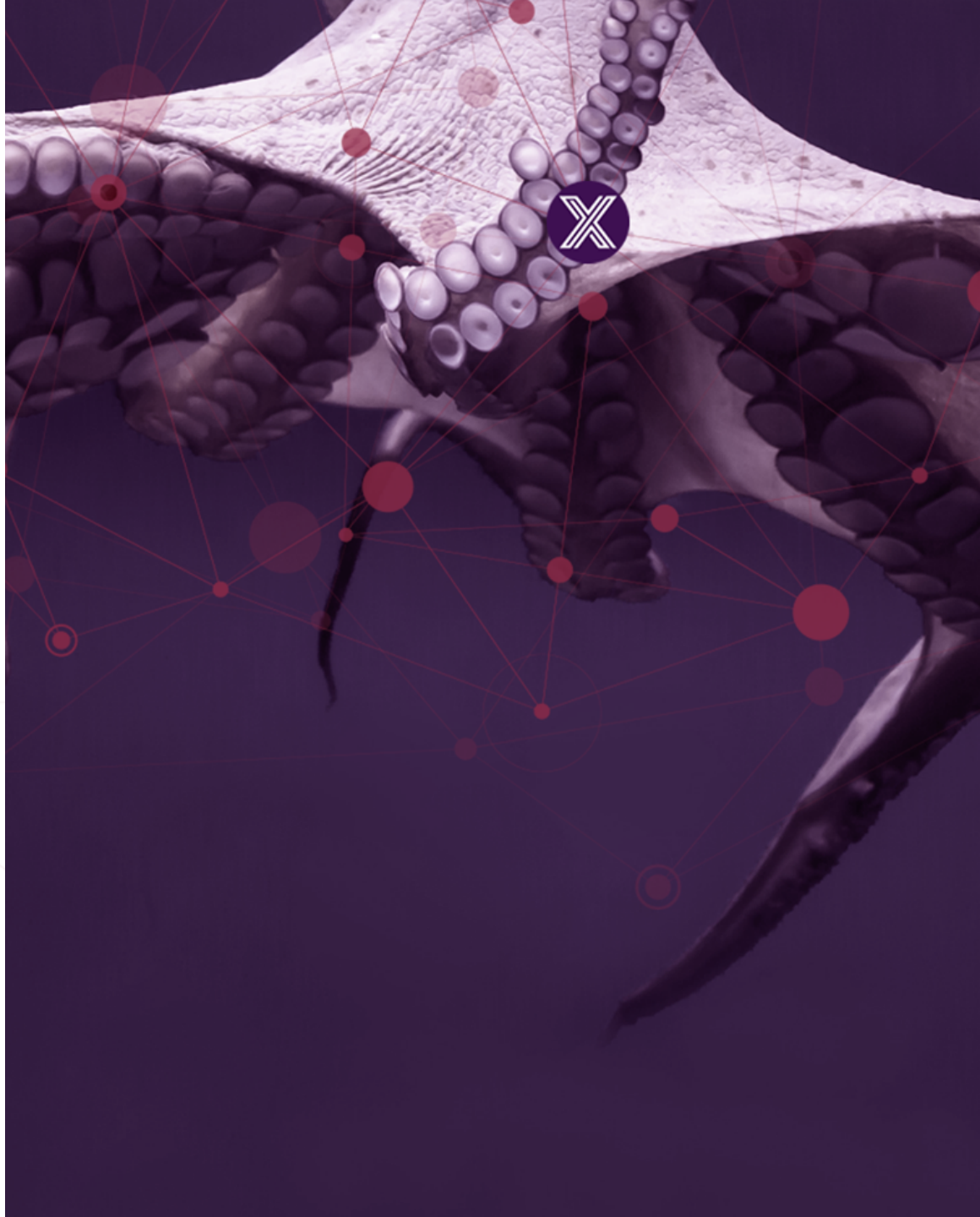
- **DISCUSSION: Priorities for Addition**
  - CAN bus
  - ULE
  - EnOcean
  - OCF
  - Thread
  - Zigbee
  - ?

Proposed Goals for Beta Release: Devices

- Guidance on performance requirements
- Improve robustness of Java Device Services SDK (refactor/simplify)
- Inter-protocol interoperability
- Explore alternative language implementations for SDK (e.g. C)
- Additional protocol reference implementations for open source use (includes guidance on licensing considerations)
- Define process/considerations for dev of proprietary Device Services with open SDK
- Improve documentation
- ?

EDGE X FOUNDRY™

Applications



Discussion: Priority Export Services

Current Reference Implementations

- MQTT
- Microsoft Azure MQTT

Discussion: Priorities for Addition

Standards

- Haystack
- OPC-UA
- ?

Proprietary / Commercial

- AWS / Greengrass
- Google IoT Core
- SAP HANA
- IBM Watson IoT
- ?

Additional proprietary implementations will be up to private companies.

Addressing the “Fog” (2018+)

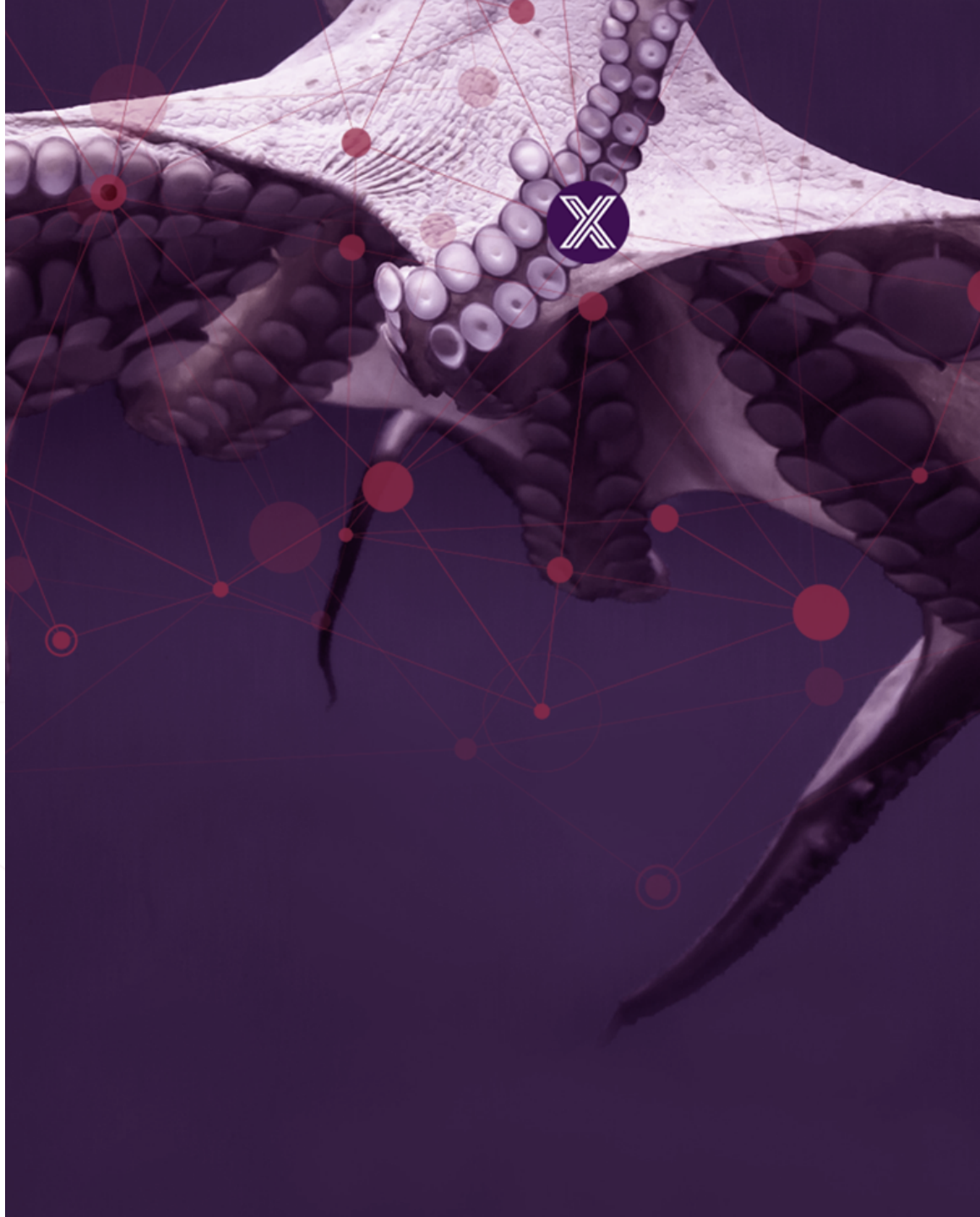
- Facilitate East-West capability
  - Scale/Load balance across EdgeX instances
  - Failover between EdgeX host systems
  - Cluster of EdgeX node management
- Facilitate North-South capability
  - Gateway (EdgeX) Device Service
  - Gateway (EdgeX) to Gateway (EdgeX) export
  - Export/integration with other node types

Proposed Goals for Beta Release: Applications

- Guidance on MVP deployment and associated performance requirements for applications
- Refine Export Services SDK
- Create additional export implementations
- Evaluate “fog” requirements including east-west data orchestration as well as failover, redundancy and load balancing

EDGE X FOUNDRY™

Security



EdgeX Foundry Security Services – Proposed Approach

- EdgeX security services are envisioned to be architected in a fashion consistent with other EdgeX services
  - APIs define the fundamental capability
  - Various implementations open to the implementer (some may be proprietary)
  - A reference implementation should be provided for basic capability
  - Use of open source technologies may be used to provide reference implementations
- Three security microservices are key to providing the bulk of EdgeX security capability
  - Other security services are layered on top of these 3 services
  - Details and APIs to be defined

EdgeX Foundry Security Services

EdgeX Base Security Services

- Trusted Execution Service
  - Attests executable code elements (from EdgeX) are legitimate, authenticated, and can be trusted
  - Maintains a registry of trusted/uncompromised services
  - May have some connection/integration to existing EdgeX Config/Registry service
- Identity Management Service
  - Provides authentication and authorization services
  - Verification that an entity is who/what it claims to be
  - Governs what can an entity access (with regard to other EdgeX services)
- Key Management Service
  - Generate, exchange, store, and destroy cryptographic keys

Additional EdgeX Security Services

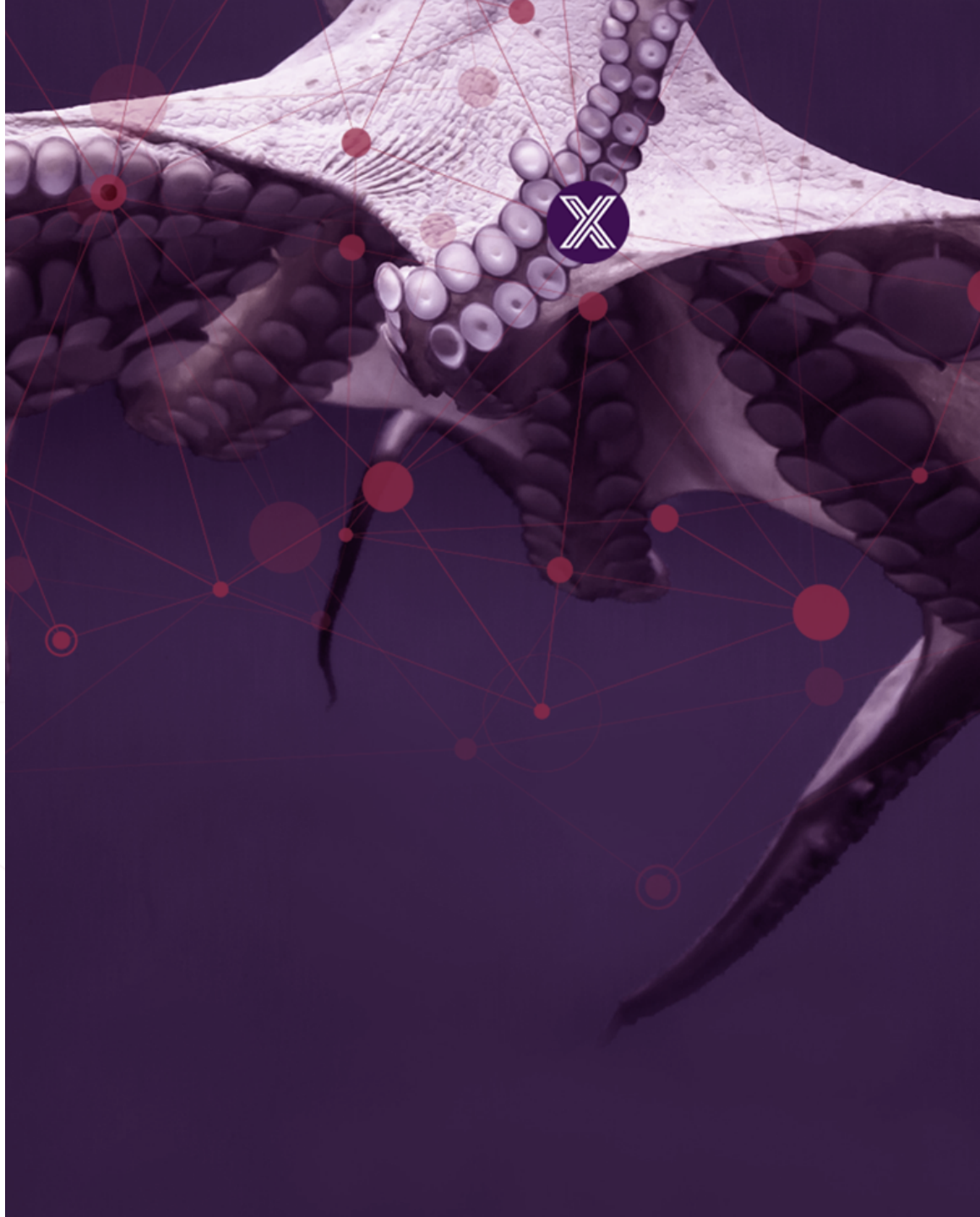
- Additional EdgeX Security Services may be required
- Most of these additional services would use or be based on the 3 base services
- Examples include
  - Data Protection service – sign and encrypt data at rest
  - Log Monitor service – monitor EdgeX logs for suspicious security activity and report/alert when security violations are detected or suspected
  - Message Protection service – sign and encrypt data in motion
    - HTTPS and key management capability is envisioned to secure EdgeX REST communications
    - Additional message protection services are envisioned to encrypt and protect data in other message protocols used in EdgeX (MQTT, AMQP, DDS, etc.)
- Need to balance reference vs. opportunity for 3rd party value-add

Proposed Goals for Beta Release: Security

- Agreement on security requirements and potential high-priority EdgeX Foundry threats that must be addressed, establish key tenets
- Recommendation on MVP security service APIs
- Reference implementation for MVP services to include Key Management, Identity Management, Code Validation services
- Assist in security integration to the security services in other / existing EdgeX microservices
- Develop security test plans and security testing procedures
- Documentation
- ?

EDGE X FOUNDRY™

System Management



System Management (1 of 2)

General Management of Host Hardware (need to define boundary of scope)

- Power Cycle
- Provide data about the operation and status/health of the host hardware
- Setup, configuration, and modification of Ethernet and wireless connections
- Support multiple platforms/OS (Linux, Windows, x86, ARM, …)
- Management of OS and BIOS likely out of scope but EdgeX must persist updates to underlying OS, I/O device drivers and BIOS

Management of Connected Devices

- Discovery of new devices
- Update Device Service
- Update device firmware
- Report on device status/health

EdgeX Supporting Software Management

- Install/uninstall, start/stop & configure databases
- Install/uninstall, start/stop & configure message brokers/message systems
- Base service implementation in all micro services
- Defines interface and API hooks for start, stop, restart, etc. of services

System Management (2 of 2)

EdgeX Micro Service Registry & Management

- Recognize and report on installed EdgeX services
- Install/uninstall services
- Start/Stop services
- Report service health/status (heartbeat, revision, responsiveness, etc…)
- Update service (provide rollback in some cases)

Manage configuration (add, update, delete)

- Define the port a service runs on
- Seed service data (example: addressable for device services)
- Notify services of other service state changes
- Store/understand/manage microservice dependency information (example: point all services to new security service provider)
- Manage blacklist services (turn off a service for maintenance, etc.)
- Assist device services provision/remove new devices
- Discover new devices being connected

User interface to provide administration

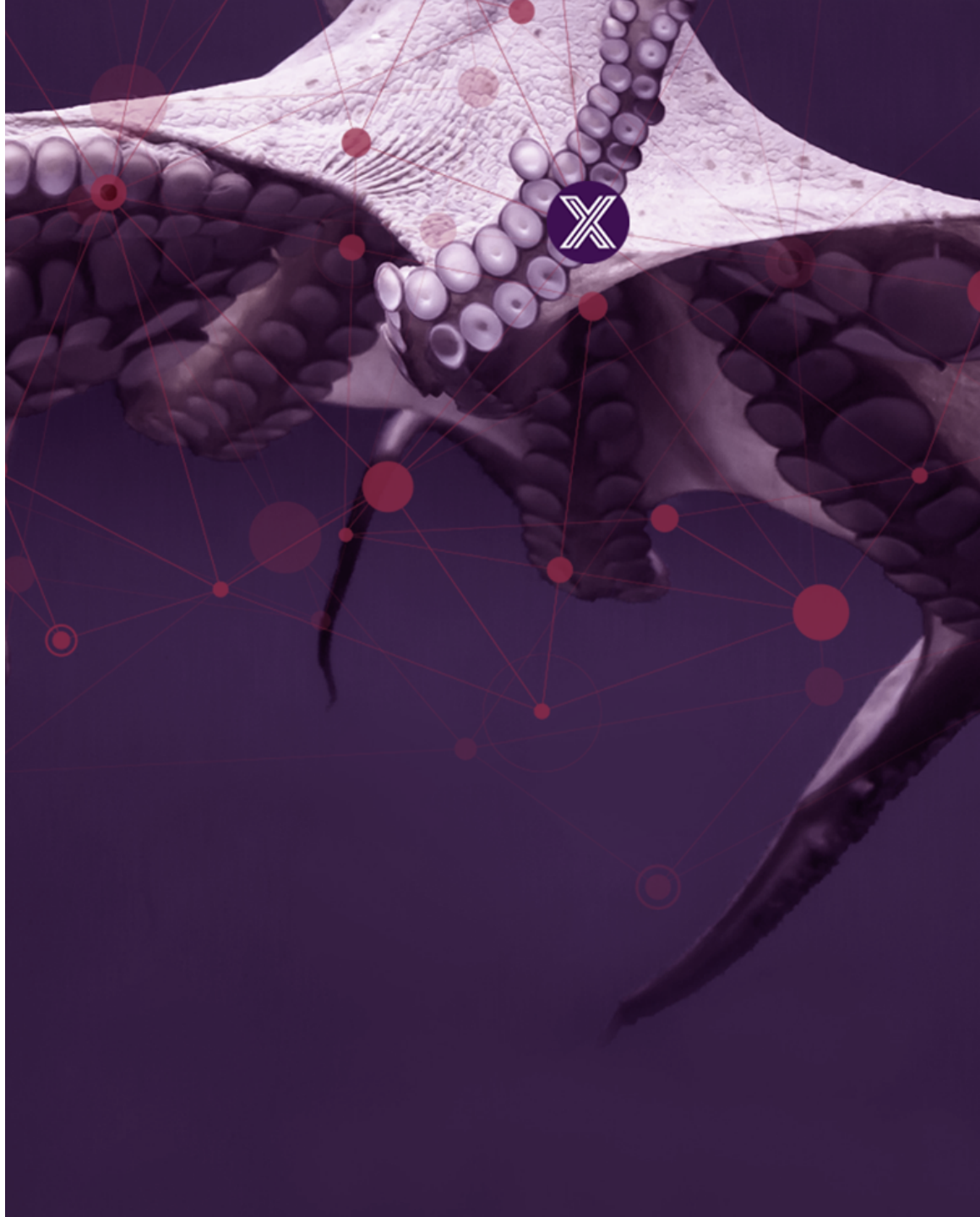
- Create "single pane of glass" reference implementation for managing the host, connected devices and EdgeX leveraging existing standards
- Test integrations with commercial sys mgmt products (Dell Edge Device Manager, VMware Pulse, System Center, etc.)

Proposed Goals for Beta Release: System Management

- Clear definition of what's in scope for EdgeX vs. all 3rd party
- Establish key tenets for system management
- Recommendation for MVP system management services and APIs
- Assist with implementation of APIs in code base
- Create reference implementation for remote and local consoles
- Test EdgeX APIs with various 3rd party consoles
- Documentation
- ?

EDGE X FOUNDRY™

Working Group Breakouts



Breakout Organization

Main Room:

- Combined: Core Architecture, Applications and Devices

Two Satellite Rooms:

1. Security
2. System Management

Breakout Considerations

Tips:

- Leverage starter slides in previous section and incorporate feedback from today's group alignment.
- Discuss in context of current code base and your potential contributions
- Consider other key functional areas like QA, DevOps, Commercial Enablement, Business Needs, etc.
- Balance tactical and strategic project goals

Checklist for Readout on Day 2

- ❑ Overall working group priorities, key tenets
- ❑ Recommendations on key technologies to consider
- ❑ Recommended MVP functionality for Beta (prioritized for organization into sprints)
- ❑ Suggested key milestones on the path to Beta
- ❑ Recommended action plan for first 30-day sprint
- ❑ Proposed resourcing / member contributions
- ❑ Proposal for final TSC structure, working groups and sub-projects
- ❑ What you need from other working groups

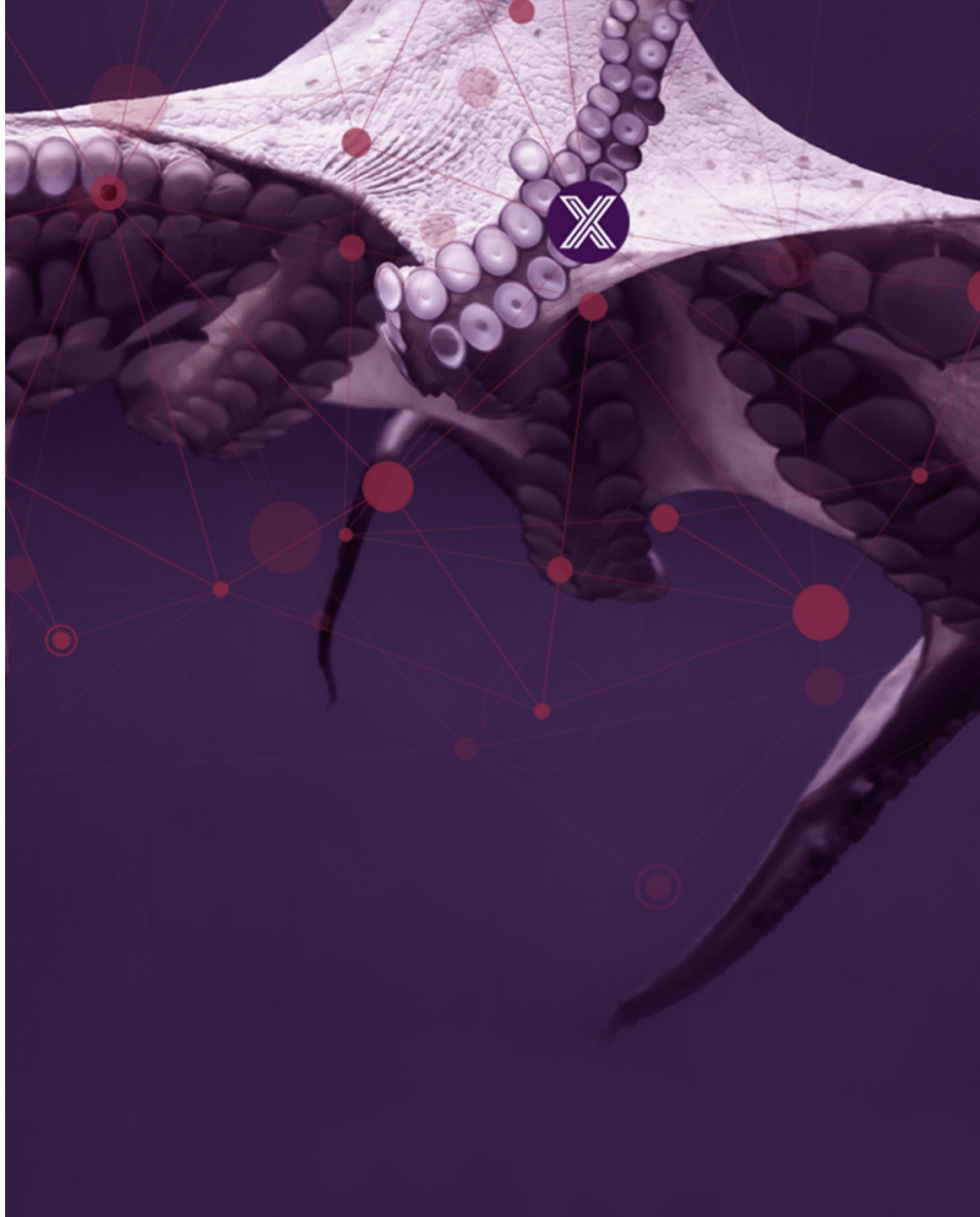
EDGE X FOUNDRY™

Technical Workshop

Day 2: June 2, 2017

EDGE X FOUNDRY™

Working Group Readouts



Breakout Readouts

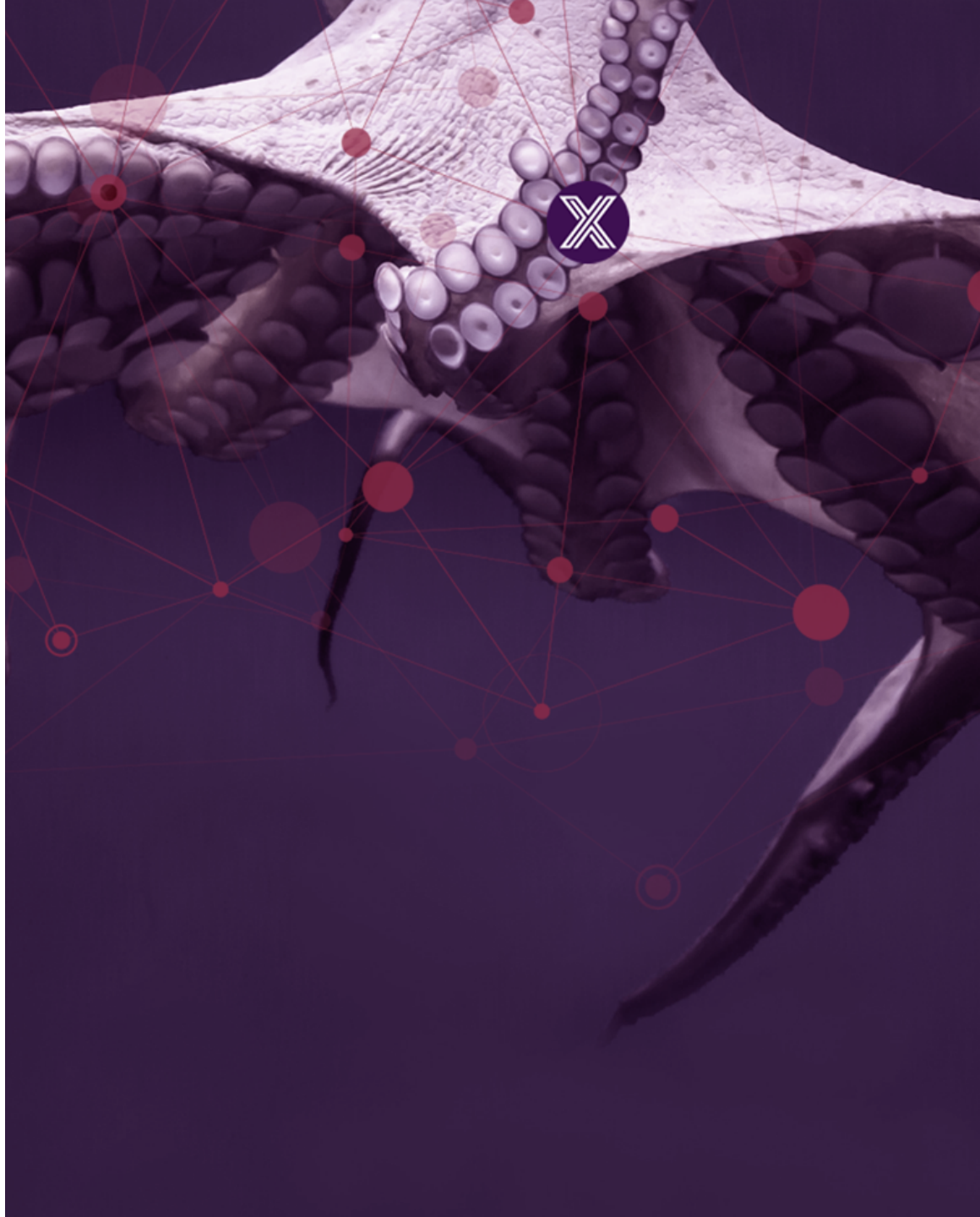
Working Groups to report on based on the checklist.

- 60 minutes: Core Architecture, Applications, Devices
- 20 minutes: Security
- 20 minutes: System Management

30 minutes to assimilate output

EDGE X FOUNDRY™

Next Steps



TSC Structure

Working Groups

Potential Projects

Core | Applications | Security | System Management | Devices | DevOps? | QA?

Performance/ Benchmarking | Commercial Enablement | Certifications | Engagement | Education | Interop (IoTX) | ?

Maintaining Stability in Interop Framework

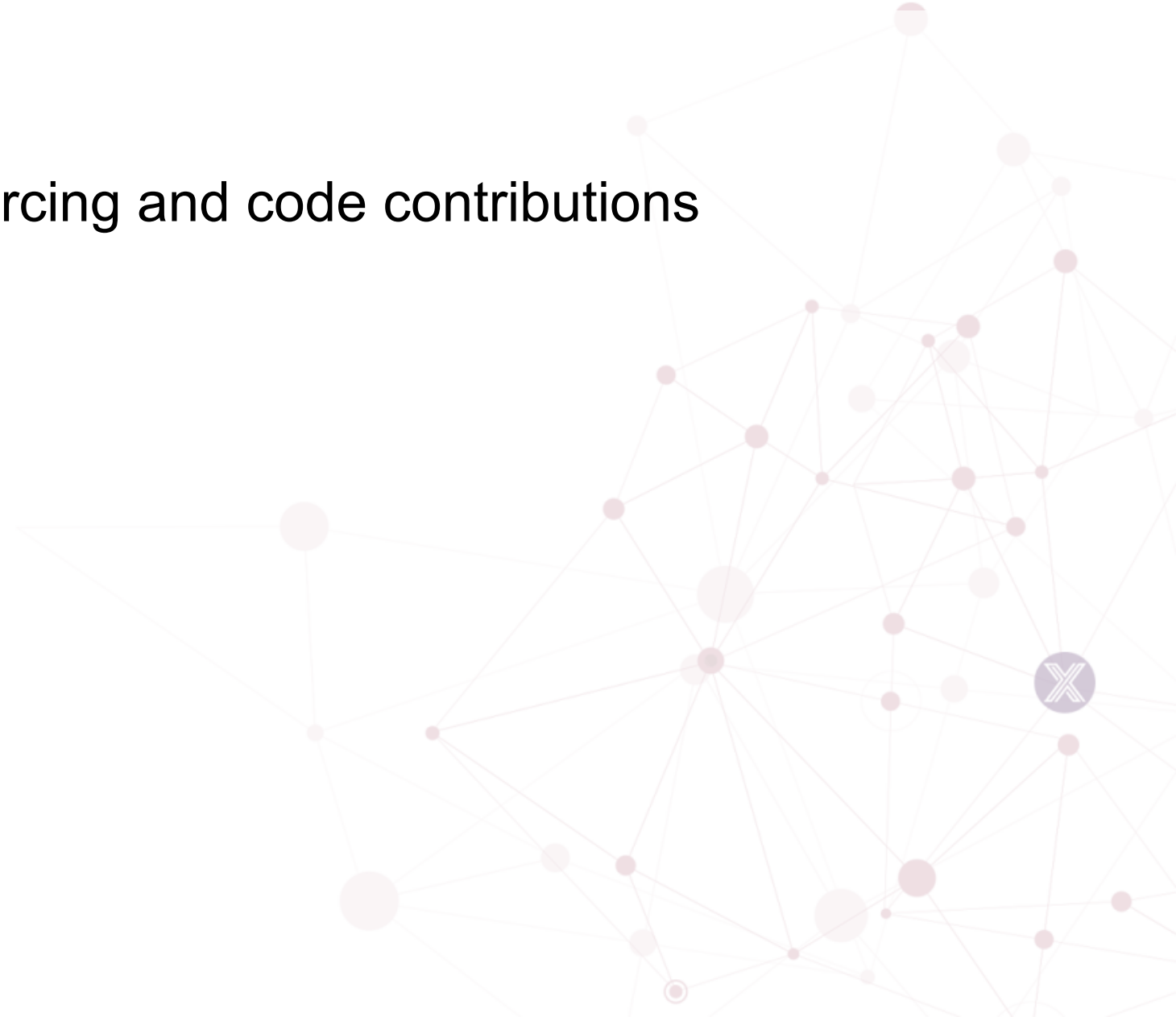
- Open discussion

Future Certification Program

- Timeline?
- Next steps?

Resources

- Open discussion on resourcing and code contributions



Meeting/Development Structure

- Meeting Cadence/Structure
- Development Sprints

Summarize and Adjourn