Project EVE

Providing zero touch, zero trust, for any app on any network

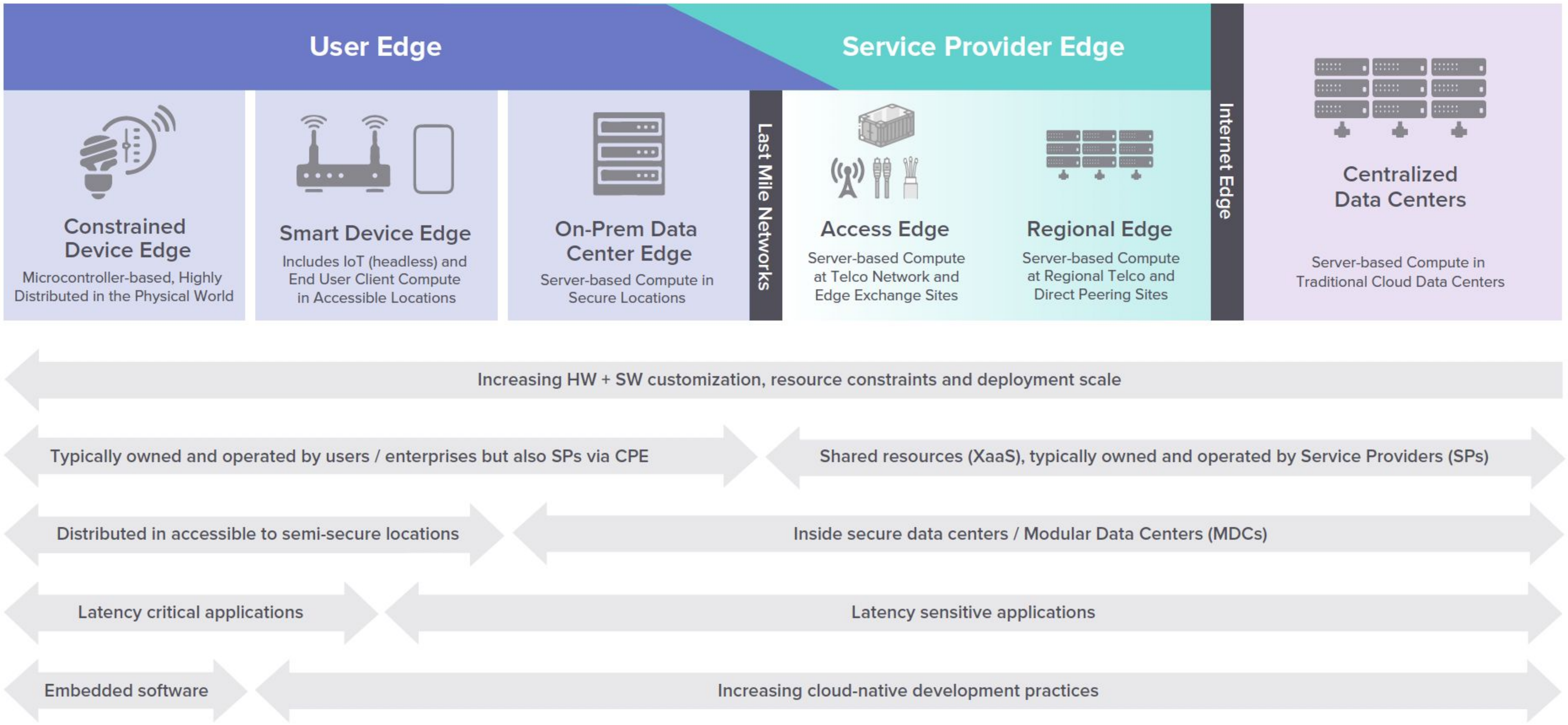
Erik Nordmark, Chief Architect, ZEDEDA

Roman Shaposhnik, VP Product & Open Source, ZEDEDA

THE LINUX FOUNDATION

The Edge, EVE, and LF-Edge

Edge means different things to different people



User Edge

Service Provider Edge

Constrained Device Edge
Microcontroller-based, Highly Distributed in the Physical World

Smart Device Edge
Includes IoT (headless) and End User Client Compute in Accessible Locations

On-Prem Data Center Edge
Server-based Compute in Secure Locations

Last Mile Networks

Access Edge
Server-based Compute at Telco Network and Edge Exchange Sites

Regional Edge
Server-based Compute at Regional Telco and Direct Peering Sites

Internet Edge

Centralized Data Centers
Server-based Compute in Traditional Cloud Data Centers

Increasing HW + SW customization, resource constraints and deployment scale

Typically owned and operated by users / enterprises but also SPs via CPE

Shared resources (XaaS), typically owned and operated by Service Providers (SPs)

Distributed in accessible to semi-secure locations

Inside secure data centers / Modular Data Centers (MDCs)

Latency critical applications

Latency sensitive applications

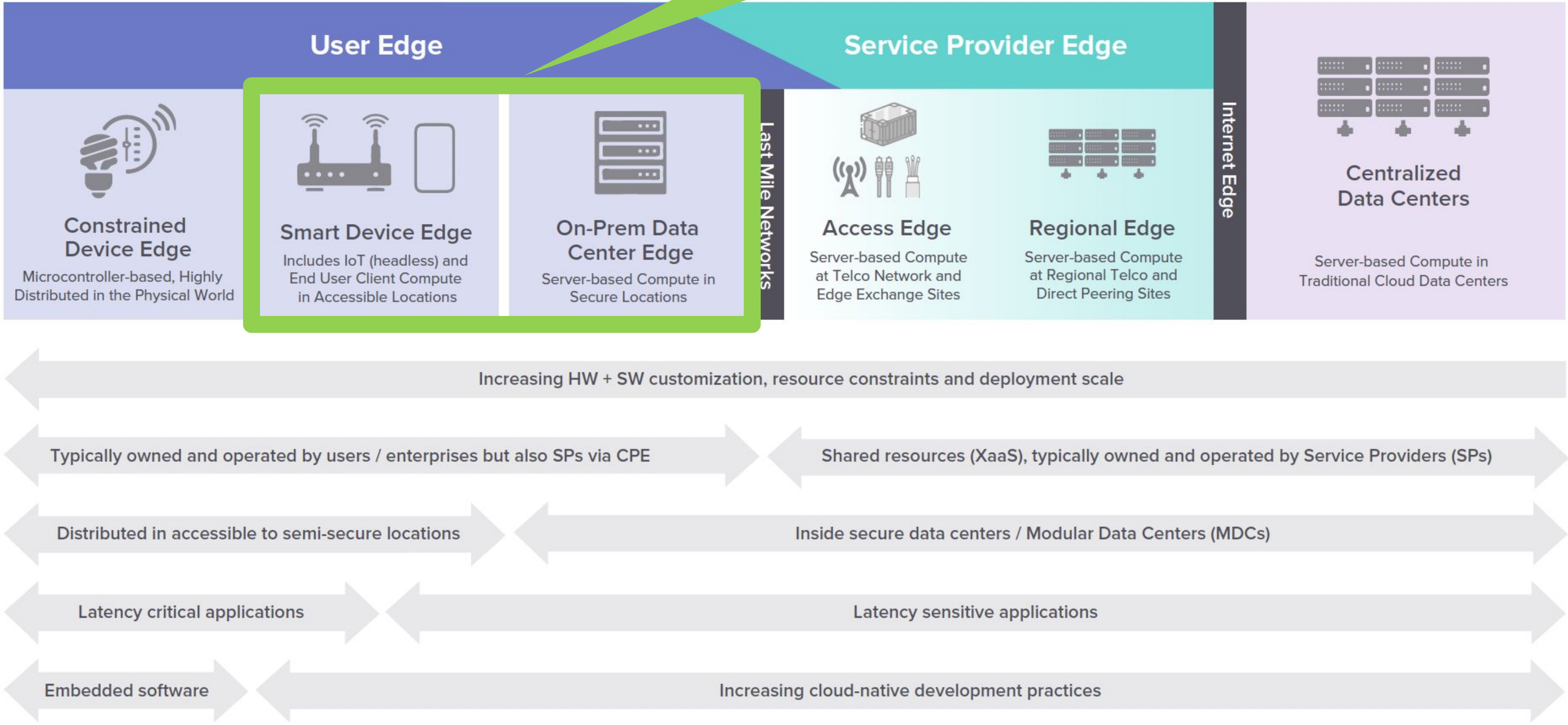
Embedded software

Increasing cloud-native development practices

See https://www.lfedge.org/resources/publication-download/

Fit in Edge Continuum

Project EVE is focused on IoT workloads at the Smart Device Edge



User Edge

Service Provider Edge

Constrained Device Edge

Microcontroller-based, Highly Distributed in the Physical World

Smart Device Edge

Includes IoT (headless) and End User Client Compute in Accessible Locations

On-Prem Data Center Edge

Server-based Compute in Secure Locations

Last Mile Networks

Access Edge

Server-based Compute at Telco Network and Edge Exchange Sites

Regional Edge

Server-based Compute at Regional Telco and Direct Peering Sites

Internet Edge

Centralized Data Centers

Server-based Compute in Traditional Cloud Data Centers

Increasing HW + SW customization, resource constraints and deployment scale

Typically owned and operated by users / enterprises but also SPs via CPE

Shared resources (XaaS), typically owned and operated by Service Providers (SPs)

Distributed in accessible to semi-secure locations

Inside secure data centers / Modular Data Centers (MDCs)

Latency critical applications

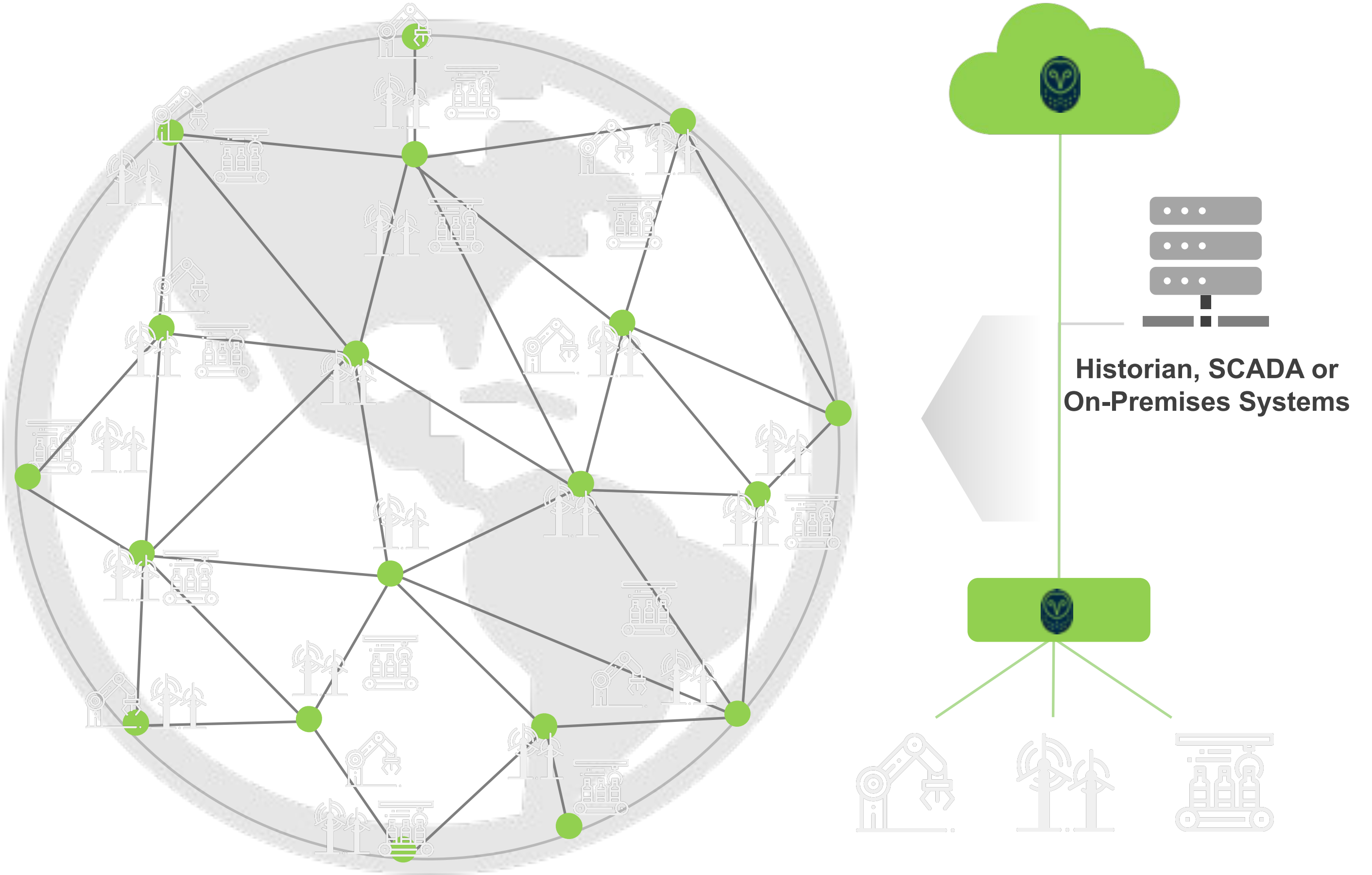
Latency sensitive applications

Embedded software

Increasing cloud-native development practices

Challenges at the User Edge

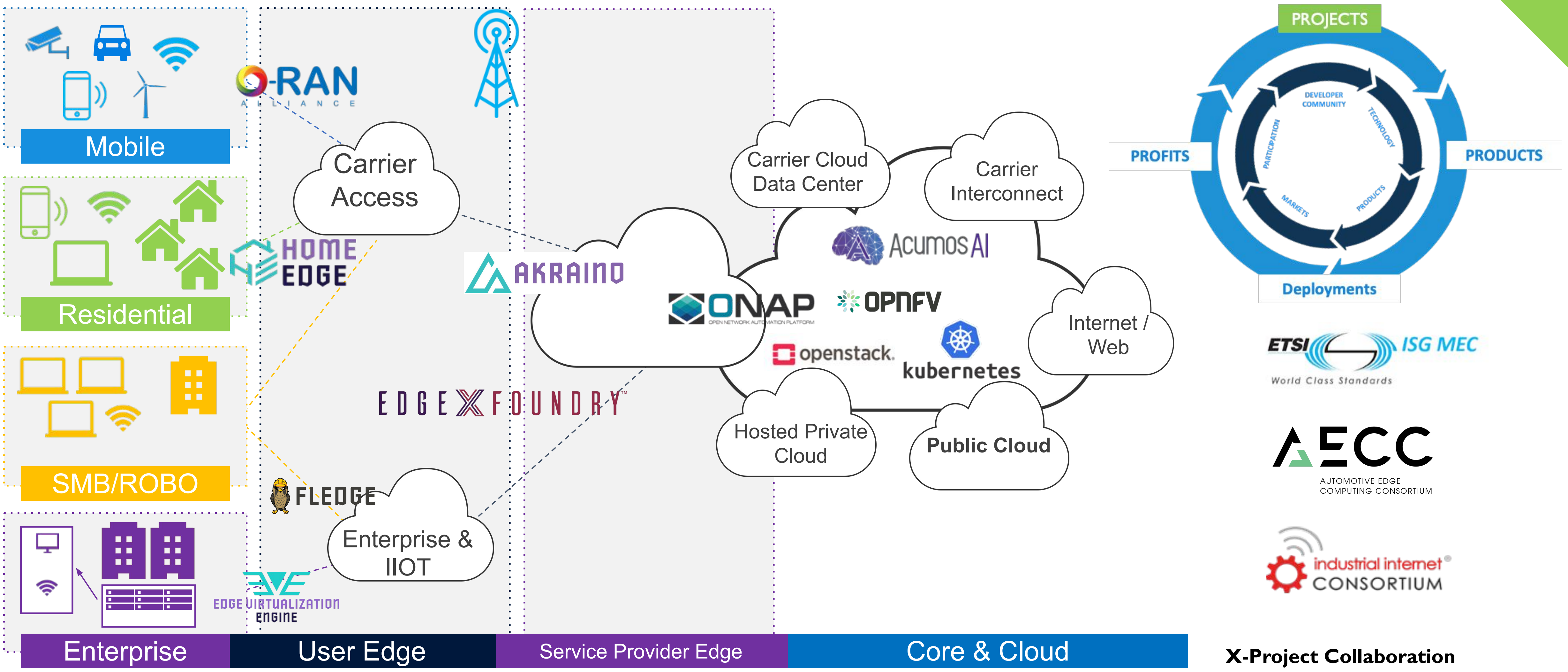
- **Diversity of gateways and apps**
  - Infrastructure management
  - Orchestration of apps
  - Some apps with cloud assumptions
- **Scale and automation**
  - Geographically disperse
  - Deployment and maintenance
  - Long deployment lifecycle 7+ years
- **Security – increased threat vector**
  - No perimeter network security
  - No perimeter physical security
  - Varying requirements - OT and IT
- **Diverse connectivity**
  - Upstream and downstream
  - Might not control enterprise network



LF Edge - the end to end context

Deployment ready Open Source - use cases

OSS+SDO



LF Edge Summary

Vision: Our software & projects enable rapid productization of Edge platforms by leveraging end user input to drive and supply the necessary building blocks (and/or frameworks, reference solutions) to facilitate integration and interoperability for Edge Computing across Telecom Service Providers, Cloud Providers, IOT & Enterprises

Projects

IMPACT - STAGE 3

AKRAINO EDGE X FOUNDRY™

GROWTH - STAGE 2

EVE FLEDGE HOME EDGE STATE OF THE EDGE

AT LARGE - STAGE 1

BAETYL OPEN HORIZON SECURE DEVICE ONBOARD

Premier Members

altran arm AT&T Bai du 百度 Charter COMMUNICATIONS DELL Technologies DIANOMIC EQUINIX ERICSSON

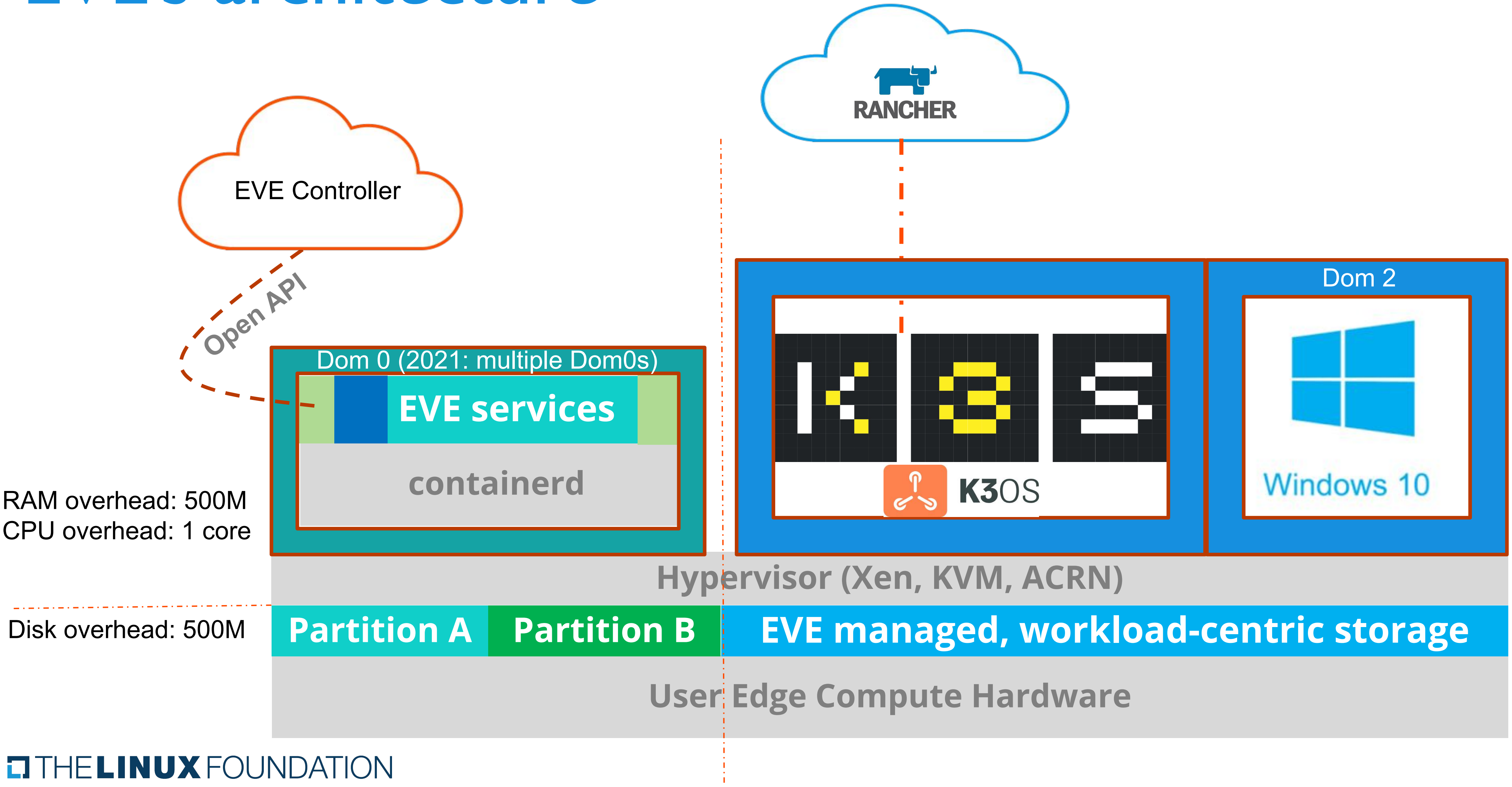
FUJITSU FUTUREWEI Technologies hp HUAWEI IBM intel NOKIA NTT OSIsoft.

Radisys redhat SAMSUNG Tencent 腾讯 WD Western Digital wipro ZEDEDA

EVE Introduction and Security

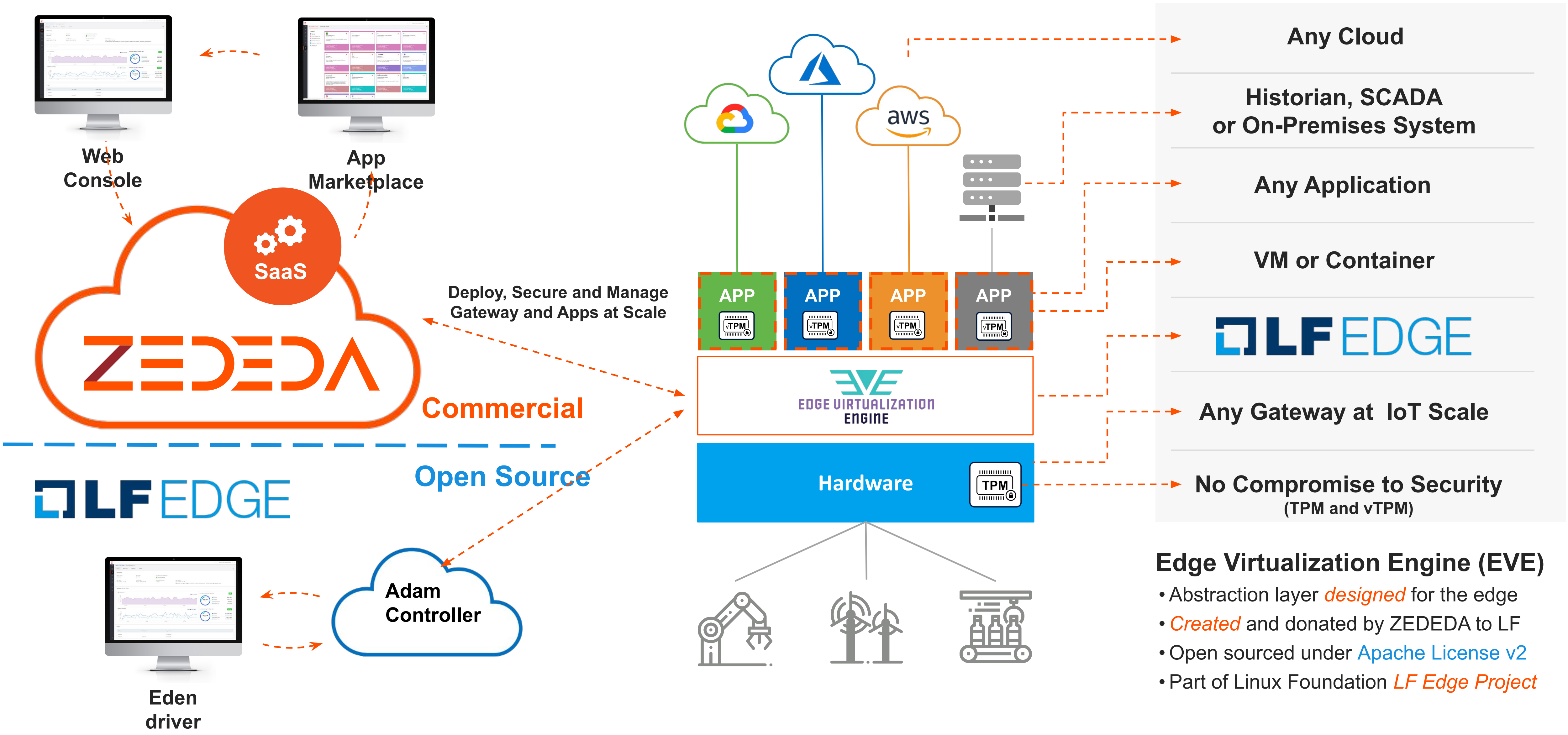
THE LINUX FOUNDATION

EVE’s architecture



THE LINUX FOUNDATION

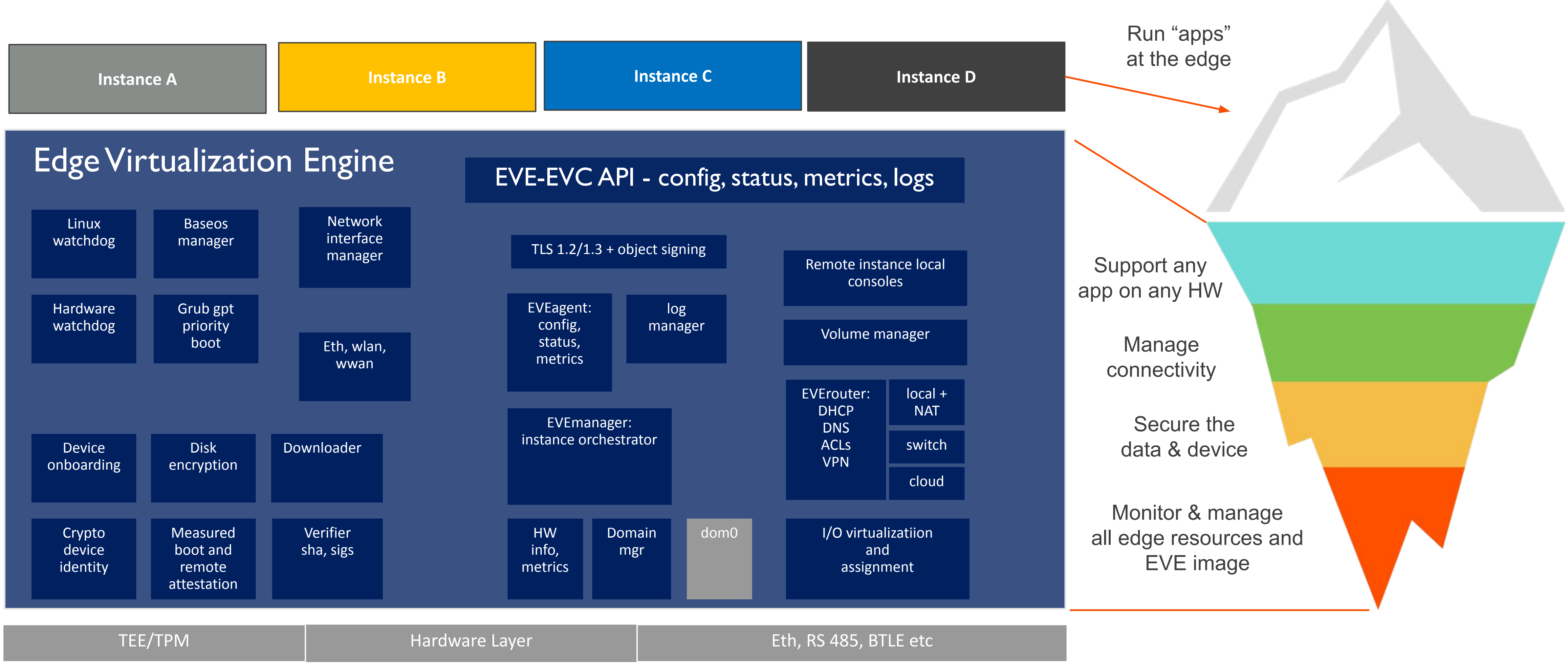
Challenges Solved with Edge Virtualization



Edge Virtualization Engine (EVE)

- Abstraction layer *designed* for the edge
- *Created* and donated by ZEDEDA to LF
- Open sourced under Apache License v2
- Part of Linux Foundation *LF Edge Project*

App deployment is but the tip of the iceberg



Instance A

Instance B

Instance C

Instance D

Edge Virtualization Engine

EVE-EVC API - config, status, metrics, logs

Linux watchdog

Baseos manager

Network interface manager

Hardware watchdog

Grub gpt priority boot

Eth, wlan, wwan

Device onboarding

Disk encryption

Downloader

Crypto device identity

Measured boot and remote attestation

Verifier sha, sigs

TLS 1.2/1.3 + object signing

EVEagent: config, status, metrics

log manager

EVEmanager: instance orchestrator

HW info, metrics

Domain mgr

dom0

Remote instance local consoles

Volume manager

EVErouter: DHCP DNS ACLs VPN

local + NAT

switch

cloud

I/O virtualizatiion and assignment

TEE/TPM

Hardware Layer

Eth, RS 485, BTLE etc

Run “apps” at the edge

Support any app on any HW

Manage connectivity

Secure the data & device

Monitor & manage all edge resources and EVE image

THE LINUX FOUNDATION

Common Insertion Points for EVE

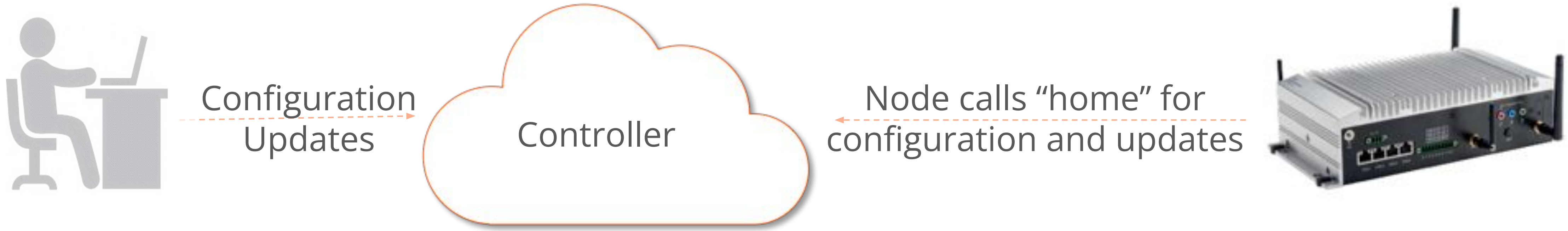
- Application/container is already working at small scale
  - Cloud connectivity etc worked out
  - Need to deploy at much larger scale with less manual work
  - Need to operate at scale handling day 2 issues (patch, update, etc)
- Mixture of legacy application (Linux, Windows) and new
  - Desire to run legacy as VM, while deploying containers/clusters
- Deploying containers but concerned about edge security
  - Hardware root of trust; firewall rules; VPN integration
  - How to securely update container runtime and OS
- Need richer connectivity for containers or VMs
  - Edge-to-edge, VPN to cloud

Zero Touch

- Enable drop ship to installer
  - Factory/supply chain installs EVE; handles unique device identity
  - Installer connects power and network/serial cables
  - Visual feedback to installer that device connected to controller in cloud
- Everything else done from the cloud
  - Edge container lifecycle (install, update, pause, snapshot)
  - Device lifecycle (EVE patch/update, EVE connectivity changes)
  - Without any risk of turning the device into a brick
- Only broken hardware or cabling changes requires touching the device

Remotely Manage Any Edge Node

No field expertise required



Configuration Updates

Controller

Node calls “home” for configuration and updates

- Any type of silicon and node
- Automated on-boarding
- Autonomous operations

Any Silicon

intel NVIDIA ARM

Any User Edge Compute Node

ADVANTECH *Enabling an Intelligent Planet* DELL Hewlett Packard Enterprise Lanner SUPERMICRO

Any Edge Node

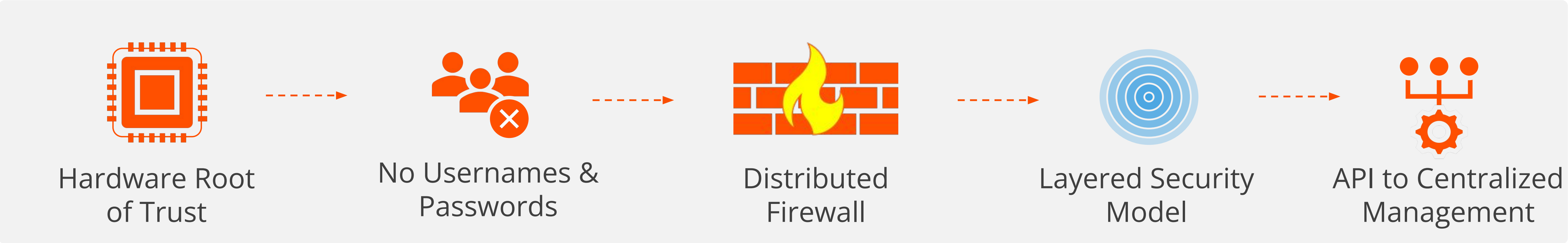
- EVE today supports ARM and Intel/AMD
  - Requires processor support for type 1 hypervisor (VT-x etc)
- Supports a range of upstream and downstream IP connectivity
  - Ethernet, WiFi, LTE, and anything else supported by Linux
- Supports a range of downstream I/O connectivity
  - RS-232, RS-485 serial ports
  - USB, Audio, etc
- Runs any application (Edge Container)
  - Existing VMs, containers, clusters (including EdgeX Foundry, Fledge, Azure IoT Edge, AWS Greengrass Core), future Unikernels
  - Applications are not concerned with the variations in IP connectivity

Security threats at the User Edge

- User access - poor usernames/passwords
- Physical access
  - USB stick, ethernet cable
- Theft
  - Disk/SSD
  - Clone device
- Network
  - DDoS of device
  - Attacks exploiting software bugs in OS/runtime
- Device becoming part of botnet attacking others

Zero Trust

People, Process and Technology



Hardware Root of Trust → No Usernames & Passwords → Distributed Firewall → Layered Security Model → API to Centralized Management

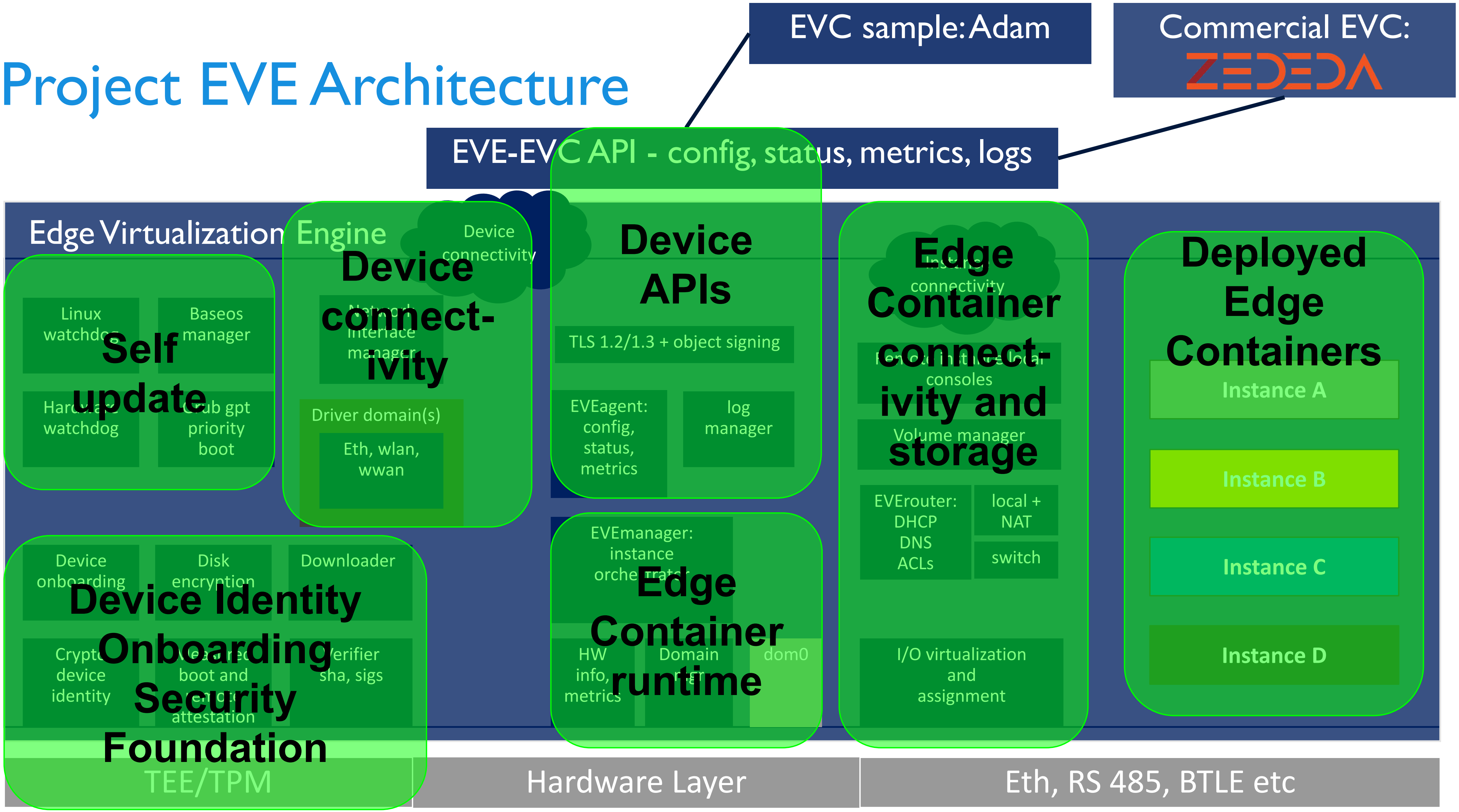
- People
  - Remove need for device usernames/passwords
  - RBAC and multi-tenancy in controller
- Processes - handle 7+ year lifetime at edge
  - Secure, scalable distribution of updates
  - API reports (resource usage, firewall violations) enable analytics in controller

- Standard Security Technologies for the User Edge
  - Hardware root of trust (e.g., TPM)
  - Crypto-based identification
  - Measured boot and remote attestation
  - Encryption at rest and in-flight (TLS); keys sealed by TPM
  - Signed images for EVE-OS and applications
  - Use hypervisors for strong isolation and defense in depth
  - Distributed firewall for every app
  - Physical security—port isolation
  - Support deployment of virtual security appliances

EVE Architecture

THE LINUX FOUNDATION

Project EVE Architecture



EVC sample: Adam

Commercial EVC: ZEDEDA

EVE-EVC API - config, status, metrics, logs

Edge Virtualization Engine

Self update

Linux watchdog

Baseos manager

Hardware watchdog

Grub gpt priority boot

Device connect-ivity

Device connectivity

Network interface manager

Driver domain(s)

Eth, wlan, wwan

Device APIs

TLS 1.2/1.3 + object signing

EVEagent: config, status, metrics

log manager

Edge Container connect-ivity and storage

Instance connectivity

Remote instance local consoles

Volume manager

EVErouter: DHCP DNS ACLs

local + NAT

switch

I/O virtualization and assignment

Deployed Edge Containers

Instance A

Instance B

Instance C

Instance D

Device Identity Onboarding Security Foundation

Device onboarding

Disk encryption

Downloader

Crypto device identity

Measured boot and remote attestation

Verifier sha, sigs

TEE/TPM

Edge Container runtime

EVEmanager: instance orchestrator

HW info, metrics

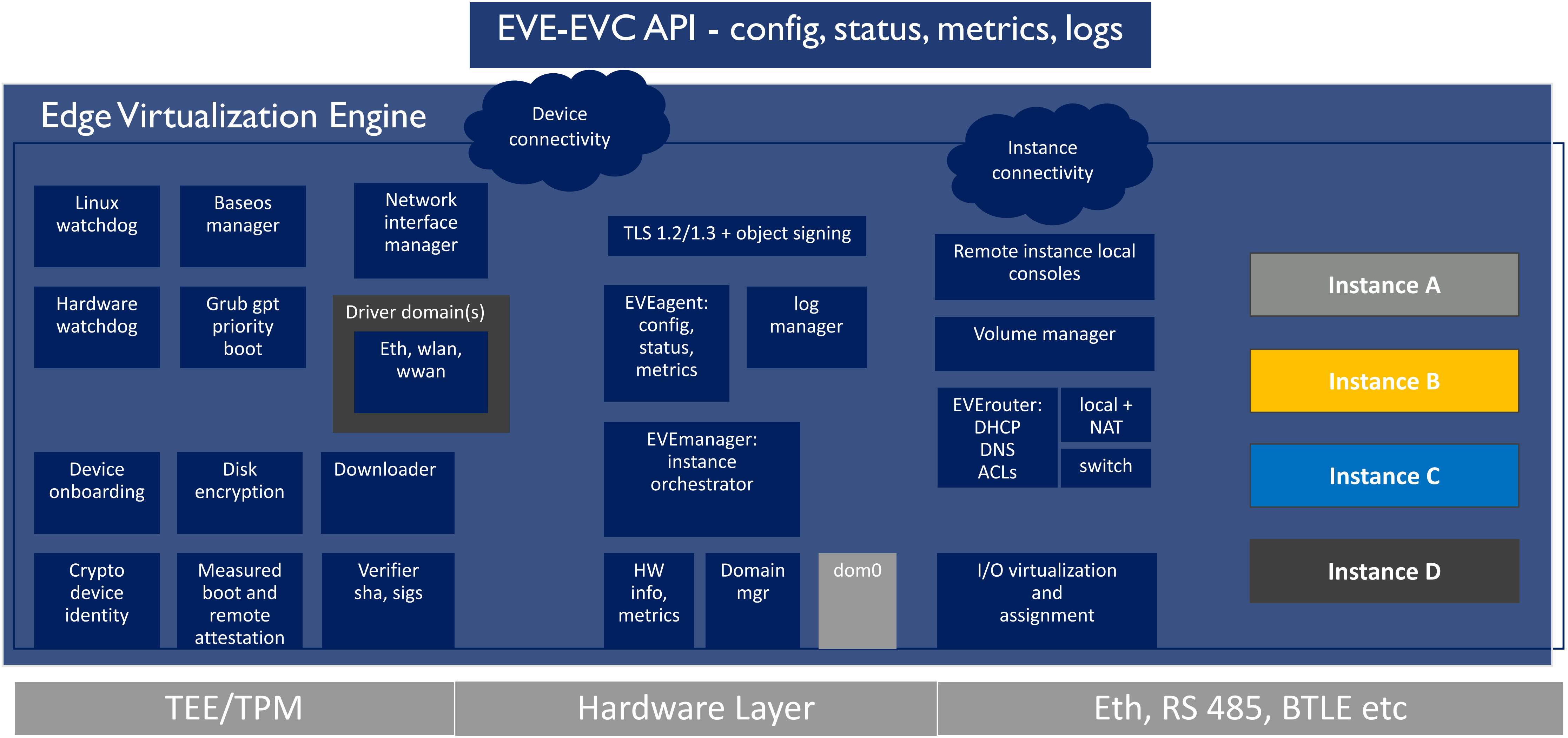
Domain mgr

dom0

Hardware Layer

Eth, RS 485, BTLE etc

Project EVE Architecture



Device Onboarding

- Cryptographic device identity is created when EVE installed (factory)
  - Key pair generated in TPM; private key never leaves TPM
  - Device is imprinted with the controller to trust (a root CA certificate)
- Different processes to extract device certificate, serial number(s) to ship with hardware (depends on hardware vendor)
- Device can be pre-onboarded in factory to pre-install application software content
- User registers their hardware using device certificate and/or serial number
  - Controller detects attempted duplicate registrations
- See https://github.com/lf-edge/eve/blob/master/docs/REGISTRATION.md

Device Boot

- EVE is supporting different boot firmware implementations
  - generic UEFI firmware on both x86 and ARM
  - legacy PC BIOS on x86 (such as for Google Compute Platform)
  - open source Coreboot via the legacy PC BIOS payload
  - board specific u-boot firmware (such as on Raspberry Pi ARM platform)
- Uses GPT partition tables with A/B boot partitions for failover
- Performs measured boot and remote attestation
  - Different measurements: require remote attestation to controller to unlock application disks
  - Same measurements: unlock and start applications even without controller connectivity
  - See https://wiki.lfedge.org/display/EVE/Measured+Boot+and+Remote+Attestation
  - Detects rouge firmware and unsupported EVE builds
- See https://github.com/lf-edge/eve/blob/master/docs/BOOTING.md

Device Connectivity - Network Interface Manager

- Device must have some connectivity to the controller
  - Can be redundant e.g., Ethernet plus LTE
  - Can be active/active or active/standby
- Default is to initially try all Ethernets with DHCP to reach controller
- Can be overridden with a file on a USB key specifying
  - static IPs, http proxies, WiFi credentials, etc
- Once controller is reached the controller will specify the device connectivity parameters
- Any change to the parameters is tested by EVE
  - verify controller is reachable before committing to new parameters
- See https://github.com/lf-edge/eve/blob/master/docs/DEVICE-CONNECTIVITY.md

EVE Self Update - BaseOS manager

- Update all of EVE-OS including hypervisor
- Handle any failures
  - Power failure when writing to flash
  - Bad new EVE image resulting in not being able to connect to controller
- Controller specifies EVE image in API
  - EVE downloads, verifies the SHA checksum, copies to partition, reboots
  - Grub boot loader uses priority encoded in GPT partition
    - on failure, timeout, or reset it switches back to previous partition
- EVE runs for 10 minutes to verify
  - connectivity to controller, remote attestation completes, no EVE failures
  - Then commit to the new EVE image
- See https://github.com/lf-edge/eve/blob/master/docs/BASEIMAGE-UPDATE.md

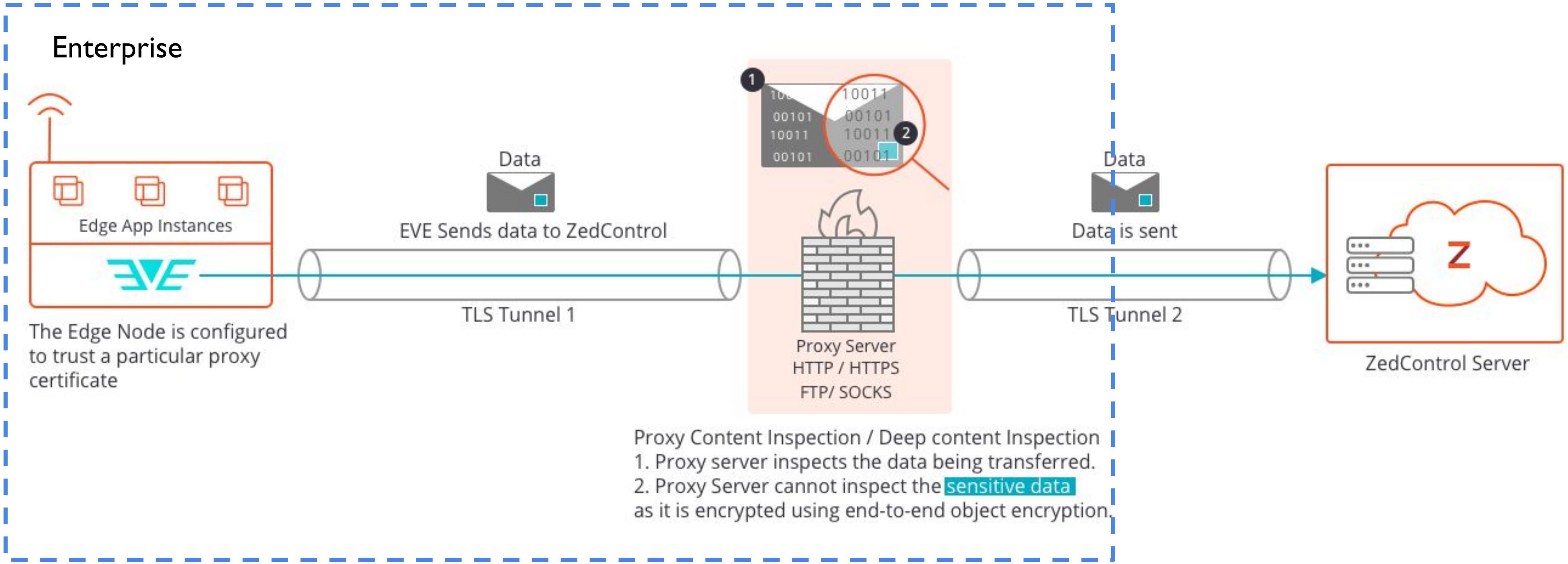
Ongoing EVE self-monitoring - watchdogs etc

- Hardware watchdog timer catches hardware that is stuck
  - During initial boot of EVE, or during ongoing operation
- Software watchdog daemon verifies that EVE services run and are responsive
- Watchdog(s) firing result in saving information and rebooting
- Should connectivity to the controller be lost for (default) one week
  - Reboot EVE
  - Needed to handle misbehaving network adapters and drivers
- Some monitoring of S.M.A.R.T. disk/SSD counters

EVE API

- Assumptions
  - Asymmetric connectivity - need to phone home to controller
  - Unpredictable connectivity - eventual consistency, compressible metrics
  - Support both end-to-end security for OT safety, and enterprise IT security concerns like content inspection
- Different API endpoints to enable scalability
  - config, info/status, metrics, logs, flow logs, attestation
- Using TLS 1.2/1.3 plus end-to-end object signing
- User secrets additionally protected by end-to-end object encryption
  - To avoid leaking e.g., datastore credentials and cloud-init secrets

API Security - Three Layers



Proxy Content Inspection / Deep content Inspection
1. Proxy server inspects the data being transferred.
2. Proxy Server cannot inspect the sensitive data as it is encrypted using end-to-end object encryption.

1. TLS to trusted parties (controller and/or proxy)
2. End-to-end signature over payload (proxy can not modify)
3. Sensitive data encrypted end-to-end (also at rest)

EVE API Endpoints

- Different services:
  - POST api/v1/edgedevice/register for device onboarding
  - GET api/v1/edgedevice/ping for connectivity test
  - GET api/v1/edgedevice/config for complete device + instance config
  - POST api/v1/edgedevice/info for triggered device/instance status
  - POST api/v1/edgedevice/metrics for periodic device/instance metrics
  - POST api/v1/edgedevice/logs for logs from microservices on device
  - POST api/v1/edgedevice/flowlog for ECO network flows logs
- All messages encoded using protobuf
- See https://github.com/lf-edge/eve/tree/master/api

App Runtime - domainmgr, containerd, and hypervisors

- Provide an abstraction over different container and VM runtimes
- EVE uses KVM hypervisor by default
  - Xen and ACRN also work
  - Open to other hypervisors; type 1 have smaller attack surface
- OCI containers can be run directly
  - Without a hypervisor
- EVE abstracts resource assignment (CPU, memory) and usage metrics
- EVE abstracts I/O assignment (networking, PCI, serial, etc)
  - Hypervisor tools chain used set up virtual network connectivity, and any direct device assignment/passthrough
- See https://github.com/lf-edge/eve/blob/master/docs/TASKS.md
-

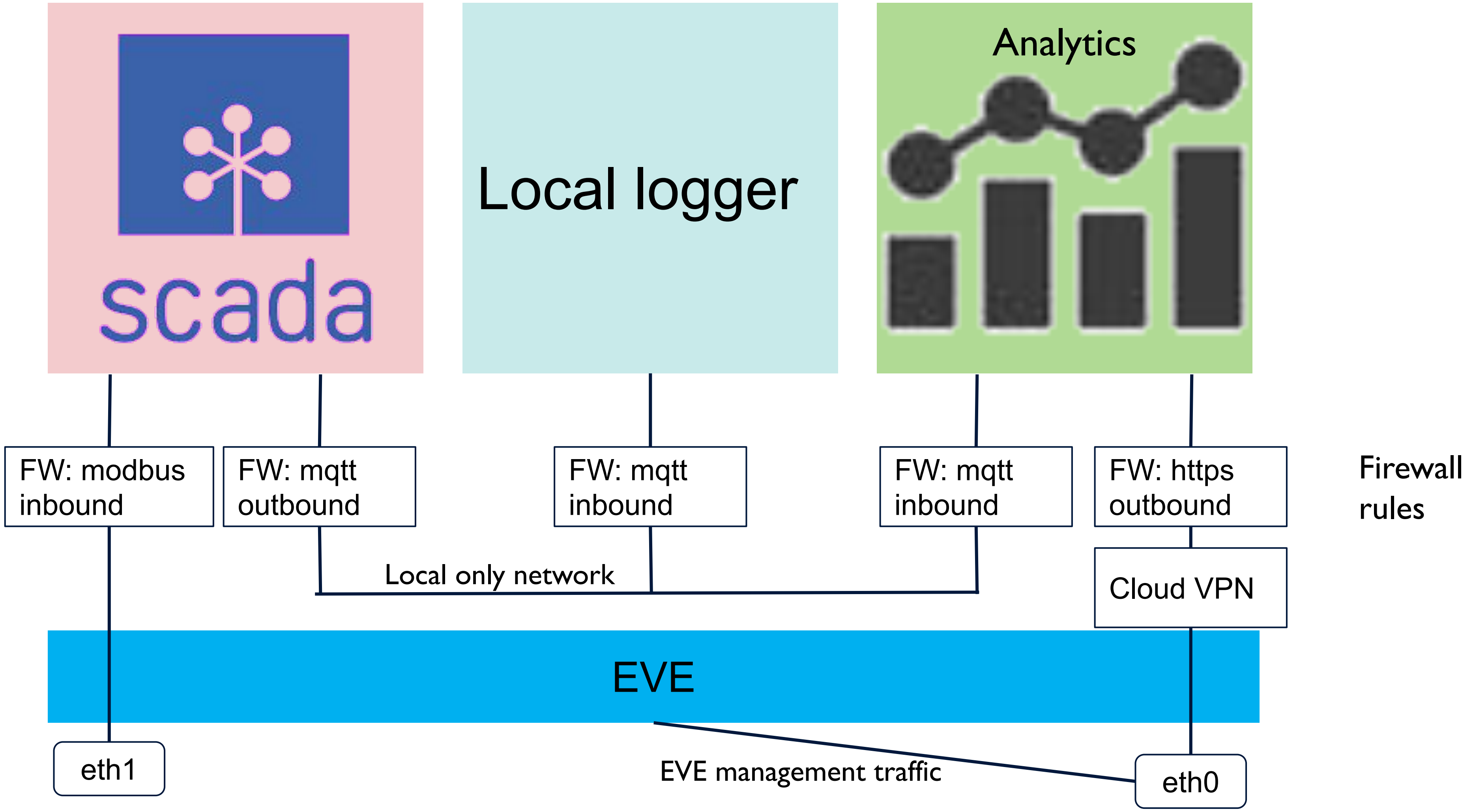
Storage and Volumes - volumemgr, downloader, verifier

- Four layers:
  - datastores - where to get content (could be your http server, docker hub, S3, Azure, etc)
  - content trees - generalized OCI structure for layered content
  - volumes - read-only or read-write for the applications
  - deployment of applications will mount the volumes needed
- Controller provides meta-data (including sha checksums)
- EVE uses make-before-break when a volume needs to be refreshed with new content (“purge” operation)
- Structuring your applications as OCI layers means smaller downloads
- See https://github.com/lf-edge/eve/blob/master/pkg/pillar/docs/volumemgr.md

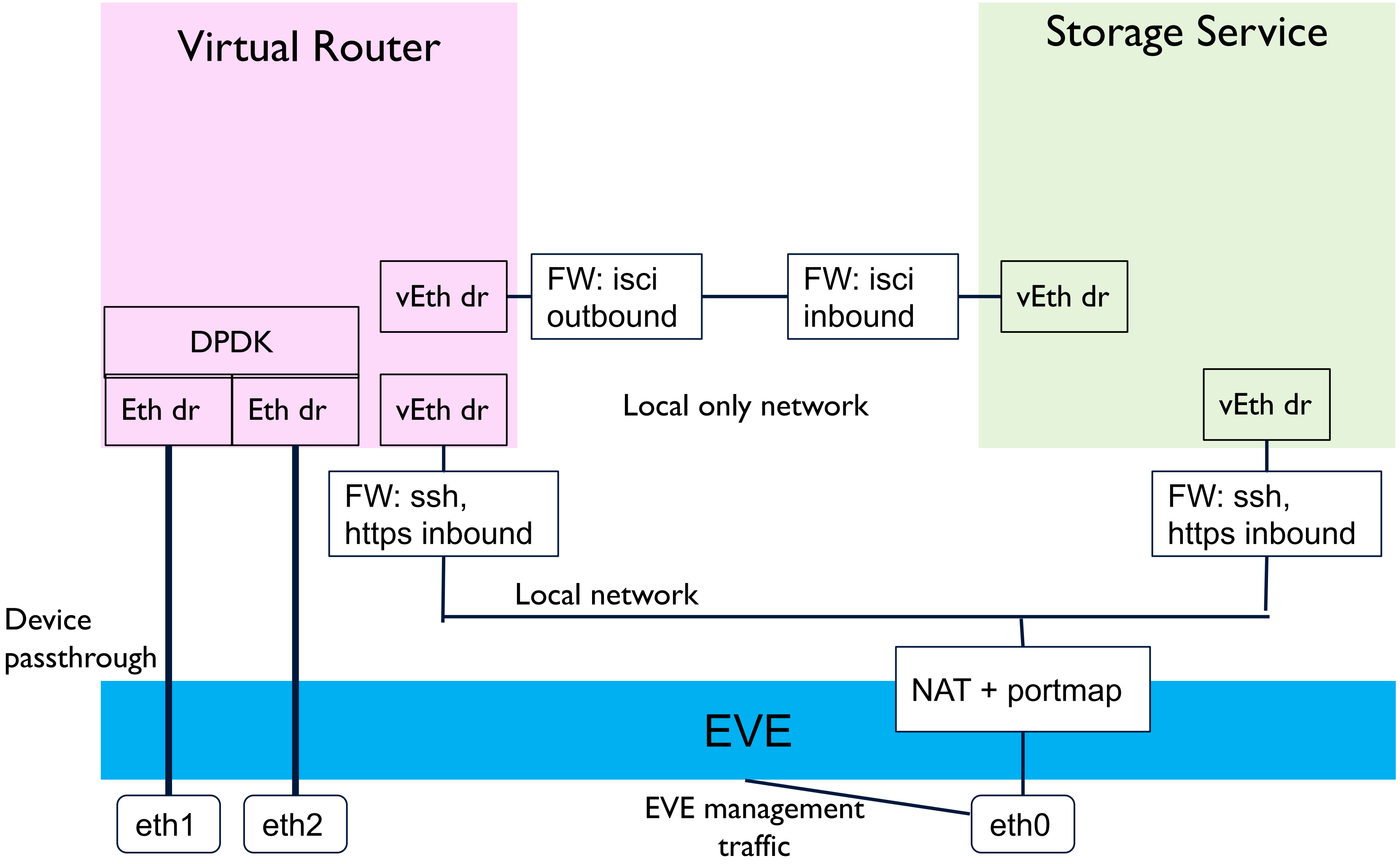
App Connectivity - zedrouter

- Different network connectivity options
  - Switch connectivity for transparent L2 connectivity (IP and non-IP)
  - Entirely local to host (between app instances), or local + NAT externally
- Network connectivity needs firewall rules - default deny
- Different I/O connectivity
  - Assignment of a complete I/O device (NIC, audio, USB controller, GPU)
  - Serial ports (RS 232, RS 485)
- Remote console to application from your web browser
- Can deploy e.g., SD-WAN as applications on EVE
  - serving other applications and network ports
- See https://github.com/lf-edge/eve/blob/master/docs/NETWORK-MODELS.md

App Connectivity Example - securely connect legacy



App Connectivity Example - high-performance networking



Recent changes

- Metadata internal endpoint (accessible on 169.254.169.254/eve/v1/kubeconfig) to send data from the app instance to EVC.
- Radio silence mode to disable all interfaces in danger areas
- Support for empty volumes to create them without downloading from datastore
- Support for Intel VGA passthrough into Windows VM
- Work on expanding the list of supported ECO containers
- Generation of security keys during installation
- Reducing of network traffic

Open issues

- EVE-OS installation
  - IPXE installation from GitHub/controller
  - Scale installation of devices: network installation, installation data (Inventory) collection
  - Expand supported device (edge-nodes and connected devices) database
  - Handle hardware/model variants better (with/without LTE, more disk or memory, etc)

Open issues

- EVE-OS connectivity
  - select and re-implement VPN connection type?
  - geolocation using GPS
  - support L2 network segmentation - VLANs
  - reduce network traffic between controller and edge-node
  - Link aggregation (LAG, bonding)?

Open issues

- EVE-OS configuration
  - rework config partition: detach options for installer, rework partitions layout
  - workout ways to change config for fleet of devices during installation
  - make config generator tools

Open issues

- EVE-OS observability
  - better ways to access device to obtain debug info:
    - allow only predefined subset of commands
    - define commands to query device enumeration/capabilities/logs to identify issues
    - document best practices to get needed information from device
  - work on filtering and aggregation of logs from device

Open issues

- EVE-OS objects
  - support download resume
  - pending changes and operations indication
  - support unikernels
  - support iso boot
  - support ipxe boot
  - support vTPM

Open issues

- EVE-OS testing
  - expand tests with (v)TPM edge-nodes
  - add arm64 targets