Project EVE Architecture and Security

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

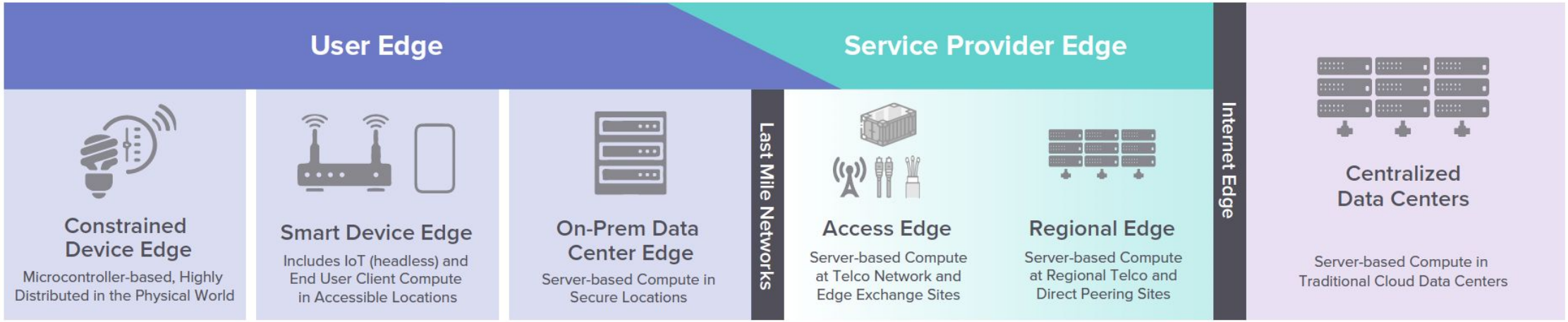
Erik Nordmark, CTO, ZEDEDA

Sept 2022

The Edge, EVE, and LF-Edge

THE LINUX FOUNDATION

Edge means different things to different people



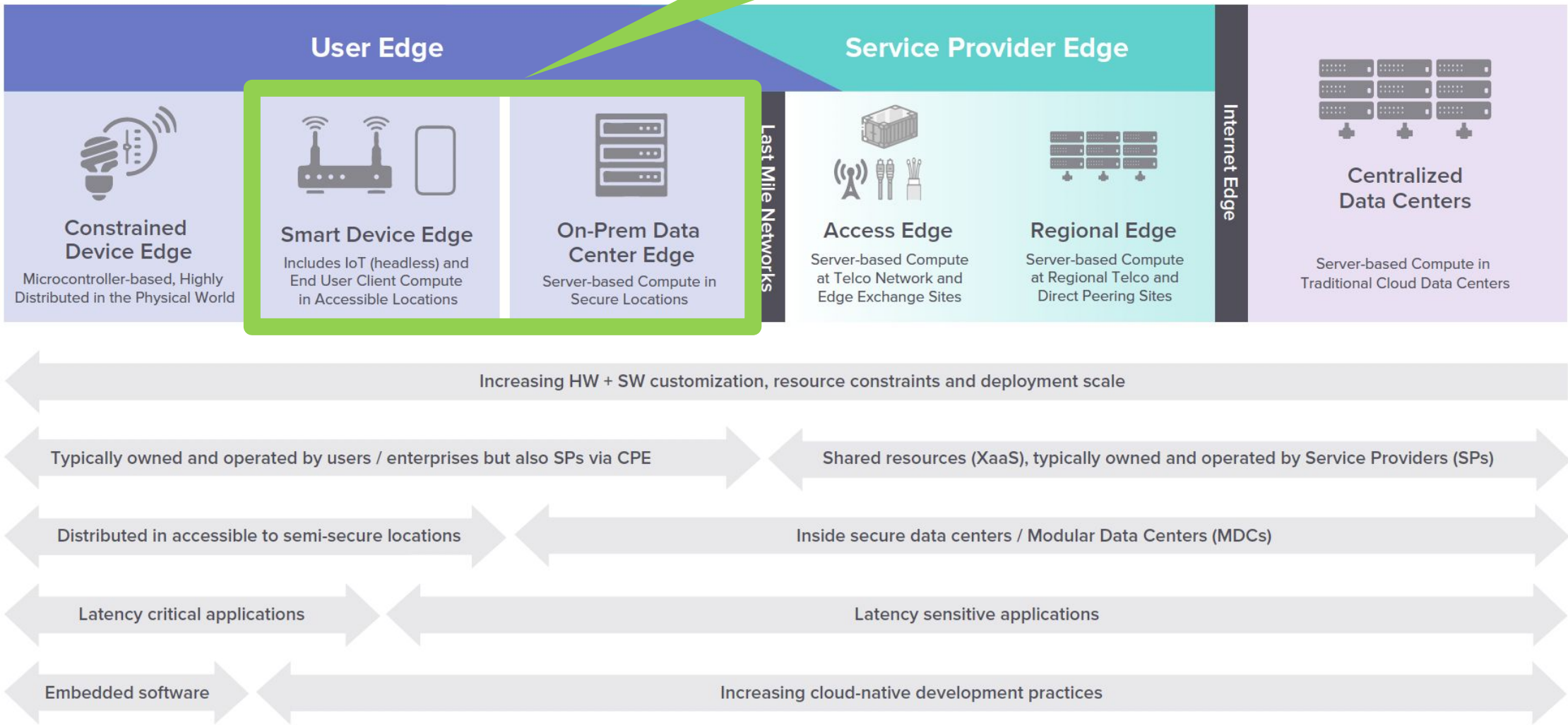
THE LINUX FOUNDATION

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

LF EDGE

Fit in Edge Continuu...

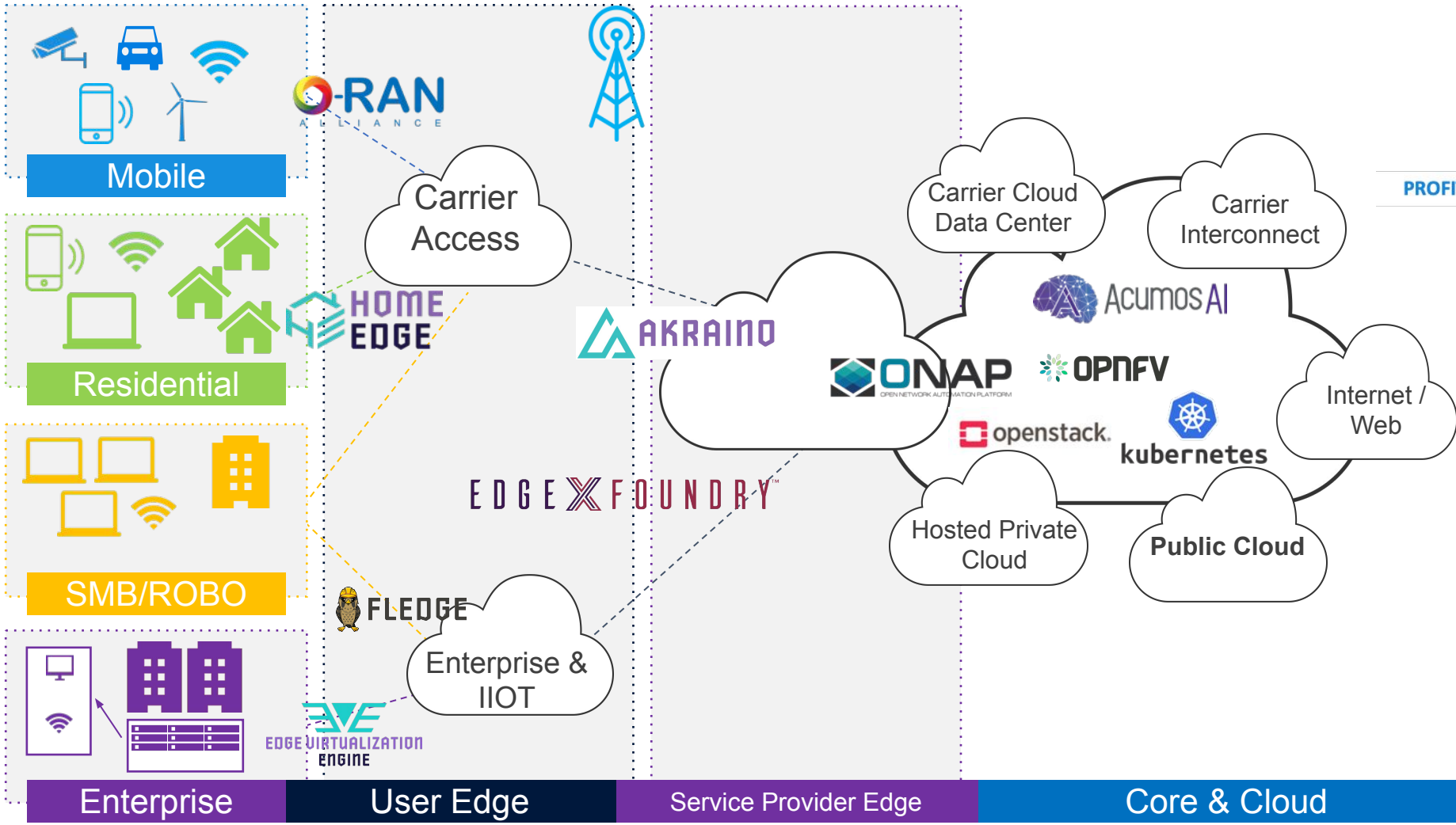
Project EVE is focused on IoT workloads at the Smart Device Edge

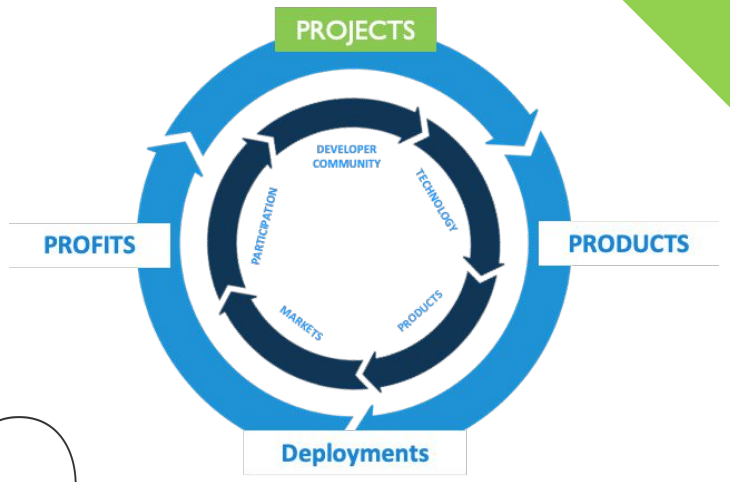


LF Edge - the end to end context

Deployment ready Open Source - use cases

OSS+SDO





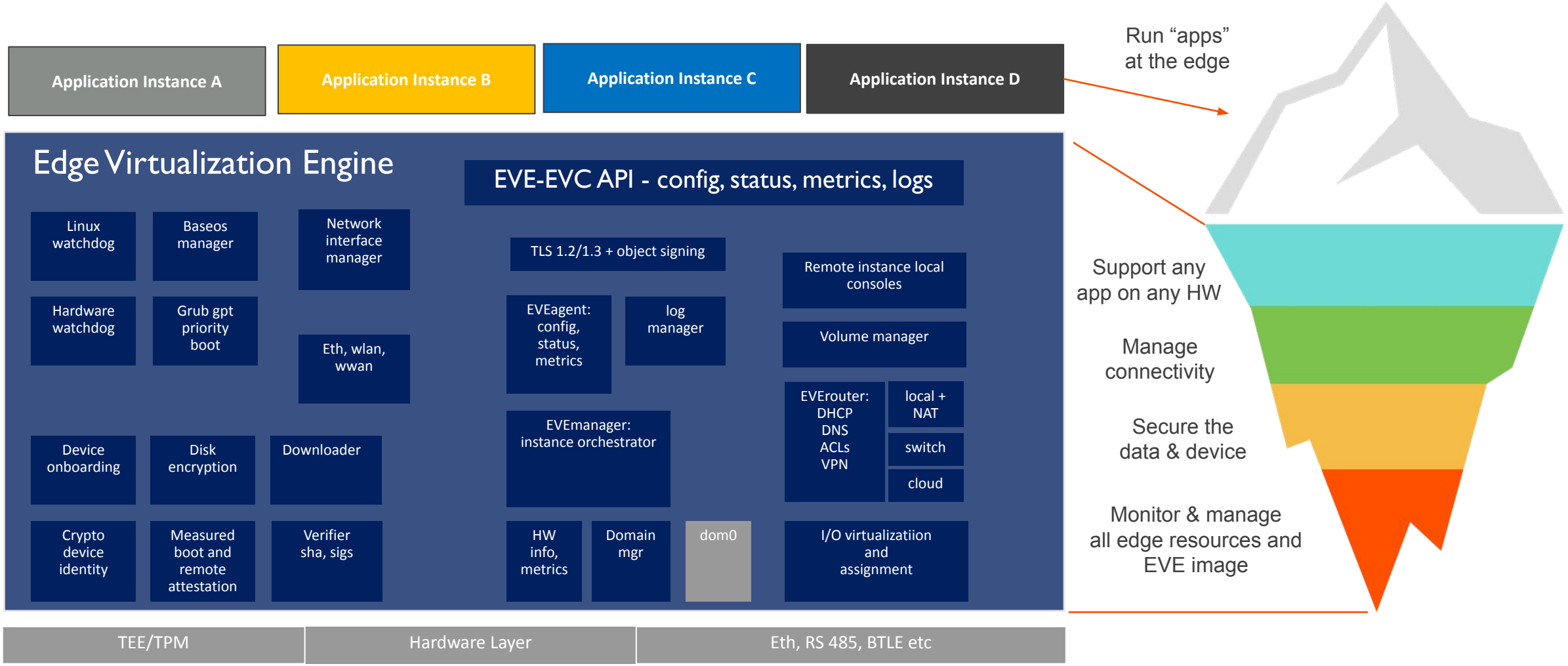
ETSI ISG MEC
World Class Standards

AECC
AUTOMOTIVE EDGE COMPUTING CONSORTIUM

industrial internet CONSORTIUM

X-Project Collaboration

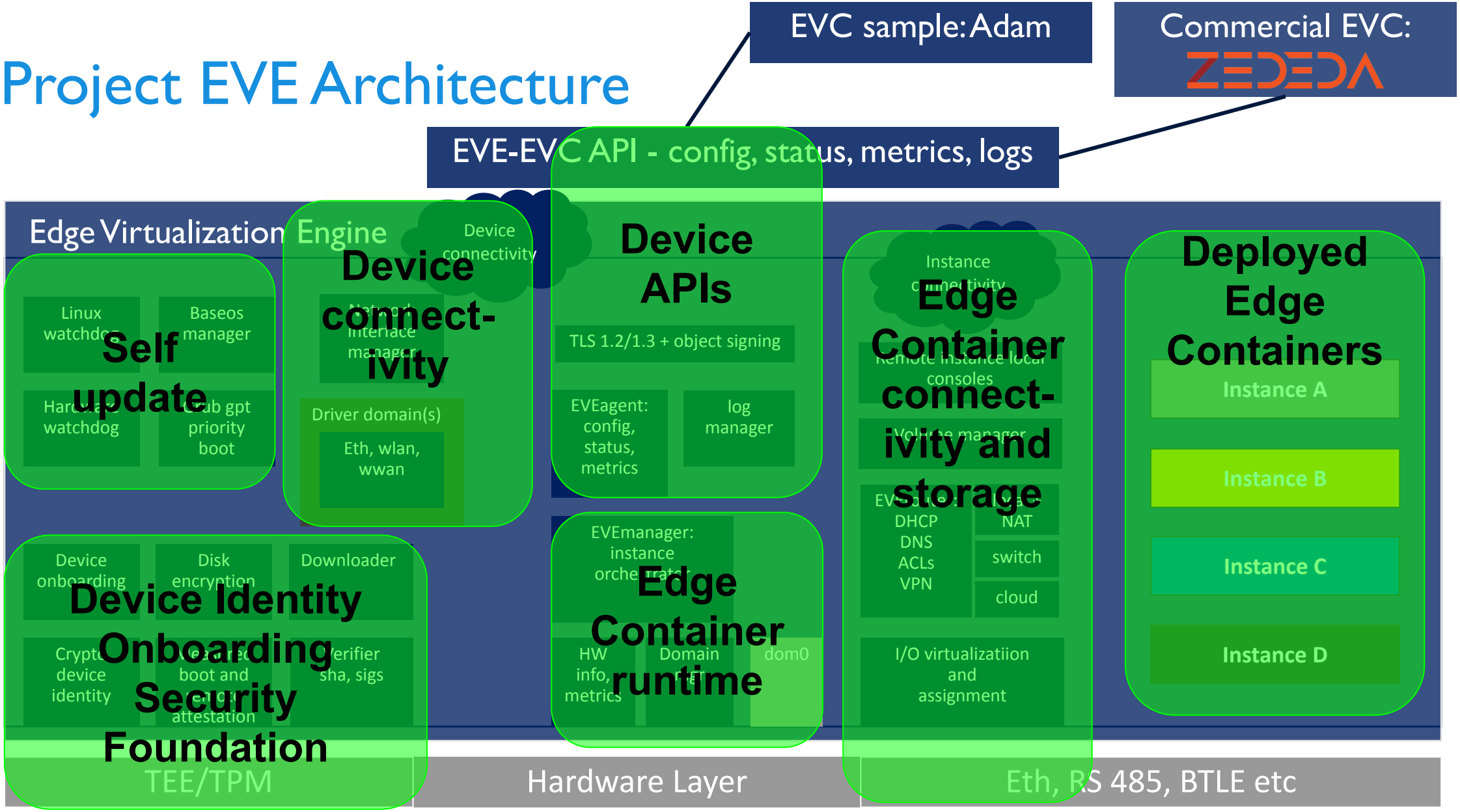
App deployment is but the tip of the iceberg



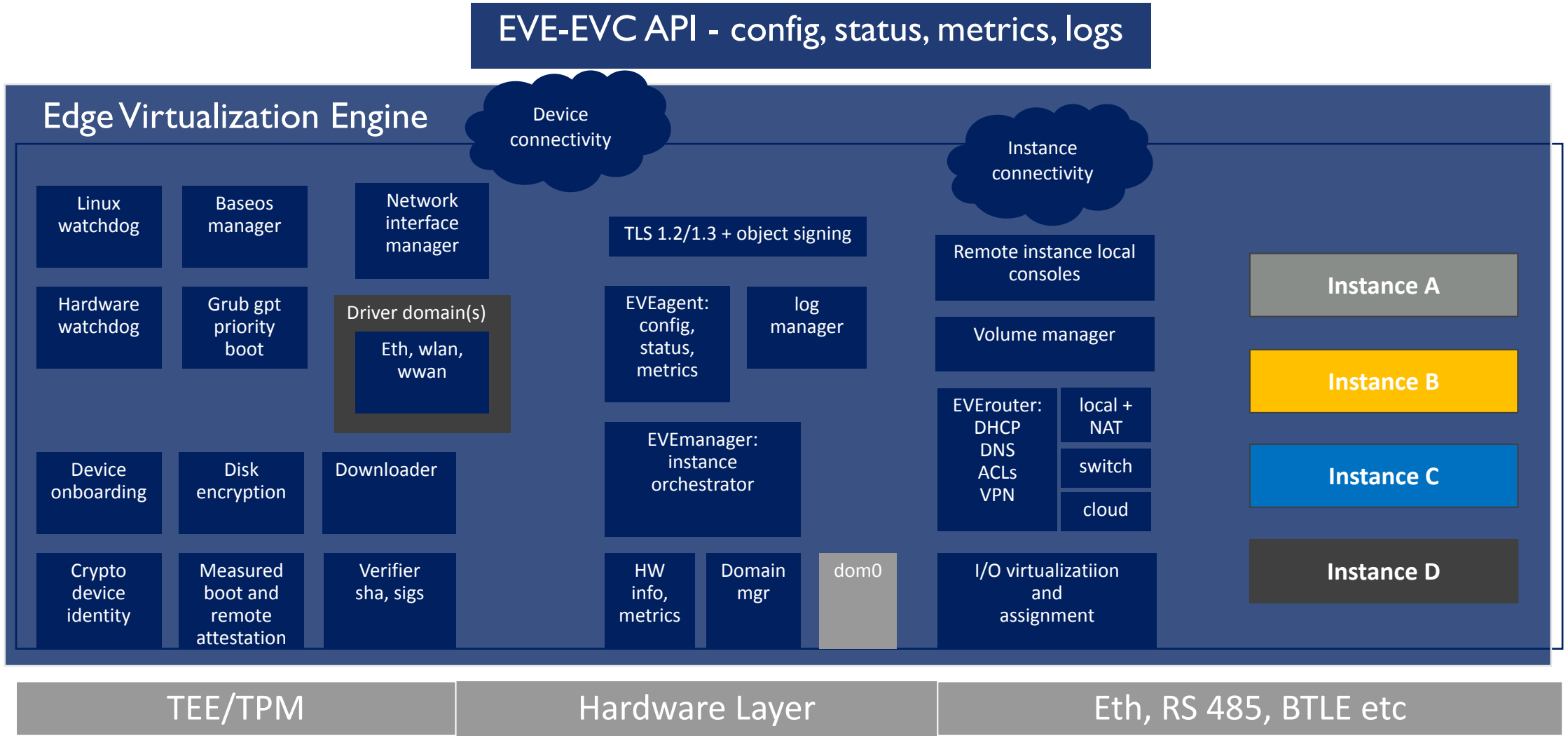
EVE Architecture and Security

THE LINUX FOUNDATION

Project EVE Architecture



Project EVE Architecture



Device Identity

- A device is identified by an X.509 certificate
  - Generated by the TPM on first boot
  - Currently self-signed and 20 year lifetime
  - Only used by for the management traffic to the controller
- Controller can detect misbehaving devices (remote attestation, anomaly detection) and quarantine them (no applications run etc)
  - No need for short certificate lifetimes nor CRLs for the device certs
- Device is imprinted with the controller to trust (a root CA certificate)
- Note: controller certs is normal server -> intermediate -> root CA certificate hierarchy

Device Onboarding

- 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 Onboarding - Most Secure

- EVE image is (pre-) installed; could be in untrustworthy environment
- In secure environment:
  - EVE is booted (generates device key/cert in TPM)
  - Device cert is extracted from the device and saved by user
  - Device is powered off
- Device is shipped to installation site
- Device cert is used to onboard the device in the controller
- Typically combined with TPM measurement of firmware+software in secure environment

Device Onboarding - Easier to use

- EVE image is installed
  - Install image includes an onboarding token (X.509 cert + private key)
  - Could use unique token per device, or shared for entire production line
  - Onboarding certificate(s) plus hardware serial numbers delivered electronically from factory to user
- Device is shipped to installation site
  - Powered on (EVE generates device cert using TPM) and self-registers
- Onboarding cert + serial number used to onboard device in controller
  - Attacker buys one device with onboarding token guesses serial numbers
  - Will detect duplicate and refuse second registration

Device Onboarding - Middle Ground

- EVE image is installed
  - Install image includes an onboarding token (X.509 cert + private key)
  - Could use unique token per device, or shared for entire production line
  - EVE generates random 128 bit soft serial and saves on USB stick
  - Onboarding certificate(s) plus soft serial numbers delivered electronically from factory to user
- Device is shipped to installation site
  - Powered on (EVE generates device cert using TPM) and self-registers
- Onboarding cert + soft serial used to onboard device in controller
  - Attacker needs to guess 128 bit number

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
  - Detects rogue firmware and/or EVE/OS
  - [In progress] also measuring hardware chassis intrusion log
  - Changed measurements => require remote attestation to unlock application disks
  - Same measurements => unlock and start applications (without controller connectivity)
  - See https://wiki.lfedge.org/display/EVE/Measured+Boot+and+Remote+Attestation
- See https://github.com/lf-edge/eve/blob/master/docs/BOOTING.md

Measured Boot and Remote Attestation

- Requirements
  - If no firmware/software change, then applications **must** come up after reboot without talking to controller
  - If a change is detected the application **must** disks/volumes remain unreadable until remote attestation to controller has completed
  - Integratable with secure boot and various BIOS update schemes
  - Avoid any insecure maintenance state of device
- Approach
  - Measured boot/remote attestation with TPM sealed keys as baseline
  - Secure boot is optional
  - Device needs to contact controller after a BIOS or EVE update (already needed to validate that EVE update worked)

Measured Boot - EVE Update Workflow

- Unchanged from the user perspective if no issues
- If EVE with unknown sha's is installed (e.g., opensource developer build)
  - UI will flag as "**Unknown Update Detected"**
    Includes identifying the component with the mismatched hash
  - Applications will not be started on device (their disks /volumes can not be decrypted)
  - If the hashes are later added as acceptable to controller, then the applications will start. Or can update EVE to a known version with known hashes
- Above UUD flagged if there is a compromise to the EVE image as well

Measured Boot - BIOS Update Workflow

- Can handle any form of BIOS update to support different hardware
  - E.g., physical access with USB stick and keyboard/screen
  - Or service running in EVE (as application) to do this via a BMC
- The BIOS version+hash needs to be uploaded to controller as an acceptable one
- If EVE with unknown BIOS sha is installed:
  - UI will flag as "**Unknown Update Detected**"
  - Applications will not be started on device (their disks/volumes can not be decrypted)
  - If the hashes are later added as acceptable in controller, then the applications will start.
- Above UUD applies if there is a compromise to the BIOS image as well

Measured Boot - Key Unlock Implementation details

- The application disks/volumes are encrypted using fscrypt or ZFS
  - Filesystem has key(s) to encrypt the files, plus a key encryption key
- That KeK is sealed under the TPM thus can be retrieved when the PCR values are unchanged (after a power cycle or reboot)
- That KeK is also encrypted under the TPM private key and sent to controller as a “backup”
  - Thus only the device with this particular TPM can decrypt it
- If the PCR values have changed, then controller will check the PCR values and the attestation chain it receives from the device
  - If that corresponds to a known version/hash of EVE and BIOS, then controller will send the “backup“ encrypted KeK to the device
  - Device will then seal that received KeK under the new PCR values

Key Takeaways

- Provide secure and scalable deployment/orchestration of devices, applications, volumes
- Make few environmental assumptions (no physical security, intermittent network connectivity)
- State of the art security foundation
- Scale from raspberry-pi size to edge servers with multiple GPUs, multiple drives, SR-IOV NICs
- Application developer can focus on their business logic
  - Deploying at distributed edge similar to deploying in cloud/DC
  - Application might need to handle intermittent connectivity