Project EVE: Enterprise/On-prem/Deep Edge Virtualization

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

Erik Nordmark, Chief Architect, ZEDEDA
Enterprise/On-prem/Deep Edge challenges

- Very distributed deployment - location is key
  - Minimize device installation and onboarding effort; maximize security
  - Never visit for software issues; have no separate management connectivity
  - Limited physical security; can’t assume firewalls
- Long lifecycle - patch applications/containers and runtime for 7+ years
- Rich downstream connectivity
  - Legacy serial ports, Industrial Ethernet, various radio technologies
- Diverse upstream connectivity - might need redundancy
  - Ethernet, LTE, etc
  - Might not control network; NATs, proxies etc deployed by someone else
- Yet want same/similar applications/containers and devops as in cloud
  - Apps shouldn’t need to worry about above differences
The need for edge virtualization: IIoT 1.0 → IIoT 2.0

**IIoT 1.0:**
- Vertical data silos & platform lock-in
- Data/edge sovereignty & control issues
- Hardware-defined & unmanaged edge

**IIoT 2.0:**
- Open IoT data architecture, no lock-in
- Data & edge belong to the enterprise
- Software-defined & ubiquitous edge
The Enterprise Cyber-Physical Edge Stack

Customer Business Outcomes
- Reduce outages
- Improve predictability
- Increase efficiencies

Cloud/DC

Edge Software
- Azure
- IoT Edge
- Greengrass

Data Services Layer: Abstract & Distribute IoT Data

Edge Hardware
- EVE: Edge Virtualization Engine
  Infra Services Layer: Virtualize & Abstract Edge

Machines & Assets

Sensors, Equipment, PLCs...

Open source edge runtime for ubiquity
Monetize visibility, control, security, apps, and plugins (EV-Central & EV-Catalog)
The virtualized, software-defined & composable edge

Cyber-Physical Edge

Integrated Edge Boards

“Composable” Edge Gateways

Device Protocol

Edge App

Network Service

Edge Virtualization, Abstraction, Trust, Visibility & Control

Hardware

Hardware

Hardware

Dell

Huawei

Hewlett Packard Enterprise

Cloud Orchestration

Useful Data

All-IP

Microsoft

Data Insights

aws

Fleet Analytics

Data Warehouse

Raw & High Bandwidth Data

Legacy & Analog Interfaces

THE LINUX FOUNDATION
Key Requirements

› Bare metal device onboarding and management at scale
› Application/runtime deployment, update, and management at scale
› On devices in remote locations
› Don't introduce security issues
› Run different applications/runtimes
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
Any/Freedom - Edge Virtualization

- EVE today supports ARM and Intel/AMD
  - Requires processor support for type 1 hypervisor
- 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, VMs with container runtimes (EdgeX Foundry, Fledge, Azure IoT Edge, AWS Greengrass Core), future Unikernels
  - Applications are not concerned with the variations in IP connectivity
Zero Trust - threats

- Poor usernames/passwords
- Poor usernames/passwords
- Poor usernames/passwords
Zero Trust - threats

› Poor usernames/passwords
› Physical access and plug in USB stick etc
› Steal disk/ssd
› Steal and clone device
› Network attacks exploiting software bugs in EVE and in Edge Containers
› Network DDoS of device
› Device becoming part of botnet attacking others
Zero Trust - People, Processes, and Technology

› People
  › Remove need for usernames and passwords on edge; use cryptographic device identity and APIs
  › Role-based access control and multi-tenancy in cloud controller

› Processes - handle 7+ year lifetime at edge
  › Secure, scalable distribution of updates for EVE and Edge Containers
  › Anomaly detection across edge fleet in controller

› Technologies for the edge
  › Use hardware root of trust (TPM/TEE); express phone home as root CA certificate(s)
  › Remotely control and block hardware connectors such as USB
  › Use trusted platform module w/ remote attestation to detect rootkits
  › Use signed images for EVE software and Edge Container Images
  › Use encrypted filesystems; keys sealed by TPM
  › Leverage hypervisors for strong isolation and defense in depth
  › Built-in firewall rules and VPN cloud connectivity for Edge Containers
  › EVE can be used to deploy additional security applications (IDS etc)
  › Cloud controller can interface with additional analytics backends
Edge Containers

- Edge Container Image (ECI)
  - A self-contained, binary representation of an Edge Container
  - Supports traditional VMs, OCI/docker Image format, Unikernels
- Edge Container Object (ECO)
  - A live copy of an Edge Container
  - ECI plus bindings to specific resources on a device
- See [https://github.com/lf-edge/eve/blob/master/docs/ECOS.md](https://github.com/lf-edge/eve/blob/master/docs/ECOS.md)
- EVE TSC is starting to formalize the definitions
Edge Container Connectivity

- Default is local network with NATed connectivity
- Can provision a switch network - an L2 network e.g., on eth1
- Can provision USB controller or COM port if instance has its own drivers (Industrial Ethernet, TSN, BTLE, Modbus over serial)
- Can provision a cloud network - connect to AWS, Azure VPN
- Can provision a mesh network - connect device to device
  - Handles multihoming, mobility, NAT traversal, authentication, encryption
  - No changes to container; uses DHCP to get IP addresses as normal
- Can provision a local network with no external port; local-only
- If vnc is enabled in manifest can use Guacamole for remote console
EVE Device API

› Connection from device (through NAT) using TLS1.2 (soon 1.3)
› 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
Register API

- Used in some onboarding scenarios
  - A device starts out with a (single use) onboarding token
  - Device creates device certificate using its TPM on first boot
  - Register API binds the device certificate to enterprise/user via token
- POST to /api/v1/edgeDevice/register
- Simple message:
  ```
  message ZRegisterMsg {
    bytes pemCert = 2;
    string serial = 3;
    string softSerial = 4;
  }
  ```
Config API
https://github.com/lf-edge/eve/blob/master/api/proto/config/devconfig.proto

- All of the configuration from the cloud to the device
  - Device configuration
  - Edge Container configuration
  - (Local) network instance configuration for ECOs
  - Datastores (for images and other objects)
- GET to /api/v1/edgeDevice/config
- Typically used on boot and periodically to check for updates
Config: Device parts

- PhysicalIO - describe the physical (networking, USB, etc) ports on device
- SystemAdapter - describe the device’ desired IP+DNS+proxy configuration
- NetworkConfig - referenced by SystemAdapter for static IP and proxy
- DeviceListDetails - specify parameters for LISP overlay network
- BaseOsConfig - specify update of EVE itself
- DevOpsCmd - specify device reboot etc
- ConfigItem - open-ended way to specify e.g., timers, debug settings, etc
Config: Edge Container Object parts

- DataStoreConfig - from where to fetch images (S3, https, sftp)
- NetworkInstanceConfig - virtual networks for the ECO
- Each ECO includes:
  - CPU, memory requirements
  - Virtual disks, including SHA and signatures of images
  - Physical Adapter assignment (e.g., USB and serial ports)
  - Virtual Adapter assignment (virtual Ethernets)
    - Attachment to the network instances
    - Includes Access Control Lists with optional rate limits
- UserData for cloud-init
- Enabling of remote console
Info API
https://github.com/lf-edge/eve/blob/master/api/proto/info/info.proto

› Reports sent on state change for device or for ECO:
› Device:
  › Hardware and BIOS info (serial numbers, versions, TPM info)
  › Status of EVE version and any version update in progress
  › Resource usage (CPU, memory, disk, physical adapters)
  › Network status (IP, DNS, tried/failed information)
› Edge Container Object:
  › Up/down, boot time
  › Image download status and progress
  › Virtual network status
Metrics API


› Sent periodically by device; cumulative counters
  › Missed messages merely result in reduced time resolution
› Device metrics:
  › CPU, memory, disk, network usage
› Edge Container Object metrics:
  › CPU, memory, disk, network usage
  › ACL violations; network rate limits exceeded
› Network instance metrics:
  › Including VPN and LISP counters
Log API


› Device logs for debugging
  › Amount of logging can be controlled with a configItem
› Logs are bundled in a LogBundle and sent when full or after timeout
› POST to /api/v1/edgeDevice/logs
Flowlog API
https://github.com/lf-edge/eve/blob/master/api/proto/flowlog/flowlog.proto

› For Edge Container network flows
  › Accepted
  › Dropped
  › Plus hostname to IP address mappings as seen by device
› Brand new; implementation underway
› POST to /api/v1/edgeDevice/flowlog
More info

https://www.lfedge.org/projects/eve/
https://github.com/lf-edge/eve
https://github.com/lf-edge/eve/blob/master/docs/ECOS.md
https://github.com/lf-edge/eve/tree/master/api
Brief demo