QNX® Containers

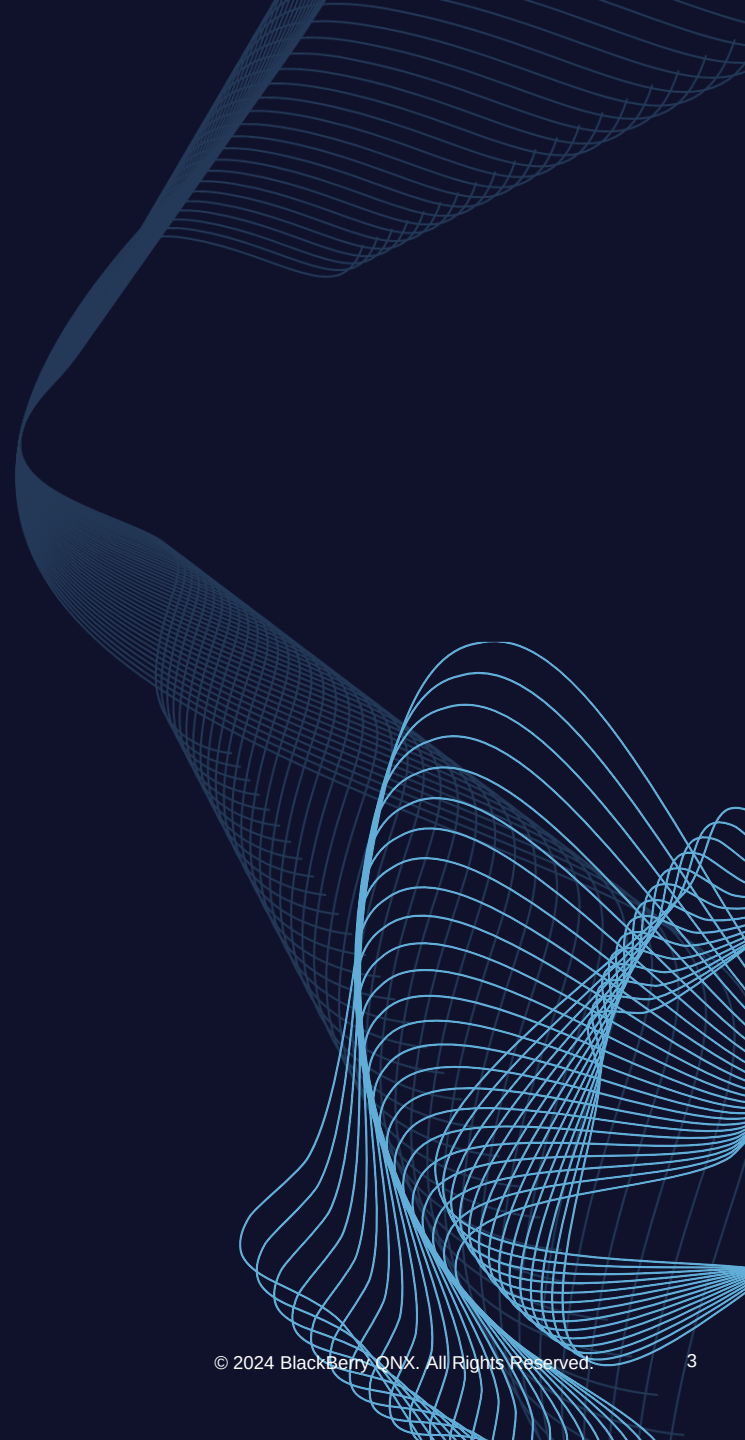
Virtualization for Reliable Systems

Patrick Perron
Senior Product Manager

Agenda

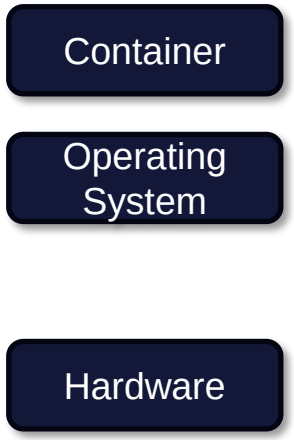
1. Virtualization Overview
2. Containers Overview
3. QNX Containers

Virtualization Overview



The Many Models of Virtualization

OS Virtualization



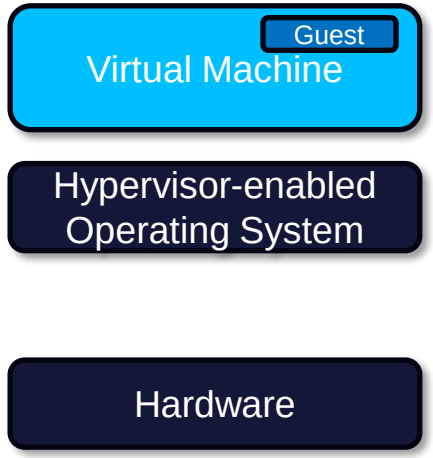
Operating systems must match:
e.g. QNX Container on QNX OS, Linux Container on Linux OS.

Container holds a set of programs that run in a restricted environment

Container Model

VS

HW Virtualization

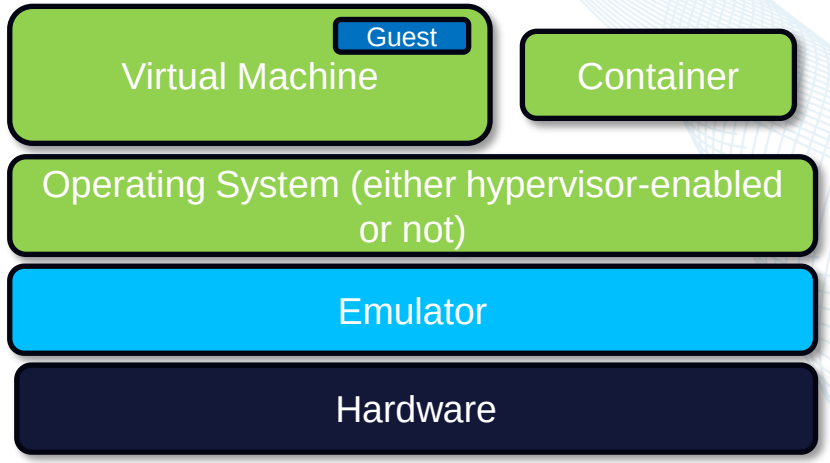


Hardware must match between virtual machine and operating system:
But Guest OS running in virtual machine can be a different OS than underlying Operating System, but must be same hardware.

Virtual Machine Model

VS

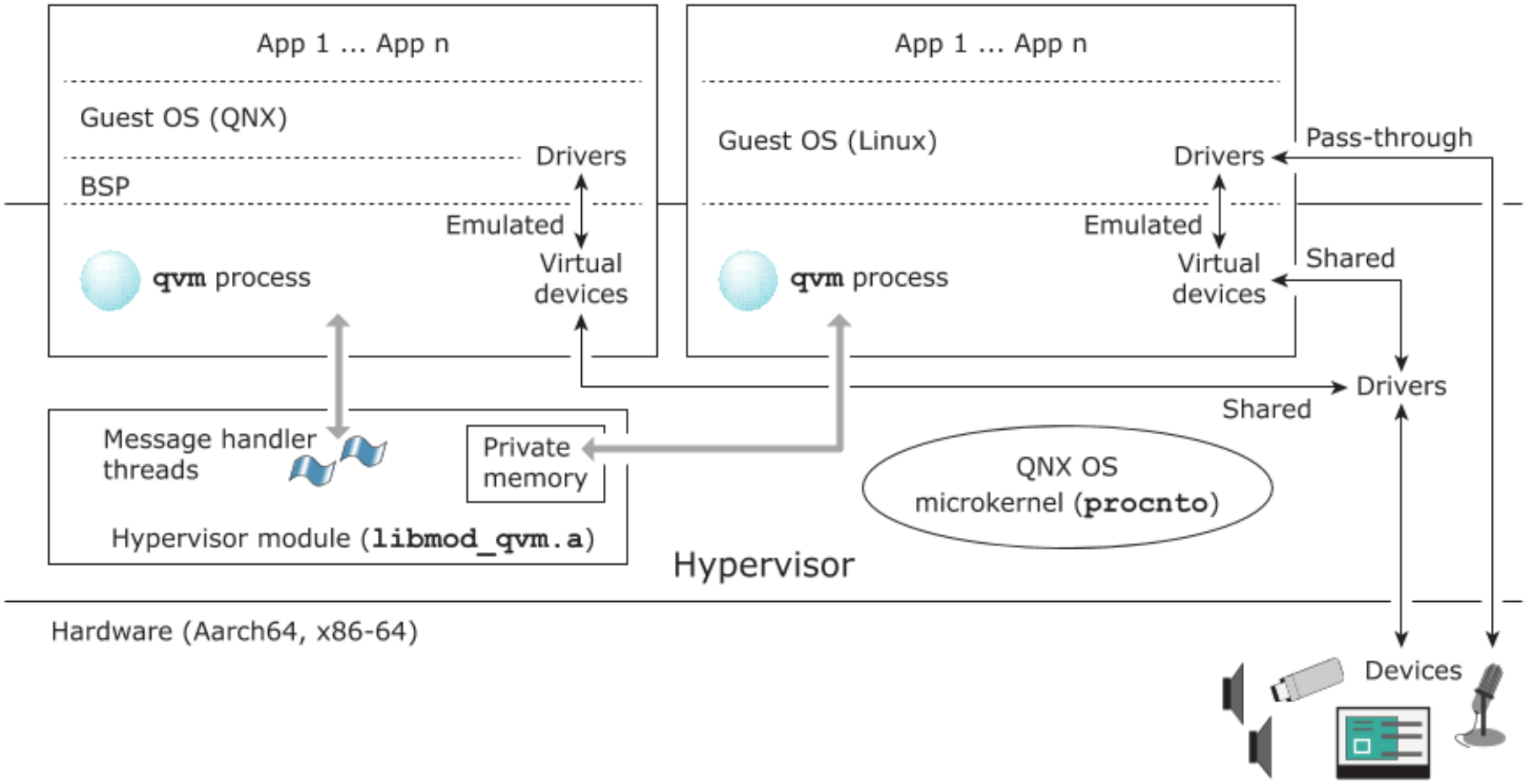
HW Emulation



Emulator can present different hardware to the operating system than real hardware.
A Virtual machine will use HW virtualization on top of emulator.
Container will use OS virtualization on top of emulator.

Emulation Model

QNX Hypervisor - Architecture



https://www.qnx.com/developers/docs/7.1/#com.qnx.doc.hypervisor.safety.user/topic/virt/arch.html

The Evolution of VIRTIO and QNX

2016

QNX engineering start shared graphics initiatives with Android.
Jaguar XJ demo @ CES2017

2019

OEMs start standardizing on native but custom Android

2020

Beginning of Google & QNX VirtIO engineering collaboration

2021

Google VirtIO initial implementation Cuttlefish & Android 11 = slow

Beta of automotive version of VirtIO “Trout”

2022

Official release of automotive version of VirtIO “Trout”

VirtIO components appear in production vehicles (e.g. VirtIO Video & VirtIO Sound)

2024

Next generation cockpits adopt full VirtIO in support of cloud, multi-SoC and global platform strategies

OEMs asking for standard Android not SoC custom Android

Google

STELLANTIS

VOLVO

QNX

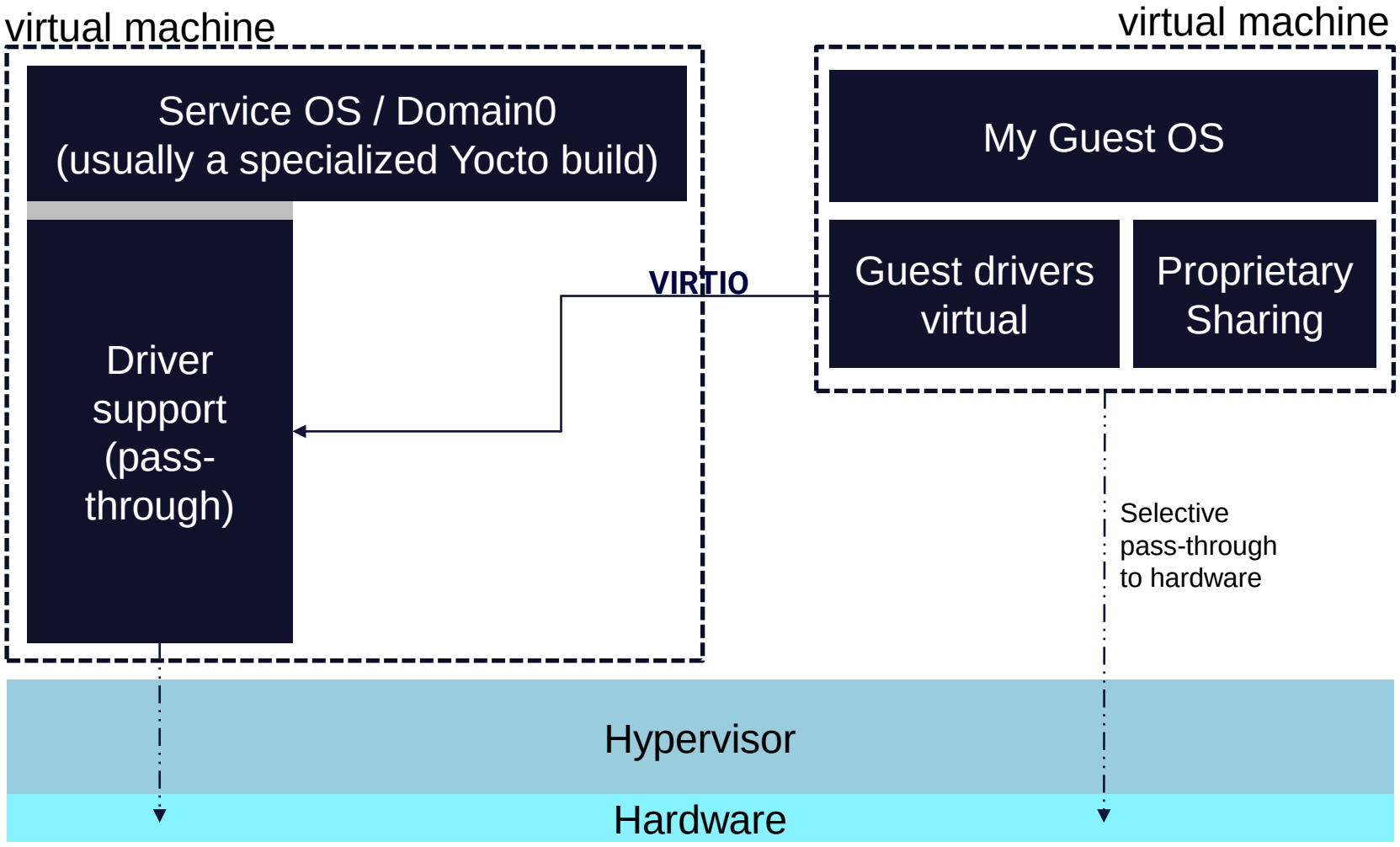
... and more.

Hypervisor Design is still the most important choice

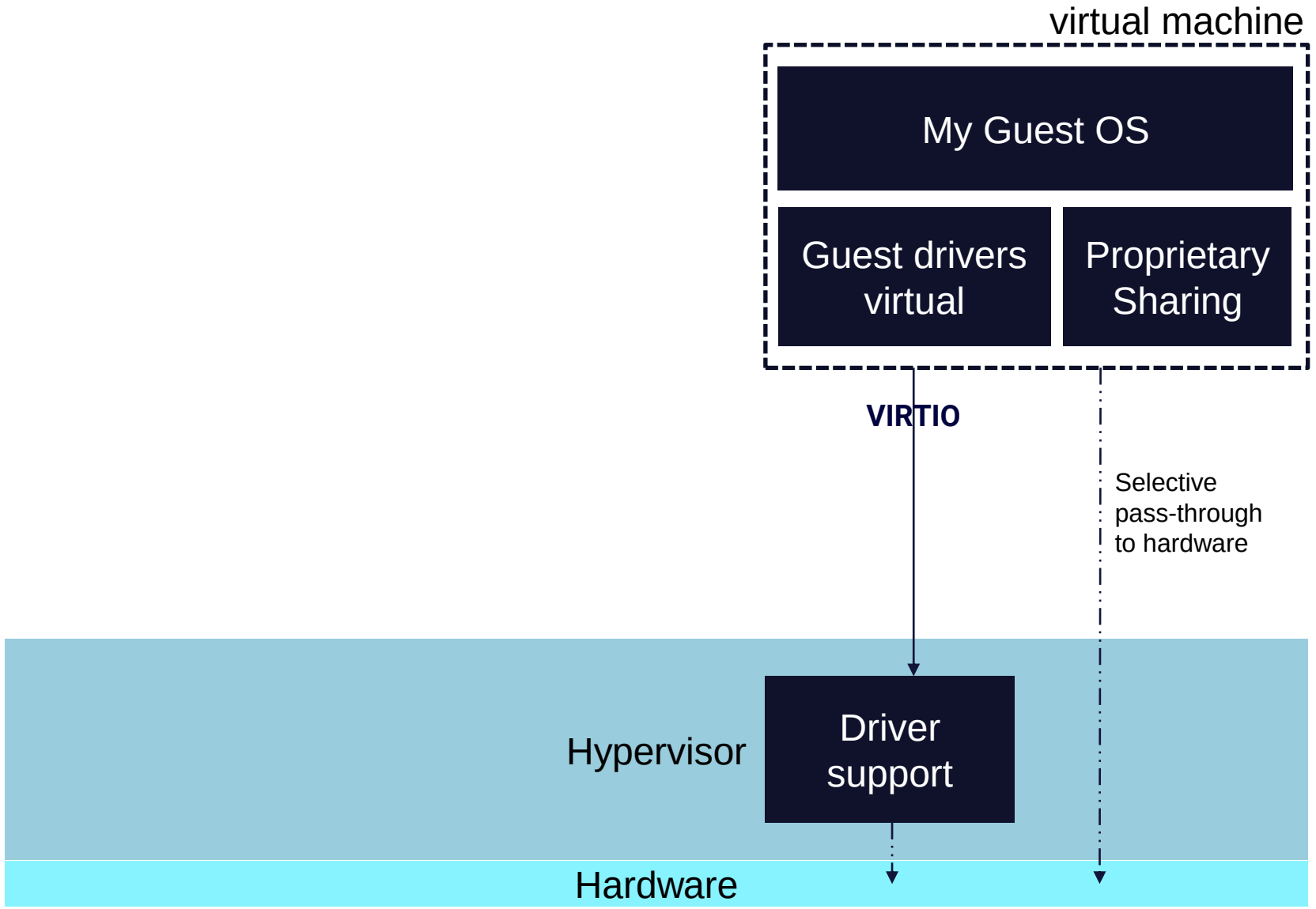
1. Service OS in its own Virtual Machine

2. Service OS in the Hypervisor Host

Service OS exists in its own Virtual Machine



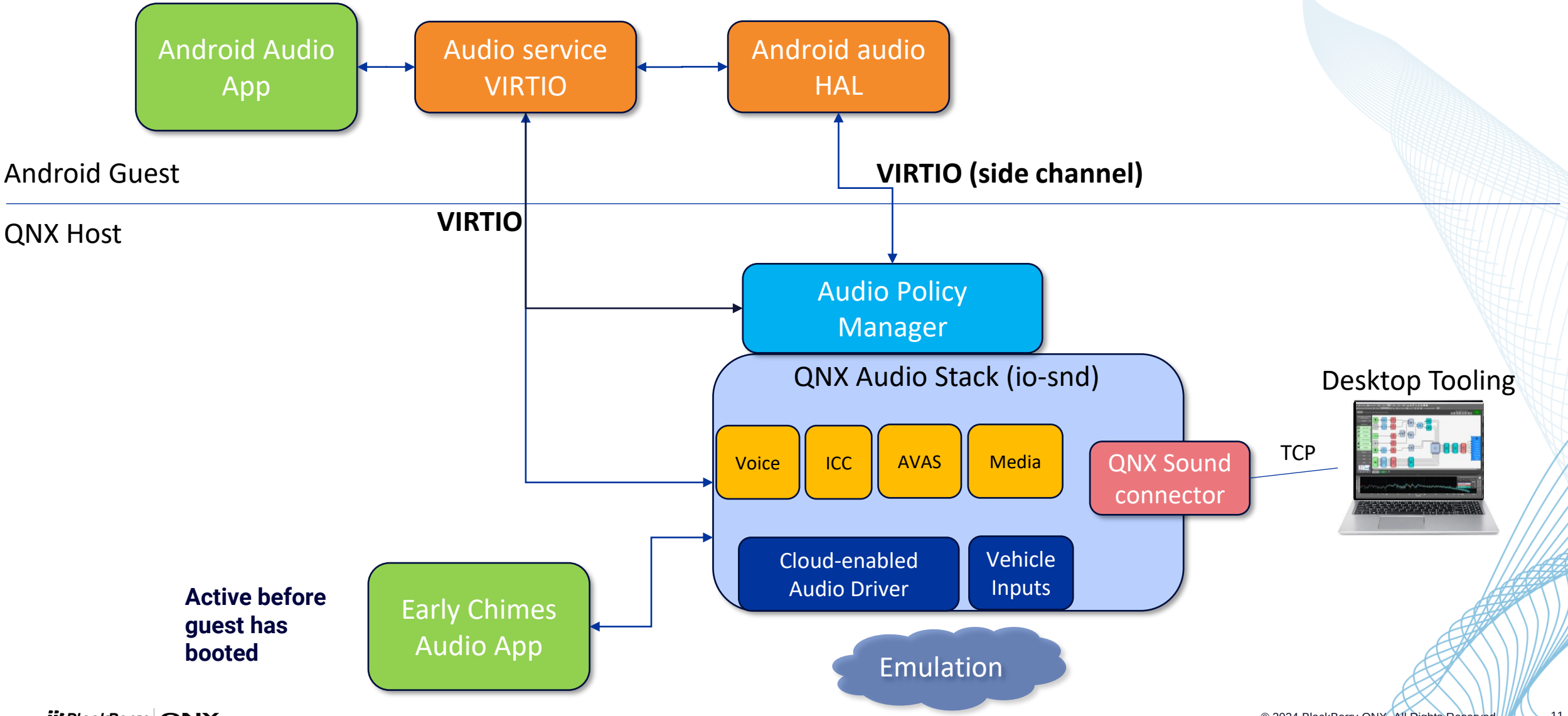
Service OS exists in the Hypervisor Host



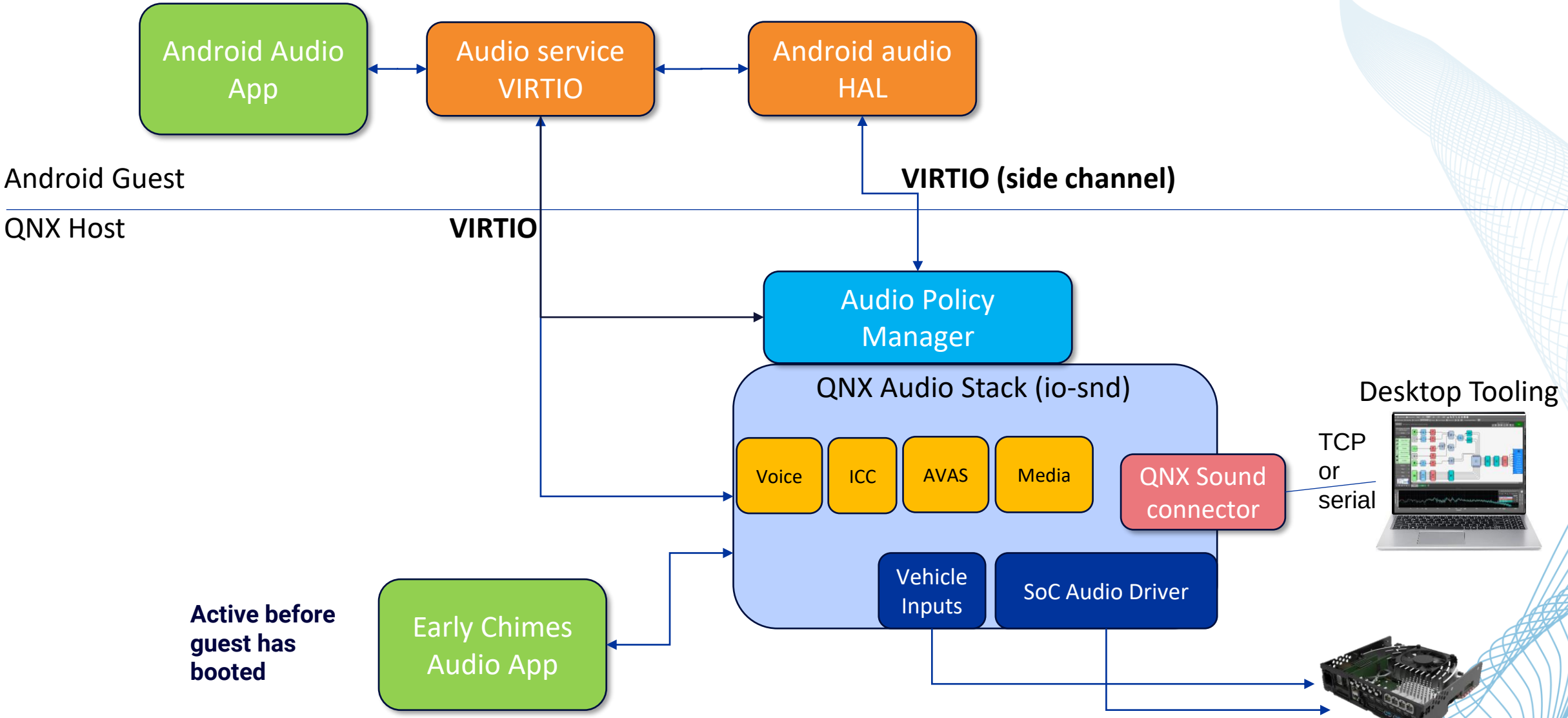
Sharing Example: Audio Requirements (Acoustics)

1. Want audio available (chime) as soon as target boots (< 1second)
2. Want Audio to be performant when combining many sources (music, alerts, etc.)
3. Ability to easily add in new features over the life-cycle of the sound architecture
4. Want to be able to move software between SoC vendors
5. Cloud-first design and test

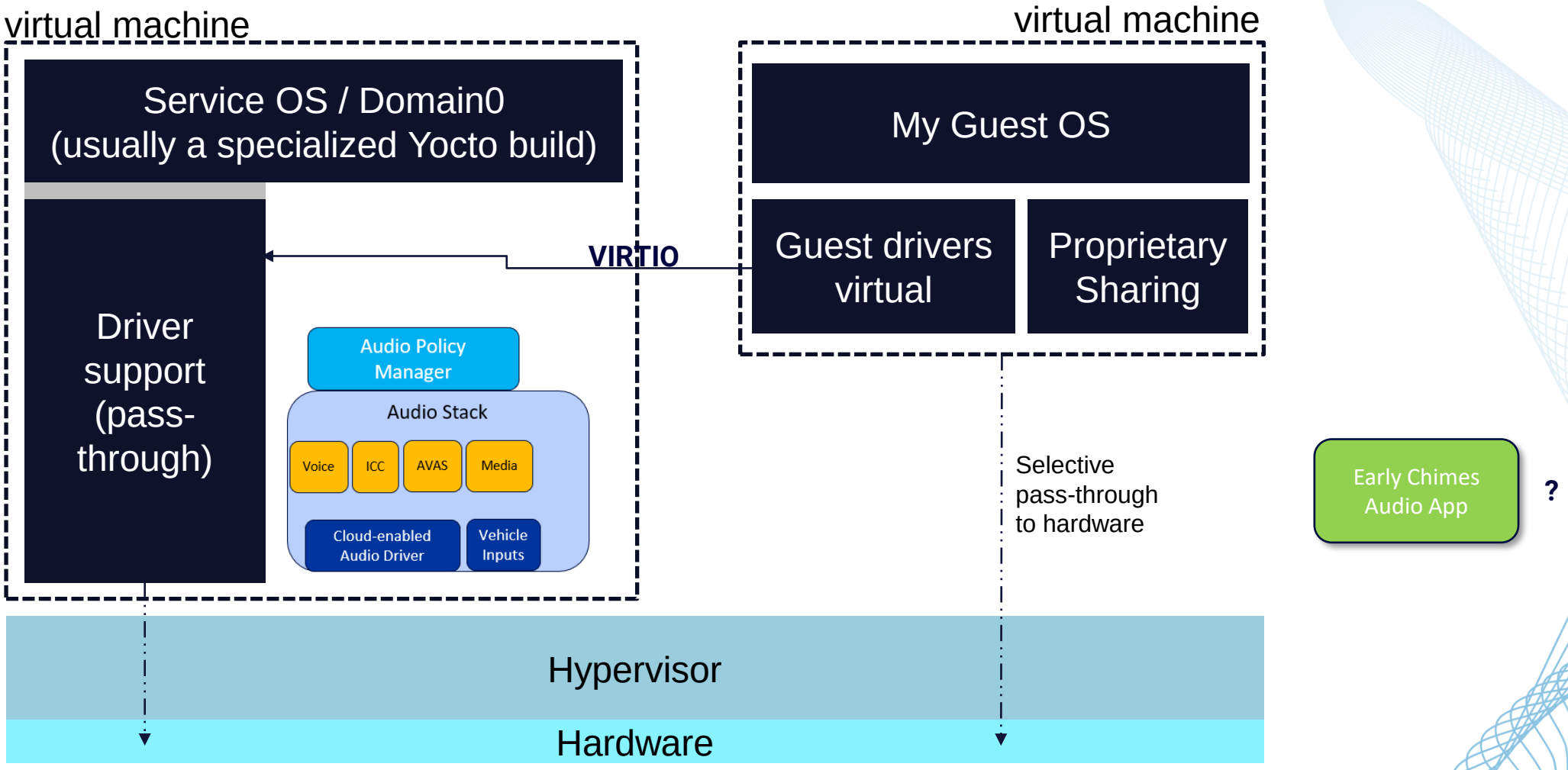
Acoustics in the Cloud



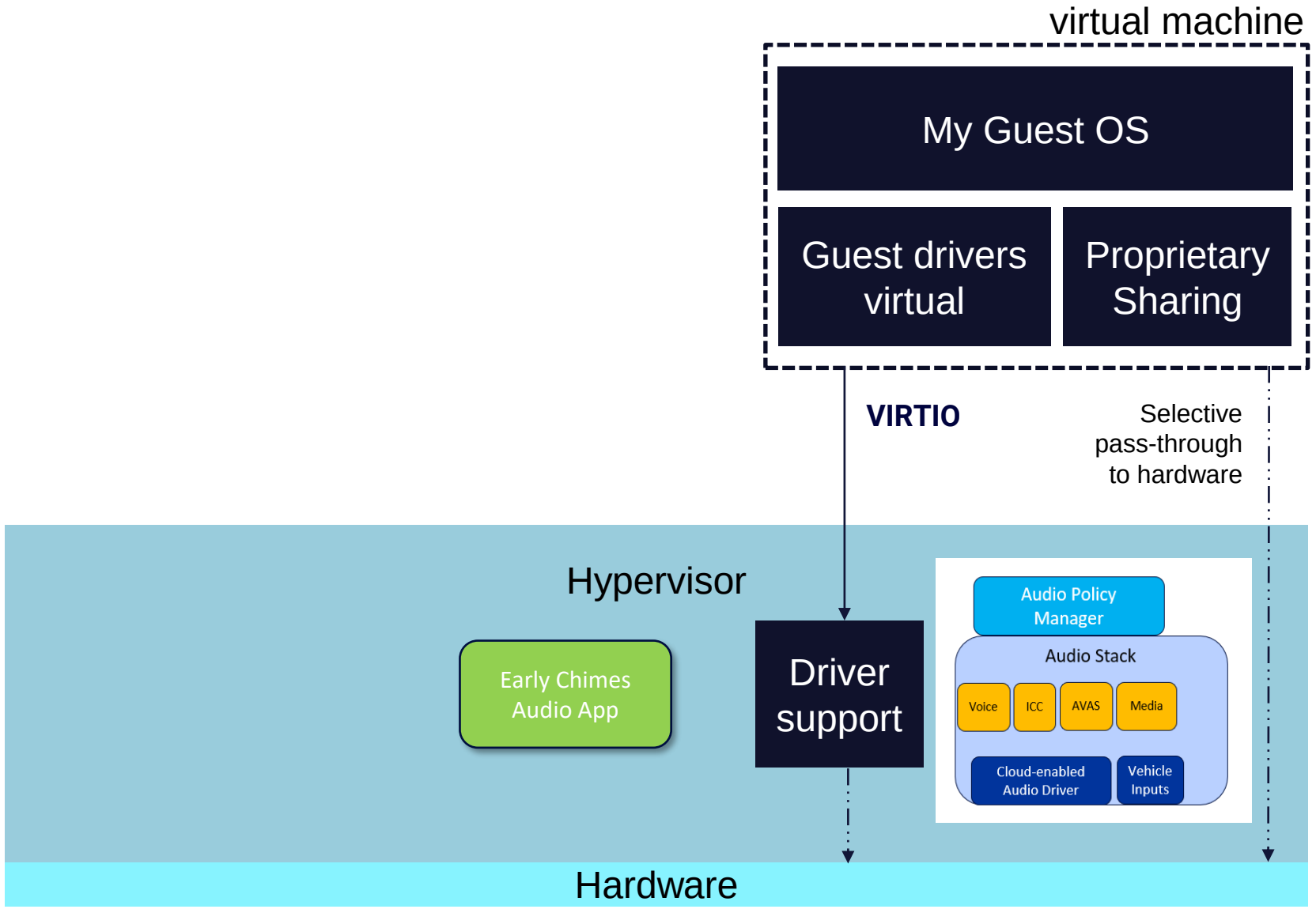
Acoustics on the SoC



How would I build audio sharing in a Service OS design?



How would I build audio sharing in a Hypervisor Host?



QNX Hypervisor 8.0 - Core Features

- **Leverage the performance, real-time, and security features of QNX SDP 8.0**
  - GCC 12.2 with updated security (ALSR, RELRO, Fortified function, stack canaries)
  - Core clusters, jitter-free scheduling, Memory allocator, interrupt thread processing

- **Enhancing our Virtual Device Developers Toolkit**
  - VIRTIO and non-VIRTIO: Virtio-media, USB over IP (cloud), Cloud Bluetooth, DRM (Widevine L1), machine learning, Android SDV
  - More examples, more feature support (e.g. power awareness)

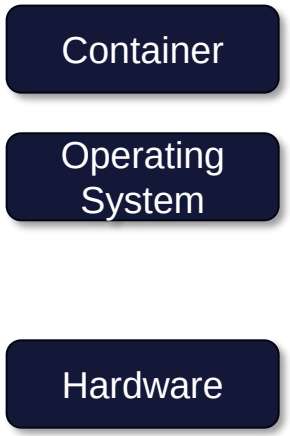
- **Virtualization Host Extensions (VHE)**
  - If SoC is ARMv8.1 or higher, enable VHE and greatly reduce/eliminate guest exit/enter times
  - Estimated at 20% typical performance boost

- **QNX Hypervisor for Safety 8.0**

Containers Overview

Containers

OS Virtualization



Operating systems must match:
e.g. QNX Container on QNX OS, Linux Container on Linux OS.

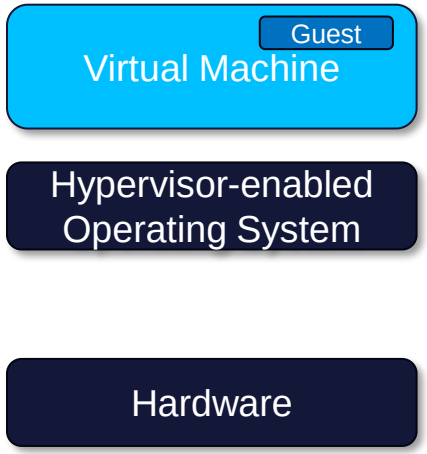
Container holds a set of programs that run in a restricted environment

Container Model

QNX Containers

VS

HW Virtualization



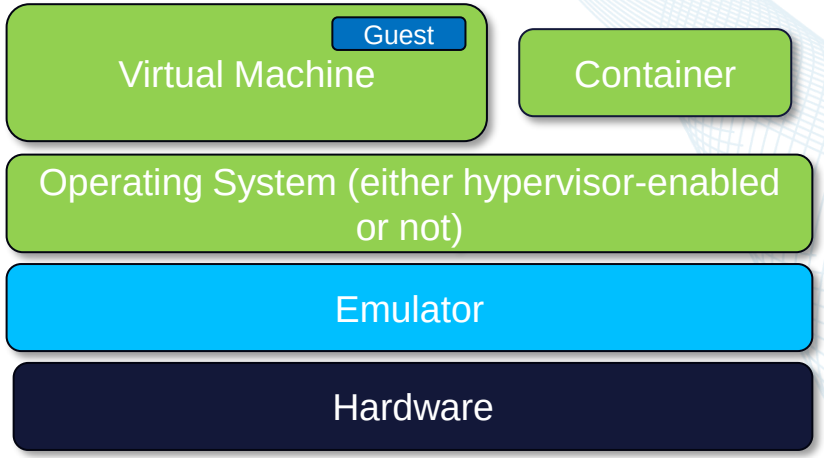
Hardware must match between virtual machine and operating system:
But Guest OS running in virtual machine can be a different OS than underlying Operating System, but must be same hardware.

Virtual Machine Model

QNX Hypervisor

VS

HW Emulation



Emulator can present different hardware to the operating system than real hardware.
A Virtual machine will use HW virtualization on top of emulator.
Container will use OS virtualization on top of emulator.

Emulation Model

Glossary

- **Container**
  - “Containers are executable units of software that package application code along with its libraries and dependencies. They allow code to run in any computing environment, whether it be desktop, traditional IT or cloud infrastructure.”

- **OCI - Open Container Initiative** https://opencontainers.org/
  - an open governance structure for creating open industry standards around container formats and runtimes (specifications are free to all)

- **Docker** www.docker.com
  - an ‘open’ platform for developing, shipping, and running applications
  - Commercially licensed via subscription. Free options available for personal use only.

- **Kubernetes** https://Kubernetes.io
  - open-source system for automating deployment, scaling, and management of containerized applications (Apache v2, free to all)
  - Uses CRI Container Runtime Interface. Specific to Kubernetes.
    https://kubernetes.io/docs/concepts/architecture/cri/
    - Allows any vendor to support Kubernetes orchestration
  - CRI-O (Orchestrator) is the implementation

Why use Containers in Embedded

- **Packaging Solution:** Bundle the necessary software artifacts into a contained environment for testing and deployment.
- **Over the Air Updates:** Utilize entire ecosystem of container orchestration tools like Kubernetes to manage deployment to Target.
- **Restricted Runtime:** Run a set of processes in a restricted environment.
- **CI/CD Pipelines:** Develop and test in the Cloud at scale; code deployed in containers can be easily integrated with customer's CI/CD pipeline designed using open-source tools and frameworks.
- **Isolated In-Field Testing:** Deploy beta version of software stack to target without interfering in existing processes. Isolation across container instances ensure changes in one do not affect the others.
- **SBOMs**: Software dependencies embedded within the container make generating an SBOM (Software Bill of Materials) more convenient.

Container Trends

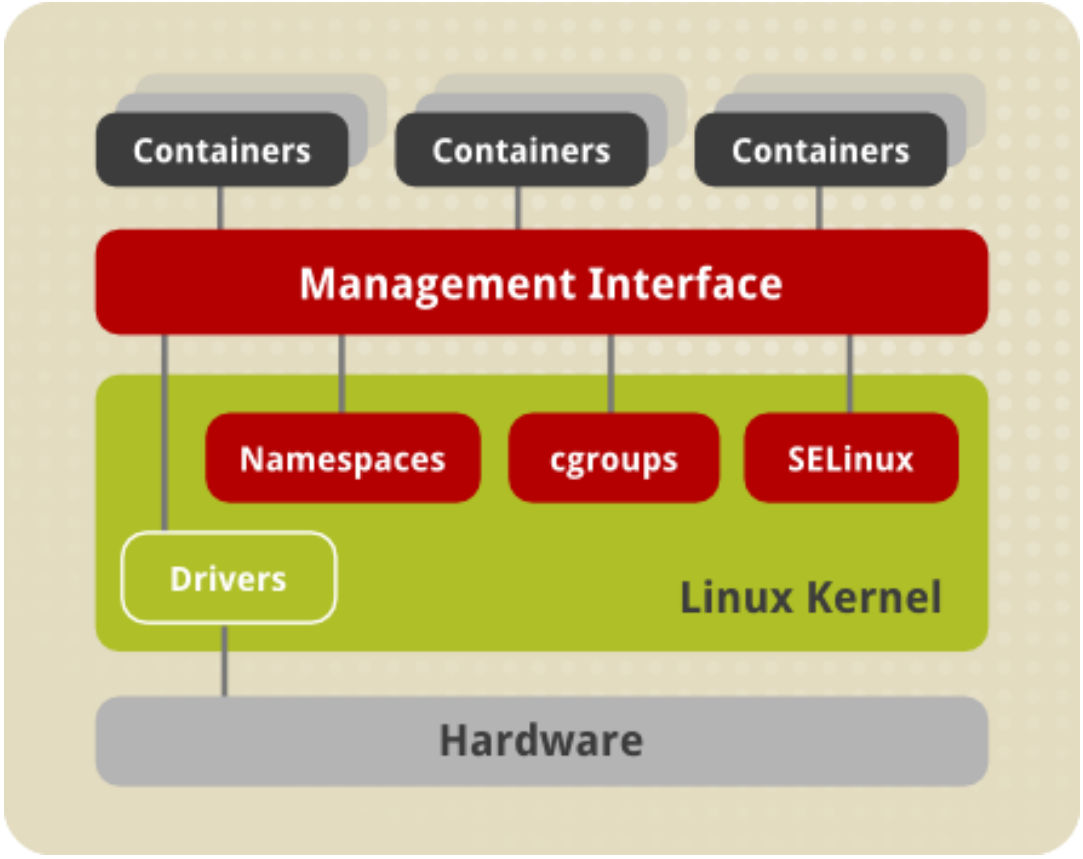
As per VDC Research Report 2023:

- Over 50% of automotive OEMs either already have or expect to have OTA (Over-The-Air) software support within next 3 years
  - ISO 21434 (Security update requirements) will be a force driving the adoption of containers used for OTA

Kubernetes is lead orchestration software (as the base connection agent for most tool solutions)

- Source
  - https://www.datatronic.hu/en/containerisation-in-automotive-industry/

Linux Containers used as Foundation



- **Namespaces**
  - Filesystem isolation (restricted view from within container)
  - Container has its own hostname (Nodename and domainname)
  - Isolation of shared memory objects and message queues per container
  - Unique process view of processes per container
  - Separate virtual network stack, firewall, routing
  - Root operation available within container but not root access to outside

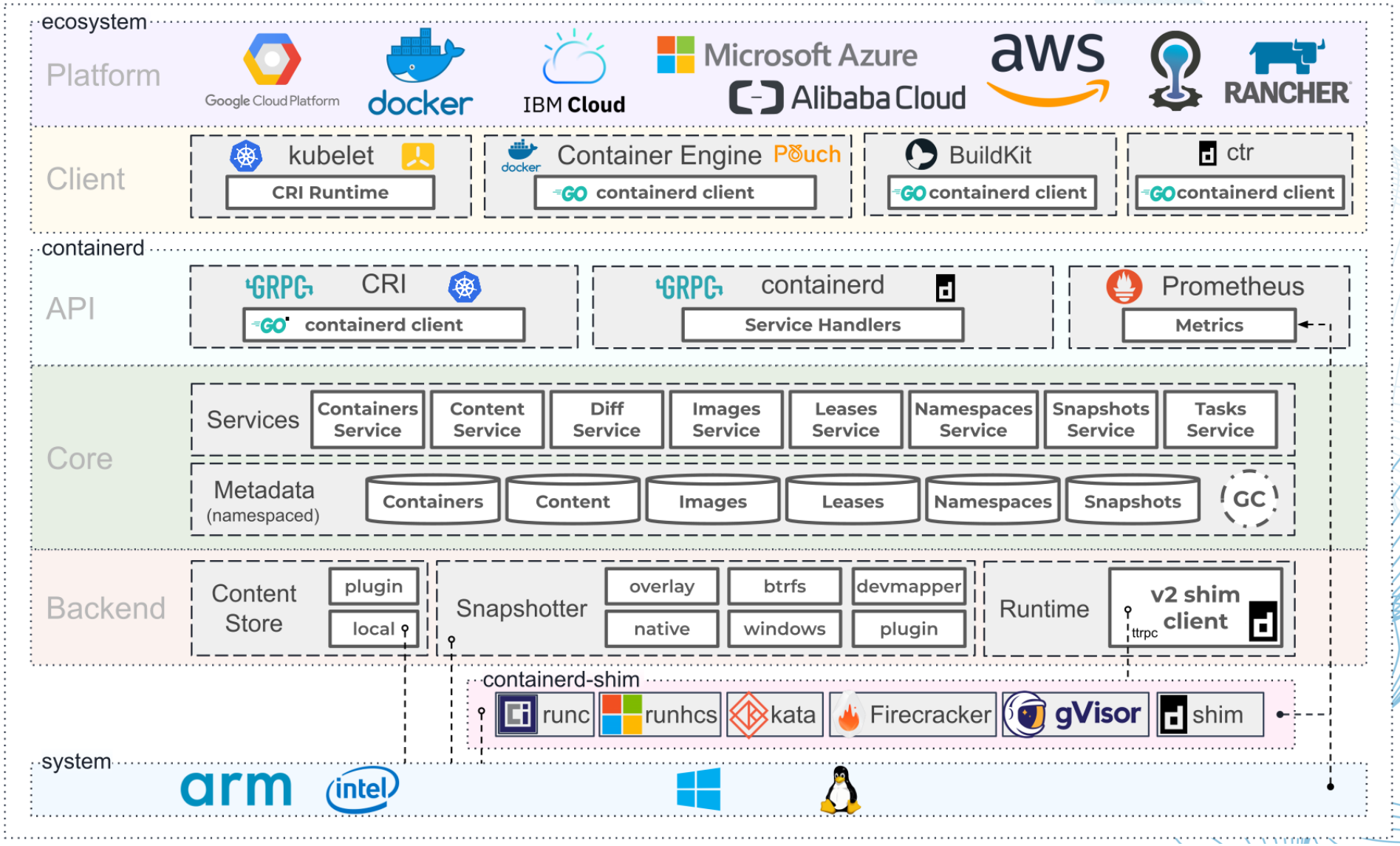
- **Control Groups** (cgroups)
  - Allocation/restriction of
    - CPU time
    - Core allocation
    - Device allocation
    - System memory
    - Network bandwidth
    - Monitoring
    - Disk bandwidth (limits on input/output access to block devices)

- **SELinux**
  - Security policy provides secure separation of containers
  - Including virtual devices (sVirt)

But how to bring this to Embedded?

Linux/Windows ‘containerd’ example: millions of lines of code

Manages the complete container lifecycle of its host system, from image transfer and storage to container execution and supervision to low-level storage to network attachments and beyond.



BlackBerry | QNX

QNX Containers

QNX Containers: Value Statement

- QNX Containers provide a QNX-based edge device with a standards-based environment for the running and management of container technology.

- **Standards-based** solution:
  - OCI compliant.
  - Kubernetes-based toolchains for creation, deployment and management.
  - Docker (industry standard) repositories are used for remote storage and retrieval. Local storage is also supported.

- Follows the restrictions and security features available with QNX SDP8 including restrictions on networking, filesystems, devices, memory, communications, access control and CPU.

- QNX Containers has its own TARA (Threat and Risk Analysis) as per our product release policy and ISO 21434 compliance.

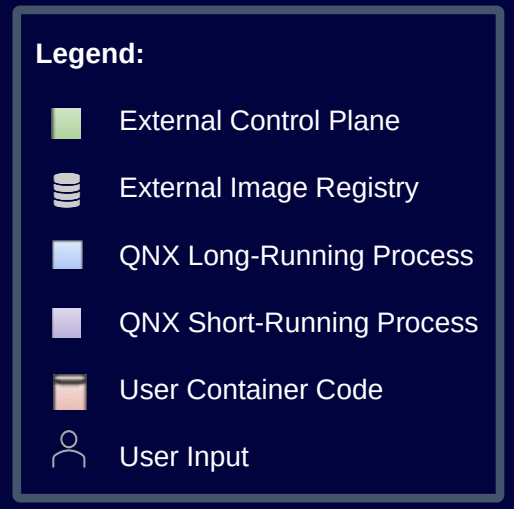
- A **safety-certified** version is planned for future product.

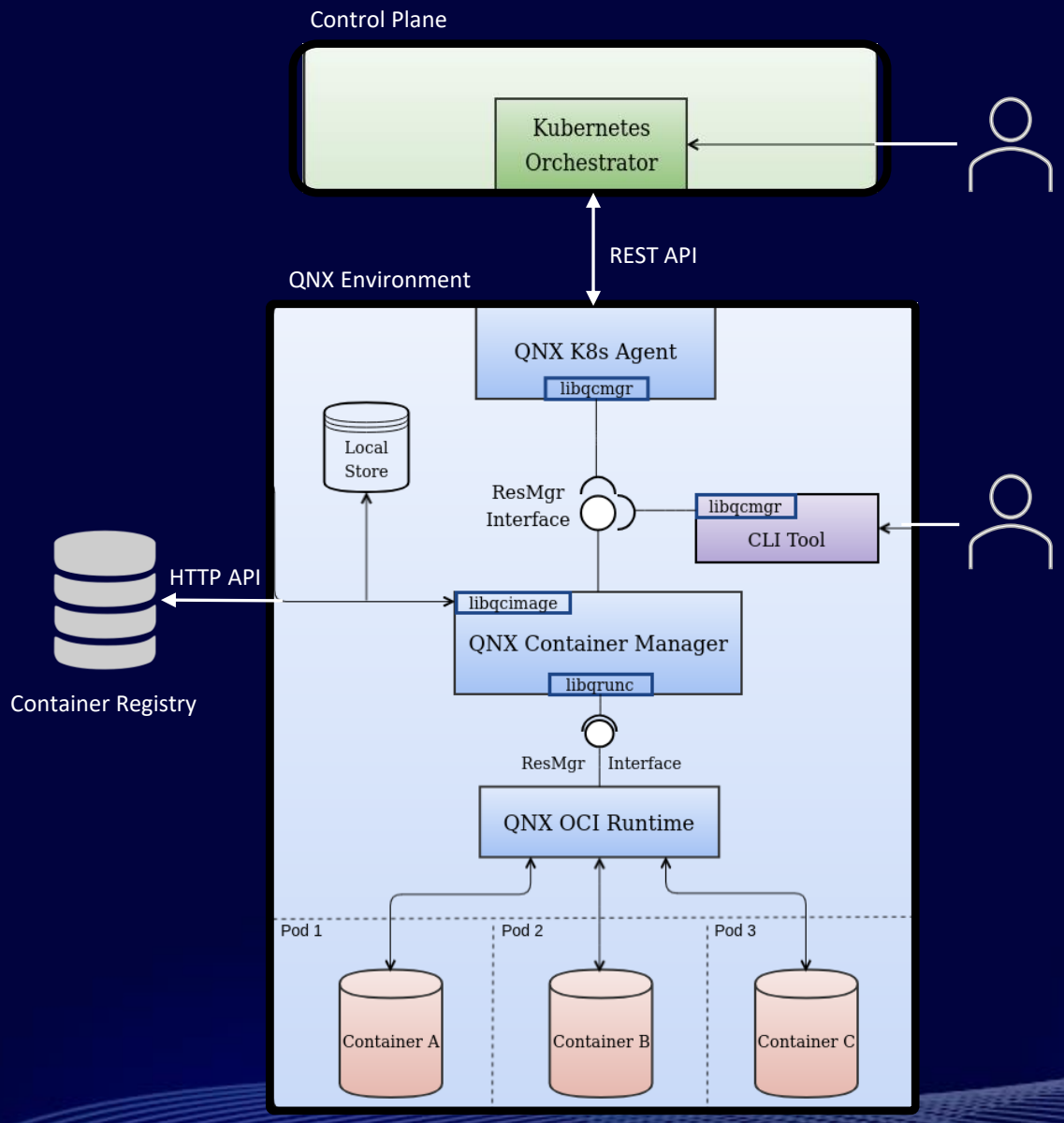
- This restriction set provides **highly secure and isolated embedded containers** while still maintaining the high performance and hard realtime nature of the QNX SDP8 operating system.

- QNX Containers co-exist with the QNX Hypervisor environment, allowing for **simultaneous use of both virtual machines and containers.**

- QNX Containers are **extendable** without compromising existing design.

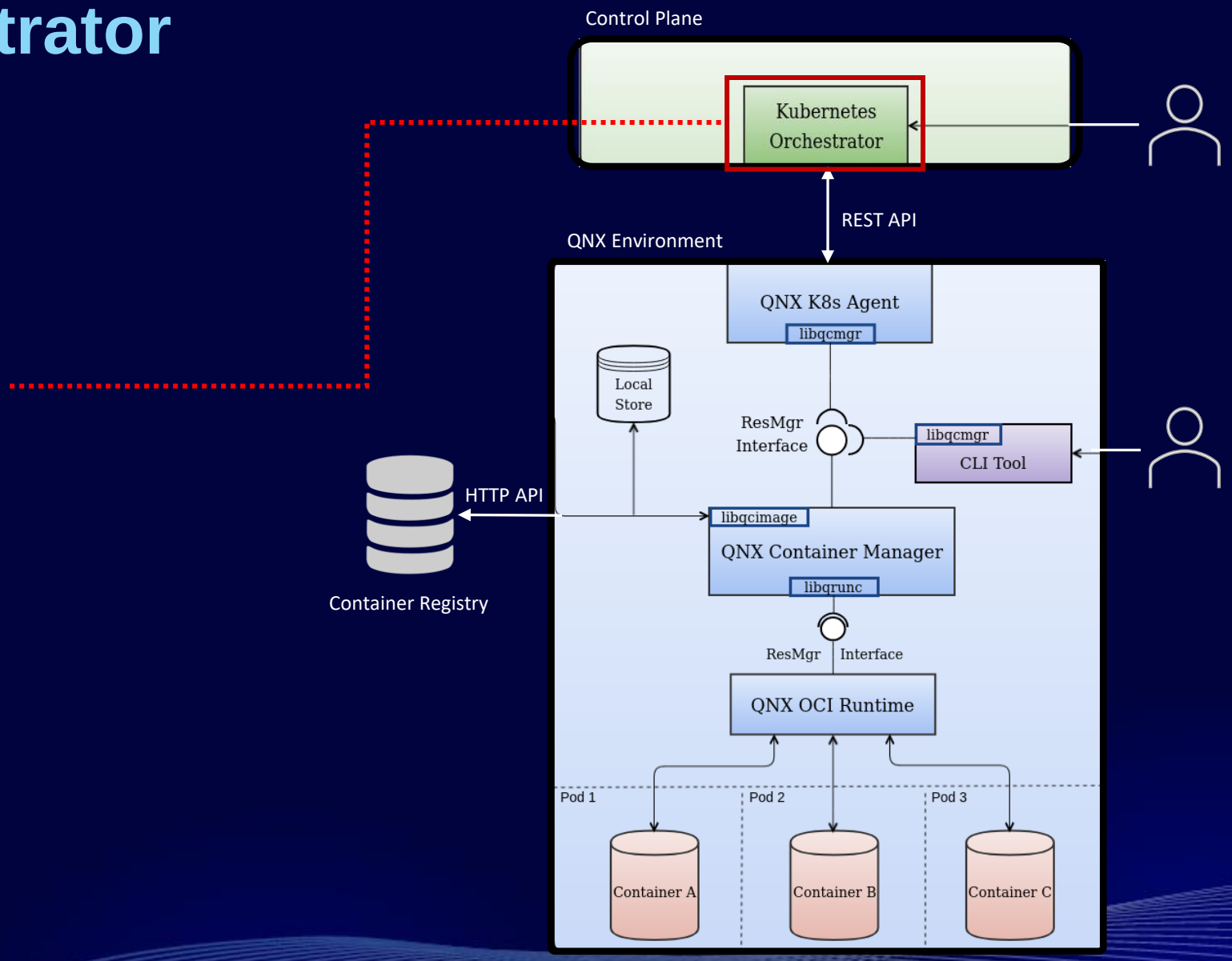
System Architecture





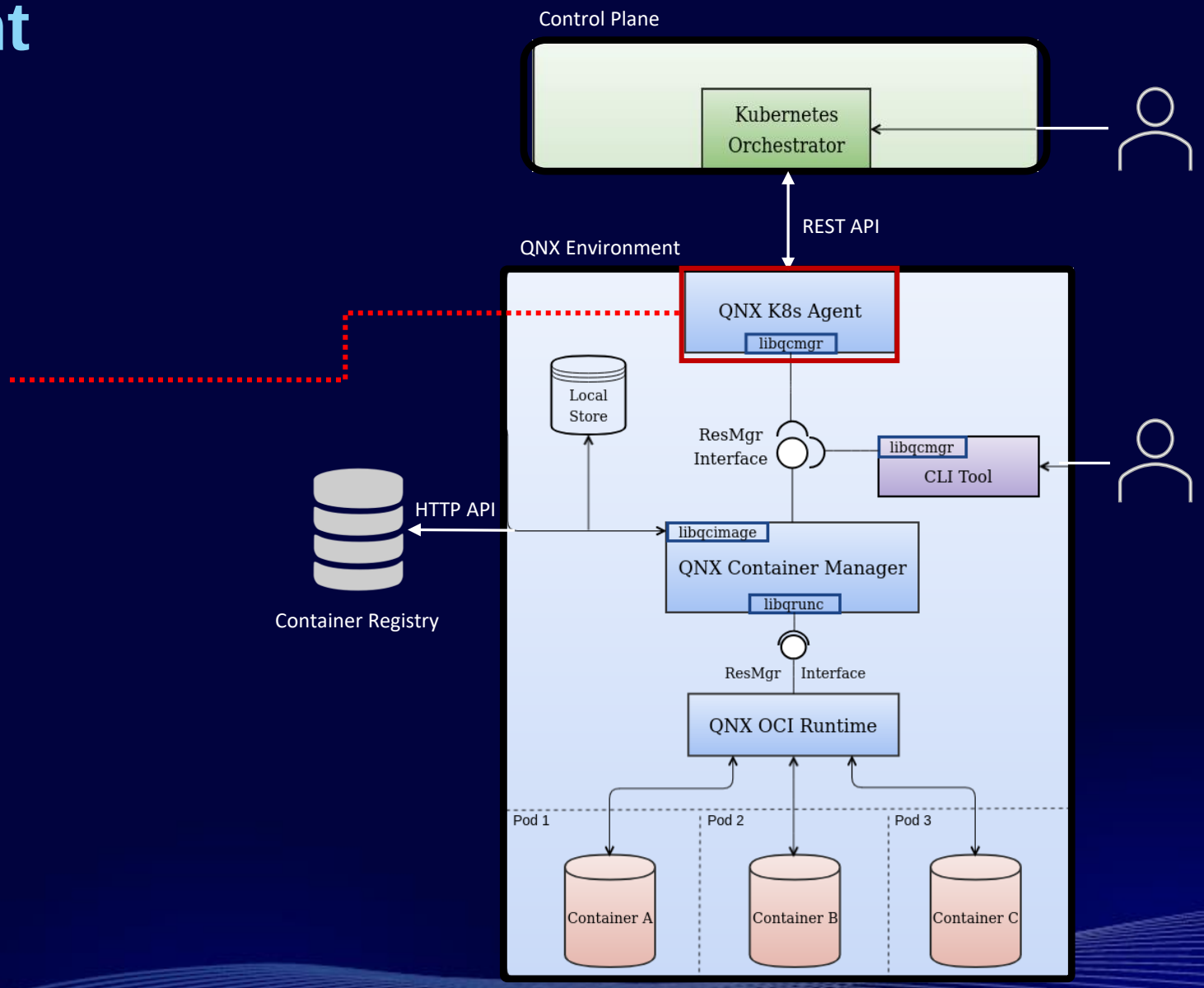
Kubernetes Orchestrator

- Chose an orchestrator-driven solution
- Orchestration tools manage multiple nodes and automate provisioning, deployment, networking, scaling, availability and lifecycle management of containers across nodes
- Kubernetes is a widely adopted open-source container orchestration tool popular in cloud computing
- Typically hosted separate and designed to be used as a control plane of a multi-node container solution



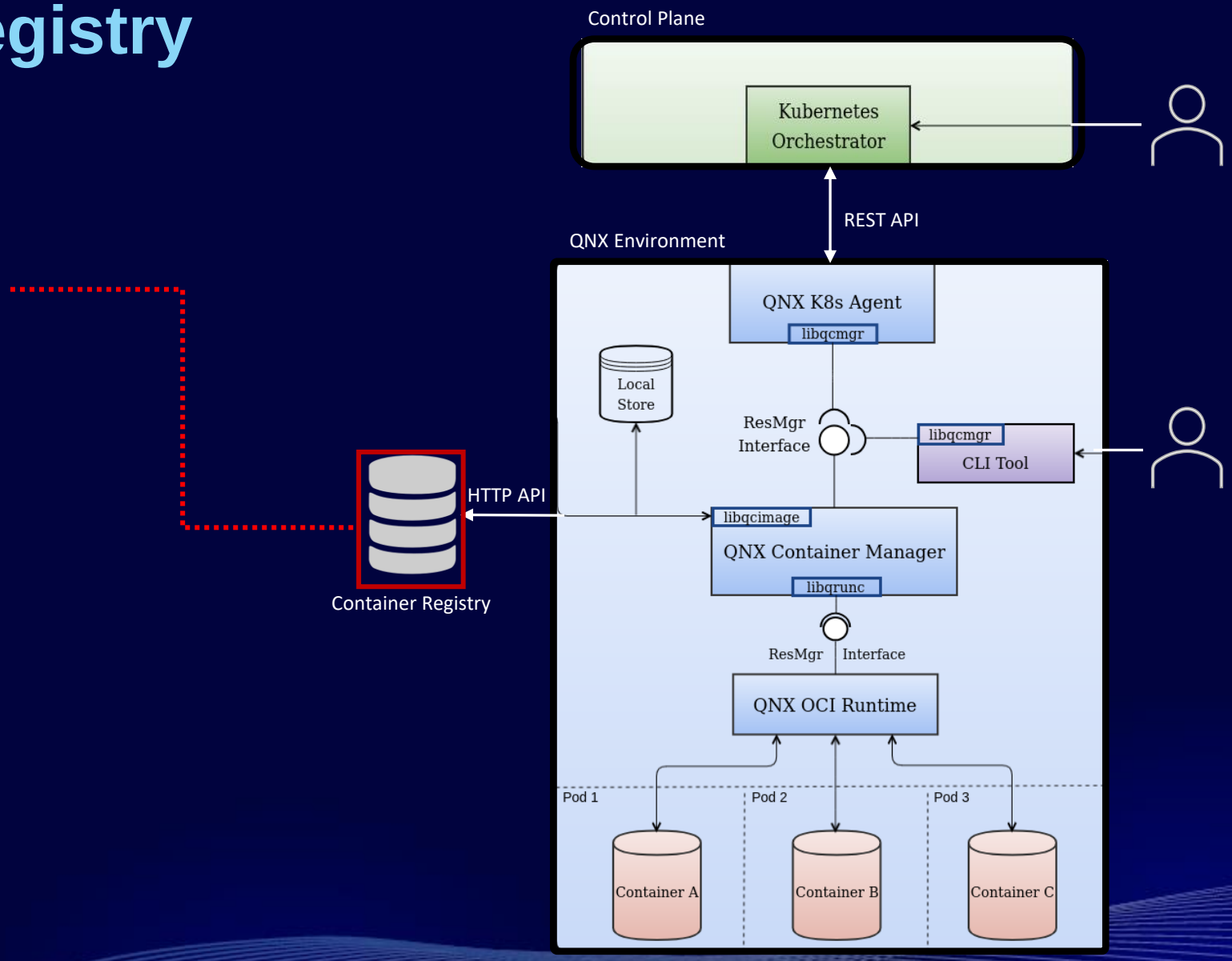
QNX Kubernetes Agent

- Managed Kubernetes nodes require an agent, a Kubelet (Qubelet in QNX)
- Responsible for communication with Kubernetes controller for creating and running pods
- Pods are the smallest unit of configuration, representing a set of one or more containers and their configurations
- RESTful APIs are used for communication (representational state transfer)
- Listens for new pod configuration messages from the control plane and forwards requests through QNX resmgr framework to the QNX Container Manager



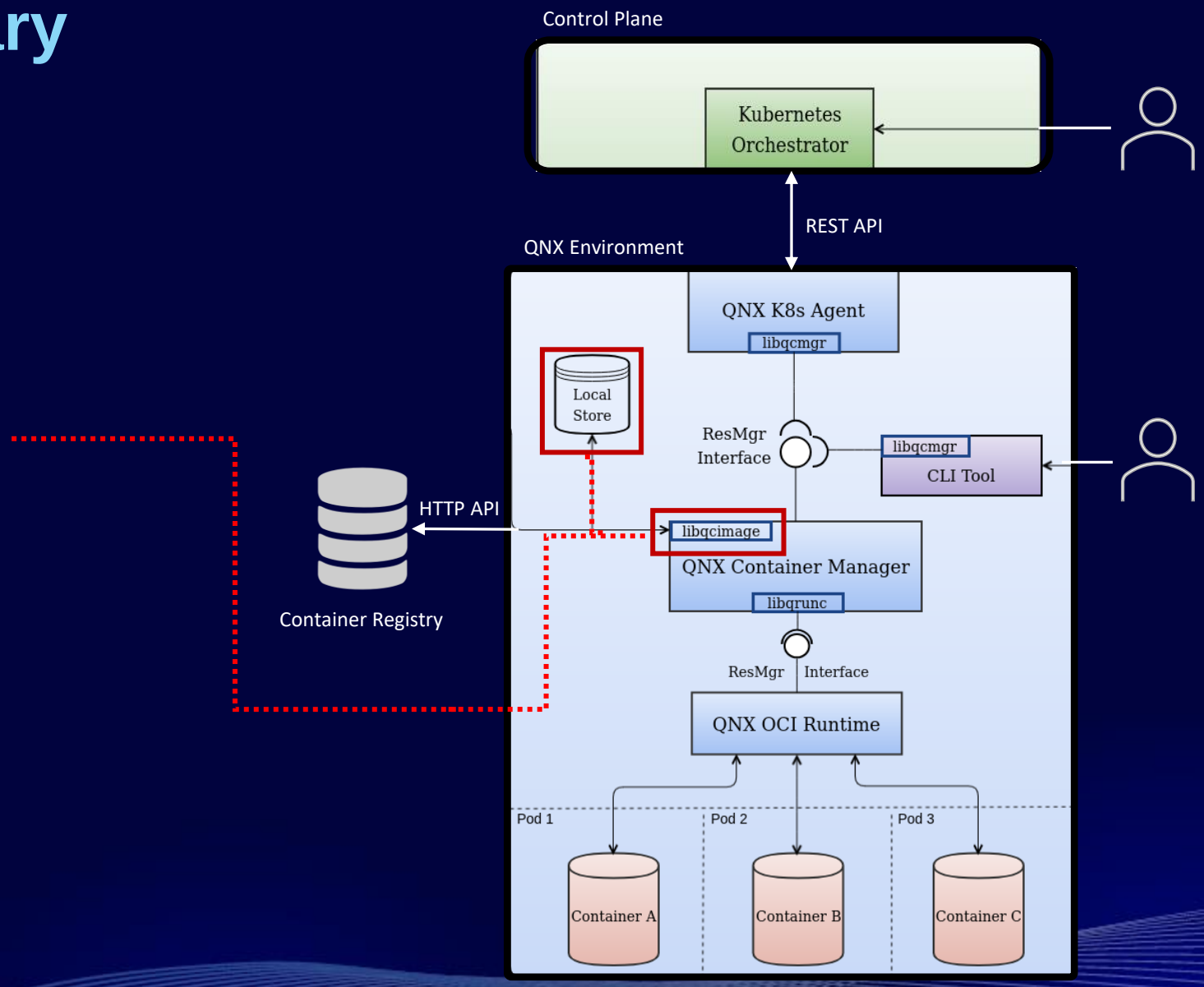
Container Image Registry

- Third party server housing a collection of container image repositories
- Provides developers the means to easily store / share container images, while managing access control, permissions and authentication
- Docker Hub and AWS Elastic Container Registry (ECR) are popular examples



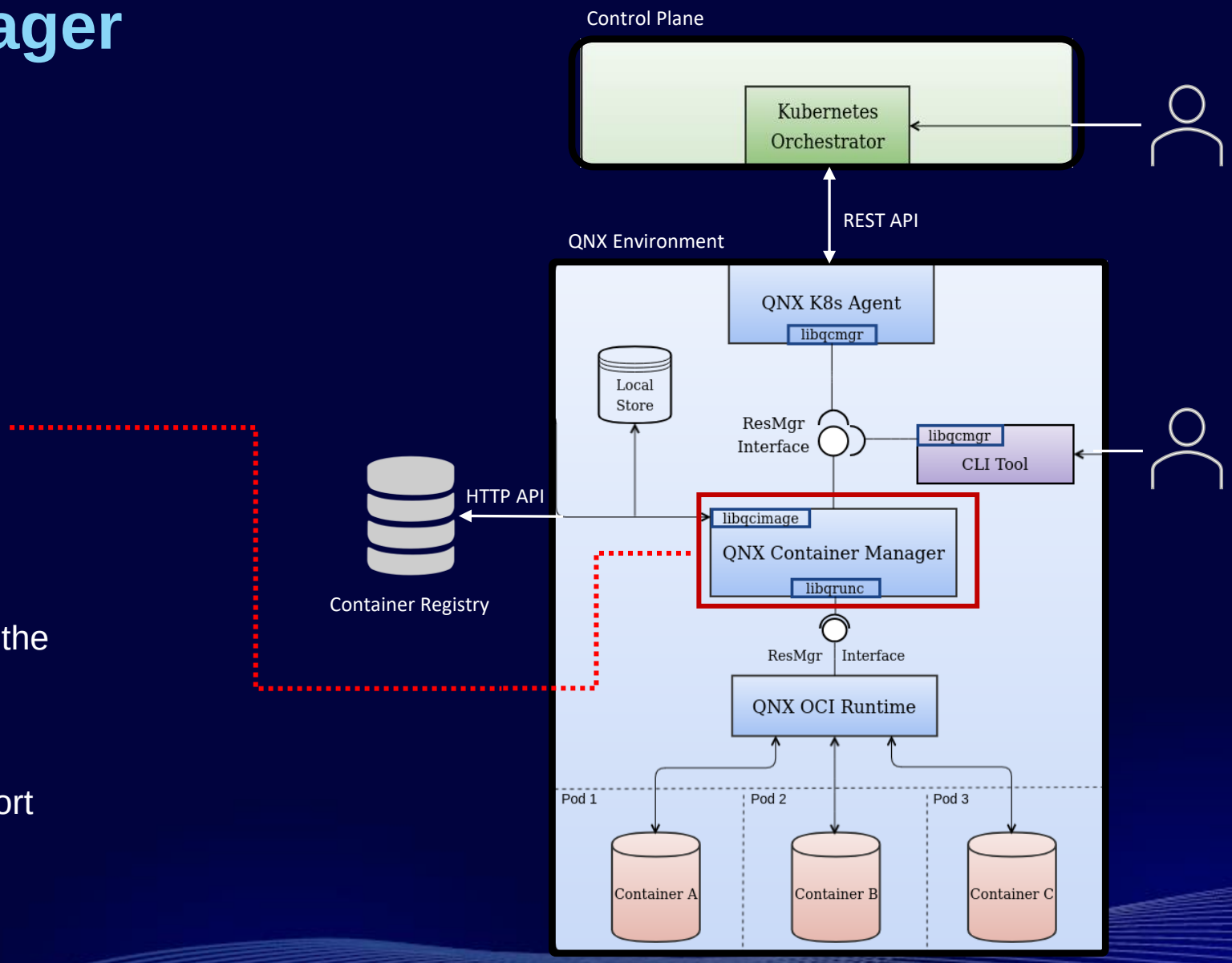
Container Image Library

- Library responsible for downloading and managing images from a supported container registry
- Supports Docker Registry HTTP API V2
  - Any Private/Public AWS ECR
  - Docker Hub public registry
- Supports standard image formats
  - Docker Image V2, Schema 2
  - OCI Image Specification
- Uses local store to cache downloaded images and provide on-target access
- Once downloaded, image is unpacked into an Open Container Initiative (OCI) runtime bundle for the Low-level Container Runtime to consume



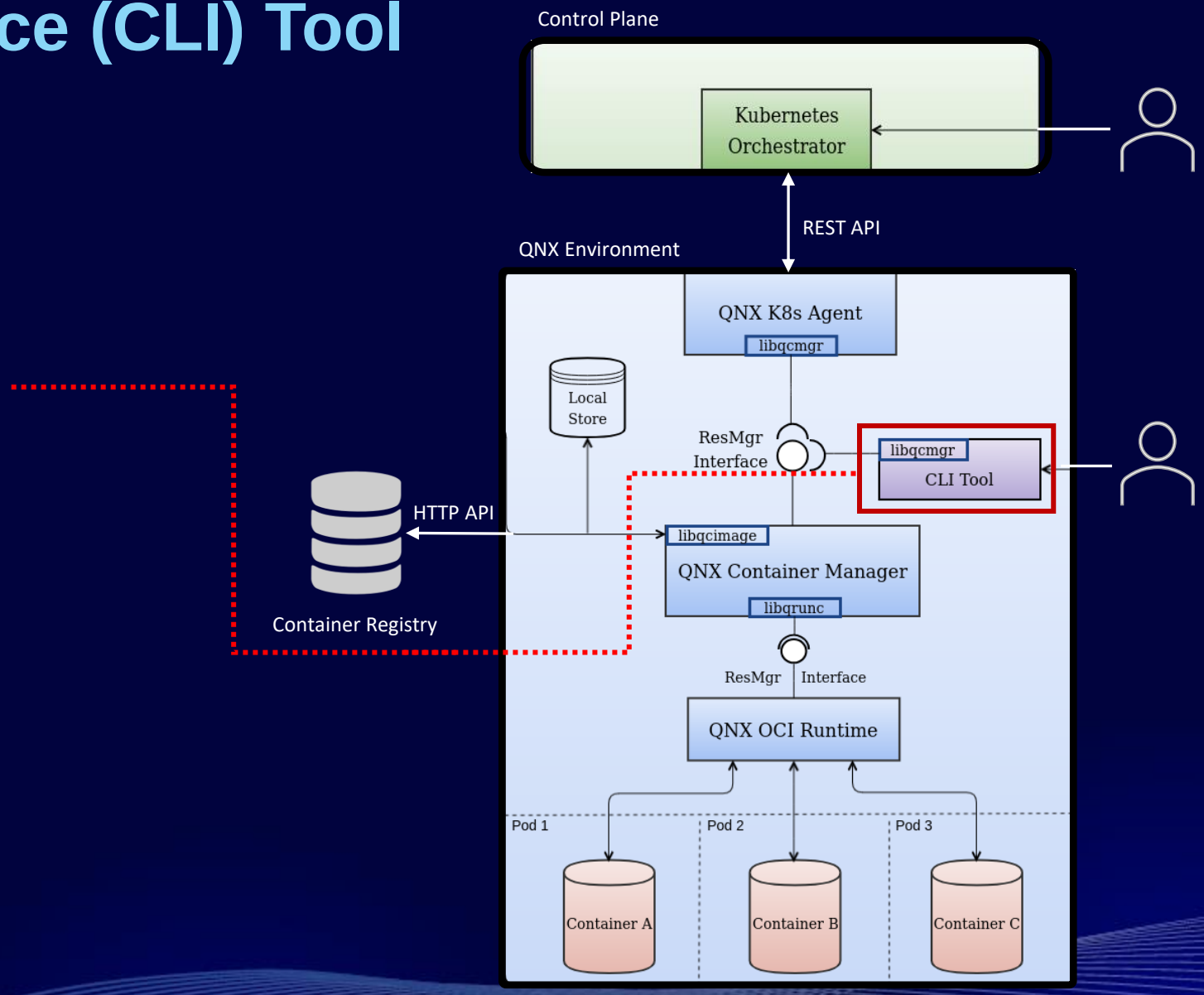
QNX Container Manager

- Responsible for the management of containers on a single node (QNX Runtime Environment)
- Listens and acts on control messages from the orchestration control plane
- Interacts with the Container Image Library for image management
- Interacts with the Low-level Container Runtime for runtime management
- Monitors container states and notifies the control plane of any updates
- Qubelet separation from Qcmgr hides K8s details and allows for future support of other orchestration tools



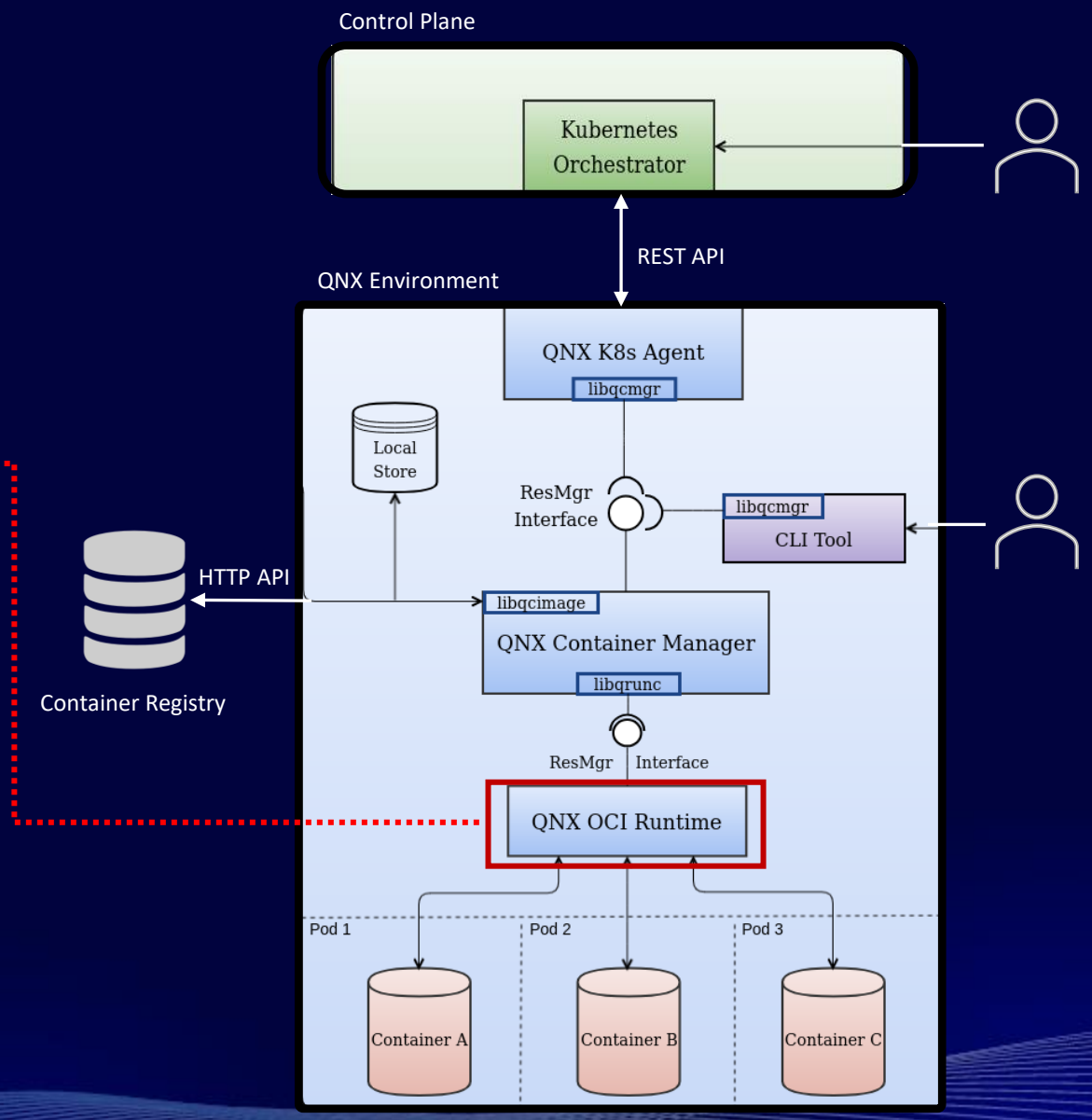
Command Line Interface (CLI) Tool

- Command and Control CLI tool to manage container functionality
- Designed to utilize the same resource manager interface between Qubelet and the Container Manager
- Provided as an option for users who want to either:
  - Use the QNX Containers solution without Kubernetes orchestration and/or
  - Have an on-device method to manage containers without the need for external access



Low-level Container Runtime

- Low level OCI-compliant component for running containers
- Responsible for creation and deletion of the container environment within the QNX system
- This environment consists of:
  - File system isolation
  - Network isolation
  - CPU and Memory Isolation
  - Launching and monitoring of container process
  - Handling container process logging
  - Stopping container process and cleaning up resources



Thank you