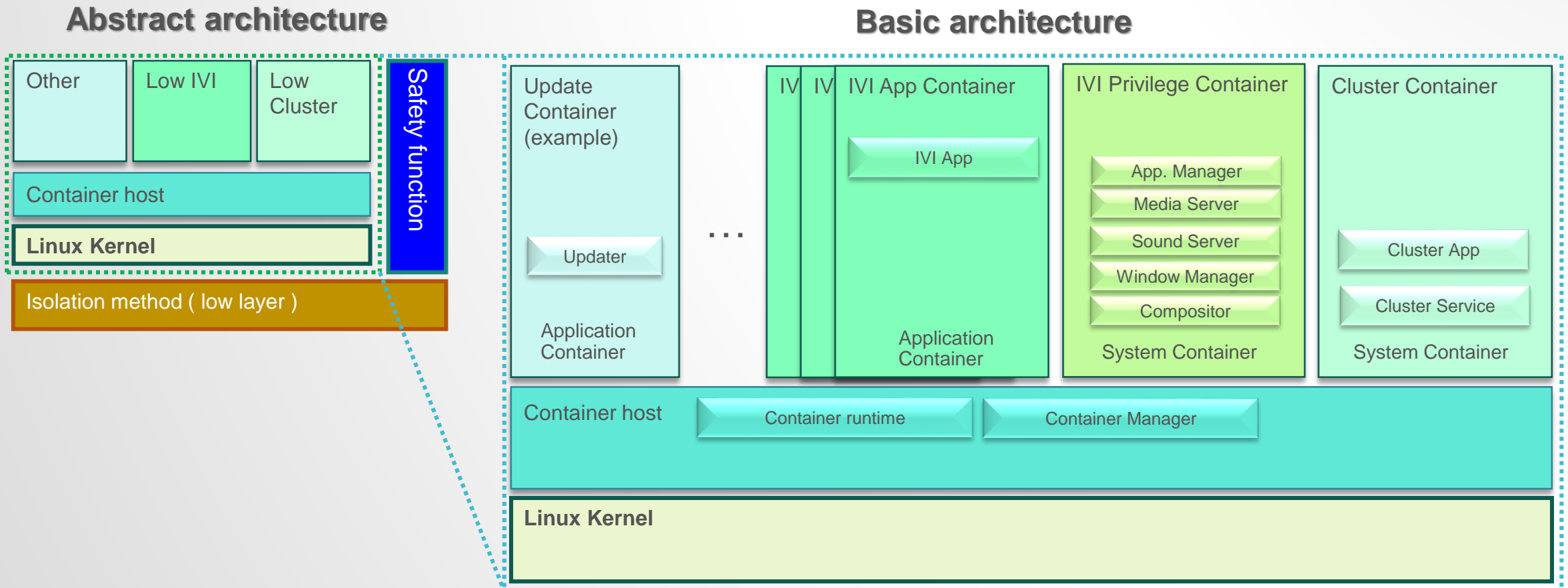# Topics

- **Architecture overview**

- Graphics

- Sound

- CAN

Container based architecture
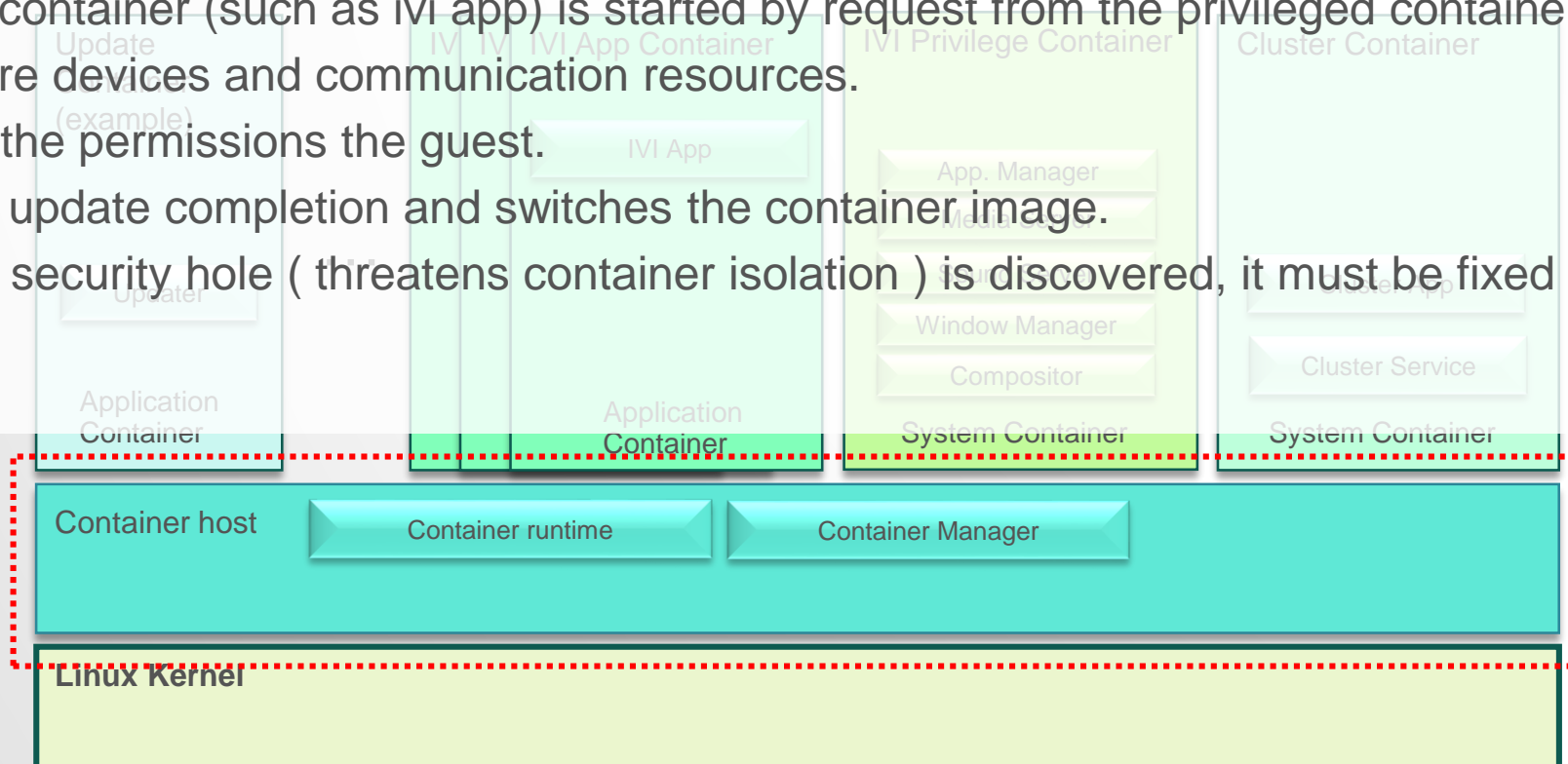
- Show Basic architecture
  - It is a breakdown of the abstract architecture.

Abstract architecture

Other	Low IVI	Low Cluster

Container host

Linux Kernel

Isolation method (low layer)

Safety function

Basic architecture

Update Container (example)

Updater

Application Container

. . .

IV IV IVI App Container

IVI App

Application Container

IVI Privilege Container

App. Manager

Media Server

Sound Server

Window Manager

Compositor

System Container

Cluster Container

Cluster App

Cluster Service

System Container

Container host

Container runtime

Container Manager

Linux Kernel

AUTOMOTIVE GRADE LINUX

Automotive Use Case – Role of each container

- ## Container host
  - Manage the lifecycle of each container.
    - Needs to be lightweight to realize fast boot.
    - The static service container (such as cluster, ivi privilege) is started at boot time, and the dynamic service container (such as ivi app) is started by request from the privileged container.
    - Configure devices and communication resources.
    - Control the permissions the guest.
    - Detects update completion and switches the container image.
    - When a security hole (threatens container isolation) is discovered, it must be fixed quickly.

Update
Service
(example)

Updater

Application
Container

IVI IVI IVI App Container

IVI App

Application
Container

IVI Privilege Container

App. Manager

Media Server

Window Manager

Compositor

System Container

Cluster Container

Cluster App

Cluster Service

System Container

Container host | Container runtime | Container Manager
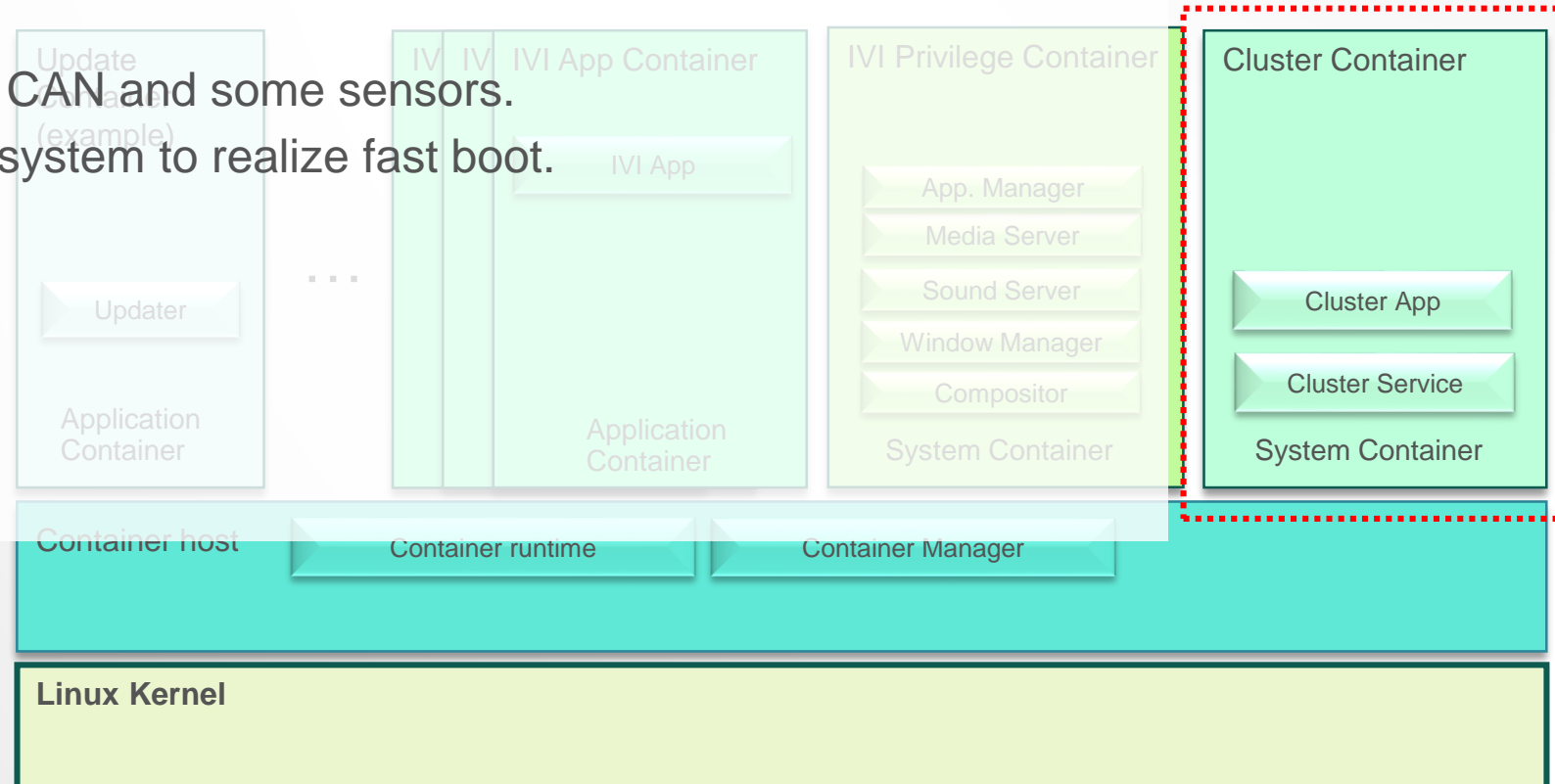
Linux Kernel

Automotive Use Case – Role of each container

- Cluster
  - Provides cluster function
    - Cluster software such as meter drawing, fuel calculation, etc. is included.
    - Built with a limited software stack. It integrate using advanced quality management method.
    - Needs a display, GPU, sound, CAN and some sensors.
   - Must be started second in the system to realize fast boot.

Update Container (example)

Updater

Application Container

...

IVI App Container

IVI App

Application Container

IVI Privilege Container

App. Manager

Media Server

Sound Server

Window Manager

Compositor

System Container

Cluster Container

Cluster App

Cluster Service

System Container

Container host   Container runtime   Container Manager

Linux Kernel

AUTOMOTIVE GRADE LINUX

Automotive Use Case – Role of each container

- ## IVI Privilege
  - ### In charge of management.
    - #### Manage to sound and graphics for guests excluding cluster.
    - #### Manage to IVI applications using container manager API.
      - ##### Capabilities, resources, etc.
    - #### Needs a display, GPU, sound.
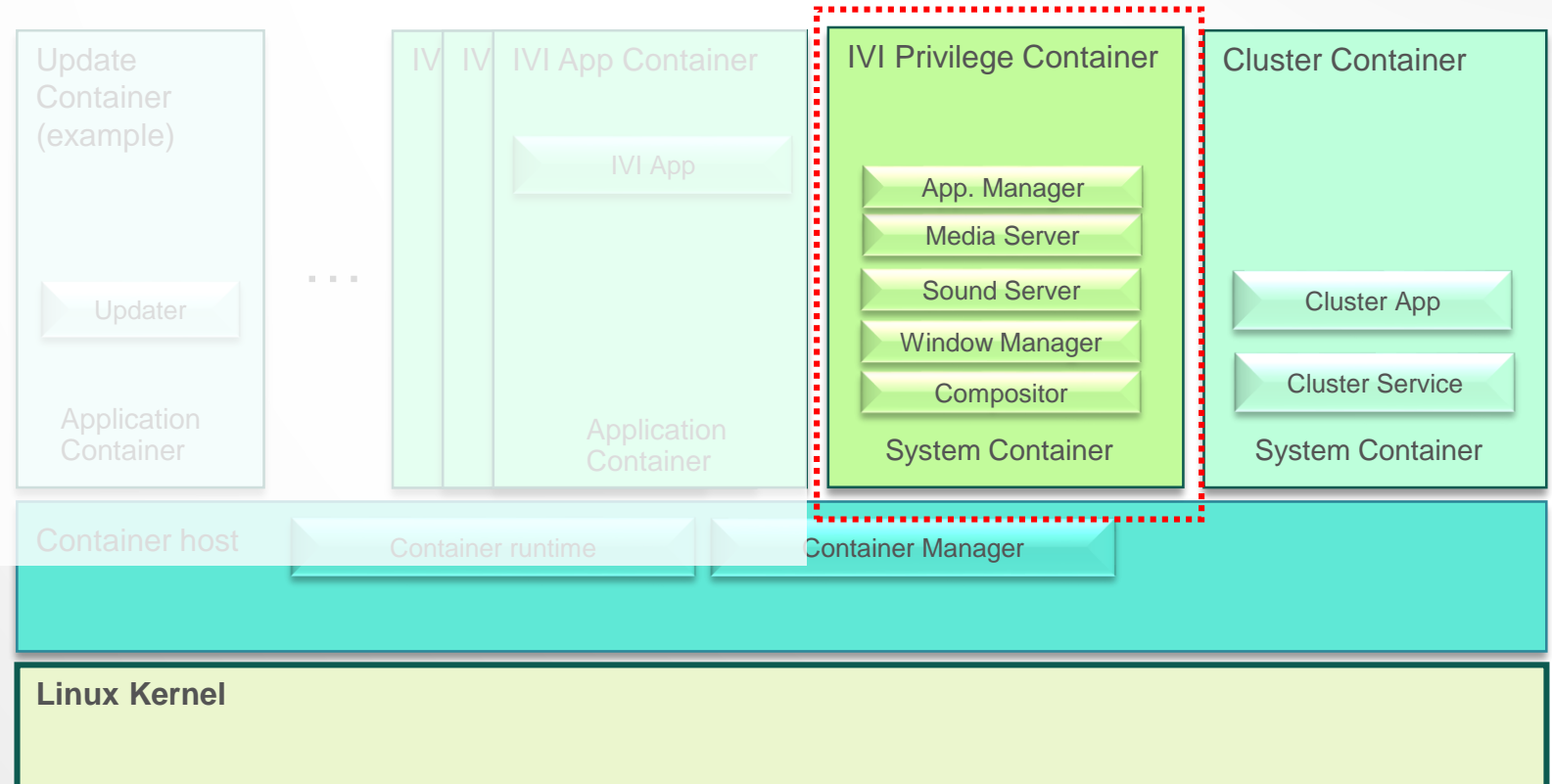
Automotive Use Case – Role of each container
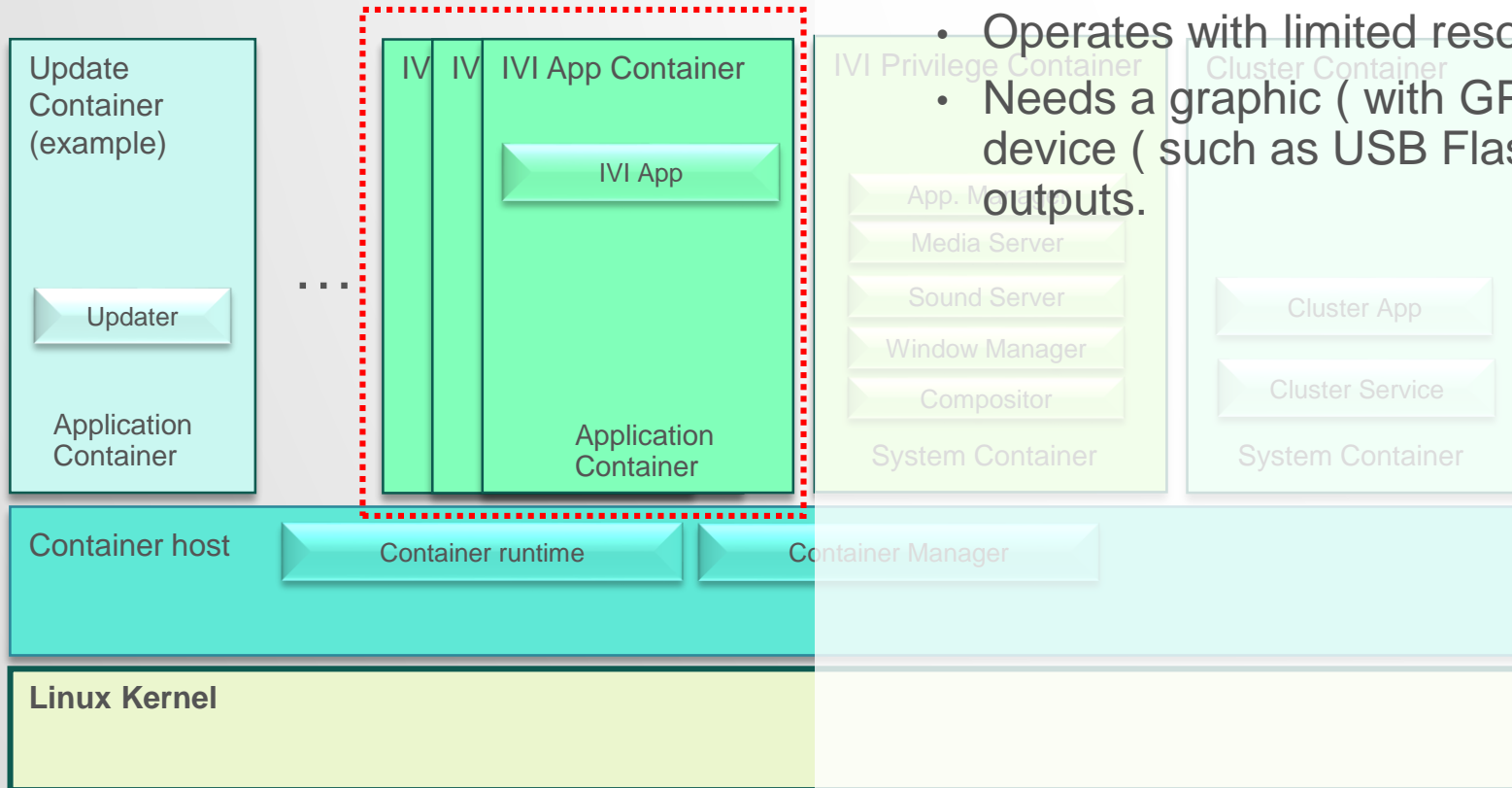
- IVI App
  - Provides IVI function
    - Divide in units of IVI applications such as silicon audio player, telephony, Car Play, etc.
    - Operates with limited resources and permissions.
    - Needs a graphic (with GPU), sound, CAN, IP network, dynamic device (such as USB Flash, SD Card) and some inputs and outputs.

Update Container (example)		IVI IVI IVI App Container	IVI Privilege Container
			Cluster Container

Updater

IVI App

App. M

Media Server

Sound Server

Cluster App

Window Manager

Cluster Service

Compositor

Application Container

Application Container

System Container

System Container

Container host

Container runtime

Container Manager

Linux Kernel

AUTOMOTIVE
GRADE LINUX

Automotive Use Case – Many Issues

- Key Issue
  - Graphics Management
    - How to isolate and share the graphics stacks.
  - Sound Management
    - How to isolate and share the sound device.
  - CAN Network Management
    - How to deliver and hide the CAN data.

- Other Issue
  - Dynamic Device (USB, SD Card, etc.) Management
  - IP Network Management
  - Container Management and Update

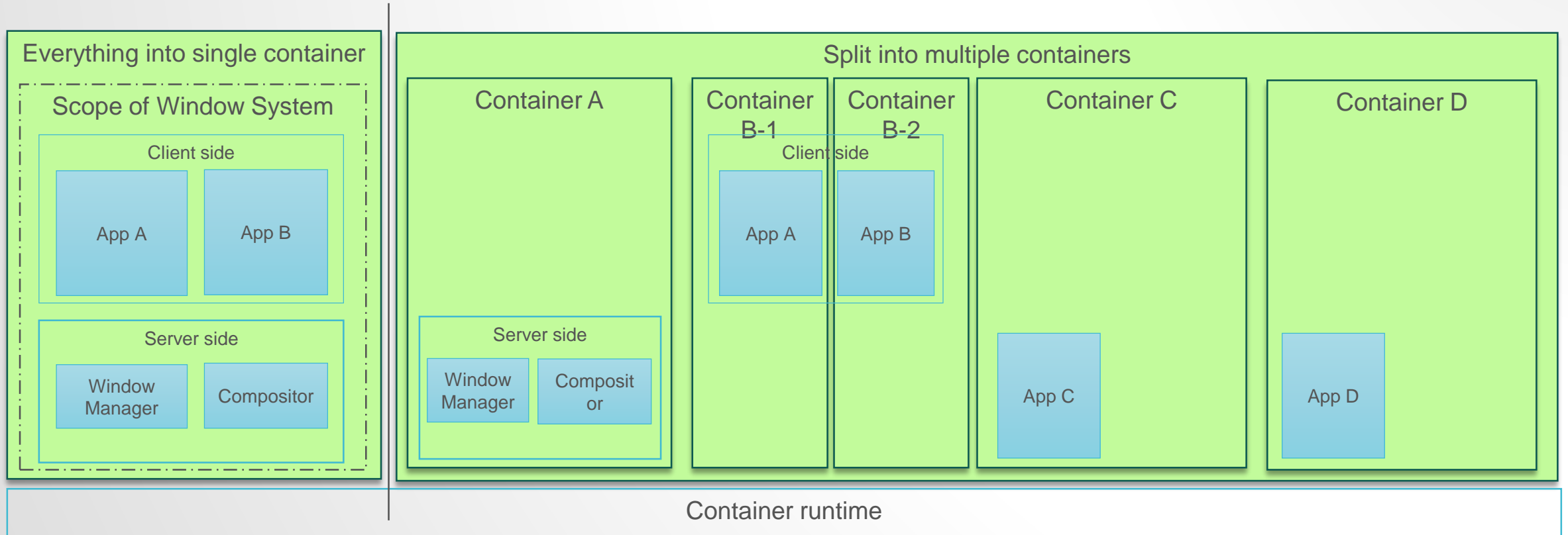# Topics

- Architecture overview

- **Graphics**

- Sound

- CAN

Architecture overview graphics

- Basic Linux graphics block diagram / types of GUI application
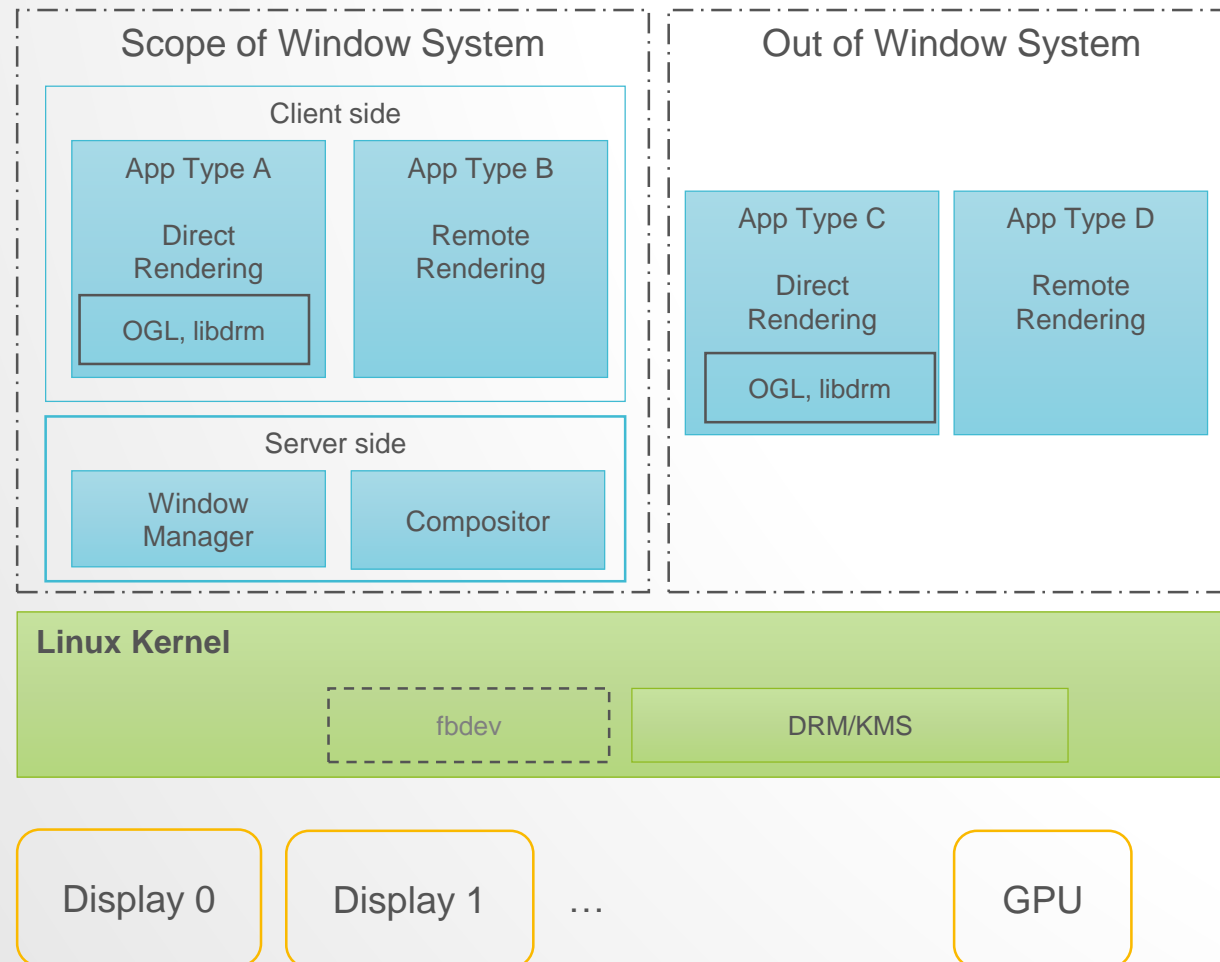
Architecture overview graphics

- Selections of Linux container integration for GUI apps

Everything into single container

Scope of Window System

Client side

App A	App B

Server side

Window Manager	Compositor

Split into multiple containers

Container A

Server side

Window Manager	Composit or

Container B-1

Client side

App A

Container B-2

App B

Container C

App C

Container D

App D

Container runtime

Linux Kernel   fbdev   DRM/KMS

Display 0	Display 1	…	GPU

Architecture overview graphics

- Selections of Linux container integration for compositor
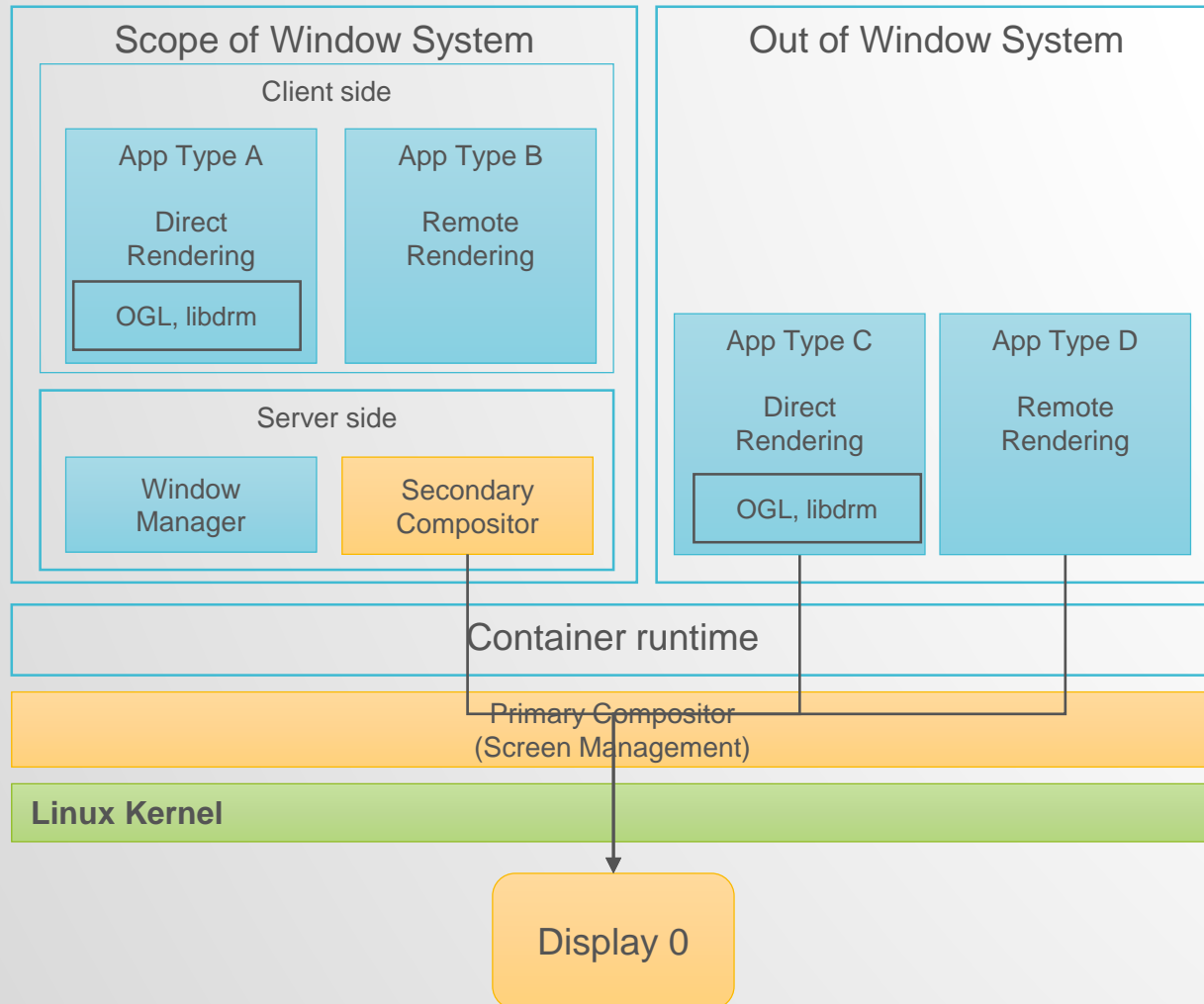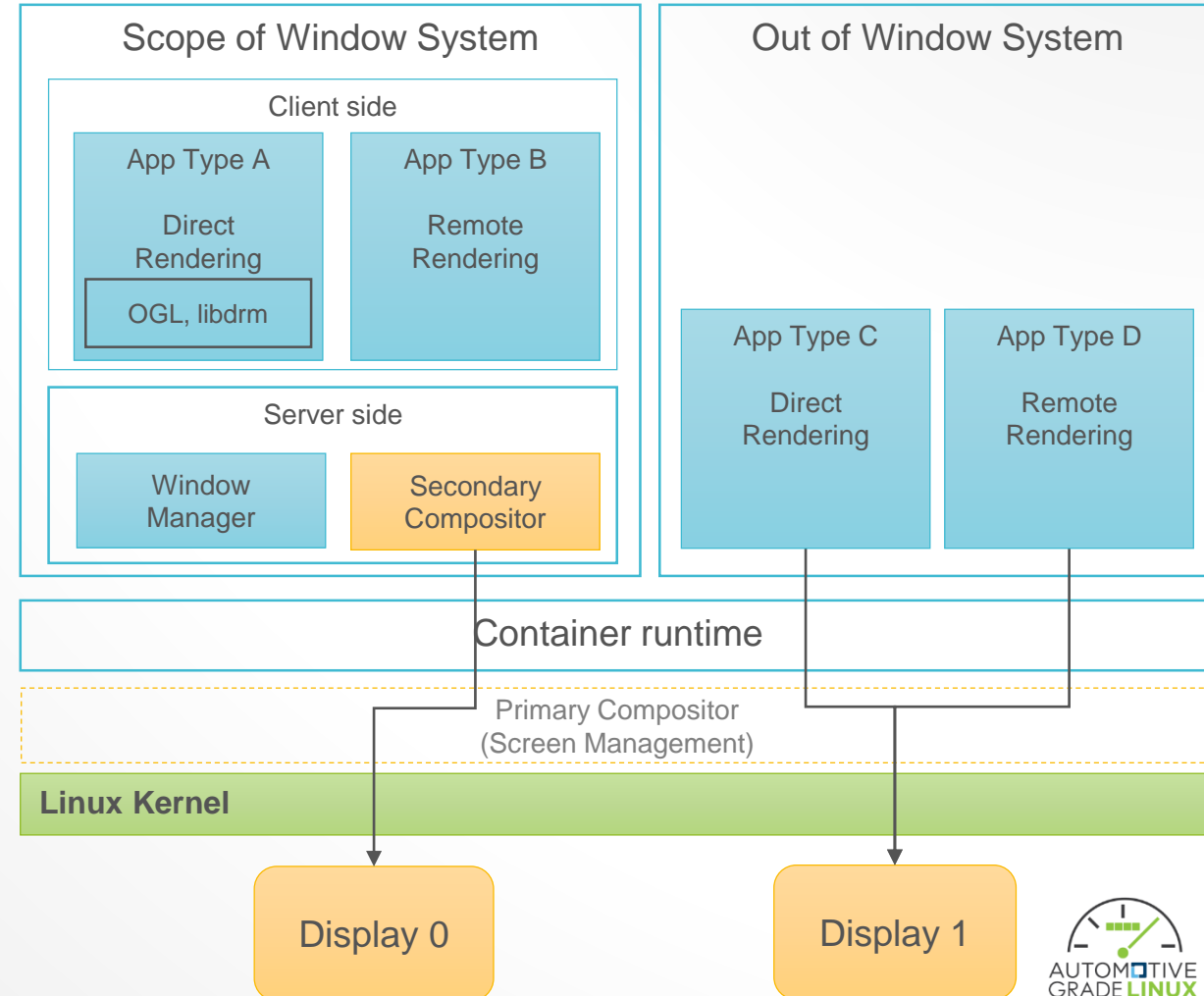
Sharing device between Safety and Main functions

- Example of sharing display-output between Safety RTOS and Linux IVI/Cluster
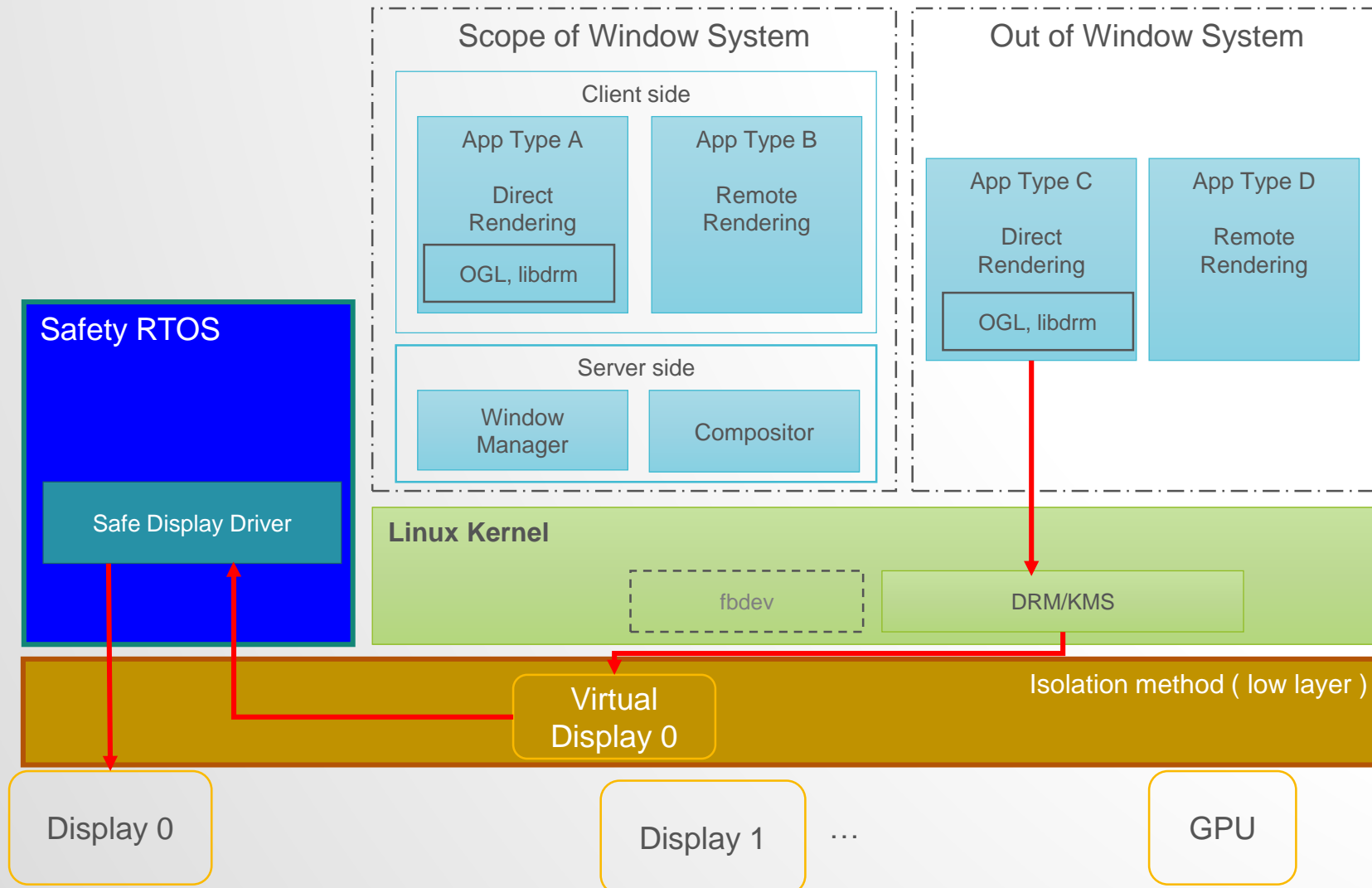
Topics

- Architecture overview

- Graphics
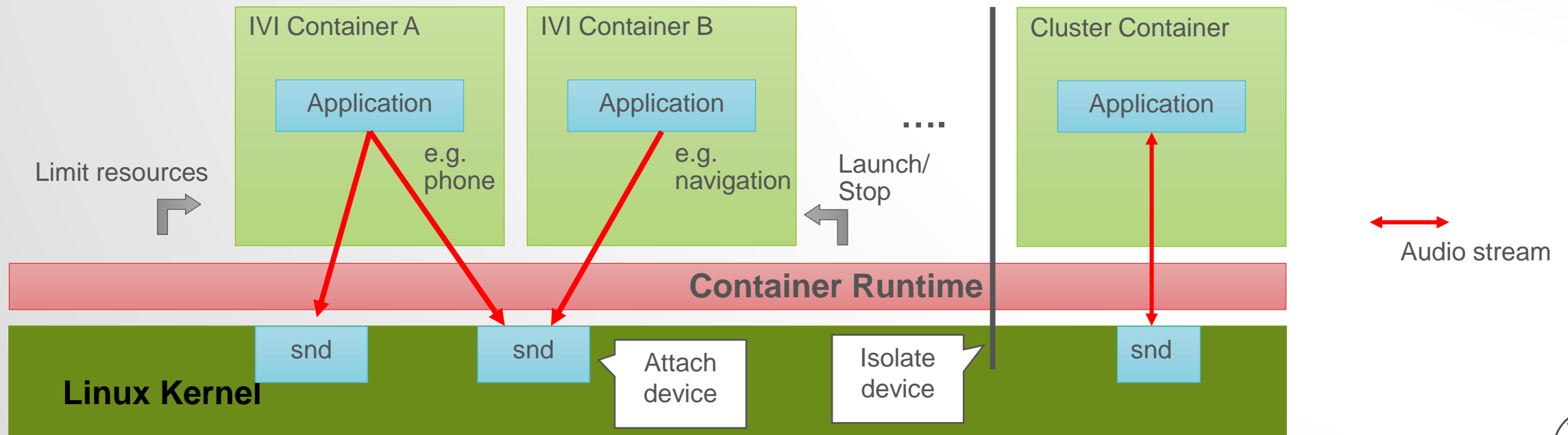
- **Sound**

- CAN

Requirement from sound usecase

Requirement of container runtime role from sound view point of view

Container resource management

- Approve container to look devices/files/sockets to output audio stream and realize isolation of device.

- Guarantee specific container can work even if IVI container consumes resource high

Container Lifecycle Management

- Launch/Stop containers dynamically (e.g. SDL container starts when device is plugged)

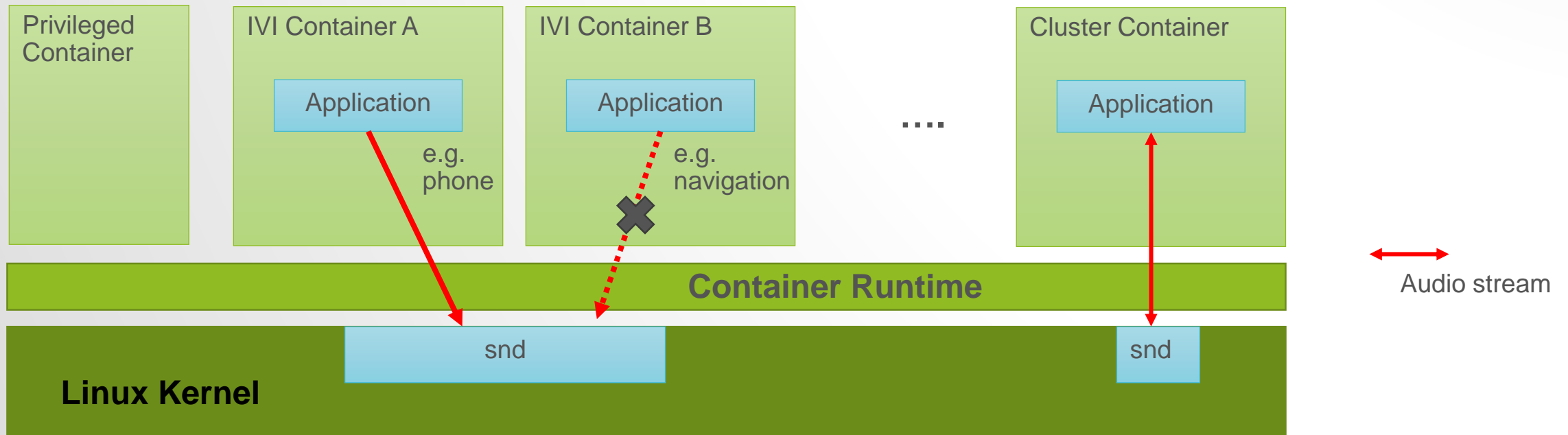# Issue of sound architecture in container

Isolate sound device

- For QM container, it is better to isolate from IVI containers to avoid conflict

- To prohibit other containers access to the device QM container access is good solution to cost down the verification.
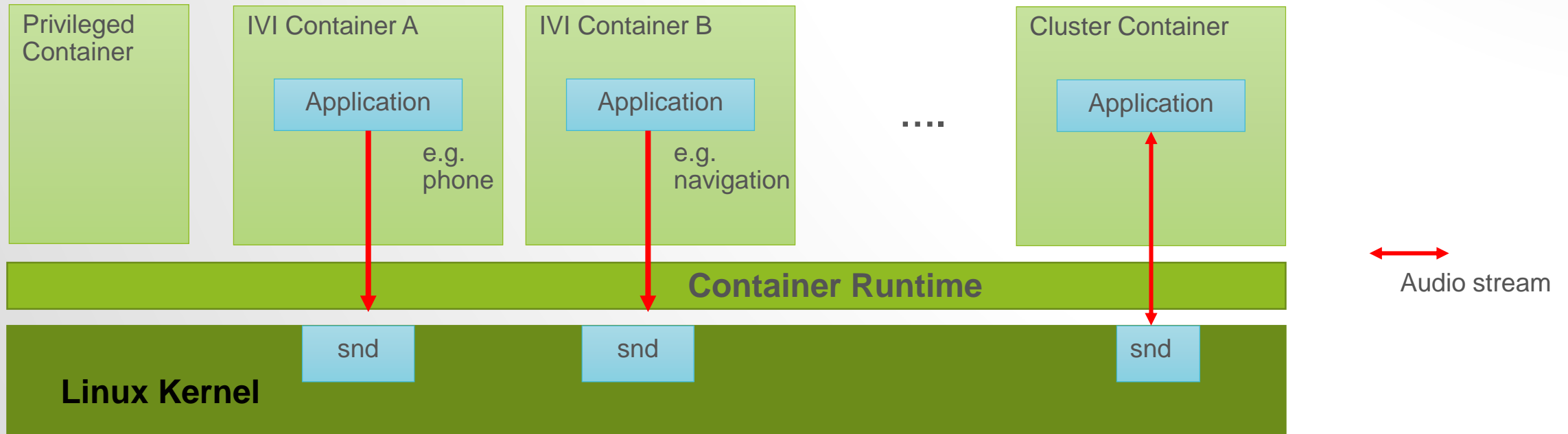
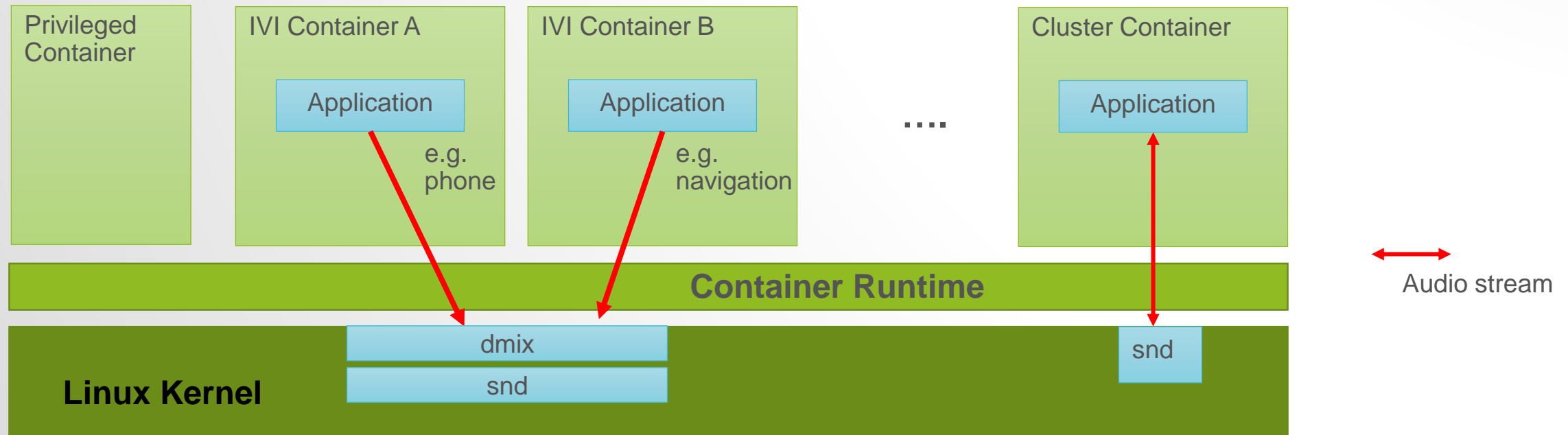# Issue of sound architecture in container

Sharing sound devices

- Container Runtime approves containers to access the ALSA sound devices.

- In the case that a sound device is approved to access to several containers

- But only one application can access the device, so can't manage(mixing, routing)

# Issue of sound architecture in container

Sharing sound devices

- ALSA dmix can provide mixing but it's too simple.

- This can't handle complex audio situations.

Sound architecture in container

Use Sound Server and provide sockets to the containers

- Typical Linux sound architecture

- Container Runtime approves the sockets for IVI containers sound server provides

- Location of sockets is defined in entire system (e.g. /run/cointainer/pulse).

- Sound server in a container collects the data then mix, cork, route and so on.
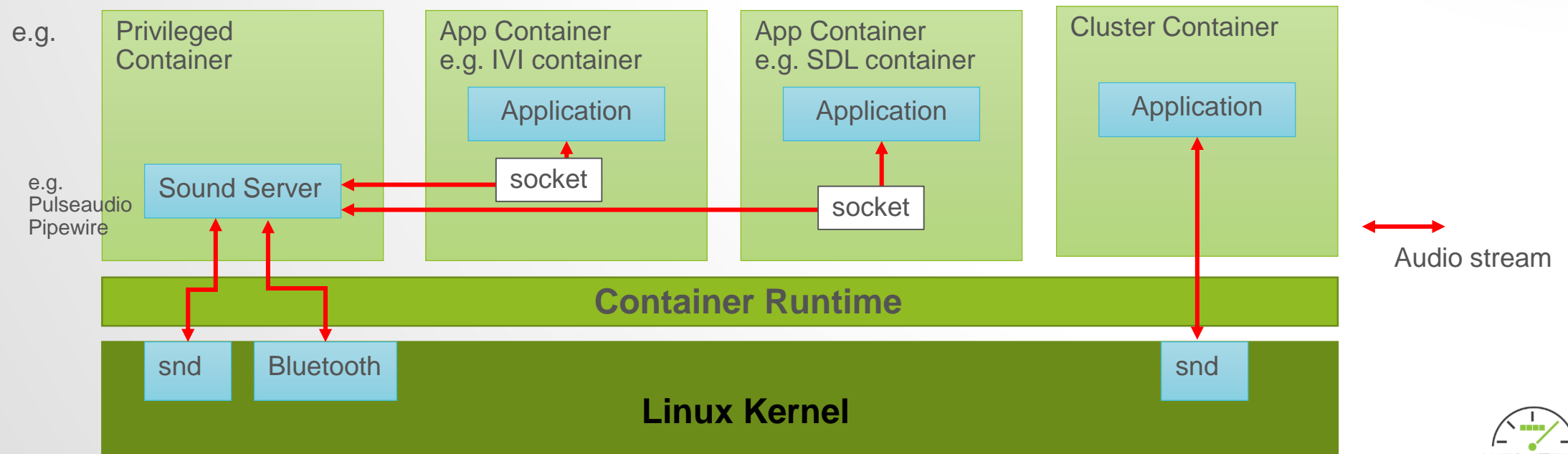
Conclusion - architecture overview sound

- Container runtime approves sound device to appropriate container

- Other containers can't access to sound device

- Sound server controls audio streams from other containers depending on situations

- Other container can find socket to sound server if Container Runtime approves
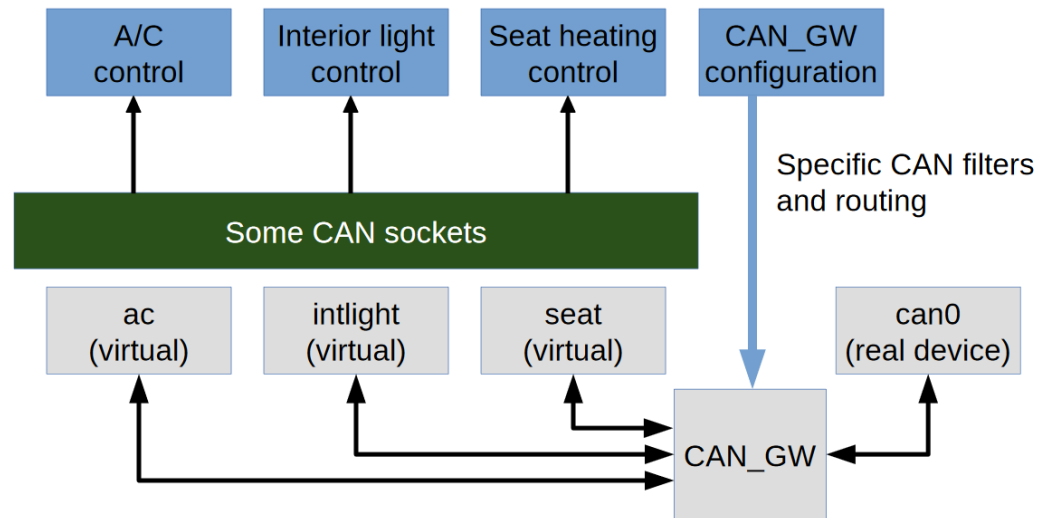
Topics

- Architecture overview

- Graphics

- Sound

- **CAN**
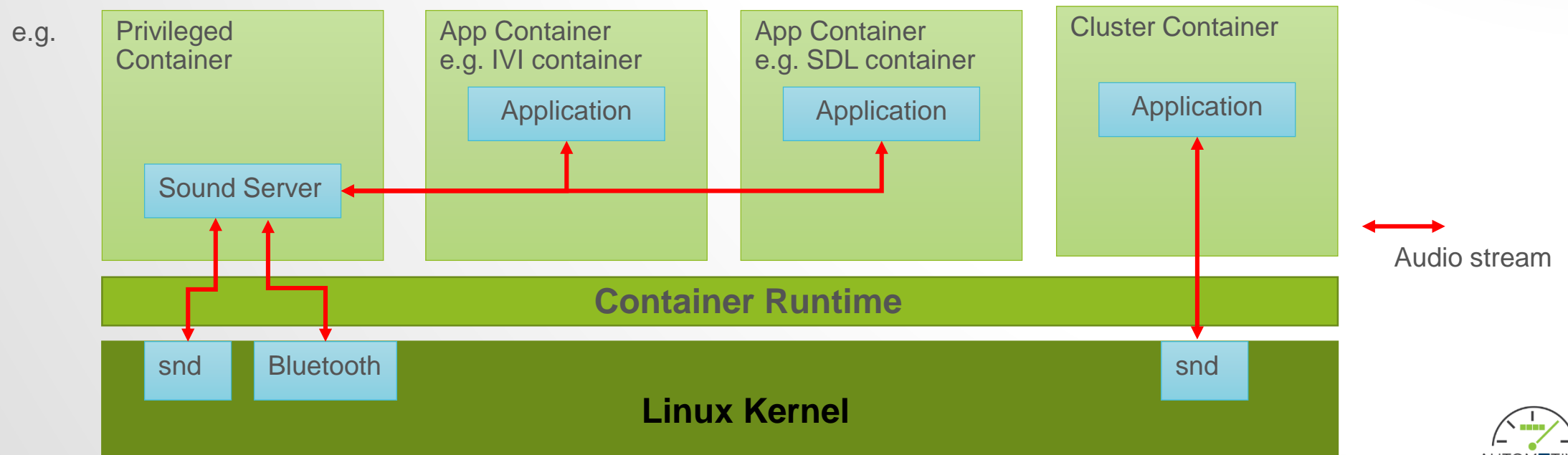
Architecture overview CAN

- Current Linux include various CAN network support.
  - https://wiki.automotivelinux.org/_media/agl-distro/agl2018-socketcan.pdf
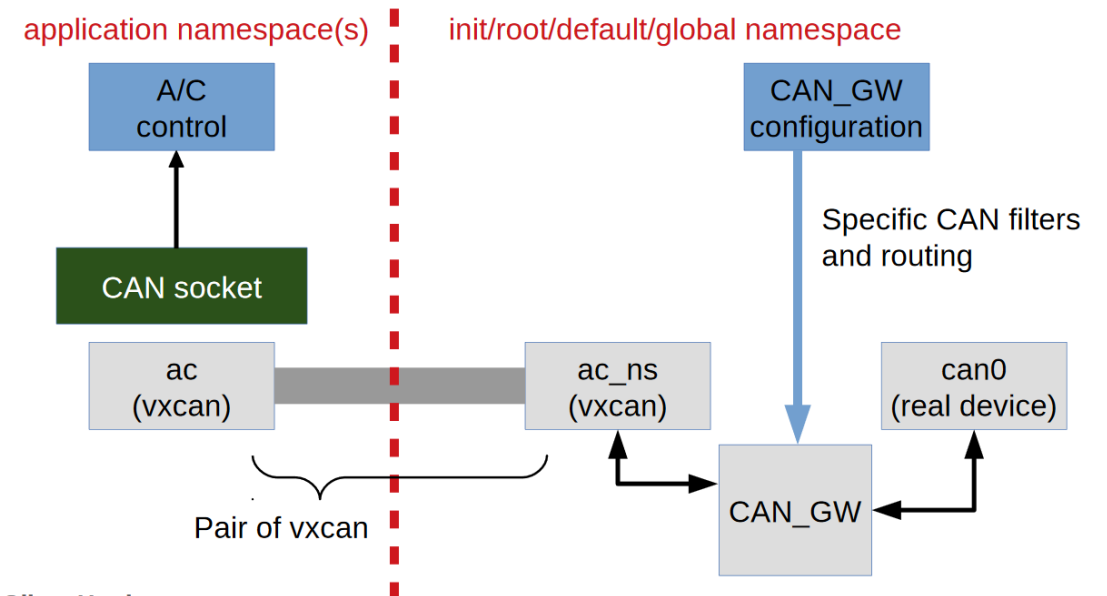 - Very good solutions!

Architecture overview CAN

- GAN_GW use for routing CAN data between the container.
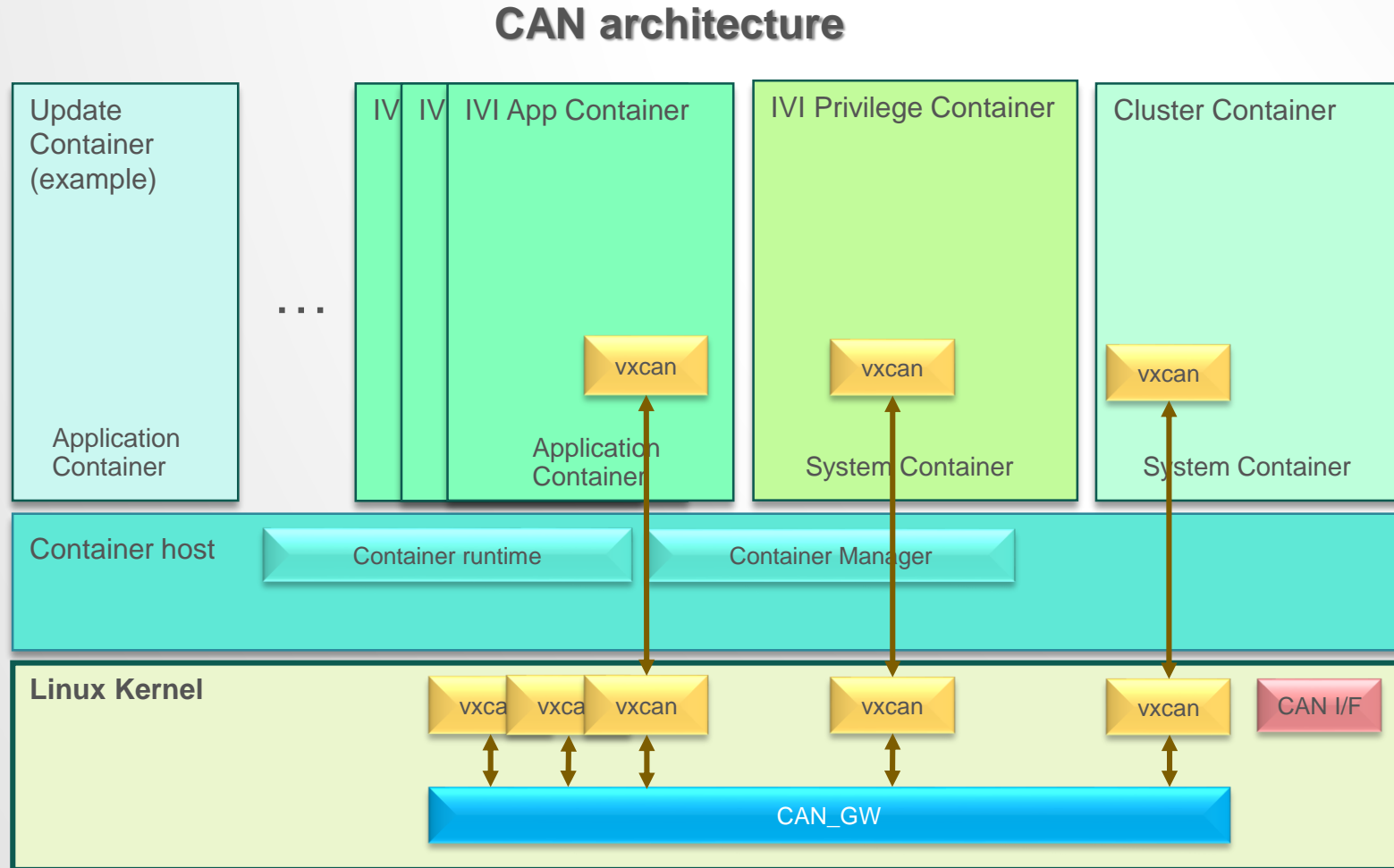
CAN architecture

How to connect CAN Bus

- How to connect CAN Bus?

CAN architecture

CAN Proxy transfers data acquired from CAN Unit to a virtual bus in Linux.
Cluster service and application should receive data directly from CAN Proxy to reduce latency.

Update Container (example)

Application Container

IVI App Container

IV | IV

vxcan

Application Container

IVI Privilege Container

vxcan

System Container

Cluster Container

Cluster App

Cluster Service

CAN Proxy

vxcan

System Container

...

Container host

Container runtime

Container Manager

Linux Kernel

vxca | vxca | vxcan

vxcan

vxcan

SPI/ SHM

CAN Communicator
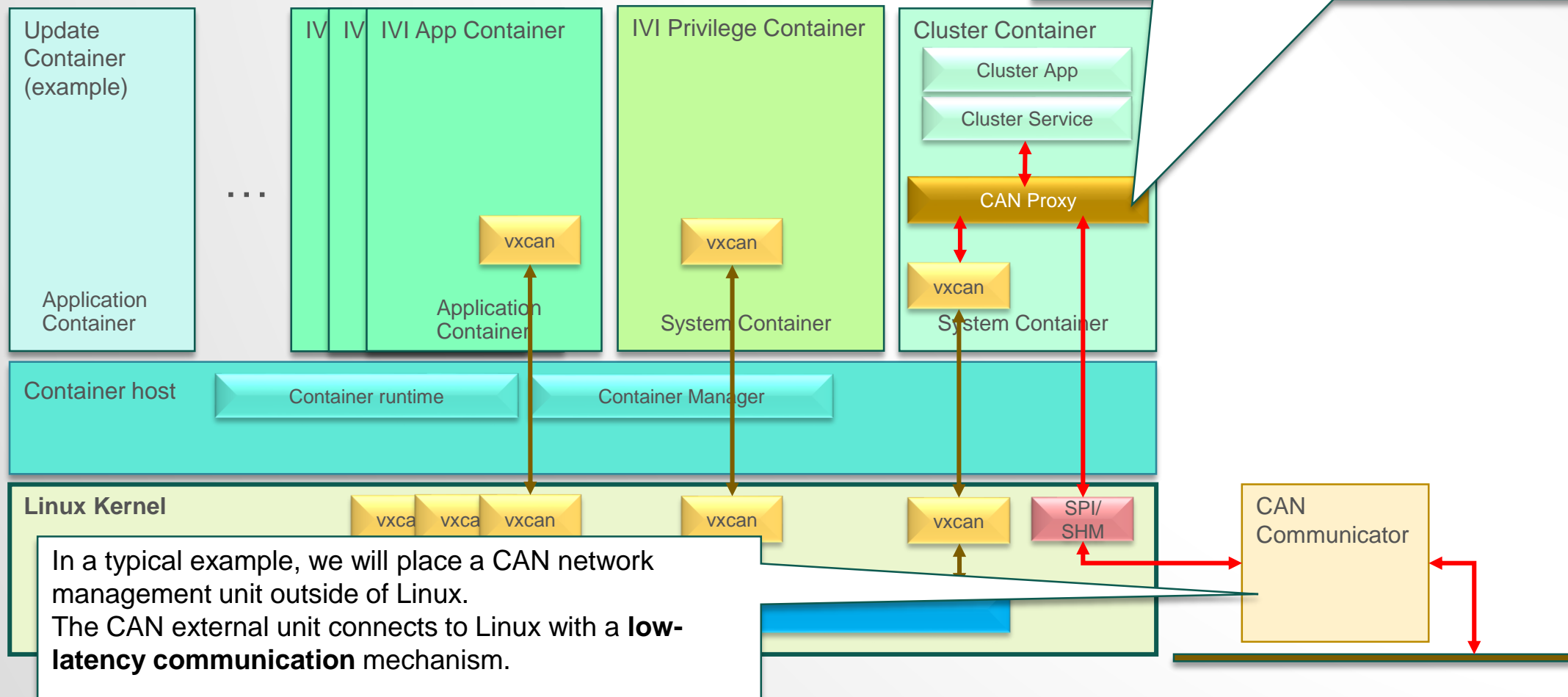
In a typical example, we will place a CAN network management unit outside of Linux.
The CAN external unit connects to Linux with a **low-latency communication** mechanism.

CAN Bus

AUTOMOTIVE GRADE **LINUX**

How to connect CAN Bus

- How to connect CAN Bus?
  - Another case

CAN architecture

In this case, CAN Proxy exists only to achieve common API.

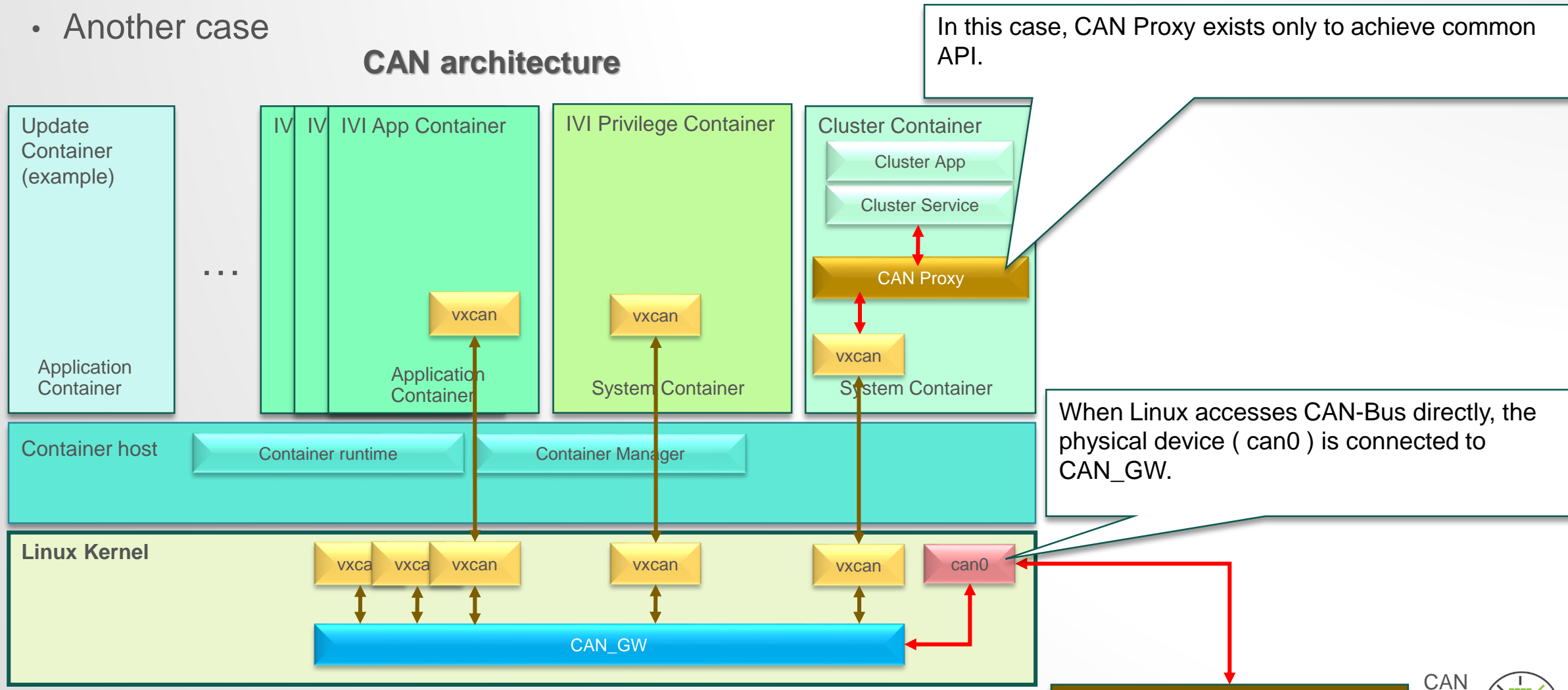| Update Container (example)	...	IV	IV	IVI App Container	IVI Privilege Container	Cluster Container

Cluster App

Cluster Service

CAN Proxy

vxcan

vxcan

vxcan

Application Container

Application Container

System Container

System Container

Container host — Container runtime — Container Manager

When Linux accesses CAN-Bus directly, the physical device (can0) is connected to CAN_GW.

Linux Kernel

vxca | vxca | vxcan | vxcan | vxcan | can0

CAN_GW

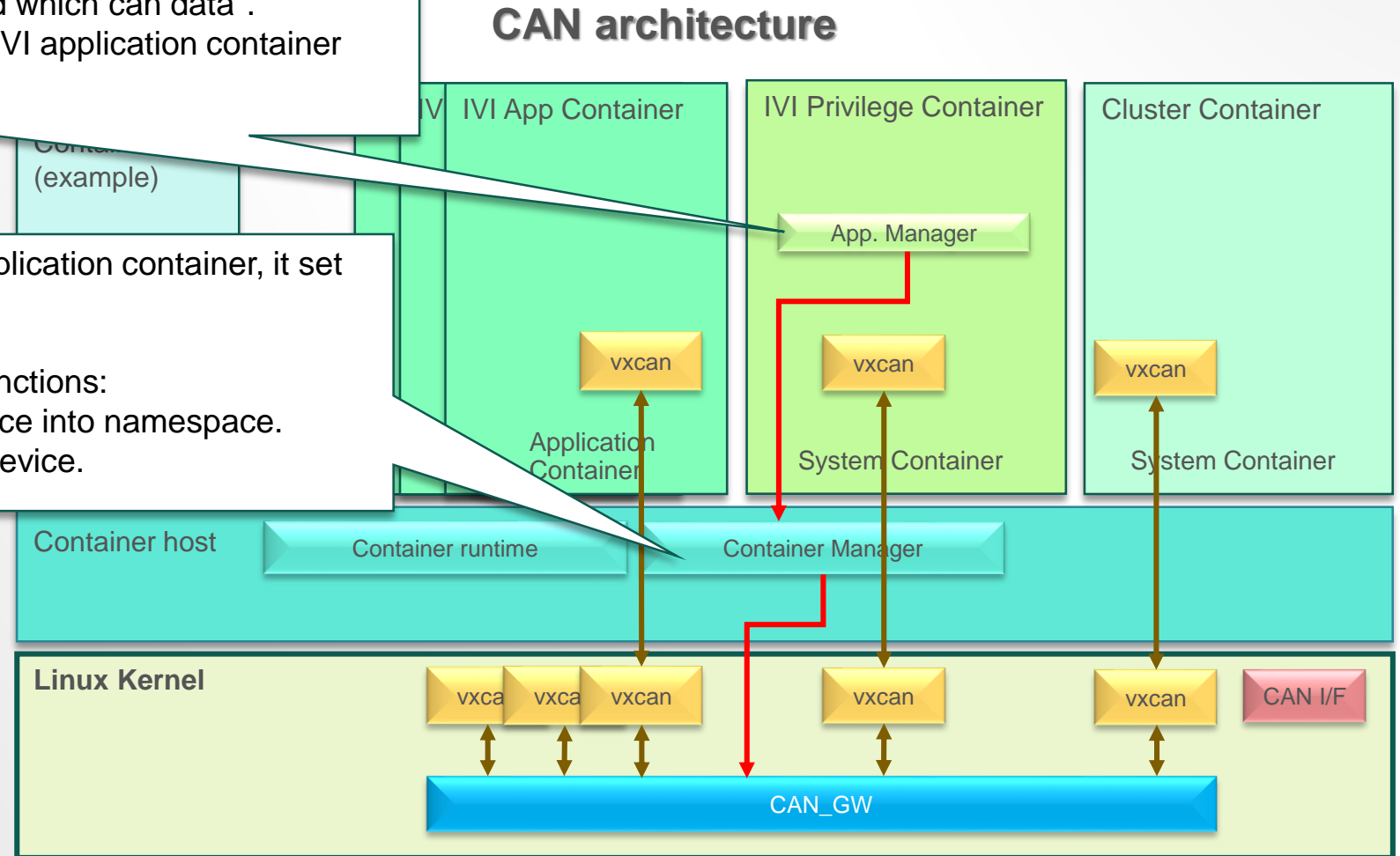CAN Bus

How to configure CAN_GW

- How to configure CAN_GW

IVI application manager know "who need which can data".
IVI application manager request launch IVI application container with CAN configuration.

When container manager launch to IVI application container, it set up CAN_GW at the same time.

Container manager needs the following functions:
- Create vxcan device pair and one device into namespace.
- Set up CAN_GW for host side vxcan device.

CAN architecture

Container (example)

IVI App Container

IVI Privilege Container

Cluster Container

App. Manager

vxcan

vxcan

vxcan

Application Container

System Container

System Container

Container host

Container runtime

Container Manager

Linux Kernel

vxca

vxca

vxcan

vxcan

vxcan

CAN I/F

CAN_GW

How to abstract CAN data
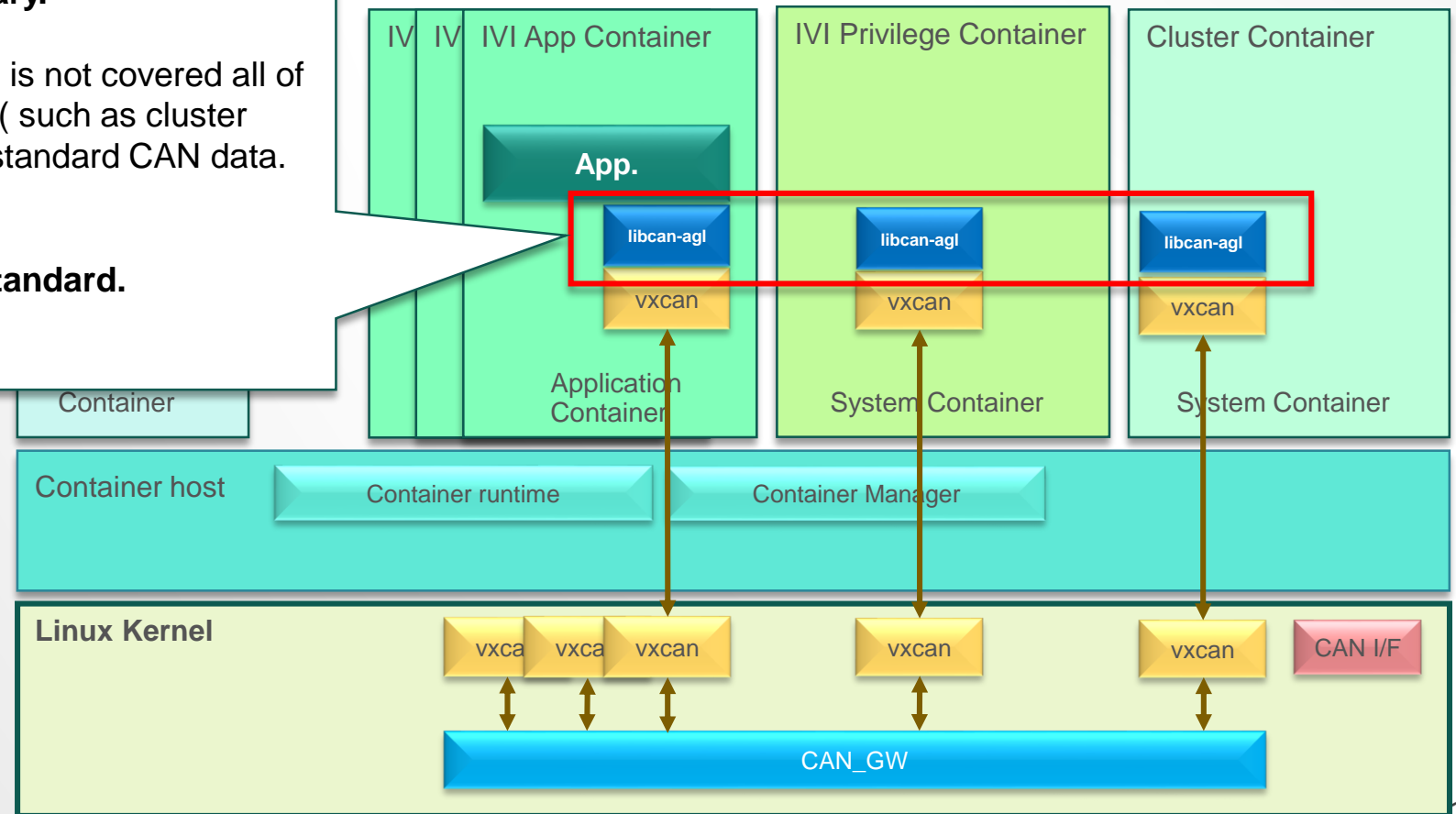
- How to abstract CAN data.

CAN data is abstracted by a user space library.

CAN data standard (sucn as OpenXC, ODBII) is not covered all of CAN data. On the other hand, some functions (such as cluster service and diagnostic functions) require non-standard CAN data.

Virtual CAN bus should not use CAN data standard. Convert by user space library.

CAN architecture

How to abstract CAN data

- How to abstract CAN data.
  - When AGL IVI into system container.

CAN architecture

If your system want to use AGL IVI.

Virtual CAN bus carry OEM specific data.
low-can-service convert from OEM specific data to own api.
It's same of current AGL IVI.

AGL IVI Container

AGL IVI App.

AGL IVI App.

low-can-binder

vxcan

System Container

IVI Privilege Container

libcan-agl

vxcan

System Container

Cluster Container

libcan-agl

vxcan

System Container

Update

Application Container

Container host

Container runtime

Container Manager

Linux Kernel

vxca   vxca   vxcan

vxcan

vxcan

CAN I/F

CAN_GW