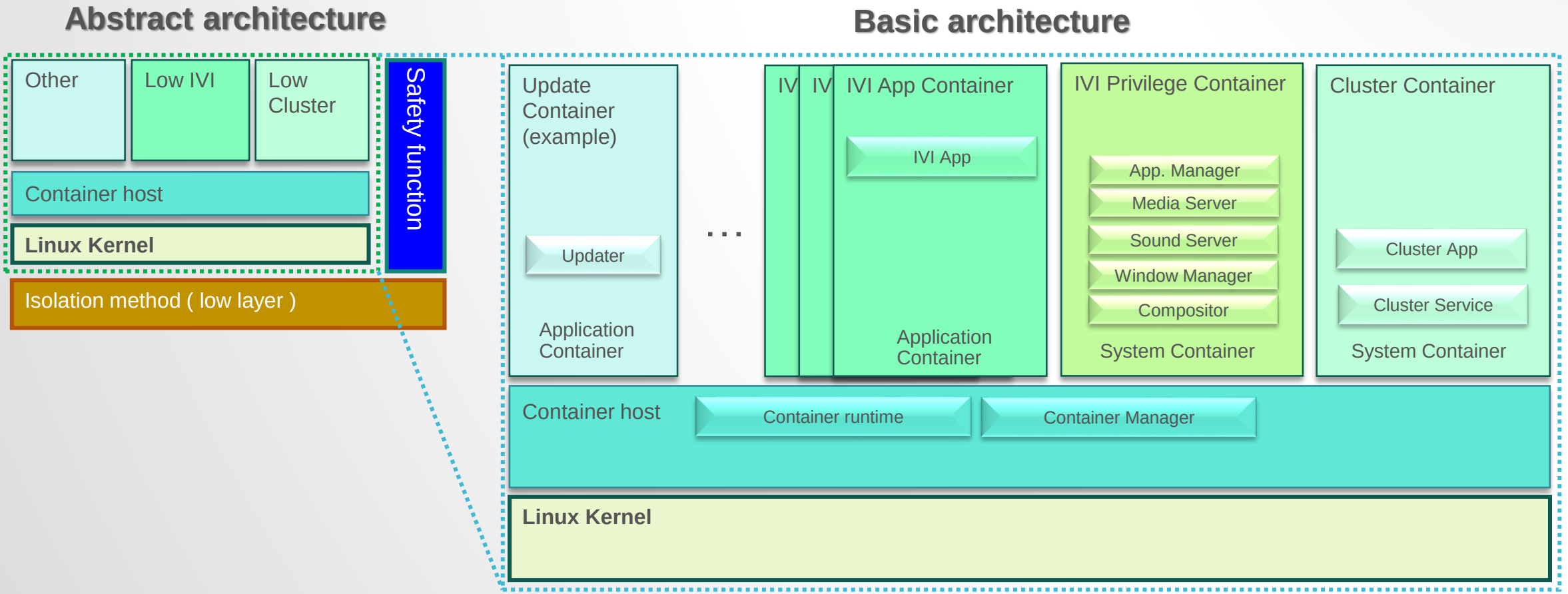
Topics

- **Architecture overview**
- Graphics
- Sound
- CAN

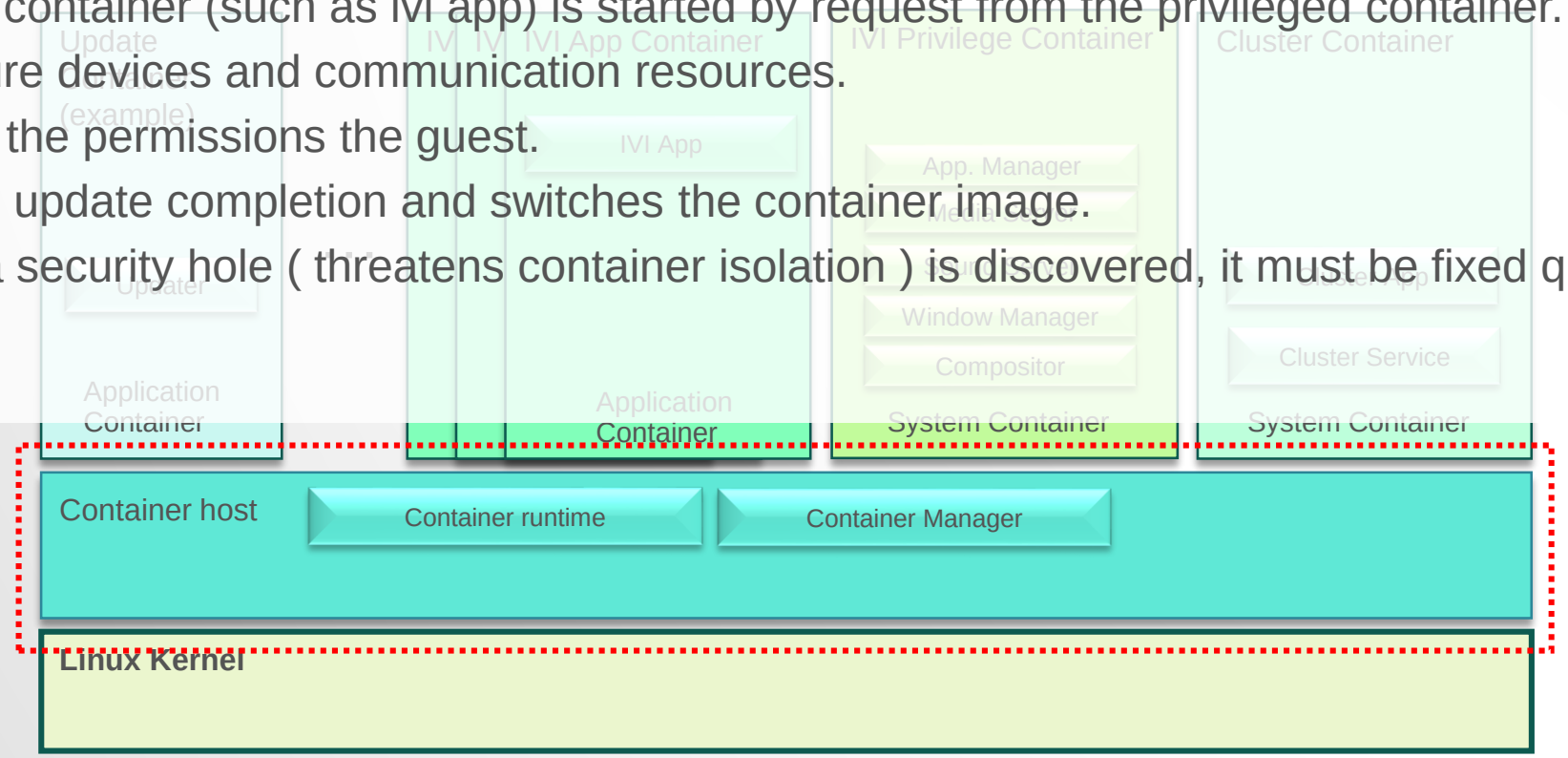
Container based architecture

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



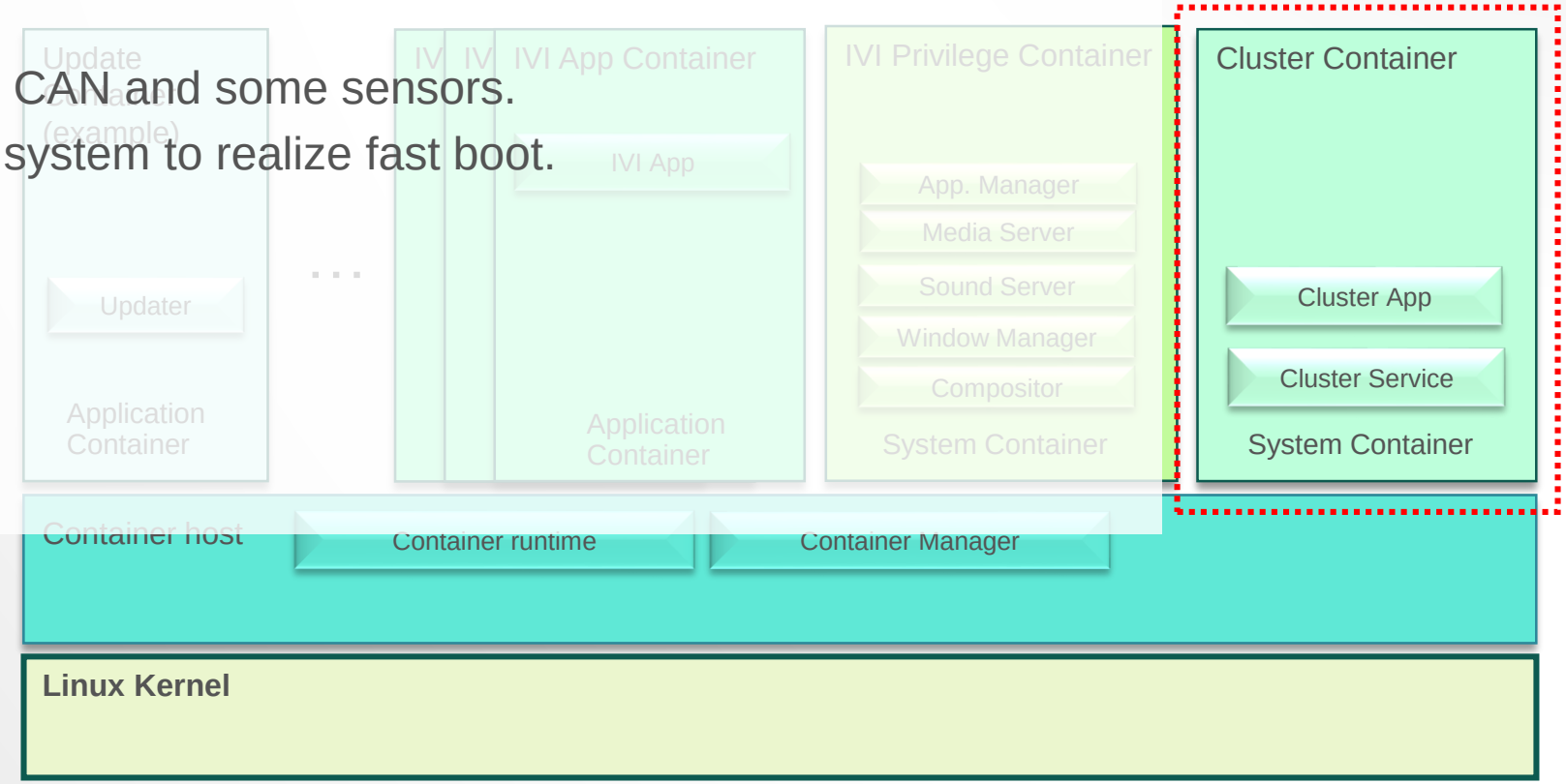
Automotive Use Case – Role of each container

- 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.



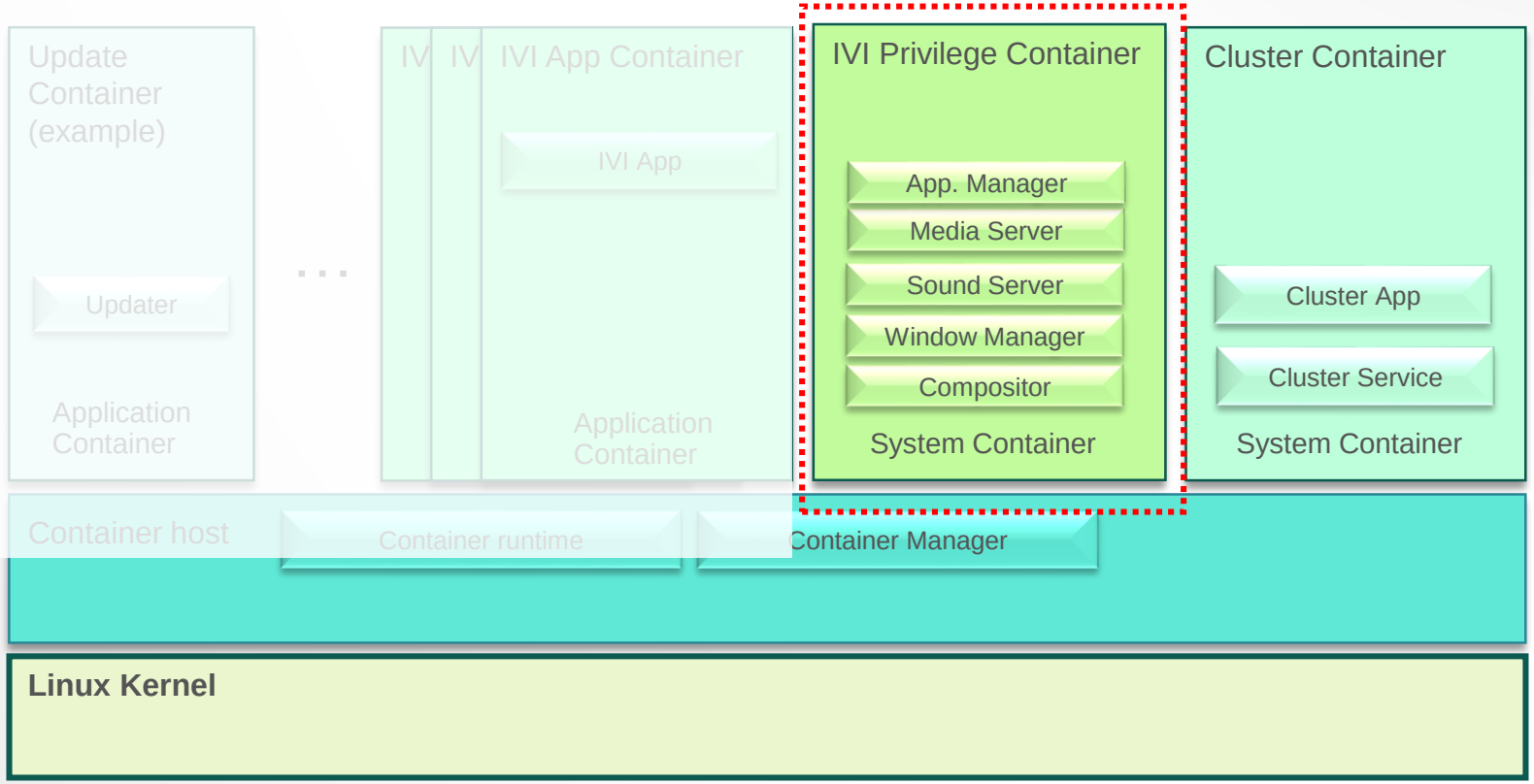
Automotive Use Case – Role of each container

- 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.

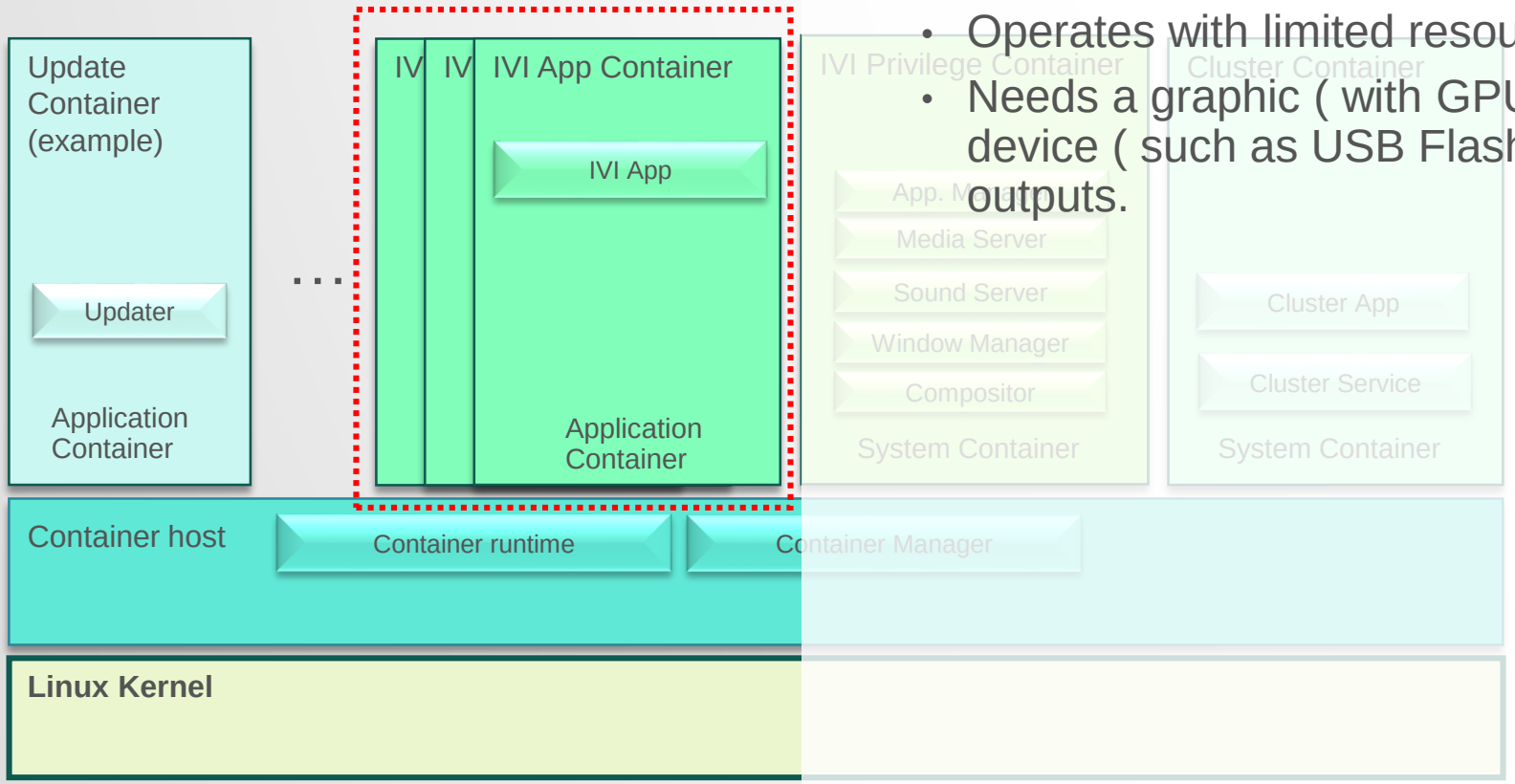


Automotive Use Case – Role of each container

- 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.

AUTOMOTIVE GRADE LINUX

Automotive Use Case – Many Issues

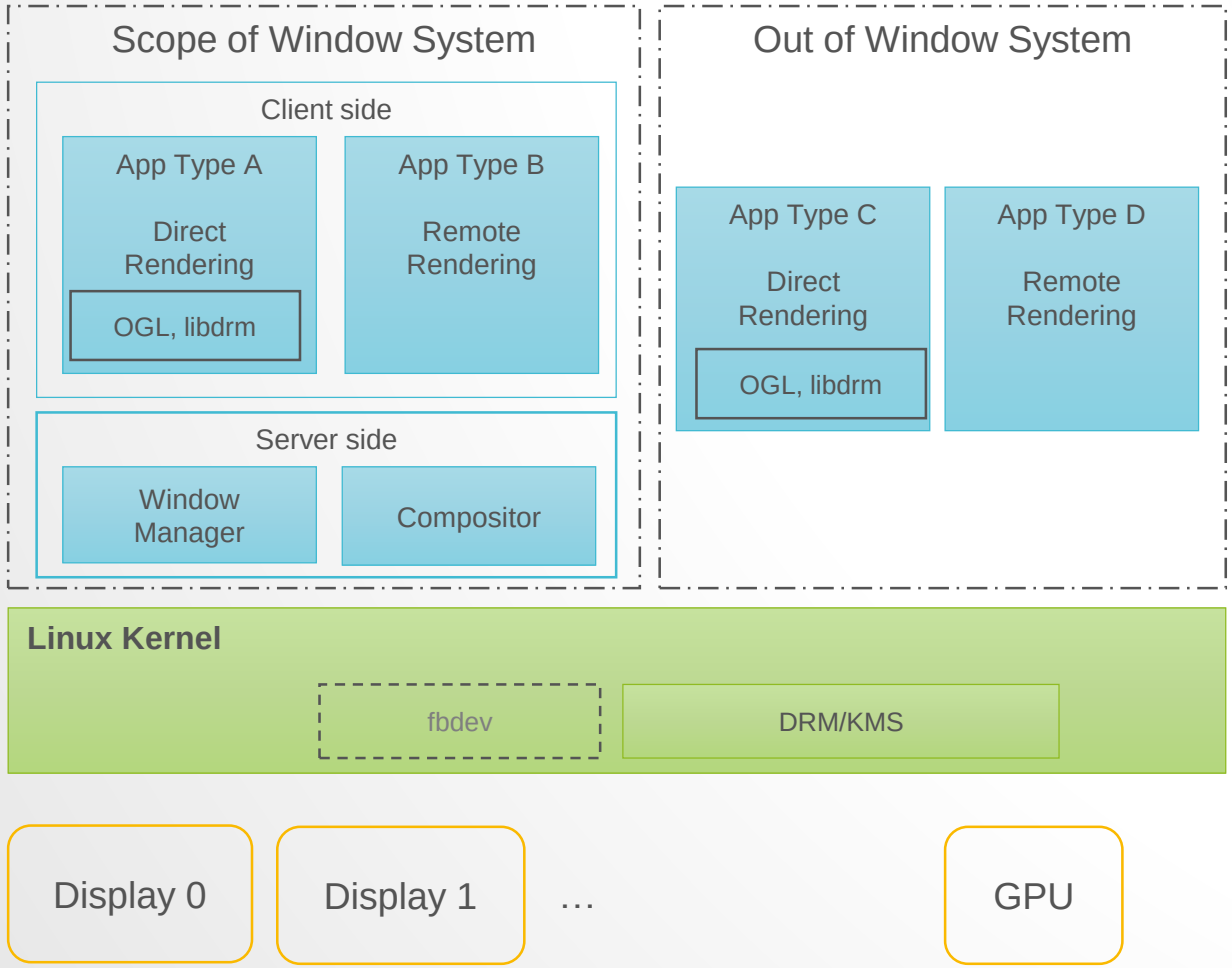
- Key Issue
  - Graphics Management
    - How to isolate and share the graphics stacks.
  - Sound Management
    - How to isolate and share the sound device.
  - CAN Network Management
    - How to deliver and hide the CAN data.
- Other Issue
  - Dynamic Device (USB, SD Card, etc.) Management
  - IP Network Management
  - Container Management and Update

Topics

- Architecture overview
- **Graphics**
- Sound
- CAN

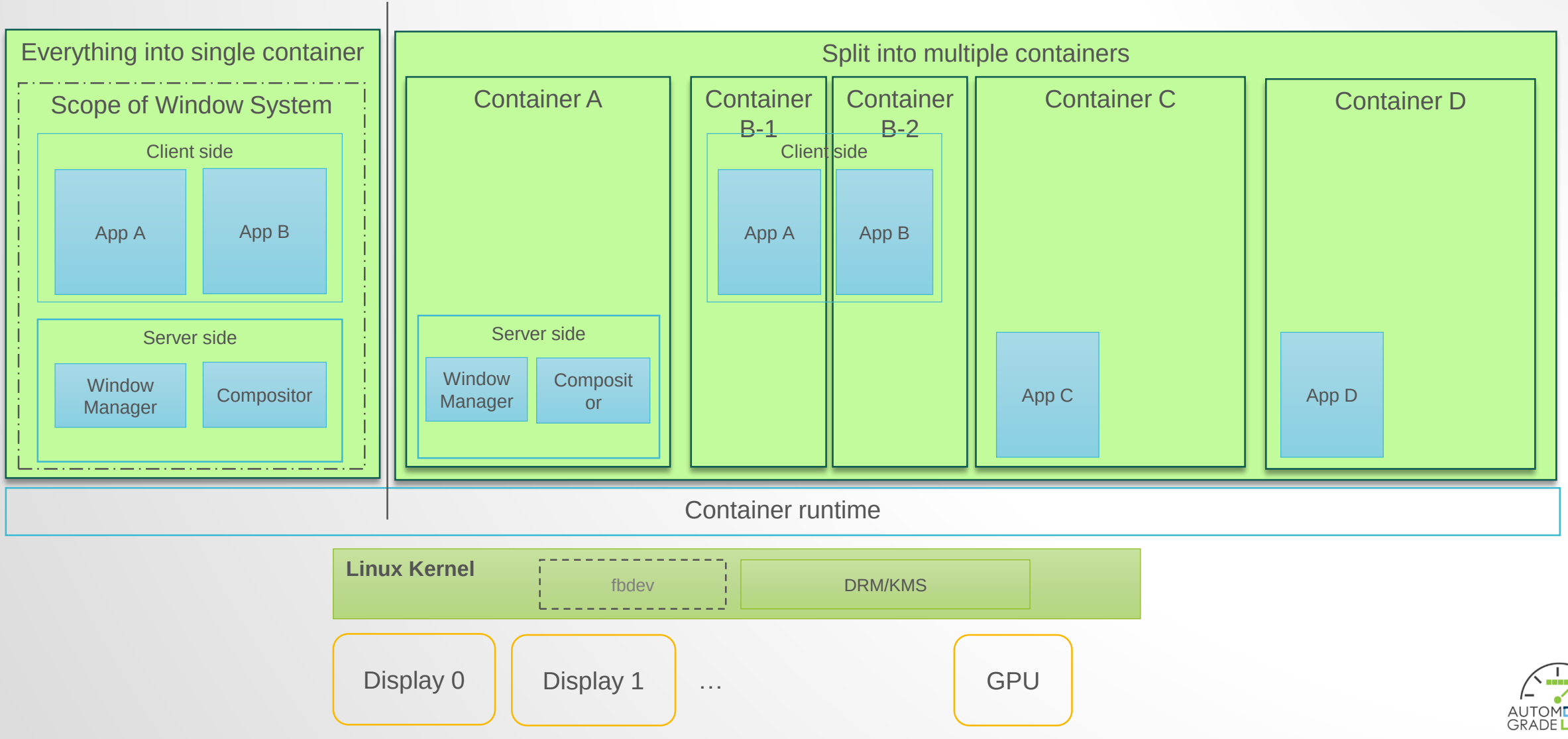
Architecture overview graphics

- Basic Linux graphics block diagram / types of GUI application



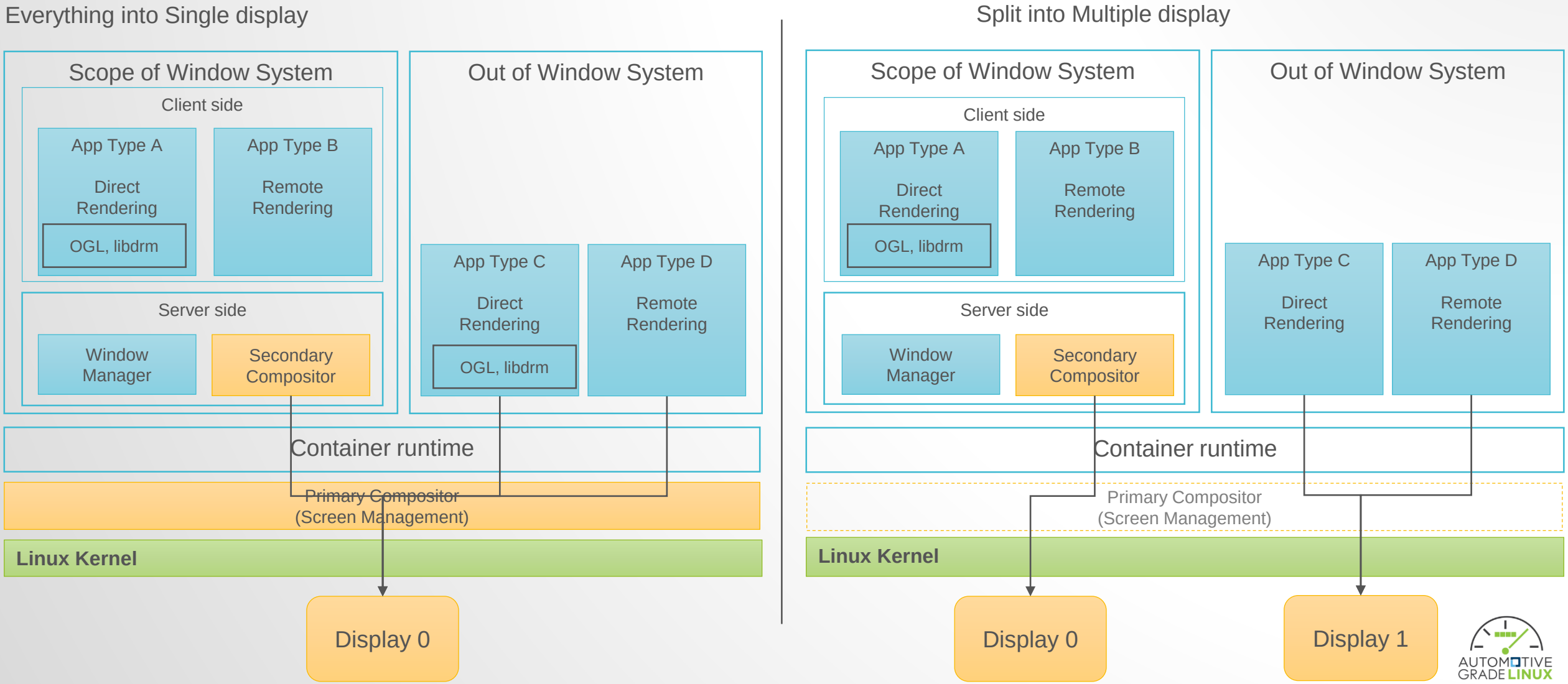
Architecture overview graphics

- Selections of Linux container integration for GUI apps



Architecture overview graphics

- Selections of Linux container integration for compositor



Topics

- Architecture overview
- **Graphics**
- Sound
- CAN

Requirement from sound usecase

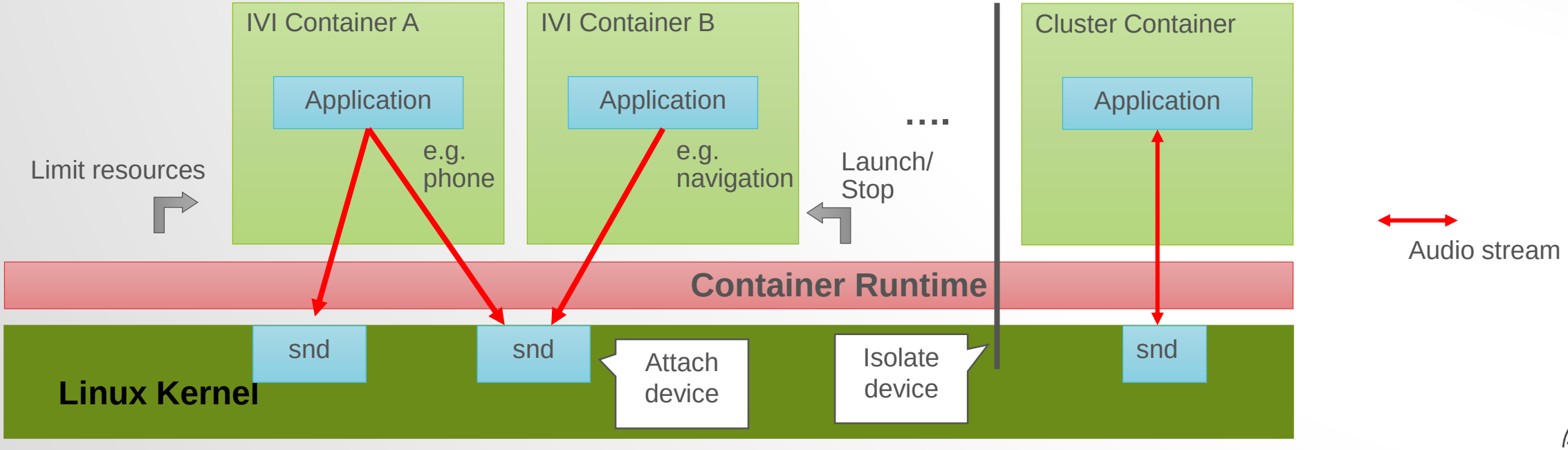
Requirement of container runtime role from sound view point of view

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

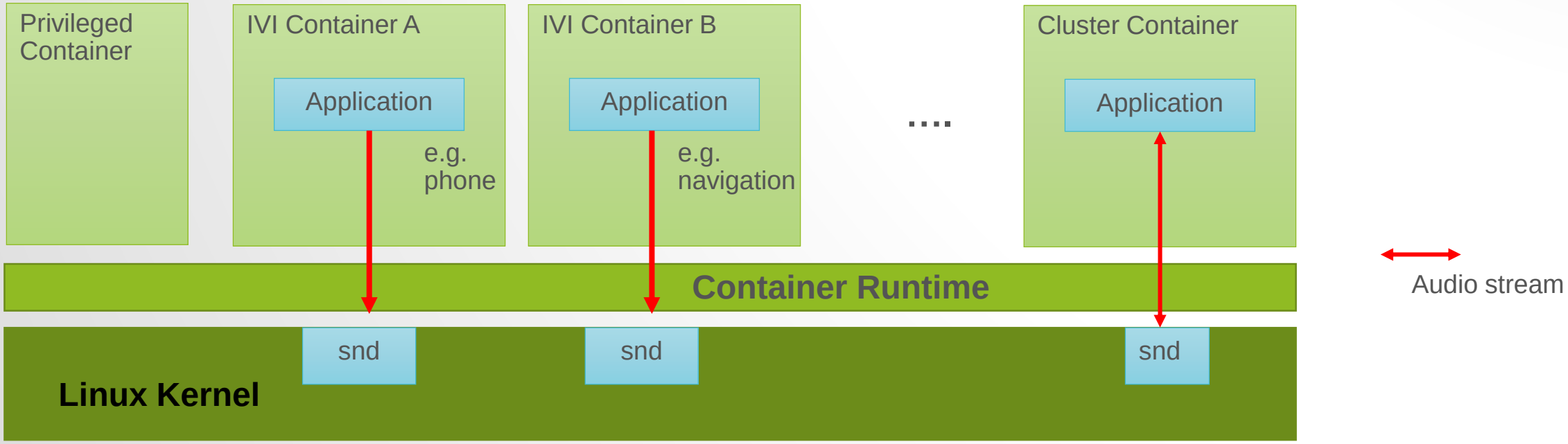
- Launch/Stop containers dynamically (e.g. SDL container starts when device is plugged)



Issue of sound architecture in container

Isolate sound device

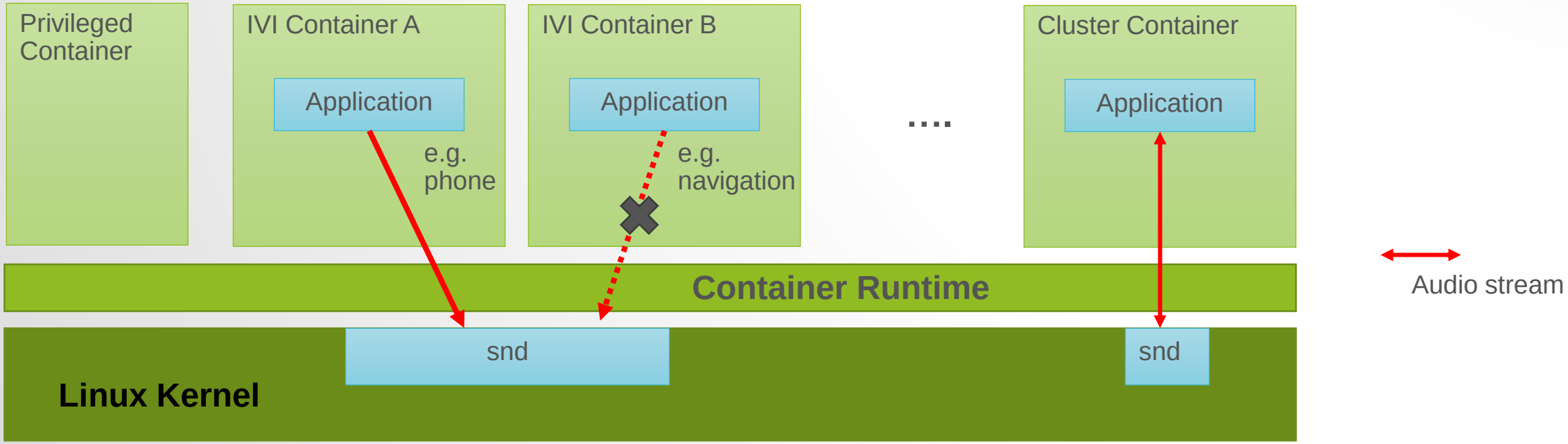
- For QM container, it is better to isolate from IVI containers to avoid conflict
- To prohibit other containers access to the device QM container access is good solution to cost down the verification.



Issue of sound architecture in container

Sharing sound devices

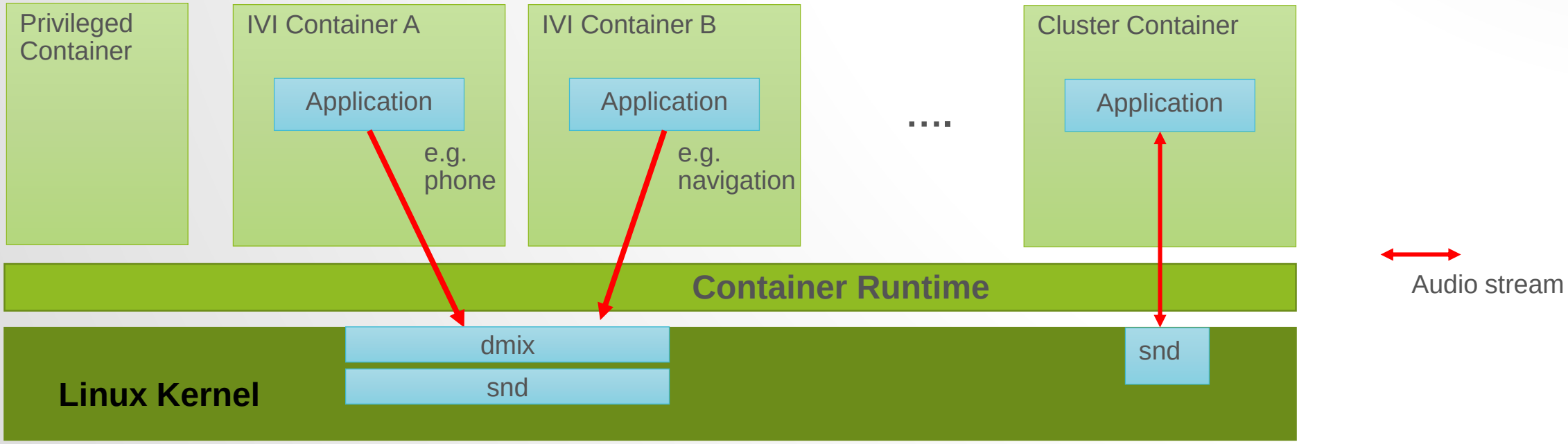
- Container Runtime approves containers to access the ALSA sound devices.
- In the case that a sound device is approved to access to several containers
- But only one application can access the device, so can't manage(mixing, routing)



Issue of sound architecture in container

Sharing sound devices

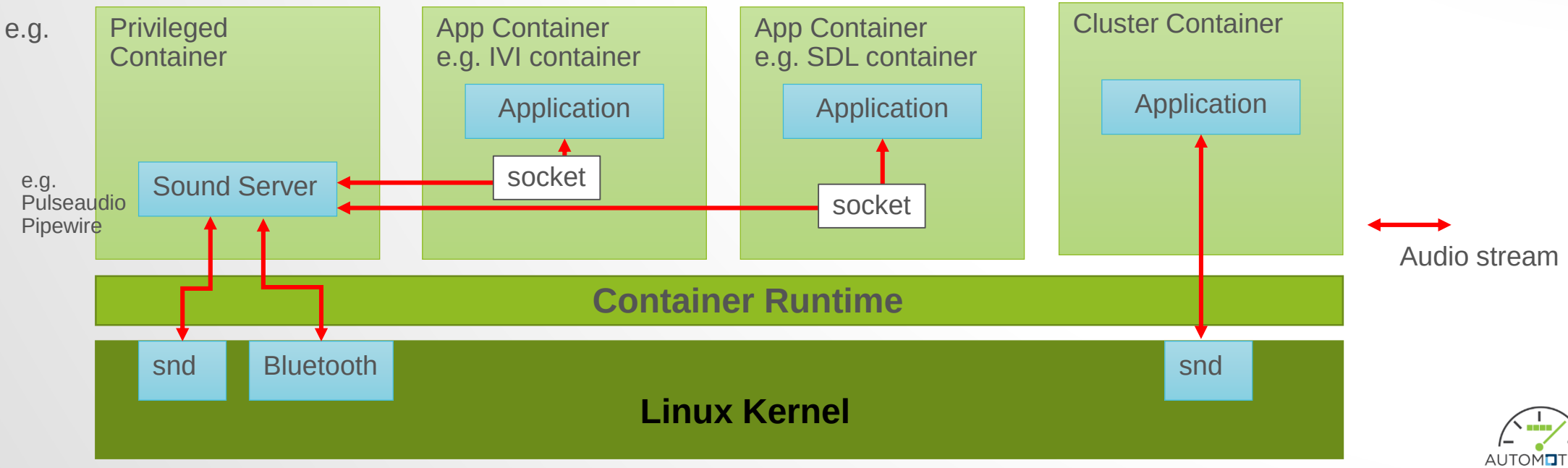
- ALSA dmix can provide mixing but it's too simple.
- This can't handle complex audio situations.



Sound architecture in container

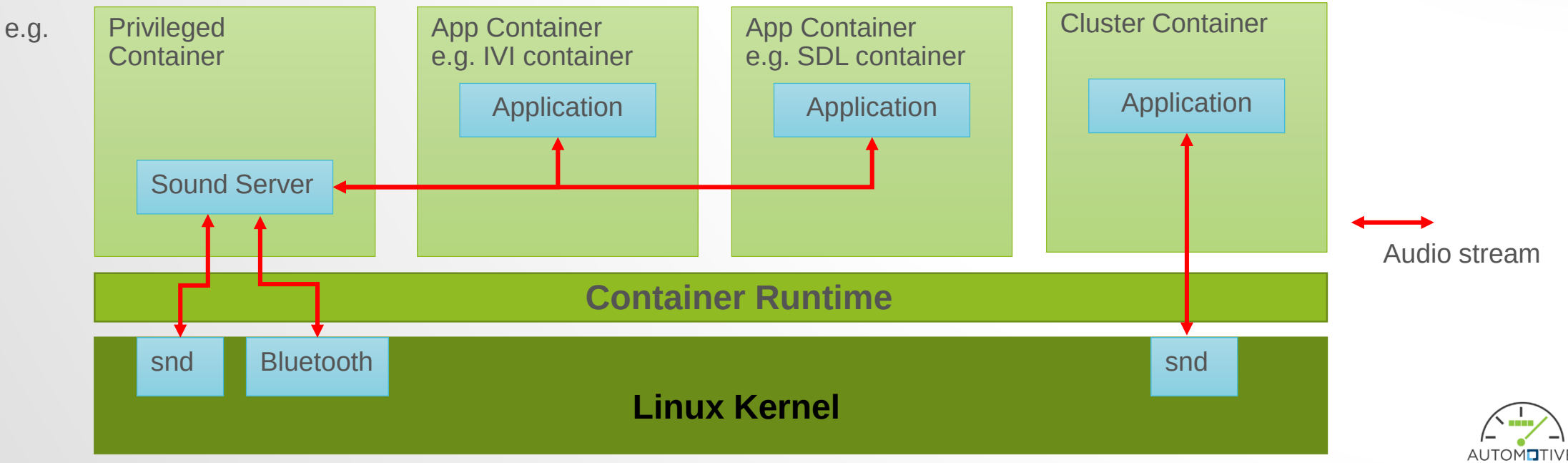
Use Sound Server and provide sockets to the containers

- 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

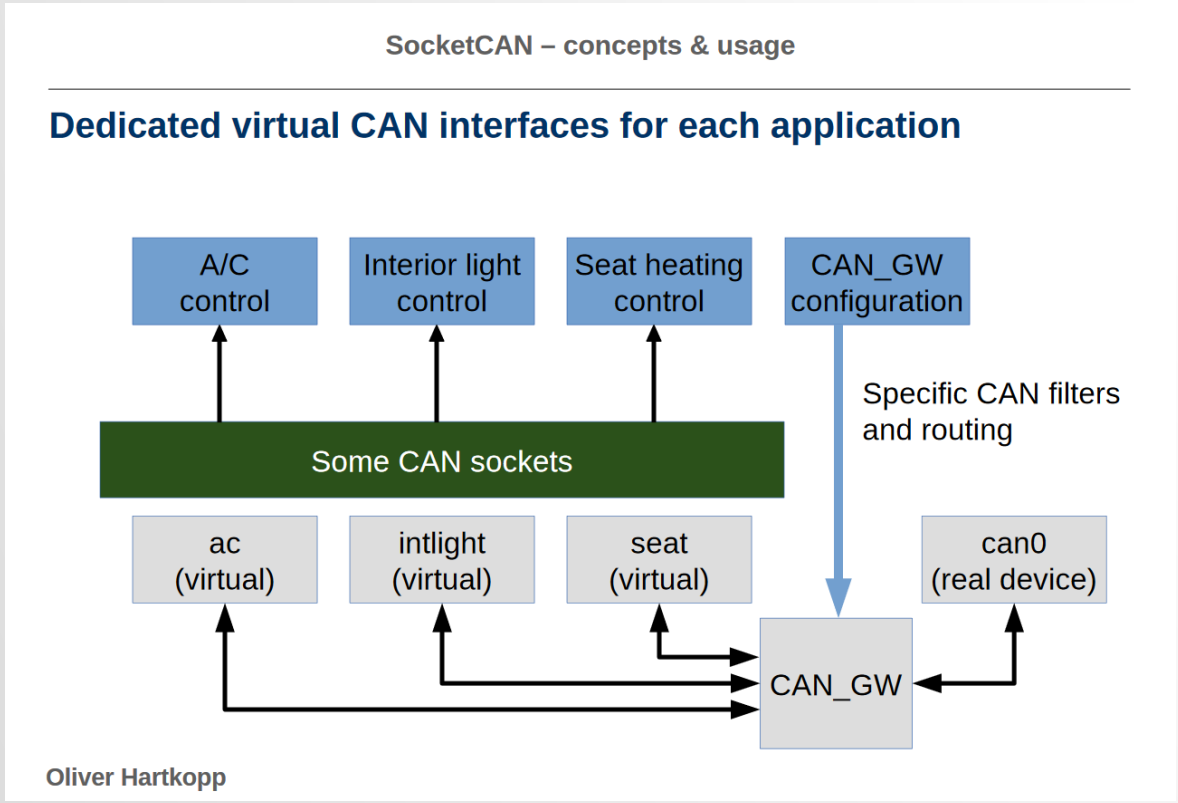


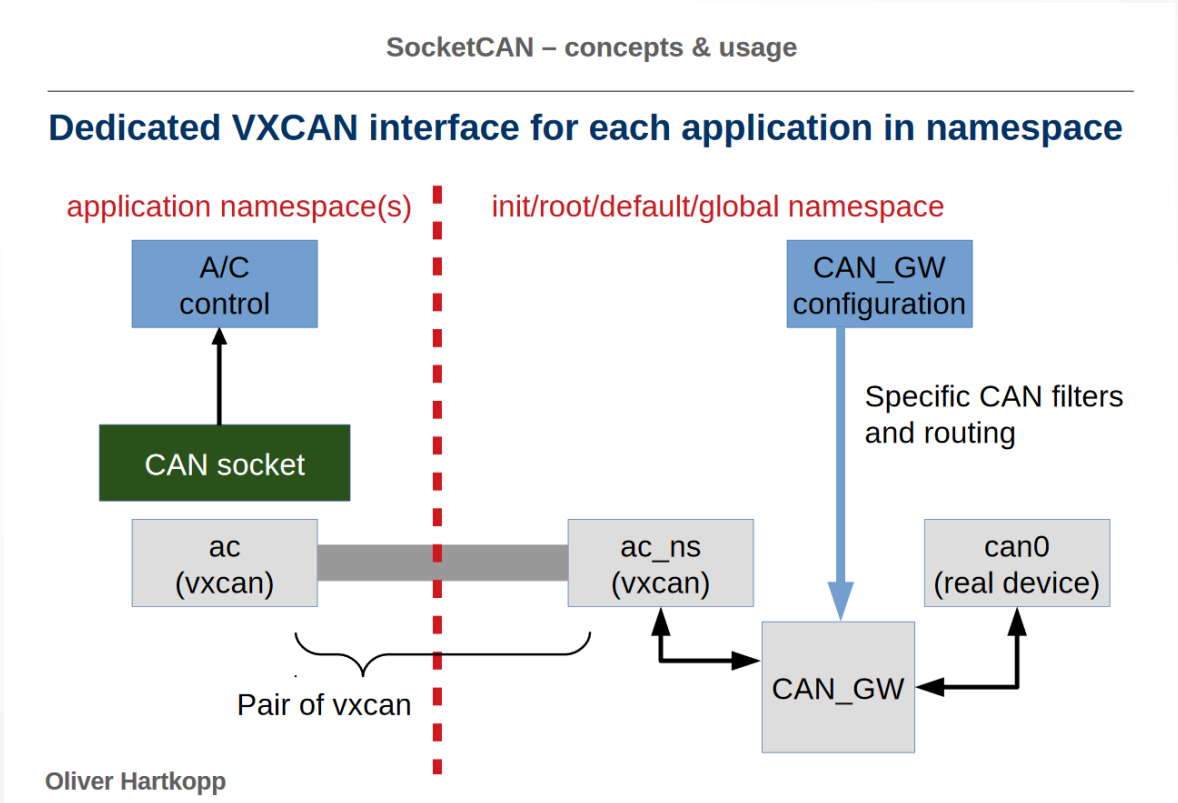
Topics

- Architecture overview
- Graphics
- Sound
- **CAN**

Architecture overview CAN

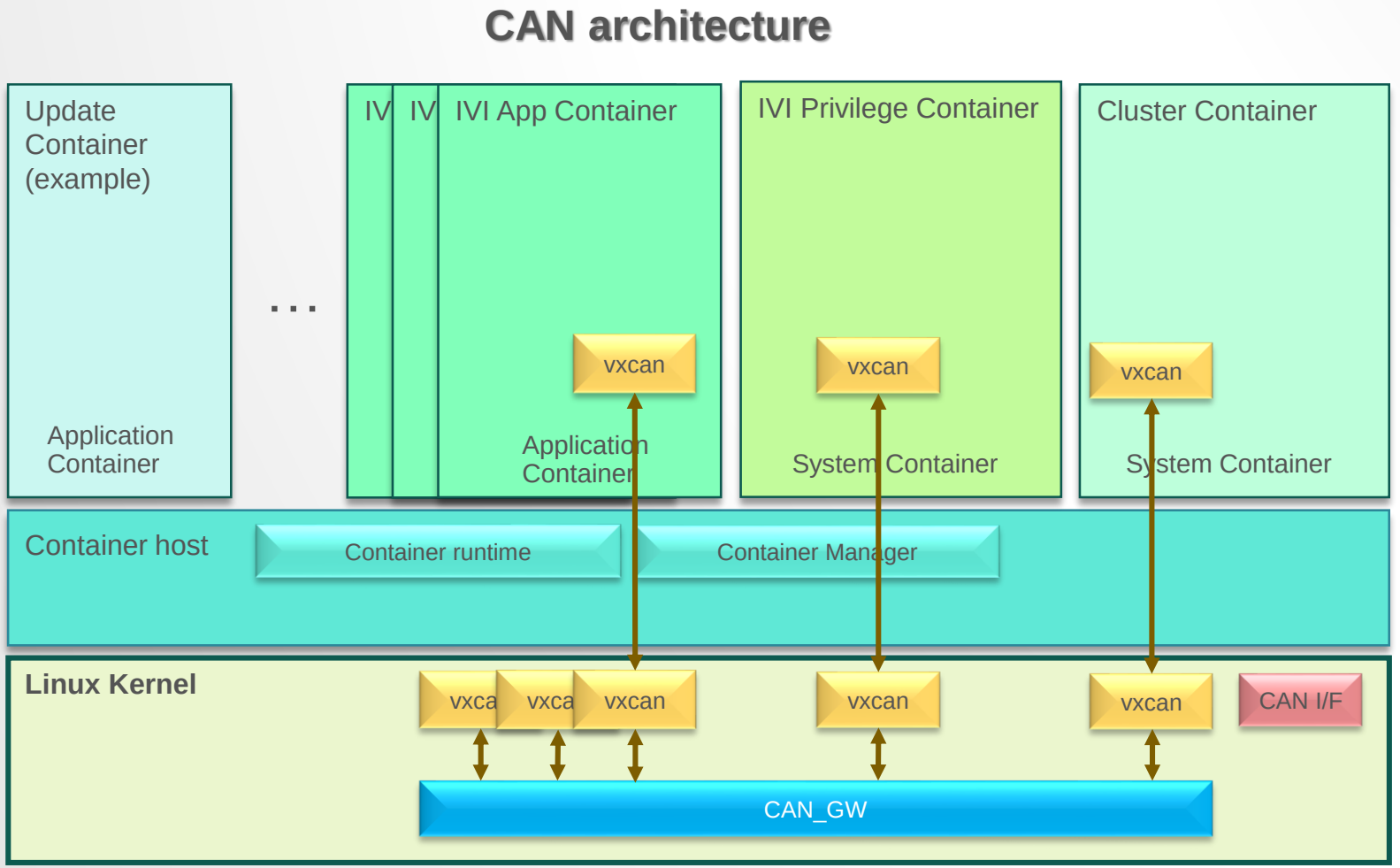
- Current Linux include various CAN network support.
  - https://wiki.automotivelinux.org/_media/agl-distro/agl2018-socketcan.pdf
  - Very good solutions!





Architecture overview CAN

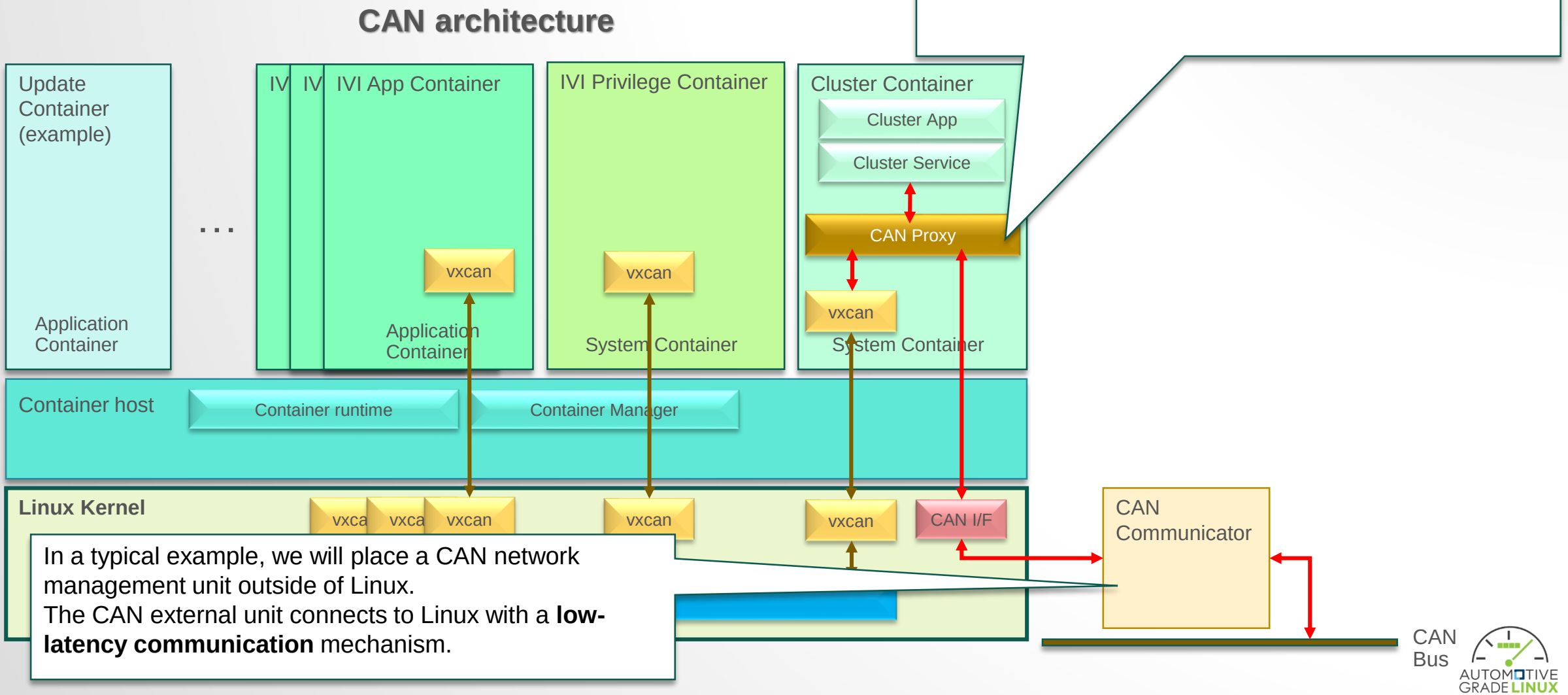
- GAN_GW use for routing CAN data between the container.



How to connect CAN Bus

- How to connect CAN Bus?

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.

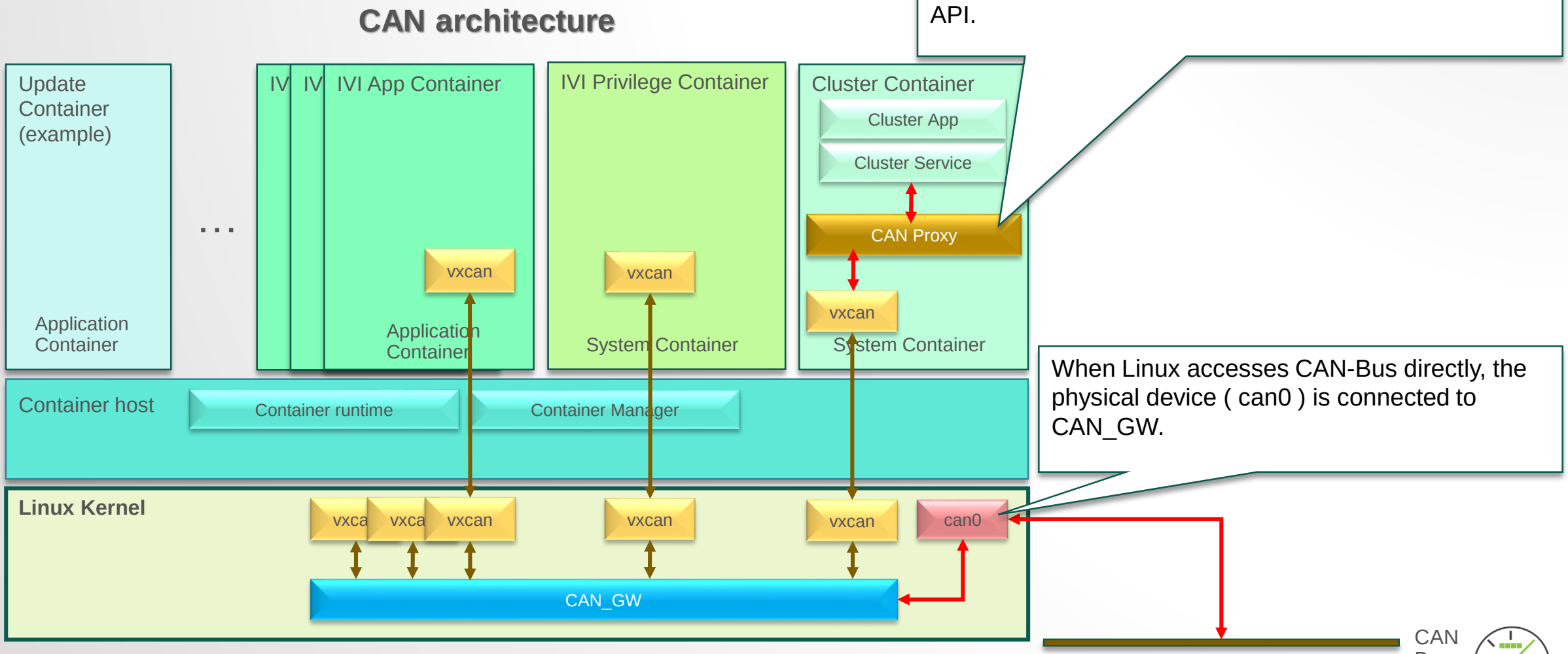


In a typical example, we will place a CAN network management unit outside of Linux.
The CAN external unit connects to Linux with a **low-latency communication** mechanism.

AUTOMOTIVE GRADE LINUX

How to connect CAN Bus

- How to connect CAN Bus?
  - Another case



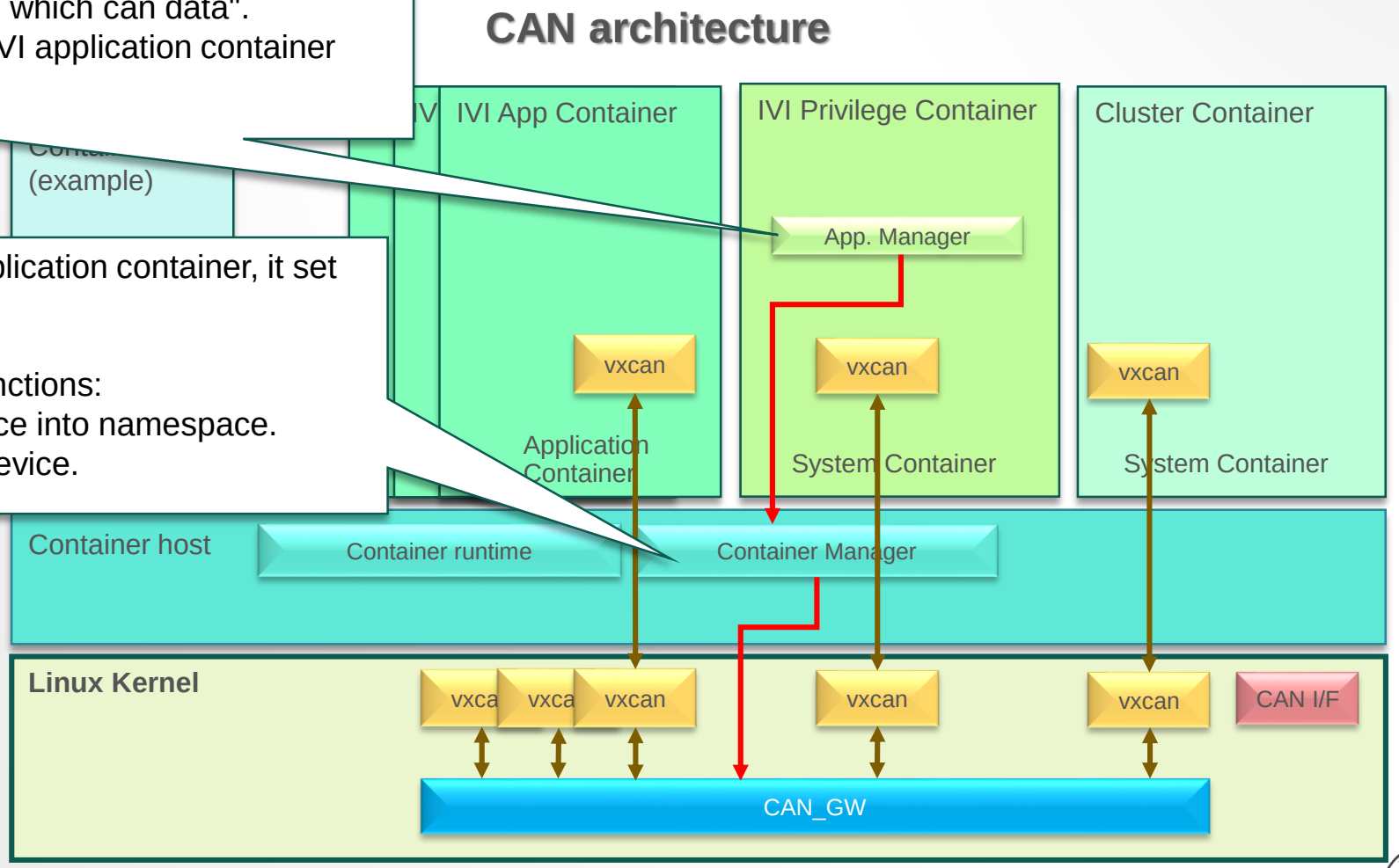
How to configure CAN_GW

- 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.



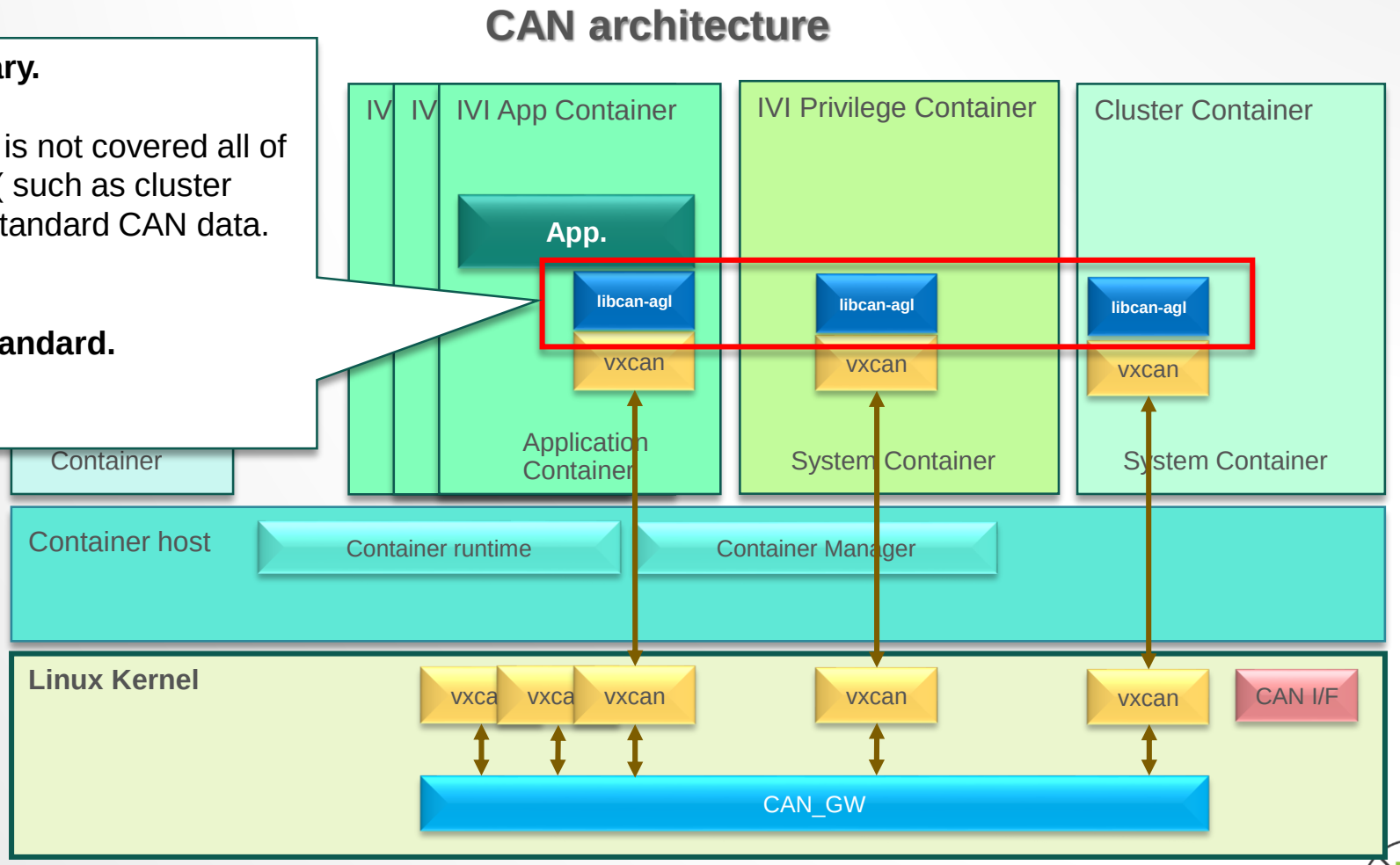
How to abstract CAN data

- 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.



How to abstract CAN data

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

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.