AUTOMOTIVE LINUX SUMMIT

CLOUD CONNECTED VEHICLE BASED ON OPEN SOURCE SOFTWARE

Alex Agizim, EPAM

May, 2017

NEWS SNAPSHOT

First autonomous Toyota to be available in 2020

BMW to launch autonomous iNext in 2021

Fully autonomous vehicles could be ready by 2025, predicts Daimler chairman

Sergey Brin plans to have Google driverless car in the market by 2018

Driverless cars will be in use all over the world by 2025

Elon Musk now expects first fully autonomous Tesla by 2018, approved by 2021

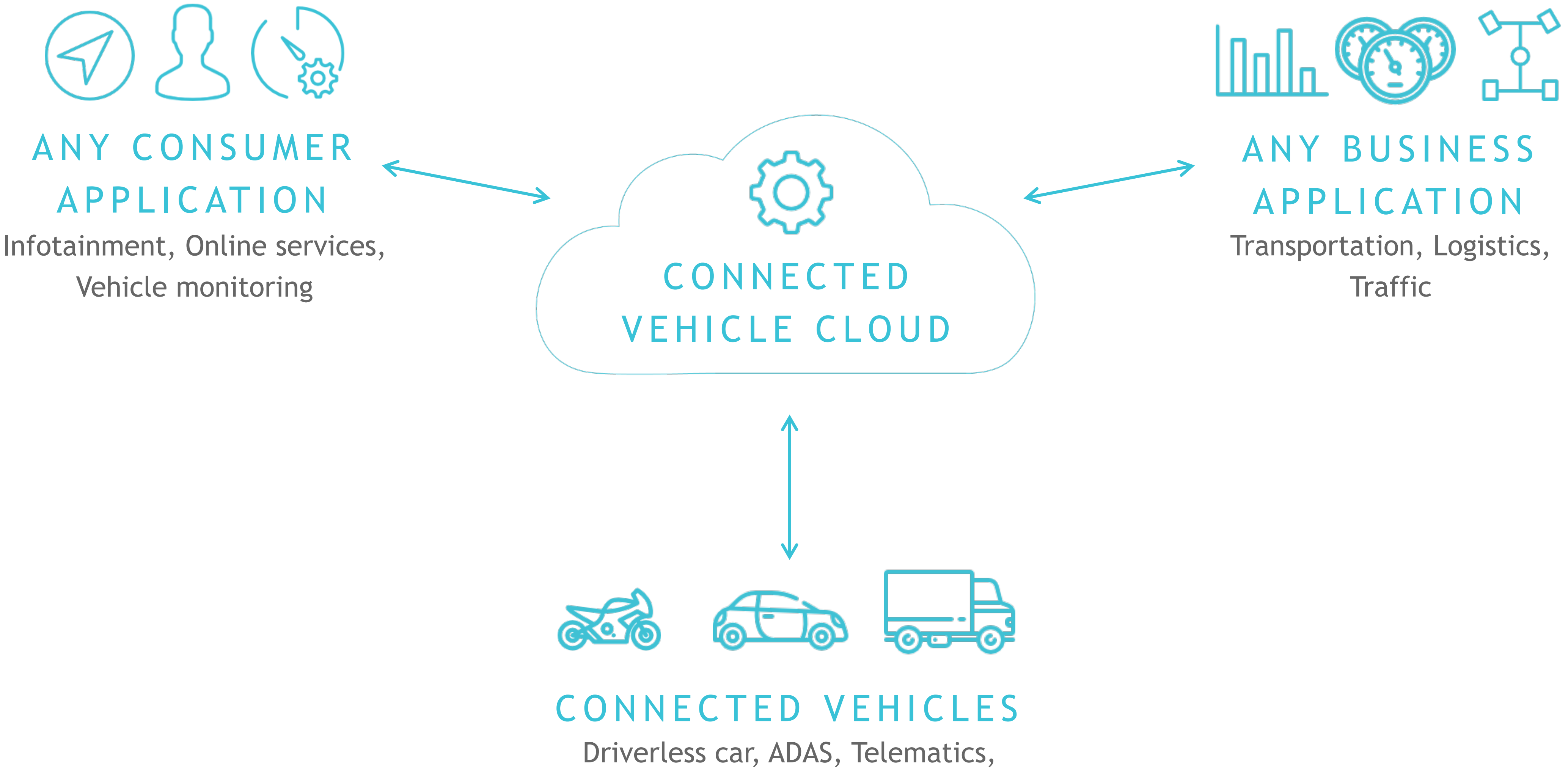
Delphi and MobilEye to provide off-the-shelf self-driving system by 2019

Next generation Audi A8 capable of fully autonomous driving in 2017

Samsung and Siemens — announced M&A deals in the automotive field

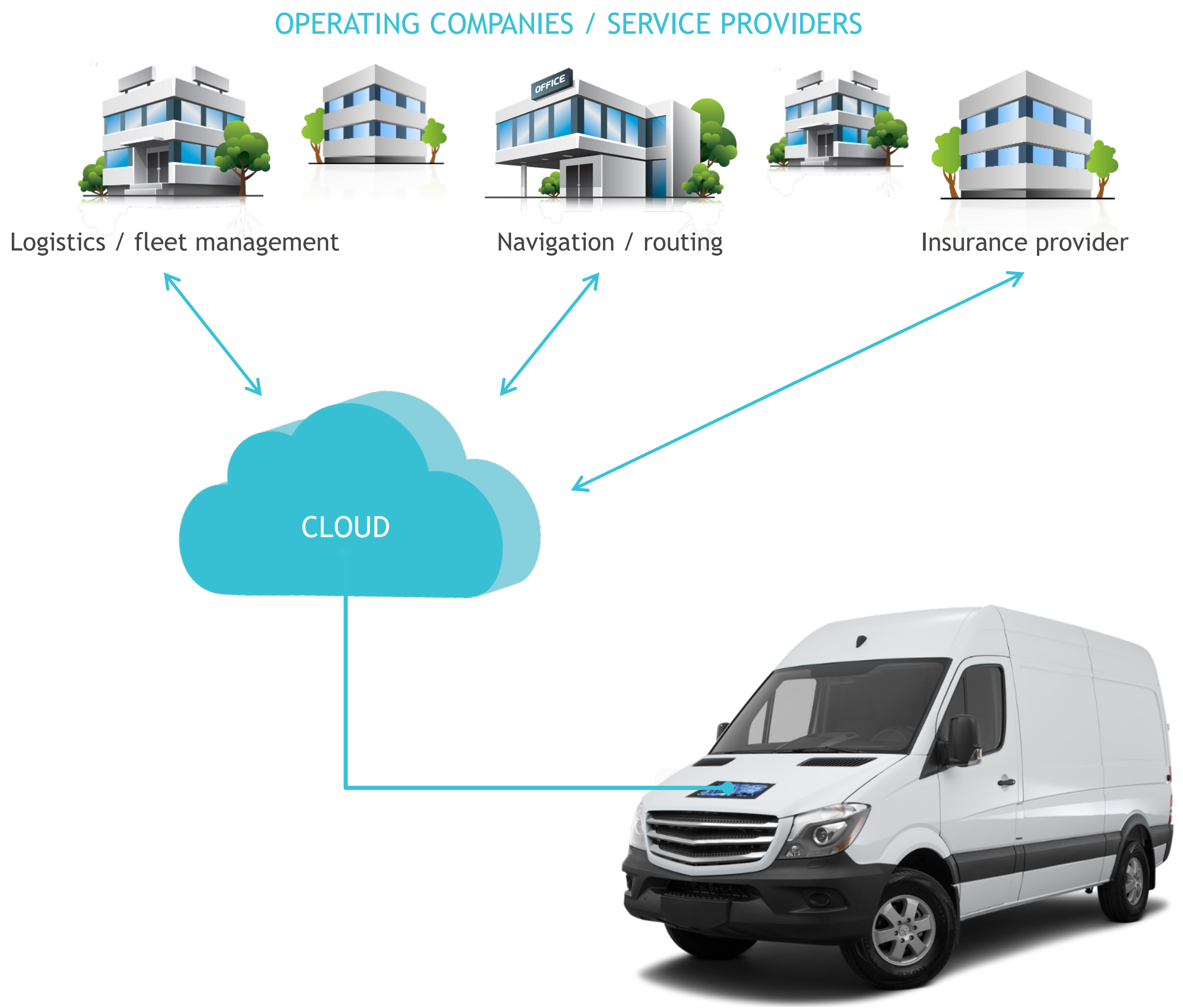
There will be 700 million connected cars by 2022

CONNECTED VEHICLES



CONNECTED VEHICLE CONCEPT

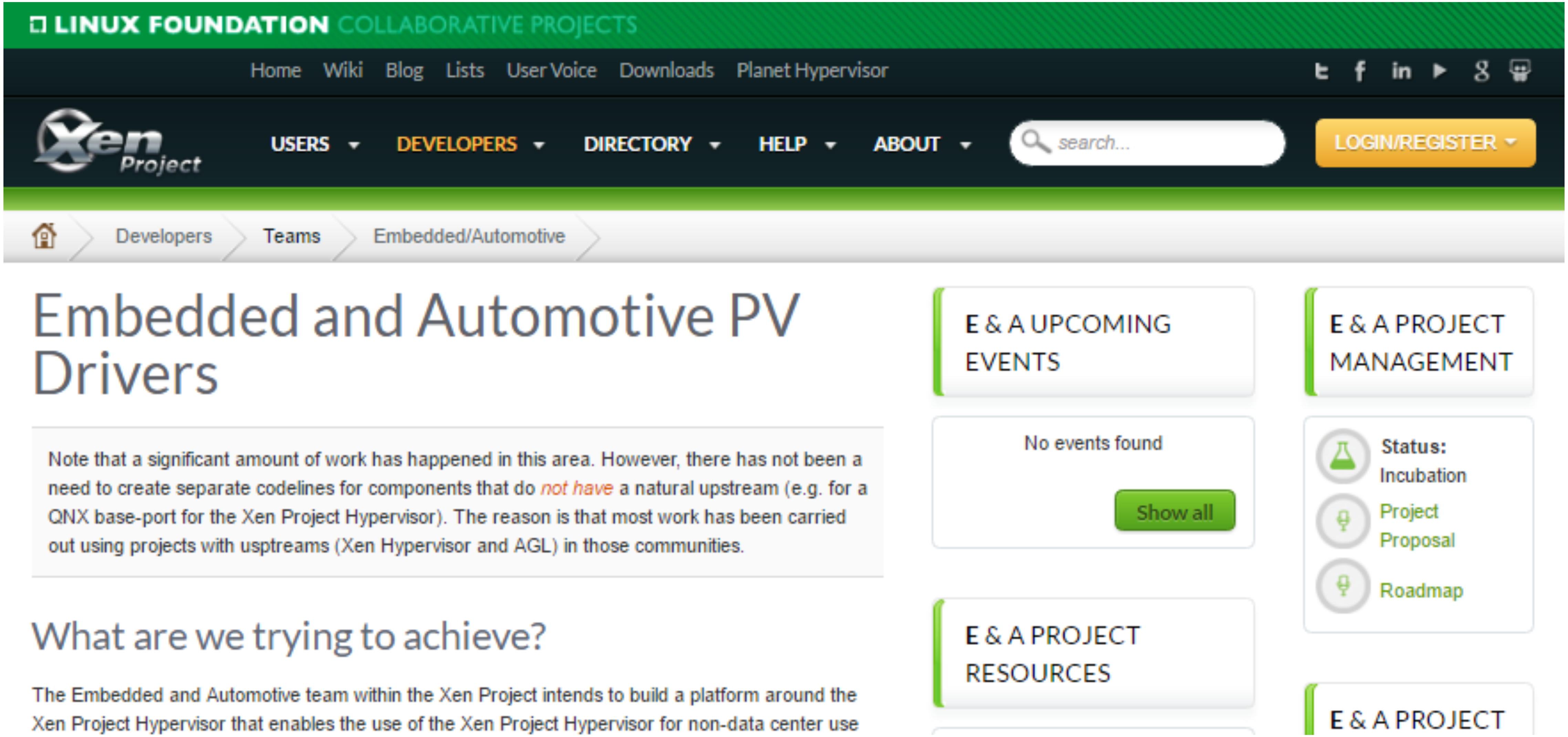
- Develop a platform that enables deployment of service providers' software into vehicles
- Allow developing applications using popular programming languages (c#, java, python, etc.)



HOW IT’S DONE – OUR APPROACH

Isolate safety regulated software from 3rd party apps/services	Develop virtualization solution for software system separation
Integrate with vehicle on-board platform	Adapt Xen hypervisor for Renesas Salvator X board (and other automotive SoCs) • Resources sharing & protection • Real-time scheduling • Security improvements using TEE
Build connected framework enabling in-vehicle 3rd party apps & services development and deployment	
Build service distribution framework	

EPAM LEADS THE XEN FOR AUTOMOTIVE PROJECT



https://www.xenproject.org/developers/teams/embedded-and-automotive.html

XEN FOR AUTOMOTIVE: HW SHARING AND PROTECTION

HW can be directly mapped to guest domains

- Direct interrupts routing
- IO memory regions 1-1 mapping (bit unsafe…)

To provide sharing capabilities PV drivers model exist for SoC peripherals that don’t have SMMU

- Guest domains implement “frontend” drivers that communicate to …
- “backend” drivers that talk to HW, implemented in host (or “driver”) domains

Devices that support ARM SMMU (or custom IO-MMUs) implement best possible sharing option

- Xen controls SMMU
- Memory ranges are switched dynamically
- Interrupts are injected using vGIC

EPAM is driving development for peripherals sharing in Xen

- PV drivers interfaces: sound, display, input
- IO MMU subsystem drivers: ARM SMMU, Renesas IPMMU

XEN FOR AUTOMOTIVE: TEE INTEGRATION

OP-TEE is a Linaro's reference TEE implementation

- Fully Open Source, BSD 2-clause license (GPLv2 for Linux driver, test suite)
- Integrated with Linux kernel (driver upstreamed) and ARM Trusted Firmware (dispatcher merged)

GlobalPlatform APIs support in OP-TEE

- TEE Client API Specification v1.0 ✓
- TEE Internal Core API Specification v.1.1.1 ✓
- TEE Secure Element API Specification v1.1.1 ✓
- TEE Sockets API Specification v1.0 ✗
- Trusted User Interface API Specification v1.0 ✗
- TEE TA Debug Specification v.1.0.1 ✗

EPAM is driving integration of Xen & OP-TEE

- Memory allocation for TAs
- OS ID support for SMC calls
- TEE driver in hypervisor EL0 or stub domain EL1 (TBD)

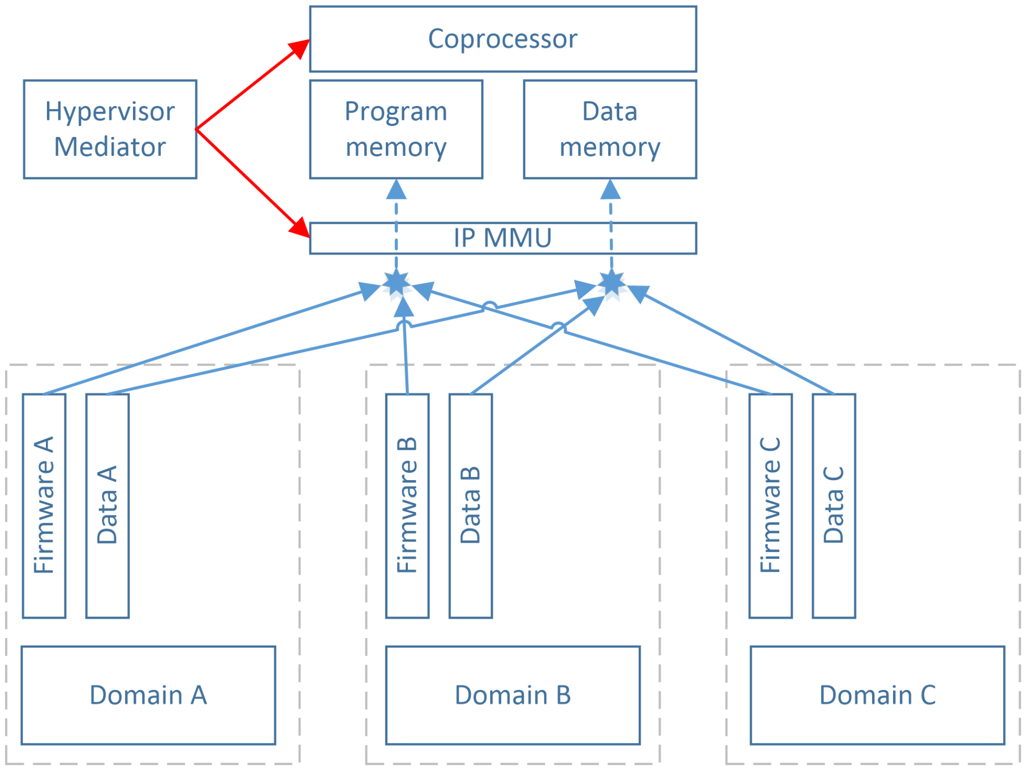
OP-TEE
.org

XEN FOR AUTOMOTIVE: COPROCESSORS SHARING

Coprocessor (any kind of programmable or "smart" peripheral computing device – GPU, DSP, IPU, etc. sharing RAM with main CPU) can be used by different domains concurrently and independently within some time slice

Mediated pass-through approach

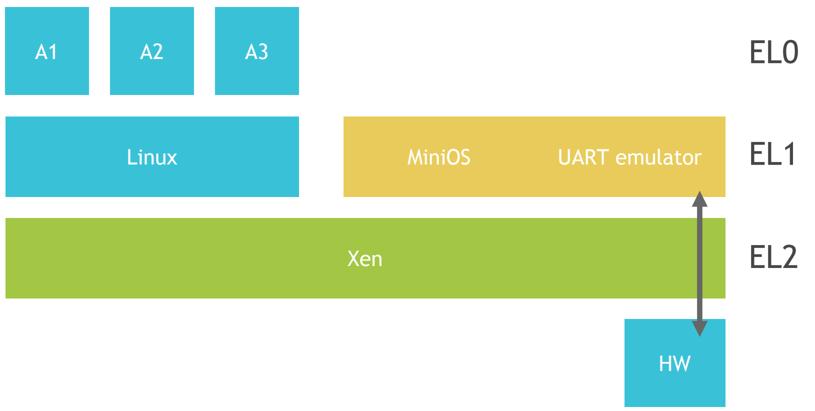
- Different VMs may execute different program stacks (firmwares) on a single coprocessor
- Domains are isolated better since both command and data contexts on a coprocessor are being switched; better isolation leads to improved robustness and security
- Scheduling is more tunable and configurable, e.g. with respect to prioritization or budgeting



XEN FOR AUTOMOTIVE: NATIVE APPLICATIONS & DRIVERS

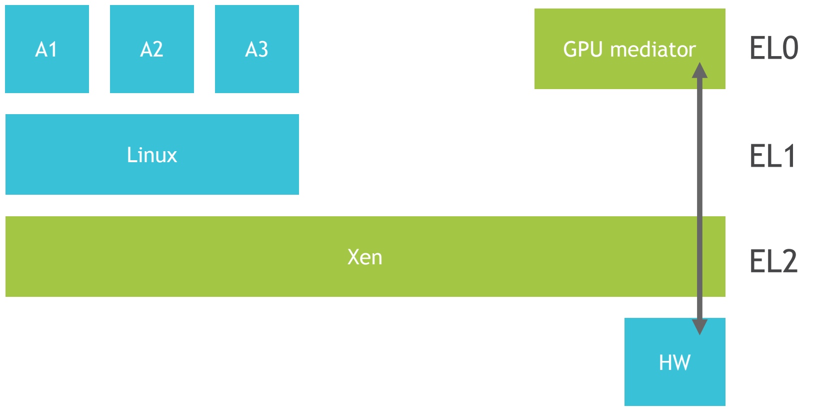
Xen stub domains (initial implementation for ARM exist)

- Loaded as regular EL1 domains (can handle interrupts and other exceptions) but implement simplistic monolithic OS without EL0 applications
- Used for implementing:
  - Device emulation models (fully virtual HW)
  - Hypervisor native drivers



De-privileged applications

- Loaded as ELF modules into Xen and executed with de-privilege bit set effectively putting them to EL0 without underlying EL1 OS (interrupts & other exceptions handling is routed to hypervisor)
- Used for implementing:
  - Hypervisor native applications
  - Platform-dependent out-of-tree hypervisor extensions

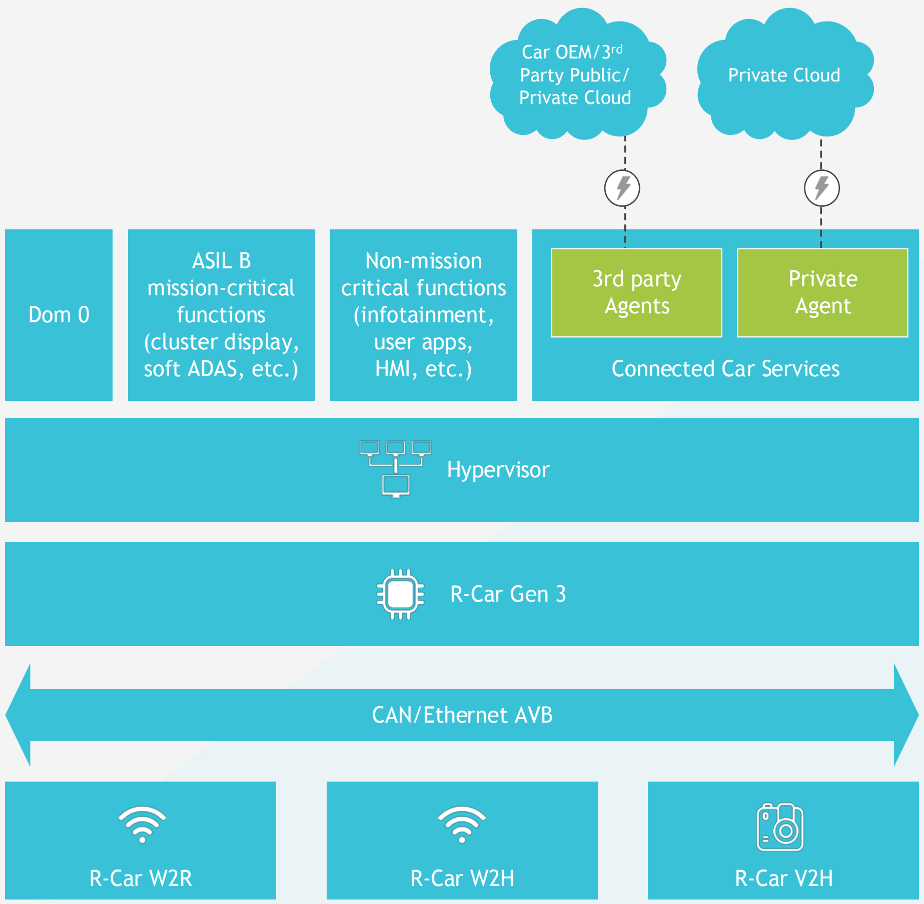


HOW IT’S DONE – OUR APPROACH

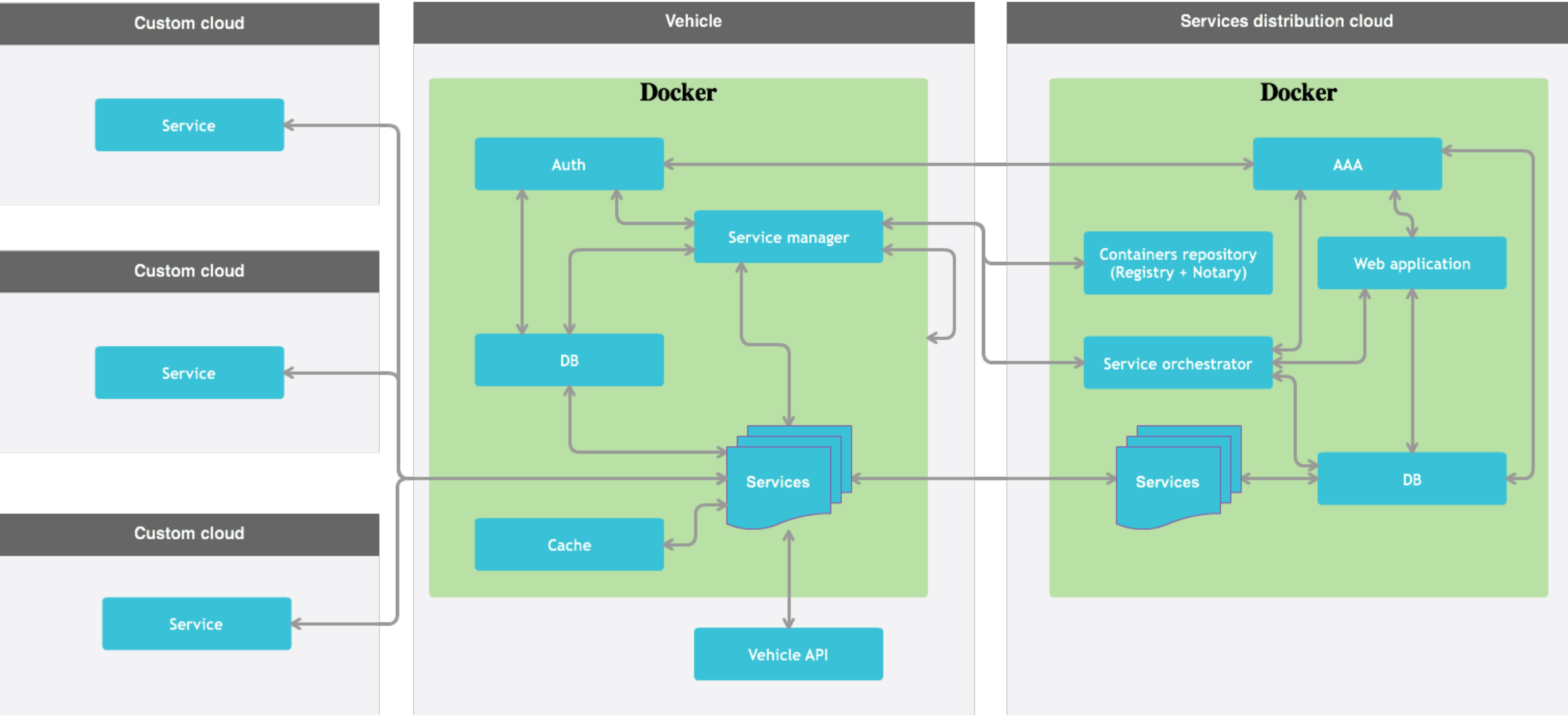
Isolate safety regulated software from 3rd party apps/services	Develop virtualization solution for software system separation
Integrate with vehicle on-board platform	Adapt Xen hypervisor for Renesas Salvator X board (and other automotive SoCs) • Resources sharing & protection • Real-time scheduling • Security improvements using TEE
Build connected framework enabling in-vehicle 3rd party apps & services development and deployment	Integrate Vehicle into Cloud with FUSION solution • Docker-based containers in vehicle • W3C vehicle data & control API • Transparent cloud service development
Build service distribution framework	Define service orchestration architecture

EPAM FUSION

EPAM Fusion is a vehicle service orchestrator which provides ability to easily develop, install, upgrade and remove services on any connected vehicle.



GENERAL SCHEME OF FUSION PLATFORM



CONTAINER SERVICES

- Pre-installed Docker and Docker Compose on each vehicle
- Docker for containerization and Compose for containers management
- Also, pre-installed layers for services or layers for running other services (e.g. Python - this could be a Python-alpine layer which is 89 MB, GoLang- this could be a GoLang-alpine layer which is 258 MB, BusyBox-for C++ code, this is 3,3 MB etc.)
- Docker engine uses about 10 MB of RAM and 230 MB of HDD
- We run each vehicles service in separate Docker container so each service is isolated
- Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries. This guarantees that software will always run the same, regardless of its environment.

Comparison of containers and virtual machines

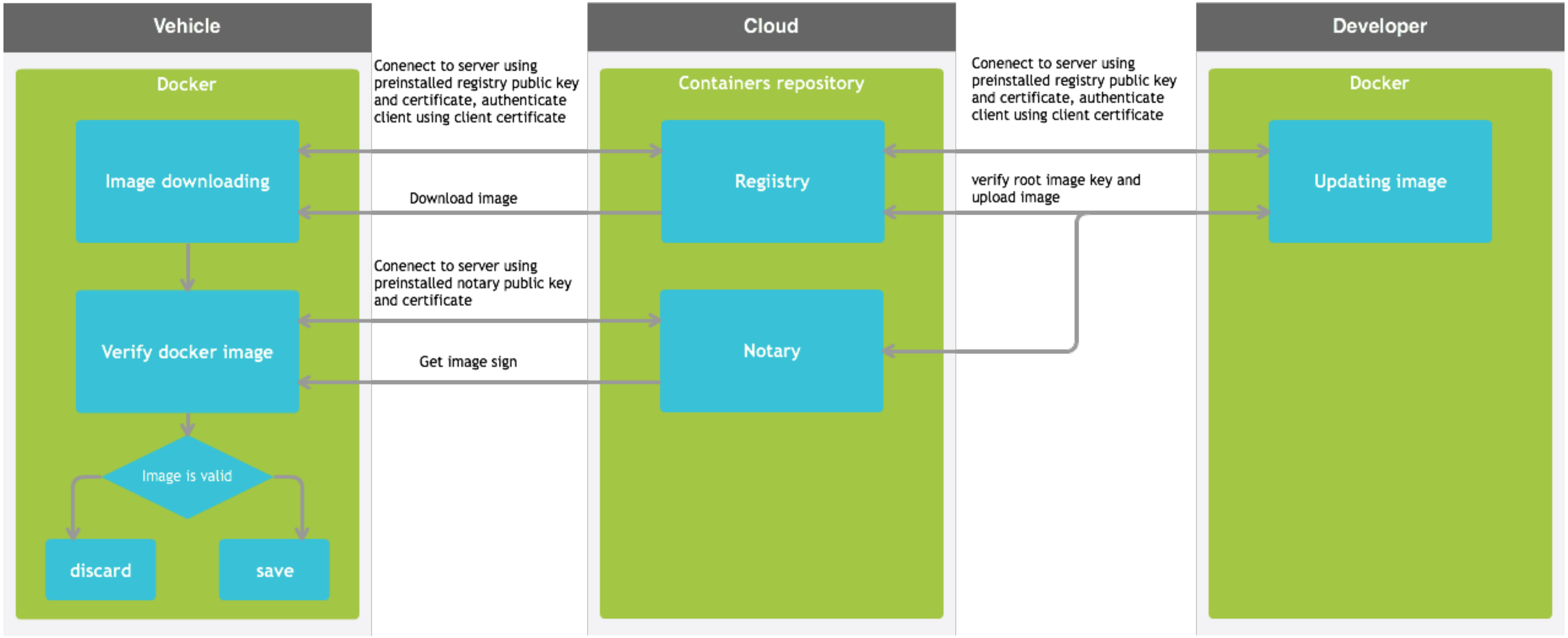
LIGHTWEIGHT SECURE BY DEFAULT

There are four major areas to consider when reviewing Docker security:

- the intrinsic security of the kernel and its support for namespaces and cgroups
- the attack surface of the Docker daemon itself
- loopholes in the container configuration profile, either by default, or when customized by users
- the “hardening” security features of the kernel and how they interact with containers

END-TO END NETWORK DATA ENCRYPTION

While installing or updating services Docker Notary and Registy services are providing security



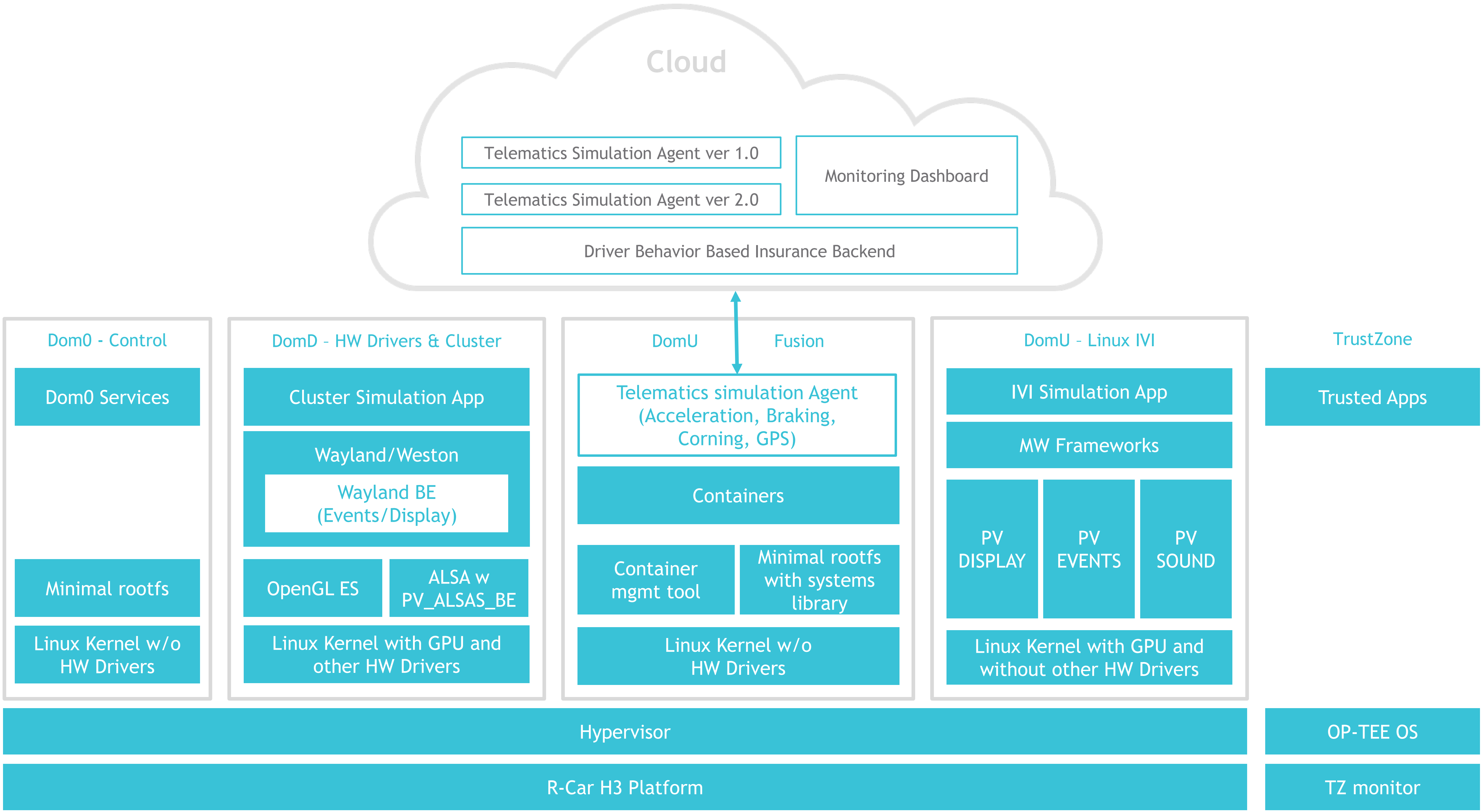
Security in services end-to end network data transmissions is based on how services are written.

VEHICLE DATA AND CONTROL APIS (W3C)

Each service or container communicate with vehicle only through API provided by our platform or manufacturer. API is based on the standards being made by W3C (https://www.w3.org/auto/wg/).

SERVICES DATA FLOW:

- Single vehicle’s service and cloud components would exchange data in their own way. This could generate duplicated network traffic. To minimize the quantity of requests to CAN bus we can make 1 request for a parameter needed, store it in cache and make accessible to all interested services.
- We could have one build-in data collecting service on vehicle sending all telemetry data to a database that will support all services. It will allow to store all historical data (with no gaps) of vehicle or driver. Vehicle resources would be used wisely and in controllable manner. Network traffic would be minimized as only changes are sent to the database. Small amount of requests to CAN bus. We could have some legal issues though.


Cloud
Telematics Simulation Agent ver 1.0
Telematics Simulation Agent ver 2.0
Monitoring Dashboard
Driver Behavior Based Insurance Backend
Dom0 - Control
Dom0 Services
Minimal rootfs
Linux Kernel w/o HW Drivers
DomD - HW Drivers & Cluster
Cluster Simulation App
Wayland/Weston
Wayland BE (Events/Display)
OpenGL ES
ALSA w PV_ALSAS_BE
Linux Kernel with GPU and other HW Drivers
DomU
Fusion
Telematics simulation Agent (Acceleration, Braking, Corning, GPS)
Containers
Container mgmt tool
Minimal rootfs with systems library
Linux Kernel w/o HW Drivers
DomU - Linux IVI
IVI Simulation App
MW Frameworks
PV DISPLAY
PV EVENTS
PV SOUND
Linux Kernel with GPU and without other HW Drivers
TrustZone
Trusted Apps
Hypervisor
OP-TEE OS
R-Car H3 Platform
TZ monitor

FUSION DEMO VIDEO

Let’s look at demo now

WHAT WAS DONE

- Safety regulated software isolated from 3rd party apps/services

- Connected vehicle services controlled by service vendor. Service vendor may handle connectivity problems by implementing off-line actions on vehicle service

- Service developers don’t need any specific knowledge in automotive embedded domain

THANK YOU!

Alex **AGIZIM**

CTO Automotive & Embedded,
EPAM

Alex_Agizim@epam.com