Xen for Embedded, IoT, Edge

Stefano Stabellini
Xen Maintainer, Principal System Software Engineer
2018/09

XILINX

Lightning Xen Update

XILINX

Very Embedded Requirements

- Real Time
  - Low Deterministic IRQ Latency
  - Static Partitioning
  - Real Time Schedulers
- Short Boot Times
- Device Virtualization
  - Device Assignment
  - Device Sharing
  - Driver Domains
  - VM to VM communications
- Certifications
  - Small Code Base
  - Type-1

Static Partitioning

sched=null vwfi=native

Static Partitioning

sched=null vwfi=native

2.5 us

XILINX

VM to VM communication mechanisms

>**Libvchan**

- >>Linux LIbrary
- >>Direct VM to VM channel based on a ring on shared memory
- >>libxenvchan_send and libxenvchan_recv

>**PVCalls**

- >>Socket API virtualization
- >>VM to VM communications mediated by the backend domain (dom0)
- >>“Io” as an inter-VMs communication namespace

>**V4V**

- >>Linux library and hypercall
- >>VM to VM communication mediated by Xen
- >>Trivial to implement in your kernel
- >>Not fully upstream

Shared Memory

>**Completely Configurable**

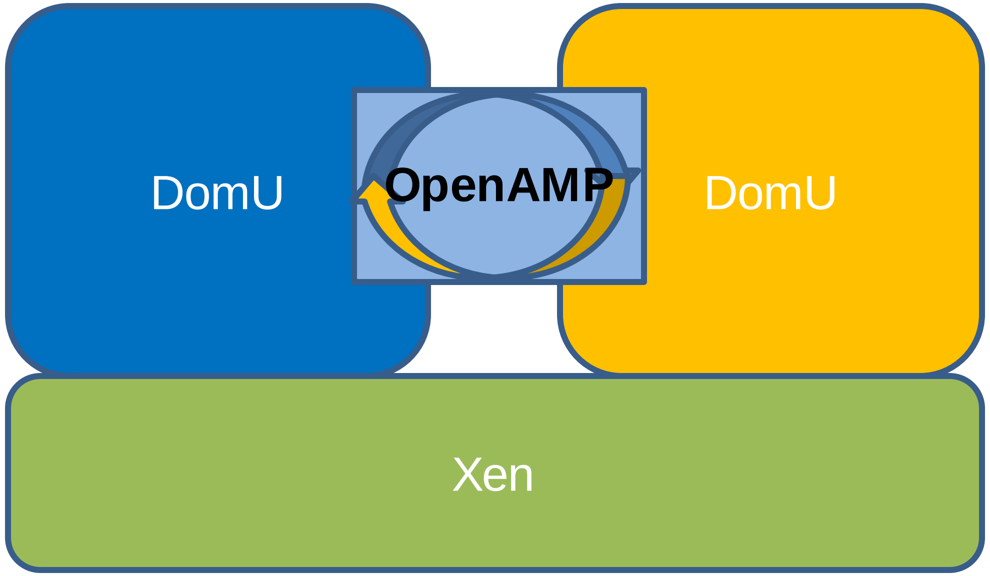
>>Support any memory attributes including cacheable memory

>**No need for Xen support to use it**

>**Can export the memory to Linux userspace and use OpenAMP**

```
static_shm = ["id=ID1, begin=0x40000000, size=0x1000, role=master"]
```

```
static_shm = ["id=ID1, offset=0, begin=0x48000000, size=0x1000, role=slave"]
```

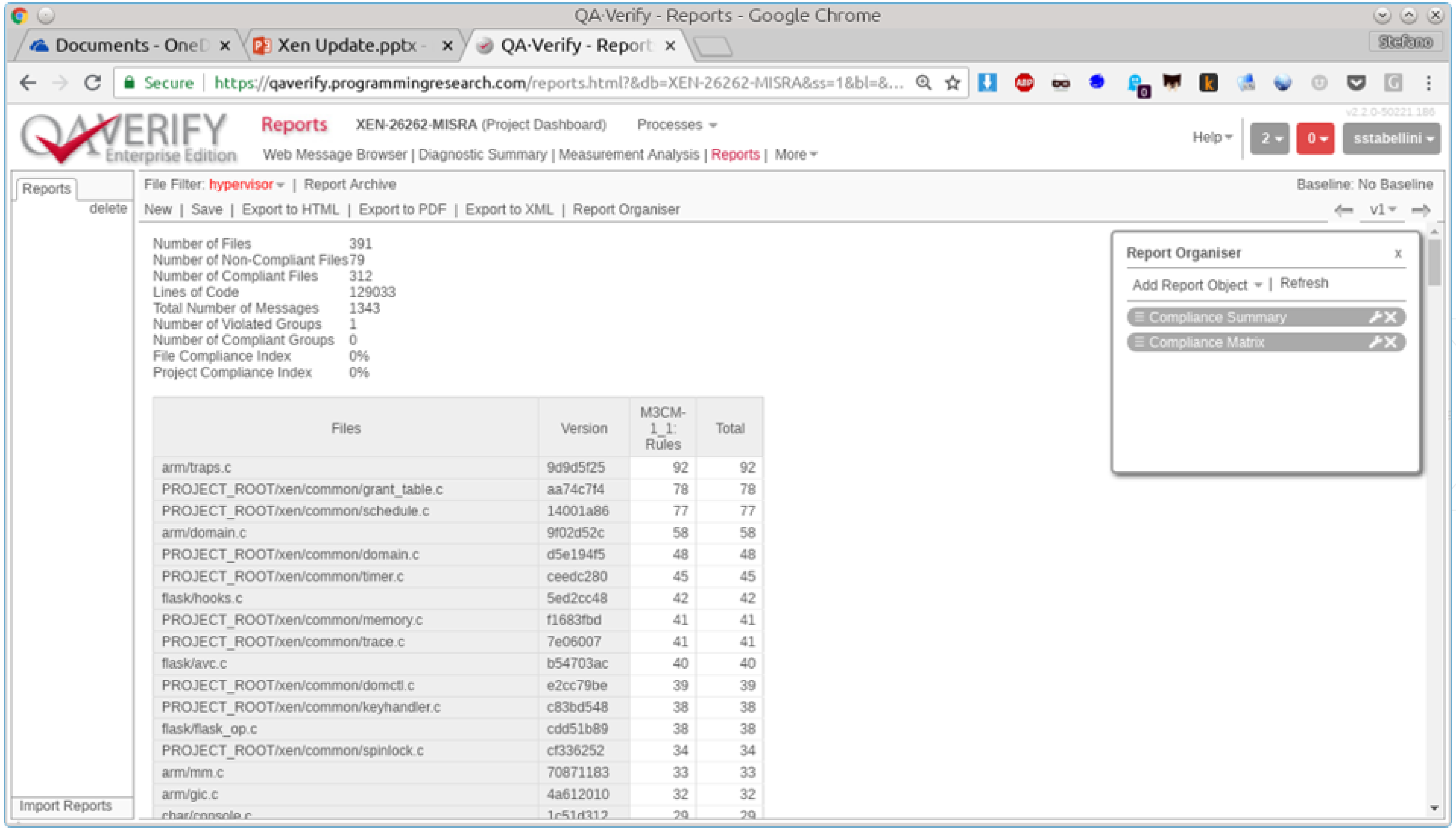


Reducing Code Size

```
() work – Konsole
File  Edit  View  Bookmarks  Settings  Help
stefanos@xsjstefanos50:/local/repos/xen/xen$ make cloc
cloc --list-file=/tmp/tmp.lL2EdV9dLA
     143 text files.
     143 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.60  T=0.26 s (546.4 files/s, 262525.6 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                              126          10527          10408          45144
Assembly                        17            249            937           1439
-------------------------------------------------------------------------------
SUM:                           143          10776          11345          46583
-------------------------------------------------------------------------------
rm /tmp/tmp.lL2EdV9dLA
stefanos@xsjstefanos50:/local/repos/xen/xen$ 
sstabellini : bash        () work
```

Certifications



Files	Version	M3CM-1_1: Rules	Total
arm/traps.c	9d9d5f25	92	92
PROJECT_ROOT/xen/common/grant_table.c	aa74c7f4	78	78
PROJECT_ROOT/xen/common/schedule.c	14001a86	77	77
arm/domain.c	9f02d52c	58	58
PROJECT_ROOT/xen/common/domain.c	d5e194f5	48	48
PROJECT_ROOT/xen/common/timer.c	ceedc280	45	45
flask/hooks.c	5ed2cc48	42	42
PROJECT_ROOT/xen/common/memory.c	f1683fbd	41	41
PROJECT_ROOT/xen/common/trace.c	7e06007	41	41
flask/avc.c	b54703ac	40	40
PROJECT_ROOT/xen/common/domctl.c	e2cc79be	39	39
PROJECT_ROOT/xen/common/keyhandler.c	c83bd548	38	38
flask/flask_op.c	cdd51b89	38	38
PROJECT_ROOT/xen/common/spinlock.c	cf336252	34	34
arm/mm.c	70871183	33	33
arm/gic.c	4a612010	32	32
char/console.c	1c51d312	29	29

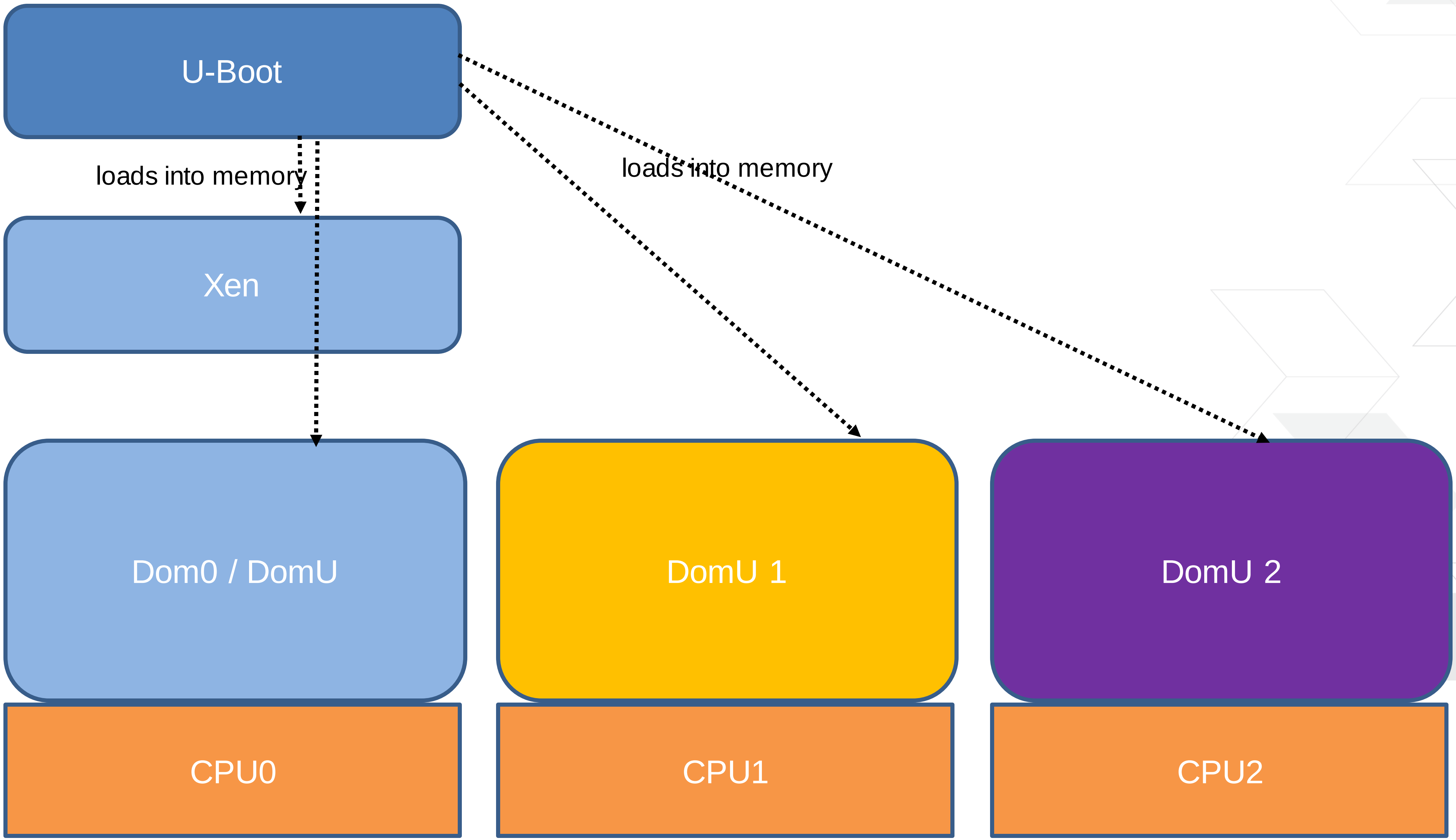
Make xen.git certifiable:

- Reduce code size
- Fix compliance violations reported by PRQA

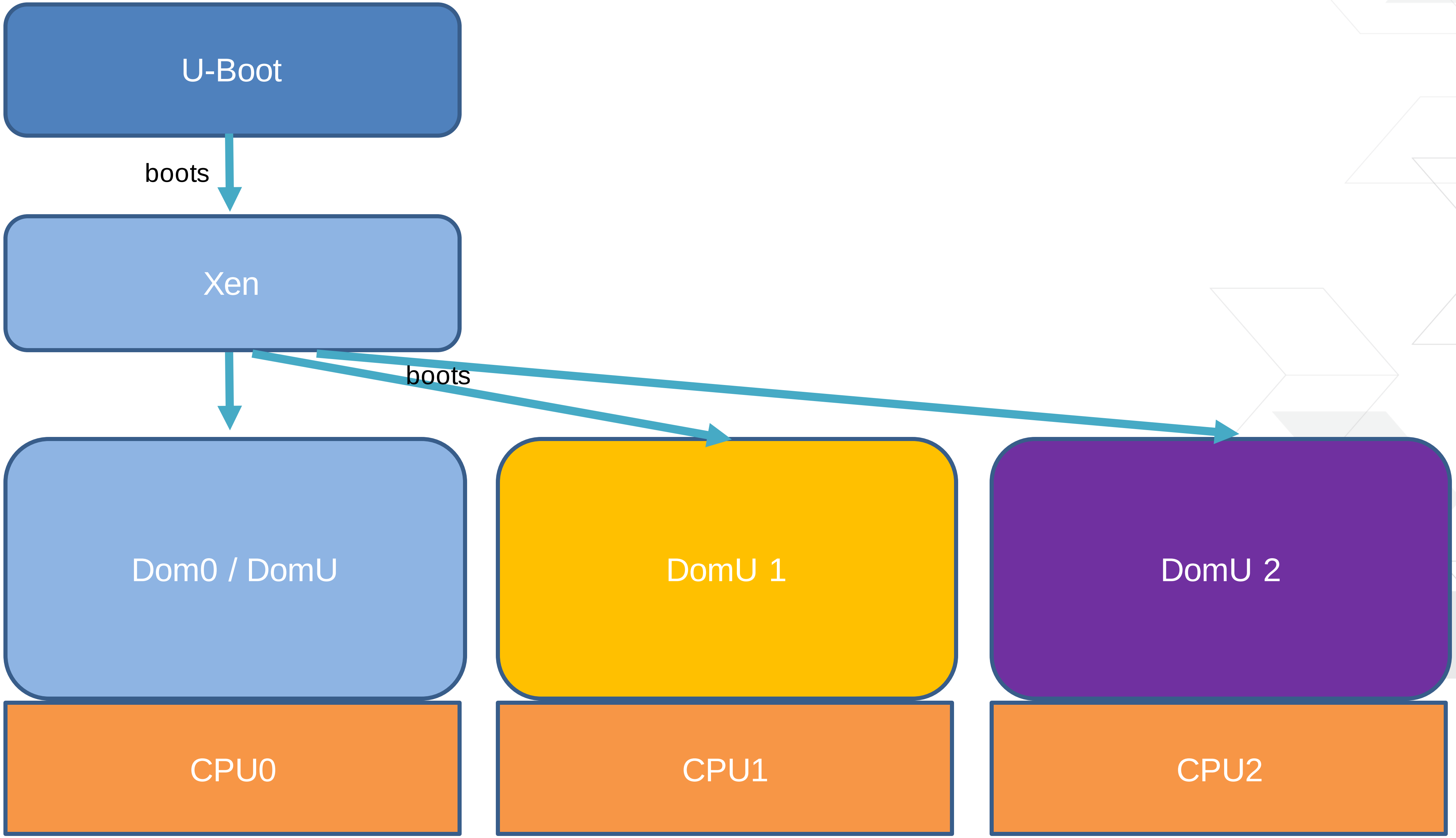
Ideas on how to do certifications in a Xen Project (Linux Foundation) context

XILINX

Dom0-less


U-Boot
loads into memory
Xen
loads into memory
Dom0 / DomU
DomU 1
DomU 2
CPU0
CPU1
CPU2

Dom0-less



XILINX.

Secure Containers at the Edge

The Problem

Package applications for the target

- Contain all dependencies
- Easy to update
- Independent lifecycle

Run applications on the target

- Run in isolation
- No interference between applications

XILINX

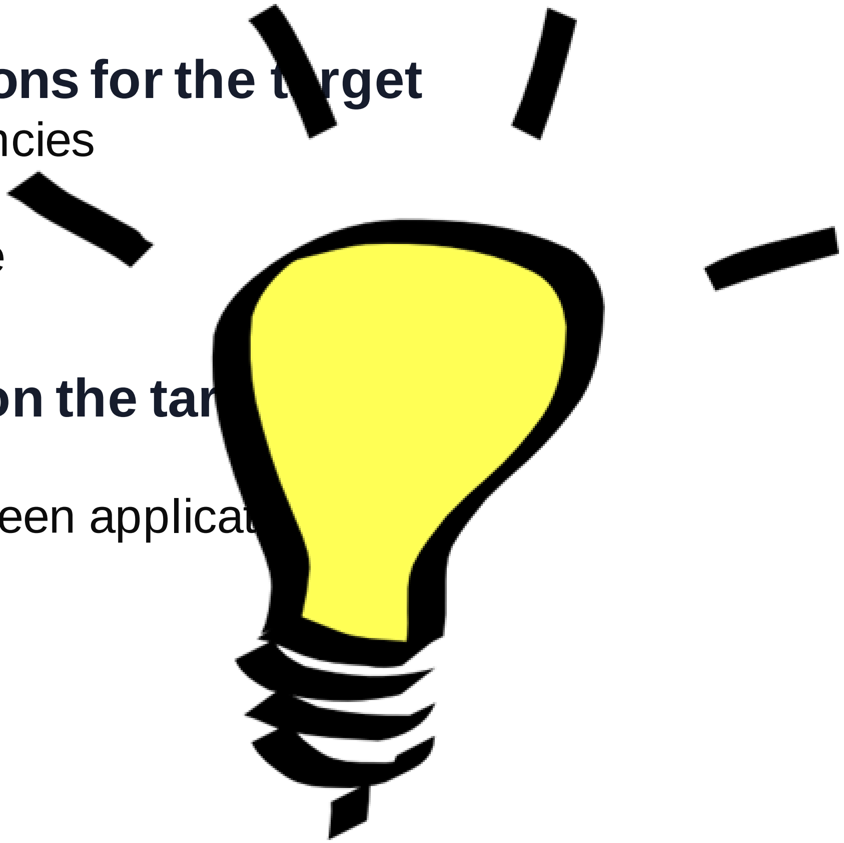
The Problem

Package applications for the target

- Contain all dependencies
- Easy to update
- Independent lifecycle

Run applications on the tar

- Run in isolation
- No interference between applicat



XILINX

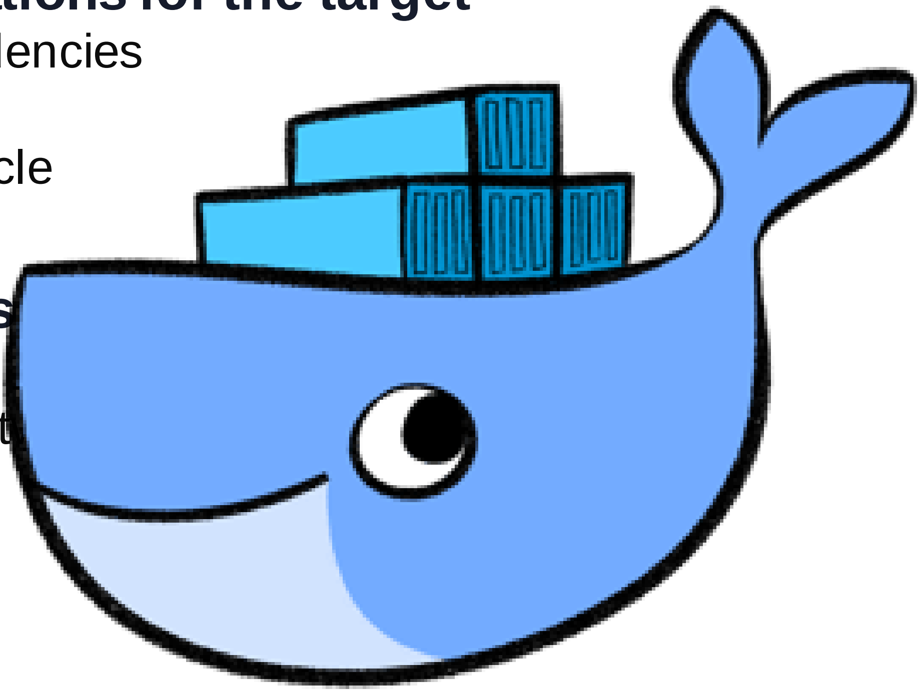
The Problem

Package applications for the target

- Contain all dependencies
- Easy to update
- Independent lifecycle

Run applications

- Run in isolation
- No interference bet

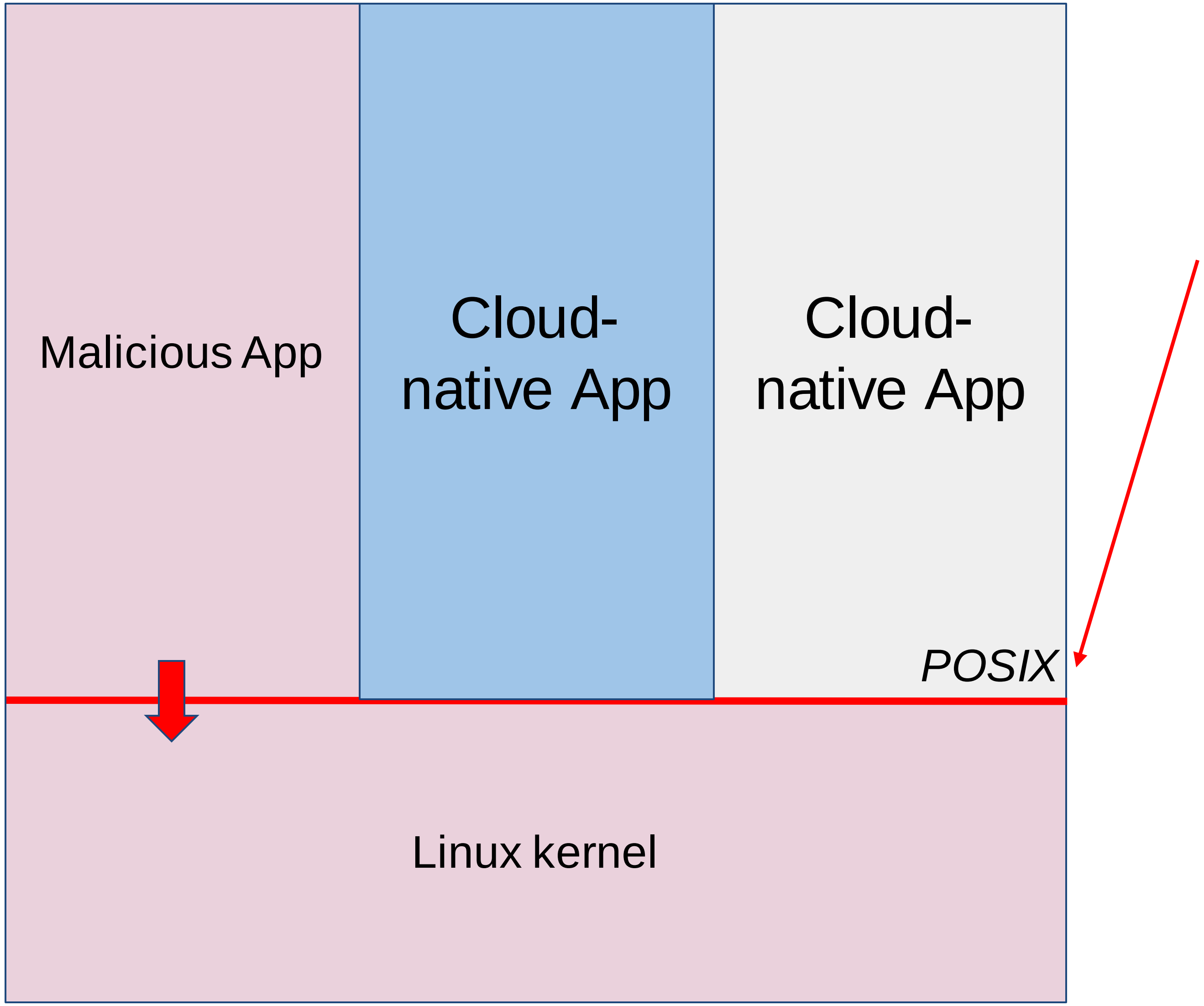


XILINX

Packaging vs. Runtime

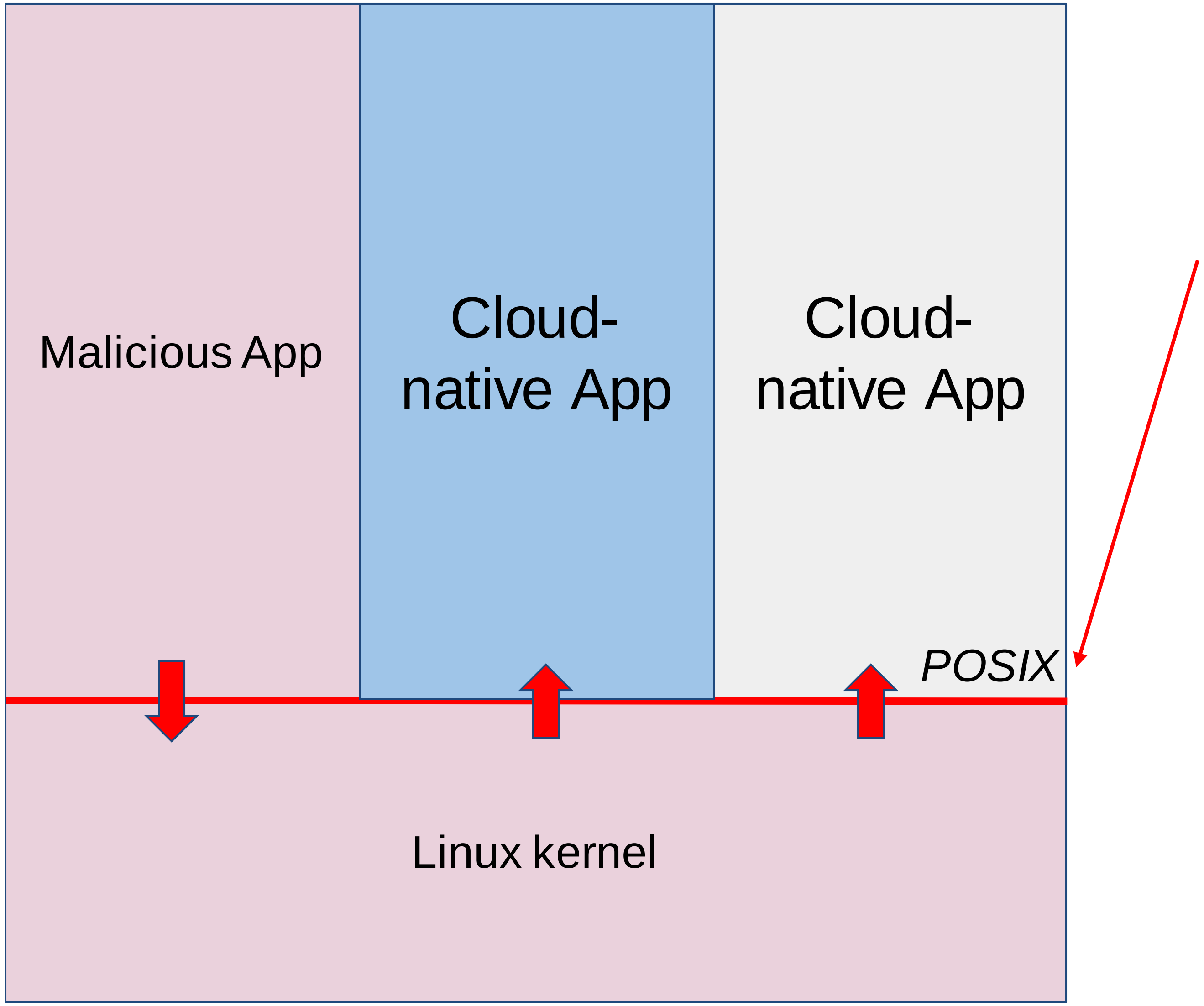
OCI Image Spec vs. OCI Runtime Spec

The problem with Linux namespaces



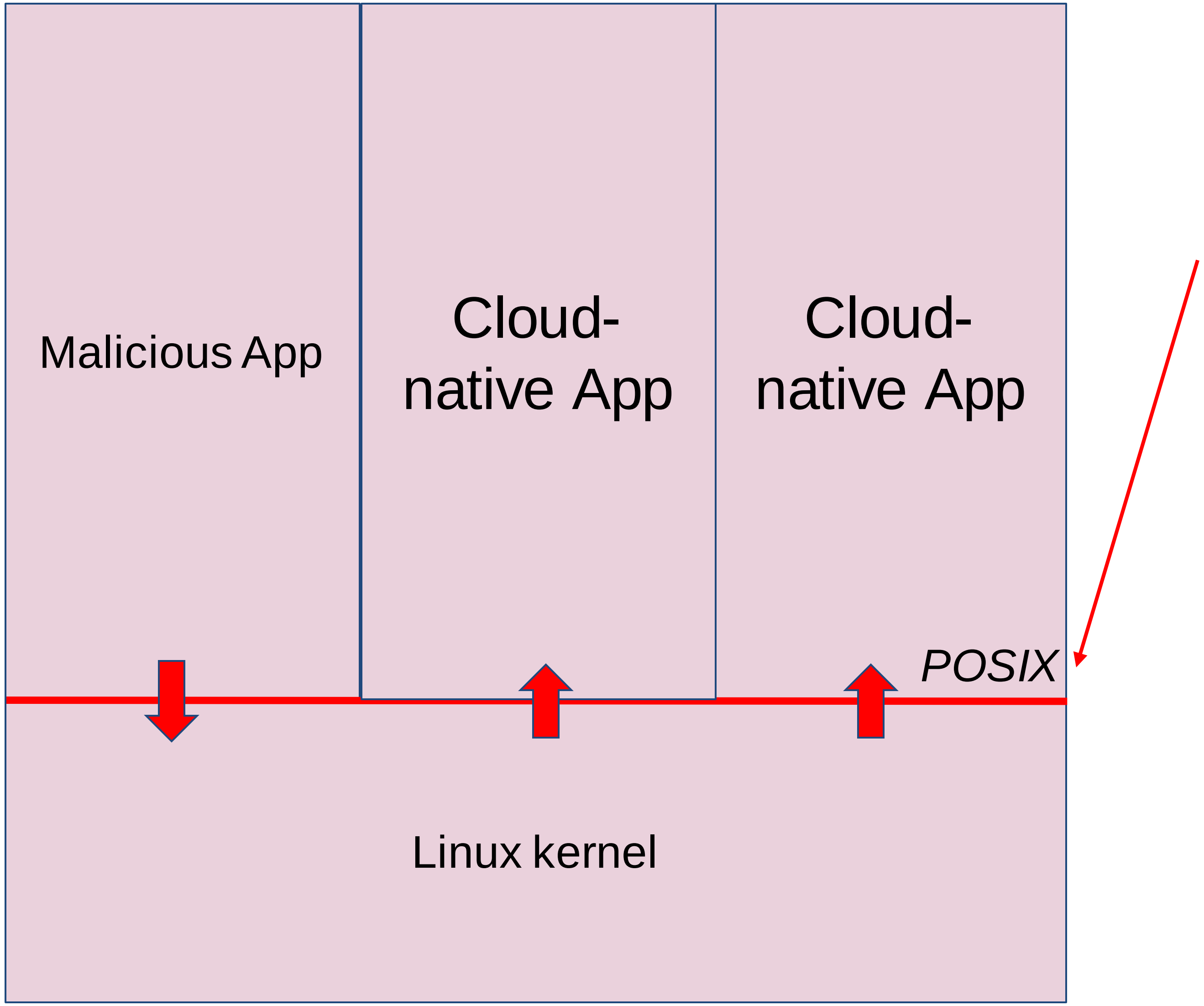
Large surface of attack

On average, 3 privilege escalation vulnerabilities per Linux release!



Large surface of attack

On average, 3 privilege escalation vulnerabilities per Linux release!



Large surface of attack

On average, 3 privilege escalation vulnerabilities per Linux release!

Security hardening techniques

From “Understanding and Hardening Linux Containers” by NCC Group:

- Run unprivileged containers (user namespaces, root capability, dropping)
- Apply a Mandatory Access Control system, such as SELinux
- Build a custom kernel binary with as few modules as possible
- Apply sysctl hardening
- Apply disk and storage limits
- Control device access and limit resource usage with cgroups
- Drop any capabilities which are not required for the application within the container

[...]

Security hardening techniques

[...]

- Use custom mount options to increase defense in depth
- Apply GRSecurity and PAX patches to Linux
- Reduce Linux attack surface with Seccomp-bpf
- Isolate containers based on trust and exposure
- Logging, auditing and monitoring is important for container deployment
- **Use hardware virtualization along application trust zones**

Security hardening techniques

Securing Linux Namespaces is possible but very difficult

- It requires specific knowledge of the cloud native app
- Auditing and monitoring should performed everywhere

Using virtualization for isolation is still recommended

Linux Namespaces: very embedded problems

>**Mixed-criticality** is not supported

>**Limits** on resource utilization are hard to enforce

>**Real-Time** support is difficult

>**Certifications** are very difficult

The Solution: Xen as Container Runtime

- **Security, Isolation and Partitioning**
  - Multi-Tenancy
  - Mixed-Criticality Workloads
- **Hardware Access to Applications**
- **Real-Time Support**

> ViryaOS: a ready-to-use runtime environment for VMs and Secure Containers

The Problem #2

>Cross-building multiple VMs is difficult

>Assembling the output in a single runnable image is a manual process

Embedded and IoT use pattern

>**Typically users know all the VMs they need beforehand**

>**They still need to:**

- >>Build them all, plus Xen and Dom0
- >>Install all images on target
- >>Partition the hardware using device assignment
  - Edit the Dom0 device tree
  - Generate appropriate device trees for DomUs with device nodes
- >>Plan for images upgrades and security fixes

It’s a lot of work!


You think this is bad enough...
...then you try disaggregation

Current Status

- **Everybody has their own scripts and handcrafted solutions**
  - They are limited
  - Only target one use-case
  - Limited support for driver domains and service domains
  - Only support one hardware platform

- **We would all benefit from a unifying effort**

ViryaOS

A proposal for a new Xen Project sub-project

ViryaOS

>a **Secure** Xen based runtime

>**Containers** supported natively

>a turnkey solution

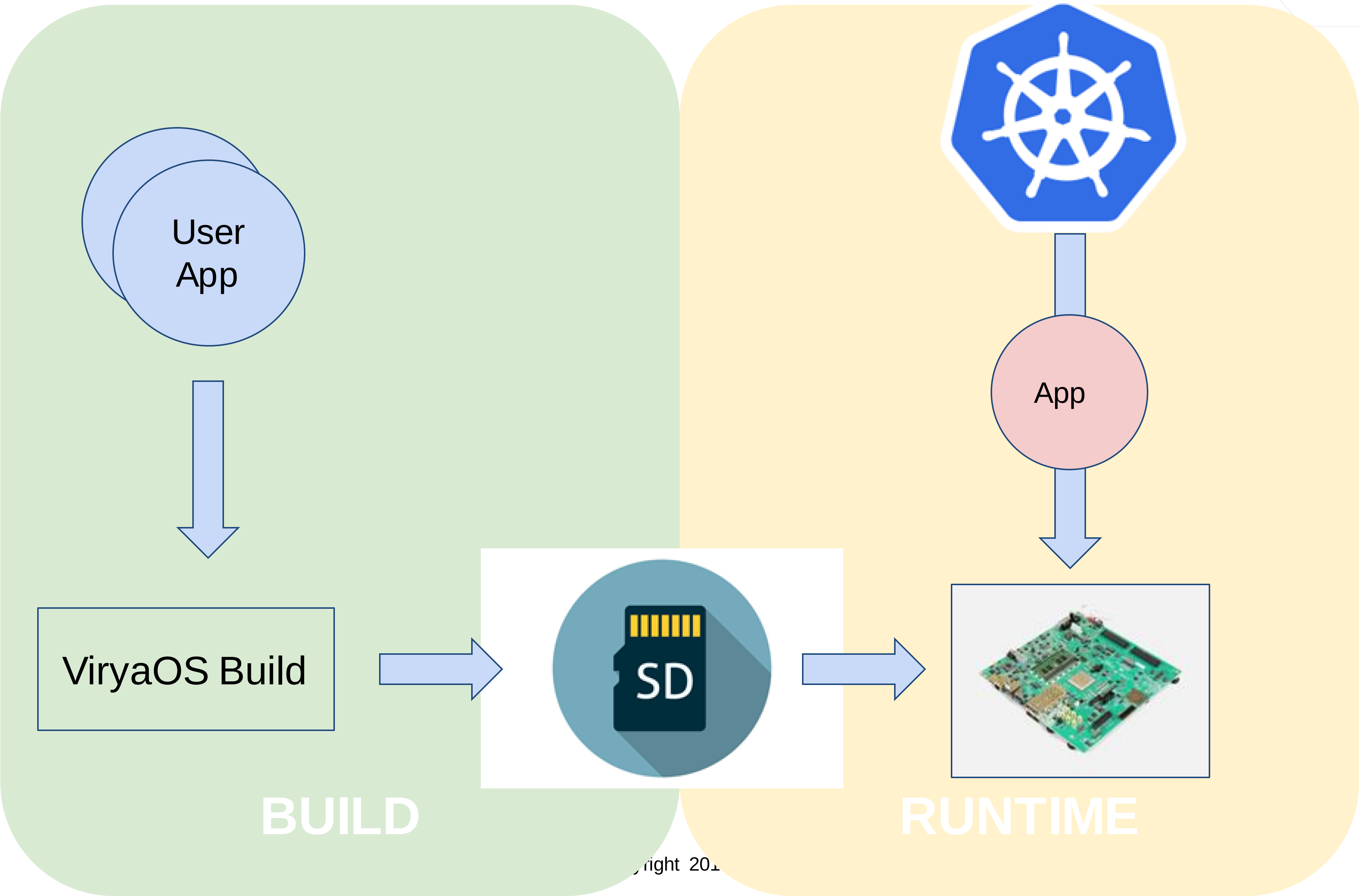
>a **Flexible** build system

>support aarch64 and x86_64

>Targeted at embedded and IoT

XILINX

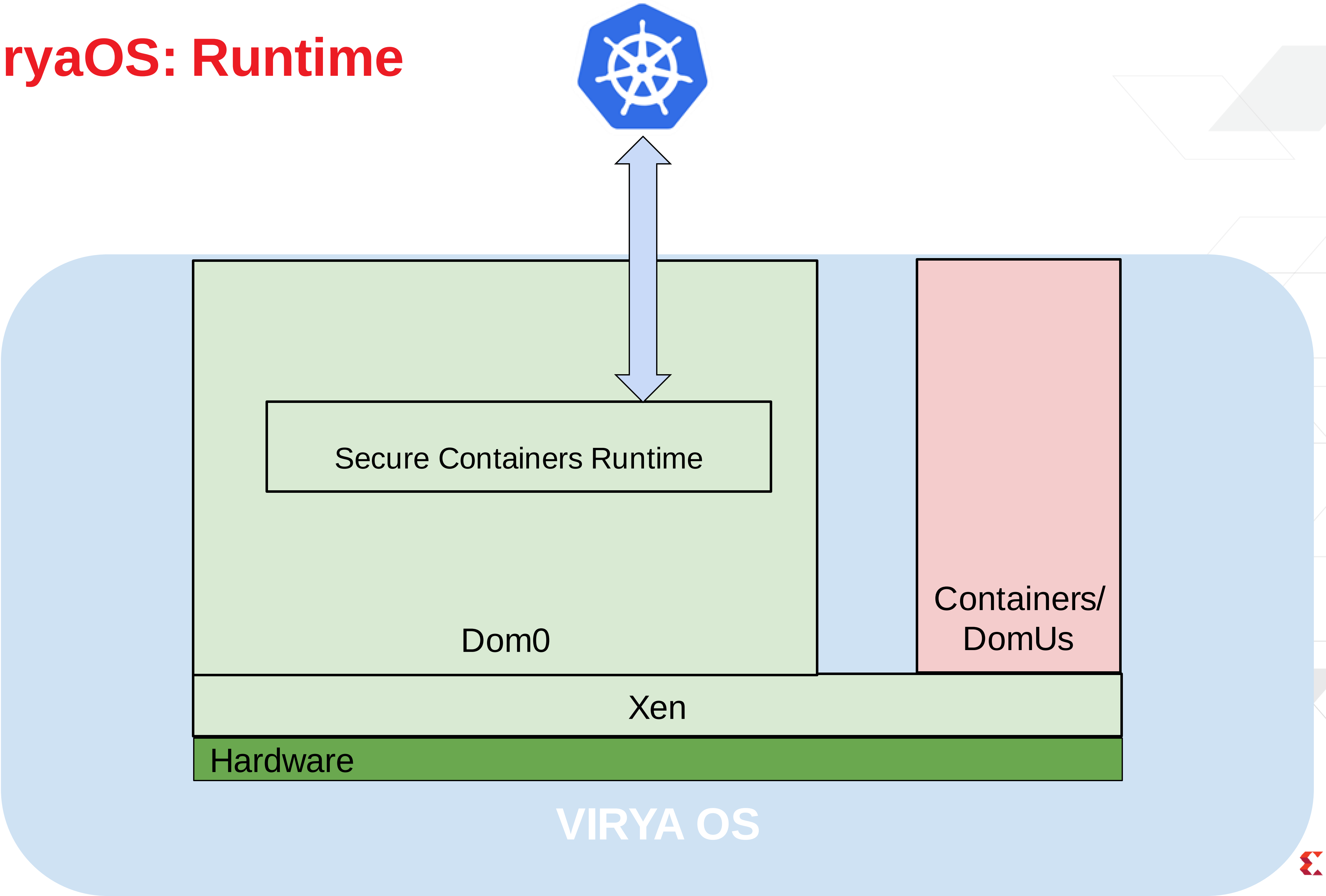
ViryaOS

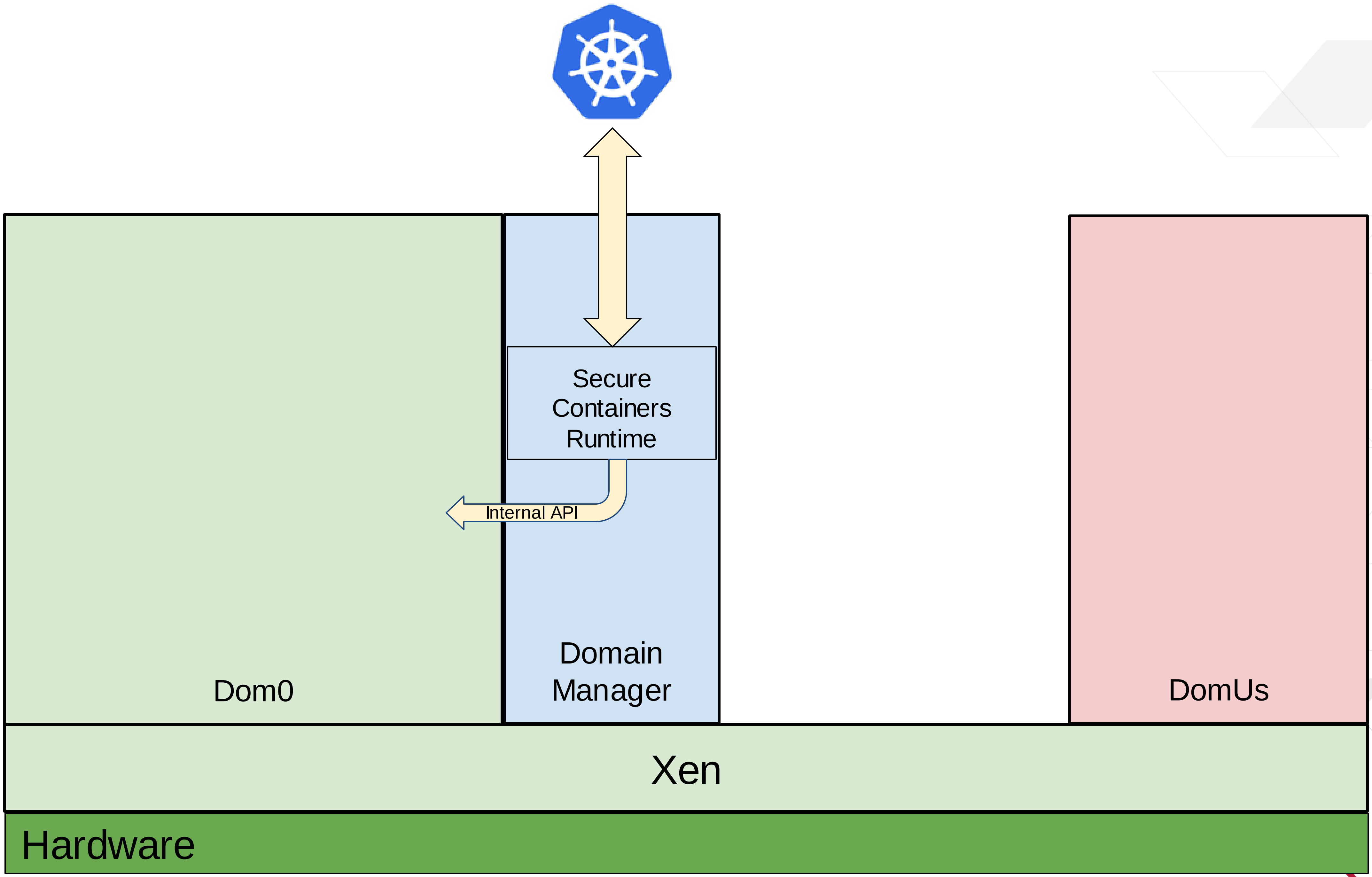


ViryaOS: Runtime

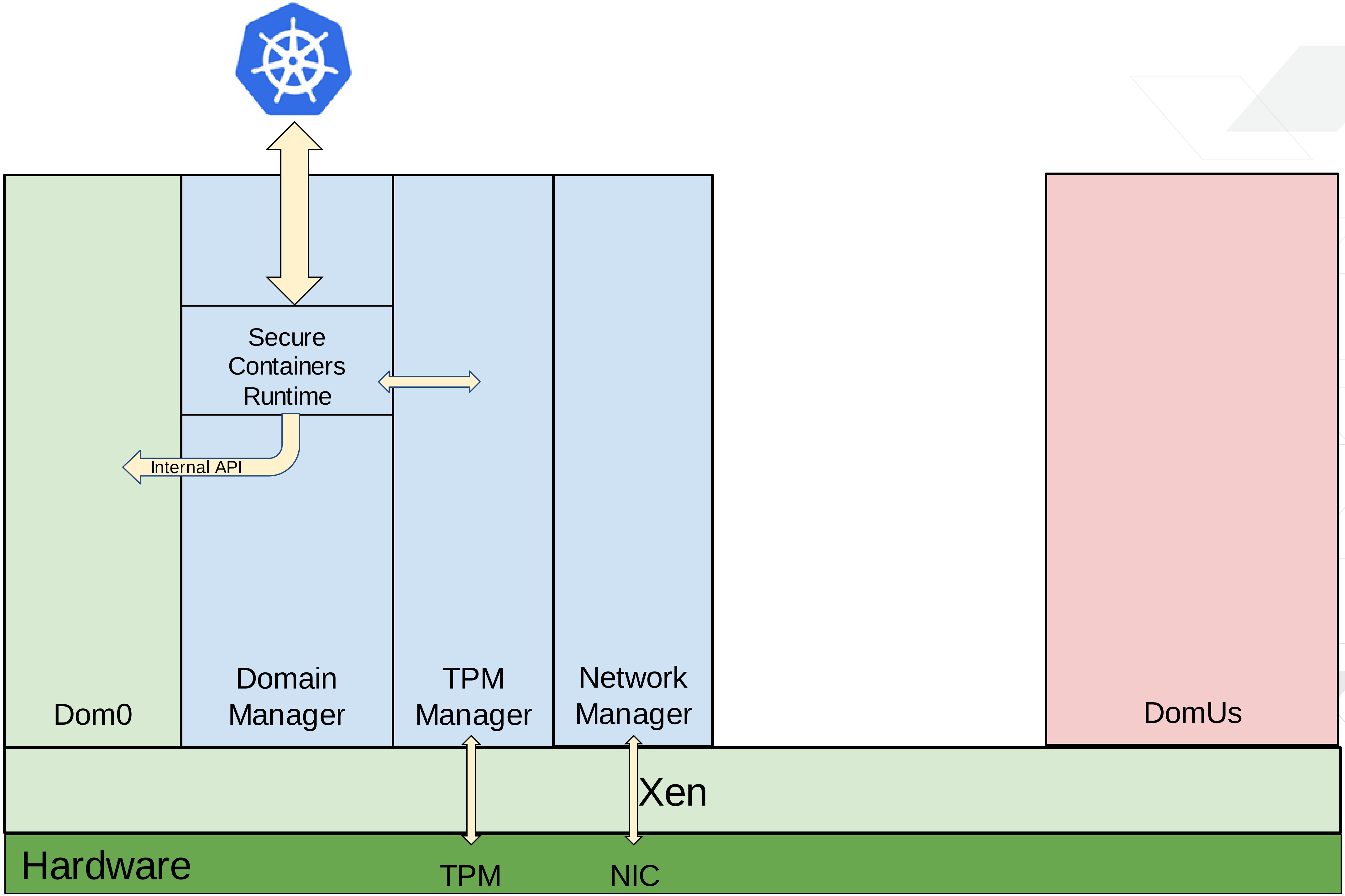
- >**Dynamically deploy VMs and Secure Containers**
- >**Containers are run securely, transparently as Xen VMs**
  - >>1 Kubernetes Pod per VM
  - >>*See KataContainers and stage1-xen*
- >**Measure Boot**
- >**System Software updates and Containers updates**
- >**Uses Disaggregation, Service Domains, and Driver Domains**

ViryaOS: Runtime





Secure Containers Runtime
Internal API
Dom0
Domain Manager
DomUs
Xen
Hardware


Secure
Containers
Runtime
Internal API
Dom0
Domain
Manager
TPM
Manager
Network
Manager
DomUs
Xen
Hardware
TPM
NIC

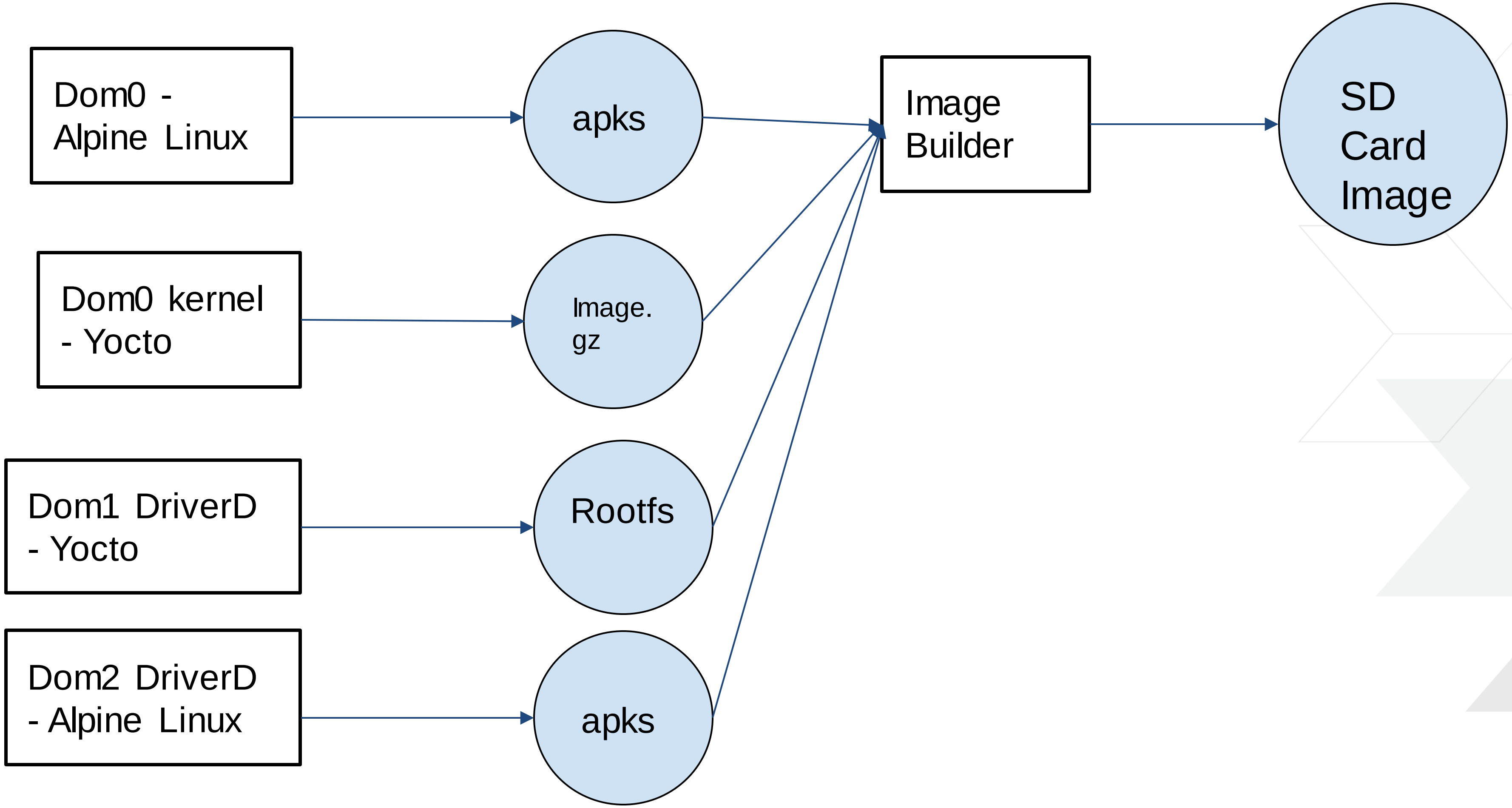
XILINX

ViryaOS: Build

- >a multi-domain build system
  - >>builds multiple domains in one go
- >Create a runnable SD Card image from multiple domain builds
- >Each domain build is independent and run in a Container
- >Pre-configures device assignments to VMs
- >Made for disaggregated architectures

XILINX.

ViryaOS: Build



ViryaOS: Build

- >**Everything builds in a Container**
- >**Support cross-builds (aarch64 on x86) with qemu-user**
- >**Support any build systems for domain builds**
  - >>Enable mixed Alpine Linux / Yocto environments
  - >>Rootfs and kernel can be built independently
- >**Support multiple DomU build output formats**
- >**The DomU build output is stored in a container**
  - >>Intermediate artifacts can be pulled from the Docker Hub to speed up the build

Status

>**Very early stage, experimental**

>**Interest, but no company backers yet, community driver**

>**Subscribe to the mailing list to learn more and participate!**

>**Initial implementation available for:**

- >>SDK
- >>Containers-driven build
- >>Yocto kernel build
- >>Imagebuilder

Adaptable.
Intelligent.

sstabellini@kernel.org
stefanos@xilinx.com

XILINX