Unikernels: General Introduction

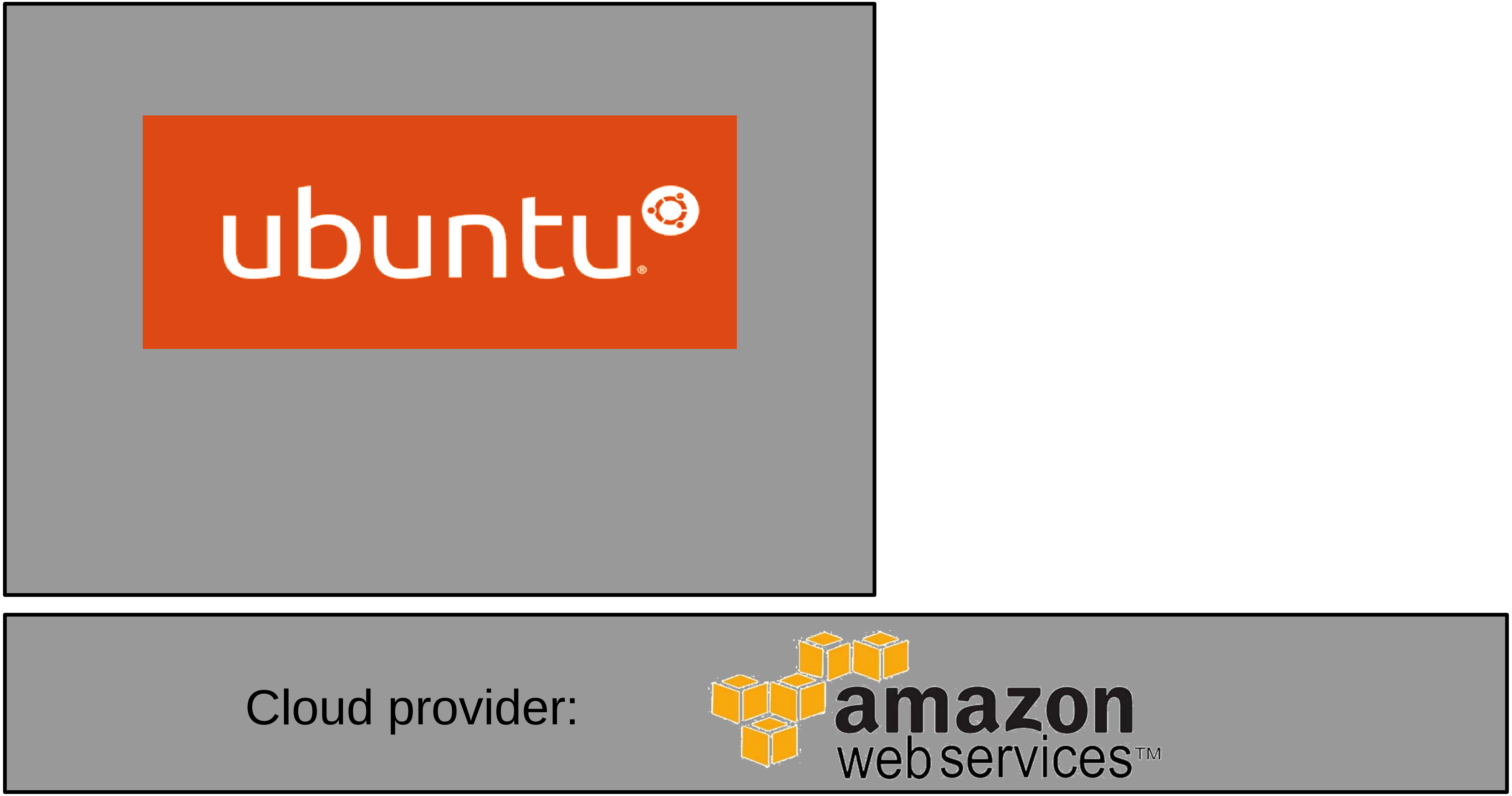
Pierre Olivier

The University of Manchester

ASPLOS 2022 Unikraft Tutorial, *March 1st*

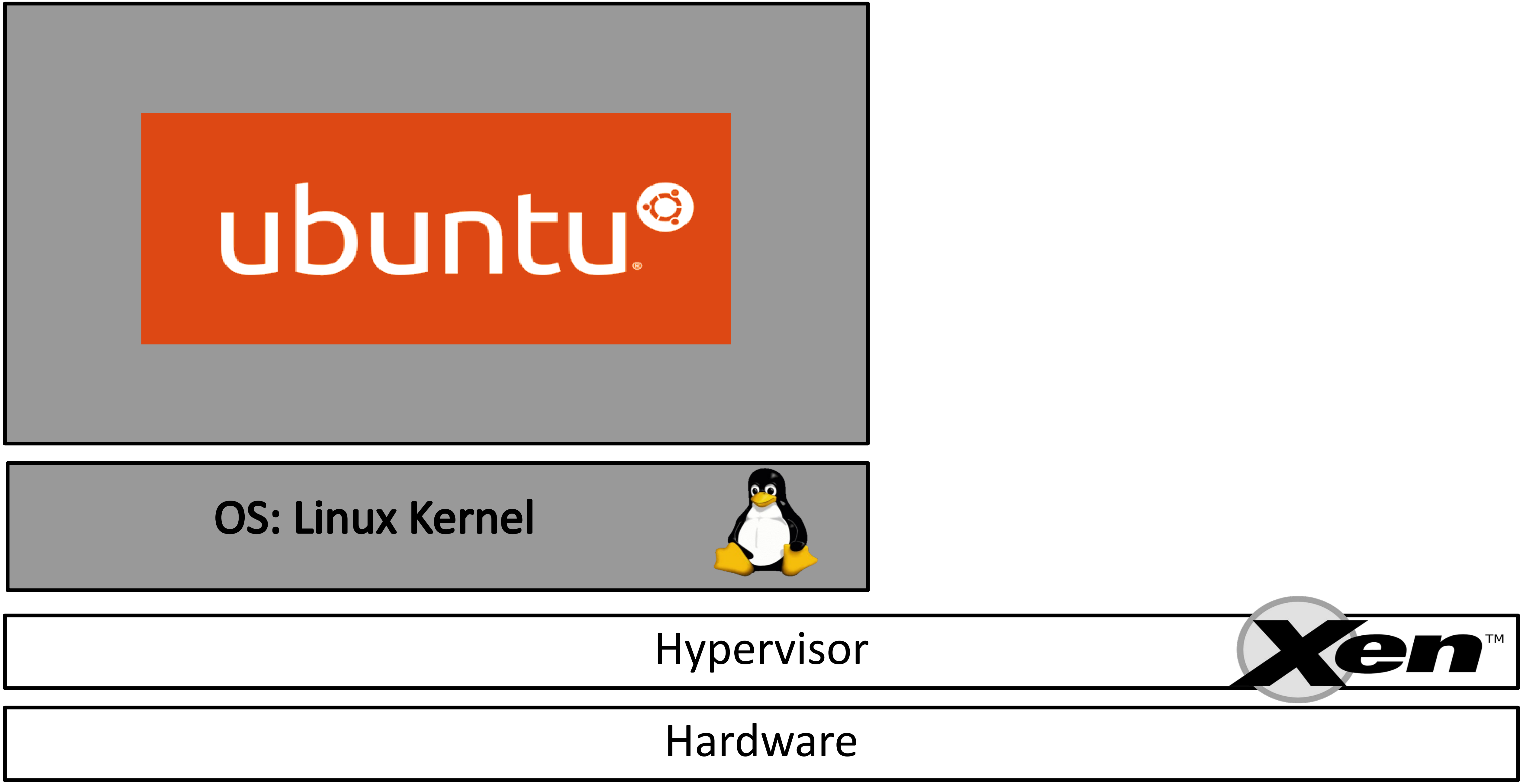
Introductory example:
my website in the cloud

Full-fledged Virtual Machine



Cloud provider:

Full-fledged Virtual Machine

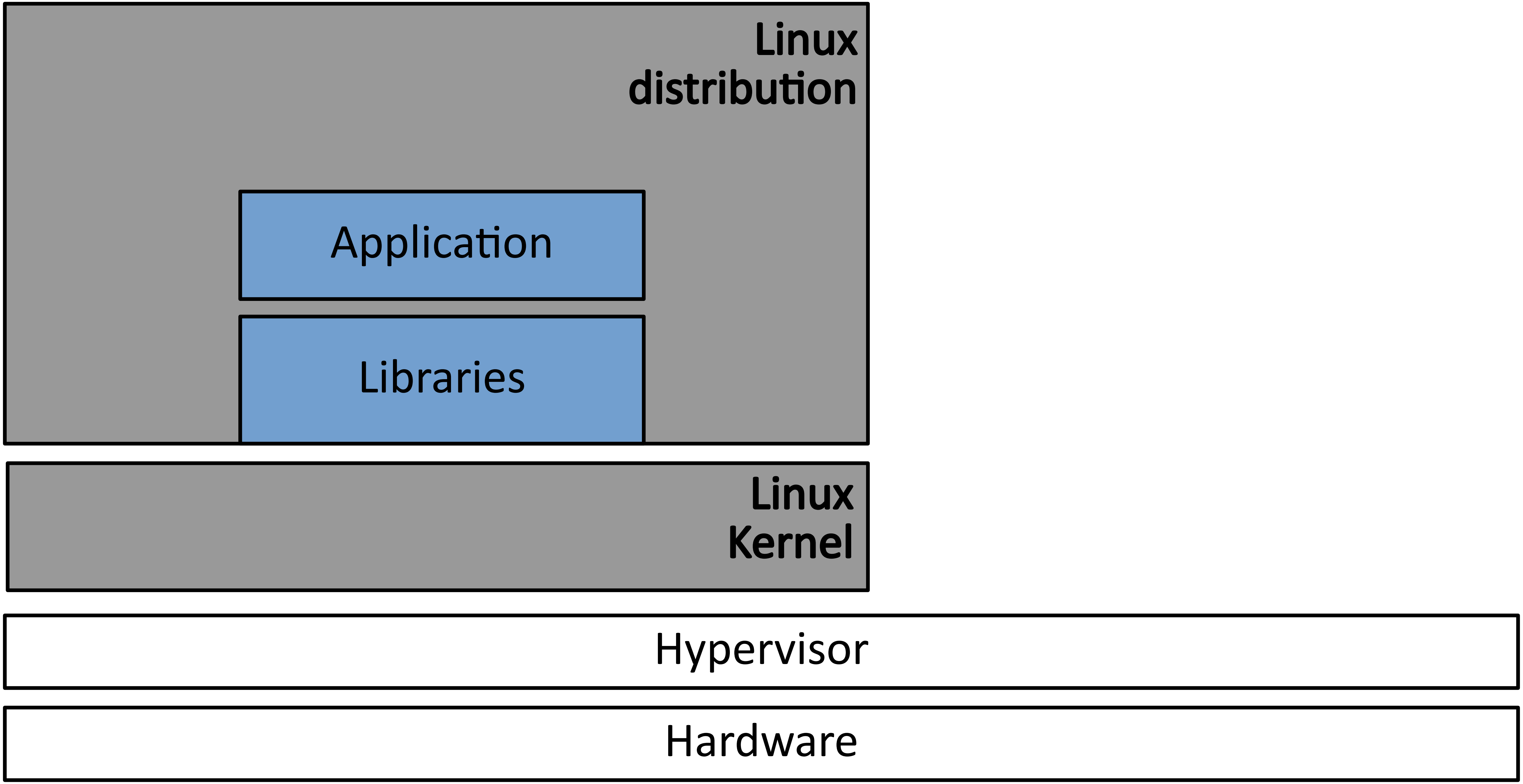


OS: Linux Kernel

Hypervisor

Hardware

Full-fledged Virtual Machine



Linux distribution

Application

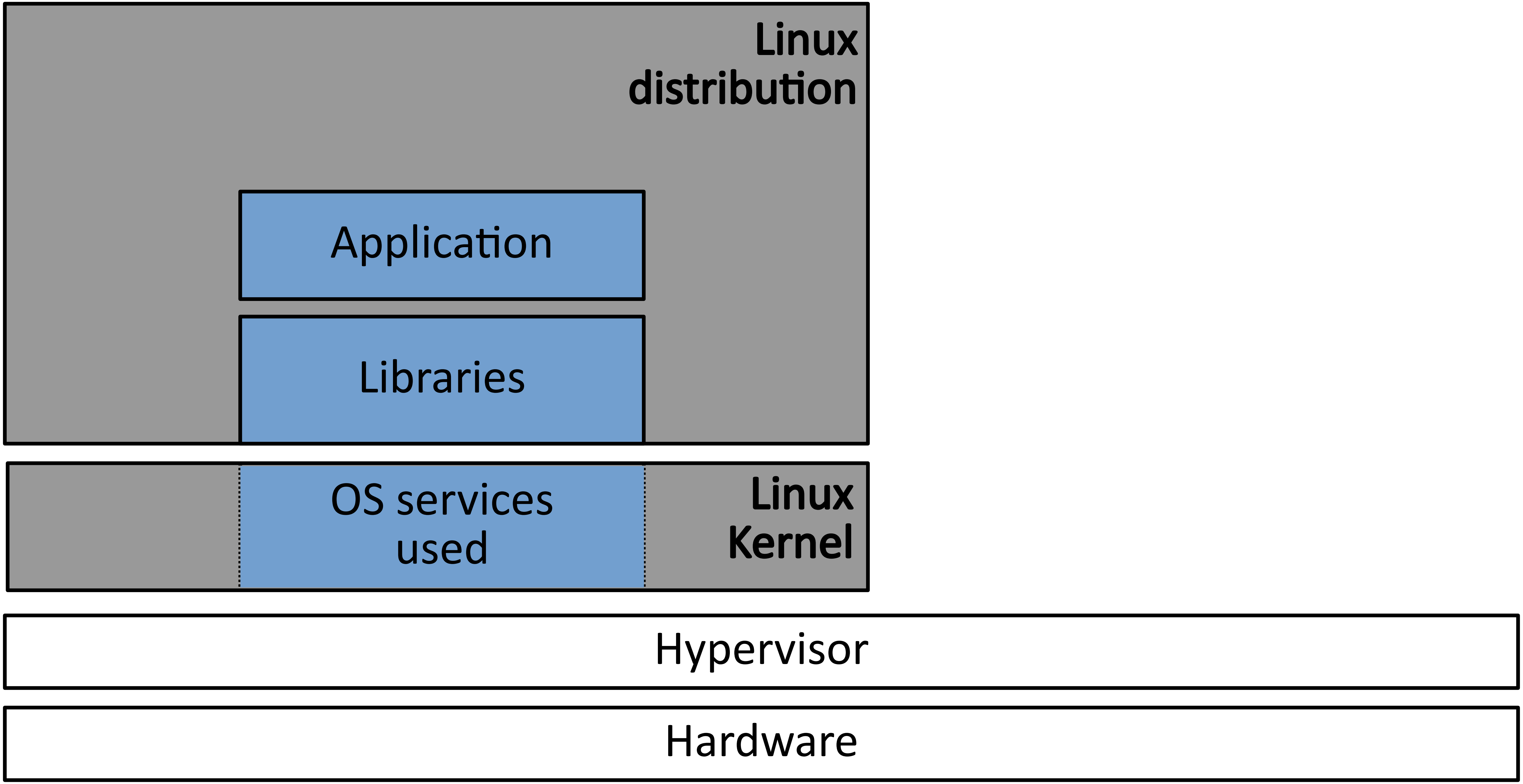
Libraries

Linux Kernel

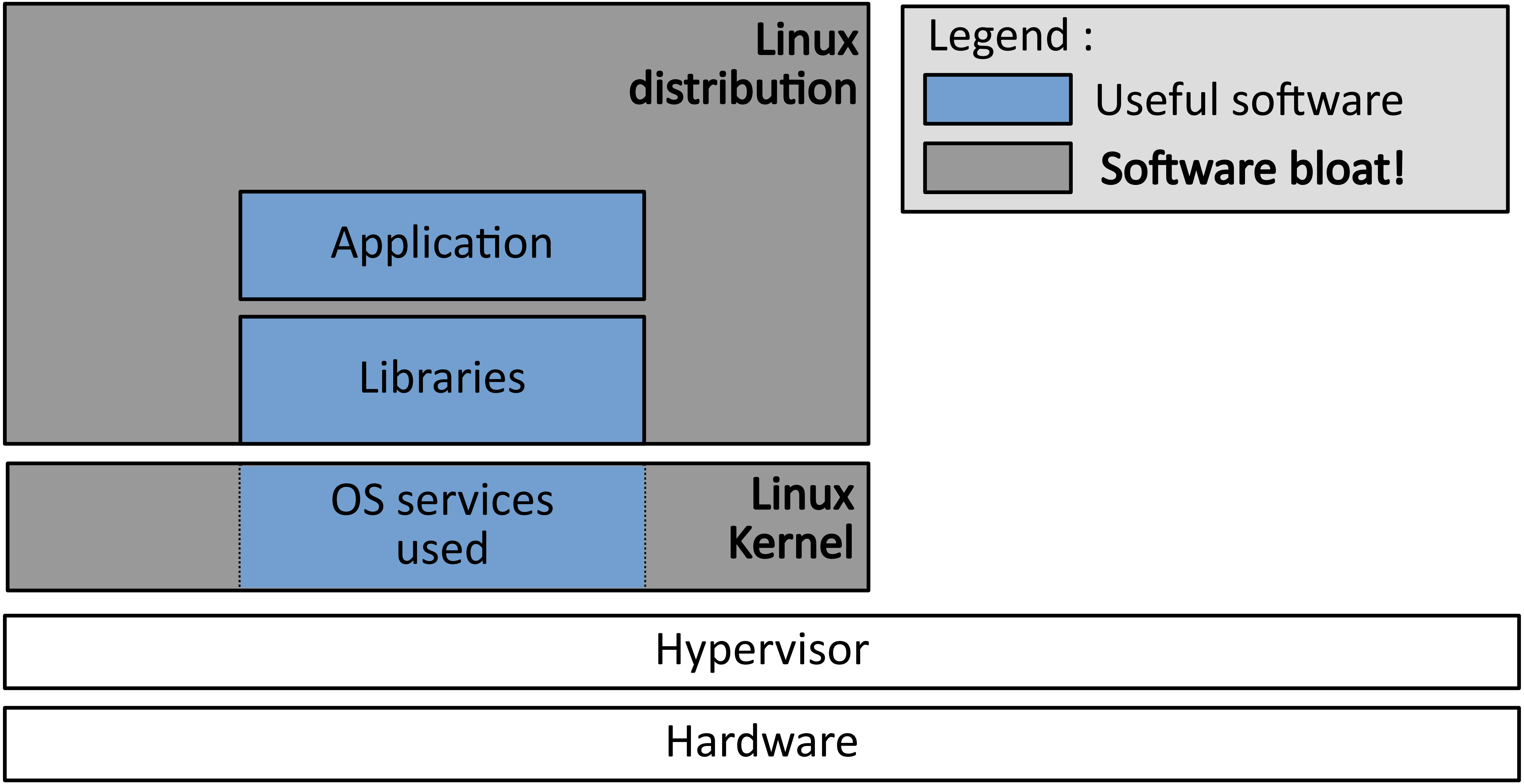
Hypervisor

Hardware

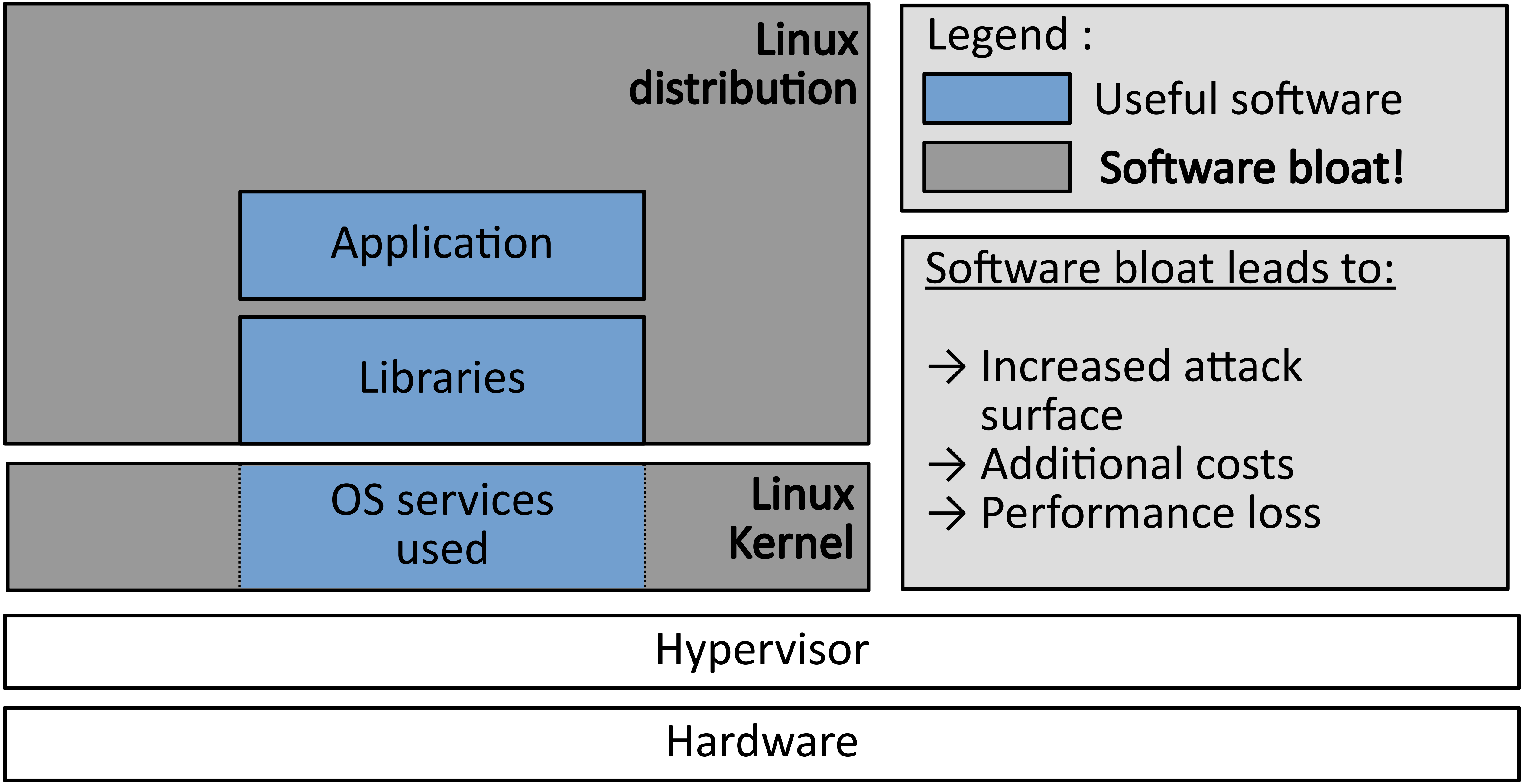
Full-fledged Virtual Machine



Full-fledged Virtual Machine



Full-fledged Virtual Machine



Linux distribution

Application

Libraries

OS services used

Linux Kernel

Hypervisor

Hardware

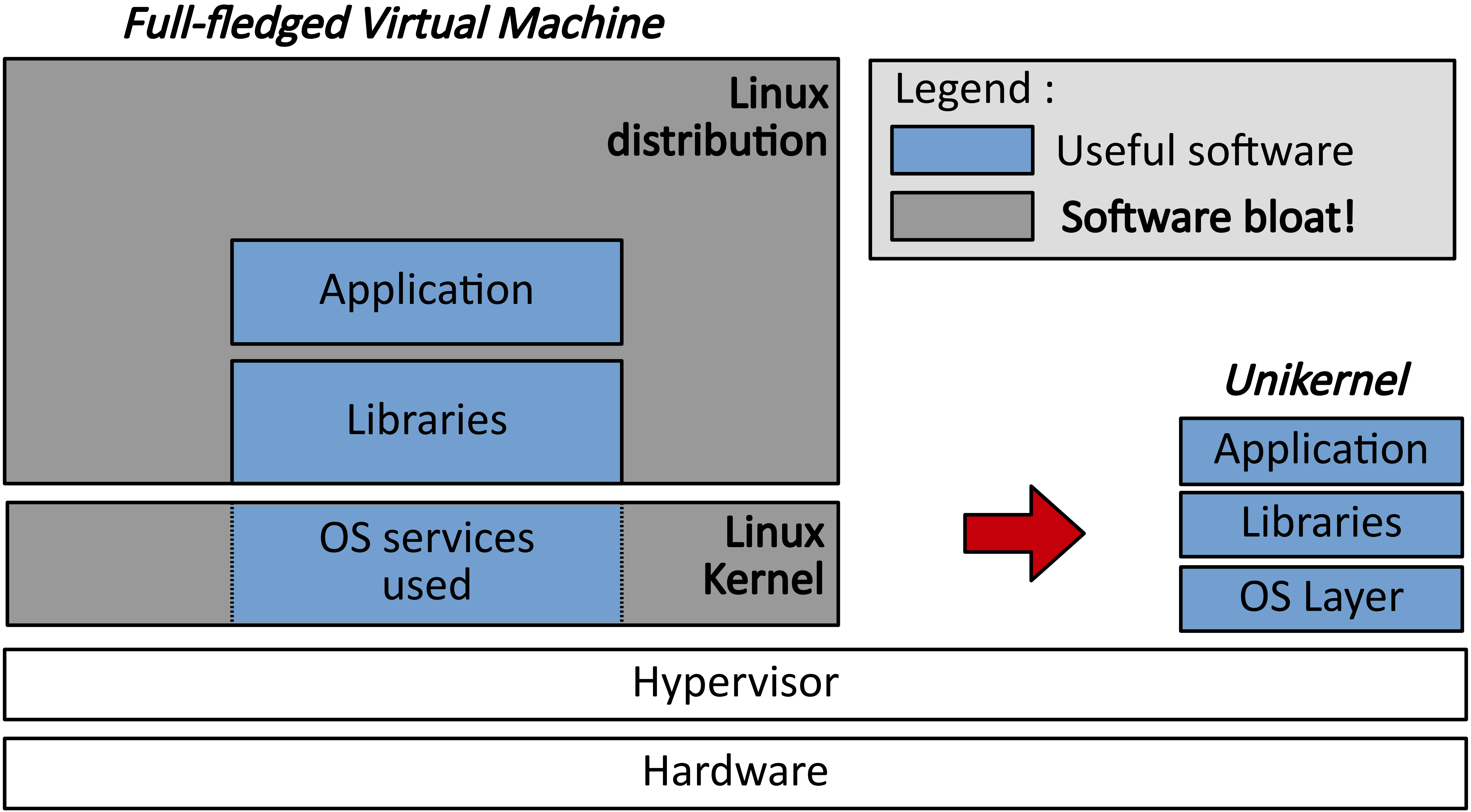
Legend :

🟦 Useful software

▨ **Software bloat!**

Software bloat leads to:

→ Increased attack surface
→ Additional costs
→ Performance loss



Full-fledged Virtual Machine

Linux distribution

Application

Libraries

OS services used

Linux Kernel

Legend :

🟦 Useful software

⬜ **Software bloat!**

Unikernel

Application

Libraries

OS Layer

Hypervisor

Hardware

Definition

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor [1]

[1] Madhavapeddy et al., “Unikernels: Library Operating Systems for the Cloud”, ASPLOS’13

Definition

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor [1]

Single-*

- **Single-purpose: run 1 application**
  - Want to run multiple applications? run multiple unikernels
- **Single-process**
  - Want to run a multi-process application? run multiple unikernels [2]
  - However, SMP (multicores) and multithreading are supported
- **Single-binary and single address space for application + kernel**
  - No kenel/user isolation, everything runs with full privileges

[1] Madhavapeddy et al., “Unikernels: Library Operating Systems for the Cloud”, ASPLOS’13

[2] Zhang et al., “KylinX: A Dynamic Library Operating System for Simplified and Efficient Cloud Virtualization, ATC’18

Benefits

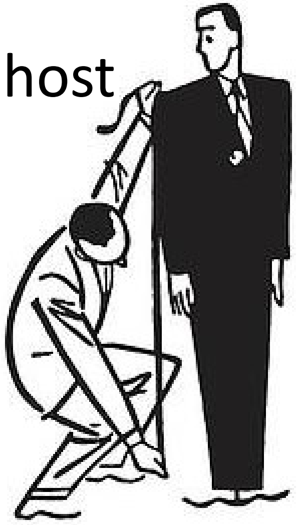
Lightweight virtualization

- Contain and run only what is absolutely necessary to the application
- Security advantage: small attack surface
- Cost advantage: memory/disk footprint reduction
- Considered as a secure alternative to containers
  - Strong inter-unikernels (i.e. VMs) isolation on a host

Benefits

Lightweight virtualization

- Contain and run only what is absolutely necessary to the application
- Security advantage: small attack surface
- Cost advantage: memory/disk footprint reduction
- Considered as a secure alternative to containers
  - Strong inter-unikernels (i.e. VMs) isolation on a host



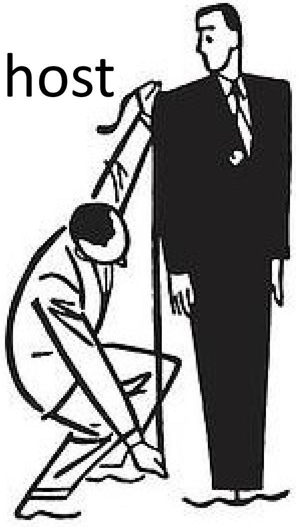
Per-application tailored kernel

- LibOS/Exokernel model
- The kernel itself contains only what is needed

Benefits

Lightweight virtualization

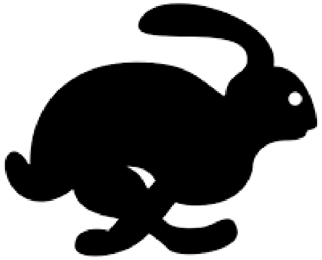
- Contain and run only what is absolutely necessary to the application
- Security advantage: small attack surface
- Cost advantage: memory/disk footprint reduction
- Considered as a secure alternative to containers
  - Strong inter-unikernels (i.e. VMs) isolation on a host



Per-application tailored kernel

- LibOS/Exokernel model
- The kernel itself contains only what is needed

Reduced OS noise, increased performance



- Sub-second boot time
- Low system call latency
  - App + kernel run with full privileges (ring 0), system calls are function calls

Application Domains

- Cloud applications: servers, micro-services, SaaS, Network Function Virtualization
- Embedded virtualization, Edge computing, IoT
- VM introspection, malware analysis, secure desktop applications
- HPC

Unikernel Models

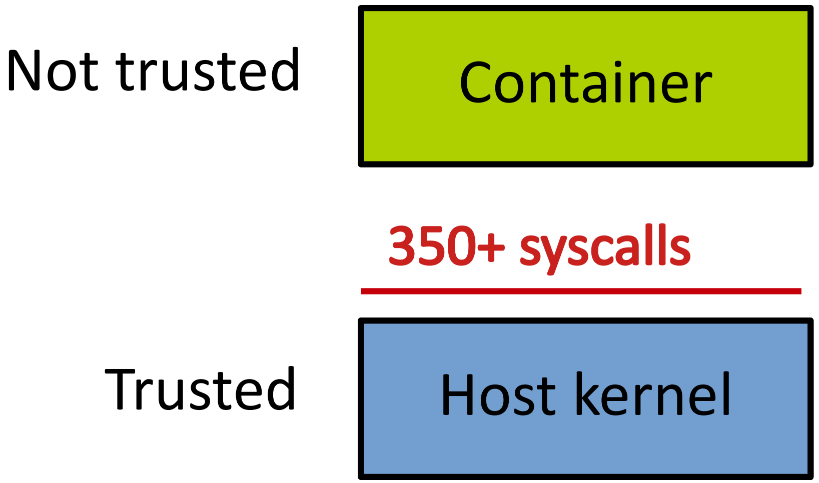
Unikernels can be classified based on the targeted language/level of compatibility for supported applications:

- *Pure memory safe languages* (OCamL, Erlang, Haskell): MirageOS [3], LING [4], HalVM [5]
- *C/C++ source-level semi-posix API*: HermitCore [6], Rumprun [7]
- *Various levels of binary-compatibility*: **Unikraft** (syscalls) [8], HermiTux (syscalls) [9], Lupine Linux (libc) [10], OSv (libc) [11]
- *Rust/Go*: RustyHermit [12], Clive [13]
- More: http://unikernel.org/projects/, https://github.com/topics/unikernel

Unikernel vs. Containers

Reduced attack surface vs. containers

- Important in multi-tenant environment (e.g. cloud) when untrusting tenants share a physical machine

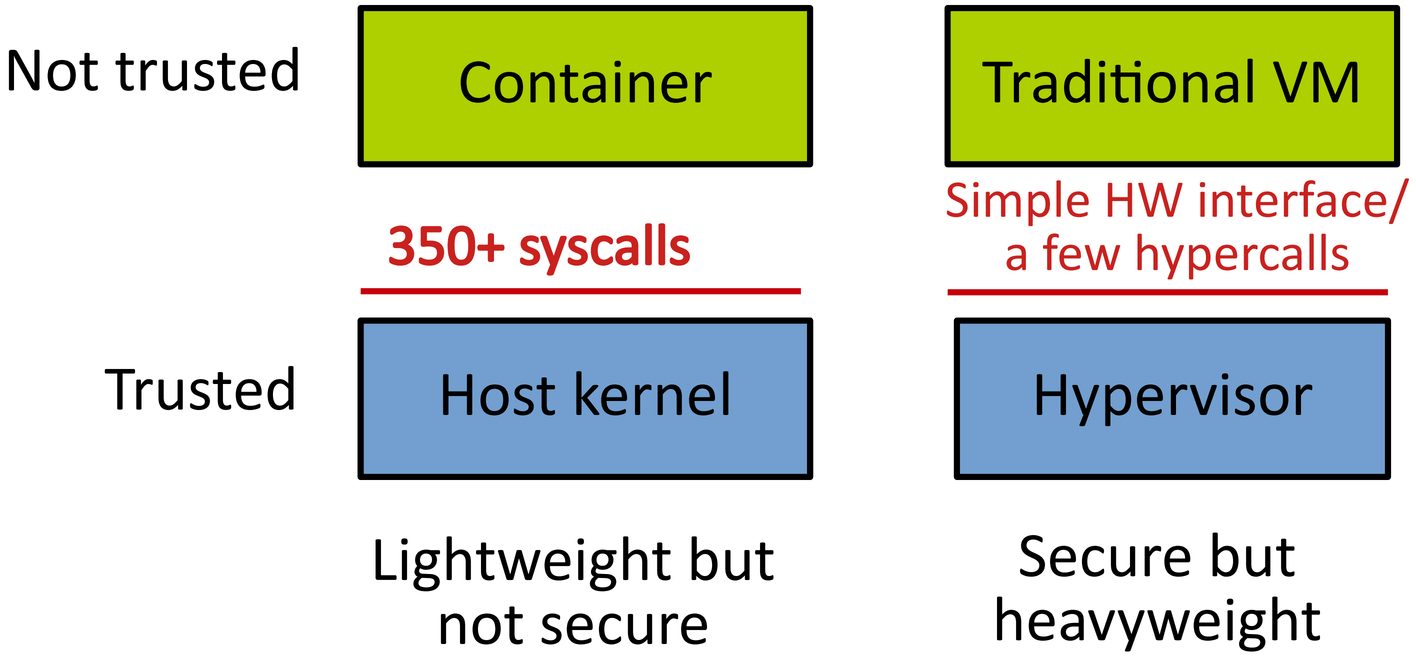


Lightweight, not secure

Unikernel vs. Containers

Reduced attack surface vs. containers

- Important in multi-tenant environment (e.g. cloud) when untrusting tenants share a physical machine



Not trusted

Trusted

Container

350+ syscalls

Host kernel

Lightweight but not secure

Traditional VM

Simple HW interface/ a few hypercalls

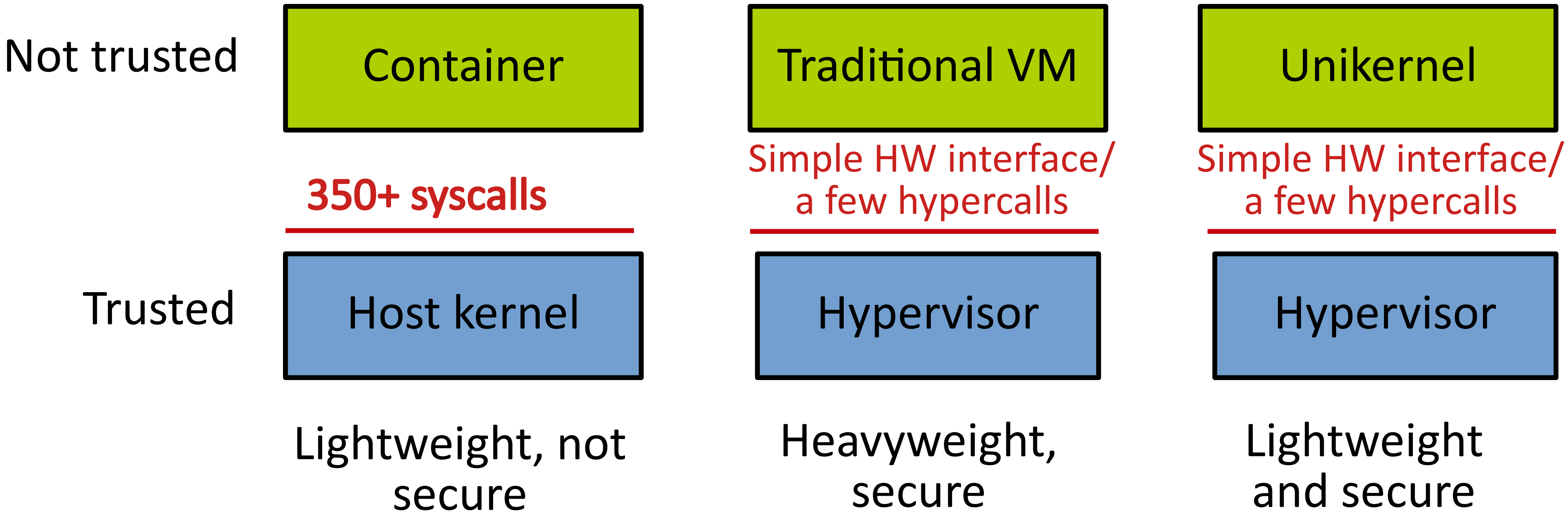
Hypervisor

Secure but heavyweight

Unikernel vs. Containers

Reduced attack surface vs. containers

- Important in multi-tenant environment (e.g. cloud) when untrusting tenants share a physical machine



Not trusted	Container	Traditional VM	Unikernel
	350+ syscalls	Simple HW interface/ a few hypercalls	Simple HW interface/ a few hypercalls
Trusted	Host kernel	Hypervisor	Hypervisor
	Lightweight, not secure	Heavyweight, secure	Lightweight and secure

Ongoing Challenges

Compatibility

- Many models require source code access
- Unsupported OS features & languages
- Burden of porting is generally on the application's programmer

Ongoing Challenges

Compatibility

- Many models require source code access
- Unsupported OS features & languages
- Burden of porting is generally on the application's programmer

Compatibility issues addressed in Unikraft through *binary-compatibility*

- Shifts the porting effort for the app. programmer into a *supporting* one placed on the kernel developers

Ongoing Challenges

Compatibility

- Many models require source code access
- Unsupported OS features & languages
- Burden of porting is generally on the application's programmer

Compatibility issues addressed in Unikraft through *binary-compatibility*

- Shifts the porting effort for the app. programmer into a *supporting* one placed on the kernel developers

Maturity: unikernels are still research prototypes and there are many bugs and standard features lacking. Most are academic projects and it’s hard to get support

- Unikraft is growing fast and has a huge community of contributors