ICA0002: IT Infrastructure Services

Virtualization

Roman Kuchin
Juri Hudolejev
2024

Virtualization

Virtualization

- Sharing the same physical resources with multiple separated environments
- Goal: hardware (or software) access separation and control

Virtualization

Virtualization

- Sharing the same physical resources with multiple separated environments
- Goal: hardware (or software) access separation and control

Emulation

- Implementation of the software or hardware behavior in another software (or hardware)
- Goal: integrating the incompatible software and/or hardware

Virtualization

Hardware virtualization

Memory virtualization

Storage virtualization

Network virtualization

Operating system virtualization

Software virtualization

Virtualization

Hardware virtualization

Memory virtualization

Storage virtualization

Network virtualization

Operating system virtualization

Software virtualization

Hardware virtualization

Also known as platform virtualization, server virtualization

Virtualization of single devices or complete hardware platforms

Goal: run full (usually unmodified) operating systems in virtual environments

Hypervisor -- software that controls virtualization

Host system -- machine that virtual environment is running on

Guest system (virtual machine) -- machine that runs **inside** virtual environment

Why hardware virtualization?

Why hardware virtualization?

More efficient resource utilization

Why hardware virtualization?

More efficient resource utilization

Cost efficiency

Why hardware virtualization?

More efficient resource utilization

Cost efficiency

Resource isolation

Why hardware virtualization?

More efficient resource utilization

Cost efficiency

Resource isolation

Faster instance provisioning

Why hardware virtualization?

More efficient resource utilization

Cost efficiency

Resource isolation

Faster instance provisioning

Easier migrations across servers/platforms

Why hardware virtualization?

More efficient resource utilization

Cost efficiency

Resource isolation

Faster instance provisioning

Easier migrations across servers/platforms

Virtual machine snapshots

Why hardware virtualization?

More efficient resource utilization

Cost efficiency

Resource isolation

Faster instance provisioning

Easier migrations across servers/platforms

Virtual machine snapshots

etc.

Hardware virtualization problems

Virtualization overhead (mostly caused by emulation)

Overprovisioning

"Noisy neighbors"

There are known issues with virtualization, but still it is very widely used today

But do we *really* need a separate OS in the virtual machine?

Typical scenarios:

- Hypervisor: RHEL, CentOS, Debian or Ubuntu
- Guest systems: RHEL, CentOS, Debian, Ubuntu, ...

Common thing: both hypervisor and guest systems running the same OS

Versions and distributions may vary, but the Linux kernel is still there

Operating system virtualization

Also known as containerization

Virtualization of operating system resources

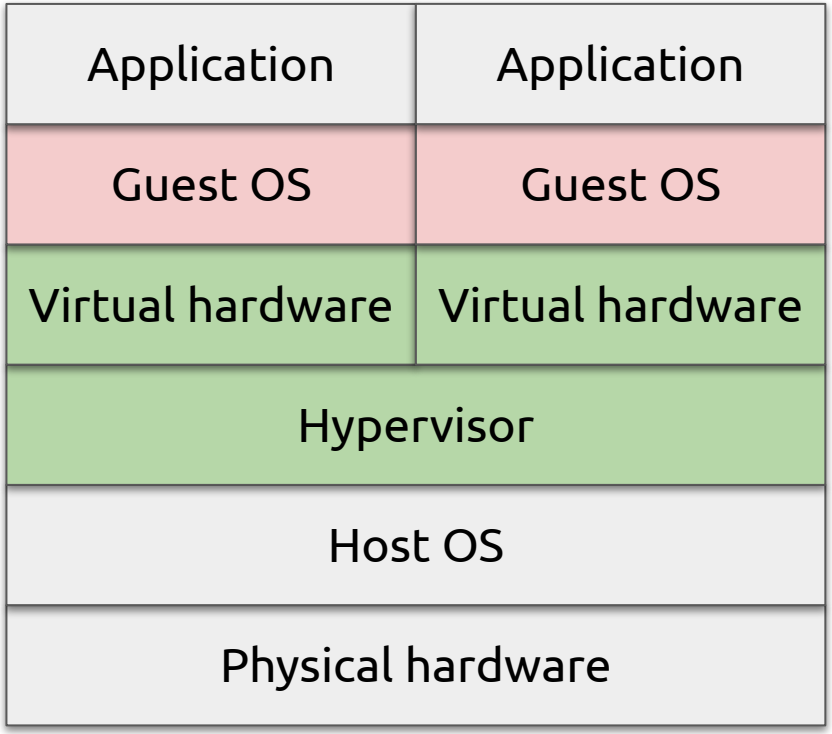
One OS kernel, multiple user spaces

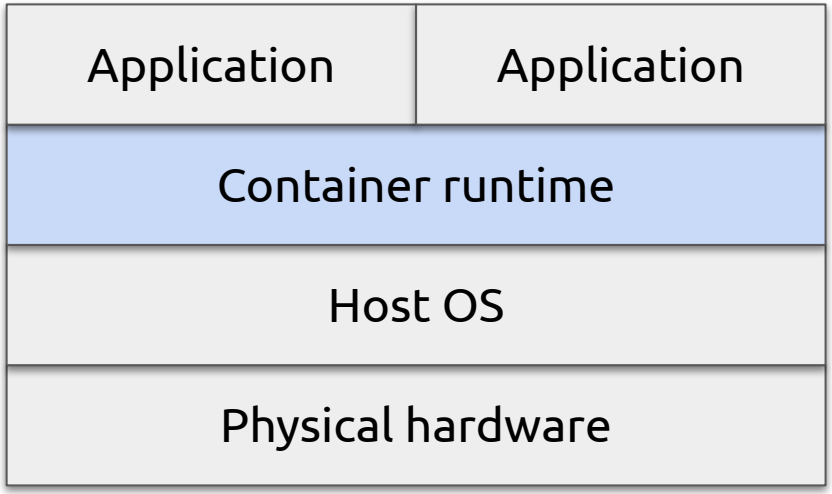
Goal: isolate individual processes or process groups

Host system -- "shared" operating system

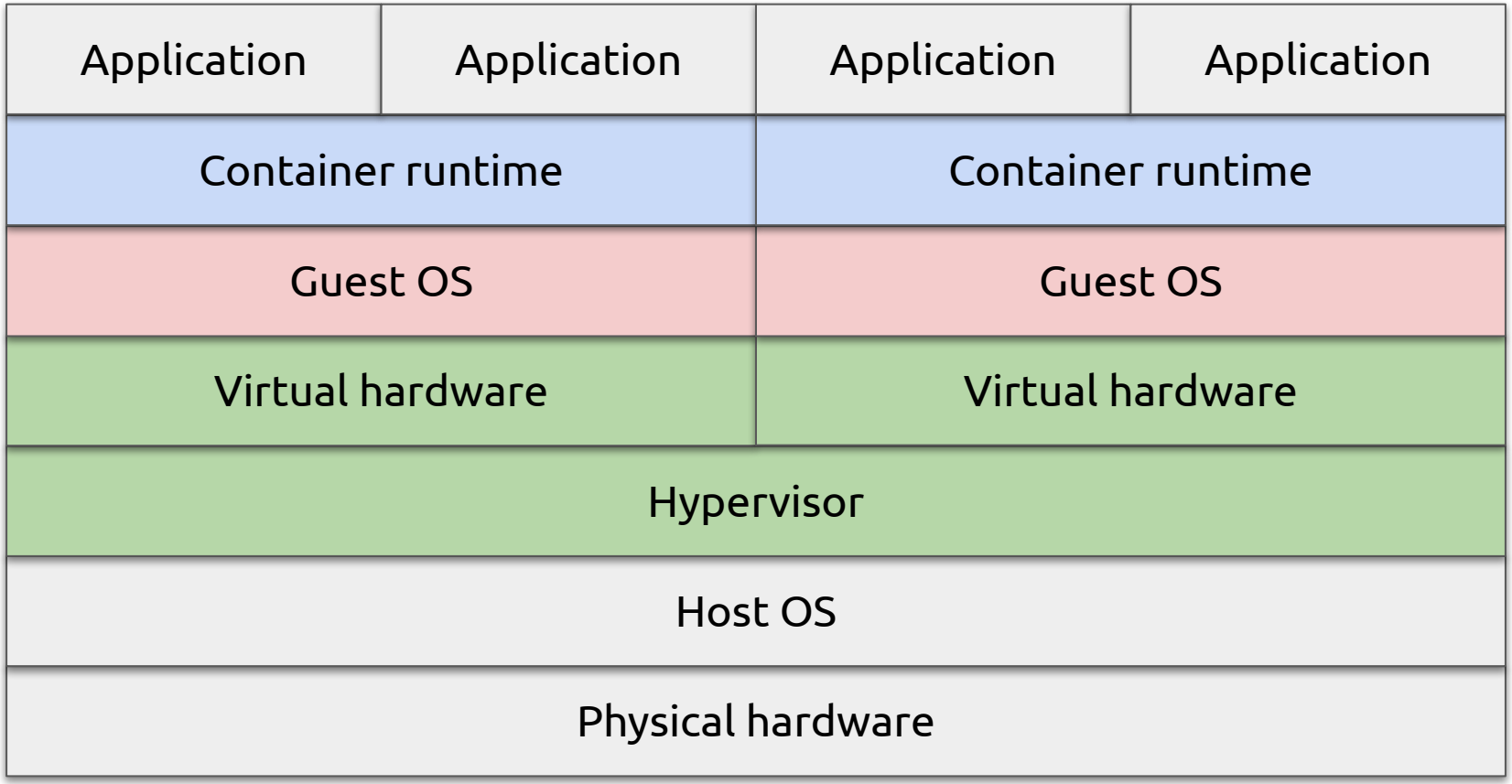
Container -- files needed to run the application (all dependencies except the OS)

Hardware vs. operating system virtualization





Both are often combined



Why operating system virtualization?

Almost all the benefits of hardware virtualization

Less virtualization overhead

But:

- Same shared OS for host and containers
- "Noisy neighbor" problems are still there, sometimes even worse

Operating system virtualization examples

chroot ("change root")

- Since 1982
- Simplest possible way to isolate processes
- Only isolates files but not CPU, memory, network etc.

Not quite a virtualization yet -- just file isolation method

Operating system virtualization examples

OpenVZ

- Since 2005
- The most advanced containerization toolset for Linux?
- Requires Linux kernel patching to enable all the features

Operating system virtualization examples

LXC (Linux Containers)

- Linux kernel namespaces: 2002 -- resource isolation
- Linux kernel cgroups (control groups): 2008 -- resource limits and priorities
- LXC: 2008
- Supports vanilla Linux kernels (utilizes cgroups and namespaces)
- Rather low-level tool

Operating system virtualization examples

Docker

- Since 2013
- Initially used LXC to utilize isolation mechanisms from Linux kernel
- Later replaced LXC with own API called **`libcontainer`**

High-level interface for containerization tools provided by Linux kernel

Almost the de-facto standard to package and run containers on Linux systems

Docker components

Daemon (**dockerd**): container manager and runtime

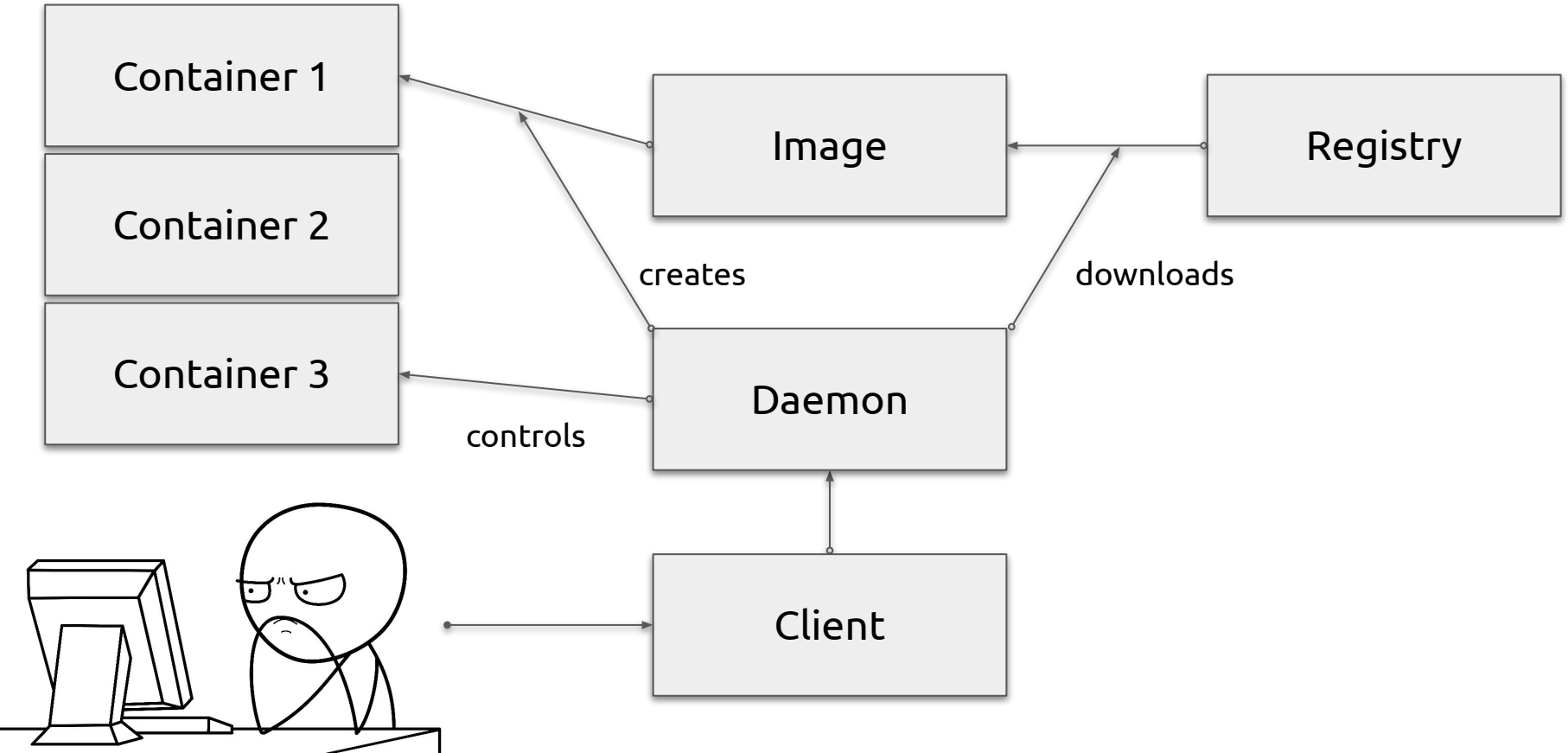
Client (**docker**): command line utility to interact with Docker daemon

Container: environment that runs application

Image: template to create Docker containers

Registry: repository of Docker images

Docker components



Demo time!

Questions?