Content

- Introduction to virtualization
- Virtualization through VM
- Virtualization through Containers

UVA HPC & BIG DATA COURSE

Virtualisation

Adam Belloum

… From ~ 1986 to ~ 2023

From mono-core to exa-scale computer

Aggregation

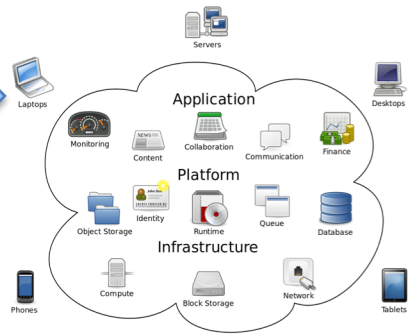
Unreliable not very fast network

central control

Virtualisation

Abstraction

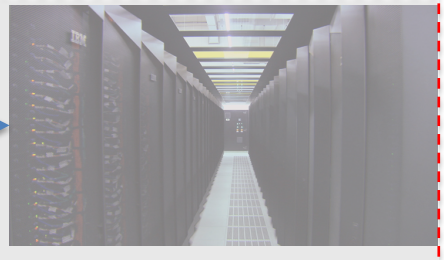
No central control



Cloud computing

Reliable fast network

central control



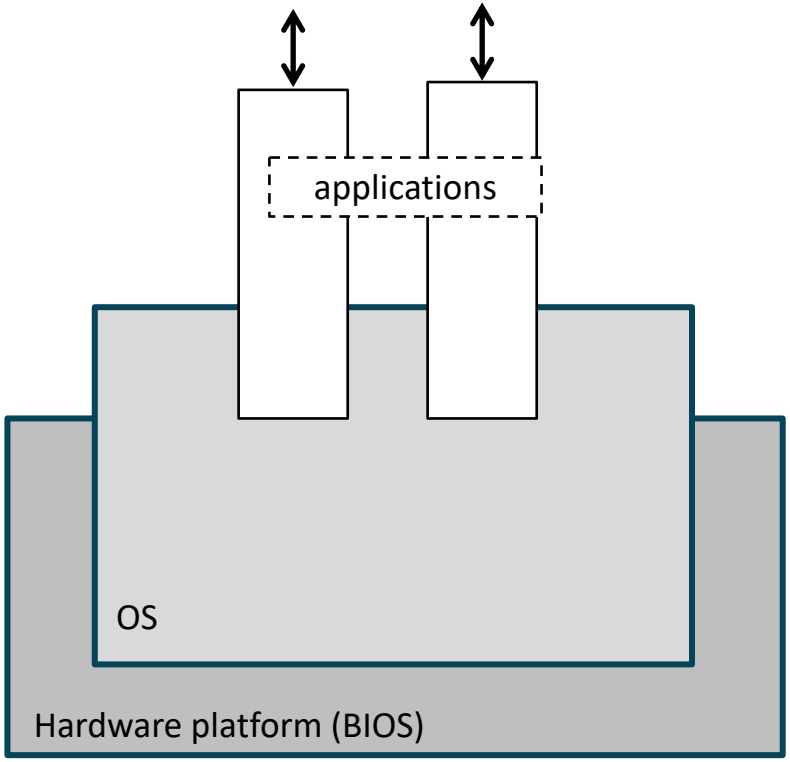
central control

Bigger systems

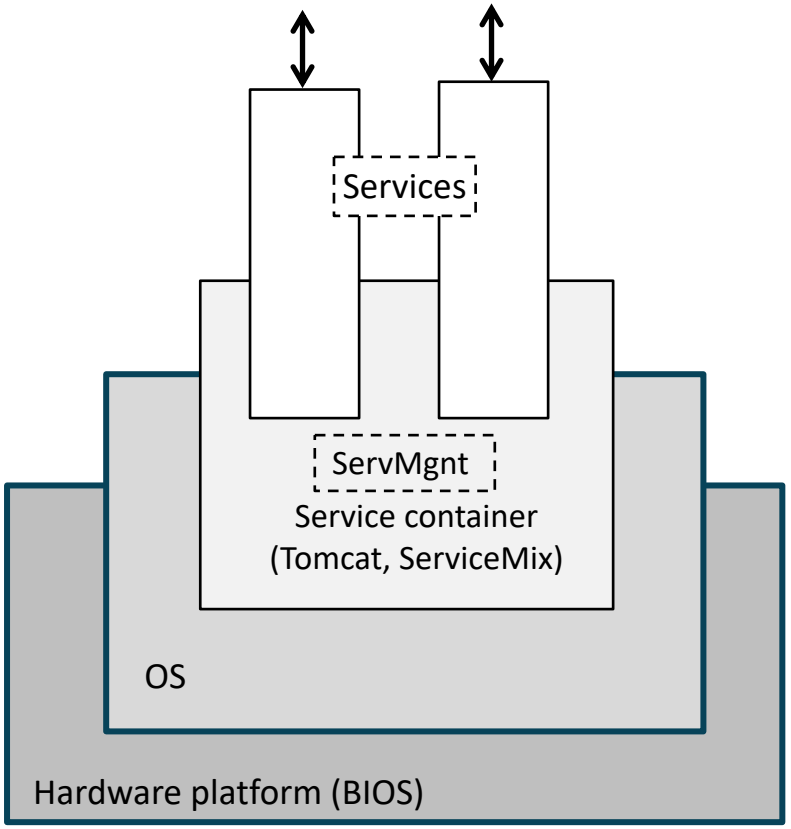
Virtualisation environment components

Service container allows for running multiple services on one computer/OS

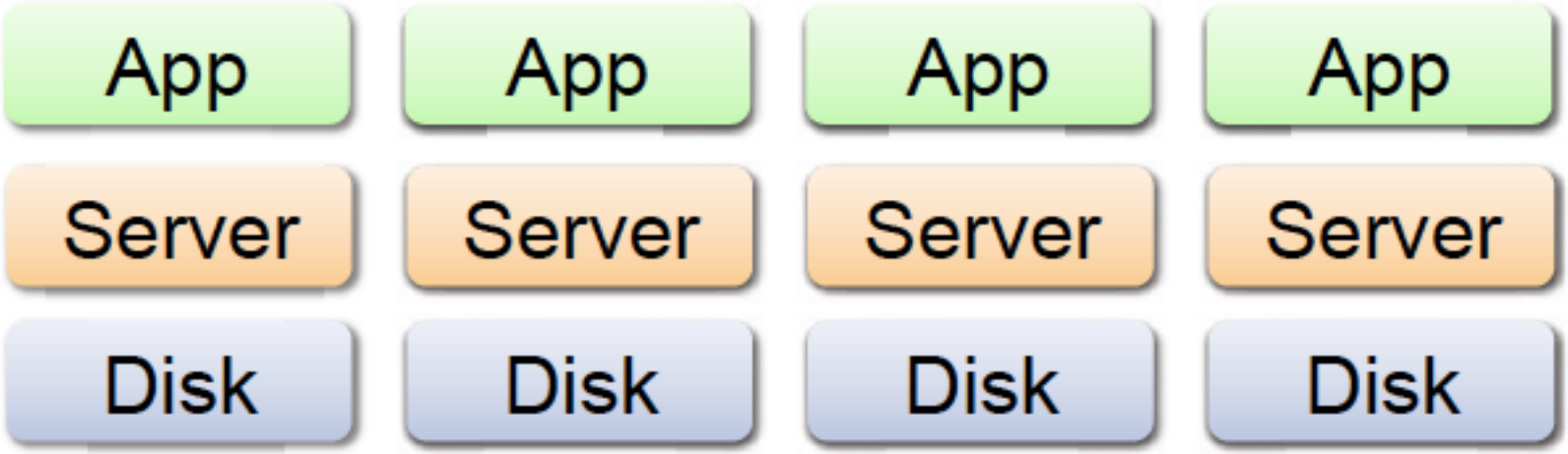
- Web Services, WSDL
- Services isolation



Web based Application and Web Services



Abstract Pool automate



Content

- Virtualization
- Virtualization through VM
- Virtualization through Containers

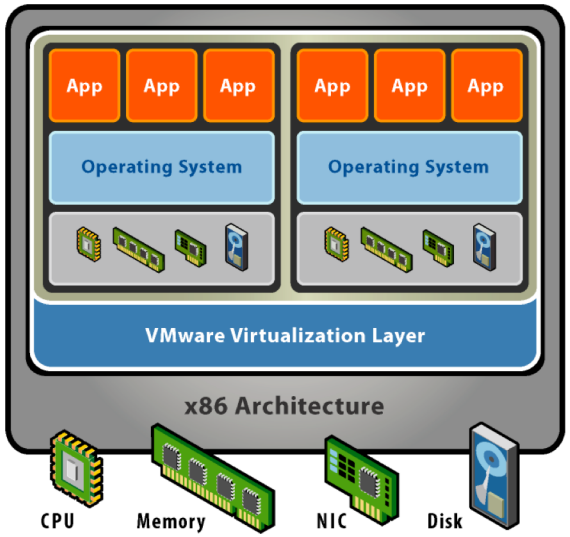
Origins of the Virtualization

- During the initial surge of interest in virtualization in the 1960s the **motivating factors** were strong.
  - Operating systems ran directly on hardware, providing services directly to applications
  - **BUT compatibility** was **a major issue** due to the number of **different architectures** being pioneered at the time

Frist attempt

- IBM developed the VM370 in the **early 70's**,
  "the limitations of the hardware of the time along with **inadequate/awkward architectures** hampered progress"

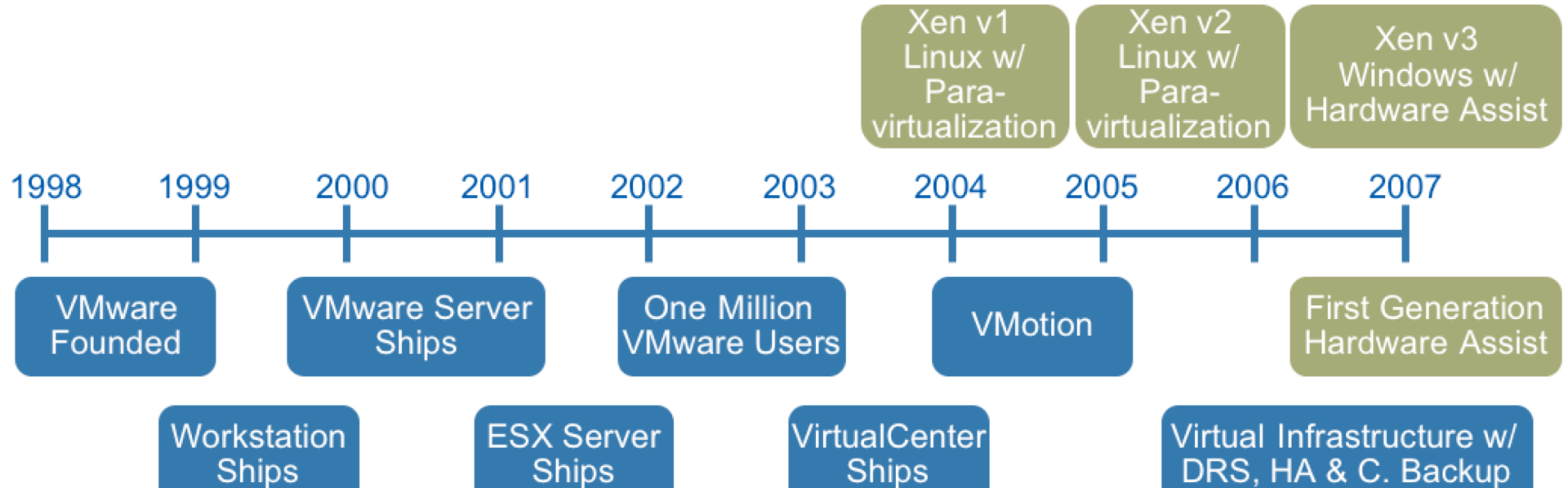
- In **1998**, VMware figured out how to virtualize the **x86 platform**, once thought to be impossible, and created the market for x86 virtualization.



An old idea: x86 hardware virtualization
http://www.os2museum.com/wp/?p=1213

Virtualization types

- **Operating system**
- **Paravirtualization** (OS assisted Virtualization)
- **Full virtualization**
- **Hardware assisted**



Virtualization is now a must

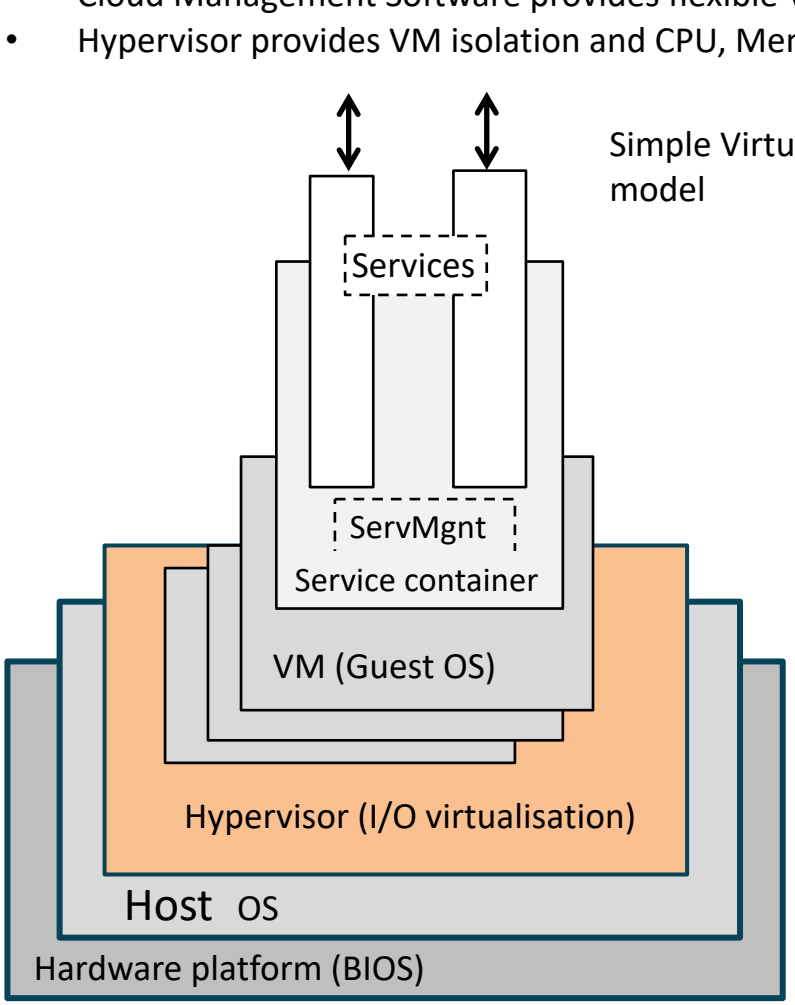
- as **data centres** and **server farm** populations grew from **hundreds** to **hundreds of thousands** of servers
  - maintaining Large collection of **small physical machines** became inefficient and expensive to run

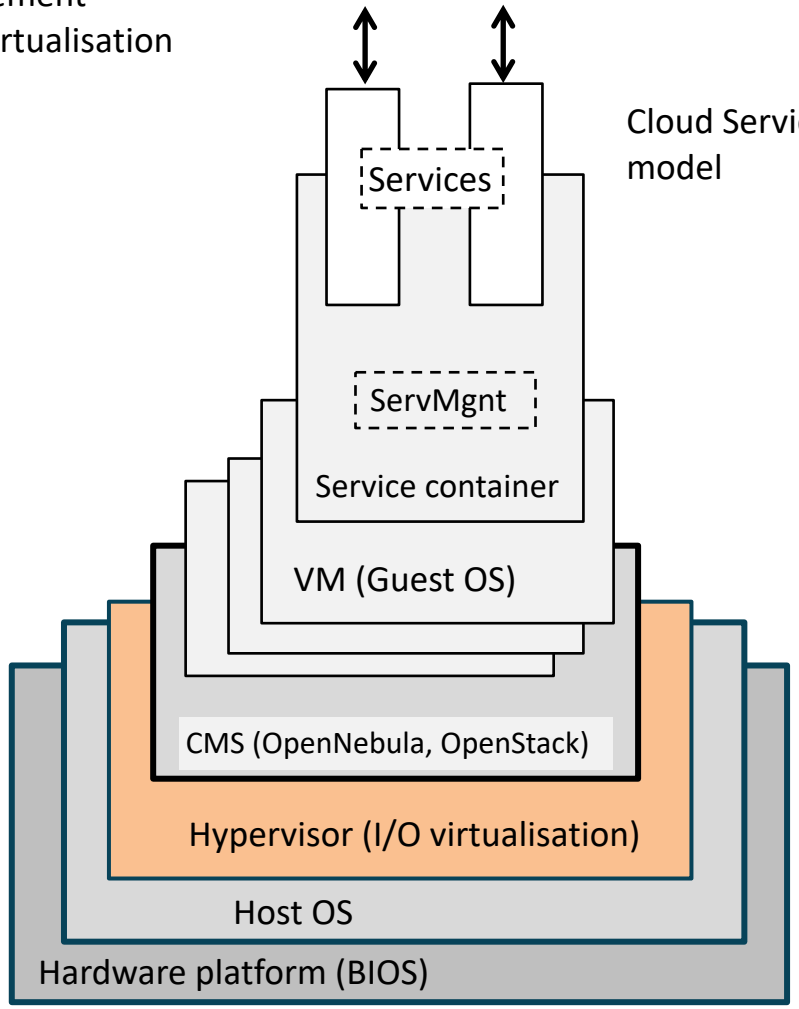
Virtualization offers the opportunity to consolidate a large number of small machines on one larger server, easing manageability and allowing resources to be effectively prioritized

Virtualisation environment components

Virtualisation and hypervisor allows for running multiple OS on one computer/OS

- Cloud Management Software provides flexible VM management
- Hypervisor provides VM isolation and CPU, Memory, I/O virtualisation





Content

- Virtualization
- Virtualization through VM
- Virtualization through Containers

Containers from Scratch

containers are just **isolated groups of processe(s)** running on a single host. That isolation leverages several underlying technologies built into the Linux kernel:

- **namespaces**
- **cgroups**

prominent advantages of containers

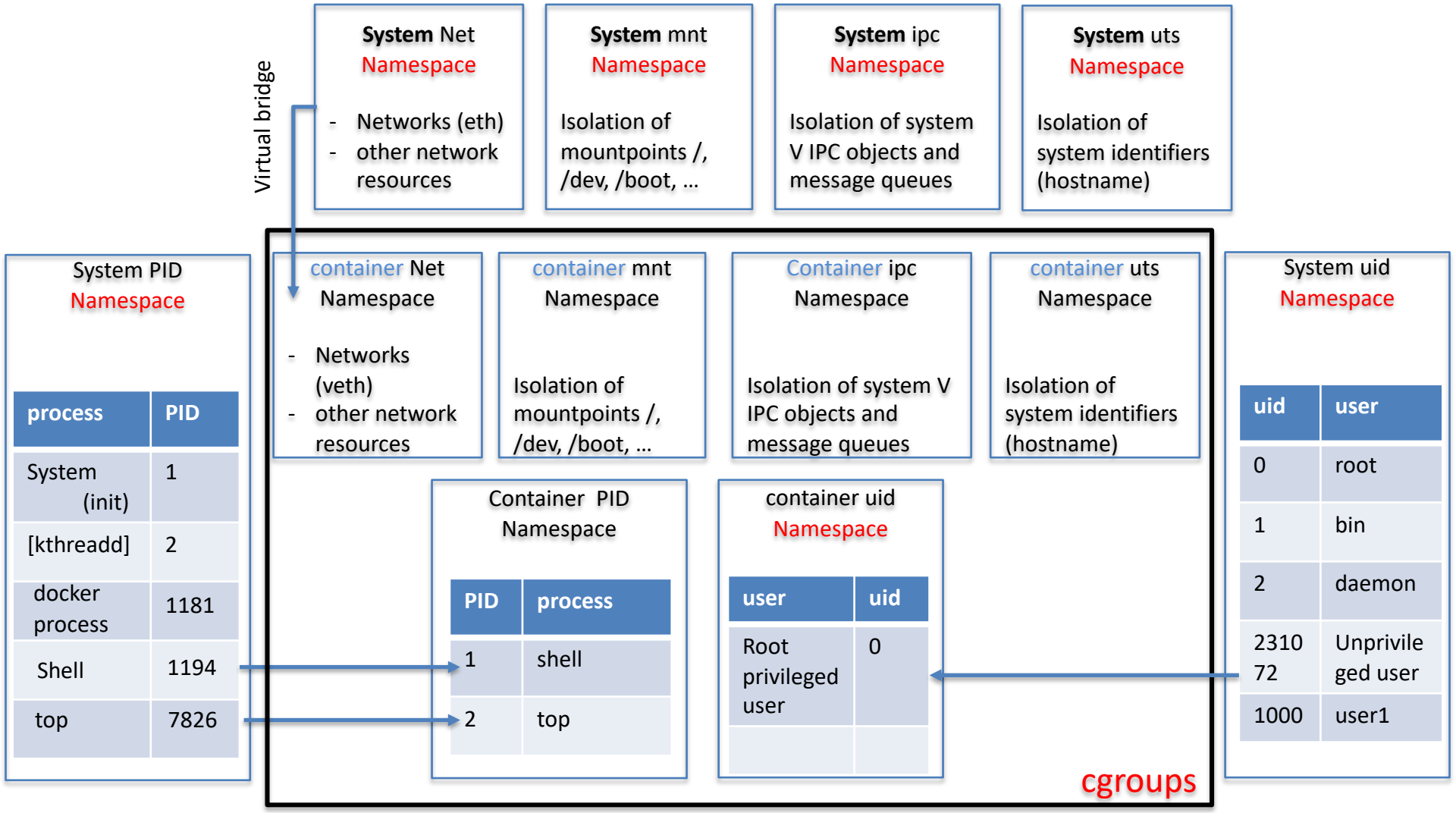
- Flexibility
- Convenience
- Consistent
- Reproducibility

Source: Eric Chiang “Containers from Scratch”
https://ericchiang.github.io/post/containers-from-scratch/

Linux Kernel Support

- cgroups: limit how much resource the process can use
  - CPU
  - Memory
  - Network
- namespaces: limit what the process can see
  - pid
  - net
  - mnt
  - ipc

Process of creating a container



Container vs Container-image

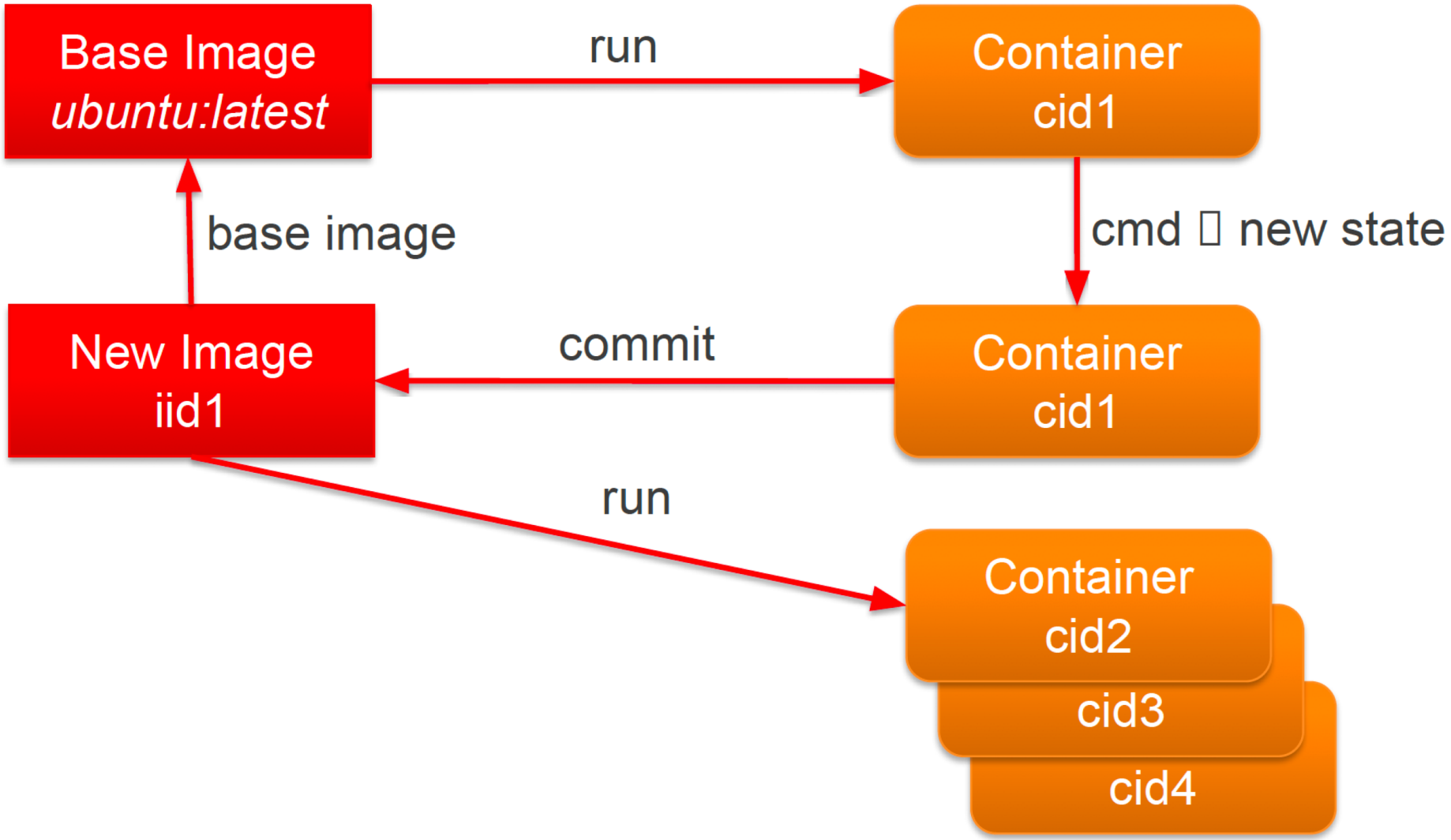
Container-Image

- Binary representation of the container stored on disk
- image layering: Parent child relationships
  - Image can be created starting from an existing image
  - The tree structure of image helps to fix vulnerability
- Images can be created/built from "configuration file" (dockerfile)
  - but also from running containers (save container instance

Container

- a running instance of Container-Image
- packaged with its dependencies
  - Nothing is installed on the host multiple container with conflicting libs can run on the same host.
  - When the container stops everything go to clean state
- The lifecycle of a process running into the container is tight to the lifecycle of the container
  - process state

Container vs Container-image



Terminology

Container Image

- Persisted snapshot that can be run
  - images: List all local images
  - `run`: Create a container from an image & execute a command in it
  - `tag`: Tag an image
  - `pull`: Download image from repository
  - `rmi`: Delete a local image
    - This will also remove intermediate images if no longer used

Container

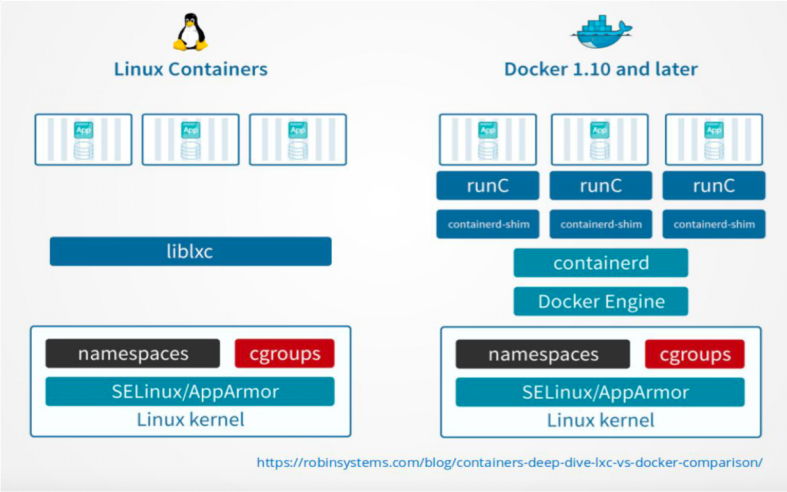
- Runnable instance of an image
  - `ps`: List all running containers
  - `ps –a`: List all container (incl. stopped)
  - `top`: Display processes of a container
  - `start`: Start a stopped container
  - `stop`: Stop a running container
  - `pause`: Pause all processes within a container
  - `rm`: Delete a container
  - `commit`: Create an image from a container

Example of container technology Docker

- …
- 2008: LXC (linux Container)
- 2013: Docker ➔ build on LXC
- 2016: Docker 1.10 ➔ runc and containerd

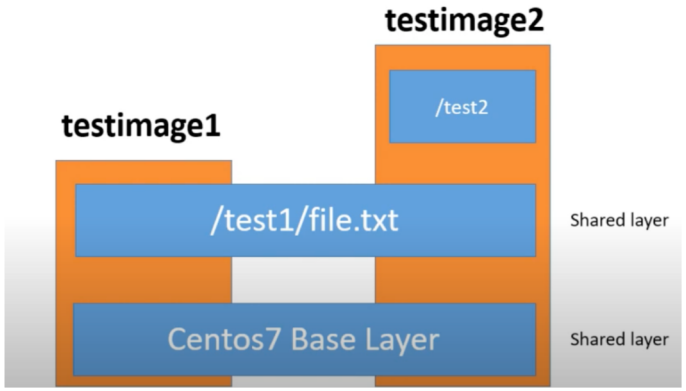
Docker image

- is a the union of one more read-only layers
- Created following instruction defined in Dockerfile
- Cache option to share layers.
- Uses volumes to store data outside the container
- Default docker storage /var/lib/docker/

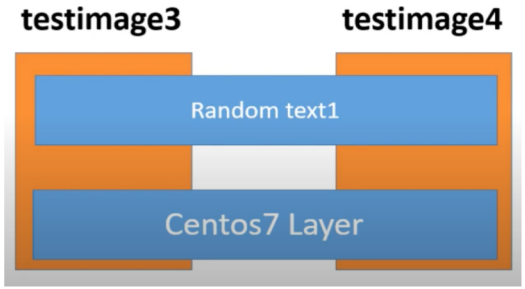


Sharing layers across containers

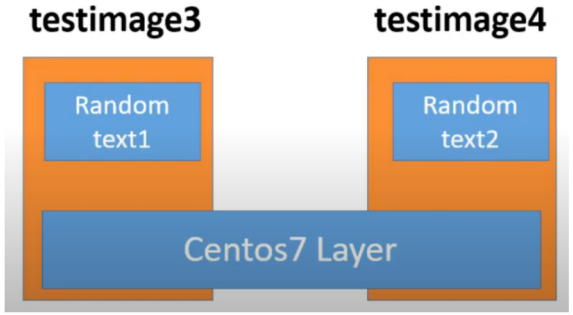
(a)



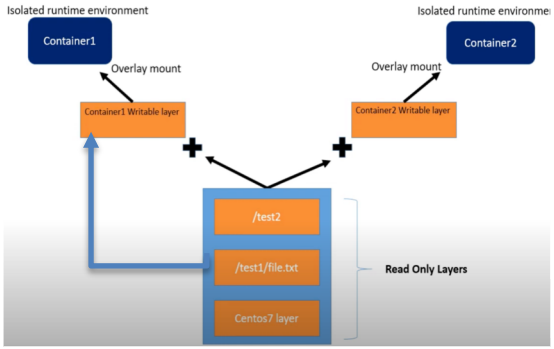
(b)



(c)



(d)

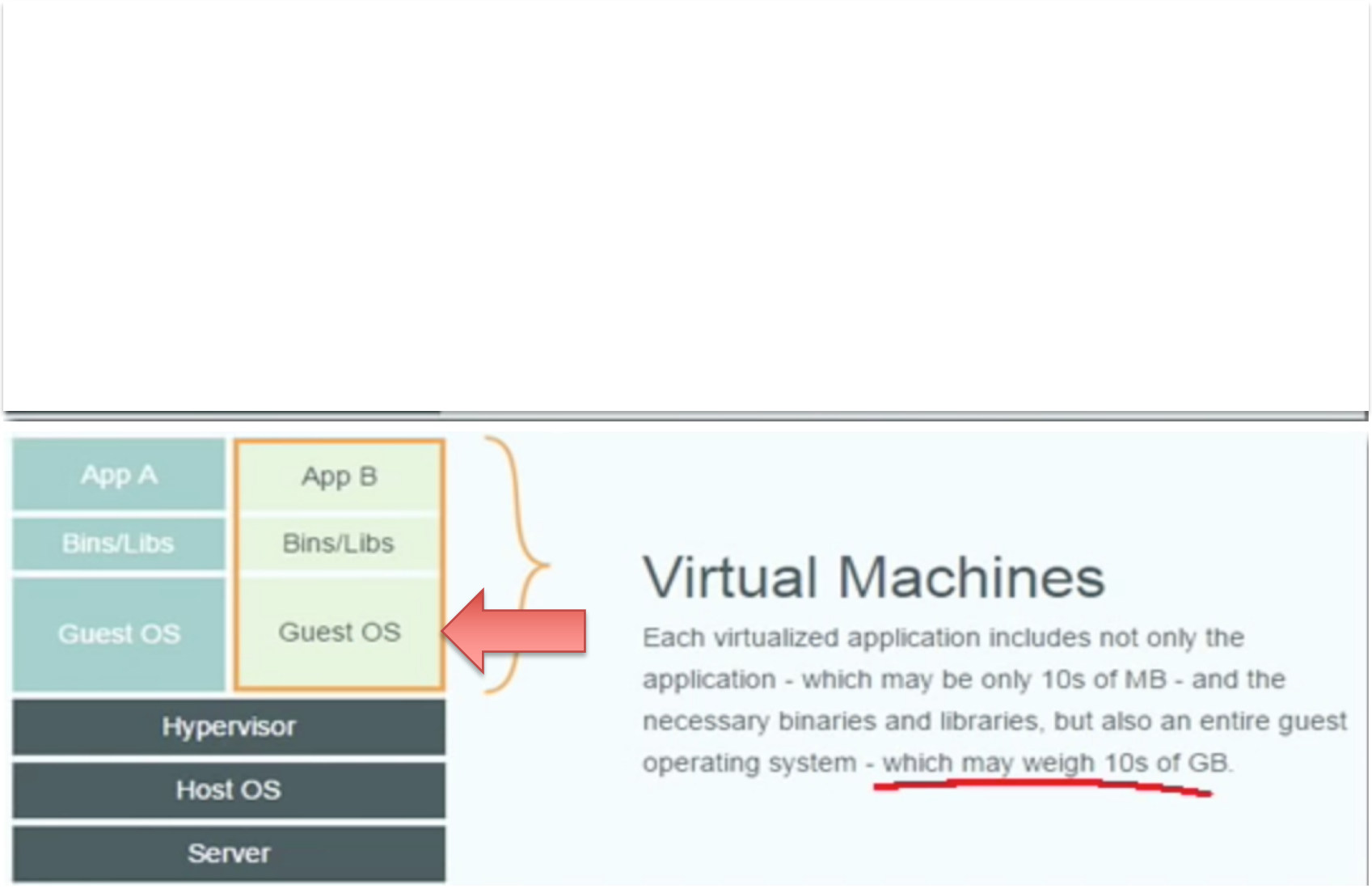


(e)

Content

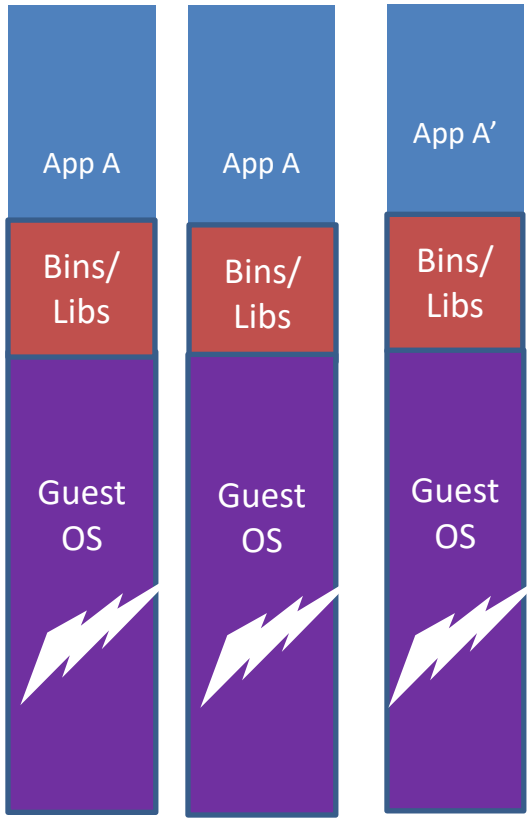
- Virtualization
- Virtualization through VM
- Virtualization through Containers
- VM vs Contianers

VM vs containers



Why are containers lightweight?

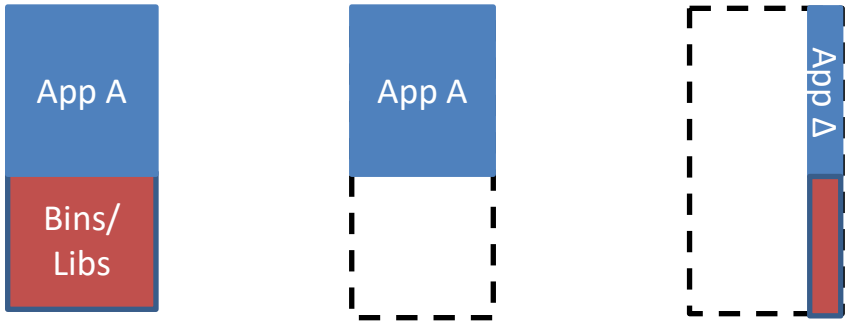
VMs



VMs

Every app, every copy of an app, and every slight modification of the app requires a new virtual server

Containers



Original App

(No OS to take up space, resources, or require restart)

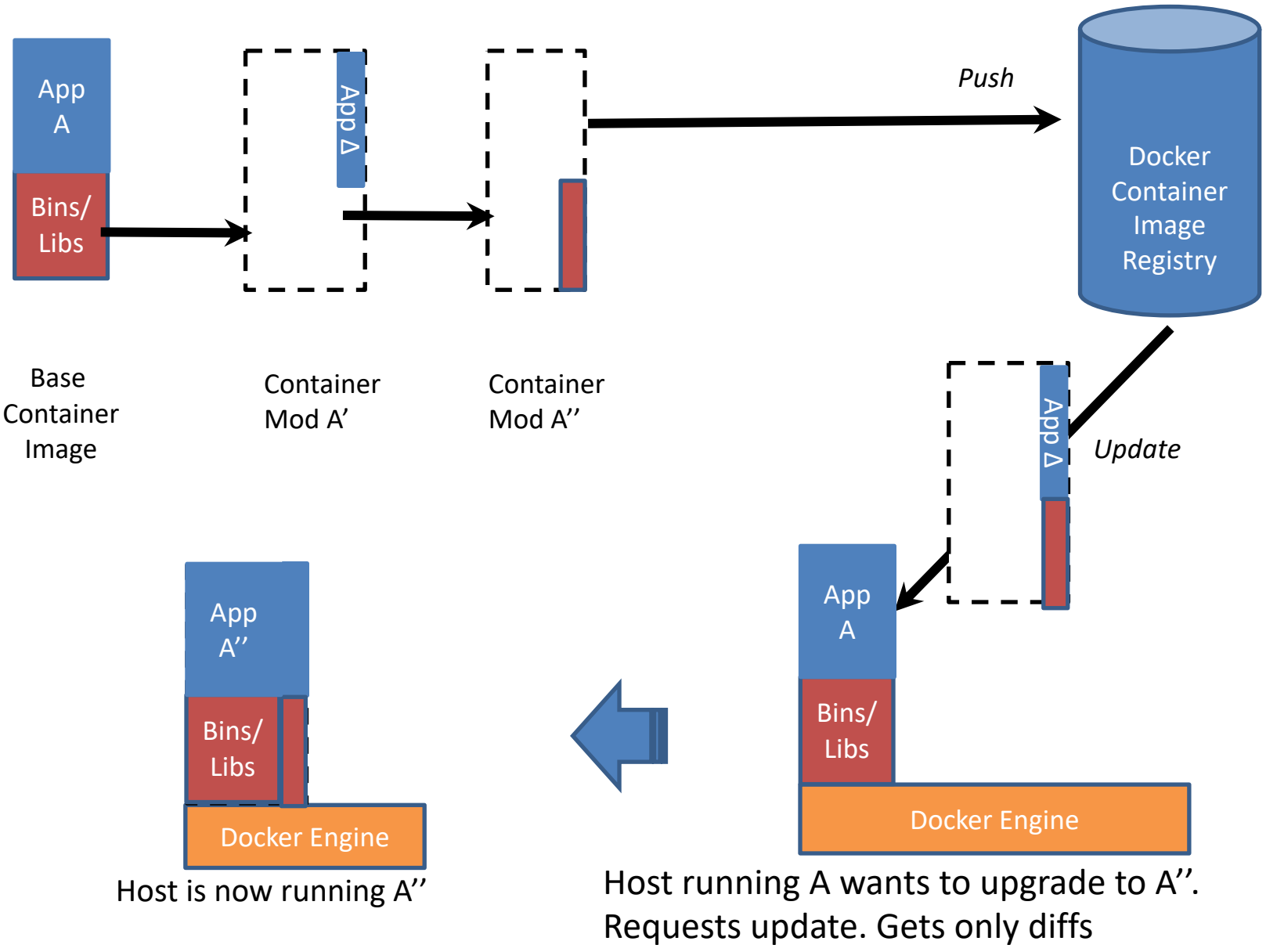
Copy of App

No OS. Can Share bins/libs

Modified App

Copy on write capabilities allow us to **only save** the **diffs** Between container A & container A'

Changes and Updates



Containers lifecycle management tools

1. Containerization
2. Discovery and Global Configuration Stores
3. Networking Tools
4. Scheduling, Cluster Management, and Orchestration

Reference: The Docker Ecosystem: An Introduction to Common Components
https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-an-introduction-to-common-components

Container technologies

- Docker [1]
- Singularity [2]
- Charliecloud[3]
- Shifter[4]
- LXC, OpenVZ, uDocker …

(1) https://docker.org
(2) https://singularity.lbl.gov
(3) https://charliecloud.readthedocs.io/en/latest/
(4) https://www.nersc.gov/research-and-development/user-defined-images/