Already have virtualization, why containerize?

Andy Lee

Table of Content

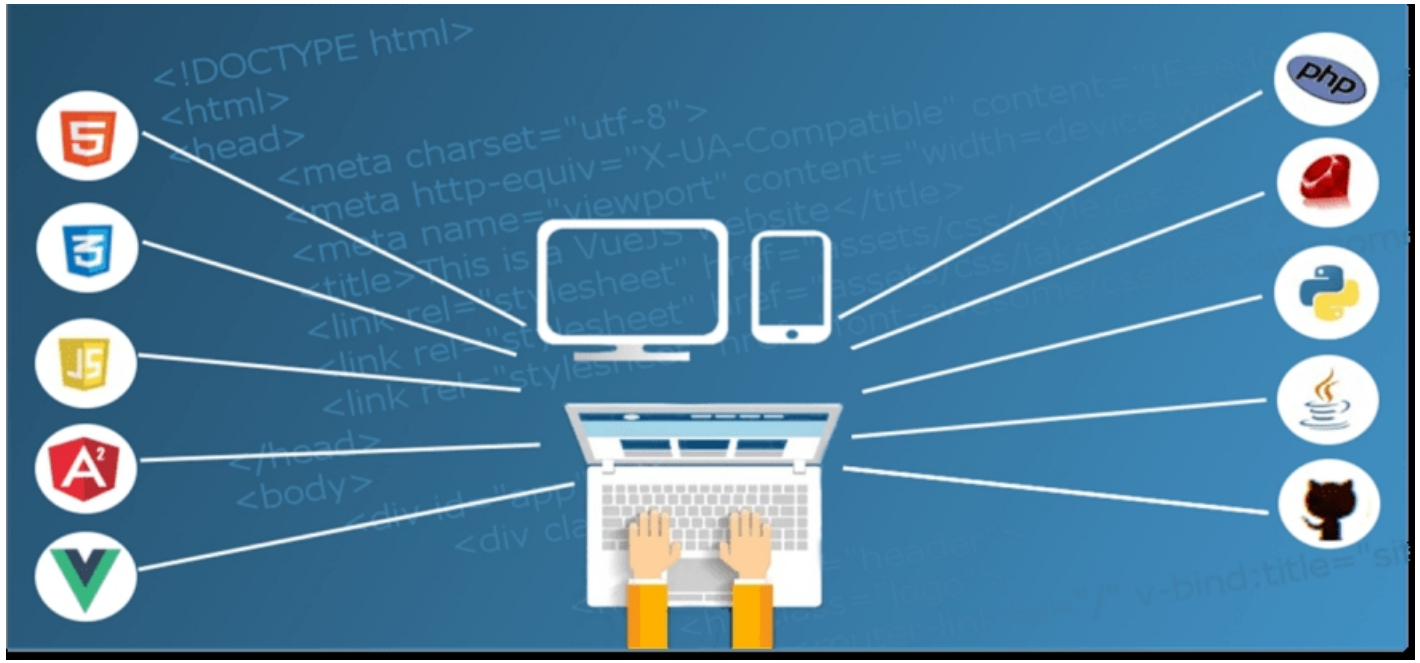
House Rules

- Don’t just sit back
- Correct me at anytime for the imperfect presentation which came from half a day
- Relate topic to situation you might have already encoutnerd while working with VM vs Docker
- It’s easy to forget to mention someting or make some connection - please help me fill any gaps

What is the problem?

- Have you ever struggled to get your developer environment set up as a new user on an existing project?
- Have you ever struggled to configure you computer environment set up when you work on a new computer?
- Or have you ever had difficult reproducing a bug or error that another team member is experiencing?
- Or that only exists in your production environment but not in your development environment?

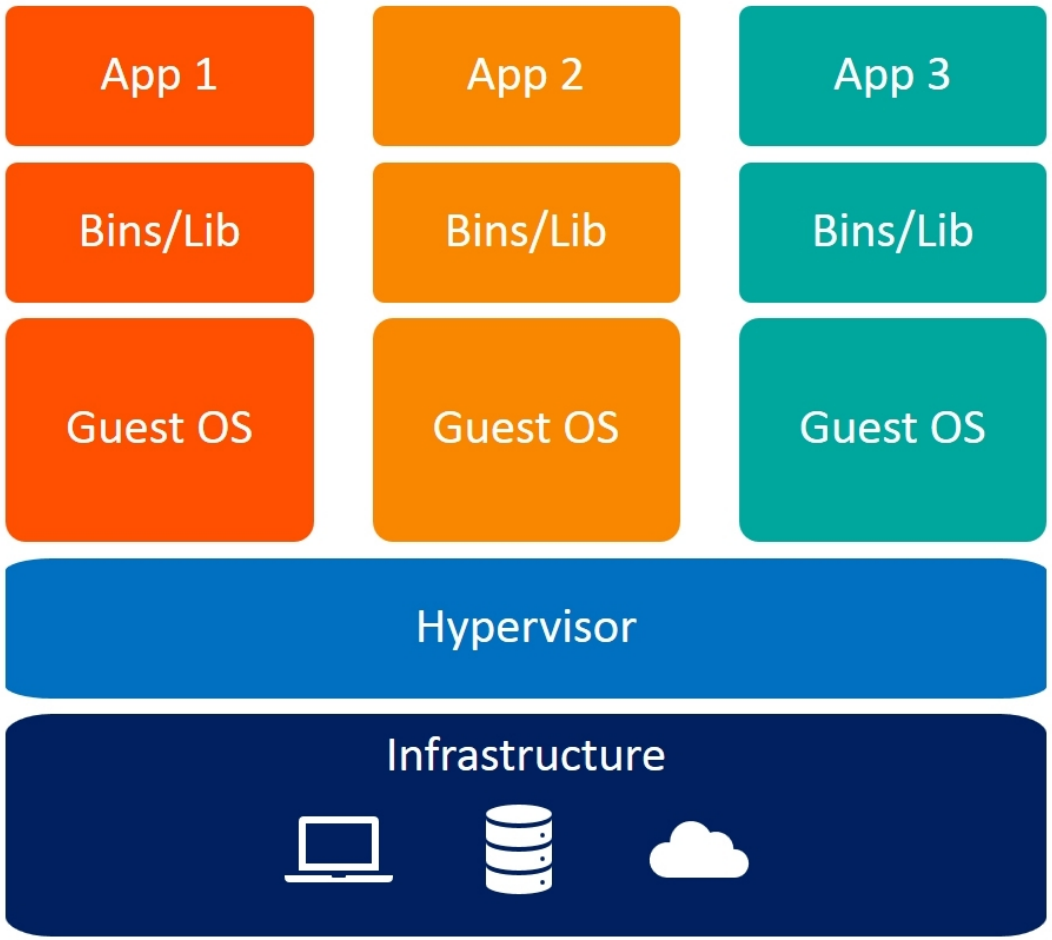
What is the problem?



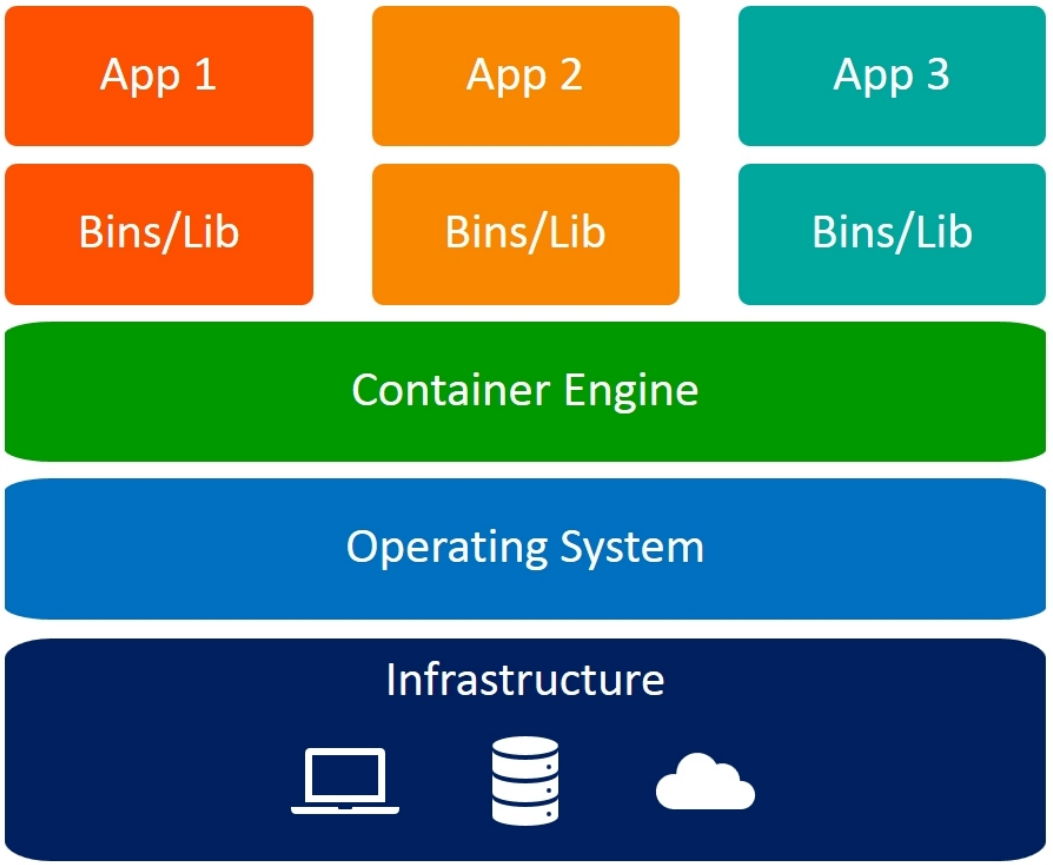
- How to experience multi OSes on one device?
- How to “build once, deploy anywhere”?
- How to use hardware efficently?

Possible Solutions

- Virtualization and Containerization



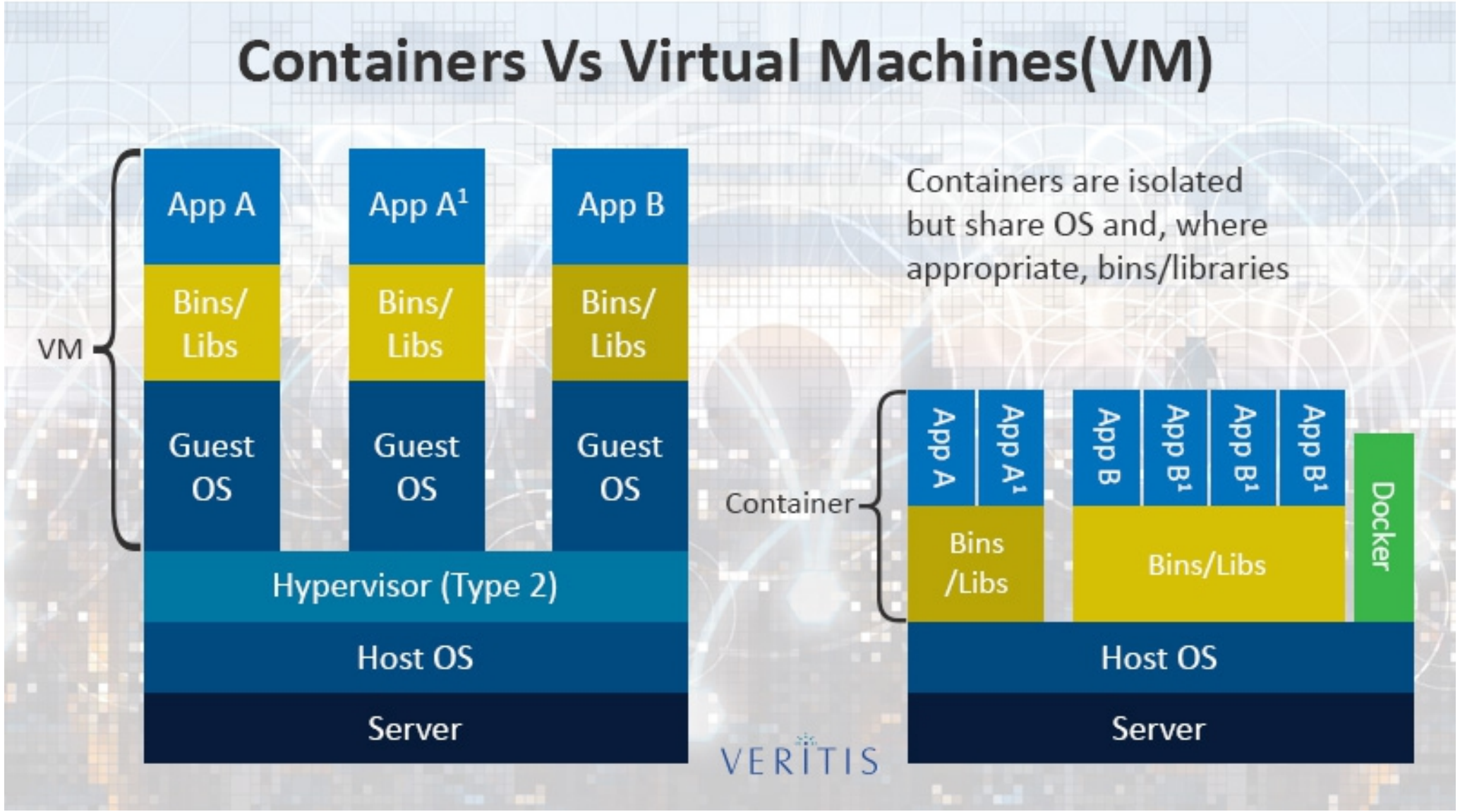
Virtual Machines



Containers

Possible Solutions

- Virtualization and Containerization



Possible Solutions

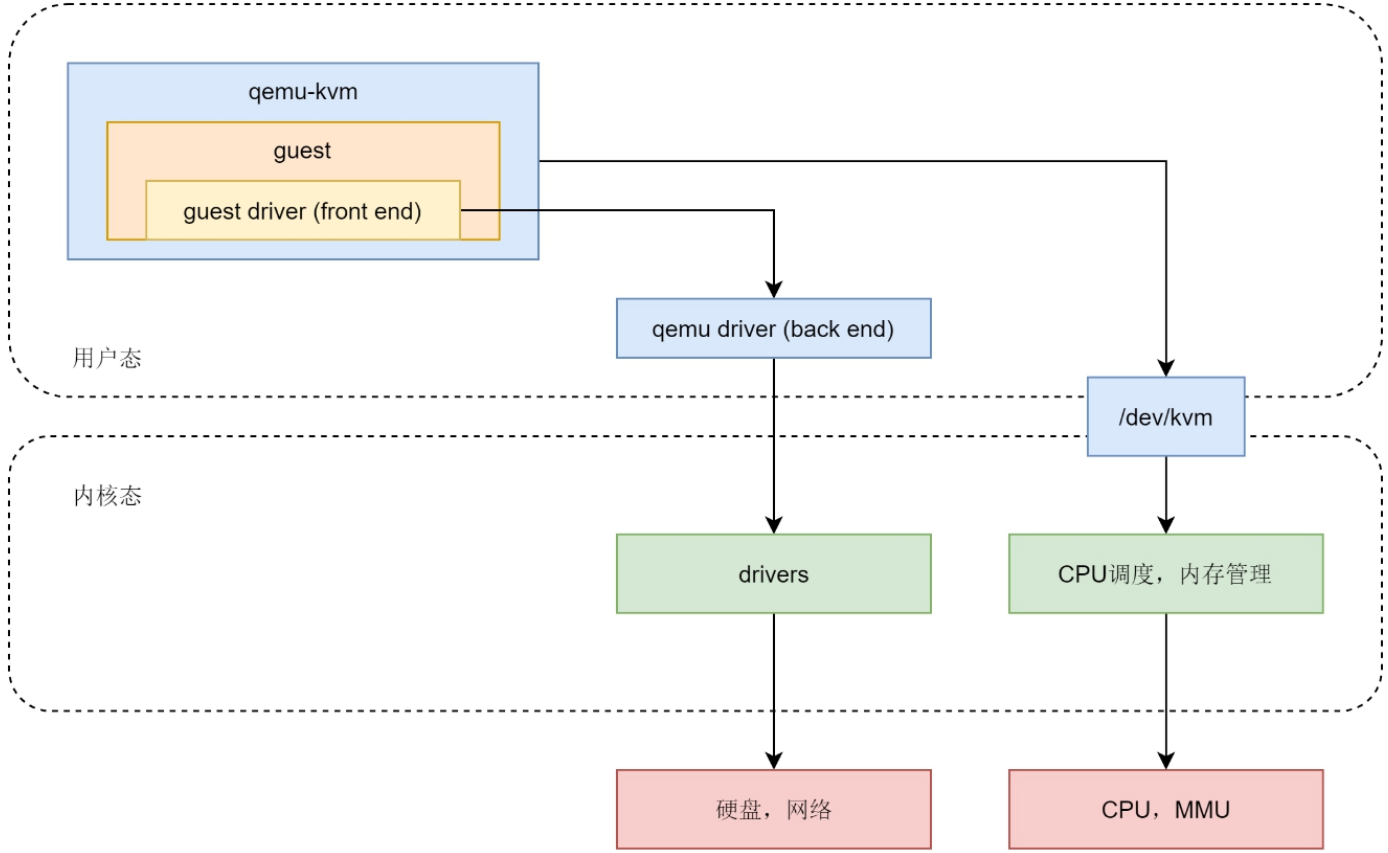
- Virtualization and Containerization

Is this problem not easy to solve? Abstract a layer!

What is VM?

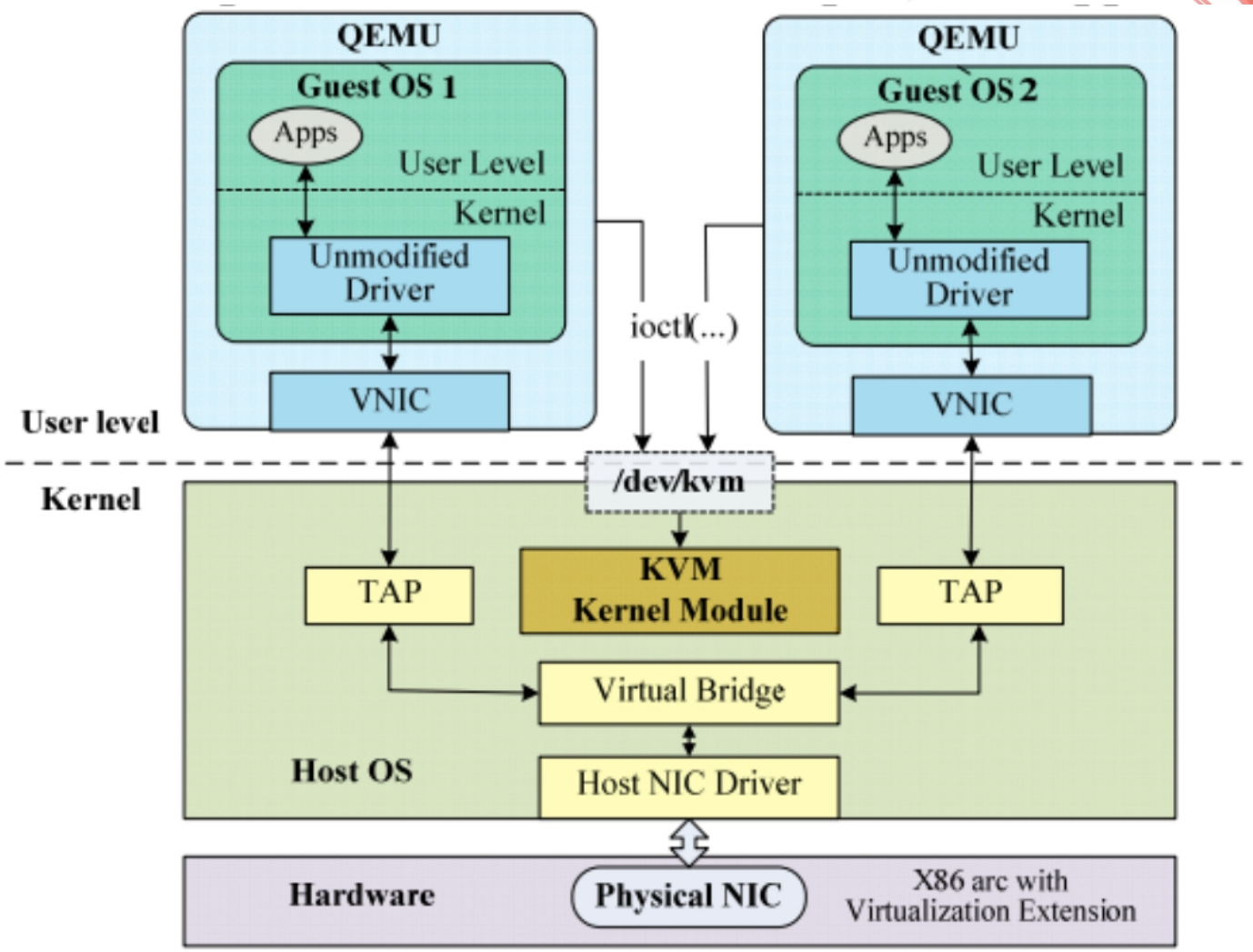
- Full Virtualization
  - QEMU
- Hardware-Assisted Virtualization
  - QEMU-KVM (Intel-VT/AMD-V)
  - CPU/Memory
- Paravirtualization
  - QEMU-Driver
  - Device/Driver

Like a subsidiary company.



KVM : Kernel-based Virtual Machine

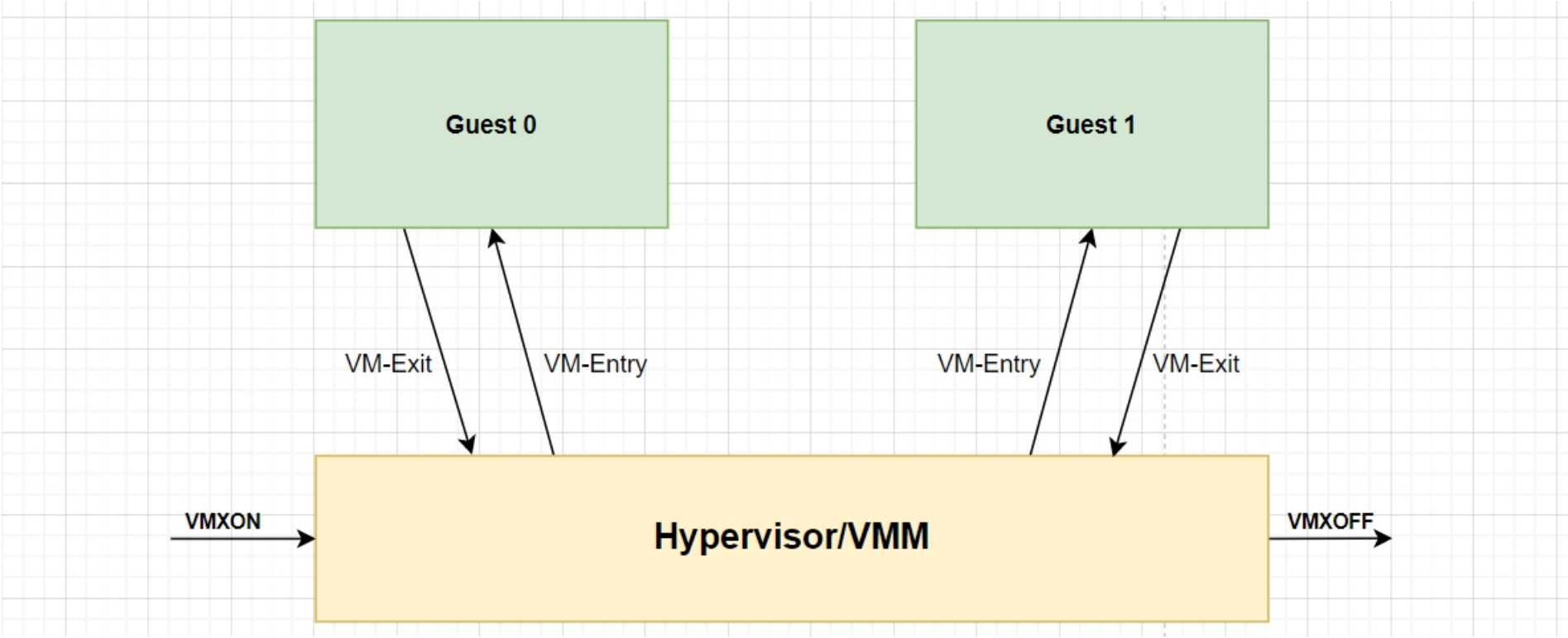
- A full virtualization solution
- Containing virtualization extensions (Intel VT or AMD-V)
- A loadable kernel module, kvm.ko, that provides the core virtualization infrastructure
- A processor specific module, kvm-intel.ko or kvm-amd.ko



Reference:
https://faculty.cc.gatech.edu/~lingliu/papers 2013/YiRen-Cloud.pdf

CPU Virtualizition

- Virtual Machine Extension (VMX) and VT-x instruction set.
  - VMX root operation: Hypervisor/VMM run here with control of HW
  - VMX non-root operation: Guest OS and guest SW run here



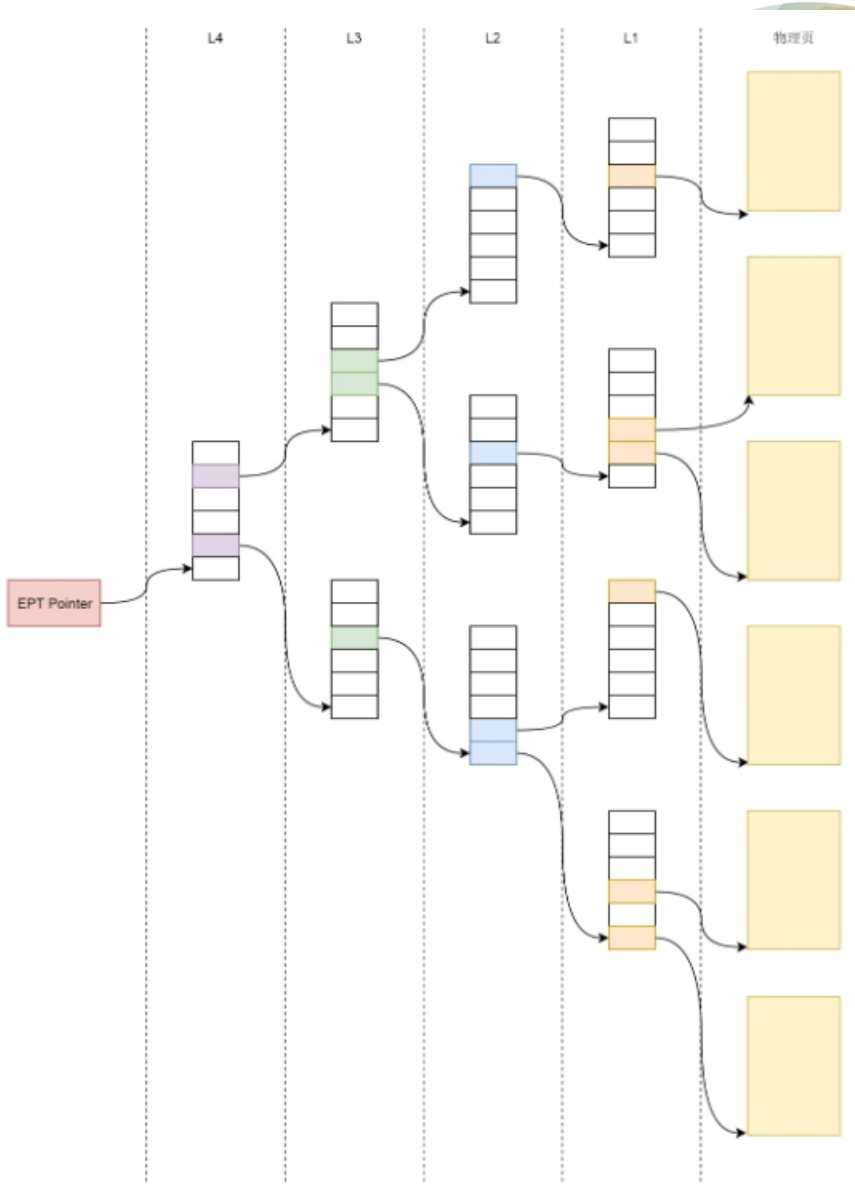
Memory Virtualizition

Memories:

- Guest OS Virtual Memory , GVA
- Guest OS Physical Memory , GPA
- Host Virtual Memory , HVA
- Host Physical Memory , HPA

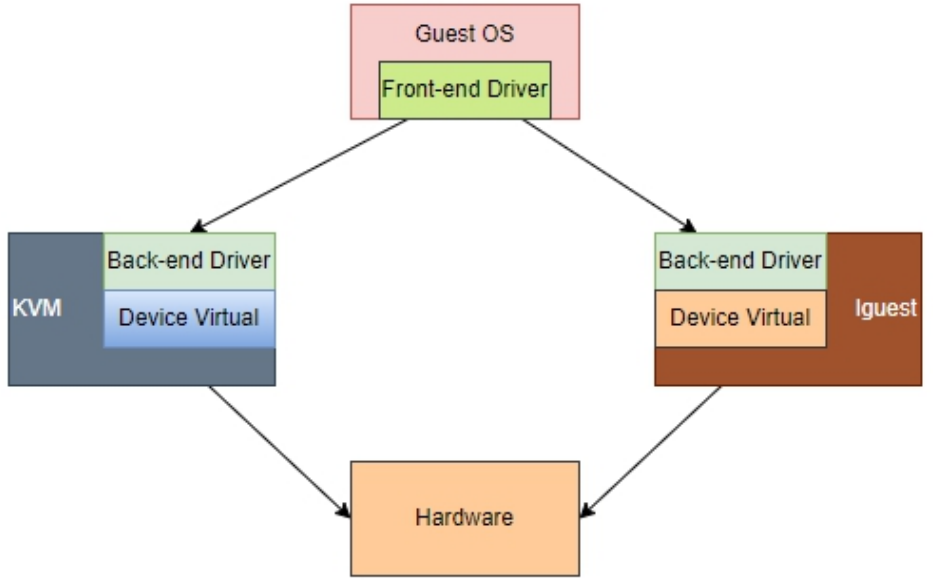
Page allocation and mapping:

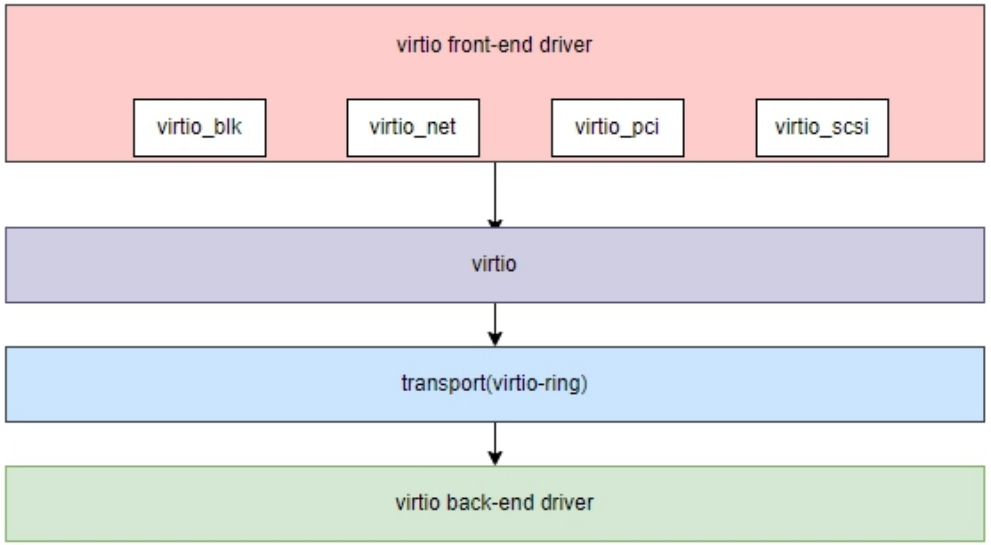
- Shadow Page Table
  - Create for each Guest OS
- EPT (Extended Page Tables)
  - Two dimentional paging, from GPA to HVA



I/O Virtualizition

- Implement front-end driver in Guest OS
- virtio + virtio-ring layer to comunicate
- Implment back-end drvier in KVM

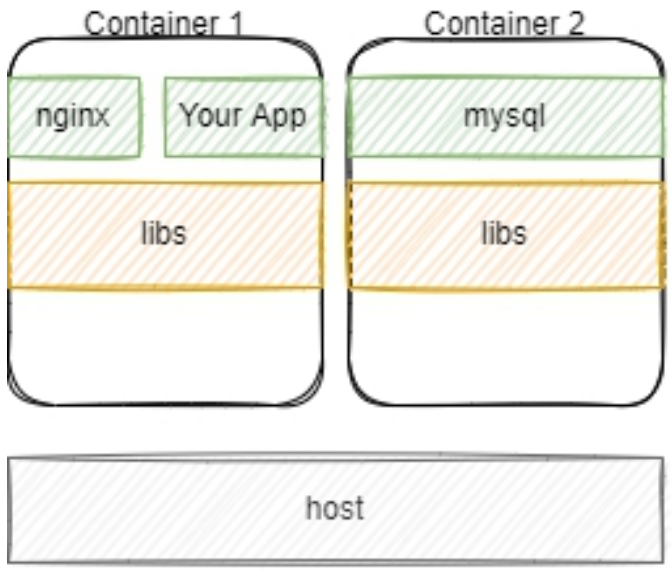




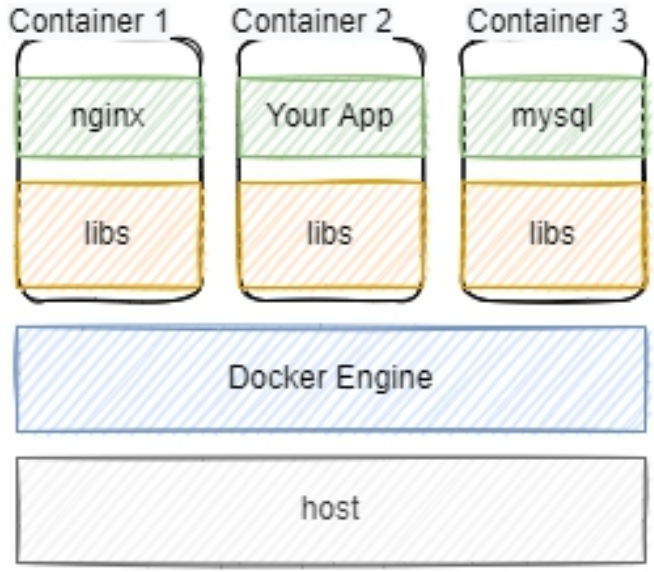
Containerization: LXC vs Docker

- LXC: Enables you to create and run multiple OS simultaneously on a single Linux machine (LXC host)
  - Use Linux kernel features to create isolated processes and filesystems
- Docker: Focuses on running a single application in an isolated environment.
  - Developed its own replacement for LXC

Intrapreneurship, sharing resources.



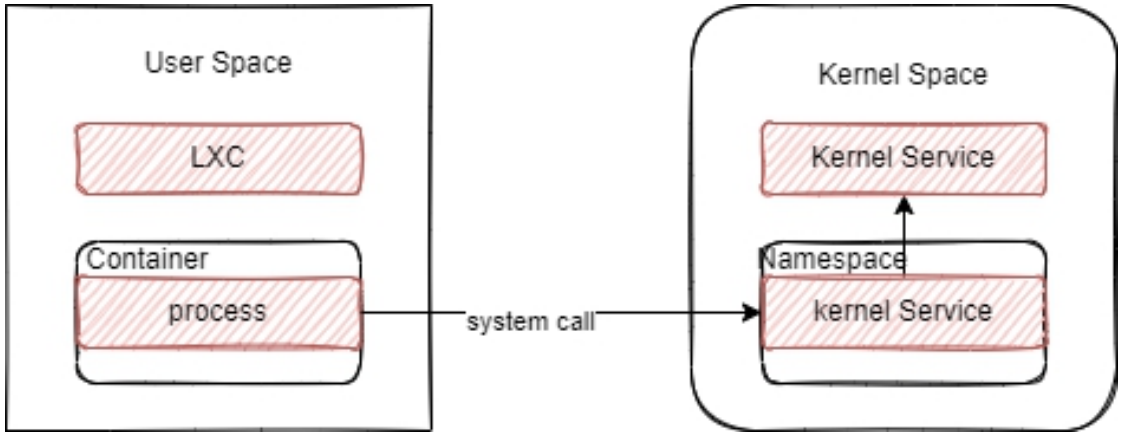
LXC



Docker

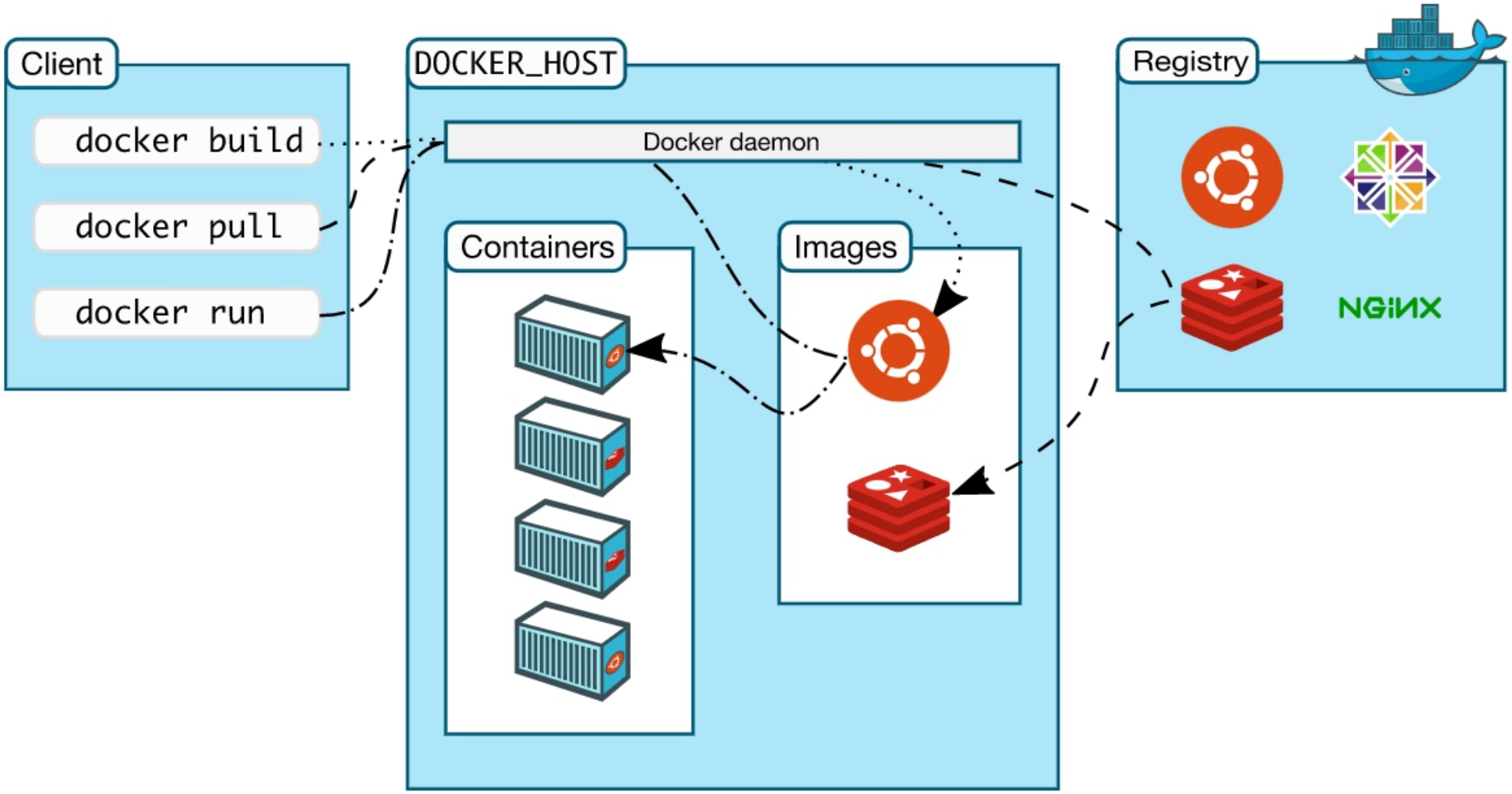
LXC

- A userspace interface for the Linux kernel containment features
- Used Features
  - **Kernel namespaces** (ipc, uts, mount, pid, network and user)
  - Apparmor and SELinux profiles
  - Seccomp policies
  - Chroots (using pivot_root)
  - Kernel capabilities
  - **Cgroups** (control groups)
- Used Components
  - The **liblxc** library
  - Several language bindings for the API: python3/lua/Go/ruby/Haskell
  - A set of standard tools to control the containers
  - Distribution container templates



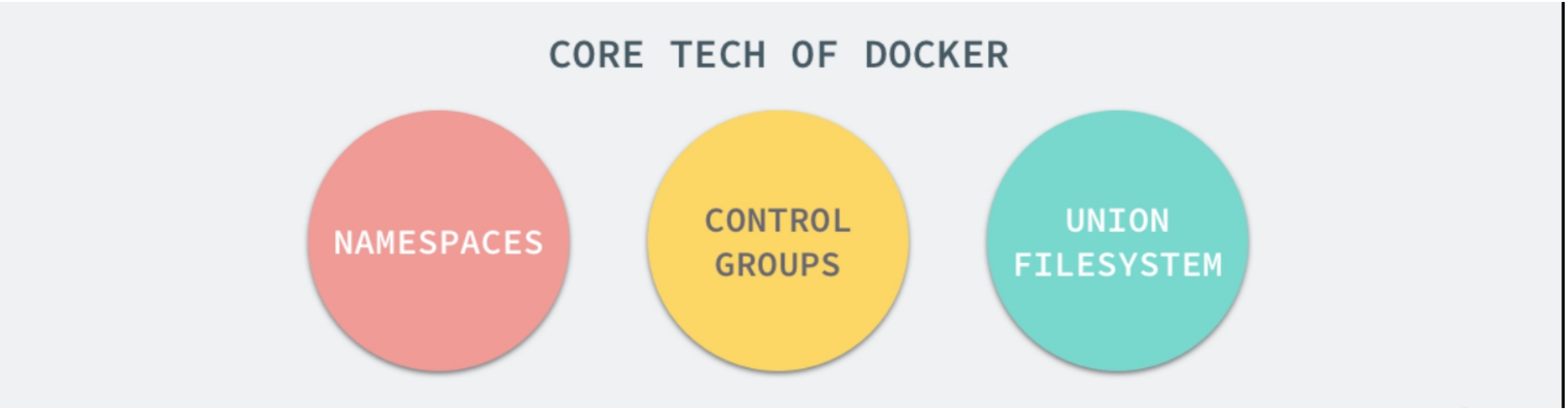
What is the docker?

- **Image is a great development and design. Do you know why?**



What is the docker?

- **Docker is a “single process” model, which usualy relpaces the start up process with systemd/supervisord**



Namespaces

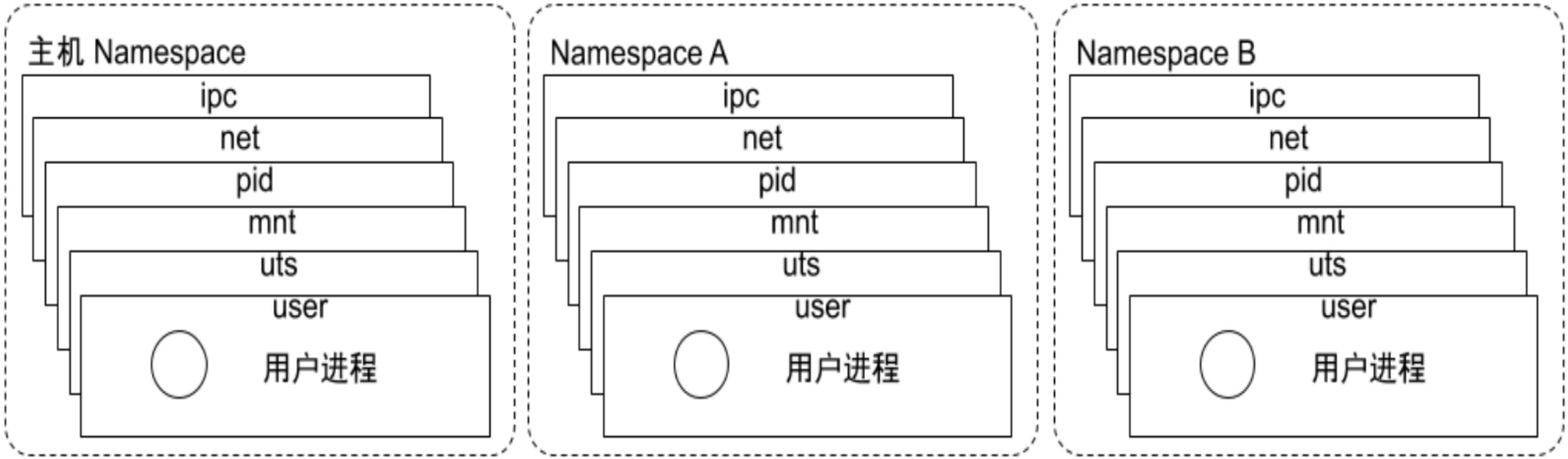
A solution to isolate environment on kernel Level, which is similiar to chroot(man chroot).

Catogory	Constant	Kerenl Version	Isolates (man namespaces)
Mount namespace	CLONE_NEWNS	Linux 2.4.19	Mount points
UTS namespace	CLONE_NEWUTS	Linux 2.6.19	Hostname, NIS domain name
IPC namespace	CLONE_NEWIPC	Linux 2.6.19	IPC, message queues
PID namespace	CLONE_NEWPID	Linux 2.6.24	Process IDs
Network namespace	CLONE_NEWNET	Linux 2.6.29	Network devices, stacks, ports, etc.
User namespace	CLONE_NEWUSER	Linux 3.8	User and group IDs
Cgroup namespace	CLONE_NEWCGROUP	Linux 4.6	Cgroup root directory

System calls:

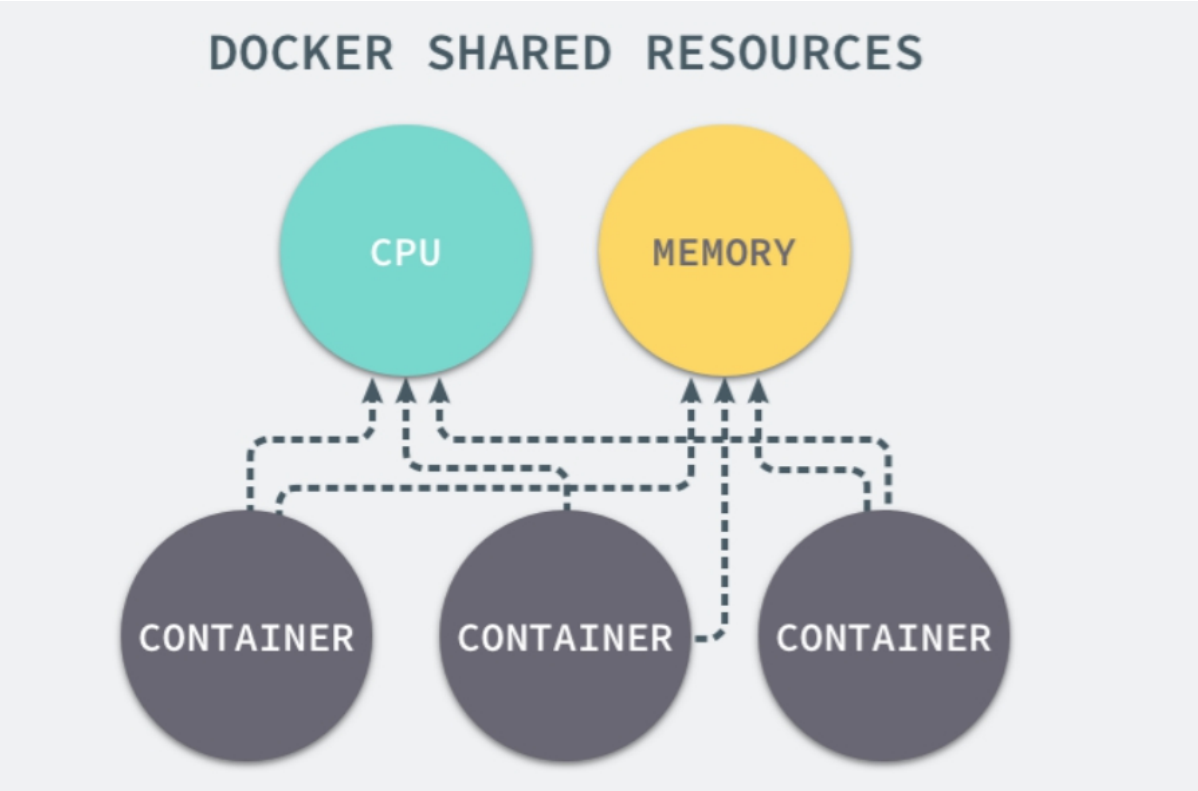
- clone(): Create a child process(task), and isolate with namesapce
- unshare(): Unshare(remove) namespace from parent
- setns(): Reassociate thread with a namespace

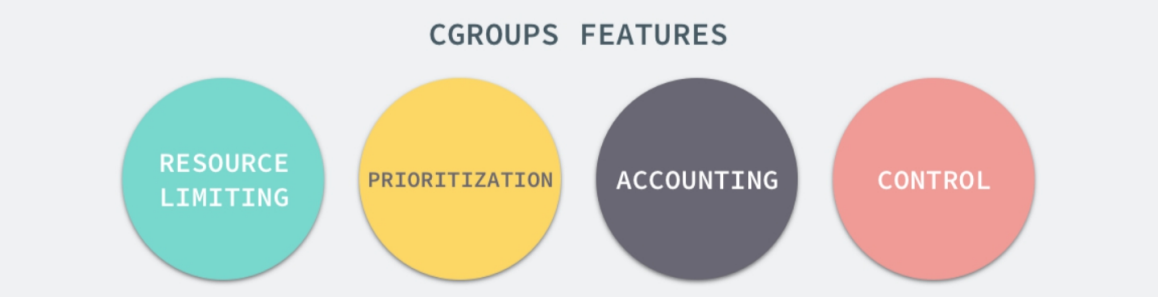
Namespaces


主机 Namespace
ipc
net
pid
mnt
uts
user
用户进程
Namespace A
ipc
net
pid
mnt
uts
user
用户进程
Namespace B
ipc
net
pid
mnt
uts
user
用户进程

Cgroups

A solution to isolate hardware, which controls, tracks and limits usage of resources used b





Cgroup

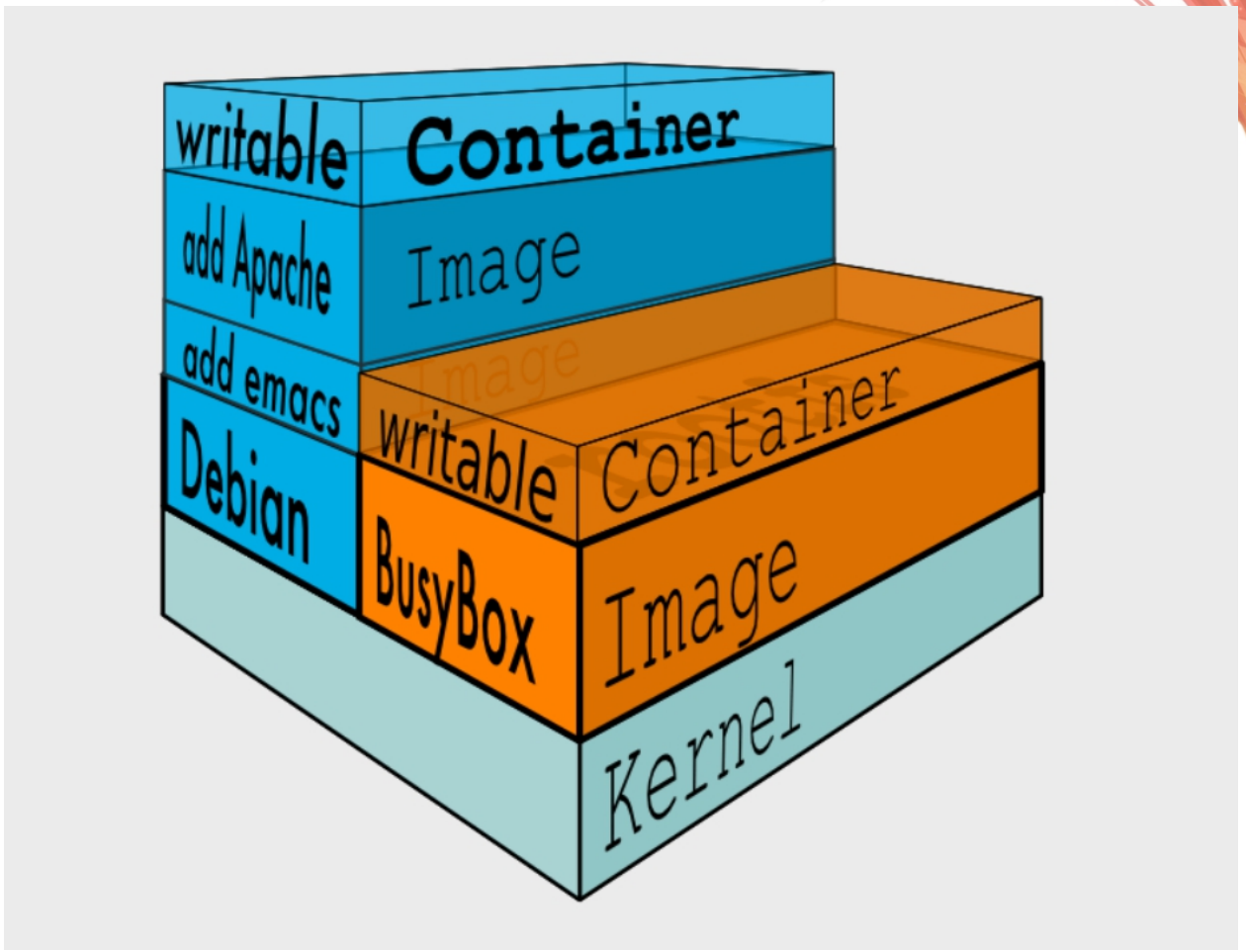
Subsystem	Description (man cgroups)
blkio	Set limits on I/O access to and from block devices such as physical drives
cpu	Uses the scheduler to provide cgroup tasks access to the CPU
cpuacct	Generate automatic reports on CPU resources
cpuset	Assigns individual CPUs and memory nodes
devices	Allows or denies access to devices
freezer	Suspends or resumes tasks
memory	Sets limits on memory use by tasks and generate automatic reports
net_cls	Tags network packets with a class identifier(classid) that allows the linux traffic controller to identify packets originating
net_prio	Provides a way to dynamically set the priority of network traffic per network interface
pids	Permits limiting the number of process that may be crreated
perf_event	Allows perf monitoring of the set of processes

To know more, try: *lssubsys -m ls /sys/fs/cgroup*

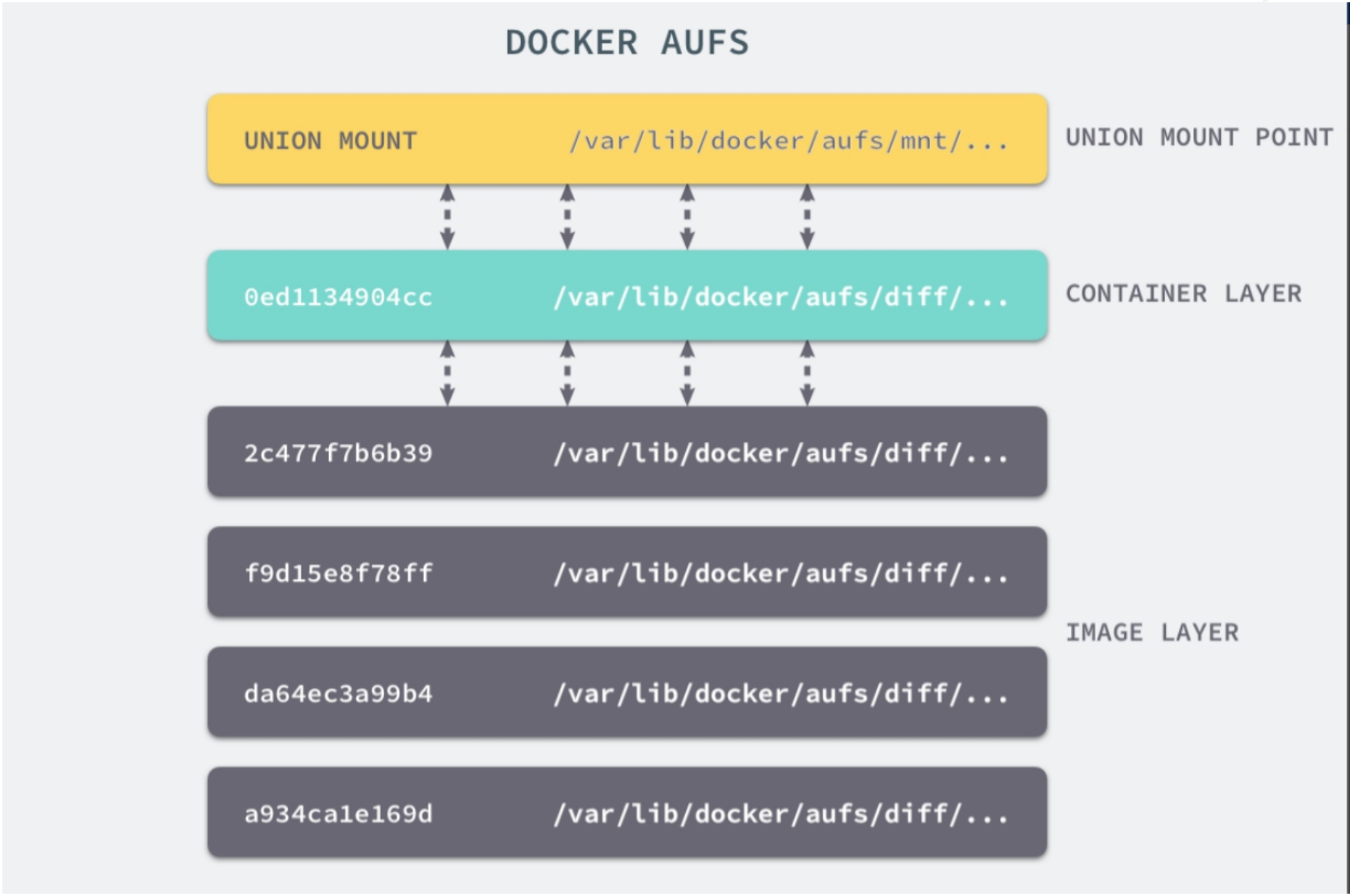
UnionFS/AuFS

- UnionFS: Implements a union mount for other file systems.
  - It allows files and directories of separate file systems
  - Likes: Run a system burned on CD/DVD with U disk, any changes won’t affect CD/DVD.
- AuFS: the advance UnionFS the author thought

Only top branch(layer) is rw, the others are ro+wh



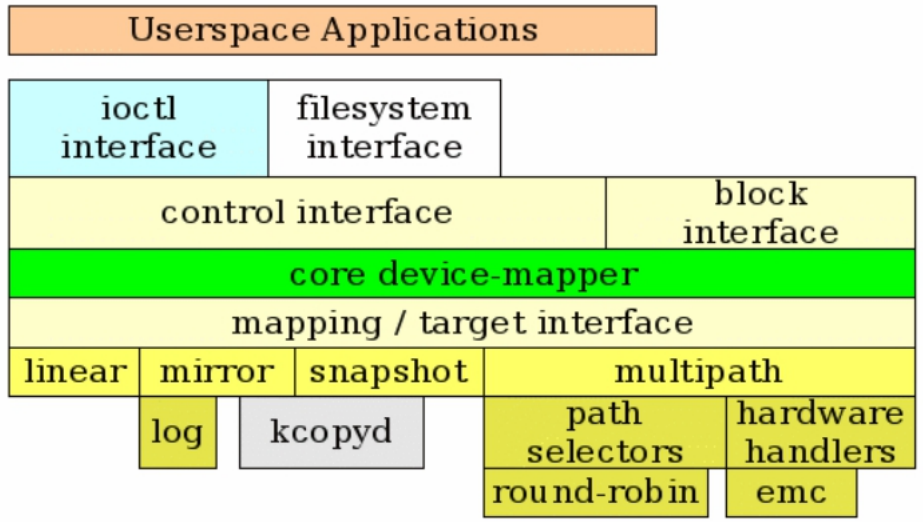
AuFS/UnionFS

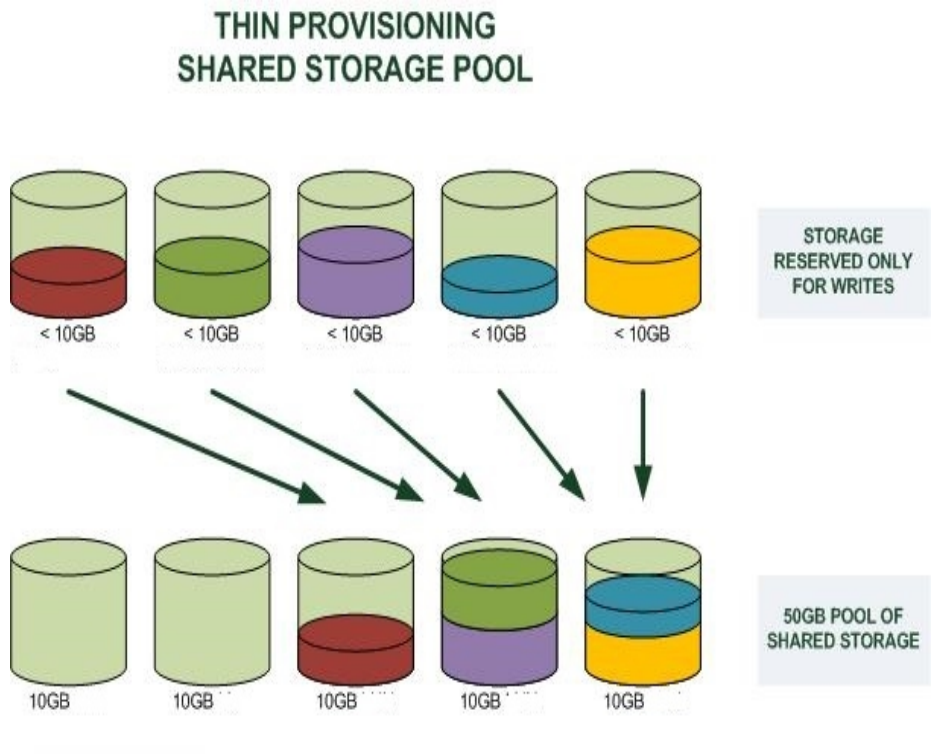


Device Mapper

redhat

Device Mapper Kernel Architecture

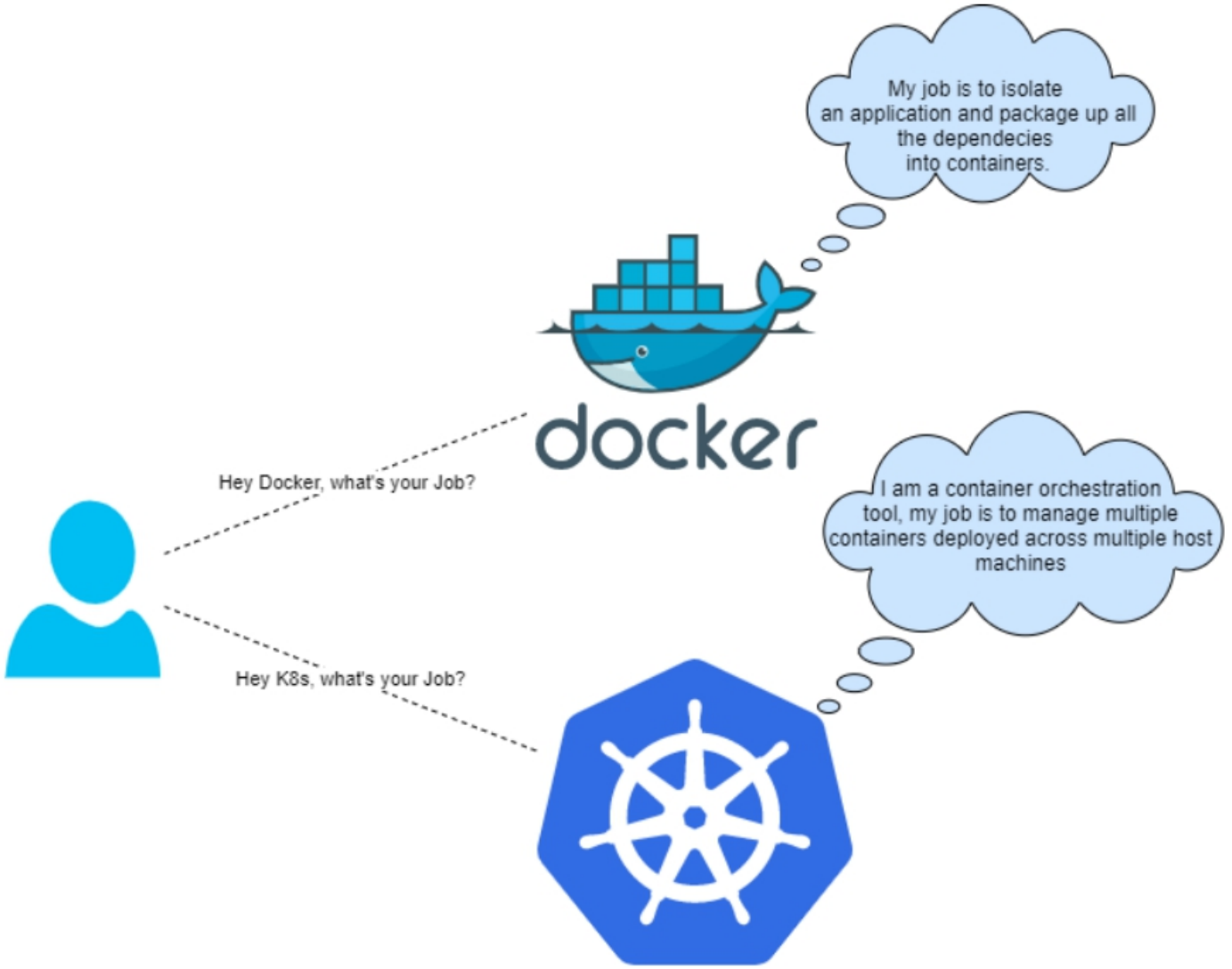




- **'Born for' the unsupported-AuFS OS**
- **Easy to install plugin**

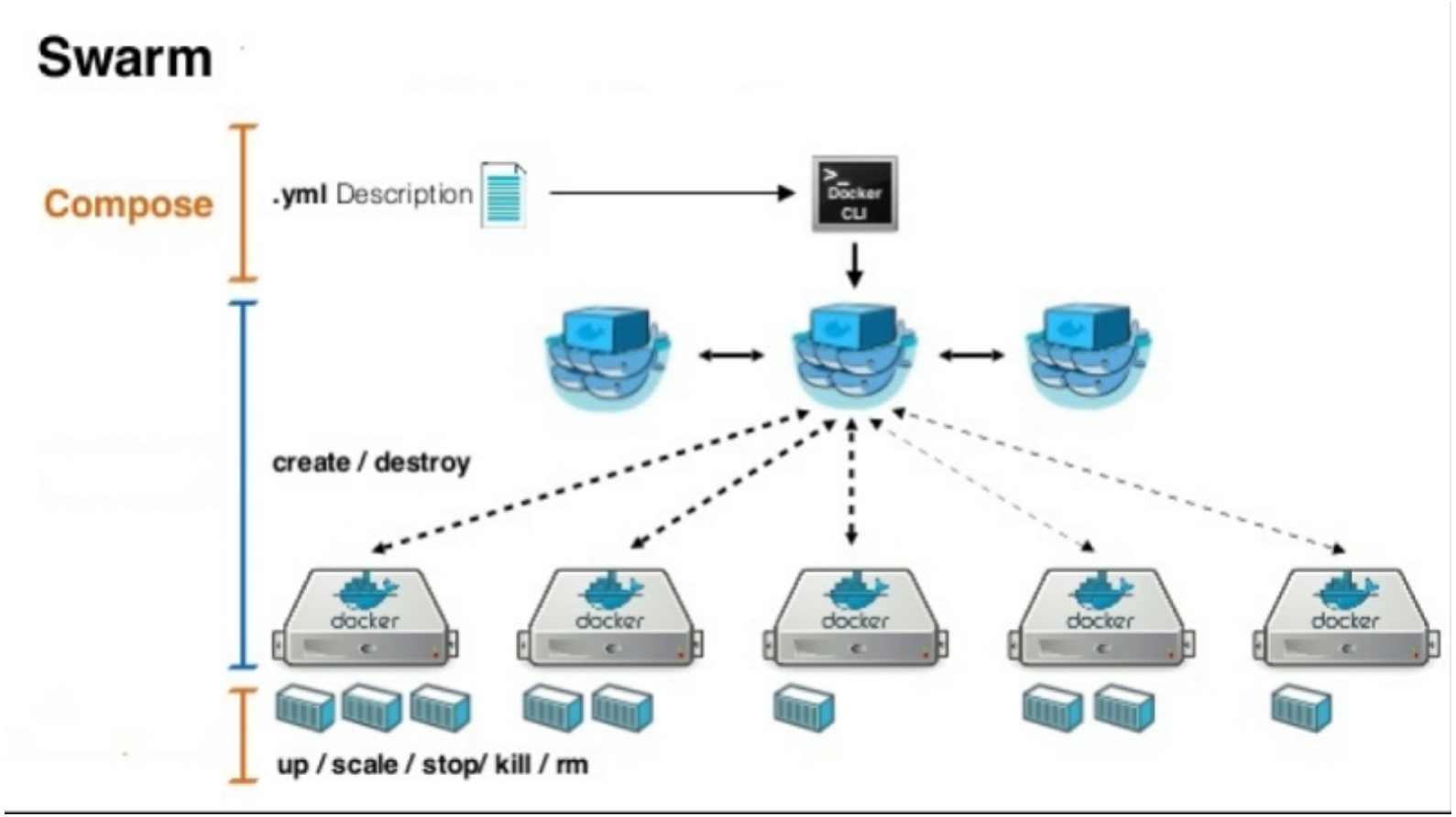
- Compare to virtual machine, docker is more lightweight, economical, and scalable.
- The docker itself does not have much value, the main value comes from the container orchestration

Kubernetes vs Docker Compose + Swarm



Kubernetes vs Docker Compose + Swarm

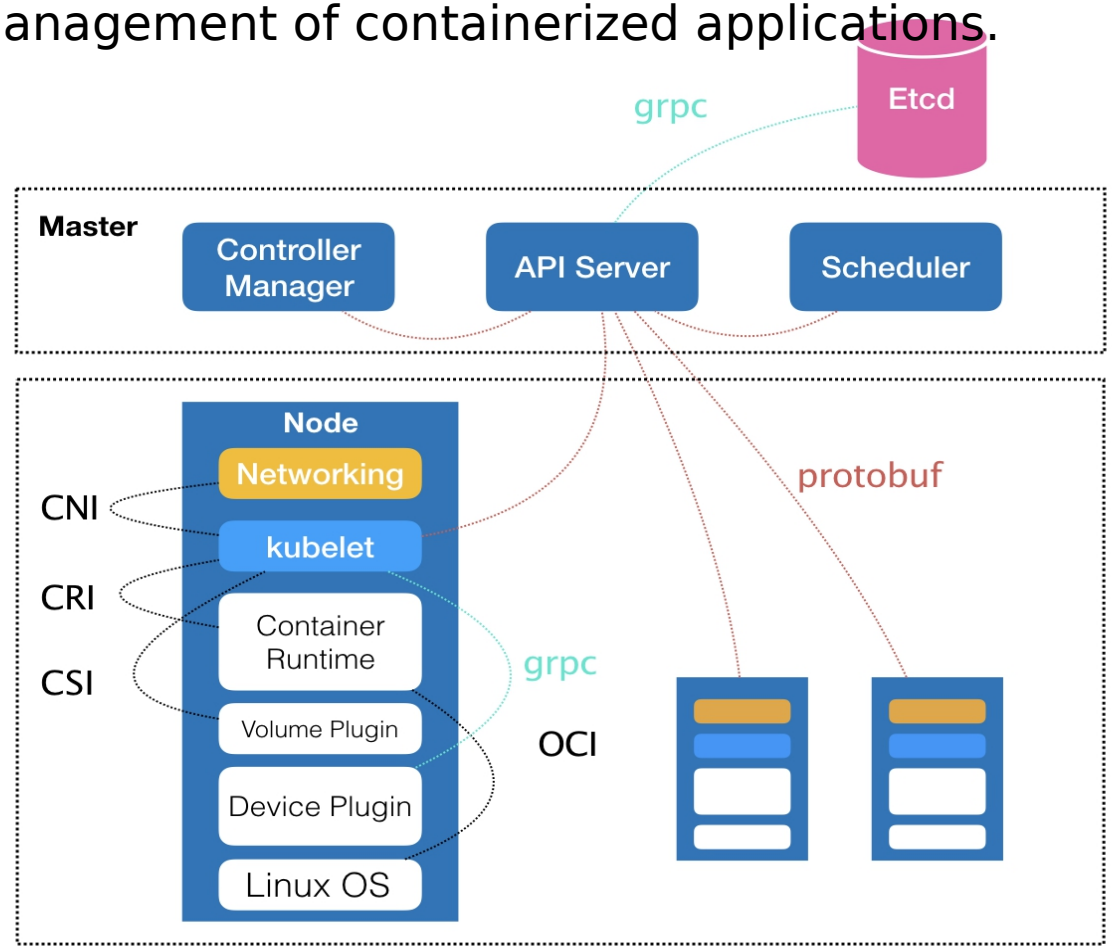
Docker Swarm is an container orchestration platform built and maintained by Docker.



- Compose
  - build multi dockers with yaml
- Nodes
  - Individual instances of the docker engine.
- Sevices and taks
- Load balancers
  - To route reuests across nodes

Kubernetes vs Docker Compose + Swarm

K8s is an container orchestration engine for automating deployment, scaling, and management of containerized applications.



- Master
  - Controller Manager for orchestration
  - API Server for API service
  - Scheduler for scheduling
- Node
  - Kubelet is responsible for docker runtime with CRI (Container Runtime Interface)
  - Handle network and volume with CNI (Container Networking Interface) and CSI (Container Storage Interface)

- Docker Swarm born for easy and agile development , Kubernetes born for medium to large clusters running complex applications

Docker vs VM - Main Differences

Virtual Machine	Docker Container
Hardware-level process isolation	OS level process isolation
Each VM has a separate OS	Each container can share OS
Boots in minutes	Boots in seconds
VMs are of few GBs	Containers are lightweight (KBs/MBs)
Ready-made VMs are difficult to find	Pre-built docker containers are easily available
VMs can move to new host easily	Containers are destroyed and re-created rather than moving
Creating VM takes a relatively longer time	Containers can be created in seconds
More resource usage	Less resource usage

VM likes you rent a house, Docker likes you register a hotel.

Takeaways

- VM vs Container: do you want a house or a hotel?
- Encoutner a problem? Maybe you need to abstract a layer or upgrade hardware.
- Learning and accumulation
  - New technology comes from basis, the new “old stuff”.
  - Basis is the first, doesn’t change for years.
  - If you do know Linux, you almost know anything.
  - Implement a docker by yourself? Refer: https://github.com/p8952/bocker

Questions And Suggestions?

Thanks