

容器技术

容器技术与 K8S 集成介绍

狄卫华 2018.10.9

目录

❖ 1. 容器技术

- ❖ 1.1 容器进程与进程
- ❖ 1.2 容器文件系统
- ❖ 1.3 容器与虚拟机
- ❖ 1.4 OCI 标准
- ❖ 1.5 容器安全

❖ 2. 容器 Runtime

❖ 3. 容器引擎

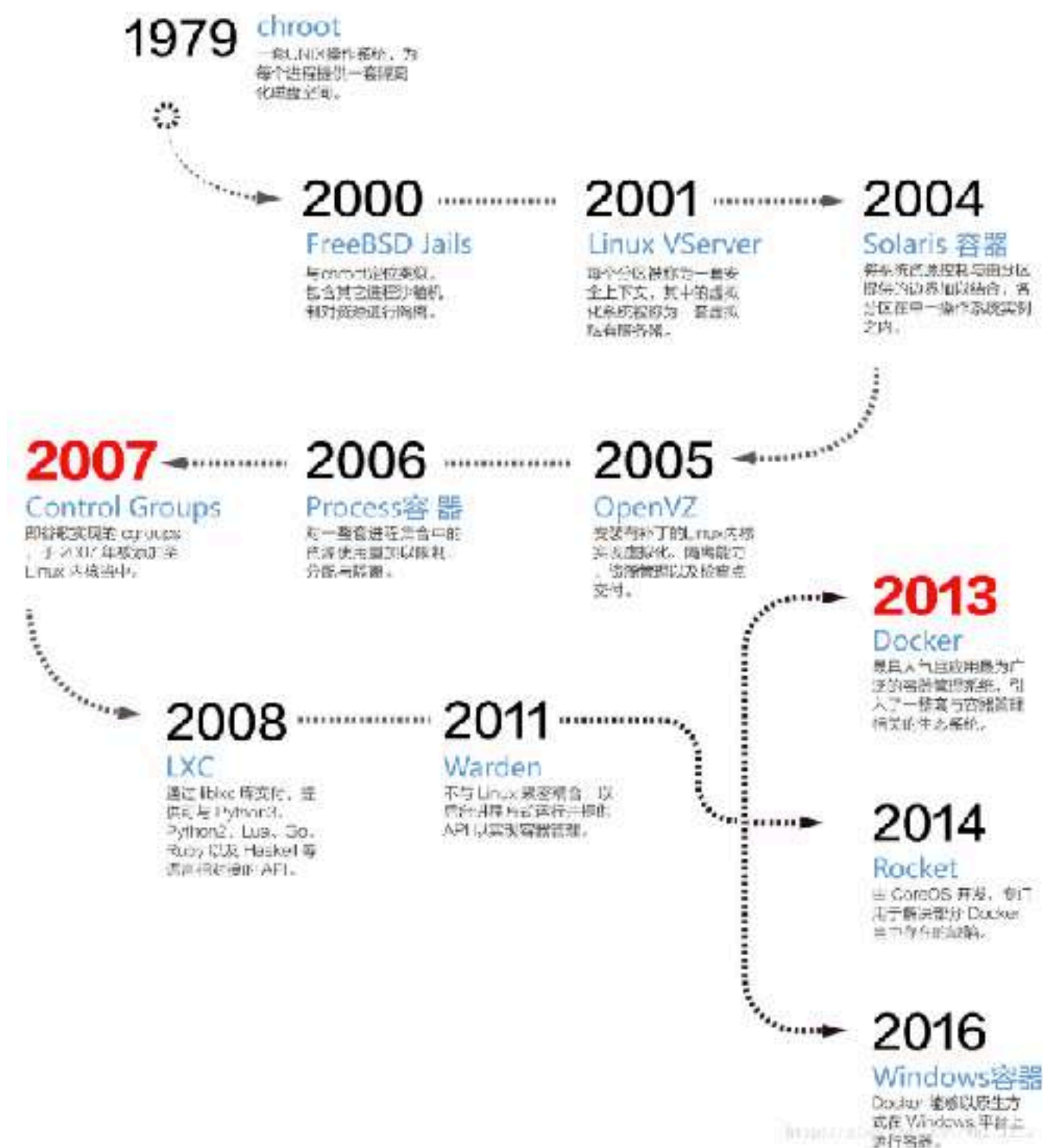
❖ 4. 容器编排

❖ 5. Kubernetes 与容器 Runtime

- ❖ 5.1 CRI 接口
- ❖ 5.2 集成方案总览
- ❖ 5.3 使用 containerd 集成
- ❖ 5.4 使用 cri-o 集成

1. 容器技术

❖ 容器不是一项新技术



1.1 容器进程与进程



进程

seLinux(redhat...), seccomp,
capabilities、apparmor(Ubuntu...)

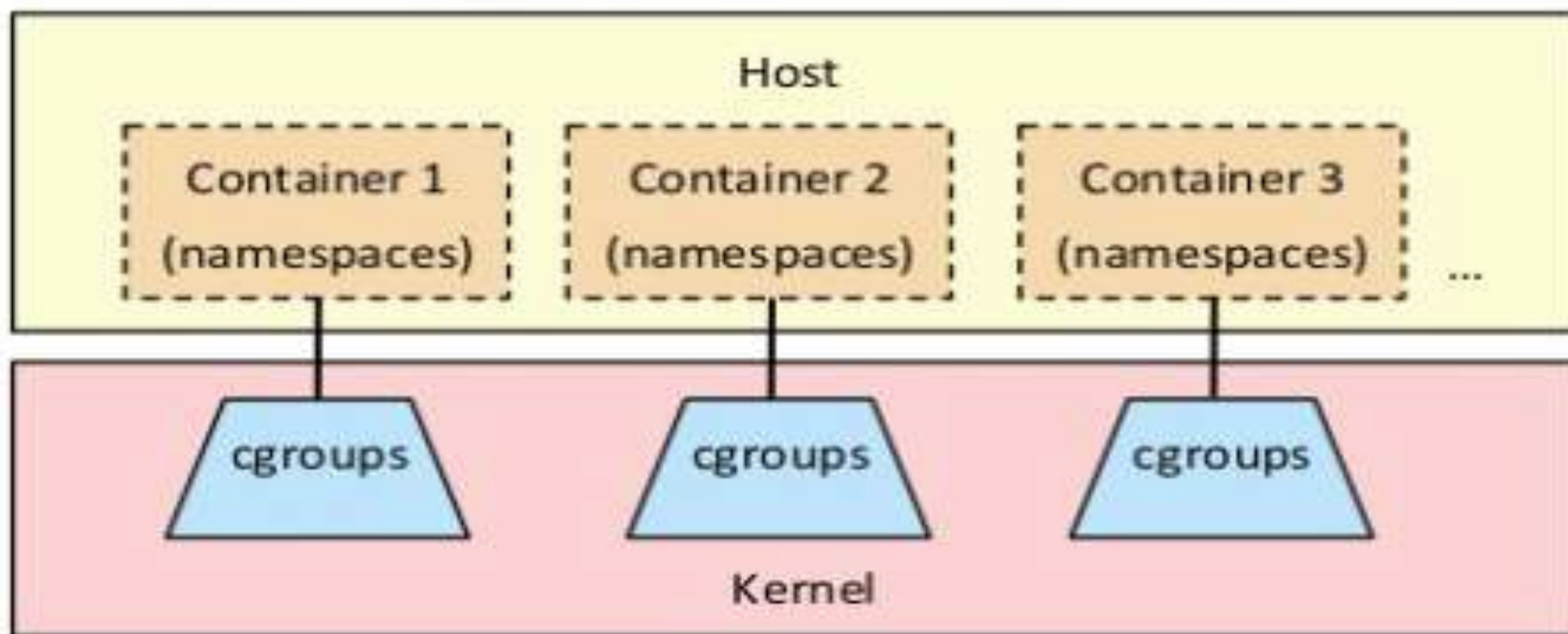
cgroups

容器进程

namespace



Linux 容器



NAMESPACES

LOGAN

namespaces

- ❖ Mount Linux 2.6.19 mount points / filesystems
- ❖ UTS Linux 2.6.19 nodename / domainname
- ❖ IPC Linux 2.6.19 System V IPC
- ❖ PID Linux 2.6.24
- ❖ Network Linux 2.6.24 - 29
- ❖ User Linux 2.6.23 - 3.8 (Docker 1.10 支持)

Linux 2.6.19 - 3.18 完全完成 2002 - 2013 年

Time?

```
int clone(int (*fn)(void *), void *child_stack, int flags, void *arg, ...);
```

示例

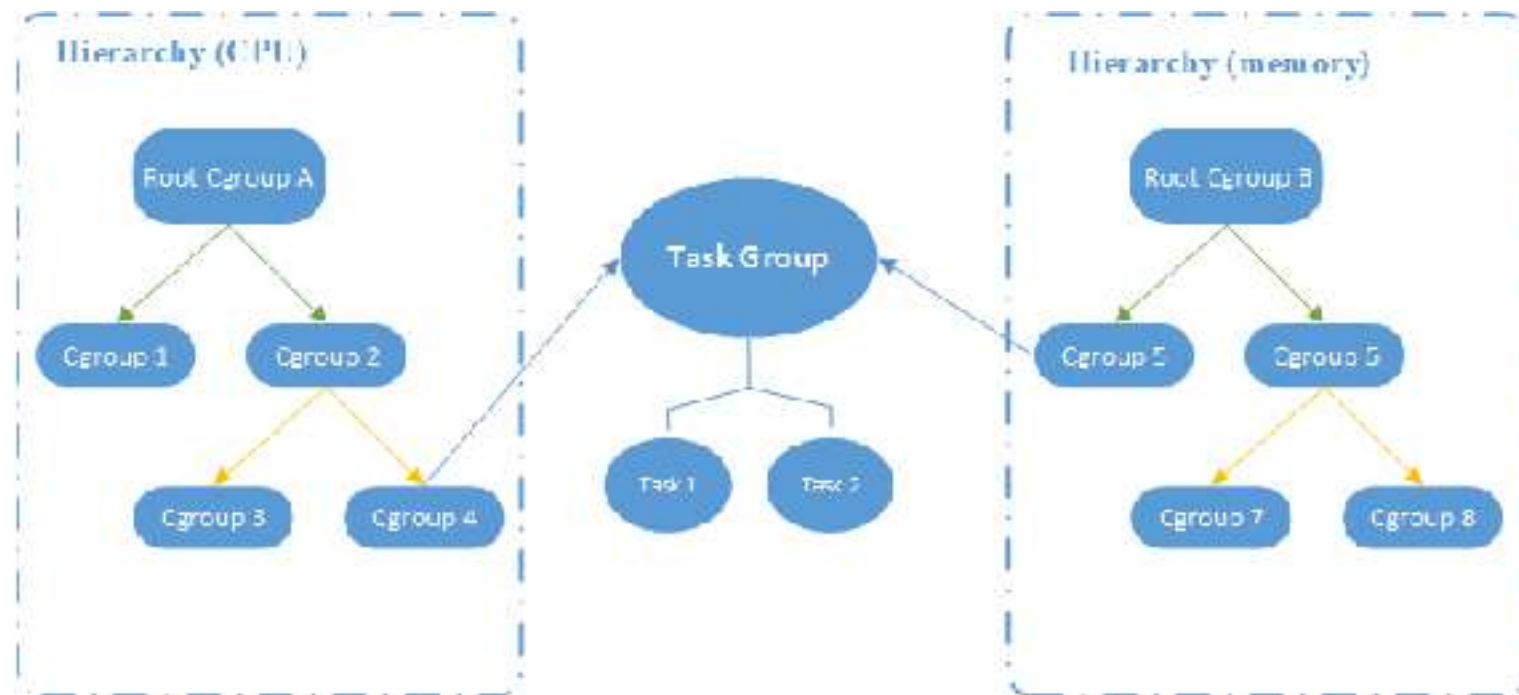
```
# ls -ahl /proc/1/ns
```

lrwxrwxrwx	1	root	root	0 Oct 8 07:17	cgroup -> cgroup:[4026531835]
lrwxrwxrwx	1	root	root	0 Oct 8 07:17	ipc -> ipc:[4026533592]
lrwxrwxrwx	1	root	root	0 Oct 8 07:17	mnt -> mnt:[4026533590]
lrwxrwxrwx	1	root	root	0 Oct 8 07:17	net -> net:[4026533595]
lrwxrwxrwx	1	root	root	0 Oct 8 07:17	pid -> pid:[4026533593]
lrwxrwxrwx	1	root	root	0 Oct 8 07:17	user -> user:[4026531837]
lrwxrwxrwx	1	root	root	0 Oct 8 07:17	uts -> uts:[4026533591]



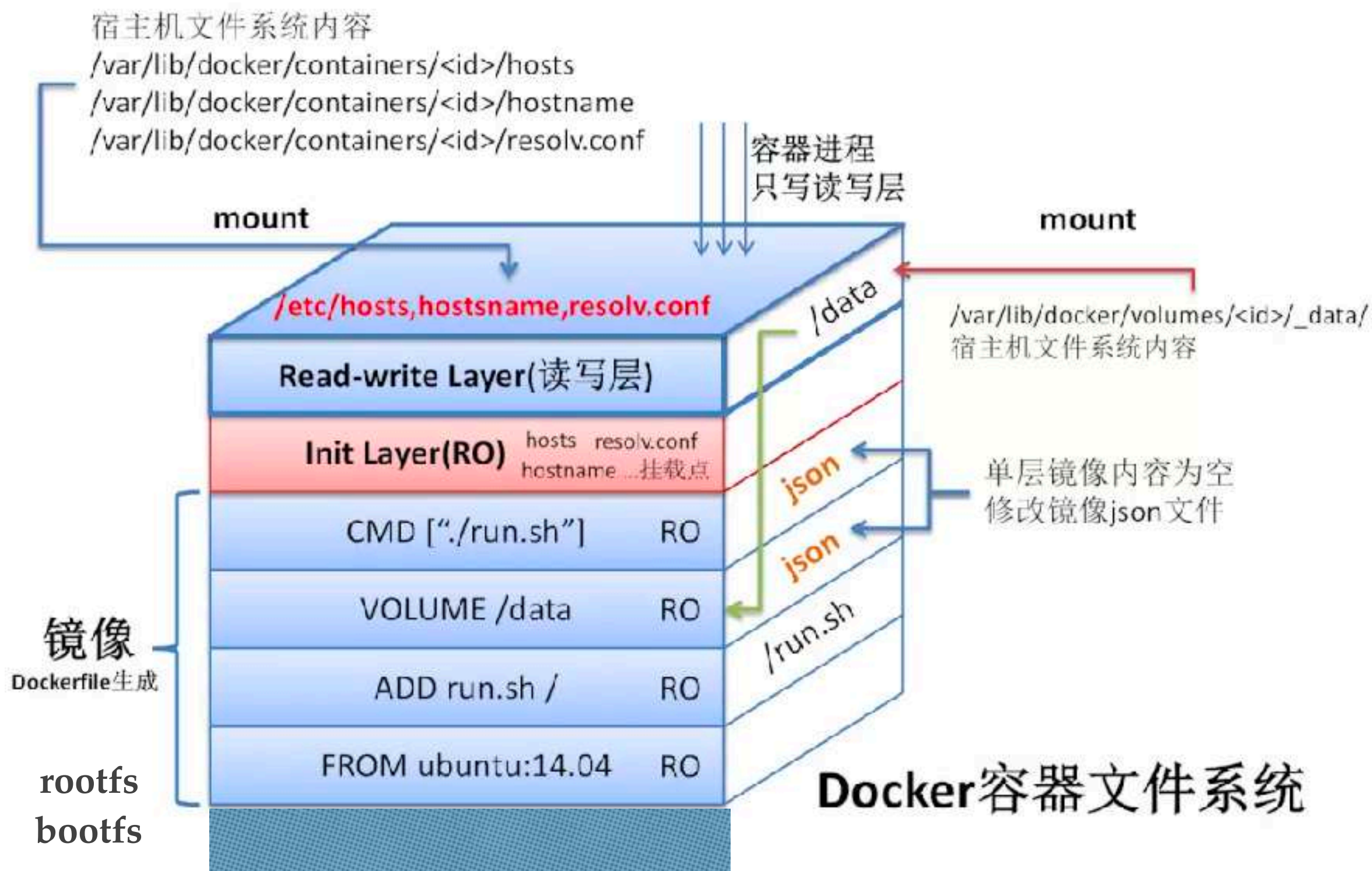
cgroups

- ❖ 资源限制 (memory / cpu / cpuset / blkio...)
- ❖ 优先级分配
- ❖ 资源统计
- ❖ 任务控制



2007 Linux 2.6.24 Google Paul Menage

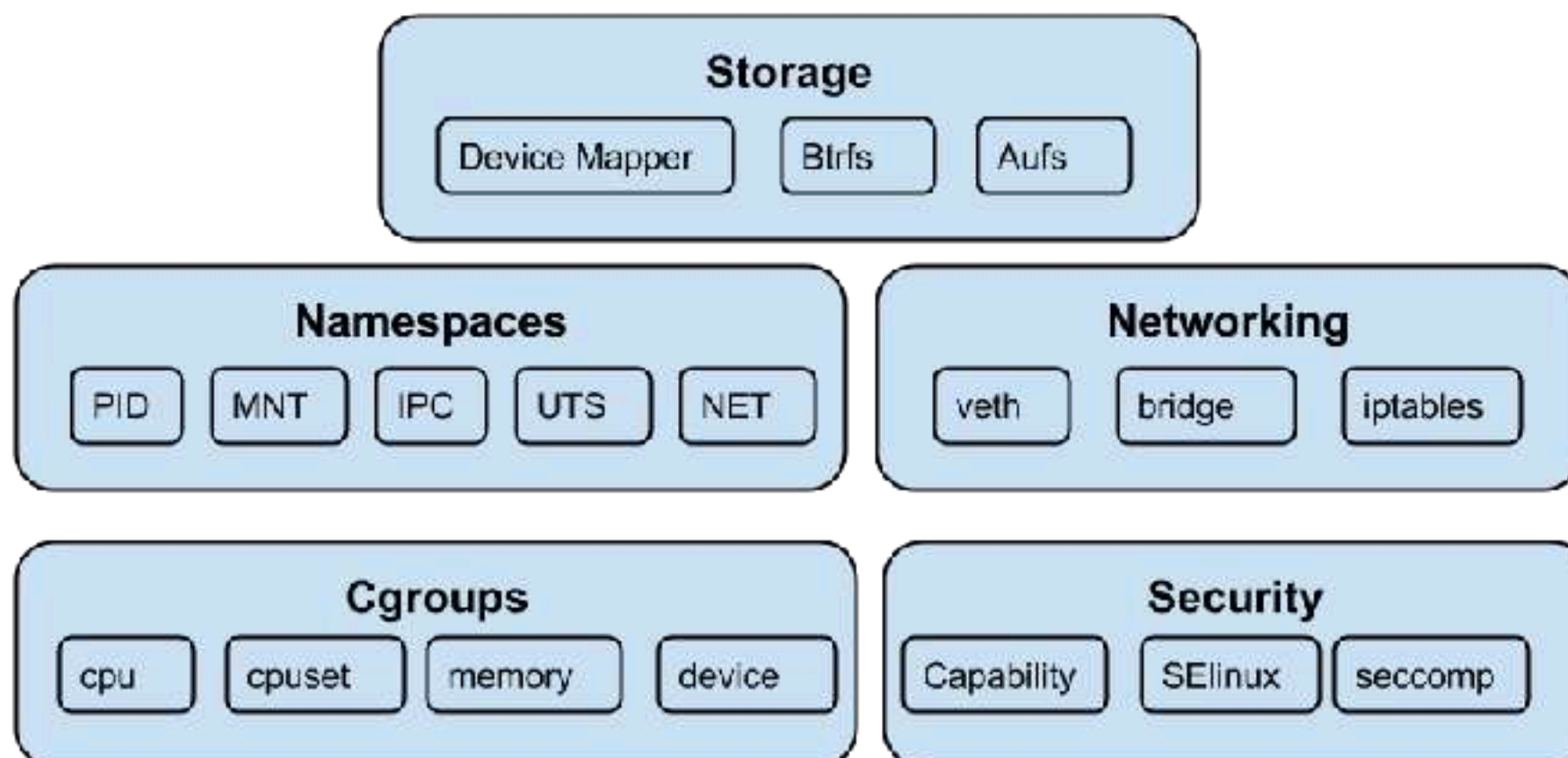
1.2 容器文件系统



Docker 示意图

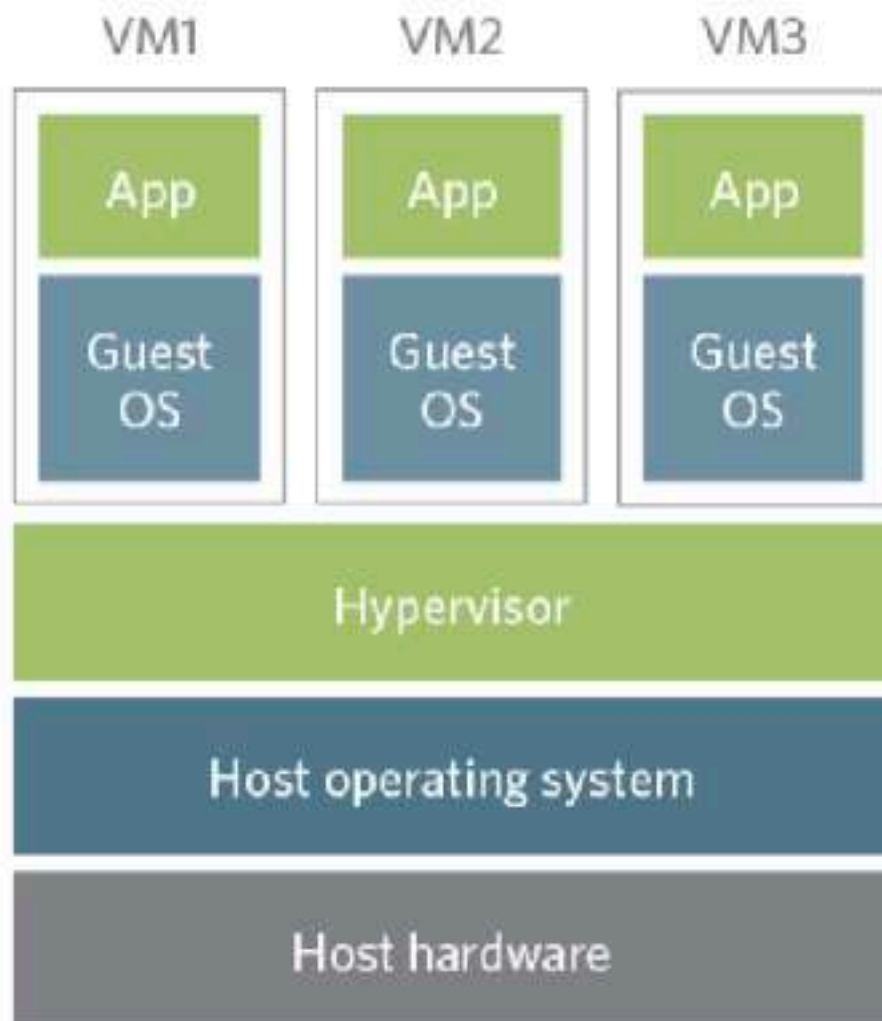


Linux Kernel

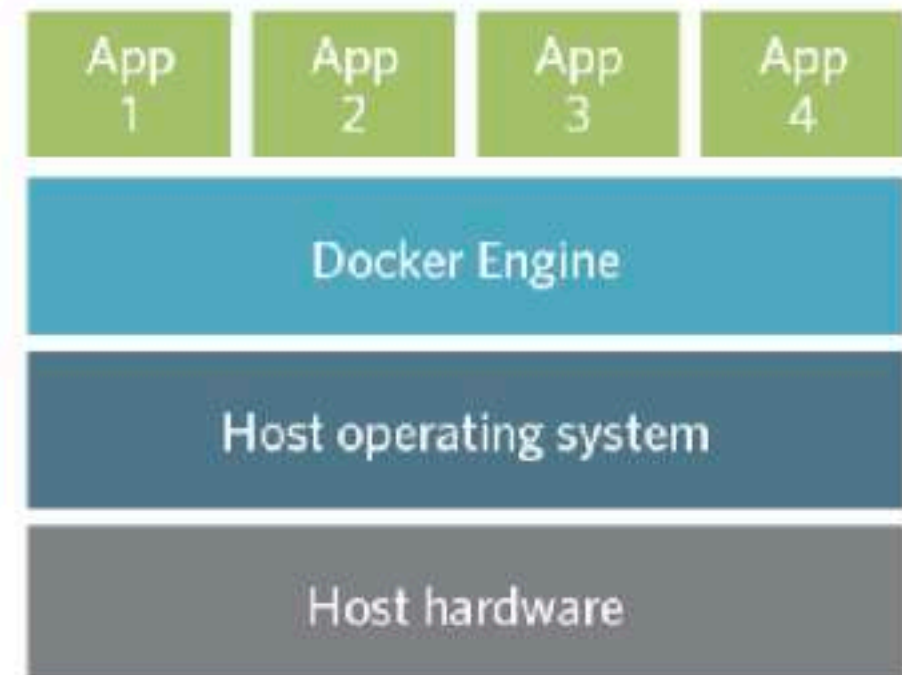


1.3 容器与虚拟机

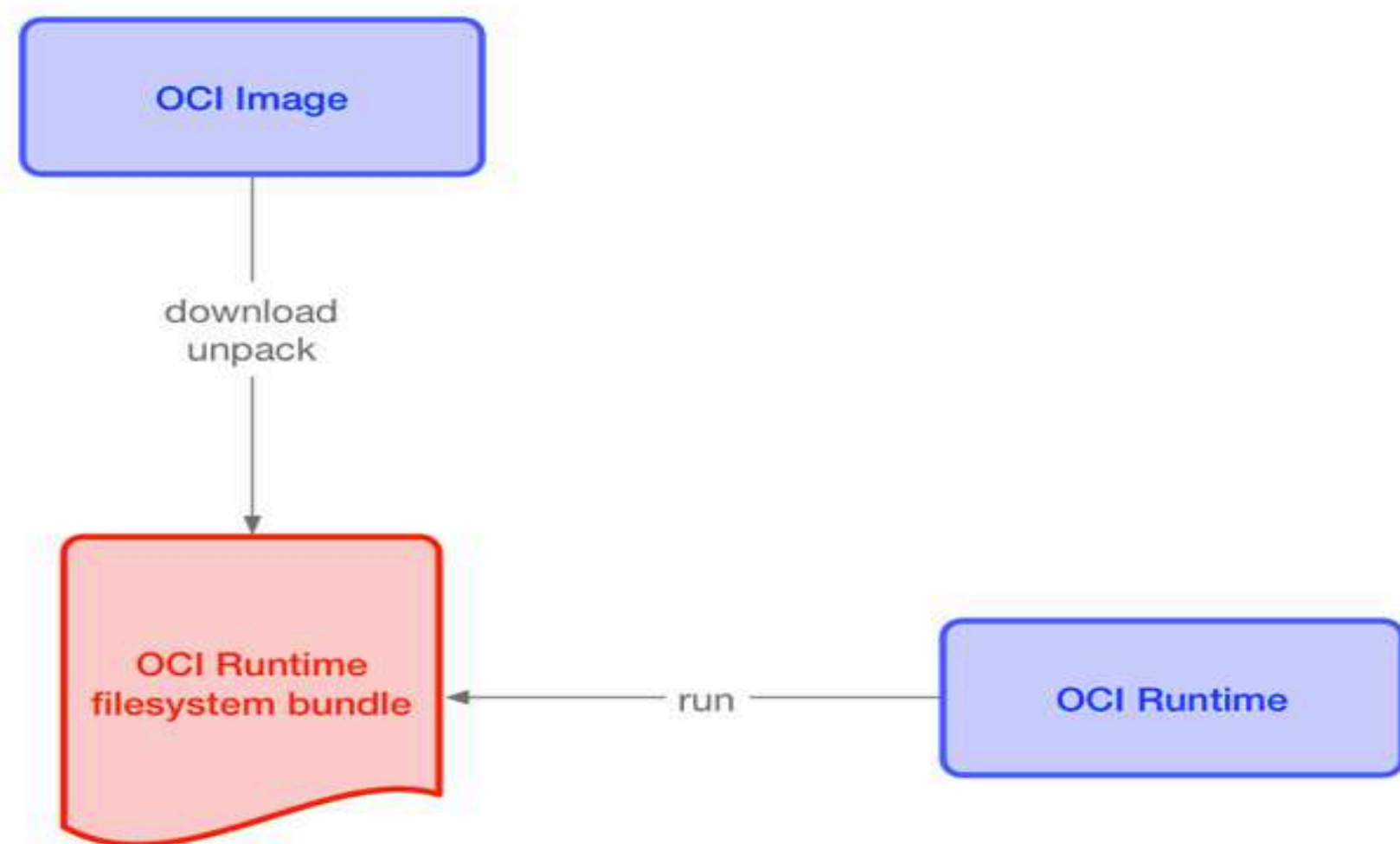
VIRTUAL MACHINES



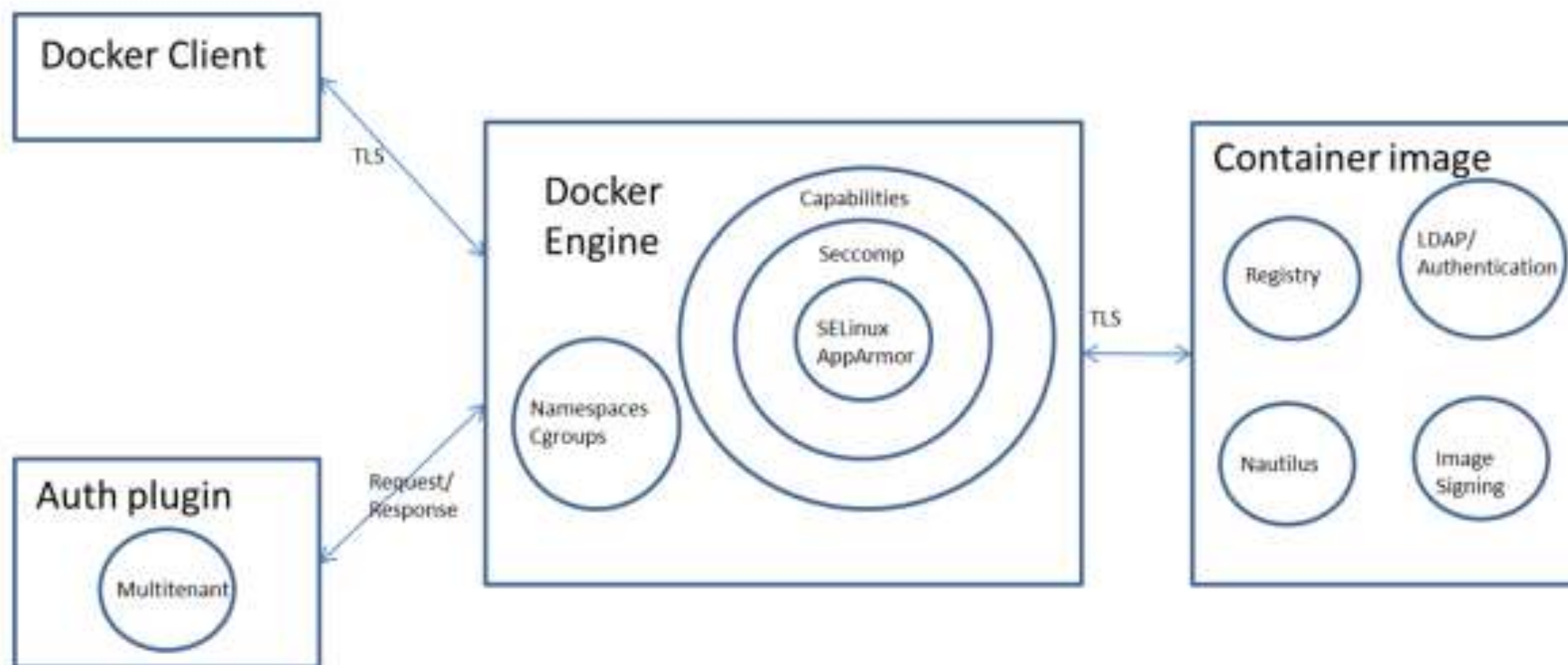
CONTAINERS



1.4 OCI 标准



1.5 容器安全



容器安全

- ❖ 镜像安全 Notary TUF (CNCF)
- ❖ 隔离不彻底 `/proc(lxcfs) /sys /dev time seLinux syslog` 内核级别未隔离
- ❖ 资源攻击 硬盘资源、网卡流量、Fork 💣
- ❖ 容器逃逸 通过 fd 接口或者漏洞获取宿主机权限

Docker 1.10 以后对于安全做了大量的工作

目录

❖ 1. 容器技术

- ❖ 1.1 容器进程与进程
- ❖ 1.2 容器文件系统
- ❖ 1.3 容器与虚拟机
- ❖ 1.4 OCI 标准
- ❖ 1.5 容器安全

❖ 2. 容器 Runtime

❖ 3. 容器引擎

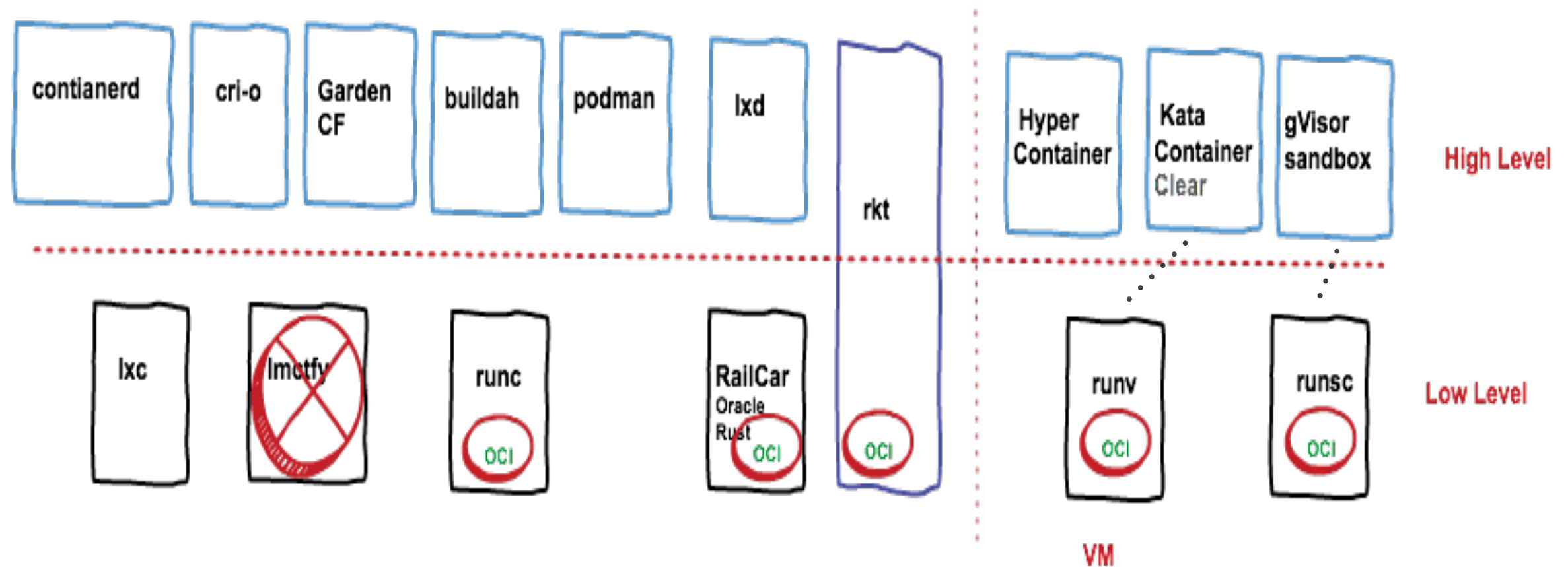
❖ 4. 容器编排

❖ 5. Kubernetes 与容器 Runtime

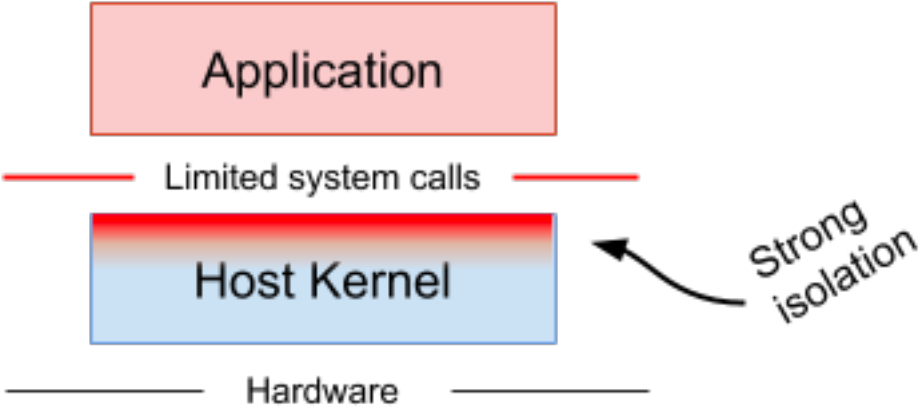
- ❖ 5.1 CRI 接口
- ❖ 5.2 集成方案总览
- ❖ 5.3 使用 containerd 集成
- ❖ 5.4 使用 cri-o 集成

2. 容器 Runtime

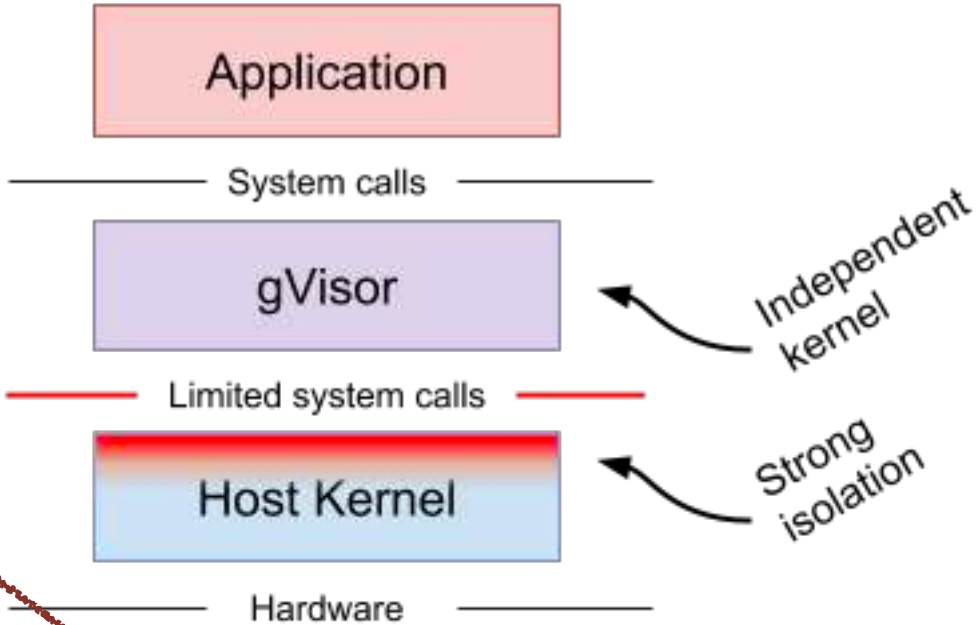
Container Runtime



seccomp/SELinux/AppArmor

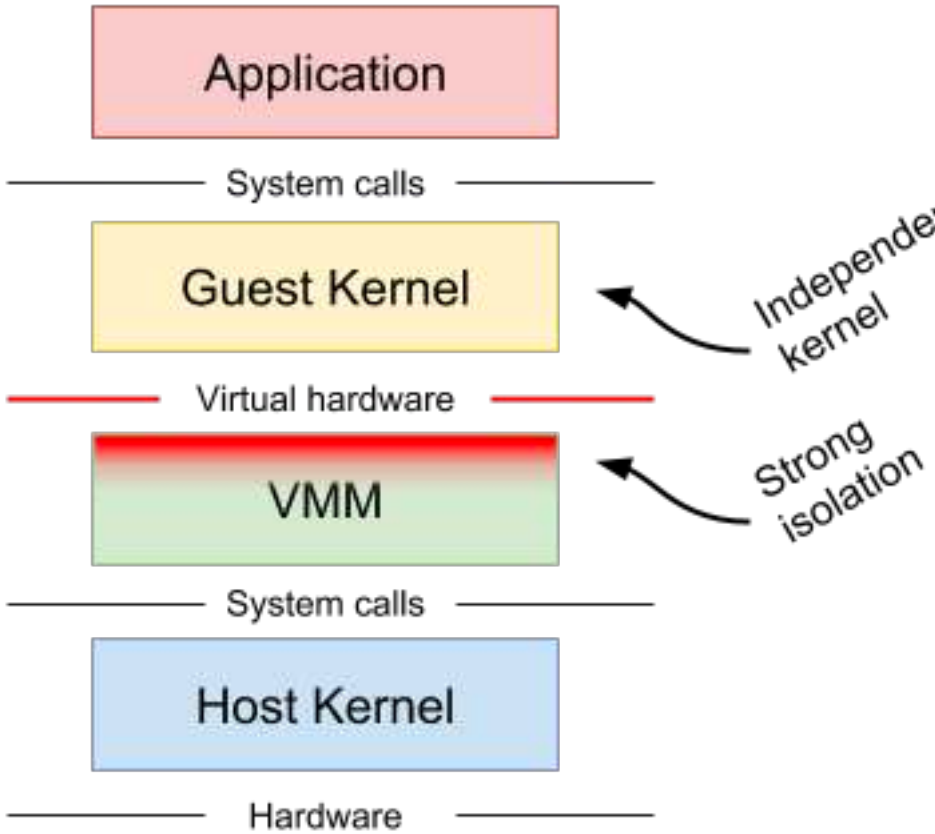


gVisor 2018.5

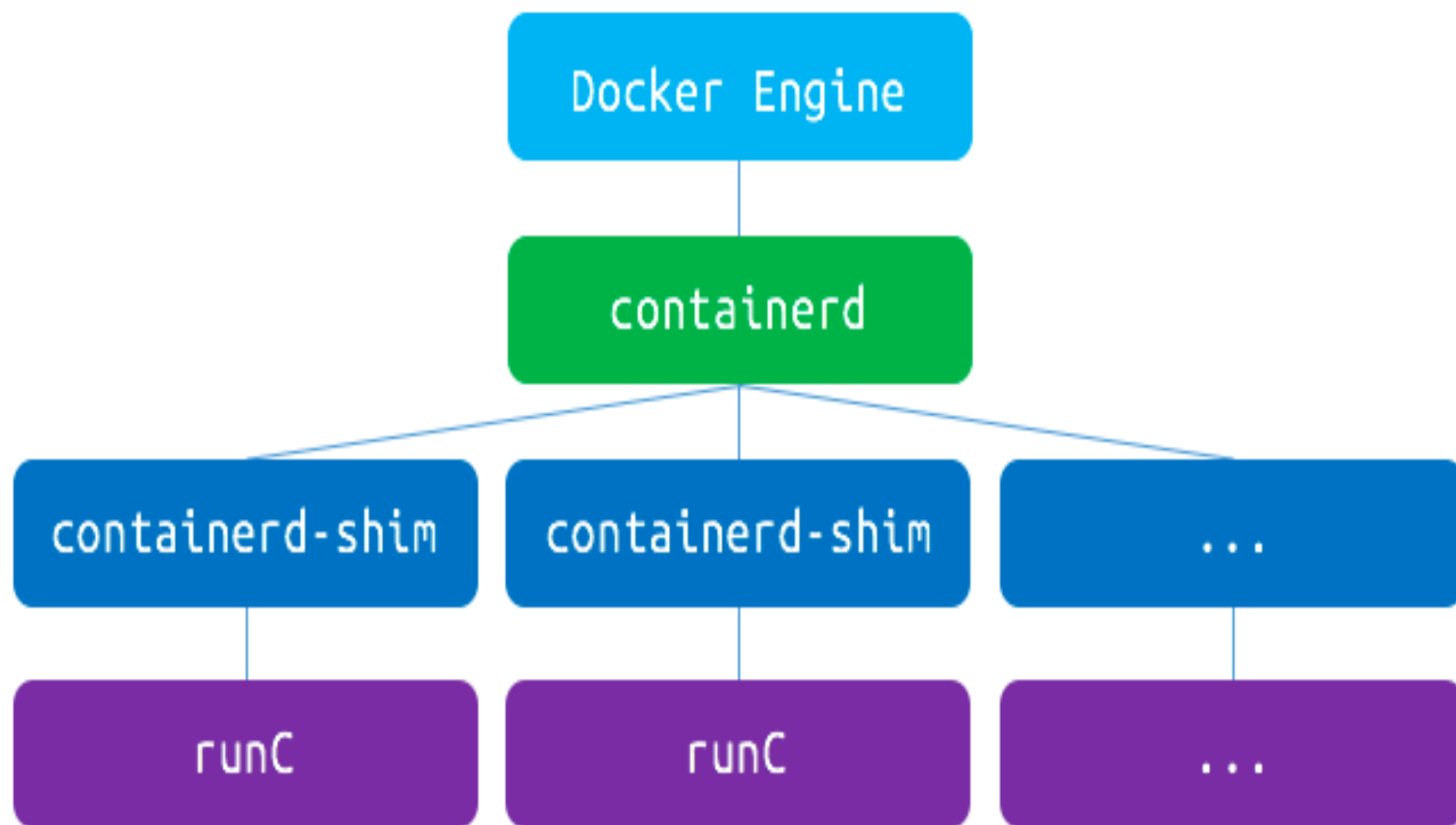


安全性和隔离度

VM



Docker 架构图



目录

❖ 1. 容器技术

- ❖ 1.1 容器进程与进程
- ❖ 1.2 容器文件系统
- ❖ 1.3 容器与虚拟机
- ❖ 1.4 OCI 标准
- ❖ 1.5 容器安全

❖ 2. 容器 Runtime

❖ 3. 容器引擎

❖ 4. 容器编排

❖ 5. Kubernetes 与容器 Runtime

- ❖ 5.1 CRI 接口
- ❖ 5.2 集成方案总览
- ❖ 5.3 使用 containerd 集成
- ❖ 5.4 使用 cri-o 集成

3. 容器引擎

- ❖ 容器引擎，或者说容器平台，不仅包含对于容器的生命周期的管理，还包括了对于容器生态的管理，比如对于镜像、存储、网络等相关的管理。
- ❖ 现在主要有： Docker、rkt、Pouch
- ❖ Docker 借助于其仓库和生态，最流行也最成熟。240亿下载，83% 份额
- ❖ PouchContainer， 是阿里内部基于阿里的运维现状、基于 LXC 技术重新实现了容器引擎。2017年11月 开源。核心亮点： P2P 镜像分发。

目录

❖ 1. 容器技术

- ❖ 1.1 容器进程与进程
- ❖ 1.2 容器文件系统
- ❖ 1.3 容器与虚拟机
- ❖ 1.4 OCI 标准
- ❖ 1.5 容器安全

❖ 2. 容器 Runtime

❖ 3. 容器引擎

❖ 4. 容器编排

❖ 5. Kubernetes 与容器 Runtime

- ❖ 5.1 CRI 接口
- ❖ 5.2 集成方案总览
- ❖ 5.3 使用 containerd 集成
- ❖ 5.4 使用 cri-o 集成

4. 容器编排

“

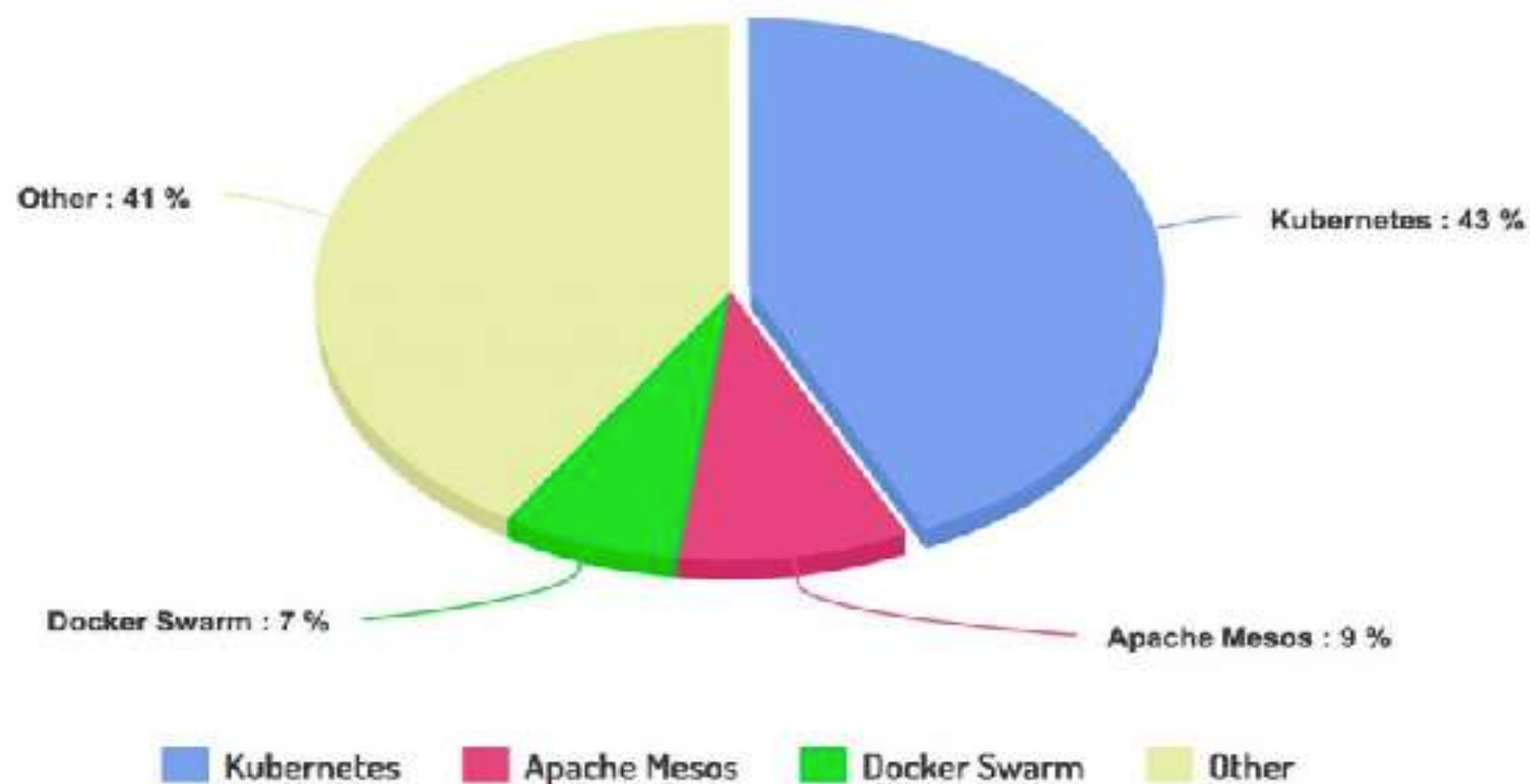
容器编排

是生产环境中成功部署和容器治理的关键。

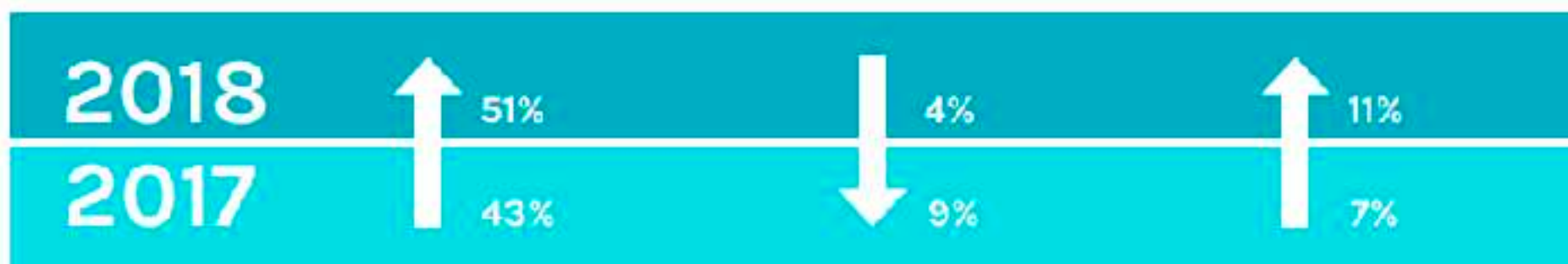
”



容器编排布局



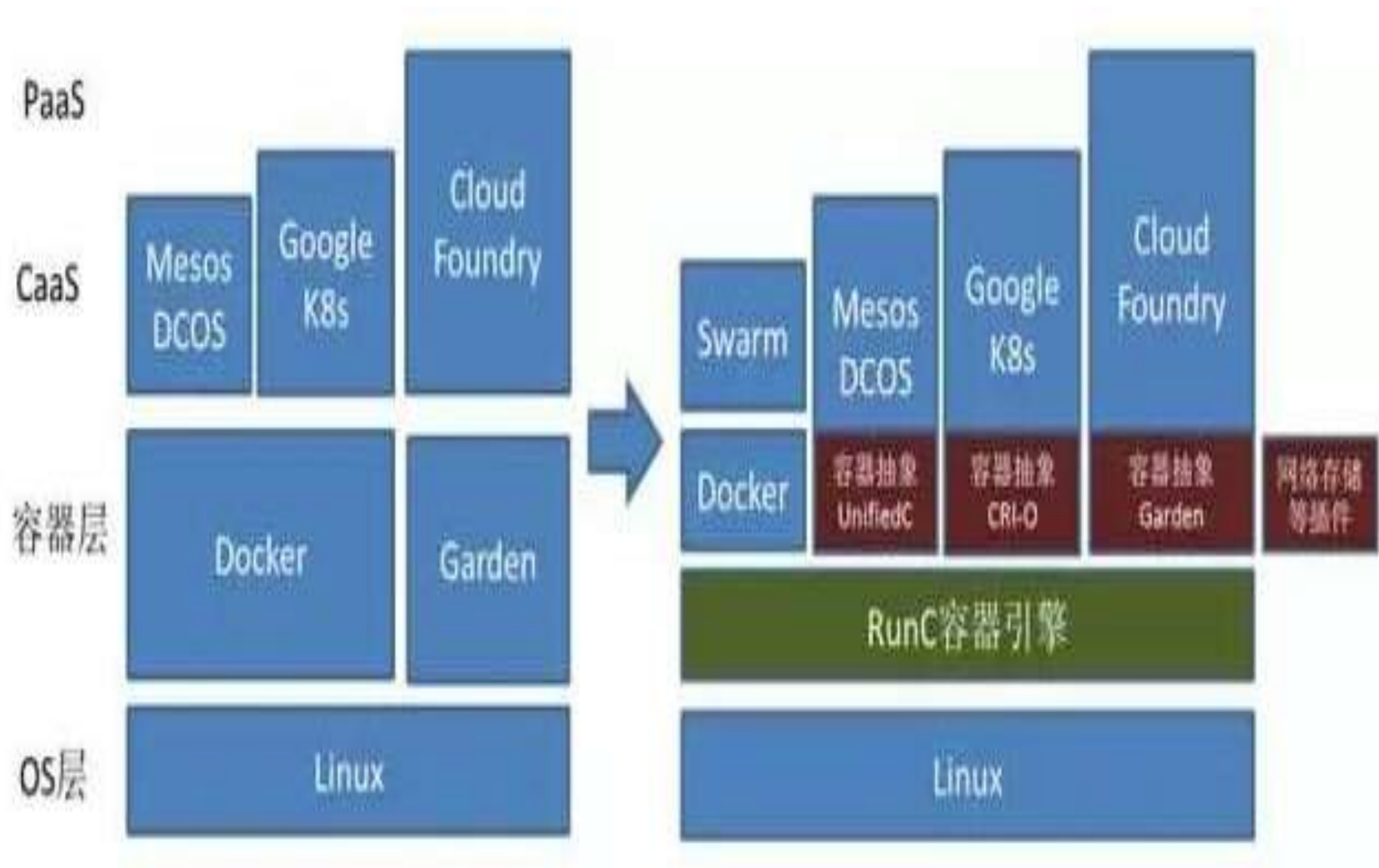
Container Orchestrators in Sysdig's 2017 Docker Usage Report



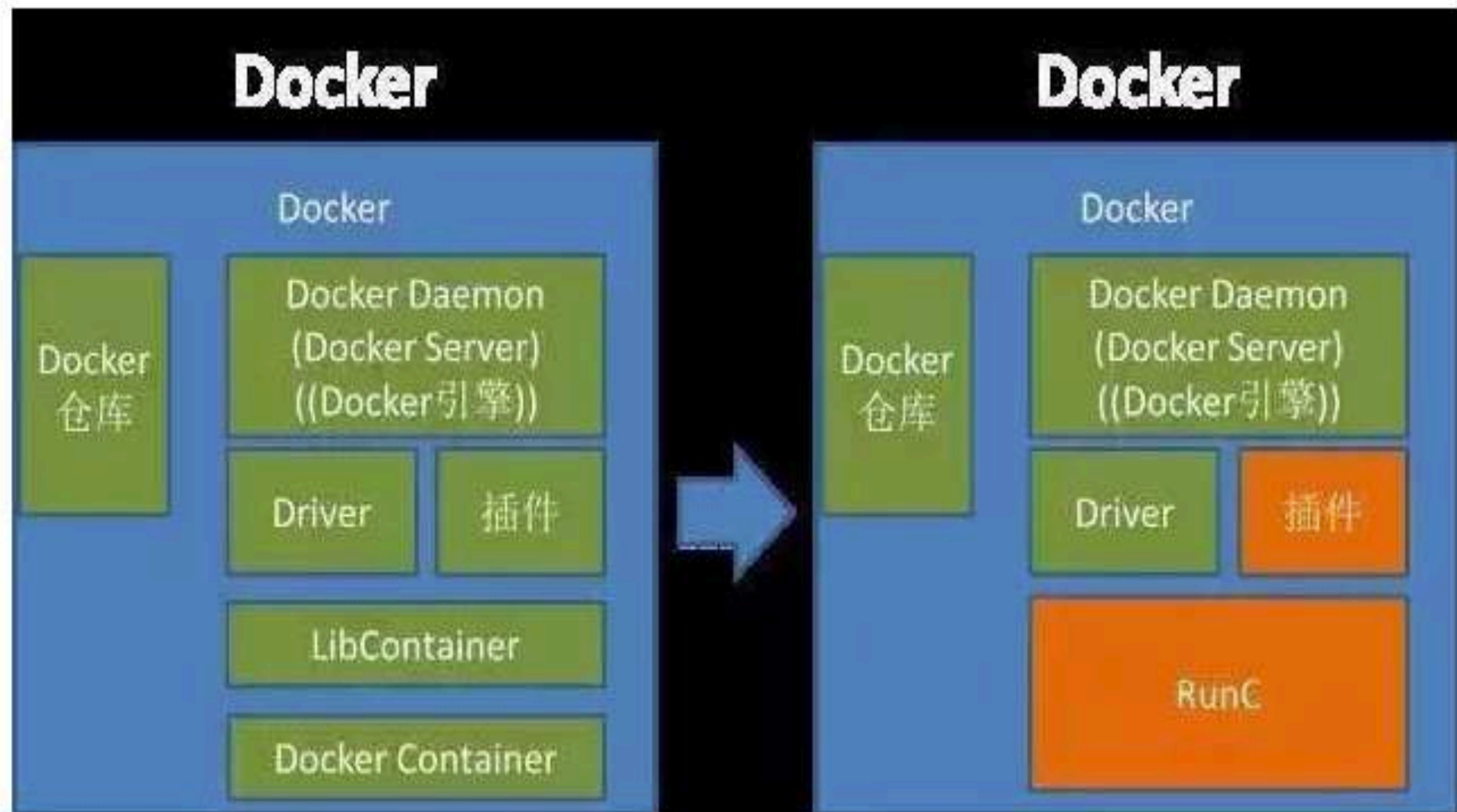
小插曲

- ❖ 从 2016 年 7 月底开始，Google Kubernetes 布道师 Kelsey Hightower 和 Docker 的 CTO Solomon Hykes 在 Twitter 上发生了一场撕逼大战，主题是要不要用 RunC 或其他容器来取代 Docker 引擎以及 OCI 的意义。
- ❖ Kelsey: “我一直相信 Docker 会给容器带来很多，但是我真的担心一个人想控制的太多”

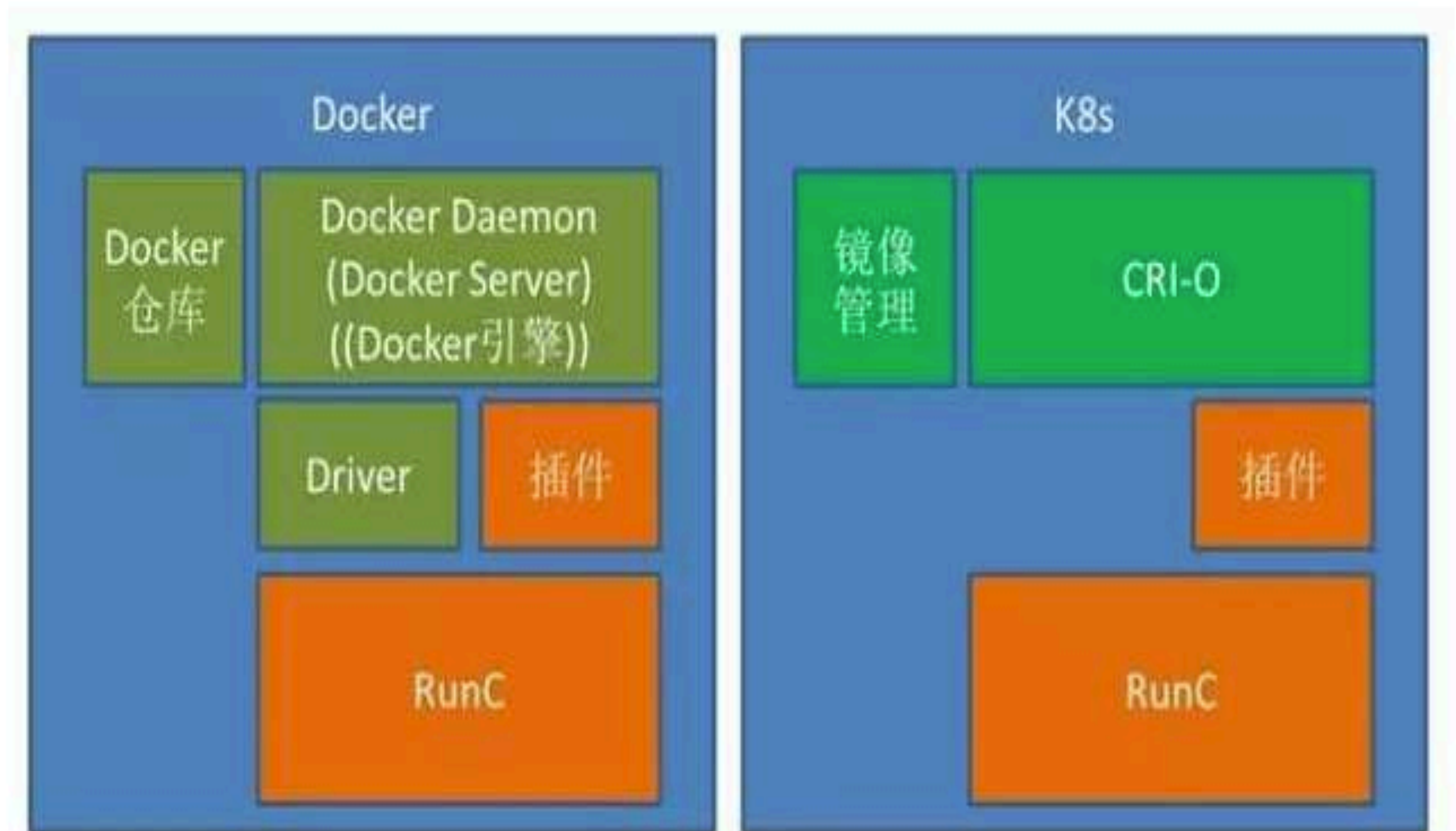
CaaS 架构演进



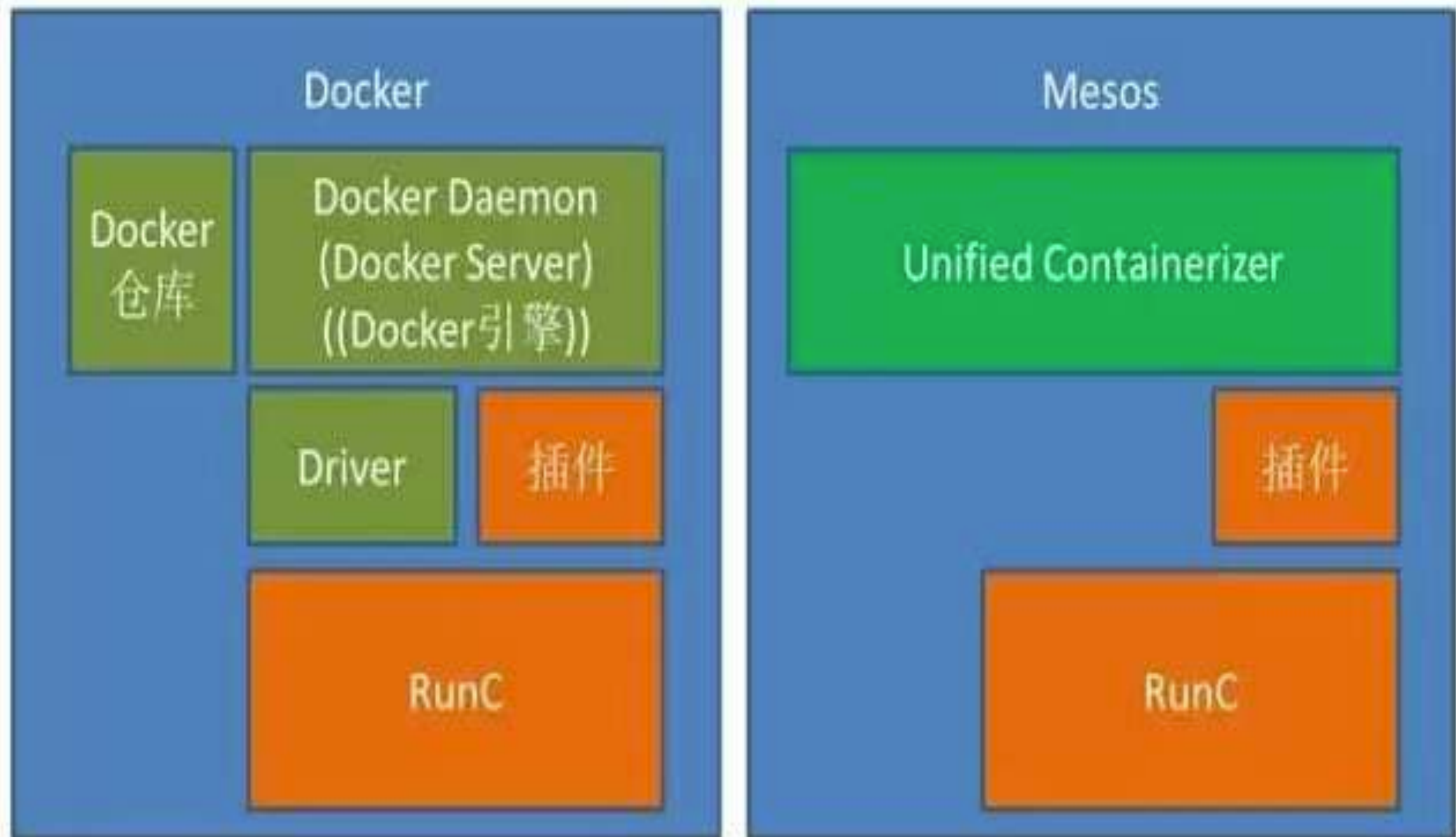
Docker



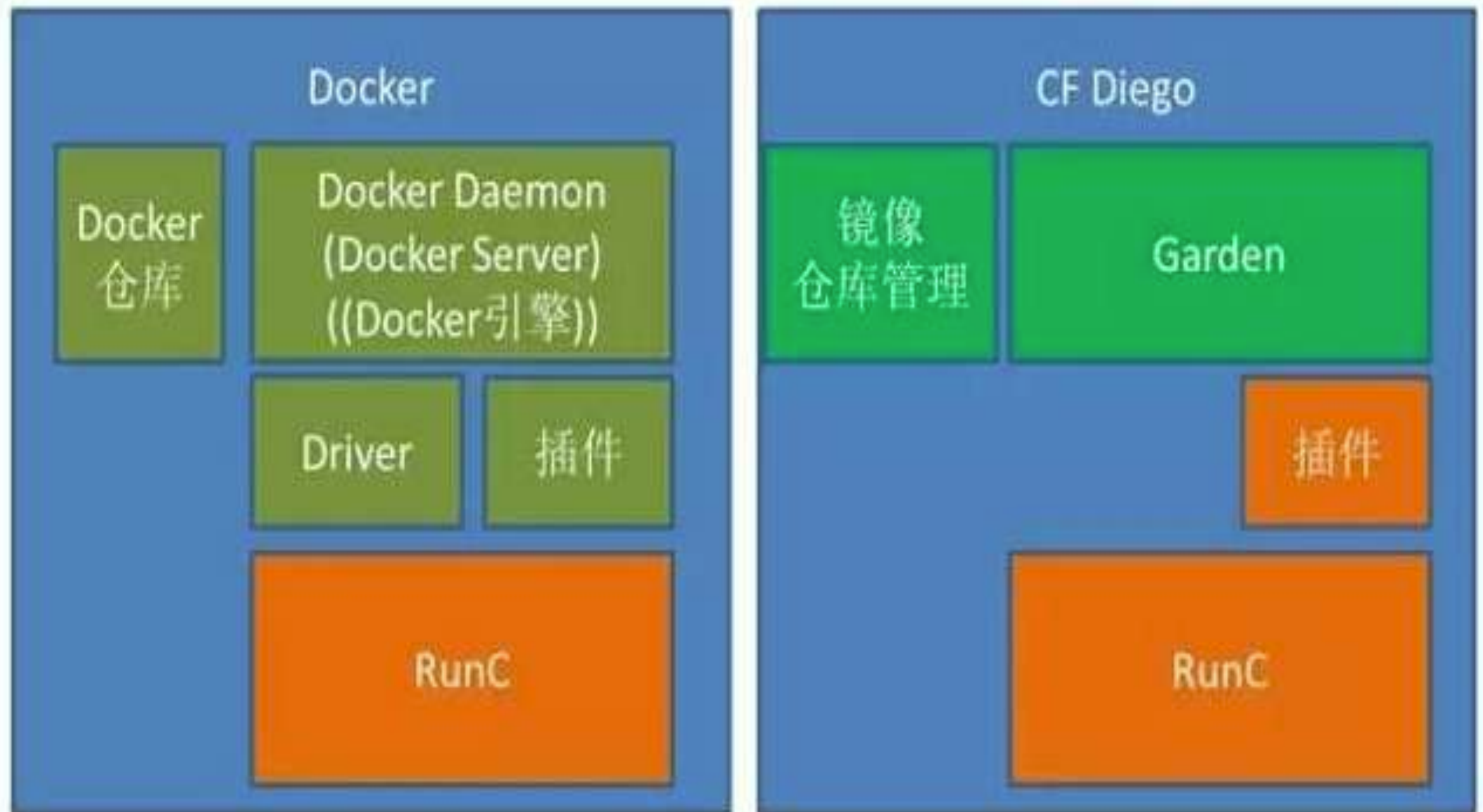
Kubernetes



Mesos



Could Foundry



目录

❖ 1. 容器技术

- ❖ 1.1 容器进程与进程
- ❖ 1.2 容器文件系统
- ❖ 1.3 容器与虚拟机
- ❖ 1.4 OCI 标准
- ❖ 1.5 容器安全

❖ 2. 容器 Runtime

❖ 3. 容器引擎

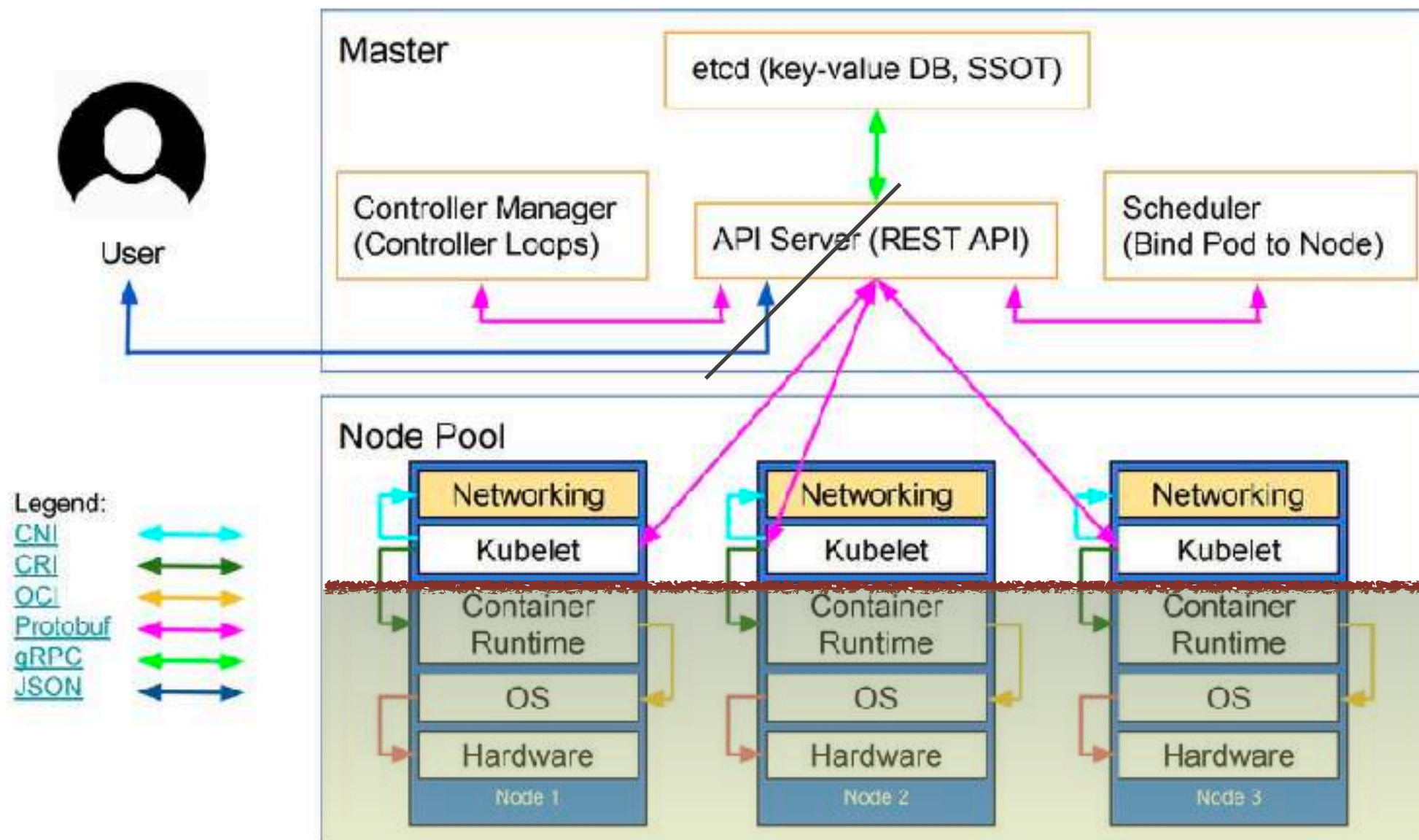
❖ 4. 容器编排

❖ 5. Kubernetes 与容器 Runtime

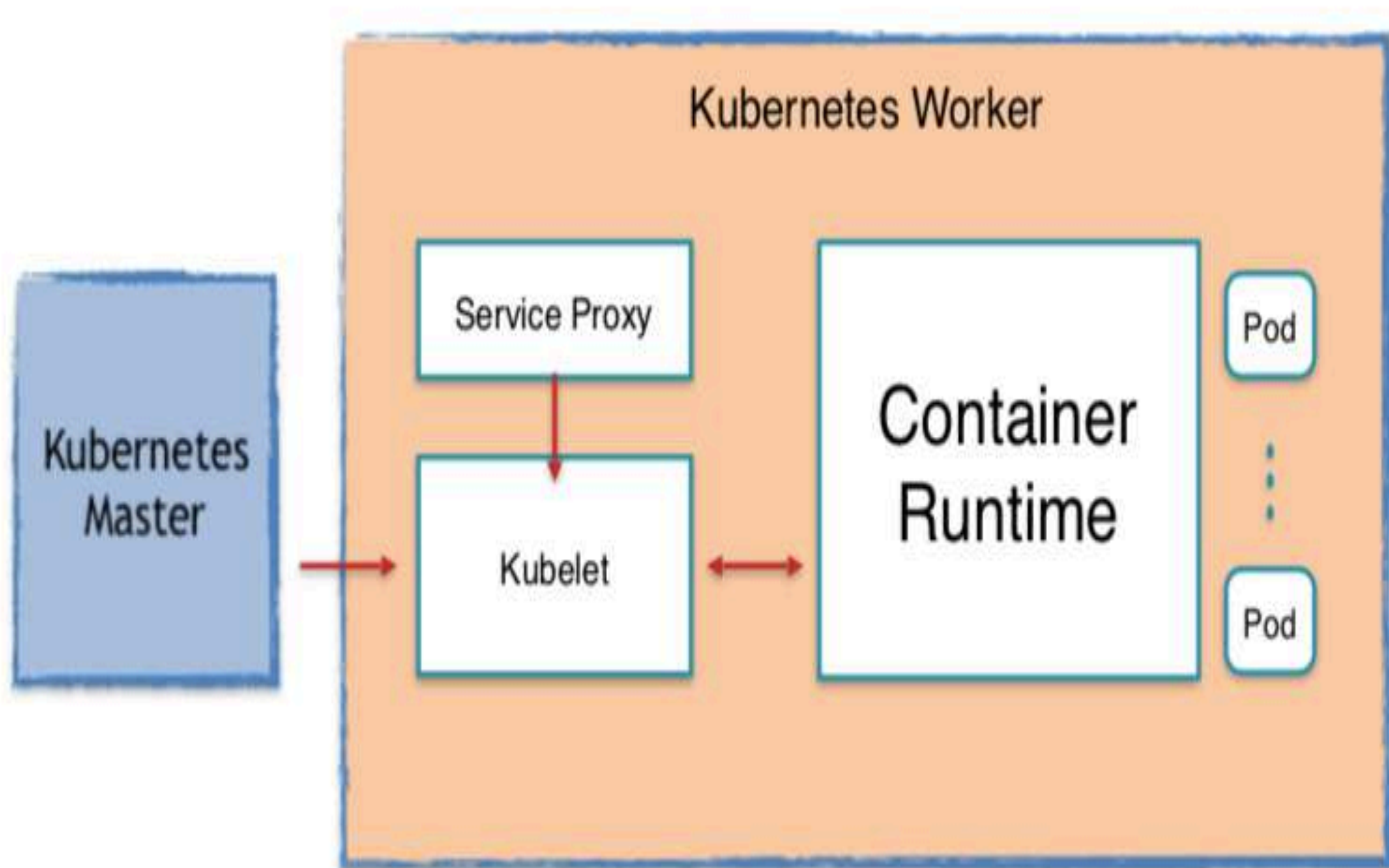
- ❖ 5.1 CRI 接口
- ❖ 5.2 集成方案总览
- ❖ 5.3 使用 containerd 集成
- ❖ 5.4 使用 cri-o 集成

5. Kubernetes 与 容器 Runtime

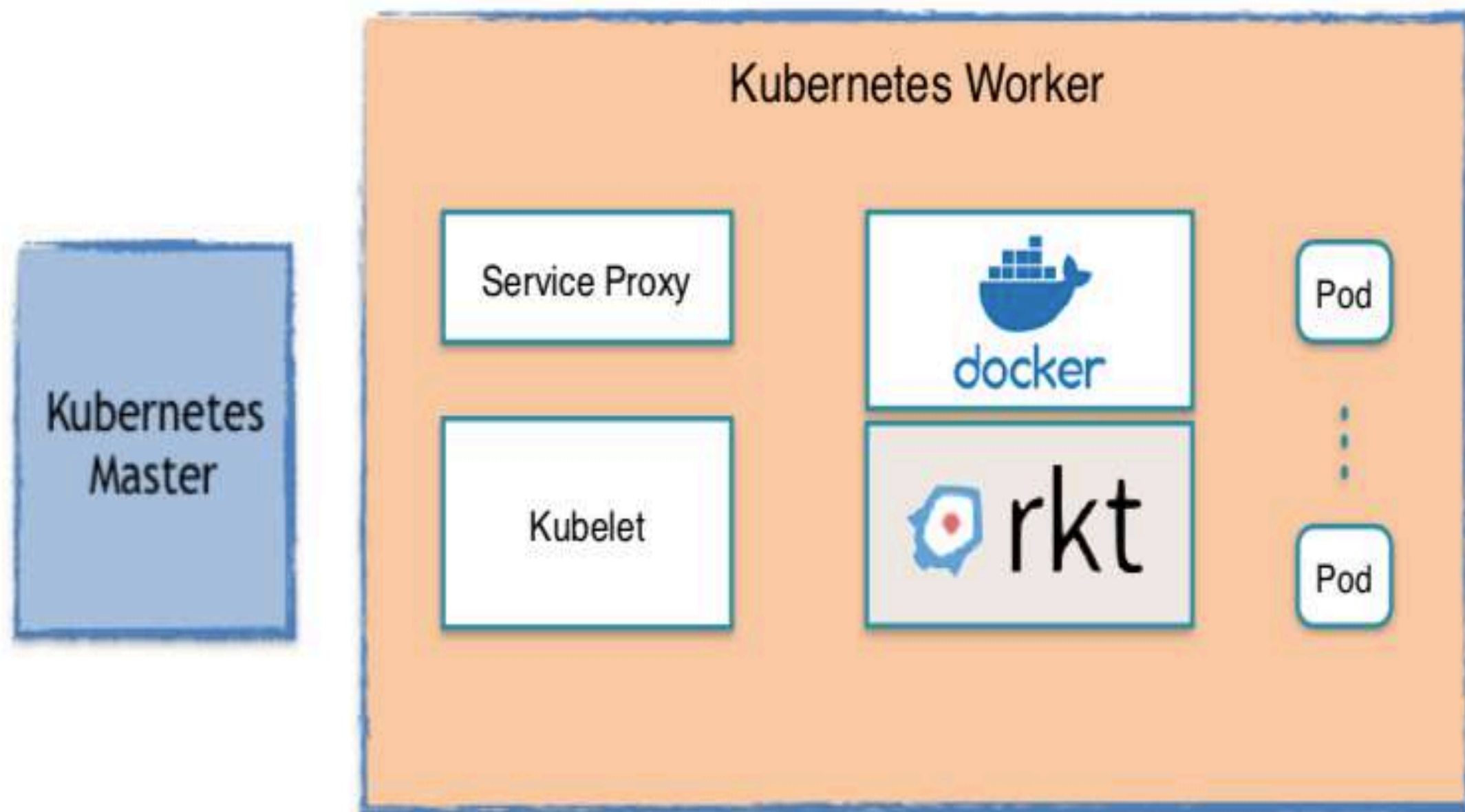
Kubernetes' high-level component architecture



5. Kubernetes 与 容器 Runtime



集成支持



5.1 CRI 接口 + CNI

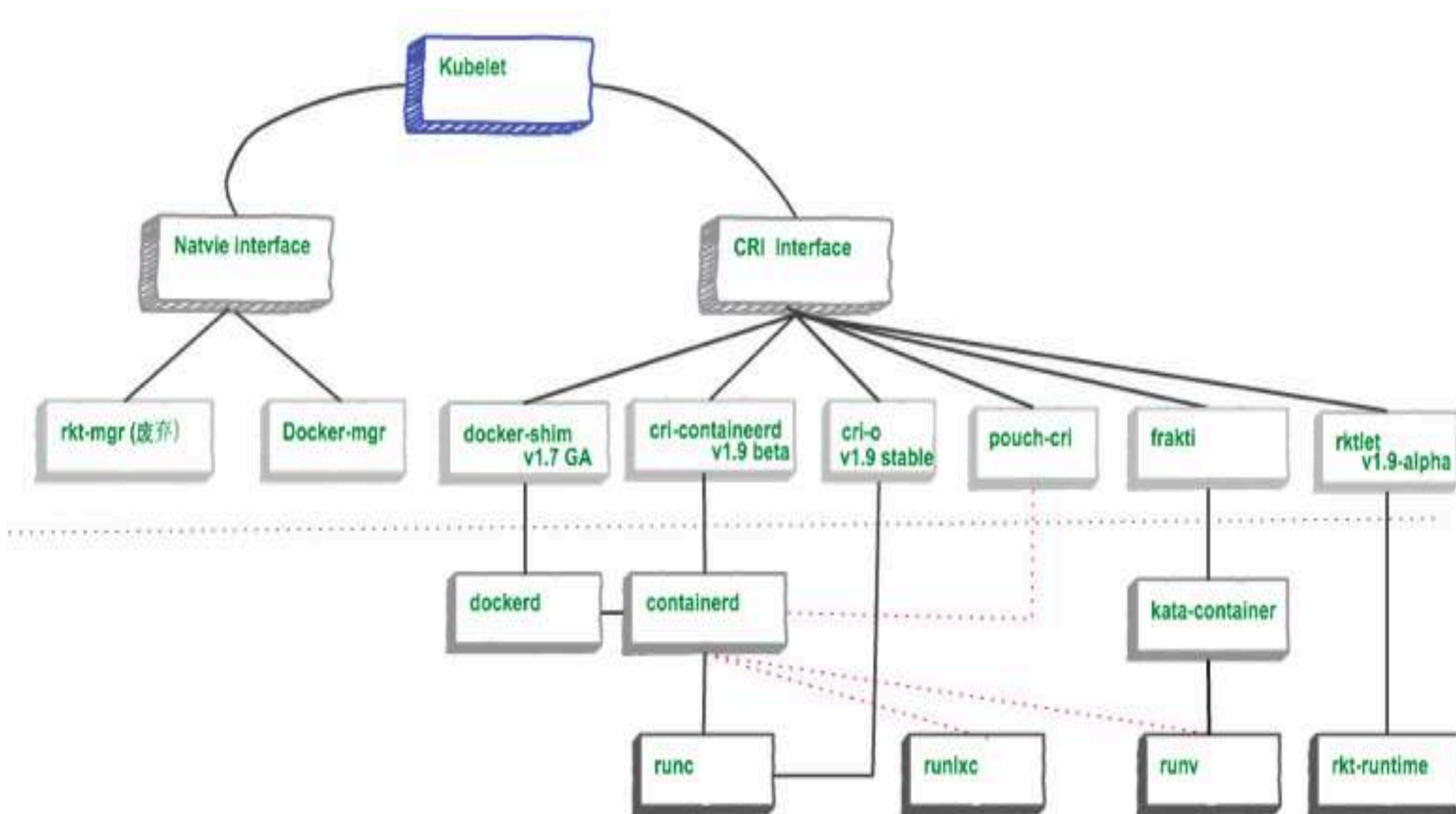
```
service RuntimeService {
    // Sandbox operations.
    rpc RunPodSandbox(RunPodSandboxRequest) returns (RunPodSandboxResponse) {}
    rpc StopPodSandbox(StopPodSandboxRequest) returns (StopPodSandboxResponse) {}
    rpc RemovePodSandbox(RemovePodSandboxRequest) returns (RemovePodSandboxResponse) {}
    rpc PodSandboxStatus(PodSandboxStatusRequest) returns (PodSandboxStatusResponse) {}
    rpc ListPodSandbox(ListPodSandboxRequest) returns (ListPodSandboxResponse) {}

    // Container operations.
    rpc CreateContainer(CreateContainerRequest) returns (CreateContainerResponse) {}
    rpc StartContainer(StartContainerRequest) returns (StartContainerResponse) {}
    rpc StopContainer(StopContainerRequest) returns (StopContainerResponse) {}
    rpc RemoveContainer(RemoveContainerRequest) returns (RemoveContainerResponse) {}
    rpc ListContainers(ListContainersRequest) returns (ListContainersResponse) {}
    rpc ContainerStatus(ContainerStatusRequest) returns (ContainerStatusResponse) {}

    ...
}

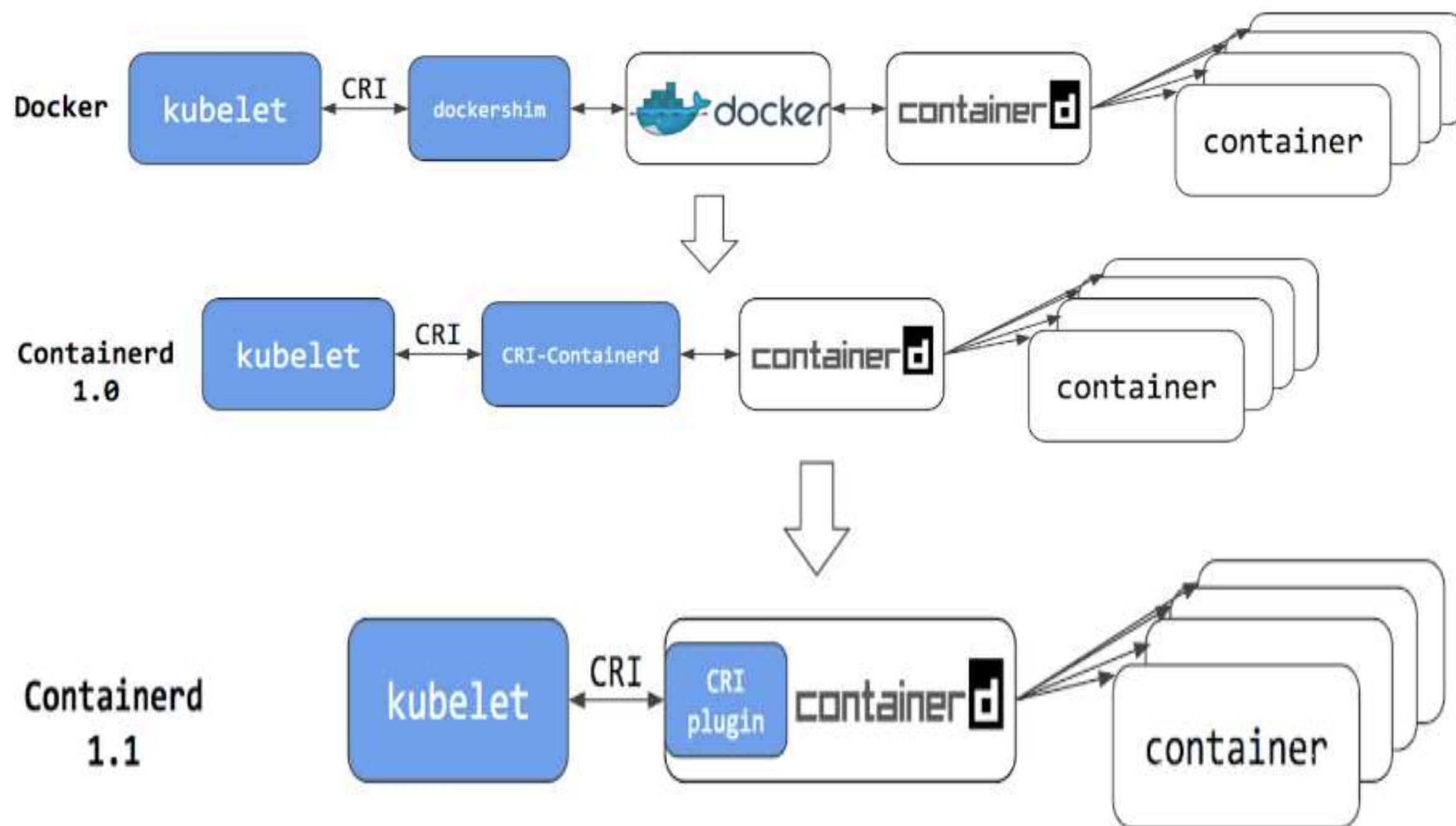
service ImageService {
    // ImageService defines the public APIs for managing images.
    rpc ListImages(ListImagesRequest) returns (ListImagesResponse) {}
    rpc ImageStatus(ImageStatusRequest) returns (ImageStatusResponse) {}
    rpc PullImage(PullImageRequest) returns (PullImageResponse) {}
    rpc RemoveImage(RemoveImageRequest) returns (RemoveImageResponse) {}
    rpc ImageFsInfo(ImageFsInfoRequest) returns (ImageFsInfoResponse) {}
}
```

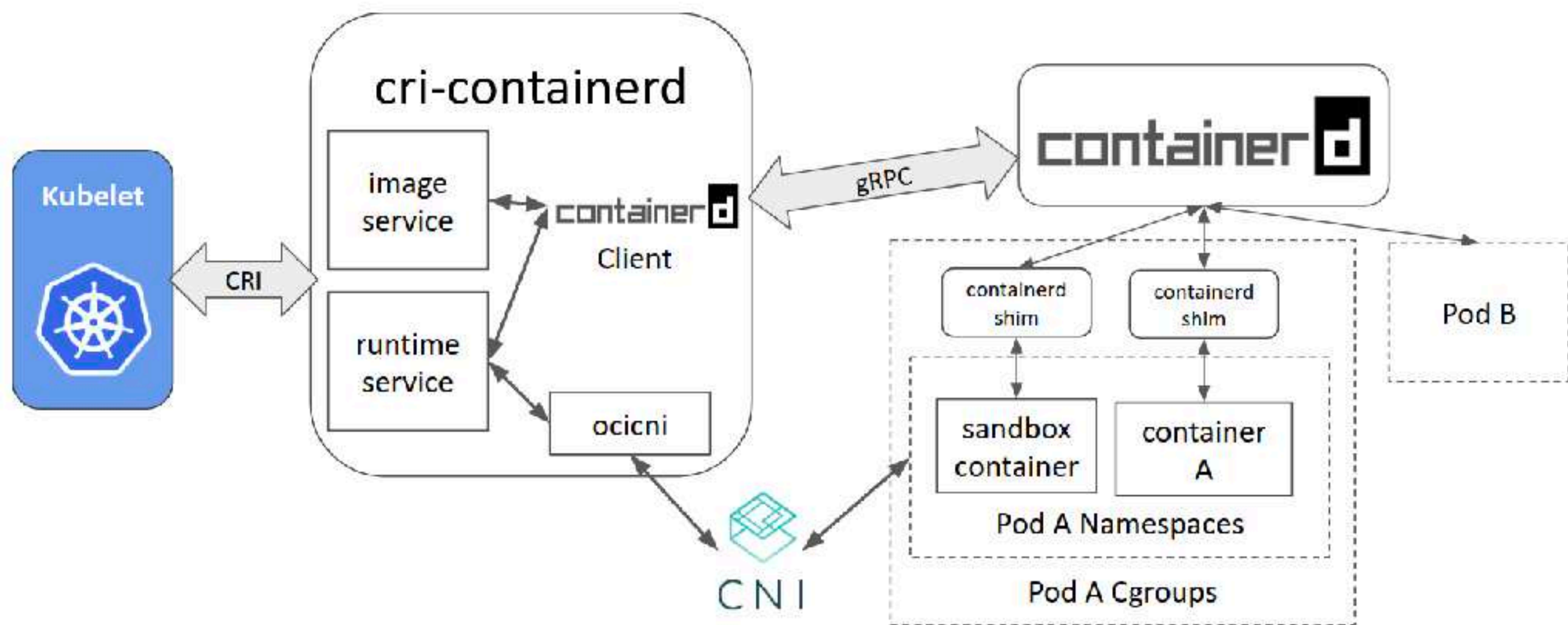
5.2 集成方案总览



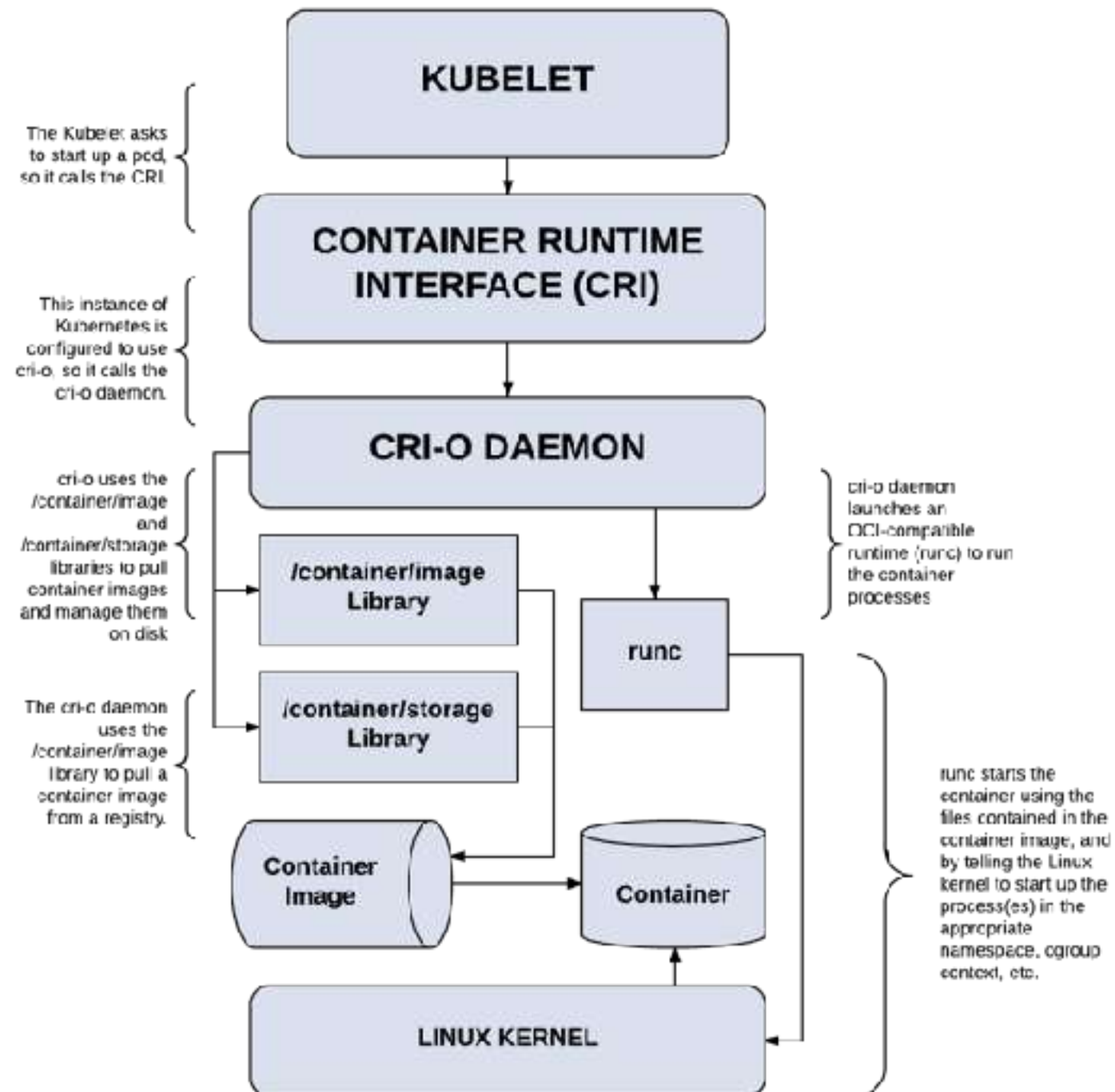
5.3 使用 containerd 集成

Containerd 1.0 - CRI-Containerd (end of life)





5.4 使用 cri-o 集成



Q & A

- ❖ 容器编排市场的三足鼎立到一家独大带来的启示?

Next:

“Kubernetes CNI 介绍”

Thank You !