从 docker 到 k8s

由进程说开去

王承锐
chengruiwang213807@sohu-inc.com

docker 是什么

docker = zip(os + Application)

- Docker是一个开放源代码软件项目，让应用程序布署在软件容器下的工作可以自动化进行，借此在Linux操作系统上，提供一个额外的软件抽象层，以及操作系统层虚拟化的自动管理机制。

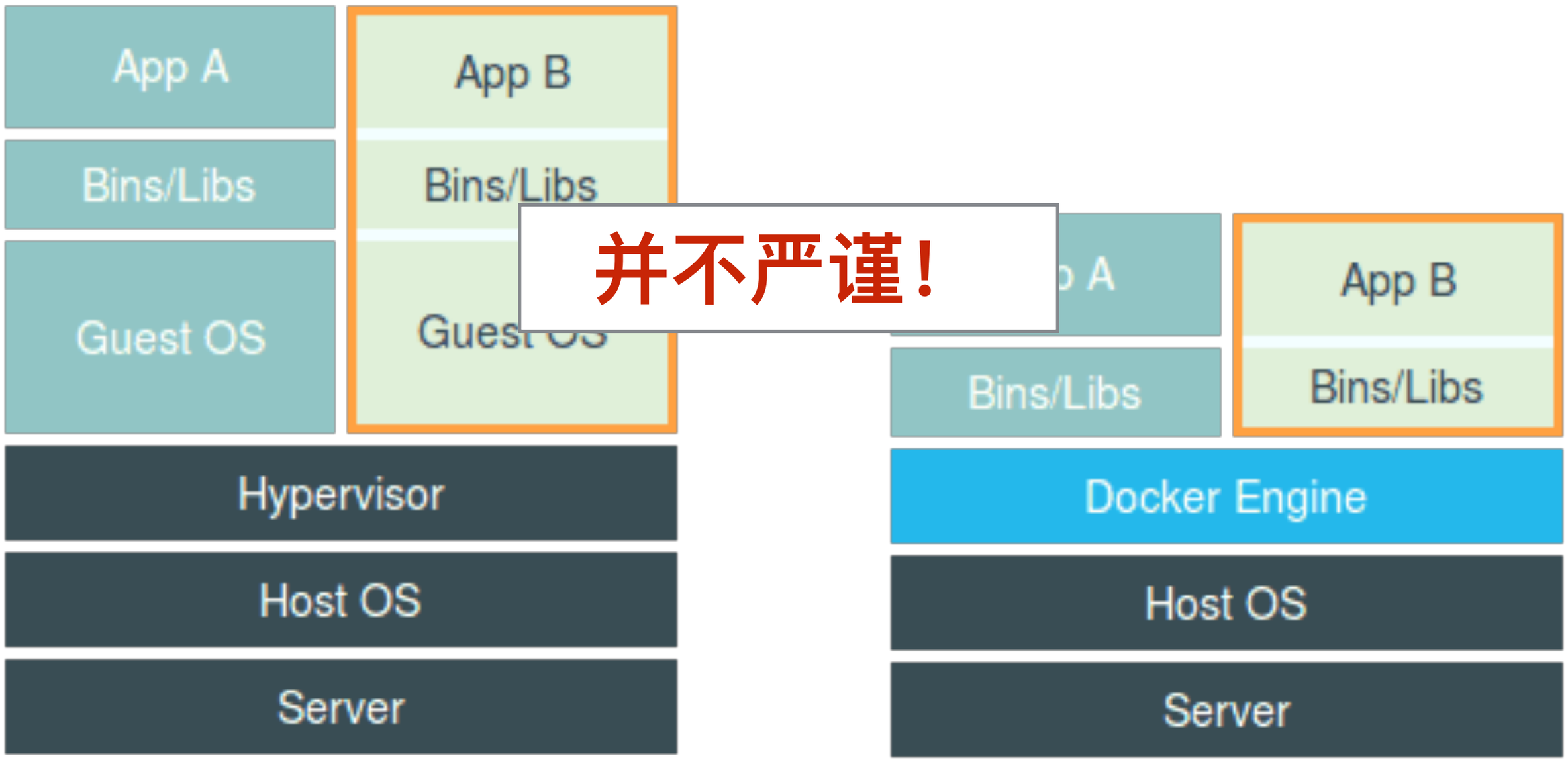
docker 以前的 pass

- 应用 + 部署脚本打包
- 部署困难，环境不可控
- namespace + group

docker

- 应用 + os 打包
- 高度一致
- namespace + cgroup

docker 和虚拟机的区别



docker 基本原理——Linux 容器

结论：docker 是跑在宿主机上的特殊进程

docker 基本原理——Linux 容器

- Linux 容器是提供多个隔离的 Linux 环境的操作系统级虚拟技术
- 容器们共享宿主机的内核
- 由于不需要专用的操作系统，因此容器要比虚拟器启动快得多

docker 基本原理——Linux 容器

- Namespace 修改进程视图
- Cgroups 制造进程资源约束
- Rootfs 提供进程隔离后执行环境

实验一

- Namespace 修改进程视图
- Cgroups 限制资源使用
- docker 资源使用限制

实验二

- clone 命令添加参数 =》 修改进程pid
- mount 挂载 /proc =》 进程视图隔离（ps、top命令）

实验三

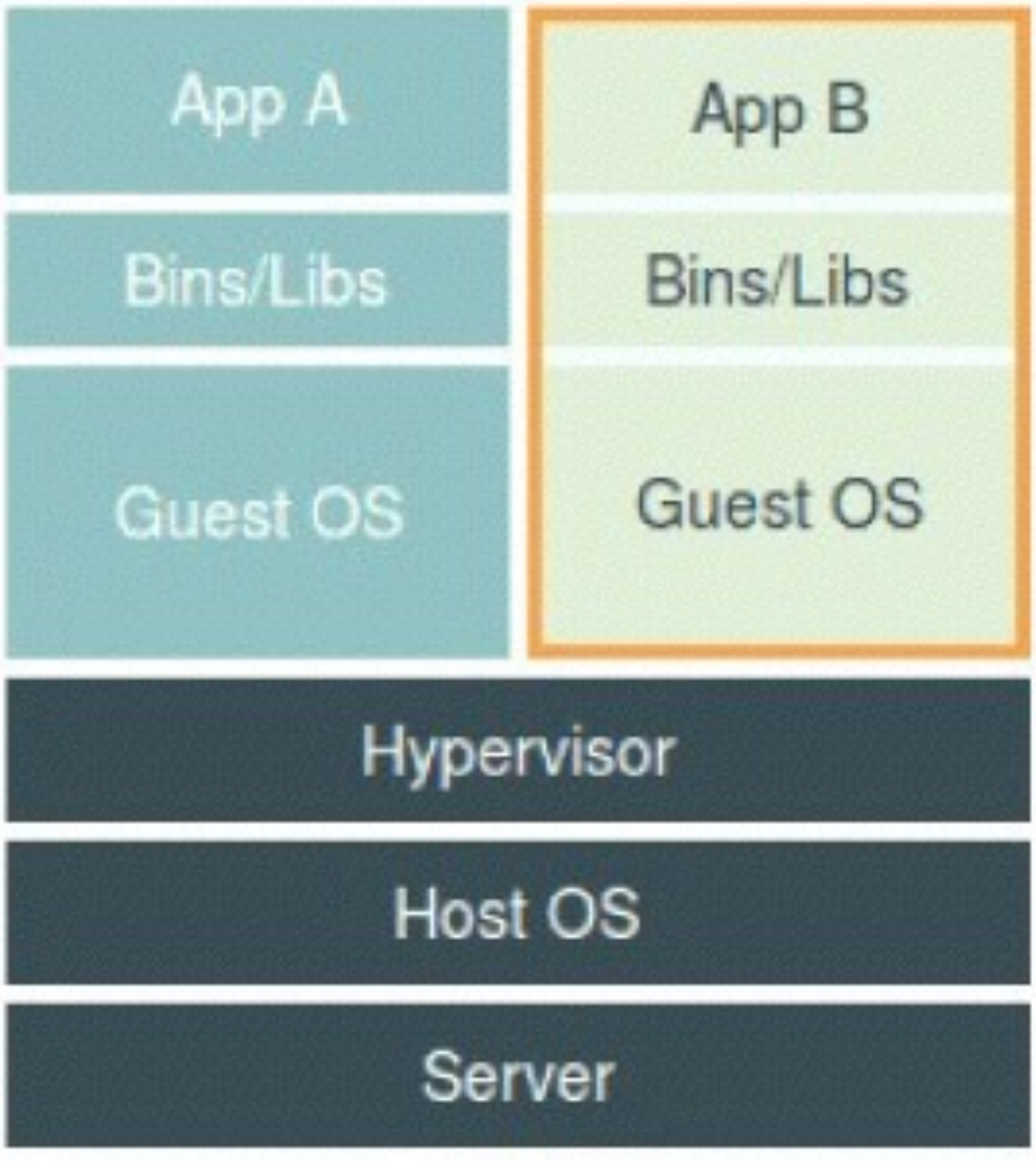
- 挂载文件 =》隔离后执行环境

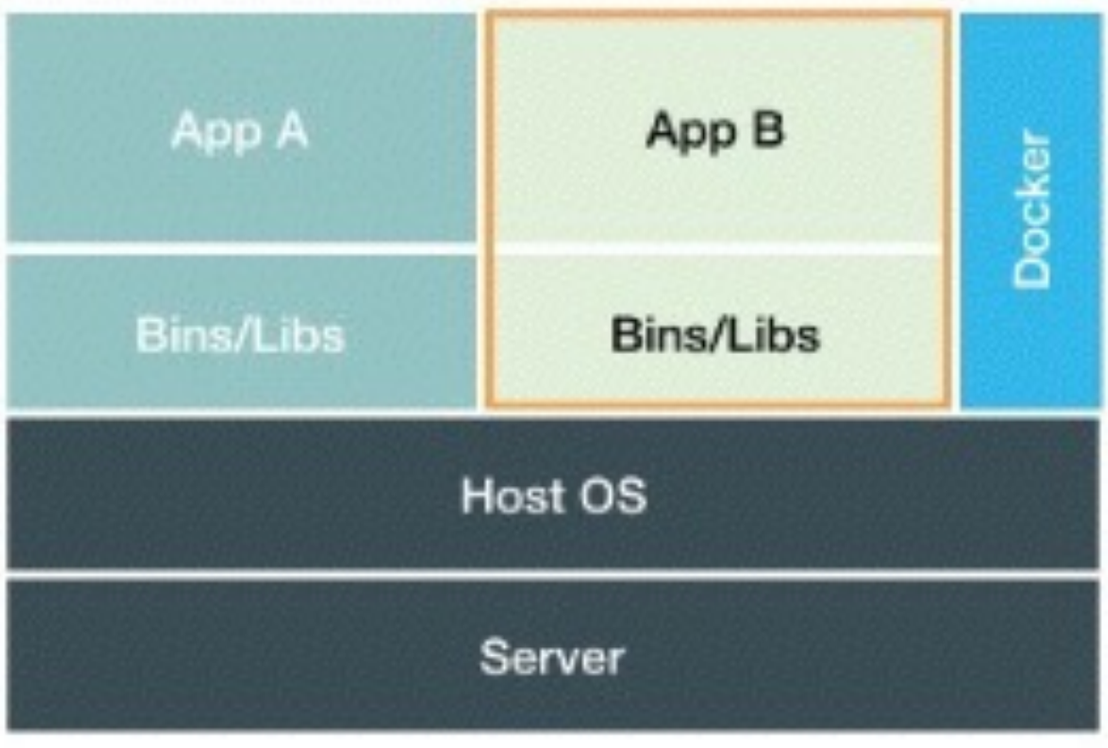
docker 存在的问题

- 多容器共享主机内核，所以低版本宿主机不能运行高版本容器，容器不能运行不同版本的内核

- 内核中的很多资源不能 namespace 化（时间），容器中修改了时间，宿主机也会被改变 —— 应用越狱

docker 和虚拟机的区别





docker（容器）并不重要，重要的是编排！

linux 容器

- 一组联合挂载在 /var/lib/docker/aufs/mnt 上的 rootfs，这一部分我们称为“容器镜像”（Container Image），是容器的静态视图；

- 一个由 Namespace+Cgroups 构成的隔离环境，这一部分我们称为“容器运行时”（Container Runtime），是容器的动态视图。

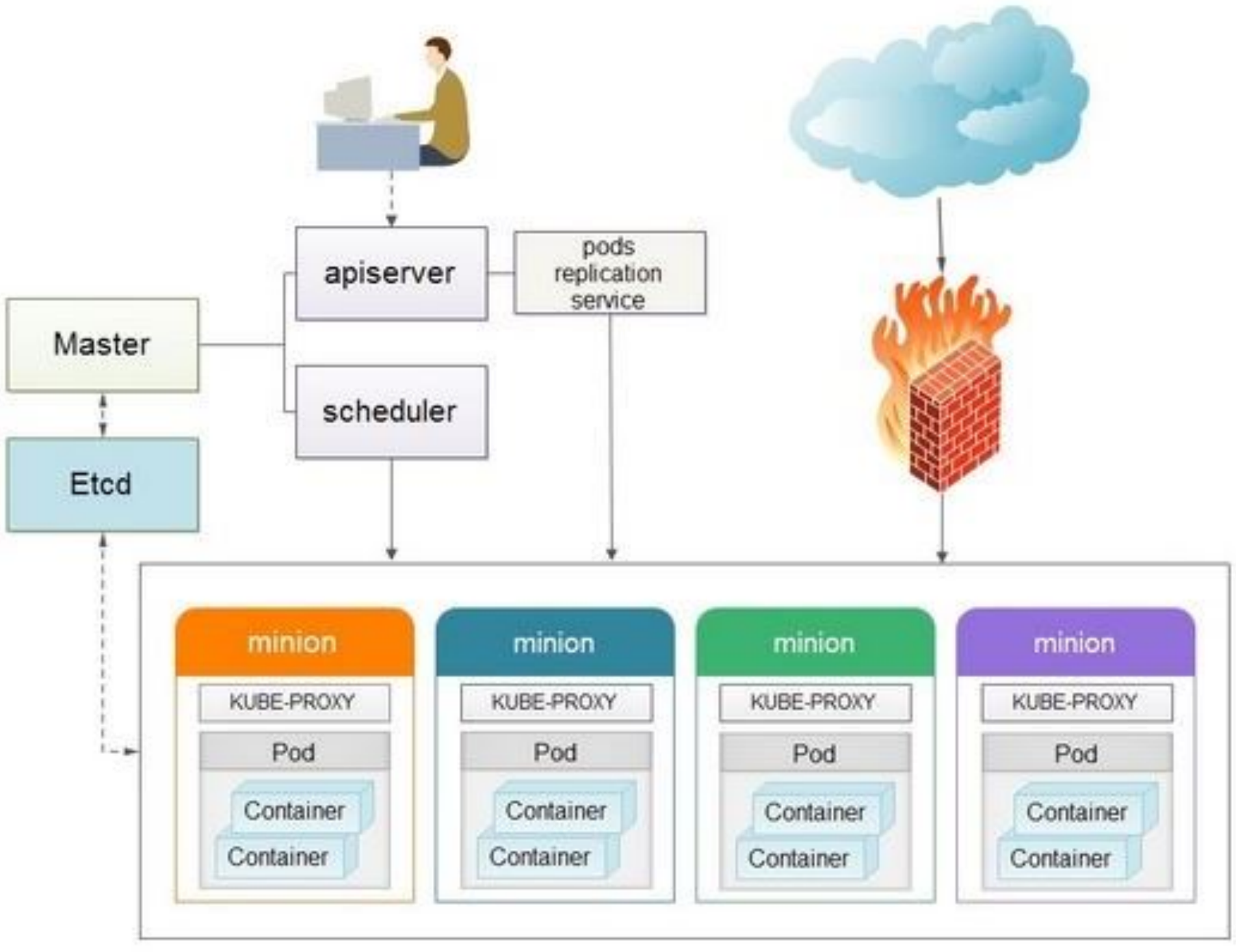
- 对于开发者，我并不关心容器运行时的差异。因为，在整个“开发 - 测试 - 发布”的流程中，真正承载着容器信息进行传递的，是容器镜像，而不是容器运行时。

- 所以我们现在的目标变成了如何运行，管理，编排容器镜像。

k8s

- Kubernetes（k8s）是自动化容器操作的开源平台，这些操作包括部署，调度和节点集群间扩展。

k8s 架构


apiserver
pods
replication
service
Master
scheduler
Etcd
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container

k8s 在焦点的应用

- 健康检查，资源限制，自动重启
- 扩容，升级，回滚
- 日志收集
- 服务发现
- 。。。。