Container

Background:

应用部署演化：**物理机时代——虚拟机时代——容器化时代**

**一、物理机时代**（单台物理主机，在这台物理机上需要安装一系列的应用部署安装和运行维护）

1. 部署非常慢

要采购主机，找地方存放，安装系统，安装各种组件

1. 成本高

一个外网的应用服务器一般都需要过万的价格进行采购

1. 资源浪费

资源不平衡，比如A机上安装了一个应用服务是对CPU要求非常高的，但是对内存磁盘的要求比较低，那我们的内存和磁盘就得不到应用

1. 难于扩展和迁移

对机器进行横向扩展是一件比较麻烦的事情，比如主机迁移

1. 受制于硬件

**二、虚拟机时代**（由于物理机部署的种种问题，虚级化技术就随之产生了，在一台物理主机上，通过硬件底层的虚拟化支持可以虚拟出多台虚拟主机，每一个虚拟机都是一台独立的电脑，都拥有自己的硬件（CPU，内存，硬盘），和操作系统（在操作系统上安装我们需要的软件）。虚拟化技术的出现可以让我们的物理机硬件资源得到更加充分的利用）

虚拟化技术的特点：

1. 多部署

一台主机上可以部署多个虚拟主机，各个虚拟主机间资源相互独立，任何一个虚拟机内部出现了问题都不会对其他虚拟机造成影响

1. 资源池

虚拟化时代把硬件资源看做一个池子，这个池子中的资源是可以做共享的，会有一个（虚拟机监视器 Hypervisor（vmWare是一种具体的实现））对物理机的所有资源进行统计，在需要进行新的虚拟机安装时由Hypervisor进行指定资源的分配，同时也对各个虚拟主机进行管理

1. 资源隔离
2. 很容易扩展
3. VM需要安装操作系统

当我们安装一个很小的应用时，需要单独部署，如果部署在虚拟机上时就需要先安装一个操作系统，只有安装了操作系统之后才能进行应用的部署。这时又造成了资源的浪费

1. **容器化时代** （为避免单一物理机低效的硬件利用和虚拟机低效的操作系统的利用）容器化技术应运而生。

Feature：

一、前身： Linux Container consists of Namespace and Cgroup

***Namespaces*** *are a feature of the Linux kernel that* ***partitions kernel resources*** *such that one set of processes sees one set of resources while another set of processes sees a different set of resources.”*

Namespaces are one of the technologies that containers are built on, used to enforce segregation of resources.

**A control group (cgroup)** is a Linux kernel feature that **limits, accounts for, and isolates the resource usage** (CPU, memory, disk I/O, network, and so on) of a collection of processes

* **Resource limits** – You can configure a cgroup to limit how much of a particular resource (memory or CPU, for example) a process can use.
* **Prioritization** – You can control how much of a resource (CPU, disk, or network) a process can use compared to processes in another cgroup when there is resource contention.
* **Accounting** – Resource limits are monitored and reported at the cgroup level.
* **Control** – You can change the status (frozen, stopped, or restarted) of all processes in a cgroup with a single command.

**Comparison**: Namespaces provide isolation of system resources, and cgroups allow for fine‑grained control and enforcement of limits for those resources.

1. 特点

* 1. 极其轻量：只打包了必要的Bin/Lib；
* 2. 秒级部署：根据镜像的不同，容器的部署大概在毫秒与秒之间（比虚拟机强很多）；
* 3. 易于移植：一次构建，随处部署；
* 4. 弹性伸缩：Kubernetes、Swam、Mesos这类开源、方便、好使的容器管理平台有着非常强大的弹性管理能力。

Standardization：

在2015年，由Google、 Docker、CoreOS、IBM、微软、红帽等厂商联合发起的OCI（Open Container Initiative）组织成立了，并于2016年4月推出了第一个开放容器标准。标准主要包括runtime运行时标准和image镜像标准。

1.容器运行时标准 （runtime spec）

2.容器镜像标准（image spec）

There are 3 major standards to ensure interoperability of container technologies—the OCI Image, Distribution, and Runtime specifications.

Application：

1. 容器化传统应用 容器不仅能提高现有应用的安全性和可移植性，还能节约成本
2. 持续集成和持续部署 (CI/CD) 通过 Docker 加速应用管道自动化和应用部署，交付速度提高至少 13 倍
3. 微服务 加速应用架构现代化进程
4. IT 基础设施优化 充分利用基础设施，节省资金。