# 基础技术-容器化能力选型方案

# 目 录

## 背景和目标

随着研发团队的发展和项目的交付，机器运维的工作日益增加，包括对内部服务器的管理以及对甲方服务器的管理。随之而来的是硬件资源利用率低，安装部署步骤复杂等问题。为了满足统一的服务器管理，提升硬件资源利用率，简化运维流程等要求，现对容器化技术做技术选型。

### 容器化

Linux Container容器技术的诞生（2008年）就解决了IT世界里“集装箱运输”的问题。Linux Container（简称LXC）它是一种内核轻量级的操作系统层虚拟化技术。Linux Container主要由Namespace和Cgroup两大机制来保证实现。Namespace也是一样的作用，做隔离。光有隔离还没用，我们还需要对货物进行资源的管理。Cgroup就负责资源管理控制作用，比如进程组使用CPU/MEM的限制，进程组的优先级控制，进程组的挂起和恢复等等。

容器化具有如下特点：

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。

### 容器标准化

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。标准的推出，有助于替成长中市场带来稳定性，让企业能放心采用容器技术，用户在打包、部署应用程序后，可以自由选择不同的容器Runtime。容器标准化包含如下两方面：

1. 运行时标准

运行时标准规范容器本身是如何被创建，更新，销毁的。

1. 镜像标准

镜像标准规范容器如何存储镜像数据，相关meta数据，应用运行的数据等。

## 方案简介

目前主流的容器化方案包括Docker（Docker Swarm），Kubernetes（K8S）和Mesos等。Docker是最早开源的容器化解决方案并且现有的容器化标准是基于docker进化而来; K8S是Google主导的容器化方案，前身来源于Google内部的容器化经验并且在Google公司得到了验证;Mesos是Apache基金会主导的开源项目，它不仅仅是简单的容器管理，更是集机器管理，资源抽象以及容器化等一体的综合解决方案。

由于团队本身已经对Docker有一定的认识和实践经验，本选型方案就Docker Swarm和K8S做比较，Mesos本身过于庞大，目前暂不作为选型候选。Docker是底层的运行时和镜像的总称而Docker Swarm则是基于Docker产生的管理平台; K8S则是基于标准运行时（包括Docker, Containerd和Cri-o等）。

### 方案对比

本章就易用性，健壮性，灵活性和简易性等方面做比较，选择适合团队本身的容器化方案。

|  |  |  |
| --- | --- | --- |
| 比对项\方案 | Docker Swarm | K8S |
| 集群部署 | √ | √ |
| 自动Replication | √ | √ |
| 数据持久化 | 插件支持 | 插件支持 |
| 监控机制 | 开源方案支持 | 开源方案支持 |
| 标准运行时支持 | X | √ |
| 开源社区支持 | √ | √ |
| 镜像仓库 | √ | 兼容多个私有仓库实现 |
| HA | √ | √ |
| LoadBalance | √ | √（功能更强大） |
| 发布策略 | 自动渐进更新 | 自动渐进更新 |
| 服务发现 | 通过IP和DNS实现 | 通过etcd实现 |
| 健康检查 | 命令行方式检测 | 命令行方式检测  http方式检测  TCP方式检测  检测存活和可用 |
| 学习曲线 | 低 | 中 |
| 集群隔离 | X | √ |

通过上述表格比较，K8S和Docker Swarm功能相似，K8S更适合复杂情况下的容器调度，功能也更丰富。因此选用K8S作为容器化的基础方案。