1 第一章 kubernetes介绍

1.1 应用部署方式演变

在部署应用程序的方式上，主要经历了三个时代：

- 传统部署：互联网早期，会直接将应用程序部署在物理机上

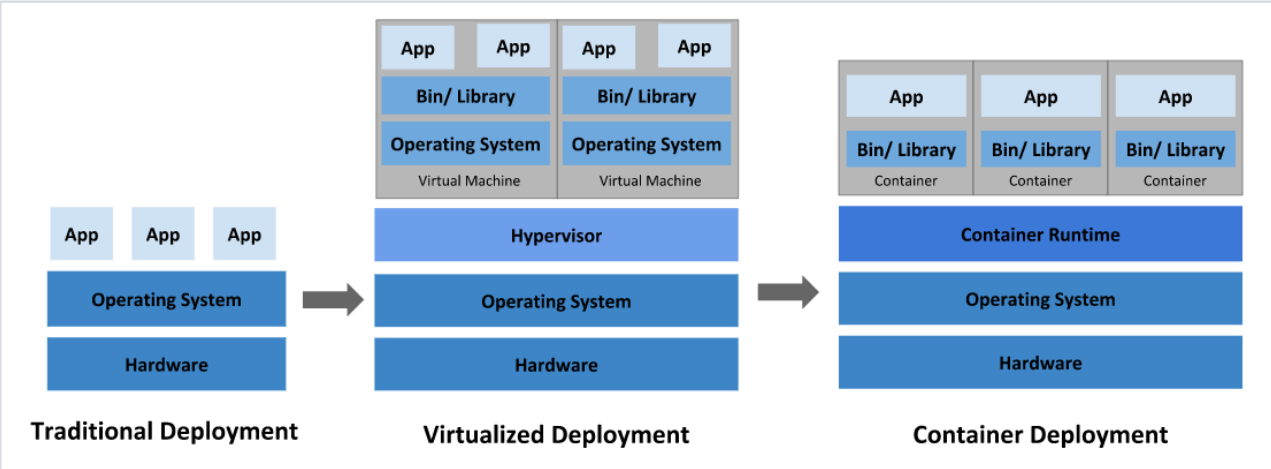
  > 优点：简单，不需要其它技术的参与
  >
  > 缺点：不能为应用程序定义资源使用边界，很难合理地分配计算资源，而且程序之间容易产生影响

- 虚拟化部署：可以在一台物理机上运行多个虚拟机，每个虚拟机都是独立的一个环境

  > 优点：程序环境不会相互产生影响，提供了一定程度的安全性
  >
  > 缺点：增加了操作系统，浪费了部分资源

- 容器化部署：与虚拟化类似，但是共享了操作系统

  > 优点：可以保证每个容器拥有自己的文件系统、CPU、内存、进程空间等运行应用程序所需要的资源都被容器包装，并和底层基础架构解耦容器化的应用程序可以跨云服务商、跨Linux操作系统发行版进行部署



容器化部署方式给带来很多的便利，但是也会出现一些问题，比如说：

- 一个容器故障停机了，怎么样让另外一个容器立刻启动去替补停机的容器
- 当并发访问量变大的时候，怎么样做到横向扩展容器数量

这些容器管理的问题统称为**容器编排**问题，为了解决这些容器编排问题，就产生了一些容器编排的软件：

- Swarm：Docker自己的容器编排工具
- Mesos：Apache的一个资源统一管控的工具，需要和Marathon结合使用
- Kubernetes：Google开源的的容器编排工具

1.2 kubernetes简介

kubernetes，是一个全新的基于容器技术的分布式架构领先方案，是谷歌严格保密十几年的秘密武器----Borg系统的一个开源版本，于2014年9月发布第一个版本，2015年7月发布第一个正式版本。

kubernetes的本质是**一组服务器集群**，它可以在集群的每个节点上运行特定的程序，来对节点中的容器进行管理。目的是实现资源管理的自动化，主要提供了如下的主要功能：

- 自我修复：一旦某一个容器崩溃，能够在1秒中左右迅速启动新的容器
- 弹性伸缩：可以根据需要，自动对集群中正在运行的容器数量进行调整
- 服务发现：服务可以通过自动发现的形式找到它所依赖的服务
- 负载均衡：如果一个服务起动了多个容器，能够自动实现请求的负载均衡
- 版本回退：如果发现新发布的程序版本有问题，可以立即回退到原来的版本
- 存储编排：可以根据容器自身的需求自动创建存储卷

1.3 kubernetes组件

一个kubernetes集群主要是由**控制节点(master)**、**工作节点(node)**构成，每个节点上都会安装不同的组件。

master：集群的控制平面，负责集群的决策（ 管理 ）

> ApiServer : 资源操作的唯一入口，接收用户输入的命令，提供认证、授权、API注册和发现等机制
> Scheduler : 负责集群资源调度，按照预定的调度策略将Pod调度到相应的node节点上
> ControllerManager : 负责维护集群的状态，比如程序部署安排、故障检测、自动扩展、滚动更新等
> Etcd ： 负责存储集群中各种资源对象的信息

node：集群的数据平面，负责为容器提供运行环境（ 干活 ）

> Kubelet : 负责维护容器的生命周期，即通过控制docker，来创建、更新、销毁容器
> KubeProxy : 负责提供集群内部的服务发现和负载均衡
> Docker : 负责节点上容器的各种操作

下面，以部署一个nginx服务来说明kubernetes系统各个组件调用关系：

1. 首先要明确，一旦kubernetes环境启动之后，master和node都会将自身的信息存储到etcd数据库中
2. 一个nginx服务的安装请求会首先被发送到master节点的apiServer组件
3. apiServer组件会调用scheduler组件来决定到底应该把这个服务安装到哪个node节点上

在此时，它会从etcd中读取各个node节点的信息，然后按照一定的算法进行选择，并将结果告知apiServer

4. apiServer调用controller-manager去调度Node节点安装nginx服务
5. kubelet接收到指令后，会通知docker，然后由docker来启动一个nginx的pod

pod是kubernetes的最小操作单元，容器必须跑在pod中至此，

6. 一个nginx服务就运行了，如果需要访问nginx，就需要通过kube-proxy来对pod产生访问的代理

这样，外界用户就可以访问集群中的nginx服务了

1.4 kubernetes概念

Master：集群控制节点，每个集群需要至少一个master节点负责集群的管控

Node：工作负载节点，由master分配容器到这些node工作节点上，然后node节点上的docker负责容器的运行

Pod：kubernetes的最小控制单元，容器都是运行在pod中的，一个pod中可以有1个或者多个容器

Controller：控制器，通过它来实现对pod的管理，比如启动pod、停止pod、伸缩pod的数量等等

Service：pod对外服务的统一入口，下面可以维护者同一类的多个pod

Label：标签，用于对pod进行分类，同一类pod会拥有相同的标签

NameSpace：命名空间，用来隔离pod的运行环境