kubernetes

作用与意义

1、Kubernetes 可以使用 DNS 名称或自己的 IP 地址公开容器，如果进入容器的流量很大，Kubernetes 可以负载均衡并分配网络流量，从而使部署稳定

2、Kubernetes 允许用户自动挂载用户选择的存储系统，例如本地存储、公共云提供商等

3、Kubernetes 描述已部署容器的所需状态，它可以以受控的速率将实际状态 更改为期望状态。例如，你可以自动化 Kubernetes 来为你的部署创建新容器， 删除现有容器并将它们的所有资源用于新容器

4、Kubernetes 允许你指定每个容器所需 CPU 和内存（RAM）。 当容器指定了资源请求时，Kubernetes 可以做出更好的决策来管理容器的资源。

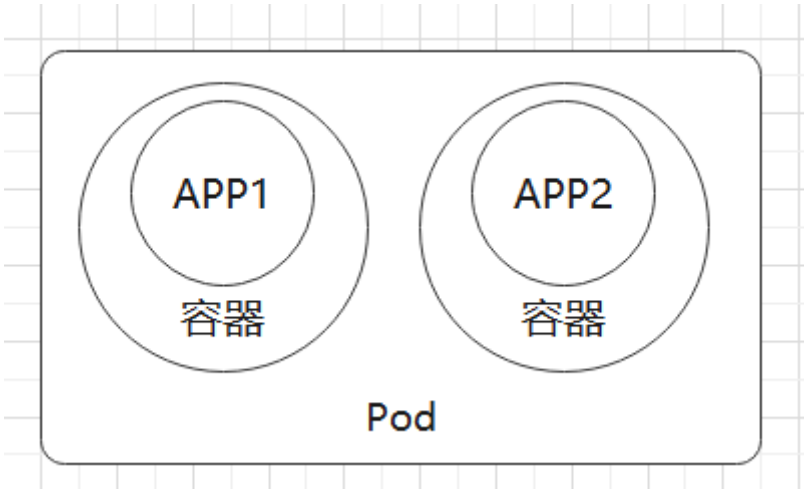
5、Kubernetes 重新启动失败的容器、替换容器、杀死不响应用户定义的 运行状况检查的容器，并且在准备好服务之前不将其通告给客户端

6、Kubernetes 允许你存储和管理敏感信息，例如密码、OAuth 令牌和 ssh 密钥。 你可以在不重建容器镜像的情况下部署和更新密钥和应用程序配置，也无需在堆栈配置中暴露密钥。

Kubernetes集群的组成

Pod

Pod 是 Kubernetes 中管理和调度的最小工作单位，Pod 中可以包含多个容器。这些容器会共享 Pod 中的网络等资源。当部署 Pod 时，会把一组关联性较强的容器部署到同一个节点上。

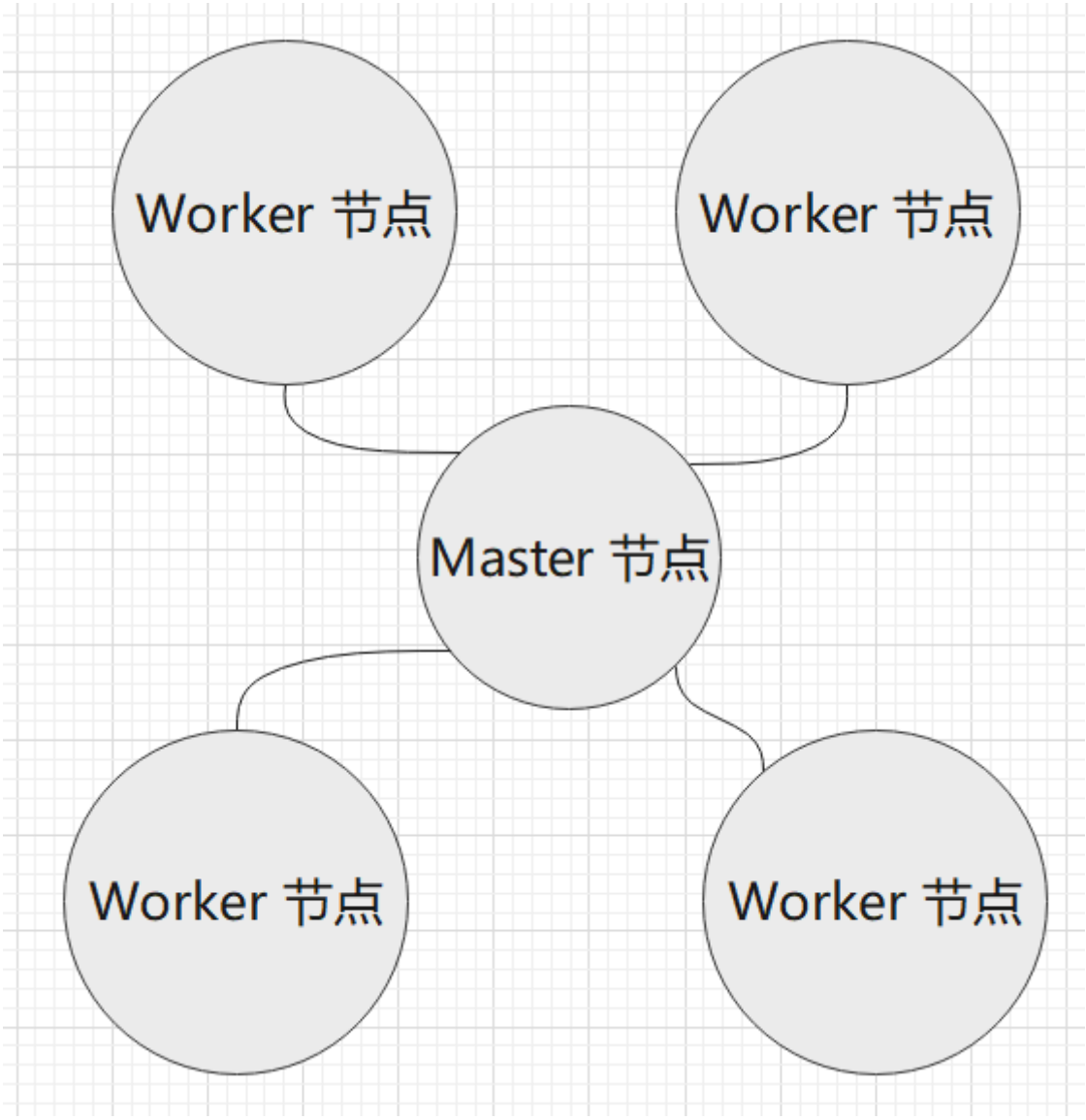


Node

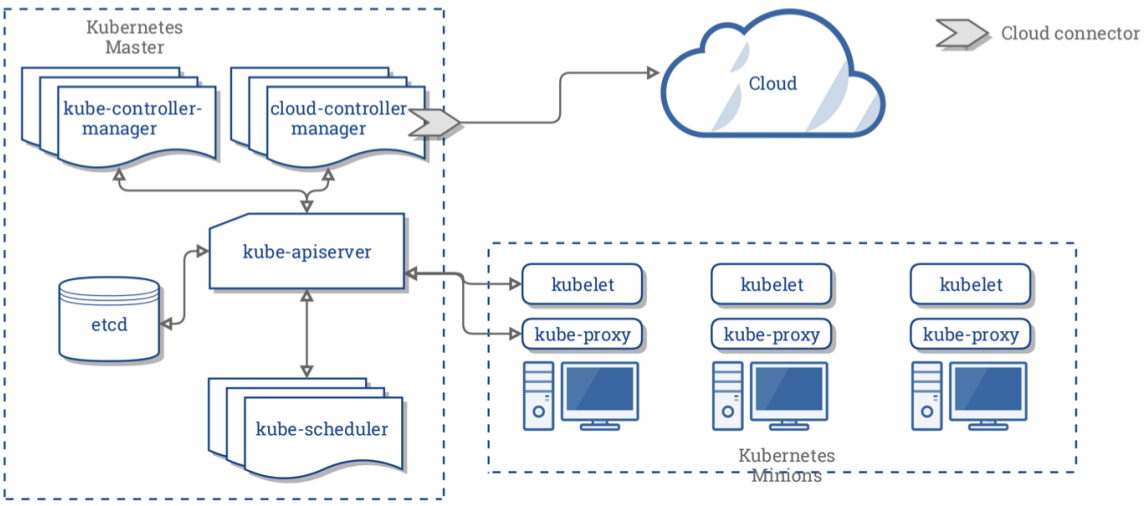
Node是指一台服务器、虚拟机等，运行着一个完整的操作系统，提供了 CPU、内存等计算资源，一个节点可以部署多个 Pod。

Cluster

一个集群(Cluster)之中，运行着 N 台服务器，即 N 个节点。这些节点有两种，一种是 master 节点，一种是 worker 节点。master 节点运行着 Kubernetes 系统组件，而 worker 节点负责运行用户的程序。所有节点都归 master 管，我们通过命令、API 的方式管理 Kubernetes 集群时，是通过发送命令或请求到 master 节点上的系统组件，然后控制整个集群。



Kubernetes结构



简单的 kubernetes 结构

左侧虚线方框中，是 master 节点，运行着各种各样的组件，master 节点负责控制整个集群，当然在很大的集群中也可以有多个 master 节点；而右侧是三个工作节点，负责运行容器应用。在图中， 主节点由多个组件构成，结构比较复杂， 主节点中记录了整个集群的工作数据，负责控制整个集群的运行。

主节点：

- 一个 API 服务(kube-apiserver)
- 一个调度器(kube-scheduler)
- 各种各样的控制器(上图有两个控制器)
- 一个存储系统(这个组件称为etcd)，存储集群的状态、容器的设置、网络配置等数据。

kube-apiserver

为了控制集群的运行，Kubernetes 官方提供了一个名为 kubectl 的二进制命令行工具，正是 apiserver 提供了接口服务，kubectl 解析用户输入的指令后，向 apiserver 发起 HTTP 请求，再将结果反馈给用户。

etcd

etcd 是兼具一致性和高可用性的键值数据库，作为保存 Kubernetes 所有集群数据的后台数据库。apiserver 的所有操作结果都会存储到 etcd 数据库中，etcd 主要存储 k8s 的状态、网络配置以及其它持久化数据，etcd 是使用 B+ 树实现的，etcd 是非常重要的组件，需要及时备份数据。

kube-scheduler

scheduler 负责监视新创建的 pod，并把 pod 分配到节点上。当要运行容器时，发送的请求会被调度器转发到 API；调度器还可以寻找一个合适的节点运行这个容器。

kube-controller-manager

kube-controller-manager 中包含了多个控制器，它们都被编译到一个二进制文件中，但是启动后会产生不同的进程。

Node节点

维护运行的 Pod 并提供 Kubernetes 运行环境。

Kubelet

一个在集群中每个节点（node）上运行的代理。 它保证容器（containers）都运行在 Pod 中。

kubelet 接收一组通过各类机制提供给它的 PodSpecs，确保这些 PodSpecs 中描述的容器处于运行状态。 kubelet 不管理不是由 Kubernetes 创建的容器。

Kube-proxy

是集群中每个节点上运行的网络代理， 实现 Kubernetes 服务（Service） 概念的一部分。

kube-proxy 维护节点上的网络规则。这些网络规则允许从集群内部或外部的网络会话与 Pod 进行网络通信。

插件说明

CoreDNS

可以为集群中的SVC创建一个域名IP的对应关系解析

Dashboard

给K8S集群提供一个B/S结构访问体系

Pod概念

运行Pod，首先启动pause容器，后续创建的容器将共用pause的网络栈（即容器不再有自己的独立IP地址），故在同一个Pod中不同容器的端口不能冲突。同时，在同一个Pod中的不同容器也共享存储卷。

自主式Pod

控制器管理Pod

Pod控制器类型：ReplicationController & ReplicaSet & Deployment(支持滚动更新和回滚)

HPA(Horizontal Pod Autoscaling)水平自动扩展

StatefulSet:解决有状态服务的问题

- 稳定的持久化存储
- 稳定的网络标志
- 有序部署
- 有序回收

DaemonSet:确保全部Node上运行一个Pod副本，当有Node加入集群时，会新建一个Pod。当有Node从集群移除时候，这些Pod会被回收。

网络通讯方式

Kubernetes的网络模型假定所有Pod都在一个可以直接连通的扁平网络空间中，在私有云里搭建Kubernetes集群，就不能假定这个网络已经存在。为了实现网络假设，需要将不同节点上的Docker容器之间的互相访问先打通，然后运行Kubernetes

同一个Pod内的多个容器之间共用网络栈，自然连通

各pod之间的通讯：Oerlay Network

Pod与Service之间的通讯：各节点的Iptables规则

Flannel的功能是让集群中的不同节点主机创建的Docker容器都具有全集群唯一的虚拟IP地址，而且在这些IP地址之间建立一个覆盖网络，通过这个覆盖网络，将数据包原封不动地传递到目标容器内。

ETCD对于Flannel的作用

- 存储管理Flannel可分配的IP地址段资源
- 监控ETCD中每个Pod的实际地址，并在内存中建立维护Pod节点路由表