1. K8S权威指北学习(2019-06)
2. Kubernetes概述
   * 1. 起源、特点和发展历程
3. Kubernetes起源于古希腊, 为舵手之意, 由谷歌开发, 用来管理Docker容器。Docker将自己定义为装着集装箱的鲸鱼, 因此, 谷歌以Kubernetes掌握大航海时代的话语权。
4. Kubernetes第一个正式版本 -- Kubbernetes 1.0于2015年7月发布, 如今, 已经拥有了巨大的影响力, 吸引了包括IBM、惠普、微软、红帽、Intel、VMWare、CoreOS、Docker、Mesosphere、Mirantis等在内的众多业界巨头纷纷加入。红帽, 这个虚拟化领域的领导者之一, 在容器上已经完全跟从谷歌, 将自家第三代OpenShift产品的架构底层换成了Docker+Kubernetes, 还直接在新一代容器操作系统Atomic内原生集成Kubernetes。
5. Kubernetes是第一个将"一切以服务为中心,一切围绕服务运转"作为指导思想的创新性产品, 它的功能和架构设计自始至终都遵循这一指导思想, 构建在Kubernetes上的系统不仅可以独立运行在物理机、虚拟机集群或者企业私有云上, 也可以被托管在公有云中。
6. Kubernetes方案的另一个了亮点是自动化, 在Kubernetes的解决方案中, 一个服务可以自我扩展、自我诊断, 并且容易升级, 在收到服务扩容的请求后, Kubernetes会触发调度流程, 最终在选定的目标节点上启动相应数量的服务实例副本, 这些副本在启动成功后会自动加入负载均衡器中病生效, 整个过程无需额外的人工操作。另外, Kubernetes会定时巡查每一个服务的所有实例的可用性, 确保服务实例的数量始终保持为预期的数量, 当它发现某个实例不可用时,会自动重启实例或从其他节点重新调度并运行一个新实例, 而这个过程中不需要人工干预。
   * 1. K8S与Docker
7. 通常我们把Kubernetes看做Docker的上层架构, 就好像Java与J2EE一样: J2EE是以Java为基础的企业级软件架构, 而Kubernetes则以Docker为基础打造了一个云计算时代的全新分布式系统架构。 但在Kubernetes的架构里, Docker只是其目前支持的两种底层容器技术之一, 另一个容器技术是Rocket, 后者来源于CoreOS这个Docker昔日的"恋人"所推出的竞争产品。
8. Kubernetes同时支持这两种互相竞争的容器技术是有深刻的历史原因的。快速发展的Docker打败了谷歌曾经名噪一时的开源容器技术Imctfy, 并迅速风靡世界。但是, 作为一个已经对全球IT公司产生重要影响的技术, Docker背后的容器标准的制定注定不可能被任何一个公司私有控制, 于是就有了后来引发危机的CoreOS与Docker分手事件: CoreOS撇开Docker推出了与其相抗的开源容器项目 -- Rocket, 并动员一些知名IT公司成立委员会来试图主导容器技术的标准化, 该分手事件愈演愈烈, 最终导致CoreOS傍上谷歌背离Docker阵营, 共同发起了基于CoreOS+Rocket+Kubernetes的新项目Teconic。最后, 由Linux基金会调节, 谷歌, CoreOS与Docker一起加入OCP项目。Docker被迫放弃了自己的独家控制权, 作为回报, Docker容器格式被OCP采纳为新标准的基础, 且由Docker负责起草OCP草案规范的初稿文档 -- 代价是Docker提交自己的容器执行引擎的源码作为OCP的启动资源。
9. 谷歌不仅为当年失败的Imctfy报了一箭之仇, 还重新掌控了容器技术的未来。之后,谷歌加入Openstack阵营, 确保Linux容器及关联的容器管理技术Kubernetes被Openstack生态圈所容纳, 且成为Openstack平台上与KVM一样的一等公民。以容器为代表的应用形态与以虚拟化为代表的系统形态将会完美融合与Openstack智商,并与软件定义网络和软件定义存储一起统治下一代数据中心。
   1. Kubernetes入门
      1. Kubernetes是什么
10. 它是一个全新的基于容器技术的分布式架构领先方案, 它的前身是Borg
11. 如果我们的系统设计遵循Kubernetes的设计思想, 那么传统系统架构中和业务无关的底层代码和功能模块可以立刻从我们视线中消失, 不必再费心于负载均衡器的选型和部署实施问题, 不必再考虑引入或开发一个复杂的服务治理框架, 不用再关心服务监控和故障处理模块的开发。我们可以节省不少于30%的开发成本, 同时可以将精力更加集中于业务本身, 而且由于Kubernetes提供强大的自动化机制, 后期的运维难度和运维成本会大幅降低。
12. Kubernetes是一个开放的开发平台。与J2EE不同, 它不局限于任何一种语言, 没有限定编程接口, 因此适配所有语言编写的服务, 并且Kubernetes对现有的编程语言、编程框架没有任何侵入性, 因此现有的系统很容易改造升级到K8S平台上
13. 最后, K8S是一个完备的分布式系统支撑平台。K8S具有完备的集群管理能力, 包括多层次的安全防护和准入机制, 及多租户应用支撑能力、透明的服务注册和服务发现机制、內建智能负载均衡器、强大的故障发现和自我修复能力、服务滚动升级和在线扩容能力、可扩展的资源自动调度机制, 以及多粒度的资源配额顾艳丽能力。同时, Kubernetes提供了完善的管理工具, 这些工具涵盖了包括开发、部署测试、运维监控在内的各个环节。因此, K8S是一个全新的基于容器技术的分布式架构解决方案, 并且是一个一站式的完备的分布式系统开发和支撑平台。
    * 1. Kubernetes基本知识
14. 在K8S中,服务是分布式集群架构的核心, 一个服务对象拥有如下关键特征
15. 拥有一个唯一指定的名字(比如mysql-server)
16. 拥有一个虚拟IP(Cluster IP、Service IP或VIP)和端口号
17. 能够提供某种远程服务能力
18. 被映射到了提供这种服务能力的一组容器应用上
19. 服务进程目前都基于Socket通信方式对外提供服务, 比如Redis、Memcache、MySQL、Web Server, 或者是实现了某个具体业务的一个特定的TCP Server进程。虽然一个Service通常由多个相关的服务进程来提供服务, 每个服务进程都有一个独立的Endpoint(IP+Port)访问点, 但Kubernetes能够让我们通过Service(虚拟集群IP+服务端口)连接到制定的服务上。有了Kubernetes內建的同名负载均衡和故障恢复机制, 不管后端有多少服务进程, 也不管某个服务进程是否会由于发生故障重新部署到其他机器, 都不会影响到我们对服务的正常调用。更重要的是这个Service本身一旦创建就不再变化, 这意味着在K8S集群中, 我们再也不用为了服务的IP改变而头痛。
20. K8S设计了Pod对象, 将每个服务进程包装到相应的Pod中, 使其成为Pod中运行的一个容器。为了建立服务和Pod之间的关联关系, Kubernetes首先给每个Pod贴上一个标签, 给运行MySQL的Pod贴上name=mysql标签, 给运行PHP的Pod贴上name=php的标签, 然后给相应的服务定义标签选择器(Label Selector), 比如MySQL的标签选择器的选择条件为anme=mysql
21. Pod运行在一个我们称之为节点(Node)的环境中, 这个节点既可以是物理机, 也可以是公有云或者私有云中的一个虚拟机, 通常在一个节点山公允性几百个Pod, 其次, 每个Pod里运行这一个特殊的被称之为Pause的容器, 其他容器则为业务容器, 这些业务容器共享Pause容器的网络栈
    1. 为什么要用Kubernetes
    2. 从一个简单的例子开始
    3. Kubernetes基本概念和术语