部署相关 CRD

请确认你已经完成了准备工作中的内容,这里我们来看一下如何搭建一个非高可用的 Kube-OVN。

部署相关 CRD

Kube-OVN 创建了 Subnet 和 IP 两种 CRD 资源方便网络的管理

kubectl apply -f https://raw.githubusercontent.com/alauda/kube-ovn/v0.10.2/yamls/crd.yaml

查看 CRD

[root@node2 yamls]# kubectl get crd

NAME CREATED AT

ips.kubeovn.io 2019-08-06T09:08:44Z

subnets.kubeovn.io 2019-08-06T09:08:44Z

部署 OVN

Kube-OVN 底层依赖 Open vSwitch 社区提供的 OVS 和 OVN，这里我们通过 yaml 来进行部署。

给部署 ovndb 的机器打标签

ovndb 需要将数据存在宿主机硬盘来持久化状态，选择一个节点增加标签

kubectl label node <Node on which to deploy OVN DB> kube-ovn/role=master

查看标签状态

[root@node2 yamls]# kubectl get no --show-labels

NAME STATUS ROLES AGE VERSION LABELS

node1 Ready master 7d21h v1.15.1 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kube-ovn/role=master,kubernetes.io/arch=amd64,kubernetes.io/hostname=node1,kubernetes.io/os=linux,node-role.kubernetes.io/master=

node2 Ready master 7d21h v1.15.1 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=node2,kubernetes.io/os=linux,node-role.kubernetes.io/master=

node3 Ready <none> 7d21h v1.15.1 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=node3,kubernetes.io/os=linux

部署 OVN/OVS

部署 yaml

kubectl apply -f https://raw.githubusercontent.com/alauda/kube-ovn/v0.10.2/yamls/ovn.yaml

查看 pod 状态

[root@node2 yamls]# kubectl get pod -n kube-ovn

NAME READY STATUS RESTARTS AGE

ovn-central-6d5d566cc8-vktvm 1/1 Running 0 32s

ovs-ovn-47wxk 1/1 Running 0 32s

ovs-ovn-l8t6g 1/1 Running 0 32s

ovs-ovn-v225f 1/1 Running 0 32s

安装 Kube-OVN Controller 及 CNIServer

Kube-OVN Controller 和 CNIServer 中有大量可配参数，这里为了快速上手，我们不做更改。默认配置下 Kube-OVN 会使用 10.16.0.0/16 作为默认子网，100.64.0.1/16 作为主机和 Pod 通信子网，使用 Kubernetes 中的 Node 主网卡作为 Pod 流量通信使用网卡，并开启流量镜像功能。

部署 yaml

kubectl apply -f https://raw.githubusercontent.com/alauda/kube-ovn/v0.10.2/yamls/kube-ovn.yaml

观察部署状态：

[root@node2 yamls]# kubectl get pod -n kube-ovn

NAME READY STATUS RESTARTS AGE

kube-ovn-cni-29784 1/1 Running 0 17s

kube-ovn-cni-5572f 1/1 Running 0 17s

kube-ovn-cni-hdffl 1/1 Running 0 17s

kube-ovn-controller-65b5cb8574-262vg 1/1 Running 0 17s

kube-ovn-controller-65b5cb8574-ss94l 1/1 Running 0 17s

ovn-central-6d5d566cc8-vktvm 1/1 Running 0 10m

ovs-ovn-47wxk 1/1 Running 0 10m

ovs-ovn-l8t6g 1/1 Running 0 10m

ovs-ovn-v225f 1/1 Running 0 10m

观察自动创建的 Subnet:

[root@node2 yamls]# kubectl get subnet

NAME PROTOCOL CIDR PRIVATE NAT

join IPv4 100.64.0.0/16 false false

ovn-default IPv4 10.16.0.0/16 false true

安装 Kubectl 插件

为了更好的对网络质量进行监控以及对故障进行排查，Kube-OVN 提供了 kubectl 的插件，推荐安装。

下载 kubectl-ko 文件

wget https://raw.githubusercontent.com/alauda/kube-ovn/v0.10.2/dist/images/kubectl-ko

将文件复制到 $PATH 下的某个目录

mv kubectl-ko /usr/local/bin/kubectl-ko

给 kubectl-ko增加可执行权限

chmod +x /usr/local/bin/kubectl-ko

检查插件状态

[root@kube-ovn01 ~]# kubectl plugin list

The following compatible plugins are available:

/usr/local/bin/kubectl-ko

对网络质量进行检查

[root@kube-ovn01 ~]# kubectl ko diagnose all

好了！你已经完成了 Kube-OVN 的安装，让我们运行一个 Pod 来观察一下网络。

kubectl run nginx --image=nginx

观察容器地址

[root@node2 yamls]# kubectl get pod -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

nginx-74d5899f46-x5ghb 1/1 Running 0 2m58s 10.16.0.2 node1 <none> <none>

进入容器进行一些简单网络操作

[root@node2 yamls]# kubectl exec -it nginx-74d5899f46-x5ghb sh

/ # ping -c 1 10.16.0.1

PING 10.16.0.1 (10.16.0.1): 56 data bytes

64 bytes from 10.16.0.1: seq=0 ttl=254 time=0.379 ms

--- 10.16.0.1 ping statistics ---

1 packets transmitted, 1 packets received, 0% packet loss

round-trip min/avg/max = 0.379/0.379/0.379 ms

/ # ping -c 1 100.64.0.1

PING 100.64.0.1 (100.64.0.1): 56 data bytes

64 bytes from 100.64.0.1: seq=0 ttl=254 time=0.447 ms

--- 100.64.0.1 ping statistics ---

1 packets transmitted, 1 packets received, 0% packet loss

round-trip min/avg/max = 0.447/0.447/0.447 ms

/ # ping -c 1 114.114.114.114

PING 114.114.114.114 (114.114.114.114): 56 data bytes

64 bytes from 114.114.114.114: seq=0 ttl=87 time=40.669 ms

--- 114.114.114.114 ping statistics ---

1 packets transmitted, 1 packets received, 0% packet loss

round-trip min/avg/max = 40.669/40.669/40.669 ms