版本验证1.16.7

samctfu 20210612

一、检查环境，软件

检查是否以root用户登录ubuntu

检查hostname是否小写（hostname）

检查subctl 是否需要重新安装（subctl version）

检查docker是否安装（docker version）

检查calicoctl是否安装（calicoctl version）

检查yq是否安装（yq --help）

检查conntrack是否安装（conntrack verison）

检查dos2unix是否安装

检查ssh能否链接其他主机

给三台云服务器命名

version-verify-cluster-a

version-verify-cluster-b

version-verify-cluster-c

二、更新subctl

# 先手动把自己编译的subctl上传到/root/ 目录下，然后复制到指定的文件夹，把原来的subctl覆盖掉

mv subctl /root/.local/bin/

# 给subctl文件赋予执行权限

chmod +x /root/.local/bin/subctl

# 先手动把自己编译的subctl上传到/root/ 目录下，然后复制到指定的文件夹

mv subctl-release-0.9-198fe39-linux-amd64 /root/.local/bin/

# 给subctl-release-0.9-198fe39-linux-amd64文件赋予执行权限

chmod +x /root/.local/bin/subctl-release-0.9-198fe39-linux-amd64

# 进入指定的文件夹

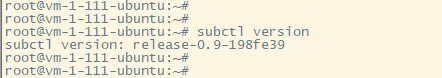
cd /root/.local/bin/

# 把原来的subctl覆盖掉（可以用mv命令或者rename命令）

mv subctl-release-0.9-198fe39-linux-amd64 subctl

# 检查是否覆盖成功

subctl version



三、创建EndpointSlices.yaml

# 创建一个EndpointSlices.yaml文件，文件的内容也接着给出了，粘贴一下啦

vi /root/EndpointSlices.yaml

apiVersion: apiextensions.k8s.io/v1beta1

kind: CustomResourceDefinition

metadata:

name: endpointslices.discovery.k8s.io

spec:

group: discovery.k8s.io

version: v1beta1

scope: Namespaced

names:

kind: EndpointSlice

plural: endpointslices

singular: endpointslice

（下面这行语句写在shell脚本里）

# 在指定路径文件下创建resource，kubectl命令用于根据文件或输入创建集群resource。如果已经定义了相应resource的yaml或json文件，直接kubectl create -f filename即可创建文件内定义的resource

kubectl create -f /root/EndpointSlices.yaml

五、执行脚本前的检查

检查cluster-a

检查cluster-1.txt文件是否存在，检查EndpointSlices.yaml文件是否存在，检查脚本的内容是不是1.16.7

如果已经安装了calicoctl，检查calicoctl的位置 /usr/local/bin/ 还是 /usr/bin/ 是否与shell脚本相匹配

检查cluster-b

检查cluster-2.txt文件是否存在，检查EndpointSlices.yaml文件是否存在，检查broker-info.subm是否存在，检查脚本的内容是不是1.16.7

如果已经安装了calicoctl，检查calicoctl的位置 /usr/local/bin/ 还是 /usr/bin/ 是否与shell脚本相匹配

然后把这个脚本命名为cluster-a.sh 上传到云服务器，放在/home/myscript目录下

# 用dos2unix转换脚本

dos2unix /home/myscript/cluster-a.sh

# 给脚本赋予执行权限

chmod +x /home/myscript/cluster-a.sh

# 执行脚本

/home/myscript/cluster-a.sh

六、cluster-b安装nginx

# 先创建一个deployment对象，对象的名字叫做nginx，然后这个deployment对象运行nginx镜像

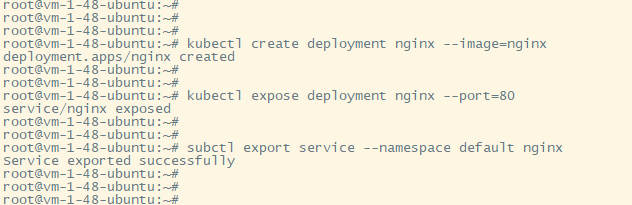
kubectl create deployment nginx --image=nginx

# 为deployment对象的nginx创建service，并通过Service的80端口转发至容器的80端口上。

kubectl expose deployment nginx --port=80

#使用subctl创建ServiceExport对象，使service可见，使得submariner中的其他集群可以发现这个service（执行这条指令可能会卡住大概10秒）

subctl export service --namespace default nginx



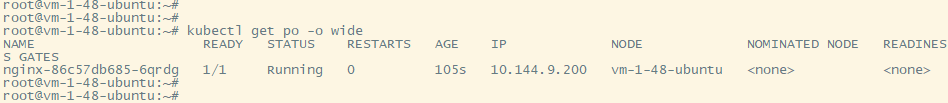
七、kubectl指令

# 获取当前运行的所有node节点的状态，一共三个字段name，status（ready notready），age（存活时间）

kubectl get nodes

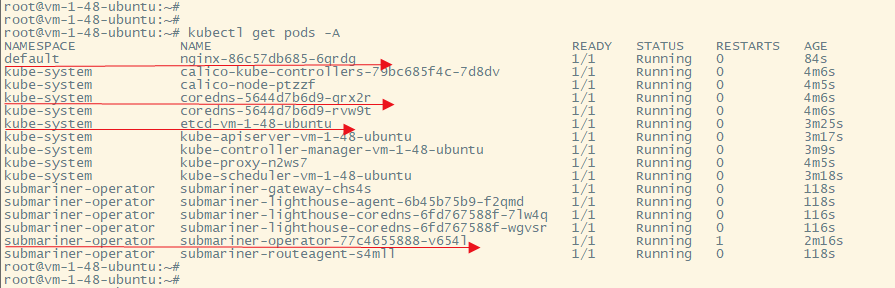
# 获取当前运行的namespace所有pods的信息（比如获取nginx的IP），包括pod运行在哪个节点上

kubectl get po -o wide



# 查看所有namespace的pod

kubectl get pods -A



这里有一些值得关注的地方，比如namespace有default kube-system 和submariner-operator三个。在default里面只有一个nginx的pod在工作；在kube-system里面有几个比较熟悉的pod，比如coredns，etcd；

submariner-operator是我们自己编译出来的subctl-release-0.9-198fe39-linux-amd64所部署的

# 如果某个pods部署失败，查看日志

kubectl logs -f submariner-gateway-p6kbb -n submariner-operator

# 查看日志并且抓取error

kubectl logs -f submariner-operator-5cff6f87b4-mwvfx -n submariner-operator | grep error

#####################################################################################

namespace是对一组resource和对象的抽象集合，比如将系统内部的对象划分为不同的项目组或用户组。常见的pods, services, replication controllers和deployments等都是属于某一个namespace的（默认是default），而node, persistentVolumes等则不属于任何namespace。

一个kubernetes集群中可以拥有多个namespace，它们在逻辑上彼此隔离，所以namespace常用来隔离不同的用户。 namespace提供组织，安全和性能方面的提升（有的地方说namespace会影响性能，看你怎么理解了，比如我的namespace划分得很合理，然后我的工作进行得很顺利，是不是就相当于提升了kubernetes的性能）。大多数的kubernetes中有是3个namespace：

（1）default：你的service和app默认被创建在这里

（2）kube-system：kubernetes系统组件使用，kubernetes自带的服务一般运行在这里

（3）kube-public：公共资源使用，实际上现在并不常用

kubernetes中的namespace与docker中的namespace不同，kubernetes中的namespace只是做了一个逻辑上的隔离。

# 创建一个名叫test的namespace

kubectl create namespace test

# 用yaml文件创建namespace （又是yaml文件，刚才在EndpointSlices.yaml见过了）

kubectl apply -f test.yaml

kind: Namespace

apiVersion: v1

metadata:

name: test

labels:

name: test

kubernetes一开始默认的激活的命名空间是default。因此，当你在其他namespace创建了资源，那么每次使用kubectl命令都要带上namespace。幸好，kubens 能够解决这个问题。

# 执行kubens命令，它会高亮当前的namespace

kubens

# 如果要更换到名字为test的namespace

kubens test

现在，你的所有的命令会在这个叫做test的namespace下执行

#####################################################################################

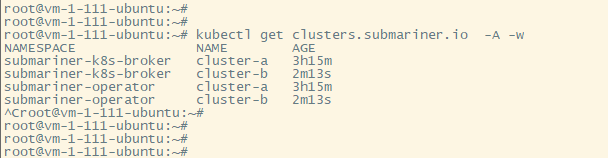
kubectl label node vm-1-48-ubuntu submariner.io/gateway=true

为kubernetes集群的resource打标签，通过打标签可以对resource分组。还可以对nodes打标签，这样在编排容器时，可以为容器指定nodeSelector将容器调度到指定lable的机器上，比如对集群中的IO密集型，计算密集型的机器分组，可以将不同的机器打上不同标签，然后将不同特征的容器调度到不同分组上。

# submariner指令

# 获取当前运行的所有cluster的信息（cluster-a上执行该指令），看看其他的cluster有没有加入进来

kubectl get clusters.submariner.io -A -w



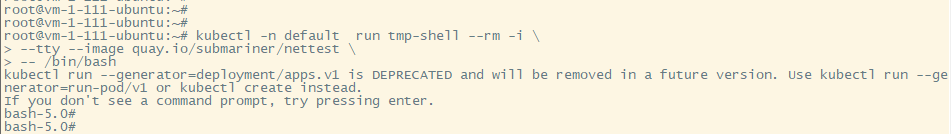
# 从Kubernetes集群中的命令行运行容器，分两种情况

# （1）如果你的cluster-a没安装tmp-shell（执行这条指令可能会卡住大概10秒）

kubectl -n default run tmp-shell --rm -i \

--tty --image quay.io/submariner/nettest \

-- /bin/bash



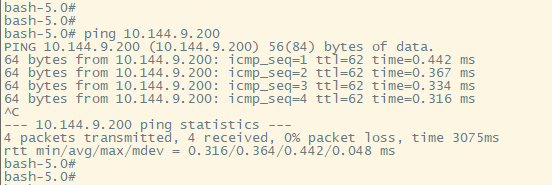
# （2）如果你的cluster-a安装了tmp-shell

kubectl exec -it tmp-shell sh

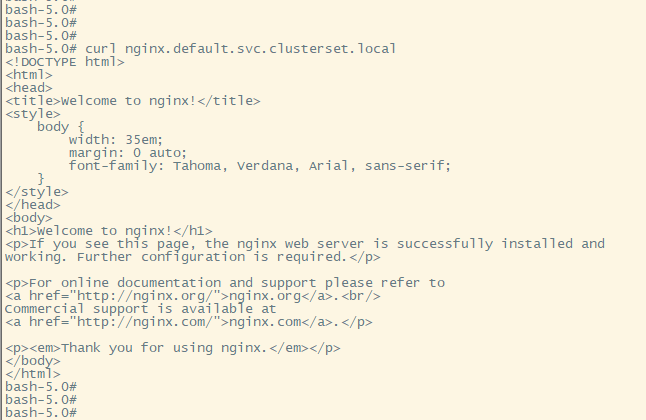
# 在cluster-a上检查nginx能否连接上curl 10.144.9.200

curl <cluster-b显示的nginx的IP>

ping <cluster-b显示的nginx的IP>



curl nginx.default.svc.clusterset.local



1.16.7脚本cluster-a5.sh命令解析

# kubernetes初始化

kubeadm init \

--apiserver-advertise-address=10.1.1.111 \

--apiserver-cert-extra-sans=localhost,127.0.0.1,10.1.1.111,43.128.252.173 \

--pod-network-cidr=10.44.0.0/16 --service-cidr=10.45.0.0/16 \

--kubernetes-version v1.16.7 \

--ignore-preflight-errors=FileExisting-conntrack

# yq的官方网站（<https://mikefarah.gitbook.io/yq/operators>）；

# -i表示设置yaml文件的缩进；.clusters[].cluster.server |= sub("10.1.1.111", "43.128.252.173") 表示把clusters下面的cluster的server指定字段替换，“|=”是指定字段替换符

yq -i eval \

'.clusters[].cluster.server |= sub("10.1.1.111", "43.128.252.173") | .contexts[].name = "cluster-a" | .current-context = "cluster-a"' \

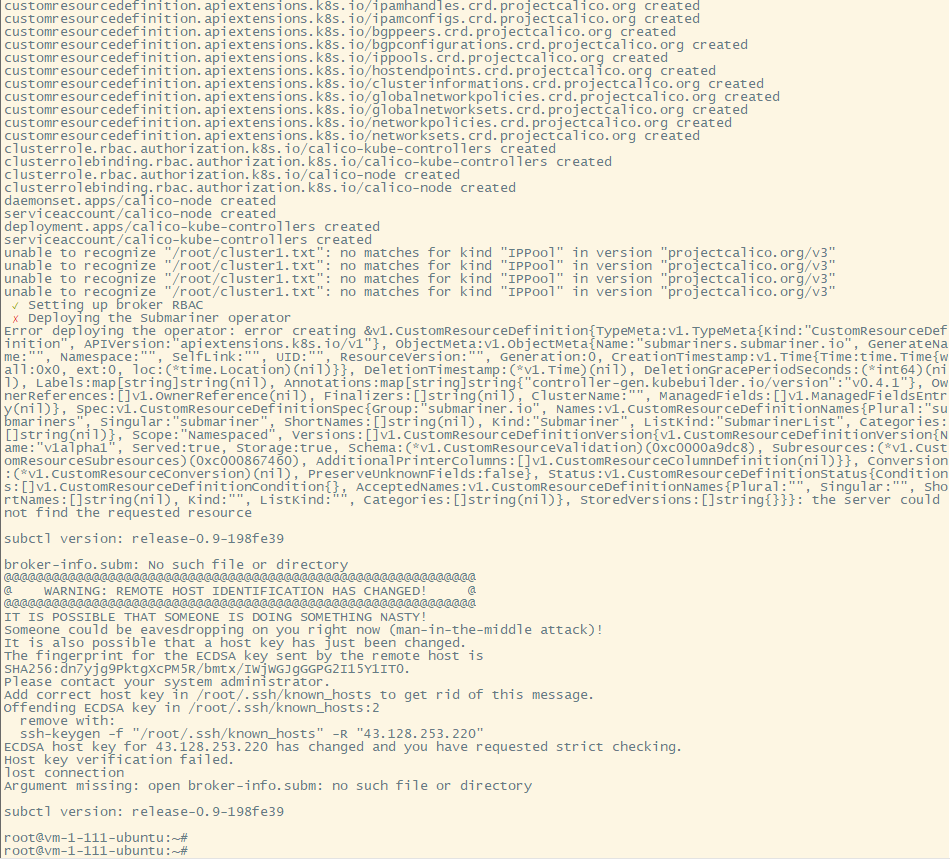
$HOME/.kube/config

版本验证1.15.7

更新cluster-a.txt

curl -o /usr/local/bin/calicoctl -O -L https://github.com/projectcalico/calicoctl/releases/download/v3.8.1/calicoctl && chmod +x /usr/local/bin/calicoctl

|  |  |  |  |
| --- | --- | --- | --- |
| docker, yq, | | | |
| kubeadm 1.15.7 | subctl release-0.9-198fe39 | calicoctl 3.18.4 | ✗ Deploying the Submariner operator |
| kubeadm 1.15.7 | subctl release-0.9-198fe39 | calicoctl 3.8.1 | ✗ Deploying the Submariner operator |
| kubeadm 1.15.7 | subctl release-0.9-198fe39 | calicoctl 3.8.1  cluster-a9.sh | ✗ Deploying the Submariner operator |
| kubeadm 1.15.7 | subctl v0.7.0-dev | calicoctl 3.8.1 | deploy succeed but verify failed |
|  |  |  |  |



✓ Setting up broker RBAC

✗ Deploying the Submariner operator

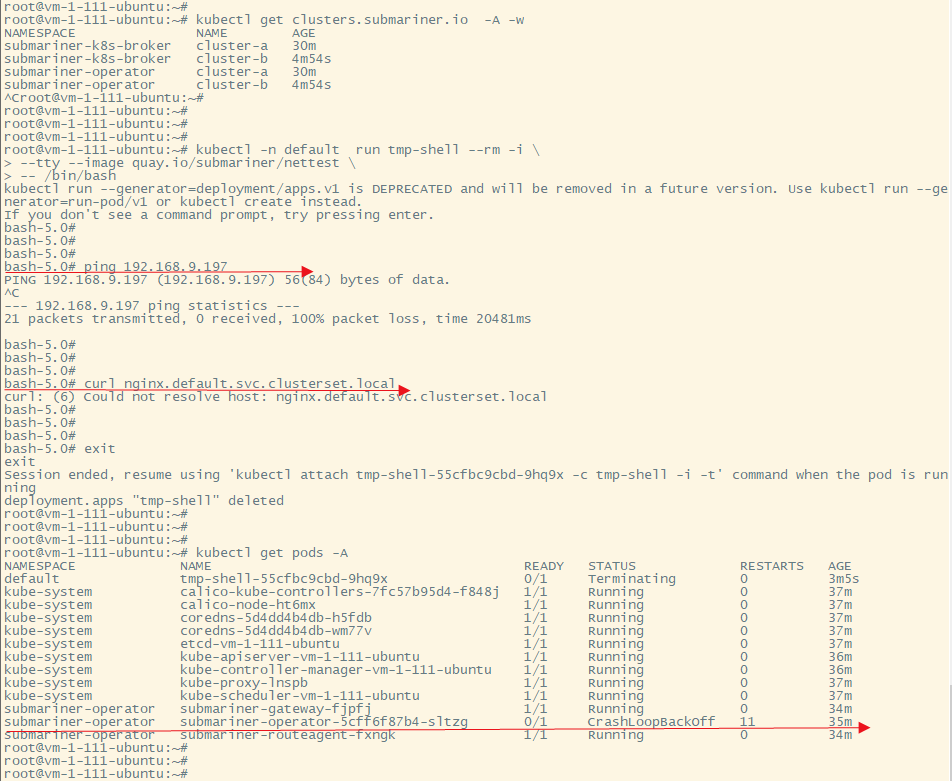
Error deploying the operator: error creating &v1.CustomResourceDefinition{TypeMeta:v1.TypeMeta{Kind:"CustomResourceDefinition", APIVersion:"apiextensions.k8s.io/v1"}, ObjectMeta:v1.ObjectMeta{Name:"submariners.submariner.io", GenerateName:"", Namespace:"", SelfLink:"", UID:"", ResourceVersion:"", Generation:0, CreationTimestamp:v1.Time{Time:time.Time{wall:0x0, ext:0, loc:(\*time.Location)(nil)}}, DeletionTimestamp:(\*v1.Time)(nil), DeletionGracePeriodSeconds:(\*int64)(nil), Labels:map[string]string(nil), Annotations:map[string]string{"controller-gen.kubebuilder.io/version":"v0.4.1"}, OwnerReferences:[]v1.OwnerReference(nil), Finalizers:[]string(nil), ClusterName:"", ManagedFields:[]v1.ManagedFieldsEntry(nil)}, Spec:v1.CustomResourceDefinitionSpec{Group:"submariner.io", Names:v1.CustomResourceDefinitionNames{Plural:"submariners", Singular:"submariner", ShortNames:[]string(nil), Kind:"Submariner", ListKind:"SubmarinerList", Categories:[]string(nil)}, Scope:"Namespaced", Versions:[]v1.CustomResourceDefinitionVersion{v1.CustomResourceDefinitionVersion{Name:"v1alpha1", Served:true, Storage:true, Schema:(\*v1.CustomResourceValidation)(0xc0000a9dc8), Subresources:(\*v1.CustomResourceSubresources)(0xc000867460), AdditionalPrinterColumns:[]v1.CustomResourceColumnDefinition(nil)}}, Conversion:(\*v1.CustomResourceConversion)(nil), PreserveUnknownFields:false}, Status:v1.CustomResourceDefinitionStatus{Conditions:[]v1.CustomResourceDefinitionCondition{}, AcceptedNames:v1.CustomResourceDefinitionNames{Plural:"", Singular:"", ShortNames:[]string(nil), Kind:"", ListKind:"", Categories:[]string(nil)}, StoredVersions:[]string{}}}: the server could not find the requested resource

subctl version: release-0.9-198fe39

broker-info.subm: No such file or directory

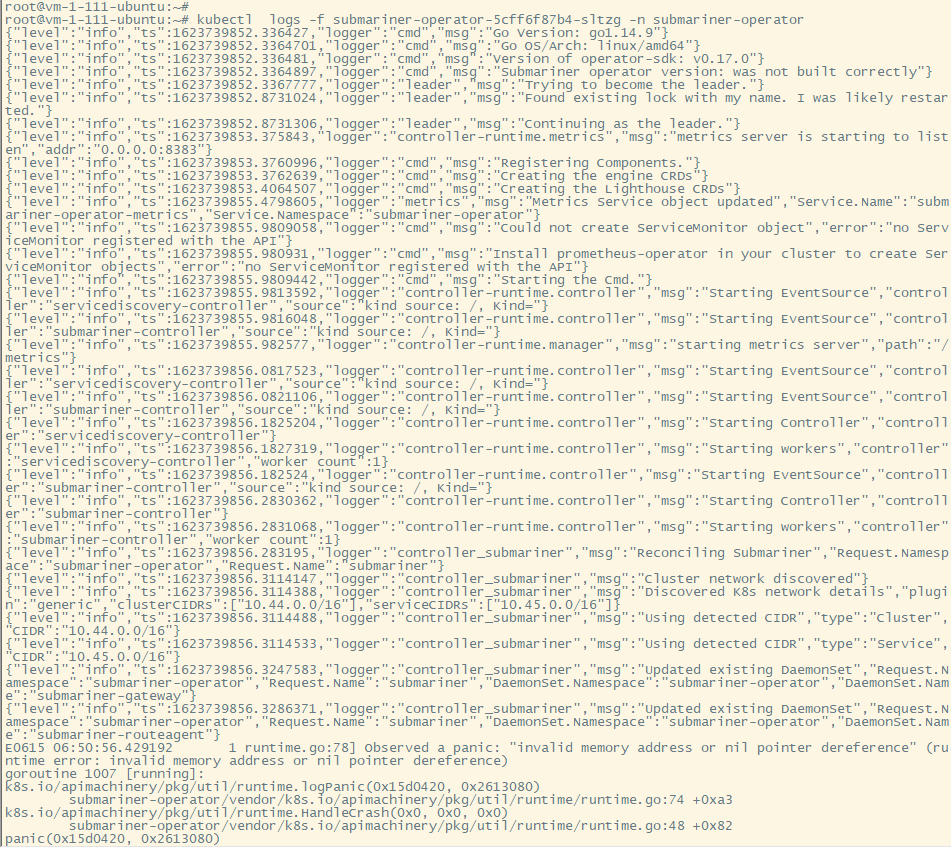
|  |  |  |  |
| --- | --- | --- | --- |
| kubeadm 1.15.7 | subctl v0.7.0-dev | calicoctl 3.8.1 | deploy succeed but verify failed |

cluster-a



# 查找日志记录

kubectl logs -f submariner-operator-5cff6f87b4-mwvfx -n submariner-operator | grep error

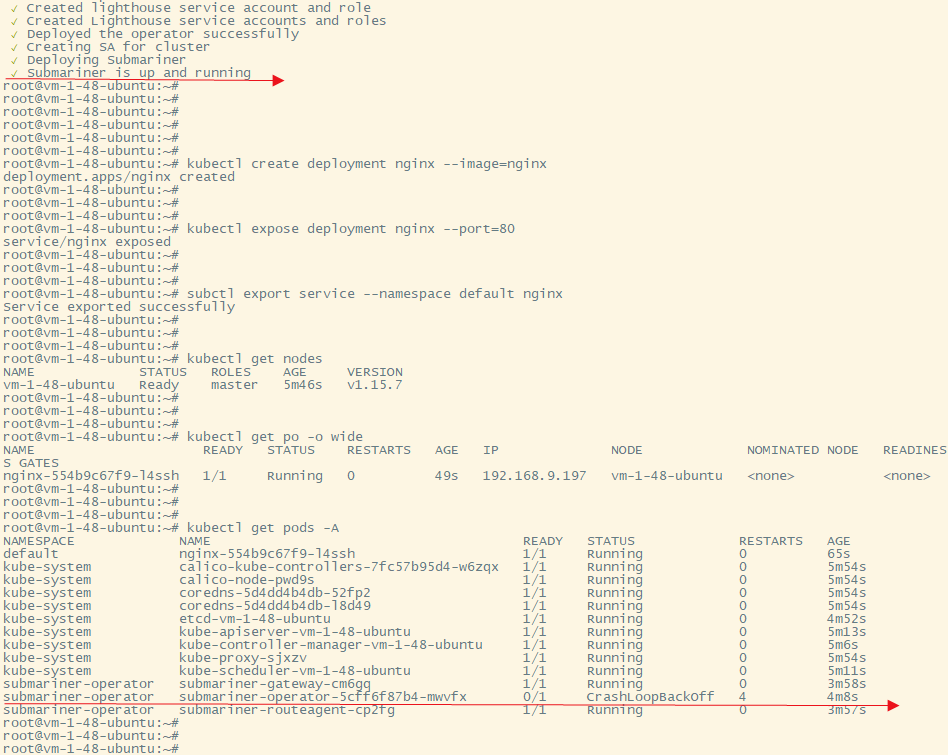


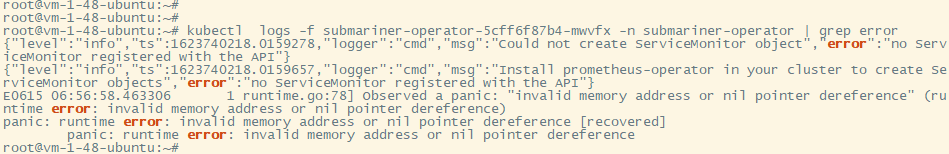
cluster-b

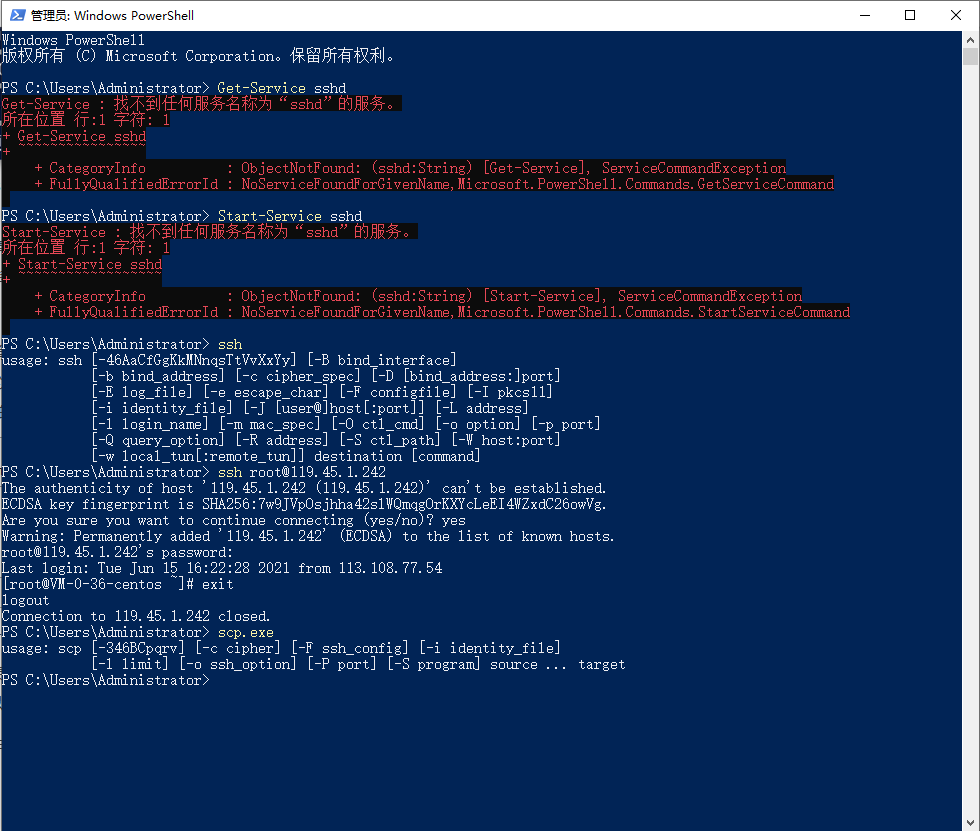
kubectl get pods -A

NAMESPACE NAME READY STATUS RESTARTS AGE

submariner-operator submariner-operator-5cff6f87b4-mwvfx 0/1 CrashLoopBackOff 4 4m8s







版本验证1.19.1的ipvs

#####################################################################################

从k8s的1.8版本开始，kube-proxy引入了ipvs（ip virtual service）模式，ipvs模式与iptables同样基于Netfilter（linux的内核之一），考察底层的数据结构，ipvs采用的hash表，iptables采用一条条的规则列表。ipvs处理连接请求的时间复杂度是O(1)，而iptables是O(n)。

所以iptables对于大型集群的性能不如ipvs。在5000个node的集群中使用 NodePort 服务，如果一共有2000个service并且每个service有10个 pod，这将在每个node上至少产生20000个 iptable 记录，这会造成内核非常繁忙。

#####################################################################################

# 检查ssh是否能连接，可能会出现下面这种错误

remove with: ssh-keygen -f "/root/.ssh/known\_hosts" -R

原因分析，本来cluster-a通过ssh能够连上cluster-b和cluster-c，但是后来cluster-c重装了系统，cluster-a通过ssh能够连上cluster-b，但是不能再连上cluster-c，并且报出上述错误

# 解决方法，在cluster-a执行

sudo ssh-keygen -R <cluster-c的内网ip>

一、检查环境，软件

二、subctl不用dev版本，按照官网的指引下载安装即可

三、不需要yaml文件

# 把脚本中这行语句去掉

kubectl create -f /root/EndpointSlices.yaml

四、编写脚本，安装ipvs需要的软件

# 下载ipvs需要的软件，写成一个脚本，放在/home/myscript，然后执行这个脚本

cat > /home/myscript/ipvs.modules <<EOF

#!/bin/bash

modprobe -- ip\_vs

modprobe -- ip\_vs\_rr

modprobe -- ip\_vs\_wrr

modprobe -- ip\_vs\_sh

modprobe -- nf\_conntrack\_ipv4

EOF

# 给脚本赋予执行权限，然后执行该脚本

chmod +x /home/myscript/ipvs.modules && \

bash /home/myscript/ipvs.modules && \

lsmod | grep -e ip\_vs -e nf\_conntrack\_ipv4

# 下载ipset ipvsadm

apt-get install ipset ipvsadm -y

五、部署cluster-a和cluster-b前的检查

检查cluster-a

检查cluster-1.txt文件是否存在，

检查脚本的内容是不是1.19.1，

检查myinit-a.yaml是否存在

如果已经安装了calicoctl，检查calicoctl的位置 /usr/local/bin/ 还是 /usr/bin/ 是否与shell脚本相匹配

检查cluster-b

检查cluster-2.txt文件是否存在，

检查broker-info.subm是否存在，

检查脚本的内容是不是1.19.1，

检查myinit-b.yaml是否存在

如果已经安装了calicoctl，检查calicoctl的位置 /usr/local/bin/ 还是 /usr/bin/ 是否与shell脚本相匹配

然后把这个脚本命名为cluster-a.sh 上传到云服务器，放在/home/myscript目录下

# 用dos2unix转换脚本

dos2unix /home/myscript/cluster-a.sh

# 给脚本赋予执行权限

chmod +x /home/myscript/cluster-a.sh

# 执行脚本

/home/myscript/cluster-a.sh

注意，这里用脚本部署cluster-a和以前不一样，因为kubernetes在1.19及以后的版本不支持用--feature -gates string的方法配置ipvs，所以这里采用配置文件文件部署（并且打印了一份日志）

kubeadm init --config=myinit.yaml | tee kubeadm-init.log

# 对于cluster-a，配置文件myinit.yaml如下。如果直接在linux中新建yaml文件然后粘贴内容，可能会出现文字的缩进问题，正确的做法是，先新建一个myinit文件（没有后缀名），然后粘贴配置文件，在把myinit文件重命名成myinit.yaml，这里advertiseAddress填的是cluster-a的内网ip，例如这里的10.1.1.111。

apiVersion: kubeadm.k8s.io/v1beta2

kind: InitConfiguration

localAPIEndpoint:

advertiseAddress: 10.1.1.111

bindPort: 6443

nodeRegistration:

taints:

- effect: NoSchedule

key: node-role.kubernetes.io/master

---

apiVersion: kubeadm.k8s.io/v1beta2

imageRepository: k8s.gcr.io

kind: ClusterConfiguration

kubernetesVersion: v1.19.1

apiServer:

certSANs:

- localhost

- 127.0.0.1

- 10.1.1.111

- 43.128.252.173

extraArgs:

authorization-mode: Node,RBAC

timeoutForControlPlane: 4m0s

networking:

podSubnet: 10.44.0.0/16

serviceSubnet: 10.45.0.0/16

---

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

featureGates:

SupportIPVSProxyMode: true

mode: ipvs

# 对于cluster-b，配置文件myinit.yaml如下

apiVersion: kubeadm.k8s.io/v1beta2

kind: InitConfiguration

localAPIEndpoint:

advertiseAddress: 10.1.1.48

bindPort: 6443

nodeRegistration:

taints:

- effect: NoSchedule

key: node-role.kubernetes.io/master

---

apiVersion: kubeadm.k8s.io/v1beta2

imageRepository: k8s.gcr.io

kind: ClusterConfiguration

kubernetesVersion: v1.19.1

networking:

podSubnet: 10.144.0.0/16

serviceSubnet: 10.145.0.0/16

---

apiVersion: kubeproxy.config.k8s.io/v1alpha1

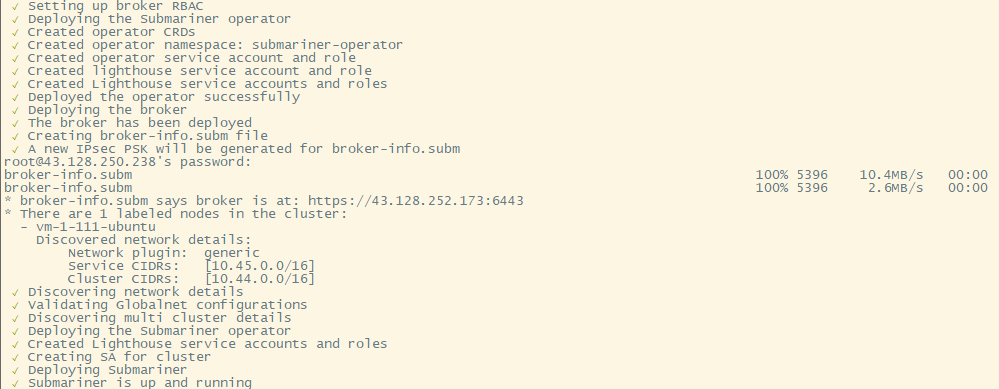
kind: KubeProxyConfiguration

featureGates:

SupportIPVSProxyMode: true

mode: ipvs

集群搭建完成，大概会是下面的样子



# 导出init的配置文件（导出来学习一下，可以不导出）

kubectl get cm -n kube-system kubeadm-config -o yaml > init.yml

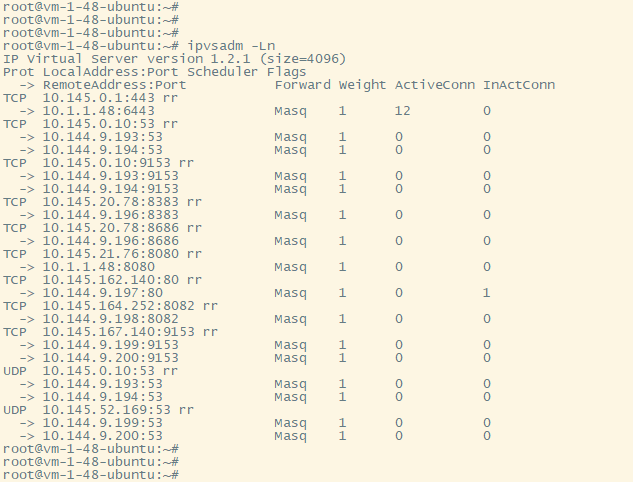
# 检查当前是不是ipvs

kubectl get cm -n kube-system kube-proxy -o yaml | grep -i mode



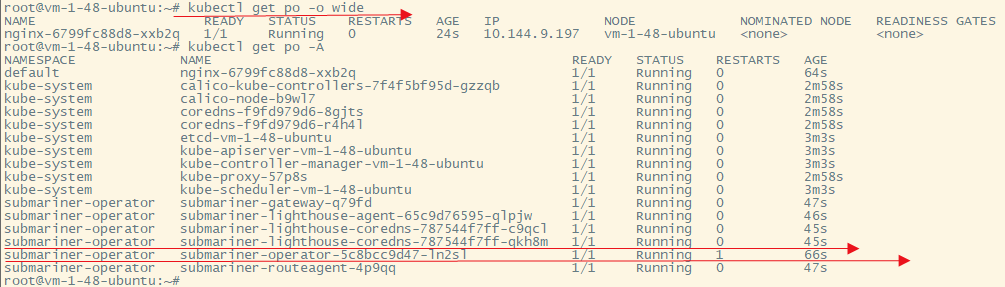
# 查看当前配置的虚拟服务和各个resource的权重

ipvsadm -Ln



# 检查各个pod工作状态

kubectl get po -A



六、安装nginx并且测试是否连通，参考这个（六、cluster-b安装nginx）

七、各种报错的处理

（1）error converting YAML to JSON: yaml: line 3: mapping values are not allowed in this context

大概意思是说yaml文件的格式不对，按照刚才在第五部中的步骤写配置文件就没有问题了

|  |  |  |  |
| --- | --- | --- | --- |
| k8s version | pod/pod connectivity across cluster | service discovery across cluster | note |
| 1.15.7 | ok | ok | flannel v0.14.0; subctl v0.6.0-dev, iptables mode |
| 1.16.7 | ok | ok | calicoctl v3.18.4; subctl v0.9.0-dev, iptables mode |
| 1.19.1 | ok | ok | calicoctl v3.18.4; subctl v0.9.0, ipvs mode |
| 1.19.1 | ok | ok | calicoctl v3.18.4; subctl v0.9.0, iptables mode |
|  |  |  |  |
| ubuntu 18.04; VPC CIDR: 10.0.0.0/16; POD CIDR: 10.0.32.0/24; SERVICE CIDR: 10.45.0.0/24; iptables | | | |
| ubuntu 18.04; VPC CIDR: 10.0.0.0/16; POD CIDR: 10.0.1.0/24; SERVICE CIDR: 10.145.0.0/24; iptables | | | |
| 1.18.4 |  |  |  |
|  |  |  |  |