pods像一个k8s虚拟出来的小型虚拟机

node像是 k8s本身需要用到的物理机

用标签识别pod，举例：创建4个nginx，4个nginx的标签都是一样的。

查看仓库

yum repolist

确保所有节点值为1

cat /proc/sys/net/bridge/bridge-nf-call-ip6tables

kubeadm init --kubernetes-version=v1.14.1 --pod-network-cidr=10.244.0.0/16

10.244.0.0/16 是flannel网络

允许使用swap

vim /etc/sysconfig/kubelet

KUBELET\_EXTRA\_ARGS=”—fail-swap-on=false”

kubeadm init --apiserver-advertise-address=192.168.1.201 --kubernetes-version=v1.14.1 --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=Swap

执行一些初始化命令

mkdir –p $HOME/.kube

cp –i /etc/kubernetes/admin.conf $HOME/.kube/config

安装flannel

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

获取组件状态

kubectl get cs

部署容器

kubectl run nginx-deploy --image=nginx:1.14-alpine --port=80 --replicas=1

通过service暴露端口

kubectl expose deployment nginx-deploy --name=nginx-service --port=80 --target-port=80 --protocol=TCP

kubectl get service

nginx-service ClusterIP 10.102.223.254 <none> 80/TCP 104s

在集群中就可以通过 curl 10.102.223.254来访问

kubectl get pods -n kube-system -o wide

coredns-fb8b8dccf-gz6gs 1/1 Running 0 4h24m 10.244.0.3 k8s001 <none>

coredns-fb8b8dccf-jmzxj 1/1 Running 0 4h24m 10.244.0.2 k8s001 <none>

dns服务在pod网络中

kube-system下的服务

kubectl get svc -n kube-system

kube-dns ClusterIP 10.96.0.10 <none> 53/UDP,53/TCP,9153/TCP

交互式进入busybox客户端

kubectl run client --image=busybox --replicas=1 -it --restart=Never

cat /etc/resolv.conf

nameserver 10.96.0.10

pod中的DNS就是kube-system命名空间下的kube-dns服务

master上

yum install -y bind-utils

让kube-system命名空间下的kube-dns服务来解析nginx-service.default.svc.cluster.local

dig -t A nginx-service.default.svc.cluster.local @10.96.0.10

解析出来的结果如下，域名nginx-service.default.svc.cluster.local对应pod的ip：10.102.223.254

nginx-service.default.svc.cluster.local. 5 IN A 10.102.223.254

**在busybox中执行wget -O - -q http://nginx-service 能访问到pod中的nginx，大致过程如下：**

在busybox容器中发送请求http://nginx-service，此请求需要DNS解析，busybox的DNS服务器是（kube-dns）10.96.0.10。kube-dns解析出来nginx-service.default.svc.cluster.local是ip是10.102.223.254，10.102.223.254就是nginx-service的ip。nginx-service的80端口已经映射到nginx-deploy-55d8d67cf-zt52t的80端口

展示pod的label

kubectl get pods --show-labels

nginx-deploy-55d8d67cf-zt52t 1/1 Running 0 3h18m pod-template-hash=55d8d67cf,run=nginx-deploy

kubectl describe svc nginx-service

Name: nginx-service

Namespace: default

Labels: run=nginx-deploy

Annotations: <none>

Selector: run=nginx-deploy

Type: ClusterIP

IP: 10.102.223.254

Port: <unset> 80/TCP

TargetPort: 80/TCP

Endpoints: 10.244.1.2:80

Session Affinity: None

Events: <none>

假设删除Endpoints: 10.244.1.2:80， k8s会重新再起一个新pod，新pod的ip会变化，但是label仍然是run=nginx-deploy，nginx-service会通过label选择器选中新pod。然后修改Endpoints:为新pod的ip

service给pod提供固定访问端点

kubectl run myapp --image=ikubernetes/myapp:v1 --replicas=2

暴露端口

kubectl expose deployment myapp --name=myapp-service --port=80

kubectl get svc

获取到的hostname会变化，从结果来看，是随机调度算法

curl 10.99.67.114/hostname.html

kubectl scale --replicas=5 deployment myapp

通过set image来改变deployment的版本，k8s一个个pod更新版本（灰度升级）

kubectl set image deployment myapp myapp=ikubernetes/myapp:v2

busybox中执行，查看版本

while true;do wget -O - -q myapp-service; sleep 1;done

deployment版本回滚

kubectl rollout undo deployment myapp

service是iptables中的规则地址

kubectl get svc

ping 10.99.67.114 无法ping通

curl 10.99.67.114 但可以访问，因为这iptables规则会转发到其他ip

yaml字段说明

kubectl explain pod

kubectl explain pod.spec

创建第一个pod，一个pod运行两个容器

vim pod-demo.yaml

apiVersion: v1

kind: Pod

metadata:

name: pod-demo

namespace: default

labels:

app: myapp

tier: frontend

spec:

containers:

- name: myapp

image: ikubernetes/myapp:v1

- name: busybox

image: busybox:latest

command:

- "bin/sh"

- "-c"

- "echo $(date) >> /usr/share/nginx/html/index.html; sleep 5"

通过yaml配置文件创建pod

kubectl create -f pod-demo.yaml

进入容器中

kubectl exec -it pod-demo -c myapp -- /bin/sh

删除资源清单中定义的资源

kubectl delete -f pod-demo.yaml

kubectl run 是通过控制器创建pod，删除pod后，控制器还在，pod会自动重建

通过yaml创建pod未指定控制器，删除pod后不会自动创建新pod