Master/node

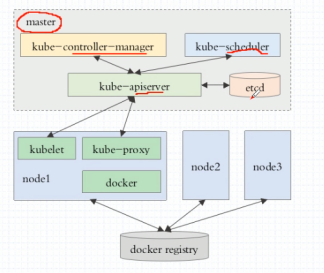
Master核心组件： API server，Scheduler，Controller-Manager etcd（存储组件）

Node核心组件： kubelet（核心组件）， docke（容器引擎（支持的引擎不止docker一个））， kube-proxy

Pod Label， label selector

Label： key=value

Label Selector：



Pod：

自主式Pod

控制器管理的Pod

ReplictionController(老版本控制器)

ReplicaSet（新版本控制器）

Deployment #通过控制replicaset来控制pod，最应该掌握的控制器之一

StatefuSet

Job，Ctonjob

AddOns： 附加组件

环境准备：、

Master， etcd： 172.18.0.70

Node1： 172.18.0.67

Node2： 172.18.0.68

前提：

1. 基于主机名通信： /etc/hosts
2. 时间同步 要不然会出一大推乱七八糟的报错
3. 关闭firewall和iptables.service

OS： Centos 7.3.1611 Rxtras仓库中

安装配置步骤：

1. Etcd cluster， 近master节点
2. Flannel，集群的所有节点
3. 配置k8s的master： 仅master节点

Kubernetes-master

启动的服务：

Kube-apiserver， kube-scheduler， kube-controller-manager

1. 配置k8s的各node节点：

Kubernetes-node

先设定启动docker服务

启动的k8s的服务

Kube-proxy， kubelet

k8s

先准备两台机器：

172.18.0.70 master #

172.18.0.67 node1

172.18.0.68 node2

注意： 一定要做好时间同步 要不然会有一大推莫名其妙的报错

然后两台机器都要配置docker和k8s的yum源

#docker18.09版本和k8s1.15.3版本

wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -O /etc/yum.repos.d/docker-ce.repo

https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/

直接都使用阿里的镜像源

[root@master ~]# yum list docker-ce --showduplicates | sort -r

#如果对版本有要求可以使用这个查看 18.09，安装对应的包

所有机器都安装

[root@master ~]# yum install docker-ce-18.09.3-3.el7

yum -y install kubelet kubeadm kubectl（api命令行工具）

然后准备启动docker，在启动docker之前建议先定义一个环境变量 随便另起一行写

[root@master ~]# vim /usr/lib/systemd/system/docker.service

Environment="HTTPS\_PROXY=http://www.ik8s.io:10080""

#意思是我们访问docker服务的时候通过代理下载相关的镜像文件

Environment="NO\_PROXY=127.0.0.8,172.18.0.0/16" #不用代理 本机

：wq

docker国内加速

mkdir -p /etc/docker

vim /etc/docker/daemon.json

{

  "registry-mirrors": ["https://lvb4p7mn.mirror.aliyuncs.com"]

}

上面这个貌似不太好用，

#!/bin/bash

# download k8s 1.15.3 images

# get image-list by 'kubeadm config images list --kubernetes-version=v1.15.3'

# gcr.azk8s.cn/google-containers == k8s.gcr.io

images=(

kube-apiserver:v1.15.3

kube-controller-manager:v1.15.3

kube-scheduler:v1.15.3

kube-proxy:v1.15.3

pause:3.1

etcd:3.3.10

coredns:1.3.1

)

for imageName in ${images[@]};do

docker pull gcr.azk8s.cn/google-containers/$imageName

docker tag gcr.azk8s.cn/google-containers/$imageName k8s.gcr.io/$imageName

docker rmi gcr.azk8s.cn/google-containers/$imageName

Done

#使用此脚本可以避免因为镜像源的问题！！！非常重要！！！

加载环境变量

[root@master ~]# systemctl daemon-reload

启动docker并使用docker info 查看

[root@master ~]# systemctl start docker && docker info

[root@master ~]# cat /proc/sys/net/bridge/bridge-nf-call-iptables

#默认是1就不用改

[root@master ~]# rpm -ql kubelet

/etc/kubernetes/manifests #清单目录

/etc/sysconfig/kubelet #配置文件

/usr/bin/kubelet

/usr/lib/systemd/system/kubelet.service

[root@master ~]# systemctl enable kubelet.service #设置成开机自启动就可以了

[root@master ~]# systemctl enable docker

[root@master ~]# kubelet init --help

#准备初始化

要把swap交换分区关了！！！！！！！！

初始化：

[root@master ~]# kubeadm init --kubernetes-version=v1.15.3 --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12

##ps： 如果因为镜像问题可以使用上面的脚本执行，直接拉取镜像

成功之后会有以下几条命令，执行一下 注意一定不要忘记执行!!

mkdir -p $HOME/.kube

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

sudo chown $(id -u):$(id -g) $HOME/.kube/config

初始化完成之后会有一段token哈希值，加入集群用的非常重要，可以把他先保存下来，如下：

kubeadm join 172.18.0.70:6443 --token 231d0r.ln9zof37rvnjplgm \

--discovery-token-ca-cert-hash sha256:fe3d0e900cd6dd51477be22089445dadf3608476f6857a061663e020147bc59c

[root@master ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

k8s.gcr.io/kube-proxy v1.15.3 232b5c793146 8 days ago 82.4MB

k8s.gcr.io/kube-apiserver v1.15.3 5eb2d3fc7a44 8 days ago 207MB

k8s.gcr.io/kube-controller-manager v1.15.3 e77c31de5547 8 days ago 159MB

k8s.gcr.io/kube-scheduler v1.15.3 703f9c69a5d5 8 days ago 81.1MB

k8s.gcr.io/coredns 1.3.1 eb516548c180 7 months ago 40.3MB

k8s.gcr.io/etcd 3.3.10 2c4adeb21b4f 9 months ago 258MB

k8s.gcr.io/pause 3.1 da86e6ba6ca1 20 months ago

成功之后在node1和node2上面都安装—

[root@node1 ~]# yum install docker-ce-18.09.7-3.el7 kubectl.x86\_64 kubelet.x86\_64 kubeadm.x86\_64

[root@node2 ~]# yum install docker-ce-18.09.7-3.el7 kubectl.x86\_64 kubelet.x86\_64 kubeadm.x86\_64

swap没关的话就忽略swap参数

vim /etc/sysconfig/kubelet

KUBELET\_EXTRA\_ARGS="--fail-swap-on=false"

KUBE\_PROXY\_MODE=ipvs

[root@master ~]# kubectl get cs #查看组件信息

NAME STATUS MESSAGE ERROR

scheduler Healthy ok

controller-manager Healthy ok

etcd-0 Healthy {"health":"true"}

[root@master ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

master NotReady master 32m v1.15.3

#这里的状态信息显示还未准备好是因为缺少一个重要组件flannel

在git上查找flannel获取以下命令：

[root@master ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

#这是一个在线的部署清单，基于此清单下载镜像flannel

[root@master ~]# kubectl get node

NAME STATUS ROLES AGE VERSION

master Ready master 41m v1.15.3

#现在就已经准备好了

[root@master ~]# kubectl get ns #查看名称空间

NAME STATUS AGE

default Active 43m

kube-node-lease Active 43m

kube-public Active 43m

kube-system Active 43m

把master的配置文件分别拷给node1和node2

[root@master ~] scp /usr/lib/systemd/system/docker.service node1:/usr/lib/systemd/system/docker.service

[root@master ~]# scp /usr/lib/systemd/system/docker.service node2:/usr/lib/systemd/system/docker.service

[root@master ~]# scp /etc/sysconfig/kubelet node1:/etc/sysconfig/kubelet

[root@master ~]# scp /etc/sysconfig/kubelet node2:/etc/sysconfig/kubelet

分别在node1和node2上面启动docker并设置成开启自启

[root@node1 ~]# systemctl start docker #启动docker

[root@node1 ~]# systemctl enable docker.service kubelet.service #设置成开启自启

[root@node2 ~]# systemctl start docker #启动docker

[root@node2 ~]# systemctl enable docker.service kubelet.service #设置成开启自启

[root@node1 ~]# kubeadm join 172.18.0.70:6443 --token k0q4vt.2ok77anvdhzip6s8 --discovery-token-ca-cert-hash sha256:77b7364b7fc2c886aa889488d3c06d6c1f3a8a1cfc48a8857354afa749c37630 --ignore-preflight-errors=Swap

[root@node2 ~]# kubeadm join 172.18.0.70:6443 --token k0q4vt.2ok77anvdhzip6s8 --discovery-token-ca-cert-hash sha256:77b7364b7fc2c886aa889488d3c06d6c1f3a8a1cfc48a8857354afa749c37630 --ignore-preflight-errors=Swap

#加入集群

[root@master ~]# kubectl describe node master #查看节点的详细信息，比较常用

[root@master ~]# kubectl cluster-info #查看集群信息

使用k8s进行增删改查

测试：

[root@master ~]# kubectl run nginx-ceshi --image=nginx --port=80 --replicas=1

#启动一个nginx pod

[root@master ~]# kubectl get pod #查看pod

NAME READY STATUS RESTARTS AGE

nginx-ceshi-748587595b-975t7 0/1 ContainerCreating 0 8s

[root@master ~]# kubectl get svc #查看服务

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 93m

myapp ClusterIP 10.97.57.166 <none> 80/TCP 22s

nginx ClusterIP 10.108.78.248 <none> 80/TCP 72m

[root@master ~]# kubectl scale --replicas=5 deployment myapp

#扩容副本，后面数值多少就是几个副本

[root@master ~]# kubectl run client1 --image=busybox --replicas=1 -it --restart=Never

#加一个-it表示创建完直接以交互式方式进去，创建一个客户端进行测试

/ # wget -O - -q 10.244.1.5/hostname.html #查看是从哪个节点上运行的

myapp-84cd4b7f95-mj6k6

[root@master ~]# kubectl describe deployment nginx-ceshi

#查看选择器详细信息

[root@master ~]# kubectl get deployment -w

#-w实时查看

[root@master ~]# kubectl set image deployment myapp myapp=ikubernetes/myapp:v2

deployment.extensions/myapp image updated

#使用set image进行版本升级

[root@master ~]# kubectl rollout undo deployment myapp

#使用rollout out进行回滚版本

[root@master ~]# kubectl edit svc myapp

##修改svc内容

27： type: NodePort

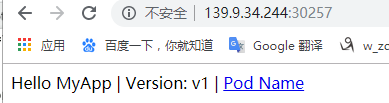
#大约27行，吧他修改为NodePort，就可以使用外网访问了，大小写不能错！

[root@master ~]# kubectl get svc #修改完之后使用get svc查看就会发现多一个端口

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 155m

myapp NodePort 10.97.57.166 <none> 80:30257/TCP 62m



删除node节点：

一共分两步：

1.排干此节点

kubectl drain 节点名称 --delete-local-data --force --ignore-daemonsets

2.删除此节点

kubectl delete node 节点名称

Yaml：

创建资源的方法：

apiserver仅接受JSON格式的资源定义；

yaml格式提供配置清单，apiserver可自动 将其转化为JSON格式，而后再提交。

大部分资源的配置清单：

[root@master ~]# kubectl explain pod #查看字段，相当于一个帮助

[root@master ~]# kubectl explain pods.status

##查看字段的详细帮助

apiversion： group/version

kind： 资源类别

metadata： 元数据

name

namespace

Latels

Annotations

每个资源的引用PATH

/api/GROUP/VERSION/namespeaces/NAMESPEACE/TYPE/NAME

Sepc: 期望的状态

Status： 当前状态，current state， 这五个一级字段由kubernetes集群维护；

下面是创建了一个pod里面有两个容器：格式注意下 yaml文件

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"

[root@master yamls]# kubectl create -f pod-damo.yaml

#根据清单创建pod

[root@master yamls]# kubectl delete -f pod-damo.yaml

[root@master yamls]# kubectl delete pods pod-demo

#删除pod

标签：

Key=value 键名和键值都必须小于63个字符

Key只能使用： 字母 数字 下划线 点号 ， 只能使用字母或数字开头

Value 可以为空，键值不能为空，只能以字母或数字开头及结尾，之间可使用

[root@master yamls]# kubectl get pod -l app #-l标签过滤

[root@master yamls]# kubectl label pods pod-demo release=ccc ##打标签

[root@master yamls]# kubectl get pods -l app --show-labels #查看

节点选择器：

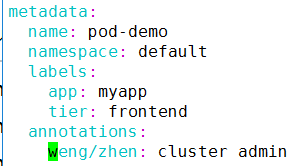
nodeSelector: yaml参数指定在哪个标签的node上运行，和containers平行

KEY=VALUE

NodeName :

标识：

annotations: #与label不同的地方在于，它不能用于挑选资源对象，仅用于为对象提供“元数据”，，写在metadata里面，如下



使用kubectl describe pods pod名字 查看

Pod的生命周期：

状态： Pending（挂起）, Running，Failed，Succeeded，Unknown

Pod生命周期中的重要行为：

初始化容器

容器探测

Liveness（存活性探测）

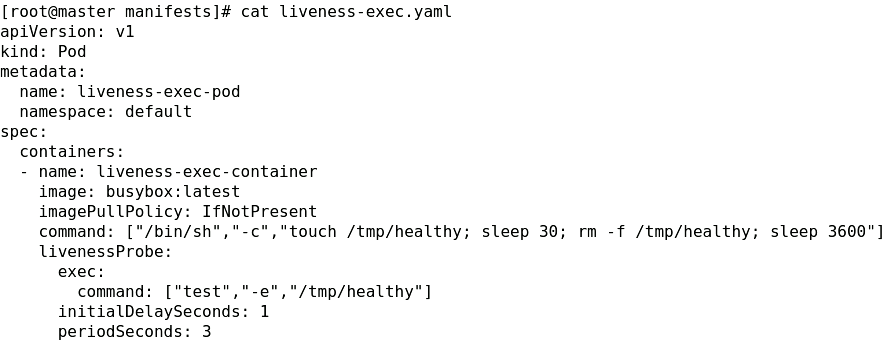
Readinessi(就绪与否)

--必须做的

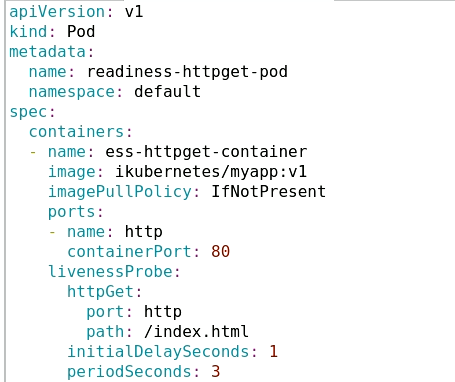
探针类型有三种：

ExecAction TCPSocketAction HTTPGetAction

示例：



HTTPGetAction：



[root@master ym]# kubectl explain pods.spec.container.lifecycle #查看pod生命周期帮助

有点晕这里

回顾： Pod

apiVersion, kind, metadata, status(只读)

spec:

containers:

nodeSelector

nodeName

restartPolicy: #重启策略

Always, Never, OnFailure

containers:

name

image

imagePullPolicy: Always Never IfNotPresent

ports:

name

contarnerPort #暴露端口

livenessProbe #存活性探测

readlinessProbe #就绪状态探测

Liftcycle

ExecAction： exec

TCPocketAction: tcpSocket

HTTPGetAction: httpGet

Pod 控制器

ReplictionController(老版本控制器)

ReplicaSet（新版本控制器）

Deployment #通过控制replicaset来控制pod，最应该掌握的控制器之一

DaemoSet #控制每个node上都有一个pod副本

Job #一次性

Cronjob

StatefulSet

[root@master ~]# kubectl explain rs #查看新版本控制器的手册