有道笔记地址： <https://note.youdao.com/s/3a7YwyAY>

**安装步骤**

选择4核8G（master）、8核16G（node1）、8核16G（node2） 三台机器，CentOS7.9

安装Docker

安装Kubernetes

安装KubeSphere前置环境

安装KubeSphere

**准备机器**

搭建K8S集群，准备三台2核4G的虚拟机(内存至少2G以上)，操作系统选择用centos 7以上版本

|  |  |
| --- | --- |
| 主机 | 说明 |
| 192.168.65.71 | k8s-master01 |
| 192.168.65.186 | k8s-node01 |
| 192.168.65.223 | k8s-node02 |

**1、安装Docker**

sudo yum remove docker\*

sudo yum install -y yum-utils

#配置docker的yum地址

sudo yum-config-manager \

--add-repo \

http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

#安装指定版本

sudo yum install -y docker-ce-20.10.7 docker-ce-cli-20.10.7 containerd.io-1.4.6

#启动&开机启动docker

systemctl enable docker --now

# docker镜像加速器配置

sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["https://jbw52uwf.mirror.aliyuncs.com"],

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

sudo systemctl daemon-reload

sudo systemctl restart docker

**2、安装Kubernetes**

**2.1 基本环境**

所有机器执行以下操作

每个机器使用内网ip互通

#关闭防火墙并设置开机不启动

systemctl stop firewalld

systemctl disable firewalld

#设置每个机器自己的hostname

hostnamectl set-hostname xxx

#配置hosts

cat >> /etc/hosts << EOF

192.168.65.71 k8s-master01

192.168.65.111 k8s-node01

192.168.65.223 k8s-node02

EOF

# 将 SELinux 设置为 permissive 模式（相当于将其禁用）

sudo setenforce 0

sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

#关闭swap

swapoff -a

sed -ri 's/.\*swap.\*/#&/' /etc/fstab

#允许 iptables 检查桥接流量

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

br\_netfilter

EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sudo sysctl --system

**2.2 安装kubelet、kubeadm、kubectl**

#配置k8s的yum源地址

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=0

repo\_gpgcheck=0

gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg

http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

#安装 kubelet，kubeadm，kubectl

sudo yum install -y kubelet-1.20.9 kubeadm-1.20.9 kubectl-1.20.9

#启动kubelet

sudo systemctl enable --now kubelet

**2.3 初始化master节点**

**2.3.1 下载各个机器需要的镜像**

sudo tee ./images.sh <<-'EOF'

#!/bin/bash

images=(

kube-apiserver:v1.20.9

kube-proxy:v1.20.9

kube-controller-manager:v1.20.9

kube-scheduler:v1.20.9

coredns:1.7.0

etcd:3.4.13-0

pause:3.2

)

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

docker pull registry.cn-hangzhou.aliyuncs.com/lfy\_k8s\_images/$imageName

done

EOF

chmod +x ./images.sh && ./images.sh

**2.3.2 初始化主节点**

# apiserver-advertise-address改为自己的master节点ip

#所有网络范围不重叠

kubeadm init \

--apiserver-advertise-address=192.168.65.71 \

--control-plane-endpoint=k8s-master01 \

--image-repository registry.cn-hangzhou.aliyuncs.com/lfy\_k8s\_images \

--kubernetes-version v1.20.9 \

--service-cidr=10.96.0.0/16 \

--pod-network-cidr=10.244.0.0/16

# 可以查看kubelet日志

journalctl -xefu kubelet

#如果初始化失败，重置kubeadm

kubeadm reset

rm -rf /etc/cni/net.d $HOME/.kube/config

#清理 iptables 规则

iptables -F

iptables -X

iptables -t nat -F

iptables -t nat -X

iptables -t mangle -F

iptables -t mangle -X

iptables -P INPUT ACCEPT

iptables -P FORWARD ACCEPT

iptables -P OUTPUT ACCEPT

**2.3.3 根据提示继续**

master成功后提示如下：

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

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

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of control-plane nodes by copying certificate authorities

and service account keys on each node and then running the following as root:

kubeadm join k8s-master01:6443 --token 50rexj.yb0ys92ynnxxbo2s \

--discovery-token-ca-cert-hash sha256:10fd9d2a9f4e2d7dff502aa3fb31a80f0372666efc92defde3707b499ba000e9 \

--control-plane

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join k8s-master01:6443 --token 50rexj.yb0ys92ynnxxbo2s \

--discovery-token-ca-cert-hash sha256:10fd9d2a9f4e2d7dff502aa3fb31a80f0372666efc92defde3707b499ba000e9

**设置.kube/config**

mkdir -p $HOME/.kube

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

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

**安装Calico网络插件**

curl https://docs.projectcalico.org/archive/v3.20/manifests/calico.yaml -O

kubectl apply -f calico.yaml

**2.3.4 加入node节点**

kubeadm join k8s-master01:6443 --token 50rexj.yb0ys92ynnxxbo2s \

--discovery-token-ca-cert-hash sha256:10fd9d2a9f4e2d7dff502aa3fb31a80f0372666efc92defde3707b499ba000e9

可以通过下面的命令重新生成令牌

kubeadm token create --print-join-command

**2.3.5 验证集群节点状态**

kubectl get nodes

**3、安装KubeSphere前置环境**

在 Kubernetes 上最小化安装 KubeSphere:<https://v3-1.docs.kubesphere.io/zh/docs/quick-start/minimal-kubesphere-on-k8s/>

如需在 Kubernetes 上安装 KubeSphere v3.1.1，您的 Kubernetes 版本必须为：1.17.x、1.18.x、1.19.x 或 1.20.x。

确保您的机器满足最低硬件要求：CPU > 1 核，内存 > 2 GB。

在安装之前，需要配置 Kubernetes 集群中的**默认**存储类型。

**3.1 nfs文件系统**

**3.1.1 安装nfs-server**

# 在每个机器。

yum install -y nfs-utils

# 在master 执行以下命令

echo "/nfs/data/ \*(insecure,rw,sync,no\_root\_squash)" > /etc/exports

# 执行以下命令，启动 nfs 服务;创建共享目录

mkdir -p /nfs/data

# 在master执行

systemctl enable rpcbind

systemctl enable nfs-server

systemctl start rpcbind

systemctl start nfs-server

# 使配置生效

exportfs -r

#检查配置是否生效

exportfs

**3.1.2 配置nfs-client（选做）**

showmount -e 192.168.65.71

mkdir -p /nfs/data

mount -t nfs 192.168.65.71:/nfs/data /nfs/data

**3.1.3 配置默认存储**

配置动态供应的默认存储类

## 创建了一个存储类

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: nfs-storage

annotations:

storageclass.kubernetes.io/is-default-class: "true"

provisioner: k8s-sigs.io/nfs-subdir-external-provisioner

parameters:

archiveOnDelete: "true" ## 删除pv的时候，pv的内容是否要备份

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nfs-client-provisioner

labels:

app: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: default

spec:

replicas: 1

strategy:

type: Recreate

selector:

matchLabels:

app: nfs-client-provisioner

template:

metadata:

labels:

app: nfs-client-provisioner

spec:

serviceAccountName: nfs-client-provisioner

containers:

- name: nfs-client-provisioner

image: registry.cn-hangzhou.aliyuncs.com/lfy\_k8s\_images/nfs-subdir-external-provisioner:v4.0.2

# resources:

# limits:

# cpu: 10m

# requests:

# cpu: 10m

volumeMounts:

- name: nfs-client-root

mountPath: /persistentvolumes

env:

- name: PROVISIONER\_NAME

value: k8s-sigs.io/nfs-subdir-external-provisioner

- name: NFS\_SERVER

value: 192.168.65.71 ## 指定自己nfs服务器地址

- name: NFS\_PATH

value: /nfs/data ## nfs服务器共享的目录

volumes:

- name: nfs-client-root

nfs:

server: 192.168.65.71

path: /nfs/data

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: default

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: nfs-client-provisioner-runner

rules:

- apiGroups: [""]

resources: ["nodes"]

verbs: ["get", "list", "watch"]

- apiGroups: [""]

resources: ["persistentvolumes"]

verbs: ["get", "list", "watch", "create", "delete"]

- apiGroups: [""]

resources: ["persistentvolumeclaims"]

verbs: ["get", "list", "watch", "update"]

- apiGroups: ["storage.k8s.io"]

resources: ["storageclasses"]

verbs: ["get", "list", "watch"]

- apiGroups: [""]

resources: ["events"]

verbs: ["create", "update", "patch"]

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: run-nfs-client-provisioner

subjects:

- kind: ServiceAccount

name: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: default

roleRef:

kind: ClusterRole

name: nfs-client-provisioner-runner

apiGroup: rbac.authorization.k8s.io

---

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: leader-locking-nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: default

rules:

- apiGroups: [""]

resources: ["endpoints"]

verbs: ["get", "list", "watch", "create", "update", "patch"]

---

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: leader-locking-nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: default

subjects:

- kind: ServiceAccount

name: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: default

roleRef:

kind: Role

name: leader-locking-nfs-client-provisioner

apiGroup: rbac.authorization.k8s.io

kubectl apply -f sc.yaml

#确认配置是否生效

kubectl get sc

**3.2 metrics-server**

集群指标监控组件

apiVersion: v1

kind: ServiceAccount

metadata:

labels:

k8s-app: metrics-server

name: metrics-server

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

labels:

k8s-app: metrics-server

rbac.authorization.k8s.io/aggregate-to-admin: "true"

rbac.authorization.k8s.io/aggregate-to-edit: "true"

rbac.authorization.k8s.io/aggregate-to-view: "true"

name: system:aggregated-metrics-reader

rules:

- apiGroups:

- metrics.k8s.io

resources:

- pods

- nodes

verbs:

- get

- list

- watch

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

labels:

k8s-app: metrics-server

name: system:metrics-server

rules:

- apiGroups:

- ""

resources:

- pods

- nodes

- nodes/stats

- namespaces

- configmaps

verbs:

- get

- list

- watch

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

labels:

k8s-app: metrics-server

name: metrics-server-auth-reader

namespace: kube-system

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: extension-apiserver-authentication-reader

subjects:

- kind: ServiceAccount

name: metrics-server

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

labels:

k8s-app: metrics-server

name: metrics-server:system:auth-delegator

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: system:auth-delegator

subjects:

- kind: ServiceAccount

name: metrics-server

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

labels:

k8s-app: metrics-server

name: system:metrics-server

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: system:metrics-server

subjects:

- kind: ServiceAccount

name: metrics-server

namespace: kube-system

---

apiVersion: v1

kind: Service

metadata:

labels:

k8s-app: metrics-server

name: metrics-server

namespace: kube-system

spec:

ports:

- name: https

port: 443

protocol: TCP

targetPort: https

selector:

k8s-app: metrics-server

---

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

k8s-app: metrics-server

name: metrics-server

namespace: kube-system

spec:

selector:

matchLabels:

k8s-app: metrics-server

strategy:

rollingUpdate:

maxUnavailable: 0

template:

metadata:

labels:

k8s-app: metrics-server

spec:

containers:

- args:

- --cert-dir=/tmp

- --kubelet-insecure-tls

- --secure-port=4443

- --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname

- --kubelet-use-node-status-port

image: registry.cn-hangzhou.aliyuncs.com/lfy\_k8s\_images/metrics-server:v0.4.3

imagePullPolicy: IfNotPresent

livenessProbe:

failureThreshold: 3

httpGet:

path: /livez

port: https

scheme: HTTPS

periodSeconds: 10

name: metrics-server

ports:

- containerPort: 4443

name: https

protocol: TCP

readinessProbe:

failureThreshold: 3

httpGet:

path: /readyz

port: https

scheme: HTTPS

periodSeconds: 10

securityContext:

readOnlyRootFilesystem: true

runAsNonRoot: true

runAsUser: 1000

volumeMounts:

- mountPath: /tmp

name: tmp-dir

nodeSelector:

kubernetes.io/os: linux

priorityClassName: system-cluster-critical

serviceAccountName: metrics-server

volumes:

- emptyDir: {}

name: tmp-dir

---

apiVersion: apiregistration.k8s.io/v1

kind: APIService

metadata:

labels:

k8s-app: metrics-server

name: v1beta1.metrics.k8s.io

spec:

group: metrics.k8s.io

groupPriorityMinimum: 100

insecureSkipTLSVerify: true

service:

name: metrics-server

namespace: kube-system

version: v1beta1

versionPriority: 100

**4、安装KubeSphere**

[官网：https://kubesphere.com.cn/](https://kubesphere.com.cn/)

安装文档：<https://v3-1.docs.kubesphere.io/zh/docs/quick-start/minimal-kubesphere-on-k8s/>

**4.1 下载核心文件**

如果下载不到，请复制附录的内容

wget https://github.com/kubesphere/ks-installer/releases/download/v3.1.1/kubesphere-installer.yaml

wget https://github.com/kubesphere/ks-installer/releases/download/v3.1.1/cluster-configuration.yaml

**4.2 修改cluster-configuration**

在 cluster-configuration.yaml中指定我们需要开启的功能

参照官网“启用可插拔组件”

<https://v3-1.docs.kubesphere.io/zh/docs/pluggable-components/>

**4.3 执行安装**

kubectl apply -f kubesphere-installer.yaml

kubectl apply -f cluster-configuration.yaml

**4.4 查看安装进度**

kubectl logs -n kubesphere-system $(kubectl get pod -n kubesphere-system -l app=ks-install -o jsonpath='{.items[0].metadata.name}') -f