**Kubernetes集群部署**

目录

[序 言 2](#_Toc40038723)

[实验拓扑 2](#_Toc40038724)

[一、 实验准备： 2](#_Toc40038725)

[1.1 配置host解析 2](#_Toc40038726)

[1.2 配置防火墙和selinux 2](#_Toc40038727)

[1.3 配置swap分区 3](#_Toc40038728)

[1.4 配置yum源 3](#_Toc40038729)

[二、 初始化kubernetes集群 4](#_Toc40038730)

[2.1 安装依赖软件 4](#_Toc40038731)

[2.2 安装kubeadm组件 4](#_Toc40038732)

[2.3初始化k8s集群 5](#_Toc40038733)

[2.4 安装calico 5](#_Toc40038734)

[2.5 安装dashboard监控 6](#_Toc40038735)

[三、部署ranchenr 14](#_Toc40038736)

[3.1 安装rancher 14](#_Toc40038737)

[3.2 配置rancher集群 14](#_Toc40038738)

# 序 言

# 实验拓扑

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IP | 主机名 | 配置 | OS版本 | K8s版本 |
| 192.168.80.100 | master.example.com | 2H4G | CentOS 7.6 | K8s 1.18.2 |
| 192.168.80.101 | node1.example.com | 2H4G | CentOS 7.6 | K8s 1.18.2 |
| 192.168.80.102 | node2.exaample.com | 2H4G | CentOS 7.6 | K8s 1.18.2 |

# 一、 实验准备：

## 1.1 配置host解析

在所有节点配置host解析，有条件的也可以配置dns解析

cat <<EOF > /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.80.100 master.example.com master

192.168.80.101 node1.example.com node1

192.168.80.102 node2.example.com node2

EOF

## 1.2 配置防火墙和selinux

注：测试环境为了操作方便，所以关闭防火墙，生产环境根据公司需求操作

systemctl disable firewalld && systemctl stop firewalld

sed -i 's#SELINUX=disabled#SELINUX=enforcing#g' /etc/selinux/config

## 1.3 配置swap分区

临时关闭：

swapof -a

永久关闭：

vim /etc/fstab 注销sawp

## 1.4 配置yum源

**配置Docker源**

# step 1: 安装必要的一些系统工具

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

# Step 2: 添加软件源信息

sudo yum-config-manager --add-repo https:*//mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo*

# Step 3: 更新并安装Docker-CE

sudo yum makecache fast

sudo yum -y install docker-ce

# Step 4: 开启Docker服务

sudo service docker start

**配置epel源**

wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo

**配置Kubernetes源**

cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

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

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

**配置镜像加速器：**

针对Docker客户端版本大于 1.10.0 的用户

您可以通过修改daemon配置文件/etc/docker/daemon.json来使用加速器

sudo mkdir -p /etc/docker

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

{

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

}

EOF

sudo systemctl daemon-reload

sudo systemctl restart docker

**配置内核属性：**

cat <<EOF > /etc/sysctl.d/k8s.conf

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

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

net.ipv4.ip\_forward = 1

EOF

**配置资源限制：**

echo "\* soft nofile 655360" >> /etc/security/limits.conf

echo "\* hard nofile 655360" >> /etc/security/limits.conf

echo "\* soft nproc 655360" >> /etc/security/limits.conf

echo "\* hard nproc 655360" >> /etc/security/limits.conf

echo "\* soft memlock unlimited" >> /etc/security/limits.conf

echo "\* hard memlock unlimited" >> /etc/security/limits.conf

echo "DefaultLimitNOFILE=1024000" >> /etc/systemd/system.conf

echo "DefaultLimitNPROC=1024000" >> /etc/systemd/system.conf

# 二、 初始化kubernetes集群

## 2.1 安装依赖软件

yum install ‐y conntrack ipvsadm ipset jq sysstat curl iptables libseccomp bash‐completion yum‐utils device‐mapper

‐persistent‐data lvm2 net‐tools conntrack‐tools vim libtool‐ltdl

## 2.2 安装kubeadm组件

yum -y install kubelet kubeadm kubectl

## 2.3初始化k8s集群

kubeadm init --image-repository registry.aliyuncs.com/google\_containers --kubernetes-version=v1.18.2 --pod-network-cidr=10.244.0.0/16

初始化完毕之后，会返回以下信息

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

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/

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

kubeadm join 192.168.80.100:6443 --token 4gkvbk.orh2e9w1a56m8cla \

--discovery-token-ca-cert-hash sha256:9cd9db2a3855446fbc45754a08be45277a58986ee3b49802a23c2e6d2bcb4396

根据提示，需要执行以下命令，配置管理员用户操作权限

mkdir -p $HOME/.kube

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

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

注：上面的token并不是永久生效的，过一段时间之后，会过期，需要重新生成

## 2.4 安装calico

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

## 2.5 安装dashboard监控

wget <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-rc7/aio/deploy/recommended.yaml> #下载yaml文件

kubectl apply -f recommended.yaml #部署dashboard

注：由于网络原因，不能直接访问国外网站，可以复制以下配置文件，直接部署即可

apiVersion: v1

kind: Namespace

metadata:

name: kubernetes-dashboard

---

apiVersion: v1

kind: ServiceAccount

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kubernetes-dashboard

---

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kubernetes-dashboard

spec:

type: NodePort

ports:

- port: 443

targetPort: 8443

selector:

k8s-app: kubernetes-dashboard

---

apiVersion: v1

kind: Secret

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard-certs

namespace: kubernetes-dashboard

type: Opaque

---

apiVersion: v1

kind: Secret

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard-csrf

namespace: kubernetes-dashboard

type: Opaque

data:

csrf: ""

---

apiVersion: v1

kind: Secret

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard-key-holder

namespace: kubernetes-dashboard

type: Opaque

---

kind: ConfigMap

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard-settings

namespace: kubernetes-dashboard

---

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kubernetes-dashboard

rules:

# Allow Dashboard to get, update and delete Dashboard exclusive secrets.

- apiGroups: [""]

resources: ["secrets"]

resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dashboard-certs", "kubernetes-dashboard-csrf"]

verbs: ["get", "update", "delete"]

# Allow Dashboard to get and update 'kubernetes-dashboard-settings' config map.

- apiGroups: [""]

resources: ["configmaps"]

resourceNames: ["kubernetes-dashboard-settings"]

verbs: ["get", "update"]

# Allow Dashboard to get metrics.

- apiGroups: [""]

resources: ["services"]

resourceNames: ["heapster", "dashboard-metrics-scraper"]

verbs: ["proxy"]

- apiGroups: [""]

resources: ["services/proxy"]

resourceNames: ["heapster", "http:heapster:", "https:heapster:", "dashboard-metrics-scraper", "http:dashboard-metrics-scraper"]

verbs: ["get"]

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

rules:

# Allow Metrics Scraper to get metrics from the Metrics server

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

resources: ["pods", "nodes"]

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

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kubernetes-dashboard

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: kubernetes-dashboard

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: kubernetes-dashboard

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: kubernetes-dashboard

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kubernetes-dashboard

---

kind: Deployment

apiVersion: apps/v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kubernetes-dashboard

spec:

replicas: 1

revisionHistoryLimit: 10

selector:

matchLabels:

k8s-app: kubernetes-dashboard

template:

metadata:

labels:

k8s-app: kubernetes-dashboard

spec:

containers:

- name: kubernetes-dashboard

image: kubernetesui/dashboard:v2.0.0

imagePullPolicy: Always

ports:

- containerPort: 8443

protocol: TCP

args:

- --auto-generate-certificates

- --namespace=kubernetes-dashboard

# Uncomment the following line to manually specify Kubernetes API server Host

# If not specified, Dashboard will attempt to auto discover the API server and connect

# to it. Uncomment only if the default does not work.

# - --apiserver-host=http://my-address:port

volumeMounts:

- name: kubernetes-dashboard-certs

mountPath: /certs

# Create on-disk volume to store exec logs

- mountPath: /tmp

name: tmp-volume

livenessProbe:

httpGet:

scheme: HTTPS

path: /

port: 8443

initialDelaySeconds: 30

timeoutSeconds: 30

securityContext:

allowPrivilegeEscalation: false

readOnlyRootFilesystem: true

runAsUser: 1001

runAsGroup: 2001

volumes:

- name: kubernetes-dashboard-certs

secret:

secretName: kubernetes-dashboard-certs

- name: tmp-volume

emptyDir: {}

serviceAccountName: kubernetes-dashboard

nodeSelector:

"kubernetes.io/os": linux

# Comment the following tolerations if Dashboard must not be deployed on master

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

---

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: dashboard-metrics-scraper

name: dashboard-metrics-scraper

namespace: kubernetes-dashboard

spec:

ports:

- port: 8000

targetPort: 8000

selector:

k8s-app: dashboard-metrics-scraper

---

kind: Deployment

apiVersion: apps/v1

metadata:

labels:

k8s-app: dashboard-metrics-scraper

name: dashboard-metrics-scraper

namespace: kubernetes-dashboard

spec:

replicas: 1

revisionHistoryLimit: 10

selector:

matchLabels:

k8s-app: dashboard-metrics-scraper

template:

metadata:

labels:

k8s-app: dashboard-metrics-scraper

annotations:

seccomp.security.alpha.kubernetes.io/pod: 'runtime/default'

spec:

containers:

- name: dashboard-metrics-scraper

image: kubernetesui/metrics-scraper:v1.0.4

ports:

- containerPort: 8000

protocol: TCP

livenessProbe:

httpGet:

scheme: HTTP

path: /

port: 8000

initialDelaySeconds: 30

timeoutSeconds: 30

volumeMounts:

- mountPath: /tmp

name: tmp-volume

securityContext:

allowPrivilegeEscalation: false

readOnlyRootFilesystem: true

runAsUser: 1001

runAsGroup: 2001

serviceAccountName: kubernetes-dashboard

nodeSelector:

"kubernetes.io/os": linux

# Comment the following tolerations if Dashboard must not be deployed on master

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

volumes:

- name: tmp-volume

emptyDir: {}

注：不管是master还是node，如果想清空kubernetes设置的话，需要执行kubeadm reset命令。

# 三、部署ranchenr

## 3.1 安装rancher

sudo docker run -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher

## 3.2 配置rancher集群

将现有kubernetes集群加入