无锡城市职业技术学院

**实 验（实 训）报 告**

**所属课程名称**  云平台构建与运维

**班 级** 计网2201

**学 号**  1422040311

**姓 名**  马紫艳

**指导教师**  徐峰

无锡城市职业技术学院教务处监制

**项目名称：kubernetes安装和使用** 日期：2024.4.24

|  |
| --- |
| **一、实验（实训）概述：** |
| **【目的及要求】**  **Kubernetes集群各节点的规划：各节点内存均为8GB，使用CentOS-7-x86\_64-DVD-1804.iso安装操作系统。**  **【实验环境】**  **k8s-master虚拟机、k8s-node虚拟机** |
| **二、实验（实训）内容：** |
| **【实验（实训）过程】（步骤、记录、数据、程序等）**  **原生Kubernetes云平台部署**  **1、创建虚拟机，新建k8s-master虚拟机和k8s-node虚拟机，创建完成后均拍摄快照，然后查看对应的网络IP地址，如图10-1-1，图10-1-2，图10-1-3，图10-1-4，图10-1-5所示。**  屏幕截图 2024-04-24 202810  图10-1-1  屏幕截图 2024-04-24 202958  图10-1-2  屏幕截图 2024-04-24 203053  图10-1-3  屏幕截图 2024-04-24 203836  图10-1-4  屏幕截图 2024-04-24 203958  图10-1-5   1. **配置hosts：**   **两个节点配置相同，以master节点为例，如图10-1-6，图10-1-7所示。**  [root@master ~]# vi /etc/hosts  192.168.100.131 master  192.168.100.132 node  屏幕截图 2024-04-24 210222  图10-1-6  屏幕截图 2024-04-24 210203  图10-1-7   1. **停止防火墙和SElinux：**   **在master和node节点进行以下配置，以master节点为例。**   1. **停止防火墙，如图10-1-8所示。**   [root@master ~]# systemctl stop firewalld  [root@master ~]# systemctl disable firewalld  屏幕截图 2024-04-24 210457  图10-1-8   1. **停止SELinux，如图10-1-9，图10-1-10所示。**   [root@master ~]# setenforce 0  [root@master ~]# vi /etc/selinux/config  SELINUX=permissive  屏幕截图 2024-04-24 210629  图10-1-9  屏幕截图 2024-04-24 210621  图10-1-10  **4、关闭swap：**  **在master和node节点进行以下配置，以master节点为例。**   1. **停用Swap，删除fstab中的SWAP挂载，如图10-1-11，图10-1-12所示。**   [root@master ~]# swapoff -a  [root@master ~]# vi /etc/fstab  屏幕截图 2024-04-24 210730  图10-1-11  删除：  /dev/mapper/centos-swap swap swap defaults 0 0  屏幕截图 2024-04-24 210757  图10-1-12  **5、配置YUM：**  **在master进行以下配置：**   1. **将CentOS-7-x86\_64-DVD-1806.iso和K8S.tar.gz通过SFTP传输到master节点/root目录，如图10-1-13所示。**   [root@master ~]# ls  anaconda-ks.cfg CentOS-7-x86\_64-DVD-1804.iso K8S.tar.gz  屏幕截图 2024-04-24 213336  图10-1-13   1. **创建挂载目录，编辑fstab，在最下面添加一行命令：/root/CentOS-7-x86\_64-DVD-1804.iso /opt/centos iso9660 defaults,ro,loop 0 0,如图10-1-14,图10-1-15所示。**   [root@master ~]# mkdir /opt/centos 屏幕截图 2024-04-24 213433  图10-1-14  [root@master ~]# vi /etc/fstab  在最下面添加一行：  /root/CentOS-7-x86\_64-DVD-1804.iso /opt/centos iso9660  defaults,ro,loop 0 0  屏幕截图 2024-04-24 213811  图10-1-15   1. **挂载ISO，如图10-1-16所示。**   [root@master ~]# mount -a  [root@master ~]# mount  屏幕截图 2024-04-24 214316  图10-1-16   1. **解压K8S，如图10-1-17所示。**   [root@master ~]# mkdir /opt/K8S  [root@master ~]# tar -zxf K8S.tar.gz -C /opt/K8S 屏幕截图 2024-04-24 214623  图10-1-17   1. **编辑YUM配置文件，如图10-1-18，图10-1-19所示。**   [root@master ~]# cd /etc/yum.repos.d/  [root@master yum.repos.d]# rm -f \*  [root@master yum.repos.d]# vi local.repo  屏幕截图 2024-04-24 214908  图10-1-18  [centos]  name=centos  baseurl=file:///opt/centos  gpgcheck=0  [K8S]  name=K8S  baseurl=file:///opt/K8S/Kubernetes  gpgcheck=0  屏幕截图 2024-04-24 215305  图10-1-19   1. **清空缓存，如图10-1-20所示。**   [root@master ~]# yum clean all  屏幕截图 2024-04-24 215423  图10-1-20   1. **生成新缓存，如图10-1-21所示。**   [root@master ~]# yum makecache 屏幕截图 2024-04-24 215432  图10-1-21   1. **安装vsftpd，如图10-1-22所示。**   [root@master ~]# yum -y install vsftpd  屏幕截图 2024-04-24 215537  图10-1-22   1. **编辑vsftpd配置文件，增加一行：anon\_root=/opt命令，如图10-1-23所示。**   [root@master ~]# vi /etc/vsftpd/vsftpd.conf  增加一行：  anon\_root=/opt  屏幕截图 2024-04-24 215811  图10-1-23   1. **启动和启用服务，如图10-1-24所示。**   [root@master ~]# systemctl start vsftpd  [root@master ~]# systemctl enable vsftpd  屏幕截图 2024-04-24 220115  图10-1-24   1. **在node节点编辑YUM配置文件，如图10-1-25，图10-1-26所示。**   [root@node ~]# cd /etc/yum.repos.d/  [root@node yum.repos.d]# rm -f \*  [root@node yum.repos.d]# vi ftp.repo  屏幕截图 2024-04-24 220306  图10-1-25  [centos]  name=centos  baseurl=ftp://192.168.100.60/centos  gpgcheck=0  [K8S]  name=K8S  baseurl=ftp://192.168.100.60/K8S/Kubernetes  gpgcheck=0  **屏幕截图 2024-04-24 220809**  **图10-1-26**   1. **在node节点清空缓存，如图10-1-27所示。**   [root@node ~]# yum clean all  **屏幕截图 2024-04-24 220831**  **图10-1-27**   1. **在node节点生成新缓存，如图10-1-28所示。**   [root@node ~]# yum makecache  **屏幕截图 2024-04-24 220900**  **图10-1-28**  **6、配置NTP：**  **（1）在master节点上安装chrony，编辑配置文件，master节点修改/etc/chrony.conf文件，删除默认NTP服务器，指定上游公共NTP服务器，并允许其他节点同步时间，如图10-1-29所示。**  [root@master ~]# yum -y install chrony  **屏幕截图 2024-04-24 221235**  **图10-1-29**  **删除4个server，添加以下配置，如图10-1-30所示。**  [root@master ~]# vi /etc/chrony.conf  删除 4 个 server，添加以下配置：  local stratum 10  server master iburst  allow all  **屏幕截图 2024-04-24 221346**  **图10-1-30**   1. **启动服务及启用服务，如图10-1-31所示。**   [root@master ~]# systemctl start chronyd  [root@master ~]# systemctl enable chronyd  **屏幕截图 2024-04-24 221458**  **图10-1-31**   1. **查看时间同步源，如图10-1-32所示。**   [root@master ~]# chronyc sources -v  MS Name/IP address Stratum Poll Reach LastRx Last sample  ==================================================================  =============  ^\* master 10 6 77 8 -1ns[-8190ns] +/- 20us  **屏幕截图 2024-04-24 221556**  **图10-1-32**  **7、配置路由转发：**  **RHEL/7CentOS7上的一些用户报告了由于iptables被绕过而导致流量路由不正确的问题，所以需要在各节点开启路由转发。**  **在master和node节点进行以下配置，以master节点为例。**   1. **创建/etc/sysctl.d/k8s.conf,如图10-1-33，图10-1-34所示。**   [root@master ~]# vi /etc/sysctl.d/k8s.conf  **屏幕截图 2024-04-24 222405**  **图10-1-33**  net.ipv4.ip\_forward = 1  net.bridge.bridge-nf-call-ip6tables = 1  net.bridge.bridge-nf-call-iptables = 1  **屏幕截图 2024-04-24 222623**  **图10-1-34**   1. **载入内核模块，使配置生效，如图10-1-35所示。**   [root@master ~]# modprobe br\_netfilter  [root@master ~]# sysctl -p /etc/sysctl.d/k8s.conf  **图10-1-35**  **8、配置IPVS：**  **由于IPVS已经加入到了内核的主干，所以为kube-proxy开启IPVS的前提需要加载以下的内核模块。在所有节点执行以下操作。**  **在master和node节点下进行以下配置，以master节点为例。**   1. **编辑文件，如图10-1-36，图10-1-37所示。**   [root@master ~]# vi /etc/sysconfig/modules/ipvs.modules  **屏幕截图 2024-04-24 223454**  **图10-1-36**  #!/bin/bash  modprobe -- ip\_vs  modprobe -- ip\_vs\_rr  modprobe -- ip\_vs\_wrr  modprobe -- ip\_vs\_sh  modprobe -- nf\_conntrack\_ipv4  **屏幕截图 2024-04-24 223710**  **图10-1-37**   1. **增加执行权限，执行脚本，显示已载入的模块，如图10-1-28所示。**   [root@master ~]# chmod 755 /etc/sysconfig/modules/ipvs.modules  [root@master ~]# bash /etc/sysconfig/modules/ipvs.modules  [root@master ~]# lsmod | grep -e ip\_vs -e nf\_conntrack\_ipv4  nf\_conntrack\_ipv4 15053 0  nf\_defrag\_ipv4 12729 1 nf\_conntrack\_ipv4  ip\_vs\_sh 12688 0 ip\_vs\_wrr 12697 0  ip\_vs\_rr 12600 0  ip\_vs 141432 6 ip\_vs\_rr,ip\_vs\_sh,ip\_vs\_wrr  nf\_conntrack 133053 2 ip\_vs,nf\_conntrack\_ipv4  libcrc32c 12644 3 xfs,ip\_vs,nf\_conntrack  **屏幕截图 2024-04-24 224129**  **图10-1-38**   1. **安装ipset软件包，如图10-1-39所示。**   [root@master ~]# yum -y install ipset ipvsadm  **屏幕截图 2024-04-24 224141**  **图10-1-39**   1. **配置Docker：**   **Kubernetes默认的容器运行时仍然是Docker，使用的是Kubelet中内置dockershim CRI实现。需要注意的是，这里统一使用Docker18.09版本。**  **在master和node节点进行以下配置，以node节点为例。**   1. **安装yum-utils，如图10-1-40所示。**   [root@master ~]# yum -y install yum-utils  **屏幕截图 2024-04-24 230823**  **图10-1-40**   1. **安装支持软件，如图10-1-41所示。**   [root@master ~]# yum install -y device-mapper-persistent-data lvm2  **屏幕截图 2024-04-24 230836**  **图10-1-41**   1. **安装Docker，如图10-1-42所示。**   [root@master ~]# yum -y install docker-ce-18.09.6 docker-ce-cli-18.09.6 containerd.io  **屏幕截图 2024-04-24 230849**  **图10-1-42**   1. **创建目录，编辑Docker配置文件，如图10-1-43，图10-1-44所示。**   [root@master ~]# mkdir /etc/docker  **屏幕截图 2024-04-24 231322**  **图10-1-43**  [root@master ~]# vi /etc/docker/daemon.json  {  "exec-opts": ["native.cgroupdriver=systemd"]  }  **屏幕截图 2024-04-24 231653**  **图10-1-44**   1. **启动及启用Docker服务，如图10-1-45所示。**   [root@master ~]# systemctl daemon-reload  [root@master ~]# systemctl start docker  [root@master ~]# systemctl enable docker  **屏幕截图 2024-04-24 232004**  **图10-1-45**   1. **查看docker info，如图10-1-46所示。**   [root@master ~]# docker info | grep Cgroup  Cgroup Driver: systemd  **屏幕截图 2024-04-24 232034**  **图10-1-46**   1. **安装Kubeadm工具：**   **Kubeadm负责与其他节点集群通信，并进行本节点Pod和容器生命周期的管理。Kubeadm是Kubernetes的自动化部署工具，降低了部署难度，提高效率。Kubectl是Kubernetes集群管理工具。**  **在master和node节点进行以下配置，以master节点为例。**   1. **安装软件，如图10-1-47所示。**   [root@master ~]# yum -y install kubelet-1.14.1 kubeadm-1.14.1 kubectl-1.14.1    **图10-1-47**   1. **启动和启用kubelet,如图10-1-48所示。**   [root@master ~]# systemctl enable kubelet  [root@master ~]# systemctl start kubelet    **图10-1-48**   1. **初始化Kubernetes集群：**   **在master节点进行以下操作。**   1. **加载镜像，如图10-1-49所示。**   [root@master ~]# cd /opt/K8S  [root@master K8S]# ./kubernetes\_base.sh    **图10-1-49**   1. **初始化Kubernetes集群，如图10-1-50，图10-1-51所示。**   [root@master ~]#  kubeadm init  --apiserver-advertise-address 192.168.100.60  --kubernetes-version="v1.14.1"  --pod-network-cidr=10.16.0.0/16  --image-repository=registry.aliyuncs.com/google\_containers  ……  Your Kubernetes control-plane has initialized successfully!  ……  Then you can join any number of worker nodes by running the following on each as root:  kubeadm join 192.168.100.60:6443 --token tn8fi0.32ubkm8u987igkbo \  --discovery-token-ca-cert-hash  sha256:897afbc11a85991f995d0d8b007b152460a94478a554cb89d895b99bf835f041    图10-1-50    **图10-1-51**   1. **后续配置**   **Kubectl默认会在执行的用户home目录下面的.kube目录下寻找config文件，配置kubectl工具。如图10-1-52所示。**  [root@master ~]# mkdir -p $HOME/.kube  [root@master ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  [root@master ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config    **图10-1-52**   1. **检测集群状态，如图10-1-53所示。**   [root@master ~]# kubectl get cs  NAME STATUS MESSAGE ERROR  scheduler Healthy ok  controller-manager Healthy ok  etcd-0 Healthy {"health":"true"}    **图10-1-53**   1. **配置Kubernetes网络：**   **在master节点部署flannel网络，使用kubectl apply命令安装网络。**  **（1）进入yaml目录，如图10-1-54所示。**  [root@master ~]# cd /opt/K8S/yaml    **图10-1-54**  **（2）部署flannel网络，如图10-1-55所示。**  [root@master yaml]# kubectl apply -f kube-flannel.yaml    **图10-1-55**  **（3）查看状态，如图10-1-56所示。**  [root@master ~]# kubectl get pods -n kube-system  NAME READY STATUS RESTARTS AGE  coredns-8686dcc4fd-mfsln 1/1 Running 0 6m35s  coredns-8686dcc4fd-nnjk7 1/1 Running 0 6m35s  etcd-master 1/1 Running 0 5m47s  kube-apiserver-master 1/1 Running 0 5m41s  kube-controller-manager-master 1/1 Running 0 5m55s  kube-flannel-ds-amd64-lfvp8 1/1 Running 0 42skube-proxy-kcdcx 1/1 Running 0 6m35s  kube-scheduler-master 1/1 Running 0 5m32s    **图10-1-56**   1. **将node节点加入集群：**   **（1）将K8S.tar.gz通过SFTP传输到node节点的/root目录，如图10-1-57所示。**  [root@node ~]# ls  anaconda-ks.cfg K8S.tar.gz    **图10-1-57**  **（2）解压K8S，如图10-1-58所示。**  [root@node ~]# tar -zxf K8S.tar.gz    **图10-1-58**  **（3）加载镜像，如图10-1-59，图10-1-60所示。**  [root@node ~]# ls  anaconda-ks.cfg images K8S.tar.gz Kubernetes kubernetes\_base.sh yaml  [root@node ~]# ./kubernetes\_base.sh    图10-1-59    **图10-1-60**  **（4）在node节点执行之前初始化Kubernetes集群时最后提供的加入节点的命令，如图10-1-61所示。**  [root@node ~]# kubeadm join 192.168.100.60:6443 --token tn8fi0.32ubkm8u987igkbo \  >  --discovery-token-ca-cert-hash  sha256:897afbc11a85991f995d0d8b007b152460a94478a554cb89d895b99bf835f041  ……  This node has joined the cluster:  ……    **图10-1-61**  **（5）如果没有记下当时给出的提示，可以在 master 节点执行以下命令重新生成 Token 并查看，如图10-1-62所示。**  [root@master ~]# kubeadm token create --print-join-command  kubeadm join 192.168.100.60:6443  --token ldcvqa.ijb0u7jyj8rwv5w0  --discovery-token-ca-cert-hash  sha256:897afbc11a85991f995d0d8b007b152460a94478a554cb89d895b99bf835f041    **图10-1-62**   1. **在master节点检查各节点状态，如图10-1-63所示。**   [root@master ~]# kubectl get nodes  NAME STATUS ROLES AGE VERSION  master Ready master 6m40s v1.14.1  node Ready <none> 48s v1.14.1    **图10-1-63**   1. **安装Dashboard：** 2. **进入/opt/K8S/yaml目录，使用kubectl create命令安装Dashboard，如图10-1-64所示。**   [root@master ~]# cd /opt/K8S/yaml  [root@master yaml]# kubectl create -f kubernetes-dashboard.yaml  [root@master yaml]# kubectl create -f dashboard-adminuser.yaml    **图10-1-64**   1. **检测所有Pod状态，如图10-1-65所示。**   [root@master ~]# kubectl get pods --all-namespaces -o wide  NAMESPACE NAME READY STATUS RESTARTS AGE IP NODE NOMINATED  NODE READINESS GATES  kube-system kubernetes-dashboard-5f7b999d65-g9qv5 1/1 Running 0 93s  10.16.1.2 node <none> <none>    **图10-1-65**   1. **通过命令检查到kubernetes-dashboard被调度到node节点运行，通过Firefox浏览器中输入node节点地址“http://192.168.200.132:30000”,即可访问Kubernetes Dashboard,如图10-1-66所示。**   图10-1-66  **使用Kubectl运行容器**   1. **载入或下载镜像：**   **（1）在node节点载入nginx:latest镜像。**  [root@node ~]# ls  anaconda-ks.cfg nginx\_latest.tar  [root@node ~]# docker load -i nginx\_latest.tar  **（2）或从Docker Hub下载镜像，如图10-1-67所示。**  [root@node ~]# docker pull nginx:latest    图10-1-67  **（3）查看镜像，如图10-1-68所示。**  [root@node ~]# docker images | grep nginx  nginx latest 540a289bab6c 2 months ago 126MB    图10-1-68   1. **使用kubectl运行容器：**   **（1）创建depioyment，如图10-1-69所示。**  [root@master ~]# kubectl create deployment nginx --image=nginx  deployment.apps/nginx created    图10-1-69  **（2）查看Pods，验证Pods是否正常运行，如图10-1-70所示。**  [root@master ~]# kubectl get pods  NAME READY STATUS RESTARTS AGE  nginx-65f88748fd-wf62w 1/1 Running 0 12s  可以看到容器在运行中。    图10-1-70  **（3）查看所有deployment，如图10-1-71所示。**  [root@master ~]# kubectl get deployment  NAME READY UP-TO-DATE AVAILABLE AGE  nginx 1/1 1 1 35s    图10-1-71  **（4）采用NodePort的方式来暴露nginx服务，如图10-1-72所示。**  [root@master ~]# kubectl expose deployment nginx --port=80 --type=NodePortservice/nginx exposed    图10-1-72  **（5）查看Service，如图10-1-73所示。**  [root@master ~]# kubectl get svc  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 23h  nginx NodePort 10.100.233.224 <none> 80:31507/TCP 19s    图10-1-73  **（6）在本机通过浏览器访问Nginx应用，如图10-1-74所示。**    图10-1-74  **（7）pod动态伸缩。**  **运行一下命令，将容器数量更新为3个，如图10-1-75所示。**  [root@master ~]# kubectl scale deployment nginx --replicas=3  deployment.extensions/nginx scaled    图10-1-75  **（8）查看pod的容器数量，如图10-1-76所示。**  [root@master ~]# kubectl get pods  NAME READY STATUS RESTARTS AGE  nginx-65f88748fd-ckfhf 1/1 Running 0 43s  nginx-65f88748fd-hp99g 1/1 Running 0 43s  nginx-65f88748fd-wf62w 1/1 Running 0 5m42s    图10-1-76   1. **删除deployment，如图10-1-77所示。**   [root@master ~]# kubectl delete deployment nginx  deployment.extensions "nginx" deleted    图10-1-77   1. **删除Service，如图10-1-78所示。**   [root@master ~]# kubectl delete svc nginx  service "nginx" deleted    图10-1-78  **【结论】（结果、分析）**  **Kubernetes是一个开源的容器编排平台，用于自动化部署、扩展和管理容器化应用程序。它提供了一种分布式系统的框架，用于通过自动化和编排容器的生命周期，实现应用程序的高可用性、弹性伸缩、服务发现和负载均衡等功能。Kubernetes提供了容器编排的能力，包括部署和管理容器、自动扩展、负载均衡、弹性伸缩、故障恢复等，使得应用程序可以在Kubernetes集群中高效和稳定地运行。**  **在实际应用中，通常会使用Docker来构建和打包应用程序的容器镜像，然后使用Kubernetes来部署和管理这些容器镜像，实现容器化应用程序的编排和运行。因此，Docker和Kubernetes通常是一起使用的，分别解决了容器化和容器编排的问题。容器占用资源少、部署快，每个应用可以被打包成一个容器镜像，每个应用与容器间成一对一关系也使容器有更大优势，使用容器可以在build或release 的阶段，为应用创建容器镜像，因为每个应用不需要与其余的应用堆栈组合，也不依赖于生产环境基础结构，这使得从研发到测试、生产能提供一致环境。类似地，容器比虚拟机轻量、更“透明”，这更便于监控和管理。** |
| **三、指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：徐峰**  **批阅日期：2023年 月 日** |