# Kubeadm离线安装k8s高可用集群

## 系统初始化

**服务器规划：**

192.168.30.24 k8s-master1192.168.30.25 k8s-master2192.168.30.26 k8s-node1192.168.30.30 k8s-node2192.168.30.31 k8s-node3192.168.30.32 k8s-VIP192.168.30.33 k8s-slb2

分别修改服务器主机名

hostnamectl set-hostname k8s-master01hostnamectl set-hostname k8s-master02hostnamectl set-hostname k8s-node01hostnamectl set-hostname k8s-node02hostnamectl set-hostname k8s-node03

**系统初始化：**

|  |
| --- |
| 关闭防火墙：  # systemctl stop firewalld  # systemctl disable firewalld    关闭selinux：  # setenforce 0 # 临时  # sed -i 's/enforcing/disabled/' /etc/selinux/config # 永久  sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/sysconfig/selinuxsed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/selinux/config  关闭swap：  # swapoff -a # 临时  # vim /etc/fstab # 永久  vim /etf/fstab #ubuntu  # sed -ri 's/.\*swap.\*/#&/' /etc/fstab  sed -ri '/^[^#]\*swap/s@^@#@' /etc/fstab  swapoff -a && sysctl -w vm.swappiness=0  同步系统时间：  # cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime    调整系统时区  timedatectl set-timezone Asia/Shanghai  timedatectl set-local-rtc 0  systemctl restart rsyslog  systemctl restart crond  添加hosts：  # vim /etc/hosts  192.168.10.142 k8s-master1  192.168.10.145 k8s-master2  192.168.10.146 k8s-node1  192.168.10.144 k8s-node2  192.168.10.144 k8s-node2  配置网桥过滤功能  cat >/etc/sysctl.d/k8s.conf<<EOF  net.bridge.bridge-nf-call-ip6tables = 1  net.bridge.bridge-nf-call-iptables = 1  net.ipv4.ip\_forward = 1  vm.swappiness=0  #####################  net.ipv4.tcp\_tw\_recycle=0  vm.panic\_on\_oom=0  fs.inotify.max\_user\_watches=89100  fs.file-max=52706963  fs.nr\_open=52706963  net.ipv6.conf.all.disable\_ipv6=1  net.netfilter.nf\_conntrack\_max=2310720  EOF  挂载br\_netfilter和modprobe nf\_conntrack模块  modprobe br\_netfilter  modprobe nf\_conntrack  ubuntu  sudo modprobe overlay  sudo modprobe br\_netfilter  使配置生效  sysctl -p /etc/sysctl.d/k8s.conf  查看：  sysctl --system  #查看是否生成相关文件  ls /proc/sys/net/bridge  资源配置文件  echo "\* soft nofile 65536" >> /etc/security/limits.conf  echo "\* hard nofile 65536" >> /etc/security/limits.conf  echo "\* soft nproc 65536" >> /etc/security/limits.conf  echo "\* hard nproc 65536" >> /etc/security/limits.conf  echo "\* soft memlock unlimited" >> /etc/security/limits.conf  echo "\* hard memlock unlimited" >> /etc/security/limits.conf  升级内核 不联网请自行百度（实在不行也不影响集群使用）  rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm  yum --enablerepo=elrepo-kernel install -y kernel-lt  grub2-set-default 'CentOS Linux (4.4.189-1.el7.elrepo.x86\_64) 7 (Core)' |

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 【lvs需要ipvs模块实现】

cat > /etc/sysconfig/modules/ipvs.modules <<EOF

#!/bin/bash

modprobe -- ip\_vs

modprobe -- ip\_vs\_rr

modprobe -- ip\_vs\_wrr

modprobe -- ip\_vs\_sh

modprobe -- nf\_conntrack\_ipv4

EOF

执行脚本并查看是否正常加载内核模块：

修改脚本权限

chmod 755 /etc/sysconfig/modules/ipvs.modules

执行脚本

bash /etc/sysconfig/modules/ipvs.modules

查看是否已经正确加载所需的内核模块

lsmod | grep -e ip\_vs -e nf\_conntrack\_ipv4

然后安装 ipset 和 ipvsadm

yum install -y ipset ipvsadm

**非高可用集群下面这步无需操作**

## 安装haproxy和keepalived（VIP）

所有master节点安装

yum install -y haproxy

配置文件 vim /etc/haproxy/haproxy.cfg

|  |
| --- |
| #---------------------------------------------------------------------  # Example configuration for a possible web application. See the  # full configuration options online.  #  # https://www.haproxy.org/download/2.1/doc/configuration.txt  # https://cbonte.github.io/haproxy-dconv/2.1/configuration.html  #  #---------------------------------------------------------------------  #---------------------------------------------------------------------  # Global settings  #---------------------------------------------------------------------  global  # to have these messages end up in /var/log/haproxy.log you will  # need to:  #  # 1) configure syslog to accept network log events. This is done  # by adding the '-r' option to the SYSLOGD\_OPTIONS in  # /etc/sysconfig/syslog  #  # 2) configure local2 events to go to the /var/log/haproxy.log  # file. A line like the following can be added to  # /etc/sysconfig/syslog  #  # local2.\* /var/log/haproxy.log  #  log 127.0.0.1 local2  # chroot /var/lib/haproxy  pidfile /var/run/haproxy.pid  maxconn 4000  # user haproxy  # group haproxy  # daemon  # turn on stats unix socket  #stats socket /var/lib/haproxy/stats  #---------------------------------------------------------------------  # common defaults that all the 'listen' and 'backend' sections will  # use if not designated in their block  #---------------------------------------------------------------------  defaults  mode http  log global  option httplog  option dontlognull  option http-server-close  option forwardfor except 127.0.0.0/8  option redispatch  retries 3  timeout http-request 10s  timeout queue 1m  timeout connect 10s  timeout client 1m  timeout server 1m  timeout http-keep-alive 10s  timeout check 10s  maxconn 3000  #---------------------------------------------------------------------  # main frontend which proxys to the backends  #---------------------------------------------------------------------  frontend kubernetes-apiserver  mode tcp  bind \*:16443 ## 监听16443端口  # bind \*:443 ssl # To be completed ....  acl url\_static path\_beg -i /static /images /javascript /stylesheets  acl url\_static path\_end -i .jpg .gif .png .css .js  default\_backend kubernetes-apiserver  #---------------------------------------------------------------------  # round robin balancing between the various backends  #---------------------------------------------------------------------  backend kubernetes-apiserver  mode tcp # 模式tcp  balance roundrobin # 采用轮询的负载算法  # k8s-apiservers backend # 配置apiserver，端口6443  server k8s-master01 192.168.1.58:6443 check  server k8s-master02 192.168.0.22:6443 check  server k8s-master03 192.168.0.41:6443 check |

分别在2个master节点启动haproxy

systemctl restart haproxy && systemctl status haproxy

systemctl enable haproxy 设置开机自启

netstat -lntup|grep haproxy

yum install -y keepalived

vim /etc/keepalived/keepalived.conf

|  |
| --- |
| ! Configuration File for keepalived  global\_defs {  router\_id k8s  script\_user root  enable\_script\_security  }  vrrp\_script check\_haproxy {  script "/bin/bash -c 'if [[ $(netstat -nlp | grep 16443) ]]; then exit 0; else systemctl stop keepalived;fi'"  interval 3  weight -2  fall 10  rise 2  }  vrrp\_instance VI\_1 {  state MASTER #其他节点改成BACKUP  interface ens192 #网卡  virtual\_router\_id 51  priority 500  #以下的配置是当你的上游交换机禁止ARP广播的时候需要添加的配置，如果不添加的话，keepalived会出现脑裂的情况  unicast\_src\_ip 192.168.1.58 #本机ip  unicast\_peer {  192.168.0.22 #对端ip 其他节点的ip  192.168.0.41  }  advert\_int 1  authentication {  auth\_type PASS  auth\_pass 88888888  }  virtual\_ipaddress {  192.168.0.235  }  track\_script {  check\_haproxy  }  } |

分别在2台master节点启动keepalived

systemctl start keepalived && systemctl status keepalived

systemctl enable keepalived

修改keepalived.conf文件中的virtual\_router\_id值，修改priority值值越大权重越大主节点必须改成最大值 auth\_pass 密码需一致

## 安装 kubelet、kubectl、kubeadm

在台联网的机器 设置k8s的yum源

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

[kubernetes]

name=Kubernetes Repo

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

enabled=1

gpgcheck=1

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

EOF

kubelet: 在集群中的每个节点上用来启动 pod 和 container 等。

kubectl: 用来与集群通信的命令行工具。

kubeadm: 用来初始化集群的指令。

下载kube到本地

yum -y install kubelet-1.23.6 kubectl-1.23.6 kubeadm-1.23.6

设置开机自启

systemctl start kubelet && systemctl enable kubelet

## 安装docker

Docker -v 查看是否已安装

未安装用提供的离线安装包一键安装

在线安装：

yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

yum install docker-ce docker-ce-cli containerd.io -y

cri-dockerd安装

下载rpm包

<https://github.com/Mirantis/cri-dockerd/releases>

yum install cri-dockerd-0.3.16-3.fc35.x86\_64.rpm 安装即可

修改配置

vim /etc/docker/daemon.json

{

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

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

"log-opts": {

"max-size": "100m"

}

}

sudo systemctl stop docker.socket

systemctl daemon-reload

## 安装master

生成kubeadm安装yaml

kubeadm config print init-defaults > kubeadm-config.yaml

|  |
| --- |
| apiVersion: kubeadm.k8s.io/v1beta2  bootstrapTokens:  - groups:  - system:bootstrappers:kubeadm:default-node-token  token: 7t2weq.bjbawausm0jaxury #自动生成  ttl: 24h0m0s  usages:  - signing  - authentication  kind: InitConfiguration  localAPIEndpoint:  advertiseAddress: 10.0.0.201 # 本机ip  bindPort: 6443  nodeRegistration:  criSocket: /run/containerd/containerd.sock  name: k8s-master01  taints:  - effect: NoSchedule  key: node-role.kubernetes.io/master  ---  apiServer:  certSANs:  - 192.168.0.235  - 192.168.1.58  - 192.168.0.22  - 192.168.0.41  - 127.0.0.1  extraArgs:  authorization-mode: Node,RBAC  timeoutForControlPlane: 4m0s  apiVersion: kubeadm.k8s.io/v1beta3  certificatesDir: /etc/kubernetes/pki  clusterName: kubernetes  controlPlaneEndpoint: "192.168.0.235:16443" #VIP地址或主master地址  controllerManager: {}  dns:  type: CoreDNS  etcd:  local:  dataDir: /var/lib/etcd  imageRepository: registry.aliyuncs.com/google\_containers  kind: ClusterConfiguration  kubernetesVersion: v1.23.6  networking:  dnsDomain: cluster.local  podSubnet: 10.244.0.0/16  serviceSubnet: 10.96.0.0/12  scheduler: {}  # 这个配置让k8s使用ipvs功能 无需ipvs无需配置  apiVersion: kubeproxy.config.k8s.io/v1alpha1  kind: KubeProxyConfiguration  mode: ipvs # iptables / ipvs ,ipvs 在1.18版本，需要linux 内核 4.x  ---  # 这个配置让k8s使用systemd  apiVersion: kubelet.config.k8s.io/v1beta1  kind: KubeletConfiguration  cgroupDriver: systemd |

进行集群初始化安装

kubeadm init --config kubeadm-config.yaml --upload-certs

简易的初始化步骤

kubeadm init --kubernetes-version=v1.23.0 --image-repository registry.aliyuncs.com/google\_containers --service-cidr=10.96.0.0/12 --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=Swap

如果安装失败**kubeadm reset -f 清除上次安装信息**

然后**rm -rf /var/lib/etcd rm -rf /root/.kube rm -rf /etc/kube…**

## 部署calico网络

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

sed -i 's/192.168.0.0/10.244.0.0/g' calico.yaml

搜索name: CALICO\_IPV4POOL\_CIDR 修改value: "10.96.0.0/12"

https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

## 安装kubectl自动补全

yum -y install bash-completion

source /usr/share/bash-completion/bash\_completion #执行加载

source <(kubectl completion bash) #临时生效

echo "source <(kubectl completion bash)" >> ~/.bashrc #当前用户永久生效

## 动态创建pv&pvc

官方github地址https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner

创建ServiceAccout账号

vim rbac.yaml

|  |
| --- |
| apiVersion: v1  kind: ServiceAccount  metadata:  name: nfs-client-provisioner  namespace: default  ---  kind: ClusterRole  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: nfs-client-provisioner-runner  rules:  - 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  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 |

创建NFS provisioner

vim nfs-provisioner.yaml

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: nfs-client-provisioner  labels:  app: nfs-client-provisioner  # replace with namespace where provisioner is deployed  namespace: default #与RBAC文件中的namespace保持一致  spec:  replicas: 1  selector:  matchLabels:  app: nfs-client-provisioner  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/open-ali/nfs-client-provisioner  volumeMounts:  - name: nfs-client-root  mountPath: /persistentvolumes  env:  - name: PROVISIONER\_NAME  value: nfs-storage-provisioner #provisioner名称,请确保该名称与 nfs-StorageClass.yaml文件中的provisioner名称保持一致  - name: NFS\_SERVER  value: 192.168.1.60  - name: NFS\_PATH  value: /opt/data  volumes:  - name: nfs-client-root  nfs:  server: 192.168.1.60  path: /opt/data |

创建storageclass

vim nfs-storageclass.yaml

|  |
| --- |
| apiVersion: storage.k8s.io/v1  kind: StorageClass  metadata:  name: managed-nfs-storage1  provisioner: nfs-storage-provisioner  reclaimPolicy: Retain  parameters:  archiveOnDelete: "true" |

创建测试的pvc申请空间

vim test\_pvc.yaml

|  |
| --- |
| apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: pv-1g  namespace: default  spec:  storageClassName: managed-nfs-storage1 #与storageclass的name保持一致  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 1Gi |

报错信息Normal ExternalProvisioning 8s (x2 over 13s) persistentvolume-controller waiting for a volume to be created, either by external provisioner "qgg-nfs-storage" or manually created by system administrator

修改apiserver的配置

cat /etc/kubernetes/manifests/kube-apiserver.yaml

搜索 - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key下面添加

- --feature-gates=RemoveSelfLink=false # 添加这个配置