Kubernetes 节点配置文档

```
## 1. 配置 DNS
cat >> /etc/resolv.conf <<EOF
nameserver 223.5.5.5
nameserver 114.114.114.114
EOF
```

2. 安装依赖包

```
yum update -y
yum reinstall conntrack socat curl ebtables ipset ipvsadm chrony libseccomp -y
```

3. 设置服务器时区

```
timedatectl set-timezone Asia/Shanghai
```

4. 配置时间同步

```
sed -i 's/^pool pool.*/pool cn.pool.ntp.org iburst/g' /etc/chrony.conf
systemctl enable chronyd --now
chronyc sourcestats -v
```

5. 关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

6. 禁用 SELinux

```
sed -i 's/^SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
setenforce 0
```

8. 配置内核swappiness 参数调整

```
cat <<EOF > /etc/sysctl.d/k8s.conf
vm.swappiness = 0
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
modprobe br_netfilter
sysctl -p /etc/sysctl.d/k8s.conf
```

9. 配置 IPVS 模块

```
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
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntrack_ipv4

# 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
```

11. 禁用 Swap

```
swapoff -a
sed -i '/swap/s/^\(.*\)$/#\1/g' /etc/fstab
```

12. 配置 Hosts 文件

```
cat >> /etc/hosts << EOF
10.0.0.100 master
10.0.0.101 node01
EOF
```

配置iptables的ACCEPT规则

```
iptables -F && iptables -X && iptables -F -t nat && iptables -X -t nat && iptables
-P FORWARD ACCEPT
```

下载安装 Containerd

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

# 查看可安装版本
[root@master ~]# yum list | grep containerd
containerd.io.x86_64                     1.6.10-3.1.el7              @docker-ce-
stable
# 执行安装
[root@master ~]# yum -y install containerd.io
# 查看
[root@master ~]# rpm -qa | grep containerd
containerd.io-1.6.10-3.1.el7.x86_64


# 创建目录
mkdir -p /etc/containerd
containerd config default > /etc/containerd/config.toml
# 替换配置文件
sed -i 's#SystemdCgroup = false#SystemdCgroup = true#' /etc/containerd/config.toml
sed -i 's|registry.k8s.io/pause:|registry.aliyuncs.com/google_containers/pause:|'
/etc/containerd/config.toml

systemctl enable containerd
systemctl start containerd
systemctl status containerd

[root@master ~]# ctr version
Client:
  Version:  1.6.10
  Revision: 770bd0108c32f3fb5c73ae1264f7e503fe7b2661
  Go version: go1.18.8

Server:
  Version:  1.6.10
  Revision: 770bd0108c32f3fb5c73ae1264f7e503fe7b2661
  UUID: 10b91012-6b24-4059-bf92-d71d269a5fbc
```

containerd 加速

参考

https://zhuanlan.zhihu.com/p/702497587

https://github.com/containerd/containerd/blob/main/docs/hosts.md

https://github.com/DaoCloud/public-image-mirror

https://www.cnblogs.com/plain-coder/p/18502024

```
# 修改config.toml

vim /etc/containerd/config.toml

# 追加内容
[plugins."io.containerd.grpc.v1.cri".registry]
   config_path = "/etc/containerd/certs.d"


mkdir -p /etc/containerd/certs.d && cd /etc/containerd/certs.d/

# 加速


# docker hub镜像加速
mkdir -p /etc/containerd/certs.d/docker.io
cat > /etc/containerd/certs.d/docker.io/hosts.toml << EOF
server = "https://docker.io"
[host."https://dockerproxy.cn"]
  capabilities = ["pull", "resolve"]

[host."https://docker.m.daocloud.io"]
  capabilities = ["pull", "resolve"]
EOF

# registry.k8s.io镜像加速
mkdir -p /etc/containerd/certs.d/registry.k8s.io
tee /etc/containerd/certs.d/registry.k8s.io/hosts.toml << 'EOF'
server = "https://registry.k8s.io"

[host."https://k8s.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# docker.elastic.co镜像加速
mkdir -p /etc/containerd/certs.d/docker.elastic.co
tee /etc/containerd/certs.d/docker.elastic.co/hosts.toml << 'EOF'
server = "https://docker.elastic.co"

[host."https://elastic.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# gcr.io镜像加速
mkdir -p /etc/containerd/certs.d/gcr.io
tee /etc/containerd/certs.d/gcr.io/hosts.toml << 'EOF'
server = "https://gcr.io"

[host."https://gcr.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF
```

```
# ghcr.io镜像加速
mkdir -p /etc/containerd/certs.d/ghcr.io
tee /etc/containerd/certs.d/ghcr.io/hosts.toml << 'EOF'
server = "https://ghcr.io"

[host."https://ghcr.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# k8s.gcr.io镜像加速
mkdir -p /etc/containerd/certs.d/k8s.gcr.io
tee /etc/containerd/certs.d/k8s.gcr.io/hosts.toml << 'EOF'
server = "https://k8s.gcr.io"

[host."https://k8s-gcr.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# mcr.m.daocloud.io镜像加速
mkdir -p /etc/containerd/certs.d/mcr.microsoft.com
tee /etc/containerd/certs.d/mcr.microsoft.com/hosts.toml << 'EOF'
server = "https://mcr.microsoft.com"

[host."https://mcr.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# nvcr.io镜像加速
mkdir -p /etc/containerd/certs.d/nvcr.io
tee /etc/containerd/certs.d/nvcr.io/hosts.toml << 'EOF'
server = "https://nvcr.io"

[host."https://nvcr.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# quay.io镜像加速
mkdir -p /etc/containerd/certs.d/quay.io
tee /etc/containerd/certs.d/quay.io/hosts.toml << 'EOF'
server = "https://quay.io"

[host."https://quay.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

# registry.jujucharms.com镜像加速
mkdir -p /etc/containerd/certs.d/registry.jujucharms.com
tee /etc/containerd/certs.d/registry.jujucharms.com/hosts.toml << 'EOF'
server = "https://registry.jujucharms.com"

[host."https://jujucharms.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF
```

```
# rocks.canonical.com镜像加速
mkdir -p /etc/containerd/certs.d/rocks.canonical.com
tee /etc/containerd/certs.d/rocks.canonical.com/hosts.toml << 'EOF'
server = "https://rocks.canonical.com"

[host."https://rocks-canonical.m.daocloud.io"]
  capabilities = ["pull", "resolve", "push"]
EOF

systemctl restart containerd.service
## 测试
ctr images pull --hosts-dir "/etc/containerd/certs.d" docker.io/nginx:latest


crictl 命令， 会自动使用/etc/containerd/certs.d目录下的配置镜像加速 （推荐）
```

master 安装三大件（ kubelet、kubeadm、kubectl）

添加 Kubernetes 的 APT 软件源：

```
cat > /etc/yum.repos.d/kubernetes.repo << EOF
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF

yum install -y kubelet kubeadm kubectl
## 安装指定 yum install -y kubelet-1.25.0 kubeadm-1.25.0 kubectl-1.25.0

crictl config runtime-endpoint /run/containerd/containerd.sock


# 启动 kubelet 并设置开机自启
systemctl daemon-reload
systemctl enable kubelet && systemctl start kubelet


# -apiserver-advertise-address=192.168.1.100  master节点
# --service-cidr=10.96.0.0/12 Kubernetes 的默认值
# --pod-network-cidr
# Flannel 插件：通常使用 10.244.0.0/16，这是 Flannel 默认的网络范围。
# Calico 插件：通常使用 192.168.0.0/16，但你可以根据需要配置其他范围。
# Weave 插件：通常使用 10.32.0.0/12。


# 生成配置文件
```

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

默认的config的配置文件

```
# 修改内容
advertiseAddress: 192.168.1.92  # 修改为自己的master节点IP
name: master    # 修改为master主机名
imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers  # 修改为阿里云镜像地址
kubernetesVersion: 1.25.0     # 确认是否为要安装版本，版本根据执行：kubelet --version得来
podSubnet: 172.16.0.0/16      # networking: 下添加pod网络
scheduler: {}                 # 添加模式为 ipvs
cgroupDriver: systemd         # 指定 cgroupDriver 为 systemd
```

修改后的配置文件

```
apiVersion: kubeadm.k8s.io/v1beta3
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 10.0.0.100
  bindPort: 6443
nodeRegistration:
  criSocket: unix:///var/run/containerd/containerd.sock
  imagePullPolicy: IfNotPresent
  name: master
  taints: null
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta3
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
dns: {}
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers
kind: ClusterConfiguration
```

```
kubernetesVersion: 1.28.0
networking:
  dnsDomain: cluster.local
  podSubnet: 172.16.0.0/16
  serviceSubnet: 10.96.0.0/12
scheduler: {}
---

apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
mode: ipvs

---

apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
cgroupDriver: systemd
```

```
# 初始化部署
kubeadm init --config=kubeadm.yaml --v=5
```

```
# 初始化后删除
rm -rf /etc/kubernetes/manifests/*
kubeadm reset --cri-socket unix:///var/run/cri-dockerd.sock
# 删除 CNI 配置文件（尽量不做删除）
rm -rf /etc/cni/net.d/

# 清理 iptables 规则
iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X

# 如果使用 IPVS，清理 IPVS 规则
ipvsadm --clear

# 最好重启（或者回归初始化）
reboot
```

执行拷贝 kubeconfig 文件

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

添加节点（node1、node2）

记住初始化集群上面的配置和操作要提前做好，将 master 节点上面的 $HOME/.kube/config 文件拷贝到

node 节点对应的文件中，安装 kubeadm、kubelet、kubectl，然后执行上面初始化完成后提示的 join 命 令即可：

- 如果忘记了上面的 join 命令可以使用命令kubeadm token create --print-join-command重新获取。

添加节点

```
## node1:

[root@node1 ~]# kubeadm join 10.0.0.100:6443 --token abcdef.0123456789abcdef \
    --discovery-token-ca-cert-hash
sha256:c6eddcbad9bf812f226cbb503a9a0323b7098136245fa7817fd624b94aed0722

[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get
cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

查看集群状态：

```
[root@master /opt]$ kubectl get nodes
NAME     STATUS     ROLES           AGE    VERSION
master   NotReady   control-plane   4h1m   v1.28.2
node01   NotReady   <none>          4s     v1.28.2
```

安装网络插件

可以看到是 NotReady 状态，这是因为还没有安装网络插件，**必须部署一个** 容器网络接口 **(CNI) 基于 Pod 网络附加组件，以便您的 Pod 可以相互通信。在安装网络之前，集群 DNS (CoreDNS) 不会启动。**接下来安装网络插件，可以在以下两个任一地址中选择需要安装的网络插件(我选用的第二个地址安装)，这里我们安装 calio

- https:// kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
- https://projectcalico.docs.tigera.io/archive/v3.23/getting-started/kubernetes/self-managed-onprem/onpremises#install-calico

```
#下载calico文件
curl
https://projectcalico.docs.tigera.io/archive/v3.23/manifests/calico.yaml -O
编辑calico.yaml文件:

注:文件默认IP为:192.168.0.0/16
- name: CALICO_IPV4POOL_CIDR     # 由于在init的时候配置的172网段,所以这里需要修
改
  value: "172.16.0.0/16"


## 安装yq
curl -L
https://gh.tryxd.cn/https://github.com/mikefarah/yq/releases/download/v4.6.1
/yq_linux_amd64.tar.gz | tar xz
mv yq_linux_amd64 /usr/local/bin/yq
chmod +x /usr/local/bin/yq

##  提取所需要的镜像
[root@master /opt]$ yq e '.spec.template.spec.containers[].image'
calico.yaml
docker.io/calico/node:v3.23.5
---
docker.io/calico/kube-controllers:v3.23.5
[root@master /opt]$
## 拉取镜像(前面配置的加速)
ctr images pull --hosts-dir "/etc/containerd/certs.d"
docker.io/calico/node:v3.23.5
ctr images pull --hosts-dir "/etc/containerd/certs.d" docker.io/calico/kube-
controllers:v3.23.5
## 第二个方法
ctr images pull swr.cn-north-4.myhuaweicloud.com/ddn-
k8s/docker.io/calico/node:v3.23.5
ctr images tag  swr.cn-north-4.myhuaweicloud.com/ddn-
k8s/docker.io/calico/node:v3.23.5  docker.io/calico/node:v3.23.5

ctr images pull swr.cn-north-4.myhuaweicloud.com/ddn-
k8s/docker.io/calico/kube-controllers:v3.23.5
ctr images tag  swr.cn-north-4.myhuaweicloud.com/ddn-
k8s/docker.io/calico/kube-controllers:v3.23.5  docker.io/calico/kube-
controllers:v3.23.5

ctr images pull swr.cn-north-4.myhuaweicloud.com/ddn-
k8s/docker.io/calico/cni:v3.23.5
ctr images tag  swr.cn-north-4.myhuaweicloud.com/ddn-
k8s/docker.io/calico/cni:v3.23.5  docker.io/calico/cni:v3.23.5



## 结果
[root@master /opt]$ kubectl get pods -A -owide
NAMESPACE     NAME                                       READY   STATUS
RESTARTS   AGE     IP               NODE     NOMINATED NODE   READINESS
```

```
GATES
kube-system   calico-kube-controllers-7cb4fd5784-l48rh   1/1     Running   0
29m     172.16.196.131   node01   <none>           <none>
kube-system   calico-node-wnclq                         1/1     Running   0
29m     10.0.0.101       node01   <none>           <none>
kube-system   calico-node-xgv5b                         1/1     Running   0
62s     10.0.0.100       master   <none>           <none>
kube-system   coredns-6554b8b87f-4kh64                  1/1     Running   0
4h37m   172.16.196.129   node01   <none>           <none>
kube-system   coredns-6554b8b87f-dfsz7                  1/1     Running   0
4h37m   172.16.196.130   node01   <none>           <none>
kube-system   etcd-master                               1/1     Running   0
4h37m   10.0.0.100       master   <none>           <none>
kube-system   kube-apiserver-master                     1/1     Running   0
4h37m   10.0.0.100       master   <none>           <none>
kube-system   kube-controller-manager-master            1/1     Running   0
4h37m   10.0.0.100       master   <none>           <none>
kube-system   kube-proxy-jj9xj                          1/1     Running   0
36m     10.0.0.101       node01   <none>           <none>
kube-system   kube-proxy-vq7dz                          1/1     Running   0
4h37m   10.0.0.100       master   <none>           <none>
kube-system   kube-scheduler-master                     1/1     Running   0
4h37m   10.0.0.100       master   <none>           <none>
[root@master /opt]$
```

监控

Metrics Server

```
# 下载文件
wget https://gh.tryxd.cn/https://github.com/kubernetes-sigs/metrics-
server/releases/download/v0.7.2/components.yaml
# 提取images
[root@master /opt]$ grep "image:" components.yaml
        image: registry.k8s.io/metrics-server/metrics-server:v0.7.2

# 安装镜像
ctr images pull swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/metrics-
server/metrics-server:v0.7.2
ctr images tag  swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/metrics-
server/metrics-server:v0.7.2  registry.k8s.io/metrics-server/metrics-server:v0.7.2

# 执行命令
kubectl apply -f components.yaml
记得添加文件中的        - --kubelet-insecure-tls 跳过证书，不然不生效

    spec:
      containers:
      - args:
        - --cert-dir=/tmp
```

```
        - --secure-port=10250
        - --kubelet-insecure-tls
```