**参考资料**

参考官网[https://kubernetes.io/zh/docs/setup/production-environment/tools/kubeadm/install-kubeadm](https://kubernetes.io/zh/docs/setup/production-environment/tools/kubeadm/install-kubeadm/)

docker 源

国内

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

wget-O/etc/yum.repos.d/CentOS-Base.repohttp://mirrors.aliyun.com/repo/Centos-7.repo

官网参考

<https://docs.docker.com/engine/install/centos/#install-using-the-repository>

**二进制部署参考**

<https://blog.csdn.net/weixin_42072280/article/details/113405732>

k8s源

国内

vim kubernetes.repo  
[kubernetes]  
name=Kubernetes Repo  
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/  
gpgcheck=0  
enable=1

官网 （阿里云连不上）

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo  
[kubernetes]  
name=Kubernetes  
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch  
enabled=1  
gpgcheck=1  
repo\_gpgcheck=1  
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg  
exclude=kubelet kubeadm kubectl  
EOF

版本偏差

<https://kubernetes.io/zh/docs/setup/release/version-skew-policy/>

**安装过程**

参考<https://www.cnblogs.com/along21/p/10303495.html>

<https://www.cnblogs.com/along21/p/10304362.html>

**安装docker**

yum -y install docker-ce-18.06.1.ce-3.el7

#添加docker加速器到配置文件

mkdir -p /etc/docker

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

{

"registry-mirrors": ["https://registry.docker-cn.com"]

}

EOF

#启动docker服务

systemctl daemon-reload

systemctl start docker

systemctl enable docker.service

#打开iptables内生的桥接相关功能，已经默认开启了，没开启的自行开启

cat /proc/sys/net/bridge/bridge-nf-call-ip6tables

cat /proc/sys/net/bridge/bridge-nf-call-iptables

**安装kubeadm,kubectl kubelet**

#安装kubelet、kubeadm、kubectl

yum -y install kubeadm-1.17.0 kubelet-1.17.0 kubectl-1.17.0

#配置启动kubelet服务

vim /etc/sysconfig/kubelet

KUBELET\_EXTRA\_ARGS="--fail-swap-on=false"

KUBE\_PROXY=MODE=ipvs

#设为开机自启

systemctl enable kubelet.service

**初始化kubernetes master节点**

#使用kubeadm init 进行初始化（需要进行很多操作，所以要等待一段时间）

kubeadm init --apiserver-advertise-address=10.0.0.93 --kubernetes-version=v1.17.0 --image-repository registry.aliyuncs.com/google\_containers --service-cidr=10.96.0.0/12 --pod-network-cidr 10.244.0.0/16

会输出一个token，添加worker 节点时要用到

注释：

--kubernetes-version：指定kubeadm版本；

--pod-network-cidr：指定pod所属网络

--service-cidr：指定service网段

--ignore-preflight-errors=Swap/all：忽略 swap/所有 报错

备注：

#注意

#如果是root用户执行下面命令

#初始化命令成功后，创建.kube目录

mkdir -p $HOME/.kube

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

#验证

kubectl get cs

kubectl get node 此时node notready

**部署网络插件flannel**

#直接使用kubectl 执行gitlab上的flannel

sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

#或者下载到本地，再安装

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

kubectl apply -f kube-flannel.yml

#验证

kubectl get node 此时node ready

kube-flannel

---

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: psp.flannel.unprivileged

annotations:

seccomp.security.alpha.kubernetes.io/allowedProfileNames: docker/default

seccomp.security.alpha.kubernetes.io/defaultProfileName: docker/default

apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/default

apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default

spec:

privileged: false

volumes:

- configMap

- secret

- emptyDir

- hostPath

allowedHostPaths:

- pathPrefix: "/etc/cni/net.d"

- pathPrefix: "/etc/kube-flannel"

- pathPrefix: "/run/flannel"

readOnlyRootFilesystem: false

# Users and groups

runAsUser:

rule: RunAsAny

supplementalGroups:

rule: RunAsAny

fsGroup:

rule: RunAsAny

# Privilege Escalation

allowPrivilegeEscalation: false

defaultAllowPrivilegeEscalation: false

# Capabilities

allowedCapabilities: ['NET\_ADMIN', 'NET\_RAW']

defaultAddCapabilities: []

requiredDropCapabilities: []

# Host namespaces

hostPID: false

hostIPC: false

hostNetwork: true

hostPorts:

- min: 0

max: 65535

# SELinux

seLinux:

# SELinux is unused in CaaSP

rule: 'RunAsAny'

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: flannel

rules:

- apiGroups: ['extensions']

resources: ['podsecuritypolicies']

verbs: ['use']

resourceNames: ['psp.flannel.unprivileged']

- apiGroups:

- ""

resources:

- pods

verbs:

- get

- apiGroups:

- ""

resources:

- nodes

verbs:

- list

- watch

- apiGroups:

- ""

resources:

- nodes/status

verbs:

- patch

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: flannel

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: flannel

subjects:

- kind: ServiceAccount

name: flannel

namespace: kube-system

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: flannel

namespace: kube-system

---

kind: ConfigMap

apiVersion: v1

metadata:

name: kube-flannel-cfg

namespace: kube-system

labels:

tier: node

app: flannel

data:

cni-conf.json: |

{

"name": "cbr0",

"cniVersion": "0.3.1",

"plugins": [

{

"type": "flannel",

"delegate": {

"hairpinMode": true,

"isDefaultGateway": true

}

},

{

"type": "portmap",

"capabilities": {

"portMappings": true

}

}

]

}

net-conf.json: |

{

"Network": "10.244.0.0/16",

"Backend": {

"Type": "vxlan"

}

}

---

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: kube-flannel-ds

namespace: kube-system

labels:

tier: node

app: flannel

spec:

selector:

matchLabels:

app: flannel

template:

metadata:

labels:

tier: node

app: flannel

spec:

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: kubernetes.io/os

operator: In

values:

- linux

hostNetwork: true

priorityClassName: system-node-critical

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.14.0-rc1

command:

- cp

args:

- -f

- /etc/kube-flannel/cni-conf.json

- /etc/cni/net.d/10-flannel.conflist

volumeMounts:

- name: cni

mountPath: /etc/cni/net.d

- name: flannel-cfg

mountPath: /etc/kube-flannel/

containers:

- name: kube-flannel

image: quay.io/coreos/flannel:v0.14.0-rc1

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: false

capabilities:

add: ["NET\_ADMIN", "NET\_RAW"]

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run/flannel

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run/flannel

- name: cni

hostPath:

path: /etc/cni/net.d

- name: flannel-cfg

configMap:

name: kube-flannel-cfg

**添加worker 节点**

1. 在节点先执行之前的所有步骤，然后执行kubeadm reset
2. 执行master节点输出的token. 可以重新输出kubeadmtokencreate--print-join-command

kubeadm join 10.0.0.94:6443 --token xxx \

--discovery-token-ca-cert-hash xxx

**添加master 节点**

参考<https://kubernetes.io/zh/docs/setup/production-environment/tools/kubeadm/high-availability/>

原先节点需要重新reset 后重新执行init

apiserver 需要添加一个slb LOAD\_BALANCER\_DNS:LOAD\_BALANCER\_PORT

#master1 节点

kubeadm reset

kubeadm init --apiserver-advertise-address=10.0.0.94 --kubernetes-version=v1.17.0 --image-repository registry.aliyuncs.com/google\_containers --service-cidr=10.96.0.0/12 --pod-network-cidr 10.244.0.0/16 --control-plane-endpoint "LOAD\_BALANCER\_DNS:LOAD\_BALANCER\_PORT" --upload-certs

命令会输出添加worker的命令以及添加master 的命令

#master2 节点

#执行添加master的命令

kubeadm join LOAD\_BALANCER\_DNS:LOAD\_BALANCER\_PORT --token xxx --discovery-token-ca-cert-hash xxx --control-plane --certificate-key xxx