节点IP地址：192.168.30.141

节点主机名：k8s-master-01

证书存放目录：/etc/kubernetes/kubelet/ssl/

配置文件目录：/etc/kubernetes/kubelet/conf/

日志存放目录：/var/log/kubernetes/kubelet/

创建所需目录

mkdir -p /etc/kubernetes/kubelet/{ssl,conf} \

/var/log/kubernetes/kubelet/

创建配置文件

cat > /etc/kubernetes/kubelet/conf/kubelet.conf <<EOF

KUBELET\_OPTS="--v=2 --logtostderr=false \

--log-dir=/var/log/kubernetes/kubelet \

--network-plugin=cni --cni-bin-dir=/opt/cni/bin \

--kubeconfig=/etc/kubernetes/kubelet/conf/kubelet.kubeconfig \

--bootstrap-kubeconfig=/etc/kubernetes/kubelet/conf/bootstrap.kubeconfig \

--config=/etc/kubernetes/kubelet/conf/kubelet-config.yml \

--cert-dir=/etc/kubernetes/kubelet/ssl/ \

--pod-infra-container-image=registry.cn-hangzhou.aliyuncs.com/google\_containers/pause:3.2"

EOF

cat > /etc/kubernetes/kubelet/conf/kubelet-config.yml <<EOF

apiVersion: kubelet.config.k8s.io/v1beta1

kind: KubeletConfiguration

authentication:

anonymous:

enabled: false

webhook:

cacheTTL: 2m0s

enabled: true

x509:

clientCAFile: /etc/kubernetes/ssl/ca.pem

authorization:

mode: Webhook

webhook:

cacheAuthorizedTTL: 5m0s

cacheUnauthorizedTTL: 30s

cgroupDriver: systemd

clusterDNS:

- 10.96.0.10

clusterDomain: cluster.local

address: 0.0.0.0

port: 10250

readOnlyPort: 0

failSwapOn: false

evictionHard:

imagefs.available: 15%

memory.available: 100Mi

nodefs.available: 10%

nodefs.inodesFree: 5%

maxOpenFiles: 1000000

maxPods: 110

EOF

复制kubelet二进制文件到/usr/bin/目录

cp /usr/local/src/kubernetes/server/bin/kubelet /usr/bin/

获取token

TOKEN=$(awk -F',' '{print $1}' /etc/kubernetes/kube-apiserver/ssl/bootstrap-token.csv)

echo $TOKEN && awk -F',' '{print $1}' /etc/kubernetes/kube-apiserver/ssl/bootstrap-token.csv

生成bootstrap.kubeconfig配置文件（在master节点执行以下命令）

kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=https://192.168.30.141:6443 \

--kubeconfig=bootstrap.kubeconfig

kubectl config set-credentials "kubelet-bootstrap" \

--token=这里填写上面获取到的TOKEN \

--kubeconfig=bootstrap.kubeconfig

kubectl config set-context default \

--cluster=kubernetes \

--user="kubelet-bootstrap" \

--kubeconfig=bootstrap.kubeconfig

kubectl config use-context default --kubeconfig=bootstrap.kubeconfig

将bootstrap.kubeconfig复制到配置文件目录

cp bootstrap.kubeconfig /etc/kubernetes/kubelet/conf/

创建服务文件

cat > /usr/lib/systemd/system/kubelet.service <<EOF

[Unit]

Description=Kubernetes Kubelet

After=docker.service

[Service]

EnvironmentFile=-/etc/kubernetes/kubelet/conf/kubelet.conf

ExecStart=/usr/bin/kubelet \$KUBELET\_OPTS

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

启动服务并设置为开机自启动（docker启动成功后再启动kubelet）

systemctl daemon-reload

systemctl enable kubelet

systemctl start kubelet

systemctl status kubelet

授权apiserver访问kubelet

cat > /etc/kubernetes/apiserver-to-kubelet-rbac.yaml <<EOF

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

annotations:

rbac.authorization.kubernetes.io/autoupdate: "true"

labels:

kubernetes.io/bootstrapping: rbac-defaults

name: system:kube-apiserver-to-kubelet

rules:

- apiGroups:

- ""

resources:

- nodes/proxy

- nodes/stats

- nodes/log

- nodes/spec

- nodes/metrics

- pods/log

verbs:

- "\*"

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: system:kube-apiserver

namespace: ""

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: system:kube-apiserver-to-kubelet

subjects:

- apiGroup: rbac.authorization.k8s.io

kind: User

name: kubernetes

EOF

kubectl apply -f /etc/kubernetes/apiserver-to-kubelet-rbac.yaml

查看kubelet证书请求

kubectl get csr

批准申请（<NAME>填node开头的那一大串内容）

kubectl certificate approve <NAME>

查看节点

kubectl get nodes