**Kubernetes 1.13.3 的安装部署文档**

**系统：centos7.4**

**系统参数：4核8G 磁盘：100G**

|  |  |  |
| --- | --- | --- |
| **主机名** | **Ip** | **角色** |
| **K8s-master** | **10.0.220.15** | **K8s主节点** |
| **K8s-node01** | **10.0.220.65** | **K8s从节点01** |
| **K8s-node02** | **10.0.220.111** | **K8s从节点02** |

* 操作系统：CentOS-7.4-64Bit
* Docker版本：1.18.3
* Kubernetes版本：1.13.3

**准备工作**

* 所有节点关闭防火墙

systemctl disable firewalld.service

systemctl stop firewalld.service

* 禁用SELINUX

setenforce 0

vi /etc/selinux/config

SELINUX=disabled

* 所有节点关闭 swap

swapoff -a

* 设置所有节点主机名

hostnamectl --static set-hostname k8s-master

hostnamectl --static set-hostname k8s-node01

hostnamectl --static set-hostname k8s-node02

* 所有节点 主机名/IP加入 hosts解析

编辑 /etc/hosts文件，加入以下内容：

**10.0.220.15** k8s-master

**10.0.220.65** k8s-node01

**10.0.220.111** k8s-node02

**各个节点秘钥通信**

**ssh-keygen**

**生成的秘钥让各个节点通信**

安装docker

**1.设置使用国内Yum源**

[root@linux-node1 ~]# cd /etc/yum.repos.d/

[root@linux-node1 yum.repos.d]# wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

**2.安装指定的Docker版本**

由于kubeadm对Docker的版本是有要求的，需要安装与kubeadm匹配的版本。

[root@linux-node1 ~]# yum list docker-ce.x86\_64 --showduplicates | sort -r

\* updates: mirrors.aliyun.com

Loading mirror speeds from cached hostfile

Loaded plugins: fastestmirror

\* extras: mirrors.aliyun.com

\* epel: mirrors.aliyun.com

docker-ce.x86\_64 3:18.09.0-3.el7 docker-ce-stable

docker-ce.x86\_64 18.06.1.ce-3.el7 docker-ce-stable

docker-ce.x86\_64 18.06.0.ce-3.el7 docker-ce-stable

docker-ce.x86\_64 18.03.1.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 18.03.0.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.12.1.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.12.0.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.09.1.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.09.0.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.06.2.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.06.1.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.06.0.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.03.3.ce-1.el7 docker-ce-stable

docker-ce.x86\_64 17.03.2.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.03.1.ce-1.el7.centos docker-ce-stable

docker-ce.x86\_64 17.03.0.ce-1.el7.centos docker-ce-stable

\* base: mirrors.aliyun.com

Available Packages

安装Docker18.06版本

[root@k8s-master ~]# yum -y install docker-ce-18.06.1.ce-3.el7

**3.启动后台进程**

[root@k8s-master ~]# systemctl enable docker && systemctl start docker

查看Docker版本

[root@k8s-master ~]# docker --version

**.设置kubernetes YUM仓库**

[root@k8s-master ~]# vim /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

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

enabled=1

gpgcheck=1

repo\_gpgcheck=1

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

**5.安装软件包**

由于版本更新频繁，请指定对应的版本号，本文采用1.13.3版本，其它版本未经测试。

[root@k8s-master ~]# yum list --showduplicates | grep 'kubeadm\|kubectl\|kubelet'

yum install -y kubelet-1.13.3 kubeadm-1.13.3 kubectl-1.13.3 kubernetes-cni-0.6.0-0

本地拉取镜像

docker pull mirrorgooglecontainers/kube-apiserver:v1.13.3

docker pull mirrorgooglecontainers/kube-controller-manager:v1.13.3

docker pull mirrorgooglecontainers/kube-scheduler:v1.13.3

docker pull mirrorgooglecontainers/kube-proxy:v1.13.3

docker pull mirrorgooglecontainers/pause:3.1

docker pull mirrorgooglecontainers/etcd:3.2.24

docker pull coredns/coredns:1.2.6

docker pull registry.cn-shenzhen.aliyuncs.com/cp\_m/flannel:v0.10.0-amd64

docker tag mirrorgooglecontainers/kube-apiserver:v1.13.3 k8s.gcr.io/kube-apiserver:v1.13.3

docker tag mirrorgooglecontainers/kube-controller-manager:v1.13.3 k8s.gcr.io/kube-controller-manager:v1.13.3

docker tag mirrorgooglecontainers/kube-scheduler:v1.13.3 k8s.gcr.io/kube-scheduler:v1.13.3

docker tag mirrorgooglecontainers/kube-proxy:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3

docker tag mirrorgooglecontainers/pause:3.1 k8s.gcr.io/pause:3.1

docker tag mirrorgooglecontainers/etcd:3.2.24 k8s.gcr.io/etcd:3.2.24

docker tag coredns/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6

docker tag registry.cn-shenzhen.aliyuncs.com/cp\_m/flannel:v0.10.0-amd64 quay.io/coreos/flannel:v0.10.0-amd64

docker rmi mirrorgooglecontainers/kube-apiserver:v1.13.3

docker rmi mirrorgooglecontainers/kube-controller-manager:v1.13.3

docker rmi mirrorgooglecontainers/kube-scheduler:v1.13.3

docker rmi mirrorgooglecontainers/kube-proxy:v1.13.3

docker rmi mirrorgooglecontainers/pause:3.1

docker rmi mirrorgooglecontainers/etcd:3.2.24

docker rmi coredns/coredns:1.2.6

docker rmi registry.cn-shenzhen.aliyuncs.com/cp\_m/flannel:v0.10.0-amd64

**设置内核参数**

[root@k8s-master ~]# cat <<EOF > /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

EOF

使配置生效

[root@k8s-master ~]# sysctl –system

**启动kubelet并设置开机启动**

注意，此时kubelet是无法正常启动的，可以查看/var/log/messages有报错信息，等待执行初始化之后即可正常，为正常现象。

[root@k8s-master ~]# systemctl enable kubelet && systemctl start kubelet

*以上步骤请在Kubernetes的所有节点上执行，本实验环境是需要在k8s-master、k8s-node01、k8s-node02这三台机器上均安装Docker和kubeadm*

### 初始化集群部署Master

在所有节点上安装完毕后，在linux-node1这台Master节点上进行集群的初始化工作。

**1.执行初始化操作**

kubeadm init --kubernetes-version=v1.13.3 --apiserver-advertise-address 10.0.220.15 --pod-network-cidr=10.244.0.0/16

先忽略报错，我们来看一下，初始化选项的意义：

* --apiserver-advertise-address：指定用 Master 的哪个IP地址与 Cluster的其他节点通信。
* --service-cidr：指定Service网络的范围，即负载均衡VIP使用的IP地址段。
* --pod-network-cidr：指定Pod网络的范围，即Pod的IP地址段。
* --image-repository：Kubenetes默认Registries地址是 k8s.gcr.io，在国内并不能访问 gcr.io，在1.13版本中我们可以增加-image-repository参数，默认值是 k8s.gcr.io，将其指定为阿里云镜像地址：registry.aliyuncs.com/google\_containers。
* --kubernetes-version=v1.13.3：指定要安装的版本号。
* --ignore-preflight-errors=：忽略运行时的错误，例如上面目前存在[ERROR NumCPU]和[ERROR Swap]，忽略这两个报错就是增加--ignore-preflight-errors=NumCPU 和--ignore-preflight-errors=Swap的配置即可。

**初始化的结果**

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

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

sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node

as root:

kubeadm join 10.0.220.15:6443 --token i8nxlt.ox0bzax19jak1tyq --discovery-token-ca-cert-hash sha256:02e8fd59a30c53e792f5f822409762bfab5aef329fd24c48f994a20f752c5738

**node节点**

kubeadm join 10.0.220.15:6443 --token i8nxlt.ox0bzax19jak1tyq --discovery-token-ca-cert-hash sha256:02e8fd59a30c53e792f5f822409762bfab5aef329fd24c48f994a20f752c5738

## 配置 kubectl

在 Master上用 root用户执行下列命令来配置 kubectl：

echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> /etc/profile

source /etc/profile

echo $KUBECONFIG

**安装Pod网络**

安装 Pod网络是 Pod之间进行通信的必要条件，k8s支持众多网络方案，这里我们依然选用经典的 flannel方案

* 首先设置系统参数：

sysctl net.bridge.bridge-nf-call-iptables=1

* 然后在 Master节点上执行如下命令：

kubectl apply -f kube-flannel.yaml

kube-flannel.yaml内容

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1beta1

metadata:

name: flannel

rules:

- apiGroups:

- ""

resources:

- pods

verbs:

- get

- apiGroups:

- ""

resources:

- nodes

verbs:

- list

- watch

- apiGroups:

- ""

resources:

- nodes/status

verbs:

- patch

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1beta1

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",

"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: extensions/v1beta1

kind: DaemonSet

metadata:

name: kube-flannel-ds-amd64

namespace: kube-system

labels:

tier: node

app: flannel

spec:

template:

metadata:

labels:

tier: node

app: flannel

spec:

hostNetwork: true

nodeSelector:

beta.kubernetes.io/arch: amd64

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.10.0-amd64

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:v0.10.0-amd64

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: true

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run

- name: cni

hostPath:

path: /etc/cni/net.d

- name: flannel-cfg

configMap:

name: kube-flannel-cfg

---

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: kube-flannel-ds-arm64

namespace: kube-system

labels:

tier: node

app: flannel

spec:

template:

metadata:

labels:

tier: node

app: flannel

spec:

hostNetwork: true

nodeSelector:

beta.kubernetes.io/arch: arm64

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.10.0-arm64

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:v0.10.0-arm64

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: true

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run

- name: cni

hostPath:

path: /etc/cni/net.d

- name: flannel-cfg

configMap:

name: kube-flannel-cfg

---

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: kube-flannel-ds-arm

namespace: kube-system

labels:

tier: node

app: flannel

spec:

template:

metadata:

labels:

tier: node

app: flannel

spec:

hostNetwork: true

nodeSelector:

beta.kubernetes.io/arch: arm

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.10.0-arm

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:v0.10.0-arm

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: true

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run

- name: cni

hostPath:

path: /etc/cni/net.d

- name: flannel-cfg

configMap:

name: kube-flannel-cfg

---

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: kube-flannel-ds-ppc64le

namespace: kube-system

labels:

tier: node

app: flannel

spec:

template:

metadata:

labels:

tier: node

app: flannel

spec:

hostNetwork: true

nodeSelector:

beta.kubernetes.io/arch: ppc64le

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.10.0-ppc64le

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:v0.10.0-ppc64le

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: true

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run

- name: cni

hostPath:

path: /etc/cni/net.d

- name: flannel-cfg

configMap:

name: kube-flannel-cfg

---

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: kube-flannel-ds-s390x

namespace: kube-system

labels:

tier: node

app: flannel

spec:

template:

metadata:

labels:

tier: node

app: flannel

spec:

hostNetwork: true

nodeSelector:

beta.kubernetes.io/arch: s390x

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.10.0-s390x

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:v0.10.0-s390x

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: true

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run

- name: cni

hostPath:

path: /etc/cni/net.d

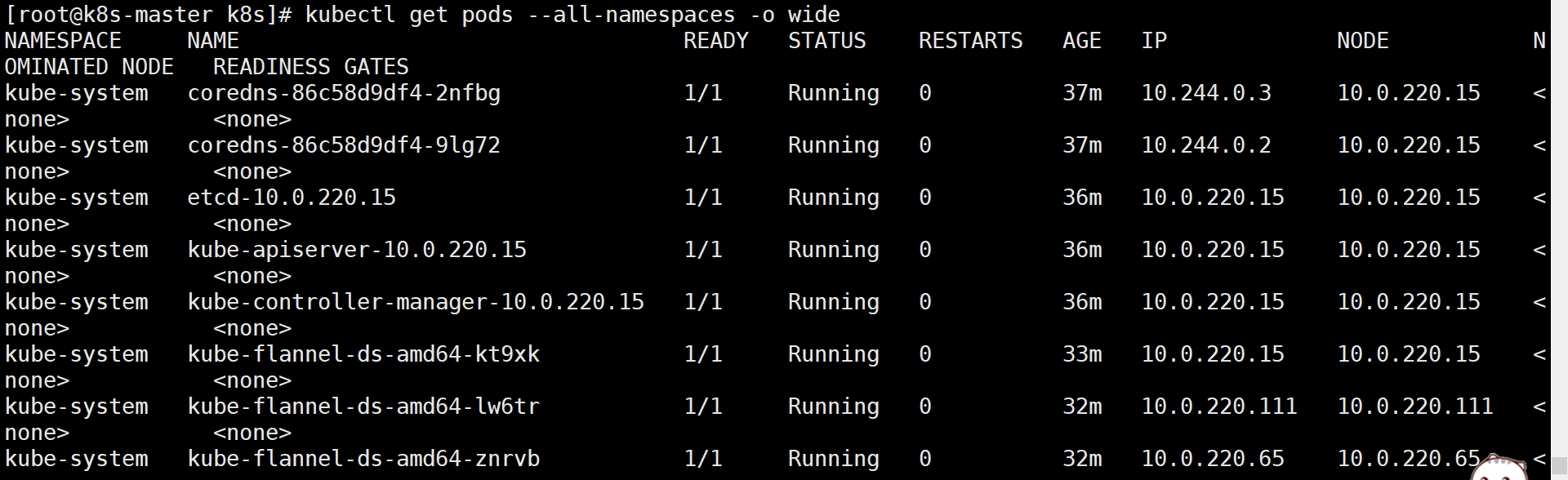
- name: flannel-cfg

configMap:

name: kube-flannel-cfg

一旦 Pod网络安装完成，可以执行如下命令检查一下 CoreDNS Pod此刻是否正常运行起来了，一旦其正常运行起来，则可以继续后续步骤

kubectl get pods --all-namespaces -o wide



同时我们可以看到主节点已经就绪：kubectl get nodes

