CentOS7下快速部署kubernetes1.15高可用集群-kubeadm篇

Gamemss

kubernetes官方给我提供了一个简易的集群部署工具Kubeadm，目的就是让我们能快速的部署一个可用的生产环境集群。今天我就和大家一起来体验一下。

一、环境准备（所有节点）

在开始学习安装K8S集群之前我们需要对环境进行一些调整，以便我们能快速部署。

版本

操作系统：CentOS7.6 64位

Kubernetes版本：v1.15.1

虚拟机

主机名	角色	配置	网络	备注
master1	管理节点	2核2G内存10G硬盘	桥接	必须2核及以上
node1	POD节点	1核1G内存10G硬盘	桥接	
node2	POD节点	1核1G内存10G硬盘	桥接	

同步时间

ntpdate time5.aliyun.com

//各节点时间必须正确，否者有可能产生证书认证过去不的情况

设置hosts

vim /etc/hosts

192.168.100.41 master

192.168.100.42 node1

192.168.100.43 node2

设置主机名

master节点

hostnamectl --static set-hostname master

node1节点

hostnamectl --static set-hostname node1

node2节点

hostnamectl --static set-hostname node2

su - root

//就能看到主机名已经生效了

关闭Selinux

vi /etc/sysconfig/selinux

SELINUX=disabled

setenforce 0

getenforce

//检查selinux是否关闭

关闭防火墙

systemctl stop firewalld

systemctl disable firewalld

iptables -L -n

//检查防火墙是否关闭

创建/etc/sysctl.d/k8s.conf，添加如下内容

vi /etc/sysctl.d/k8s.conf

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

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

net.ipv4.ip_forward = 1

vm.swappiness=0

modprobe br_netfilter

sysctl -p

sysctl --system

//验证内核参数是否生效

关闭swap

vi /etc/fstab

注释掉挂载 swap这行，保存推出，结果如下。

```
/dev/mapper/centos-root /                       ext4    defaults        1 1
UUID=09181933-4108-4adb-a21c-ae11e07feccf /boot                   ext4    defaults        1 2
#UUID=464c8c96-0c4f-47e6-bb0f-130785cb790d swap                    swap    defaults        0 0
```

swapoff -a

free -m

//验证swap是否关闭

安装一些系统工具

yum install yum-utils yum-fastestmirror ntp net-tools vim

使用阿里云的k8s仓库

cat <<EOF > /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

EOF

ls -al /etc/yum.repos.d/kubernetes.repo

//验证是否生产k8s的yum仓库源

二、docker安装（所有节点）

Docker分为了Docker-CE和Docker-EE两个版本，CE为社区版即免费版，EE为企业版即商业版。这里选择CE版。

安装docker仓库

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

ls -al /etc/yum.repos.d/docker-ce.repo
//验证是否生产docker的yum仓库源

yum makecache fast
//cache一下索引
```

查找我们需要的版本

```
yum list docker-ce.x86_64  --showduplicates |sort -r
```

安装docker社区版

```
yum install docker-ce-18.09.8-3.el7.x86_64 docker-ce-cli-18.09.8-3.el7.x86_64
//这里为了保证server和client版本一致，我们手动指定版本
//或者直接yum install docker-ce
```

启动docker

```
systemctl start docker
systemctl enable docker

docker version
//验证docker是否启动
```

k8s推荐docker的cgroup driver为systemd，默认是cgroupfs

创建或修改/etc/docker/daemon.json

```
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

重启

```
systemctl restart docker
```

检查Cgroup Driver是否已经修改为systemd

```
docker info | grep Cgroup
```

三、kubeadm安装（所有节点）

安装k8s组件

```
yum install kubelet kubeadm kubectl --disableexcludes=kubernetes
systemctl enable kubelet.service
```

kube-proxy配置ipvs模式，加载内核

```
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
```

安装ipvs管理工具

```
yum install ipvsadm
```

安装防火墙扩展工具

```
yum install ipset
```

添加开机加载ipvs脚本

```
vim /etc/sysconfig/modules/ipvs.modules
```

```
#!/bin/sh
/sbin/modinfo -F filename ip_vs_wrr > /dev/null 2>&1
if [ $? -eq 0 ]; then
    /sbin/modprobe -- ip_vs
    /sbin/modprobe -- ip_vs_rr
    /sbin/modprobe -- ip_vs_wrr
    /sbin/modprobe -- ip_vs_sh
    /sbin/modprobe -- nf_conntrack_ipv4
fi
```

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

```
lsmod|grep ip_vs
```

//检查内核是否加载ipvs参数

四、kubeadm配置（master节点操作）

打印集群初始化默认的使用的配置

cd /opt

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

修改项

advertiseAddress: 10.8.9.27

//改成自己的master地址

imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers

//改成阿里云docker镜像仓库

kubernetesVersion: v1.15.1

//你要的k8s的版本

serviceSubnet: 10.96.0.0/16

//service子网网段

podSubnet: 10.244.0.0/16

//pod子网网段，这个地址要和flannel中一样

验证仓库联通性以及检查镜像

kubeadm config images list --config kubeadm.yaml

```
[root@master opt]# kubeadm config images list --config kubeadm.yaml
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-apiserver:v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-controller-manager:v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler:v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-proxy:v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/pause:3.1
registry.cn-hangzhou.aliyuncs.com/google_containers/etcd:3.3.10
registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:1.3.1
```

拉取镜像

kubeadm config images pull --config kubeadm.yaml

//这个需要点时间，请耐心等待

docker images

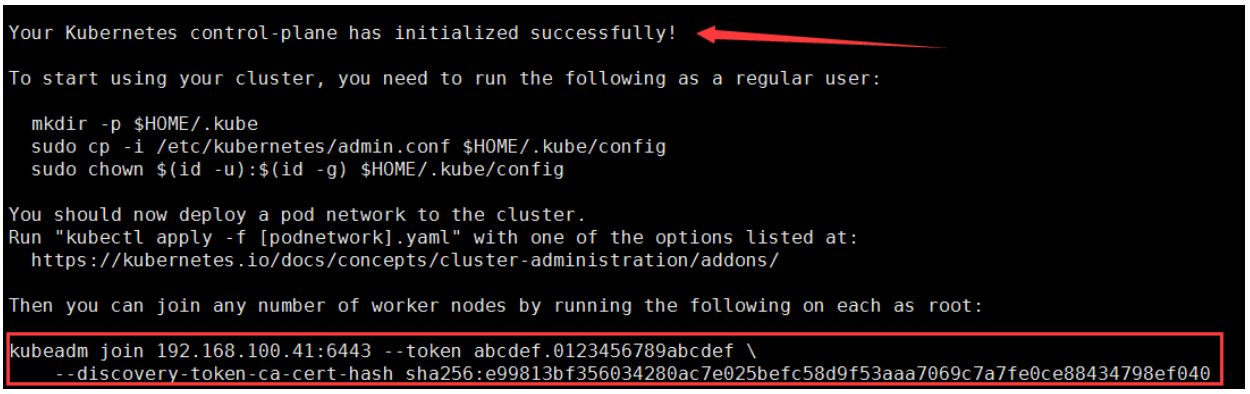
//检查镜像是否都拉取下来了

```
[root@master opt]# docker images
REPOSITORY                                                                    TAG
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-proxy                v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-apiserver            v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-controller-manager   v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler            v1.15.1
registry.cn-hangzhou.aliyuncs.com/google_containers/coredns                   1.3.1
registry.cn-hangzhou.aliyuncs.com/google_containers/etcd                      3.3.10
registry.cn-hangzhou.aliyuncs.com/google_containers/pause                     3.1
[root@master opt]#
```

初始化集群

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

--upload-certs 自动签发证书，方便我们加入更多的master节点



初始化成功后会出现success字样，如上。

记得保存红色边框的内容，因为加入pod节点的时候需要这段。

配置用户环境，方便各节点kubectl可以访问集群

mkdir -p $HOME/.kube

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

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

或者临时使用

export KUBECONFIG=/etc/kubernetes/admin.conf

查看集群状态（必须配置用户环境，否者提示8080被拒绝）

kubectl get cs

//查看服务状态

kubectl get nodes

//查看集群状态

kubectl get pod --all-namespaces -o wide

//查看所有pod状态

```
aster   NotReady   master   6m32s   v1.13.1
root@master opt]# kubectl get pod --all-namespaces -o wide
AMESPACE     NAME                              READY   STATUS    RESTARTS   AGE     IP              NODE     NOMINATED NODE   READINESS GATES
ube-system   coredns-6967fb4995-5br2g          0/1     Pending   0          7m9s    <none>          <none>   <none>           <none>
ube-system   coredns-6967fb4995-vqfdv          0/1     Pending   0          7m9s    <none>          <none>   <none>           <none>
ube-system   etcd-master                       1/1     Running   0          6m16s   192.168.100.41  master   <none>           <none>
ube-system   kube-apiserver-master             1/1     Running   0          6m32s   192.168.100.41  master   <none>           <none>
ube-system   kube-controller-manager-master    1/1     Running   0          6m30s   192.168.100.41  master   <none>           <none>
ube-system   kube-proxy-ddfn6                  1/1     Running   0          7m9s    192.168.100.41  master   <none>           <none>
ube-system   kube-scheduler-master             1/1     Running   0          6m7s    192.168.100.41  master   <none>           <none>
root@master opt]#
```

dnscore还没有建立成功，这是正常现象，因为k8s网络还没有部署。

五、网络配置（master节点操作）

这个时候应该可以看到除了coreDNS之外其他容器都已经启动。原因是还没有配置网络。现在我们来设置一个网络，这里选fannel

下载yaml文件

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

//这个文件里面是一个写好了的网络模板，包括rbac等模块，本篇主要是讲解部署，关于网络以后会单独开篇详细介绍。

注意这里，网络要和kubuadm.yaml中podSubnet: 10.244.0.0/16一致

```
  net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
        "Type": "vxlan"
      }
    }
```

配置flannel网络

kubectl apply -f kube-flannel.yml

```
[root@master opt]# kubectl apply -f kube-flannel.yml
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds-amd64 created
daemonset.apps/kube-flannel-ds-arm64 created
daemonset.apps/kube-flannel-ds-arm created
daemonset.apps/kube-flannel-ds-ppc64le created
daemonset.apps/kube-flannel-ds-s390x created
[root@master opt]#
```

稍等一会，执行在查看pod状态

kubectl get pod --all-namespaces -o wide

```
[root@master ~]# kubectl get pod --all-namespaces -o wide
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE   IP
kube-system   coredns-6967fb4995-5br2g          1/1     Running   0          28m   10.244.0.2
kube-system   coredns-6967fb4995-vqfdv          1/1     Running   0          28m   10.244.0.3
kube-system   etcd-master                       1/1     Running   1          27m   192.168.100.41
kube-system   kube-apiserver-master             1/1     Running   1          27m   192.168.100.41
kube-system   kube-controller-manager-master    1/1     Running   1          27m   192.168.100.41
kube-system   kube-flannel-ds-amd64-7sjgl       1/1     Running   0          8s    192.168.100.41
kube-system   kube-proxy-ddfn6                  1/1     Running   1          28m   192.168.100.41
kube-system   kube-scheduler-master             1/1     Running   1          27m   192.168.100.41
```

//如果遇到很久都running不起来的情况就删掉pod重新建立，命令如下

kubectl delete pod {pod名} -n kube-system

如果都running就说明正常了，master节点部署完成。

六、增加node节点（pod节点操作）

把master上生成的命令，在pod节点上执行即可加入集群。（初始化master章节红色边框的内容）

kubeadm join 192.168.100.41:6443 --token abcdef.0123456789abcdef \
--discovery-token-ca-cert-hash sha256:e99813bf356034280ac7e025befc58d9f53aaa7069c7a7fe0ce88434798ef040

```
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.

[root@node1 ~]#
```

出现红色变空即表示成功。

//因为pod节点需要pull3个镜像，因此需要过一会才能看到节点加入集群的成功信息。

//可通过docker image 来判断，当3个镜像都pull完成后，kubelet会自动配置集群信息。

kubectl get nodes

//查看集群状态

```
[root@master opt]# kubectl get nodes
NAME     STATUS   ROLES    AGE   VERSION
master   Ready    master   58m   v1.15.1
node1    Ready    <none>   20m   v1.15.1
```

完成。

创建token

默认token的有效期为24小时，当过期之后，该token就不可用了，以后加入节点需要新token

在master机器上

```
kubeadm token create
```

//master重新生成新的token

```
kubeadm token list
```

//查看token列表

```
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt|openssl rsa -pubin -outform der 2>/dev/null|openssl dgst -sha256 -hex|awk '{print $NF}'
```

//获取ca证书sha256编码hash值

pod节点机器

```
ntpdate time5.aliyun.com
kubeadm join --token {token} 10.8.9.27:6443 --discovery-token-ca-cert-hash sha256:{hash}
```

将上面命令的token和hash换成你生成的值，执行。

```
kubectl get nodes
```

//查看集群状态

```
kubectl get pod --all-namespaces -o wide
```

//查看所有pod状态

kube-proxy开启ipvs

新版的k8s支持ipvs代理模式，在性能方面会比iptables模式更好。

在master上执行

kubectl edit cm kube-proxy -n kube-system
修改 mode 为 ipvs 保存退出

kubectl get pod -n kube-system | grep kube-proxy | awk '{system("kubectl delete pod "$1" -n kube-system")}'
//重启kube-proxy

测试
kubectl --namespace kube-system logs { kube-proxy的pod名}

```
[root@master opt]# kubectl --namespace kube-system logs kube-proxy-5vpg6
I0804 04:47:45.588672       1 server_others.go:170] Using ipvs Proxier.
W0804 04:47:45.589061       1 proxier.go:401] IPVS scheduler not specified, use rr by default
I0804 04:47:45.589311       1 server.go:534] Version: v1.15.1
I0804 04:47:45.598280       1 conntrack.go:52] Setting nf_conntrack_max to 131072
I0804 04:47:45.602984       1 config.go:187] Starting service config controller
I0804 04:47:45.603014       1 controller_utils.go:1029] Waiting for caches to sync for service config controller
I0804 04:47:45.603095       1 config.go:96] Starting endpoints config controller
I0804 04:47:45.603113       1 controller_utils.go:1029] Waiting for caches to sync for endpoints config controller
I0804 04:47:45.703613       1 controller_utils.go:1036] Caches are synced for endpoints config controller
I0804 04:47:45.703698       1 controller_utils.go:1036] Caches are synced for service config controller
[root@master opt]#
```

出现红色部分表示成功

node2节点同上，最终集群如下。

```
[root@master opt]# kubectl get pod --all-namespaces -o wide
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE     IP               NODE
kube-system   coredns-6967fb4995-5br2g          1/1     Running   0          76m     10.244.0.2       master
kube-system   coredns-6967fb4995-vqfdv          1/1     Running   0          76m     10.244.0.3       master
kube-system   etcd-master                       1/1     Running   1          75m     192.168.100.41   master
kube-system   kube-apiserver-master             1/1     Running   1          76m     192.168.100.41   master
kube-system   kube-controller-manager-master    1/1     Running   1          76m     192.168.100.41   master
kube-system   kube-flannel-ds-amd64-7sjgl       1/1     Running   0          48m     192.168.100.41   master
kube-system   kube-flannel-ds-amd64-hxlrk       1/1     Running   3          6m26s   192.168.100.43   node2
kube-system   kube-flannel-ds-amd64-wxd9p       1/1     Running   0          39m     192.168.100.42   node1
kube-system   kube-proxy-5vpg6                  1/1     Running   0          13m     192.168.100.42   node1
kube-system   kube-proxy-nks8b                  1/1     Running   0          6m26s   192.168.100.43   node2
kube-system   kube-proxy-sst8t                  1/1     Running   0          13m     192.168.100.41   master
kube-system   kube-scheduler-master             1/1     Running   1          75m     192.168.100.41   master
```

七、集群测试

这里部署一个简单的web服务来测试集群，运行时暴露8000端口，同时访问/info路径会显示容器的主机名。
准备2个文件
vim deployment-web.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: web
```

```
  replicas: 4
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - image: lingtony/goweb
        name: web
        ports:
        - containerPort: 8000
```

vim svc-goweb.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: websvc
spec:
  selector:
    app: web
  ports:
  - name: default
    protocol: TCP
    port: 80
    targetPort: 8000
```

部署服务

kubectl apply -f deployment-web.yaml

kubectl apply -f service-web.yaml

查看pod及服务

kubectl get po -o wide

```
[root@master opt]# kubectl get pod -n default -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP           NODE
web-69dfd77d45-6z6sp   1/1     Running   0          48s   10.244.2.2   node2
web-69dfd77d45-bhrff   1/1     Running   0          48s   10.244.2.3   node2
web-69dfd77d45-ktqct   1/1     Running   0          48s   10.244.1.3   node1
web-69dfd77d45-rzwmn   1/1     Running   0          48s   10.244.1.2   node1
```

kubectl get svc

```
[root@master opt]# kubectl  get svc
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes    ClusterIP   10.96.0.1      <none>        443/TCP   79m
service-web   ClusterIP   10.96.0.124    <none>        80/TCP    9s
```

测试访问

curl http://10.96.0.124/info

```
[root@master opt]# curl http://10.96.0.124/info
Hostname: web-69dfd77d45-bhrff[root@master opt]# curl http://10.96.0.124/info
Hostname: web-69dfd77d45-6z6sp[root@master opt]# curl http://10.96.0.124/info
Hostname: web-69dfd77d45-ktqct[root@master opt]# curl http://10.96.0.124/info
Hostname: web-69dfd77d45-rzwmn[root@master opt]# curl http://10.96.0.124/info
Hostname: web-69dfd77d45-bhrff[root@master opt]#
```

每次访问都返回不同的主机。表示成功。

看一下lvs

```
Hostname: web-69dfd77d45-bhrff[root@master opt]# ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.96.0.1:443 rr
  -> 192.168.100.41:6443          Masq    1      0          0
TCP  10.96.0.10:53 rr
  -> 10.244.0.2:53                Masq    1      0          0
  -> 10.244.0.3:53                Masq    1      0          0
TCP  10.96.0.10:9153 rr
  -> 10.244.0.2:9153              Masq    1      0          0
  -> 10.244.0.3:9153              Masq    1      0          0
TCP  10.96.0.124:80 rr
  -> 10.244.1.2:8000              Masq    1      0          1
  -> 10.244.1.3:8000              Masq    1      0          1
  -> 10.244.2.2:8000              Masq    1      0          1
  -> 10.244.2.3:8000              Masq    1      0          2
UDP  10.96.0.10:53 rr
  -> 10.244.0.2:53                Masq    1      0          0
  -> 10.244.0.3:53                Masq    1      0          0
[root@master opt]#
```

发布service为外部提供访问

现在只能在集群内部访问这个web，我们想在本机浏览器上访问。我们才用NodePort的方式

修改service-web.yaml,添加 type: NodePort

```
apiVersion: v1
kind: Service
metadata:
  name: service-web
spec:
  selector:
    app: web
  ports:
  - name: default
    protocol: TCP
    port: 80
    targetPort: 8000
  type: NodePort
~
```

执行

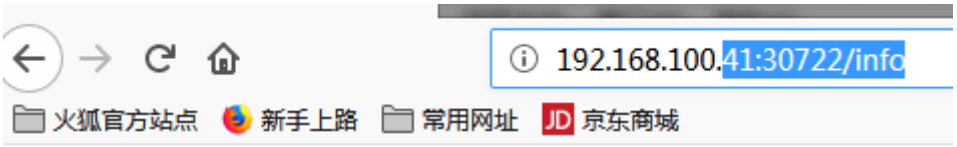
kubectl apply -f service-web.yaml

kubectl get svc

查看服务，红色部分就是我们暴露的端口号30722

```
[root@master opt]# kubectl  get svc
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.96.0.1      <none>        443/TCP        90m
service-web   NodePort    10.96.0.124    <none>        80:30722/TCP   10m
```

打开我们的浏览器 输入http://{任何一个节点ip}:{暴露的端口}



Hostname: web-69dfd77d45-6z6sp

小结

本文简单介绍了kubeadm如何部署一个高可用的kubernetes集群，相比二进制部署，方便快捷了许多，使我们能更快的进入K8S学习阶段，建议大家在学习阶段使用该种模式。由于kubeadm还在beta阶段，因此生产环境，该种方案还需谨慎考虑。