k8s集群部署

一、场景

通过上次课程的学习，播仔已经掌握了k8s集群架构，那么接下来播仔需要开始着手部署k8s集群，可是部署都有哪些要求或部署注意事项呢？播仔还是非常模糊的，例如：

- k8s集群部署工具有哪些？
- 需要几台物理服务器？
- 物理服务器硬件配置有哪些？
- 每台物理服务器部署什么组件？
- 部署结束后应该如何测试集群可用性？
- 等等

二、学习目标

- [ ] 了解k8s集群部署工具
- [ ] 掌握使用kubeadm部署k8s集群方法
- [ ] 掌握验证k8s集群可用性方法

三、学习步骤

序号	步骤	备注
1	k8s集群部署工具	
2	使用kubeadm部署k8s集群方法	
3	验证k8s集群可用性方法	

四、课程内容

4.1 k8s集群部署工具

4.1.1 二进制源码包部署

- 获取源码包
- 部署在各节点中
- 启动服务
  - Master
    - api-server
    - etcd
    - scheduler
    - controller manager
  - Worker

        - kubelet
        - kube-proxy
        - docker
- 生成证书
    - http
    - https

4.1.2 使用kubeadm部署

- 安装软件 kubelet kubeadm kubectl
- 初始化集群
- 添加node到集群中
- 证书自动生成
- 集群管理系统是以容器方式存在，容器运行在master
- 容器镜像是谷歌提供
    - 阿里云下载容器镜像，需要重新打标记
    - 谷歌下载

4.2 使用kubeadm部署kubernetes集群方法

> 使用kubeadm部署单Master节点k8s集群

4.2.1 主机要求

> 实际生产中，适当提高硬件配置

- 硬件

主机名	操作系统	CPU	MEM	角色要求
master1	CentOS7.6	2	2G	master
worker1	CentOS7.6	2	2G	worker
worker2	CentOS7.6	2	2G	worker

4.2.2 主机准备

> 所有主机均要配置

4.2.2.0 准备主机操作系统

主机操作系统	硬件配置	硬盘分区	IP
CentOS7u6 最小化	2C 2G 100G	/boot、/	192.168.216.100
CentOS7u6 最小化	2C 2G 100G	/boot、/	192.168.216.101
CentOS7u6 最小化	2C 2G 100G	/boot、/	192.168.216.102

4.2.2.1 主机名

```
[root@XXX ~]# hostnamectl set-hostname XXX
```

```
主机名列表
192.168.216.100 master1
192.168.216.101 worker1
192.168.216.102 worker2
```

4.2.2.2 主机IP地址

> IP地址段根据自己主机实际情况进行配置
>
> 本次使用VMWare Workstation Pro虚拟机部署，需要注意网关设置。

```
[root@XXX ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.216.XXX
NETMASK=255.255.255.0
GATEWAY=192.168.216.2
DNS1=119.29.29.29
```

4.2.2.3 主机名称解析

```
[root@XXX ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.216.100 master1
192.168.216.101 worker1
192.168.216.102 worker2
```

4.2.2.4 主机安全配置

4.2.2.4.1 关闭firewalld

```
[root@XXX ~]# systemctl stop firewalld
[root@XXX ~]# systemctl disable firewalld

确认是否运行
[root@XXX ~]# firewall-cmd --state
not running
```

4.2.2.4.2 SELINUX配置

> 做出下述修改，一定要重启系统才能生效。

```
[root@XXX ~]# sed -ri
's/SELINUX=enforcing/SELINUX=disabled/'
/etc/selinux/config
```

4.2.2.5 主机时间同步

> 由于最小化安装系统，需要单独安装ntpdate

```
[root@XXX ~]# yum -y install ntpdate
```

```
[root@XXX ~]# crontab -l
0 */1 * * * ntpdate time1.aliyun.com
```

4.2.2.6 永久关闭swap分区

> 使用kubeadm部署必须关闭swap分区，修改配置文件后需要重启操作系统。

```
[root@XXX ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Tue Mar  5 11:40:13 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root /                       xfs     defaults        0 0
```

```
UUID=2791cef9-e7dd-40cb-917c-52f8bc061339 /boot                   xfs     defaults        0 0
#/dev/mapper/centos-swap swap                    swap    defaults        0 0

在swap文件系统对应的行，行首添加#表示注释。
```

4.2.2.7 添加网桥过滤

```
添加网桥过滤及地址转发
[root@XXX ~]# cat /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

加载br_netfilter模块
[root@XXX ~]# modprobe br_netfilter

查看是否加载
[root@XXX ~]# lsmod | grep br_netfilter
br_netfilter           22256  0
bridge                151336  1 br_netfilter

加载网桥过滤配置文件
[root@XXX ~]# sysctl -p /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
```

4.2.2.8 开启ipvs

4.2.2.8.1 安装ipset及ipvsadm

```
[root@XXX ~]# yum -y install ipset ipvsadm
```

4.2.2.8.2 在所有节点执行如下脚本

```
添加需要加载的模块
[root@XXX ~]# 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
```

```
授权、运行、检查是否加载
[root@XXX ~]# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntrack_ipv4
```

```
检查是否加载
[root@XXX ~]# lsmod | grep -e ipvs -e nf_conntrack_ipv4
```

4.3 在manager节点及worker节点安装指定版本的docker-ce

4.3.1 YUM源获取

> 建议使用清华镜像源，官方提供的镜像源由于网络速度原因下载较慢

```
[root@XXX ~]# wget -O /etc/yum.repos.d/docker-ce.repo https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/centos/docker-ce.repo
```

4.3.2 查看docker-ce版本

> 对版本进行排序

```
[root@XXX ~]# yum list docker-ce.x86_64 --showduplicates | sort -r
```

4.3.3 安装指定版本docker-ce

> 安装指定版本docker-ce，此版本不需要修改服务启动文件及iptables默认规则链策略。

```
[root@XXX ~]# yum -y install --setopt=obsoletes=0 docker-ce-18.06.3.ce-3.el7
```

4.3.4 修改docker-ce服务配置文件

> 修改其目的是为了后续使用/etc/docker/daemon.json来进行更多配置。

```
修改内容如下
[root@XXX ~]# cat /usr/lib/systemd/system/docker.service
[Unit]
...

[Service]
...
ExecStart=/usr/bin/dockerd  #如果原文件此行后面有-H选项，请删除-H(含)后面所有内容。
```

```
...

[Install]
...


注意：有些版本不需要修改，请注意观察



#在/etc/docker/daemon.json添加如下内容：
[root@localhost ~]# cat
/etc/docker/daemon.json
{
        "exec-opts":
["native.cgroupdriver=systemd"]
}

```

4.4、部署软件及配置

4.4.1 软件安装

> 所有k8s集群节点均需安装，默认YUM源是谷歌，可以使用阿里云YUM

需求	kubeadm	kubelet	kubectl	docker-ce
值	初始化集群、管理集群等，版本为：1.17.2	用于接收api-server指令，对pod生命周期进行管理，版本为：1.17.2	集群命令行管理工具，版本为：1.17.2	18.06.3

- 谷歌YUM源

```
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
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
```

- 阿里云YUM源

```
[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
```

- 安装指定版本 kubeadm kubelet kubectl

```
[root@XXX ~]# yum list kubeadm.x86_64 --showduplicates | sort -r

[root@XXX ~]# yum -y install --setopt=obsoletes=0 kubeadm-1.17.2-0  kubelet-1.17.2-0 kubectl-1.17.2-0
```

4.4.2 软件设置

> 主要配置kubelet，如果不配置可能会导致k8s集群无法启动。

```
为了实现docker使用的cgroupdriver与kubelet使用的cgroup的一致性，建议修改如下文件内容。
[root@XXX ~]# vim /etc/sysconfig/kubelet
KUBELET_EXTRA_ARGS="--cgroup-driver=systemd"
```

```
设置为开机自启动即可，由于没有生成配置文件，集群初始化后自动启动
[root@XXX ~]# systemctl enable kubelet
```

4.5、k8s集群容器镜像准备

> 由于使用kubeadm部署集群，集群所有核心组件均以Pod运行，需要为主机准备镜像，不同角色主机准备不同镜像
>
> 建议使用科学上网方式下载镜像

4.5.1 Master主机镜像

```
查看集群使用的容器镜像
[root@master1 ~]# kubeadm config images list
k8s.gcr.io/kube-apiserver:v1.17.2
k8s.gcr.io/kube-controller-manager:v1.17.2
k8s.gcr.io/kube-scheduler:v1.17.2
k8s.gcr.io/kube-proxy:v1.17.2
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.4.3-0
k8s.gcr.io/coredns:1.6.5

列出镜像列表到文件中，便于下载使用。
[root@master1 ~]# kubeadm config images list >> image.list

查看已列出镜像文件列表
[root@master1 ~]# cat image.list

编写镜像下载脚本
[root@master1 ~]# cat image.pull
#!/bin/bash
img_list='
k8s.gcr.io/kube-apiserver:v1.17.2
k8s.gcr.io/kube-controller-manager:v1.17.2
k8s.gcr.io/kube-scheduler:v1.17.2
k8s.gcr.io/kube-proxy:v1.17.2
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.4.3-0
k8s.gcr.io/coredns:1.6.5
'

for img in $img_list
do
        docker pull $img
```

```
done


执镜像下载脚本
[root@master1 ~]# sh image.pull


查看已下载镜像

[root@master1 ~]# docker images
REPOSITORY                                 TAG
            IMAGE ID            CREATED
        SIZE
k8s.gcr.io/kube-proxy                      v1.17.2
            cba2a99699bd        2 weeks ago
        116MB
k8s.gcr.io/kube-apiserver                  v1.17.2
            41ef50a5f06a        2 weeks ago
        171MB
k8s.gcr.io/kube-controller-manager         v1.17.2
            da5fd66c4068        2 weeks ago
        161MB
k8s.gcr.io/kube-scheduler                  v1.17.2
            f52d4c527ef2        2 weeks ago
        94.4MB
k8s.gcr.io/etcd                            3.4.3-0
            303ce5db0e90        3 months ago
       288MB
k8s.gcr.io/coredns                         1.6.2
            bf261d157914        5 months ago
       44.1MB
```

```
k8s.gcr.io/pause                    3.1
              da86e6ba6ca1        2 years ago
          742kB
```

- Worker主机镜像

```
保存镜像为tar
[root@master1 ~]# docker save -o kube-proxy.tar  k8s.gcr.io/kube-proxy:v1.17.2
[root@master1 ~]# docker save -o pause.tar k8s.gcr.io/pause:3.1
[root@master1 ~]# ls
kube-proxy.tar  pause.tar

复制tar到worker节点
[root@master1 ~]# scp kube-proxy.tar pause.tar work1:/root

[root@master1 ~]# scp kube-proxy.tar pause.tar work2:/root


在worker节点导入镜像
[root@work1 ~]# ls
kube-proxy.tar  pause.tar
[root@work1 ~]# docker load -i kube-proxy.tar
[root@work1 ~]# docker load -i pause.tar


```

```
[root@work2 ~]# ls
kube-proxy.tar  pause.tar
[root@work2 ~]# docker load -i kube-proxy.tar
[root@work2 ~]# docker load -i pause.tar

```

4.6、k8s集群初始化

> 在master节点操作

```
[root@master1 ~]# kubeadm init --kubernetes-version=v1.17.2 --pod-network-cidr=172.16.0.0/16 --apiserver-advertise-address=192.168.216.100
```

```
初始化过程中导出结果
......
Your Kubernetes control-plane 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/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.216.100:6443 --token mOrpym.522tija0299geb8h \
    --discovery-token-ca-cert-hash sha256:cef7351d9fefc67868f22aa3122165dd01f63e95870d2fb22197ee66c61b18d6
```

4.6.1 准备集群管理文件

```
[root@master1 ~]# mkdir .kube

[root@master1 ~]# cp /etc/kubernetes/admin.conf ./.kube/config

[root@master1 ~]# chown $(id -u):$(id -g) .kube/config

```

4.6.2 网络插件使用

4.6.2.1 calico镜像准备

```
[root@master1 calico39]# ls
calico-cni.tar   calico.yml
 pod2daemon-flexvol.tar
calico-node.tar  kube-controllers.tar
[root@master1 calico39]# docker load -i calico-cni.tar

[root@master1 calico39]# docker load -i calico-node.tar

[root@master1 calico39]# docker load -i pod2daemon-flexvol.tar

```

```
[root@master1 calico39]# docker load -i kube-controllers.tar

[root@master1 calico39]# docker images
REPOSITORY                                 TAG                 IMAGE ID             CREATED             SIZE

calico/node                                v3.9.0              f9d62fb5edb1         7 weeks ago         190MB
calico/pod2daemon-flexvol                  v3.9.0              aa79ce3237eb         7 weeks ago         9.78MB
calico/cni                                 v3.9.0              56c7969ed8e6         7 weeks ago         160MB
calico/kube-controllers                    v3.9.0              f5cc48269a09         7 weeks ago         50.4MB


```

4.6.2.2 修改calico资源清单文件

```
由于calico自身网络发现机制有问题，因为需要修改calico使用的物理网卡，添加607及608行
602                - name: CLUSTER_TYPE
603                  value: "k8s,bgp"
604                # Auto-detect the BGP IP address.
605                - name: IP
606                value: "autodetect"
607                - name: IP_AUTODETECTION_METHOD
608                  value: "interface=eth.*"


修改为初始化时设置的pod-network-cidr
619                - name: CALICO_IPV4POOL_CIDR
620                  value: "172.16.0.0/16"
```

4.6.2.3 应用calico资源清单文件

> 在应用caclico资源清单文件之前，需要把calico所有的镜像导入到node节点中。

```
[root@master1 calico39]# kubectl apply -f calico.yml
```

```
输出结果
```

```
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
```

```
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
```

4.6.3 添加工作节点到集群

```
[root@work1 ~]# kubeadm join 192.168.122.100:6443 --token mOrpym.522tija0299geb8h \
>     --discovery-token-ca-cert-hash sha256:cef7351d9fefc67868f22aa3122165dd01f63e95870d2fb22197ee66c61b18d6
......

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

```

```
[root@work2 ~]# kubeadm join 192.168.122.100:6443 --token mOrpym.522tija0299geb8h \
>     --discovery-token-ca-cert-hash sha256:cef7351d9fefc67868f22aa3122165dd01f63e95870d2fb22197ee66c61b18d6
......

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

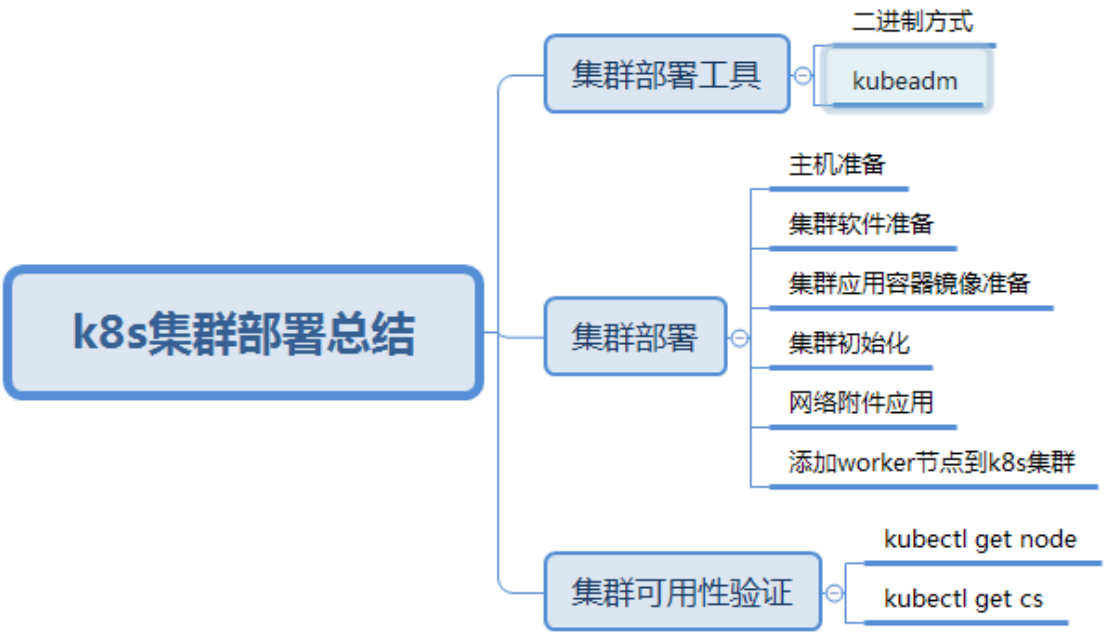
4.7 验证k8s集群可用性方法

```
查看节点状态
[root@master1 ~]# kubectl get nodes


查看集群健康状态
[root@master1 ~]# kubectl get cs
或
[root@master1 ~]# kubectl cluster-info
Kubernetes master is running at https://192.168.122.11:6443
KubeDNS is running at https://192.168.122.11:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

五、学习总结



六、课程预约

关于kubernetes高可用模式部署、生产环境部署实战等，可预约《kubernetes从入门到企业应用实战》相关课程。