

# 13. 操作命令 - 容器化数据中心建设

## 1. 服务器准备

### 1.1 服务器 Labs-K8s-Master-1：系统初始化

```
1 #关闭防火墙和SELINUX
2 systemctl disable --now firewalld
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #关闭交换分区
7 swapoff -a
8 sed -ri 's/.*swap.*/#&/' /etc/fstab
9
10 #设置主机名称，每台主机不同
11 hostnamectl set-hostname Labs-K8s-Master-1
12 reboot
13
14 #配置本地hosts
15 cat >> /etc/hosts << EOF
16 10.10.2.151 Labs-K8s-Master-1
17 10.10.2.152 Labs-K8s-Master-2
18 10.10.2.153 Labs-K8s-Master-3
19 10.10.2.154 Labs-K8s-Woker-1
20 EOF
21
22 #配置和加载 Linux 内核模块，启用容器网络过滤功能
23 cat >> /etc/modules-load.d/k8s.conf << EOF
24 overlay
25 br_netfilter
26 EOF
27
28 modprobe overlay
29 modprobe br_netfilter
30
31 #修改内核参数
32 cat >> /etc/sysctl.d/k8s.conf << EOF
33 net.bridge.bridge-nf-call-iptables = 1
34 net.bridge.bridge-nf-call-ip6tables = 1
35 net.ipv4.ip_forward = 1
36 EOF
37 sysctl -p /etc/sysctl.d/k8s.conf
38
39 #配置时间同步
40 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
41 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
42 systemctl enable chronyd
```

```
43 systemctl start chronyd
44
45 #配置支持 IPVS负载均衡功能
46 yum install -y ipset ipvsadm
47 cat >> /etc/sysconfig/modules/ipvs.modules << EOF
48 modprobe -- ip_vs
49 modprobe -- ip_vs_rr
50 modprobe -- ip_vs_wrr
51 modprobe -- ip_vs_sh
52 modprobe -- nf_conntrack
53 EOF
54
55 #加载并验证内核模块
56 chmod 755 /etc/sysconfig/modules/ipvs.modules
57 bash /etc/sysconfig/modules/ipvs.modules
58 lsmod | grep -e ip_vs -e nf_conntrack
```

## 1.2 服务器 Labs-K8s-Master-2: 系统初始化

```
1 #关闭防火墙和SELINUX
2 systemctl disable --now firewalld
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #关闭交换分区
7 swapoff -a
8 sed -ri 's/.*swap.*/#&/' /etc/fstab
9
10 #设置主机名称，每台主机不同
11 hostnamectl set-hostname Labs-K8s-Master-2
12 reboot
13
14 #配置本地hosts
15 cat >> /etc/hosts << EOF
16 10.10.2.151 Labs-K8s-Master-1
17 10.10.2.152 Labs-K8s-Master-2
18 10.10.2.153 Labs-K8s-Master-3
19 10.10.2.154 Labs-K8s-Woker-1
20 EOF
21
22 #配置和加载 Linux 内核模块，启用容器网络过滤功能
23 cat >> /etc/modules-load.d/k8s.conf << EOF
24 overlay
25 br_netfilter
26 EOF
27 modprobe overlay
28 modprobe br_netfilter
29
30 #修改内核参数
31 cat >> /etc/sysctl.d/k8s.conf << EOF
32 net.bridge.bridge-nf-call-iptables = 1
33 net.bridge.bridge-nf-call-ip6tables = 1
34 net.ipv4.ip_forward = 1
35 EOF
36 sysctl -p /etc/sysctl.d/k8s.conf
37
38 #配置时间同步
39 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
40 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
41 systemctl enable chronyd
42 systemctl start chronyd
```

```
43
44 #配置支持 IPVS负载均衡功能
45 yum install -y ipset ipvsadm
46 cat >> /etc/sysconfig/modules/ipvs.modules << EOF
47 modprobe -- ip_vs
48 modprobe -- ip_vs_rr
49 modprobe -- ip_vs_wrr
50 modprobe -- ip_vs_sh
51 modprobe -- nf_conntrack
52 EOF
53
54 #加载并验证内核模块
55 chmod 755 /etc/sysconfig/modules/ipvs.modules
56 bash /etc/sysconfig/modules/ipvs.modules
57 lsmod | grep -e ip_vs -e nf_conntrack
```

### 1.3 服务器 Labs-K8s-Master-3：系统初始化

```
1 #关闭防火墙和SELINUX
2 systemctl disable --now firewalld
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #关闭交换分区
7 swapoff -a
8 sed -ri 's/.*swap.*/#&/' /etc/fstab
9
10 #设置主机名称，每台主机不同
11 hostnamectl set-hostname Labs-K8s-Master-3
12 reboot
13
14 #配置本地hosts
15 cat >> /etc/hosts << EOF
16 10.10.2.151 Labs-K8s-Master-1
17 10.10.2.152 Labs-K8s-Master-2
18 10.10.2.153 Labs-K8s-Master-3
19 10.10.2.154 Labs-K8s-Woker-1
20 EOF
21
22 #配置和加载 Linux 内核模块，启用容器网络过滤功能
23 cat >> /etc/modules-load.d/k8s.conf << EOF
24 overlay
25 br_netfilter
26 EOF
27 modprobe overlay
28 modprobe br_netfilter
29
30 #修改内核参数
31 cat >> /etc/sysctl.d/k8s.conf << EOF
32 net.bridge.bridge-nf-call-iptables = 1
33 net.bridge.bridge-nf-call-ip6tables = 1
34 net.ipv4.ip_forward = 1
35 EOF
36 sysctl -p /etc/sysctl.d/k8s.conf
37
38 #配置时间同步
39 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
40 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
41 systemctl enable chronyd
42 systemctl start chronyd
```

```
43
44 #配置支持 IPVS负载均衡功能
45 yum install -y ipset ipvsadm
46 cat >> /etc/sysconfig/modules/ipvs.modules << EOF
47 modprobe -- ip_vs
48 modprobe -- ip_vs_rr
49 modprobe -- ip_vs_wrr
50 modprobe -- ip_vs_sh
51 modprobe -- nf_conntrack
52 EOF
53
54 #加载并验证内核模块
55 chmod 755 /etc/sysconfig/modules/ipvs.modules
56 bash /etc/sysconfig/modules/ipvs.modules
57 lsmod | grep -e ip_vs -e nf_conntrack
```

## 1.4 服务器 Labs-K8s-Woker-1: 系统初始化

```
1 #关闭防火墙和SELINUX
2 systemctl disable --now firewalld
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #关闭交换分区
7 swapoff -a
8 sed -ri 's/.*swap.*/#&/' /etc/fstab
9
10 #设置主机名称，每台主机不同
11 hostnamectl set-hostname Labs-K8s-Woker-1
12 reboot
13
14 #配置本地hosts
15 cat >> /etc/hosts << EOF
16 10.10.2.151 Labs-K8s-Master-1
17 10.10.2.152 Labs-K8s-Master-2
18 10.10.2.153 Labs-K8s-Master-3
19 10.10.2.154 Labs-K8s-Woker-1
20 EOF
21
22 #配置和加载 Linux 内核模块，启用容器网络过滤功能
23 cat >> /etc/modules-load.d/k8s.conf << EOF
24 overlay
25 br_netfilter
26 EOF
27 modprobe overlay
28 modprobe br_netfilter
29
30 #修改内核参数
31 cat >> /etc/sysctl.d/k8s.conf << EOF
32 net.bridge.bridge-nf-call-iptables = 1
33 net.bridge.bridge-nf-call-ip6tables = 1
34 net.ipv4.ip_forward = 1
35 EOF
36 sysctl -p /etc/sysctl.d/k8s.conf
37
38 #配置时间同步
39 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
40 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
41 systemctl enable chronyd
42 systemctl start chronyd
```

```
43
44 #配置支持 IPVS负载均衡功能
45 yum install -y ipset ipvsadm
46 cat >> /etc/sysconfig/modules/ipvs.modules << EOF
47 modprobe -- ip_vs
48 modprobe -- ip_vs_rr
49 modprobe -- ip_vs_wrr
50 modprobe -- ip_vs_sh
51 modprobe -- nf_conntrack
52 EOF
53
54 #加载并验证内核模块
55 chmod 755 /etc/sysconfig/modules/ipvs.modules
56 bash /etc/sysconfig/modules/ipvs.modules
57 lsmod | grep -e ip_vs -e nf_conntrack
```

## 2. Kubernetes 部署

### 2.1 服务器 Labs-K8s-Master-1: 安装基础软件

## Shell

```
1 #添加阿里云Docker镜像源, 并安装Docker
2 yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
3 sed -i 's/\$releasever/8/g' /etc/yum.repos.d/docker-ce.repo
4 yum install -y docker-ce docker-ce-cli containerd.io
5
6 #配置Docker镜像加速
7 cat >> /etc/docker/daemon.json <<EOF
8 {
9     "registry-mirrors": [
10         "https://registry.cn-hangzhou.aliyuncs.com",
11         "https://hub.xdark.top",
12         "https://hub.littlediary.cn",
13         "https://dockerpull.org",
14         "https://hub.crdz.gq",
15         "https://docker.1panel.live",
16         "https://docker.mirrors.ustc.edu.cn",
17         "https://docker.m.daocloud.io",
18         "https://noohub.ru",
19         "https://huecker.io",
20         "https://dockerhub.timeweb.cloud",
21         "https://docker.1panel.dev",
22         "https://docker.unsee.tech",
23         "https://docker.1panel.live"
24     ],
25     "exec-opts": ["native.cgroupdriver=systemd"],
26     "log-driver": "json-file",
27     "log-opts": {
28         "max-size": "100m"
29     }
30 }
31 EOF
32
33 #重新加载 systemd 配置, 启动docke并设置开机自起
34 systemctl daemon-reload
35 systemctl start docker
36 systemctl enable docker
37
38 #配置kubernetes源
39 cat >> /etc/yum.repos.d/kubernetes.repo << EOF
40 [kubernetes]
41 name=Kubernetes
```

```
42 baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-
    x86_64
43 enabled=1
44 gpgcheck=0
45 repo_gpgcheck=0
46 gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg http
    s://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
47 EOF
48
49 #安装Kubernetes组件
50 yum install -y kubelet-1.28.0 kubeadm-1.28.0 kubectl-1.28.0
51
52 #配置 kubelet 使用 systemd 作为 cgroup 驱动
53 sed -i '$a KUBELET_CGROUP_ARGS="--cgroup-driver=systemd"' /etc/sysconfi
    g/kubelet
54
55 #配置 kube-proxy 使用 ipvs 模式进行负载均衡。
56 sed -i '$a KUBE_PROXY_MODE="ipvs"' /etc/sysconfig/kubelet
57 systemctl enable kubelet
58
59 # 创建目录
60 mkdir -p /var/kubernetes
61
62 #获取并安装cri-dockerd
63 wget -P /var/kubernetes https://github.com/Mirantis/cri-dockerd/release
    s/download/v0.3.12/cri-dockerd-0.3.12-3.el7.x86_64.rpm
64 rpm -ivh /var/kubernetes/cri-dockerd-0.3.12-3.el7.x86_64.rpm
65
66 #使用指定的 pause 镜像作为 Pod 的基础容器镜像
67 sed -i '/ExecStart=\usr\bin\cri-dockerd/ s|$| --pod-infra-container-
    image=registry.aliyuncs.com/google_containers/pause:3.9|' /usr/lib/syst
    emd/systemd/cri-docker.service
68
69 # 加载配置并开启服务
70 systemctl daemon-reload
71 systemctl enable cri-docker && systemctl start cri-docker
72
73 #安装高可用和负载均衡工具
74 yum -y install keepalived haproxy
75
76 #备份haproxy配置文件
77 cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak
78
79 #重定向haproxy的配置文件haproxy.cfg
```

```
80 tee /etc/haproxy/haproxy.cfg <<EOF
81 global
82     maxconn 2000
83     ulimit-n 16384
84     log 127.0.0.1 local0 err
85     stats timeout 30s
86 #定义默认参数,
87 defaults
88     log global
89     mode http
90     option httplog
91     timeout connect 5000
92     timeout client 50000
93     timeout server 50000
94     timeout http-request 15s
95     timeout http-keep-alive 15s
96 #定义前端监控 haproxy 的状态
97 frontend monitor-in
98     bind *:33305
99     mode http
100    option httplog
101    monitor-uri /monitor
102 #定义前端, 负载均衡 Kubernetes 主节点的流量
103 frontend k8s-master
104     bind 0.0.0.0:16443
105     bind 127.0.0.1:16443
106     mode tcp
107     option tcplog
108     tcp-request inspect-delay 5s
109     default_backend k8s-master
110 #定义后端, 处理来自 k8s-master 前端的请求
111 backend k8s-master
112     mode tcp
113     option tcplog
114     option tcp-check
115     balance roundrobin
116     default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s max
        conn 250 maxqueue 256 weight 100
117     server Book-Cloud-K8s-Master-1      10.10.2.151:6443 check
118     server Book-Cloud-K8s-Master-2      10.10.2.152:6443 check
119     server Book-Cloud-K8s-Master-3      10.10.2.153:6443 check
120 EOF
121
122 #keepalived配置文件内容
```

```
123 cat >> /etc/keepalived/keepalived.conf << EOF
124 global_defs {
125     router_id LVS_DEVEL
126 }
127 vrrp_script check_apiserver {
128     script "/etc/keepalived/check_apiserver.sh"
129     interval 3
130     weight -2
131     fall 10
132     rise 2
133 }
134 vrrp_instance VI_1 {
135     state MASTER                #当前节点为 MASTER
136     interface ens33            #网络接口，根据环境选择接口
137     mcast_src_ip 10.10.2.151
138     virtual_router_id 51
139     priority 100               #优先级
140     authentication {
141         auth_type PASS
142         auth_pass K8SAUTH
143     }
144     virtual_ipaddress {
145         10.10.2.155            #配置虚拟 IP 地址 (VIP)
146     }
147     track_script {
148         check_apiserver
149     }
150 }
151 EOF
152
153 #创建健康检查脚本
154 cat >> /etc/keepalived/check_apiserver.sh << EOF
155 #!/bin/bash
156 API_SERVER="127.0.0.1:6443"
157 API_HEALTH_CHECK_URL="https://${API_SERVER}/healthz"
158 API_HEALTH_CHECK_TIMEOUT=5
159 API_HEALTH_CHECK_INTERVAL=3
160
161 if curl --silent --max-time ${API_HEALTH_CHECK_TIMEOUT} --insecure ${API_HEALTH_CHECK_URL} | grep -q "ok"; then
162     exit 0
163 else
164     exit 1
165 fi
```

```
166 EOF
167
168 chmod +x /etc/keepalived/check_apiserver.sh
169
170 systemctl daemon-reload
171 systemctl enable --now haproxy
172 systemctl enable --now keepalived
```

## 2.2 服务器 Labs-K8s-Master-2: 安装基础软件

## Shell

```
1 #添加阿里云Docker镜像源, 并安装Docker
2 yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
3 sed -i 's/\$releasever/8/g' /etc/yum.repos.d/docker-ce.repo
4 yum install -y docker-ce docker-ce-cli containerd.io
5
6 #配置Docker镜像加速
7 cat >> /etc/docker/daemon.json <<EOF
8 {
9     "registry-mirrors": [
10         "https://registry.cn-hangzhou.aliyuncs.com",
11         "https://hub.xdark.top",
12         "https://hub.littlediary.cn",
13         "https://dockerpull.org",
14         "https://hub.crdz.gq",
15         "https://docker.1panel.live",
16         "https://docker.mirrors.ustc.edu.cn",
17         "https://docker.m.daocloud.io",
18         "https://noohub.ru",
19         "https://huecker.io",
20         "https://dockerhub.timeweb.cloud",
21         "https://docker.1panel.dev",
22         "https://docker.unsee.tech",
23         "https://docker.1panel.live"
24     ],
25     "exec-opts": ["native.cgroupdriver=systemd"],
26     "log-driver": "json-file",
27     "log-opts": {
28         "max-size": "100m"
29     }
30 }
31 EOF
32
33 #重新加载 systemd 配置, 启动docke并设置开机自起
34 systemctl daemon-reload
35 systemctl start docker
36 systemctl enable docker
37
38 #配置kubernetes源
39 cat >> /etc/yum.repos.d/kubernetes.repo << EOF
40 [kubernetes]
41 name=Kubernetes
```

```
42 baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-
    x86_64
43 enabled=1
44 gpgcheck=0
45 repo_gpgcheck=0
46 gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg http
    s://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
47 EOF
48
49 #安装Kubernetes组件
50 yum install -y kubelet-1.28.0 kubeadm-1.28.0 kubectl-1.28.0
51
52 #配置 kubelet 使用 systemd 作为 cgroup 驱动
53 sed -i '$a KUBELET_CGROUP_ARGS="--cgroup-driver=systemd"' /etc/sysconfi
    g/kubelet
54
55 #配置 kube-proxy 使用 ipvs 模式进行负载均衡。
56 sed -i '$a KUBE_PROXY_MODE="ipvs"' /etc/sysconfig/kubelet
57 systemctl enable kubelet
58
59 # 创建目录
60 mkdir -p /var/kubernetes
61
62 #获取并安装cri-dockerd
63 wget -P /var/kubernetes https://github.com/Mirantis/cri-dockerd/release
    s/download/v0.3.12/cri-dockerd-0.3.12-3.el7.x86_64.rpm
64 rpm -ivh /var/kubernetes/cri-dockerd-0.3.12-3.el7.x86_64.rpm
65
66 #使用指定的 pause 镜像作为 Pod 的基础容器镜像
67 sed -i '/ExecStart=\usr\bin\cri-dockerd/ s|$| --pod-infra-container-
    image=registry.aliyuncs.com/google_containers/pause:3.9|' /usr/lib/syst
    emd/system/cri-docker.service
68
69 # 加载配置并开启服务
70 systemctl daemon-reload
71 systemctl enable cri-docker && systemctl start cri-docker
72
73 #安装高可用和负载均衡工具
74 yum -y install keepalived haproxy
75
76 #备份haproxy配置文件
77 cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak
78
79 #重定向haproxy的配置文件haproxy.cfg
```

```
80 tee /etc/haproxy/haproxy.cfg <<EOF
81 global
82     maxconn 2000
83     ulimit-n 16384
84     log 127.0.0.1 local0 err
85     stats timeout 30s
86 #定义默认参数,
87 defaults
88     log global
89     mode http
90     option httplog
91     timeout connect 5000
92     timeout client 50000
93     timeout server 50000
94     timeout http-request 15s
95     timeout http-keep-alive 15s
96 #定义前端监控 haproxy 的状态
97 frontend monitor-in
98     bind *:33305
99     mode http
100    option httplog
101    monitor-uri /monitor
102 #定义前端, 负载均衡 Kubernetes 主节点的流量
103 frontend k8s-master
104     bind 0.0.0.0:16443
105     bind 127.0.0.1:16443
106     mode tcp
107     option tcplog
108     tcp-request inspect-delay 5s
109     default_backend k8s-master
110 #定义后端, 处理来自 k8s-master 前端的请求
111 backend k8s-master
112     mode tcp
113     option tcplog
114     option tcp-check
115     balance roundrobin
116     default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s max
        conn 250 maxqueue 256 weight 100
117     server Book-Cloud-K8s-Master-1 10.10.2.151:6443 check
118     server Book-Cloud-K8s-Master-2 10.10.2.152:6443 check
119     server Book-Cloud-K8s-Master-3 10.10.2.153:6443 check
120 EOF
121
122 #keepalived配置文件内容
```

```
123 cat >> /etc/keepalived/keepalived.conf << EOF
124 global_defs {
125     router_id LVS_DEVEL
126 }
127 vrrp_script check_apiserver {
128     script "/etc/keepalived/check_apiserver.sh"
129     interval 3
130     weight -2
131     fall 10
132     rise 2
133 }
134 vrrp_instance VI_1 {
135     state BACKUP                #当前节点为 BACKUP
136     interface ens33            #网络接口，根据环境选择接口
137     mcast_src_ip 10.10.2.152
138     virtual_router_id 51
139     priority 50                #优先级
140     authentication {
141         auth_type PASS
142         auth_pass K8SAUTH
143     }
144     virtual_ipaddress {
145         10.10.2.155            #配置虚拟 IP 地址 (VIP)
146     }
147     track_script {
148         check_apiserver
149     }
150 }
151 EOF
152
153 #创建健康检查脚本
154 cat >> /etc/keepalived/check_apiserver.sh << EOF
155 #!/bin/bash
156 API_SERVER="127.0.0.1:6443"
157 API_HEALTH_CHECK_URL="https://${API_SERVER}/healthz"
158 API_HEALTH_CHECK_TIMEOUT=5
159 API_HEALTH_CHECK_INTERVAL=3
160
161 if curl --silent --max-time ${API_HEALTH_CHECK_TIMEOUT} --insecure ${API_HEALTH_CHECK_URL} | grep -q "ok"; then
162     exit 0
163 else
164     exit 1
165 fi
```

```
166 EOF
167
168 chmod +x /etc/keepalived/check_apiserver.sh
169
170 systemctl daemon-reload
171 systemctl enable --now haproxy
172 systemctl enable --now keepalived
```

## 2.4 服务器 Labs-K8s-Master-3: 安装基础软件

```
1 #添加阿里云Docker镜像源, 并安装Docker
2 yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
3 sed -i 's/\$releasever/8/g' /etc/yum.repos.d/docker-ce.repo
4 yum install -y docker-ce docker-ce-cli containerd.io
5
6 #配置Docker镜像加速
7 cat >> /etc/docker/daemon.json <<EOF
8 {
9     "registry-mirrors": [
10         "https://registry.cn-hangzhou.aliyuncs.com",
11         "https://hub.xdark.top",
12         "https://hub.littlediary.cn",
13         "https://dockerpull.org",
14         "https://hub.crdz.gq",
15         "https://docker.1panel.live",
16         "https://docker.mirrors.ustc.edu.cn",
17         "https://docker.m.daocloud.io",
18         "https://noohub.ru",
19         "https://huecker.io",
20         "https://dockerhub.timeweb.cloud",
21         "https://docker.1panel.dev",
22         "https://docker.unsee.tech",
23         "https://docker.1panel.live"
24     ],
25     "exec-opts": ["native.cgroupdriver=systemd"],
26     "log-driver": "json-file",
27     "log-opts": {
28         "max-size": "100m"
29     }
30 }
31 EOF
32
33 #重新加载 systemd 配置, 启动docke并设置开机自起
34 systemctl daemon-reload
35 systemctl start docker
36 systemctl enable docker
37
38 #配置kubernetes源
39 cat >> /etc/yum.repos.d/kubernetes.repo << EOF
40 [kubernetes]
41 name=Kubernetes
```

```
42 baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-
    x86_64
43 enabled=1
44 gpgcheck=0
45 repo_gpgcheck=0
46 gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg http
    s://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
47 EOF
48
49 #安装Kubernetes组件
50 yum install -y kubelet-1.28.0 kubeadm-1.28.0 kubectl-1.28.0
51
52 #配置 kubelet 使用 systemd 作为 cgroup 驱动
53 sed -i '$a KUBELET_CGROUP_ARGS="--cgroup-driver=systemd"' /etc/sysconfi
    g/kubelet
54
55 #配置 kube-proxy 使用 ipvs 模式进行负载均衡。
56 sed -i '$a KUBE_PROXY_MODE="ipvs"' /etc/sysconfig/kubelet
57 systemctl enable kubelet
58
59 # 创建目录
60 mkdir -p /var/kubernetes
61
62 #获取并安装cri-dockerd
63 wget -P /var/kubernetes https://github.com/Mirantis/cri-dockerd/release
    s/download/v0.3.12/cri-dockerd-0.3.12-3.el7.x86_64.rpm
64 rpm -ivh /var/kubernetes/cri-dockerd-0.3.12-3.el7.x86_64.rpm
65
66 #使用指定的 pause 镜像作为 Pod 的基础容器镜像
67 sed -i '/ExecStart=\usr\bin\cri-dockerd/ s|$| --pod-infra-container-
    image=registry.aliyuncs.com/google_containers/pause:3.9|' /usr/lib/syst
    emd/system/cri-docker.service
68
69 # 加载配置并开启服务
70 systemctl daemon-reload
71 systemctl enable cri-docker && systemctl start cri-docker
72
73 #安装高可用和负载均衡工具
74 yum -y install keepalived haproxy
75
76 #备份haproxy配置文件
77 cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak
78
79 #重定向haproxy的配置文件haproxy.cfg
```

```
80 tee /etc/haproxy/haproxy.cfg <<EOF
81 global
82     maxconn 2000
83     ulimit-n 16384
84     log 127.0.0.1 local0 err
85     stats timeout 30s
86 #定义默认参数,
87 defaults
88     log global
89     mode http
90     option httplog
91     timeout connect 5000
92     timeout client 50000
93     timeout server 50000
94     timeout http-request 15s
95     timeout http-keep-alive 15s
96 #定义前端监控 haproxy 的状态
97 frontend monitor-in
98     bind *:33305
99     mode http
100    option httplog
101    monitor-uri /monitor
102 #定义前端, 负载均衡 Kubernetes 主节点的流量
103 frontend k8s-master
104     bind 0.0.0.0:16443
105     bind 127.0.0.1:16443
106     mode tcp
107     option tcplog
108     tcp-request inspect-delay 5s
109     default_backend k8s-master
110 #定义后端, 处理来自 k8s-master 前端的请求
111 backend k8s-master
112     mode tcp
113     option tcplog
114     option tcp-check
115     balance roundrobin
116     default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s max
        conn 250 maxqueue 256 weight 100
117     server Book-Cloud-K8s-Master-1 10.10.2.151:6443 check
118     server Book-Cloud-K8s-Master-2 10.10.2.152:6443 check
119     server Book-Cloud-K8s-Master-3 10.10.2.153:6443 check
120 EOF
121
122 #keepalived配置文件内容
```

```
123 cat >> /etc/keepalived/keepalived.conf << EOF
124 global_defs {
125     router_id LVS_DEVEL
126 }
127 vrrp_script check_apiserver {
128     script "/etc/keepalived/check_apiserver.sh"
129     interval 3
130     weight -2
131     fall 10
132     rise 2
133 }
134 vrrp_instance VI_1 {
135     state BACKUP                #当前节点为 MASTER
136     interface ens33            #网络接口，根据环境选择接口
137     mcast_src_ip 10.10.2.153
138     virtual_router_id 51
139     priority 49                #优先级
140     authentication {
141         auth_type PASS
142         auth_pass K8SAUTH
143     }
144     virtual_ipaddress {
145         10.10.2.155            #配置虚拟 IP 地址 (VIP)
146     }
147     track_script {
148         check_apiserver
149     }
150 }
151 EOF
152
153 #创建健康检查脚本
154 cat >> /etc/keepalived/check_apiserver.sh << EOF
155 #!/bin/bash
156 API_SERVER="127.0.0.1:6443"
157 API_HEALTH_CHECK_URL="https://${API_SERVER}/healthz"
158 API_HEALTH_CHECK_TIMEOUT=5
159 API_HEALTH_CHECK_INTERVAL=3
160
161 if curl --silent --max-time ${API_HEALTH_CHECK_TIMEOUT} --insecure ${API_HEALTH_CHECK_URL} | grep -q "ok"; then
162     exit 0
163 else
164     exit 1
165 fi
```

```
166 EOF
167
168 chmod +x /etc/keepalived/check_apiserver.sh
169
170 systemctl daemon-reload
171 systemctl enable --now haproxy
172 systemctl enable --now keepalived
```

## 2.5 服务器 Labs-K8s-Woker-1: 安装基础软件

## Shell

```
1 #添加阿里云Docker镜像源, 并安装Docker
2 yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
3 sed -i 's/\$releasever/8/g' /etc/yum.repos.d/docker-ce.repo
4 yum install -y docker-ce docker-ce-cli containerd.io
5
6 #配置Docker镜像加速
7 cat >> /etc/docker/daemon.json <<EOF
8 {
9     "registry-mirrors": [
10         "https://registry.cn-hangzhou.aliyuncs.com",
11         "https://hub.xdark.top",
12         "https://hub.littlediary.cn",
13         "https://dockerpull.org",
14         "https://hub.crdz.gq",
15         "https://docker.1panel.live",
16         "https://docker.mirrors.ustc.edu.cn",
17         "https://docker.m.daocloud.io",
18         "https://noohub.ru",
19         "https://huecker.io",
20         "https://dockerhub.timeweb.cloud",
21         "https://docker.1panel.dev",
22         "https://docker.unsee.tech",
23         "https://docker.1panel.live"
24     ],
25     "exec-opts": ["native.cgroupdriver=systemd"],
26     "log-driver": "json-file",
27     "log-opts": {
28         "max-size": "100m"
29     }
30 }
31 EOF
32
33 #重新加载 systemd 配置, 启动docke并设置开机自起
34 systemctl daemon-reload
35 systemctl start docker
36 systemctl enable docker
37
38 #配置kubernetes源
39 cat >> /etc/yum.repos.d/kubernetes.repo << EOF
40 [kubernetes]
41 name=Kubernetes
```

```
42 baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-
    x86_64
43 enabled=1
44 gpgcheck=0
45 repo_gpgcheck=0
46 gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg http
    s://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
47 EOF
48
49 #安装Kubernetes组件
50 yum install -y kubelet-1.28.0 kubeadm-1.28.0 kubectl-1.28.0
51
52 #配置 kubelet 使用 systemd 作为 cgroup 驱动
53 sed -i '$a KUBELET_CGROUP_ARGS="--cgroup-driver=systemd"' /etc/sysconfi
    g/kubelet
54
55 #配置 kube-proxy 使用 ipvs 模式进行负载均衡。
56 sed -i '$a KUBE_PROXY_MODE="ipvs"' /etc/sysconfig/kubelet
57 systemctl enable kubelet
58
59 # 创建目录
60 mkdir -p /var/kubernetes
61
62 #获取并安装cri-dockerd
63 wget -P /var/kubernetes https://github.com/Mirantis/cri-dockerd/release
    s/download/v0.3.12/cri-dockerd-0.3.12-3.el7.x86_64.rpm
64 rpm -ivh /var/kubernetes/cri-dockerd-0.3.12-3.el7.x86_64.rpm
65
66 #使用指定的 pause 镜像作为 Pod 的基础容器镜像，在 “ExecStart=/usr/bin/cri-doc
    kerd --container-runtime-endpoint fd://" 这一行增加 “-pod-infra-container
    -image=registry.aliyuncs.com/google_containers/pause:3.9”
67 sed -i '/ExecStart=\usr\bin\cri-dockerd/ s|$| --pod-infra-container-
    image=registry.aliyuncs.com/google_containers/pause:3.9|' /usr/lib/syst
    emd/systemd/cri-docker.service
68
69 # 加载配置并开启服务
70 systemctl daemon-reload
71 systemctl enable cri-docker && systemctl start cri-docker
```

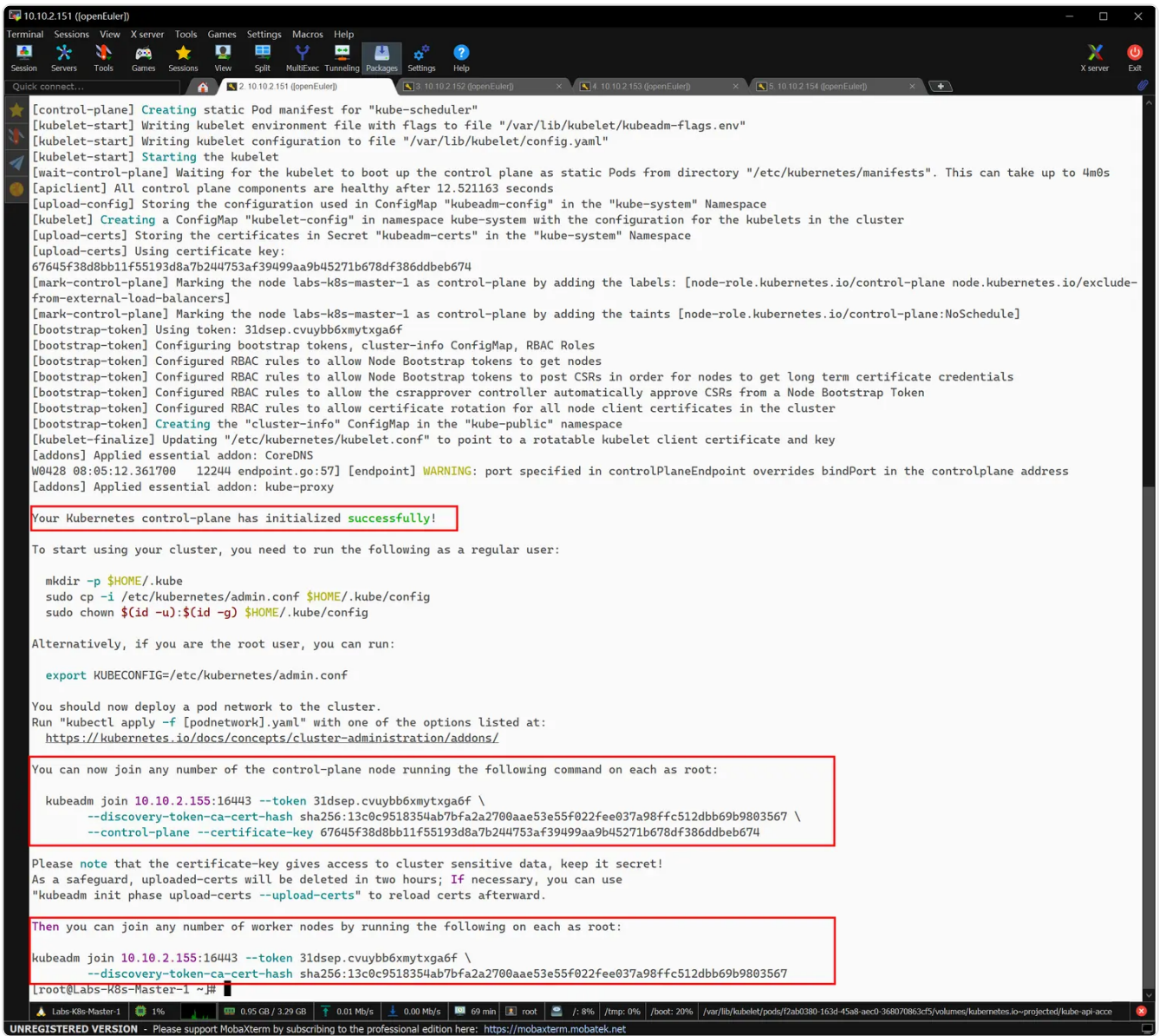
## 3. 初始化集群

### 3.1 服务器 Labs-K8s-Master-1：初始化集群

## Shell

```
1 kubeadm init \  
2 --apiserver-advertise-address=10.10.2.151 \  
3 --control-plane-endpoint "10.10.2.155:16443" \  
4 --upload-certs \  
5 --image-repository=registry.aliyuncs.com/google_containers \  
6 --kubernetes-version=v1.28.0 \  
7 --service-cidr=10.96.0.0/12 \  
8 --pod-network-cidr=10.244.0.0/16 \  
9 --cri-socket=unix:///var/run/cri-dockerd.sock
```

初始化成功提示，如图 1 所示。



```
[control-plane] Creating static Pod manifest for "kube-scheduler"  
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"  
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"  
[kubelet-start] Starting the kubelet  
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s  
[apiclient] All control plane components are healthy after 12.521163 seconds  
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace  
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster  
[upload-certs] Storing the certificates in Secret "kubeadm-certs" in the "kube-system" Namespace  
[upload-certs] Using certificate key:  
67645f38d8bb11f55193d8a7b244753af39499aa9b45271b678df386ddb6b674  
[mark-control-plane] Marking the node labs-k8s-master-1 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]  
[mark-control-plane] Marking the node labs-k8s-master-1 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]  
[bootstrap-token] Using token: 31dsep.cvuymb6xmytxga6f  
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles  
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes  
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials  
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token  
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster  
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace  
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key  
[addons] Applied essential addon: CoreDNS  
W0428 08:05:12.361700 12244 endpoint.go:57] [endpoint] WARNING: port specified in controlPlaneEndpoint overrides bindPort in the controlplane address  
[addons] Applied essential addon: kube-proxy  
  
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  
  
Alternatively, if you are the root user, you can run:  
  
export KUBECONFIG=/etc/kubernetes/admin.conf  
  
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 the control-plane node running the following command on each as root:  
  
kubeadm join 10.10.2.155:16443 --token 31dsep.cvuymb6xmytxga6f \  
--discovery-token-ca-cert-hash sha256:13c0c9518354ab7bfa2a2700aae53e55f022fee037a98ffc512dbb69b9803567 \  
--control-plane --certificate-key 67645f38d8bb11f55193d8a7b244753af39499aa9b45271b678df386ddb6b674  
  
Please note that the certificate-key gives access to cluster sensitive data, keep it secret!  
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use  
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.  
  
Then you can join any number of worker nodes by running the following on each as root:  
  
kubeadm join 10.10.2.155:16443 --token 31dsep.cvuymb6xmytxga6f \  
--discovery-token-ca-cert-hash sha256:13c0c9518354ab7bfa2a2700aae53e55f022fee037a98ffc512dbb69b9803567  
[root@labs-k8s-master-1 ~]#
```

图1

Shell

```
1 #在服务器Labs-K8s-Master-1上执行
2 mkdir -p $HOME/.kube
3 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
4 sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

## 3.2 服务器 Labs-K8s-Master-2：加入集群

Shell

```
1 #按实际参数调整，将节点加入控制平面节点
2 #最后加上--cri-socket=unix:///var/run/cri-dockerd.sock
3 #token、sha256的值需要根据Master-1创建的结果进行修改
4 kubeadm join 10.10.2.155:16443 --token u2f9c8.bw7mzbo5mdws17u3 \
5 --discovery-token-ca-cert-hash sha256:2865be67e64b690b65f8aff04c05600cd
   edf26ef6a42088ebe08fda372892188 \
6 --control-plane --certificate-key 078727d1573683c93356371a5392a6f6a4450
   7134e30237e178eeb0974b3a4d4 \
7 --cri-socket=unix:///var/run/cri-dockerd.sock
8
9 #加入后均需要执行
10 mkdir -p $HOME/.kube
11 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
12 sudo chown $(id -u):$(id -g) $HOME/.kube/config
13
14 #查看集群节点
15 kubectl get node
```

## 3.3 服务器 Labs-K8s-Master-3：加入集群

#### Plain Text

```
1 #按实际参数调整，将节点加入控制平面节点
2 #最后加上--cri-socket=unix:///var/run/cri-dockerd.sock
3 #token、sha256的值需要根据Master-1创建的结果进行修改
4 kubeadm join 10.10.2.155:16443 --token u2f9c8.bw7mzbo5mdws17u3 \
5 --discovery-token-ca-cert-hash sha256:2865be67e64b690b65f8aff04c05600cd
   edf26ef6a42088ebe08fda372892188 \
6 --control-plane --certificate-key 078727d1573683c93356371a5392a6f6a4450
   7134e30237e178eeb0974b3a4d4 \
7 --cri-socket=unix:///var/run/cri-dockerd.sock
8
9 #加入后均需要执行
10 mkdir -p $HOME/.kube
11 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
12 sudo chown $(id -u):$(id -g) $HOME/.kube/config
13
14 #查看集群节点
15 kubectl get node
```

### 3.4 服务器 Labs-K8s-Woker-1：加入集群

#### Shell

```
1 #按实际参数调整，将节点加入工作节点
2 #最后加上--cri-socket=unix:///var/run/cri-dockerd.sock
3 #token、sha256的值需要根据Master-1创建的结果进行修改
4 kubeadm join 10.10.2.155:16443 --token u2f9c8.bw7mzbo5mdws17u3 \
5 --discovery-token-ca-cert-hash sha256:2865be67e64b690b65f8aff04c05600cd
   edf26ef6a42088ebe08fda372892188 \
6 --cri-socket=unix:///var/run/cri-dockerd.sock
```

## 4. 部署网络插件

### 4.1 服务器 Labs-K8s-Master-1：部署网络插件

## Shell

```
1 wget -P /var/kubernetes https://docs.projectcalico.org/manifests/calico
  o.yaml
2 sed -i 's/^\(\\s*\)# \(- name: CALICO_IPV4POOL_CIDR\)\/\\1\\2\/' /var/kubern
  etes/calico.yaml
3 sed -i 's/^\(\\s*\)# \(\s*value: "192.168.0.0\/16"\)\/\\1\\2\/' /var/kuberne
  tes/calico.yaml
4 sed -i 's/\\(value: "192.168.0.0\/16"\)\/value: "10.244.0.0\/16"\/' /var/k
  ubernetes/calico.yaml
5
6 #部署
7 kubectl apply -f /var/kubernetes/calico.yaml
8
9 #查看部署情况，等待部署完成，需要几分钟
10 kubectl get pods -n kube-system | grep calico
11 kubectl get node
```

需要等待一段事件后，用于下载镜像和部署，如图2所示。

```
[root@Labs-K8s-Master-1 ~]# kubectl get pods -n kube-system | grep calico
calico-kube-controllers-658d97c59c-z4r8x      0/1      ContainerCreating    0                  4m41s
calico-node-g6xhs                             0/1      Init:2/3              0                  4m41s
calico-node-kk4lg                             0/1      Init:2/3              0                  4m41s
calico-node-pptpf                             0/1      Init:2/3              0                  4m41s
calico-node-wg5cg                             0/1      Init:2/3              0                  4m41s
[root@Labs-K8s-Master-1 ~]# kubectl get pods -n kube-system | grep calico
calico-kube-controllers-658d97c59c-z4r8x      1/1      Running               0                  7m5s
calico-node-g6xhs                             1/1      Running               0                  7m5s
calico-node-kk4lg                             0/1      Init:2/3              0                  7m5s
calico-node-pptpf                             0/1      Init:2/3              0                  7m5s
calico-node-wg5cg                             1/1      Running               0                  7m5s
[root@Labs-K8s-Master-1 ~]# kubectl get pods -n kube-system | grep calico
calico-kube-controllers-658d97c59c-z4r8x      1/1      Running              0                  11m
calico-node-g6xhs                             1/1      Running              0                  11m
calico-node-kk4lg                             1/1      Running              0                  11m
calico-node-pptpf                             1/1      Running              0                  11m
calico-node-wg5cg                             1/1      Running              0                  11m
[root@Labs-K8s-Master-1 ~]# kubectl get node
NAME                STATUS    ROLES    AGE   VERSION
labs-k8s-master-1   Ready    control-plane   29m   v1.28.0
labs-k8s-master-2   Ready    control-plane   22m   v1.28.0
labs-k8s-master-3   Ready    control-plane   16m   v1.28.0
labs-k8s-woker-1    Ready    <none>         15m   v1.28.0
[root@Labs-K8s-Master-1 ~]#
```

图2

## 5. 使用 NFS 实现共享存储

## 5.1 服务器 Labs-K8s-NFS：部署 NFS 共享存储服务

Shell

```
1 #修改主机名
2 hostnamectl set-hostname Labs-K8s-NFS
3 reboot
4
5 #安装nfs-utils rpcbind
6 yum install -y nfs-utils rpcbind
7 systemctl enable rpcbind --now
8 systemctl enable nfs-server --now
9 systemctl status rpcbind
10 systemctl status nfs-server
11
12 #配置firewalld防火墙规则
13 firewall-cmd --add-service=nfs --permanent
14 firewall-cmd --add-service=rpc-bind --permanent
15 firewall-cmd --add-service=mountd --permanent
16 firewall-cmd --reload
17 firewall-cmd --list-all
18
19 #格式化磁盘并挂载到共享目录
20 fdisk -l
21 mkfs.ext4 /dev/sdb
22 mkdir /K8s-NFS
23 chmod 777 /K8s-NFS
24 mount /dev/sdb /K8s-NFS
25 df -h
26 sed -i '$a /dev/sdb /K8s-NFS ext4 defaults 0 0' /etc/fstab
27
28 #允许客户端访问
29 cat >> /etc/exports <<EOF
30 /K8s-NFS 10.10.2.0/24(rw,sync,no_root_squash)
31 EOF
32
33 #重启服务和系统，验证NFS服务
34 systemctl restart nfs-server
35 reboot
36
37 #列出共享目录信息
38 showmount -e
```

## 5.2 服务器 Labs-K8s-Master-1：安装 NFS 客户端

Shell

```
1 #安装NFS客户端
2 yum install -y nfs-utils
```

## 5.3 服务器 Labs-K8s-Master-2：安装 NFS 客户端

Shell

```
1 #安装NFS客户端
2 yum install -y nfs-utils
```

## 5.4 服务器 Labs-K8s-Master-3：安装 NFS 客户端

Shell

```
1 #安装NFS客户端
2 yum install -y nfs-utils
```

## 5.5 服务器 Labs-K8s-Worker-1：安装 NFS 客户端

Shell

```
1 #安装NFS客户端
2 yum install -y nfs-utils
```

## 5.6 服务器 Labs-K8s-Master-1：创建 NFS 动态存储类

## Shell

```
1 wget -P /var/kubernetes https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner/archive/refs/tags/nfs-subdir-external-provisioner-4.0.18.tar.gz
2 tar -zxvf /var/kubernetes/nfs-subdir-external-provisioner-4.0.18.tar.gz -C /var/kubernetes/
3 cd /var/kubernetes/nfs-subdir-external-provisioner-nfs-subdir-external-provisioner-4.0.18/deploy/
4
5 #重定向deployment.yaml内容
6 tee deployment.yaml <<EOF
7 apiVersion: apps/v1
8 kind: Deployment
9 metadata:
10   name: nfs-client-provisioner
11   labels:
12     app: nfs-client-provisioner
13   # replace with namespace where provisioner is deployed
14   namespace: default
15 spec:
16   replicas: 1
17   strategy:
18     type: Recreate
19   selector:
20     matchLabels:
21       app: nfs-client-provisioner
22   template:
23     metadata:
24       labels:
25         app: nfs-client-provisioner
26     spec:
27       serviceAccountName: nfs-client-provisioner
28       containers:
29         - name: nfs-client-provisioner
30           # image: registry.k8s.io/sig-storage/nfs-subdir-external-provisioner:v4.0.2
31           # 更换国内镜像源
32           image: swr.cn-east-2.myhuaweicloud.com/kuboard-dependency/nfs-subdir-external-provisioner:v4.0.2
33           volumeMounts:
34             - name: nfs-client-root
35               mountPath: /persistentvolumes
36           env:
```

```

37         - name: PROVISIONER_NAME
38           value: k8s-sigs.io/nfs-subdir-external-provisioner
39         - name: NFS_SERVER
40           # value: 10.3.243.101
41           # 修改NFS_SERVER环境变量的值为10.10.2.159
42           value: 10.10.2.159
43         - name: NFS_PATH
44           # value: /ifs/kubernetes
45           # 修改NFS_PATH环境变量的值为/K8s-NFS
46           value: /K8s-NFS
47     volumes:
48       - name: nfs-client-root
49         nfs:
50           # server: 10.3.243.101
51           # 修改NFS服务器地址为10.10.2.159
52           server: 10.10.2.159
53           #path: /ifs/kubernetes
54           # 修改NFS服务器共享目录为/K8s-NFS
55           path: /K8s-NFS
56 EOF
57
58 #重定向class.yaml内容
59 tee class.yaml <<EOF
60 apiVersion: storage.k8s.io/v1
61 kind: StorageClass
62 metadata:
63   # name: nfs-client
64   # 修改动态存储供应的名称为k8s-nfs
65   name: k8s-nfs
66 provisioner: k8s-sigs.io/nfs-subdir-external-provisioner # or choose an
   other name, must match deployment's env PROVISIONER_NAME'
67 parameters:
68   archiveOnDelete: "false"
69 EOF

```

## 5.7 服务器 Labs-K8s-Master-1：部署 NFS 动态存储驱动

```
1 #在deploy目录下操作
2 kubectl create namespace nfs-provisioner
3 kubectl get namespace
4
5 #yaml资源清单文件中namespace的值为nfs-provisioner
6 sed -i 's/namespace: default/namespace: nfs-provisioner/g' `grep -rl 'namespace: default' ./`
7
8 #检查namespace的值是否修改成功
9 cat >> check-namespace.sh << 'EOF'
10 #!/bin/bash
11 yamls=$(grep -rl 'namespace:' ./*)
12 while IFS= read -r yaml; do
13     echo "=== 文件: $yaml ==="
14     grep 'namespace:' "$yaml"
15 done <<< "$yamls"
16 EOF
17
18 chmod +x check-namespace.sh
19 bash check-namespace.sh
20 kubectl apply -k .
21 kubectl get all -o wide -n nfs-provisioner
22 kubectl get storageclass
23
24 #返回个人目录
25 cd ~
```

## 6. 使用 Ceph 实现分布式存储

使用 SSH 远程管理终端软件，分别连接主机 10.10.2.156 (K8s-Ceph-1)、10.10.2.157 (K8s-Ceph-2)、10.10.2.158 (K8s-Ceph-3)。

### 6.1 服务器 Labs-K8s-Ceph-1: Ceph 存储集群准备

```
1 #关闭防火墙和SELINUX
2 systemctl disable firewalld --now
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #设置主机名
7 hostnamectl set-hostname Labs-K8s-Ceph-1
8 reboot
9
10 #配置本地hosts文件
11 cat << EOF >> /etc/hosts
12 10.10.2.157 Labs-K8s-Ceph-1
13 10.10.2.158 Labs-K8s-Ceph-2
14 10.10.2.159 Labs-K8s-Ceph-3
15 EOF
16
17 #配置时间同步
18 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
19 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
20 systemctl enable chronyd
21 systemctl restart chronyd
22
23 #安装podman、cephadm
24 yum install -y podman cephadm
25
26 # 添加Ceph源
27 cat >> /etc/yum.repos.d/ceph.repo <<EOF
28 [ceph]
29 name=ceph x86_64
30 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/x86_64
31 enabled=1
32 gpgcheck=0
33 [ceph-noarch]
34 name=ceph noarch
35 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/noarch
36 enabled=1
37 gpgcheck=0
38 [ceph-source]
39 name=ceph SRPMS
40 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/SRPMS
41 enabled=1
```

```
42 gpgcheck=0
```

```
43 EOF
```

## 6.2 服务器 Labs-K8s-Ceph-2: Ceph 存储集群准备

```
1 #关闭防火墙
2 systemctl disable firewalld --now
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #设置主机名
7 hostnamectl set-hostname Labs-K8s-Ceph-2
8 reboot
9
10 #配置本地hosts文件
11 cat << EOF >> /etc/hosts
12 10.10.2.157 Labs-K8s-Ceph-1
13 10.10.2.158 Labs-K8s-Ceph-2
14 10.10.2.159 Labs-K8s-Ceph-3
15 EOF
16
17 #配置时间同步
18 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
19 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
20 systemctl enable chronyd
21 systemctl restart chronyd
22
23 #安装podman、cephadm
24 yum install -y podman cephadm
25
26 # 添加Ceph源
27 cat >> /etc/yum.repos.d/ceph.repo <<EOF
28 [ceph]
29 name=ceph x86_64
30 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/x86_64
31 enabled=1
32 gpgcheck=0
33 [ceph-noarch]
34 name=ceph noarch
35 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/noarch
36 enabled=1
37 gpgcheck=0
38 [ceph-source]
39 name=ceph SRPMS
40 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/SRPMS
41 enabled=1
```

```
42 gpgcheck=0
```

```
43 EOF
```

## 6.3 服务器 Labs-K8s-Ceph-3: Ceph 存储集群准备

```
1 #配置防火墙
2 systemctl disable firewalld --now
3 setenforce 0
4 sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
5
6 #设置主机名
7 hostnamectl set-hostname Labs-K8s-Ceph-3
8 reboot
9
10 #配置本地hosts文件
11 cat << EOF >> /etc/hosts
12 10.10.2.157 Labs-K8s-Ceph-1
13 10.10.2.158 Labs-K8s-Ceph-2
14 10.10.2.159 Labs-K8s-Ceph-3
15 EOF
16
17 #配置时间同步
18 echo 'pool tiger.sina.com.cn iburst' >> /etc/chrony.conf
19 echo 'pool ntp1.aliyun.com iburst' >> /etc/chrony.conf
20 systemctl enable chronyd
21 systemctl restart chronyd
22
23 #安装podman、cephadm
24 yum install -y podman cephadm
25
26 # 添加Ceph源
27 cat >> /etc/yum.repos.d/ceph.repo <<EOF
28 [ceph]
29 name=ceph x86_64
30 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/x86_64
31 enabled=1
32 gpgcheck=0
33 [ceph-noarch]
34 name=ceph noarch
35 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/noarch
36 enabled=1
37 gpgcheck=0
38 [ceph-source]
39 name=ceph SRPMS
40 baseurl=https://repo.huaweicloud.com/ceph/rpm-pacific/el8/SRPMS
41 enabled=1
```

```
42 gpgcheck=0
43 EOF
```

## 6.4 服务器 Labs-K8s-Ceph-1：初始化 Ceph 集群

Shell

```
1 cephadm bootstrap --mon-ip 10.10.2.157 --allow-fqdn-hostname --initial-
  dashboard-user admin --initial-dashboard-password ceph@2025 --dashboard
  -password-noupdate
```

初始化完成提示 “Bootstrap complete.” 证明 Ceph 集群初始化完成，如图 3 所示。

```
Ceph Dashboard is now available at:

    URL: https://Labs-K8s-Ceph-1:8443/
    User: admin
    Password: ceph@2025

Enabling client.admin keyring and conf on hosts with "admin" label
Saving cluster configuration to /var/lib/ceph/b4ec2d20-254f-11f0-a96d-000c295da077/config directory
Enabling autotune for osd_memory_target
You can access the Ceph CLI as following in case of multi-cluster or non-default config:

    sudo /usr/sbin/cephadm shell --fsid b4ec2d20-254f-11f0-a96d-000c295da077 -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.keyring

Or, if you are only running a single cluster on this host:

    sudo /usr/sbin/cephadm shell

Please consider enabling telemetry to help improve Ceph:

    ceph telemetry on

For more information see:

    https://docs.ceph.com/en/latest/mgr/telemetry/

Bootstrap complete.
```

图3

使用浏览器访问 <https://10.10.2.157:8443> 进入 Ceph dashboard 登录界面。

账号：admin

密码：ceph@2025

## 6.5 服务器 Labs-K8s-Ceph-1：为集群添加 node 和 osd

Shell

```
1 #安装ceph-common
2 yum install -y ceph-common
3
4 #创建Ceph公钥并分发
5 ceph cephadm get-pub-key > ~/ceph.pub
6 ssh-copy-id -f -i ~/ceph.pub root@Labs-K8s-Ceph-2
7 ssh-copy-id -f -i ~/ceph.pub root@Labs-K8s-Ceph-3
8
9 #添加节点进入Ceph集群
10 ceph orch host add Labs-K8s-Ceph-2
11 ceph orch host add Labs-K8s-Ceph-3
12 ceph orch host ls
13
14 #为Ceph集群节点添加标签
15 ceph orch host label add Labs-K8s-Ceph-2 _admin
16 ceph orch host label add Labs-K8s-Ceph-3 _admin
17 ceph orch host ls
18
19 #手动为某一块硬盘添加OSD
20 ceph orch daemon add osd Labs-K8s-Ceph-1:/dev/sdb
21 # 为Ceph集群节点所有空闲硬盘添加OSD
22 ceph orch apply osd --all-available-devices
23 #查看OSD
24 ceph osd ls
```

登录 Ceph dashboard 控制台界面，可查看到集群状态为 “HEALTH\_OK”，主机数量为 3，OSD 数量为 3 且均为 up 状态。

## 7. 部署 Kubernetes 管理平台

### 7.1 Labs-K8s-Master-1：部署 Dashboard

创建并编辑 /var/kubernetes/dashboard.yaml 文件，内容如下。

## Shell

```
1 tee /var/kubernetes/dashboard.yaml << EOF
2 # Copyright 2017 The Kubernetes Authors.
3 #
4 # Licensed under the Apache License, Version 2.0 (the "License");
5 # you may not use this file except in compliance with the License.
6 # You may obtain a copy of the License at
7 #
8 #     http://www.apache.org/licenses/LICENSE-2.0
9 #
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the License is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 # See the License for the specific language governing permissions and
14 # limitations under the License.
15
16 apiVersion: v1
17 kind: Namespace
18 metadata:
19   name: kubernetes-dashboard
20
21 ---
22
23 apiVersion: v1
24 kind: ServiceAccount
25 metadata:
26   labels:
27     k8s-app: kubernetes-dashboard
28   name: kubernetes-dashboard
29   namespace: kubernetes-dashboard
30
31 ---
32
33 kind: Service
34 apiVersion: v1
35 metadata:
36   labels:
37     k8s-app: kubernetes-dashboard
38   name: kubernetes-dashboard
39   namespace: kubernetes-dashboard
40 spec:
41   type: NodePort
```

```
42   ports:
43     - port: 443
44       targetPort: 8443
45       nodePort: 30084
46
47   selector:
48     k8s-app: kubernetes-dashboard
49
50 ---
51
52 apiVersion: v1
53 kind: Secret
54 metadata:
55   labels:
56     k8s-app: kubernetes-dashboard
57   name: kubernetes-dashboard-certs
58   namespace: kubernetes-dashboard
59 type: Opaque
60
61 ---
62
63 apiVersion: v1
64 kind: Secret
65 metadata:
66   labels:
67     k8s-app: kubernetes-dashboard
68   name: kubernetes-dashboard-csrf
69   namespace: kubernetes-dashboard
70 type: Opaque
71 data:
72   csrf: ""
73
74 ---
75
76 apiVersion: v1
77 kind: Secret
78 metadata:
79   labels:
80     k8s-app: kubernetes-dashboard
81   name: kubernetes-dashboard-key-holder
82   namespace: kubernetes-dashboard
83 type: Opaque
84
85 ---
```

```
86
87 kind: ConfigMap
88 apiVersion: v1
89 metadata:
90   labels:
91     k8s-app: kubernetes-dashboard
92   name: kubernetes-dashboard-settings
93   namespace: kubernetes-dashboard
94
95 ---
96
97 kind: Role
98 apiVersion: rbac.authorization.k8s.io/v1
99 metadata:
100   labels:
101     k8s-app: kubernetes-dashboard
102   name: kubernetes-dashboard
103   namespace: kubernetes-dashboard
104 rules:
105   # Allow Dashboard to get, update and delete Dashboard exclusive secrets.
106   - apiGroups: [""]
107     resources: ["secrets"]
108     resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dashboard-certs", "kubernetes-dashboard-csrf"]
109     verbs: ["get", "update", "delete"]
110   # Allow Dashboard to get and update 'kubernetes-dashboard-settings' config map.
111   - apiGroups: [""]
112     resources: ["configmaps"]
113     resourceNames: ["kubernetes-dashboard-settings"]
114     verbs: ["get", "update"]
115   # Allow Dashboard to get metrics.
116   - apiGroups: [""]
117     resources: ["services"]
118     resourceNames: ["heapster", "dashboard-metrics-scraper"]
119     verbs: ["proxy"]
120   - apiGroups: [""]
121     resources: ["services/proxy"]
122     resourceNames: ["heapster", "http:heapster:", "https:heapster:", "dashboard-metrics-scraper", "http:dashboard-metrics-scraper"]
123     verbs: ["get"]
124
125 ---
```

```
126
127 kind: ClusterRole
128 apiVersion: rbac.authorization.k8s.io/v1
129 metadata:
130   labels:
131     k8s-app: kubernetes-dashboard
132   name: kubernetes-dashboard
133 rules:
134   # Allow Metrics Scraper to get metrics from the Metrics server
135   - apiGroups: ["metrics.k8s.io"]
136     resources: ["pods", "nodes"]
137     verbs: ["get", "list", "watch"]
138
139 ---
140
141 apiVersion: rbac.authorization.k8s.io/v1
142 kind: RoleBinding
143 metadata:
144   labels:
145     k8s-app: kubernetes-dashboard
146   name: kubernetes-dashboard
147   namespace: kubernetes-dashboard
148 roleRef:
149   apiGroup: rbac.authorization.k8s.io
150   kind: Role
151   name: kubernetes-dashboard
152 subjects:
153   - kind: ServiceAccount
154     name: kubernetes-dashboard
155     namespace: kubernetes-dashboard
156
157 ---
158
159 apiVersion: rbac.authorization.k8s.io/v1
160 kind: ClusterRoleBinding
161 metadata:
162   name: kubernetes-dashboard
163 roleRef:
164   apiGroup: rbac.authorization.k8s.io
165   kind: ClusterRole
166   name: kubernetes-dashboard
167 subjects:
168   - kind: ServiceAccount
169     name: kubernetes-dashboard
```

```
170     namespace: kubernetes-dashboard
171
172 ---
173
174 kind: Deployment
175 apiVersion: apps/v1
176 metadata:
177   labels:
178     k8s-app: kubernetes-dashboard
179   name: kubernetes-dashboard
180   namespace: kubernetes-dashboard
181 spec:
182   replicas: 1
183   revisionHistoryLimit: 10
184   selector:
185     matchLabels:
186       k8s-app: kubernetes-dashboard
187   template:
188     metadata:
189       labels:
190         k8s-app: kubernetes-dashboard
191     spec:
192       securityContext:
193         seccompProfile:
194           type: RuntimeDefault
195       containers:
196       - name: kubernetes-dashboard
197         image: kubernetesui/dashboard:v2.7.0
198         imagePullPolicy: Always
199         ports:
200         - containerPort: 8443
201           protocol: TCP
202         args:
203         - --auto-generate-certificates
204         - --namespace=kubernetes-dashboard
205         # Uncomment the following line to manually specify Kubernetes
206           # If not specified, Dashboard will attempt to auto discover
207           # to it. Uncomment only if the default does not work.
208         # - --apiserver-host=http://my-address:port
209       volumeMounts:
210       - name: kubernetes-dashboard-certs
211         mountPath: /certs
```

```
212         # Create on-disk volume to store exec logs
213         - mountPath: /tmp
214           name: tmp-volume
215       livenessProbe:
216         httpGet:
217           scheme: HTTPS
218           path: /
219           port: 8443
220           initialDelaySeconds: 30
221           timeoutSeconds: 30
222       securityContext:
223         allowPrivilegeEscalation: false
224         readOnlyRootFilesystem: true
225         runAsUser: 1001
226         runAsGroup: 2001
227       volumes:
228         - name: kubernetes-dashboard-certs
229           secret:
230             secretName: kubernetes-dashboard-certs
231         - name: tmp-volume
232           emptyDir: {}
233       serviceAccountName: kubernetes-dashboard
234       nodeSelector:
235         "kubernetes.io/os": linux
236       # Comment the following tolerations if Dashboard must not be depl
oyed on master
237       tolerations:
238         - key: node-role.kubernetes.io/master
239           effect: NoSchedule
240
241 ---
242
243 kind: Service
244 apiVersion: v1
245 metadata:
246   labels:
247     k8s-app: dashboard-metrics-scraper
248   name: dashboard-metrics-scraper
249   namespace: kubernetes-dashboard
250 spec:
251   ports:
252     - port: 8000
253       targetPort: 8000
254   selector:
```

```
255     k8s-app: dashboard-metrics-scraper
256
257 ---
258
259 kind: Deployment
260 apiVersion: apps/v1
261 metadata:
262   labels:
263     k8s-app: dashboard-metrics-scraper
264   name: dashboard-metrics-scraper
265   namespace: kubernetes-dashboard
266 spec:
267   replicas: 1
268   revisionHistoryLimit: 10
269   selector:
270     matchLabels:
271       k8s-app: dashboard-metrics-scraper
272   template:
273     metadata:
274       labels:
275         k8s-app: dashboard-metrics-scraper
276     spec:
277       securityContext:
278         seccompProfile:
279           type: RuntimeDefault
280       containers:
281       - name: dashboard-metrics-scraper
282         image: kubernetesui/metrics-scraper:v1.0.8
283         ports:
284         - containerPort: 8000
285           protocol: TCP
286         livenessProbe:
287           httpGet:
288             scheme: HTTP
289             path: /
290             port: 8000
291           initialDelaySeconds: 30
292           timeoutSeconds: 30
293         volumeMounts:
294         - mountPath: /tmp
295           name: tmp-volume
296         securityContext:
297           allowPrivilegeEscalation: false
298           readOnlyRootFilesystem: true
```

```
299         runAsUser: 1001
300         runAsGroup: 2001
301         serviceAccountName: kubernetes-dashboard
302         nodeSelector:
303             "kubernetes.io/os": linux
304         # Comment the following tolerations if Dashboard must not be deployed on master
305         tolerations:
306             - key: node-role.kubernetes.io/master
307               effect: NoSchedule
308         volumes:
309             - name: tmp-volume
310               emptyDir: {}
311 EOF
```

Shell

```
1 #部署dashboard
2 kubectl apply -f /var/kubernetes/dashboard.yaml
3 kubectl get pod -n kubernetes-dashboard
```

创建账户文件 /var/kubernetes/dashboard-rbac.yaml，内容如下。

Shell

```
1 tee /var/kubernetes/dashboard-rbac.yaml << EOF
2 apiVersion: v1
3 kind: ServiceAccount
4 metadata:
5   name: dashboard-admin
6   namespace: kube-system
7 ---
8 kind: ClusterRoleBinding
9 apiVersion: rbac.authorization.k8s.io/v1
10 metadata:
11   name: dashboard-admin
12 subjects:
13   - kind: ServiceAccount
14     name: dashboard-admin
15     namespace: kube-system
16 roleRef:
17   kind: ClusterRole
18   name: cluster-admin
19 apiGroup: rbac.authorization.k8s.io
20 EOF
```

Shell

```
1 #部署账户文件
2 kubectl apply -f /var/kubernetes/dashboard-rbac.yaml
3
4 获取Token令牌
5 kubectl create token dashboard-admin --duration=720h --namespace kube-
  system
```

使用浏览器访问 <https://10.10.2.151:30084>，将输出的 Token 填入浏览器中登录 Dashboard。

## 7.2 Labs-K8s-Master-1: 部署 Kuboard

## Shell

```
1 #为节点添加 k8s.kuboard.cn/role=etcd 的标签
2 kubectl label nodes labs-k8s-master-1 k8s.kuboard.cn/role=etcd
3 kubectl label nodes labs-k8s-master-2 k8s.kuboard.cn/role=etcd
4 kubectl label nodes labs-k8s-master-3 k8s.kuboard.cn/role=etcd
5 kubectl label nodes labs-k8s-woker-1 k8s.kuboard.cn/role=etcd
6
7 #下载 kuboard 部署清单
8 wget -P /var/kubernetes https://addons.kuboard.cn/kuboard/kuboard-v3-sw
  r.yaml
9
10 #部署 kuboard
11 kubectl apply -f /var/kubernetes/kuboard-v3-sw.yaml
12
13 #查看 pod 信息
14 kubectl get pods -n kuboard
```

## 7.3 Labs-K8s-Master-1: 安装 metrics-server 工具

## Shell

```
1 #拉取镜像
2 docker pull swr.cn-east-2.myhuaweicloud.com/kuboard-dependency/metrics-
  server:v0.6.2
3
4 #创建角色
5 kubectl create clusterrolebinding kube-proxy-cluster-admin --clusterrol
  e=cluster-admin --user=system:kube-proxy
6
7 #修订并部署
8 cat >> /var/kubernetes/metrics-server.yaml << EOF
9 apiVersion: v1
10 kind: ServiceAccount
11 metadata:
12   labels:
13     k8s-app: metrics-server
14   name: metrics-server
15   namespace: kube-system
16 ---
17 apiVersion: rbac.authorization.k8s.io/v1
18 kind: ClusterRole
19 metadata:
20   labels:
21     k8s-app: metrics-server
22     rbac.authorization.k8s.io/aggregate-to-admin: "true"
23     rbac.authorization.k8s.io/aggregate-to-edit: "true"
24     rbac.authorization.k8s.io/aggregate-to-view: "true"
25   name: system:aggregated-metrics-reader
26 rules:
27 - apiGroups:
28   - metrics.k8s.io
29   resources:
30   - pods
31   - nodes
32   verbs:
33   - get
34   - list
35   - watch
36 ---
37 apiVersion: rbac.authorization.k8s.io/v1
38 kind: ClusterRole
39 metadata:
40   labels:
```

```
41     k8s-app: metrics-server
42   name: system:metrics-server
43 rules:
44 - apiGroups:
45   - ""
46   resources:
47   - nodes/metrics
48   verbs:
49   - get
50 - apiGroups:
51   - ""
52   resources:
53   - pods
54   - nodes
55   verbs:
56   - get
57   - list
58   - watch
59 ---
60 apiVersion: rbac.authorization.k8s.io/v1
61 kind: RoleBinding
62 metadata:
63   labels:
64     k8s-app: metrics-server
65   name: metrics-server-auth-reader
66   namespace: kube-system
67 roleRef:
68   apiGroup: rbac.authorization.k8s.io
69   kind: Role
70   name: extension-apiserver-authentication-reader
71 subjects:
72 - kind: ServiceAccount
73   name: metrics-server
74   namespace: kube-system
75 ---
76 apiVersion: rbac.authorization.k8s.io/v1
77 kind: ClusterRoleBinding
78 metadata:
79   labels:
80     k8s-app: metrics-server
81   name: metrics-server:system:auth-delegator
82 roleRef:
83   apiGroup: rbac.authorization.k8s.io
84   kind: ClusterRole
```

```
85   name: system:auth-delegator
86 subjects:
87 - kind: ServiceAccount
88   name: metrics-server
89   namespace: kube-system
90 ---
91 apiVersion: rbac.authorization.k8s.io/v1
92 kind: ClusterRoleBinding
93 metadata:
94   labels:
95     k8s-app: metrics-server
96   name: system:metrics-server
97 roleRef:
98   apiGroup: rbac.authorization.k8s.io
99   kind: ClusterRole
100  name: system:metrics-server
101 subjects:
102 - kind: ServiceAccount
103   name: metrics-server
104   namespace: kube-system
105 ---
106 apiVersion: v1
107 kind: Service
108 metadata:
109   labels:
110     k8s-app: metrics-server
111   name: metrics-server
112   namespace: kube-system
113 spec:
114   ports:
115   - name: https
116     port: 443
117     protocol: TCP
118     targetPort: https
119   selector:
120     k8s-app: metrics-server
121 ---
122 apiVersion: apps/v1
123 kind: Deployment
124 metadata:
125   labels:
126     k8s-app: metrics-server
127   name: metrics-server
128   namespace: kube-system
```

```
129 spec:
130   selector:
131     matchLabels:
132       k8s-app: metrics-server
133   strategy:
134     rollingUpdate:
135       maxUnavailable: 0
136   template:
137     metadata:
138       labels:
139         k8s-app: metrics-server
140     spec:
141       containers:
142         - args:
143           - --cert-dir=/tmp
144           - --secure-port=4443
145           - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostn
ame
146           - --kubelet-use-node-status-port
147           - --metric-resolution=15s
148           - --kubelet-insecure-tls
149           image: swr.cn-east-2.myhuaweicloud.com/kuboard-dependency/metri
cs-server:v0.6.2
150           imagePullPolicy: IfNotPresent
151         livenessProbe:
152           failureThreshold: 3
153           httpGet:
154             path: /livez
155             port: https
156             scheme: HTTPS
157           periodSeconds: 10
158         name: metrics-server
159         ports:
160           - containerPort: 4443
161             name: https
162             protocol: TCP
163         readinessProbe:
164           failureThreshold: 3
165           httpGet:
166             path: /readyz
167             port: https
168             scheme: HTTPS
169           initialDelaySeconds: 20
170           periodSeconds: 10
```

```
171     resources:
172         requests:
173             cpu: 100m
174             memory: 200Mi
175     securityContext:
176         allowPrivilegeEscalation: false
177         readOnlyRootFilesystem: true
178         runAsNonRoot: true
179         runAsUser: 1000
180     volumeMounts:
181     - mountPath: /tmp
182       name: tmp-dir
183     nodeSelector:
184         kubernetes.io/os: linux
185     priorityClassName: system-cluster-critical
186     serviceAccountName: metrics-server
187     volumes:
188     - emptyDir: {}
189       name: tmp-dir
190 ---
191 apiVersion: apiregistration.k8s.io/v1
192 kind: APIService
193 metadata:
194     labels:
195         k8s-app: metrics-server
196     name: v1beta1.metrics.k8s.io
197 spec:
198     group: metrics.k8s.io
199     groupPriorityMinimum: 100
200     insecureSkipTLSVerify: true
201     service:
202         name: metrics-server
203         namespace: kube-system
204     version: v1beta1
205     versionPriority: 100
206 EOF
207
208 kubectl apply -f /var/kubernetes/metrics-server.yaml
209
210 #查看 pod 信息
211 kubectl get pods -n kube-system
```

使用浏览器访问 <http://10.10.2.151:30080>。

账号: admin

密码: Kuboard123