Kubernetes 企业级高可用部署

目录

文档信息

文档作者	房佳亮
文档版本	Version1.0
文档版权	内部资料禁止传播
文档归类	Kubernetes 项目实战训练营
系统环境	CentOS-7.X-x86_64
作者邮箱	crushlinux@163.com
修订信息	2021-03-31
技术交流	

文档约定

[绿色背景] 知识重点
[红色背景] 错误警告
[黄色背景] 注意事项

执行命令

1、Kubernetes 高可用项目介绍

单 master 节点的可靠性不高，并不适合实际的生产环境。Kubernetes 高可用集群是保证 Master 节点中 API Server 服务的高可用。API Server 提供了 Kubernetes 各类资源对象增删改查的唯一访问入口，是整个 Kubernetes 系统的数据总线和数据中心。采用负载均衡（Load Balance）连接多个 Master 节点可以提供稳定容器云业务。

2、项目架构设计

2.1、项目主机信息

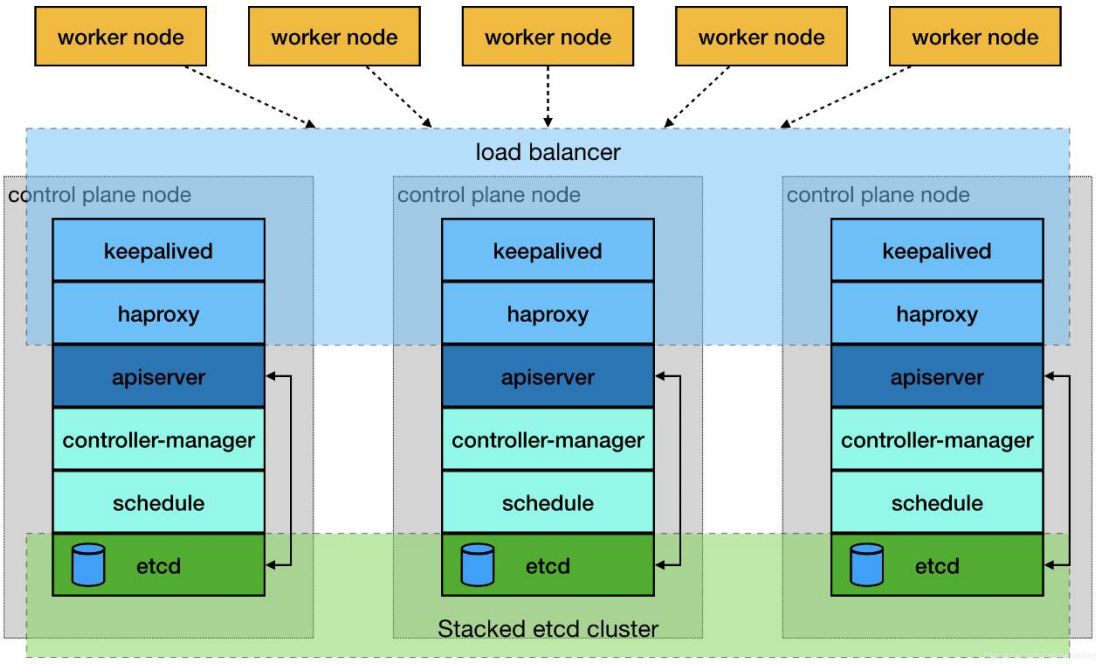
准备 6 台虚拟机，3 台 master 节点，3 台 node 节点，保证 master 节点数为>=3 的奇数。
硬件：2 核 CPU+、2G 内存+、硬盘 20G+

网络：所有机器网络互通、可以访问外网

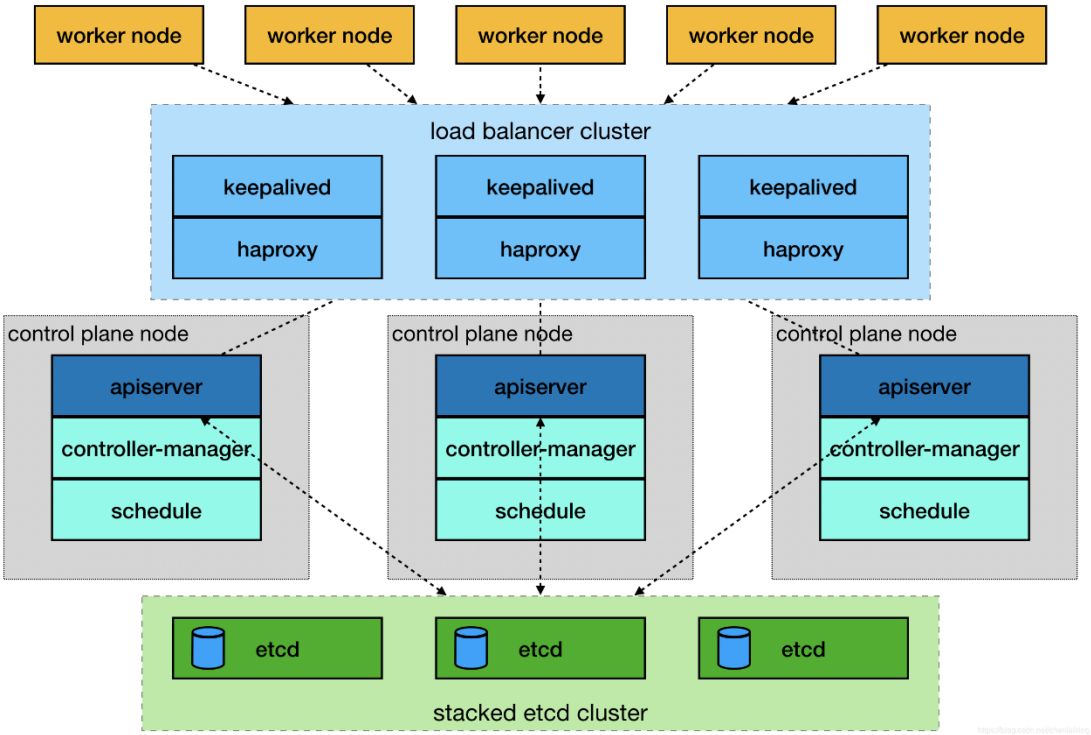
操作系统	IP 地址	角色	主机名
CentOS7-x86-64	192.168.200.111	master	k8s-master1
CentOS7-x86-64	192.168.200.112	master	k8s-master2
CentOS7-x86-64	192.168.200.113	master	k8s-master3
CentOS7-x86-64	192.168.200.114	node	k8s-node1
CentOS7-x86-64	192.168.200.115	node	k8s-node2
CentOS7-x86-64	192.168.200.116	node	k8s-node3
	192.168.200.154	VIP	master.k8s.io

2.2、项目架构图

多 master 节点负载均衡的 kubernetes 集群。官网给出了两种拓扑结构：堆叠 control plane node 和 external etcd node，本文基于第一种拓扑结构进行搭建。



（堆叠 control plane node）



（external etcd node）

2.3、项目实施思路

master 节点需要部署 etcd、apiserver、controller-manager、scheduler 这 4 种服务，其中 etcd、controller-manager、scheduler 这三种服务 kubernetes 自身已经实现了高可用，在多 master 节点的情况下，每个 master 节点都会启动这三种服务，同一时间只有一个生效。因此要实现 kubernetes 的高可用，只需要 apiserver 服务高可用。

keepalived 是一种高性能的服务器高可用或热备解决方案，可以用来防止服务器单点故障导致服务中断的问题。keepalived 使用主备模式，至少需要两台服务器才能正常工作。比如 keepalived 将三台服务器搭建成一个集群，对外提供一个唯一 IP，正常情况下只有一台服务器上可以看到这个 IP 的虚拟网卡。如果这台服务异常，那么 keepalived 会立即将 IP 移动到剩下的两台服务器中的一台上，使得 IP 可以正常使用。

haproxy 是一款提供高可用性、负载均衡以及基于 TCP（第四层）和 HTTP（第七层）应用的代理软件，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。使用 haproxy 负载均衡后端的 apiserver 服务，达到 apiserver 服务高可用的目的。

本文使用的 keepalived+haproxy 方案，使用 keepalived 对外提供稳定的入口，使用 haproxy 对内均衡负载。因为 haproxy 运行在 master 节点上，当 master 节点异常后，haproxy 服务也会停止，为了避免这种情况，我们在每一台 master 节点都部署 haproxy 服务，达到 haproxy 服务高可用的目的。由于多 master 节点会出现投票竞选的问题，因此 master 节点的数据最好是单数，避免票数相同的情况。

3、项目实施过程

3.1、系统初始化

关闭防火墙

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

关闭 selinux

```
[root@localhost ~]# sed -i 's/enforcing/disabled/' /etc/selinux/config
[root@localhost ~]# setenforce 0
```

关闭 swap

```
[root@localhost ~]# swapoff -a
[root@localhost ~]# sed -ri 's/.*swap.*/#&/' /etc/fstab
```

修改主机名（根据主机角色不同，做相应修改）

```
hostname k8s-node3
bash
```

主机名映射

```
[root@k8s-master1 ~]# cat >> /etc/hosts << EOF
192.168.200.111 master1.k8s.io k8s-master1
192.168.200.112 master2.k8s.io k8s-master2
192.168.200.113 master3.k8s.io k8s-master3
192.168.200.114 node1.k8s.io k8s-node1
192.168.200.115 node2.k8s.io k8s-node2
192.168.200.116 node3.k8s.io k8s-node3
192.168.200.154 master.k8s.io k8s-vip
EOF
```

将桥接的 IPv4 流量传递到 iptables 的链

```
[root@k8s-master1 ~]# cat << EOF >> /etc/sysctl.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
[root@k8s-master1 ~]# modprobe br_netfilter
[root@k8s-master1 ~]# sysctl -p
```

时间同步

```
[root@k8s-master1 ~]# yum install ntpdate -y
```

```
[root@k8s-master1 ~]# ntpdate time.windows.com
```

3.2、配置部署 keepalived 服务

安装 Keepalived（所有 master 主机）

```
[root@k8s-master1 ~]# yum install -y keepalived
```

k8s-master1 节点配置

```
[root@k8s-master1 ~]# cat > /etc/keepalived/keepalived.conf <<EOF
! Configuration File for keepalived
global_defs {
   router_id k8s
}
vrrp_script check_haproxy {
   script "killall -0 haproxy"
   interval 3
   weight -2
   fall 10
   rise 2
}
vrrp_instance VI_1 {
   state MASTER
   interface ens33
   virtual_router_id 51
   priority 100
   advert_int 1
   authentication {
      auth_type PASS
      auth_pass 1111
   }
virtual_ipaddress {
   192.168.200.154
}
track_script {
   check_haproxy
}
}
EOF
```

k8s-master2 节点配置

```
[root@k8s-master2 ~]# cat > /etc/keepalived/keepalived.conf <<EOF
! Configuration File for keepalived
global_defs {
```

```
    router_id k8s
}
vrrp_script check_haproxy {
    script "killall -0 haproxy"
    interval 3
    weight -2
    fall 10
    rise 2
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    virtual_router_id 51
    priority 90
    adver_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
virtual_ipaddress {
    192.168.200.154
}
track_script {
    check_haproxy
}
}
EOF
```

k8s-master3 节点配置

```
[root@k8s-master3 ~]# cat > /etc/keepalived/keepalived.conf <<EOF
! Configuration File for keepalived
global_defs {
    router_id k8s
}
vrrp_script check_haproxy {
    script "killall -0 haproxy"
    interval 3
    weight -2
    fall 10
    rise 2
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens33
```

```
    virtual_router_id 51
    priority 80
    adver_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
virtual_ipaddress {
    192.168.200.154
}
track_script {
    check_haproxy
}
}
EOF
```

启动和检查

所有 master 节点都要执行

```
[root@k8s-master1 ~]# systemctl start keepalived
[root@k8s-master1 ~]# systemctl enable keepalived
```

查看启动状态

```
[root@k8s-master1 ~]# systemctl status keepalived
Created symlink from /etc/systemd/system/multi-user.target.wants/keepalived.
service to /usr/lib/systemd/system/keepalived.service.[root@k8s-master1 ~]# systemctl status
keepalived
● keepalived.service - LVS and VRRP High Availability Monitor
    Loaded: loaded (/usr/lib/systemd/system/keepalived.service; enabled; vendor preset:
disabled)
    Active: active (running) since 日 2021-01-03 20:44:16 CST; 8s ago
  Main PID: 1670 (keepalived)
    CGroup: /system.slice/keepalived.service
            ├─1670 /usr/sbin/keepalived -D
            ├─1671 /usr/sbin/keepalived -D
            └─1672 /usr/sbin/keepalived -D

1 月 03 20:44:23 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for
192.168.200.154
1 月 03 20:44:23 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for
192.168.200.154
1 月 03 20:44:23 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for
192.168.200.154
1 月 03 20:44:23 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for
192.168.200.154
1 月 03 20:44:24 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for
```

```
192.168.200.154
1 月  03 20:44:24 k8s-master1 Keepalived_vrrp[1672]: VRRP_Instance(VI_1) Sending/queueing gratuitous ARPs on ens33 for 192.168.200.154
1 月  03 20:44:24 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for 192.168.200.154
1 月  03 20:44:24 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for 192.168.200.154
1 月  03 20:44:24 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for 192.168.200.154
1 月  03 20:44:24 k8s-master1 Keepalived_vrrp[1672]: Sending gratuitous ARP on ens33 for 192.168.200.154
```

启动完成后在 master1 查看网络信息

```
[root@k8s-master1 ~]# ip a s ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:ce:ef:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.111/24 brd 192.168.200.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet 192.168.200.154/32 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fece:ef5f/64 scope link
       valid_lft forever preferred_lft forever
```

3.3、配置部署 haproxy 服务

所有 master 主机安装 haproxy

```
[root@k8s-master1 ~]# yum install -y haproxy
```

每台 master 节点中的配置均相同，配置中声明了后端代理的每个 master 节点服务器，指定了 haproxy 的端口为 16443，因此 16443 端口为集群的入口。

```
[root@k8s-master1 ~]# cat > /etc/haproxy/haproxy.cfg << EOF
#---------------------------------
# Global settings
#---------------------------------
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
```

```
    stats socket /var/lib/haproxy/stats
#---------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# usr if not designated in their block
#---------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
#---------------------------------
# kubernetes apiserver frontend which proxys to the backends
#---------------------------------
frontend kubernetes-apiserver
    mode                    tcp
    bind                    *:16443
    option                  tcplog
    default_backend         kubernetes-apiserver
#---------------------------------
#round robin balancing between the various backends
#---------------------------------
backend kubernetes-apiserver
    mode                    tcp
    balance                 roundrobin
    server                  master1.k8s.io      192.168.200.111:6443 check
    server                  master2.k8s.io      192.168.200.112:6443 check
    server                  master3.k8s.io      192.168.200.113:6443 check
#---------------------------------
# collection haproxy statistics message
#---------------------------------
listen stats
    bind                    *:1080
```

```
  stats auth          admin:awesomePassword
  stats refresh       5s
  stats realm         HAProxy\ Statistics
  stats uri           /admin?stats
EOF
```

启动和检查

所有 master 节点都要执行

```
[root@k8s-master1 ~]# systemctl start haproxy
[root@k8s-master1 ~]# systemctl enable haproxy
```

查看启动状态

```
[root@k8s-master1 ~]# systemctl status haproxy
[root@k8s-master1 ~]# systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; vendor preset:
disabled)
   Active: active (running) since 日 2021-01-03 20:45:39 CST; 11s ago
 Main PID: 1778 (haproxy-systemd)
   CGroup: /system.slice/haproxy.service
           ├─1778 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.cfg -p
/run/haproxy.pid
           ├─1779 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds
           └─1780 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds

1 月 03 20:45:39 k8s-master1 systemd[1]: Started HAProxy Load Balancer.
1 月 03 20:45:39 k8s-master1 haproxy-systemd-wrapper[1778]: haproxy-systemd-wrapper:
executing /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /ru...pid -Ds
1 月 03 20:45:39 k8s-master1 haproxy-systemd-wrapper[1778]: [WARNING] 002/204539
(1779) : config : 'option forwardfor' ignored for frontend 'kubern...P mode.
1 月 03 20:45:39 k8s-master1 haproxy-systemd-wrapper[1778]: [WARNING] 002/204539
(1779) : config : 'option forwardfor' ignored for backend 'kuberne...P mode.
Hint: Some lines were ellipsized, use -l to show in full.
```

检查端口

```
[root@k8s-master1 ~]# netstat -lntup|grep haproxy
tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN
1780/haproxy
tcp        0      0 0.0.0.0:16443           0.0.0.0:*               LISTEN
1780/haproxy
udp        0      0 0.0.0.0:36433           0.0.0.0:*
1779/haproxy
```

3.4、配置部署 Docker 服务

所有主机上分别部署 Docker 环境，因为 Kubernetes 对容器的编排需要 Docker 的支持。

```
[root@k8s-master ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
[root@k8s-master ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
```

使用 YUM 方式安装 Docker 时，推荐使用阿里的 YUM 源。

```
[root@k8s-master ~]# yum-config-manager --add-repo https://mirrors.aliyun.com/docker-
ce/linux/centos/docker-ce.repo
[root@k8s-master ~]# yum clean all && yum makecache fast

[root@k8s-master ~]# yum -y install docker-ce
[root@k8s-master ~]# systemctl start docker
[root@k8s-master ~]# systemctl enable docker
```

镜像加速器（所有主机配置）

```
[root@k8s-master ~]# cat << END > /etc/docker/daemon.json
{
        "registry-mirrors":[ "https://nyakyfun.mirror.aliyuncs.com" ]
}
END
[root@k8s-master ~]# systemctl daemon-reload
[root@k8s-master ~]# systemctl restart docker
```

3.5、部署 kubelet kubeadm kubectl 工具

使用 YUM 方式安装 Kubernetes 时，推荐使用阿里的 yum。
所有主机配置

```
[root@k8s-master ~]# 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

[root@k8s-master ~]# ls /etc/yum.repos.d/
backup  Centos-7.repo  CentOS-Media.repo  CentOS-x86_64-kernel.repo  docker-ce.repo
```

```
kubernetes.repo
```

安装 kubelet kubeadm kubectl
所有主机配置

```
[root@k8s-master ~]# yum install -y kubelet kubeadm kubectl
[root@k8s-master ~]# systemctl enable kubelet
```

3.6、部署 Kubernetes Master

在具有 vip 的 master 上操作。此处的 vip 节点为 k8s-master1。
创建 kubeadm-config.yaml 文件

```
[root@k8s-master1 ~]# cat > kubeadm-config.yaml << EOF
apiServer:
  certSANs:
    - k8s-master1
    - k8s-master2
    - k8s-master3
    - master.k8s.io
    - 192.168.200.111
    - 192.168.200.112
    - 192.168.200.113
    - 192.168.200.154
    - 127.0.0.1
  extraArgs:
    authorization-mode: Node,RBAC
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta1
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: "master.k8s.io:6443"
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: registry.aliyuncs.com/google_containers
kind: ClusterConfiguration
kubernetesVersion: v1.20.0
networking:
  dnsDomain: cluster.local
  podSubnet: 10.244.0.0/16
  serviceSubnet: 10.1.0.0/16
```

```
scheduler: {}
EOF
```

查看所需镜像信息

```
[root@k8s-master1 ~]# kubeadm config images list --config kubeadm-config.yaml
W0103 21:00:29.765926       3145 common.go:77] your configuration file uses a deprecated API spec: "kubeadm.k8s.io/v1beta1". Please use 'kubeadm config migrate --old-config old.yaml --new-config new.yaml', which will write the new, similar spec using a newer API version.
registry.aliyuncs.com/google_containers/kube-apiserver:v1.20.0
registry.aliyuncs.com/google_containers/kube-controller-manager:v1.20.0
registry.aliyuncs.com/google_containers/kube-scheduler:v1.20.0
registry.aliyuncs.com/google_containers/kube-proxy:v1.20.0
registry.aliyuncs.com/google_containers/pause:3.2
registry.aliyuncs.com/google_containers/etcd:3.4.13-0
registry.aliyuncs.com/google_containers/coredns:1.7.0
```

上传 k8s 所需的镜像并导入（所有 master 主机）

```
[root@k8s-master1 ~]# cd master/
[root@k8s-master1 master]# ls
coredns_1.7.0.tar    kube-apiserver_v1.20.0.tar          kube-proxy_v1.20.0.tar
pause_3.2.tar etcd_3.4.13-0.tar    kube-controller-manager_v1.20.0.tar    kube-scheduler_v1.20.0.tar
[root@k8s-master1 master]# ls | while read line
do
docker load < $line
done
```

使用 kubeadm 命令初始化 k8s

```
[root@k8s-master1 ~]# kubeadm init --config kubeadm-config.yaml
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 control-plane nodes by copying certificate authorities
and service account keys on each node and then running the following as root:

  kubeadm join master.k8s.io:6443 --token vhqazv.c398m3ohhpl8avd0 \  加入master时使用
    --discovery-token-ca-cert-hash sha256:40fc58440b1ea96c64816050c197b6ef0cd8ac42f687d0e54b8b2f9c392a5617 \
    --control-plane

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

kubeadm join master.k8s.io:6443 --token vhqazv.c398m3ohhpl8avd0 \    加入node时使用
    --discovery-token-ca-cert-hash sha256:40fc58440b1ea96c64816050c197b6ef0cd8ac42f687d0e54b8b2f9c392a5617
```

初始化中的错误：

[ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables contents are not set to 1

执行以下命令后重新执行初始化命令

```
echo "1" >/proc/sys/net/bridge/bridge-nf-call-iptables
```

根据初始化的结果操作

```
[root@k8s-master1 ~]# mkdir -p $HOME/.kube
[root@k8s-master1 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master1 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

查看集群状态

[root@k8s-master1 manifests]# kubectl get cs

```
[root@k8s-master1 ~]# kubectl get cs
Warning: v1 ComponentStatus is deprecated in v1.19+
NAME                 STATUS      MESSAGE                                                                                     ERROR
controller-manager   Unhealthy   Get "http://127.0.0.1:10252/healthz": dial tcp 127.0.0.1:10252: connect: connection refused
scheduler            Unhealthy   Get "http://127.0.0.1:10251/healthz": dial tcp 127.0.0.1:10251: connect: connection refused
etcd-0               Healthy     {"health":"true"}
```

注意：出现以上错误情况，是因为/etc/kubernetes/manifests/下的 kube-controller-manager.yaml 和 kube-scheduler.yaml 设置的默认端口为 0 导致的，解决方式是注释掉对应的 port 即可

修改 kube-controller-manager.yaml 文件

```
[root@k8s-master1 ~]# cat /etc/kubernetes/manifests/kube-controller-manager.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-controller-manager
    tier: control-plane
  name: kube-controller-manager
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-controller-manager
    - --allocate-node-cidrs=true
    - --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf
    - --authorization-kubeconfig=/etc/kubernetes/controller-manager.conf
    - --bind-address=127.0.0.1
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --cluster-cidr=10.244.0.0/16
    - --cluster-name=kubernetes
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
#    - --port=0
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
```

修改 kube-scheduler.yaml 文件

```
[root@k8s-master1 ~]# cat /etc/kubernetes/manifests/kube-scheduler.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-scheduler
    tier: control-plane
  name: kube-scheduler
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-scheduler
    - --authentication-kubeconfig=/etc/kubernetes/scheduler.conf
    - --authorization-kubeconfig=/etc/kubernetes/scheduler.conf
    - --bind-address=127.0.0.1
    - --kubeconfig=/etc/kubernetes/scheduler.conf
    - --leader-elect=true
#    - --port=0
    image: registry.aliyuncs.com/google_containers/kube-scheduler:v1.20.0
```

查看集群状态

```
[root@k8s-master1 ~]# kubectl get cs
Warning: v1 ComponentStatus is deprecated in v1.19+
NAME                 STATUS    MESSAGE             ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health":"true"}
```

查看 pod 信息

```
[root@k8s-master1 ~]# kubectl get pods -n kube-system
NAME                                 READY   STATUS    RESTARTS   AGE
coredns-7f89b7bc75-52kjr             0/1     Pending   0          8m30s
coredns-7f89b7bc75-7h8js             0/1     Pending   0          8m30s
etcd-k8s-master1                     1/1     Running   0          8m38s
kube-apiserver-k8s-master1           1/1     Running   0          8m38s
kube-controller-manager-k8s-master1  1/1     Running   0          2m11s
kube-proxy-s24dn                     1/1     Running   0          8m31s
kube-scheduler-k8s-master1           1/1     Running   0          119s
```

查看节点信息

```
[root@k8s-master1 ~]# kubectl get nodes
NAME          STATUS     ROLES                  AGE     VERSION
k8s-master1   NotReady   control-plane,master   9m24s   v1.20.0
```

3.7、安装集群网络

在 k8s-master1 节点执行

```
[root@k8s-master1 ~]# docker load < flannel_v0.12.0-amd64.tar
```

```
[root@k8s-master1 ~]# kubectl apply -f kube-flannel.yml
```

再次查看节点信息：

```
[root@k8s-master1 ~]# kubectl get nodes
NAME          STATUS   ROLES                  AGE     VERSION
k8s-master1   Ready    control-plane,master   8m13s   v1.20.1
```

3.8、添加 master 节点

在 k8s-master2 和 k8s-master3 节点创建文件夹

```
[root@k8s-master2 ~]# mkdir -p /etc/kubernetes/pki/etcd

[root@k8s-master3 ~]# mkdir -p /etc/kubernetes/pki/etcd
```

在 k8s-master1 节点执行
从 k8s-master1 复制秘钥和相关文件到 k8s-master2 和 k8s-master3

```
[root@k8s-master1 ~]# scp /etc/kubernetes/admin.conf
root@192.168.200.112:/etc/kubernetes
[root@k8s-master1 ~]# scp /etc/kubernetes/admin.conf
root@192.168.200.113:/etc/kubernetes

[root@k8s-master1 ~]# scp /etc/kubernetes/pki/{ca.*,sa.*,front-proxy-ca.*}
root@192.168.200.112:/etc/kubernetes/pki
[root@k8s-master1 ~]# scp /etc/kubernetes/pki/{ca.*,sa.*,front-proxy-ca.*}
root@192.168.200.113:/etc/kubernetes/pki

[root@k8s-master1 ~]# scp /etc/kubernetes/pki/etcd/ca.*
root@192.168.200.112:/etc/kubernetes/pki/etcd
[root@k8s-master1 ~]# scp /etc/kubernetes/pki/etcd/ca.*
root@192.168.200.113:/etc/kubernetes/pki/etcd
```

将其他 master 节点加入集群
注意：kubeadm init 生成的 token 有效期只有 1 天，生成不过期 token

```
[root@k8s-master1 manifests]# kubeadm token create --ttl 0 --print-join-command
[root@k8s-master1 manifests]# kubeadm token list
```

```
[root@k8s-master1 ~]# kubeadm token create --ttl 0 --print-join-command
kubeadm join master.k8s.io:6443 --token 5zvcjk.i6r9vijc869svkke     --discovery-token-ca-cert-hash sha256:40fc58440b1ea96c64816050c197b6ef0cd8ac42f687d0e54b8b2f9c392a5617
[root@k8s-master1 ~]# kubeadm token list
TOKEN                     TTL         EXPIRES   USAGES                   DESCRIPTION                                                EXTRA GROUPS
5zvcjk.i6r9vijc869svkke   <forever>   <never>   authentication,signing   <none>                                                     system:bootstrappers:kubeadm:default-node-token
vhqazv.c398m3ohhpl8avd0   23h         2021-01-04T21:06:38+08:00   authentication,signing   <none>                                                     system:bootstrappers:kubeadm:default-node-token
```

k8s-master2 和 k8s-master3 都需要加入

```
[root@k8s-master2 ~]# kubeadm join master.k8s.io:6443 --token 5zvcjk.i6r9vijc869svkke \
    --discovery-token-ca-cert-hash
```

```
sha256:40fc58440b1ea96c64816050c197b6ef0cd8ac42f687d0e54b8b2f9c392a5617 \
    --control-plane

[root@k8s-master2 ~]# mkdir -p $HOME/.kube
[root@k8s-master2 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master2 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config

[root@k8s-master2 ~]# docker load < flannel_v0.12.0-amd64.tar
```

```
[root@k8s-master1 ~]# kubectl get nodes
NAME          STATUS   ROLES                  AGE     VERSION
k8s-master1   Ready    control-plane,master   19m     v1.20.0
k8s-master2   Ready    control-plane,master   2m20s   v1.20.0
k8s-master3   Ready    control-plane,master   95s     v1.20.0

[root@k8s-master1 manifests]# kubectl get pods --all-namespaces
[root@k8s-master1 ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE
kube-system   coredns-7f89b7bc75-52kjr               1/1     Running   0          19m
kube-system   coredns-7f89b7bc75-7h8js               1/1     Running   0          19m
kube-system   etcd-k8s-master1                       1/1     Running   0          19m
kube-system   etcd-k8s-master2                       1/1     Running   0          2m34s
kube-system   etcd-k8s-master3                       1/1     Running   0          43s
kube-system   kube-apiserver-k8s-master1             1/1     Running   0          19m
kube-system   kube-apiserver-k8s-master2             1/1     Running   0          2m38s
kube-system   kube-apiserver-k8s-master3             1/1     Running   1          110s
kube-system   kube-controller-manager-k8s-master1    1/1     Running   1          13m
kube-system   kube-controller-manager-k8s-master2    1/1     Running   0          2m38s
kube-system   kube-controller-manager-k8s-master3    1/1     Running   0          36s
kube-system   kube-flannel-ds-amd64-nfxcc            1/1     Running   0          115s
kube-system   kube-flannel-ds-amd64-pb94t            1/1     Running   0          2m40s
kube-system   kube-flannel-ds-amd64-x8lck            1/1     Running   0          9m31s
kube-system   kube-proxy-gdb5l                       1/1     Running   0          2m40s
kube-system   kube-proxy-s24dn                       1/1     Running   0          19m
kube-system   kube-proxy-x89vb                       1/1     Running   0          115s
kube-system   kube-scheduler-k8s-master1             1/1     Running   1          12m
kube-system   kube-scheduler-k8s-master2             1/1     Running   0          2m38s
kube-system   kube-scheduler-k8s-master3             1/1     Running   0          35s
```

3.9、加入 Kubernetes Node

直接在 node 节点服务器上执行 k8s-master1 初始化成功后的消息即可：

```
[root@k8s-node1 ~]# kubeadm join master.k8s.io:6443 --token 5zvcjk.i6r9vijc869svkke \
    --discovery-token-ca-cert-hash
sha256:40fc58440b1ea96c64816050c197b6ef0cd8ac42f687d0e54b8b2f9c392a5617

[root@k8s-node1 ~]# docker load < flannel_v0.12.0-amd64.tar
```

查看节点信息

```
[root@k8s-master1 ~]# kubectl get nodes
NAME          STATUS   ROLES                  AGE     VERSION
k8s-master1   Ready    control-plane,master   23m     v1.20.0
```

```
k8s-master2    Ready    control-plane,master    6m42s    v1.20.0
k8s-master3    Ready    control-plane,master    5m57s    v1.20.0
k8s-node1      Ready    <none>                  82s      v1.20.0
k8s-node2      Ready    <none>                  77s      v1.20.0
k8s-node3      Ready    <none>                  60s      v1.20.0
```

3.10、测试 Kubernetes 集群

所有 node 主机导入测试镜像

```
[root@k8s-node1 ~]# docker load < nginx-1.19.tar
[root@k8s-node1 ~]# docker tag nginx nginx:1.19.6
```

在 Kubernetes 集群中创建一个 pod，验证是否正常运行。

```
[root@k8s-master1 ~]# mkdir demo
[root@k8s-master1 ~]# cd demo
[root@k8s-master1 demo]# vim nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.19.6
        ports:
        - containerPort: 80
```

创建完 Deployment 的资源清单之后，使用 create 执行资源清单来创建容器。通过 get pods 可以查看到 Pod 容器资源已经自动创建完成。

```
[root@k8s-master1 demo]# kubectl create -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

```
[root@k8s-master1 demo]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-76ccf9dd9d-c5bjw   1/1     Running   0          10s
nginx-deployment-76ccf9dd9d-jxm9z   1/1     Running   0          10s
nginx-deployment-76ccf9dd9d-kzdfz   1/1     Running   0          10s

[root@k8s-master1 ~]# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP
NODE          NOMINATED NODE   READINESS GATES
nginx-deployment-76ccf9dd9d-c5bjw   1/1     Running   0          62m     10.244.4.2
k8s-node2     <none>           <none>
nginx-deployment-76ccf9dd9d-jxm9z   1/1     Running   0          62m     10.244.5.2
k8s-node3     <none>           <none>
nginx-deployment-76ccf9dd9d-kzdfz   1/1     Running   0          62m     10.244.3.2
k8s-node1     <none>           <none>
```

创建 Service 资源清单

在创建的 nginx-service 资源清单中，定义名称为 nginx-service 的 Service、标签选择器为 app: nginx、type 为 NodePort 指明外部流量可以访问内部容器。在 ports 中定义暴露的端口库号列表，对外暴露访问的端口是 80，容器内部的端口也是 80。

```
[root@k8s-master1 demo]# vim nginx-service.yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80

[root@k8s-master1 demo]# kubectl create -f nginx-service.yaml
service/nginx-service created
[root@k8s-master1 demo]# kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.1.0.1        <none>        443/TCP        38m
nginx-service   NodePort    10.1.161.204    <none>        80:30373/TCP   4s
```

通过浏览器访问 nginx：http://master.k8s.io:30373 域名或者 VIP 地址

```
[root@k8s-master1 demo]# elinks --dump http://master.k8s.io:30373
                                Welcome to nginx!
```

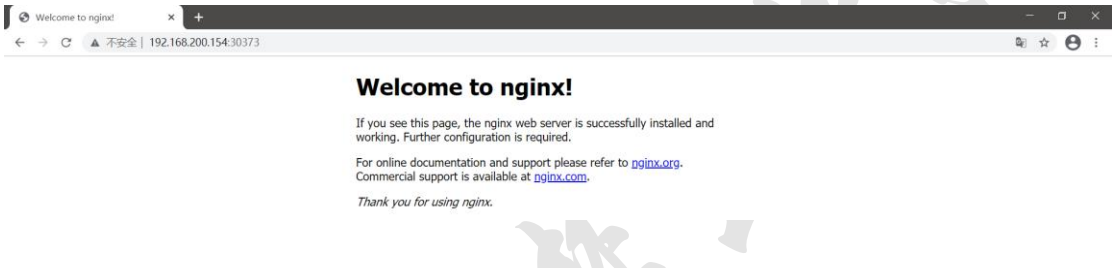
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [1]nginx.org. Commercial support is available at [2]nginx.com.

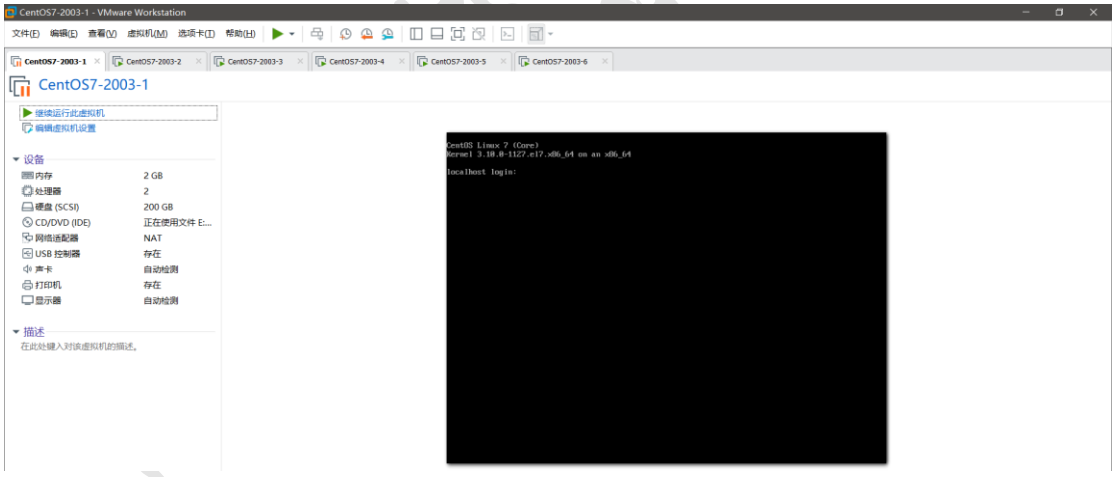
Thank you for using nginx.

References

Visible links
1. http://nginx.org/
2. http://nginx.com/



挂起 k8s-master1 节点，刷新页面还是能访问 nginx，说明高可用集群部署成功。



检查会发现 VIP 已经转移到 k8s-master2 节点上

[root@k8s-master2 ~]# ip a s ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

```
    link/ether 00:0c:29:3a:76:54 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.112/24 brd 192.168.200.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet 192.168.200.154/32 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe3a:7654/64 scope link
       valid_lft forever preferred_lft forever
```

至此 Kubernetes 企业级高可用环境完美实现。

4、项目总结

1、集群中只要有一个 master 节点正常运行就可以正常对外提供业务服务。
2、如果需要在 master 节点使用 kubectl 相关的命令，必须保证至少有 2 个 master 节点正常运行才可以使用，不然会有 Unable to connect to the server: net/http: TLS handshake timeout 这样的错误。
3、节点故障时 pod 自动转移：当 pod 所在的节点宕机后，根据 controller-manager 的–pod-eviction-timeout 配置，默认是 5 分钟，5 分钟后 k8s 会把 pod 状态设置为 unkown， 然后在其它节点启动 pod。当故障节点恢复后，k8s 会删除故障节点上面的 unkown pod。如果你想立即强制迁移，可以用 kubectl drain nodename
4、为了保证集群的高可用性，建议 master 节点和 node 节点至少分别部署 3 台及以上，且 master 节点应该部署基数个实例(3、5、7、9)。