麒麟V11服务器系统安装Kubernetes高可用集群

环境介绍

硬件环境：`3C6000`

操作系统版本：`Kylin-server-v11`

内核版本：`6.6.0-32.7.v2505.ky11.loongarch64`

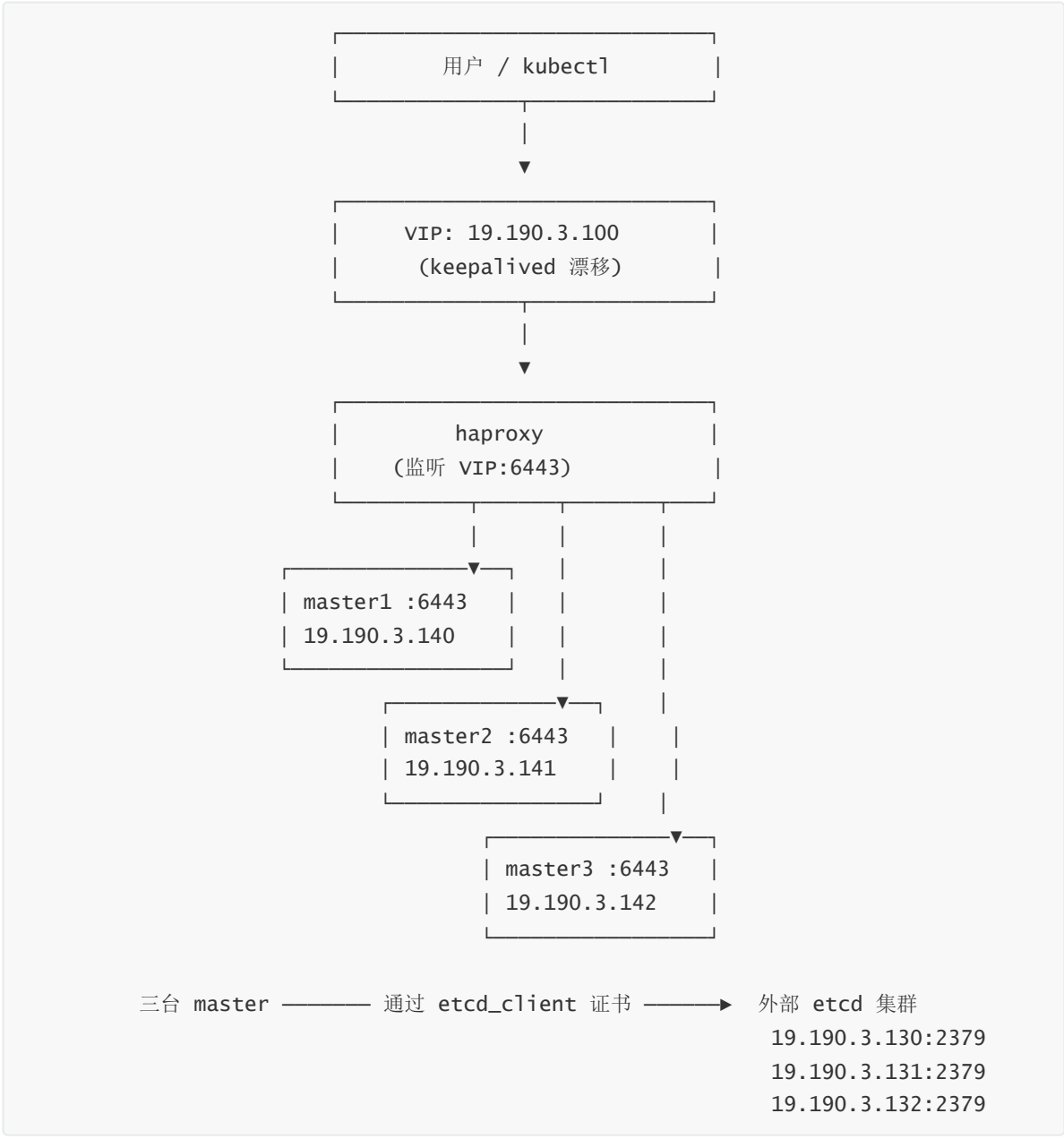
集群规划

角色	主机名	IP地址	组件
master-01	master1	19.190.3.140	kube-apiserver, kube-controller-manager, kube-scheduler, keepalived, haproxy
master-02	master2	19.190.3.141	kube-apiserver, kube-controller-manager, kube-scheduler, keepalived, haproxy
master-03	master3	19.190.3.142	kube-apiserver, kube-controller-manager, kube-scheduler, keepalived, haproxy
VIP	—	19.190.3.100	虚拟IP，由 keepalived 漂移
worker-01	node1	19.190.3.143	kubelet, kube-proxy

外部 etcd 集群（已部署，复用已有集群）：

角色	主机名	IP地址
etcd1	etcd1	19.190.3.130

角色	主机名	IP地址
etcd2	etcd2	19.190.3.131
etcd3	etcd3	19.190.3.132



> **IP 划分说明：**
>
> - `19.190.3.130-132` 已被外部 etcd 集群占用，K8s master 使用 `19.190.3.140-142`，互不冲突。
> - VIP `19.190.3.100` 为独立虚拟 IP，不与任何物理节点冲突。

前置条件：

- 所有节点已完成 containerd 及 CNI 基础插件的安装。
- 外部 etcd 集群（`19.190.3.130-132`）已部署并正常运行，证书存放在各 master 节点的 `/root/etcd_crt/` 目录下。

系统初始化

1. 配置系统参数

创建文件`/etc/sysctl.d/k8s.conf`，内容如下：

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.ipv4.conf.all.rp_filter = 1
```

然后执行命令`sudo sysctl --system`

2. 设置其他参数

- 关闭swap

```
$ sudo swapoff -a
```

- 关闭selinux

```
$ sudo setenforce 0
$ sudo sed -i "s#SELINUX=enforcing#SELINUX=disabled#g" /etc/selinux/config
```

- 关闭firewalld

```
$ systemctl stop firewalld.service

$ systemctl  disable  firewalld.service
```

3. 安装ipadm

kubernetes的kube-proxy默认使用的是iptables的转发，如果想使用需要安装ipvsadm及ipset

```
yum  install -y  ipvsadm   ipset
```

配置加载ipvs相关模块

```
# 加载
cat > /etc/sysconfig/modules/ipvs.modules << EOF
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack
EOF
```

给ipvs 赋予权限

```
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntrack
```

安装 kubeadm、kubelet、kubectl

> 以下操作在所有 master 节点（`19.190.3.140-142`）和 worker 节点上执行。

1. 下载 Kubernetes 软件包

```
[root@master1 ~]# wget https://wh.wujf.cn/Package/19_kubernetes/kubernetes-
v1.32/kubernetes-src.tar.gz
```

2. 解压并安装二进制文件

```
[root@master1 ~]# tar -xvf kubernetes-src.tar.gz
[root@master1 ~]# cd kubernetes/server/bin/
[root@master1 bin]# cp kubeadm kubelet kubectl /usr/local/bin/
[root@master1 bin]# chmod +x /usr/local/bin/kubeadm /usr/local/bin/kubelet
/usr/local/bin/kubectl
```

3. 验证版本

```
[root@master1 bin]# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"32", GitVersion:"v1.32.x",
...}
[root@master1 bin]# kubelet --version
Kubernetes v1.32.x
[root@master1 bin]# kubectl version --client
Client Version: v1.32.x
```

4. 创建 kubelet service 文件

```
[root@master1 ~]# cat > /usr/lib/systemd/system/kubelet.service << EOF
[Unit]
Description=Kubernetes Kubelet
Documentation=https://github.com/kubernetes/kubernetes
After=containerd.service
Requires=containerd.service

[Service]
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/config.yaml \\
  --container-runtime-endpoint=unix:///run/containerd/containerd.sock \\
  --kubeconfig=/etc/kubernetes/kubelet.conf \\
  --network-plugin=cni \\
  --v=2
Restart=on-failure
RestartSec=5
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF
```

5. 创建 kubelet 所需目录

```
[root@master1 ~]# mkdir -p /var/lib/kubelet
[root@master1 ~]# mkdir -p /etc/kubernetes
```

安装配置 haproxy

> 在三台 master 节点（`19.190.3.140-142`）上都执行以下操作。

1. 安装 haproxy

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

2. 配置 haproxy

```
[root@master1 ~]# cat > /etc/haproxy/haproxy.cfg << EOF
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

defaults
    mode                    tcp
    log                     global
    option                  tcplog
    option                  dontlognull
    timeout connect         10s
    timeout client          1m
    timeout server          1m

frontend k8s-apiserver
    bind *:6443
    mode tcp
    default_backend k8s-masters

backend k8s-masters
    mode tcp
    option tcp-check
    balance roundrobin
    server master1 19.190.3.140:6443 check fall 3 rise 2
    server master2 19.190.3.141:6443 check fall 3 rise 2
    server master3 19.190.3.142:6443 check fall 3 rise 2

listen stats
    bind *:8080
    mode http
    stats enable
    stats uri /stats
    stats realm HAProxy\ Statistics
    stats auth admin:admin123
EOF
```

> - `frontend` 监听所有地址的 6443 端口（keepalived 将 VIP 绑定到本机后生效）。
> - `backend` 使用 roundrobin 策略在三台 master 之间做负载均衡。
> - `stats` 页面可通过 `http://<节点IP>:8080/stats`（账密 `admin/admin123`）查看 haproxy 状态。

3. 启动 haproxy

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

4. 验证 haproxy 监听

```
[root@master1 ~]# netstat -tlnp | grep 6443
```

安装配置 keepalived

> 在三台 master 节点上都执行，注意修改各节点的 `priority`、`unicast_src_ip` 和 `unicast_peer`。

1. 安装 keepalived

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

2. 配置 keepalived（master1 — `19.190.3.140`）

```
[root@master1 ~]# cat > /etc/keepalived/keepalived.conf << EOF
! Configuration File for keepalived

global_defs {
    router_id LVS_MASTER1
}

vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    interval 2
    weight -20
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    unicast_src_ip 19.190.3.140
    unicast_peer {
        19.190.3.141
        19.190.3.142
    }
    authentication {
        auth_type PASS
        auth_pass K8SHA_KA_AUTH
```

```
    }
    virtual_ipaddress {
        19.190.3.100/24 dev eth0
    }
    track_script {
        chk_haproxy
    }
}
EOF
```

3. 配置 keepalived（master2 — `19.190.3.141`）

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

global_defs {
    router_id LVS_MASTER2
}

vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    interval 2
    weight -20
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 90
    advert_int 1
    unicast_src_ip 19.190.3.141
    unicast_peer {
        19.190.3.140
        19.190.3.142
    }
    authentication {
        auth_type PASS
        auth_pass K8SHA_KA_AUTH
    }
    virtual_ipaddress {
        19.190.3.100/24 dev eth0
    }
    track_script {
        chk_haproxy
    }
}
EOF
```

4. 配置 keepalived（master3 — `19.190.3.142`）

```
[root@master3 ~]# cat > /etc/keepalived/keepalived.conf << EOF
! Configuration File for keepalived
```

```
global_defs {
    router_id LVS_MASTER3
}

vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    interval 2
    weight -20
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 80
    advert_int 1
    unicast_src_ip 19.190.3.142
    unicast_peer {
        19.190.3.140
        19.190.3.141
    }
    authentication {
        auth_type PASS
        auth_pass K8SHA_KA_AUTH
    }
    virtual_ipaddress {
        19.190.3.100/24 dev eth0
    }
    track_script {
        chk_haproxy
    }
}
EOF
```

> **关键参数说明：**
>
> - `state` 统一设为 `BACKUP`，避免节点恢复后抢占导致 VIP 频繁漂移。
> - `priority`：master1=100, master2=90, master3=80。值越大优先级越高。
> - `unicast_src_ip` 填写本机 IP，`unicast_peer` 填写另外两台 master 的 IP。
> - `track_script` 关联 haproxy 健康检查：haproxy 异常时自动降低优先级，触发 VIP 漂移。
> - `interface` 根据实际网卡名修改（通过 `ip a` 查看）。

5. 启动 keepalived

```
## 三台节点都执行
[root@master1 ~]# systemctl daemon-reload
[root@master1 ~]# systemctl start keepalived
[root@master1 ~]# systemctl enable keepalived
```

6. 验证 VIP 是否生效

```
[root@master1 ~]# ip a | grep 19.190.3.100
    inet 19.190.3.100/24 scope global secondary eth0
```

> VIP 只会出现在 priority 最高的存活节点上。正常情况下 master1（priority=100）持有 VIP。

从 etcd 节点同步证书到 master 节点

> 在 etcd1（`19.190.3.130`）上执行，将 etcd_client 证书分发到三台 K8s master 节点。

```
[root@etcd1 ~]# scp -r /root/etcd_crt root@19.190.3.140:/root/
[root@etcd1 ~]# scp -r /root/etcd_crt root@19.190.3.141:/root/
[root@etcd1 ~]# scp -r /root/etcd_crt root@19.190.3.142:/root/
```

> 分发完成后，各 master 节点的 `/root/etcd_crt/` 目录下应包含以下文件：
>
> ```
> [root@master1 ~]# ls -l /root/etcd_crt/
> ca.crt
> ca.key          # 不需要，可删除
> etcd_client.crt
> etcd_client.key
> etcd_server.crt # 不需要，可删除
> etcd_server.key # 不需要，可删除
> etcd_ssl.cnf    # 不需要，可删除
> ```
>
> K8s apiserver 连接 etcd 只需要 `ca.crt`、`etcd_client.crt`、`etcd_client.key` 三个文件。

初始化第一个 master 节点

> 在 master1（`19.190.3.140`）上执行。

1. 创建 kubeadm 配置文件

```
[root@master1 ~]# cat  kubeadm.yml
apiVersion: kubeadm.k8s.io/v1beta4
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 19.190.3.140       ## 本机ip
  bindPort: 6443
nodeRegistration:
  criSocket: unix:///run/containerd/containerd.sock
  imagePullPolicy: IfNotPresent
  name: master1
  taints: null
timeouts:
  controlPlaneComponentHealthCheck: 4m0s
  discovery: 5m0s
  etcdAPICall: 2m0s
  kubeletHealthCheck: 4m0s
```

```
  kubernetesAPICall: 1m0s
  tlsBootstrap: 5m0s
  upgradeManifests: 5m0s
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1                    ## 配置kube-
proxy模式为ipvs
kind: KubeProxyConfiguration
mode: ipvs
---
apiServer:
  certSANs:
    - 19.190.3.100              # VIP
    - 19.190.3.140
    - 19.190.3.141
    - 19.190.3.142
    - master1
    - master2
    - master3
    - localhost
    - kubernetes
    - kubernetes.default
    - kubernetes.default.svc
    - kubernetes.default.svc.cluster.local
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta4
caCertificateValidityPeriod: 87600h0m0s
certificateValidityPeriod: 8760h0m0s
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: "19.190.3.100:6443"
controllerManager: {}
dns: {}
encryptionAlgorithm: RSA-2048
etcd:
  external:
    endpoints:                                          # etcd 集群节点地址列表
      - "https://19.190.3.130:2379"
      - "https://19.190.3.131:2379"
      - "https://19.190.3.132:2379"
    caFile: "/root/etcd_crt/ca.crt"             # etcd CA 证书路径
    certFile: "/root/etcd_crt/etcd_client.crt"  # 客户端证书路径
    keyFile: "/root/etcd_crt/etcd_client.key"   # 客户端私钥路径
imageRepository: lcr.loongnix.cn/kubernetes
kind: ClusterConfiguration
kubernetesVersion: 1.32.1
networking:
  dnsDomain: cluster.local
  serviceSubnet: 10.201.0.0/16
  podSubnet: 10.200.0.0/16
proxy: {}
scheduler: {}
---
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
```

```
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
- 10.201.0.10
clusterDomain: cluster.local
containerRuntimeEndpoint: unix:///run/containerd/containerd.sock
cpuManagerReconcilePeriod: 0s
crashLoopBackOff: {}
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMaximumGCAge: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
logging:
  flushFrequency: 0
  options:
    json:
      infoBufferSize: "0"
    text:
      infoBufferSize: "0"
  verbosity: 0
memorySwap: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
```

> - `controlPlaneEndpoint` 指向 VIP+haproxy 端口，后续所有节点加入均通过该地址。
> - `etcd.external` 指向已部署的外部 etcd 集群（`19.190.3.130-132`），与 K8s master 节点 IP 不冲突。
> - `certSANs` 包含 VIP 和三台 master 的 IP 及主机名，确保 kubectl 通过 VIP 访问时证书验证通过。
> - `imageRepository` 使用龙芯镜像仓库。

2. 执行初始化

```
[root@master1 ~]# kubeadm init --config /root/kubeadm-config.yaml --upload-certs
```

> 初始化成功后，输出类似以下内容：
>
> ```
> Your Kubernetes control-plane has initialized successfully!
>
> You can now join any number of control-plane nodes by running the following command:
>
> kubeadm join 19.190.3.100:6443 --token <token> \
>    --discovery-token-ca-cert-hash sha256:<hash> \
>    --control-plane --certificate-key <cert-key>
>
> Then you can join any number of worker nodes by running:
>
> kubeadm join 19.190.3.100:6443 --token <token> \
>    --discovery-token-ca-cert-hash sha256:<hash>
> ```
>
> **请妥善保存上述两条 join 命令**，后续加入 master 和 worker 节点时需要使用。

3. 配置 kubectl 管理集群

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

4. 验证节点状态

```
[root@master1 ~]# kubectl get nodes
NAME      STATUS     ROLES           AGE   VERSION
master1   NotReady   control-plane   30s   v1.32.1
```

> 此时节点为 `NotReady` 状态是正常的，因为尚未安装 CNI 网络插件。

加入其他 master 节点

> 在 master2（`19.190.3.141`）和 master3（`19.190.3.142`）上分别执行。

1. 执行 join 命令（master2）

```
[root@master2 ~]# kubeadm join 19.190.3.100:6443 \
    --token <token> \
    --discovery-token-ca-cert-hash sha256:<hash> \
    --control-plane --certificate-key <cert-key>
```

2. 执行 join 命令（master3）

```
[root@master3 ~]# kubeadm join 19.190.3.100:6443 \
    --token <token> \
    --discovery-token-ca-cert-hash sha256:<hash> \
    --control-plane --certificate-key <cert-key>
```

> `--control-plane` 表示加入为控制平面节点。
> `--certificate-key` 用于解密集群证书，确保新 master 可以共享同一套证书。

3. 配置 kubectl（master2、master3）

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

```
[root@master3 ~]# mkdir -p $HOME/.kube
[root@master3 ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@master3 ~]# chown $(id -u):$(id -g) $HOME/.kube/config
```

4. 验证集群控制平面

```
[root@master1 ~]# kubectl get nodes
NAME      STATUS     ROLES           AGE     VERSION
master1   NotReady   control-plane   5m32s   v1.32.1
master2   NotReady   control-plane   2m10s   v1.32.1
master3   NotReady   control-plane   45s     v1.32.1
```

> 若 token 过期（默认 24 小时），在 master1 上重新生成：
>
> ```
> [root@master1 ~]# kubeadm token create --print-join-command
> [root@master1 ~]# kubeadm init phase upload-certs --upload-certs
> ```

加入 worker 节点

1. 在 master1 上获取 worker 节点的 join 命令

```
[root@master1 ~]# kubeadm token create --print-join-command
```

2. 在 worker 节点（`19.190.3.143`）上执行 join

```
[root@node1 ~]# kubeadm join 19.190.3.100:6443 \
    --token <token> \
    --discovery-token-ca-cert-hash sha256:<hash>
```

> worker 节点的 join 命令 **不带** `--control-plane` 和 `--certificate-key` 参数。

安装 Calico 网络插件

> 在 master1 上执行。

1. 下载 Calico 清单文件

```
[root@master1 ~]# wget https://github.com/Loongson-Cloud-
Community/calico/releases/download/v3.26.1/calico.yaml
```

2. 修改 pod 网段（需与 kubeadm 配置中 `podSubnet` 保持一致）

```
[root@master1 ~]# sed -i 's|192.168.0.0/16|10.200.0.0/16|g' calico.yaml
```

3. 应用 Calico

```
[root@master1 ~]# kubectl apply -f calico.yaml
```

4. 等待所有 Pod 就绪

```
[root@master1 ~]# kubectl get pods -n kube-system -w
```

5. 验证节点状态

```
[root@master1 ~]# kubectl get nodes
NAME      STATUS   ROLES           AGE     VERSION
master1   Ready    control-plane   10m     v1.32.1
master2   Ready    control-plane   7m      v1.32.1
master3   Ready    control-plane   5m      v1.32.1
node1     Ready    <none>          2m      v1.32.1
```

> 所有节点从 `NotReady` 变为 `Ready`，说明 CNI 网络已正常工作。

验证集群高可用

1. 查看控制平面组件状态

```
[root@master1 ~]# kubectl get pods -n kube-system | grep -E 'kube-
apiserver|kube-controller|kube-scheduler'
kube-apiserver-master1             1/1     Running   0   10m
kube-apiserver-master2             1/1     Running   0   7m
kube-apiserver-master3             1/1     Running   0   5m
kube-controller-manager-master1    1/1     Running   0   10m
kube-controller-manager-master2    1/1     Running   0   7m
kube-controller-manager-master3    1/1     Running   0   5m
kube-scheduler-master1             1/1     Running   0   10m
kube-scheduler-master2             1/1     Running   0   7m
kube-scheduler-master3             1/1     Running   0   5m
```

2. 验证通过 VIP 访问 apiserver

```
[root@master1 ~]# kubectl cluster-info
Kubernetes control plane is running at https://19.190.3.100:6443
```

3. 模拟 master1 故障，验证 VIP 漂移

```
## 在 master1（19.190.3.140）上停止 haproxy，模拟故障
[root@master1 ~]# systemctl stop haproxy

## 在 master2 上查看 VIP 是否已漂移
[root@master2 ~]# ip a | grep 19.190.3.100
    inet 19.190.3.100/24 scope global secondary eth0
```

> VIP 从 master1 漂移至 master2，此时通过 VIP 访问 apiserver 仍然正常，高可用架构生效。

4. 恢复 master1

```
[root@master1 ~]# systemctl start haproxy
```

> 恢复后 VIP 不会自动回切（所有节点 state 均为 BACKUP 且未开启抢占），避免连接中断。

5. 创建测试 Pod 验证集群功能

```
[root@master1 ~]# kubectl create deployment nginx-test --
image=lcr.loongnix.cn/library/nginx:latest --replicas=3
[root@master1 ~]# kubectl expose deployment nginx-test --port=80 --
type=NodePort
[root@master1 ~]# kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP
  NODE
nginx-test-xxxxxxxxxx-xxxxx   1/1     Running   0          10s   10.200.x.x
  node1
nginx-test-xxxxxxxxxx-xxxxx   1/1     Running   0          10s   10.200.x.x
  node1
nginx-test-xxxxxxxxxx-xxxxx   1/1     Running   0          10s   10.200.x.x
  node1
```

> Pod 能正常创建并分配到 `10.200.0.0/16` 网段的 IP，说明 CNI 和集群调度均正常工作。