Kubernetes(K8S)笔记03

记录下Kubernetes学习笔记03，方便以后复习，具体参考网络资料Kubernetes 手册。

搭建k8s集群

搭建Kubernetes集群

搭建方案

- minikube
- kubeadm
- 二进制安装
- 命令行工具

命令行工具

- kubectl

API概述

- 类型
  - Alpha
  - Beta
  - Stable
- 访问控制
  - 认证
  - 授权
- 废弃 api 说明

这里使用kubeadm搭建k8s集群

服务器要求

- 3台服务器
  - k8s-master： `192.168.64.134`
  - k8s-node1： `192.168.64.135`
  - k8s-node2： `192.168.64.136`
- 最低配置：2核、2G内存、20G硬盘
- 最好能联网

软件环境

- 操作系统：Ubuntu 24.04
- Docker：20+
- k8s：1.23.6

安装部署

- 初始操作
- 安装基础软件（所有节点）
  - 安装 Docker
  - 添加阿里云 yum 源

    - 安装 kubeadm、kubelet、kubectl
- 部署 kubernetes master
- 加入 kubernetes node
- 部署 CNI 网络插件
- 测试 kubernetes 集群

前置步骤

关闭防火墙

查看当前的防火墙状态： `sudo ufw status`

关闭防火墙： `sudo ufw disable`

关闭swap

暂时关闭： `sudo swapoff -a`

永久关闭：先关闭swap，再删除Swap分区文件： `sudo rm /swap.img` ，然后编辑 `/etc/fstab` 文件，注释或者删除 `/swap.img none swap sw 0 0` 这一行。

总的命令：

```
sudo swapoff -a
sudo rm /swap.img
sudo vim /etc/fstab
```

开启网络转发等配置

overlay 是文件系统。由于Docker是分层的，上层的文件会覆盖下层的文件，使用到了overlay文件系统。

br_netfilter 网络转发。br_netfilter模块可以使 iptables 规则可以在 Linux Bridges 上面工作，用于将桥接的流量转发至iptables链。如果没有加载br_netfilter模块，那么并不会影响不同node上的pod之间的通信，但是会影响同node内的pod之间通过service来通信。

1.加载两个内核模块

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

持久化上述的两个模块：

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
```

通过运行 `lsmod | grep br_netfilter` 和 `lsmod | grep overlay` 来检查模块是否已加载。

2.设置内核参数，确保二层的网桥在转发包时也会被iptables的forward规则所过滤

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

3.应用sysctl配置

```
sudo sysctl --system
```

总的命令：

```
sudo modprobe overlay
sudo modprobe br_netfilter
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sudo sysctl --system
```

设置主机名

查看当前主机名Check the Current Hostname： `hostname`

```
caicloudcat@caicloudcat-VMware-Virtual-Platform:~$ hostname
caicloudcat-VMware-Virtual-Platform
caicloudcat@caicloudcat-VMware-Virtual-Platform:~$
```

临时更改主机名： `sudo hostname linuxconfig`

永久更改主机名： `sudo hostnamectl set-hostname linuxconfig` ,再重启

使用命令检查主机名更改： `hostnamectl`

```
caicloudcat@caicloudcat-VMware-Virtual-Platform:~$ sudo hostnamectl set-hostname
 k8s-master
caicloudcat@caicloudcat-VMware-Virtual-Platform:~$ hostnamectl
 Static hostname: k8s-master
       Icon name: computer-vm
         Chassis: vm 🖴
      Machine ID: 023a52ebed874f119a06cfb86ea5167e
         Boot ID: 604cdb4cae4c4f2ab8201a3d9d26bcb6
  Virtualization: vmware
Operating System: Ubuntu 24.04 LTS
          Kernel: Linux 6.8.0-51-generic
    Architecture: x86-64
 Hardware Vendor: VMware, Inc.
  Hardware Model: VMware Virtual Platform
Firmware Version: 6.00
   Firmware Date: Thu 2020-11-12
    Firmware Age: 4y 1month 2w 2d
caicloudcat@caicloudcat-VMware-Virtual-Platform:~$
```

最后修改hosts文件： `sudo vim /etc/hosts`

详见Setting the Hostname on Ubuntu 24.04

在master节点设置hosts

```
sudo sh -c 'cat >> /etc/hosts <<EOF
192.168.64.134 k8s-master
192.168.64.135 k8s-node1
192.168.64.136 k8s-node2
EOF'
```

设置后，可以通过 `ping k8s-node1` 来测试是否设置成功。

设置远程连接

安装OpenSSH： `sudo apt install openssh-server`

启动OpenSSH： `sudo systemctl start ssh`

设置开机自启： `sudo systemctl enable ssh`

安装Docker

具体详见k8s的官方下载手册Install Docker Engine on Debian

卸载旧版本：

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

官方文档安装

使用apt存储库安装：

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

安装Docker软件包

安装最新版： `sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`

要安装特定版本，首先列出存储库中可用的版本：

```
# List the available versions:
apt-cache madison docker-ce | awk '{ print $3 }'

5:27.4.0-1~debian.12~bookworm
5:27.3.1-1~debian.12~bookworm
...
```

选择所需的版本安装：

```
VERSION_STRING=5:27.4.0-1~debian.12~bookworm
sudo apt-get install docker-ce=$VERSION_STRING docker-ce-cli=$VERSION_STRING containerd.io docker-buildx-plugin docker-compose-p
```

国内镜像源安装

当然，考虑到国内网络问题，上面的安装可能会失败，可以参考安装手册最新安装Docker教程

具体步骤如下：

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://mirrors.cloud.tencent.com/docker-ce/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo   "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://mirrors.cloud.tencent.com/docker-
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" |   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo systemctl start docker
sudo systemctl enable docker

sudo docker version
sudo docker compose version
```

验证安装： `sudo docker run hello-world`

配置Docker镜像源

具体可以用的镜像源可以参考：Docker镜像源、容器镜像库

如果安装上面的验证安装成功，则可以跳过。

第一步：新建或编辑daemon.json： `sudo vim /etc/docker/daemon.json`

第二步：daemon.json中编辑如下：

```
{
    "registry-mirrors": [
      "https://docker.m.daocloud.io",
      "https://docker.1panel.live"
      ]
}
```

第三步：重启docker： `sudo systemctl restart docker.service`

第四步：执行docker info查看是否修改成功： `docker info`

再次验证安装： `sudo docker run hello-world`

启动Docker，设置开机自启： `sudo systemctl enable docker --now`

安装kubeadm、kubelet、kubectl

```
curl -fsSL https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-archiv
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xen
sudo apt-get update
```

安装最新版本

```
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

安装特定版本

查看可用版本

```
sudo apt-cache madison kubelet
sudo apt-cache madison kubeadm
sudo apt-cache madison kubectl
```

我们安装的是 `kubeadm=1.23.6-00`

```
sudo apt-get install kubelet=1.23.6-00
sudo apt-get install kubeadm=1.23.6-00
sudo apt-get install kubectl=1.23.6-00
sudo apt-mark hold kubelet kubeadm kubectl
```

查看安装的版本：

```
kubelet --version
kubeadm version
kubectl version
```

设置开机自启： `sudo systemctl enable kubelet --now`

安装cri-dockerd

初始化master节点

接下来在master节点上进行初始化操作。

```
sudo kubeadm init \
--apiserver-advertise-address=192.168.64.134 \
--image-repository registry.aliyuncs.com/google_containers \
--kubernetes-version v1.23.6 \
--service-cidr=10.96.0.0/12 \
--pod-network-cidr=10.244.0.0/16
```

讲解：

- `--apiserver-advertise-address=192.168.64.134` ： 指定API Server的IP地址，这里使用的是master节点的IP地址。
- `--image-repository registry.aliyuncs.com/google_containers` ： 指定镜像仓库，这里使用的是阿里云的镜像仓库。
- `--kubernetes-version v1.23.6` ： 指定Kubernetes的版本，这里使用的是v1.23.6。
- `--service-cidr=10.96.0.0/12` ： 指定Service的CIDR范围，这里使用的是10.96.0.0/12。
- `--pod-network-cidr=10.244.0.0/16` ： 指定Pod的CIDR范围，这里使用的是10.244.0.0/16。

最后的输出：

```
[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
[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/

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

kubeadm join 192.168.64.134:6443 --token j9wtio.0lecxfnvse3ywakg \
        --discovery-token-ca-cert-hash sha256:ac4da8ae0a220ab1669e3128cdef89f3d45116791f000c8a3490b73e57730869
caicloudcat@k8s-master:~$
```

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
```

执行 `kubectl get nodes` ， 可以看到master节点已经加入到集群中。

```
caicloudcat@k8s-master:~$ kubectl get nodes
NAME          STATUS     ROLES                  AGE   VERSION
k8s-master    NotReady   control-plane,master   71m   v1.23.6
caicloudcat@k8s-master:~$
```

加入k8s-node节点

在k8s-node节点（k8s-node1、k8s-node2）上执行：

```
sudo kubeadm join 192.168.64.134:6443 --token <master控制台的token> \
--discovery-token-ca-cert-hash sha256:<master控制台的hash值>
```

如果初始化的token不小心清空了，可以通过下面的命令获取或者重新申请：

```
kubeadm token list
```

```
caicloudcat@k8s-master:~$ kubeadm token list
TOKEN                     TTL         EXPIRES                USAGES                   DESCRIPTION
          EXTRA GROUPS
j9wtio.0lecxfnvse3ywakg   22h         2024-12-30T10:11:44Z   authentication,signing   The default bootstrap token generated by 'kubeadm
 init'.   system:bootstrappers:kubeadm:default-node-token
caicloudcat@k8s-master:~$
```

可以看到这个token值为： `j9wtio.0lecxfnvse3ywakg`

如果已经过期，则需要重新申请：

```
kubeadm token create
```

接下来获取master节点的hash值：

```
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex |
```

```
caicloudcat@k8s-master:~$ openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl d
gst -sha256 -hex | sed 's/^.* //'
ac4da8ae0a220ab1669e3128cdef89f3d45116791f000c8a3490b73e57730869
caicloudcat@k8s-master:~$
```

这段命令的作用是从Kubernetes的CA证书中提取公钥，并计算其SHA-256哈希值，以便在加入节点时使用，这里的hash值为 `ac4da8ae0a220ab1669e3128cdef89f3d45116791f000c8a3490b73e57730869` 。

在Kubernetes中，使用--discovery-token-ca-cert-hash参数时，要求提供的hash值必须以 `sha256:` 前缀开头。这是因为Kubernetes需要明确知道所提供的hash值是使用SHA-256算法计算的。

因此最后的命令为（在k8s-node节点上执行）：

```
sudo kubeadm join 192.168.64.134:6443 --token j9wtio.0lecxfnvse3ywakg \
--discovery-token-ca-cert-hash sha256:ac4da8ae0a220ab1669e3128cdef89f3d45116791f000c8a3490b73e57730869
```

```
caicloudcat@k8s-node1:~$ sudo kubeadm join 192.168.64.134:6443 --token j9wtio.0lecxfnvse3ywakg \
--discovery-token-ca-cert-hash sha256:ac4da8ae0a220ab1669e3128cdef89f3d45116791f000c8a3490b73e57730869
[sudo] password for caicloudcat:
[preflight] Running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 27.4.1. Latest validated version: 2
0.10
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
W1229 19:39:07.416059    7682 utils.go:69] The recommended value for "resolvConf" in "KubeletConfiguration" is: /run/systemd/resolve/re
solv.conf; the provided value is: /run/systemd/resolve/resolv.conf
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

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

caicloudcat@k8s-node1:~$
```

执行 `kubectl get nodes` ，可以看到k8s-node1、k8s-node2节点已经加入到集群中。

```
caicloudcat@k8s-master:~$ kubectl get nodes
NAME          STATUS     ROLES                  AGE    VERSION
k8s-master    NotReady   control-plane,master   89m    v1.23.6
k8s-node1     NotReady   <none>                 108s   v1.23.6
k8s-node2     NotReady   <none>                 77s    v1.23.6
caicloudcat@k8s-master:~$
```

部署CNI网络插件

在使用 `kubectl get nodes` 命令时，如果看到 `STATUS` 为 `NotReady` ，则需要部署CNI网络插件。

我们又用 `kubectl get pods -n kube-system` 命令查看，可以发现有两个pod没有正常运行。

```
caicloudcat@k8s-master:~$ kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS        AGE
coredns-6d8c4cb4d-25rv7               0/1     Pending   0               107m
coredns-6d8c4cb4d-2xzsf               0/1     Pending   0               107m
etcd-k8s-master                       1/1     Running   1 (2m24s ago)   107m
kube-apiserver-k8s-master             1/1     Running   1 (2m24s ago)   107m
kube-controller-manager-k8s-master    1/1     Running   1 (2m24s ago)   107m
kube-proxy-gjmc7                      1/1     Running   1 (96s ago)     19m
kube-proxy-ksn8f                      1/1     Running   1 (2m24s ago)   107m
kube-proxy-szbjz                      1/1     Running   1 (87s ago)     19m
kube-scheduler-k8s-master             1/1     Running   1 (2m24s ago)   107m
caicloudcat@k8s-master:~$
```

在这里我们使用calico网络插件，在master节点执行命令。

首先创建对应的文件夹进行管理：

```
cd /opt
mkdir k8s
cd k8s
```

下载calico的yaml文件：

```
sudo wget https://docs.tigera.io/archive/v3.25/manifests/calico.yaml
```

修改calico.yaml文件中的 `CALICO_IPV4POOL_CIDR` 配置，修改为与初始化的 `--pod-network-cidr` 相同，具体看初始化master节点：

```
sudo vim calico.yaml
```

`CALICO_IPV4POOL_CIDR` 配置，大致在4601行，修改为 `10.244.0.0/16` 。

```
          key: veth_mtu
    # The default IPv4 pool to create on startup if none exists. Pod IPs will be
    # chosen from this range. Changing this value after installation will have
    # no effect. This should fall within `--cluster-cidr`.
    # - name: CALICO_IPV4POOL_CIDR
    #   value: "192.168.0.0/16"
    # Disable file logging so `kubectl logs` works.
    - name: CALICO_DISABLE_FILE_LOGGING
      value: "true"
    # Set Felix endpoint to host default action to ACCEPT.
    - name: FELIX_DEFAULTENDPOINTTOHOSTACTION
      value: "ACCEPT"
    # Disable IPv6 on Kubernetes.
    - name: FELIX_IPV6SUPPORT
      value: "false"
```

接下来配置 `IP_AUTODETECTION_METHOD` 。

完毕后，查看下载calico.yaml文件对应需要的镜像 `grep image calico.yaml` ：

```
caicloudcat@k8s-master:/opt/k8s$ grep image calico.yaml
          image: docker.io/calico/cni:v3.25.0
          imagePullPolicy: IfNotPresent
          image: docker.io/calico/cni:v3.25.0
          imagePullPolicy: IfNotPresent
          image: docker.io/calico/node:v3.25.0
          imagePullPolicy: IfNotPresent
          image: docker.io/calico/node:v3.25.0
          imagePullPolicy: IfNotPresent
          image: docker.io/calico/kube-controllers:v3.25.0
          imagePullPolicy: IfNotPresent
```

可以看到，镜像前面有 `docker.io/` 前缀，为避免下载过慢导致失败,因而可以删除该前缀。

```
sudo sed -i 's#docker.io/##g' calico.yaml
```

然后下载镜像：

```
sudo docker pull calico/cni:v3.25.0
sudo docker pull calico/node:v3.25.0
sudo docker pull calico/kube-controllers:v3.25.0
```

```
kubectl apply -f calico.yaml
```

执行 `kubectl get pods -n kube-system` ，可以看到已经有显示对应的calico的Pod。

```
caicloudcat@k8s-master:/opt/k8s$ kubectl get po -n kube-system
NAME                                       READY   STATUS     RESTARTS      AGE
calico-kube-controllers-64cc74d646-lv7mc   0/1     Pending    0             27s
calico-node-5pz64                          0/1     Init:0/3   0             27s
calico-node-fkf7l                          0/1     Init:0/3   0             27s
calico-node-fwqdd                          0/1     Init:0/3   0             27s
coredns-6d8c4cb4d-25rv7                    0/1     Pending    0             146m
coredns-6d8c4cb4d-2xzsf                    0/1     Pending    0             146m
etcd-k8s-master                            1/1     Running    1 (41m ago)   146m
kube-apiserver-k8s-master                  1/1     Running    1 (41m ago)   146m
kube-controller-manager-k8s-master         1/1     Running    1 (41m ago)   146m
kube-proxy-gjmc7                           1/1     Running    1 (40m ago)   59m
kube-proxy-ksn8f                           1/1     Running    1 (41m ago)   146m
kube-proxy-szbjz                           1/1     Running    1 (40m ago)   58m
kube-scheduler-k8s-master                  1/1     Running    1 (41m ago)   146m
caicloudcat@k8s-master:/opt/k8s$
```

期间可以使用 `kubectl describe po xxx -n kube-system` 查看对应的Pod的详细信息。

如果遇到 `Init:ErrorImagePull` ，具体可以参考K8s的Pod出现Init:ImagePullBackOff问题的解决(以calico为例)

测试k8s集群

```
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --port=80 --type=NodePort
kubectl get pod,svc
```

```
caicloudcat@k8s-master:/opt/k8s$ kubectl get pod,svc
NAME                        READY   STATUS    RESTARTS   AGE
pod/nginx-85b98978db-jqp8m  1/1     Running   0          31m

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        6h24m
service/nginx        NodePort    10.109.62.151   <none>        80:32602/TCP   30m
caicloudcat@k8s-master:/opt/k8s$ curl 192.168.64.135:32602
```

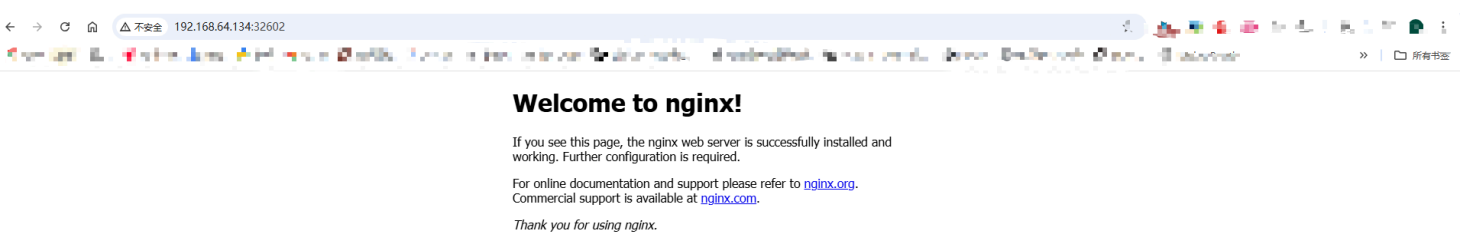
使用 `curl 192.168.64.134` ，可以看到nginx的页面。

```
caicloudcat@k8s-master:/opt/k8s$ curl 192.168.64.134:32602
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
caicloudcat@k8s-master:/opt/k8s$
```

同理，也可以使用 `192.168.64.135` 、 `192.168.64.136` 访问到nginx的页面。



参考资料

k8s集群搭建（基于v1.23.6） 2024-05-18

深入架构原理与实践 8.4.1 Pod

基于Ubuntu下安装kubernetes集群指南 2023-08-10

How Install Kubernetes on Ubuntu 24.04 (Step-by-Step Guide)，在csdn上找到一个翻译过得Ubuntu 24.04 上安装 Kubernetes，超级详细的教程！，但是标签显示的是“原创”，稍微有点搞笑，所以还是看英文的吧。

一文讲明白K8S各核心架构组件

Pod 解析

k8s版本号说明 - 个人博客

Pod控制器