【Kubernetes k8s】 (两万字超详细) Ubuntu-22.04搭建 k8s-1.30.1集群,开启D rd-2.7.0、部署ingress-nginx-1.10.1



Ubuntu-22.04 搭建 k8s-1.30.1 集群,开启 Dashboard-v2.7.0 (以及Token不生成的问题)、部署 i nginx-1.10.1

引言

最近在研究分布式计算 ,想将分布式计算都容器化,使用 k8s 来调度,所以从0开始学 k8s ,这是我遇到坑后百度、查资料一点一点总结的搭建 这方便以后自己找,也希望对你能有所帮助,后续我会不定时更新这篇博客的内容

*** 特别注意: 当 k8s 集群 init 了以后的 kubectl 命令最好都等上一步的命令执行完了再进行下一步操作,不要抢着执行,随时看 Pod 的状态 get pods -A , 只要还有一个没有 Running 就不要着急下一步

2024-09-06 实测 v1.31.0 也可用本文的方式搭建

-、系统环境准备

*** 以下在所有节点上都要做

用 Ubuntu-22.04 版本, CentOS 操作核心是一样的,只是命令不同罢了。

1、关闭 SWap

官方要求关闭 swap ,虚拟内存相关,因为 Kubernetes 无法读取虚拟内存相关的数据,开启这个可能会导致 Kubernetes 的内存问题

要想永久关,得先解除程序占用,临时关闭 swap:

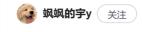
AI写代码 1 swapoff -a

修改配置文件永久关闭, 注释这个文件:

AI写代码

1 | vi /etc/fstab

中的关于 swap 的那一行,如图,注释掉它:











```
/etc/fstab: static file system information.
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
# <file system> <mount point> <type> <options>
                                                       <dump> <pass>
# / was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/48172609-ccdb-48ab-889a-50513e3b31c3 / ext4 defaults 0 1
#/swap.img
               none
                       swap
                               SW
                                                              CSDN @飒飒的宇y
```

2、关闭 SELinux

Ubuntu 没有这个东西可以不管,CentOS 执行:

AI写代码 sh

1 | sudo sed -i 's/^SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config

然后重启系统即可。 SELinux 是系统安全方面的,如果你懂怎么弄可以自己配置 SELinux 就不用关,很麻烦,所以大家基本都关了的

3、关闭防火墙

需要关闭防火墙,因为节点之间要互相通信,开防火墙容易出问题

sh AI写代码

- 1 | ufw disable
- 2 # 或者
- 3 | systemctl disable --now ufw

4、设置时区(根据自己的时间情况可选)

因为关系到集群机器之间的通信,需要一个时间同步大家的行为。设置为上海时区

AI写代码

1 | timedatectl set-timezone Asia/Shanghai

重启时间同步服务

AI写代码

1 | systemctl restart systemd-timesyncd.service

看一下时间对不对:

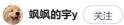
AI写代码 sh

- timedatectl status 1
- # 或者
- 3 date

5、修改 /etc/hosts (可选) 、 /etc/hostname (可选但建议)

这个是为了后面填地址的时候方便,不用填IP地址,直接填名字就行

修改 hosts 文件,将自己的IP地址和主机名值讲去:













sh Al写代码

```
1 echo "192.168.10.10 k8s-master" | sudo tee -a /etc/hosts
```

其他节点的你想填也填进去

修改 hostsname 文件,两种办法,一种是 hostnamectl 命令,一种是直接修改文件,修改文件的方式:

sh AI写代码

```
1 echo "k8s-master" | sudo tee /etc/hostname # 不同的节点改不同的名字
```

修改完主机名文件后最好重启一次ssh终端或系统

6、开启流量转发

开这个的原因是让各个主机承担起网络路由的角色,因为后续还要安装网络插件,要有一个路由器各个 Pod 才能互相通信。执行:

```
sh

1 | cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
2 | net.ipv4.ip_forward = 1
```

应用参数:

3 E0F

sh Al写代码

1 | sudo sysctl --system

查看是否开启成功:

sh AI写代码

```
1 | sysctl net.ipv4.ip_forward
```

没错的话会看到结果:

sh AI写代码

```
1 net.ipv4.ip_forward = 1 # 开启成功
```

7、其他相关知识(不重要)

I、IP地址固定以及网络相关知识

你有可能想固定机器的IP地址,编辑 /etc/netplan 下的 xxx.yaml 文件,如果只有一张网卡,就只有一个文件,如果有多张网卡根据网卡名来,wi 件名也带有 wifi 字样。以下是一个有线网卡的固定IP例子,根据你的需求改就行:

yaml AI写代码

```
1
  network:
2
    ethernets:
3
     ens33: # 要固定的网卡名
4
       dhcp4: false # false是关闭自动获取地址, true是开启
5
       addresses:
6
         - 192.168.10.10/24 # 你要固定的IP地址
7
       gateway4: 192.16
8
                        《 飒飒的宇y ( 关注 )
                                                                                                       < 分
       nameservers:
                                                                                        146 108
                                                                              73
9
         addresses: # D
```

改完文件后应用更改:

sh AI写代码

1 | netplan apply

验证是否成功,执行:

sh Al写代码

1 ip a # 查看所有网卡以及对应的IP地址

我看到评论区有关于网络的提问,我来粗俗简单的解释一下IP网段以及/数字的问题,详细的去学习网络相关的知识:

如果你有一个子网是 192.168.1.0 网段,那么它的子网掩码就是 255.255.255.0 ,缩写就是 /24 ,那么这个子网可分配的IP地址就是从 192.168.1 192.168.1.254 结束,共 254 个IP地址。

如果你的机器很多,那么这254个地址不够用,可以提升网络,我们将之前的 192.168.1.0 子网再提升一下,变成 192.168.0.0 网段,它的子网掩码 255.255.0.0 ,缩写就是 /16 ,可分配地址是从 192.168.1.1 开始到 192.168.255.254 结束,共 65536 个IP地址,这下就够用了。

那为什么是 /24 、 /16 呢,你可以理解为 x.x.x.x,四个 x ,一 x 个占 8 位,我们刚才用到的 192.168.1.0 ,只将 192.168.1.x 分配给主机,所以 位。同样的 192.168.0.0 可分配的地址就是 192.168.x.x 就是 16 位, 192.x.x.x 就是 8 位

二、安装 containerd 运行时环境

*** 以下在所有节点上都要做

Kubernetes-1.24 版本移除了 dockershim 的支持,所以之前的先安装 docker 再安装 Kubernets 的方式已经不可行了,可能会导致 Kubelet 所以我们要先安装 containerd 运行时环境。这时可能就有人问了,安装 Docker 时不是带有 containerd 了吗,你说得对,安装 Docker 时确实 install -y docker-ce docker-ce-cli containerd.io ,确实安装了 containerd ,但是这是 Docker 的 containerd 啊,这么安装的 conta Docker 管理的,k8s 无法管理,所以就会导致报错,踩过的坑啊。

运行时环境除了 containerd 以外还有 CRI-0 、 Mirantis Container Runtime 、 Docker Engine ,但我们用 containerd 就行,其他的运行时安装手册

安装 containerd 有两种方式:

1、下载最新二进制文件安装

这种方式虽然麻烦,但是可以安装最新版的 containerd , 具体操作:

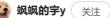
I、安装 runc 客户端

因为我们是手动安装 containerd ,所以也得手动安装 runc ,这里就不上 Github 上找包了, apt 工具的 runc 已经很新了。执行:

Sh AI写代码

1 | sudo apt install -y runc













去 GitHub 上下载二进制包, 想要其他版本的话替换两处 1.7.17 为你想要的版本即可:

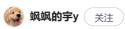
```
AI写代码
   1 \mid \text{curl -0 https://github.com/containerd/containerd/releases/download/v1.7.17/containerd-1.7.17-linux-amd64.tar.gz}
  解压文件到当前目录:
                                                                                                                AI写代码
   1 | tar -zxvf containerd-1.7.17-linux-amd64.tar.gz
  会得到一个文件夹 bin, 移动 bin 中的所有文件到目录 /bin 中去:
                                                                                                                 AI写代码
   1 | mv bin/* /bin
  由于我们是手动解压二进制文件安装的,所以得先生成一个 Service ,创建这个 Service 文件并且添加内容:
                                                                                                                 AI写代码
 bash
   1
     cat <<EOF | sudo tee /etc/systemd/system/containerd.service</pre>
   3
      Description=containerd container runtime
      Documentation=https://containerd.io
   5
      After=network.target
   6
   7
     [Service]
   8
     ExecStart=/bin/containerd
   9 Type=notify
                                                               展开 🗸
  然后重新加载 systemd 并启动 containerd:
                                                                                                                 AI写代码
 sh
   1
      sudo systemctl daemon-reload && \
      sudo systemctl start containerd && \
   3 | sudo systemctl enable containerd
  执行看版本正不正常:
                                                                                                                 AI写代码
 sh
   1 | containerd -v
root@k8s-master:/app/k8s/dashboard# containerd -v
rooteks muster , app nes, as new containerd v1.7.17 3a4de459a68952ffb703bbe7f2290861a75b6b67
CSDN @ஆஸ்
root@k8s-master:/app/k8s/dashboard#
```

2、通过包管理器安装

这种方式最简单,但是一般来说版本会滞后三四个版本左右,例如我们手动安装的版本是 v1.7.17 ,通过 apt 安装的版本一般为 v1.7.12

I、安装

先更新源再安装,这里不用安装













AI写代码

1 apt update && apt install -y containerd

Ⅱ、查看版本

执行命令版本没问题就可以了(一般落后最新版三四个版本左右):

AI写代码 sh

1 | containerd -v

3、生成配置文件(重要)

还要手动生成配置文件,不管是二进制文件安装的还是包管理器安装的,都要执行:

sh AI写代码

```
1 | sudo mkdir -p /etc/containerd && \
```

sudo containerd config default > /etc/containerd/config.toml

修改 /etc/containerd/config.toml 文件中:

```
[plugins."io.containerd.grpc.vl.cri".containerd.runtimes.runc] 下
[plugins."io.containerd.grpc.vl.cri".containerd.runtimes.runc.options] 下的 SystemdCgroup 为 true:
```

```
snapsnotter
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
 BinaryName = ""
 CriuImagePath = ""
 CriuPath = ""
 CriuWorkPath = ""
 IoGid = 0
  IoUid = 0
 NoNewKeyring = false
 NoPivotRoot = false
 Root =
  ShimCgroup = ""
 SystemdCgroup = true
                                                              CSDN @飒飒的宇y
```

修改这个配置是因为 kubelet 和底层容器运行时(我们用的是 containerd)都需要对接控制组来强制为 Pod 和容器管理资源,并且运行时和 k8s 需 初始化系统,Ubuntu 默认使用的初始化系统是 systemd , k8s v1.22 起,如果没有在 KubeletConfiguration 下设置 cgroupDriver 字段, kubea systemd ,所以我们只需要设置 containerd 就行了。 k8s官方解释地址

改完了别忘记重启一下 containerd:

AI写代码

1 | systemctl restart containerd

三、安装 kubeadm、kubelet、kubectl

*** 以下在所有节点上都要做

1、简单介绍

kubeadm 是自动引导整个集群的工具。本质上 k8s 就是一些容器服务相互配合完成管理生群的任务。如果你知道且体安装哪些容器那么可以不用这

kubalet 是各个节点的总管,

















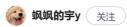
2、安装

首先得保证源都是新的:

AI写代码 sh 1 sudo apt update && \ sudo apt upgrade -y 然后安装一些必要工具: AI写代码 1 | sudo apt install -y apt-transport-https ca-certificates curl gpg 如果 /etc/apt/keyrings 目录不存在, 先创建 AI写代码 1 | sudo mkdir -p -m 755 /etc/apt/keyrings 下载 k8s 包仓库的公共签名密钥。解释一下,密钥中有一个 v1.30 ,所有版本都是用的这个格式的密钥,即使你改为其他版本,下载的都是一个密 响,不过你想改也行: AI写代码 sh curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | \ 1 sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg && \ sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg 添加 k8s 的apt仓库,使用其他版本的替换地址中的 v1.30 就行: AI写代码 $1 \mid \mathsf{echo} \mid \mathsf{deb} \mid \mathsf{[signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]} \mid \mathsf{https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' \setminus \mathsf{v1.30/deb/} \mid \mathsf{v2.30/deb/} \mid \mathsf{v3.30/deb/} \mid \mathsf{v$ 2 | sudo tee /etc/apt/sources.list.d/kubernetes.list 更新 apt 索引,并且安装,还要防止软件更新,三步: AI写代码 sh sudo apt update && \ sudo apt install -y kubelet kubectl kubeadm && \ sudo apt-mark hold kubelet kubeadm kubectl 启动 kubelet , 并且设置开机自启: AI写代码 1 | sudo systemctl enable --now kubelet 看看有没有安装成功:

1 kubeadm version

sh











AI写代码

```
root@k8s-master:/app/k8s/dashboard#
root@k8s-master:/app/k8s/dashboard#
root@k8s-master:/app/k8s/dashboard#
root@k8s-master:/app/k8s/dashboard# kube
root@k8s-master:/app/k8s/dashboard# kubeadm version
kubeadm version: &version.lnfo{Major:"1", Minor:"30", GitVersion:"v1.30.1", GitCommit:"6911225c3f747e1cd9d109c305436d08b668f086", GitTreeState:"clean", BuildDate:"2024-05-14T10:49:05Z", Go
Compiler:"gc", Platform:"linux/amd64")
root@k8s-master:/app/k8s/dashboard# []
```

四、初始化主节点 (只在主节点上做)

1、提前拉取镜像(可选)

如果你觉得慢或者出了什么未知的问题,可以提前将所需的镜像拉取下来,因为之前说过了, k8s 实质上是一堆容器服务组合,调度管理其他容器的 容器就得需要镜像。你可以在 init 前运行这个命令:

AI写代码 sh

- sudo kubeadm config images pull \
- --kubernetes-version=v1.30.1 \
- --cri-socket=unix:///run/containerd/containerd.sock
- 4 | # --image-repository=registry.aliyuncs.com/google_containers \ # 觉得慢加上这个

这个命令会将所需的镜像提前拉取下来, 然后再 init 就会快很多

2、初始化节点

有了 kubeadm , 就能一键初始化集群主节点了, 运行:

sh AI写代码

- 1 sudo kubeadm init \
- --apiserver-advertise-address=192.168.10.10 \
- --control-plane-endpoint=k8s-master \
- --kubernetes-version=v1.30.1 \
- 5 --service-cidr=10.50.0.0/16 \
- 6 --pod-network-cidr=10.60.0.0/16 \
- --cri-socket=unix:///run/containerd/containerd.sock
- # --image-repository=registry.aliyuncs.com/google_containers \ # 嫌慢的可以加上这句,用阿里云的镜像,我科学上网没试能不能用

apiserver-advertise-address 填主节点的IP地址

control-plane-endpoint , 还记得我们在 /etc/hosts 文件中配置的映射关系吗,填主节点的地址或者主机名

kubernetes-version 版本不多说

service-cidr 这是 Service 负载均衡的网络,就是你运行了一堆容器后有一个将它们统一对外暴露的地址,并且将对它们的请求统一收集并负载 节点,得为它配置一个网段

pod-network-cidr 每个 Pod 所在的网段

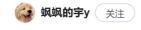
cri-socket 指定容器化环境

如果 init 失败, 而且失败的原因是没有连接上 api-server 的话, 使用命令查看 kubelet 的日志:

sh AI写代码

1 | journalctl -u kubelet -xe

如果其中有类似这样的错误,无法拉取的镜像叫 pause:3.8 的话 (新版本 v1.31.0 实测没有这个问题):













```
ourceVersion:,FieldPath:,},Reason:NodeHasSufficientMemory,Message:Node
Kind:Node,Namespace:,Name:k8s-master,UID:k8s-master,APIVersion:,ResourceVersion:,FieldPath:,},Reason:NodeHasSufficientMemory,Message:Node k8s-master status is now: NodeHasSufficientMemo west2-docker.pkg.dev/v2/k8s-artifacts-prod/images/pause/manifests/3.8\": dial tcp 142.250.157.82:443: i/o timeout"

-docker.pkg.dev/v2/k8s-artifacts-prod/images/pause/manifests/3.8\": dial tcp 142.250.157.82:443: i/o timeout" pod="kube-system/kube-scheduler-k8s-master"

2-docker.pkg.dev/v2/k8s-artifacts-prod/images/pause/manifests/3.8\": dial tcp 142.250.157.82:443: i/o timeout" pod="kube-system/kube-scheduler-k8s-master"

-docker.pkg.dev/v2/k8s-artifacts-prod/images/pause/manifests/3.8\": dial tcp 142.250.157.82:443: i/o timeout" pod="kube-system/kube-scheduler-k8s-master"
istry.k8s.io/pause:3.8\\\": failed to pull image \\\"registry.k8s.io/pause:3.8\\\": failed to pull and unpack image \\\"registry.k8s.io/pause:3.8\\\": failed to resolve reference \\\"re
   * 如果不是 pause:3.8 , 那就是镜像拉取失败, 可能是没有指定国内的源去国外下载失败了, 需要指定国内的源并提前拉取镜像;
   *还有可能是我们指定的源(比如阿里源)没有这个镜像,因为 k8s-v1.30.1 这个版本会默认使用 3.8 版本的沙箱,不知道什么原因拉取不下来
  所以只能拉取 3.9 下来改为 3.8:
                                                                                                                                                                     AI写代码
       ctr --namespace k8s.io image pull registry.aliyuncs.com/google_containers/pause:3.9
   2 ctr --namespace k8s.io image tag registry.aliyuncs.com/google_containers/pause:3.9 registry.k8s.io/pause:3.8
   init 失败后需要重置再重新 init , 执行:
                                                                                                                                                                     AI写代码
 sh
       sudo kubeadm reset # 重置 kubeadm , 执行这个后需要敲 y 回车
        sudo rm -rf /etc/cni/net.d # 删除上次 init 生成的文件
       sudo rm -rf /var/lib/etcd # 删除上次 init 生成的文件
  其他问题请参阅: 故障排查
  再次 init, 当然, 你也可以选择配置文件的方式, 和命令行的方式二选一:
 yaml
                                                                                                                                                                     AI写代码
       apiVersion: kubeadm.k8s.io/v1beta3
   1
       kind: ClusterConfiguration
   2
   3
        kubernetesVersion: v1.30.1
        controlPlaneEndpoint: "k8s-master:6443"
   5
        networking:
   6
          podSubnet: "10.100.2.0/24"
   7
          serviceSubnet: "10.100.1.0/24"
   8
       apiServer:
   9
          extraArgs:
                                                                                           展开 ~
  这个只需要 init 时指定配置文件就行:
                                                                                                                                                                     AI写代码
 sh
    1 | kubeadm init --config conf.yaml
   init 后成功的话会看到类似:
```

AI写代码

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 k8s-master:6443 --token is5atc.cc70psy934ptmb4j \

--discovery-to

2

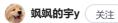
3 4

5

6

7

--control-plan















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

展开~

的东西,在这些命令之前还有几个命令,都执行一下,这是固定的,都这么执行:

sh AI写代码

- 1 mkdir -p \$HOME/.kube
- sudo cp -i /etc/kubernetes/admin.conf \$HOME/.kube/config
- 3 | sudo chown \$(id -u):\$(id -g) \$HOME/.kube/config

执行完这三条后我们就可以查看主节点上的 Pod 是否正常了:

AI写代码 sh

1 kubectl get pods -A # 记住, kubectl命令只能在主节点上执行, 其他节点上执行会被拒绝

会看到类似下图的,就是初始化主节点成功:

root@k8s-master:/home/user# kubectl get pods -A									
NAMESPACE	NAME	READY	STATUS	RESTAR	TS AGE				
kube-system	coredns-6f6b679f8f-b9qt2	0/1	Pending	0	4m40s				
kube-system	coredns-6f6b679f8f-mmrfm	0/1	Pending	0	4m40s				
kube-system	etcd-k8s-master	1/1	Running	1	4m47s				
kube-system	kube-apiserver-k8s-master	1/1	Running	1	4m47s				
kube-system	kube-controller-manager-k8s-master	1/1	Running	1	4m47s				
kube-system	kube-proxy-gjqkq	1/1	Running	0	4m41s				
kube-system	kube-scheduler-k8s-master	1/1	Running	1 cs	SDN @柳柳寫宇y				

还有, 我们执行完 init 后控制台打印的命令:

sh AI写代码

- kubeadm join k8s-master:6443 --token is5atc.cc70psy934ptmb4j \ 1
- 2 --discovery-token-ca-cert-hash sha256:cc01bdb1c2c0677ce9043af9f4996352320ae29b81c567a88c42f510f1817715
- 3 --control-plane

一个有 --control-plane ,一个没有,没有的那个是子节点运行的,一个子节点只要按照上面的步骤走到安装 kubelet 、 kubectl 、 kubeadm 后 -control-plane 的这部分命令就可以加入集群作为一个子节点,同样的,带 --control-plane 的是加入集群作为主节点,当真正的主节点挂后, 主节点就有可能成为主节点

3、注意的问题

*** 特别注意, apiserver-advertise-address 、 service-cidr 、 pod-network-cidr 三者的IP网段 不能重叠 不能重叠 不能重叠 ,不但三者之 叠,三者每个也不能与互联网上的地址重叠,不然会出问题,后两个一般用 10.x.x.x 网段,这个网段是留给内网的

关于IP地址的设定, 请参阅本文 第一大节 > 第7小节 > 第1节 IP地址固定以及网络相关知识

五、安装网络插件 Calico (重要, 只在主节点上做)

我们走到这步后还没有完成,因为集群只是在主节点上初始化了,其他机器要想加入集群,还得使用网络插件将它们连接起来,所以得安装一个网络 多个,选 Calico 就行。按照官网给的教程安装:

sh

VIE代证









AI写代码 sh

1 wget https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/tigera-operator.yaml

2 # 成者

3 curl -0 https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/tigera-operator.yaml

下载下来后一定不要用 kubectl apply -f 来执行,会报错:

AI写代码 sh

The CustomResourceDefinition "installations.operator.tigera.io" is invalid: metadata.annotations:

Too long: must have at most 262144 bytes

```
root@k8s-master:/hard_disk/gdyh/k8s/network# kubectl apply -f tigera-operator.yaml
namespace/tigera-operator unchanged
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
custom resource definition. a piextensions. k8s. io/felix configurations. crd. project calico.org\ created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/qlobalnetworksets.crd.projectcalico.org created
custom resource definition. a piextensions. k8s. io/host endpoints. crd. project calico.org\ created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org_created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
custom resource definition. a piextensions. k8s.io/ippools.crd.project calico.org\ created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
custom resource definition. a piextensions. k8s. io/kube controllers configurations. crd. project calico.org\ created
custom resource definition. a piextensions. k8s.io/network policies. crd. project calico.org\ created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org_created
customresourcedefinition.apiextensions.k8s.io/apiservers.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/imagesets.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/tigerastatuses.operator.tigera.io created
serviceaccount/tigera-operator configured
clusterrole.rbac.authorization.k8s.io/tigera-operator unchanged
clusterrolebinding.rbac.authorization.k8s.io/tigera-operator unchanged
deployment.apps/tigera-operator created
The CustomResourceDefinition "installations.operator.tigera.io" is invalid: metadata.annotations: Too long: must have at most
```

意思是 annotation 长度过长了,原因是 apply 和 create 的处理不同,这点 GitHub 上也有人在吐槽,这是 GitHub上的吐槽地址 改配置文件中这个选项的长度就不改了,我们不用 apply 使用 create:

sh

1 | kubectl create -f tigera-operator.yaml

没报错就没问题

第二步将配置文件下载下来,因为要改内容:

AI写代码

1 curl -0 https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/custom-resources.yaml

修改这个文件中的 192.168.0.0 为你刚才 init 时指定的 --pod-network-cidr:













```
# This section includes base Calico installation configuration.
# For more information, see: https://docs.tigera.io/calico/latest/reference/installation/api#operator.tigera.io/v1.In
apiVersion: operator.tigera.io/v1
kind: Installation
metadata:
      name: default
spec:
       # Configures Calico networking.
      calicoNetwork:
              ipPools:
                - name: default-ipv4-ippool
                    blockSize: 26
                  cidr: <u>10.60.0.0</u>/16
                      encapsulation: VXLANCrossSubnet
                      natOutgoing: Enabled
                      nodeSelector: all()
# This section configures the Calico API server.
 \textit{\# For more information, see: } \underline{\text{https://docs.tigera.io/calico/latest/reference/installation/api\#operator.tigera.io/v1.API.} \\ \underline{\text{https://docs.tigera.io/calico/latest/reference/installation/api\#operator.tigera.io/v1.API.} \\ \underline{\text{https://docs.tigera.io/calico/latest/reference/installation/api\#operator.tigera.io/v1.API.} \\ \underline{\text{https://docs.tigera.io/calico/latest/reference/installation/api\#operator.tigera.io/v1.API.} \\ \underline{\text{https://docs.tigera.io/calico/latest/reference/installation/api#operator.tigera.io/v1.API.} \\ \underline{\text{https://docs.tigera.io/calico/latest/reference
apiVersion: operator.tigera.io/v1
kind: APIServer
metadata:
      name: default
spec: {}
                                                                                                                                                                                                                                                                                                                                                                                                                               CSDN @
      原本是 192.168.0.0 改为你指定的IP地址
      改好后执行命令,这个文件可以用 apply 因为没有超限制(乐):
                                                                                                                                                                                                                                                                                                                                                                                                                                   AI写代码
```

1 | kubectl apply -f custom-resources.yaml

就会开始初始化网络插件,耐心等待,直到:

AI写代码 sh

1 kubectl get pods -A

显示的所有容器都 Running 就完成了:





```
root@k8s-master:/app/k8s/network# kubectl get pods -A
                                                                      READY
                                                                               STATUS
                                                                                         RESTARTS
NAMESPACE
                       NAME
                                                                                                         AGE
calico-apiserver
                       calico-apiserver-8d66bf649-7hqc5
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
calico-apiserver
                       calico-apiserver-8d66bf649-hmkfk
                                                                               Running
                                                                      1/1
                                                                                         1 (118m ago)
                                                                                                         19h
calico-system
                       calico-kube-controllers-77574cdd4d-fpx2z
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
calico-system
                       calico-node-49dfb
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
                                                                                         1 (118m ago)
                       calico-node-pm9rx
                                                                               Running
                                                                                                         16h
calico-system
                                                                      1/1
calico-system
                       calico-node-tccsg
                                                                      1/1
                                                                               Running
                                                                                           (118m ago)
                                                                                                         18h
                       calico-typha-6cc89f6c97-7pt97
calico-system
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
                       calico-typha-6cc89f6c97-m4rrz
                                                                                           (118m ago)
                                                                                                         16h
calico-system
                                                                               Running
                                                                               Running
                                                                                         2 (118m ago)
                       csi-node-driver-4t7tn
                                                                                                         18h
calico-system
calico-system
                       csi-node-driver-tksr8
                                                                               Running
                                                                                           (118m ago)
                                                                                                         16h
                       csi-node-driver-vpmrs
calico-system
                                                                               Running
                                                                                           (118m ago)
                                                                                                         19h
                       nginx-dep-649bd465f4-5nc7l
default
                                                                               Running
                                                                                        0
                                                                                                         115m
default
                       nginx-dep-649bd465f4-9slcl
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         16h
                       nginx-dep-649bd465f4-bsxkg
                                                                                                         16h
default
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
kube-system
                       coredns-7db6d8ff4d-clzqp
                                                                               Running
                                                                                           (118m ago)
                                                                                                         19h
                                                                      1/1
                       coredns-7db6d8ff4d-s4pg6
kube-system
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
                       etcd-k8s-master
kube-system
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
                       kube-apiserver-k8s-master
kube-system
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
                       kube-controller-manager-k8s-master
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         19h
kube-system
                                                                      1/1
kube-system
                       kube-proxy-p4hxv
                                                                               Running
                                                                                           (118m ago)
                                                                                                         19h
                       kube-proxy-qxct2
                                                                                                         16h
kube-system
                                                                      1/1
                                                                               Runnina
                                                                                         1 (118m ago)
kube-system
                       kube-proxy-zxmdc
                                                                      1/1
                                                                               Running
                                                                                         1 (118m ago)
                                                                                                         18h
                                                                               Running
                                                                                                         19h
kube-system
                       kube-scheduler-k8s-master
                                                                      1/1
                                                                                         1 (118m ago)
                       dashboard-metrics-scraper-795895d745-mssm8
                                                                                                         93m
kubernetes-dashboard
                                                                               Runnina
                                                                                         0
kubernetes-dashboard
                       kubernetes-dashboard-56cf4b97c5-9t42n
                                                                               Running
                                                                      1/1
                                                                                                         93m
                       tigera-operator-76ff79f7fd-771gp
tigera-operator
                                                                      1/1
                                                                               Running
                                                                                         1 (CSBM) a@ 姚 飒 郎 阵 v
```

六、其他节点加入集群 (可选)

1、其他节点加入集群

这是可选的,如果只有一个主节点,那么就要去除主节点上的污点,否则主节点上无法启动我们自己的 Pod。 去除污点参阅 本节的 第二小节

首先得保证这个节点能与主节点网络连通,然后执行本文目录中 一 、 二 和 三 的步骤,三个步骤一点都不要漏。

*** 需要注意的是,我们上面遇到的那个 沙箱 的问题, pause-3.9 、 pause-3.8 在这个新节点上也要手动拉(v1.31.0 版本实测没有这个问题), 前。

还记得我们之前 init 初始化主节点成功时得到的类似:

```
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 \ k8s-master: 6443 \ --token \ 9qk132.qldpmmgf4hpv371x \ \backslash \ + token \ (a) \ --token \ (b) \ --token \ (c) \ 
                                   --discovery-token-ca-cert-hash sha256:08f59184a092f52d114b114f89a4e05079f63985bf7e62<u>a68a3c254</u>aea
                                   --control-plane
Then you can join any number of worker nodes by running the following on each as root:
kubeadm join k8s-master:6443 --token 9qk132.qldpmmgf4hpv37lx \
                                    --discovery-token-ca-cert-hash sha256:08f59184a092f52d114b114f89a4e05079f63985bf7e62a68a3625A4kæ
```

的字符串吗? 复制下面那串:

sh

1 kubeadm join k8s-master:6443 --token 9qk132.qldpmmgf4hpv37lx \ --discovery-tc'-- -- cont book \ 2 3













到要加入集群的这个节点中去执行,但是前提是 一 、二 、三 中的步骤你都完全执行完了并且没有报错。

这个命令中有一个 --token , 它是会过期的, 过期时间好像是 24小时 , 如果 token 过期了, 执行:

sh Al写代码

1 kubeadm token create

生成新的 token, 替换命令中的 --token。

此外, kubeadm join 命令后面的 --discovery-token-ca-cert-hash 如果你也没记下来的话,可以从主节点的CA证书中提取哈希值,执行:

sh Al写代码

```
l openssl x509 <mark>-pubkey -in</mark> /etc/kubernetes/pki/ca.crt | openssl rsa <mark>-pubin -outform</mark> der <mark>2></mark>/dev/null \
2  | sha256sum | awk '{print $1}'
```

同样的,提取到的哈希值替换命令中 --discovery-token-ca-cert-hash 的值就行。

工作节点加入集群成功类似这样:

```
root@k8s-worker01:/gdyh_app/k8s/runtime# kubeadm join k8s-master:6443 --token 9qk132.qldpmmgf4hpv37lx \
         -discovery-token-ca-cert-hash sha256:08f59184a092f52d114b114f89a4e05079f63985bf7e62a68a3c254aea8
[preflight] Running pre-flight checks
[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 y
[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-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.143527ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap
This node has joined the cluster:
\star 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.
                                                                                                  CSDN @郊
```

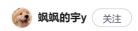
执行完后这里的成功其实还不算完全成功,要等它 init 完成才算,你只需要在主节点上盯着所有的 Pod 都 Running 了后就可以继续下一步了。

2、主节点成为工作节点 worker node (可选)

如果你只想在本机上运行所有的 Pod ,那么只需要将主节点配置为工作节点,以便它可以调度并运行工作负载。在主节点上启用调度器,一般是移脉污点(taint),这些污点会阻止调度器将工作负载调度到主节点。执行命令查看都有哪些污点:

sh AI写代码

然后再删除这个污点:









AI写代码

1 | kubectl taint nodes k8s-master node-role.kubernetes.io/control-plane-

现在主节点上也会被部署 Pod 了。

3、如果节点初始化失败

如果这个节点初始化失败,需要重置集群中关于这个节点的东西:

AI写代码 sh

- 1 kubectl drain <节点名称> --ignore-daemonsets --delete-emptydir-data # 驱逐节点上的所有 Pod
- 2 kubectl delete node <节点名称> # 从集群中删除节点

然后到节点上执行:

sh AI写代码

```
1
   sudo kubeadm reset # 执行后按 y
2
3
   sudo rm -rf /etc/cni/net.d # 移除 CNI 配置
4
5
   # 清除 iptables 规则
6
  sudo iptables -F
   sudo iptables -t nat -F
   sudo iptables -t mangle -F
9 | sudo iptables -t raw -F
```

展开 ~

重置后再根据情况重新初始化

======== 以下对整个集群的操作均在主节点上

七、安装 Kubernetes Dashboard 前端控制面板 (可选)

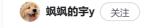
一直使用命令行手都敲累了,这时候你可以选择安装一个前端可视化页面来控制整个集群。

首先你得安装官方提供的前端页面,下载这个配置文件,你可以将链接中的 v2.7.0 替换成你想要的版本:

AI写代码

 $1 \quad \text{curl -0 https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.yamlusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.yamlusercontent.y$

需要修改差不多二三十行处的









```
kind: Service
 apiVersion: v1
 metadata:
   labels:
    k8s-app: kubernetes-dashboard
   name: kubernetes-dashboard
   namespace: kubernetes-dashboard
 spec:
  type: NodePort 加上这个
   ports:
     - port: 443
      targetPort: 8443
   selector:
    k8s-app: kubernetes-dashboard
                    CSDN @飒飒的宇y
再执行:
                                                                                                     AI写代码
 1 kubectl apply -f recommended.yaml
然后就是等待它结束后创建一个配置,用于配置账户和获取token以登录页面,一定要等上面的结束再接着(结束标志是所有的 Pod 都 Running):
                                                                                                     AI写代码
 1 | vi admin.yaml
添加以下内容:
yaml
                                                                                                     AI写代码
 1 | apiVersion: v1
 2
   kind: ServiceAccount
 3 metadata:
 4
    name: admin-user
 5
    namespace: kubernetes-dashboard
 6
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRoleBinding
 9 metadata:
                                                       展开 ٧
然后再执行:
                                                                                                     AI写代码
 1 kubectl apply -f k8s-admin-user.yaml # 这句是创建账户
做到这步后网上的大多数帖子都是让执行:
                                                                                                     AI写代码
 1 | kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{|
他们都能生成Token, 但是我的不行我的执行了这句代码后出来的是这样的:
```

 sh Al写代码

root@k8s-master:/app/k8s/dashboard# kubectl -n kubernetes-dashboard describe secret \

1

```
2
     $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')
  3
     Name:
                  kubernetes-dashboard-certs
  4
                  kubernetes-dashboard
     Namespace:
  5
     Labels:
                  k8s-app=kubernetes-dashboard
  6
     Annotations: <none>
  7
  8
     Type: Opaque
  9
                                                              展开 ٧
  并没有 Token ,可能是由于 API 服务器还没有为我们创建的账户创建默认的 Token ,这时候需要自己手动生成密钥,新建一个 yaml 文件,加入:
                                                                                                                AI写代码
 yaml
     apiVersion: v1
     kind: Secret
  3
     metadata:
  4
      name: admin-user-token
  5
       namespace: kubernetes-dashboard
  6
       annotations:
  7
        kubernetes.io/service-account.name: admin-user # 如果上面的用户名你自定义了,记得替换这里
  8 \mid \mathsf{type: kubernetes.io/service-account-token}
  创建密钥:
                                                                                                                AI写代码
  1 | kubectl apply -f admin-user-secret.yaml
  检查一下生成没有:
 sh
                                                                                                                AI写代码
  1 | kubectl -n kubernetes-dashboard get secret | grep admin-user
root@kos-master:/app/kos/dashboard
root@k8s-master:/app/k8s/dashboard#
root@k8s-master:/app/k8s/dashboard# kubectl apply -f account-secret.yaml
secret/admin-user-token created
root@k8s-master:/app/k8s/dashboard# kubectl -n kubernetes-dashboard get secret | grep admin-user
     -user-token
                                  kubernetes.io/service-account-token
root@k8s-master:/app/k8s/dashboard#
                                                                                CSDN @飒飒的宇y
root@k8s-master:/app/k8s/dashboard#
  发现有,这时候我们再查看 Token:
                                                                                                                AI写代码
 sh
  1 \mid \text{kubectl} \mid \text{-n} \mid \text{kubernetes-dashboard describe secret token}的名字,这里我们的是admin-user-token
                             《 飒飒的宇y ( 关注 )
                                                                                                  146 108
                                                                                                                  < 分
```

root@k8s-master:/app/k8s/dashboard# kubectl -n kubernetes-dashboard describe secret admin-user-token

Name: admin-user-token
Namespace: kubernetes-dashboard

Labels: <none>

Annotations: kubernetes.io/service-account.name: admin-user

kubernetes.io/service-account.uid: f6f74210-f4f1-4b38-9e47-3c1682a18805

Type: kubernetes.io/service-account-token

Data

token: eyJhbGci0iJSUzI1NiIsImtpZCI6IkEwM1EzclFDaWVZazBzNHkyWWIwcjBRcDVyWE5qVDV4SUJkbTdYNUVKaXMifQ.eyJpc3MiD:
dWJlcm5ldGVzLWRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUi0iJhZG1pbi11c2VyLXRva2VuIiw ia
ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQi0iJmNmY3NDIxMC1mNGYxLTRiMzgtOWU0Ny0zYzE20DJhMTg4MDUiLCJ;
26kUZNc1BEKp2oTFD00Jha6Se0CSMc8keoB_AR74NsQJTAbUEhm2WnXRF7eCT4vmIZ_mW2Bfhr0SpKK9L4jf6xmdQlQITMk-_4gHAgviPdaySwGJ0
kzn1PJhzTi0sprnbJG82Ztza2CFcPCjI9x27CcTCr5DOSfS4qt6Z4alujx1No0AfLes4j9fR1RRPxHUd2k0ZZhle0_nyffu8Q0hcgzK8zxd6_Bwol

ca.crt: 1107 bytes namespace: 20 bytes

CSDN @飒飒的宇\

生成了,是不是有点像 RSA 密钥?复制这个 Token 保存一下,以后要用就直接找文件了。

查看前端页面暴露出去的端口是多少:

sh AI写代码

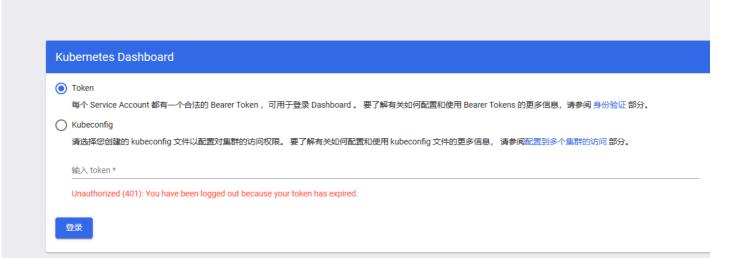
1 kubectl get svc -n kubernetes-dashboard

root@k8s-master:/app/k8s/dashboard# root@k8s-master:/app/k8s/dashboard# root@k8s-master:/app/k8s/dashboard# root@k8s-master:/app/k8s/dashboard# kubectl get svc -n kubernetes-dashboard NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE ClusterIP 8000/TCP 109m dashboard-metrics-scraper 10.50.195.141 kubernetes-dashboard NodePort 443:32713/TCP 109m 10.50.247.162 root@k8s-master:/app/k8s/dashboard# root@k8s-master:/app/k8s/dashboard# root@k8s-master:/app/k8s/dashboard# root@k8s-master:/app/k8s/dashboard# CSDN @飒飒的宇y root@k8s-master:/app/k8s/dashboard#

很明显,我的这个端口是 32713 ,网页访问任意一台节点的 https://IP:端口 ,就能打开页面,记住,必须是 https 协议哦。 打开页面有警告不管,点击高级继续访问,你会看到:







还记得之前复制的那个像 RSA 密钥的token吗? 粘贴进去就能登录了。

如果上面的步骤还是存在问题,可以删除并重新创建 ServiceAccount 和 ClusterRoleBinding:

AI写代码 sh

```
kubectl delete serviceaccount admin-user -n kubernetes-dashboard
   kubectl delete clusterrolebinding admin-user
3
  kubectl apply -f - <<EOF</pre>
4
5
   apiVersion: v1
6
   kind: ServiceAccount
7
   metadata:
8
    name: admin-user
9 namespace: kubernetes-dashboard
```

展开 ٧

等待几分钟, 然后再次检查生成的 Secret。

八、部署 ingress 接收外部流量转发到 service (可选)

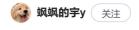
1、简介

大白话就是: 在此之前我们都是直接访问 service , 让 service 负载均衡到 Pod 上, 优点是直接, 缺点是随着 service 的增多端口会越来越多 于是我们在 service 之上再套一层,统一管理众多的 service

流量流向是: 流量 --> ingress --> service --> pod

2、部署 ingress-nginx

首先得明确 ingress 其实也是一个 service, 它接收外部的流量转发到配置好的指定了的 service, 所以当我们部署 ingress-nginx 时会发现经 关于 ingress 的 service;













k8s 的 ingress 实现有很多个,就不一一列举了,大家都用 ingress-nginx;

执行语句,从 GitHub 上拉取 yaml 开始配置 (需要其他版本的改 v1.10.1, 但是得注意兼容情况):

AI写代码

1 curl -0 https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.1/deploy/static/provider/cloud/deploy/static/p

修改 kind: ConfigMap 下面 kind: Service 处的两个地方:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
    app.kubernetes.io/version: 1.10.1
  name: ingress-nginx-controller
 namespace: ingress-nginx
spec:
  externalTrafficPolicy: Local
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
   appProtocol: http
    name: http
    port: 80
   protocol: TCP
    targetPort: http
    appProtocol: https
    name: https
    protocol: TCP
    targetPort: https
  selector:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    ann_kubernetes_io/name: ingress-nginx
  type: LoadBalancer
                                    CSDN @飒飒的宇v
```

*** 上面的从 Local 改为 Cluster ,如果是 Local ,那么只有到达节点的流量才会被处理,而非自己节点的流量没有反应,意思就是说如果你的 192.168.10.10 , 恰好 ingress-nginx 又没有部署在主节点上, 那么你想通过 192.168.10.10 访问 ingress 就不行, 改为 Cluster 的话, 你 何一台机器上用 IP:port 访问 ingress 都可以,Cluster 的好处是不管从哪都能访问,缺点是会增加集群内部的流量,因为你的请求流量会在集 发,而且你的源IP地址也会变化,因为当你访问到一个没有部署 ingress 的节点上时,它会被路由到有 ingress 的节点上,所以内部流量才会增 变

*** 下面的 LoadBalancer 改为 NodePort , LoadBalancer 是给云服务器或者自己弄了负载均衡的用的

应用配置文件:

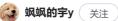
AI写代码

1 kubectl apply -f deploy.yaml

查看 ingress 暴露的端口:

AI写代码 sh













找一个类似这样的:

calico-system	calico-typha	ClusterIP	10.50.37.18	<none></none>	5473/TCP
default	kubernetes	ClusterIP	10.50.0.1		443/TCP
default	my-nginx	NodePort	10.50.6.71	<none></none>	80:30100/TCP
ingress-nginx	ingress-nginx-controller	NodePort	10.50.106.222		80:32536/TCP,443:30779/TC
ingress-nginx	ingress-nginx-controller-admission	ClusterIP	10.50.106.127	<none></none>	443/TCP
kube-system	kube-dns	ClusterIP	10.50.0.10		53/UDP,53/TCP,9153/TCP
kubernetes-dashboard	dashboard-metrics-scraper	ClusterIP	10.50.207.181		8000/TCP CSDN
kubernetes-dashboard	kubernetes-dashboard	NodePort	10.50.110.161	<none></none>	443:32757/TCP

我这里就是 32536 , 配置一个 ingress 资源, 编辑一个 yaml 加入:

AI写代码 yaml apiVersion: networking.k8s.io/v1 1

2 kind: Ingress 3 metadata: 4 name: my-ingress 5 namespace: default 6 spec: 7 rules: 8 - host: mynginx.com 9 http:

展开 🗸

这个配置文件的作用是创建一个 Ingress , 并且将 mynginx.com 的流量都转发到 my-nginx 这个 service 的 80 端口中去, 在win机上编辑 host (C:\Windows\System32\drivers\etc),加入 192.168.10.10 mynginx.com,浏览器访问 http://mynginx.com:32536 就能访问到我们部署的 Pod了。

不想这么麻烦的也可以在集群外搭建一个 Metall B 实现负载均衡上面的步骤就都不用了,会给你生成一个能直接访问的集群外网地址, EXTERNAL-<none> 或 <pending> 了。

root@k8s-master:~# root@k8s-master:∼# kgs NAMESPACE NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) calico-apiserver calico-api ClusterIP 10.50.222.208 443/TCP calico-kube-controllers-metrics calico-system ClusterIP 9094/TCP 10.50.37.18 calico-system calico-typha ClusterIP 5473/TCP 10.50.0.1 443/TCP default kubernetes ClusterIP NodePort 10.50.6.71 80:30100/TCP default mv-nainx 10.50.106.222 ingress-nginx-controller NodePort 80:32536/TCP,443:30779/TC ingress-nginx ingress-nginx ingress-nginx-controller-admission ClusterIP 10.50.106.127 443/TCP 10.50.0.10 53/UDP, 53/TCP, 9153/TCP kube-system kube-dns ClusterIP 10.50.207.181 kubernetes-dashboard dashboard-metrics-scraper 8000/TCP ClusterIP 10.50.110.161 443:32757/TCP kubernetes-dashboard kubernetes-dashboard NodePort root@k8s-master:~# kgi NAMESPACE NAME CLASS **HOSTS ADDRESS PORTS** AGE nainx

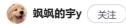
图里的命令是我取的别名,不是正规命令

九、结尾的话

全部内容大概就是这样,给个免责声明吧:以上仅代表我个人观点,不可能全对,也可能有错的地方,如果后续有错了我会回来改正。后续不定时更 的内容。

参考文献

kubernetes官方文档 在Ubuntu22.04 LTS上搭建Kubernetes集群 k8s1.24+ dashboard不能自动生成token的问题









腾讯技术创作特训营 6.29 深圳特别场

极客创作茶话会:如何通过技术文章和知识号运营打造个人影响力?

Ubuntu22.04基于Calico+Containerd部署v1.28.2 最新发布 以下是。

Q2200439

108 条评论

m0_75116040 热评 大佬, 自己手动下了缺少, 但官方镜像里面缺少csi的镜像导致master节点起不来notready。该怎么办? calico-system csi-...

ubuntu24.04安装Kubernetes1.31.0(k8s1.30.0)高可用集群

文章浏览阅读1.9k次,点赞18次,收藏20次。首先,在负载均衡器节点上,停止haproxy和keepalived服务,负载均衡器节点(hep-kubernetes-apiserver-lb-prd-01 和 hep-kubernetes-

Ubuntu上手动安装Kubernetes ubuntu24.04 安装kubernetes

文章浏览阅读9.5k次,点赞3次,收藏7次。背景两台Ubuntu16.04服务器:ip分别为192.168.56.160和192.168.56.161。。Kubernetes版本:1.5.5 Docker版本:1.12.6 etcd版本:2.2

ubuntu基于sealos搭建k8s集群, helm3安装配置自动化扩容Prometheus, grafana出图展示,以及动态web搭建

2302 81156108

Dances with Cloud

原创不易,关注支持!运维技术的学习支持请关注公众号tools0!

使用kubeadm基于ubuntu20.04部署K8S v1.29集群

CKA考试基础练习环境配置

在ubuntu16.4 安装Kubernetes1.9_kubectl unbuntu16

文章浏览阅读1w次,点赞2次,收藏9次。本文详细介绍如何在Ubuntu系统上安装和配置Kubernetes集群,包括安装Docker、Kubernetes组件及网络插件Flannel,并通过实例展示效

k8sv1.30安装教程基于docker

文章浏览阅读1.7k次,点赞20次,收藏15次。基于docker的最新版k8s安装

Ubuntu22.04部署K8s集群 夜半一碗凉泡面

K8s集群 — Win11 + VM Ware17.0 + Ubuntu 22.04 + K8s 1.27 + docker 24.0 + cri-docker 0.3.4.3 + flannel

问题分析 错误提示 仓库 "https://apt.kubernetes.io kubernetes-xenial Release" 没有 Release 文件

独坐一隅,凝神遐想,是种幸福! -

问题分析 错误提示 仓库 "https://apt.kubernetes.io kubernetes-xenial Release" 没有 Release 文件

在Ubuntu 24.04 LTS (Noble Numbat)上搭建Kubernetes1.30集群

文章浏览阅读2.5k次,点赞45次,收藏28次。准备工作系统准备主机和IP地址准备编辑<mark>安装</mark>步骤安装前准备关闭防火墙设置服务器时区关闭 swap 分区关闭SELinux配置hosts配法

...ubuntu 22.04 使用kubeadm 安装kubernetes 1.30 docker

Ubuntu 22.04 部署 Kubernetes v1.30

Ubuntu 22.04 部署 Kubernetes v1.30

【Ubuntu22.04配置k8s集群】

goislai

田记寨码

为了防止机器之间的请求被防火墙拦截,需要把3台机器的防火墙都关了(这里是为了省事,测试环境中使用,上线时请开放对应所需端口),另外我们要把iptables也关了。这个只

ubuntu安装k8s1.30 containerd 1.7.17

文章浏览阅读930次,点赞9次,收藏5次。查看最新版本修改下行的版本号(1.7.17)下载即可:(下载的是 cri-containerd-XXX-linux-amd64.tar.gz)_containerd 1.7.17

基于ubuntu containerd 部署kubernetes v1.30.3

文章浏览阅读1k次,点赞23次,收藏8次。基于ubuntu containerd 部署kubernetes v1.30.3软件版本ubuntu 22.04 LTScontainerd 1.7.17kubernetes v1.30.1一,基础配置1.1配置hc

Ubuntu22.04搭建k8s集群,看这一篇就够啦!

m0_43445928

本文适合k8s初学者,跟着文章步骤走即可,一般不会出错。

基于ubuntu 22.04 安装k8s一主一从集群 (包括dashboard)

zxw824442924

照着命令输就完事了,遇到问题就查一查,逢山开路遇水搭桥。

Ubuntu 22.04安装K8S集群

brig

【代码】Ubuntu 22.04安装K8S集群。

ubuntu22.04使用kubeadm部署k8s集群

BY_xiaopeng

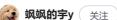
ubuntu22.04使用kubeadm部署一个k8s集群,1个master+2个worker节点。

【VirtualBox中Ubuntu-22.04下用国内源安装minikube1.33.1】

Q59512884°

还记得上面我们已经<mark>安装</mark>了kubectl、kubeadm吗,此时我们只需要将<mark>安装</mark>好的命令链接到minikube目录就好了,从报错信息我们知道了要链接的目录是,你已经<mark>安装</mark>好了一个

Ubuntu22.04安装k8s 1.27.+Dosht















温馨提示请仔细阅读: ♥♥♥♥♥♥♥♥♥♥此教程为k8s当前官方最新版1.27.1集群搭建教程(一切基于k8s官方文档进行搭建, 若遇到教程以外问题请仔细参考官方文档,

ubuntu22.04 kubeadm安装k8s集群(从零到有)

qq_44637753

k8s 三master 三nodekubeadm版本要求。

Ubuntu22.04 安装k8s集群 v1.27

ubuntu22.04安装k8s v1.27

qq_45744253

ubuntu22.04安装k8s集群(cri-docker+haproxy+keepalived)

使用haproxy和keeplived搭建高可用的多主K8s集群

SeeYouGoodBye

【代码】ubuntu22.04安装k8s1.26.9。

ubuntu22.04安装k8s1.26.9

new ac

Ubuntu安装k8s

前提集群已经搭好了,而且都已经安装好docker。 注意: 红色为必做操作 蓝色为选做操作 安装依赖工具 一下脚本都要以root用户运行 更新源: apt-get update &&an

ubuntu22.04安装k8s

在Ubuntu 22.04上安装Kubernetes (k8s) 需要几个步骤,因为Kubernetes是一个复杂的服务,通常涉及系统配置、依赖包安装以及集群部署。以下是简化的安装流程: 1.**更

关于我们 招贤纳士 商务合作 寻求报道 ☎ 400-660-0108 窗 kefu@csdn.net 👨 在线客服 工作时间 8:30-22:00 公安备案号11010502030143 京ICP备19004658号 京网文 [2020] 1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照 ©1999-2025北京创新乐知网络技术有限公司





