实验二 基于Kubernetes的高可用Redis数据库

22121630 汪江豪

- 先从腾讯云上租借一台服务器，并到Docker官网，输入以下命令安装Docker到ubuntu系统中。

```
# 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/ubuntu/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/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

查看docker版本号列表

```
apt-cache madison docker-ce
```

```
ubuntu@slave1:~$ apt-cache madison docker-ce
 docker-ce | 5:28.0.4-1~ubuntu.20.04~focal | https://download.docker.com/linux/ubunt
u focal/stable amd64 Packages
 docker-ce | 5:28.0.3-1~ubuntu.20.04~focal | https://download.docker.com/linux/ubunt
u focal/stable amd64 Packages
 docker-ce | 5:28.0.2-1~ubuntu.20.04~focal | https://download.docker.com/linux/ubunt
u focal/stable amd64 Packages
 docker-ce | 5:28.0.1-1~ubuntu.20.04~focal | https://download.docker.com/linux/ubunt
u focal/stable amd64 Packages
 docker-ce | 5:28.0.0-1~ubuntu.20.04~focal | https://download.docker.com/linux/ubunt
u focal/stable amd64 Packages
 docker-ce | 5:27.5.1-1~ubuntu.20.04~focal | https://download.docker.com/linux/ubunt
u focal/stable amd64 Packages
```

运行示例程序hello-world

```
ubuntu@VM-0-12-ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

成功输出，说明Docker安装成功。

- 安装Kubernetes
  - 先将镜像换源成阿里后，查看可安装的版本号,然后安装相应版本（注意尽量安装Docker和K8s相同版本）

```
sudo apt-cache madison kubeadm
```

```
ubuntu@slave1:~$ sudo apt-cache madison kubeadm
   kubeadm |  1.28.2-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenia
l/main amd64 Packages
   kubeadm |  1.28.1-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenia
l/main amd64 Packages
   kubeadm |  1.28.0-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenia
l/main amd64 Packages
   kubeadm |  1.27.6-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenia
l/main amd64 Packages
   kubeadm |  1.27.5-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenia
l/main amd64 Packages
```

- 通过openssh-server创建公钥密钥，添加到authorized_keys后，保存该镜像。
- 从腾讯云上再创建2个该镜像服务器，并通过ssh设置3台节点的无密码连接，设置主机名分别为master,slave1,slave2。
- 在master节点上启动k8s集群：

```
# 关闭虚拟内存
sudo swapoff -a

# 初始化单节点集群（address切换成自己的内网ip）
sudo kubeadm init --apiserver-advertise-address=172.29.0.8 --image-repository
registry.aliyuncs.com/google_containers --service-cidr=10.96.0.0/12  --pod-
network-cidr=10.244.0.0/16
```

```
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

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 172.29.0.8:6443 --token v9ulm9.p75raq7i7wzlb0is \
    --discovery-token-ca-cert-hash sha256:c8083f4039f4c203087bc0de1fbb00f765bfbce9b385217aa131c0c14309a1b6
ubuntu@master:~/.ssh$ cd ..
```

看到生成的随机token表示启动成功。

- 到slave1和slave2节点先关闭虚拟内存，再将上图最后两行命令前加上sudo，运行。让这两个节点加入k8s集群。
- 回到master节点，配置kubectl客户端的访问权限，下载网络插件，并列出集群所有命名空间下的pod

```
# 配置 kubectl 的客户端访问权限
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# 下载网络插件
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml


```

```
ubuntu@master:~$ kubectl get pods --all-namespaces
NAMESPACE      NAME                              READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-96fp7             1/1     Running   0          53s
kube-flannel   kube-flannel-ds-dw9c2             1/1     Running   0          53s
kube-flannel   kube-flannel-ds-r2s6r             1/1     Running   0          53s
kube-system    coredns-7ff77c879f-6bnts          1/1     Running   0          7m18s
kube-system    coredns-7ff77c879f-q5s8x          1/1     Running   0          7m18s
kube-system    etcd-master                       1/1     Running   0          7m27s
kube-system    kube-apiserver-master             1/1     Running   0          7m27s
kube-system    kube-controller-manager-master    1/1     Running   0          7m27s
kube-system    kube-proxy-2bp2v                  1/1     Running   0          7m18s
kube-system    kube-proxy-ptvdr                  1/1     Running   0          5m8s
kube-system    kube-proxy-vcv95                  1/1     Running   0          4m55s
kube-system    kube-scheduler-master             1/1     Running   0          7m27s
```

STATUS一栏全部显示Running，表示成功，随后开始部署redis。

- 创建4个文件redis-follower-deployment.yaml，redis-leader-deployment.yaml，redis-follower-service.yaml，redis-leader-service.yaml，并输入:

```
kubectl apply -f [文件名]
```

应用这4个文件。

- 创建前端配置文件frontend-deployment.yaml,frontend-service.yaml并同样应用，

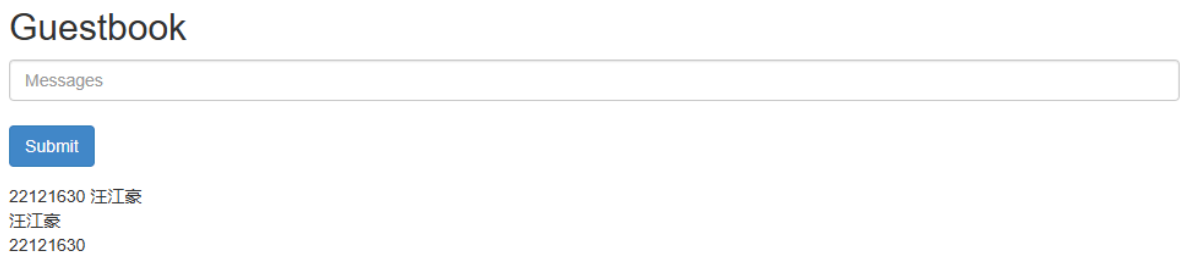
```
kubectl get services
```

随后查看正在运行的服务：

```
ubuntu@master:~$ kubectl get services
NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
frontend         NodePort    10.109.133.243   <none>        80:32309/TCP   10s
kubernetes       ClusterIP   10.96.0.1        <none>        443/TCP        4m40s
redis-follower   ClusterIP   10.99.26.120     <none>        6379/TCP       23s
redis-leader     ClusterIP   10.107.0.88      <none>        6379/TCP       34s
```

可以看到，前端页面暴露在32309端口。

- 随后在腾讯云上，修改3台结点的安全组入站规则，暴露TCP:32309端口，允许所有地址入站。
- 打开本地浏览器，输入三台结点任意一台的公网IP地址，加上:32309，跳转到前端页面。输入信息并提交，成功存储。



至此，基于基于Kubernetes的高可用Redis数据库及其前端页面搭建完成，实验成功。