链滴

kubernetes 网络简介（上）

作者：Leif160519
原文链接：https://ld246.com/article/1592043040204
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20190626.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h2 id="一-Service">一、Service</h2>
<h3 id="1-1-Service存在的意义-">1.1 Service 存在的意义：</h3>

防止 Pod 失联（服务发现）
定义一组 Pod 的访问策略（负载均衡）

<h3 id="1-2-Pod与Service的关系">1.2 Pod 与 Service 的关系</h3>

通过 label-selector 相关联
通过 Servic 实现 Pod 的负载均衡（ TCP/UDP 4 层）

创建 service 的时候必须打标签，并且与创建的 deployment 或者 pod 的标签一致
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2020/06/image-08ea8dab.png?imageView2/2/interlace/1/format/jpg"></
>
通过 <code>kubectl create deployment web --image=nginx --dry-run=client -o yaml &gt
web-dp.yaml</code> 命令导出 deployment 文件：
yaml 示例：
<pre><code class="highlight-chroma"><span class="highlight
cl">apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web
deployment的标签
  name: web
spec:
  replicas: 3
  selector:
标签选择器
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
Pod的标签
        project: blog
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
</code></pre>
<blockquote>
注意：

标签选择器里的标签是筛选 Pod 用的
Pod 的标签支持多个

</blockquote>
应用：

<pre><code class="highlight-chroma">kubectl apply -f web-dp.yaml
</code></pre>
查看标签：
<pre><code class="highlight-chroma">[root@k8s-master k8s]# kubectl get pod --show-labels
NAME
READY STATUS   RESTARTS AGE LABELS
web-5b9bff6674-wjlq 1/1   Running 0   12s app=web,pod-template-hash=5b9bff6674,project=blog
web-5b9bff6674-9ltd 1/1   Running 0   33s app=web,pod-template-hash=5b9bff6674,project=blog
web-5b9bff6674-kgsj 1/1   Running 0   15s app=web,pod-template-hash=5b9bff6674,project=blog
</code></pre>
暴露服务：
<pre><code class="highlight-chroma">kubectl expose deployment web --port=80 --target-port=80 --name=web --dry-run=client -o yaml > web-svc.yaml
</code></pre>
<blockquote>
参数解释：

<code>--port</code>：k8s 集群内部访问端口
<code>--target-port</code>：容器中服务提供端口，即应用程序端口，如 nginx 提供 80，ysql 提供 3306
<code>--protocol</code>：指定协议类型，如 TCP、UDP，SCTP 等
<code>--name</code>：给 svc 起名，一般 svc 的名称与 deployment 一致

</blockquote>
yaml 示例：
<pre><code class="highlight-chroma">
```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: web
  name: web
spec:
  ports:
  - port: 80          # 内部访问端口
    protocol: TCP
    targetPort: 80    # 容器提供服务的端口
  selector:
    app: web          # 标签与deployment中Pod定义的标签一致
    project: blog
```
</code></pre>
应用：
<pre><code class="highlight-chroma">
```
kubectl apply -f web-svc.yaml
```
</code></pre>

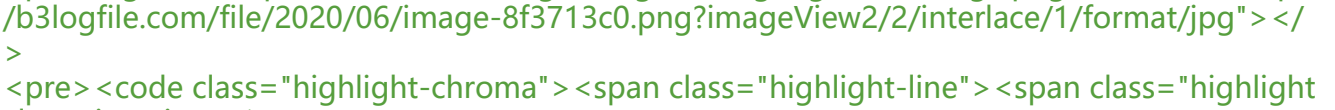
查看 svc 标签：
<pre><code class="highlight-chroma">[root@k8s-master k8s]# kubectl get svc --show-labels
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE LABELS
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 47m component=apiserver,provider=kubernets
web ClusterIP 10.101.205.162 <none> 80/TCP 18m app=web
</code></pre>
<blockquote>
注意：从中可以发现，web 这个 Service 的标签中并没有 Project=blog 这个标签，这是因为 svc 中 app=web 标签是 svc 本身的，它不是用于去关联 Pod 的，svc 的详细信息可以使用 <code>kubctl get svc -o wide</code>:
</blockquote>
<pre><code class="highlight-chroma">

[root@k8s-master k8s]# kubectl get svc -o wide

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 47m <none>

web ClusterIP 10.101.205.162 <none> 80/TCP 18m app=web,project=blog
</code></pre>
正常来讲，在任何节点机器上 <code>curl 10.101.205.162 </code> 是可以访问到的,而且 svc 中的 IP 地址是非常稳定的，只要这个 svc 资源不被删除，这个 IP 就会一直存在。
<h3 id="1-3-Service三种常用类型">1.3 Service 三种常用类型</h3>
<h4 id="1-3-1--集群内部使用">1.3.1 <code>ClusterIP</code>：集群内部使用</h4>
expose 的默认类型，为一组 pod 分配一个稳定的虚拟 IP，作为这组 Pod 提供统一入口，集群外无法问问，只能在集群内部访问
(同 Namespace 内的 Pod)


<h4 id="1-3-2---对外暴露应用">1.3.2 <code>NodePort</code>：对外暴露应用</h4>
在每个节点上启用一个端口来暴露服务，可以在集群外部访问。也会分配一个稳定内部集群 IP 址。访问地址：<code><NodeIP>:<NodePort></code>


<pre><code class="highlight-chroma">

apiVersion: v1

kind: Service

metadata:

  labels:

    app: web

  name: web

spec:

  ports:

  - port: 80

    protocol: TCP

    targetPort: 80

  selector:

  app: web
  project: blog
 type: NodePort
</code></pre>
<blockquote>
注意：<code>type</code> 与 <code>ports</code> 同级
</blockquote>
<pre><code class="highlight-chroma">[root@k8s-master k8s]# kubectl get svc
NAME   TYPE CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1   <none>   443/TCP   61m
web   NodePort   10.101.205.162   <none>   80:31495/TCP   11m
</code></pre>
其中，<code>31495</code> 为宿主机端口,每个节点都会监听这个端口，而且这个端口是 <code>kube-proxy</code> 创建的
<pre><code class="highlight-chroma">[root@k8s-master k8s]# netstat -antp | grep 31495
tcp   0   0 0.0.0.0:31495   0.0.0.0:*   LISTEN   2581/kube-proxy</code></pre>
当然，这个宿主机端口号也可以固定，写法如下（端口号范围：<code>3000</code>-<code>32767</code>）：
<pre><code class="highlight-chroma">apiVersion: v1
kind: Service
metadata:
  labels:
    app: web
  name: web
spec:
  ports:
  - port: 80 # 集群内部端口
    protocol: TCP # 协议
    targetPort: 80 # 容器端口
    nodePort: 3000 # 节点端口
  selector: # 标签选择器，关联对应Pod
    app: web
    project: blog
  type: NodePort # 指定类型</code></pre>
<pre><code class="highlight-chroma">[root@k8s-master k8s]# kubectl get svc
NAME   TYPE CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE</code></pre>

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 74m
web NodePort 10.101.205.162 <none> 80:30000/TCP 23m

[root@k8s-master k8s]# netstat -antp | grep 3000
tcp 0 0 0.0.0.0:30000 0.0.0.0:* LISTEN 2581/kube-proxy
</code></pre>
<blockquote>
补充：<code>kube-proxy</code>：实现 Service 的功能，包含服务发现和提供负载均衡的力。
</blockquote>
<h4 id="1-3-3--对外暴露应用-适用公有云">1.3.3 <code>LoadBalancer</code>：对外暴露应，适用公有云</h4>
与 <code>NodePort</code> 类似，在每个节点上启用一个端口来暴露服务。除此之外，Kubernetes 会请求底层云平台上的负载均衡器，将每个 Node （<code>[NodeIP]:[NodePort]</code> 作为后端添加进去。
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2020/06/image-49ddc161.png?imageView2/2/interlace/1/format/jpg"></>
LB 解决的问题
前面加一个负载均衡器（公网）：

把内网节点的端口提供的服务给暴露到公网
为 nodeport 提供高可用能力

工作流程：<code>user</code> -> <code>lb</code> -> <code>node:port</code> -> <code>[service]</code> -> <code>pod</code>
<h3 id="1-4-Service代理模式">1.4 Service 代理模式</h3>
Service 代理模式分为：<code>Iptables</code> 和 <code>IPVS</code>
proxy 模式：

<code>iptables</code> （默认使用）:用户态的工具主要用于 netfilter 规则管理
入口流量规则-> 轮训 pod 机制-> 实际 DNAT 规则(目标地址转化)-> 容器
<code>ipvs</code> :

<pre><code class="highlight-chroma"> # kubectl edit configmap kube-proxy -n kube-system
  mode: "ipvs"
  # kubectl delete od kube-proxy-btz4p -n kube-system
  # yum install ipvadm -y
  # ipvsadm -L -n
</code></pre>
查看 Service 网络规则
<pre><code class="highlight-chroma">iptables-save | grep <svc-name>
</code></pre>
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2020/06/image-79cc834e.png?imageView2/2/interlace/1/format/jpg"></>

<h4 id="1-4-1-iptables改为ipvs步骤-">1.4.1 iptables 改为 ipvs 步骤：</h4>
<h5 id="二进制部署">二进制部署</h5>
1.将 <code>/opt/kubernetes/cfg/kube-proxy-config.yaml</code> 配置文件中，注销 <code>mode</code> 参数，并在后面添加 <code>ipvs</code>（下面的一些参数是 <code>ipvs</code> 的调度算法）
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2020/06/image-424a2f12.png?imageView2/2/interlace/1/format/jpg">
2.重启 kube-proxy 服务即可生效：
<pre><code class="highlight-chroma">systemctl restart kube-proxy
</code></pre>
3.安装 <code>ipvsadm</code> 工具去查看 ipvs 规则：
<pre><code class="highlight-chroma">
yum -y install ipvsadm
ipvsadm -L -n
</code></pre>
参数解释：

<code>-L</code>：列出规则
<code>-n</code>：以数字而不是以主机名的形式显示 IP 地址

<h5 id="kubeadm部署">kubeadm 部署</h5>
1.由于 <code>kube-proxy</code> 保存在 k8s 资源中，故需要编辑 <code>configmap</code> 的配置文件
<pre><code class="highlight-chroma">

[root@k8s-master k8s]# kubectl get pod -n kube-system
NAME
READY STATUS RESTARTS AGE
kube-proxy-grnpw
1/1 Running 11 12d
kube-proxy-mshjk
1/1 Running 11 12d
kube-proxy-nkkk4
1/1 Running 11 12d

[root@k8s-master k8s]# kubectl get configmaps -n kube-system
NAME
DATA AGE
kube-proxy
2 12d
</code></pre>
2.打开 kube-peoxy 的 configmap：
<pre><code class="highlight-chroma">

kubectl edit configmap kube-proxy -n kube-system
</code></pre>
3.找到 <code>mode</code> 参数，在引号中写入 <code>ipvs</code>，保存即可
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2020/06/image-78889ce8.png?imageView2/2/interlace/1/format/jpg">
4.删除 <code>kube-proxy</code> 的 pod 重建,等待集群自动拉起，ipvs 即可生效：
<pre><code class="highlight-chroma">

kubectl delete pod kube-proxy-grnpw -n kube-system

```
</span></span><span class="highlight-line"><span class="highlight-cl">kubectl delete pod
kube-proxy-mshjk -n kube-system
</span></span><span class="highlight-line"><span class="highlight-cl">kubectl delete pod
kube-proxy-nkkk4 -n kube-system
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">[root@k8s-master
k8s]# kubectl get pod -n kube-system
</span></span><span class="highlight-line"><span class="highlight-cl">NAME
                READY   STATUS    RESTARTS   AGE
</span></span><span class="highlight-line"><span class="highlight-cl">kube-proxy-272qn
                       1/1     Running   0          42s
</span></span><span class="highlight-line"><span class="highlight-cl">kube-proxy-gvq2n
                       1/1     Running   0          33s
</span></span><span class="highlight-line"><span class="highlight-cl">kube-proxy-jbjfc
                    1/1     Running   0          61s
</span></span></code></pre>
<p>5.利用 <code>ipvsadm -L -n</code> 命令查看规则</p>
<h4 id="1-4-2-Iptables-VS-IPVS-优缺点">1.4.2 Iptables VS IPVS 优缺点</h4>
<p><em>Iptables</em>：</p>
<ul>
<li>灵活，功能强大</li>
<li>规则遍历匹配和更新，呈线性时延</li>
</ul>
<p><em>IPVS</em>：</p>
<ul>
<li>工作在内核态，有更好的性能</li>
<li>调度算法丰富：rr，wrr，lc，wlc，ip hash...</li>
</ul>
<p>参数解释：</p>
<ul>
<li><code>rr</code>：轮询模式</li>
<li><code>wrr</code>：加权轮询模式</li>
<li><code>lc</code>：最小连接模式</li>
<li><code>wlc</code>：加权连接模式</li>
<li><code>ip hash</code>：</li>
</ul>
<h3 id="1-5-Service-DNS名称">1.5 Service DNS 名称</h3>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">[root@k8s-master k8s]# kubectl get pod -n kube-system
</span></span><span class="highlight-line"><span class="highlight-cl">NAME
                READY   STATUS    RESTARTS   AGE
</span></span><span class="highlight-line"><span class="highlight-cl">coredns-7ff77c87
f-cgfjw                  1/1     Running   11         12d
</span></span><span class="highlight-line"><span class="highlight-cl">coredns-7ff77c87
f-pn8qk                  1/1     Running   12         12d
</span></span></code></pre>
<p>DNS 服务监视 Kubernetes API，为每一个 Service 创建 DNS 记录用于域名解析。</p>
<p>测试 <code>coredns</code>，使用 <code>busybox:1.28.4</code> 镜像并进入 pod 中</
>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">[root@k8s-master k8s]# kubectl run  -it --rm --image=busybox:1.28.4 sh
</span></span><span class="highlight-line"><span class="highlight-cl">If you don't see a
ommand prompt, try pressing enter.
</span></span><span class="highlight-line"><span class="highlight-cl">/ #
```

</code></pre>
正常来讲，在这个 busybox 的 Pod 中是 ping 的通上面创建的 svc 的 IP 地址的，也可以访问 sv 的页面。
<pre><code class="highlight-chroma">/ # ping 10.101.205.162
PING 10.101.205.62 (10.101.205.162): 56 data bytes
64 bytes from 10.01.205.162: seq=0 ttl=64 time=0.101 ms
64 bytes from 10.01.205.162: seq=1 ttl=64 time=0.099 ms
64 bytes from 10.01.205.162: seq=2 ttl=64 time=0.101 ms
64 bytes from 10.01.205.162: seq=3 ttl=64 time=0.107 ms
^C
--- 10.101.205.162 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/av/max = 0.099/0.102/0.107 ms
/ # wget 10.101.25.162
Connecting to 10.01.205.162 (10.101.205.162:80)
index.html
00% |***|
12 0:00:00 ETA
/ #
</code></pre>
可以使用 <code>nslookup</code> 命令解析 dns 名称
<pre><code class="highlight-chroma">/ # nslookup web
Server: 10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: web
Address 1: 10.101.205.162 web.default.svc.cluster.local
</code></pre>
解析出来的 IP 对应 svc 的 IP 地址
<blockquote>
注意：<code>nslookup</code> 默认不能跨命名空间使用，若想解析跨命名空间的 dns，则要使用全格式
</blockquote>
ClusterIP A 记录格式：<code><service-name>.<namespace-name>.svc.cluster.local</code>
示例：<code>my-svc.my-namespace.svc.cluster.local</code>
Pod 在发送 dns 请求的时候，实际上是请求的 <code>/etc/resolv.conf</code> 文件中的 dns，这个 dns 就是部署 <code>coredns</code> 的 <code>Service</code>,所以执行 <code>nslookup</code> 命令时是向 coredns 发出请求，而 <code>coredns</code> 里面有 Service 对应 IP

的记录，之后响应记录结果，并且 <code>coredns</code> 对域名也有区分，若判断为外部域名则走上层宿主机 dns 进行解析，然后再响应给 Pod。
总结：
pod -> coredns service(10.0.0.2) -> coredns(service/clusterip 记录) -> 响应 A 记录结

<pre><code class="highlight-chroma"><span class="highlight
cl">/ # cat /etc/resolv.conf
nameserver 10.96.
.10
search default.svc.
luster.local svc.cluster.local cluster.local
options ndots:5

[root@k8s-node1
manifests]# kubectl get svc -n kube-system
NAME TYP
  CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kube-dns Clu
terIP 10.96.0.10 <none> 53/UDP,53/TCP,9153/TCP 12d
</code></pre>
总结：

采用 NodePort 对外暴露应用，前面加一个 LB 实现统一访问入口

优先使用 IPVS 代理模式
IPVS 性能高，调度算法丰富，可以满足多业务大并发的场景下

集群内应用采用 DNS 名称访问
当切换集群或者换 ServiceIP 的时候对应于程序没什么影响

