Kubernetes权威指南

Subtitle

2022/09/25

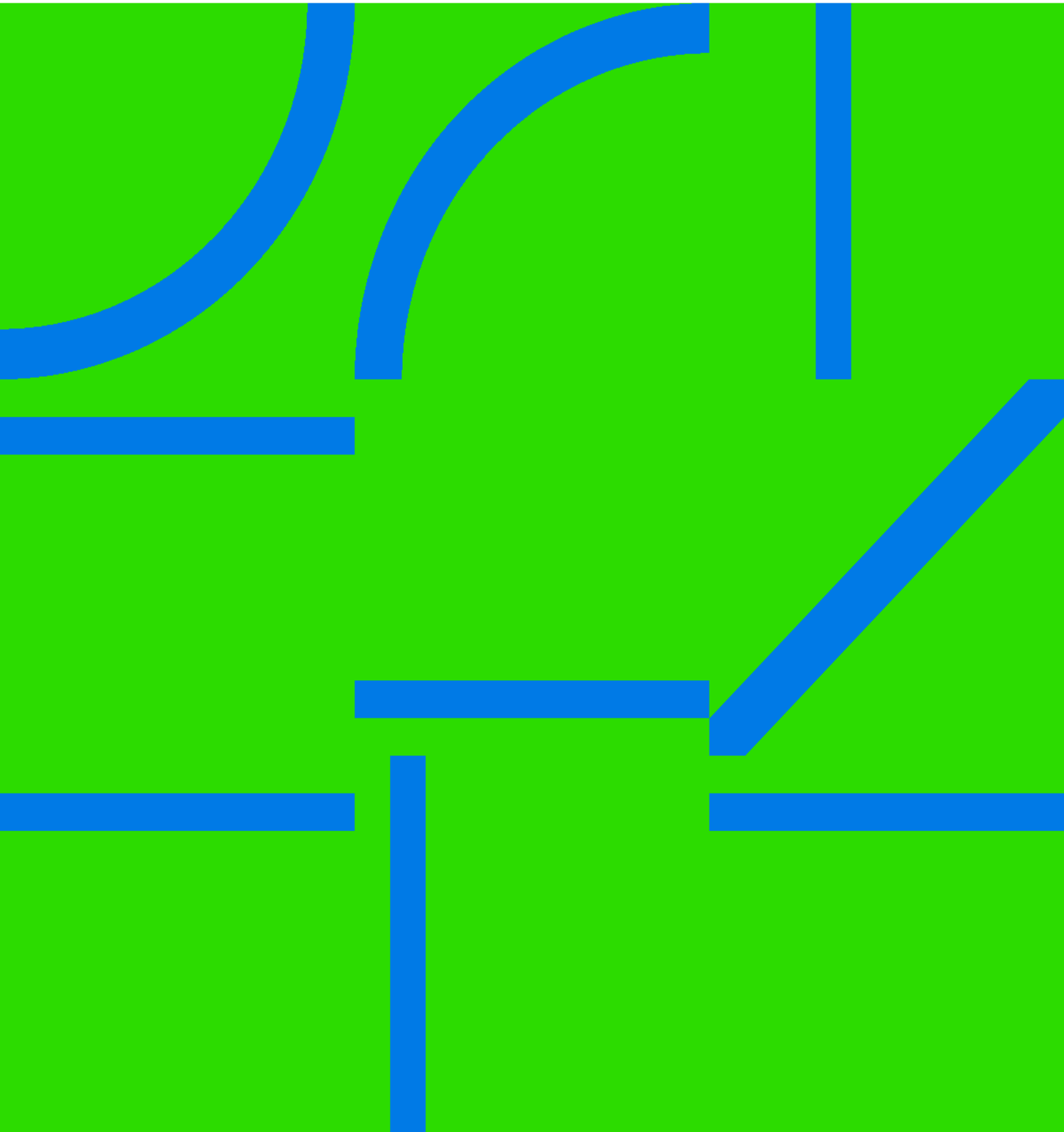

Table of Contents

Kubernetes权威指南

概念

组件

核心组件

Add-ons

Service

Ingress

Ingress实现类似mpaas router的功能

Ingress使用service name作为proxy_pass，借助kubernetes内部dns解析为service cluster ip，这样配置文件变更不算频繁，可以直接使用Ingress来实现一个router

- Ingress指定Node部署，并暴露宿主机端口
- 根据用户配置生成或更新ingress配置文件
- 指定的Node做lvs接受用户访问

Docker网络

Netfilter挂接规则点

SSL/TLS证书

SAN证书

SAN是指Subject Alternative Name，是 SSL 标准 x509 中定义的一个扩展。使用了 SAN 字段的 SSL 证书，可以扩展此证书支持的域名（或IP），使得一个证书可以支持多个不同域名（或IP）的解析。以本站证书

为例：

```
# 首先下载证书，  </dev/null是向前一个命令输入空
[root@repo ~]# openssl s_client -connect wiki.annhe.net:443 -showcerts </dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > annhe.pem
# 查看证书信息
[root@repo ~]# openssl x509 -noout -text -in annhe.pem |grep -A 1 'Altern'
            X509v3 Subject Alternative Name:
                DNS:annhe.net, DNS:att.annhe.net, DNS:att.tecbbs.com, DNS:gg.annhe.net, DNS:hnu.tecbbs.com, DNS:m.tecbbs.com, DNS:p.annhe.net, DNS:tecbbs.com, DNS:tmp.annhe.net, DNS:wiki.annhe.net, DNS:www.annhe.net, DNS:www.tecbbs.com
```

双向认证

```
# 服务端 -Verify 参数要求客户端必须发送证书
openssl s_server -cert test.pem -CAfile ca.pem -key test-key.pem -Verify 1
```

客户端未发送证书

```
openssl s_client -connect 127.0.0.1:4433 -CAfile ca.pem

# 服务端响应：
ERROR
139826282153800:error:140890C7:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:peer did not return a certificate:s3_srvr.c:3312:
shutting down SSL
CONNECTION CLOSED
```

客户端发送证书

```
openssl s_client -connect 127.0.0.1:4433 -cert admin.pem -key admin-key.pem -CAfile ca.pem

# 服务端响应：
depth=1 C = CN, ST = Beijing, L = Beijing, O = Letv, OU = Scloud, CN = k8s.product
verify return:1
depth=0 C = CN, ST = Beijing, L = Beijing, O = system:masters, OU = Scloud, CN = admin
verify return:1
```

双向认证测试**SAN**证书效果

Nginx配置

```
ssl_certificate /usr/local/nginx/ssl/server.crt;
ssl_certificate_key /usr/local/nginx/ssl/server.key;
ssl_client_certificate /root/ssl/ca.pem;
ssl_verify_client on;
```

证书SAN信息：

```
[root@repo ssl]# openssl x509 -noout -text -in test.pem |grep -A 1 "Alt"
            X509v3 Subject Alternative Name:
```

```
                DNS:kubernetes.default, IP Address:127.0.0.1
```

使用curl访问（需要用较高版本的curl□centos6上的版本较低需要升级，可以使用city-fan.org源），指定客户端证书，CA

```
[root@repo ssl]# curl -s --cert ./admin.pem --key ./admin-key.pem --cacert ./ca.pem
"https://kubernetes.default" -v
* Rebuilt URL to: https://kubernetes.default/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to kubernetes.default (127.0.0.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  CAfile: ./ca.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
... 部分略
* Server certificate:
*  subject: C=CN; ST=Beijing; L=Beijing; O=Letv; OU=Scloud; CN=k8s.product
*  start date: Mar 22 02:17:00 2018 GMT
*  expire date: Mar 19 02:17:00 2028 GMT
*  subjectAltName: host "kubernetes.default" matched cert's "kubernetes.default"
*  issuer: C=CN; ST=Beijing; L=Beijing; O=Letv; OU=Scloud; CN=k8s.product
*  SSL certificate verify ok.
> GET / HTTP/1.1
> Host: kubernetes.default
> User-Agent: curl/7.59.0
> Accept: */*
>
< HTTP/1.1 200 OK
...
```

使用Altname中没有的IP或者主机名会报错：

*** SSL: no alternative certificate subject name matches target host name '192.168.60.10'**

不指定CA:

*** SSL certificate problem: unable to get local issuer certificate**

指定CA但不指定客户端证书

```
[root@repo ssl]# curl -s --cacert ./ca.pem "https://127.0.0.1" -v
* Rebuilt URL to: https://127.0.0.1/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
```

```
  CAfile: ./ca.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
...部分略
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* Server certificate:
*  subject: C=CN; ST=Beijing; L=Beijing; O=Letv; OU=Scloud; CN=k8s.product
*  start date: Mar 22 02:17:00 2018 GMT
*  expire date: Mar 19 02:17:00 2028 GMT
*  subjectAltName: host "127.0.0.1" matched cert's IP address!
*  issuer: C=CN; ST=Beijing; L=Beijing; O=Letv; OU=Scloud; CN=k8s.product
*  SSL certificate verify ok.
> GET / HTTP/1.1
> Host: 127.0.0.1
> User-Agent: curl/7.59.0
> Accept: */*
>
< HTTP/1.1 400 Bad Request
< Server: nginx
< Date: Thu, 22 Mar 2018 08:30:08 GMT
< Content-Type: text/html
< Content-Length: 246
< Connection: close
<
<html>
<head><title>400 No required SSL certificate was sent</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<center>No required SSL certificate was sent</center>
<hr><center>nginx</center>
</body>
</html>
* Closing connection 0
* TLSv1.2 (OUT), TLS alert, Client hello (1):
```

不设置SAN的测试

如果没有设置SAN（cfssl工具csr配置中hosts为空），将以CN字段来检查证书：

```
[root@repo ssl]# curl -s --cert admin.pem --key admin-key.pem --cacert ./ca.pem "https://127.0.0.1" -v
...
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* Server certificate:
*  subject: C=CN; ST=Beijing; L=Beijing; O=Letv; OU=Scloud; CN=k8s.product
*  start date: Mar 22 08:36:00 2018 GMT
*  expire date: Mar 19 08:36:00 2028 GMT
* SSL: certificate subject name 'k8s.product' does not match target host name '127.0.0.1'
* stopped the pause stream!
```

```
* Closing connection 0
* TLSv1.2 (OUT), TLS alert, Client hello (1):
```

由此可见在kubernetes环境中使用时有必要设置SAN

故障

网络问题

书中hello-world例子，环境概览

- 1 kube Master + 2 kube Node
- Docker使用默认参数启动，未设置bip，2个Node bip均为 172.17.0.1/16
- 未配置容器网络互通
- redis-slave replicas为2，2个Node上各一个
- redis-master replicas为1， 在Node1上
- 通过在redis-slave容器中执行telnet redis-master-clusterIP 6379，Node1上成功，Node2上失败

解决方案：

- Docker使用不同网段
- 2台Node互加docker0网段路由

引用书中内容：

之前提到，Kubernetes的网络对Pod的地址是平面的和直达的，所以这些Pod的IP规划也很重要，不能有冲突。只要没有冲突，我们就可以想办法在整个Kubernetes的集群中找到它。

综上所述，要想支持不同Node上的Pod之间的通信，就要达到两个条件：

（1）在整个Kubernetes集群中对Pod的IP分配进行规划，不能有冲突；

（2）找到一种办法，将Pod的IP和所在Node的IP关联起来，通过这个关联让Pod可以互相访问。

根据条件1的要求，我们需要在部署Kubernetes的时候，对docker0的IP地址进行规划，保证每一个Node上的docker0地址没有冲突。我们可以在规划后手工配置到每个Node上，或者做一个分配规则，由安装的程序自己去分配占用。例如Kubernetes的网络增强开源软件Flannel就能够管理资源池的分配。

ServiceAccount

按照书中所述直接关闭了ServiceAccount，在使用nginx-ingress时遇到问题：

```
kubectl describe pod nginx-ingress-ljy43
...
Error syncing pod, skipping: failed to "StartContainer" for "nginx" with CrashLoopBackOff: "Back-off 20s restarting failed container=nginx pod=nginx-ingress-ljy43_default(2e1d4c3a-1fa7-11e8-ac1d-
```

```
fa168f21c6d7)"
```

查看pod日志，可以看到

```
2018/03/04 12:28:26 Failed to create client: open /var/run/secrets/kubernetes.io/serviceaccount/token: no such file or directory.
```

搜索得知需要开启ServiceAccount

出处：
http://stackoverflow.com/questions/34464779/pod-mysql-is-forbidden-no-api-token-found-for-service-account-default-default

To get your setup working, you can do the same thing local-up-cluster.sh is doing:

Generate a signing key: openssl genrsa -out /tmp/serviceaccount.key 2048

Update /etc/kubernetes/apiserver:
KUBE*API*ARGS="–service*account*key_file=/tmp/serviceaccount.key"

Update /etc/kubernetes/controller-manager:
KUBE*CONTROLLER*MANAGER*ARGS="–service*account*private*key_file=/tmp/serviceaccount.key"

From
https://github.com/kubernetes/kubernetes/issues/11355#issuecomment-127378691

etcd不可用的影响

停掉etcd, kubectl会报错：

```
[root@k8s.master ~]# kubectl get po
client: etcd cluster is unavailable or misconfigured
```

已经部署的服务不受影响

技巧

kubectl命令自动补全

```
# yum install -y bash-completion
# locate bash_completion
/usr/share/bash-completion/bash_completion
# source /usr/share/bash-completion/bash_completion
# source <(kubectl completion bash)
```

加入profile配置文件使登录时生效

```
[root@k8s.master ~]# cat /etc/profile.d/kubectl.sh
source <(kubectl completion bash)
```

设置docker bip参数

```
[root@k8s.node ~]# cat /etc/sysconfig/docker-network
# /etc/sysconfig/docker-network
DOCKER_NETWORK_OPTIONS='--bip="172.19.0.1/16"'
```

docker镜像仓库

使用163的：hub-mirror.c.163.com

node主机名解析

kube-apiserver调用node上的kubelet时，使用的是https://node-hostname:10250/，因此需要做域名解析。使用定时任务，添加dnsmasq域名劫持配置，脚本如下：

k8s-node-resolv.sh

```
#!/bin/bash
#主机名格式为k8s.node.1-1-1-1.xxx.com，即主机名中包含IP地址
#如果主机名不是这种方式，需要执行kubectl describe node nodeName来获取IP地址

function dns() {
    nodes=`kubectl get node |awk 'NR>1{print $1}'`
    for node in $nodes;do
        ip=`echo $node |cut -f3 -d'.' |sed 's/-/./g'`
        echo "address=/$node/$ip"
    done
}

dns >/tmp/k8snode.conf
diff -q /tmp/k8snode.conf /etc/dnsmasq.d/k8snode.conf
if [ $? -ne 0 ];then
    echo "update dnsmasq..."
      dns > /etc/dnsmasq.d/k8snode.conf
      systemctl restart dnsmasq.service
fi
```

Printed on: 2022/09/25 09:44