Ingress Nginx

Subtitle

2022/09/28

Table of Contents

Ingress Nginx

未更新upstream

某一个节点未更新upstream，导致部分5xx

可能是dnsPolicy问题？启用了hostNetwork时，可能无法解析内部域名(kubernetes.default)

ssl-ciphers配置问题

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3; # 开启sslv3，客户端有时会使用sslv3
ssl_ciphers   AES128-SHA:AES256-SHA:ECDHE-RSA-AES128-GCM-
SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE; # 加密算法和slb保持一致
```

按照以上配置更改，由于全局HTTP2，浏览器无法访问https服务，
报ERR*SPDY*INADEQUATE*TRANSPORT*SECURITY， curl用http 1.1可以访问。

```
无法访问此网站
网址为 https://ke.xxx.com/ 的网页可能暂时无法连接，或者它已永久性地移动到了新网址。
ERR_SPDY_INADEQUATE_TRANSPORT_SECURITY
```

解决方案，恢复ingress默认的加密算法

nf_conntrack: table full

解决方案

更彻底的解决方案，router节点添加iptables规则不跟踪80,443端口的请求

- http://blog.51cto.com/michaelkang/1090945
- http://blog.51cto.com/362475097/1892169

```
iptables -t raw -A PREROUTING -d 1.2.15.21 -p tcp --dport 80 -j NOTRACK
iptables -t raw -A OUTPUT -s 1.2.15.21 -p tcp --sport 80 -j NOTRACK
iptables -t raw -A PREROUTING -d 1.2.15.21 -p tcp --dport 443 -j NOTRACK
iptables -t raw -A OUTPUT -s 1.2.15.21 -p tcp --sport 443 -j NOTRACK
```

参考 文章:

> IPVS uses its own simple and fast connection tracking for performance reasons, instead of using netfilter connection tracking. So, if you don’t use firewalling feature at load balancer and you need an extremely fast load balancer, do not load netfilter conntrack modules into you system, because there is no need to do double tracking. Note that LVS/NAT should work too without the conntrack modules.
>
> Julian compared the performance of IPVS with ip_conntrack and without ip_conntrack. See http://archive.linuxvirtualserver.org/html/lvs-users/2001-12/msg00141.html
>
> 默认情况下LVS自身会记录连接信息，但是 iptables 也会记录 connection 的状态，但是很多情况下，我们并不需要 iptables 来做这件事，

> 我们可以告诉它 NOTRACK，不要记录这些信息。

以下脚本添加计划任务

```
#!/bin/bash
vip=`ip addr show lo |grep "/32" |awk '{print $2}' |cut -f1 -d'/'`

[ "$vip"x == ""x ] && echo "no vip" && exit 1

iptables -vnL -t raw |grep "$vip" |grep "dpt:443" &>/dev/null || iptables -t raw -A PREROUTING -d $vip -p tcp --dport 443 -j NOTRACK
iptables -vnL -t raw |grep "$vip" |grep "dpt:80" &>/dev/null || iptables -t raw -A PREROUTING -d $vip -p tcp --dport 80 -j NOTRACK
iptables -vnL -t raw |grep "$vip" |grep "spt:443" &>/dev/null || iptables -t raw -A OUTPUT -s $vip -p tcp --sport 443 -j NOTRACK
iptables -vnL -t raw |grep "$vip" |grep "spt:80" &>/dev/null || iptables -t raw -A OUTPUT -s $vip -p tcp --sport 80 -j NOTRACK

iptables -vnL -t raw |grep $vip
```

kube-proxy有参数可以设置这些值

```
# kube-proxy --help |grep conntrac
      --conntrack-max-per-core int32                Maximum number of NAT connections to track per CPU core (0 to leave the limit as-is and ignore conntrack-min). (default 32768)
      --conntrack-min int32                         Minimum number of conntrack entries to allocate, regardless of conntrack-max-per-core (set conntrack-max-per-core=0 to leave the limit as-is). (default 131072)
      --conntrack-tcp-timeout-close-wait duration    NAT timeout for TCP connections in the CLOSE_WAIT state (default 1h0m0s)
      --conntrack-tcp-timeout-established duration   Idle timeout for established TCP connections (0 to leave as-is) (default 24h0m0s)
```

如果使用了配置文件，需要在配置文件中配置，否则不生效

```
conntrack:
  min: 1572864
  maxPerCore: 131072
```

参考文章

- http://www.10tiao.com/html/488/201701/2247484116/1.html

```
[Tue Nov 20 22:30:22 2018] nf_conntrack: table full, dropping packet
[Tue Nov 20 22:30:27 2018] nf_conntrack: table full, dropping packet
[Tue Nov 20 22:30:27 2018] nf_conntrack: table full, dropping packet
[Tue Nov 20 22:30:27 2018] nf_conntrack: table full, dropping packet
```

原因是 kube-proxy会修改 nf*conntrack*max

```
I1120 15:18:19.757786   26448 conntrack.go:52] Setting nf_conntrack_max to 786432
```

用puppet改大。

思考

域名从slb变更解析到k8s router之后出现此问题。~~链接跟踪是否跟客户端数量有关系？（slb就10几个，真实用户很多）~~

conntrack -L命令，看到连接追踪表:

```
tcp      6 11 LAST_ACK src=23.16.22.25 dst=13.9.25.29 sport=5300 dport=443 src=13.9.25.29
dst=23.16.22.25 sport=443 dport=5300 [ASSURED] mark=0 use=1
tcp      6 1442 CLOSE_WAIT src=17.50.3.8 dst=13.9.25.29 sport=42287 dport=80 src=13.9.25.29
dst=17.50.3.8 sport=80 dport=42287 [ASSURED] mark=0 use=1
tcp      6 119 TIME_WAIT src=18.21.16.1 dst=13.9.25.29 sport=45390 dport=80 src=13.9.25.29
dst=18.21.16.1 sport=80 dport=45390 [ASSURED] mark=0 use=1
```

可以看到记录了每一个请求。

对比前面还有一层LB的mpaas router情况

```
tcp      6 83 TIME_WAIT src=10.122.130.61 dst=10.124.152.24 sport=30103 dport=80
src=10.124.152.24 dst=10.122.130.61 sport=80 dport=30103 [ASSURED] mark=0 secmark=0
use=2
tcp      6 110 TIME_WAIT src=10.110.220.54 dst=10.124.152.24 sport=53696 dport=80
src=10.124.152.24 dst=10.110.220.54 sport=80 dport=53696 [ASSURED] mark=0 secmark=0
use=2
tcp      6 52 TIME_WAIT src=10.110.220.54 dst=10.124.152.24 sport=41104 dport=80
src=10.124.152.24 dst=10.110.220.54 sport=80 dport=41104 [ASSURED] mark=0 secmark=0
use=2
tcp      6 74 TIME_WAIT src=10.110.220.42 dst=10.124.152.24 sport=22298 dport=80
src=10.124.152.24 dst=10.110.220.42 sport=80 dport=22298 [ASSURED] mark=0 secmark=0
use=2
tcp      6 7821 ESTABLISHED src=10.124.152.24 dst=10.110.91.154 sport=21453 dport=31293
src=10.110.91.154 dst=10.124.152.24 sport=31293 dport=21453 [ASSURED] mark=0 secmark=0
use=2
```

应该是kubernetes用了iptables k8s intra集群

```
# sysctl net.netfilter.nf_conntrack_count
net.netfilter.nf_conntrack_count = 246399
```

k8s普通node节点

```
# sysctl net.netfilter.nf_conntrack_count
net.netfilter.nf_conntrack_count = 58418
```

普通nginx集群

```
# sysctl net.netfilter.nf_conntrack_count
net.netfilter.nf_conntrack_count = 96938
```

影响

丢包导致ingress 10254端口的健康检查失败，频繁重启ingress，相当于流量入口挂了。最好将健康检查阈值调大，避免频繁重启

监控

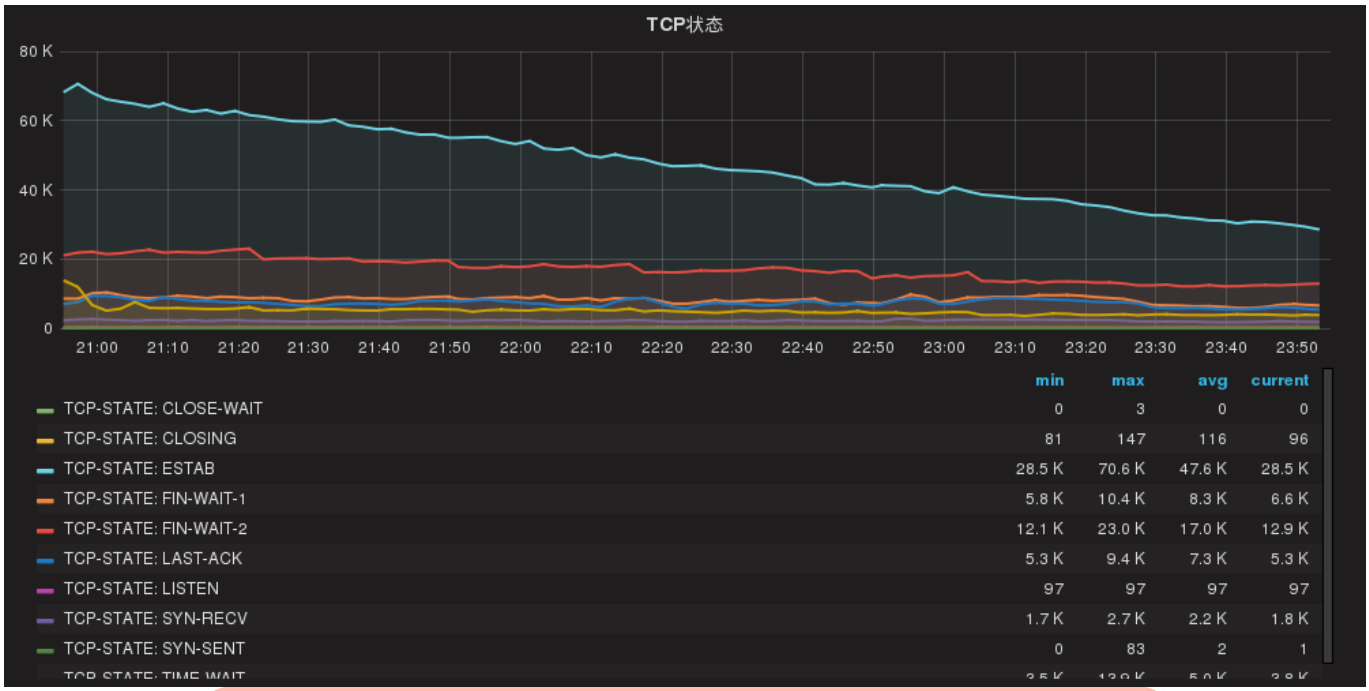
使用zabbix自定义key

```
#!/bin/bash
[ $# -lt 1 ] && exit 1

k=`echo $1 |tr '.' '/'`

cat /proc/sys/$k 2>/dev/null
```

海外客户端访问

公司海外业务下线，域名也删除了解析，后来k8s router使用了域名泛解析，之前海外域名通过泛解析请求打到了k8s，可能由于防火墙的原因，或者单纯的网络质量问题，造成一些tcp状态堆积:

- SYN-RECV 服务端ACK没有送达客户端，或者客户端没有回ACK
- FIN-WAIT 服务端主动关闭连接时没有收到客户端ACK
- LAST-ACK 服务端被动关闭时FIN=1没有收到主动端ACK



以下方法没有减少以上tcp状态

考虑添加配置，将此类域名返回 444

```
server {
    listen       80  ;
    server_name  *.g.abc.com;
    return       444;
```

conntrack 统计

```
# conntrack -L |head -n 200000 |awk '{print $4}' |sort |uniq -c |sort -n |grep -v src
      2 SYN_SENT
     92 FIN_WAIT
    520 SYN_RECV
   1626 LAST_ACK
   3842 CLOSE
  17738 ESTABLISHED
  55386 TIME_WAIT
 120770 CLOSE_WAIT
```

close-wait占比非常大，主要原因是 kube-proxy默认的–conntrack-tcp-timeout-close-wait是1小时

```
--conntrack-tcp-timeout-close-wait duration    NAT timeout for TCP connections in the CLOSE_WAIT
state (default 1h0m0s)
```

~~考虑在router节点上减少该值~~，默认值是60s（**更彻底的解决方案见上文，禁用vip的连接跟踪**）

搜到一些kubernetes设置该值的原因

- https://github.com/kubernetes/kubernetes/issues/32551
- https://marc.info/?l=netfilter-devel&m=117568928824030&w=2
- https://github.com/kubernetes/kubernetes/issues/32551#issuecomment-249708406

A little from column A and a little from column B? I think we can tune the close*wait timeout to 60 minutes - it will only affect sockets in CLOSE*WAIT, which isn't a normal state to linger in (I think?!). I think the other thing you suggest is the real fix. We want to fix that ANYWAY, but we should probably think hard about how to fix it properly. I don't think a simple flag is enough - I'd love to see this go into the per-node configmaps...

Printed on: 2022/09/28 06:43

Convert to img Failed!