openshift tcp-router

https://github.com/wangzheng422/docker_env/blob/master/redhat/ocp4/4.3/4.3.haproxy.md

本文描述，如何通过定制化 haproxy template，通过给 route 添加 annotation，就可以向外界开放 tcp 路由。本文相关脚本和文件，在 scripts 目录中。

初衷和原理

经常会在 openshift 的 poc 中遇到 L4 负载均衡的测试，我们知道默认 ocp router 是 haproxy 做的，而且默认只支持 http, https，虽然 tls/sni 也算是支持 tcp 的一种方式，但是这个也还是 7 层的。官方文档只是简单的说，如果有其他的需求，就定制 haproxy template 来满足，不过定制说的很少，例子也不多。本文就是一个通过定制化 haproxy template，来达到动态监听 route 配置，并动态开放 tcp 端口。

定制 haproxy template 需要了解 openshift router 的一些原理要点

- openshift router 不仅仅是 haproxy，它还有一个 go 程序，监听了 openshift 的配置，并且写入了一堆的 map 文件，这个文件是非常关键的配置 haproxy template 的配置文件。
- openshift router 里面的 tls passthrough 方式，对应到 haproxy 的配置里面，就是 tcp 的模式，我们的定制点就是在这里。

- 定制过程集中在，屏蔽掉 http/https 的 edge 和 reencrypt 部分，对于打 annotation 的 route，开放 tls passthrough 的 frontend
- route annotation 配置形式是 haproxy.router.openshift.io/external-tcp-port: "13306"
- 当然，ocp4 现在还不支持定制化 route template，所以本文直接创建了一个 route 的 deployment。
- 现场实施的时候，注意更改 router 的 image，每个版本的 image 可以去 release.txt 文件中找到。

既然是面向 poc 的，就肯定有局限

- route annotation 定义的开放 tcp 端口，是手动定义，而且面向整个集群各个 project 开放，必然会导致 tcp 端口冲突。需要已有端口管理方案，这个就交给红帽实施团队。

以下是 route 的配置示例

```
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: ottcache-002
  annotations:
    haproxy.router.openshift.io/wzh-router-name: "wzh-router-1"
    haproxy.router.openshift.io/external-tcp-port: "6620"
spec:
  to:
    kind: Service
    name: ottcache-002-service
  port:
    targetPort: 6620
  tls:
    termination: passthrough
```

```
    insecureEdgeTerminationPolicy: None
```

以下是 template 里面，关键的定制点

```
{{/*try to add tcp support*/}}

{{- if eq (env "WZH_ROUTER_NAME" "wzh-router-name") (index $cfg.Annotations
"haproxy.router.openshift.io/wzh-router-name") }}
  {{- if (isInteger (index $cfg.Annotations
"haproxy.router.openshift.io/external-tcp-port")) }}
  frontend tcp-{{ (index $cfg.Annotations
"haproxy.router.openshift.io/external-tcp-port") }}
    bind *:{{ (index $cfg.Annotations
"haproxy.router.openshift.io/external-tcp-port") }}
    mode tcp
    default_backend {{genBackendNamePrefix
$cfg.TLSTermination}}:{{$cfgIdx}}

  {{- end}}{{/* end haproxy.router.openshift.io */}}
{{- end}}{{/* end WZH_ROUTER_NAME */}}

{{/*end try to add tcp support*/}}
```

测试步骤

测试步骤不复杂，就是创建一个新的 router，然后就可以去其他 project 创建应用，给 route 打 annotation 就可以了。

本文的例子，包含两个应用，一个是 web 应用，一个是 mysql，都通过 tcp 端口对外开放。

```
# tcp-router will install in the same project with openshift router
oc project openshift-ingress

# install the tcp-router and demo
oc create configmap customrouter-wzh --from-file=haproxy-config.template
oc apply -f haproxy.router.yaml

oc apply -f haproxy.demo.yaml
```

```
# test your tcp-router, replace ip with router ip, both command will
success.
curl 192.168.7.18:18080

podman run -it --rm registry.redhat.ren:5443/docker.io/mysql mysql -h
192.168.7.18 -P 13306 -u user -D db -p

# if you want to delete the tcp-router and demo
oc delete -f haproxy.router.yaml
oc delete configmap customrouter-wzh

oc delete -f haproxy.demo.yaml

# oc set volume deployment/router-wzh --add --overwrite \
#     --name=config-volume \
#     --mount-path=/var/lib/haproxy/conf/custom \
#     --source='{"configMap": { "name": "customrouter-wzh"}}'

# oc set env dc/router \
#     TEMPLATE_FILE=/var/lib/haproxy/conf/custom/haproxy-config.template
```

参考

https://docs.openshift.com/container-platform/3.11/install_config/router/customized_haproxy_router.html#go-template-actions

https://www.haproxy.com/blog/introduction-to-haproxy-maps/

https://access.redhat.com/solutions/3495011

https://blog.zhaw.ch/icclab/openshift-custom-router-with-tcpsni-support/

以下是弯路

分析源码，我们可以看到，openshift router 还是对 haproxy 做了扩展的，那些 map 文件，都是 router 的扩展生成的，目的是对接 endpoint，绕过 service。所以我们想做 tcp 转发，可以借助 sni-tcp 来实现 tcp 转发。