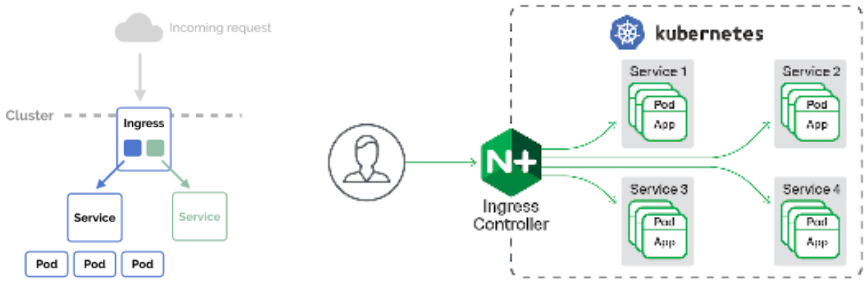
K8S Ingress介绍

1 Ingress基本介绍

1.1 ingress工作原理图



- ingress是K8S中的一个API对象，用来管理集群外部访问集群内部服务的方式(如，流量路由方式)
- 为了使得ingress资源正常工作，集群中必须要有一个Ingress Controller来实现Ingress的资源配置/定义

1.2 ingress功能

- 负载均衡
- SSL termination
- 基于域名路由

1.3 ingress使用

- 资源定义：

```
# cat example-ingress.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  namespace: default
spec:
  backend:
    serviceName: web-default
    servicePort: 8080
  rules:
  - host: web1.info
    http:
      paths:
      - path: /
        backend:
          serviceName: web1
          servicePort: 8080
  - host: web2.info
    http:
      paths:
      - path: /
        backend:
          serviceName: web2
          servicePort: 8080
# kubectl  apply -f example-ingress.yaml
ingress.networking.k8s.io/example-ingress created
#
```

说明：

- 创建的 Ingress 必须要和对外暴露的 Service 在同一namespace下，即ingress跟namespaces关联（看API文档的话也可发现）
- backend表示为默认后端服务，在匹配所有rules失败的情况下，就会使用默认后端服务
- backend指定的默认后端服务，必须使用域名方式访问(同时域名必须是host选项中的一个，不可以是其他域名），不可以使用IP（即使rules中的host选项没有配置）
- 更多用法查看官方API手册：https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.16/#ingress-v1beta1-networking-k8s-io

- 资源查看：

```
# kubectl get ing example-ingress
NAME              HOSTS                 ADDRESS         PORTS   AGE
example-ingress   web1.info,web2.info                   80      29s
# kubectl describe ing example-ingress
Name:             example-ingress
Namespace:        default
Address:
Default backend:  web-default:8080 (<none>)
Rules:
  Host       Path  Backends
  ----       ----  --------
  web1.info
             /   web1:8080 (<none>)
  web2.info
             /   web2:8080 (<none>)
Annotations:
  kubectl.kubernetes.io/last-applied-configuration:  {"apiVersion":"networking.k8s.io/v1beta1","kind":"
Ingress","metadata":{"annotations":{},"name":"example-ingress","namespace":"default"},"spec":{"backend":
{"serviceName":"web-default","servicePort":8080},"rules":[{"host":"web1.info","http":{"paths":
[{"backend":{"serviceName":"web1","servicePort":8080},"path":"/"}]}},{"host":"web2.info","http":{"paths":
[{"backend":{"serviceName":"web2","servicePort":8080},"path":"/"}]}}]}}

Events:
  Type    Reason  Age   From                      Message
  ----    ------  ----  ----                      -------
  Normal  CREATE  33s   nginx-ingress-controller  Ingress default/example-ingress
  Normal  UPDATE  28s   nginx-ingress-controller  Ingress default/example-ingress

# kubectl delete ing example-ingress
ingress.extensions "example-ingress" deleted
#
```

说明：

- 在定义ingress资源时，如果不指定host选项，” HOSTS “一栏将为星号。此时，可以使用任何域名进行访问，同时还可以直接使用IP进行访问
- 由于集群里还没有Ingress Controller，故ingress的“ADDRESS”一栏为空

1.4 ingress TLS

- 当前只支持配置单个TLS，即只支持配置一个secret
- 如果在Ingress配置了多个host(多个域名），需要在证书中配置支持多个host(多个域名)
- Ingress TLS配置格式，如下：

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-tls
  namespace: default
spec:
  tls:
  - hosts:
    - web1.info
    - web2.info
    secretName: ingress-tls
  backend:
    serviceName: web-default
    servicePort: 8080
  rules:
  - host: web1.info
    http:
      paths:
      - path: /
        backend:
          serviceName: web1
          servicePort: 8080
  - host: web2.info
    http:
      paths:
      - path: /
        backend:
          serviceName: web2
          servicePort: 8080
```

secret ingress-tls需要包含 tls.crt 和 tls.key：(可以通过`kubectl create secret tls ingress-tls --key xxx --cert xxx`创建tls secret)

```
apiVersion: v1
kind: Secret
metadata:
  name: ingress-tls
  namespace: default
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
type: kubernetes.io/tls
```

1.5 ingress限制

- ingress资源仅支持配置HTTP流量的规则，详见：https://kubernetes.io/docs/concepts/services-networking/ingress/#the-ingress-resource
- ingress资源无法配置一些高级特性，如负载均衡的算法，sessions affinity等，这些都需要在Ingress Controller中配置

2 Ingress Controller基本介绍

2.1 Ingress Controller功能

- Ingress Controller通过watch APIServer /ingresses 获取ingress资源定义
- Ingress Controller基于ingress的定义，生成load balancer(如nginx）所需的配置文件(如/nginx.conf)
- Ingress Controller重新加载load balancer的配置文件内容(如nginx重新加载： nginx -s reload)

2.2 Ingress Controller选择

- Ingress Controller的现实有多个选择，当前K8S官方维护的是: ingress-nginx
- 关于不同Ingress Controller之间的区别，可查看：https://www.kubernetes.org.cn/5948.html
- 在K8S集群可同时部署多个Ingress Controller，但此时在定义Ingress时应该使用annotate来表示该应该由哪个Ingress Controller来负责实现，如：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
  namespace: default
  annotations:
    kubernetes.io/ingress.class: "nginx"  <=== nginx Ingress Controller
spec:
  backend:
    serviceName: web
    servicePort: 8080
```

3 Ingress-nginx基本介绍

3.1 安装部署

- 部署nginx-ingress-controller

```
# git clone https://github.com/kubernetes/ingress-nginx.git
# cd ingress/deploy/static
# kubectl apply -f mandatory.yaml
namespace/ingress-nginx created
configmap/nginx-configuration created
configmap/tcp-services created
configmap/udp-services created
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created
deployment.apps/nginx-ingress-controller created
#
```

说明：

  - 当前采用YAML文件直接部署，也可以采用helm去部署ingress-ngnix
  - 当前采用YAML部署的是Deployment，也可以部署为DaemonSet
  - `namespace/configmap/serviceaccount/clusterrole/deployment`

- 暴露对外访问

```
# cat provider/baremetal/service-nodeport.yaml
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
spec:
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
    - name: https
      port: 443
      targetPort: 443
      protocol: TCP
  selector:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
# kubectl apply -f provider/baremetal/service-nodeport.yaml
service/ingress-nginx created
# kubectl  get svc -n ingress-nginx
NAME            TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx   NodePort   11.111.90.195   <none>        80:32144/TCP,443:32655/TCP   25s
#
```

说明：当前默认采用nodeport方式暴露对外端口

3.2 暴露服务方式

- nodePort

```
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
spec:
# this setting is to make sure the source IP address is preserved
  externalTrafficPolicy: Local
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
    - name: https
      port: 443
      targetPort: 443
      protocol: TCP
```

说明：

  - NodePort默认访问端口为30000-32767，可通过参数`--service-node-port-range`来调整支持80和443，但强烈不建议这么做
  - 由于NodePort默认采用NAT转发数据包，使得K8S Pod收到的数据包中的源IP非Client的真实IP地址，而是K8S节点IP地址(收到数据包的那个节点)
  - 采用`externalTrafficPolicy: Local`可以使得K8S Pod获取Client的真实IP地址，但也会使得当前K8S节点如果没有响应的Pod，在收到Client请求后，会丢掉数据包，不会转发到其他节点

- hostNetwork

```
template:
  spec:
    hostNetwork: true
    dnsPolicy: ClusterFirstWithHostNet
```

说明：

- 由于使用hostNetwork，会直接在Host上监听Pod启动的端口
- 由于Host上无法监听两个相同的端口，在运行多副本的情况下，需要确保Pod不能调度到同一个节点上（可以采用DaemonSet或者Pod anti-affinity来解决）
- 使用hostNetwork后，Pod将直接使用Host的DNS解析，无法使用K8S内部的DNS解析，可使用`dnsPolicy: ClusterFirstWithHostNet`来使得Pod能够使用K8S内部DNS解析

- hostPort
  hostPort是通过iptables实现，不会直接在Host上监听端口，功能与hostNetwork基本一致，一般用的较少
- 其他方式
  - MetalLB
  - External IPs
  - LoadBalance

3.3 高级特性

- 支持4层代理
  ingress配置只支持7层代理，无法支持4层代理。但nginx本身是支持4层代理的，可通过配置tcp-services和udp-services这两个configmap来支持4层代理：

```
      containers:
        - name: nginx-ingress-controller
          image: quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.26.1
          args:
            - /nginx-ingress-controller
            - --configmap=$(POD_NAMESPACE)/nginx-configuration
            - --tcp-services-configmap=$(POD_NAMESPACE)/tcp-services         <===
            - --udp-services-configmap=$(POD_NAMESPACE)/udp-services         <===
            - --publish-service=$(POD_NAMESPACE)/ingress-nginx
            - --annotations-prefix=nginx.ingress.kubernetes.io
```

说明：

- 默认的tcp-services configmap为：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: tcp-services
  namespace: ingress-nginx
#data:
#  9000: "default/web:8080"
```

  注：data的格式为` listen_port: "<namespace/service_name>:<service_port>"`
- 默认的udp-services configmap为：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: udp-services
  namespace: ingress-nginx
#data:
#  53: "kube-system/kube-dns:53"
```

- 代理https协议
nginx默认代理到后端的Service都是HTTP协议，添加`nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"`，可使得nginx代理https协议

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  namespace: kubernetes-dashboard
  name: dashboard-ingress
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"     <===
spec:
  rules:
  - host: hello-dashboard.info
    http:
      paths:
      - path: /
        backend:
          serviceName: kubernetes-dashboard
          servicePort: 443
```

说明：

- 默认情况下，ingress-nginx会默认创建一个TLS证书，监听在443端口
- 细节参考：https://stackoverflow.com/questions/48324760/ingress-configuration-for-dashboard
- 更多annotations使用参考：https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/

3.4 实现细节

- nginx频繁重新加载配置会影响response latency和load balancing功能
- nginx-ingress通过Lua模块(lua-nginx-module)主动发送Post请求到nginx去更新upstream配置，避免仅因upstream变化导致nginx重新加载配置
- ingress配置错误时(如配置了错误的annotation语法，但在ingress的角度来看是正常的)，会导致生成的配置无法被nginx被重新加载
- 为了避免ingress错误配置导致nginx无法重新加载，ingress-nginx提供了Validating webhook(默认是关闭的)来检查ingress创建的配置是nginx可重新加载的

4 Ingress使用

4.1 准备工作

- 创建一个web deployment和service

```
# kubectl run web --image=gcr.io/google-samples/hello-app:1.0 --port=8080
# kubectl expose deployment web --target-port=8080
# kubectl  get deployment web
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
web    1/1     1            1           44h
# kubectl  get svc  web
NAME   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
web    ClusterIP   11.110.123.54   <none>        8080/TCP   2m20s
#
```

- 更新集群节点/etc/hosts文件，如下：

```
# cat >>/etc/hosts <<-EOF
> 11.111.90.195  web1.info
> 11.111.90.195  web2.info
> 11.111.90.195  web3.info
> EOF
# tail -3 /etc/hosts
11.111.90.195  web1.info
11.111.90.195  web2.info
11.111.90.195  web3.info
#
```

说明： 11.111.90.195 为ingress-nginx的cluster ip

4.2 域名访问

- 创建ingress

```
# cat example2-ingress.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: example2-ingress
  namespace: default
spec:
  backend:
    serviceName: web
    servicePort: 8080
  rules:
  - host: web1.info
    http:
      paths:
      - path: /abc
        backend:
          serviceName: web
          servicePort: 8080
  - host: web2.info
    http:
      paths:
      - path: /abc
        backend:
          serviceName: web
          servicePort: 8080
# kubectl apply -f example2-ingress.yaml
ingress.networking.k8s.io/example2-ingress created
#
```

- 访问测试

```
# kubectl get ing
NAME               HOSTS                 ADDRESS         PORTS   AGE
example2-ingress   web1.info,web2.info   11.111.90.195   80      70
# curl web1.info/abc
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl web2.info/abc
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl web1.info/edf
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl web2.info/edf
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl web3.info/edf
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>openresty/1.15.8.2</center>
</body>
</html>
# curl 11.111.90.195/abc -H "Host: web1.info"
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl 11.111.90.195/abc -H "Host: web2.info"
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl 11.111.90.195/abc -H "Host: web3.info"
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>openresty/1.15.8.2</center>
</body>
</html>
# kubectl delete ing example2-ingress
ingress.extensions "example2-ingress" deleted
#
```

说明：

- 只可以通过web1.info和web2.info访问
- 在匹配rules中的path都失败时，会使用默认的backend配置的服务

4.3 IP访问

- 创建ingress

```
# cat example3-ingress.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: example3-ingress
  namespace: default
spec:
  backend:
    serviceName: web
    servicePort: 8080
  rules:
  - http:
      paths:
      - path: /abc
        backend:
          serviceName: web
          servicePort: 8080
# kubectl apply -f example3-ingress.yaml
ingress.networking.k8s.io/example3-ingress configured
#
```

- 访问测试：

```
# kubectl get ing
NAME              HOSTS   ADDRESS          PORTS   AGE
example3-ingress   *      11.111.90.195   80      5m12s
# curl 11.111.90.195/abc
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl 11.111.90.195/abc  -H "Host: abc.com"
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl 11.111.90.195/edf
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>openresty/1.15.8.2</center>
</body>
</html>
# curl 11.111.90.195/edf  -H "Host: abc.com"
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>openresty/1.15.8.2</center>
</body>
</html>
# kubectl delete ing example3-ingress
ingress.extensions "example3-ingress" deleted
#
```

说明：

- 当ingress定义时没有配置rules的host选项，可通过访问nginx的Cluster IP来可以直接访问web服务
- 默认的backend不可通过IP去访问，必须使用域名，但是由于使用的域名必须是rules中定义的host。如果上面的示例没有配置host，所以永远不会走默认的backend路由

4.4 启用TLS

- 创建证书
  - 生成CA自签证书

```
# mkdir cert && cd cert
# openssl genrsa -out ca-key.pem 2048
# openssl req -x509 -new -nodes -key ca-key.pem -days 10000 -out ca.pem -subj "/CN=kube-ca"
```

- 更新openss配置

```
cp /etc/pki/tls/openssl.cnf .
vim openssl.cnf

#
[req]
req_extensions = v3_req #
#
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = web1.info   #  web1.info  web2.info
DNS.2 = web2.info
```

- 生成证书

```
# openssl genrsa -out ingress-key.pem 2048
# openssl req -new -key ingress-key.pem -out ingress.csr -subj "/CN=kube-ingress" -config openssl.
cnf
# openssl x509 -req -in ingress.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -out ingress.pem -
days 365 -extensions v3_req -extfile openssl.cnf
```

- 查看证书

```
# ls
ca-key.pem  ca.pem  ca.srl  ingress.csr  ingress-key.pem  ingress.pem  openssl.cnf
# openssl x509 -noout -text -in ingress.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e8:a2:4e:02:4d:3e:c5:9c
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=kube-ca
        Validity
            Not Before: Dec  4 09:25:50 2019 GMT
            Not After : Dec  3 09:25:50 2020 GMT
        Subject: CN=kube-ingress
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c5:04:70:dc:d4:66:e4:a8:59:2b:92:39:53:cb:
                    0d:97:5b:b3:4e:f2:e0:ed:a1:96:bc:4c:db:27:88:
                    30:3f:cf:ee:f7:7d:e9:42:8b:67:b1:d3:0e:77:b9:
                    12:89:4f:fd:e8:72:b7:61:5a:b2:0d:82:60:40:74:
                    2a:bd:96:d2:75:2e:50:44:1e:1f:b0:33:53:99:f7:
                    b5:0f:2b:17:54:4d:72:21:99:98:0d:86:a5:84:32:
                    22:05:94:94:f0:d2:06:82:58:3f:2d:59:86:7a:b9:
                    e0:e9:9c:b7:61:66:cd:50:fe:30:6e:24:3b:5b:c9:
                    64:db:16:c1:b2:39:40:97:b9:a5:d6:2c:be:f5:f1:
                    e9:7c:06:7f:a8:7c:32:21:09:4c:81:34:4c:72:34:
                    24:0f:d0:ec:9f:57:e4:82:d3:54:2c:e1:39:40:1f:
                    a4:d0:5c:83:6d:70:2f:bf:e4:64:e4:ae:32:33:39:
                    52:c0:1d:1d:52:a7:2b:b1:7b:cb:cb:bb:31:f9:10:
                    08:95:04:67:e6:68:ed:8f:dc:21:6a:19:b4:b9:4e:
                    71:d1:b1:32:d1:51:34:0a:67:49:89:77:a4:07:55:
                    90:03:0f:25:ff:14:24:6d:83:89:6d:9a:0f:10:89:
                    6d:bb:7f:1f:b0:21:54:00:53:71:d1:0b:62:7b:db:
                    e9:bd
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:web1.info, DNS:web2.info
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
    Signature Algorithm: sha256WithRSAEncryption
         6e:29:95:c7:72:42:8f:5b:c6:78:9e:b3:8c:95:0d:4e:6c:af:
         ca:b1:81:8a:9d:0d:15:cb:2e:22:0a:63:08:d2:02:dc:39:10:
         bf:83:1a:0d:58:ae:96:38:40:ce:6e:5d:18:c7:84:b1:fc:9c:
         d0:ce:91:22:45:d1:f1:b1:dd:26:9c:a7:3c:ee:87:76:a3:43:
         df:7d:6a:f6:be:60:5d:32:f7:66:49:df:17:11:1d:89:46:6e:
         ed:01:ca:d9:9b:c5:26:b3:57:75:e6:78:5e:37:ea:9e:7b:f3:
         32:5e:3d:d6:25:5f:61:84:9b:13:93:b5:48:71:ae:5e:2d:ab:
         a2:e6:29:a6:45:98:78:3c:ab:66:70:e0:d4:b7:d5:f1:e5:6a:
         ef:e0:e0:63:57:97:f1:c6:f0:b6:c3:f0:fa:44:15:49:9e:18:
         89:1a:6f:8e:9d:58:c8:f1:9e:24:b9:6e:77:1b:ac:64:0e:f2:
         cb:9b:8f:86:55:ce:27:44:2e:cb:f9:a5:84:5e:c5:02:0c:e8:
         8c:d5:da:5e:5e:a3:3b:0f:d7:3e:26:96:6a:98:4d:55:6b:ac:
         55:d8:3f:13:ec:93:b7:ce:b7:52:ef:f5:11:25:5c:af:cf:67:
         75:ee:22:b4:e0:fc:7a:02:de:15:78:f4:a0:a0:ca:00:37:e9:
         45:f5:7b:e2
#
```

- 创建secret

```
# kubectl create secret tls ingress-tls --key ingress-key.pem  --cert ingress.pem
secret/ingress-tls created
#
```

- 创建ingress

```
# cat example4-ingress.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: example4-ingress
spec:
  tls:
  - hosts:
    - web1.info
    - web2.info
    - web3.info
    secretName: ingress-tls
  rules:
  - host: web1.info
    http:
      paths:
      - path: /abc
        backend:
          serviceName: web
          servicePort: 8080
  - host: web2.info
    http:
      paths:
      - path: /abc
        backend:
          serviceName: web
          servicePort: 8080
  - host: web3.info
    http:
      paths:
      - path: /abc
        backend:
          serviceName: web
          servicePort: 8080
# kubectl apply -f yilong-ingress.yaml
ingress.networking.k8s.io/example2-ingress created
#
```

- 访问测试：

```
# kubectl get ingress
NAME               HOSTS                            ADDRESS         PORTS     AGE
example4-ingress   web1.info,web2.info,web3.info   11.111.90.195   80, 443   111s
[root@app static]# curl https://web1.info/abc --cacert ca.pem
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
# curl https://web2.info/abc --cacert ca.pem
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
[root@app static]# curl https://web3.info/abc --cacert ca.pem
curl: (60) Issuer certificate is invalid.
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.
# kubectl delete ing example4-ingress
ingress.extensions "example4-ingress" deleted
#
```

说明：由于证书中没有配置支持域名web3.info，所以访问时出错

4.5 4层代理

- 更新tcp-services

```
# cat configmap-tcp-services.yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: tcp-services
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
data:
  9000: "default/web:8080"
# kubectl apply -f configmap-tcp-services.yaml
configmap/tcp-services configured
#
```

注：可配置多个监听端口

- 查看监听端口(在nginx-ingress-controller运行节点运行)

```
# netstat  -ntlp | grep 9000
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN      3118264/nginx: mast
tcp6       0      0 :::9000                 :::*                    LISTEN      3118264/nginx: mast
#
```

- 访问测试(在nginx-ingress-controller运行节点运行)

```
# hostname -i
192.168.50.113
# curl  192.168.50.113:9000
Hello, world!
Version: 1.0.0
Hostname: web-6d4657545d-2qll5
#
```

4.6 https转发

- 创建ingress

```
# cat example5-ingress.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  namespace: kubernetes-dashboard
  name: example5-ingress
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  rules:
  - host: k8s-dashboard.info
    http:
      paths:
      - path: /
        backend:
          serviceName: kubernetes-dashboard
          servicePort: 443
# kubectl apply -f example5-ingress.yaml
ingress.networking.k8s.io/example5-ingress created
#
```

注：使用了annotations: `nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"`

- 访问测试

```
# kubectl get ing -n kubernetes-dashboard
NAME                HOSTS                ADDRESS       PORTS   AGE
example5-ingress    k8s-dashboard.info   11.111.90.195   80      2m17s
# echo "11.111.90.195  k8s-dashboard.info" >> /etc/hosts
# tail -1 /etc/hosts
11.111.90.195 k8s-dashboard.info
# curl  -k https://k8s-dashboard.info
<!--
Copyright 2017 The Kubernetes Authors.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->

<!doctype html>
<html>

<head>
  <meta charset="utf-8">
  <title>Kubernetes Dashboard</title>
  <link rel="icon"
        type="image/png"
        href="assets/images/kubernetes-logo.png" />
  <meta name="viewport"
        content="width=device-width">
<link rel="stylesheet" href="styles.091eb6c35947fb31b457.css"></head>

<body>
  <kd-root></kd-root>
<script src="runtime.3c37eeed2119f000b256.js" defer></script><script src="polyfills-es5.
fbc7b8163a3af6d5e025.js" nomodule defer></script><script src="polyfills.8fbdb061b9fee8b5a878.js" defer><
/script><script src="scripts.f72393e836125e26b85a.js" defer></script><script src="main.
dba02aeed05a3f2a5a0c.js" defer></script></body>

</html>
# kubectl delete example5-ingress ing -n kubernetes-dashboard
ingress.extensions "example5-ingress" deleted
#
```

5 参考

- https://kubernetes.io/docs/concepts/services-networking/ingress/
- https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/
- https://kubernetes.github.io/ingress-nginx/
- https://mritd.me/2017/03/04/how-to-use-nginx-ingress/