SSL configuration on Ingress

Step: 1

First generate the SSL with crt and key or if you already have Cert and key provided by the client.

TLS Secrets

Anytime we reference a TLS secret, we mean a PEM-encoded X.509, RSA (2048) secret.

You can generate a self-signed certificate and private key with:

[root@master ssl\_default]# kubectl create secret tls ingress-nginx-ssl --key key.pem --cert cert.pem

Or if you have let’s encrypt cert or have CA signed cert that can be also user. The thing required is key.pem and cert.pem.

If want to convert client.key >>> client.pem

openssl rsa -in client.key -out key.unencrypted.pem

## Command Line Method

Now lets create secret

[root@master ssl\_default]# ls

cert.pem key.pem

#kubectl create secret tls ingress-nginx-ssl --key key.pem --cert cert.pem

secret/tls-secret created

## Config file Method

## Encode your ssl certs with base64

I assume that you have two ssl certs file one is nginx.key other is nginx.crt Create base64 encoded version of the both file. I trimmed to output for better reading.

the trick is that you have to base64 encode the key and certificate data. In Bash:

|  |  |
| --- | --- |
|  | cat tls.crt | base64  cat tls.key | base64 |

Paste each piece of base64 encoded data into the appropriate sections of the YAML file as one line. Make sure your text editor doesn’t add any carriage returns to wrap the lines.

**$ base64 nginx.key**   
LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1JSUV2QUlCQURBTkJna3Foa2lHOXcwQkFRRUZBQVNDQktZd2dnU2lBZ0VBQW9JQkFRQ1lrL2hMaEMzalh2Y3kKUHY1VDdNcU1OMWR5STlQNVM5MlpUUllNT1VZb2JiUXREeE1KbWxMd3g4c0owQURlWjVzTWRSQkYwWjJzNVBrMApHL3V2d2c2c2JpSTFCaXVqaVBzdnRwWVpIaC9nZVdJUG5zSlk5dWpJenFyZ3Q0UUoxNzkvRjhncjliVUpJdlNQCnZ2YTQycjRFMEdoUzFnaVNUWENSbk…

**$ base64 nginx.crt**   
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURHRENDQWdBQ0NRRHJDajdxWHFhR1VqQU5CZ2txaGtpRzl3MEJBUXN…

## Create a ssl secret file

**$ cat sslsecret.yml**  
apiVersion: v1

data:

tls.crt: 

tls.key: 

kind: Secret

metadata:

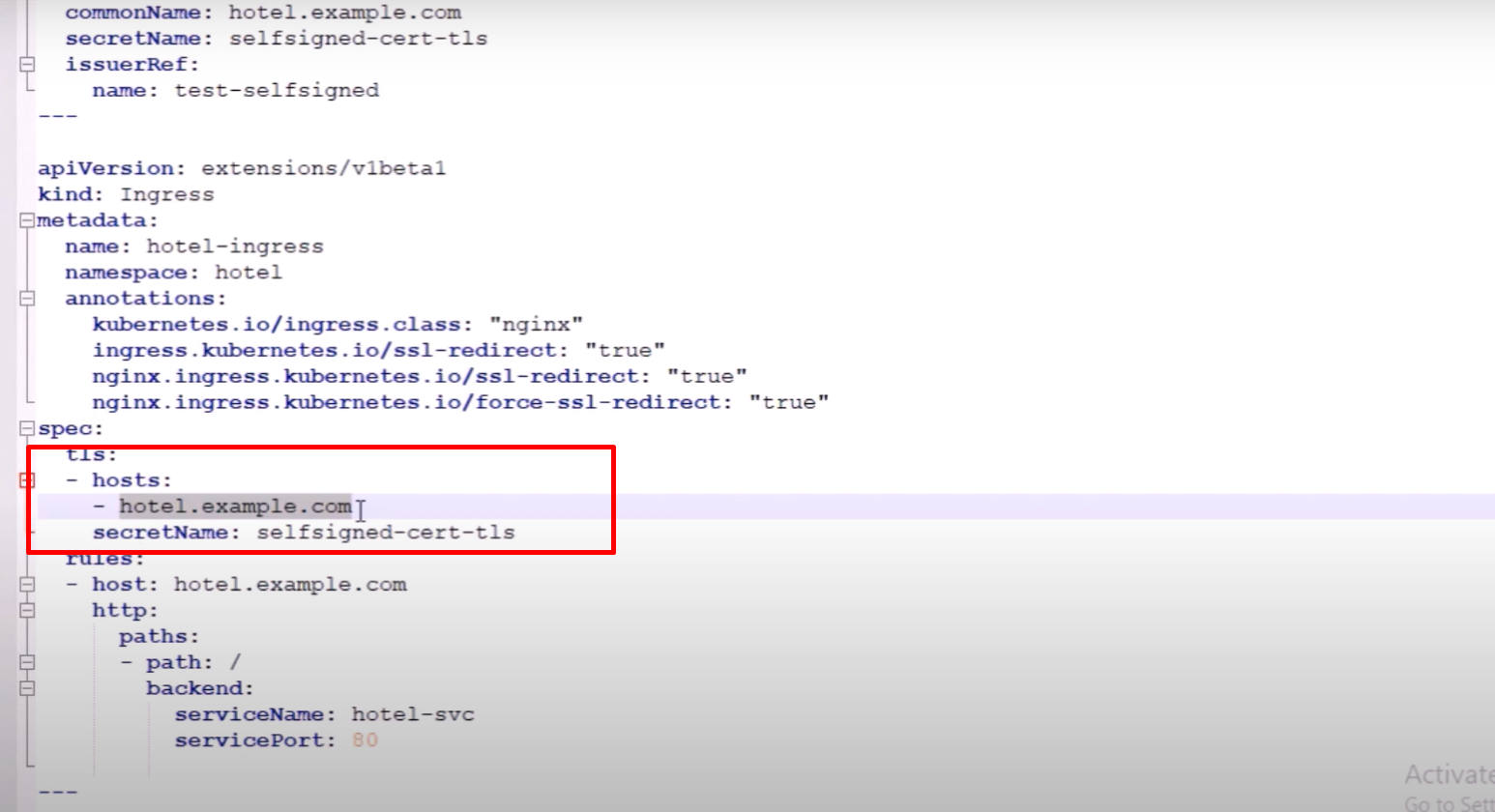
name: shaukat-ingress-nginx

namespace: default

type: kubernetes.io/tls --------🡪 FOT SSL data for a TLS client or server

| **Builtin Type** | **Usage** |
| --- | --- |
| Opaque | arbitrary user-defined data |
| kubernetes.io/service-account-token | service account token |
| kubernetes.io/dockercfg | serialized ~/.dockercfg file |
| kubernetes.io/dockerconfigjson | serialized ~/.docker/config.json file |
| kubernetes.io/basic-auth | credentials for basic authentication |
| kubernetes.io/ssh-auth | credentials for SSH authentication |
| kubernetes.io/tls | data for a TLS client or server |
| bootstrap.kubernetes.io/token | bootstrap token data |

Use this secret in ingress resource (rule)



### **Ingress yaml**

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: hotel-ingress

namespace: hotel

annotations:

kubernetes.io/ingress.class: "nginx"

ingress.kubernetes.io/ssl-redirect: "true"

nginx.ingress.kubernetes.io/ssl-redirect: "true"

nginx.ingress.kubernetes.io/force-ssl-redirect: "true"

spec:

tls:

- hosts:

- hotel.example.com

secretName: selfsigned-cert-tls

rules:

- host: hotel.example.com

http:

paths:

- path: /

backend:

serviceName: hotel-svc

servicePort: 80



## Deployment and service yaml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: hotel

namespace: hotel

spec:

replicas: 2

selector:

matchLabels:

app: hotel

template:

metadata:

labels:

app: hotel

spec:

containers:

- name: hotel

image: nginxdemos/hello:plain-text

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: hotel-svc

namespace: hotel

spec:

ports:

- port: 80

targetPort: 80

protocol: TCP

name: http

selector:

app: hotel