Installation of Traefik on an kubernetes (k8s) cluster.

Dennis Lomans

Before you start make sure you have `helm` installed.

NOTE

For the examples the url `*.traefik.demo` is used. Be sure to add `*.traefik.demo` to you host(s) file, e.g. `127.0.0.1 *.traefik.demo`

When reading the trying the examples I assue you change to the directory of the example.

Installation using helm

```
helm repo add traefik https://traefik.github.io/charts
helm repo update
helm install traefik traefik/traefik
```

Traefik Dashboard

Traefik comes with a dashboard which can either be exposed via port 9000 or via an *IngressRoute*.

CAUTION	When the URL to the dashboard is entered manually do not foget the forward slash '/' after dashboard, otherwise you'll be presented with a nice `404` error message.

Exposure via port 9000

To expose the dashboard via port 9000, run the following command in a new terminal.

```
kubectl port-forward $(kubectl get pods --selector "app.kubernetes.io/name=traefik"
--output=name) 9000:9000
```

NOTE	For a local kubernetes cluster, the dashboard is accessible from here.

Exposure via the IngressRoute

To expose the dashboard via the *IngressRoute*, run the following command in a new terminal.

```
kubectl.exe apply -f .\dashboard.yaml
```

The dashboard will be available on the default http port.

NOTE	For a local kubernetes cluster, the dashboard is accessible from here.

Simple HTTP Proxy

Creating a deployment and a service

The file `service-deployment.yaml` contains a simple depoyment and service.

In a terminal run the command:

```
kubectl.exe apply -f ./service-deployment.yaml
```

Exposing the service via an IngressRoute

The yaml file:

```
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: http-proxy
spec:
  entryPoints:
    - web
  routes:
    - match: Host(`http.traefik.me`)
      kind: Rule
      services:
        - name: traefic-demo
          port: 80
```

Where the entrypoint web is the default for port 80. In a terminal run the command:

```
kubectl.exe apply -f ./http-proxy.yaml
```

The http-demo should be available on here

Simple HTTPS Proxy

Creating a deployment and a service

The file `service-deployment.yaml` contains a simple depoyment and service.

In a terminal run the command:

```
  kubectl.exe apply -f ./service-deployment.yaml
```

The deployed service should then be available on here

Certificates

Before we can add an *IngressRoute*, we need to add server certificates to the k8s cluster. Assuming you have openssl installed, if not please install it.

Creating certificates using the command :

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
-subj "/CN=https.traefik.me"
```

Create a k8s secret using the server credential file and key file :

```
kubectl create secret tls https-cert-store --key .\server.key --cert .\server.crt
```

Exposing the service via an IngressRoute

The yaml file:

```
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: https-proxy
spec:
  entryPoints:
    - websecure
  routes:
    - match: Host(`https.traefik.me`)
      kind: Rule
      services:
        - name: traefic-demo
          port: 80
  tls:
```

```
    secretName: https-cert-store
```

Where the entrypoint web is the default for port 80. In a terminal run the command:

```
kubectl.exe apply -f ./https-proxy.yaml
```

The http-demo should be available on here

You will be presented with the warning screen, I assume you know how to deal ith that.

Simple mTls Proxy

Creating a deployment and a service

The file `service-deployment.yaml` contains a simple depoyment and service.

In a terminal run the command:

```
kubectl.exe apply -f ./service-deployment.yaml
```

The deployed service should then be available on here

Certificates

Before we can add an *IngressRoute*, we need to add server certificates to the k8s cluster. Assuming you have openssl installed, if not please install it.

Creating certificates:

For mTls you need multiple certificates.

Needed certificates

1. CA Root Certificate
   The one certificate we all trust
2. Signed Client certificate
   The certificate for the server to trust the client
3. Server certificate The certificate for the client to trust the server

Root CA Certificate

```
openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes
-subj '/CN=My Cert Authority'
```

Signed Client certificate

Steps to take

- Create a *normal* client certificate

```
openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj
/CN=My Client
```

- Sign the certificate with the Root CA

```
openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key
-set_serial 02 -out client.crt
```

- Create the Server certificate

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
-subj "/CN=mtls.traefik.me"
```

- Create a k8s CA secret using the CA credential:

```
kubectl create secret generic secret-ca --from-file=ca.crt
```

- Create a k8s secret using the server credential file and key file :

```
kubectl create secret tls mtls-cert-store --key .\server.key --cert .\server.crt
```

Exposing the service via an IngressRoute

The yaml file:

```
apiVersion: traefik.io/v1alpha1
kind: TLSOption
metadata:
  name: mtlsoption
  namespace: default

spec:
  minVersion: VersionTLS12
  maxVersion: VersionTLS13
  curvePreferences:
    - CurveP521
    - CurveP384
  cipherSuites:
    - TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  clientAuth:
    secretNames:
      - secret-ca
    clientAuthType: VerifyClientCertIfGiven
  sniStrict: false
```

```
---
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: mtls-proxy
  annotations:
spec:
  entryPoints:
    - websecure
  routes:
    - match: Host(`mtls.traefik.me`)
      kind: Rule
      services:
        - name: traefic-demo
          port: 80
  tls:
    secretName: mtls-cert-store
    options:
      name: mtlsoption
      namespace: default
```

Where the entrypoint web is the default for port 80. In a terminal run the command:

```
  kubectl.exe apply -f ./mts-proxy.yaml
```

The http-demo should be available on here

You will be presented with the warning screen, I assume you know how to deal ith that.