

# Traefik Exercise

[Pre reqs](#)

[First Challenge](#)

[Deploying an MVP](#)

[Second Challenge](#)

[The K8s-native Fix](#)

[Integrating Traefik](#)

[The Third Challenge](#)

[Improvements](#)

[Time for Security](#)

[Auditing](#)

[Observability and Metrics](#)

[Redeploying Using the Local Built Image](#)

[Refs](#)

## Pre reqs

- Linux (WSL2 with Ubuntu in my case): <https://learn.microsoft.com/en-us/windows/wsl/install>
- Kubernetes (Docker Desktop engine with kubeadm):  
<https://www.docker.com/blog/how-to-set-up-a-kubernetes-cluster-on-docker-desktop/>
- Helm: <https://helm.sh/docs/intro/install/>
- Git and Github: <https://github.com/gustavosaviano/traefik-exercise>
- foobar-api: <https://github.com/containous/foobar-api>

## First Challenge

Right from the start, I noticed the `containous/foobar-api` image was no longer available in Docker Hub repos. When inspecting the manifest, this is what I received:

```
gus@coxa-pc:~$ docker manifest inspect containous/foobar-api
errors:
denied: requested access to the resource is denied
unauthorized: authentication required

gus@coxa-pc:~$ docker manifest inspect traefik/foobar-api
errors:
denied: requested access to the resource is denied
unauthorized: authentication required
```

To work around this, I initially created a Dockerfile to build the image myself and pushed it to the Docker Desktop engine. While this worked, it added an extra step to the workflow (more on this later on).

Later, while double-checking the image availability again, I found a similar image in Fernando's repo. Since Fernando appears to be the same person listed in the tech meeting invite, I decided to use this image ([fernandobenegasa/foobar-api](#)):

```
gus@coxa-pc:~$ docker search fernandobenegasa/foobar-api
NAME                                     DESCRIPTION
STARS          OFFICIAL
fernandobenegasa/foobar-api
```

## Deploying an MVP

I decided to go in small, incremental steps. My goal was to get the process flowing and show my thought throughout the exercise. So to start, I deployed a MVP of the [foobar-api](#) just to see the application running:

```
gus@coxa-pc:~/traefik-exercise$ cat 01-foobar.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: foobar-api
spec:
```

```

replicas: 1
selector:
  matchLabels:
    app: foobar-api
template:
  metadata:
    labels:
      app: foobar-api
spec:
  containers:
    - name: foobar-api
      image: fernandobenegasa/foobar-api
      ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: foobar-api-svc
spec:
  selector:
    app: foobar-api
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80

```

```

gus@coxa-pc:~/traefik-exercise$ k apply -f 01-foobar.yaml
deployment.apps/foobar-api created
service/foobar-api-svc created

```

```

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                  READY   STATUS             RES
TARTS   AGE
foobar-api-6446857f59-mpk2z   0/1     ContainerCreating   0

```

4s

```
gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                      READY   STATUS    RESTARTS
AGE
foobar-api-6446857f59-mpk2z   0/1     Error      2 (18s ago)
27s
```

## Second Challenge

Well, that didn't go as expected. Right after deployment, the `foobar-api` pod entered an Error status and fell into a `CrashLoopBackOff` loop:

```
gus@coxa-pc:~/traefik-exercise$ k get pod foobar-api-6446857f
59-mpk2z -o yaml | tail -30
  terminated:
    containerID: docker://1c39f6a569c2f7e4dcac02e6b872450
1b0f1c528cd59a6cd374958e7f066f45c
    exitCode: 1
    finishedAt: "2026-01-17T20:34:28Z"
    reason: Error
    startedAt: "2026-01-17T20:34:28Z"
  name: foobar-api
  ready: false
  resources: {}
  restartCount: 5
  started: false
  state:
    waiting:
      message: back-off 2m40s restarting failed container=
oobar-api pod=foobar-api-6446857f59-mpk2z_default(48d8bdc0-de
2c-411d-b80b-d47f8775bc3f)
      reason: CrashLoopBackOff
  volumeMounts:
  - mountPath: /var/run/secrets/kubernetes.io/serviceaccoun
```

```
t
  name: kube-api-access-vm64f
  readOnly: true
  recursiveReadOnly: Disabled
hostIP: 192.168.65.3
hostIPs:
- ip: 192.168.65.3
observedGeneration: 1
phase: Running
podIP: 10.x.x.xx
podIPs:
- ip: 10.x.x.xx
qosClass: BestEffort
startTime: "2026-01-17T20:31:20Z"
```

So it was time to troubleshoot. The pod logs provided the answer right away: the application was failing because it required a certificate.

```
gus@coxa-pc:~/traefik-exercise$ k logs foobar-api-6446857f59-
mpk2z
Starting up on port 443
2026/01/17 20:37:19 You need to provide a certificate
```

Checking the `app.go` source code confirmed this: the requirement for `cert.pem` and `key.pem` is hardcoded and the application expects these files to exist in the `/cert` directory before it can start:

```
gus@coxa-pc:~/foobar-api$ grep -C 3 "You need to provide a ce
rtificate" app.go

_, err := os.Stat("/cert/cert.pem")
if err != nil {
    log.Fatal("You need to provide a certificat
e")
}
```

```

_, err = os.Stat("/cert/key.pem")
if err != nil {
    log.Fatal("You need to provide a certificate")
}
log.Fatal(server.ListenAndServeTLS("/cert/cert.pem",
"/cert/key.pem"))
}

```

So, unfortunately, my initial idea of deploying an MVP didn't make it. However, this was a good learning moment. I now had to find a way to make it work without modifying the code itself to go through the certificate check.

## The K8s-native Fix

I wanted to keep it simple and stick to native K8s solutions. I considered using `cert-manager` but that would require external installations and extra steps. Instead, I went for an InitContainer.

So I edited the `01-foobar.yaml` to include:

1. A PVC to hold the `cert.pem` and `key.pem` files.
2. An initContainer running a simple script to check if the files exist in the `/cert` volume and, if not, use `openssl` to generate a self-signed certificate and key:

```

gus@coxa-pc:~/traefik-exercise$ cat 01-foobar.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: foobar-api-certs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
  ---

```

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: foobar-api
  labels:
    app: foobar-api
spec:
  replicas: 1
  selector:
    matchLabels:
      app: foobar-api
  template:
    metadata:
      labels:
        app: foobar-api
    spec:
      volumes:
        - name: cert-volume
          persistentVolumeClaim:
            claimName: foobar-api-certs-pvc

      initContainers:
        - name: cert-gen
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args:
            - |
              apk add openssl;
              if [ ! -f /cert/cert.pem ]; then
                openssl req -x509 -newkey rsa:4096 -keyout /cert/key.pem -out /cert/cert.pem -days 365 -nodes -subj '/CN=foobar-api.local';
              else
                echo "Cert already exists";
              fi
      volumeMounts:

```

```

    - name: cert-volume
      mountPath: /cert

  containers:
    - name: foobar-api
      image: fernandobenegasa/foobar-api:latest
      ports:
        - containerPort: 80
      volumeMounts:
        - name: cert-volume
          mountPath: /cert

---
apiVersion: v1
kind: Service
metadata:
  name: foobar-api-svc
spec:
  selector:
    app: foobar-api
  ports:
    - protocol: TCP
      port: 443
      targetPort: 443

```

```

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                  READY   STATUS    REST
ARTS      AGE
foobar-api-6446857f59-mpk2z   0/1     CrashLoopBackOff   12
(3m36s ago)   40m

```

```

gus@coxa-pc:~/traefik-exercise$ k delete pod foobar-api-6446857f59-mpk2z
pod "foobar-api-6446857f59-mpk2z" deleted from default namespace

```

```

gus@coxa-pc:~/traefik-exercise$ k apply -f 01-foobar.yaml
persistentvolumeclaim/foobar-api-certs-pvc created
deployment.apps/foobar-api configured
service/foobar-api-svc configured

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                               READY   STATUS      REST
ARTS     AGE
foobar-api-598687ff9b-4nrzf     0/1     Init:0/1    0
4s
foobar-api-6446857f59-9vjqs     0/1     CrashLoopBackOff 1 (7
s ago)   11s

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                               READY   STATUS      RESTARTS   AG
E
foobar-api-598687ff9b-4nrzf     1/1     Running    0          14
s

gus@coxa-pc:~/traefik-exercise$ k logs foobar-api-598687ff9b-
4nrzf
Defaulted container "foobar-api" out of: foobar-api, cert-gen
(init)
Starting up on port 443

gus@coxa-pc:~/traefik-exercise$ k exec -it foobar-api-598687f
f9b-4nrzf -- ls -l /cert
Defaulted container "foobar-api" out of: foobar-api, cert-gen
(init)
total 8
-rw-r--r--    1 root      root           1826 Jan 17 21:12 cer
t.pem
-rw-----    1 root      root           3272 Jan 17 21:12 key.
pem

```

Simulating the script logic:

```
gus@coxa-pc:~/traefik-exercise$ k delete pod -l app=foobar-api
pod "foobar-api-598687ff9b-4nrzf" deleted from default namespace

gus@coxa-pc:~/traefik-exercise$ k apply -f 01-foobar.yaml
persistentvolumeclaim/foobar-api-certs-pvc unchanged
deployment.apps/foobar-api unchanged
service/foobar-api-svc unchanged

gus@coxa-pc:~/traefik-exercise$ k logs deploy/foobar-api -c cert-gen
(1/1) Installing openssl (3.5.4-r0)
Executing busybox-1.37.0-r30.trigger
OK: 9023 KiB in 17 packages
Cert already exists
```

Now, I had a working MVP. Confirmed it by running a port-forward and hitting the API endpoints (`/data`, `/echo`, `/bench`) via `curl`.

```
gus@coxa-pc:~/traefik-exercise$ k port-forward deploy/foobar-api 8443:443
Forwarding from 127.0.0.1:8443 -> 443
Forwarding from [::1]:8443 -> 443
Handling connection for 8443
Handling connection for 8443
Handling connection for 8443

gus@coxa-pc:~$ curl -k https://localhost:8443/data
| 

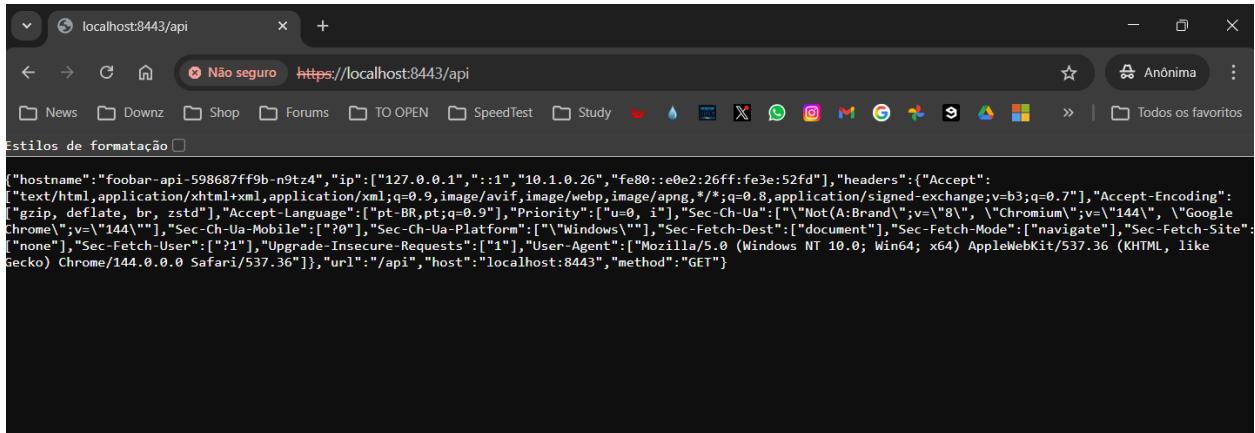
gus@coxa-pc:~$ curl -k https://localhost:8443/echo
Bad Request

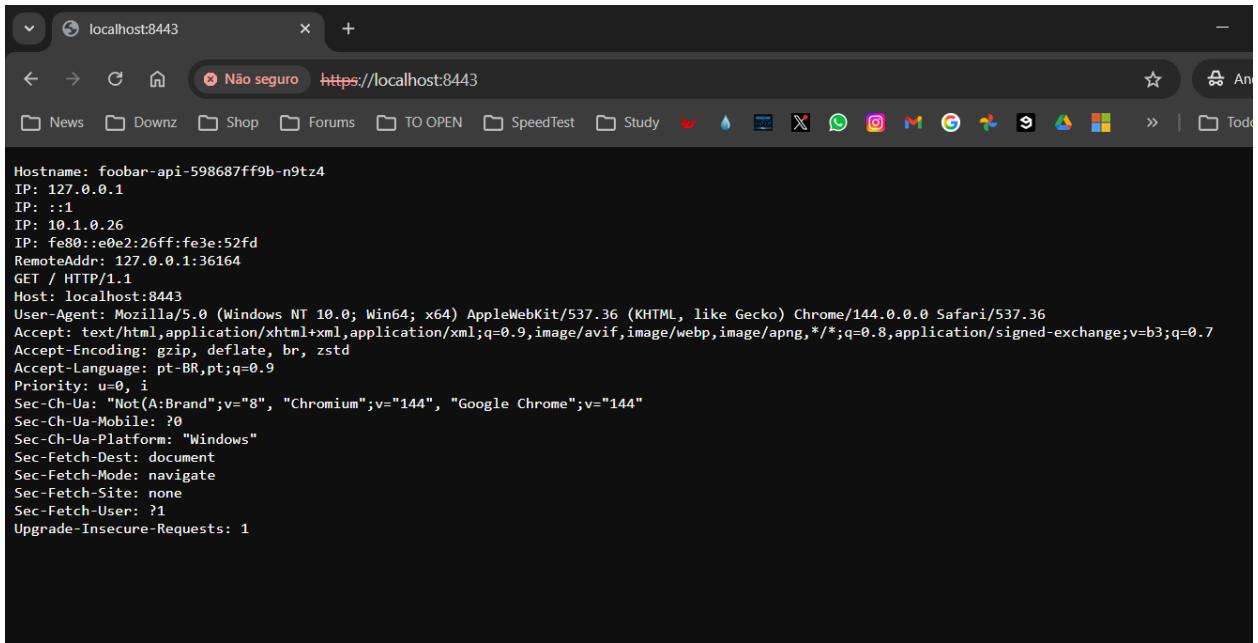
gus@coxa-pc:~$ curl -k https://localhost:8443/bench
```

1

```
gus@coxa-pc:~$ curl -k https://localhost:8443/
Hostname: foobar-api-598687ff9b-n9tz4
IP: 127.0.0.1
IP: ::1
IP: 10.x.x.xx
IP: fe80::e0e2:26ff:fe3e:52fd
RemoteAddr: 127.0.0.1:45206
GET / HTTP/1.1
Host: localhost:8443
User-Agent: curl/8.5.0
Accept: */*
```

```
gus@coxa-pc:~$ curl -k https://localhost:8443/api
{"hostname": "foobar-api-598687ff9b-n9tz4", "ip": ["127.0.0.1", "::1", "10.x.x.xx", "fe80::e0e2:26ff:fe3e:52fd"], "headers": {"Accept": ["*/*"], "User-Agent": ["curl/8.5.0"]}, "url": "/api", "host": "localhost:8443", "method": "GET"}
```





Hostname: foobar-api-598687ff9b-n9tz4  
IP: 127.0.0.1  
IP: ::1  
IP: 10.1.0.26  
IP: fe80::e0e2:26ff:fe3e:52fd  
RemoteAddr: 127.0.0.1:36164  
GET / HTTP/1.1  
Host: localhost:8443  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Accept-Encoding: gzip, deflate, br, zstd  
Accept-Language: pt-BR,pt;q=0.9  
Priority: u=0, i  
Sec-Ch-Ua: "Not(A:Brand";v="8", "Chromium";v="144", "Google Chrome";v="144"  
Sec-Ch-Ua-Mobile: ?0  
Sec-Ch-Ua-Platform: "Windows"  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: none  
Sec-Fetch-User: ?1  
Upgrade-Insecure-Requests: 1

## Integrating Traefik

With the application stable, it was finally time to integrate Traefik as part of the infra as the reverse proxy.

So I created the `02-traefik-values.yaml` file to define the custom configs (like enabling persistence and the Traefik dashboard), so that Helm could merge them with the default chart settings. Then, I installed Traefik via Helm:

```
gus@coxa-pc:~/traefik-exercise$ cat 02-traefik-values.yaml
persistence:
  enabled: true
  name: data
  accessMode: ReadWriteOnce
  size: 10Mi
  path: /data

ingressRoute:
  dashboard:
    enabled: true

additionalArguments:
```

```
- "--certificatesresolvers.myresolver.acme.storage=/data/acme.json"
- "--certificatesresolvers.myresolver.acme.email=gustavosaviano@gmail.com"
- "--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web"
```

```
gus@coxa-pc:~/traefik-exercise$ helm repo add traefik https://traefik.github.io/charts
"traefik" already exists with the same configuration, skipping
```

```
gus@coxa-pc:~/traefik-exercise$ helm install traefik traefik/traefik -f 02-traefik-values.yaml
NAME: traefik
LAST DEPLOYED: Sat Jan 17 18:59:56 2026
NAMESPACE: default
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
NOTES:
  traefik with docker.io/traefik:v3.6.6 has been deployed successfully on default namespace!
```

```
gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                           READY   STATUS    RESTARTS   AGE
foobar-api-598687ff9b-n9tz4   1/1     Running   0          43m
traefik-5f5775fb87-rwlqs      1/1     Running   0          34s
```

## The Third Challenge

Since I had to bind a self-signed certificate to start the Go app, I faced a new issue: Traefik did not trust this certificate with the default settings.

When Traefik attempted to communicate with the backend service, it failed with a `500 Internal Server Error`. The logs confirmed the issue:

```
gus@coxa-pc:~$ curl -k https://foobar-api.localhost
Internal Server Error

gus@coxa-pc:~/traefik-exercise$ k logs deploy/traefik | grep
x509
2026-01-17T22:15:56Z ERR 500 Internal Server Error error="tl
s: failed to verify certificate: x509: cannot validate certif
icate for 10.x.x.xx because it doesn't contain any IP SANs"
2026-01-17T22:16:06Z ERR 500 Internal Server Error error="tl
s: failed to verify certificate: x509: cannot validate certif
icate for 10.x.x.xx because it doesn't contain any IP SANs"
2026-01-17T22:18:15Z ERR 500 Internal Server Error error="tl
s: failed to verify certificate: x509: cannot validate certif
icate for 10.x.x.xx because it doesn't contain any IP SANs"
```

To resolve this, I checked Traefik docs and found the `insecureSkipVerify` option in the `ServersTransport` configuration. So I created a custom `ServersTransport` resource named `trust-self` and applied it to the IngressRoute. This told Traefik to skip certificate validation for this specific backend service, allowing the connection to proceed:

```
gus@coxa-pc:~/traefik-exercise$ k get crd | grep serverstrans
ports
servertransports.traefik.io                               2026-01-17T12:
52:30Z

gus@coxa-pc:~/traefik-exercise$ cat 03-ingress.yaml
apiVersion: traefik.io/v1alpha1
kind: ServersTransport
metadata:
```

```

    name: trust-self
spec:
  insecureSkipVerify: true
  ...
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: foobar-api-ingress
spec:
  entryPoints:
    - websecure
  routes:
    - match: Host(`foobar-api.localhost`)
      kind: Rule
      services:
        - name: foobar-api-svc
          port: 443
          scheme: https
          serversTransport: trust-self
  tls:
    certResolver: myresolver

```

With this fix in place, the architecture was complete. I verified the setup by curling via Traefik's entrypoint:

```

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                               READY   STATUS    RESTARTS   AGE
E
foobar-api-598687ff9b-n9tz4     1/1     Running   0          64m
m
traefik-5f5775fb87-rwlqs       1/1     Running   0          22m

gus@coxa-pc:~/traefik-exercise$ k get services
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE

```

```
foobar-api-svc   ClusterIP      10.106.142.135   <none>
443/TCP          110m
kubernetes       ClusterIP      10.96.0.1        <none>
443/TCP          10h
traefik          LoadBalancer   10.108.110.115   localhost
80:31868/TCP,443:31609/TCP   22m
```

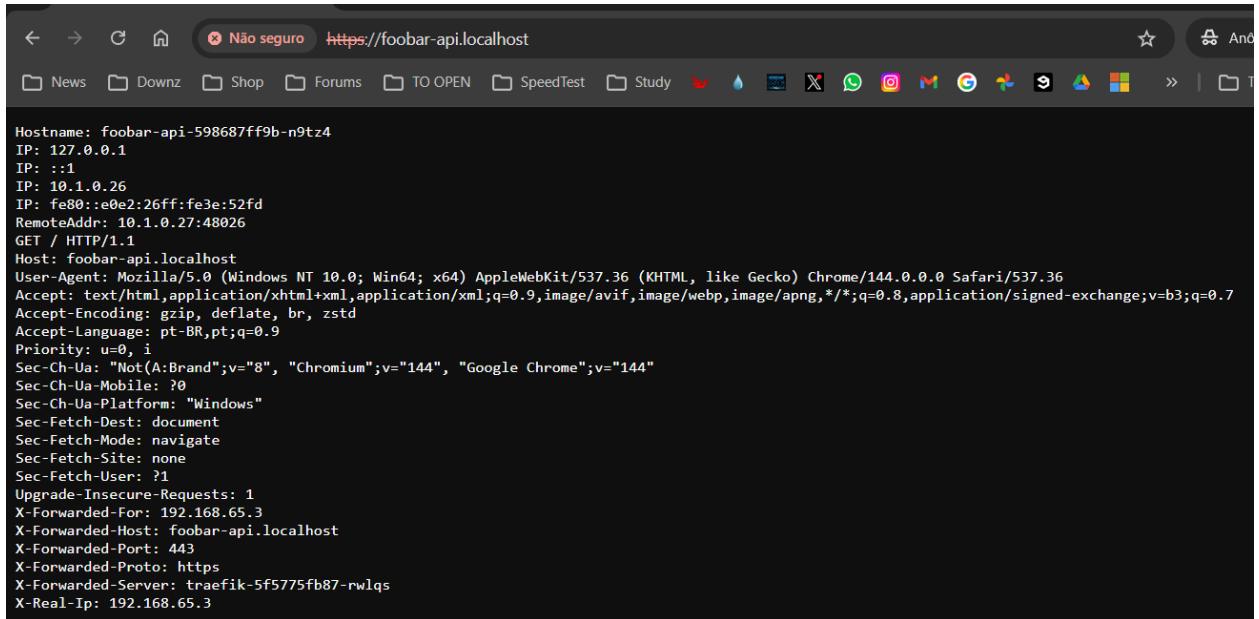
```
gus@coxa-pc:~/traefik-exercise$ k get serverstransport
NAME      AGE
trust-self 77s
```

```
gus@coxa-pc:~$ curl -v -k https://foobar-api.localhost
* Host foobar-api.localhost was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:443...
* Connected to foobar-api.localhost (::1) port 443
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / X25
519 / RSASSA-PSS
* ALPN: server accepted h2
* Server certificate:
* subject: CN=TRAEFIK DEFAULT CERT
* start date: Jan 17 22:22:03 2026 GMT
* expire date: Jan 17 22:22:03 2027 GMT
* issuer: CN=TRAEFIK DEFAULT CERT
* SSL certificate verify result: self-signed certificate (1
8), continuing anyway.
```

```
* Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* using HTTP/2
* [HTTP/2] [1] OPENED stream for https://foobar-api.localhost/
* [HTTP/2] [1] [:method: GET]
* [HTTP/2] [1] [:scheme: https]
* [HTTP/2] [1] [:authority: foobar-api.localhost]
* [HTTP/2] [1] [:path: /]
* [HTTP/2] [1] [user-agent: curl/8.5.0]
* [HTTP/2] [1] [accept: */*]
> GET / HTTP/2
> Host: foobar-api.localhost
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/2 200
< content-type: text/plain; charset=utf-8
< date: Sat, 17 Jan 2026 22:30:33 GMT
< content-length: 434
<
Hostname: foobar-api-598687ff9b-n9tz4
IP: 127.0.0.1
IP: ::1
IP: 10.x.x.xx
IP: fe80::e0e2:26ff:fe3e:52fd
RemoteAddr: 10.1.0.27:53634
GET / HTTP/1.1
Host: foobar-api.localhost
User-Agent: curl/8.5.0
Accept: */*
Accept-Encoding: gzip
X-Forwarded-For: 127.0.0.1
X-Forwarded-Host: foobar-api.localhost
X-Forwarded-Port: 443
```

```
X-Forwarded-Proto: https
X-Forwarded-Server: traefik-5f5775fb87-rwlqs
X-Real-Ip: 127.0.0.1

* Connection #0 to host foobar-api.localhost left intact
```



## Improvements

Now that the application is running fine, I decided to focus on hardening the infra, as requested.

To start with, I updated the deployment configuration to include Readiness and Liveness probes, as per what was requested in the email:

- **Readiness:** configured to hit the root path ( / ) on port 443.
- **Liveness:** configured to hit the `/health` endpoint as the application logic allows us to manipulate its status, so I could test failure scenarios.

```
gus@coxa-pc:~/traefik-exercise$ grep -A 14 liveness 01-fooba
r.yaml
livenessProbe:
  httpGet:
```

```

    path: /health
    port: 443
    scheme: HTTPS
    initialDelaySeconds: 5
    periodSeconds: 10
  readinessProbe:
    httpGet:
      path: /
      port: 443
      scheme: HTTPS
    initialDelaySeconds: 2
    periodSeconds: 5
  ...

```

Let's test it out:

```

gus@coxa-pc:~/traefik-exercise$ k apply -f 01-foobar.yaml
persistentvolumeclaim/foobar-api-certs-pvc unchanged
deployment.apps/foobar-api configured
service/foobar-api-svc unchanged

gus@coxa-pc:~/traefik-exercise$ k describe pod -l app=foobar-
api | grep -iE "liveness|readiness"
  Liveness:          http-get https://:443/health delay=5s tim
eout=1s period=10s #success=1 #failure=3
  Readiness:         http-get https://:443/ delay=2s timeout=1
s period=5s #success=1 #failure=3

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                  READY   STATUS    RESTARTS
AGE
foobar-api-7d755ddf5f-cjzl9   1/1     Running   0
105s
traefik-5f5775fb87-rwlqs     1/1     Running   1 (124m ago)
3h6m

```

To prove the configuration works, I performed a failure simulation. Since the application's `/health` accepts a POST request to manually set its status code, I "poisoned" the probe by sending a HTTP 500 to the endpoint:

```
gus@coxa-pc:~$ curl -k -X POST -d "500" https://foobar-api.lo  
calhost/health

gus@coxa-pc:~$ k describe pod -l app=foobar-api | grep -iE "l  
iveness|readiness"
    Liveness: http-get https://:443/health delay=5s tim  
eout=1s period=10s #success=1 #failure=3
    Readiness: http-get https://:443/ delay=2s timeout=1  
s period=5s #success=1 #failure=3
    Warning Unhealthy 15s (x3 over 35s) kubelet
  Liveness probe failed: HTTP probe failed with statuscode: 500
    Normal Killing 15s kubelet
Container foobar-api failed liveness probe, will be restarted

gus@coxa-pc:~$ k get pods
NAME          READY   STATUS    RESTARTS
AGE
foobar-api-7d755ddf5f-cjzl9  1/1     Running   1 (2m ago)
5m51s
traefik-5f5775fb87-rwlqs    1/1     Running   1 (128m ago)
3h10m
```

Back to normal:

```
gus@coxa-pc:~$ curl -k -X POST -d "200" https://foobar-api.lo  
calhost/health
```

## Time for Security

Given the requirement to secure access to the program asked in the email (and since I plan to expose the admin Traefik dashboard as well) I decided to

implement an auth layer using Traefik CRD Middleware. It is always good practice to add multiple security layers to the infra. Right now, we have TLS certificates at both the frontend and backend and now, we will add basic auth to control who can access these resources:

```
gus@coxa-pc:~$ k get crds | grep -i middleware
middlewares.traefik.io                               2026-01-17T12:
52:30Z
middlewarescps.traefik.io                           2026-01-17T12:
52:30Z
```

While Middleware has many options (as Rate Limiting or IP Whitelisting), I decided to keep the implementation simple and straightforward by only adding basicAuth, as follows:

```
gus@coxa-pc:~/traefik-exercise$ k create secret generic foobar-api-auth-secret --from-literal=users='test:$apr1$0FCEKvM.$D3qeQB/DIxqr18jLfaAQd.'
secret/foobar-api-auth-secret created

gus@coxa-pc:~/traefik-exercise$ cat 04-middleware.yaml
apiVersion: traefik.io/v1alpha1
kind: Middleware
metadata:
  name: foobar-api-auth
spec:
  basicAuth:
    secret: foobar-api-auth-secret

gus@coxa-pc:~/traefik-exercise$ cat 03-ingress.yaml | grep -A 1 middleware
  middlewares:
    - name: foobar-api-auth

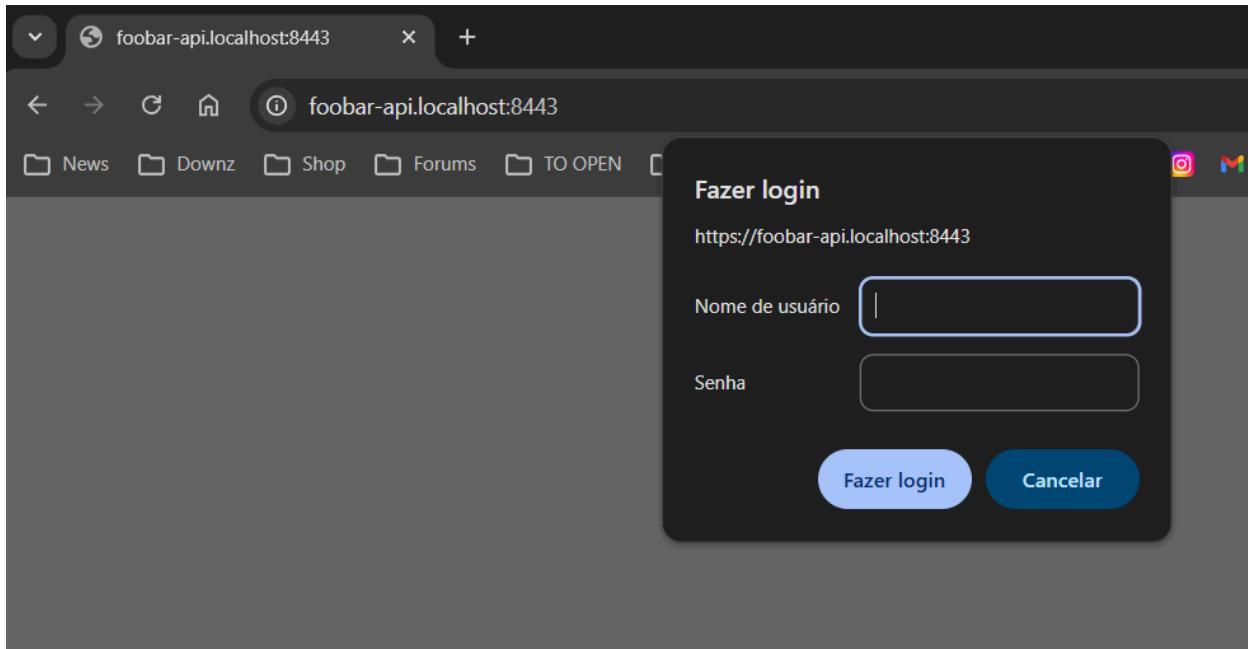
gus@coxa-pc:~/traefik-exercise$ k apply -f 04-middleware.yaml
middleware.traefik.io/foobar-api-auth unchanged
```

```
gus@coxa-pc:~/traefik-exercise$ k apply -f 03-ingress.yaml
servertransport.traefik.io/trust-self unchanged
ingressroute.traefik.io/foobar-api-ingress unchanged
```

Testing:

```
gus@coxa-pc:~$ curl -k https://foobar-api.localhost
401 Unauthorized

gus@coxa-pc:~$ curl -k -u test:test https://foobar-api.localhost
Hostname: foobar-api-7d755ddf5f-cjzl9
IP: 127.0.0.1
IP: ::1
IP: 10.x.x.xx
IP: fe80::5c7f:63ff:fe11:ad26
RemoteAddr: 10.x.x.xx:57372
GET / HTTP/1.1
Host: foobar-api.localhost
User-Agent: curl/8.5.0
Accept: */*
Accept-Encoding: gzip
Authorization: Basic dGVzdDp0ZXN0
X-Forwarded-For: 127.0.0.1
X-Forwarded-Host: foobar-api.localhost
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Forwarded-Server: traefik-5f5775fb87-6dfkh
X-Real-Ip: 127.0.0.1
```



```
foobar-api.localhost:8443
Nome de usuário
Senha
Fazer login Cancelar

Hostname: foobar-api-7d755ddf5f-cjz19
IP: 127.0.0.1
IP: ::1
IP: 10.1.0.41
IP: fe80::5c7f:63ff:fe11:ad26
RemoteAddr: 10.1.0.38:50732
GET / HTTP/1.1
Host: foobar-api.localhost:8443
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: pt-BR,pt;q=0.9
Authorization: Basic dGVzdDp0ZXN0
Cache-Control: max-age=0
Priority: u=0, i
Sec-Ch-Ua: "Not(A:Brand";v="8", "Chromium";v="144", "Google Chrome";v="144"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
X-Forwarded-For: 127.0.0.1
X-Forwarded-Host: foobar-api.localhost:8443
X-Forwarded-Port: 8443
X-Forwarded-Proto: https
X-Forwarded-Server: traefik-5f5775fb87-6dfkh
X-Real-Ip: 127.0.0.1
```

## Auditing

To meet the requirement for full visibility into the infrastructure, I moved on to adjusting Traefik's logging features and implementing tracing. This allows us to trace requests "back-to-back" and understand exactly what is happening behind the scenes.

First, I adjusted the logging config. I changed the general log level from the default `ERROR` to `DEBUG` to provide more details about Traefik's internals.

I also enabled Access Logging. Unlike system logs, access logs track the actual traffic activity, showing us details for every request such as client IP, response status codes, and processing duration:

```
gus@coxa-pc:~/traefik-exercise$ cat 02-traefik-values.yaml |  
grep logs -A10  
logs:  
  general:  
    level: DEBUG  
  access:  
    enabled: true  
    format: json  
    fields:  
      general:  
        defaultmode: keep  
      headers:  
        defaultmode: keep
```

Testing:

```
gus@coxa-pc:~/traefik-exercise$ helm upgrade traefik traefik/  
traefik -f 02-traefik-values.yaml  
Release "traefik" has been upgraded. Happy Helming!  
NAME: traefik  
LAST DEPLOYED: Sun Jan 18 12:16:07 2026  
NAMESPACE: default  
STATUS: deployed  
REVISION: 2  
DESCRIPTION: Upgrade complete  
TEST SUITE: None  
NOTES:  
  traefik with docker.io/traefik:v3.6.6 has been deployed succe  
ssfully on default namespace!
```

```
gus@coxa-pc:~/traefik-exercise$ k logs deploy/traefik | tail
2026-01-18T15:16:10Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/crd/kubernetes_http.go:777 > No secret name provided providerName=kubernetescrd
2026-01-18T15:16:10Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/ingress/kubernetes.go:186 > Skipping Kubernetes event kind *v1.EndpointSlice providerName=kubernetes
2026-01-18T15:16:10Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/crd/kubernetes.go:179 > Skipping Kubernetes event kind *v1.EndpointSlice providerName=kubernetescrd
2026-01-18T15:16:10Z ERR github.com/traefik/traefik/v3/pkg/provider/acme/provider.go:501 > Unable to obtain ACME certificate for domains error="unable to generate a certificate for the domains [foobar-api.localhost]: acme: error: 400 :: POST :: https://acme-v02.api.letsencrypt.org/acme/new-order :: urn:ietf:params:acme:error:rejectedIdentifier :: Invalid identifiers requested :: Cannot issue for \"foobar-api.localhost\": Domain name does not end with a valid public suffix (TLD)" ACME CA=https://acme-v02.api.letsencrypt.org/directory acmeCA=http://acme-v02.api.letsencrypt.org/directory domains=["foobar-api.localhost"] providerName=myresolver.acme routerName=default-foobar-api-ingress-a23eb72bc207fc8fa5d6@kubernetescrd rule=Host(`foobar-api.localhost`)
2026-01-18T15:16:10Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/ingress/kubernetes.go:186 > Skipping Kubernetes event kind *v1.EndpointSlice providerName=kubernetes
2026-01-18T15:16:10Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/crd/kubernetes_http.go:777 > No secret name provided providerName=kubernetescrd
2026-01-18T15:16:10Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/crd/kubernetes.go:179 > Skipping Kubernetes event kind *v1.EndpointSlice providerName=kubernetescrd
2026-01-18T15:16:27Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/ingress/kubernetes.go:186 > Skipping Kubernetes event kind *v1.Node providerName=kubernetes
```

```
2026-01-18T15:16:27Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/crd/kubernetes_http.go:777 > No secret name provided providerName=kubernetescrd
2026-01-18T15:16:27Z DBG github.com/traefik/traefik/v3/pkg/provider/kubernetes/crd/kubernetes.go:179 > Skipping Kubernetes event kind *v1.Node providerName=kubernetescrd
```

```
gus@coxa-pc:~$ k logs deploy/traefik | tail -2
{"ClientAddr":"127.0.0.1:39090","ClientHost":"127.0.0.1","ClientPort":"39090","ClientUsername":"test","DownstreamContentSize":469,"DownstreamStatus":200,"Duration":8146727,"OriginContentSize":469,"OriginDuration":6942660,"OriginStatus":200,"Overhead":1204067,"RequestAddr":"foobar-api.localhost:443","RequestContentSize":0,"RequestCount":1,"RequestHost":"foobar-api.localhost","RequestMethod":"GET","RequestPath":"/","RequestPort":443,"RequestProtocol":"HTTP/2.0","RequestScheme":"https","RetryAttempts":0,"RouterName":"default-foobar-api-ingress-a23eb72bc207fc8fa5d6@kubernetescrd","ServiceAddr":"10.1.0.41:443","ServiceName":"default-foobar-api-ingress-a23eb72bc207fc8fa5d6@kubernetescrd","ServiceURL":"https://10.x.x.xx:443","StartLocal":"2026-01-18T15:17:35.087656757Z","StartUTC":"2026-01-18T15:17:35.087656757Z","TLSCipher":"TLS_AES_128_GCM_SHA256","TLSVersion":"1.3","downstream_Content-Length":469,"downstream_Content-Type":"text/plain; charset=utf-8","downstream_Date":"Sun, 18 Jan 2026 15:17:35 GMT","entryPointName":"websecure","level":"info","msg":"","origin_ContentLength":469,"origin_Content-Type":"text/plain; charset=utf-8","origin_Date":"Sun, 18 Jan 2026 15:17:35 GMT","request_Accept":"/","request_Authorization":"Basic dGVzdDp0ZXN0","request_User-Agent":"curl/8.5.0","request_X-Forwarded-Host": "foobar-api.localhost", "request_X-Forwarded-Port": "443", "request_X-Forwarded-Proto": "https", "request_X-Forwarded-Server": "traefik-74b7c689bc-zwrgt", "request_X-Real-Ip": "127.0.0.1", "time": "2026-01-18T15:17:35Z"}
{"ClientAddr":"127.0.0.1:58146","ClientHost":"127.0.0.1","ClientPort":58146,"ClientUsername":"","DownstreamContentSize":
```

```
17,"DownstreamStatus":401,"Duration":275951,"GzipRatio":0,"OriginContentSize":0,"OriginDuration":0,"OriginStatus":0,"Overhead":275951,"RequestAddr":"foobar-api.localhost:443","RequestContentSize":0,"RequestCount":2,"RequestHost":"foobar-api.localhost","RequestMethod":"GET","RequestPath":"/","RequestPort":"443","RequestProtocol":"HTTP/2.0","RequestScheme":"http","RetryAttempts":0,"RouterName":"default-foobar-api-ingress-a23eb72bc207fc8fa5d6@kubernetescrd","StartLocal":"2026-01-18T15:17:48.943890857Z","StartUTC":"2026-01-18T15:17:48.943890857Z","TLSCipher":"TLS_AES_128_GCM_SHA256","TLSVersion":"1.3","downstream_Content-Type":"text/plain","downstream_Www-Authenticate":"Basic realm=\\"traefik\\\"", "entryPointName": "websecure", "level": "info", "msg": "", "request_Accept": "*/*", "request_User-Agent": "curl/8.5.0", "request_X-Forwarded-Host": "foobar-api.localhost:443", "request_X-Forwarded-Port": "443", "request_X-Forwarded-Proto": "https", "request_X-Forwarded-Server": "traefik-74b7c689bc-zwrgt", "request_X-Real-Ip": "127.0.0.1", "time": "2026-01-18T15:17:48Z"}
```

With logs in place, I moved to the Tracing piece. I chose to use the OpenTelemetry framework, as described in Traefik official docs, for generating traces, with Jaeger as the backend to process and visualize them:

```
gus@coxa-pc:~/traefik-exercise$ cat 05-tracing.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jaeger
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jaeger
  template:
    metadata:
      labels:
```

```

        app: jaeger
spec:
  containers:
    - name: jaeger
      image: jaegertracing/all-in-one:latest
      env:
        - name: COLLECTOR_OTLP_ENABLED
          value: "true"
      ports:
        - containerPort: 16686
        - containerPort: 4317
    ...
apiVersion: v1
kind: Service
metadata:
  name: jaeger
spec:
  ports:
    - port: 16686
      name: ui
    - port: 4317
      name: otel-grpc
  selector:
    app: jaeger

```

```

gus@coxa-pc:~/traefik-exercise$ k apply -f 05-tracing.yaml
deployment.apps/jaeger created
service/jaeger created

```

```

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME                  READY   STATUS    RESTARTS   AGE
foobar-api-7d755ddf5f-cjzl9   1/1     Running   2 (41m ago)
14h
jaeger-6c996c4fd4-k9tvk      1/1     Running   0

```

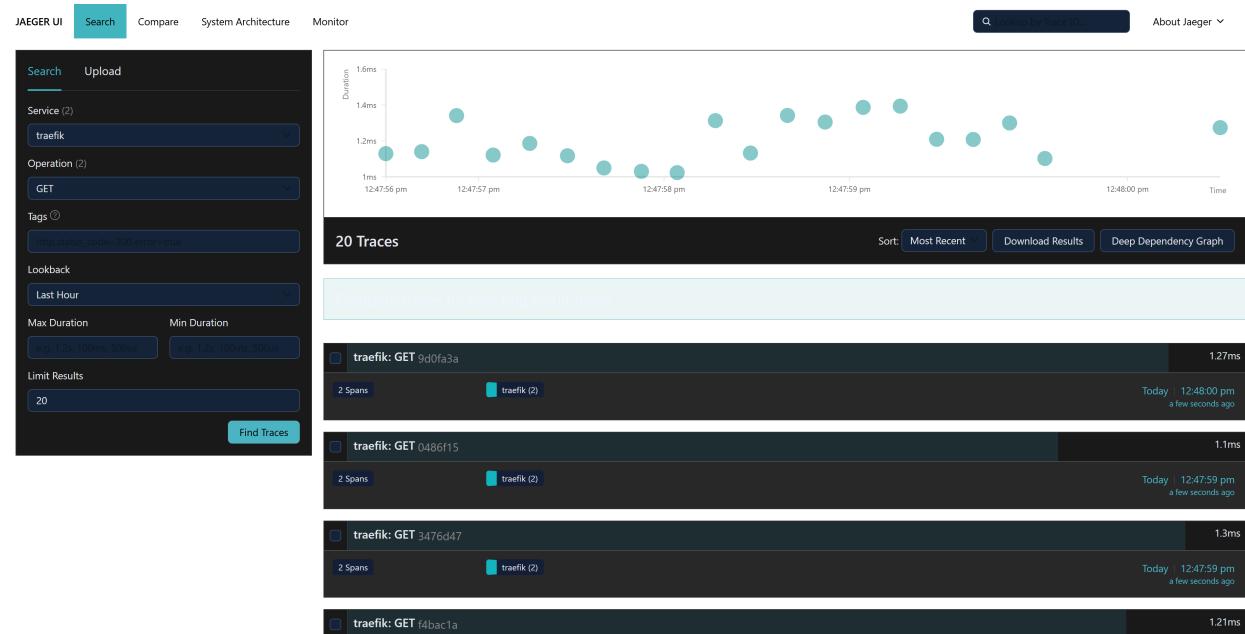
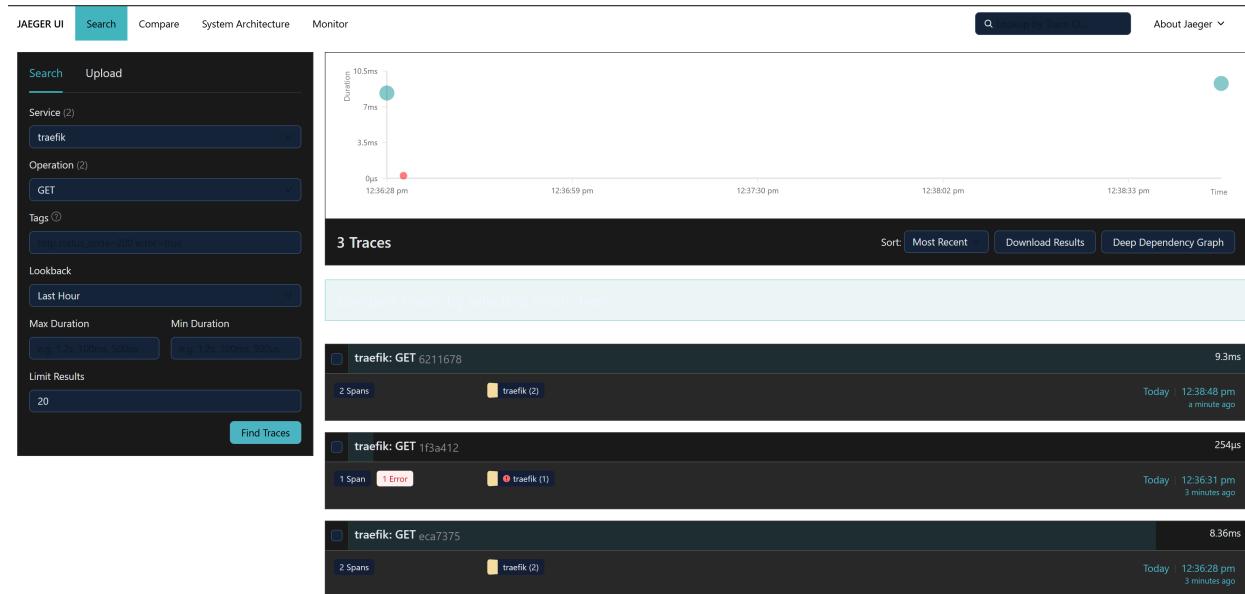
```
8s
traefik-74b7c689bc-zwrgt      1/1      Running   0
5m41s
```

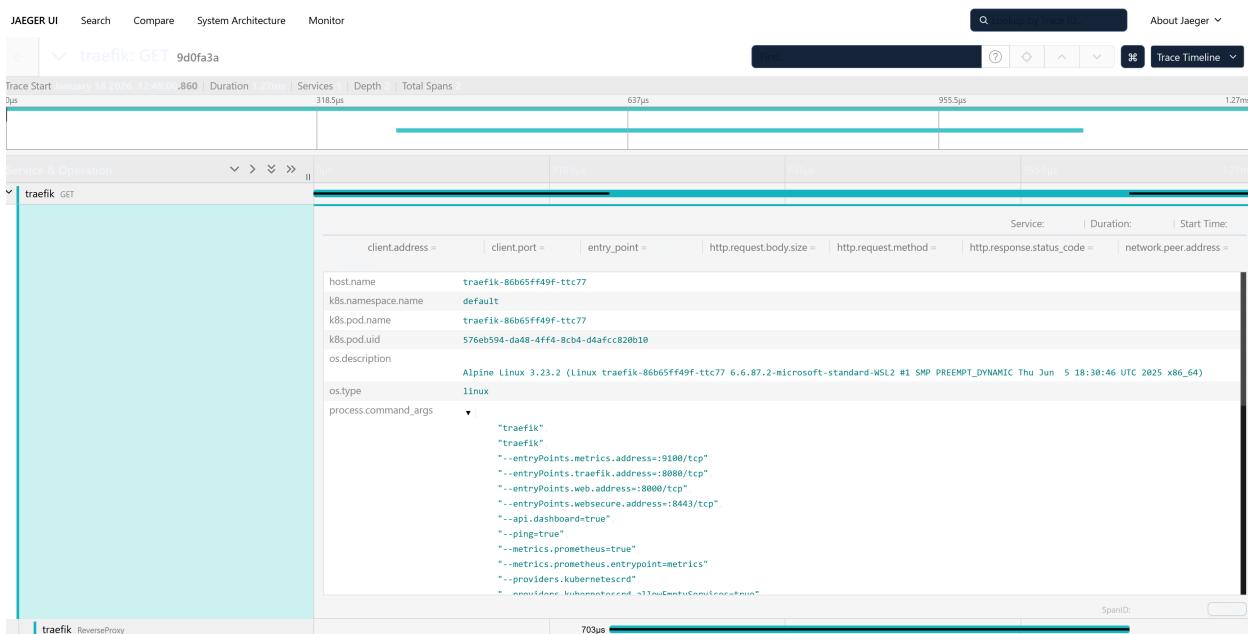
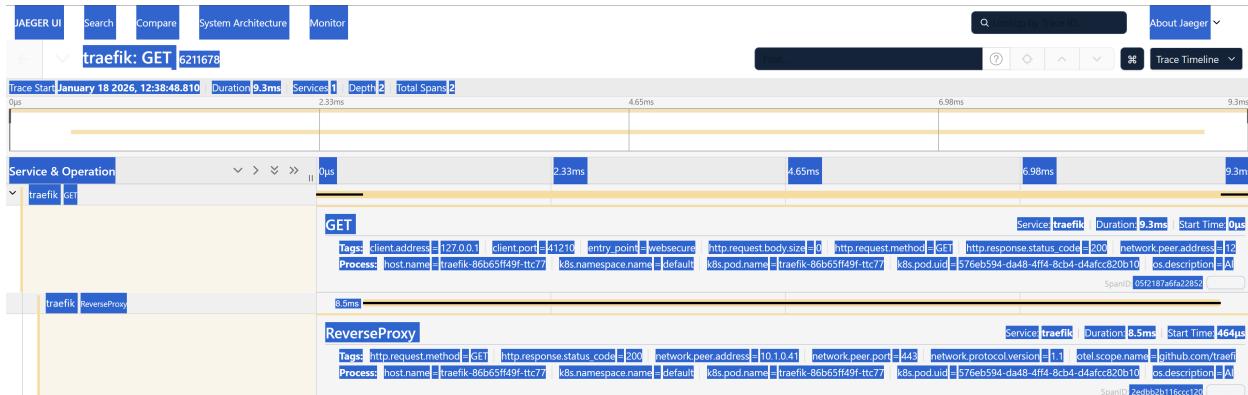
```
gus@coxa-pc:~/traefik-exercise$ grep tracing 02-traefik-values.yaml
  - "--tracing.otlp.grpc.endpoint=jaeger.default.svc.cluster.local:4317"
  - "--tracing.otlp.grpc.insecure=true"

gus@coxa-pc:~/traefik-exercise$ helm upgrade traefik traefik/
traefik -f 02-traefik-values.yaml
Release "traefik" has been upgraded. Happy Helming!
NAME: traefik
LAST DEPLOYED: Sun Jan 18 12:35:40 2026
NAMESPACE: default
STATUS: deployed
REVISION: 4
DESCRIPTION: Upgrade complete
TEST SUITE: None
NOTES:
  traefik with docker.io/traefik:v3.6.6 has been deployed successfully on default namespace!
```

Checking the Jaeger UI:

```
gus@coxa-pc:~/traefik-exercise$ k port-forward svc/jaeger 16686:16686
Forwarding from 127.0.0.1:16686 -> 16686
Forwarding from [::1]:16686 -> 16686
```





## Observability and Metrics

Finally, to satisfy the metrics requirements requested in the email and complete the infra enhancements, I implemented monitoring and observability. This includes the Traefik Dashboard, Prometheus for metrics collection, and Grafana for data visualization.

So first, I implemented the Traefik Dashboard. While this can be enabled automatically via Helm, I chose to define it using an `IngressRoute`. This approach gives me control over the routing and allows me to secure the dashboard using the same Basic Auth Middleware we created earlier:

```

gus@coxa-pc:~/traefik-exercise$ grep -C10 traefik.localhost 0
3-ingress.yaml
---
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: traefik-dashboard
spec:
  entryPoints:
    - websecure
  routes:
    - match: Host(`traefik.localhost`)
      kind: Rule
      middlewares:
        - name: foobar-api-auth
      services:
        - name: api@internal
          kind: TraefikService

gus@coxa-pc:~/traefik-exercise$ k apply -f 03-ingress.yaml
servertransport.traefik.io/trust-self unchanged
ingressroute.traefik.io/foobar-api-ingress unchanged
ingressroute.traefik.io/traefik-dashboard created

gus@coxa-pc:~/traefik-exercise$ k get ingressroutes
NAME           AGE
foobar-api-ingress   17h
traefik-dashboard   18s

```

And voila:

The Traefik dashboard provides a high-level overview of the system's health and configuration. Key sections include:

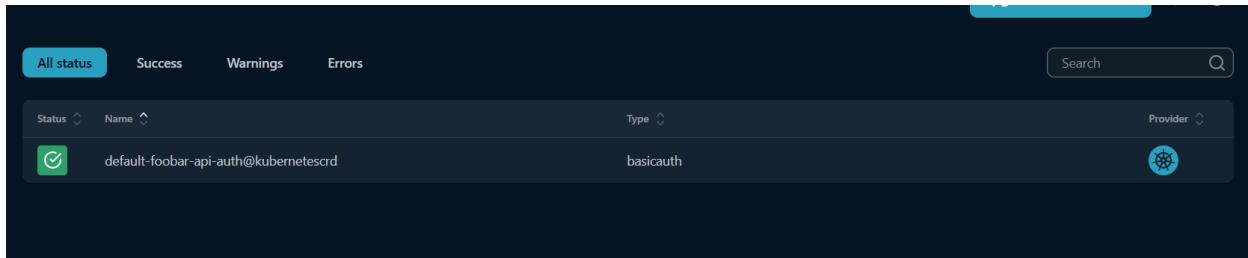
- Entrypoints:** Shows metrics for various ports: :9100/tcp (Metrics), :8080/tcp (Traefik), :8000/tcp (Web), and :8443/tcp (Websecure).
- HTTP:** Displays three circular dashboards for Routers, Services, and Middlewares, each showing success, warnings, and errors counts.
- TCP:** Shows no related objects.
- UDP:** Shows no related objects.
- Features:** Shows tracing (OTLP), metrics (Prometheus), and accesslog (ON).
- Providers:** Lists KubeIngress and KubeInferior.

The "Success" tab of the entrypoint list shows the following configurations:

Status	TLS	Rule	Entryports	Name	Service	Provider	Priority
Success		PathPrefix('/well-known/acme-c...')	web	acme- <a href="#">http@internal</a>	acme- <a href="#">http@internal</a>	<a href="#">æ</a>	92237...
Success	W	Host('foobar-api.localhost')	websecure	default-foobar-api-ingress-a23e...	default-foobar-api-ingress-a23e...	<a href="#">æ</a>	28
Success	W	Host('traefik.localhost')	websecure	default-traefik-dashboard-8ad1b...	api@internal	<a href="#">æ</a>	25
Success		PathPrefix('/ping')	traefik	ping@internal	ping@internal	<a href="#">æ</a>	92237...
Success		PathPrefix('/metrics')	metrics	prometheus@internal	prometheus@internal	<a href="#">æ</a>	92237...

The "Success" tab of the router list shows the following configurations:

Status	Name	Type	Servers	Provider
Success	acme- <a href="#">http@internal</a>	-	0	<a href="#">æ</a>
Success	api@internal	-	0	<a href="#">æ</a>
Success	dashboard@internal	-	0	<a href="#">æ</a>
Success	default-foobar-api-ingress-a23eb72bc207fc8fa5d6@kubernetescrd	loadbalancer	1	<a href="#">æ</a>
Success	noop@internal	-	0	<a href="#">æ</a>
Success	ping@internal	-	0	<a href="#">æ</a>
Success	prometheus@internal	-	0	<a href="#">æ</a>



With the dashboard active, I moved on to the metrics. I deployed Prometheus to scrape data and Grafana to visualize it:

```
gus@coxa-pc:~/traefik-exercise$ cat 06-prometheus.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  labels:
    app: prometheus
data:
  prometheus.yml: |
    global:
      scrape_interval: 10s
    scrape_configs:
      - job_name: 'traefik'
        static_configs:
          - targets: ['traefik.default.svc.cluster.local:9100']
    ---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
```

```

template:
  metadata:
    labels:
      app: prometheus
  spec:
    containers:
      - name: prometheus
        image: prom/prometheus
        args:
          - "--config.file=/etc/prometheus/prometheus.yml"
    ports:
      - containerPort: 9090
    volumeMounts:
      - name: config-volume
        mountPath: /etc/prometheus/
  volumes:
    - name: config-volume
      configMap:
        name: prometheus-config
  ---
apiVersion: v1
kind: Service
metadata:
  name: prometheus
spec:
  selector:
    app: prometheus
  ports:
    - port: 9090
      targetPort: 9090

```

```

gus@coxa-pc:~/traefik-exercise$ cat 07-grafana.yaml
apiVersion: v1
kind: ConfigMap
metadata:

```

```
name: grafana-datasources
data:
  prometheus.yaml: |
    apiVersion: 1
    datasources:
      - name: Prometheus
        type: prometheus
        url: http://prometheus:9090
        access: proxy
        isDefault: true
    ---
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: grafana
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: grafana
      template:
        metadata:
          labels:
            app: grafana
      spec:
        containers:
          - name: grafana
            image: grafana/grafana
            ports:
              - containerPort: 3000
            volumeMounts:
              - name: grafana-datasources
                mountPath: /etc/grafana/provisioning/datasources/
        volumes:
          - name: grafana-datasources
            configMap:
```

```

        name: grafana-datasources
        ...
apiVersion: v1
kind: Service
metadata:
  name: grafana
spec:
  selector:
    app: grafana
  ports:
    - port: 3000
      targetPort: 3000

```

Deploying them:

```

gus@coxa-pc:~/traefik-exercise$ k apply -f 06-prometheus.yaml
configmap/prometheus-config created
deployment.apps/prometheus created
service/prometheus created

```

```

gus@coxa-pc:~/traefik-exercise$ k apply -f 07-grafana.yaml
configmap/grafana-datasources created
deployment.apps/grafana created
service/grafana created

```

NAME	READY	STATUS	RESTARTS
AGE			
foobar-api-7d755ddf5f-cjzl9	1/1	Running	2 (95m ago)
15h			
grafana-79697b5f8-jzzk5	1/1	Running	0
50s			
jaeger-6c996c4fd4-k9tvk	1/1	Running	0
54m			
prometheus-54488fdf45-8vr4v	1/1	Running	0
53s			

traefik-86b65ff49f-ttc77	1/1	Running	0
40m			

For Prometheus to actually see anything, I had to update the Traefik Helm configuration:

```
gus@coxa-pc:~/traefik-exercise$ helm upgrade traefik traefik/
traefik -f 02-traefik-values.yaml
Release "traefik" has been upgraded. Happy Helming!
NAME: traefik
LAST DEPLOYED: Sun Jan 18 13:21:34 2026
NAMESPACE: default
STATUS: deployed
REVISION: 8
DESCRIPTION: Upgrade complete
TEST SUITE: None
NOTES:
  traefik with docker.io/traefik:v3.6.6 has been deployed successfully on default namespace!
```

With both deployed and traffic flowing, I accessed Prometheus UI and confirmed that the Traefik target was up and metrics were being actively scraped. Then I

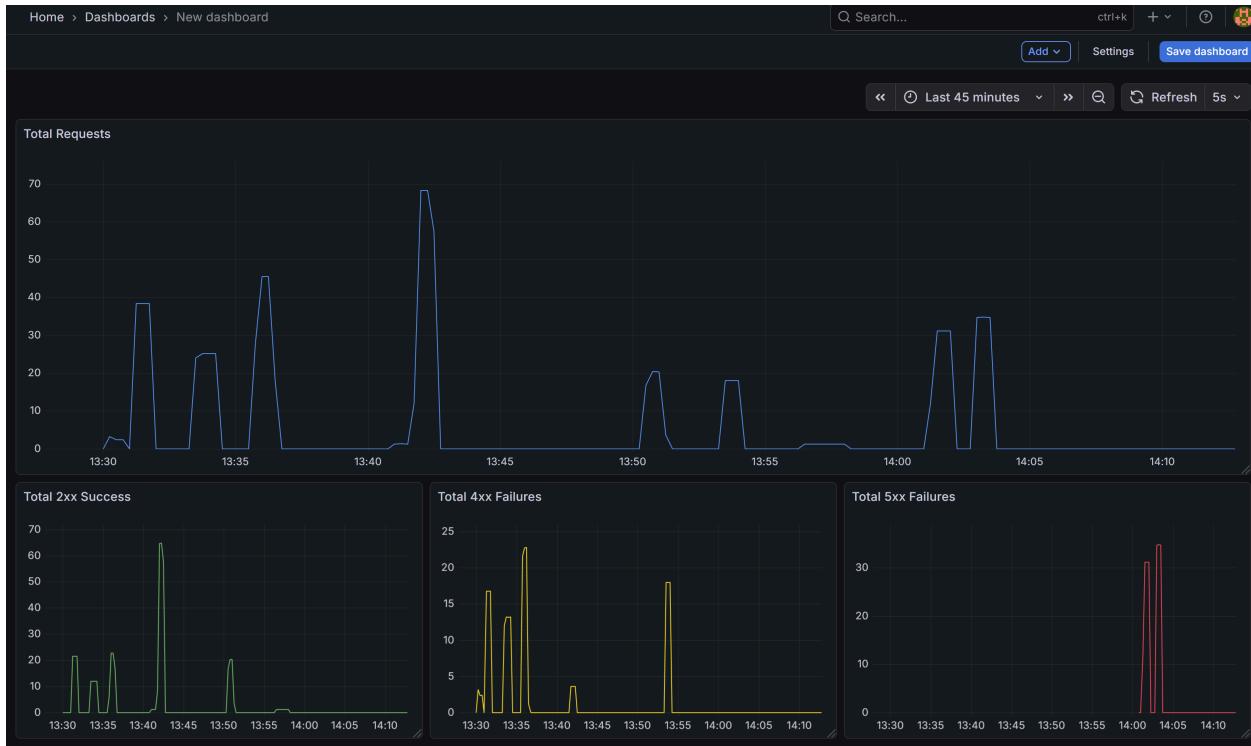
built some dashboards in Grafana to visualize the rates and traffic flow:

The image shows two screenshots of the Prometheus web interface.

The top screenshot displays the "Status > Target health" page. It shows a single target named "traefik" with the URL `http://traefik.default.svc.cluster.local:9100/metrics`. The status is "1 / 1 up" with a green "UP" indicator. Below the target details are "Last scrape" and "State" metrics: "4.406s ago" and "2ms".

The bottom screenshot shows the "Metrics browser" interface. A search bar at the top contains the query `>_ traefik`. Below the search bar is a tree view of metrics under the "traefik" job. The metrics listed are:

- `traefik_config_last_reload_success` (gauge)
- `traefik_config_reloads_total` (counter)
- `traefik_entrypoint_request_duration_seconds_bucket` (counter)
- `traefik_entrypoint_request_duration_seconds_count` (counter)
- `traefik_entrypoint_request_duration_seconds_sum` (counter)
- `traefik_entrypoint_requests_bytes_total` (counter)
- `traefik_entrypoint_requests_tls_total` (counter)
- `traefik_entrypoint_requests_total` (counter)
- `traefik_entrypoint_responses_bytes_total` (counter)
- `traefik_open_connections` (gauge)
- `traefik_service_request_duration_seconds_bucket` (counter)
- `traefik_service_request_duration_seconds_count` (counter)
- `traefik_service_request_duration_seconds_sum` (counter)
- `traefik_service_requests_bytes_total` (counter)
- `traefik_service_requests_tls_total` (counter)
- `traefik_service_requests_total` (counter)
- `traefik_service_responses_bytes_total` (counter)



## Redeploying Using the Local Built Image

Just to double-check that I have done everything correctly, at least slightly, I am going to delete every object from the K8s cluster, including the Helm releases, and redeploy from scratch. This time, I will use the custom-built image created from the cloned repository provided in the email.

First, building the image:

```
gus@coxa-pc:~/foobar-api$ cat Dockerfile
FROM golang:1.21 as builder
WORKDIR /app

COPY go.mod go.sum ./
RUN go mod download

COPY . .
RUN CGO_ENABLED=0 go build -a -trimpath -ldflags="-s" -o whoami .
```

```
FROM alpine:latest
WORKDIR /app
COPY --from=builder /app/whoami .
EXPOSE 80
CMD [ "./whoami"]

gus@coxa-pc:~/foobar-api$ docker build -t local/foobar-api:la
test .
[+] Building 8.1s (16/16) FINISHED
docker:default
=> [internal] load build definition from Dockerfile
0.0s
=> => transferring dockerfile: 293B
0.0s
=> WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do n
ot match (line 1)
0.0s
=> [internal] load metadata for docker.io/library/golang:1.2
1
1.3s
=> [internal] load metadata for docker.io/library/alpine:lat
est
0.1s
=> [auth] library/golang:pull token for registry-1.docker.io
0.0s
=> [internal] load .dockerignore
0.0s
=> => transferring context: 2B
0.0s
=> [stage-1 1/3] FROM docker.io/library/alpine:latest@sha25
6:865b95f46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da
4eb62
0.1s
=> => resolve docker.io/library/alpine:latest@sha256:865b95f
46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da4eb62
0.1s
```

```
=> [internal] load build context
0.0s
=> => transferring context: 2.44kB
0.0s
=> [builder 1/6] FROM docker.io/library/golang:1.21@sha256:4
746d26432a9117a5f58e95cb9f954ddf0de128e9d5816886514199316e4a2
fb
0.1s
=> => resolve docker.io/library/golang:1.21@sha256:4746d2643
2a9117a5f58e95cb9f954ddf0de128e9d5816886514199316e4a2fb
0.1s
=> CACHED [builder 2/6] WORKDIR /app
0.0s
=> CACHED [builder 3/6] COPY go.mod go.sum ./
0.0s
=> CACHED [builder 4/6] RUN go mod download
0.0s
=> [builder 5/6] COPY . .
0.1s
=> [builder 6/6] RUN CGO_ENABLED=0 go build -a -trimpath -ld
flags="-s" -o whoami .
6.2s
=> CACHED [stage-1 2/3] WORKDIR /app
0.0s
=> CACHED [stage-1 3/3] COPY --from=builder /app/whoami .
0.0s
=> exporting to image
0.2s
=> => exporting layers
0.0s
=> => exporting manifest sha256:9b77ff8256161d7e09f8eee57f62
b94039eba7060948697eef6f9e435a887b29
0.0s
=> => exporting config sha256:bcd0f6497b3c108814532c599a3538
0600b2840c0bab518c944b72dbdec2a8dd
0.0s
```

```
=> => exporting attestation manifest sha256:604195b7e9385ac9  
22d9775962423d4ee8bf764d39b77a71838f1c6b4eaa4513  
0.1s  
=> => exporting manifest list sha256:23b94cdf75e048a08774c45  
1e554b2a9b1e1ba4d25a5637b293b591f3a5613e7  
0.0s  
=> => naming to docker.io/local/foobar-api:latest  
0.0s  
=> => unpacking to docker.io/local/foobar-api:latest  
0.0s  
  
1 warning found (use docker --debug to expand):  
- FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)  
  
gus@coxa-pc:~/foobar-api$ docker images | grep local/foobar-api  
WARNING: This output is designed for human readability. For machine-readable output, please use --format.  
local/foobar-api:latest
```

Then, deleting everything

```
gus@coxa-pc:~/foobar-api$ helm uninstall traefik  
These resources were kept due to the resource policy:  
[PersistentVolumeClaim] traefik  
  
release "traefik" uninstalled  
  
gus@coxa-pc:~/traefik-exercise$ k delete -f .  
persistentvolumeclaim "foobar-api-certs-pvc" deleted from default namespace  
deployment.apps "foobar-api" deleted from default namespace  
service "foobar-api-svc" deleted from default namespace  
servertransport.traefik.io "trust-self" deleted from default namespace
```

```
ingressroute.traefik.io "foobar-api-ingress" deleted from default namespace
ingressroute.traefik.io "traefik-dashboard" deleted from default namespace
middleware.traefik.io "foobar-api-auth" deleted from default namespace
deployment.apps "jaeger" deleted from default namespace
service "jaeger" deleted from default namespace
configmap "prometheus-config" deleted from default namespace
deployment.apps "prometheus" deleted from default namespace
service "prometheus" deleted from default namespace
configmap "grafana-datasources" deleted from default namespace
deployment.apps "grafana" deleted from default namespace
service "grafana" deleted from default namespace
error: unable to decode "02-traefik-values.yaml": Object 'Kind' is missing in '{"additionalArguments":["--certificatesresolvers.myresolver.acme.storage=/data/acme.json","--certificatesresolvers.myresolver.acme.email=gustavosaviano@gmail.com","--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web","--tracing.otlp.grpc.endpoint=jaeger.default.svc.cluster.local:4317","--tracing.otlp.grpc.insecure=true"],"logs":{"access":{"enabled":true,"fields":{"general":{"defaultmode":"keep"},"headers":{"defaultmode":"keep"}}, "format":"json"}, "general":{"level":"DEBUG"}, "metrics":{"prometheus":{"enabled":true}}, "persistence":{"accessMode": "ReadWriteOnce", "enabled":true, "name": "data", "path": "/data", "size": "128Mi"}, "ports":{"metrics":{"expose":{"default":true}}, "exposedPort":9100, "port":9100, "protocol": "TCP"}}}
```

```
gus@coxa-pc:~/traefik-exercise$ k delete secret foobar-api-auth-secret
secret "foobar-api-auth-secret" deleted from default namespace
```

```
gus@coxa-pc:~/traefik-exercise$ k delete pvc traefik
```

```
persistentvolumeclaim "traefik" deleted from default namespace
```

```
gus@coxa-pc:~/traefik-exercise$ k get all
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      P
ORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        4
43/TCP   29h
```

Now updating the deployment:

```
gus@coxa-pc:~/traefik-exercise$ grep -A2 fernandobenegasa 01-
foobar.yaml
      #image: fernandobenegasa/foobar-api
      image: local/foobar-api:latest
      imagePullPolicy: Never
```

And redeploying (Ps: I had to tweak a few settings regarding the app/container ports (switching between 443 <> 80) because the image from Fernando seems slightly different from the one built directly from GitHub. Without these tweaks, the `foobar-api` pod would continue in a `CrashLoopBackOff` failure state):

```
gus@coxa-pc:~/traefik-exercise$ helm install traefik traefik/
traefik -f 02-traefik-values.yaml
NAME: traefik
LAST DEPLOYED: Sun Jan 18 14:43:49 2026
NAMESPACE: default
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
NOTES:
  traefik with docker.io/traefik:v3.6.6 has been deployed successfully on default namespace!

gus@coxa-pc:~/traefik-exercise$ k create secret generic fooba
```

```
r-api-auth-secret --from-literal=users='test:$apr1$0FCEKvM.$D
3qeQB/DIxqr18jLfaAQd.'
secret/foobar-api-auth-secret created

gus@coxa-pc:~/traefik-exercise$ k apply -f .
persistentvolumeclaim/foobar-api-certs-pvc created
deployment.apps/foobar-api created
service/foobar-api-svc created
servertransport.traefik.io/trust-self created
ingressroute.traefik.io/foobar-api-ingress created
ingressroute.traefik.io/traefik-dashboard created
middleware.traefik.io/foobar-api-auth created
deployment.apps/jaeger created
service/jaeger created
configmap/prometheus-config created
deployment.apps/prometheus created
service/prometheus created
configmap/grafana-datasources created
deployment.apps/grafana created
service/grafana created
error: error validating "02-traefik-values.yaml": error validating data: [apiVersion not set, kind not set]; if you choose to ignore these errors, turn validation off with --validate=false

gus@coxa-pc:~/traefik-exercise$ k get pods
NAME          READY   STATUS    RESTARTS   AGE
E
foobar-api-56d8fcc4fd-rcnq5   1/1     Running   0          41
S
grafana-79697b5f8-gs8z5       1/1     Running   0          41
S
jaeger-6c996c4fd4-lqdv6       1/1     Running   0          41
S
prometheus-54488fdf45-f8hz8  1/1     Running   0          41
S
```

traefik-86b65ff49f-bkb2c	1/1	Running	0	53s
--------------------------	-----	---------	---	-----

```
gus@coxa-pc:~/traefik-exercise$ k get all
```

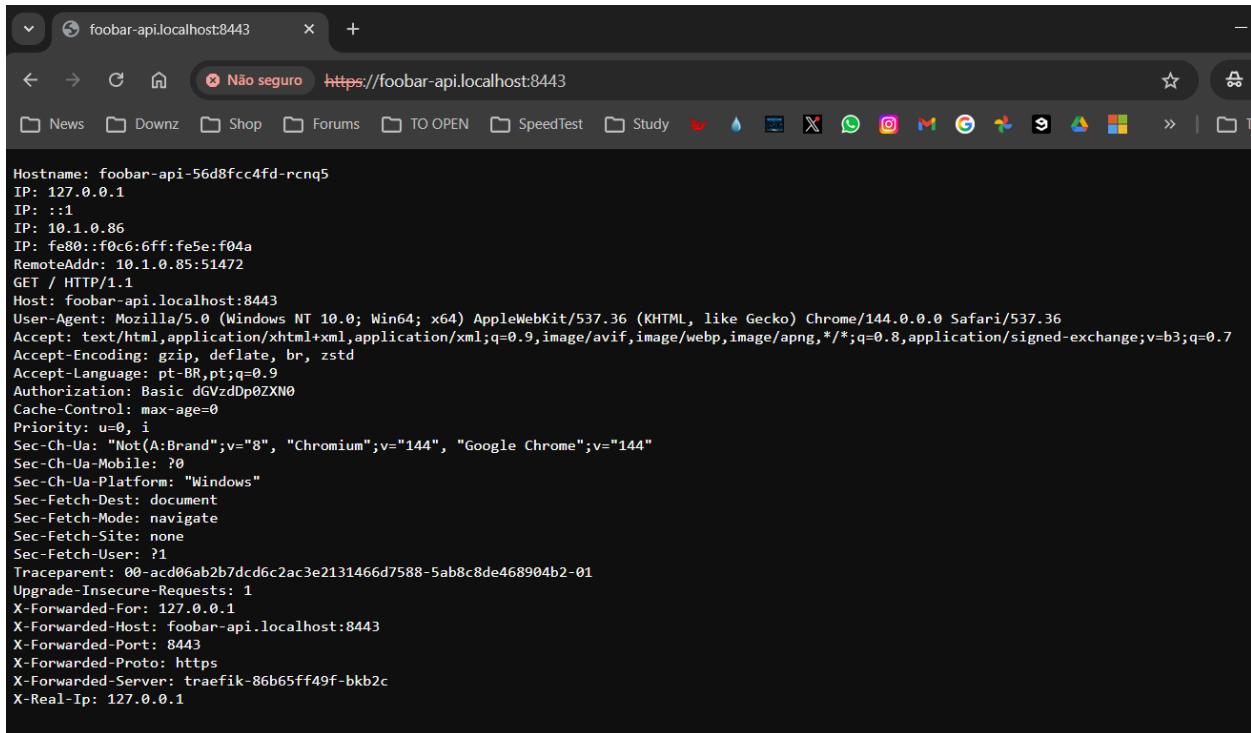
NAME	READY	STATUS	RESTARTS	AGE
pod/foobar-api-56d8fcc4fd-rcnq5	1/1	Running	0	45s
pod/grafana-79697b5f8-gs8z5	1/1	Running	0	45s
pod/jaeger-6c996c4fd4-lqdv6	1/1	Running	0	45s
pod/prometheus-54488fdf45-f8hz8	1/1	Running	0	45s
pod/traefik-86b65ff49f-bkb2c	1/1	Running	0	57s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	AGE
NAMESPACE	PORT(S)			
service/foobar-api-svc	ClusterIP	10.106.60.94	<none>	> 80/TCP 45s
service/grafana	ClusterIP	10.97.206.132	<none>	> 3000/TCP 45s
service/jaeger	ClusterIP	10.96.110.91	<none>	> 16686/TCP, 4317/TCP 45s
service/kubernetes	ClusterIP	10.96.0.1	<none>	> 443/TCP 29h
service/prometheus	ClusterIP	10.106.241.28	<none>	> 9090/TCP 45s
service/traefik	LoadBalancer	10.97.209.191	local	host 9100:31541/TCP, 80:32687/TCP, 443:31492/TCP 57s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/foobar-api	1/1	1	1	45s

deployment.apps/grafana	1/1	1	1
45s			
deployment.apps/jaeger	1/1	1	1
45s			
deployment.apps/prometheus	1/1	1	1
45s			
deployment.apps/traefik	1/1	1	1
57s			

NAME	READY	AGE	DESIRED	CURRENT	R
replicaset.apps/foobar-api-56d8fcc4fd	1	45s	1	1	1
replicaset.apps/grafana-79697b5f8		45s	1	1	1
replicaset.apps/jaeger-6c996c4fd4		45s	1	1	1
replicaset.apps/prometheus-54488fdf45		45s	1	1	1
replicaset.apps/traefik-86b65ff49f		57s	1	1	1



A screenshot of a browser window showing a request log. The URL is <https://foobar-api.localhost:8443>. The log output is as follows:

```
Hostname: foobar-api-56d8fcc4fd-rcnq5
IP: 127.0.0.1
IP: ::1
IP: 10.1.0.86
IP: fe80::f0c6:6fff:fe5e:f04a
RemoteAddr: 10.1.0.85:51472
GET / HTTP/1.1
Host: foobar-api.localhost:8443
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: pt-BR,pt;q=0.9
Authorization: Basic d6VzdDp0ZXN0
Cache-Control: max-age=0
Priority: u0, i
Sec-Ch-Ua: "Not(A:Brand";v="8", "Chromium";v="144", "Google Chrome";v="144"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Traceparent: 00-acd06ab2b7dc06c2ac3e2131466d7588-5ab8c8de468904b2-01
Upgrade-Insecure-Requests: 1
X-Forwarded-For: 127.0.0.1
X-Forwarded-Host: foobar-api.localhost:8443
X-Forwarded-Port: 8443
X-Forwarded-Proto: https
X-Forwarded-Server: traefik-86b65ff49f-bkb2c
X-Real-Ip: 127.0.0.1
```

## Refs

- <https://doc.traefik.io/traefik/getting-started/quick-start-with-kubernetes/>
- <https://doc.traefik.io/traefik/setup/kubernetes/>
- <https://doc.traefik.io/traefik/expose/kubernetes/>
- <https://doc.traefik.io/traefik/reference/routing-configuration/http/load-balancing/serverstransport/>
- <https://doc.traefik.io/traefik/reference/routing-configuration/http/middlewares/overview/>
- <https://doc.traefik.io/traefik/reference/routing-configuration/kubernetes/crd/http/middleware/>
- <https://doc.traefik.io/traefik/observe/logs-and-access-logs/>
- <https://doc.traefik.io/traefik/observe/metrics/>
- <https://doc.traefik.io/traefik/observe/tracing/>
- <https://www.jaegertracing.io/docs/1.76/getting-started/>

- <https://www.jaegertracing.io/docs/1.76/deployment/operator/#allinone-default-strategy>
- <https://www.jaegertracing.io/docs/1.76/deployment/>
- <https://opentelemetry.io/blog/2023/jaeger-exporter-collector-migration/>
- <https://doc.traefik.io/traefik/v3.0/observability/tracing/opentelemetry/>
- <https://doc.traefik.io/traefik/reference/install-configuration/api-dashboard/>
- <https://doc.traefik.io/traefik-enterprise/v2.0/operating/dashboard/>
- <https://prometheus.io/docs/prometheus/latest/configuration/configuration/>
- <https://grafana.com/docs/grafana/latest/setup-grafana/installation/kubernetes/>
- <https://docs.docker.com/get-started/docker-concepts/building-images/>