Announcing Tetrate Istio Subscription! 100% Upstream Istio, FIPS-Verified

Read More

tetrate

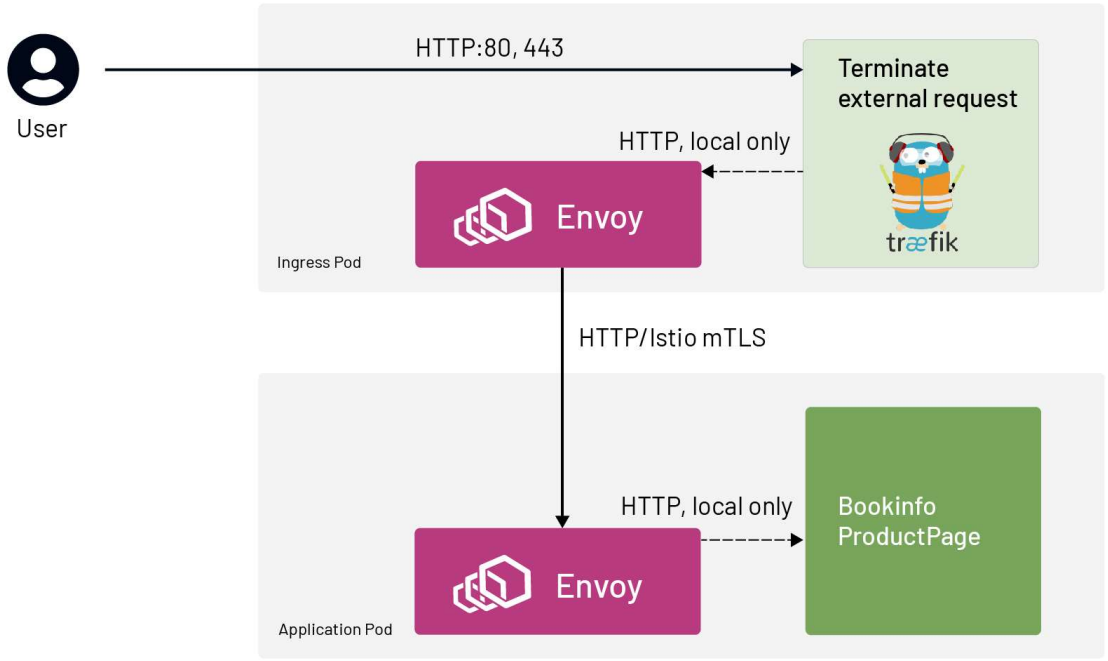
Using Traefik Ingress Controller with Istio Service Mesh

February 24, 2021 | Author: Petr Ankudinov



The Istio service mesh comes with its own ingress, but we see customers with requirements to use a non-Istio ingress all the time. Previously, we've covered integrating NGINX with Istio. Recently we've been working with customers that are using Traefik ingress. With some slight adjustments to the approach we suggested previously, we at Tetrate learned how to implement Traefik as the ingress gateway to your Istio Service Mesh. This article will show you how.

The flow of traffic is shown on the diagram below. As soon as requests arrive at the service mesh from the Traefik ingress, Istio has the ability to apply security, observability and traffic steering rules to the request:



Incoming traffic bypasses the Istio sidecar and arrives directly at Traefik, so the requests terminate at the Traefik ingress.

Traefik uses the `IngressRoute` config to rewrite the "Host" header to match the destination, and forwards the request to the targeted service, which is a several step process:

- Requests exiting Traefik Ingress are redirected to the Istio sidecar (by iptables).
- The sidecar receives the request, encrypts it (because our Istio `PeerAuthentication` policy dictates STRICT mTLS), and forwards the request to a pod of the target service.

Below is an end-to-end walkthrough of an example deployment, using Istio's bookinfo demo application but fronting the entire deployment with a Traefik ingress. In short, to get this to work in your own environment:

- Deploy Traefik controller with an Istio sidecar, annotating the deployment so that inbound traffic bypasses the Istio Sidecar:

```
# Exclude the ports that Traefik receives traffic
ontraffic.sidecar.istio.io/excludeInboundPorts: "80"
 # Make sure Traefik controller can talk to the Kubernetes API server


traffic.sidecar.istio.io/excludeOutboundIPRanges: X.X.X.X/32
```

- Enable Istio sidecar injection in the application namespace and deploy any Istio-specific config you need.
- Create `IngressRoute` with a Traefik `Middleware` object that rewrites the hostname to one recognized by the mesh (i.e. a service in the cluster; this is discussed below in the details with an example).

Bookinfo with Traefik Ingress

The rest of this post covers deploying Istio's Bookinfo sample application, using Traefik as the ingress proxy for the deployment.

Setting up the Environment

To follow this example yourself:

1. Deploy a Kubernetes cluster of at least version 1.17 (minimal Istio 1.8 supported version). We use a Google Kubernetes Engine cluster created by:

```
gcloud container clusters create istio-traefik \
  --cluster-version=1.17 \
  --region  \
  --machine-type=e2-standard-4 \
  --project  \
  --num-nodes 1 \
  --node-locations  # i.e us-west2-b (otherwise 1 node per zone)
```

2. Download Istio 1.8.x.

```
curl -sL https://git.io/getLatestIstio |\
 ISTIO_VERSION=1.8.1 sh -
```

3. Install it with enabled HTTP access logs.

```
./istio-1.8.1/bin/istioctl install \
  --set meshConfig.accessLogFile=/dev/stdout \
  --skip-confirmation
```

Deploying Bookinfo Application

With Istio installed, we can start deploying our application. We'll use Istio's Bookinfo application for our demonstration. This sample application is part of Istio distribution (in the `./istio-1.8.1/samples/` folder)

4. Create bookinfo namespace.

```
kubectl create ns bookinfo
```

5. Label it for the sidecar injection.

```
kubectl label namespace bookinfo istio-injection=enabled
```

6. Deploy bookinfo application in that namespace.

```
kubectl apply -f istio-1.8.1/samples/bookinfo/platform/kube/bookinfo.yaml
```

Confirm that all the pods are started and have sidecars deployed with them.

Enable Istio mTLS for Service-to-Service Communications for the Application Namespace

```
cat <<EOF | kubectl apply -f -
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: bookinfo
spec:
  mtls:
    mode: STRICT
EOF
```

Deploy Traefik Ingress

Now it's time to deploy Traefik by following v2.3 documentation (The most recent version of Traefik as of this post is 2.3, but it'll work with any version of Traefik if you adjust the `IngressRoute` and `Middleware` resources as required for your version). 7. Deploy Traefik constructs. Please note that there are some modifications to the documented deployment on the Traefik website (instead of default namespace in Traefik documentation, bookinfo namespace will be specified). The file can be accessed here and applied as follows:

```
$ kubectl apply -f https://bit.ly/Traefik-CRDs-and-Roles
customresourcedefinition.apiextensions.k8s.io/ingressroutes.traefik.contai
customresourcedefinition.apiextensions.k8s.io/middlewares.traefik.containo
customresourcedefinition.apiextensions.k8s.io/ingressroutetcps.traefik.con
customresourcedefinition.apiextensions.k8s.io/ingressrouteudps.traefik.con
customresourcedefinition.apiextensions.k8s.io/tlsoptions.traefik.containo.
customresourcedefinition.apiextensions.k8s.io/tlsstores.traefik.containo.u
customresourcedefinition.apiextensions.k8s.io/traefikservices.traefik.cont
clusterrole.rbac.authorization.k8s.io/traefik-ingress-lb created
clusterrolebinding.rbac.authorization.k8s.io/traefik-ingress-lb created
```

8. Create a service for incoming requests. The service will receive the external IP address. (There are a few changes to the example from the Traefik website):

a. The Namespace needs to be specified.

b. Only two ports are published: 80 for the Bookinfo application and 8080 for Traefik management.

c. The service needs to point to Traefik with a label (traefik-ingress-lb) that is used here.

d. " `Type: Loadbalancer` " is added to tell GCP to assign an external IP to the service.

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: traefik
  namespace: bookinfo
spec:
  ports:
    - protocol: TCP
      name: web
      port: 80
    - protocol: TCP
      name: admin
      port: 8080
  selector:
    app: traefik-ingress-lb
  type: LoadBalancer
EOF
```

9. Confirm that the service is created as expected:

```
$ kubectl get svc traefik -n bookinfo
NAME      TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
traefik   LoadBalancer   10.35.244.227   35.236.XXX.XXX   80:31718/TCP,808
```

10. As the Traefik website describes in detail, the Kubernetes Deployment with `ServiceAccount` needs to be applied. Besides the name and namespace, the following changes are introduced to the website example:

a. Secure endpoint removed for simplicity.

b. `Accesslog` – added " `=true` " as it didn't work without the value.

c. `Log.level` set to DEBUG will help us to see what's happening.

d. Added `traffic.sidecar.istio.io` annotations (For more details please refer to the previously mentioned Tetrate NGINX article).

KUBERNETES_SVC_IP=$(kubectl get svc kubernetes -n default -o jsonpath='{.spec.clusterIP}')

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  namespace: bookinfo
  name: traefik-ingress-lb

---
kind: Deployment
apiVersion: apps/v1
metadata:
  namespace: bookinfo
  name: traefik-ingress-lb
  labels:
    app: traefik-ingress-lb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: traefik-ingress-lb
  template:
    metadata:
      labels:
        app: traefik-ingress-lb
      annotations:
        traffic.sidecar.istio.io/excludeInboundPorts: "80"
        traffic.sidecar.istio.io/excludeOutboundIPRanges: ${KUBERNETES_SVC
    spec:
      serviceAccountName: traefik-ingress-lb
      containers:
        - name: traefik-ingress-lb
          image: traefik:v2.3
          args:
            - --api.insecure
            - --accesslog=true
            - --providers.kubernetescrd
            - --entrypoints.web.address=:80
            - --log.level=DEBUG
          ports:
```

```
            - name: web
              containerPort: 80
            - name: admin
              containerPort: 8080
EOF
```

11. Confirm the deployment of Traefik in the Bookinfo Namespace:

```
$  kubectl get pods -n bookinfo -l app=traefik-ingress-lb
NAME                                    READY   STATUS    RESTARTS   AGE
traefik-ingress-lb-669fc4b77d-74mpx   2/2     Running   0          2m35s
```

12. Get the service IP and record BOOKINFO_IP variable value.

```
BOOKINFO_IP=$(kubectl -n bookinfo get service traefik -o jsonpath='{.statu
```

13. Test response from the Ingress port 80 and see that it doesn't have route to the application.

```
curl -I $BOOKINFO_IP
```

Make sure it returns "404 Not Found" — not-200 response is expected as the ingress rules are not implemented yet.

Configure Traefik Ingress Rules

1. Traefik's `Middleware` header rewrite functionality will allow Istio Service mesh to function correctly. In this example, the host needs to be defined as "productpage.bookinfo.svc". The header can be defined according to the Traefik documentation:

```
cat <<EOF | kubectl apply -f -
apiVersion: traefik.containo.us/v1alpha1
kind: Middleware
metadata:
  name: productpage-header
  namespace: bookinfo
```

```
  spec:
    headers:
      customRequestHeaders:
        Host: productpage.bookinfo.svc
  EOF
```

2. The final step is to specify the routing logic for the ingress requests, since the focus of this article is Service Mesh integration. The definition is very simple and forwards all incoming requests arriving on port 80 to the fronting bookinfo application service called ProductPage (serving traffic on port 9080). It also uses the middleware object created in the previous step:

```
  cat <
  apiVersion: traefik.containo.us/v1alpha1
  kind: IngressRoute
  metadata:
    name: productpage
    namespace: bookinfo
  spec:
    entryPoints:
      - web
    routes:
    - match: PathPrefix(\`/\`)
      kind: Rule
      middlewares:
      - name: productpage-header
      services:
      - name: productpage
        port: 9080
  EOF
```

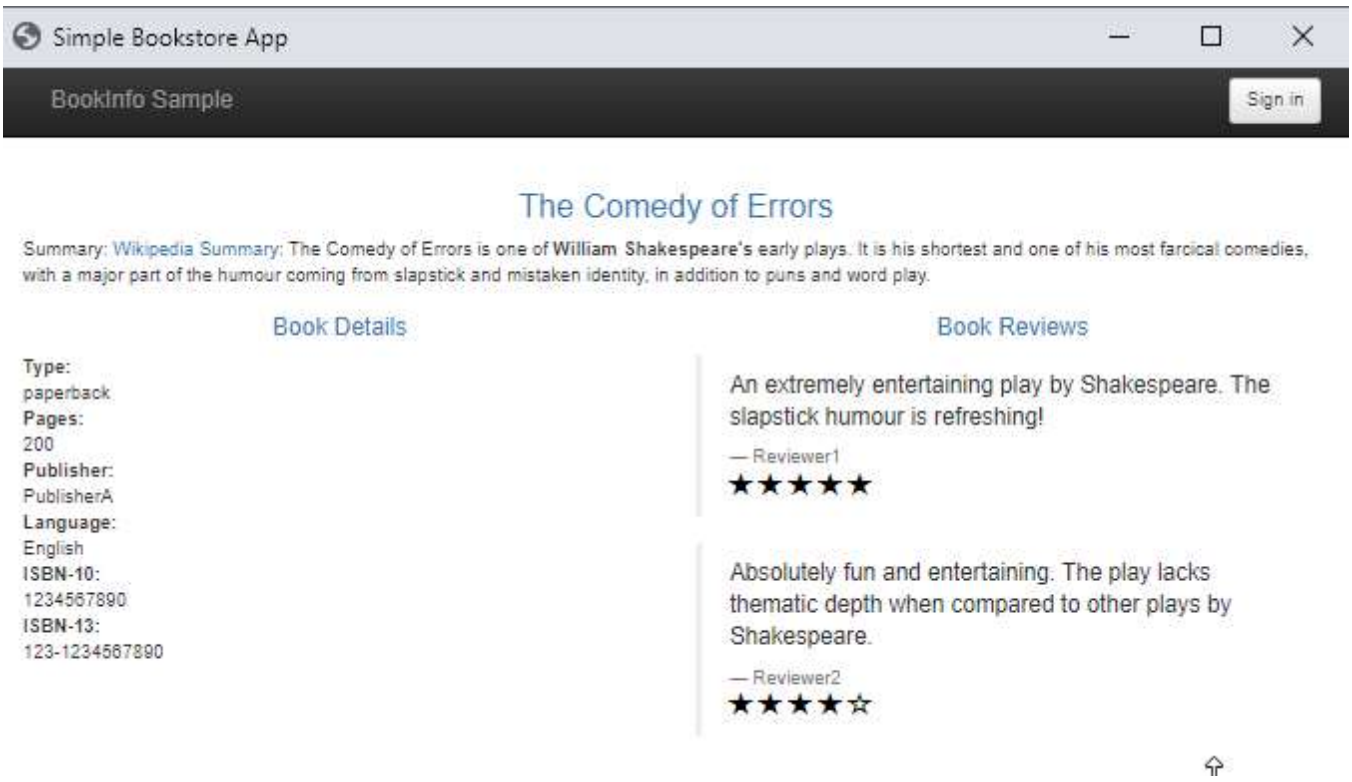
Validate Your Deployment Functionality

1. Retest the application response:

```
  curl -I $BOOKINFO_IP
```

We'll receive "200 OK" response. It can also be tested via the browser using https://<BOOKINFO_IP value from above>/productpage:

2. If /productpage is added to the url https://<BOOKINFO_IP value>/productpage, it will return the application response:



3. By querying the Traefik pod logs in the bookinfo namespace of the istio-proxy container, the outgoing request to the application can be seen in istio-proxy logs. There are no incoming requests, since they reach the Traefik Ingress directly.

```
TRAEFIK_POD=$( kubectl -n bookinfo get pods -l app=traefik-ingress-lb -o j

kubectl -n bookinfo logs ${TRAEFIK_POD} -c istio-proxy
```

Please note that the logs take a few seconds to be displayed after the request is processed. Logs are only available if the Istio install was done with the “meshConfig.accessLogFile=/dev/stdout” flag:

```
[2021-01-05T20:13:55.015Z] "GET /productpage HTTP/1.1" 200 - "-" 0 5179 10
[2021-01-05T20:13:56.301Z] "GET /static/bootstrap/fonts/glyphicons-halflin
```

Summary

This article demonstrates how Traefik Ingress can be implemented as an entry point to an Istio service mesh. The basic approach applied here should be

applicable even if your environment is different from the one used in our example. The Traefik / Service Mesh integration can be successfully implemented in different clouds with brand-new or existing (a.k.a. brownfield) deployments of Traefik, when the service mesh is introduced during Day Two implementations stage. In the end, you're getting the best of two worlds: Istio Service Mesh integrating with the Ingress controller of your choice!

This article was originally published in The New Stack.

Peter McAllister is a Tetrate engineer. Tetrate makes it easier for enterprises to adopt a service mesh and offers a service mesh management platform designed for multi-cluster and multi-cloud.

Let's Get In Touch

For more information or a demo request, just drop us a message.

Contact Us

Why Tetrate

Our Customers

Products

Tetrate Istio Subscription – Commercial Istio Support

Tetrate Istio Distro

Tetrate Enterprise Envoy Gateway

Tetrate Service Express

Tetrate Service Bridge

Solutions

Learn

FIPS

Government

Zero Trust

Blog

Resources

Learning Center

Tetrate Academy

Documentation

Events

Company

About

Leadership

Investors

Partners

Newsroom

Careers

Get Support

Contact Us

Legal

Subscribe to our newsletter

Email

Privacy Policy

Submit

Follow us

Terms of Use Privacy Policy

Copyright © Tetrate 2023. All rights reserved.

