

How to Use Istio for Ingress Gateway without TLS Termination on VKE

Learn how to configure Istio as an ingress gateway on VKE without TLS termination. Step-by-step guide for Kubernetes traffic management without SSL offloading.

Contents

| | | |
|----|---|----|
| 01 | Introduction | 3 |
| 02 | Prerequisites | 3 |
| 03 | 1. Install Istio on Vultr Kubernetes Engine (VKE) | 3 |
| 04 | 2. Set up NGINX for SSL Termination | 4 |
| 05 | 3. Deploy an Ingress Gateway | 7 |
| 06 | 4. Deploy a Virtual Service | 8 |
| 07 | 5. Set up DNS Record for Your Domain | 10 |
| 08 | More Information | 10 |

Introduction

Istio is an open-source service mesh that provides a way to manage the communication and data sharing between microservices. Istio gives you the ability to find-grained control over what happens to your traffic. Istio simplifies the process of setting up important tasks such as A/B testing, canary rollouts, and percentage-based traffic splits.

Vultr Kubernetes Engine (VKE) is a fully-managed Kubernetes product. When deploying an application to VKE, Kubernetes automatically spreads Pods across different nodes in a cluster for better availability.

This article explains how to deploy and configure Istio for Ingress Gateway without TLS Termination on VKE, using an NGINX server to handle TLS/SSL traffic and Istio Ingress Gateway to send encrypted SSL requests directly to the NGINX server.

Prerequisites

Before you begin, you should:

- Deploy a Vultr Kubernetes Cluster with at least 4GB of RAM per node.
- Configure `kubectl` and `git` on your machine.
- Have a fully-qualified domain name registered in DNS. This guide uses **nginx.example.com**.

1. Install Istio on Vultr Kubernetes Engine (VKE)

1. Go to [istio release page](#) to download the `istioctl` for your OS or run the following command to download the latest release automatically into your local machine (Linux or Mac OS)

```
$ curl -L https://istio.io/downloadIstio | sh -
```

2. Install `istio` with `istioctl`

```
$ istioctl install
```

3. The result should look like

```
This will install the Istio 1.14.1 default profile with ["Istio core" "Istiod"
"Ingress gateways"] components into the cluster. Proceed? (y/N) y
✓ Istio core installed
✓ Istiod installed
✓ Ingress gateways installed
✓ Installation complete
Making this installation the default for injection and validation.

Thank you for installing Istio 1.14.
```

4. If you don't have enough resources, the installation can't finish. Check the events for more information.

```
$ kubectl get events -n istio-system
```

2. Set up NGINX for SSL Termination

This section shows how to deploy an NGINX server for HTTPS traffic on port 443. This NGINX server also redirects all HTTP traffic on port 80 to the corresponding HTTPS endpoint.

Here are some approaches to obtaining TLS/SSL Certificates:

- **Self-Signed Certificates:** Use your own Certificate Authority to create and sign TLS/SSL certificates. This is a great option for development environments.
- **Purchase TLS/SSL Certificates:** You need to buy a TLS/SSL certificate from a well-known Certificate Authority for production use-cases.

- **Use Free TLS/SSL Certificates:** Use free TLS/SSL certificates from Let's Encrypt or ZeroSSL.

1. Download a copy of your TLS/SSL certificates. You need:

```
* A certificate bundle file contains your TLS/SSL certificate, the intermediate certificate, and the root certificate in one file
* A private key of your TLS/SSL certificate.
```

2. Create a Kubernetes Secret to hold the TLS certificates. Replace `nginx.example.com.key` and `nginx.example.com.crt` with your certificate filenames.

```
$ kubectl create secret tls nginx-server-certs --key nginx.example.com.key --cert nginx.example.com.crt
```

3. Create a file named `nginx.conf` for the NGINX config. The NGINX server performs SSL termination with the above certificate. This also configures the NGINX server to redirect all the HTTP traffic to HTTPS. Replace `nginx.example.com` with your domain name.

```
events {
}

http {
    log_format main '$remote_addr - $remote_user [$time_local] $status '
        '$request' $body_bytes_sent "$http_referer" '
        '$http_user_agent' "$http_x_forwarded_for";
    access_log /var/log/nginx/access.log main;
    error_log /var/log/nginx/error.log;

    server {
        listen 80;
        server_name nginx.example.com;
        return 301 https://$host$request_uri;
    }

    server {
        listen 443 ssl;

        root /usr/share/nginx/html;
```

```
index index.html;

server_name nginx.example.com;
ssl_certificate /etc/nginx-server-certs/tls.crt;
ssl_certificate_key /etc/nginx-server-certs/tls.key;
}
}
```

4. Create the ConfigMap with the above `nginx.conf`.

```
$ kubectl create configmap nginx-configmap --from-file=nginx.conf=./nginx.conf
```

5. Create a deployment file named `deployment.yml`. This file includes a `Service` to route the traffic to the correct pods.

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
    - port: 443
      protocol: TCP
      name: https
    - port: 80
      protocol: TCP
      name: http
      targetPort: 80
  selector:
    run: my-nginx
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 1
  template:
```

```
metadata:
  labels:
    run: my-nginx
spec:
  containers:
    - name: my-nginx
      image: nginx
      ports:
        - containerPort: 443
        - containerPort: 80
      volumeMounts:
        - name: nginx-config
          mountPath: /etc/nginx
          readOnly: true
        - name: nginx-server-certs
          mountPath: /etc/nginx-server-certs
          readOnly: true
  volumes:
    - name: nginx-config
      configMap:
        name: nginx-configmap
    - name: nginx-server-certs
      secret:
        secretName: nginx-server-certs
```

6. Deploy the `deployment.yml` with `istioctl` to automatically inject the Istio sidecar proxy to the NGINX pod.

```
$ istioctl kube-inject -f deployment.yml | kubectl apply -f -
```

3. Deploy an Ingress Gateway

This section configures an Istio Ingress Gateway to manage the ingress traffic. This gateway exposes the Virtual Service in the next step to users outside the Kubernetes Cluster.

Note that the `PASSTHROUGH` TLS mode instructs the gateway to pass the traffic to the target service without terminating TLS.

1. Create a file named `gateway.yml` as follows. Replace `nginx.example.com` with your domain name

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: mygateway
spec:
  selector:
    istio: ingressgateway # use istio default ingress gateway
  servers:
    - port:
        number: 80
        name: http
        protocol: HTTP
      hosts:
        - nginx.example.com
    - port:
        number: 443
        name: https
        protocol: HTTPS
      tls:
        mode: PASSTHROUGH
      hosts:
        - nginx.example.com
```

2. Deploy the `gateway.yml` with `kubectl`

```
$ kubectl apply -f gateway.yml
```

4. Deploy a Virtual Service

A Virtual Service lets you configure how to route the traffic to a Service within the cluster.

This section configures routes for traffic entering via the gateway (`mygateway`) in the previous step.

1. Create a file named `virtual-service.yml` with the following routing rules. Replace `nginx.example.com` with your domain name.

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: nginx
spec:
  hosts:
    - nginx.example.com
  gateways:
    - mygateway
  http:
    - match:
        - uri:
            prefix: /
      route:
        - destination:
            host: my-nginx
            port:
              number: 80
    - match:
        - port: 443
          sniHosts:
            - nginx.example.com
      route:
        - destination:
            host: my-nginx
            port:
              number: 443
```

2. Deploy the `virtual-service.yml` with `kubectl`

```
$ kubectl apply -f virtual-service.yml
```

5. Set up DNS Record for Your Domain

1. Find the IP of the load balancer. You can also go to [the Load Balancer page in the Customer Portal](#) to inspect your Load Balancers.

```
$ kubectl get service istio-ingressgateway -n istio-system
```

2. Create an A Record on your domain, such as `nginx.example.com`, that points to the Vultr Load Balancer IP address.
3. Navigate to `https://<YOUR_DOMAIN>` to access your NGINX server.

More Information

To learn more about Istio, see [the project documentation](#).



VULTR

