This guide shows you how to configure Google Kubernetes Engine or Kubernetes pod hosts and the load balancing components that Traffic Director requires.

The configuration examples are for demonstration purposes. You might need to deploy additional components based o
nment and requirements.

Before you follow the instructions in this guide, review Preparing for Traffic Director setup (/traffic-director/docs/setting-up-traffic-director) and make sure that you have completed the prerequisites.

This section describes the required steps to enable GKE/Kubernetes clusters to work with Traffic Director.

GKE clusters must meet the following requirements:

- Network Endpoint Groups support must be enabled. For more information and examples, refer to Standalone network endpoint groups (/kubernetes-engine/docs/how-to/standalone-neg). The standalone NEGs feature is available in General Availability for Traffic Director.
- The cluster nodes instances' service account must have permission to access the Traffic Director API. For more information on the required permissions, refer to Enabling the service account to access the Traffic Director API (/traffic-director/docs/setting-up-traffic-director#enable-service-account).
- The containers must have access to the Traffic Director API, protected by OAuth authentication. For more information, refer to host configuration (/traffic-director/docs/set-up-gce-vms#configure-vm-host).

The following example shows how to create a GKE cluster called `traffic-director-cluster` in the `us-central1-a` zone.

For GKE, switch to the cluster(2) you just created by issuing the following command. This points kubectl to the correct cluster.

This section shows how to prepare Kubernetes deployment specifications to work with Traffic Director. This consists of configuring services with NEGs as well as injecting sidecar proxies into pods that require access to the services managed by Traffic Director.

To verify that the backend pods are running, you must configure a firewall rule allowing the health checker IP address ranges.

To learn more, see configure firewal rule for health checks (/load-balancing/docs/health-checks#firewall_rules).

The first step in configuring GKE / Kubernetes services with NEGs is to expose the services that need to be managed by Traffic Director. To be exposed through NEGs, each specification must have the following annotation, matching the port that you want to expose.

For each service, a standalone NEG is created, containing endpoints that are the pod's IP addresses and ports. For more information and examples, refer to Standalone network endpoint groups (/kubernetes-engine/docs/how-to/standalone-neg).

For demonstration purposes, you can deploy a sample service that serves its hostname over HTTP on port 80:

Verify that the new service hostname is created and the application pod is running:

This returns:

This returns:

Find the NEG created from the example above and record the NEG's name.

The instructions in this section ensure that GKE services are accessible on the service VIP load balanced by Traffic Director, using a load balancing configuration similar to other Google Cloud Load Balancing (/load-balancing/docs/) products.

You must configure the following components:

- A health check. For more information on health checks, read Health Check Concepts (/load-balancing/docs/health-check-concepts) and Creating Health Checks (/load-balancing/docs/health-checks).
- A backend service. For more information on backend services, read Backend Services (/load-balancing/docs/backend-service).
- A route rule. This includes creating a forwarding rule and a URL map. For more information, read Using forwarding rules (/load-balancing/docs/forwarding-rules) and Using URL maps (/load-balancing/docs/https/url-map).

The Traffic Director configuration example that follows makes these assumptions:

1. The NEGs and all other resources are created in `default` network, with auto mode, in the zone `us-central1-a`.
2. The NEG name for the cluster is stored in the `${NEG_NAME}` variable.

Create the health check (/load-balancing/docs/health-checks).

Create a global backend service (/load-balancing/docs/backend-service#create-backend-service) with a load balancing scheme of **INTERNAL_SELF_MANAGED**. In the Cloud Console, the load balancing scheme is set implicitly. Add the health check to the backend service.

Use these instructions to create the route rule, forwarding rule, and internal IP address for your Traffic Director configuration.

Traffic sent to the internal IP address is intercepted by the Envoy proxy and sent to the appropriate service according to the host and path rules.

The forwarding rule is created as a global forwarding rule with the `load-balancing-scheme` set to `INTERNAL_SELF_MANAGED`.

You can set the address of your forwarding rule to `0.0.0.0`. If you do, traffic is routed based on the HTTP hostname and path information configured in the URL map, regardless of the actual IP address that the hostname resolves to. In this case, the URLs (hostname plus URL path) of your services, as configured in the host rules, must be unique within your service mesh configuration. That is, you cannot have two different services, with different set of backends, that both use the same hostname and path combination.

Alternatively, you can enable routing based on the actual destination VIP of the service. If you configure the VIP of your service as an `address` parameter of the forwarding rule, only requests destined to this IP address are routed based on the HTTP parameters specified in the URL map.

Regardless of the method selected, traffic to the VIP address that your service hostname resolves to must be intercepte
ecar proxy on the client host that needs to access your service. Refer to Configuring a single Compute Engine VM for Tra
or (#per_host_config) for full host configuration details.

Each forwarding rule in a VPC network must have a unique combination of IP address and port per VPC network, includ
0.0.0 address. If you create more than one forwarding rule with the same IP address and port in a particular VPC netw
e first forwarding rule is valid. The others are ignored.

At this point, Traffic Director is configured to load balance traffic for the services specified in the URL map across backends in the network endpoint group.

Depending on how your microservices are distributed on your network, you might need to add more forwarding rules or more host and path rules to the URL map.

This section shows how to reach Traffic Director backends from a client application.

To demonstrate functionality, you can deploy a sample pod running Busybox. The pod has access to `service-test`, which was created in the previous section and receives traffic that is load balanced by Traffic Director.

To access a service managed by Traffic Director, a pod must have an xDS API-compatible sidecar proxy installed.

The following steps provide a reference configuration for a given application deployment with sidecar injection.

1. Download the reference specification from https://storage.googleapis.com/traffic-director/trafficdirector_istio_sidecar.yaml (https://storage.googleapis.com/traffic-director/trafficdirector_istio_sidecar.yaml).
2. Modify your application deployment specification by adding two more container specifications from the `trafficdirector_istio_sidecar.yaml` file:
    a. Sidecar proxy container that sets up the Envoy bootstrap configuration and starts Envoy (see the example of Istio-proxy container defined in `trafficdirector_istio_sidecar.yaml`).
    b. Init container that sets up the required netfilter rules for interception, and runs with the permissions required to modify netfilter (see the example of initContainers defined in `trafficdirector_istio_sidecar.yaml`).
3. (Optional) Specify the IP address range for traffic to be intercepted by a sidecar proxy. For this, replace the '-i "*"' parameter in the args section of the **initContainers** definition with an IP address of the service for which traffic is redirected. By default, all traffic is intercepted.
    a. For example, you can specify "-i "10.0.0.0/24" to only redirect traffic for the 10.0.0.0/24 range.
4. Re-deploy pods using the new deployment specification.

In this example, you deploy a Busybox client with an Istio-proxy sidecar and init containers added to the deployment using the reference specification:

The Busybox pod has two containers running. The first container is the client based on the Busybox image and the second container is the Envoy proxy injected as a sidecar. You can get more information

about the pod by running the following command:

Once configured, applications on pods that have a sidecar proxy injected can access services managed by Traffic Director services. To verify the configuration, you can access a shell on one of the containers.

If you used the demo configuration provided in this guide, you can execute the following verification command to make sure that the hostname of the serving pod is returned.

Note that, in this example, when the Busybox client makes requests to the backend service, each request is proxied by the sidecar proxy.

This demonstration application uses the Envoy proxy. Because of that, the client sees 'server: envoy' in the header of server responses.

To confirm this, use the following commands:

In this example, you created a forwarding rule using the VIP address 0.0.0.0. This means that Traffic Director forwards requests to the backend based on the `Host` header only. In this case, the destination IP address can be any address as long as the request host header matches the host defined in the URL map `service-test`.

To confirm that, run the following test commands: