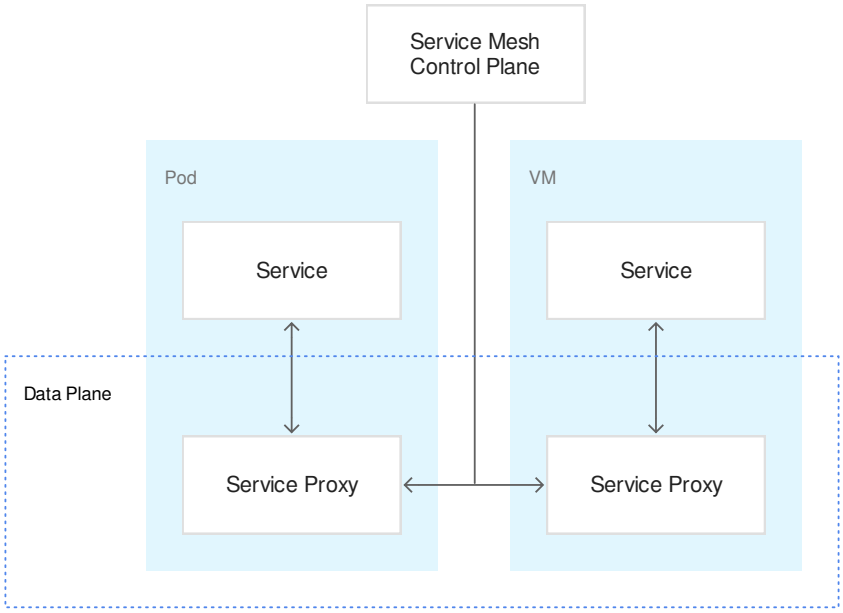
Service meshes are increasingly popular for deploying microservices and other modern applications. In a service mesh, the data plane uses service proxies, such as Envoy (https://www.envoyproxy.io/), to control traffic flow, while the service mesh control plane provides policy, configuration, and intelligence to the sidecar proxies.

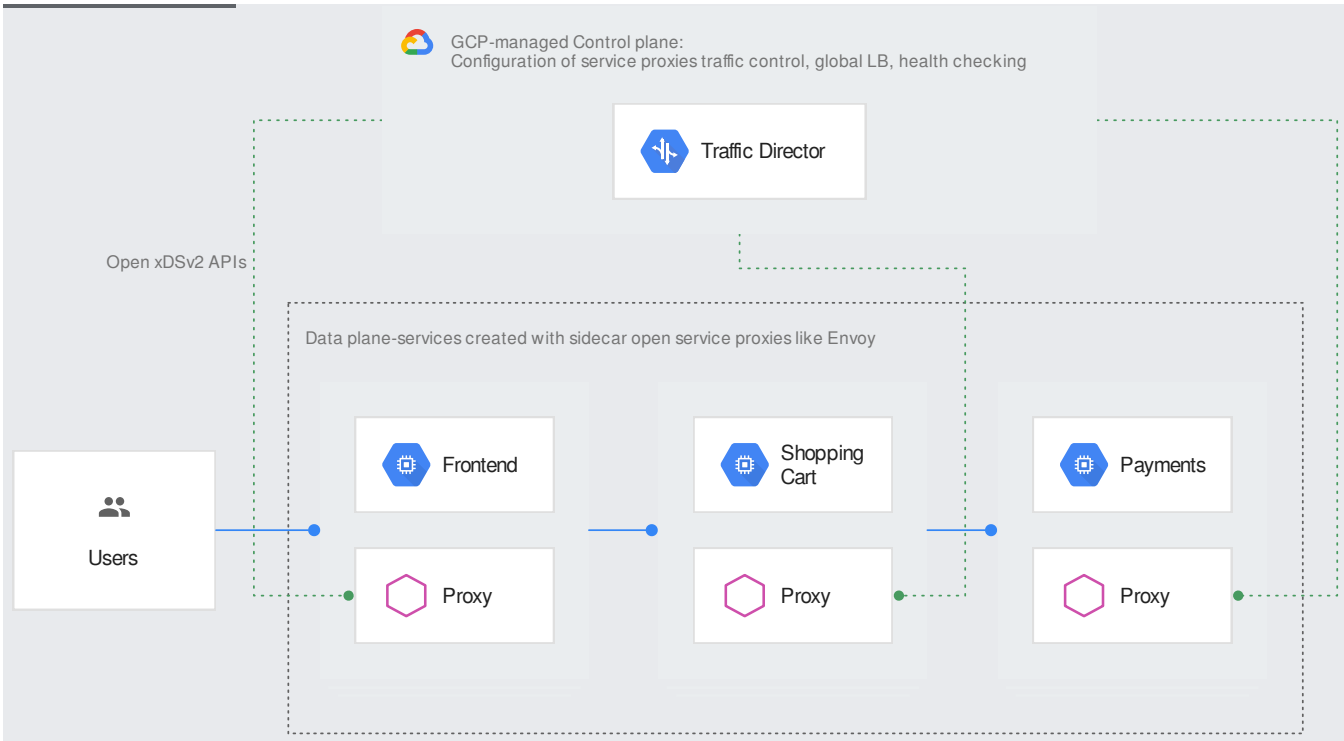


(/traffic-director/images/td-service-mesh.svg)

Service mesh as the control plane (click to enlarge)

The service mesh reduces the amount of networking code that you must write and maintain to run your services. Instead, the sidecar proxy performs networking functions alongside your business logic. A services control plane is required to manage the sidecar proxies.

Traffic Director is Google Cloud's fully-managed traffic control plane for service meshes. Traffic Director allows you to easily deploy global load balancing across clusters and VM instances in multiple regions and offload health checking from the sidecar proxies. Traffic Director uses open standard APIs (xDS v2) (https://www.envoyproxy.io/docs/envoy/latest/api-v2/api) to communicate with the sidecar proxies in the data plane, ensuring that you can use the service mesh data plane of your choice.



(/traffic-director/images/td-overview.svg)

Traffic Director as the control plane in a microservices environment (click to enlarge)

Traffic Director provides the following features:

- A Google Cloud (Google Cloud)-managed traffic management control plane for open standard (xDS v2) sidecar proxies such as Envoy.
- Ability to deploy Traffic Director-managed VM service instances running on managed instance groups (/compute/docs/instance-groups/) and on container instances that use network endpoint groups (/load-balancing/docs/negs/).
- Traffic management
- Service discovery for endpoints
- Global load balancing, because you can deploy services in multiple regions using a single service IP address
- Demand-driven autoscaling
- Request routing and traffic policies

- Health checking at scale
- Observability

Istio (https://istio.io/) provides a control plane to secure, connect, and monitor microservices. It has three components: Pilot for traffic management, Mixer for observability and Istio Security for service-to-service security.

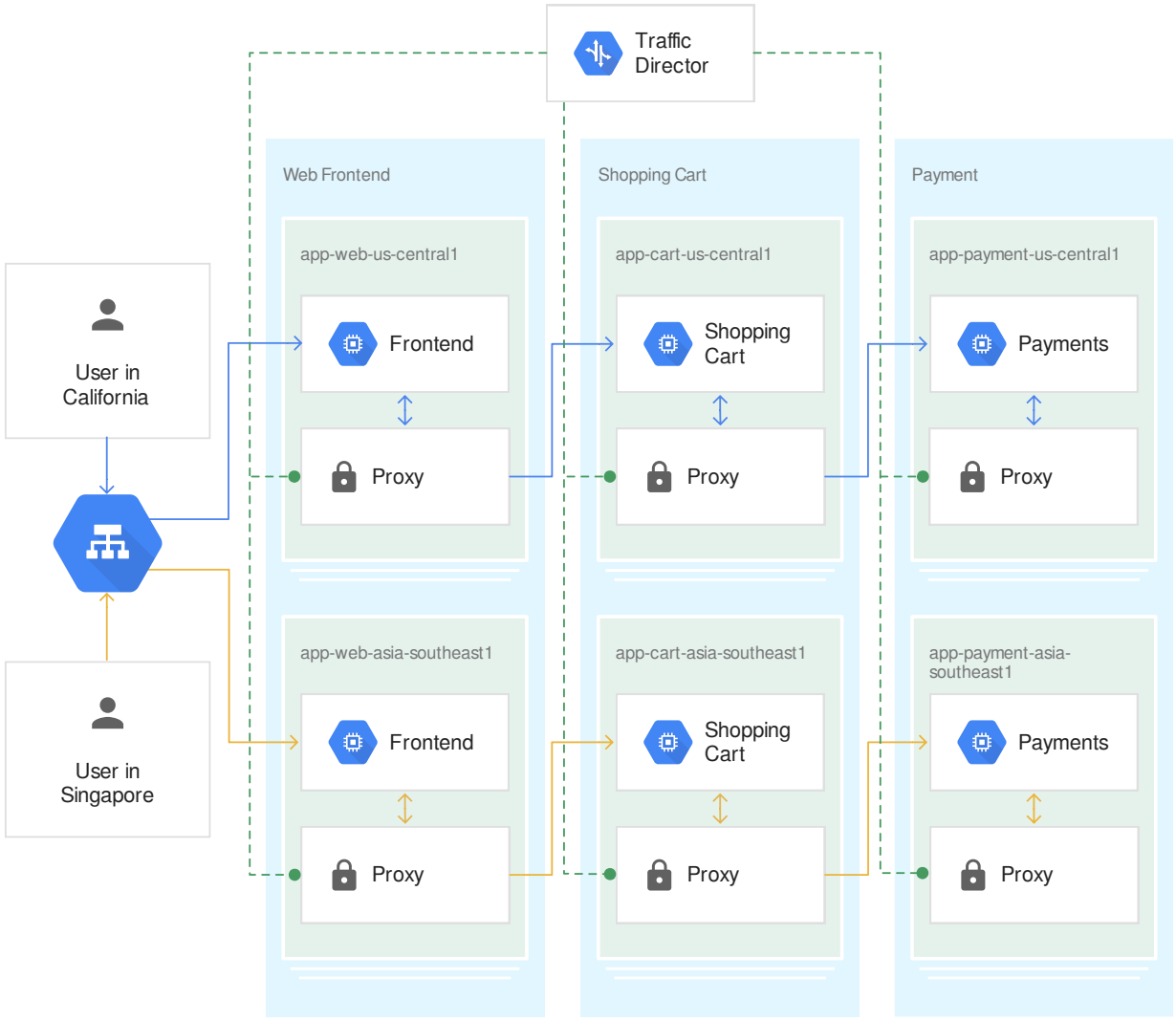
Traffic Director delivers a Google Cloud-managed Pilot along with additional capabilities such as global load balancing and centralized health checking. However, note that Traffic Director cannot be configured using Istio APIs; you can use GCP APIs for configuration. Both Traffic Director and Pilot use open standard APIs (xDS v2) to communicate with sidecar proxies.

Traffic Director delivers global load balancing for your internal microservices with sidecar proxies. You can deploy internal microservices (sidecar proxy-based) with instances in multiple regions. Traffic Director provides health, routing, and backend information to the sidecar proxies, enabling them to perform optimal traffic routing to application instances in multiple cloud regions for a service.

In the following illustration, user traffic enters a Google Cloud deployment through an external global load balancer. The external load balancer distributes traffic to the Front End microservice in either `us-central1` or `asia-southeast1`, depending on the location of the end user.

The internal deployment features three global microservices: Front End, Shopping Cart, and Payments. Each service runs on managed instance groups in two regions, `us-central1` and `asia-southeast1`. Traffic Director uses a global load balancing algorithm that directs traffic from the user in California to the microservices deployed in `us-central1`, while requests from the user in Singapore are directed to the microservices in `asia-southeast1`.

An incoming user request is routed to the Front End microservice. The service proxy installed on the host with the Front End then directs traffic to the Shopping Cart. The sidecar proxy installed on the host with the Shopping Cart directs traffic to the Payments microservice.



(/traffic-director/images/td-global-lb.svg)

Traffic Director in a global load balancing deployment (click to enlarge)

In the following example, if Traffic Director receives health check results indicating that the VMs running the Shopping Cart microservice in `us-central1` are unhealthy, Traffic Director instructs the sidecar proxy for the Frontend microservices to fail over traffic to the Shopping Cart microservice running in `asia-southeast1`. Because autoscaling is integrated with traffic management in Google Cloud, Traffic Director notifies the managed instance group in `asia-southeast1` of the additional traffic, and the managed instance group increases in size.

Traffic Director detects that all backends of the Payments microservice are healthy, so Traffic Director instructs Envoy's proxy for the Shopping Cart to send a portion of the traffic, up to the customer's configured capacity, to `asia-southeast1` and overflow the rest to `us-central1`.

Failover with Traffic Director in a global load balancing deployment (click to enlarge)

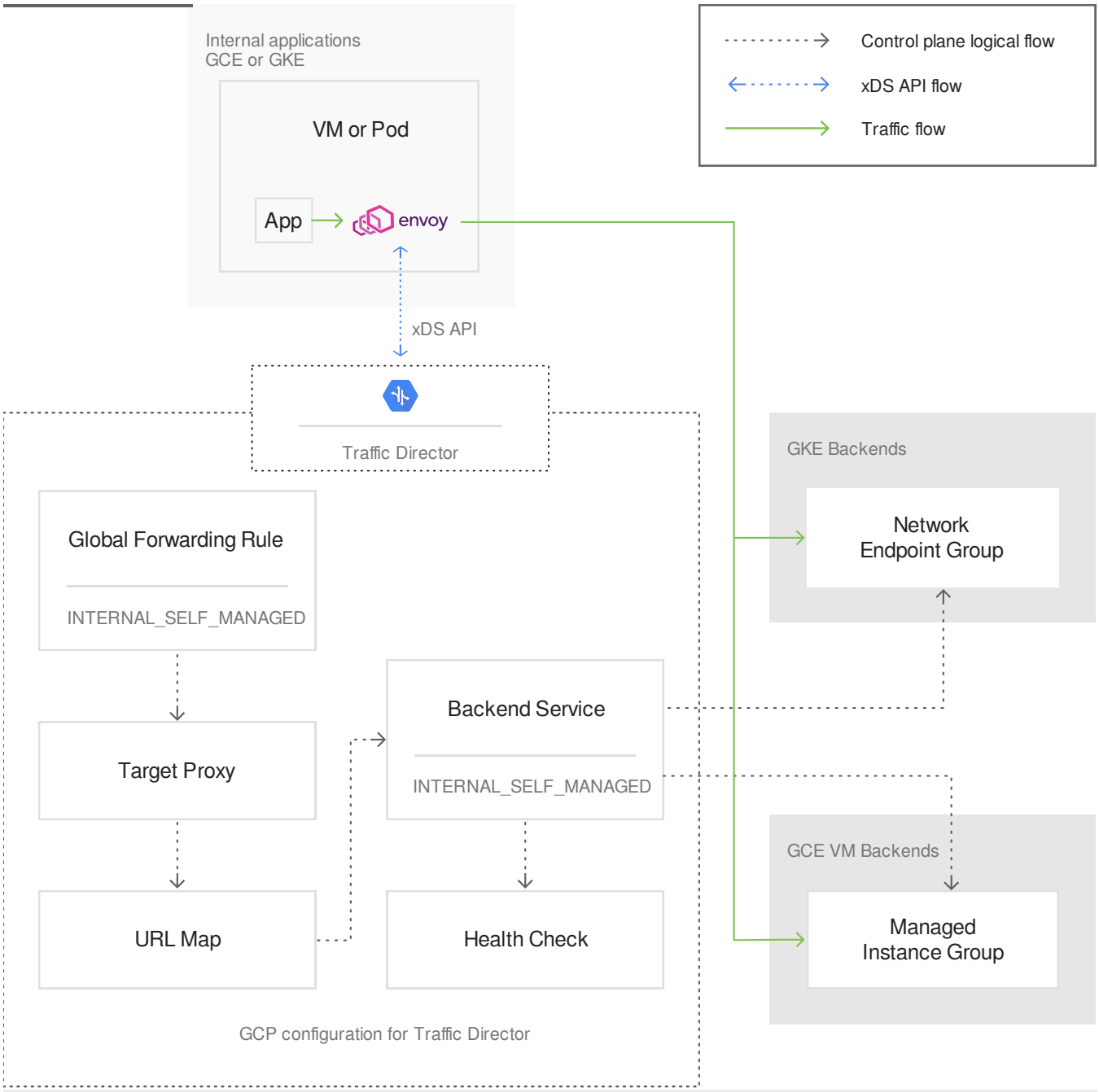
(/traffic-director/images/td-global-lb-failover.svg)

Failover with Traffic Director in a global load balancing deployment (click to enlarge)

During Traffic Director setup, you configure several load balancing components:

- A global forwarding rule, which includes the VIP address; the target proxy; and the URL map. All of these are part of Traffic Director's traffic routing mechanism. The target proxy must be a target HTTP proxy.
- The backend service, which contains configuration values.
- A health check, which provides health checking for the VMs and pods in your deployment.

The following diagram shows an application running on Compute Engine VMs or Google Kubernetes Engine pods, the components, and the traffic flow in a Traffic Director deployment:

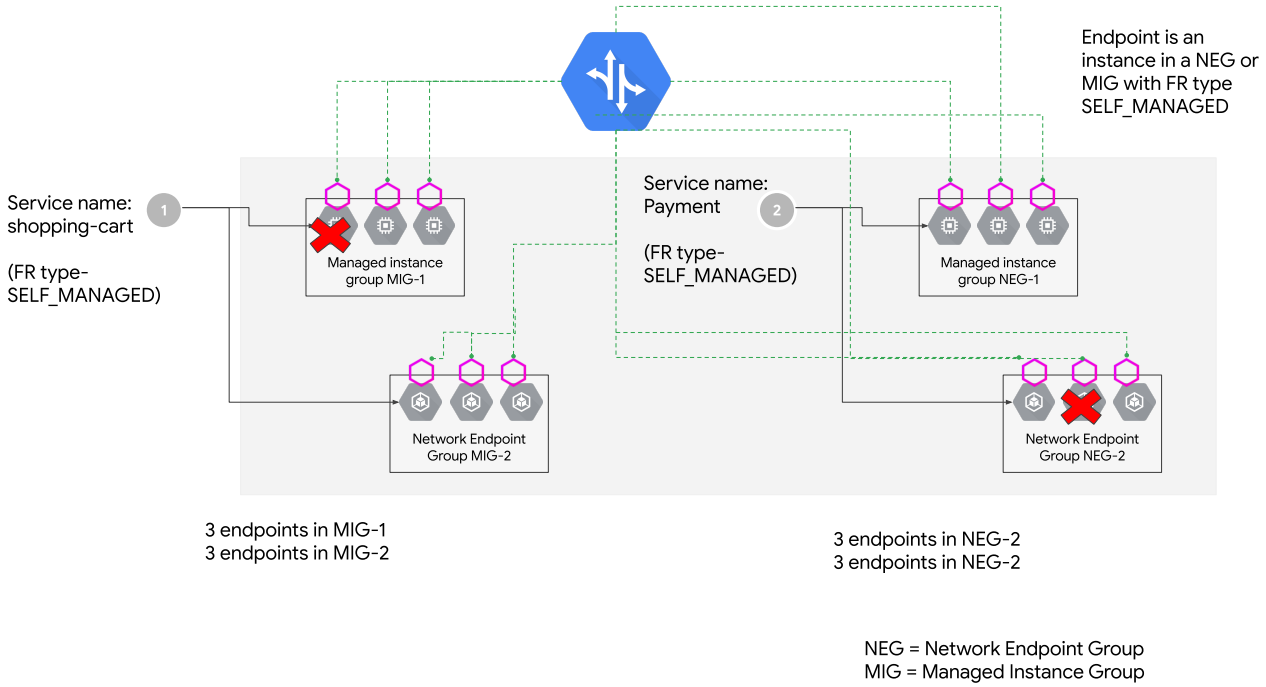


(/traffic-director/images/td-resources.svg)

Traffic Director resources to be configured (click to enlarge)

The diagram shows Traffic Director and the Google Cloud load balancing resources it uses to determine traffic routing. An xDS API-compatible sidecar proxy (such as Envoy, as shown) runs on a client VM instance or in a Kubernetes pod. Traffic Director serves as the control plane and communicates directly with each proxy using xDS APIs. In the data plane, the application sends traffic to the VIP address configured in the Google Cloud forwarding rule. The traffic is intercepted by the sidecar proxy and redirected to the appropriate backend. The next section contains more information about traffic interception and load balancing.

Traffic Director provides service discovery for both VM and container endpoints. You can add your service endpoints to either a managed instance group (MIG) for VMs or a network endpoint group (NEG) for containers. These service endpoints are associated with a service name. Traffic Director provides service discovery for services. It does this by mapping the hostname from the client to a target service name, and providing load balanced and health checked endpoints for the service name.



(/traffic-director/images/td-service-discovery.svg)

Traffic Director service discovery (click to enlarge)

In the above example, Traffic Director returns the two healthy endpoints in MIG-1 and three healthy endpoints in MIG-2 for the service `shopping-cart`. Beyond adding endpoints into MIGs or NEGs and setting up Traffic Director, you don't need any additional configuration to enable service discovery with Traffic Director.

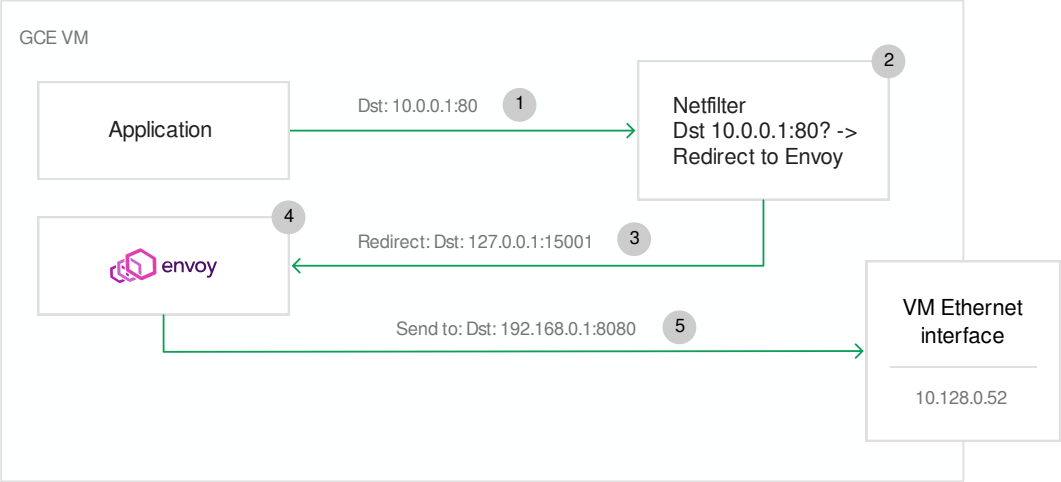
Traffic Director uses the sidecar proxy model. Under this model, when an application sends traffic to its destination, the traffic is intercepted by a sidecar proxy on the host where the

application is running. The sidecar proxy decides how to load balance the traffic, then sends the traffic to its destination.

In the following diagram, which assumes that Traffic Director is correctly configured, Envoy is the sidecar proxy. The sidecar proxy is running on the same host as the application.

A sample service, called **Web**, is configured on VIP 10.0.0.1, port TCP:80, where it can be discovered and load-balanced by Traffic Director. Traffic Director discovers the setup through forwarding rule configuration, which provides the VIP and port. The backends for the service **Web** are configured and function, but they are located outside the Compute Engine VM host in the diagram.

Traffic Director decides that the optimal backend for traffic to the service **Web** from the host is 192.168.0.1, TCP:8080.



(/traffic-director/images/td-vm-host-networking.svg)

Traffic Director host networking (click to enlarge)

The traffic flow in the diagram is this:

1. The application sends traffic to the service **Web**, which resolves to the IP address **10.0.0.1, TCP:80**.
2. The netfilter on the host is configured so that traffic destined to **10.0.0.1 TCP:80** is redirected to the IP address **127.0.0.1 TCP:15001**.
3. Traffic is redirected to **127.0.0.1:15001**, the interception port of the Envoy proxy.

4. The Envoy proxy interception listener on `127.0.0.1:15001` receives the traffic and performs a lookup for the original destination of the request (`10.0.0.1:80`). The lookup results in `192.168.0.1:8080` being selected as an optimal backend, as programmed by Traffic Director.
5. Envoy establishes a connection over the network with `192.168.0.1:8080` and proxies HTTP traffic between the application and this backend until the connection is terminated.

All existing forwarding rule, backend service, and other load balancing limits and quotas per project apply to Traffic Director deployments.

- Traffic Director only supports Google Cloud APIs for this release. Traffic Director does not support Istio APIs for this release.
- Traffic Director is supported only with HTTP traffic.
- Traffic Director supports Shared VPC (/vpc/docs/shared-vpc) either with the forwarding rule, target HTTP proxy, URL map, backend service, and backends in the host project or in the service project. However, only the service accounts of projects that have at least one forwarding rule configuration defined with the shared VPC network name can be used to access Traffic Director.
- Traffic Director does not support VPC Network Peering (/vpc/docs/vpc-peering).
- Traffic Director supports load balancing for clients only within a VPC network, the name of which is specified in the forwarding rule.
- You cannot use Traffic Director with services running in Knative or Google Cloud Serverless Computing (/serverless).
- You can only connect services running in Google Cloud using Traffic Director.
- You can only configure Google Cloud endpoints with Traffic Director in this release. We do not support endpoints on-prem or in another cloud.

- You can only configure Google Cloud backend VMs or endpoints in a NEG with Traffic Director in this release. We do not support VMs or endpoints on-premises or in another cloud.
- This document discusses Envoy proxies, but you can use any open standard API (xDS v2) proxy (https://www.envoyproxy.io/docs/envoy/latest/api-v2/api) with Traffic Director. However, note that Google has tested Traffic Director only with the Envoy proxy.
- Envoy must be version 1.9.1 or later to work with Traffic Director.
- We strongly recommend using the latest Envoy version to ensure that all known security vulnerabilities are mitigated.
- For information on Envoy security advisories, read Envoy Security Advisories (https://github.com/envoyproxy/envoy/security/advisories).

- Read Setting Up Traffic Director (/traffic-director/docs/setting-up-traffic-director) for information on configuring Traffic Director for your use case.