This guide shows you how to configure the Compute Engine VM hosts and the load balancing components that Traffic Director requires.

n: The configuration examples are for demonstration purposes. You might need to deploy additional components ba
r environment 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 prerequisite tasks described in that document.

This document provides the configuration process for services that run on Compute Engine VMs. The configuration process for the client VMs consists of setting up a sidecar proxy and traffic interception on a Compute Engine VM host. Then you configure load balancing using Google Cloud load balancing APIs.

This setup guide provides information on how to obtain and inject Envoy proxies from third-party sources that are not managed by Google.

When an application sends traffic to the service configured in Traffic Director, the traffic is intercepted by the xDS API-compatible sidecar proxy and then load balanced to the backends according to the configuration in the GCP load balancing components. For more information on host networking and traffic interception, read How traffic interception and load balancing work in Traffic Director (/traffic-director/docs/traffic-director-concepts#traffic-interception-host-networking).

For each VM host that requires access to Traffic Director services, perform the following steps:

1. Set the API access scope of the VM to allow full access to the Google Cloud APIs.
    - When you create the VMs, under **Identity and API access**, click **Allow full access to all Cloud APIs**.

      Go to the VM instances page (https://console.cloud.google.com/compute/instances).
    - With the `gcloud` command line tool, specify the following:

`--scopes=https://www.googleapis.com/auth/cloud-platform`.

2. Allow outgoing connections to `trafficdirector.googleapis.com` (TCP, port 443) from the VM, so that the sidecar proxy can connect to the APIs over gRPC. Outgoing connections to port 443 are enabled by default.
3. Deploy an xDS API-compatible sidecar proxy (such as Envoy), with a bootstrap configuration pointing to `trafficdirector.googleapis.com:443` as its xDS server. Refer to this sample bootstrap configuration file (https://storage.googleapis.com/traffic-director/traffic_director_sample_envoy_bootstrap.yaml) as an example.
4. Redirect IP traffic that is destined to the services to the sidecar proxy interception listener port.
    a. The sidecar proxy interception listener port is defined as `TRAFFICDIRECTOR_INTERCEPTION_PORT` in the proxy's bootstrap metadata configuration and is set to 15001 in the sample bootstrap configuration file (https://storage.googleapis.com/traffic-director/traffic_director_sample_envoy_bootstrap.yaml).
    b. The Istio `iptables` script (https://github.com/istio/cni/blob/master/tools/packaging/common/istio-iptables.sh) can be used to set up traffic interception.

For an example implementation of steps 3 and 4, read the next section.

Use this procedure as an example of how sidecar proxy deployment and traffic interception can be implemented to provide a VM with access to Traffic Director services.

First, download and prepare the configuration files and sample scripts.

1. Log in to the Linux host you are using during the setup process.
2. Download the archive of required files to the Linux host and untar the archive:

The archive contains the following files:

- **sidecar.env** – Config file with environment variables.

- **pull_envoy.sh** – Sample script to pull the Envoy binary from a provided Docker image tag. If no tag is provided, the script pulls from https://hub.docker.com/r/istio/proxyv2/tags (https://hub.docker.com/r/istio/proxyv2/tags).
- **iptables.sh** – Script for setting up netfilter rules.
- **bootstrap_template.yaml** – Bootstrap template file for Envoy.
- **run.sh** – Top-level script that uses the `sidecar.env` configuration file to set up `iptables` for interception and to run the Envoy sidecar proxy.

3. On each host that runs a sidecar proxy, create a system user to run the Envoy proxy process. This is the Envoy proxy user. Login is disabled for the Envoy proxy user.

4. Edit the `sidecar.env` file to modify the configuration. Read the inline comments in the configuration file for a detailed description of each variable.
    a. In the `sidecar.env` file, set the `ENVOY_USER` variable to the username that you choose to be the Envoy proxy user.
5. Copy your own Envoy binary into the `traffic-director` directory or follow these steps to obtain an Envoy binary:
    a. Install Docker tools (https://docs.docker.com/install/) on the Linux host that you are using. The files for each supported operating system are in the Supported platforms (https://docs.docker.com/install/#supported-platforms) section.
    b. Run the `pull_envoy.sh` script to extract the Envoy binary.

Next, for each VM host that runs applications using Traffic Director, perform the following steps:

1. Copy the entire `traffic-director` directory with the modified `sidecar.env` file and Envoy binary to each VM host running applications that you expect to use Traffic Director.
2. Add the `run.sh` script to your system's startup script, which ensures that the script is run after every VM reboot.
3. On each VM host, run the `run.sh` script:

4. Verify that the Envoy proxy started correctly.

   You should see the following output:

   Alternatively, you can confirm that the proxy process is running by using the `ps` command. Make sure that `envoy` appears in the output.

   You can verify traffic interception direction by using the following:

   The expected output is:

With this configuration, all TCP traffic, except traffic on port tcp/22, is intercepted by the sidecar proxy. Traffic to TCP
es that are not configured in Traffic Director are silently discarded by the proxy. To avoid losing traffic, follow the Adva
interception configuration (#advanced_interception) section below.

If intercepting all outgoing VM traffic is not acceptable for your deployment, you can redirect specific traffic to the sidecar proxy, while the rest of the traffic will follow the regular route defined by your host networking configuration.

To achieve this, modify the earlier Compute Engine VM host setup procedure as follows:

1. Decide on the range of IP addresses that Traffic Director-controlled services should resolve to. Traffic destined to these IP addresses is intercepted by the sidecar proxy. The range is specific to your deployment.
2. In the `sidecar.env` file, set the value of `SERVICE_CIDR` to this range. Traffic to these IP addresses is redirected by netfilter to a sidecar proxy and load balanced according to the configuration provided by Traffic Director.
3. The sidecar proxy is not strictly required to run as a dedicated user that is excluded from traffic interception. However, this is still recommended.
4. After you execute the `run.sh` script as directed in the main procedure, iptables is configured to intercept this specific range only.
5. Run the following command to verify that the netfilter is configured correctly. For ${SERVICE_CIDR}, substitute the value you configured as the intercepted IP address range.

Managed instance groups (/compute/docs/instance-groups/#managed_instance_groups) create new backend VMs by using autoscaling. If these backend VMs run applications that need access to Traffic Director services, you must install a sidecar proxy and configure traffic interception on the backend VMs.

If the managed instance group VMs serve as a backend only and its applications do not need access to Traffic Direct es, you can skip this section.

The following example configures each Compute Engine VM instance in a managed instance group similarly to the configuration described for a single Compute Engine VM in the previous section. This example shows how to:

- Create a VM template with a full Envoy configuration and a sample service that serves its hostname using the HTTP protocol.
- Configure a managed instance group using this template.

First, create the Compute Engine VM instance template. This template auto-configures the Envoy sidecar proxy and sample apache2 web service through the `startup-script` parameter.

If you don't have a managed instance group with services running, create a managed instance group (/sdk/gcloud/reference/compute/backend-services/create), using a VM template such as the one shown in the previous section. This example uses the instance template created in the previous section to demonstrate functionality. You do not have to use the instance template.

The instructions in this section configure Traffic Director load balancing for your services. You 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 firewall rule, to enable the health check probes to reach the backends. See Health Check Concepts (/load-balancing/docs/health-check-concepts) for more information.
- A backend service. For more information on backend services, read Backend Services (/load-balancing/docs/backend-service).
- A routing rule map. 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).

Use the following instructions to create a health check. For more information, refer to Creating health checks (/load-balancing/docs/health-checks).

Create the backend service. If you use the `gcloud` command-line tool, you must designate it as a global backend service (/load-balancing/docs/backend-service#create-backend-service) with a load

balancing scheme of **INTERNAL_SELF_MANAGED**. Add the health check and a managed or unmanaged instance group to the backend service. Note that this example uses the managed instance group with Compute Engine VM template that runs the sample HTTP service created in Creating the managed instance group (#create_mig).

At this point, Traffic Director is configured to load balance traffic for the services specified in the URL map across backends in the managed instance 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. For more information on forwarding rules and URL maps, read the following documents:

- Using forwarding rules (/load-balancing/docs/forwarding-rules)
- Forwarding rules REST resource (/compute/docs/reference/rest/v1/forwardingRules)
- Global forwarding rules REST resource (/compute/docs/reference/rest/v1/globalForwardingRules)
- Using URL maps (/load-balancing/docs/https/url-map)
- URL Map REST resource (/compute/docs/reference/rest/v1/urlMaps)

When the configuration is complete, each Compute Engine VM that has a sidecar proxy can access services configured in Traffic Director using the HTTP protocol.

If you followed the specific examples in this guide, using the Compute Engine VM template with the demonstration HTTP server and service hostname `service-test`, use these steps to verify the configuration.

1. Log in to one of the VM hosts that has a sidecar proxy installed.
2. Execute the command `curl -H 'Host: service-test' 10.0.0.1`. This request returns the hostname of the managed instance group backend that served the request.

In step 2, note that you can use any IP address. For example, the command `curl -I -H 'Host: service-test' 1.2.3.4` would work in Step 2.

This is because the forwarding rule has the address parameter set to 0.0.0.0, which instructs Traffic Director to match based on the host defined in the URL map. In the example configuration, the host name is `service-test`.