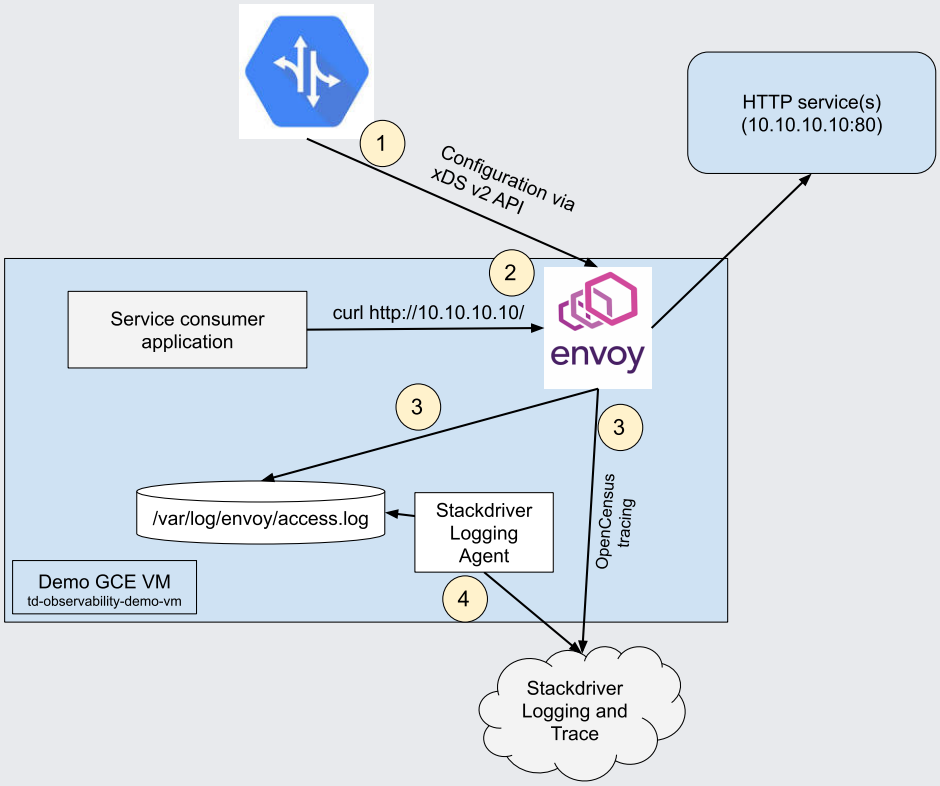
Using a service mesh gives you the ability to observe traffic to and from services, which allows for richer monitoring and debugging without code changes in the service itself. In the sidecar proxy architecture that Traffic Director uses, the proxy is the component that processes requests and provides the necessary telemetry information. Telemetry information must be collected and stored in a centralized location for further use, such as data analysis, alerting, and troubleshooting.

This guide demonstrates how to generate tracing (/trace/docs/) and logging (/logging/docs/) for the Envoy proxy. This guide also shows you how to export the information to Stackdriver Trace (/trace/docs/) and Stackdriver Logging (/logging/docs/).

Depending on the volume of requests handled by the application, enabling logging and or tracing of the requests can sidecar proxy performance. For more information, see the documentation for your xDS API-compatible proxy.

This guide uses the following configuration to demonstrate tracing and logging:

- A single application that listens on the HTTP port and returns the hostname of the VM that served the request. In the diagram, this application is in the upper-right-hand corner, labeled **HTTP service(s) (10.10.10.10:80)**. One or more VMs can provide this service.
- A single Compute Engine VM running a consumer of this service. In the diagram, this is labeled **Demo Compute Engine VM**.
- An Envoy sidecar proxy installed and configured by Traffic Director. In the diagram, this is labeled **Envoy**.
- A service consumer application, shown in the gray box, is the consumer of the HTTP service running on `10.10.10.10:80`.



(/traffic-director/images/td-stackdriver-envoy.svg)

Demonstration application for Stackdriver logging and monitoring for Envoy (click to enlarge)

1. Traffic Director configures the Envoy proxy to load balance traffic for the `10.10.10.10:80` service, to store access log information for each request issued for this service, and to generate tracing information for the service.
2. After the consumer sends a request to `10.10.10.10`, the sidecar proxy routes the request to the correct destination.
3. The Sidecar proxy also generates the necessary telemetry information:
    a. Adds an entry to the access log on the local disk with additional information about the request
    b. Generates a trace entry and sends it to Stackdriver Trace using OpenCensus Envoy tracing.
4. The logging agent exports this data to the Stackdriver Logging APIs, so that the data becomes available in the Stackdriver interface.

Ensure that:

1. The Traffic Director API is enabled and other prerequisites are met, as described in Preparing for Traffic Director setup (/traffic-director/docs/setting-up-traffic-director).
2. The service account that the Compute Engine VM uses has the **Cloud Trace Agent** role configured. See the Stackdriver Trace Access Control (/trace/docs/iam) page for more information.
3. The service account that the Compute Engine VM uses has the **Logs Writer** role configured. See the Stackdriver Logging Access Control (/logging/docs/access-control) page for more information.

This guide uses several shell scripts to perform the steps required to configure the demonstration service. Review the scripts to understand the specific steps they perform.

1. Start a Compute Engine VM and configure the HTTP service on the VM.

   The `setup_demo_service.sh` script creates a VM template that launches apache2 when a VM starts and a managed instance group that uses this template. The script launches a single instance without autoscaling enabled.

2. Configure routing for the `10.10.10.10` service using Traffic Director

   The `setup_demo_trafficdirector.sh` script configures the necessary parameters for the Traffic Director managed service, similar to the configuration described in the Setting up Traffic Director for Compute Engine with VMs (/traffic-director/docs/set-up-gce-vms).

3. Start a Compute Engine VM that runs a consumer of the HTTP service, with the sidecar proxy installed and configured on the VM. In the command below, replace the variable `gcp_project_id` with the project ID to which Stackdriver Trace information should be sent. This is typically the same project to which your VM belongs.

   The `setup_demo_client.sh` script creates a Compute Engine VM that has an Envoy proxy preconfigured to use Traffic Director. This is similar to the configuration described in Setting up Traffic Director for Compute Engine with VMs (/traffic-director/docs/set-up-gce-vms).

The following additional configuration settings enable tracing and logging:

- The `TRAFFICDIRECTOR_ACCESS_LOG_PATH` and `TRAFFICDIRECTOR_ENABLE_TRACING` bootstrap node metadata variables enable logging and tracing, as described in the Configuring additional attributes for sidecar proxies (/traffic-director/docs/traffic-director-per-proxy-config).
- Static bootstrap configuration, enabling export of trace information to Stackdriver Trace using OpenCensus.

NOTE: You can add and modify additional tracing parameters through the Envoy runtime (https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_conn_man/runtime#config-http-conn-man-runtime-random-sampling) configuration.

After running these scripts, you can log in to the `td-observability-demo-client` VM and access the HTTP service available at `10.10.10.10`.

At this point, Envoy generates access log and tracing information. The following section describes how to export logs and tracing information to Stackdriver.

The Envoy bootstrap configuration that you created when you ran the `setup-demo-client.sh` script is sufficient to generate tracing information. All other configuration is optional. If you want to configure additional parameters, see the OpenCensus Envoy configuration page

(https://www.envoyproxy.io/docs/envoy/latest/api-v2/config/trace/v2/trace.proto#envoy-api-msg-config-trace-v2-opencensusconfig)
and modify the tracing options in the Envoy bootstrap configuration.

After you issue a sample request to the demonstration server (`curl 10.10.10.10`), go to the Stackdriver Trace interface (Stackdriver > Trace > Trace list (https://console.cloud.google.com/traces/traces)) in the Google Cloud Console. You see a trace record corresponding to the request that you issued.

For more information on how to use Stackdriver Trace, see its documentation (/trace/docs/).

At this stage, Envoy should be recording access log information to the local disk of the VM where it is running. To export these records to Stackdriver Logging (/logging/docs), you must install the Stackdriver Logging agent locally. This requires installing and configuring the Stackdriver agent.

Install the Stackdriver Logging agent on the VM from which logging information is exported. For this example configuration, the VM is `td-observability-demo-vm`.

For more information on Stackdriver Logging agent installation, see the guide Installing the agent (https://cloud.google.com/logging/docs/agent/installation).

You can export the Envoy logs as either unstructured or structured text.

This option exports log records from the access log to Stackdriver Logging as raw text. Each entry in the access log is exported as a single entry to Stackdriver Logging. This configuration is easier to install, because it relies on a parser that is distributed with the current version of the Stackdriver Logging agent, but it is more difficult to filter and process raw text log entries using this option.

1. Download and install the Envoy access log unstructured export config file.

2. Restart the agent. The changes take effect when the agent starts up.

1. Install the Envoy access log parser (https://github.com/salrashid123/fluent-plugin-envoy-parser) from GitHub.

2. Download and install the configuration file for exporting Envoy access logs in a structured format:

3. Restart the agent. The changes take effect when the agent starts up.

For more information on Stackdriver agent configuration, see Configuring the agent (https://cloud.google.com/logging/docs/agent/configuration).

1. From the sidecar proxy VM, generate a request to the demonstration service. This creates a new local log record. For example, you can run `curl 10.10.10.10`.
2. Go to Stackdriver Logging > Logs Viewer (https://console.cloud.google.com/logs/viewer) in the Cloud Console. In the drop-down menu, select the **envoy-access** log type. You see a log entry for the most recent request in the unstructured or structured format, depending on the configuration type you chose earlier.

If configuration is complete, but you do not see trace or logging entries available in Stackdriver, verify the following:

1. The service accounts for the Compute Engine VM have the necessary Stackdriver Trace and Logging IAM permissions, as specified in the prerequisites. For information on Stackdriver Trace IAM permissions, see Access control (/trace/docs/iam). For information on Stackdriver Logging permissions, see Access control (/logging/docs/access-control).
2. For logging: Ensure that there are no errors in /var/log/google-fluentd/google-fluentd.log.
3. For logging: Ensure that new entries appear in the local access log file when requests are issued.