This document provides information on how to configure advanced traffic management for your Traffic Director deployment.

Follow the instructions in Setting Up Traffic Director (/traffic-director/docs/setting-up-traffic-director), including configuring Traffic Director and any VM hosts or GKE clusters you need. Create the required Google Cloud resources.

These instructions assume the following:

- Your Traffic Director deployment has a URL map called `review-url-map`.
- The URL map sends all traffic to one backend service, called `review1`, which serves as the default backend service as well.
- You plan to route 5% of traffic to a new version of a service. That service is running on a backend VM or endpoint in a NEG associated with the backend service `review2`.
- No host rules or path matchers are used.

To set up traffic splitting, follow these steps:

After you are satisfied with the new version, you can gradually adjust the weights of the two services and eventually send all traffic to `review2`.

Advanced traffic management enables you to configure session affinity based on a provided cookie. To configure HTTP_COOKIE based session affinity for a backend service named `service1`, follow these directions.

To set up session affinity using `HTTP_COOKIE`:

`MetadataFilters` are enabled with forwarding rules and `HttpRouteRuleMatch`. Use this feature to control a particular forwarding rule or route rule so that the control plane sends the forwarding rule or route rule only to proxies whose node metadata matches the metadatafilter setting. If you do not specify any `MetadataFilters`, the rule is sent to all Envoy proxies.

This feature makes it easy to operate a staged deployment of a configuration. For example, create a forwarding rule named `forwarding-rule1`, which you want to be pushed only to Envoys whose node metadata contains 'app: review' and 'version: canary'.

Follow the steps below to add `MetadataFilters` to the forwarding rule.

To add `MetadataFilter` to a forwarding rule, follow these steps:

1. Use the `gcloud export` command to get the forwarding rule config.

2. Update the `forwarding-rule1-config.yaml` file.

3. Use the `gcloud import` command to update the `forwarding-rule1-config.yaml` file.

4. Use these instructions to add node metadata to Envoy before starting Envoy. Note that only string value is supported.

   a. For a VM-based deployment, in bootstrap_template.yaml, add the following under metadata section:

   b. For Google Kubernetes Engine-based or Kubernetes-based deployment, in trafficdirector_istio_sidecar.yaml, add the following under the env section:

Use the following instructions for a scenario in which multiple projects are in the same Shared VPC network and you want each service project's Traffic Director resources to be visible to proxies in the same project. This example corresponds to the first use case in Forwarding rule resource (/traffic-director/docs/traffic-control#forwarding-rule-resource). The second use case is configured similarly, but using different metadata key-value pairs.

The Shared VPC set-up is as follows:

- Host project name: `vpc-host-project`
- Service projects: `project1`, `project2`
- Backend services with backend instances or endpoints running an xDS compliant proxy in `project1` and `project2`

To configure Traffic Director to isolate `project1`:

1. Create all forwarding rules in `project1` with the following metadata filter:

2. When you configure the proxies deployed to instances or endpoints in `project1`, include the following metadata in the node metadata section of the bootstrap file:

3. Verify that the proxies already deployed in `project2` did not receive the forwarding rule created in the first step. To do this, try to access services in `project1` from a system running a proxy in `project2`. For information on verifying that a Traffic Director configuration is functioning correctly, see Verifying the configuration (/traffic-director/docs/set-up-gce-vms#verifying_the_configuration).

The following example corresponds to the third use case in Forwarding rule resource (/traffic-director/docs/traffic-control#forwarding-rule-resource).

To test a new configuration on a subset of proxies before you make it available to all proxies:

1. Start the proxies that you are using for testing with the following node metadata. Do not include this node metadata for proxies that you are not using for testing.

2. For each new forwarding rule that you want to test, include the following metadata filter:

3. Test the new configuration by sending traffic to the test proxies, and make any necessary changes. If the new configuration is working correctly, only the proxies that you test receive the new configuration. The remaining proxies do not receive the new configuration and are not able to use it.
4. When you confirm that the new configuration works correctly, remove the metadata filter associated with it. This allows all proxies to receive the new configuration.

Use this information for troubleshooting when traffic is not being routed according to the route rules and traffic policies that you configured.

Symptoms:

- Increased traffic to services in rules above the rule in question.
- An unexpected increase in 4xx and 5xx HTTP responses for a given route rule.

Solution: Review the priority assigned to each rule, because route rules are interpreted in priority order.

When you define route rules, check to be sure that rules with higher priority (that is, with lower priority numbers) do not inadvertently route traffic that would otherwise have been routed by a subsequent route rule. Consider the following example:

- First route rule
    - Route rule match pathPrefix = '/shopping/';
    - Redirect action: send traffic to backend service `service-1`
    - Rule priority: 4
- Second route rule
    - Route rule match regexMatch = '/shopping/cart/ordering/.*';
    - Redirect action: send traffic to backend service `service-2`
    - Rule priority: 8

In this case, a request with the path '/shopping/cart/ordering/cart.html' is routed to `service-1`. Even though the second rule would have matched, it is ignored because the first rule had priority.

- If the value of `BackendService.sessionAffinity` is not NONE, and `BackendService.localityLbPolicy` is set to a load balancing policy other than `MAGLEV` or `RING_HASH`, the session affinity settings will not take effect.
- `UrlMap.headerAction`, `UrlMap.defaultRouteAction` and `UrlMap.defaultUrlRedirect` do not currently work as intended. You must specify `UrlMap.defaultService` for handling traffic that does not match any of the hosts in `UrlMap.hostRules[]` in that `UrlMap`. Similarly, `UrlMap.pathMatchers[].headerAction`, `UrlMap.pathMatchers[].defaultRouteAction` and `UrlMap.pathMatchers[].defaultUrlRedirect` do not currently work. You must specify `UrlMap.pathMatchers[].defaultService` for handling traffic that does not match any of the `routeRules` for that `pathMatcher`.
- The `gcloud import` command doesn't delete top-level fields of the resource, such as the backend service and the URL map. For example, if a backend service is created with settings for `circuitBreakers`, those settings can be updated via a subsequent `gcloud import` command. However, those settings cannot be deleted from the backend service. The resource itself can be deleted and recreated without the `circuitBreakers` settings.
- Import for forwarding rules doesn't work properly. An exported YAML file can't be re-imported. The workaround is to export the config file, make changes, delete the forwarding rule, and import the configuration file.