Getting Started

This section gets you started with a very simple configuration and provides some example configurations.

Envoy does not currently provide separate pre-built binaries, but does provide Docker images. This is the fastest way to get started using Envoy. Should you wish to use Envoy outside of a Docker container, you will need to build it.

These examples use the v2 Envoy API, but use only the static configuration feature of the API, which is most useful for simple requirements. For more complex requirements Dynamic Configuration is supported.

Quick Start to Run Simple Example

These instructions run from files in the Envoy repo. The sections below give a more detailed explanation of the configuration file and execution steps for the same configuration.

A very minimal Envoy configuration that can be used to validate basic plain HTTP proxying is available in configs/google_com_proxy.v2.yaml. This is not intended to represent a realistic Envoy deployment:

```
$ docker pull envoyproxy/envoy:latest
$ docker run --rm -d -p 10000:10000 envoyproxy/envoy:latest
$ curl -v localhost:10000
```

The Docker image used will contain the latest version of Envoy and a basic Envoy configuration. This basic configuration tells Envoy to route incoming requests to *.google.com.

Simple Configuration

Envoy can be configured using a single YAML file passed in as an argument on the command line.

The admin message is required to configure the administration server. The *address* key specifies the listening address which in this case is simply *0.0.0.0:9901*.

```
admin:
  access_log_path: /tmp/admin_access.log
  address:
    socket_address: { address: 0.0.0.0, port_value: 9901 }
```

The static_resources contains everything that is configured statically when Envoy starts, as opposed to the means of configuring resources dynamically when Envoy is running. The v2 API Overview describes this.

```
static_resources:
```

The specification of the listeners.

```
listeners:
- name: listener_0
  address:
    socket_address: { address: 0.0.0.0, port_value: 10000 }
  filter_chains:
  - filters:
    - name: envoy.http_connection_manager
      config:
        stat_prefix: ingress_http
        codec_type: AUTO
        route_config:
          name: local_route
          virtual_hosts:
          - name: local_service
            domains: ["*"]
            routes:
            - match: { prefix: "/" }
              route: { host_rewrite: www.google.com, cluster: service_google }
        http_filters:
        - name: envoy.router
```

The specification of the clusters.

```
clusters:
- name: service_google
  connect_timeout: 0.25s
  type: LOGICAL_DNS
  # Comment out the following line to test on v6 networks
  dns_lookup_family: V4_ONLY
  lb_policy: ROUND_ROBIN
  hosts: [{ socket_address: { address: google.com, port_value: 443 }}]
  tls_context: { sni: www.google.com }
```

Using the Envoy Docker Image

Create a simple Dockerfile to execute Envoy, which assumes that envoy.yaml (described above) is in your local directory. You can refer to the Command line options.

```
FROM envoyproxy/envoy:latest
COPY envoy.yaml /etc/envoy/envoy.yaml
```

Build the Docker image that runs your configuration using:

```
$ docker build -t envoy:v1 .
```

And now you can execute it with:

```
$ docker run -d --name envoy -p 9901:9901 -p 10000:10000 envoy:v1
```

And finally, test it using:

```
$ curl -v localhost:10000
```

If you would like to use envoy with docker-compose you can overwrite the provided configuration file by using a volume.

Sandboxes

We've created a number of sandboxes using Docker Compose that set up different environments to test out Envoy's features and show sample configurations. As we gauge peoples' interests we will add more sandboxes demonstrating different features. The following sandboxes are available:

- Front Proxy
- Zipkin Tracing
- Jaeger Tracing
- Jaeger Native Tracing
- gRPC Bridge
- Lua
- Fault Injection

Other use cases

In addition to the proxy itself, Envoy is also bundled as part of several open source distributions that target specific use cases.

- Envoy as an API Gateway in Kubernetes