Consul

Network Observability with Envoy

HashiCorp

Network Observability with Envoy

Nic Jackson

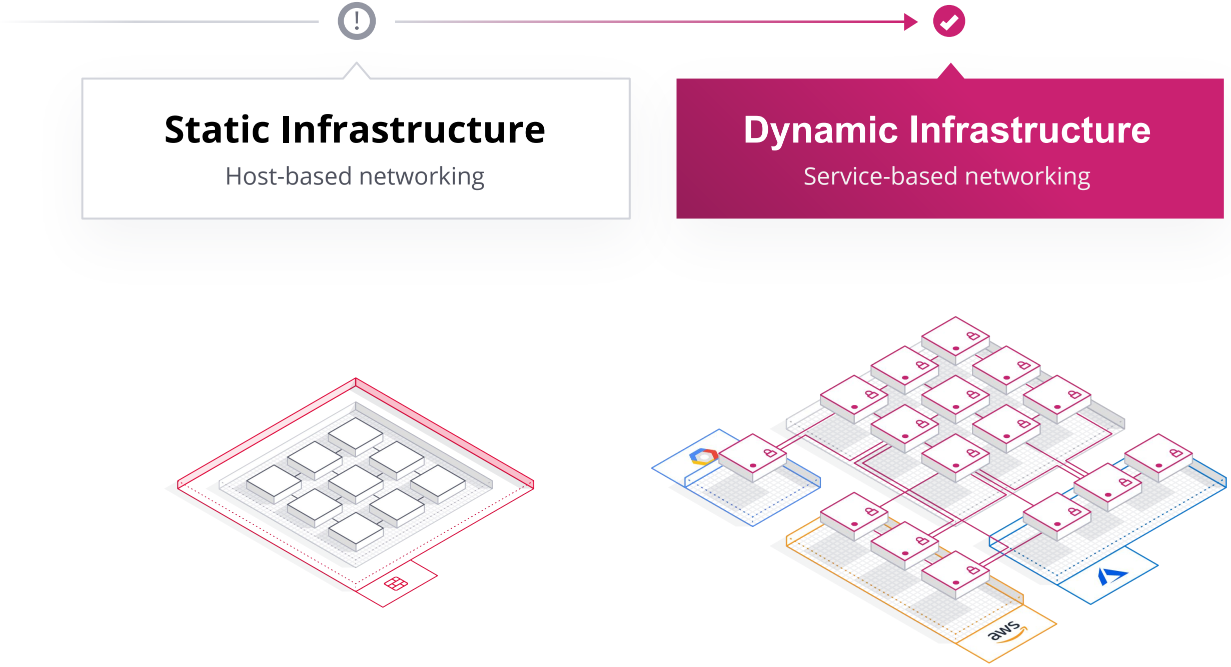
Developer Advocate at HashiCorp

Agenda

1. **Introduction** - create a common vocabulary.
2. **Metrics** - configure and use metrics.
3. **Tracing** - configure and use tracing.
4. **Logging** - collect access logs.

Introduction

The shift **from static to dynamic networking**



Static Infrastructure

Host-based networking

Dynamic Infrastructure

Service-based networking

The shift **from static to dynamic networking**

Static Infrastructure

Host-based networking

Homogeneous infrastructure with static IPs, primarily north-south traffic, protected by coarse-grained segments.

TRADITIONAL APPROACH

- Load balancers to create artificial static IPs
- Firewall sprawl to constrict service traffic
- Configuration management to deploy services

Dynamic Infrastructure

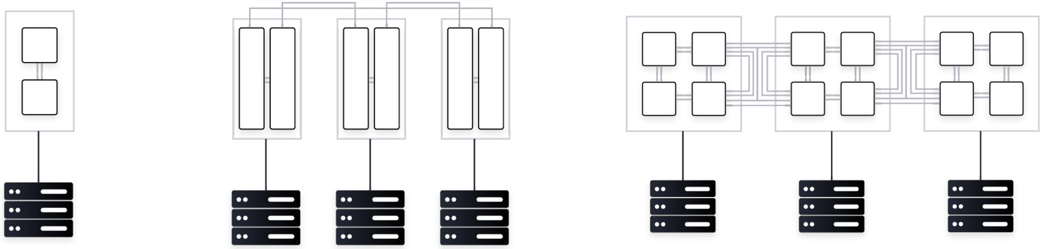
Service-based networking

Heterogeneous infrastructure with dynamic IPs, dominated by east-west traffic, without a clear network perimeter.

CONSUL APPROACH

- Service Discovery for connectivity
- Service Segmentation for security
- Service Configuration for runtime configuration

Market trend from **monoliths to microservices**



Single, Physical Server

Dynamic Virtual Machines

Smaller, Ephemeral Containers

Business challenges of dynamic infrastructure

Reduced Productivity

Waiting for manual updates to load balancers and firewalls blocks development throughput.

Increased Risk

Firewall rule sprawl is complex to manage and mistakes create security vulnerabilities.

Increased Cost

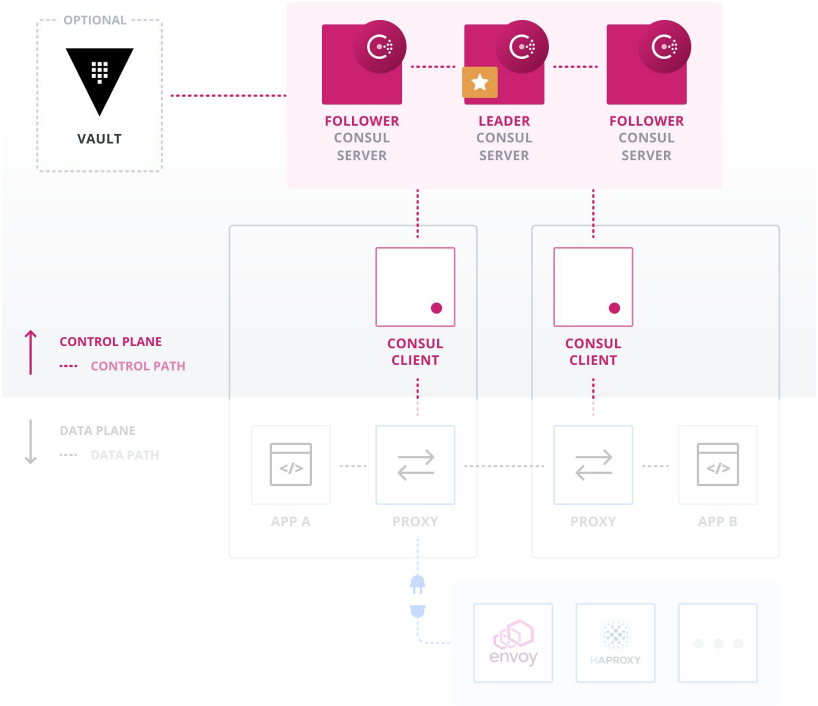
Load balancers and firewalls are expensive and costly to maintain.

Components

Service Mesh

Control Plane

- Service to service communication policy
- Service Catalog
- CA and x509 certificate generation
- Configuration and proxy management

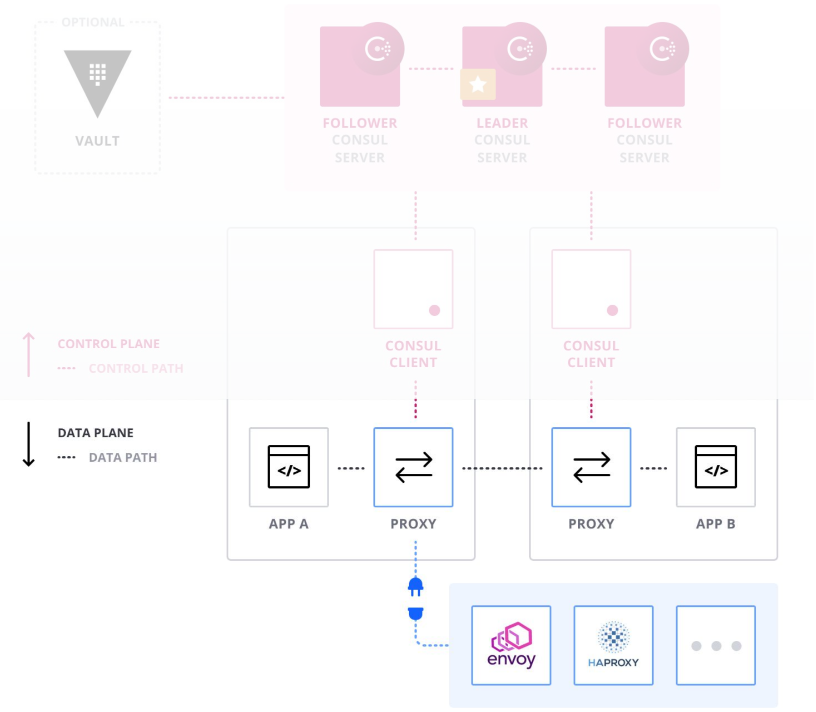


Components

Service Mesh

Data Plane

- Authorization
- Request tracing
- Traffic shaping
- Load balancing
- Service discovery
- Circuit breaking
- Retry logic
- Networking statistics



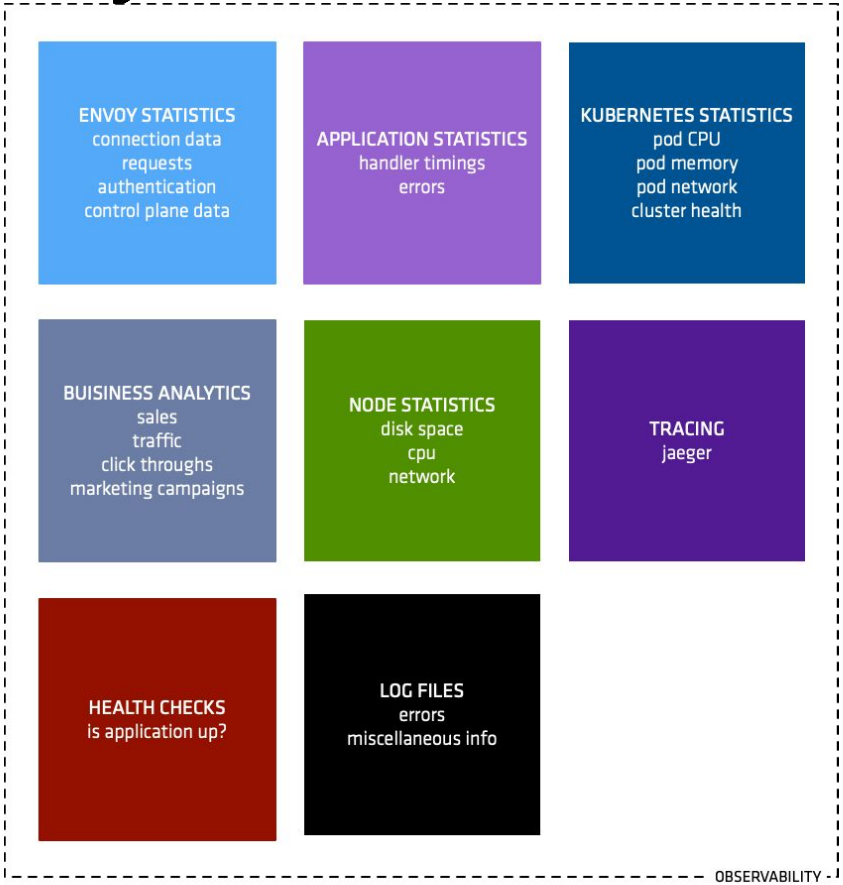
Networks are **not** 100% stable and **often** experience transient failure.

You can’t do **Reliability** without **Observability**.

Observability, is it **just a buzzword**?

Observability is a measure of how well **internal states** of a system can be **inferred from knowledge of external outputs**.

Observability



ENVOY STATISTICS
connection data
requests
authentication
control plane data

APPLICATION STATISTICS
handler timings
errors

KUBERNETES STATISTICS
pod CPU
pod memory
pod network
cluster health

BUISINESS ANALYTICS
sales
traffic
click throughs
marketing campaigns

NODE STATISTICS
disk space
cpu
network

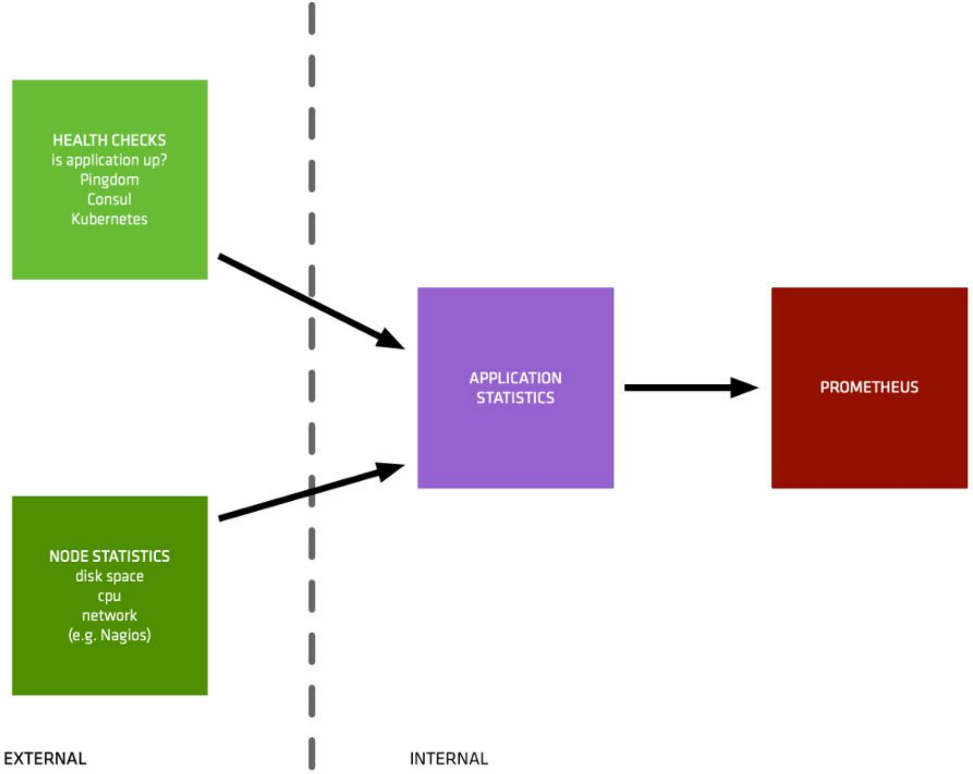
TRACING
jaeger

HEALTH CHECKS
is application up?

LOG FILES
errors
miscellaneous info

OBSERVABILITY

Internal and external instrumentation

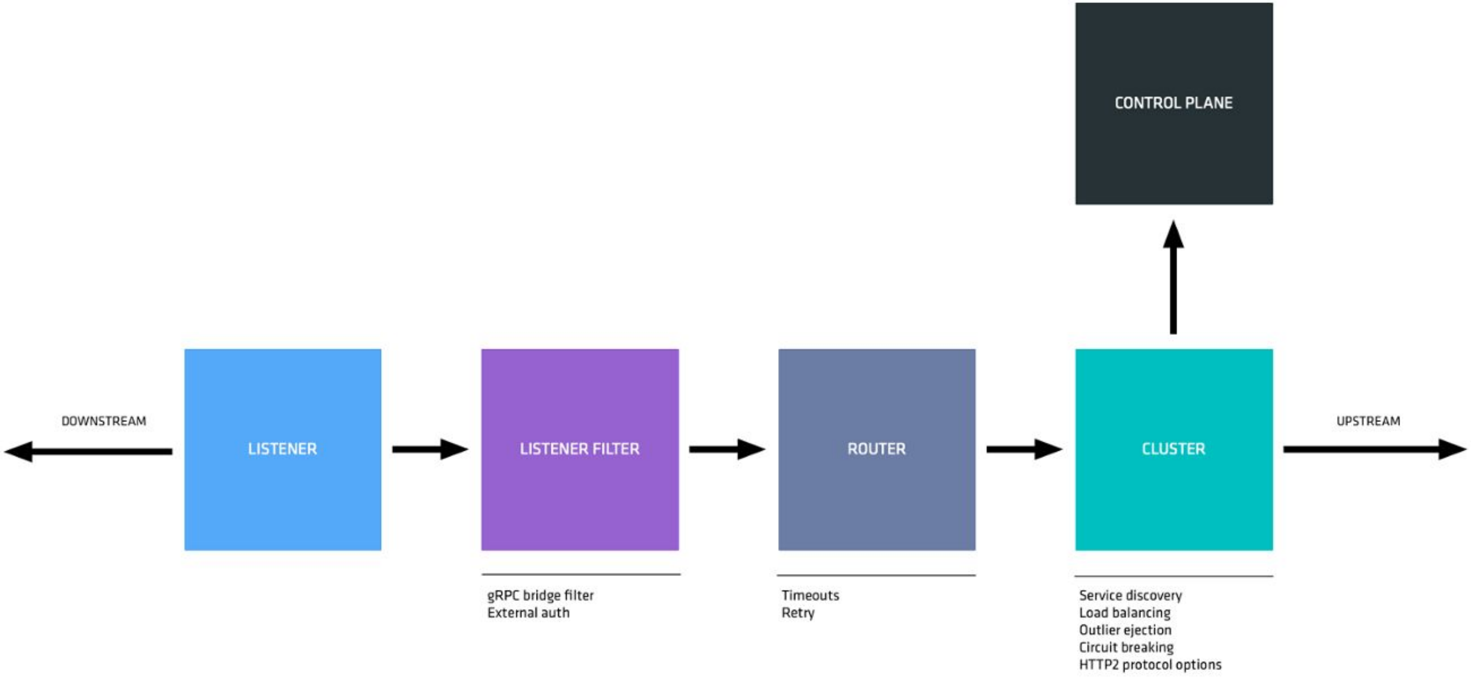


Metrics

Envoy Architecture

Metrics

Terminology



DOWNSTREAM

LISTENER

LISTENER FILTER

ROUTER

CLUSTER

UPSTREAM

CONTROL PLANE

gRPC bridge filter
External auth

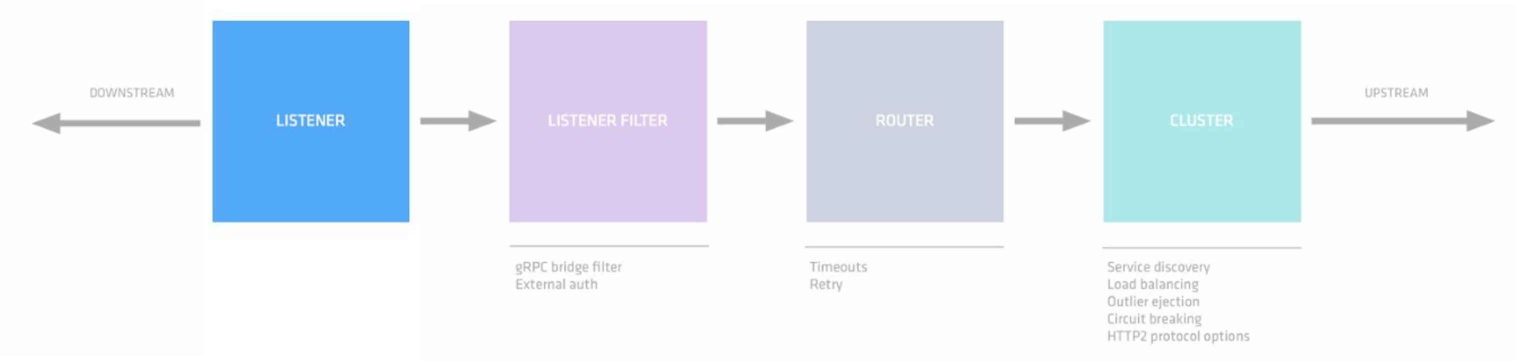
Timeouts
Retry

Service discovery
Load balancing
Outlier ejection
Circuit breaking
HTTP2 protocol options

Listener

Terminology

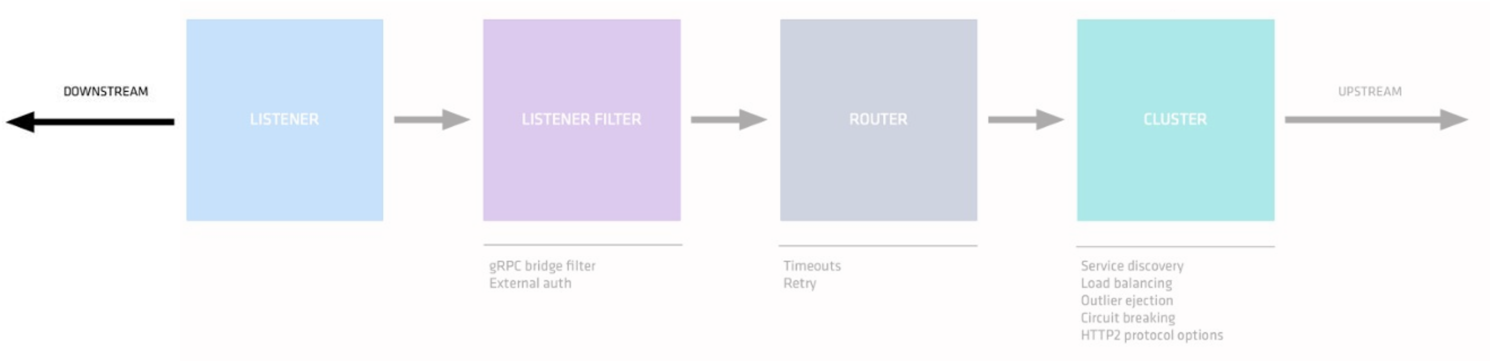
A listener is a named network location (e.g., port, unix domain socket, etc.) that can be connected to by downstream clients. Envoy exposes one or more listeners that downstream hosts connect to.



Downstream

Terminology

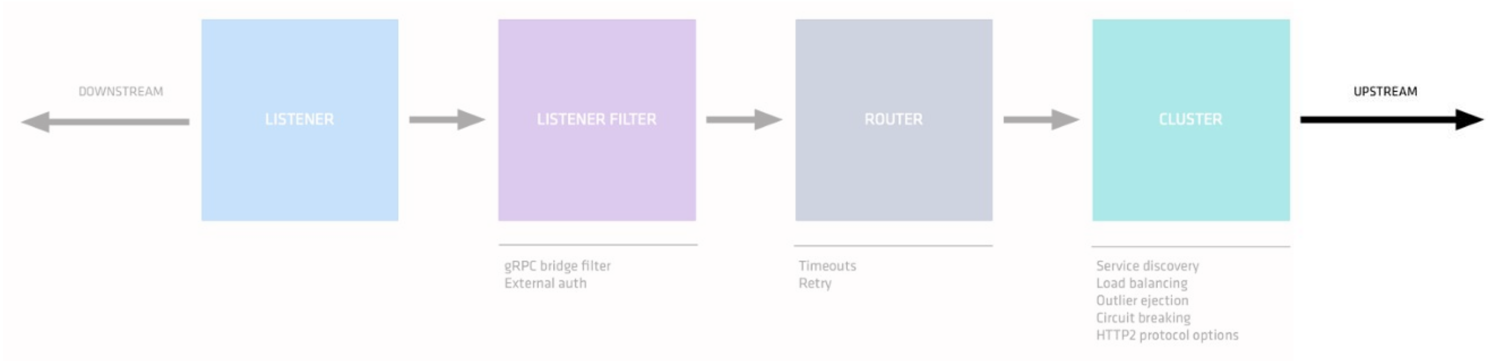
A downstream host connects to Envoy, sends requests, and receives responses.



Upstream

Terminology

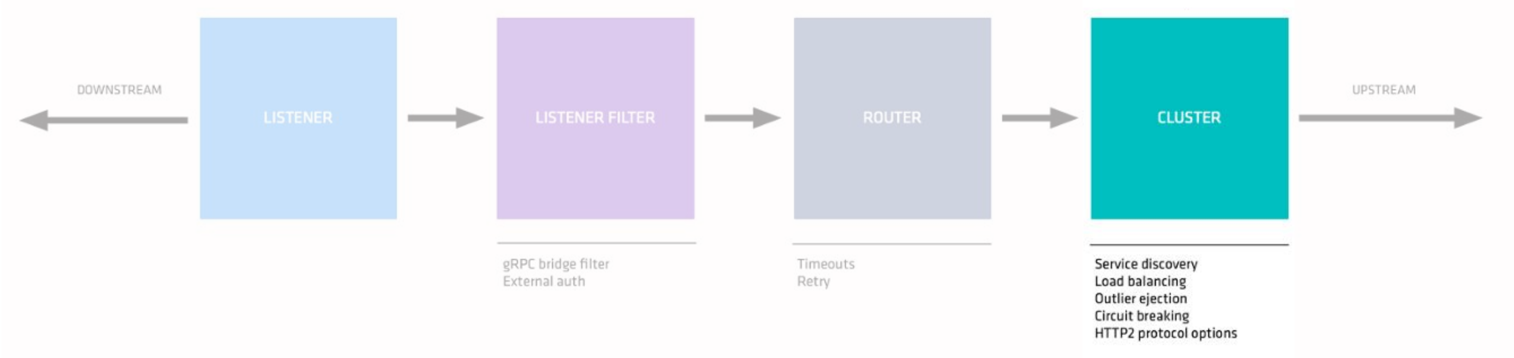
An upstream host sends requests from Envoy to other services and returns responses.



Cluster

Terminology

A cluster is a group of logically similar upstream hosts that Envoy connects to. Envoy discovers the members of a cluster via **service discovery**. The cluster member that Envoy routes a request to is determined by the **load balancing policy**.

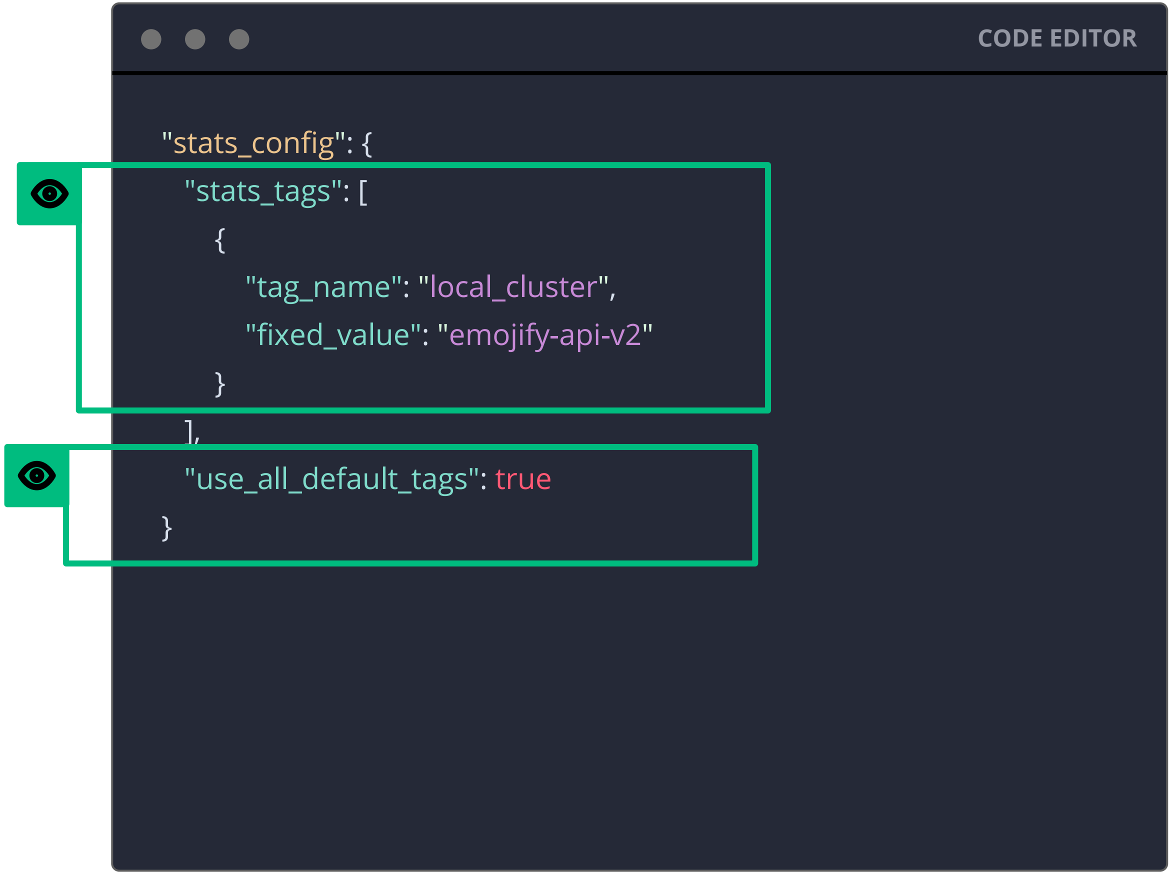


Configuration

Metrics

Envoy metrics.

Envoy does not label the metrics with the application name, so **add tags** to be able **to differentiate** between metrics.



```
"stats_config": {
  "stats_tags": [
    {
      "tag_name": "local_cluster",
      "fixed_value": "emojify-api-v2"
    }
  ],
  "use_all_default_tags": true
}
```

Envoy Prometheus Metrics

Metrics

- **1.10** introduces histograms for Prometheus metrics
- Metrics exposed with **unsecured** admin endpoint (/stats/prometheus),
- Exposure of metrics **needs** to be configured with **loopback** route to **avoid** exposing **admin endpoints**

Using metrics

Metrics

StatsD

- Originally created by Etsy
- Push based metrics
- Lightweight UDP protocol
- **No support for metadata**

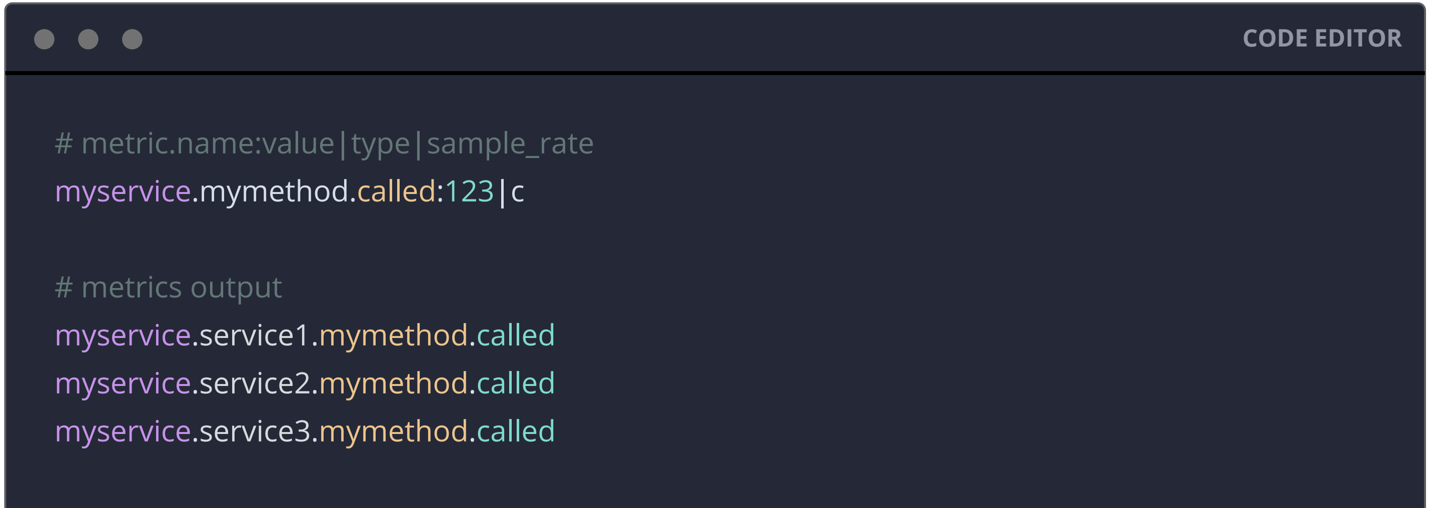
Metrics types

StatsD

Type	Description
Counter	Increment value, e.g. number of method calls.
Gauge	Value over time, e.g. CPU consumption, memory usage.
Timing	Time taken to perform a task, e.g. time take to perform a method call.
Set	Set of unique values over collection period.

Metrics format

StatsD does not support basic metric labels.



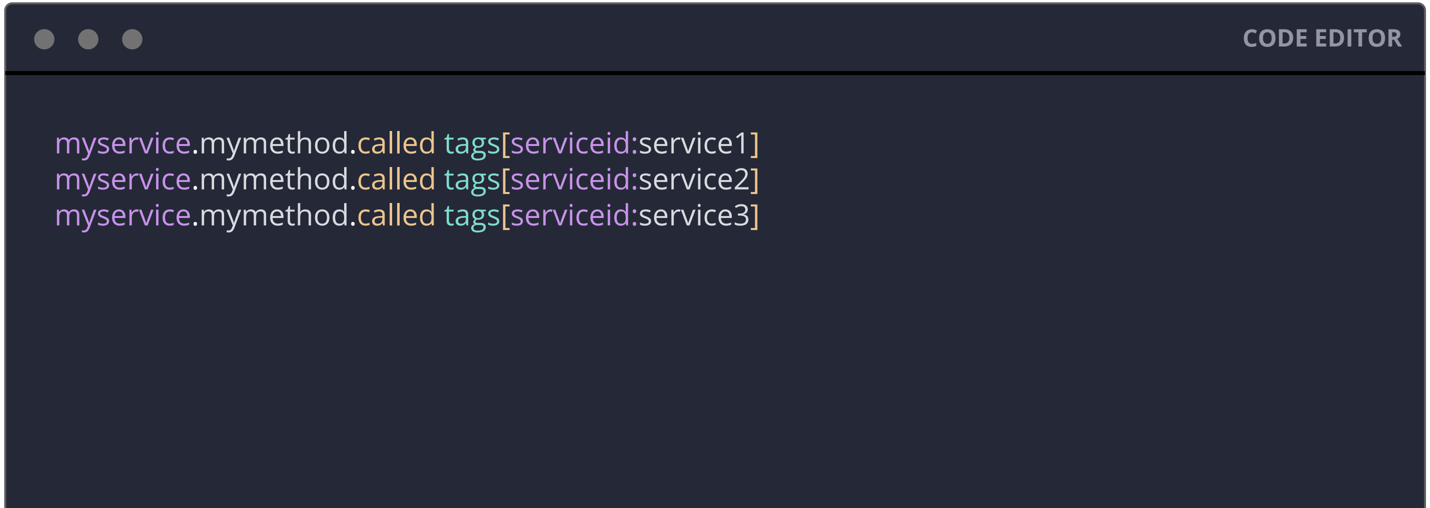
```
# metric.name:value|type|sample_rate
myservice.mymethod.called:123|c

# metrics output
myservice.service1.mymethod.called
myservice.service2.mymethod.called
myservice.service3.mymethod.called
```

DogStatsD

- Created by DataDog based on StatsD protocol
- Push based metrics
- Lightweight UDP protocol
- **Support for metadata through tags**

Metrics format

DogStatsD



```
myservice.mymethod.called tags[serviceid:service1]
myservice.mymethod.called tags[serviceid:service2]
myservice.mymethod.called tags[serviceid:service3]
```

Prometheus

- Pull based approach from central server
- Service implements HTTP endpoint exposing metrics
- **Supports metadata by default**

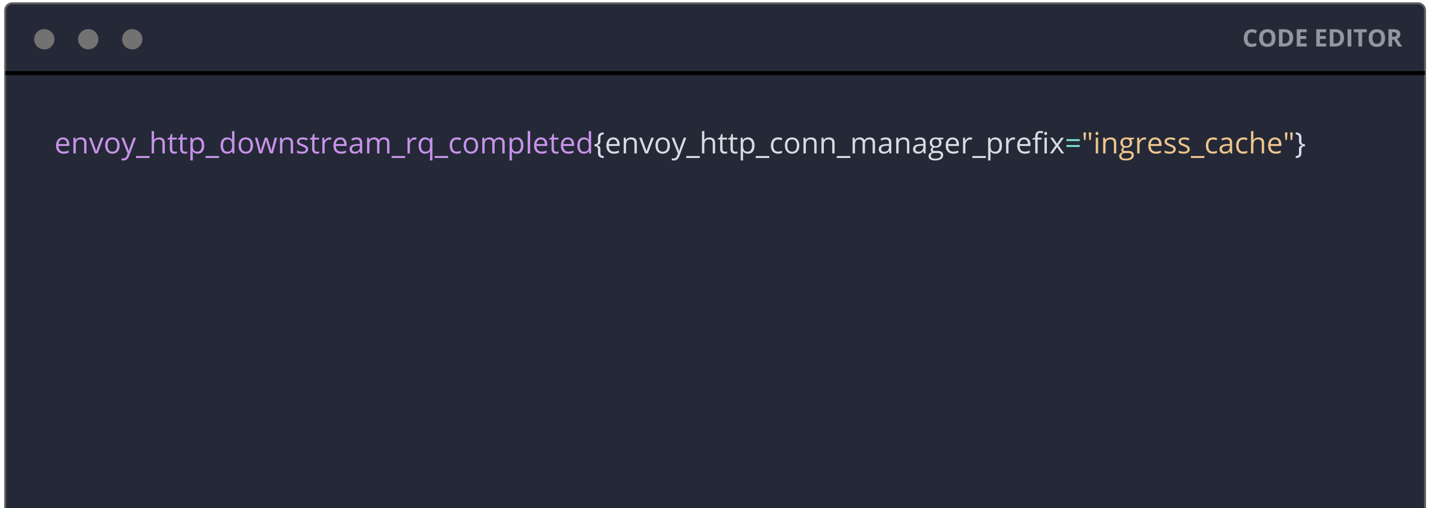
Metrics types

Prometheus

Type	Description
Counter	Cumulative metric, representing a monotonically increasing counter, e.g. number of method calls.
Gauge	Single numerical value that can arbitrarily go up and down, e.g. CPU consumption.
Histogram	Samples observations and counts them in configurable buckets, e.g. request timings.

Metrics format

Prometheus



```
envoy_http_downstream_rq_completed{envoy_http_conn_manager_prefix="ingress_cache"}
```

Choosing a format

- Tagging is **essential** to effectively build dashboards
- Metrics **need** to be tagged with **Metadata** such as pod name, node, etc

Listener

Metrics

Key service metrics

Listener - Connections

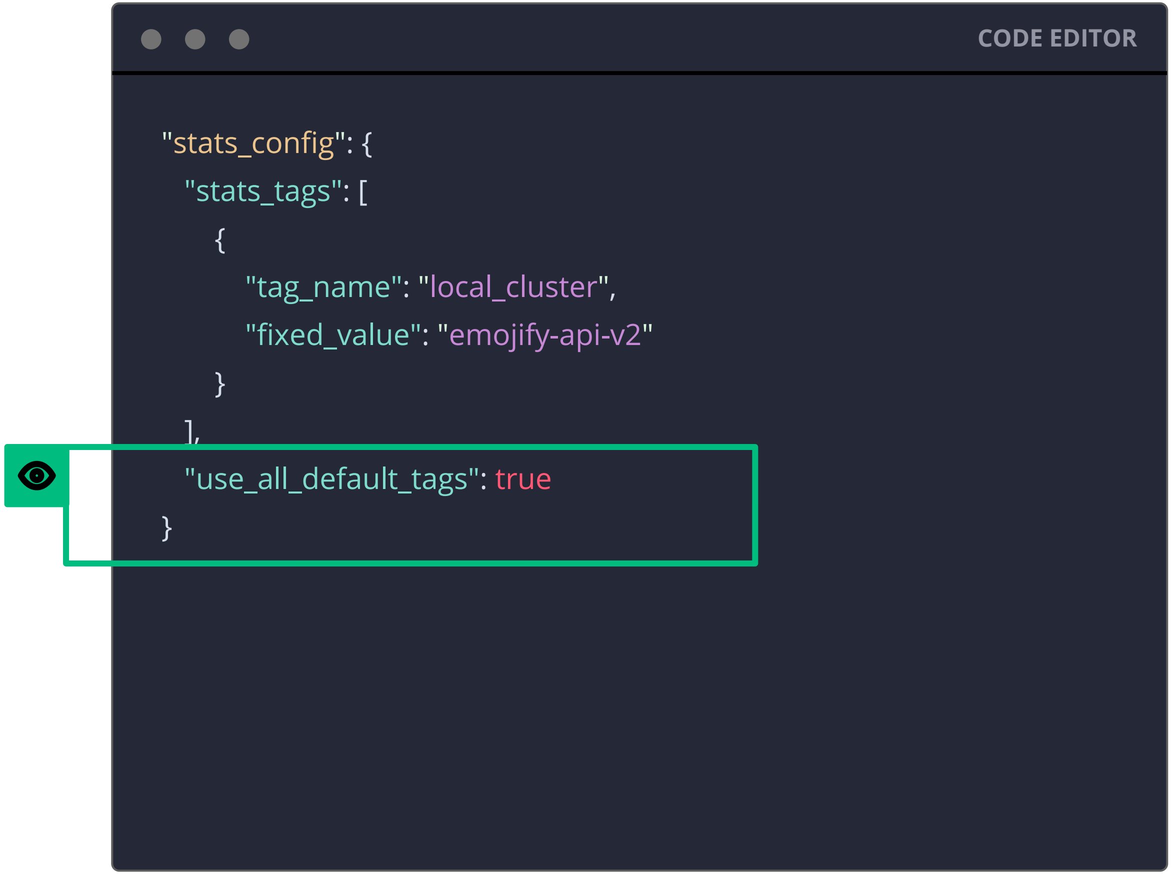
Every listener has a statistics tree rooted at <prefix>*.listener.<address>*. with the following statistics:

downstream_cx_total	Counter	Total connections
downstream_cx_destroy	Counter	Total destroyed connections
downstream_cx_active	Gauge	Total active connections

https://www.envoyproxy.io/docs/envoy/latest/configuration/listeners/stats#listener-manager

Envoy metrics.

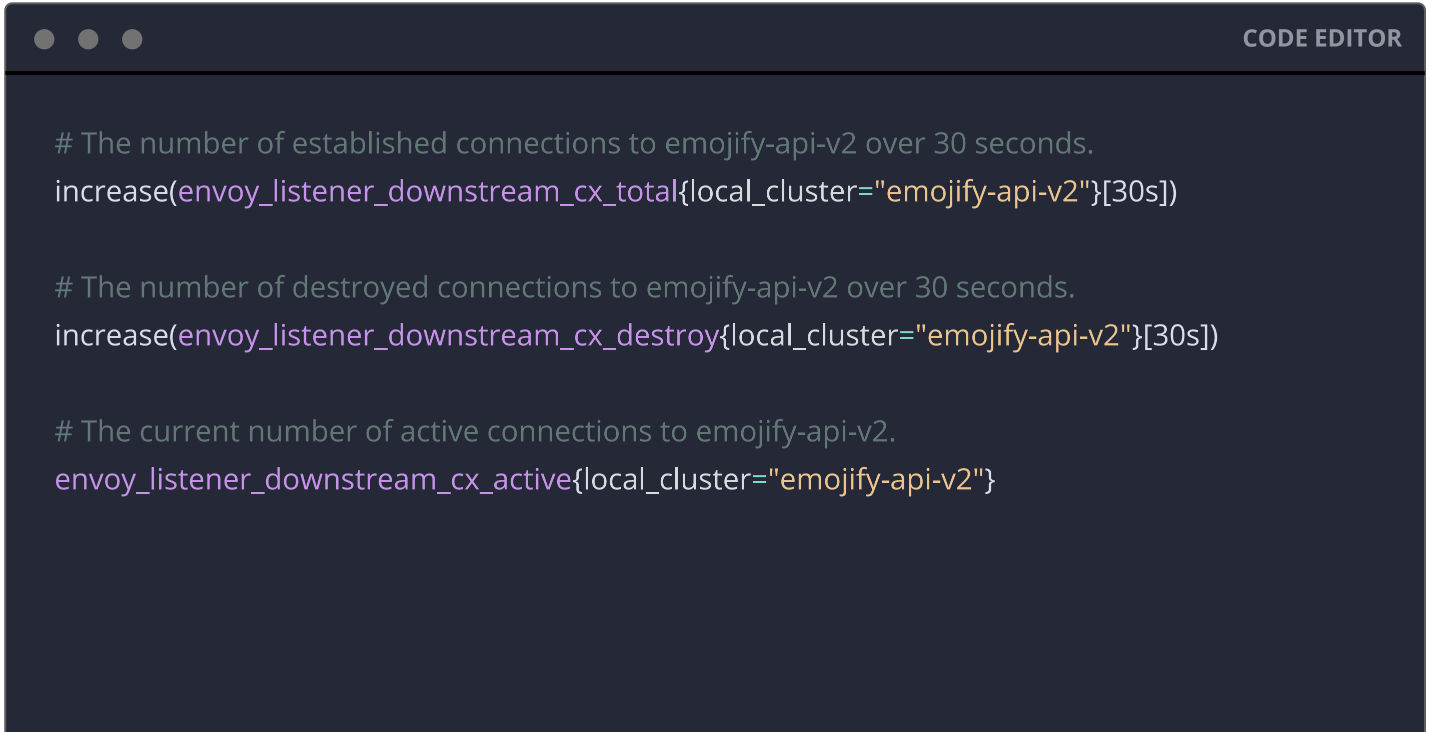
use_all_default_tags extracts **common components** from metric names and **adds as tags**



```
"stats_config": {
  "stats_tags": [
    {
      "tag_name": "local_cluster",
      "fixed_value": "emojify-api-v2"
    }
  ],
  "use_all_default_tags": true
}
```

Metrics queries

Prometheus

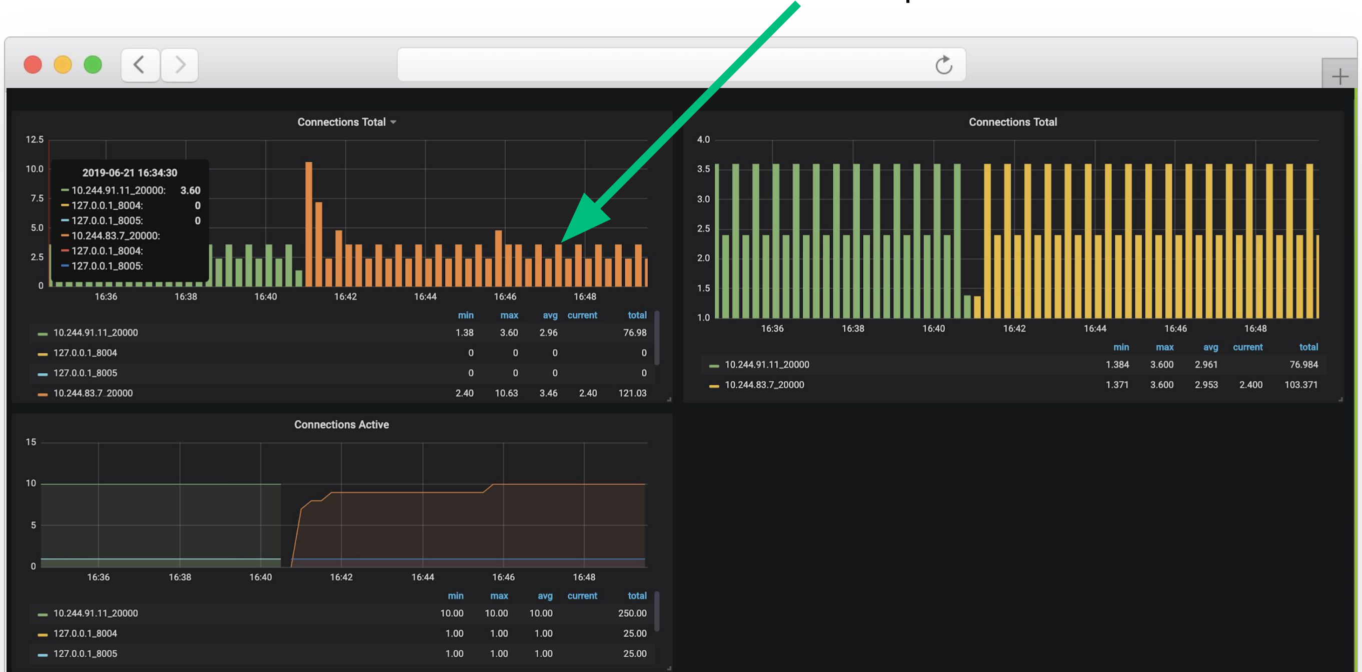


```
# The number of established connections to emojify-api-v2 over 30 seconds.
increase(envoy_listener_downstream_cx_total{local_cluster="emojify-api-v2"}[30s])

# The number of destroyed connections to emojify-api-v2 over 30 seconds.
increase(envoy_listener_downstream_cx_destroy{local_cluster="emojify-api-v2"}[30s])

# The current number of active connections to emojify-api-v2.
envoy_listener_downstream_cx_active{local_cluster="emojify-api-v2"}
```

Total connections

Grafana

New pod started



Connections Total

2019-06-21 16:34:30
- 10.244.91.11_20000: 3.60
- 127.0.0.1_8004: 0
- 127.0.0.1_8005: 0
- 10.244.83.7_20000:
- 127.0.0.1_8004:
- 127.0.0.1_8005:

	min	max	avg	current	total
10.244.91.11_20000	1.38	3.60	2.96		76.98
127.0.0.1_8004	0	0	0		0
127.0.0.1_8005	0	0	0		0
10.244.83.7_20000	2.40	10.63	3.46	2.40	121.03

Connections Active

	min	max	avg	current	total
10.244.91.11_20000	10.00	10.00	10.00		250.00
127.0.0.1_8004	1.00	1.00	1.00		25.00
127.0.0.1_8005	1.00	1.00	1.00		25.00

Connections Total

	min	max	avg	current	total
10.244.91.11_20000	1.384	3.600	2.961		76.984
10.244.83.7_20000	1.371	3.600	2.953	2.400	103.371

Key diagnostics metrics

Listener

Every listener has a statistics tree rooted at <prefix>*.listener.<address>*. with the following statistics:

ssl.fail_verify_no_cert	Counter	Total TLS connections that failed because of missing client certificate
ssl.connection_error	Counter	Total TLS connection errors not including failed certificate verifications
ssl.fail_verify_error	Counter	Total TLS connections that failed CA verification
ssl.fail_verify_san	Counter	Total TLS connections that failed SAN verification
downstream_pre_cx_timeout	Counter	Sockets that timed out during listener filter processing
downstream_pre_cx_active	Gauge	Sockets currently undergoing listener filter processing
downstream_cx_length_ms	Histogram	Connection length milliseconds

https://www.envoyproxy.io/docs/envoy/latest/configuration/listeners/stats#listener-manager

Requests HTTP / GRPC

Metrics

Key metrics

Listener - Requests HTTP

Every listener has a statistics tree rooted at <prefix>*.http.<address>*. with the following statistics:

downstream_rq_1xx	Counter	Total 1xx responses
downstream_rq_2xx	Counter	Total 2xx responses
downstream_rq_3xx	Counter	Total 3xx responses
downstream_rq_4xx	Counter	Total 4xx responses
downstream_rq_5xx	Counter	Total 5xx responses
downstream_rq_ws_on_non_ws_route	Counter	Total WebSocket upgrade requests rejected by non WebSocket routes
downstream_rq_time	Histogram	Total time for request and response (milliseconds)
downstream_rq_timeout	Counter	Total requests closed due to a timeout on the request path

https://www.envoyproxy.io/docs/envoy/latest/configuration/http_conn_man/stats

Key metrics

Listener - Requests HTTP

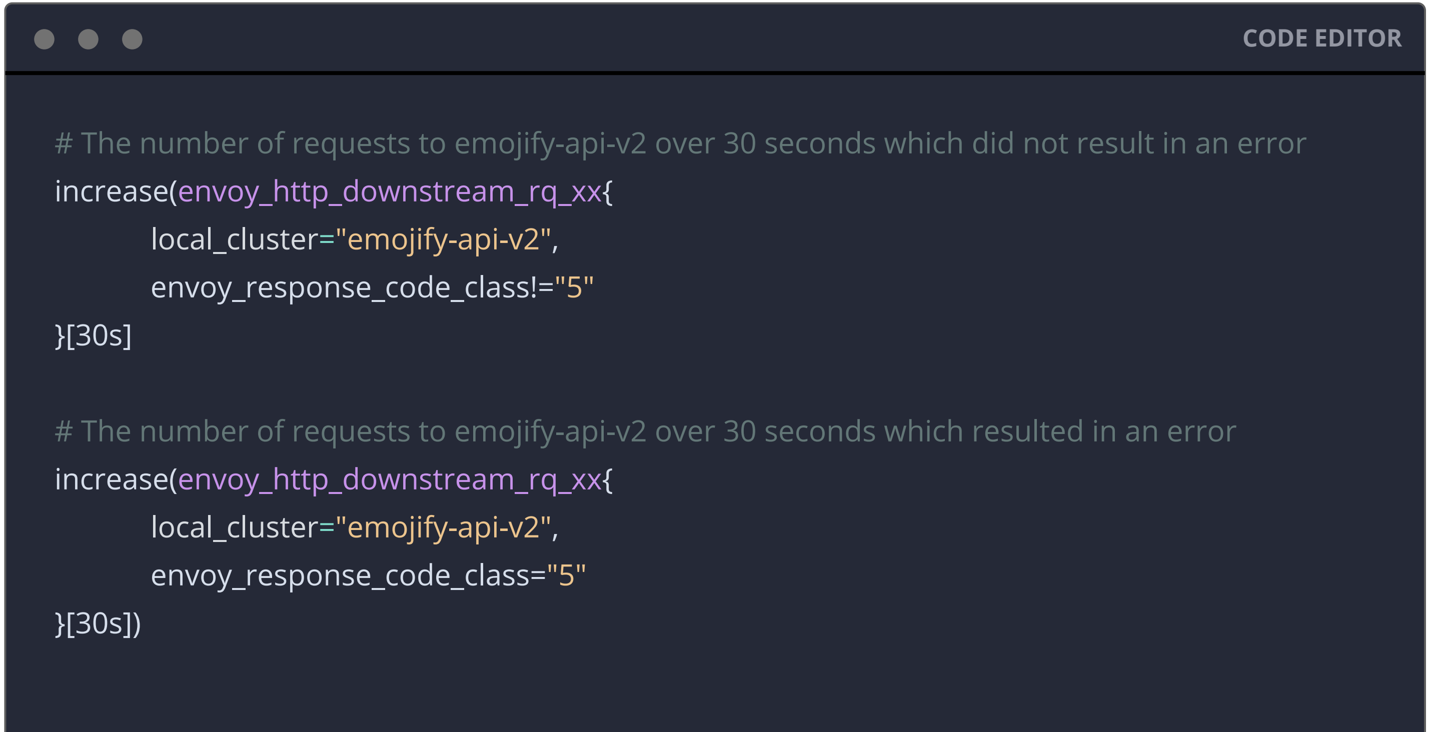
Every listener has a statistics tree rooted at <prefix>*.http.<address>*. with the following statistics:

downstream_rq_total	Counter	Total requests
downstream_rq_http1_total	Counter	Total HTTP/1.1 requests
downstream_rq_http2_total	Counter	Total HTTP/2 requests
downstream_rq_too_large	Counter	Total requests resulting in a 413 due to buffering an overly large body
downstream_rq_completed	Counter	Total requests that resulted in a response (e.g. does not include aborted requests)

https://www.envoyproxy.io/docs/envoy/latest/configuration/http_conn_man/stats

Metrics queries

Prometheus

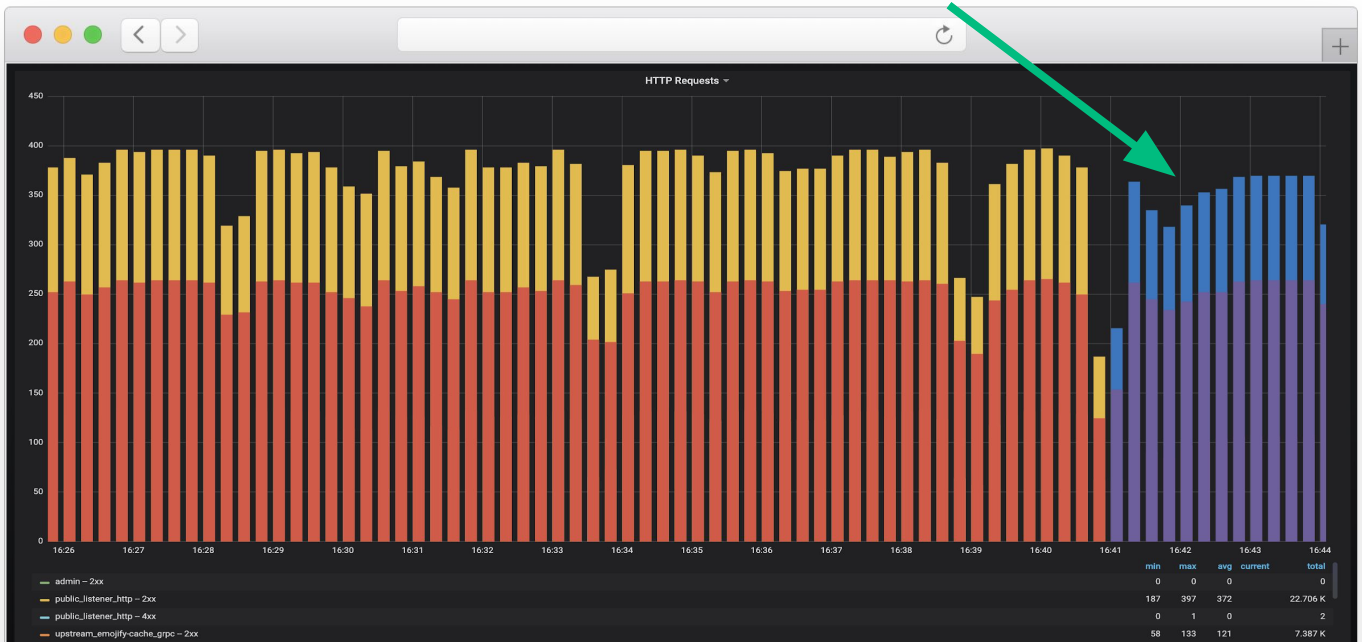


```
# The number of requests to emojify-api-v2 over 30 seconds which did not result in an error
increase(envoy_http_downstream_rq_xx{
          local_cluster="emojify-api-v2",
          envoy_response_code_class!="5"
}[30s]

# The number of requests to emojify-api-v2 over 30 seconds which resulted in an error
increase(envoy_http_downstream_rq_xx{
          local_cluster="emojify-api-v2",
          envoy_response_code_class="5"
}[30s])
```

Total Requests - all listeners for a proxy

Grafana

New pod started

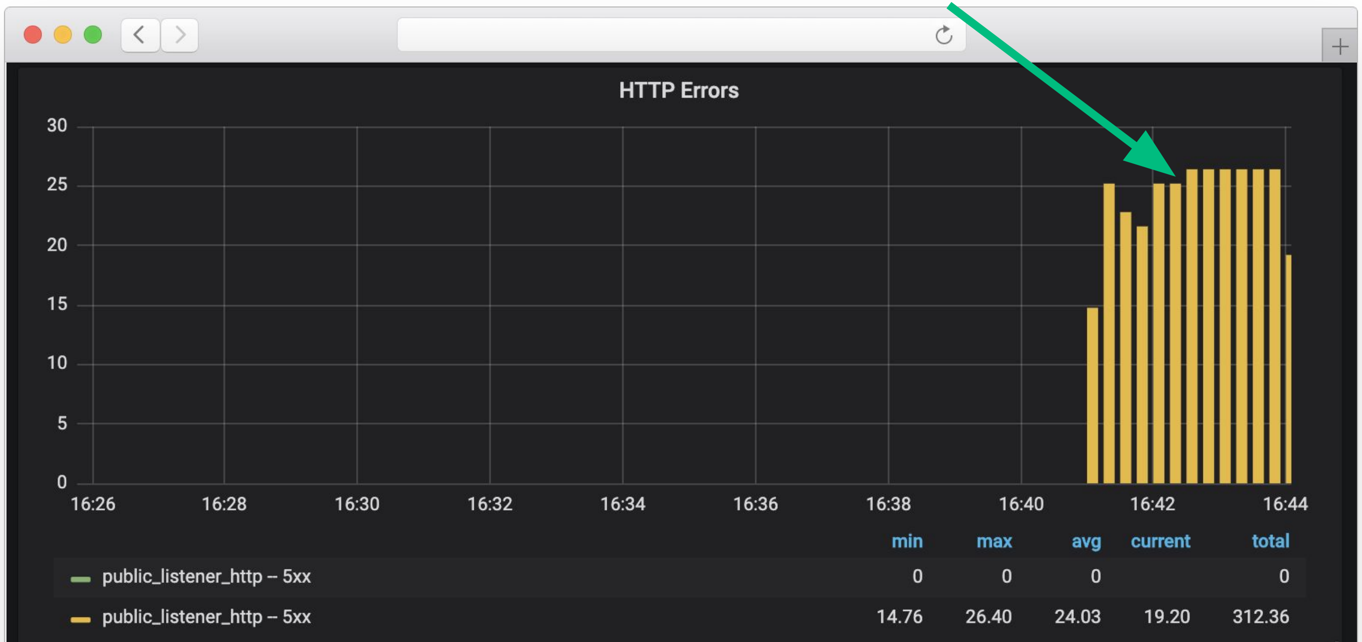


	min	max	avg	current	total
admin – 2xx	0	0	0		0
public_listener_http – 2xx	187	397	372		22.706 K
public_listener_http – 4xx	0	1	0		2
upstream_emojify-cache_grpc – 2xx	58	133	121		7.387 K

Request Errors

Grafana

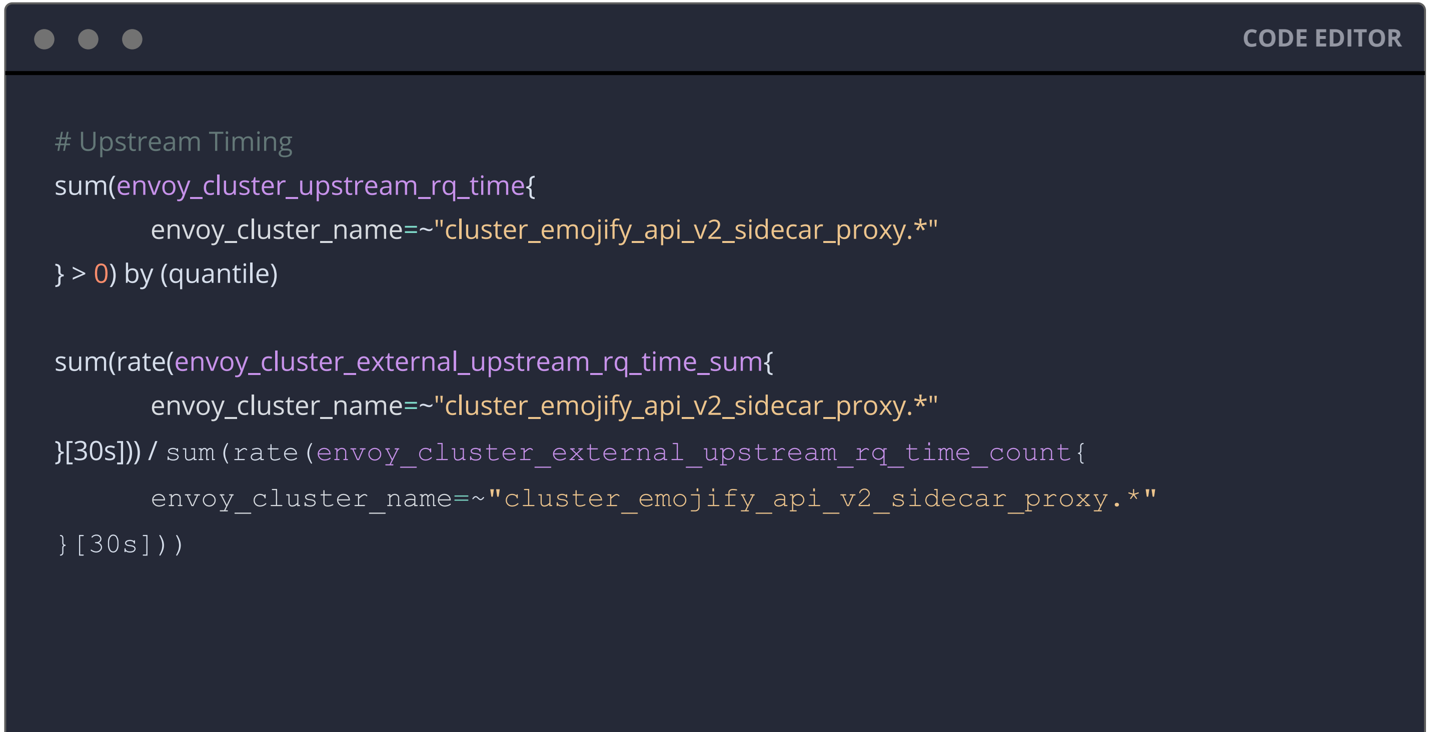
New pod started



	min	max	avg	current	total
🟩 public_listener_http – 5xx	0	0	0		0
🟨 public_listener_http – 5xx	14.76	26.40	24.03	19.20	312.36

Metrics queries - Timing

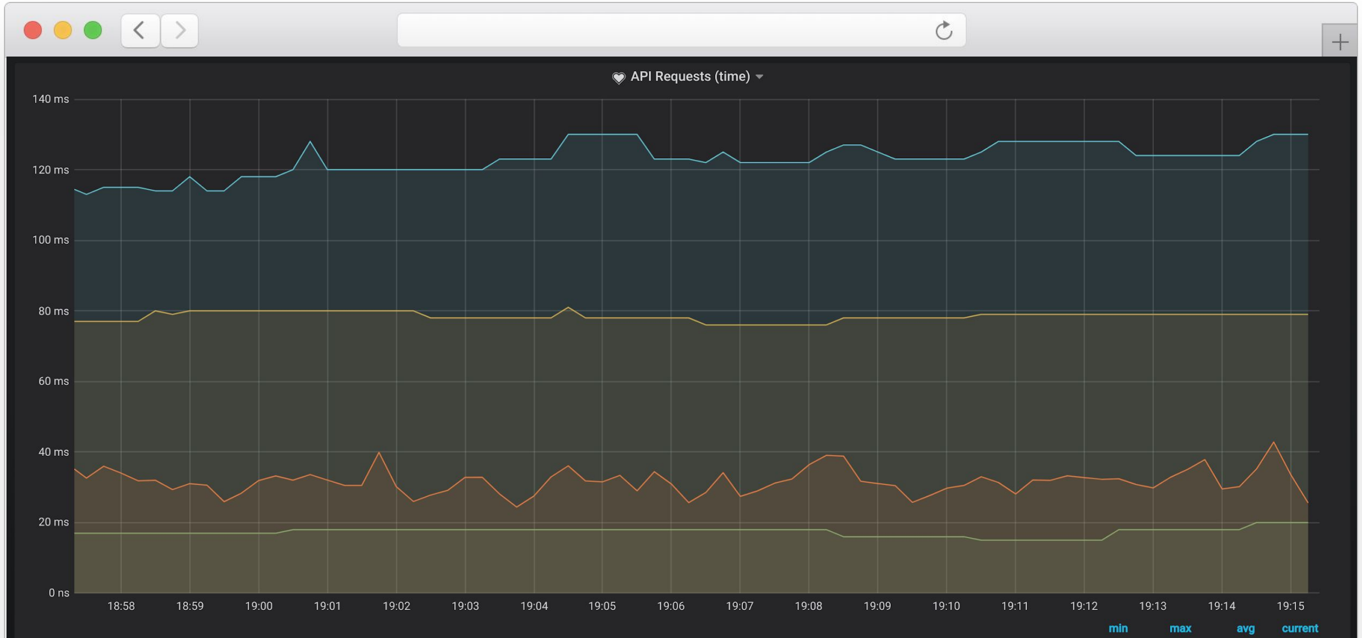
Prometheus



```
# Upstream Timing
sum(envoy_cluster_upstream_rq_time{
          envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"
} > 0) by (quantile)

sum(rate(envoy_cluster_external_upstream_rq_time_sum{
          envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"
}[30s])) / sum(rate(envoy_cluster_external_upstream_rq_time_count{
          envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"
}[30s]))
```

Request Time

Grafana



Key metrics

Listener - Requests gRPC

The filter emits statistics in the cluster.<route target cluster>.grpc. namespace

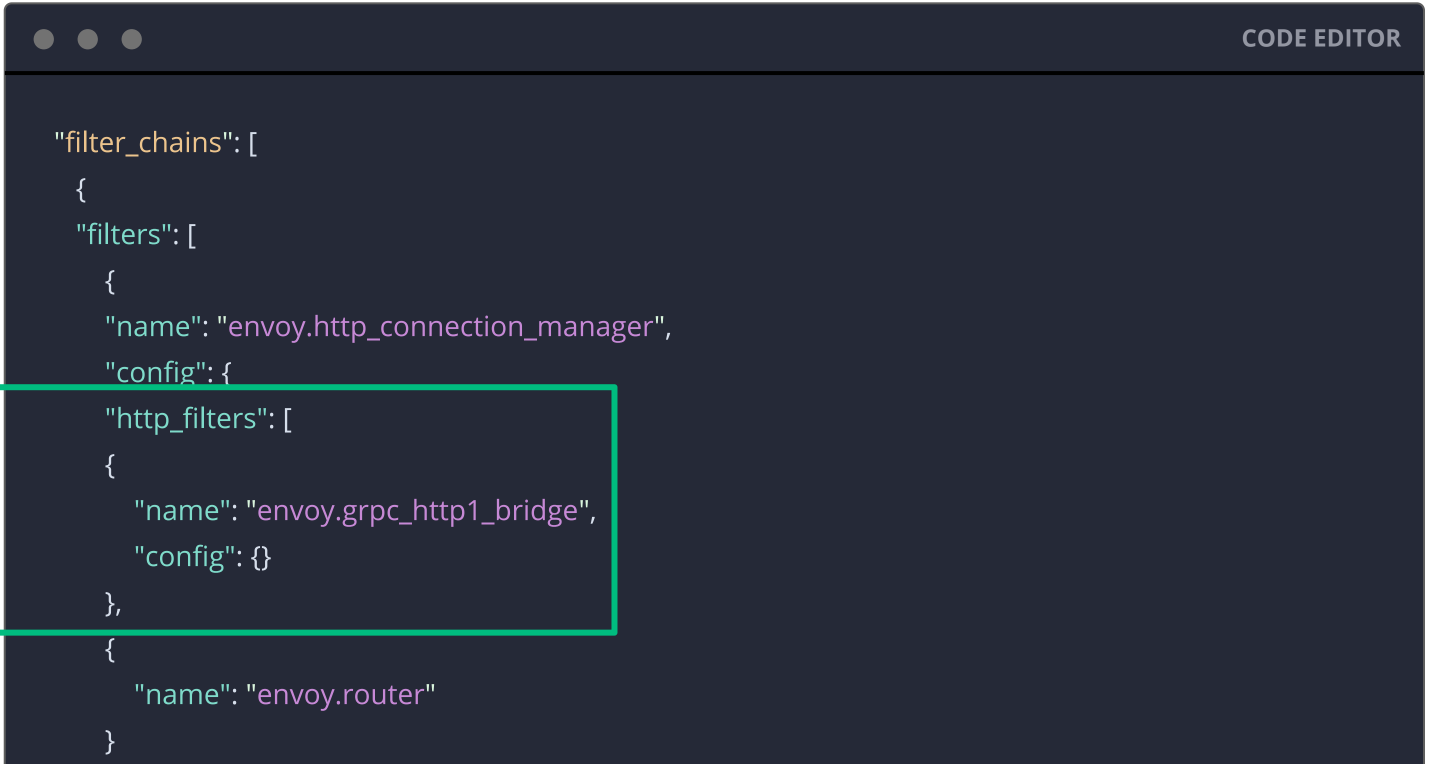
<grpc service>.<grpc method>.success	Counter	Total successful service/method calls
<grpc service>.<grpc method>.failure	Counter	Total failed service/method calls
<grpc service>.<grpc method>.total	Counter	Total service/method calls

- GRPC does **not** use HTTP status codes
- Status Codes are part of the **Protocol** and are reported as **individual** metrics

https://www.envoyproxy.io/docs/envoy/latest/configuration/listeners/stats#listener-manager

gRPC Bridge Filter

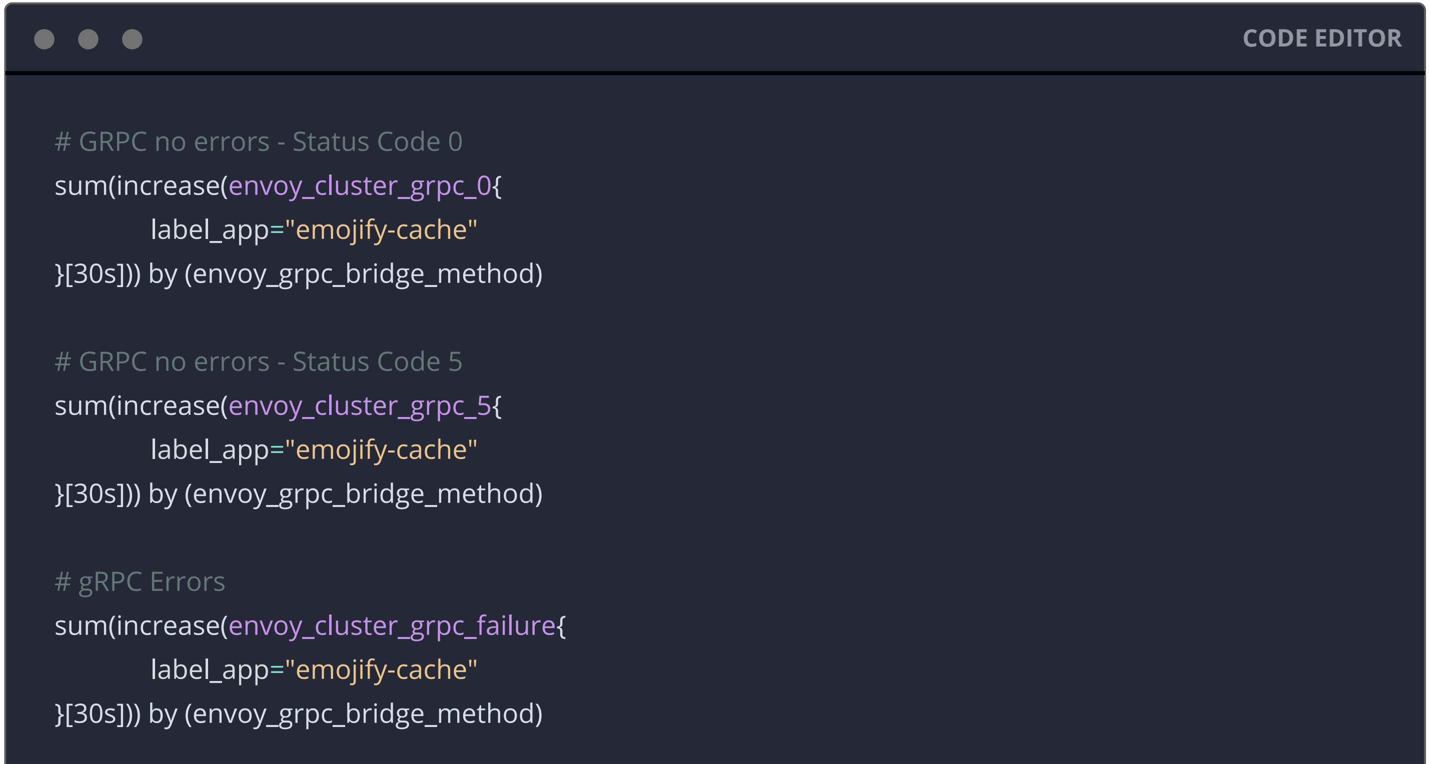
Configuration



```
"filter_chains": [
 {
 "filters": [
   {
   "name": "envoy.http_connection_manager",
   "config": {
   "http_filters": [
   {
     "name": "envoy.grpc_http1_bridge",
     "config": {}
   },
   {
     "name": "envoy.router"
   }
```

Metrics queries

Prometheus

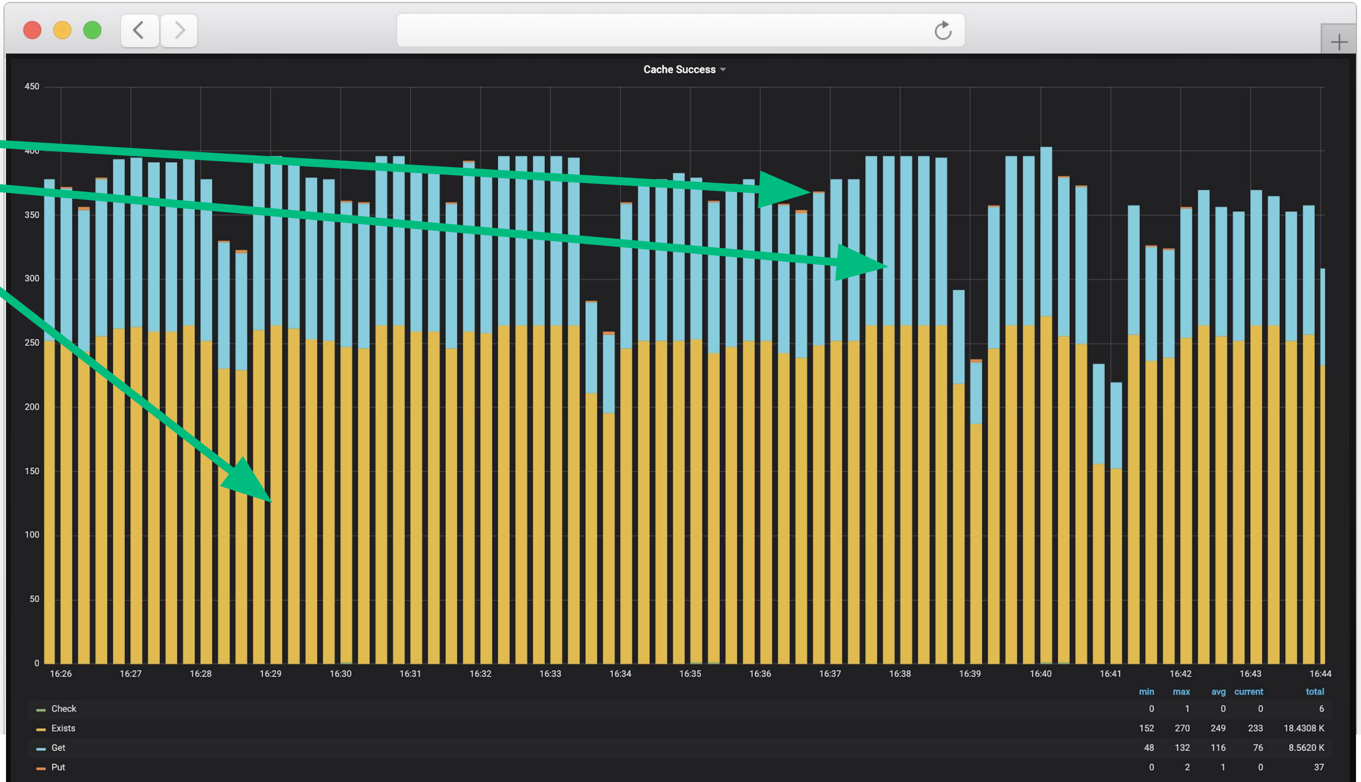


```
# GRPC no errors - Status Code 0
sum(increase(envoy_cluster_grpc_0{
        label_app="emojify-cache"
}[30s])) by (envoy_grpc_bridge_method)

# GRPC no errors - Status Code 5
sum(increase(envoy_cluster_grpc_5{
        label_app="emojify-cache"
}[30s])) by (envoy_grpc_bridge_method)

# gRPC Errors
sum(increase(envoy_cluster_grpc_failure{
        label_app="emojify-cache"
}[30s])) by (envoy_grpc_bridge_method)
```

gRPC - Success

Grafana

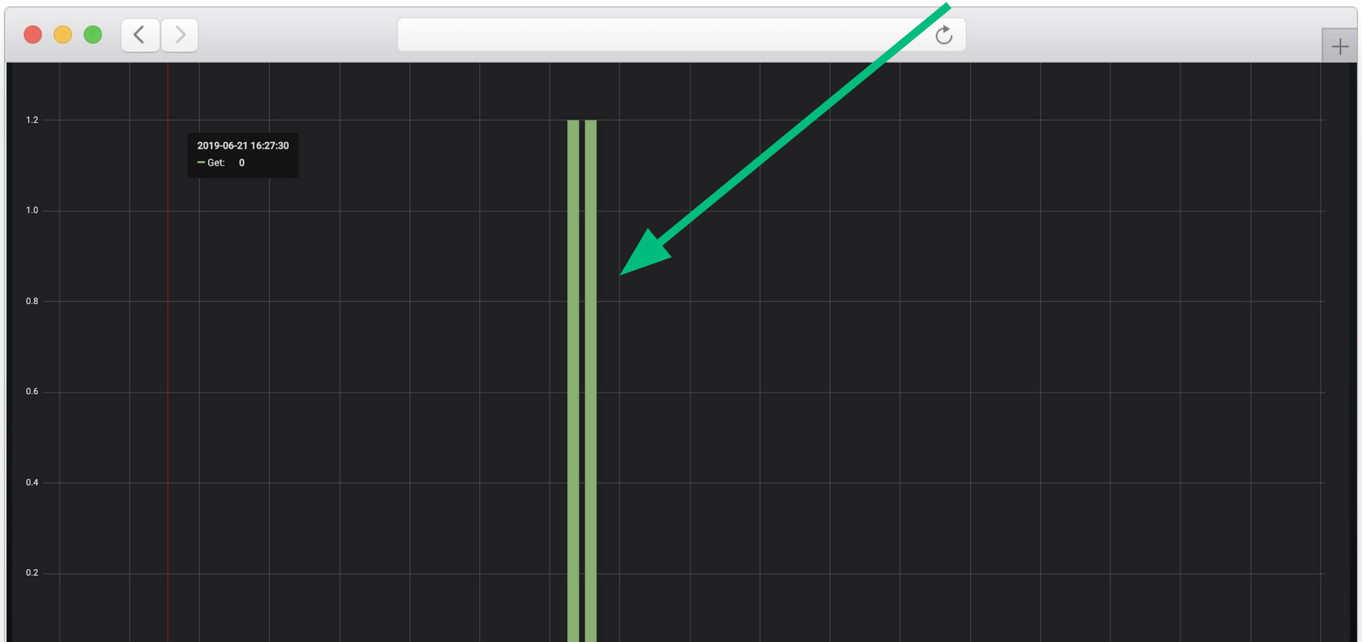
Methods:
Put
Get
Exists



	min	max	avg	current	total
Check	0	1	0	0	6
Exists	152	270	249	233	18.4308 K
Get	48	132	116	76	8.5620 K
Put	0	2	1	0	37

gRPC Error

Grafana



HTTP Response 5xx

Clusters

Metrics

Key metrics

Cluster

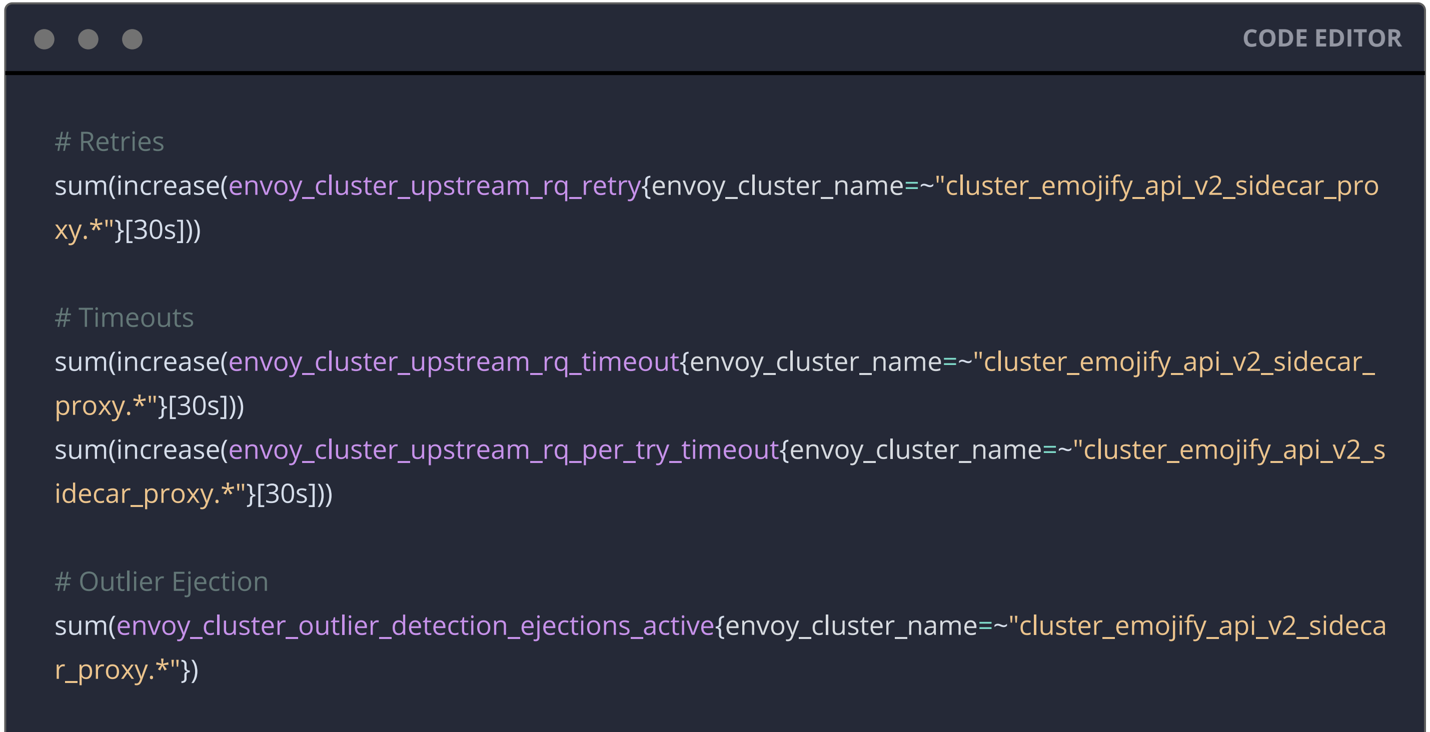
Every listener has a statistics tree rooted at <prefix>*.http.<address>*. with the following statistics:

upstream_rq_timeout	Counter	Total requests that timed out waiting for a response
upstream_rq_per_try_timeout	Counter	Total requests that hit the per try timeout
upstream_rq_retry	Counter	Total request retries
upstream_rq_retry_success	Counter	Total request retry successes
ejections_active	Counter	Number of currently ejected hosts

https://www.envoyproxy.io/docs/envoy/latest/configuration/listeners/stats#listener-manager

Metrics queries

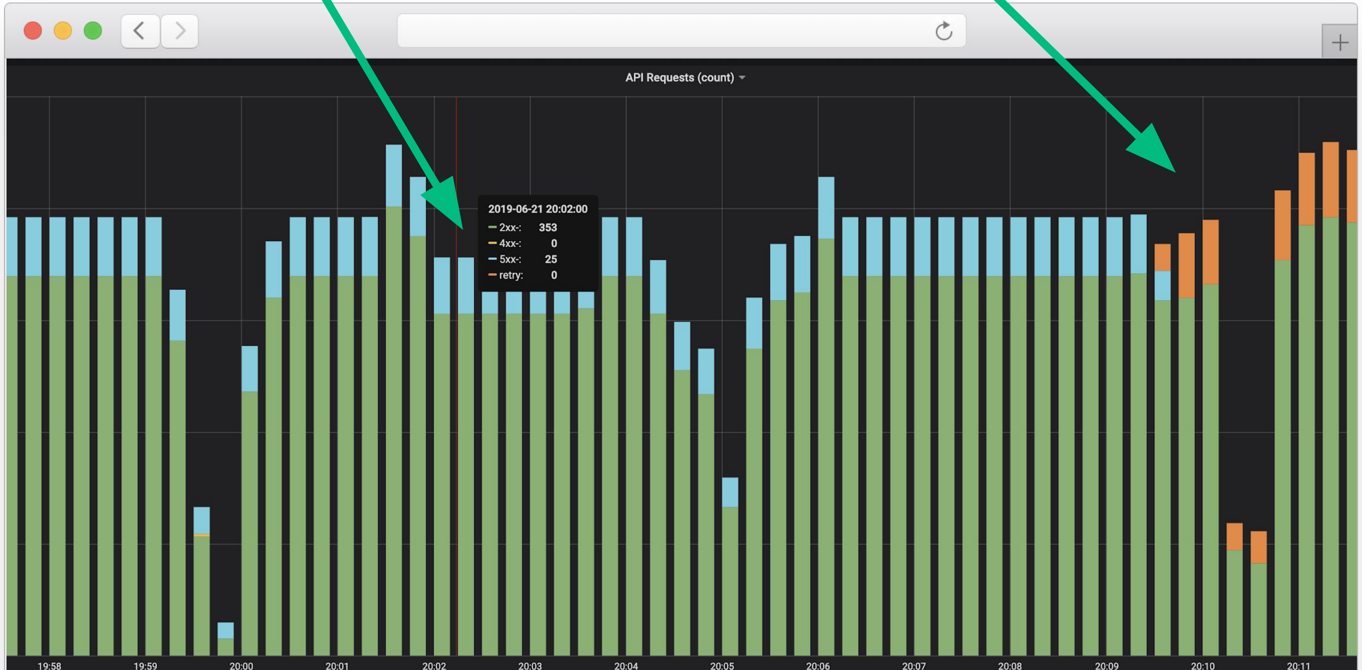
Prometheus



```
# Retries
sum(increase(envoy_cluster_upstream_rq_retry{envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"}[30s]))

# Timeouts
sum(increase(envoy_cluster_upstream_rq_timeout{envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"}[30s]))
sum(increase(envoy_cluster_upstream_rq_per_try_timeout{envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"}[30s]))

# Outlier Ejection
sum(envoy_cluster_outlier_detection_ejections_active{envoy_cluster_name=~"cluster_emojify_api_v2_sidecar_proxy.*"})
```

Retries

Grafana

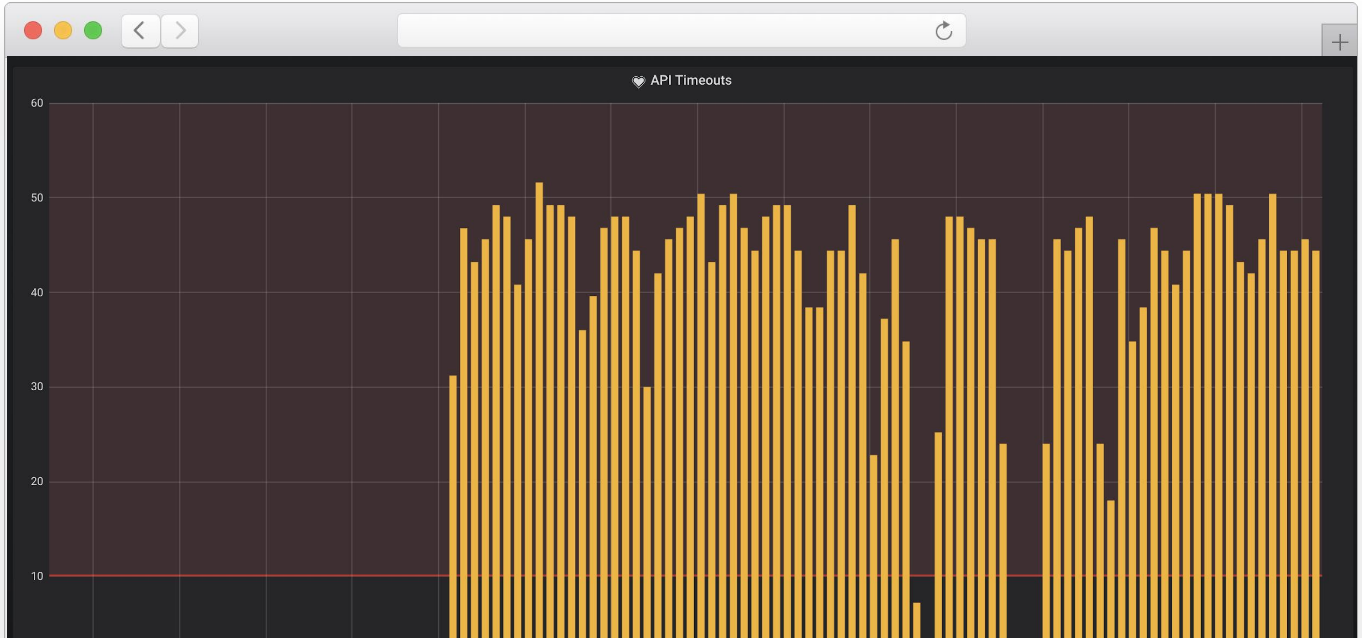


Service Errors

Retry Applied: No errors to user

Timeouts

Grafana



Outlier Ejection

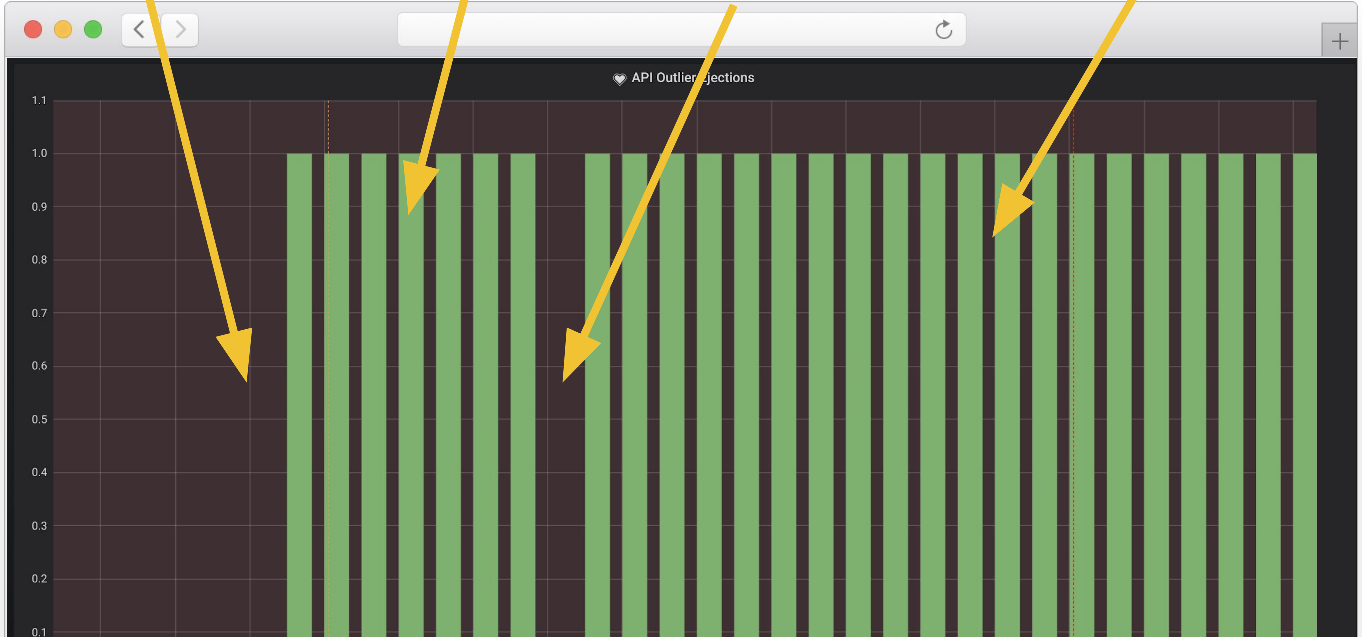
Grafana

New pod started
Constant errors

After a fixed number of
consecutive errors endpoint
removed from cluster

Envoy retries failing
endpoint

Ejection interval
increases



AuthZ

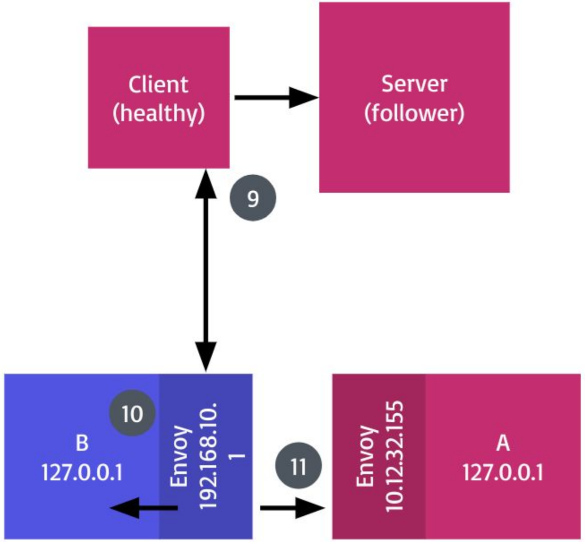
Metrics

Control plane

AuthZ

External Authorization

1. Envoy validates that the connections is allowed by calling the ext_authz filters api (once per new connection).
2. If allowed the request is passed to the upstream service.
3. Send the response to the caller.



Control plane

AuthZ

- External authorization API is normally called when establishing a new connection to an upstream.
- Failed authorization is an indication of a failing control plane, misconfiguration of security policy, or malicious activity.

Key AuthZ metrics

AuthZ

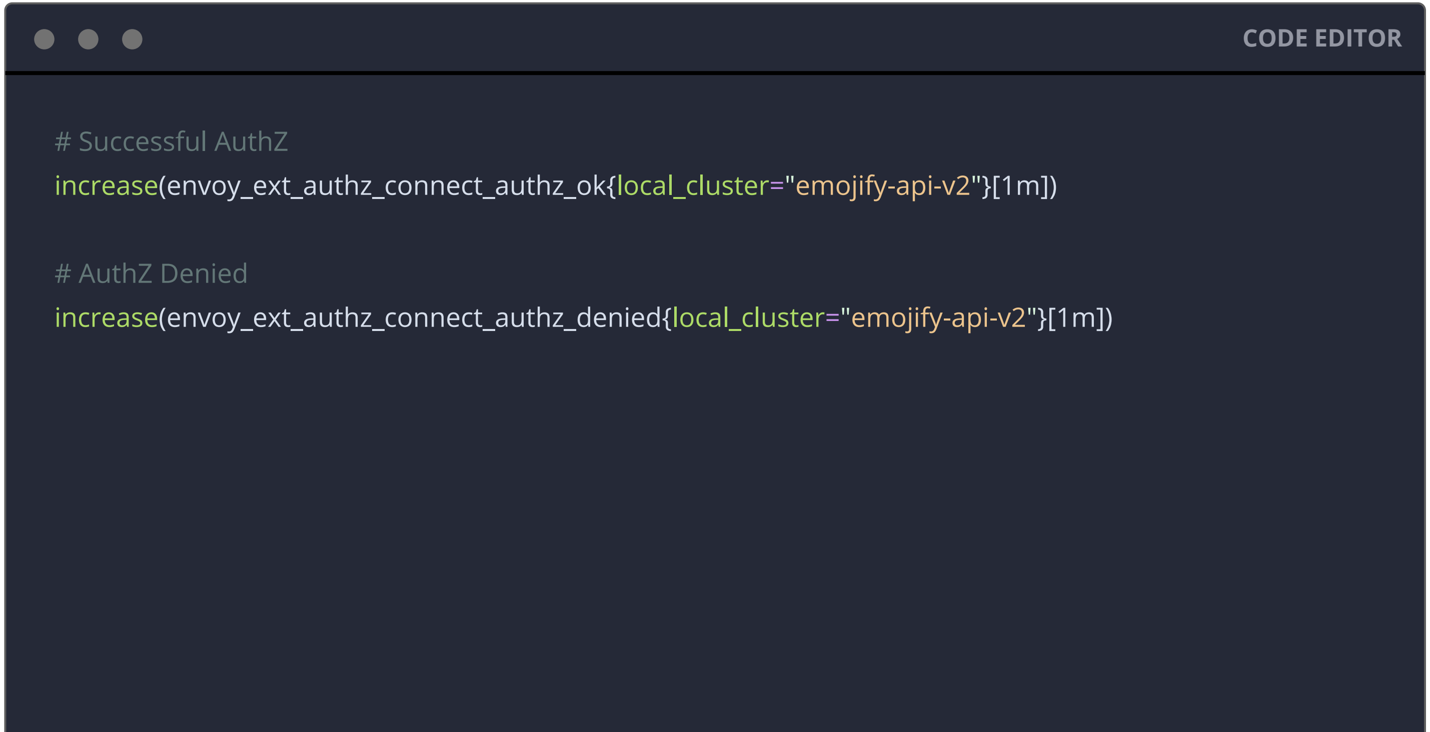
The network filter outputs statistics in the config.ext_authz. namespace, with the following statistics:

total	Counter	Total responses from the filter.
error	Counter	Total errors contacting the external service.
denied	Counter	Total responses from the authorizations service that were to deny the traffic.
failure_mode_allowed	Counter	Total requests that were error(s) but were allowed through because of failure_mode_allow set to true.
ok	Counter	Total responses from the authorization service that were to allow the traffic.
cx_closed	Counter	Total connections that were closed.
active	Gauge	Total currently active requests in transit to the authorization service.

https://www.envoyproxy.io/docs/envoy/latest/configuration/network_filters/ext_authz_filter.html

Metrics queries

Prometheus



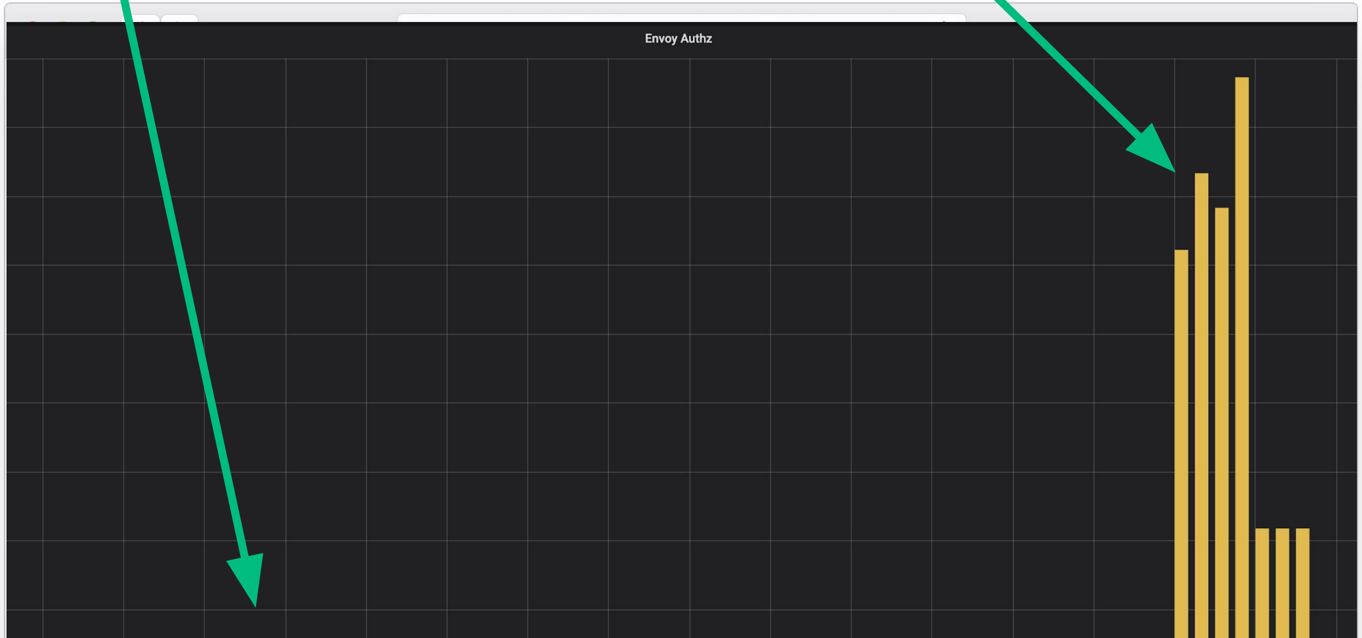
```
# Successful AuthZ
increase(envoy_ext_authz_connect_authz_ok{local_cluster="emojify-api-v2"}[1m])

# AuthZ Denied
increase(envoy_ext_authz_connect_authz_denied{local_cluster="emojify-api-v2"}[1m])
```

AuthZ OK

Grafana

Cached Authorization - no metrics

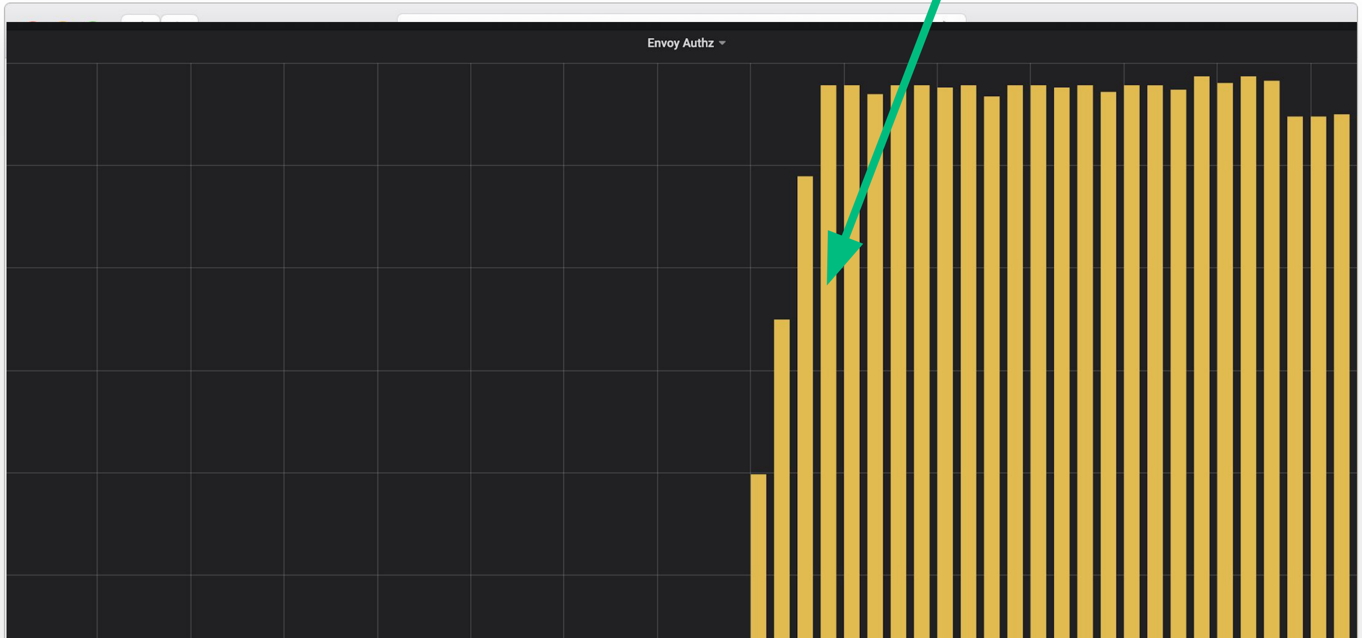
New pod started



AuthZ Failed

Grafana

AuthZ failure, either:
Misconfiguration or Attack



Tracing

What is tracing?

Tracing

Distributed tracing, also called distributed request tracing, is a **method used to profile and monitor applications**, especially those built using a microservices architecture. **Distributed tracing helps pinpoint where failures occur and what causes poor performance**.

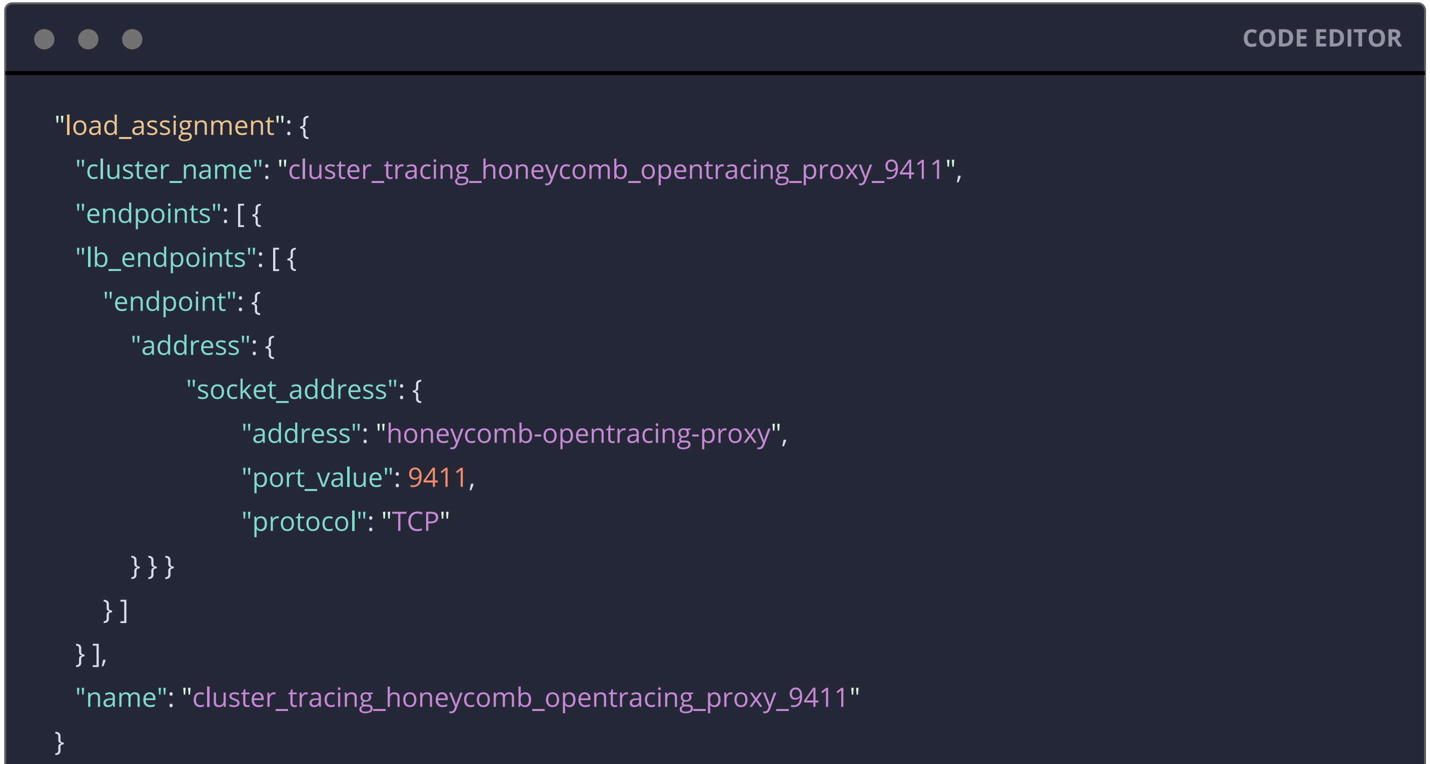
https://opentracing.io/docs/overview/what-is-tracing/

Configuration

Tracing

Tracing Cluster

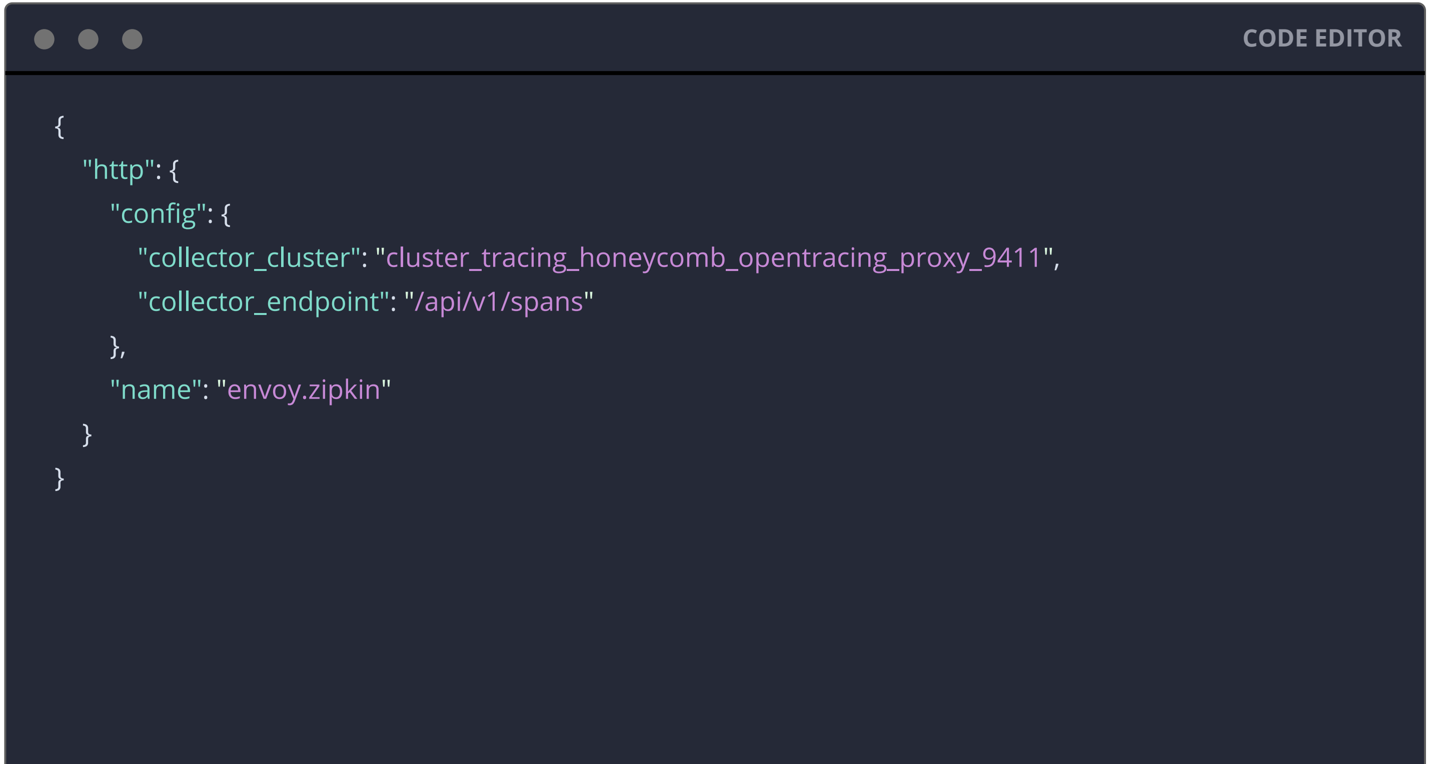
Configuration



```
"load_assignment": {
  "cluster_name": "cluster_tracing_honeycomb_opentracing_proxy_9411",
  "endpoints": [ {
  "lb_endpoints": [ {
    "endpoint": {
      "address": {
          "socket_address": {
              "address": "honeycomb-opentracing-proxy",
              "port_value": 9411,
              "protocol": "TCP"
      } } }
    } ]
  } ],
  "name": "cluster_tracing_honeycomb_opentracing_proxy_9411"
}
```

Tracing Configuration

Configuration



```
{
  "http": {
    "config": {
      "collector_cluster": "cluster_tracing_honeycomb_opentracing_proxy_9411",
      "collector_endpoint": "/api/v1/spans"
    },
    "name": "envoy.zipkin"
  }
}
```

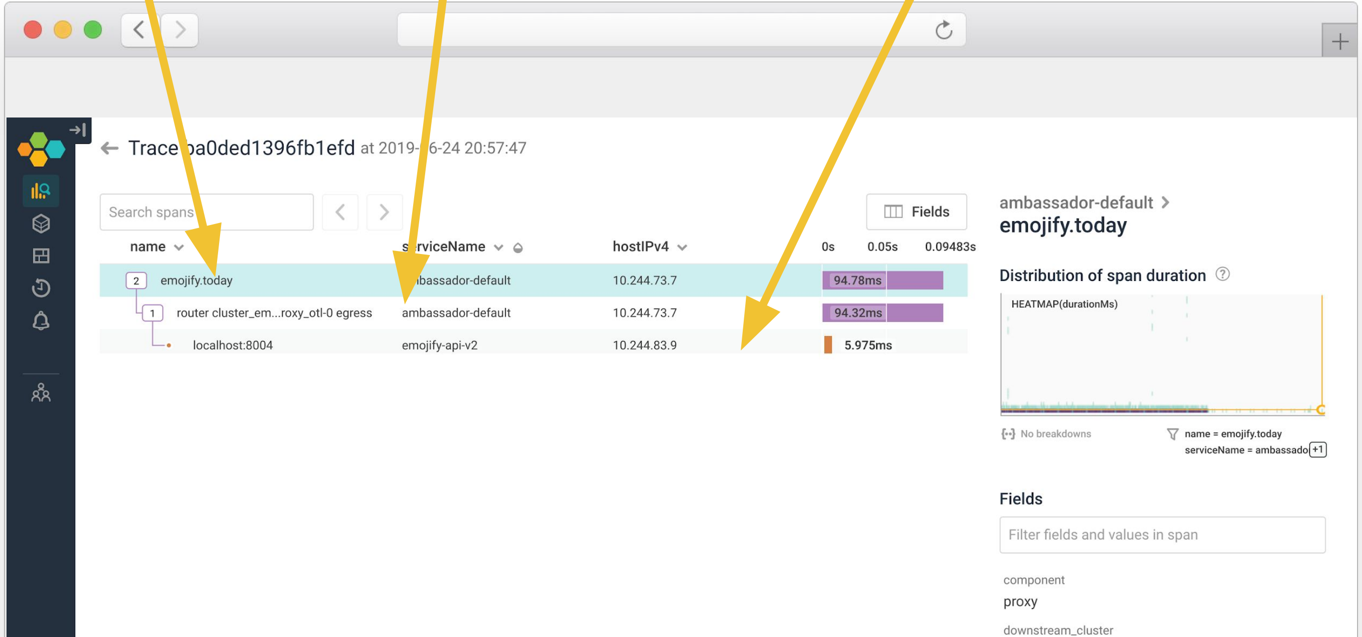
Trace - HTTP Post

honeycomb.io

Public Ingress

Route upstream to API

External upstream gRPC



← Trace ba0ded1396fb1efd at 2019-06-24 20:57:47

name	serviceName	hostIPv4	0s 0.05s 0.09483s
2 emojify.today	ambassador-default	10.244.73.7	94.78ms
1 router cluster_em...roxy_otl-0 egress	ambassador-default	10.244.73.7	94.32ms
localhost:8004	emojify-api-v2	10.244.83.9	5.975ms

ambassador-default >

emojify.today

Distribution of span duration

No breakdowns

name = emojify.today
serviceName = ambassado +1

Fields

component
proxy

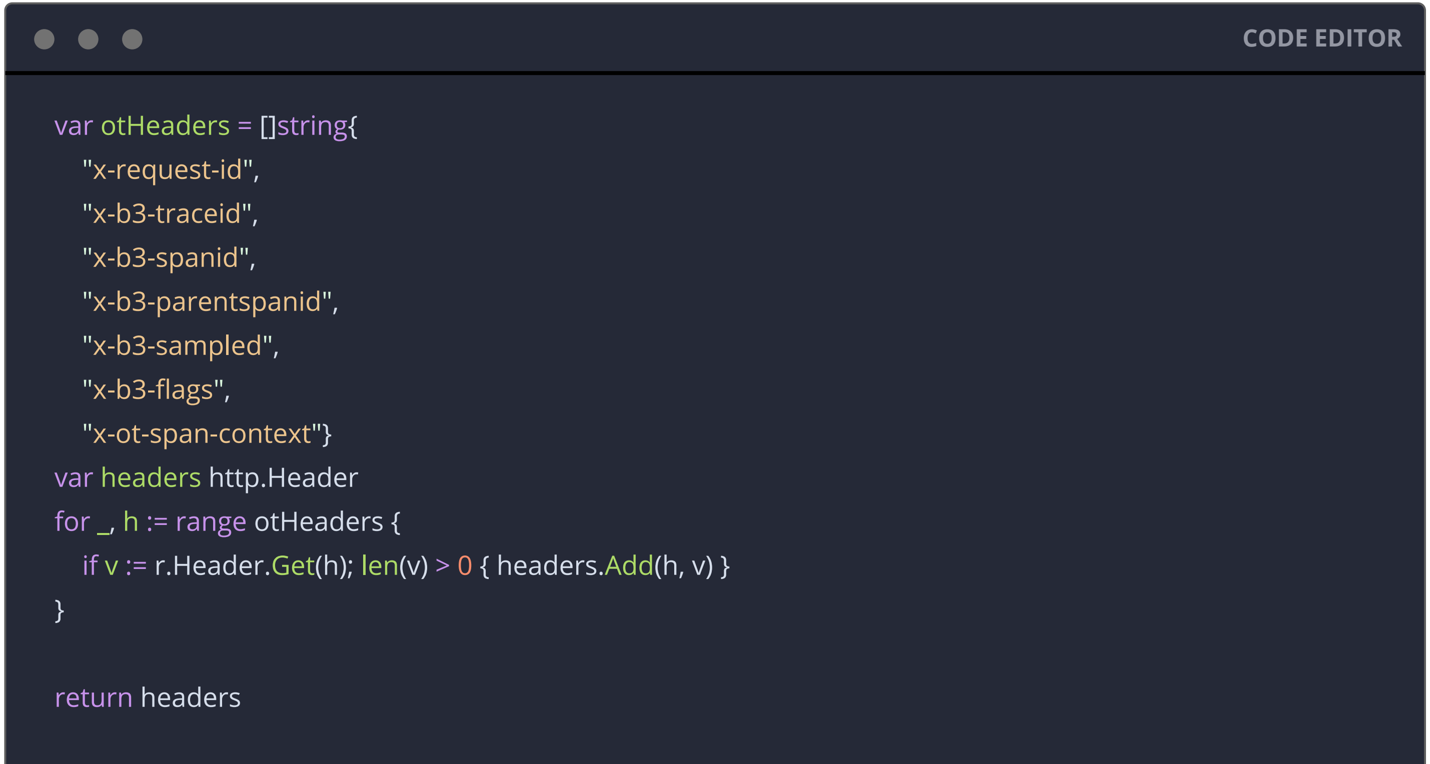
downstream_cluster

Handling tracing spans

Tracing

Adding Headers

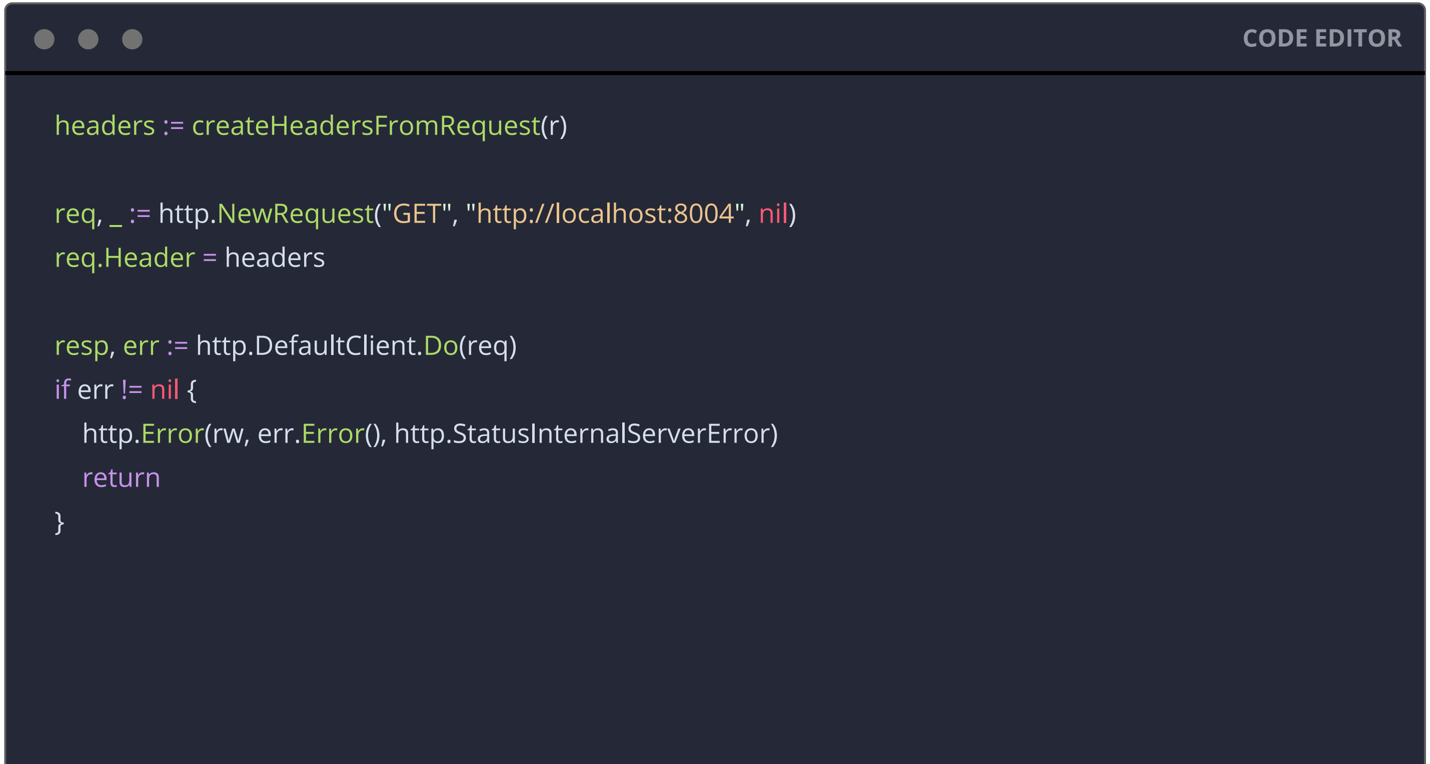
HTTP



```
var otHeaders = []string{
    "x-request-id",
    "x-b3-traceid",
    "x-b3-spanid",
    "x-b3-parentspanid",
    "x-b3-sampled",
    "x-b3-flags",
    "x-ot-span-context"}
var headers http.Header
for _, h := range otHeaders {
    if v := r.Header.Get(h); len(v) > 0 { headers.Add(h, v) }
}

return headers
```

Adding Headers

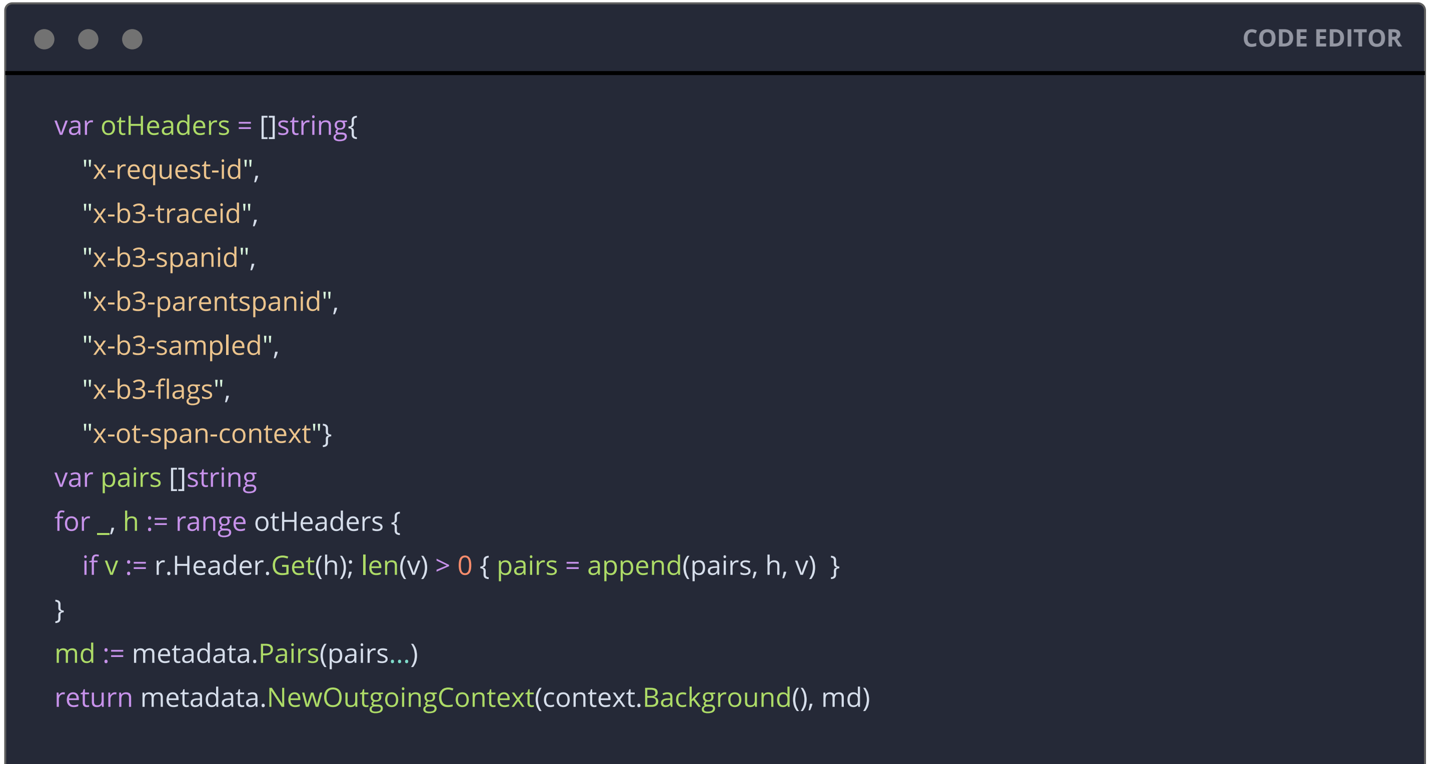
HTTP



```
headers := createHeadersFromRequest(r)

req, _ := http.NewRequest("GET", "http://localhost:8004", nil)
req.Header = headers

resp, err := http.DefaultClient.Do(req)
if err != nil {
    http.Error(rw, err.Error(), http.StatusInternalServerError)
    return
}
```

Adding Headers

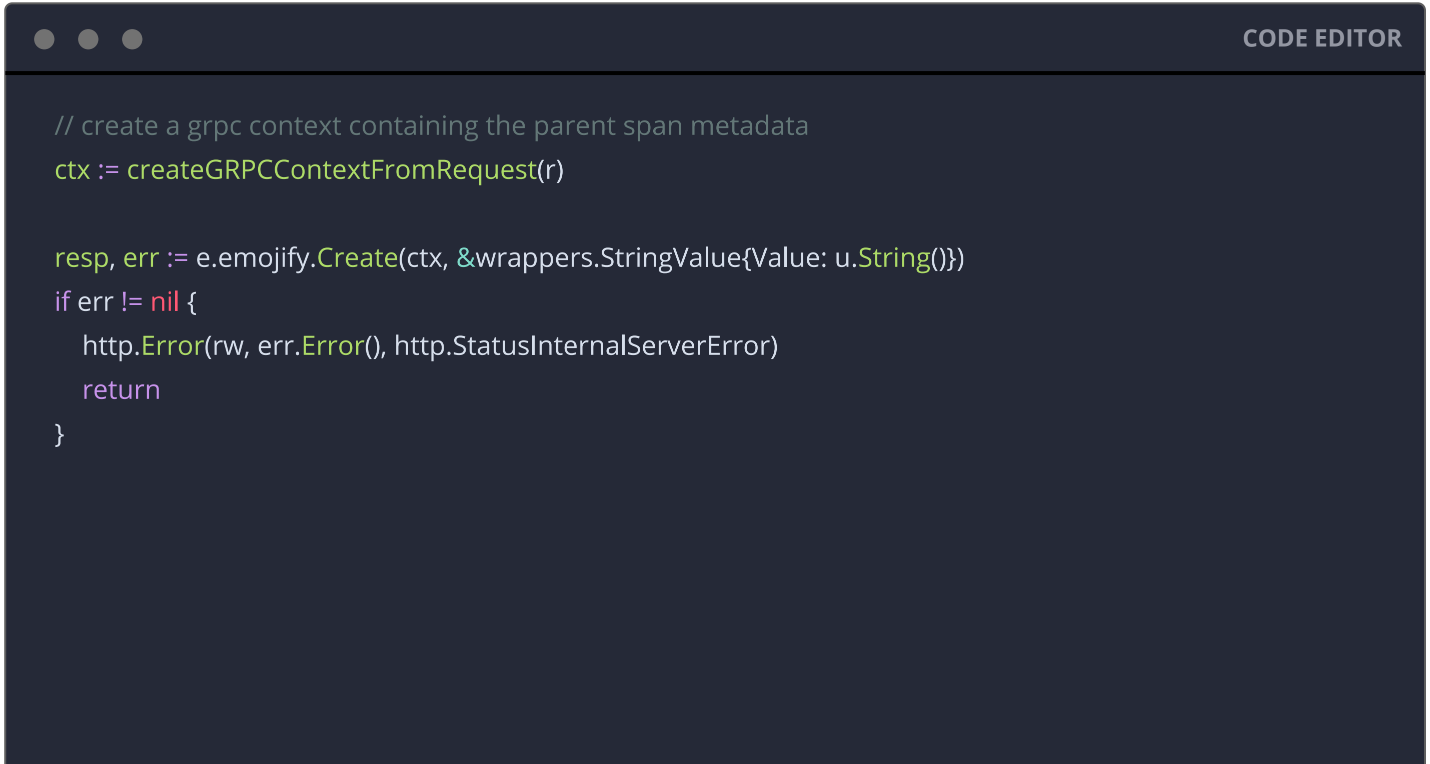
gRPC



```
var otHeaders = []string{
   "x-request-id",
   "x-b3-traceid",
   "x-b3-spanid",
   "x-b3-parentspanid",
   "x-b3-sampled",
   "x-b3-flags",
   "x-ot-span-context"}
var pairs []string
for _, h := range otHeaders {
   if v := r.Header.Get(h); len(v) > 0 { pairs = append(pairs, h, v)  }
}
md := metadata.Pairs(pairs...)
return metadata.NewOutgoingContext(context.Background(), md)
```

Adding Headers

gRPC

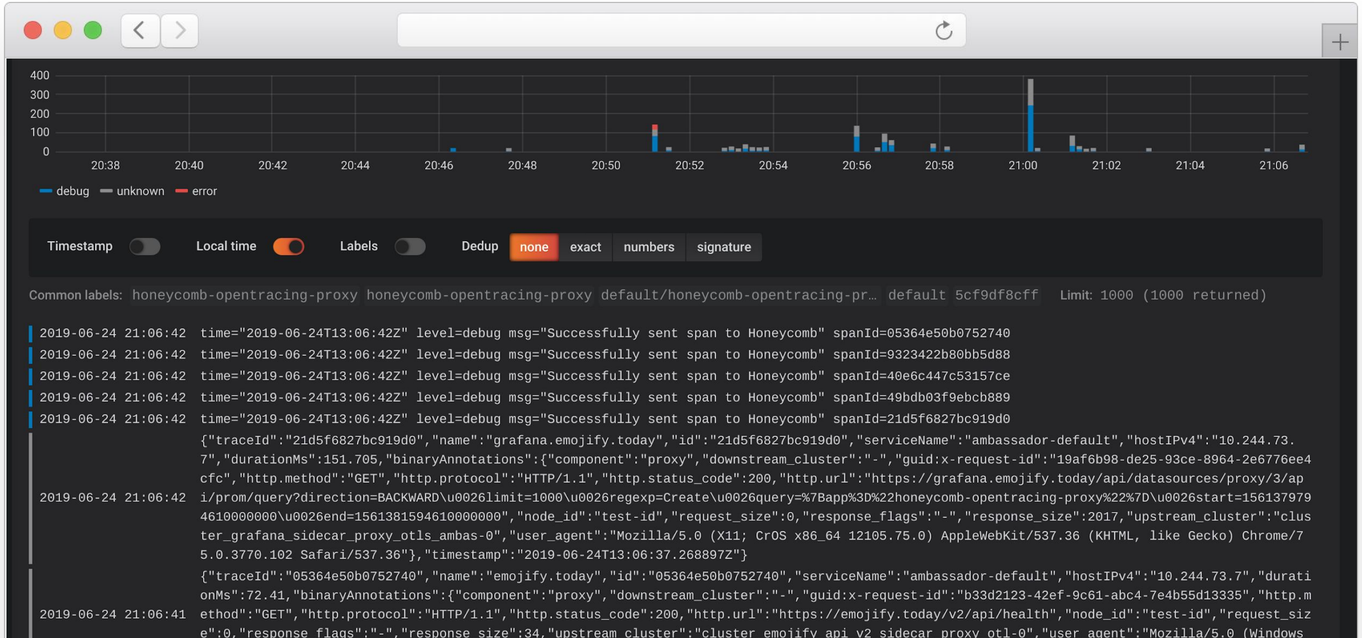


```go
// create a grpc context containing the parent span metadata
ctx := createGRPCContextFromRequest(r)

resp, err := e.emojify.Create(ctx, &wrappers.StringValue{Value: u.String()})
if err != nil {
   http.Error(rw, err.Error(), http.StatusInternalServerError)
   return
}
```

Logging

Logging

Loki



Timestamp ○ Local time ● Labels ○ Dedup [✓ none] exact numbers signature

Common labels: honeycomb-opentracing-proxy honeycomb-opentracing-proxy default/honeycomb-opentracing-pr… default 5cf9df8cff Limit: 1000 (1000 returned)

```
2019-06-24 21:06:42  time="2019-06-24T13:06:42Z" level=debug msg="Successfully sent span to Honeycomb" spanId=05364e50b0752740
2019-06-24 21:06:42  time="2019-06-24T13:06:42Z" level=debug msg="Successfully sent span to Honeycomb" spanId=9323422b80bb5d88
2019-06-24 21:06:42  time="2019-06-24T13:06:42Z" level=debug msg="Successfully sent span to Honeycomb" spanId=40e6c447c53157ce
2019-06-24 21:06:42  time="2019-06-24T13:06:42Z" level=debug msg="Successfully sent span to Honeycomb" spanId=49bdb03f9ebcb889
2019-06-24 21:06:42  time="2019-06-24T13:06:42Z" level=debug msg="Successfully sent span to Honeycomb" spanId=21d5f6827bc919d0
                     {"traceId":"21d5f6827bc919d0","name":"grafana.emojify.today","id":"21d5f6827bc919d0","serviceName":"ambassador-default","hostIPv4":"10.244.73.
                     7","durationMs":151.705,"binaryAnnotations":{"component":"proxy","downstream_cluster":"-","guid:x-request-id":"19af6b98-de25-93ce-8964-2e6776ee4
                     cfc","http.method":"GET","http.protocol":"HTTP/1.1","http.status_code":200,"http.url":"https://grafana.emojify.today/api/datasources/proxy/3/ap
2019-06-24 21:06:42  i/prom/query?direction=BACKWARD\u0026limit=1000\u0026regexp=Create\u0026query=%7Bapp%3D%22honeycomb-opentracing-proxy%22%7D\u0026start=156137979
                     4610000000\u0026end=1561381594610000000","node_id":"test-id","request_size":0,"response_flags":"-","response_size":2017,"upstream_cluster":"clus
                     ter_grafana_sidecar_proxy_otls_ambas-0","user_agent":"Mozilla/5.0 (X11; CrOS x86_64 12105.75.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7
                     5.0.3770.102 Safari/537.36"},"timestamp":"2019-06-24T13:06:37.268897Z"}
                     {"traceId":"05364e50b0752740","name":"emojify.today","id":"05364e50b0752740","serviceName":"ambassador-default","hostIPv4":"10.244.73.7","durati
                     onMs":72.41,"binaryAnnotations":{"component":"proxy","downstream_cluster":"-","guid:x-request-id":"b33d2123-42ef-9c61-abc4-7e4b55d13335","http.m
2019-06-24 21:06:41  ethod":"GET","http.protocol":"HTTP/1.1","http.status_code":200,"http.url":"https://emojify.today/v2/api/health","node_id":"test-id","request_siz
                     e":0,"response_flags":"-","response_size":34,"upstream_cluster":"cluster_emojify_api_v2_sidecar_proxy_otl-0","user_agent":"Mozilla/5.0 (Windows
```

Thank You

nic@hashicorp.com
www.hashicorp.com