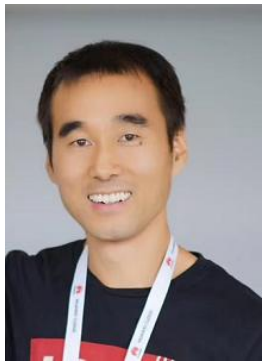
Istio 非侵入可观测性架构的演进解析和实践

张超盟 华为云

讲师简介



张超盟

华为云应用服务网格首席架构师

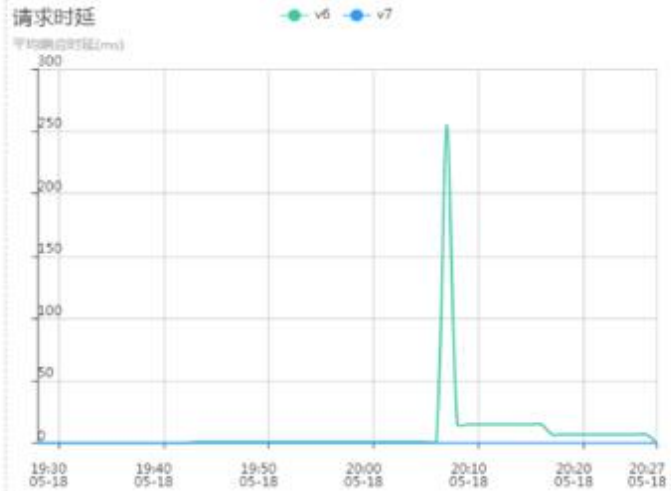
先后负责华为云容器应用运维、微服务平台、云服务目录、服务网格等产品架构设计与开发工作，在服务网格、Kubernetes 容器服务、微服务架构、大数据、应用性能管理、数据中间件及 DevOps 工具等领域有深入研究与实践。开源爱好者，Istio 社区成员，KubeCon，IstioCon、ServiceMeshCon等演讲者。著有《云原生服务网格 Istio：原理、实践、架构与源码解析》一书。

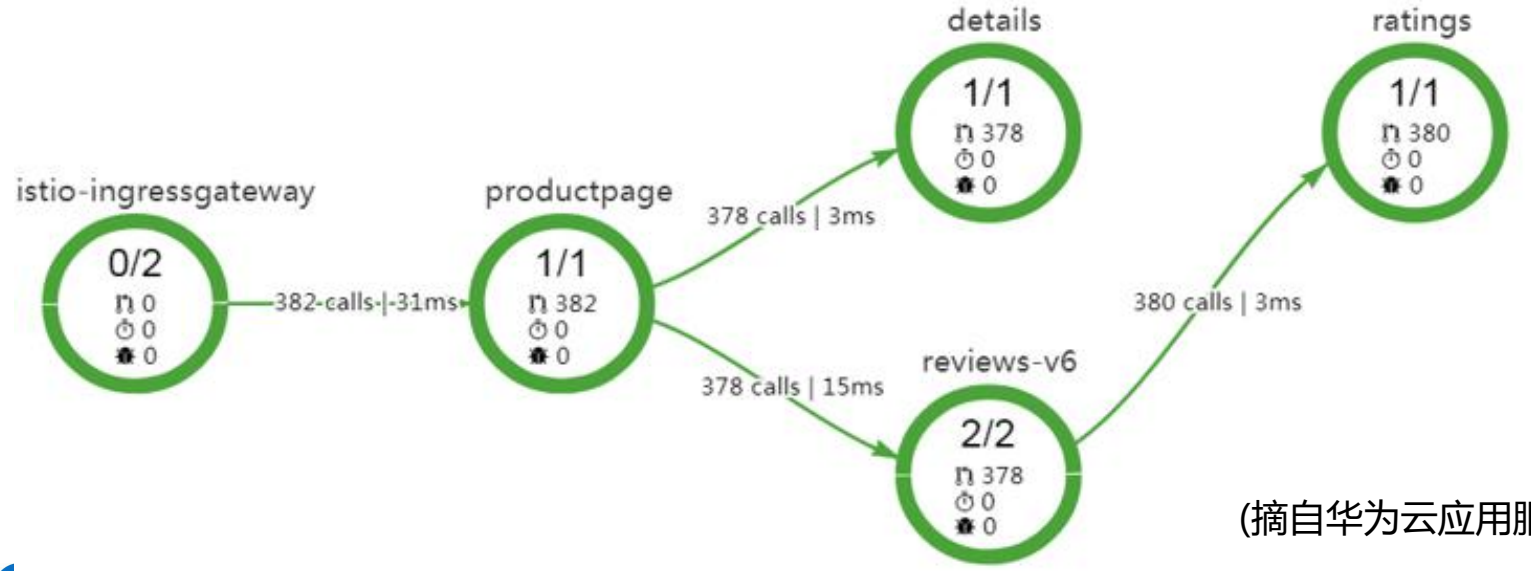
目录

- 非侵入可扩展架构特点
- Metric 架构原理和实践
- Tracing 架构原理和实践
- AccessLog 架构原理和实践
- 发展和规划

可观测性是服务运维的基础

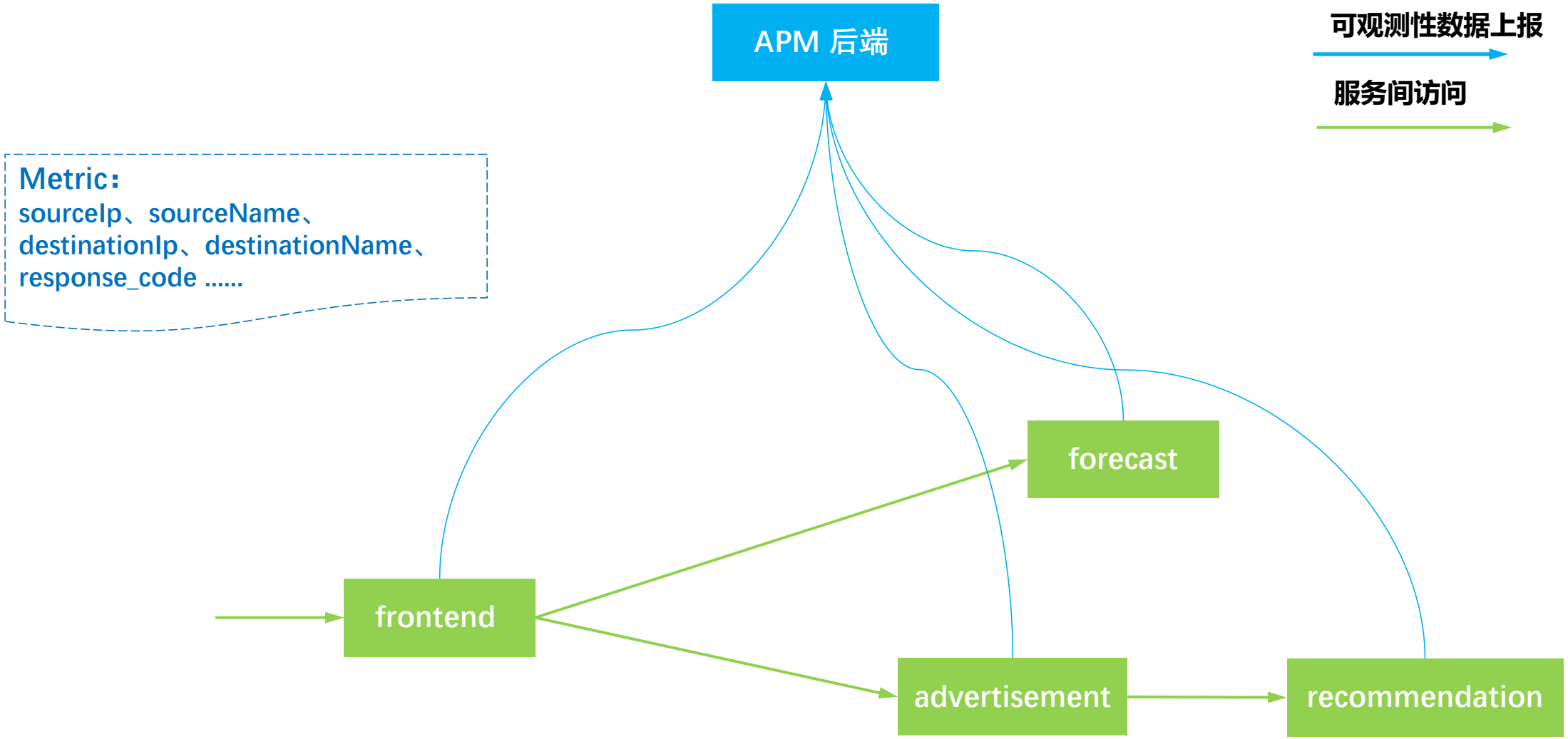
msup®



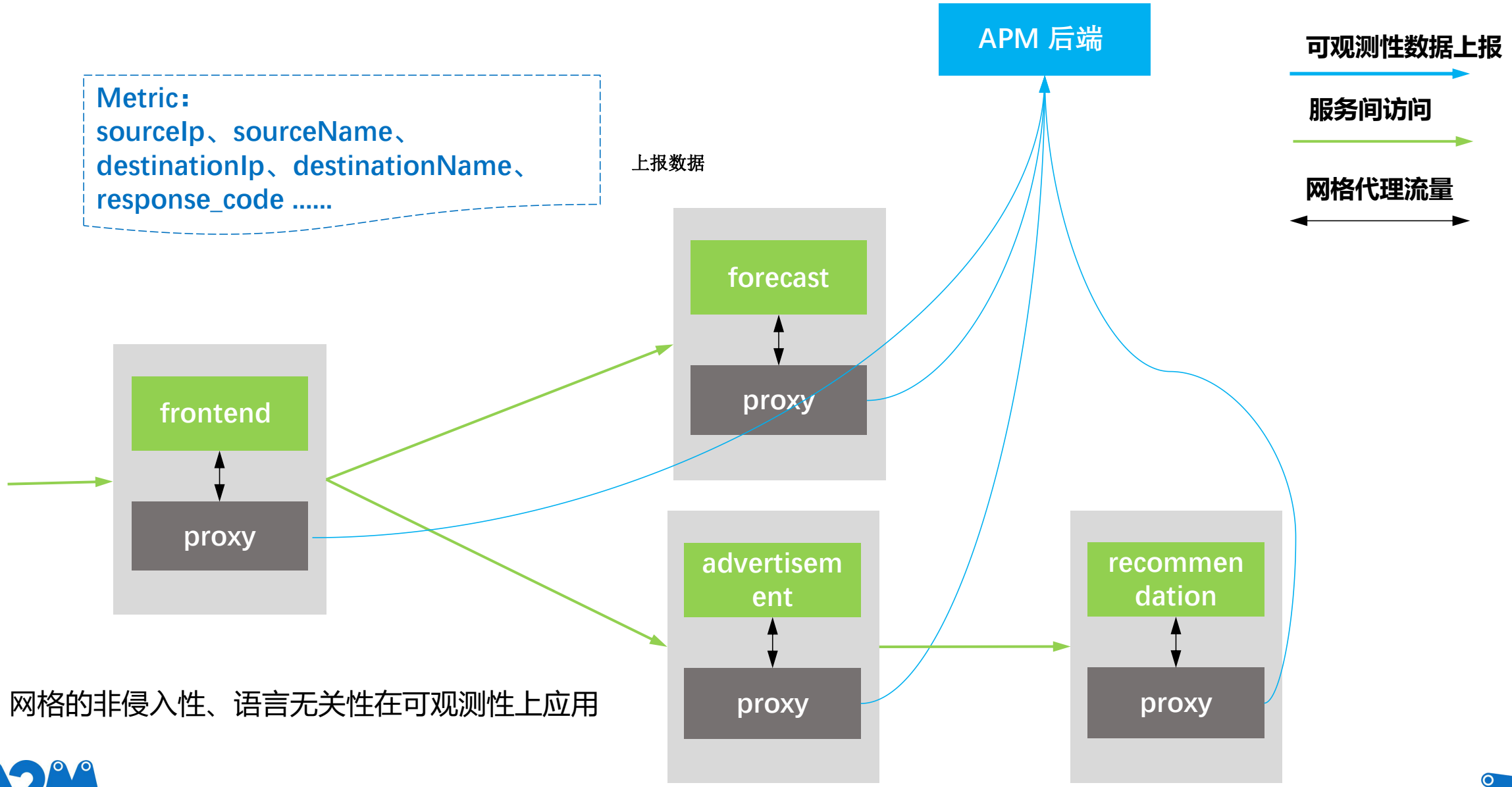


(摘自华为云应用服务网格ASM)

可观测性数据采集

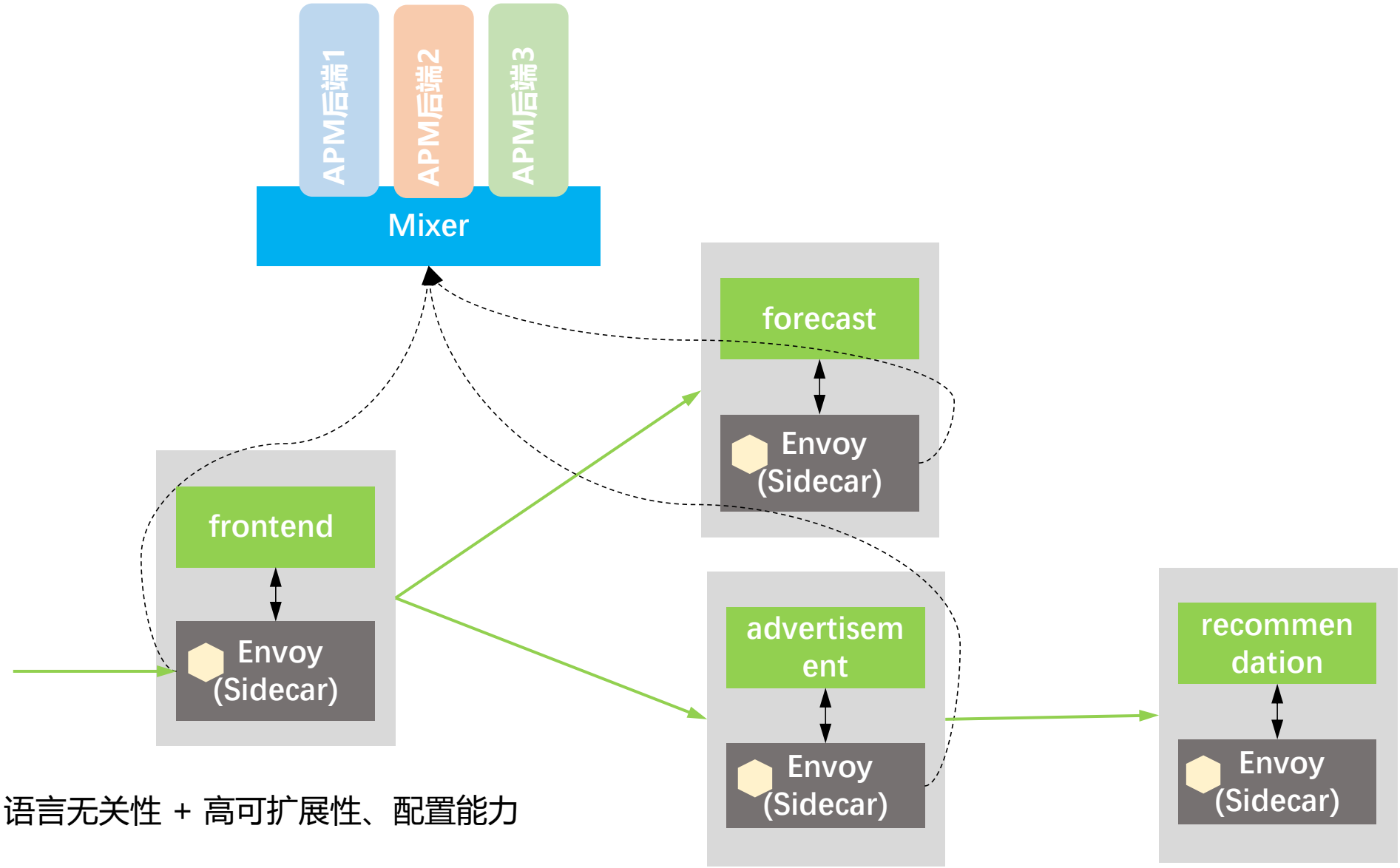

APM 后端
可观测性数据上报
服务间访问
Metric：
sourceIp、sourceName、
destinationIp、destinationName、
response_code .....
forecast
frontend
advertisement
recommendation

基于网格的可观测性数据收集



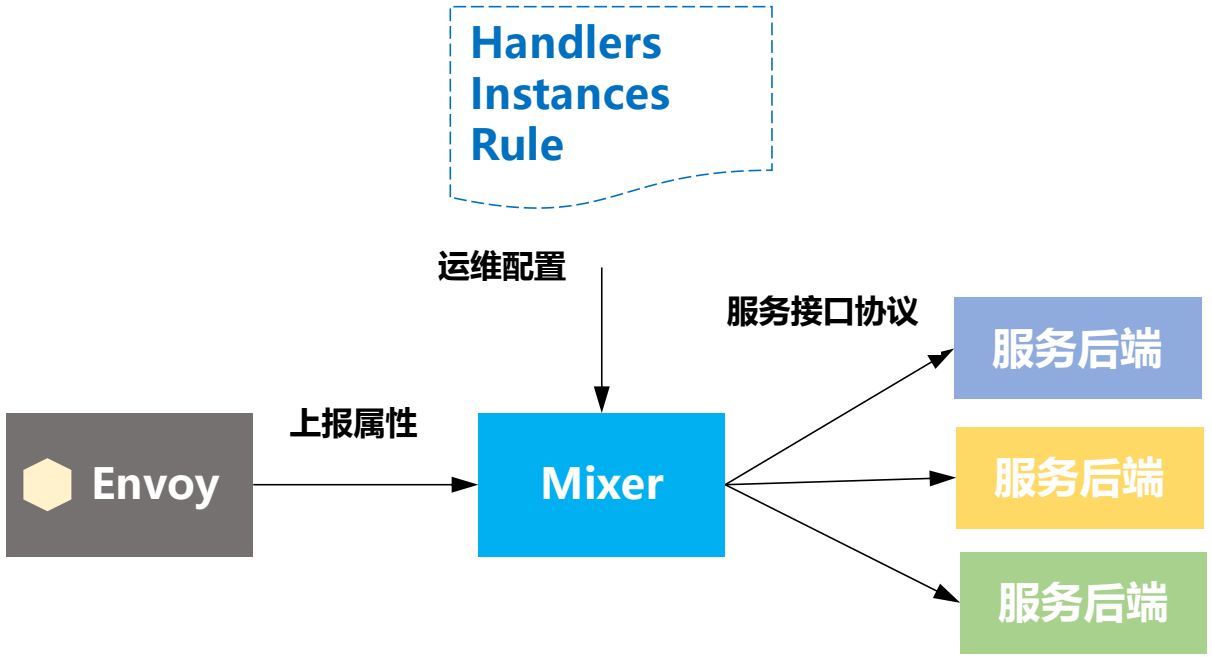
网格的非侵入性、语言无关性在可观测性上应用

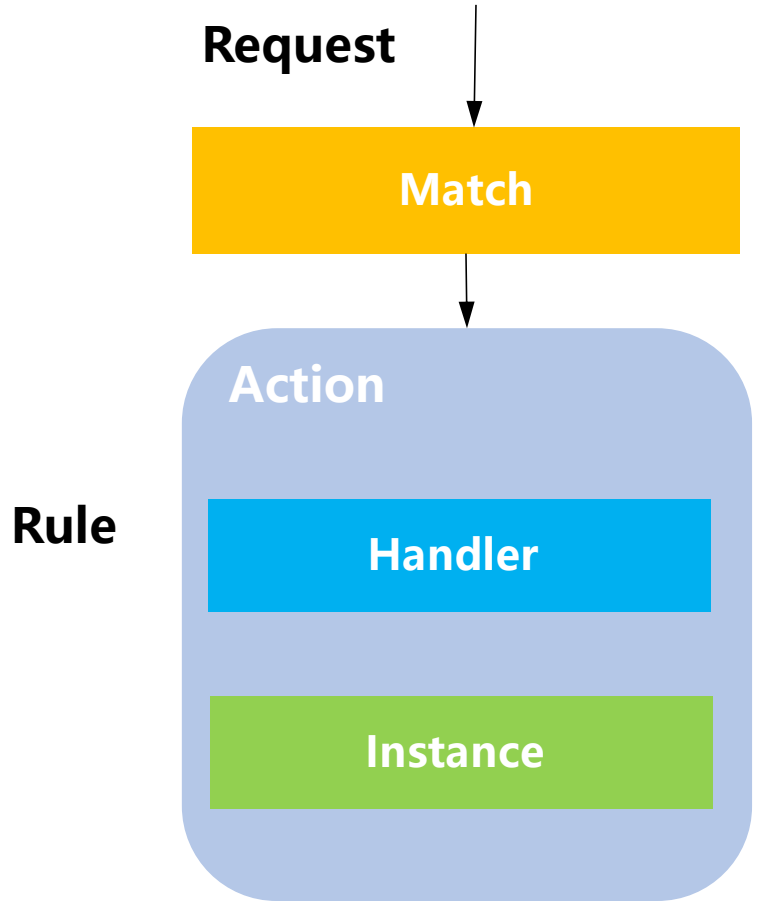
基于Istio的观测性数据收集(Telemetry V1)



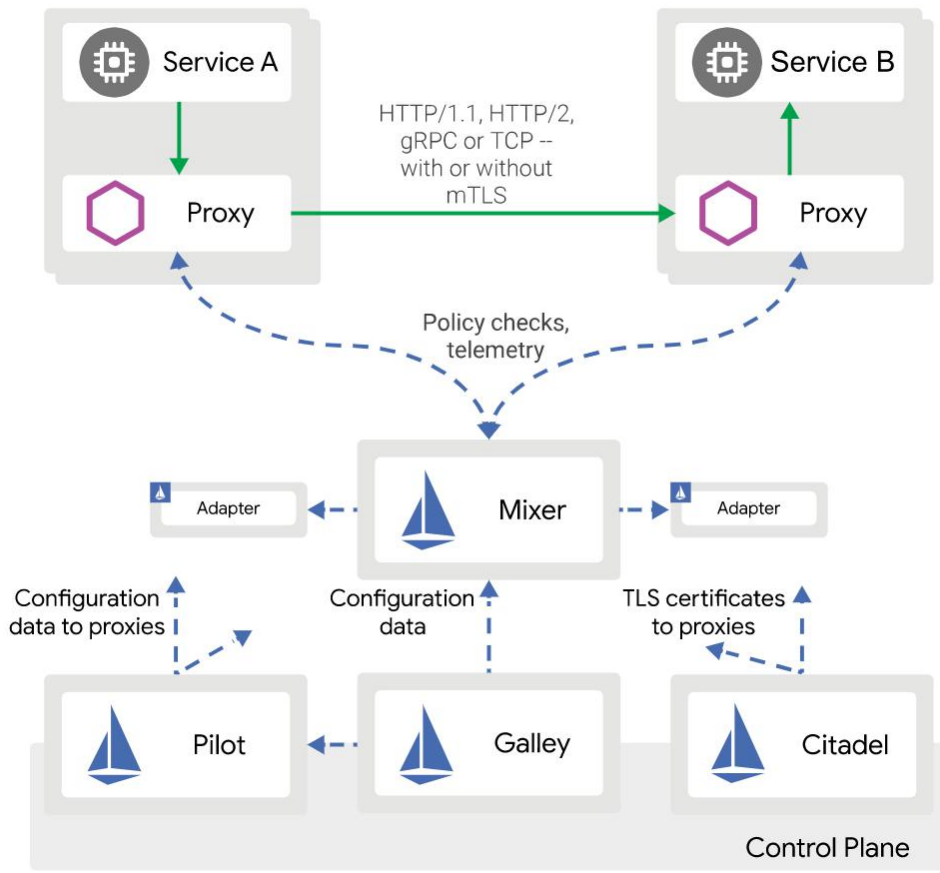
网格的非侵入性、语言无关性 + 高可扩展性、配置能力

Istio Telemetry V1 Adapter 机制

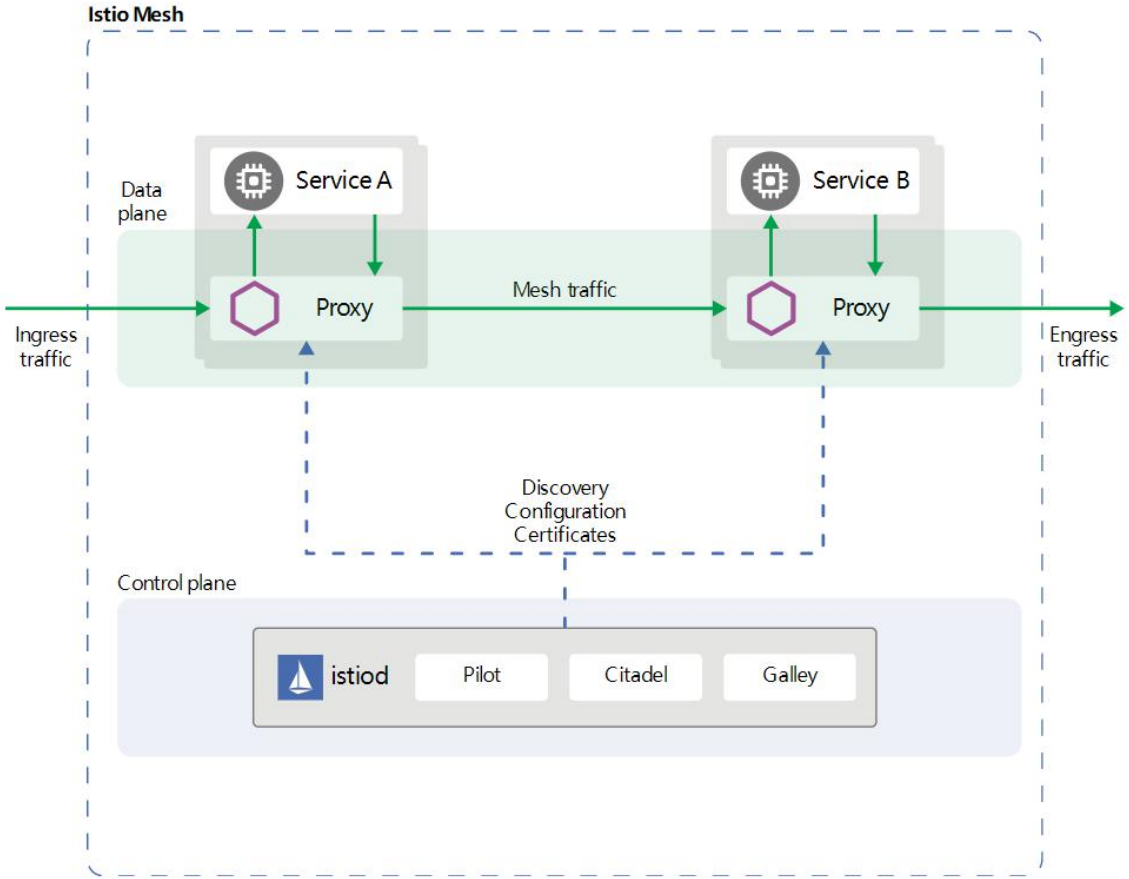




Istio 架构变化



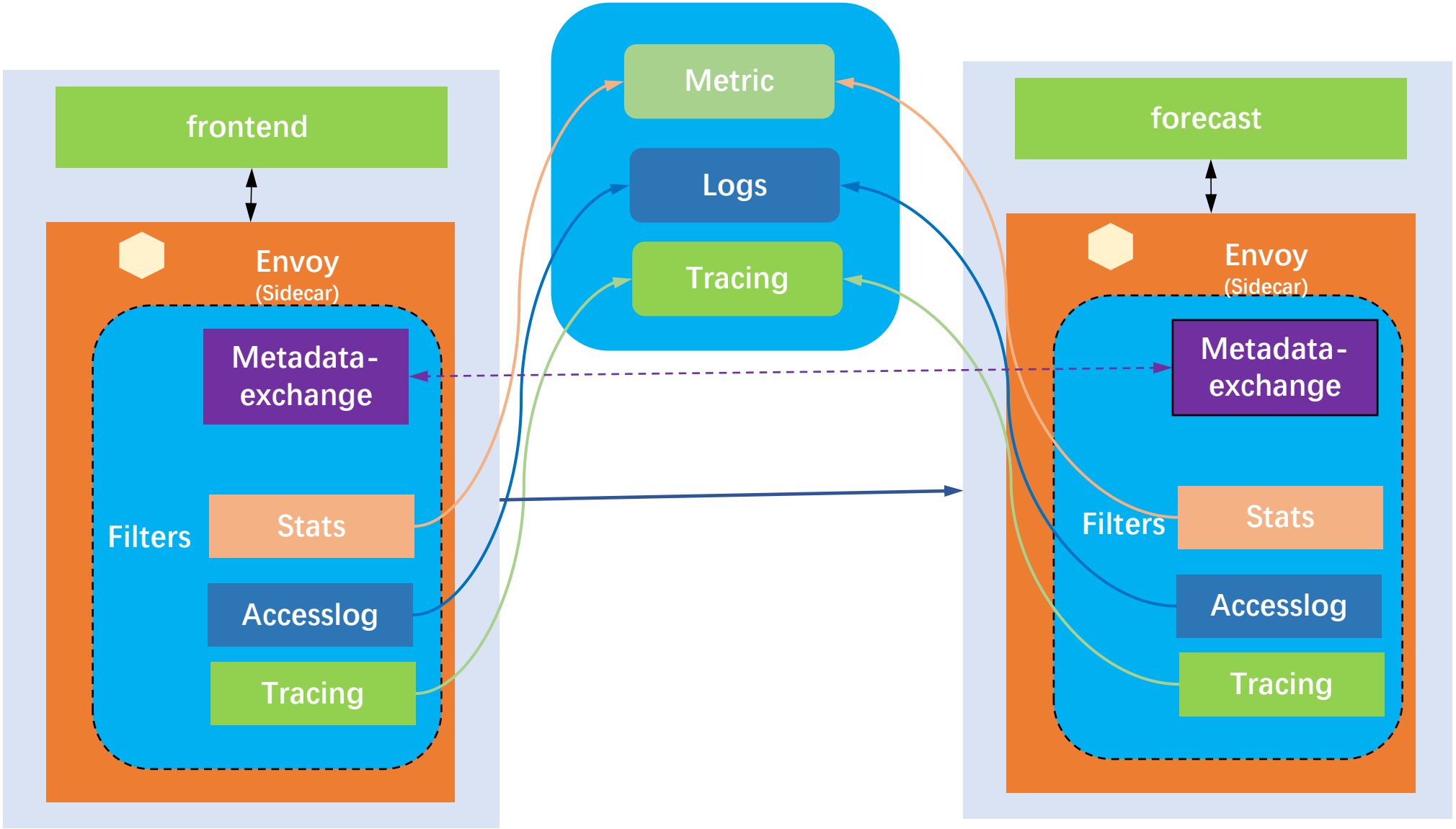
v1.3



v1.5+

- 控制面组件合一
- Mixer下线

基于Istio的观测性数据收集(Telemetry V2)



目录

- 非侵入可扩展架构特点
- Metric 架构原理和实践
- Tracing 架构原理和实践
- AccessLog 架构原理和实践
- 发展和规划

Metric 架构原理(Telemetry V1)

1. Envoy通过Report接口上报数据给Mixer。
2. Mixer根据配置将请求分发给Prometheus Adapter。
3. Prometheus Adapter通过HTTP接口发布Metric数据。
4. Prometheus服务作为Addon在集群中进行安装，并拉取、存储Metric数据，提供Query接口进行检索。
5. 集群内的Dashboard如Grafana通过Prometheus的检索API访问Metric数据。

Metric配置实践(Telemetry V1)

msup

Prometheus handler 配置

```
apiVersion: "config.istio.io/v1alpha2"
kind: handler
metadata:
  name: prometheus
  namespace: istio-system
spec:
  compiledAdapter: prometheus
  params:
    metricsExpirationPolicy:
      metricsExpiryDuration: 15s
    metrics:
    - name: requests_total
      instance_name: requestcount.metric.istio-system
      kind: COUNTER
      label_names:
      - source_app
      - source_principal
      - destination_service_name
……
    - name: request_duration_seconds
      instance_name: requestduration.metric.istio-system
      kind: DISTRIBUTION
……
    - name: request_bytes
      instance_name: requestsize.metric.istio-system
      kind: DISTRIBUTION
……
    - name: response_bytes
      instance_name: responsesize.metric.istio-system
      kind: DISTRIBUTION
```

定义Metric，配置Metric的字段

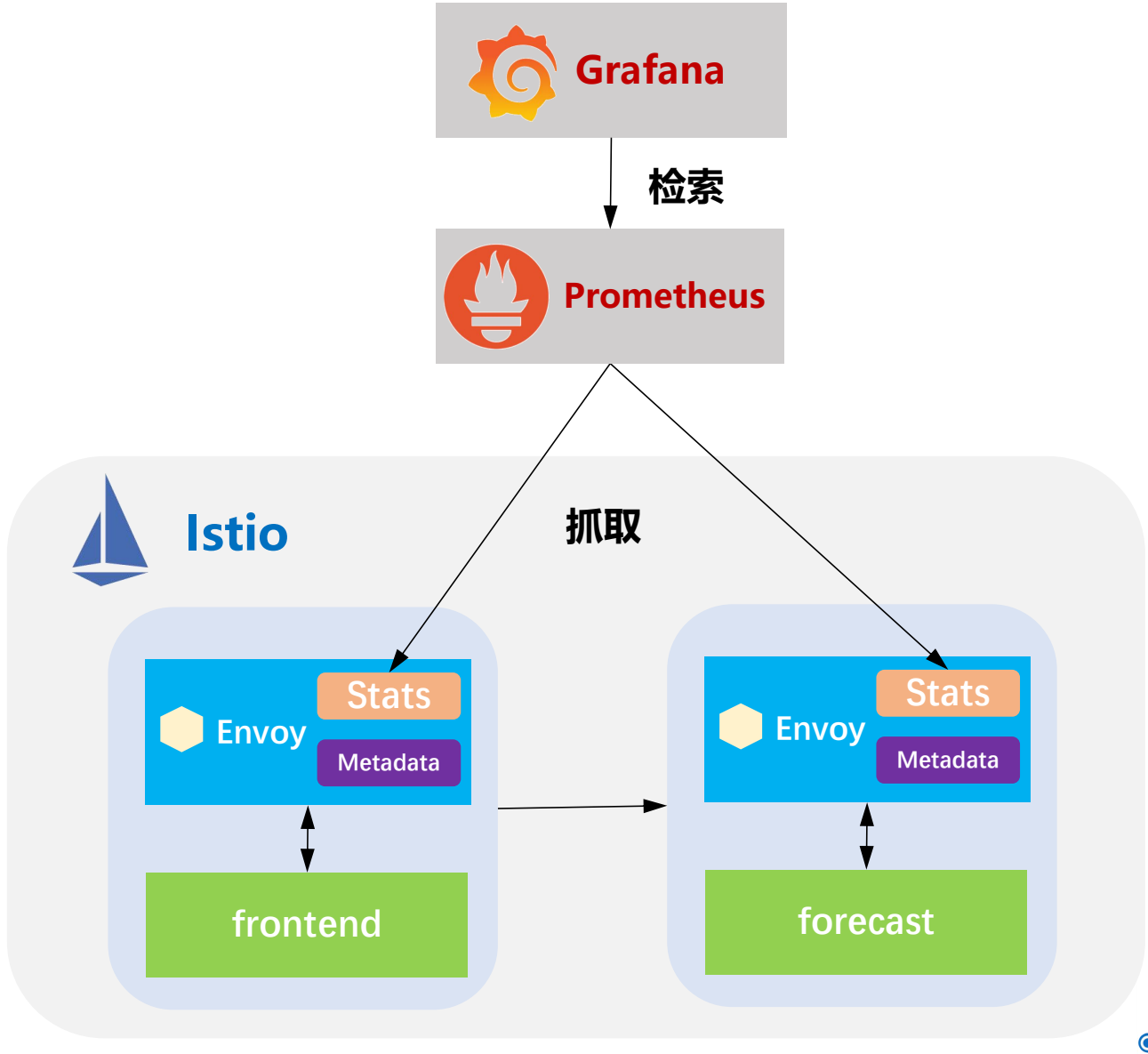
```
apiVersion: "config.istio.io/v1alpha2"
kind: instance
metadata:
  name: requestcount
  namespace: istio-system
spec:
  compiledTemplate: metric
  params:
    value: "1" # count each request twice
    dimensions:
      reporter: conditional((context.reporter.kind | "inbound") ==
"outbound", "source", "destination")
      source_workload_namespace: source.workload.namespace |
"unknown"
      source_principal: source.principal | "unknown"
      source_app: source.labels["app"] | "unknown"
      source_version: source.labels["version"] | "unknown"
      destination_workload: destination.workload.name | "unknown"
      destination_workload_namespace:
destination.workload.namespace | "unknown"
      destination_principal: destination.principal | "unknown"
      destination_app: destination.labels["app"] | "unknown"
      destination_version: destination.labels["version"] | "unknown"
      destination_service: destination.service.host | "unknown"
      destination_service_name: destination.service.name |
"unknown"
      destination_service_namespace: destination.service.namespace
| "unknown"
      request_protocol: api.protocol | context.protocol | "unknown"
      response_code: response.code | 200
      response_flags: context.proxy_error_code | "-"
```

关联Hander和Instance

```
apiVersion: "config.istio.io/v1alpha2"
kind: rule
metadata:
  name: promhttp
  namespace: istio-system
spec:
  match: (context.protocol == "http" ||
context.protocol == "grpc") &&
(match((request.useragent | "-"), "kube-probe*")
== false)
  actions:
  - handler: prometheus
    instances:
    - requestcount
    - requestduration
    - requestsize
    - responsesize
```

Metric 架构原理(Telemetry V2)

1. Envoy代理拦截到服务间访问，计算本代理上生成的访问指标。
2. Envoy通过Metadata-exchange交换源和目标的元数据信息，形成完整Metric数据。
3. Envoy通过Prometheus Exporter接口发布Metric数据。
4. Prometheus服务作为Addon在集群中进行安装，并拉取、存储Metric数据，提供Query接口进行检索。
5. 集群内的Dashboard如Grafana通过Prometheus的检索API访问Metric数据。



Metric配置实践(Telemetry V2)

msup

启用Telemetry v2

```
spec:
  addonComponents:
    prometheus:
      enabled: true
  values:
    telemetry:
      v2:
        enabled: true
```

定义Metric，配置Metric的字段

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    telemetry:
      v2:
        prometheus:
          configOverride:
            outboundSidecar:
              metrics:
                - name: requests_total
                  dimensions:
                    destination_port: string(destination.port)
                    request_host: request.host
                    request_method: request.method
                    request_url: request.url_path
                    source_pod: node.metadata['NAME']
                    destination_pod: upstream_peer.name
                  tags_to_remove: ['response_flags']
            inboundSidecar:
            ...
            gateway:
            ...
```

配置extraStatTags来定义要提取的Metric标签

```
meshConfig:
  defaultConfig:
    extraStatTags:
      - destination_pod
      - source_pod
      - request_method
      - request_url
      - source_mesh_id
      - source_mesh_name
      - source_cluster_id
      - source_cluster_name
      - destination_mesh_id
      - destination_mesh_name
      - destination_cluster_id
      - destination_cluster_name
```

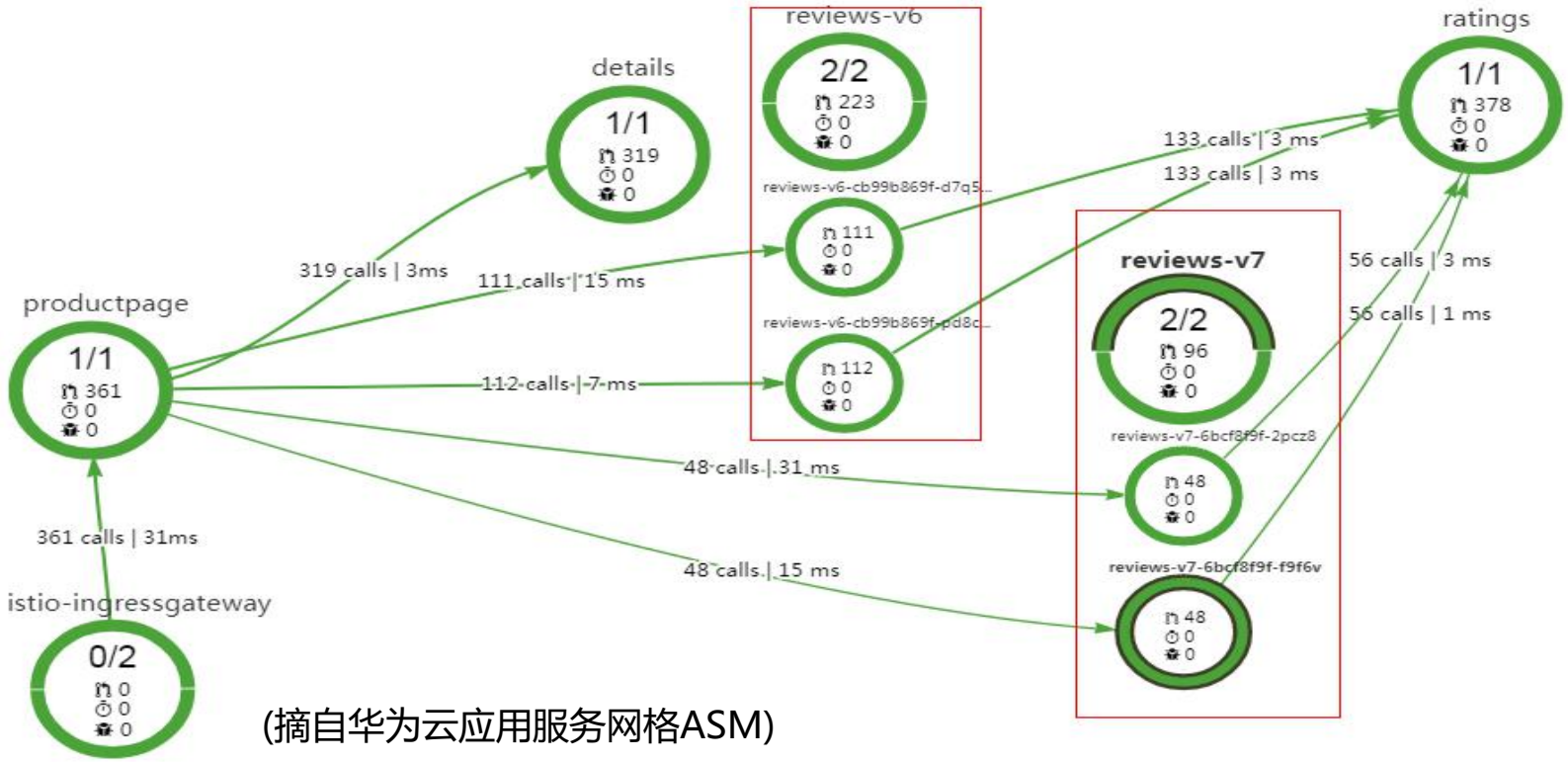
网格典型 Metric

指标	协议	V2指标名	V1 Instance	指标类型	指标含义
请求数	HTTP	istio_requests_total	requestcount	Counter	计数处理的每个请求。可以基于此计算服务单位时间内处理的请求数，表示目标服务的吞吐量。可以通过计数错误的请求数，进而得到错误率等指标。
请求耗时	HTTP	istio_request_duration_milliseconds	requestduration	Histogram	一个请求耗费的时间。服务响应时延，衡量服务性能的重要指标，表现为服务的响应效率。
请求大小	HTTP	istio_request_bytes	requestsize	Histogram	HTTP请求体大小。
应答大小	HTTP	istio_response_bytes	responsesize	Histogram	HTTP应答体大小。
gRPC 请求数	gRPC	(istio_request_messages_total		Counter	计数gRPC客户端的发送的请求数。
gRPC 应答数	gRPC	istio_response_messages_total		Counter	计数gRPC服务端发送的应答数。
TCP发送字节数	TCP	istio_tcp_sent_bytes_total	tcpbytesent	Counter	一个TCP连接上上行流量字节数的计数。
TCP接收字节数	TCP	istio_tcp_received_bytes_total	tcpbytereceived	Counter	一个TCP连接上下行流量字节数的计数。
TCP打开连接数	TCP	istio_tcp_connections_opened_total	tcpconnectionsopened	Counter	打开的TCP连接数。
TCP关闭连接数	TCP	istio_tcp_connections_closed_total	tcpconnectionsopened	Counter	关闭的TCP连接数

Metric 标签

标签	用法
Reporter	表示上报着的角色，如果来自目标服务的代理则设置为destination；如果来自源服务代理或者Gateway则设置为source
Source Workload	源服务负载名
Source Workload Namespace	源服务负载的Namespace
Source Principal	源服务主体身份，在启用了双向认证时使用。
Source App	源服务App标签
Source Version	源服务版本
Destination Workload	目标服务负载名
Destination Workload Namespace	目标服务负载的Namespace
Destination Principal	目标服务主体身份，在启用了双向认证时使用。
Destination App	目标服务App标签
Destination Version	目标服务版本
Destination Service	目标服务访问Host信息
Destination Service Name	目标服务名
Destination Service Namespace	目标服务Namespace
Request Protocol	请求协议
Response Code	响应码
Connection Security Policy	认证策略
Response Flags	应答标记
Destination Cluster	目标集群
Source Cluster	源集群

Metric 应用



(摘自华为云应用服务网格ASM)

服务版本粒度、实例粒度上的服务访问的响应时间、流量、错误率等指标，服务健康状态。

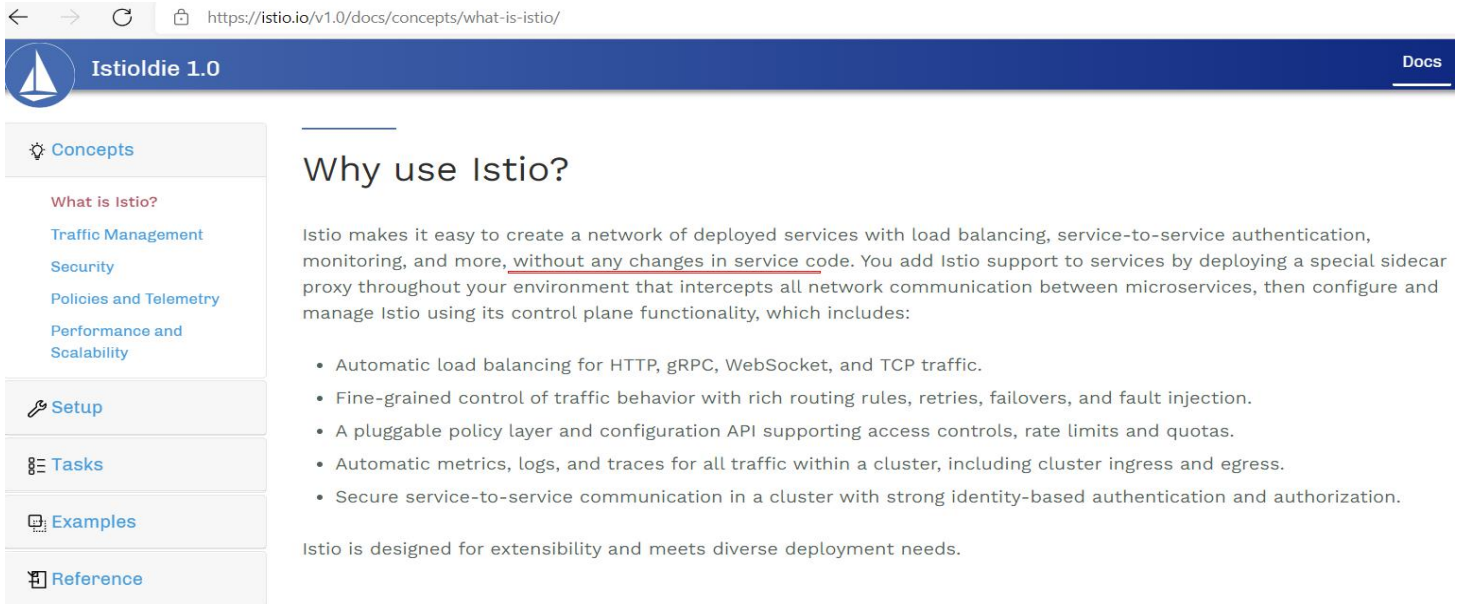
目录

- 非侵入可扩展架构特点
- Metric 架构原理和实践
- Tracing 架构原理和实践
- AccessLog 架构原理和实践
- 发展和规划

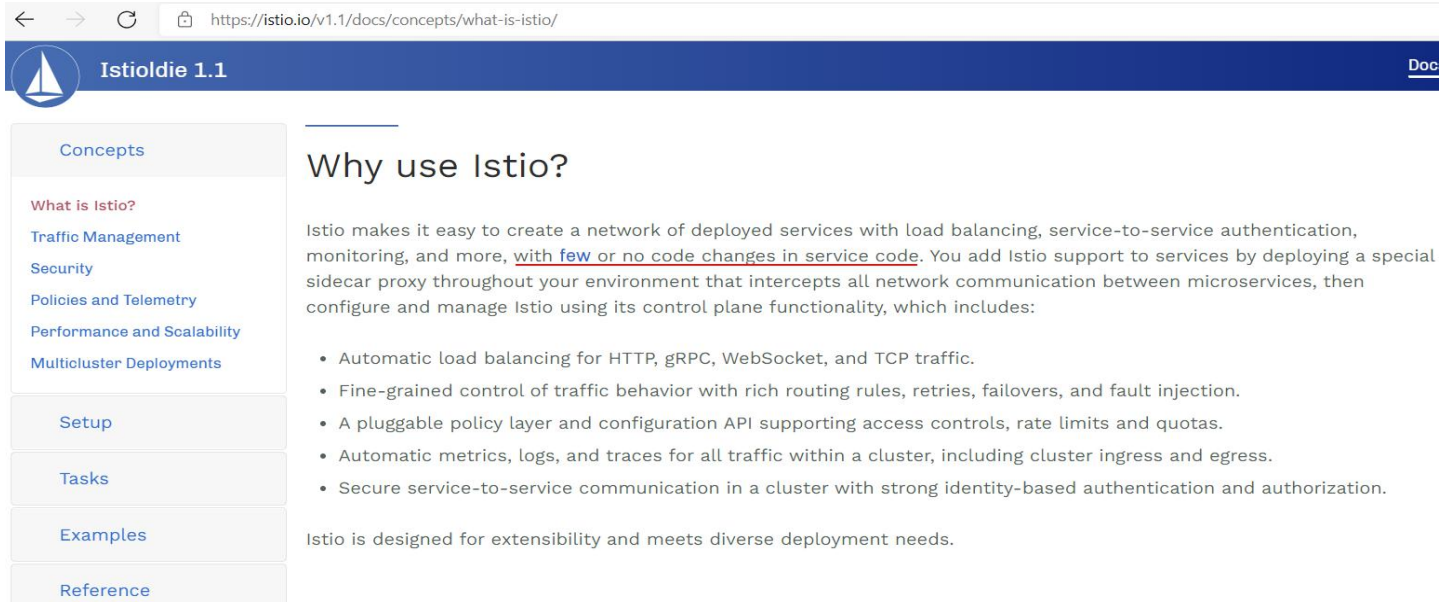
Istio等网格调用链并非完全应用零修改

msup

v1.0

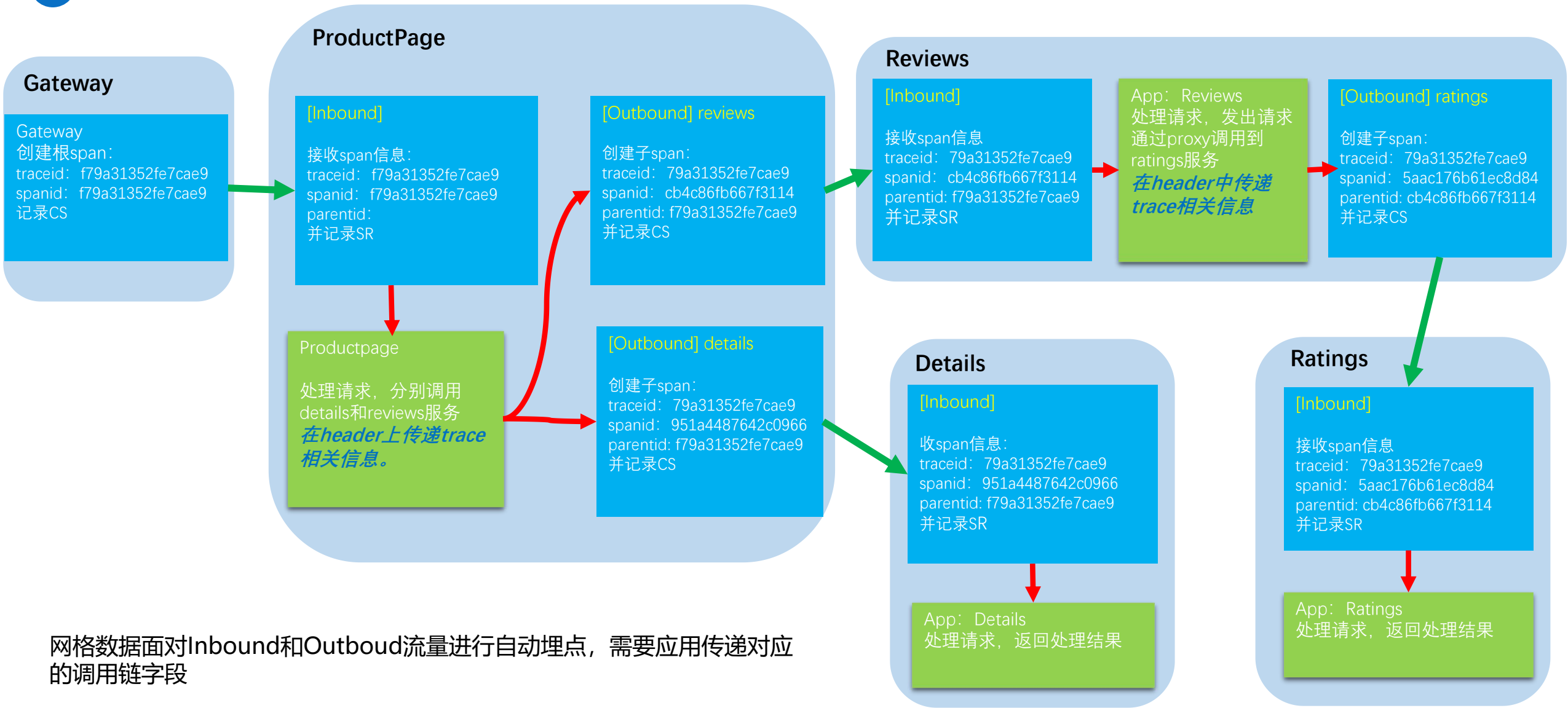


v1.1



Istio 调用链埋点原理剖析—是否真的“零修改”？

Istio调用链埋数据面埋点剖析



网格数据面对Inbound和Outboud流量进行自动埋点，需要应用传递对应的调用链字段

调用链架构原理(Telemetry V1)

1.网格数据面拦截流量，并代替应用进行调用链埋点

2.数据面埋点数据tracespan上报Mixer

3.Mixer分发调用链数据给Zipkin Adapter

4.Adapter写数据到调用链后端服务Zipkin或Jaeger等。

调用链配置实践(Telemetry V1)

msup

Zipkin handler 配置

```
apiVersion: "config.istio.io/v1alpha2"
kind: handler
metadata:
  name: zipkin
  namespace: istio-system
spec:
  compiledAdapter: zipkin
  params:
    url: "zipkin:9411"
    sampleProbability: 0.01
```

定义Span，配置Span的字段

```
apiVersion: "config.istio.io/v1alpha2"
kind: instance
metadata:
  name: tracespan
  namespace: istio-system
spec:
  compiledTemplate: tracespan
  params:
    severity: '"Default"'
    traceId: request.headers["x-b3-traceid"]
    spanId: request.headers["x-b3-spanid"] | ""
    parentSpanId: request.headers["x-b3-parentspanid"] | ""
    traceId: request.headers["x-b3-traceid"]
    spanId: request.headers["x-b3-spanid"] | ""
    parentSpanId: request.headers["x-b3-parentspanid"] | ""
    spanName: request.path | "/"
    startTime: request.time
    endTime: response.time
    clientSpan: (context.reporter.kind | "inbound") == "inbound"
    rewriteClientSpanId: false
    spanTags:
      http.method: request.method | ""
      http.status_code: response.code | 200
      http.url: request.path | ""
      request.size: request.size | 0
      response.size: response.size | 0
      source.principal: source.principal | ""
      source.version: source.labels["version"] | ""
```

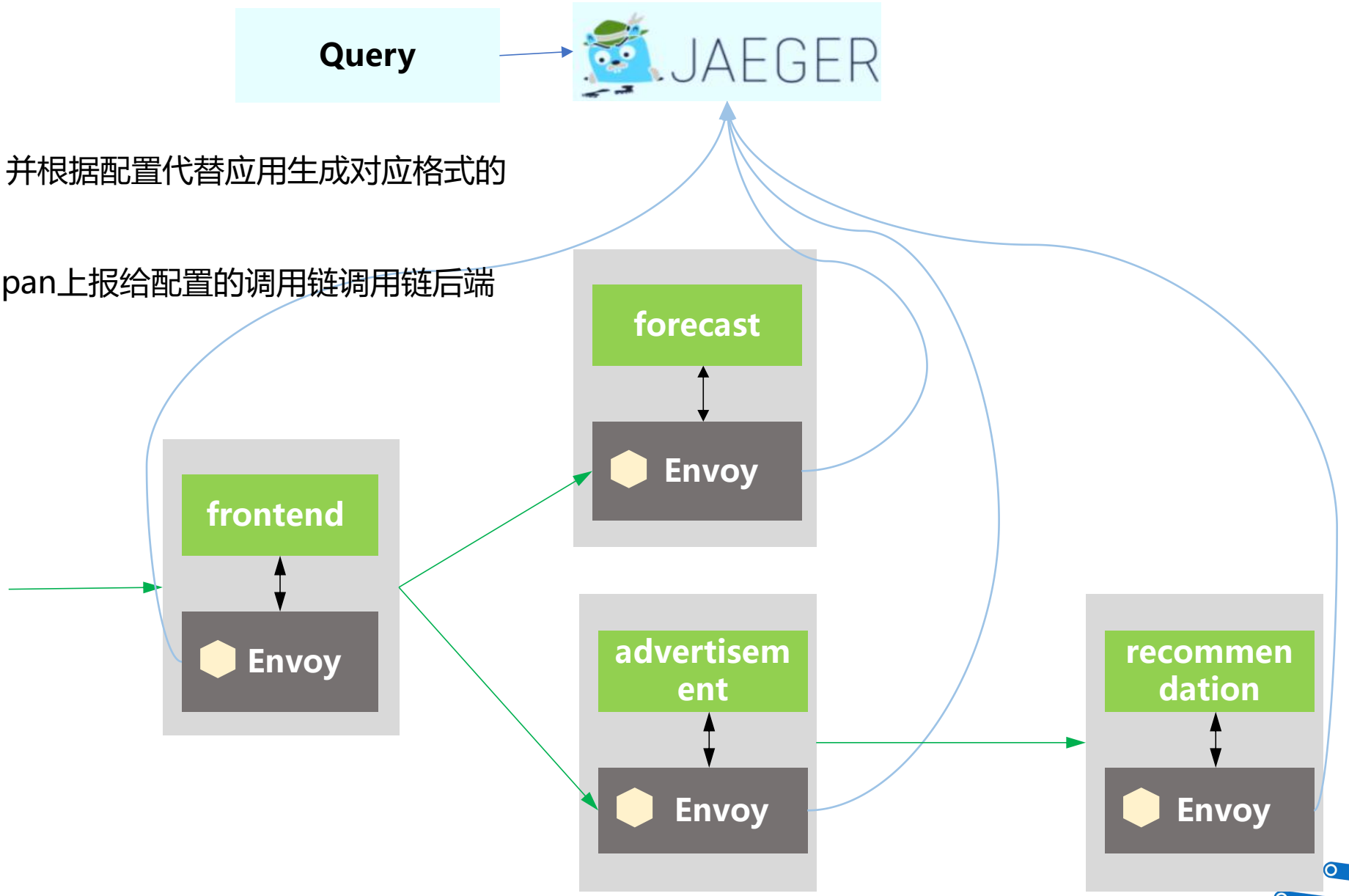
关联Hander和Instance

```
apiVersion: "config.istio.io/v1alpha2"
kind: rule
metadata:
  name: zipkin
  namespace: istio-system
spec:
  actions:
   - handler: zipkin
     instances:
     - tracespan
```

调用链架构原理(Telemetry V2)

1.网格数据面拦截流量，并根据配置代替应用生成对应格式的调用链埋点数据

2.数据面埋点数据tracespan上报给配置的调用链调用链后端



调用链配置实践(Telemetry V2)

msup

调用链生成和上报配置

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    enableTracing: true
    defaultConfig:
      tracing:
        tlsSettings:
          caCertificates: "/var/run/secrets/cloud/ca.crt"
          mode: SIMPLE
        zipkin:
          address: cloud-receiver.com:32677
        sampling: 1.0
        max_path_tag_length: 256
        custom_tags:
          cluster_id:
            environment:
              defaultValue: unknown
              name: ISTIO_META_CLUSTER_ID
```

Span的字段扩展

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    enableTracing: true
    defaultConfig:
      tracing:
        sampling: 1.0
        max_path_tag_length: 256
        custom_tags:
          cluster_id:
            environment:
              defaultValue: unknown
              name: ISTIO_META_CLUSTER_ID
          group:
            header:
              defaultValue: unknown
              name: group
          workload:
            environment:
              defaultValue: unknown
              name: ISTIO_META_WORKLOAD_NAME
          mesh_id:
            literal:
              value: 88888888888888888888
```

Istio支持多种不同的调用链类型，不同的类型的调用链有不同的配置。

目录

- 非侵入可扩展架构特点
- Metric 架构原理和实践
- Tracing 架构原理和实践
- AccessLog 架构原理和实践
- 发展和规划

访问日志架构原理(Telemetry V1)

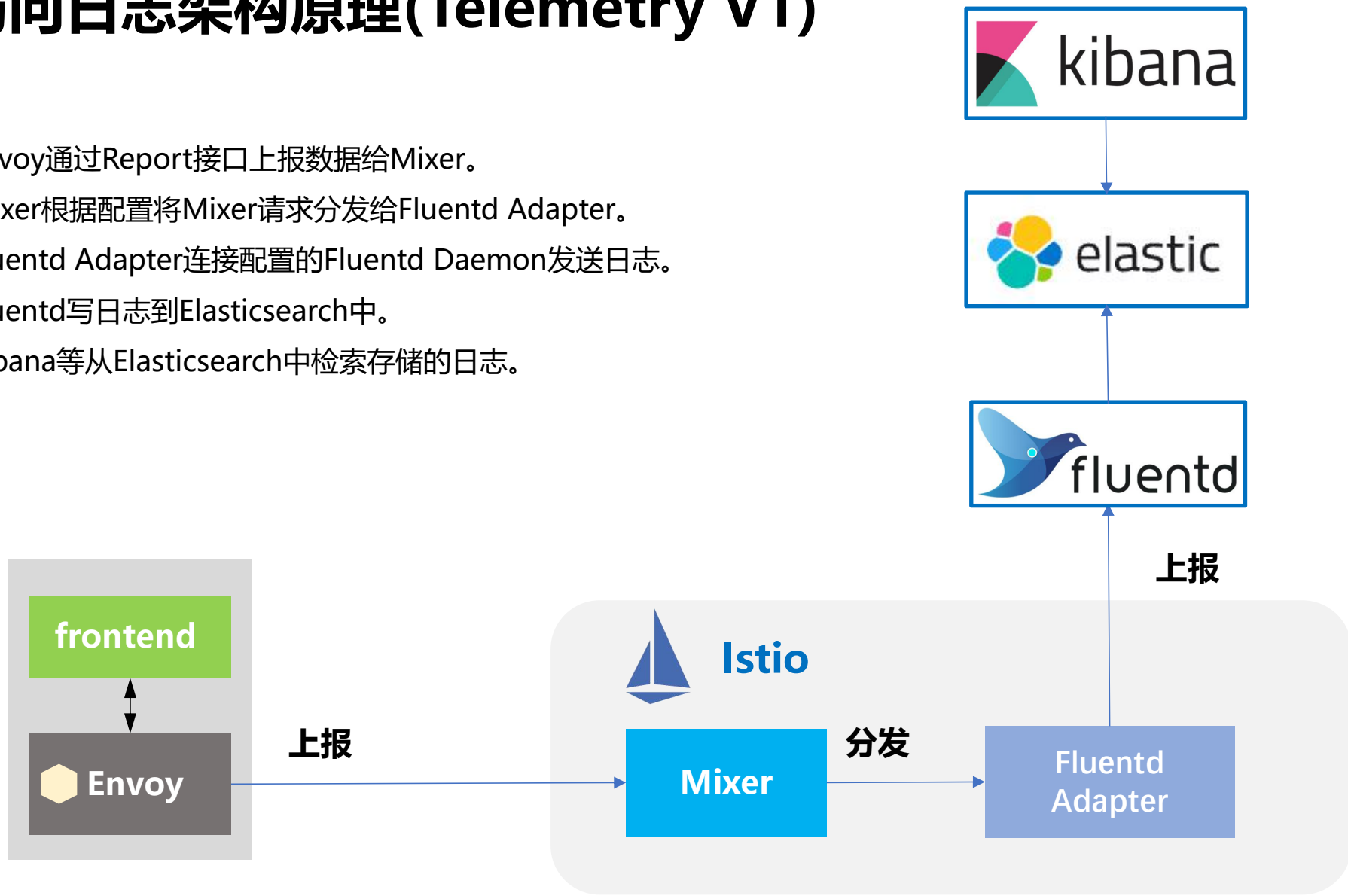
1.Envoy通过Report接口上报数据给Mixer。

2.Mixer根据配置将Mixer请求分发给Fluentd Adapter。

3.Fluentd Adapter连接配置的Fluentd Daemon发送日志。

4.Fluentd写日志到Elasticsearch中。

5.Kibana等从Elasticsearch中检索存储的日志。



访问日志配置实践(Telemetry V1)

msup

Fluentd handler 配置

```
apiVersion: "config.istio.io/v1alpha2"
kind: handler
metadata:
  name: fluentd
  namespace: istio-system
spec:
  compiledAdapter: fluentd
  params:
    address: "fluentd:24224"
```

定义日志，配置日志的字段

```
apiVersion: "config.istio.io/v1alpha2"
kind: instance
metadata:
  name: logentry
  namespace: istio-system
spec:
  compiledTemplate: logentry
  params:
    severity: '"Default"'
    timestamp: request.time
    variables:
      sourceIp: source.ip | ip("0.0.0.0")
      destinationIp: destination.ip | ip("0.0.0.0")
      sourceUser: source.principal | ""
      method: request.method | ""
      url: request.path | ""
      protocol: request.scheme | "http"
      responseCode: response.code | 0
      responseSize: response.size | 0
      requestSize: request.size | 0
      latency: response.duration | "0ms"
    monitored_resource_type: '"UNSPECIFIED"'
```

关联Hander和Instane

```
apiVersion: "config.istio.io/v1alpha2"
kind: rule
metadata:
  name: logfluentd
  namespace: istio-system
spec:
  match: "true" # match for all requests
  actions:
   - handler: fluentd
     instances:
     - logentry
```

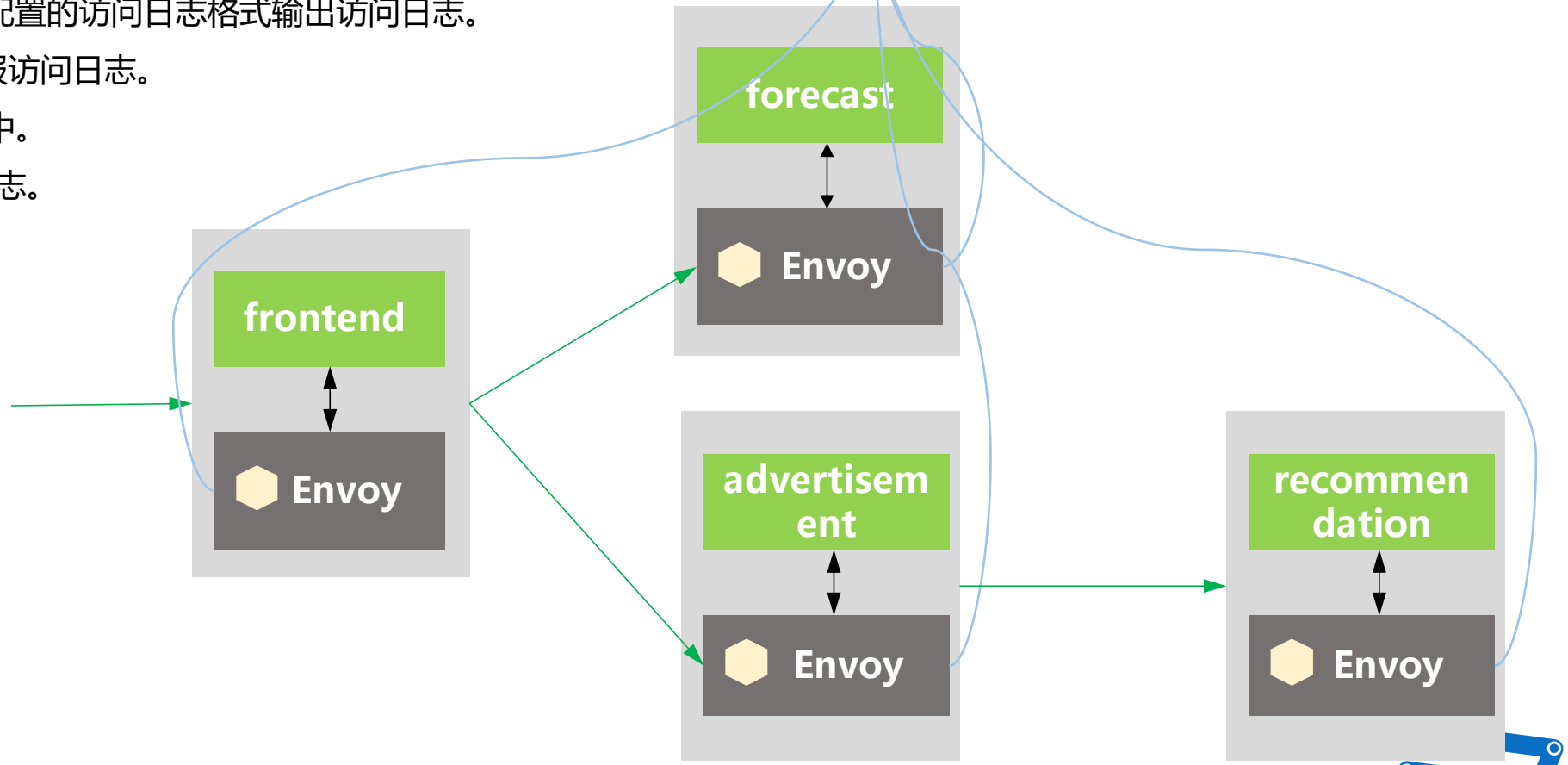
访问日志架构原理(Telemetry V2)

1.网格数据面拦截流量，并根据配置的访问日志格式输出访问日志。

2.数据面根据配置的ALS地址上报访问日志。

3.ALS将日志Export到ES等存储中。

4.Kibana等从ES中检索存储的日志。



访问日志架构原理(Telemetry V2)

msup

配置访问日志输出到文件

```
spec:
  meshConfig:
    accessLogEncoding: TEXT
    accessLogFile: "/var/log/istio/default_access.log"
    accessLogFormat: ""
```

配置访问日志输出到日志服务

```
spec:
  meshConfig:
    enableEnvoyAccessLogService: true
    defaultConfig:
      envoyAccessLogService:
        address: accesslog-collector:9090
```

访问日志默认格式

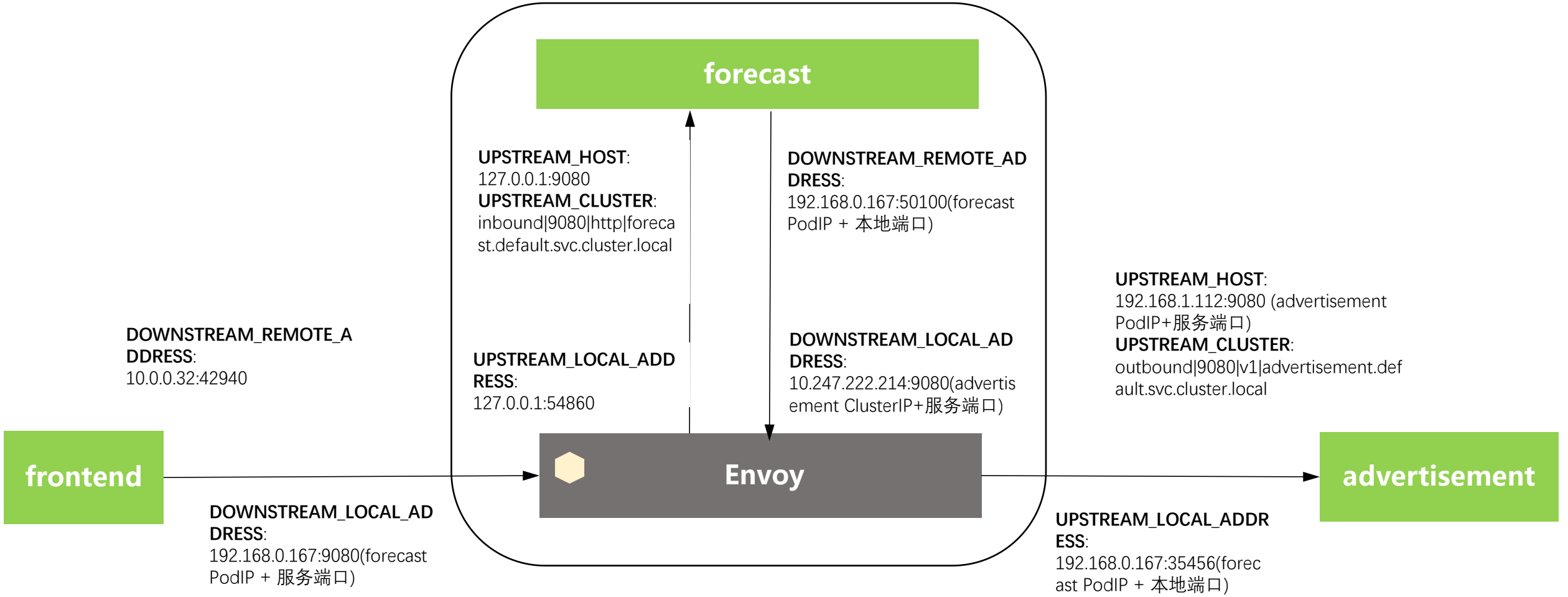
```
[%START_TIME%] "%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-PATH?:PATH)% %PROTOCOL%" %RESPONSE_CODE%
%RESPONSE_FLAGS% %BYTES_RECEIVED% %BYTES_SENT% %DURATION% %RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)%
"%REQ(X-FORWARDED-FOR)%" "%REQ(USER-AGENT)%" "%REQ(X-REQUEST-ID)%" "%REQ(:AUTHORITY)%"
"%UPSTREAM_HOST%"\n
```

访问日志解析(1)：字段解析

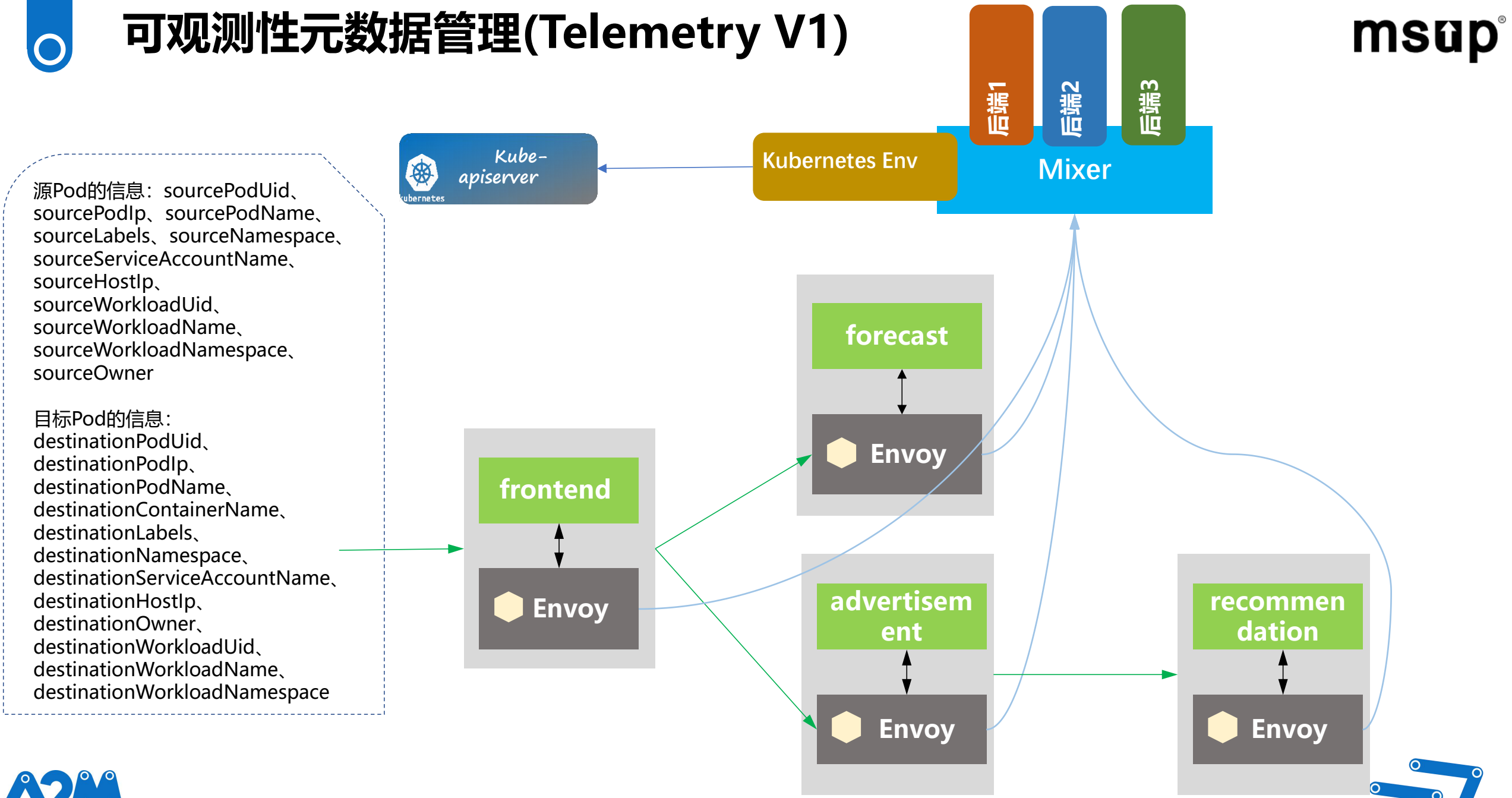
msup

字段名	字段说明	Inbound日志示例	Outbound日志示例
START_TIME	开始时间	[2021-05-26T03:48:10.784Z]	[2021-05-26T03:49:16.678Z]
"%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-PATH?:PATH)% %PROTOCOL%\"	从Request头域提取METHOD、PATH等信息，并拼接访问协议	GET /forecast HTTP/1.1	GET /advertisement HTTP/1.1
RESPONSE_CODE	HTTP响应码	200	200
RESPONSE_FLAGS	响应标记	-	-
UPSTREAM_TRANSPORT_FAILURE_REASON	上游传输失败的原因	-	-
BYTES_RECEIVED	HTTP协议，表示应答Body大小；TCP协议，表示连接上收到的字节数。	0	0
BYTES_SENT	HTTP协议表示发送的Body大小	379	48
DURATION	HTTP协议，表示处理一个请求的时间	2889	78
RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)	从Response中提取的X-ENVOY-UPSTREAM-SERVICE-TIME。表示从下游连接完成开始，到收到上游连接第一个字节所经过的时间	2887	76
REQ(X-FORWARDED-FOR)	从请求头域中提取X-FORWARDED-FOR，HTTP扩展头域，记录请求端真实IP	-	-
REQ(USER-AGENT)	从请求头域中提取的USER-AGENT	python-requests/2.18.4	python-requests/2.18.4
REQ(X-REQUEST-ID)	从Request中提取X-REQUEST-ID头域，用于唯一标识一个请求。	0cd2d0dc-b576-9f16-858f-8256966c6f58	defff25d-7cde-9172-aaa7-01741dc5f398
REQ(:AUTHORITY)	从Request中提取的AUTHORITY	forecast:9080	advertisement:9080
UPSTREAM_HOST	上游主机的URL	127.0.0.1:9080	192.168.1.112:9080
UPSTREAM_CLUSTER	上游集群名	inbound|9080|http|forecast.default.svc.cluster.local	outbound|9080|v1|advertisement.default.svc.cluster.local
UPSTREAM_LOCAL_ADDRESS	上游连接的本地地址	127.0.0.1:54860	192.168.0.167:35456
DOWNSTREAM_LOCAL_ADDRESS	下游连接本地地址，	192.168.0.167:9080	10.247.222.214:9080
DOWNSTREAM_REMOTE_ADDRESS	下游连接远端地址	10.0.0.32:42940	192.168.0.167:50100
REQUESTED_SERVER_NAME		-	-
ROUTE_NAME		Default	-

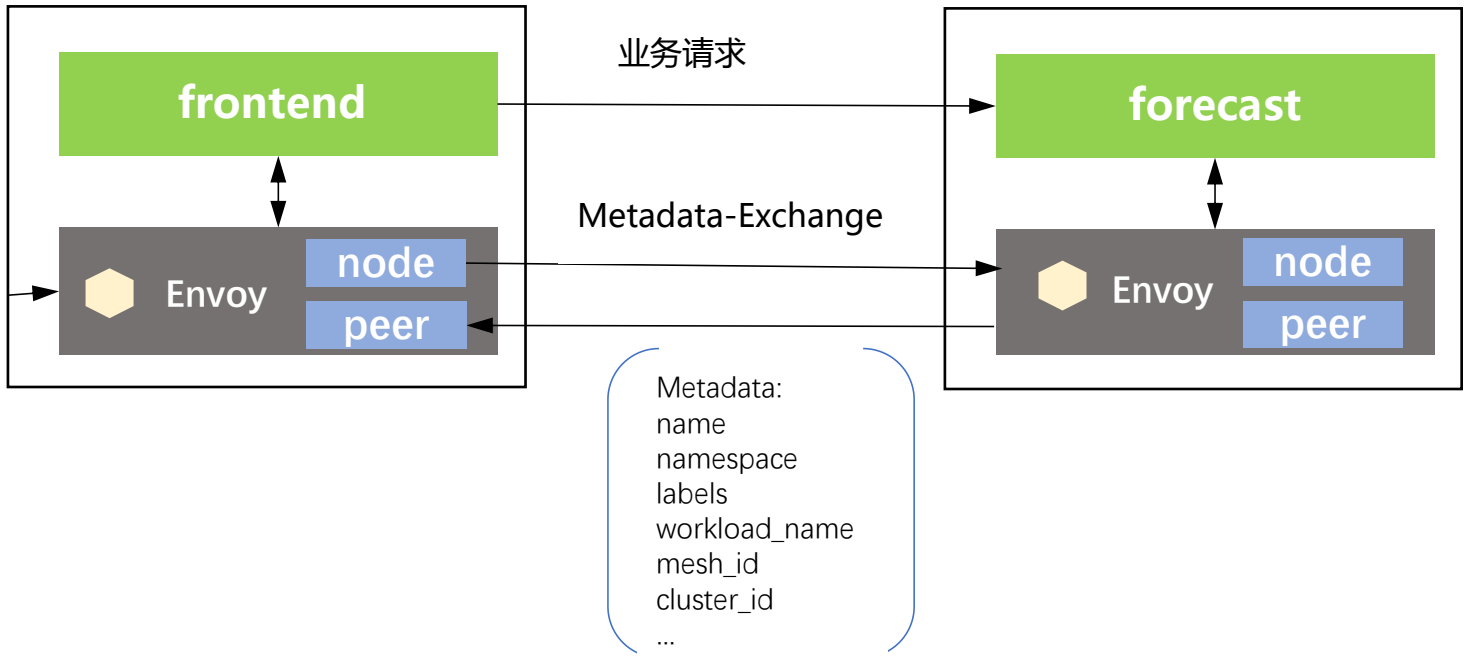
访问日志解析(2)：字段解析



可观测性元数据管理(Telemetry V1)



可观测性元数据管理(Telemetry V2)



Field	Type
name	string
namespace	string
labels	map
owner	string
workload_name	string
platform_metadata	map
istio_version	string
mesh_id	string
app_containers	list<string>
cluster_id	string

请求数据："request_url": "request.url_path",

本地元数据："source_pod": "node.metadata['NAME']",

交换元数据：**"destination_pod": "upstream_peer.name"**

请求数据："request_url": "request.url_path",

本地元数据：**"source_pod": "downstream_peer.name"**

交换元数据："destination_pod": "node.metadata['NAME']"

总结

msup

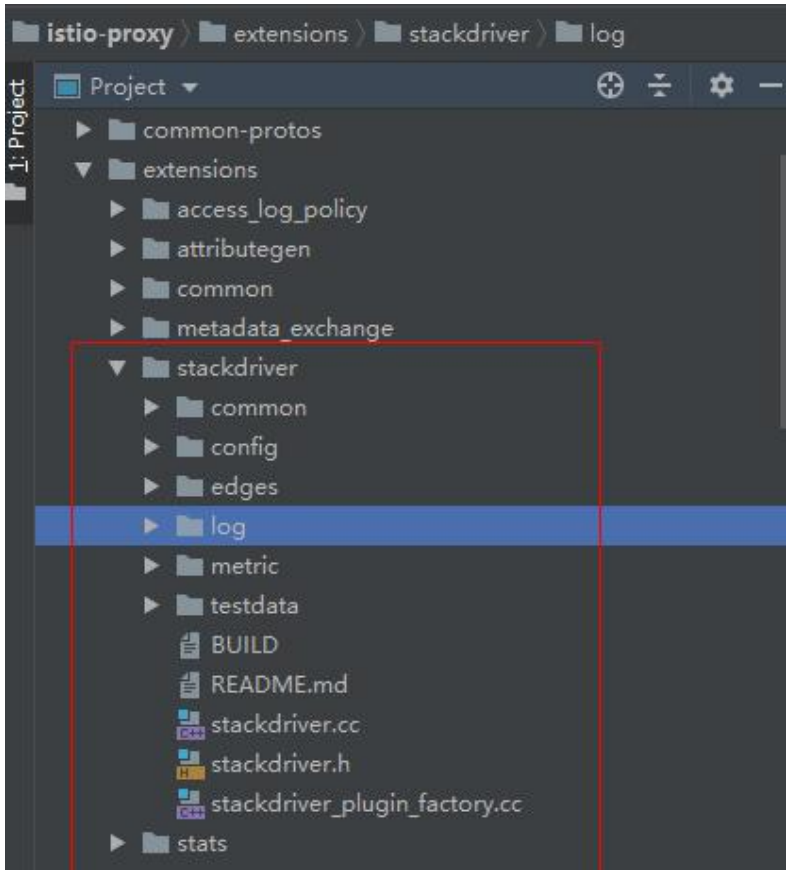
	Telemetry V1	Telemetry V2
扩展方式	Mixer adapter	Envoy filter/ Meshconfig
Metadata	Kubernetes env adapter	Metadata exchange
Metric 采集	抓取Prometheus adapter exporter	抓取Envoy exporter
Tracing 采集	可以直接报给调用链后端，也可以通过tracing的mixer adapter	只能通过Envoy上报给Tracing后端
限流	使用Mixer的check在服务端有Mem quota和Redis quota	Envoy本地限流和Envoy连接限流后端限流服务的全局限流
配置定义	Mixer Instance	Envoy filter/ Meshconfig
采集规则	Mixer Rule	Envoy filter/ Meshconfig
扩展性	强	一般
可配置性	强	一般

目录

- 非侵入可扩展架构特点
- Metric 架构原理和实践
- Tracing 架构原理和实践
- AccessLog 架构原理和实践
- 发展和规划

Istio 可观测性其他机制

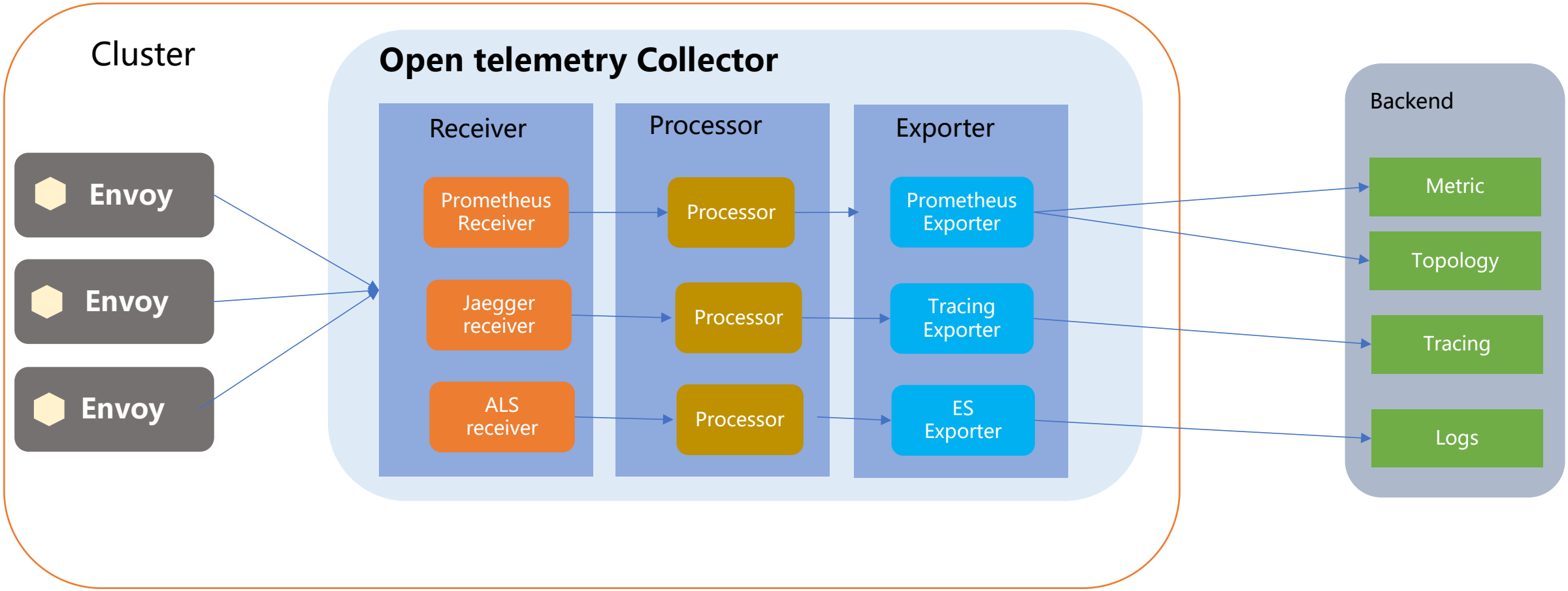
在数据面扩展Filter实现，如extensions/stackdriver/stackdriver.cc
在Istio-proxy中扩展Filter实现，定期上报Metric和访问日志，而不是Istio标准的Prometheus采集Metric或基于Envoy的访问日志机制。




```
metric:
  labels:
    destination_canonical_revision: version-1
    destination_canonical_service_name: ratings
    destination_canonical_service_namespace: default
    destination_owner: kubernetes://apis/apps/v1/namespaces/default/deployments/ratings-v
    destination_port: '{{ .Ports.ServerPort }}'
    {{- if .Vars.DestinationPrincipal }}
    destination_principal: "{{ .Vars.DestinationPrincipal }}"
    {{- else }}
    destination_principal: unknown
    {{- end }}
    destination_service_name: server
    destination_service_namespace: default
    destination_workload_name: ratings-v1
    destination_workload_namespace: default
    mesh_uid: proj-123
    request_operation: GET
    request_protocol: http
    response_code: "200"
    service_authentication_policy: unknown # TODO: upstream TLS indicator is not reported
    source_canonical_revision: version-1
    source_canonical_service_name: productpage-v1
    source_canonical_service_namespace: default
    source_owner: kubernetes://apis/apps/v1/namespaces/default/deployments/productpage-v1
    {{- if .Vars.SourcePrincipal }}
    source_principal: "{{ .Vars.SourcePrincipal }}"
    {{- else }}
    source_principal: unknown
    {{- end }}
    source_workload_name: productpage-v1
    source_workload_namespace: default
  type: istio.io/service/client/request_count
points:
- value:
    int64Value: "10"
```

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: stackdriver-filter
spec:
  configPatches:
    - applyTo: HTTP_FILTER
      match:
        context: SIDECAR_OUTBOUND
        listener:
          filterChain:
            filter:
              name: "envoy.http_connection_manager"
              subFilter:
                name: "envoy.router"
      patch:
        operation: INSERT_BEFORE
        value:
          name: envoy.filters.http.wasm
          config:
            config:
              root_id: stackdriver_outbound
              configuration: |
                {}
              vm_config:
                vm_id: stackdriver_outbound
                runtime: envoy.wasm.runtime.null
                code:
                  local: { inline_string: envoy.wasm.null.stackdriver
}
```

基于Open telemetry 可观测性数据采集



未来：版本基于 Telemetry 独立API配置

规划在后续版本提供独立的telemetry/v1alpha1 API。

```
message Telemetry {
  // Optional. The selector decides where to apply the Telemetry policy.
  // If not set, the Telemetry policy will be applied to all workloads in the
  // same namespace as the Telemetry policy.
  istio.type.v1beta1.WorkloadSelector selector = 1;

  // Optional. Tracing configures the tracing behavior for all
  // selected workloads.
  repeated Tracing tracing = 2;

  // Optional. Metrics configure the metrics behavior for all
  // selected workloads.
  repeated Metrics metrics = 3;

  // Optional. AccessLogging configures the access logging behavior for all
  // selected workloads.
  repeated AccessLogging access_logging = 4;
}

...
```

https://istio.io/docs/reference/config/telemetry/telemetry.html

关注msup公众号

获取更多AI落地实践

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。