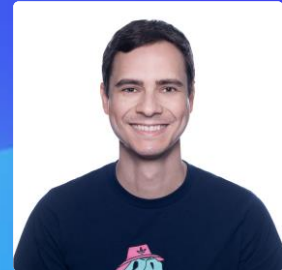


Observability at the Edge: Envoy distributed tracing with OpenTelemetry



PRESENTER

Joao Grassi
Dynatrace

Introduction

- Senior OpenSource Software Engineer @ Dynatrace
- OpenTelemetry semantic conventions maintainer
- Creator of OpenTelemetry recipes (otelrecipes.com)
- Ex .NET back-end developer
- Interested in open source, observability and online privacy



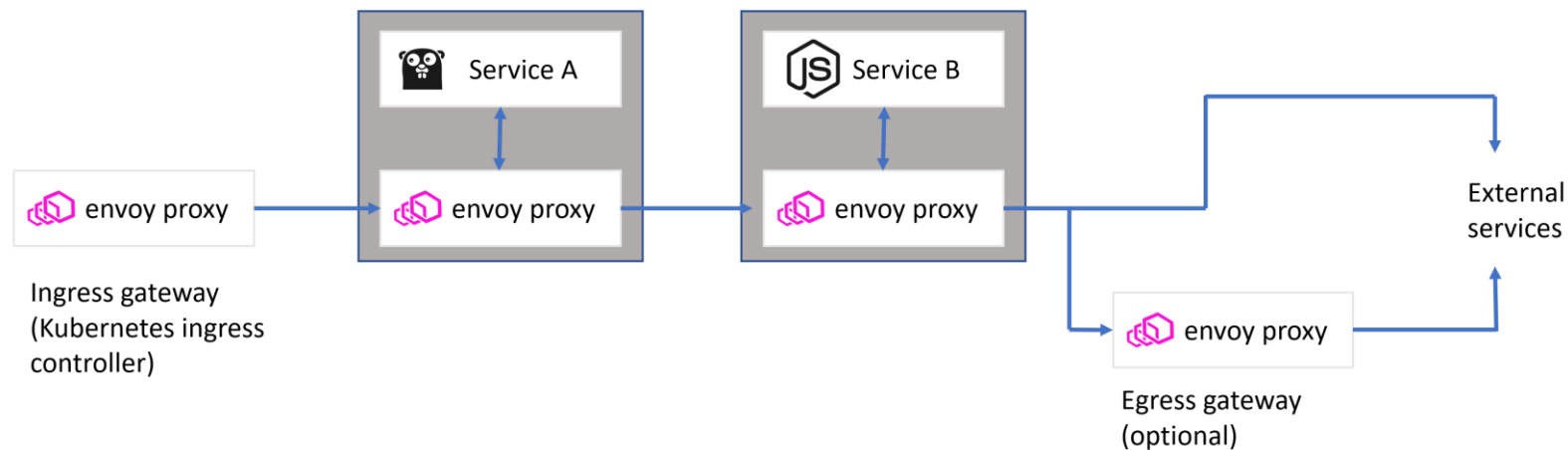
Agenda

- Quick intro to Envoy, Istio and OpenTelemetry
- What was wrong?
- New OTel features in Envoy and Istio
- Overview of tracing configuration in Istio
- Viewing the telemetry (demo) 🙌

What are these?

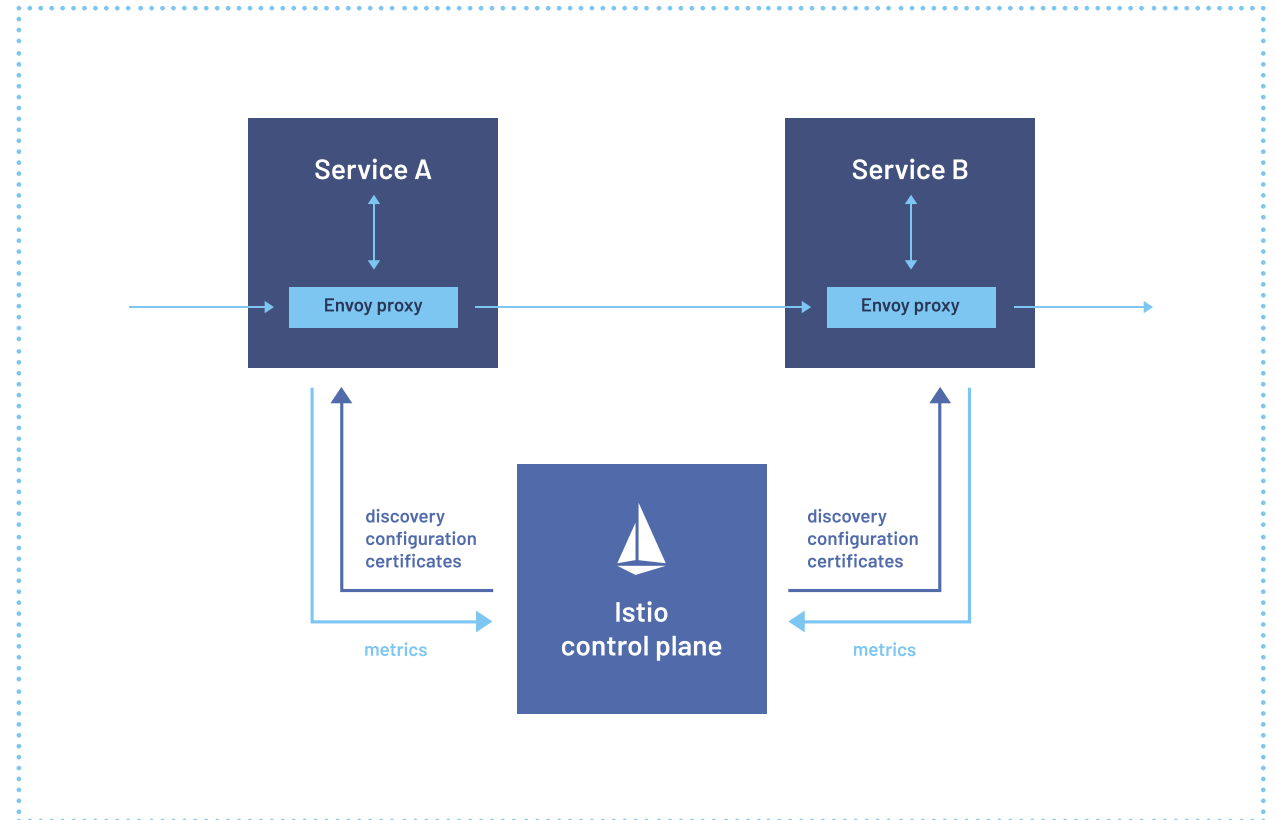


Open-source edge and service proxy, designed for cloud-native applications



What are these?

Open-source service mesh that allows users add capabilities like observability, traffic management, and security, without adding them to your own code



What are these?

OpenTelemetry, also known as OTel, is a vendor-neutral open source [Observability](#) framework for instrumenting, generating, collecting, and exporting telemetry data such as [traces](#), [metrics](#), and [logs](#).



What changed?

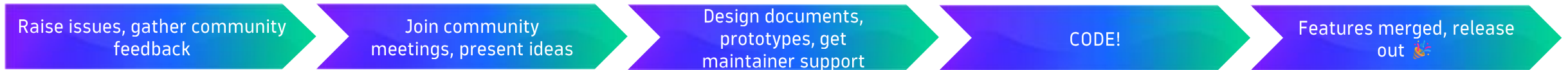
Wait, don't we already have OpenTelemetry tracing support in Envoy?

Limited set of OTel tracing features in Envoy



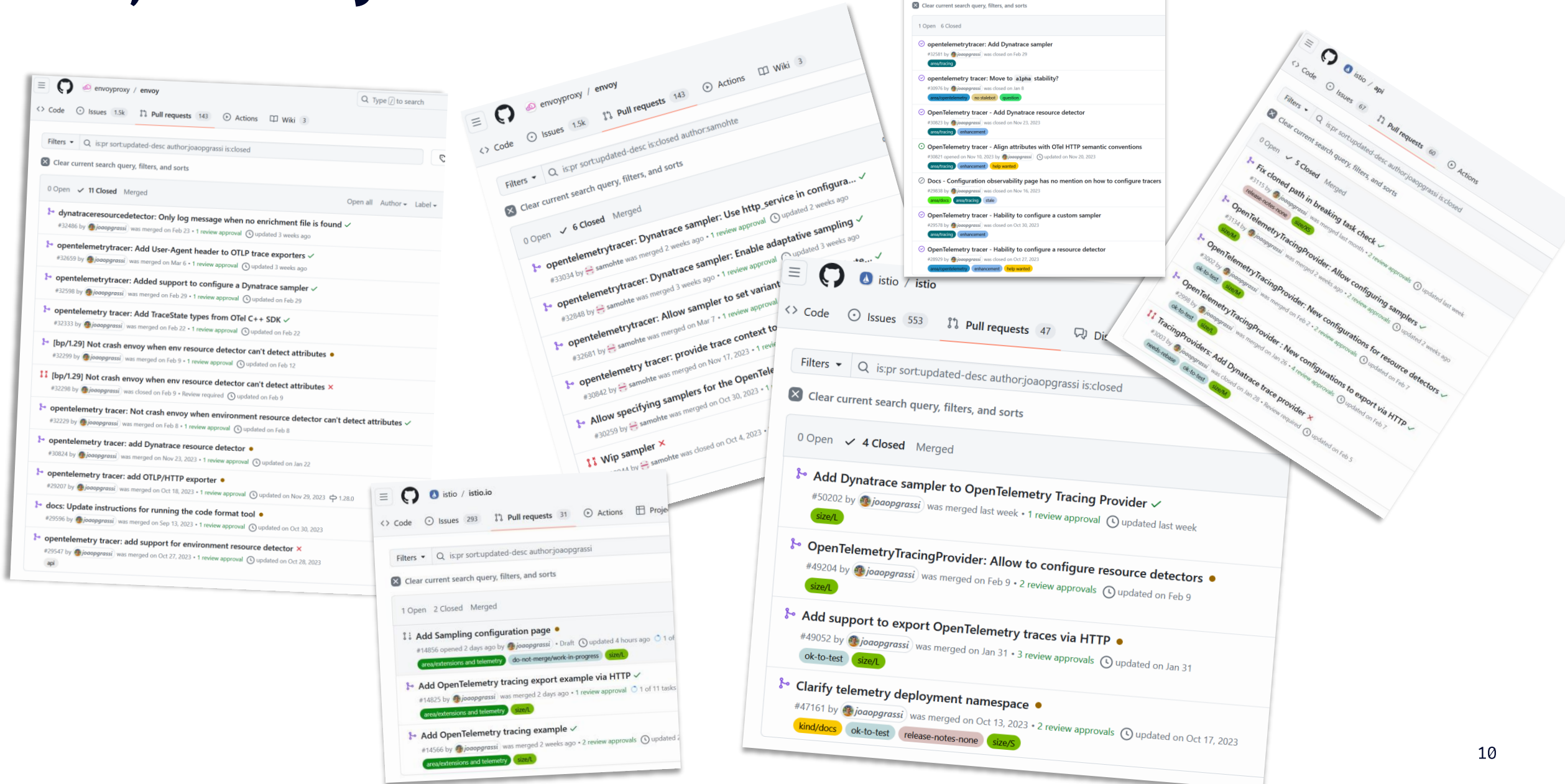
- Only exporting via gRPC
- Cannot enrich spans with resource attributes – Host information, process etc
- Only offers a simple percentage-based sampler

OSS collaboration FTW!



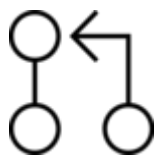
Networking and getting support from community and maintainers was key!

PRs, PRs everywhere



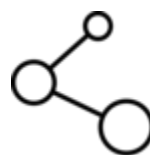
New OTel tracing features in Envoy/Istio

HTTP Exporter



Allows exporting spans via HTTP to Dynatrace

Resource Detector



Allows enrichment of spans with resource attributes

Custom samplers



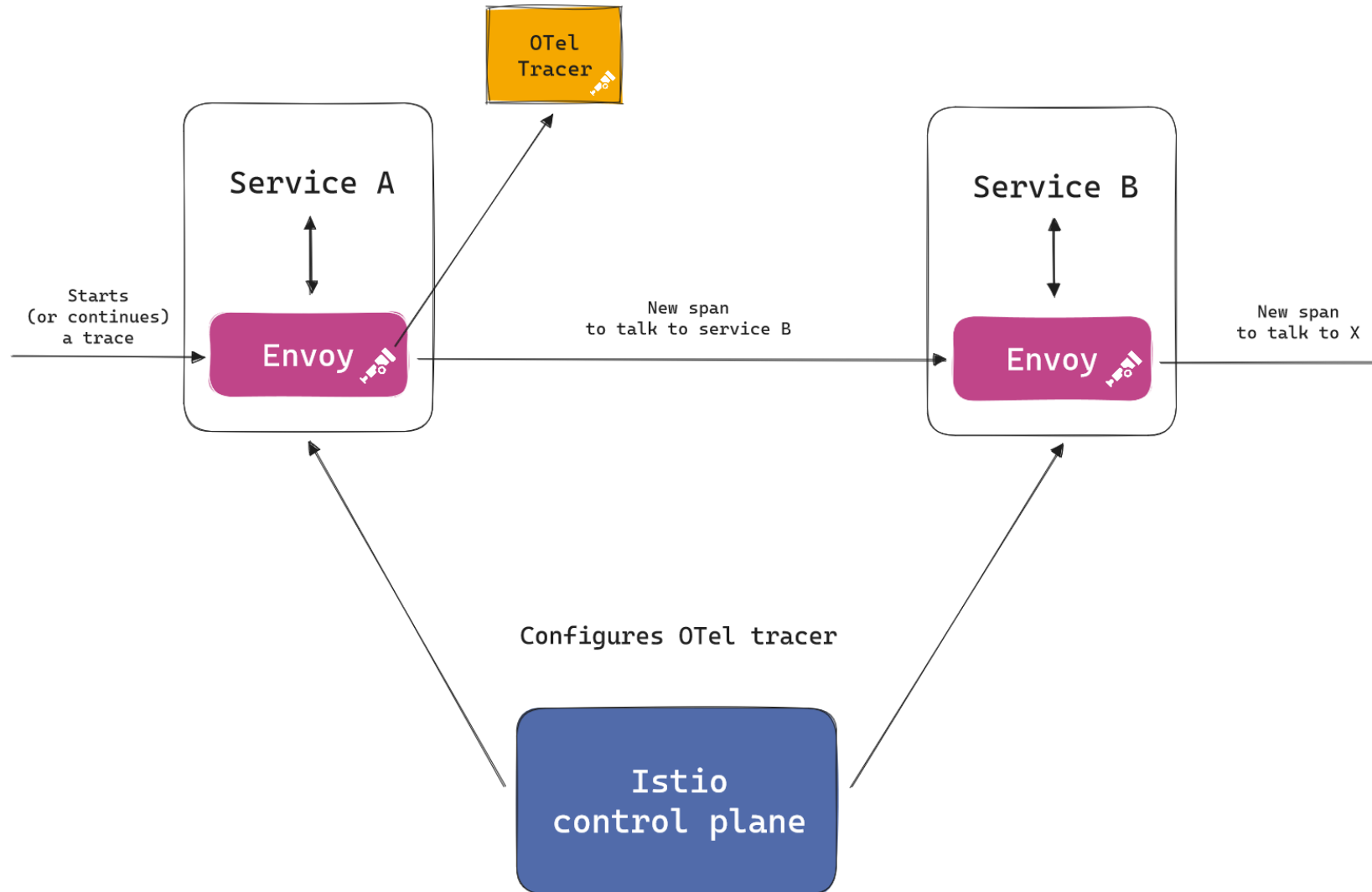
Allows defining more intelligent sampling mechanisms



Istio + Envoy



How are things configured?



Istio Telemetry API

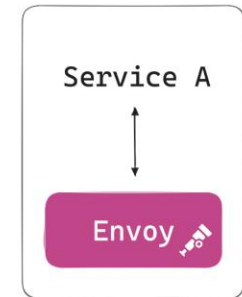
- **Preferred** mechanism for configuring Istio observability going forward
- Work around the concept of **Telemetry Providers**
- **Flexible**: Telemetry can be enabled for the entire mesh, for single namespaces or workloads

Istio Mesh Config

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    enableTracing: true
    extensionProviders:
      - name: otel-tracing
    opentelemetry:
      port: 4318
      service: opentelemetry-collector.observability.svc.cluster.local
      http:
        path: "/v1/traces"
        timeout: 5s
        headers:
          - name: "custom-header"
            value: "custom value"
      resource_detectors:
        environment: {}
```

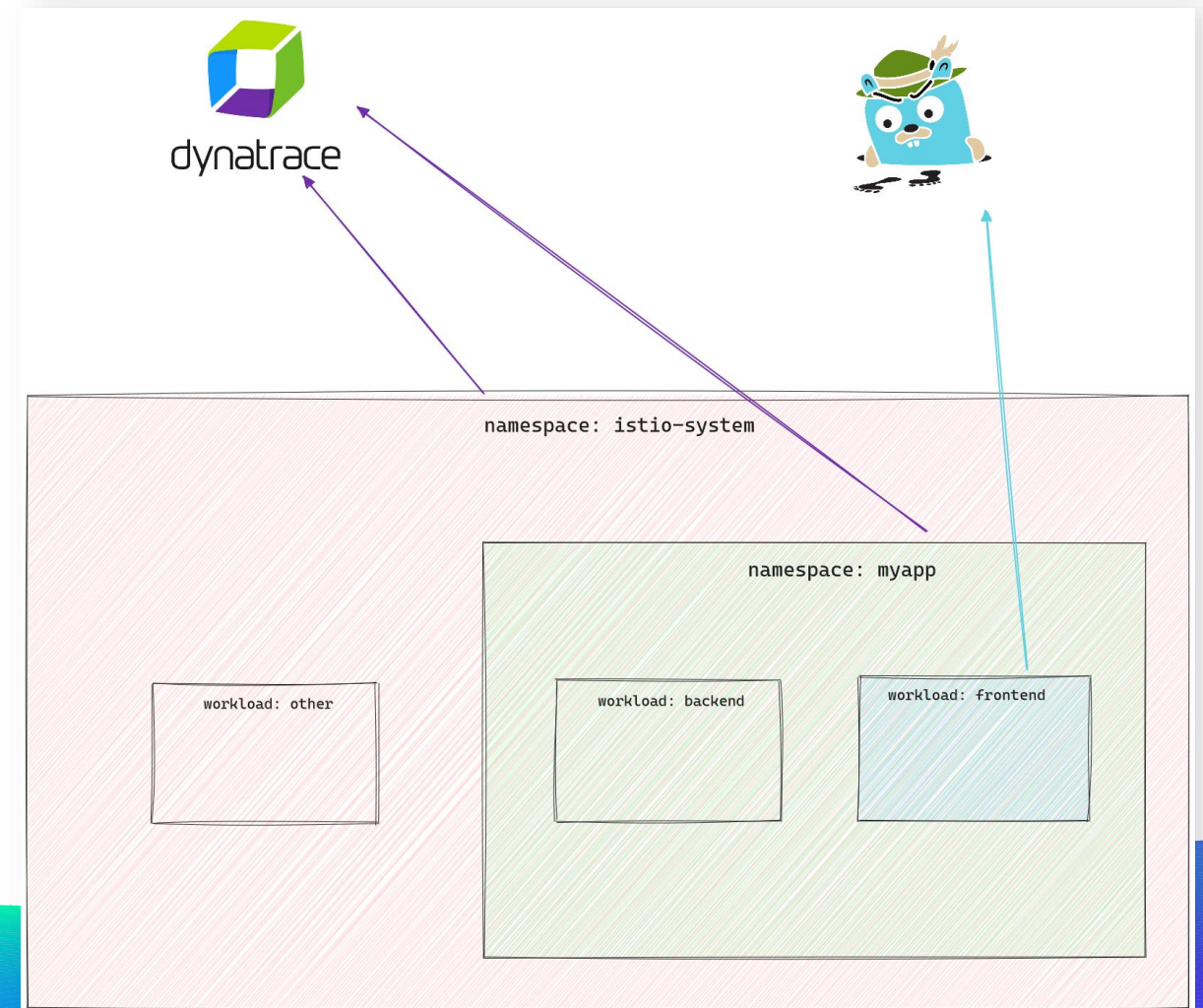
Telemetry API

```
apiVersion: telemetry.istio.io/v1alpha1
kind: Telemetry
metadata:
  name: otel-demo
spec:
  tracing:
    providers:
      - name: otel-tracing
      randomSamplingPercentage: 100
      customTags:
        "my-attribute":
          literal:
            value: "default-value"
```



Flexible tracing configuration via Istio Telemetry API

- Tracing can be configured for different scopes: Entire mesh, per namespace or per workload
- Allows to choose a different tracing “strategy” for each scope
- For example: Send traces to Staging/Prod Dynatrace tenants, or to a different tool, like Jaeger for testing



Flexible tracing configuration via Istio Telemetry API

Namespace scope – Send to Dynatrace



```
apiVersion: telemetry.istio.io/v1alpha1
kind: Telemetry
metadata:
  name: namespace-override
  namespace: myapp
spec:
  tracing:
    - providers:
      - name: dynatrace-otel
```

Workload scope – Send to Jaeger



```
apiVersion: telemetry.istio.io/v1alpha1
kind: Telemetry
metadata:
  name: workload-override
  namespace: myapp
spec:
  selector:
    matchLabels:
      service.istio.io/canonical-name: frontend
  tracing:
    - providers:
      - name: jaeger
```


Questions?

Thank you!