Monitoring & Observability with Istio

Emma - Yang XY Yang/China/IBM
bjyangyy@cn.ibm.com

About me

杨洋

bjyangyy@cn.ibm.com

Advisory Software Engineer
CTO OSS Monitor team, IBM

Why Istio for monitoring?

Micro services

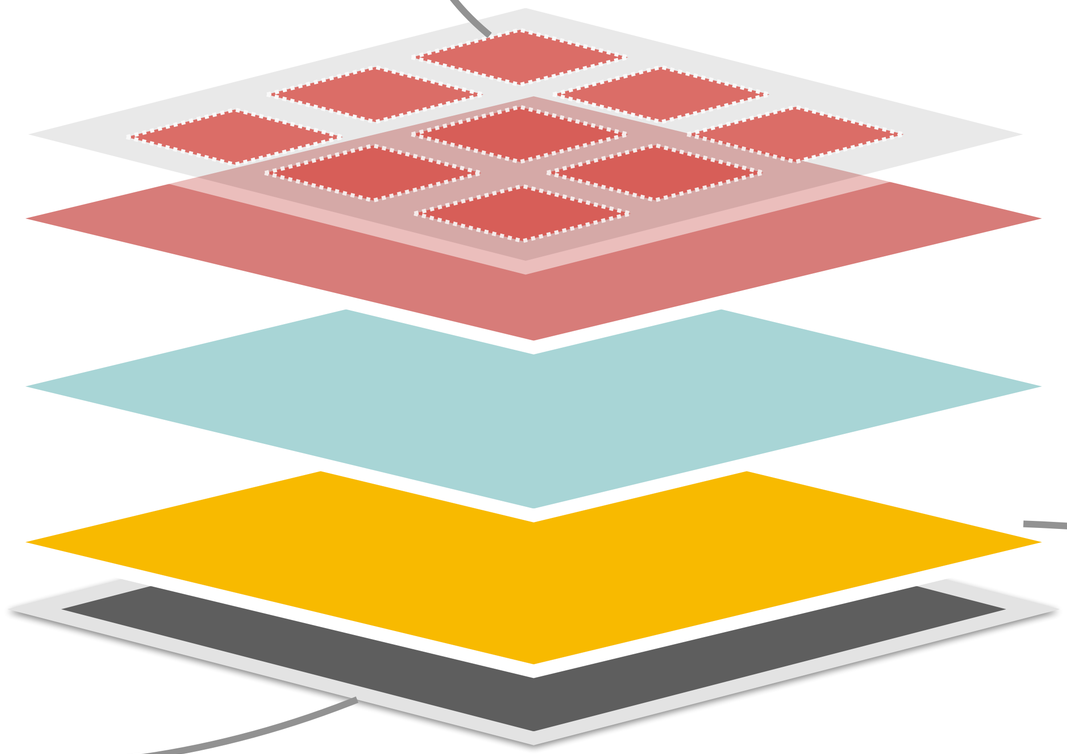
- telemetry — monitoring
- traffic management — governance
- security — TLS termination, certificate management

Services, Runtime & Apps

- availability
- performance
- …

Core Components

- component specific
- eg:
  - database — db2, redis
  - cf component — router
  - web server



Network

- network usage
- abnormal traffic detect
- firewall
- device specific
- …

Infrastructure

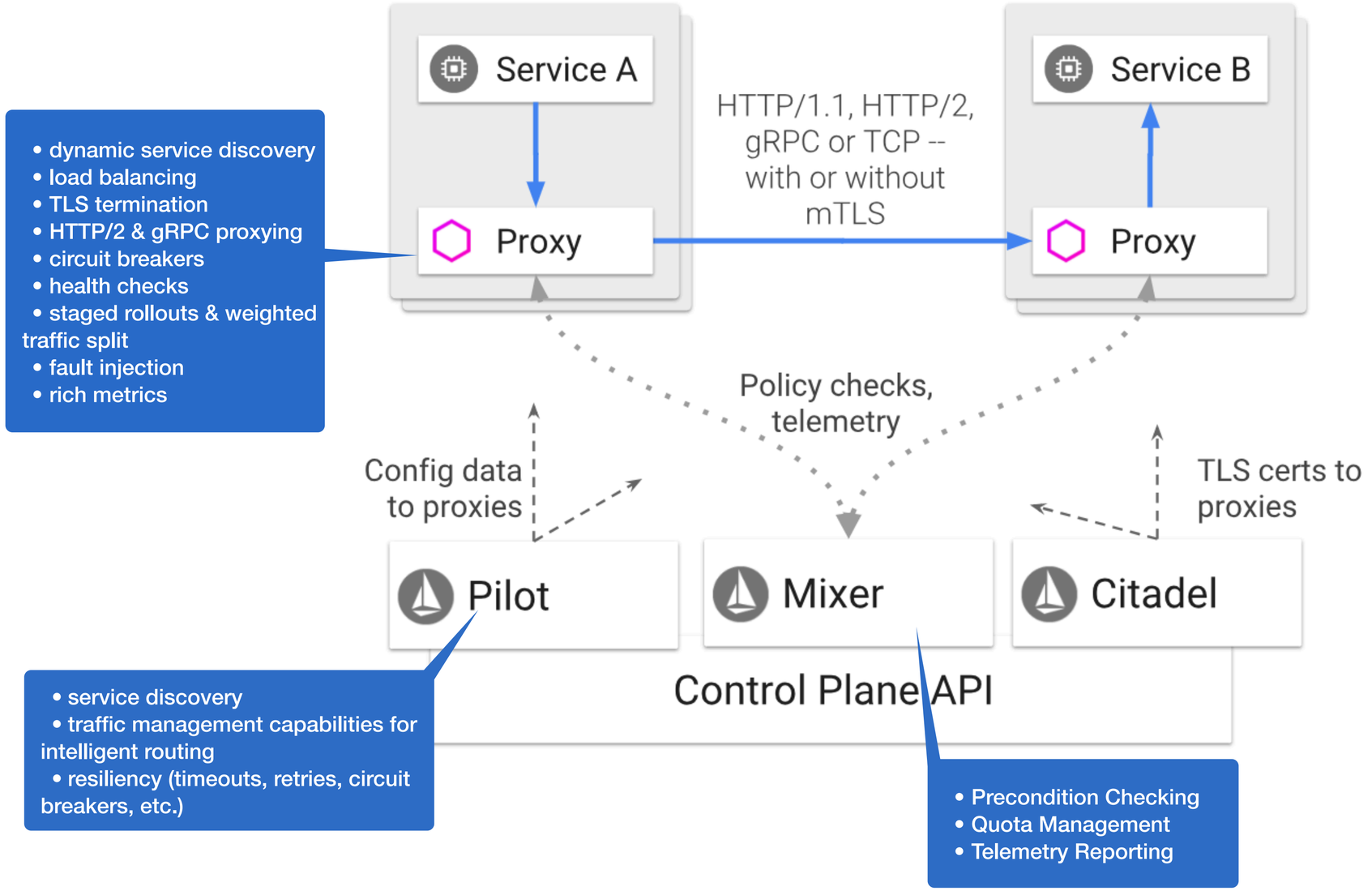
- CPU, memory, disk
- process
- usage
- …

Istio

- **Istio** lets you connect, secure, control, and observe services
- Heavy investments from industry-leading companies such as **Google, IBM, and Lyft**
- Release **1.1**
- Active community

Core Features

- **Telemetry**
- **Fine-grained traffic management**
- **Rich load balancing**
- **Circuit breaking, timeouts, retries**
- **Request Tracing**
- Fault Injection
- Rate limits
- Load shedding

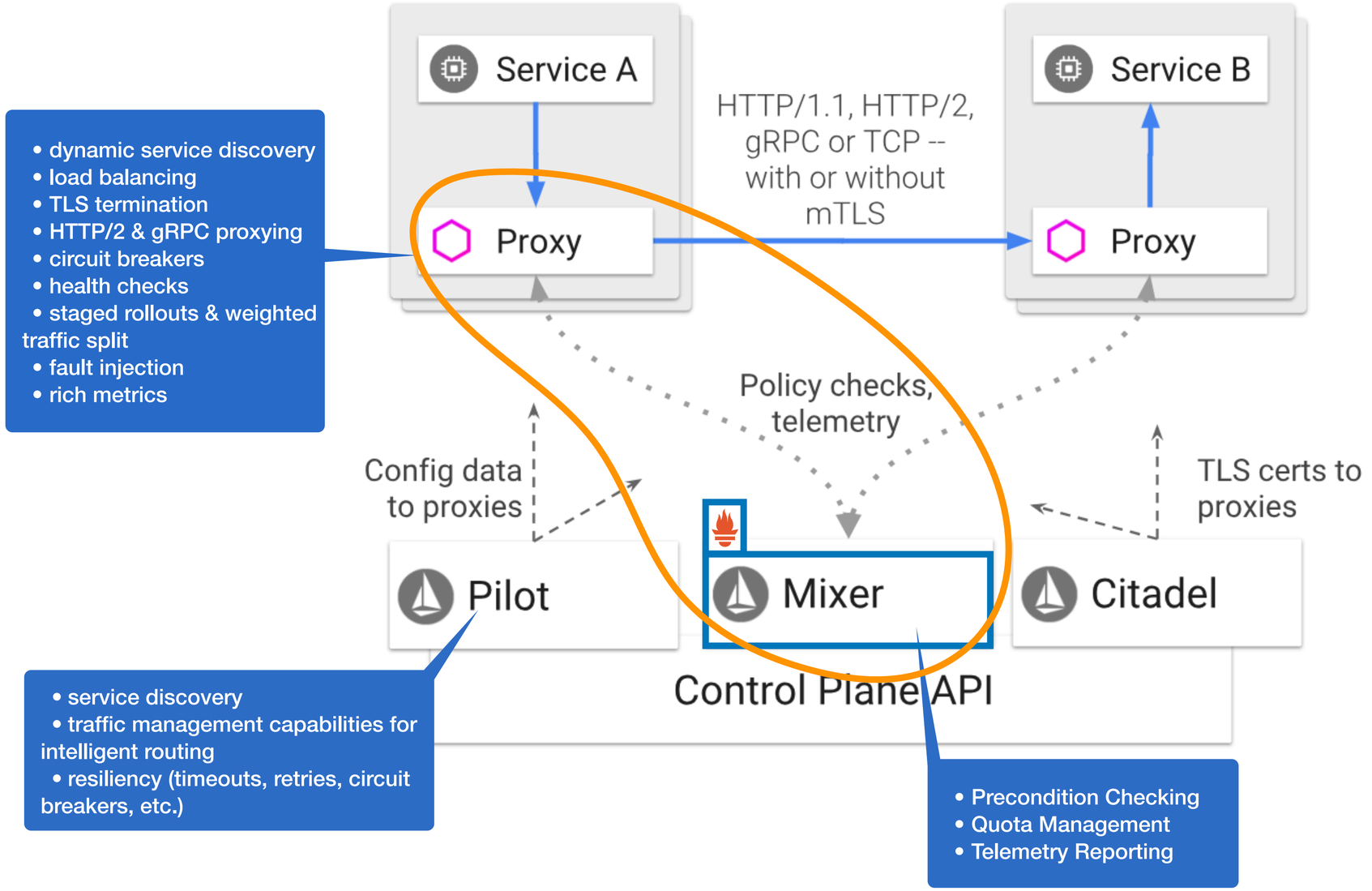


Istio

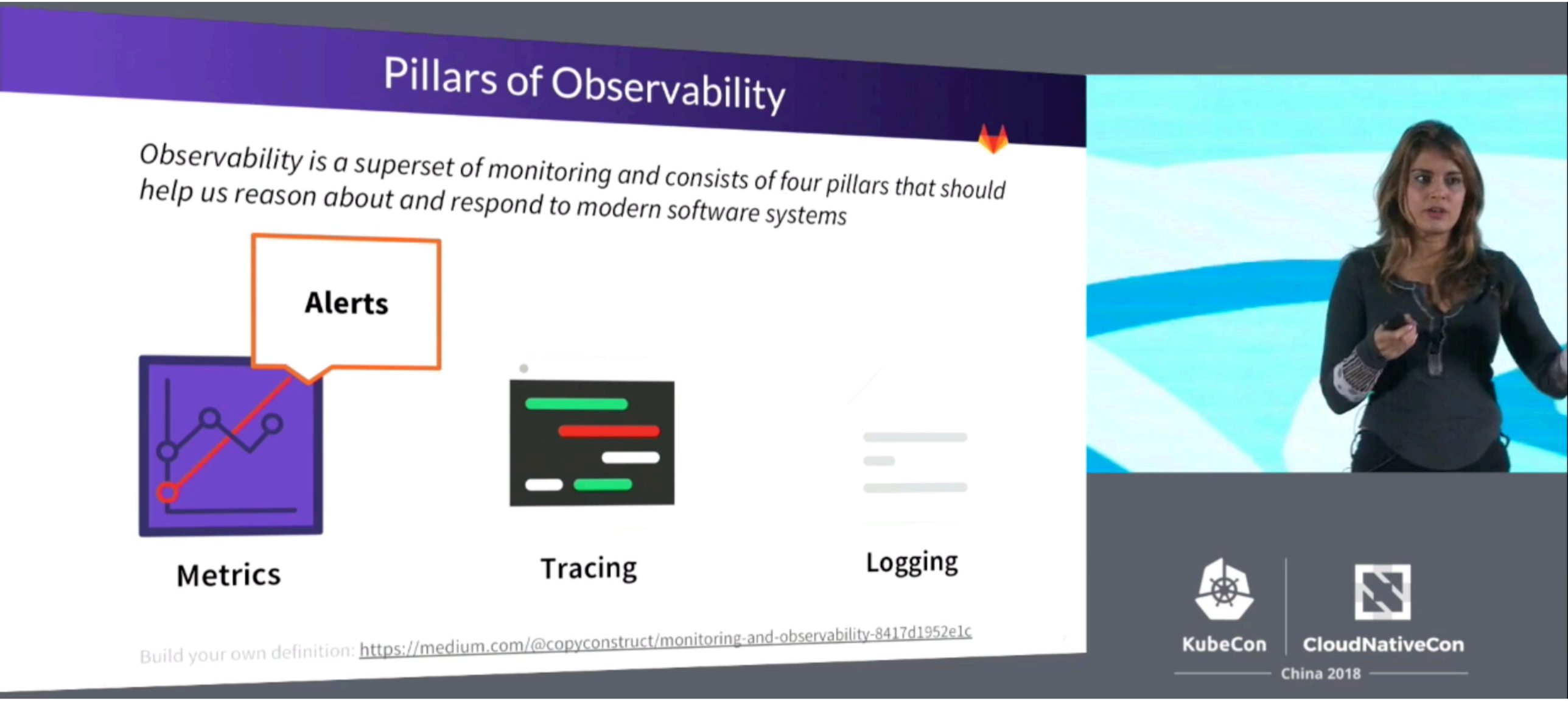
- **Istio** lets you connect, secure, control, and observe services
- Heavy investments from industry-leading companies such as **Google, IBM, and Lyft**
- Release **1.1**
- Active community

Core Features

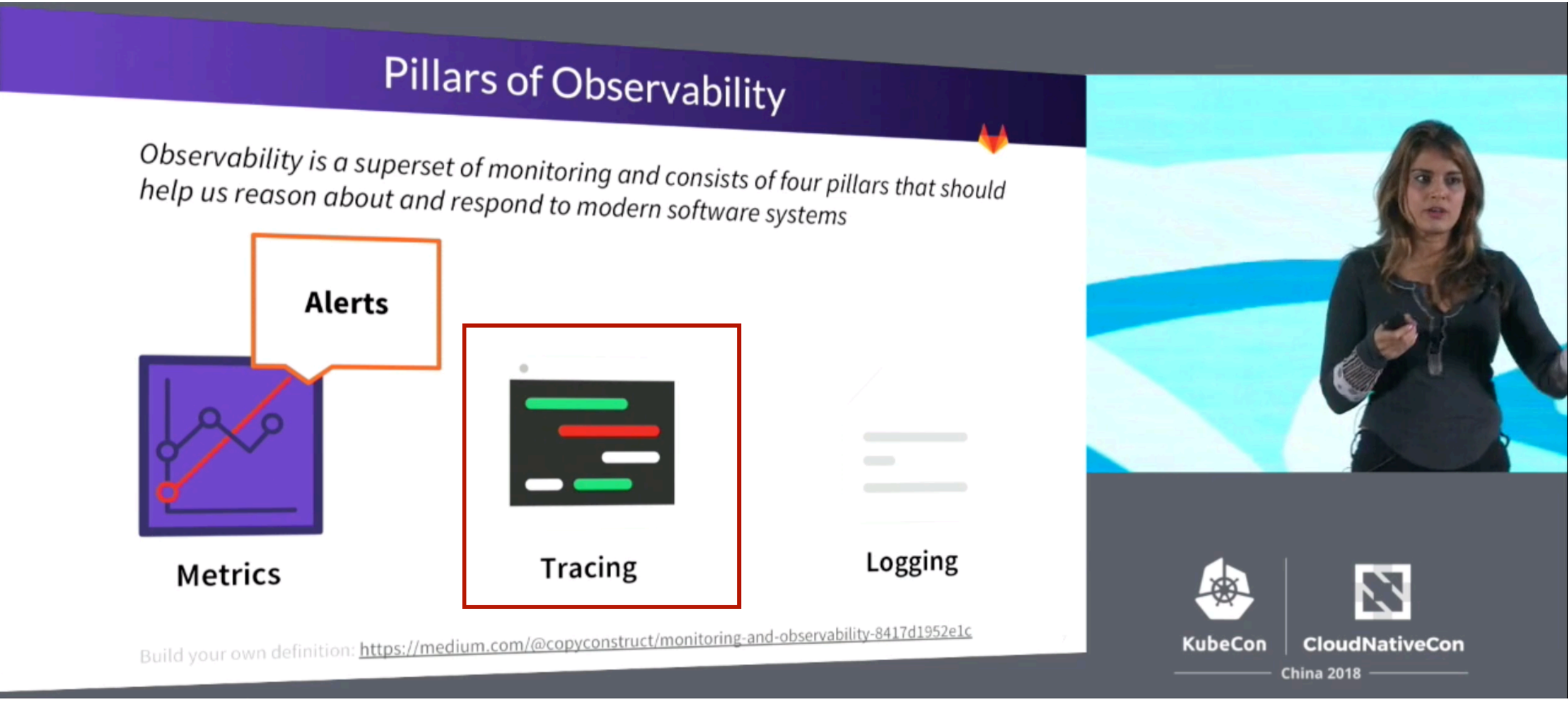
- **Telemetry**
- **Fine-grained traffic management**
- **Rich load balancing**
- **Circuit breaking, timeouts, retries**
- **Request Tracing**
- Fault Injection
- Rate limits
- Load shedding



Monitoring vs. Observability



Monitoring vs. Observability



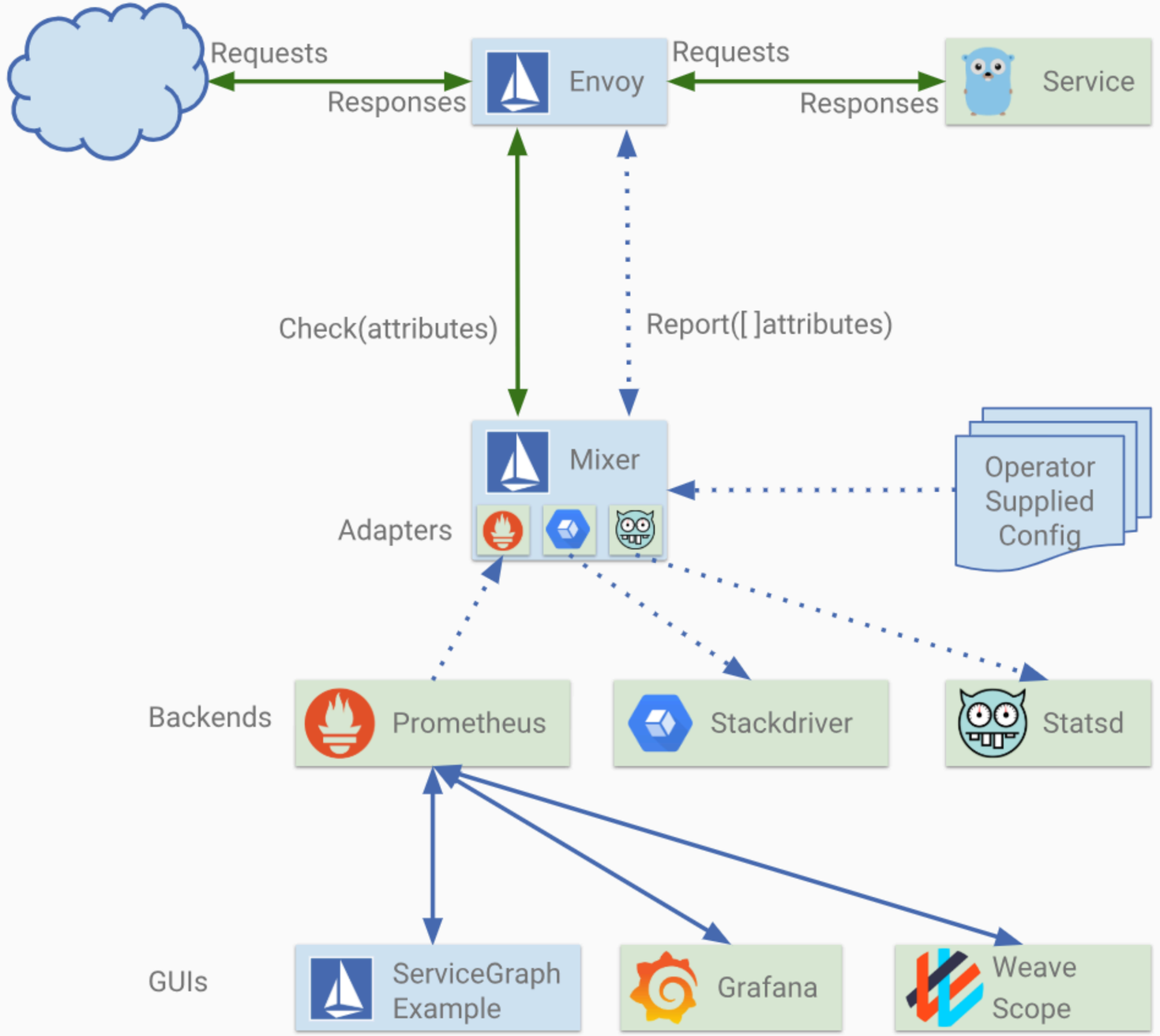
Telemetry Generation

Envoy sends information to Mixer about requests and responses in the form of *attributes* (typed key-value pairs).

Mixer transforms attribute sets into structured values, according to operator-supplied configuration.

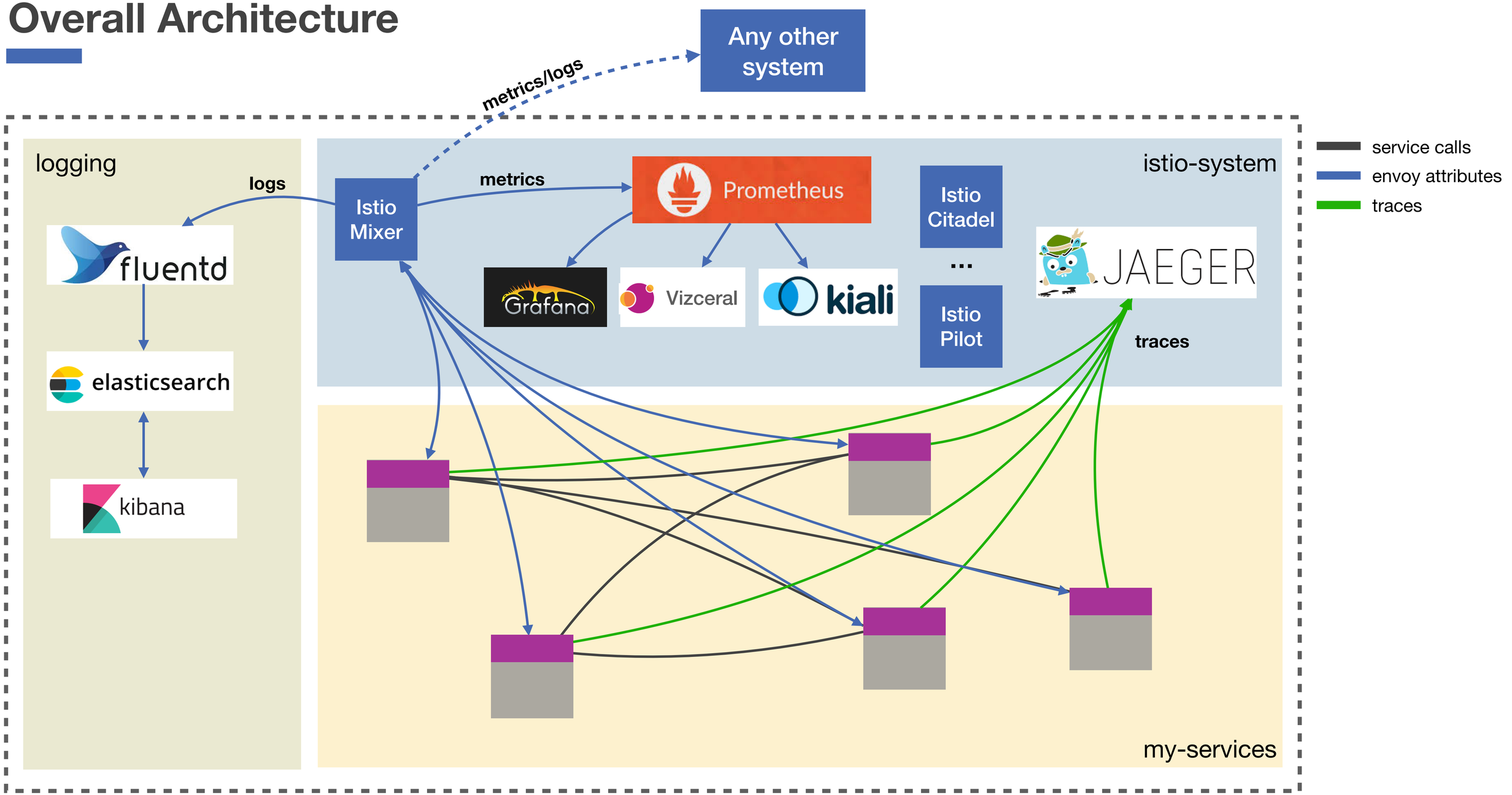
Mixer dispatches the derived values to a set of *adapters*, according to the operator-supplied configuration.

Adapters publish telemetry data to backend systems, making it available for further consumption and analysis.



https://docs.google.com/presentation/d/1vZtKzu4hBOShmnli-DbaqRowK4ErJ6oXelRcgM0n2ak/edit#slide=id.g2c1b15911f_0_542

Overall Architecture



Metrics

- By default, everything is in **Prometheus**
- **istio-mesh**, service mesh telemetry data generated by envoy
- **istio-telemetry**, metrics of telemetry itself
- Grafana for metrics visualization — **Make full usage of Grafana**
- Demo

```
root@qcloud:~/workspace# k get po -n istio-system -o wide
NAME                                      READY   STATUS      RESTARTS   AGE   IP
grafana-85f97b7b6-pj68c                   1/1     Running     0          23h   10.244.1.32
istio-citadel-657777464f-dz8pp            1/1     Running     0          23h   10.244.1.25
istio-cleanup-secrets-dxbn6               0/1     Completed   0          23h   10.244.0.38
istio-egressgateway-6497dfdfc-v4spf       1/1     Running     0          23h   10.244.0.40
istio-galley-5664cf4486-czx2z             1/1     Running     0          23h   10.244.0.39
istio-grafana-post-install-4tf9m          0/1     Completed   0          23h   10.244.1.31
istio-ingressgateway-787bf4cb5b-tmtkn     1/1     Running     0          23h   10.244.0.41
istio-pilot-54b6f64977-vbf6v              2/2     Running     0          23h   10.244.0.44
istio-policy-659559675c-trx8c             2/2     Running     0          23h   10.244.0.43
istio-sidecar-injector-7ddfb55469-99jkj   1/1     Running     0          23h   10.244.1.35
istio-statsd-prom-bridge-67bbcc746c-fsvzq 1/1     Running     0          23h   10.244.1.34
istio-telemetry-7d4f794c8d-q7vct          2/2     Running     0          23h   10.244.2.24
istio-tracing-6445d6dbbf-wtqj6            1/1     Running     0          23h   10.244.2.26
kiali-b765dfc99-swbv7                     1/1     Running     0          23h   10.244.0.42
prometheus-6967c997cf-r4nl2               1/1     Running     0          23h   10.244.1.33
```

Targets

All | Unhealthy

envoy (1/1 up) show more

galley (1/1 up) show more

istio-mesh (1/1 up) show less

Endpoint	State
http://10.244.2.24:42422/metrics	UP

istio-policy (1/1 up) show more

istio-telemetry (1/1 up) show less

Endpoint	State
http://10.244.2.24:9093/metrics	UP

kubernetes-apiservers (1/1 up) show more

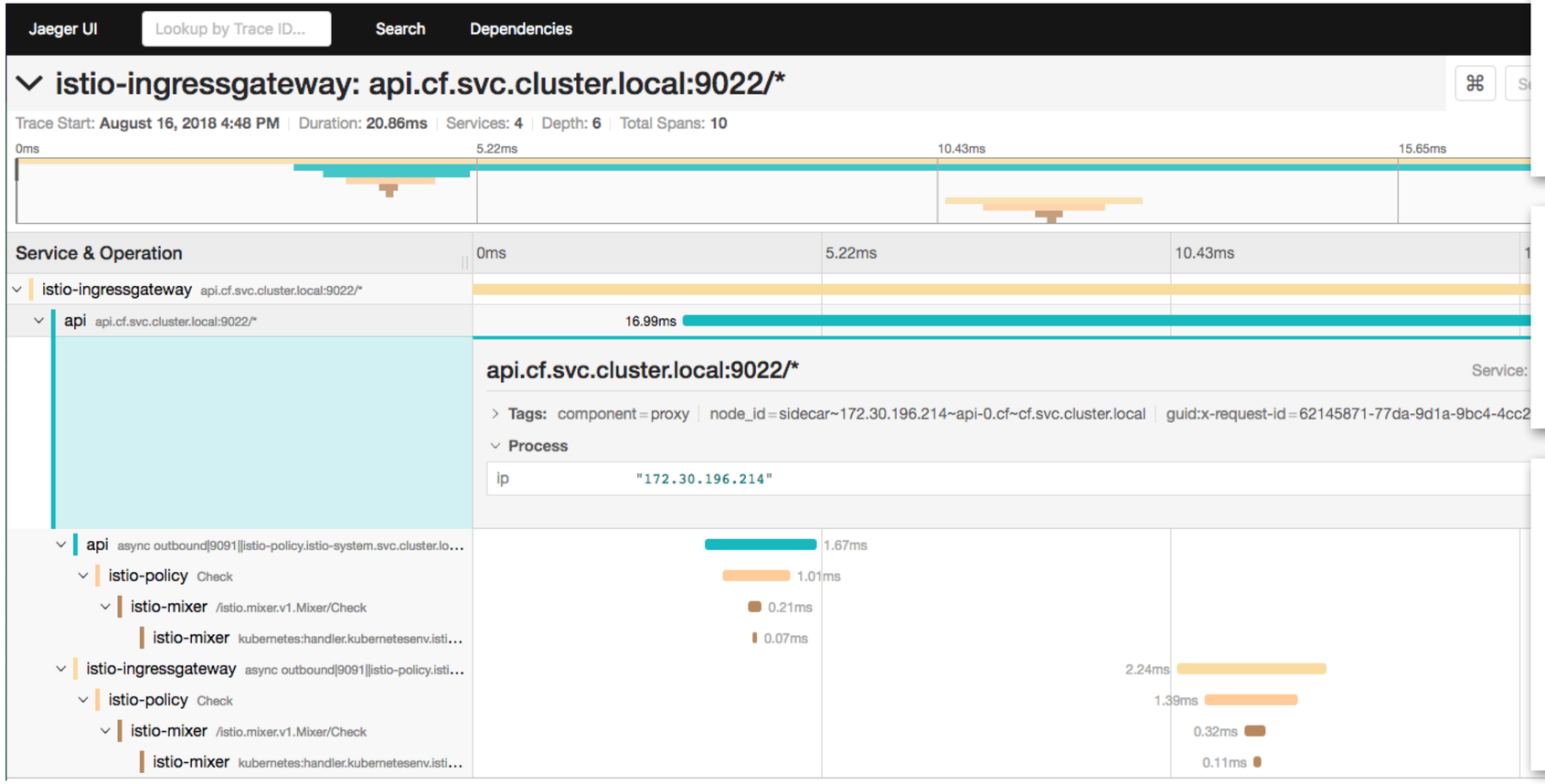
kubernetes-cadvisor (3/3 up) show more

kubernetes-nodes (3/3 up) show more

kubernetes-service-endpoints (3/3 up) show more

pilot (1/1 up) show more

Tracing — Jaeger



Problems that Jaeger addresses

- distributed transaction monitoring
- performance and latency optimization
- root cause analysis
- service dependency analysis
- distributed context propagation

What is Jaeger

- CNCF project, native support for OpenTracing
- Find service upstream/downstream dependencies by check opentracing standard headers
- Supports Istio by default

How we want to use it — scenarios

- Find out where consumes most of the time in request chain — **Bottle neck identification**
- **Trouble shooting for specific requests**

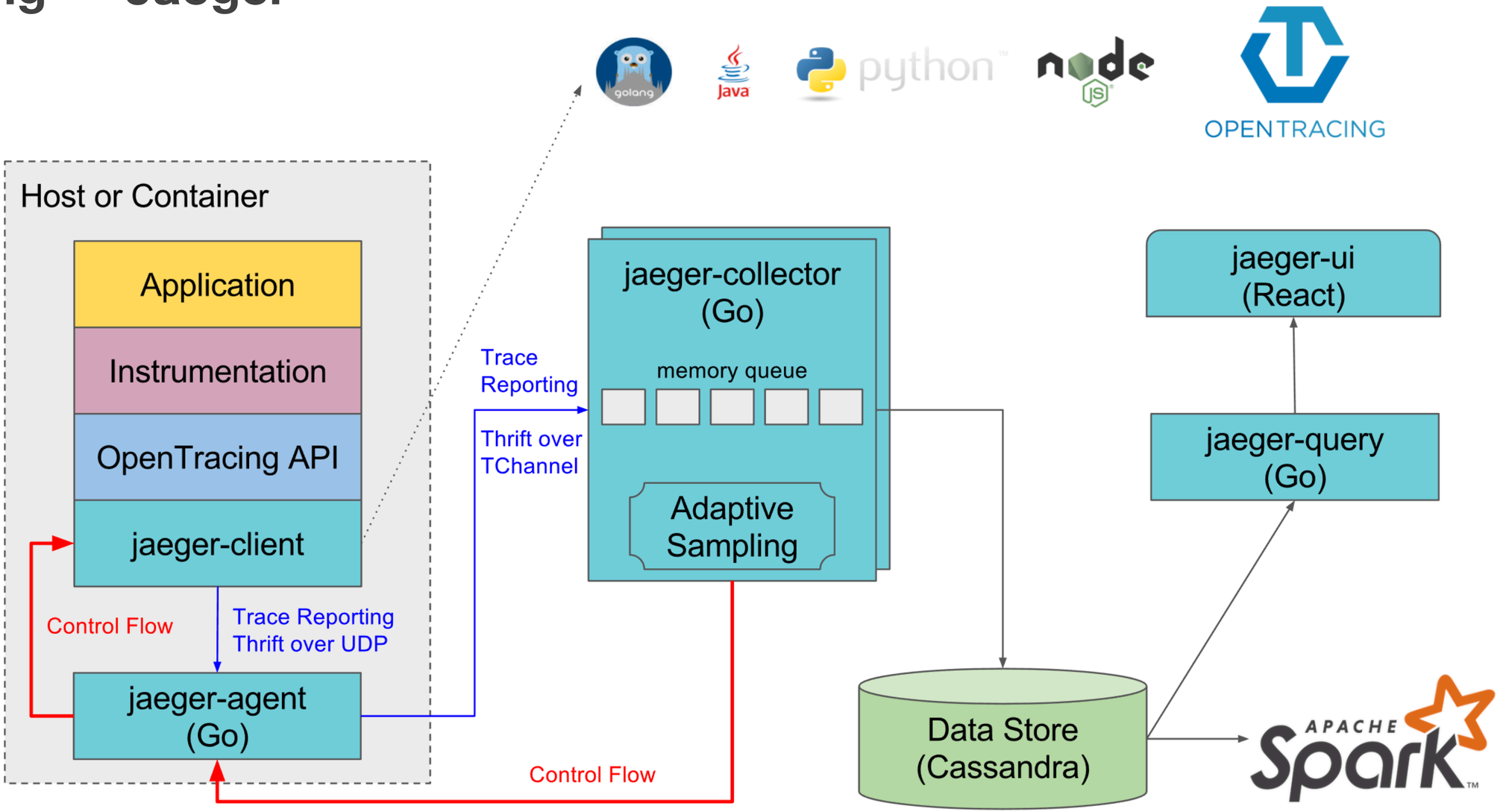
What is SPAN in OpenTracing

- An operation name
- A start timestamp and finish timestamp
- A set of key:value span Tags
- A set of key:value span Logs
- A SpanContext

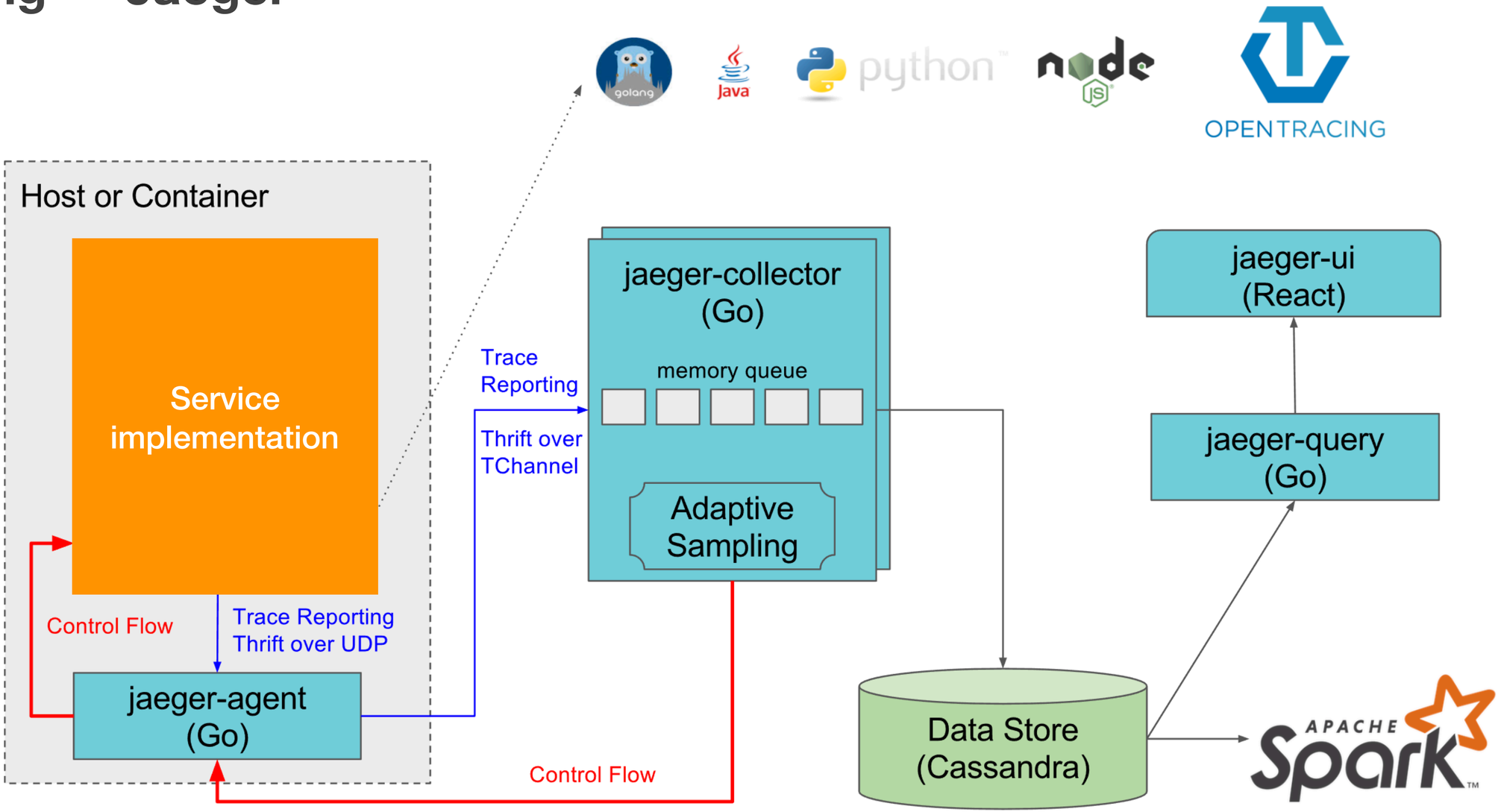
What is SPAN in Jaeger

A **span** represents a logical unit of work in Jaeger that has an ***operation name***, the ***start time*** of the operation, and the ***duration***. Spans may be nested and ordered to model causal relationships.

Tracing — Jaeger



Tracing — Jaeger



Tracing

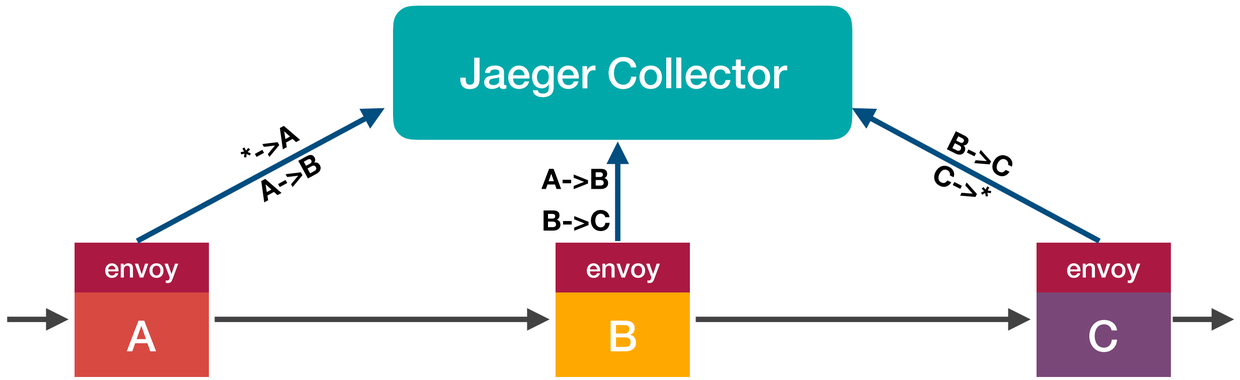
- Tracing is a **super powerful tool** to help with trouble shooting and performance analysis in real world operation
- Inspired by Dapper and Zipkin, initiated by Uber
- Implemented by following OpenTracing https://opentracing.io/docs/overview/
- In Istio, spans will be generated by each envoy sidecar and send to Jaeger by default — separated spans
- To link spans together, code change is required — “**context propagation**”

bookinfo — details.rb

```
def get_forward_headers(request)
  headers = {}
  incoming_headers = [ 'x-request-id',
                       'x-b3-traceid',
                       'x-b3-spanid',
                       'x-b3-parentspanid',
                       'x-b3-sampled',
                       'x-b3-flags',
                       'x-ot-span-context'
                     ]

  request.each do |header, value|
    if incoming_headers.include? header then
      headers[header] = value
    end
  end

  return headers
end

server.start
```

Tracing

- Tracing is a **super powerful tool** to help with trouble shooting and performance analysis in real world operation
- Inspired by Dapper and Zipkin, initiated by Uber
- Implemented by following OpenTracing https://opentracing.io/docs/overview/
- In Istio, spans will be generated by each envoy sidecar and send to Jaeger by default — separated spans
- To link spans together, code change is required — "**context propagation**"

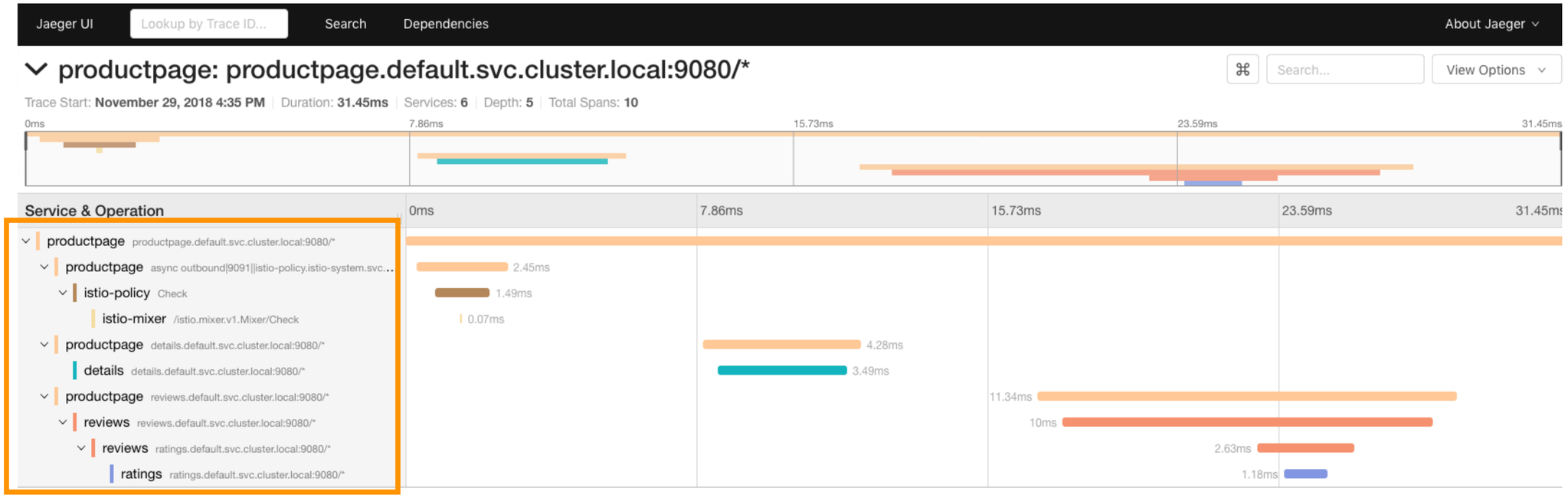


bookinfo — details.rb

```
def get_forward_headers(request)
  headers = {}
  incoming_headers = [ 'x-request-id',
                       'x-b3-traceid',
                       'x-b3-spanid',
                       'x-b3-parentspanid',
                       'x-b3-sampled',
                       'x-b3-flags',
                       'x-ot-span-context'
                     ]

  request.each do |header, value|
    if incoming_headers.include? header then
      headers[header] = value
    end
  end

  return headers
end

server.start
```

Tracing

Separated



Linked

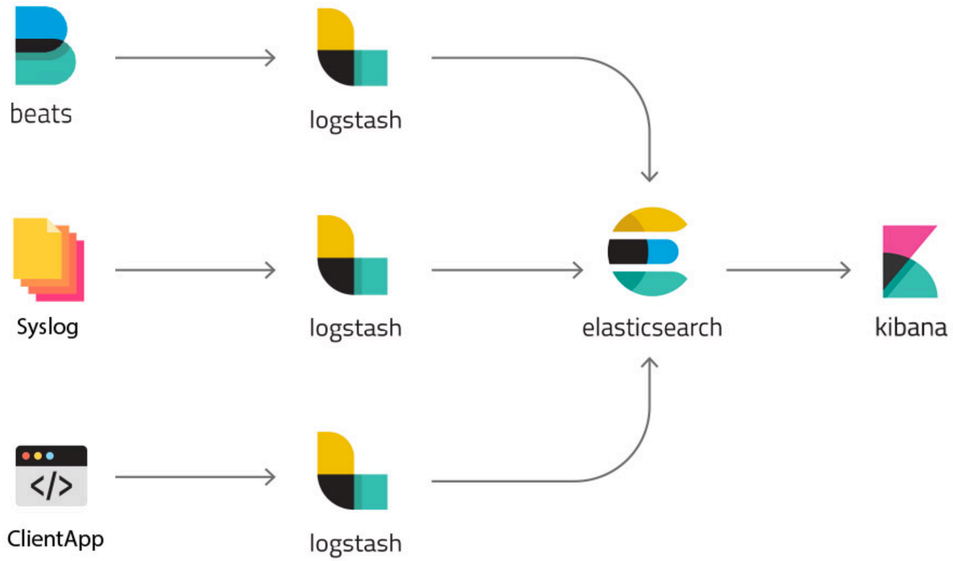
Tracing - Production Readiness

- ☢ Current steps described in official document "Distributed Tracing" is not a production ready solution — “ALL_IN_ONE” mode, and ***SPAN_STORAGE_TYPEIS*** is ***in memory***
- Change backend storage to Cassandra or Elasticsearch or Kafka for data persistent
- HPA for jaeger-collector

```
cmd [master●●] k get po -n istio-system
NAME                                      READY     STATUS
grafana-546d9997bb-fq5fg                  1/1       Running
istio-citadel-6955bc9cb7-5r47x            1/1       Running
istio-egressgateway-7dc5cbbc56-w7tvb      1/1       Running
istio-galley-545b6b8f5b-mrnch             1/1       Running
istio-ingressgateway-7958d776b5-kvjvf     1/1       Running
istio-pilot-56bfdbffff-xfd2w              2/2       Running
istio-policy-5c689f446f-g948f             2/2       Running
istio-sidecar-injector-99b476b7b-zdlrf    1/1       Running
istio-telemetry-55d68b5dfb-d65v9          2/2       Running
istio-tracing-6445d6dbbf-8zzrn            1/1       Running
kiali-ddf8fbbb-qvmgt                      1/1       Running
prometheus-65d6f6b6c-59jq7                1/1       Running
servicegraph-57c8cbc56f-trvmn             1/1       Running
cmd [master●●] k get svc -n istio-system
NAME                       TYPE           CLUSTER-IP       E
grafana                    NodePort       10.111.216.82    <
istio-citadel              ClusterIP      10.102.88.145    <
istio-egressgateway        ClusterIP      10.109.197.155   <
istio-galley               ClusterIP      10.96.50.109     <
istio-ingressgateway       LoadBalancer   10.108.209.24    <
istio-pilot                ClusterIP      10.99.181.25     <
istio-policy               ClusterIP      10.109.129.152   <
istio-sidecar-injector     ClusterIP      10.102.110.132   <
istio-telemetry            NodePort       10.96.87.57      <
jaeger-agent               ClusterIP      None             <
jaeger-collector           ClusterIP      10.102.108.54    <
jaeger-query               ClusterIP      10.98.62.81      <
kiali                      NodePort       10.98.86.230     <
prometheus                 NodePort       10.97.65.25      <
servicegraph               ClusterIP      10.105.130.215   <
tracing                    ClusterIP      10.105.44.158    <
zipkin                     ClusterIP      10.109.202.116   <
cmd [master●●]
```

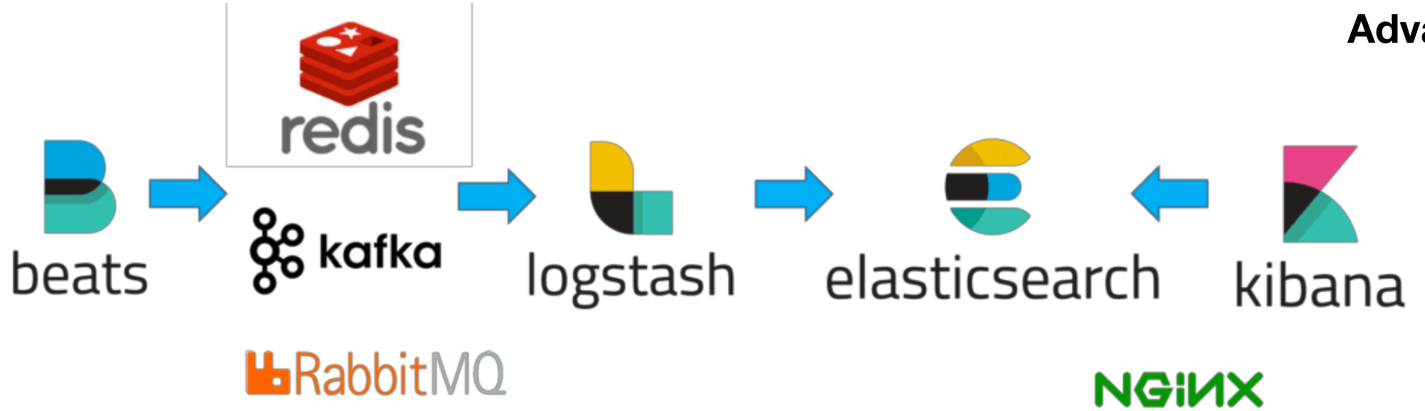
```
spec:
  containers:
    - name: jaeger-agent
      image: jaegertracing/jaeger-agent
      imagePullPolicy: IfNotPresent
      ports:
        - containerPort: 5778
          protocol: TCP
        - containerPort: 5775
          protocol: UDP
        - containerPort: 6831
          protocol: UDP
        - containerPort: 6832
          protocol: UDP
    - name: jaeger-query
      image: jaegertracing/jaeger-query
      imagePullPolicy: IfNotPresent
      args:
      - "--es.server-urls=http://k8s-support-cluster2.fyre.ibm.com:9200"
      ports:
        - containerPort: 16686
        protocol: TCP
        - containerPort: 16687
          protocol: TCP
      env:
      - name: SPAN_STORAGE_TYPE
        value: "elasticsearch"
    - name: jaeger-collector
      image: jaegertracing/jaeger-collector
      imagePullPolicy: IfNotPresent
      args:
      - "--es.server-urls=http://k8s-support-cluster2.fyre.ibm.com:9200"
      - "--es.num-shards=1"
      ports:
        - containerPort: 14269
          protocol: TCP
        - containerPort: 14268
          protocol: TCP
        - containerPort: 14267
          protocol: TCP
        - containerPort: 9411
          protocol: TCP
      env:
      - name: COLLECTOR_ZIPKIN_HTTP_PORT
        value: "9411"
      - name: SPAN_STORAGE_TYPE
        value: "elasticsearch"
```

Logging — in the old days…



Basic

Advanced



Data Collection

Buffering

Data Aggregation & Processing

Indexing & storage

Analysis & visualization

Logging

What is fluentd

- CNCF project, work with Elasticsearch and Kibana well, and also Kubernetes
- Light weight, very little foot print
- Rich plugins and community support

fluentd config from `logging-stack.yaml`

```
# Fluentd ConfigMap, contains config files.
kind: ConfigMap
apiVersion: v1
data:
  forward.input.conf: |-
    # Takes the messages sent over TCP
    <source>
      type forward
    </source>
  output.conf: |-
    <match **>
       type elasticsearch
       log_level info
       include_tag_key true
       host elasticsearch
       port 9200
       logstash_format true
       # Set the chunk limits.
       buffer_chunk_limit 2M
       buffer_queue_limit 8
       flush_interval 5s
       # Never wait longer than 5 minutes between
retries.
       max_retry_wait 30
       # Disable the limit on the number of retries
(retry forever).
       disable_retry_limit
       # Use multiple threads for processing.
       num_threads 2
    </match>
metadata:
  name: fluentd-es-config
  namespace: logging
```

Logging

- ☢ Current steps described in official document "Logging with Fluentd" is not a production ready solution
- “logging” in Istio, seems to be another way of collecting request specific data
- What about application logs?
- *Investigation in progress*: how to extend istio logging to include application logs as well
- Demo

```
# Configuration for logentry instances
apiVersion: "config.istio.io/v1alpha2"
kind: logentry
metadata:
  name: newlog
  namespace: istio-system
spec:
  severity: '"info"'
  timestamp: request.time
  variables:
    source: source.labels["app"] |
source.workload.name | "unknown"
    user: source.user | "unknown"
    destination: destination.labels["app"] |
destination.workload.name | "unknown"
    responseCode: response.code | 0
    responseSize: response.size | 0
    latency: response.duration | "0ms"
  monitored_resource_type: '"UNSPECIFIED"'
---
# Configuration for a fluentd handler
apiVersion: "config.istio.io/v1alpha2"
kind: fluentd
metadata:
  name: handler
  namespace: istio-system
spec:
  address: "fluentd-es.logging:24224"
---
# Rule to send logentry instances to the
fluentd handler
apiVersion: "config.istio.io/v1alpha2"
kind: rule
metadata:
  name: newlogtofluentd
  namespace: istio-system
spec:
  match: "true" # match for all requests
  actions:
   - handler: handler.fluentd
     instances:
     - newlog.logentry
```

Kiali

- *“Kiali is originated from the greek word κιάλι meaning monocular or spyglass.”*
- Initiated by Redhat
- https://www.kiali.io
- Officially included in Istio deployment since version v1.1: Visualizing Your Mesh
- Very active community and development sprints
- Demo

Features

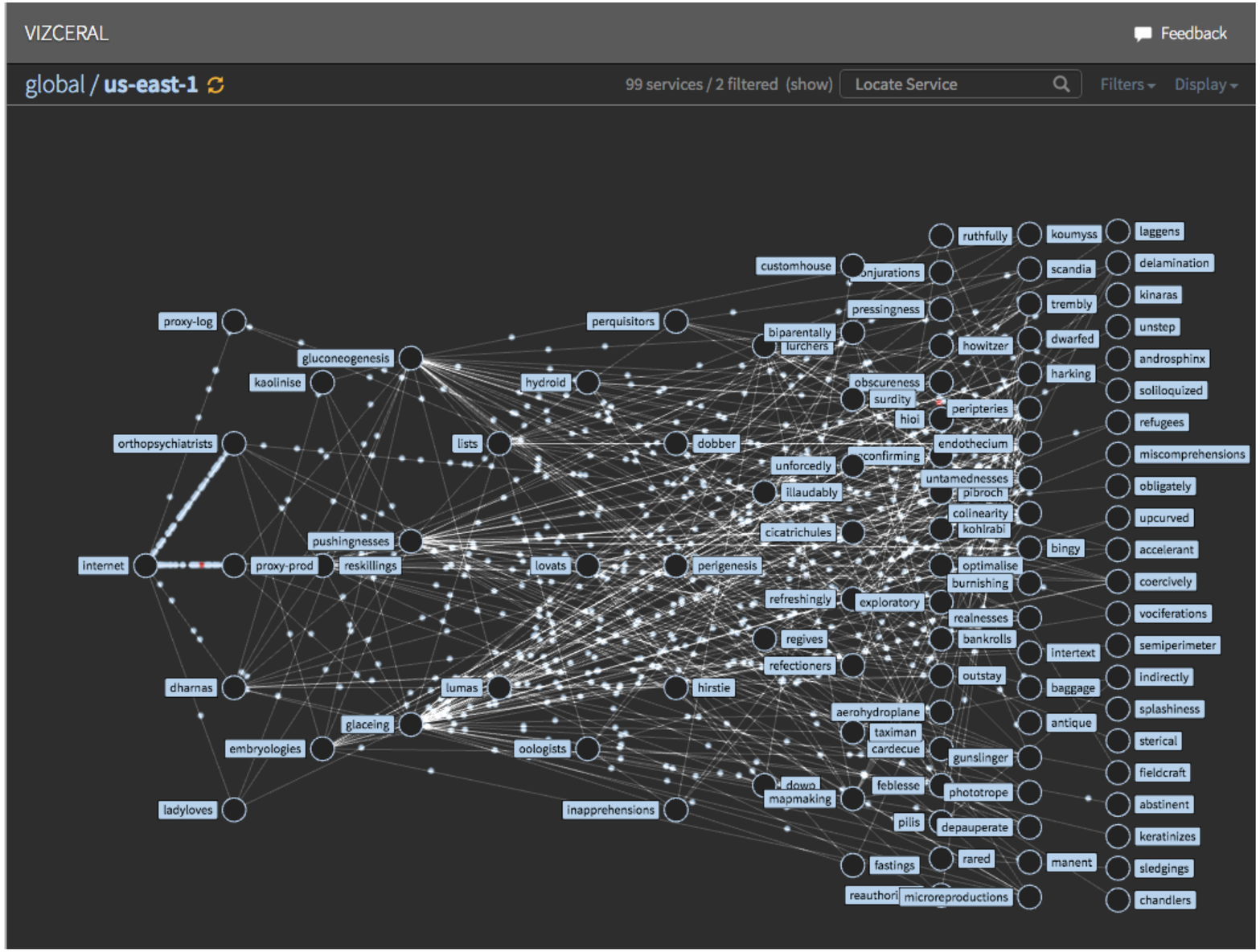
A growing set of features

- Service graph representation
- Distributed tracing
- Metrics collection and graphs
- Configuration validation
- Health computation/display
- Service discovery

Vizceral

- Vizceral is only a visualization font-end written in javascript, and it can support any type of datasource, as long as the data is formatted correctly.
- To have Prometheus working with Vizceral, there's a project called promviz, which reads data from Prometheus, and convert the data in to Vizceral format to consume.
- ☢ Vizceral is **not actively maintained** by Netflix anymore, there's a fork project called promviz-front, and it can work with promviz perfectly, and also have several important new features, like replay, connection chart, etc.

Sample by Netflix



Resources

- ★ **NewRelic adapter repo: https://github.com/IBM/newrelic-istio-adapter**
- Istio official website: https://istio.io
- IBM Istio 101: https://github.com/ibm/istio101

Technical blogs we posted:

- Enable Istio for your service: https://www.ibm.com/blogs/bluemix/2018/11/enable-istio-for-your-service/
- How to deploy a gRPC mode Istio Mixer adapter into k8s: https://www.ibm.com/blogs/bluemix/2018/09/how-to-deploy-a-grpc-mode-istio-mixer-adapter-into-kubernetes/
- How to Enable Kiali for Istio on IBM Cloud Kubernetes Service: https://www.ibm.com/blogs/bluemix/2018/08/how-to-enable-kiali-for-istio-on-ibm-cloud-kubernetes-service/

Thanks!