

Observability on Service Mesh

吴 晟

Apache SkyWalking 创始人、PPMC
Microsoft MVP
比特大陆 资深技术专家
Tetrate



Observability

CNCF Landscape

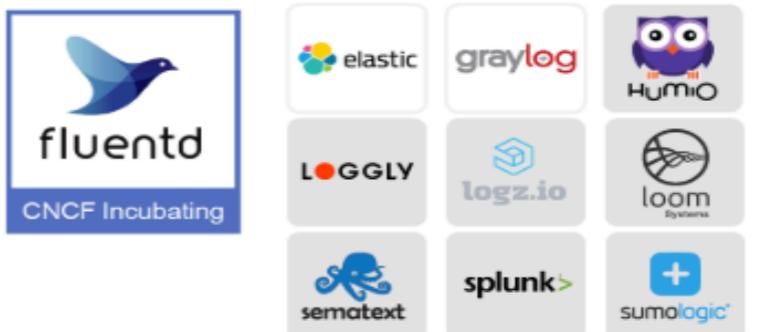


Observability & Analysis

Monitoring



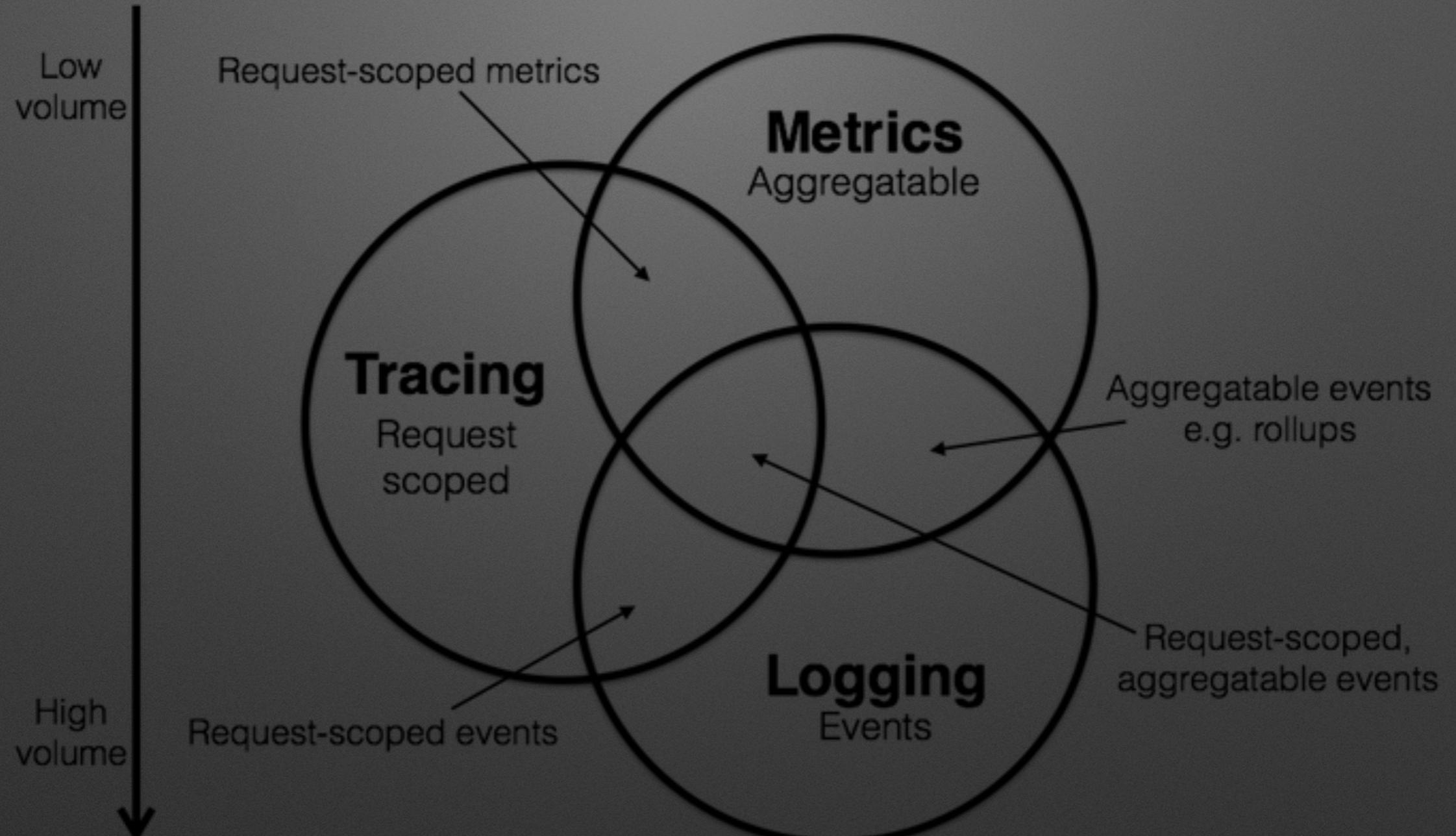
Logging

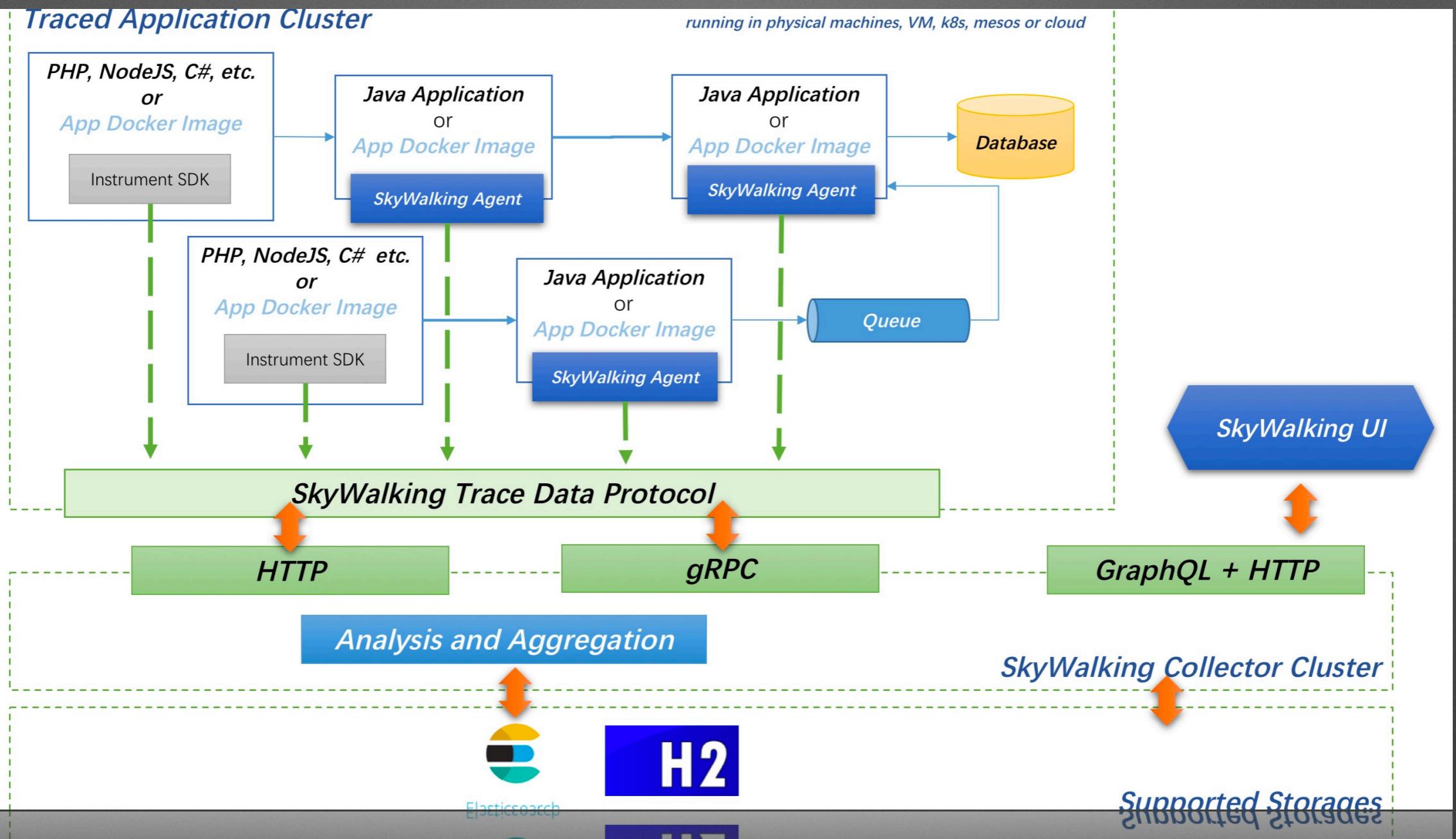


Tracing



Metric, Tracing, Logging





Traditional Observability

How does agent work?

```
PreparedStatement updateSales = con.prepareStatement(  
    "UPDATE COFFEES SETSALES = ? WHERE COF_NAME LIKE ? ");  
updateSales.setInt(1, 75);  
updateSales.setString(2, "Colombian");  
updateSales.executeUpdate();
```



```
PreparedStatement updateSales = con.prepareStatement(  
    "UPDATE COFFEES SETSALES = ? WHERE COF_NAME LIKE ? ");  
tracer.cacheSQL("UPDATE COFFEES SETSALES = ? WHERE COF_NAME LIKE ? ");  
updateSales.setInt(1, 75);  
tracer.cacheDbParam(1, 75);  
updateSales.setString(2, "Colombian");  
tracer.cacheDbParam(2, "Colombian");  
Span span = tracer.createSpan().start();  
updateSales.executeUpdate();  
span.tag(Tags.Basic.TYPE, "Database");  
span.tag(Tags.SQL, tracer.getCachedSQL());  
span.tag(Tags.SQL_PARAMETERS, tracer.getCachedDbParams());  
span.stop();
```

Application Performance Management

- Agent based
- Manipulate source codes
 - Auto
 - Manual
- CPU, Memory, Latency Cost, GC

Id	APM	采样率	线程数	请求总数	平均请求时间 ms	最小请求时间 ms	最大请求时间 ms	90%Line	错误率 %	CPU	memory	Throughput /sec
1	none	1	500	15000	17	9	824	21	0	45%	50%	1385
2	Zipkin	1	500	15000	117	10	2101	263	0	56%	55%	990
3	Skywalking	1	500	15000	22	10	1026	23	0	50%	52%	1228
4	Pinpoint	1	500	15000	201	10	7236	746	0	48%	52%	774
5	none	1	750	22500	321	10	15107	991	0	56%	48%	956
6	Zipkin	1	750	22500	489	10	27614	1169	0	63%	55%	582
7	Skywalking	1	750	22500	396	10	16478	941	0	55%	50%	908
8	Pinpoint	1	750	22500	681	10	28138	1919	0	56%	48%	559
9	none	1	1000	30000	704	10	39772	1621	0	59%	53%	557
10	Zipkin	1	1000	30000	1021	10	36836	1978	0	63%	55%	533
11	Skywalking	1	1000	30000	824	10	25983	1758	0	62%	55%	667
12	Pinpoint	1	1000	30000	1148	10	40971	2648	0	60%	52%	514

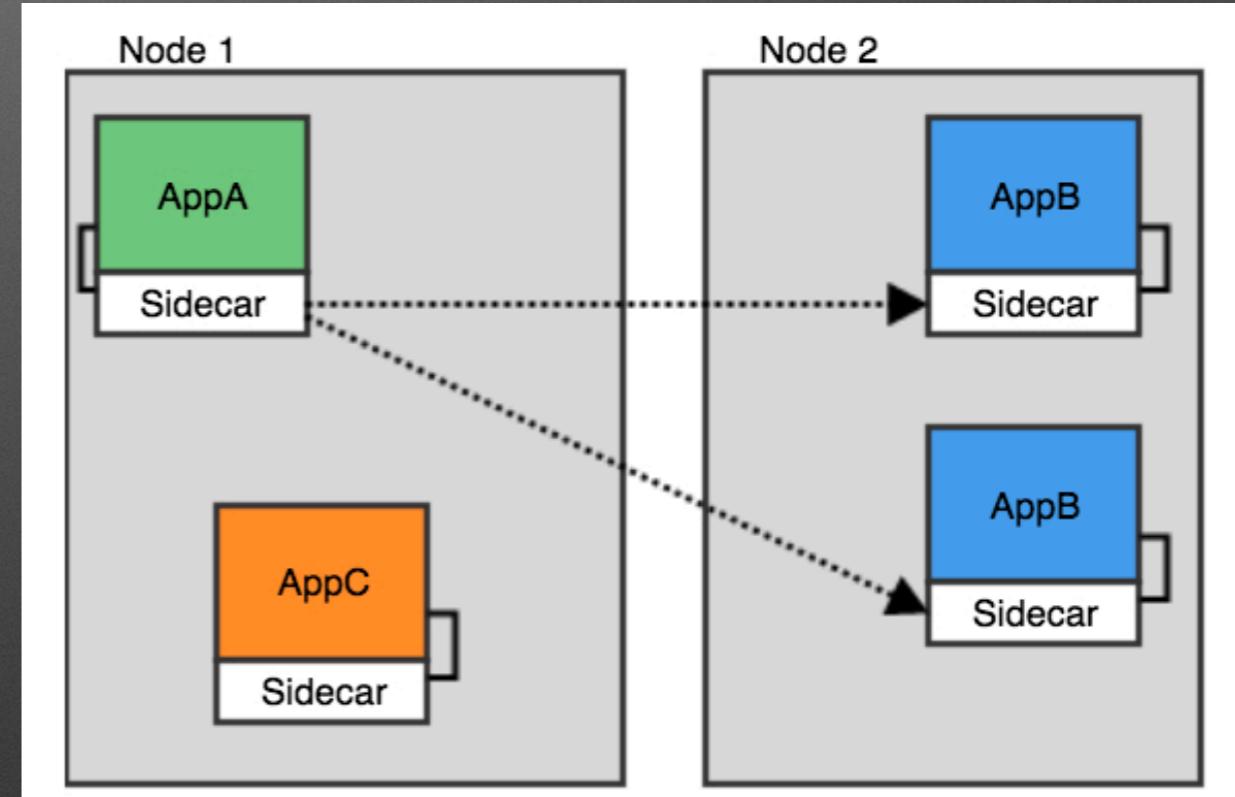
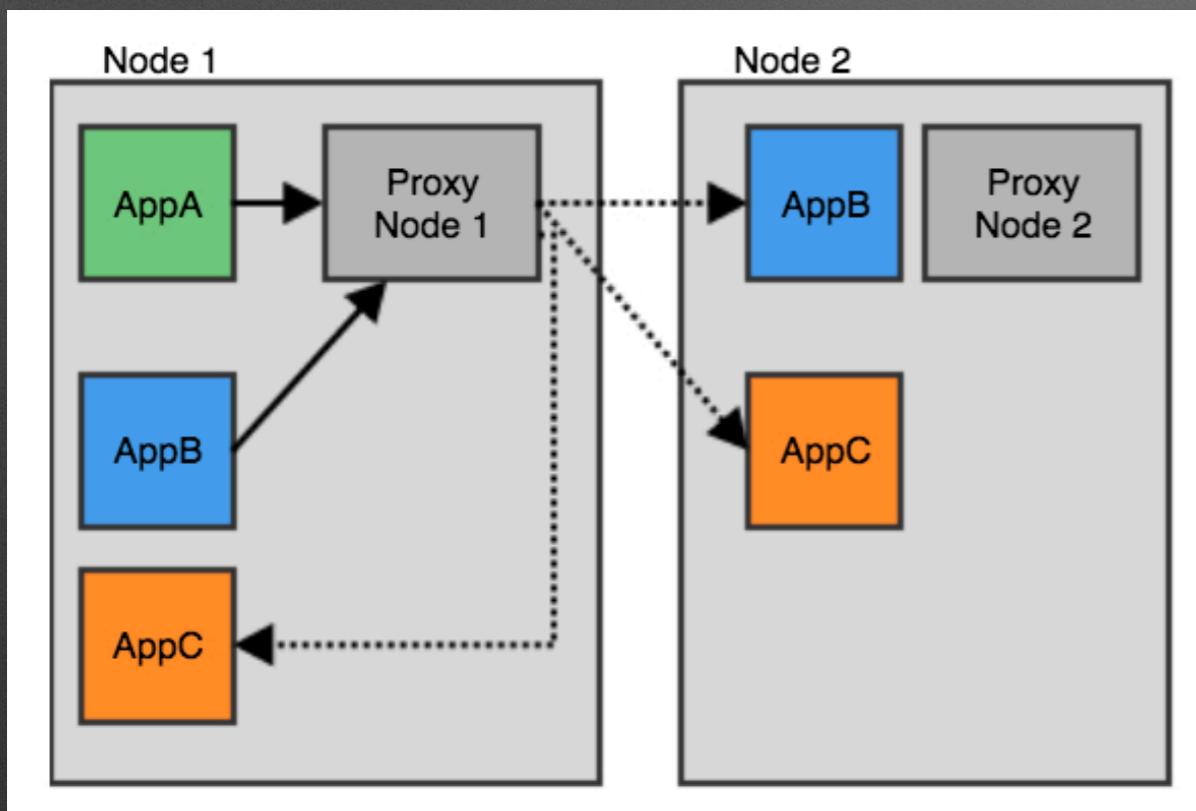
Cost of open source solutions

Don't consider this as an competition. It is a fact that everyone costs resources.

What change?

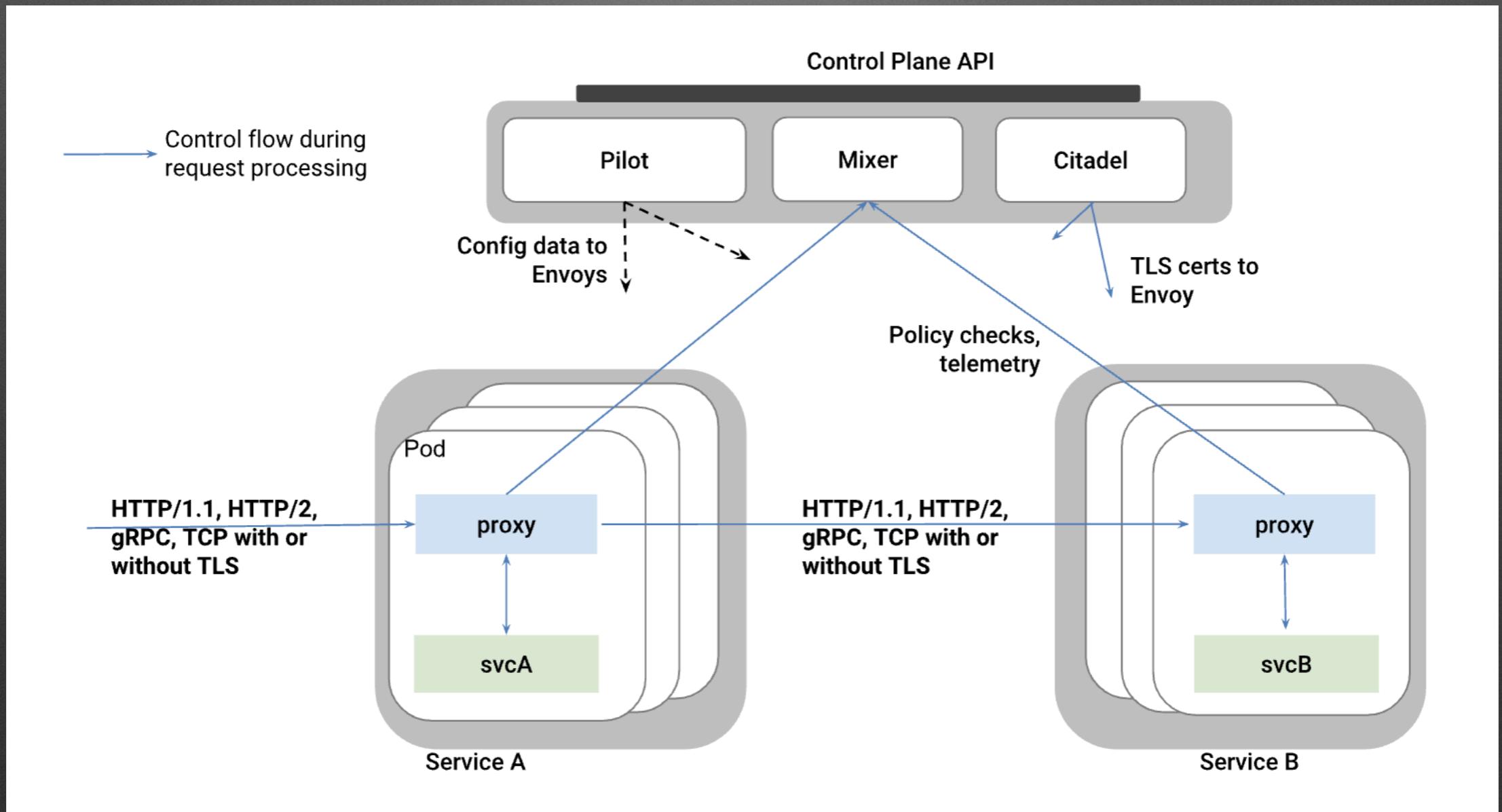
Container, K8s and Service Mesh

Proxy and Sidecar

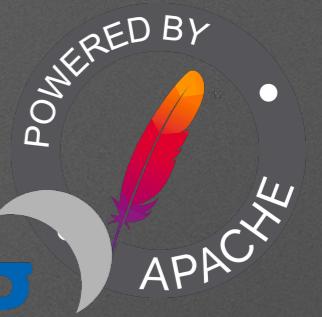


Istio + Envoy

Representative Service Mesh implementor

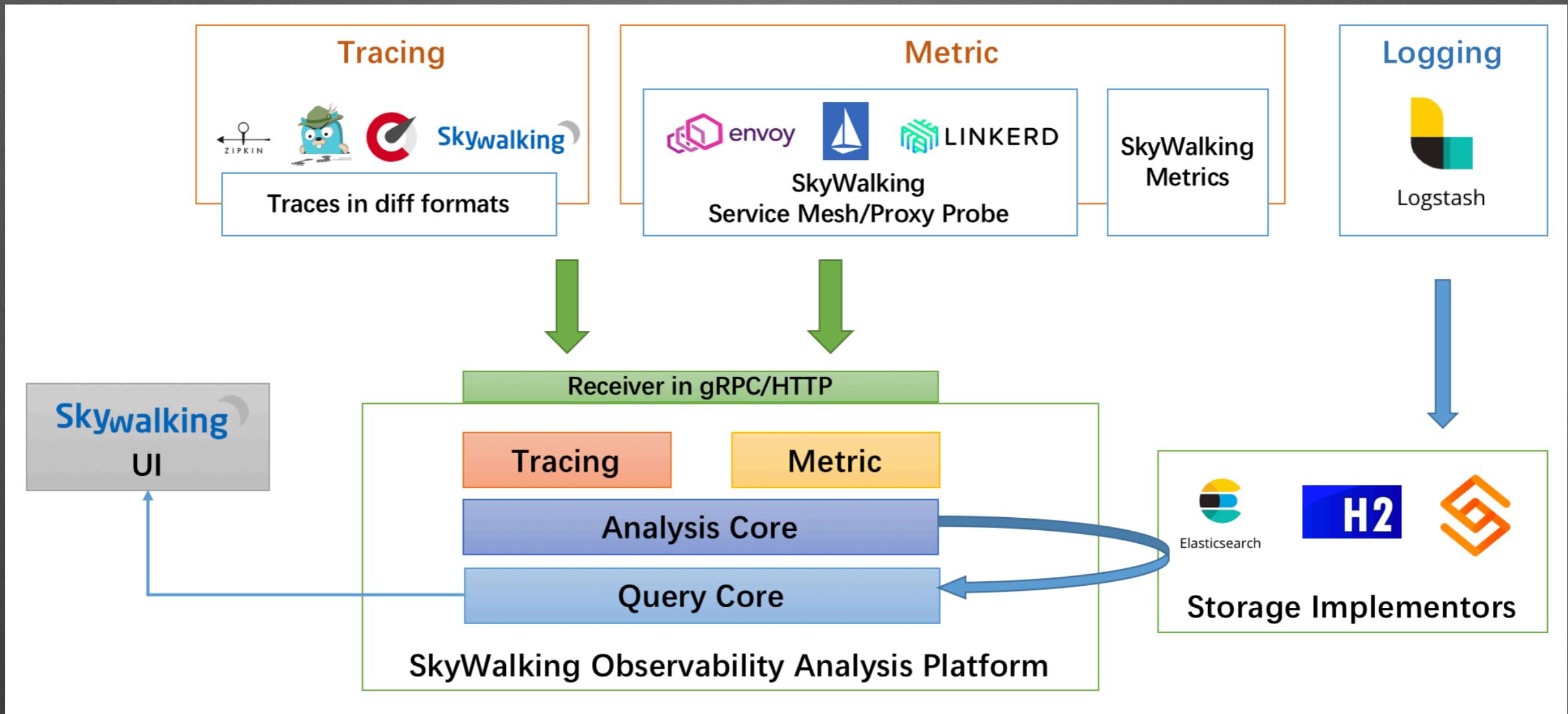


Skywalking



SkyWalking 6.0

Observability Analysis Platform



OAP > APM

Core Concepts

- **Service.** Represent a set/group of workloads to provide the same behaviors for incoming requests. You can define the service name when you are using instrument agents or SDKs. Or SkyWalking uses the name you defined in platform such as Istio.
- **Service Instance.** Each one workload in the Service group is named as an instance. Like pods in Kubernetes, it doesn't need to be a single process in OS. Also if you are using instrument agents, an instance is actually a real process in OS.
- **Endpoint.** It is a path in the certain service for incoming requests, such as HTTP URI path or gRPC service class + method signature.

OAP Key Feature

Multiple telemetry sources

- 1. Language based Agent**
- 2. Service Mesh Probe**
- 3. Other eco-system, like Zipkin**

Observability Analysis Language

- A compile language
- Scopes
 - All
 - Service
 - ServiceInstance
 - Endpoint
 - ServiceRelation
 - ServiceInstanceRelation
 - EndpointRelation
- Extendable Aggregation Functions

```
// Calculate p99 of both Endpoint1 and Endpoint2
Endpoint_p99 = from(Endpoint.latency).filter(name in ("Endpoint1", "Endpoint2")).summary(0.99)

// Calculate p99 of Endpoint name started with `serv`
serv_Endpoint_p99 = from(Endpoint.latency).filter(name like ("serv%")).summary(0.99)

// Calculate the avg response time of each Endpoint
Endpoint_avg = from(Endpoint.latency).avg()

// Calculate the histogram of each Endpoint by 50 ms steps.
// Always thermodynamic diagram in UI matches this metric.
Endpoint_histogram = from(Endpoint.latency).histogram(50)

// Calculate the percent of response status is true, for each service.
Endpoint_success = from(Endpoint.*).filter(status = "true").percent()

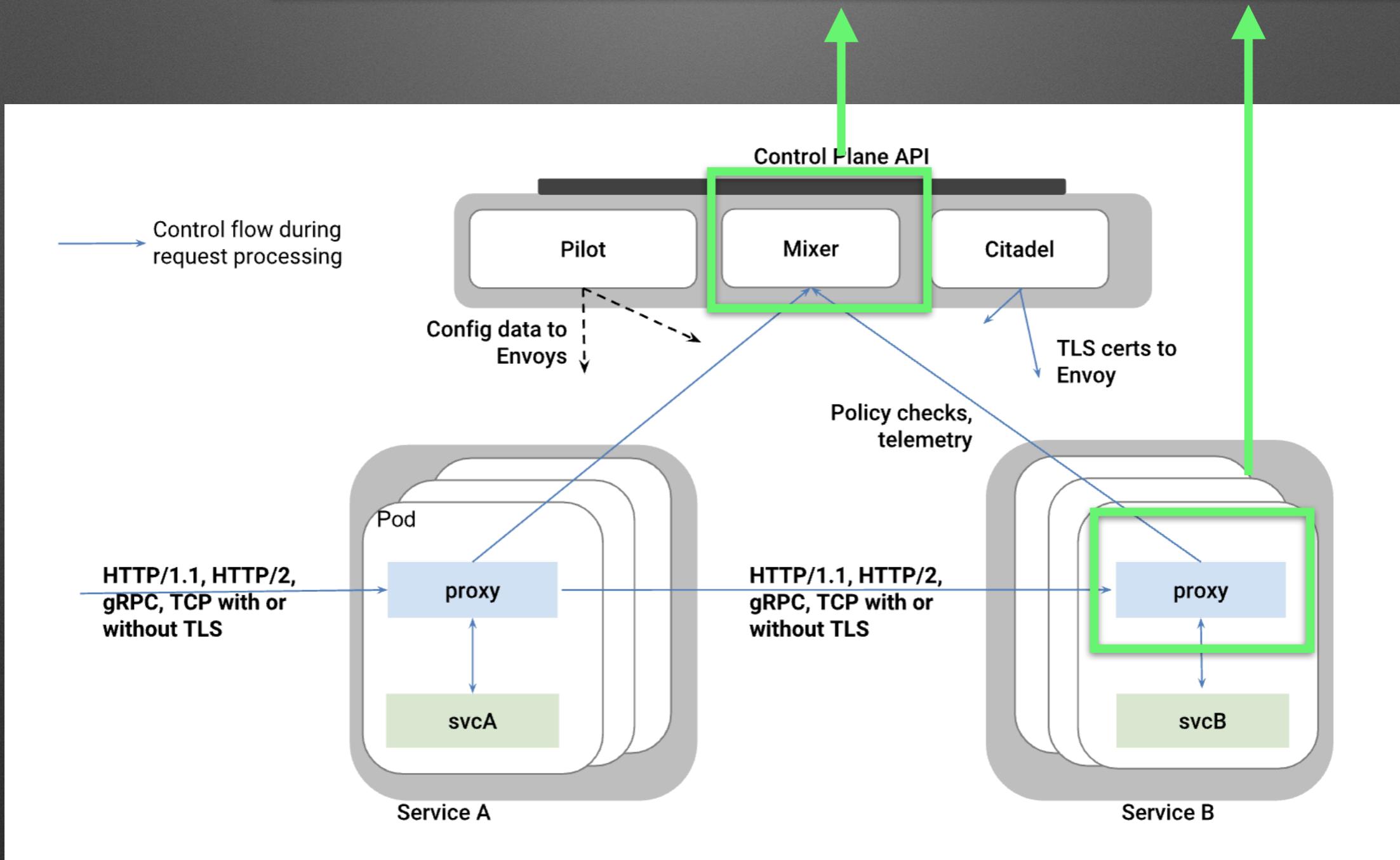
// Calculate the percent of response code in [200, 299], for each service.
Endpoint_200 = from(Endpoint.*).filter(responseCode like "2%").percent()

// Calculate the percent of response code in [500, 599], for each service.
Endpoint_500 = from(Endpoint.*).filter(responseCode like "5%").percent()

// Calculate the sum of calls for each service.
EndpointCalls = from(Endpoint.*).sum()
```

What is Service Mesh probe?

Metric from Service Mesh by native supported



Metric Data Structure

- Service Names at both sides.
- Service Instance Names at both sides.
- Endpoint. URI in HTTP, service method full signature in gRPC.
- Start Time. Request time.
- Latency. In milliseconds.
- Response code in HTTP
- Status. Success or fail.
- Protocol. HTTP, gRPC
- DetectPoint. In Service Mesh sidecar, client or server. In normal L7 proxy, value is proxy.

We need your star on GitHub

- <https://github.com/apache/incubator-skywalking>

The screenshot shows the GitHub profile for the Apache Incubator-SkyWalking project. At the top, the repository name "apache / incubator-skywalking" is displayed, along with metrics: 448 commits, 3,919 stars (highlighted with a red box), 1,233 forks, and 86 issues. Below this, there are tabs for Code, Issues (86), Pull requests (6), and Insights. A summary bar at the bottom provides details: 4,050 commits, 7 branches, 25 releases, 46 contributors, and Apache-2.0 licensing. Action buttons include Branch: master ▾, New pull request, Create new file, Upload files, Find file, and Clone or download ▾.

A distributed tracing system, and APM (Application Performance Monitoring) <https://skywalking.incubator.apache.org/>

apache / incubator-skywalking

448 3,919

Star

Fork 1,233

Code Issues 86 Pull requests 6 Insights

4,050 commits 7 branches 25 releases 46 contributors Apache-2.0

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

Thanks