Open Telemetry - Basic Concepts Understanding

Mukund

What is OpenTelemetry?

OPENTRACING

(Google)

Language-specific libraries
to instrument code

+

OpenCensus

(CNCF)

API for sending data to an back-end

=

OpenTelemetry

(CNCF
(Cloud Native Computing Foundation)

Second-busiest CNCF project
52 GH events/hour on average)
Behind Kubernetes

NimbusPay Technologies

Core Contributors

Google

Microsoft

honeycomb.io

SignalFx

Pivotal

dynatrace

omnition

Uber

solarwinds

INSTANA

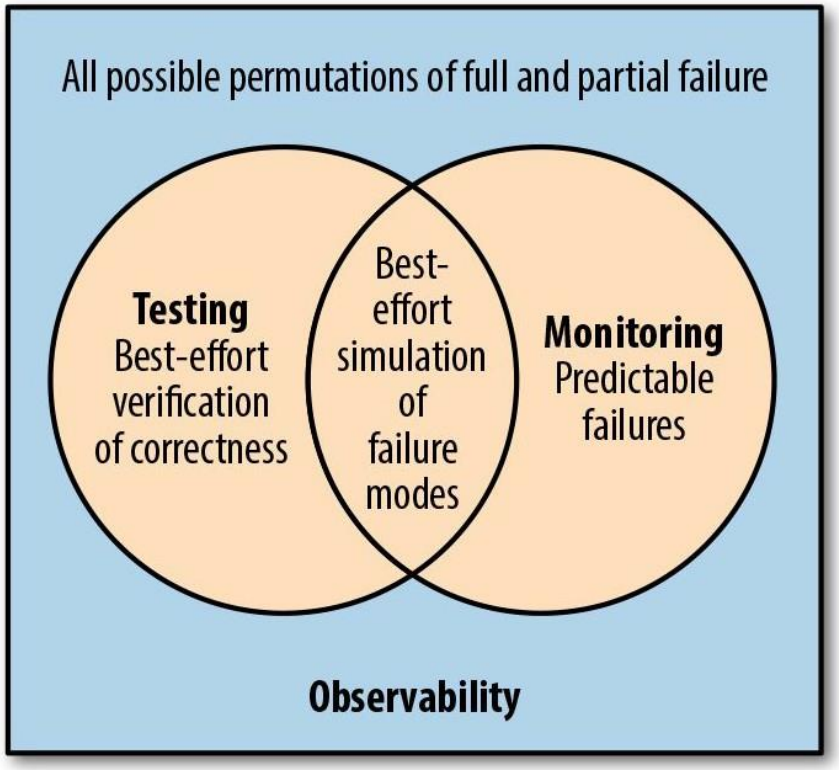
DATADOG

LIGHTSTEP

NimbusPay Technologies

Observability (O11Y) Vs Monitoring

OpenTelemetry



- Monitoring tells you whether the system works.
- Observability lets you ask why it is not working.
- The ability to infer the state of a system by examining its output

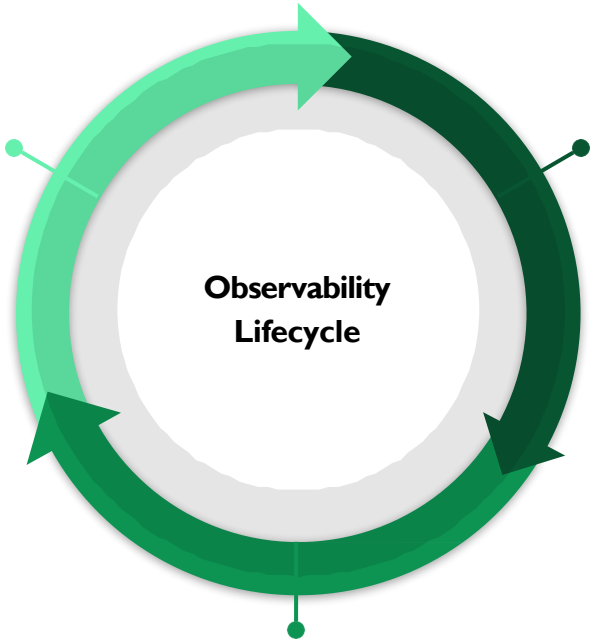
NimbusPay Technologies

Observability Lifecycle

OpenTelemetry

Track System Health

Capture traces, metrics and logs, create alerts on data



Detect Problems

Locality, networking, scheduling, dependencies

Fix & Refine

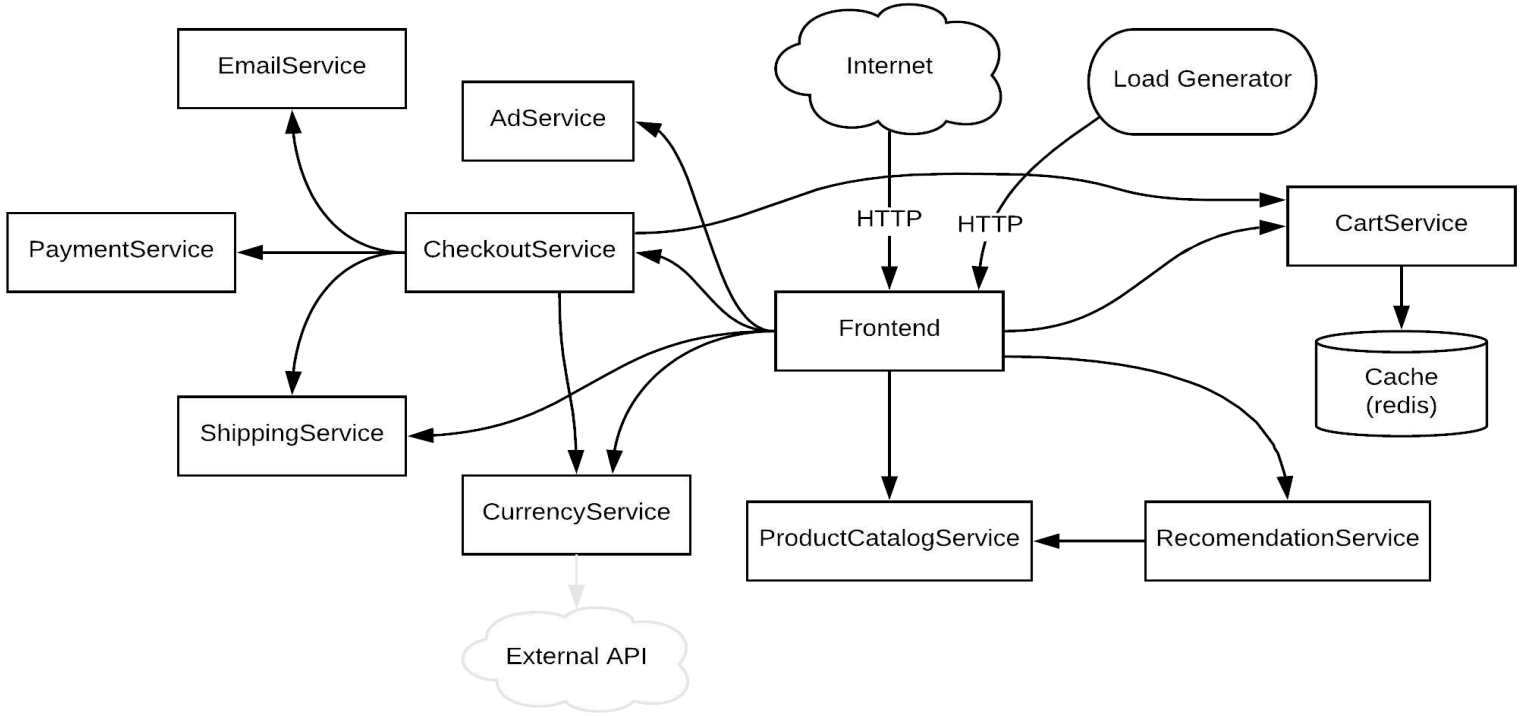
Optimize performance and cost of services

NimbusPay Technologies

Why Telemetry

OpenTelemetry

- Context propagation between components
- Multiple environments
- External dependencies



A **Context** is an object that contains the information for the sending and receiving service

Propagation is the mechanism that moves context between services and processes. It serializes or de-serializes the context object

Open Telemetry uses **Propagators** to serialize and de-serialize cross-cutting concern values such as Spans

NimbusPay Technologies

What is Telemetry

OpenTelemetry

- **Open Telemetry (Towel) is an open-source observability framework that allows development teams to generate, process, and transmit telemetry data in a single, unified format.**
- The purpose of Open-Telemetry is to collect, process, and export signals.
- **OpenTelemetry** is integrated set of APIs and libraries to generate, collect and describe telemetry in distributed systems

Problems OpenTelemetry solves:

- Vendor neutrality for tracing, monitoring and logging APIs
- Context propagation

NimbusPay Technologies

What is Telemetry Data?

OpenTelemetry

Telemetry data consists of the **logs, metrics and traces** collected from a distributed system. Known as the "**pillars of observability,**" these three categories of data helps, developers, DevOps and IT teams understand the behavior and performance of their systems

Traces:

- Understand the full path through your distributed application
- Traces can have multiple sub-paths (Spans)
- A group of spans constitutes a trace.

Metrics:

- A **metric** is a **measurement** of a service captured at runtime.
- The moment of capturing a measurements is known as a **metric event.**
- Metrics are numeric values measured over intervals of time, often known as time series data.

Logs:

- A **log** is a timestamped text record, either structured (recommended) or unstructured, with metadata.

To make a system observable, it must be instrumented. That is, the code must emit traces, metrics, or logsThe instrumented data must then be sent to an observability backend.

NimbusPay Technologies

Logs

OpenTelemetry

Field Name	Description
Timestamp	Time when the event occurred.
ObservedTimestamp	Time when the event was observed.
TraceId (Optional)	Request trace ID.
SpanId (Optional.	Request span ID.
TraceFlags (Optional)	W3C trace flag.
SeverityText	The severity text (also known as log level). INFO,DEBUG,WARNING,FATAL etc.
SeverityNumber	Numerical value of the severity.
Body (Optional)	The body of the log record. Human-readable string message
Resource (Optional)	Describes the source of the log.
InstrumentationScope (Optional)	Describes the scope that emitted the log.
Attributes (Optional)	Additional information about the event

Log Record - A log record represents the recording of an event.

Logger - A Logger creates log records. Loggers are created from Log Providers.

Logger Provider - Is a factory of Loggers.

Log Record Exporters - Send log records to a consumer.

NimbusPay Technologies

Log Lever and Severity Numbers

OpenTelemetry

Severity Number range	Range name	Meaning
1-4	TRACE	A fine-grained debugging event. Typically disabled in default configurations. (TRACE, TRACE2, TRACE3, TRACE4)
5-8	DEBUG	A debugging event.
9-12	INFO	An informational event. Indicates that an event happened.
13-16	WARN	A warning event. Not an error but is likely more important than an informational event.
17-20	ERROR	An error event. Something went wrong.
21-24	FATAL	A fatal error such as application or system crash.

Traces

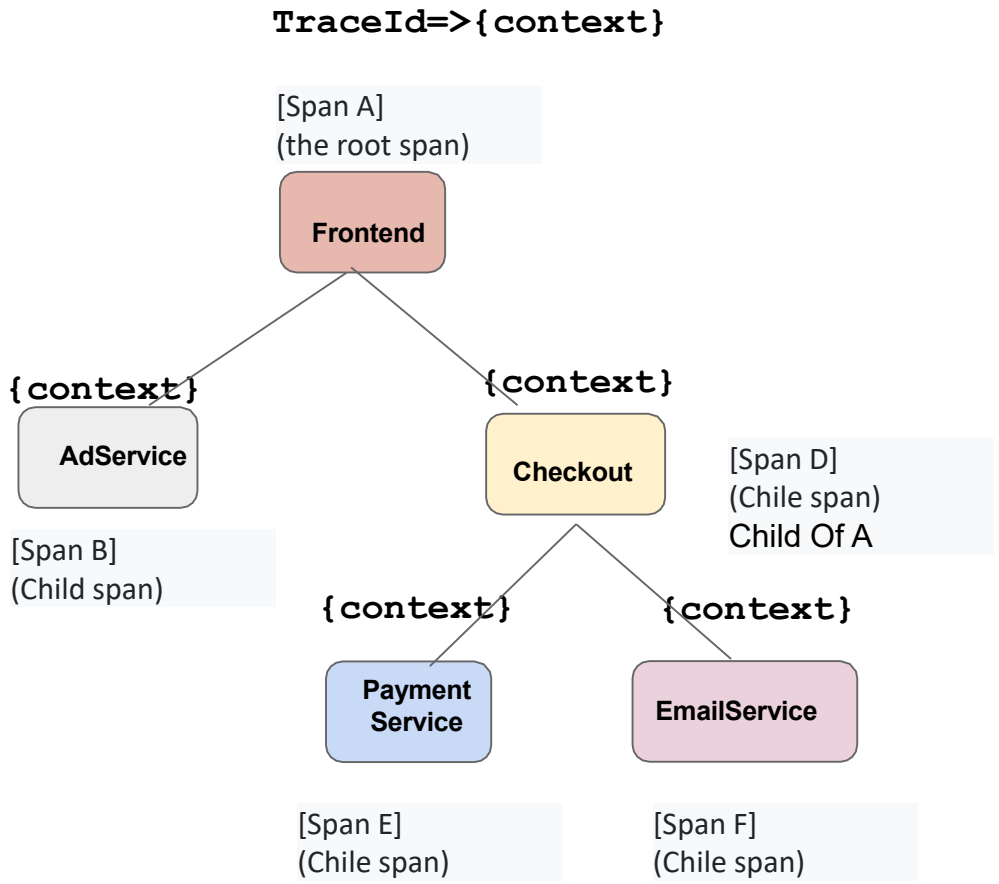
OpenTelemetry

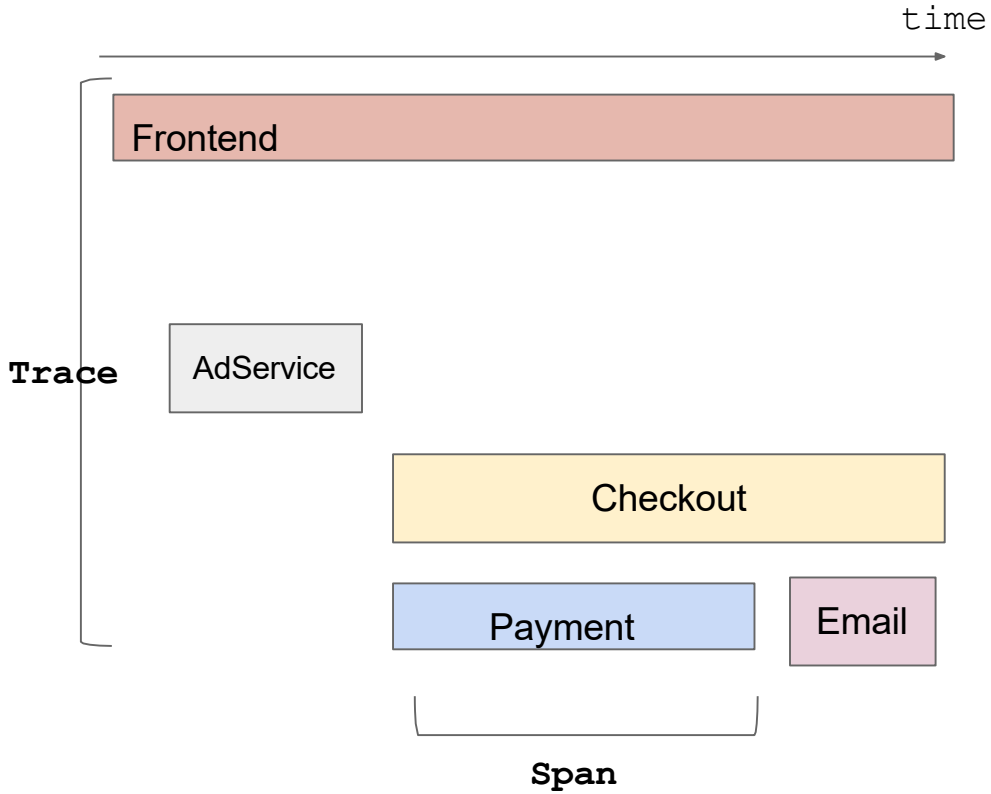
- Traces are collection of “structured logs with context”, correlation, hierarchy, and more. These “structured logs” can come from different processes, services, VMs, data centers, and so on. This is what allows tracing to represent an end-to-end view of any system.
- Hierarchy is maintained by tree structure of Spans.

Tracer	- A Tracer creates spans containing more information about what is happening for a given operation, such as a request in a service
Trace Provider	- Is a factory of Tracers. Tracer Provider is initialized once and its lifecycle matches the application’s lifecycle.
Trace Record Exporters	-Send log records to a consumer.
Spam	-A **span** represents a unit of work or operation. Spans are the building blocks of Traces
Sampler	- Decides whether to export the Span Sampling is a mechanism to control the noise and overhead introduced by Open Telemetry by reducing the number of samples of traces collected and sent to the backend.

NimbusPay Technologies

Traces – Hierarchy Of Spans





Span

OpenTelemetry

A span represents an operation within a transaction.

Name		
Parent span ID		(empty for root spans)
Start &End Timestamps		
Span Context	Trace ID	Representing the trace that the span is a part of
	Span ID	
	Trace Flags	Binary encoding containing information about the trace
	Trace State	List of key-value pairs that can carry vendor-specific trace information
Attributes		key-value pairs that contain metadata that you can use to annotate a Span to carry information about the operation it is tracking Keys must be non-null string values Values must be a non-null string, Boolean, floating point value, integer, or an array of these values

Span

OpenTelemetry

A span represents an operation within a transaction.

Span Event		Can be thought of as a structured log message Typically used to denote a meaningful, singular point in time during the Span's duration
Span Links (Optional)		When we would like to associate the trace for the subsequent operations with the first trace, but we cannot predict when the subsequent operations will start. We need to associate these two traces, so we will use a span link.
Span Status	Unset	Operation it tracked successfully completed without an error.
	Error	Error occurred in the operation it tracks
	Ok	Means the span was explicitly marked as error-free by the developer of an application
Span Kind		Client, Server, Internal, Producer, or Consumer

Span Kind Meaning

OpenTelemetry

Span Kind	Synchronous	Asynchronous	Remote Incoming	Remote Outgoing
CLIENT	yes			yes
SERVER	yes		yes	
PRODUCER		yes		maybe
CONSUMER		yes	maybe	
INTERNAL				

Problem with Trace

OpenTelemetry

Problem: Traced apps can generate many traces with many spans

- Higher operational costs (Heady)
- Noise

Solution: Smart Sampling

- Head-based (sampled at the beginning of the trace). Head sampling is a sampling technique used to make a sampling decision as early as possible.
- Tail-based (sampled at the end of the trace)
  Tail sampling is where the decision to sample a trace takes place by considering all or most of the spans within the trace

The idea behind sampling is to control the spans you send to your observability backend, resulting in lower ingest costs. Sampling is a process that restricts the amount of spans that are generated by a system.

NimbusPay Technologies

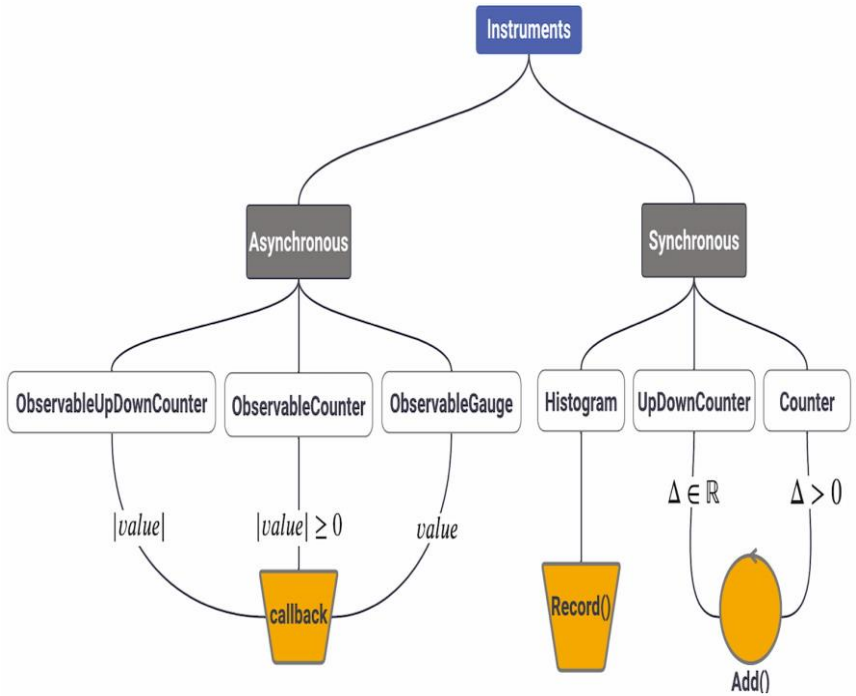
Metrics

OpenTelemetry

Meter	- A Meter creates metric instruments, capturing measurements about a service at runtime. Meters are created from Meter Providers.
Meter Provider	- Is a factory of Meter. Meter Provider is initialized once and its lifecycle matches the application's lifecycle.
Trace Record Exporters	-Metric Exporters send metric data to a consumer..
Metric Instrument	-Measurements are captured by **metric instruments**. A metric instrument is defined by: Name, Kind, Unit (Optional), Description (Optional)

NimbusPay Technologies

Metric -Instrument



Counter	A value that accumulates over time
Asynchronous Counter	Same as collector but collected once for each export.
UpDownCounter	A value that accumulates over time, but can also go down again, (Like queue)
Asynchronous UpDownCounter:	Same as Up Down Counter but collected once for each export.
Gauge	Measures a current value at the time it is read . (They are asynchronous)
Histogram	A client-side aggregation of values, such as request latencies

Metric Aggregation

OpenTelemetry

Aggregation: Large number of measurements are combined into either exact or estimated statistics about metric events that took place during a time window. The OTLP protocol transports such aggregated metrics

Temporality refers to the way additive quantities are expressed, in relation to time, indicating whether reported values incorporate previous measurements or not.

> ***Cumulative temporality*** means that successive data points repeat the starting timestamp. For example, from start time T_0, cumulative data points cover time ranges $(T_0, T_1]$, $(T_0, T_2]$, $(T_0, T_3]$, and so on.
>
> ***Delta temporality*** means that successive data points advance the starting timestamp. For example, from start time T_0, delta data points cover time ranges $(T_0, T_1]$, $(T_1, T_2]$, $(T_2, T_3]$, and so on

Reaggregation

OpenTelemetry

Temporal reaggregation:

Metrics that are collected at a high-frequency can be re-aggregated into longer intervals, allowing \low-resolution time-series to be pre-calculated or used in place of the original metric data.

Spatial reaggregation:

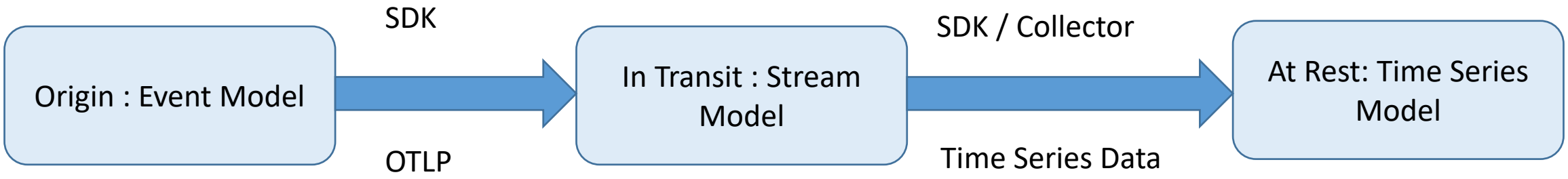
Metrics that are produced with unwanted attributes can be re-aggregated into metrics having fewer attributes.

Delta-to-Cumulative:

Metrics that are input and output with Delta temporality unburden the client from keep high-cardinality state

NimbusPay Technologies

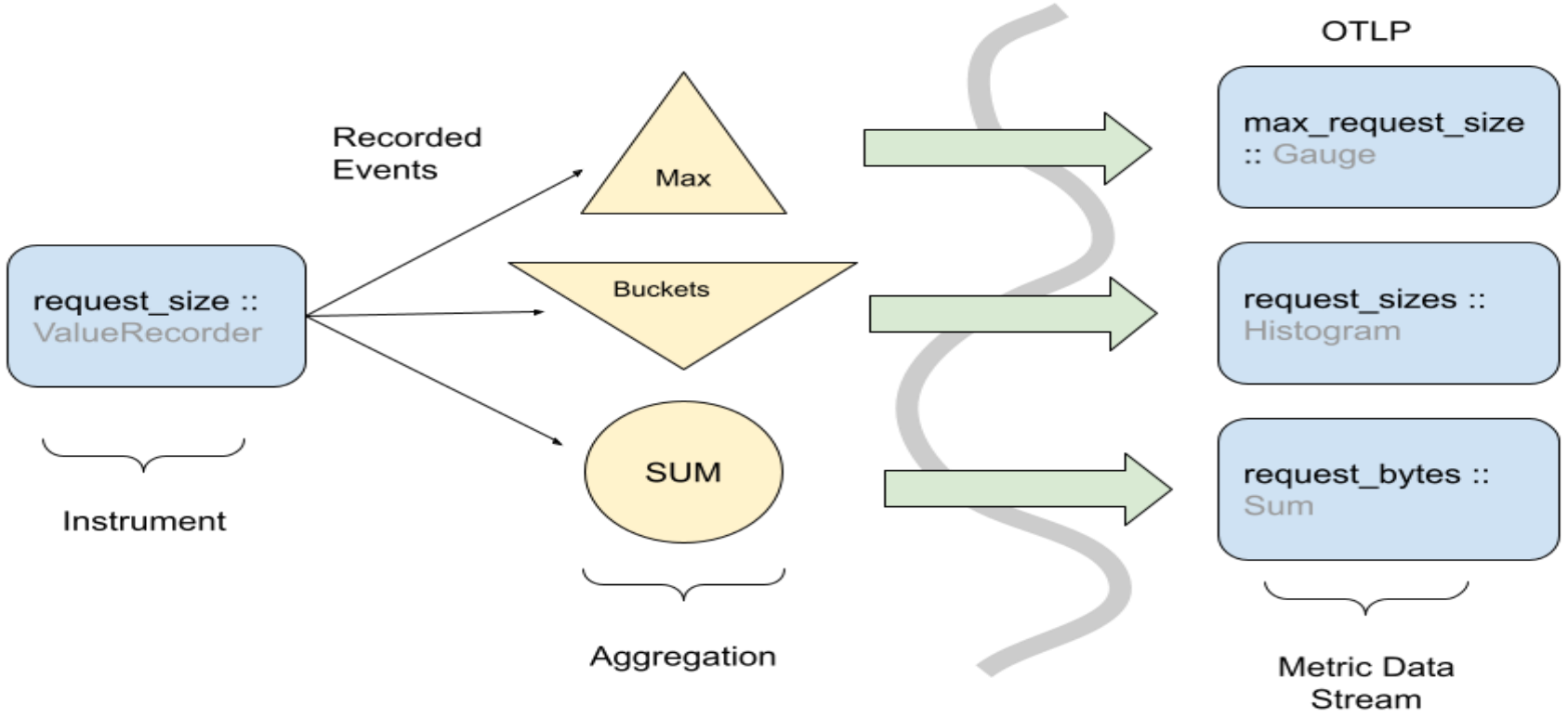
Metrics Data Model



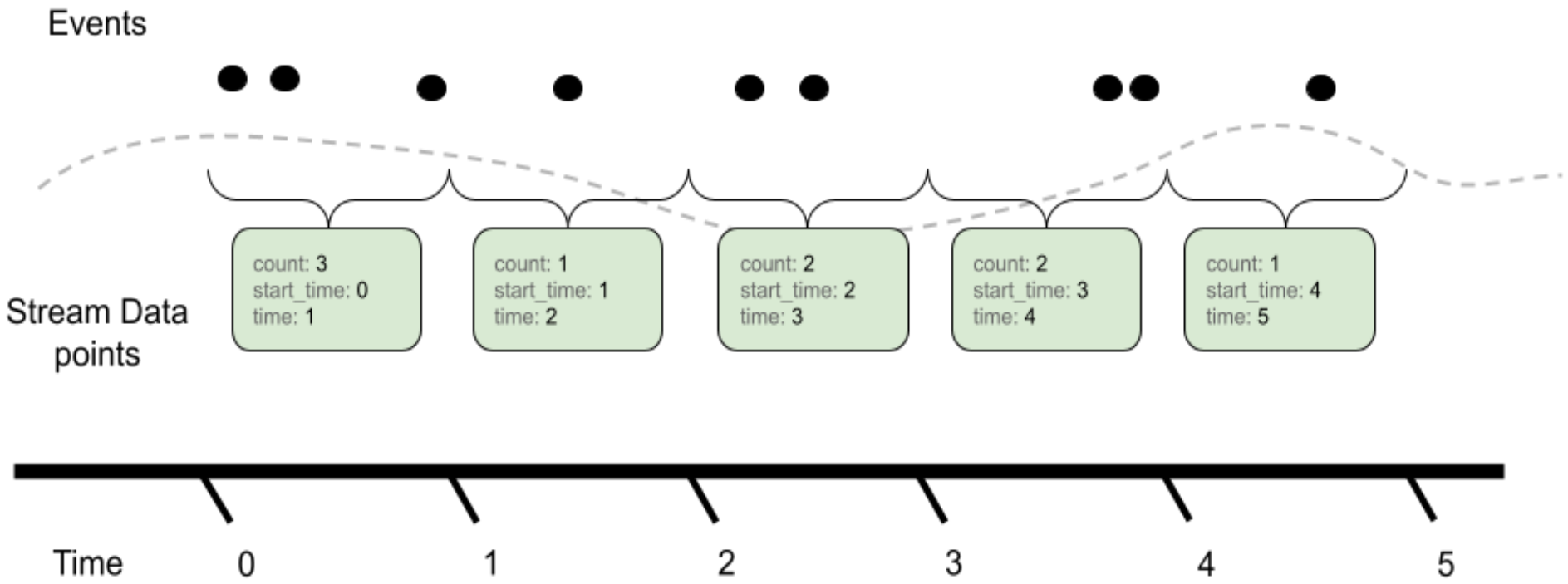
- An Event model, representing how instrumentation reports metric data.
- A Time-series model, representing how back-ends store metric data.
- A Metric Stream model, defining the *O*pen *Tel*emetry *P*rotocol (OTLP) representing how metric data streams are manipulated and transmitted between the Event model and the Time-series storage.

Metric Event Model

OpenTelemetry



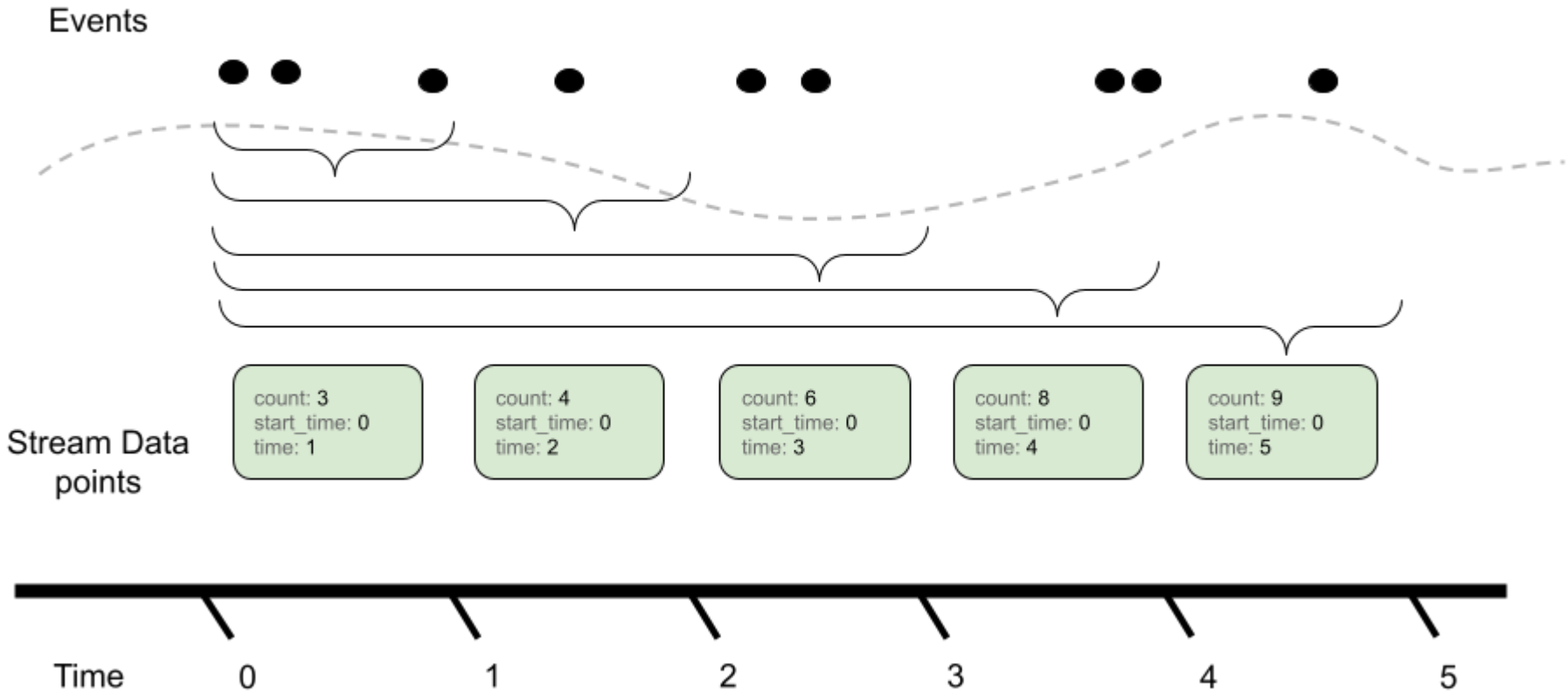
Metric Time Series- Sums - Aggregation temporality



- An *Aggregation Temporality* of delta or cumulative.

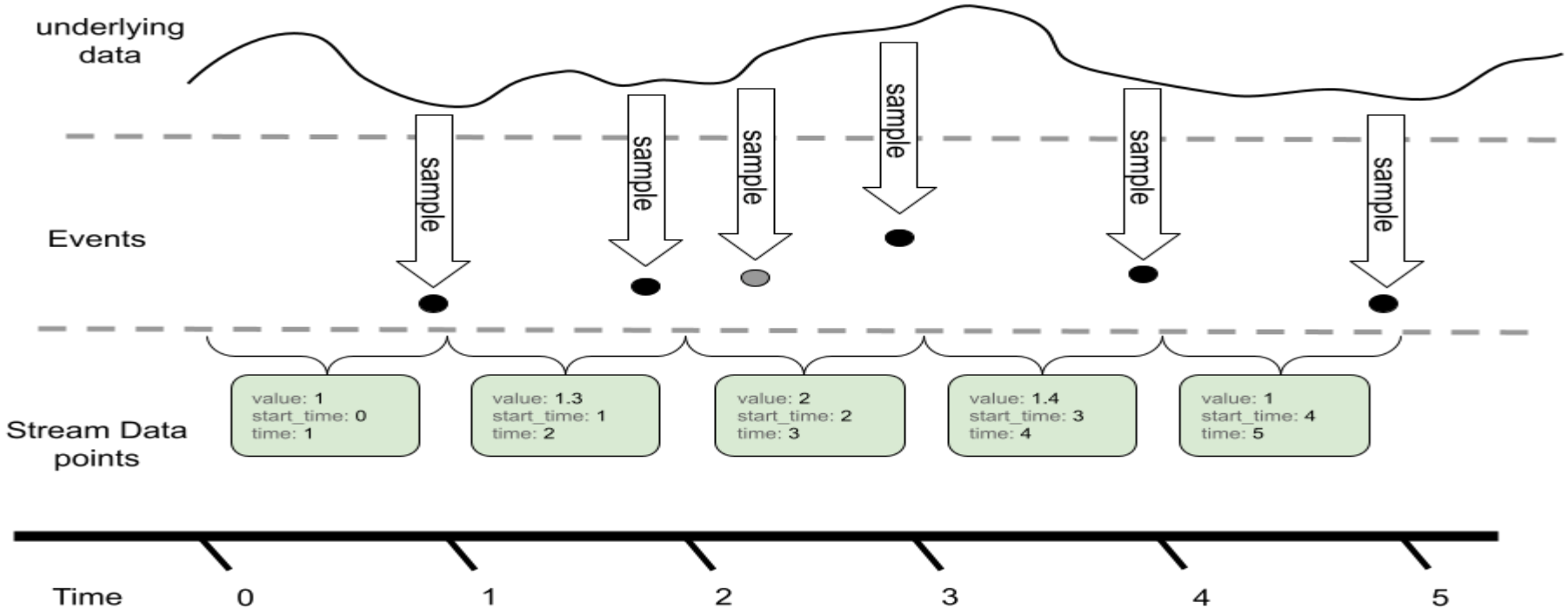
Metric Time Series- Sums – Cumulative Aggregation

OpenTelemetry



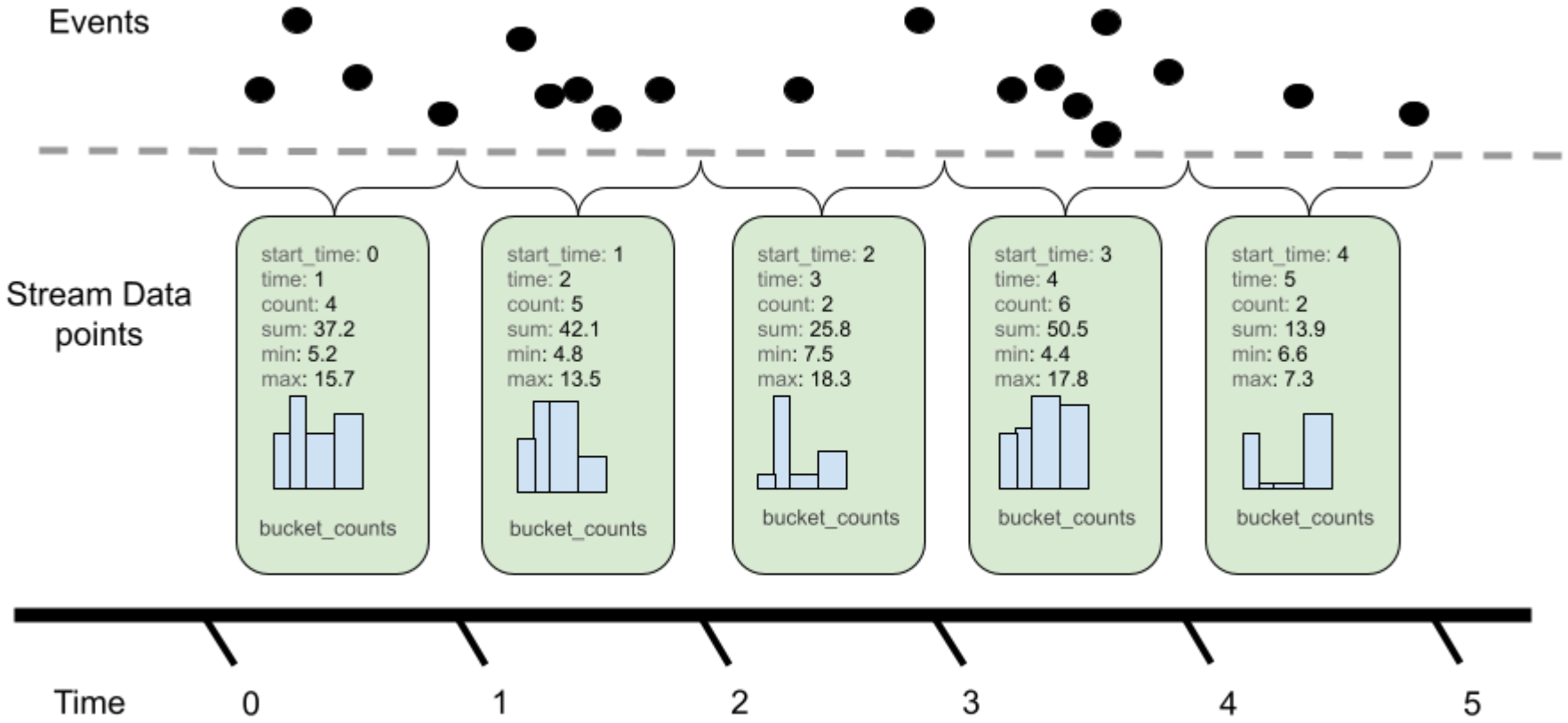
NimbusPay Technologies

Gauge – Time Series



only the last value is reported in the metric stream via OTLP

Histogram



An *Aggregation Temporality* of delta or cumulative.

Baggage – Additional Metadata

OpenTelemetry

- Baggage is a set of application-defined properties contextually associated with a distributed request or workflow execution.
- Baggage can be used, among other things, to annotate telemetry, adding contextual information to metrics, traces, and logs.
- It is represented as a set of name/value pairs describing user-defined properties
- Baggage names and values are any valid UTF-8 strings
- Baggage APIs -- Get Value, Get All Values, Set Value, Remove value
  These operations are basically for extracting/inserting baggage from Context

Context - This context contains information relevant to a specific operation or transaction, such as trace identifiers, span identifiers, and other metadata

Instrumentation Scope

The Instrumentation Scope represents a logical unit within the application code with which the emitted telemetry can be associated.

In your observability backend, this allows you to slice and dice your telemetry data by the Instrumentation Scope,

The Instrumentation Scope is defined by a name and version pair when a Tracer, Meter or Logger instance is obtained from a provider. Each span, metric or log record created by the instance will be associated with the provided Instrumentation Scope.

Resource

OpenTelemetry

Resource represents the entity producing telemetry as resource attributes – like running in a container on Kubernetes has a process name, a pod name, a namespace, and possibly a deployment name

A resource is added to the TraceProvider or MetricProvider when they are created during initialization. This association can not be changed later.

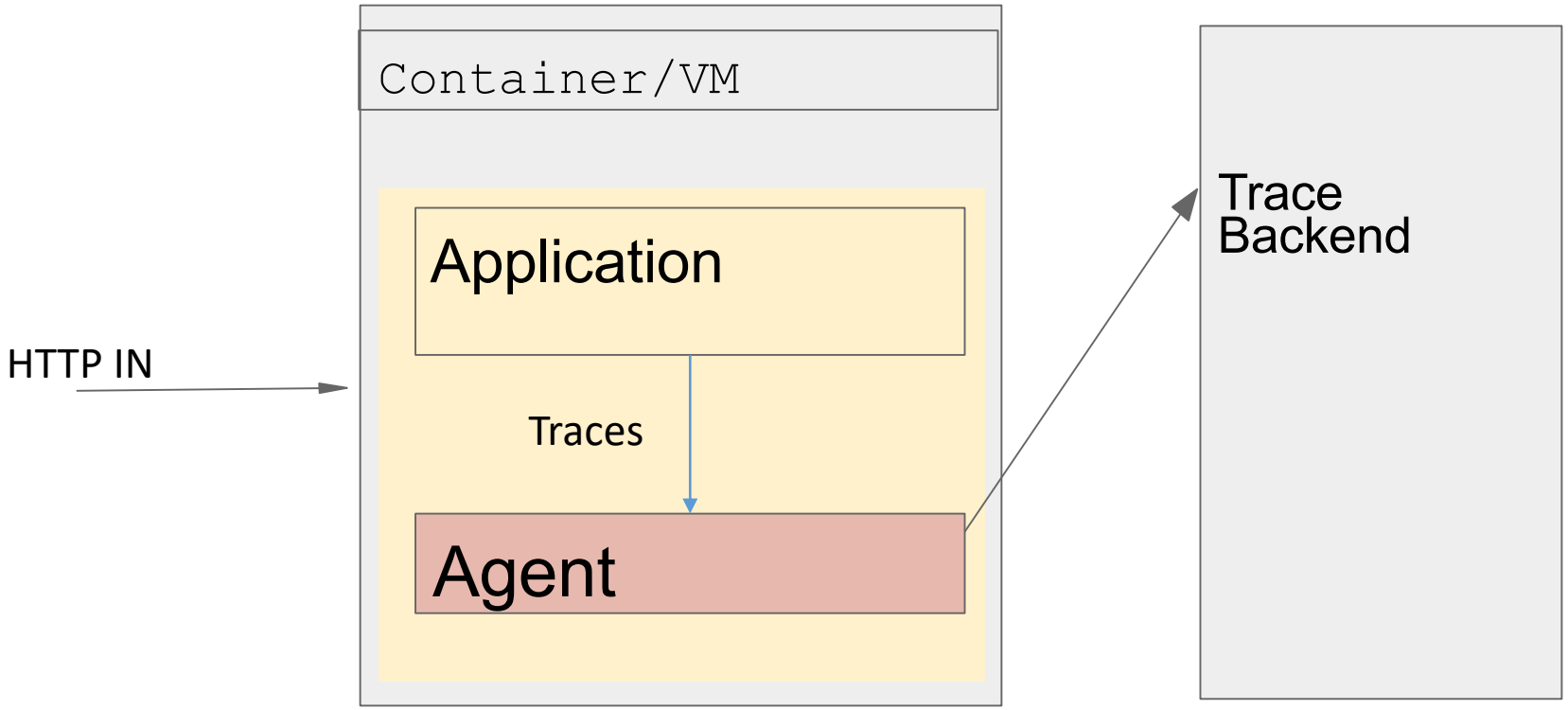
Resource detectors that can be used to automatically detect resource information from the environment.

- Operating System
- Host
- Process and Process Runtime
- Container
- Kubernetes
- Cloud-Provider-Specific Attributes
- and more

NimbusPay Technologies

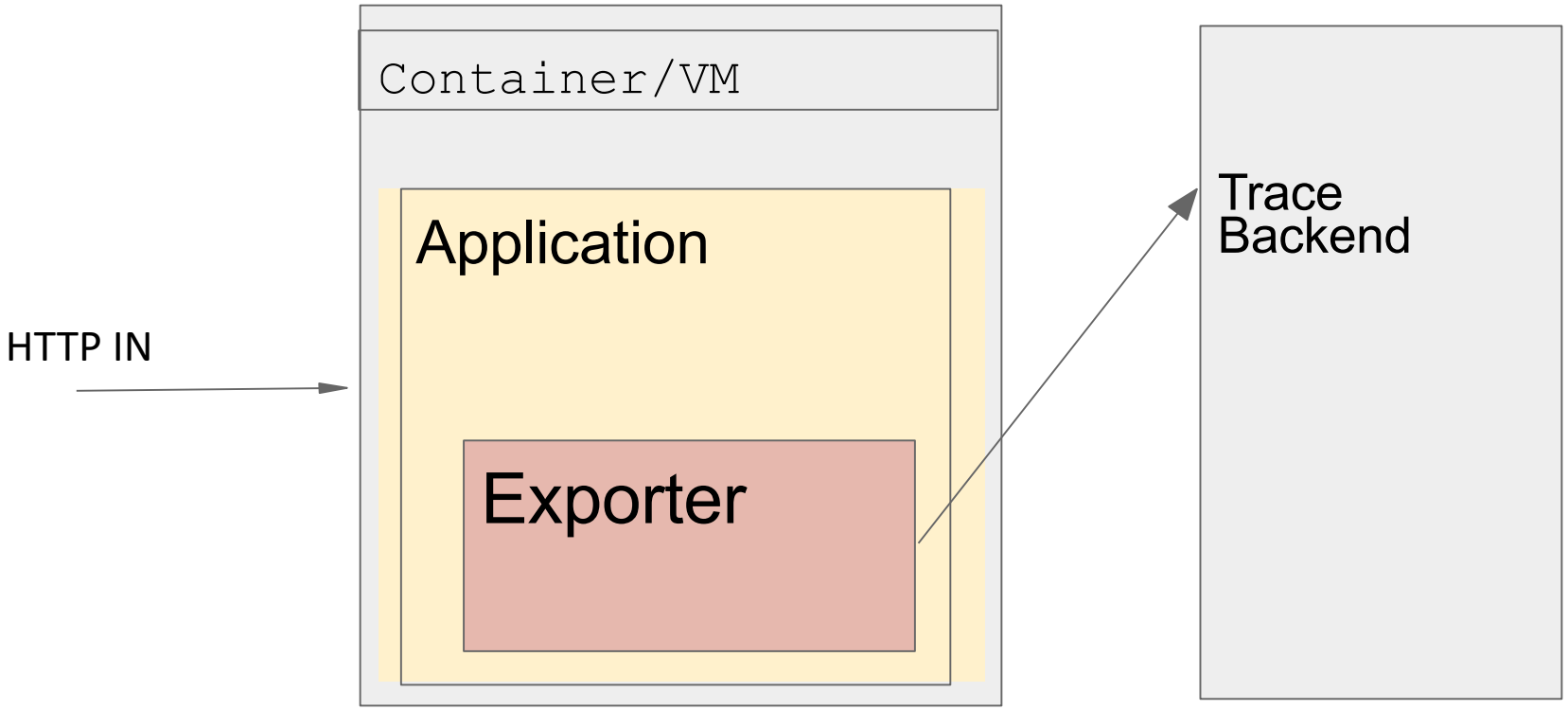
Telemetry Options - Agent

OpenTelemetry

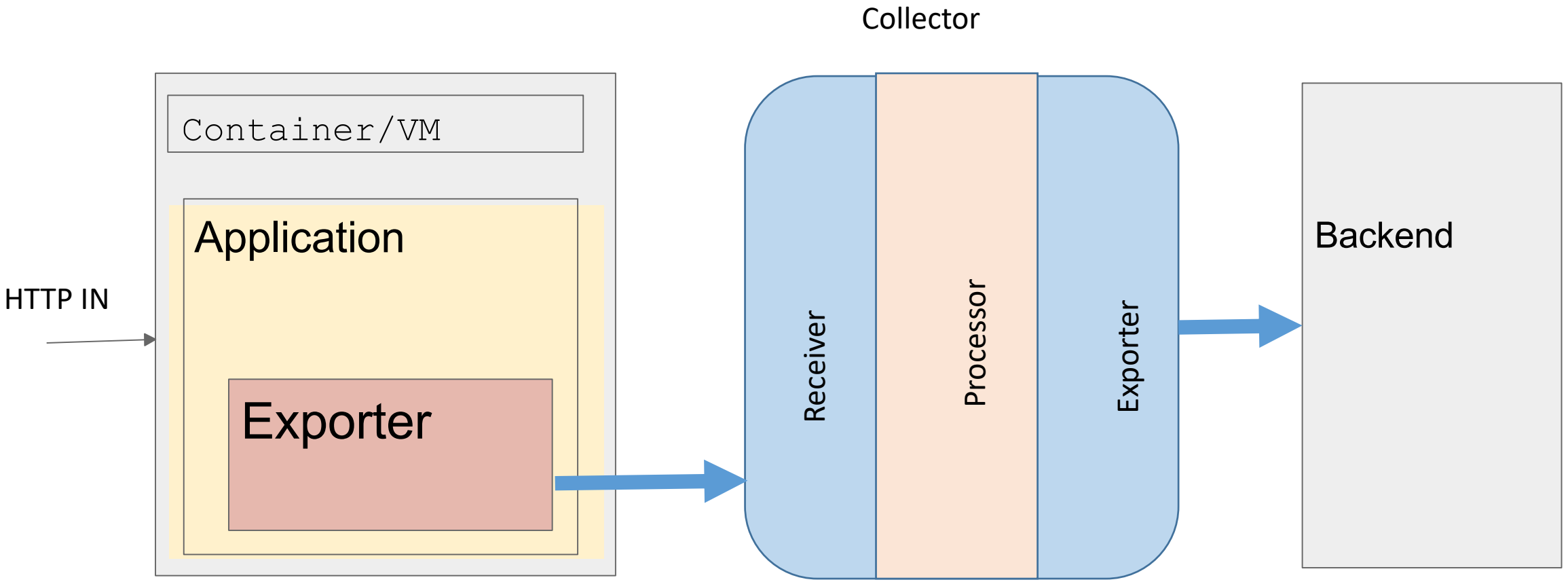


NimbusPay Technologies

Telemetry Options – Direct Export

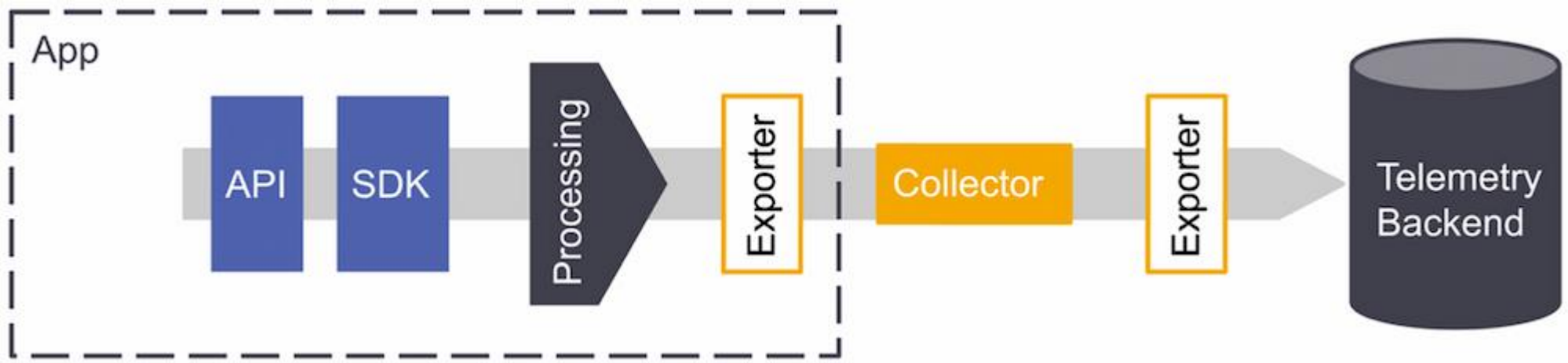


Telemetry Options – Collector Export



Telemetry Pipelines

OpenTelemetry



NimbusPay Technologies

Collector - Basics

What Are Next Topics

OpenTelemetry

- OTLP Specification 1.3.0 – OTLP/ gRPC protocol
- API and SDK – Classes
- Collector in Detail
- Collector Deployments and configurations
- Collector Troubleshooting
- Collector Patterns
- Various Exporters and Integrations
- Best Practices
- Open Telemetry – Echo System/ Components
  Jaeger, Prometheus, Zipkin etc.
- Advanced Topics
- Demo

NimbusPay Technologies

Why Otel

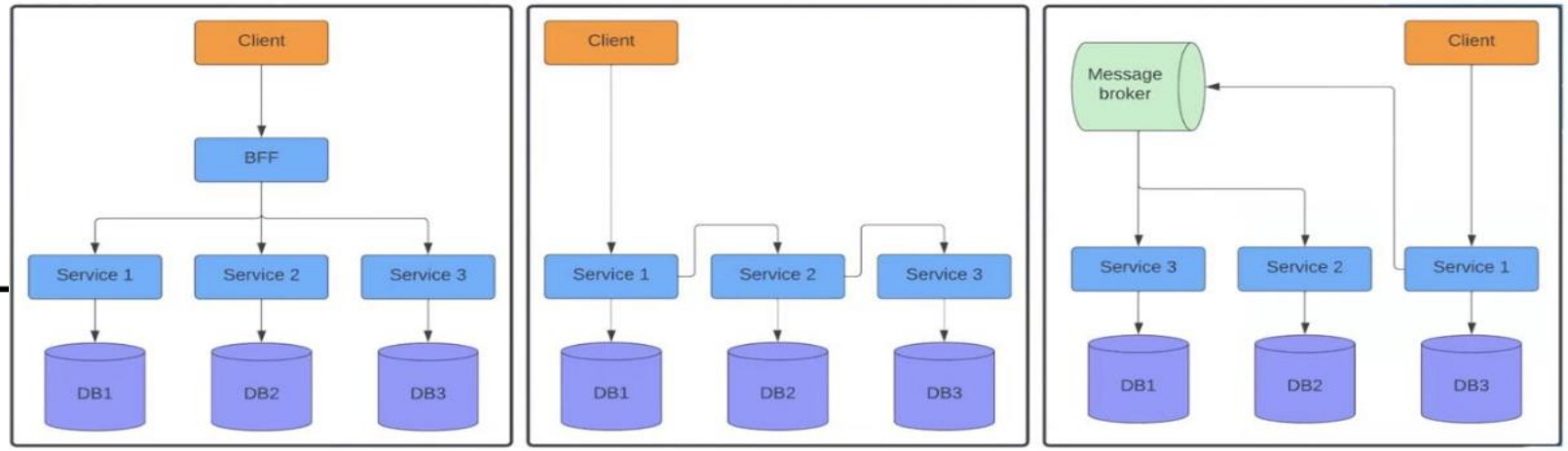
OpenTelemetry

Additional Slides

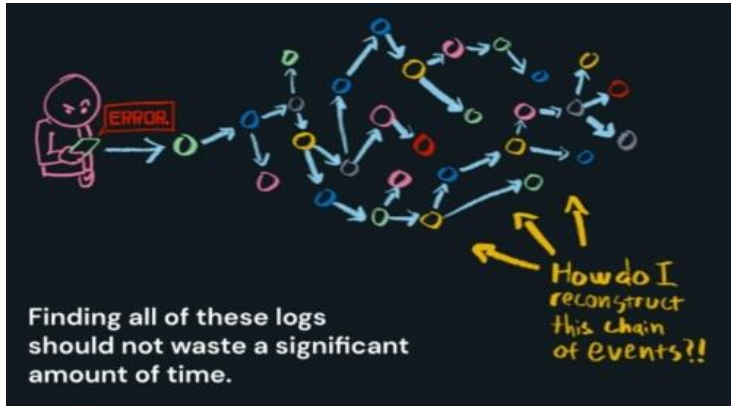
NimbusPay Technologies

OpenTelemetry

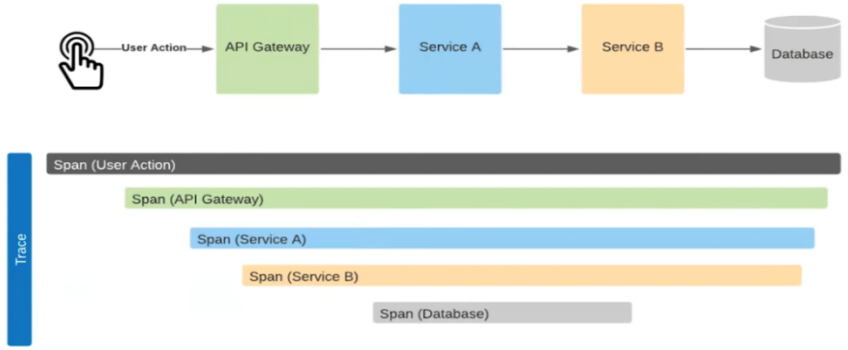
Current Distributed Systems

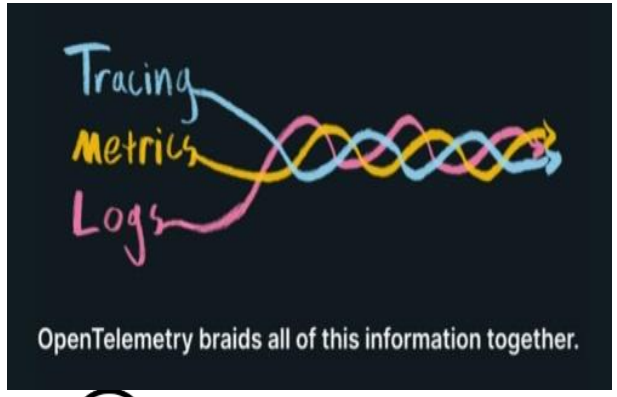


Problems



Otel Offering





NimbusPay Technologies

Sample Log, Metric, Trace-- JSON

log.json

metric.json

Trace.json

JSON Protobuf encoded payloads use proto3 standard defined JSON Mapping