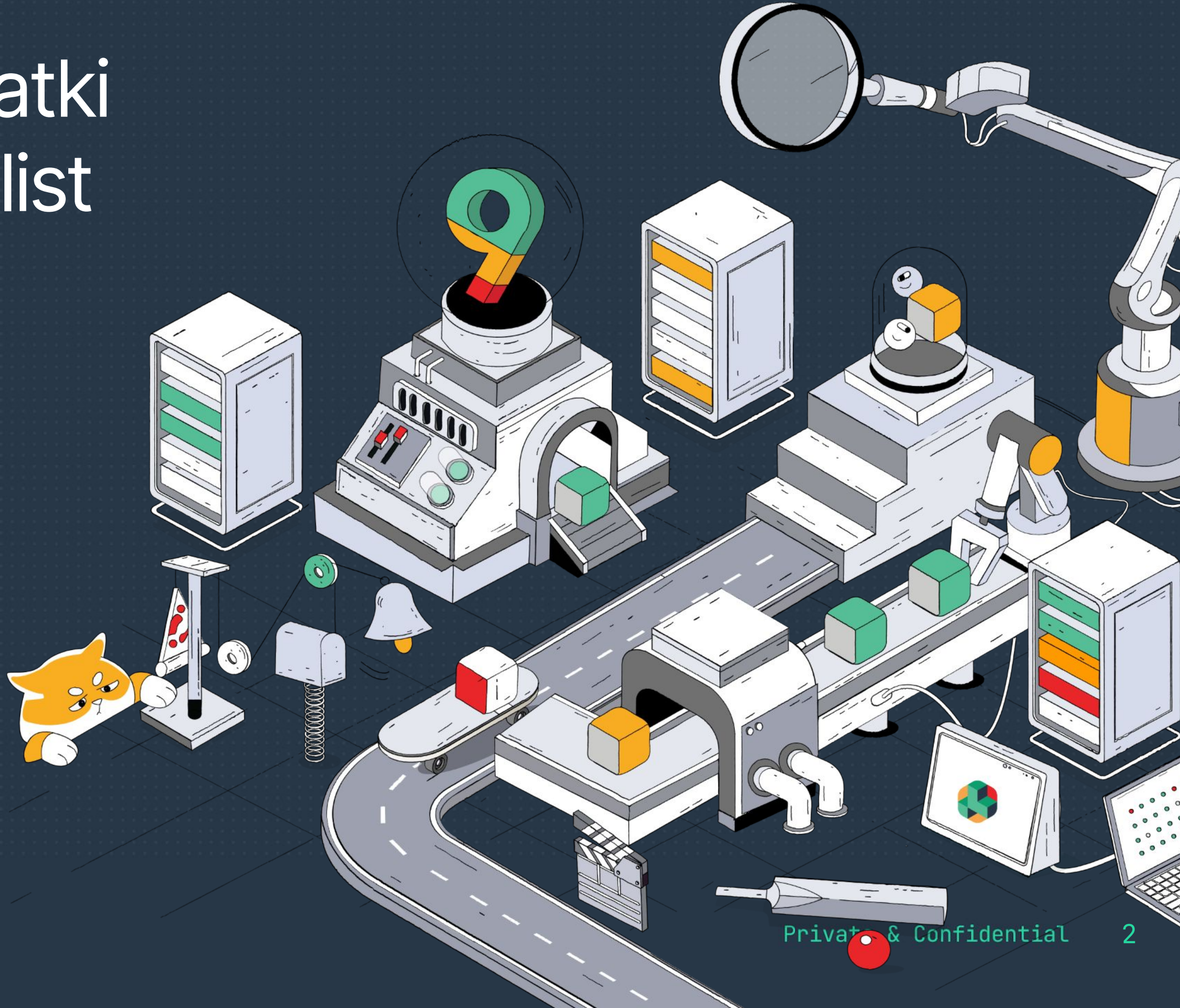
The Complete Handbook to OpenTelemetry Metrics.

Last9

Prathamesh Sonpatki
Developer Evangelist
Last9.io



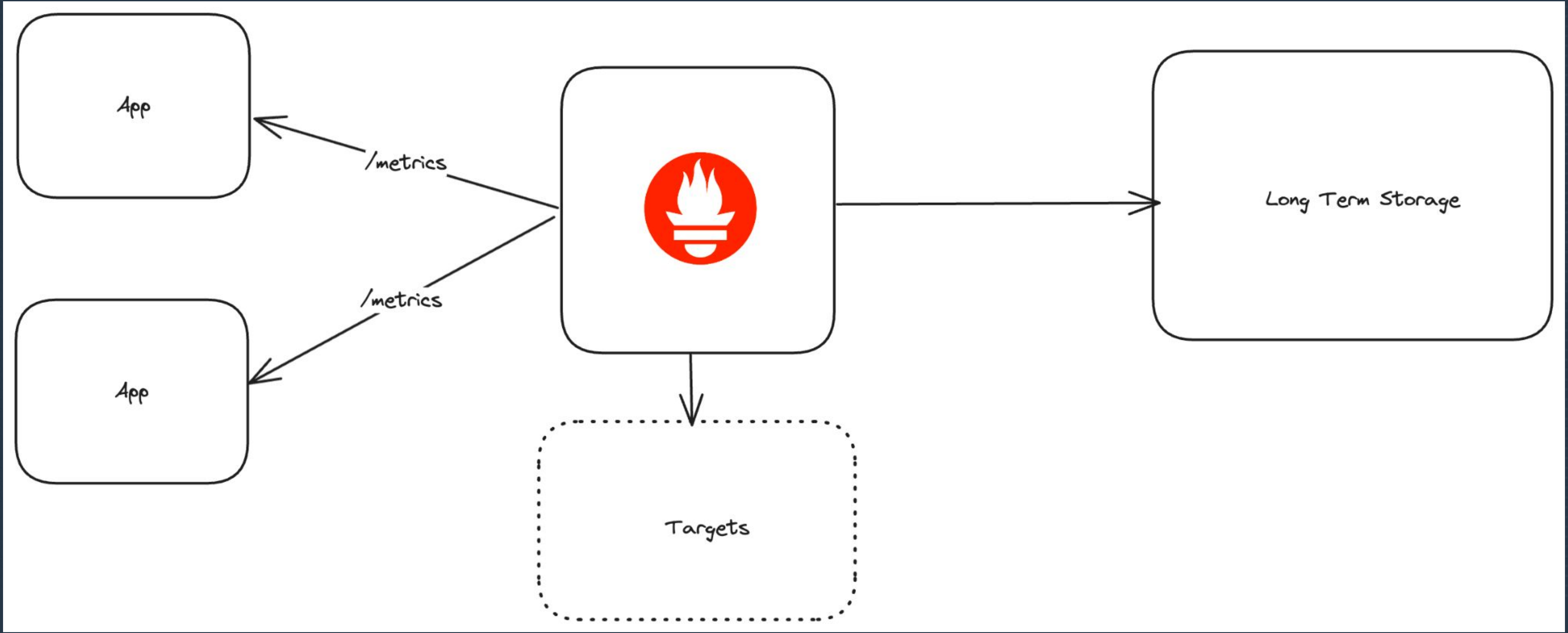
Agenda

- Why should you care?
- Prometheus vs. OpenTelemetry Metrics
- OpenTelemetry Collector
- OpenTelemetry Semantic Conventions
- Conversion Gotchas
- Temporality - Cumulative vs. Delta
- OpenTelemetry <> Prometheus @ Today
- OpenTelemetry <> Prometheus @ Tomorrow

Why should you care?

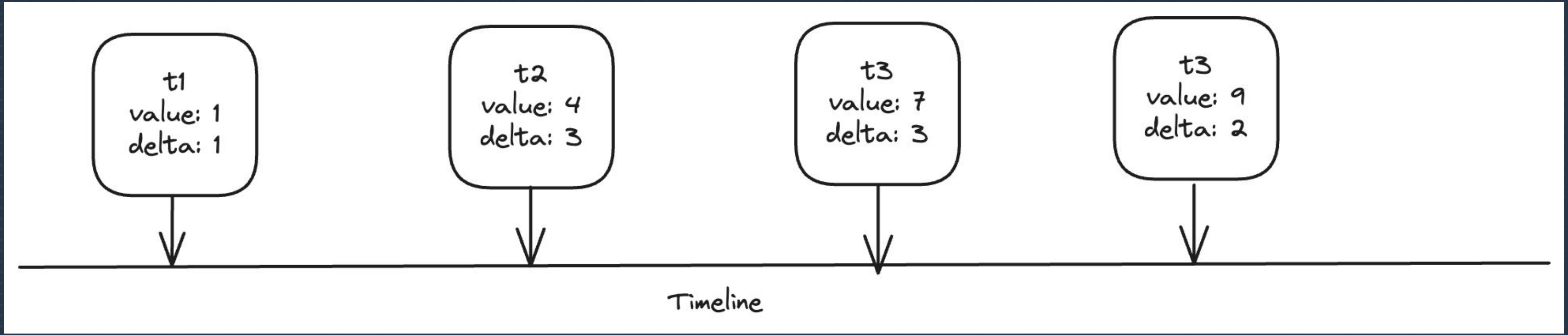
- OpenTelemetry is gaining wild attention and adoption is 🚀.
- It brings standardization.
- Vendor neutrality.
- Signal correlation.
- Support for more languages and SDKs for Otel metrics.
- Native support for OpenTelemetry Metrics in Prometheus is 🏗️.

Prometheus



Prometheus

- Scrape metrics from `/metrics`
- Optionally write to Remote Write Storages like Levitate
- Data is reported in Cumulatives



Prometheus

- Text Exposition Format
- OpenMetrics Format
- Float values
- Label based data model
- Pull based scrape model

OpenTelemetry

- OpenTelemetry is a collection of APIs, SDKs, and tools.
- Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.
- OpenTelemetry is GA.

From https://opentelemetry.io

OpenTelemetry

Standards & Specifications

SDKs, Client Libraries

Middleware Tools

OpenTelemetry Middleware Tools

Otel Collector

Otel Operator

Connects source to destination

Manages Collectors, self service instrumentation in K8s

OpenTelemetry

- OpenTelemetry does not have storage backends.
- It can work with multiple backends such as Levitate, Prometheus, New Relic, Datadog.

The OpenTelemetry Promise

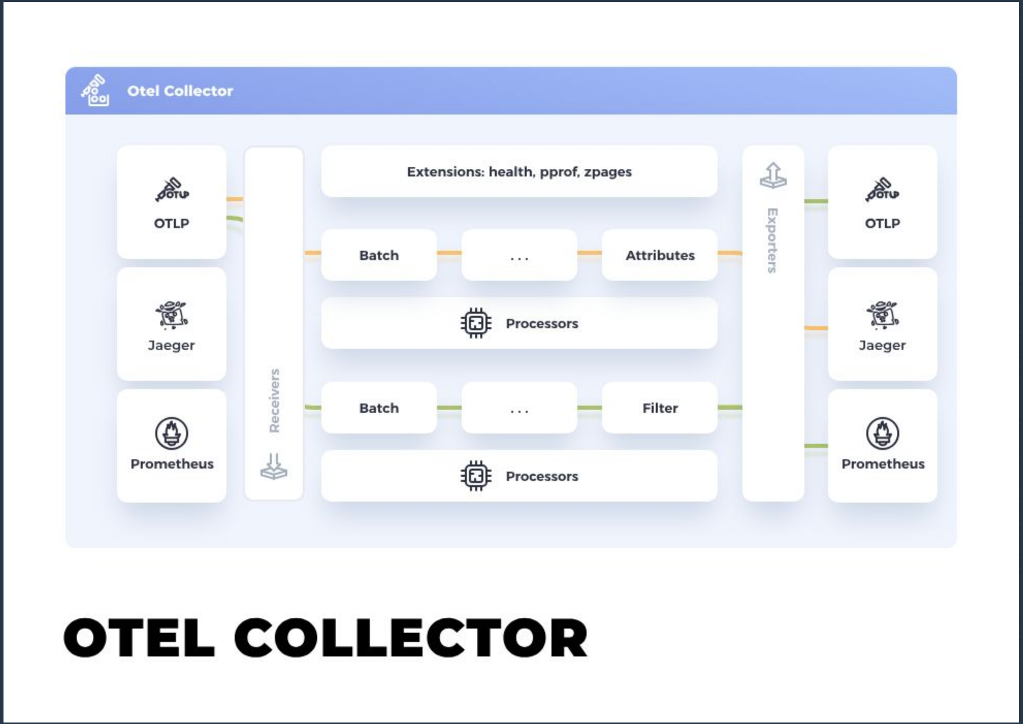
- Vendor neutral
- Semantic Conventions
- Signal Correlation
- Better Performance?

OpenTelemetry Metrics Project Goals

- Being able to connect metrics to other signals.
- Providing a path to OpenCensus customers to migrate to OpenTelemetry
- Working with existing metrics instrumentation standards and protocols such as Prometheus and statsD.

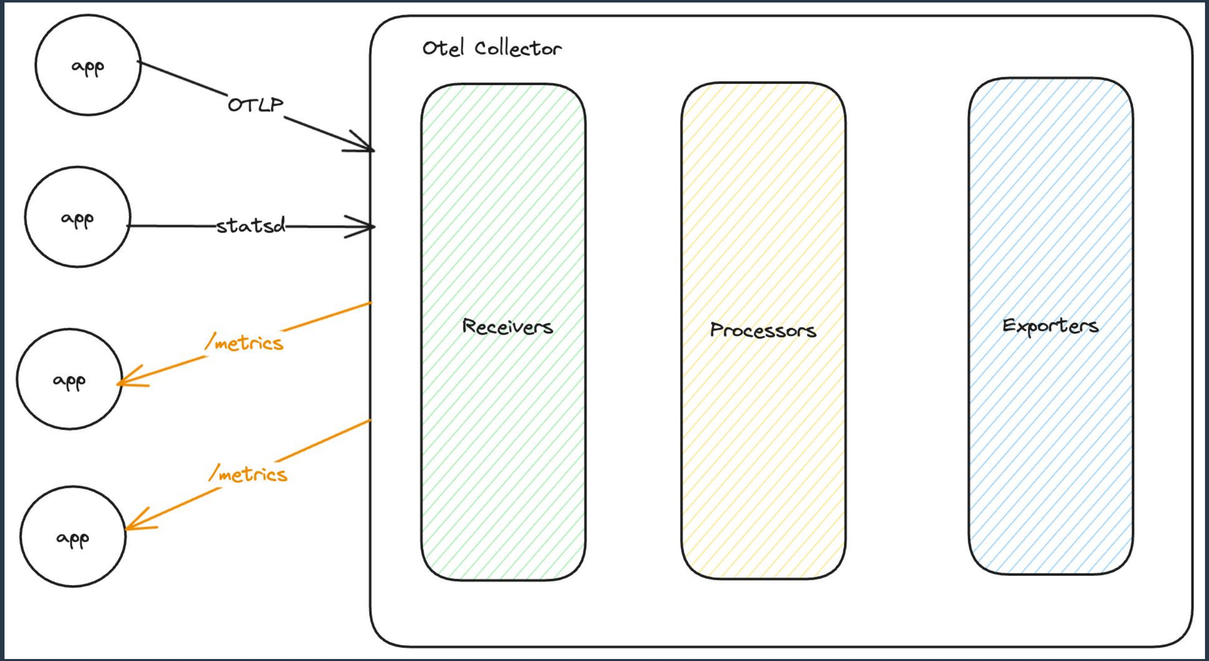
From https://opentelemetry.io/docs/specs/otel/metrics/

OpenTelemetry Collector

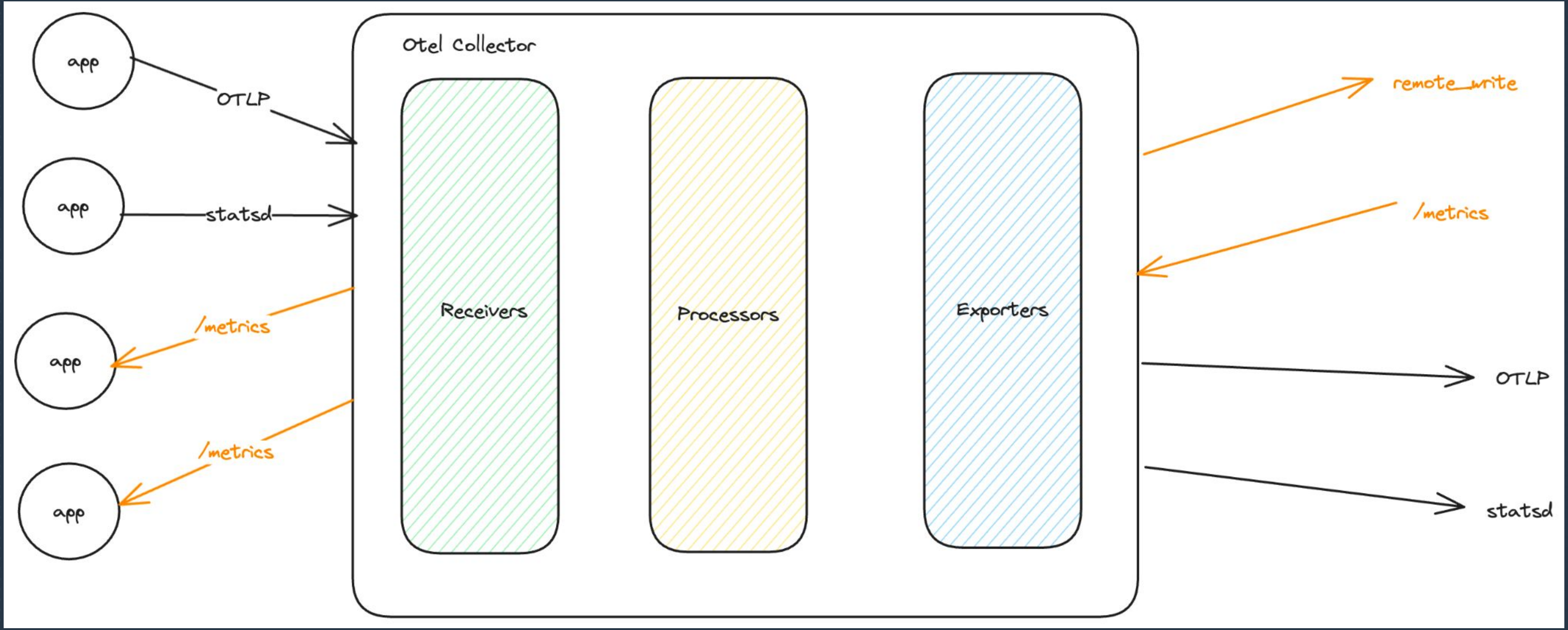


From https://opentelemetry.io

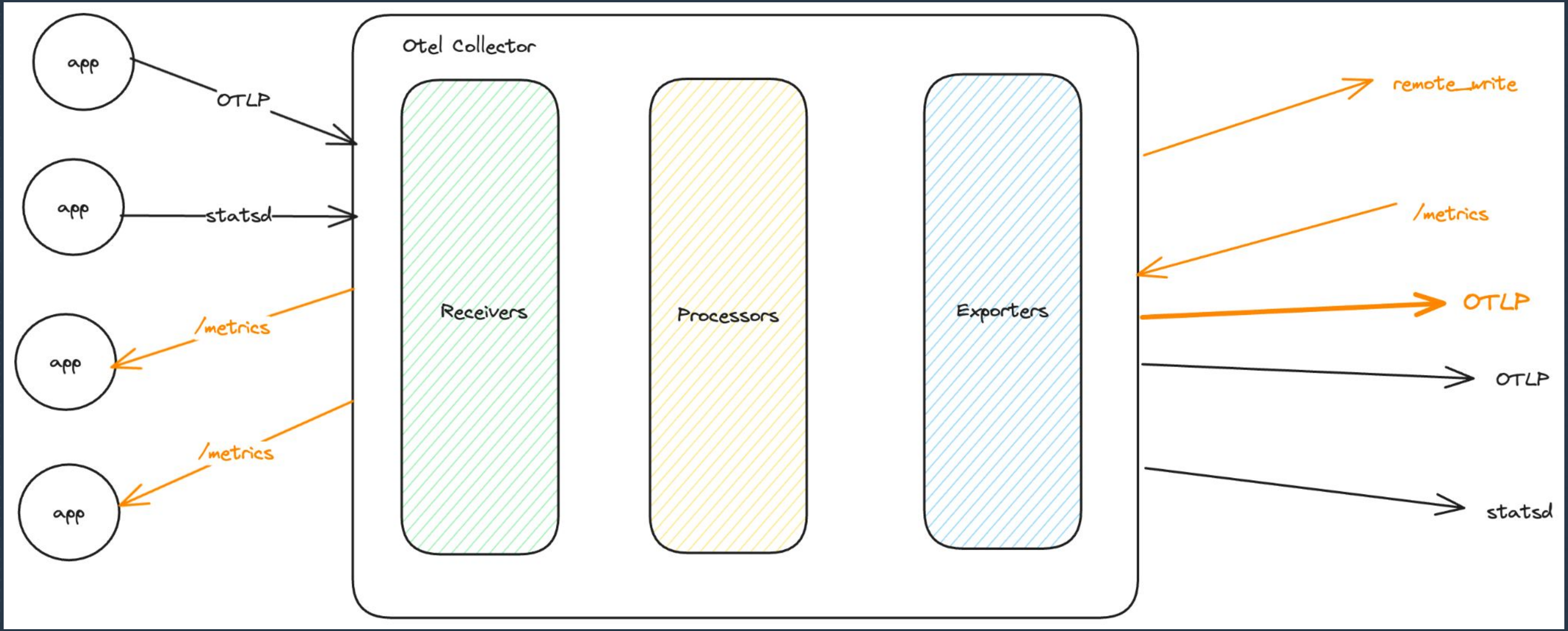
OpenTelemetry Collector



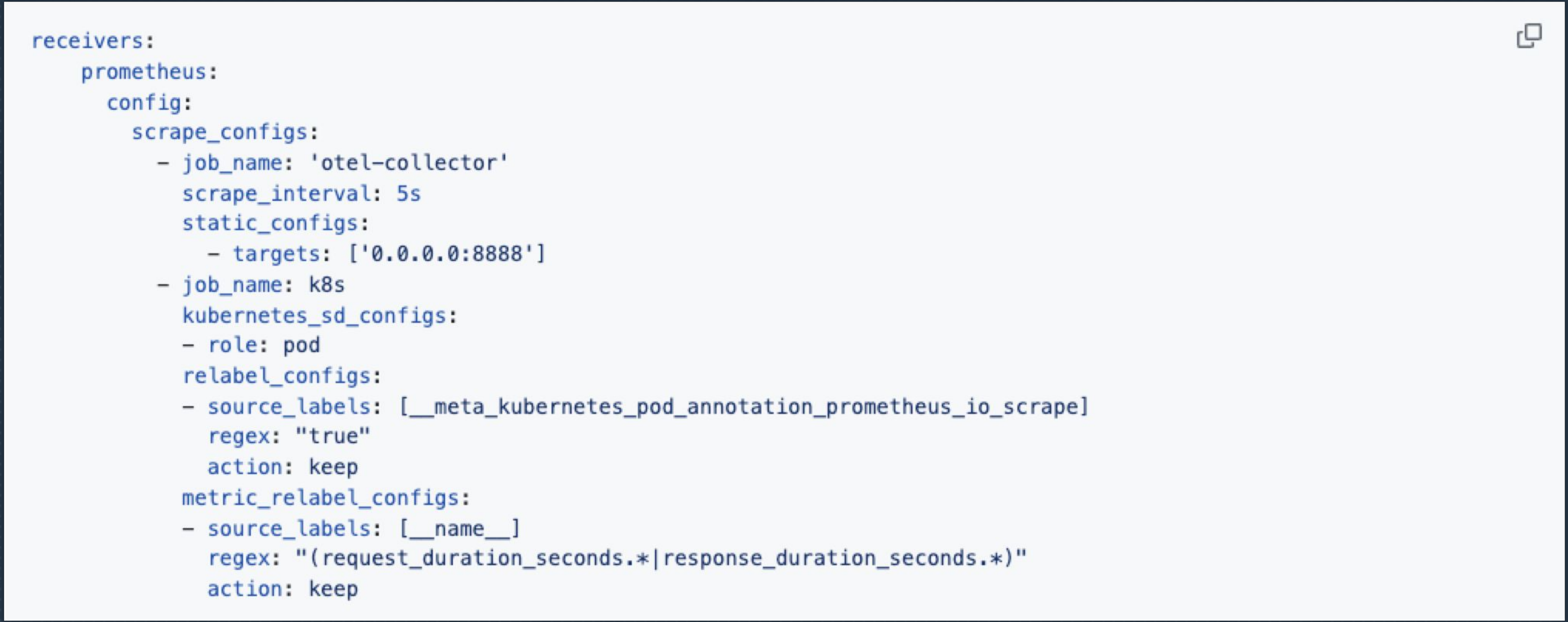
OpenTelemetry Collector



OpenTelemetry Collector



Prometheus Receiver



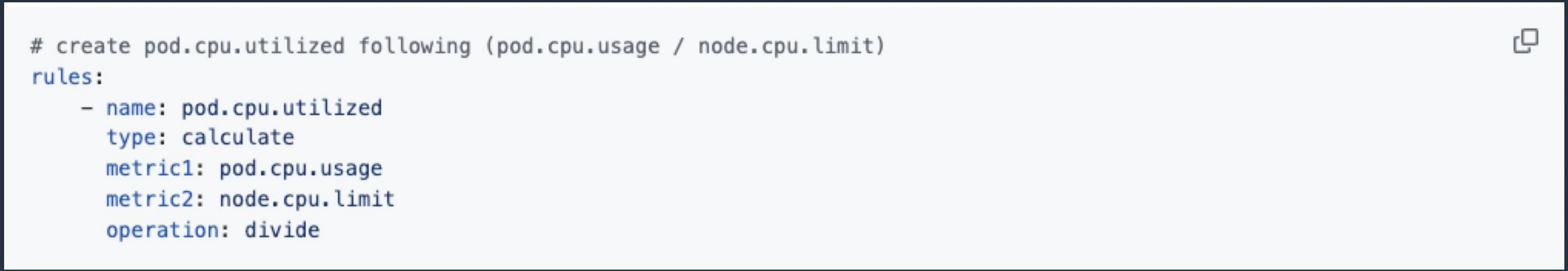
```yaml
receivers:
  prometheus:
   config:
    scrape_configs:
     - job_name: 'otel-collector'
      scrape_interval: 5s
      static_configs:
       - targets: ['0.0.0.0:8888']
     - job_name: k8s
      kubernetes_sd_configs:
      - role: pod
      relabel_configs:
      - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
       regex: "true"
       action: keep
      metric_relabel_configs:
      - source_labels: [__name__]
       regex: "(request_duration_seconds.*|response_duration_seconds.*)"
       action: keep
```

Processors

- Batch processor
- Memory Limiter
- Redaction
- Attributes

Processors

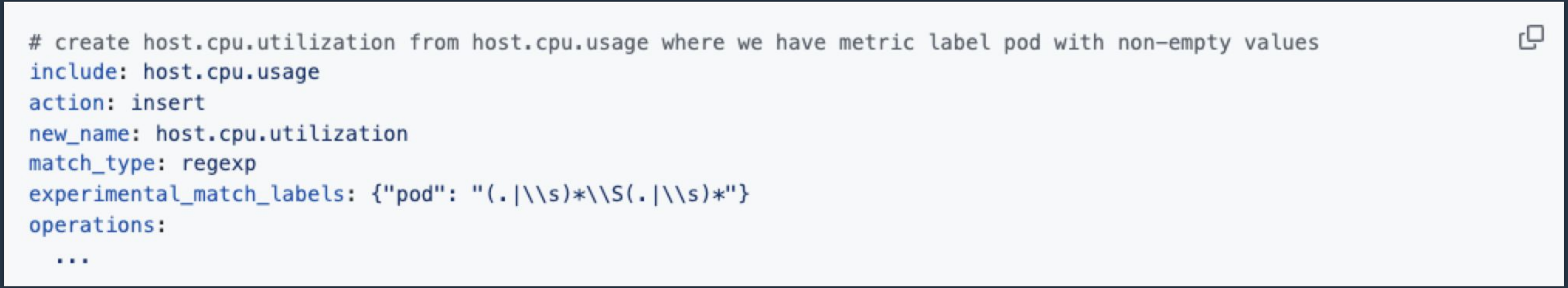
- Metric Generation



```
# create pod.cpu.utilized following (pod.cpu.usage / node.cpu.limit)
rules:
    - name: pod.cpu.utilized
      type: calculate
      metric1: pod.cpu.usage
      metric2: node.cpu.limit
      operation: divide
```

Processors

- Metric Transformation
- Rename
- Drop
- Aggregate
- High Cardinality workflows



```
# create host.cpu.utilization from host.cpu.usage where we have metric label pod with non-empty values
include: host.cpu.usage
action: insert
new_name: host.cpu.utilization
match_type: regexp
experimental_match_labels: {"pod": "(.|\\s)*\\S(.|\\s)*"}
operations:
  ...
```

Last9

Exporters

- Scrape `/metrics` exposed by Collector
- Remote Write from collector to long term storage like Levitate
- OTLP push to Prometheus

https://last9.io/blog/native-support-for-opentelemetry-metrics-in-prometheus/

Shipping Otel Metrics to Prometheus

- Different Metric Types!
- Cumulative vs. Delta Temporality!
- Different naming conventions!
- Different data types!
- Out of Order Metrics!

Different Metric Types

Otel Metrics

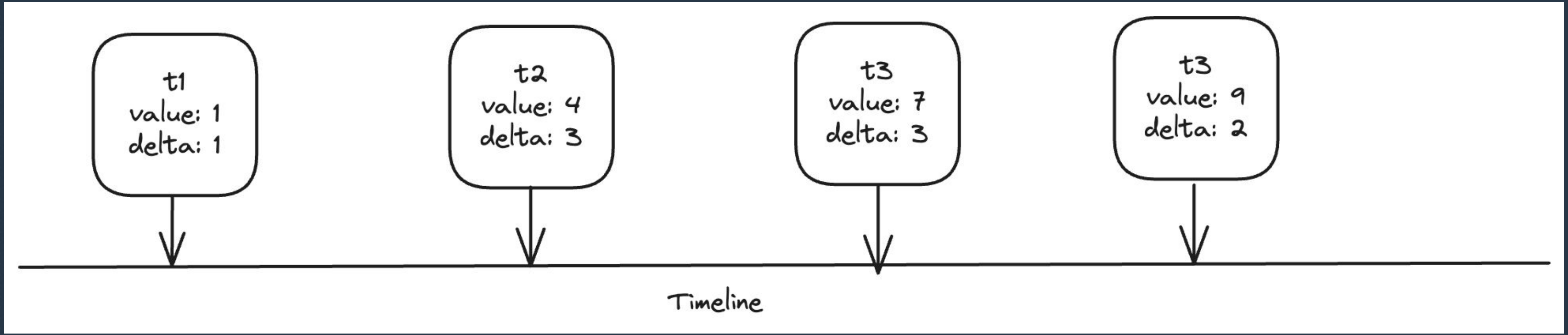
- Counter
- Asynchronous Counter
- UpDown Counter
- Asynchronous UpDown Counter
- Gauge
- Histogram

Prometheus Metrics

- Counter
- Gague
- Summary
- Histogram
- Sparse Histograms

Cumulative vs. Delta

- Cumulative temporality means that the value will be the overall value since the start of the measurement.
- Delta temporality means that the value will be the difference in the measurement since the last time it was reported.



Naming Conventions

- Otel → `http.requests.duration` with unit milliseconds
- Prometheus → `http_requests_duration_milliseconds_count`
- Prometheus receiver & exporters support normalization
- No conversion between units
- Prometheus → Otel Metrics is also possible

OpenTelemetry Metrics @ Today

- API Specification
- SDKs
- Collector
- Exporters
- Processors
- Receivers
- Push vs. Pull mechanism

Prometheus <> OpenTelemetry @ ~~Tomorrow~~ soon..

- OOO support enhancement
- UTF-8 support for label and metric names
- Delta Temporality support
- Handle OTEL resource attributes
- Store metric metadata in Prometheus
- Performance improvements

Recap

- Why should you care?
- Prometheus vs. OpenTelemetry Metrics
- OpenTelemetry Collector
- OpenTelemetry Semantic Conventions
- Conversion Gotchas
- Temporality - Cumulative vs. Delta
- OpenTelemetry <> Prometheus @ Today
- OpenTelemetry <> Prometheus @ Tomorrow

Thank you!

[@prathamesh2_]
[@last9io]
[Levitate - Otel compatible TSDB]

Last9