Grafana Labs

OTel me more!
A practical guide for adopting Open Telemetry

Kyle Eckhart
Principal Software Engineer
Grafana Labs

Kyle Eckhart
Principal Software Engineer
Grafana Labs

About me

- Live outside of Cleveland
- Dad of two boys (6 and 10)
- Play board games and video games
- I love being outside

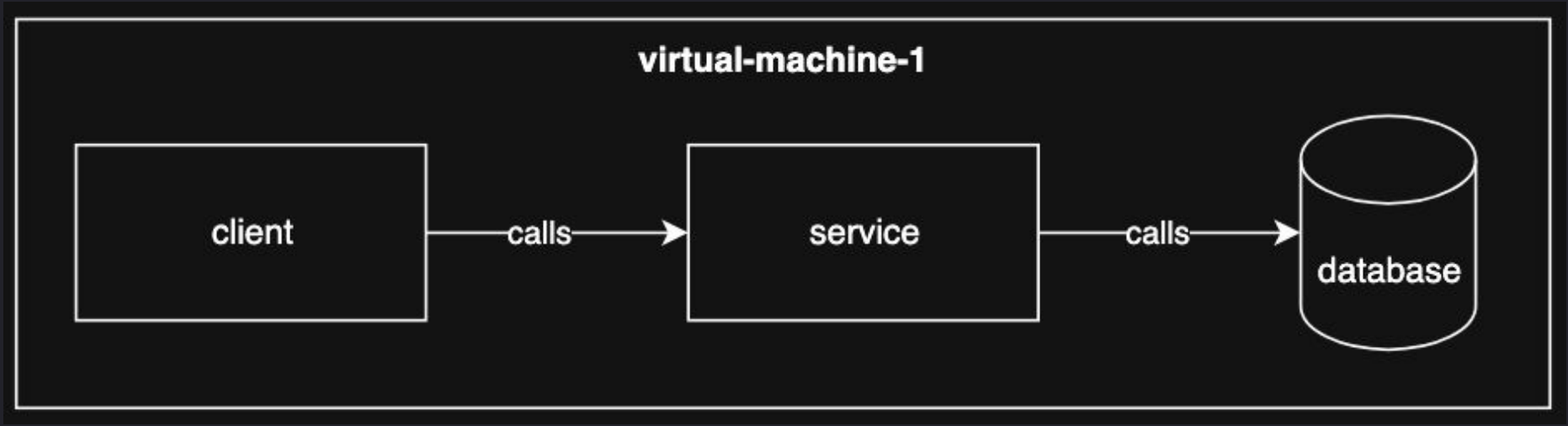
- Backend developer for 15 years: **golang,** dotnet, Kotlin/JVM
- Work on Grafana Alloy - Grafana's Open Telemetry Distribution
- I love efficient systems

Observability and the core signals

Instrumenting and collecting Telemetry

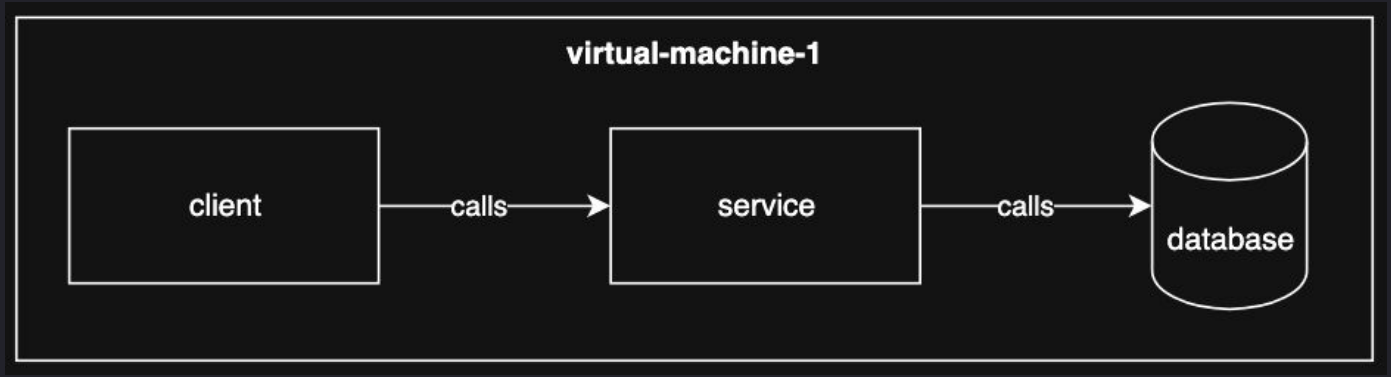
Intelligently using your Telemetry

What is observability?


virtual-machine-1
client
calls
service
calls
database

Monitoring Our System

- Tell me when CPU > 80%
- Tell me when Memory > 80%
- Tell me when average response times are > 1 sec



Monitoring vs Observability

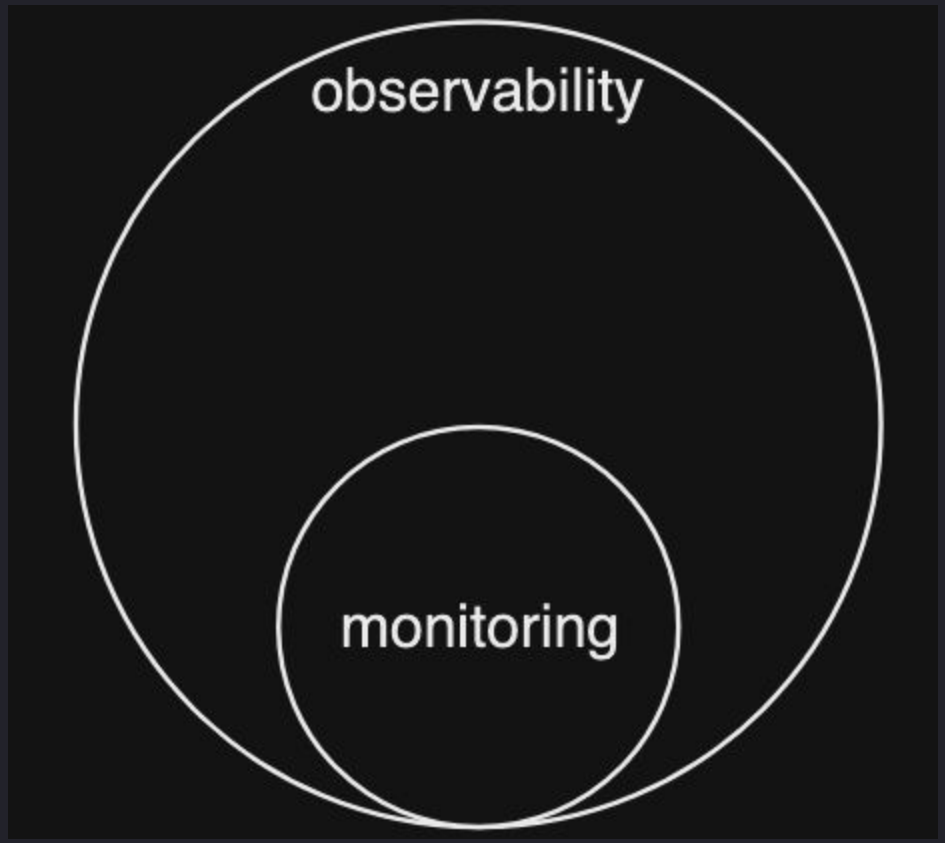
Monitoring

- CPU > 80%
- Memory > 80%
- Average response times are > 1 sec

vs

Observability

- Traffic doubled so CPU > 80%
- The APIs being called are resource intensive so Memory > 80%
- The machine is starved for resources so average response times are > 1 sec

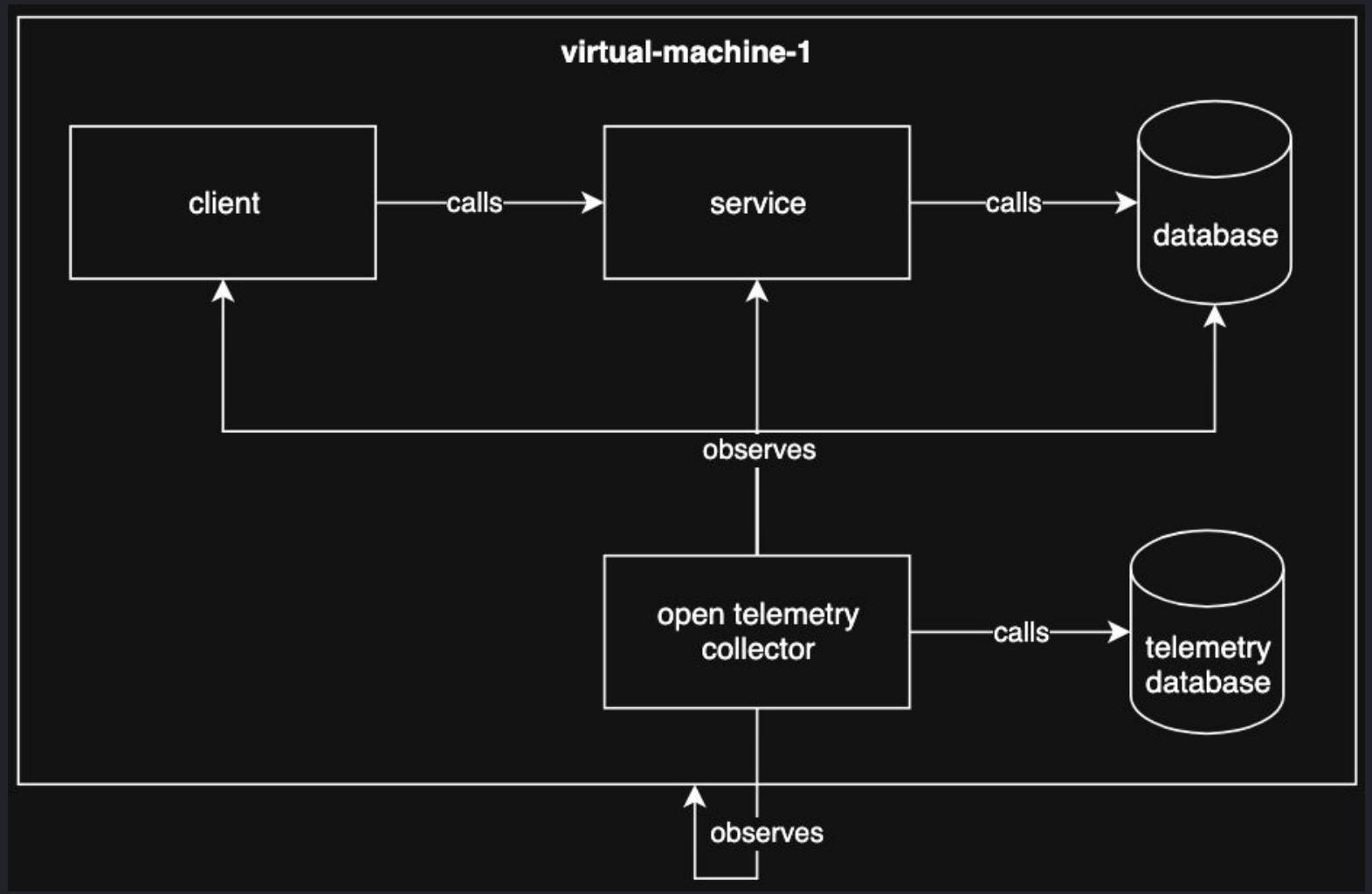

observability
monitoring

Grafana Labs

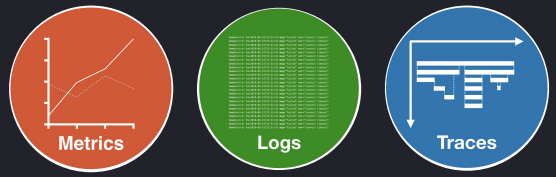
Open Telemetry (OTel)

Is a **vendor-neutral open source** Observability framework...

https://opentelemetry.io/docs/


virtual-machine-1
client
calls
service
calls
database
observes
open telemetry collector
calls
telemetry database
observes

And many many more



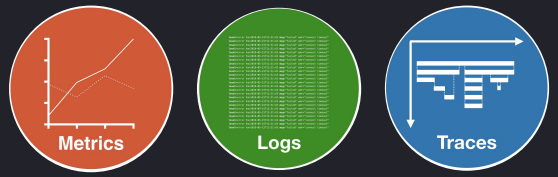
OpenTelemetry semantic conventions 1.38.0

The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications



OpenTelemetry semantic conventions 1.38.0

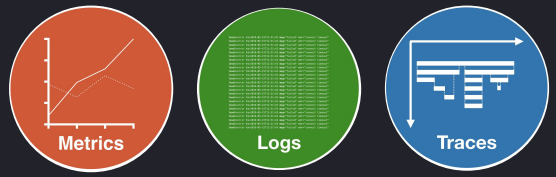
The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

Instrumentation



OpenTelemetry semantic conventions 1.38.0

The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

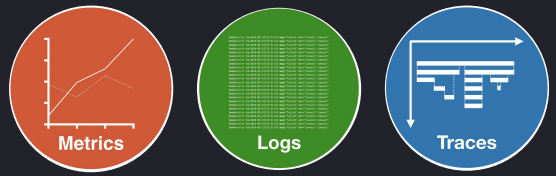
Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

Instrumentation

Collecting, Transforming, and Exporting



Specifications

Metrics

- A measurement captured at runtime
- They have,
  - A Name: http.client.request.duration
  - Optional Unit: seconds
  - Optional Description
  - **A Kind**
  - **Attributes**

Metric Kind: Counter

- A value that accumulates over time only going up

Metric Kind: Counter

Metric Kind: Gauge / UpDownCounter

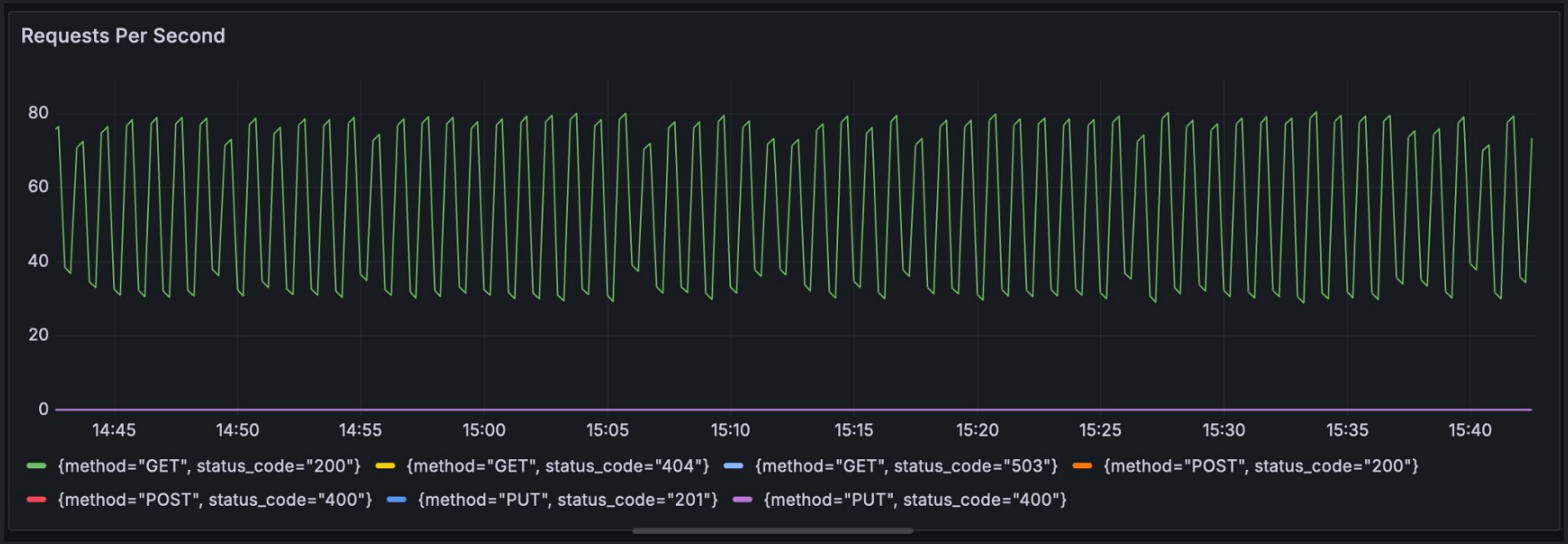
- Tracks the current value at the time it is read

Metric Kind: Histogram

- Tracks the statistical distribution of an event in a system

Metric Attributes

Metric Attributes



Logs

- A timestamped record of an event with metadata
- Structured (preferred)

```
{
    "timestamp": "2022-12-23T12:34:56Z",
    "level": "error",
    "message": "There was an error processing the request",
    "request_id": "1234567890",
    "user_id": "abcdefghij"
}
```

- Unstructured

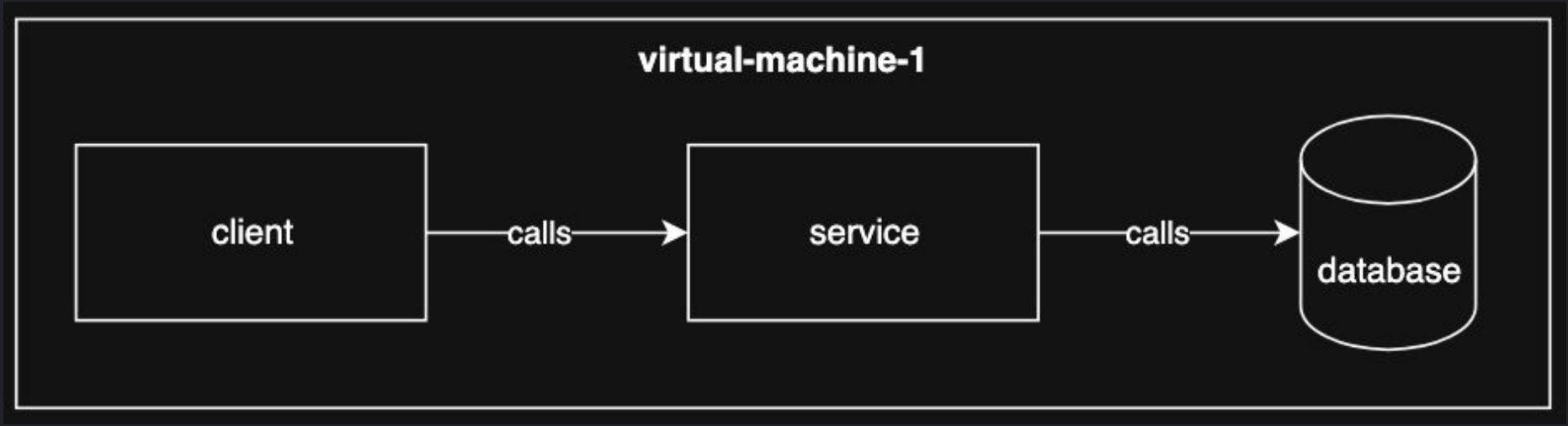
```
TLSv1.2 AES128-SHA 1.1.1.1 "Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0"
TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 3.3.3.3 "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0"
TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 4.4.4.4 "Mozilla/5.0 (Android 4.4.2; Tablet; rv:65.0) Gecko/65.0 Firefox/65.0"
TLSv1 AES128-SHA 5.5.5.5 "Mozilla/5.0 (Android 4.4.2; Tablet; rv:65.0) Gecko/65.0 Firefox/65.0"
```

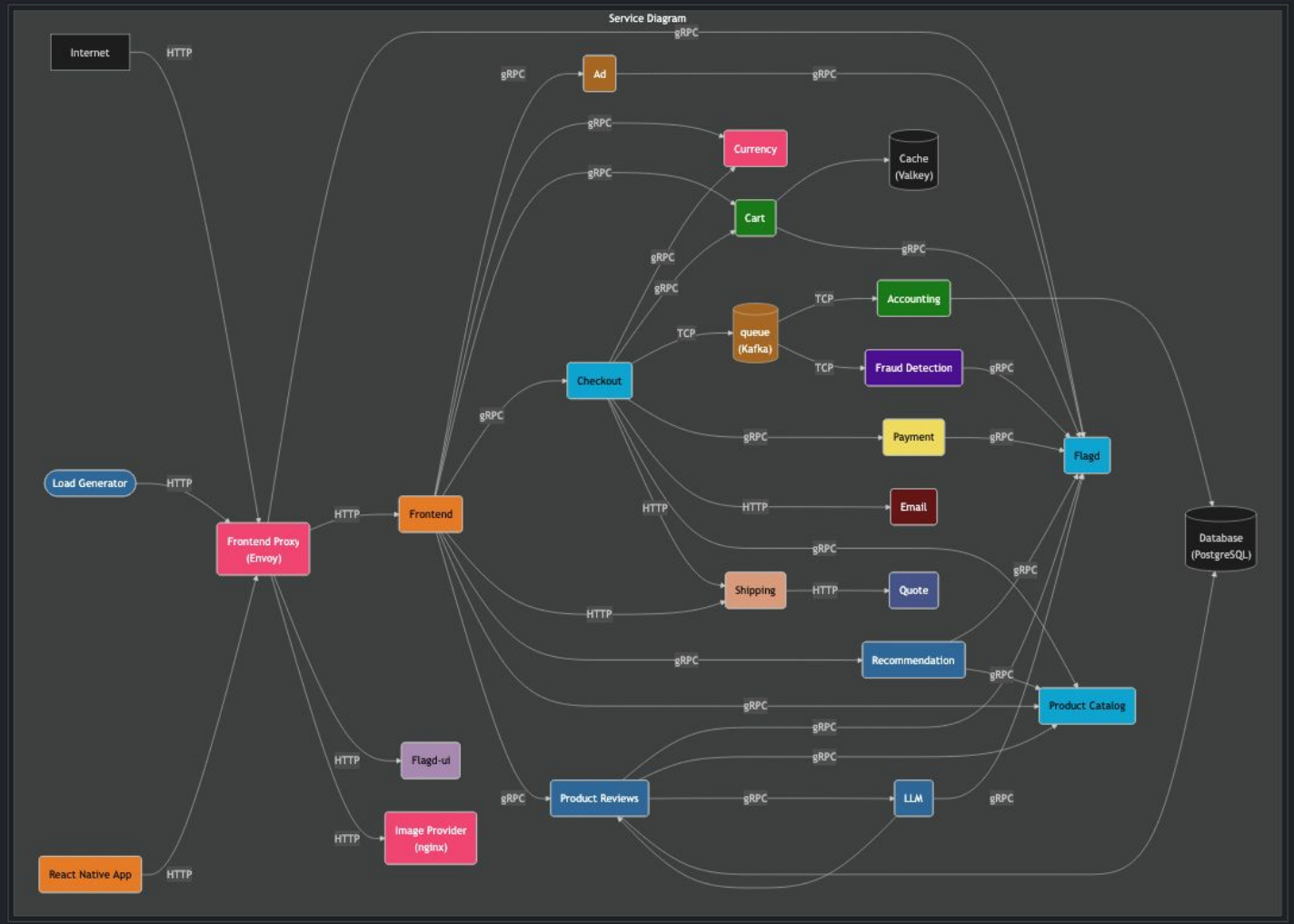
Logs by OTEL

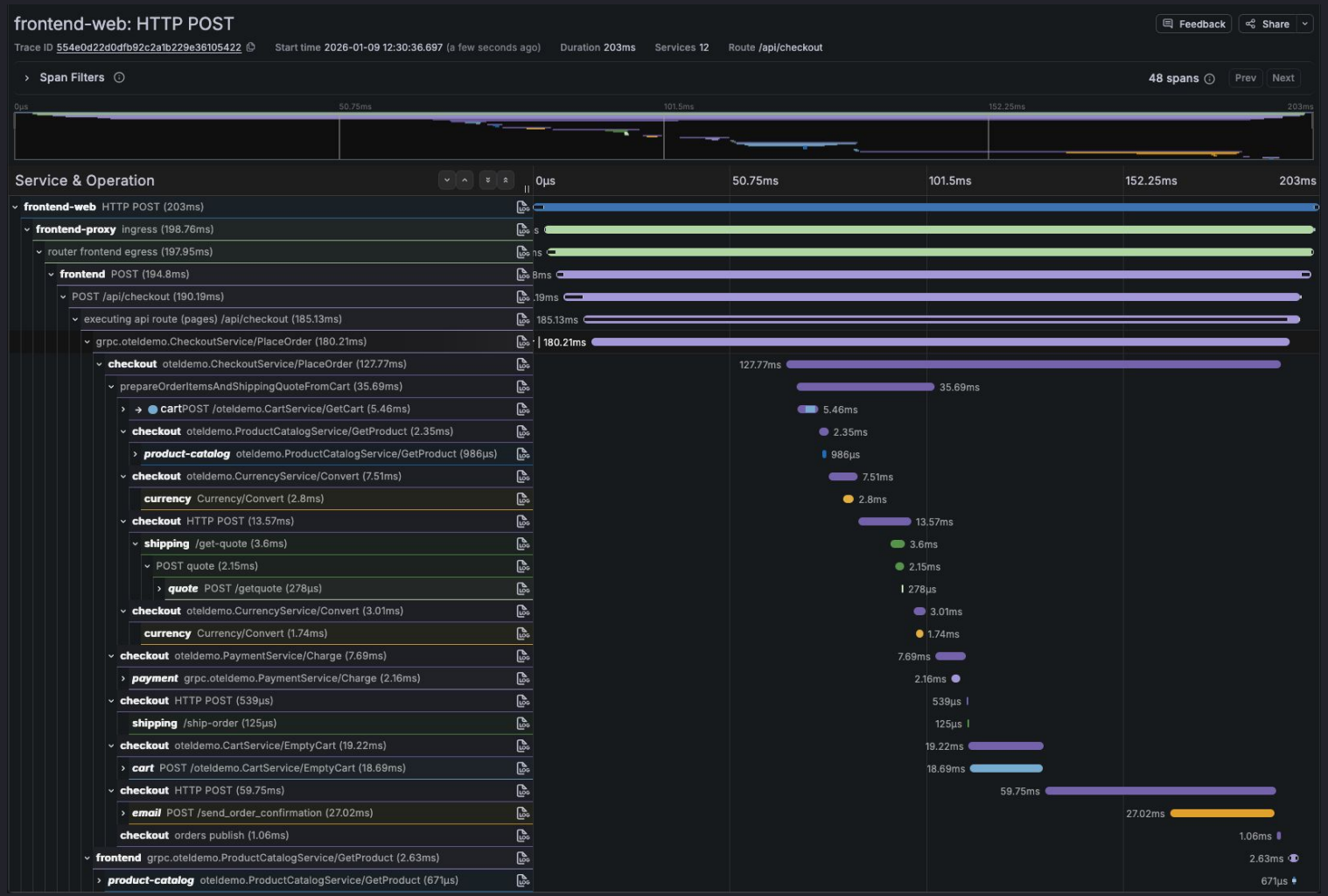
- Timestamp: Time when the event occurred.
- SeverityText: The severity text (also known as log level).
- Body: The body of the log record.
- **Resource**: Describes the source of the log.
- **TraceId: Request trace ID.**
- …. (there’s 6 more fields)

Traces

- The path of a request through your application
- A trace is represented as a collection of “spans” where each span is a unit of work or operation
- Context Propagation helps govern how the necessary information flows through our systems to ensure we can properly associate all spans in to a single trace

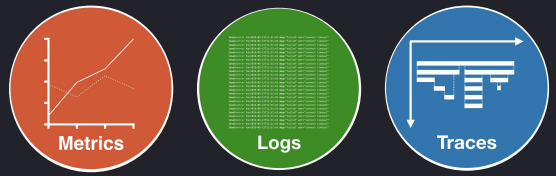

virtual-machine-1
client
calls
service
calls
database


Service Diagram
Internet
HTTP
Load Generator
HTTP
Frontend Proxy (Envoy)
HTTP
Frontend
gRPC
Ad
gRPC
Currency
gRPC
Cart
gRPC
Checkout
gRPC
queue (Kafka)
TCP
Accounting
TCP
Fraud Detection
gRPC
Payment
gRPC
Flagd
HTTP
Email
gRPC
Shipping
HTTP
Quote
gRPC
Recommendation
gRPC
Product Catalog
gRPC
LLM
gRPC
Product Reviews
gRPC
Cache (Valkey)
gRPC
Database (PostgreSQL)
HTTP
Flagd-ui
HTTP
Image Provider (nginx)
React Native App
HTTP

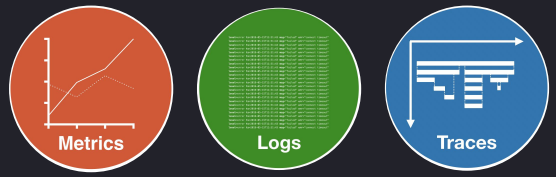

frontend-web: HTTP POST
Feedback
Share
Trace ID 554e0d22d0dfb92c2a1b229e36105422
Start time 2026-01-09 12:30:36.697 (a few seconds ago)
Duration 203ms
Services 12
Route /api/checkout
Span Filters
48 spans
Prev
Next
0µs
50.75ms
101.5ms
152.25ms
203ms
Service & Operation
frontend-web HTTP POST (203ms)
frontend-proxy ingress (198.76ms)
router frontend egress (197.95ms)
frontend POST (194.8ms)
POST /api/checkout (190.19ms)
executing api route (pages) /api/checkout (185.13ms)
grpc.oteldemo.CheckoutService/PlaceOrder (180.21ms)
checkout oteldemo.CheckoutService/PlaceOrder (127.77ms)
prepareOrderItemsAndShippingQuoteFromCart (35.69ms)
cart POST /oteldemo.CartService/GetCart (5.46ms)
checkout oteldemo.ProductCatalogService/GetProduct (2.35ms)
product-catalog oteldemo.ProductCatalogService/GetProduct (986µs)
checkout oteldemo.CurrencyService/Convert (7.51ms)
currency Currency/Convert (2.8ms)
checkout HTTP POST (13.57ms)
shipping /get-quote (3.6ms)
POST quote (2.15ms)
quote POST /getquote (278µs)
checkout oteldemo.CurrencyService/Convert (3.01ms)
currency Currency/Convert (1.74ms)
checkout oteldemo.PaymentService/Charge (7.69ms)
payment grpc.oteldemo.PaymentService/Charge (2.16ms)
checkout HTTP POST (539µs)
shipping /ship-order (125µs)
checkout oteldemo.CartService/EmptyCart (19.22ms)
cart POST /oteldemo.CartService/EmptyCart (18.69ms)
checkout HTTP POST (59.75ms)
email POST /send_order_confirmation (27.02ms)
checkout orders publish (1.06ms)
frontend grpc.oteldemo.ProductCatalogService/GetProduct (2.63ms)
product-catalog oteldemo.ProductCatalogService/GetProduct (671µs)
180.21ms
127.77ms
35.69ms
5.46ms
2.35ms
986µs
7.51ms
2.8ms
13.57ms
3.6ms
2.15ms
278µs
3.01ms
1.74ms
7.69ms
2.16ms
539µs
125µs
19.22ms
18.69ms
59.75ms
27.02ms
1.06ms
2.63ms
671µs

How I use various signals

- Metrics: “big picture”
  - Monitoring
  - System troubleshooting
  - Usage tracking
- Logs: “fine grained”
  - Bug troubleshooting
  - Audit + compliance
- Traces: “what is going on here?”
  - Bug troubleshooting in a distributed system
  - Performance Analysis



Specifications



OpenTelemetry semantic conventions 1.38.0

The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

HTTP Server Metrics

Metric: `http.server.request.duration`

This metric is required.

Metric: `http.server.active_requests`

This metric is optional.

Metric: `http.server.request.body.size`

This metric is optional.

Metric: `http.server.response.body.size`

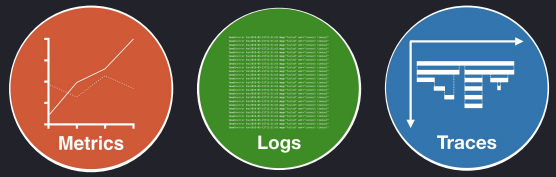
This metric is optional.

https://opentelemetry.io/docs/specs/semconv/http/http-metrics

Name	Instrument Type	Unit (UCUM)	Description	Stability	Entity Associations
`http.server.request.duration`	Histogram	`s`	Duration of HTTP server requests.	stable	

Attributes:

Key	Stability	Requirement Level	Value Type	Description	Example Values
`http.request.method`	stable	`Required`	string	HTTP request method. [1]	`GET` ; `POST` ; `HEAD`



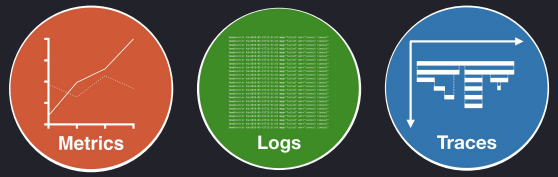
OpenTelemetry semantic conventions 1.38.0

The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications



OpenTelemetry semantic conventions 1.38.0

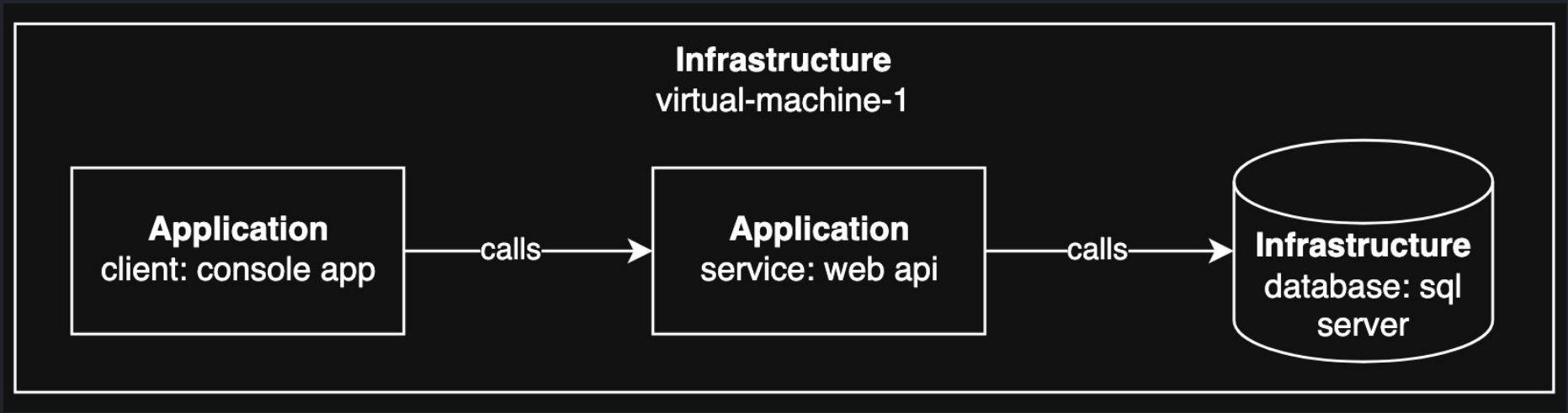
The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

Instrumentation


Infrastructure
virtual-machine-1
Application
client: console app
calls
Application
service: web api
calls
Infrastructure
database: sql
server

Application SDKs

Language	Traces	Metrics	Logs
C++	Stable	Stable	Stable
C#/.NET	Stable	Stable	Stable
Erlang/Elixir	Stable	Development	Development
Go	Stable	Stable	Beta
Java	Stable	Stable	Stable
JavaScript	Stable	Stable	Development
PHP	Stable	Stable	Stable
Python	Stable	Stable	Development
Ruby	Stable	Development	Development
Rust	Beta	Beta	Beta
Swift	Stable	Development	Development

https://opentelemetry.io/docs/languages/

dotnet: Metric

```
using System.Diagnostics.Metrics;

using var meter = new Meter("Examples.Service", "1.0");
var successCounter = meter.CreateCounter<long>("srv.successes.count", description: "Number of successful responses");

async Task<string> Handler()
{
    // .NET Diagnostics: update the metric
    successCounter.Add(1);

    return "Hello there";
}
```

dotnet: Logs

```
async Task<string> Handler(ILogger<Program> logger)
{
    // .NET ILogger: create a log
    logger.LogInformation("Success! Today is: {Date:MMMM dd, yyyy}", DateTimeOffset.UtcNow);

    return "Hello there";
}
```

dotnet: Trace

```
using System.Diagnostics;

// .NET Diagnostics: create the span factory
using var activitySource = new ActivitySource("Examples.Service");

async Task<string> Handler(ILogger<Program> logger)
{
    // .NET Diagnostics: create a manual span
    using (var activity = activitySource.StartActivity("SayHello"))
    {
        activity?.SetTag("foo", 1);
        activity?.SetTag("bar", "Hello, World!");
        activity?.SetTag("baz", new int[] { 1, 2, 3 });
        activity?.SetStatus(ActivityStatusCode.Ok);
    }
    return "Hello there";
}
```


YOU ARE DOING TOO MUCH
DO LESS
quickmeme.com

Grafana Labs

Auto (zero-code) Instrumentation

Supported in

- .NET
- Java
- Javascript
- PHP
- Python
- Go - WIP

https://opentelemetry.io/docs/zero-code/

dotnet: OOTB Metrics

ID	Instrumented library	Documentation
`ASPNET`	ASP.NET Framework [1] **Not supported on .NET**	ASP.NET metrics
`ASPNETCORE`	ASP.NET Core **Not supported on .NET Framework**	ASP.NET Core metrics
`HTTPCLIENT`	System.Net.Http.HttpClient and System.Net.HttpWebRequest	HttpClient metrics
`NETRUNTIME`	OpenTelemetry.Instrumentation.Runtime	Runtime metrics
`NPGSQL`	Npgsql **Not supported on .NET Framework**	Npgsql metrics
`NSERVICEBUS`	NServiceBus	NServiceBus metrics
`PROCESS`	OpenTelemetry.Instrumentation.Process	Process metrics
`SQLCLIENT`	Microsoft.Data.SqlClient, System.Data.SqlClient [2] and `System.Data` (shipped with .NET Framework)	SqlClient metrics

dotnet: OOTB Trace Support

ID	Instrumented library	Supported versions
ASPNET	ASP.NET (.NET Framework) MVC / WebApi [1] **Not supported on .NET**	* [2]
ASPNETCORE	ASP.NET Core **Not supported on .NET Framework**	*
AZURE	Azure SDK	[3]
ELASTICSEARCH	Elastic.Clients.Elasticsearch	* [4]
ELASTICTRANSPORT	Elastic.Transport	≥0.4.16
ENTITYFRAMEWORKCORE	Microsoft.EntityFrameworkCore **Not supported on .NET Framework**	≥6.0.12
GRAPHQL	GraphQL **Not supported on .NET Framework**	≥7.5.0
GRPCNETCLIENT	Grpc.Net.Client	≥2.52.0 & < 3.0.0
HTTPCLIENT	System.Net.Http.HttpClient and System.Net.HttpWebRequest	*
KAFKA	Confluent.Kafka	≥1.4.0 & < 3.0.0 [5]
MASSTRANSIT	MassTransit **Not supported on .NET Framework**	≥8.0.0
MONGODB	MongoDB.Driver.Core	≥2.13.3 & < 3.0.0
MYSQLCONNECTOR	MySqlConnector	≥2.0.0
MYSQLDATA	MySql.Data **Not supported on .NET Framework**	≥8.1.0
NPGSQL	Npgsql	≥6.0.0
NSERVICEBUS	NServiceBus	≥8.0.0 & < 10.0.0
ORACLEMDA	Oracle.ManagedDataAccess.Core and Oracle.ManagedDataAccess **Not supported on ARM64**	≥23.4.0
QUARTZ	Quartz **Not supported on .NET Framework 4.7.1 and older**	≥3.4.0
SQLCLIENT	Microsoft.Data.SqlClient, System.Data.SqlClient and `System.Data` (shipped with .NET Framework)	* [6]
STACKEXCHANGEREDIS	StackExchange.Redis **Not supported on .NET Framework**	≥2.0.405 & < 3.0.0
WCFCLIENT	WCF	*
WCFSERVICE	WCF **Not supported on .NET.**	*

dotnet: OOTB Log Support

Status: Experimental.

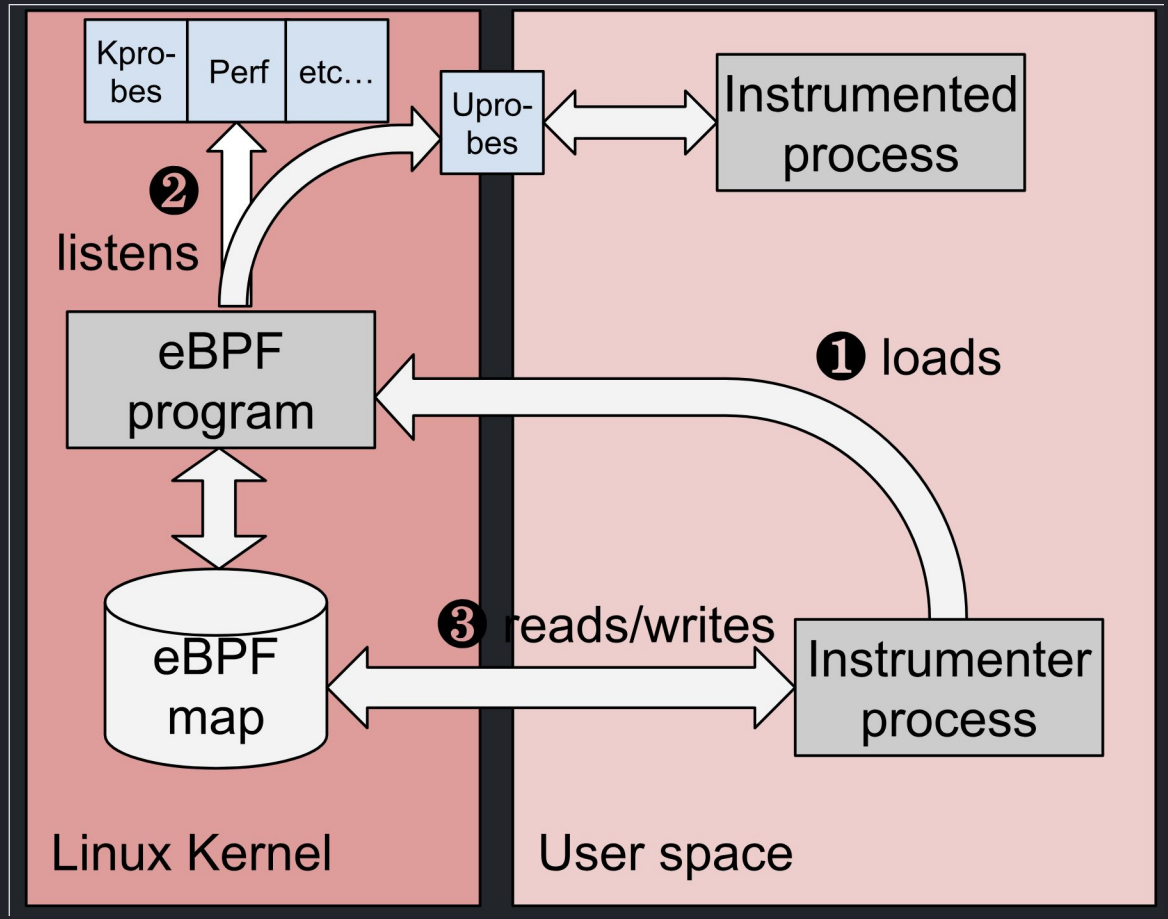
ID	Instrumented library	Supported versions
`ILOGGER`	Microsoft.Extensions.Logging **Not supported on .NET Framework**	≥8.0.0
`LOG4NET`	log4net	≥2.0.13 && < 4.0.0


YOU ARE DOING TOO MUCH
DO LESS
quickmeme.com

Grafana Labs

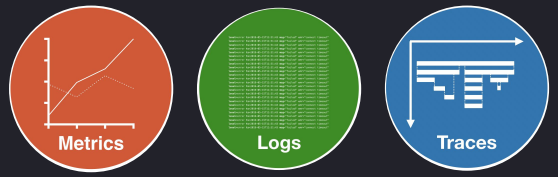
OpenTelemetry eBPF Instrumentation (OBI)

https://opentelemetry.io/docs/zero-code/obi/


Kpro-
bes
Perf
etc…
Upro-
bes
Instrumented
process
❷
listens
eBPF
program
❶ loads
eBPF
map
❸ reads/writes
Instrumenter
process
Linux Kernel
User space

OBI

- Originally Grafana Bayla before being donated
- Can capture **high level** application metrics
  - IE: Not a full replacement for language / manual instrumentation
- Languages Supported: Java, .NET, Go, Python, Ruby, Node.js, C, C++, and Rust
- Other features
  - Networking metrics
  - Process metrics
  - Trace capture



OpenTelemetry semantic conventions 1.38.0

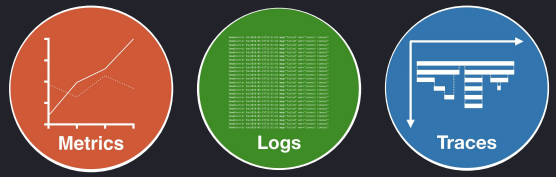
The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

Instrumentation



OpenTelemetry semantic conventions 1.38.0

The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

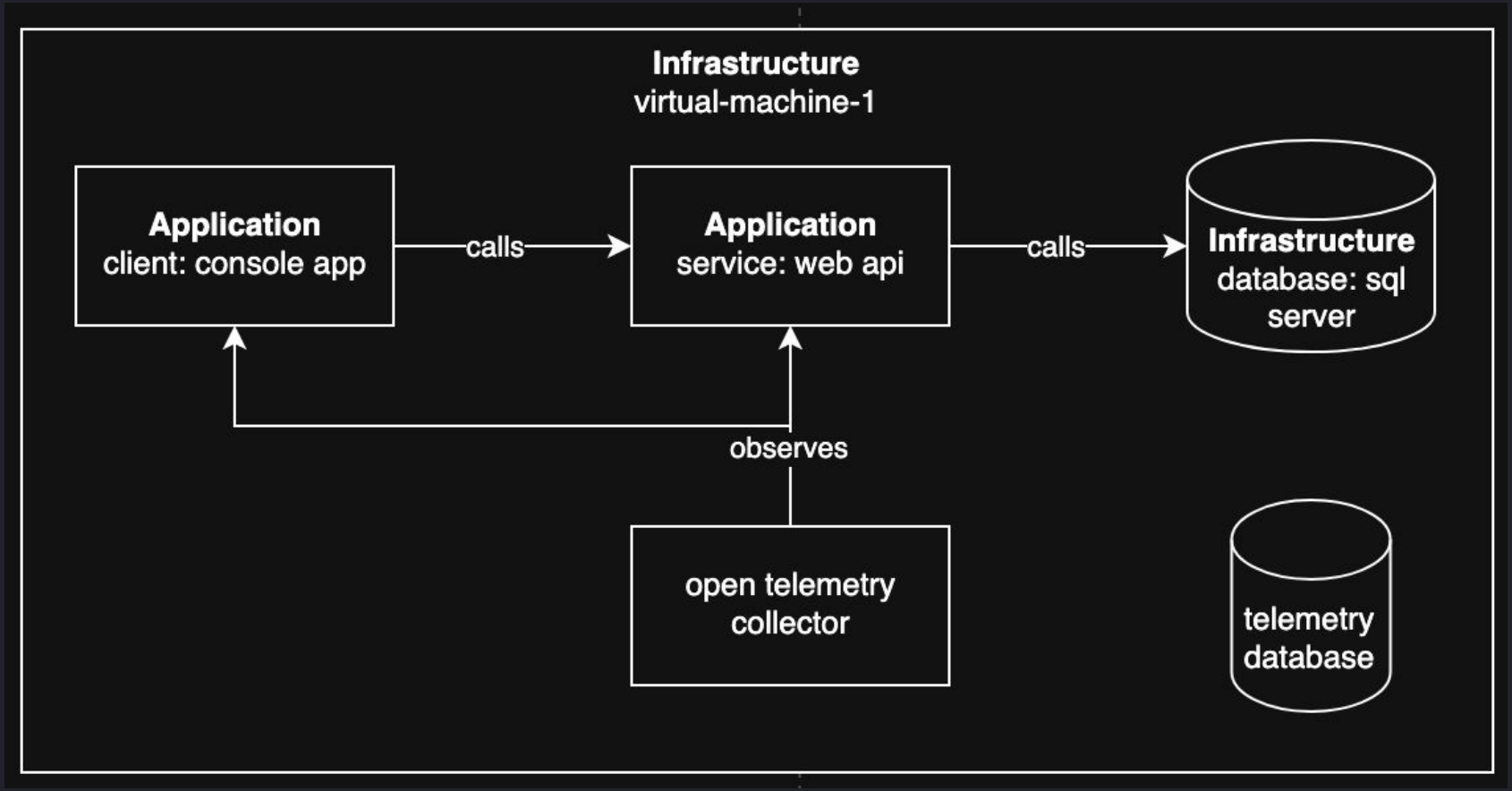
Semantic Conventions are defined for the following areas:

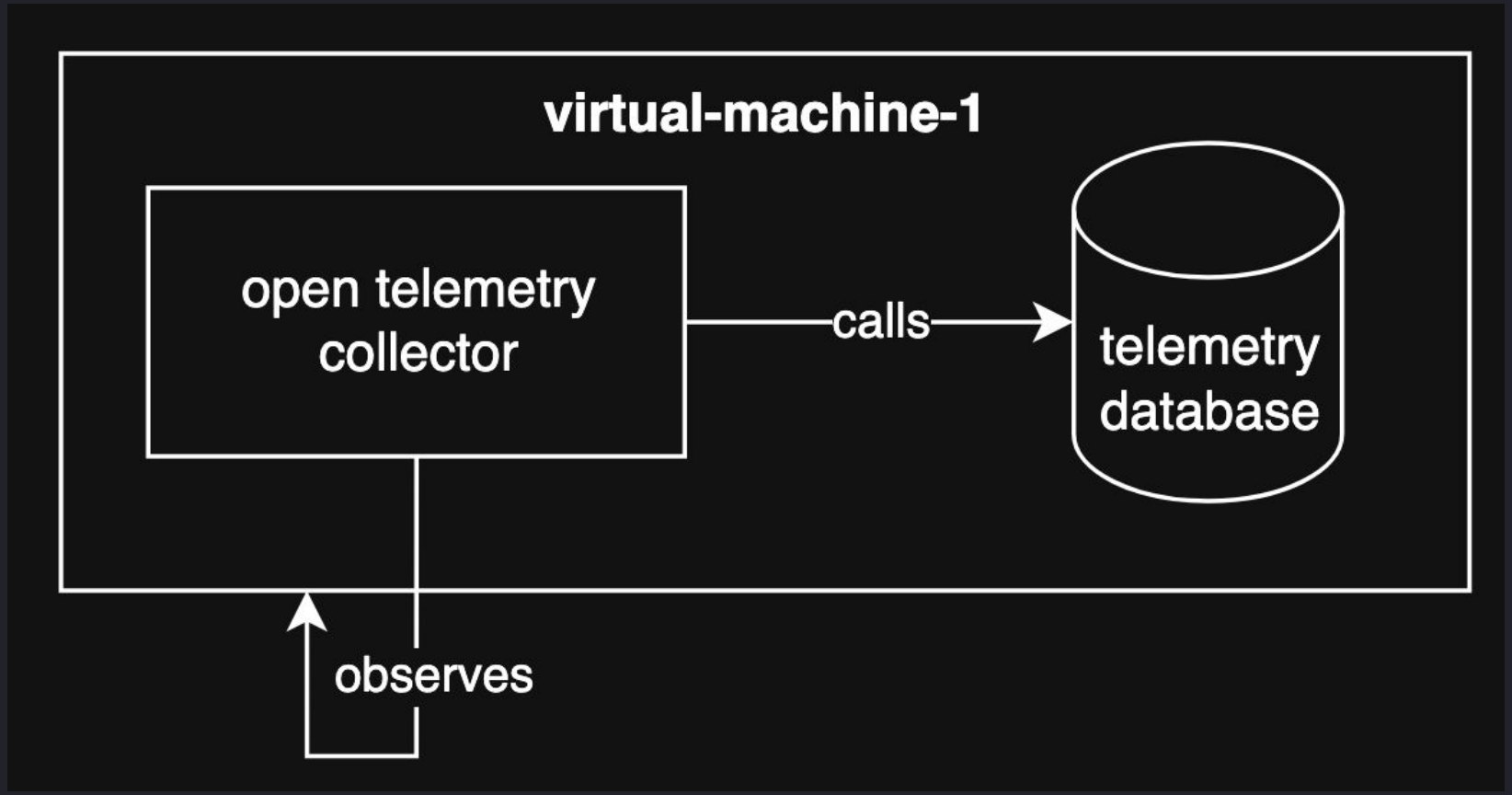
- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

Instrumentation

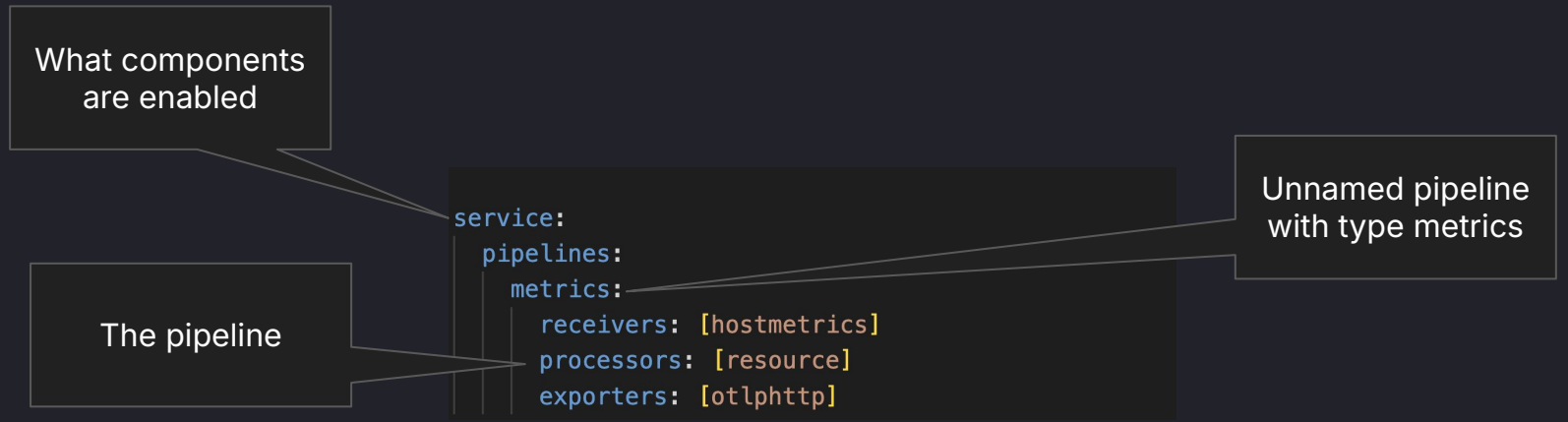
Collecting and Exporting


Infrastructure
virtual-machine-1
Application
client: console app
calls
Application
service: web api
calls
Infrastructure
database: sql
server
observes
open telemetry
collector
telemetry
database


virtual-machine-1
open telemetry collector
calls
telemetry database
observes


Otel Collector
OTLP
Jaeger
Prometheus
Receivers
Extensions: health, pprof, zpages
Batch
...
Attributes
Processors
Batch
...
Filter
Processors
Exporters
OTLP
Jaeger
Prometheus

```yaml
receivers:
  hostmetrics:
    collection_interval: 15s
    scrapers:
      cpu:
      memory:
      disk:
      network:

processors:
  resource:
    attributes:
      - key: service.name
        value: demo-host
        action: upsert

exporters:
  otlphttp:
    endpoint: http://localhost:9090/api/v1/otlp
    tls:
      insecure: true

service:
  pipelines:
    metrics:
      receivers: [hostmetrics]
      processors: [resource]
      exporters: [otlphttp]
```


What components are enabled
Unnamed pipeline with type metrics
The pipeline
service:
pipelines:
metrics:
receivers: [hostmetrics]
processors: [resource]
exporters: [otlphttp]

Receiver components available for pipelines

```yaml
receivers:
  hostmetrics:
    collection_interval: 15s
    scrapers:
      cpu:
      memory:
      disk:
      network:
```

Defines and configures the hostmetrics receiver

Enables the hostmetrics receiver

```yaml
service:
  pipelines:
    metrics:
      receivers: [hostmetrics]
      processors: [resource]
      exporters: [otlphttp]
```

```yaml
receivers:
  hostmetrics:
    collection_interval: 15s
    scrapers:
      cpu:
      memory:
      disk:
      network:

processors:
  resource:
    attributes:
      - key: service.name
        value: demo-host
        action: upsert
```

Processor components available for pipelines

Defines and configures the resource processor

```yaml
service:
  pipelines:
    metrics:
      receivers: [hostmetrics]
      processors: [resource]
      exporters: [otlphttp]
```

Enables the resource processor

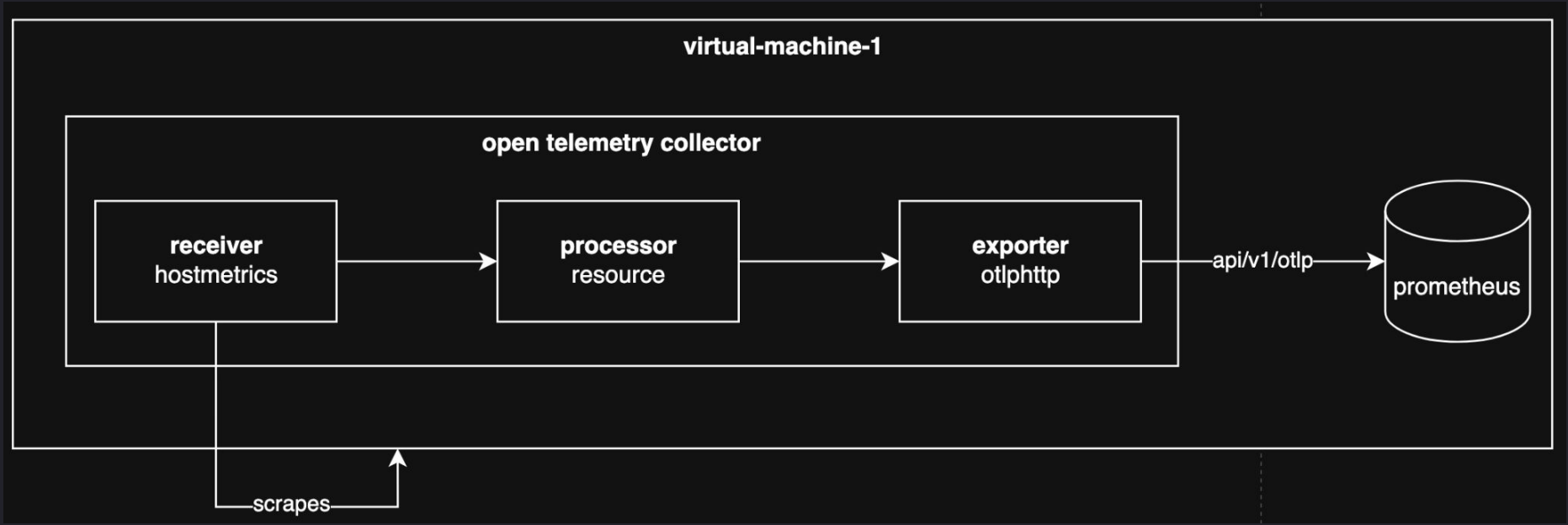
```yaml
receivers:
  hostmetrics:
    collection_interval: 15s
    scrapers:
      cpu:
      memory:
      disk:
      network:

processors:
  resource:
    attributes:
      - key: service.name
        value: demo-host
        action: upsert

exporters:
  otlphttp:
    endpoint: http://localhost:9090/api/v1/otlp
    tls:
      insecure: true

service:
  pipelines:
    metrics:
      receivers: [hostmetrics]
      processors: [resource]
      exporters: [otlphttp]
```

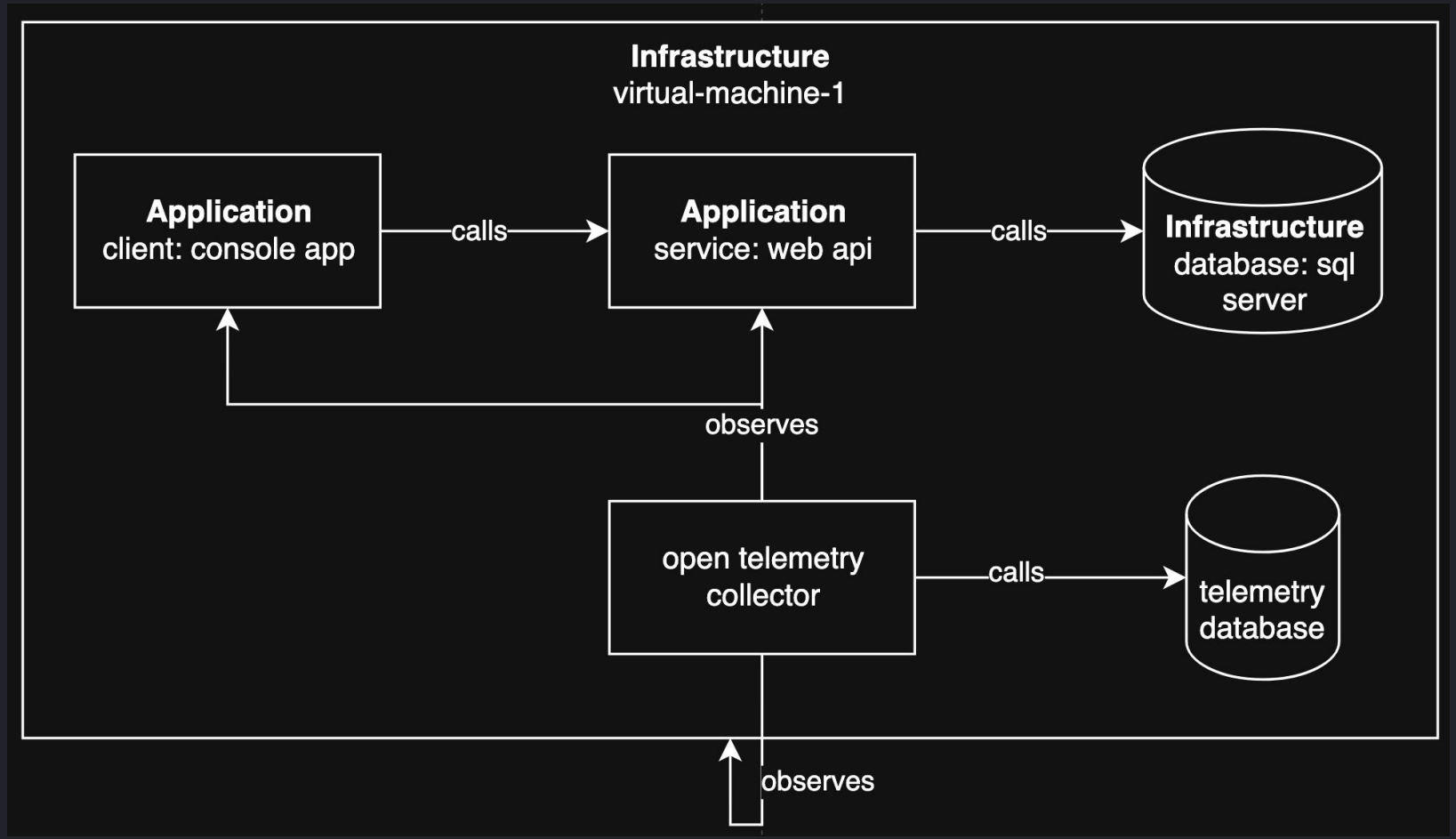
Exporter components available for pipelines

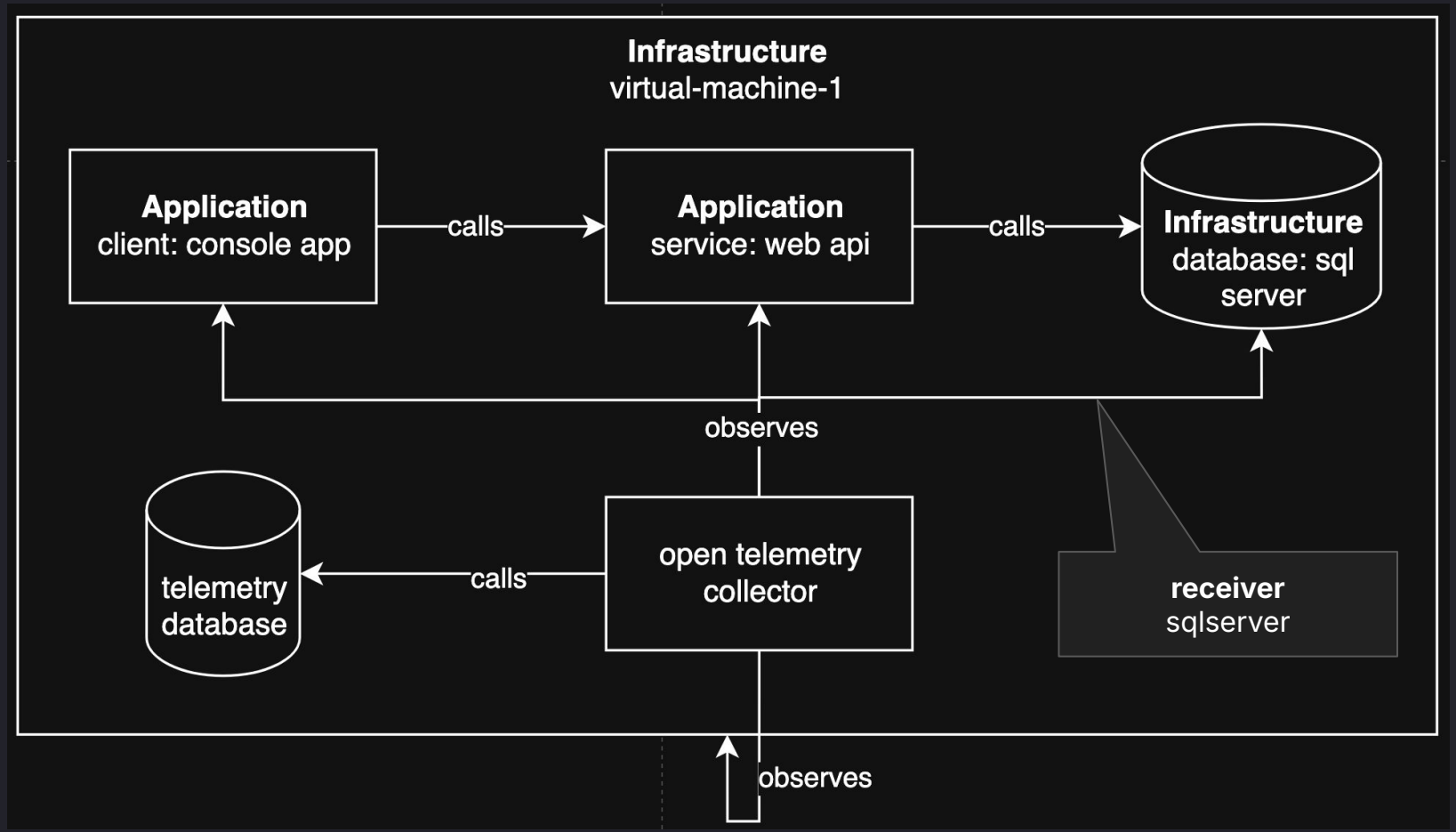
Defines and configures the otlphttp exporter

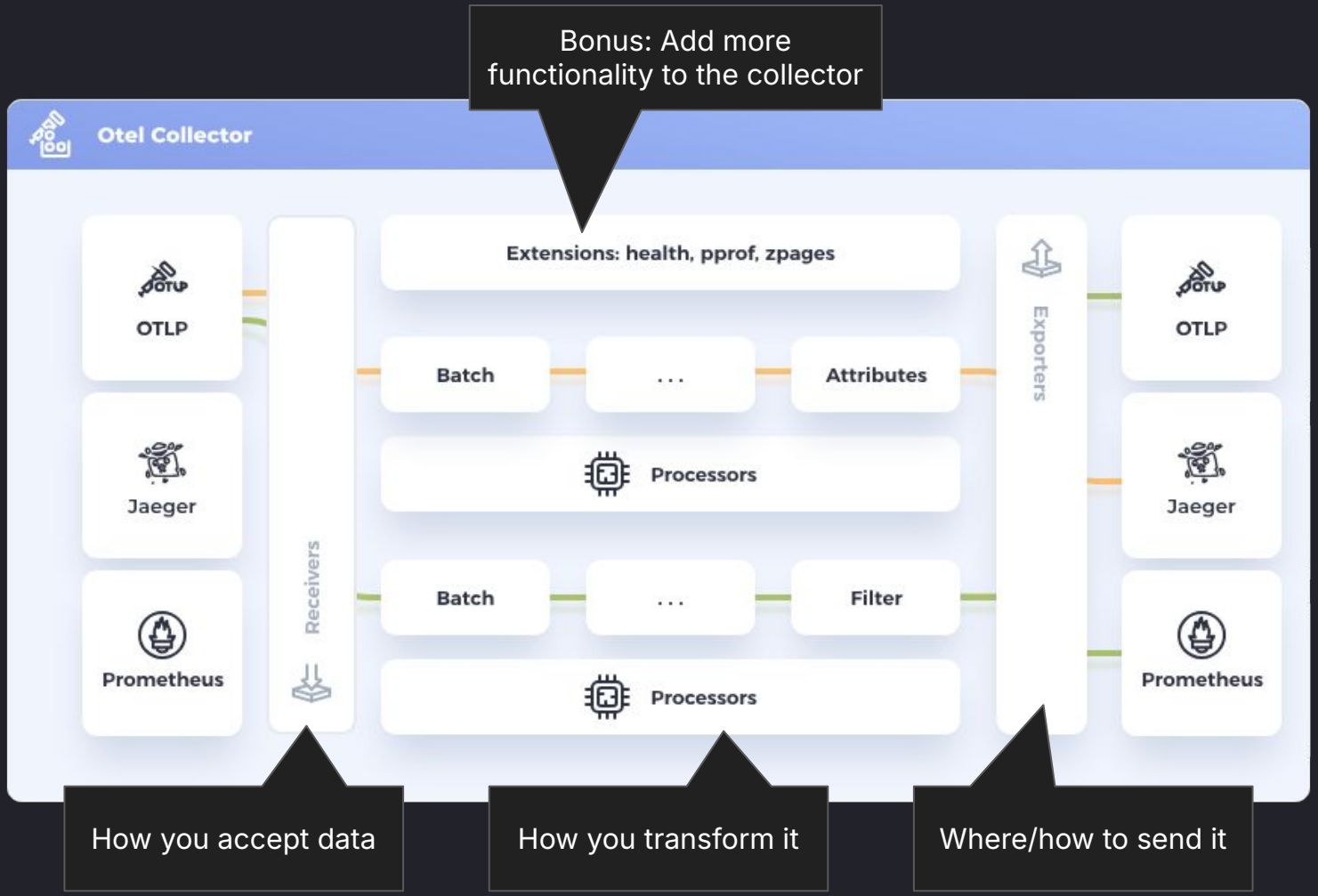
otlp = Open Telemetry Line Protocol

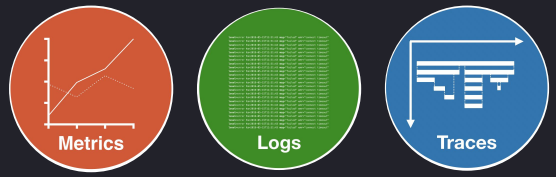
Enables the otlphttp exporter


virtual-machine-1
open telemetry collector
receiver
hostmetrics
processor
resource
exporter
otlphttp
api/v1/otlp
prometheus
scrapes


Infrastructure
virtual-machine-1
Application
client: console app
calls
Application
service: web api
calls
Infrastructure
database: sql
server
observes
open telemetry
collector
calls
telemetry
database
observes


Infrastructure
virtual-machine-1
Application
client: console app
calls
Application
service: web api
calls
Infrastructure
database: sql
server
observes
telemetry
database
calls
open telemetry
collector
receiver
sqlserver
observes


Bonus: Add more functionality to the collector
Otel Collector
OTLP
Jaeger
Prometheus
Receivers
Extensions: health, pprof, zpages
Batch
...
Attributes
Processors
Batch
...
Filter
Processors
Exporters
OTLP
Jaeger
Prometheus
How you accept data
How you transform it
Where/how to send it



OpenTelemetry semantic conventions 1.38.0

The Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it. The Semantic Conventions specify among other things span names and kind, metric instruments and units as well as attribute names, types, meaning and valid values. For a detailed definition of the Semantic Conventions' scope see Semantic Conventions Stability. The benefit to using Semantic Conventions is in following a common naming scheme that can be standardized across a codebase, libraries, and platforms. This allows easier correlation and consumption of data.

Semantic Conventions are defined for the following areas:

- General: **General Semantic Conventions**.
- CICD: Semantic Conventions for CICD systems.
- Cloud Providers: Semantic Conventions for cloud providers libraries.
- CloudEvents: Semantic Conventions for the CloudEvents specification.

Specifications

Instrumentation

Collecting and Exporting

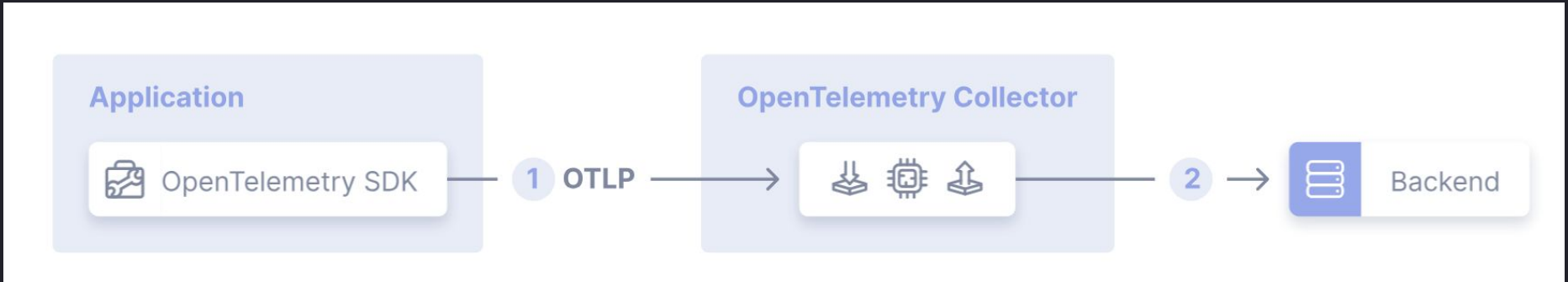
Grafana Labs

The end.....?

Grafana Labs

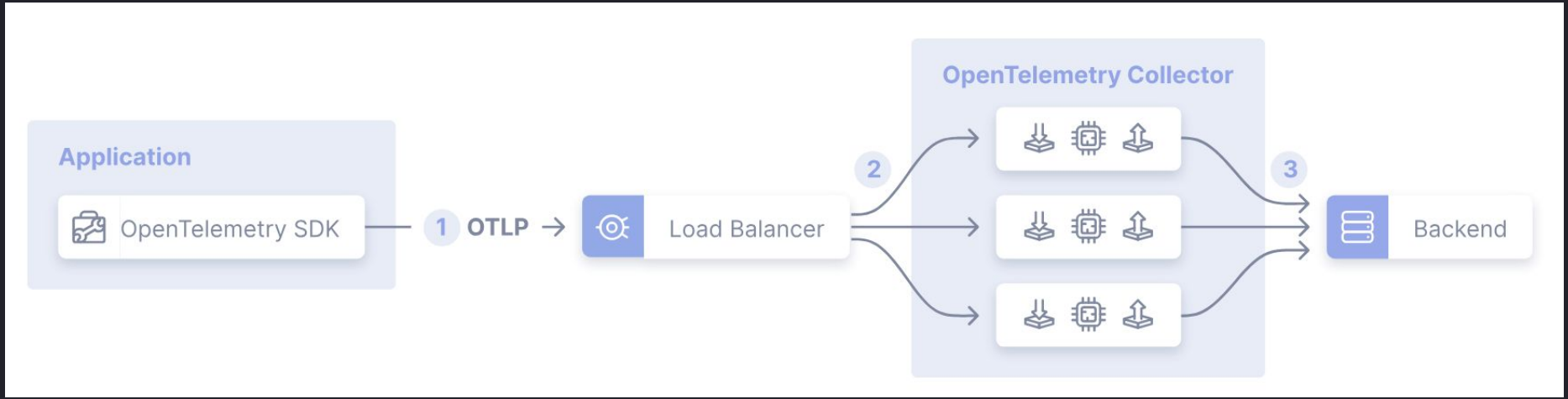
Deployment Patterns

“Agent”



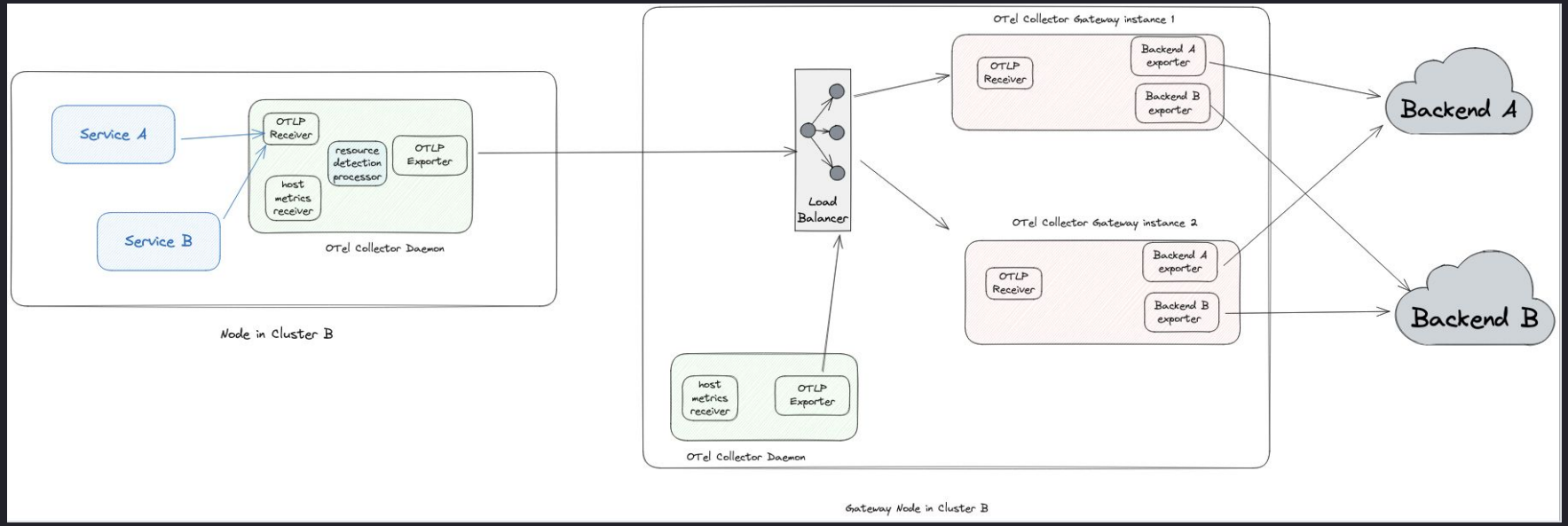
https://opentelemetry.io/docs/collector/deploy/agent/

“Gateway” Deployment



https://opentelemetry.io/docs/collector/deploy/gateway/

Agent + Gateway Deployment



https://opentelemetry.io/docs/collector/deploy/gateway/#combined-deployment-of-collectors-as-agents-and-gateways

Grafana Labs

What else?

Running on k8s?

- Open Telemetry Distribution for k8s:
- Consider using the open telemetry operator: https://opentelemetry.io/docs/platforms/kubernetes/operator/
- Integrate auto-instrumentation injection (annotation based): https://opentelemetry.io/docs/platforms/kubernetes/operator/automatic/
- Keep an eye on https://github.com/open-telemetry/opentelemetry-injector to remove the annotation requirement

Operational Considerations

- Enable batching - Exposed by all exporters via send_queue settings
- Use the memorylimiter processor
  - Puts hard a soft limits are memory preventing OOM crashes
  - Will refuse data when limits are breached
- Consider persisting telemetry
  - Exposed via filestorage extensions with send_queue
  - Useful for “gateway” or critical “agent” instances

Grafana Labs

Open Agent Management Protocol (OPAMP)

https://opentelemetry.io/docs/collector/management/

Grafana Labs

The end.....?

Intelligently using your Telemetry

Grafana Labs

Open Telemetry

Is a vendor-neutral open source Observability framework for **instrumenting, generating, collecting, and exporting telemetry data**

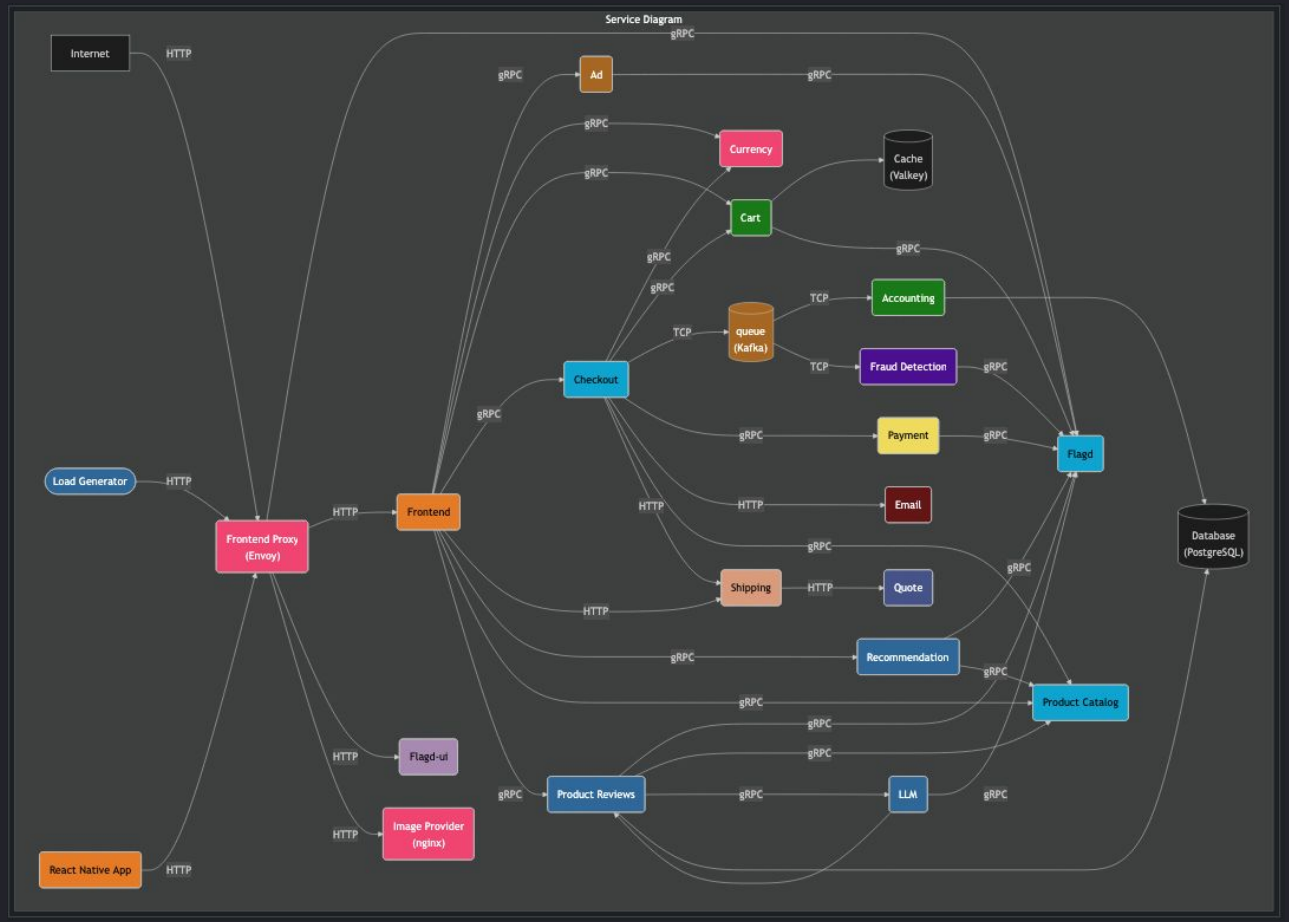

SAY THE LINE, SENIOR DEVELOPER
"IT DEPENDS"
imgflip.com

Understand system load in real time

Detect whether a recent deploy is causing instability

Catch bottlenecks and errors before customers report them

Map internal issues directly to customer impact

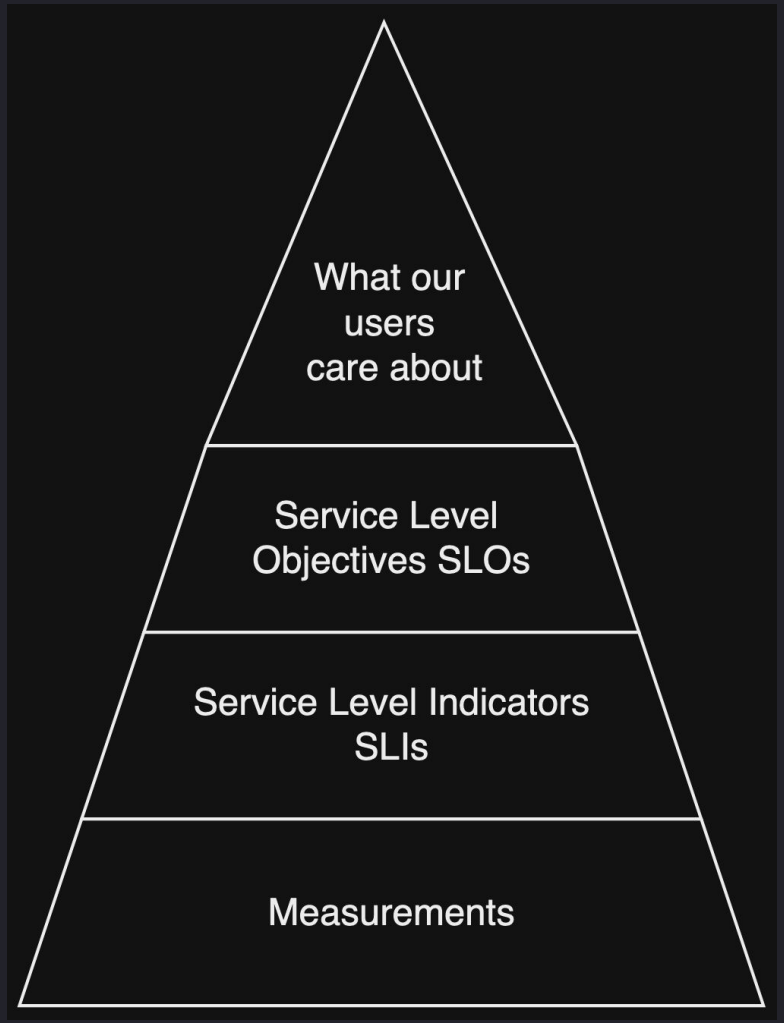


https://opentelemetry.io/docs/demo/architecture/

It’s impossible to manage a service correctly, let alone well, without understanding which behaviors really matter for that service

Chapter 4 - Service Level Objectives

Site Reliability Engineering
How Google Runs Production Systems


What our users care about
Service Level Objectives SLOs
Service Level Indicators SLIs
Measurements

The Four Golden Measurements

- Latency: How long does it take to process a request
- Traffic: How much demand is being placed on your system
  - HTTP services - number of HTTP requests per second by route
  - Database - operations per second by Query/Manipulation
- Errors: The rate of requests that fail
  - Unsuccessful response codes
  - Breaches of an agreed upon response time
- Saturation: How “full” your service is
  - CPU or Memory utilization being high
  - Disk drive filling up

Chapter 6 - Monitoring Distributed Systems

The Four Golden Measurements

- Latency: How long does it take to process a request → Histogram
- Traffic: How much demand is being placed on your system → Counter
- Errors: The rate of requests that fail → Counter
- Saturation: How “full” your service is - Gauge

Know your system

Measurement

- High latency
- Unexpected flood of traffic
- High error rate
- Disk nearing saturation

vs

User Impact

- High page load times
- Rate limiting errors / high load times?
- Pages failing to be rendered
- Potential data loss imminent

Example

“We will always accept new orders”

- SLO: The order creation API will succeed 99.999% of the time in under 15s
- SLIs
  - Latency
  - Errors
- Measurements:
  - Synthetic Monitoring
  - Checkout service http server

Now what?


Number of Alloy Instances
18
17.5
17
16.5
16
14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10
v1.12.1-devel+f8b1de8
CPU usage
5000%
4800%
4600%
4400%
4200%
4000%
3800%
3600%
14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10
Total
Memory (RSS)
310 GB
300 GB
290 GB
280 GB
270 GB
260 GB
14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10
Total
WAL delay
1 min
50 s
40 s
30 s
20 s
10 s
14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10
remote_write - max
remote_write - avg
WAL Data In
1.27M c/s
1.25M c/s
1.23M c/s
1.20M c/s
1.18M c/s
1.15M c/s
14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10
remote_write - Total
Samples In vs Out
400K c/s
200K c/s
0 c/s
-200K c/s
14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10
cortex-dev-01 - Total
ge-metrics - Total
mimir-dev-15 - Total

Observability and the core signals

Instrumenting and collecting Telemetry

Intelligently using your Telemetry

Grafana Labs

Session Feedback

OTel me more! A practical guide for adopting OpenTelemetry

Questions?

Linkedin