MySQL 8.3 Enterprise Edition & Open Telemetry

Safe Harbor

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

MySQL 8.4 Enterprise Edition Features

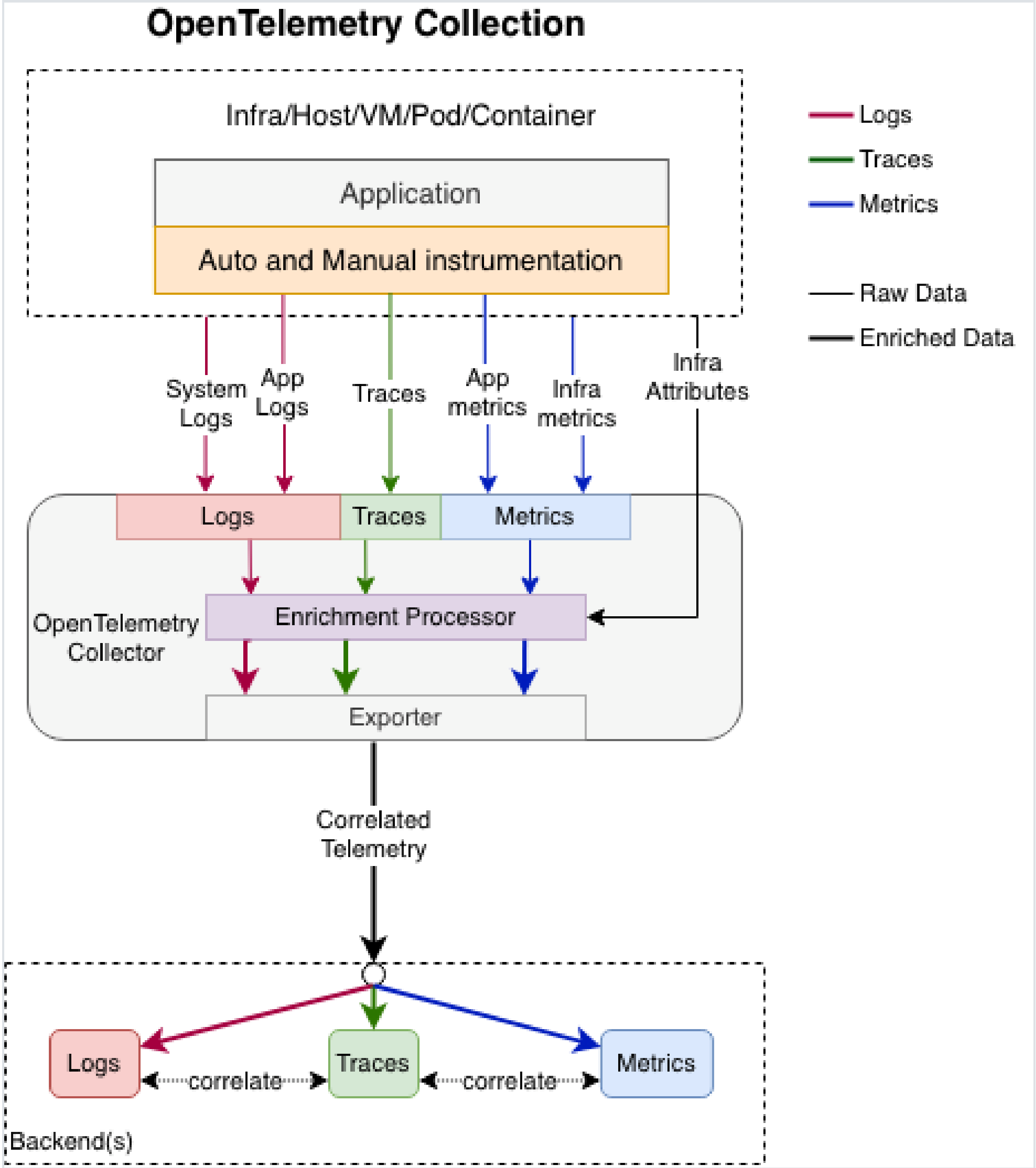
- EE Encryption
- EE Audit
- EE Firewall
- EE Thread Pool
- EE Data Masking
- EE LDAP authentication
- EE Kerberos authentication
- EE PAM authentication
- EE Windows authentication

- EE WebauthN authentication
- EE AWS keyring
- EE Encrypted File keyring
- EE Hashicorp keyring
- EE Oracle Key Vault keyring
- EE OCI Keyring
- EE JavaScript Stored Programs
- EE OpenTelemetry

Introduction to OpenTelemetry

https://opentelemetry.io/

The OpenTelemetry (OTel) project is an open-source, vendor-neutral observability framework, providing a common observability standard. It enables users to instrument their applications in order to export observability data: traces, metrics and logs, enabling increased granularity of debugging and testing.



OpenTelemetry Architecture

Signals flowing from Applications to Backend

Traces , **Metrics** , **Logs**

MySQL plays the role of **application** within in the OpenTelemetry observability framework

OpenTelemetry Infrastructure Components

- Optionally install and start the OpenTelemetry **Collector**

  Adds additional flexibility and options, such as sending signals to many backends

- Install and start vendor specific **Backend**

  Example: Jaeger, Prometheus, …

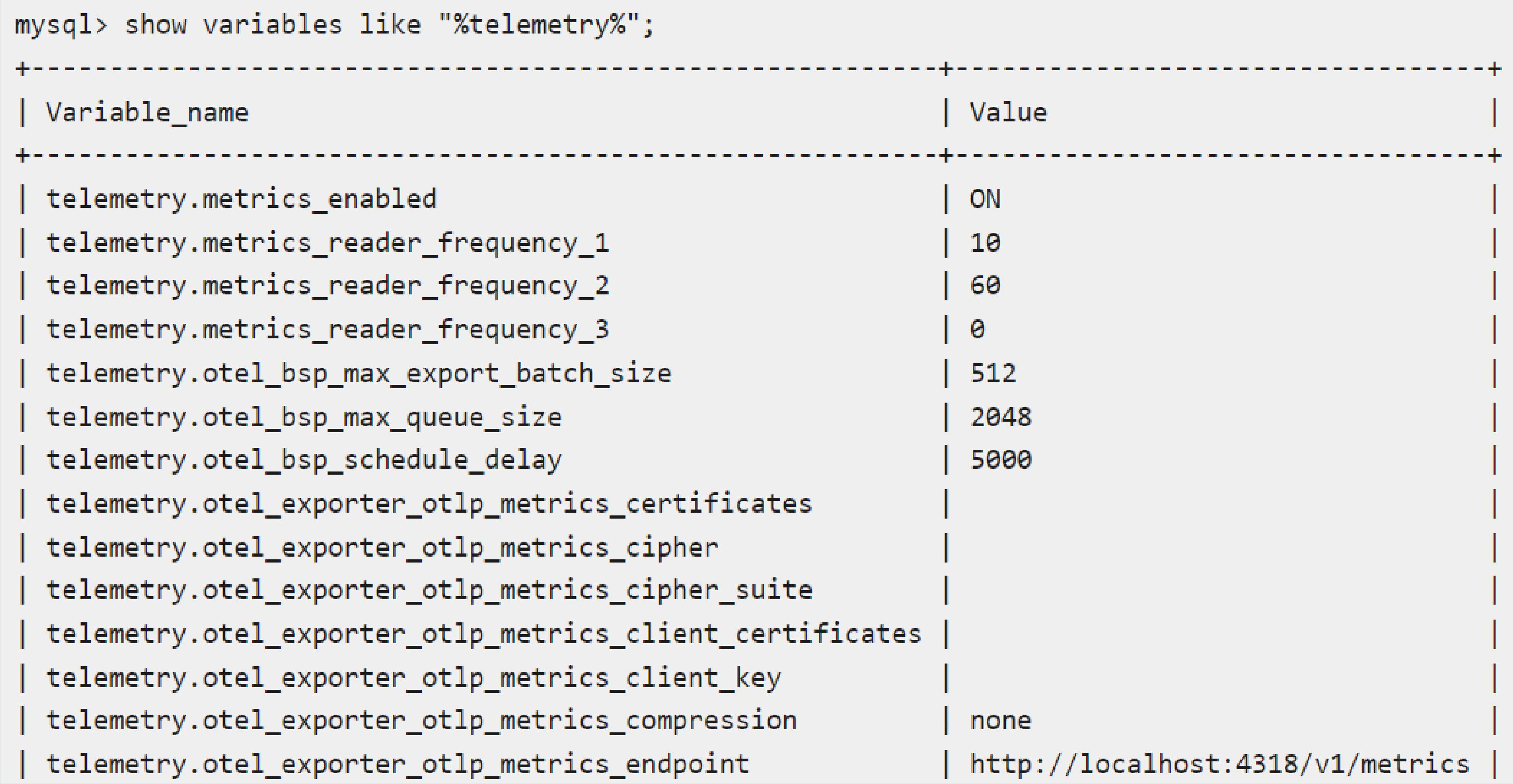
- Use vendor specific **Visualization** tools

  Example: Grafana is a popular choice

MySQL EE – Telemetry Introduction

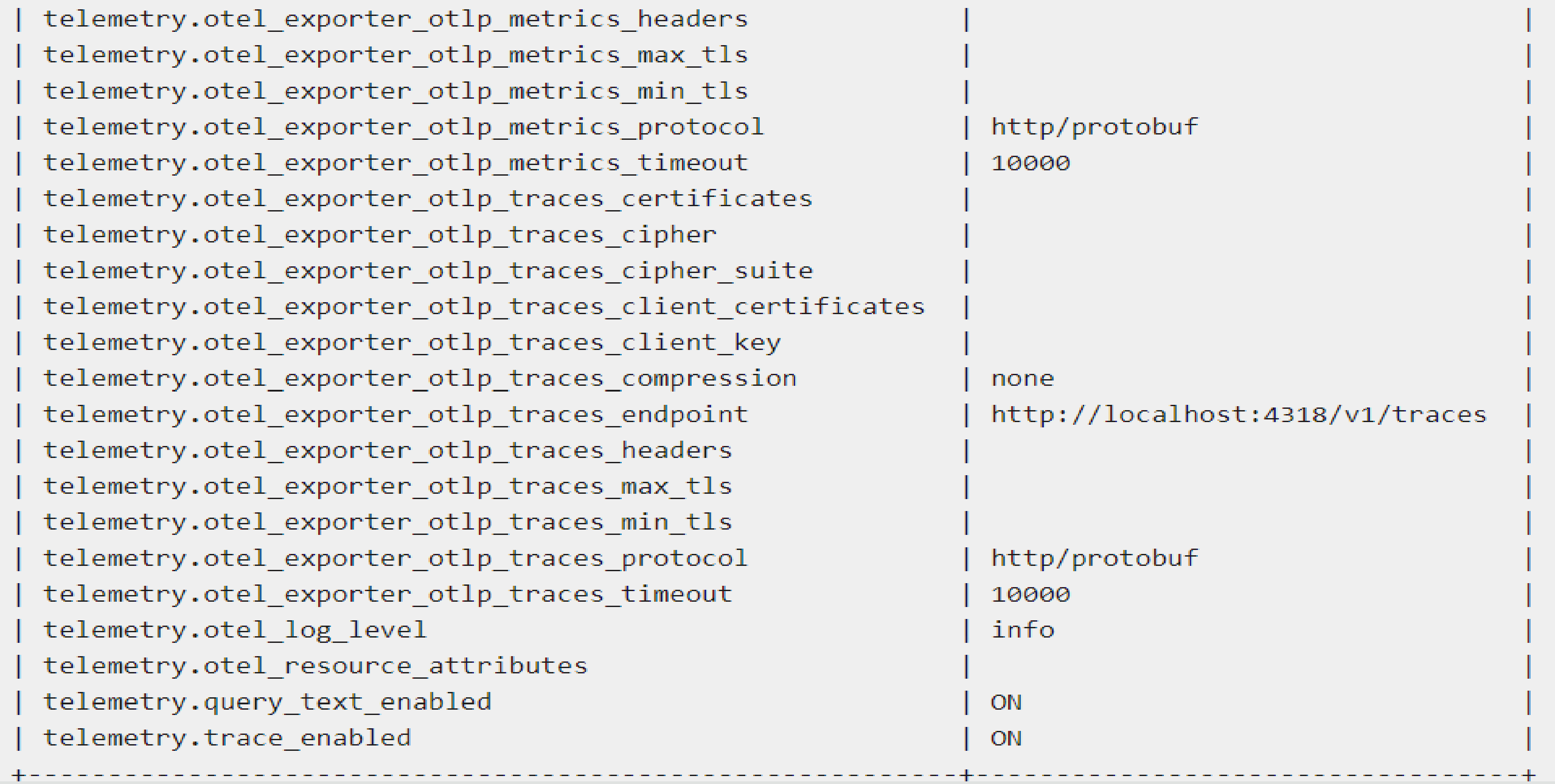
- Supporting Traces and Metrics in 8.3
- Supported on all MySQL EE platforms
- Supported by all MySQL Connectors
- Install component 'file://component_telemetry’;
- Enable/Disable Traces and Metrics
- Can configure different endpoints for Traces and Metrics

Telemetry-specific System variables



```
mysql> show variables like "%telemetry%";
+----------------------------------------------------------+----------------------------------+
| Variable_name                                            | Value                            |
+----------------------------------------------------------+----------------------------------+
| telemetry.metrics_enabled                                | ON                               |
| telemetry.metrics_reader_frequency_1                     | 10                               |
| telemetry.metrics_reader_frequency_2                     | 60                               |
| telemetry.metrics_reader_frequency_3                     | 0                                |
| telemetry.otel_bsp_max_export_batch_size                 | 512                              |
| telemetry.otel_bsp_max_queue_size                        | 2048                             |
| telemetry.otel_bsp_schedule_delay                        | 5000                             |
| telemetry.otel_exporter_otlp_metrics_certificates        |                                  |
| telemetry.otel_exporter_otlp_metrics_cipher              |                                  |
| telemetry.otel_exporter_otlp_metrics_cipher_suite        |                                  |
| telemetry.otel_exporter_otlp_metrics_client_certificates |                                  |
| telemetry.otel_exporter_otlp_metrics_client_key          |                                  |
| telemetry.otel_exporter_otlp_metrics_compression         | none                             |
| telemetry.otel_exporter_otlp_metrics_endpoint            | http://localhost:4318/v1/metrics |
```

Telemetry-specific System variables



```
| telemetry.otel_exporter_otlp_metrics_headers            |                                 |
| telemetry.otel_exporter_otlp_metrics_max_tls            |                                 |
| telemetry.otel_exporter_otlp_metrics_min_tls            |                                 |
| telemetry.otel_exporter_otlp_metrics_protocol           | http/protobuf                   |
| telemetry.otel_exporter_otlp_metrics_timeout            | 10000                           |
| telemetry.otel_exporter_otlp_traces_certificates        |                                 |
| telemetry.otel_exporter_otlp_traces_cipher              |                                 |
| telemetry.otel_exporter_otlp_traces_cipher_suite        |                                 |
| telemetry.otel_exporter_otlp_traces_client_certificates |                                 |
| telemetry.otel_exporter_otlp_traces_client_key          |                                 |
| telemetry.otel_exporter_otlp_traces_compression         | none                            |
| telemetry.otel_exporter_otlp_traces_endpoint            | http://localhost:4318/v1/traces |
| telemetry.otel_exporter_otlp_traces_headers             |                                 |
| telemetry.otel_exporter_otlp_traces_max_tls             |                                 |
| telemetry.otel_exporter_otlp_traces_min_tls             |                                 |
| telemetry.otel_exporter_otlp_traces_protocol            | http/protobuf                   |
| telemetry.otel_exporter_otlp_traces_timeout             | 10000                           |
| telemetry.otel_log_level                                | info                            |
| telemetry.otel_resource_attributes                      |                                 |
| telemetry.query_text_enabled                            | ON                              |
| telemetry.trace_enabled                                 | ON                              |
+---------------------------------------------------------+---------------------------------+
```

Traces

Traces

- Understand the full path through your distributed application
- By passing a **trace context** from one distributed component to the next
- And using this **trace context** as an identifier for reporting information about computing steps taken by individual components within the overall distributed execution path
- The trace context is given by the **trace_id** (the whole tree) and **span_id** (a given node in the tree)

Spans

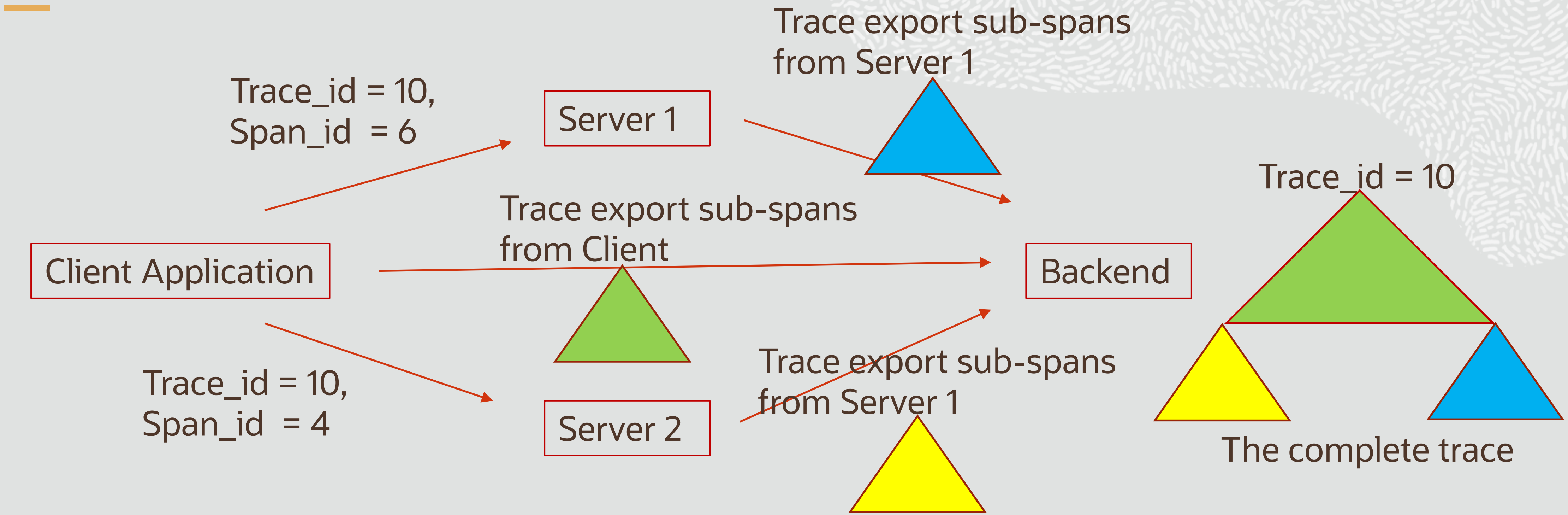
- A span represents a unit of work or operation
- Spans are the building blocks of Traces.
- A span include the information shown here =>

- Name
- Parent span ID (empty for root spans)
- Start and End Timestamps
- Span Context
- Attributes
- Span Events
- Span Links
- Span Status

Nested Spans

- Spans can be nested, as is implied by the presence of a parent span ID
- Child spans represent sub-operations. This allows spans to more accurately capture the work done in an application

Only the Backend sees the full picture



Trace export sub-spans from Server 1

Trace_id = 10,
Span_id = 6

Server 1

Trace_id = 10

Trace export sub-spans from Client

Client Application

Backend

Trace export sub-spans from Server 1

Trace_id = 10,
Span_id = 4

Server 2

The complete trace

MySQL Support for Traces

- MySQL **Connectors** are extended with the ability to pass on the **trace context** to the MySQL Server
- Trace context is passed on as **query attributes** over the MySQL **protocol**
- The MySQL **Server** understands the incoming trace context and reports its own traces in the same context and according to the OpenTelemetry specification
- When no context is provided then a new context is created

MySQL Span Type : Control

Issued when the telemetry configuration changes, notifying downstream system of which signal collection has been enabled or disabled

No parent span_id

Span attributes

- Name: Control
- trace_enabled: Boolean.
- metrics_enabled: Boolean.
- logs_enabled: Boolean
- details:

MySQL Span Type : Session

Issued when a **client session ends**, recording data relevant to that session from initial connection to close of session

No parent span_id

From performance_schema table

session_connect_attrs

Span attributes

- Name: Session
- mysql.processlist_id
- mysql.thread_id
- mysql.user
- mysql.host
- mysql.group
- + session attributes

MySQL Span Type : Statement

Issued when a **statement execution ends** in the server, recording all relevant statement information from the start of the execution to its completion

Parent span_id

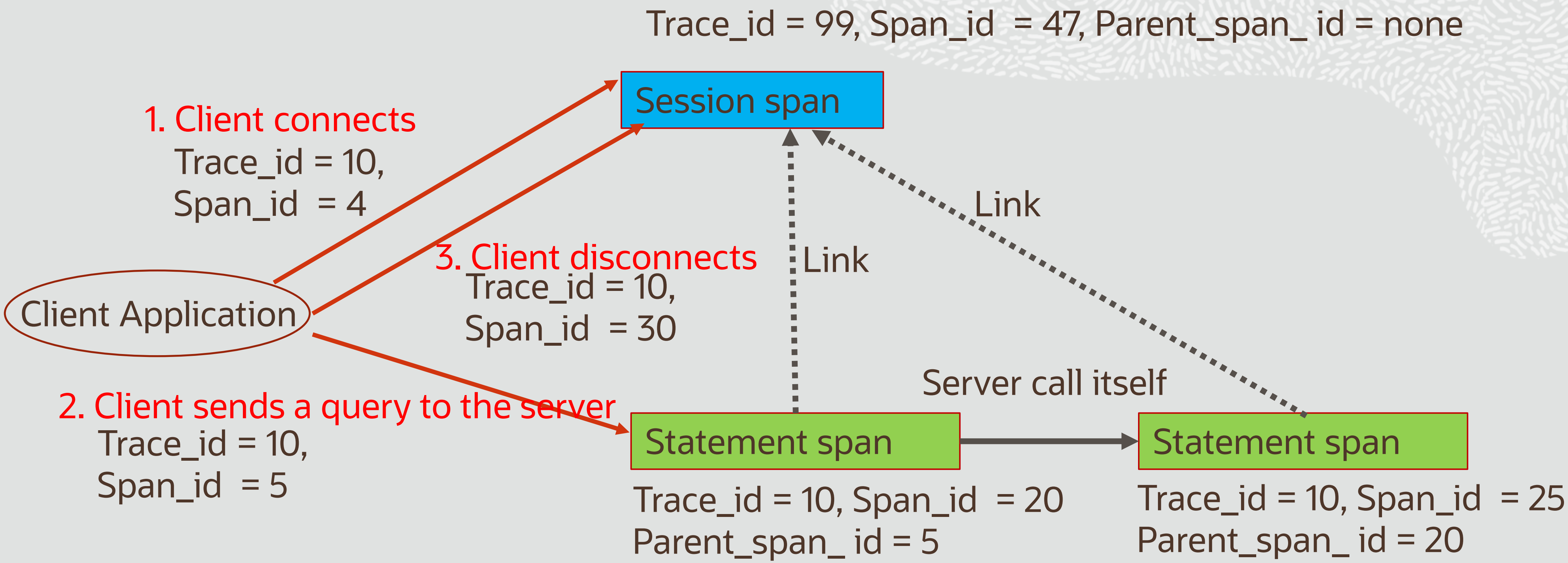
From performance_schema table

events_statements_current

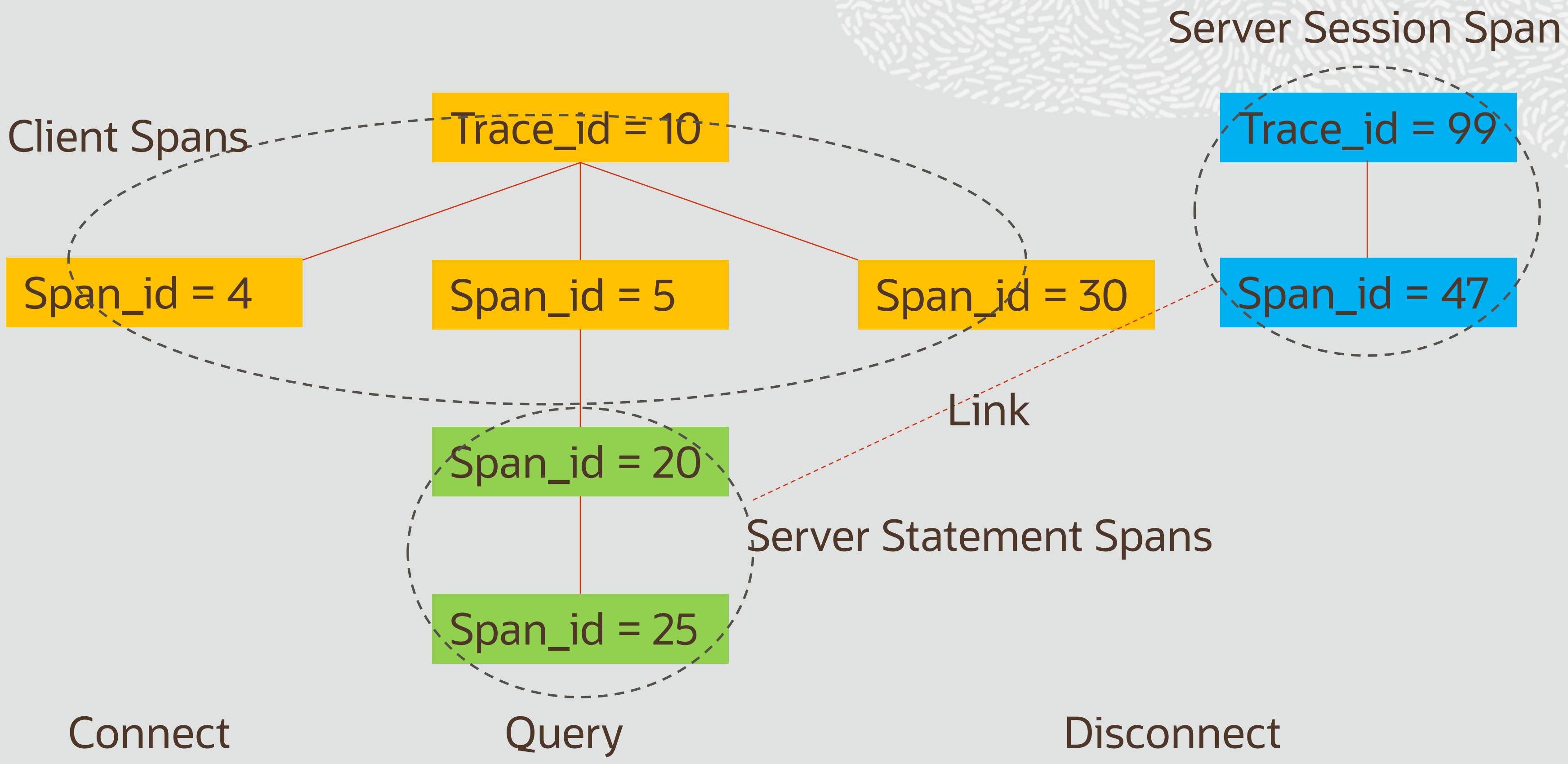
Span attributes

- Name: stmt
- mysql.event_name
- mysql.lock_time
- mysql.sql_text
- mysql.digest_text
- mysql.current_schema
- mysql.object_type
- mysql.object_schema
- mysql.object_name
- mysql.sql_errno
- mysql.sqlstate
- mysql.message_text
- + Many more ….

Example: Session and Statement Spans



Example: Resulting Trace



Metrics

OpenTelemetry Metrics

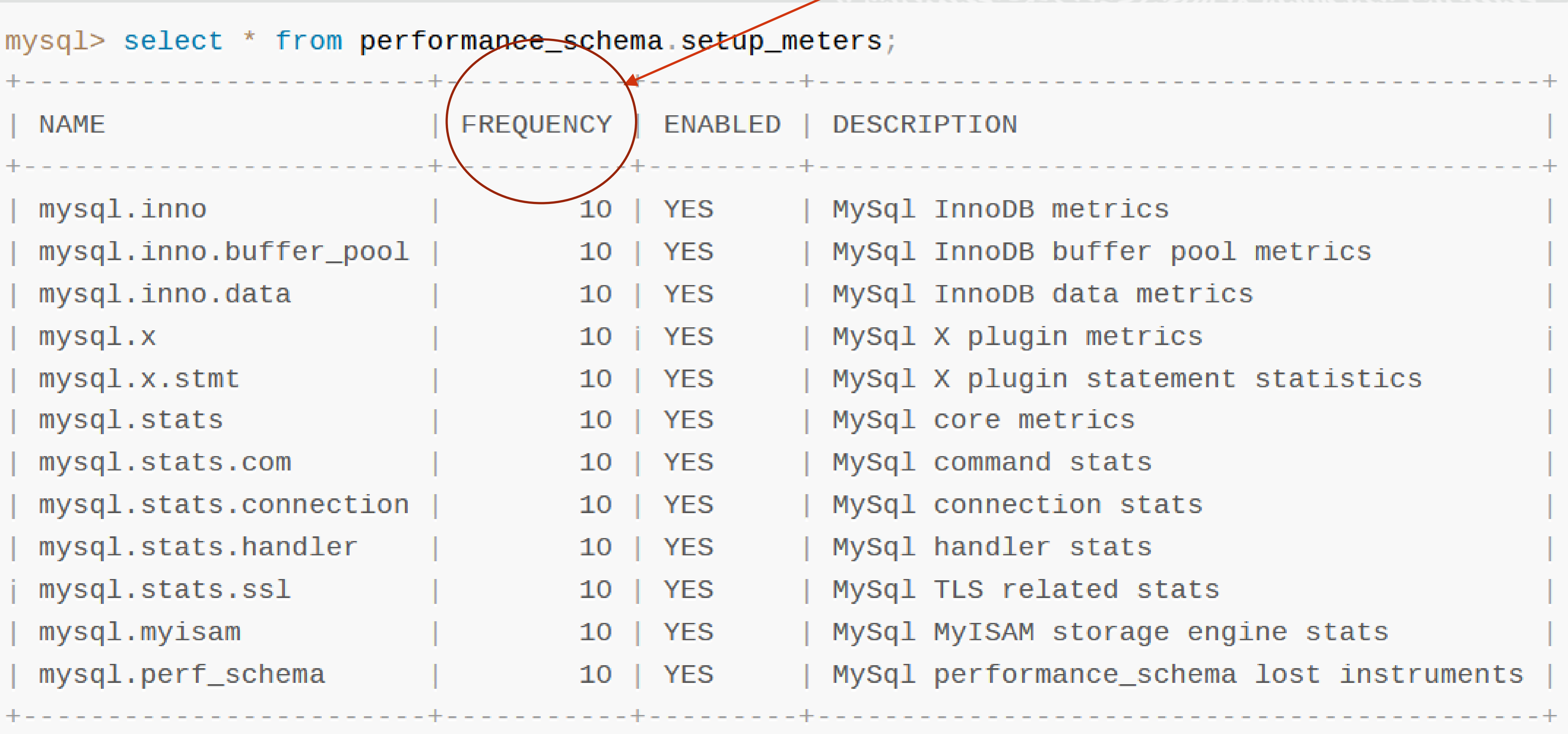
- **Metric**: A measurement captured at runtime
- **Metric Event**: The moment of capturing a measurement (value, timestamp, associated meta-data)
- **Meter**: Group of metrics, created by Meter Providers (e.g. MySQL).
- **Metric instrument**: Name, Kind, Unit (opt), Description (opt)
- Kind: Counter, Async Counter, UpDownCounter, Gauge, …
- **Metric Exporters**: Send metric data to a consumer

MySQL Meters & Metrics instruments

- MySQL 8.3 has defined 474 metrics instruments grouped into 15 meters
- **Meters** are observable: **performance_schema.setup_meters**
- **Metrics instruments** are observable : **performance_schema.setup_metrics**
- Configuration (on/off, export frequency) is defined per Meter
- Everything in SHOW GLOBAL STATUS is exposed as metrics.

MySQL Meters

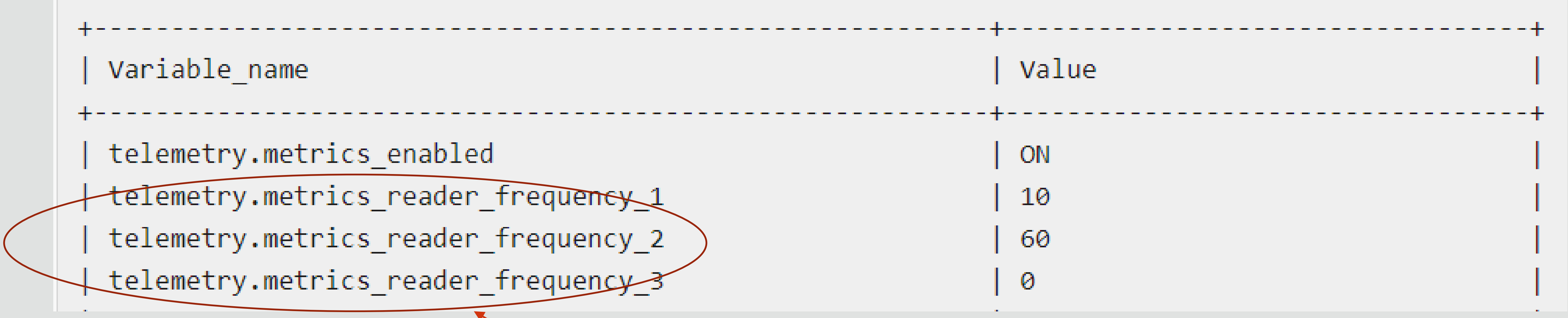
Intention: Actual Frequency rounded up to metrics_reader_frequency_1 , 2, 3 (see next slide)



```
mysql> select * from performance_schema.setup_meters;
+-------------------------+-----------+---------+-------------------------------------------+
| NAME                    | FREQUENCY | ENABLED | DESCRIPTION                               |
+-------------------------+-----------+---------+-------------------------------------------+
| mysql.inno              |        10 | YES     | MySql InnoDB metrics                      |
| mysql.inno.buffer_pool  |        10 | YES     | MySql InnoDB buffer pool metrics          |
| mysql.inno.data         |        10 | YES     | MySql InnoDB data metrics                 |
| mysql.x                 |        10 | YES     | MySql X plugin metrics                    |
| mysql.x.stmt            |        10 | YES     | MySql X plugin statement statistics       |
| mysql.stats             |        10 | YES     | MySql core metrics                        |
| mysql.stats.com         |        10 | YES     | MySql command stats                       |
| mysql.stats.connection  |        10 | YES     | MySql connection stats                    |
| mysql.stats.handler     |        10 | YES     | MySql handler stats                       |
| mysql.stats.ssl         |        10 | YES     | MySql TLS related stats                   |
| mysql.myisam            |        10 | YES     | MySql MyISAM storage engine stats         |
| mysql.perf_schema       |        10 | YES     | MySql performance_schema lost instruments |
+-------------------------+-----------+---------+-------------------------------------------+
```

MySQL Reader frequency

- Reader frequency is configurable but limited to 3 groups (reader threads)

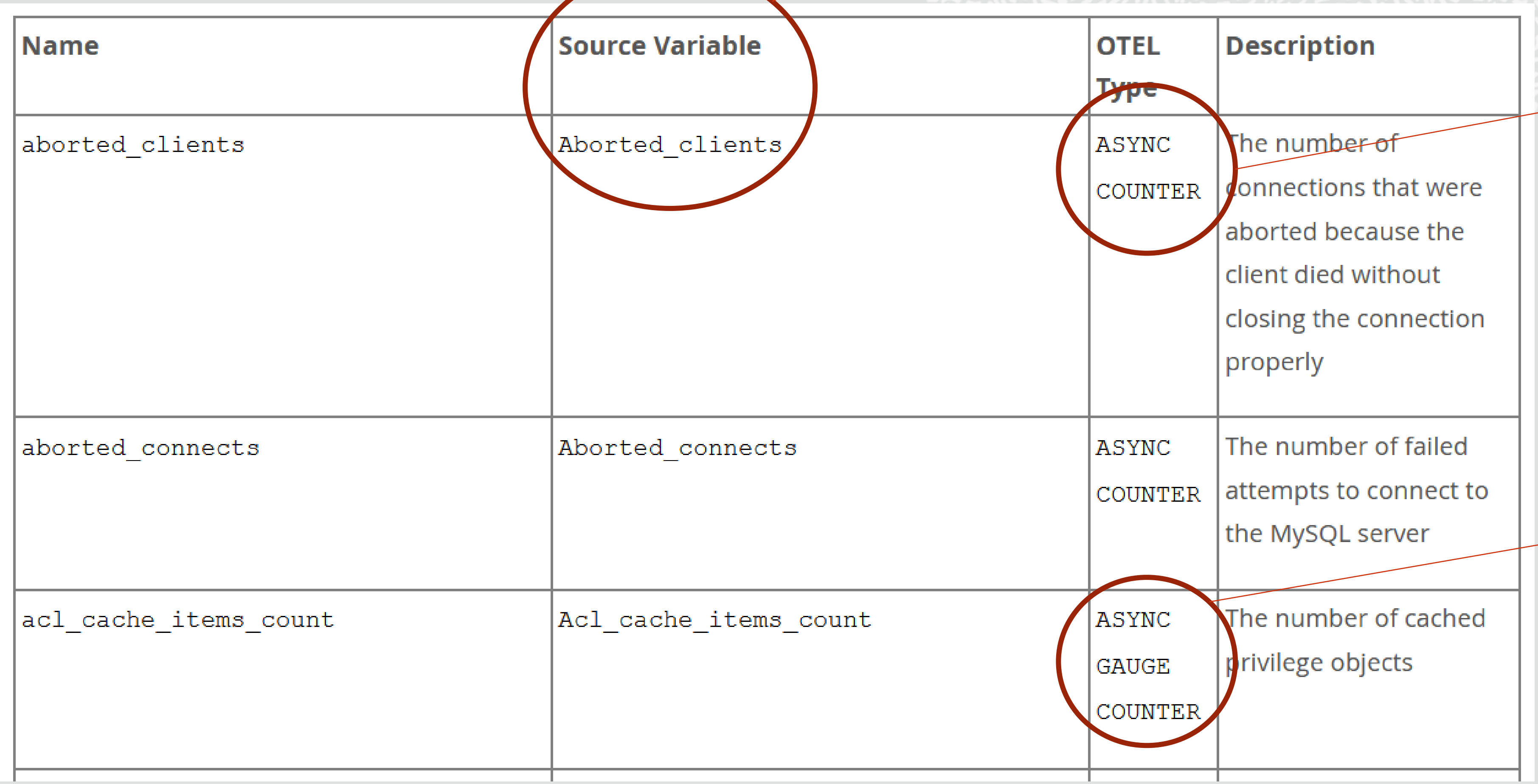


```
+-----------------------------------------------+----------------------------+
| Variable_name                                 | Value                      |
+-----------------------------------------------+----------------------------+
| telemetry.metrics_enabled                     | ON                         |
| telemetry.metrics_reader_frequency_1          | 10                         |
| telemetry.metrics_reader_frequency_2          | 60                         |
| telemetry.metrics_reader_frequency_3          | 0                          |
```

Meter frequency rounded up to metrics_reader_frequency_1 , 2, 3

MySQL Metrics

Source code variable = Metric instrument = GLOBAL STATUS



Name	Source Variable	OTEL Type	Description
aborted_clients	Aborted_clients	ASYNC COUNTER	The number of connections that were aborted because the client died without closing the connection properly
aborted_connects	Aborted_connects	ASYNC COUNTER	The number of failed attempts to connect to the MySQL server
acl_cache_items_count	Acl_cache_items_count	ASYNC GAUGE COUNTER	The number of cached privilege objects

Counter: Always increasing

All Counters are Async

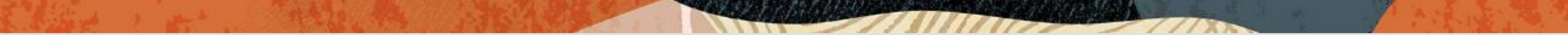
Gauge Counter: Value “as of now”

Final notes

- Background thread reads all metrics with a given frequency and then exported directly to backend (no buffering)
- It is asynchronous since we do not export in real time
- There is only one background thread reding a given metric, thus there is no race conditions.
- Metrics are not protected by mutexes (generally) , thus occasional garbage values may occur

References

- https://opentelemetry.io/
- https://dev.mysql.com/doc/refman/8.3/en/telemetry.html
- https://blogs.oracle.com/observability/post/application-monitoring-with-opentelemetry



ORACLE