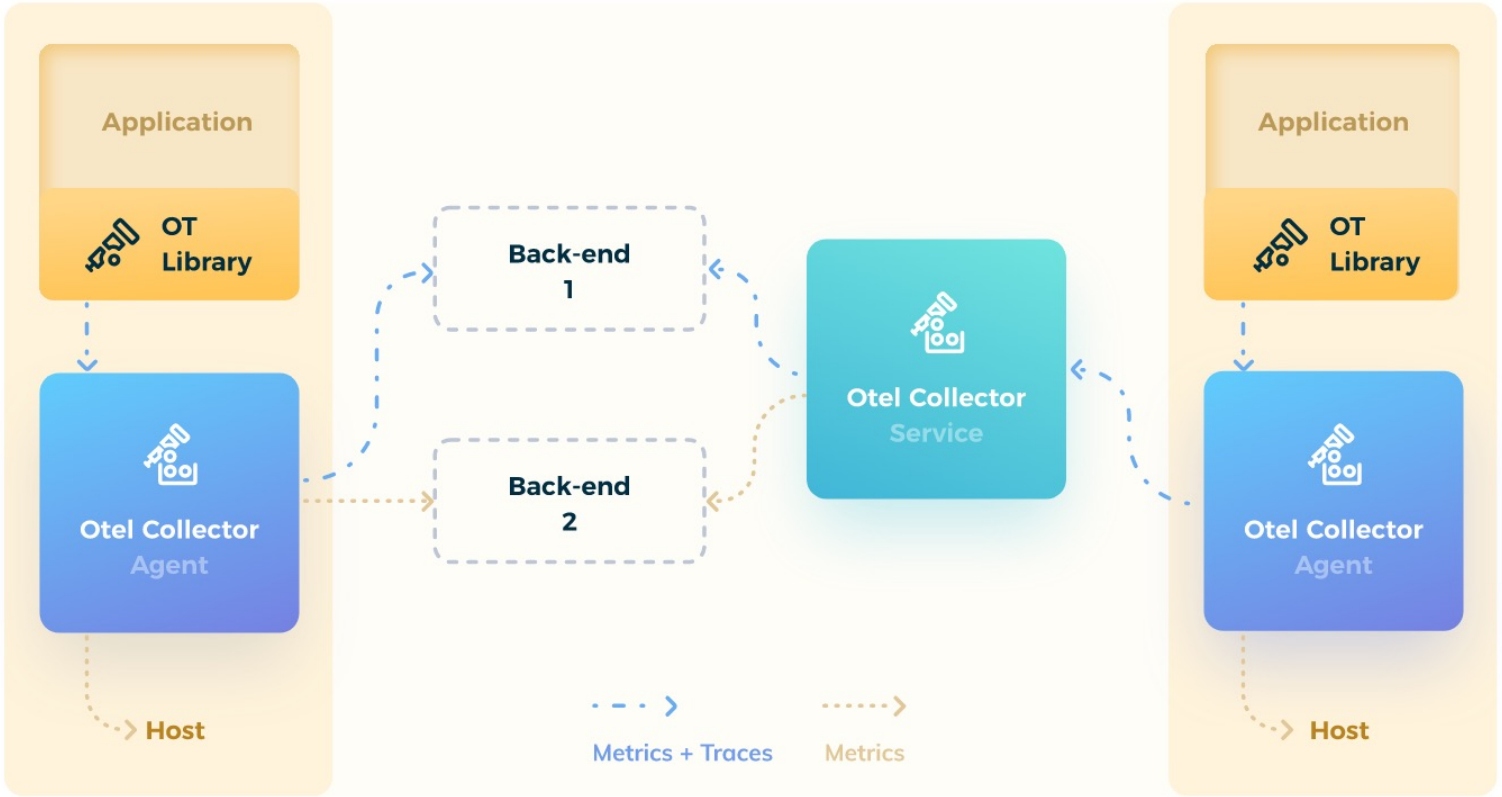
HIDDEN SECRETS ABOUT INSTRUMENTING JVMS FOR OPENTELEMETRY

RICARDO FERREIRA

@RIFERREI

OPENTELEMETRY CRASH SOURCE

OPENTELEMETRY OVERVIEW



QUICK DEMO

https://github.com/riferrei/otel-with-java


MUCH
BETTER
NOW 👍

RICARDO FERREIRA

DEVELOPER ADVOCATE

- ELASTIC COMMUNITY TEAM
- HASHICORP AMBASSADOR
- BEFORE WORKING FOR ELASTIC:

  CONFLUENT, ORACLE, RED HAT
- DISTRIBUTED SYSTEMS, O11Y, STREAMING SYSTEMS, DATABASES
- RIFERREI@ELASTIC.CO
- RIFERREI@RIFERREI.COM

elastic

JVM INSTRUMENTATION OPTIONS

```
java -javaagent:./${OTEL_AGENT} \
  -Dotel.traces.exporter=otlp \
  -Dotel.metrics.exporter=otlp \
  -jar target/app.jar
```

API, SDK, SDK-EXTENSIONS

80%

20%

MANUAL INSTRUMENTATION

```
SdkTracerProvider sdkTracerProvider = SdkTracerProvider.builder()
    .addSpanProcessor(BatchSpanProcessor.builder(OtlpGrpcSpanExporter.builder().build()).build())
    .build();

OpenTelemetry openTelemetry = OpenTelemetrySdk.builder()
    .setTracerProvider(sdkTracerProvider)
    .setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInstance()))
    .buildAndRegisterGlobal();
```

```
Tracer tracer =
    openTelemetry.getTracer("instrumentation-library-name", "1.0.0");
```

```
Span span = tracer.spanBuilder("my span").startSpan();
```

MANUAL INSTRUMENTATION

```
<dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-sdk-extension-autoconfigure</artifactId>
    <version>${version}</version>
</dependency>
```

```
OpenTelemetrySdk sdk = OpenTelemetrySdkAutoConfiguration.initialize();
```

```
Tracer tracer =
    openTelemetry.getTracer("instrumentation-library-name", "1.0.0");
```

```
Span span = tracer.spanBuilder("my span").startSpan();
```

DOWN THE RIVER VIA @WITHSPAN

```
<dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-extension-annotations</artifactId>
    <version>${version}</version>
</dependency>
```

```
@WithSpan 👍
public void someMethod() {
}
```

DISABLING AUTO INSTRUMENTATION

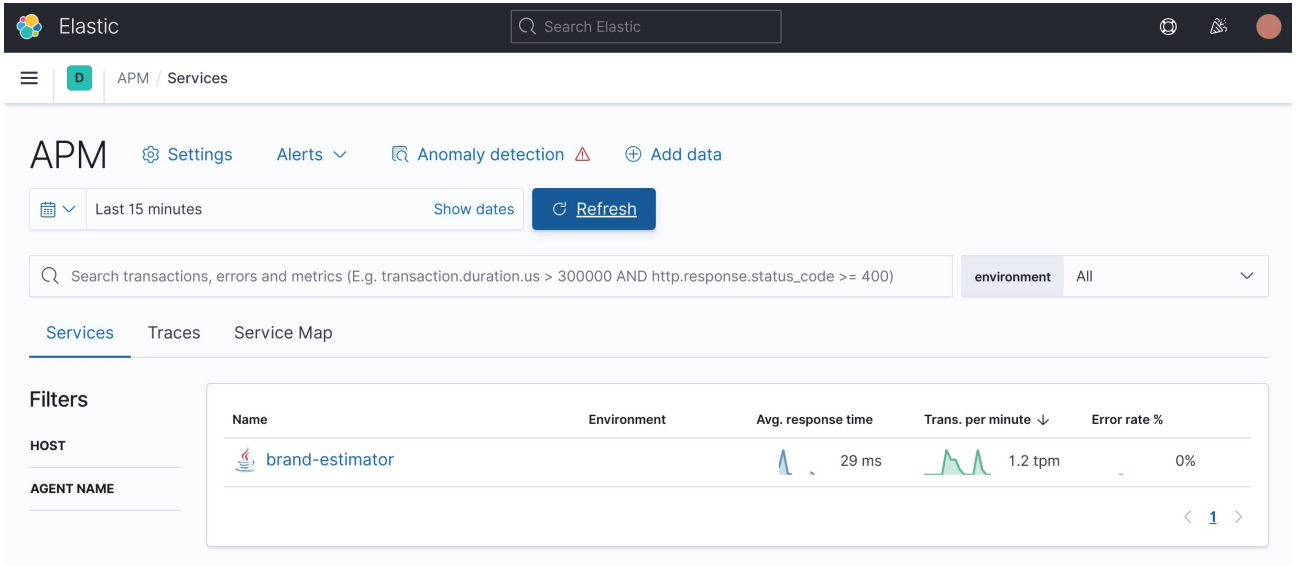
```
-Dotel.instrumentation.common.default-enabled=false
```

ENABLING ONLY MANUAL INSTRUMENTATION

```
-Dotel.instrumentation.opentelemetry-annotations.enabled=true
```

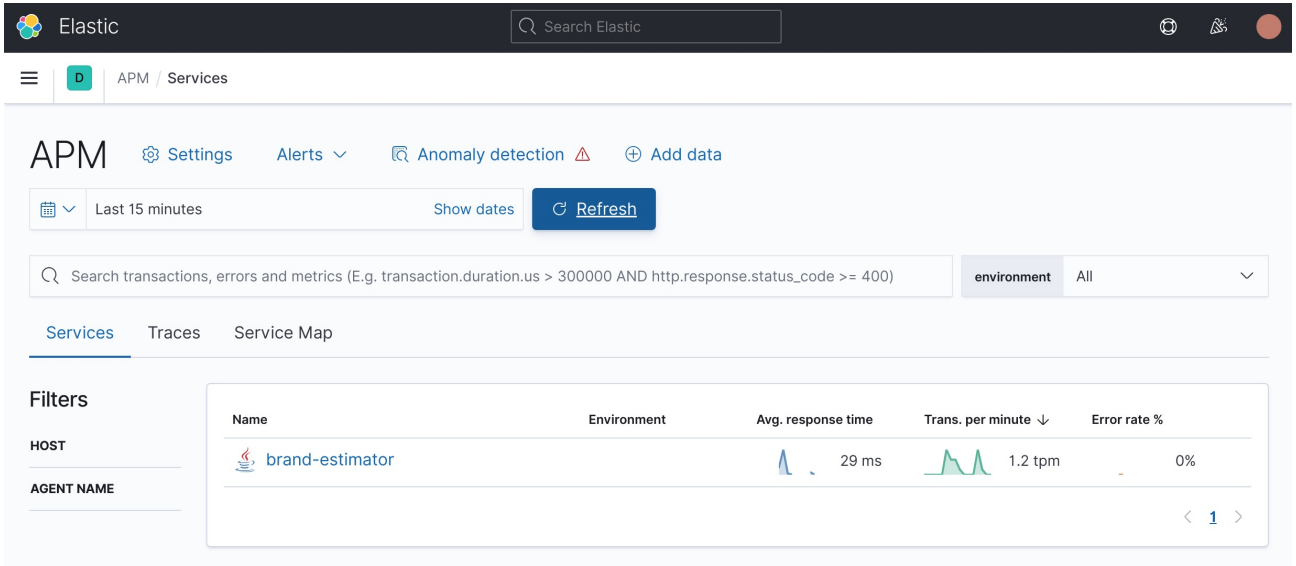
MEANINGFUL NAMES FOR SERVICES

```
-Dotel.resource.attributes=service.name= \
brand-estimator,service.version=1.0
```



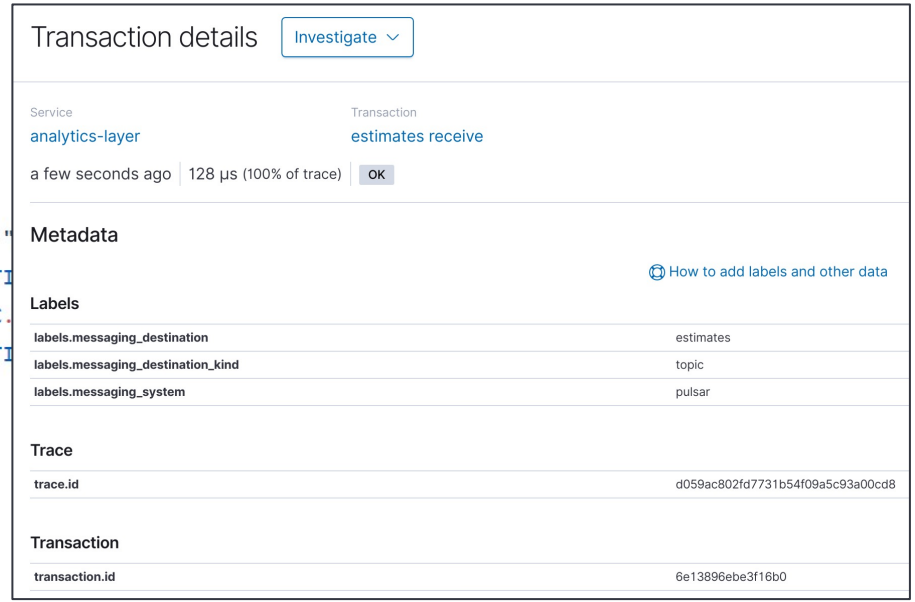
MEANINGFUL NAMES FOR SERVICES

```
export OTEL_RESOURCE_ATTRIBUTES=service.name= \
brand-estimator,service.version=1.0
```



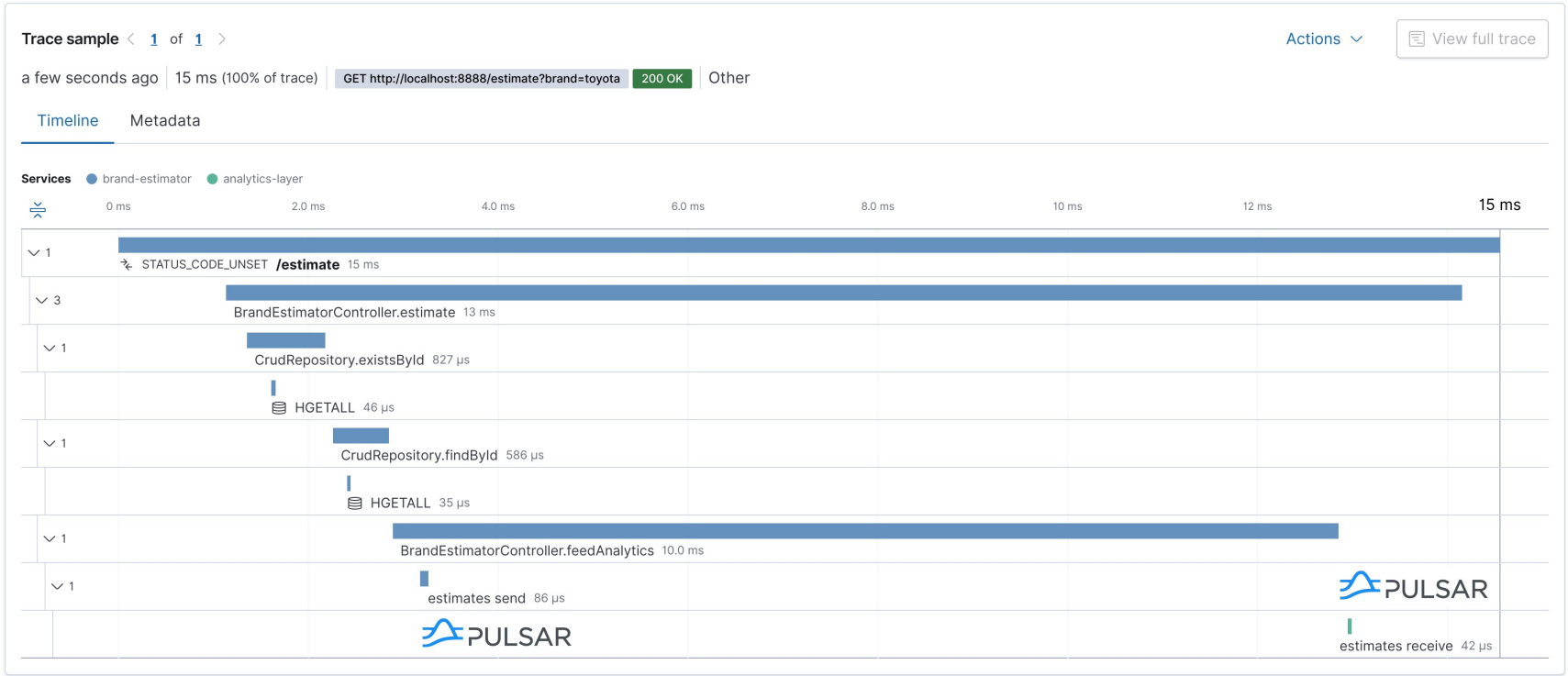
MAKING YOUR TRACES OBSERVABLE

```
String spanName = producer.getTopic() + " send";
Span sendSpan = tracer.spanBuilder(spanName)
    .setSpanKind(Span.Kind.PRODUCER)
    .startSpan();

Context newContext = Context.current().with(sendSpan);

try (Scope scope = newContext.makeCurrent()) {
    sendSpan.setAttribute(SemanticAttributes.MESSAGING_SYSTEM, "
    sendSpan.setAttribute(SemanticAttributes.MESSAGING_DESTINATI
        SemanticAttributes.MessagingDestinationKindValues.TOPIC.
    sendSpan.setAttribute(SemanticAttributes.MESSAGING_DESTINATI
    storeContextOnMessage(newContext, message);
} finally {
    sendSpan.end();
}
```



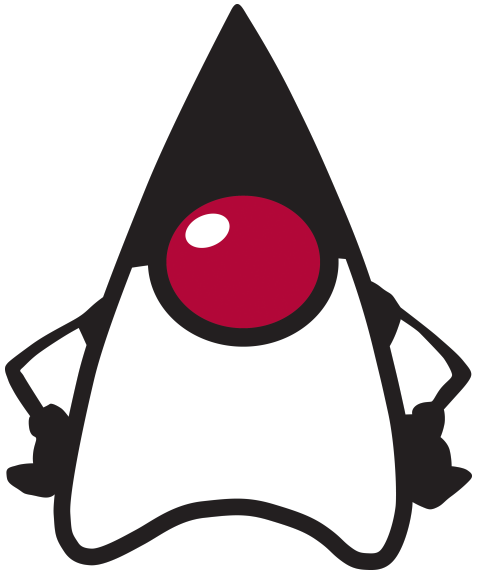
elastic

USING TEXT-MAP-PROPAGATORS



elastic

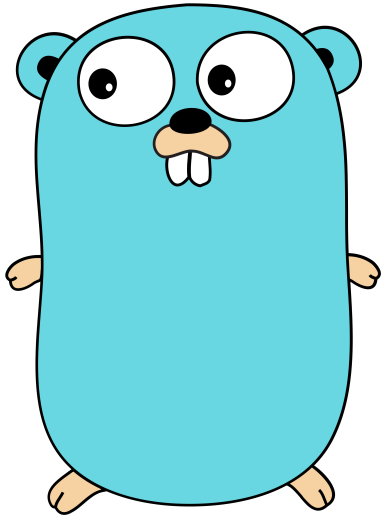
USING TEXT-MAP-PROPAGATORS



```
TextMapSetter<Message<?>> setter =
    new TextMapSetter<>(){

        @Override
        public void set(Message<?> message, String key, String value) {
            MessageImpl<?> msg = null;
            if (message instanceof MessageImpl<?>) {
                msg = (MessageImpl<?>) message;
            } else if (message instanceof TopicMessageImpl<?>) {
                msg = (MessageImpl<?>) ((TopicMessageImpl<?>) message).getMessage();
            }
            if (msg != null) {
                msg.getMessageBuilder().addProperties(
                    PulsarApi.KeyValue.newBuilder()
                    .setKey(key).setValue(value));
            }
        }
```

```
GlobalOpenTelemetry.getPropagators().getTextMapPropagator()
    .inject(context, message, setter);
```

USING TEXT-MAP-PROPAGATORS



```
extractedContext := otel.GetTextMapPropagator().Extract(ctx, PulsarCarrier{message})
_, receiveSpan := tracer.Start(extractedContext, topicName+" receive",
    trace.WithAttributes(
        semconv.MessagingSystemKey.String("pulsar"),
        semconv.MessagingDestinationKindKey.String("topic"),
        semconv.MessagingDestinationKey.String(topicName),
    ))
```

```
type PulsarCarrier struct {
    Message pulsar.Message
}

func (pulsar PulsarCarrier) Get(key string) string {
    return pulsar.Message.Properties()[key]
}

func (pulsar PulsarCarrier) Set(key string, value string) {
    pulsar.Message.Properties()[key] = value
}

func (pulsar PulsarCarrier) Keys() []string {
    return []string{pulsar.Message.Key()}
}
```

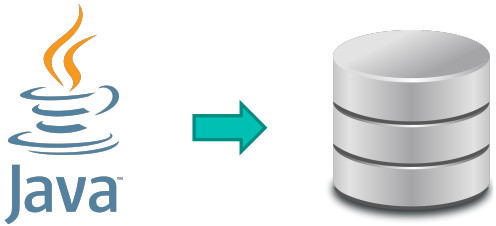
CAN I TELL YOU A SECRET? 🤫

```
SELECT a FROM b WHERE password=?
```

```
SELECT a FROM b WHERE password="secret"
```

System property	Environment variable	Description
`otel.instrumentation.common.db-statement-sanitizer.enabled`	`OTEL_INSTRUMENTATION_COMMON_DB_STATEMENT_SANITIZER_ENABLED`	Enables the DB statement sanitization. The default value is `true`.

DATABASES CAN BE A BIT CHATTY

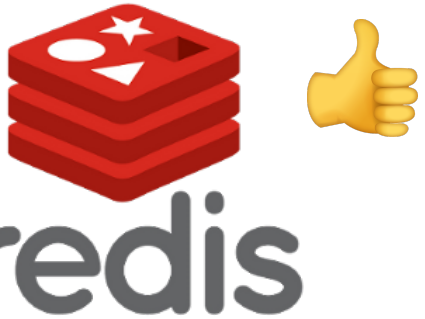


`java.sql.Datasource#getConnection()`

`-Dotel.instrumentation.jdbc-datasource.enabled=`**`true`**

EXCLUDING NOISY FRAMEWORKS

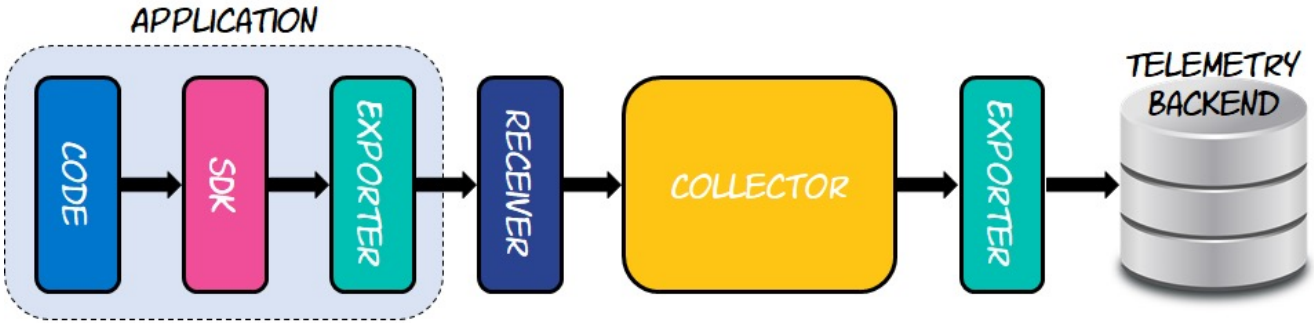
👎



👍

System property	Environment variable	Purpose
otel.javaagent.exclude-classes	OTEL_JAVAAGENT_EXCLUDE_CLASSES	Suppresses all instrumentation for specific classes, format is "my.package.MyClass,my.package2.*"

USING THE BATCH-SPAN-PROCESSOR



System property	Environment variable	Description
otel.bsp.schedule.delay	OTEL_BSP_SCHEDULE_DELAY	The interval, in milliseconds, between two consecutive exports. Default is `5000`.
otel.bsp.max.queue.size	OTEL_BSP_MAX_QUEUE_SIZE	The maximum queue size. Default is `2048`.
otel.bsp.max.export.batch.size	OTEL_BSP_MAX_EXPORT_BATCH_SIZE	The maximum batch size. Default is `512`.
otel.bsp.export.timeout	OTEL_BSP_EXPORT_TIMEOUT	The maximum allowed time, in milliseconds, to export data. Default is `30000`.

PLANNING YOUR SAMPLING STRATEGY

System property	Environment variable	Description
otel.traces.sampler	OTEL_TRACES_SAMPLER	The sampler to use for tracing. Defaults to `parentbased_always_on`
otel.traces.sampler.arg	OTEL_TRACES_SAMPLER_ARG	An argument to the configured tracer if supported, for example a ratio.

Supported values for `otel.traces.sampler` are

- "always_on": AlwaysOnSampler
- "always_off": AlwaysOffSampler
- "traceidratio": TraceIdRatioBased. `otel.traces.sampler.arg` sets the ratio.
- "parentbased_always_on": ParentBased(root=AlwaysOnSampler)
- "parentbased_always_off": ParentBased(root=AlwaysOffSampler)
- "parentbased_traceidratio": ParentBased(root=TraceIdRatioBased). `otel.traces.sampler.arg` sets the ratio.

DEBUGGING THE AGENT

```
-Dotel.javaagent.debug=true
```

```
export OTEL_JAVAAGENT_DEBUG=true
```

DISABLING THE AGENT

```
-Dotel.javaagent.enabled=false
```

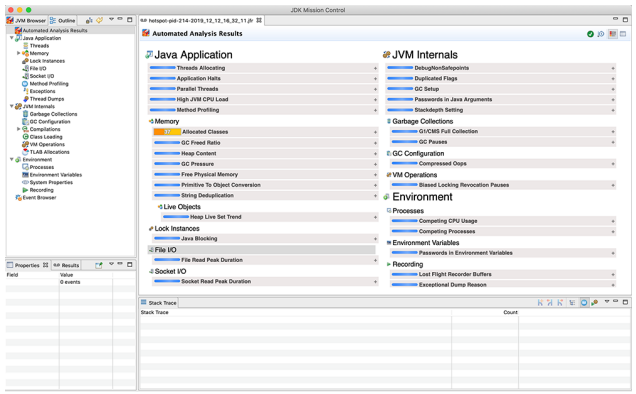
```
export OTEL_JAVAAGENT_ENABLED=false
```

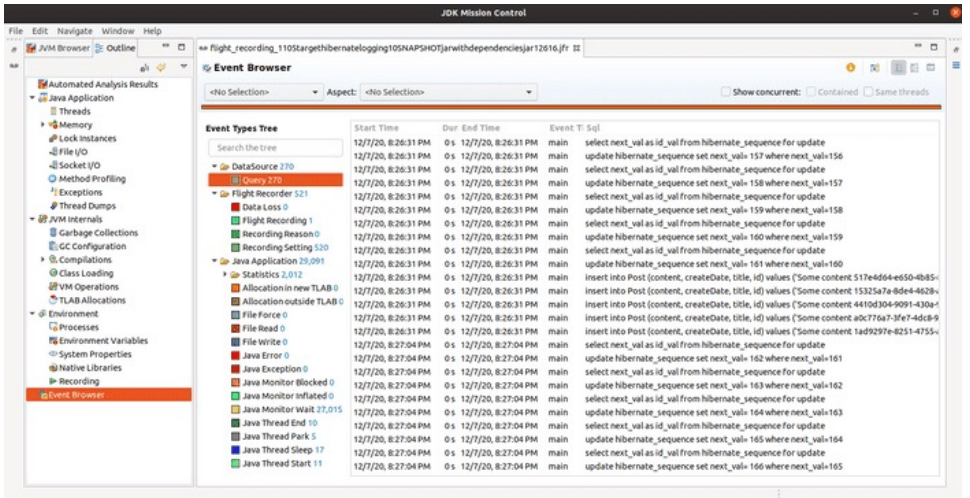
OBSERVABLE METRICS: NO PUN INTENDED

```
private static final LongValueObserver heapMemory =
    meter
        .longValueObserverBuilder(HEAP_MEMORY_NAME)
        .setDescription(HEAP_MEMORY_DESCRIPTION)
        .setUnit("byte")
        .setUpdater(
            result -> result.observe(
                getRuntime().totalMemory() - getRuntime().freeMemory(),
                Labels.of(HEAP_MEMORY_NAME, HEAP_MEMORY_DESCRIPTION))
        )
        .build();
```

INSPECTING TELEMETRY DATA WITH JMC

```
<dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-sdk-extension-jfr-events</artifactId>
    <version>${version}</version>
</dependency>
```





elastic

THANK YOU

RICARDO FERREIRA

@RIFERREI