Observing Cloud-Native Java Apps using OpenTelemetry on AWS, GCP and Azure

Bernhard Lubomski, RETIT GmbH, 2022/03/15
Three Pillars of Observability

- Traces
- Metrics
- Logs

focus of this talk
Traces and Metrics

Trace = directed, acyclic graph of spans
Span = represents unit of work. Properties:
  • parent/child relation
  • start time, end time
  • tags, logs, errors...

Metric: time series of numbers to compute statistical values
  • count
  • average
  • percentiles (median, 25th, 90th, ...)
  • ...
OpenTelemetry (OTEL)

- Observability Framework
- Collection of telemetry data: traces, metrics, logs

- Defines **data formats** and **protocols** for traces, metrics and logs
- Provides **APIs** for recording traces, metrics and logs
- Provides **SDKs** with implementation of trace, metrics and logs* recording
- Provides **Java auto-instrumentation** agent (https://github.com/open-telemetry/opentelemetry-java-instrumentation)

- Excludes: Tools for trace/metric/log storage, retrieval and visualization

*Logs: in draft stage (2022/03/07) https://opentelemetry.io/status/
Anatomy of an Observability Solution

Scope of OTEL

- telemetry data format and protocols
- agent API / SDK for multiple languages
- Collector implementation
- Java: auto-instrumentation agent

OpenTelemetry Java Auto-Instrumentation

Common libraries and frameworks are supported, and calls are automatically traced.

Source 2022/01/27: https://github.com/open-telemetry/opentelemetry-java-instrumentation/blob/main/docs/supported-libraries.md


Application Server:
Jetty 9.4.x, 10.0.x, 11.0.x, Payara 5.0.x, 5.1.x, Tomcat 7.0.x, Tomcat 7.0.x, 8.5.x, 9.0.x, 10.0.x, TomEE 7.x, 8.x, Websphere Liberty Profile 20.x, 21.x, Websphere Traditional 8.5.5.x, 9.0.x, WildFly 13.x, WildFly 17.x, 21.x, 25.x

Spring-*
JDBC, Hibernate
HTTP-Clients
JAX-WS, JAX-RS
JMS
Loggers
...

† http4k is an alternative to Ktor.
Typical OTEL Setup for Java

- Application
  - OTEL Java auto-instrumentation agent
- OTEL Collector
- Zipkin
- Jaeger
- Prometheus

Flow:
- Application -> OTEL Collector
- Traces: Application -> Jaeger
- Metrics: Application -> Prometheus
OpenTelemetry: Supported by Cloud Providers

• How is OTEL integrated and supported in AWS, GCP and Azure?

• Which are the benefits of the integration?

• How to make applications observable with OTEL in these cloud environments?
OTEL Integration by Cloud Providers

Agent → Collector → Server → UI

Features:
- storage, visualization, analysis, alerting, access control...

+ cloud-specific extensions

Providers:
- aws
- cloud
- a
OpenTelemetry support: use managed trace and metrics services through OpenTelemetry + extensions. Normally, usage of vendor proprietary SDKs to record traces and metrics is required.

**OTEL benefits over vendor proprietary SDKs:**
- Java auto-instrumentation
- Vendor neutral OTEL API for (manually) recording traces and metrics

<table>
<thead>
<tr>
<th>Trace Service</th>
<th>Metrics Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>X-Ray</td>
</tr>
<tr>
<td>GCP</td>
<td>Cloud Trace (formerly “Stackdriver”)</td>
</tr>
<tr>
<td>Azure</td>
<td>Azure Monitor Application Insights</td>
</tr>
</tbody>
</table>
Architectural Blueprint for OTEL integration by Cloud Providers

Application

OTEL SDK + ?

OTEL Collector + ?

AWS / GCP / Azure (/ other)

Tracing Service

Metrics Service

AWS / GCP / Azure managed services

Traces

Metrics
AWS Distribution for OpenTelemetry (ADOT) consists of:

- Java-auto-instrumentation agent redistribution with AWS pre-configuration (ADOT Agent)
  https://github.com/aws-observability/aws-otel-java-instrumentation

- Collector redistribution with AWS exporters for sending telemetry to X-Ray and CloudWatch (ADOT Collector)
  https://github.com/aws-observability/aws-otel-collector
AWS OTEL Integration, Collector Sidecar

Application

ADOT Agent

ADOT Collector
(e.g., sidecar deployment)

EC2/ECS/EKS/Fargate/Lambda

Traces

AWS X-Ray

AWS CloudWatch

Metrics

AWS managed services

AWS OTEL Integration, Collector Sidecar

15. March 2022 • www.retit.de • 13
AWS OTEL Integration, Collector Service

Application

ADOT Agent

ADOT Collector (central collector(s))

EC2/ECS/EKS/Fargate/Lambda

Traces

AWS X-Ray

AWS CloudWatch

Metrics

AWS managed services

EC2/ECS/EKS/Fargate/Lambda

AWS managed services
AWS CloudWatch

- Metrics storage
- Metrics visualization
- Alerts
- …

X-Ray console shown on last slide was recently integrated into CloudWatch view

Source https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/graph_a_metric.html
OTEL in GCP

- Relies on OTEL API/SDK and code to manually record traces and metrics.
- Transmission of telemetry to GCP via extension of OTEL SDK:

- **Traces** support: stable
- **Metrics** support: still based on OTEL Alpha SDK for metrics (2022/03/07)

- **OTEL auto-instrumentation** support: Exists, but based on still unstable custom exporter feature in OTEL, therefor considered “proof-of-concept“ (2022/03/07)
GCP OTEL Integration

Cloud Trace

Cloud Monitoring Metrics

managed services of the GCP Operations Suite (formerly Stackdriver)

Application

OTEL metrics collection code

OTEL trace collection code

Google Cloud Trace Exporter

Google Cloud Monitoring Exporter

OTEL API/SDK

GCP / on-premise / other Cloud Provider

Traces

Metrics
Workaround for recording Traces using Java- Auto-instrumentation in GCP

GCP auto-instrumentation exporter considered proof-of-concept, but Zipkin trace exporter is integrated into the base OTEL auto-instrumentation client.

Cloud Trace

Cloud Trace

Traces (Zipkin format)

GCP / on-premise / Other Cloud Provider

OTEL Java auto-instrumentation agent

Zipkin exporter

GCP / on-premise / Other Cloud Provider

Cloud Trace Zipkin Collector

https://github.com/openzipkin/zipkin-gcp
GCP Cloud Trace

- Trace storage
- Trace visualization
- Statistics on traces
- Suggestions on recognized bottlenecks
GCP Cloud Monitoring Metrics

- Metrics storage
- Metrics visualization
- Alerts
- …
OTEL in Azure

- Java support based on OTEL auto-instrumentation, GA‘d 2020
  - https://github.com/microsoft/ApplicationInsights-Java
- Direct transmission of traces to Azure application insights.

- No support for custom traces and metrics captured through OTEL API, (yet?).
  - Custom traces: application-insights.2.x SDK API
  - Custom metrics: Micrometer API, application-insights.2.x SDK API
Azure OTEL Integration

Azure Monitor Auto-Instrumentation for Java Agent (based on OTEL + Azure SDK + Micrometer + ...)

Application

on-premises / *-cloud

Traces

Metrics

Azure Monitor Application Insights
(Traces + Metrics)

Azure managed services
Azure Monitor Application Insights (Transactions)

- Trace storage
- Trace visualization
- Profiling
- ...

Source:
https://docs.microsoft.com/en-us/azure/azure-monitor/app/java-2x-trace-logs
https://docs.microsoft.com/en-us/azure/azure-monitor/app/transaction-diagnostics
Azure Monitor Metrics

- Metrics storage
- Metrics visualization
- Alerts
- ...

Source: https://docs.microsoft.com/en-us/azure/azure-monitor/essentials/data-platform-metrics
Auto-Instrumentation Config for AWS and Azure

Add Java agent to Java process, configure service name for telemetry data via env-property:

```
OTEL_RESOURCE_ATTRIBUTES=service.name=MyApp,service.namespace=MyTeam
java -javaagent:path/to/agent.jar -jar myapp.jar
```

Agents

AWS: https://github.com/aws-observability/aws-otel-java-instrumentation/releases
Azure: https://github.com/microsoft/ApplicationInsights-Java/releases
OTEL: https://github.com/open-telemetry/opentelemetry-java-instrumentation/releases

Additional Configurations

AWS:

- OTEL_TRACES_SAMPLER=parentbased_traceidratio
- OTEL_TRACES_SAMPLER_ARG=0.1
- OTEL_EXPORTER_OTLP_ENDPOINT=https://mycollectorhost:4317

+ ADOT Collector Setup: https://aws-otel.github.io/docs/getting-started/collector

Azure:

- APPLICATIONINSIGHTS_CONNECTION_STRING=InstrumentationKey=...
Setup GCP OTEL Traces Recording

Dependencies (Gradle):

GCP Trace exporter:
implementation 'com.google.cloud.opentelemetry:exporter-trace:0.20.0' (exporter-metrics-0.20.0-alpha)

Using only the OTEL SDK:
implementation 'io.opentelemetry:opentelemetry-sdk:1.10.1' (opentelemetry-sdk-metrics separate!)

```java
import com.google.cloud.opentelemetry.trace.TraceConfiguration;
import com.google.cloud.opentelemetry.trace.TraceExporter;
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;

TraceExporter = TraceExporter.createWithConfiguration(TraceConfiguration.builder()
    .setProjectId("MY_GCP_PROJECT_ID").build());
//...
OpenTelemetrySdk opentelemetry = OpenTelemetrySdk.builder()
    .setTracerProvider(SdkTracerProvider.builder()
        .addSpanProcessor(BatchSpanProcessor.builder(traceExporter).build())
        .build()).buildAndRegisterGlobal();
```

Refer to: https://github.com/GoogleCloudPlatform/opentelemetry-operations-java/blob/main/exporters/trace/README.md
Recording Custom OTEL Traces

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

// ...

Span span = tracer.spanBuilder("new span").startSpan();
// put the span into the current Context
try (Scope scope = span.makeCurrent()) {
    // ... your business code
} catch (Throwable t) {
    span.setStatus(StatusCode.ERROR, "Error Message");
} finally {
    span.end();
}
```

Refer to: https://opentelemetry.io/docs/instrumentation/java/manual/#tracing
Recording Custom OTEL Metrics

// MeterProvider => access required, API in alpha, depends on OTEL release
// Gets or creates a named meter instance
Meter meter = meterProvider.meterBuilder("instrumentation-lib-name")
    .setInstrumentationVersion("1.0.0").build();

// Build counter e.g. LongCounter
LongCounter counter = meter.counterBuilder("my_counter")
    .setDescription("My Counter").setUnit("1").build();

// It is recommended that the API user keep a reference to a Bound
// Counter for the entire time or call unbind when no-longer needed.
BoundLongCounter someWorkCounter = counter.bind(
    Attributes.of(stringKey("Key"), "SomeWork"));

// Record data
someWorkCounter.add(123);

Refer to: https://opentelemetry.io/docs/instrumentation/java/manual/#metrics-alpha-only
## Cost

<table>
<thead>
<tr>
<th>Provider (Price Overview Link)</th>
<th>Billed by</th>
<th>Free tier?</th>
<th>cost / unit (/month)</th>
<th>Pricing Calculator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS</strong></td>
<td># traces recorded&lt;br&gt;# traces retrieved&lt;br&gt;# custom metrics, ingested data volume, archived data volume …</td>
<td>Yes</td>
<td>$5.00 / 1 Mio. traces recorded, $0.50 / 1 Mio. traces retrieved, $0.30 / custom metric, $0.63 / 1GB ingested, … (region: EU Frankfurt)</td>
<td><a href="https://calculator.aws/#/createCalculator/xray">https://calculator.aws/#/createCalculator/xray</a>&lt;br&gt;<a href="https://calculator.aws/#/createCalculator/CloudWatch">https://calculator.aws/#/createCalculator/CloudWatch</a></td>
</tr>
<tr>
<td><strong>GCP</strong></td>
<td># spans recorded, metrics data volume&lt;br&gt;# Monitoring API calls</td>
<td>Yes</td>
<td>$0.20 / Mio. spans recorded, $0.2580 / MiB (first 150-100k MiB, …) …</td>
<td><a href="https://cloud.google.com/products/calculator">https://cloud.google.com/products/calculator</a></td>
</tr>
<tr>
<td><strong>Azure</strong></td>
<td>ingested data volume (<em>&quot;As-You-Go&quot; or Commitment-tiered</em>)&lt;br&gt;# custom metrics, # metrics queries …</td>
<td>Yes</td>
<td>$2.99 / GB (Pay-As-You-Go)&lt;br&gt;$0.258/MB (first 150-100k MB, …) (region: Germany West Central)</td>
<td><a href="https://azure.microsoft.com/en-us/pricing/calculator/">https://azure.microsoft.com/en-us/pricing/calculator/</a></td>
</tr>
</tbody>
</table>

=> Use the pricing calculators!
Migration to Other Commercial Observability Tools

- OTEL is supported by Dynatrace, AppDynamics, NewRelic, Datadog, …
  - These observability tools can ingest OTEL telemetry.

- Custom OTEL trace/metrics recording requires little to no adaption, due to being vendor neutral.
  - Azure: Exception, since custom recording code is vendor specific.

- ADOT Collector: Out of the box support to configure telemetry export to commercial tools.
Thanks a lot for your attention!

Questions?

lubomski@retit.de