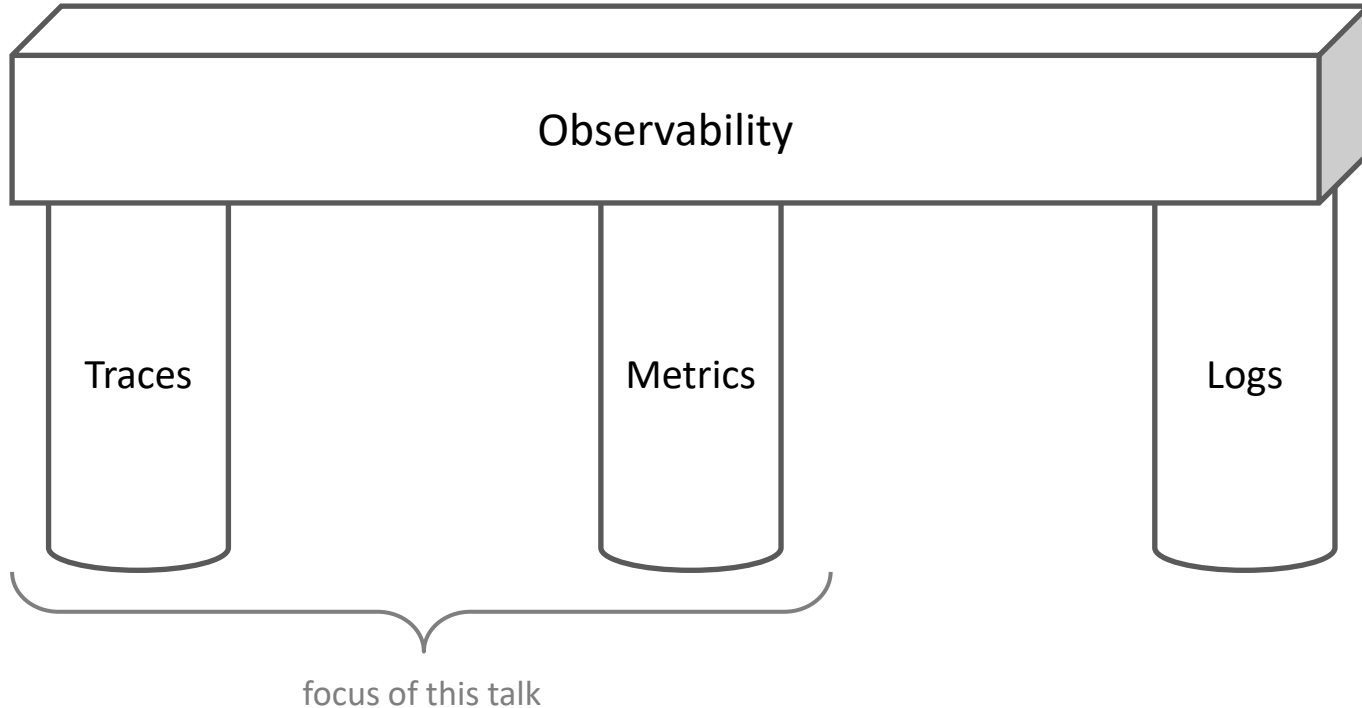


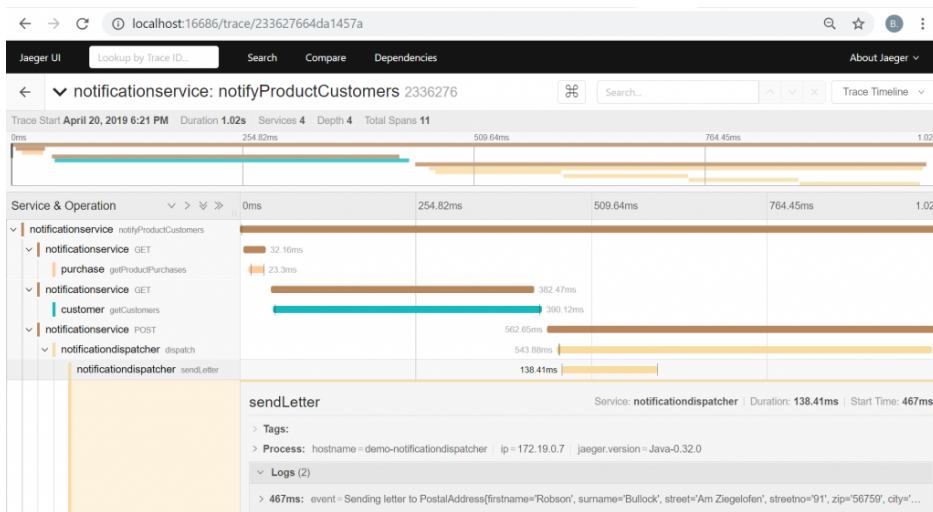
Observing Cloud-Native Java Apps using OpenTelemetry on AWS, GCP and Azure

Bernhard Lubomski, RETIT GmbH, 2022/03/15

Three Pillars of Observability



Traces and Metrics



Jaeger

Trace = directed, acyclic graph of spans
Span = represents unit of work. Properties:

- parent/child relation
- start time, end time
- tags, logs, errors...



Source: <https://grafana.com/products/cloud>

Metric: time series of numbers to compute statistical values

- count
- average
- percentiles (median, 25th, 90th, ...)
- ...

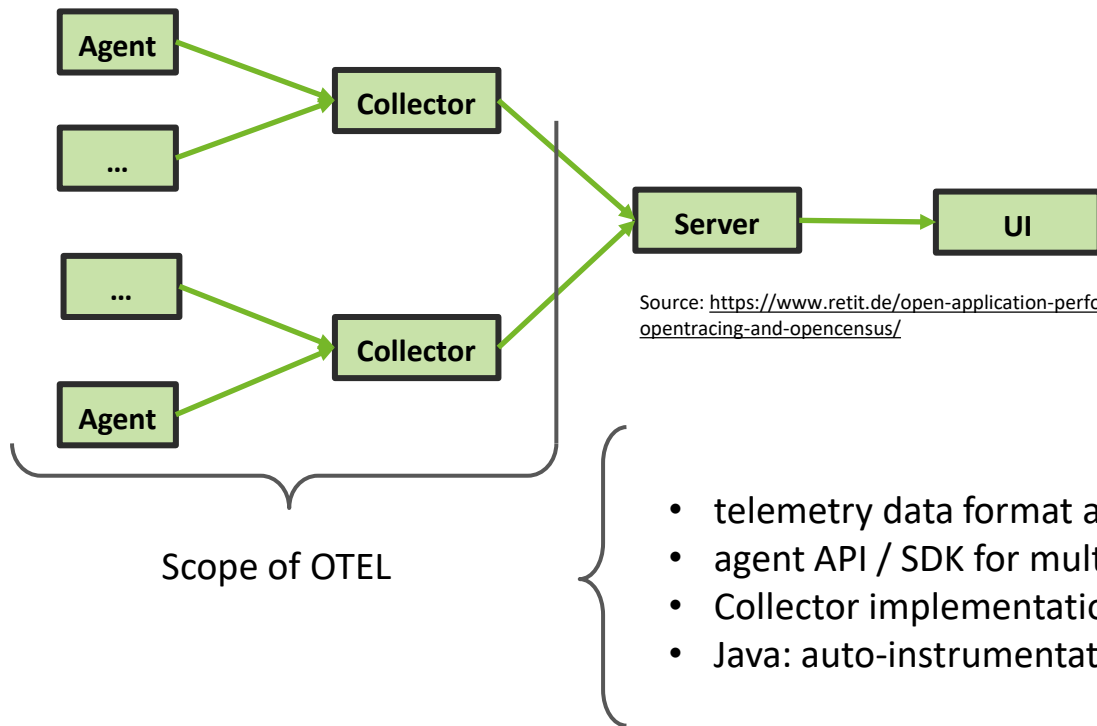
OpenTelemetry (OTEL)

<https://opentelemetry.io>

- 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



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>

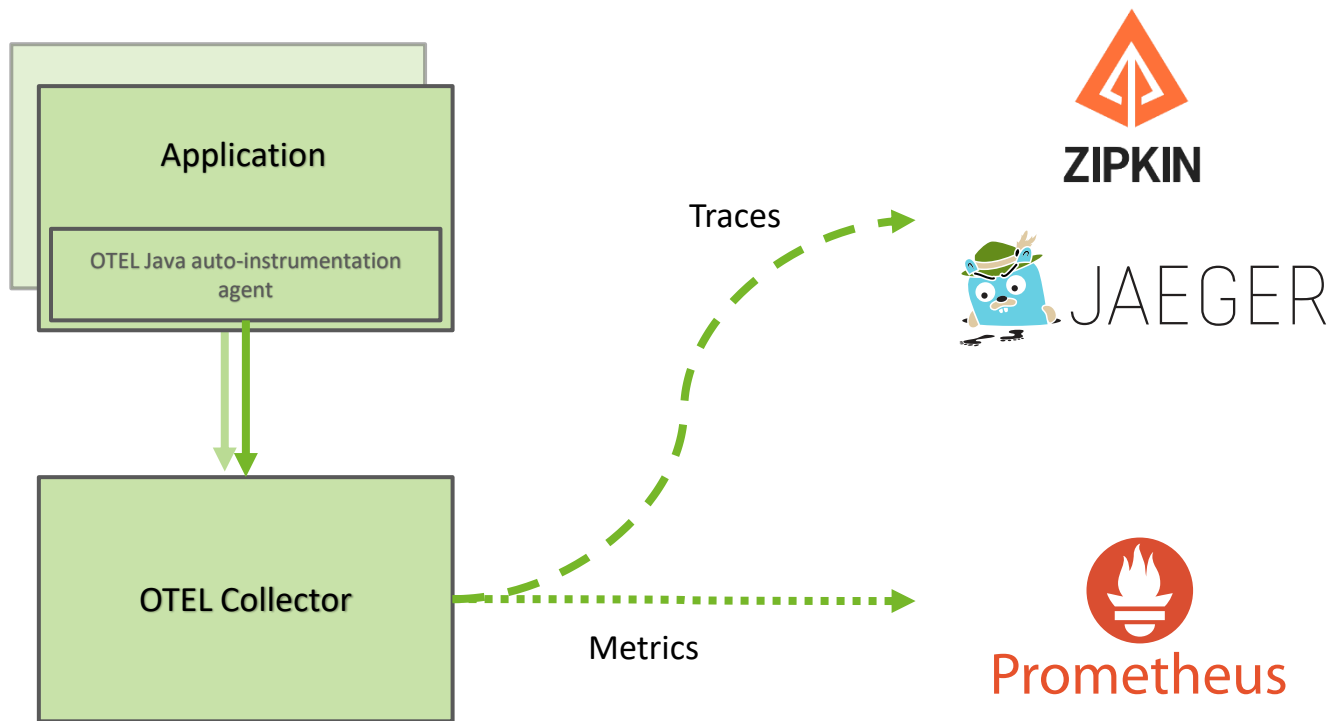
Akka Actors, Akka HTTP, Apache Axis2, Apache Camel, Apache CXF JAX-RS, Apache CXF JAX-RS Client, Apache Dubbo, Apache HttpAsyncClient, Apache HttpClient, Apache Kafka Producer/Consumer API, Apache MyFaces, Apache RocketMQ, Apache Struts 2, Apache Tapestry, Apache Wicket, Armeria, AWS Lambda, AWS SDK, Cassandra Driver, Couchbase Client, Dropwizard Views, Eclipse Grizzly, Eclipse Jetty, Eclipse Metro, Eclipse Mojarra, Elasticsearch API, Elasticsearch REST Client, Finatra, Geode Client, Grails, gRPC, Guava ListenableFuture, GWT, Hibernate, HttpURLConnection, http4k, Hystrix, Java EE, java.util.logging, JAX-RS, JAX-RS Client, JAX-WS, JDBC, Jedis, JMS, JSP, Kotlin Coroutines, Kubernetes, Log4j 2, Logback, Micrometer, MongoDB Driver, Netty, OkHttp, Play, Play WS, Quartz, RabbitMQ Client, Reactor Netty, Redis, Redisson, RESTEasy, Restlet, RMI, RxJava, Scala ForkJoinPool, Servlet, Spark, Spring Batch, Spring Data, Spring Integration, Spring Kafka, Spring RabbitMQ, Spring Scheduling, Spring Web Services, Spring WebFlux, Spymemcached, Twilio, Undertow, Vaadin, Vert.x Web, Vert.x HttpClient, Vert.x

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

Application Server:

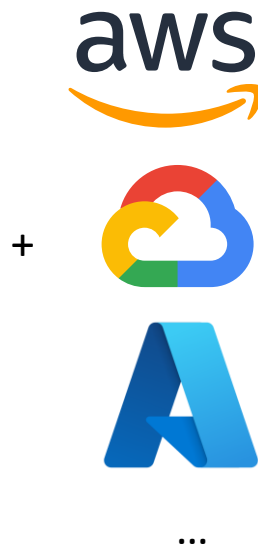
Jetty 9.4.x, 10.0.x, 11.0.x, Payara 5.0.x, 5.1.x, Tomcat 7.0.x, 8.5.x, 9.0.x, 10.0.x, TomEE 7.0.x, Liberty Profile 20.x, 21.x, Websphere Traditional 8.5.5.x, 9.0.x, WildFly 13.x, WildFly 17.x, 21.x, 25.x

Typical OTEL Setup for Java

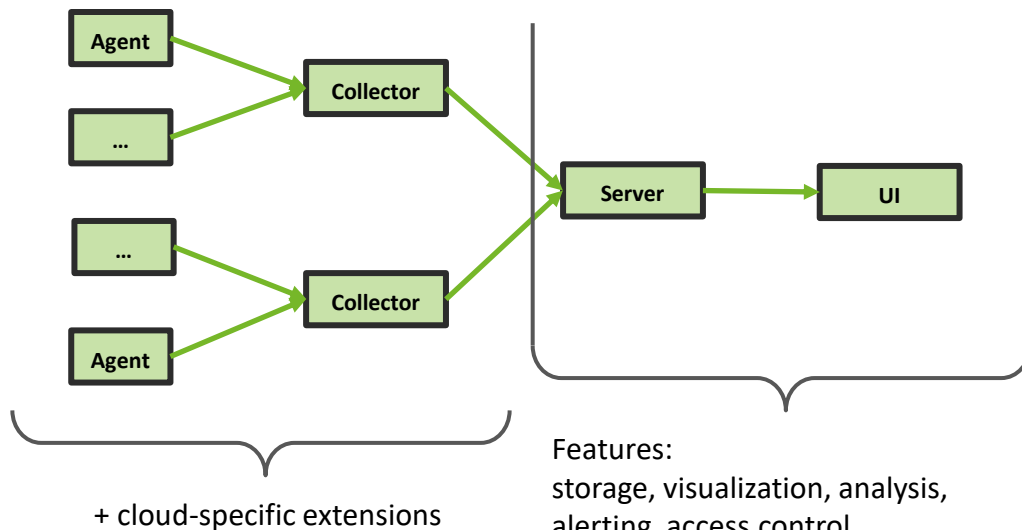


OpenTelemetry: Supported by Cloud Providers

- 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



Managed Trace & Metrics Services

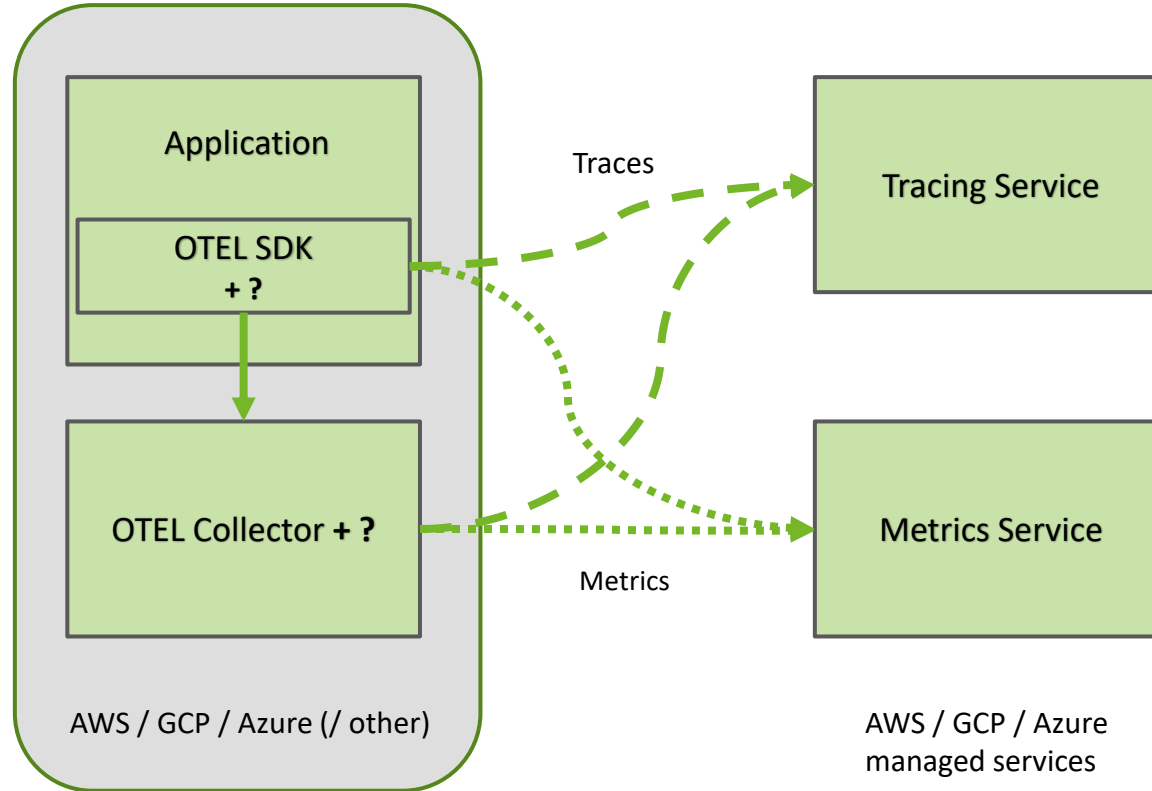
OpenTelemetry support: use managed trace and metrics services through OpenTelemetry + extensions. Normally, **usage of vendor proprietary SDKs** to record traces and metrics is required.

OTEL benefits over vendor proprietary SDKs:

- Java auto-instrumentation
- Vendor neutral OTEL API for (manually) recording traces and metrics

	Trace Service	Metrics Service
AWS	<u>X-Ray</u>	<u>CloudWatch</u>
GCP	<u>Cloud Trace</u> (formerly “Stackdriver“)	<u>Cloud Monitoring Metrics</u> (formerly “Stackdriver“)
Azure	<u>Azure Monitor Application Insights</u>	<u>Azure Monitor Metrics</u> (Not supported through OTEL API, yet)

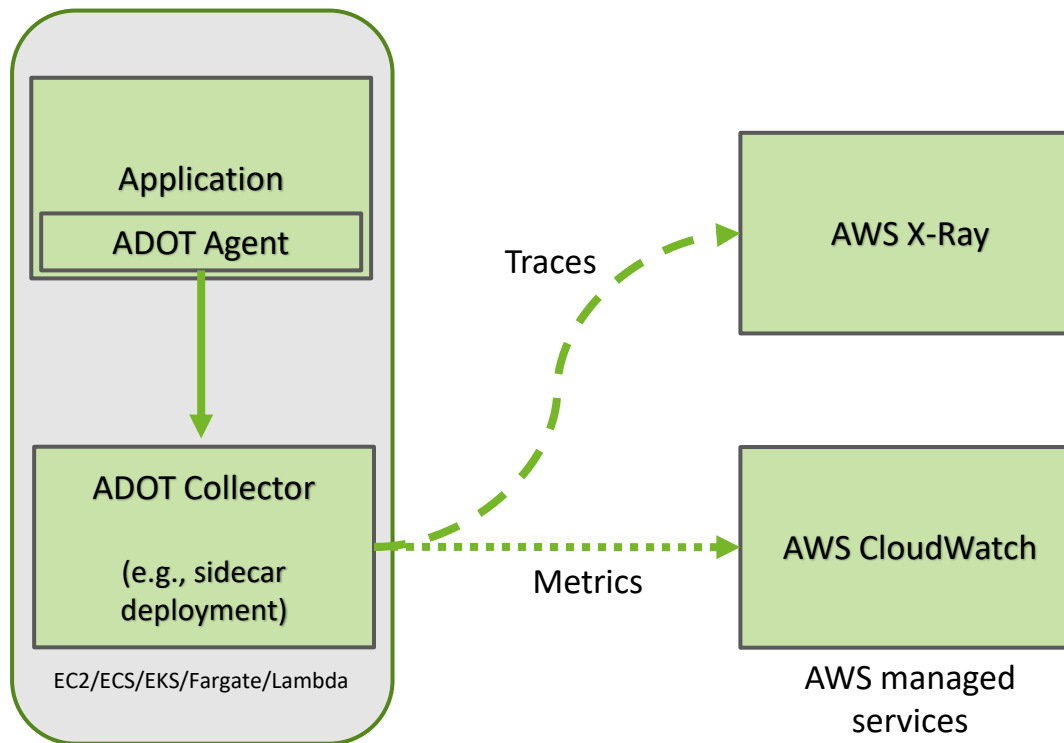
Architectural Blueprint for OTEL integration by Cloud Providers



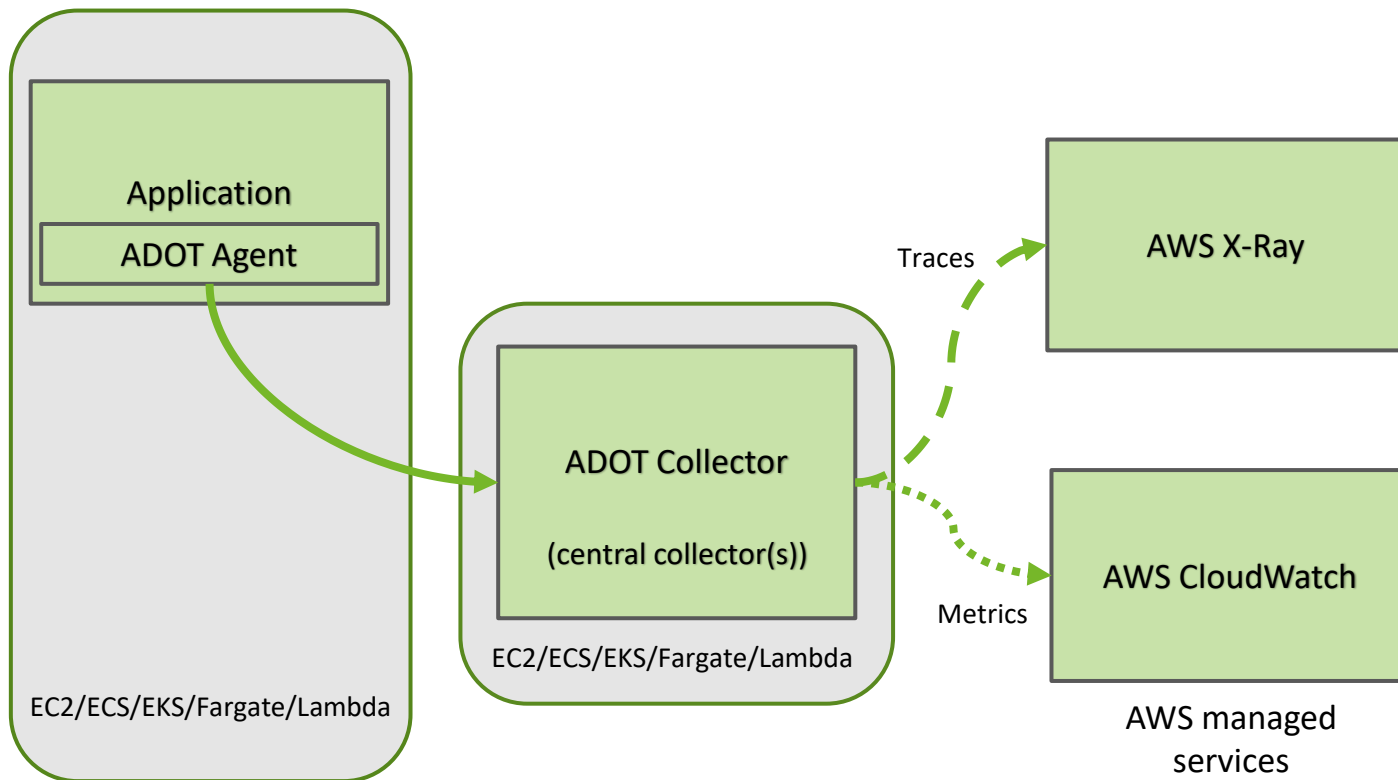
AWS Distribution for OpenTelemetry (ADOT) consists of:

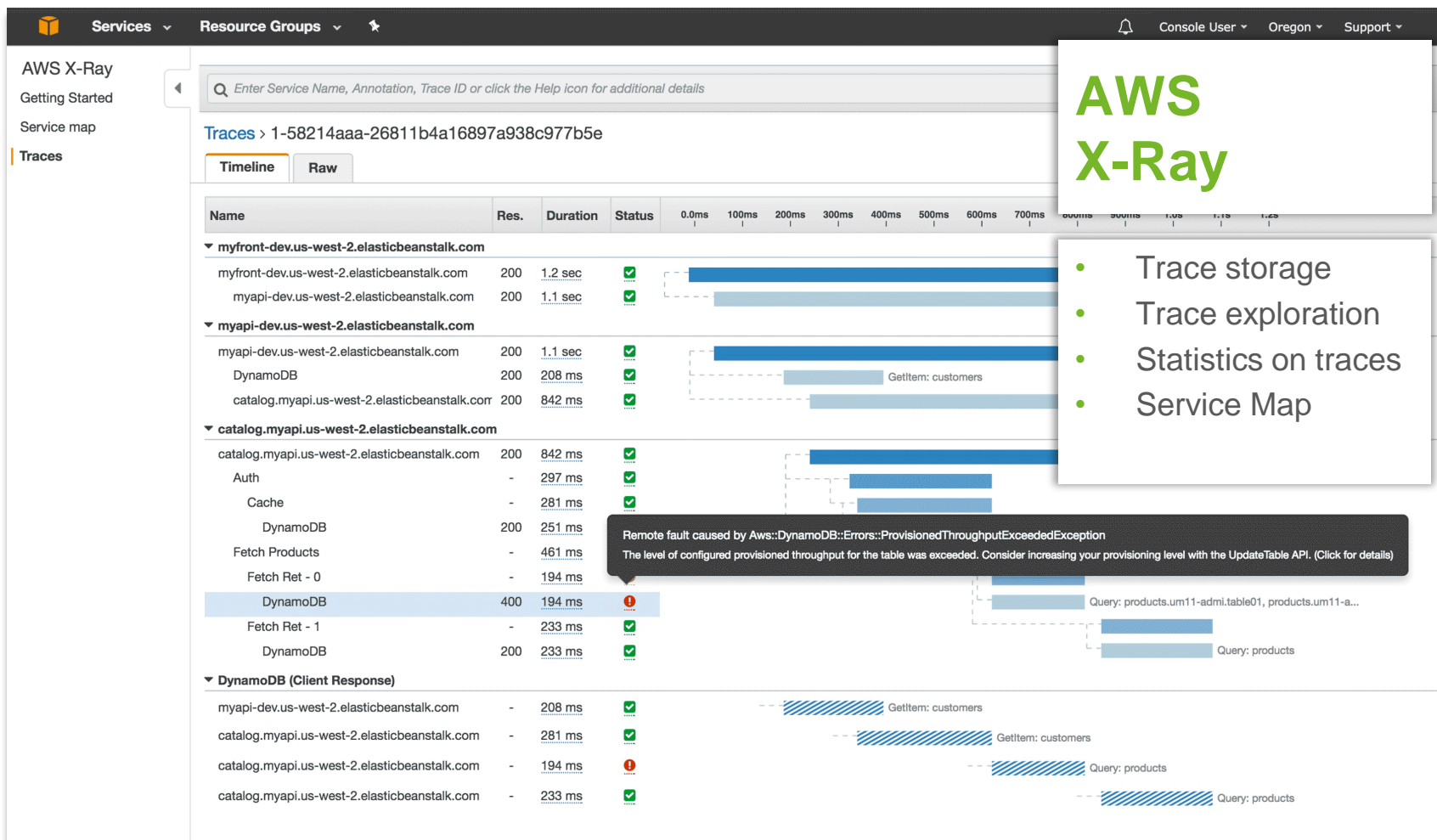
- Java-auto-instrumentation agent redistribution with AWS pre-configuration (ADOT Agent)
<https://github.com/aws-observability/aws-otel-java-instrumentation>
- Collector redistribution with AWS exporters for sending telemetry to X-Ray and CloudWatch (ADOT Collector)
<https://github.com/aws-observability/aws-otel-collector>


AWS OTEL Integration, Collector Sidecar



AWS OTEL Integration, Collector Service






Services

[Alt+S]

CloudWatch

Favorites

Dashboards

Alarms 0 0 0

Logs

Metrics

All metrics

Explorer

Streams

X-Ray traces

Service Map

Traces

Events

Application monitor

Insights

Settings

Getting Started

Untitled graph ✎

1h 3h 12h 1d 3d 1w custom (6m)

1

0.8

0.6

0.4

0.2

0

07/31 08/07 08/14

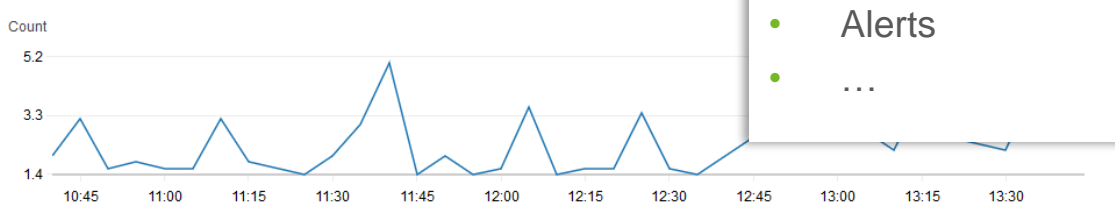
Your CloudWatch graph is empty.
Select some metrics to appear here.

All metrics

Graphed metrics

Math expression ?

Label



Count

5.2

3.3

1.4

10:45 11:00 11:15 11:30 11:45 12:00 12:15 12:30 12:45 13:00 13:15 13:30

NetworkPacketsOut

All metrics Graphed metrics (1) Graph options Source

+ Add a math expression ? Dynamic labels ▼ Statistic: Average Period: 5 Minutes Remove all

<input checked="" type="checkbox"/>	Label	Details	Statistic	Period	Y Axis	Actions
<input checked="" type="checkbox"/>	NetworkPacketsOut	EC2 • NetworkPacketsOut • InstanceId: ...	Average	5 Minutes	◀ > 📈 🔔 📄 ⊕	

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

AWS CloudWatch

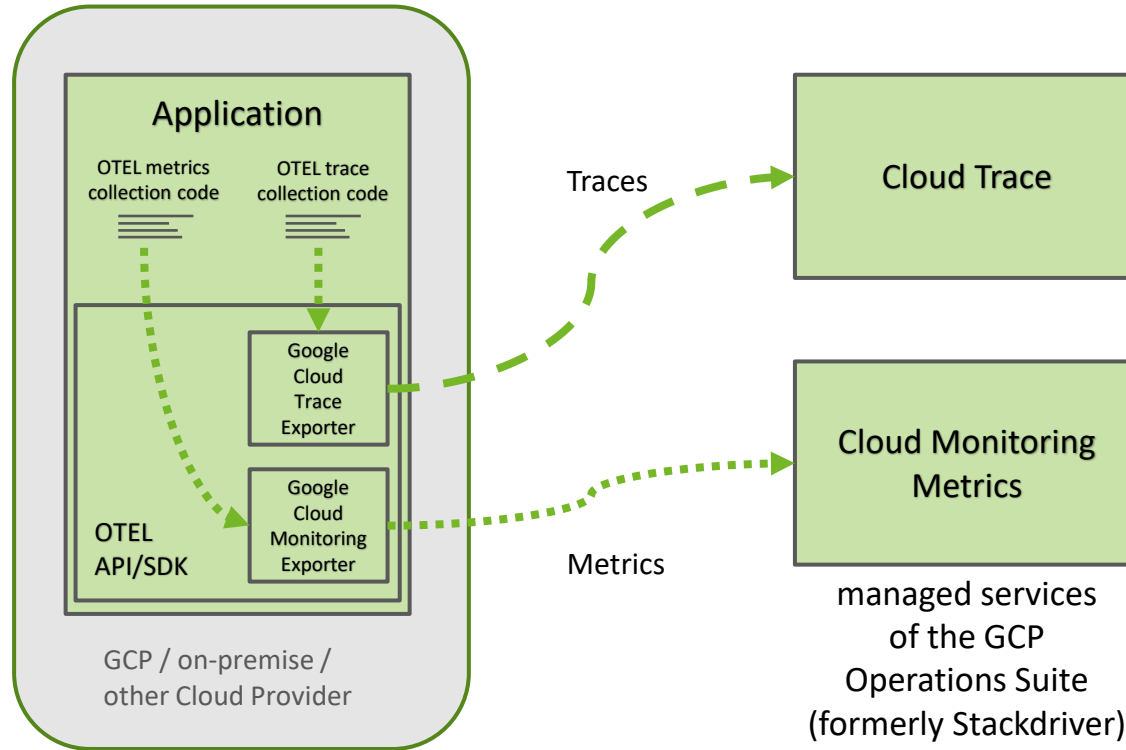
- Metrics storage
- Metrics visualization
- Alerts
- ...

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



- Relies on OTEL API/SDK and code to manually record traces and metrics.
- Transmission of telemetry to GCP via extension of OTEL SDK:
 - <https://github.com/GoogleCloudPlatform/opentelemetry-operations-java/tree/main/exporters/trace>
 - <https://github.com/GoogleCloudPlatform/opentelemetry-operations-java/tree/main/exporters/metrics>
- **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)
 - <https://github.com/GoogleCloudPlatform/opentelemetry-operations-java/tree/main/exporters/auto>

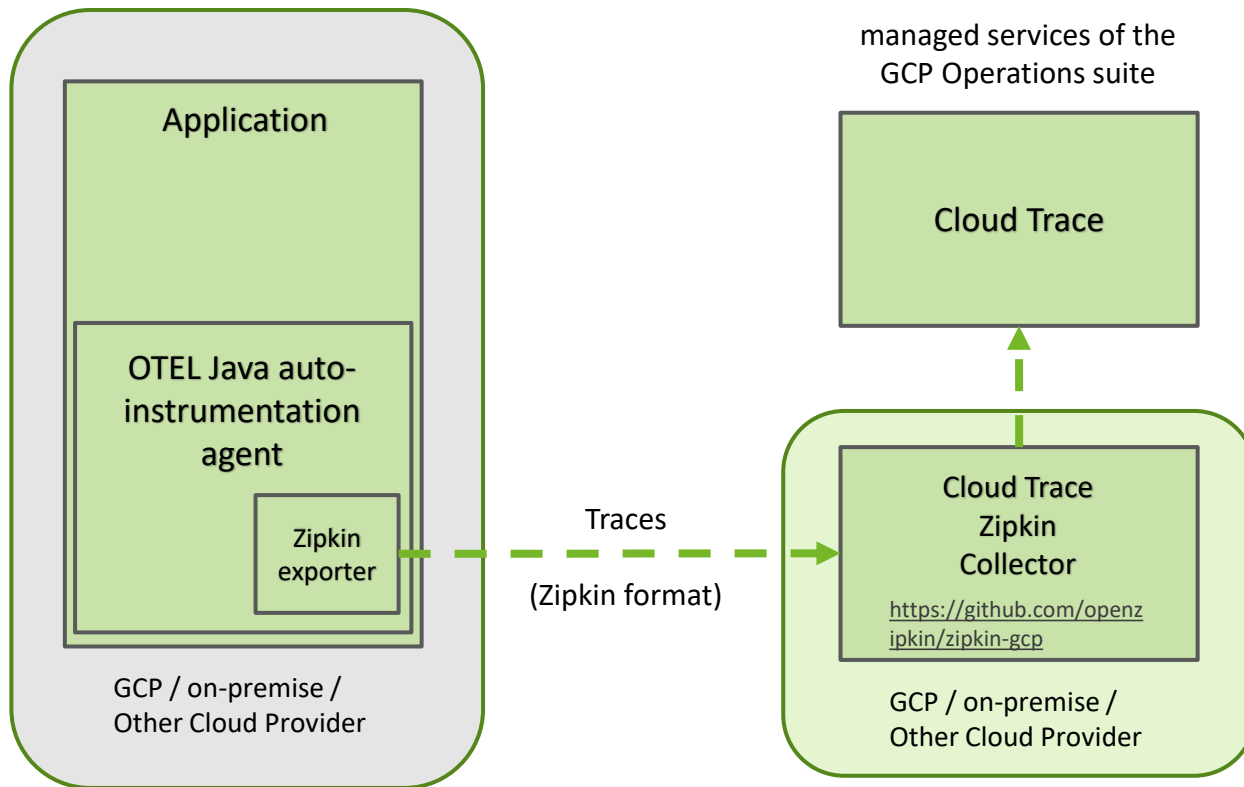
GCP OTEL Integration

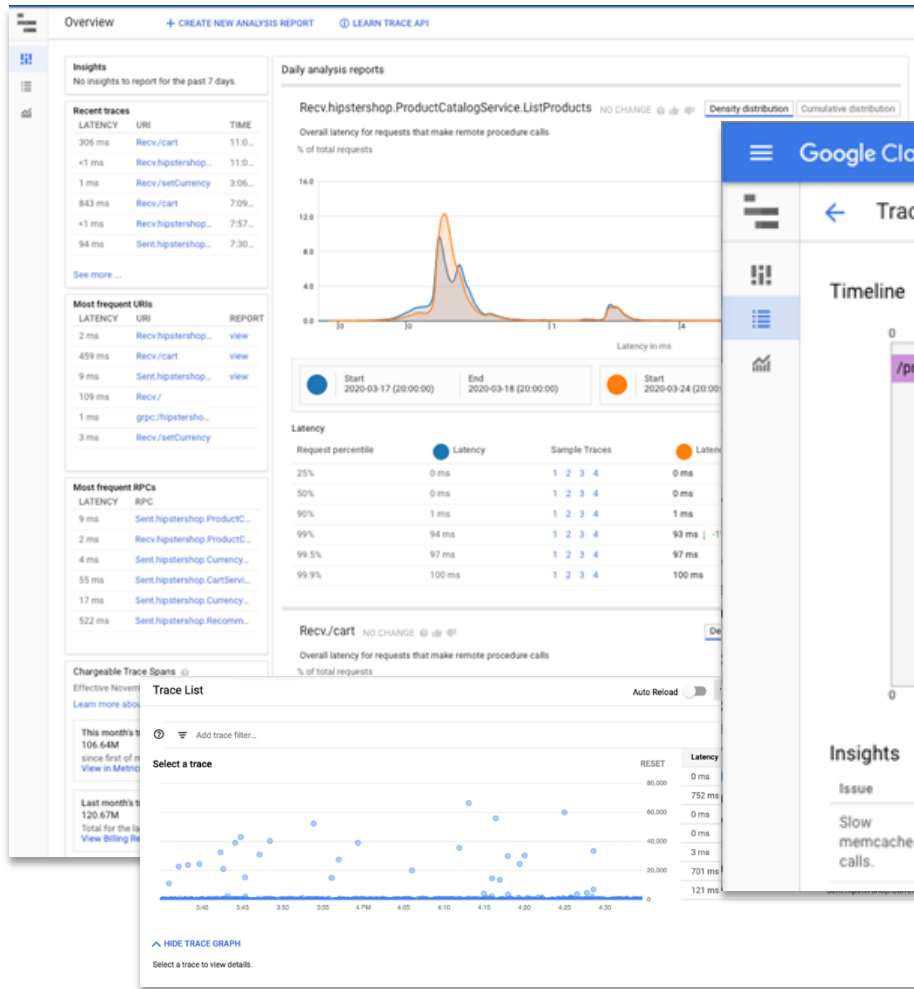


Workaround for recording Traces using Java- Auto-instrumentation in GCP



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





GCP Cloud Trace

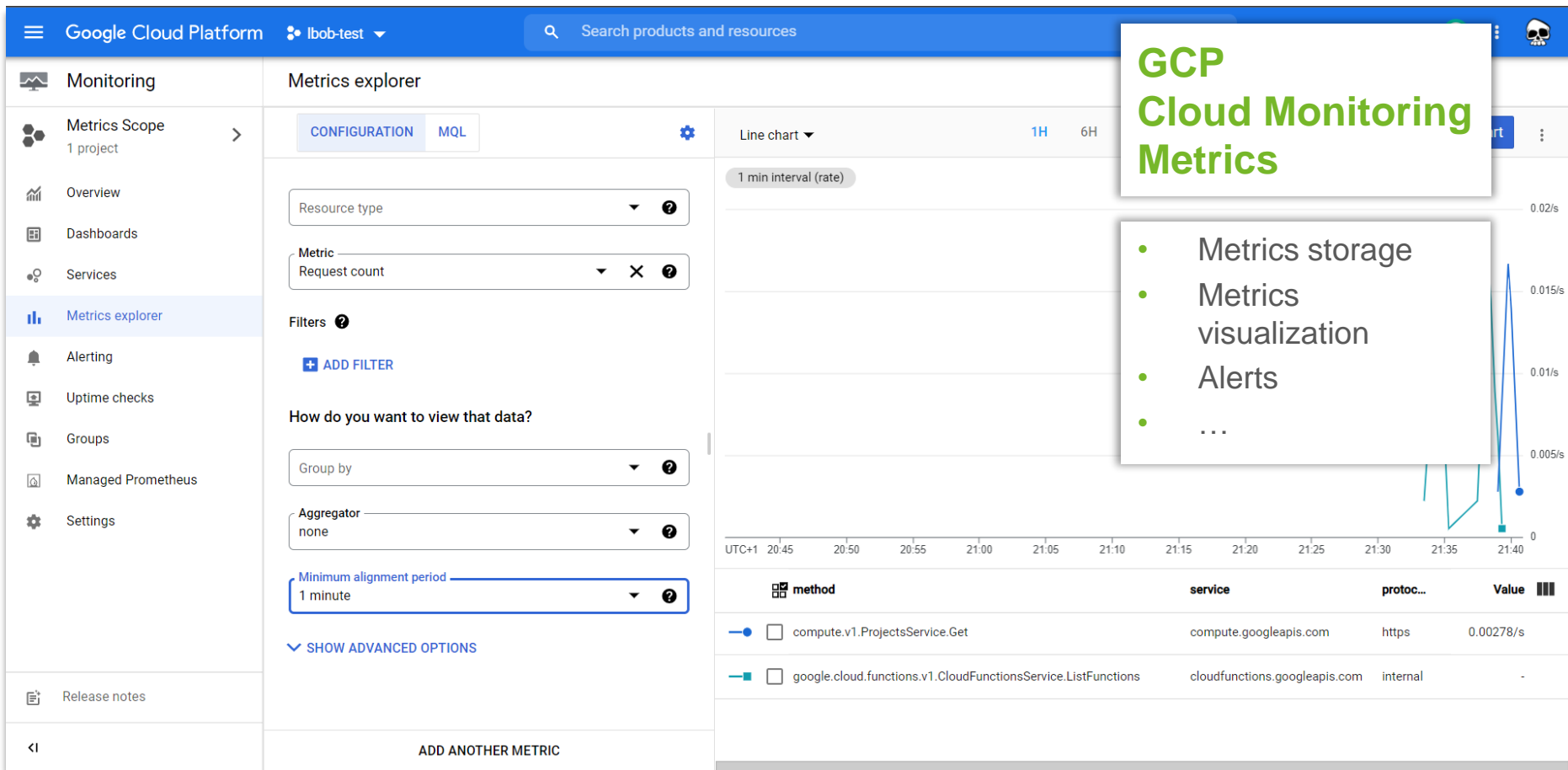
- Trace storage
- Trace visualization
- Statistics on traces
- Suggestions on recognized bottlenecks

Details

Timestamp 2017-02-12 (14:54:00.774)

Insights

Issue	Description	Recommendation
Slow memcache calls.	Your app made 1 remote procedure calls to memcache that took more than 50 ms and the slowest call took 61 ms.	Consider reducing the value of max_concurrent_requests for your app.



GCP Cloud Monitoring Metrics

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

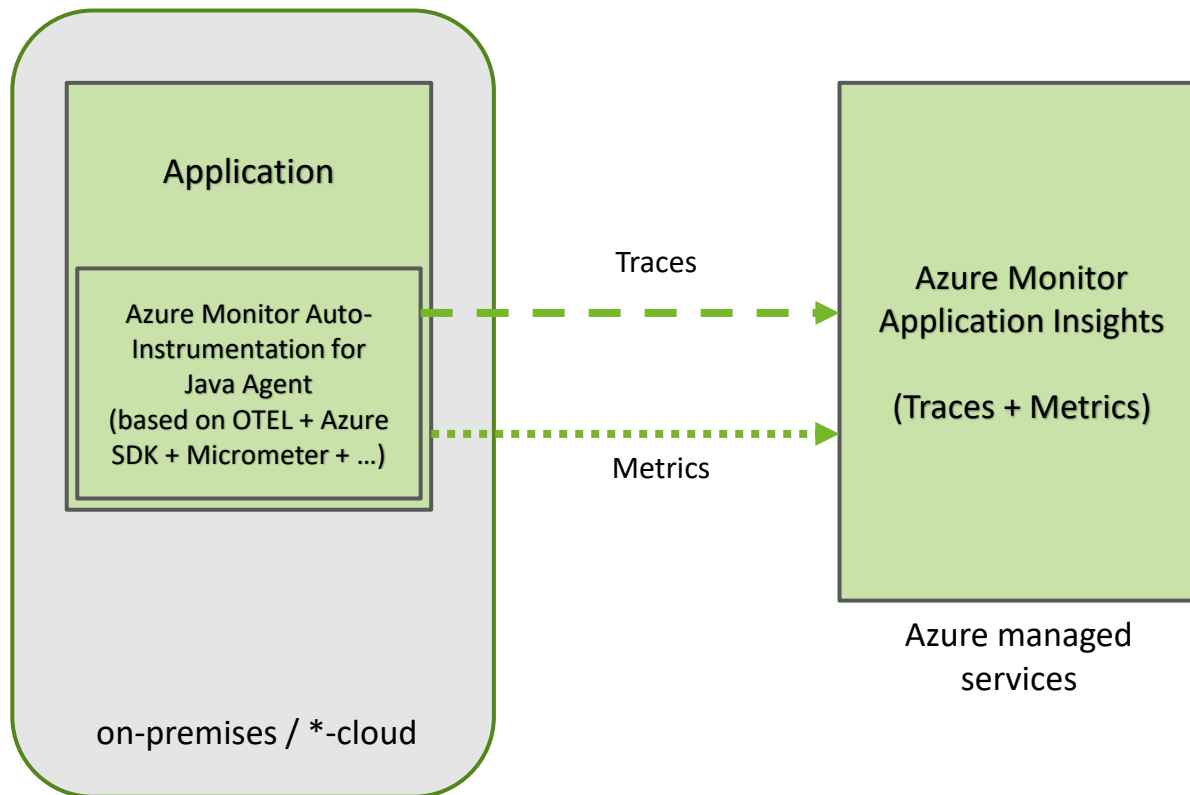



OTEL in Azure




- 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
(<https://docs.microsoft.com/en-us/azure/azure-monitor/app/java-in-process-agent#send-custom-telemetry-by-using-the-2x-sdk>)




Azure OTEL Integration

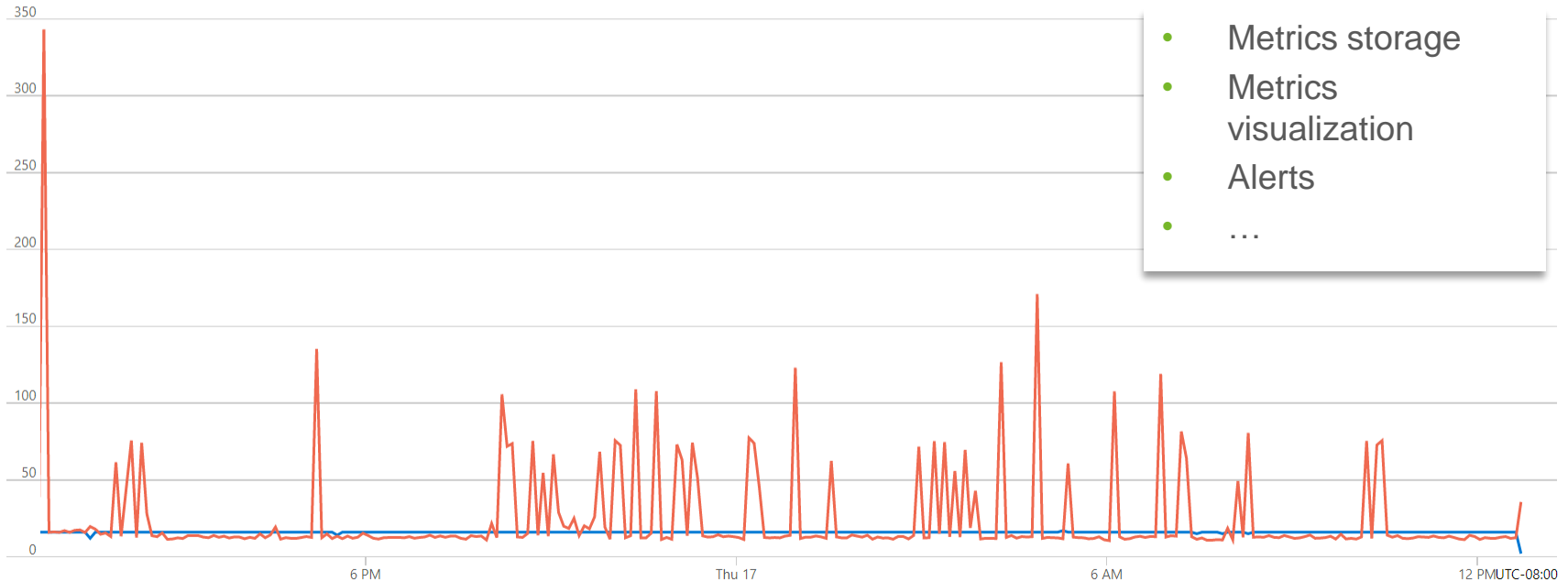


Sum Server requests, Avg Server response time, and Sum Failed requests for tailspintoyswebapp 

 Add metric  Add filter  Apply splitting

 Line chart  Drill in

Azure Monitor Metrics



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

Server requests (Sum)
tailspintoyswebapp
4.59 k

Server response time...
tailspintoyswebapp
24.30 ms

Failed requests (Sum)
tailspintoyswebapp
0

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!)
```

```
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

```
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

Provider (Price Overview Link)	Billed by	Free tier?	cost / unit (/month)	Pricing Calculator
AWS X-Ray, CloudWatch	# traces recorded # traces retrieved # custom metrics, ingested data volume, archived data volume ...	Yes	\$5.00 / 1 Mio. traces recorded, \$0.50 / 1 Mio. traces retrieved, \$0.30 / custom metric, \$0.63 / 1GB ingested, ... (region: EU Frankfurt)	https://calculator.aws/#/createCalculator/xray https://calculator.aws/#/createCalculator/CloudWatch
GCP Cloud Trace, Cloud Monitoring Metrics	# spans recorded, metrics data volume # Monitoring API calls	Yes	\$0.20 / Mio. spans recorded, \$0.2580 / MiB (first 150-100k MiB, ...)	https://cloud.google.com/products/calculator
Azure Azure Monitor	ingested data volume ("As-You-Go" or Commitment-tiered) # custom metrics, # metrics queries ...	Yes	\$2.99 / GB (Pay-As-You-Go) \$0.258/MB (first 150-100k MB, ...)	https://azure.microsoft.com/en-us/pricing/calculator/

=> 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



Resource Efficient Technologies & IT Systems