Demo:

OpenTelemetry

Java Auto-Instrumentation

Bernhard Lubomski, 2020/11/24

OpenTelemetry

https://opentelemetry.io
https://github.com/open-telemetry/opentelemetry-java-instrumentation

Demo Based on:

https://medium.com/wwblog/auto-instrumentation-with-opentelemetry-3b096fdd068f
https://github.com/williewheeler/otel-demo

OpenTelemetry

https://opentelemetry.io

- Observability Framework
  - Traces
  - Metrics
  - (Logs – still in incubation)
- Collect telemetry data, forward to analysis tool
- One API and SDK per language
- Licensing: Apache 2.0
- RETIT blog post on OpenCensus & OpenTracing:

  https://www.retit.de/open-application-performance-monitoring-apm-standards-opentracing-and-opencensus-2

OpenTelemetry

Scope of OpenTelemetry

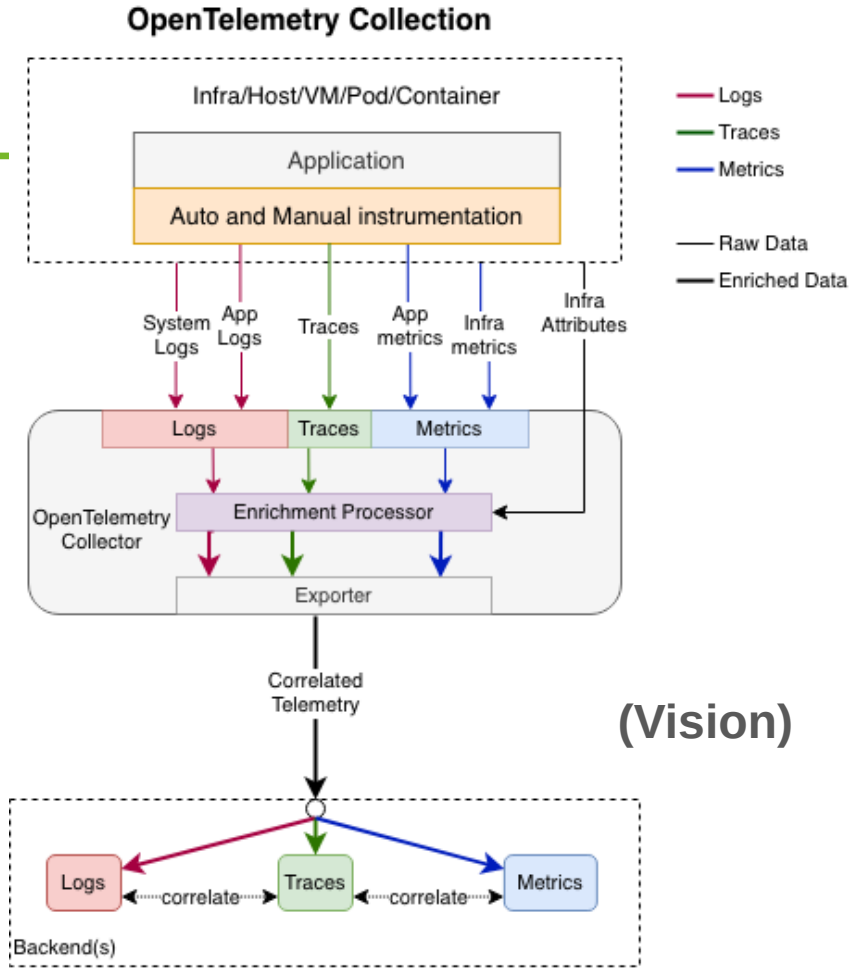
Automatic instrumentation agents to collect telemetry

Optional: manually adding traces/metrics/logs

Language specific SDK: Provides exporters to send traces and metrics to supported backends

Alternative: OpenTelemetry Collector to collect data from SDK and export to supported backend

Currently ready in Java auto instrumentation: traces, metrics, (logs as part of traces)



Source: https://github.com/open-telemetry/opentelemetry-specification/blob/master/specification/logs/overview.md

Supported libraries and frameworks for auto-instrumentation

Languages:

- Java
- C#
- Go
- JavaScript
- Python
- Rust
- C++
- Erlang/Elixir
- …

Library/Framework	Versions
Akka HTTP	10.0+
Apache HttpAsyncClient	4.0+
Apache HttpClient	2.0+
Armeria	0.99.8+
AWS Lambda	1.0+
AWS SDK	1.11.x and 2.2.0+
Cassandra Driver	3.0+
Couchbase Client	2.0+ (not including 3.x yet)
Dropwizard Views	0.7+
Elasticsearch API	5.0+ (not including 7.x yet)
Elasticsearch REST Client	5.0+
Finatra	2.9+
Geode Client	1.4+
Google HTTP Client	1.19+
Grizzly	2.0+ (disabled by default, see below)
Grizzly Client	1.9+
gRPC	1.5+
Hibernate	3.3+
HttpURLConnection	Java 7+
Hystrix	1.4+
JAX-RS	0.5+
JAX-RS Client	2.0+
JDBC	Java 7+
Jedis	1.4+
Jetty	8.0+
JMS	1.1+
JSP	2.3+
Kafka	0.11+
khttp	0.1+
Kubernetes Client	7.0+
Lettuce	4.0+ (not including 6.x yet)
Log4j 1	1.2+
Log4j 2	2.7+
Logback	1.0+
MongoDB Drivers	3.3+
Netty	3.8+
OkHttp	3.0+
Play	2.3+ (not including 2.8.x yet)
Play WS	1.0+
RabbitMQ Client	2.7+
Ratpack	1.4+
Reactor	3.1+
Rediscala	1.8+
Redisson	3.0+
RMI	Java 7+
RxJava	1.0+
Servlet	2.2+
Spark Web Framework	2.3+
Spring Data	1.8+
Spring Scheduling	3.1+
Spring Web MVC	3.1+
Spring Webflux	5.0+
Spymemcached	2.12+
Twilio	6.6+ (not including 8.x yet)
Vert.x	3.0+
Vert.x RxJava2	3.5+

As of 2020/11/22
https://opentelemetry.io/docs/java/automatic_instrumentation/

Demo Application: “Travel Deals”

Travel Deals

Here are today's deals on flights from Seattle to Las Vegas:

Origin	Destination	Airline	Departing	Returning	Class	Price (USD)
SEA	LAS	Deltoid	2020-12-01T12:10:00.000+00:00	2020-12-09T11:40:00.000+00:00	Basic Economy	$105
SEA	LAS	Unitely	2020-12-01T13:25:00.000+00:00	2020-12-09T18:20:00.000+00:00	Basic Economy	$110
SEA	LAS	Pan-Um	2020-12-01T09:15:00.000+00:00	2020-12-09T06:35:00.000+00:00	Basic Economy	$180
SEA	LAS	OrionAir	2020-12-01T12:10:00.000+00:00	2020-12-09T11:40:00.000+00:00	Basic Economy	$112
SEA	LAS	Southeast	2020-12-01T13:25:00.000+00:00	2020-12-09T18:20:00.000+00:00	Basic Economy	$124
SEA	LAS	JetBlack	2020-12-01T09:15:00.000+00:00	2020-12-09T06:35:00.000+00:00	Basic Economy	$150

Source: Auto-instrumentation with OpenTelemetry, Willie Wheeler

Blog: https://medium.com/wwblog/auto-instrumentation-with-opentelemetry-3b096fdd068f

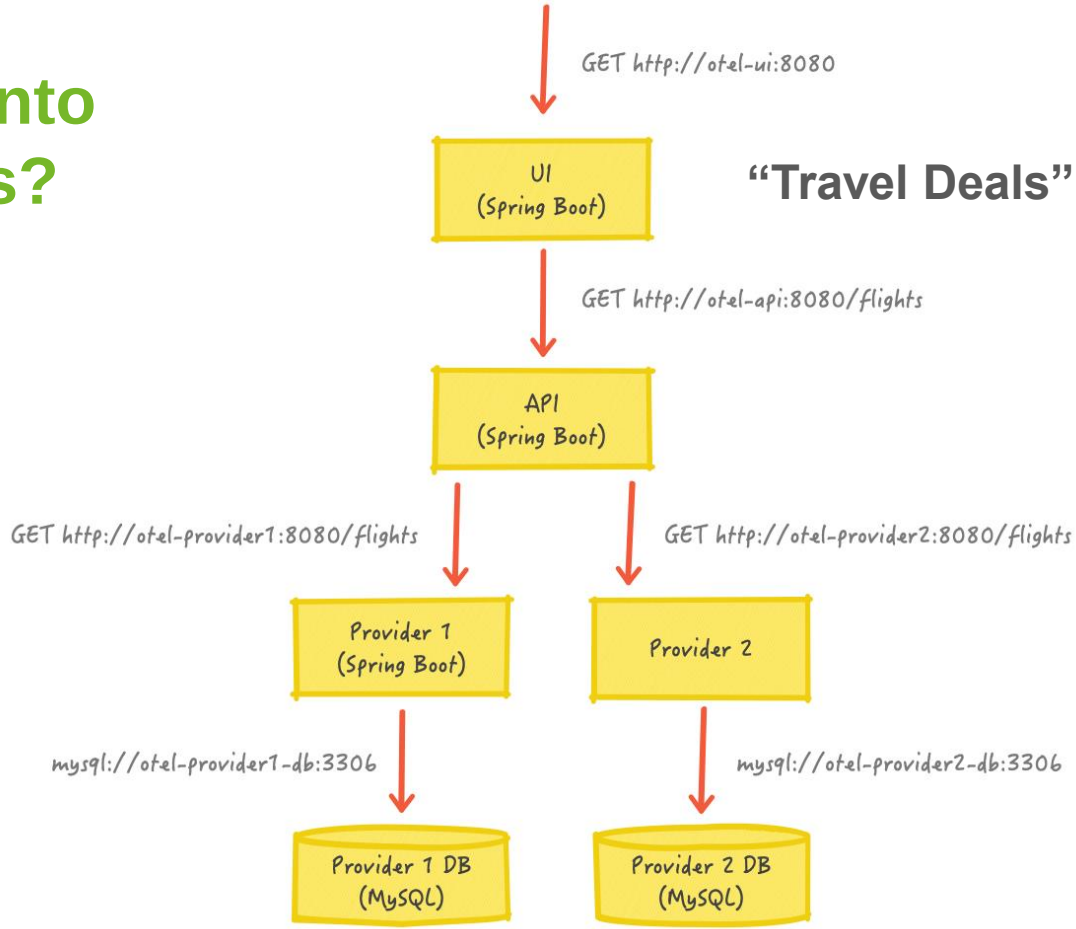
Source: https://github.com/williewheeler/otel-demo

How to gain insight into complex applications?

- Multiple processes / applications / containers
- Different interfaces (REST, JMS, …)
- Database connections

“Travel Deals” demo Application:

- Build tool: Maven
- Build orchestrated using “Make”
- Deployed as Docker containers
- docker-compose setup to launch containers



Source: https://medium.com/wwblog/auto-instrumentation-with-opentelemetry-3b096fdd068f

Travel Deals

Here are today's deals on flights from Seattle to Las Vegas:

Origin	Destination	Airline	Departing	Returning	Class	Price (USD)
SEA	LAS	Deltoid	2020-12-01T12:10:00.000+00:00	2020-12-09T11:40:00.000+00:00	Basic Economy	$105
SEA	LAS	Unitely	2020-12-01T13:25:00.000+00:00	2020-12-09T18:20:00.000+00:00	Basic Economy	$110
SEA	LAS	Pan-Um	2020-12-01T09:15:00.000+00:00	2020-12-09T06:35:00.000+00:00	Basic Economy	$180
SEA	LAS	OrionAir	2020-12-01T12:10:00.000+00:00	2020-12-09T11:40:00.000+00:00	Basic Economy	$112
SEA	LAS	Southeast	2020-12-01T13:25:00.000+00:00	2020-12-09T18:20:00.000+00:00	Basic Economy	$124
SEA	LAS	JetBlack	2020-12-01T09:15:00.000+00:00	2020-12-09T06:35:00.000+00:00	Basic Economy	$150

DEMO

DEMO: Starting the application (no tracing)

Travel Deals

Here are today's deals on flights from Seattle to Las Vegas:

Origin	Destination	Airline	Departing	Returning	Class	Price (USD)
SEA	LAS	Deltoid	2020-12-01T12:10:00.000+00:00	2020-12-09T11:40:00.000+00:00	Basic Economy	$105
SEA	LAS	Unitely	2020-12-01T13:25:00.000+00:00	2020-12-09T18:20:00.000+00:00	Basic Economy	$110
SEA	LAS	Pan-Um	2020-12-01T09:15:00.000+00:00	2020-12-09T06:35:00.000+00:00	Basic Economy	$180
SEA	LAS	OrionAir	2020-12-01T12:10:00.000+00:00	2020-12-09T11:40:00.000+00:00	Basic Economy	$112
SEA	LAS	Southeast	2020-12-01T13:25:00.000+00:00	2020-12-09T18:20:00.000+00:00	Basic Economy	$124
SEA	LAS	JetBlack	2020-12-01T09:15:00.000+00:00	2020-12-09T06:35:00.000+00:00	Basic Economy	$150

1. start using docker-compose up
2. web UI lives at http://localhost:8080

3. How to trace requests to the application?

DEMO: Enable OpenTelemetry instrumentation

```
JAVA_OPTS="
-Dapplication.name=${APP_NAME} \
-Dapplication.home=${APP_HOME} \
-Dotel.exporter=logging \
-javaagent:${APP_HOME}/opentelemetry-javaagent-all.jar"

exec java ${JAVA_OPTS} \
  -jar "${APP_HOME}/${APP_NAME}.jar" \
  --spring.config.location=/config/application.yml
```

"The logging exporter simply prints the name of the span along with its attributes to stdout. It is used mainly for testing and debugging."

Auto-instrumentation configuration reference:
https://opentelemetry.io/docs/java/automatic_instrumentation/

DEMO: How to enable OpenTelemetry trace export

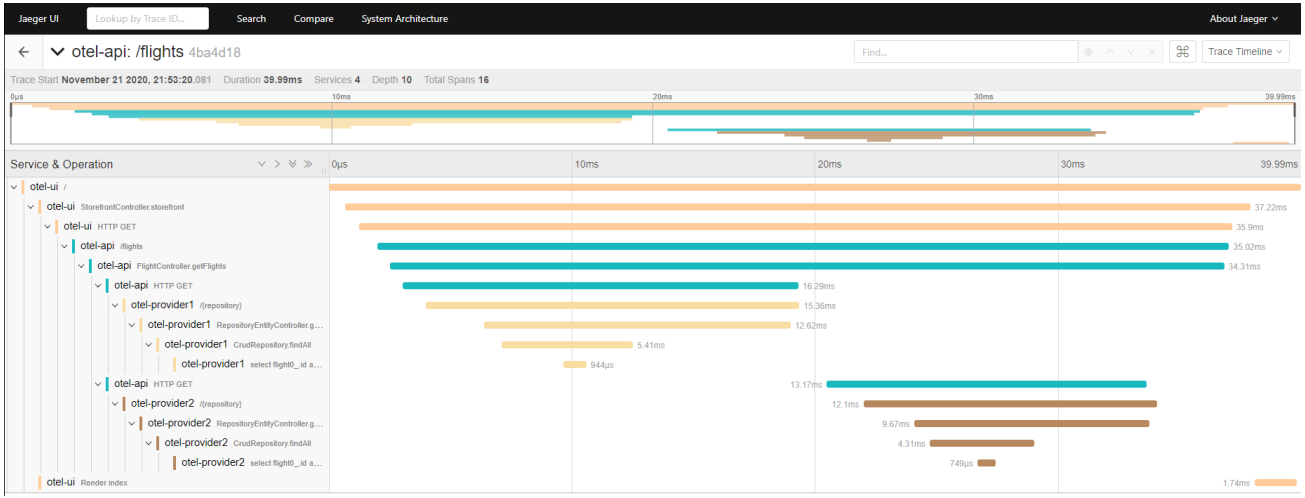
```
JAVA_OPTS="
-Dapplication.name=${APP_NAME} \
-Dapplication.home=${APP_HOME} \
-Dotel.exporter=jaeger \
-Dotel.exporter.jaeger.endpoint=jaeger:14250 \
-Dotel.exporter.jaeger.service.name=otel-ui \
-javaagent:${APP_HOME}/opentelemetry-javaagent-all.jar"
```

Enables Jaeger exporter

Traces sent to Jaeger Collector (host:port - jaeger:14250)

“jaeger.service” => Name of the service as it appears in the traces.

DEMO: launch Jaeger, explore traces



Command to launch a local Jaeger system *:

```
docker run -d --name jaeger -e COLLECTOR_ZIPKIN_HTTP_PORT=9411 -p 5775:5775/udp -p
6831:6831/udp -p 6832:6832/udp -p 5778:5778 -p 16686:16686 -p 14268:14268 -p
9411:9411 jaegertracing/all-in-one:latest
```

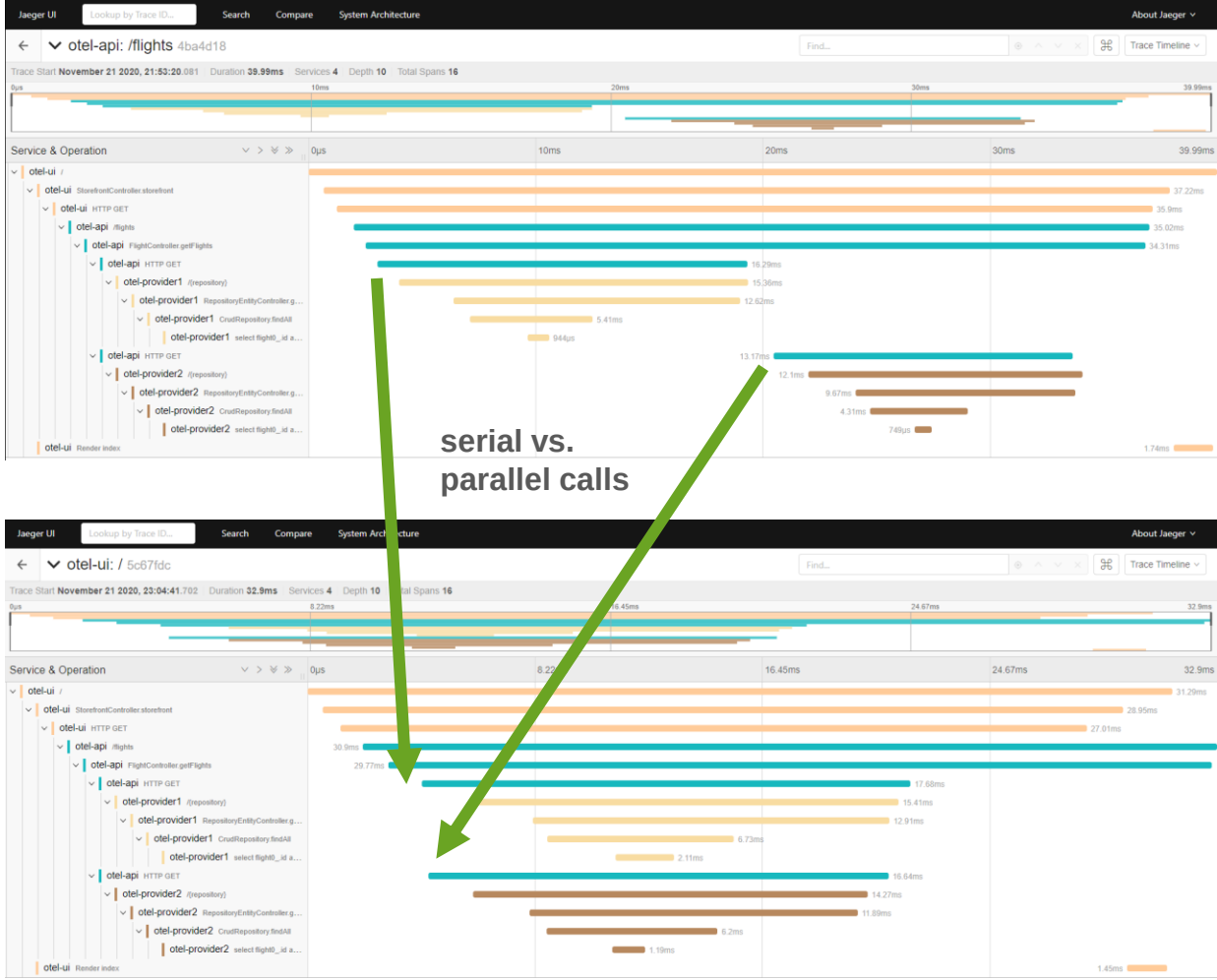
* The demo app starts Jaeger through docker-compose instead.

DEMO:

Discover issues and improve performance

com.williewheeler.demos.otel.api.service.
FlightServiceImpl.getFlights()

Switch between serial and parallel calls leads to the displayed traces.



DEMO: How to enable OpenTelemetry metrics

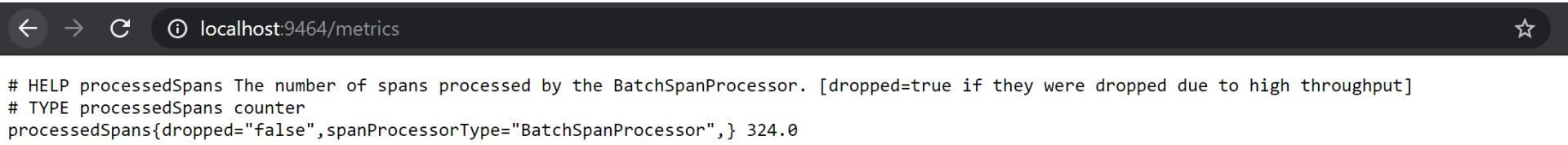
```
JAVA_OPTS="
-Dapplication.name=${APP_NAME} \
-Dapplication.home=${APP_HOME} \
-Dotel.exporter=jaeger,prometheus \
-Dotel.exporter.jaeger.endpoint=jaeger:14250 \
-Dotel.exporter.jaeger.service.name=otel-ui \
-Dotel.exporter.prometheus.port=9464 \
-Dotel.exporter.prometheus.host=0.0.0.0 \
-javaagent:${APP_HOME}/opentelemetry-javaagent-all.jar"
```

Constraint: „Only one metrics exporter is supported“ (but multiple trace exporters)

Add a prometheus „exporter“ (it creates an endpoint to query rather than exporting data to an external endpoint)

A Prometheus server will be able to query metrics from: <apphost>:9464

DEMO: Auto instrumentation metrics



*(*tumbleweed*)*

- In June: Need for “auto metrics” is acknowledged, but semantic conventions had yet to be stabilized (gitter discussion).
- In November: Beta v0.10.1 adds support for system metrics (release notes)
  - Requires Oshi (https://github.com/oshi/oshi), MIT License
- Bugs still exist (see issue 1637)
- Manual configuration is needed as well as documentation (see issue 1566)
- Discussion around inclusion is ongoing (see issue 2078)
- Additional metrics must be added programmatically (example)

DEMO: Making metrics visible

Launch Prometheus system, configured with exporters to scrape the instrumented applications' metrics endpoints.

Example prometheus.yml:

```
global:
  scrape_interval: 15s
  scrape_timeout: 10s
  evaluation_interval: 15s
alerting:
  alertmanagers:
    - static_configs:
        - targets: []
      scheme: http
      timeout: 10s
      api_version: v1
scrape_configs:
  - job_name: otel_demo_provider1
    honor_timestamps: true
    scrape_interval: 15s
    scrape_timeout: 10s
    metrics_path: /metrics
    scheme: http
    static_configs:
      - targets:
          - otel-provider1:9464
...
```

Demo docker-compose.yml extract:

```
...

prometheus:
  image: "prom/prometheus:latest"
  ports:
    - "9090:9090"
  volumes:
    - "./prometheus.yml:/etc/prometheus/prometheus.yml"

...
```

DEMO – Changes to the demo code from Github

- Updated opentelemetry-javaagent-all.jar is required
- Here: Beta v 1.10.1 (newer than version included in “Travel Deals” demo)
  - Adds support for multiple exporters
  - Adds support for metrics (Prometheus) exporter

- Adapting config properties to changed keys:
  - otel.jaeger.* => otel.exporter.jaeger.* in “start-app.sh”

- Added Prometheus metrics exporter to “start-app.sh” configs

- Added Prometheus setup to docker-compose

Summary: OpenTelemetry Java auto-instrumentation

- Freely available, Open Source, Apache 2.0 licensing

- Easy to get started – even for dev machines.

- Traces: collecting/export is well developed.
- Metrics: requires manual work to leverage the technical foundation.
- Logs: Not yet available.

- OpenTelementry is continuously being developed and expanded.

Thank you for your attention!

Questions?

Bernhard Lubomski – lubomski@retit.de