FLOWX.AI

PLATFORM DEEP DIVE / Integrations / jaeger-setup-for-microservices

Contents

PLATFORM DEEP DIVE / Integrations / Jaeger setup for microservices

The scope of this document is to present some basic information on how to include Jaeger tracing into a Java based project.

Required dependencies

```
<dependency>
    <groupId>io.jaegertracing</groupId>
    <artifactId>jaeger-client</artifactId>
    <version>1.4.0</version>
</dependency>
<dependency>
    <groupId>io.opentracing.contrib</groupId>
    <artifactId>opentracing-kafka-client</artifactId>
    <version>0.1.13</version>
</dependency>
```

Needed configs

Add Kafka interceptors for Tracing

```
kafka:
    producer:
        properties:
            interceptor:
                classes:
io.opentracing.contrib.kafka.TracingProducerInterceptor

kafka:
    consumer:
        properties:
            interceptor:
                classes:
io.opentracing.contrib.kafka.TracingConsumerInterceptor
```

Extract Jaeger span context from received Kafka message

```
@KafkaListener(topics = "${TOPIC_NAME}")
public void listen(ConsumerRecord<String, String> record) {
    // some code
    SpanContext spanContext =
TracingKafkaUtils.extractSpanContext(record.headers(),
tracer);
    // some other code
}
```

TIP

Use this context to create child spans of it and log events from adapter:

```
Span span =
tracer.buildSpan(JAEGER_SPAN_NAME).asChildOf(spanContext).
```

Send span context with outgoing Kafka messages

```
ProducerRecord<String, Object> producerRecord = new
ProducerRecord<>(responseTopic, responseMessage);

TracingKafkaUtils.inject(span.context(),
producerRecord.headers(), tracer);

kafkaTemplate.send(producerRecord);
```

Was this page helpful?