*1.* *We need to add some dependencies of Opentelemetry and Jaeger and Prometheus:*

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-registry-prometheus</artifactId>

</dependency>

<dependency>

<groupId>io.prometheus</groupId>

<artifactId>simpleclient\_httpserver</artifactId>

</dependency>

<dependency>

<groupId>io.opentracing.contrib</groupId>

<artifactId>opentracing-spring-jaeger-web-starter</artifactId>

<version>3.3.1</version>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-bom</artifactId>

<version>1.28.0</version>

<type>pom</type>

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.jaegertracing</groupId>

<artifactId>jaeger-core</artifactId>

<version>1.8.1</version>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-api</artifactId>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-sdk</artifactId>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-exporter-jaeger</artifactId>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-exporters-jaeger</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-exporter-prometheus</artifactId>

<version>1.28.0-alpha</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-exporter-otlp</artifactId>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-exporters-prometheus</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>io.opentelemetry</groupId>

<artifactId>opentelemetry-semconv</artifactId>

<version>1.28.0-alpha</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-tracing-bridge-otel</artifactId>

</dependency>

<dependency>

<groupId>io.grpc</groupId>

<artifactId>grpc-netty-shaded</artifactId>

<version>1.41.0</version>

</dependency>

2. Create a Class OpenTelemetryConfig and Implement the following configuration

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import io.opentelemetry.api.OpenTelemetry;

import io.opentelemetry.api.common.Attributes;

import io.opentelemetry.api.trace.propagation.W3CTraceContextPropagator;

import io.opentelemetry.context.propagation.ContextPropagators;

import io.opentelemetry.exporter.otlp.metrics.OtlpGrpcMetricExporter;

import io.opentelemetry.exporter.otlp.trace.OtlpGrpcSpanExporter;

import io.opentelemetry.sdk.OpenTelemetrySdk;

import io.opentelemetry.sdk.metrics.SdkMeterProvider;

import io.opentelemetry.sdk.metrics.export.PeriodicMetricReader;

import io.opentelemetry.sdk.resources.Resource;

import io.opentelemetry.sdk.trace.SdkTracerProvider;

import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;

import io.opentelemetry.semconv.resource.attributes.ResourceAttributes;

@Configuration

public class OpenTelemetryConfig {

@Bean

public OpenTelemetry openTelemetry() {

Resource resource = Resource.getDefault()

.merge(Resource.create(Attributes.of(ResourceAttributes.SERVICE\_NAME, "jaegerTraces")));

OtlpGrpcSpanExporter spanExporter = OtlpGrpcSpanExporter.*builder*()

.setEndpoint("http://192.168.1.83:4317/")

.build();

SdkTracerProvider sdkTracerProvider = SdkTracerProvider.builder()

.addSpanProcessor(BatchSpanProcessor.builder(spanExporter).build())

.setResource(resource)

.build();

OpenTelemetry openTelemetry = OpenTelemetrySdk.builder()

.setTracerProvider(sdkTracerProvider)

.setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInstance()))

.buildAndRegisterGlobal();

return openTelemetry;

}

3. Create a Class MyServiceImpl and Implement the following configuration

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.web.bind.annotation.RestController;

import com.Elastic8SpringBoot3.service.UserService;

import io.opentelemetry.api.trace.Tracer;

@RestController

@Service

public class MyServiceImpl{

private final Tracer tracer;

@Autowired

public UserService userService;

public MyServiceImpl(Tracer tracer) {

this.tracer = tracer;

}

}

4. Implement Changes in the Main Function

ConfigurableApplicationContext context = SpringApplication.run(ElasticwithSpringBoot3Application.class, args);

MyServiceImpl tracer = context.getBean(MyServiceImpl.class);

5. Create a Class UserController and Implement following Configuration on the Class

@RestController

@RequestMapping("/apidata")

6. Implement a function for demo purpose in the User Controller

@GetMapping("/SayHello")

public String sayHello() {

var span = tracer.spanBuilder("hellospan").setSpanKind(SpanKind.SERVER).startSpan();

try{

requestCounter.add(1);

return "Hello ";

}

catch(Exception e)

{

errorCounter.add(1);

throw e;

}

finally {

span.end();

}

}