Observability Assignment

Table of Contents

Overview

- Observability allows engineers to measure and monitor the behavior and performance of a system. This often involves collecting and analyzing data from several sources, including application, infrastructure, and network components.
- With the increasing complexity of modern, distributed systems, it's critical to have insights into every part of the system so that detecting, diagnosing, and resolving issues can happen as quickly as possible. These insights can also help engineers optimize performance and make their system components run more efficiently.
- OpenTelemetry is an open-source Observability framework for instrumenting, generating, collecting, and exporting telemetry data such as traces, metrics, logs.

Instructions

➔ Conduct a thorough review of the OpenTelemetry documentation until you have a good understanding of its purpose and features.

➔ *NOTE: You are free to use any application in a different language if you're comfortable; however, we will be using Javascript for this example. See the documentation on how to proceed for all the other supported languages.*

➔ Before moving forward, make sure you have docker installed on your system as it is one of the prerequisites. Once installed, start running it.

➔ For backend tracing you can use applications like Zipkin and Jaeger. In these instructions we will be using Jaeger, but you can go with Zipkin as well.

➔ Make a public Github repository and start with creating a new Node project.

  ➔ Install all the necessary packages.

```
npm install --save @opentelemetry/api
npm install --save @opentelemetry/sdk-trace-node
npm install --save opentelemetry-instrumentation-express
npm install --save @opentelemetry/instrumentation-mongodb
npm install --save @opentelemetry/instrumentation-http
npm install --save express
npm install --save mongodb@4.3.1
```

➔ Create an index.js file with the code below
(The code below helps set up the Express server that then connects to your local mongo instance. It also uses OpenTelemetry trace to capture tracing data. Here the startServer() initializes the database with some sample data and starts the Express server.)

```
const tracer = require("./tracing")("todo-service");
const express = require("express");
const { MongoClient } = require("mongodb");
const app = express();
app.use(express.json());
const port = 3000;
let db;
const startServer = async () => {
    const client = await MongoClient.connect("mongodb://localhost:27017/");
    db = client.db("todo");
    await db.collection("todos").insertMany([
        { id: "1", title: "Buy groceries" },
        { id: "2", title: "Install Aspecto" },
        { id: "3", title: "buy my own name domain" },
    ]);
    app.listen(port, () => {
        console.log(`Example app listening on port ${port}`);
    });
};
startServer();
app.get("/todo", async (req, res) => {
    const todos = await db.collection("todos").find({}).toArray();
    res.send(todos);
});
app.get("/todo/:id", async (req, res) => {
    const todo = await db.collection("todos").findOne({ id: req.params.id });
    res.send(todo);
});
```

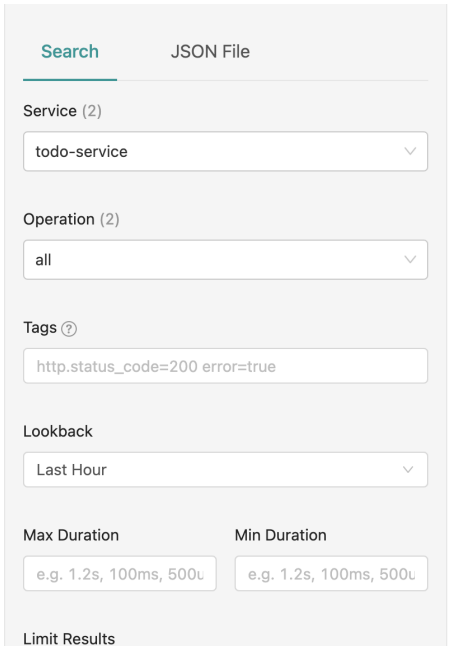
➔ Now create an tracing.js file with the following code
(This sets up the OpenTelemetry tracing for the service.)

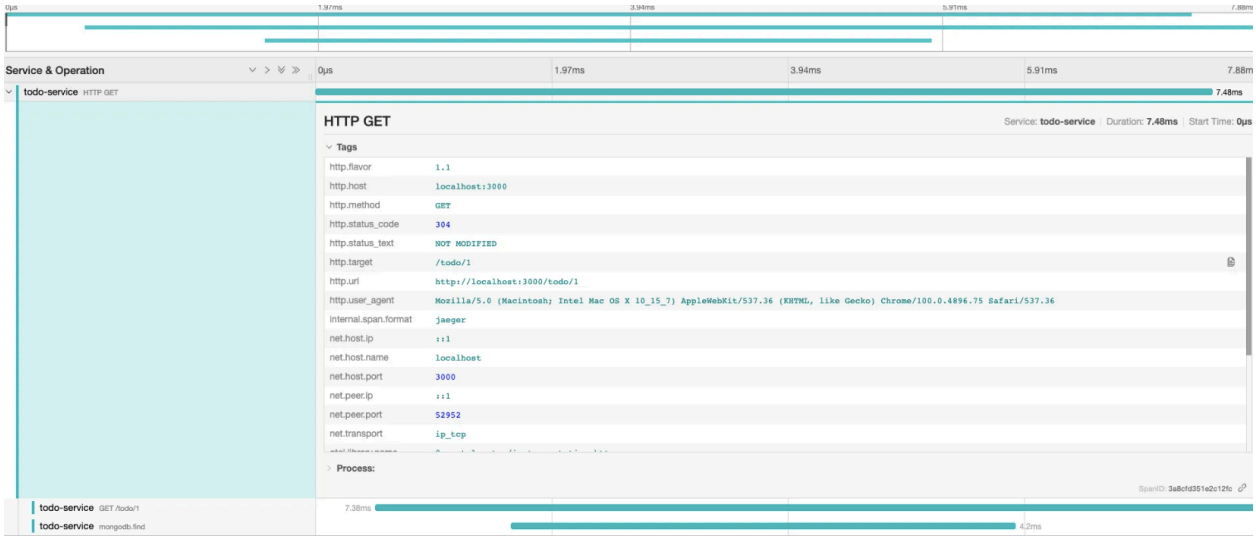
```
const { Resource } = require("@opentelemetry/resources");
const { SemanticResourceAttributes } =
require("@opentelemetry/semantic-conventions");
const { ConsoleSpanExporter } = require('@opentelemetry/sdk-trace-base');
const { SimpleSpanProcessor } = require("@opentelemetry/sdk-trace-base");
const { NodeTracerProvider } = require("@opentelemetry/sdk-trace-node");
const { trace } = require("@opentelemetry/api");
//Instrumentations
const { ExpressInstrumentation } =
require("opentelemetry-instrumentation-express");
const { MongoDBInstrumentation } =
require("@opentelemetry/instrumentation-mongodb");
const { HttpInstrumentation } = require("@opentelemetry/instrumentation-http");
const { registerInstrumentations } = require("@opentelemetry/instrumentation");
//Exporter
module.exports = (serviceName) => {
    const exporter = new ConsoleSpanExporter();
    const provider = new NodeTracerProvider({
        resource: new Resource({
            [SemanticResourceAttributes.SERVICE_NAME]: serviceName,
        }),
    });
    provider.addSpanProcessor(new SimpleSpanProcessor(exporter));
    provider.register();
    registerInstrumentations({
        instrumentations: [
            new HttpInstrumentation(),
            new ExpressInstrumentation(),
            new MongoDBInstrumentation(),
        ],
        tracerProvider: provider,
    });
    return trace.getTracer(serviceName);
};
```

- ➔ Run mongo instance inside Docker container, using the following command
  `docker run -d -p 27017:27017 mongo`
- ➔ Open another terminal or command prompt window and run `node index.js`
- ➔ which will start running the server
- ➔ Run the following command, this command uses the ‘curl’ tool to make HTTP get request to the URL which then displays the logs and spans in the console
  `curl http://localhost:3000/todo`
- ➔ To get started with tracing using Jaeger run it locally using the following command which runs Jaeger all-in-one container in the Docker environment with various port mapping.

```
docker run -d --name jaeger \
  -e COLLECTOR_ZIPKIN_HOST_PORT=:9411 \
  -p 5775:5775/udp \
  -p 6831:6831/udp \
  -p 6832:6832/udp \
  -p 5778:5778 \
  -p 16686:16686 \
  -p 14250:14250 \
  -p 14268:14268 \
  -p 14269:14269 \
  -p 9411:9411 \
  jaegertracing/all-in-one:1.32
```

- ➔ Install the exporter `npm install --save @opentelemetry/exporter-jaeger`
- ➔ Now modify the given tracing.js code to use Jaeger Exporter and then run the code again (use the documentation for help).
- ➔ Open http://localhost:16686/ in your browser which will direct to the Jaeger UI where you can select the to-do list from the search panel. After opening you will be able to see more details.





➔ Take some time to review and understand what information is given in the Jaeger UI. Take a screenshot like the one above.

➔ Analyze the telemetry data and give a summary of your understanding of what the results mean.

Grading (6 points total)

Create and submit a google doc that can be accessed by anybody who has the link. In that document, include the following:

- 1.5 points: Link to your public GitHub with the code for this assignment.

- 1.5 point: Screenshot of your back-end tracing (Jaeger or Zipkin).
- 3 points: Summary of your understanding of the Telemetry data, explaining traces, metrics and logs collected from your application (ideally around 250-350 words)