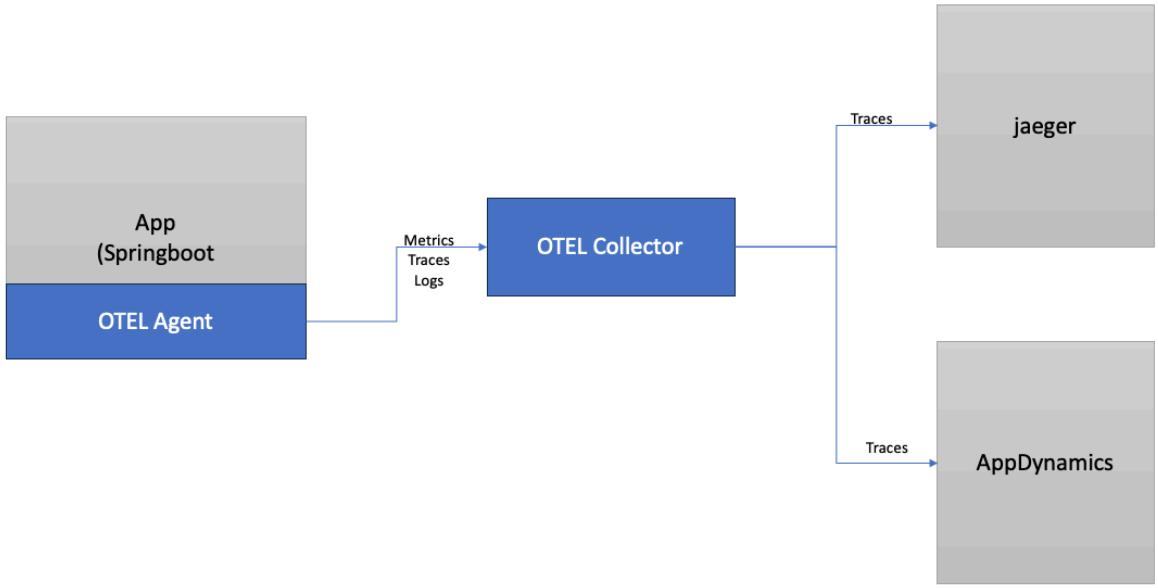
Automatic Instrumentation with OTEL java agent

Architecture



Lab

The first is to download the executable file from the link below and place it in the *lab1_otel_autointrumentation* folder.

https://github.com/lof000/otel-cco-labs/releases/download/v1.0/banking-2.1.0.jar

In this first step we will validate if the app is working without any instrumentation. Run all the commands inside the *lab1_otel_autointrumentation* folder.

Start the app.

```
./run.sh
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.0.6)

2024-02-02T09:58:50.566-03:00  INFO 12217 --- [           main] i.d.a.d.DigisicapisApplication           : Starting DigisicapisApplication v2.1.0.${buildNumber} using Java 17.0.4 with PI
ledeoliv/WORK/LABS/GALICIA/lab1-autointrumentation/banking-2.1.0.jar started by ledeoliv in /Users/ledeoliv/WORK/LABS/GALICIA/lab1-autointrumentation)
2024-02-02T09:58:50.568-03:00  INFO 12217 --- [           main] i.d.a.d.DigisicapisApplication           : No active profile set, falling back to 1 default profile: "default"
2024-02-02T09:58:51.029-03:00  INFO 12217 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2024-02-02T09:58:51.034-03:00  INFO 12217 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-02-02T09:58:51.035-03:00  INFO 12217 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.8]
2024-02-02T09:58:51.084-03:00  INFO 12217 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-02-02T09:58:51.085-03:00  INFO 12217 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 489 ms
2024-02-02T09:58:51.294-03:00  INFO 12217 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with context path ''
2024-02-02T09:58:51.305-03:00  INFO 12217 --- [           main] i.d.a.d.DigisicapisApplication           : Started DigisicapisApplication in 0.948 seconds (process running for 1.217)
```

Check API Response

```
curl http://localhost:8081/banking/pago\?total\=1231\&customerId\=12
```

Expected result.

```
➜  lab1_otel_autointrumentation curl http://localhost:8081/banking/pago\?total\=1231\&customerId\=12
{'transactionId':'kadsbflajkhdfas','status':'OK'}%
➜  lab1_otel_autointrumentation
➜  lab1_otel_autointrumentation
```

Stop App

```
CTRL + C
```

Now let's auto instrument with the OTEL Agent

Download the latest version of the OTEL java agent using the link below and place it in the *lab1_otel_autointrumentation* folder:
https://github.com/open-telemetry/opentelemetry-java-instrumentation/releases/latest/download/opentelemetry-javaagent.jar

In this step we will configure the start script to add the open telemetry java agent. This agent will instrument the application automatically. All the traces will be sent to jaeger for now.

The agent will send the information to the OTEL Collector, and the Collector will forward the traces to Jaeger.

The first thing we need to do is to start the collector and Jager. There is a docker-compose file that makes that for us.

Start the collector.

```
docker-compose up -d
```

```
[+] Running 2/2
 ✔ Container jaeger          Started
 ✔ Container otel-collector  Started
➜  lab1-autointrumentation
```

Check collector logs.

```
docker logs otel-collector
```

```
2024-02-02T12:55:23.412Z    info    pipelines/pipelines.go:106      Receiver started.       {"kind": "receiver", "name": "otlp", "pipeline": "traces"}
2024-02-02T12:55:23.412Z    info    pipelines/pipelines.go:102      Receiver is starting... {"kind": "receiver", "name": "otlp", "pipeline": "metrics"}
2024-02-02T12:55:23.412Z    info    pipelines/pipelines.go:106      Receiver started.       {"kind": "receiver", "name": "otlp", "pipeline": "metrics"}
2024-02-02T12:55:23.412Z    info    pipelines/pipelines.go:102      Receiver is starting... {"kind": "receiver", "name": "otlp", "pipeline": "logs"}
2024-02-02T12:55:23.412Z    info    pipelines/pipelines.go:106      Receiver started.       {"kind": "receiver", "name": "otlp", "pipeline": "logs"}
2024-02-02T12:55:23.412Z    info    service/service.go:106  Everything is ready. Begin running and processing data.
2024-02-02T12:55:24.419Z    info    jaegerexporter@v0.64.0/exporter.go:180  State of the connection with the Jaeger Collector backend   {"kind": "exporter", "data_type": "traces", "name": "jaege
```

Now we will change the agent start script to add the agent. Please replace the string <YOURID>> with your unique ID. This will make it easier to find your application in AppDynamics later.

Edit run.sh

>>> remember to replace <<YOURID>>

```
#ENV
ID=<<YOURID>>

#ENV VARIABLES FOR OTEL
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4318
export
OTEL_RESOURCE_ATTRIBUTES="service.name=banking,service.namespace=bankingDemo$ID"

#APM AGENT VARIABLES
export JAVA_TOOL_OPTIONS="-javaagent:opentelemetry-javaagent.jar"

java -jar banking-2.1.0.jar
```

The OTEL_RESOURCE_ATTRIBUTES environment variable contains mandatory information to configure opentelemetry. OTEL_EXPORTER_OTLP_ENDPOINT by default points to https://localhost:4317

Start the app again

```
./run.sh
```

Check the app logs to see agent message.

```
➜  lab1-autointrumentation ./run.sh
Picked up JAVA_TOOL_OPTIONS: -javaagent:opentelemetry-javaagent.jar
[otel.javaagent 2024-02-02 10:03:31:670 -0300] [main] INFO io.opentelemetry.javaagent.tooling.VersionLogger - opentelemetry-javaagent - version: 2.0.0

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.0.6)

2024-02-02T10:03:33.377-03:00  INFO 12691 --- [           main] i.d.a.d.DigisicapisApplication           : Starting DigisicapisApplication v2.1.0.${buildNum
ledeoliv/WORK/LABS/GALICIA/lab1-autointrumentation/banking-2.1.0.jar started by ledeoliv in /Users/ledeoliv/WORK/LABS/GALICIA/lab1-autointrumentation)
```

Check the first lines in the log. It shows that the opentelemetry agent is active.

Now we are going to put some load in the application.

```
./load.sh
```

If you check the OTEL Collector logs you will see information about traces, metrics and logs being received.

Check collector logs.

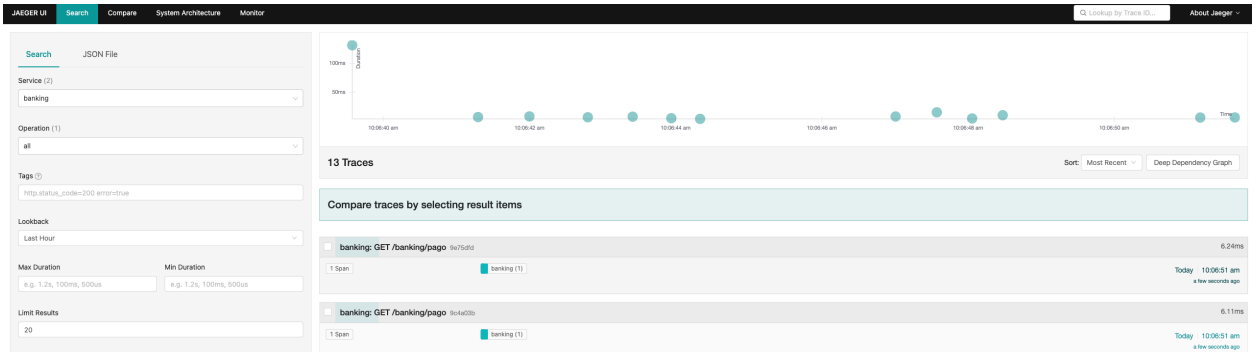
```
docker logs otel-collector
```

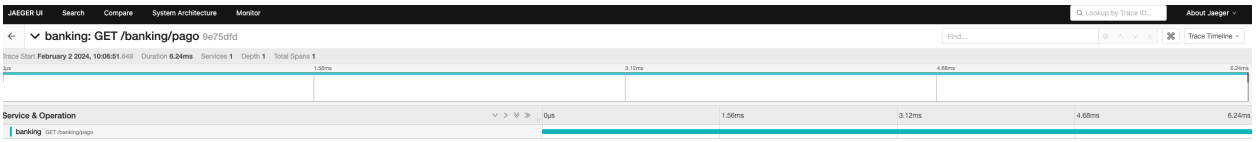
```
ScopeLogs #0
ScopeLogs SchemaURL:
InstrumentationScope io.demo.apis.digisicapis.DigisicapisApplication
LogRecord #0
ObservedTimestamp: 2024-02-02 13:03:34.241262 +0000 UTC
Timestamp: 2024-02-02 13:03:34.241 +0000 UTC
SeverityText: INFO
SeverityNumber: Info(9)
Body: Str(Started DigisicapisApplication in 1.129 seconds (process running for 2.746))
Trace ID:
Span ID:
Flags: 0
ScopeLogs #1
ScopeLogs SchemaURL:
InstrumentationScope org.springframework.boot.web.embedded.tomcat.TomcatWebServer
LogRecord #0
ObservedTimestamp: 2024-02-02 13:03:34.235763 +0000 UTC
Timestamp: 2024-02-02 13:03:34.235 +0000 UTC
SeverityText: INFO
SeverityNumber: Info(9)
Body: Str(Tomcat started on port(s): 8081 (http) with context path '')
Trace ID:
Span ID:
Flags: 0
        {"kind": "exporter", "data_type": "logs", "name": "logging"}
```

Now we will check Jaeger and validate if we can see traces.

Check Jaeger

http://localhost:16686/search





Now let's send the information to jaeger and AppDynamics.

In this last step we will change the OTEL Collector Configuration to send the information to Jaeger and to AppDynamics at the same time. To do that we just need to add a new exporter to the OTEL Collector pipeline.

Stop the app.

```
CTRL + C
```

Stop the collector.

```
docker-compose down
```

Edit otel-collector-config.yaml like this.

```yaml
receivers:
  otlp:
    protocols:
      grpc:
      http:

exporters:
  logging:
    verbosity: detailed
  jaeger:
    endpoint: jaeger:14250
    tls:
      insecure: true

  otlphttp:
    endpoint: https://pdx-sls-agent-api.saas.appdynamics.com
    headers: {"x-api-key": "<<<yourkey>>>"}

processors:
  resource:
    attributes:
      - key: appdynamics.controller.account
        action: upsert
        value: "<<your_appd_account>>"
      - key: appdynamics.controller.host
        action: upsert
        value: ""<<your_appd_controller_host>>"
      - key: appdynamics.controller.port
        action: upsert
        value: 443
  batch:
    timeout: 30s
    send_batch_size: 90
```

```yaml
service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [resource, batch]
      exporters: [logging,jaeger,otlphttp]
    metrics:
      receivers: [otlp]
      exporters: [logging]
    logs:
      receivers: [otlp]
      exporters: [logging]
```

Check that a new session (*otlphttp)* was added. This is the new exporter that will send the information to AppDynamics.

The other new session (processors.resource.attributes), contains the information to connect to the AppDynamics controller. Check that we also changed the service.pipelines.traces session adding the processors and the otlphttp exporter.

run the app again.

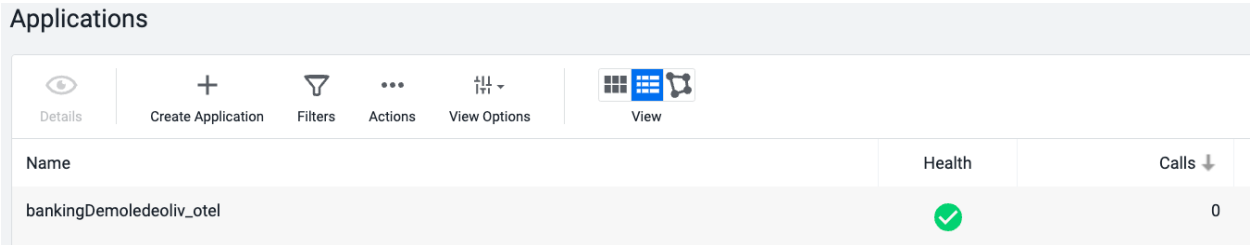
```
./run.sh
```

Put some load.

```
./load.sh
```

Now you can open the controller and find your application there. Check that _otel was append to your application name.

Check AppDynamics



bankingDemoledeoliv_otel

Dashboard | Network Dashboard | Events | Top Business Transactions | **Transaction Snapshots** | Transaction Score

All Snapshots

Details | Filters | Analyze | Actions | Configure

Time	Exe Time (ms)	URL	Business Transaction	Tier
02/02/24 10:26:14 AM	12	-	GET /banking/pago	banking_otel

Check Jaeger