

# Sleuth & Zipkin

Distributed Tracing

# Distributed Tracing

- In a microservices architecture, it can be challenging to trace requests as they propagate through multiple services.
- Distributed tracing helps in tracking the flow of requests and diagnosing performance issues.
- Spring Cloud Sleuth and Zipkin are popular tools for implementing distributed tracing in Spring Boot applications.

# Spring Cloud Sleuth

Spring Cloud Sleuth provides Spring Boot auto-configuration for distributed tracing.

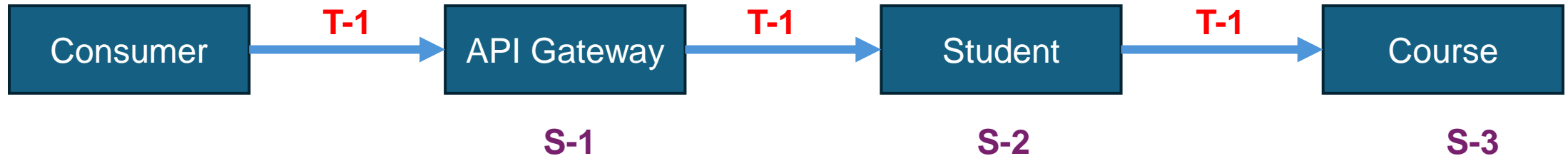
It adds trace and span IDs to the logs, which helps in correlating logs from different services.

## Key Features

- **Automatic Instrumentation:** Sleuth automatically instruments common libraries like Spring MVC, Spring WebFlux, and RestTemplate.
- **Trace and Span IDs:** Adds unique trace and span IDs to logs for easy correlation.
- **Integration with Zipkin:** Sleuth can send trace data to Zipkin for visualization.

# Sleuth – Distributed Tracing Logs

- {Service-name, Trace Id, Span Id}
- Trace Id is unique for a request across all services.
- Span Id is unique for a request within same service.



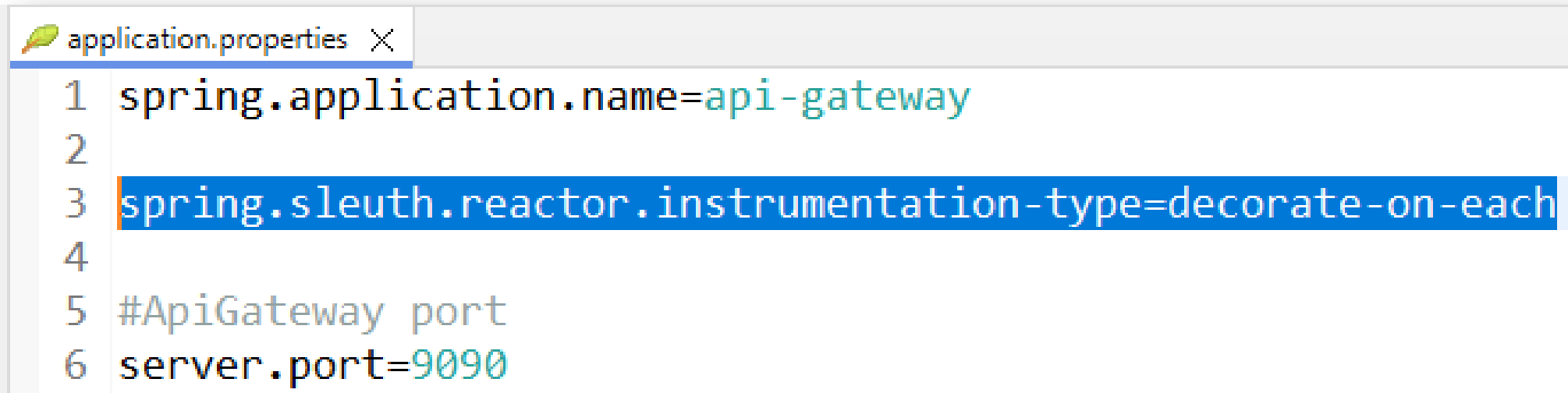
# Add Sleuth Dependency

```
<dependency>  
  <groupId>org.springframework.cloud</groupId>  
  <artifactId>spring-cloud-starter-sleuth</artifactId>  
  <version>3.1.6</version>  
</dependency>
```

- We need add sleuth dependency in
  - API-Gateway
  - Student-Service
  - Course-Service

# In API-Gateway

- Add following property only in api-gateway and not with other services
- This setting is useful for debugging and monitoring, as it allows you to see the tracing context at each stage of the reactive pipeline.



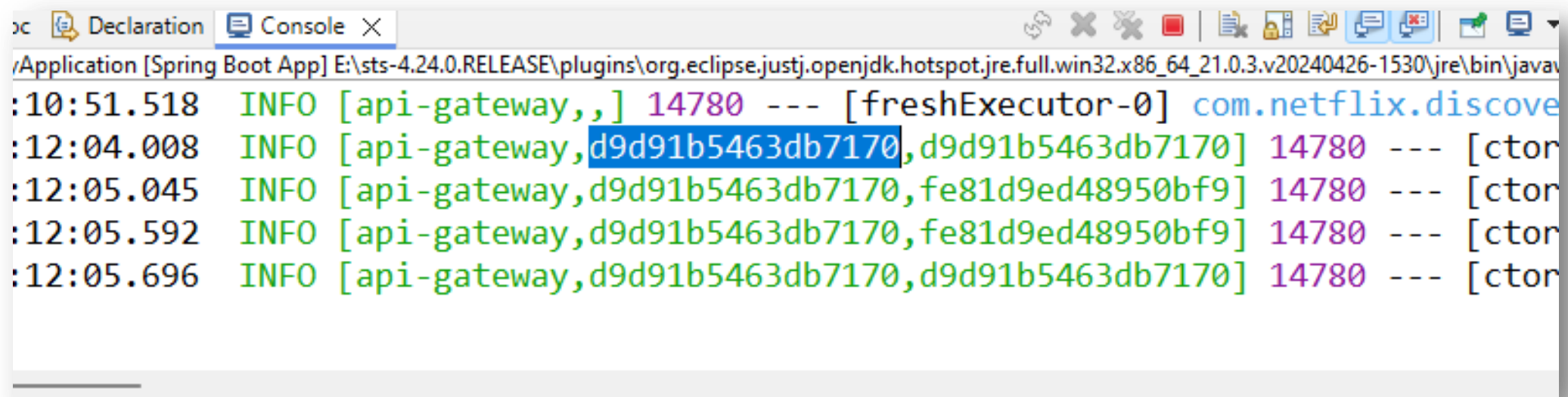
```
application.properties X
1 spring.application.name=api-gateway
2
3 spring.sleuth.reactor.instrumentation-type=decorate-on-each
4
5 #ApiGateway port
6 server.port=9090
```

# Add Loggers in Student and Course

```
public CourseResponse getById(long id) {  
    logger.info("In Course getById : "+id);  
    Course course = courseRepository.findById(id).get();  
}
```

```
public StudentResponse getById(long id) {  
    logger.info("In Student getById : "+id);  
    Student student = studentRepository.findById(id).get();  
}
```

# Check the log for Trace Id & Span Id



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Declaration' and 'Console'. The console output shows several log messages from a Spring Boot application. The messages are color-coded: timestamps in green, log level in green, service name in green, Trace Id in blue, Span Id in green, and other details in purple. The Trace Id '14780' is consistent across all messages. The Span Id 'd9d91b5463db7170' is highlighted in blue in the second message.

```
Application [Spring Boot App] E:\sts-4.24.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw
:10:51.518 INFO [api-gateway,,] 14780 --- [freshExecutor-0] com.netflix.discover
:12:04.008 INFO [api-gateway,d9d91b5463db7170,d9d91b5463db7170] 14780 --- [ctor
:12:05.045 INFO [api-gateway,d9d91b5463db7170,fe81d9ed48950bf9] 14780 --- [ctor
:12:05.592 INFO [api-gateway,d9d91b5463db7170,fe81d9ed48950bf9] 14780 --- [ctor
:12:05.696 INFO [api-gateway,d9d91b5463db7170,d9d91b5463db7170] 14780 --- [ctor
```



# Zipkin

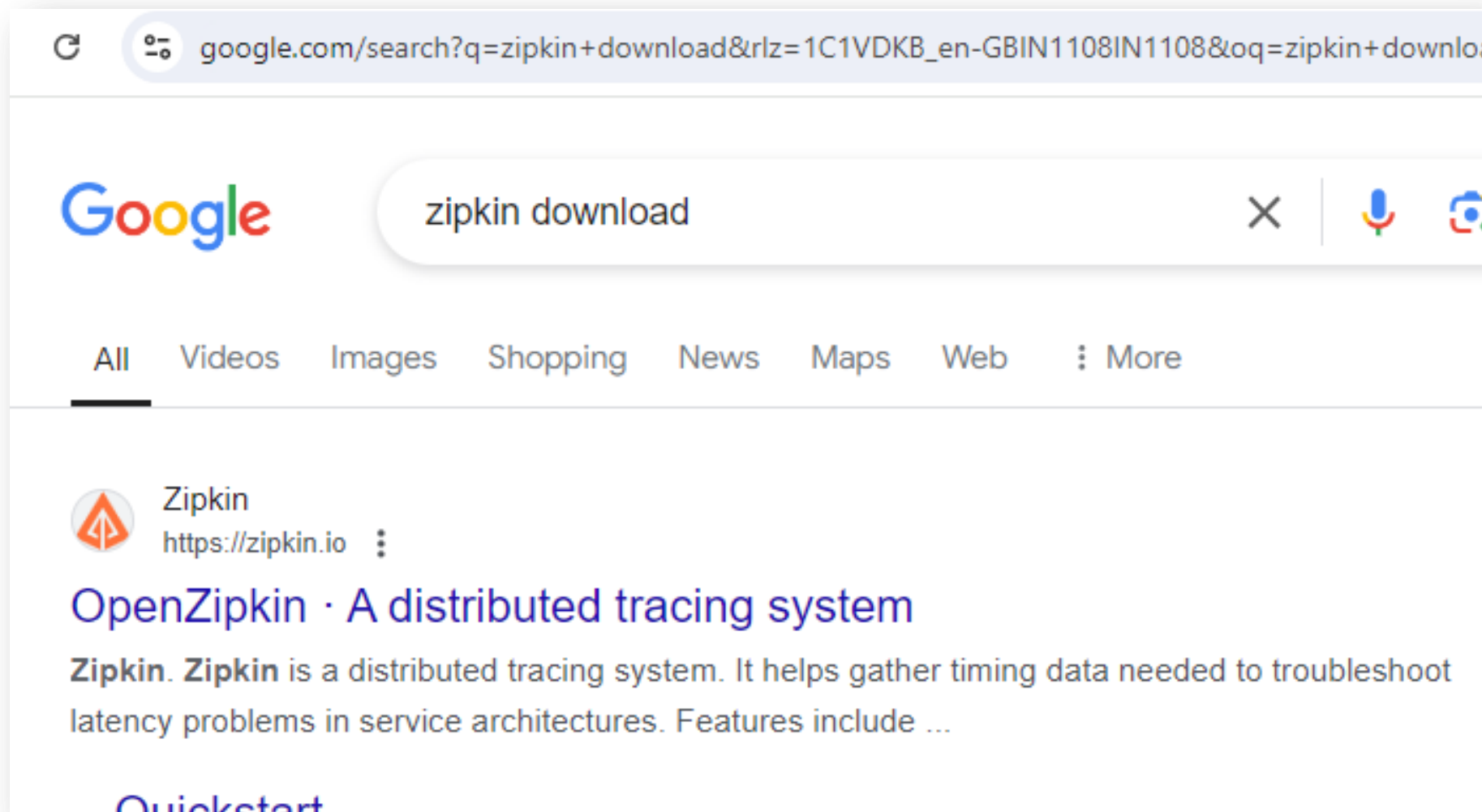
Zipkin is an open-source distributed tracing system.

It helps gather timing data needed to troubleshoot latency problems in service architectures.

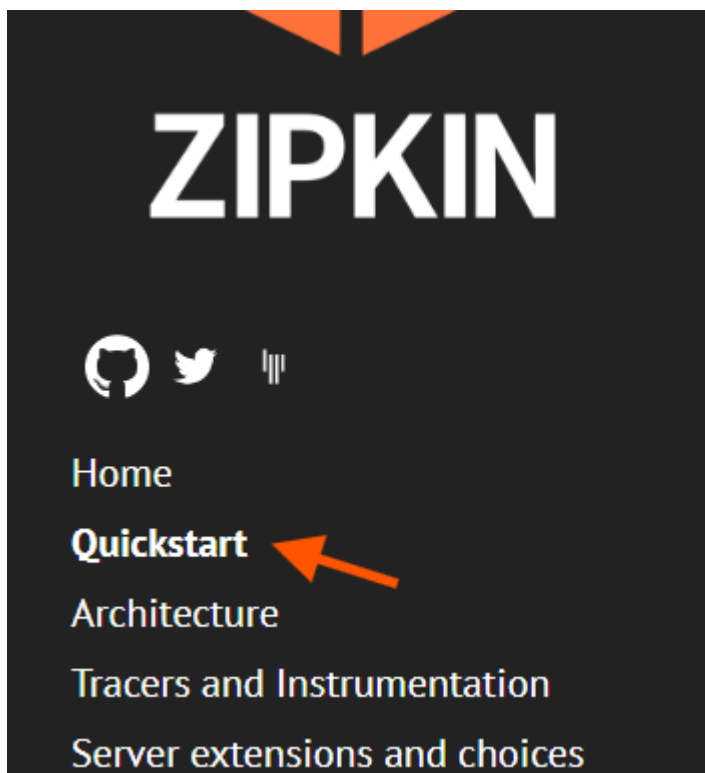
## Key Features

- **Data Collection:** Collects trace data from various services.
- **Storage:** Stores trace data in backend systems like Cassandra, Elasticsearch, or MySQL.
- **Visualization:** Provides a web UI to visualize traces and identify performance bottlenecks.

# Download zipkin jar



# Download & Run Jar file



the latest image directly:

```
docker run -d -p 9411:9411 openzipkin,
```

## Java

If you have Java 17 or higher installed, the **latest release** as a self-contained executable

```
curl -sSL https://zipkin.io/quickstar  
java -jar zipkin.jar
```

## Homebrew

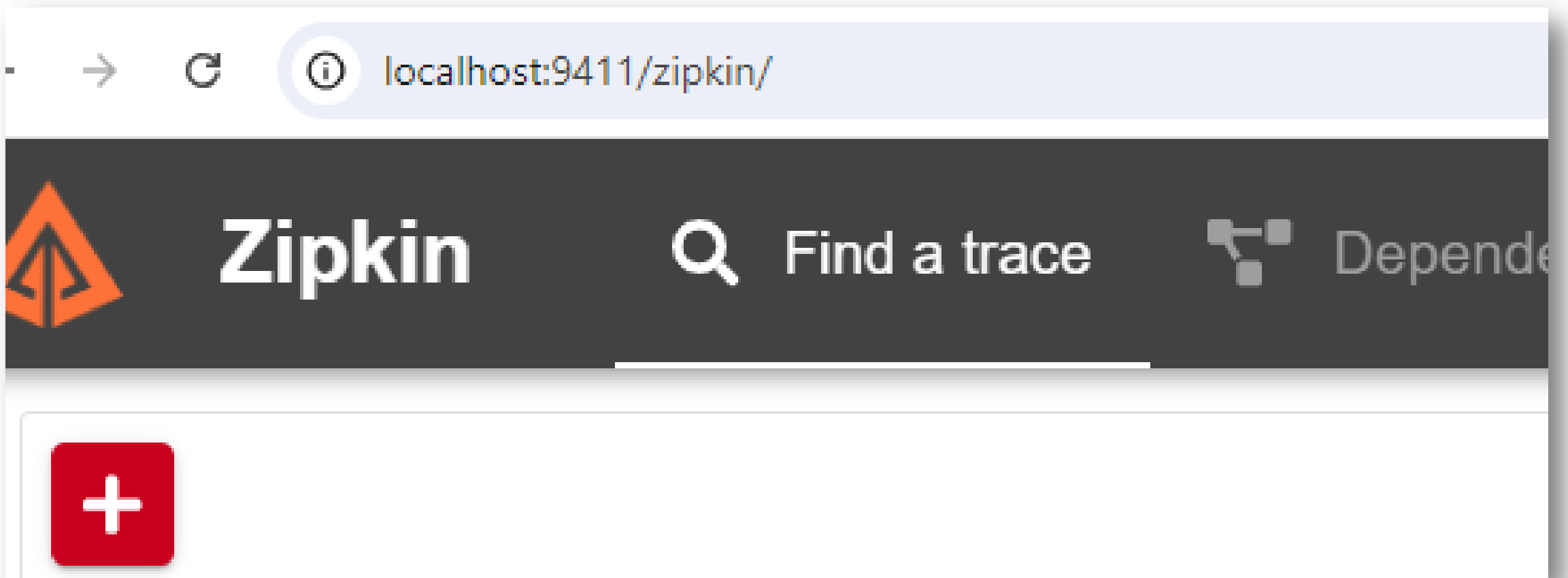
```
C:\Windows\System32\cmd.e X + v
E:\SoftwareBackup>java -jar zipkin-server-3.4.1-exec.jar

      oo
     oooo
    oooooo
   oooooooo
  oooooooooo
 oooooooooooo
oooooooooooooooo
  oooooooo  oooooooo
 oooooooo  oooooooo
ooooooo    oooooooo
ooooooooo  oooooooo
```

This PC > Drive (E:) > SoftwareBackup >

Name	Date modified	Type
springjars	8/15/2024 10:30 PM	File fold
springjars	8/15/2024 10:13 PM	Compre
spring-tool-suite-4-4.24.0.RELEASE-e4.32...	8/8/2024 11:51 AM	Executa
zipkin-server-3.4.1-exec	8/16/2024 3:22 PM	Executa

```
logger to see a stack trace of the call configuring this Meter
2024-08-16T15:28:36.980+05:30 INFO [/] 17592 --- [oss-http-*:
TTP at /[0:0:0:0:0:0:0:0]:9411 - http://127.0.0.1:9411/
```



# Add Zipkin Dependency

```
<dependency>  
    <groupId>org.springframework.cloud</groupId>  
    <artifactId>spring-cloud-starter-zipkin</artifactId>  
</dependency>
```







- We need add zipkin dependency in
  - API-Gateway
  - Student-Service
  - Course-Service




# Add Zipkin url in properties

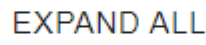
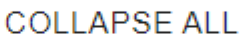

- `spring.zipkin.base-url=http://localhost:9411`
- We need add zipkin base-url in api-gateway & microservices
  - API-Gateway
  - Student-Service
  - Course-Service


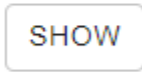
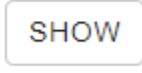
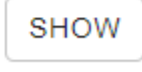
# Simply click RunQuery

localhost:9411/zipkin/?lookback=15m&endTs=1723807871439&limit=10

 Zipkin  Find a trace  Dependencies  Search by trace ID |  EN 

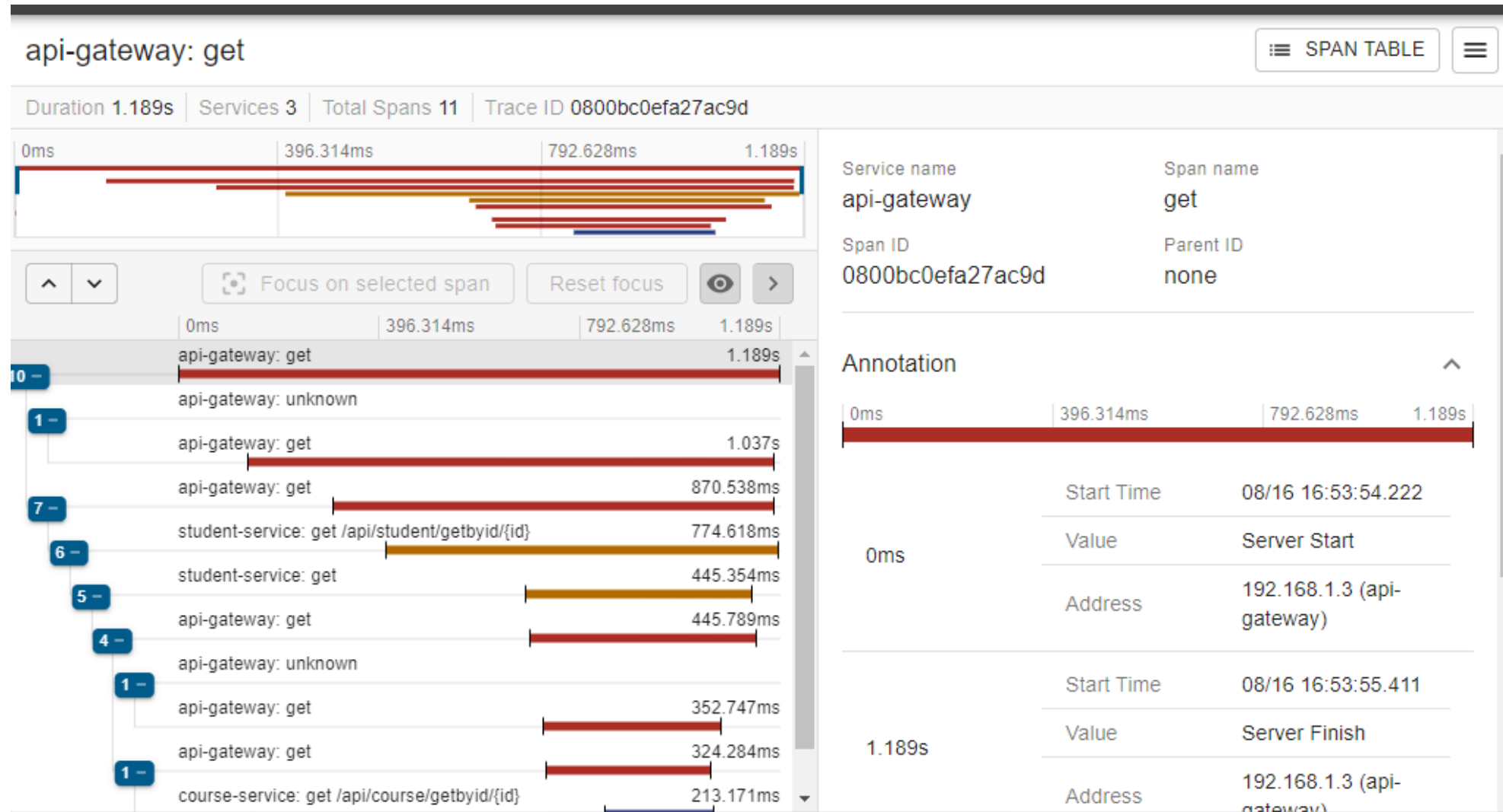
  

Results   Service filters 

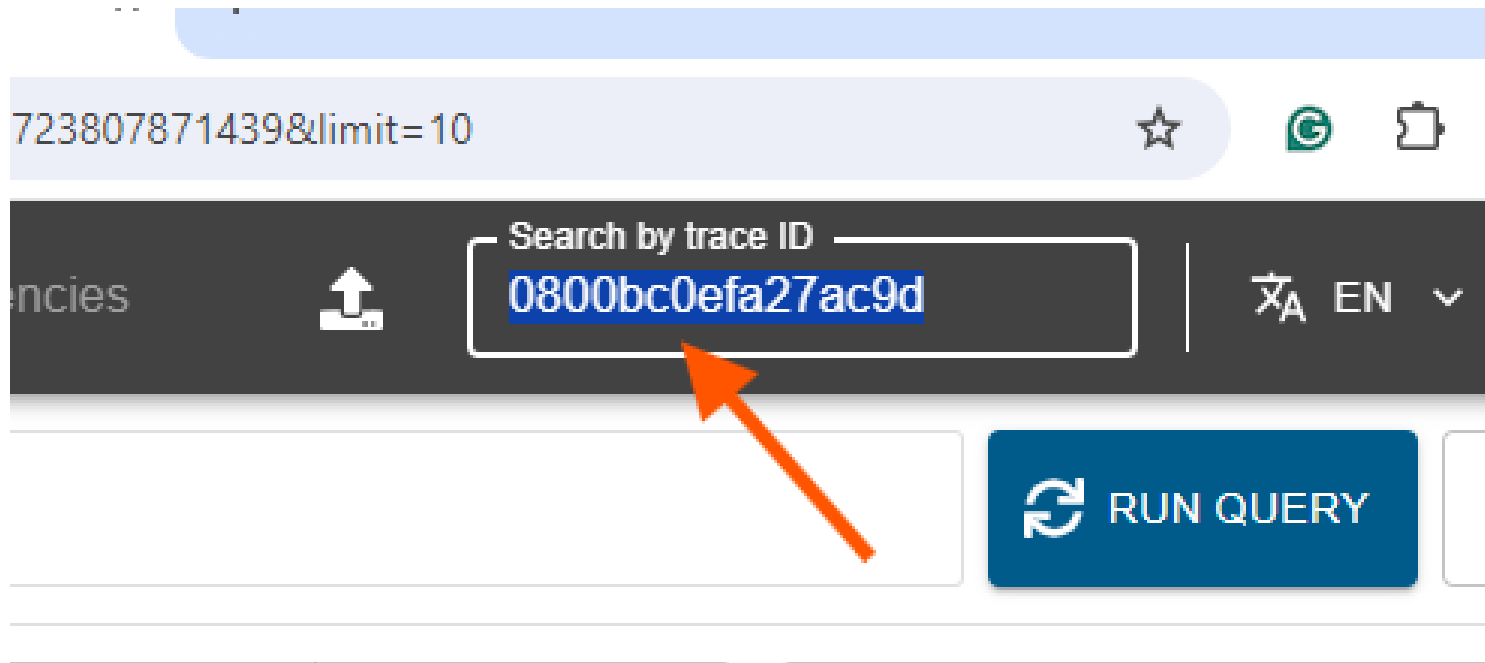
	Root	Start Time	Spans	Duration	
▼	 api-gateway: get	14 minutes ago (08/16 16:46:42:974)	1	21.371s	
▼	api-gateway: get	7 minutes ago (08/16 16:53:54:222)	11	1.189s	
▼	api-gateway: get	7 minutes ago (08/16 16:54:25:069)	9	41.651ms	



# Click Show button for details



# Search by Trace Id



The screenshot shows a web browser window with a search bar. The search bar is labeled "Search by trace ID" and contains the text "0800bc0efa27ac9d". An orange arrow points to the search bar. To the right of the search bar is a "RUN QUERY" button. The browser's address bar shows a URL ending in "&limit=10".

723807871439&limit=10

Search by trace ID

0800bc0efa27ac9d

RUN QUERY

Thank you