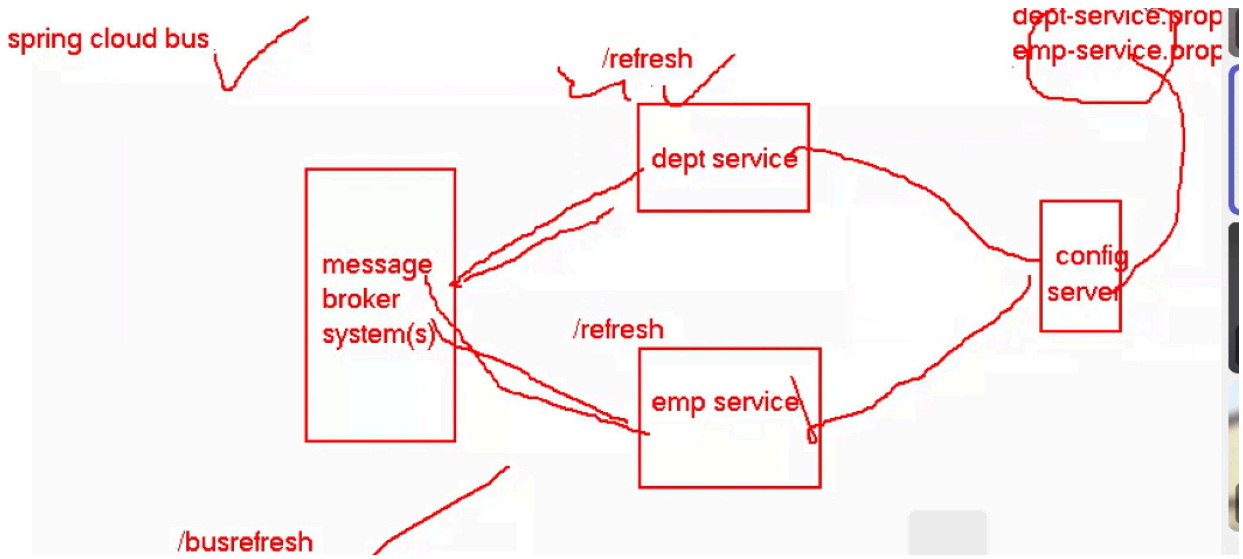
04-03-2025

AutoRefresh configuration using the spring cloud bus

Spring Cloud Bus is a component of Spring Cloud that enables communication between distributed services using a lightweight messaging system. It is primarily used to propagate configuration changes and events across microservices in a system.



Now we need add those dependencies into the pom.xml of employee and department

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-bus-amqp</artifactId> <!-- For RabbitMQ -->
</dependency>
```

This is the development steps

1.Add spring-cloud-starter-bus-amqp dependency to deparrment service and employee service

2.Install RabbitMQ using Docker

3.RabbitMQ configuration in application.properties of department service and employeeservic

4.create simple rest api in employeeservice

5.change department-service and employee-service properties files and call /busrefresh

Search **hub.docker.com**
And search for rabbit mq and copy the tags from it

docker pull rabbitmq:3.13.7-management-alpine

Paste the above command in the cmd

```
C:\Users\miniMiracle>docker pull rabbitmq:3.13.7-management-alpine
3.13.7-management-alpine: Pulling from library/rabbitmq
0e7939ed00bf: Download complete
3419d2ef6b7a: Download complete
17c3ba4c82ff: Download complete
9f90c4b291e2: Download complete
f18232174bc9: Download complete
3563992d595a: Download complete
65cbc02d62f7: Download complete
043e1163cb8d: Download complete
4fa533d28a5a: Download complete
54704e908263: Download complete
Digest: sha256:d759525efd682402f84b84579c4bc7af1f43641e86fb1776e2d7ffea7d42d80a
Status: Downloaded newer image for rabbitmq:3.13.7-management-alpine
docker.io/library/rabbitmq:3.13.7-management-alpine
```

Now check the docker images in cmd

```
C:\Users\miniMiracle>docker images
REPOSITORY   TAG                       IMAGE ID       CREATED        SIZE
rabbitmq     3.13.7-management-alpine   d759525efd68   5 months ago   279MB
```

Now we need to configure the properties in the both department and employee [properties

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
```

Run this in cmd

```
docker run --rm --it -p 5672:5672 rabbitmq:3.13.7-management-alpine
```

```
2025-03-04 06:36:57.914974+00:00 [info] <0.696.0> Server startup complete; 5 plugins st
2025-03-04 06:36:57.914974+00:00 [info] <0.696.0>  * rabbitmq_prometheus
2025-03-04 06:36:57.914974+00:00 [info] <0.696.0>  * rabbitmq_federation
2025-03-04 06:36:57.914974+00:00 [info] <0.696.0>  * rabbitmq_management
2025-03-04 06:36:57.914974+00:00 [info] <0.696.0>  * rabbitmq_management_agent
2025-03-04 06:36:57.914974+00:00 [info] <0.696.0>  * rabbitmq_web_dispatch
2025-03-04 06:36:57.943480+00:00 [info] <0.9.0> Time to start RabbitMQ: 8327 ms
```

Now go to the github make the changes

PavanKalyan96Dev Update employee-service.properties

Code Blame 17 lines (13 loc) · 563 Bytes Code 55% faster with GitHub Copilot

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/wipro
spring.datasource.username=root
spring.datasource.password=\#Mahadev7

# JPA & Hibernate
server.port=8081
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.boot.message=hello, employee service updated spring cloud bus
```

Now go to the employee and create one message controller in controller package

```java
package com.wipro.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RefreshScope
@RestController
public class MessageController
{


    @Value("${spring.boot.message}")
    private String message;

    @GetMapping("/message")
    public String message()
    {
        return message;
    }

}

```

ConfigServer / Department-Service.properties

PavanKalyan96Dev Update Department-Service.properties

Code Blame 16 lines (13 loc) · 612 Bytes Code 55% faster with GitHub Copilot

```
server.port:9848
spring.config.import=optional:configserver:http://localhost:8888

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/wipro
spring.datasource.username=root
spring.datasource.password="#Mahadev7"


spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.boot.message=hello, department service updated spring cloud bus
```

Do this in department service also

Run the service registry
Run the config server
Run the department service

Run the employee service

GET localhost:8081/message Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 124 ms 212 B

Raw Preview Visualize

```
1  hello, employee service updated spring cloud bus
```

Check this in postman

GET localhost:9848/message Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 103 ms 214 B

Raw Preview Visualize

```
1  hello, department service updated spring cloud bus
```

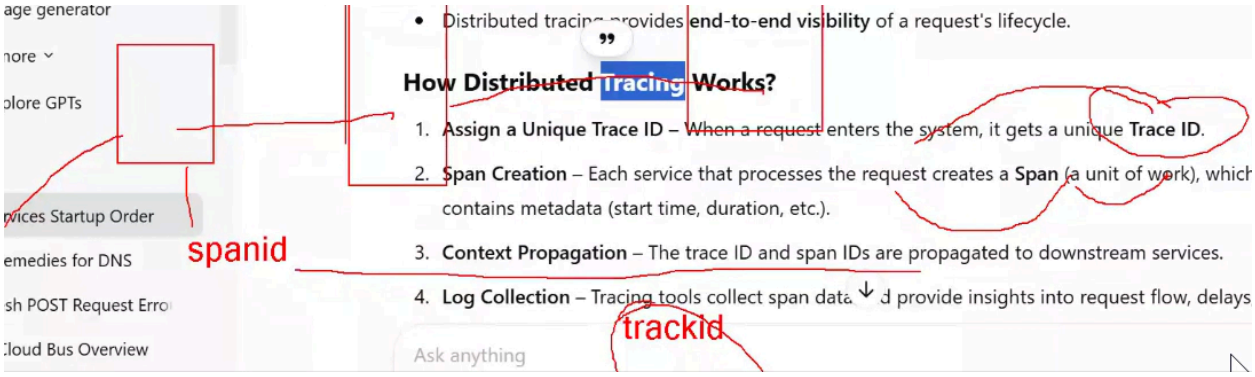
what is distributed tracing in microservice?

Distributed Tracing in Microservices

Distributed tracing is a technique used in microservices architecture to track and monitor requests as they travel across multiple services. It helps in understanding how a request propagates through different microservices and where delays or failures occur.

Why is Distributed Tracing Needed?

- In a **monolithic** system, tracking requests is easy because everything happens within a single application.
- In **microservices**, a single request may pass through multiple services, making debugging difficult.
- Distributed tracing provides **end-to-end visibility** of a request's lifecycle.



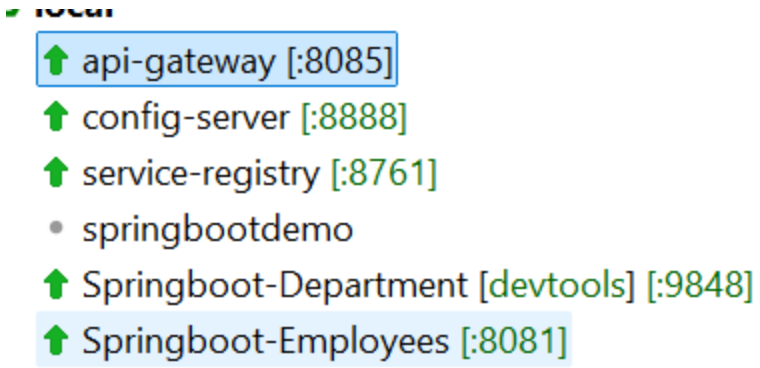
Popular Distributed Tracing Tools

- **Jaeger** – Open-source tool for tracing and monitoring.
- **Zipkin** – Distributed tracing system originally developed by Twitter.
- **Spring Cloud Sleuth** – Adds trace and span IDs to logs for distributed tracing in Spring Boot.
- **OpenTelemetry** – Standard for observability, supporting tracing, metrics, and logs.

Example Scenario

- A user request enters the **API Gateway**.
- The request is forwarded to **Employee Service**, which calls **Department Service**.
- Each service logs trace and span details.
- Using **Jaeger/Zipkin**, you can visualize the request flow and detect slow services.

what steps we need to follow to implement this micrometer tracing



We need to add few dependencies to the employee and dept micro services

```
<dependency>
   <groupId>io.micrometer</groupId>
   <artifactId>micrometer-tracing</artifactId>
</dependency>

<dependency>
   <groupId>io.micrometer</groupId>
   <artifactId>micrometer-tracing-bridge-brave</artifactId>
</dependency>

<dependency>
   <groupId>io.zipkin.reporter2</groupId>
   <artifactId>zipkin-reporter-brave</artifactId>
</dependency>
```

Now open the docker hub and search for openzipkin and copy the tag
docker pull openzipkin/zipkin:latest

Paste it in the cmd

```
C:\Users\miniMiracle>docker pull openzipkin/zipkin:latest
latest: Pulling from openzipkin/zipkin
fa1761cde767: Download complete
fe1bd6a77cbf: Download complete
954be67f124b: Download complete
5b24a42be3cf: Download complete
740e41e36047: Download complete
0219dbc309bb: Download complete
feef8f32146e: Download complete
b5137681fb36: Download complete
9212666e9465: Download complete
Digest: sha256:d9316e7ff757a256e5dd22ff97547e722649811bc5bfa428ecae7005045d5dbe
Status: Downloaded newer image for openzipkin/zipkin:latest
docker.io/openzipkin/zipkin:latest
```

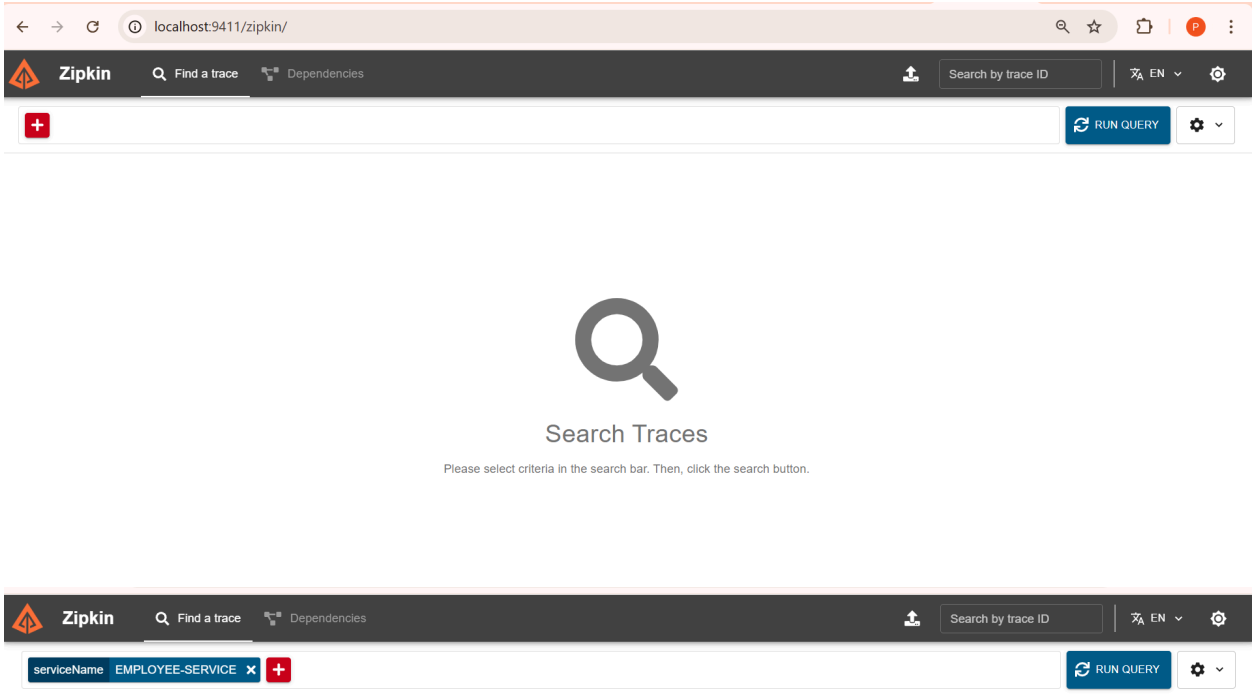
```
C:\Users\miniMiracle>docker images
REPOSITORY          TAG                       IMAGE ID       CREATED        SIZE
openzipkin/zipkin   latest                    d9316e7ff757   2 weeks ago    377MB
rabbitmq            3.13.7-management-alpine  d759525efd68   5 months ago   279MB
```

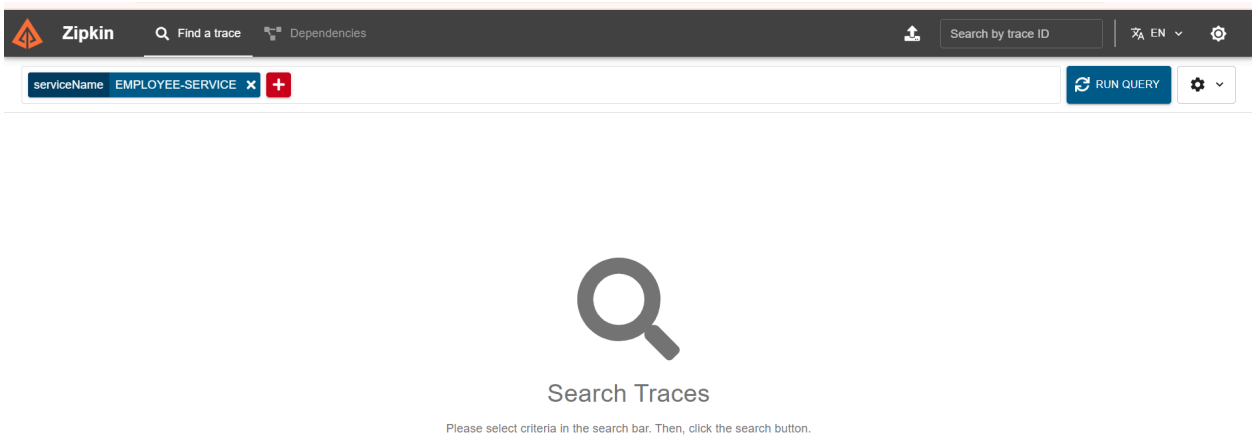
Now run the zipkin by using the docker command in cmd

```sh
docker run -d -p 9411:9411 --name zipkin openzipkin/zipkin
```

```
C:\Users\miniMiracle>docker run -d -p 9411:9411 --name zipkin openzipkin/zipkin
746c753022606c8ac3e6f5c7690a6e2fa561b7de8fbdf9bd37ed2eea08544996
```

We got the process id ….now we need to check whether it is running or not





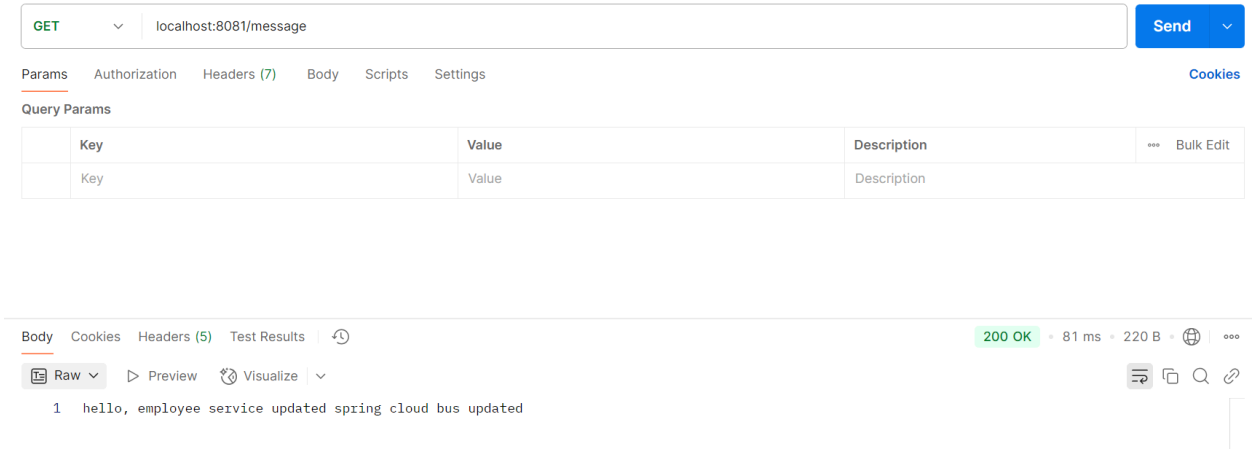
We have to get the visualizations….for that we need to add some properties

```
# Enable tracing and set sampling probability (1.0 = 100% of requests)
management.tracing.sampling.probability=1.0

# Configure Zipkin endpoint
management.zipkin.tracing.endpoint=http://localhost:9411/api/v2/spans

# Enable HTTP tracing
spring.application.name=employee-service
logging.level.org.springframework.web=DEBUG
```

Add those into properties of emp and dept too

```
spring.application.name=Department-Service
spring.config.import=optional:configserver:http://localhost:8888
management.endpoints.web.exposure.include=*

spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest




# Enable tracing and set sampling probability (1.0 = 100% of requests)
management.tracing.sampling.probability=1.0

# Configure Zipkin endpoint
management.zipkin.tracing.endpoint=http://localhost:9411/api/v2/spans


logging.level.org.springframework.web=DEBUG
```

Now restart two microservices



make req in the postman

Now run the zipkin

Zipkin | Find a trace | Dependencies | Search by trace ID | EN

RUN QUERY

Results

EXPAND ALL | COLLAPSE ALL | Service filters

Root	Start Time	Spans	Duration	
employee-service: http get	a few seconds ago (03/05 09:34:20:046)	1	180.365ms	SHOW
department-service: http get	a few seconds ago (03/05 09:34:20:260)	1	166.494ms	SHOW
department-service: http post	a few seconds ago (03/05 09:34:20:466)	1	64.398ms	SHOW
department-service: http get /message	a few seconds ago (03/05 09:34:22:919)	1	57.624ms	SHOW
employee-service: http post	a few seconds ago (03/05 09:34:20:259)	1	42.818ms	SHOW
employee-service: http get /message	a few seconds ago (03/05 09:34:34:428)	1	31.303ms	SHOW
employee-service: http get	a few seconds ago (03/05 09:34:50:251)	1	15.807ms	SHOW
employee-service: http put	a few seconds ago (03/05 09:34:50:251)	1	14.941ms	SHOW
department-service: http put	a few seconds ago (03/05 09:34:50:452)	1	8.406ms	SHOW
department-service: http get	a few seconds ago (03/05 09:34:50:452)	1	5.612ms	SHOW

Zipkin | Find a trace | Dependencies | Search by trace ID | EN

employee-service: http get

SPAN TABLE

Duration 180.365ms | Services 1 | Total Spans 1 | Trace ID 67c7cd4429970e0b614da676c924cf40

0ms | 60.122ms | 120.243ms | 180.365ms

Focus on selected span | Reset focus

0ms | 60.122ms | 120.243ms | 180.365ms

employee-service: http get 180.365ms

Service name: employee-service

Span name: http get

Span ID: 614da676c924cf40

Parent ID: none

Annotation

0ms | 60.122ms | 120.243ms | 180.365ms

0ms	Start Time	03/05 09:34:20.046
	Value	Client Start
	Address	192.168.0.125 (employee-service)
180.365ms	Start Time	03/05 09:34:20.227
	Value	Client Finish
	Address	192.168.0.125 (employee-service)

Tags

Zipkin | Find a trace | Dependencies | Search by trace ID | EN

serviceName department-service

RUN QUERY

Results

EXPAND ALL | COLLAPSE ALL | Service filters

Root	Start Time	Spans	Duration	
employee-service: http get	a few seconds ago (03/05 09:36:50:347)	1	19.772ms	SHOW
department-service: http get	a minute ago (03/05 09:35:50:481)	1	19.515ms	SHOW
employee-service: http put	a few seconds ago (03/05 09:36:50:347)	1	18.891ms	SHOW
employee-service: http get	a minute ago (03/05 09:36:20:326)	1	18.052ms	SHOW
department-service: http put	a minute ago (03/05 09:35:50:481)	1	17.943ms	SHOW
department-service: http get	a few seconds ago (03/05 09:36:50:534)	1	17.835ms	SHOW
department-service: http get	a minute ago (03/05 09:36:20:511)	1	17.409ms	SHOW
department-service: http put	a minute ago (03/05 09:36:20:511)	1	16.886ms	SHOW
employee-service: http put	a minute ago (03/05 09:36:20:326)	1	16.797ms	SHOW
department-service: http put	a few seconds ago (03/05 09:36:50:534)	1	16.533ms	SHOW