

# Spring Cloud로 개발하는 マイクロ서비스 アプリケーション



+



Spring  
Cloud

```
CLASS Book
def save(f, title, price, author):
    self.title = title
    self.price = price
    self.author = author
    var fs = require('fs');
    fs.readFile('JSONE.txt' /* 1 */,
    function (err, data) {
        console.log(data); // 3
    });

@Interface NextInnovationDelegate : NSObject < UIApplicationDelegate >
```

## Section 4. 분산 추적

- CircuitBreaker
- Resilience4j
- Distributed Tracing
- Trace ID and Span ID
- Zipkin server 활용
- Monitoring

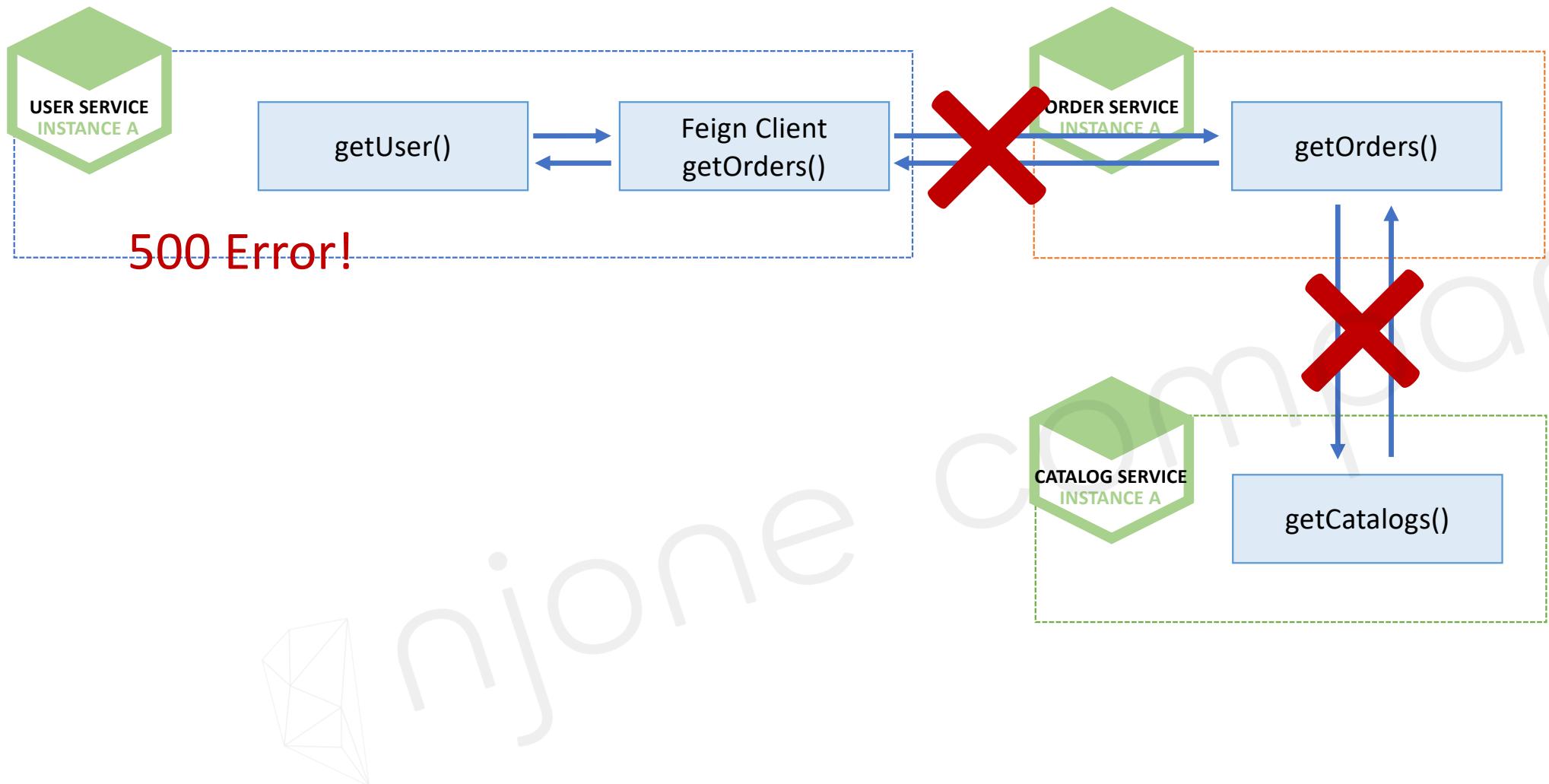
# Microservice 통신 시 연쇄 오류

The screenshot shows a Postman request for a GET endpoint at `127.0.0.1:8000/user-service/users/282d7347-9c0f-4b2c-b85a-282d2c27847a`. The response status is 500 Internal Server Error, with a time of 260 ms and a size of 12.76 KB. The response body is a JSON object containing:

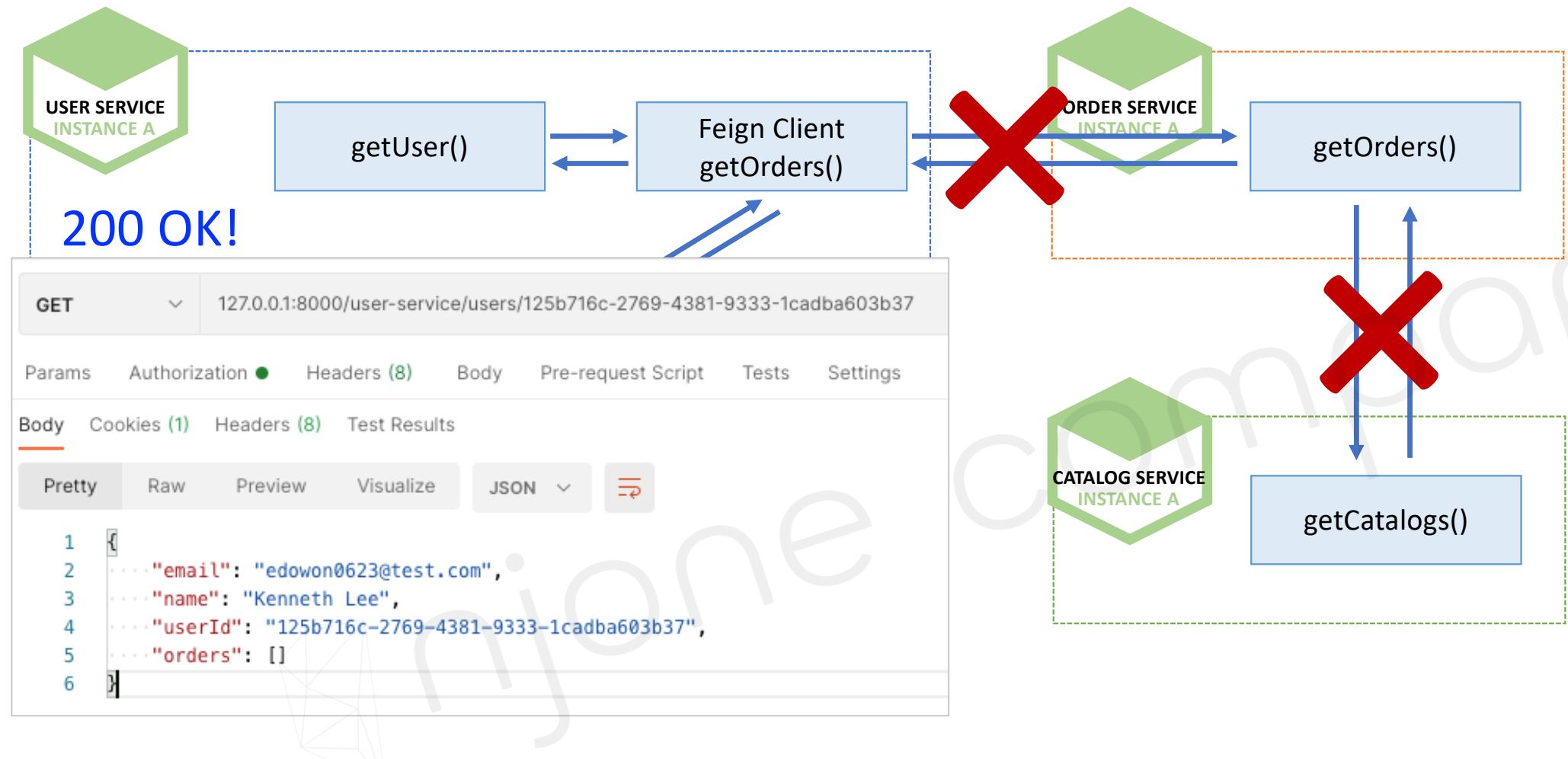
```
1 {  
2   "timestamp": "2021-03-01T13:19:05.019+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "trace": "feign.RetryableException: order-service executing GET http://order-service/order-service/282d7347-9c0f-4b2c-b85a-282d2c27847a"}  
Status: 500 Internal Server Error Time: 260 ms Size: 12.76 KB
```

```
java.net.UnknownHostException Create breakpoint : order-service  
at java.base/sun.nio.ch.NioSocketImpl.connect(NioSocketImpl.java:567) ~[na:na]  
at java.base/java.net.Socket.connect(Socket.java:648) ~[na:na]  
at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:177) ~[na:na]  
at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:474) ~[na:na]
```

# Microservice 통신 시 연쇄 오류



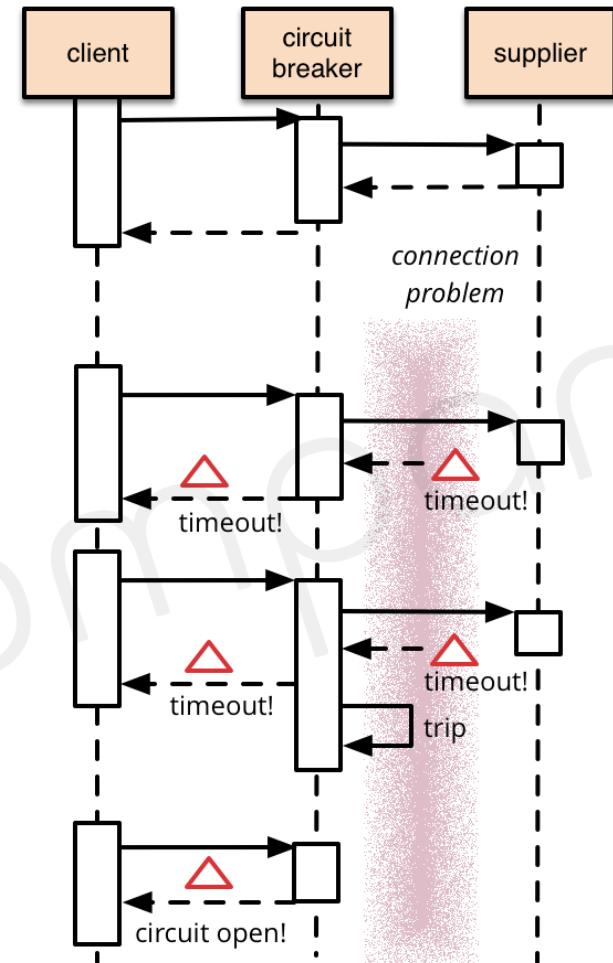
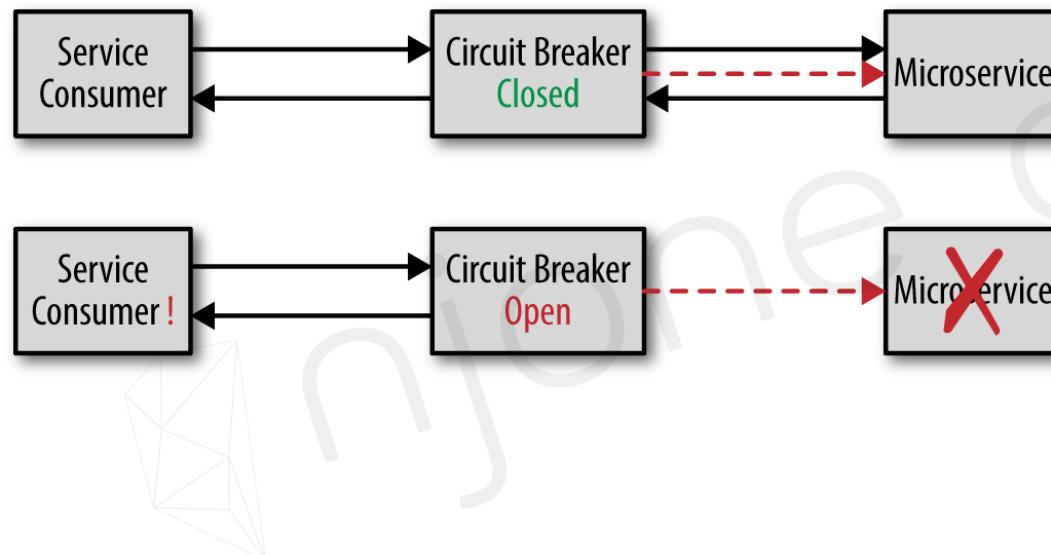
# Microservice 통신 시 연쇄 오류



# CircuitBreaker

## ■ CircuitBreaker

- <https://martinfowler.com/bliki/CircuitBreaker.html>
- 장애가 발생하는 서비스에 반복적인 호출이 되지 못하게 차단
- 특정 서비스가 정상적으로 동작하지 않을 경우 다른 기능으로 대체 수행 → 장애 회피





# Spring Cloud Netflix Hystrix

- Spring Cloud Netflix Hystrix

CURRENT	REPLACEMENT
Hystrix	Resilience4j
Hystrix Dashboard / Turbine	Micrometer + Monitoring System
Ribbon	Spring Cloud Loadbalancer
Zuul 1	Spring Cloud Gateway
Archaius 1	Spring Boot external config + Spring Cloud Config

Replacements technologies. Taken from [1]

er-ws)

r-ws)



# Resilience4j

---

- Resilience4j

- resilience4j-circuitbreaker: Circuit breaking
- resilience4j-ratelimiter: Rate limiting
- resilience4j-bulkhead: Bulkheading
- resilience4j-retry: Automatic retrying (sync and async)
- resilience4j-timelimiter: Timeout handling
- resilience4j-cache: Result caching
- <https://resilience4j.readme.io/docs/getting-started>
- <https://github.com/resilience4j/resilience4j>

## Introduction

---

Resilience4j is a lightweight, easy-to-use fault tolerance library inspired by [Netflix Hystrix](#), but designed for Java 8 and functional programming. Lightweight, because the library only uses [Vavr](#), which does not have any other external library dependencies. Netflix Hystrix, in contrast, has a compile dependency to Archaius which has many more external library dependencies such as Guava and Apache Commons Configuration.





# Resilience4j – CircuitBreaker

- DefaultConfiguration

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>
</dependency>
```

pom.xml

```
@Autowired
CircuitBreakerFactory circuitBreakerFactory;

// 
/* ErrorDecoder */
List<ResponseOrder> ordersList = orderServiceClient.getOrders(userId);

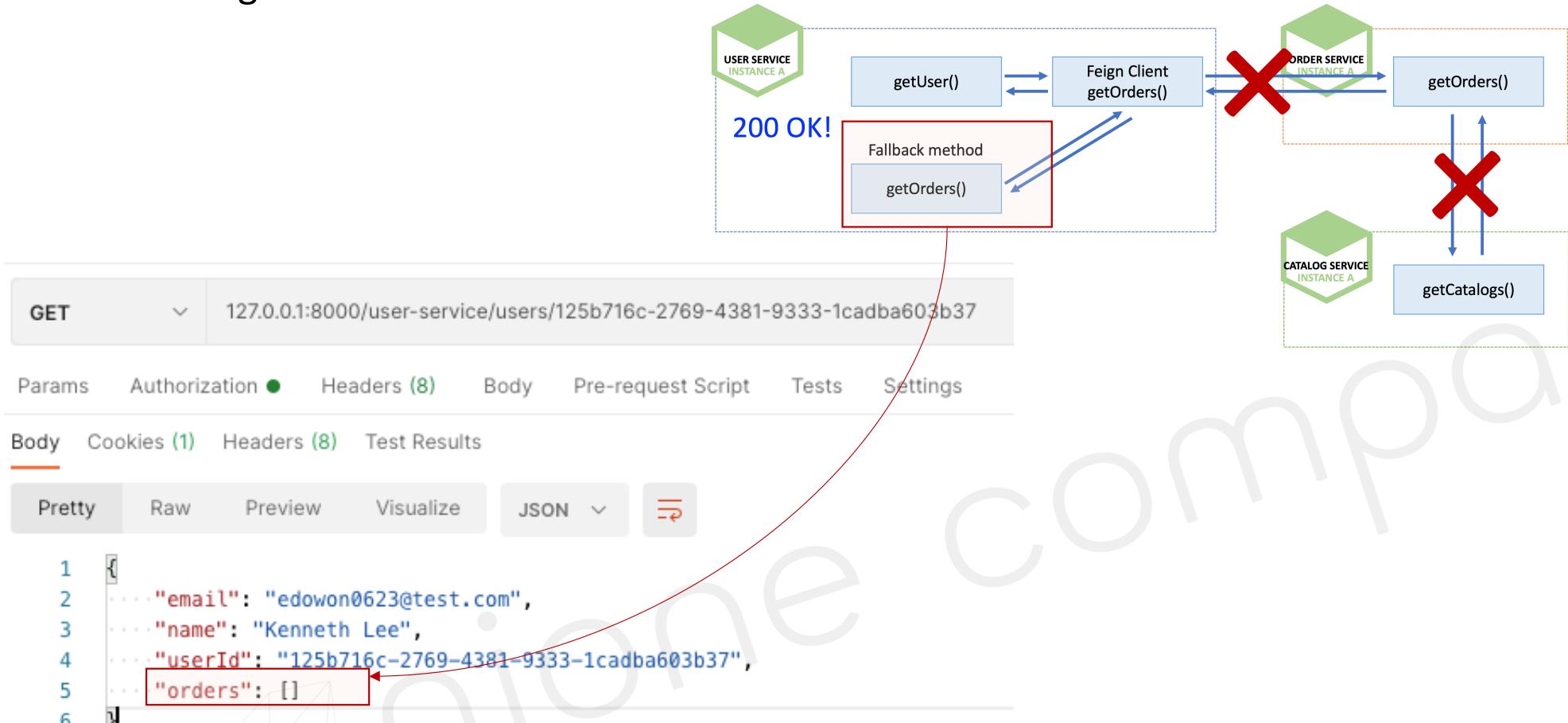
/* CircuitBreaker */
CircuitBreaker circuitBreaker = circuitBreakerFactory.create("circuitbreaker");
List<ResponseOrder> ordersList = circuitBreaker.run(() -> orderServiceClient.getOrders(userId),
    throwable -> new ArrayList<>());

userDto.setOrders(ordersList);
```

UserServiceImpl.java

# Resilience4j – CircuitBreaker

## ▪ DefaultConfiguration



# Resilience4j – CircuitBreaker

## ■ Customize CircuitBreakerFactory → *Resilience4JCircuitBreakerFactory*

```
@Configuration  
public class Resilience4JConfiguration {  
    @Bean  
    public Customizer<Resilience4JCircuitBreakerFactory> globalCustomConfiguration() {  
  
        // the circuitBreakerConfig and timeLimiterConfig objects  
        CircuitBreakerConfig circuitBreakerConfig = CircuitBreakerConfig.custom()  
            .failureRateThreshold(4)  
            .waitDurationInOpenState(Duration.ofMillis(1000))  
            .slidingWindowType(CircuitBreakerConfig.SlidingWindowType.COUNT_BASED)  
            .slidingWindowSize(2)  
            .build();  
    }  
}
```

- CircuitBreaker가 종료(Close) 되기 위한 조건을 결정하는 타입
- COUNT\_BASED or TIME\_BASED

- CircuitBreaker가 종료(Close) 되기 위한 조건 값의 크기
- default: 100

- CircuitBreaker를 실행 할지(Open) 결정하는 failure rate threshold percentage
- default: 50

- CircuitBreaker를 open한 상태를 유지하는 지속 기간을 의미
- 이 기간 이후에 Half-open 상태
- default: 60seconds



# Resilience4j – CircuitBreaker

- Customize CircuitBreakerFactory → **Resilience4JCircuitBreakerFactory**

```
TimeLimiterConfig timeLimiterConfig = TimeLimiterConfig.custom()
    .timeoutDuration(Duration.ofSeconds(4))
    .build();

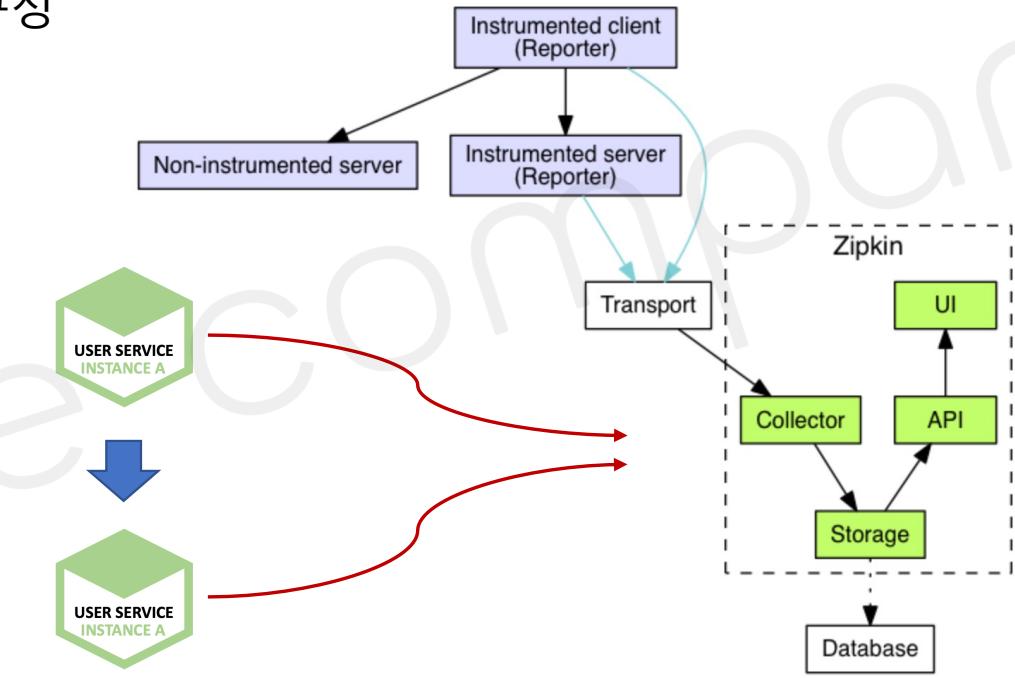
return factory -> factory.configureDefault(id -> new Resilience4JConfigBuilder(id)
    .timeLimiterConfig(timeLimiterConfig)
    .circuitBreakerConfig(circuitBreakerConfig)
    .build());
}
```

- TimeLimiter는 future supplier의 time limit을 정하는 API
- default: 1초

# Microservice 분산 추적

## ■ Zipkin

- <https://zipkin.io/>
- Twitter에서 사용하는 분산 환경의 Timing 데이터 수집, 추적 시스템 (오픈소스)
- Google Drapper에서 발전하였으며, 분산환경에서의 시스템 병목 현상 파악
- Collector, Query Service, Databasem WebUI로 구성
- **Span**
  - 하나의 요청에 사용되는 작업의 단위
  - 64 bit unique ID
- **Trace**
  - 트리 구조로 이뤄진 Span 셋
  - 하나의 요청에 대한 같은 Trace ID 발급





# Microservice 분산 추적

## ■ Spring Cloud Sleuth

- 스프링 부트 애플리케이션을 Zipkin과 연동
- 요청 값에 따른 Trace ID, Span ID 부여
- Trace와 Span Ids를 로그에 추가 가능
  - servlet filter
  - rest template
  - scheduled actions
  - message channels
  - feign client

## Spring Cloud Sleuth 3.0.1



OVERVIEW

LEARN

SAMPLES

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

### Features

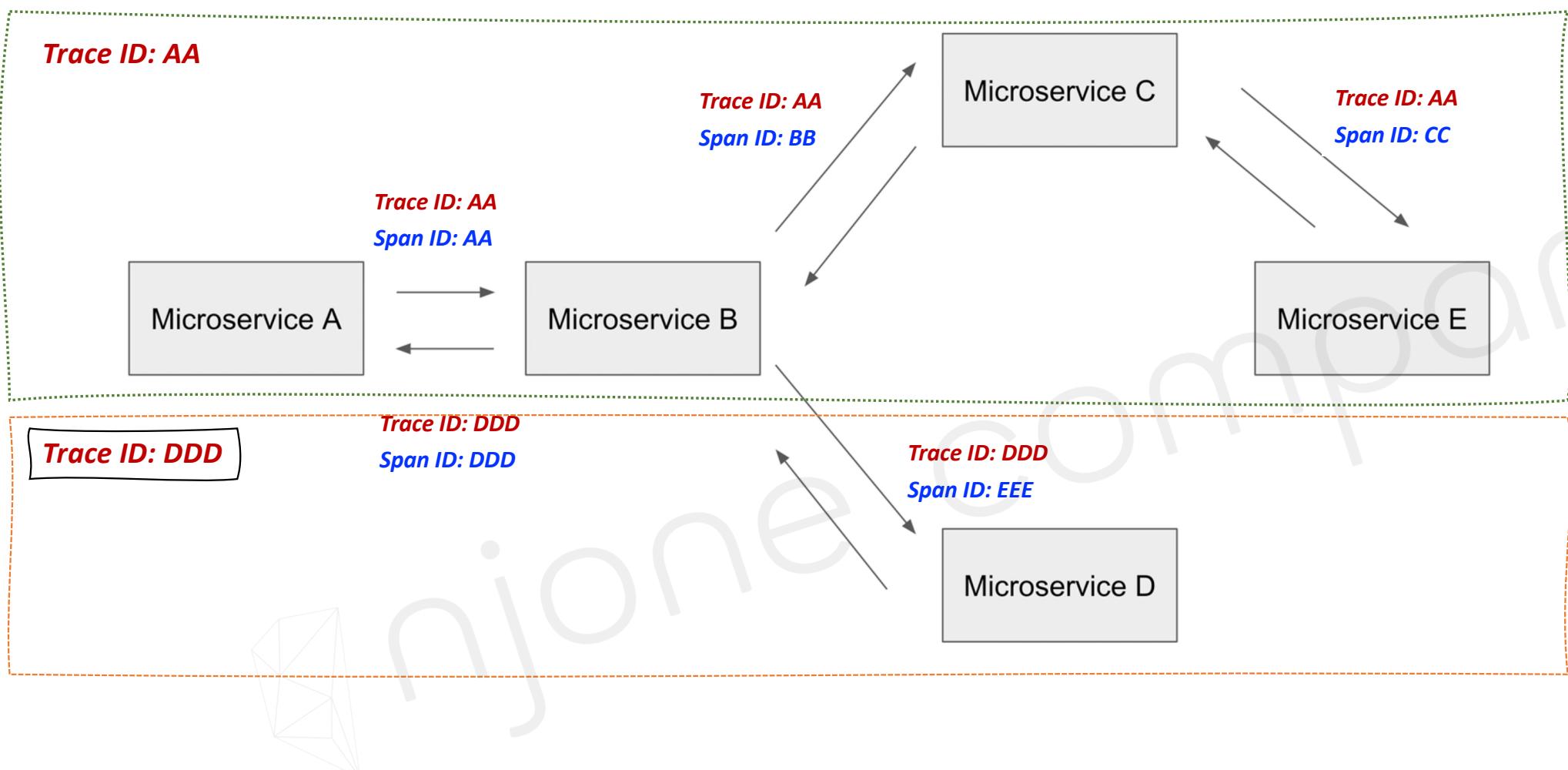
Sleuth configures everything you need to get started. This includes where trace data (spans) are reported to, how many traces to keep (sampling), if remote fields (baggage) are sent, and which libraries are traced.

Specifically, Spring Cloud Sleuth...

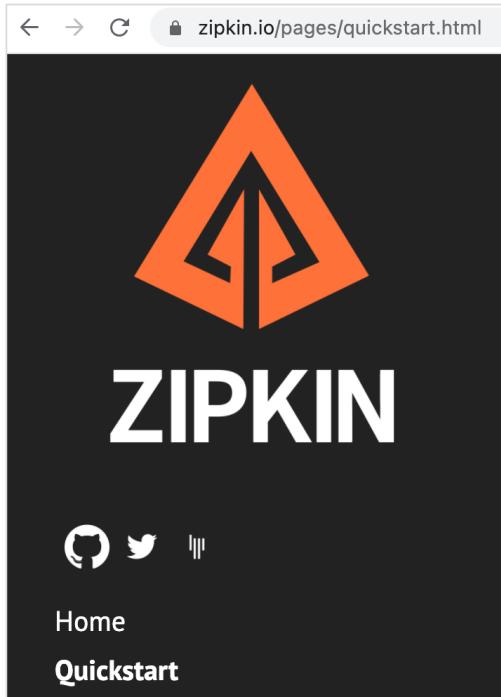
- Adds trace and span ids to the Slf4J MDC, so you can extract all the logs from a given trace or span in a log aggregator.
- Instruments common ingress and egress points from Spring applications (servlet filter, rest template, scheduled actions, message channels, feign client).
- If `spring-cloud-sleuth-zipkin` is available then the app will generate and report Zipkin-compatible traces via HTTP. By default it sends them to a Zipkin collector service on localhost (port 9411). Configure the location of the service using `spring.zipkin.baseUrl`.

# Microservice 분산 추적

- Spring Cloud Sleuth + Zipkin



# Zipkin server 설치



The screenshot shows the Zipkin quickstart page at [zipkin.io/pages/quickstart.html](https://zipkin.io/pages/quickstart.html). The page has a header with a keyboard icon and the title "Zipkin server 설치". The main content area contains the Zipkin logo and navigation links for "Home" and "Quickstart". On the right, there are sections for "Docker" and "Java", each with a command-line example. Below these is a terminal window showing the execution of the Java command.

Home

Quickstart

## Docker

The Docker Zipkin project is able to build docker images, provide scripts and a `docker-compose.yml` for launching pre-built images. The quickest start is to run the latest image directly:

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

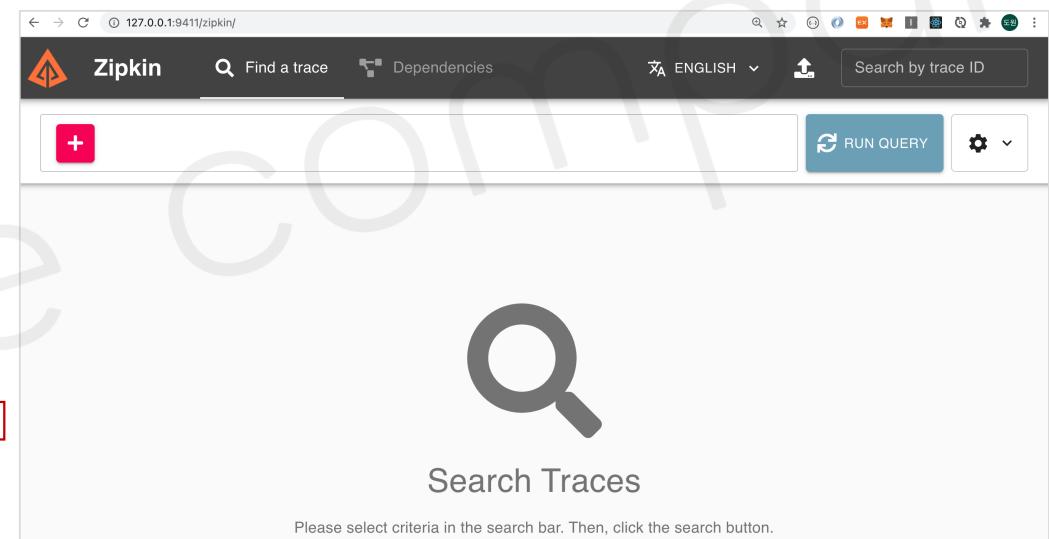
## Java

If you have Java 8 or higher installed, the quickest way to get started is to fetch the latest release as a self-contained executable jar:

```
curl -sSL https://zipkin.io/quickstart.sh | bash -s  
java -jar zipkin.jar
```

```
▶ curl -sSL https://zipkin.io/quickstart.sh | bash -s  
Thank you for trying Zipkin!  
This installer is provided as a quick-start helper, so you can try Zipkin out  
without a lengthy installation process.  
  
Fetching version number of latest io.zipkin:zipkin-server release...  
Latest release of io.zipkin:zipkin-server seems to be 2.23.2  
  
Downloading io.zipkin:zipkin-server:2.23.2:exec to zipkin.jar...
```

# Zipkin server 기동



# Users Microservice 수정

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-zipkin</artifactId>
    <version>2.2.3.RELEASE</version>
</dependency>
```

pom.xml

```
spring:
  application:
    name: user-service
  zipkin:
    base-url: http://localhost:9411
    enabled: true
  sleuth:
    sampler:
      probability: 1.0
```

application.yml

```
/* CircuitBreaker */
log.info("Before call orders microservice");
CircuitBreaker circuitBreaker = circuitBreakerFactory.create("my_circuit_breaker");
List<ResponseOrder> ordersList = circuitBreaker.run(() -> orderServiceClient.getOrders(userId),
    throwable -> new ArrayList<>());
log.info("after called orders microservice");

userDto.setOrders(ordersList);
```

UsersServiceImpl.java

# Trace ID and Span ID

POST http://127.0.0.1:8000/order-service/a09741d4-b5c4-44fb-a9a6-8a6312e9b58c/orders

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "productId": "CATALOG-003",
3   ... "qty": 5,
4   ... "unitPrice": 1500
5 }
```

Body Cookies (1) Headers (3) Test Results Status: 201 Created

Pretty Raw Preview Visualize JSON

```
1 {
2   ... "productId": "CATALOG-003",
3   ... "qty": 5,
4   ... "unitPrice": 1500,
5   ... "totalPrice": 7500,
6   ... "orderId": "6f3bed3b-0ba4-483b-83b1-2358b3bba500"
7 }
```

2021-03-02 23:47:35.314 INFO [order-service,a04b6cf3c9338a13,a04b6cf3c9338a13] 96789 --  
Hibernate: call next value for hibernate\_sequence  
Hibernate: insert into orders (order\_id, product\_id, qty, total\_price, unit\_price, user\_id, id) values (?, ?, ?, ?, ?, ?, ?)  
[o-auto-1-exec-1] c.e.o.controller.OrdersController : Before add orders data  
[o-auto-1-exec-1] c.e.o.controller.OrdersController : After added orders data

Trace ID Span ID

# Trace ID and Span ID

Zipkin

Find a trace Dependencies ENGLISH

Search by trace ID  
a04b6cf3c9338a13

ORDER-SERVICE: post /order-service/{userid}/orders

Duration: 153.720ms Services: 1 Depth: 1 Total Spans: 1 Trace ID: a04b6cf3c9338a13

DOWNLOAD JSON

0ms | 51.240ms | 102.480ms | 153.720ms

ORDER-SERVICE post /order-service/{userid}/orders [153.720ms]

ORDER-SERVICE post /order-service/{userid}/orders

Span ID:a04b6cf3c9338a13 Parent ID:None

Annotations

SHOW ALL ANNOTATIONS

Tags

http.method POST

http.path /order-service/a09741d4-b5c4-44fb-a9a6-8a6312e9b58c/orders

mvc.controller.class OrdersController

mvc.controller.method createOrder

Trace ID

2021-03-02 23:47:35.314 INFO [order-service,a04b6cf3c9338a13,a04b6cf3c9338a13] 96789 -->  
Hibernate: call next value for hibernate\_sequence  
Hibernate: insert into orders (order\_id, product\_id, qty, total\_price, unit\_price, user\_id) values (?, ?, ?, ?, ?, ?)

2021-03-02 23:47:35.413 INFO [order-service,a04b6cf3c9338a13,a04b6cf3c9338a13] 96789 -->

# Trace ID and Span ID

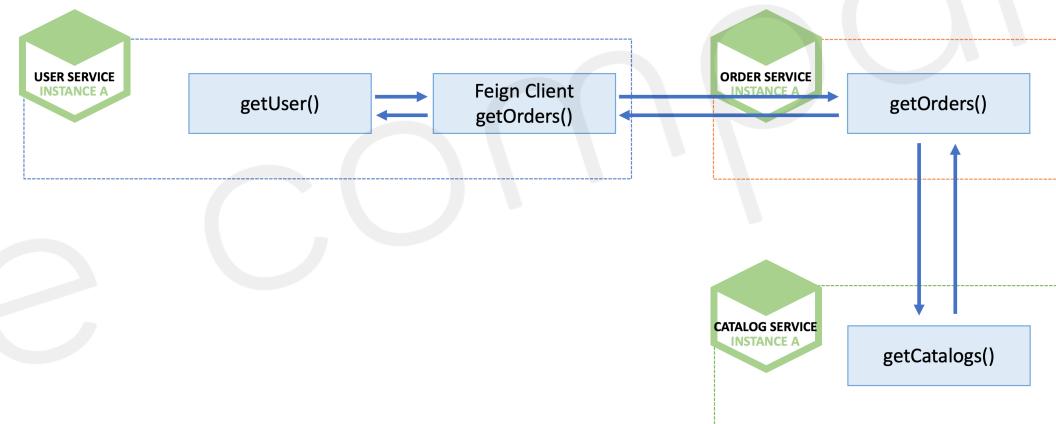
GET 127.0.0.1:8000/user-service/users/a09741d4-b5c4-44fb-a9a6-8a6312e9b58c

Params Authorization ● Headers (8) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON ↻

```
1 "email": "edown0623@test.com",
2 "name": "Kenneth Lee",
3 "userId": "a09741d4-b5c4-44fb-a9a6-8a6312e9b58c",
4 "orders": [
5     {
6         "productId": "CATALOG-004",
7         "qty": 10,
8         "unitPrice": 1500,
9         "totalPrice": 15000,
10        "createdAt": "2021-03-02T14:47:35.406+00:00",
11        "orderId": "73b93f3d-8dee-4bd0-9a23-05d465be75f9"
12    },
13 ]
```



# Trace ID and Span ID



# Zipkin server 활용

Zipkin Find a trace Dependencies ENGLISH e132359d05efde94

USER-SERVICE: get /users/{userid}

Duration: 3.462s Services: 2 Depth: 4 Total Spans: 4 Trace ID: e132359d05efde94 DOWNLOAD JSON

USER-SERVICE

0ms 1.154s 2.308s 3.462s

get /users/{userid} [3.462s]

userserviceimpl\$\$lambda\$1778/0x00000008016cc040 [3.276s]

get [3.147s]

get /order-service/{userid}/orders [3.139s]

ORDER-SERVICE

get /order-service/{userid}/orders [3.139s]

Annotations

Span ID: 4e9274e54e747ed0 Parent ID: 585681926fe50087 SHOW ALL ANNOTATIONS

Tags

http.method GET

http.path /order-service/a09741d4-b5c4-44fb-a9a6-8a6312e9b58c/orders

mvc.controller.class OrdersController

mvc.controller.method getOrder

# Zipkin server 활용

The screenshot shows the Zipkin web interface. At the top, there is a navigation bar with the Zipkin logo, a search bar labeled "Find a trace", a "Dependencies" button, language selection ("ENGLISH"), and a "Search by trace ID" input field. Below the search bar, there is a filter input field containing "serviceName user-service" with a red border and a red "+" button. To the right of the input field are "RUN QUERY" and "Service filters" buttons.

The main area displays "2 Results". There are two trace entries:

Root	Start Time	Spans	Duration	Action
user-service: get /users/{userid}	13 minutes ago (03/02 23:48:17:274)	4	3.462s	SHOW
user-service: post /users	14 minutes ago (03/02 23:47:24:587)	1	278.307ms	SHOW

# Zipkin에서의 추적

The screenshot shows the Zipkin web interface with the following details:

- Header:** Zipkin logo, "Find a trace" search bar, "Dependencies" tab (highlighted in red), "ENGLISH" language switch, and "Search by trace ID" input field.
- Time Range:** Start Time: 03/02/2021 00:01:41, End Time: 03/03/2021 01:02:41.
- Dependency Trace:** A visual representation of a dependency trace from "user-service" to "order-service". A blue dot labeled "user-service" is connected by a line to a blue dot labeled "order-service".
- Service Overview:** A panel for "user-service" showing its dependencies.
  - Uses (traced requests):** A large green circle representing the service's dependencies.
  - Summary:** "This service uses 1 service".
  - Table:** A table showing the service's interactions with "order-service".

Service Name	Call	Error
order-service	4	0



# Orders Microservice 장애 발생

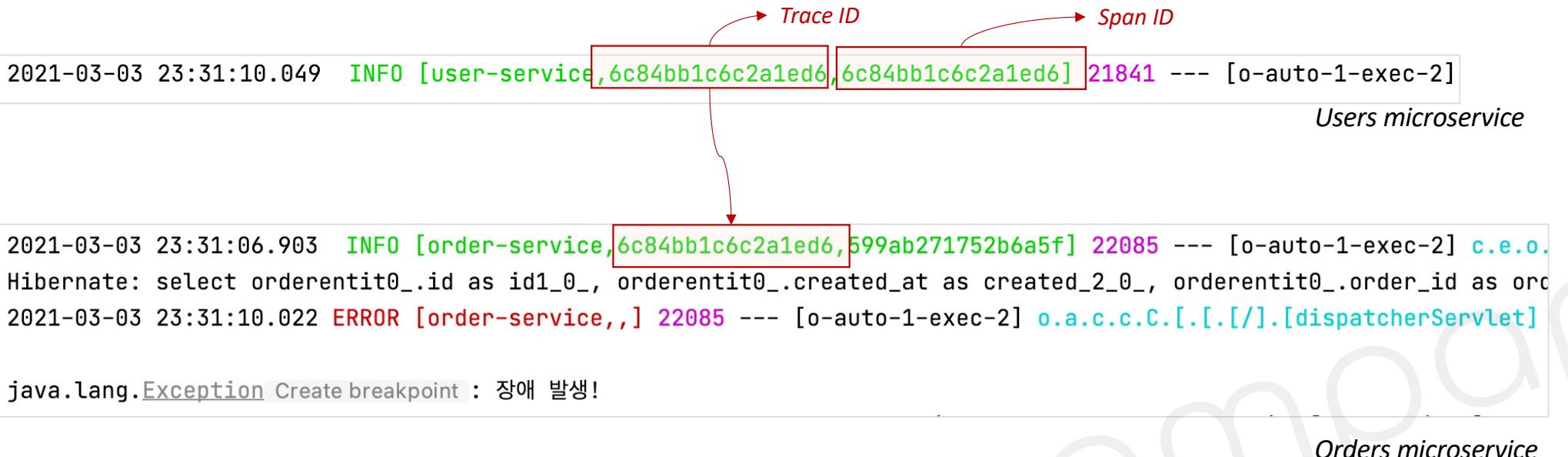
```
@GetMapping(value="/{userId}/orders")
public ResponseEntity<List<ResponseOrder>> getOrder(@PathVariable("userId") String userId) throws Exception {
    log.info("Before retrieve orders data");
    Iterable<OrderEntity> orderList = ordersService.getOrdersByUserId(userId);

    List<ResponseOrder> result = new ArrayList<>();
    orderList.forEach(v -> {
        result.add(new ModelMapper().map(v, ResponseOrder.class));
    });

    try {
        Thread.sleep( millis: 3000);
        throw new Exception("장애 발생!");
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    log.info("After retrieved orders data");

    return ResponseEntity.status(HttpStatus.OK).body(result);
}
```

# Orders Microservice 장애 발생

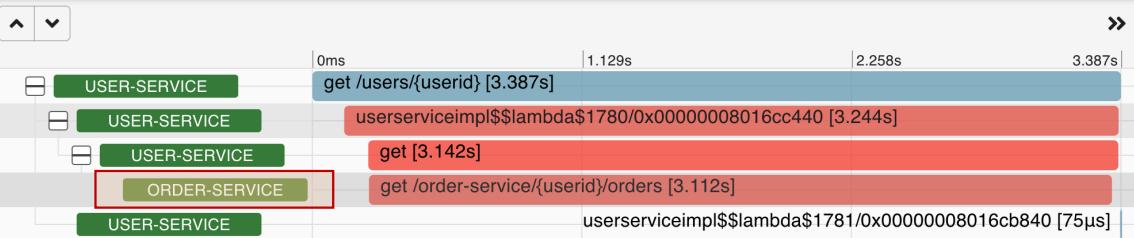


# Orders Microservice 장애 발생

USER-SERVICE: get /users/{userid}

Duration: 3.387s Services: 2 Depth: 4 Total Spans: 5 Trace ID: 6c84bb1c6c2a1ed6

[DOWNLOAD JSON](#)



get /order-service/{userid}/orders

Span ID: 599ab271752b6a5f Parent ID: 33fcbe0b65c0a1e9

## Annotations

## Tags

error
Request processing failed; nested exception is java.lang.Exception: 장애 발생!
http.method
GET
http.path
/order-service/352e4ef3-40d8-48d3-be57-37a99aaa7e0c/orders



order-service

[TRACES](#)

Used (traced requests)



This service is used by 1 service

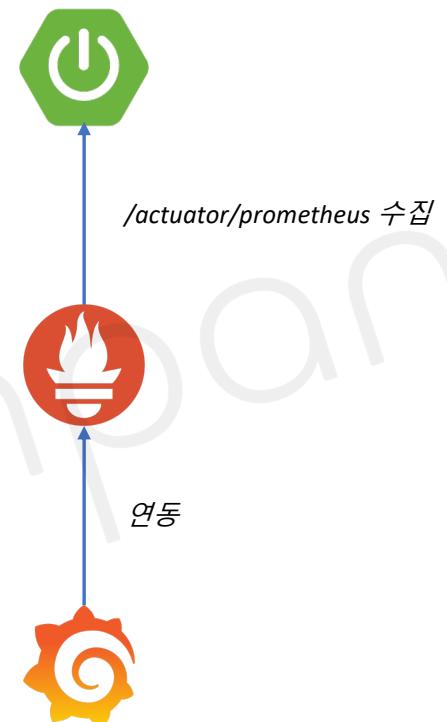
Service Name	Call	Error
user-service	1	1



# Prometheus + Grafana

## ■ Prometheus

- Metrics를 수집하고 모니터링 및 알람에 사용되는 오픈소스 애플리케이션
- 2016년부터 CNCF에서 관리되는 2번째 공식 프로젝트
  - Level DB → Time Series Database(TSDB)
- Pull 방식의 구조와 다양한 Metric Exporter 제공
- 시계열 DB에 Metrics 저장 → 조회 가능 (Query)



## ■ Grafana

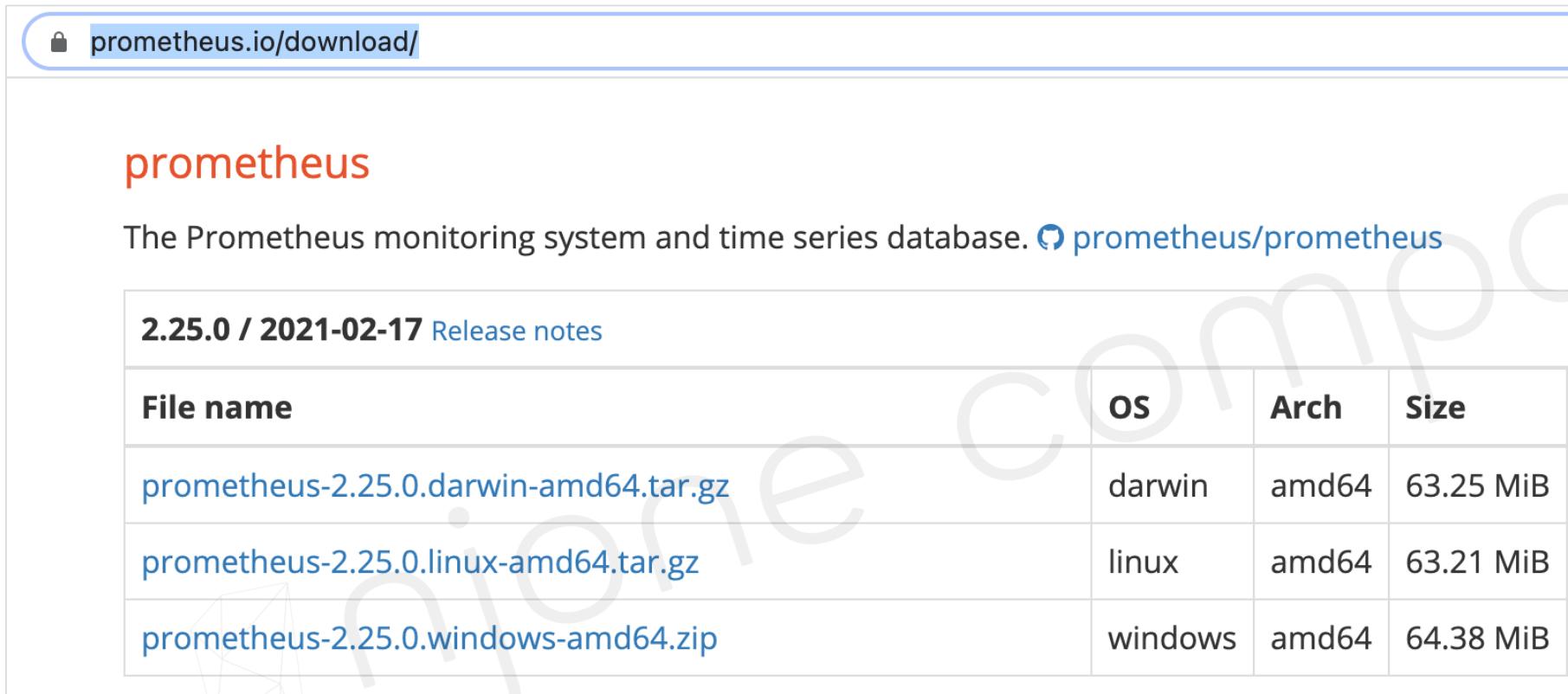
- 데이터 시각화, 모니터링 및 분석을 위한 오픈소스 애플리케이션
- 시계열 데이터를 시각화하기 위한 대시보드 제공



# Prometheus 설치

## ■ Prometheus 다운로드

- <https://prometheus.io/download/>



The screenshot shows the official Prometheus download page at <https://prometheus.io/download/>. The page title is "prometheus" in red. Below it, the text reads "The Prometheus monitoring system and time series database." followed by a GitHub link. A table lists the available releases:

File name	OS	Arch	Size
<a href="#">prometheus-2.25.0.darwin-amd64.tar.gz</a>	darwin	amd64	63.25 MiB
<a href="#">prometheus-2.25.0.linux-amd64.tar.gz</a>	linux	amd64	63.21 MiB
<a href="#">prometheus-2.25.0.windows-amd64.zip</a>	windows	amd64	64.38 MiB



# Prometheus

- prometheus.yml 파일 수정

- target 지정

```
21 scrape_configs:  
22   # The job name is added as a label `job=<job_name>` to any timeseries  
scraped from this config.  
23   - job_name: 'prometheus'  
24  
25     # metrics_path defaults to '/metrics'  
26     # scheme defaults to 'http'.  
27  
28     static_configs:  
29       - targets: ['localhost:9090']  
30  
31   - job_name: 'user-service'  
32     scrape_interval: 15s  
33     metrics_path: '/user-service/actuator/prometheus'  
34     static_configs:  
35       - targets: ['localhost:8000']
```

# Prometheus

## ■ Prometheus 서버 실행

```
▶ ls -l
total 334208
-rw-r--r--@ 1 downonlee staff 11357 2 18 01:11 LICENSE
-rw-r--r--@ 1 downonlee staff 3420 2 18 01:11 NOTICE
drwxr-xr-x@ 4 downonlee staff 128 2 18 01:11 console_libraries
drwxr-xr-x@ 9 downonlee staff 288 2 18 01:11 consoles
drwxr-xr-x 12 downonlee staff 384 3 5 00:00 data
-rwxr-xr-x@ 1 downonlee staff 90568240 2 17 23:28 prometheus
-rw-r--r--@ 1 downonlee staff 1089 3 1 23:28 prometheus.yml
-rwxr-xr-x@ 1 downonlee staff 80523216 2 17 23:30 promtool
```

▶ 로컬 디스크 (C:) > Work > prometheus-2.25.0.windows-amd64		
이름	수정한 날짜	유형
console_libraries	2021-03-06 오전 12:31	파일 폴더
consoles	2021-03-06 오전 12:31	파일 폴더
data	2021-03-06 오전 12:32	파일 폴더
LICENSE	2021-03-06 오전 12:31	파일
NOTICE	2021-03-06 오전 12:31	파일
<b>prometheus.exe</b>	2021-03-06 오전 12:31	응용 프로그램
<b>prometheus.yml</b>	2021-03-06 오전 12:31	YML 파일
<b>promtool.exe</b>	2021-03-06 오전 12:31	응용 프로그램

```
$ ./prometheus --config.file=prometheus.yml
```

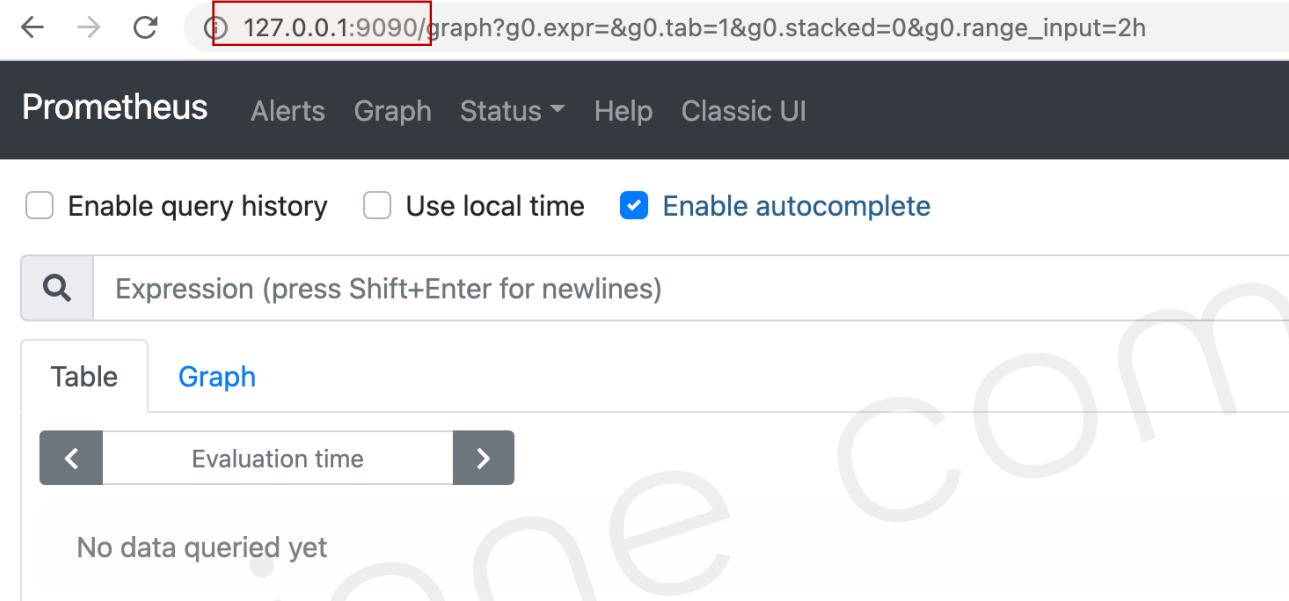
```
▶ ./prometheus --config.file=prometheus.yml
level=info ts=2021-03-04T07:51:29.289Z caller=main.go:366 msg="No time or size retention was set so using the default time retention"
level=info ts=2021-03-04T07:51:29.289Z caller=main.go:404 msg="Starting Prometheus" version="(version=2.25.0, branch=HEAD, revision=0e4bd0bd4c00ad6e2)"
level=info ts=2021-03-04T07:51:29.289Z caller=main.go:409 build_context="(go=go1.15.8, user=root@615f028225c9, date=20210217-14:26:1"
level=info ts=2021-03-04T07:51:29.289Z caller=main.go:410 host_details=(darwin)
level=info ts=2021-03-04T07:51:29.289Z caller=main.go:411 fd_limits="(soft=256, hard=unlimited)"
level=info ts=2021-03-04T07:51:29.289Z caller=main.go:412 vm_limits="(soft=unlimited, hard=unlimited)"
level=info ts=2021-03-04T07:51:29.293Z caller=web.go:532 component=web msg="Start listening for connections" address=0.0.0.0:9090
level=info ts=2021-03-04T07:51:29.293Z caller=main.go:779 msg="Starting TSDB ..."
level=info ts=2021-03-04T07:51:29.295Z caller=tls_config.go:191 component=web msg="TLS is disabled." http2=false
```



# Prometheus

## ■ Prometheus Dashboard

- <http://127.0.0.1:9090>



The screenshot shows the Prometheus web interface. At the top, there is a navigation bar with links for Prometheus, Alerts, Graph, Status, Help, and Classic UI. Below the navigation bar, there are three checkboxes: "Enable query history" (unchecked), "Use local time" (unchecked), and "Enable autocomplete" (checked). A search bar labeled "Expression (press Shift+Enter for newlines)" is present. Below the search bar, there are two tabs: "Table" (selected) and "Graph". A "Evaluation time" input field with arrows for navigation is located below the tabs. The main content area displays the message "No data queried yet". At the bottom left, there is a blue button labeled "Add Panel" next to a small geometric icon.

# Prometheus

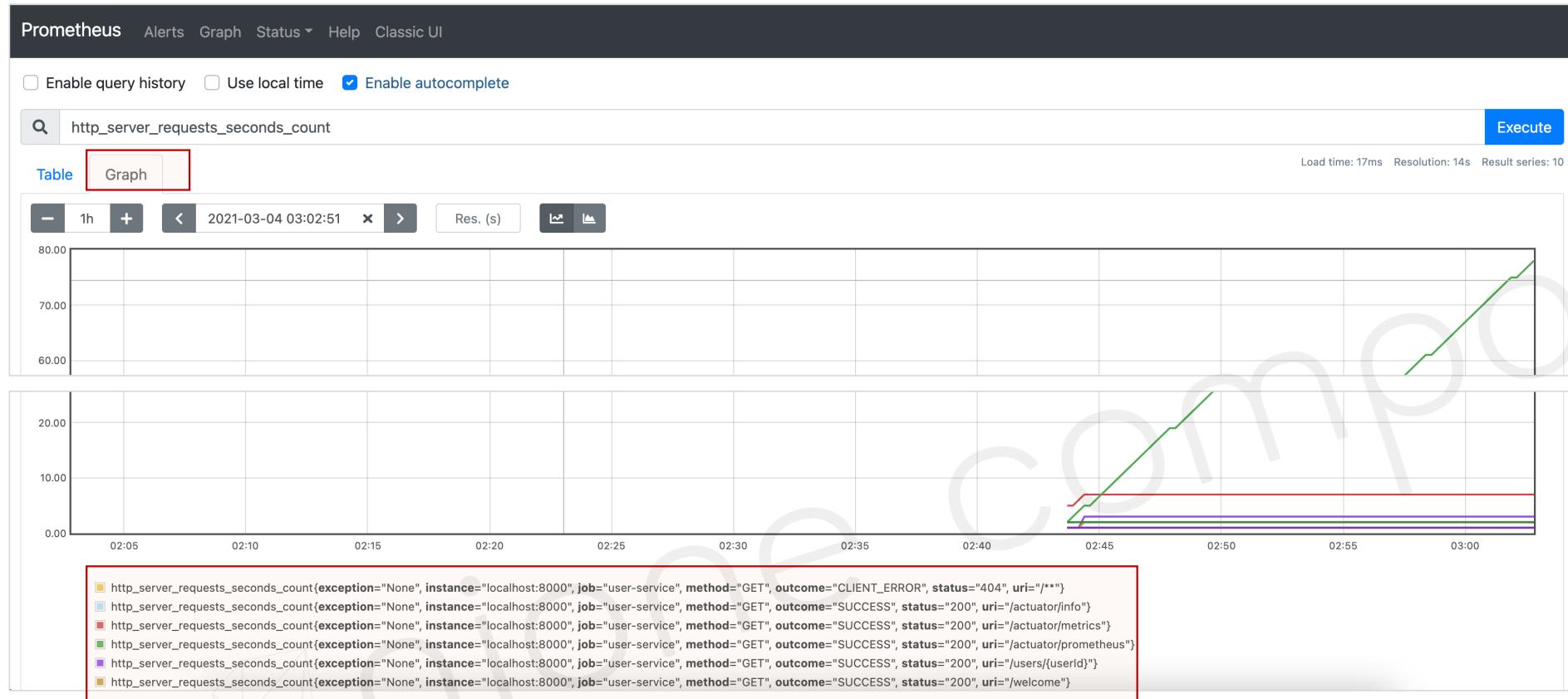
## ■ metrics 검사 – Table

The screenshot shows the Prometheus web interface with the following details:

- Header:** Prometheus, Alerts, Graph, Status, Help, Classic UI.
- Search Bar:** Contains the query `http_server_requests_seconds_count`.
- Filter Buttons:** Enable query history (unchecked), Use local time (unchecked), Enable autocomplete (checked).
- View Selection:** Table (selected) and Graph.
- Evaluation Time:** Evaluation time range with arrows for navigation.
- Table Content:** A list of metric results. Each result is a line of text starting with `http_server_requests_seconds_count{...}`. The results are:
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="GET", outcome="CLIENT_ERROR", status="404", uri="/**"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="GET", outcome="SUCCESS", status="200", uri="/actuator/info"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="GET", outcome="SUCCESS", status="200", uri="/actuator/metrics"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="GET", outcome="SUCCESS", status="200", uri="/actuator/prometheus"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="GET", outcome="SUCCESS", status="200", uri="/users/{userId}"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="GET", outcome="SUCCESS", status="200", uri="/welcome"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="POST", outcome="SUCCESS", status="200", uri="root"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="POST", outcome="SUCCESS", status="201", uri="/users"}`
  - `http_server_requests_seconds_count{exception="None", instance="localhost:8000", job="user-service", method="POST", outcome="SUCCESS", status="204", uri="/actuator/busrefresh"}`
  - `http_server_requests_seconds_count{exception="RuntimeException", instance="localhost:8000", job="user-service", method="POST", outcome="SUCCESS", status="200", uri="root"}`

# Prometheus

## ■ metrics 검사 – Graph





# Grafana

## ■ Grafana 다운로드 – MacOS

grafana.com/grafana/download?platform=mac

Features Contribute Dashboards Plugins Download

Linux Windows Mac Docker ARM

OS X (via Homebrew)

```
brew update  
brew install grafana
```

Read the MacOS [installation guide](#) for more information.

Standalone MacOS/Darwin Binaries (64 Bit) SHA256: e8c75c2569b4464a801ba3adcdefcff

Read the MacOS [installation guide](#) for more information.

```
curl -O https://dl.grafana.com/oss/release/grafana-7.4.3.darwin-amd64.tar.gz  
tar -zxvf grafana-7.4.3.darwin-amd64.tar.gz
```

ls -l

```
total 48  
-rw-r--r--@ 1 downonlee staff 11343 2 24 20:45 LICENSE  
-rw-r--r--@ 1 downonlee staff 108 2 24 20:45 NOTICE.md  
-rw-r--r--@ 1 downonlee staff 2804 2 24 20:45 README.md  
-rw-r--r--@ 1 downonlee staff 5 2 24 20:56 VERSION  
drwxr-xr-x@ 6 downonlee staff 192 2 24 20:56 bin  
drwxr-xr-x@ 7 downonlee staff 224 2 24 20:56 conf  
drwxr-xr-x@ 3 downonlee staff 96 2 24 20:56 plugins-bundled  
drwxr-xr-x@ 13 downonlee staff 416 2 24 20:56 public  
drwxr-xr-x@ 29 downonlee staff 928 2 24 20:56 scripts
```

\$ ./bin/grafana-server web

# Grafana

## ■ Grafana 다운로드 – Windows

grafana.com/grafana/download?platform=windows

Features Contribute Dashboards Plugins [Download](#)

The [Enterprise Edition](#) includes all the features of the [Open Source Edition](#). All open source the full [paid Enterprise feature set](#), including support for [Enterprise plugins](#).

 Linux  Windows  Mac  Docker  ARM

Windows Installer (64 Bit) SHA256: 4be06da2e4fbbd90d8c0fa68d27ae64843a21e9a8a89.  
[Download the installer](#) (grafana-7.4.3.windows-amd64.msi) and run it.

Standalone Windows Binaries (64 Bit) SHA256: e422737a4bf97a826cbff408ef53b76b  
[Download the zip file](#) (grafana-7.4.3.windows-amd64.zip) and follow the instructions in the i

이름	수정한 날짜	유형
bin	2021-02-24 오후 8:56	파일 폴더
conf	2021-02-24 오후 8:56	파일 폴더
plugins-bundled	2021-02-24 오후 8:56	파일 폴더
public	2021-02-24 오후 8:56	파일 폴더
scripts	2021-02-24 오후 8:56	파일 폴더
tools	2021-02-24 오후 8:56	파일 폴더
LICENSE	2021-02-24 오후 8:45	파일
NOTICE.md	2021-02-24 오후 8:45	MD 파일
README.md	2021-02-24 오후 8:45	MD 파일
VERSION	2021-02-24 오후 8:56	파일

이름	수정한 날짜	유형
grafana-cli.exe	2021-02-24 오후 8:56	응용 프로그램
grafana-cli.exe.md5	2021-02-24 오후 8:56	MD5 파일
grafana-server.exe	2021-02-24 오후 8:56	응용 프로그램
grafana-server.exe.md5	2021-02-24 오후 8:56	MD5 파일

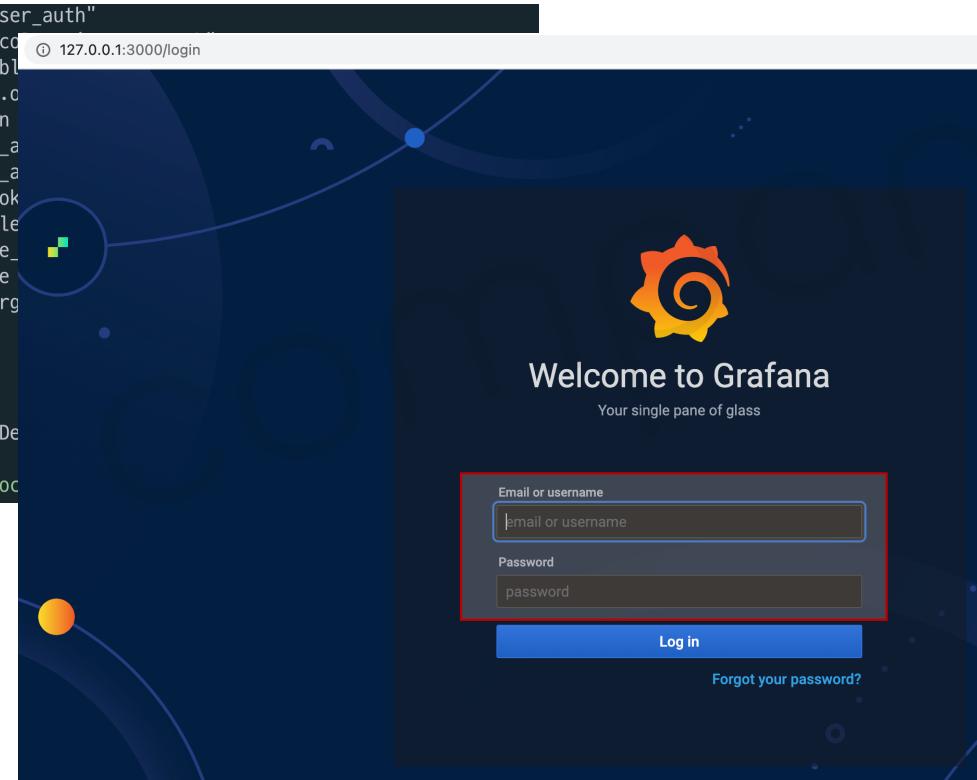
# Grafana

## ■ Grafana 실행

- <http://127.0.0.1:3000>
- ID: admin, PW: admin

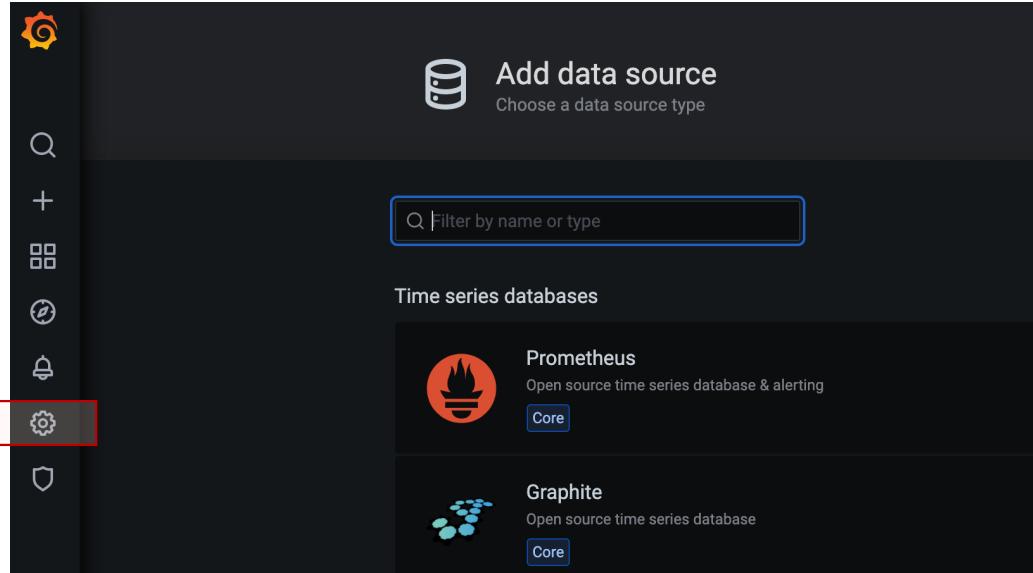
```
INFO[03-06|00:57:08] Executing migration  
INFO[03-06|00:57:08] Created default admin  
INFO[03-06|00:57:08] Created default organization  
INFO[03-06|00:57:08] Starting plugin search  
INFO[03-06|00:57:08] Registering plugin  
INFO[03-06|00:57:08] External plugins directory created  
ns  
INFO[03-06|00:57:08] HTTP Server Listen
```

```
logger=migrator id="Add OAuth expiry to user_auth"  
logger=migrator id="Add index to user_id co ① 127.0.0.1:3000/login  
logger=migrator id="create server_lock table  
logger=migrator id="add index server_lock.c  
logger=migrator id="create user auth token  
logger=migrator id="add unique index user_a  
logger=migrator id="add unique index user_a  
logger=migrator id="add index user_auth_tok  
logger=migrator id="create cache_data table  
logger=migrator id="add unique index cache_  
logger=migrator id="create short_url table  
logger=migrator id="add index short_url.org  
logger=sqlstore user=admin  
logger=sqlstore  
logger=plugins  
logger=plugins id=input  
logger=plugins directory=/Users/dwonlee/De  
  
logger=http.server address=[::]:3000 protoc
```



# Grafana

## ■ Grafana – Prometheus 연동



This screenshot shows the 'Data Sources / Prometheus' configuration screen. At the top, it says 'Type: Prometheus'. There are two tabs: 'Settings' (which is active) and 'Dashboards'. Below the tabs, a heading says 'Configure your Prometheus data source below' and provides a link to 'Grafana Cloud plan'. The 'HTTP' section contains fields for 'Name' (set to 'Prometheus'), 'URL' (set to 'http://127.0.0.1:9090'), 'Access' (set to 'Server (default)'), and 'Whitelisted Cookies' (with an 'Add Name' button). A 'Help' link is also present. At the bottom, there is a green box with a checkmark and the text 'Data source is working'. At the very bottom are three buttons: 'Save & Test' (blue), 'Delete' (red), and 'Back' (grey).

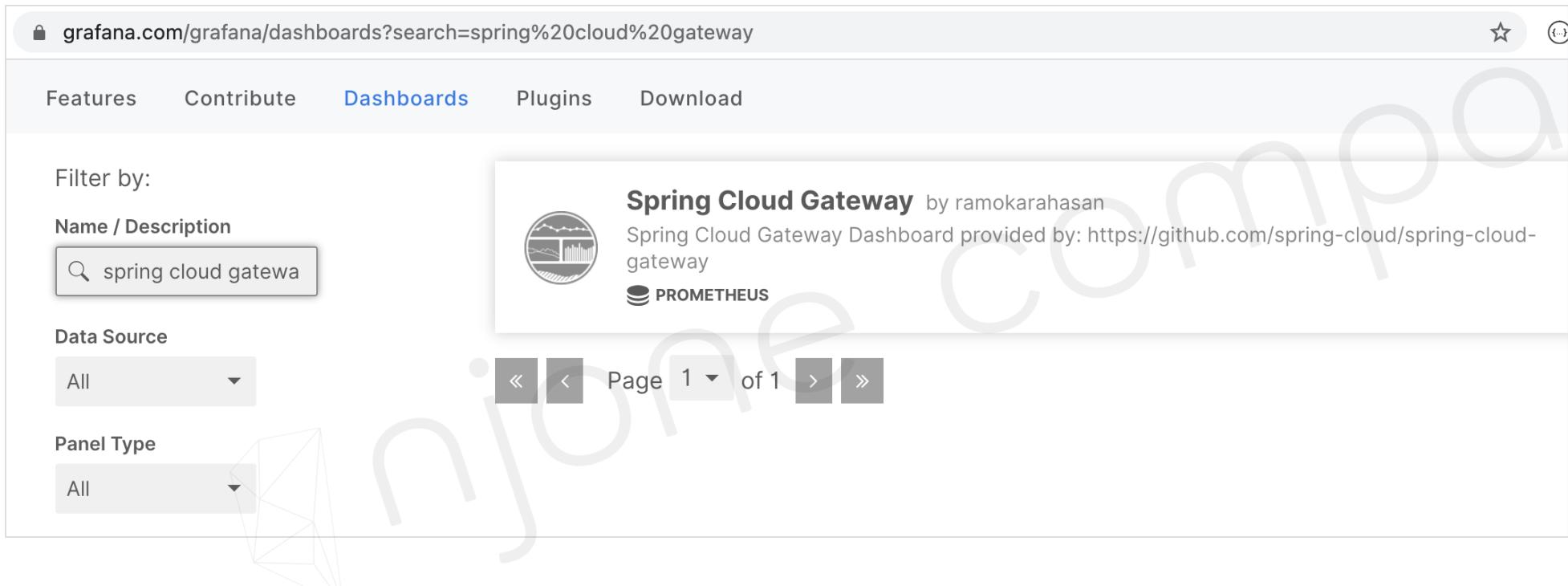




# Grafana

## ■ Grafana Dashboard

- JVM(Micrometer)
- Prometheus
- Spring Cloud Gateway



The screenshot shows a web browser displaying the Grafana dashboard search results at [grafana.com/grafana/dashboards?search=spring%20cloud%20gateway](https://grafana.com/grafana/dashboards?search=spring%20cloud%20gateway). The page has a navigation bar with links for Features, Contribute, Dashboards (which is highlighted in blue), Plugins, and Download. On the left, there are filters for Name / Description (with a search input containing "spring cloud gateway") and Data Source (set to All). Below these filters is a Panel Type dropdown set to All. In the center, a single dashboard card is displayed for "Spring Cloud Gateway" by ramokarahaasan. The card includes a circular icon with a mountain and chart graphic, the title "Spring Cloud Gateway", the author "ramokarahaasan", a description "Spring Cloud Gateway Dashboard provided by: <https://github.com/spring-cloud/spring-cloud-gateway>", and a Prometheus logo. At the bottom of the page, there is a pagination control showing "Page 1 of 1".

# Grafana

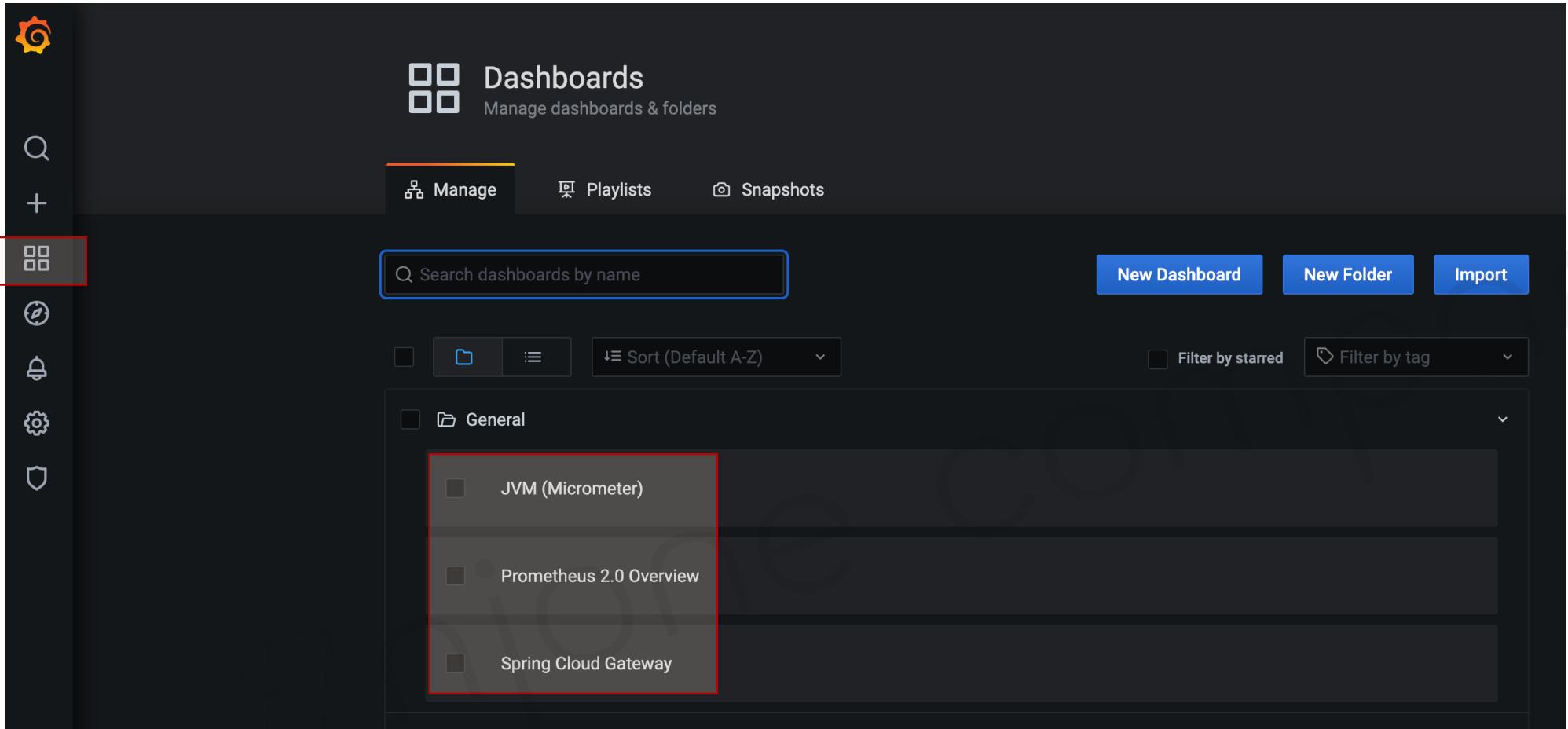
## ■ Grafana – Prometheus 연동

The image shows two screenshots of the Grafana interface. On the left, the main 'Import' screen is displayed, featuring a sidebar with various icons and a central area with two main import options: 'Import via grafana.com' and 'Import via panel json'. The 'Import via grafana.com' section contains a text input field for 'Grafana.com dashboard url or id' and a 'Load' button. A red box highlights the '+' icon in the sidebar and the 'Grafana.com dashboard url or id' input field. On the right, a detailed 'Importing Dashboard from Grafana.com' dialog is shown. It includes fields for 'Published by' (Jin Shang), 'Updated on' (2018-11-10 14:40:07), and 'Options' for 'Name' (JVM (Micrometer)) and 'Folder' (General). A 'Unique identifier (uid)' section explains that uid allows for consistent URLs. A dropdown menu for 'Prometheus' is highlighted with a red box, showing 'Prometheus' selected. At the bottom are 'Import' and 'Cancel' buttons.



# Grafana

## ■ Grafana Dashboard



The screenshot shows the Grafana dashboard management interface. On the left is a sidebar with icons for Dashboards, Playlists, Snapshots, Search, and Create. The main area has a title "Dashboards" with a subtitle "Manage dashboards & folders". It includes tabs for "Manage", "Playlists", and "Snapshots", and buttons for "New Dashboard", "New Folder", and "Import". A search bar "Search dashboards by name" is present. Below these are filters for "Sort (Default A-Z)", "Filter by starred", and "Filter by tag". The main content area displays a list of dashboards under a "General" folder. Three specific dashboards are highlighted with a red box: "JVM (Micrometer)", "Prometheus 2.0 Overview", and "Spring Cloud Gateway".

Dashboard
JVM (Micrometer)
Prometheus 2.0 Overview
Spring Cloud Gateway

# Grafana

## ■ Spring Cloud Gateway dashboard

