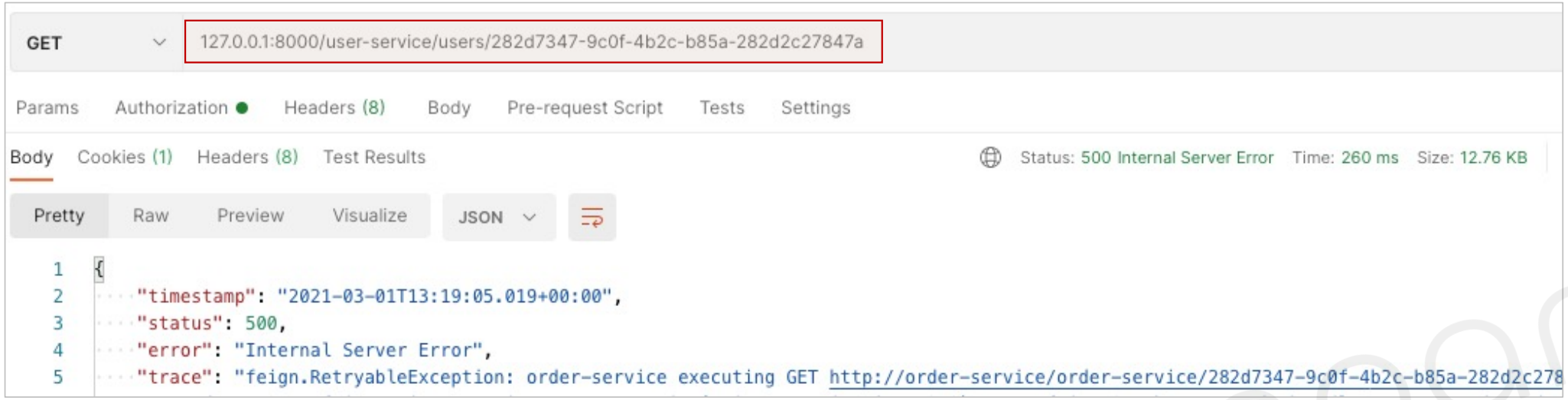
Spring Cloud로 개발하는 마이크로서비스 애플리케이션

Section 4.
분산 추적

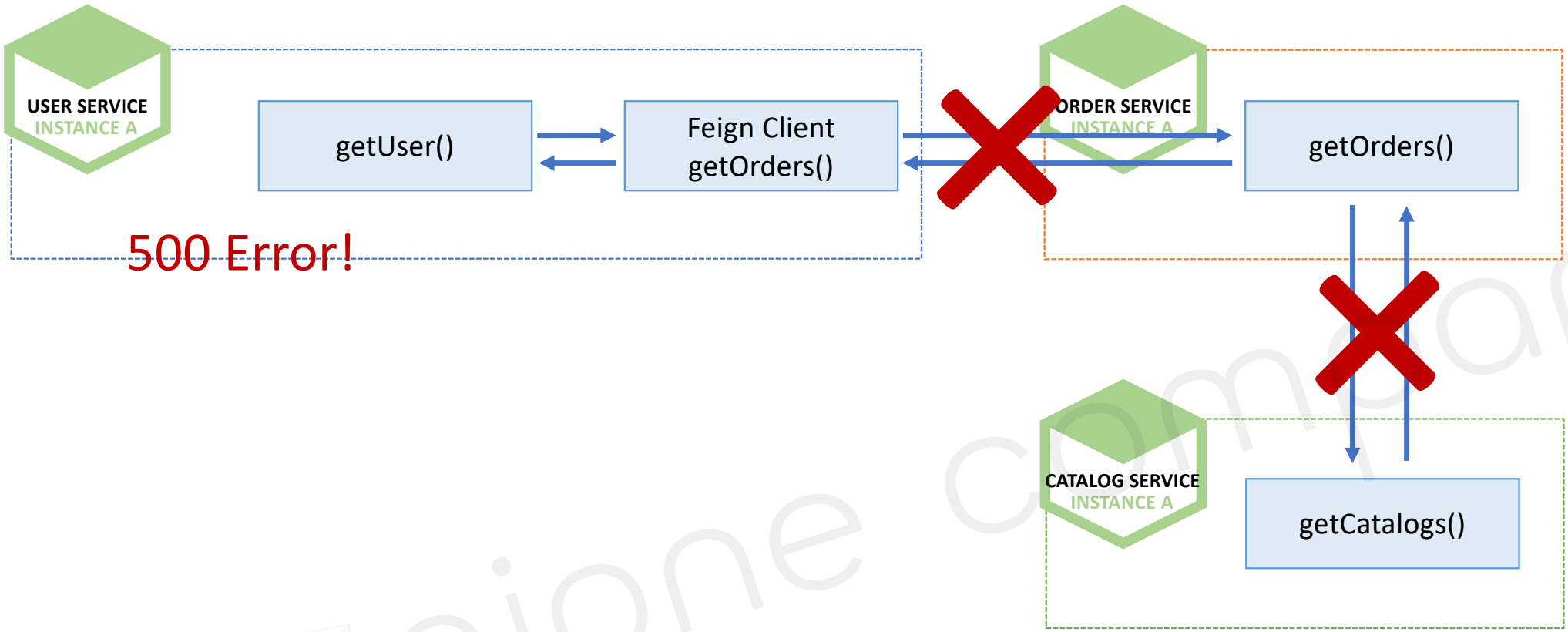
- CircuitBreaker
- Resilience4j
- Distributed Tracing
- Trace ID and Span ID
- Zipkin server 활용
- Monitoring

Microservice 통신 시 연쇄 오류

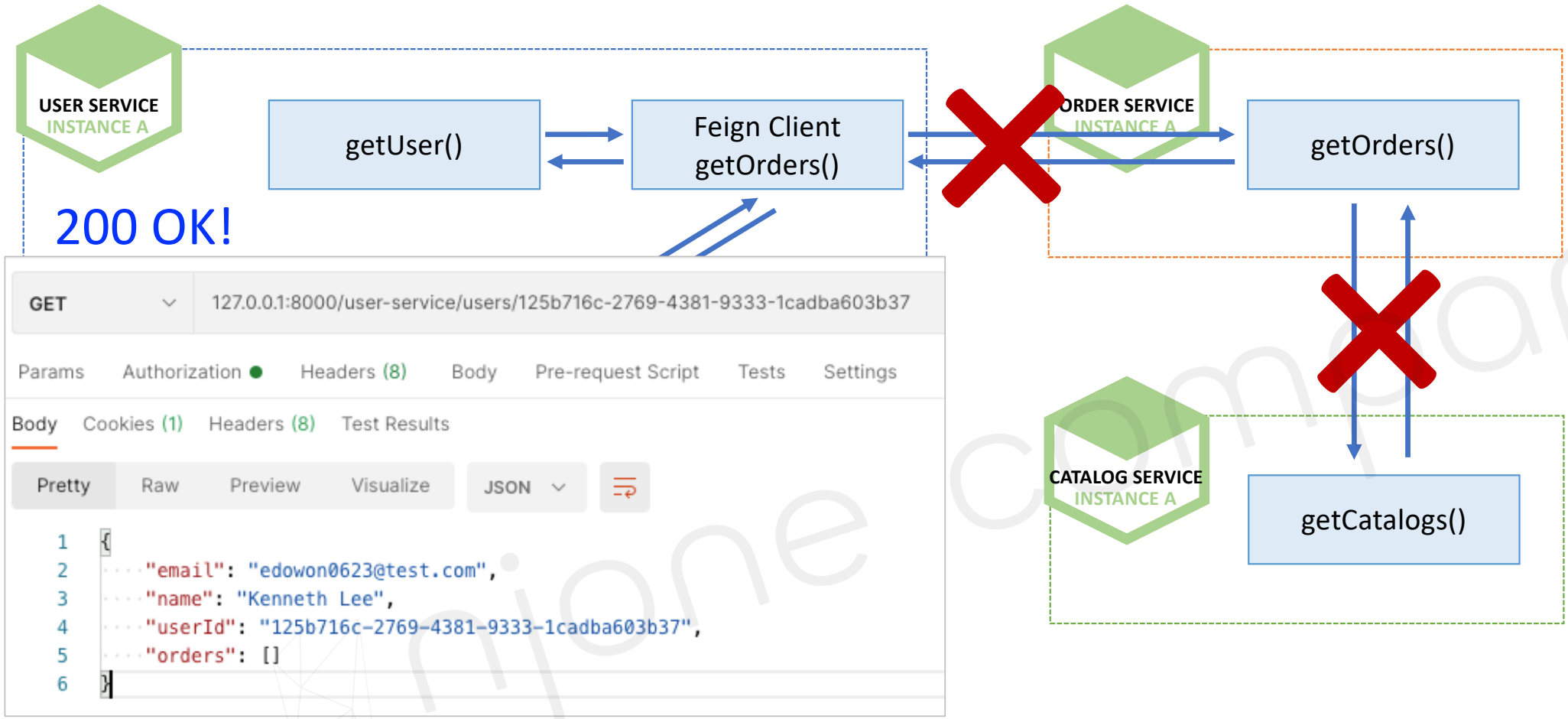



```
java.net.UnknownHostException Create breakpoint : order-service
    at java.base/sun.nio.ch.NioSocketImpl.connect(NioSocketImpl.java:567) ~[na:na]
    at java.base/java.net.Socket.connect(Socket.java:648) ~[na:na]
    at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:177) ~[na:na]
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

Microservice 통신 시 연쇄 오류

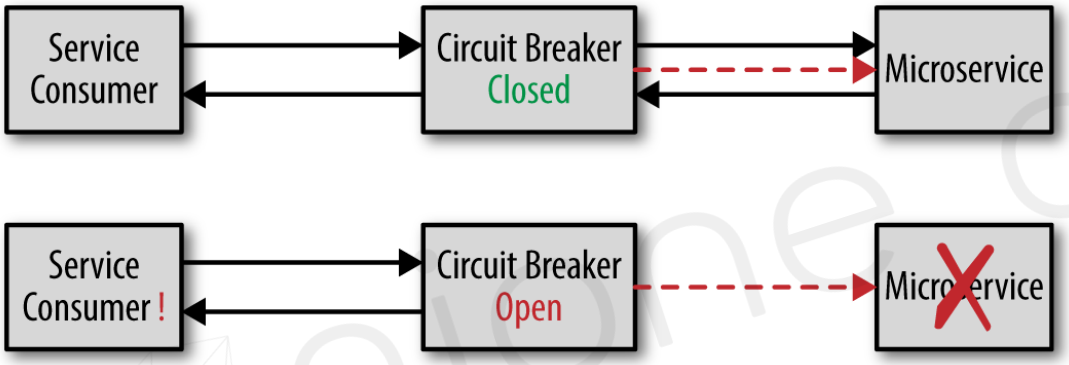


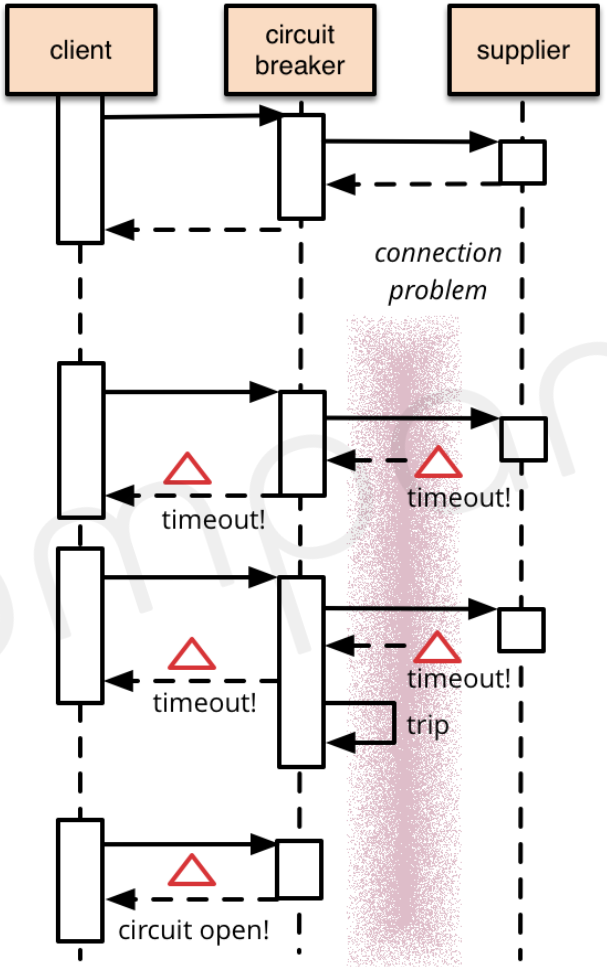
Microservice 통신 시 연쇄 오류



CircuitBreaker

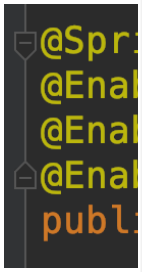
- CircuitBreaker
  - https://martinfowler.com/bliki/CircuitBreaker.html
  - 장애가 발생하는 서비스에 반복적인 호출이 되지 못하게 차단
  - 특정 서비스가 정상적으로 동작하지 않을 경우 다른 기능으로 대체 수행 → 장애 회피





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

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

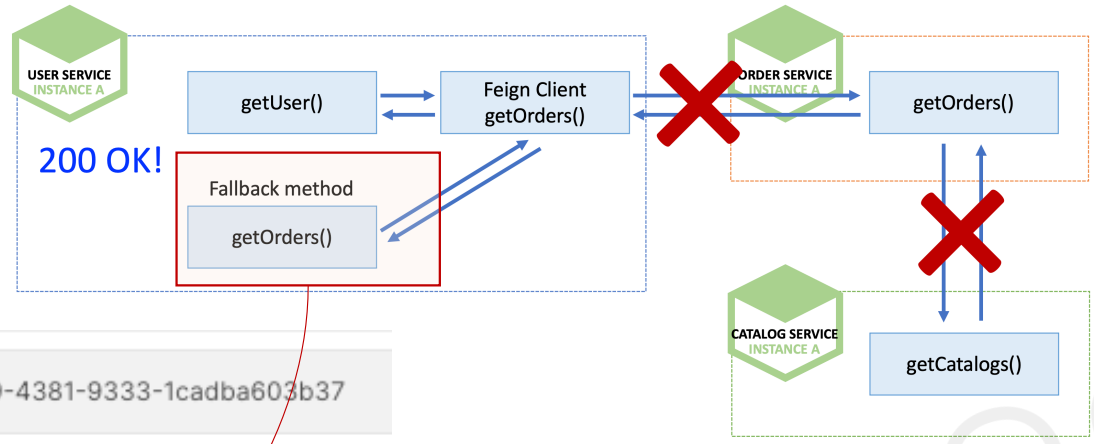
pom.xml

```
@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



GET 127.0.0.1:8000/user-service/users/125b716c-2769-4381-9333-1cadba603b37

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

Body Cookies (1) Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "email": "edowon0623@test.com",
    "name": "Kenneth Lee",
    "userId": "125b716c-2769-4381-9333-1cadba603b37",
    "orders": []
}
```

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();
```

- CirucitBreaker를 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

- CircuitBreaker가 종료(Close) 되기 위한 조건을 결정하는 타입
- COUNT_BASED or TIME_BASED

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

Resilience4j – CircuitBreaker

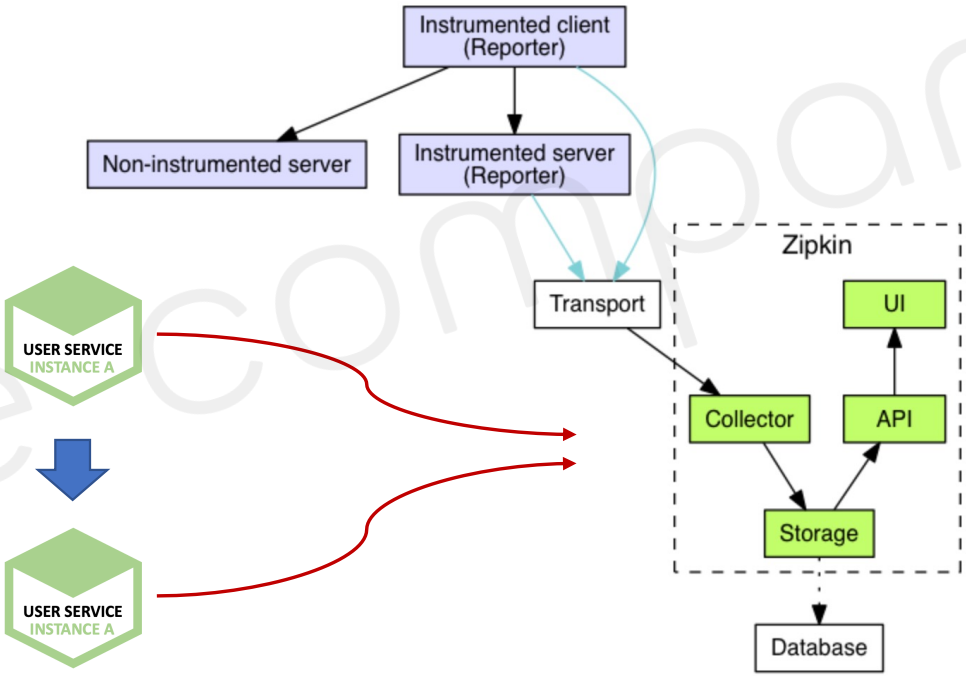
- Customize CircuitBreakerFactory → ***Resilience4JCircuitBreakerFactory***

```
        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

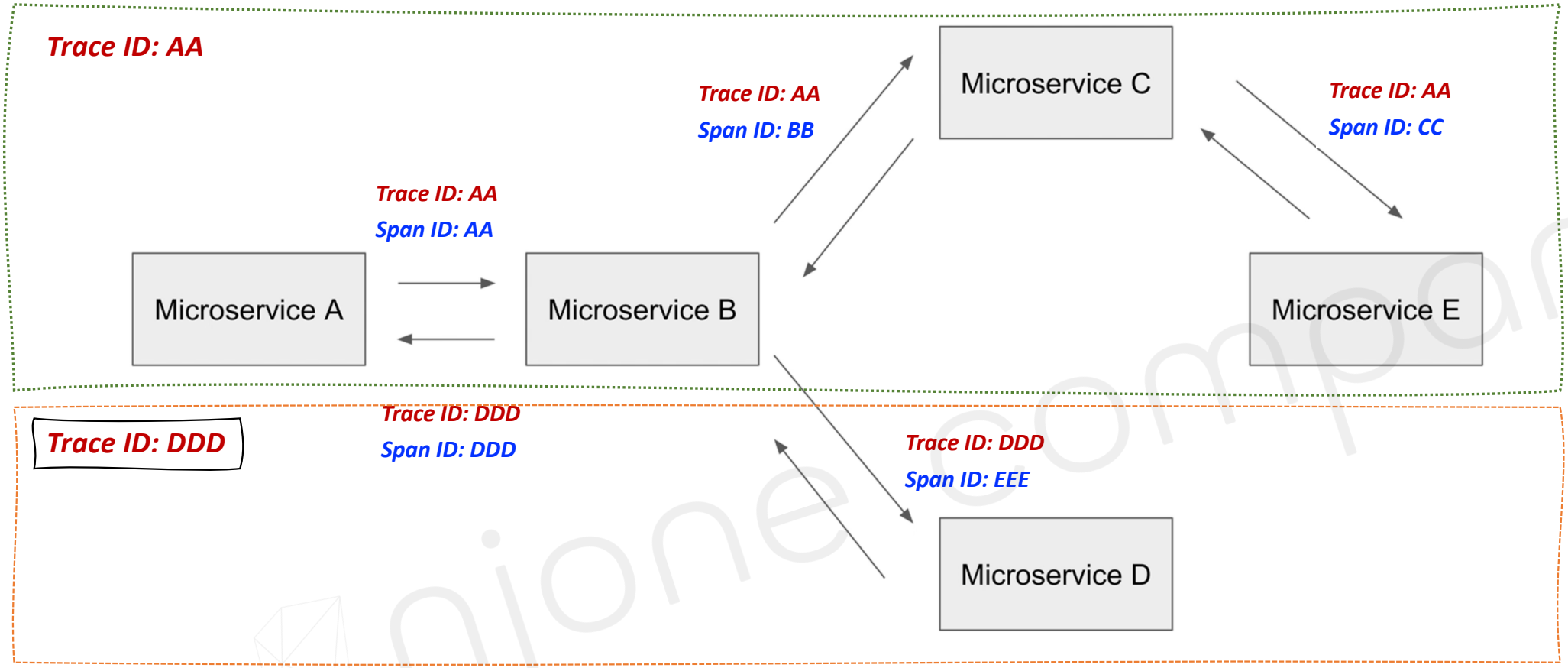
Sleuth configures everything you need to get started. This includes where trace data (spans) are reported to, how many traces to keep (sampling), if remote fields (baggage) are sent, and which libraries are traced.

Specifically, Spring Cloud Sleuth...

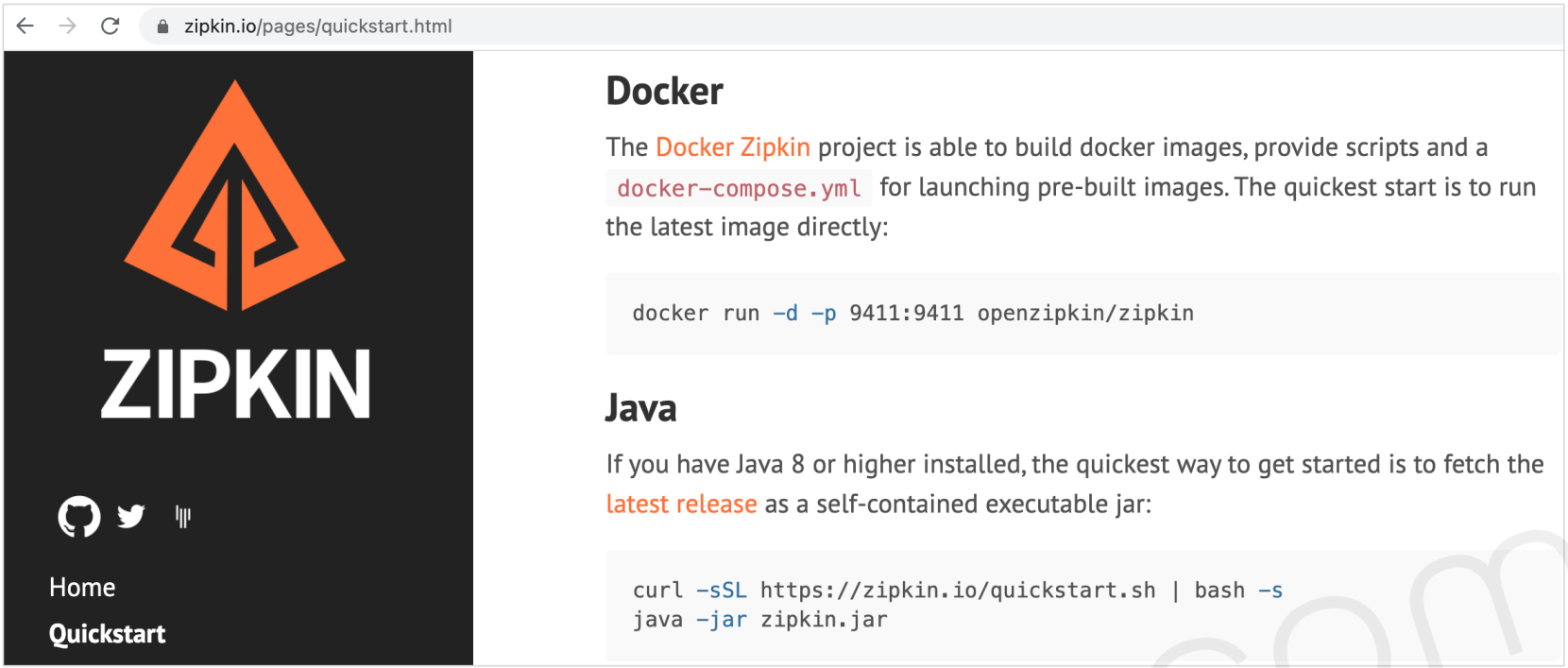
- Adds trace and span ids to the Slf4J MDC, so you can extract all the logs from a given trace or span in a log aggregator.
- Instruments common ingress and egress points from Spring applications (servlet filter, rest template, scheduled actions, message channels, feign client).
- If `spring-cloud-sleuth-zipkin` 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

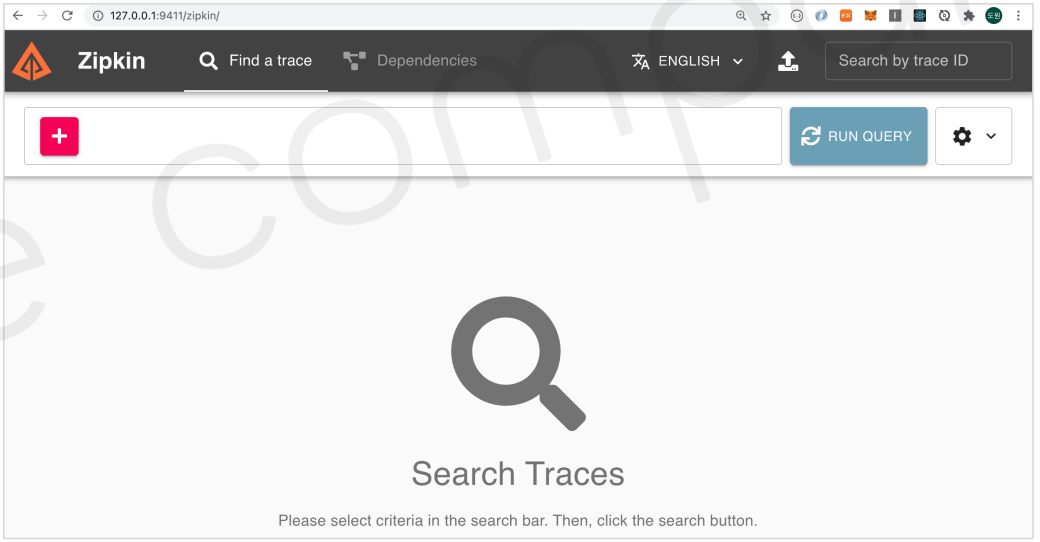



```
curl -sSL https://zipkin.io/quickstart.sh | bash -s
Thank you for trying Zipkin!
This installer is provided as a quick-start helper, so you can try Zipkin out
without a lengthy installation process.

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

```
java -jar zipkin.jar
```

```
                  oo
                 oooo
                oooooo
               oooooooo
              oooooooooo
             oooooooooooo
           oooooooo  oooooooo
          oooooo      oooooooo
         oooooo        oooooooo
        oooooo    o  o    oooooo
       oooooo    oo  oo    oooooo
     ooooooo  oooo  oooo  ooooooo
    oooooo    ooooo  ooooo  ooooooo
   oooooo    oooooo  oooooo  ooooooo
  oooooooo      oo  oo      oooooooo
  ooooooooooooo oo  oo ooooooooooooo
     oooooooooooo    oooooooooooo
         oooooooo  oooooooo
             oooo  oooo


   _______ ____   _  _____ _   _
  |__   /_ _|  _ \| |/ /_ _| \ | |
    / / | || |_) | ' / | ||  \| |
   / /_ | ||  __/| . \ | || |\  |
  |____|___|_|   |_|\_\___|_| \_|

:: version 2.23.2 :: commit 7bf3aab ::

2021-03-02 01:14:25.648  INFO [/] 87530 --- [oss-http-*:9411] c.l.a.s.Server
                : Serving HTTP at /[0:0:0:0:0:0:0:0]:9411 - http://127.0.0.1:9411/
```



Users Microservice 수정

```xml
<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

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

application.yml

```java
/* 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

```
{
    "productId": "CATALOG-003",
    "qty": 5,
    "unitPrice": 1500
}
```

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

Pretty Raw Preview Visualize JSON

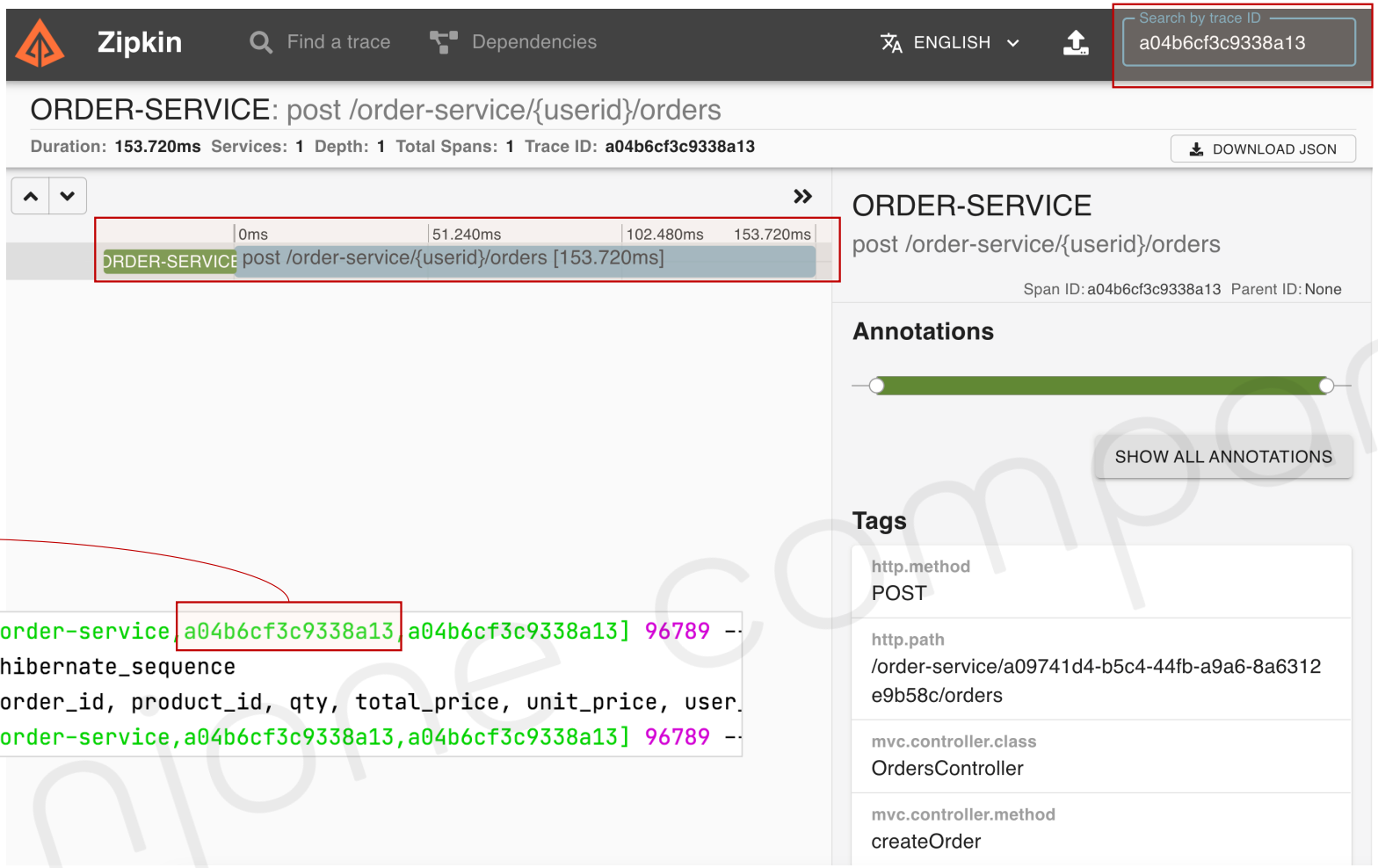
```
{
    "productId": "CATALOG-003",
    "qty": 5,
    "unitPrice": 1500,
    "totalPrice": 7500,
    "orderId": "6f3bed3b-0ba4-483b-83b1-2358b3bba500"
}
```

Trace ID Span 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
2021-03-02 23:47:35.413  INFO [order-service,a04b6cf3c9338a13,a04b6cf3c9338a13] 96789 --
```

```
[o-auto-1-exec-1] c.e.o.controller.OrdersController        : Before add orders data

d, id) values (?, ?, ?, ?, ?, ?, ?)
[o-auto-1-exec-1] c.e.o.controller.OrdersController        : After added orders data
```

Trace ID and Span ID



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

Trace ID and Span ID

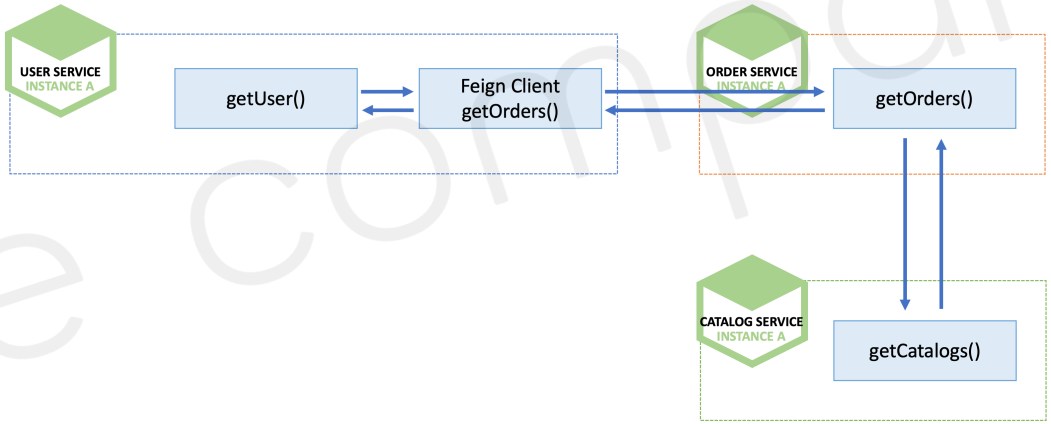
GET 127.0.0.1:8000/user-service/users/a09741d4-b5c4-44fb-a9a6-8a6312e9b58c

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

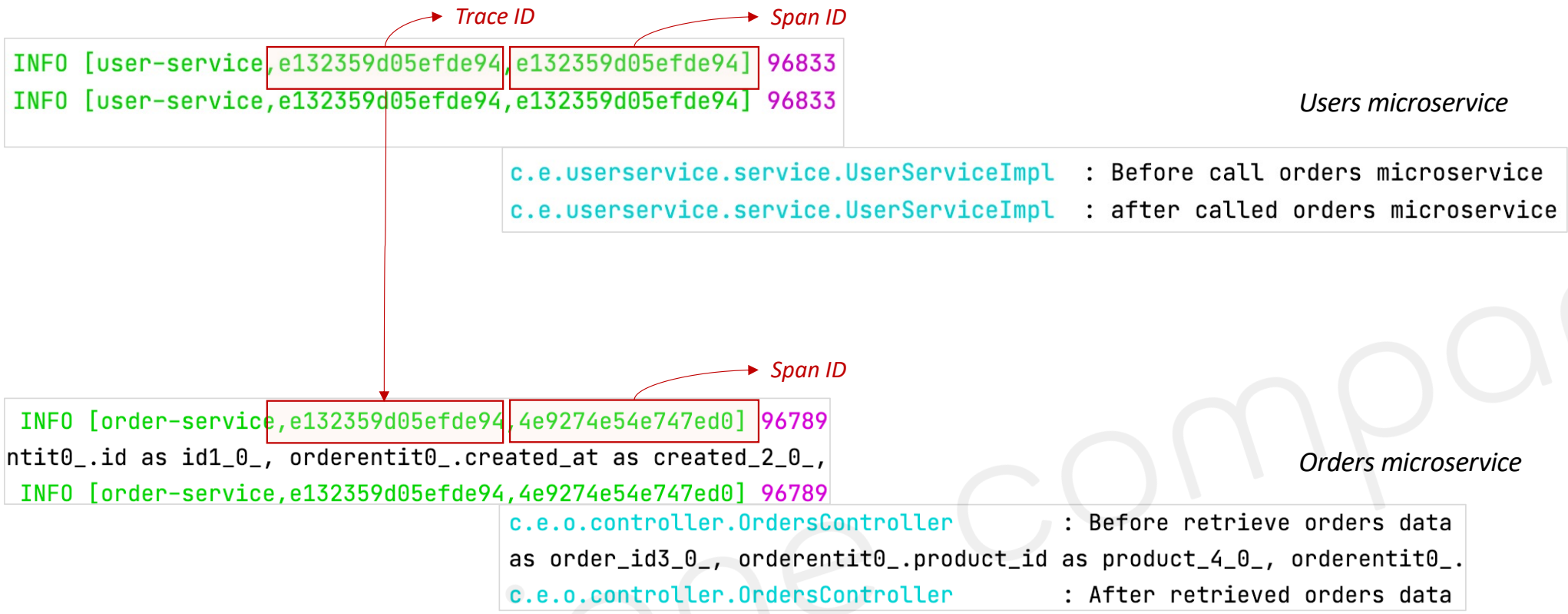
Body Cookies (1) Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

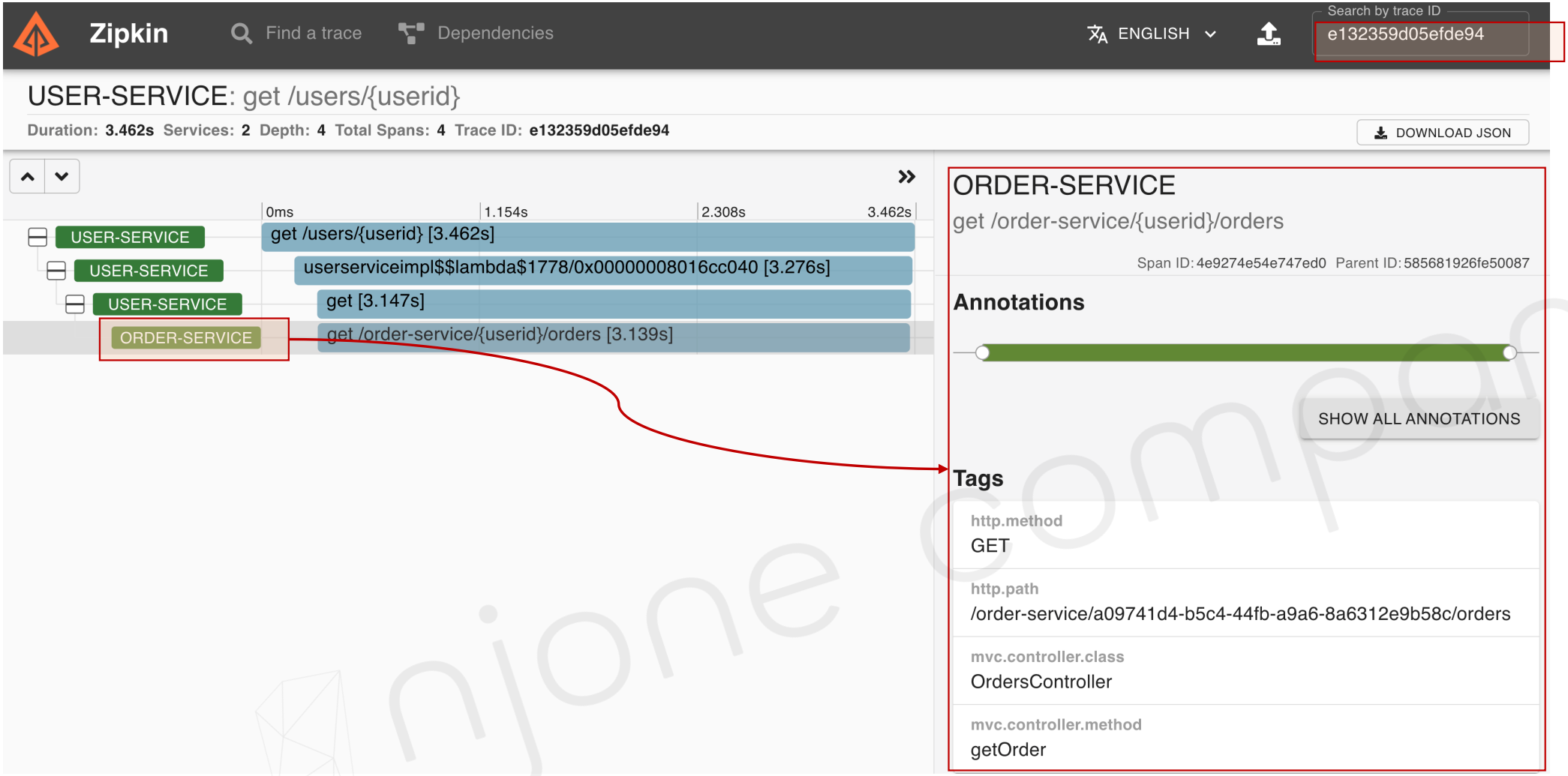

```
{
    "email": "edowon0623@test.com",
    "name": "Kenneth Lee",
    "userId": "a09741d4-b5c4-44fb-a9a6-8a6312e9b58c",
    "orders": [
        {
            "productId": "CATALOG-004",
            "qty": 10,
            "unitPrice": 1500,
            "totalPrice": 15000,
            "createdAt": "2021-03-02T14:47:35.406+00:00",
            "orderId": "73b93f3d-8dee-4bd0-9a23-05d465be75f9"
        },
```

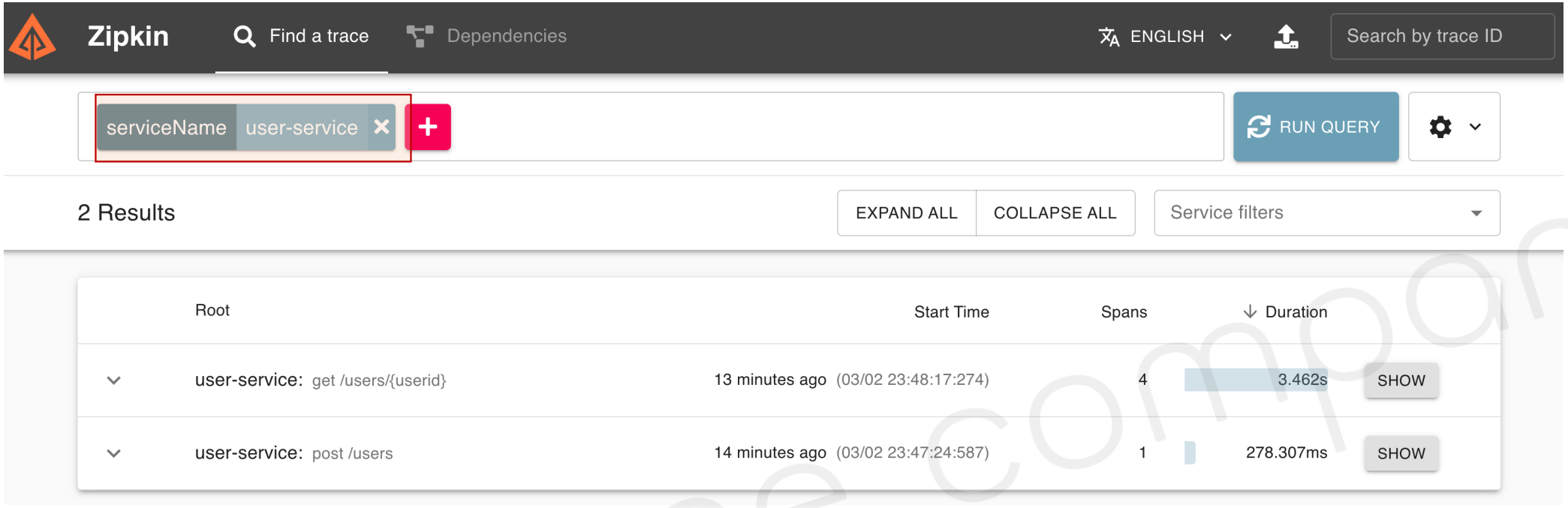
Trace ID and Span ID



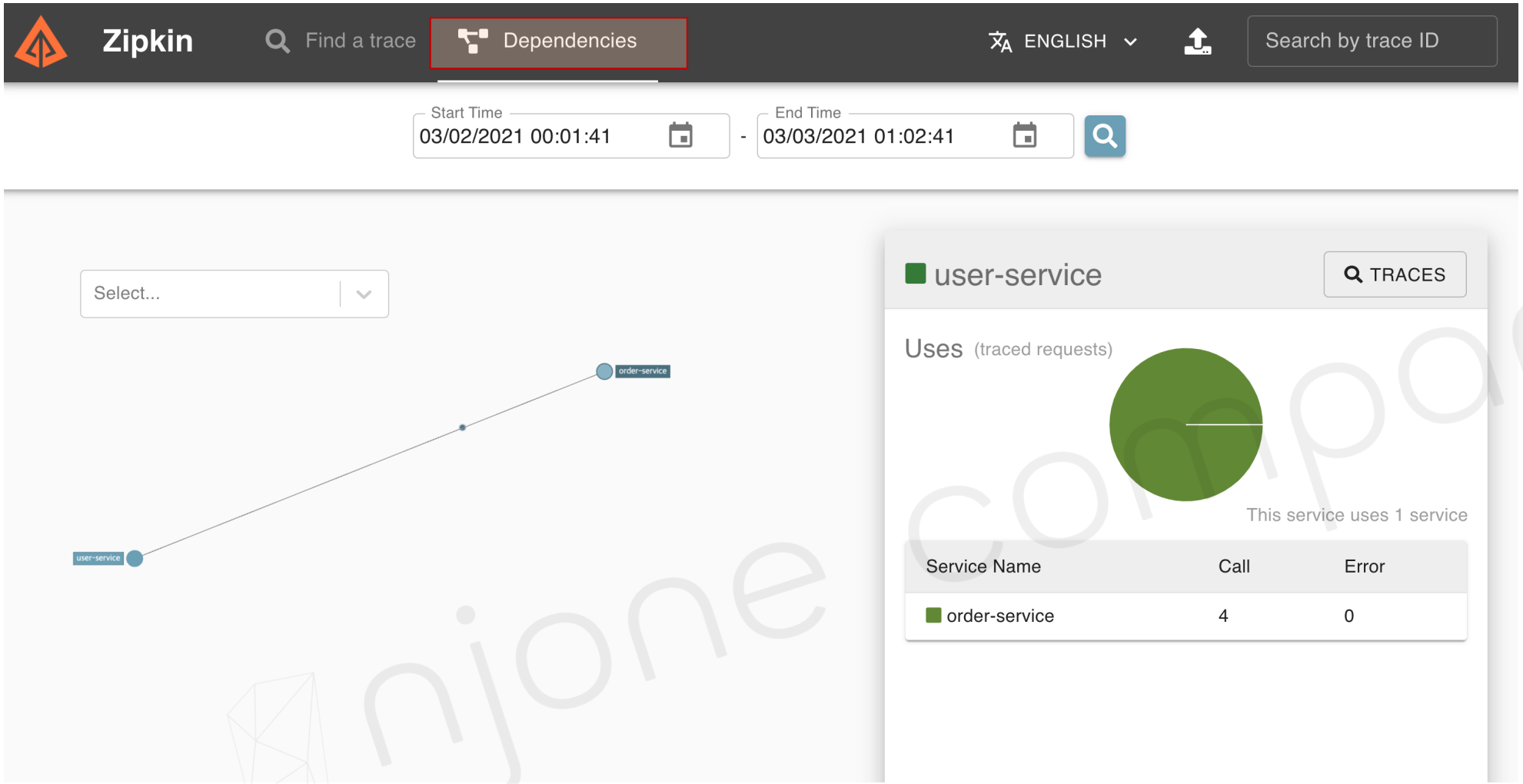
Zipkin server 활용



Zipkin server 활용



Zipkin 에서의 추적

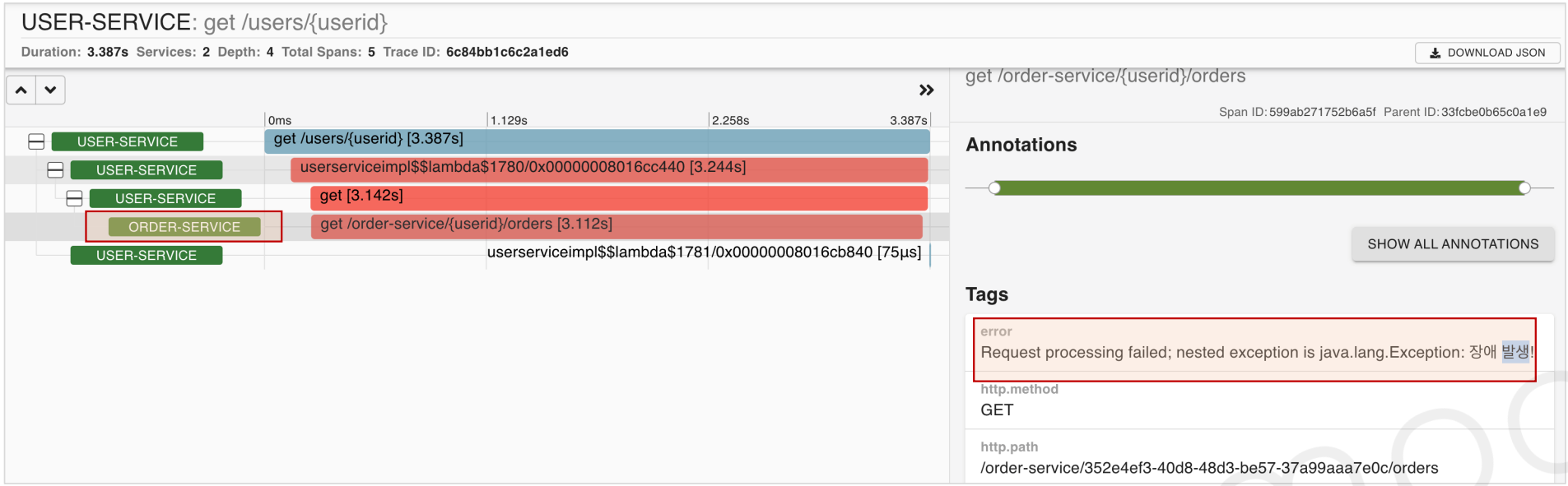


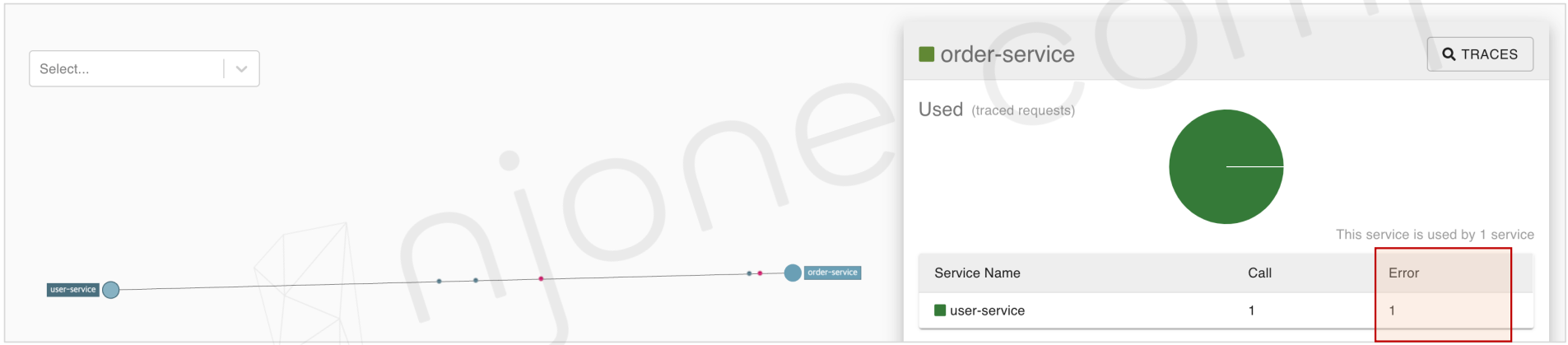
Orders Microservice 장애 발생

```
@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

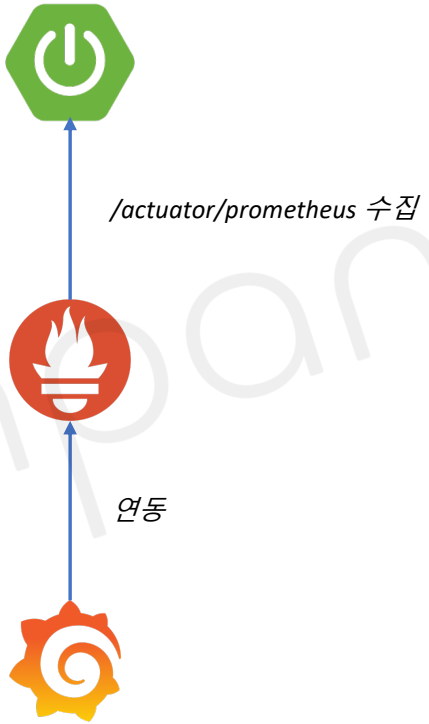
Orders Microservice 장애 발생





Prometheus + Grafana

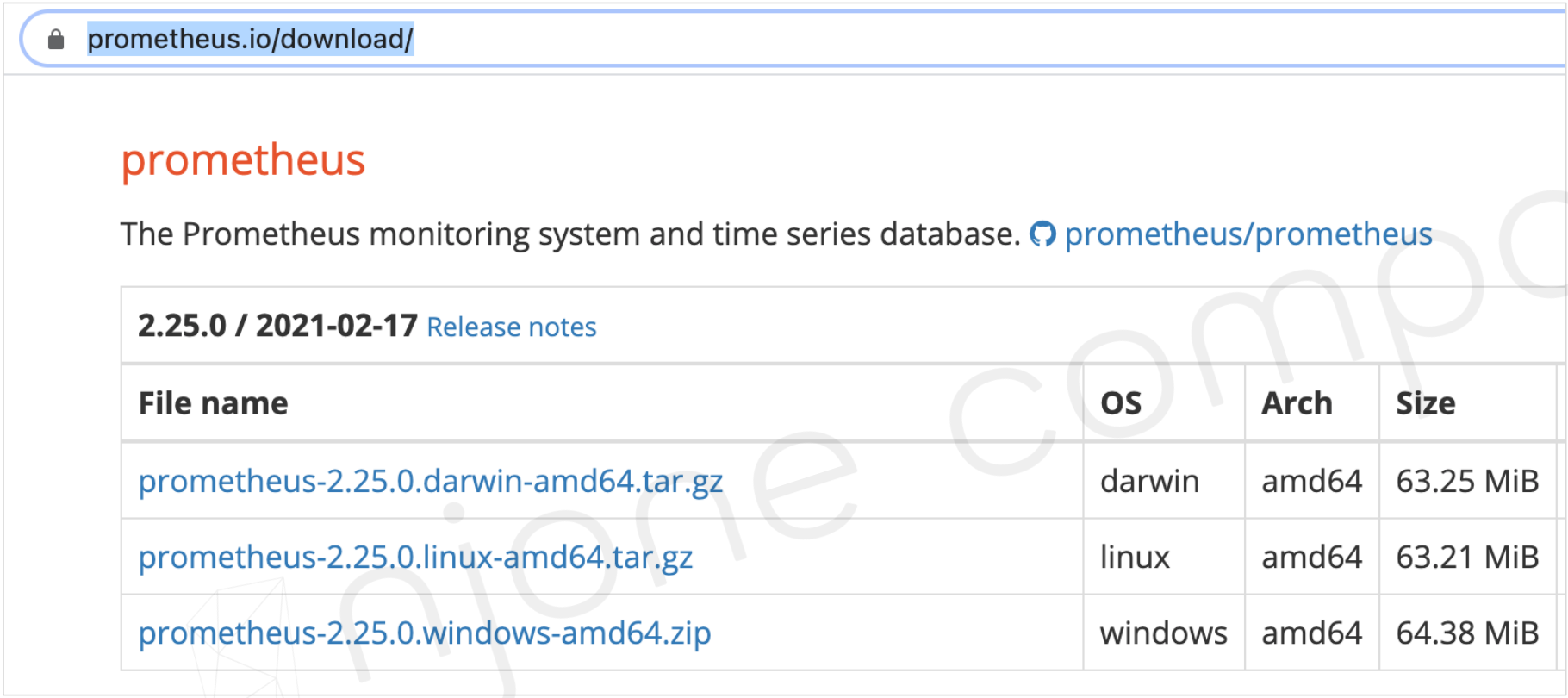
- Prometheus
  - Metrics를 수집하고 모니터링 및 알람에 사용되는 오픈소스 애플리케이션
  - 2016년부터 CNCF에서 관리되는 2번째 공식 프로젝트
    - Level DB → Time Series Database(TSDB)
  - Pull 방식의 구조와 다양한 Metric Exporter 제공
  - 시계열 DB에 Metrics 저장 → 조회 가능 (Query)



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

Prometheus 설치

- Prometheus 다운로드
  - https://prometheus.io/download/



Prometheus

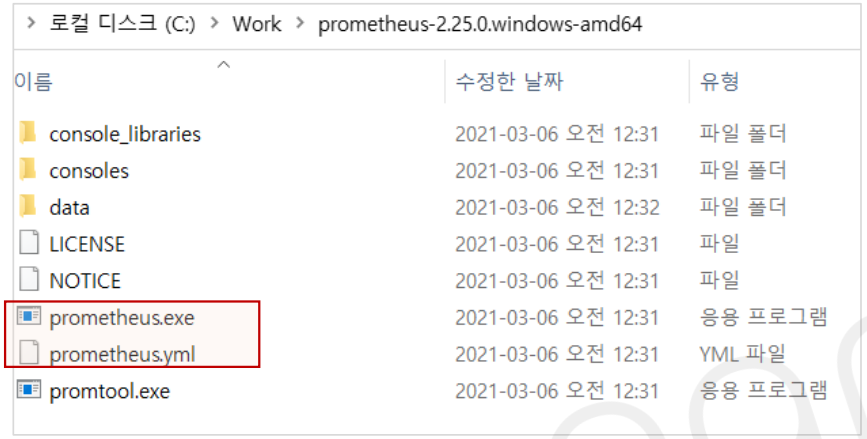
- prometheus.yml 파일 수정
  - target 지정

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

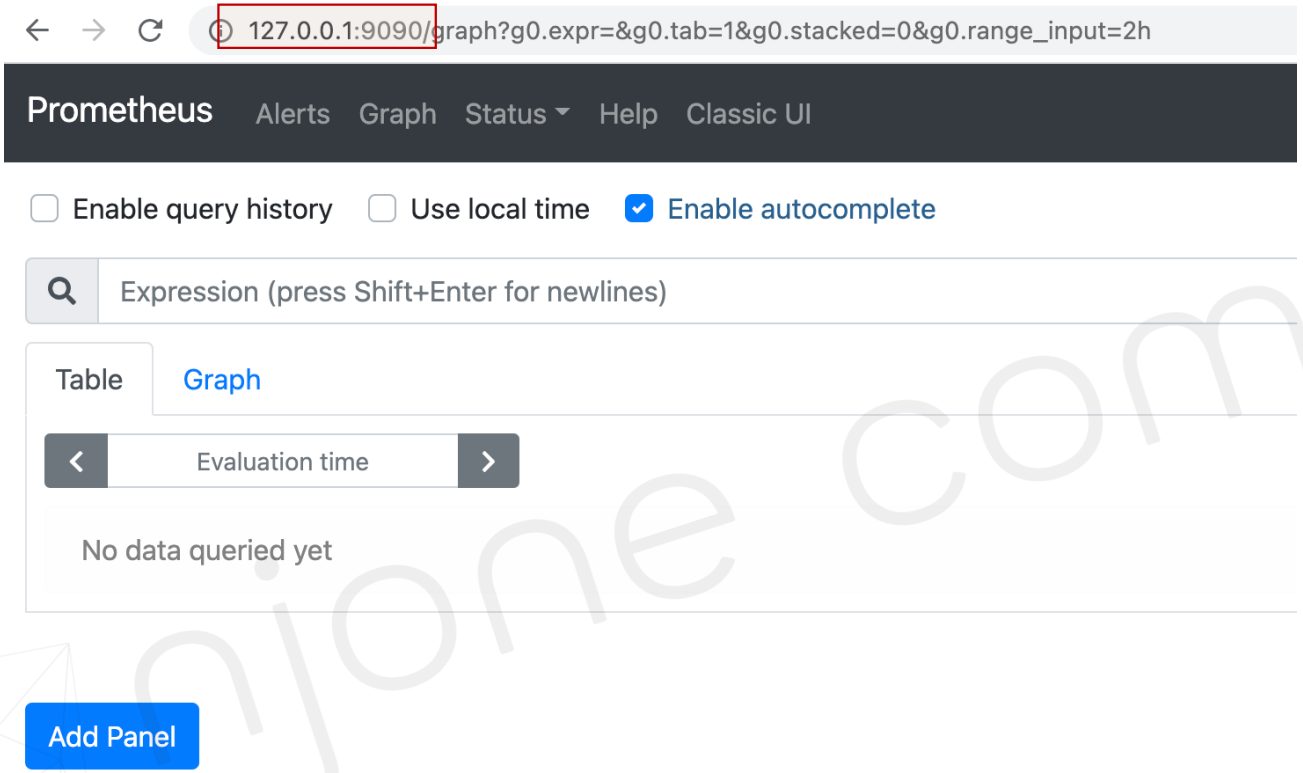


$ ./prometheus --config.file=prometheus.yml

```
./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 retentior
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
```

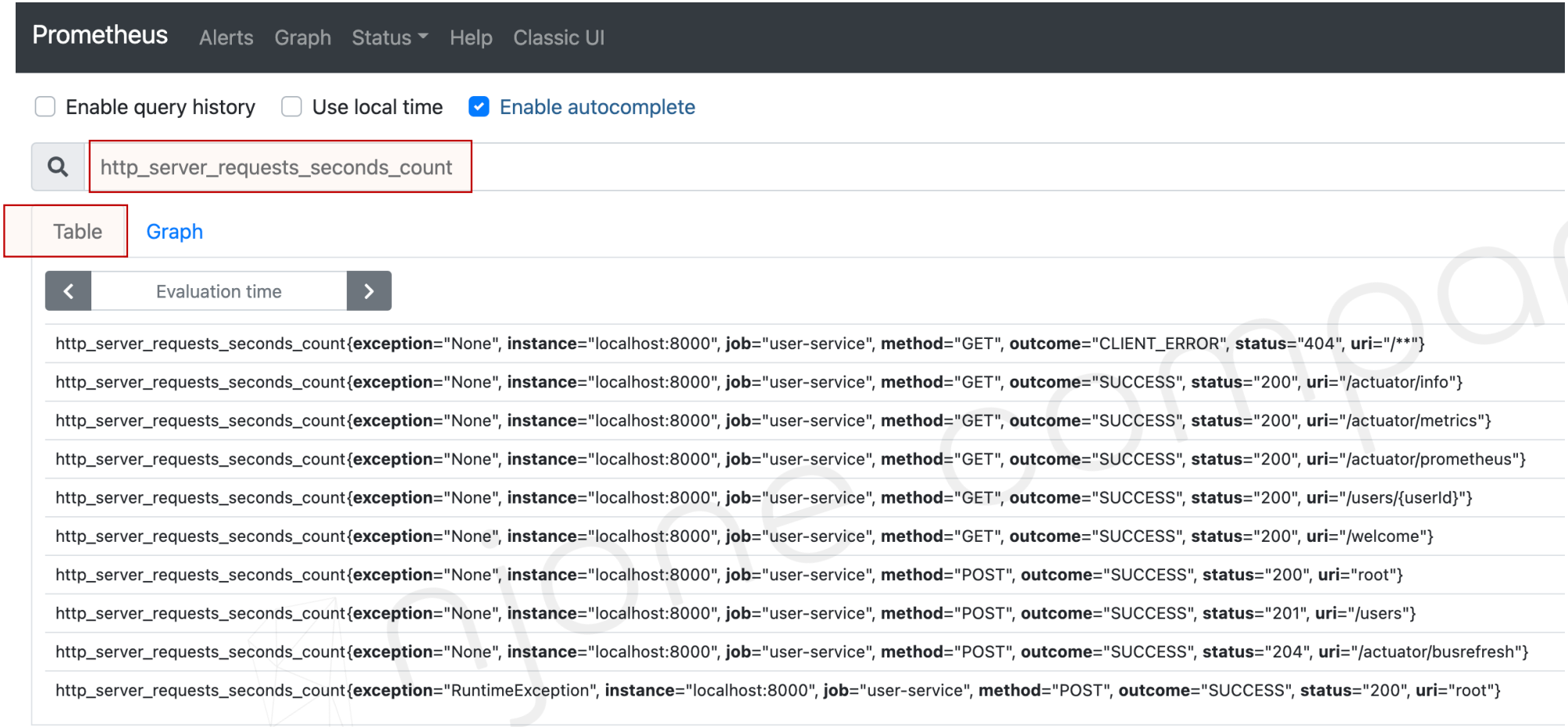
Prometheus

- Prometheus Dashboard
  - http://127.0.0.1:9090



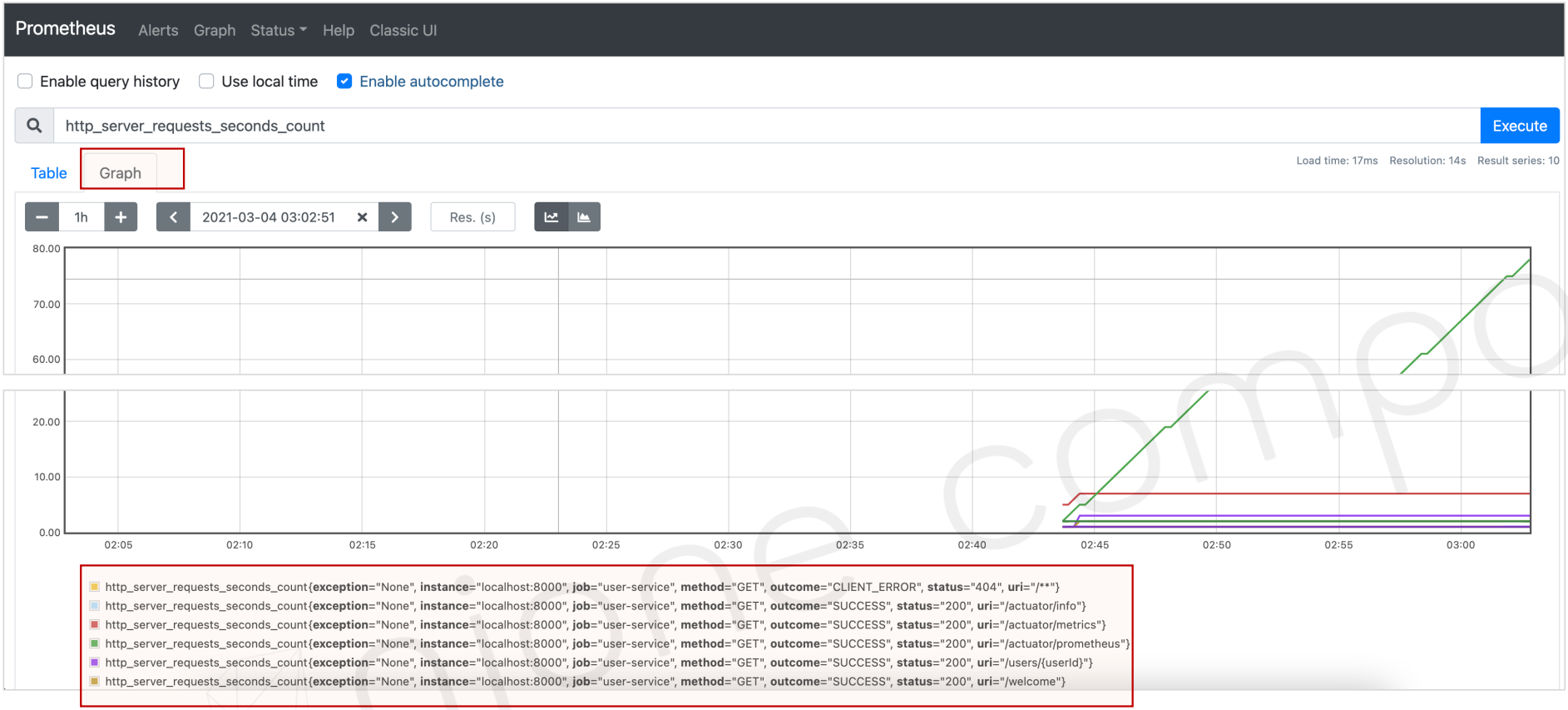
Prometheus

- metrics 검사 – Table



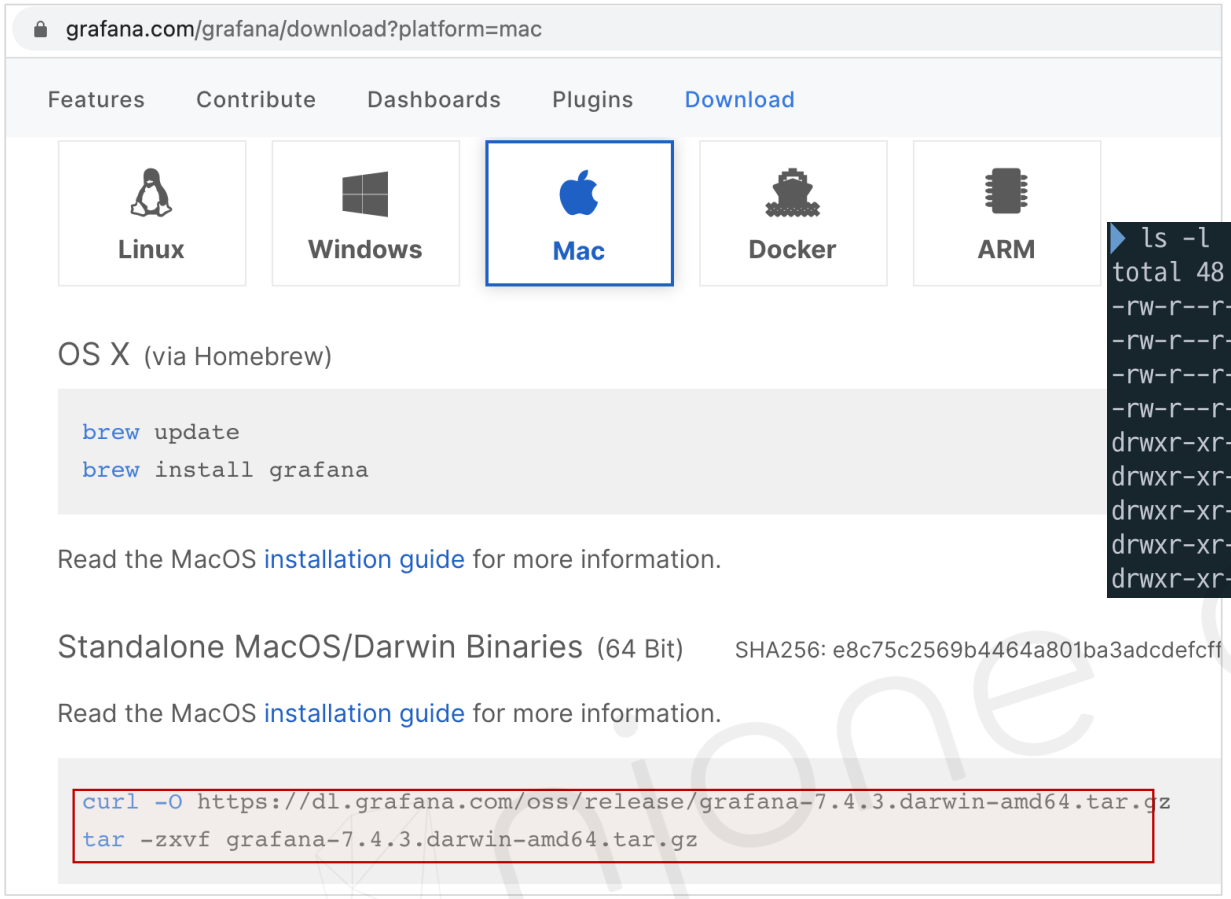
Prometheus

- metrics 검사 – Graph



Grafana

- Grafana 다운로드 – MacOS

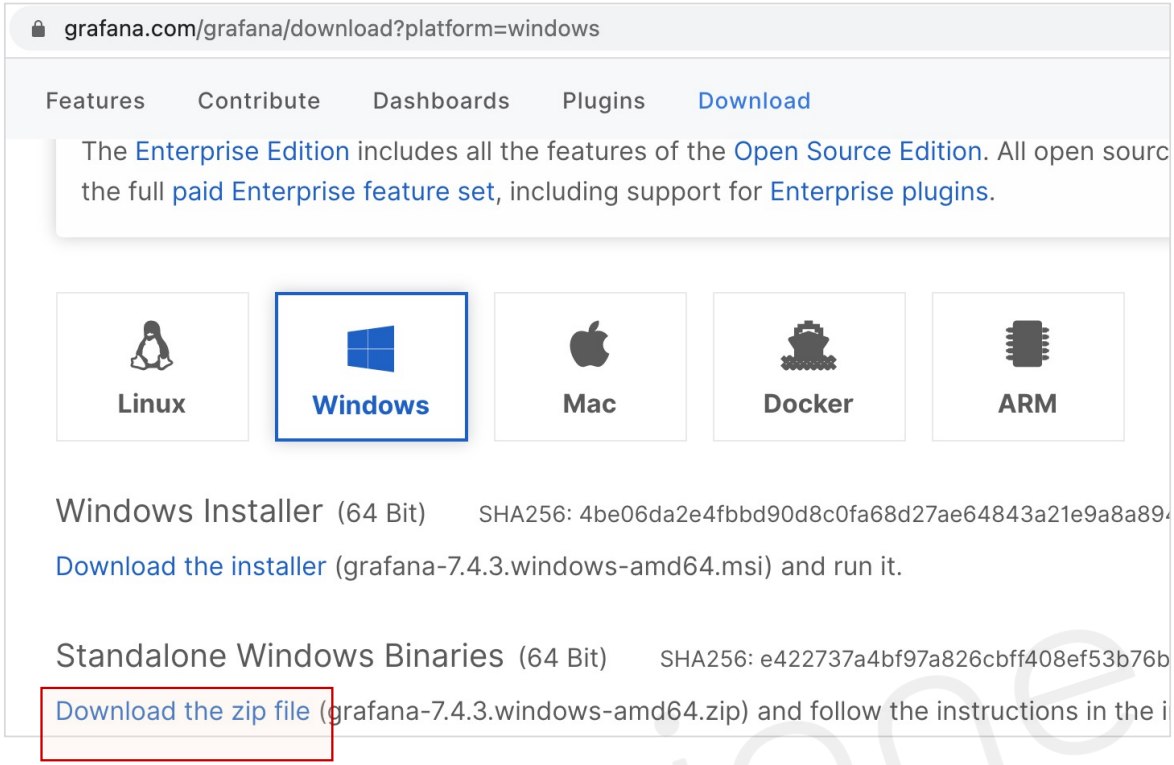


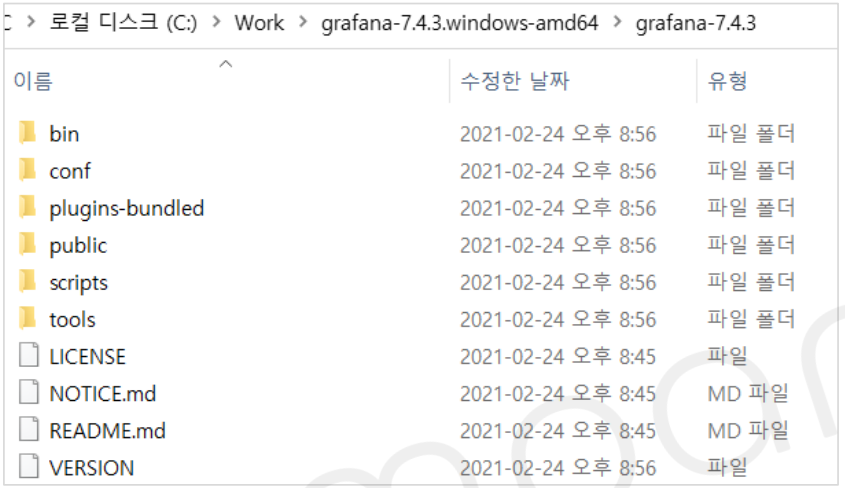

```
ls -l
total 48
-rw-r--r--@  1 dowonlee  staff  11343  2 24 20:45 LICENSE
-rw-r--r--@  1 dowonlee  staff    108  2 24 20:45 NOTICE.md
-rw-r--r--@  1 dowonlee  staff   2804  2 24 20:45 README.md
-rw-r--r--@  1 dowonlee  staff      5  2 24 20:56 VERSION
drwxr-xr-x@  6 dowonlee  staff    192  2 24 20:56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

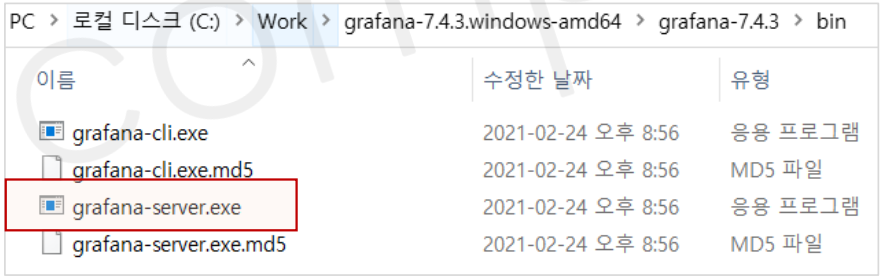
$ ./bin/grafana-server web

Grafana

- Grafana 다운로드 – Windows



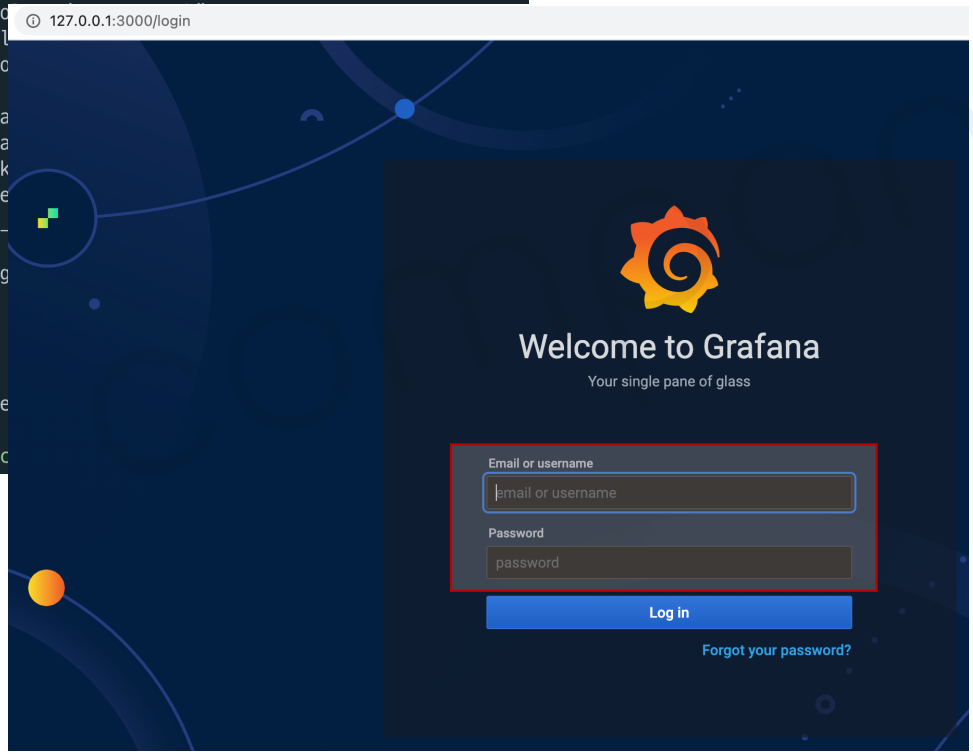




Grafana

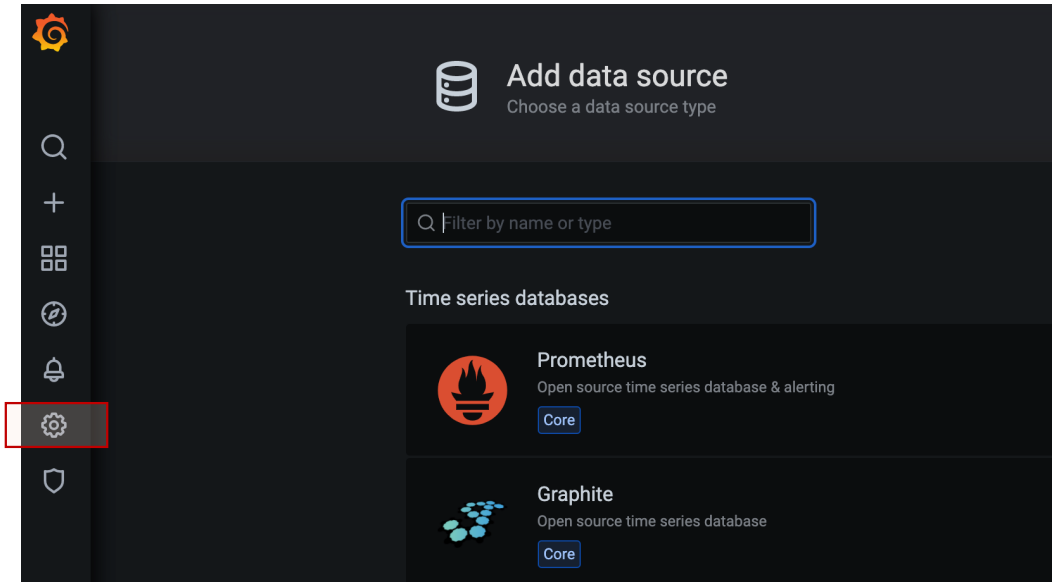
- Grafana 실행
  - http://127.0.0.1:3000
  - ID: admin, PW: admin

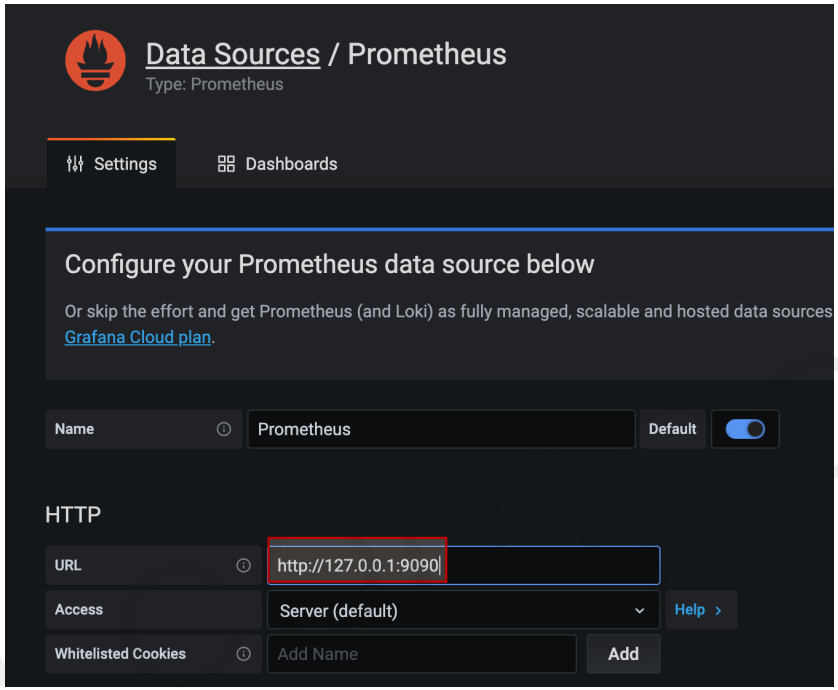
```
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="Add OAuth expiry to user_auth"
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="Add index to user_id co
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="create server_lock tabl
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="add index server_lock.o
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="create user auth token
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="add unique index user_a
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="add unique index user_a
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="add index user_auth_tok
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="create cache_data table
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="add unique index cache_
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="create short_url table
INFO[03-06|00:57:08] Executing migration                  logger=migrator id="add index short_url.org
INFO[03-06|00:57:08] Created default admin                logger=sqlstore user=admin
INFO[03-06|00:57:08] Created default organization         logger=sqlstore
INFO[03-06|00:57:08] Starting plugin search               logger=plugins
INFO[03-06|00:57:08] Registering plugin                   logger=plugins id=input
INFO[03-06|00:57:08] External plugins directory created   logger=plugins directory=/Users/dowonlee/De
ns
INFO[03-06|00:57:08] HTTP Server Listen                   logger=http.server address=[::]:3000 protoc
```

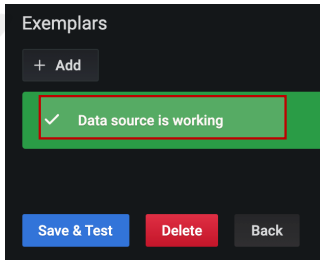


Grafana

- Grafana – Prometheus 연동

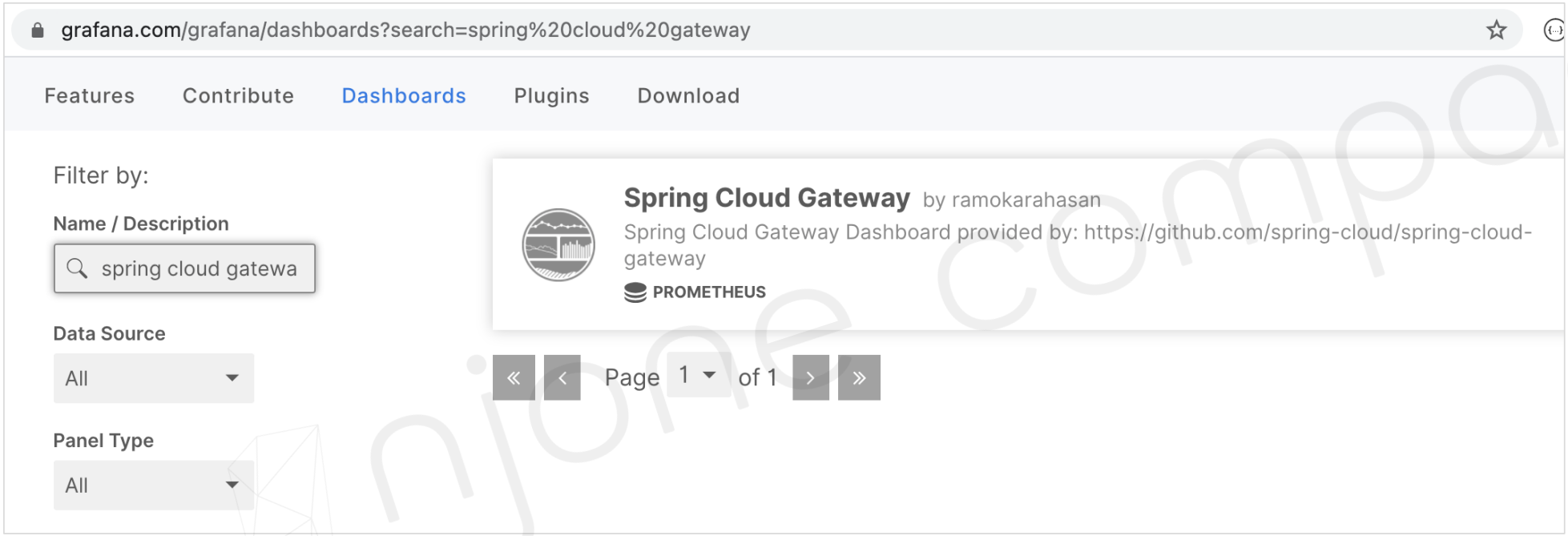






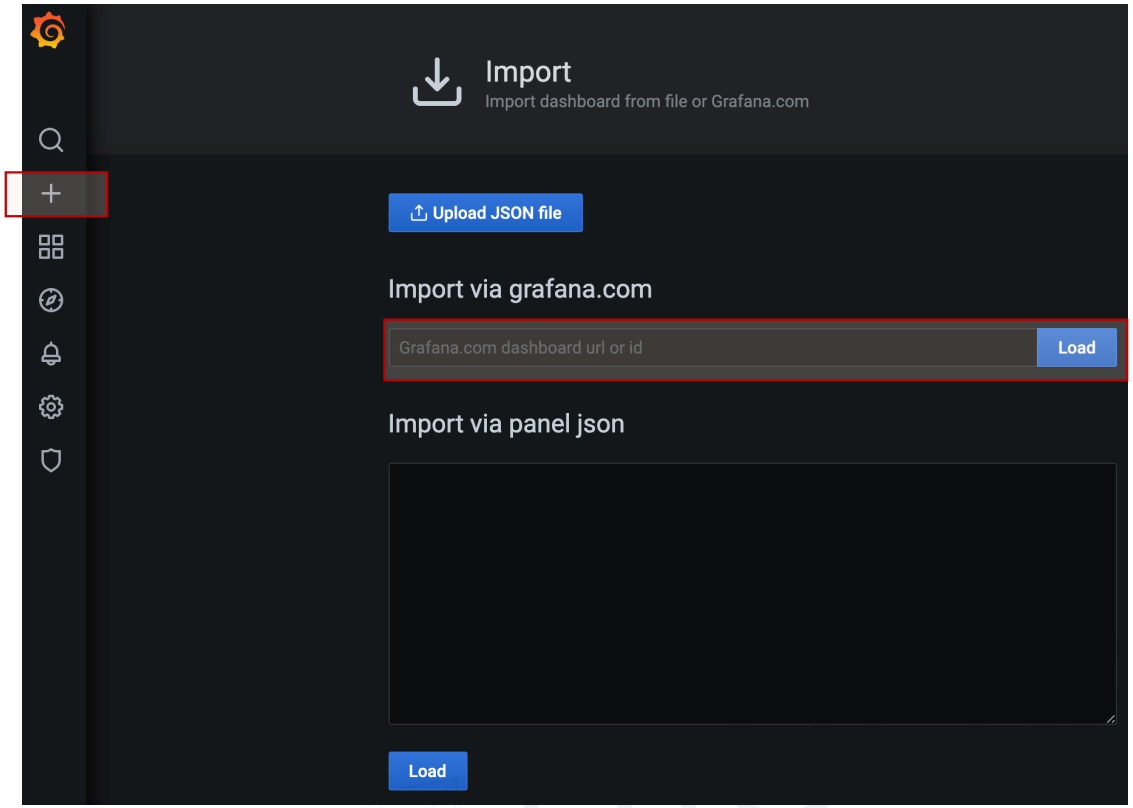
Grafana

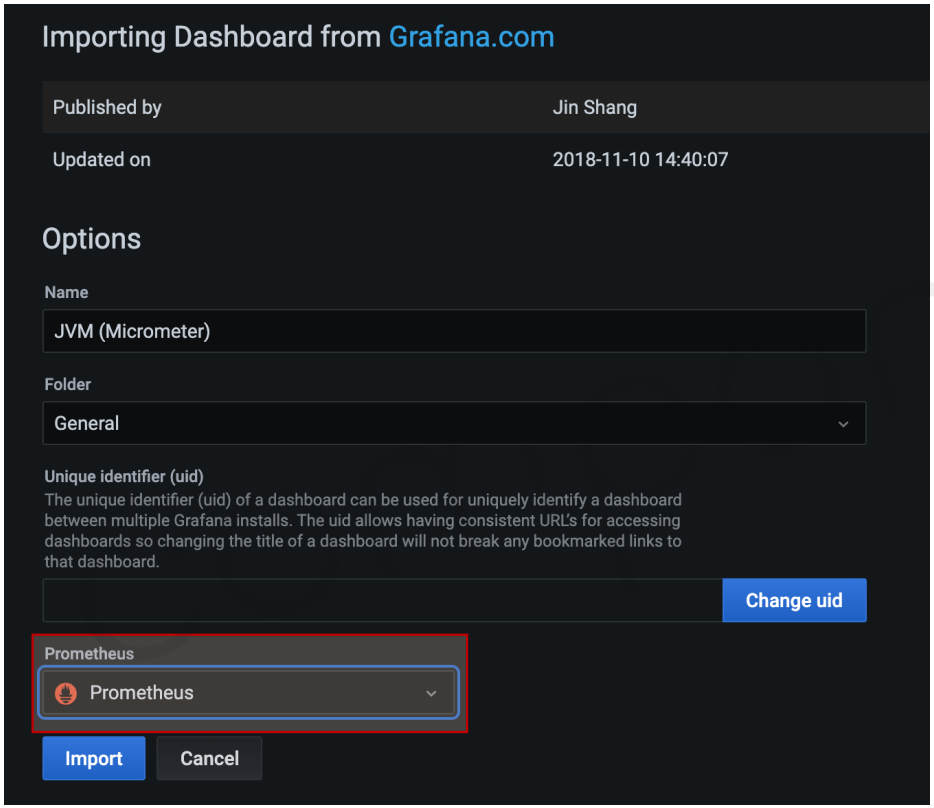
- Grafana Dashboard
  - JVM(Micrometer)
  - Prometheus
  - Spring Cloud Gateway



Grafana

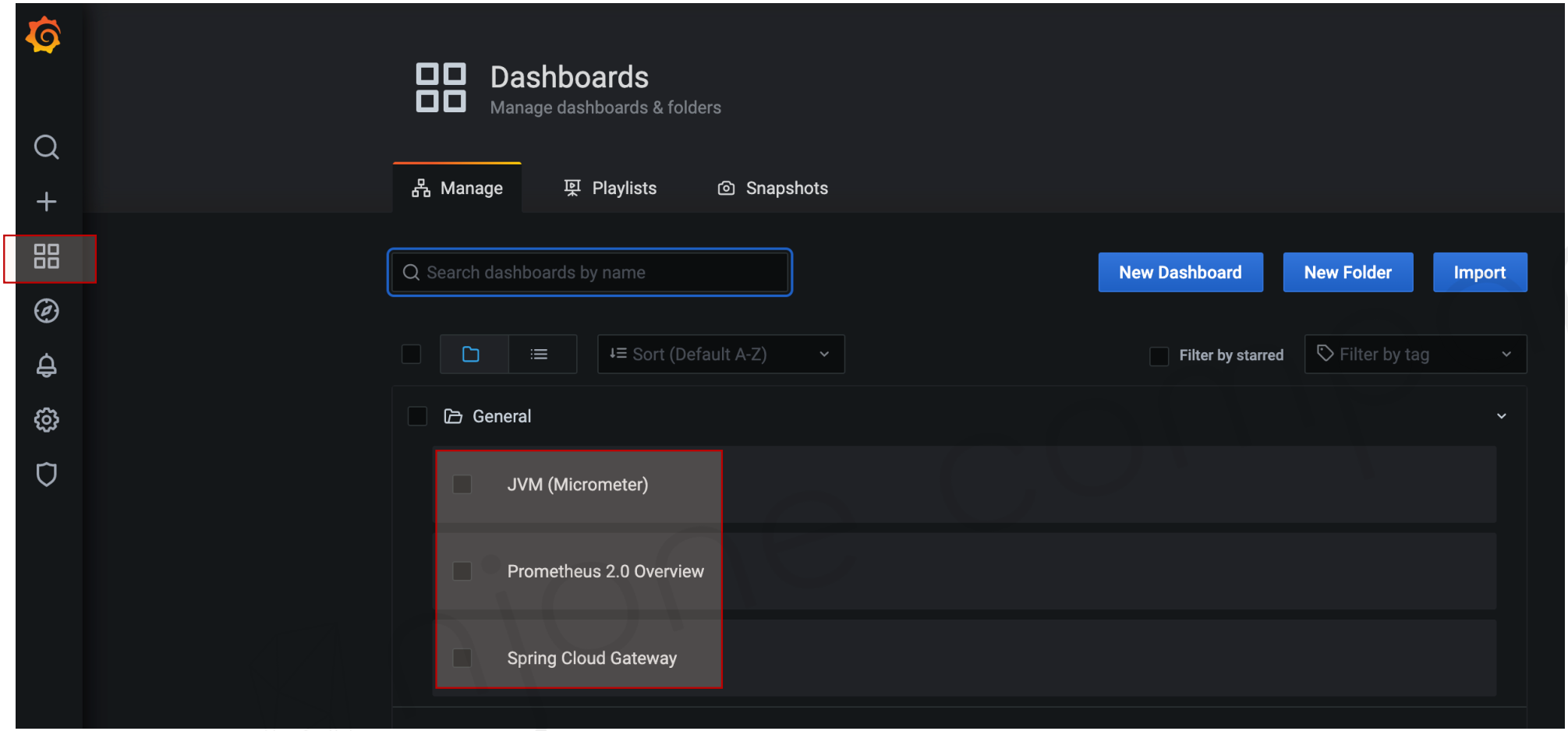
- Grafana – Prometheus 연동





Grafana

- Grafana Dashboard



Grafana

- ***Spring Cloud Gateway*** dashboard