

# Spring Cloud로 개발하는 マイクロ서비스 アプリケイ션



Microservices

+



Spring  
Cloud

```
class Book {
    def self, title, price, author;
    self.title = title
    self.price = price
    self.author = author
}

public static void main(String[] args)
{
    var fs = require('fs');
    fs.readFile('/JONE.txt' /* 1 */,
        function (err, data) {
            console.log(data); // 3
        });
}

<@interface NextInnovationDelegate : NSObject <UIApplicationDelegate> >

<@implementation NextInnovationDelegate >
<@end>
```



# 목차



- Section 9: 암호화 처리를 위한 Encryption과 Decryption
- Section 10: 마이크로서비스간 통신
- Section 11: 데이터 동기화를 위한 Kafka 활용 ①
- Section 12: 데이터 동기화를 위한 Kafka 활용 ②
- Section 13: 장애 처리와 Microservice 분산 추적
- **Section 14: Microservice 모니터링**
- Section 15: 애플리케이션 배포를 위한 컨테이너 가상화
- Section 16: 애플리케이션 배포 – Docker Container
- Appendix: Microservice 패턴

## Section 14.

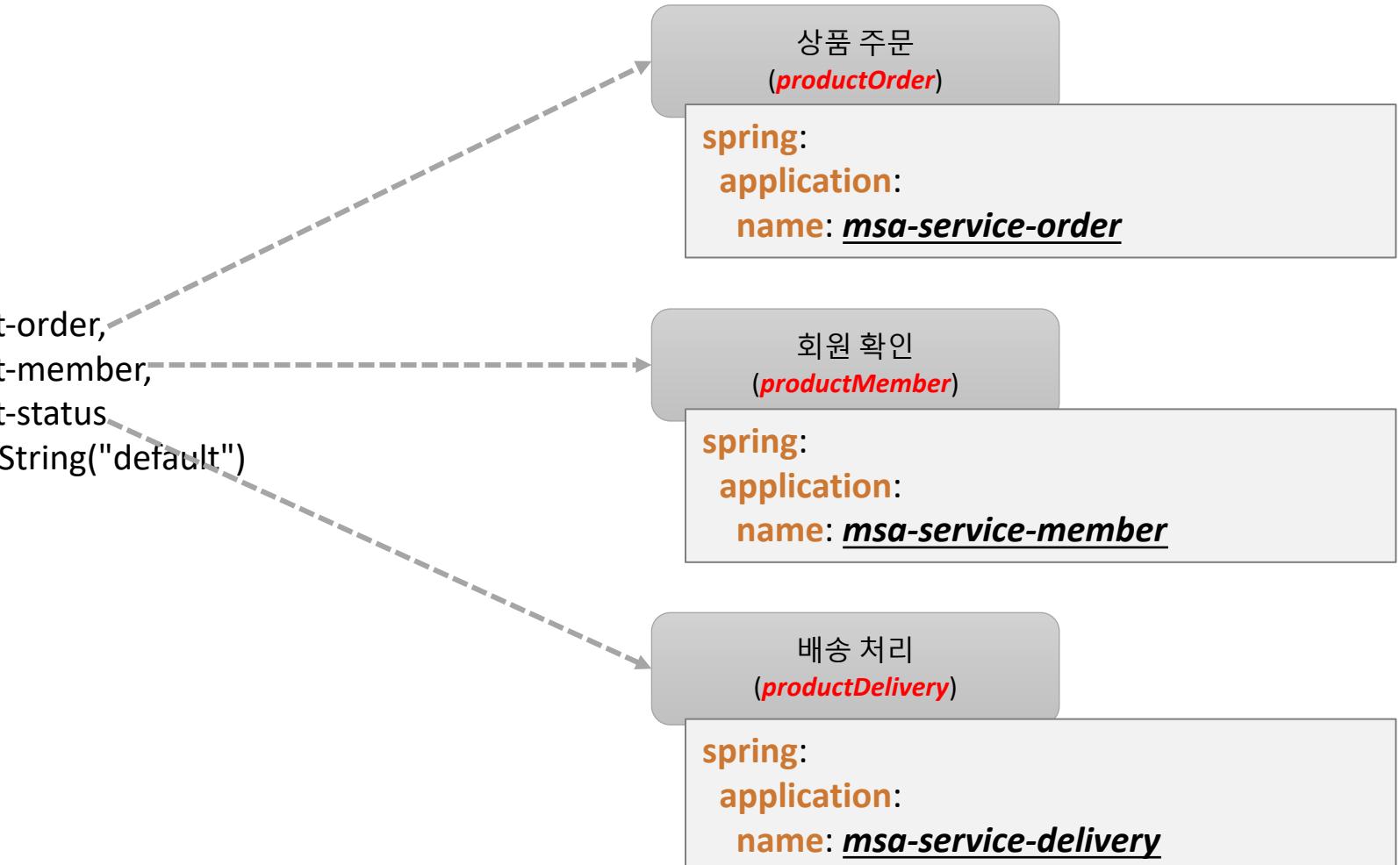
# Microservice 모니터링

- Hystrix Dashboard + Turbine Server
- Micrometer
- Prometheus
- Grafana

# Turbine Server

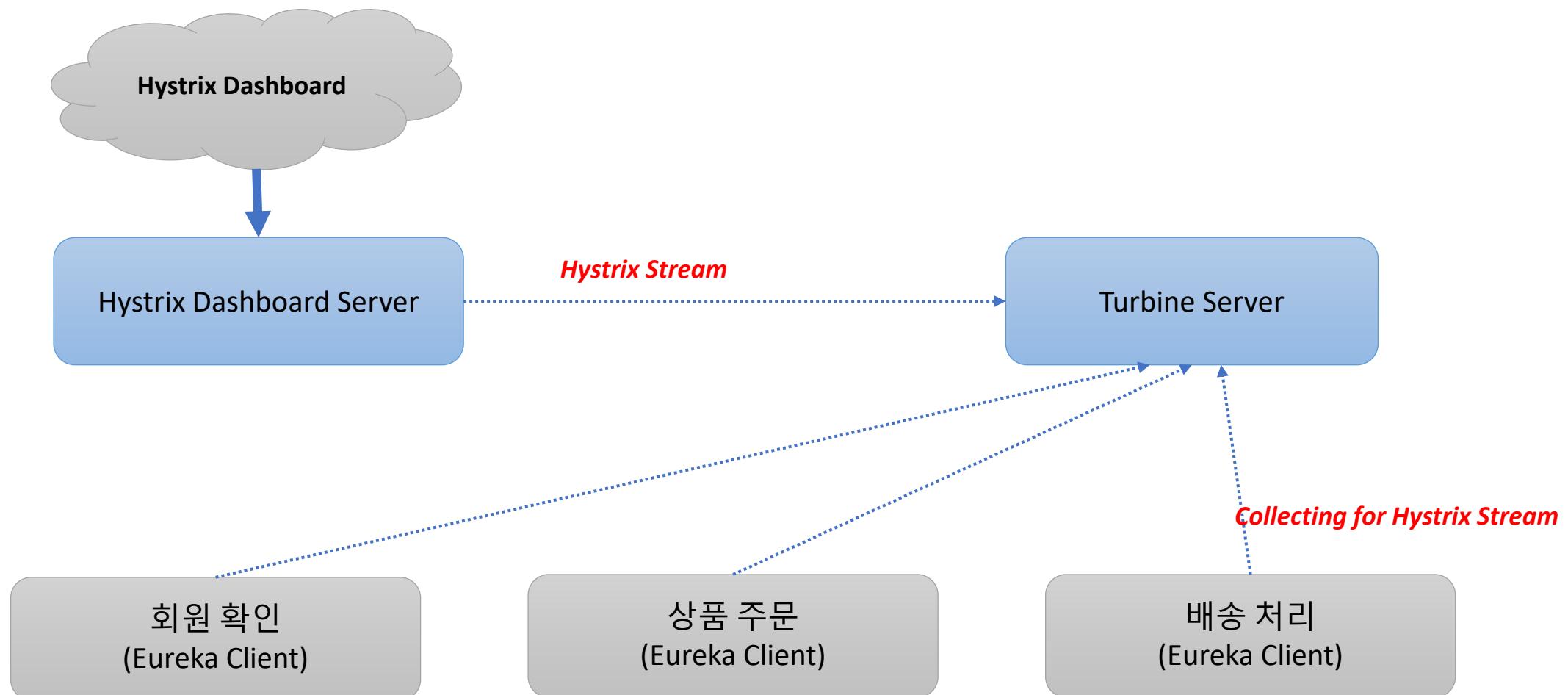
- 마이크로서비스에 설치된 Hystrix 클라이언트의 스트림을 통합
  - 마이크로서비스에서 생성되는 Hystrix 클라이언트 스트림 메시지를 터빈 서버로 수집

```
#Turbine Server
turbine:
  appConfig:
    msa-service-product-order,
    msa-service-product-member,
    msa-service-product-status
  clusterNameExpression: new String("default")
```



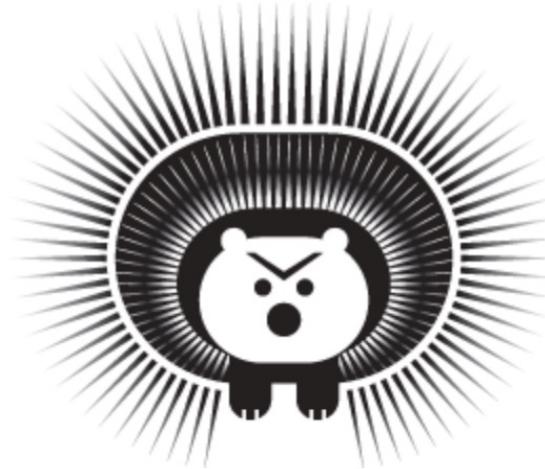
# Hystrix Dashboard

- Hystrix 클라이언트에서 생성하는 스트림을 시각화
  - Web Dashboard



# Hystrix Dashboard

localhost:7070/hystrix



## Hystrix Dashboard

*http://localhost:9999/turbine.stream*

*Cluster via Turbine (default cluster): http://turbine-hostname:port/turbine.stream*  
*Cluster via Turbine (custom cluster): http://turbine-hostname:port/turbine.stream?cluster=[clusterName]*  
*Single Hystrix App: http://hystrix-app:port/hystrix.stream*

Delay:  ms    Title:

Monitor Stream

# Hystrix Dashboard



localhost:7070/hystrix/monitor?stream=http%3A%2F%2Flocalhost%3A9999%2Fturbine.stream



## Hystrix Stream: http://localhost:9999/turbine.stream



Circuit Sort: [Error then Volume](#) | [Alphabetical](#) | [Volume](#) | [Error](#) | [Mean](#) | [Median](#) | [90](#) | [99](#) | [99.5](#) | [Success](#) | [Short-Circuited](#) | [Bad Request](#) | [Timeout](#) | [Rejected](#) | [Failure](#) | [Error %](#)



Host: 0.0/s

Cluster: 0.0/s

Circuit Closed

Hosts	1	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

Host: 0.0/s

Cluster: 0.0/s

Circuit Closed

Hosts	1	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

## Thread Pools

Sort: [Alphabetical](#) | [Volume](#) |

### CoffeeOrderRestController

Host: 0.0/s

Cluster: 0.0/s

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	2	Queue Size	5

### CoffeeMemberRestController

Host: 0.0/s

Cluster: 0.0/s

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	2	Queue Size	5



# *Micrometer + Monitoring*

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]



# Micrometer

## ■ Micrometer

- <https://micrometer.io/>
- JVM기반의 애플리케이션의 Metrics 제공
- Spring Framework 5, Spring Boot 2부터 Spring의 Metrics 처리
- Prometheus등의 다양한 모니터링 시스템 지원

## ■ Timer

- 짧은 자연 시간, 이벤트의 사용 빈도를 측정
- 시계열로 이벤트의 시간, 호출 빈도 등을 제공
- @Timed 제공

# Microservice 설정

```
<!-- micrometer -->  
<dependency>  
    <groupId>io.micrometer</groupId>  
    <artifactId>micrometer-registry-prometheus</artifactId>  
</dependency>
```

pom.xml

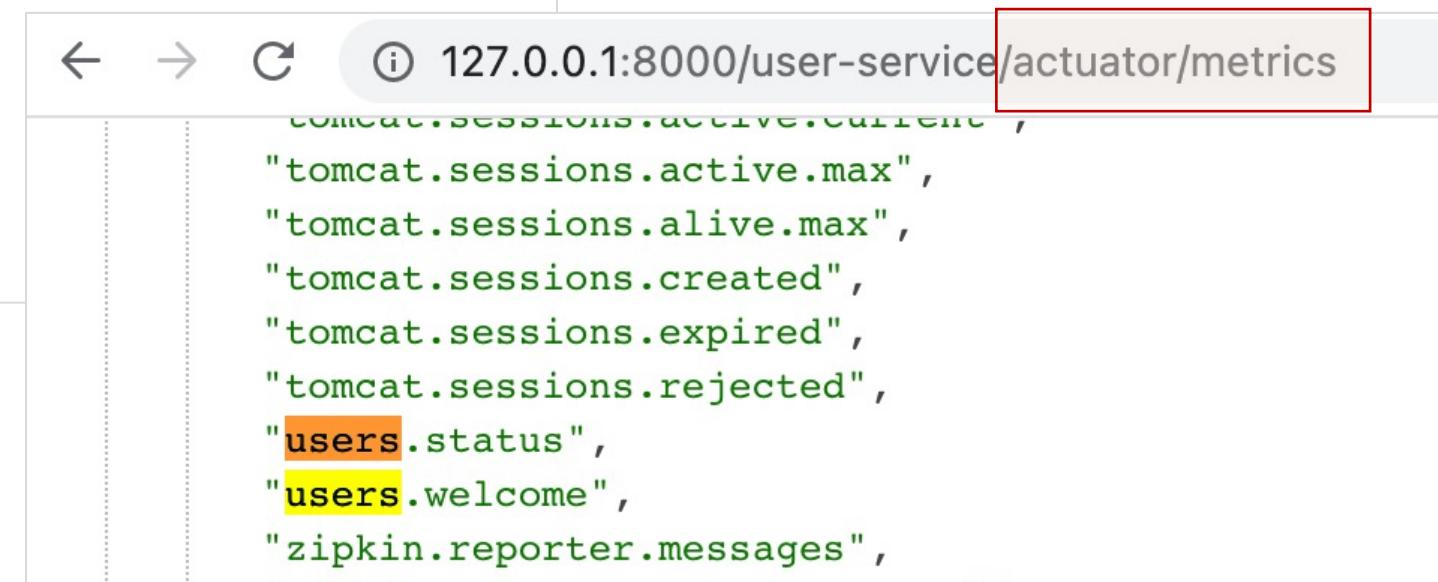
```
management:  
    endpoints:  
        web:  
            exposure:  
                include: refresh,health,beans,busrefresh,info,prometheus,metrics
```

application.yml (Users microservice)

# Microservice 설정

```
@GetMapping("/health_check")
@Timed(value="users.status", longTask = true)
public String status(HttpServletRequest request) {
    return String.format("It's Working in User Service"
        + ", port(local.server.port)=" + env.getProperty("local.server.port")
        + ", port(server.port)=" + env.getProperty("server.port")
        + ", with token secret=" + env.getProperty("token.secret")
        + ", with token time=" + env.getProperty("token.expiration_time"));
}
```

```
@GetMapping("/welcome")
@Timed(value="users.welcome", longTask = true)
public String welcome() {
    return env.getProperty("greeting.message");
    log.info(">>>GET");
    return greeting.getMessage();
}
```



# Microservice 설정

## ■ metrics 정보

◀ ▶ ⌂ ▲ 주의 요함 | 10.204.136.204:52257/actuator/prometheus

```
# HELP logback_events_total Number of error level events that made it to the logs
# TYPE logback_events_total counter
logback_events_total{level="warn",} 2.0
logback_events_total{level="debug",} 0.0
logback_events_total{level="error",} 0.0
logback_events_total{level="trace",} 0.0
logback_events_total{level="info",} 25.0
# HELP process_files_max_files The maximum file descriptor count
# TYPE process_files_max_files gauge
process_files_max_files 10240.0
# HELP jvm_gc_pause_seconds Time spent in GC pause

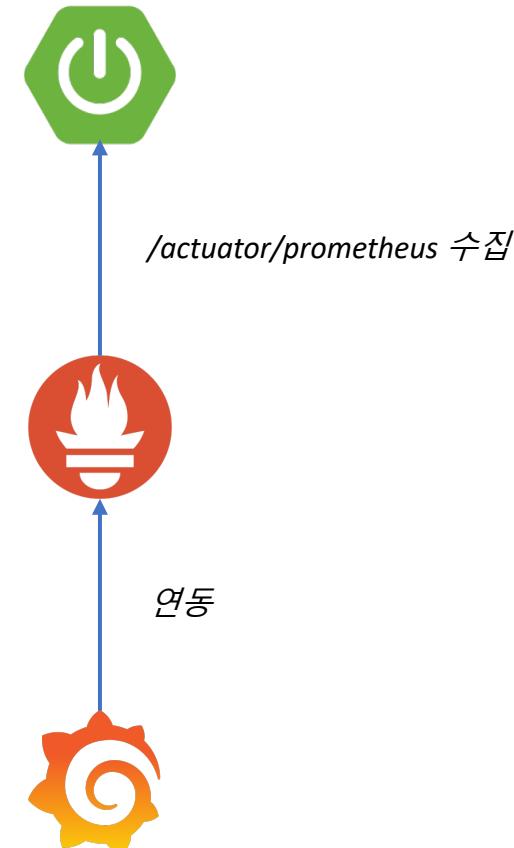
# HELP users_welcome_seconds_max
# TYPE users_welcome_seconds_max gauge
users_welcome_seconds_max{method="GET",uri="/welcome",} 0.0
# HELP users_welcome_seconds
# TYPE users_welcome_seconds summary
users_welcome_seconds_active_count{method="GET",uri="/welcome",} 0.0
users_welcome_seconds_duration_sum{method="GET",uri="/welcome",} 0.0

# HELP users_status_seconds_max
# TYPE users_status_seconds_max gauge
users_status_seconds_max{method="GET",uri="/health_check",} 0.0
# HELP users_status_seconds
# TYPE users_status_seconds summary
users_status_seconds_active_count{method="GET",uri="/health_check",} 0.0
users_status_seconds_duration_sum{method="GET",uri="/health_check",} 0.0
```

# 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

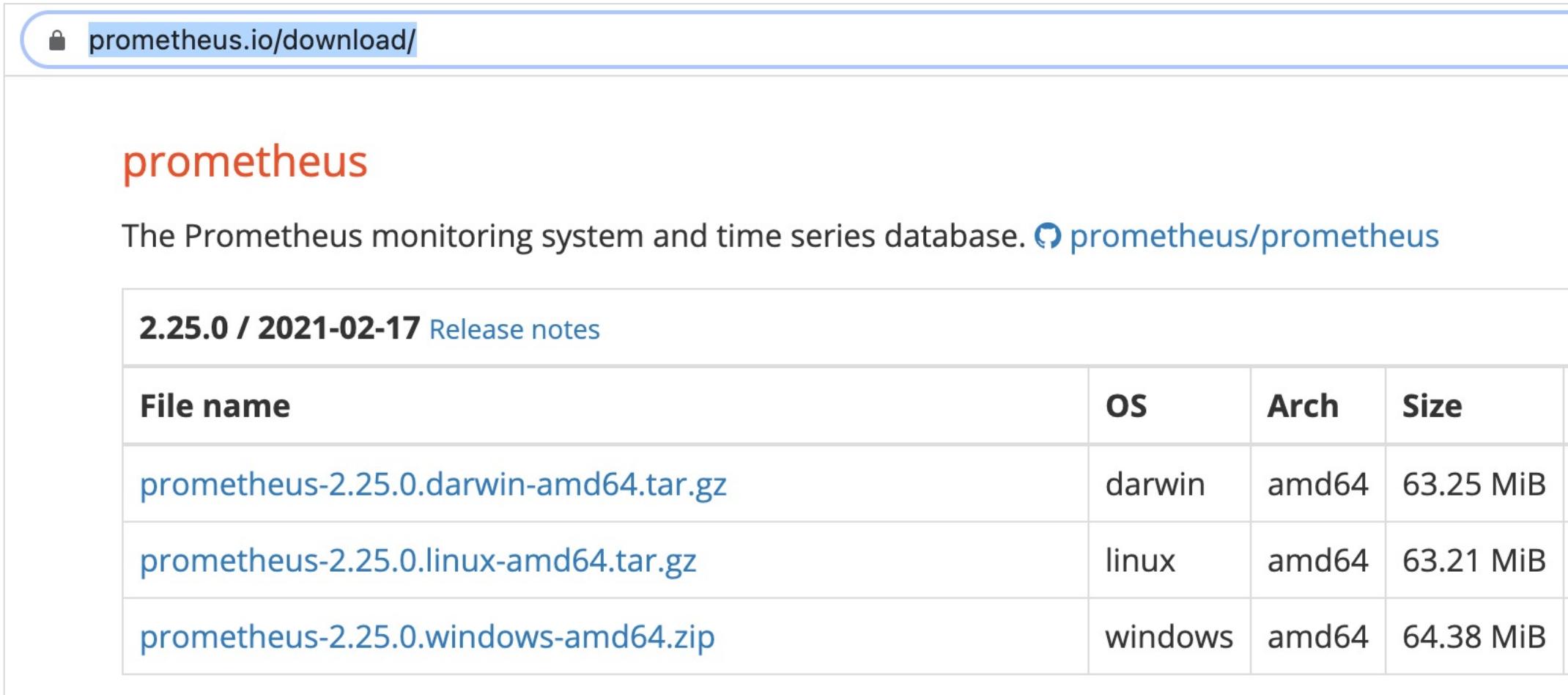
- 데이터 시각화, 모니터링 및 분석을 위한 오픈소스 애플리케이션
- 시계열 데이터를 시각화하기 위한 대시보드 제공



# Prometheus 설치

## ■ Prometheus 다운로드

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



The screenshot shows the official Prometheus download page. At the top, there's a navigation bar with links for Home, Documentation, and GitHub. Below it is a search bar. The main content area has a heading "prometheus" in red, followed by a brief description: "The Prometheus monitoring system and time series database." It includes a GitHub link. A section for "2.25.0 / 2021-02-17 Release notes" is shown. Below that is a table listing four file downloads:

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
```

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". Underneath the tabs, there is a "Evaluation time" input field with arrows for navigating between time points. The main content area displays the message "No data queried yet". At the bottom left, there is a blue button labeled "Add Panel". The URL "127.0.0.1:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.range\_input=2h" is visible in the browser's address bar.

# Prometheus

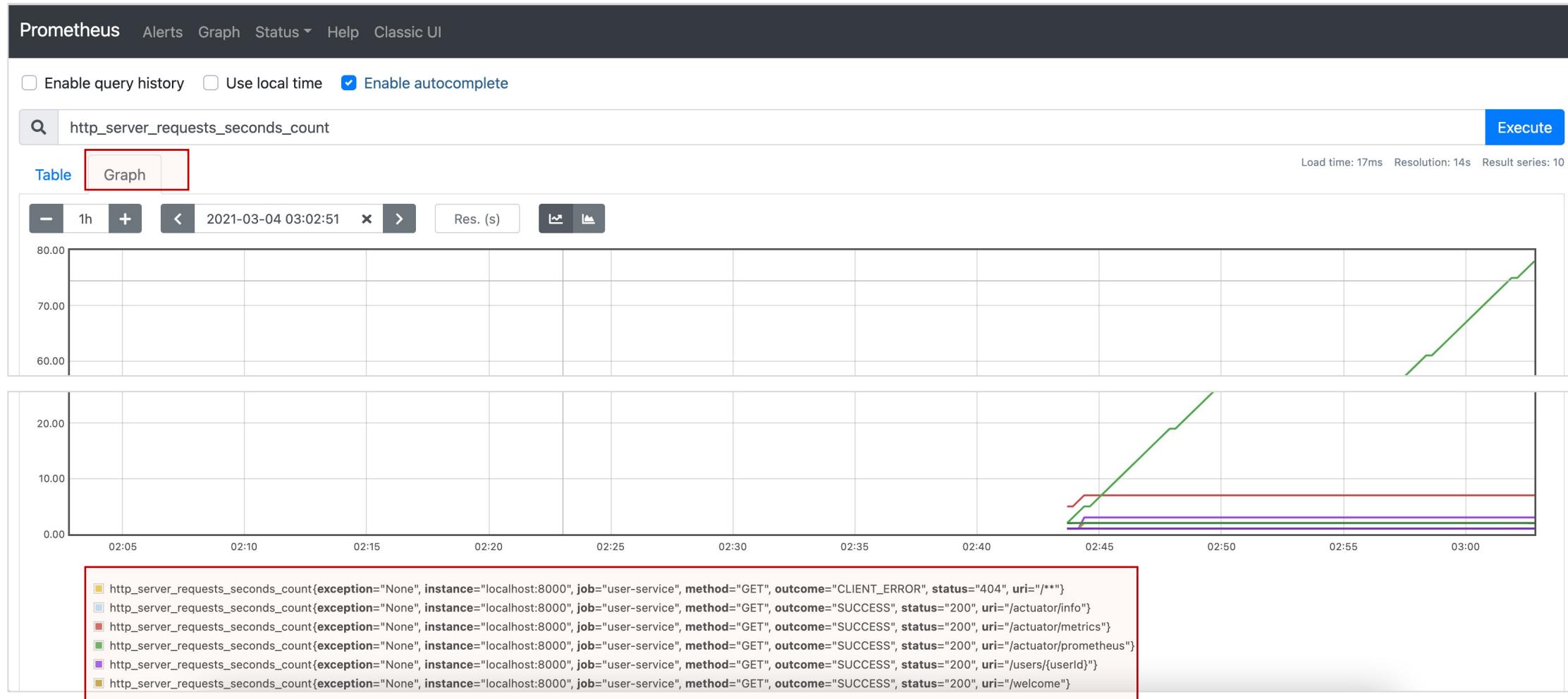
## ■ metrics 검사 – Table

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

- Header:** Prometheus, Alerts, Graph, Status ▾, Help, Classic UI.
- Search Bar:** A search bar containing "http\_server\_requests\_seconds\_count" with a magnifying glass icon, highlighted with a red border.
- Filter Buttons:** Table (selected) and Graph.
- Evaluation Time:** Evaluation time buttons with arrows for navigation.
- Table Content:** A list of metric query results:
  - 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

```
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: 4be06da2e4fbbd90d8c0fa68d27ae64843a21e9a8a894

[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

C:\> 로컬 디스크 (C:) > Work > grafana-7.4.3.windows-amd64 > grafana-7.4.3		
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

PC > 로컬 디스크 (C:) > Work > grafana-7.4.3.windows-amd64 > grafana-7.4.3 > bin		
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

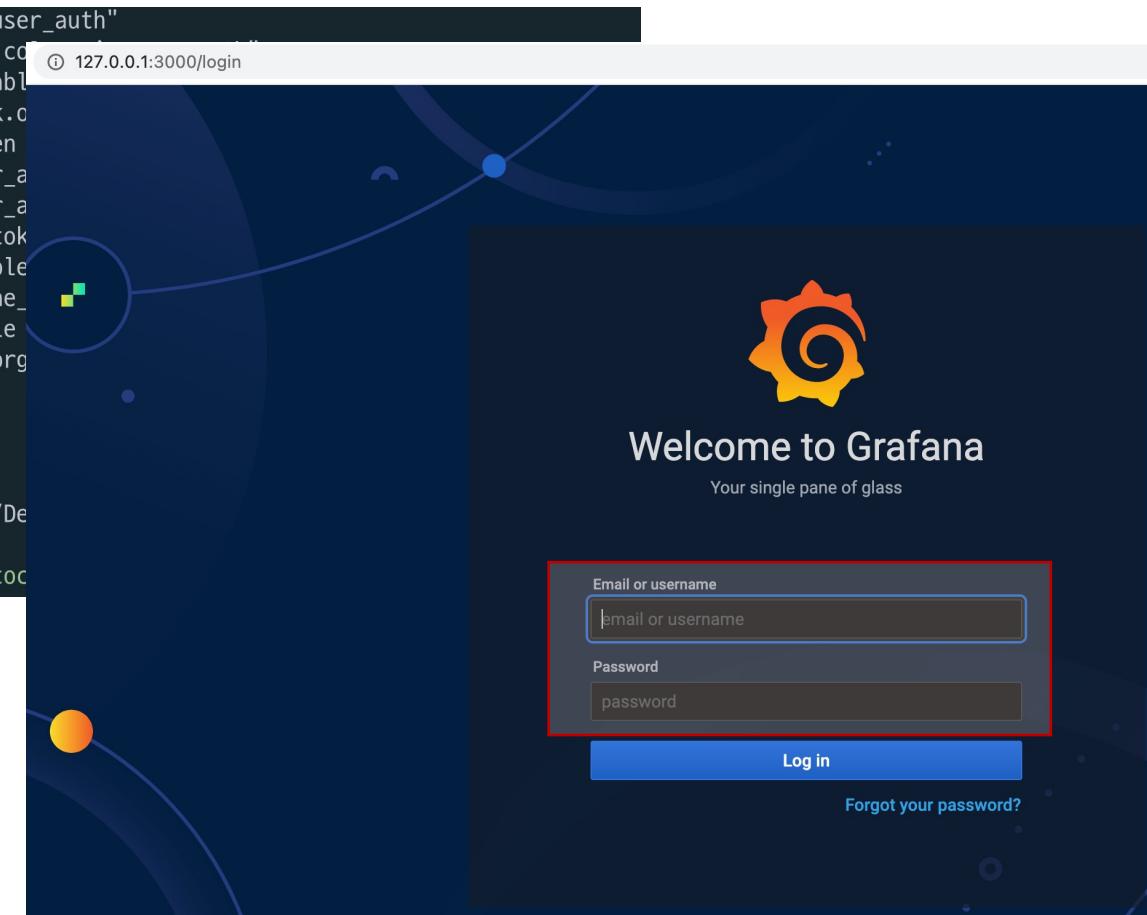
# Grafana

## ■ Grafana 실행

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

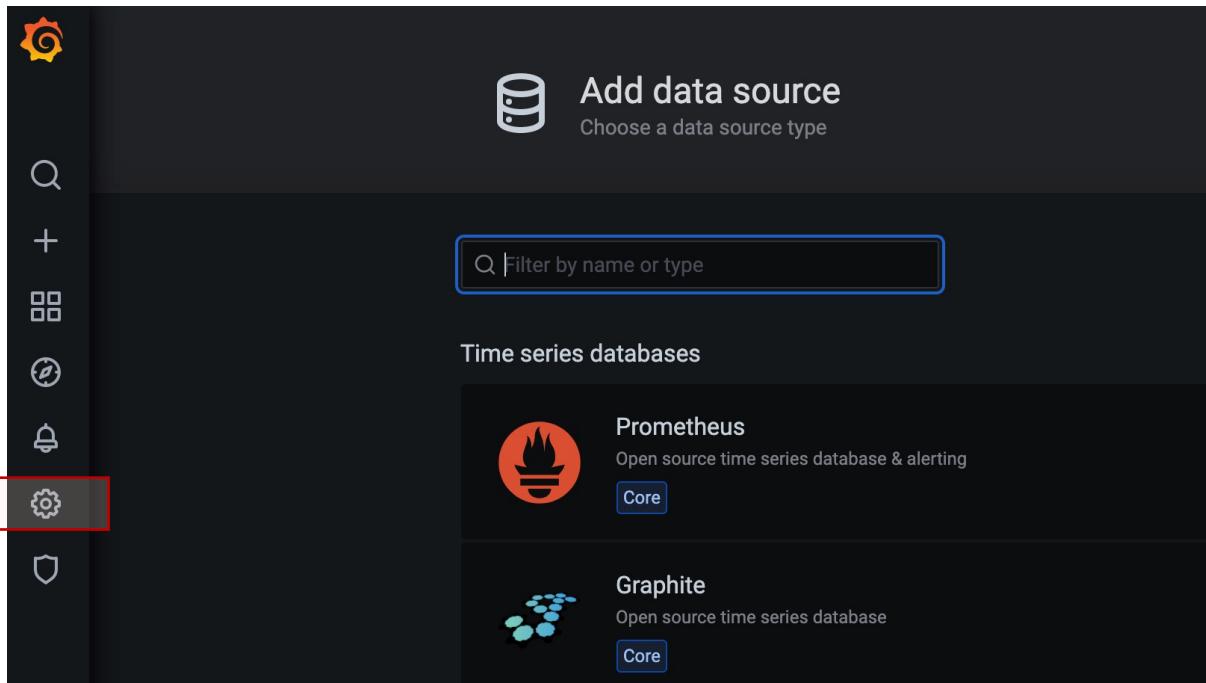
```
INFO[03-06|00:57:08] Executing migration
INFO[03-06|00:57:08] Created default admin
INFO[03-06|00:57:08] Created default organization
INFO[03-06|00:57:08] Starting plugin search
INFO[03-06|00:57:08] Registering plugin
INFO[03-06|00:57:08] External plugins directory created
ns
INFO[03-06|00:57:08]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

```
logger=migrator id="Add OAuth expiry to user_auth"
logger=migrator id="Add index to user_id column"
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/dowonlee/De
logger=http.server address=[::]:3000 protocol
```

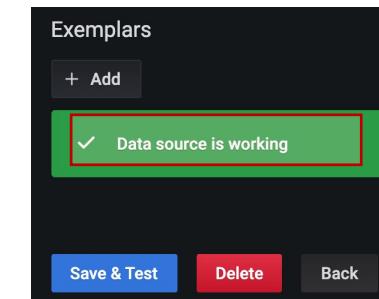


# Grafana

## ■ Grafana – Prometheus 연동



This screenshot shows the 'Data Sources / Prometheus' configuration page. At the top, it says 'Type: Prometheus'. There are two tabs: 'Settings' (selected) and 'Dashboards'. The main section is titled 'Configure your Prometheus data source below' with a sub-instruction: 'Or skip the effort and get Prometheus (and Loki) as fully managed, scalable and hosted data sources [Grafana Cloud plan](#)'. It includes fields for 'Name' (set to 'Prometheus'), 'Default' (checkbox checked), and 'HTTP' settings. Under 'HTTP', the 'URL' field contains 'http://127.0.0.1:9090' (also highlighted with a red box). The 'Access' dropdown is set to 'Server (default)'. The 'Whitelisted Cookies' section has an 'Add Name' button. At the bottom, there's a green 'Exemplars' section with a success message: '✓ Data source is working' (also highlighted with a red box).



# Grafana

## ■ Grafana Dashboard

- JVM(Micrometer)
- Prometheus
- Spring Cloud Gateway

grafana.com/grafana/dashboards?search=spring%20cloud%20gateway

Features Contribute Dashboards Plugins Download

Filter by:

Name / Description

spring cloud gateway

Data Source

All

Panel Type

All

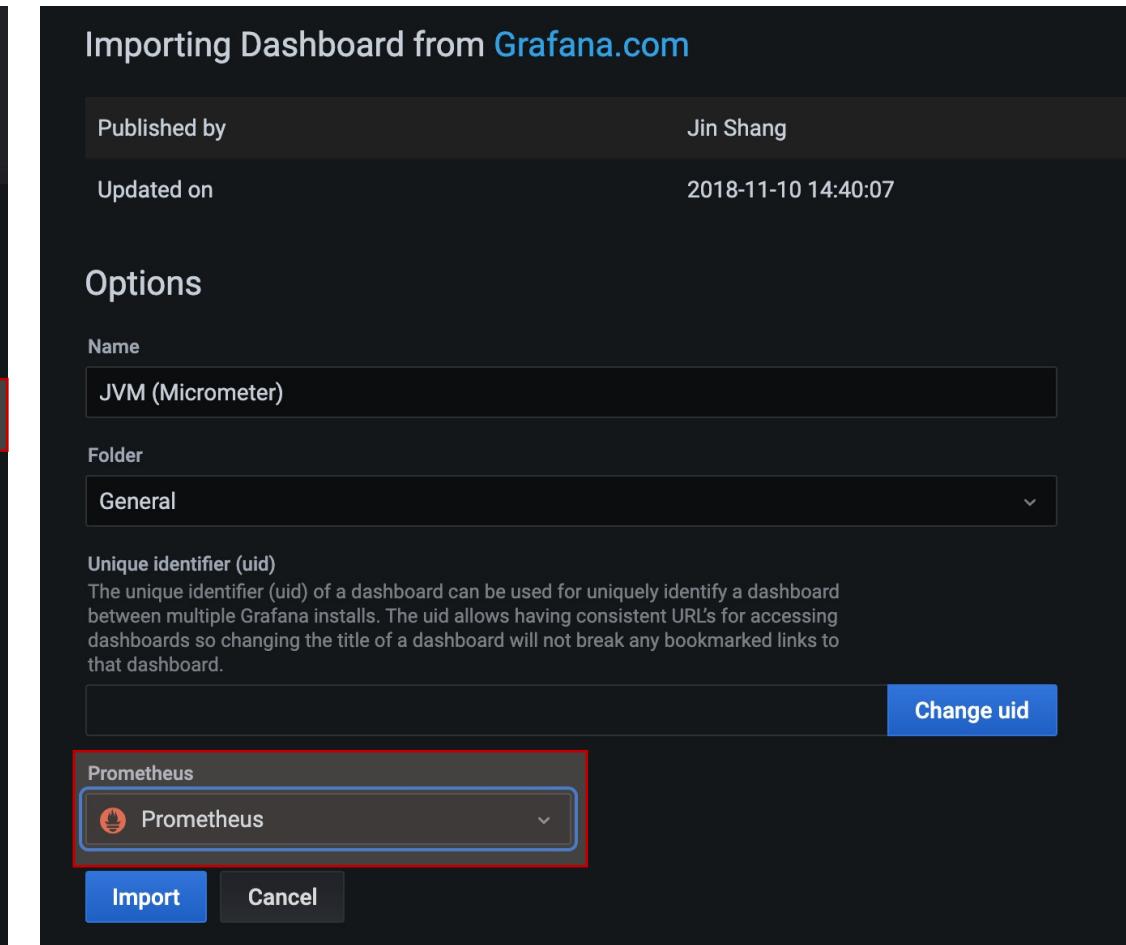
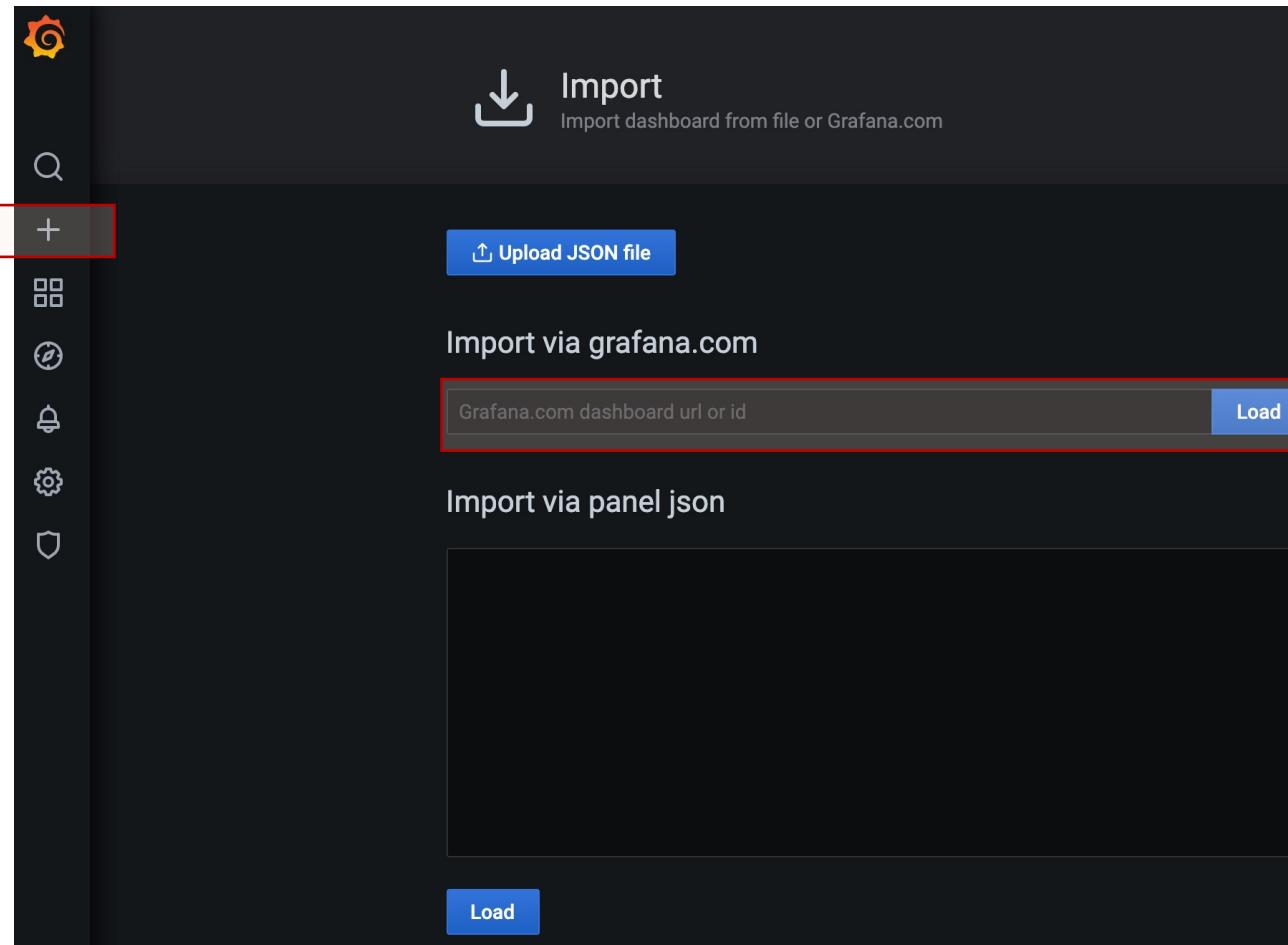


**Spring Cloud Gateway** by ramokarahasan  
Spring Cloud Gateway Dashboard provided by: <https://github.com/spring-cloud/spring-cloud-gateway>  
 PROMETHEUS

« < Page 1 of 1 > »

# Grafana

## ■ Grafana – Prometheus 연동



# Grafana

## ■ Grafana Dashboard

The screenshot shows the Grafana dashboard management interface. On the left, there is a sidebar with icons for settings, search, and other management functions. The main area is titled "Dashboards" and includes a subtitle "Manage dashboards & folders". Below this are three buttons: "Manage", "Playlists", and "Snapshots". A search bar labeled "Search dashboards by name" is present. At the top right are three blue buttons: "New Dashboard", "New Folder", and "Import". Below the search bar are filters for "Sort (Default A-Z)", "Filter by starred", and "Filter by tag". A list of dashboards is shown under a "General" folder. Three specific dashboards are highlighted with a red box: "JVM (Micrometer)", "Prometheus 2.0 Overview", and "Spring Cloud Gateway".

Dashboards

Manage dashboards & folders

Manage Playlists Snapshots

Search dashboards by name

New Dashboard New Folder Import

Sort (Default A-Z)

Filter by starred Filter by tag

JVM (Micrometer)

Prometheus 2.0 Overview

Spring Cloud Gateway

# Grafana

## ■ Spring Cloud Gateway dashboard

