


12장. 모니터링 시스템

서버와 관련된 통계 데이터들을 저장하는 데이터베이스에 대해 알아보고,
이를 시각화 할 수 있는 솔루션을 사용해본다.

데이터베이스의 종류

- 흔히 알고있는 MySQL, Oracle Database, MariaDB는 RDBMS에 속한다.
 - RDBMS는 오른쪽과 같은 테이블 형태를 가진다.
 - 전통적인 데이터베이스 구조이며 가장 널리 사용되는 구조이다.
- RDBMS=Relational Database Management System

Department		Employee		
Primary key		Primary key		Foreign key
Deptno	Dname	Empno	Ename	Deptno
10	Train	185	Pierre	10
20	Sales	186	Smith	20
30	R&D	187	Tom	10



데이터베이스의 종류

- RDBMS를 포함한 데이터베이스의 종류는 아래와 같다.
 - RDBMS(Relational Database Management System)
 - Oracle, MySQL, Microsoft SQL Server, PostgreSQL, MariaDB
 - Document
 - MongoDB
 - Key-value
 - Redis, Memcached
 - Time Series
 - InfluxDB, Prometheus, Graphite

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the [method](#) of calculating the scores.



388 systems in ranking, March 2022

Rank			DBMS	Database Model	Score		
Mar 2022	Feb 2022	Mar 2021			Mar 2022	Feb 2022	Mar 2021
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1251.32	-5.51	-70.42
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1198.23	-16.45	-56.59
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	933.78	-15.27	-81.52
4.	4.	4.	PostgreSQL + 💬	Relational, Multi-model ⓘ	616.93	+7.54	+67.64
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	485.66	-2.98	+23.27
6.	6.	↑ 7.	Redis +	Key-value, Multi-model ⓘ	176.76	+0.96	+22.61
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model ⓘ	162.15	-0.73	+6.14
8.	8.	8.	Elasticsearch	Search engine, Multi-model ⓘ	159.95	-2.35	+7.61
9.	9.	↑ 10.	Microsoft Access	Relational	135.43	+4.17	+17.29
10.	10.	↓ 9.	SQLite +	Relational	132.18	+3.81	+9.54
11.	11.	11.	Cassandra +	Wide column	122.14	-1.83	+8.51
12.	12.	12.	MariaDB +	Relational, Multi-model ⓘ	108.31	+1.20	+13.85
13.	13.	13.	Splunk	Search engine	95.36	+4.55	+8.44
14.	↑ 15.	↑ 30.	Snowflake +	Relational	86.23	+3.05	+63.04
15.	↓ 14.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model ⓘ	84.68	-0.28	+13.79
16.	↑ 17.	↑ 17.	Amazon DynamoDB +	Multi-model ⓘ	81.80	+1.45	+12.91
17.	↓ 16.	↓ 14.	Hive +	Relational	81.22	-0.66	+5.18
18.	18.	↓ 15.	Teradata +	Relational, Multi-model ⓘ	68.85	+0.28	-2.58
19.	↑ 20.	↓ 18.	Neo4j +	Graph	59.67	+1.43	+7.35
20.	↓ 19.	↑ 21.	Solr	Search engine, Multi-model ⓘ	59.05	+0.52	+8.84
21.	21.	↓ 20.	SAP HANA +	Relational, Multi-model ⓘ	56.01	-0.30	+5.02

시계열 데이터베이스란?

- 시계열 데이터베이스(Time Series Database)는 시간에 따른 데이터들을 저장하는 데이터베이스이다.
 - 주가 데이터, 시간대별 동시접속 인원수, 웹 접속 수, 온도 정보 등이 시계열 데이터가 될 수 있다.



시계열 데이터베이스 vs RDBMS

- 시계열 데이터베이스는 시계열 데이터의 저장과 조회에 특화되어 있다.
- RDBMS에서 사용하는 SQL이 아닌 전용 쿼리 언어를 사용하게 된다.
 - 예) Prometheus -> PromQL

1 Answer

Active

Oldest

Votes



Time Series Database Advantages:

27



1. Throughout 100K+ to 1M+ inserts per second
2. Bytes stored per time/value tuple: 2-10 vs 30-100 (rdbms)
3. Built-in time series transformation and aggregations functions
4. Schema optimized for time-series arrays with built-in sharding and indexing

Relational Database Advantages:

1. Full SQL support
2. Ability to store any data other than time-series
3. Extensive DBA resources and tooling

By writing your own application code and stored procedures you can accomplish almost everything that TSDBs provide, but you may end up with an implementation that is slower and more demanding in terms of underlying compute and storage resources.

For all practical purposes and with some compromises to referential integrity, you can have both databases run concurrently: store your extended app schema in relational database and time-series in TSDB.

시계열 데이터베이스 – Prometheus

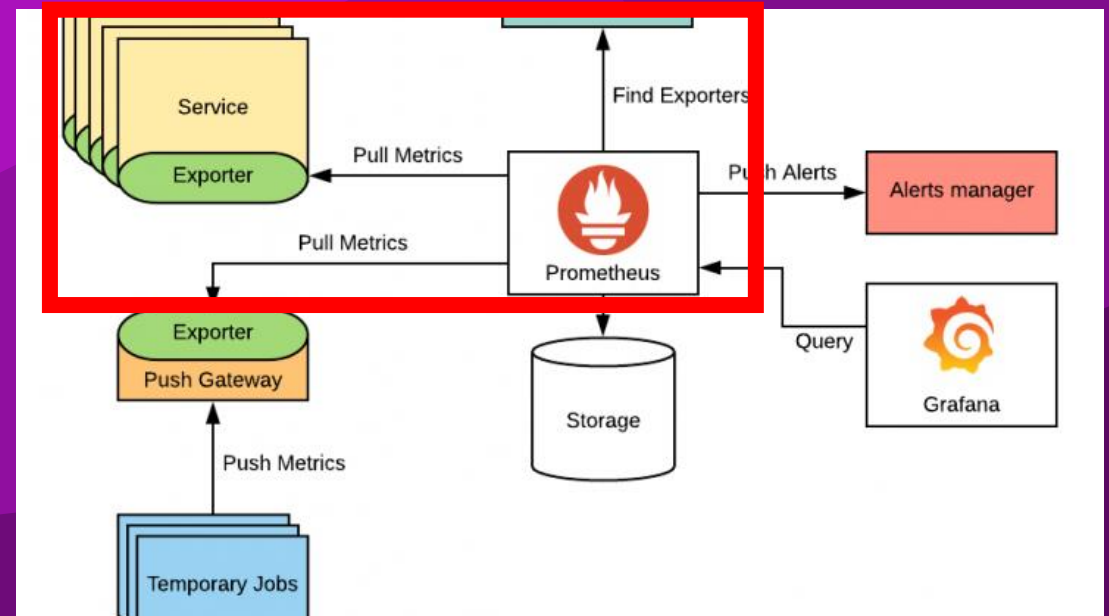
- Prometheus는 자주 사용하는 시계열 데이터베이스 중 하나이다.
- 간단한 웹 인터페이스를 제공한다.
- Prometheus에서 수집하는 시계열 데이터는 메트릭(Metric)이라고 한다.



Prometheus

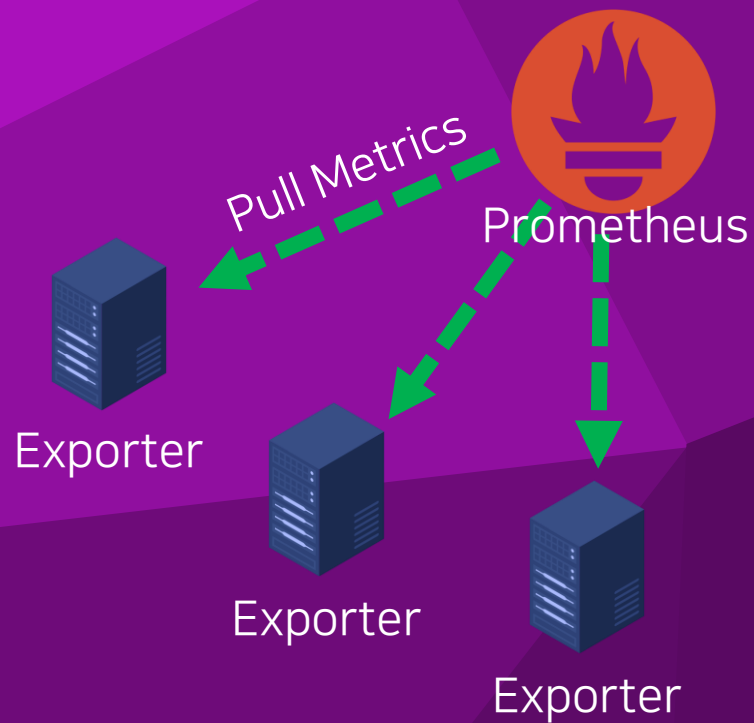
Prometheus와 Exporter

- Prometheus는 두 가지로 구성되어 있다.
 - Prometheus: 데이터가 저장되는 서버
 - Exporter: 시계열 데이터를 보내내기 위한 서버
- Exporter는 웹 서버이다.
 - `http://exporter-ip/metrics` 로 접근한다.
 - Prometheus는 위의 주소로 접속하여 데이터를 모은다.
- 이와 같은 동작을 Pull 이라고 한다.



Prometheus와 Exporter

- Pull Metrics 동작은 크롤링이라 볼 수 있다.
- Exporter는 웹 서버를 켜두고, Prometheus는 일정 주기로 크롤링을 한다.
- 데이터 수집 주기를 Prometheus 측에서 조정할 수 있는 장점이 있다.



[실습] Prometheus 설치

- docker-compose.yml에서 오른쪽과 같이 서비스를 추가한다.
- 컨테이너 내의 /etc/prometheus는 설정 파일이 저장되는 경로이다.
 - 이는 ./prometheus 폴더로 연결되도록 하였다.
- Prometheus에는 retention 옵션이 있다.
 - 오래된 데이터를 삭제하는 정책이다.
 - retention.time: 데이터 보관 기간 설정
 - retention.size: 데이터 최대 용량 설정

```
69
70 ▼ prometheus:
71     container_name: prometheus
72     image: prom/prometheus
73     restart: unless-stopped
74     user: 1000:1000
75     ports:
76     - 9090:9090
77 ▼ volumes:
78     - ./prometheus:/etc/prometheus
79     - ./prometheus/data:/prometheus
80 ▼ command:
81     - --config.file=/etc/prometheus/prometheus.yml
82     - --storage.tsdb.path=/prometheus
83     - --storage.tsdb.retention.time=5y # 기본 값은 15일
84     - --storage.tsdb.retention.size=20GB # 기본 값은 0 (무제한)
85     - --web.console.libraries=/usr/share/prometheus/console_libraries
86     - --web.console.templates=/usr/share/prometheus/consoles
87     - --web.enable-admin-api
88
```

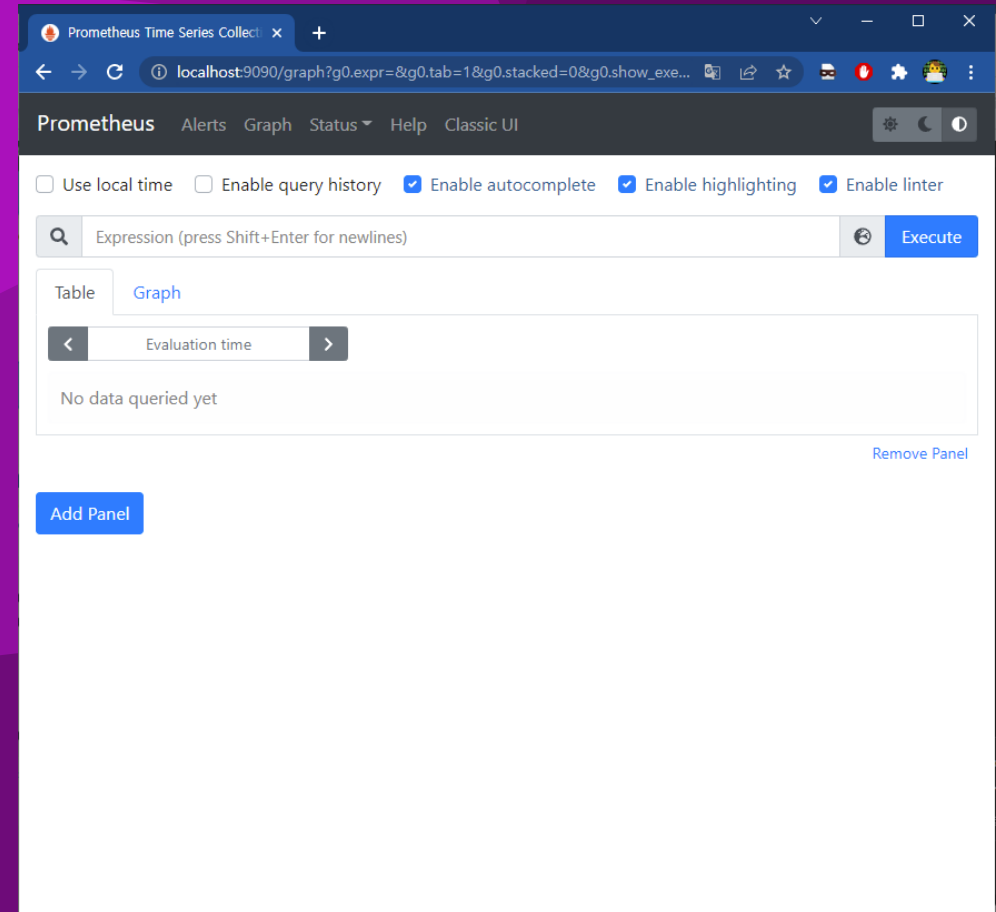
[실습] Prometheus 설치

- Prometheus가 읽을 수 있는 설정 파일을 추가하여야 한다.
- prometheus 폴더를 생성하고, prometheus.yml 파일을 생성하여 오른쪽과 같이 작성한 후 저장한다.
- Prometheus가 Pull 하는 주기는 15초로 설정한다.

```
1 ▾ global:
2   scrape_interval: 15s
3   evaluation_interval: 15s
4
5   # Attach these extra labels to all timeseries collected by this Prometheus instance.
6   external_labels:
7     monitor: minecraft
8
```

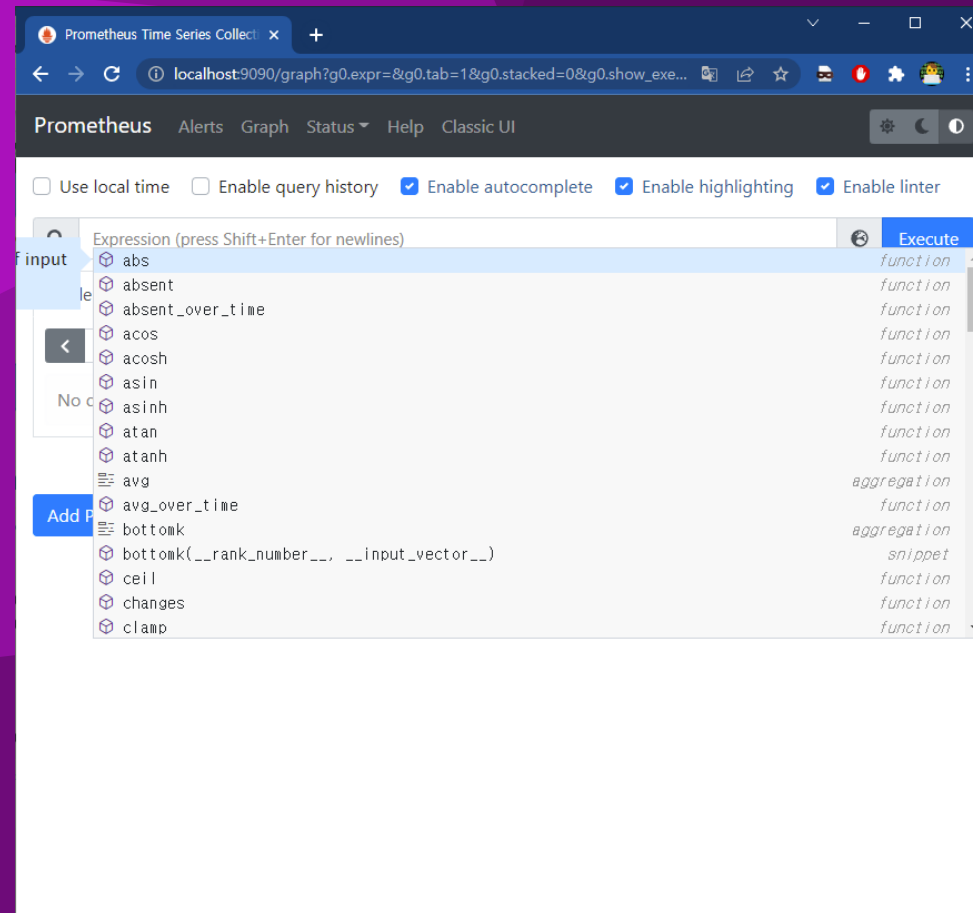
[실습] Prometheus 설치

- 터미널에 `docker-compose up -d prometheus` 를 입력하여 컨테이너를 생성하고,
- `http://localhost:9090` 으로 접속해보자.
- 오른쪽과 같은 웹 인터페이스가 나타날 것이다.



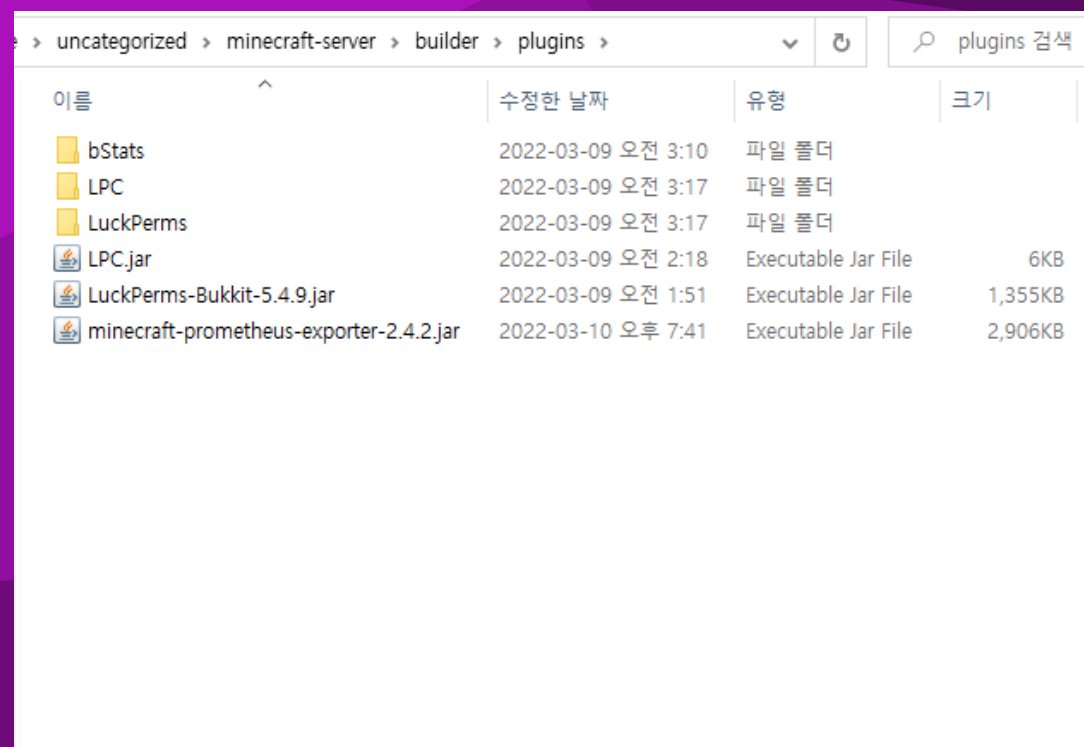
[실습] Prometheus 설치

- 쿼리 입력 칸에서 Ctrl + Space를 입력하면 자동 완성을 띄울 수 있다.
- 하지만 메트릭이 나타나진 않는다.
- Exporter를 추가하여 메트릭을 조회할 수 있도록 한다.



[실습] Prometheus와 마인크래프트 연동

- 마인크래프트 서버에 Exporter를 추가하여 Prometheus에서 메트릭을 조회해보자.
- 먼저, minecraft-prometheus-exporter 플러그인을 다운받아 서버에 적용한다.
 - lobby와 builder 모두 적용한다.



[실습] Prometheus와 마인크래프트 연동

- lobby, builder 서버를 한번 재부팅 후 config.yml 을 연다.
- Prometheus 컨테이너에서 lobby, builder 컨테이너로 접속하여도 되게끔 수정한다.
 - host: localhost 를
 - host: 0.0.0.0 으로 수정한다.

```
1 host: 0.0.0.0
2 port: 9225
3 enable_metrics:
4   entities_total: true
5   villagers_total: true
6   loaded_chunks_total: true
7   jvm_memory: true
8   players_online_total: true
9   players_total: true
10  tps: true
11  jvm_threads: true
12  jvm_gc: true
13  tick_duration_median: true
14  tick_duration_average: true
15  tick_duration_min: false
16  tick_duration_max: true
17  player_online: false
18  player_statistic: false
19
```

[실습] Prometheus와 마인크래프트 연동

- minecraft-Prometheus-exporter는 9225 포트로 웹 서버를 연다.
- Prometheus 컨테이너에서 lobby, builder 컨테이너의 9225 포트로 접근할 수 있게끔 expose 옵션을 추가한다.
- 만일 호스트에서도 접근을 한다면 expose 대신 ports에 추가한다.

```
18 ▾ lobby:
19     container_name: minecraft-lobby
20     image: itzg/minecraft-server
21     stdin_open: true
22     tty: true
23     restart: unless-stopped
24     volumes:
25     - ./lobby:/data
26     ... expose: [9225]
27 ▾ environment:
28     - EULA=TRUE
29     - TYPE=PAPER
30     - VERSION=1.18.2
31     - TZ=Asia/Seoul
32     - OVERRIDE_SERVER_PROPERTIES=true
33     - MAX_PLAYERS=50
34     - MODE=creative
35     - LEVEL=lobby
36     - MOTD=Welcome Server
37     - ALLOW_NETHER=FALSE
38     - VIEW_DISTANCE=50
39     - PVP=FALSE
40     - ONLINE_MODE=FALSE
41
42 ▾ builder:
43     container_name: minecraft-builder
44     image: itzg/minecraft-server
45     stdin_open: true
46     tty: true
47     restart: unless-stopped
48     ... expose: [9225]
49     volumes:
50     - ./builder:/data
51 ▾ environment:
52     - EULA=TRUE
53     - TYPE=PAPER
54     - VERSION=1.18.2
55     - TZ=Asia/Seoul
56     - OVERRIDE_SERVER_PROPERTIES=true
57     - MAX_PLAYERS=50
58     - MODE=creative
59     - LEVEL=lobby
```

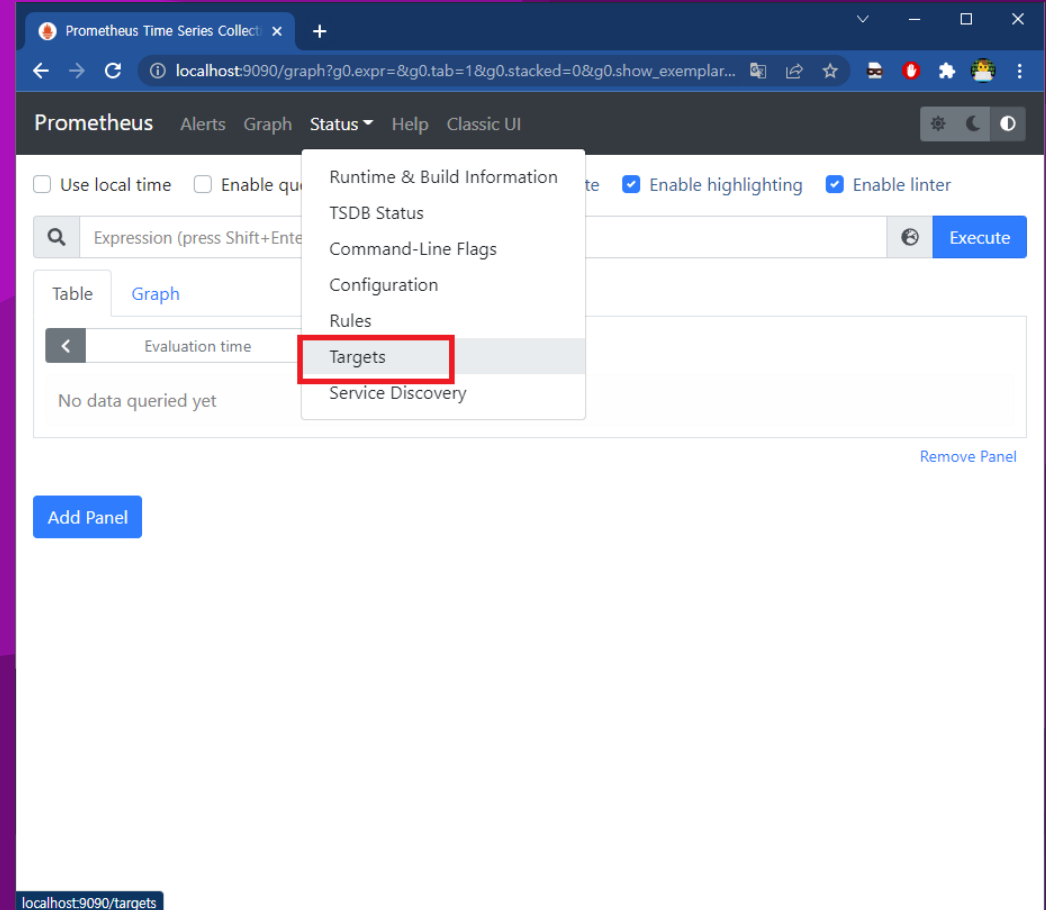
[실습] Prometheus와 마인크래프트 연동

- prometheus/prometheus.yml 파일을 수정하여 minecraft-prometheus-exporter를 추가한다.
- 등록해놓은 exporter는 scrape_interval마다 크롤링되어 Prometheus 데이터베이스에 저장된다.

```
1 ▾ global:
2   scrape_interval: 15s
3   evaluation_interval: 15s
4
5   # Attach these extra labels to all timeseries collected by
6   external_labels:
7     monitor: minecraft
8
9
10 ▾ scrape_configs:
11 ▾   - job_name: minecraft-lobby
12     static_configs:
13       - targets: ['lobby:9225']
14
15 ▾   - job_name: minecraft-builder
16     static_configs:
17       - targets: ['builder:9225']
18
```

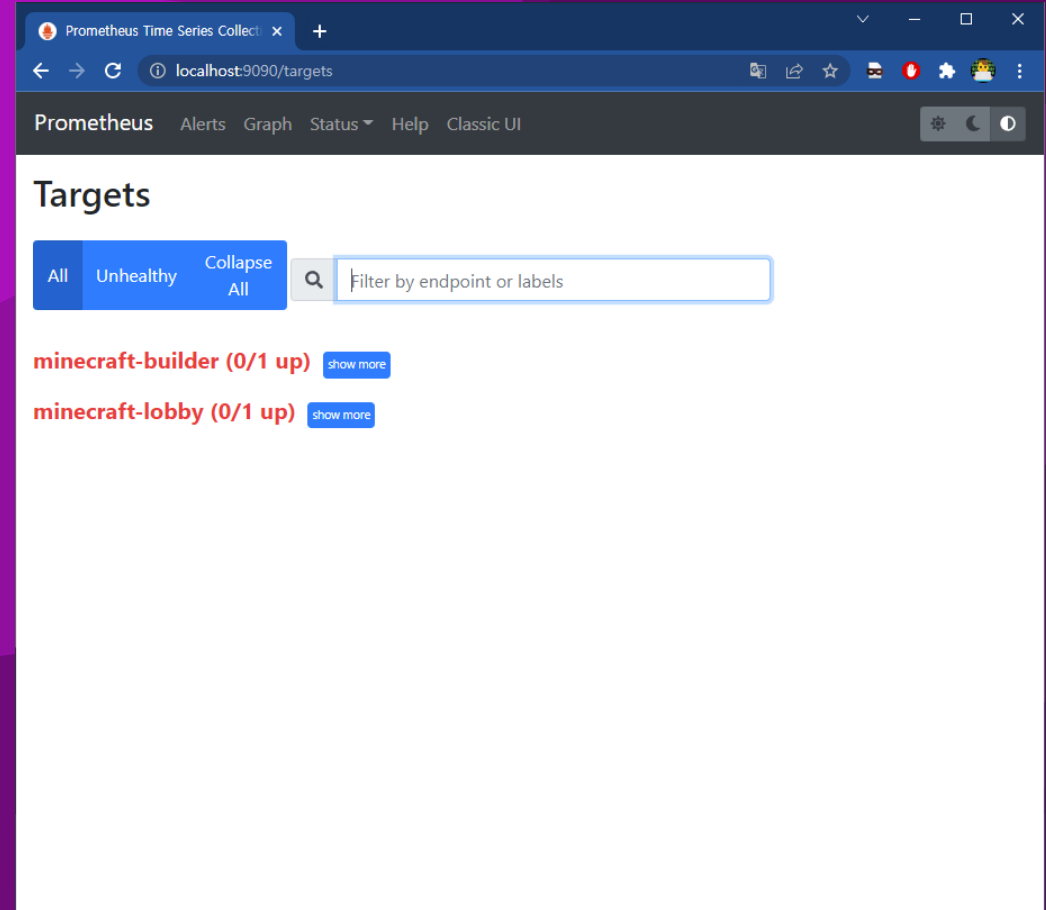

[실습] Prometheus와 마인크래프트 연동

- `http://localhost:9090` 으로 접속한 후,
- Status > Targets 메뉴로 들어가보자.



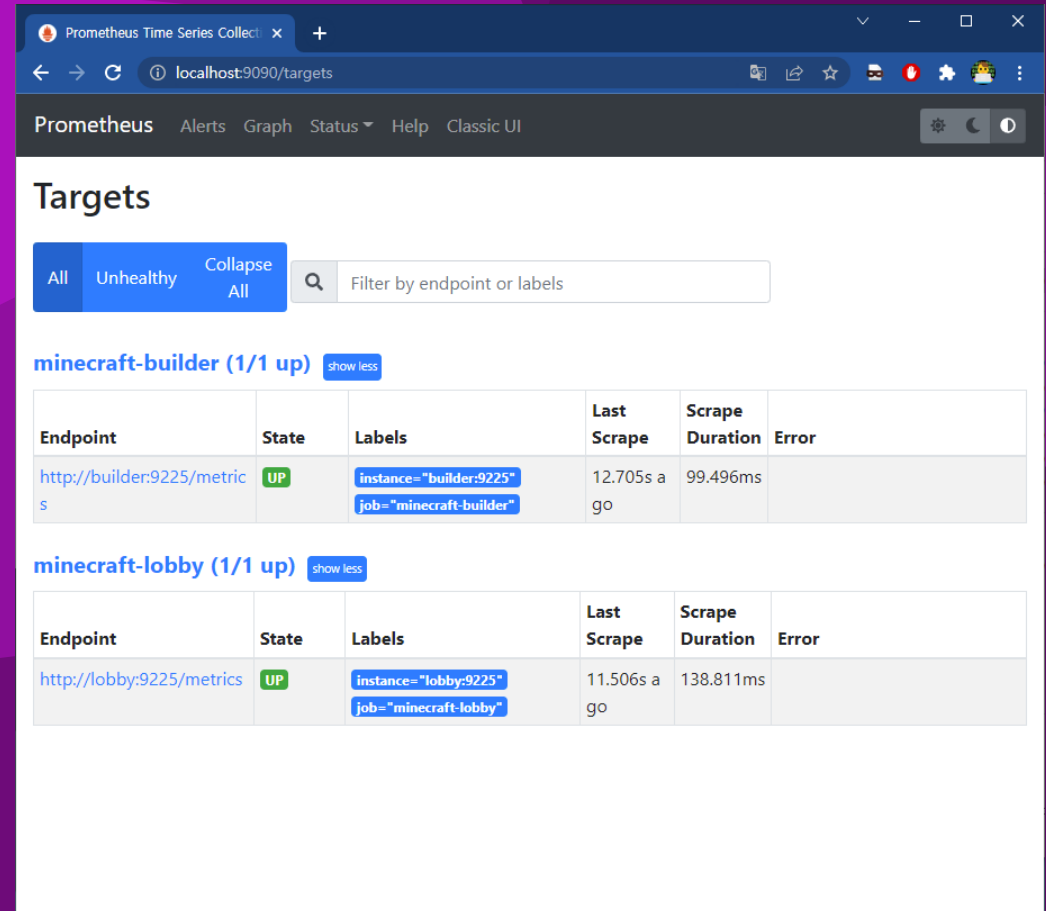
[실습] Prometheus와 마인크래프트 연동

- 만일 설정이 제대로 안되었거나 연결에 문제가 발생하였다면 오른쪽과 같이 빨간 글자로 표시될 것이다.



[실습] Prometheus와 마인크래프트 연동

- 연동이 잘 된 모습이다.



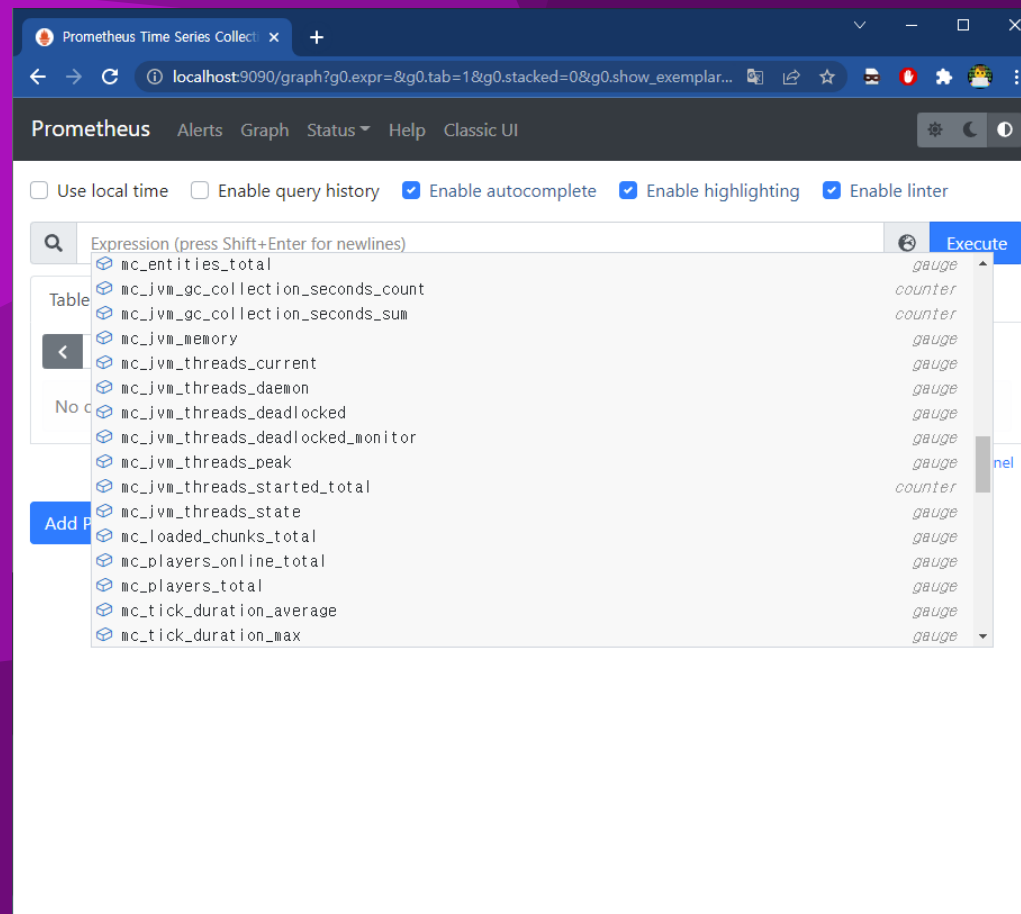
The screenshot shows the Prometheus web interface at localhost:9090/targets. The page displays two target groups, both of which are 'UP'. The first group, 'minecraft-builder', has one target with the endpoint 'http://builder:9225/metrics'. The second group, 'minecraft-lobby', also has one target with the endpoint 'http://lobby:9225/metrics'. Both targets show their last scrape time and duration.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://builder:9225/metrics	UP	instance="builder:9225" job="minecraft-builder"	12.705s ago	99.496ms	

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://lobby:9225/metrics	UP	instance="lobby:9225" job="minecraft-lobby"	11.506s ago	138.811ms	

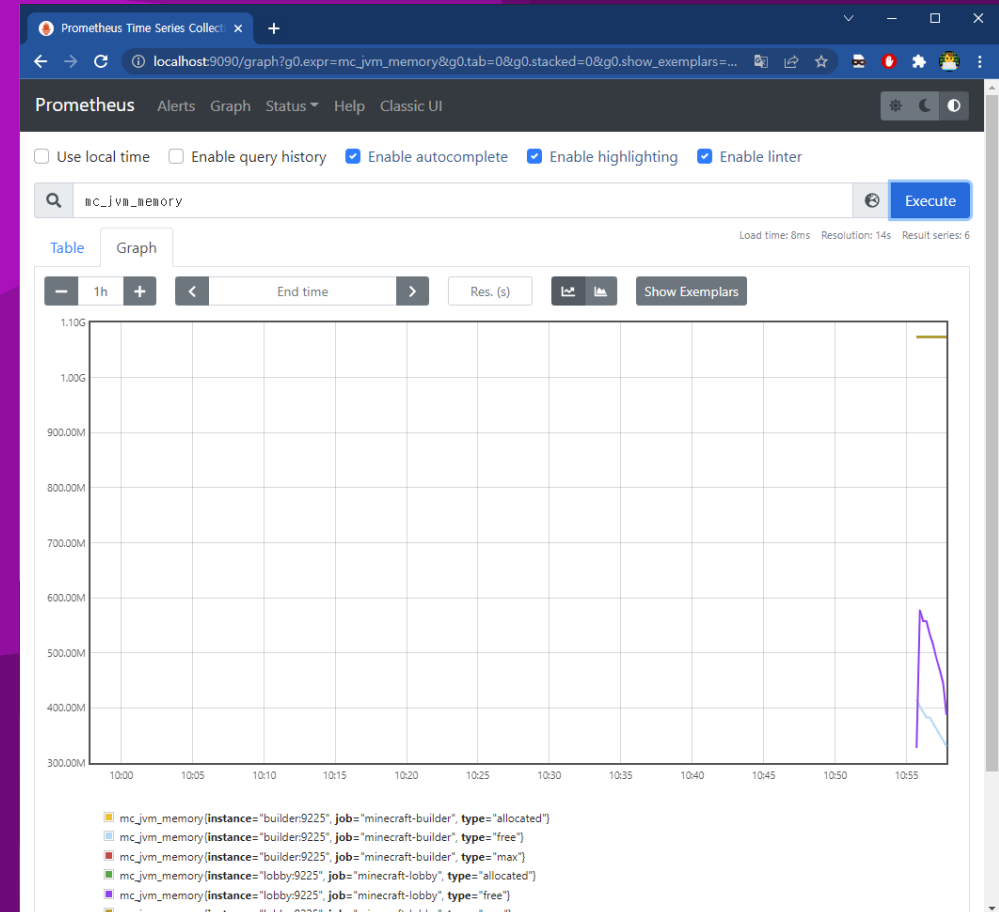
[실습] Prometheus와 마인크래프트 연동

- 연동을 성공적으로 마친 후 Graph에서 메트릭을 조회하면 오른쪽과 같이 잘 나타나는 것을 확인할 수 있다.



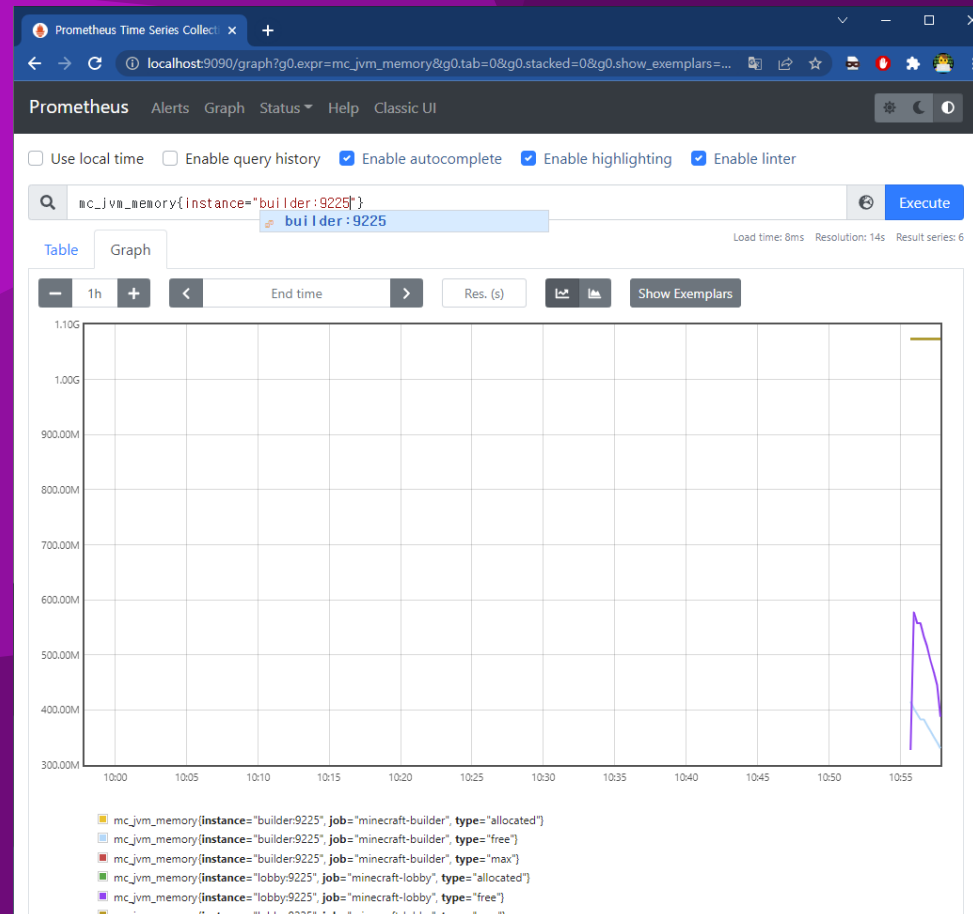
[실습] Prometheus 쿼리

- mc_jvm_memory 메트릭을 조회해보자.
- 오른쪽 그림처럼 차트가 나타날 것이다.



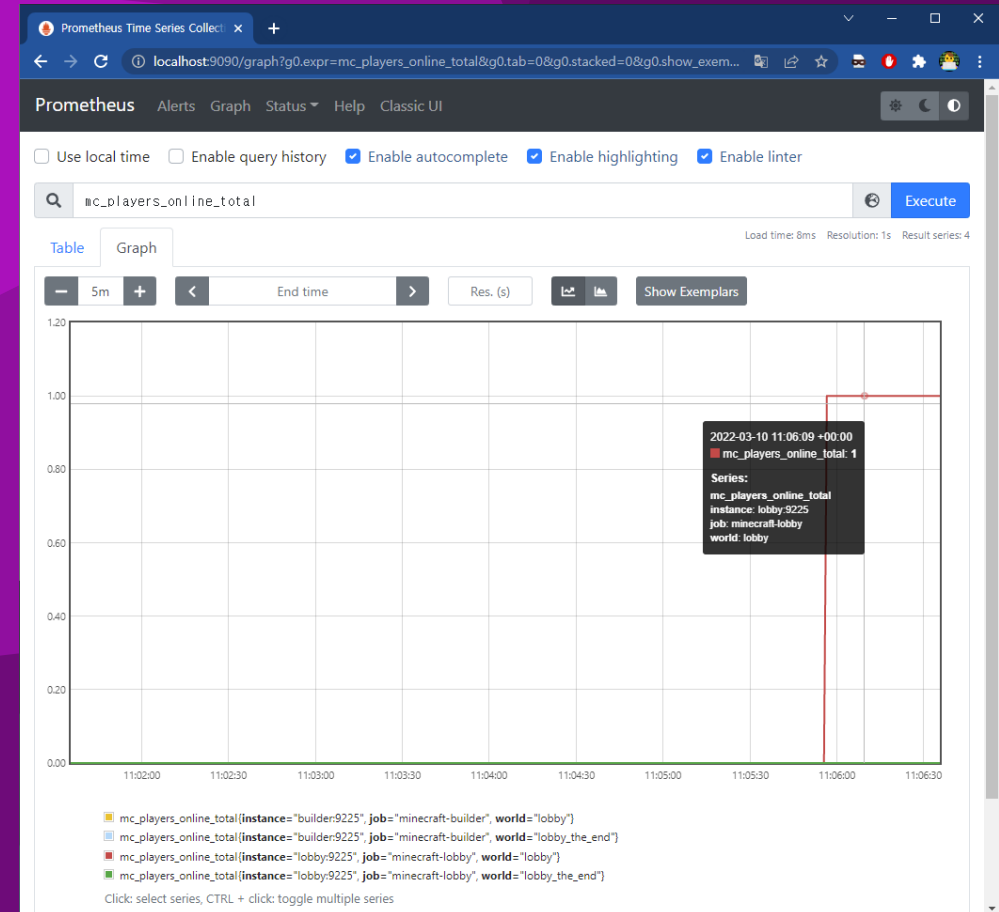
[실습] Prometheus 쿼리

- mc_jvm_memory 부분은 메트릭이며,
- 중괄호 {} 내에 있는 것들은 레이블(label)이다.
- mc_jvm_memory에는 총 3종류의 레이블이 있다.
 - instance: 인스턴스, 즉 서버를 뜻한다.
 - job: prometheus.yml에 등록해놓은 job의 명칭이다.
 - type: 메모리 공간 사용 유형이다.
- 쿼리 입력 란에서, 중괄호 {}를 사용하여 레이블을 필터링할 수 있다.



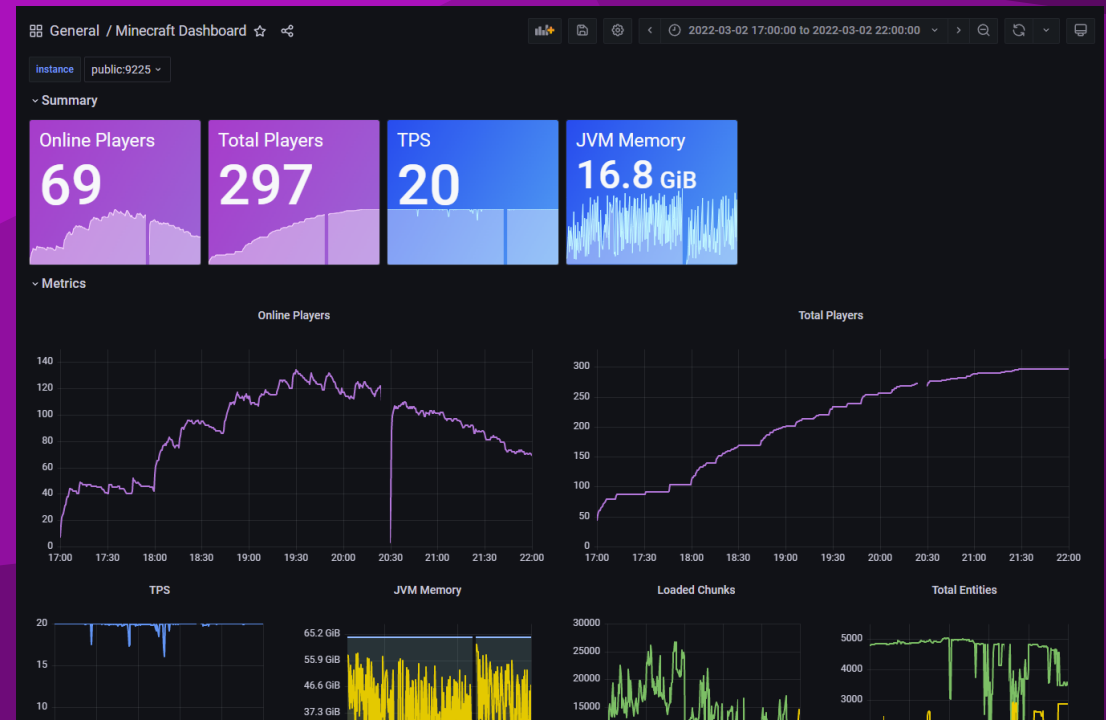
[실습] Prometheus 쿼리

- 서버에 접속한 후, mc_players_online_total을 조회해보자.



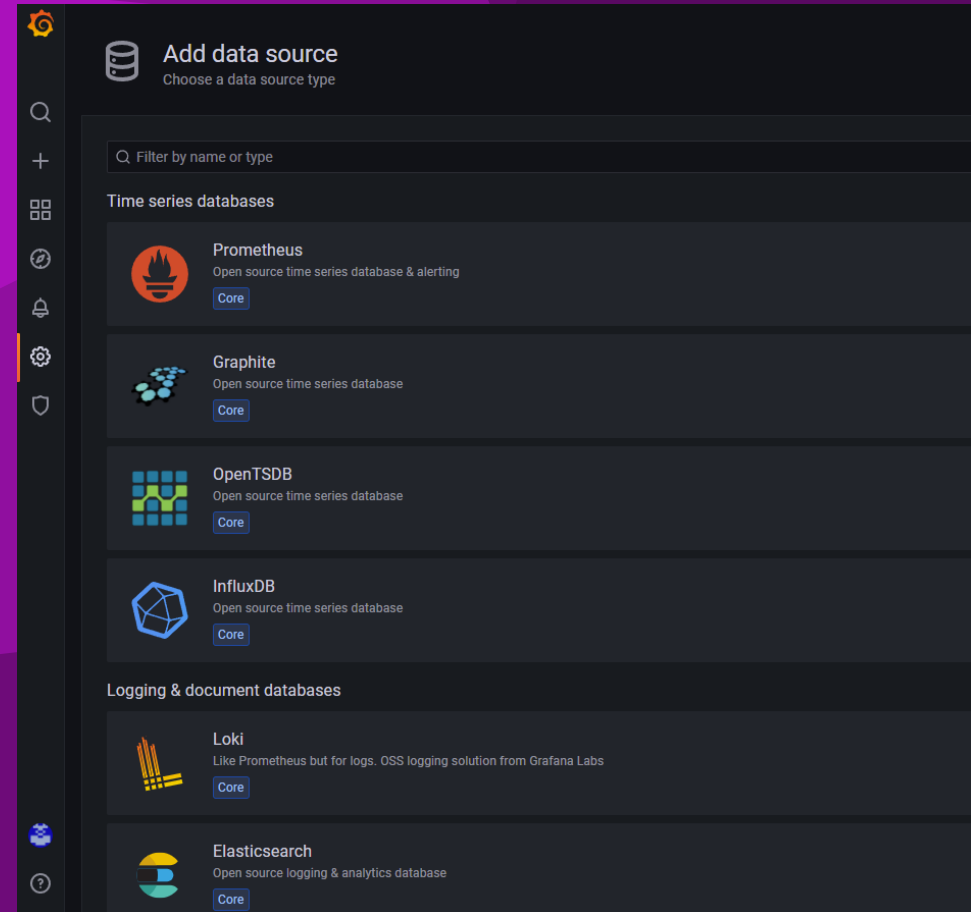
Grafana란?

- 앞서 Prometheus의 웹 인터페이스를 통해 시계열 데이터를 간단하게 조회할 수 있었다.
- Grafana는 Gauge, Bar Chart, Heatmap 등의 다양한 시각화 도구와 커스터마이징을 지원한다.
- 레이아웃을 구성하고 데이터를 적절히 배치할 수 있다.



Grafana란?

- Grafana는 시계열 데이터베이스부터 RDBMS까지 다양한 데이터베이스를 지원한다.
 - 시계열: Prometheus, Graphite, OpenTSDB, InfluxDB
 - RDBMS: MySQL, PostgreSQL, Microsoft SQL Server
 - Cloud: Google Cloud Monitoring, CloudWatch, Azure Monitor, Grafana Cloud
- 이들은 Data Source라고 한다.



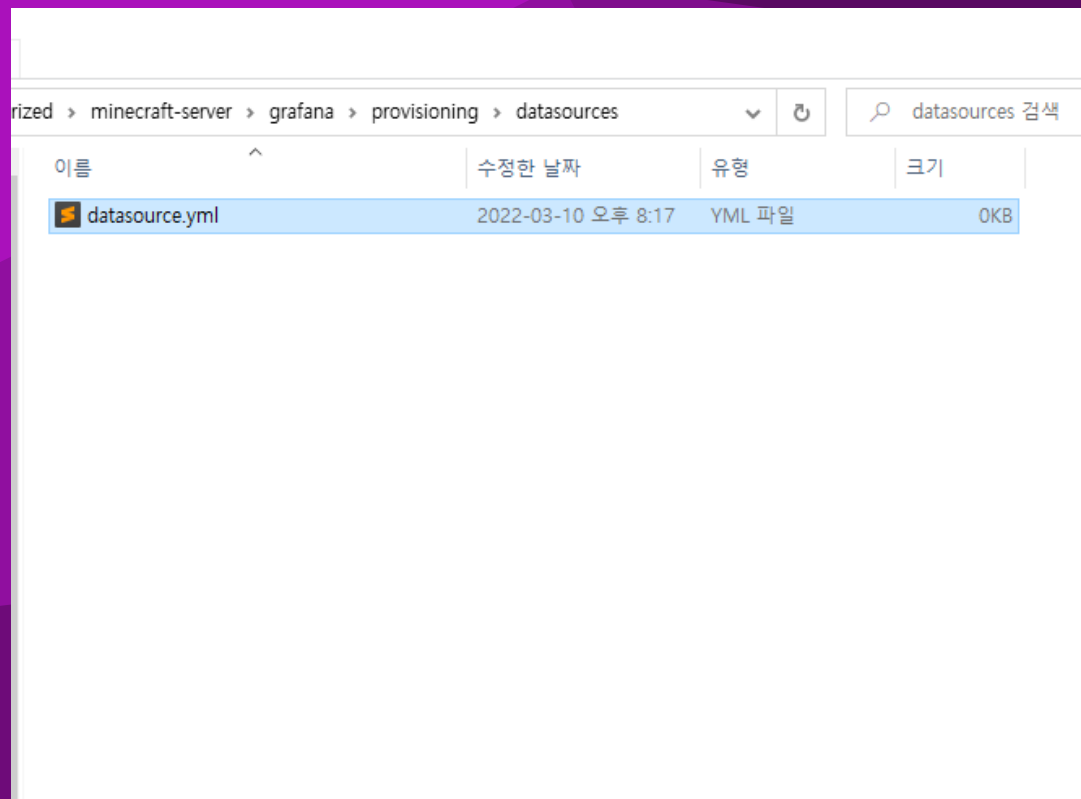
[실습] Grafana 설치

- docker-compose.yml의 Prometheus 서비스 아래에 Grafana 서비스를 추가한다.
- provisioning 폴더에는 Data Source를 추가할 수 있다.
- 외부 유저가 가입을 하고 데이터를 열람하는 것을 방지하기 위해 GF_USERS_ALLOW_SIGN_UP 옵션을 꺼둔다.

```
98
99 ▼ grafana:
100     container_name: grafana
101     image: grafana/grafana
102     restart: unless-stopped
103     user: 1000:1000
104     ports:
105         - 3000:3000
106 ▼     volumes:
107         - ./grafana/provisioning:/etc/grafana/provisioning
108         - ./grafana/data:/var/lib/grafana
109 ▼     environment:
110         - GF_SECURITY_ADMIN_USER=myuser
111         - GF_SECURITY_ADMIN_PASSWORD=myspassword
112         - GF_USERS_ALLOW_SIGN_UP=false
113
```

[실습] Grafana 설치

- grafana/provisioning/datasources 폴더를 만들고,
- 해당 폴더에 datasource.yml 파일을 생성한다.



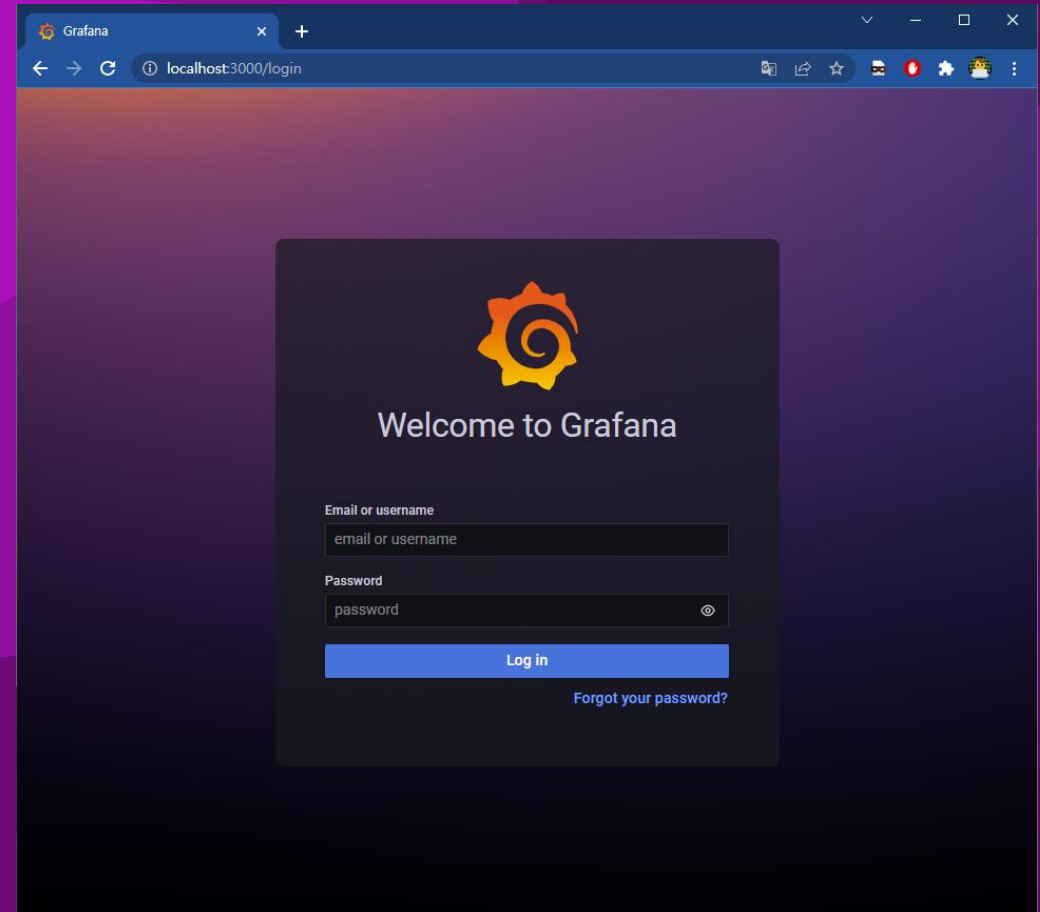
[실습] Grafana 설치

- datasource.yml의 내용을 오른쪽과 같이 작성한 후 저장한다.
- 그리고 docker-compose up -d grafana 명령을 입력하여 Grafana 컨테이너를 실행시켜준다.

```
1  apiVersion: 1
2
3  datasources:
4    - name: Prometheus
5      type: prometheus
6      access: proxy
7      orgId: 1
8      url: http://prometheus:9090
9      basicAuth: false
10     isDefault: true
11     editable: true
12
```

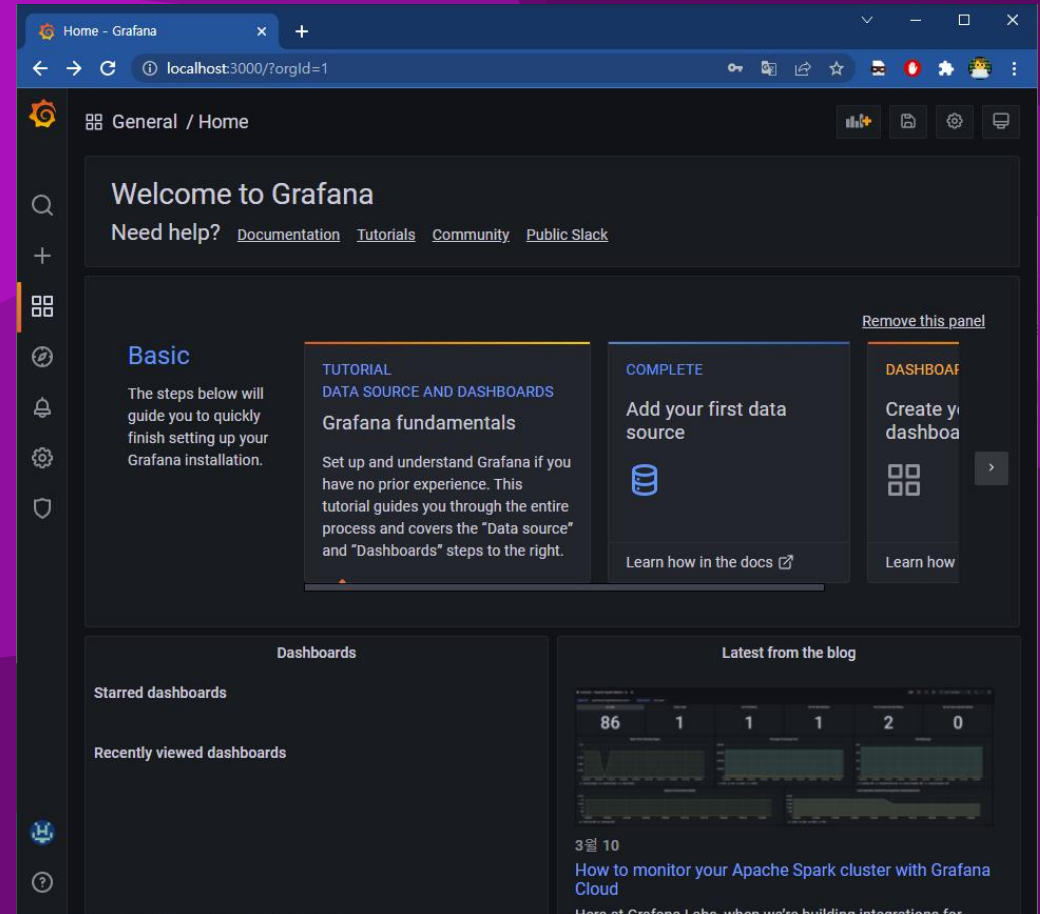
[실습] Grafana 설치

- `http://localhost:3000` 으로 접속하면 Grafana 로그인 화면을 볼 수 있다.
- `docker-compose.yml` 에서 설정한 계정과 비밀번호로 접속한다.



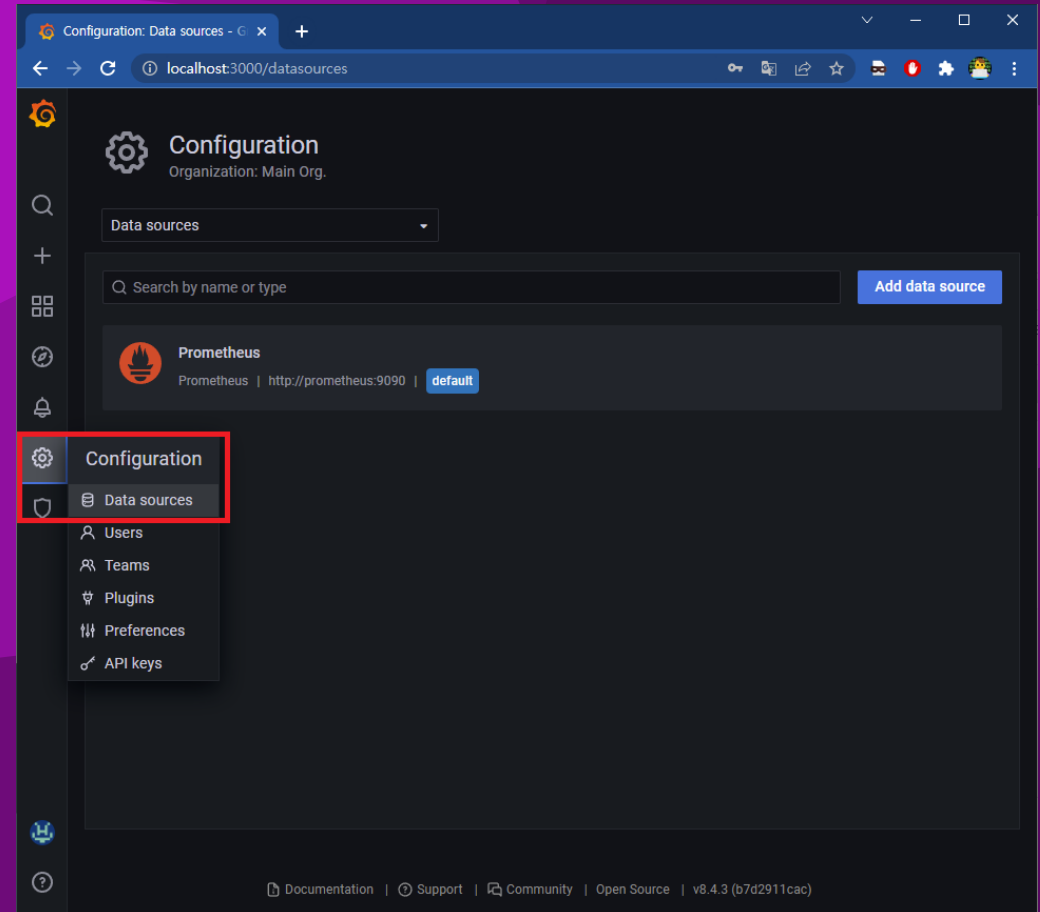
[실습] Grafana 설치

- 성공적으로 로그인 되었다면 오른쪽과 같은 화면을 볼 수 있을 것이다.



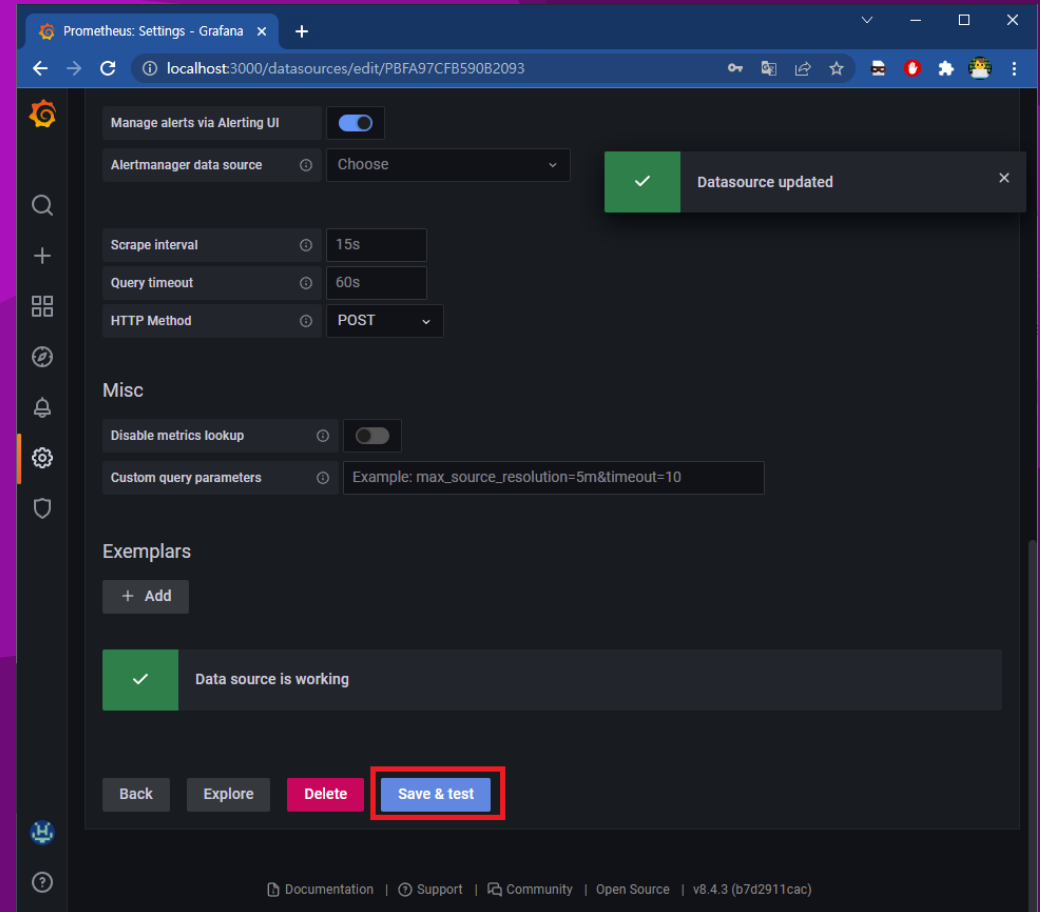
[실습] Grafana 설치

- 설정 아이콘 > Data sources 로 이동하면 Data Source 목록을 확인할 수 있다.
- Prometheus를 클릭한다.



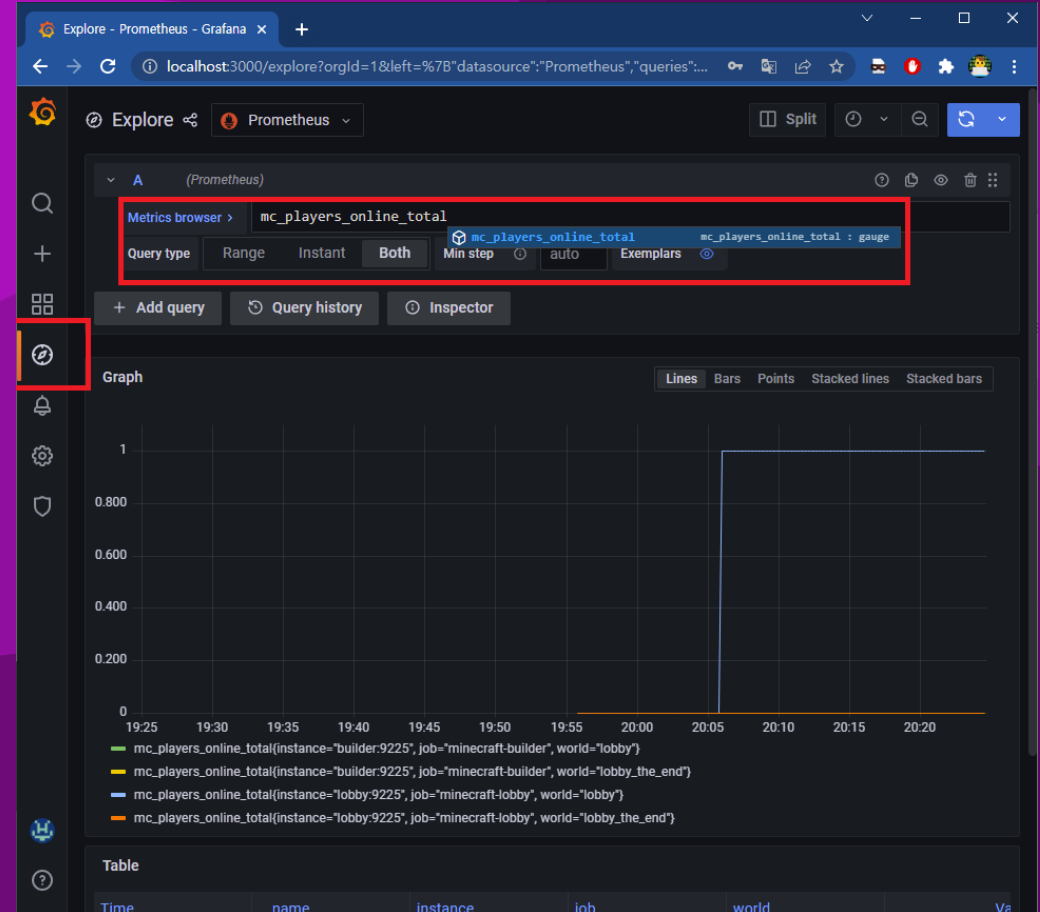
[실습] Grafana 설치

- 가장 아래에 Save & test가 있다.
- 이 버튼을 클릭하여 Data Source가 잘 작동하는지 확인할 수 있다.



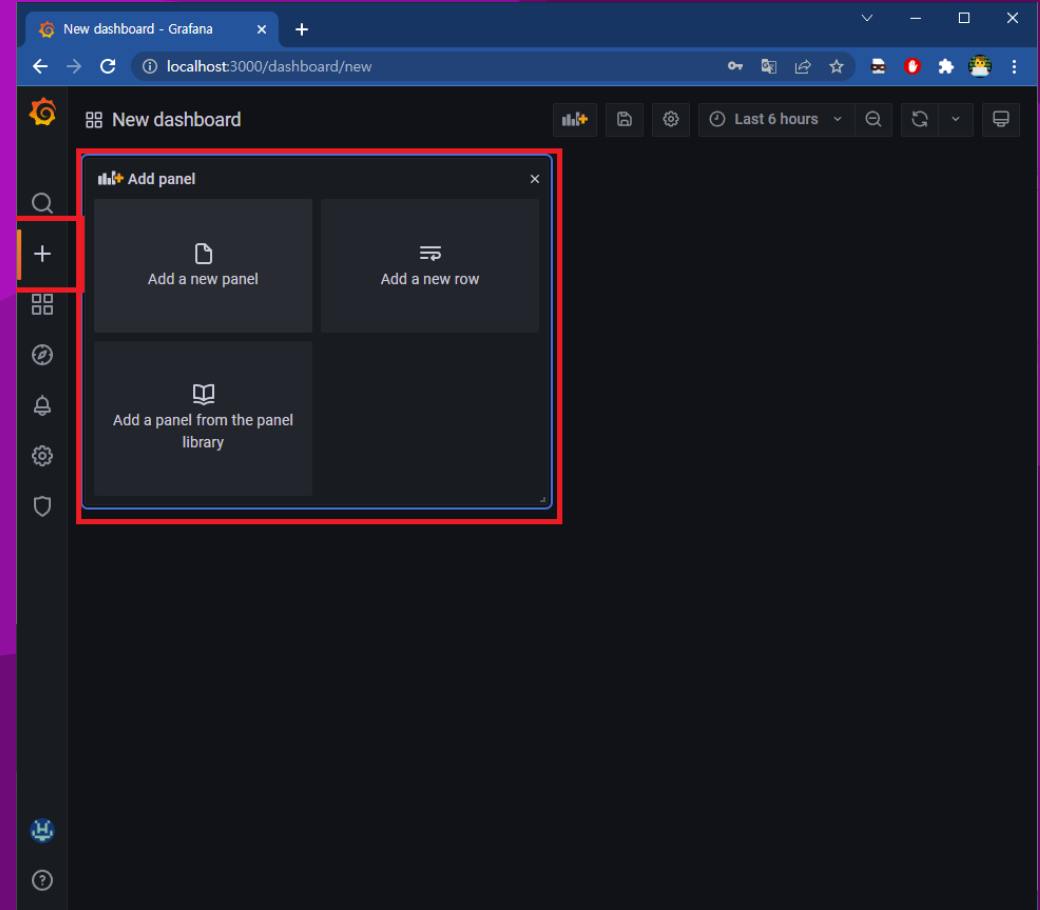
[실습] Grafana 설치

- 왼쪽의 나침반 아이콘을 클릭해보자.
 - Explorer라는 기능으로, 메트릭을 조회할 수 있다.
- 쿼리 입력란에 mc_players_online_total을 입력해보자.
 - Ctrl + Space로 자동완성 기능을 사용할 수 있다.
- 오른쪽 위에 있는 새로고침 버튼을 누르면 쿼리를 조회할 수 있다.



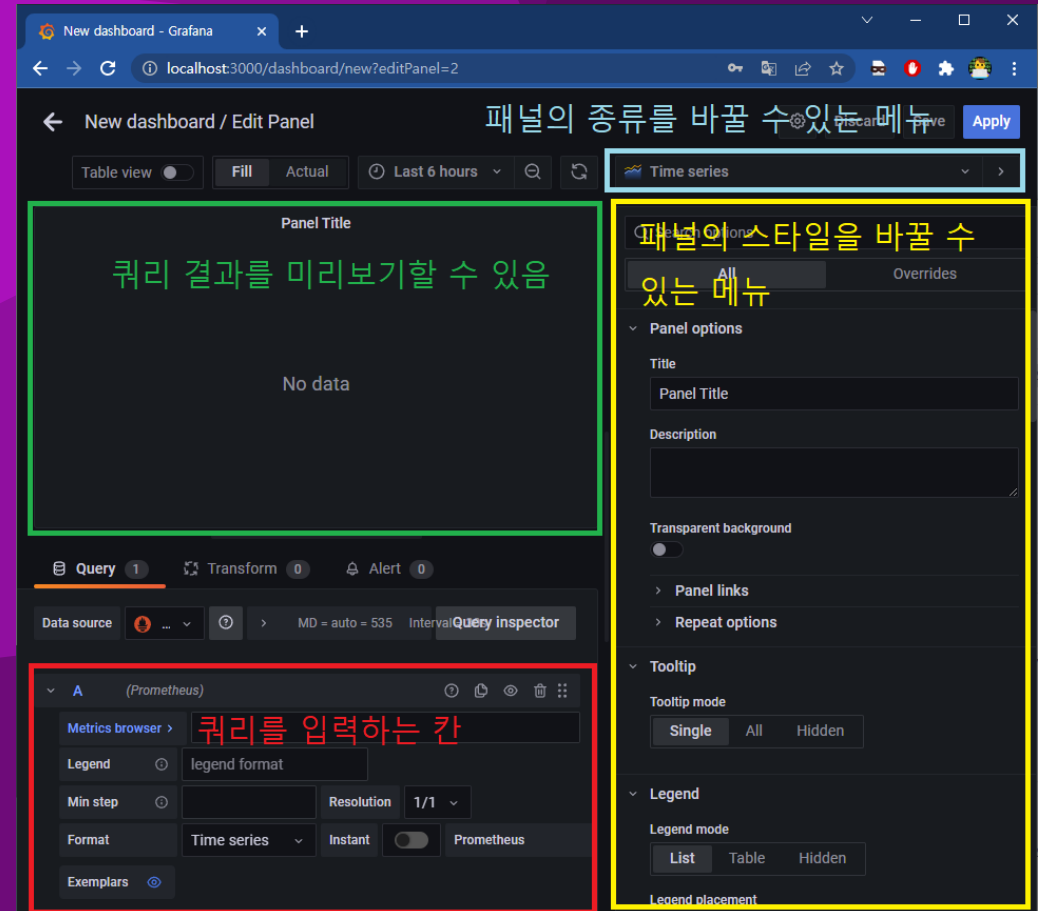
[실습] Grafana Dashboard 만들기

- 왼쪽의 + 모양의 아이콘을 클릭하면 새 대시보드를 만들 수 있다.
- Add a new panel 을 클릭하여 패널을 추가해보자.



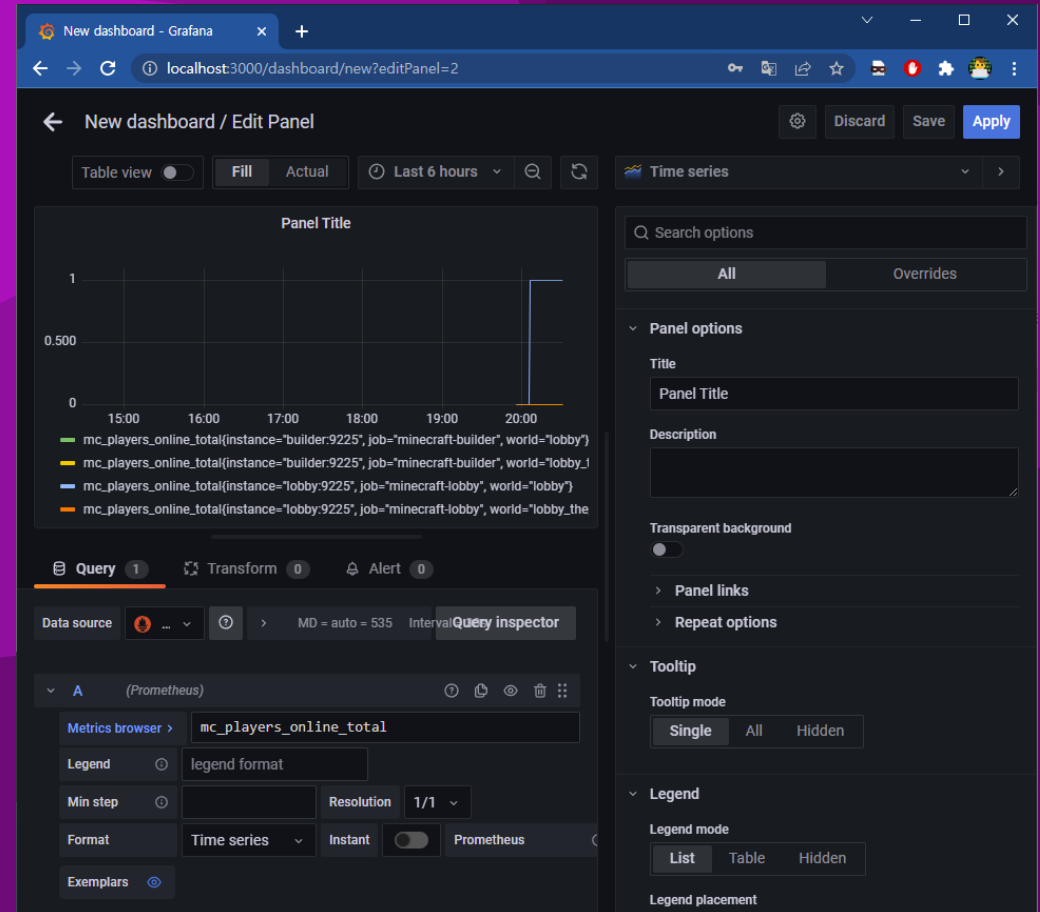
[실습] Grafana Dashboard 만들기

- 패널을 편집하는 창이 나타난다.
- 빨간색 박스 내의 입력 란을 사용하여 쿼리를 입력할 수 있다.



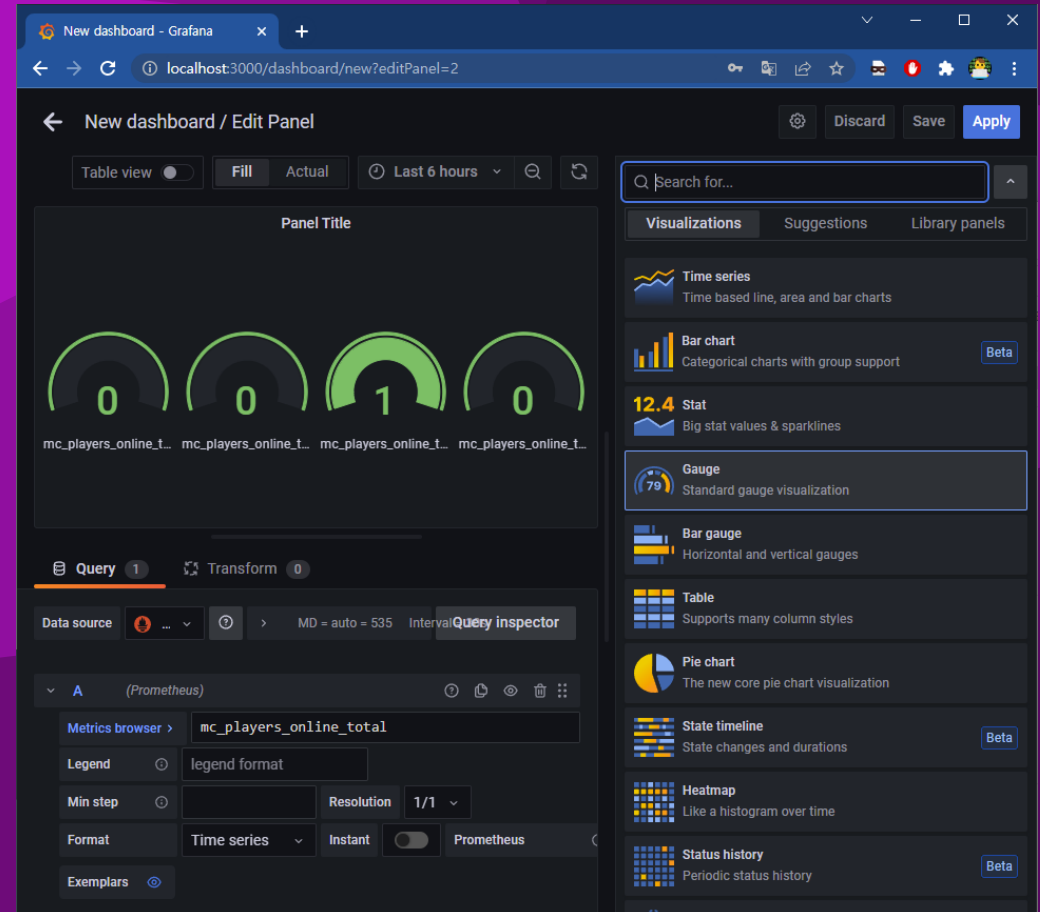
[실습] Grafana Dashboard 만들기

- 쿼리 입력 칸에는 Prometheus에서 사용하였던 쿼리를 그대로 입력하면 된다.
- mc_players_online_total을 입력해보자.
- 플레이 중인 유저 수가 표시된다.



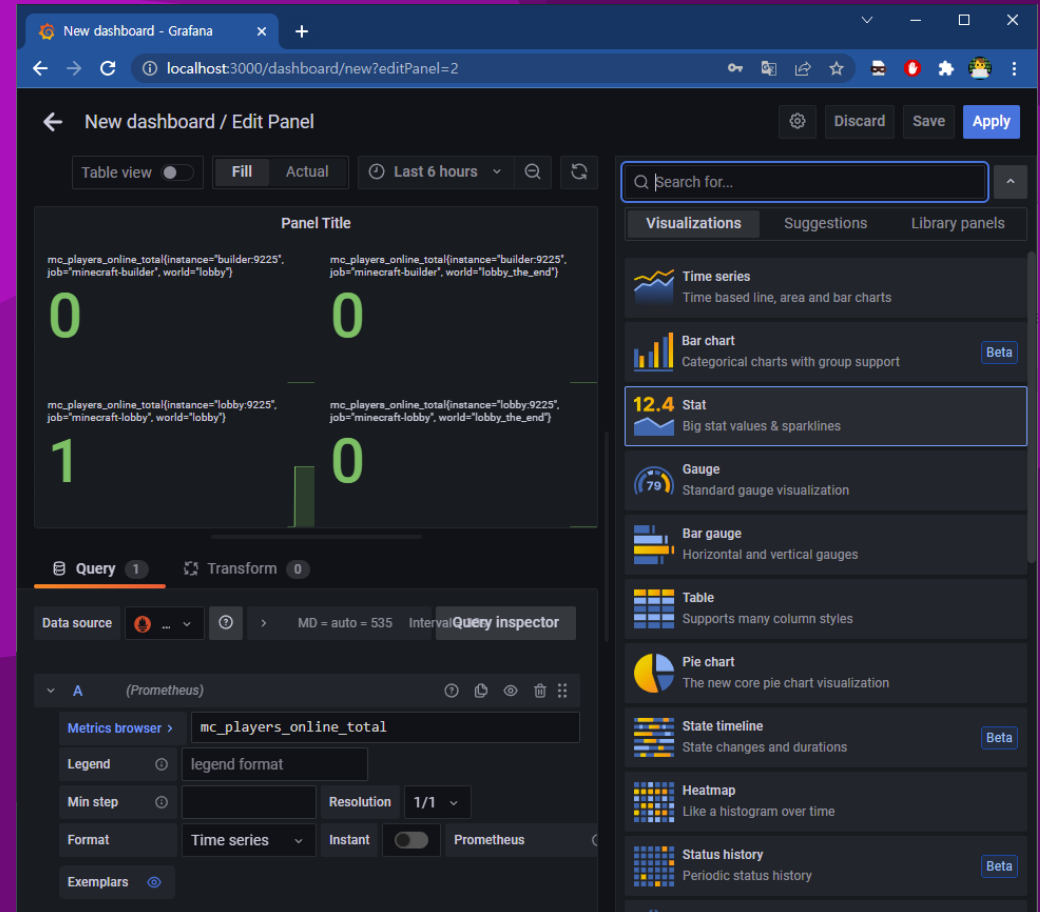
[실습] Grafana Dashboard 만들기

- 패널의 종류를 Gauge로 바꿔보자.
- 자동차의 속도계와 같은 모양을 볼 수 있다.



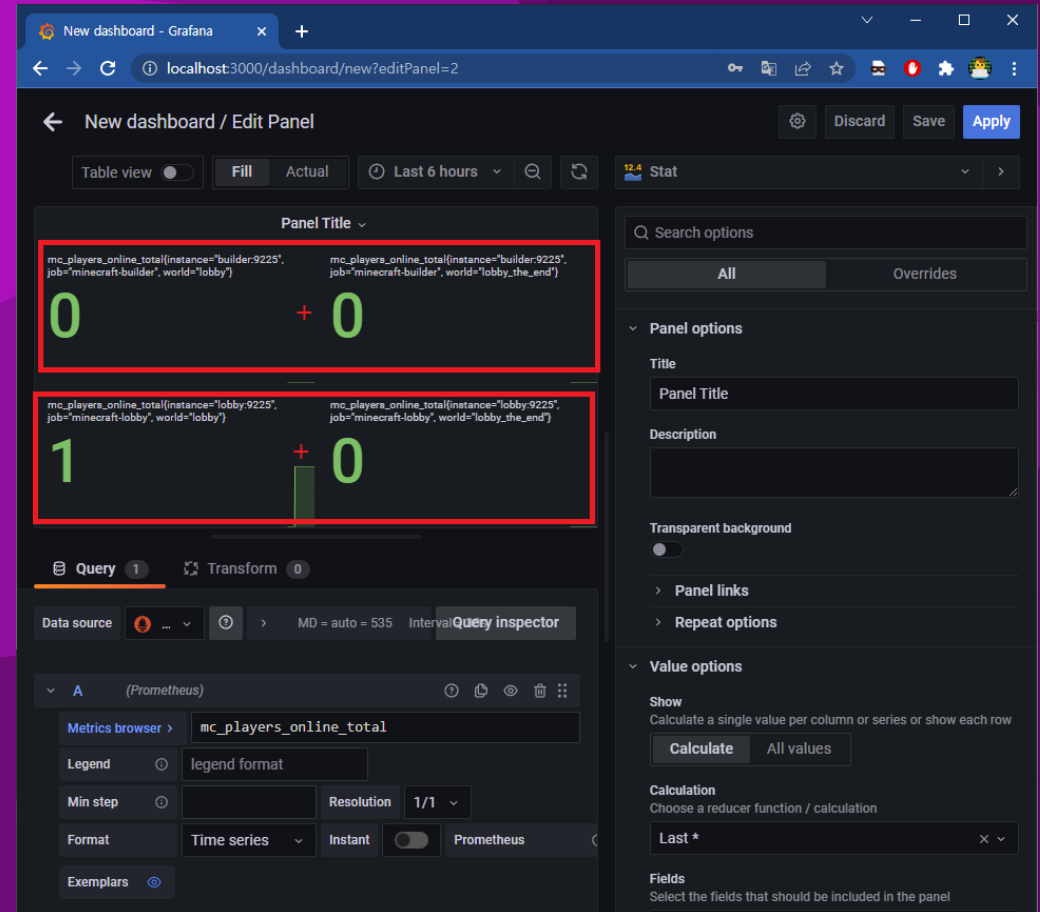
[실습] Grafana Dashboard 만들기

- 패널 종류를 Stat으로 바꿔보자.
- 레이블의 종류만큼 Stat이 표시되는 것을 볼 수 있다.



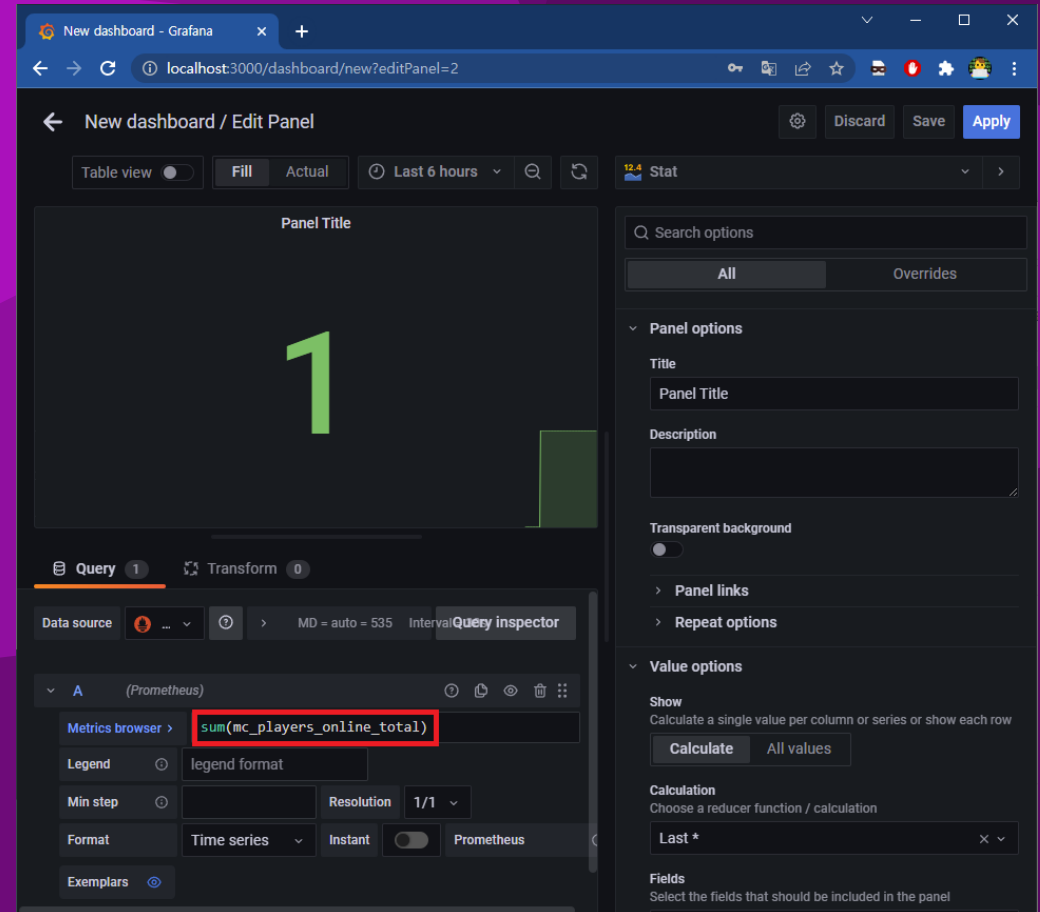
[실습] Grafana Dashboard 만들기

- 잘 보면, world별 플레이어, 서버 별 플레이어로 총 4개의 Stat이 표시되는 것을 확인할 수 있다.
- world별 플레이어는 모두 합하여, 결과적으로 서버 별 플레이어를 표시해보도록 하자.



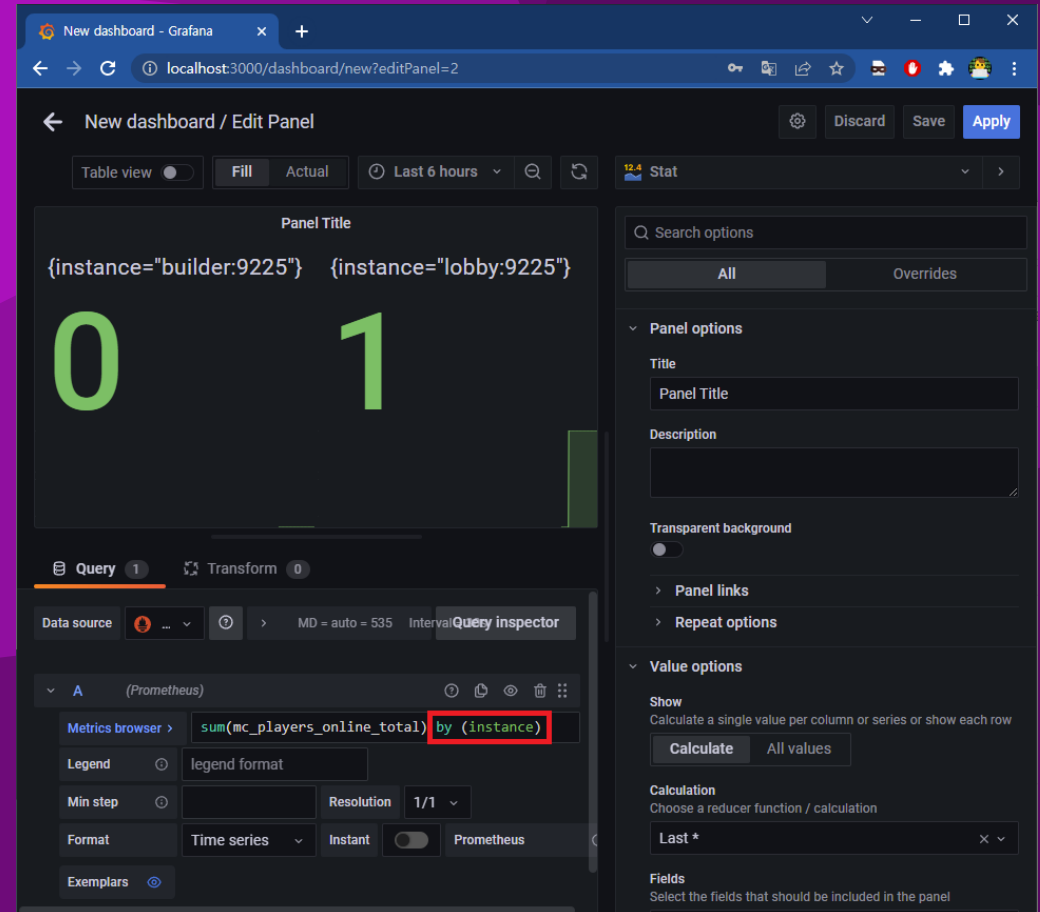
[실습] Grafana Dashboard 만들기

- 우선, 모든 동점을 합할 수 있다.
- sum 함수를 사용한다.
- sum(메트릭) 과 같이 사용하면 모든 레이블 값을 합하여 출력한다.



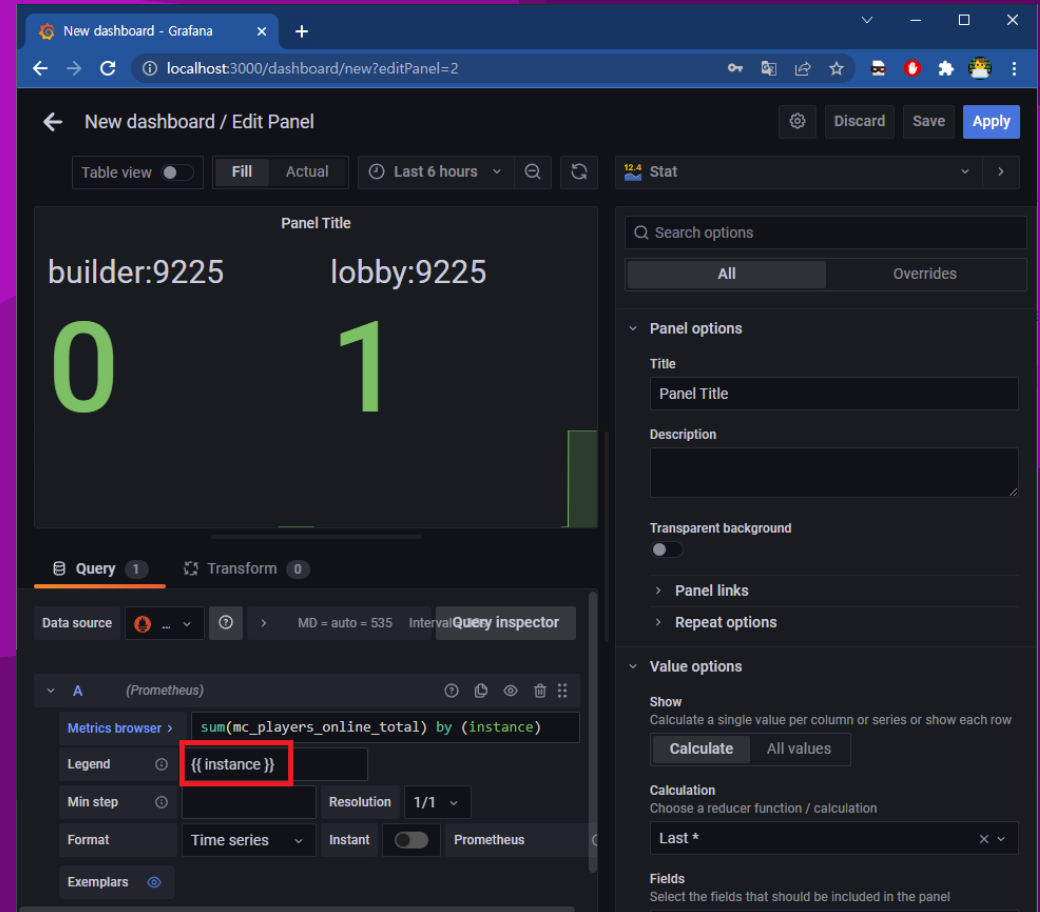
[실습] Grafana Dashboard 만들기

- `sum(mc_players_online_total)` 뒤에 `by` 를 붙이고 레이블을 적으면 해당 레이블을 기준으로 집계 (aggregate)된다.
- `sum()` 뒤에 `by (instance)` 를 붙이면, 인스턴스(서버) 기준으로 집계되는 것을 볼 수 있다.



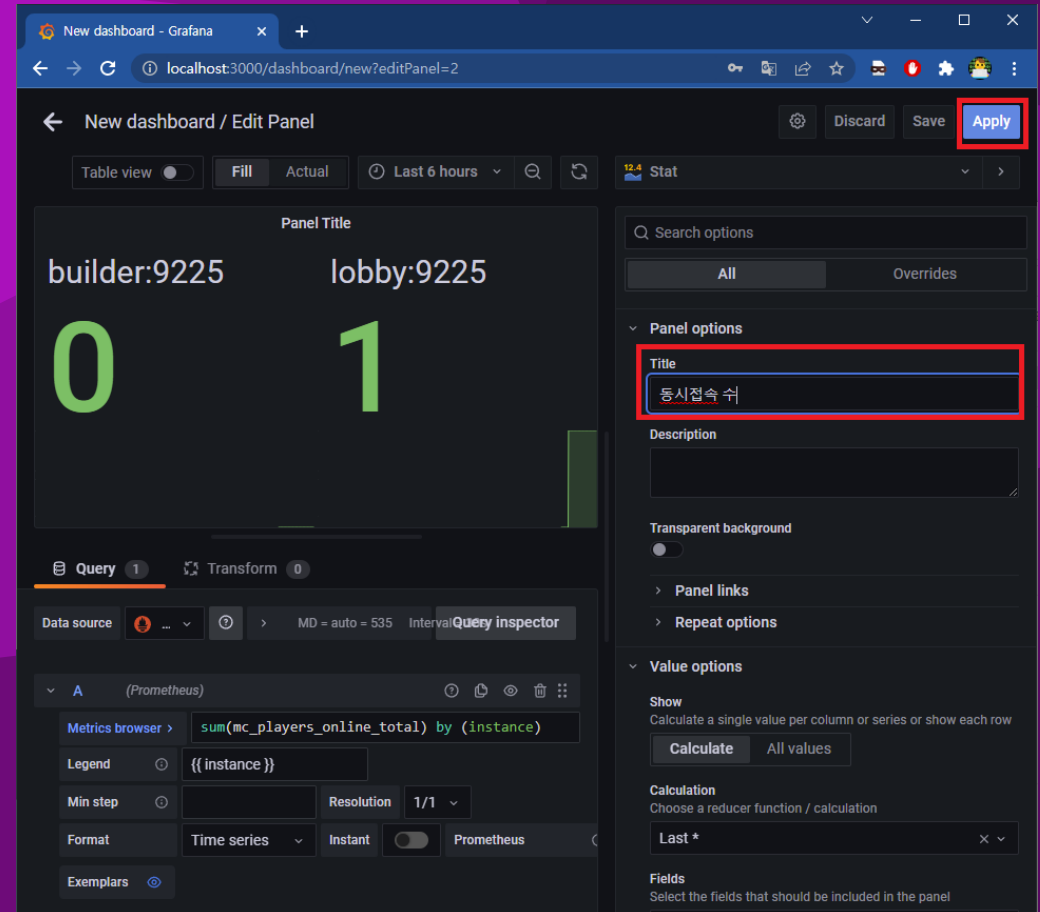
[실습] Grafana Dashboard 만들기

- 상단의 텍스트를 좀 더 간략하게 나타내보자.
- 위의 텍스트는 legend라고 한다.
- legend를 수정하여 표시되는 형식을 수정할 수 있다.
- 중괄호를 두 번 열고 레이블을 넣으면 해당 레이블이 표시된다.
 - 예) `{{ instance }}` -> builder:9225
 - 예) `{{ world }}` -> lobby_the_end



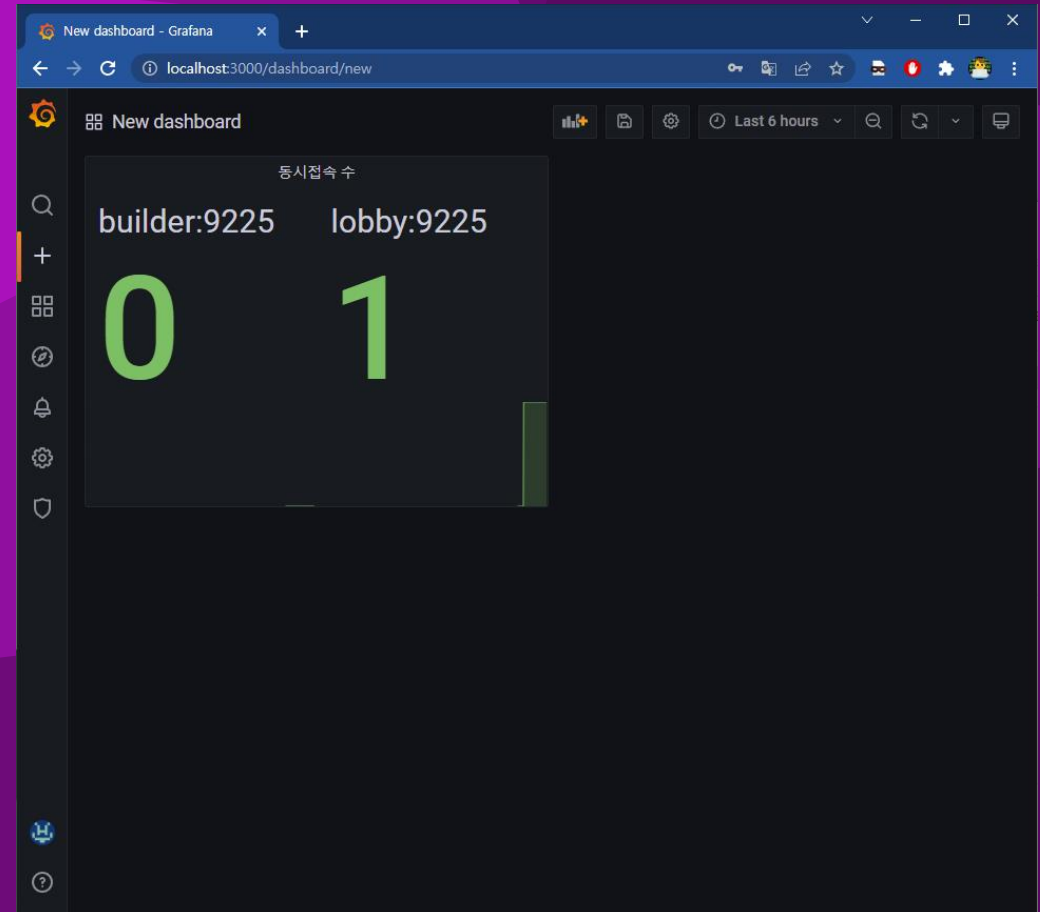
[실습] Grafana Dashboard 만들기

- 패널의 제목을 입력하고 저장해보자.



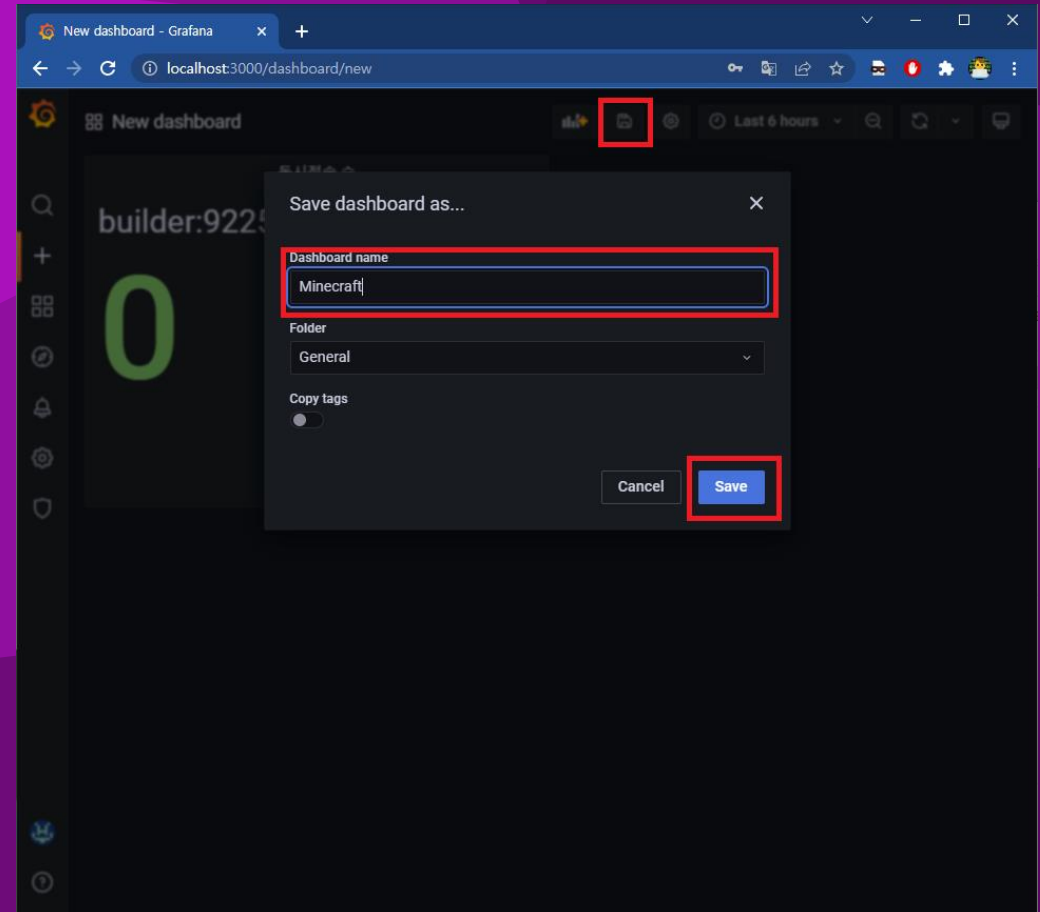
[실습] Grafana Dashboard 만들기

- 대시보드 화면에서 패널이 성공적으로 저장된 것을 확인할 수 있다.



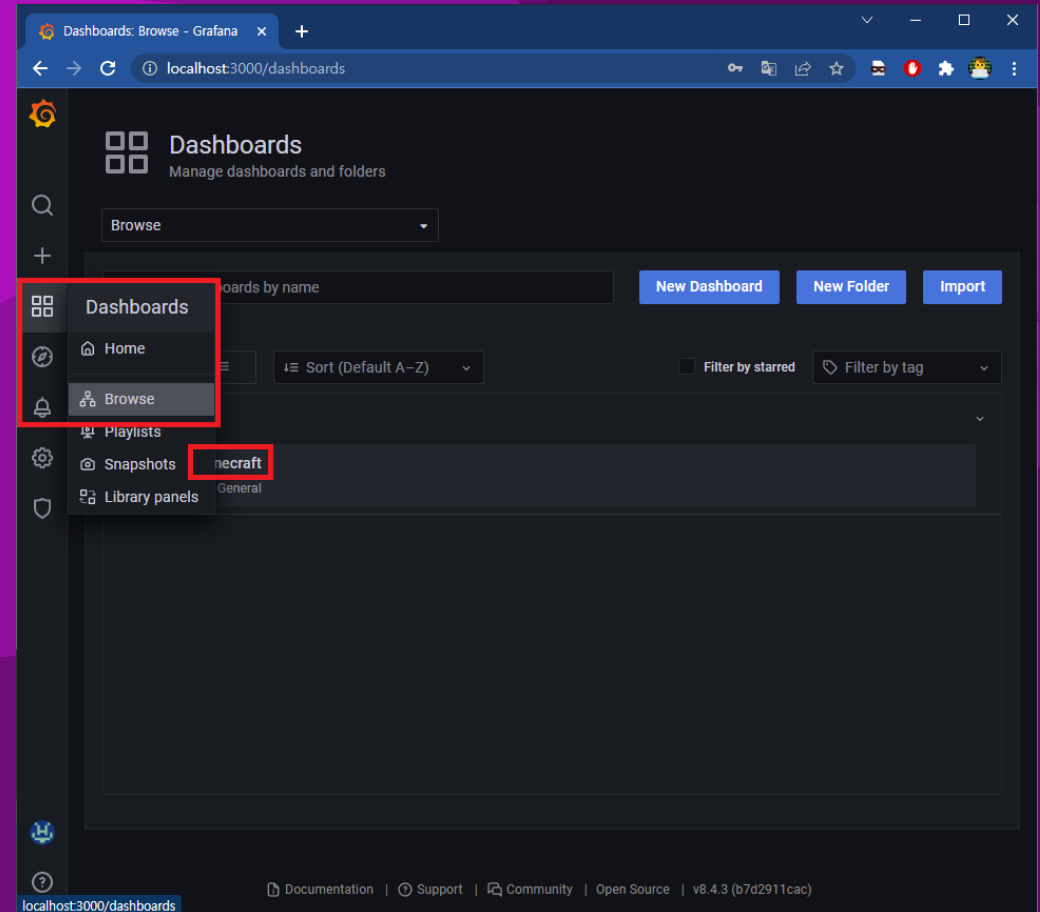
[실습] Grafana Dashboard 만들기

- 저장 버튼을 눌러 대시보드를 저장하자.



[실습] Grafana Dashboard 만들기

- 저장된 대시보드는 대시보드 목록 메뉴에서 확인할 수 있다.



최종 네트워크 다이어그램

