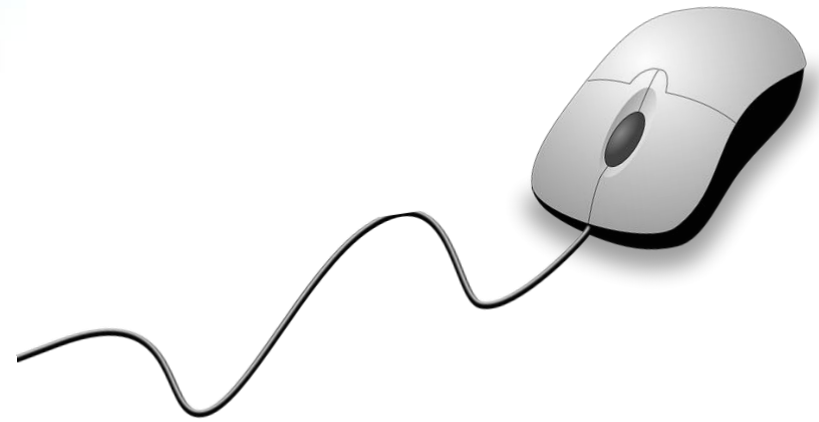


공개SW 솔루션 설치 & 활용 가이드

시스템SW > 자원관리



제대로 배워보자

How to Use Open Source Software

---

Open Source Software Installation & Application Guide



# CONTENTS

1. 개요
2. 기능요약
3. 실행환경
4. 설치 및 실행
5. 기능소개
6. 활용예제
7. FAQ
8. 용어정리

# 1. 개요



<b>소개</b>	<ul style="list-style-type: none"> <li>• 2012년 SoundCloud 에서 개발된 시스템 모니터링 및 경고 툴킷</li> <li>• 2016년 Kubernetes 이후 두번째로 Cloud Native Computing Foundation 에 편입됨</li> </ul>		
<b>주요기능</b>	<ul style="list-style-type: none"> <li>• Metric 이름과 키 / 값 쌍으로 식별되는 시계열데이터가 있는 다차원 데이터 모델</li> <li>• PromQL (Prometheus Query Language)이라는 기능적 쿼리 언어를 제공</li> <li>• 다양한 그래프 및 대시 보드 지원 모드</li> <li>• Target은 서비스 검색 또는 정적 구성을 통해 검색</li> </ul>		
<b>대분류</b>	<ul style="list-style-type: none"> <li>• 시스템SW</li> </ul>	<b>소분류</b>	<ul style="list-style-type: none"> <li>• 자원관리</li> </ul>
<b>라이선스형태</b>	<ul style="list-style-type: none"> <li>• Apache License v2.0</li> </ul>	<b>사전설치 솔루션</b>	<ul style="list-style-type: none"> <li>• 없음</li> </ul>
		<b>버전</b>	<ul style="list-style-type: none"> <li>• 2.11.0 (2019년 7월 기준)</li> </ul>
<b>특징</b>	<ul style="list-style-type: none"> <li>• Prometheus는 주로 Go로 작성</li> <li>• Go,Java/JVM, C#/Net, Python, Ruby, Node.js, Haskell, Erlang, Rust등의 언어와 런타임에 클라이언트 라이브러리 지원</li> <li>• Kubernetes 및 Docker 에는 Prometheus 클라이언트 라이브러리가 설치되어 있음</li> <li>• Grafana 와 같은 대시보드 시스템에서 그래프로 나타낼 수 있음</li> </ul>		
<b>개발회사/커뮤니티</b>	<ul style="list-style-type: none"> <li>• Cloud Native Computing Foundation</li> </ul>		
<b>공식 홈페이지</b>	<ul style="list-style-type: none"> <li>• <a href="https://prometheus.io/">https://prometheus.io/</a></li> </ul>		

## 2. 기능요약



- Prometheus 주요 기능

다차원 데이터	<ul style="list-style-type: none"><li>• 고도의 차원 데이터 모델을 구현합니다. 시계열은 Metric 이름과 키 - 값 쌍으로 식별됩니다.</li></ul>
강력한 쿼리	<ul style="list-style-type: none"><li>• PromQL을 사용하면 임시 그래프, 테이블 및 경고를 생성하기 위해 수집된 시계열 데이터를 분할 및 다이싱 할 수 있습니다.</li></ul>
시각화	<ul style="list-style-type: none"><li>• 내장된 표현 브라우저, Grafana 통합 및 콘솔 템플릿 언어와 같이 시각화를 위한 여러 모드가 있습니다.</li></ul>
효율적인 저장	<ul style="list-style-type: none"><li>• 효율적인 사용자 정의 형식으로 메모리 및 로컬 디스크에 Time series를 저장합니다. 스케일링은 Functional sharding 및 federation으로 달성됩니다.</li></ul>
간단한 조작	<ul style="list-style-type: none"><li>• 각 서버는 안정성을 위해 독립적이며 로컬 저장소에만 의존합니다. Go로 작성된 모든 바이너리는 정적으로 연결되어 배포가 용이합니다.</li></ul>

# 3. 실행환경



- OS 플랫폼 종류에 따른 지원
  - Linux: 32-bit, 64-bit, and ARM.
  - Mac OS X: 32-bit and 64-bit.
  - FreeBSD: 32-bit, 64-bit, and ARM.
  - OpenBSD: 32-bit, 64-bit, and ARM.
  - NetBSD: 32-bit, 64-bit, and ARM.
  - Microsoft Windows: 32-bit and 64-bit.
  - DragonFly: 64-bit



# 4. 설치 및 실행



## 세부 목차

- 4.1 Prometheus 서버 설치 및 실행(CentOS – Systemd기반)
- 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)
- 4.3 Alertmanager 설치 및 실행 (CentOS – Systemd기반)



# 4. 설치 및 실행



## 4.1 Prometheus 서버 설치 및 실행(CentOS - systemd)(1/5)

- OS Version : CentOS Linux release 7.6.1810 (Core)
- Prometheus Version : prometheus-2.11.0.linux-amd64
- Prometheus 공식 홈페이지 에서 다운

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

<2019년 7월 10일 기준으로 최신버전이 2.11>

### 1. 소스 파일 다운로드

```
# wget https://github.com/Prometheus/Prometheus/releases/download/v2.11.0/Prometheus-2.11.0.linux-amd64.tar.gz
```

### 2. 소스 파일 압축 해제

```
# tar zxvf prometheus-2.11.0.linux-amd64.tar.gz
```

### 3. 해당 폴더로 이동

```
# cd prometheus-2.11.0.linux-amd64
```

### 4. 버전 확인

```
# ./prometheus --version
```

```
[root@prometheus prometheus-2.11.0.linux-amd64]# ./prometheus --version
prometheus, version 2.11.0 (branch: HEAD, revision: 4ef66003d9855ed2b7a41e987b33828ec36db34d)
 build user:   root@0dc27cf95f36
 build date:   20190709-09:54:35
 go version:   go1.12.7
```



# 4. 설치 및 실행



## 4.1 Prometheus 서버 설치 및 실행(CentOS - systemd)(2/5)

### 5. User 생성

```
# useradd --no-create-home --shell /bin/false prometheus
```

### 6. 필요한 디렉토리 생성

```
# mkdir /etc/prometheus
```

```
# mkdir /var/lib/prometheus
```

### 7. 생성한 디렉토리에 소유권 설정

```
# chown Prometheus:Prometheus /etc/prometheus
```

```
# chown prometheus:prometheus /var/lib/prometheus
```

### 8. 이전에 압축해제한 폴더에서 바이너리 파일을 전역폴더로 복사

```
# cp prometheus-2.11.0.linux-amd64/prometheus /usr/local/bin/
```

```
# cp prometheus-2.11.0.linux-amd64/promtool /usr/local/bin/
```





# 4. 설치 및 실행



## 4.1 Prometheus 서버 설치 및 실행(CentOS - systemd)(3/5)

9. 복사한 파일에 대해서 소유권 설정

```
#chown prometheus:prometheus /usr/local/bin/prometheus
```

```
#chown Prometheus:Prometheus /usr/local/bin/promtool
```

10. 이전 소스폴더에서 consoles, console\_libraries 디렉토리를 전체 복사한다.

```
# cp -r prometheus-2.11.0.linux-amd64/consoles /etc/prometheus/
```

```
# cp -r prometheus-2.11.0.linux-amd64/console_libraries /etc/prometheus/
```

11. prometheus 설정 파일 생성

```
# vim /etc/Prometheus/Prometheus.yml
```

# 파일 구성

```
[root@prometheus ~]# cat /etc/prometheus/prometheus.yml
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: 'prometheus_master'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
```

# 4. 설치 및 실행



## 4.1 Prometheus 서버 설치 및 실행(CentOS - systemd)(4/5)

### 12. 설정파일 소유권 설정

```
# chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

### 13. Systemd 설정 파일 구성

```
# vim /etc/systemd/system/prometheus.service
```

```
[root@prometheus ~]# cat /etc/systemd/system/prometheus.service
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \#
--config.file /etc/prometheus/prometheus.yml \#
--storage.tsdb.path /var/lib/prometheus/ \#
--web.console.templates=/etc/prometheus/consoles \#
--web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
```



# 4. 설치 및 실행



## 4.1 Prometheus 서버 설치 및 실행(CentOS - systemd)(5/5)

### 14. Systemd 구성 및 시작

```
# systemctl daemon-reload  
# systemctl start prometheus
```

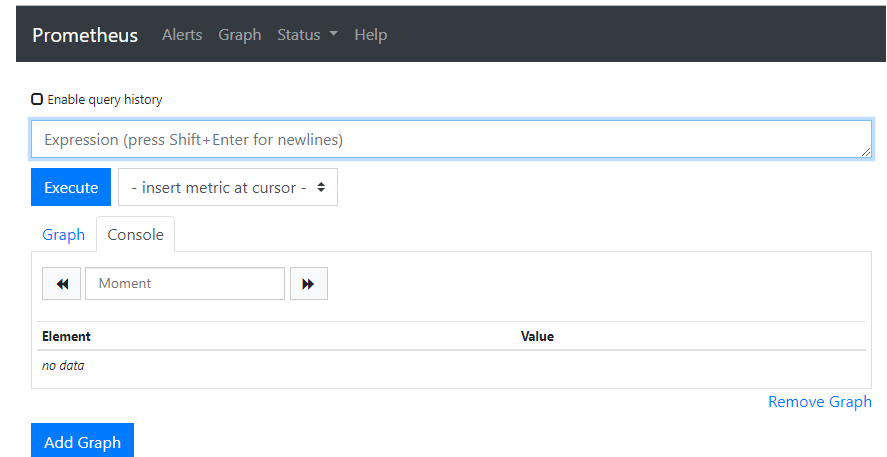
### 15. Firewall 포트 설정

```
# firewall-cmd --zone=public --add-port=9090/tcp --permanent  
# systemctl reload firewalld
```

### 16. 브라우저에서 접속 테스트

```
# http://<IP주소>:9090
```

오른쪽 이미지처럼 나온다면 설치완료



# 4. 설치 및 실행



## 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)(5/5)

- OS Version : CentOS Linux release 7.6.1810 (Core)
- Node-exporter version
- Server의 상태를 모니터링 하기 위해 설치
- Prometheus 공식 홈페이지 에서 다운

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

<2019년 7월 10일 기준으로 최신버전이 2.11>

### 1. 소스 파일 다운로드

```
# wget https://github.com/prometheus/node\_exporter/releases/download/v0.18.1/node\_exporter-0.18.1.linux-amd64.tar.gz
```

### 2. 소스 압축 해제

```
# tar zxvf node_exporter-0.18.1.linux-amd64.tar.gz
```

### 3. 버전 확인

```
# ./node_exporter --version
```

```
[root@prometheus node_exporter-0.18.1.linux-amd64]# ./node_exporter --version
node_exporter, version 0.18.1 (branch: HEAD, revision: 3db77732e925c08f675d7404a8c46466b2ece83e)
 build user:   root@550852a1acba
 build date:   20190604-16:41:18
 go version:   go1.12.5
```



# 4. 설치 및 실행



## 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)(5/5)

4. Node-exporter를 위한유저 생성

```
# useradd -rs /bin/false nodeusr
```

5. 이전에 압축해제한 폴더에서 바이너리 파일을 전역폴더로 복사

```
# cp node_exporter-0.18.1.linux-amd64/node_exporter /usr/local/bin/
```

6. Systemd 설정 파일 구성

```
# vim /etc/systemd/system/node_exporter.service
```

```
[root@prometheus bin]# cat /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=nodeusr
Group=nodeusr
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

# 4. 설치 및 실행



## 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)(5/5)

### 7. Systemd 구성 및 시작

```
# systemctl daemon-reload  
# systemctl start node_exporter  
# systemctl status node_exporter  
# systemctl enable node_exporter
```

### 8. Firewall 포트 설정

```
# firewall-cmd --zone=public --add-port=9100/tcp --permanent  
# systemctl reload firewalld
```

# 4. 설치 및 실행



## 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)(5/5)

### 9. Web 으로 접속하여 실행상태 확인

```
192.168.94.100:9100/metrics x 설정 x +  
← → ↻ 🏠 ⓘ 192.168.94.100:9100/metrics 📄 ⋮ 🛡️ ☆ 🗑️ 📄 🔄 🌐 ☰  
# HELP go_gc_duration_seconds A summary of the GC invocation durations.  
# TYPE go_gc_duration_seconds summary  
go_gc_duration_seconds{quantile="0"} 0  
go_gc_duration_seconds{quantile="0.25"} 0  
go_gc_duration_seconds{quantile="0.5"} 0  
go_gc_duration_seconds{quantile="0.75"} 0  
go_gc_duration_seconds{quantile="1"} 0  
go_gc_duration_seconds_sum 0  
go_gc_duration_seconds_count 0  
# HELP go_goroutines Number of goroutines that currently exist.  
# TYPE go_goroutines gauge  
go_goroutines 7  
# HELP go_info Information about the Go environment.  
# TYPE go_info gauge  
go_info{version="go1.12.5"} 1  
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.  
# TYPE go_memstats_alloc_bytes gauge  
go_memstats_alloc_bytes 2.721968e+06  
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.  
# TYPE go_memstats_alloc_bytes_total counter  
go_memstats_alloc_bytes_total 2.721968e+06  
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.  
# TYPE go_memstats_buck_hash_sys_bytes gauge  
go_memstats_buck_hash_sys_bytes 1.444017e+06
```



# 4. 설치 및 실행



## 4.1 Prometheus 서버 설치 및 실행(CentOS - systemd)(1/5)

- OS Version : CentOS Linux release 7.6.1810 (Core)
- Alertmanager Version : alertmanager-0.18.0.linux-amd64
- Prometheus 공식 홈페이지 에서 다운

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

<2019년 7월 10일 기준으로 최신버전이 0.18>

### 1. 소스 파일 다운로드

```
# wget https://github.com/prometheus/alertmanager/releases/download/v0.18.0/alertmanager-0.18.0.linux-amd64.tar.gz
```

### 2. 소스 파일 압축 해제

```
# tar zxvf alertmanager-0.18.0.linux-amd64.tar.gz
```

### 3. 해당 폴더로 이동

```
# cd alertmanager-0.18.0.linux-amd64
```

### 4. 버전 확인

```
# ./alertmanager --version
```

```
[root@prometheus-alertmanager-0.18.0.linux-amd64]# ./alertmanager --version
alertmanager, version 0.18.0 (branch: HEAD, revision: 1ace0f76b7101cccc149d7298022df36039858ca)
build user:      root@888885ed3ed0
build date:      20190708-14:31:49
go version:      go1.12.6
```





# 4. 설치 및 실행



## 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)(5/5)

### 5. 유저 생성

```
# useradd --no-create-home --shell /bin/false alertmanager
```

### 6. 필요한 디렉토리 생성

```
# sudo mkdir /etc/alertmanager  
# sudo mkdir /etc/alertmanager/template  
# sudo mkdir -p /var/lib/alertmanager/data
```

### 7. 생성한 디렉토리에 소유권 설정

```
# sudo chown -R alertmanager:alertmanager /etc/alertmanager  
# sudo chown -R alertmanager:alertmanager /var/lib/alertmanager
```

### 8. 실행 파일 복사

```
# cp alertmanager-0.18.0.linux-amd64/alertmanager /usr/local/bin/  
# cp alertmanager-0.18.0.linux-amd64/amttool /usr/local/bin/
```

# 4. 설치 및 실행



## 4.2 Node-exporter 설치 및 실행 (CentOS – Systemd기반)(5/5)

### 9. Systemd 설정 파일 구성

```
# vim /etc/systemd/system/alertmanager.service
```

```
[root@prometheus ~]# cat /etc/systemd/system/alertmanager.service
[Unit]
Description=Prometheus Alertmanager Service
Wants=network-online.target
After=network.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
ExecStart=/usr/local/bin/alertmanager \#
    --config.file /etc/alertmanager/alertmanager.yml \#
    --storage.path /var/lib/alertmanager/data
Restart=always

[Install]
WantedBy=multi-user.target
```

### 10. Systemd 구성

```
# systemctl daemon-reload
# systemctl enable alertmanager
```



# 5. 기능소개



## 세부 목차

5.1 Prometheus Dashboard에서 PromQL으로 검색

5.2 모니터링 대상 검색

5.3 그래프 확인

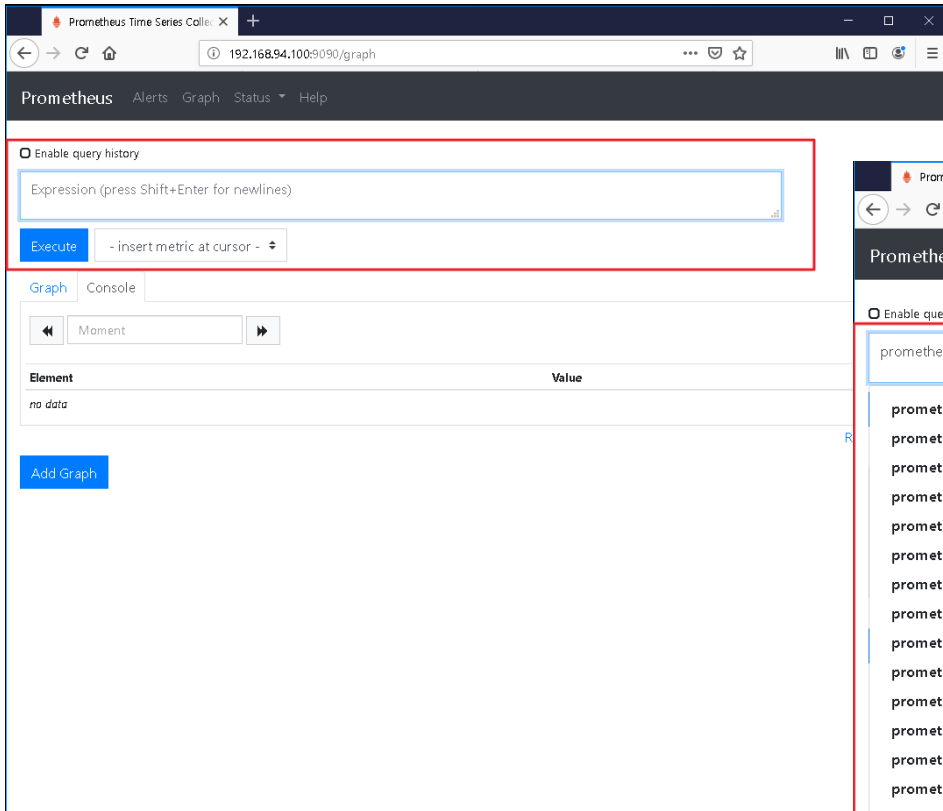


# 5. 기능소개

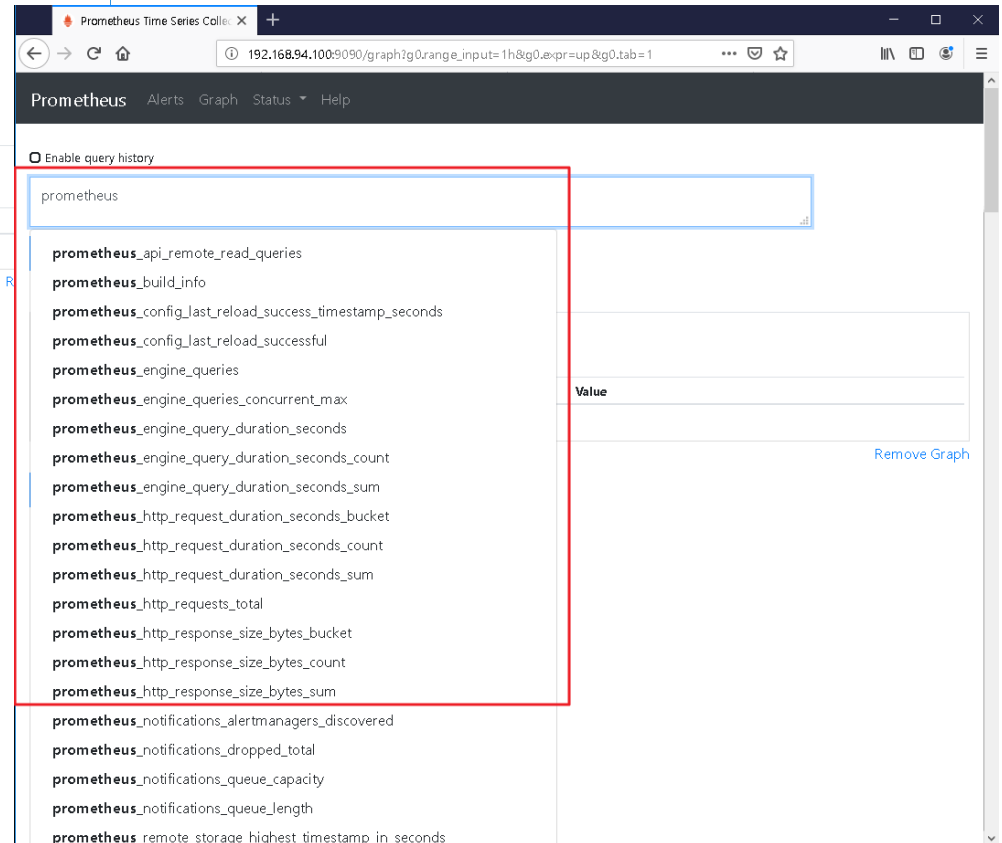


## 5.1 Prometheus Dashboard에서 PromQL으로 검색

- Prometheus의 검색창에서 다양한 값으로 검색할 수 있음



The screenshot shows the Prometheus dashboard interface. The search bar is highlighted with a red box, containing the text "Expression (press Shift+Enter for newlines)". Below the search bar is an "Execute" button and a dropdown menu with the text "- insert metric at cursor -". The dashboard also shows a "Graph" tab and a "Console" tab, with a "Moment" button and a "Value" column header.



The screenshot shows the Prometheus dashboard interface with a search query "prometheus" entered in the search bar. A list of search results is displayed below the search bar, enclosed in a red box. The results include various Prometheus metrics such as `prometheus_api_remote_read_queries`, `prometheus_build_info`, `prometheus_config_last_reload_success_timestamp_seconds`, `prometheus_engine_queries`, `prometheus_engine_queries_concurrent_max`, `prometheus_engine_query_duration_seconds`, `prometheus_engine_query_duration_seconds_count`, `prometheus_engine_query_duration_seconds_sum`, `prometheus_http_request_duration_seconds_bucket`, `prometheus_http_request_duration_seconds_count`, `prometheus_http_request_duration_seconds_sum`, `prometheus_http_requests_total`, `prometheus_http_response_size_bytes_bucket`, `prometheus_http_response_size_bytes_count`, `prometheus_http_response_size_bytes_sum`, `prometheus_notifications_alertmanagers_discovered`, `prometheus_notifications_dropped_total`, `prometheus_notifications_queue_capacity`, `prometheus_notifications_queue_length`, and `prometheus_remote_storage_highest_timestamp_in_seconds`. The search bar and the list of results are highlighted with a red box.

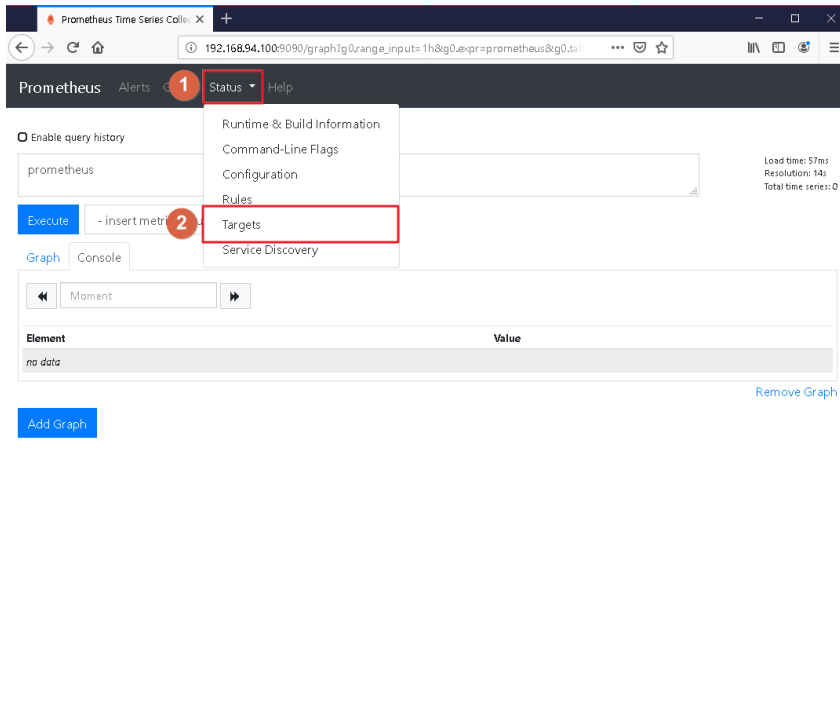


# 5. 기능소개

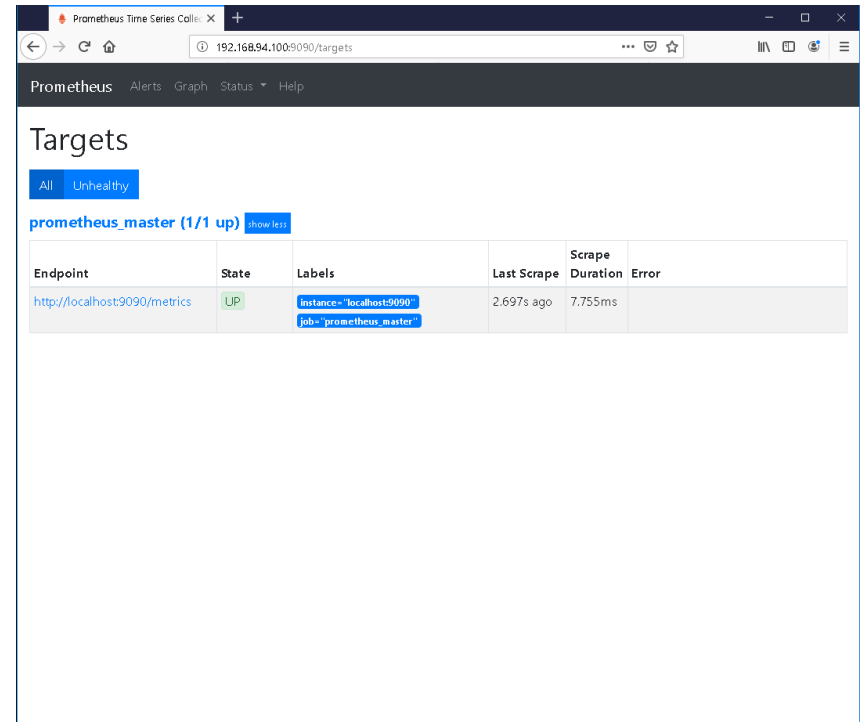


## 5.2 모니터링 대상 확인

- Prometheus의 Dashboard에서 모니터링 되고 있는 대상이 확인이 가능함



The screenshot shows the Prometheus Dashboard interface. The 'Status' menu is open, and the 'Targets' option is highlighted with a red box. A red circle with the number '1' is next to the 'Status' menu, and another red circle with the number '2' is next to the 'Targets' option. The dashboard also shows a search bar with 'prometheus' entered, an 'Execute' button, and a 'Graph' section with a 'Moment' time range selector.



The screenshot shows the 'Targets' page in the Prometheus Dashboard. The page title is 'Targets'. There are two tabs: 'All' (selected) and 'Unhealthy'. Below the tabs, there is a section for 'prometheus\_master (1/1 up)' with a 'show list' link. A table displays the following data:

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus_master"	2.697s ago	7.755ms	

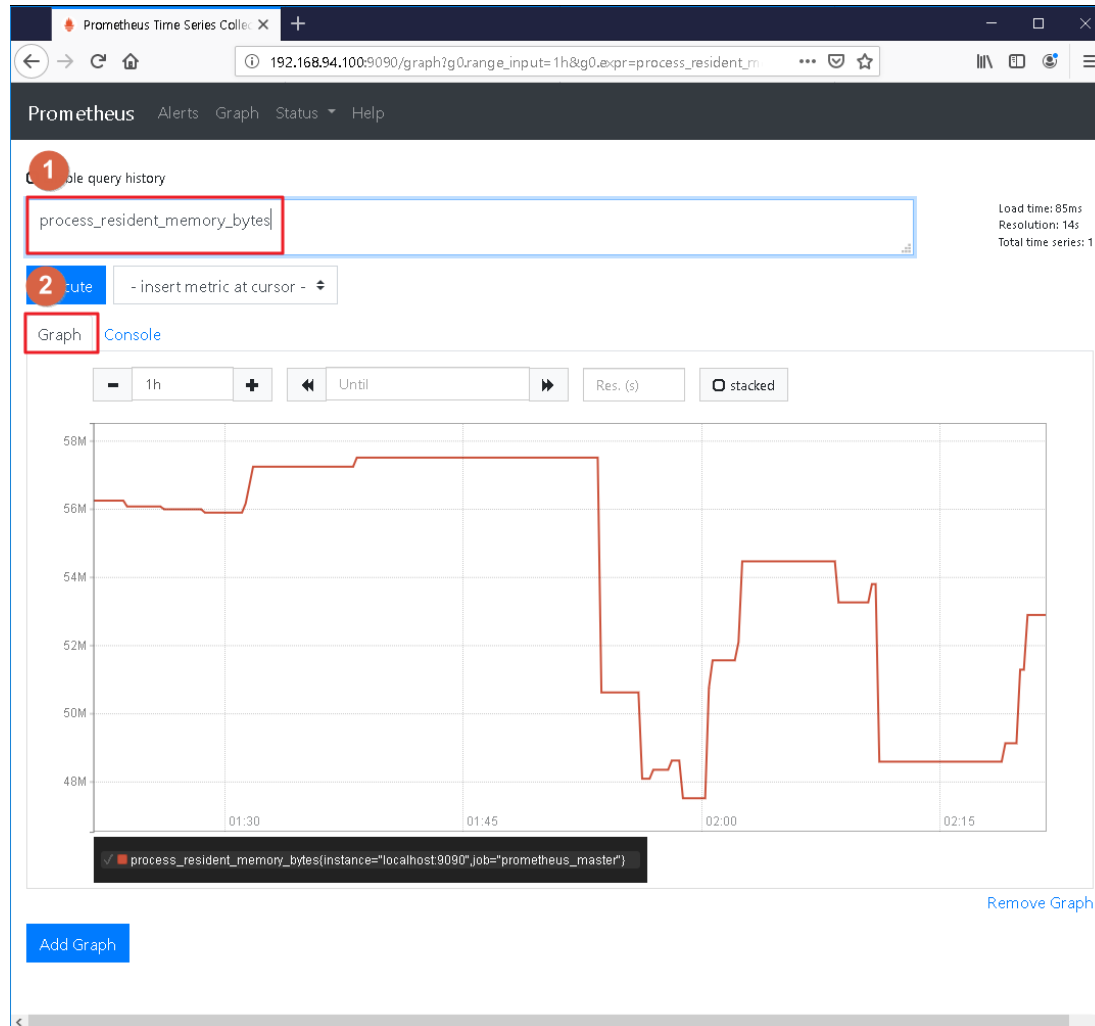


# 5. 기능소개



## 5.2 모니터링 대상 확인

- Prometheus의 Dashboard에서 모니터링 되고 있는 그래프 확인 가능



# 6. 활용예제



## 세부 목차

- 6.1 모니터링 대상 노드 추가
- 6.2 Alertmanager를 통해 이메일로 통보
- 6.3 Grafana와 연동하여 시각화 설정



# 6. 활용예제



## 6.1 모니터링 대상 노드 추가

- Prometheus에 노드를 추가하여 모니터링

1. Prometheus 설정파일 맨 아래라인에 빨간 내용을 추가

```
# vim /etc/prometheus/prometheus.yml
```

```
scrape_configs:
```

```
- job_name: 'prometheus_master'
```

```
  scrape_interval: 5s
```

```
  static_configs:
```

```
    - targets: ['localhost:9090']
```

```
- job_name: node
```

```
  static_configs:
```

```
    - targets: ['localhost:9100']
```



# 6. 활용예제

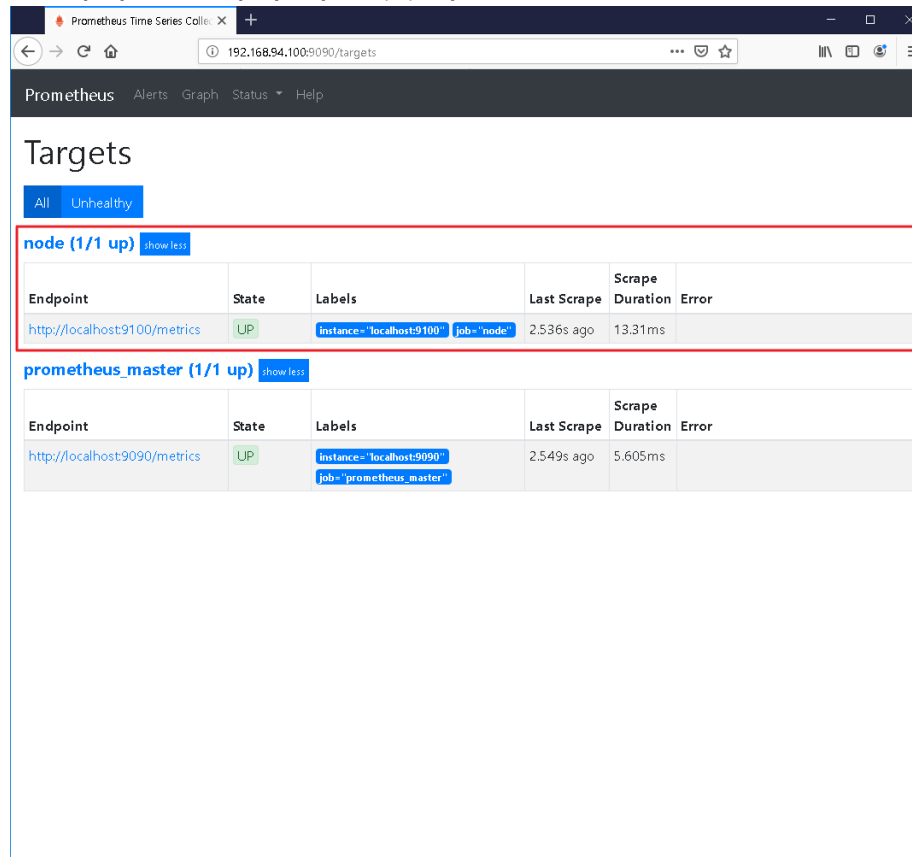


## 6.1 모니터링 대상 노드 추가

### 2. Prometheus 서버 재시작

```
# systemctl restart Prometheus
```

### 3. Prometheus Dashboard 에서 노드가 추가된 것 확인



Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<b>node (1/1 up)</b> <a href="#">show less</a>					
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node"	2,536s ago	13.31 ms	
<b>prometheus_master (1/1 up)</b> <a href="#">show less</a>					
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus_master"	2,549s ago	5.605 ms	



# 6. 활용예제

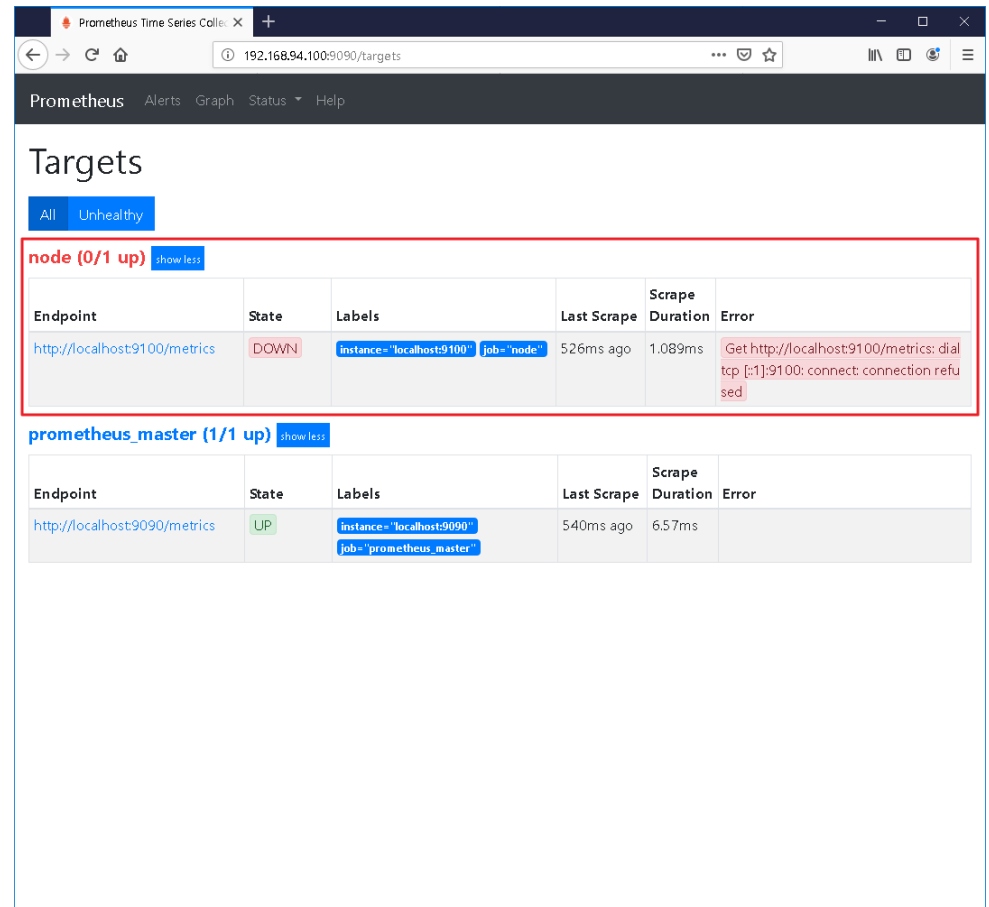


## 6.2 Alertmanager를 통해 이메일로 통보

- Alertmanager를 통해 이메일로 통보를 받는 설정

### 1. 먼저 알람 상태를 만들기 위해 node\_exporter 정지하여 down 상태 설정

```
# systemctl stop node_exporter
```



The screenshot shows the Prometheus web interface. The 'Targets' section is active, displaying two target groups. The 'node' group (0/1 up) has one target in a 'DOWN' state. The 'prometheus\_master' group (1/1 up) has one target in an 'UP' state.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost9100/metrics	DOWN	instance="localhost9100" job="node"	526ms ago	1.089ms	Get http://localhost9100/metrics: dial tcp [:1]:9100: connect: connection refused
http://localhost9090/metrics	UP	instance="localhost9090" job="prometheus_master"	540ms ago	6.57ms	



# 6. 활용예제



## 6.2 Alertmanager를 통해 이메일로 통보

2. Prometheus 설정파일에 아래 빨간 부분을 추가

```
# vim /etc/prometheus/prometheus.yml
```

global:

```
scrape_interval: 10s
```

rule\_files:

```
- rules.yml
```

alerting:

alertmanagers:

```
- static_configs:
```

```
- targets: ['localhost:9093']
```

scrape\_configs:

```
- job_name: 'prometheus_master'
```

```
scrape_interval: 5s
```

```
static_configs:
```

```
- targets: ['localhost:9090']
```

```
- job_name: node
```

```
static_configs:
```

```
- targets: ['localhost:9100']
```

```
[root@prometheus ~]# cat /etc/prometheus/prometheus.yml
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: 'prometheus_master'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
  - job_name: node
    static_configs:
      - targets: ['localhost:9100']
```



# 6. 활용예제



## 6.2 Alertmanager를 통해 이메일로 통보

### 3. rules.yml 파일 설정

```
# vim /etc/prometheus/rules.yml
```

groups:

- name: example

rules:

- alert: InstanceDown

```
  expr: up == 0
```

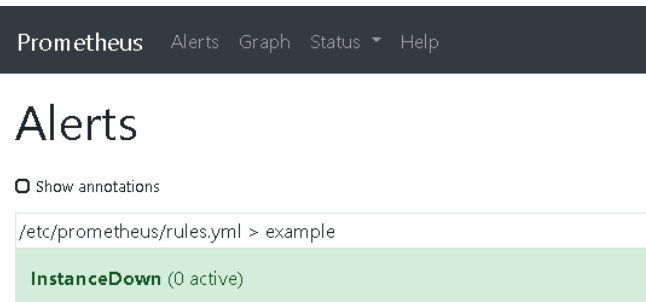
```
  for: 1m
```

```
root@prometheus ~]# cat /etc/prometheus/rules.yml
groups:
- name: example
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
```

### 5. Prometheus 서버 재시작

```
# systemctl restart Prometheus
```

### 6. Prometheus 에서 alert이 생성된것을 확인



# 6. 활용예제



## 6.2 Alertmanager를 통해 이메일로 통보

7. Alertmanager 설정파일을 다음과 같이 구성

```
# vim /etc/alertmanager/alertmanager.yml
```

Route:

```
group_by: [Alertname]
# Send all notifications to me.
receiver: email-notifications
```

receivers:

```
- name: email-notifications
email_configs:
- to: example@example.com
  from: example@example.com
  smarthost: smtp.address:<port number>
  auth_username: "<example@example.com>"
  auth_identity: "<example@example.com>"
  auth_password: "<user-password>"
```

### NOTE

Google과 같은 계정을 통해 이메일 통보를 설정할 수 있음

route:

```
group_by: [Alertname]
# Send all notifications to me.
receiver: email-notifications
```

receivers:

```
- name: email-notifications
email_configs:
- to: <username>@gmail.com
  from: <username>@gmail.com
  smarthost: smtp.gmail.com:587
  auth_username: "<username>@gmail.com"
  auth_identity: "<username>@gmail.com"
  auth_password: "<user-password>"
```



# 6. 활용예제



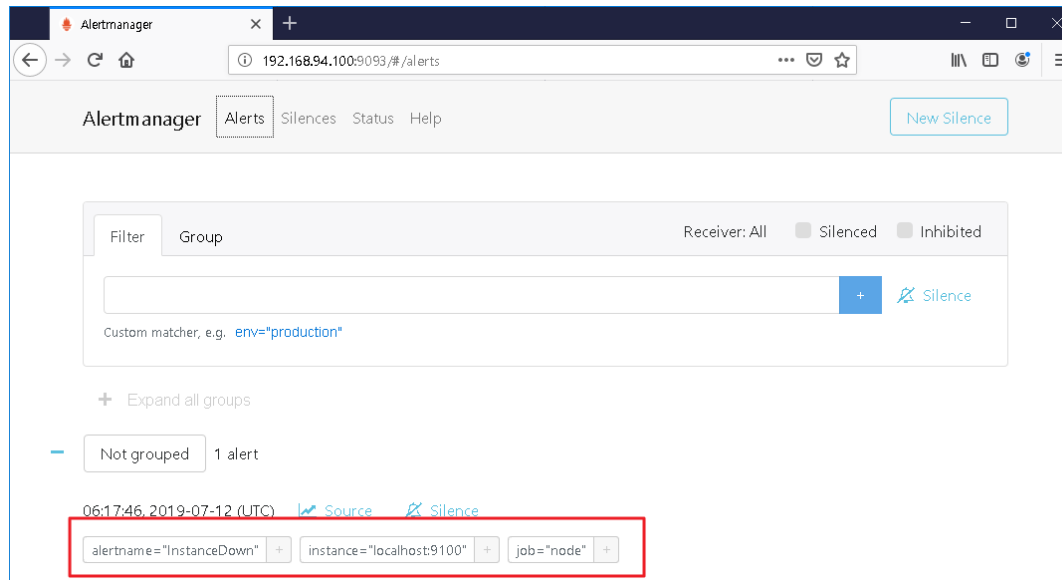
## 6.2 Alertmanager를 통해 이메일로 통보

### 8. Alertmanager 를 재시작

# systemctl restart alertmanager

### 9. Prometheus Alertmanager 에서 확인

# 9093 포트로 접속 다음과 같이 Alert 이 등록된것을 확인

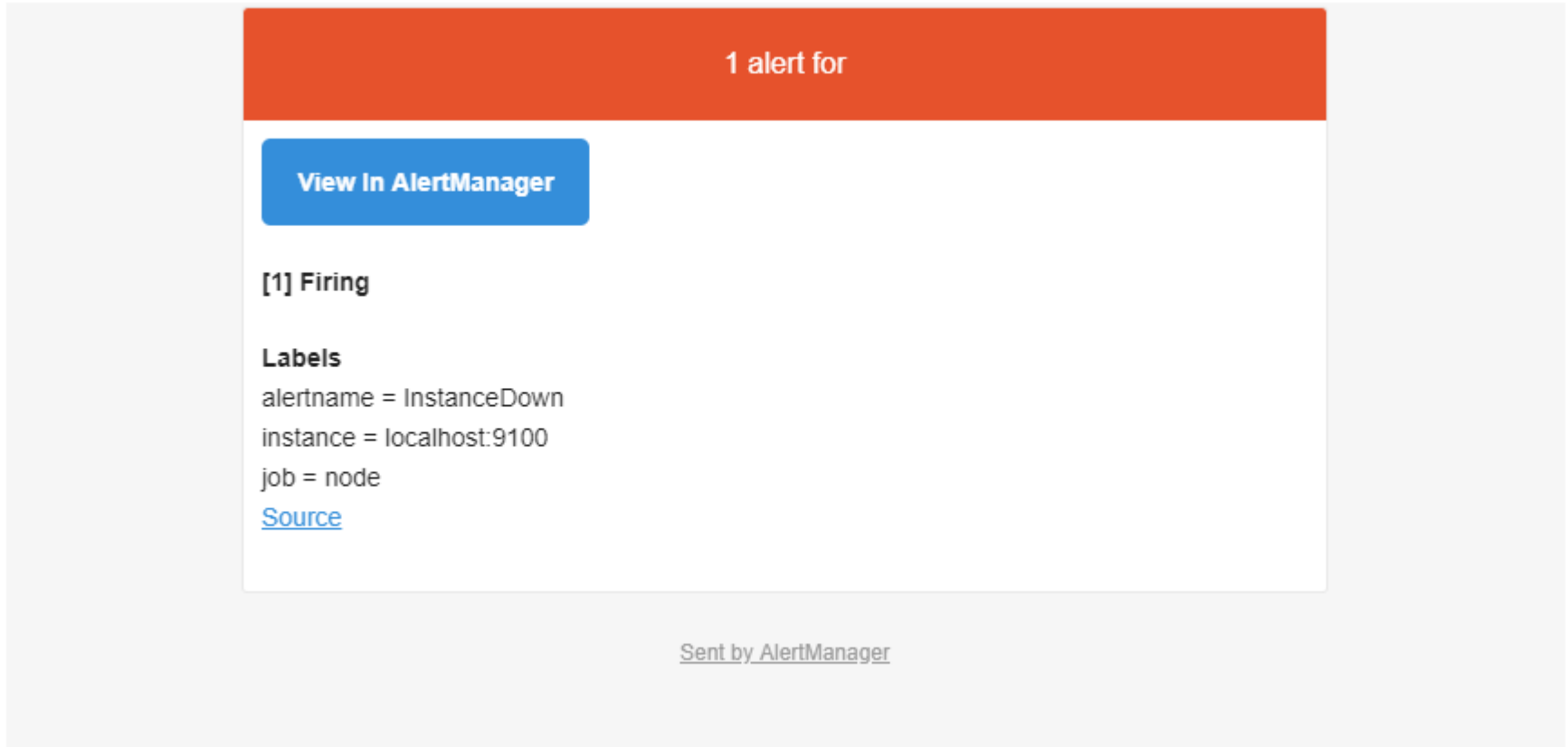


# 6. 활용예제



## 6.2 Alertmanager를 통해 이메일로 통보

### 10. Alertmanager에 의해서 보내진 이메일 확인



The screenshot shows an email notification interface. At the top, a red bar contains the text "1 alert for". Below this is a blue button labeled "View In AlertManager". Underneath the button, the text "[1] Firing" is displayed. A section titled "Labels" lists the following information: "alertname = InstanceDown", "instance = localhost:9100", and "job = node". A blue link labeled "Source" is positioned below the labels. At the bottom of the notification area, the text "Sent by AlertManager" is visible.



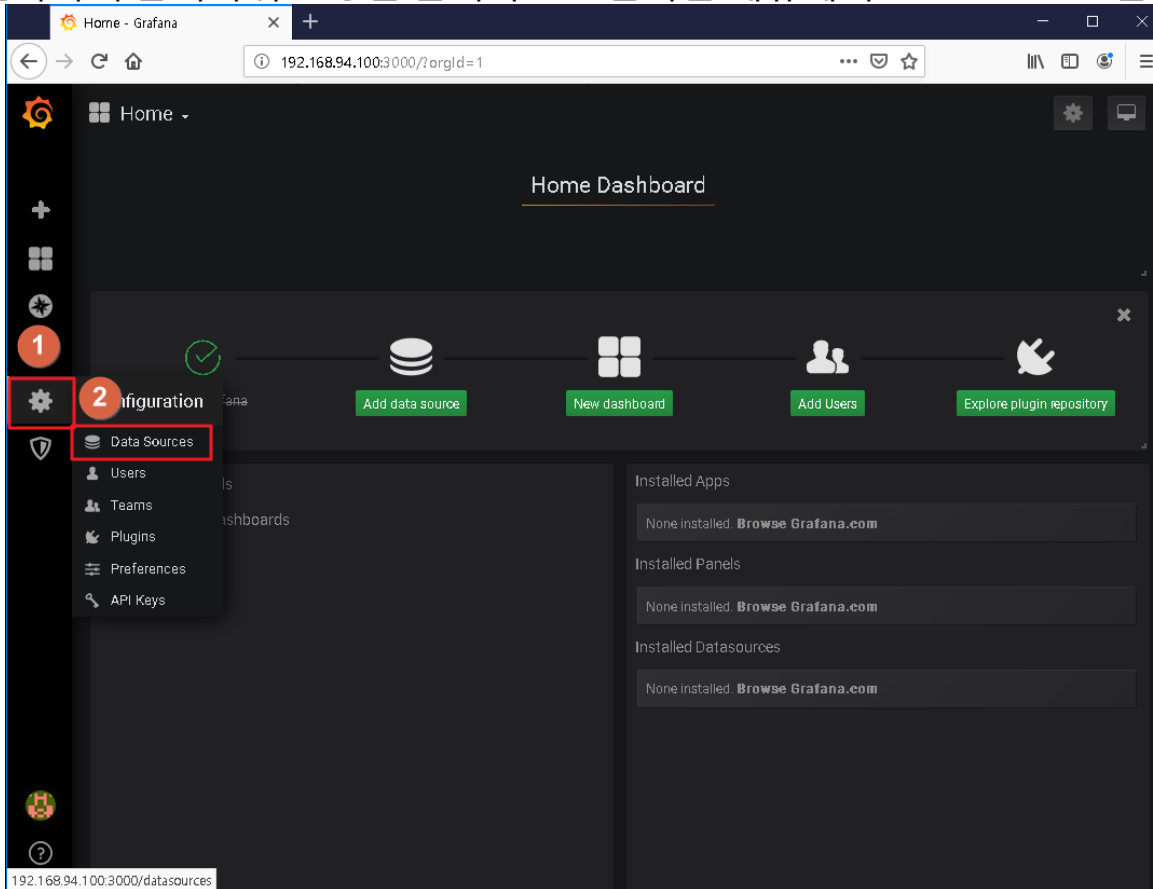
# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

- Grafana는 데이터를 전달받아 Metric을 쿼리하고 시각화하는 도구
- Grafana가 설치되어 있다고 가정하고 설정을 진행

1. Grafana에 접속하여 톱니바퀴 모양을 클릭하고 드롭다운 메뉴에서 “Data Sources”를 클릭.



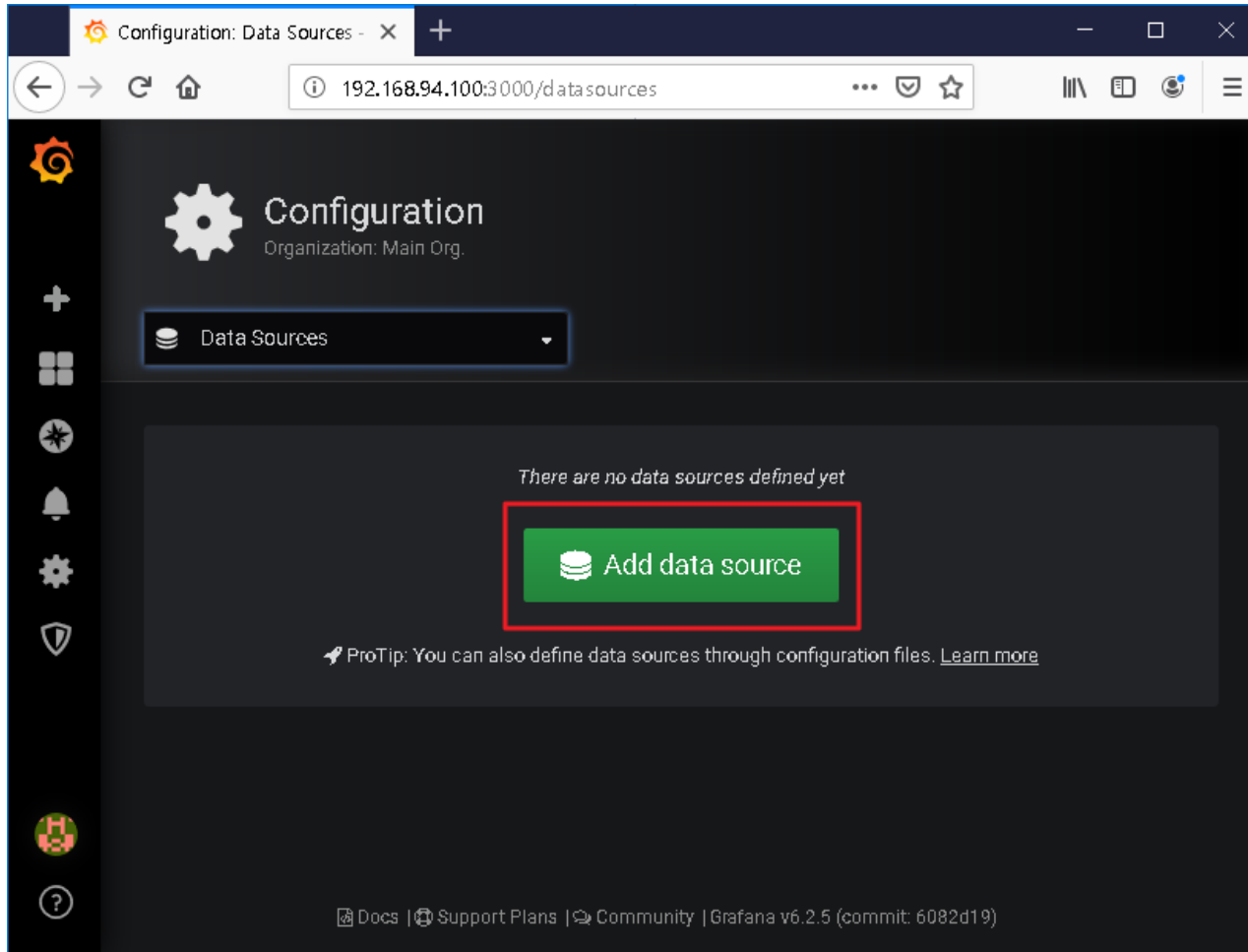


# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

### 2. "Add data source" 클릭

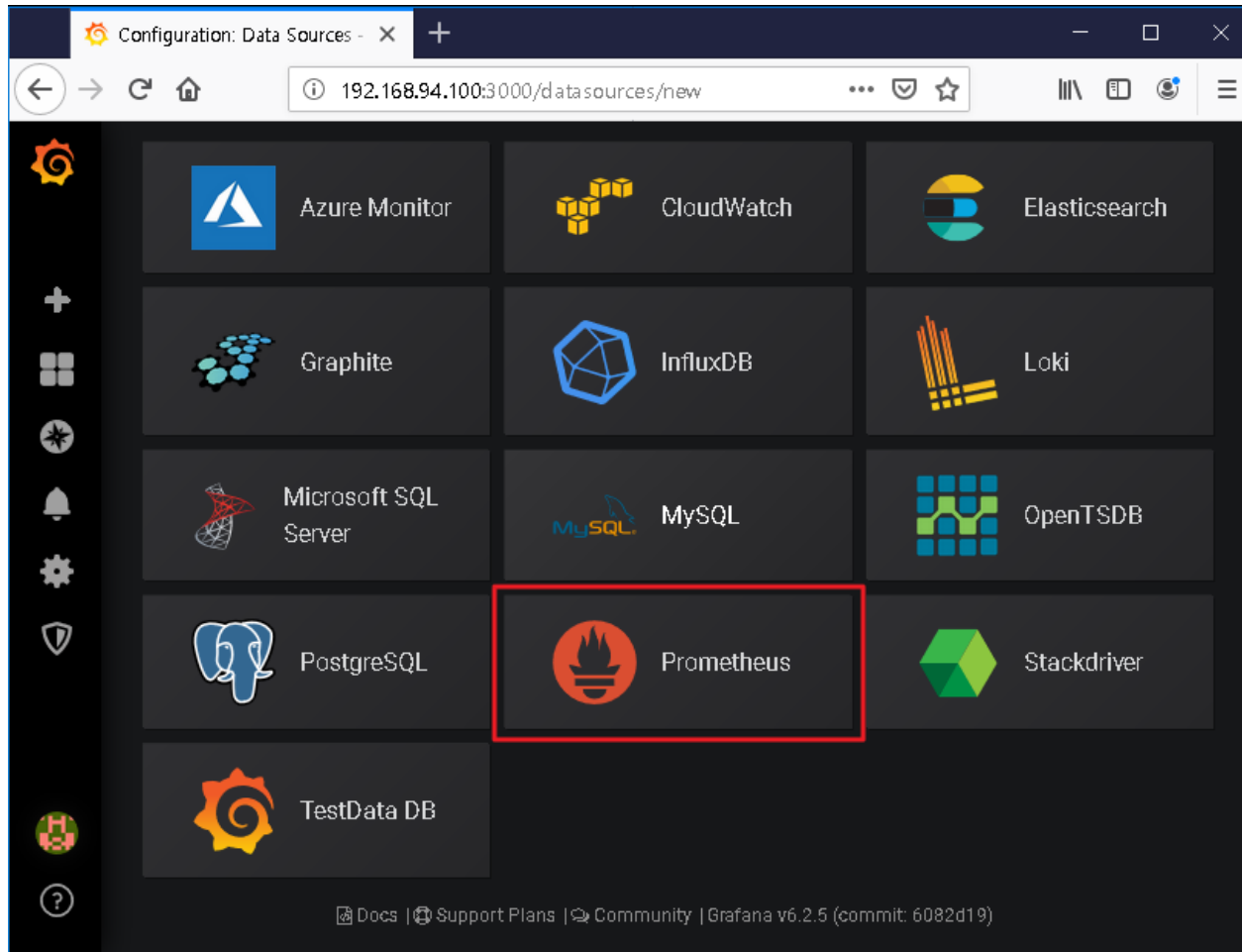


# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

3. Data Sources 리스트에서 “Prometheus” 를 클릭



# 6. 활용예제



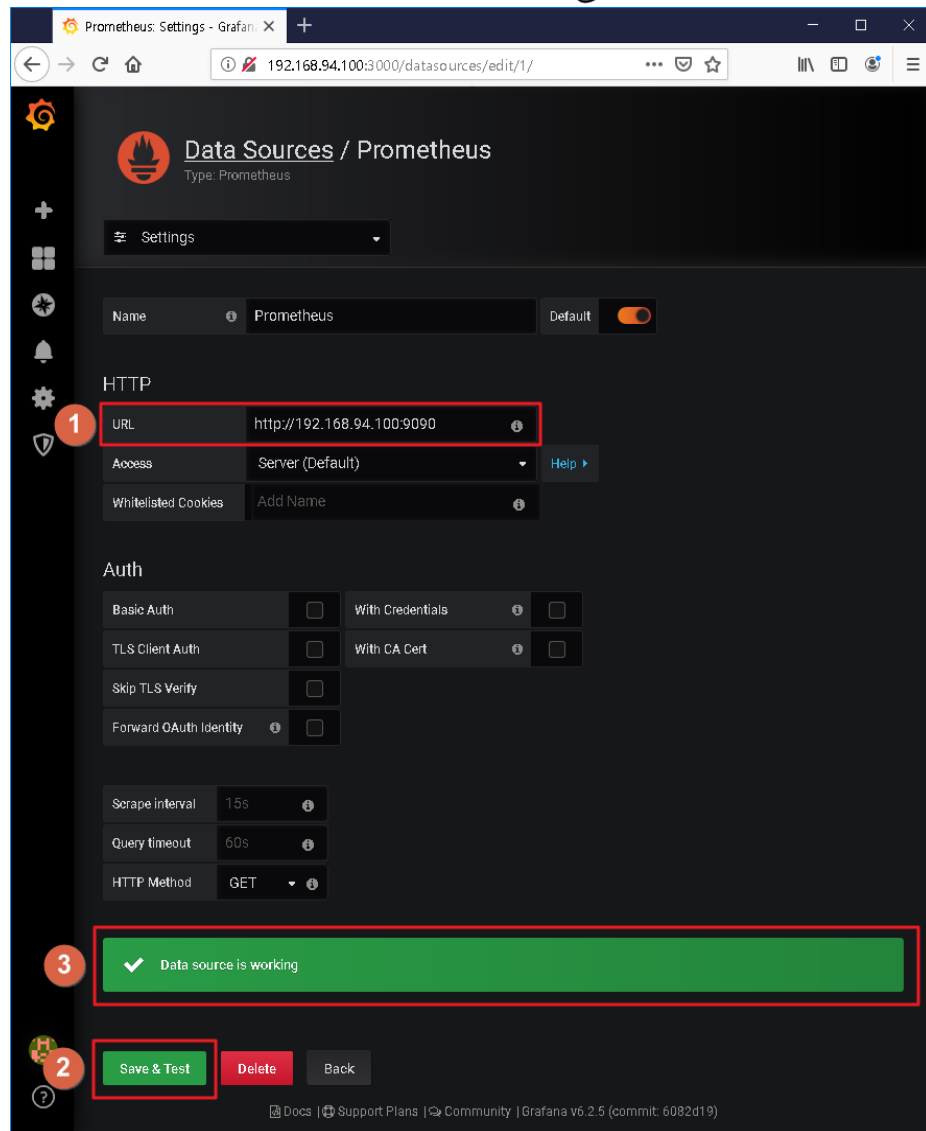
## 6.3 Grafana와 연동하여 시각화 설정

### 4. Prometheus 연동하기 위한 설정

4-1. “URL” 주소에 Prometheus 서버주소  
및 포트번호 입력

4-2. “Save & Test”를 클릭하여 저장

4-3. “Data source is working”표시가 나오며  
동작하는 것을 확인



The screenshot shows the Grafana interface for configuring a Prometheus data source. The browser address bar shows the URL `192.168.94.100:3000/datasources/edit/1/`. The page title is "Data Sources / Prometheus". The "Settings" dropdown is open. The "Name" field is "Prometheus" and the "Default" toggle is turned on. Under the "HTTP" section, the "URL" field is highlighted with a red box and a "1" in a circle, containing the value `http://192.168.94.100:9090`. Below it, the "Access" dropdown is set to "Server (Default)". Under the "Auth" section, there are several checkboxes for authentication methods, all of which are currently unchecked. At the bottom, the "Scrape interval" is set to "15s", "Query timeout" is "60s", and "HTTP Method" is "GET". A green success message "Data source is working" is displayed at the bottom, highlighted with a red box and a "3" in a circle. The "Save & Test" button is highlighted with a red box and a "2" in a circle.

# 6. 활용예제

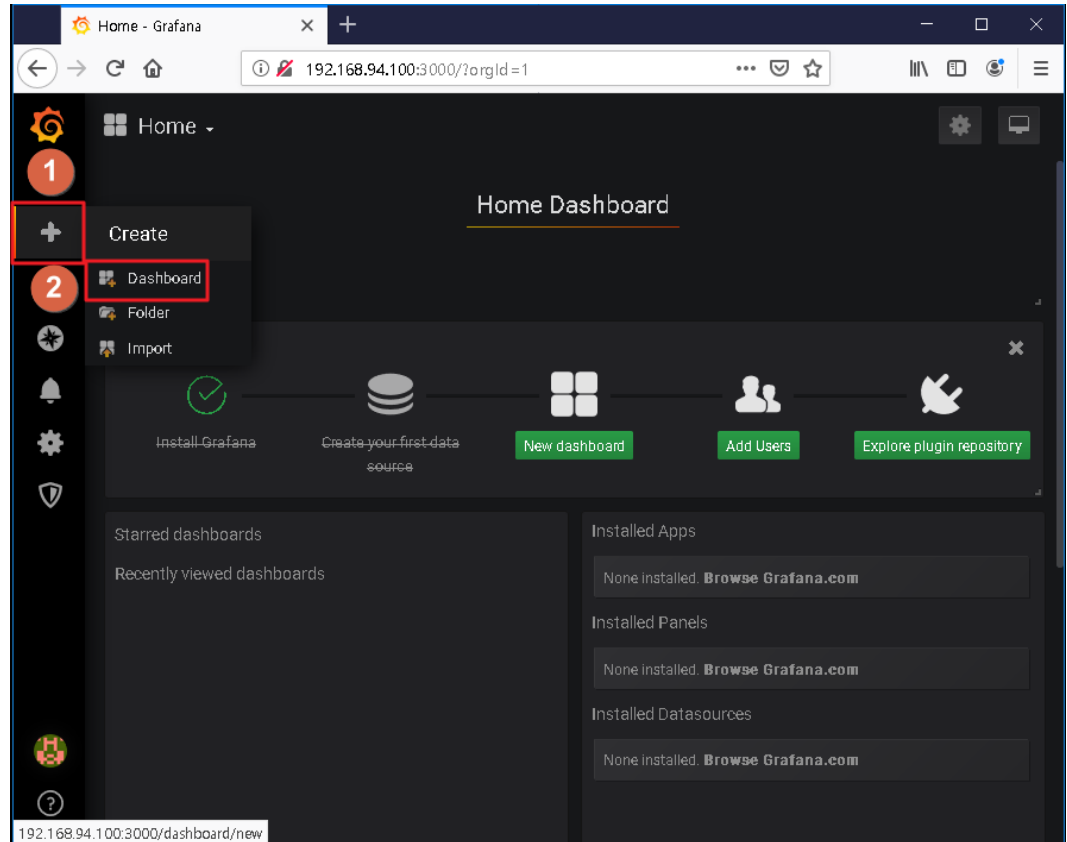


## 6.3 Grafana와 연동하여 시각화 설정

### 5. Dashboard 생성

#### 5-1. "+" 를 클릭

#### 5-2. "Dashboard"를 클릭

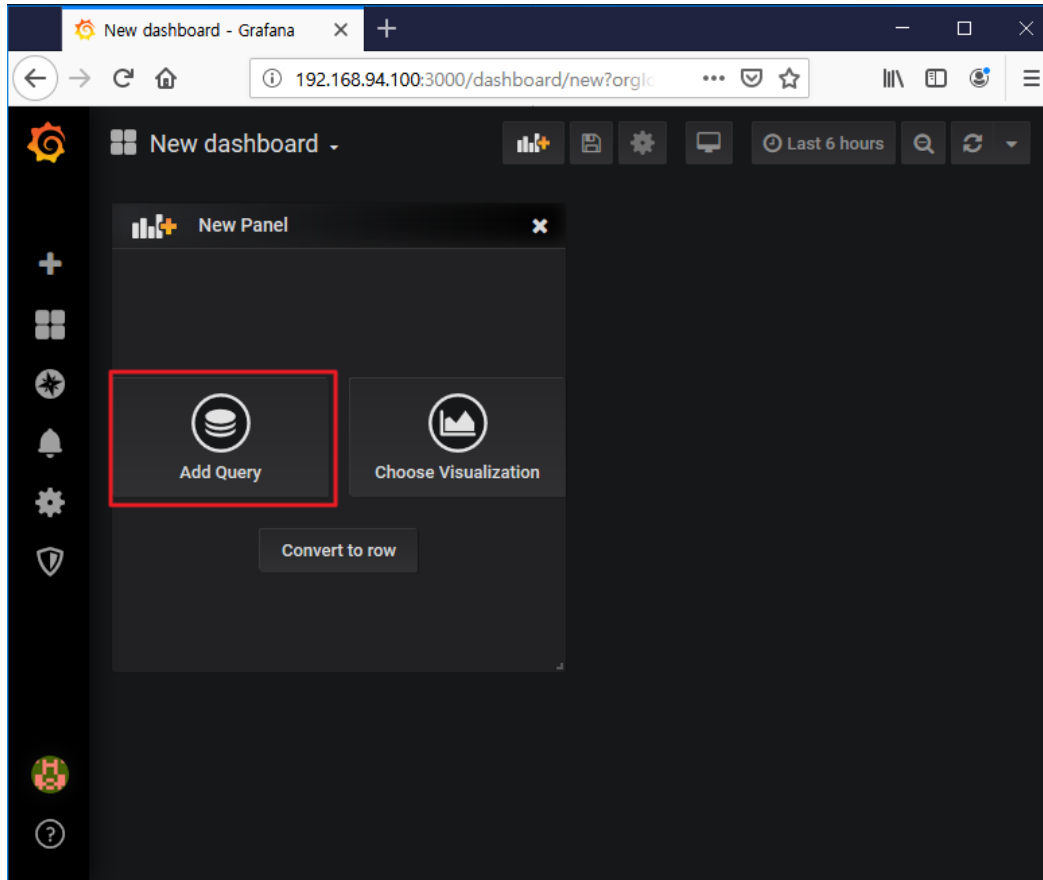


# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

### 6. "Add Query" 클릭

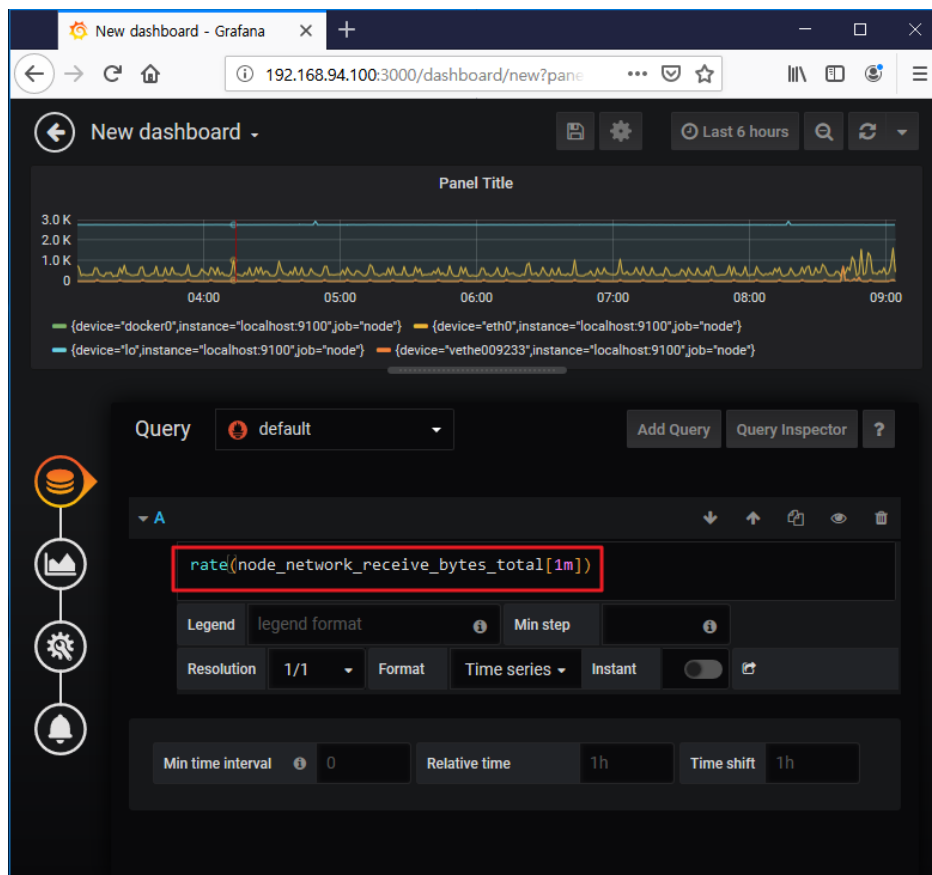


# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

7. 입력창에 “rate(node\_network\_receive\_bytes\_total[1m])”를 입력하여 그래프가 정상적으로 생성되는지 확인



# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

8. 톱니바퀴 모양을 클릭 후 대시보드의 이름을 설정

The screenshot shows the Grafana dashboard editor interface. At the top right, there is a settings gear icon highlighted with a red box. Below the dashboard, the query editor shows the following query: `rate(node_network_receive_bytes_total[1m])`. The legend below the query shows four data series with their respective labels and values.

The screenshot shows the Grafana dashboard settings page. The 'Name' field is set to 'Prometheus' and is highlighted with a red box. The 'Save' button is also highlighted with a red box. The settings page includes various options such as Annotations, Variables, Links, JSON Model, Time Options, and Panel Options.

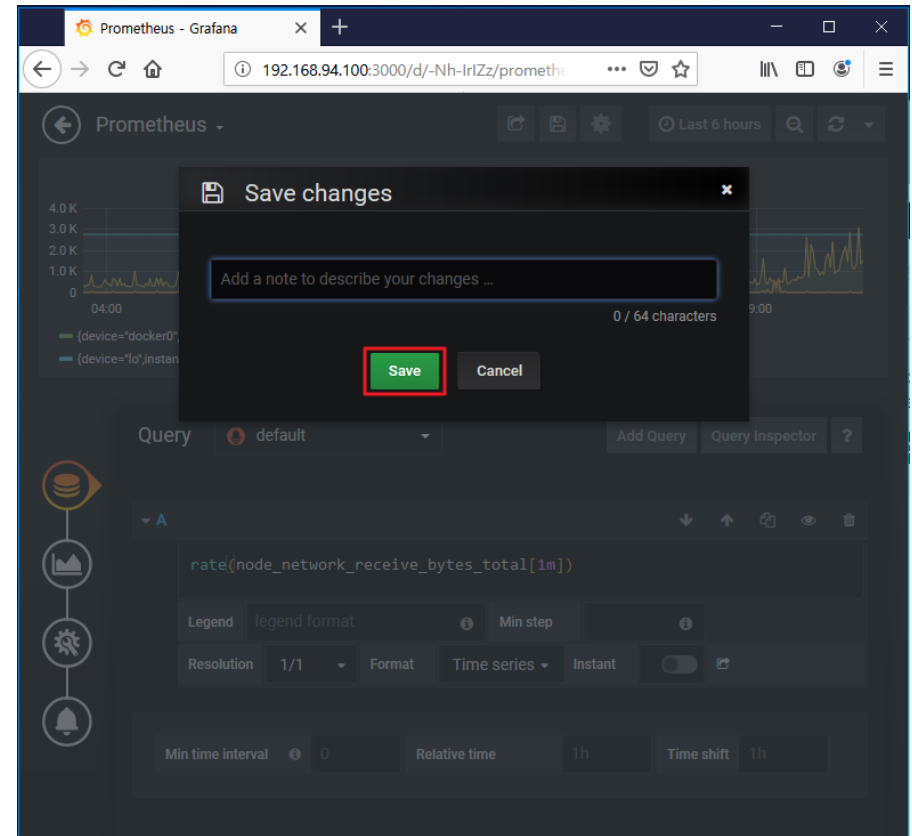
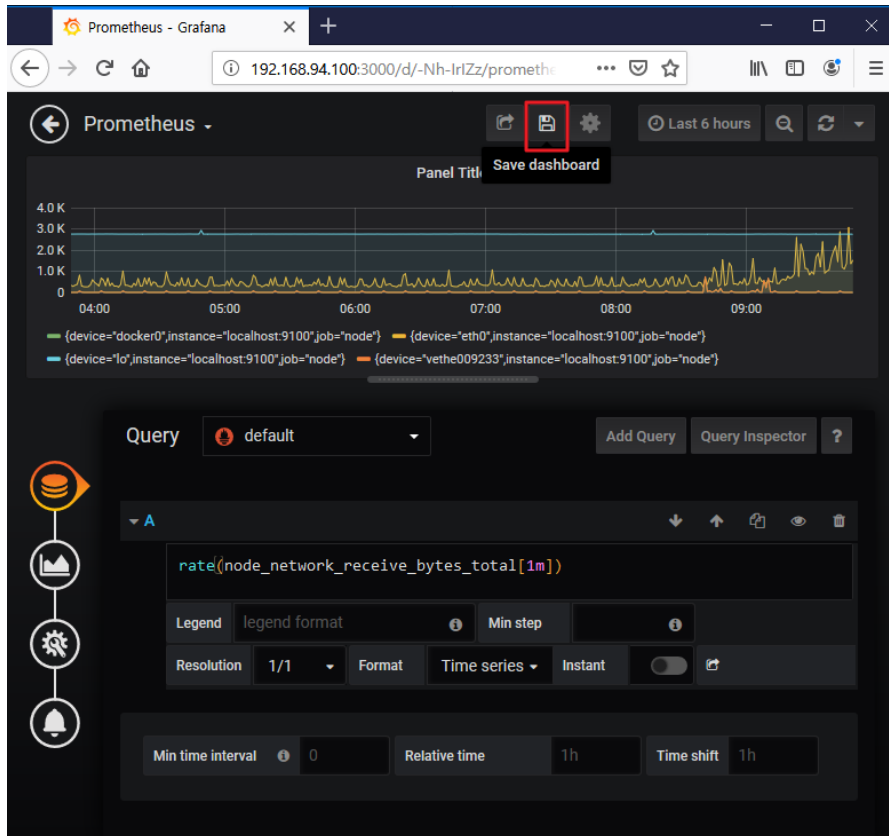


# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

9. Save dashboard를 클릭하여 Dashboard를 저장.



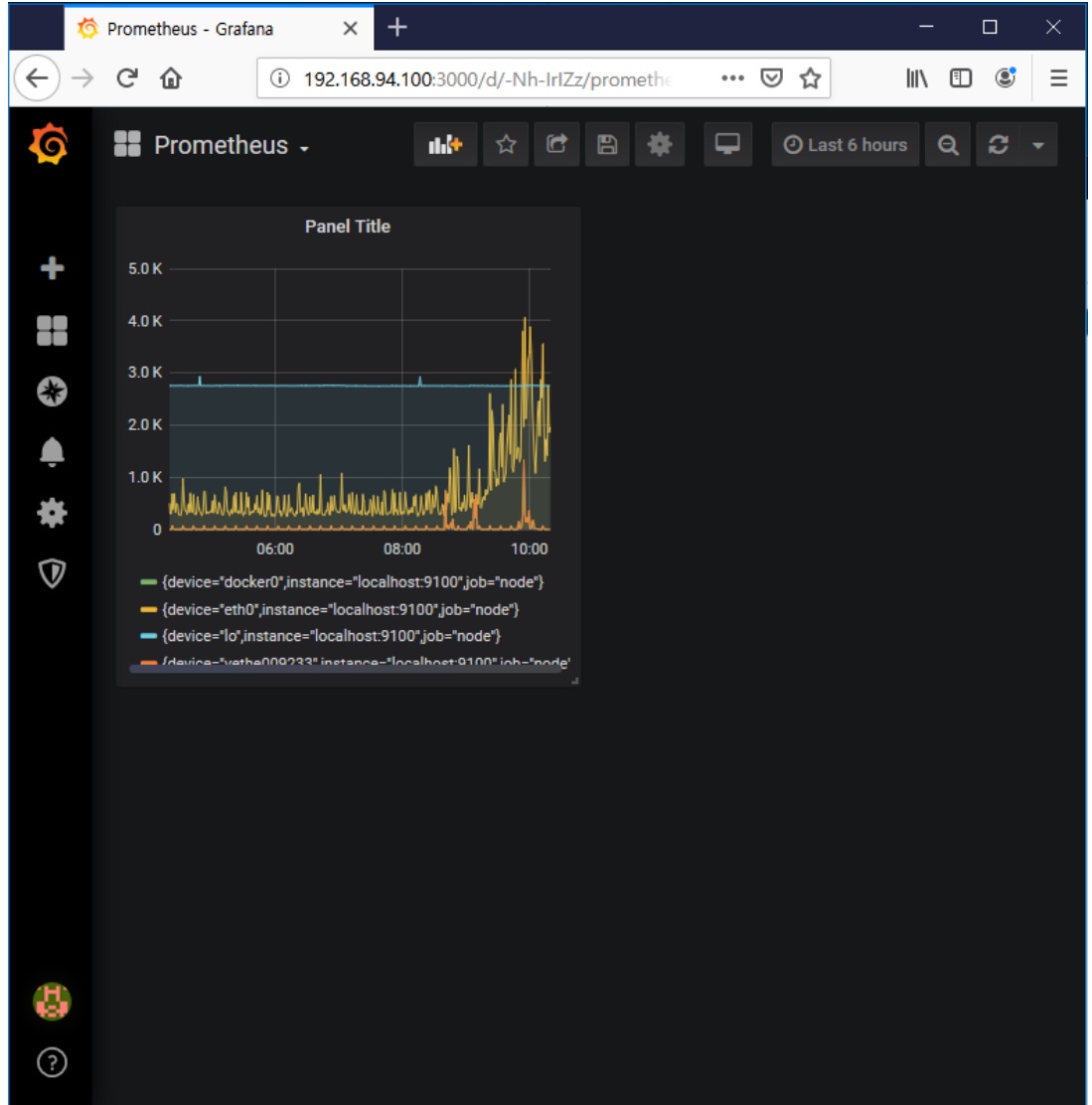


# 6. 활용예제



## 6.3 Grafana와 연동하여 시각화 설정

### 10. Prometheus와 Grafana 연동완료





**Q** Prometheus를 통해서 로그를 수집할 수 있나요?

&

**A** Prometheus는 이벤트 로깅 시스템이 아니라 측정 기준을 수집하고 처리하는 시스템입니다. 로그를 수집하려면 ELK같은 솔루션을 사용해야 합니다.

**Q** Prometheus는 HA구성이 가능한가요?

&

**A** 동일한 Prometheus 서버를 여러대 실행하여 HA구성이 가능합니다.





**Q** Alert를 보낼수 있나요?

&

**A** Alertmanager는 다음의 외부시스템을 Alert을 발송할수 있습니다.  
Email, Generic Webhooks, HipChat, OpsGenie, PagerDuty,  
Pushover, Slack

**Q** Prometheus는 Dashboard를 생성할수 있나요?

&

**A** Grafana를 통해 시각화를 할 수 있습니다.



# 8. 용어정리



용어	설명
Metric	시간상 특정 지점에서 시스템의 일부측면을 설명하는 숫자 값. Log와는 달리 주기적으로 보내게 됩니다. Log는 보통 무언가가 발생했을 때 로그 파일에 추가합니다. 정기적으로 수집해서 시스템의 추이를 살펴볼수 있는 정보.
PromQL	Prometheus는 사용자가 실시간으로 시계열 데이터를 선택하고 집계 할 수있는 PromQL (Prometheus Query Language)이라는 기능적 쿼리 언어를 제공합니다.
TSDB	Time Series DBMS는 time series data를 처리하기 위해 최적화 된 데이터베이스 관리 시스템입니다.
Alertmanager	Alertmanager는 Prometheus 서버와 같은 클라이언트 응용 프로그램에서 보낸 경고를 처리합니다.
Expoter	필요한 매트릭을 수집하고 이를 HTTP 엔드포인트로 Prometheus에 노출하는 역할을 합니다.



# Open Source Software Installation & Application Guide

**nipa** 공개SW역량프라자



이 저작물은 크리에이티브 커먼즈 [저작자표시-비영리-동일조건 변경허락 2.0대한민국 라이선스]에 따라 이용하실 수 있습니다.