

"미래를 여는 열쇠, 오픈인프라"

OpenInfra Days Korea 2025



AI를 활용한 오피버빌리티 똑딱 만들기

#AI #AI개발자모드 #Golang간단개발

#Prometheus #Grafana #Observability



AI(ChatGPT)를 활용한 오픈서버빌리티 똑딱 만들기

Part 1. 인프라 구성

준비물

본 세션관련 깃헙 공간

터미널상으로 Docker를 쉽게 제어하고 확인하기 위한 docker-cli-dashboard

Docker-compose example 제작 요청 (ChatGPT 4o)

Docker-compose를 통한 테스트 서비스 On

Docker-compose를 통한 테스트 서비스 확인

Part 3. 원격 관리도구 만들기

ChatGPT에게 요청할 Prompt 작성 및 전달

ChatGPT가 작성한 코드 테스트 및 Github commit

개발된 웹콘솔을 통해 원격지 개발/테스트 환경 설정 관리하기!

Part 2. Prometheus+Grafana 구성

관리 대상 시스템 환경분석 요청

Grafana + Prometheus 형태의 시스템 구축을 위한 설정

Grafana + Prometheus 형태의 시스템 아키텍처

Part 4. 마무리

완료된 업무에 관련한 매뉴얼 작성 요청

업무 생산성 비교



Part 1. 인프라 구성

AI를 활용한 오피서버빌리티 똑딱 만들기

Part 1. 인프라 구성

준비물

서버 or PC



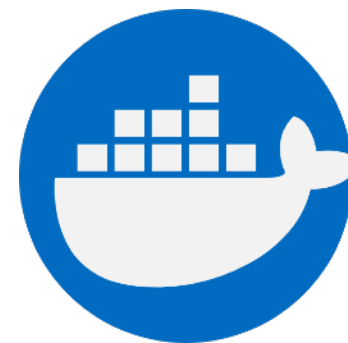
생성형AI 서비스



Gemini



개발환경



기타 오픈소스 도구

[ralf_session](#)

[docker-cli-dashboard](#)

[docker-compose-webctl](#)

Part 1. 인프라 구성

본 세션관련 깃헙 공간

ralfyang / ralf_session

Type / to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

ralf_session

Public

Pin

Watch 0

Fork 0

Star 0

main

Go to file

Code

ralfyang

update for scratch

63e2e3b · 22 minutes ago

1 Commit

monitoring	update for scratch	22 minutes ago
README.md	update for scratch	22 minutes ago
docker-compose.yml	update for scratch	22 minutes ago
nginx.conf	update for scratch	22 minutes ago
package-lock.json	update for scratch	22 minutes ago
package.json	update for scratch	22 minutes ago
server.js	update for scratch	22 minutes ago
setup-monitoring.sh	update for scratch	22 minutes ago
setup-monitoring_update.sh	update for scratch	22 minutes ago

README

세미나 관련 세션 자료 입니다 by Ralf.

준비물

터미널, Docker or [Docker-Desktop](#), [Golang](#), [docker-cli-dashboard](#)

실행

```
git clone https://github.com/ralfyang/ralf_session.git
cd ralf_session
docker compose up -d
```

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

웨бина 자료용 깃헙 공간
https://github.com/ralfyang/ralf_session

Part 1. 인프라 구성

터미널상으로 Docker를 쉽게 제어하고 확인하기 위한 docker-cli-dashboard

docker_cli_dashboard

Public

Unpin

Unwatch 9

Fork 30

Starred 136

master

2 Branches

0 Tags

Go to file

t+

<> Code

ralfyang

Update README.md

83c32c0 · 35 minutes ago

🕒 173 Commits

snap

Create snapcraft.yaml

5 years ago

README.md

Update README.md

35 minutes ago

README_KR.md

Update README_KR.md

5 years ago

dcs

update for ubuntu c\lear command option &...

4 years ago

sample01.png

Add files via upload

9 years ago

README

Here is an execution file http://bit.ly/ralf_dcs

Download & Install

```
curl -sL bit.ly/ralf_dcs -o ./dcs  
sudo chmod 755 ./dcs  
sudo mv ./dcs /usr/local/bin/
```

docker_cli_dashboard

CLi mode dashboard for Docker(in redhat, ubuntu, OSX)

```
[:: Docker Images ::]  
+-----+  
| REPOSITORY | TAG      | IMAGE ID   | CREATED    | SIZE     |  
+-----+  
| python_test | 0.1      | af8e9daaa82 | 31 seconds ago | 183MB    |  
| ubuntu      | latest   | 845da4ddbc  | 4 weeks ago  | 117MB    |  
| busybox     | latest   | 5bd59b26729 | 5 weeks ago  | 1.15MB   |  
+-----+  
  
[:: Active Processors ::]  
+-----+  
| CONTAINER ID | IMAGE          | COMMAND                  | CREATED    | STATUS    | PORTS              | NAMES |  
+-----+  
| 53eebf01750 | 845da4ddbc     | "/bin/sh"                | 2 minutes ago | Up 2 minutes | 0.0.0.0:8080->8080/tcp | test  |  
+-----+  
  
[:: All Processors ::]  
+-----+  
| CONTAINER ID | IMAGE          | COMMAND                  | CREATED    | STATUS    | PORTS              | NAMES |  
+-----+  
| 53eebf01750 | 845da4ddbc     | "/bin/sh"                | 2 minutes ago | Up 2 minutes | 0.0.0.0:8080->8080/tcp | test  |  
+-----+  
  
Please Insert a command as below :)  
+-----+  
| Processor | Image | Command | Status | Ports | Names |  
+-----+  
| Run | sa | Pull Image | [ pull ] | [ Commit & Push ] | [ commit ] |  
Start & Attach | sa | Remove Image | [ rmi ] | Login Registry | [ login ] |  
Stop | stop | Build Image | [ build ] | Tagging | [ push ] |  
Send Command | exec | Purge volume | [ clean ] | Frame work/volume/network | [ prune ] |  
Into the running docker | it | Save All images to tarball | [ save ] | * Star here! https://bit.ly/glt_dcs upgrade check & upgrade | Exit | Ctrl+c |  
Remove PS | rm | Load the image from DIN | [ load ] | Docker version 17.12.1-ce |  
Show log | log | CLI | [ Type free as below ] |  
Purge Processor | purge | ex) stop 8f9b3b 3389 3d2b39 8918 |  
All Processor stop | ps |  
+-----+  
  
Push  
+-----+  
| REPOSITORY | TAG      | IMAGE ID   | CREATED    | SIZE     |  
+-----+  
| python_test | 0.1      | af8e9daaa82 | 37 seconds ago | 183MB    |  
| ubuntu      | latest   | 845da4ddbc  | 4 weeks ago  | 117MB    |  
| busybox     | latest   | 5bd59b26729 | 5 weeks ago  | 1.15MB   |  
+-----+  
== Please choose a number for tag and push [0] - 3 ]  
[ 0 ]  
-- Please login/register to The Docker registry -- ex) docker login  
Input Image python_test:0.1  
-- Do you need to make a tag ? [y / n]  
y  
Please insert a tag name:  
goodbye
```

About

CLI mode dashboard for Docker

Readme

Activity

136 stars

9 watching

30 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 7

Deployments 49

✓ github-pages 2 years ago

+ 48 deployments

Languages

Shell 100.0%

Docker CLI Dashboard

https://github.com/ralfyang/docker_cli_dashboard

```
curl -sL bit.ly/ralf_dcs -o ./dcs
```

```
sudo chmod 755 ./dcs
```

```
sudo mv ./dcs /usr/local/bin/
```

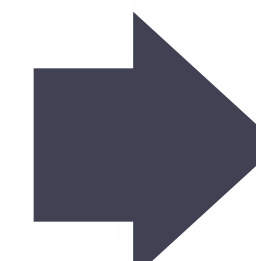
Part 1. 인프라 구성

Docker-compose example 제작 요청 (ChatGPT 4o)

무엇을 도와드릴까요?

docker compose를 통해 개발환경을 구축하려고 합니다.
Nodejs+mysql+nginx 구성으로 진행하려고 합니다. 바로 구동이 가능한
docker-compose.yml 설정을 알려주세요. credential 관련해서는 .env 파일
로 분리해서 작성해주세요
디렉토리는 하위에 두지 않고 하나의 디렉토리에서 다 처리 될 수 있도록 해주세요.
별도 빌드 과정 없이 바로 테스트 할 수 있는 샘플이 필요합니다.

+ 🔍 검색 🔍 심층 리서치 🖼️ 이미지 그리기 ... ⬆



📁 디렉토리 구성

```
pgsql
project-root/
├── docker-compose.yml
├── .env
├── nginx.conf
├── server.js
└── package.json
```

1 .env

```
env
MYSQL_ROOT_PASSWORD=rootpass
MYSQL_DATABASE=testdb
MYSQL_USER=testuser
MYSQL_PASSWORD=testpass
NODE_PORT=3000
```

docker compose를 통해 개발환경을 구축하려고 합니다. Nodejs+mysql+nginx 구성으로 진행하려고 합니다.
바로 구동이 가능한 docker-compose.yml 설정을 알려주세요. credential 관련해서는 .env 파일로 분리해서 작성해주세요
디렉토리는 하위에 두지 않고 하나의 디렉토리에서 다 처리 될 수 있도록 해주세요.
별도 빌드 과정 없이 바로 테스트 할 수 있는 샘플이 필요합니다.

Part 1. 인프라 구성

Test환경 구축을 위한 Docker-compose example 제작 요청 (ChatGPT 4o)

2 package.json

```
json
{
  "name": "node-mysql-app",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^3.4.0"
  }
}
```

4 nginx.conf

```
nginx
events {}

http {
  server {
    listen 80;
    location / {
      proxy_pass http://node:3000;
    }
  }
}
```

3 server.js

```
js
const express = require('express');
const mysql = require('mysql2');
const app = express();

const db = mysql.createConnection({
  host: 'mysql',
  user: process.env.MYSQL_USER || 'testuser',
  password: process.env.MYSQL_PASSWORD || 'testpass',
  database: process.env.MYSQL_DATABASE || 'testdb'
});

db.connect(err => {
  if (err) {
    console.error('DB 연결 실패:', err.message);
  } else {
    console.log('MySQL 연결 성공');
  }
});

app.get('/', (req, res) => {
  db.query('SELECT NOW() as now', (err, results) => {
    if (err) return res.status(500).send(err);
    res.send(`Hello World! DB Time: ${results[0].now}`);
  });
});

const port = process.env.NODE_PORT || 3000;
app.listen(port, () => {
  console.log(`서버 실행 중: http://localhost:${port}`);
});
```

5 docker-compose.yml

```
yml
version: '3.8'

services:
  node:
    image: node:18
    working_dir: /app
    volumes:
      - ./app
    command: sh -c "npm install && npm start"
    ports:
      - "3000:3000"
    env_file:
      - .env
    depends_on:
      - mysql

  mysql:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql

  nginx:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - node

volumes:
  mysql-data:
```

< 예제: https://github.com/ralfyang/ralf_session >

Part 1. 인프라 구성

Test환경 구축을 위한 Docker-compose를 통한 테스트 서비스 On

```
5624650 drwxr-xr-x@ 10 Ralf staff 320 4 4 09:30 .
5491487 drwxr-xr-x@ 4 Ralf staff 128 4 4 09:28 ..
5625685 -rw-r--r--@ 1 Ralf staff 115 4 4 09:28 .env
5626255 drwxr-xr-x@ 12 Ralf staff 384 4 4 09:31 .git
5626254 -rw-r--r--@ 1 Ralf staff 29 4 4 09:30 README.md
5625686 -rw-r--r--@ 1 Ralf staff 737 4 4 09:28 docker-compose.yml
5625687 -rw-r--r--@ 1 Ralf staff 118 4 4 09:28 nginx.conf
5625683 -rw-r--r--@ 1 Ralf staff 35124 4 4 09:28 package-lock.json
5625684 -rw-r--r--@ 1 Ralf staff 212 4 4 09:28 package.json
5624651 -rw-r--r--@ 1 Ralf staff 811 4 4 09:28 server.js
Ralf@Mac example %
Ralf@Mac example %
Ralf@Mac example % docker compose up -d
WARN[0000] /Users/Ralf/dev/byline_network/example/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 15/28
 : nginx [.....] 25.17MB / 68.61MB Pulling 10.9s
   0994e771ba34 Downloading [=====] ... 8.0s
   8e77214beb25 Download complete 1.5s
   d9b636547744 Downloading [=====] ... 8.0s
   4c7c1a5bd3af Download complete 1.5s
   bef2ee7fab45 Download complete 1.4s
   589701e352f8 Download complete 1.6s
   13f89c653285 Download complete 0.2s
 : node [.....] 51.38MB / 384.7MB Pulling 10.9s
   4378a6c11dea Downloading [=====] ... 8.0s
   9af593591e82 Downloading [=====] ... 8.0s
   ca40eb2cc73f Download complete 1.7s
   57497454cc98 Download complete 1.4s
   65dda36d40c1 Download complete 2.3s
   545aa82ec479 Downloading [=====] ... 8.0s
   140d15be2fea Downloading [=====] ... 8.0s
   1d9d474cce08 Downloading [=====] ... 8.0s
 : mysql [.....] 49.28MB / 227.4MB Pulling 10.9s
   475627eb8905 Download complete 0.3s
   8c0d86ac38d6 Download complete 7.7s
   25df6f366614 Download complete 0.9s
   3a25639bb1e3 Download complete 1.6s
   0a1d246ada2d Download complete 1.0s
   403da7fd01e2 Download complete 0.8s
   fa07288abb8b Downloading [=====] ... 8.0s
   4570cc21e155 Downloading [=====] ... 8.0s
   e70c15a7f3f1 Download complete 1.8s
   8227bc928402 Downloading [=====] ... 8.0s
```

git clone https://github.com/ralfyang/ralf_session

cd ralf_session

docker compose up -d

Part 1. 인프라 구성

Test환경 구축을 위한 Docker-compose를 통한 테스트 서비스 확인

```
=====
[:: Docker Images ::]
=====
| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
01 | node | 18 | 3c5624851070 | 7 days ago | 1.56GB |
02 | nginx | latest | 124b44bfc9cc | 8 weeks ago | 280MB |
03 | mysql | 8 | 23818b7d7de4 | 2 months ago | 1.05GB |

=====
[:: Active Processors ::]
=====
| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
01 | f16950a8ba5e | nginx:latest | "/docker-entrypoint.s..." | About a minute ago | Up About a minute | 0.0.0.0:8080->80/tcp | bylinenetwork_ralf_session-nginx-1 |
02 | 506ef75ba95d | node:18 | "docker-entrypoint.s..." | About a minute ago | Up About a minute | 0.0.0.0:3000->3000/tcp | bylinenetwork_ralf_session-node-1 |
03 | b2452514ddbb | mysql:8 | "docker-entrypoint.s..." | About a minute ago | Up About a minute | 0.0.0.0:3306->3306/tcp, 33060/tcp | bylinenetwork_ralf_session-mysql-1 |

=====
[:: All Processors ::]
=====
| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
01 | f16950a8ba5e | nginx:latest | "/docker-entrypoint.s..." | About a minute ago | Up About a minute | 0.0.0.0:8080->80/tcp | bylinenetwork_ralf_session-nginx-1 |
02 | 506ef75ba95d | node:18 | "docker-entrypoint.s..." | About a minute ago | Up About a minute | 0.0.0.0:3000->3000/tcp | bylinenetwork_ralf_session-node-1 |
03 | b2452514ddbb | mysql:8 | "docker-entrypoint.s..." | About a minute ago | Up About a minute | 0.0.0.0:3306->3306/tcp, 33060/tcp | bylinenetwork_ralf_session-mysql-1 |

=====
Please Insert a command as below :)
=====
= Processor ===== = Image ===== = Commit & Push =====
Run the container [ run ] Pull Image [ pull ] Commit [commit]
Start [ start ] Remove Image [ rmi ] Login to registry [ login ]
Stop [ stop ] Build Image [ build ] Tagging & Push [ push ]
Send command [ exec ] Purge volume [ clean ] Prune image/volume/network - [ prune ]
Into the running docker [ it ] Save All images to tarball - [ save ] Container Stats [ stat ]
Remove PS [ rm ] Load the image from DIR [ load ] * Star here! - https://bit.ly/git_dcs
Show log [ log ] upgrade check & upgrade upgrade
Purge Processor [ purge ] Cli [ Type free as below ] Exit Ctrl+c
All Processor stop!! [ aps ] ex) stop 910930 19809 102039 09103 Docker version 28.0.1
=====
CMD>> |
```

← → ↺ ⓘ localhost:8080

pretty print 적용 ☒

```
{
  "fatal": true
}
```




Part 2. Prometheus+Grafana 구성

AI를 활용한 오피서버빌리티 똑딱 만들기

Part 2. Prometheus+Grafana 구성

관리 대상 시스템 환경분석 요청

오피서버빌리티 예제 만들기

개요

- 바이라인 네트워크 웨비나 세션에서 다룰 오피서버빌리티 예제를 만들고 테스트를 진행하고자 함

주요 구성 방안

- Docker-compose를 활용하여 개발/테스트 환경을 구축하였으며, 관련된 시스템 설정은 하위 docker-compose.yml을 참조
- 관리 대상서버와 그라파나 대시보는 동일 서버상에 존재하지만, 분리 가능하도록 docker-compose.yml 파일과 디렉토리를 분리해서 관리할 예정
- Prometheus + Grafana 구성으로 오피서버빌리티 기초를 구성할 예정이며, Grafana Dashboard 구성을 위한 화면 설정 도움 필요

관리대상 시스템 구성

• docker-compose.yml

```
version: '3.8'

services:
  node:
    image: node:18
    working_dir: /app
    volumes:
      - ./app
    command: sh -c "npm install && npm start"
    ports:
      - "3000:3000"
    env_file:
      - .env
    depends_on:
      - mysql

  mysql:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql

  nginx:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - node

volumes:
  mysql-data:|
```

• .env

```
MYSQL_ROOT_PASSWORD=rootpass
MYSQL_DATABASE=testdb
MYSQL_USER=testuser
MYSQL_PASSWORD=testpass
NODE_PORT=3000
```



- 첨부 이미지의 조건 확인 후 해당 시스템 구축을 위해 docker-compose를 통해 쉽게 구축 할 수 있도록 docker-compose.yml, prometheus.yml 등 필요 파일을 생성해주세요.
- 기존 docker-compose.yml은 그대로 두고, prometheus와 exporter, grafana에 관련된 별도의 docker-compose 파일을 생성해주세요
- 시스템 상태 확인을 위한 cadvisor 등의 설정도 추가해주세요



- 첨부 이미지의 조건 확인 후 해당 시스템 구축을 위해 docker-compose를 통해 쉽게 구축 할 수 있도록 docker-compose.yml, prometheus.yml 등 필요 파일을 생성해주세요.
- 기존 docker-compose.yml은 그대로 두고, prometheus와 exporter, grafana에 관련된 별도의 docker-compose 파일을 생성해주세요
- 시스템 상태 확인을 위한 cadvisor 등의 설정도 추가해주세요

Part 2. Prometheus+Grafana 구성

관리 대상 시스템 환경분석 요청: 결과생성

좋습니다! 첨부해주신 정보를 기반으로, 기존 `docker-compose.yml` 은 그대로 유지하고, **Prometheus, Grafana, Node Exporter, cAdvisor**를 활용한 오피서빌리티 구성을 위한 별도의 `docker-compose.monitoring.yml`, `prometheus.yml`, 그리고 Grafana 설정 디렉토리 구조 등을 아래에 제안드립니다.

📁 디렉토리 구조 예시

```
project-root/
├── docker-compose.yml
├── .env
├── monitoring/
│   ├── docker-compose.monitoring.yml
│   ├── prometheus/
│   │   └── prometheus.yml
│   └── grafana/
│       ├── provisioning/
│       │   ├── datasources/
│       │   │   └── datasource.yml
│       │   └── dashboards/
│       │       └── dashboard.json (원하는 경우)
```

```
monitoring/docker-compose.monitoring.yml

yaml

version: '3.8'

services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    volumes:
      - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - "9090:9090"
  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3001:3000"
    volumes:
      - grafana-storage:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning
    depends_on:
      - prometheus
  node-exporter:
    image: prom/node-exporter:latest
    container_name: node-exporter
    ports:
      - "9100:9100"
  cAdvisor:
    image: gcr.io/cadvisor/cadvisor:latest
    container_name: cAdvisor
    ports:
      - "8081:8080"
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:ro
      - /sys:/sys:ro
      - /var/lib/docker:/var/lib/docker:ro

volumes:
  grafana-storage:
```

```
monitoring/prometheus/prometheus.yml

yaml

global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['prometheus:9090']
  - job_name: 'node-exporter'
    static_configs:
      - targets: ['node-exporter:9100']
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:8080']
```

📊 Grafana 기본 데이터소스 설정 (선택)

monitoring/grafana/provisioning/datasources/datasource.yml

```
yaml

apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    url: http://prometheus:9090
    isDefault: true
```

💡 필요한 경우 Dashboards도 provisioning/dashboards/ 에 json 파일로 넣어 자동 설정 가능.

실행 방법

```
bash

cd monitoring
docker-compose -f docker-compose.monitoring.yml up -d
```


Part 2. Prometheus+Grafana 구성

Grafana + Prometheus 형태의 시스템 구축을 위한 설정 요청

공유하기



앞서 알려주신 디렉토리 구조와 yml 등의 설정 파일을 한번에 입력할 수 있게 bash script로 작성해 주실 수 있나요?
예를 들어, macos의 콘솔의 현재 디렉토리 하위로 mkdir로 monitoring 디렉토리를 만들고, cat 과 EOF 를 사용해서 ./monitoring/docker-compose.monitoring.yml 에 바로 설정값들이 기입될 수 있도록 inline command를 작성해주시면 좋겠습니다.

앞서 알려주신 디렉토리 구조와 yml 등의 설정 파일을 한번에 입력할 수 있게 bash script로 작성해 주실 수 있나요? 예를 들어, macos의 콘솔의 현재 디렉토리 하위로 mkdir로 monitoring 디렉토리를 만들고, cat 과 EOF 를 사용해서 ./monitoring/docker-compose.monitoring.yml 에 바로 설정값들이 기입될 수 있도록 inline command를 작성해주시면 좋겠습니다.

물론입니다! 아래는 macOS 콘솔 환경에서 실행 가능한 Bash 스크립트로, 현재 디렉토리 하위에 monitoring/ 디렉토리를 생성하고, docker-compose.monitoring.yml, prometheus.yml, Grafana provisioning 설정 (datasource.yml) 파일까지 한 번에 생성해주는 예제입니다.

https://github.com/ralfyang/ralf_session/blob/main/setup-monitoring.sh

✓ Bash 스크립트 (한 번에 복사-붙여넣기 가능)

bash

복사

편집

#!/bin/bash

monitoring 디렉토리 생성

mkdir -p monitoring/prometheus

mkdir -p monitoring/grafana/provisioning/datasources

mkdir -p monitoring/grafana/provisioning/dashboards

docker-compose.monitoring.yml 생성

cat <<EOF > monitoring/docker-compose.monitoring.yml

version: '3.8'

services:

prometheus:

image: prom/prometheus:latest

container_name: prometheus

volumes:

- ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml

ports:

- "9090:9090"

grafana:

image: grafana/grafana

container_name: grafana

ports:

- "3001:3000"

volumes:

- grafana-storage:/var/lib/grafana

- ./grafana/provisioning:/etc/grafana/provisioning

depends_on:

- prometheus

node-exporter:

📌 사용법

1. 위 코드를 복사하여 setup-monitoring.sh 같은 이름으로 저장하세요.

2. 실행 권한 부여:

bash

복사

편집

chmod +x setup-monitoring.sh

3. 실행:

bash

복사

편집

./setup-monitoring.sh

✓ 결과적으로

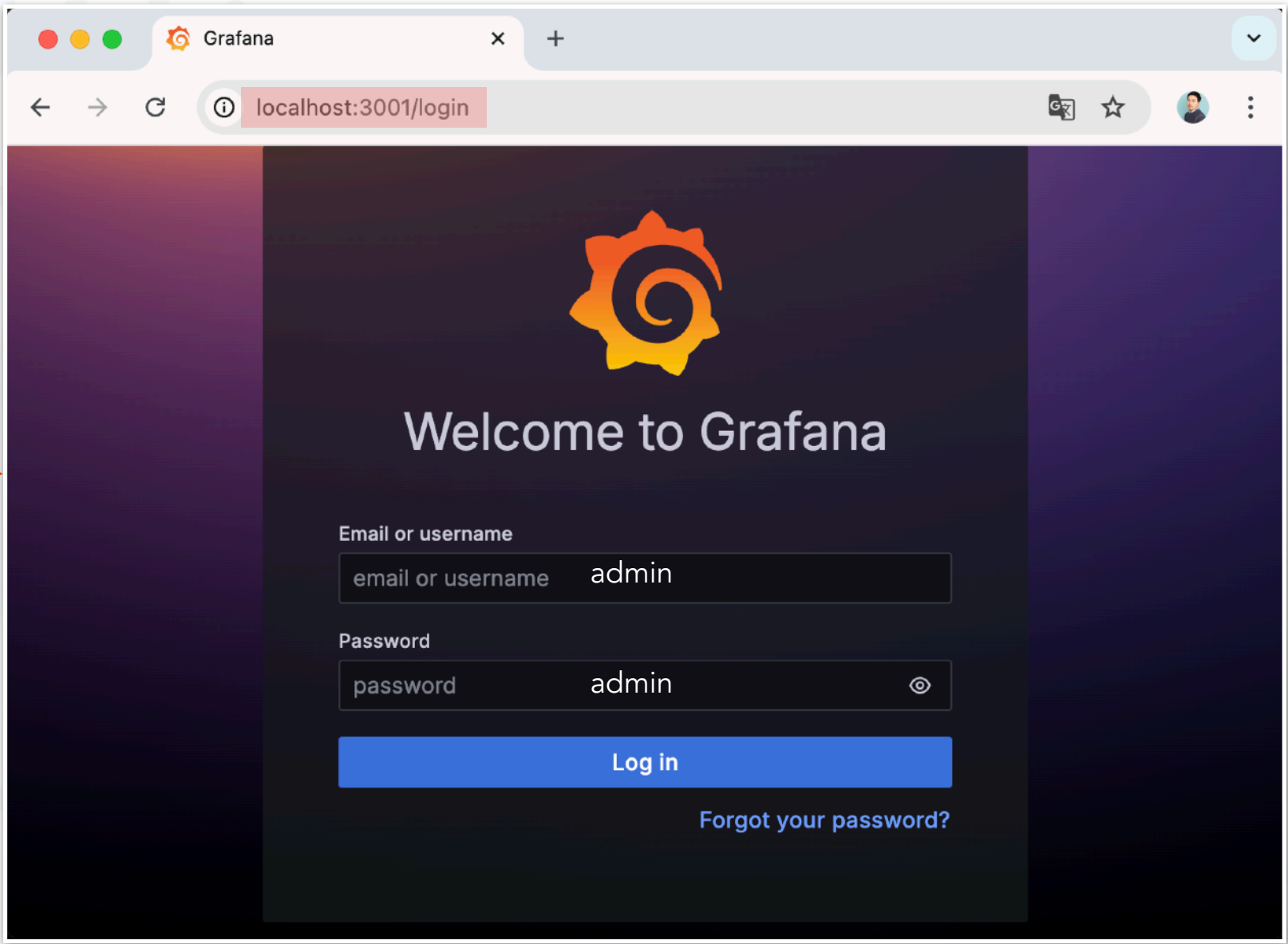
- http://localhost:9090 → Prometheus
- http://localhost:3001 → Grafana
- http://localhost:8081 → cAdvisor 웹 UI도 사용 가능

Part 2. Prometheus+Grafana 구성

Grafana + Prometheus 형태의 시스템 구축을 위한 설정 적용

```
Ralf@Mac bylinenetwork_ralf_session % vi setup-monitoring.sh
Ralf@Mac bylinenetwork_ralf_session % chmod +x setup-monitoring.sh

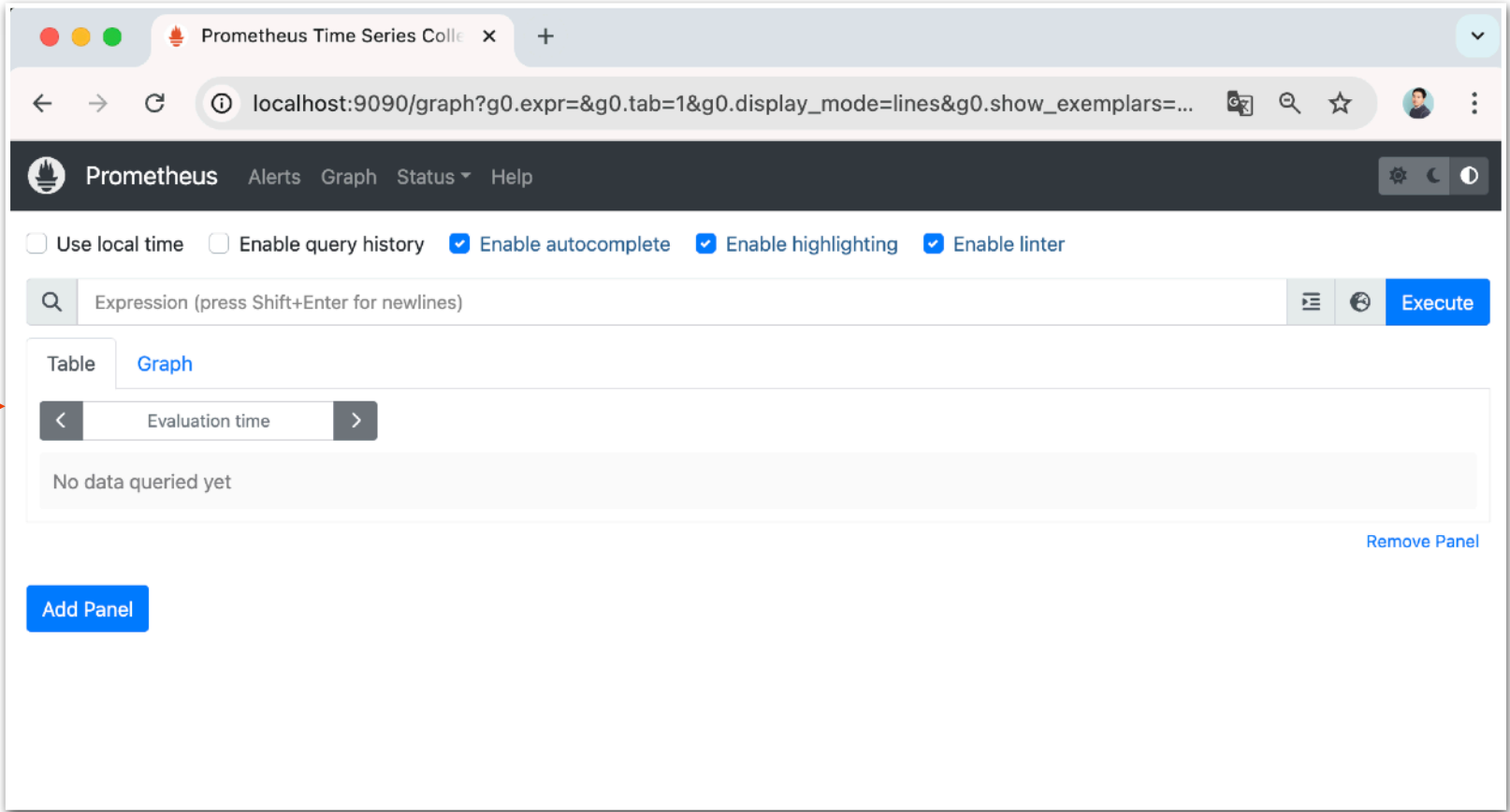
Ralf@Mac bylinenetwork_ralf_session % ll
total 128
5638903 drwxr-xr-x@ 12 Ralf  staff   384  4  4 10:55 .
5491487 drwxr-xr-x@  4 Ralf  staff   128  4  4 10:12 ..
5638977 -rw-r--r--@  1 Ralf  staff   115  4  4 10:12 .env
5638904 drwxr-xr-x@ 12 Ralf  staff   384  4  4 10:12 .git
5638978 -rw-r--r--@  1 Ralf  staff   484  4  4 10:12 README.md
5638979 -rw-r--r--@  1 Ralf  staff   737  4  4 10:12 docker-compose.yml
5638980 -rw-r--r--@  1 Ralf  staff   118  4  4 10:12 nginx.conf
5638987 drwxr-xr-x  82 Ralf  staff  2624  4  4 10:12 node_modules
5638981 -rw-r--r--@  1 Ralf  staff  35124  4  4 10:12 package-lock.json
5638982 -rw-r--r--@  1 Ralf  staff   212  4  4 10:12 package.json
5638983 -rw-r--r--@  1 Ralf  staff   811  4  4 10:12 server.js
5651276 -rwxr-xr-x@  1 Ralf  staff   1927  4  4 10:55 setup-monitoring.sh
Ralf@Mac bylinenetwork_ralf_session % ./setup-monitoring.sh
[✓] Monitoring 설정 파일이 생성되었습니다.
[➡] 실행 : cd monitoring && docker-compose -f docker-compose.monitoring.yml up -d
Ralf@Mac bylinenetwork_ralf_session % cd monitoring && docker-compose -f docker-compose.monitoring.yml up -d
WARN[0000] /Users/Ralf/dev/byline_network/bylinenetwork_ralf_session/monitoring/docker-compose.monitoring.yml: the attribute `version`
[+] Running 30/32
  grafana [#####] 165.4MB / 166.7MB Pulling
  ✓ babadc1b811e Download complete
  ✓ 6e771e15690e Pull complete
  ✓ 3dd8d8d6a974 Pull complete
  ✓ b23536fc5777 Pull complete
  ✓ 3a118e96ded1 Pull complete
  ✓ 8318ded4e9cc Pull complete
  ✓ 756b878b6aa7 Pull complete
  ✓ de210d0897ff Pull complete
  : bf988e0055d8 Extracting      1 s
  ✓ 1cb82048f91d Download complete
  ✓ node-exporter Pulled
```



[:: Docker Images ::]				
	REPOSITORY	TAG	IMAGE ID	CREATED
01	prom/node-exporter	latest	d00a542e409e	2 days ago
02	node	18	3c5624851070	7 days ago
03	grafana/grafana	latest	62d2b9d20a19	13 days ago
04	prom/prometheus	latest	502ad90314c7	2 weeks ago
05	nginx	latest	124b44bfc9cc	8 weeks ago
06	mysql	8	23818b7d7de4	2 months ago
07	gcr.io/cadvisor/cadvisor	latest	3cde6faf0791	13 months ago

[:: Active Processors ::]				
	CONTAINER ID	IMAGE	COMMAND	CREATED
01	f1a61a068894	grafana/grafana:latest	"/run.sh"	10 s
02	b07b8f188671	prom/prometheus:latest	"/bin/prometheus --c..."	10 s
03	1d855be29784	gcr.io/cadvisor/cadvisor:latest	"/usr/bin/cadvisor -..."	10 s
04	1e9a1930f56b	prom/node-exporter:latest	"/bin/node_exporter"	10 s
05	f16950a8ba5e	nginx:latest	"/docker-entrypoint.s..."	45 minutes ago
06	506ef75ba95d	node:18	"/docker-entrypoint.s..."	45 minutes ago
07	b2452514ddb	mysql:8	"/docker-entrypoint.s..."	45 minutes ago

PORTS	NAMES
0.0.0.0:3001->3000/tcp	grafana
0.0.0.0:9090->9090/tcp	prometheus
0.0.0.0:8081->8080/tcp	cadvisor



Part 2. Prometheus+Grafana 구성

Grafana + Prometheus 형태의 시스템 구축을 위한 설정 적용 (nginx, nodejs, mysql exporter 추가 설치)

nginx, nodejs, mysql 상태 확인을 위한 exporter는 설정은 안되어있네요?

맞습니다! 😊
현재 docker-compose.monitoring.yml 구성에는 node-exporter (시스템), cadvisor (컨테이너)만 있고, 애플리케이션 레벨 (nginx, node.js, mysql) 상태를 수집하는 exporter 설정은 빠져 있습니다.

✅ 각 애플리케이션별 Exporter 정리

서비스	Exporter	Docker Image	설명
MySQL	mysqld-exporter	prom/mysqld-exporter	DB 상태, 쿼리, 커넥션 등
Node.js	prom-client 내장 + custom endpoint	앱 코드 수정 필요	/metrics endpoint 노출 필요
Nginx	nginx-exporter or nginx-lua-prometheus	nginx/nginx-prometheus-exporter	요청 수, 응답 시간 등

좋습니다! 요청하신 대로:

- Prometheus + Grafana + Node Exporter + cAdvisor
- MySQL Exporter
- NGINX Exporter
- .env 설정 포함
- Grafana provisioning 포함

위 구성을 한 번에 설정할 수 있도록 하는 setup-monitoring.sh 스크립트를 아래에 작성해드렸습니다. macOS 또는 Linux에서 바로 실행 가능합니다.

- MySQL: Dashboard ID 7362
- Node.js: 직접 커스터마이징 필요 (혹은 Prometheus Stats)
- NGINX: Dashboard ID 2949

원하시면 전체 통합된 docker-compose.monitoring.yml 예제와 함께 .env 값까지 정리해 드릴게요. 계속 도와드릴까요? 😊

네 전체 통합된 docker-compose.monitoring.yml 파일과 .env 값 등이 한번의 스크립트로 정리되도록, setup-monitoring.sh 파일을 작성해주세요!

✅ setup-monitoring.sh

```
bash
#!/bin/bash

# monitoring 디렉토리 생성
mkdir -p monitoring/{prometheus,grafana/provisioning/datasources,grafana/provisioning}

# docker-compose.monitoring.yml 작성
cat <<EOF > monitoring/docker-compose.monitoring.yml
version: '3.8'

services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    volumes:
      - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
```

Part 2. Prometheus+Grafana 구성

Grafana + Prometheus 형태의 시스템 구축을 위한 설정 적용 (nginx, nodejs, mysql exporter 추가 설치)

```
Ralf@Mac bylinenetwork_ralf_session % vi setup-monitoring_update.sh
Ralf@Mac bylinenetwork_ralf_session %
Ralf@Mac bylinenetwork_ralf_session %
Ralf@Mac bylinenetwork_ralf_session % ./setup-monitoring
zsh: no such file or directory: ./setup-monitoring
Ralf@Mac bylinenetwork_ralf_session % ./setup-monitoring_update.sh
✅ Monitoring 구성 파일이 모두 생성되었습니다!
📁 경로 : ./monitoring/
🚀 실행 : cd monitoring && docker compose -f docker-compose.monitoring.yml up -d
👤 Grafana 로그인 → http://localhost:3001 (admin / admin)
Ralf@Mac bylinenetwork_ralf_session % cd monitoring && docker compose -f docker-compose.monitoring.yml up -d
[+] Running 29/37
✔ mysql_exporter Pulled 11.6s
✔ c656b4d8066d Pull complete 6.7s
✔ grafana [#####] 29.44MB / 166.8MB Pulling 12.1s
✔ babadc1b811e Download complete 1.0s
✔ 8318ded4e9cc Pull complete 1.3s
✔ 756b878b6aa7 Pull complete 1.4s
✔ 6e771e15690e Pull complete 4.9s
✔ de210d0897ff Downloading [=====] 15.73MB/95.64MB 8.5s
✔ 3dd8d8d6a974 Pull complete 1.0s
✔ 1cb82048f91d Download complete 1.7s
✔ 3a118e96ded1 Pull complete 1.7s
✔ b23536fc5777 Pull complete 7.6s
✔ bf988e0055d8 Downloading [=====] 7.34MB/63.48MB 8.5s
✔ nginx_exporter Pulled 5.5s
✔ 6baa17dd27f6 Pull complete 2.0s
✔ e1a1073c0356 Pull complete 2.0s
✔ prometheus [#####] 51.39MB / 99.44MB Pulling 12.1s
✔ 8c7398207505 Download complete 1.3s
✔ 3b41400fc002 Download complete 0.9s
✔ 072b85192f37 Download complete 1.3s
✔ f76e2e4626a7 Downloading [=====] 23.07MB/47.25MB 8.5s
✔ 3b84d8789434 Download complete 1.5s
✔ 9e81dc407c65 Download complete 1.3s
✔ 1cc8183b9830 Download complete 2.5s
✔ ef625c52d165 Download complete 1.8s
✔ aee954281d32 Downloading [=====] 28.31MB/52.18MB 8.5s
✔ ba3925e593cc Download complete 2.6s
✔ cadvisor [#####] 24.37MB / 29.22MB Pulling 12.1s
✔ 9526bc162c17 Download complete 5.5s
✔ c6b39de5b339 Pull complete 3.2s
✔ 6319d9aee214 Download complete 2.2s
✔ 4f4fb700ef54 Already exists 0.0s
✔ 5b10debc8f1f Downloading [=====] 4.194MB/6.254MB 9.4s
✔ node_exporter Pulled 6.6s
✔ cceed781d889 Pull complete 3.1s
✔ 6ce8b87a9754 Download complete 1.4s
✔ d2f8aae8d80e Pull complete 1.5s
```


Part 2. Prometheus+Grafana 구성

mysql에는 exporter가 접근해서 확인 할 수 있도록 db 설정이 필요할 것 같습니다

Grafana + Prometheus 형태의 시스템 구축을 위한 설정 적용 상태 확인



Targets

All scrape pools ▾

All Unhealthy Collapse All

Filter by endpoint or labels

Unknown Unhealthy Healthy

cadvisor (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://cadvisor:8080/metrics	UP	instance="cadvisor:8080" job="cadvisor"	12.831s ago	22.776ms	

mysql-exporter (0/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://mysql_exporter:9104/metrics	DOWN	instance="mysql_exporter:9104" job="mysql-exporter"	7.351s ago	11.814ms	Get "http://mysql_exporter:9104/metrics": dial tcp: lookup mysql_exporter on 127.0.0.11:53: no such host

nginx-exporter (1/1 up) show more

node-exporter (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://node_exporter:9100/metrics	UP	instance="node_exporter:9100" job="node-exporter"	13.709s ago	68.527ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
----------	-------	--------	-------------	-----------------	-------

맞습니다! 아주 잘 짚으셨습니다 🙌

현재 Prometheus가 mysql_exporter 를 찾지 못하거나, mysql_exporter 컨테이너가 MySQL DB에 인증 실패 혹은 접속 거부로 인해 Down 상태입니다.

✅ 해결 단계 1: MySQL Exporter 사용자 생성 (MySQL DB 설정)

MySQL에 아래 명령어를 실행해 주세요:

```
sql
CREATE USER 'testuser'@'%' IDENTIFIED BY 'testpass';
GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'testuser'@'%';
FLUSH PRIVILEGES;
```

testuser / testpass 는 .env 파일에서 사용한 값과 일치해야 합니다.
% 는 모든 IP에서의 접속 허용, Docker 컨테이너에서의 연결을 허용하기 위함입니다.

✅ 해결 단계 2: .env 파일 확인 (이미 있으나 재확인)

```
env
MYSQL_USER=testuser
MYSQL_PASSWORD=testpass
MYSQL_HOST=mysql
```

위와 같이 되어 있어야 하며, docker-compose 의 mysql_exporter 는 이를 참조합니다:

```
yaml
environment:
  - DATA_SOURCE_NAME=${MYSQL_USER}:${MYSQL_PASSWORD}@(${MYSQL_HOST}:3306)/
```

Part 2. Prometheus+Grafana 구성

Grafana + Prometheus 형태의 시스템 구축을 위한 설정 적용 상태 확인

Prometheus Alerts Graph Status ▾ Help

Targets

All scrape pools ▾

All Unhealthy Expand All

🔍

Filter by endpoint or labels

cadvisor (1/1 up)

show less

Endpoint	State	Labels	Last Scrape
http://cadvisor:8080/metrics	UP	instance="cadvisor:8080" job="cadvisor" ▾	8.113s ago

mysql-exporter (1/1 up)

show less

Endpoint	State	Labels	Last Scrape
http://mysql_exporter:9104/metrics	UP	instance="mysql_exporter:9104" job="mysql-exporter" ▾	7.108s ago

nginx-exporter (1/1 up)

show more

node-exporter (1/1 up)

show less

Endpoint	State	Labels	Last Scrape
http://node_exporter:9100/metrics	UP	instance="node_exporter:9100" job="node-exporter" ▾	360.000ms ago

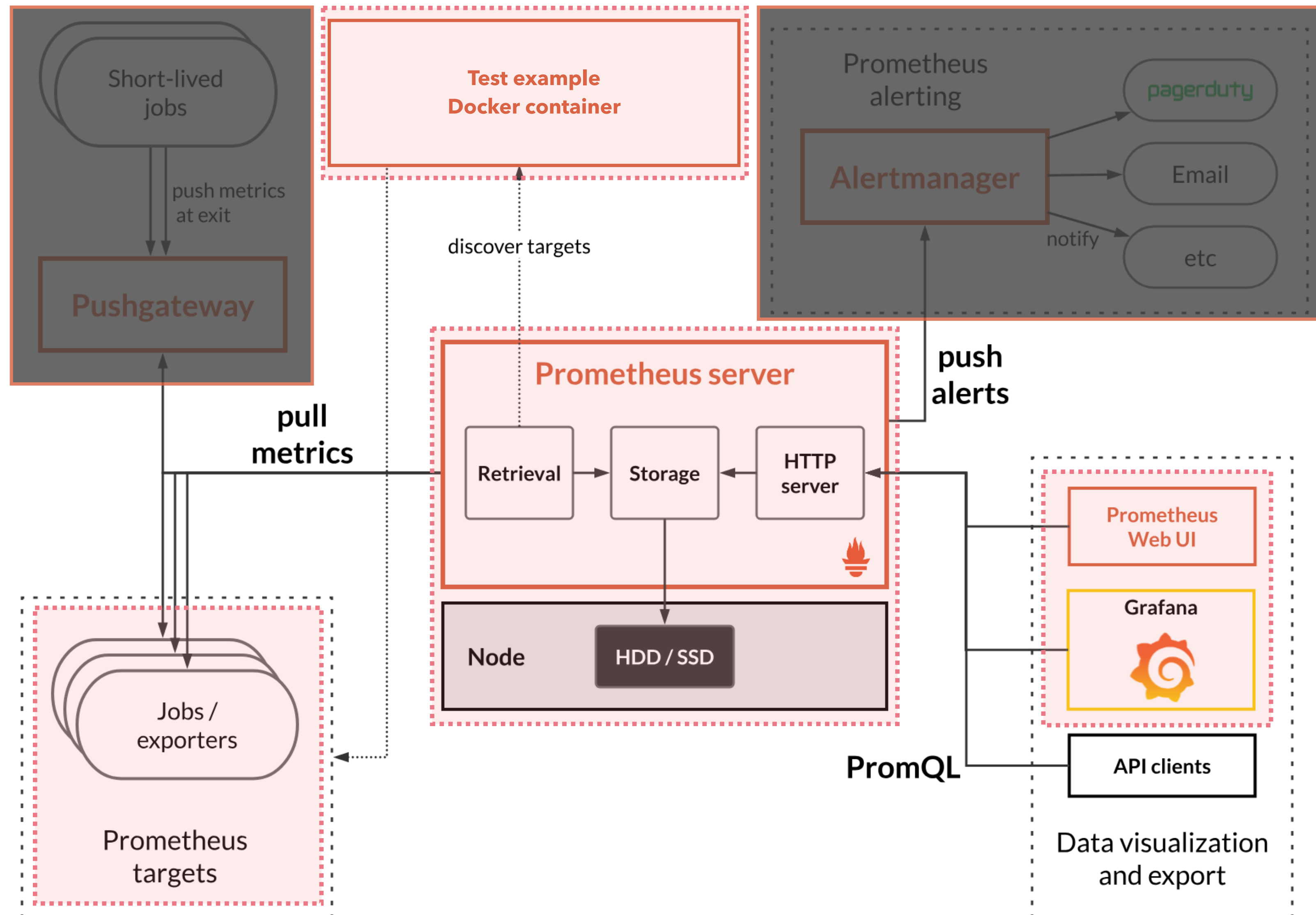
prometheus (1/1 up)

show less

Endpoint	State	Labels	Last Scrape
http://prometheus:9090/metrics	UP	instance="prometheus:9090" job="prometheus" ▾	8.101s ago

Part 2. Prometheus+Grafana 구성

(오늘의)Grafana + Prometheus 형태의 시스템 아키텍처





2106	B10	11:05am	On Time
4547	D01	11:15am	Boarding
780	B12	1:30pm	On Time
4649	C03	11:05am	Boarding
5296	E83	3:00pm	On Time
6729	E83	2:00pm	On Time
7383	D09	11:00am	Boarding
	E70	11:10am	On Time
	B7	11:09am	On Time
			Boarding

Part 2. Grafana Dashboard 설정

AI를 활용한 옴저버빌리티 똑딱 만들기

Part 2. Grafana Dashboard 설정

Prometheus exporter 상태 확인 및 적절한 Grafana dashboard ID 추천 요청

PrometheusAlertsGraphStatus▼Help

Targets

All scrape pools▼AllUnhealthyExpand All

Filter by endpoint or labels

cadvisor (1/1 up)show less

Endpoint	State	Labels	Last Scrape
http://cadvisor:8080/metrics	UP	instance="cadvisor:8080"job="cadvisor"▼	8.113s ago

mysql-exporter (1/1 up)show less

Endpoint	State	Labels	Last Scrape
http://mysql_exporter:9104/metrics	UP	instance="mysql_exporter:9104"job="mysql-exporter"▼	7.108s ago

nginx-exporter (1/1 up)show more

node-exporter (1/1 up)show less

Endpoint	State	Labels	Last Scrape
http://node_exporter:9100/metrics	UP	instance="node_exporter:9100"job="node-exporter"▼	360.000ms ago

prometheus (1/1 up)show less

Endpoint	State	Labels	Last Scrape
http://prometheus:9090/metrics	UP	instance="prometheus:9090"job="prometheus"▼	8.101s ago

지금까지의 설정을 그라파나 대시보드로 구성하기 위해 추천 ID 값을 알려주세요

지금까지 Prometheus에 정상적으로 연결된 Exporter들이 모두 잘 수집되고 있으니, 이제 Grafana에 아래 추천 Dashboard ID들을 추가하면 바로 시각화가 가능합니다! 🔥



✅ 추천 Grafana Dashboard ID 목록

모듈	Dashboard 이름	Dashboard ID	설명
Node Exporter	Node Exporter Full	1860	CPU, 메모리, 디스크, 네트워크 등 시스템 리소스 전반
cAdvisor (Docker)	Docker + cAdvisor Metrics	193	컨테이너별 리소스 상태 (CPU, 메모리 등)
MySQL Exporter	MySQL Server Dashboard	7362	쿼리, InnoDB, 커넥션, 캐시, 슬로우 쿼리 등
Nginx Exporter	Nginx Server Stats	2949	요청 수, 응답 속도, 상태코드 등 NGINX 모니터링
Prometheus 자체 상태	Prometheus Stats	2	Prometheus 자체 상태 (scrape 수, target 등)

Part 2. Grafana Dashboard 설정

Grafana dashboard ID를 활용한 Dashboard Import

모듈	Dashboard 이름	Dashboard ID	설명
Node Exporter	Node Exporter Full	1860	CPU, 메모리, 디스크, 네트워크 등 시스템 리소스 전반
Cadvisor (Docker)	Docker + CAdvisor Metrics	193	컨테이너별 리소스 상태 (CPU, 메모리 등)

Home > Dashboards > Home

Welcome to Grafana

Home > Dashboards > Import dashboard

Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse

Accepted file types: .json, .txt

Find and import dashboards for common applications at grafana.com/dashboards

193

Load

Import dashboard

Import dashboard from file or Grafana.com

Importing dashboard from Grafana.com

Published by: kokorinav

Updated on: 2021-04-21 21:31:33

Options

Name: Cadvisor exporter

Folder: Dashboards

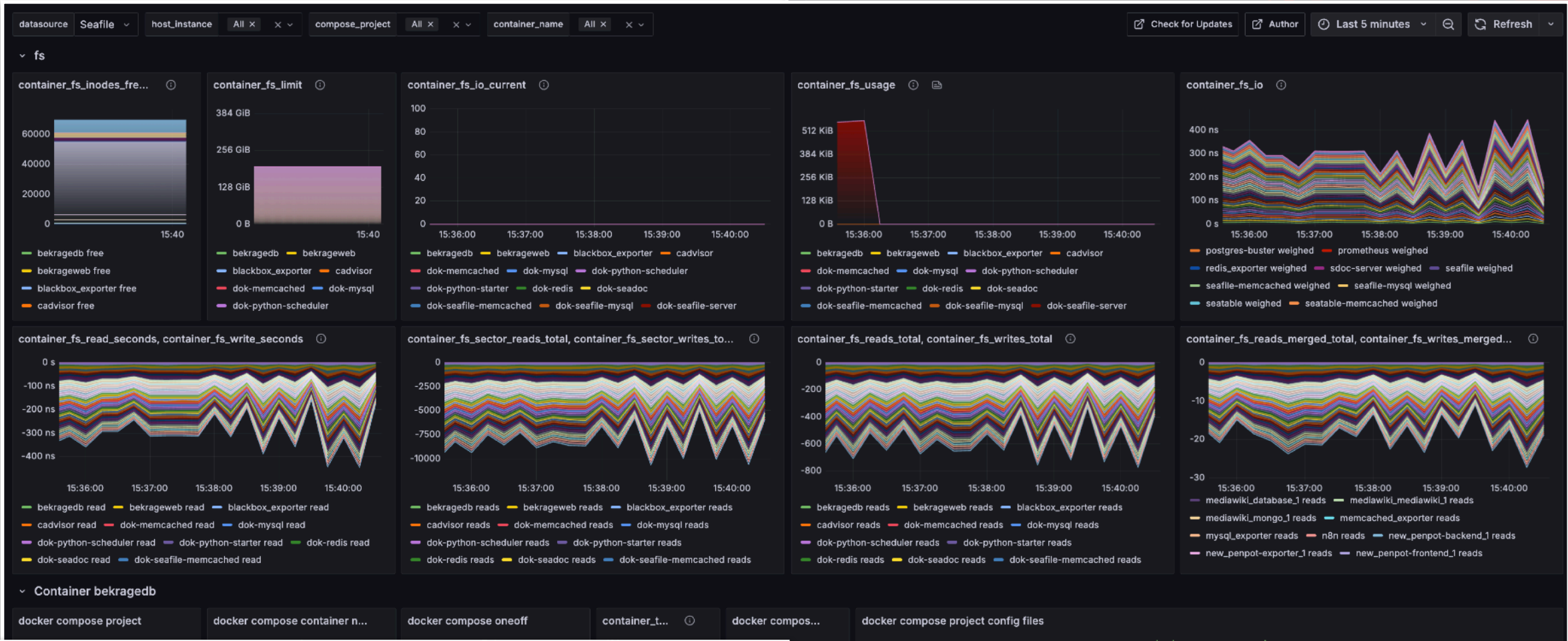
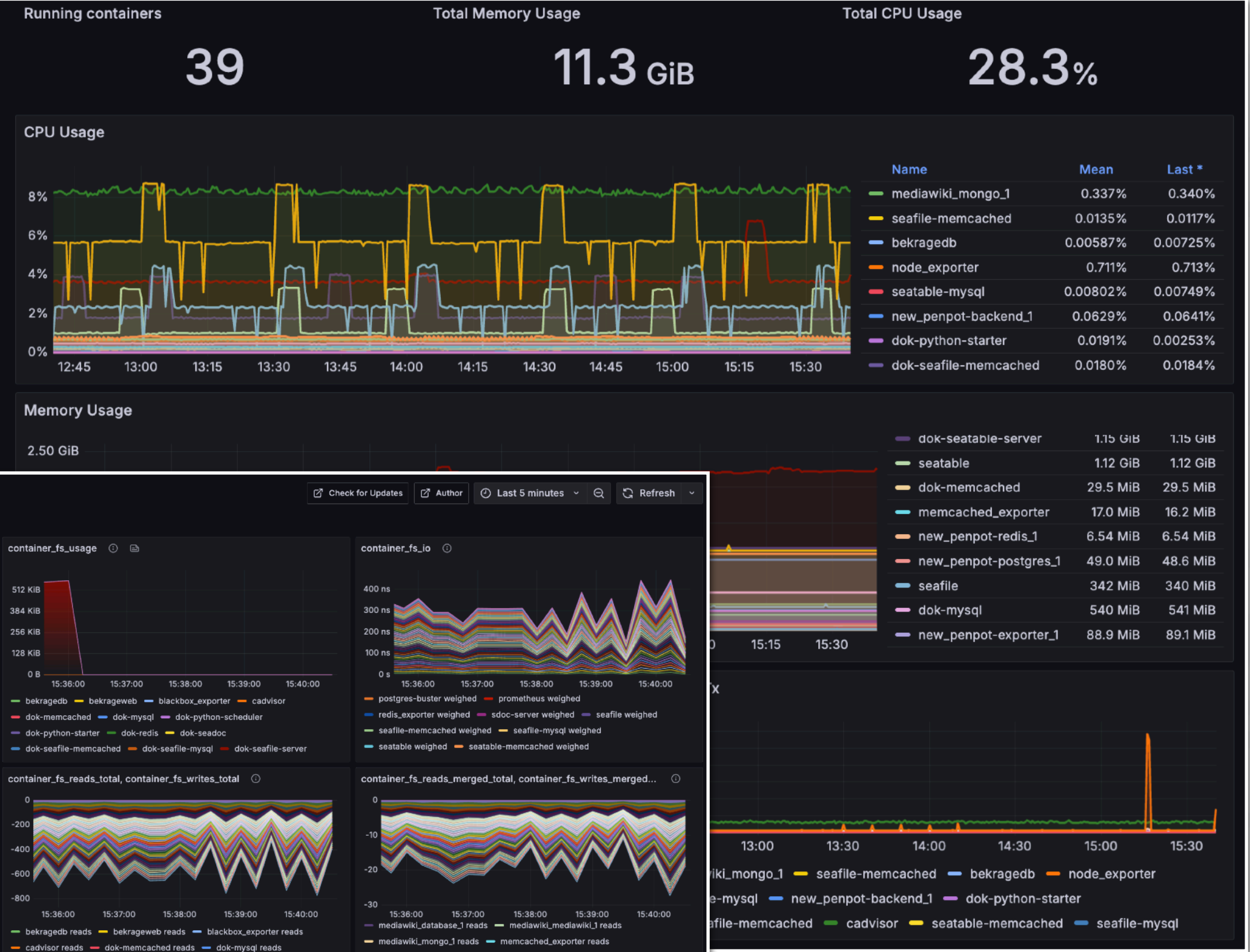
Unique identifier (UID): pMEd7m0Mz [Change uid](#)

Prometheus: Prometheus as the datasource is obligatory

Prometheus default Prometheus

Part 2. Grafana Dashboard 설정

Grafana dashboard 구성완료!





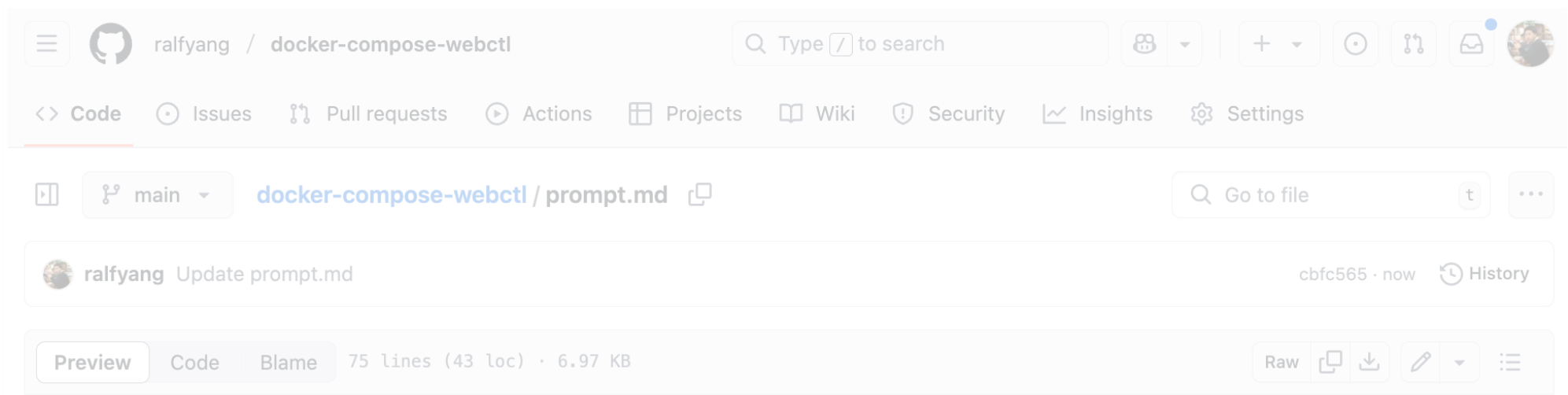
Part 3. 원격 관리도구 만들기

AI를 활용한 옴저버빌리티 똑딱 만들기

Part 3. 원격 관리도구 만들기

ChatGPT에게 요청할 Prompt 작성 및 전달(ChatGTP o3-mini-high)

<https://github.com/ralfyang/docker-compose-webctl/blob/main/prompt.md>



Go lang을 통해 간단하게 리눅스 시스템을 컨트롤할 수 있는 웹콘솔을 개발하려고 합니다.

간단하게 회원가입이 가능하고, 로그인한 첫번째 사람이 어드민 권한을 갖습니다.

어드민이 아닐 경우 로그인을 하더라도 어드민 권한이 필요하다는 메시지만 확인이 됩니다.

회원 가입 시, 이메일을 통해 어드민에게 권한 승인 이메일이 발송됩니다.

웹을 통해 docker-compose.yml 파일의 위치와 파일명을 등록 할 수 있으며, 웹 에디터를 통해 yaml 파일 편집을 위한 편집기가 제공됩니다.

웹을 통해 컨트롤을 하는 대상은 특정 경로의 docker-compose.yml 파일입니다.

웹을 통해 docker-compose.yml 파일의 위치와 파일명을 등록 할 수 있으며, 웹 에디터를 통해 yaml 파일 편집을 위한 편집기가 제공됩니다.

편집기를 통해 편집된 파일은 앞에서 지정해둔 파일로 저장되지만, 기존의 파일명은 [기존파일명_날짜시간초.yaml]로 별도로 백업파일을 저장합니다.

도커 컴포즈의 재시작 명령어는 docker-compose -f [지정해둔 파일명] down; sleep 2; docker-compose up -d 이며, 해당 도커 컴포즈 파일이 있는 위치에서 실행을 합니다.

사용자 정보는 별도의 DB가 없는 관계로, 현재 디렉토리에 .account 파일로 저장하며, 형식은, 이메일,해시된 비밀번호,어드민 유무 형태로 3개의 인자로 구성해서 적용해주세요

답변은 한글로 해주시고, 코드를 작성해주세요

docker-compose 설정파일의 수정에 있어서, 경로를 웹콘솔에서 아무곳이나 지정 할 경우 보안상 큰 위험이 될 수 있을것 같습니다. 웹콘솔이 동작하는 하위 디렉토리에 위치한 디렉토리와 그 안에 파일만 터치 할 수 있도록 해야 하겠습니다. 그리고 웹 콘솔은 현재 왼쪽 정렬로 되어있는데, 가운데 정렬로 바꿔주세요. 그리고 보여지는 화면이 좀 더 이뻐지면 좋겠어요!

1. 사용자 인증 및 권한 관리

- 회원가입 및 최초 어드민 지정:
 - 회원가입 시 기본적으로 '권한없음'으로 등록합니다.
 - 최초 가입 사용자가 로그인하면 자동으로 어드민 권한이 부여됩니다.
- 이메일 승인 프로세스:
 - 신규 회원 가입 시, 시스템은 등록된 이메일로 어드민에게 승인 요청 메일을 발송합니다.
 - 어드민은 전용 권한관리 메뉴에서 '어드민' 또는 '권한없음'을 할당할 수 있습니다.
- 접근 제어:
 - 로그인 후 어드민 권한이 없는 사용자는 "어드민 권한이 필요하다"는 메시지만 확인할 수 있도록 미들웨어 또는 권한 체크 로직을 추가합니다.

2. 웹 인터페이스 및 기능

- 웹 프레임워크:
 - Golang 기반의 프레임워크(예: Gin, Echo 등)를 사용하여 RESTful API 및 템플릿 기반 UI를 구현합니다.
- 메뉴 및 네비게이션:
 - 상단 좌측의 collapsed menu icon을 통해 권한 관리 메뉴에 접근할 수 있게 합니다.
- 백업 및 리비전 관리:
 - "Revision" 버튼 클릭 시, 지정된 경로에서 [기존파일명_날짜시간초.yaml] 형식으로 저장된 백업파일 리스트를 보여줍니다.
 - 리스트에서 특정 백업 파일을 선택하면 롤백 여부를 묻고, '예'를 선택하면 해당 백업 파일을 현재 docker-compose 파일로 복원 후 도커 시스템을 재시작합니다.



Part 3. 원격 관리도구 만들기

ChatGPT가 작성한 코드 테스트 및 Github commit

<https://github.com/ralfyang/docker-compose-webctl>

The screenshot shows the GitHub repository page for 'docker-compose-webctl' by user 'ralfyang'. The repository is public and has 5 stars and 0 forks. The main branch is selected. The repository description is 'Docker-compose web controller(web console) for the remote edit & restart the docker-compose'. The repository contains 39 commits. The file list on the left includes: src, templates, .gitignore, README.md, README_KR.md, dc_webconsole, dc_webconsole_arm, docker-controller.sh, go.mod, go.sum, main.go, and prompt.md. The 'About' section shows the repository's purpose. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section shows the repository is primarily Go (72.7%), with HTML (25.8%) and Shell (1.5%) also present.

File	Description	Commit Time
src	update for the sync	2 months ago
templates	update for the backup file & rollback f...	2 months ago
.gitignore	update for the backup file & rollback f...	2 months ago
README.md	Update README.md	4 days ago
README_KR.md	Update README_KR.md	4 days ago
dc_webconsole	update for bin command	2 months ago
dc_webconsole_arm	made a new execute bin file for macO...	2 months ago
docker-controller.sh	start from scratch!	2 months ago
go.mod	update for the backup file & rollback f...	2 months ago
go.sum	update for the backup file & rollback f...	2 months ago
main.go	update for docker-compose or docker...	2 months ago
prompt.md	Update prompt.md	28 minutes ago

The screenshot shows the 'main.go' file in the 'docker-compose-webctl' repository. The file is 873 lines long (744 loc) and 25.9 KB. The code is in Go and defines a 'User' struct and a 'main' function. The 'User' struct has fields 'Email', 'Password', and 'Role'. The 'main' function initializes a 'users' map and sets an 'accountFile'.

```
1 package main
2
3 import (
4     "bufio"
5     "errors"
6     "fmt"
7     "io"
8     "io/ioutil"
9     "log"
10    "net/http"
11    "os"
12    "os/exec"
13    "path/filepath"
14    "sort"
15    "strconv"
16    "strings"
17    "time"
18
19    "github.com/gin-contrib/sessions"
20    "github.com/gin-contrib/sessions/cookie"
21    "github.com/gin-gonic/gin"
22    "github.com/joho/godotenv"
23    "golang.org/x/crypto/bcrypt"
24 )
25
26 // =====
27 // 1. 전역 설정
28 // =====
29
30 type User struct {
31     Email    string
32     Password string // bcrypt 해시
33     Role     string // "admin" 또는 "none"
34 }
35
36 var users = make(map[string]*User)
37
38 const accountFile = ".account"
39
```


Part 3. 원격 관리도구 만들기

개발된 웹콘솔을 통해 원격지 개발/테스트 환경 설정 관리하기!

```
total 28
5784669 drwxr-xr-x@ 8 Ralf  staff   256  4  7 16:09 .
5539483 drwxr-xr-x@ 4 Ralf  staff   128  4  7 15:49 ..
5784678 -rw-r--r--@ 1 Ralf  staff    61  4  7 15:49 .env
5784918 drwxr-xr-x@ 3 Ralf  staff    96  4  7 15:51 backups
5784670 -rw-r--r--@ 1 Ralf  staff  2313  4  7 15:51 docker-compose.yml
5784671 drwxr-xr-x@ 3 Ralf  staff    96  4  7 15:49 grafana
5784676 drwxr-xr-x@ 3 Ralf  staff    96  4  7 15:49 nginx
5784679 drwxr-xr-x@ 3 Ralf  staff    96  4  7 15:49 prometheus
Ralf@Mac monitoring % cd ..
Ralf@Mac docker-compose-list % ll
total 0
5539483 drwxr-xr-x@ 4 Ralf  staff   128  4  7 15:49 .
5492349 drwxr-xr-x@ 19 Ralf  staff   608  4  7 16:05 ..
5612679 drwxr-xr-x@ 10 Ralf  staff   320  4  4 09:03 example
5784669 drwxr-xr-x@ 8 Ralf  staff   256  4  7 16:09 monitoring
Ralf@Mac docker-compose-list % cd ..
Ralf@Mac docker-compose-webctl % ll
total 67968
5492349 drwxr-xr-x@ 19 Ralf  staff     608  4  7 16:05 .
5491487 drwxr-xr-x@ 4 Ralf  staff     128  4  4 10:12 ..
5612602 -rw-r--r--@ 1 Ralf  staff      79  4  4 08:51 .account
5493754 -rw-r--r--@ 1 Ralf  staff      70  4  3 15:56 .env
5492350 drwxr-xr-x@ 14 Ralf  staff    448  4  7 15:47 .git
5492424 -rw-r--r--@ 1 Ralf  staff      77  4  3 15:52 .gitignore
5526977 drwxr-xr-x@ 7 Ralf  staff    224  4  3 16:16 .venv
5612364 -rw-r--r--@ 1 Ralf  staff   5392  4  4 08:49 README.md
5612365 -rw-r--r--@ 1 Ralf  staff   5349  4  4 08:49 README_KR.md
5539067 -rwxr-xr-x@ 1 Ralf  staff 17347842  4  3 16:17 dc_webconsole
5612564 -rw-r--r--@ 1 Ralf  staff   16786  4  7 15:52 dc_webconsole.log
5492428 -rwxr-xr-x@ 1 Ralf  staff 17347762  4  3 15:52 dc_webconsole_arm
5539483 drwxr-xr-x@ 4 Ralf  staff     128  4  7 15:49 docker-compose-list
5492429 -rwxr-xr-x@ 1 Ralf  staff     562  4  3 15:52 docker-controller.sh
5492430 -rw-r--r--@ 1 Ralf  staff   1588  4  3 16:17 go.mod
5492431 -rw-r--r--@ 1 Ralf  staff   8988  4  3 15:52 go.sum
5492432 -rw-r--r--@ 1 Ralf  staff  26552  4  3 15:52 main.go
5492433 drwxr-xr-x@ 3 Ralf  staff      96  4  3 15:52 src
5492435 drwxr-xr-x@ 6 Ralf  staff     192  4  3 15:52 templates
Ralf@Mac docker-compose-webctl % ./
```



Part 4. 마무리

AI를 활용한 오피서버빌리티 똑딱 만들기

Part 4. 마무리

완료된 업무에 관련한 매뉴얼 작성 요청

- README.md 작성 요청
- Notion 사용자 매뉴얼 작성 요청

좋습니다! 앞서 만든 개발 산출물에 대하여 github README.md 파일을 작성해주세요

1초 동안 Reasoned about README.md creation
아래는 GitHub README.md 예시입니다.
본 문서는 Go + Gin 기반의 docker-compose 웹 콘솔 프로젝트를 안내합니다.

Docker-Compose Web Console

이 프로젝트는 Go 언어와 Gin 프레임워크를 이용해,
docker-compose를 웹에서 간편하게 관리할 수 있는 웹 콘솔을 제공합니다.

주요 기능

- 1. 회원가입/로그인
 - 첫 가입자는 자동으로 어드민 권한 획득
 - 이후 가입자는 none 권한이며, 어드민이 승인(권한 변경)해야 관리자 기능 사용 가능
- 2. 도커 컴포즈 웹 콘솔
 - 디렉토리 생성: docker-compose-list 하위 디렉토리
 - 파일 생성: 각 디렉토리 내에 .yaml 등 설정 파일
 - 파일 편집: 웹 에디터 제공
 - 저장 & 리스타트: docker-compose -f [파일] down; sleep 2; up -d 로 컨테이너 재시작

지금 까지 내용을 종합적으로 정리하여 Notion에 기록하기 위한 매뉴얼을 작성해주세요

📌 Notion에 기록할 Grafana + Prometheus 모니터링 구축 매뉴얼

다음은 [redacted] 서버의 Docker 컨테이너를 모니터링하기 위한 Grafana + Prometheus 구축 매뉴얼입니다.
이 문서는 Notion에 정리하여 관리할 수 있도록 구성되었습니다.

🚀 Grafana + Prometheus 기반 Docker 모니터링 구축 매뉴얼

본 문서는 [redacted] 서버에서 실행 중인 Docker 컨테이너들을 Grafana + Prometheus를 활용해 모니터링하는 방법을 설명합니다.
Grafana는 [redacted]에서 운영됩니다.

1. 시스템 개요

◆ 서버 정보

역할	서버 정보
Grafana 서버	[redacted]
컨테이너 운영 서버	
Prometheus 실행 서버	

Part 4. 마무리

업무 생산성 비교





Thank you! :)

DevOps Korea Group: <http://devopskorea.com>

유튜바 킹모드: https://www.youtube.com/@ralf_NCI

KBTdot Inc.: <https://kbt.lol> - **AX(AI 전환) 관련 문의**