

포팅 메뉴얼 - coflo

1. 개발환경

1.1 Backend

1.2 Frontend

1.3 Database

1.4 Infra

1.5 Cooperation

2. 환경 변수 설정

2.1 Frontend: .env

2.2 Backend: application.yml

3. 빌드 및 배포 문서

3.1 소프트웨어 설치

3.2 Ngnix Proxy Manager

3.3 Spring

3.4 React

4. Database

4.1 Postgresql

4.2 Redis

4.3 Prometheus

5. Gitlab CI/CD

5.1 gitlab-runner

5.2 gitlab-cli

6. 기타 모니터링 설정

6.1 Grafana 설치

6.2 Grafana 실행

1. 개발환경

1.1 Backend

- Java==21
- Swagger==2.0.2
- Spring Boot==3.3.4
- Gradle==8.10.2

- lombok==1.18.34
- spring-boot-starter-data-jpa==3.3.4
- spring-boot-starter-web==3.3.4
- spring-boot-starter-oauth2-client==3.3.4
- spring-boot-starter-security==3.3.4
- spring-boot-starter-data-redis==3.3.4
- spring-boot-starter-graphql==3.3.4
- spring-boot-starter-validation==3.3.4
- spring-boot-starter-integration==3.3.4
- spring-boot-starter-batch==3.3.4
- spring-boot-starter-webflux==3.3.4
- spring-boot-starter-actuator==3.3.4
- jjwt-api==0.11.5
- jjwt-impl==0.11.5
- jjwt-jackson==0.11.5
- querydsl-jpa==5.0.0
- micrometer-registry-prometheus==1.13.4
- mockito-junit-jupiter==5.14.2
- mockito-core==5.14.2

1.2 Frontend

- axios==1.7.7
- react==18.3.1
- react-dom==18.3.1
- react-router-dom==6.24.1
- typescript==5.5.3
- vite==5.4.8

- lucide-react==0.453.0
- tailwind-merge==2.5.4
- @tanstack/react-query==5.59.13
- jotai==2.10.0
- msw==0.36.3
- jest==29.5.13

1.3 Database

- PostgreSQL==16.4
- Redis==7.4.1

1.4 Infra

- docker==27.3.1
- docker-compose==2.29.7
- nginx-proxy-manager==2.12.1
- nginx==openresty/1.25.3.2
- loki==3.0.0
- promtail==3.0.0
- prometheus==2.8.4

1.5 Cooperation

- Git
- Gitlab
- Jira
- MatterMost
- Figma
- Notion

2. 환경 변수 설정

2.1 Frontend: .env

- 환경변수 설정 위치

```
S11P31A210
└─ frontend
    └─ .env
```

- .env

```
VITE_API_BASE_URL=https://www.coflo.co.kr
VITE_REDIRECT_URL=https://www.coflo.co.kr
```

- 로컬호스트에서 https로 정상 가동을 위해 mkcert SSL 인증 필요

2.2 Backend: application.yml

- 환경변수 설정 위치

```
S11P31A210
└─ backend
    └─ src
        └─ main
            └─ resources
                ├── application.yml
                ├── application-local.yml
                ├── application-prod.yml
                └─ application-oauth.yml
```

- application.yml

```
spring:
  application:
```

```

    name: coflo

profiles:
  active: ${SPRING_PROFILES_ACTIVE}
  include: oauth

output:
  ansi:
    enabled: always

openai:
  api:
    key: ${OPENAI_API_KEY}
    organization: ${OPENAI_ORGANIZATION}
    project: ${OPENAI_PROJECT}

management:
  endpoints:
    web:
      exposure:
        include: health,metrics,prometheus
        exclude: env,beans,heapdump,threaddump,configprops,quarts
      prometheus:
        metrics:
          export:
            enabled: true
    web:
      server:
        auto-time-requests: true

logging:
  level:
    com:
      reviewping:
        coflo: debug
    web: debug
  config: classpath:logging/logback-spring.xml

```

```
crypto-key: ${CRYPTO_KEY}
```

```
domain-webhook-url: ${COFLO_WEBHOOK_URL}
```

- application-local.yml

```
spring:
  config:
    activate:
      on-profile: local

  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/coflo
    username: postgres
    password: 1234

  jpa:
    show-sql: true
    hibernate:
      ddl-auto: create-drop
    properties:
      hibernate:
        dialect: org.hibernate.dialect.PostgreSQLDialect
        format_sql: true
    defer-datasource-initialization: true

  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}

  sql:
    init:
      mode: always
```

```
mattermost-logger:
  base-url: ${LOCAL_ERROR_WEBHOOK_URL}
```

- application-prod.yml

```
spring:
  config:
    activate:
      on-profile: prod

  datasource:
    driver-class-name: org.postgresql.Driver
    url: ${DATASOURCE_URL}
    username: ${DATASOURCE_USERNAME}
    password: ${DATASOURCE_PASSWORD}

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
    show-sql: true
    defer-datasource-initialization: true

  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}

#  sql:
#    init:
#      mode: always

mattermost-logger:
  base-url: ${PROD_ERROR_WEBHOOK_URL}
```

- application-scret.yml

```

spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret: ${KAKAO_CLIENT_SECRET}
            client-authentication-method: client_secret_
post
            redirect-uri: ${KAKAO_REDIRECT_URL}
            authorization-grant-type: authorization_code
            client-name: kakao
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET}
            client-authentication-method: client_secret_
basic
            redirect-uri: ${GOOGLE_REDIRECT_URL}
            authorization-grant-type: authorization_code
            client-name: google
            scope: email
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/o
auth/authorize?prompt=login
            token-uri: https://kauth.kakao.com/oauth/tok
en
            user-info-uri: https://kapi.kakao.com/v2/use
r/me
            user-name-attribute: id
          google:
            authorization-uri: https://accounts.google.c
om/o/oauth2/auth
            token-uri: https://oauth2.googleapis.com/tok
en

```



```
        user-info-uri: https://www.googleapis.com/oauth2/v3/userinfo
        user-name-attribute: sub

jwt:
  secretKey: ${JWT_SECRET_KEY}
  access:
    expiration: ${ACCESS_EXPIRATION}
    name: ${ACCESS_NAME}

  refresh:
    expiration: ${REFRESH_EXPIRATION}
    name: ${REFRESH_NAME}
```

3. 빌드 및 배포 문서

3.1 소프트웨어 설치

3.1.1 Docker 설치

```
# 1. 도커 apt 리포지토리 설정
# 도커 공식 GPG key 추가
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Apt 소스에 도커 리포지토리 추가
echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ub
```

```
untu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable"
| \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# 2. 도커 최신버전 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

3.1.2 docker-compose 설치

```
# 1. docker-compose 설치
$ sudo curl -SL https://github.com/docker/compose/releases/
download/v2.29.2/docker-compose-linux-x86_64 -o /usr/local/
bin/docker-compose

# 2. 실행 권한 주기
$ sudo chmod +x /usr/local/bin/docker-compose
```

3.1.3 Java 설치

```
# 1. java 설치
$ sudo apt-get install openjdk-21-jdk
$ java -version
$ which java
$ readlink -f /usr/bin/java

# 2. 환경변수 추가
$ sudo vi /etc/profile

# 아래 내용 추가
export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.ja
r
```

```
# 3. 환경변수 적용
$ source /etc/profile
```

3.2 Ngnix Proxy Manager

3.2.1 compose.yml


```
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '8081:81'
      - '443:443'
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
      - /home/ubuntu/coflo/coflo-fe/dist:/mnt/NginxProxyManager/websites
```

3.2.2 Nginx Proxy Manager 설정



<http://호스트명:npm포트> 로 접속, coflo는 아래 경로 사용했음

<http://www.coflo.co.kr:8081> <http://k11a210.p.ssafy.io:8081>



NGINX
PROXY MANAGER

v2.12.1

Login to your account

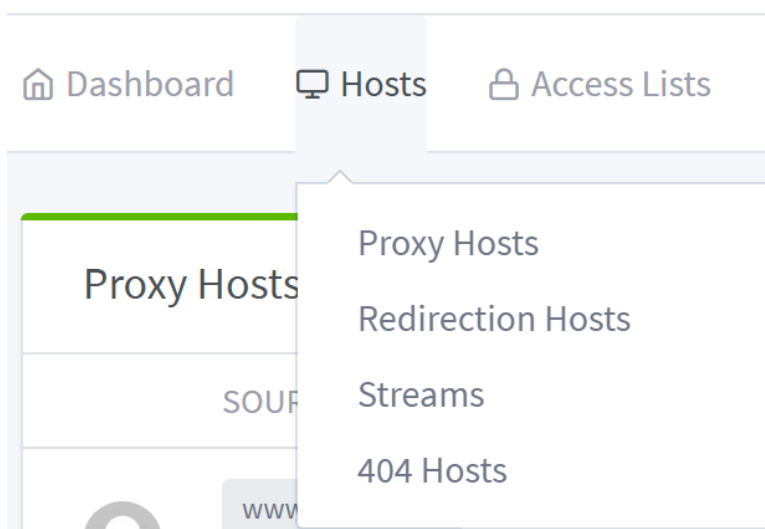
Email address

Password

Sign in

디폴트 ID, PASSWORD
Email: admin@example.com
Password: changeme

- 상단 바 Hosts → Proxy Hosts → Add Proxy Host



Edit Proxy Host

Details
Custom locations
SSL
Advanced

Domain Names *

www.coflo.co.kr

Scheme *
Forward Hostname / IP *
Forward Port *

http

k11a210.p.ssafy.io

80

☐ Cache Assets
☒ Block Common Exploits

☐ Websockets Support

Access List

Publicly Accessible

Cancel
Save

- 도메인 이름과 포워딩 할 호스트 or IP와 포트 기입

New Proxy Host

Details
Custom locations
SSL
Advanced

SSL Certificate

Request a new SSL Certificate

☒ Force SSL
☒ HTTP/2 Support

☒ HSTS Enabled
☐ HSTS Subdomains

☐ Use a DNS Challenge

Email Address for Let's Encrypt *

fview1020@gmail.com

☒ I Agree to the Let's Encrypt Terms of Service *

Cancel
Save

- SSL에서 인증서 옵션 설정, 이메일 입력 하고 동의 체크하면 letsencrypt 자동 생성됨
- Advanced에 아래와 같이 location 등 추가 설정 정보 입력

```

client_max_body_size 0;
root /mnt/NginxProxyManager/websites;
index index.html;

location / {
    try_files $uri /index.html;
}

location /api/ {
    proxy_pass http://k11a210.p.ssafy.io:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

location ~ ^/(swagger|webjars|configuration|swagger-resources) {
    proxy_pass http://k11a210.p.ssafy.io:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

```

3.2.3 생성된 conf 파일

```

# -----
# www.coflo.co.kr
# -----

map $scheme $hsts_header {
    https    "max-age=63072000; preload";
}

```

```

server {
    set $forward_scheme http;
    set $server            "k11a210.p.ssafy.io";
    set $port              80;

    listen 80;
listen [::]:80;

listen 443 ssl http2;
listen [::]:443 ssl http2;

    server_name www.coflo.co.kr;

    # Let's Encrypt SSL
    include conf.d/include/letsencrypt-acme-challenge.conf;
    include conf.d/include/ssl-ciphers.conf;
    ssl_certificate /etc/letsencrypt/live/npm-3/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/npm-3/privkey.pem

    # Block Exploits
    include conf.d/include/block-exploits.conf;

    # HSTS (ngx_http_headers_module is required) (63072000 seco
    add_header Strict-Transport-Security $hsts_header always;

    # Force SSL
    include conf.d/include/force-ssl.conf;

    access_log /data/logs/proxy-host-1_access.log proxy;
    error_log /data/logs/proxy-host-1_error.log warn;

    client_max_body_size 0;
    root /mnt/NginxProxyManager/websites;
    index index.html;

    location / {
        try_files $uri /index.html;
    }

```

```

location /api/ {
    proxy_pass http://k11a210.p.ssafy.io:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

location ~ ^/(swagger|webjars|configuration|swagger-resources) {
    proxy_pass http://k11a210.p.ssafy.io:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

# Custom
include /data/nginx/custom/server_proxy[.]conf;
}

```

3.3 Spring

3.3.1 Dockerfile

- spring-docker

```

FROM openjdk:21-jdk

# JAR 파일을 컨테이너에 복사
COPY coflo-0.0.1-SNAPSHOT.jar app.jar

# 환경 변수 설정 (Spring Profile 설정)
ENV SPRING_PROFILES_ACTIVE=prod
# JWT
ENV JWT_SECRET_KEY=DG3K2NG9lK3T2FLfn0283H01NFLAY9FGJ23UM9Rv92
ENV ACCESS_EXPIRATION=600000
ENV ACCESS_NAME=Authorization

```



```

ENV REFRESH_EXPIRATION=6000000
ENV REFRESH_NAME=Authorization-Refresh
# DB 암호화 키
ENV CRYPTO_KEY=G09m5LqyB2XWSxbcAQCl/Otth/tGmwypWPulNoMt8i9UMo
# OAuth
ENV KAKAO_CLIENT_ID=f5409614dc9b60e690dbdd54b35dd1d6
ENV KAKAO_CLIENT_SECRET=fkgNSSla
ENV KAKAO_REDIRECT_URL=https://www.coflo.co.kr/api/login/oauth
ENV GOOGLE_CLIENT_ID=869174603937-drss2h8jlugffrn92005movb2b
ENV GOOGLE_CLIENT_SECRET=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
ENV GOOGLE_REDIRECT_URL=https://www.coflo.co.kr/api/login/oauth
ENV REDIRECT_REGIST_URL=https://www.coflo.co.kr/regist
ENV REDIRECT_MAIN_URL=https://www.coflo.co.kr/main
# OPEN AI
ENV OPENAI_API_KEY=sk-svcacct-zYxu-jLD_9fX6kbEHk0aNaQTXN3kTmV
ENV OPENAI_ORGANIZATION=org-XCRpfniwHKW9je7MqaXoQRFP
ENV OPENAI_PROJECT=proj_M9DlIVLNGDaSGGZ7Bm30lWuS
# postgresQL
ENV DATASOURCE_URL=jdbc:postgresql://172.26.7.112:5432/coflo
ENV DATASOURCE_USERNAME=reviewping
ENV DATASOURCE_PASSWORD=coflo1234!
# REDIS
ENV REDIS_HOST=redis
ENV REDIS_PORT=6379
ENV REDIS_PASSWORD=coflo1234!
# WEBHOOK
ENV COFLO_WEBHOOK_URL=https://www.coflo.co.kr/webhook
# LOGGER
ENV LOCAL_ERROR_WEBHOOK_URL=https://meeting.ssafy.com/hooks/h
ENV PROD_ERROR_WEBHOOK_URL=https://meeting.ssafy.com/hooks/mp
ENTRYPOINT ["java", "-jar", "app.jar"]

```

- spring-docker-ai

```

FROM openjdk:21-jdk

# JAR 파일을 컨테이너에 복사
COPY coflo-0.0.1-SNAPSHOT.jar app.jar

```

```

## openai
ENV OPENAI_API_KEY=sk-svcacct-zYxu-jLD_9fX6kbEHk0aNaQTXN3kTmV
ENV OPENAI_ORGANIZATION=org-XCRpfniwHKW9je7MqaXoQRFp
ENV OPENAI_PROJECT=proj_M9DlIVLNGDaSGGZ7Bm30lWuS

ENV REDIS_HOST=redis
ENV REDIS_PORT=6379
ENV REDIS_PASSWORD=coflo1234!

ENV POSTGRES_JDBC_URL=jdbc:postgresql://172.26.7.112:5432/cof
ENV POSTGRES_USERNAME=reviewping
ENV POSTGRES_PASSWORD=coflo1234!

ENV GIT_CLONE_DIRECTORY=/tmp/coflo/git-projects/
ENV MATTERMOST_WEBHOOK_URL=https://meeting.ssafy.com/hooks/dp

ENV SERVICE_URL=https://www.coflo.co.kr

ENTRYPOINT ["java", "-jar", "app.jar", "--spring.profiles.act

```

3.3.2 spring.sh 스크립트

- spring-docker

```

#!/bin/bash

IMAGE_NAME="coflo-spring"
IMAGE_TAG="latest"
CONTAINER_NAME="coflo-spring-container"
LOG_DIR=~/.coflo_log
WEBHOOK_URL="https://meeting.ssafy.com/hooks/e3jbm544pjboudkr
now_port=8080

TIMEOUT=20
SECONDS=0

# Docker 이미지 빌드

```

```

docker build -t ${IMAGE_NAME}:${IMAGE_TAG} ~/coflo/spring-doc

if [ $? -eq 0 ]; then
    curl -i -X POST -H 'Content-Type: application/json' -d '{
else
    curl -i -X POST -H 'Content-Type: application/json' -d '{
    exit 1
fi

# 기존에 실행 중인 컨테이너가 있으면 중지 및 제거
existing_container=$(docker ps -aq -f name=${CONTAINER_NAME})

if [ -n "$existing_container" ]; then
    echo "Stopping and removing existing container..."
    docker stop ${CONTAINER_NAME} || true # 이미 중지된 경우 무
    docker rm -f ${CONTAINER_NAME} # 강제 삭제
fi

# 도커 컨테이너 실행
docker run -d --name ${CONTAINER_NAME} --network coflo-networ

# 사용하지 않는 이미지를 제거
dangling_images=$(docker images -f "dangling=true" -q)
if [ -n "$dangling_images" ]; then
    docker rmi $dangling_images
fi

sleep 5

# webhook
while true; do
    response=$(curl -s -o /dev/null -w "%{http_code}" http://

    if [ "$response" -eq 200 ]; then
        curl -i -X POST -H 'Content-Type: application/json' -d {
        break
    fi

```

```

# 시간 초과 확인
if [ $SECONDS -ge $TIMEOUT ]; then
    curl -i -X POST -H 'Content-Type: application/json' -d '{
    exit 1
fi

sleep 1 # 1초 대기 후 다시 확인
echo "다시 확인중"
done

```

- spring-docker-ai

```

#!/bin/bash

IMAGE_NAME="coflo-spring-ai"
IMAGE_TAG="latest"
CONTAINER_NAME="coflo-spring-ai-container"
LOG_DIR=~/.coflo_log_ai
now_port=8083

TIMEOUT=20
SECONDS=0

# Docker 이미지 빌드
docker build -t ${IMAGE_NAME}:${IMAGE_TAG} ~/coflo/spring-docker-ai

if [ $? -eq 0 ]; then
    curl -i -X POST -H 'Content-Type: application/json' -d '{
else
    curl -i -X POST -H 'Content-Type: application/json' -d '{
    exit 1
fi

# 기존에 실행 중인 컨테이너가 있으면 중지 및 제거
existing_container=$(docker ps -aq -f name=${CONTAINER_NAME})

if [ -n "$existing_container" ]; then
    echo "Stopping and removing existing container..."

```

```

        docker stop ${CONTAINER_NAME} || true # 이미 중지된 경우 무시
        docker rm -f ${CONTAINER_NAME} # 강제 삭제
    fi

# 도커 컨테이너 실행
docker run -d --name ${CONTAINER_NAME} --network coflo-networ

# 사용하지 않는 이미지를 제거
dangling_images=$(docker images -f "dangling=true" -q)
if [ -n "$dangling_images" ]; then
    docker rmi $dangling_images
fi

sleep 5

# 실행
chmod +x ./spring.sh
./spring.sh

```

3.4 React

3.4.1 react.sh

```

#!/bin/bash

WEBHOOK_URL="https://meeting.ssafy.com/hooks/nne3i8r1x3fcfq
s8ta6ddr34th"
BUILD_FILE=~/.coflo/coflo-fe/dist/index.html
TIMEOUT=60

if [ -f "$BUILD_FILE" ] && [ $(( $(date +%s) - $(stat -c %Y
"$BUILD_FILE"))) -lt $TIMEOUT ]; then
    curl -i -X POST -H 'Content-Type: application/json' \
    -d '{"text": "***[FE]** React 빌드 완료"}' $WEBHOOK_URL
else

```

```
curl -i -X POST -H 'Content-Type: application/json' \
-d '{"text": "***[FE]** React 빌드 실패: 빌드 결과를 찾을 수
없거나 최신 빌드가 아닙니다."}' $WEBHOOK_URL
exit 1
fi
```

4. Database

4.1 Postgresql

4.1.1 postgres 도커 이미지 받기

```
docker pull postgres:16.4
```

4.1.2 postgres 실행

```
$ docker run -d -p 5432:5432 --name postgres \
-e POSTGRES_PASSWORD=coflo1234! \
-e TZ=Asia/Seoul \
-v /home/ubuntu/coflo/pgdata:/var/lib/postgresql/data \
-d postgres
```

4.1.3 PostgreSQL 데이터베이스 및 유저 생성

```
docker exec -it [CONTAINER ID] bash # postgres 컨테이너 접속
psql -U postgres # postgresql 접속
create database coflo;
CREATE USER reviewping WITH PASSWORD 'coflo1234!' SUPERUSE
R;
GRANT ALL PRIVILEGES ON DATABASE coflo TO reviewping;
```

4.1.4 pgvector 설치

```
sudo apt install -y postgresql-common
sudo /usr/share/postgresql-common/pgdg/apt.postgresql.org.sh
sudo apt update
sudo apt install postgresql-16-pgvector
```

4.2 Redis

4.2.1 Redis 설치 및 실행

```
$ docker pull redis:latest
$ docker run --name redis --restart=always --network coflo-
network -p 6379:6379 \
    -v /home/ubuntu/coflo/redis/redis.conf:/usr/local/etc/r
edis/redis.conf \
    -d redis redis-server /usr/local/etc/redis/redis.conf -
-requirepass 'coflo1234!'
```

4.3 Prometheus

4.3.1 prometheus.yml

```
global:
  scrape_interval:      10s
  evaluation_interval: 5s
scrape_configs:
  - job_name: 'prometheus'
    metrics_path: /metrics
    scheme: http
    static_configs:
      - targets: ['172.26.7.112:9090']

  - job_name: 'node'
    metrics_path: /metrics
```

```

    scheme: http
    static_configs:
      - targets: ['172.26.7.112:9100']

- job_name: 'spring-actuator'
  metrics_path: '/actuator/prometheus'
  scheme: http
  static_configs:
    - targets: ['172.26.7.112']

```

4.3.2 prometheus 실행

```

docker run -d --name prometheus -h prometheus -p 9090:9090 \
  -v /home/ubuntu/coflo/prometheus/prometheus.yml:/etc/promet
  prom/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --web.enable-lifecycle

```

5. Gitlab CI/CD

5.1 gitlab-runner

5.1.1 gitlab-runner 설치

```

# Download the binary for your system
sudo curl -L --output /usr/local/bin/gitlab-runner https://
gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gi
tlab-runner-linux-amd64

# Give it permission to execute
sudo chmod +x /usr/local/bin/gitlab-runner

gitlab-runner --version

```



```
gitlab-runner register --url https://lab.ssafy.com --token glrt-M2kyWmwFZcz_roEGFfc-  
  
sudo gitlab-runner run
```

5.1.2 gitlab-runner 등록

```
gitlab-runner register --url https://lab.ssafy.com --token glrt-M2kyWmwFZcz_roEGFfc-  
  
gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner #gitlab runner 서비스로 설치  
  
sudo gitlab-runner run # 포그라운드 실행  
sudo gitlab-runner start # 백그라운드 실행  
  
# 백그라운드 실행  
sudo systemctl start gitlab-runner  
sudo systemctl enable gitlab-runner  
sudo systemctl status gitlab-runner  
sudo systemctl stop gitlab-runner
```

5.1.3 .gitlab-runner/config.toml

```
concurrent = 1  
check_interval = 0  
shutdown_timeout = 0  
  
[session_server]  
  session_timeout = 1800  
  listen_address = "0.0.0.0:8093"  
  advertise_address = "43.203.201.104:8093"
```

```

[[runners]]
  name = "ip-172-26-7-112"
  url = "https://lab.ssafy.com"
  id = 844
  token = "glrt-moh4ndXu-4CXDsKSUZYb"
  token_obtained_at = 2024-10-29T07:43:29Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]

```

5.2 gitlab-cli

5.2.1 .gitlab-cli.yml

```

stages:
  - build

build_BE:
  stage: build
  image: gradle:8.8-jdk21
  before_script:
    - echo "[INFO] YML Settings"
    - chmod +x backend/gradlew
  script:
    - cd backend
    - ./gradlew clean build -x test -Dspring.profiles.active=
    - cp build/libs/coflo-0.0.1-SNAPSHOT.jar ~/coflo/spring-d
    - sh ~/coflo/spring-docker/spring.sh
  rules:
    - if: '$CI_COMMIT_BRANCH == "be/dev"'
  tags:

```

```

    - prod

build_FE:
  stage: build
  image: node
  before_script:
    - echo "[INFO] YML Settings"
    - cd frontend/coflo
    - printenv | grep 'VITE_' > .env
  script:
    - yarn install
    - yarn build
    - sudo cp -R dist ~/coflo/coflo-fe
    - sh ~/coflo/coflo-fe/react.sh
  rules:
    - if: '$CI_COMMIT_BRANCH == "fe/dev"'
  tags:
    - prod

build_AI:
  stage: build
  image: gradle:8.8-jdk21
  before_script:
    - echo "[INFO] YML Settings"
    - chmod +x ai/gradlew
  script:
    - cd ai
    - ./gradlew clean build -x test -Dspring.profiles.active=
    - cp build/libs/coflo-0.0.1-SNAPSHOT.jar ~/coflo/spring-d
    - sh ~/coflo/spring-docker-ai/spring.sh
  rules:
    - if: '$CI_COMMIT_BRANCH == "ai/dev"'
  tags:
    - prod

```

6. 기타 모니터링 설정

6.1 Grafana 설치

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-
echo "deb https://packages.grafana.com/oss/deb stable main" |

sudo apt-get update
sudo apt-get install -y grafana
```

6.2 Grafana 실행

```
sudo systemctl edit grafana-server.service
```

위 명령어 입력 하면 빈 파일 나오는데 다음 내용 입력 후 저장

```
[Service]
# Give the CAP_NET_BIND_SERVICE capability
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
AmbientCapabilities=CAP_NET_BIND_SERVICE

# A private user cannot have process capabilities on the host
# namespace and thus CAP_NET_BIND_SERVICE has no effect.
PrivateUsers=false
```

실행 명령어

```
sudo service grafana-server start
sudo service grafana-server status

sudo update-rc.d grafana-server defaults
```