🔷

Porting manuals

프로젝트 사용 도구

Common

- Issue Tracking : Jira
- Configuration Management : Gitlab
- Communication : Notion, Mattermost
- Degine : Figma

프로젝트 환경

Server

- Ubuntu 20.0.4

Database

- MySQL : 8.0.29
- Redis : 7.0.4

Frontend [React]

- React 18.2.0
- Typescript 4.8.2
- Recoli 0.7.5
- Axios 0.27.2
- Phaser 3.55.2
- Tailwindcss 3.1.8
- Node 16.17.0

Backend [Django]

- Python 3.9.13
- Django 3.2.12
- Gunicorn 20.1.0
- Numpy 1.21.6
- Pandas 1.3.5
- Tensorflow 2.10.0
- Scikit-learn 1.0.2

Backend [Spring]

- Spring 2.7.3
- Java 1.8
- Spring cloud 2021.0.3
- Spring data JPA
- Eureka - Netflix
- Config server
- sleuth
- zipkin 2.2.3
- prometheus

Nginx

Nignx 설치

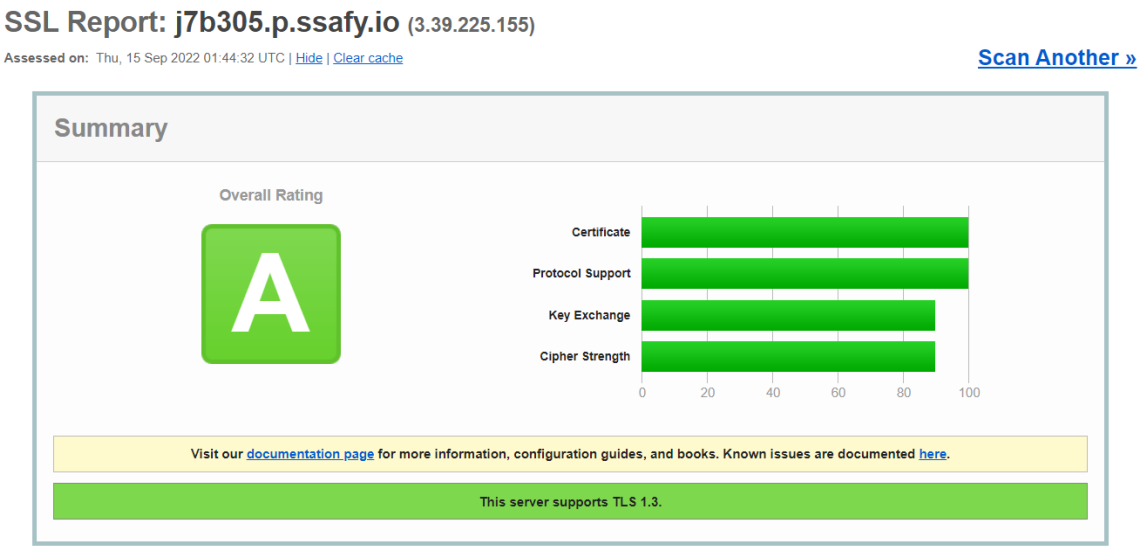
```
sudo apt install nginx
```

Certbot 설치

```
sudo apt install certbot
```

Certbot SSL 설정

```
sudo certbot --nginx -d j7b305.p.ssafy.io
```



https://www.ssllabs.com/ssltest/ - SSL 적용 확인 및 평가

Nginx 설정

path : /etc/nginx/sites-available/default

```
server {
    if ($host = j7b305.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        listen [::]:80;

        server_name i7b307.p.ssafy.io;

        return 301 https://j7b305.p.ssafy.io$request_uri;


}

server {
        listen          443 ssl;
        listen          [::]:443;

        server_name     j7b305.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/j7b305.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/j7b305.p.ssafy.io/privkey.pem; # managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


        location / {
#                root /home/ubuntu/dist;
#                root /var/www/html;
#               index index.nginx-debian.html;
#
                try_files $uri $uri/ /index.html;

# 도커이미지화 했을 경우
                proxy_pass http://localhost:8080;
                proxy_redirect off;
                charset utf-8;
```

```
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-NginX-Proxy true;
        }

        location /api {
                proxy_pass http://localhost:8080;
                proxy_redirect off;
                charset utf-8;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-NginX-Proxy true;
        }


}
```

Database

MySQL image pull

```
docker pull mysql:8.0.22
# 8.0.22: Pulling from library/mysql
# Digest: sha256:8c17271df53ee3b843d6e16d46cff13f22c9c04d6982eb15a9a47bd5c9ac7e2d
# Status: Downloaded newer image for mysql:8.0.22
# docker.io/library/mysql:8.0.22
```

볼륨 폴더 생성

```
sudo mkdir /opt/lib/mysql
```

Mysql Container 실행

```
docker run --name mysql -e MYSQL_ROOT_PASSWORD=root -v /opt/lib/mysql:/var/lib/mysql -d -p 3306:3306 mysql:8.0.22
```

Docker Container 접속

```
docker exec -it  [컨테이너 명] /bin/bash
```

MySQL 유저 생성 및 권한 부여

```
# mysql 접속
mysql -u root -p

# root 계정 비밀번호 변경
alter user 'root'@'localhost' identified with mysql_native_password by 'new password';
flush privileges;

# user 생성 및 권한 부여
create user '[username]'@'%' identified by '[password]';
grant all privileges on *.* to '[username]'@'%' with grant option;
flush privileges;
```

Redis image pull

```
docker image pull redis
```

Redis와 Redis-cli 연결을 위한 Redis net 생성

```
docker network create redis-net
# 생성 확인
docker network ls
```

Redis Container Run

```
docker run --name redis -p 6379:6379 --network redis-net -v /etc/ubuntu/redisDir -d redis:latest redis-server --appendonly yes

docker run --network redis-net -v /etc/redisDir:/data --name redis -d -p 6379:6379 redis redis-server /data/redis.conf
```

Redis-cli 접속

```
docker run -it --network redis-net --rm redis redis-cli -h redis
```

Message Queue

RabbitMQ image pull

```
docker pull rabbitmq:management
```

RabbitMQ image Run

```
docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 --restart=unless-stopped rabbitmq:management
```

- 비공인 IP 주소를 사용하여 VM 간에 RabbitMQ와 연동하는 경우에는, <비공인 IP>:5672 주소를 통해 접속할 수 있습니다.
- RabbitMQ의 Management UI Plugin은 Web 대시보드를 통해 관리할 수 있는 도구를 제공합니다. 이를 위해서는 ACG에 15672 포트가 추가되어 있어야 하며 공인 IP 주소를 할당받아 서버에 부여해야 합니다.
- RabbitMQ의 Management UI Plugin의 주소는 http://<공인IP주소>:15672/입니다. 접속되지 않는다면 ACG가 추가되어 있는지 확인하거나, 터미널에서 Management UI가 실행되어 있는지를 확인합니다.

Docker

Docker 설치

```
sudo apt-get update
curl -fsSL https://get.docker.com/ | sudo sh
```

Docker 권한 설정

```
sudo usermod -aG docker $USER
sudo service docker restart

sudo su
sudo su ubuntu

docker ps
```

Dockerfile

- React

```
# build
FROM node:16.17.0 as builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build


FROM nginx:stable-alpine
RUN rm -rf /etc/nginx/conf.d/defalut.conf
COPY --from=builder /app/nginx/default.conf /etc/nginx/conf.d/default.conf

RUN rm -rf /usr/share/nginx/html/*
COPY --from=builder /app/build /usr/share/nginx/html

EXPOSE 80
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

- Django

```
FROM python:3.9.13
WORKDIR /var/jenkins_home/workspace/Backend/jeonwoochi_django
COPY requirements.txt ./

RUN pip install --upgrade pip
RUN pip install -r requirements.txt
COPY . .
EXPOSE 8989
CMD ["python", "manage.py", "runserver", "--noreload", "0.0.0.0:8989"]
```

- Spring

```
FROM openjdk:8 AS builder
VOLUME /tmp

FROM openjdk:8
COPY build/libs/config-service-0.0.1-SNAPSHOT.jar ConfigServer.jar
EXPOSE 8888
ENTRYPOINT ["java","-jar","ConfigServer.jar"]

-----------------------------------------------------------------------------------------------------------------------------

FROM openjdk:8 AS builder
VOLUME /tmp

FROM openjdk:8
COPY build/libs/discovery-service-0.0.1-SNAPSHOT.jar DiscoveryServer.jar
EXPOSE 8761
ENTRYPOINT ["java","-jar","DiscoveryServer.jar"]

-----------------------------------------------------------------------------------------------------------------------------

FROM openjdk:8 AS builder
VOLUME /tmp

FROM openjdk:8
COPY build/libs/gateway-service-1.0-SNAPSHOT.jar GatewayServer.jar
EXPOSE 8000
ENTRYPOINT ["java","-jar","GatewayServer.jar"]

-----------------------------------------------------------------------------------------------------------------------------

FROM openjdk:8 AS builder
VOLUME /tmp

FROM openjdk:8
COPY build/libs/festival-service-0.0.1-SNAPSHOT.jar FestivalServer.jar
ENTRYPOINT ["java","-jar","FestivalServer.jar"]

-----------------------------------------------------------------------------------------------------------------------------

FROM openjdk:8 AS builder
```

```
VOLUME /tmp

FROM openjdk:8
COPY build/libs/main-service-0.0.1-SNAPSHOT.jar MainServer.jar
ENTRYPOINT ["java","-jar","MainServer.jar"]

------------------------------------------------------------------------------------------------------------------------

FROM openjdk:8 AS builder
VOLUME /tmp

FROM openjdk:8
COPY build/libs/user-service-0.0.1-SNAPSHOT.jar UserServer.jar
ENTRYPOINT ["java","-jar","UserServer.jar"]
```

Jenkins

Docker In Docker 방식

Jenkins 볼륨 폴더 생성

```
sudo mkdir /home/ubuntu/jenkinsDir
```

Jenkins 실행

```
[-d : 백그라운드 실행 ]
[-p : 컨테이너와 호스트 PC간 연결을 위한 포트 지정 ]
[-v : 이미지의 /var/jenkins_home 디렉토리를 호스트 PC내에 마운트 - Jenkins 설치 시 ssh 키값 생성, 저장소 참조 등을 용이하게 하기 위함 ]]
docker run --name jenkins -d -p 9090:8080 -p 50000:50000 -v /home/ubuntu/jenkinsDir:/var/jenkins_home -v /var/run/docker.sock:/var/run
```

Jenkins에 도커 설치

```
docker exec -it "[Jenkins Container Name]" /bin/bash

sudo apt-get update
curl -fsSL https://get.docker.com/ | sh
```

Plug In

- 기본 권장 설치
- 블루오션
- 깃랩
- 메러모스트 노티피케이션

Jenkins Pipeline

- Frontend

```
// FrontJenkinsFile
pipeline{
    agent any
    environment {
        FRONT_CONTAINER_NAME="jeonwoochi_react_container"
        FRONT_NAME = "jeonwoochi_react_image"
    }
    stages {
        stage('Build') {
            steps {
                script{
                    try{
```

```
                        sh "docker build -t ${FRONT_NAME} ./front/."
                        mattermostSend (
                            color: "good",
                            message: "Front Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }catch(e){
                        mattermostSend (
                            color: "danger",
                            message: "Front Build FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }
                }
            }
        }
        stage('Clean'){
            steps{
                script {
                    try{
                        sh "docker stop ${FRONT_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${FRONT_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }
                    mattermostSend (
                        color: "good",
                        message: "Front Clean SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                    )
                }
            }
        }
        stage('Deploy'){
            steps {
                script{
                    try{
                        sh "docker run -d --name=${FRONT_CONTAINER_NAME} -p 8080:80 ${FRONT_NAME}"

                        sh "docker image prune -a --force"
                        mattermostSend (
                            color: "good",
                            message: "Front Deploy SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }catch(e){
                        mattermostSend (
                            color: "danger",
                            message: "Front Deploy FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }
                }
            }
        }
    }
}
```

- Backend

```
//BackJenkinsFile
pipeline{
    agent any
    environment {
       DISCOVERY_CONTAINER_NAME="discovery-service-container"
       DISCOVERY_NAME = "discovery-service-image"

       CONFIG_CONTAINER_NAME="config-service-container"
       CONFIG_NAME = "config-service-image"

       GATEWAY_CONTAINER_NAME="gateway-service-container"
       GATEWAY_NAME = "gateway-service-image"

       USER_CONTAINER_NAME="user-service-container"
       USER_NAME = "user-service-image"

       FESTIVAL_CONTAINER_NAME="festival-service-container"
       FESTIVAL_NAME = "festival-service-image"

       MAIN_CONTAINER_NAME="main-service-container"
       MAIN_NAME = "main-service-image"

       RECOMM_CONTAINER_NAME="recomm-service-container"
       RECOMM_NAME = "recomm-service-image"

    }
    stages {
        stage('Checkout') {
```

```
            steps{
                // Get some code from a Git repository
                checkout scm
                mattermostSend (
                    color: "good",
                    message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                )
            }
        }
        stage('Container Clean'){
            steps{
                script {
                    try{
                        sh "docker stop ${DISCOVERY_CONTAINER_NAME}"
                        sh "docker stop ${CONFIG_CONTAINER_NAME}"
                        sh "docker stop ${GATEWAY_CONTAINER_NAME}"

                        sh "docker stop ${USER_CONTAINER_NAME}"
                        sh "docker stop ${FESTIVAL_CONTAINER_NAME}"
                        sh "docker stop ${MAIN_CONTAINER_NAME}"
                        sh "docker stop ${RECOMM_CONTAINER_NAME}"
                        //sh "docker stop ${LOG_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${DISCOVERY_CONTAINER_NAME}"
                        sh "docker rm ${CONFIG_CONTAINER_NAME}"
                        sh "docker rm ${GATEWAY_CONTAINER_NAME}"

                        sh "docker rm ${USER_CONTAINER_NAME}"
                        sh "docker rm ${FESTIVAL_CONTAINER_NAME}"
                        sh "docker rm ${MAIN_CONTAINER_NAME}"
                        sh "docker rm ${RECOMM_CONTAINER_NAME}"
                        //sh "docker rm ${LOG_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }finally{
                        sh "docker image prune -a --force"
                        mattermostSend (
                            color: "good",
                            message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }
                }
            }
        }
    stage('Build Gradle') {
            steps {
                script{
                    try{
                        sh 'cd back/jeonwoochi_spring;chmod +x gradlew;./gradlew bootJar'
                        sh 'exit 0'
                        mattermostSend (
                            color: "good",
                            message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }catch(e){
                        mattermostSend (
                            color: "danger",
                            message: "Back Build FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }
                }
            }
        }
        stage('Docker Image Build') {
            steps {
                script{
                    try{
                                        // spring-cloud 빌드
                        sh "docker build -t ${DISCOVERY_NAME} ./back/jeonwoochi_spring/spring-cloud/discovery-service/."
                        sh "docker build -t ${CONFIG_NAME} ./back/jeonwoochi_spring/spring-cloud/config-service/."
                        sh "docker build -t ${GATEWAY_NAME} ./back/jeonwoochi_spring/spring-cloud/gateway-service/."

                                        // Spring micro-service 빌드
                        sh "docker build -t ${USER_NAME} ./back/jeonwoochi_spring/microservices/user-service/."
                        sh "docker build -t ${FESTIVAL_NAME} ./back/jeonwoochi_spring/microservices/festival-service/."
                        sh "docker build -t ${MAIN_NAME} ./back/jeonwoochi_spring/microservices/main-service/."

                                        // Django service 빌드
                        sh "docker build -t ${RECOMM_NAME} ./back/jeonwoochi_django/."
                        mattermostSend (
                            color: "good",
                            message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }catch(e){
                        mattermostSend (
                                color: "danger",
                                message: "Back Build FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
```

```
                    )
                }
            }
        }
    }

    stage('Eureka Depoly'){
            steps {
                script{
                    try{
                    sh "docker run -d --name=${DISCOVERY_CONTAINER_NAME} -p 8761:8761 --net msa ${DISCOVERY_NAME}"
                    sh 'sleep 5'
                    mattermostSend (
                        color: "good",
                        message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                    )
                    }catch(e){
                        mattermostSend (
                            color: "danger",
                            message: "Back Build FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }
                }
            }
        }

    stage('Config and Gateway Depoly'){
                steps {
                    script{
                        try{
                        sh "docker run -d --name=${CONFIG_CONTAINER_NAME} -p 8888:8888 --net msa ${CONFIG_NAME}"
                        sh "docker run -d --name=${GATEWAY_CONTAINER_NAME} -p 8000:8000 --net msa ${GATEWAY_NAME}"
                        sh 'sleep 15'
                        mattermostSend (
                            color: "good",
                            message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                        }catch(e){
                            mattermostSend (
                                color: "danger",
                                message: "Back Build FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                            )
                        }
                    }
                }
        }

        stage('Micro Service Deploy'){
            steps {
                script{
                    try{
                        sh "docker run -d --name=${USER_CONTAINER_NAME} --net msa ${USER_NAME}"
                        sh "docker run -d --name=${FESTIVAL_CONTAINER_NAME} --net msa ${FESTIVAL_NAME}"
                        sh "docker run -d --name=${MAIN_CONTAINER_NAME} --net msa ${MAIN_NAME}"
                        sh "docker run -d --name=${RECOMM_CONTAINER_NAME} --net msa ${RECOMM_NAME}"
                        //sh "docker run -d --name=${LOG_CONTAINER_NAME} --net msa ${LOG_NAME}"
                        mattermostSend (
                            color: "good",
                            message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }catch(e){
                        mattermostSend (
                            color: "danger",
                            message: "Back Build FAILED: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                        )
                    }
                }
            }
        }
        stage('Image Clean'){
            steps {
                script{
                    sh "docker image prune -a --force"
                    mattermostSend (
                        color: "good",
                        message: "Back Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to build>)"
                    )
                }
            }
        }
    }
}
```

Prometheus + Grafana

Prometheus 설정 파일 생성

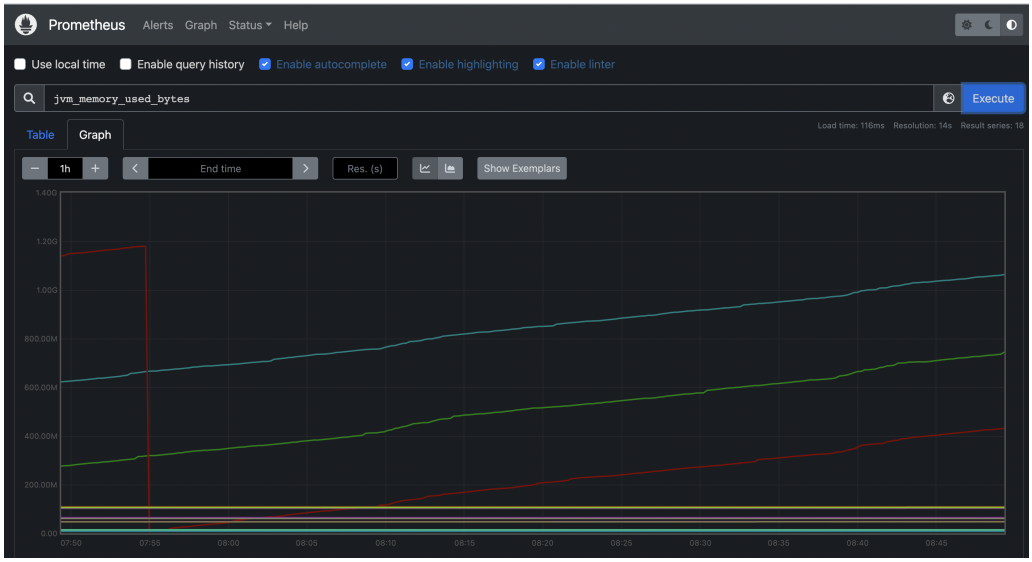
```
sudo mkdir /home/ubuntu/prometheus
sudo vi /home/ubuntu/prometheus/prometheus.yml
```

prometheus.yml

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.

scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets: ['localhost:9090']
  # 추가
  - job_name: 'user-service'
    scrape_interval: 15s
    metrics_path: '/api/user-service/actuator/prometheus'
    static_configs:
      - targets: ['j7b305.p.ssafy.io:8000']

  - job_name: 'festival-service'
    scrape_interval: 15s
    metrics_path: '/api/festival-service/actuator/prometheus'
    static_configs:
      - targets: ['j7b305.p.ssafy.io:8000']

  - job_name: 'main-service'
    scrape_interval: 15s
    metrics_path: '/api/main-service/actuator/prometheus'
    static_configs:
      - targets: ['j7b305.p.ssafy.io:8000']
```

Prometheus 실행

```
docker run -p 9191:9090 -v /home/ubuntu/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml --name prometheus -d prom/prometheus
```



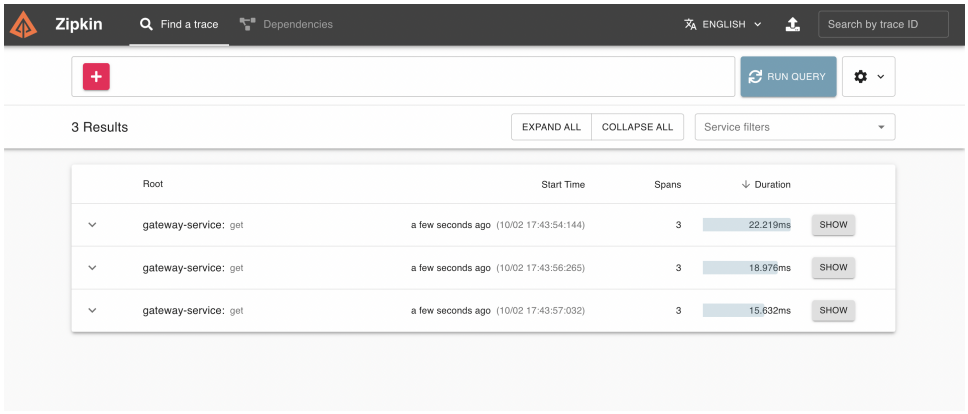
Grafana 실행

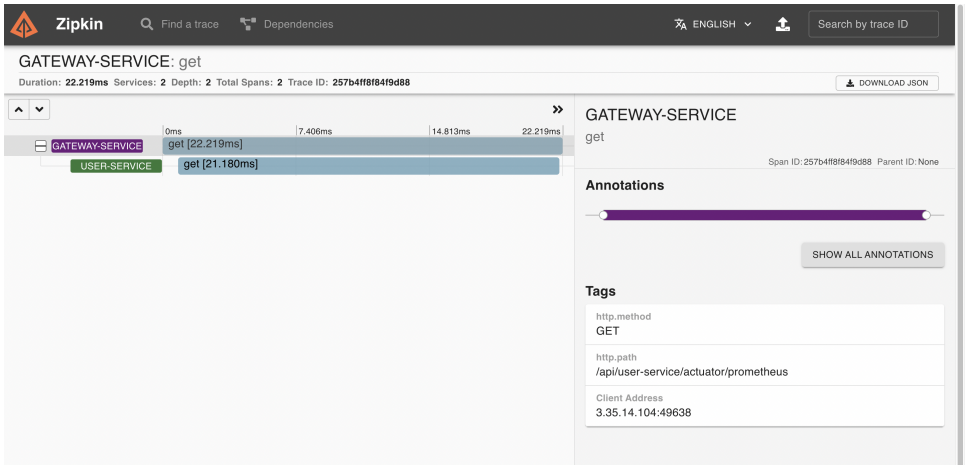
```
$ docker run -d --name=grafana -p 3000:3000 grafana/grafana
```

Zipkin

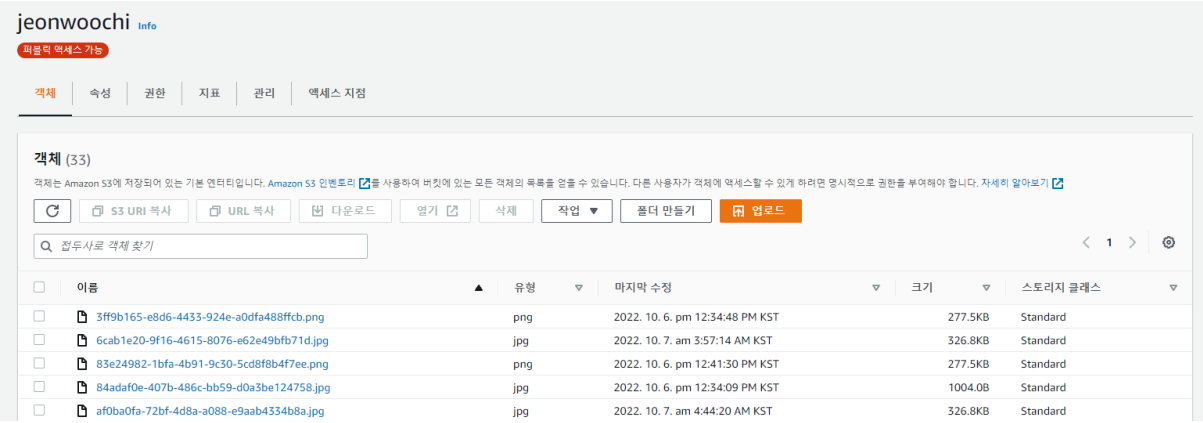
Zipkin 설치

```
docker run -d -p 9411:9411 openzipkin/zipkin
```




Zipkin
Find a trace
Dependencies
ENGLISH
Search by trace ID
GATEWAY-SERVICE: get
Duration: 22.219ms Services: 2 Depth: 2 Total Spans: 2 Trace ID: 257b4ff8f84f9d88
DOWNLOAD JSON
0ms
7.406ms
14.813ms
22.219ms
GATEWAY-SERVICE
get [22.219ms]
USER-SERVICE
get [21.180ms]
GATEWAY-SERVICE
get
Span ID: 257b4ff8f84f9d88 Parent ID: None
Annotations
SHOW ALL ANNOTATIONS
Tags
http.method
GET
http.path
/api/user-service/actuator/prometheus
Client Address
3.35.14.104:49638

S3


jeonwoochi Info
퍼블릭 액세스 가능
객체
속성
권한
지표
관리
액세스 지점
객체 (33)
객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. Amazon S3 인벤토리를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. 자세히 알아보기
S3 URI 복사
URL 복사
다운로드
열기
삭제
작업
폴더 만들기
업로드
접두사로 객체 찾기
이름
유형
마지막 수정
크기
스토리지 클래스
3ff9b165-e8d6-4433-924e-a0dfa488ffcb.png png 2022. 10. 6. pm 12:34:48 PM KST 277.5KB Standard
6cab1e20-9f16-4615-8076-e62e49bfb71d.jpg jpg 2022. 10. 7. am 3:57:14 AM KST 326.8KB Standard
83e24982-1bfa-4b91-9c30-5cd8f8b4f7ee.png png 2022. 10. 6. pm 12:41:30 PM KST 277.5KB Standard
84adaf0e-407b-486c-bb59-d0a3be124758.jpg jpg 2022. 10. 6. pm 12:34:09 PM KST 1004.0B Standard
af0ba0fa-72bf-4d8a-a088-e9aab4334b8a.jpg jpg 2022. 10. 7. am 4:44:20 AM KST 326.8KB Standard

Kakao


앱 키
플랫폼
앱 키
재발급
네이티브 앱 키
복사
재발급
REST API 키
복사
재발급
JavaScript 키
복사
재발급
Admin 키
복사
재발급
• 네이티브 앱 키: Android, iOS SDK에서 API를 호출할 때 사용합니다.
• JavaScript 키: JavaScript SDK에서 API를 호출할 때 사용합니다.
• REST API 키: REST API를 호출할 때 사용합니다.
• Admin 키: 모든 권한을 갖고 있는 키입니다. 노출이 되지 않도록 주의가 필요합니다.