1s Before 포팅 메뉴얼

태그

본 문서는 `1s Before`를 **실행에 필요한** 방법들을 기술하였습니다.

1. 개발 환경

Backend

- Java
  - Java OpenJDK 17
  - Spring Boot 3.2.11
    - JPA
    - Lombok
  - Gradle
- MySQL 9.1.0
- Redis 7.4.1
- Jenkins
- Docker
- Nignx
- Nginx Proxy Manger
- Minio 8.3.4
- ElasticSearch 7.17.25

Frontend

- NodeJS (22.15)

- Next.JS
- typescript
- tailwind css 20.18.0
- zustand
- tanstack/react-query

Server

- Ubuntu 20.04.6 1대
- AWS Linux c6i.2xlarge 2대

협업 툴

- API 공유: Notion, Postman, Swagger
- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost, Miro

2. 환경 세팅

총 3개의 서버를 운용하였습니다.

간단한 실행를 위하여 docker, docker-compose를 이용합니다.

docker-compose는 배포 환경을 기준으로 작성되었으며, 포트 번호를 참고하여 겹치지 않게 설정하면 로컬에서도 실행할 수 있습니다.

각 서버의 주 사용처는 아래와 같습니다.

- **Ubuntu**: Backend/Participation 서비스

  `participation`를 실행시키며 이벤트 참여 - Participation과 채팅 - Chat 관련된 서비스를 제공합니다.

  이벤트 참여와 관련이 있는 DB를 분리하여 운용합니다.

- **AWS Linux**: Frontend서비스와 Backend/Solution 서비스

  Next.JS를 사용한 `frontend`와 `Auth`, `Event`, `Search`, `Member` 서비스를 제공합니다.

- **AWS Linux** : Monitering용

  나머지 두 서버의 CPU 사용량, Memory 사용량, DB 접근 시간 등을 모니터링하는 서비스가 올라가는 서버입니다.

2-1) 공통

```
git clone https://lab.ssafy.com/s11-final/S11P31A404.git
```

docker container끼리의 통신을 위해 network를 구성합니다.

```
docker network create 404_dream_solutions_network
```

2-2) Frontend

```
cd ./frontend/
```

```
NEXT_PUBLIC_BASE_URL=https://59seconds.site/api/v2
NEXT_PUBLIC_WEBSOCKET_URL=wss://k11a404.p.ssafy.io/api/v2/ws
NEXT_PUBLIC_KAKAO_REST_API_KEY=${SECRET}
NEXT_PUBLIC_KAKAO_REDIRECT_URL=${SECRET}
```

2-3) Backend

각 SpringBoot의 `src/main/resources/` 아래 파일을 올립니다.

participation의 application-secret.yml

```
mysql:
  host: participation-mysql
  port: 3306
  database: dreamsolution
  username: ${SECRET}
  password: ${SECRET}

redis:
  host: my-redis
  port: 6379

spring:
  elasticsearch:
    uris: http://elasticsearch:9200
    username:
    password:

jwt:
  token:
    secret-key: ${SECRET}
  access:
    expiration: 3600
  refresh:
    expiration: 43200

random-nickname:
  size: 8000 # 랜덤 닉네임의 개수
```

solution의 application-secret.yml

```
mysql:
  host: solution-mysql
  port: 3306
  database: dreamsolution
```

```
  username: ${SECRET}
  password: ${SECRET}

redis:
  host: my-redis
  port: 6379

mongodb:
  host:
  post:
  username:
  password:

spring:
  elasticsearch:
    uris: http://elasticsearch:9200
    username:
    password:

jwt:
  token:
    secret-key: ${SECRET}
  access:
    expiration: 3600
  refresh:
    expiration: 43200

kakao:
  client:
    id: ${SECRET}
    secret: ${SECRET}
    redirect-uri: https://59seconds.site
  auth-url: https://kauth.kakao.com/oauth/authorize
  token-url: https://kauth.kakao.com/oauth/token
  user-info-url: https://kapi.kakao.com/v2/user/me
  grant-type: authorization_code
  scope: openid
```

```
random-nickname:
  size: 8000 # 랜덤 닉네임의 개수

minio:
  access-key: ${SECRET}
  secret-key: ${SECRET}
```

3. 실행

서비스명	Port
participation	9091:8000
mysql	3306:3306
minio	8998, 8999
redis	6379:6379

```
version: '3.7'

services:
  frontend:
    image: 404dreamsolutions/frontend:latest
    container_name: frontend
    ports:
      - "3000:3000"
    networks:
      - 404_dream_solutions_network

  solution:
    image: 404dreamsolutions/solution:latest
    container_name: solution
    ports:
      - "9090:8080"
    networks:
```

```
      - 404_dream_solutions_network

  solution-mysql:
    image: mysql:latest
    container_name: solution-mysql
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: dreamsolution
    MYSQL_USER: ${SECRET}
    MYSQL_PASSWORD: ${SECRET}
    ports:
      - "3306:3306"
    networks:
      - 404_dream_solutions_network

  redis:
    image: redis:latest
    container_name: my-redis
    ports:
      - "6379:6379"
    networks:
      - 404_dream_solutions_network

networks:
  404_dream_solutions_network:
    name: 404_dream_solutions_network
```

AWS EC2 (2)

서비스명	Port
frontend	3000:3000
solution	9090:8000
mysql	3306:3306
redis	6379:6379

서비스명	Port
elasticsearch	9200:9200
zookeeper	22, 2181:2181 2888, 3888
npm	3999:81
grafana	8086:8086

```
version: '3.7'

services:
 participation:
   image: ${PARTICIPATION_DOCKERHUB_REPO}:latest
   container_name: participation
   ports:
     - "9091:8080"
   environment:
     - SPRING_PROFILES_ACTIVE=prod
   networks:
     - 404_dream_solutions_network

 participation-mysql:
   image: mysql:latest
   container_name: participation-mysql
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: dreamsolution
    MYSQL_USER: ${SECRET}
    MYSQL_PASSWORD: ${SECRET}
   ports:
     - "3306:3306"
   networks:
     - 404_dream_solutions_network

 minio:
   image: minio/minio
   container_name: minio
   ports:
     - "8998-8999:8998-8999"
```

```
    networks:
      - 404_dream_solutions_network

 mysqld_exporter:
    image: prom/mysqld-exporter:latest
    container_name: mysqld_exporter
    ports:
      - "9104:9104"
    networks:
      - 404_dream_solutions_network

 redis:
    image: redis:latest
    container_name: my-redis
    ports:
      - "6379:6379"
    networks:
      - 404_dream_solutions_network

networks:
 404_dream_solutions_network:
    name: 404_dream_solutions_network
```

AWS EC2 (3)

모니터링 서버입니다.

```
version: '3'

services:
  grafana:
    image: grafana/grafana
    container_name: monitoring-grafana-1
    ports:
      - "3000:3000"
```

```
    command: /run.sh
    networks:
      - monitoring

  jaeger:
    image: jaegertracing/all-in-one:latest
    container_name: monitoring-jaeger-1
    ports:
      - "4317-4318:4317-4318"
      - "14250:14250"
      - "14268:14268"
      - "6831:6831/udp"
      - "8091:16686"
    environment:
      - COLLECTOR_ZIPKIN_HOST_PORT=:9411
    networks:
      - monitoring

  prometheus:
    image: prom/prometheus
    container_name: monitoring-prometheus-1
    ports:
      - "8095:9090"
    command:
      - --config.file=/etc/prometheus/prometheus.yml
    networks:
      - monitoring

networks:
  monitoring:
    driver: bridge
```