포팅메뉴얼

1. 개발환경

1.1. Frontend

- Node.js `v20.18.0`
- React `v18.3.1`
- Typerscript
- Zustand

1.2. Backend

- Springboot `v3.3.4`
- Gradle `v8.10.2`
- FastAPI `v0.115.4`

1.3. Database

- PostgreSQL `v16.4`
- Redis `v7.4.1`
- Elasticsearch `v8.13.4`

1.5. IDE

- Intellij `2024.2.1 (Ultimate Edition)`
- Visualstudio `1.91.0`

1.6. Infra

- AWS EC2 `Ubuntu 20.04.6 LTS`
- AWS S3
- AWS Lambda
- Dockerhub
- Docker `v27.3.1`
- Docker Compose `v2.29.3`
- nginx `v1.18.0`
- Jenkins `v2.483`

1.7. 형상/이슈관리

- Gitlab
- Jira
- Mattermost

1.8. 기타 툴

- Postman
- Figma
- Notion

2. 환경변수

2.1. Frontend

env

```
VITE_BASE_URL=

VITE_MUI_LiCENSE=

VITE_TOSS_PAYMENTS_NORMAL_CLIENT_KEY=
VITE_TOSS_PAYMENTS_NORMAL_SECRET_KEY=
VITE_TOSS_PAYMENTS_BILLING_CLIENT_KEY=
VITE_TOSS_PAYMENTS_BILLING_SECRET_KEY=
```

2.2. Backend

env

```
CONFIG_USERNAME=
CONFIG_PASSWORD=

DB_USERNAME=
DB_PASSWORD=

REDIS_PASSWORD=
```

Spring Cloud Config Server(Github)

sogoo-main-local.yml

```
server:
  servlet:
    context-path:
spring:
  application:
    name:
  datasource:
    url:
    username:
    password:
  jpa:
    hibernate:
      dialect:
      ddl-auto:
    properties:
      hibernate:
        format_sql:
        default_batch_fetch_size:
    show-sql:
  jwt:
    secret:
    time:
      access:
      refresh:
  mail:
```

```
    host:
    port:
    username:
    password:
    properties:
      mail:
        smtp:
          auth:
          starttls:
            enable:
            required:
          connectiontimeout:
          timeout:
          writetimeout:
    auth-code-expiration-millis:
  sql:
    init:
      data-locations:
      mode:
      encoding:
  servlet:
    multipart:
      max-file-size:
      max-request-size:
  ai:
    openai:
      api-key:
      chat:
        options:
          model:
          temperature:
  elasticsearch:
    uris:
  data:
    redis:
      host:
      port:
      password:

cloud:
  aws:
    s3:
      buckets:
        original:
        resized:
    stack:
      auto:
    region:
      static:
    credentials:
      accessKey:
      secretKey:

sangmoo:
  data:
    url:

toss:
```

```
  widget:
    secret-key:
  api:
    secret-key:

management:
  endpoints:
    web:
      exposure:
        include:

logging.level:
  org.hibernate.SQL:
  org.springframework.web:
```

sogoo-main-prod.yml

```
server:
  servlet:
    context-path:

spring:
  application:
    name:
  datasource:
    url:
    username:
    password:
  jpa:
    hibernate:
      dialect:
      ddl-auto:
    properties:
      hibernate:
        format_sql:
        default_batch_fetch_size:
    show-sql:
  jwt:
    secret:
    time:
      access:
      refresh:
  mail:
    host:
    port:
    username:
    password:
    properties:
      mail:
        smtp:
          auth:
          starttls:
            enable:
            required:
          connectiontimeout:
          timeout:
```

```
        writetimeout:
    auth-code-expiration-millis:
  sql:
    init:
      data-locations:
      mode:
      encoding:
  servlet:
    multipart:
      max-file-size:
      max-request-size:
  ai:
    openai:
      api-key:
      chat:
        options:
          model:
          temperature:
  elasticsearch:
    uris:
  data:
    redis:
      host:
      port:
      password:

sangmoo:
  data:
    url:

cloud:
  aws:
    s3:
      buckets:
        original:
        resized:
    stack:
      auto:
    region:
      static:
    credentials:
      accessKey:
      secretKey:

toss:
  widget:
    secret-key:
  api:
    secret-key:

management:
  endpoints:
    web:
      exposure:
        include:

logging.level:
```

```
  org.hibernate.SQL:
  org.springframework.web:
```

main - application.yml

```
spring:
  config:
    import:
  cloud:
    config:
      fail-fast:
      label:
      name:
      username: ${CONFIG_USERNAME}
      password: ${CONFIG_PASSWORD}

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

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

2.3. 설정파일

2.3.1 Nginx: Nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
```

```
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    gzip on;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

2.3.2 Nginx: default

```
# HTTP 요청을 HTTPS www.sogoo.kr로 리디렉션하는 서버 블록
server {
    listen 80;
    listen [::]:80;

    server_name sogoo.kr www.sogoo.kr k11c107.p.ssafy.io;

    # 모든 HTTP 요청을 HTTPS www.sogoo.kr로 리디렉션
    return 301 https://www.sogoo.kr$request_uri;
}

# 루트 도메인(sogoo.kr) 요청을 www.sogoo.kr로 리디렉션하는 서버 블록
server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name sogoo.kr;

    # HTTPS 요청을 www.sogoo.kr로 리디렉션
    return 301 https://www.sogoo.kr$request_uri;

    # SSL 인증서 설정 (Certbot에 의해 관리됨)
    ssl_certificate /etc/letsencrypt/live/sogoo.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/sogoo.kr/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

# www.sogoo.kr 요청 처리 서버 블록
server {
    listen 443 ssl;
    listen [::]:443 ssl ipv6only=on;

    server_name www.sogoo.kr k11c107.p.ssafy.io;

    # SSL 인증서 설정 (Certbot에 의해 관리됨)
    ssl_certificate /etc/letsencrypt/live/sogoo.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/sogoo.kr/privkey.pem;
```

```
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # 업로드 파일 크기 제한
    client_max_body_size 30M;

        location / {
        proxy_pass http://localhost:5173;
            proxy_http_version 1.1;
            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;

            # 프록시 버퍼 설정 (필요 시 조정)
            proxy_buffer_size 128k;
            proxy_buffers 4 256k;
            proxy_busy_buffers_size 256k;
    }

    # 백엔드 메인 애플리케이션
    location /api/ {
            proxy_pass http://localhost:8080;
            proxy_http_version 1.1;
            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;

            # 프록시 버퍼 설정
            proxy_buffer_size 128k;
            proxy_buffers 4 256k;
            proxy_busy_buffers_size 256k;
    }

       # Config Server 리버스 프록시 설정
       location /config/ {
            proxy_pass http://localhost:8888/;
            proxy_http_version 1.1;
            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;
        }

    # Elasticsearch 리버스 프록시 설정 예시
        location /elasticsearch/ {
        proxy_pass http://localhost:9200/;
        proxy_http_version 1.1;
            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;
    }
}
```

2.3.3 Jenkins: Jenkinsfile

main

```
@Library('shared-library@develop/back') _

pipeline {
    agent any
    environment {
        SSH_CREDENTIALS = 'ec2-ssh-key'  // EC2 SSH 키 설정
    }

    stages {
        stage('Check Environment Variables') {
            steps {
                sh 'echo "EC2_IP is ${EC2_IP}"'
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'develop/back',
                    url: 'https://lab.ssafy.com/s11-final/S11P31C107.git',
                    credentialsId: 'c107-sogoo-project-access-token'

                // Git 상태와 로그 확인
                sh 'echo "Current Git Status:"'
                sh 'git status'
                sh 'echo "Latest Commit:"'
                sh 'git log -1'
            }
        }
        stage('Check for Changes') {
            steps {
                script {
                    // 변경 사항이 있는지 체크하여 env.CHANGES 변수에 저장
                    env.CHANGES = sh(script: "git diff --name-only HEAD~20 HEAD", returnStdou
t: true).trim()
                    echo "Changes detected: ${env.CHANGES}"

                    // 지정된 폴더에 변경 사항이 없으면 파이프라인을 종료
                    if (!env.CHANGES.contains("backend/main/")) {
                        echo "No changes detected in './backend/main/'. Skipping the build."
                        currentBuild.result = 'ABORTED'
                        error("No changes in the relevant folder, stopping the build.")
                    } else {
                        echo "Changes detected in './backend/main/'. Proceeding with the buil
d."
                    }
                }
            }
        }

        stage('Set Permission') {
            when {
                expression { env.CHANGES.contains("backend/main/") }
            }
            steps {
```

```
                dir('./backend/main') {
                    sh 'chmod +x ./gradlew'  // 실행 권한 부여
                }
            }
        }
        stage('Build Jar') {
            when {
                expression { env.CHANGES.contains("backend/main/") }
            }
            steps {
                dir('./backend/main') {
                    sh './gradlew clean build -x test'  // Gradle 빌드
                }
            }
        }
        stage('Build Docker Image') {
            when {
                expression { env.CHANGES.contains("backend/main/") }
            }
            steps {
                // Docker 빌드 로그 추가
                echo "Building Docker Image 'sogoo-be-main:main-latest' from './backend/main'
directory"
                sh 'docker build --no-cache -t sogoo-be-main:main-latest ./backend/main'  //
Docker 이미지 빌드
            }
        }
        stage('Push and Deploy') {
            when {
                expression { env.CHANGES.contains("backend/main/") }
            }
            steps {
                script {
                    withCredentials([usernamePassword(credentialsId: 'c107-sogoo-dockerhub-ac
cess-token', usernameVariable: 'DOCKERHUB_USER', passwordVariable: 'DOCKERHUB_PASS')]) {
                        // Docker Hub에 로그인하고 이미지 푸시
                        echo "Logging into Docker Hub and pushing image"
                        sh 'echo $DOCKERHUB_PASS | docker login -u $DOCKERHUB_USER --password
-stdin'
                        sh 'docker tag sogoo-be-main:main-latest $DOCKERHUB_USER/sogoo-be-mai
n:main-latest'
                        sh 'docker push $DOCKERHUB_USER/sogoo-be-main:main-latest'

                        // EC2에 배포
                        echo "Deploying Docker image to EC2 at ${EC2_IP}"
                        sshagent([SSH_CREDENTIALS]) {
                            sh """
                            # docker-compose.yml 파일을 EC2로 전송
                            scp -o StrictHostKeyChecking=no ./docker-compose.be-main.yml ubun
tu@${EC2_IP}:/home/ubuntu/sogoo/docker-compose.be-main.yml
                            # EC2에서 Docker Compose 실행 (Docker Hub 크레덴셜 필요 없음)
                            ssh -o StrictHostKeyChecking=no ubuntu@${EC2_IP} '
                                cd /home/ubuntu/sogoo
                                docker-compose -f docker-compose.be-main.yml down
                                docker-compose -f docker-compose.be-main.yml pull
                                docker-compose -f docker-compose.be-main.yml up -d --build
                                '
                            """
```

```
                    }
                }
            }
        }
    }
    stage('Clean Up Docker Images') {
        when {
            expression { env.CHANGES.contains("backend/main/") }
        }
        steps {
            echo "Cleaning up unused Docker images"
            sh 'docker image prune -a -f'  // 사용되지 않는 모든 도커 이미지 삭제
        }
    }
    stage('Refresh Config') {
        steps {
            script {
                echo "Triggering /actuator/refresh to apply new configuration"
                    sh 'curl -X POST -u ${CONFIG_USERNAME}:${CONFIG_PASSWORD} https://www.sog
oo.kr/config/actuator/refresh'

                }
            }
        }
    }
    post {
        success {
            echo 'sogoo be main deploy success'
            sendNotification('success')
        }
        aborted {
            echo 'sogoo be main no change.'
        }
        failure {
            echo 'sogoo be main deploy failure'
            sendNotification('failure')
        }
    }
}
```

config

```
@Library('shared-library@develop/back') _

pipeline {
    agent any
    environment {
        SSH_CREDENTIALS = 'ec2-ssh-key'  // EC2 SSH 키 설정
    }

    stages {
        stage('Check Environment Variables') {
            steps {
                sh 'echo "EC2_IP is ${EC2_IP}"'
            }
        }
```

```
        stage('Git Clone') {
            steps {
                git branch: 'develop/back',
                    url: 'https://lab.ssafy.com/s11-final/S11P31C107.git',
                    credentialsId: 'c107-sogoo-project-access-token'

                // Git 상태와 로그 확인
                sh 'echo "Current Git Status:"'
                sh 'git status'
                sh 'echo "Latest Commit:"'
                sh 'git log -1'
            }
        }
        stage('Check for Changes') {
            steps {
                script {
                    // 변경 사항이 있는지 체크하여 env.CHANGES 변수에 저장
                    env.CHANGES = sh(script: "git diff --name-only HEAD~1 HEAD", returnStdou
t: true).trim()
                    echo "Changes detected: ${env.CHANGES}"

                    // 지정된 폴더에 변경 사항이 없으면 파이프라인을 종료
                    if (!env.CHANGES.contains("backend/config/")) {
                        echo "No changes detected in './backend/config/'. Skipping the buil
d."
                        currentBuild.result = 'ABORTED'
                        error("No changes in the relevant folder, stopping the build.")
                    } else {
                        echo "Changes detected in './backend/config/'. Proceeding with the bu
ild."
                    }
                }
            }
        }
        stage('Setup') {
            when {
                expression { env.CHANGES.contains("backend/config/") }
            }
            steps {
                withCredentials([file(credentialsId: 'c107-sogoo-back-config-yml', variable:
'MY_SECRET_FILE')]) {
                    sh 'mkdir -p ./backend/config/src/main/resources'  // 디렉토리 생성
                    sh 'chmod -R 755 ./backend/config/src/main/resources'  // 권한 설정
                    sh 'cp $MY_SECRET_FILE ./backend/config/src/main/resources/application.ym
l'  // 파일 복사
                }
            }
        }
        stage('Set Permission') {
            when {
                expression { env.CHANGES.contains("backend/config/") }
            }
            steps {
                dir('./backend/config') {
                    sh 'chmod +x ./gradlew'  // 실행 권한 부여
                }
            }
        }
```

```
        stage('Build Jar') {
            when {
                expression { env.CHANGES.contains("backend/config/") }
            }
            steps {
                dir('./backend/config') {
                    sh './gradlew clean build -x test'  // Gradle 빌드
                }
            }
        }
        stage('Build Docker Image') {
            when {
                expression { env.CHANGES.contains("backend/config/") }
            }
            steps {
                // Docker 빌드 로그 추가
                echo "Building Docker Image 'sogoo-be-main:main-latest' from './backend/confi
g' directory"
                sh 'docker build -t sogoo-be-config:config-latest ./backend/config'  // Docke
r 이미지 빌드
            }
        }
        stage('Push and Deploy') {
            when {
                expression { env.CHANGES.contains("backend/config/") }
            }
            steps {
                script {
                    withCredentials([usernamePassword(credentialsId: 'c107-sogoo-dockerhub-ac
cess-token', usernameVariable: 'DOCKERHUB_USER', passwordVariable: 'DOCKERHUB_PASS')]) {
                        // Docker Hub에 로그인하고 이미지 푸시
                        echo "Logging into Docker Hub and pushing image"
                        sh 'echo $DOCKERHUB_PASS | docker login -u $DOCKERHUB_USER --password
-stdin'
                        sh 'docker tag sogoo-be-config:config-latest $DOCKERHUB_USER/sogoo-be
-config:config-latest'
                        sh 'docker push $DOCKERHUB_USER/sogoo-be-config:config-latest'

                        // EC2에 배포
                        echo "Deploying Docker image to EC2 at ${EC2_IP}"
                        sshagent([SSH_CREDENTIALS]) {
                            sh """
                            # docker-compose.yml 파일을 EC2로 전송
                            scp -o StrictHostKeyChecking=no ./docker-compose.be-config.yml ub
untu@${EC2_IP}:/home/ubuntu/sogoo/docker-compose.be-config.yml
                            # EC2에서 Docker Compose 실행 (Docker Hub 크레덴셜 필요 없음)
                            ssh -o StrictHostKeyChecking=no ubuntu@${EC2_IP} '
                                cd /home/ubuntu/sogoo
                                docker-compose -f docker-compose.be-config.yml down
                                docker-compose -f docker-compose.be-config.yml pull
                                docker-compose -f docker-compose.be-config.yml up -d --build
                                '
                            """
                        }
                    }
                }
            }
        }
```

```
        stage('Clean Up Docker Images') {
            when {
                expression { env.CHANGES.contains("backend/config/") }
            }
            steps {
                echo "Cleaning up unused Docker images"
                sh 'docker image prune -a -f'  // 사용되지 않는 모든 도커 이미지 삭제
            }
        }
    }
    post {
        success {
            echo 'sogoo be config deploy success'
            sendNotification('success')
        }
        aborted {
            echo 'sogoo be config no change.'
        }
        failure {
            echo 'sogoo be config deploy failure'
            sendNotification('failure')
        }
    }
}
```

Front

```
@Library('shared-library@develop/front') _

pipeline {
    agent any
    environment {
        SSH_CREDENTIALS = 'ec2-ssh-key'  // EC2 SSH 키 설정
    }

    stages {
        stage('Load Environment Variables from Secret File') {
            steps {
                withCredentials([file(credentialsId: 'c107-sogoo-front-env', variable: 'SECRE
T_ENV_FILE')]) {
                    sshagent([SSH_CREDENTIALS]) {
                        sh 'scp -o StrictHostKeyChecking=no ${SECRET_ENV_FILE} ubuntu@${EC2_I
P}:/home/ubuntu/sogoo/.env'
                        // 배포 서버에 .env 파일 전송
                    }
                }
            }
        }
        stage('Check Environment Variables') {
            steps {
                sh 'echo "EC2_IP is ${EC2_IP}"'
                sh 'echo "VITE_BASE_URL is ${VITE_BASE_URL}"'  // 추가 확인
            }
        }
        stage('Git Clone') {
            steps {
```

```
                git branch: 'develop/front',
                    url: 'https://lab.ssafy.com/s11-final/S11P31C107.git',
                    credentialsId: 'c107-sogoo-project-access-token'
            }
        }
        stage('Install Dependencies') {
            steps {
                dir('./frontend') {
                    sh 'npm install'
                }
            }
        }
        stage('Build Frontend') {
            steps {
                dir('./frontend') {
                    // VITE_BASE_URL이 포함된 .env가 제대로 적용되었는지 확인 후 빌드
                    sh 'echo "Building frontend with VITE_BASE_URL=$VITE_BASE_URL"'
                    sh 'npm run build'
                }
            }
        }
        stage('Front Docker Image') {
            steps {
                sh '''
                    docker build \
                        --build-arg VITE_BASE_URL=$VITE_BASE_URL \
                        --build-arg VITE_TOSS_PAYMENTS_NORMAL_CLIENT_KEY=$VITE_TOSS_PAYMENTS_
NORMAL_CLIENT_KEY \
                        --build-arg VITE_TOSS_PAYMENTS_NORMAL_SECRET_KEY=$VITE_TOSS_PAYMENTS_
NORMAL_SECRET_KEY \
                        --build-arg VITE_TOSS_PAYMENTS_BILLING_CLIENT_KEY=$VITE_TOSS_PAYMENTS
_BILLING_CLIENT_KEY \
                        --build-arg VITE_TOSS_PAYMENTS_BILLING_SECRET_KEY=$VITE_TOSS_PAYMENTS
_BILLING_SECRET_KEY \
                        -t sogoo-front:latest ./frontend
                '''
            }
        }
        stage('Push and Deploy') {
            steps {
                script {
                    withCredentials([usernamePassword(credentialsId: 'c107-sogoo-dockerhub-ac
cess-token', usernameVariable: 'DOCKERHUB_USER', passwordVariable: 'DOCKERHUB_PASS')]) {
                        sh 'echo $DOCKERHUB_PASS | docker login -u $DOCKERHUB_USER --password
-stdin'
                        sh 'docker tag sogoo-front:latest $DOCKERHUB_USER/sogoo-front:latest'
                        sh 'docker push $DOCKERHUB_USER/sogoo-front:latest'

                        sshagent([SSH_CREDENTIALS]) {
                            sh """
                                scp -o StrictHostKeyChecking=no ./docker-compose.frontend.yml
ubuntu@${EC2_IP}:/home/ubuntu/sogoo/docker-compose.frontend.yml
                                ssh -o StrictHostKeyChecking=no ubuntu@${EC2_IP} '
                                    cd /home/ubuntu/sogoo
                                    docker-compose -f docker-compose.frontend.yml down
                                    docker-compose -f docker-compose.frontend.yml pull
                                    docker-compose -f docker-compose.frontend.yml up -d
                                '
```

```
                            """
                        }
                    }
                }
            }
        }
        stage('Clean Up Docker Images') {
            steps {
                sh 'docker image prune -a -f'
            }
        }
    }
    post {
        success {
            echo 'sogoo fe deploy success'
            sendNotification('success')
        }
        failure {
            echo 'sogoo fe deploy failure'
            sendNotification('failure')
        }
    }
}
```

2.3.4 Dockerfile: Backend

main

```
FROM openjdk:17-jdk
WORKDIR /app
COPY build/libs/*.jar app.jar

ENV TZ=Asia/Seoul
ENV SPRING_PROFILES_ACTIVE=prod

ADD https://raw.githubusercontent.com/vishnubob/wait-for-it/master/wait-for-it.sh /wait-for-i
t.sh
RUN chmod +x /wait-for-it.sh

CMD ["/wait-for-it.sh", "postgres:5432", "--timeout=30", "--strict", "--", "java", "-Duser.ti
mezone=Asia/Seoul", "-jar", "app.jar"]
```

config

```
FROM openjdk:17

ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar

ENV TZ=Asia/Seoul

ENTRYPOINT ["java", "-jar", "app.jar"]
```

2.3.5 Dockerfile: Front

```
# 1단계: Node.js 이미지를 사용하여 React 애플리케이션을 빌드
FROM node:20.18.0 AS build

# 빌드 인자로 받아 설정
ARG VITE_BASE_URL
ARG VITE_TOSS_PAYMENTS_NORMAL_CLIENT_KEY
ARG VITE_TOSS_PAYMENTS_NORMAL_SECRET_KEY
ARG VITE_TOSS_PAYMENTS_BILLING_CLIENT_KEY
ARG VITE_TOSS_PAYMENTS_BILLING_SECRET_KEY

# 환경 변수 설정
ENV VITE_BASE_URL=$VITE_BASE_URL
ENV VITE_TOSS_PAYMENTS_NORMAL_CLIENT_KEY=$VITE_TOSS_PAYMENTS_NORMAL_CLIENT_KEY
ENV VITE_TOSS_PAYMENTS_NORMAL_SECRET_KEY=$VITE_TOSS_PAYMENTS_NORMAL_SECRET_KEY
ENV VITE_TOSS_PAYMENTS_BILLING_CLIENT_KEY=$VITE_TOSS_PAYMENTS_BILLING_CLIENT_KEY
ENV VITE_TOSS_PAYMENTS_BILLING_SECRET_KEY=$VITE_TOSS_PAYMENTS_BILLING_SECRET_KEY

# 작업 디렉토리를 /app으로 설정
WORKDIR /app
ENV TZ=Asia/Seoul

# package.json과 package-lock.json 복사 후 의존성 설치
COPY package.json package-lock.json* ./
RUN npm install

# 모든 소스 코드를 복사하여 빌드 실행
COPY . .

RUN npm run build

# 2단계: NGINX 이미지를 사용하여 빌드된 정적 파일 서빙
FROM nginx:alpine

# NGINX 설정 파일 복사
COPY ./nginx.conf /etc/nginx/conf.d/default.conf

# Node.js 빌드 단계에서 생성된 dist 폴더를 NGINX의 기본 경로로 복사
COPY --from=build /app/dist /usr/share/nginx/html

# NGINX가 80 포트에서 HTTP 요청을 수신
EXPOSE 80

# NGINX를 포그라운드 모드로 실행
CMD ["nginx", "-g", "daemon off;"]
```

2.3.6 Front: nginx.conf

```
server {
    listen 80;  # 컨테이너 내부에서 80 포트로 청취
    server_name localhost;

    # 정적 파일 제공 위치 설정
    root /usr/share/nginx/html;
    index index.html;
```

```
    # 모든 요청을 index.html로 전달 (React Router 대응)
    location / {
        try_files $uri $uri/ /index.html;
    }

    # 캐싱 설정
    # location ~* \.(js|css|png|jpg|jpeg|gif|svg|ico)$ {
    #     expires 7d;
    #     add_header Cache-Control "public, max-age=604800, immutable";
    # }

    # Gzip 압축 설정
    gzip on;
    gzip_types text/css application/javascript application/json image/svg+xml;
    gzip_min_length 256;
}
```

2.3.7 DockerCompose: Backend

docker-compose.be-main.yml

```
services:
  backend:
    image: zzun73/sogoo-be-main:main-latest
    container_name: sogoo-be-main
    ports:
      - "8080:8080"
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://postgres:5432/sosang
      SPRING_DATASOURCE_USERNAME: ${DB_USERNAME}
      SPRING_DATASOURCE_PASSWORD: ${DB_PASSWORD}
      CONFIG_USERNAME: ${CONFIG_USERNAME}
      CONFIG_PASSWORD: ${CONFIG_PASSWORD}
    depends_on:
      - postgres
      - elasticsearch
    networks:
      - sogoo-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8080 || exit 1"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

  postgres:
    image: postgres:16.4
    container_name: sogoo-postgres
    environment:
      POSTGRES_USER: ${DB_USERNAME}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres-data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    networks:
```

```
      - sogoo-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${DB_USERNAME}"]
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 10s

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.13.4
    container_name: sogoo-elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - ES_JAVA_OPTS=-Xms3g -Xmx3g
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - sogoo-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:9200 || exit 1"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 10s
    command: >
      bash -c "./bin/elasticsearch-plugin install analysis-nori || exit 1;
      /usr/local/bin/docker-entrypoint.sh"

  redis:
    image: redis:latest
    container_name: sogoo-redis
    ports:
      - "6379:6379"
    networks:
      - sogoo-network
    volumes:
      - redis-data:/data
    restart: always
    command: ["redis-server", "--appendonly", "yes", "--requirepass", "${REDIS_PASSWORD}"]


networks:
  sogoo-network:
    driver: bridge

volumes:
  postgres-data:
  redis-data:
```

docker-compose-be-config.yml

```
services:
  backend-config:
    image: zzun73/sogoo-be-config:config-latest
```

```
    container_name: sogoo-be-config
    ports:
      - "8888:8888"
    networks:
      - sogoo-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8888 || exit 1"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

networks:
  sogoo-network:
    driver: bridge
```

docker-compose.be-securities.yml

```
services:
  backend-securities:
    image: zzun73/pigin-be-securities:securities-latest
    container_name: pigin-be-securities
    ports:
      - "8089:8089"
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://pigin-postgres:5432/postgres
      SPRING_DATASOURCE_USERNAME: ${DB_USERNAME}
      SPRING_DATASOURCE_PASSWORD: ${DB_PASSWORD}
    networks:
      - myapp_pigin-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8089 || exit 1"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

networks:
  myapp_pigin-network:
    external: true
```

docker-compose.be-wallet.yml

```
services:
  backend-wallet:
    image: zzun73/pigin-be-wallet:wallet-latest
    container_name: pigin-be-wallet
    ports:
      - "8088:8088"
    networks:
      - pigin-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8088 || exit 1"]
      interval: 30s
      timeout: 10s
```

```
      retries: 3
      start_period: 40s

networks:
  pigin-network:
    driver: bridge
```

2.3.8 DockerCompose: Frontend

```
services:
  frontend:
    image: zzun73/sogoo-front:latest
    container_name: sogoo-front
    ports:
      - "5173:80"
    env_file: /home/ubuntu/sogoo/.env
    networks:
      - sogoo-network
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost || exit 1"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

networks:
  sogoo-network:
    driver: bridge
```

3. EC2 세팅

3.1. Docker 설치

```
# 패키지 업데이트
sudo apt-get update

# 패키지 설치
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common

# Docker의 공식 GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# Docker 저장소 추가
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"

# 패키지 재 업데이트
```

```
sudo apt-get update

# Docker CE 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io

# Docker 서비스 시작
sudo systemctl start docker

# Docker 설치 확인
docker --version
```

3.2. Docker 실행 권한 설정

```
# 도커그룹생성
sudo groupadd docker

# 도커그룹에 유저추가 (Docker 데몬에 대한 접근 권한을 부여)
sudo usermod -aG docker ${USER} or ${whoami}
# or
sudo gpasswd -a $USER docker

# 도커 재시작
sudo service docker restart
# or
newgrp docker # 현재 세션의 사용자 그룹을 변경


# 현재 사용자 로그아웃 및 재로그인 필수
# exit 후 su username

# 테스트
docker ps
```

3.3. Docker Compose 설치

```
# docker-compose 설치
sudo curl -L "https://github.com/docker/compose/releases/download/v2.29.3/docker-compose-$(un
ame -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# 테스트
docker compose --version
```

3.4. Nginx 설치

```
# Ubuntu에서 Certbot 설치하기
sudo apt update
sudo apt install certbot
sudo apt install python3-certbot-nginx

# Certbot을 이용한 SSL 인증서 발급
sudo certbot --nginx

# certbot 버전 확인
```

```
certbot --version
certbot 0.40.0
```

3.5. 방화벽(UFW) 설정

```
sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
80                         ALLOW       Anywhere
9090/tcp                   ALLOW       Anywhere
5432/tcp                   ALLOW       Anywhere
9200                       ALLOW       Anywhere
6379                       ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
8989 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
9090/tcp (v6)              ALLOW       Anywhere (v6)
5432/tcp (v6)              ALLOW       Anywhere (v6)
9200 (v6)                  ALLOW       Anywhere (v6)
6379 (v6)                  ALLOW       Anywhere (v6)
```