포팅 메뉴얼

사용 서비스 및 버전

프론트

vite 5.4.1

npm 10.8.1

react 18.3.1

typescript 5.5.4

redux 5.0.1

react-query 3.39.3

tailwind 3.4.10

백엔드

Jdk17

Spring boot 3.3.3

Query-dsl 5.0.0

Swagger 2.0.2

DB

MySQL 9.0.1

Redis 7.4.0

Infra

Docker 27.2.1

Jenkins 2.476

Nginx 1.27.1

Ubuntu 20.04.6 LTS

AWS EC2, S3

프로젝트 설정

백엔드 설정 파일 (application.yml)

```
server:
  port: 8081

spring:
  application:
    name: stockolm

  servlet:
    multipart:
      max-file-size: 25MB
      max-request-size: 25MB

  jpa:
    open-in-view: false
    properties:
      hibernate:
        default_batch_fetch_size: 50
        show_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
        jdbc:
          time_zone: Asia/Seoul

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://mysql:3306/stockolm
    username: stockolm
    password: stockolm
  sql:
```

```
    init:
      mode: always

  mail:
    host: smtp.naver.com
    port: 587
    username: 메일 샌더 계정
    password: 매일 샌더 계정 비밀번호
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true
  data:
    web:
      pageable:
        max-page-size: 2000
        default-page-size: 10
    redis:
      host: redis
      port: 6379

jwt:
  salt: JWT salt 값
  access-token:
    expiretime: 3600000
  refresh-token:
    expiretime: 2592000000

web-client:
  korea-invest:
    domain: https://openapi.koreainvestment.com:9443
    url:
      simple-read: /uapi/domestic-stock/v1/quotations/search-stock-info
    key:
      app_key: 한국 투자 증권 API
      app_secret: 한국 투자 증권 API
      access_token: 한국 투자 증권 API
  open-dart:
    key: 오픈 다트 API 키

cloud:
  aws:
    s3:
      bucket: 버켓 이름
    stack.auto: false
    region.static: ap-northeast-2
    credentials:
      accessKey: S3 키 값
      secretKey: S3 키 값
```

프론트 설정 파일 (.env)

```
VITE_STOCK_APP_KEY = 한국 투자 증권 api APP_KEY 키
VITE_STOCK_APP_SECRET = 한국 투자 증권 api APP_SECRET 키
VITE_STOCK_ACCESS_TOKEN = 한국 투자 증권 api ACCESS_TOKEN 키

VITE_SUMMARY_ACCESS_TOKEN = 구글 OCR ACCESS_TOKEN 키
```

배포 설정

도커 컴포즈 파일(docker-compose.yml)

```
version: "3.7"

services:
  nginx:
```

```
    image: nginx
    container_name: nginx
    restart: unless-stopped
    volumes:
      - ./nginx_data/nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx_data/nginx/conf.d/:/etc/nginx/conf.d/
      - ./nginx_data/certbot/letsencrypt:/etc/letsencrypt
      - ./nginx_data/certbot/www:/var/www/certbot
      - ./nginx_data/nginx_log:/var/log/nginx
    ports:
      - "80:80"
      - "443:443"
    environment:
      TZ: "Asia/Seoul"
    command: "/bin/sh -c 'while :; do sleep 360h & wait $${!}; nginx -s reload; done & nginx -g \"daemon off;\"'"
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "1"
    networks:
      - stockolmnet

  nginx-frontend:
    image: nginx
    container_name: nginx-frontend
    restart: unless-stopped
    volumes:
      - ./nginx_frontend_data/nginx.conf:/etc/nginx/nginx.conf
      - ./frontend/dist:/usr/share/nginx/html
      - ./nginx_frontend_data/nginx_log:/var/log/nginx
    ports:
      - "3000:3000"
    environment:
      TZ: "Asia/Seoul"
    command: "/bin/sh -c 'while :; do sleep 360h & wait $${!}; nginx -s reload; done & nginx -g \"daemon off;\"'"
    networks:
      - stockolmnet

  certbot:
    image: certbot/certbot
    container_name: certbot
    volumes:
      - ./nginx_data/certbot/letsencrypt:/etc/letsencrypt
      - ./nginx_data/certbot/www:/var/www/certbot
    entrypoint: /bin/sh -c 'trap exit TERM; while :; do sleep 12h & wait $${!}; certbot renew; done;'
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "1"
    networks:
      - stockolmnet

  jenkins:
    image: jenkins/jenkins
    container_name: jenkins
    restart: unless-stopped
    volumes:
      - jenkins_home:/var/jenkins_home
      - ./jenkins_data/jenkins:/var/jenkins_shared
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/local/compose:/usr/local/compose
      - /usr/bin/docker:/usr/bin/docker
      - /home/ubuntu/stockolm:/home/ubuntu/stockolm
      # - ./gcp:/var/cache/jenkins
    ports:
      - "8080:8080"
      - "50000:50000"
    environment:
      JENKINS_OPTS: --prefix=/jenkins
      TZ: "Asia/Seoul"
    user: root
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
```

```
        max-file: "1"
    networks:
      - stockolmnet

  backend:
    image: backend
    container_name: backend
    environment:
      SPRING_DATASOURCE_HIKARI_JDBC-URL: jdbc:mysql://mysql:3306/stockolm?serverTimezone=Asia/Seoul&useUniCode=yes&charact
erEncoding=UTF-8
      SPRING_DATASOURCE_HIKARI_USERNAME: stockolm
      SPRING_DATASOURCE_HIKARI_PASSWORD: stockolm
      SPRING_DATASOURCE_HIKARI_DRIVER-CLASS-NAME: com.mysql.cj.jdbc.Driver
      SPRING_JPA_PROPERTIES_HIBERNATE_DIALECT: org.hibernate.dialect.MySQL8Dialect
      TZ: "Asia/Seoul"
    ports:
      - "8081:8081"
    networks:
      - stockolmnet
    depends_on:
      - mysql

  frontend:
    image: frontend
    container_name: frontend
    env_file:
      - .env
    restart: "no"
    command: sh -c "npm run build && sleep infinity"
    volumes:
      - ./frontend/dist:/app/dist
    networks:
      - stockolmnet

  mysql:
    image: mysql
    container_name: mysql
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: stockolm
      MYSQL_DATABASE: stockolm
      MYSQL_CHARSET: utf8mb4
      MYSQL_USER: stockolm
      MYSQL_PASSWORD: stockolm
      MYSQL_COLLACTION: utf8mb4_unicode_ci
      TZ: "Asia/Seoul"
    volumes:
      - ./mysql_data/data:/var/lib/mysql
    ports:
      - "3306:3306"
    networks:
      - stockolmnet

  redis:
    image: redis
    container_name: redis
    restart: unless-stopped
    environment:
      TZ: "Asia/Seoul"
    ports:
      - "6379:6379"
    volumes:
      - ./redis.conf:/usr/local/etc/redis/redis.conf
    command: redis-server /usr/local/etc/redis/redis.conf
    networks:
      - stockolmnet

volumes:
  jenkins_home:

networks:
  stockolmnet:
```

리버스 프록싱 Nginx 설정 파일 (nginx.conf)

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

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

    proxy_headers_hash_max_size 51200;
    proxy_headers_hash_bucket_size 6400;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"'
                      '"$host" "$server_name" "$request_uri" "$uri" '
                      '"$request_body" "$args" "$upstream_addr" "$upstream_status"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;

    # 요청 제한 설정
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=1r/s;

    server {
        listen 80;
        server_name j11b201.p.ssafy.io;
        server_tokens off;

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
        }

        #301 https://$host$request_uri;

        location / {
            return 301 https://$host$request_uri;
        }

    }

    server {
        listen 443 ssl;
        server_name j11b201.p.ssafy.io;
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/j11b201.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j11b201.p.ssafy.io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # Proxy
        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;
        proxy_set_header X-Forwarded-Proto 443;

        # Websockets
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        # 요청 제한 설정 적용
        location / {
            proxy_pass http://nginx-frontend:3000;
        }

        location /jenkins {
            proxy_pass http://jenkins:8080/jenkins;
```

```
        }

        location /jenkins-jnlp {
            proxy_pass http://jenkins:50000;
        }

        location /api {
            proxy_pass http://backend:8081;
        }

        location ~ ^/(swagger-ui|v3) {
            proxy_pass http://backend:8081;
        }

        location /.well-known/acme-challenge/ {
           root /var/www/certbot;
        }

    }

}
```

프론트 Nginx 설정 파일 (nginx.conf)

```
user nginx;
worker_processes 1;

events {
  worker_connections 1024;
}

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

  sendfile on;
  keepalive_timeout 65;

  server {
    listen 3000;
    server_name j11b201.p.ssafy.io;

    location / {
      add_header 'Access-Control-Allow-Origin' '*';
      add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
      add_header 'Access-Control-Allow-Headers' 'Origin, Content-Type, Accept, Authorization';

      root /usr/share/nginx/html;
      index index.html;
      try_files $uri $uri/ /index.html;
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
      root /usr/share/nginx/html;
    }
  }
}
```

백엔드 도커 파일 (Dockerfile)

```
FROM openjdk:17
CMD ["./gradlew", "clean", "build"]
ARG JAR_FILE=build/libs/stockolm-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

프론트 도커 파일 (Dockerfile)

```
FROM node:20.15.0-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
```

젠킨스 파이프라인

백엔드 파이프라인

```
pipeline {
    agent any

    environment {
        TARGET_BRANCH = 'backend'
        JENKINS_USER = 'stockolm'
        JENKINS_TOKEN = credentials('dbstjr9898')
        PEM_KEY = credentials('pem-key')
    }

    stages {
        stage('Cleanup Workspace') {
            steps {
                deleteDir() // 작업 공간 초기화
            }
        }
        stage('Checkout') {
            steps {
                script {
                    checkout([$class: 'GitSCM',
                              branches: [[name: '*/backend']],
                              doGenerateSubmoduleConfigurations: false,
                              extensions: [[$class: 'CleanCheckout']],
                              userRemoteConfigs: [[url: 'https://lab.ssafy.com/s11-fintech-finance-sub2/S11P22B201.git',
                              credentialsId: 'dbstjr9898']]  // credentialsId를 올바르게 설정
                    ])
                }
            }
        }
        stage('Create Directories') {
            steps {
                script {
                    sh 'mkdir -p ${WORKSPACE}/backend/stockolm/src/main/resources'
                }
            }
        }
        stage('Copy application.yml') {
            steps {
                script {
                    withCredentials([file(credentialsId: 'application-properties', variable: 'APP_PROPERTIES')]) {
                        sh 'cp $APP_PROPERTIES backend/stockolm/src/main/resources/application.yml'
                    }
                }
            }
        }
        stage('Build with Gradle') {
            steps {
                dir('backend/stockolm') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean build --no-daemon'
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    sh 'docker build -t backend -f backend/cicd/Dockerfile backend/stockolm/.'
                }
            }
        }
        stage('Deploy with Docker Compose') {
            steps {
```

```
                script {
                    sh '''
                        ssh -o StrictHostKeyChecking=no -i ${PEM_KEY} ubuntu@j11b201.p.ssafy.io '
                        cd /home/ubuntu/stockolm &&
                        docker compose stop backend &&
                        docker compose rm -f backend &&
                        docker compose up -d backend &&
                        docker image prune -f
                        '
                    '''
                }
            }
        }

        stage('Notification') {
            steps{
                echo 'jenkins notification!'
            }
            post {
                success {
                    script {
                        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()

                        mattermostSend (
                            color: 'good',
                            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env
                            endpoint: '앤드 포인트 url',
                            channel: 'Jenkins-back'
                        )
                    }
                }
                failure {
                    script {
                        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()

                        mattermostSend (
                            color: 'danger',
                            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env
                            endpoint: '앤드 포인트 urlc',
                            channel: 'Jenkins-back'
                        )
                    }
                }
            }

        }

    }

}
```