포팅 메뉴얼

사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- 빌드 도구 : Jenkins

개발 도구

- Visual Studio Code : ver 1.90.2
- IntelliJ IDEA Ultimate : 2024.1.4
- Pycharm : 2024.1.6
- DataGrip : 2024.1.4

외부 서비스

- Google OAuth
- 기상청_단기예보 API (공공데이터포털)

개발 환경

Frontend

kotlin	1.5.1
jdk	11
android sdk	30

Backend

Java	openjdk 21
Spring Boot	3.2.11
Python	3.10
Redis	7.4.0
MySQL	Ver 9.0.1 for Linux on x86_64
mongoDB	latest

Server

AWS S3
AWS EC2

Infra

Docker	27.1.1
Ubuntu	20.04.6 LTS
Jenkins	2.452.3

환경 변수

.env (중요 정보 생략)

```
## COMMON
JWT_SECRET_KEY=hbfdbnrtsnbdfgntreyteryertyertrteyertyasdasdasfdgddfndfgnrgdfndfgnrtntrnrgfdhrdfgjrdjhdb
JWT_SALT=asdfasdasfsdfsafgdfghfgjcertytyreyerertyertyrtydfasdfdzsdsdtytrfuughkijbpliupibouyihiasddassda
JWT_ISSUER=s-care.com

SWAGGER_UI_PATH=/s-care/swagger-ui
SWAGGER_API_DOCS_PATH=/s-care/api-docs

## Localhost
LOCAL_MYSQL_URL=jdbc:mysql://localhost:3306/scare?serverTimezone=UTC&useUnicode=yes&characterEncoding=U
LOCAL_MYSQL_USER=root
LOCAL_MYSQL_PASSWORD=0714

LOCAL_MONGODB_URI=mongodb://scare:scare123456@k11a408.p.ssafy.io:27017
LOCAL_MONGODB_DATABASE=scare

JWT_ACCESS_TOKEN_EXPIRATION=720000000
JWT_REFRESH_TOKEN_EXPIRATION=1440000000

## DEV
DEV_MYSQL_URL=jdbc:mysql://k11a408.p.ssafy.io:3306/scare?useUnicode=true&characterEncoding=UTF-8&useSSL
DEV_MYSQL_USER=scare
DEV_MYSQL_PASSWORD=scare123456

DEV_MONGODB_URI=mongodb://scare:scare123456@k11a408.p.ssafy.io:27017
DEV_MONGODB_DATABASE=scare

DEV_JWT_ACCESS_TOKEN_EXPIRETIME=720000000
DEV_JWT_REFRESH_TOKEN_EXPIRETIME=1440000000

SENTRY_DSN=https://9ba275c4e1b33e482fe3d98d2cfb3983@o4508220168929280.ingest.us.sentry.io/4508220171681

## FAST
FAST_API_BASE_URL=https://k11a408.p.ssafy.io/fast
STRESS_ENDPOINT=/stress/overview

## FCM
FCM_SERVICE_ACCOUNT_KEY=firebase/service-account-key.json
```

CI/CD

jenkins

기본 plugin 외에 추가 설치

- SSH Agent Plugin
- Docker plugin

credentials 설정

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	GITLAB_API_TOKEN	GitLab API token
		System	(global)	DOCKER_REPO_API	simhani1/******
		System	(global)	DOCKER_REPO_FRONT	simhani1/******
		System	(global)	EC2_SERVER_IP	EC2_SERVER_IP
		System	(global)	SSH_CREDENTIAL	ubuntu
		System	(global)	simhani1	simhani1@gmail.com/******
		System	(global)	DOCKER_USER	simhani1/******
		System	(global)	BACK_ENV	.env
		System	(global)	FRONT_ENV	.env

- GitLab Token 등록
- Docker hub 로그인 정보 등록
- Docker image push를 위한 repo 정보 등록
- SSH 접속을 위해 EC2 IP 정보와 .pem키 정보 등록
- .env 파일 등록

backend pipeline

```
pipeline {
    agent any

    environment {
        BACKEND_IMAGE_NAME = "scare-backend"
        BACKEND_CONTAINER_NAME = "backend"
        NETWORK_NAME = "scare_network"
    }

    stages {
        stage('Checkout') {
            steps {
                dir('./') {
                    git branch: 'develop/be', url: 'https://lab.ssafy.com/s11-final/S11P31A408.git', cr
                }
                script {
                    GIT_AUTHOR_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                    GIT_AUTHOR_NAME = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                }
            }
        }

        // Backend Stages
        stage('copy env') {
            steps {
                dir('backend/api') {
                    // Copy .env file
                    withCredentials([file(credentialsId: 'back_env', variable: 'ENV_FILE')]) {
                        sh 'cp $ENV_FILE .env'
                        sh 'cat .env'
                    }

                    // Copy Firebase service account key
                    withCredentials([file(credentialsId: 'fcm-service-key', variable: 'FCM_KEY_FILE')])
                        sh 'mkdir -p src/main/resources/firebase'
                        sh 'cp $FCM_KEY_FILE src/main/resources/firebase/service-account-key.json'
                        sh 'cat src/main/resources/firebase/service-account-key.json'
                    }
                }
            }
        }
        stage('Build Backend Docker Image') {
            steps {
                dir('./backend/api') {
                    sh 'docker build -t ${BACKEND_IMAGE_NAME}:${BUILD_NUMBER} .'
                }
            }
        }
        stage('Deploy Backend') {
            steps {
                script {
                        // 컨테이너 중지 및 제거
                    sh "docker stop ${BACKEND_CONTAINER_NAME} || true"
                    sh "docker rm ${BACKEND_CONTAINER_NAME} || true"
```

```
                    // 이전 이미지 정리
                    sh "docker image prune -f"

                    // 새 컨테이너 실행
                    sh """
                    docker run -d \
                        --name ${BACKEND_CONTAINER_NAME} \
                        --network ${NETWORK_NAME} \
                        --restart=unless-stopped \
                        -p 8080:8080 \
                        ${BACKEND_IMAGE_NAME}:${BUILD_NUMBER}
                    """
                }
            }
        }
    }

    post {
        success {
            script {
                mattermostSend(color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${GIT_AUTHOR_ID}(${GIT_AUTHO
                    endpoint: 'https://meeting.ssafy.com/hooks/g7a39a4zwjy4pbwh8ymqowng4w',
                    channel: 'A408_build_result')
            }
        }

        failure {
            script {
                mattermostSend(color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${GIT_AUTHOR_ID}(${GIT_A
                    endpoint: 'https://meeting.ssafy.com/hooks/g7a39a4zwjy4pbwh8ymqowng4w',
                    channel: 'A408_build_result')
            }
        }

        always {
            cleanWs()
        }
    }
}
```

fast api pipeline

```
pipeline {
    agent any

    environment {
        FAST_API_IMAGE_NAME = "fast-api"
        FAST_API_CONTAINER_NAME = "fast-api"
        NETWORK_NAME = "scare_network"
    }

    stages {
        stage('Checkout') {
            steps {
                dir('./') {
                    git branch: 'develop/fast', url: 'https://lab.ssafy.com/s11-final/S11P31A408.git',
                }
                script {
                    GIT_AUTHOR_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                    GIT_AUTHOR_NAME = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
```

```
                }
            }
        }

        stage('Build Fast Api Docker Image') {
            steps {
                dir('./backend/fast-api') {
                    sh 'docker build -t ${FAST_API_IMAGE_NAME}:${BUILD_NUMBER} .'
                }
            }
        }
        stage('Deploy AI') {
            steps {
                script {
                    sh "docker stop ${FAST_API_IMAGE_NAME} || true"
                    sh "docker rm ${FAST_API_CONTAINER_NAME} || true"

                    // 이전 이미지 정리
                    sh "docker image prune -f"

                    sh """
                    docker run -d \
                        --name ${FAST_API_CONTAINER_NAME} \
                        --network ${NETWORK_NAME} \
                        -p 8002:8000 \
                        ${FAST_API_IMAGE_NAME}:${BUILD_NUMBER}
                    """
                }
            }
        }
    }

    post {
        success {
            script {
                mattermostSend(color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${GIT_AUTHOR_ID}(${GIT_AUTHO
                    endpoint: 'https://meeting.ssafy.com/hooks/g7a39a4zwjy4pbwh8ymqowng4w',
                    channel: 'A408_build_result')
            }
        }

        failure {
            script {
                mattermostSend(color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${GIT_AUTHOR_ID}(${GIT_A
                    endpoint: 'https://meeting.ssafy.com/hooks/g7a39a4zwjy4pbwh8ymqowng4w',
                    channel: 'A408_build_result')
            }
        }

        always {
            cleanWs()
        }
    }
}
```

빌드 및 실행

docker-compose.yml

```
services:
  jenkins:
```

```
    build:
      context: ./jenkins
      dockerfile: Dockerfile
    container_name: jenkins
    user: root
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./jenkins:/var/jenkins_home
    ports:
      - '8000:8080'
    restart: on-failure
    networks:
      - scare_network

  redis:
    image: redis:latest
    container_name: redis
    hostname: redis
    command: redis-server --port 6379
    ports:
      - "6379:6379"
    volumes:
      - ./redis/redis_data:/data
      - ./redis:/usr/local/etc/redis/redis.conf
    networks:
      - scare_network

  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot
    depends_on:
      - certbot

  certbot:
    image: certbot/certbot
    volumes:
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot

  mysql:
    image: mysql:latest
    restart: always
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: scare
      MYSQL_USER: scare
      MYSQL_PASSWORD: scare123456
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - ./mysql:/var/lib/mysql
    networks:
      - scare_network
```

```
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci

  mongodb:
    image: mongo:latest
    container_name: mongodb
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: scare
      MONGO_INITDB_ROOT_PASSWORD: scare123456
    ports:
      - "27017:27017"
    volumes:
      - ./mongodb_data:/data/db
    networks:
      - scare_network

  mongo-express:
    image: mongo-express:latest
    container_name: mongo-express
    restart: always
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_SERVER: mongodb
      ME_CONFIG_MONGODB_ADMINUSERNAME: scare
      ME_CONFIG_MONGODB_ADMINPASSWORD: scare123456
      ME_CONFIG_BASICAUTH_USERNAME: scare
      ME_CONFIG_BASICAUTH_PASSWORD: scare123456
      ME_CONFIG_MONGODB_URL: mongodb://scare:scare123456@mongodb:27017
    networks:
      - scare_network
    depends_on:
      - mongodb

networks:
  scare_network:
    external: true
```

nginx.conf

```
services:
  jenkins:
    build:
      context: ./jenkins
      dockerfile: Dockerfile
    container_name: jenkins
    user: root
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./jenkins:/var/jenkins_home
    ports:
      - '8000:8080'
    restart: on-failure
    networks:
      - scare_network

  redis:
    image: redis:latest
    container_name: redis
```

```
    hostname: redis
    command: redis-server --port 6379
    ports:
      - "6379:6379"
    volumes:
      - ./redis/redis_data:/data
      - ./redis:/usr/local/etc/redis/redis.conf
    networks:
      - scare_network

  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot
    depends_on:
      - certbot

  certbot:
    image: certbot/certbot
    volumes:
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot

  mysql:
    image: mysql:latest
    restart: always
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: scare
      MYSQL_USER: scare
      MYSQL_PASSWORD: scare123456
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - ./mysql:/var/lib/mysql
    networks:
      - scare_network
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci

  mongodb:
    image: mongo:latest
    container_name: mongodb
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: scare
      MONGO_INITDB_ROOT_PASSWORD: scare123456
    ports:
      - "27017:27017"
    volumes:
      - ./mongodb_data:/data/db
    networks:
      - scare_network

  mongo-express:
```

```
    image: mongo-express:latest
    container_name: mongo-express
    restart: always
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_SERVER: mongodb
      ME_CONFIG_MONGODB_ADMINUSERNAME: scare
      ME_CONFIG_MONGODB_ADMINPASSWORD: scare123456
      ME_CONFIG_BASICAUTH_USERNAME: scare
      ME_CONFIG_BASICAUTH_PASSWORD: scare123456
      ME_CONFIG_MONGODB_URL: mongodb://scare:scare123456@mongodb:27017
    networks:
      - scare_network
    depends_on:
      - mongodb

networks:
  scare_network:
    external: true
```