포팅메뉴얼

📌 개발 환경

Android

- Android Studio `Koala 24.1.1`
- Android Gradle Plugin `8.5.0`
- targetSDK `34`
- FireBase
    - Firebase-bom `33.1.2`
    - Firebase_messaging_ktx `33.1.2`
- RXJava2 `2.2.8`
- StompProtocolAndroid `1.6.6`
- Room
    - Room-Runtime `2.6.1`
    - Room_ktx `2.6.1`
    - Room-Compiler `2.6.1`
- TensorFlow
    - TensorFlow-Lite-Support `0.1.0`
    - TensorFlow-Lite-Metadata `0.1.0`
- Retrofit2
    - Retrofit2 `2.9.0`
    - Gson `2.9.0`
    - Scalars `2.9.0`
- OKHttp3 `4.10.0`
- Glide
    - Glide `4.11.0`
    - Glide-Compiler `4.11.0`
- gms:play-services-location `21.3.0`
- com.kakao.sdk:v2-all `2.20.3`
- Coroutine `1.7.3`
- FlexBox `3.0.0`
- Dots Indicator `5.0`
- Naver Map SDK `3.18.0`
- Zxing `4.3.0`
- AndroidX
    - ConstraintLayout `2.1.4`
    - Core_ktx `1.13.1`
    - Fragment_ktx `1.8.1`
    - RecyclerView `1.3.2`
    - Activity `1.9.0`
    - ViewBinding `8.5.1`
    - AppCompat `1.7.0`
- Material `1.12.0`

BackEnd

- Java
    - IntelliJ `2024-01`
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.2`
        - Spring Data JPA `3.3.2`
        - Spring Security `6.3.1`
        - OAuth2.0 `6.3.1`

        - websocket `6.1.11`
        - Lombok `1.18.34`
    - JWT `0.11.5`
    - Swagger `3.0.0`
    - Gradle `8.8`
    - AWS S3 Bucket Cloud `2.2.6`
    - Firebase `9.3.0`

Database

- MySQL: `8.0.38`
- AWS S3

Server

- AWS EC2: `ubuntu 20.04.6 LTS`
- Docker: `27.1.1`
- Nginx: `1.18.0`
- Docker Compose : `2.29.1`
- Docker Hub

Collaboration

- GitLab
- Jira
- Notion

📌 환경 변수 설정

[Android]

build.gradle.kts(Project)

```
plugins {
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.jetbrains.kotlin.android) apply false
    id("com.google.devtools.ksp") version "1.9.0-1.0.13"
    id("com.google.gms.google-services") version "4.4.2" apply false
}
```

build.gradle.kts(Module:app)

```
import java.io.FileInputStream
import java.util.Properties

plugins {
  alias(libs.plugins.android.application)
  alias(libs.plugins.jetbrains.kotlin.android)
  id("com.google.devtools.ksp")
  id ("com.google.gms.google-services")
}
val properties = Properties()
properties.load(FileInputStream(rootProject.file("local.properties")))
val naverClientId: String = properties.getProperty("naver_client_id")
val NATIVE_APP_KEY: String = properties.getProperty("NATIVE_APP_KEY")
val BUILD_NATIVE_APP_KEY: String = properties.getProperty("BUILD_NATIVE_APP_KEY")
val SERVER_IP: String = properties.getProperty("SERVER_IP")
val LOCAL_HISTORY_DATABASE_DB: String = properties.getProperty("LOCAL_HISTORY_DATABASE_DB")
val STEP_COUNTER_DATABASE_DB: String = properties.getProperty("STEP_COUNTER_DATABASE_DB")
android {
  namespace = "com.sansantek.sansanmulmul"
  compileSdk = 34

  defaultConfig {
    applicationId = "com.sansantek.sansanmulmul"
    minSdk = 26
    targetSdk = 34
    versionCode = 1
    versionName = "1.0"

    manifestPlaceholders["naverClientId"] = naverClientId

    manifestPlaceholders["NATIVE_APP_KEY"] = NATIVE_APP_KEY
    buildConfigField("String", "NATIVE_APP_KEY", BUILD_NATIVE_APP_KEY)
    buildConfigField("String", "SERVER_IP", SERVER_IP)
    buildConfigField("String", "STEP_COUNTER_DATABASE_DB", STEP_COUNTER_DATABASE_DB)
    buildConfigField("String", "LOCATION_HISTORY_DATABASE_DB", LOCAL_HISTORY_DATABASE_DB)

    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
  }

  buildFeatures {
    buildConfig = true
    mlModelBinding = true
  }

  buildTypes {
    release {
      isMinifyEnabled = false
      proguardFiles(
        getDefaultProguardFile("proguard-android-optimize.txt"),
        "proguard-rules.pro"
      )
    }
  }
  compileOptions {
    sourceCompatibility = JavaVersion.VERSION_1_8
    targetCompatibility = JavaVersion.VERSION_1_8
  }
  kotlinOptions {
    jvmTarget = "1.8"
  }

  dataBinding {
    enable = true
  }
  viewBinding {
    enable = true
  }
}

dependencies {
  implementation(platform(libs.firebase.bom))
  implementation (libs.firebase.messaging.ktx)
  implementation ("io.reactivex.rxjava2:rxjava:2.2.8")
  implementation ("com.github.NaikSoftware:StompProtocolAndroid:1.6.6")
  implementation(libs.androidx.room.runtime)
  implementation(libs.room.ktx)
  implementation(libs.firebase.firestore.ktx)
  implementation(libs.tensorflow.lite.support)
  implementation(libs.tensorflow.lite.metadata)
  annotationProcessor(libs.room.compiler)
  ksp(libs.room.compiler)
  implementation(libs.retrofit)
  implementation(libs.converter.gson)
  implementation(libs.glide.v4110)
  annotationProcessor(libs.compiler)
  implementation(libs.threetenabp)
  implementation(libs.play.services.location)
  implementation(libs.v2.all)
  implementation(libs.kotlinx.coroutines.android)
  implementation(libs.glide)
  implementation(libs.google.flexbox)
  implementation(libs.android.segmented)
  implementation(
    fileTree(
      mapOf(
        "dir" to "libs/android-segmented-control-master",
        "include" to listOf("*.aar", "*.jar")
      )
    )
  )
  implementation(libs.dotsindicator)
  implementation(libs.androidx.fragment.ktx)
  implementation(libs.retrofit)
  implementation(libs.converter.scalars)
  implementation(libs.okhttp)
  implementation(libs.converter.gson)
  implementation(libs.converter.scalars)
  implementation(libs.logging.interceptor)
  implementation(libs.gson)
  implementation(libs.androidx.constraintlayout)
  implementation(libs.androidx.core.ktx)
  implementation(libs.androidx.recyclerview)
  implementation(libs.map.sdk)
  implementation(libs.com.journeyapps.zxing.android.embedded)
  implementation(libs.androidx.activity)
  implementation(libs.androidx.constraintlayout)
  implementation(libs.androidx.viewbinding)
  implementation(libs.androidx.core.ktx)
  implementation(libs.androidx.appcompat)
  implementation(libs.material)
  testImplementation(libs.junit)
  androidTestImplementation(libs.androidx.junit)
  androidTestImplementation(libs.androidx.espresso.core)
}
```

settings.gradle.kts

```
pluginManagement {
    plugins{
        id("com.google.devtools.ksp") apply false
    }
    repositories {
        google {
            content {
                includeGroupByRegex("com\\.android.*")
                includeGroupByRegex("com\\.google.*")
                includeGroupByRegex("androidx.*")
            }
        }
        mavenCentral()
        gradlePluginPortal()
    }
}

dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        maven("https://repository.map.naver.com/archive/maven")
        maven { url = java.net.URI("https://devrepo.kakao.com/nexus/content/groups/public/") }
        maven ("https://jitpack.io" )
    }
}

rootProject.name = "SANSANMULMUL"
include(":app")
```

local.properties

```
sdk.dir=C\:\\Users\\SSAFY\\AppData\\Local\\Android\\Sdk
naver_client_id=gotznmhn0r
NATIVE_APP_KEY=8fe371759f765d96e4f4894f6f10bbc6
BUILD_NATIVE_APP_KEY="8fe371759f765d96e4f4894f6f10bbc6"
SERVER_IP="https://i11d111.p.ssafy.io/"
LOCAL_HISTORY_DATABASE_DB="LocalHistory-database.db"
STEP_COUNTER_DATABASE_DB="StepCounter-database.db"
```

[BackEnd]

📃 application.yml

```
spring:
  main:
    web-application-type: servlet
    allow-circular-references: true
  profiles:
    active: prod

  # MySQL
  datasource:
    url: jdbc:mysql://${HOSTNAME}/${DBNAME}?useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
```

```yaml
    hikari:
      maximum-pool-size: 10  # 커넥션 풀의 최대 크기 (필요에 따라 조정)

  jpa:
    hibernate:
      ddl-auto: none  # 스키마 자동 생성 전략 (update, create, create-drop 등)
    show-sql: true  # SQL 쿼리 로깅 여부
    properties:
      hibernate:
        format_sql: true  # SQL 쿼리 포맷팅 여부

# JWT 설정
jwt:
  secret-key: ssafy+gumi01+D111+sansantek+sansanmulmul+jwt+secretkey
  access-token:
    # expiretime: 3600000 # 1시간
    expiretime: 2592000000 # 30일
  refresh-token:
    expiretime: 2592000000 # 30일

server:
  port: 8080  # 서버 포트 설정
  forward-headers-strategy: framework

logging:
  level:
    org.springframework.web: DEBUG  # 웹 요청 관련 디버그 로깅 (필요에 따라 조정)

springdoc:
  swagger-ui:
    enabled: true
    path: /swagger-ui.html
    config-url: /v3/api-docs/swagger-config
    url: /v3/api-docs
  api-docs:
    path: /v3/api-docs
openapi: 3.0.1

# aws s3 bucket
cloud:
  aws:
    s3:
      bucket: sanmulbucket
    credentials:
      access-key: ${S3_ACCESS_KEY}
      secret-key: ${S3_SECRET_KEY}
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false

# multipart 설정
spring.servlet.multipart.max-file-size: 10MB
spring.servlet.multipart.max-request-size: 10MB

# firebase
firebase:
  path: sansanmulmul-firebase-key.json

# 소셜 로그인(카카오톡) api 키
kakao:
  client_id: ${KAKAO_ID}
  client_secret:  ${KAKAO_SECRET}
  redirect_uri: https://i11d111.p.ssafy.io/user/login

# 뉴스 크롤링(네이버) api 키
naver:
  client-id: ${NAVER_ID}
  client-secret: ${NAVER_SECRET}

# 일몰 일출 api 키
sun:
  api-key: ${sun-API-KEY}

#날씨 api 키
```

```
weather:
  api-key: ${openweathermap-API-KEY}
```

📌 배포 환경 설정

0. 초기 세팅

1. EC2 접속

```
# sudo ssh -i [pem키 위치] [접속 계정]@[접속할 도메인]
$ sudo ssh -i I11D111T.pem ubuntu@i11d111.p.ssafy.io
```

2. Docker & Docker Engine 설치

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $p

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubunt
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# Docker Package 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# version check
docker --version
```

3. Docker Compose 설치

```
sudo curl -L \
"https://github.com/docker/compose/releases/download/v2.29.1/dockercompose-$(uname -s)-$(uname -m)" \
-o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose  # docker-compose에 권한 부여

# 터미널 재접속하기 #

# version check
docker-compose -v
```

1. MySQL, Jenkins 도커 이미지 pull & Docker-compose file 생성

1. 도커 이미지 pull

```
# MySQL
$ sudo docker pull mysql:8.0.38

# Jenkins
$ cd /home/ubuntu && mkdir jenkins-backup
$ sudo chown 1000 /home/ubuntu/jenkins-backup
$ docker pull jenkins/jenkins:lts

# 이미지 확인
$ docker images
```

2. Docker 볼륨 폴더 설정

   : Jenkins 백업 데이터 저장을 위한 Docker 볼륨 폴더

```
$ cd /home/ubuntu && mkdir jenkins-backup
$ sudo chown 1000 /home/ubuntu/jenkins-backup
```

3. 📃 **docker-compose.yml** file 생성

   : MySQL 과 Jenkins 도커 이미지를 올리기 위함

```
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins-container
    user: root
    ports:
      - "9090:8080"
    volumes:
      - /home/ubuntu/jenkins-backup:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/bin/docker:/usr/bin/docker
    environment:
      TZ: "Asia/Seoul"
  mysql:
    image: mysql:8.0.38
    container_name: mysql-container
    ports:
      - "3306:3306"
    volumes:
      - /mysql-volume:/var/lib/mysql
    environment:
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: "Asia/Seoul"
```

4. Docker Compose 실행

```
$ sudo docker-compose up -d

# 컨테이너 확인
$ docker ps
```

2. Nginx 설치 + SSL 인증키 발급

1. Nginx 설치

```
$ sudo apt update
$ sudo apt upgrade
# Nginx 설치
$ sudo apt install nginx

$ sudo service nginx start
$ sudo service nginx status
```

2. Encrypt, Certbot 설치

```
# Encrypt 설치
$ sudo apt-get install letsencrypt

# Certbot 설치
$ sudo apt-get install certbot python3-certbot-nginx
```

3. SSL 인증서 발급

```
# Certbot 동작 (nginx 중지하고 해야함)
$ sudo systemctl stop nginx

# Nginx 상태확인 & 80번 포트 확인
$ sudo service nginx status
$ netstat -na | grep '80.*LISTEN'

# SSL 인증서 발급 (인증서 적용 및 .pem 키 발급)
$ sudo certbot --nginx
$ sudo letsencrypt certonly --standalone -d i11d111.p.ssafy.io

# 설치한 인증서 확인 및 위치 확인
$ sudo certbot certificates

# nginx 설정 적용
# nginx 재시작
$ sudo service nginx restart
$ sudo systemctl reload nginx
```

Nginx conf 설정

- /etc/nginx/sites-available/default

  + https (ssl 키 적용)

```
server {

        server_name i11d111.p.ssafy.io;

        include /etc/nginx/conf.d/service-url.inc;

        # Redirect Swagger UI requests to user HTTPS
        location /swagger-ui/ {
                proxy_pass $service_url;
                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;
        }

        # chatting socket
        location /websocket {
                proxy_read_timeout 300s;
                proxy_connect_timeout 75s;

                proxy_pass $service_url;
                proxy_http_version 1.1;

                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_redirect off;

                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;

        }

        # BackEnd
        location / {
                proxy_pass $service_url;
                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;
        }


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i11d111.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i11d111.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


}

server {
    if ($host = i11d111.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 default_server;
        listen [::]:80 default_server;

        server_name i11d111.p.ssafy.io;
    return 404; # managed by Certbot


}
```

- /etc/nginx/conf.d/service-url.inc

```
set $service_url http://127.0.0.1:8080;
```

- /etc/nginx/nginx.conf
  모바일에서 사진 업로드 시 용량 제한 존재
  이를 해결 하기 위한 http block 에 아래의 옵션 추가

  ```
  client_max_body_size 10M;
  ```

3. Jenkins 설정

📃 젠킨스 파이프라인 스크립트

: 특정 브랜치(develop)를 추적하여 자동 배포가 진행하도록 한다. BackEnd 브랜치를 최신으로 하여 배포 테스트 시 develop으로 merge하는 방식.

```
pipeline {
    agent any

    stages {
        stage('Git Clone'){
            steps {
                git branch: 'develop',
                url: 'https://lab.ssafy.com/s11-webmobile4-sub2/S11P12D111.git',
                credentialsId: '5b742f81-450e-441d-b703-a7d94297a8d6'
            }
            post {
                failure {
                  echo 'Repository clone failure !'
                }
                success {
                  echo 'Repository clone success !'
                }
            }
        }

        stage('application.yml download') {
            steps {
                withCredentials([file(credentialsId: 'properties-credentials', variable: 'ymlFile')]) {
                    script {
                        sh 'rm /var/jenkins_home/workspace/sanmul-test/BackEnd/src/main/resources/application.yml'
                        sh 'cp $ymlFile /var/jenkins_home/workspace/sanmul-test/BackEnd/src/main/resources/application.yml'
                    }
                }
            }
        }

        stage('firebase-key.json download') {
            steps {
                withCredentials([file(credentialsId: 'firebase-key', variable: 'jsonFile')]) {
                    script {
                        sh 'rm /var/jenkins_home/workspace/sanmul-test/BackEnd/src/main/resources/sansanmulmul-firebase-key.
                        sh 'cp $jsonFile /var/jenkins_home/workspace/sanmul-test/BackEnd/src/main/resources/sansanmulmul-fir
                    }
                }
            }
        }

        stage('BE-Build'){
              steps {
                  dir('/var/jenkins_home/workspace/sanmul-test/BackEnd/'){
                      sh 'pwd'
                      sh 'ls -al'
                      sh 'chmod +x ./gradlew'
                      sh 'chmod +x ./gradlew.bat'
                      sh 'java --version'
                      sh './gradlew clean build '
                  }
              }
            }

        stage('Docker Hub Login'){
            steps{
                withCredentials([usernamePassword(credentialsId: 'DOCKER_USER', passwordVariable: 'DOCKER_PASSWORD', usernam
                    sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
                }
            }
        }
        stage('Docker Build and Push') {
```

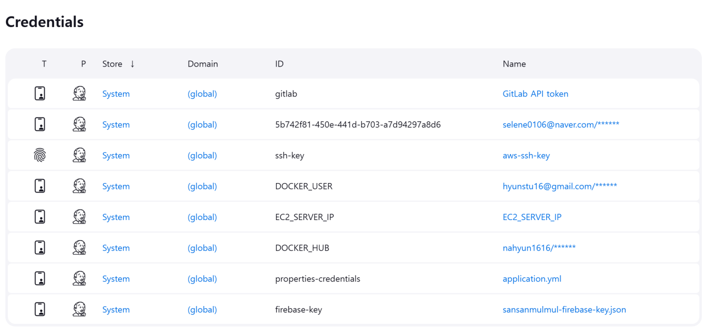
```
            steps {
                withCredentials([usernamePassword(credentialsId: 'DOCKER_HUB', passwordVariable: 'DOCKER_PROJECT', usernameV
                    sh 'cd ./BackEnd && docker build -f Dockerfile -t $DOCKER_REPO/$DOCKER_PROJECT .'
                    sh 'cd ./BackEnd && docker push $DOCKER_REPO/$DOCKER_PROJECT'
                    echo 'docker push Success!!'
                }
                echo 'docker push Success!!'
            }
        }

        stage('BE Deploy to EC2') {
            steps {
                //백엔드 이미지 땡겨오고 배포
                sshagent(credentials: ['ssh-key']) {
                  withCredentials([string(credentialsId: 'EC2_SERVER_IP', variable: 'IP')]) {
                      sh 'ssh -o StrictHostKeyChecking=no ubuntu@$IP "sudo sh deploy.sh"'
                  }
                }
            }
        }

        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: 'https://meeting.ssafy.com/hooks/ifptzq8m77rixxrwm5xz43ohpc',
                            channel: 'lucky111schedule'
                                )
                    }
                }
                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: 'https://meeting.ssafy.com/hooks/ifptzq8m77rixxrwm5xz43ohpc',
                            channel: 'lucky111schedule'
                                )
                    }
                }
            }
        }
    }
}
```

🔑Credential 관리

빌드에 필요한 env 파일들을 저장해두고 배포 시 파일을 옮겨 서버에 올린다.



Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab	GitLab API token
		System	(global)	5b742f81-450e-441d-b703-a7d94297a8d6	selene0106@naver.com/******
		System	(global)	ssh-key	aws-ssh-key
		System	(global)	DOCKER_USER	hyunstu16@gmail.com/******
		System	(global)	EC2_SERVER_IP	EC2_SERVER_IP
		System	(global)	DOCKER_HUB	nahyun1616/******
		System	(global)	properties-credentials	application.yml
		System	(global)	firebase-key	sansanmulmul-firebase-key.json

- **GitLab** : gitlab의 프로젝트를 clone 해오기위한 credential
  - `gitlab` : gitlab API 토큰
  - `5b742f81-450e. .` : gitlab ID/PW
- `ssh_key` : jenkins에서 우리의 aws ec2의 ssh에 접속하기위한 credential

- **Docker Hub** : dockerhub에 있는 백엔드 이미지를 끌어오기 위함
    - `DOCKER_USER` : 도커허브 아이디 / 비밀번호
    - `DOCKER_HUB` : 도커허브 nameSpace / 도커허브 RepositoryName
- **EC2 Server IP** : Pipeline에서 EC2 Server IP 감추기 위해
    - `EC2_SERVER_IP` : 서버주소
- **Spring 설정파일들**
  프로젝트 최종 배포시 중요한 정보들이 들어있는 Spring 설정 파일들을 gitlab에 올리지 않을 것이기 때문에 Jenkins에 미리 저장을 해주고 이걸 Backend-build 전 단계에 Backend 프로젝트의 resources 에 넣어주기 위함.
    - `properties-credentials` : SpringBoot yml파일
    - `firebase-key` : firebase관련 설정이 있는 json파일

젠킨스 플러그인 추가 설치

- GitLab
- SSH Agent
- Pipeline
- Mattermost Notification

4. 배포 위한 파일 생성

(1) SpringBoot Dockerfile 생성

📃 Dockerfile

백엔드 프로젝트 안에 백엔드 빌드를 위한 Dockerfile 작성

```
# open jdk 17 버전의 환경 구성
FROM openjdk:17-alpine

# tzdata 패키지 설치 및 타임존 설정
RUN ln -snf /usr/share/zoneinfo/Asia/Seoul /etc/localtime && echo Asia/Seoul > /etc/timezone

# build가 되는 시점에 JAR_FILE 경로에 jar파일 생성
ARG JAR_FILE=/build/libs/sansanmulmul-0.0.1-SNAPSHOT.jar

# JAR_FILE을 sanmulproject.jar로 복사
COPY ${JAR_FILE} /sanmulproject.jar

# 운영 및 개발에서 사용되는 환경 설정을 분리
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=prod", "-Duser.timezone=Asia/Seoul", "/sanmulproject.jar"]
```

(2) DockerHub에 올린 이미지 가져와서 docker compose로 서버 띄우기

📃 docker-compose.sanmul8080.yml

: docker compose로 서버를 띄우기 위한 yml 파일

```
services:
  api:
    image: nahyun1616/sanmul-be:latest
    container_name: sanmul-8080
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8080
    ports:
      - '8080:8080'
```

(3) 배포 script 작성

📃 deploy.sh

: EC2환경에서 배포하기 위한 스크립트

```
#
#deploy.sh

#0 image 갱신
sudo docker compose -p sanmul-8080 -f /home/ubuntu/docker-compose.sanmul8080.yml pull

#1
echo "8080 컨테이너 실행"
sudo docker compose -p sanmul-8080 -f /home/ubuntu/docker-compose.sanmul8080.yml up -d --force-recreate
```

```
# 2
for cnt in `seq 1 10`;
do
    echo "서버 응답 확인하는중~(${cnt}/10)";
    UP=$(curl -s http://127.0.0.1:8080/server-check)
    if [ "${UP}" != "OK" ]; then
        sleep 10
        continue
    else
        break
    fi
done

if [ $cnt -eq 10 ]; then
    echo "서버에 문제가 있어요..."
    exit 1
fi

# 3
sudo nginx -s reload
echo "Deploy Completed!!"

# 4
sudo docker image prune -f
```

(3) 참고: [EC2내 파일구조]

```
$ pwd
/home/ubuntu
$ tree
├── deploy.sh
├── docker-compose.sanmul8080.yml
├── docker-compose.yml
└── jenkins-backup
```

📁 jenkins-backup

: Jenkins 백업 데이터 저장을 위한 Docker 볼륨 폴더

📃 docker-compose.yml

: MySQL 과 Jenkins 도커 이미지를 올리기 위함

📃 docker-compose.sanmul8080.yml

: docker compose로 서버를 띄우기 위한 yml 파일

📃 deploy.sh

: EC2환경에서 배포하기 위한 스크립트

[젠킨스 컨테이너]

```
ubuntu@ip-172-26-14-75:~$ docker exec -it jenkins-container bash
root@e11ed70f5ec0:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@e11ed70f5ec0:/# cd var/jenkins_home/workspace/sanmul-test/BackEnd
root@e11ed70f5ec0:/var/jenkins_home/workspace/sanmul-test/BackEnd# ls
Dockerfile  build  build.gradle  gradle  gradlew  gradlew.bat  settings.gradle  src
```

📃 Dockerfile

: 백엔드 프로젝트 안에 백엔드 빌드를 위한 Dockerfile 작성

📌 프로젝트에 사용된 외부 서비스

- 카카오

  : OAuth 로그인을 위해 사용
- Firebase

  : FCM 알림을 위해 사용
- AWS S3

  : 사진 데이터 저장을 위해 사용
- 산림청 공공데이터 API

  https://www.forest.go.kr/kfsweb/kfi/kfs/trail/trailInformation.do?pblicDataId=PBD0000041&tabs=3&mn=NKFS_06_08_02&subTitle=등산로정보
- OpenWeather API

https://openweathermap.org/api

- 일몰일출 API

  https://www.data.go.kr/data/15012688/openapi.do