배포할 수 있도록 정리한 문서

사용한 제품들의 설정 값 및 버전

- 안드로이드 클라이언트
  - Kotlin: 1.8.10
  - JVM: 17
  - MinSDK: 23
  - TargetSDK: 34
  - 외부 라이브러리 설정 값, 버전 (MavenRepsitory 등에서 다운로드 한 의존성 `lib.version.toml`)

```
[versions]
agp = "8.7.2"
kotlin = "1.8.10"
compose = "1.4.4"
coreKtx = "1.13.1"
junit = "4.13.2"
junitVersion = "1.2.1"
espressoCore = "3.6.1"
lifecycleRuntimeKtx = "2.6.2"
activityCompose = "1.8.1"
composeBom = "2024.02.00"

# ----------- 의존성 추가 ------------
ksp                = "1.8.10-1.0.9"
room               = "2.5.2"
okhttp             = "4.11.0"
retrofit           = "2.9.0"
gson               = "2.10.1"
hilt               = "2.48"
coil               = "2.4.0"

foundationLayoutAndroid = "1.7.2"
datastore = "1.1.4"

navCompose = "2.8.5"
hiltCompose = "1.2.0"
accompanist = "0.34.0"

threetenabp = "1.4.9"

kakao-sdk = "2.21.0"

cameraCore = "1.4.2"

splashscreen = "1.0.1"

lottieCompose = "6.6.6"


paging = "3.2.1"
paging-compose = "1.0.0-alpha20"

[libraries]
```

```
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-core", version.ref = "espressoC
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-ktx", version.ref = "lifec
androidx-activity-compose = { group = "androidx.activity", name = "activity-compose", version.ref = "activityCom
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref = "composeBom"
androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-tooling-preview" }
androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-manifest" }
androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-junit4" }
androidx-material3 = { group = "androidx.compose.material3", name = "material3" }

# Room
androidx-room-runtime       = { group = "androidx.room",       name = "room-runtime",   version.ref = "room" }
androidx-room-ktx           = { group = "androidx.room",       name = "room-ktx",       version.ref = "room" }
androidx-room-compiler      = { group = "androidx.room",       name = "room-compiler",  version.ref = "room" }

# Room Paging
androidx-room-paging = { group = "androidx.room", name = "room-paging", version.ref = "room" }
androidx-paging-runtime = { group = "androidx.paging", name = "paging-runtime", version.ref = "paging" }
androidx-paging-compose = { group = "androidx.paging", name = "paging-compose", version.ref = "paging-comp

# OkHttp
okhttp                    = { group = "com.squareup.okhttp3", name = "okhttp",              version.ref = "okhttp" }
okhttp-logging-interceptor  = { group = "com.squareup.okhttp3", name = "logging-interceptor",  version.ref = "o

# Retrofit & Gson

retrofit                  = { group = "com.squareup.retrofit2", name = "retrofit",       version.ref = "retrofit" }
retrofit-converter-gson     = { group = "com.squareup.retrofit2", name = "converter-gson",  version.ref = "retrofit"
gson                      = { group = "com.google.code.gson",   name = "gson",           version.ref = "gson" }

# Hilt
hilt-android              = { group = "com.google.dagger",     name = "hilt-android",    version.ref = "hilt" }
hilt-compiler             = { group = "com.google.dagger",     name = "hilt-compiler",   version.ref = "hilt" }

# Coil
coil                      = { group = "io.coil-kt",          name = "coil",          version.ref = "coil" }
coil-compose              = { group = "io.coil-kt",          name = "coil-compose",    version.ref = "coil" }

# libs
androidx-foundation = { group = "androidx.compose.foundation", name = "foundation", version.ref = "compose" }
androidx-foundation-layout = { group = "androidx.compose.foundation", name = "foundation-layout", version.ref =
androidx-datastore-core-android = { group = "androidx.datastore", name = "datastore-core-android", version.ref
androidx-datastore-preferences = { group = "androidx.datastore", name = "datastore-preferences", version.ref =

# Navigation
androidx-navigation-compose = { group = "androidx.navigation", name = "navigation-compose", version.ref = "na
androidx-hilt-navigation-compose = { group = "androidx.hilt", name = "hilt-navigation-compose", version.ref = "hi

# Pager
accompanist-pager = { group = "com.google.accompanist", name = "accompanist-pager", version.ref = "accomp
accompanist-pager-indicators = { group = "com.google.accompanist", name = "accompanist-pager-indicators", v

threetenabp = { group = "com.jakewharton.threetenabp", name = "threetenabp", version.ref = "threetenabp" }
```

```
# Kakao
kakao-user = { module = "com.kakao.sdk:v2-user", version.ref = "kakao-sdk" }

# CameraX
androidx-camera-camera2 = { module = "androidx.camera:camera-camera2", version.ref = "cameraCore" }
androidx-camera-core = { module = "androidx.camera:camera-core", version.ref = "cameraCore" }
androidx-camera-extensions = { module = "androidx.camera:camera-extensions", version.ref = "cameraCore" }
androidx-camera-view = { module = "androidx.camera:camera-view", version.ref = "cameraCore" }
androidx-camera-lifecycle = { module = "androidx.camera:camera-lifecycle", version.ref = "cameraCore" }

# SystemUiController
accompanist-systemuicontroller = { module = "com.google.accompanist:accompanist-systemuicontroller", versio

# Splash
androidx-core-splashscreen = { group = "androidx.core", name = "core-splashscreen", version.ref = "splashscree

# Test
androidx-test-core = { group = "androidx.test", name = "core", version = "1.6.1" }
androidx-test-runner = { group = "androidx.test", name = "runner", version = "1.6.2" }
androidx-test-rules = { group = "androidx.test", name = "rules", version = "1.6.1" }

# Lottie
lottie-compose = { module = "com.airbnb.android:lottie-compose", version.ref = "lottieCompose" }


[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
hilt-android   = { id = "com.google.dagger.hilt.android", version.ref = "hilt" }
ksp = { id = "com.google.devtools.ksp", version.ref = "ksp"}
```

- AI
  - Airflow : 2.7.2
  - MLflow : 2.22.0
  - Fastapi : 0.115.6
  - python : 3.10
- Backend
  - SpringBoot: 3.3.10
  - Intellij: 21.0.5+8-b631.30 amd64
  - Gradle: 8.13
  - JVM: 17.0.3
- DB
  - MySQL: 8.0.42
  - Redis: 7.4.2

빌드 시 사용되는 환경변수

- 안드로이드
  - Build.config에서 사용되는 `local.properties` {BASE URL} 값

```
BASE_URL=https://k12s203.p.ssafy.io/api/
```

- 백엔드
  - application-prod.yml

```yaml
spring:
  application:
    name: FoodOn

  config:
    import: optional:file:.env.prod[.properties]

  servlet:
    multipart:
      max-file-size: 10MB       # 파일 1개당 최대 크기

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${DATASOURCE_URL}
    username: ${DATASOURCE_USERNAME}
    password: ${DATASOURCE_PASSWORD}
    hikari:
      maximum-pool-size: 16

data:
  redis:
    host: ${REDIS_HOST}
    port: ${REDIS_PORT}
    password: ${REDIS_PASSWORD}

jpa:
  hibernate:
    ddl-auto: update
  properties:
    hibernate:
      show_sql: true
      format_sql: true
      dialect: org.hibernate.dialect.MySQLDialect
  open-in-view: false


  auth:
    jwt:
      secret-key: ${JWT_SECRET_KEY}
      access-token-expiry: ${JWT_ACCESS_TOKEN_EXPIRY}
      refresh-token-expiry: ${JWT_REFRESH_TOKEN_EXPIRY}
      super-token-expiry: 2592000000   # 30일 기한: 30 * 24 * 60 * 60 * 1000

  tomcat:
    mbeanregistry:
      enabled: true

cloud:
  aws:
    s3:
      bucket: ${CLOUD_AWS_S3_BUCKET}
      upload-path: ${S3_UPLOAD_PATH}
      base-url: ${S3_BASE_URL}
```

```yaml
    region:
      static: ${CLOUD_AWS_REGION_STATIC}
      auto: false
    stack:
      auto: false
    credentials:
      accessKey: ${CLOUD_AWS_CREDENTIALS_ACCESS_KEY}
      secretKey: ${CLOUD_AWS_CREDENTIALS_SECRET_KEY}

meal-detect-ai-model:
  url: http://k12s203.p.ssafy.io:8000/ai/detect2

file:
  upload-dir: ${java.io.tmpdir}/files/

server:
  forward-headers-strategy: NATIVE
  servlet:
    context-path: /api/v1
  encoding:
    enabled: true
    charset: UTF-8
    force: true

kakao:
  api:
    url: https://kapi.kakao.com


logging:
  level:
    root: INFO
    org:
      springframework: DEBUG
    jdbc:
      sqlonly: DEBUG
      resultsettable: DEBUG
      audit: TRACE
      resultset: TRACE
      connection: OFF
    org.hibernate.SQL: DEBUG
    org.hibernate.type.descriptor.sql: TRACE

image:
  default:
    meal-url: ${S3_BASE_URL}${S3_UPLOAD_PATH}/0b4b1c06-dbf3-4259-882f-5b10d03657aa_20250520215957

management:
  endpoints:
    web:
      exposure:
        include: prometheus, metrics
  endpoint:
    prometheus:
      enabled: true
```

- .env 파일

```
DATASOURCE_URL=jdbc:mysql://k12s203.p.ssafy.io:3306/foodon?useSSL=false&useUnicode=true&serverTimezo
DATASOURCE_USERNAME=foodon
DATASOURCE_PASSWORD=foodon0229

REDIS_HOST=k12s203.p.ssafy.io
REDIS_PORT=6379
REDIS_PASSWORD=gozldgkwlakfkdlrjtemfdks

JWT_SECRET_KEY=705de2f7200a358b5702ddc0cfd5588c63859b80fed53740b0e66ed59c9965e0e57a04cb76e
JWT_ACCESS_TOKEN_EXPIRY=3600000
JWT_REFRESH_TOKEN_EXPIRY=1209600000

CLOUD_AWS_CREDENTIALS_ACCESS_KEY=AKIAXBSHCAAYOFCQUR7K
CLOUD_AWS_CREDENTIALS_SECRET_KEY=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
CLOUD_AWS_S3_BUCKET=foodon-bucket
CLOUD_AWS_REGION_STATIC=ap-northeast-2

S3_UPLOAD_PATH=images
S3_BASE_URL=https://foodon-bucket.s3.ap-northeast-2.amazonaws.com/
```

배포 시 특이사항

[AI 서버]

- jenkins

```
pipeline {
    agent any

    options {
        disableConcurrentBuilds()
    }

    environment {
        JAVA_HOME = '/usr/lib/jvm/java-17-openjdk-amd64'
        PATH = "${JAVA_HOME}/bin:${PATH}"

        DOCKER_BACK_IMAGE = "hyeinlee/foodon-spring"
        DOCKER_TAG = "latest"
        DOCKER_HUB_CREDENTIALS_ID = 'docker-hub-credentials'

        SPRING_CONTAINER = "foodon-spring"
    }

    tools {
        gradle 'gradle'
    }

    stages {

        stage ("Checkout") {
            steps {
                script {
                    deleteDir()

                    sh 'rm -rf .git'
```

```
                sh 'git init'
                sh 'git remote add origin https://lab.ssafy.com/s12-final/S12P31S203.git'
                sh 'git remote prune origin || true'
                sh 'git fetch --force --prune || true'

                git(
                    branch: "develop",
                    credentialsId: "l1253",
                    url: "https://lab.ssafy.com/s12-final/S12P31S203.git"
                )
            }
        }
    }

    stage ("Spring Gradle Build") {
        steps {
            dir("./FoodOn_backend") {
                sh '''
                    ls -la
                    chmod +x gradlew
                    ./gradlew clean --no-build-cache bootJar
                '''
            }
        }
    }

    stage ("Spring Docker image Push") {
        steps {
            dir("./FoodOn_backend") {
                script {
                    // 기존 spring 이미지 사용하는 컨테이너 먼저 중지 및 제거
                    sh """
                        docker ps -a --filter "ancestor=${DOCKER_BACK_IMAGE}:${DOCKER_TAG}" --format "{{.ID}}" | xarg
                        docker ps -a --filter "ancestor=${DOCKER_BACK_IMAGE}:${DOCKER_TAG}" --format "{{.ID}}" | xarg
                        docker images -q ${DOCKER_BACK_IMAGE}:${DOCKER_TAG} | xargs -r docker rmi -f
                    """

                    def jarFile = sh(
                        script: "[ -f build/libs/*.jar ] && ls build/libs/*.jar | head -n 1 || echo ''",
                        returnStdout: true
                    ).trim()

                    withCredentials([usernamePassword(credentialsId: env.DOCKER_HUB_CREDENTIALS_ID, usernameVar
                        sh """
                            docker build --pull --no-cache -t ${DOCKER_BACK_IMAGE}:${DOCKER_TAG} --build-arg JAR_FILE
                            echo "${DOCKER_PASS}" | docker login -u "${DOCKER_USER}" --password-stdin
                            docker push ${DOCKER_BACK_IMAGE}:${DOCKER_TAG}
                        """
                    }
                }
            }
        }
    }

    stage('Deploy') {
        steps {
            withCredentials([file(credentialsId: 'backend_env', variable: 'ENV_FILE')]) {
                sh '''
```

```
                docker stop ${SPRING_CONTAINER} || true
                docker rm ${SPRING_CONTAINER} || true
                docker run -d --env-file $ENV_FILE -e TZ=Asia/Seoul -e JAVA_OPTS="-Duser.timezone=Asia/Seoul" -v
            '''
        }
    }
  }
}

post {
    always {
        echo "Pipeline Completed"
    }
    success {
        echo "Pipeline Executed Successfully"
    }
    failure {
        echo "Pipeline Failed"
    }
  }
}
```

- docker-compose.yml

```
version: '3.8'

services:
  fastapi:
    container_name: fastapi_server
    build:
      context: ./fastapi
    ports:
      - "8000:8000"
    restart: always
    environment:
      - TZ=Asia/Seoul
    volumes:
      - shared-model-volume:/shared_model

  airflow-webserver:
    container_name: airflow_webserver
    build:
      context: ./airflow
    restart: always
    depends_on:
      - airflow-scheduler
      - airflow-postgres
    env_file:
      - .env
    environment:
      - AIRFLOW__CORE__EXECUTOR=LocalExecutor
      - AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@airflow-postgres:5432/a
      - AIRFLOW__CORE__LOAD_EXAMPLES=False
      - AIRFLOW__WEBSERVER__EXPOSE_CONFIG=True
      - AIRFLOW__WEBSERVER__WEB_SERVER_PORT=8787
      - AIRFLOW__WEBSERVER__SECRET_KEY=s3cr3t_k3y_4irfl0w_1234567890
    ports:
      - "8787:8787"
```

```
    command: webserver
    volumes:
      - shared-model-volume:/shared_model
      - /home/ubuntu/dataset:/train/dataset
  airflow-scheduler:
    container_name: airflow_scheduler
    build:
      context: ./airflow
    restart: always
    depends_on:
      - airflow-postgres
    env_file:
      - .env
    environment:
      - AIRFLOW__CORE__EXECUTOR=LocalExecutor
      - AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@airflow-postgres:5432/a
      - AIRFLOW__WEBSERVER__SECRET_KEY=s3cr3t_k3y_4irfl0w_1234567890
    command: scheduler
    volumes:
      - shared-model-volume:/shared_model
      - /home/ubuntu/dataset:/dataset

  airflow-postgres:
    container_name: airflow_postgres
    image: postgres:13
    restart: always
    environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow
    volumes:
      - postgres-db-volume:/var/lib/postgresql/data

  mlflow:
    container_name: mlflow_server
    image: ghcr.io/mlflow/mlflow
    command: mlflow ui --host 0.0.0.0 --port 5000
    ports:
      - "5000:5000"
    environment:
      - MLFLOW_TRACKING_URI=http://mlflow:5000
    volumes:
      - /home/ubuntu/mlflow:/mlflow/mlruns
      - shared-model-volume:/shared_model

  volumes:
    postgres-db-volume:
    shared-model-volume:
```

[Backend 서버]

- Jenkins 스크립트

```
pipeline {
    agent any

    options {
```

```
    disableConcurrentBuilds()
  }

  environment {
    JAVA_HOME = '/usr/lib/jvm/java-17-openjdk-amd64'
    PATH = "${JAVA_HOME}/bin:${PATH}"

    DOCKER_BACK_IMAGE = "hyeinlee/foodon-spring"
    DOCKER_TAG = "latest"
    DOCKER_HUB_CREDENTIALS_ID = 'docker-hub-credentials'

    SPRING_CONTAINER = "foodon-spring"
  }

  tools {
    gradle 'gradle'
  }

  stages {

    stage ("Checkout") {
      steps {
        script {
          deleteDir()

          sh 'rm -rf .git'
          sh 'git init'
          sh 'git remote add origin https://lab.ssafy.com/s12-final/S12P31S203.git'
          sh 'git remote prune origin || true'
          sh 'git fetch --force --prune || true'

          git(
            branch: "develop",
            credentialsId: "l1253",
            url: "https://lab.ssafy.com/s12-final/S12P31S203.git"
          )
        }
      }
    }

    stage ("Spring Gradle Build") {
      steps {
        dir("./FoodOn_backend") {
          sh '''
            ls -la
            chmod +x gradlew
            ./gradlew clean --no-build-cache bootJar
          '''
        }
      }
    }

    stage ("Spring Docker image Push") {
      steps {
        dir("./FoodOn_backend") {
          script {
            // 기존 spring 이미지 사용하는 컨테이너 먼저 중지 및 제거
            sh """
```

```
                docker ps -a --filter "ancestor=${DOCKER_BACK_IMAGE}:${DOCKER_TAG}" --format "{{.ID}}" | xarg
                docker ps -a --filter "ancestor=${DOCKER_BACK_IMAGE}:${DOCKER_TAG}" --format "{{.ID}}" | xarg
                docker images -q ${DOCKER_BACK_IMAGE}:${DOCKER_TAG} | xargs -r docker rmi -f
            """

            def jarFile = sh(
                script: "[ -f build/libs/*.jar ] && ls build/libs/*.jar | head -n 1 || echo ''",
                returnStdout: true
            ).trim()

            withCredentials([usernamePassword(credentialsId: env.DOCKER_HUB_CREDENTIALS_ID, usernameVar
                sh """
                    docker build --pull --no-cache -t ${DOCKER_BACK_IMAGE}:${DOCKER_TAG} --build-arg JAR_FILE
                    echo "${DOCKER_PASS}" | docker login -u "${DOCKER_USER}" --password-stdin
                    docker push ${DOCKER_BACK_IMAGE}:${DOCKER_TAG}
                """
            }
        }
    }

    stage('Deploy') {
        steps {
            withCredentials([file(credentialsId: 'backend_env', variable: 'ENV_FILE')]) {
                sh '''
                    docker stop ${SPRING_CONTAINER} || true
                    docker rm ${SPRING_CONTAINER} || true
                    docker run -d --env-file $ENV_FILE -e TZ=Asia/Seoul -e JAVA_OPTS="-Duser.timezone=Asia/Seoul" -v
                '''
            }
        }
    }
}

post {
    always {
        echo "Pipeline Completed"
    }
    success {
        echo "Pipeline Executed Successfully"
    }
    failure {
        echo "Pipeline Failed"
    }
}
```

- docker-compose.yml

```
services:
  mysql:
    image: mysql:8.0
    container_name: foodon-mysql
    ports:
      - "${MYSQL_PORT}:3306"
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}   # MySQL root 비밀번호
```

```
    MYSQL_DATABASE: ${MYSQL_DATABASE}          # 초기 생성 데이터베이스 이름
    MYSQL_USER: ${MYSQL_USER}                  # 사용자 이름
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}          # 사용자 비밀번호
    TZ: ${TIMEZONE}                            # 컨테이너의 시간대 설정
  command:
    - --character-set-server=utf8mb4           # 문자 세트
    - --collation-server=utf8mb4_unicode_ci    # 문자 정렬 기준
  volumes:
    - mysql-data:/var/lib/mysql                # 데이터 영구 저장 위치

redis:
  image: redis:7.4.2-alpine
  container_name: foodon-redis
  ports:
    - "${REDIS_PORT}:6379"
  command: ["redis-server", "--requirepass", "${REDIS_PASSWORD}"]
  volumes:
    - redis-data:/data                         # Redis 데이터 저장

volumes:
  mysql-data:
  redis-data:
```

- SpringBoot Dockerfile

```
FROM openjdk:17-jdk-alpine

WORKDIR /app

COPY build/libs/foodon-0.0.1-SNAPSHOT.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar", "--spring.profiles.active=prod"]
```

[Nginx]

- 직접 EC2 호스트 시스템에 세팅

```
docker run -d -p 80:80 -p 443:443 -v ./data/files/:/usr/share/nginx/files --name ddakdae-nginx nginx
docker exec -it ddakdae-nginx bash
apt-get update
apt-get install certbot // certbot 설치
apt-get install python3-certbot-nginx // 웹서버 플러그인 설치
certbot --nginx -d <도메인 이름>
```

- default.conf 내용

```
server {
    server_name  k12s203.p.ssafy.io;

    #access_log  /var/log/nginx/host.access.log  main;
    client_max_body_size 10M;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
```

```
    }

    # Spring Boot API (8080)
    location /api/ {
        rewrite ^/api/(.*)$ /api/v1/$1 break;
        proxy_pass http://k12s203.p.ssafy.io:8080/;
        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;
    }

    # FASI API (8000)
    location /ai/ {
        proxy_pass http://k12s203.p.ssafy.io:8000/;
        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;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k12s203.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/k12s203.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 ($host = k12s203.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen       80;
    listen  [::]:80;
    server_name  k12s203.p.ssafy.io;
    return 404; # managed by Certbot


}
```