1. 배포문서

1. 빌드 환경

Frontend

- `React` `18.2.0`
- `Node.js` `18.16.1 LTS`
- `Recoil`

Backend

- `OpenJDK Azul Zulu` `17.0.7`
- `Spring Boot` `3.0.10`
- `Spring Security`
- `Spinrg Data JPA`
- `QueryDSL`
- `Komoran` `3.3.4`
- `Django` `4.2.5`

DB

- `MariaDB`
- `Redis` `7.2.1`
- `MongoDB` `3.6.8`

Infra

- `Ubuntu` `20.04 LTS`
- `AWS EC2`
- `AWS S3`
- `GitLab`
- `Jenkins` `2.414.1`
- `Nginx` `1.18.0`

2. 환경 변수 형태

Frontend

- .env.development

```
REACT_APP_KAKAO_API_KEY=YOUR_KAKAO_API_KEY
REACT_APP_SERVER_URL=https://j9b310.p.ssafy.io
REACT_APP_BASE_URL = http://localhost:3000
```

- .env.production

```
REACT_APP_KAKAO_API_KEY=YOUR_KAKAO_API_KEY
REACT_APP_SERVER_URL=https://j9b310.p.ssafy.io
REACT_APP_BASE_URL = https://j9b310.p.ssafy.io
```

Backend

- application-MariaDB

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: MariaDB 서버주소
    username: MariaDB username
    password: MariaDB password
```

- application-MongoDB

```
spring:
  data:
    mongodb:
      host: MongoDB 서버주소
      port: MongoDB 서버포트
      username: MongoDB username
      password: MongoDB password
      authentication-database: mownimoney
      database: mwonimoney
      uri: MongoDB 주소
```

- application-Oauth

```
spring:
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: 구글 클라이언트 id
            client-secret: 구글 클라이언트 secret
            redirect-uri: http://j9b310.p.ssafy.io/api/login/oauth2/code/google
            scope:
              - email
              - profile
          kakao:
            client-id: 카카오 클라이언트 id
            client-secret: 카카오 클라이언트 secret
```

```
            scope:
              - profile_nickname
              - account_email
            client-name: Kakao
            authorization-grant-type: authorization_code
            redirect-uri: http://j9b310.p.ssafy.io/api/login/oauth2/code/kakao
            client-authentication-method: client_secret_post

        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

app:
  oauth2:
    authorizedRedirectUris: http://j9b310.p.ssafy.io/oauth/redirect

jwt:
  key: JWT 키
```

- application-Redis

```
spring:
  data:
    redis:
      lettuce:
        pool:
          max-active: 5
          max-idle: 5
          min-idle: 2
      host: 서버주소
      port: 서버포트
```

- application-S3

```
spring:
  data:
    couchbase:
      bucket-name: mwonimoney
cloud:
  aws:
    stack:
      auto: false
    region:
      static: ap-northeast-2
    credentials:
      secret-key: S3 시크릿 키
      access-key: S3 엑세스 키


aws.s3.image.goal.url: 이미지가 저장되는 URL
```

- secret.json

```
"OPENAI_API_KEY": 오픈AI API 키
```

Admin

- settings.py

```
"""
Django settings for mwonimoney project.

Generated by 'django-admin startproject' using Django 4.0.3.

For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.0/ref/settings/
"""
import os
from pathlib import Path
from .my_settings import mySECRET_KEY, myDATABASES

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = mySECRET_KEY

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
# DEBUG = False

ALLOWED_HOSTS = ["j9b310.p.ssafy.io", "127.0.0.1"]
CSRF_TRUSTED_ORIGINS = ["https://j9b310.p.ssafy.io", "https://127.0.0.1"]


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',
    'mwonimoney',
    'chatbot'
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'mwonimoney.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
```

```
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'mwonimoney.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': BASE_DIR / 'db.sqlite3',
#     }
# }

DATABASES = myDATABASES

# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = '/static_django/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')


# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

CORS_ALLOW_ALL_ORIGINS = False
CORS_ALLOW_CREDENTIALS = True
CORS_ALLOW_METHODS = (
    'GET',
    'POST',
    'PUT',
    'PATCH',
```

```
    'DELETE',
    'OPTIONS',
)

CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
    "https://j9b310.p.ssafy.io",
]
```

- my_settings.py

```
mySECRET_KEY = 'django-insecure-vce0jr_sl(8t84axzii-zi+!yigles=up!9et52=h02yv4a0k1'

myDATABASES = {
  'default' : {
          'ENGINE' : 'django.db.backends.mysql', # 벡엔드 엔진
          'NAME' : 'mwonimoneydb', # 'mysql'의 이름을 가진 데이터베이스
          'USER' : 'ssafy-b310', # 계정
          'PASSWORD' : 'j9b310p_kplyjc', #rootpassword로 지정할 숫자(6번에 나와있음)
          'HOST' : 'j9b310.p.ssafy.io',
          'PORT' : '50000'
    }
}
```

3. 배포 시 특이사항

Frontend:

1. `npm install --force` 실행
2. `npm run build` 실행
3. `nginx -g daemon off` 실행

Backend:

1. `./gradlew clean bootJar` 실행
2. `cd ./build/libs` 실행
3. `java -Djava.security.egd=file:/dev/./urandom -jar /app.jar` 실행

Admin:

1. `pip install --no-cache-dir -r requirements.txt` 실행
2. `python manage.py makemigrations` 실행
3. `python manage.py migrate` 실행
4. `python manage.py runserver 0.0.0.0:8000` 실행

Docker Container:

1. docker-compose.yml 작성
2. `sudo docker-compose up -d` 실행

4. DB 접속 정보 및 ERD에 활용되는 주요 계정 및 프로퍼티 정의

1. maria
   - 아이디: `ssafy-b310`
   - 비밀번호: `j9b310p_kplyjc`
2. mongo
   - 아이디: `ssafy-b310`
   - 비밀번호: `j9b310p_kplyjc`

```
services:
        mongodb:
                image: mongo
                volumes:
                        - ~/data:/data/db
                container_name: "mongodb"
                ports:
                        - "27017:27017"
                environment:
                        MONGO_INITDB_ROOT_USERNAME: ssafy-b310
                        MONGO_INITDB_ROOT_PASSWORD: j9b310p_kplyjc
                command: ["--bind_ip", "0.0.0.0"]
                network_mode: host
```

3. postgres
   - URL : jdbc:postgresql://i9b108.p.ssafy.io:5432/AQuh
   - 주소 : http://i9b108.p.ssafy.io:5432
   - 아이디 : postgres
   - 비번 : 2208

5. nginx 설정

```
server{
        listen 80;
        server_name j9b310.p.ssafy.io;
        return 301 https://$host$request_uri;
}

server{

        listen 443 ssl;
        server_name j9b310.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j9b310.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j9b310.p.ssafy.io/privkey.pem;

        location / {
                proxy_pass http://127.0.0.1:3000;
                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;
        }

        location /swagger-ui {
                proxy_pass http://127.0.0.1:8080;
                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;
        }

        location /admin {
                proxy_pass http://127.0.0.1:8000;
                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;
        }

        location /django {
                proxy_pass http://127.0.0.1:8000;
                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;
        }


        location /static_django/ {
                proxy_pass http://127.0.0.1:8000;
                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;
                alias /S09P22B310/admin/static;
        }

        location /api {
                client_max_body_size 50M;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
                proxy_buffering off;
                proxy_pass http://127.0.0.1:8080;
                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;
        }
}
```

6. Jenkins 설정

- django.jenkinsFile

```
pipeline {
    agent any

    environment {
        CONTAINER_NAME = "mwonimoney-admin-container"
        IMAGE_NAME = "mwonimoney-admin-image"
    }

    stages {
        stage('Checkout') {
            steps {
                //Jenkins의 SCM (소스 코드 관리) 플러그인을 사용하여 Git 저장소로부터 소스 코드를 가져오는 역할
                checkout scm
                sh 'echo "git clone완료"'
                sh 'echo "현재 디렉토리 경로"'
                sh 'pwd'
            }
        }
```

```
        stage('Docker Delete') {
            steps {
                script {
                    try{
                        sh 'echo "Docker Delete Start"'
                        // 컨테이너 존재 시 삭제
                        sh "docker stop ${CONTAINER_NAME}"
                        sh "docker rm -f ${CONTAINER_NAME}"
                    }catch (Exception e){
                        echo "Docker container ${CONTAINER_NAME} does not exist. skip"
                    }
                    try{
                        // 이미지 존재 시 삭제
                        sh "docker image rm ${IMAGE_NAME}"
                    }catch (Exception e){
                        echo "Docker image ${IMAGE_NAME} does not exist. skip"
                    }
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    // 현재 디렉토리의 파일 및 디렉토리 목록 출력
                    sh 'ls'
                    dir('admin') {
                        // 변경된 디렉토리에서 명령어 실행
                        sh 'ls'
                        sh 'echo "Hello from the changed directory"'
                        // de
                        sh "docker build -t ${IMAGE_NAME} -f Dockerfile ."

                        sh "docker images"

                        sh 'echo "images build 성공!"'

                        dir('static'){
                            sh 'ls'
                        }
                    }
                }
            }
        }

        stage('Deploy') {
            steps {
                sh "docker run -d --name ${CONTAINER_NAME} -p 8000:8000 ${IMAGE_NAME}"
                sh "docker ps"
            }
        }

    }
}
```

- react.jenkinsFile

```
pipeline {
    agent any

    environment {
        CONTAINER_NAME = "mwonimoney-front-container"
        IMAGE_NAME = "mwonimoney-front-image"
    }

    stages {
        stage('Checkout') {
            steps {
```

```
                //Jenkins의 SCM (소스 코드 관리) 플러그인을 사용하여 Git 저장소로부터 소스 코드를 가져오는 역할
                checkout scm
                sh 'echo "git clone완료"'
                sh 'echo "현재 디렉토리 경로"'
                sh 'pwd'
            }
        }

        stage('Docker Delete') {
            steps {
                script {
                    try{
                        sh 'echo "Docker Delete Start"'
                        sh "docker ps"
                        // 컨테이너 존재 시 삭제
                        sh "docker stop ${CONTAINER_NAME}"
                        sh "docker rm -f ${CONTAINER_NAME}"
                    }catch (Exception e){
                        echo "Docker container ${CONTAINER_NAME} does not exist. skip"
                    }
                    try{
                        // 이미지 존재 시 삭제
                        sh "docker image rm ${IMAGE_NAME}"
                    }catch (Exception e){
                        echo "Docker image ${IMAGE_NAME} does not exist. skip"
                    }
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    // 현재 디렉토리의 파일 및 디렉토리 목록 출력
                    sh 'ls'
                    dir('frontend-web') {
                        // 변경된 디렉토리에서 명령어 실행
                        sh 'ls'
                        sh 'echo "Hello from the changed directory"'
                        // de
                        sh "docker build -t ${IMAGE_NAME} -f Dockerfile ."

                        sh "docker images"

                        sh 'echo "images build 성공!"'
                    }
                }
            }
        }

        stage('Deploy') {
            steps {
                sh 'echo "현재 올라간 컨테이너들"'
                sh "docker ps"
                sh "docker run -d --name ${CONTAINER_NAME} -p 3000:80 ${IMAGE_NAME}"
                sh "docker ps"
            }
        }

    }
}
```

- springboot.jenkinsFile

```
pipeline {
    agent any

    environment {
```

```
        CONTAINER_NAME = "mwonimoney-back-container"
        IMAGE_NAME = "mwonimoney-back-image"
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

        stage('Docker Delete') {
            steps {
                script {
                    try{
                        sh 'echo "Docker Delete Start"'
                        sh "docker stop ${CONTAINER_NAME}"
                        sh "docker rm -f ${CONTAINER_NAME}"
                    }catch (Exception e){
                        echo "Docker container ${CONTAINER_NAME} does not exist. skip"
                    }
                    try{
                        // 이미지 존재 시 삭제
                        sh "docker image rm ${IMAGE_NAME}"
                    }catch (Exception e){
                        echo "Docker image ${IMAGE_NAME} does not exist. skip"
                    }
                }
            }
        }

        stage('Dockerizing'){
            steps{
                dir('backend'){

                    sh "echo '파일 구조 확인'"
                    sh "ls"
                    dir('build'){
                        sh "ls"
                    }
                    // de
                    sh "docker build -t ${IMAGE_NAME} -f Dockerfile ."

                    sh "docker images"

                    sh 'echo "images build 성공!"'
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    dir('backend') {
                        sh 'chmod +x gradlew'
                        sh './gradlew clean build'
                        sh 'ls -al ./build'
                    }
                }
            }
        }

        stage('Deploy') {
            steps {
                sh "docker run --name ${CONTAINER_NAME} -d -p 8080:8080 ${IMAGE_NAME}"
                sh "docker ps"

            }
        }
    }
}
```

7. 포트 설정

```
sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
443                        ALLOW       Anywhere
3000                       ALLOW       Anywhere
80                         ALLOW       Anywhere
27017                      ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
3000 (v6)                  ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
27017 (v6)                 ALLOW       Anywhere (v6)

172.17.0.3 8080/tcp        ALLOW FWD   Anywhere                  # allow jenkinscicd 8080/tcp bridge
172.18.0.2 6379/tcp        ALLOW FWD   Anywhere                  # allow local-redis 6379/tcp redis-network
172.17.0.2 3306/tcp        ALLOW FWD   Anywhere                  # allow mariadb 3306/tcp bridge
```

8. Redis 설정

- Redis 이미지 받기

```
docker pull redis:alpine
```

- 도커 네트워크 생성[디폴트값]

```
docker network create redis-network
```

- 로컬 - docker 간 6379 포트 개방

```
docker run --name redis -p 6379:6379 --network redis-network -v /redis_temp:/data -d redis:alpine redis-server --appendonly yes
```