



포팅 매뉴얼

발자취 포팅 매뉴얼

SSAFY 6기 자율 프로젝트 서울 4반 7팀 발자취

김수연 김도현 김윤하 박주미 이성재 한지희



목차

- 1 프로젝트 기술 스택
- 2 환경 설정 및 프로퍼티 파일
- 3 빌드 및 배포 방법
- 4 외부 서비스



프로젝트 기술 스택

- **Front-end**
 - React 18.0.0
 - NextJS 12.1.5
 - next-pwa 5.5.2
 - Recoil 3.4.0
 - TypeScript 4.6.3
 - Visual Studio Code 1.64.2
 - HTML5
 - CSS
 - JavaScript (ES6)
 - Axios (API 통신 라이브러리)
 - Draft.js (리액트 Rich Text Editor 프레임워크)
 - Recoil (리액트를 위한 상태관리 라이브러리)
 - styled-components (CSS in JS 라이브러리)
 - framer-motion (애니메이션 라이브러리)
 - react-calendar (리액트 캘린더 라이브러리)
 - Day.js (Javascript date 유틸리티 라이브러리)
 - react-chart-js-2 (리액트 차트 라이브러리)
 - swiper (리액트 스와이퍼 라이브러리)
 - react-toastify (리액트 토스트 라이브러리)
 - socket-io.client (실시간 채팅 라이브러리)
- **Back-end**
 - Java (Open-JDK 1.8.0_192)

- SpringBoot 2.6.7
- Spring Data JPA
- Hibernate
- Lombok
- Spring Boot Gradle 7.4.1
- QueryDSL
- Firebase Cloud Messaging
- NestJS
- Socket.IO
- IntelliJ IDEA Community Edition 2021.3.1
- **DB**
 - MySQL 5.7
- **운영체제, 서버**
 - Window10
 - Ubuntu 20.04 LTS
 - Jenkins 2.332.2
 - nginx
 - Docker 20.10.7
 - Certbot
 - AWS EC2 (Ubuntu 20.04 LTS)
 - AWS S3
 - AWS CloudFront
- **형상 관리**
 - GitLab
 - Sourcetree
- **이슈 관리**
 - Jira
 - Mattermost
- **커뮤니케이션**
 - Notion, Webex

환경 설정 및 프로퍼티 파일

Frontend

- frontend/.env.production에서 환경 변수 설정

```
BASE_URL=[HOST URL]
KAKAO_CLIENT_ID=[카카오 client ID]
KAKAO_REDIRECT_URI=[카카오 redirect URI]
NEXT_PUBLIC_GOOGLE_ANALYTICS=[구글 analytics 추적 코드]
NEXT_PUBLIC_CHAT_URL=[채팅 서버 URL]
```

Backend

- backend/src/main/resources/application-prod.properties에서 DB/JWT 환경 변수 설정

```
# Context Path Config
server.servlet.contextPath=/api

# Database
spring.datasource.url=[데이터베이스 URL]
spring.datasource.username=[유저 이름]
spring.datasource.password=[비밀번호]
spring.datasource.driver-class-name=[JDBC 드라이버]
...

# JWT
jwt.header=[JWT 헤더]
jwt.secret=[JWT 비밀키]
jwt.token-validity-in-seconds=[토큰 유효 기간]
```

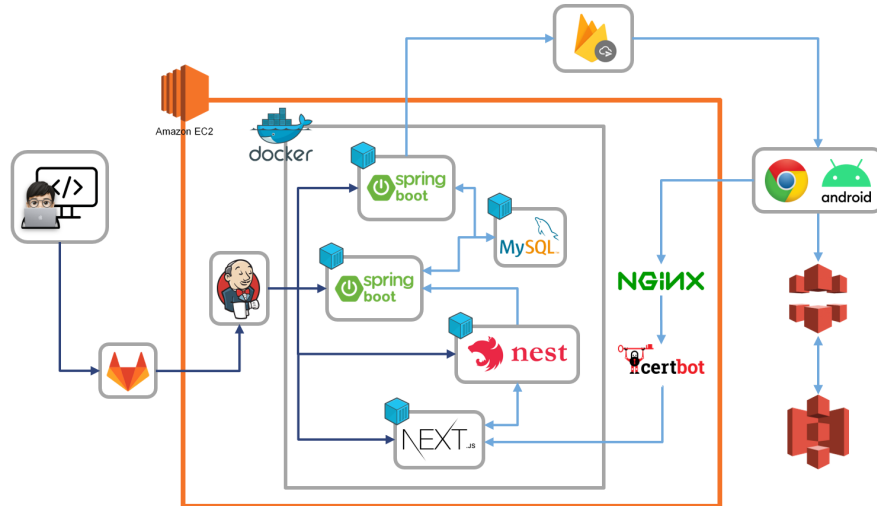
- backend/src/main/resources/application.yml에서 AWS/Kakao/Firebase 인증 정보 설정

```
# AWS Config
cloud:
  aws:
    credentials:      # AWS 초기화를 위한 인증 정보
      accessKey:
      secretKey:
    s3:               # S3 bucket 설정
      bucket: baljc   # bucket 이름
      folder:         # 폴더 구조
        profileImage: profile_image/
        categoryImage: category_image/
        boardImage: board_image/
        boardCategoryImage: board_category_image/
    cloudFront:       # CloudFront 도메인 설정
      domain:         # 도메인 URL
    region:
      static: ap-northeast-2 # 지역 설정
    stack:
      auto: false      # CloudFormation 구성
# Kakao Config
kakao:               # Kakao 로그인 API를 위한 설정
  clientId:
  redirectUri:
# Firebase Config
firebase:
  credential:        # Firebase 초기화를 위한 인증 정보
    type:
    project_id:
    private_key_id:
    private_key:
    client_email:
    client_id:
    auth_uri:
    token_uri:
    auth_provider_x509_cert_url:
    client_x509_cert_url:
  scope:             # Google API 범위 설정
```

📢 빌드 및 배포 방법

본 빌드 및 배포 과정은 Ubuntu(Linux)를 기반으로 작성되었습니다. Docker를 활용하여 빌드된 파일을 Docker image로 만들고 container로 실행시키는 과정으로 빌드와 배포를 진행합니다.

0. 서버 아키텍처



1. Docker 설치

- Ubuntu(Linux) and etc.

- Ubuntu : <https://docs.docker.com/engine/install/ubuntu/>

```
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Docker 설치
$ sudo apt-get install -y docker.io
# 사용자를 Docker 그룹에 추가
$ sudo usermod -a -G docker $USER
```

- etc : <https://docs.docker.com/engine/install/>

2. Nginx 설치 및 설정

- Nginx 설치

```
# Nginx 설치
$ sudo apt install nginx

# 실행 및 상태 확인
sudo service start nginx
sudo service status nginx
```

- Nginx 설정

/etc/nginx/sites-available/default 파일을 아래와 같이 변경

```

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    server_name baljc.com;

    return 301 https://$server_name$request_uri;

    location / {
        try_files $uri $uri/ =404;
    }
}

server {
    # client body 데이터 크기 제한
    client_max_body_size 50M;

    # SSL 인증 설정
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    ssl_certificate /etc/letsencrypt/live/baljc.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/baljc.com/privkey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!MD5;

    # frontend server
    location / {
        proxy_pass http://localhost:3000;
    }

    # backend server
    location /api {
        proxy_pass http://localhost:8080;
    }

    # chat server
    location ^~ /socket {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

```

nginx를 재실행

```
sudo service restart nginx
```

2. Git Clone

```
$ git clone https://lab.ssafy.com/s06-final/S06P31A407.git
```

3. 빌드 및 배포 (Dokerize)

• Frontend

frontend 디렉토리로 이동 후 docker image 생성, docker container 실행

```

$ cd ./frontend
$ docker build -t frontend-image .
$ docker run -d -p 3000:3000 --rm --name frontend-container frontend-image

```

• Backend

backend 디렉토리로 이동 후 docker image 생성, docker container 실행

```
$ cd ./backend
$ ./gradlew clean build
$ docker build -t backend-image .
$ docker run -d -e TZ=Asia/Seoul -p 8080:8080 --rm --name backend-container backend-image
```

• Scheduler

scheduler 디렉토리로 이동 후 docker image 생성, docker container 실행

```
$ cd ./scheduler
$ ./gradlew clean build
$ docker build -t scheduler-image .
$ docker run -d -e TZ=Asia/Seoul -p 8081:8081 --rm --name scheduler-container scheduler-image
```

• Chat

chat 디렉토리로 이동 후 docker image 생성, docker container 실행

```
$ cd ./chat
$ docker build -t chat-image .
$ docker run -d -p 5000:5000 --rm --name chat-container chat-image
```

⚙️ 외부 서비스

🔑 Amazon S3

<https://zzang9ha.tistory.com/358>

<https://artiicy.tistory.com/16>

- 개요 : 객체를 저장할 수 있는 저장소를 제공
- 버킷 생성 및 활용
 - S3에서 버킷 만들기
 - 버킷 이름 입력 / 리전 선택
 - 퍼블릭 액세스 권한 설정
 - 퍼블릭 정책 생성 및 적용
 - 액세스 키 생성
 - IAM > 사용자 > 사용자 추가 > 기존 정책 직접 연결 > AmazonS3FullAccess 체크 > 사용자 만들기 > 액세스 키 기록
- backend의 application.yml에서 AWS 초기화를 위한 인증 정보에 액세스 키 설정

🔑 Amazon CloudFront

<https://overcome-the-limits.tistory.com/379>

- 개요 : 파일의 캐싱을 위한 CDN 서비스
- 배포 생성 및 도메인 확인
 - CloudFront에서 배포 생성
 - 원본 도메인과 이름 설정, ex) S3 도메인
 - OAI 사용 및 새 OAI 생성
 - 뷰어 프로토콜 정책 Redirect HTTP to HTTPS 선택
 - 캐시 정책 설정

- 배포 후 활성화된 배포 도메인 확인
- backend의 application.yml에서 CloudFront 도메인 설정

카카오 로그인 API

<https://developers.kakao.com/docs/latest/ko/kakaologin/common>
<https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api>
<https://overcome-the-limits.tistory.com/379>

- 개요 : 카카오 계정과 애플리케이션을 연결하는 기능을 제공
- 애플리케이션 추가 및 활용
 - 애플리케이션 추가 : 로그인 후 내 애플리케이션 > 애플리케이션 추가하기
 - 도메인 등록 : 내 애플리케이션 > 앱 설정 > 플랫폼 > Web 플랫폼 등록
 - Redirect URI 등록 : 도메인 등록 후 아래의 링크를 통해 등록
 - 활성화 설정 : 상태 ON
 - 앱 키 사용 : 내 애플리케이션 > 앱 설정 > 앱 키
- frontend의 .env.production에서 client ID와 redirect URI를 설정
- backend의 application.yml에서 client ID와 redirect URI를 설정

Firebase Cloud Messaging

<https://firebase.google.com/docs/cloud-messaging?hl=ko>
<https://firebase.google.com/docs/admin/setup?hl=ko#java>

- 개요 : 등록된 기기에 메시지를 전송하는 기능을 제공
- 프로젝트 생성 및 활용
 - 프로젝트 생성 : 로그인 후 프로젝트 추가
 - 앱 추가 : 내 애플리케이션 > 설정 > 프로젝트 설정 > 일반 > 내 앱에서 앱추가 > SDK 설정 및 구성
 - frontend의 pages/_app.tsx와 public/firebase-messaging-sw.js의 firebaseConfig에 설정
 - 비공개 키 생성 : 프로젝트 설정 > 서비스 계정 > Firebase Admin SDK 사용을 위한 비공개 키 생성
 - backend의 application.yml에서 비공개 키 필드 값 설정