





포팅 매뉴얼

 작성일 2023.11.15

 작성자 류민지

 제목 DEVELOG 포팅 매뉴얼

개발 환경

형상 관리

- Gitlab

이슈 관리

- Jira

Communication

- Mattermost
- Webex
- Notion

UI/UX

- Figma

IDE

- IntelliJ IDEA
- VSCode
- Pycharm

Database

- MySQL 8.0.33
- Redis 7.0.12

BackEnd

FrontEnd

- JAVA JDK 11
- SpringBoot 2.7.17
 - Gradle 8.2.1
 - Spring Data JPA
 - Lombok
 - Springdoc Swagger
 - Spring Security
 - JWT
 - QueryDSL
- Python 3.8.10
- FastAPI 0.104.1
- Node 18.17.1
- NPM 9.6.7
- Vite + Typescript + SWC
- React 18.2.0
 - React-query
 - Framer-motion
 - Redux
 - Axios
 - react-router-dom 6.0
- 설치

```
npm install react-router-dom
npm install --save-dev @types/react-router-dom
npm install --save-dev @types/react
```

API 및 클라우드

- 카카오로그인 API
- chatGPT
- AWS

기타 편의 툴

- Postman 10.15.8

Infra

- AWS EC2
 - Ubuntu 20.04
 - Docker 24
- Nginx 1.25.1
- Jenkins

EC2 사용 포트

포트	설명
----	----

22	SSH
80	HTTP
443	HTTPS
3000	React
5012	MySQL
9090	Spring Boot
8000	FastApi
8080	Jenkins
6379	Redis

Nginx 설정

1. /etc/nginx/sites-available/default 파일에 도메인 설정

```
server {
    listen 80;
    server_name www.develop.co.kr develop.co.kr;
    return 301 https://$host$request_uri;
}
```

2. Nginx 리로드하여 default파일 설정 적용

```
# default 파일이 정상적인지 테스트
$ sudo nginx -t

# 리로드
$ sudo service nginx reload
```

3. SSL 인증서 발급(Certbot으로 Nginx에 SSL 인증서 자동 적용)

```
# Let's Encrypt 설치
$ sudo apt-get install letsencrypt

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

# Certbot 동작
sudo certbot --nginx
```

4. SSL 인증서 적용 후 최종적으로 설정한 /etc/nginx/sites-available/default 파일

```
server {

    root /var/www/html;

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

    server_name www.develop.co.kr develop.co.kr;

    location / {
        proxy_pass http://www.develop.co.kr:3000;
    }

    location /api {
        proxy_pass http://www.develop.co.kr:9090;
    }

    location /fastapi {
        proxy_pass http://www.develop.co.kr:8000;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/develop.co.kr/fullchain.pem; # managed by C
    ertbot
    ssl_certificate_key /etc/letsencrypt/live/develop.co.kr/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 {
    listen 80;
```

```

server_name www.develop.co.kr develop.co.kr;
return 301 https://$host$request_uri;

}

server{
    location / {
        try_files $uri /index.html;
    }
}

```

Docker 컨테이너

1. Jenkins

- 컨테이너 실행

```
vi /home/ubuntu/docker-compose.yml
```

```

version: '3'

services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "8080:8080"
    user: root

```

```
sudo docker compose up -d
```

- 필요한 플러그인 설치

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ Build Timeout	✓ Credentials Binding
✓ Timestampers	✓ Workspace Cleanup	✓ Ant	🔄 Gradle
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	🔄 Pipeline: Stage View
🔄 Git	🔄 SSH Build Agents	🔄 Matrix Authorization Strategy	🔄 PAM Authentication
🔄 LDAP	🔄 Email Extension	🔄 Mailer	

- gitlab 관련
 - docker 관련
 - generic webhook trigger
 - ssh agent plugin
 - ssh server
 - node 관련 (NodeJS Plugin)
- **Item 파이프라인 스크립트**

1. develop-back

```

pipeline {
    agent any

    stages {
        stage('Git clone') {
            steps {
                git branch: 'back-dev', credentialsId: 'jenkins_token', url: 'https://lab.ssafy.com/s09-final/S09P31B210.git'
            }
        }
        stage('Build') {
            steps {
                dir("./Backend/develop") {
                    sh "chmod +x ./gradlew"
                    sh "./gradlew clean build"
                }
            }
        }
        stage('Deployment') {
            steps {

```

```

        sshagent(credentials: ['ssh_key']) {
            sh '''
                ssh -o StrictHostKeyChecking=no ubuntu@k9b210.p.ssafy.io
                scp /var/jenkins_home/workspace/develog-back/Backend/develog/build/libs/develog-0.0.1-SNAPSHOT.jar ubuntu@k9b210.p.ssafy.io:/home/ubuntu/develog-back
                ssh -t ubuntu@k9b210.p.ssafy.io ./deploy_be.sh
            '''
        }
        timeout(time: 30, unit: 'SECONDS') {
        }
    }
}
}
}

```

2. develog-front

```

pipeline {
    agent any

    stages {
        stage('Git clone') {
            steps {
                git branch: 'front-dev', credentialsId: 'jenkins_token', url: 'https://lab.ssafy.com/s09-final/S09P31B210.git'
                echo "Current workspace: ${workspace}"
            }
        }

        stage('Deployment') {
            steps {
                sshagent(credentials: ['ssh_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@k9b210.p.ssafy.io
                        "rm -rf /home/ubuntu/develog-front/node_modules"
                        scp -r /var/jenkins_home/workspace/develog-front/frontend/develog/* ubuntu@k9b210.p.ssafy.io:/home/ubuntu/develog-front
                        ssh -t ubuntu@k9b210.p.ssafy.io ./deploy_fe.sh
                    '''
                }
            }
        }
    }
}

```

3. fast-api

```

pipeline {
    agent any

    stages {
        stage('Git clone') {
            steps {
                git branch: 'fast-dev', credentialsId: 'jenkins_token', url: 'https://lab.ssafy.com/s09-final/S09P31B210.git'
            }
        }
        // stage('Build') {
        //     steps {
        //         dir("./fastapi/develog") {

        //         }
        //     }
        // }
        stage('Deployment') {
            steps {
                sshagent(credentials: ['ssh_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@k9b210.p.ssafy.io
                        scp /var/jenkins_home/workspace/fast-api/fastapi/* ubuntu@k9b210.p.ssafy.io:/home/ubuntu/fastapi
                        ssh -t ubuntu@k9b210.p.ssafy.io ./deploy_fastapi.sh
                    '''
                }
                timeout(time: 30, unit: 'SECONDS') {
                }
            }
        }
    }
}

```

• Dockerfile & deploy.sh 생성

- 소유자에게 파일 실행 권한 필요 `-rwxrwx-r--`

1. BE

a. Dockerfile

```

FROM adoptopenjdk/openjdk11

ENV PUBLIC_IP=k9b210.p.ssafy.io
ENV DB_PORT=5012

```



```

ENV DB_NAME=develog
ENV DB_USER_NAME=develog
ENV DB_USER_PASSWORD=develog

ENV REDIS_HOST=k9b210.p.ssafy.io
ENV REDIS_PORT=6379
ENV REDIS_PASSWORD=redis

ENV jwt.secretKey=lksdffdsjlfssdfwfsdf2332
ENV jwt.access.expiration=3600000000
ENV jwt.refresh.expiration=1800000000

ENV oauth2.kakao.client-id=825e4ea403efb983279fe719acfe85c6
# ENV REDIRECT_URL=http://localhost:3000/login/oauth2/code/kakao
ENV REDIRECT_URL=https://develog.co.kr/login/oauth2/code/kakao

ENV S3_URL=https://s3.ap-northeast-2.amazonaws.com/develog.bucket/
ENV S3_BUCKET=develog.bucket
ENV CREDENTIALS_ACCESS_KEY=AKIA2R442DPQ2NMACDSE
ENV CREDENTIALS_SECRET_KEY=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
ENV REGION_STATIC=ap-northeast-2

COPY ./develog-0.0.1-SNAPSHOT.jar /develog-0.0.1-SNAPSHOT.jar
CMD ["java", "-jar", "develog-0.0.1-SNAPSHOT.jar"]

```

b. deploy_be.sh

```

cd develog-back
sudo docker compose down
sudo docker compose up -d --build
yes | sudo docker system prune

```

2. FE

a. Dockerfile

```

FROM node:18.17.1-alpine

WORKDIR /frontend

COPY . ./

EXPOSE 3000

RUN npm install --silent

```

```
CMD ["npm", "run", "dev"]
```

b. deploy_fe.sh

```
cd developeg-front
sudo docker compose down
sudo docker compose up -d --build

yes | sudo docker system prune

cp -r ./ignore ./src
```

c. git clone 받아와지는 디렉토리에 **ignore 디렉토리 존재해야 함.**

그 아래 두개의 파일 존재해야 함.

```
export const KakaoKey = "825e4ea403efb983279fe719acfe85c6";
// export const redirectUri = "http://localhost:3000";
export const redirectUri = "https://develog.co.kr";
```

```
export const SERVERURL = "https://www.develog.co.kr/api";
export const LOCALURL = "https://localhost:3000/api";
```

3. BE/fastAPI

a. Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.8.10

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install -r requirements.txt

# Make port 8000 available to the world outside this container
EXPOSE 8000
```

```
# Define environment variable
ENV NAME FastAPIApp

# Run app.py when the container launches
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

b. deploy_fastapi.sh

```
cd fastapi
sudo docker compose down
sudo docker compose up -d --build
yes | sudo docker system prune
```

c. git clone 받아와지는 디렉토리에 `.env` 파일 추가

```
OPENAI_KEY=sk-xPEd18QrniR081bv0ZK7T3B1bkFJnHUeWsyDQLcqPyEYxfCN

# 네이버 쪽
CLIENT_ID=b8ixpunz91
CLIENT_SECRET=dQb1RwU1ozVhT1im5pN6oIhH6mVUPUAi0qDaVC52
```

DB

MySQL

```
PUBLIC_IP=k9b210.p.ssafy.io
DB_PORT=5012
DB_NAME=develog
DB_USER_NAME=develog
DB_USER_PASSWORD=develog
```

Redis

```
REDIS_HOST=k9b210.p.ssafy.io
REDIS_PORT=6379
REDIS_PASSWORD=redis
```

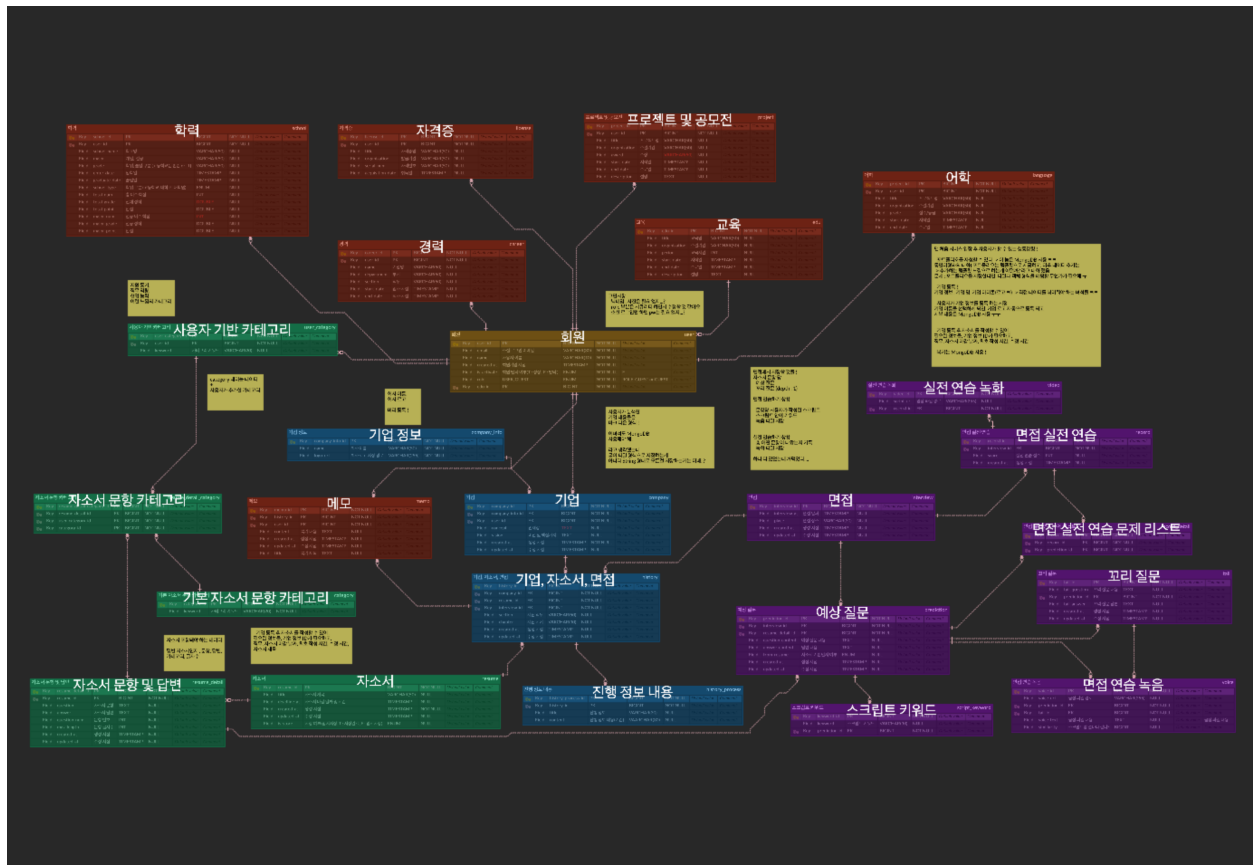
설치

```
#Redis
sudo docker pull redis
sudo docker run --name bookwave-redis -p 6379:6379 -d redis
```

#MySQL

```
sudo apt-get install mysql-server  
sudo service mysql restart
```

ERD



소셜 로그인 Redirect URI(카카오)

- <https://develog.co.kr/login/oauth2/code/kakao>