|  |
| --- |
|  |

|  |
| --- |
| **체리씨** |
| **포팅 매뉴얼** |
|  |
|  |
|  |
|  |
|  |



목차

[I. 개요 2](#_Toc158738222)

[1. 사용 도구 2](#_Toc158738223)

[2. 개발환경 2](#_Toc158738224)

[A. 프론트 엔드 2](#_Toc158738225)

[B. 백엔드 2](#_Toc158738226)

[C. 서버 2](#_Toc158738227)

[3. 외부 서비스 2](#_Toc158738228)

[A. 프론트 엔드 2](#_Toc158738229)

[B. 백엔드 2](#_Toc158738230)

[4. 환경변수 2](#_Toc158738231)

[A. 프론트 엔드 2](#_Toc158738232)

[B. 백엔드 2](#_Toc158738233)

[II. 빌드 2](#_Toc158738234)

[1. 환경변수 형태 2](#_Toc158738235)

[2. 빌드하기 4](#_Toc158738236)

[3. 배포하기 4](#_Toc158738237)

[4. 서비스 이용 방법 6](#_Toc158738238)

[가) 카카오 페이 6](#_Toc158738239)

[나) Firebase 실시간 알림 6](#_Toc158738240)

[다) Google Cloud Platform 7](#_Toc158738241)

# 사용 도구

**이슈 관리** : JIRA

**형상 관리** : Gitlab

**커뮤니케이션 및 문서화** : Notion, Mattermost

**디자인** : Figma

**CI/CD** : Docker, Jenkins

# 개발환경

* 1. 프론트 엔드

Node.js : 20.10.0

React : 18.2.0  
Visual Studio Code : 1.85.1

* 1. 백엔드

Java : 17.0.9

**MySql** : 8.0.36

**Intellij** : 2023.3.2

**ffmpeg** : 6.1.1

* 1. 서버

**OS** : AWS EC2 Ubuntu 20.04.6 LTS

**MySql** : 8.0.36

**Docker** : 25.0.1

# 외부 서비스

* 1. 프론트 엔드

OpenAI API : GPT API를 사용하였습니다.

.env.local에 키값을 저장하였고 .env.local은 gitignore로 공유저장소에 올리지 않고 따로 저장하고 있습니다.

Kakao API : 카카오 로그인과 카카오 알림 기능을 사용하였습니다.

.env.local에 키값을 저장하였고 .env.local은 gitignore로 공유저장소에 올리지 않고 따로 저장하고 있습니다.

* 1. 백엔드

S3 : 동영상 클립들을 저장하고 불러오는 저장소로 사용하였습니다.

application-secrets.yml에 접속 링크를 저장하였고 application-secrets.yml은gitignore로 공유저장소에 올리지 않고 따로 저장하고 있습니다.

# 환경변수

* 1. 프론트 엔드

.env.local : 노출되면 안되는 중요정보들을 아래와 같이 작성하여 gitignore로 공유 저장소에 올리지 않고 따로 개발환경에서 불러와 사용하였습니다

VITE\_APP\_SOCKET\_URL=

VITE\_APP\_KAKAO\_JAVASCRIPT\_KEY=

VITE\_APP\_KAKAO\_REST\_API\_KEY=

VITE\_APP\_BACKEND\_URL=

VITE\_APP\_GPT\_API\_KEY=

* 1. 백엔드

application.yml : 노출되어도 괜찮은 정보들은 아래와 같이 작성하여 저장하였습니다.

spring:  
 profiles:  
 include:  
 - secrets  
 datasource:  
 *#hikari Datasource 설정* hikari:  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 jdbc-url: jdbc:mysql://${MYSQL\_HOST:localhost}:3306/cherish?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8  
 username: ${db.username}  
 password: ${db.password}  
 pool-name: Hikari Connection Pool  
 *# 타임리프 설정* thymeleaf:  
 prefix: classpath:templates/  
 check-template-location: true  
 suffix: .html  
 mode: HTML5  
 cache: false  
 order: 0  
 *# 동영상 blob 파일 저장 용량 설정* servlet:  
 multipart:  
 maxFileSize: 100MB *# 파일 하나당 용량* maxRequestSize: 200MB *# 요청 하나당 용량  
  
# log 수준 debug로 설정*logging:  
 level:  
 root:  
 zaxxer:  
 hikari: DEBUG  
 com:  
 ssafy:  
 cherish: DEBUG  
  
*# 로컬 스토리지 저장을 위한 자체 경로 정보  
# EC2에 맞춰서 수정 필요*custom:  
 path:  
*# mac 기준  
# String ffmpegPath=System.getProperty("user.home")+"/ffmpeg/6.1.1\_3/bin";* ffmpeg: "C:\\Program Files\\ffmpeg-6.1.1-essentials\_build\\bin\\"  
 monthly-video: "C:\\Users\\SSAFY\\Documents\\cherish\_video\\monthly-video\\"  
 clip: "C:\\Users\\SSAFY\\Documents\\cherish\_video\\clip\\"  
 answer: "C:\\Users\\SSAFY\\Documents\\cherish\_video\\answer\\"  
  
server:  
 servlet:  
 context-path: /api

application-secrets.yml : 노출되면 안되는 중요정보들을 아래와 같이 작성하여 gitignore로 공유 저장소에 올리지 않고 따로 개발환경에서 불러와 사용하였습니다

db:  
 username:   
 password:   
  
cloud:  
 aws:  
 credentials:  
 access-key:   
 secret-key:

s3:  
 bucket:   
 region:  
 static:   
 stack:  
 auto:

# 빌드하기

* 1. 프론트엔드

npm I –-legacy-peer-deps

npm run build

* 1. 백엔드

gradlew clean build

# 배포하기

1. Nginx : default.conf

upstream app {

server cherish-backend:8080; # WAS 컨테이너의 이름

}

server {

listen 80;

listen 443 ssl;

server\_name i10d103.p.ssafy.io;

root /usr/share/nginx/html; # 정적 파일이 위치한 디렉토리 지정

location / {

try\_files $uri $uri/ /index.html; # 정적 파일 제공

}

location /api {

proxy\_pass http://app/api; # API 요청을 WAS로 전달

}

location /api/socket {

proxy\_pass http://app/api/socket; # 웹소켓 요청을 WAS로 전달

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

location /swagger-ui {

proxy\_pass http://app/swagger-ui; # Swagger UI를 위한 프록시 패스

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;

}

# SSL 설정

ssl\_certificate /etc/letsencrypt/live/i10d103.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/i10d103.p.ssafy.io/privkey.pem;

# HTTP 요청을 HTTPS로 리다이렉트

# if ($scheme = http) {

# return 301 https://$server\_name$request\_uri;

# }

# request size 변경

client\_max\_body\_size 100M;

# 웹소켓 시간 제한 변경

proxy\_read\_timeout 21600000; # 6 \* 60 \* 60 \* 1000

proxy\_send\_timeout 21600000; # 6 \* 60 \* 60 \* 1000

}

1. Jenkins : pipeline script

node {

try {

stage('Clone') {

// git branch: 'develop', credentialsId: 'jenkins', url: 'https://lab.ssafy.com/s10-webmobile1-sub2/S10P12D103.git'

git branch: 'develop', credentialsId: 'jenkins', url: 'https://lab.ssafy.com/s10-webmobile1-sub2/S10P12D103.git'

dir('backend/cherish/src/main/resources') {

// spring-boot-secrets credential 사용

withCredentials([file(credentialsId: 'spring-boot-secrets', variable: 'SECRETS\_FILE')

]) {

// SECRETS\_FILE 변수를 사용하여 파일을 프로젝트 디렉토리로 복사

sh 'chmod -R 755 .'

sh 'cp $SECRETS\_FILE .'

}

}

dir('frontend/cherish') {

withCredentials([file(credentialsId: 'front\_secrets', variable: 'SECRETS\_FILE')

]) {

// SECRETS\_FILE 변수를 사용하여 파일을 프로젝트 디렉토리로 복사

sh 'chmod -R 755 .'

sh 'cp $SECRETS\_FILE .'

}

}

}

stage('Build') {

dir('backend/cherish') {

sh 'chmod +x gradlew'

dir('src/main/resources') {

sh "sed -i 's/\${MYSQL\_HOST:localhost}/172.26.8.234/g' application.yml"

// ffmpeg

sh "sed -i 's|C:\\\\\\\\Program Files\\\\\\\\ffmpeg-6.1.1-essentials\_build\\\\\\\\bin\\\\\\\\|/usr/bin/|g' application.yml"

sh "sed -i 's|C:\\\\\\\\Users\\\\\\\\SSAFY\\\\\\\\Documents\\\\\\\\cherish\_video\\\\\\\\monthly-video\\\\\\\\|/home/ubuntu/ffmpeg/monthly-video/|g' application.yml"

sh "sed -i 's|C:\\\\\\\\Users\\\\\\\\SSAFY\\\\\\\\Documents\\\\\\\\cherish\_video\\\\\\\\clip\\\\\\\\|/home/ubuntu/ffmpeg/clip/|g' application.yml"

sh "cat application.yml"

}

sh './gradlew clean build'

}

dir('frontend/cherish') {

sh "sed -i 's|ws://localhost:8080|wss://i10d103.p.ssafy.io:443|g' .env.local"

sh "sed -i 's|http://localhost:8080|https://i10d103.p.ssafy.io:443|g' .env.local"

sh 'npm install --legacy-peer-deps'

sh 'npm run build'

}

}

stage('Deploy') {

// sh "sed -i 's/host.docker.internal/172.26.8.234/g' docker-compose.yml"

sh 'docker-compose down'

sh 'docker rmi -f cherish\_back\_image:0.0'

sh 'docker rmi -f cherish\_front\_image:0.0'

sh 'docker-compose up -d'

}

// 성공한 경우의 후처리

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.BUILD\_URL}|Details>)",

endpoint: 'https://meeting.ssafy.com/hooks/pqkf1tpowjr3mjwnk7rhm8mc9w',

channel: 'd103-cicd')

} catch (Exception e) {

// 실패한 경우의 후처리

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.BUILD\_URL}|Details>)",

endpoint: 'https://meeting.ssafy.com/hooks/pqkf1tpowjr3mjwnk7rhm8mc9w',

channel: 'd103-cicd')

throw e // 예외를 다시 던져서 빌드 실패 상태를 유지

}

}

1. 프론트엔드 도커 파일

FROM nginx:alpine

COPY dist/ /usr/share/nginx/html

CMD ["nginx", "-g", "daemon off;"]

1. 백엔드 도커 파일

FROM openjdk:17-jdk-oraclelinux7

#FROM openjdk:17

RUN yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm && \

    yum install -y https://download1.rpmfusion.org/free/el/rpmfusion-free-release-7.noarch.rpm && \

    yum install -y ffmpeg

ARG JAR\_FILE=/build/libs/\*.jar

COPY ${JAR\_FILE} app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

1. docker-compose.yml

version: "3"

services:

  cherish-backend:

    container\_name: cherish-backend

    build: ./backend/cherish

    image: cherish\_back\_image:0.0

    environment:

      MYSQL\_HOST: host.docker.internal

    ports:

      - "8080:8080"

    volumes:

      - /home/ubuntu/ffmpeg/monthly-video:/home/ubuntu/ffmpeg/monthly-video

      - /home/ubuntu/ffmpeg/clip:/home/ubuntu/ffmpeg/clip

    networks:

      - backend\_frontend

  cherish-frontend:

    container\_name: cherish-frontend

    build: ./frontend/cherish

    image: cherish\_front\_image:0.0

    ports:

      - "80:80"

      - "443:443"

    volumes:

      - /etc/letsencrypt:/etc/letsencrypt

      - /var/lib/letsencrypt:/var/lib/letsencrypt

      - /home/ubuntu/nginx/sites:/etc/nginx/conf.d

      - /home/ubuntu/nginx/sites/sites-enabled:/etc/nginx/sites-enabled

    networks:

      - backend\_frontend

networks:

  backend\_frontend:

    driver: bridge

1. 순서

젠킨스 파이프 라인을 실행하면 젠킨스가 git clone을 통해 프로젝트를 내려받고 프론트와 백이미지를 빌드합니다. 이후 젠킨스에 넣어놓은 .env.local, application-secretes.yml 같은 파일들을 집어넣고 localhost라고 설정되어 있는 부분을 서버에 맞게 대체시켜 준 뒤 docker를 이용하여 배포합니다.