



포팅 매뉴얼

개발 환경

형상관리

- Gitlab

이슈관리

- Jira

테스트 툴

- Postman

UI/UX

- Fligma

IDE

- IntelliJ 2022.3.1
- Vscode

배포

- AWS EC2
 - Docker : 23.0.1
 - Jenkins : 2.375.2

의사소통

- Maternmost
- Notion

DB

- MySQL : 8.0.27
- Redis : 7.0.9
- S3 : 1.12.387

Server

- WAS
 - Tomcat

백엔드

- Java, : Open JDK-11.0.18
- SpringBoot : 2.7.9
- Spring Gradle 7.6.1
 - Spring Data JPA
 - lombok
 - swagger : 2.2.9
- Jwt : 0.11.5
- querydsl : 1.0.10

프론트엔드

- React-Native : 0.70.5
 - expo : 47
- Javascript

빌드환경 & OS

- Ubuntu : 20.04.5 LTS
- Window Windows 10 10.0 amd64

CI/CD 구축

1 Ubuntu 접속

```
SSAFY@DESKTOP-DOGVPU MINGW64 ~/Desktop/devOps
$ ssh -i J8B203T.pem ubuntu@j8b203.p.ssafy.io
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Mon Mar 13 03:47:45 UTC 2023

System load:  0.0               Processes:    134
Usage of /:   0.7% of 310.15GB  Users logged in: 0
Memory usage: 2%               IPv4 address for eth0: 172.26.1.154
Swap usage:  0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

132 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

3 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-26-1-154:~$
```

내 PC > 바탕 화면 > devOps

devOps 검색

이름	수정한 날짜	유형	크기
J8B203T	2023-03-09 오후 2:59	PEM 파일	2KB
J8B203T.ppk	2023-03-10 오후 3:57	PPK 파일	2KB

1. `.pem` 파일이 있는 곳에서 터미널로 Ubuntu 접속한다.
2. `ssh -i J8B203T.pem ubuntu@j8b203.p.ssafy.io`

2 Docker 설치

1. 우분투 shell에 접속하여 도커를 설치한다.

```
# 최신 버전으로 패키지 업데이트
sudo apt-get update

# 도커 다운을 위해 필요한 패키지 설치
sudo apt-get install apt-transport-https // 패키지 관리자가 https를 통해 데이터 및 패키지에 접근할 수 있도록 해준다.
sudo apt-get install ca-certificates // certificate authority에서 발행되는 디지털 서명, SSL 인증서의 PEM 파일이 포함되어 있어 SSL 기반 앱이 SSL 연결C
sudo apt-get install curl // 특정 웹사이트에서 데이터를 다운로드 받을 때 사용한다.
sudo apt-get install software-properties-common // *PPA를 추가하거나 제거할 때 사용한다.

# curl 명령어로 도커의 공식 GPG 키를 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

```
# 도커의 공식 GPG 키가 추가된 것을 확인
sudo apt-key fingerprint 0EBFCD88

# 도커의 저장소를 추가, 등록
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

# 최신 버전으로 패키지 업데이트
sudo apt-get update

# 도커 설치
sudo apt-get install -y docker-ce

# 도커 실행해보기
sudo usermod -aG docker ubuntu

# 도커 compose 설치
sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# 권한 설정 해주기
sudo chmod +x /usr/local/bin/docker-compose

# 링크 파일 생성
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

2. docker-compose.yml을 이용하여 mysql을 설치한다.

docker 디렉토리 생성 후 현재 위치를 이동한다.

```
mkdir docker
cd docker
vi docker-compose.yml
```

3. docker-compose.yml을 작성한다.

```
version: '3'
services:
  local-db:
    image: library/mysql:8.0.32
    container_name: healing_db
    restart: always
    ports:
      - "3306:3306"
    environment:
      MYSQL_USER: healing
      MYSQL_PASSWORD: 비밀번호
      MYSQL_ROOT_PASSWORD: 비밀번호
      TZ: Asia/Seoul
    volumes:
      - ./db/mysql/data:/var/lib/mysql
      - ./db/mysql/init:/docker-entrypoint-initdb.d
```

4. mysql을 설치한다.

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

Docker Hub

1. Docker 허브에 가입 후 (public)레포지토리를 생성한다.
2. Docker Hub 로그인한다.

```
sudo docker login
```

3. Username과 Password에 Docker Hub에 가입했던 계정과 비밀번호를 입력한다.

```
ubuntu@ip-172-26-1-154:~$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: baekjeongeun
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

4 Backend 배포 준비 (for 수동 배포)

1. Backend 프로젝트 배포를 위해 프로젝트 빌드를 먼저 진행한다.
2. Dockerfile을 작성하고, 이미지를 빌드한다.

```
FROM adoptopenjdk/openjdk11

ARG JAR_FILE=build/libs/*.jar

COPY ${JAR_FILE} app.jar

EXPOSE 8080

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

5 Jenkins 설치 (for 자동 배포)

1. 레포지토리 셋업

```
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

2. 도커의 공식 GPG 키 추가

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

3. 레포지토리 셋업

```
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. 도커 엔진 설치

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

젠킨스 컨테이너 실행

1. 젠킨스 이미지 다운로드

```
$ docker pull jenkins/jenkins:lts
```

2. 젠킨스 컨테이너 띄우기

```
docker run -itd --name jenkins -p 1111:8080 -p 50000:50000 -v /docker/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docke
```

젠킨스 설정

```
$ sudo docker logs jenkins
```

아래 비밀번호 복사 후 지정한 포트의 url로 접속

```
Jenkins initial setup is required. An admin user has been created and a password generated.  
Please use the following password to proceed to installation:
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXX # 비밀번호
```

jenkins 컨테이너 내부에 직접 접속하여, `/var/jenkins_home/secrets/initialAdminPassword` 파일의 내용을 조회하는 방법도 있습니다.

```
$ sudo docker exec -it jenkins /bin/bash  
$ cat /var/jenkins_home/secrets/initialAdminPassword
```

6 Jenkins 설정

```
http://j8b203.p.ssafy.io:1111/
```





🔒 jenkins 로그인 정보

ID : healing


PWD : healing123!

Credentials 설정

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	healing	healing/***** (healing)
		System	(global)	access-token	GitLab API token (healing)

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Plugins 설치

- Dashboard > Plugins > Available plugins

다음 플러그인 검색 후 설치

- GitLab
- Docker
- SSH

Plugins

이름 ↓

사용가능

Generic Webhook Trigger Plugin 1.86.2
Can receive any HTTP request, extract any values from JSON or XML and

시스템 설정

- Dashboard > Jenkins 관리 > System

Jenkins Location

Jenkins URL ?

System Admin e-mail address ?

☒ Environment variables

키-값 목록 ?

이름

값

이름

값

환경 변수 설정

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name
A name for the connection

Gitlab host URL
The complete URL to the Gitlab server (e.g. http://git

Credentials
API Token for accessing Gitlab

Add ▾

고급 ▾

gitlab과의 연동을 위한 설정

Webhook 설정

- Gitlab 레포지토리

Project Hooks (1)

<http://j8b203.p.ssafy.io:1111/project/healing>

Push Events

SSL Verification: enabled

Test ▾

Edit

Delete


Pipeline 생성

- Dashboard > 새로운 Item


Enter an item name

healing diary

Required field


Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.


Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Pipeline General 설정

☒ GitHub project

Project url ?

https://lab.ssafy.com/s08-ai-speech-sub2/S08P22B203.git/

고급 ▾

Edited

GitLab Connection

healing

▾

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j8b203.p.ssafy.io:1111/project/healing> ?

Enabled GitLab triggers

- ☒ Push Events
- ☐ Push Events in case of branch delete
- ☒ Opened Merge Request Events
- ☐ Build only if new commits were pushed to Merge Request ?
- ☒ Accepted Merge Request Events
- ☒ Closed Merge Request Events

Rebuild open Merge Requests

Never

- ☒ Approved Merge Requests (EE-only)
- ☒ Comments

Pipeline script

```
pipeline {
  agent any
  stages {
    stage('Prepare') {
      steps {
        git branch: 'DEV/BE',
            credentialsId: 'healing',
            url: 'https://lab.ssafy.com/s08-ai-speech-sub2/S08P22B203'
      }
    }
    stage('Spring Build') {
      steps {
        dir('BE') {
          sh 'chmod +x gradlew'
          sh './gradlew clean build -x test'
          sh 'ls -al ./build'
        }
      }
    }
    stage('Docker Build') {
      steps {
        sh 'docker build --build-arg JAR_FILE=build/libs/BE-0.0.1-SNAPSHOT.jar -t healing/back ./BE'
      }
    }
    stage('Deploy') {
      steps{
        sh 'docker ps -f name=back -q | xargs --no-run-if-empty docker container stop'
        sh 'docker container ls -a -f name=back -q | xargs -r docker container rm'
        sh 'docker images --no-trunc --all --quiet --filter="dangling=true" | xargs --no-run-if-empty docker rmi'
        sh 'docker run --name back -p 8080:8080 \
          -e "ACTIVE=${ACTIVE}" \
          -e "DB_NAME=${DB_NAME}" \
          -e "DB_PASSWORD=${DB_PASSWORD}" \
          -e "ACCESS_KEY_AWS_S3=${ACCESS_KEY_AWS_S3}" \
          -e "SECRET_KEY_AWS_S3=${SECRET_KEY_AWS_S3}" \
          -e "BUCKET_ADDRESS=${BUCKET_ADDRESS}" \
          -e "SECRET_KEY_CLOVA_SPEECH=${SECRET_KEY_CLOVA_SPEECH}" \
          -e "INVOKE_URL_CLOVA_SPEECH=${INVOKE_URL_CLOVA_SPEECH}" \
          -e "CLIENT_ID_NAVER_AI=${CLIENT_ID_NAVER_AI}" \
          -e "CLIENT_SECRET_NAVER_AI=${CLIENT_SECRET_NAVER_AI}" \
          -e "JWT_SECRET=${JWT_SECRET}" \
          -e "DEFAULT_IMAGE_S3=${DEFAULT_IMAGE_S3}" \
          healing/back'
```



```

    }
    stage('Finish') {
        steps{
            sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
        }
    }
}
}

```

■ Environment Value(환경 변수)

```

ACCESS_KEY_AWS_S3=AKIASBJPUR7YSWZECGTE;

ACTIVE=prod;

BUCKET_ADDRESS=bje-s3-bucket;

CLIENT_ID_NAVER_AI=5ccr1ya64y;

CLIENT_SECRET_NAVER_AI=vQGSfBIG1I6P0ngVhsAv2R3MkyU1r4o05PK3ZAwS;

DB_NAME=healing;

DB_PASSWORD=healing123!;

DEFAULT_IMAGE_S3=https://bje-s3-bucket.s3.ap-northeast-2.amazonaws.com/diary_default_image/;

INVOKE_URL_CLOVA_SPEECH=https://clovaspeech-gw.nccloud.com/external/v1/4849/e904883f02b0baeb3c5a6b90a7ba0b083e0497dbbd588ddd22c06c89dd5

JWT_SECRET=012345678901234567890123456789HealingDiary203;

SECRET_KEY_AWS_S3=AWS_SECRET_KEY;

SECRET_KEY_CLOVA_SPEECH=CLOVA_SECRET_KEY;

```

☀ jenkins 내부에서 docker 실행

```

root@359b82e22c45:~# docker ps
bash: docker: command not found

root@359b82e22c45:~# apt-get remove docker docker-engine docker.io containerd runc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

root@359b82e22c45:~# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS
359b82e22c45   jenkins/jenkins:latest  "/usr/bin/tini -- /u..."  11 hours ago  Up 11 hours  0.0.0.0:50000->50000/tcp, :::50000->50000
c6e02130f936   mysql:8.0.32          "docker-entrypoint.s..."  11 hours ago  Up 11 hours  0.0.0.0:3306->3306/tcp, :::3306->3306/tc

```

7 Mysql 설치

```
docker pull mysql:8.0.32
```

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=비밀번호 -d -p 3306:3306 mysql:8.0.32 --character-set-server=utf8mb4 --collatio
```

```

ubuntu@ip-172-26-12-18:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
3e398889b013   mysql:8.0.32                        "docker-entrypoint.s..." 48 seconds ago Up            0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp mysql-container
dd26f7cb8841   jenkins/jenkins:lts                "/usr/bin/tini -- /u..." 6 minutes ago  Up            50000/tcp, 0.0.0.0:1111->8080/tcp, :::1111->8080/tcp jenkins
ubuntu@ip-172-26-12-18:~$ docker exec -it mysql-container bash
bash-4.4# show databases;
bash: show: command not found
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

```

권한 설정

```

ubuntu@ip-172-26-12-18:~$ docker exec -it mysql-container bash
bash-4.4# mysql -u root -

mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select user, host from user;
+-----+-----+
| user          | host          |
+-----+-----+
| healing       | %             |
| root          | %             |
| mysql.infoschema | localhost    |
| mysql.session | localhost    |
| mysql.sys     | localhost    |
| root          | localhost    |
+-----+-----+
6 rows in set (0.00 sec)

mysql> grant all on *.* to 'healing'@'%' with grant option;
Query OK, 0 rows affected (0.01 sec)

```

사용할 DB 생성하기

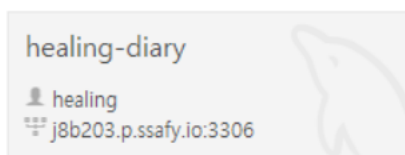
```

mysql> create database healing;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database          |
+-----+
| healing           |
| information_schema |
| mysql             |
| performance_schema |
| sys              |
+-----+
5 rows in set (0.00 sec)

```

MySQL 접속 테스트



🔒 DB 접근

Hostname : j8b203.p.ssafy.io

Username : healing

Password : healing123!

8 Redis 설치

```
docker pull redis
```

- redis 설치 확인

```
ubuntu@ip-172-26-12-18:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
healing/back         latest             c5c65007d7cb       4 hours ago        489MB
jenkins/jenkins      latest             e2999693421d       2 days ago         471MB
mysql                8.0.32            4073e6a6f542       8 days ago         530MB
jenkins/jenkins      lts               d5ed2ceef0ec       8 days ago         471MB
redis                latest            f9c173b0f012       2 weeks ago        117MB
ubuntu@ip-172-26-12-18:~$
```

```
docker run -it --link redis:redis --rm redis redis-cli -h redis -p 6379
```

```
mkdir redis
cd redis
sudo vi redis.conf

...

# 어떤 네트워크 인터페이스로부터 연결할 수 있도록 할 것인지 관리 (여기에서는 Anywhere)
bind 0.0.0.0

# 사용 포트 관리
port 6379

# Master 노드의 기본 사용자(default user)의 비밀번호 설정
requirepass [사용하고자 하는 비밀번호]

# Redis 에서 사용할 수 있는 최대 메모리 용량. 지정하지 않으면 시스템 전체 용량
maxmemory 2gb

# maxmemory 에 설정된 용량을 초과했을 때 삭제할 데이터 선정 방식
# - noeviction : 쓰기 동작에 대해 error 반환 (Default)
# - volatile-lru : expire 가 설정된 key 들중에서 LRU algorithm 에 의해서 선택된 key 제거
# - allkeys-lru : 모든 key 들 중 LRU algorithm에 의해서 선택된 key 제거
# - volatile-random : expire 가 설정된 key 들 중 임의의 key 제거
# - allkeys-random : 모든 key 들 중 임의의 key 제거
# - volatile-ttl : expire time(TTL)이 가장 적게 남은 key 제거 (minor TTL)
maxmemory-policy volatile-ttl

# DB 데이터를 주기적으로 파일로 백업하기 위한 설정입니다.
# Redis 가 재시작되면 이 백업을 통해 DB 를 복구합니다.

save 900 1      # 15분 안에 최소 1개 이상의 key 가 변경 되었을 때
save 300 10     # 5분 안에 최소 10개 이상의 key 가 변경 되었을 때
save 60 10000   # 60초 안에 최소 10000 개 이상의 key 가 변경 되었을 때
...

# 비밀번호 설정
docker exec -it redis bash
# 레디스 접속 후
redis-cli -p 6379

# 1. 비번 확인
127.0.0.1:6379>config get requirepass

# 결과 :
1) "requirepass"
2) ""

# 빈 값으로 보일 경우
# 2. 비번 세팅
127.0.0.1:6379>config set requirepass 비밀번호
OK
127.0.0.1:6379>config get requirepass
# (error) NOAUTH Authentication required. 메시지가 나오면
127.0.0.1:6379>auth 비밀번호
```

OK

```
# 3. 비번 재확인
127.0.0.1:6379>config get requirepass
```

```
# 결과 :
1) "requirepass"
2) "비밀번호"
```

외부 서비스 문서

- CLOVA Sentiment API
- CLOVA Speech API
- Kakao Oauth
- Google Oauth
- AWS S3

DB 덤프 파일

```
-- MySQL dump 10.13 Distrib 8.0.32, for Win64 (x86_64)
--
-- Host: j8b203.p.ssafy.io Database: healing
--
-- Server version 8.0.32

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `club`
--

DROP TABLE IF EXISTS `club`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `club` (
  `club_id` bigint NOT NULL,
  `club_created_date` datetime(6) DEFAULT NULL,
  `club_updated_date` datetime(6) DEFAULT NULL,
  `club_image_url` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `host_member_id` bigint DEFAULT NULL,
  PRIMARY KEY (`club_id`),
  KEY `FKgeopqsd0jp1kwte7t0ehhjj4d` (`host_member_id`),
  CONSTRAINT `FKgeopqsd0jp1kwte7t0ehhjj4d` FOREIGN KEY (`host_member_id`) REFERENCES `member` (`member_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `club`
--

LOCK TABLES `club` WRITE;
/*!40000 ALTER TABLE `club` DISABLE KEYS */;
INSERT INTO `club` VALUES (9,'2023-04-04 12:48:03.278587','2023-04-04 12:48:03.278587','https://bje-s3-bucket.s3.ap-northeast-2.amazonaws.com/club/club_image_url.jpg','club description','club name',1);
/*!40000 ALTER TABLE `club` ENABLE KEYS */;
UNLOCK TABLES;
...
```

