

C104 : 신비한 방탈출 사전 (순혈 3반)

삼성청년SW아카데미 광주캠퍼스 7기

자율프로젝트(22.10.10 ~ 22.11.21)

포팅 매뉴얼

담당 컨설턴트 : 박세영

김윤주(팀장), 권덕민, 김수빈, 유이서, 장한울, 홍찬기

목차

1. 프로젝트 기술 스택	1
가. 이슈관리	
나. 형상관리	
다. 커뮤니케이션	
라. 개발환경	
마. 상세 사용 스택	
바. 배포	
2. 빌드 및 배포	2
가. AWS EC2 DB 세팅	
나. AWS EC2 Docker, Jenkins 세팅	
다. 젠킨스 내부 gitlab 프로젝트로 도커 이미지 빌드	
라. 도커 이미지로 도커 컨테이너 생성	
3. 외부 서비스	12
가. 카카오 소셜 로그인	
4. 프로퍼티 정의	14
가. Nginx 설정	
나. Backend Properties	
다. 외부 서비스키	

포팅메뉴얼

1. 프로젝트 기술 스택

가. 이슈관리

- Jira

나. 형상관리

- Gitlab

다. 커뮤니케이션

- Mattermost
- Notion

라. 개발 환경

- a. OS : Window 10 / Mac Os /
- b. IDE
 - IntelliJ
 - Visual Studio Code
- c. 디자인
 - Figma
 - Unity asset
- d. Database
 - MySQL 8.0.31
- e. Server : AWS EC2
 - Ubuntu 20.4.1 LTS

마. 상세 사용 스택

- a. Frontend
 - 공통
 - react 18.2.0, react-dom 18.2.0
 - recoil 0.7.6, react-query 4.13.4, axios 1.1.3
 - 앱
 - react-native 0.69.6, expo 46.0.16
 - styled-components 5.3.6
 - 웹
 - craco 5.9.0, typescript 4.8.4,
 - react-router-dom 6.4.2,
 - emotion 11.10.5
- b. Backend
 - Java 8(`openjdk:8-jdk-alpine`)
 - SpringBoot 2.7.5 , Gradle
 - Spring Data JPA, Swagger 3.0.0, Lombok
 - Spring Security, JWT 0.11.5

바. 배포

- Jenkins : 2.361.3
- Docker : 20.10.21
- Nginx : 1.18.0(ubuntu)

2. 빌드 및 배포

가. AWS EC2 DB 세팅

a. MySQL 설치

```
sudo apt update # 패키지 인스톨러 업데이트
sudo apt install mysql-server # mysql 설치
mysql --version # mysql 설치 및 버전 확인
```

b. MySQL 실행 및 초기화

```
sudo mysql_secure_installation # 이후 나오는 안내에 따라 패스워드 설정
```

c. root로 MySQL 접속

```
sudo mysql -u root {password} # 설정한 암호로 mysql 접속
```

d. MySQL 외부 접속 설정

```
# mysql.conf.d 디렉토리로 이동
cd /etc/mysql/mysql.conf.d

# mysql.cnf 파일 수정
sudo vi mysqld.cnf
# => bind-address를 127.0.0.1에서 0.0.0.0으로 변경

# mysql에 접속
sudo mysql
```

```
# 외부 접속 계정 생성 & 권한 부여
create user {'계정이름'}@ '%' identified by {'패스워드'};
grant all privileges on *.* to {'계정이름'}@ '%' with grant option;
```

e. Workbench 이용해 접속 확인

나. AWS EC2 Docker, Jenkins 세팅

a. Docker 설치

1) 사전 패키지 설치

```
# 사전 패키지 설치
sudo apt update
sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

2) gpg 키 다운로드

→ 도커 설치를 위해 gpg key 다운 필요. 리눅스 패키지 툴이 프로그램 패키지가 유효한지 확인하기 위해 설치 전 gpg 키를 통해 검증하는 과정을 거치기 때문.

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

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
```

3) Docker 설치

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

b. 도커 컨테이너에 젠킨스 컨테이너 생성, 설치 및 젠킨스 계정 생성(docker-compose 이용)

1) docker-compose.yml 파일 작성

```
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:
      - "9090:8080"
    privileged: true
    user: root
```

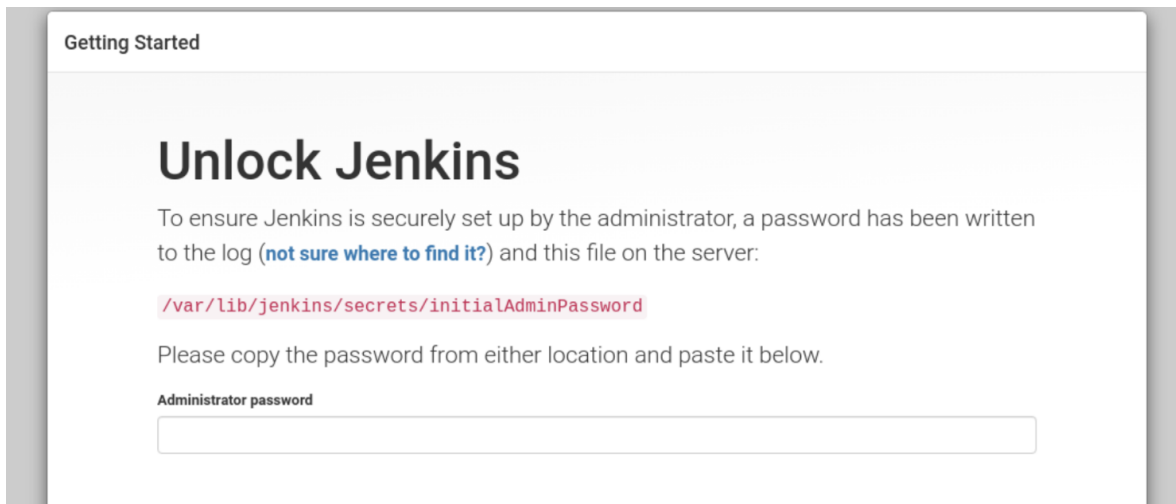
2) 도커 컨테이너 생성

```
sudo docker-compose up -d

sudo docker ps # 컨테이너 확인
```

3) 젠킨스 계정 생성 및 플러그인 설치

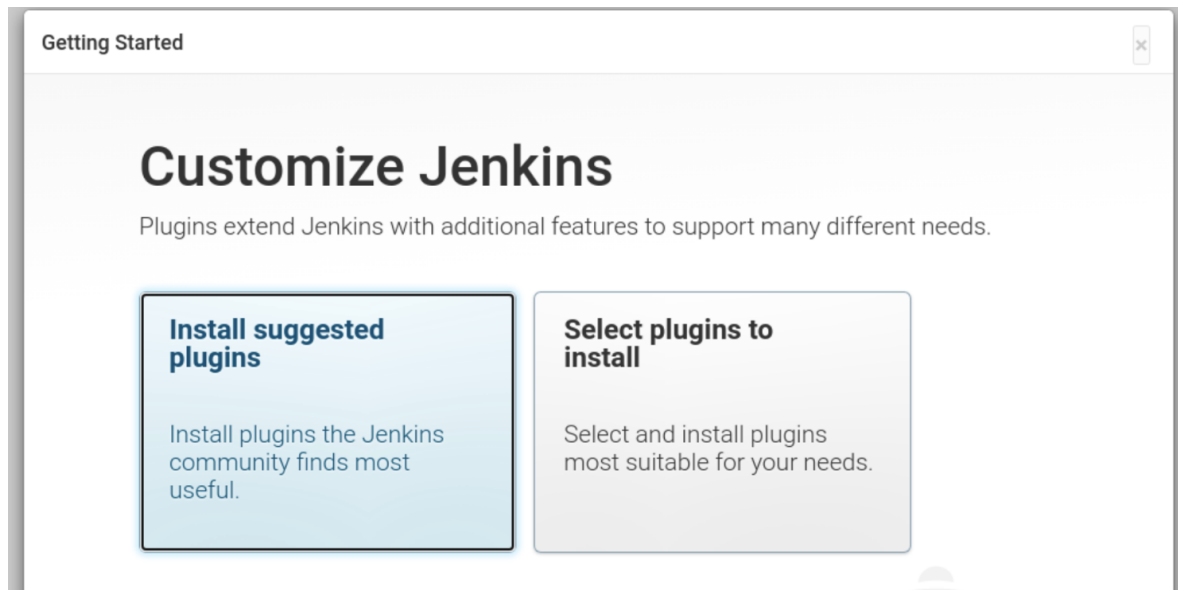
→ k7c104.p.ssafy.io:9090 접속 후



```
sudo docker logs jenkins
```

명령어 실행 결과로 나오는 password 값을 입력해 jenkins 로그인 진행.

→ jenkins 초기 플러그인 설치



- GitLab과 연결하기 위한 Plugin 설치
→ Jenkins 관리 > 플러그인 관리 > 설치 가능에서 GitLab, Generic Webhook Trigger, Gitlab API, GitLab Authentication 체크하고 `install without restart` 버튼 클릭.
- Docker 관련 Plugin 설치
→ Jenkins 관리 > 플러그인 관리 > 설치 가능에서 Docker, Docker Commons, Docker Pipeline, Docker API 체크하고 `install without restart` 버튼 클릭.
- SSH 연결 관련 Plugin 설치
→ Jenkins 관리 > 플러그인 관리 > 설치 가능에서 Publish Over SSH 체크하고 `install without restart` 버튼 클릭.

c. 젠킨스 프로젝트 생성 WebHook 설정

1) 젠킨스 프로젝트 생성 (back, frontweb 각각)

→ Jenkins 메인 > 새로운 Item > item 이름 입력 > Freestyle project 선택

2) 가져올(git pull을 실행할) 레포지토리 및 브랜치 등록

→ 소스코드 관리 탭 > Git 라디오 버튼 클릭 후 Form에 등록

→ Repository URL에 깃 레포지토리 URL 입력

→ Credentials에서 add > jenkins 선택 후 Form에 로그인 정보 등록

- Username : Gitlab username
- Password : Gitlab password
- Id : Credentials 구별할 텍스트

3) Credential 등록 후 오류메시지 사라지면 성공

Configuration

- General
- 소스 코드 관리**
- 빌드 유발
- 빌드 환경
- Build Steps
- 빌드 후 조치

Repositories ?

Repository URL ? ✕

Credentials ?

▼

Branches to build ?

Branch Specifier (blank for 'any') ? ✕

backend

Configuration

- General
- 소스 코드 관리**
- 빌드 유발
- 빌드 환경
- Build Steps
- 빌드 후 조치

Repositories ?

Repository URL ? ✕

Credentials ?

▼

Branches to build ?

Branch Specifier (blank for 'any') ? ✕

frontend web

4) Build 유발 설정

- 레포지토리가 push 됐을 때 빌드 유발을 눌러 어떤 행동을 할지 설정
- Build when a change is pushed to GitLab 선택 > push 이벤트와 merge 이벤트 감지시킴

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: https://k7c104.p.ssafy.io:9090/project/frontend_admin_deploy ?

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

→ 이후 고급 버튼 클릭 > Secret token Generate > Gitlab webhook 연결을 위해 복사해둬.

Allowed branches

☒ Allow all branches to trigger this job ?

☐ Filter branches by name ?

☐ Filter branches by regex ?

☐ Filter merge request by label

Secret token ?

71cf168c5b2280e3d60720bcb050faf6

Generate

Clear

• 연결할 Gitlab Repository Settings > WebHooks

s07-final > S07P31C104 > Webhook Settings

Search page

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

http://k7c104.p.ssafy.io:9090/project/frontend_admin_deploy/

URL must be percent-encoded if it contains one or more special characters.

Secret token

71cf168c5b2280e3d60720bcb050faf6

Used to validate received payloads. Sent with the request in the **X-Gitlab-Token** HTTP header.

Trigger

☒ Push events

Branch name or wildcard pattern to trigger on (leave blank for all)

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

- URL에 http://배포서버공인IP:9090/project/생성한jenkins프로젝트이름/ 입력
- Secret token 에 위에서 발급받은 토큰 값 입력
- Trigger 에 push events, merge request events 설정, 원하는 브랜치 설정
- Add webhook 버튼 눌러서 webhook 생성

다. 젠킨스 내부 gitlab 프로젝트로 도커 이미지 빌드

a. 젠킨스 내부에 도커 설치

1) jenkins bash shell 접근 `sudo docker exec -it jenkins bash`

2) docker 설치

i) 사전 패키지 설치

```
apt update
apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

ii) gpg 키 다운로드

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

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

iii) Docker 설치

```
apt update
apt install docker-ce docker-ce-cli containerd.io docker-compose
```

b. Docker 파일 생성

각 프로젝트에 Dockerfile 이라는 파일명으로 DockerFile 생성

- Spring Dockerfile

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=build/libs/demo-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

- React Dockerfile

```
FROM node:16.15.0 as build
WORKDIR /var/jenkins_home/workspace/frontend_admin_deploy/Frontend/esCAPE_web
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
RUN ls -la
FROM nginx:stable-alpine as production-stage
RUN rm /etc/nginx/conf.d/default.conf
COPY --from=build /var/jenkins_home/workspace/frontend_admin_deploy/Frontend/esCAPE_web/build /usr/share/nginx/html
COPY --from=build /var/jenkins_home/workspace/frontend_admin_deploy/Frontend/esCAPE_web/deploy_conf/nginx.conf /etc/nginx/conf.d/nginx.conf
COPY --from=build /var/jenkins_home/workspace/frontend_admin_deploy/Frontend/esCAPE_web/fullchain.pem /etc/letsencrypt/live/k7c104.p
COPY --from=build /var/jenkins_home/workspace/frontend_admin_deploy/Frontend/esCAPE_web/privkey.pem /etc/letsencrypt/live/k7c104.p

EXPOSE 80
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

c. Docker 이미지 생성

Jenkins > Configuration > Build Steps > Excute shell > command 에 빌드시 필요한 명령어 입력

- backend

```
docker image prune -a --force
mkdir -p /var/jenkins_home/timages
cd /var/jenkins_home/workspace/backend_deploy/BACK/sinbanga/
./gradlew build
docker build -t spring .
docker save spring > /var/jenkins_home/timages/spring.tar
ls /var/jenkins_home/timages
```

- frontend

```
docker image prune -a --force
cd /var/jenkins_home/workspace/frontend_admin_deploy/Front/escape_web/
docker build -t react .
docker save react > /var/jenkins_home/timages/react.tar
ls /var/jenkins_home/timages
```

라. 도커 이미지로 도커 컨테이너 생성

a. 젠킨스 SSH 연결 설정

- Jenkins 관리 > 시스템 설정 > Publish over SSH > SSH Servers 추가

SSH Servers

SSH Server

Name ?

backenddeploy

Hostname ?

172.26.2.42

Username ?

ubuntu

Remote Directory ?

고급...

Test Configuration


→ Name : 이름

- Hostname : EC2 IP
- Username : EC2 접속 계정 이름

고급 탭에서 key에 pem키 입력

☒ Use password authentication, or use a different key ?

Passphrase / Password ?

 Concealed

Change Password

Path to key ?

Key ?

-----BEGIN RSA PRIVATE KEY-----

b. Jenkins 빌드 후 조치에 SSH 명령어 전송(EC2 에 도커 컨테이너 생성)

- Configuration > 빌드 후 조치 > Send build artifacts over SSH

Send build artifacts over SSH ?

SSH Publishers

SSH Server

Name ?

backenddeploy

고급...

Transfers

Transfer Set

Source files ?

/README.md

Remove prefix ?

Remote directory ?

Exec command ?

sudo docker load < /jenkins/timages/spring.tar
if (sudo docker ps | grep "spring"); then sudo docker stop spring; fi
sudo docker run -it -d --rm -p 8080:8080 --name spring spring
echo "Run spring"

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

- backend Exec command

```
sudo docker load < /jenkins/timages/spring.tar
if (sudo docker ps | grep "spring"); then sudo docker stop spring; fi
sudo docker run -it -d --rm -p 8080:8080 --name spring spring
echo "Run spring"
```

- frontend Exec commend

```
sudo docker load < /jenkins/timages/react.tar
if (sudo docker ps | grep "react"); then sudo docker stop react; fi
sudo docker run -it -d -rm -p 80:80 -p 443:443 --name react react
echo "Run react"
```

3. 외부 서비스

가. 카카오 소셜 로그인

a. 애플리케이션 추리

기본 정보

앱 ID	821587
앱 이름	신방사
사업자명	c104

b. 도메인 등록

카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다.

여러개의 URI를 줄바꿈으로 추가해주세요. (최대 10개)

REST API로 개발하는 경우 필수로 설정해야 합니다.

예시: (O) <https://example.com/oauth> (X) <https://www.example.com/oauth>

```
http://localhost:8082
http://localhost:8081
http://k7c104.p.ssafy.io
```

c. 개인정보 및 접근권한 동의 항목 설정

개인정보

항목 이름	ID	상태
닉네임	profile_nickname	● 필수 동의 설정
프로필 사진	profile_image	● 필수 동의 설정
카카오계정(이메일)	account_email	● 선택 동의 설정

d. 카카오 로그인 활성화 설정

활성화 설정

상태

ON

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다.

상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.

상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

OpenID Connect 활성화 설정

상태

OFF

카카오 로그인의 확장 기능인 OpenID Connect를 활성화합니다.

이 설정을 활성화하면 카카오 로그인 시 사용자 인증 정보가 담긴 ID 토큰을 액세스 토큰과 함께 발급받을 수 있습니다.

4. 프로퍼티 정의

가. Nginx 설정

```
server {
    listen 80;
    server_name k7c104.p.ssafy.io;
    return 301 https://k7c104.p.ssafy.io$request_uri;
}
server {
    listen 443 ssl http2;
    server_name k7c104.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k7c104.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k7c104.p.ssafy.io/privkey.pem;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;

        proxy_set_header Host $http_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 /api {
        proxy_pass http://k7c104.p.ssafy.io:8080;
        proxy_set_header Host $http_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;
    }
}

server {
    if ($host = k7c104.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }
    listen 80;
    return 404;
}
```

나. Backend Properties


```

#my sql
server.port = 8080
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://k7c104.p.ssafy.io:3306/sinbangsa
spring.datasource.username=pureblood3
spring.datasource.password=sinbangsa

spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER

#jpa
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true

#jwt secret key
springboot.jwt.secret = "SinBangSaSecretKeyJWTTokenCreateKey"

```

다. 외부 서비스키

```

#kakao login

kakaoRestApiKey=6cb2dd1e35672b64fb0dac71ee59315f
kakaoRedirectUri=http://localhost:8082
kakaoClientSecret=nPbxKBveS9LdVJv8rHjV7dXXztkbK1tj

```