



포팅 매뉴얼

≡ 태그

산출물

🍺 주간일기 🍺 포팅 매뉴얼

🍺 주간일기 🍺 포팅 매뉴얼

📄 ERD

📁 시스템 아키텍처

🔧 개발환경

1. 서버 인스턴스 사양

2. 개발환경

형상 관리

UI / UX

OS

이슈 관리

Communication

Infra

Front-end

Back-end

DB

IDE

기타 편의 툴

🔧 백엔드 배포 과정

1. EC2 접속

1.1 WSL 이용하여 Linux 운영체제 구동

1.2. EC2 초기 설정

2. EC2 환경 설정

2.1. Docker 설치

2.2. Docker Compose 설치

3. Docker Compose 설정

3.1. Jenkins Docker Compose

3.2. Spring Docker Compose

3.3. MySQL Docker Compose

3.4. Redis Docker Compose

4. Jenkins 관련 추가 설정

4.1 Jenkins 설정

4.2 Gilab Webhook 설정

5. 배포 환경 구성

5.1. 도메인 구매

5.2. SSL 적용

5.3. Nginx

🔧 expo 배포 과정

0. 초기 설정

1. 개발용 배포

1.1. **.apk** 파일 배포(미리보기용)

1.2. **development build**(개발 확인용)

2. Store 출시용 배포

2.1. 환경 설정

2.2. 공통 사전 준비 사항

2.3. Play Store 출시 사전 준비

2.5. ONE Store 출시 사전 준비

2.6. App Store 출시 사전 준비

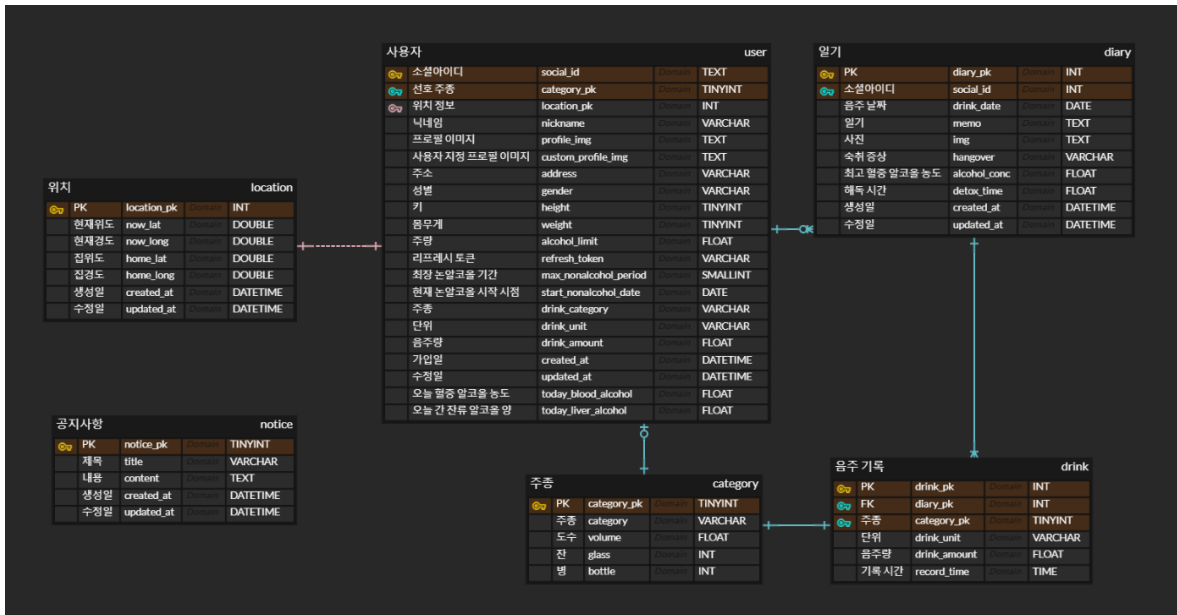
3. Store 출시 후 업데이트

3.1. 업데이트용 파일 생성

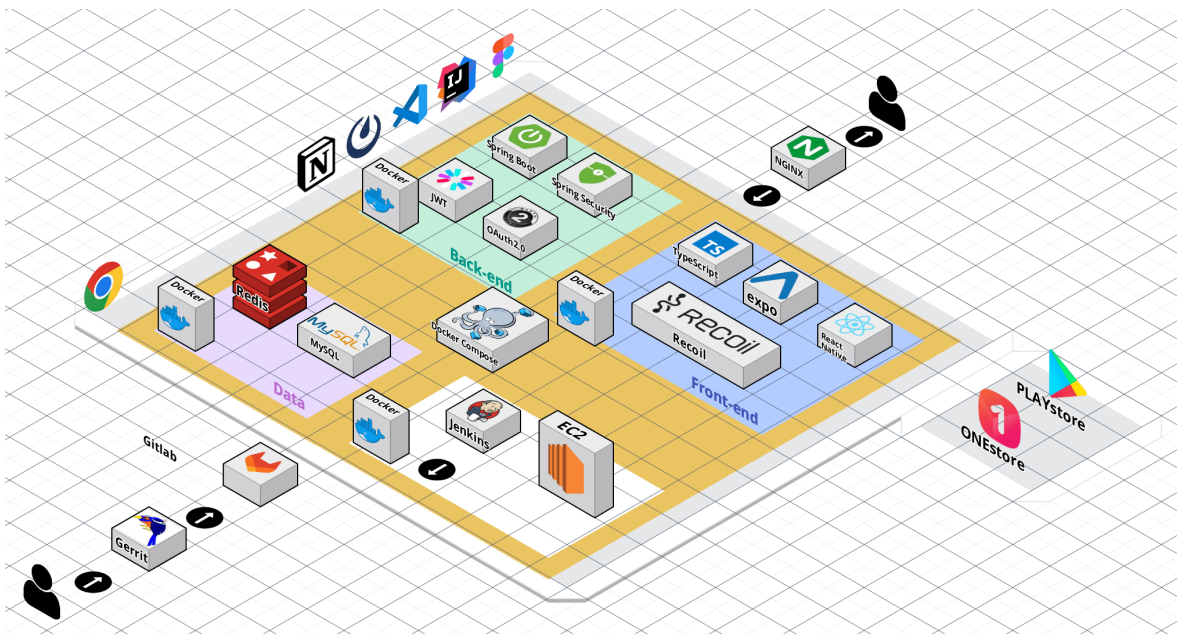
📍 외부서비스

1. Kakao 로그인

▼ ERD



▼ 시스템 아키텍처



🔧 개발환경

1. 서버 인스턴스 사양

- (개발) CPU 정보: Intel Xeon(R) Core 4개

- RAM : 16GB

- Disk : 300GB

```
ubuntu@ip-172-26-14-185:~$ cat /proc/meminfo
MemTotal: 16396056 kB
```

```
ubuntu@ip-172-26-14-185:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        311G  13G  298G   5% /
```

```

ubuntu@ip-172-28-14-185:~$ cat /proc/cpuinfo | more
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 89
model name     : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
stepping       : 1
microcode      : 0xb0000040
cpu MHz        : 2300.000
cache size     : 46080 KB
physical id    : 0
siblings       : 4
core id        : 0
cpu cores      : 4

```

- (배포) CPU 정보 : Intel Xeon(R)
Core 4개

2. 개발환경

형상 관리

- GitLab
- Gerrit
- Jira

OS

- Ubuntu 22.04.2 LTS
- window 10

Communication

- Mattermost
- Notion

UI / UX

- Figma

이슈 관리

- Jira

Infra

- Web Server : Nginx
- Jenkins : 2.414.3
- Docker : 24.0.6
- Docker-compose : 2.23.0

Front-end

- React Native : 0.72.6
- TypeScript : 5.1.3
- expo : 49.0.15
- Node.js : 18.16.1
- yarn : 1.22.19

Back-end

- Spring boot : 3.1.5
- JDK : 17
- swagger : 3
- gradle : 8.3

DB

- MySQL : 8.0.34
- Redis : 7.2.3

IDE

- IntelliJ : 2023.2.5
- Visual Studio Code : 1.84.2

기타 편의 툴

- WSL2
- Postman

백엔드 배포 과정

1. EC2 접속

1.1 WSL 이용하여 Linux 운영체제 구동

(Windows 기준) wsl 실행

- 실행 실패할 경우 : 아래 링크 참조하여 WSL 수동 설치

이전 버전 WSL의 수동 설치 단계

wsl install 명령을 사용하지 않고 이전 버전의 Windows에 WSL을 수동으로 설치하는 방법에 대한 단계별 지침입니다.

<https://learn.microsoft.com/ko-kr/windows/wsl/install-manual#step-4---download-the-linux-kernel-upd>

Microsoft Learn

WSL에 SSH 키(pem 키) 복사

```
cp /mnt/c/Users/SSAFY/Desktop/K9E103T.pem ~/
```

SSH를 이용하여 원격 서버 접속

- 현재 디렉토리를 홈 디렉토리로 변경

```
cd ~
```

- pem 키를 이용한 서버 로그인

```
# sudo ssh -i [pem키 위치] [접속 계정]@[접속할 도메인]
ssh -i K9E103T.pem ubuntu@k9e103.p.ssafy.io
```

- 아래와 같은 경로 만날 경우 권한 축소

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'k9e103.p.ssafy.io' (ssh-rsa) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!                @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0755 for 'J9E103T.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "J9E103T.pem": bad permissions
```

```
chmod 700 K9E103T.pem
```

▼ EC2 편리하게 접속하기

기존 방법

```
ssh -i K9E103T.pem ubuntu@k9e103.p.ssafy.io
```

EC2 정보가 담긴 config파일 활용하기

- ssh 전용 폴더 생성

```
mkdir ~/.ssh          # 이미 존재한다는 메시지가 뜨면 무시하기
cd ~/.ssh             # ssh 폴더 생성 및 이동
cp /home/ohyes/K9E103T.pem ~/.ssh # pem 키 복사, find ~ -name K9E103T.pem 명령어 활용
vi config             # config 파일 생성
```

- Config 내용 설정

```
# 개발 서버 설정
Host ohyes-dev
    HostName k9e103.p.ssafy.io
    User ubuntu
    IdentityFile ~/.ssh/K9E103T.pem

# 운영 서버 설정
Host ohyes-prod
    HostName k9e103a.p.ssafy.io
    User ubuntu
    IdentityFile ~/.ssh/K9E103T.pem
```

- Host 이름을 이용하여 접속하기

ssh ohyes-dev ssh ohyes-prod

▼ 로그 확인하기

```
# 도커 컨테이너 확인
$ docker ps -a

# 지금까지 찍힌 로그 확인
# Tab키 역행
$ docker logs [컨테이너이름]

# 실시간 로그 확인
$ docker logs -f [컨테이너이름]
```

1.2. EC2 초기 설정

```
# ubuntu 패키지 리스트 업데이트
$ sudo apt update

# 설치된 모든 패키지를 최신 버전으로 업그레이드
$ sudo apt upgrade

# Do you want to continue?[Y/n] y

$ sudo apt install build-essential

# Do you want to continue?[Y/n] y

# 서버 시간 한국으로 설정
$ sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime

# 시간 확인
$ date
```

2. EC2 환경 설정

2.1. Docker 설치

Docker 설치에 필요한 패키지 설치

```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

▼ 추가 설명

- `apt-transport-https` : https를 통해 패키지를 다운로드 받을 수 있게 해주는 패키지
- `ca-certificates` : ca-certificate는 certificate authority에서 발행되는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL 기반 앱이 SSL 연결이 되어있는지 확인 가능
- `curl` : 특정 웹사이트에서 데이터를 전송받는 데 사용되는 도구
- `software-properties-common` : 소프트웨어 리파지토리를 추가하거나 관리하는 데 필요한 스크립트와 애플리케이션들을 포함하고 있는 패키지

Docker의 공식 설치 스크립트를 이용하여 Docker 설치

```
$ sudo wget -qO- https://get.docker.com/ | sh
```

▼ 추가 설명

- `wget` : 인터넷에서 파일을 받을 때 사용하는 리눅스 명령어
- `o` : wget은 다운로드 경로의 마지막 슬래시 다음에 오는 단어를 파일 이름으로 한다. 여기서는 빈칸이 되니 다른 이름으로 저장하는 옵션 -O를 사용

- `q` : 출력 없이 종료
- `| sh` : `|` 는 파이프라인, 즉 wget으로 파일을 다운받은 후 shell을 실행한다는 의미

Docker 서비스의 상태 확인, 실행 및 부팅 시 자동 실행 설정

```
$ sudo systemctl status docker # Docker 시스템 확인
$ sudo systemctl start docker # Docker 시작
$ sudo systemctl enable docker # Docker 부팅시 자동 시작
```

▼ 추가 설명

- `systemctl` : 리눅스에서 서비스를 등록, 삭제(mask, unmask) / 활성화, 비활성화(enable, disable) / 시작, 중지, 재시작(start, stop, restart) / 상태 확인(status) / 서비스 확인(list-units, list-unit-files)을 할 수 있는 명령어

Docker 권한 설정

```
$ sudo usermod -aG docker $USER # Docker에 사용자 추가
$ sudo systemctl restart docker # 혹은 exit로 ubuntu 종료 후 재접속
$ sudo chown root:docker /var/run/docker.sock
```

▼ 추가 설명

- 사용자 추가로 sudo를 사용하지 않고 docker를 사용할 수 있다.(재시작 필요)

Docker 설치 확인

```
$ docker -v
```

2.2. Docker Compose 설치

```
# Docker Compose 바이너리 파일 다운로드 후 설정한 경로에 저장
$ sudo curl -L https://github.com/docker/compose/releases/download/v2.23.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

# 실행 권한 추가
$ sudo chmod +x /usr/local/bin/docker-compose

# Docker 설치 확인
$ docker-compose -v
```

3. Docker Compose 설정

3.1. Jenkins Docker Compose

Jenkins Docker Compose 설정 및 설치

- `docker-compose.yml` 생성

```
$ mkdir compose && cd compose
$ mkdir jenkins
$ mkdir .ssh
$ vim docker-compose.yml
```

- `docker-compose.yml` Jenkins 설정 추가

```
version: "3"
services:
```

```
jenkins:
  container_name: jenkins
  build:
    context: jenkins-dockerfile
    dockerfile: Dockerfile
  user: root
  ports:
    - 9090:8080
  volumes:
    - /home/ubuntu/compose:/root_home
    - /home/ubuntu/compose/jenkins:/var/jenkins_home
    - /home/ubuntu/compose/.ssh:/root/.ssh
    - /var/run/docker.sock:/var/run/docker.sock
    - /etc/nginx/conf.d:/etc/nginx/conf.d
  environment:
    - TZ=Asia/Seoul
```

- **Dockerfile** 생성

```
$ mkdir jenkins-dockerfile && cd jenkins-dockerfile
$ vim Dockerfile
$ cd ..
```

- **Dockerfile** 설정

```
FROM jenkins/jenkins:lts

USER root
RUN apt-get update && \
  apt-get install -y openssh-client && \
  curl -L https://github.com/docker/compose/releases/download/v2.23.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose && \
  chmod +x /usr/local/bin/docker-compose && \
  rm -rf /var/lib/apt/lists/*
```

Jenkins Docker 이미지 빌드

```
$ docker-compose up --build -d

# 컨테이너 정보 확인
$ docker ps -a

# 비밀번호 확인
$ docker exec -it jenkins /bin/bash
cat /var/jenkins_home/secrets/initialAdminPassword
```

```
~/compose$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
647023026a8a   compose-jenkins  "/usr/bin/tini -- /u..."  6 minutes ago  Up 6 minutes  50000/tcp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp  jenkins
```

3.2. Spring Docker Compose

Compose 설정

- 현재 위치 `~/compose/`

```
$ cd /home/ubuntu/compose
$ sudo vim docker-compose.springa.yml
$ sudo vim docker-compose.springb.yml
```

무중단 배포를 위해 컨테이너 2개 설정

- `docker-compose.springa.yml` **Spring 설정 추가**

```

version: "3"
services:
  springmeet:
    container_name: springmeet-springa
    build:
      context: spring-dockerfile
      dockerfile: Dockerfile
    restart: on-failure
    ports:
      - 8080:8080
    volumes:
      - /home/ubuntu/compose/jenkins/workspace/Back-dev/backend/SoolSool/build/libs:/deploy
    environment: # 기타 환경설정 추가
      - TZ=Asia/Seoul
      - SPRING_DATASOURCE_DRIVER_CLASS_NAME=com.mysql.cj.jdbc.Driver
      - SPRING_DATASOURCE_URL=jdbc:mysql://52.78.128.242:3306/soolsool?createDatabaseIfNotExist=true&useUnicode=true&characterEncoding=utf8
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=ohyes
      - JWT_SECRET_KEY=234secretYoonBHaYoonIHye235onSeYoun34gHyeMi234nSoJungF34ighting32425sdfasdf
      - CLIENT_ID=b6ffe641bddfb241b3677f88c7462119
      - CLIENT_SECRET=5CzHA7PJLyPAv1v2h5fgJMDq1fKxwD1A
      - ACCESS_KEY=AKIAY2WNQU4NPTTMP5H4
      - SECRET_ACCESS_KEY=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
      - ADMIN_KEY=5e40b324737c2e5a6dd6ce1247c0071f
      - ODSAY_API_KEY=i9DbX0l8wf9xs2LklJQ+nEOLklCfV1a5U6agPv/p6S8
      - REDIS_HOST=52.78.128.242
      - REDIS_PASSWORD=ohyes

networks:
  default:
    external:
      name: springnetwork

```

- `docker-compose.springb.yml` **Spring 설정 추가**

```

version: "3"
services:
  springmeet:
    container_name: springmeet-springb
    build:
      context: spring-dockerfile
      dockerfile: Dockerfile
    restart: on-failure
    ports:
      - 8081:8080
    volumes:
      - /home/ubuntu/compose/jenkins/workspace/Back-dev/backend/SoolSool/build/libs:/deploy
    environment: # 기타 환경설정 추가
      - TZ=Asia/Seoul
      - SPRING_DATASOURCE_DRIVER_CLASS_NAME=com.mysql.cj.jdbc.Driver
      - SPRING_DATASOURCE_URL=jdbc:mysql://52.78.128.242:3306/soolsool?createDatabaseIfNotExist=true&useUnicode=true&characterEncoding=utf8
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=ohyes
      - JWT_SECRET_KEY=234secretYoonBHaYoonIHye235onSeYoun34gHyeMi234nSoJungF34ighting32425sdfasdf
      - CLIENT_ID=b6ffe641bddfb241b3677f88c7462119
      - CLIENT_SECRET=5CzHA7PJLyPAv1v2h5fgJMDq1fKxwD1A
      - ACCESS_KEY=AKIAY2WNQU4NPTTMP5H4
      - SECRET_ACCESS_KEY=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
      - ADMIN_KEY=5e40b324737c2e5a6dd6ce1247c0071f
      - ODSAY_API_KEY=i9DbX0l8wf9xs2LklJQ+nEOLklCfV1a5U6agPv/p6S8
      - REDIS_HOST=52.78.128.242
      - REDIS_PASSWORD=ohyes

networks:
  default:
    external:
      name: springnetwork

```

- `Dockerfile` **생성**

```

$ mkdir spring-dockerfile && cd spring-dockerfile
$ sudo vim Dockerfile
$ cd ..

```

- `Dockerfile` **설정**


```
FROM openjdk:17-jdk

ENTRYPOINT java -jar /deploy/SoolSool-0.0.1-SNAPSHOT.jar
ENV SPRING_PROFILES_ACTIVE=dev # 환경변수 등록(application-dev.yml 파일 참고)
ENV TZ=Asia/Seoul
EXPOSE 8080
```

- **springnetwork** 생성

```
$ docker network create springnetwork
$ docker network ls
```

3.3. MySQL Docker Compose

Compose 설정

- 현재 위치 ~/compose/

```
$ vim docker-compose.yml
```

- **docker-compose.yml** **MySQL 설정** 추가 - 기존의 **jenkins** 라인에 맞춰서 추가

```
.....
mysql:
  image: mysql:8.0
  restart: always
  container_name: mysql
  ports:
    - 3306:3306
  environment:
    - MYSQL_DATABASE=soolsool
    - MYSQL_ROOT_PASSWORD=ohyes
    - TZ=Asia/Seoul
  command:
    - --character-set-server=utf8mb4
    - --collation-server=utf8mb4_unicode_ci
    - --lower_case_table_names=1
  volumes:
    - ./data:/var/lib/mysql
  networks:
    - springnetwork

networks:
  springnetwork:
    external: true
```

- Docker Compose 실행

```
$ docker-compose up --build -d
```

MySQL User 권한 설정

- Docker - mysql 컨테이너 접속

```
$ docker exec -it mysql /bin/bash
```

- MySQL - 루트 계정으로 데이터베이스 접속 및 사용자 추가

```
mysql -u root -p

# 비밀번호 : ohyes
```

```
# 예시 - create user 'user_name'@'XXX.XXX.XXX.XXX' identified by 'user_password';
create user 'ohyes'@'%' identified by 'ohyes';
```

- 사용자 권한 부여

```
# 예시 - grant all privileges on db_name.* to 'user_name'@'XXX.XXX.XXX.XXX';

grant all privileges on soolsool.* to 'ohyes'@'%';
flush privileges;
```

3.4. Redis Docker Compose

Redis Desktop 설치

<https://www.docker.com/get-started/>

```
version: "3"
services:
  jenkins:
    container_name: jenkins
    build:
      context: jenkins-dockerfile
      dockerfile: Dockerfile
    user: root
    ports:
      - 9090:8080
    volumes:
      - /home/ubuntu/compose:/root_home
      - /home/ubuntu/compose/jenkins:/var/jenkins_home
      - /home/ubuntu/compose/.ssh:/root/.ssh
      - /var/run/docker.sock:/var/run/docker.sock
      - /etc/nginx/conf.d:/etc/nginx/conf.d
    environment:
      - TZ=Asia/Seoul

  mysql:
    image: mysql:8.0
    restart: always
    container_name: mysql
    ports:
      - 3306:3306
    environment:
      - MYSQL_DATABASE=soolsool
      - MYSQL_ROOT_PASSWORD=ohyes
      - TZ=Asia/Seoul
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
      - --lower_case_table_names=1
    volumes:
      - ./data:/var/lib/mysql
    networks:
      - springnetwork

  redis:
    container_name: redis
    image: redis:latest
    command: redis-server --requirepass ohyes
    ports:
      - 6379:6379
    volumes:
      - ./redisdata:/data
    networks:
      - springnetwork

networks:
  springnetwork:
    external: true
```

```
version: '3'

services:
  redis:
    container_name: redis
    image: redis:latest
    command: redis-server --requirepass ohyes
    ports:
      - 6379:6379
```

```
$ sudo ufw allow 6379
```

4. Jenkins 관련 추가 설정

4.1 Jenkins 설정

Jenkins 사이트에서 플러그인 설치

※ 퍼블릭 [ip] 주소:9000으로 접속

개발 서버

- <http://52.78.128.242:9090/>
- <http://k9e103.p.ssafy.io:9090/>
- id : ohyes
(admin)
- password : ohyes
(395a82ab417849e7b7c92396d7e69d8d)

운영 서버

- <http://13.125.128.10:9090/>
- <http://k9e103a.p.ssafy.io:9090/>
- id : ohyes-prod
(admin)
- password : ohyes
(4ef4a1dc79674ec5830ee8b1db98f28b)

- 정상적으로 로그인 됐다면 아래의 화면에서 **Install suggested plugins** 선택

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

✓ Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	** SSH server Folders ** Trilead API
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		

- 설치가 완료되면, Adnub 계정 생성창이 나오고, 본인이 사용할 정보들을 입력 (skip)
- 앞으로 이 url로 Jenkins에 접속 가능

Jenkins 플러그인 추가 설정

- Gitlab, Docker 플러그인 받기

Dashboard > Jenkins 관리 > Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Plugins

Search available plugins

Install

Install	Name ↓	Released
<input checked="" type="checkbox"/>	GitLab 1.7.16 Build Triggers This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.	1 mo 17 days ago
<input checked="" type="checkbox"/>	Docker 1.5 Cloud Providers Cluster Management docker This plugin integrates Jenkins with Docker	1 mo 19 days ago

Jenkins Credential 등록

- Gitlab Resository에서 **Settings** → **Access Token**에 들어가 토큰 발급
- **Jenkins관리** → **Manage Credentials**에 들어간다

Security

Security
Secure Jenkins; define who is allowed to access/use the system.

Credential Providers
Configure the credential providers and types

Credentials
Configure credentials

Users
Create/delete/modify users that can log in to this Jenkins.

- **System** 선택 후 **Global credentials** 선택 후 우측 상단 **Add Credentials** 클릭

Stores scoped to Jenkins

P Store ↓

System

System

Domain ↓

Global credentials (unrestricted)

- **Credentials** 작성

New credentials

Kind

GitLab API token

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

API token

Gitlab에서 발급 받은 Access Token 붙여넣기

ID ?

식별할 수 있는 ID 값 입력

Description ?

Create

System 설정

- Dashboard > Jenkins 관리 > System > GitLab

Dashboard > Jenkins 관리 > System >

GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?

A name for the connection

Repository 이름 (ex- S09P31E103)

GitLab host URL ?

The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com

Credentials ?

API Token for accessing GitLab

GitLab API token (위에서 생성한 Credentials)

Add

고급

Test Connection


새로운 Item 추가

- Dashboard > + 새로운 Item


Enter an item name

Item 이름


» 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.


Multi-configuration project

다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.


Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates separate namespaces, so you can have multiple things of the same name as long as they are in different folders.

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssafy.com/s09-final/S09P31E103.git (Gitlab 주소 입력)

❗ Failed to connect to repository : Command "git ls-remote -h -- https://lab.ssafy.com/s09-final/S09P31E103.git HEAD" returned status code 128:
 stdout:
 stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password. See https://lab.ssafy.com/help/topics/git/troubleshooting_git#error-on-git-fetch-http-basic-access-denied

Credentials ?

- none -

Add

고급

- Credentials - Add

Configure

Repository URL ?

Domain

Global credentials (unrestricted)

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

Gitlab 계정

☐ Treat username as secret ?

Password ?

Gitlab 비밀번호

ID ?

Description ?

Add

Cancel

◦ 브랜치 설정

Git ?

Repositories ?

Repository URL ?

https://lab.ssafy.com/s09-final/S09P31E103.git

Credentials ?

hyemco@gmail.com/*****

Add

고급

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

연결할 Branch

Add Branch

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: <http://k9e103a.p.ssafy.io:9090/project/Back-prod> ?

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 ?

Comment (regex) for triggering a build ?

- 빌드 유발 - 고급 - Secret token 생성

Secret token ?

생성된 토큰 복사

Generate

- Build Steps - Execute shell

Build Steps

Add build step +

Filter

- Add a new template to all docker clouds
- Build / Publish Docker Image
- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with Timeout
- Set build status to "pending" on GitHub commit
- Start/Stop Docker Containers

```
export DOCKER_HOST=unix:///var/run/docker.sock
cd /var/jenkins_home/workspace/Back-dev/backend/SoolSool
java -version
chmod +x ./gradlew
./gradlew clean bootjar -Pspring.profiles.active=dev

cd /root_home

# 실행 중인 도커 컴포즈 확인
EXIST_A=$(docker-compose -p springmeet-springa -f docker-compose.springa.yml ps | grep Up) || true

if [ -z "${EXIST_A}" ] # -z는 문자열 길이가 0이면 true. A가 실행 중이지 않다는 의미.
then
```



```

        # B가 실행 중인 경우
        START_CONTAINER=springa
        TERMINATE_CONTAINER=springb
        START_PORT=8080
        TERMINATE_PORT=8081
    else
        # A가 실행 중인 경우
        START_CONTAINER=springb
        TERMINATE_CONTAINER=springa
        START_PORT=8081
        TERMINATE_PORT=8080
    fi

    echo "springmeet-${START_CONTAINER} up"

    # 실행해야하는 컨테이너 docker-compose로 실행
    docker-compose -p springmeet-${START_CONTAINER} -f docker-compose.${START_CONTAINER}.yaml up -d --build

    for cnt in {1..10} # 10번 실행
    do
        echo "check server start.."

        # 스프링부트에 등록했던 actuator로 실행되었는지 확인
        UP=$(curl -s https://soolsool.site/api/soolsool/health | grep 'UP') || true
        if [ -z "${UP}" ] # 실행되었다면 break
        then
            echo "server not start.."
        else
            break
        fi

        echo "wait 10 seconds" # 10 초간 대기
        sleep 10
    done

    if [ $cnt -eq 10 ] # 10번동안 실행이 안되었으면 배포 실패, 강제 종료
    then
        echo "deployment failed."
        exit 1
    fi

    echo "server start!"
    echo "change nginx server port"

    # sed 명령어를 이용해 service-url.inc의 url값 변경
    sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/service-url.inc

    # 기존에 실행 중이던 spring docker-compose는 종료
    echo "springmeet-${TERMINATE_CONTAINER} down"
    docker-compose -p springmeet-${TERMINATE_CONTAINER} -f docker-compose.${TERMINATE_CONTAINER}.yaml down
    echo "success springmeet-${START_CONTAINER} up"

    # 기존에 실행 중이던 mysql 컨테이너 내리기
    if [ "$(docker ps -q -f name=mysql)" ]; then
        docker-compose stop mysql && docker-compose rm -f mysql;
    fi

    if [ "$(docker images -q mysql:8.0)" ]; then
        docker-compose stop mysql && docker-compose rm -f mysql;
        docker rmi -f mysql:8.0;
    fi

    docker-compose up --build -d mysql
    echo "success mysql up"
    echo "success deployment"

```

※ 아래 Nginx 설정까지 해야 제대로 실행 가능

4.2 Gilab Webhook 설정

Settings - Webhooks - 오른쪽 상단

Webhook

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

Jenkins 빌드 유발 url 복사

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

☒ Show full URL
☐ Mask portions of URL
 Do not show sensitive data such as tokens in the UI.

Secret token

Jenkins 빌드 유발 - 고급 - Secret token 값

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

Trigger

☒ Push events

- ☒ All branches
- ☐ Wildcard pattern
- ☐ Regular expression

☐ Tag push events
A new tag is pushed to the repository.

☐ Comments
A comment is added to an issue or merge request.

☐ Confidential comments
A comment is added to a confidential issue.

☐ Issues events
An issue is created, updated, closed, or reopened.

☒ Confidential issues events
A confidential issue is created, updated, closed, or reopened.

☒ Merge request events
A merge request is created, updated, or merged.

☒ Job events
A job's status changes.

☒ Pipeline events
A pipeline's status changes.

☐ Wiki page events
A wiki page is created or updated.

☐ Deployment events
A deployment starts, finishes, fails, or is canceled.

☐ Feature flag events
A feature flag is turned on or off.

☐ Releases events
A release is created or updated.

SSL verification

☒ Enable SSL verification

Save changes Test

5. 배포 환경 구성

5.1. 도메인 구매

5.2. SSL 적용

<https://www.sslforfree.com/> 접속 후 Domain 입력 및 회원가입(혹은 로그인)

- Domain : `soolsoul.life`



SSL For Free

Free SSL Certificates & Wildcard SSL Certificates in Minutes

Secure | https:// **domain 입력** cure (Example: domain.com)

Create Free SSL Certificate



100% Free Forever

Never pay for SSL again. Powered by ZeroSSL with free 90-day certificates.



Widely Trusted

Our free SSL certificates are trusted in 99.9% of all major browsers worldwide.



Enjoy SSL Benefits

Protect user information, generate trust and improve Search Engine Ranking.

SSL Certificate 세팅

- Domain 입력

SSL Certificate Setup

You're on your way to issuing a brand-new SSL certificate for one or multiple domains. Before you can install your new certificate, please complete the steps below.

Domains

☐ I need a wildcard certificate PRO

Please enter at least one domain to secure. For single-domain certificates the WWW-version of your domain will always be included at no extra charge.

Enter Domains

soolsool.life

soolsool.life

www.soolsool.life

+ Add Domain PRO

Next Step →

> Validity

> CSR & Contact

> Finalize Your Order

- 유효 기간 선택 - 90-Day Certificate

포팅 매뉴얼

19

SSL Certificate Setup

You're on your way to issuing a brand-new SSL certificate for one or multiple domains. Before you can install your new certificate, please complete the steps below.

✓ Domains

✓ Validity

You can now choose between generating 90-day or one-year certificate validity. To keep manual work at a minimum, we recommend 1-year certificates.

☒ 90-Day Certificate

☐ 1-Year Certificate PRO

Next Step →

✓ CSR & Contact

> Finalize Your Order

- 도메인 인증 - DNS (CNAME)

✓ Verification Method for soolsool.site

We need you to verify ownership of each domain in your certificate. Please select your preferred verification method and click "Next Step".

☐ Email Verification

☒ DNS (CNAME)

✓ Follow the steps below

To verify your domain using a **CNAME record**, please follow the steps below:

- 1 Sign in to your DNS provider, typically the registrar of your domain.
- 2 Navigate to the section where DNS records are managed.
- 3 Add the following CNAME record:

Name

_C867DAAA6A51B9062CAE02A2120D39DB.soolsool.site

Point To

5149C72A3FFBDD6B743915D8B06A1C89.F09F57669A55B435C2CEBDCFD358C73F.6b0e91725c7d027.comodoca.com

TTL

3600 (or lower)

- 4 Save your CNAME record and click "Next Step" to continue.

- ## DNS 관리

- 인증서 발급

- 인증 성공 후 압축 파일 Download

✓ Your certificate has been issued and is ready for installation. To continue, please follow the steps below.

soolsol.site

We've prepared installation instructions for all major server types.
To download and install your certificate, please follow the steps below:

▼ Download Certificate

Your certificate is compatible with any type of web server. Download your certificate right away or make a selection below to get instructions and tutorials specific to your web server.

Server Type

Default Format
Download Certificate (.zip)
Next Step →

> Install Certificate


> Installation Complete

※ 도메인 구매 없이 SSL 인증서 발급시 참고 - 세번째 방법 이용

SSL 보안 인증서 무료 발급 받기 [SSL For Free]

Get SSL Security Certificate Free [SSL For Free] SSL [Secure Socket Layer] 서버와 사용자(브라우저) 간의 통신을 할 경우 정보를 암호화 하고 도중에 해킹을 통해 정보가 유출이 된다고 하더라도 정보의 내용을 보호할 수 있는 보안 인증 솔루션 기술이라고 생각하시면 됩니다. 최근 브라우저 제공하는 업체마다 보안 인증서(SSL)가 적용된

<https://foxydog.tistory.com/39>



SSL For Free
Get SSL Security Certificate Free

5.3. Nginx

wsl 환경에 파일 복사

- .2. SSL 적용 에서 Download 한 압축 파일 압축 해제 후 파일 경로 확인
- `cp {파일경로}/파일명 ~/`
 - `cp /mnt/c/Users/SSAFY/Desktop/soolsol.site/certificate.crt ~/`
 - `cp /mnt/c/Users/SSAFY/Desktop/soolsol.site/ca_bundle.crt ~/`
 - `cp /mnt/c/Users/SSAFY/Desktop/soolsol.site/private.key ~/`

ubuntu 환경에 파일 복사

- `scp -i [pem키] [파일 이름] ubuntu@[접속할 도메인]:~/`
 - `scp -i K9E103T.pem certificate.crt private.key ca_bundle.crt ubuntu@k9e103.p.ssfy.io:~/`

ubuntu 환경 진입

- `ssh -i K9E103T.pem ubuntu@k9e103.p.ssfy.io`
- `nginx -v` 로 nginx 설치 여부 확인
 - nginx 설치 안되어 있을 경우 설치

```
$ sudo apt update
```

```
$ sudo apt install nginx
```

파일 위치 재설정

- `/etc/nginx` 에 `ssl` 폴더 생성

```
$ sudo mkdir /etc/nginx/ssl
```

- `/etc/nginx/ssl` 위치에 파일 옮기기

```
$ sudo mv certificate.crt /etc/nginx/ssl
$ sudo mv ca_bundle.crt /etc/nginx/ssl
$ sudo mv private.key /etc/nginx/ssl

# 해당 경로 진입
$ cd /etc/nginx/ssl

# 파일 있는지 확인
$ ls
```

.crt 파일 병합

- NGINX는 SSL 설치를 허용하기 위해 모든 .crt 파일을 병합해야 함.

```
$ cat certificate.crt ca_bundle.crt >> certificate.crt
```

Nginx 설정

- `/etc/nginx/sites-available` 로 이동 후 스크립트 작성

```
$ cd /etc/nginx/sites-available
$ sudo vim default
```

- (필요가 없다면) 기존 파일 내용 지워도 무방

```
upstream backend {
    server localhost:8080;
    server localhost:8081;
}

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

    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    root /var/www/html;

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

    server_name soolsool.site www.soolsool.site;

    ssl_certificate /etc/nginx/ssl_gabia/certificate.crt;
    ssl_certificate_key /etc/nginx/ssl_gabia/private.key;
    ssl_trusted_certificate /etc/nginx/ssl_gabia/ca_bundle.crt;

    location / {
        proxy_pass http://localhost:8082;
    }

    location /api {
        proxy_pass http://backend;
    }
}
```

```

    }
}

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

    listen 443 ssl;
    listen [::]:443 ssl;

    server_name k9e103.p.ssafy.io;

    ssl_certificate /etc/nginx/ssl_ssafy/certificate.crt;
    ssl_certificate_key /etc/nginx/ssl_ssafy/private.key;
    ssl_trusted_certificate /etc/nginx/ssl_ssafy/ca_bundle.crt;

    location / {
        proxy_pass http://localhost:8082;
    }

    location /api {
        proxy_pass http://backend;
    }
}

server {
    listen 80;
    server_name soolsool.site;
    return 301 https://soolsool.site$request_uri;
}

server {
    listen 80;
    server_name k9e103.p.ssafy.io;
    return 301 https://k9e103.p.ssafy.io$request_uri;
}

```

Port 열어주기

```

# ufw 상태 및 등록된 rule 확인
$ sudo ufw status numbered

# 포트 추가하기
$ sudo ufw allow 80
$ sudo ufw allow 443
$ sudo ufw allow 8080
$ sudo ufw allow 8081
$ sudo ufw allow 8082

```

무중단 배포 설정

```

$ cd /etc/nginx/sites-available
$ sudo vim soolsool.conf

```

```

server {
    include /etc/nginx/conf.d/service-url.inc;
    server_name soolsool.site;
    charset utf-8;

    location / {
        proxy_pass $service_url;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
    }
}

```

```

$ cd /etc/nginx/conf.d
$ sudo vim service-url.inc

```



```
set $service_url http://127.0.0.1:8080;
```

Nginx 서버 재가동

```
$ sudo service nginx restart  
  
# 스크립트 명령문 검토  
$ nginx -t
```

expo 배포 과정

0. 초기 설정

최신 EAS CLI 설치

```
npm install -g eas-cli
```

Expo 로그인 및 계정 확인

```
eas login  
  
# official.ohyes@gmail.com  
# wkdbf1010$  
  
# 현재 로그인 계정 확인  
eas whoami
```

프로젝트 구성 - **eas.json** 파일 생성 확인

```
eas build:configure
```

1. 개발용 배포

1.1. **.apk** 파일 배포(미리보기용)

eas.json 설정 - 각각의 환경에 맞게 **ENV** 추가

- Android : Application 설치를 위해 **apk** 파일로 생성

```
{  
  "cli": {  
    "version": ">= 5.6.0"  
  },  
  "build": {  
    "development": {  
      "channel": "development",  
      "developmentClient": true,  
      "distribution": "internal",  
      "android": {  
        "buildType": "apk"  
      },  
      "env": {  
        "REACT_APP_BACK_URL": "https://soolsool.site/api",  
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"  
      }  
    },  
    "preview": {  
      "distribution": "internal"  
    }  
  }  
}
```

```

    },
    "emulator": {
      "android": {
        "buildType": "apk"
      },
      "env": {
        "REACT_APP_BACK_URL": "https://soolsool.site/api",
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
      }
    },
    "simulator": {
      "ios": {
        "simulator": true
      },
      "env": {
        "REACT_APP_BACK_URL": "https://soolsool.site/api",
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
      }
    },
    "production": {
      "env": {
        "REACT_APP_BACK_URL": "https://soolsool.life/api",
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
      },
      "autoIncrement": true
    }
  },
  "submit": {
    "production": {}
  }
}

```

Android, ios 용 빌드

```

# Android Emulator 빌드
$ eas build -p android --profile emulator

# iOS Simulator 빌드
$ eas build -p ios --profile simulator

```

- 최신 빌드 실행

```

$ eas build:run -p android --latest
$ eas build:run -p ios --latest

```

1.2. development build(개발 확인용)

참고 문서

eas.json 설정 - 각각의 환경에 맞게 **ENV** 추가

- Android : Application 설치를 위해 **development** 환경 설정

```

{
  "cli": {
    "version": ">= 5.6.0"
  },
  "build": {
    "development": {
      "channel": "development",
      "developmentClient": true,
      "distribution": "internal",
      "android": {
        "buildType": "apk"
      },
      "env": {
        "REACT_APP_BACK_URL": "https://soolsool.site/api",

```

```

    "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
  },
  "preview": {
    "distribution": "internal"
  },
  "emulator": {
    "android": {
      "buildType": "apk"
    },
    "env": {
      "REACT_APP_BACK_URL": "https://soolsool.site/api",
      "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
    }
  },
  "simulator": {
    "ios": {
      "simulator": true
    },
    "env": {
      "REACT_APP_BACK_URL": "https://soolsool.site/api",
      "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
    }
  },
  "production": {
    "env": {
      "REACT_APP_BACK_URL": "https://soolsool.life/api",
      "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"
    },
    "autoIncrement": true
  }
},
"submit": {
  "production": {}
}
}

```

expo-dev-client 라이브러리 설치

```
$ npx expo install expo-dev-client
```

Android, ios 용 빌드


```

# Android development 빌드
$ eas build --profile development --platform android

# iOS development 빌드
$ eas build --profile development --platform ios

```

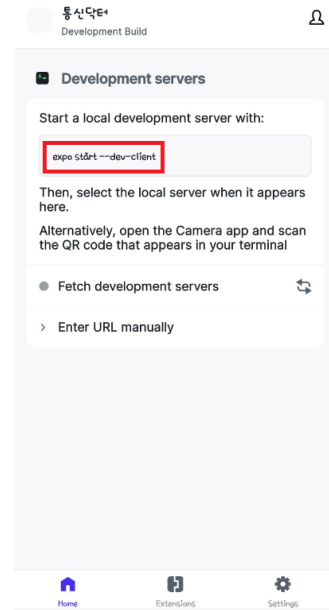
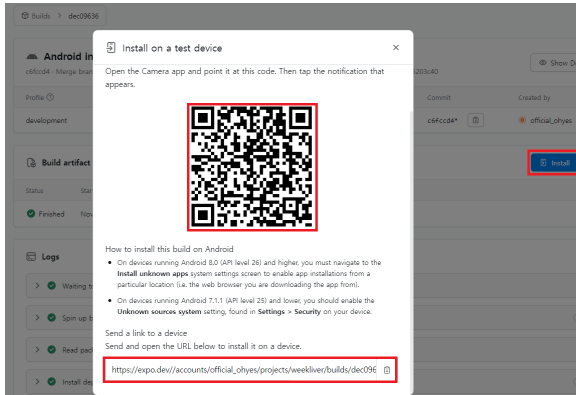
expo.development builds 에 가서 생성된 build 파일 확인 후 install

 **Android internal distribution build**
Show Details

c6fccd4 · Merge branch 'feature/FE/calendarDetailCSS' into develop/FE Change-Id: Idd5ecddfb64513c3e29cf8c07bcd728d05203c40

Profile ②	Runtime version	Channel ②	Version	Version code	Commit	Created by
development	1.0.0	development	1.0.0	2	c6fccd4*	official_ohyes

QR Code 혹은 하단의 URL로 이동 후 apk 파일 설치 및 어플 실행



2. Store 출시용 배포

2.1. 환경 설정

app.json 파일 설정

```
{
  "expo": {
    "name": "주간일기",
    "slug": "weeklivenote",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/icon.png",
    "userInterfaceStyle": "light",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#000733"
    },
    "assetBundlePatterns": ["**/*"],
    "ios": {
      "supportsTablet": true,
      "bundleIdentifier": "com.official_ohyes.weeklivenote",
      "buildNumber": "1.0.0"
    },
    "android": {
      "adaptiveIcon": {
        "foregroundImage": "./assets/adaptive-icon.png",
        "backgroundColor": "#000733",
        "package": "com.official_ohyes.weeklivenote",
        "versionCode": 1
      },
      "config": {
        "cleartext": {
          "usesCleartext": true
        }
      },
      "package": "com.official_ohyes.weeklivenote",
      "versionCode": 1
    },
    "web": {
      "favicon": "./assets/favicon.png"
    },
    "extra": {
      "eas": {

```

```

    "projectId": "b2fd2716-9256-4d89-8d34-f4db4cd13013"
  }
}
}
}

```

배포용 파일 build 실행

```

# Google Play Store용 빌드
$ eas build --platform android

# App Store용 빌드
$ eas build --platform ios

# 동시 빌드
$ eas build --platform all

```

2.2. 공통 사전 준비 사항

[1.0.0] Icon / Screen shot / Graphic design

1) Icon

- **Android** : 512 * 512 (1MB 이하) png(알파 포함)
- **ios** : 1024 * 1024

Icon 사이즈별 생성 사이트

2.3. Play Store 출시 사전 준비

참고 링크

1) Screen Shot

- 목업 기기 : 안드로이드 기기만 사용
- 정렬 방향 : 왼쪽 → 오른쪽
- 확장자 : PNG(알파 미포함) 또는 JPEG(라고 적혀 있지만, JPEG로 부탁해요 - jpeg, jpg 차이 없으니까 피그마에서 jpg로 export 하면 됩니다)
- 용량 : 장당 최대 8MB

핸드폰 스크린샷

- 장수 : 2장 ~ 8장
- 적정 크기 : 16: 9 비율 - 1080 × 1920 추천
- 가로/세로 길이 : 320px ~ 3840px 사이

7인치 태블릿 스크린샷 & 10인치 태블릿 스크린샷

- 장수 : 2장 ~ 8장
- 적정 크기 : 16: 9 비율 - 1440 × 1080 추천
- 가로/세로 길이 : 320px ~ 3840px 사이 / 1,080px ~ 7,680px 사이

2) 그래픽 이미지

- 1024 × 500 JPEG 또는 24비트 PNG(알파 미포함)

2.5. ONE Store 출시 사전 준비

1) Screen Shot

- Play Store와 동일

2) 그래픽 이미지

- 1024 × 578 PEG 또는 24비트 PNG(알파 미포함)

2.6. App Store 출시 사전 준비

3. Store 출시 후 업데이트

3.1. 업데이트용 파일 생성

1) 사용자 계정 설정

- 어플 루트 파일 경로에 아래 파일 추가

weeklivenote-dd71b33e934d.json

2) 환경 설정 및 빌드

참고 문서

app.json 설정 추가

- version / versionCode / buildNumber 수정

```
{
  "expo": {
    "name": "주간일기",
    "slug": "weeklivenote",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/icon.png",
    "userInterfaceStyle": "light",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#000733"
    },
    "assetBundlePatterns": ["**/*"],
    "ios": {
      "supportsTablet": true,
      "bundleIdentifier": "com.official_ohyes.weeklivenote",
      "buildNumber": "1.0.0"
    },
    "android": {
      "adaptiveIcon": {
        "foregroundImage": "./assets/adaptive-icon.png",
        "backgroundColor": "#000733",
        "package": "com.official_ohyes.weeklivenote",
        "versionCode": 1
      },
      "config": {
        "cleartext": {
          "usesCleartext": true
        }
      },
      "package": "com.official_ohyes.weeklivenote",
      "versionCode": 1
    },
    "web": {
      "favicon": "./assets/favicon.png"
    },
    "extra": {
      "eas": {
        "projectId": "b2fd2716-9256-4d89-8d34-f4db4cd13013"
      }
    }
  }
}
```

```
}  
}
```

eas.json 설정 추가

```
{  
  "cli": {  
    "version": ">= 5.6.0"  
  },  
  "build": {  
    "development": {  
      "channel": "development",  
      "developmentClient": true,  
      "distribution": "internal",  
      "android": {  
        "buildType": "apk"  
      },  
      "env": {  
        "REACT_APP_BACK_URL": "https://soolsool.site/api",  
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"  
      }  
    },  
    "preview": {  
      "distribution": "internal"  
    },  
    "emulator": {  
      "android": {  
        "buildType": "apk"  
      },  
      "env": {  
        "REACT_APP_BACK_URL": "https://soolsool.site/api",  
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"  
      }  
    },  
    "simulator": {  
      "ios": {  
        "simulator": true  
      },  
      "env": {  
        "REACT_APP_BACK_URL": "https://soolsool.site/api",  
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"  
      }  
    },  
    "production": {  
      "env": {  
        "REACT_APP_BACK_URL": "https://soolsool.life/api",  
        "RESTAPI_KEY": "b6ffe641bddfb241b3677f88c7462119"  
      },  
      "autoIncrement": true  
    }  
  },  
  "submit": {  
    "production": {  
      "android": {  
        "serviceAccountKeyPath": "../weeklivenote-dd71b33e934d.json"  
      }  
    }  
  }  
}
```

업데이트용 파일 build 실행

```
# Google Play Store용 빌드  
$ eas build --platform android  
  
# App Store용 빌드  
$ eas build --platform ios  
  
# 동시 빌드  
$ eas build --platform all
```

Play Store 자동 제출

```
$ eas submit -p android --latest
```

Expo.dev에서 production 모드로 변경

※ 참고 문서

내부 테스터 - <https://sodevly.github.io/react-native-bundletool/>

외부서비스

1. Kakao 로그인

1. 카카오 개발자 사이트 접속

Kakao Developers

카카오 API를 활용하여 다양한 어플리케이션을 개발해보세요. 카카오 로그인, 메시지 보내기, 친구 API, 인공지능 API 등을 제공합니다.

 <https://developers.kakao.com/>

kakao developers

2. 새 애플리케이션 생성

3. 좌측 제품 설정 → 카카오 로그인

4. 활성화 설정

5. 하단 Redirect URI에 서버 주소 추가

- <https://soolsoul.life/kakao/callback>

6. 카카오 클라이언트 ID ⇒ 앱키 - REST API 키

앱 키

네이티브 앱 키	34c
REST API 키	b6f
JavaScript 키	92a
Admin 키	5e4

7. 카카오 클라이언트 Secret 키 : 제품 설정 → 보안 → 키 발급

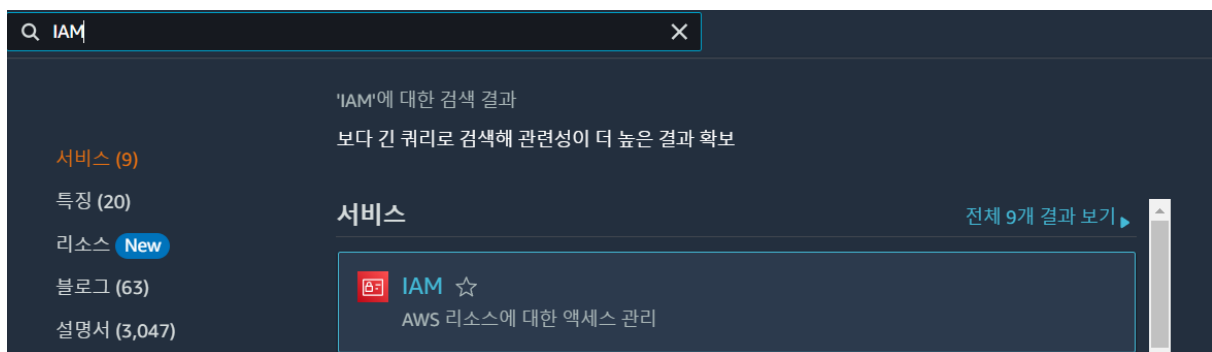
Client Secret

토큰 발급 시, 보안을 강화하기 위해 Client Secret을 사용할 수 있습니다. (REST API인 경우에 해당)

코드	50 <div></div> <div>재발급</div>
활성화 상태	사용함 <div>설정</div>

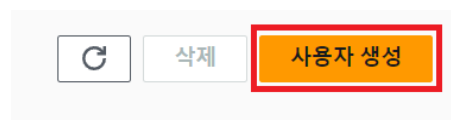
2. AWS S3 버킷

1. AWS 계정 생성
2. IAM 계정 생성 후 S3 권한 부여



IAM 대시보드

IAM 리소스				
이 AWS 계정의 리소스				
사용자 그룹	사용자	역할	정책	자격 증명 공급자
1	3	7	6	0



사용자 세부 정보

사용자 이름
test

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A-Z, a-z, 0-9 및 +, -, ., @, _ (하이픈)

☒ **AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항**
사용자에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center에서 액세스를 관리하는 것은 **모범 사례**입니다.

사용자에게 콘솔 액세스 권한을 제공하고 있습니까?

사용자 유형

- ☐ Identity Center에서 사용자 지정 - 권장
Identity Center를 사용하여 사용자에게 콘솔 액세스 권한을 제공하는 것이 좋습니다. Identity Center를 사용하면 AWS 계정 및 클라우드 애플리케이션에 대한 사용자 액세스를 중앙에서 관리할 수 있습니다.
- ☒ **IAM 사용자를 생성하고 싶음**
IAM 사용자, AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보 또는 비상 계정 액세스를 위한 백업 보안 인증 정보를 통해 프로그래밍 방식 액세스를 활성화해야 하는 경우에만 IAM 사용자를 생성하는 것이 좋습니다.

콘솔 암호

- ☒ **자동 생성된 암호**
사용자를 생성한 후 암호를 볼 수 있습니다.
- ☐ 사용자 지정 암호
사용자의 사용자 지정 암호를 입력합니다.

☐ 암호 표시

☒ **사용자는 다음 로그인 시 새 암호를 생성해야 합니다 - 권장**
사용자는 [IAMUserChangePassword](#) 정책을 자동으로 가져와 암호를 변경할 수 있도록 허용합니다.

이 IAM 사용자를 생성한 후 액세스 키 또는 AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보를 통해 프로그래밍 방식 액세스를 생성할 수 있습니다. 자세히 알아보기

취소 **다음**

권한 설정

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 직무별로 사용자의 권한을 관리하려면 그룹을 사용하는 것이 좋습니다. [자세히 알아보기](#)

권한 옵션

- ☐ 그룹에 사용자 추가
기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자 권한을 관리하는 것이 좋습니다.
- ☐ 권한 복사
기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.
- ☒ **직접 정책 연결**
관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.

권한 정책 (1/1148)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형

☒ S3 ☐ 모든 유형 15 개 일치

정책 이름	유형	연결된 엔터티
<input type="checkbox"/> AmazonDMSRedshiftS3Role	AWS 관리형	0
<input checked="" type="checkbox"/> AmazonS3FullAccess	AWS 관리형	1

3. IAM 계정으로 로그인 후 S3 버킷 생성

버킷 (1) 정보

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

이름	AWS 리전	액세스	생성 날짜
<input type="radio"/> soolsool	아시아 태평양(서울) ap-northeast-2	퍼블릭	2023. 10. 30. am 11:17:18 AM KST

4. 버킷 정보 입력

일반 구성

버킷 이름

soolsool

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

버킷 이름은 글로벌 네임스페이스 내에서 고유해야 하며 버킷 이름 지정 규칙을 따라야 합니다. [버킷 이름 지정 규칙 보기](#)

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록) 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 지점을 활성화합니다. 이 설정은 이 버킷 및 해당 객체에 적용됩니다. AWS에서는 모든 퍼블릭 액세스 지점을 활성화하는 것이 좋습니다. 이 설정을 적용하기 전에 퍼블릭 액세스가 있으며 애플리케이션이 클라이언트 측에서, 이 버킷 또는 다른 퍼블릭 버킷에서, 또는 수반의 퍼블릭 액세스가 필요한 경우 스트로지 사용 시에 맞게 이를 개형 설정을 사용자 지정할 수 있습니다. 자세한 설명은 여기를 참조하십시오.

- ☒ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 여러 개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

 - ☒ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

53은 새 버킷 또는 객체와 적용되는 퍼블릭 액세스 지점을 활성화합니다. 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 설정을 유지합니다. 이 설정은 ACL을 사용하여 53 리소스에 대한 퍼블릭 액세스를 부여하는 기존 권한을 변경하지 않습니다.
 - ☒ 임의의 액세스 제어 목록을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

53은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.
 - ☒ 새 퍼블릭 버킷 또는 객체 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

53은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 객체 지점 정책을 활성화합니다. 이 설정은 53 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.
 - ☒ 임의의 액세스 제어 및 모든 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 고가 계층 액세스 차단

53은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 객체 지점에 대한 퍼블릭 및 고가 계층 액세스를 무시합니다.

모든 퍼블릭 액세스 차단

모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다. 정책 변경 사이드 로스드와 같은 구체적인 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

- ☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있을 것을 알고 있습니다.

모든 퍼블릭 액세스 차단 설정 비활성화 관련 향후 권한 변경 사항

2023년 4월부터는 53 콘솔을 사용하여 버킷을 생성할 때 퍼블릭 액세스 차단 설정을 비활성화하기 위해 PutBucketPolicyAccessBlock 권한이 있어야 합니다. 자세한 설명은 여기를 참조하십시오.

- 권한 클릭 → 버킷 정책 편집 클릭

AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to [Amazon Web Services \(AWS\)](#) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service ☐ All Services ("*")

Use multiple statements to add permissions for more than one service.

Actions ☐ All Actions ("*")

Amazon Resource Name (ARN)

ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}.
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)

Step 3: Generate Policy

A *policy* is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Add one or more statements above to generate a policy.

- 스크롤을 아래로 내려 **Generate Policy** 버튼을 클릭한다.
 - 생성된 정책을 복사한뒤 정책 편집 부분에 붙여넣은 후 **변경 사항 저장** 버튼을 눌러 저장한다.
6. 사용자 설정
- **보안 자격 증명** 클릭 → **엑세스 키 발급** 클릭
 - 생성된 정책을 복사한뒤 정책 편집 부분에 붙여넣은 후 **변경 사항 저장** 버튼을 눌러 저장한다.