

Porting Manual

1. 프로젝트 구성
 - 1) 기술 스택
 - 2) 시스템 아키텍처
2. EC2 초기 서버 설정
 - 1) 발급받은 ec2 서버 접속
 - 2) 패키지 관리자 업데이트
 - 3) Docker 설치
 - 4) Docker-compose 설치
 - 5) JDK 설치
3. Jenkins CI 환경 구축
 - 1) Jenkins 설치
 - 2) Jenkins 초기설정
 - 3) Jenkins - GitLab 연동
 - 4) Webhook 설정
 - 5) Mattermost - Jenkins 연결
4. Nginx / SSL
 - 1) Nginx 설치
 - 2) SSL 발급
5. DB 설치
 - 1) MySQL
 - 2) Redis
 - 3) MongoDB
6. Jenkins CD 환경 구축
 - 1) Gradle 설치
 - 2) Docker hub image build&push 설정
 1. Docker hub Credential 등록
 2. Jenkins에서 Credentials 설정
 3. Jenkins sudo 권한 설정
 4. Dockerfile 작성
 5. Jenkins pipeline script
 - 4) Docker hub image pull & deploy
 1. spring project application_prod.yml 설정
 3. jenkins 환경변수
 3. 스프링 서버 배포 Shell Script 작성
 - 5) Nginx 설정
 1. 기본 포트 설정
 2. Nginx shell script 작성
7. Front-end 프로젝트 배포
8. AWS S3 설정
9. AWS CloudFront 설정
10. 외부 서비스
 1. KAKAO LOGIN
11. 시연 시나리오
 1. 메인 기능 시연
 2. 기타 기능 시연

1. 프로젝트 구성

1) 기술 스택

Backend - Spring

- IntelliJ IDE

- Java 11.0.14
- Springboot 2.7.7
- Spring Data JPA 2.7.7
- Spring Security 2.7.7
- Spring Validation 2.7.7
- Spring Web 2.7.7
- QueryDSL 5.0.0

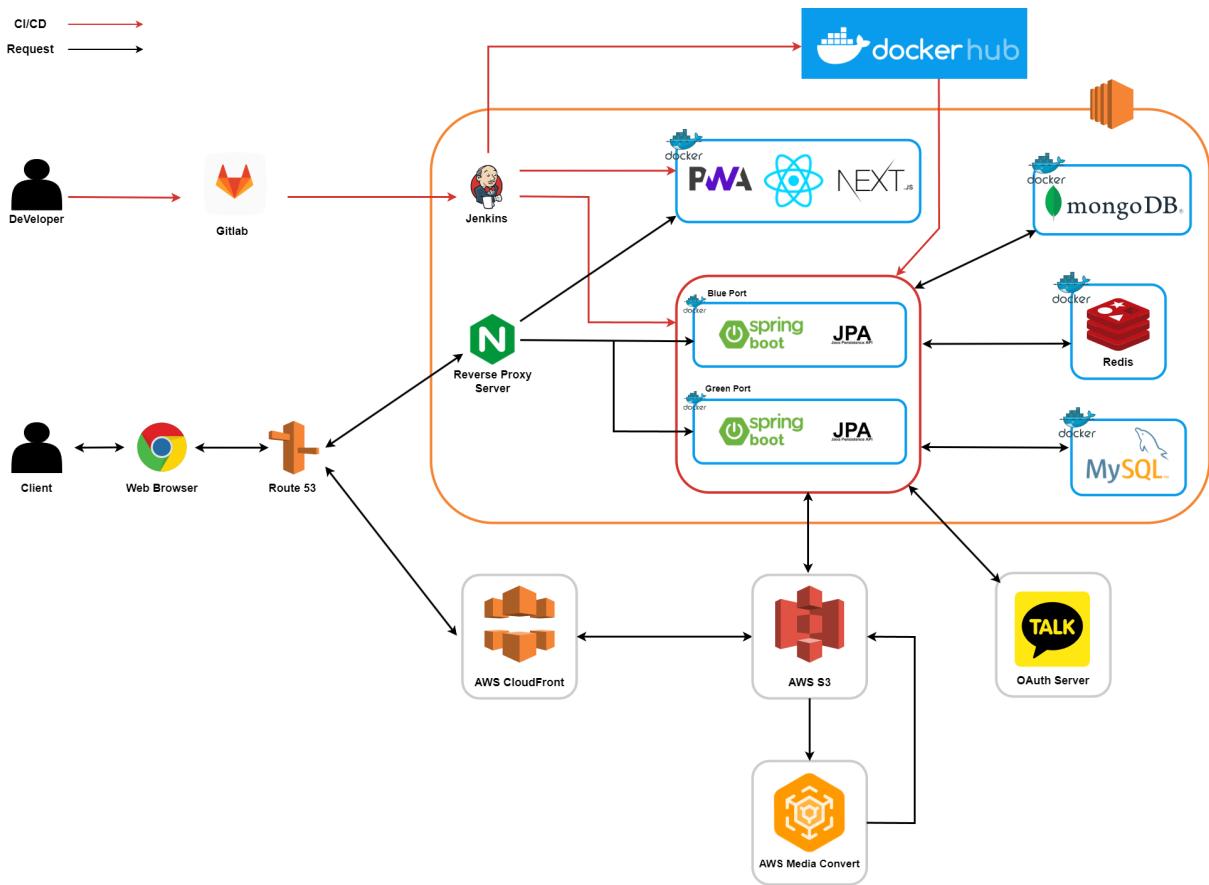
Infra

- AWS EC2
- AWS S3
- AWS CloudFront
- AWS Media Converter
- Jenkins 2.405
- Docker 23.0.6
- Docker-compose 23.0.6
- NGINX 1.18.0 (Ubuntu)
- SSL

Frontend

- Visual Studio Code IDE
- react 18.2.0
- redux 4.2.1
- typescript 5.0.4
- react-redux 8.0.5
- redux-persist 6.0.0
- styled-components 5.3.8
- styled-reset 4.4.5
- next: 13.3.0
- react-router-dom 6.8.2
- react-toastify 9.1.2
- react-responsive 9.0.2

2) 시스템 아키텍처



2. EC2 초기 서버 설정

1) 발급받은 ec2 서버 접속

```
ssh -i K8A203T.pem ubuntu@k8a2031.p.ssafy.io
```

2) 패키지 관리자 업데이트

```
sudo apt-get update
sudo apt-get upgrade
```

3) Docker 설치

```
# 오래된 버전 삭제
sudo apt-get remove docker docker-engine docker.io containerd runc

# 최신 버전 업데이트
sudo apt-get update

# apt가 HTTPS를 통해 repository를 이용하는 것을 허용할 수 있도록 해주는 패키지를 설치
sudo apt-get install \
    ca-certificates \
```

```

curl \
gnupg \
lsb-release

# 권한 추가
sudo mkdir -m 0755 -p /etc/apt/keyrings

# docker 공식 GPG key 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

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

# docker 설치
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io

docker --version
# Docker version 23.0.6

```

4) Docker-compose 설치

```

sudo apt-get update

sudo apt-get install docker-compose-plugin

docker compose version
# Docker Compose version v2.17.3

```

5) JDK 설치

```

# JDK 설치 - Jenkins는 java 기반으로 작동하기 때문에 java 설치 필요
sudo apt-get install openjdk-11-jdk

java -version
javac -version

```

3. Jenkins CI 환경 구축

1) Jenkins 설치

```

# Debian package repository of Jenkins to automate installation and upgrade.
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null

# add a Jenkins apt repository entry
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null

# Update local package index
sudo apt-get update
# sudo apt-get install fontconfig openjdk-11-jre

# install Jenkins
sudo apt-get install jenkins

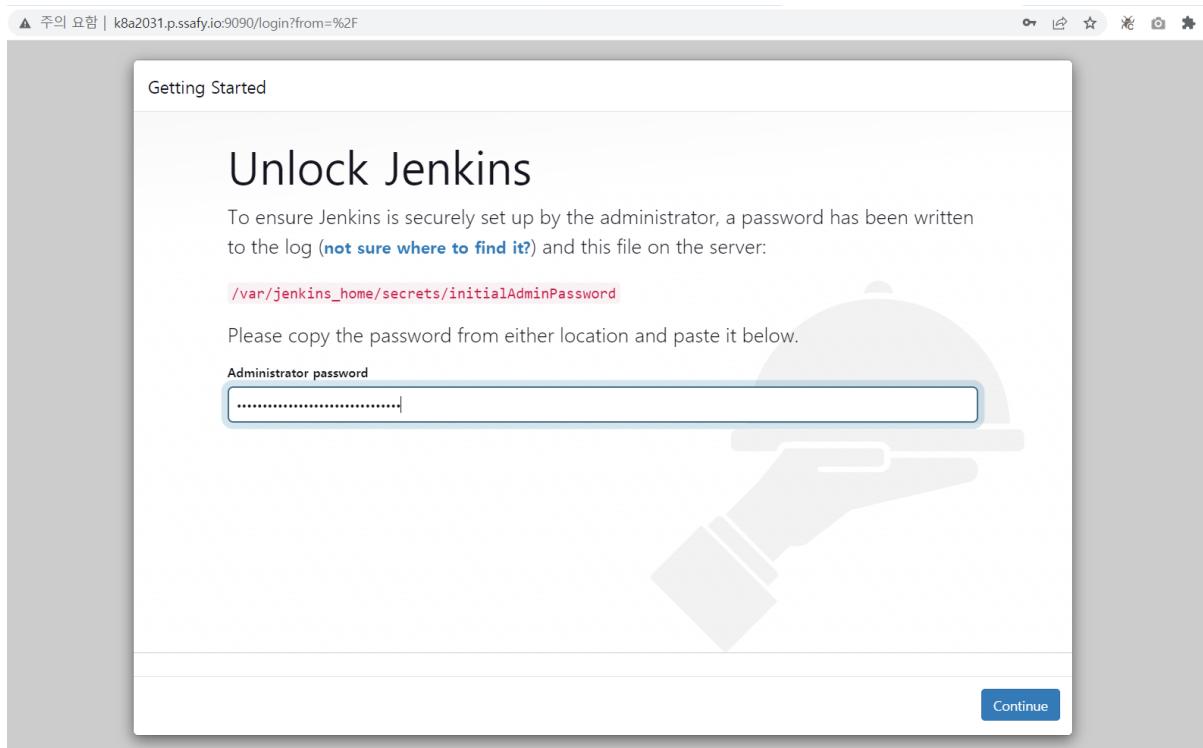
# jenkins
$ sudo systemctl status jenkins
$ sudo systemctl start jenkins
$ sudo systemctl reload jenkins

```

```
// 초기 비밀번호 확인  
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

2) Jenkins 초기설정

1. http://[domain]:8080 접속 후 초기 비밀번호 입력



2. 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.

Jenkins 2.387.3

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	...
✓ Timestamper	⌚ 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		...

```
OWASP Markup Formatter
**+ Structs
**+ Token Macro
Build Timeout
**+ Pipeline: Step API
**+ Credentials
**+ Plain Credentials
**+ Trilead API
**+ SSL Credentials
Credentials Binding
**+ SCM API
**+ Pipeline: API
**+ commons-lang3 v3.x Jenkins API
Timestamper
**+ Caffeine API
**+ Script Security
**+ JAXB
**+ SnakeYAML API
**+ Jackson 2 API
**+ commons-text API
**+ Pipeline: Supporting APIs
**+ Plugin Utilities API
**+ Font Awesome API
**+ Bootstrap 5 API
**+ JQuery3 API
**+ ECharts API
**+ - required dependency
```

Jenkins 2.387.3

3. Admin 계정 생성

Getting Started

Create First Admin User

계정명

암호

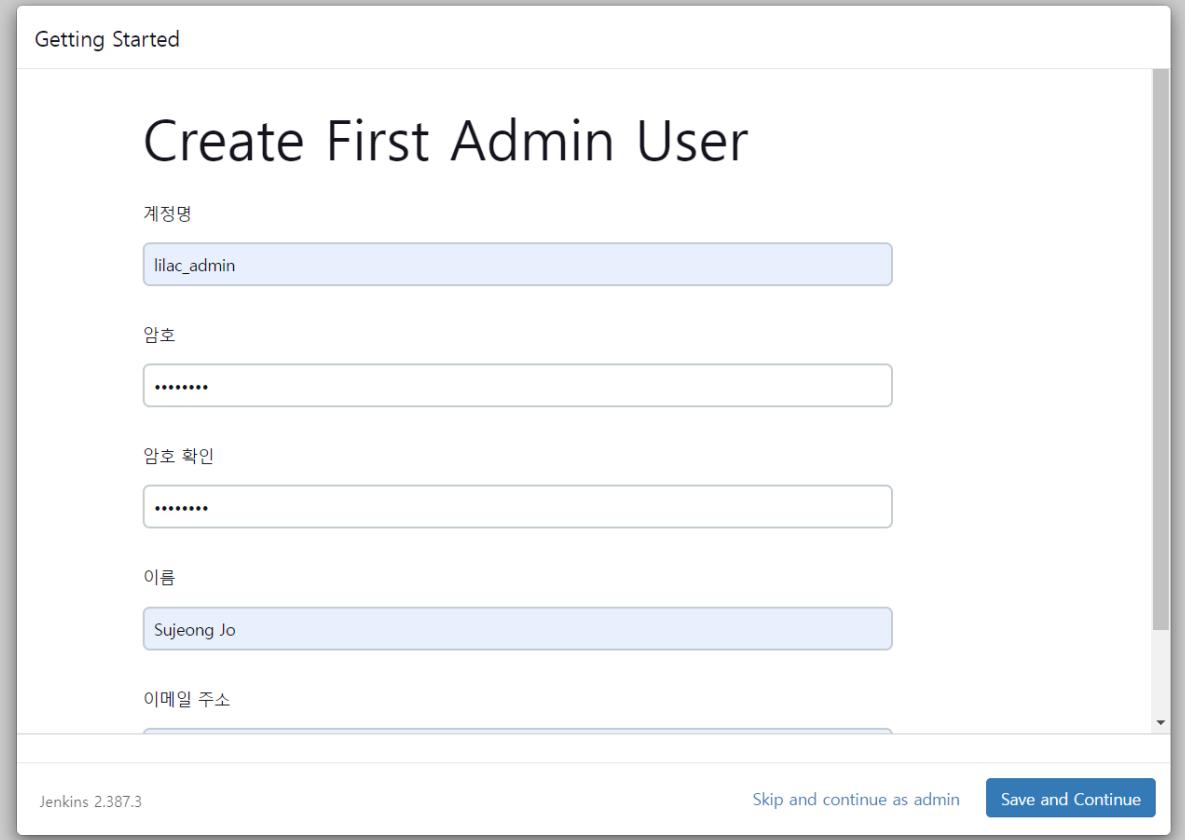
암호 확인

이름

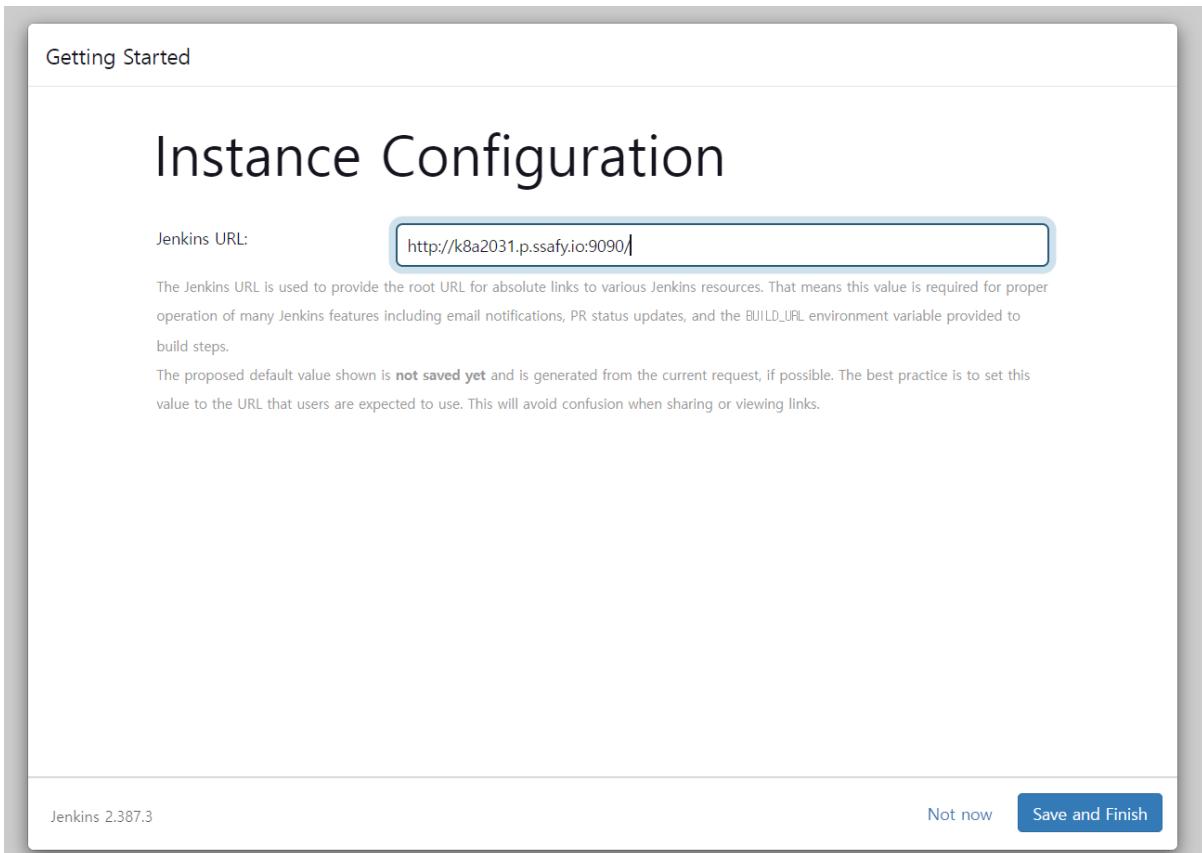
이메일 주소

Jenkins 2.387.3

Skip and continue as admin Save and Continue



4. Jenkins 접속 url 설정 (도메인:포트번호)



5. 추가 플러그인 설치

- DashBoard > Manager Jenkins > Plugin Manager > Available plugins
- gitlab, docker 상위 4개 항목 Install without restart 옵션으로 설치
 - GitLab
 - Generic Webhook Trigger
 - GitLab API
 - GitLab Authentication
 - Docker
 - Docker Commons
 - Docker Pipeline
 - Docker API

3) Jenkins - GitLab 연동

1. gitlab project access token 생성 (프로젝트 토큰: 팀장이 권한 풀어줘야 함)

Token name

jenkins_token

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Expiration date

2024-05-01

**Select scopes**

Scopes set the permission levels granted to the token. [Learn more](#).

 api

Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

 read_api

Grants read access to the API, including all groups and projects, the container registry, and the package registry.

 read_user

Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

 read_repository

Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

 write_repository

Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

Active project access tokens (1)

Token name	Scopes	Created	Last Used <small>?</small>	Expires	Role	Action
lilac_admin_token	api, read_api, read_repository	2 May, 2023	1 day ago	in 2 weeks	Maintainer	

2. Dashboard > Jenkins 관리 > Configure System > GitLab

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://gitlab.mydomain.com)

Credentials
API Token for accessing Gitlab

Add ▾

고급 ▾

3. Credential 생성

- ID, Description : 구별용으로 작성

Kind

Scope ?

API token

ID ?

Description ?

Add Cancel

- Test Connection Success 확인



4) Webhook 설정

1. new Item > Pipeline 프로젝트 생성

The screenshot shows the Jenkins dashboard with a search bar and notification icons. Below the header, it says "Dashboard > All >". A central dialog box is titled "Enter an item name" and contains a text input field with the value "lilac_backend_server", which is marked as a "Required field". Below this, there are two options: "Freestyle project" (represented by a box icon) and "Pipeline" (represented by a pipe icon). The "Freestyle project" section includes a descriptive text about its use in Jenkins.

2. Build Trigger 설정

- Build when a change is pushed to GitLab (check)
- url 저장 : GitLab webhook URL: http://k8a2031.p.ssafy.io:8080/project/lilac_backend_server
- [고급] > Secret Token > Generate (gitlab에서 연결할 때 필요)

Secret token ?

Generate

- Save

3. GitLab 설정

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://example.com/trigger-ci.json

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

Secret token

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.

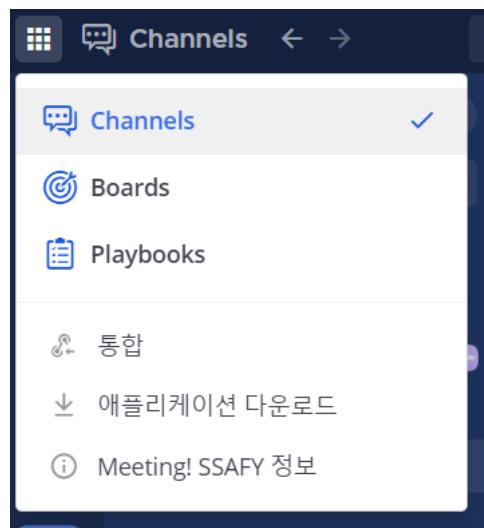
Comments

- Settings > Webhooks
- URL : GitLab webhook URL
- Secret token : Jenkins에서 발급받은 token
- Trigger : push events(연결할 branch 적기)
- add webhook
- Test > push events

(i) Hook executed successfully: HTTP 200

5) Mattermost - Jenkins 연결

1. mattermost > 통합



2. 전체 Incoming webhook 추가

☺ 커스텀 이모지

▣ 통합

전체 Incoming Webhook

전체 Outgoing Webhook

슬래시 명령어

통합

앱 디렉토리를 방문하여 Mattermost에 대한 자체 호스팅인 third-party 앱과 통합기능을 찾으십시오.



전체 Incoming Webhook

Incoming Webhook은 외부 시스템에서 메시지를 받을 수 있게합니다.



전체 Outgoing Webhook

Outgoing Webhook은 외부 시스템에 메시지를 보내고 응답받을 수 있게합니다.



슬래시 명령어

슬래시 명령어는 외부에 연결한 서비스에 이벤트를 보냅니다.

전체 Incoming Webhook > 추가

제목

웹훅 설정 페이지에 대해 최대 64자
의 제목을 지정합니다.

설명

웹훅에 대한 설명을 입력하세요.

채널

웹훅 페이로드를 수신할 기본 채널(공
개 혹은 비공개)입니다. 비공개 채널
로 웹훅을 설정할 때에는 그 채널에
속해있어야 합니다.

이 채널로 고정

설정되면, 들어오는 웹훅은 선택된 채
널에만 게시할 수 있습니다.

- 저장 후 URL 복사하기

4. Jenkins > Dashboard > Jenkins 관리 > Plugin Manager

Mattermost Notification 설치

The screenshot shows the Jenkins Plugin Manager interface. On the left, there are tabs for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', and 'Advanced settings'. A search bar at the top right contains the text 'matter'. Below the search bar, a table lists the 'Mattermost Notification' plugin. The table has columns for 'Install' (with a checked checkbox), 'Name' (set to 'Mattermost Notification 3.1.2'), and 'Released' (set to '5 mo 12 days ago'). A note below the table states: 'This plugin is a Mattermost notifier that can publish build status to Mattermost channels.'

5. Dashboard > Jenkins 관리 > Configure System > Global Mattermost Notifier Settings

Global Mattermost Notifier Settings

Endpoint [?](#)

Channel [?](#)

Icon to use [?](#)

Build Server URL [?](#)

http://k8a2031.p.ssafy.io:9090/

Mattermost Custom Proxy Settings [▼](#)

6. Dashboard > my_project > Configure > 빌드 후 조치

빌드 후 조치

≡ Mattermost Notifications ✖

- Notify Build Start
- Notify Aborted
- Notify Failure
- Notify Not Built
- Notify Success
- Notify Unstable
- Notify Back To Normal

고급 [▼](#)

4. Nginx / SSL

1) Nginx 설치

```
# Nginx 설치  
sudo apt update  
sudo apt install nginx  
  
# Nginx 실행 확인  
sudo systemctl start nginx  
sudo systemctl status nginx  
sudo systemctl stop nginx
```

2) SSL 발급

```
sudo apt-get update  
  
sudo apt-get install python3-certbot-nginx  
  
# route53 A 호스팅 영역 ip 설정(설정하는 과정에서 DNS 설정 에러 발생)  
sudo certbot --nginx -d [사용할 domain]  
  
# public ip 출력 (SSAFY에서 즐겨 확인)  
curl http://[public ip]/latest/meta-data/public-ipv4
```

5. DB 설치

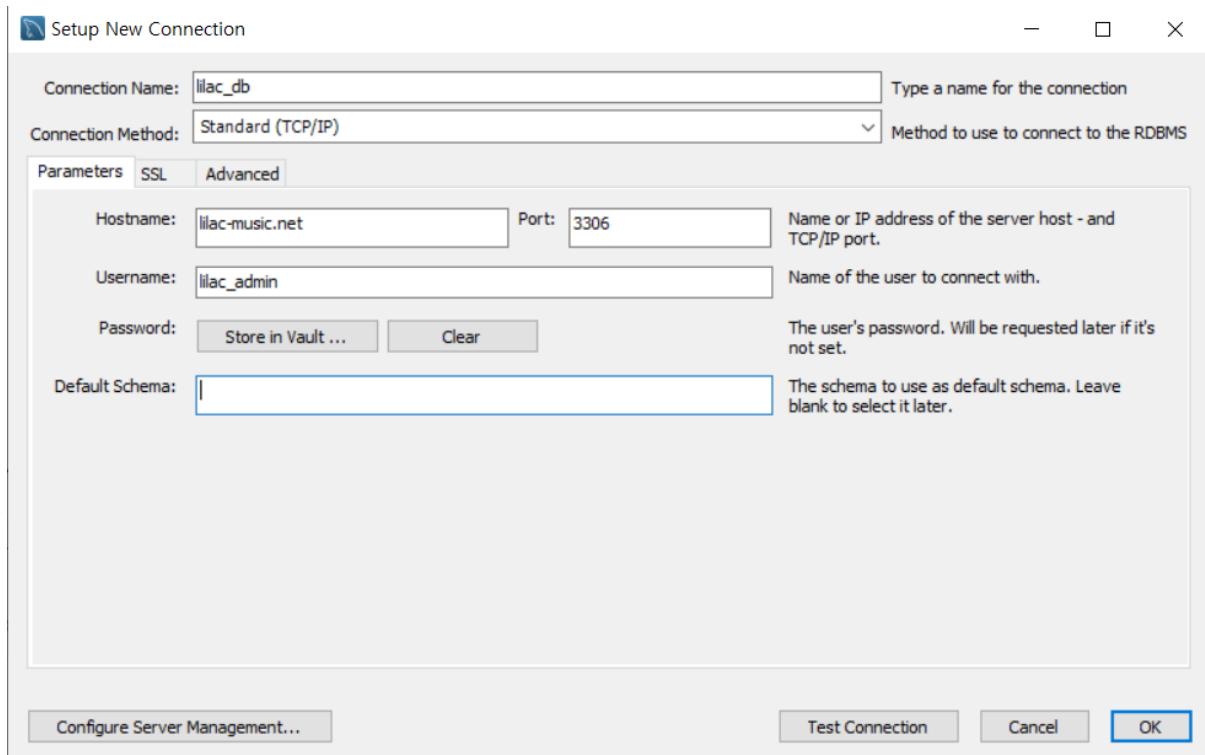
1) MySQL

1. 설치

```
# mysql image 다운로드  
docker pull mysql:latest  
  
# docker run  
docker run -d -p 3306:3306 --name mysql_container -e MYSQL_ROOT_PASSWORD=[rootpassword] -d mysql:latest
```

```
# mysql shell  
docker exec -it mysql_container /bin/bash  
mysql -u root -p  
  
# db 생성  
SHOW DATABASES;  
CREATE DATABASE lilac_db;  
  
# user 생성 및 권한 부여  
create user '아이디'@'%' identified by '비밀번호';  
grant all privileges on *.* to '아이디'@'%';  
flush privileges;
```

2. workbench 접속 확인



2) Redis

```
# redis image 다운로드
docker pull redis:latest

# Warning: no config file specified, using the default config. (redis.conf 설정파일 없음 만들어줘야 함)
docker run -d -p 6379:6379 -v /etc/redis:/usr/local/etc/redis --name redis_container redis redis-server /usr/local/etc/redis/redis.conf
```

```
# 보안 설정
# redis.conf 파일에서 username-password 설정
ubuntu@ip-172-26-9-243:~$ sudo vi /etc/redis/redis.conf

# redis.conf
user [username] on >[password] ~* +@all
requirepass [password]

# redis_container restart (설정 적용)
docker restart redis_container

# redis-cli 접속
docker exec -it redis_container redis-cli

# 설정한 유저정보로 접속
127.0.0.1:6379>
auth [username] [password]
```

3) MongoDB

```
# mongodb 설치
docker pull mongo:latest
docker run --name mongodb_container -v ~/data:/data/db -d -p 27017:27017 mongo:latest
```

```

# 보안 설정

# 서버 접속
docker exec -it mongodb_container /bin/bash

# 접속 shell에서 mongosh 입력 (mongo 하면 안됨 -> 버전 문제)
root@30ef45a2fb34:/# mongosh

# admin 으로 접속
test>
use admin

# 계정 설정
admin>
db.createUser({ user: "[user]", pwd: "[pw]", roles: [{role: "userAdminAnyDatabase", db: "admin"}]})

# 재시작
exit
docker restart mongodb_container

# mongodb 접속
docker exec -it mongodb_container /bin/bash

# mongod.conf 파일 설정 (mongod.conf.orig 파일만 있어서 복사해서 사용)
root@30ef45a2fb34:/etc# cp /etc/mongod.conf.orig /etc/mongod.conf

# vim 설치
apt-get update
apt-get install vim

# mongod.conf 파일 security 설정
vi mongod.conf

security:
  authorization: enabled

# 파일 적용
root@30ef45a2fb34:/etc# mongod --config mongod.conf

# 사용자 로그인
root@30ef45a2fb34:~# mongosh -u "[user]" -p "[pw]" --authenticationDatabase "admin"

```

6. Jenkins CD 환경 구축

1) Gradle 설치

Jenkins > Jenkins 관리 > Global Tool Configuration > Gradle

Add Gradle > 해당 프로젝트 gradle 버전 입력

Gradle

Gradle installations

List of Gradle installations on this system

Add Gradle

Gradle

name ?

gradle

Install automatically ?

Install from Gradle.org

Version

Gradle 7.6.1

Add Installer ▾

2) Docker hub image build&push 설정

1. Docker hub Credential 등록

- Access token 발급
- Docker Hub > Account Settings > Security > New Access Token > Access Token description > Generate > token 복사

2. Jenkins에서 Credentials 설정

Dashboard > Jenkins 관리 > Credentials > System > Global credentials > Add credentials

Username : Dockerhub id

password : Dockerhub token

id : credential 이름 (원하는대로: dockerhub_connect)

New credentials

Kind

Username with password

Scope ?

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

Username ?

Treat username as secret ?

Password ?

ID ?

Create

3. Jenkins sudo 권한 설정

- 설정 안하면 pipeline에서 sudo 명령어 실행 시 pwd 입력하라고 함

```
# root 로그인  
sudo su  
  
# 위치 이동  
sudo vi /etc/sudoers  
  
# 파일 수정 권한 주기  
chmod u+w sudoers  
  
# 권한 확인  
ls -l | grep sudoers  
  
# 파일 수정  
sudo vi sudoers  
  
# jenkins 로그인 권한 추가  
# %sudo ALL=(ALL:ALL) ALL 아랫줄에 추가 후 저장  
jenkins ALL=(ALL) NOPASSWD: ALL  
  
# +) sudoers 파일 사용자 권한 해제  
chmod u-w sudoers  
  
# root 권한 빠져나오기  
exit  
  
# jenkins 재시작  
sudo service jenkins restart
```

4. Dockerfile 작성

- 파이프라인 프로젝트 폴더 하위에 Dockerfile 생성 (Backend)

```
FROM adoptopenjdk/openjdk11:alpine-jre  
COPY build/libs/*.jar app.jar
```

```

EXPOSE 8080
ENV USE_PROFILE prod
ENTRYPOINT ["java","-Dspring.profiles.active=${USE_PROFILE}","-jar","/app.jar"]

```

5. Jenkins pipeline script

```

pipeline {
    agent any
    environment {
        DOCKER_REPOSITORY = "dub2y/lilac"
        DOCKERHUB_CREDENTIALS = credentials('dockerhub_connect')
    }

    stages {
        stage('Clone') {
            steps {
                git branch: "devops/feature/build",
                credentialsId: "gitlab_source_code",
                url: "https://lab.ssafy.com/s08P31A203.git"
            }
        }

        stage('Clean build'){
            steps {
                sh '''
                    cd backend/LILAC-Backend
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    ...
                '''
            }
        }

        stage('Docker build') {
            steps {
                echo 'docker build...'
                sh ''#!/bin/bash
                cd backend/LILAC-Backend
                PREV_IMAGE=docker images --filter=reference='dub2y/*' -q
                echo "prev IMAGE : \$PREV_IMAGE"
                if [[ -n \$PREV_IMAGE ]]; then
                    echo "prev image delete"
                    docker rmi \$PREV_IMAGE
                fi
                echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin
                docker build . -t $DOCKER_REPOSITORY --no-cache
                docker push $DOCKER_REPOSITORY
                ...
            }
        }

        stage('Deploy Spring Server') {
            steps {
                sh '''
                    cd Infra
                    bash deploy.sh
                    ...
                '''
            }
        }

        stage('Switch Nginx') {
            steps {
                sh '''
                    cd Infra
                    bash switch.sh
                    ...
                '''
            }
        }
    }
}

```

4) Docker hub image pull & deploy

1. spring project application_prod.yml 설정

```

spring:
  config:
    activate:
      on-profile: prod
    # mvc:
    #   pathmatch:
    #     matching-strategy: ant_path_matcher

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://lilac-music.net:3306/lilac_db?serverTimezone=Asia/Seoul
    username: ${MYSQL_USER}
    password: ${MYSQL_PASSWORD}

  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    generate-ddl: true
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        open-in-view: true

  data:
    mongodb:
      host: lilac-music.net
      port: 27017
      authentication-database: admin
      database: lilac_mongodb
      username: ${MONGO_USER}
      password: ${MONGO_PASSWORD}

  redis:
    lettuce:
      pool:
        max-active: 5
        max-idle: 5
        min-idle: 2
      host: lilac-music.net
      port: 6379
      username: ${REDIS_USER}
      password: ${REDIS_PASSWORD}

  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 100MB

  security:
    user:
      name: ${ADMIN_NAME}
      password: ${ADMIN_PASSWORD}
      roles: ADMIN
    oauth2:
      client:
        registration:
          kakao:
            client-id: ${KAKAO_ID}
            redirect-uri: "https://lilac-music.net/api/oauth2/kakao"
            client-authentication-method: POST
            client-secret: ${KAKAO_SECRET}
            authorization-grant-type: authorization_code
            scope:
              - profile.nickname
              - profile.image
              - account.email
            client_name: kakao

        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attributed: id

  jwt:
    secretKey: ${JWT_SECRET_KEY}
    access:
      expiration: 3600000 # 1시간
      header: Authorization
    refresh:
      expiration: 1209600000 # 2주
      header: Authorization-refresh

```

```

logging:
  level:
    root: info
    com.lilacmusic: debug

cloud:
  aws:
    access_key: ${AWS_ACCESS_KEY_ID}
    secret_key: ${AWS_SECRET_KEY}
    s3:
      bucket: ${S3_BUCKET}
    region:
      static: ${AWS_REGION}
    stack:
      auto: false
    mediaconvert:
      endpoint: ${MEDIACONVERT_ENDPOINT}
      role: ${IAM_ROLE}
    cloudfront:
      url_prefix: ${CLOUDFRONT_PREFIX}

management:
  endpoint:
    health:
      enabled: true
    info:
      enabled: true
  endpoints:
    web:
      base-path: /api/v1/actuator
      exposure:
        include: health, info
  enabled-by-default: false

```

3. jenkins 환경변수

```

ADMIN_NAME
ADMIN_PASSWORD
AWS_ACCESS_KEY_ID
AWS_SECRET_KEY
CLOUDFRONT_PREFIX
IAM_ROLE
JWT_SECRET_KEY
KAKAO_ID
KAKAO_SECRET
MEDIA CONVERT-ENDPOINT
MONGO_PASSWORD
MONGO_USER
MYSQL_PASSWORD
MYSQL_USER
REDIS_PASSWORD
REDIS_USER
S3_BUCKET
AWS_REGION=${AWS_REGION}

```

3. 스프링 서버 배포 Shell Script 작성

- jenkins workspace 루트 디렉토리에 Infra 폴더 생성 후 deploy.sh 생성

```

#!/usr/bin/env bash
echo "> $DOCKER_REPOSITORY"
sudo true > RESULT
sudo chmod 666 /var/run/docker.sock

# 현재 사용하고 있는 포트와 유동 상태인 포트를 체크한다.
RESPONSE=$(curl -s localhost:8082/api/v1/actuator/health)

echo "> RESPONSE : \"$RESPONSE"
IS_ACTIVE=$(echo ${RESPONSE} | grep 'UP' | wc -l)
echo "> IS_ACTIVE \"$IS_ACTIVE"
if [ $IS_ACTIVE -eq 1 ];
then
  IDLE_PORT=8083
  IDLE_PROFILE=prod-green
  CURRENT_PORT=8082

```

```

CURRENT_PROFILE=prod-blue
else
  IDLE_PORT=8082
  IDLE_PROFILE=prod-blue
  CURRENT_PORT=8083
  CURRENT_PROFILE=prod-green
fi

echo "> 다음 사용할 포트" $IDLE_PORT
echo "> 다음 사용할 프로필 " $IDLE_PROFILE

# 도커 허브에서 PULL을 한다.
docker pull $DOCKER_REPOSITORY
docker rm $(docker ps --filter status=exited -q)
docker rmi -f $(docker images -f "dangling=true" -q)

# 도커를 통해 컨테이너를 실행시킨다.
echo "> sudo nohup docker run --name lilac_backend_container -p $IDLE_PORT:8080 -e \"USE_PROFILE=prod\" $DOCKER_REPOSITORY > nohup.out 2>&1"
sudo nohup docker run --name $IDLE_PROFILE -p $IDLE_PORT:8080 -e ADMIN_NAME=${ADMIN_NAME} -e ADMIN_PASSWORD=${ADMIN_PASSWORD} -e AWS_AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} -e AWS_AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
echo "> 60초동안 5초마다 Health Check"

for RETRY in {1..12}
do
  for i in {1..5} ;
  do
    echo "> Health Check까지 " $(( 6 - i))초 남음

    sleep 1
  done

  RESPONSE=$(curl -s localhost:$IDLE_PORT/api/v1/actuator/health)
  IS_ACTIVE=$(echo ${RESPONSE} | grep 'UP' | wc -l)
  if [ $IS_ACTIVE -ge 1 ]; then
    echo "> Health Check Success"
    echo "$IDLE_PORT" $IDLE_PORT
    echo "$IDLE_PORT" > RESULT
    exit 0
  else
    echo "> Health Check Failed"
    echo "> Health Check RESPONSE : " ${RESPONSE}
  fi
  if [ $RETRY -eq 10 ]; then
    echo "> Health Check Failed"
    echo "FAIL" > RESULT
  fi
done

exit 1

```

5) Nginx 설정

1. 기본 포트 설정

```

sudo cd /etc/nginx/conf.d
sudo vim port.conf

```

```

# port.conf
set $active_server BLUE;

```

2. Nginx shell script 작성

- jenkins workspace 루트 디렉토리 Infra 폴더에 switch.sh 생성

```

#!/usr/bin/env bash
# 현재 사용중인 포트를 확인
RESPONSE=$(curl -s -k -L lilac-music.net/api/v1/actuator/health)
echo "> RESPONSE : \"$RESPONSE"

IS_ACTIVE=$(echo ${RESPONSE} | grep 'UP' | wc -l)
echo "> IS_ACTIVE" "$IS_ACTIVE"
CURRENT_PORT=$(curl -k -L lilac-music.net/port | grep 'BLUE' | wc -l)
echo "현재 구동중인 포트: " "$CURRENT_PORT"

```

```

if [ "$IS_ACTIVE" -eq 1 ];
then
    if [ "$CURRENT_PORT" -eq 1 ];
    then
        IDLE_PORT=8083
        IDLE_PROFILE=GREEN
        CURRENT_PORT=8082
        CURRENT_PROFILE=BLUE
    else
        IDLE_PORT=8082
        IDLE_PROFILE=BLUE
        CURRENT_PORT=8083
        CURRENT_PROFILE=GREEN
    fi
else
    IDLE_PORT=8082
    IDLE_PROFILE=BLUE
    CURRENT_PORT=8083
    CURRENT_PROFILE=GREEN
fi
echo "전환할 포트: " "$IDLE_PORT"

echo "> 포트 세팅 변경"
echo "set \$active_server $IDLE_PROFILE;" | sudo tee /etc/nginx/conf.d/port.conf
echo "> 기존 컨테이너 삭제"
echo "> docker kill $(sudo docker ps -qf publish=$CURRENT_PORT)"
sudo docker kill $(docker ps -qf publish=$CURRENT_PORT)
echo "> nginx 재시작"
sudo systemctl reload nginx

```

- custom.conf 파일 설정

```

# upstream 설정
upstream BLUE{
    server localhost:8082 weight=100 max_fails=3 fail_timeout=3s;
}

upstream GREEN{
    server localhost:8083 weight=100 max_fails=3 fail_timeout=3s;
}

server {
    include /etc/nginx/conf.d/port.conf; # 포트 정보 가져오기

    location / {
        proxy_pass http://localhost:3000; #front-end
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location /_next/webpack-hmr {
        proxy_pass http://localhost:3000/_next/webpack-hmr;
        proxy_http_version 1.1;
        # https websocket
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }

    location /api {
        proxy_http_version 1.1;
        proxy_pass http://$active_server; #back-end
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location ~* /api/oauth/kakao {
        proxy_pass http://$active_server/oauth2/authorization/kakao?${args}; #kakao login
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location ~* /api/oauth2/kakao {

```

```

proxy_pass http://$active_server/login/oauth2/code/kakao?${args}; #kakao login
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'upgrade';
proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;
}

location /api/v1/actuator/health {
    proxy_pass http://$active_server;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}

location /nginx_status{
    stub_status on;
    access_log off;
    allow 127.0.0.1;
    deny all;
}

location = /port {
    add_header Content-Type text/plain;
    return 200 '$active_server';
}

listen 443 ssl; # managed by Certbot
server_name lilac_music.net;
ssl_certificate /etc/letsencrypt/live/lilac-music.net/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/lilac-music.net/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

client_max_body_size 100M; #client가 보내는 파일 크기 설정 (기본이 10M)
}

server {
    listen 80;
    server_name lilac-music.net www.lilac-music.net;
    return 301 https://lilac-music.net$request_uri;
}

```

- Nginx 적용

```
sudo systemctl reload nginx
```

7. Front-end 프로젝트 배포

1. 새로운 Item 놀러서 freestyle project 생성

Enter an item name

» 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.

2. GitLab 연동

- Dashboard > [project name] > Configuration

None

Git [?](#)

Repositories [?](#)

Repository URL [?](#) ✖
https://lab.ssafy.com/s08-final/S08P31A203.git

Credentials [?](#)
suz.dev33/******** (lilac_project) ▼

Add ▾

고급 ▾

Add Repository

Branches to build [?](#)

Branch Specifier (blank for 'any') [?](#) ✖
release/0.0.7

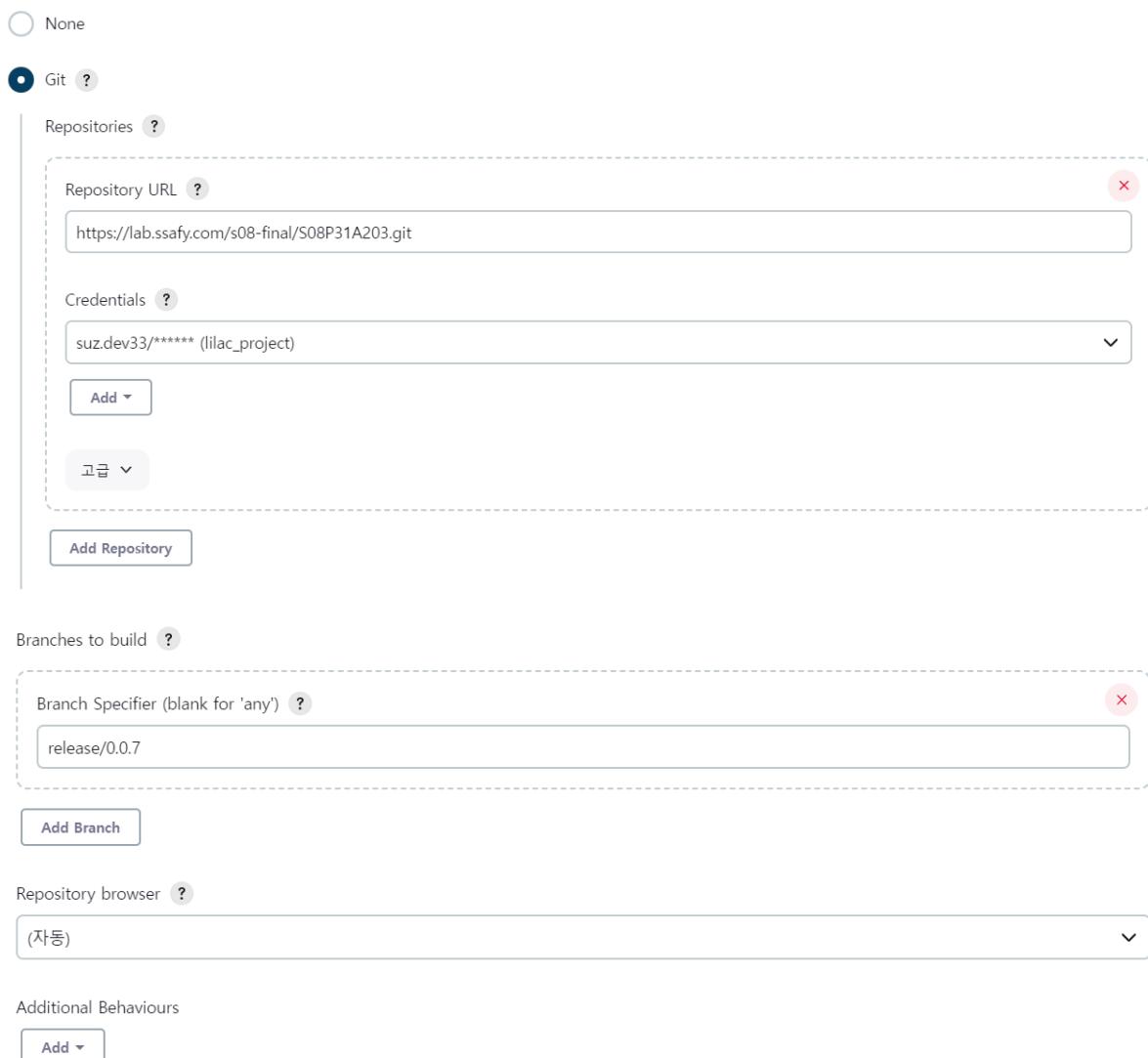
Add Branch

Repository browser [?](#)

(자동) ▼

Additional Behaviours

Add ▾



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://k8a2031.p.ssafy.io:8080/project/lilac_project ?

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

Rebuild open Merge Requests

Never

Approved Merge Requests (EE-only)

Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급 ▾

- 고급 > Secret token > Generate

3. Webhook 설정

- Gitlab > Settings > Webhooks

URL

http://k8a2031.p.ssafy.io:8080/project/lilac_project

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

Secret token

.....

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

Trigger

- Push events

release/0.0.7

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.

http://k8a2031.p.ssafy.io:8080/project/lilac_project

Test ▾

Edit

Delete

Push Events SSL Verification: enabled

4. Dockerfile 작성

```
# Stage 1 - the build process
# base image
FROM node:18.12.1-alpine as build-deps
# set working directory
WORKDIR /usr/src/app
# copy package files and install dependencies
COPY package*.json ./
RUN npm install
# copy app files
COPY . .
# build app
RUN npm run build

# Stage 2 - the production environment
FROM node:18.12.1-alpine
WORKDIR /usr/src/app
ENV NODE_ENV production
COPY --from=build-deps /usr/src/app/package*.json ./
RUN npm install --only=production
COPY --from=build-deps /usr/src/app/.next ./next
COPY --from=build-deps /usr/src/app/public ./public
COPY --from=build-deps /usr/src/app/server.ts ./server.ts
# expose port
EXPOSE 3000
# start command
CMD ["npm", "start"]
```

5. Jenkins Execute shell 작성

```

cd ../../frontend
docker stop lilac_frontend_container | true
docker rmi lilac_frontend | true
docker build -t lilac_frontend .
docker run --rm --name lilac_frontend_container -d -p 3000:3000 lilac_frontend

```

8. AWS S3 설정

1. 버킷 생성

ACL(액세스 제어 목록)

다른 AWS 계정에 기본 읽기/쓰기 권한을 부여합니다. 자세히 알아보기 [\[\]](#)

편집

i 이 버킷에는 객체 소유권에 대해 버킷 소유자 적용 설정이 적용되어 있습니다.
버킷 소유자 적용이 적용된 경우 버킷 정책을 사용하여 액세스를 제어합니다. 자세히 알아보기 [\[\]](#)

퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요할 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. 자세히 알아보기 [\[\]](#)

편집

모든 퍼블릭 액세스 차단

활성화

▼ 이 버킷의 개별 퍼블릭 액세스 차단 설정

- 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.
- 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.
- 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.
- 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

2. 버킷 정책 설정 (권한 세부 설정)

```
{
  "Version": "2012-10-17",
  "Id": "Policy1682924058680",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipalReadOnly",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
    }
  ]
}
```

```
        "Resource": "arn:aws:s3:::lilac-basket/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn": "arn:aws:cloudfront::387349920590:distribution/E3UE2KTWVT7N5Z"
            }
        }
    ]
}
```

9. AWS CloudFront 설정

설정

경로 패턴 | 정보

기본값(*)

원본 및 원본 그룹

lilac-basket.s3.ap-northeast-2.amazonaws.com ▾

자동으로 객체 압축 | 정보

No
 Yes

뷰어

뷰어 프로토콜 정책

HTTP and HTTPS
 Redirect HTTP to HTTPS
 HTTPS only

허용된 HTTP 방법

GET, HEAD
 GET, HEAD, OPTIONS
 GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

뷰어 액세스 제한

뷰어 액세스를 제한하는 경우 뷰어는 CloudFront 서명된 URL 또는 서명된 쿠키를 사용하여 사용자의 콘텐츠에 액세스해야 합니다.

No
 Yes

캐시 키 및 원본 요청

캐시 정책 및 원본 요청 정책을 사용하여 캐시 키 및 원본 요청을 제어할 것을 권장합니다.

Cache policy and origin request policy (recommended)

Legacy cache settings

캐시 정책
기존 캐시 정책을 선택하거나 새 캐시 정책을 생성합니다.

CachingOptimized Recommended for S3 ▾

Policy with caching enabled. Supports Gzip and Brotli compression.

[정책 생성](#) [정책 보기](#)

원본 요청 정책 - 선택 사항
기존 원본 요청 정책을 선택하거나 새 정책을 생성합니다.

원본 정책 선택 ▾

[정책 생성](#)

응답 헤더 정책 - 선택 사항
기존 응답 헤더 정책을 선택하거나 새 정책을 생성합니다.

SimpleCORS Allows all origins for simple CORS requests ▾

[정책 생성](#) [정책 보기](#)

▶ 추가 설정

10. 외부 서비스

1. KAKAO LOGIN

Web		삭제	수정
사이트 도메인	https://lilac-music.net http://lilac-music.net http://k8a203.p.ssafy.io http://localhost:8080 http://localhost:3000		

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

Redirect URI

삭제

수정

Redirect URI	https://lilac-music.net/login/oauth2/code/kakao http://localhost:8080/login/oauth2/code/kakao http://localhost:8082/login/oauth2/code/kakao http://localhost:8083/login/oauth2/code/kakao http://lilac-music.net/login/oauth2/code/kakao http://k8a203.p.ssafy.io/login/oauth2/code/kakao http://localhost:8080/api/oauth2/kakao http://localhost:8082/api/oauth2/code/kakao http://localhost:8083/api/oauth2/code/kakao https://lilac-music.net/api/oauth2/kakao
--------------	--

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

개인정보

항목 이름	ID	상태	
닉네임	profile.nickname	<input checked="" type="radio"/> 필수 동의	설정
프로필 사진	profile.image	<input checked="" type="radio"/> 필수 동의	설정
카카오계정(이메일)	account_email	<input checked="" type="radio"/> 필수 동의	설정

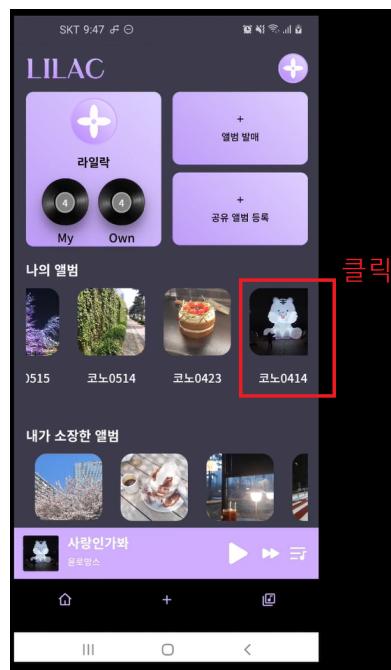
- 배포 사이트 도메인 : <https://lilac-music.net>
- 배포 redirect uri : <https://lilac-music.net/login/oauth2/code/kakao>
- client id와 client secret을 확인하여 카카오 디벨로퍼 인증
- 개인정보는 닉네임, 프로필사진, 이메일을 필수 동의로 설정

11. 시연 시나리오

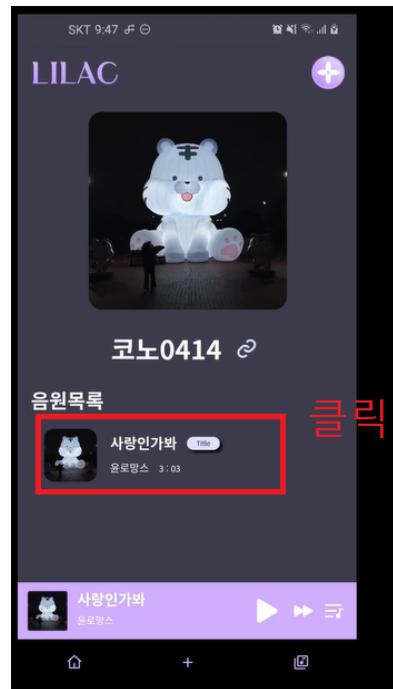
1. 메인 기능 시연



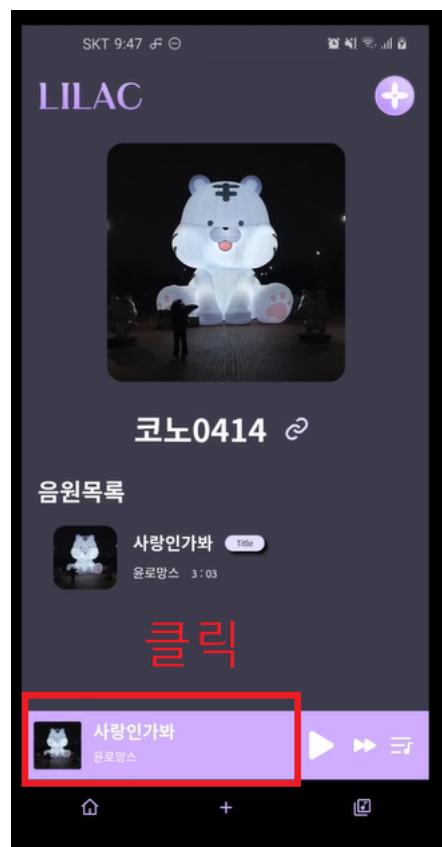
- 메인화면 소개



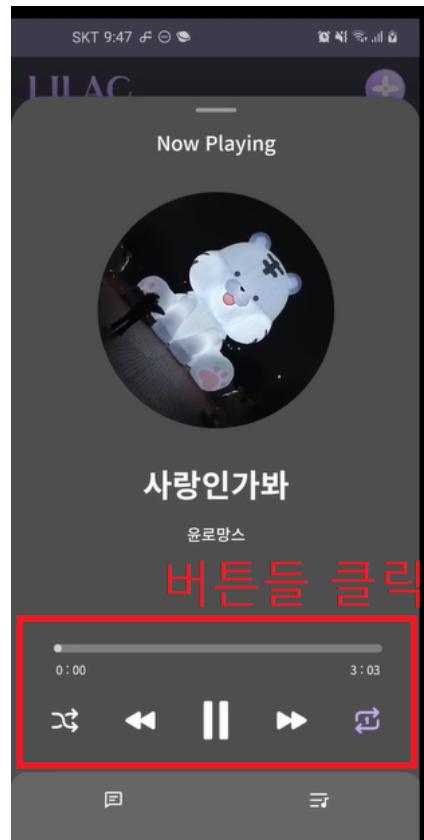
- 클릭 후 앨범 상세 화면으로 이동



- 클릭 후 재생 목록에 추가



- 클릭 후 플레이어로 이동



- 버튼들 클릭하며 기능 확인



- 백그라운드로 이동 후 백그라운드 재생 확인 다시 앱으로 복귀



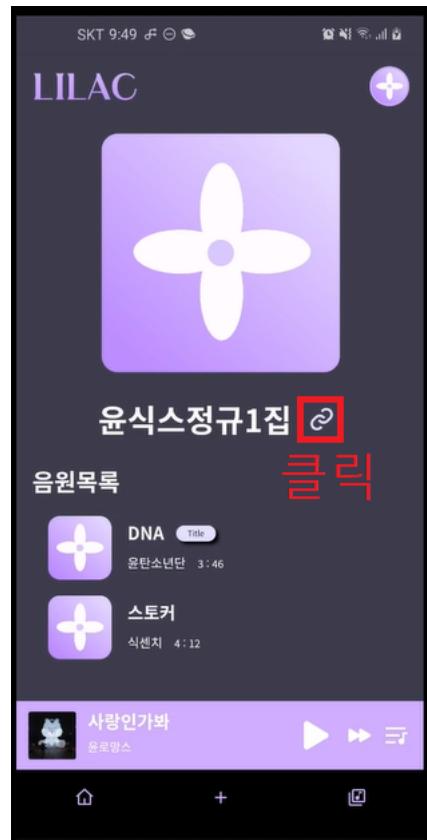
- 앨범 발매 클릭 후 앨범 발매 페이지 이동



- 위에서 부터 차례대로 앨범 커버, 앨범 제목, 음원 등록



- 앨범 등록 정보 작성 후 활성화 된 등록 버튼 클릭



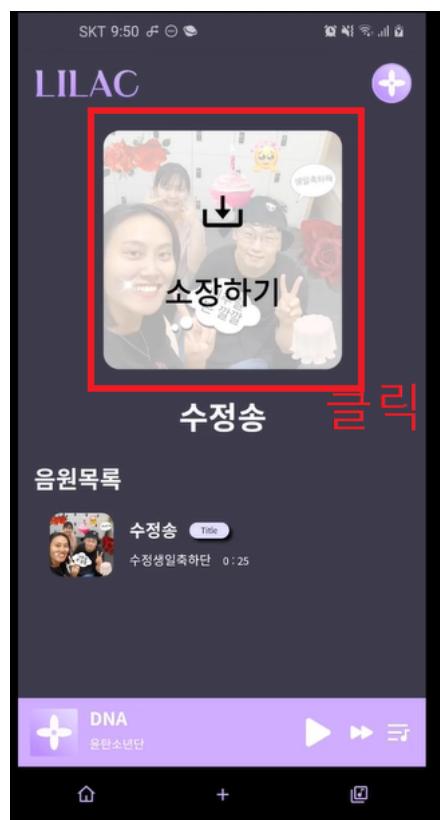
- 앨범 등록 후 화면 공유 링크 버튼 클릭 후 공유 링크 복사



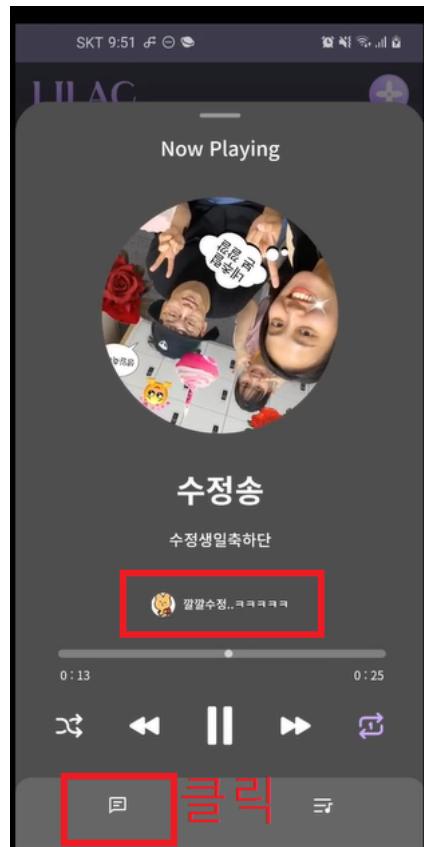
- 카카오톡으로 이동 후 공유 받을 앨범 링크 복사



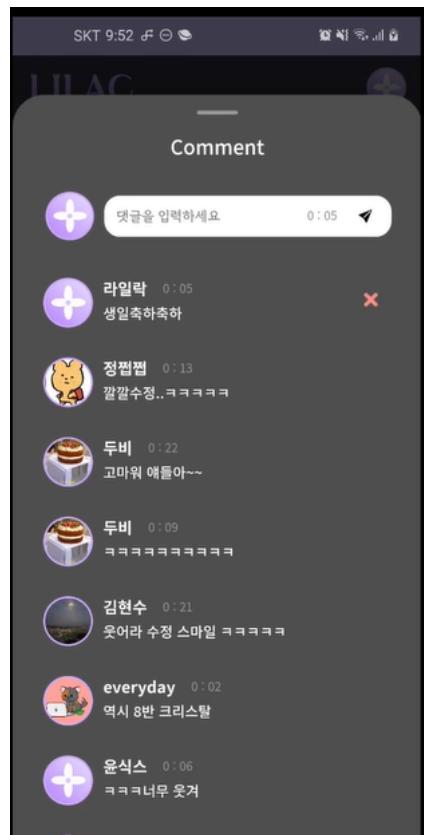
- 공유 앨범 등록 클릭 후 나오는 모달창에 복사한 앨범 공유 링크 입력 후 이동



- 소장하기 클릭 후 소장

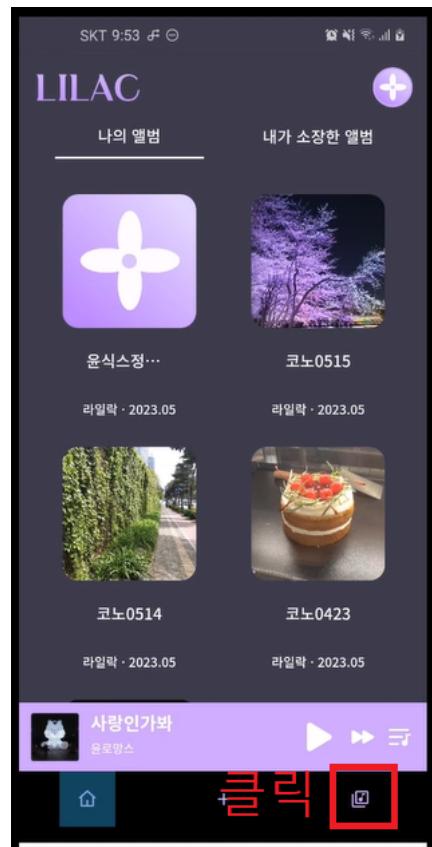


- 재생바 위 댓글 확인 후 아래 댓글 목록 클릭



- 댓글 입력후 다시 플레이어로 돌아와 댓글 정상적으로 출력되는지 확인

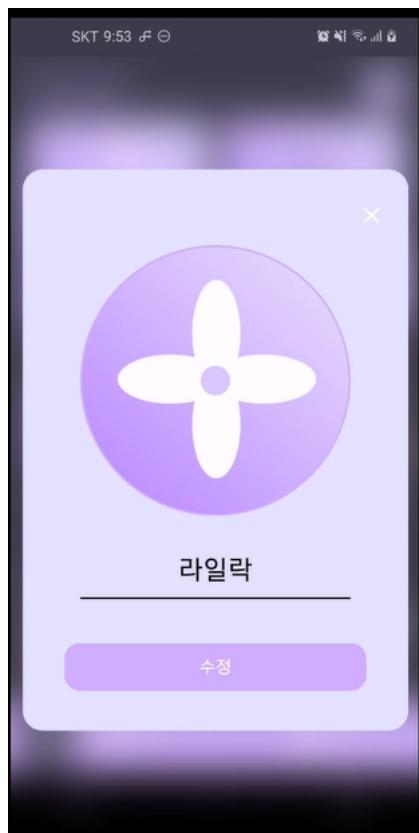
2. 기타 기능 시연



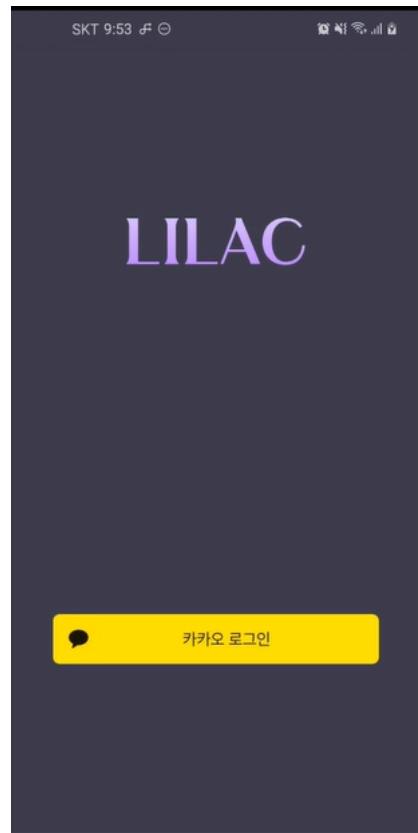
- 전체 소장앨범 확인 페이지



- 프로필 사진 클릭 후 정보수정, 로그아웃 확인



- 개인 프로필 수정 페이지



- 로그아웃 후 카카오 소셜 로그인 페이지