Song for you 배포 매뉴얼

본 문서는 안드로이드로 개발된 Run with me 를 사용하기 위한 가이드를 안드로이드와 백엔드 스프링 서버를 배포하는 과정에 대해 서술하고 있습니다.

# 안드로이드 배포

## AOS 개발 환경

**안드로이드**

- Android Studio Dolphin (2021.3.1)

- targetSDK 31

- minSDK 23

- Kotlin

# AOS 기술 스택

안드로이드

- Retrofit 2.9.0 (통신 라이브러리)

- OkHttp

- Ted Permission 3.3.0 (안드로이드 권한 라이브러리)

- Dagger-Hilt (의존성 주입 라이브러리)

- JetPack Paging3 (페이징 라이브러리)

- Coroutines Flow (비동기 데이터 처리 라이브러리)

- Glide 4.12.0 (이미지 로드 라이브러리)

- ViewModel-ktx 2.3.1

- Fragment-ktx 1.3.6

- Navigation 2.3.5 (화면 전환, 스택 관리 라이브러리)

- Lottie (안드로이드 애니메이션 라이브러리)

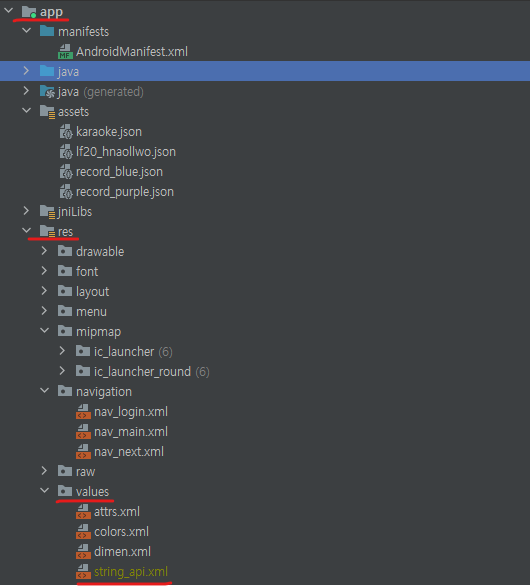
- Kakao Map (지도 라이브러리)

- OAuth (로그인 보안 라이브러리)

- Any Chart(태그 클라우드)

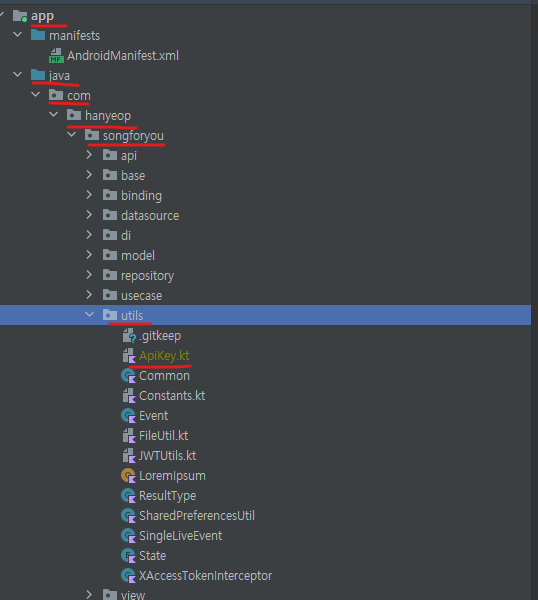
- Tag-sphere(태그 View)

# AOS 포팅 가이드

https://drive.google.com/file/d/1Tbft9nvWXnyfashF24NL6raDyHunpZRk/view?usp=sharing

App -> res -> values

string\_api.xml 추가



app -> java -> com -> hanyeop - > songforyou -> utils

ApiKey.kt 추가

# 백엔드 배포 과정 (젠킨스 이용 자동 배포)

## 개발환경 및 배포 환경

개발 환경

-Spring boot version : 2.7.1

-빌드 도구 : spring-boot-maven-plugin, 스프링 부트 버전에 해당하는 내장 메이븐을 사용한다.

-IDE : sts-3.9.14.RELEASE

배포 환경

배포 서버 : Ubuntu 20.04.4 LTS

Docker : 20.10.17

Maven : Apache Maven 3.6.3

Nginx : nginx/1.18.0 (ubuntu)

Certbot : certbot 0.40.0

## API 서버활성을 위한 배포과정

1. DataBase 서버를 먼저 배포하기 위한 준비를 한다.
2. /home/ubuntu/mysql-docker/mysql-compose/ 경로에 docker-compose.yml 파일을 만들어서 DB의 연결정보를 작성한다.

|  |
| --- |
| version: '3' # docker compose 버전  services:  local-db:  image: library/mysql:8.0.28  container\_name: mysql-compose # 컨테이너 이름  restart: always  ports:  - 13306:3306 # 로컬의 13306 포트를 컨테이너의 3306포트로 연결  environment:  MYSQL\_USER: d207sfy  MYSQL\_PASSWORD: d207!@#  MYSQL\_ROOT\_PASSWORD: d207!@#  TZ: Asia/Seoul  volumes:  - ./db/mysql/data:/var/lib/mysql  - ./db/mysql/init:/docker-entrypoint-initdb.d |

중요한 부분들이 있지만 특히 중요한 것이 volumes: 항목에서 ec2 환경의 경로 : docker compose 내부의 경로로 공유를 하도록 설정하는 것이다.

1. 위 파일을 이용해 docker를 빌드하고 아래 명령어로 권한 부여를 한다.

|  |
| --- |
| docker exec -it mysql-container bash  mysql -u root -p => 패스 워드를 입력 : 위 docker-compose.yml에 작성한 루트 비번사용  Grant All PRIVILEGES ON \*.\* TO ‘d207sfy @’%’ create database songforyou default character set utf8mb4 collate utf8mb4\_bin |

1. Git lab 의 https://lab.ssafy.com/s07-webmobile4-sub2/S07P12D101 위치로 이동하여 Clone한다. <Deploy> branch로 checkout 한다.
2. Docker compose의 정보가 담긴 docker-compose.yml파일은 미리 만들어둔다.

|  |
| --- |
| version: "3"  services:  spring:  user: root  container\_name: spring-compose  build:  context: spring-compose  dockerfile: Dockerfile  ports:  - 8080:8080  volumes:  - /var/lib/jenkins/workspace/songforyou\_spring/BE/gumid207/target:/deploy  extra\_hosts:  - "host.docker.internal:host-gateway"  environment:  - TZ=Asia/Seoul  redis:  image: redis  ports:  - 16379:6379 |

1. Docker의 실행순서를 담은 DockerFile 파일도 만들어둔다.

|  |
| --- |
| FROM openjdk:8-jdk  ENTRYPOINT java -jar -Dspring.profiles.active=dev /deploy/\*.jar  EXPOSE 8080 |

1. Jenkins의 Build Steps에서 Excute shell을 선택하여 배포 시 아래의 명령어가 실행될 수 있도록 한다.
   1. 이때, key의 안전을 위해 application.yml은 git에 업로드하지 않고 따로 보관해서 docker 내부로 복사하는 형태를 취한다.

|  |
| --- |
| cd /var/lib/jenkins/workspace/songforyou\_spring/BE/gumid207  sudo mvn clean  sudo cp /var/lib/jenkins/workspace/songforyou\_spring\_settings/application.yml /var/lib/jenkins/workspace/songforyou\_spring/BE/gumid207/src/main/resources/application.yml  sudo mvn -Dmaventest.skip=true  sudo chmod 777 target/\*.jar  cd /home/ubuntu/spring-docker  sudo docker-compose up --force-recreate --build -d  sudo docker image prune -f |

1. Django 서버도 비슷한 과정으로 배포를 진행한다. 다만, Django의 경우 SpringBoot와는 다른 방식을 시도한다는 느낌으로, DockerFile을 git으로부터 받아서 배포하는 형태로 할 것이다. 이는 개발과정에서 패키지의 import에 따른 install 명령어의 변화가 많을 수 있는 python의 특성을 고려한 선택이다
2. /var/lib/jenkins/workspace/songforyou\_django/DATA/Data-Django/ 에는 gitlab으로부터 clone된 Django 관련 파일들이 있다. 이 중 Dockerfile을 활용하기 위해 jenkins에서 다음과 같은 명령러를 실행하도록 설정한다.

|  |
| --- |
| sudo docker image prune -a --force  mkdir -p /var/lib/jenkins/image\_django  cd /var/lib/jenkins/workspace/songforyou\_django/DATA/Data-Django/  sudo docker build -t django .  sudo docker save django > /var/lib/jenkins/image\_django/django.tar  sudo ls /var/lib/jenkins/image\_django  exit |

살펴보면, 기존 장고의 이미지 삭제

tar파일 경로 생성,

Dockerfile이 있는 곳으로 디렉토리 전환,

생성한 경로에 tar파일 저장

이 순서로 빌드를 진행시킨다.

1. 빌드 후 재배포를 위해 다음과 같은 명령을 추가한다.

|  |
| --- |
| sudo docker load < /var/lib/jenkins/image\_django/django.tar  if (sudo docker ps -a | grep "django"); then sudo docker stop django; fi  sudo docker rm $(docker ps -a -f status=exited -q)  sudo docker run -it -d -p 8000:8000 --name django django |

빌드한 tar파일로 기존 도커를 멈추고 새로운 도커를 실행시키는 것이다.

1. 이제 nginx Web Server를 설치하자. Nginx 는 도커를 사용하지 않고 ubuntu 서버에 직접 설치하는 방식을 사용한다.
2. 아래의 명령어를 사용하여 nginx를 설치하자.

|  |
| --- |
| sudo apt-get update  sudo apt-get install nginx #이렇게 할 경우 nginx 공식에서 제공하는 최신 버전은 아니다.  nginx -v  systemctl status nginx 를 통해 잘 동작하는 지 확인 |

1. 설치가 완료 되었다면 스프링 부트 역프록시 설정을 하기위해 아래의 명령어를 입력한다.

|  |
| --- |
| cd /etc/nginx/sites-enabled  vi default |

1. 사용할Server 블록에 아래와 같이 삽입한다.

|  |
| --- |
| location /api {  proxy\_pass http://localhost:8080/api;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header Host $http\_host;  } |

1. 이제 http://도메인/api/\*\* 로 요청을 보내면 잘 동작할 것이다. 하지만 우리는 https 를 적용할 것이다. 따라서 우선 nginX를 종료시킨다.  
   => systemctl stop nginx  
   => sudo netstat -ntlp | grep :80 # 80 포트는 비워져있어야한다.
2. 무료로 SSL 적용을 위해 certbot을 다운 받아 standalone방식으로 인증서를 발급받고 적용한다.

|  |
| --- |
| sudo apt update  sudo apt-get install letsencrypt -y  certbot certonly –standalone -d <도메인(사이트명,ec2 도메인)> |

1. 위 과정을 거치면 /etc/letsencrypt/live/ 밑에 도메인 명을 이름으로하는 디렉터리가 생성되고 그 하위에는 여러 pem파일들이 생긴다.
2. 이제 nginx 설정을하는 ~/sites-enabled/default 파일을 수정한다.  
   13번 과정에서 설정했던 server블록에 아래와 같이 적어준다. 노란색 영역이 nginx에 ssl인증서를 적용하기 위한 핵심 부분이다. 회색 부분은 뒤에 설명할 react 배포를 위한 설정이다. 추가로 갈색 부분은 http 프로토콜을 사용하여 80포트로 접근할 경우 https를 사용하는 443포트로 redircect하기 위한 설정이다.

|  |
| --- |
| server {  listen 443 ssl default\_server;  listen [::]:443 ssl default\_server;  client\_max\_body\_size 100M;  ssl\_certificate /etc/letsencrypt/live/j7d207.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/j7d207.p.ssafy.io/privkey.pem;  server\_name i7d101.p.ssafy.io;  location /api {  # First attempt to serve request as file, then  # as directory, then fall back to displaying a 404.  proxy\_pass http://localhost:8080/api;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header Host $http\_host;  }  location /data {  proxy\_pass http://localhost:8000;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header Host $http\_host;  }  }  server {  return 301 https://$host$request\_uri;  # managed by Certbot  listen 80;  server\_name regacy\_port;  return 404; # managed by Certbot  } |

설정이 끝났으니 nginx를 재시작한다. 이제 api서버 설정은 끝났다.   
=> systemctl restart nginx

# 외부 서비스에 대한 설명

## Google Oauth, Naver Oauth , Kakao Oauth

application.yml 파일에는 해당 내용이 있어야한다.

spring:

profiles:

include:

- oauth

application-oauth.yml 에는 해당 내용이 있어야한다.

spring:

security:

oauth2:

client:

registration:

google:

client-id: <구글 GCP Oauth설정 후 받은 클라이언트 ID>

client-secret: <시크릿 키>

scope: profile, email

naver:

client-id: <네이버 Oauth 이용 신청 후 받은 ID>

client-secret: <시크릿 키>

redirect-uri: <사용안한다.>

authorization-grant-type: authorization\_code

scope: name, email, profile\_image

client-name: Naver

kakao:

client-id: <카카오 Oauth 서비스 신청 후 발급받은 아이디>

client-secret: <시크릿 키>

redirect-uri: <사용안한다.>

authorization-grant-type: authorization\_code

client-authentication-method: POST

client-name: Kakao

scope:

- profile\_nickname

- profile\_image

- account\_email

provider:

naver:

authorization\_uri: https://nid.naver.com/oauth2.0/authorize

token\_uri: https://nid.naver.com/oauth2.0/token

user-info-uri: https://openapi.naver.com/v1/nid/me

user\_name\_attribute: response

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-attribute: id

사실 위에 명시된 모든 값들이 이용되기 위해서는 클라이언트에서 권한서버로 로그인 성공 후 프로바이더가 발급해주는 id\_token 혹은 authorized\_token(프로바이더 마다 이름이 다르다.) 을 통해 리소스 서버(스프링 부트)에서 프로바이더로 accessToken을 얻고 유저 프로필에 대한 정보를 받는 과정이여야 한다. 하지만 우리는 구글의 경우 id\_token을 통해 구글 라이브러리를 사용하여 하지만, 네이버와 카카오의 경우 Android가 client 임으로 accessToken을 얻는 과정까지 수행 후 리소스 서버(스프링 부트) 클라이언트가 준 accessToken을 통해 유저 프로필을 얻고 있다. 따라서 위 속성들은 부분적으로 불필요한 경우가 있으나 확장성과 안정성을 위해 모두 입력해주기를 바란다.

## Firebase

Firebase홈페이지로 이동 후, 스프링 부트에 사용될 json(해당 파일은 클라우드 메시지 탭-> 웹 구성 -> 웹 푸시 인증서에서 발급받는다) 파일을 받은 후 src/main/resources/firebase 밑으로 이동, 붙여넣기 하면 된다.

## 4.3. Amazon S3

[[SpringBoot] SpringBoot를 이용한 AWS S3에 여러 파일 업로드 및 삭제 구현하기 — ROOPRETELCHAM (tistory.com)](https://earth-95.tistory.com/117)