마이풋트립 배포 매뉴얼

본 문서는 안드로이드로 개발된 마이풋트립을 사용하기 위한 가이드를 프론트엔드(안드로이드)와 백엔드 스프링 서버를 배포하는 과정에 대해 서술하고 있습니다.

# **1. 안드로이드 배포**

## 1.1. 안드로이드 개발 환경

- Android Studio Dolphin (2021.3.1 Patch1)

- targetSDK 32

- minSDK 21

- Kotlin

## 1.2. AOS 기술 스택

안드로이드

- Retrofit 2.9.0 (통신 라이브러리)

- OkHttp 3.11.0

- Ted Permission 3.3.0 (안드로이드 권한 라이브러리)

- JetPack Paging3 (페이징 라이브러리)

- Coroutines Flow (비동기 데이터 처리 라이브러리)

- Glide 4.13.0 (이미지 로드 라이브러리)

- ViewModel-ktx 2.4.1

- Fragment-ktx 1.4.1

- Navigation 2.4.2 (화면 전환, 스택 관리 라이브러리)

- Lottie (안드로이드 애니메이션 라이브러리)

- Naver Map 16.0.0 (네이버 지도 라이브러리)

- Naver Login(네이버 로그인 라이브러리)

- Kakao Login(카카오 로그인 라이브러리)

- FCM (파이어베이스 푸시 알림 라이브러리)

- FirebaseStorage (파이어베이스 스토리지 라이브러리)

- Room (내부 데이터 베이스 라이브러리)

- notification-badge(이미지 뱃지 라이브러리)

- swipeRefreshLayout(새로고침 라이브러리)

- whynotimagecarousel(이미지 슬라이더 라이브러리)

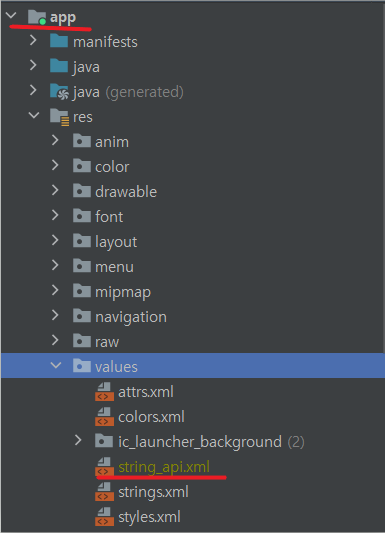
- Picasso(이미지 뷰 라이브러리)

- Google gms play service location (구글 현재 위치 라이브러리)

## 1.3. 안드로이드 포팅 가이드

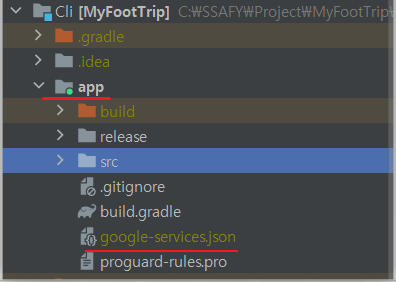
<https://drive.google.com/file/d/1Xr62xenzZSBr59Iv6rkttvdUadl8Bnsj/view?usp=sharing>

### 1.3.1. API 키



App -> res -> values -> string.xml 추가

### 1.3.2. google-service.json



Project로 변경 -> app -> google-service.json 추가

# **2. api 서버 배포 과정**

## 2.1. 개발환경

- Python Version : 3.11.1 (local) , 3.8.10(EC2 ubuntu)

- Django version : 3.2.13

-IDE : vs code latest(최신) 버전

-빌드 도구 : pip 22.3.1 과 가상환경

-서버 관련 주요 라이브러리 목록과 버전:

dj-rest-auth==2.2.5

django-allauth==0.50.0

drf-spectacular==0.24.2

google-api-core==2.11.0

google-api-python-client==2.75.0

google-auth==2.16.0

google-cloud-core==2.3.2

google-cloud-firestore==2.9.1

google-cloud-storage==2.7.0

gunicorn==20.1.0

mysqlclient==2.1.1

pilkit==2.0

Pillow==9.4.0

PyJWT==2.6.0

django-imagekit==4.1.0

djangorestframework==3.14.0

djangorestframework-simplejwt==5.2.2

## 2.2. 배포 환경

배포 서버 : Ubuntu 20.04.4 LTS

Docker : 20.10.12

Docker-compose : 1.25.0

Docker Image 목록

* Docker image 들을 docker-compose 를 활용해 일괄 관리 하고 있음.

Ubuntu : 22.04 // 장고 이미지를 위한 베이스가 됨

Nignx: latest

Mysql: latest

Cerbot : latest

Jenkins : lts-jdk11

## 2.3. 서버 배포를 위한 과정

1. 서버 환경에서 <https://lab.ssafy.com/luminaries1/mobile_pjt.git> 주소를 clone 한다.

2. mobile\_pjt->pjtback 디렉토리로 이동한다.

3. 해당 디렉토리에 소스코드 및, 두가지 쉘 스크립트가 있는데, auto\_setting.sh 실행 후 1번 입력 => cert\_bot 이 up이 안되어 있을 경우 init-letsencrypt.sh 쉘스크립트까지 이어서 실행한다.

4. 해당 쉘 스크립트가 실행되고 git 인증만 제대로 입력한다면 apt-get install 등으로 Ubuntu 환경 구축, docker-compose.yml 을 통한 이미지 종속성 및 빌드, dockerfile로 메인 web 서버 환경 설정까지 완료 되기 때문에 서버 배포는 끝이다. 모든 인풋 요청에 대해서는 y 혹은 yes로 답하면 된다.

5. 빌드 파일들 종속성 및 구조

노란색 글씨의 명령어가 다음 파일 실행의 진입점들에 해당한다.

echo " 1. 최초"

echo " 2. 소스코드 수정시 배포"

rootdir="mobile\_pjt"

appdir="mobile\_pjt/pjtback"

read value

if [ ${value} -eq 1 ]

then

    sudo apt-get update

    sudo apt-get -y dist-upgrade

    sudo apt-get install docker

    sudo apt-get -y install docker-compose

    sudo docker-compose up --build

    chmod +x init-letsencrypt.sh

    ./init-letsencrypt.sh

fi

if [ ${value} -eq 2 ]

then

    cd ~/${rootdir}/

    git restore .

    git pull origin master

    cd ~/${appdir}/

    python manage.py makemigrations

    python manage.py migrate

    sudo docker-compose stop

    sudo docker-compose rm web

    sudo docker-compose up --build

    echo "web 로그를 확인해주세요"

fi

**auto\_setting.sh**

version: "2.4"                                                # docker-compose 버전이다

services:

    nginx:

        image: nginx

        container\_name: nginx

        ports:                                              # 포트 개방 "hostPort:dockerPort"으로 포워딩한다

            - "80:80"

            - "443:443"

        restart: always

        volumes:                                            # 볼륨 마운트. 호스트의 파일을 사용하려면 필요하다

            - .:/pjtback

            - ./config/nginx/nginx.conf:/etc/nginx/conf.d/default.conf # nginx 세팅 마운트

            - ./data/certbot/conf:/etc/letsencrypt              #cerbot 관련

            - ./data/certbot/www:/var/www/certbot

        command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload; done & nginx -g \"daemon off;\"'"

        depends\_on:

            - web

    certbot: # certbot 서비스 추가

        image: certbot/certbot

        container\_name: certbot\_service

        volumes:

            - ./data/certbot/conf:/etc/letsencrypt

            - ./data/certbot/www:/var/www/certbot

        entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"

    web:

        build: .                           # 메인으로 빌드될 Dockerfile을 적는다. 지금은 현재 폴더에 있다고 알린다

        container\_name: web

        restart: always

        command: >

            bash -c "python3 manage.py collectstatic --no-input &&

            export DJANGO\_SETTINGS\_MODULE=config.settings.prod &&

            python3 manage.py makemigrations &&

            python3 manage.py migrate &&

            gunicorn pjtback.wsgi:application --bind 0.0.0.0:8000"

        depends\_on:

          db:

            condition: service\_healthy

        ports:

            - "8000:8000"

        volumes:

            - .:/pjtback

        environment:

            - TZ=Asia/Seoul

    db:

        image: mysql:latest

        container\_name: db

        ports:

            - "3307:3306"

        environment:

            MYSQL\_DATABASE: django\_mobile

            MYSQL\_PASSWORD: sangjun1324

            MYSQL\_ROOT\_PASSWORD: sangjun1324

        healthcheck:

            test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost"]

            interval: "10s"

            timeout: "3s"

            start\_period: "5s"

            retries: 3

**docker-compose.yml**

image 종속성은 DB -> WEB -> NGINX 순서로 빌드 및 동작을 확인하는 구조로 되어 있다.

### 우분투 초기 설정 ###

# ubuntu 이미지 22.04 버전을 베이스 이미지로 한다

FROM ubuntu:22.04

# apt 업그레이트 및 업데이트

RUN apt-get -y update && apt-get -y dist-upgrade

# debconf : apt-util 어쩌구 빨간 error 방지

RUN apt-get install -y --no-install-recommends apt-utils

RUN apt-get install -y python3-pip python3-dev

# Django 기본 언어를 한국어로 설정하면 파이썬 기본 인코딩과 충돌되어 한글 출력, 입력시에 에러가 난다.

# 따라서 파이썬 기본 인코딩을 한국어를 사용할 수 있는 utf-8으로 설정한다.

ENV PYTHONIOENCODING=utf-8

# pip 를 root user 로 전부 해결 하는 것에 대한 warnign ignore 옵션

# 근본적으로는 user 만들어서 image 처리 하는게 맞다.. 나중에 고려할것

ENV PIP\_ROOT\_USER\_ACTION=ignore

# frontend 어쩌구 하는 error 해결 , input 필요한 install 부분 때매 생겨난다 예시 : tzdata

ENV DEBIAN\_FRONTEND=noninteractive

# pip setuptools 업그레이드

# mysql 붙이기 위한 작업

RUN pip3 install --upgrade pip

RUN pip3 install --upgrade setuptools

RUN apt-get install -y python3-dev default-libmysqlclient-dev build-essential

### 실행 환경 구축 ###

# 컨테이너 내부에 config(설정파일)가 들어갈 폴더 생성

RUN mkdir /config

ADD /config/requirements.txt /config/

# requirements.txt에 있는 파이썬 패키지 설치

RUN pip3 install -r /config/requirements.txt

RUN ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime

RUN apt-get install tzdata

### 작업 디렉토리 ###

# Django 소스코드가 들어갈 폴더 생성

RUN mkdir /pjtback;

WORKDIR /pjtback

**Dockerfile**

#!/bin/bash

if [ $(id -u) -ne 0 ]; then exec sudo bash "$0" "$@"; exit; fi

if ! [ -x "$(command -v docker-compose)" ]; then

  echo 'Error: docker-compose is not installed.' >&2

  exit 1

fi

domains=(i8d103.p.ssafy.io)

rsa\_key\_size=4096

data\_path="./data/certbot"

email="luminaries1@naver.com" # Adding a valid address is strongly recommended

staging=0 # Set to 1 if you're testing your setup to avoid hitting request limits

if [ -d "$data\_path" ]; then

  read -p "Existing data found for $domains. Continue and replace existing certificate? (y/N) " decision

  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then

    exit

  fi

fi

if [ ! -e "$data\_path/conf/options-ssl-nginx.conf" ] || [ ! -e "$data\_path/conf/ssl-dhparams.pem" ]; then

  echo "### Downloading recommended TLS parameters ..."

  mkdir -p "$data\_path/conf"

  curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certbot\_nginx/\_internal/tls\_configs/options-ssl-nginx.conf > "$data\_path/conf/options-ssl-nginx.conf"

  curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot/certbot/ssl-dhparams.pem > "$data\_path/conf/ssl-dhparams.pem"

  echo

fi

echo "### Creating dummy certificate for $domains ..."

path="/etc/letsencrypt/live/$domains"

mkdir -p "$data\_path/conf/live/$domains"

docker-compose run --rm --entrypoint "\

  openssl req -x509 -nodes -newkey rsa:$rsa\_key\_size -days 1\

    -keyout '$path/privkey.pem' \

    -out '$path/fullchain.pem' \

    -subj '/CN=localhost'" certbot

echo

echo "### Starting nginx ..."

docker-compose up --force-recreate -d nginx

echo

echo "### Deleting dummy certificate for $domains ..."

docker-compose run --rm --entrypoint "\

  rm -Rf /etc/letsencrypt/live/$domains && \

  rm -Rf /etc/letsencrypt/archive/$domains && \

  rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot

echo

echo "### Requesting Let's Encrypt certificate for $domains ..."

#Join $domains to -d args

domain\_args=""

for domain in "${domains[@]}"; do

  domain\_args="$domain\_args -d $domain"

done

# Select appropriate email arg

case "$email" in

  "") email\_arg="--register-unsafely-without-email" ;;

  \*) email\_arg="--email $email" ;;

esac

# Enable staging mode if needed

if [ $staging != "0" ]; then staging\_arg="--staging"; fi

docker-compose run --rm --entrypoint "\

  certbot certonly --webroot -w /var/www/certbot \

    $staging\_arg \

    $email\_arg \

    $domain\_args \

    --rsa-key-size $rsa\_key\_size \

    --agree-tos \

    --force-renewal" certbot

echo

echo "### Reloading nginx ..."

docker-compose exec nginx nginx -s reload

**init-letsencrypt.sh**

https url 사용을 위해서 certbot 이미지를 빌드, 런하고 해당 라이브러리를 통해 발급받은 인증서를 이미지 안쪽에 마운트 하는 과정

6. 젠킨스 이미지를 통해 git push 로 서버 배포를 할 수 있으나 ( gitlab push 가 trigger 가 되어서 위처럼 쉘 스크립트를 실행시키는 구조로 간단하게 짜여져 있다) 계속해서 앱 배포 검토중인 과정이고, 마음대로 본 서버가 업데이트 되면 안되어서 현재는 젠킨스 이미지를 중지 시켜놓은 상태이다.

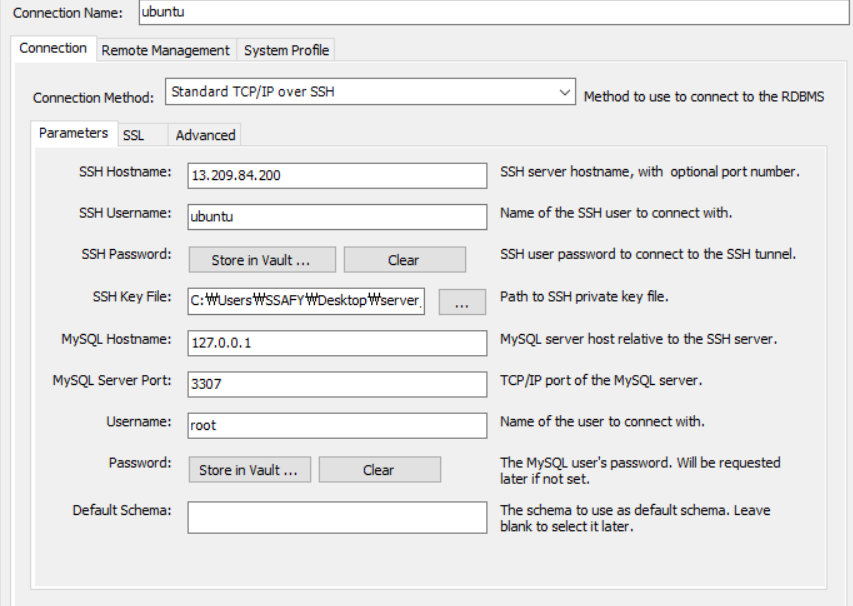
7. 관리자 페이지는 장고가 기본적으로 제공해주고 있는 admin url 을 활용하고 있다

**url: https://i8d103.p.ssafy.io/admin/**



## 2.4. DB 접속을 위한 세팅 정보

* Mysql 최신버전 이미지
* Workbench를 통한 connection 정보



* Django 와 DB 의 연결 정보는 config/settings dir의 base.py, prod.py 안에 저장되어있다.

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.mysql',

        'NAME': 'django\_mobile',

        'USER': 'root',

        'PASSWORD': 'sangjun1324',

        'HOST': 'db',

        'PORT': '3306',

        "OPTIONS": {"charset": "utf8mb4"}

    }

}

* 계정정보는 ID : root PW: sangjun1324 로 설정되도록 docker file들을 구성해놓음
* 기본적으로 사용하는 Database 는 django\_mobile

## 2.5. 배포시 특이사항

- gitignore에 static 과 media , migrations 폴더를 추가 해놓아 mount 되거나 장고 migration, migrate 시 테이블 구조 반영에 있어서 충돌 나지 않도록 설정하였음

- 이모지를 허용하여야 하기 때문에 DB 의 utf 설정을 utf8mb4 로 설정하였다.

- gunicorn 은 Django 이미지 내에서 명령어를 통해 실행된다.

- Django setting 파일을 그냥 settings.py , base.py , prod.py 로 분리 해놓았다.

- config 에는 fcm 을 위한 serviceaccount.json 과 pip install 을 위한 requirements.txt 그리고 nginx 설정 및 https proxypass 를 위한 nginx.conf 파일들이 들어있다.

- 젠킨스는 현재는 이미지 동작을 멈춰 놓은 상태이다.

- EC2 머신 내에서 배포 할 앱을 위한 본 서버 8000번 포트를 제외하고 테스트를 위한 이미지화 되지 않은 백그라운드에서 돌아가는 7777, 7778 서버 등이 돌아가는 중이다.