# 포팅 매뉴얼

## 📕 Project Skill Stack Version

| Skill | Version |
| --- | --- |
| React | 18.3.1 |
| Axios | 1.7.7 |
| Vite | 5.4.5 |
| TypeScript | 5.4.2 |
| Java | 17 |
| SpringBoot | 3.3.3 |
| Gradle | 8.10.1 |
| Python | 3.11.10 |
| Django | 4.2.16 |
| MySQL | 9.0.1 for Linux on x86_64 (MySQL Community Server - GPL) |
| Redis | 7.4.0 |
| Nginx | 1.18.0 (Ubuntu) |
| Jenkins | 2.476 |
| Docker | 27.2.1 |

## 📘 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost
- 디자인 : Figma
- CI/CD : Jenkins, Docker, DockerHub

# 📙 개발 도구

- Visual Studio Code : 1.92.2
- IntelliJ : IntelliJ IDEA 2024.1.4 (Ultimate Edition)

# 📗 EC2 포트 번호

Backend : 8080, 8081, 9090

Frontend : 80

MySQL : 3306

Django : 8000

Redis : 6379

Jenkins : 9005

# 📖 CI/CD 구축

백엔드 서버부터 구축

## Swap 메모리 설정

여러 빌드 동시처리시 물리적 메모리가 가득 찼을때 추가 작업을 위한 swap 메모리 설정

```
스왑 메모리 설정
// swap 파일을 생성해준다.
// (메모리 상태 확인 시 swap이 있었지만 디렉토리 파일은 만들어줘야한다.)
sudo mkdir /var/spool/swap
sudo touch /var/spool/swap/swapfile
sudo dd if=/dev/zero of=/var/spool/swap/swapfile count=409600

// swap 파일을 설정한다.
sudo chmod 600 /var/spool/swap/swapfile
sudo mkswap /var/spool/swap/swapfile
sudo swapon /var/spool/swap/swapfile
```

```
// swap 파일을 등록한다.
sudo echo '/var/spool/swap/swapfile none swap defaults 0 0' |

// 메모리 상태 확인
free -h
```

## JDK 설치

17로 진행

```
# 업데이트
sudo apt update

# 업그레이드
sudo apt upgrade

# 특정 버전 목록 조회
sudo apt list openjdk-17

# 설치
sudo apt install openjdk-17-jdk

# 설치 확인
java --version
```

## Docker 설치

```
# 의존성 설치
sudo apt update
sudo apt install ca-certificates curl gnupg lsb-release

# 레포지토리
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg
```

```
# 레포지토리 추가
echo "deb [arch=$(dpkg --print-architecture) \
signed-by=/etc/apt/keyrings/docker.gpg] https://download.dock
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | su

# 도커 설치하기
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker
```

## Jenkins 설치

Docker outside of Docker (DooD)방식으로 진행

젠킨스에서 도커 기반의 빌드,테스트 환경을 직접 관리

```
# 도커 소켓 마운트 하기 (젠킨스 컨테이너에서 도커 명령어 실행되도록 하기)
docker run -itd --name jenkins -p 9005:8080 -v /var/run/docke

# 도커 명령어가 젠킨스에서 실행이 안되거나 권한 오류가 나면 아래 명령어 실
sudo chmod 666 /var/run/docker.sock

# 젠킨스 컨테이너 비밀번호 확인 명령어
docker exec jenkins cat /var/jenkins_home/secrets/initialAdmi

# 젠킨스 컨테이너로 접속해서 도커 명령어 실행 여부 확인 명령어
docker exec -it <container_name_or_id> /bin/bash
docker exec -it jenkins /bin/bash

# 젠킨스 컨테이너에 접속해서 Docker 명령어 되는지 확인
docker
```

접속 후 테스트

## Nginx 설치

```
sudo apt update
sudo apt upgrade
sudo apt install nginx
sudo service nginx start
sudo service nginx status
```

## https 설정 (SSL)

무료 Let’s Encrypt

```
#Encrypt 설치
sudo apt-get install letsencrypt

#Certbot 설치
sudo apt-get install certbot python3-certbot-nginx

#Certbot 동작 (nginx 중지하고 해야함)
sudo certbot --nginx
1번 방법 sudo certbot --nginx -d [도메인 혹은 ip 주소]
2번 방법 sudo letsencrypt certonly --standalone -d [도메인 혹은 i
# 옵션 1번 선택
# 강제 리다이렉트 설정 부분에서 안한다고 함(1번 선택) http 와 https 같이

nginx 설정 적용
sudo service nginx restart
sudo systemctl reload nginx
```

## Jenkins, gitLab webhook 설정

깃랩 토큰 발급 → 젠킨스 플러그인 등록(gitlab) → 젠킨스에 API token Credentials 등록 → 연결확인

## Jenkins pipline 생성

클론을 하기 위한 기본 코드 부터 작성

```
pipeline {
    agent any
    stages {
        stage('Git Clone'){
            steps {
                git branch: 'BE', credentialsId: 'GitLab_Logi
            }
            post {
                failure {
                  echo 'Repository clone failure !'
                }
                success {
                  echo 'Repository clone success !'
                }
            }
        }
    }
}
```

## 깃랩 웹훅 등록

URL, Secret Token, Trigger 작성

## Docker Hub Setting

로그인 후 컨테이너 생성

## Docker file 작성

```
# open jdk 21 버전의 환경을 구성
FROM openjdk:17
```

```
# tzdata 패키지 설치 및 타임존 설정
RUN ln -snf /usr/share/zoneinfo/Asia/Seoul /etc/localtime &&

# build가 되는 시점에 JAR_FILE이라는 변수 명에 build/libs/*.jar 선언
# build/libs - gradle로 빌드했을 때 jar 파일이 생성되는 경로
ARG JAR_FILE=build/libs/ourClass-0.0.1-SNAPSHOT.jar

# JAR_FILE을 agaproject.jar로 복사
COPY ${JAR_FILE} ourClass.jar

# 운영 및 개발에서 사용되는 환경 설정을 분리
# -Duser.timezone=Asia/Seoul JVM 옵션을 사용하여 애플리케이션 수준에
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

## Jenkins Credental Setting

- JASYPT → Secret text
- Docker Hub → Usernaem with password
- EC2 Server IP → Secret text

## SSH 접속설정

plugin 추가(SSH Agent Plugin)

Jenkins Credentials - .pem키 복사붙여넣기

## Nginx 설정 변경

무중단 배포 (Blue-Green) 로 진행

무중단 배포 경로를 잡기 위한 service-url.inc, Deploy File 따로 작성

Frontend는 기본 포트

Backend는 /api 밑으로 들어오면 8080,8081로 연결(socket통신 포함)

### nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

worker_rlimit_nofile 2048;  # 또는 필요한 만큼 더 높은 값

events {
        worker_connections 2048;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Droppi
        ssl_prefer_server_ciphers on;
```

```
        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        gzip_vary on;
        gzip_proxied any;
        gzip_comp_level 6;
        gzip_buffers 16 8k;
        gzip_http_version 1.1;
        gzip_types text/plain text/css application/json appli

        ##
        # Virtual Host Configs
        ##


        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}


#mail {
#       # See sample authentication script at:
#       # http://wiki.nginx.org/ImapAuthenticateWithApachePhps
#
#       # auth_http localhost/auth.php;
#       # pop3_capabilities "TOP" "USER";
#       # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
```

```
#       server {
#               listen     localhost:110;
#               protocol   pop3;
#               proxy      on;
#       }

#
#       server {
#               listen     localhost:143;
#               protocol   imap;
#               proxy      on;
#       }
#}
```

## nginx/sites-enabled/default

```
server {
        listen 80;
        listen [::]:80 ;
        server_name songpicker.kro.kr;

        location / {
             return 301 https://$host$request_uri;
        }
 }


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

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

        ssl_certificate /etc/letsencrypt/live/songpicker.kro.
        ssl_certificate_key /etc/letsencrypt/live/songpicker.
```

```
        include /etc/letsencrypt/options-ssl-nginx.conf; # ma
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # mana

        server_name songpicker.kro.kr;


        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        #include /etc/nginx/conf.d/service-url.inc;


        location / {

                root /home/ubuntu/Front/dist;
                index index.html;
                try_files $uri $uri/ /index.html;

        }


        location /karaoke/ {
                alias /home/ubuntu/karaoke_front/dist/;
                index index.html;
                try_files $uri $uri/ /karaoke/index.html;

                #sub_filter_types text/xml text/css text/java
                #sub_filter '/static/' '/karaoke/static/';
                #sub_filter_once off;
        }


        location /karaoke/assets {
                alias /home/ubuntu/karaoke_front/dist/assets;
        }


        location /api/ {
```

```
                proxy_pass http://localhost:8081/;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_read_timeout 20m;

                add_header 'Access-Control-Allow-Origin' '*';
                #add_header 'Access-Control-Allow-Methods' 'G
                add_header 'Access-Control-Allow-Headers' 'Or
                add_header 'Access-Control-Allow-Credentials'

                if ($request_method = 'OPTIONS') {
                    return 204;
                }

        }


        location /api/karaoke/ {
                proxy_pass http://127.0.0.1:9090/;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_read_timeout 20m;

                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GE
                add_header 'Access-Control-Allow-Headers' 'Or
                add_header 'Access-Control-Allow-Credentials'
```

```
                if ($request_method = 'OPTIONS') {
                    return 204;
                }
        }

        location /api/data/ {
                proxy_pass http://127.0.0.1:8000/;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_read_timeout 20m;

                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GE
                add_header 'Access-Control-Allow-Headers' 'Or
                add_header 'Access-Control-Allow-Credentials'

                if ($request_method = 'OPTIONS') {
                    return 204;
                }


        }
}
```

### service_url.inc

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

## Jenkins pipline BE 작성

```
pipeline {
    agent any

    tools {
        jdk ("jdk17")
    }

    stages {
        stage('Git Clone'){
            steps {
                git branch: 'develop', credentialsId: 'GitLab
            }
            post {
                failure {
                  echo 'Repository clone failure !'
                }
                success {
                  echo 'Repository clone success !'
                }
            }
        }
        stage('Clean') {
            steps {
                sh 'rm -f ./front/front_0.1.0.tar'
                echo 'Cleanup success !'
            }
        }
        stage('secret.yml copy') {
            steps {
                withCredentials([file(credentialsId: 'applica
                    script {
                        sh 'docker cp $secretFile jenkins:/va
                    }
                }
            }
        }
        stage('fcmjson copy') {
            steps {
```

```
                withCredentials([file(credentialsId: 'fcmjson
                    script {
                        sh 'docker cp $secretFile jenkins:/va
                    }
                }
            }
        }
        stage('front.env.copy'){
            steps{
                withCredentials([file(credentialsId: 'songpic
                    script{
                        sh 'docker cp $secretFile jenkins:/va
                    }
                }
            }
        }
        stage('Build Front') {
            steps {
                dir("./front") {
                    nodejs(nodeJSInstallationName: 'NodeJS 22
                        sh 'rm -rf node_modules'
                        sh 'rm -rf package-lock.json'
                        sh 'CI=false npm install'
                        sh 'CI=false npm run build'
                    }
                }
                echo 'FE Build success !'
            }
        }
        stage('Compression') {
            steps {
                dir("./front/dist") {
                    sh 'tar -czvf ../front_0.1.0.tar .'
                }
                echo 'Compression success'
            }
        }
```

```
        stage('Build') {
            steps {
                sh 'chmod +x ./back/gradlew'
                sh 'cd ./back && ./gradlew clean build -x tes
            }
        }
        stage('Docker Hub Login') {
            steps {
                withCredentials([usernamePassword(credentials
                    sh 'echo "$DOCKER_PASSWORD" | docker logi
                }
            }
        }
        stage('Docker Build and Push') {
            steps {
                withCredentials([usernamePassword(credentials
                    sh 'cd ./back && docker build -f Dockerfi
                    sh 'cd ./back && docker push $DOCKER_REPO
                    echo 'docker push Success!!'
                }
                echo 'docker push Success!!'
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['my-ssh-crenditials'])
                    withCredentials([string(credentialsId: 'E
                        sh 'ssh -o StrictHostKeyChecking=no u
                        sh '''
                        ssh -o StrictHostKeyChecking=no ubunt
                        ssh ubuntu@$IP 'sudo chmod -R 777 /ho
                        scp /var/jenkins_home/workspace/songp
                        ssh -t ubuntu@$IP sudo sh ./deploy_fe
                        '''
                    }
                }
            }
```

```
        }
        stage('Notification') {
    steps{
        echo 'jenkins notification!'
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend(color: 'good',
                    message: "빌드 성공: ${env.JOB_NAME} #${env
                    endpoint: 'https://meeting.ssafy.com/hook
                    channel: 'd208_build_result'
                        )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend(color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env
                    endpoint: 'https://meeting.ssafy.com/hook
                    channel: 'd208_build_result'
                        )
            }
        }
    }
}
    }
}
```

## Deploy File 작성

```
#!/bin/bash
sudo docker compose -p songpicker-8080 -f /home/ubuntu/docker

sudo docker compose -p songpicker-8081 -f /home/ubuntu/docker

EXIST_GITCHAN=$(sudo docker compose -p songpicker-8080 -f /ho

if [ -z "$EXIST_GITCHAN" ]; then
    echo "8080 컨테이너 실행"
    sudo docker compose -p songpicker-8080 -f /home/ubuntu/do
    BEFORE_COLOR="8081"
    AFTER_COLOR="8080"
    BEFORE_PORT=8081
    AFTER_PORT=8080
else
    echo "8081 컨테이너 실행"
    sudo docker compose -p songpicker-8081 -f /home/ubuntu/do
    BEFORE_COLOR="8080"
    AFTER_COLOR="8081"
    BEFORE_PORT=8080
    AFTER_PORT=8081
fi

echo "${AFTER_COLOR} server up(port:${AFTER_PORT})"

# Nginx 설정 파일 직접 수정
NginxConfig="/etc/nginx/sites-enabled/default"
if [ -f "$NginxConfig" ]; then
    # 현재 포트를 찾고 새로운 포트로 변경
sudo cp "$NginxConfig" "$NginxConfig.bak"  # 백업

sudo sed -i "s|proxy_pass http://localhost:[0-9]*/;|proxy_pas

# 변경된 내용을 확인
if grep -q "proxy_pass http://localhost:${AFTER_PORT}/;" "$Ng
    echo "Nginx 설정 파일 업데이트 성공"
else
    echo "Nginx 설정 파일 업데이트 실패"
```

```
    exit 1
fi
else
    echo "Nginx 설정 파일이 존재하지 않습니다."
    exit 1
fi

# Nginx 설정 테스트 및 재시작
if sudo nginx -t && sudo systemctl reload nginx; then
    echo "Nginx 재시작 성공"
else
    echo "Nginx 재시작 실패"
    exit 1
fi

# Nginx 재시작
sudo nginx -s reload
echo "Deploy Completed!!"

# 이전 서버 중지
echo "$BEFORE_COLOR server down(port:${BEFORE_PORT})"
sudo docker compose -p songpicker-${BEFORE_COLOR} -f /home/ub

# Docker 이미지 정리
sudo docker image prune -f
```

## Docker-compose File 작성

docker-compose.nativenavs8080.yml

```
version: '3.1'

services:
  api:
    image: wonseunghyeon/songpicker:latest
    container_name: songpicker-8080
    environment:
```

```
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8080
        #- jasypt.encryptor.key=[KEY VALUE]
    ports:
      - '8080:8080'
```

docker-compose.nativenavs8081.yml

```
version: '3.1'

services:
  api:
    image: wonseunghyeon/songpicker:latest
    container_name: songpicker-8081
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8081
        #- jasypt.encryptor.key=[KEY VALUE]
    ports:
      - '8081:8080'
```

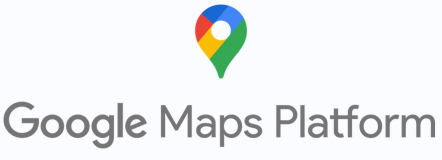
# 프론트 엔드 환경 구축

Jenkins Plugins에 NodeJS 추가

Google Maps Platform | Google for Developers

수백만 개의 웹사이트와 앱이 Google Maps Platform을 사용하여 사용자에게 효과적인 서비스 환경을 제공하고 있습니다.

https://developers.google.com/maps?hl=ko




## Jenkins pipline 작성

위 백엔드 파이프라인에 통합 구현

## deploy_fe.sh 작성

```
DIST_DIR="/home/ubuntu/Front/dist"

# 압축 해제할 임시 디렉토리 생성
TEMP_DIR="/home/ubuntu/Front/temp"
sudo mkdir -p "$TEMP_DIR"

# 압축 해제
tar -xvf /home/ubuntu/Front/front_0.1.0.tar -C "$TEMP_DIR"

# 기존 dist 디렉토리가 있다면 삭제
if [ -d "$DIST_DIR" ]; then
    sudo rm -rf "$DIST_DIR"
fi

# dist 디렉토리를 생성
sudo mkdir -p "$DIST_DIR"

# 임시 디렉토리의 내용을 dist로 이동
sudo mv "$TEMP_DIR"/* "$DIST_DIR"

# 임시 디렉토리 삭제
sudo rm -rf "$TEMP_DIR"

# Nginx 설정 재적용
sudo nginx -s reload
```

---

# 노래방 서버 구축

## 젠킨스 파이프라인 songpicker-karaoke

```
pipeline {
    agent any

    tools {
```

```
        jdk ("jdk17")
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'release/karaoke', credentialsId:
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
        stage('Clean') {
            steps {
                sh 'rm -f ./karaoke/front/front_0.1.0.tar'
                echo 'Cleanup success !'
            }
        }
        stage('.env copy') {
            steps {
                withCredentials([file(credentialsId: 'karaoke
                    script {
                        sh 'docker cp $secretFile jenkins:/va
                    }
                }
            }
        }
        stage('Build Front') {
            steps {
                dir("./karaoke/front") {
                    nodejs(nodeJSInstallationName: 'NodeJS 22
                        sh 'rm -rf node_modules'
                        sh 'rm -rf package-lock.json'
```

```
                    sh 'CI=false npm install'
                    sh 'CI=false npm run build'
                }
            }
            echo 'FE Build success !'
        }
    }
    stage('Compression') {
        steps {
            dir("./karaoke/front/dist") {
                sh 'tar -czvf ../front_0.1.0.tar .'
            }
            echo 'Compression success'
        }
    }
    stage('secret.yml copy') {
        steps {
            withCredentials([file(credentialsId: 'applica
                script {
                    sh 'docker cp $secretFile jenkins:/va
                }
            }
        }
    }

    stage('Build') {
        steps {
            sh 'chmod +x ./karaoke/back/gradlew'
            sh 'cd ./karaoke/back && ./gradlew clean buil
        }
    }
    stage('Docker Hub Login') {
        steps {
            withCredentials([usernamePassword(credentials
                sh 'echo "$DOCKER_PASSWORD" | docker logi
            }
        }
    }
```

```
        stage('Docker Build and Push') {
            steps {
                withCredentials([usernamePassword(credentials
                    sh 'cd ./karaoke/back && docker build -f
                    sh 'cd ./karaoke/back && docker push $DOC
                    echo 'docker push Success!!'
                }
                echo 'docker push Success!!'
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['my-ssh-crenditials'])
                    withCredentials([string(credentialsId: 'E
                        sh 'ssh -o StrictHostKeyChecking=no u
                        sh '''
                        ssh -o StrictHostKeyChecking=no ubunt
                        ssh ubuntu@$IP 'sudo chmod -R 777 /ho
                        scp /var/jenkins_home/workspace/songp
                        ssh -t ubuntu@$IP sudo sh ./deploy_ka
                        '''
                    }
                }
            }
        }
        stage('Notification') {
            steps {
                echo 'jenkins notification!'
            }
            post {
                success {
                    script {
                        def Author_ID = sh(script: "git show
                        def Author_Name = sh(script: "git sho
                        mattermostSend(color: 'good',
                            message: "빌드 성공: ${env.JOB_NAME
                            endpoint: 'https://meeting.ssafy.
                            channel: 'd208_build_result'
```

```
                        )
                    }
                }
                failure {
                    script {
                        def Author_ID = sh(script: "git show
                        def Author_Name = sh(script: "git sho
                        mattermostSend(color: 'danger',
                            message: "빌드 실패: ${env.JOB_NAME
                            endpoint: 'https://meeting.ssafy.
                            channel: 'd208_build_result'
                        )
                    }
                }
            }
        }
    }
}
```

## 프론트엔드 배포 deploy_karaoke_fe.sh

```
#!/bin/bash

# dist 디렉토리의 경로를 정의
DIST_DIR="/home/ubuntu/karaoke_front/dist"

# 압축 해제할 임시 디렉토리 생성
TEMP_DIR="/home/ubuntu/karaoke_front/temp"
sudo mkdir -p "$TEMP_DIR"

# 압축 해제
tar -xvf /home/ubuntu/karaoke_front/front_0.1.0.tar -C "$TEMP

# 기존 dist 디렉토리가 있다면 삭제
if [ -d "$DIST_DIR" ]; then
    sudo rm -rf "$DIST_DIR"
fi
```

```
# dist 디렉토리를 생성
sudo mkdir -p "$DIST_DIR"

# 임시 디렉토리의 내용을 dist로 이동
sudo mv "$TEMP_DIR"/* "$DIST_DIR"

# 임시 디렉토리 삭제
sudo rm -rf "$TEMP_DIR"

# Nginx 설정 재적용
sudo nginx -s reload
```

## 백엔드 배포 deploy_karaoke.sh

```
#!/bin/bash
sudo docker compose -p karaoke-9090 -f /home/ubuntu/docker-co

EXIST_GITCHAN=$(sudo docker compose -p karaoke-9090 -f /home/

if [ ! -z "$EXIST_GITCHAN" ]; then
    echo "기존 9090 컨테이너 중단"
    sudo docker compose -p karaoke-9090 -f /home/ubuntu/docke
fi

# 2
# 새로운 컨테이너 시작
echo "9090 컨테이너 실행"
sudo docker compose -p karaoke-9090 -f /home/ubuntu/docker-co


# Docker 이미지 정리
sudo docker image prune -f
```

## docker-compose.karaoke9090.yml

```
version: '3.1'

services:
  api:
    image: wonseunghyeon/karaoke:latest
    container_name: karaoke-9090
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=9090
        #- jasypt.encryptor.key=[KEY VALUE]
    ports:
      - '9090:8080'

~
```

## Django 서버 구축(추천 알고리즘 데이터 서버)

### 젠킨스 파이프라인

```
pipeline {
    agent any

//  tools {
//         python 'Python3'
//     }

    stages {
        stage('Git Clone'){
            steps {
                git branch: 'release/data', credentialsId: 'G
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
```

```
                    echo 'Repository clone success !'
                }
            }
        }
        stage('.env copy') {
            steps {
                withCredentials([file(credentialsId: 'env', v
                    script {
                        sh 'docker cp $secretFile jenkins:/va
                    }
                }
            }
        }
        stage('Check Python Installation') {
            steps {
                sh 'python3 --version'
            }
        }

        stage('Set Up Python Environment') {
            steps {
                script{
                    dir('data'){
                        // Python 환경 설정 (가상환경 등)
                        sh 'python3 -m venv venv'
                        sh '. venv/bin/activate && pip instal
                        echo 'Python environment setup succes
                    }
                }
            }
        }

        stage('Run Migrations') {
            steps {
                 script{
                    dir('data'){
                        sh '. venv/bin/activate && python man
                        echo 'Database migrations executed su
```

```
                }
            }
        }
    }

    // stage('Collect Static Files') {
    //     steps {
    //         script{
    //             dir('data'){
    //                 sh '. venv/bin/activate && python m
    //                 echo 'Static files collected succe
    //             }
    //         }
    //     }
    // }

    stage('Build Docker Image') {
        steps {

             script{
                dir('data'){
                    sh 'docker build -t django_app .'
                    echo 'Docker image built successfully
                }
            }
        }
    }
     stage('Docker Hub Login') {
        steps {
            withCredentials([usernamePassword(credentials
                sh 'echo "$DOCKER_PASSWORD" | docker logi
            }
        }
    }
    stage('Docker Build and Push') {
        steps {
            withCredentials([usernamePassword(credentials
                sh 'cd ./data && docker build -f Dockerfi
```

```
                    sh 'cd ./data && docker push wonseunghyeo
                    echo 'docker push Success!!'

                }
                echo 'docker push Success!!'
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['my-ssh-crenditials'])
                    withCredentials([string(credentialsId: 'E
                        sh 'ssh -o StrictHostKeyChecking=no u
                    }
                }
            }
        }

        stage('Notification') {
            steps {
                echo 'jenkins notification!'
            }
            post {
                success {
                    script {
                        def Author_ID = sh(script: "git show
                        def Author_Name = sh(script: "git sho
                        mattermostSend(color: 'good',
                            message: "Django Server 빌드 성공:
                            endpoint: 'https://meeting.ssafy.
                            channel: 'd208_build_result'
                        )
                    }
                }
                failure {
                    script {
                        def Author_ID = sh(script: "git show
                        def Author_Name = sh(script: "git sho
                        mattermostSend(color: 'danger',
```

```
                                message: "Django Server 빌드 실패:
                                endpoint: 'https://meeting.ssafy.
                                channel: 'd208_build_result'
                            )
                        }
                    }
                }
            }


    }
}
```

## deploy_data.sh

```
#!/bin/bash

# Docker 이미지 이름과 태그
IMAGE_NAME="wonseunghyeon/data:latest"
CONTAINER_NAME="django_app_container"

# 기존 컨테이너가 실행 중인지 확인하고 중지 및 제거
if [ "$(docker ps -q -f name=$CONTAINER_NAME)" ]; then
    echo "Stopping existing container..."
    docker stop $CONTAINER_NAME
    echo "Removing existing container..."
    docker rm $CONTAINER_NAME
fi

# 새로운 Docker 이미지 풀링
echo "Pulling the latest image..."
if ! docker pull $IMAGE_NAME; then
        echo "Failed to pull image: $IMAGE_NAME"
        exit 1
fi

# 새로운 컨테이너 실행
echo "Starting new container..."
```

```
docker run -d \
    --name $CONTAINER_NAME \
    -p 8000:8000 \
    $IMAGE_NAME
#   --env-file /var/jenkins_home/workspace/songpicker-data/da
  #  $IMAGE_NAME

# 컨테이너 상태 확인
if [ "$(docker ps -q -f name=$CONTAINER_NAME)" ]; then
    echo "Deployment successful!"
else
    echo "Deployment failed!"
    exit 1
fi
```