Git 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

1. 사용한 JVM 웹서버, WAS 제품 등의 종류와 설정 값 버전 (IDE 버전 포함) 기재

Skill	Version
JVM(JDK)	17
intellij IDEA	2024.3.1.1
SpringBoot	3 이상
MySQL	8.0
Redis	7.0
Python	3.11
FastAPI	0.95.2
Jenkins	2.492.3
docker compose	2.35.1
node	20.10.0
React	19.00

포트 매핑

서비스	서비스 설명	기술 스택	포트
server_rag	주문 파싱 및 데이터 구조화 기능 서버	Spring	8000
server_api	api 요청 처리 서버	FastAPI	8081
server_voice	stt - tts와 같은 음성-텍스트 변환 기능 서버	FastAPI	8005
front_admin	관리자 페이지 fe 컨테이너	React	3001
front_kiosk	키오스크 페이지 fe 컨테이너	React	3000
nginx-lb	로드 밸런싱 및 리버스 프록시 제어, https 연결	Nginx	80, 443
mylio-cache	서비스의 캐시 메모리 DB	Redis	6379

서비스	서비스 설명	기술 스택	포트
mylio-db	서비스 DB	MySQL	2.492.3
jenkins	서비스 CI/CD	Jenkins	8080

2. 빌드 시 사용되는 환경 변수 등의 내용 상세 기재

```
MYSQL_CONTAINER_NAME=mylio-db
MYSQL_USER=mylio
MYSQL_DATABASE=mylio_db
MYSQL_URL=jdbc:mysql://mylio-db:3306/mylio_db
MYSQL_PORT=3307
MYSQL_CONTAINER_PORT=3306
MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
MYSQL_PASSWORD=${MYSQL_PASSWORD}  //mylio가최고임

REDIS_CONTAINER_NAME=mylio-cache
REDIS_HOST=mylio-cache
REDIS_PORT=6379
REDIS_PASSWORD=${REDIS_PASSWORD}  //mylio가최고다

SPRING_IMAGE_NAME=spring-app
SPRING_CONTAINER_NAME=server_api
SPRING_PORT=8081
SPRING_CONTAINER_PORT=8080
JWT_SECRET_KEY=${JWT_SECRET_KEY}
OPEN_AI_KEY=${OPEN_AI_KEY}

VOICE_IMAGE_NAME=fastapi-voice
VOICE_CONTAINER_NAME=server_voice
VOICE_PORT=8005
VOICE_CONTAINER_PORT=8000

CLOVA_API_INVOKE_URL=https://clovaspeech-gw.ncloud.com/recog/v1/stt
CLOVA_API_SECRET=${CLOVA_API_SECRET}
CLOVA_API_KEY_ID=${CLOVA_API_KEY_ID}
```

```
GOOGLE_TTS_API_URL=texttospeech.googleapis.com/v1/voices
GOOGLE_TTS_API_SECRET=${GOOGLE_TTS_API_SECRET}

RAG_IMAGE_NAME=fastapi-rag
RAG_CONTAINER_NAME=server_rag
RAG_PORT=8000
RAG_CONTAINER_PORT=8000



KIOSK_IMAGE_NAME=kiosk-react
KIOSK_PORT=3000
KIOSK_CONTAINER_PORT=80
KIOSK_CONTAINER_NAME=front_kiosk
KIOSK_BASE_PATH=/kiosk/

KAKAO_READY_URL=https://open-api.kakaopay.com/online/v1/payment/ready
KAKAO_APPROVE_URL=https://open-api.kakaopay.com/online/v1/payment/ap
KAKAO_PAY_SECRET=${KAKAO_PAY_SECRET}

VITE_PUBLIC_API_URL=https://k12b102.p.ssafy.io/api
ALLOWED_ORIGINS=http://localhost:5173

S3_ACCESS_KEY=${S3_ACCESS_KEY}
S3_SECRET_KEY=${S3_SECRET_KEY}



ADMIN_IMAGE_NAME=admin-react
ADMIN_PORT=3001
ADMIN_CONTAINER_PORT=80
ADMIN_CONTAINER_NAME=front_admin
```

※ gcp-tts-key.json을 호스트에 저장 필요

- 호스트의 아래 폴더 위치에 gcp-tts-key.json 파일 저장

  `/home/ubuntu/MyLio/frontend/tts`

gcp-tts-key.json

- 키 값인 만큼 보안 설정 필요

```
#읽기만 허용 명령어 -> 불필요한 읽기/쓰기 차단
chmod 400 /home/ubuntu/secrets/gcp-tts-key.json

#소유자(owner)와 소유 그룹(group)을 변경 -> root만 일을 수 있게 제한
sudo chown root:root /home/ubuntu/MyLio/etc/gcp-tts-key.json
```

- fastapi 서버 빌드 시 /run/secrets 경로에 읽기 전용으로 mount

```
volumes:      - /home/ubuntu/MyLio/frontend/tts:/run/secrets:ro
```

3. DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

MySQL

- database : mylio_db
- 주요 프로퍼티
  - 계정 정보
    - 서비스 데이터 사용 가능한 계정들 데이터
      - 관리자 계정
      - 키오스크 계정
  - 메뉴 정보
    - 매장에서 판매하는 메뉴 정보
  - 영양성분 정보
    - 영양 성분 정보
    - 메뉴별 영양성분 매핑 정보

- 원재료 정보
  - 원재료 정보
  - 메뉴별 원재료 매핑 정보
- 옵션 정보
  - 옵션 상세값 정보
  - 메뉴별 옵션 매핑 정보
- 주문
  - 주문 데이터 정보
- 통계
  - 일별, 월별, 연도별 주문 통계 데이터 정보

4. 배포 시 특이 사항 기재

- docker compose 사용
- LoadBalancing 용 Nginx 설정 및 Let’s Encrypt로 https 설정
- Jenkins - Gtilab 연동 CI/CD 구축

5. 프로젝트 배포 방법

1. Jenkins 컨테이너 빌드

```
docker run -d \
  --name jenkins \
  --group-add $(stat -c '%g' /var/run/docker.sock) \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v jenkins_home:/var/jenkins_home \
  -p 8080:8080 \
  -p 50000:50000 \
     jenkins/jenkins:lts
```

Jenkins 에서 Docker 를 사용하므로 Jenkins 컨테이너 안에 Docker 를 설치한다.

```
 # Docker 설치
apt-get update && apt-get install -y docker.io

# jenkins 사용자에게 권한 부여
groupadd docker
usermod -aG docker jenkins
newgrp docker

# Docker-Compose 설치
apt-get update && apt-get install -y curl
curl -L "https://github.com/docker/compose/releases/latest/download/docker
chmod +x /usr/local/bin/docker-compose
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

3. docker compose 파일 작성

```
 version: '3.8'

services:
  nginx:
    image: nginx:latest
    container_name: nginx-lb
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /home/ubuntu/MyLio/nginx/conf.d:/etc/nginx/conf.d
      - /etc/letsencrypt:/etc/letsencrypt:ro
    networks:
      - app-network
```

```
  mysql:
    image: mysql:8.0
    container_name: ${MYSQL_CONTAINER_NAME}
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      LANG: C.UTF-8
    ports:
      - "${MYSQL_PORT}:${MYSQL_CONTAINER_PORT}"
    volumes:
      - /home/ubuntu/MyLio/mylio_data:/var/lib/mysql
    networks:
      - app-network

  redis:
    image: redis:7-alpine
    container_name: ${REDIS_CONTAINER_NAME}
    restart: always
    ports:
      - "${REDIS_PORT}:${REDIS_PORT}"
    volumes:
      - /home/ubuntu/MyLio/redis_data:/data
    command: redis-server --appendonly yes --requirepass ${REDIS_PASSWO
    networks:
      - app-network

  spring-app:
    build:
      context: ./BE
      dockerfile: Dockerfile
    image: ${SPRING_IMAGE_NAME}
    container_name: ${SPRING_CONTAINER_NAME}
    restart: always
    ports:
      - "${SPRING_PORT}:${SPRING_CONTAINER_PORT}"
```

```
    environment:
      #my sql
      MYSQL_CONTAINER_NAME: ${MYSQL_CONTAINER_NAME}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_URL: ${MYSQL_URL}

      #redis 설정
      REDIS_HOST: ${REDIS_HOST}
      REDIS_PORT: ${REDIS_PORT}
      REDIS_PASSWORD: ${REDIS_PASSWORD}

      JWT_SECRET_KEY: ${JWT_SECRET_KEY}
      OPEN_AI_KEY: ${OPEN_AI_KEY}

      S3_ACCESS_KEY: ${S3_ACCESS_KEY}
      S3_SECRET_KEY: ${S3_SECRET_KEY}

      KAKAO_PAY_SECRET: ${KAKAO_PAY_SECRET}
      KAKAO_READY_URL: ${KAKAO_READY_URL}
      KAKAO_APPROVE_URL: ${KAKAO_APPROVE_URL}

    networks:
      - app-network

  fastapi-voice:
    build:
      context: ./AI/Clova
      dockerfile: Dockerfile
    image: ${VOICE_IMAGE_NAME}
    container_name: ${VOICE_CONTAINER_NAME}
    environment:
      CLOVA_API_INVOKE_URL: ${CLOVA_API_INVOKE_URL}
      CLOVA_API_KEY_ID: ${CLOVA_API_KEY_ID}
      CLOVA_API_SECRET: ${CLOVA_API_SECRET}
      GOOGLE_TTS_API_URL: ${GOOGLE_TTS_API_URL}
      GOOGLE_TTS_API_SECRET: ${GOOGLE_TTS_API_SECRET}
      ALLOWED_ORIGINS: ${ALLOWED_ORIGINS}
```

```
    GOOGLE_APPLICATION_CREDENTIALS: /run/secrets/gcp-tts-key.json
  volumes:
    - /home/ubuntu/MyLio/frontend/tts:/run/secrets:ro

  ports:
    - "${VOICE_PORT}:${VOICE_CONTAINER_PORT}"
  restart: always
  networks:
    - app-network

chroma:
  image: chromadb/chroma:0.4.24
  container_name: ${CHROMA_CONTAINER_NAME}
  ports:
    - "${CHROMA_PORT}:${CHROMA_CONTAINER_PORT}"
  volumes:
    - /home/ubuntu/MyLio/chroma-data:/chroma/.chroma
  networks:
    - app-network

fastapi-rag:
  build:
    context: ./AI/ai-service
    dockerfile: Dockerfile
  image: ${RAG_IMAGE_NAME}
  container_name: ${RAG_CONTAINER_NAME}
  ports:
    - "${RAG_PORT}:${RAG_CONTAINER_PORT}"
  environment:
    MYSQL_HOST: ${MYSQL_CONTAINER_NAME}
    MYSQL_PORT: ${MYSQL_CONTAINER_PORT}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASS: ${MYSQL_PASSWORD}
    MYSQL_DB: ${MYSQL_DATABASE}
    OPENAI_API_KEY: ${OPEN_AI_KEY}
    REDIS_HOST: ${REDIS_HOST}
    REDIS_PORT: ${REDIS_PORT}
    REDIS_PASSWORD: ${REDIS_PASSWORD}
```

```
      CHROMA_HOST: ${CHROMA_HOST}
      COLLECTION_NAME: ${COLLECTION_NAME}
    depends_on:
      chroma:
        condition: service_started
      redis:
        condition: service_started
    restart: always
    networks:
      - app-network

  react-admin:
    build:
      context: ./FE/admin
      dockerfile: Dockerfile
      args:
        VITE_PUBLIC_API_URL: ${VITE_PUBLIC_API_URL}
        VITE_OPENAI_API_KEY: ${VITE_OPENAI_API_KEY}
    image: ${ADMIN_IMAGE_NAME}
    container_name: ${ADMIN_CONTAINER_NAME}
    ports:
      - "${ADMIN_PORT}:${ADMIN_CONTAINER_PORT}"
    volumes:
      - /home/ubuntu/MyLio/frontend/admin/default.conf:/etc/nginx/conf.d/defa
    networks:
      - app-network

  react-kiosk:
    build:
      context: ./FE/kiosk
      dockerfile: Dockerfile
      args:
        BASE_PATH: ${KIOSK_BASE_PATH}
        VITE_PUBLIC_API_URL: ${VITE_PUBLIC_API_URL}
        VITE_PUBLIC_VOICE_API_URL: ${VITE_PUBLIC_VOICE_API_URL}
        VITE_OPENAI_API_KEY: ${VITE_OPENAI_API_KEY}
    image: ${KIOSK_IMAGE_NAME}
    container_name: ${KIOSK_CONTAINER_NAME}
```

```
    ports:
      - "${KIOSK_PORT}:${KIOSK_CONTAINER_PORT}"
    volumes:
      - /home/ubuntu/MyLio/frontend/kiosk/default.conf:/etc/nginx/conf.d/defau
    networks:
      - app-network


networks:
  app-network:
    driver: bridge
```

3. Jenkins - Gitlab 연동 후 Jenkins Shell 빌드

```
#!/bin/bash

# 운영 환경 변수 설정

MYSQL_CONTAINER_NAME=mylio-db
MYSQL_USER=mylio
MYSQL_DATABASE=mylio_db
MYSQL_URL=jdbc:mysql://mylio-db:3306/mylio_db
MYSQL_PORT=3307
MYSQL_CONTAINER_PORT=3306

REDIS_CONTAINER_NAME=mylio-cache
REDIS_HOST=mylio-cache
REDIS_PORT=6379

SPRING_IMAGE_NAME=spring-app
SPRING_CONTAINER_NAME=server_api
SPRING_PORT=8081
SPRING_CONTAINER_PORT=8080

VOICE_IMAGE_NAME=fastapi-voice
VOICE_CONTAINER_NAME=server_voice
VOICE_PORT=8005
VOICE_CONTAINER_PORT=8000
```

```
RAG_IMAGE_NAME=fastapi-rag
RAG_CONTAINER_NAME=server_rag
RAG_PORT=8000
RAG_CONTAINER_PORT=8000

#CHROMA_HOST=chroma
CHROMA_HOST=http://chroma:8000
CHROMA_CONTAINER_NAME=mylio-vector
CHROMA_PORT=9000
CHROMA_CONTAINER_PORT=8000
COLLECTION_NAME=menu_embeddings

KIOSK_IMAGE_NAME=kiosk-react
KIOSK_PORT=3000
KIOSK_CONTAINER_PORT=80
KIOSK_CONTAINER_NAME=front_kiosk
KIOSK_BASE_PATH=/kiosk/

KAKAO_READY_URL=https://open-api.kakaopay.com/online/v1/payment/ready
KAKAO_APPROVE_URL=https://open-api.kakaopay.com/online/v1/payment/ap

ADMIN_IMAGE_NAME=admin-react
ADMIN_PORT=3001
ADMIN_CONTAINER_PORT=80
ADMIN_CONTAINER_NAME=front_admin

echo "🧹 기존 컨테이너 중지 및 삭제"
docker-compose down || true

# 🔄 기존 이미지 삭제
echo "🔄 기존 이미지 확인..."
for image in $SPRING_IMAGE_NAME $FASTAPI_IMAGE_NAME $VOICE_IMAGE
    if [ "$(docker images -q $image)" ]; then
        echo "⚠️ 기존 이미지 ($image) 삭제..."
        docker rmi -f $image
    fi
done
```

```
# 📝 기존 .env 파일 삭제 후 새로 생성
echo "📝 기존 .env 파일 확인..."
if [ -f ".env" ]; then
    echo "⚠️ 기존 .env 파일 삭제..."
    rm .env
fi


echo "🚀 새로운 .env 파일 생성"
cat <<EOF > .env
MYSQL_CONTAINER_NAME=$MYSQL_CONTAINER_NAME
MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD
MYSQL_USER=$MYSQL_USER
MYSQL_PASSWORD=$MYSQL_PASSWORD
MYSQL_DATABASE=$MYSQL_DATABASE
MYSQL_URL=$MYSQL_URL
MYSQL_PORT=$MYSQL_PORT
MYSQL_CONTAINER_PORT=$MYSQL_CONTAINER_PORT

REDIS_CONTAINER_NAME=$REDIS_CONTAINER_NAME
REDIS_HOST=$REDIS_HOST
REDIS_PORT=$REDIS_PORT
REDIS_PASSWORD=$REDIS_PASSWORD

SPRING_IMAGE_NAME=$SPRING_IMAGE_NAME
SPRING_CONTAINER_NAME=$SPRING_CONTAINER_NAME
SPRING_PORT=$SPRING_PORT
SPRING_CONTAINER_PORT=$SPRING_CONTAINER_PORT

S3_ACCESS_KEY=$S3_ACCESS_KEY
S3_SECRET_KEY=$S3_SECRET_KEY

KAKAO_PAY_SECRET=$KAKAO_PAY_SECRET
KAKAO_READY_URL=$KAKAO_READY_URL
KAKAO_APPROVE_URL=$KAKAO_APPROVE_URL

VOICE_IMAGE_NAME=$VOICE_IMAGE_NAME
```

```
VOICE_CONTAINER_NAME=$VOICE_CONTAINER_NAME
VOICE_PORT=$VOICE_PORT
VOICE_CONTAINER_PORT=$VOICE_CONTAINER_PORT

CLOVA_API_INVOKE_URL=$CLOVA_API_INVOKE_URL
CLOVA_API_KEY_ID=$CLOVA_API_KEY_ID
CLOVA_API_SECRET=$CLOVA_API_SECRET
ALLOWED_ORIGINS=$ALLOWED_ORIGINS

GOOGLE_TTS_API_URL=$GOOGLE_TTS_API_URL
GOOGLE_TTS_API_SECRET=$GOOGLE_TTS_API_SECRET
GCP_KEY_PATH=$GCP_KEY_PATH

RAG_IMAGE_NAME=$RAG_IMAGE_NAME
RAG_CONTAINER_NAME=$RAG_CONTAINER_NAME
RAG_PORT=$RAG_PORT
RAG_CONTAINER_PORT=$RAG_CONTAINER_PORT

CHROMA_HOST=$CHROMA_HOST
CHROMA_CONTAINER_NAME=$CHROMA_CONTAINER_NAME
CHROMA_PORT=$CHROMA_PORT
CHROMA_CONTAINER_PORT=$CHROMA_CONTAINER_PORT
COLLECTION_NAME=$COLLECTION_NAME

VITE_PUBLIC_VOICE_API_URL=$VITE_PUBLIC_VOICE_API_URL
VITE_PUBLIC_API_URL=$VITE_PUBLIC_API_URL
VITE_OPENAI_API_KEY=$OPEN_AI_KEY

KIOSK_IMAGE_NAME=$KIOSK_IMAGE_NAME
KIOSK_PORT=$KIOSK_PORT
KIOSK_CONTAINER_PORT=$KIOSK_CONTAINER_PORT
KIOSK_CONTAINER_NAME=$KIOSK_CONTAINER_NAME
KIOSK_BASE_PATH=$KIOSK_BASE_PATH

ADMIN_IMAGE_NAME=$ADMIN_IMAGE_NAME
ADMIN_PORT=$ADMIN_PORT
ADMIN_CONTAINER_PORT=$ADMIN_CONTAINER_PORT
ADMIN_CONTAINER_NAME=$ADMIN_CONTAINER_NAME
```

```
JWT_SECRET_KEY=$JWT_SECRET_KEY

OPEN_AI_KEY=$OPEN_AI_KEY
EOF


# 🔄 Docker 네트워크 확인 및 생성
echo "🔄 Docker 네트워크 확인..."
if ! docker network ls | grep -q "app-network"; then
    echo "🚀 네트워크(app-network) 생성..."
    docker network create app-network
fi

# 🚀 컨테이너 실행
echo "🚀 새로운 컨테이너 실행"
docker-compose up -d --build
```