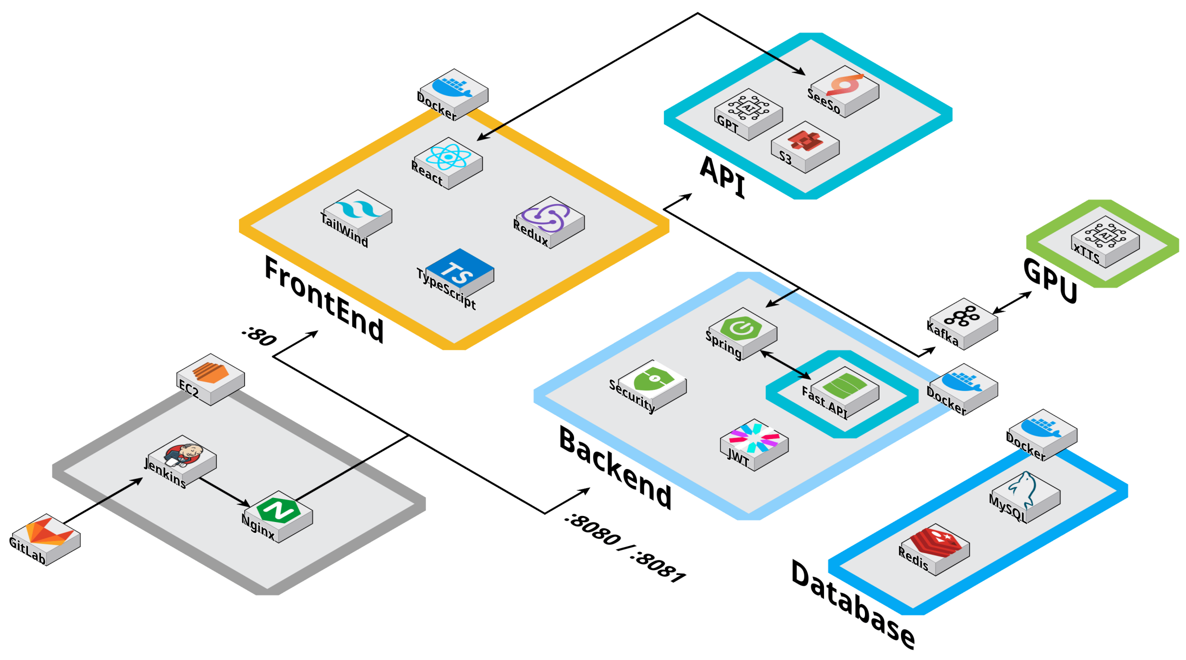
D105_포팅 매뉴얼

1. 아키텍쳐 구상도



---

2. 버전

1) 백엔드

- IntelliJ IDEA ultimate: 2024.2.0.1
- JDK : 17
- Spring Boot: 3.3.4
- MySQL: 8.0
- Redis: 7.4.0
- Kafka: 2.8.1
- Elastic Search: 8.5.3
- Longstash: 8.5.3
- JPA: 3.3.4

2) 프론트엔드

- VSCode: 1.91.0
- React: 18.3.1
- Node.js: 20.9.0
- TypeScript: 5.6.2
- Tailwind: 3.4.14
- Redux: 9.1.2

3) CI/CD

- Jenkins: 2.462.3
- Docker: 27.1.1
- Nginx: 1.24.0

---

3. 환경변수

1) 백엔드

```
MYSQL_HOST=k11d105.p.ssafy.io
MYSQL_PORT=2020
MYSQL_DB=ijoa
MYSQL_USER=ijoa
MYSQL_PASSWORD=dkdlwhdk105
REDIS_HOST=k11d105.p.ssafy.io
REDIS_PASSWORD=dkdlwhdk105
DDL_AUTO_OPTION=update
CORS_URL=https://k11d105.p.ssafy.io, http://localhost:5173, https://ijoaa.com, https://www.ijoaa.com
EMAIL_USER=checkitoutd105@gmail.com
EMAIL_PASSWORD=fcyioppjqkjpropb
JWT_SECRET_KEY=SSSSAFY_D105_CHECKITOUT_IJOA_JWTSKEY
S3_ACCESSKEY=AKIAVYV52FCPGVFE3U5G
S3_SECRETKEY=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
GIRL_PROFILE_DEFAULT_URL=https://checkitout-bucket.s3.ap-northeast-2.amazonaws.com/profile/girl.png
BOY_PROFILE_DEFAULT_URL=https://checkitout-bucket.s3.ap-northeast-2.amazonaws.com/profile/boy.png
KAFKA_HOST=k11d105.p.ssafy.io
KAFKA_PORT=9092
GPT_KEY=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
CHILD_LEVEL1=1
CHILD_LEVEL2=5
CHILD_LEVEL3=10
RECOMMENDATION_COUNT=8
ELASTIC_URI=http://k11d105.p.ssafy.io:9200
ELASTIC_USER=elastic
ELASTIC_PW=dkdlwhdk105
FASTAPI_URL=http://fastapi:8000
```

2) 프론트엔드

```
VITE_SEESO_SDK_KEY=prod_z06nfpcxvr5wycfqcjr9wdp5bejaazui32u93plj
```

3) Fast API

```
DB_HOST=k11d105.p.ssafy.io
DB_PORT=2020
DB_USER=ijoa
DB_PASSWORD=dkdlwhdk105
```

```
DB_NAME=ijoa
RECOMMEND_LIMIT=8
```

---

4. CI/CD

1) Dockerfile-Jenkins

```
FROM jenkins/jenkins:jdk17

USER root

RUN apt-get update && \
    apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-prope
rties-common && \
    curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
    add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_re
lease -cs) stable" && \
    apt-get update && \
    apt-get install -y docker-ce-cli iputils-ping netcat-openbsd && \
    apt-get clean

RUN groupadd -f docker

RUN usermod -aG docker jenkins

USER jenkins
```

2) docker-compose.yml

```
services:

  mysql:
    image: mysql:8.0
    container_name: mysql
    restart: always
    environment:
      MYSQL_USER: ijoa
      MYSQL_PASSWORD: dkdlwhdk105
      MYSQL_ROOT_PASSWORD: dkdlwhdk105
      MYSQL_DATABASE: ijoa
      LANG: C.UTF-8
      LANGUAGE: C.UTF-8
      LC_ALL: C.UTF-8
      TZ: Asia/Seoul
    ports:
      - "2020:3306"
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
      - --bind-address=0.0.0.0
    volumes:
      - /home/ubuntu/mysql-data:/var/lib/mysql
    networks:
      - ijoa-network
  jenkins:
    build:
      context: .
      dockerfile: Dockerfile-jenkins
    container_name: jenkins
    user: root
    restart: always
    volumes:
      - /home/ubuntu/jenkins-data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - "9090:8080"
      - "50000:50000"
    environment:
      - TZ=Asia/Seoul

  redis:
    image: redis:latest
    container_name: redis
    ports:
      - "6379:6379"
    environment:
      - REDIS_PASSWORD=dkdlwhdk105
    volumes:
      - /home/ubuntu/redis-data:/var/lib/redis
    networks:
      - ijoa-network

  nginx:
    image: nginx:1.24.0
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /home/ubuntu/jenkins-data/workspace/frontend/frontend/ijoa-project/dist:/usr/shar
e/nginx/html
      - /home/ubuntu/docker/nginx/nginx.conf:/etc/nginx/nginx.conf
      - /home/ubuntu/docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
      - /etc/letsencrypt:/etc/letsencrypt:ro
    networks:
      - ijoa-network


  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    platform: linux/amd64
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - "2181:2181"
    networks:
      - ijoa-network

  kafka:
    image: wurstmeister/kafka:latest
    platform: linux/amd64
    depends_on:
      - zookeeper
    environment:
      KAFKA_ADVERTISED_LISTENERS: INSIDE://kafka:29092,OUTSIDE://k11d105.p.ssafy.io:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT
      KAFKA_LISTENERS: INSIDE://0.0.0.0:29092,OUTSIDE://0.0.0.0:9092
      KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    ports:
      - "9092:9092"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - ijoa-network

  kafka-ui:
    image: provectuslabs/kafka-ui:latest
    platform: linux/amd64
    ports:
      - "8085:8085"
    environment:
      SERVER_PORT: 8085
      KAFKA_CLUSTERS_0_NAME: k11d105.p.ssafy.io
      KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka:29092
      KAFKA_CLUSTERS_0_ZOOKEEPER: zookeeper:2181
      KAFKA_CLUSTERS_0_READONLY: "false"
    networks:
      - ijoa-network

  es:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.5.3
    container_name: es
    environment:
      - node.name=es-node
      - cluster.name=search-cluster
      - discovery.type=single-node
      - xpack.security.enabled=true
      - ELASTIC_PASSWORD=dkdlwhdk105
      - ES_JAVA_OPTS=-Xms512m -Xmx512m
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - ijoa-network

  logstash:
    image: docker.elastic.co/logstash/logstash:8.5.3
    container_name: logstash
    volumes:
      - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
      - ./mysql-connector-java-8.0.30.jar:/usr/share/logstash/mysql_driver/mysql-connector
-java-8.0.30.jar
    environment:
      LS_JAVA_OPTS: "-Xmx256m -Xms256m"
      LOG_LEVEL: info
    ports:
      - "5044:5044"
    depends_on:
      - mysql
      - es
    networks:
      - ijoa-network

volumes:
  jenkins_home:
    driver: local

networks:
  ijoa-network:
    external: true
```

3) backend-compose.yml

```
services:

   backend-blue:
    image: backend-blue
    container_name: backend-blue
    environment:
      - SPRING_PROFILES_ACTIVE=blue
      - TZ=Asia/Seoul
    env_file:
      - /home/ubuntu/env/backend/.env
    ports:
      - "8080:8080"
    networks:
      - ijoa-network
    volumes:
      - /home/ubuntu/backend/image:/root/test

   backend-green:
    image: backend-green
    container_name: backend-green
    environment:
      - SPRING_PROFILES_ACTIVE=green
      - TZ=Asia/Seoul
    env_file:
      - /home/ubuntu/env/backend/.env
    ports:
      - "8081:8081"
    networks:
      - ijoa-network
    volumes:
      - /home/ubuntu/backend/image:/root/test


networks:
  ijoa-network:
    external: true
```

4) fastapi-compose.yml

```
services:
  fastapi:
    image: fastapi:latest
    container_name: fastapi
    ports:
      - "8000:8000"
    networks:
      - ijoa-network
    volumes:
      - /home/ubuntu/env/fastapi/.env:/.env
    environment:
      - TZ=Asia/Seoul
    restart: always

networks:
  ijoa-network:
    external: true
```

5) deploy.sh

```
echo "실행"
EXIST_BLUE=$(docker inspect -f '{{.State.Running}}' backend-blue 2>/dev/null)
if [ "$EXIST_BLUE" == "true" ]; then
    echo "Blue server is running. Starting backend-green..."
    docker-compose -f backend-compose.yml up -d backend-green
    BEFORE_COLOR="blue"
    AFTER_COLOR="green"
    BEFORE_PORT=8080
    AFTER_PORT=8081
else
    echo "Blue server is not running. Proceeding with backend-blue..."
    docker-compose -f backend-compose.yml up -d backend-blue
    BEFORE_COLOR="green"
    AFTER_COLOR="blue"
    BEFORE_PORT=8081
    AFTER_PORT=8080
fi

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

# 2
for cnt in {1..10}
do
    echo "===== 서버 응답 확인중(${cnt}/10) =====";
    UP=$(curl -s http://k11d105.p.ssafy.io:${AFTER_PORT}/api/v1/actuator/health)
    if [ -z "${UP}" ]
        then
            sleep 10
            continue
        else
            break
    fi
done

if [ $cnt -eq 10 ]
then
```

```
    echo "===== 서버 실행 실패 ====="
    exit 1
fi

# 3
echo "===== Nginx 설정 변경 ====="
sudo sed -i "s/${BEFORE_PORT}/${AFTER_PORT}/g" /home/ubuntu/docker/nginx/default.conf && d
ocker exec -it nginx nginx -s reload

echo "$BEFORE_COLOR server down(port:${BEFORE_PORT})"
docker-compose -f backend-compose.yml stop backend-${BEFORE_COLOR}
docker-compose restart nginx

# 4. 이전 컨테이너 정리
echo "===== 이전 컨테이너 정리 ====="
if docker ps -a --filter "name=backend-${BEFORE_COLOR}" | grep -q "backend-${BEFORE_COLO
R}"; then
    docker rm -f backend-${BEFORE_COLOR}
fi

docker-compose restart nginx
```

6) nginx.conf

```
user www-data;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

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

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    client_max_body_size 50M;
    include /etc/nginx/conf.d/*.conf;
}
```

7) default.conf

```
server {
    listen       443 ssl;
    server_name k11d105.p.ssafy.io;


    location /api/v1/ {

        proxy_pass http://k11d105.p.ssafy.io:8080;

        add_header Cache-Control "no-store, must-revalidate, no-cache";
        add_header Pragma "no-cache";
        expires -1;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }


    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;

        add_header Cross-Origin-Opener-Policy same-origin;
        add_header Cross-Origin-Embedder-Policy credentialless;
    }

    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }

    ssl_certificate /etc/letsencrypt/live/k11d105.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k11d105.p.ssafy.io/privkey.pem;

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}


server {
    if ($host = k11d105.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    if ($host = www.ijoaa.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
    if ($host = ijoaa.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name ijoaa.com www.ijoaa.com k11d105.p.ssafy.io;

    return 301 https://$host$request_uri;
}
```

---

5. 외부 서비스

1. S3

- 용도: 이미지, TTS 음성 데이터 저장을 위해 사용
- AWS 공식 페이지 : https://aws.amazon.com/ko/pm/serv-s3/?gclid=CjwKCAiAxea5BhBeEiwAh4t5K25uGVSC2O_8xega9_4GMsuGaUXhZRyHjQA5U1cdu_HPUR7Ywa-DfhoCh-gQAvD_BwE&trk=024bf255-8753-410e-9b2f-8015932510e8&sc_channel=ps&ef_id=CjwKCAiAxea5BhBeEiwAh4t5K25uGVSC2O_8xega9_4GMsuGaUXhZRyHjQA5
DfhoCh-gQAvD_BwE:G:s&s_kwcid=AL!4422!3!588924203916!e!!g!!aws%20s3!16390143117!134236388536

2. GPT

- 용도: 아이 흥미 유발용 퀴즈 생성을 위해 사용
- OpenAI GPT 공식 페이지: https://openai.com/

3. SeeSo

- 용도: 사용자의 시선 추적 기술을 활용하여 동화책 읽기 중 집중도 분석
- SeeSo 공식 페이지: https://visual.camp/seeso-sdk/