포팅 메뉴얼

목차

I. 개요

1. 프로젝트 사용 도구

- 이슈 관리: Jira
- 형상 관리: Git lab(SSAFY Git)
- 커뮤니케이션: Notion, Mattermost
- 디자인: Figma
- UCC: Movavi

2. 개발 환경

- Back-End
  - IntelliJ IDEA Ultimate
  - Java: 11 (Zulu 11.0.2)
  - Gradle: 8.2.1
  - Python 3.11.4
  - PIP 23.2.1
  - Spring Boot 2.7.15
  - Apache Spark 3.4.1

- Front-End
  - Android Studio
  - Flutter 3.13.3

- Stroage
  - MySQL 8.0.3
  - Kafka 3.1.2

- Dev-Ops, Infra
  - Docker
  - AWS EC2
  - FCM(Firebase Cloud Messaging)
  - ZooKeeper 3.6.4

II. 빌드

1. 환경 변수

   서버 배포 EC2 URL: J9A505.p.ssafy.io

   DB 전용 EC2 URL: J9A505A.p.ssafy.io

   - Spring Boot Server 공통
     - DB_URL= {DB_URL}
     - DB_USER={DB_USER}
     - DB_PASSWORD={DB Password}
   - Auth Server
     - SECRET_KEY={JWT Key값}
     - REDIS_HOST={Redis 저장 서버 URL}
     - REDIS_MASTER=mymaster
     - REDIS_PORT={Redis Port}
   - Business Server
     - REDIS_HOST={서버 URL}
     - REDIS_MASTER=mymaster
     - REDIS_PORT={Redis Port}

- NotificationProducer Server
  - KAFKA_URL={카프카 설치된 서버 URL}
- NotificationConsumer Server
  - KAFKA_URL={카프카 설치된 서버 URL}

2. 빌드하기
   a. Front: Flutter
      i. apk파일 다운로드 후 실행
   b. Back: Spring boot jar파일 실행
      i. backend/NotificationConsumer/src/main/resources/firebase 폴더에 a505_fcm_sdk.json 추가 필요

```
{
  "type": "service_account",
  "project_id": "{project id}",
  "private_key_id": "{개인 key id}",
  "private_key": "-----BEGIN PRIVATE KEY-----\{private key}-----END PRIVATE KEY-----\n",
  "client_email": "firebase-adminsdk-2nq6t@a5052-59303.iam.gserviceaccount.com",
  "client_id": "{client id}",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-2nq6t%40a5052-59303.iam.gservic
  "universe_domain": "googleapis.com"
}
```

3. 배포하기

- Dockerfile

```
FROM gradle:7.6.1-jdk11 AS builder
WORKDIR /build

# 그래들 파일이 변경되었을 때만 새롭게 의존패키지 다운로드 받게함.
COPY build.gradle settings.gradle /build/
RUN gradle build -x test --parallel --continue > /dev/null 2>&1 || true

# 빌더 이미지에서 애플리케이션 빌드
COPY . /build
RUN gradle build -x test --parallel

# APP
FROM openjdk:11.0-slim
WORKDIR /app

# 빌더 이미지에서 jar 파일만 복사
COPY --from=builder /build/build/libs/*-SNAPSHOT.jar ./app.jar

EXPOSE {port번호}

# root 대신 nobody 권한으로 실행
USER nobody
ENTRYPOINT [                                                 \
   "java",                                                   \
   "-jar",                                                   \
   "-Djava.security.egd=file:/dev/./urandom",                \
   "-Dsun.net.inetaddr.ttl=0",                               \
   "app.jar"              \
]
```

- Git Clone 후
  deploy.sh 파일 실행

```
cd S09P22A505

git pull origin master
```

```
docker stop {auth server container name}
docker stop {business server container name}
docker stop {notification-producer server container name}
docker stop {notification-consumer server container name}

docker rm {auth server container name}
docker rm {business server container name}
docker rm {notification-producer server container name}
docker rm {notification-consumer server container name}

docker rmi -f {auth server container name}
docker rmi -f {business server container name}
docker rmi -f {notification-producer server container name}
docker rmi -f {notification-consumer server container name}

cd backend/Auth
docker build -t {auth server image name} .

cd ../Business
docker build -t {business server image name} .

cd ../NotificationProducer
docker build -t {notification-producer server image name} .

cd ../NotificationConsumer
docker build -t {notification-consumer server image name} .

echo y | docker image prune

# Business
docker run -p {business port}:8080 -d --name {business server container name} --network {network name} -e DB_PASSWORD={DB Password} -e

# Auth
docker run -p {auth port}:8081 -d --name {auth server container name} --network {network name} -e DB_PASSWORD={DB Password} -e DB_URL=

# NotificationProducer
docker run -p {NotificationProducer port}:8082 -d --name {notification-producer server container name} --network {network name} -e DB_

# NotificationConsumer
docker run -p NotificationConsumer port}:8083 -d --name {notification-consumer server container name} --network {network name} -e DB_PA
```

Kafka

```
version: '3.9'

networks:
  backend:
    driver: bridge
    external: true

services:
  pyspark:
    container_name: pyspark
    image: jupyter/pyspark-notebook
    restart: unless-stopped
    volumes:
      - ./spark/pyspark:/home/jovyan/work
      - ./spark/jdbc_drivers:/usr/local/spark/myjars #for jdbc
    environment:
      - NB_GID=100
      - GRANT_SUDO=yes
    user: "root"
    ports:
      - "8888:8888"
    networks:
      - backend
  zookeeper:
    hostname: zookeeper
    image: bitnami/zookeeper:latest
    restart: unless-stopped
    networks:
      - backend
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes
      - ZOOKEEPER_SERVER_ID=1
      - ZOOKEEPER_CLIENT_PORT=2181
```

```
      - ZOOKEEPER_TICK_TIME=2000
      - ZOOKEEPER_INIT_LIMIT=5
      - ZOOKEEPER_SYNC_LIMIT=2
  kafka:
    hostname: kafka
    image: bitnami/kafka:latest
    restart: unless-stopped
    volumes:
      - ./kafka/kafka-persistence:/bitnami/kafka
    networks:
      - backend
    container_name: kafka
    depends_on:
      - zookeeper
    ports:
      - "9092:9092"
      - "9094:9094"
    environment:
      - KAFKA_BROKER_ID=1
      - KAFKA_ADVERTISED_HOST_NAME=j9a505.p.ssafy.io
      - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,EXTERNAL://:9094
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROLLER:PLAINTEXT,EXTERNAL:PLAINTEXT,PLAINTEXT:PLAINTEXT
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://j9a505.p.ssafy.io:9092,EXTERNAL://j9a505.p.ssafy.io:9094
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9092
  kafka-ui:
    image: provectuslabs/kafka-ui:latest
    networks:
      - backend
    container_name: kafka-ui
    restart: unless-stopped
    depends_on:
      - kafka
    ports:
      - "8089:8080"
    environment:
      - DYNAMIC_CONFIG_ENABLED=true
      - KAFKA_CLUSTERS_0_NAME=kafka
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9092
```

SPARK 설치

```
sudo apt install openjdk-17-jdk

# .tgz 파일 다운로드
wget https://dlcdn.apache.org/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.tgz

# .tgz 파일 압축 해제
tar xvf spark-3.4.1-bin-hadoop3.tgz

# 압축 해제한 내용물을 /opt/spark 디렉토리 내부로 옮기기
sudo mv spark-3.4.1-bin-hadoop3.tgz/ /opt/spark

# 환경변수 설정
vim ~/.bashrc
```

```
#추가
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

```
# bash 설정 파일 적용
source ~/.bashrc

# 마스터 노드 실행
start-master.sh --host j9a505.p.ssafy.io --port 7077 --webui-port 8086
# 워커 노드 실행
start-worker.sh spark://j9a505.p.ssafy.io:7077 --memory 10G --cores 4 --port 8001

#같은 과정을 다른 ec2서버에서도 한 뒤 마스터 노드 실행만 하지 않음
```

학습된 모델호출 서버

```
sudo nano /etc/systemd/system/my_flask_app.service
```

```
[Unit]
Description=My Flask App
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/ubuntu/test.py
Restart=always
User=ubuntu

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl start my_flask_app
```

spark 스케줄러 설정

```
crontab -e
```

```
* * * * * /opt/hadoop_spark/bin/spark-submit --jars /home/ubuntu/spark/jdbc_drivers/mysql-connector-j-8.1.0.jar --master spark://j9a505
* 0 * * * /opt/hadoop_spark/bin/spark-submit --jars /home/ubuntu/spark/jdbc_drivers/mysql-connector-j-8.1.0.jar --master spark://j9a505
* 0 * * * /opt/hadoop_spark/bin/spark-submit --jars /home/ubuntu/spark/jdbc_drivers/mysql-connector-j-8.1.0.jar --master spark://j9a505
* 0 * * * /opt/haddop_spark/bin/spark-submit --jars /home/ubuntu/spark/jdbc_drivers/mysql-connector-j-8.1.0.jar --master spark://j9a505
```

III. APK 설치

https://drive.google.com/file/d/1VV5-1rCFSyNwWHQNUzyFc60KhEqraVO7/view?usp=sharing