포팅 메뉴얼!

목차

1. 사용 도구

- 이슈 관리 : JIRA
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost
- 디자인 : Figma
- CI/CD : Jenkins, Docker
- 실행 환경 : Android 14

2. 개발 환경

Frontend (최신화 필요)

name	version
VSCode	1.91.1
React	18.2.0
React Native	0.74.5
TypeScript	5.3.3
Zustand	4.5.5

name	version
Tanstack-Query	5.54.1
Emotion/native	11.11.0
Expo	SDK 51
npm	10.8.2

Backend

name	version
IntelliJ	2024.1.1
Java	17.0.11 2024-04-16 LTS
Spring Boot	3.3.3

Server

name	version
AWS EC2	Ubuntu 20.04.6 LTS
AWS S3	-

Service

name	version
Docker	27.1.1
NginX	1.27.0-alpine
Jenkins	2.476
MySQL	8.0.39
redis	7.2.5-alpine
Kafka	2.13-3.7.1
MariaDB	-

Port

name	port
서비스 접속	443
보너스 지급 서버	8088

name	port
jenkins	8081
mysql	3306
redis	6379
Kafka	9092
zookeeper	2181

Backend 의존 라이브러리

```
  //WEB
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.apache.commons:commons-lang3:3.12.0'

    //JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-impl:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.12.3'

    //DB
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    runtimeOnly 'com.mysql:mysql-connector-j'

    //LOMBOK
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    //TEST
    testImplementation 'org.springframework.boot:spring-boot-
    testImplementation 'org.springframework.security:spring-s
    testRuntimeOnly 'org.junit.platform:junit-platform-launch

    // Google Cloud Vision API 클라이언트 라이브러리
```

```
    implementation 'com.google.cloud:google-cloud-vision:3.34

    //spring cloud starter-aws
    implementation 'org.springframework.cloud:spring-cloud-st

    implementation 'org.springframework.kafka:spring-kafka'
```

3. 외부 서비스

- cloudvision (OCR)

4. 환경 변수 형태

백엔드 환경 변수 목록

- DB 정보
- SSAFY_BANK 키 값
- AWS S3

application.properties [메인 서버]

```
spring.config.import=optional:file:.env[.properties]
spring.application.name=Arbaguette
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://j11c101.p.ssafy.io:3306/ar
spring.datasource.username=${DB_NAME}
spring.datasource.password=${DB_PASSWORD}
spring.jwt.secret=vmfhaltmskdlstkfkdgodyroqkfwkdbalroqkfwkdba
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect
finopenapi.url=https://finopenapi.ssafy.io/ssafy/api
finopenapi.key=${SSAFY_BANK_KEY}
# AWS S3 Configuration
cloud.aws.s3.bucket=arbaguette
cloud.aws.stack.auto=false
```

```
cloud.aws.region.static=ap-northeast-2
cloud.aws.credentials.accessKey=${AWS_ACCESS_KEY}
cloud.aws.credentials.secretKey=${AWS_SECRET_KEY}
#Redis
spring.data.redis.host=j11c101.p.ssafy.io
spring.data.redis.port=6379
#100h
token.access.expired.time=360000000
#1000h
token.refresh.expired.time=3600000000
#Multipart file max size
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
##Kafka
auto.create.topics.enable=true
```

application.properties [보너스 서비스 서버]

```
spring.config.import=optional:file:.env[.properties]
spring.application.name=bonus

spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://stg-yswa-kr-practice-db-
spring.datasource.username=${DB_NAME}
spring.datasource.password=${DB_PASSWORD}

spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect
finopenapi.url=https://finopenapi.ssafy.io/ssafy/api
finopenapi.key=${SSAFY_BANK_KEY}

spring.kafka.bootstrap-servers=13.124.197.79:9092
```

5. 배포하기

Jenkins 설치 및 실행 (Docker in Docker)

1. jenkins-compose.yml 설정

```
version: '3'

services:
  jenkins:
    container_name: jenkins
    image: jenkins/jenkins
    user: root
    restart: unless-stopped
    ports:
      - "8081:8080"
      - "50000:50000"
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/ubuntu/.ssh:/home/.ssh
    environment:
        - TZ=Asia/Seoul
volumes:
  jenkins_home:
```

2. jenkins 실행

```
sudo docker-compose -f jenkins_compose.yml up -d
```

3. jenkins 접속 후 설정

```
apt-get update
apt-get install sudo
sudo apt-get update

sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" |
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

apt-get install docker-ce-cli

y
```

Jenkinsfile 작성

주의! docker 실행 시 cloudvision 관련 설정 필요!

```
pipeline {
    agent any

    stages {
        stage('Build Backend') {
            steps {
                script {
                    dir('backend/Arbaguette') {
                        sh 'chmod +x ./gradlew'

                        sh './gradlew clean build -x test'

                        sh """
sed -i 's/\${AWS_ACCESS_KEY}/${AWS_ACCESS_KEY}/' ./src/main/r
sed -i 's/\${AWS_SECRET_KEY}/${AWS_SECRET_KEY}/' ./src/main/r
sed -i 's/\${DB_NAME}/${DB_NAME}/' ./src/main/resources/appli
sed -i 's/\${DB_PASSWORD}/${DB_PASSWORD}/' ./src/main/resourc
sed -i 's/\${SSAFY_BANK_KEY}/${SSAFY_BANK_KEY}/' ./src/main/r
"""



                        // Check if any container is named "b
def containerNamedBackend = sh(script: "docker ps -a --filter
```

```
if (containerNamedBackend) {
    // Stop and remove the container named "backend"
    sh "docker stop ${containerNamedBackend}"
    sh "docker rm ${containerNamedBackend}"
}


                        // Build and run the new backend cont
                        sh """
docker build \
  --build-arg AWS_ACCESS_KEY=${AWS_ACCESS_KEY} \
  --build-arg AWS_SECRET_KEY=${AWS_SECRET_KEY} \
  --build-arg DB_NAME=${DB_NAME} \
  --build-arg DB_PASSWORD=${DB_PASSWORD} \
  --build-arg SSAFY_BANK_KEY=${SSAFY_BANK_KEY} \
  -t backend .
"""
                        sh """
                            docker run --name backend -d -p 8
-v /home/ubuntu/api_key/cloudvision-434807-1bea29b95286.json:
-e GOOGLE_APPLICATION_CREDENTIALS=${GOOGLE_APPLICATION_CREDEN
-e AWS_ACCESS_KEY=${AWS_ACCESS_KEY} \
-e AWS_SECRET_KEY=${AWS_SECRET_KEY} \
-e DB_NAME=${DB_NAME} \
-e DB_PASSWORD=${DB_PASSWORD} \
-e SSAFY_BANK_KEY=${SSAFY_BANK_KEY} \
-e TZ=Asiz/Seoul \
backend
                        """
                    }

                    dir('backend/bonus') {
                        sh 'chmod +x ./gradlew'

                        sh './gradlew clean build -x test'

                        sh """
```

```
sed -i 's/\${DB_NAME}/${MARIA_NAME}/' ./src/main/resources/ap
sed -i 's/\${DB_PASSWORD}/${MARIA_PASSWORD}/' ./src/main/reso
sed -i 's/\${SSAFY_BANK_KEY}/${SSAFY_BANK_KEY}/' ./src/main/r
"""




                        // Check if any container is named "b
def containerNamedBackend = sh(script: "docker ps -a --filter

if (containerNamedBackend) {
    // Stop and remove the container named "bonus"
    sh "docker stop ${containerNamedBackend}"
    sh "docker rm ${containerNamedBackend}"
}



                        // Build and run the new backend cont
                        sh """
docker build \
  --build-arg DB_NAME=${MARIA_NAME} \
  --build-arg DB_PASSWORD=${MARIA_PASSWORD} \
  --build-arg SSAFY_BANK_KEY=${SSAFY_BANK_KEY} \
  -t bonus .
"""
                        sh """
                            docker run --name bonus -d -p 808
-e DB_NAME=${MARIA_NAME} \
-e DB_PASSWORD=${MARIA_PASSWORD} \
-e SSAFY_BANK_KEY=${SSAFY_BANK_KEY} \
-e TZ=Asiz/Seoul \
bonus
                        """
                    }
                }
            }
        }
    }
```

```
    post {
        success {
            script {
                // 빌드를 실행한 사용자 정보 가져오기
                def user = sh(script: 'git log -1 --pretty=fo

                mattermostSend (color: 'good',
                message: "배포 성공. ${user}",
                )
            }
        }
        failure {
            script {
                // 빌드를 실행한 사용자 정보 가져오기
                // Git 정보를 통해 푸시한 사용자 확인
                def user = sh(script: 'git log -1 --pretty=fo

                mattermostSend (color: 'danger',
                message: "배포 실패. 범인 : ${user}",
                )
            }
        }
    }
}
```

Dockerfile 작성

```
FROM openjdk:17
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Nginx 설정

1. Nginx 설치

```
# nginx 설치
docker pull nginx

# HTTP(80)로 들어오면 HTTPS(443)로 연결
sudo docker run --name nginx -d -p 80:80 -p 443:443 nginx

# 컨테이너 내부로 진입
sudo docker exec -it nginx /bin/bash

# 아래는 Nginx 컨테이너 내부 설치
# Let’s Encrypt 설치
apt-get update

apt-get install vim

apt-get install certbot

apt-get install python3-certbot-nginx
```

2. Nginx 설정

```
docker exec -it nginx(컨테이너이름) /bin/bash

cd etc/nginx/conf.d

vi 아무거나.conf
```

2-1. .conf 작성

```
server {
    server_name 도메인 주소
}
```

2-2. cert 설정

```
certbot --nginx -d <도메인1>
```

2-3. .conf 파일 최종 작성

```
server {
        server_name j11c101.p.ssafy.io;


    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j11c101.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/j11c101.p.ssafy
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed


  location / {
        proxy_pass http://j11c101.p.ssafy.io:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }

}
server {
    if ($host = j11c101.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        server_name j11c101.p.ssafy.io;
    listen 80;
    return 404; # managed by Certbot


}
```

Redis 설치

```
docker pull redis
docker run -p 6379:6379 -d redis
```

Kafka 설치

오류로 인해 직접 설치

- 카프카 파일을 바이너리로 다운로드
- 바이너리 파일 압축 해제
- 설치폴더에 config폴더를 들어가면 server.properties에서 advertised.listener 주석 해제 후 URL {도메인:9092} 변경
- zookeeper 실행
- kafka broker 실행
- 방화벽 port 해제

6. 빌드 및 실행

- Jenkinsfile 및 Dockerfile 확인
- 빌드 시 cloudvision json 파일 설정 필요