D108 : Favor Eat

삼성청년 SW 아카데미 구미캠퍼스 8 기

공통프로젝트[2023.02.20. – 2023.04.07.]

포팅 매뉴얼

담당 컨설턴트 : 오형남

오민준, 장예은, 김호균, 이효진, 박재희, 이창민

1. 프로젝트 작업 환경

1.1 이슈관리 : JIRA

1.2 형상관리 : GITLAB

1.3 커뮤니케이션 : MATTERMOST, NOTION, FIGMA

1.4 개발환경

1.4.1 OS : WINDOWS10

1.4.2 IDE

- INTELLIJ 2022.3.1(ULTIMATE EDITION)
- VISUAL STUDIO CODE

1.4.3 DATABASE : MYSQL 5.7.39

1.4.4 OS : WINDOWS10

1.5 배포관리 : NGINX, JENKINS, DOCKER

2. 빌드 상세내용

● 코드 병합 과정

1. 프로젝트 파일의 작업 내용을 Git 명령어를 통해 원격 저장소에 병합 요청합니다.

2. 작업 내용을 각 파트 (Front-End, Back-End)의 팀장이 리뷰하고 컨벤션에 적합하지 않다면 피드백을 작성합니다.

3. 각 파트원은 피드백을 반영하여 코드를 수정한 후 병합을 재요청합니다.

4. 코드가 병합된다면 GItLab 의 WebHook 을 통해서 오픈소스 CI 툴인 Jenkins 를 통해서 빌드하고 Docker.sock 을 통해 EC2 컨테이너로 배포합니다.

● CI/CD 과정

1. Jenkins 는 싸피에서 제공하는 Amazon EC2 에 설치하였습니다
2. EC2 에 Docker 를 설치합니다.
3. Docker Hub 에서 Jenkins 공식 이미지를 설치합니다.
4. Jenkins 이미지를 컨테이너화하고 회원 설정과 플러그인을 설치합니다.
5. WebHook 연결을 위해 GitLab 관련 플러그인도 설치합니다.
6. BE_Prod, Test, FE_Prod, Test 라는 이름으로 Item 을 생성합니다.
7. Item 의 환경설정을 통해 GitLab WebHook 과 연결합니다.
   Jenkins 의 Build Steps 을 Execute shell 을 통해 설정합니다.
8. Jenkins 설정이 모두 완료되었다면 Git 원격 저장소에 브랜치를 병합하게 되면 도커 소켓을 통해 EC2 Ubuntu 에 있는 Docker 에 프로젝트 빌드 파일이 컨테이너화되어 Delivery 됩니다.

- 리버스 프록시 과정

1. 저희는 Next.js 를 사용하여 SSR 을 사용하기 때문에 웹 서버를 따로 설치하지 않았습니다.
2. 따라서, 리버스 프록시와 로드 밸런스만을 위한 별도의 서버가 필요했고 간편한 환경설정과 SSL 처리를 위해서 NGINX 를 사용하였습니다.
3. NGINX 를 on-promise 방식으로 EC2 에 설치합니다. 컨테이너 방식을 사용해서 얻는 이점보다 도커 설정하는 과정의 복잡성 비용이 크다고 생각하기 때문입니다.
4. Certbot 을 활용하여 NGINX 에 손쉽게 SSL 인증서를 설치할 수 있었습니다.
5. Certbot 을 통해서 letsencrypt 인증서로 HTTPS 를 처리합니다.
6. nginx.conf 를 확인하면 인증서 처리가 되어있는 것을 알 수 있습니다.
7. 443 listen 을 하면 각 컨테이너의 포트번호를 리버스 프록시 해줍니다.

3. 배포 특이사항

- Jenkins Build Steps (Execute shell)

```
docker build -t /*Item name*/ ./ /*Item name*/ /FavorEat

if (docker ps | grep "/*Item name*/ "); then docker stop /*Item name*/; fi

docker run -it -d --rm -p 8000:8080 --name /*Item name*/ /*Item name*/

echo "Run /*Item name*/ Project"
```

- Spring Boot Dockerfile

```
FROM adoptopenjdk/openjdk11 AS builder
VOLUME /tmp
COPY gradlew .
COPY gradle gradleCOPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

- Next.js Dockerfile

```
FROM node:18.15.0

WORKDIR /var/jenkins_home/workspace/Frontend/frontend
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000
CMD ["npm", "run", "start"]
```

- Next.js Dockerfile

```
FROM python:3.9.13

WORKDIR /var/jenkins_home/workspace/Bigdata/bigdata

COPY . .
RUN pip install --no-cache-dir --upgrade -r ./requirements.txt
EXPOSE 6000

CMD ["uvicorn", "bigdata.main:app", "--host", "0.0.0.0", "--port", "6000"]
```