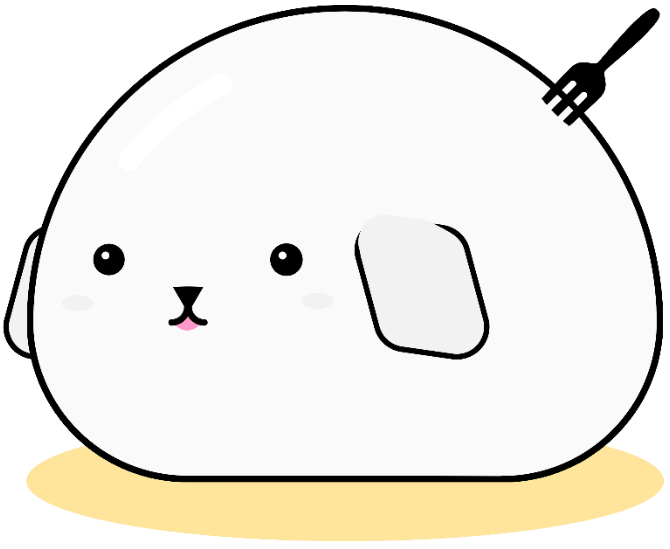
빌드 및 배포 가이드



개떡찰떡

삼성 청년 SW 아카데미 7기 대전 특화 1반 B301

진행 기간 : 2022. 08. 22 ~ 2022. 10. 07

고승효 김애리 노정현 박준혁 장종환 홍성덕

1. 기술 스택 및 버전

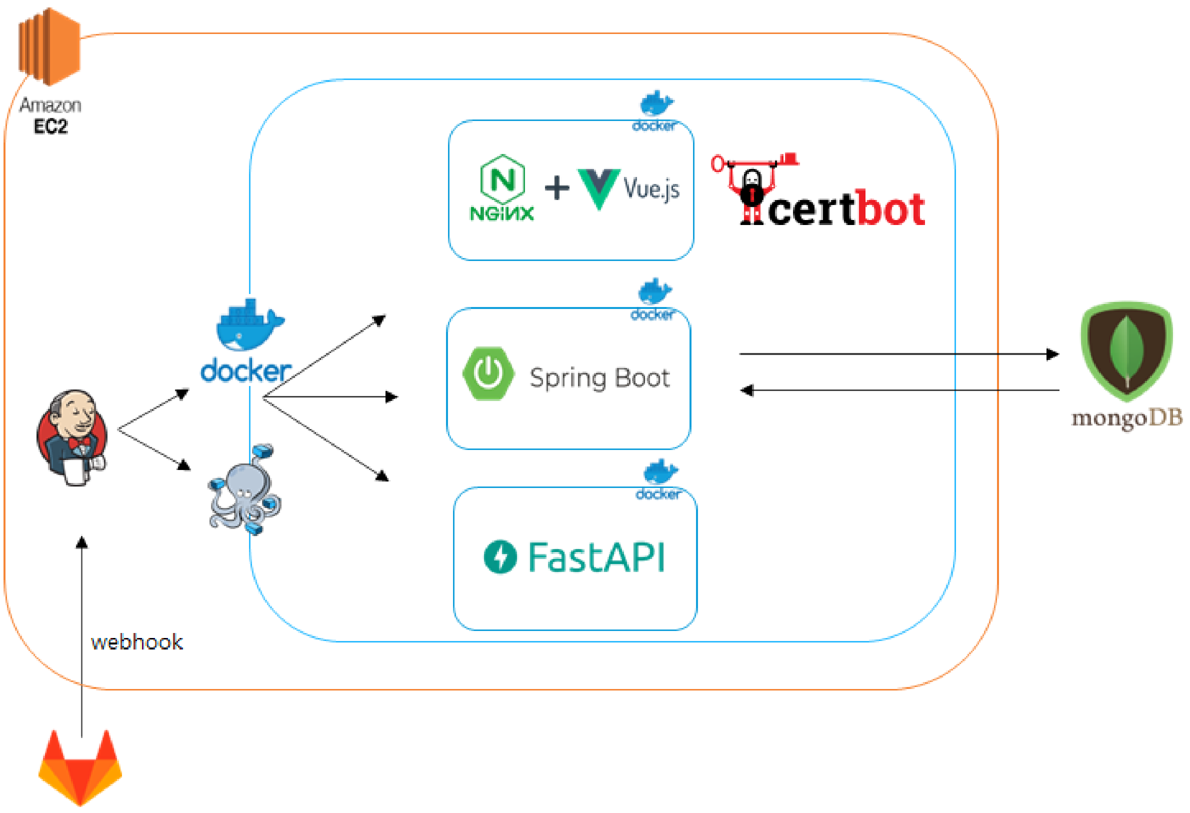
- 기술 스택 상세 내용

구분	사용 목적	사용 기술	기술 스택	버전
협업	형상 관리	GitLab		
	이슈 관리	Jira		
	커뮤니케이션	Mattermost		
		Notion		
		Webex		
		Google Sheets		
Server	배포	OS	Ubuntu	20.04.1 LTS
		배포	Docker	20.10.18
			Docker Compose	1.29.0
		CI/CD	Jenkins	LTS
		웹 서버	Nginx	1.15-alpine
Front-End	개발	JavaScript	Node.js	8.13.2
		Framework	Vue3.js	3.2.13
			Vue-Router	4.0.3
Back-End	개발	DBMS	MongoDB	5.0.13 Enterprise
		DB API	Mongo Repository	2.7.4
		Java	Zulu	1.8
		Framework	Spring Boot	2.7.4
		Security	Spring Security	2.7.4
			JWT	0.9.1
		WAS	Tomcat	9.0.65
		Build	Gradle	7.5
Core	개발	Language	Python	3.8.13
		Deep Learning	PyTorch	1.12.1
			Jupyter Lab	3.4.5
		Image Processing	OpenCV	4.6.0.66
			Numpy	1.23.3
			Scipy	1.9.1
		배포	FastAPI	0.84.0
			Uvicorn	0.18.3
			pydantic	1.10.2

2. 상세 내용

1. 배포 흐름

배포 환경 및 배포 흐름은 다음과 같습니다.



각 프로젝트들을 Build하여 Image로 변환합니다. 이후 Docker Compose를 활용하여 해당 Image들을 다중 컨테이너 앱으로 구성하여 AWS EC2 인스턴스에 배포합니다. 배포된 앱은 Nginx의 Proxy Server를 통해 접근이 가능합니다.

2. 포트 번호

No.	포트 번호	이름
1	22	SSH
2	80	HTTP
3	443	HTTPS
4	8081	Spring Boot Container
5	8082	AI Server Container

3. 환경 설정

(1) Nginx : app.conf

```
server {
    listen 80;
    listen [::]:80;
    server_name j7b301.p.ssafy.io;
    server_tokens off;

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

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name j7b301.p.ssafy.io;
    server_tokens off;
    client_max_body_size 5M;

    ssl_certificate /etc/letsencrypt/live/j7b301.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j7b301.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # Proxy
    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_set_header X-Forwarded-Proto https;
    proxy_headers_hash_bucket_size 512;
    proxy_redirect off;

    # Websockets
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

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

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
    location /api/ {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_pass http://spring:8081/api/;

        proxy_connect_timeout      150;
        proxy_send_timeout         100;
        proxy_read_timeout         100;

        proxy_buffer_size          8k;
        proxy_buffers              4 32k;
        proxy_busy_buffers_size    64k;
        proxy_temp_file_write_size 64k;
    }

    location /core/ {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_pass http://core:8082/;

        proxy_connect_timeout      150;
        proxy_send_timeout         100;
        proxy_read_timeout         100;

        proxy_buffer_size          8k;
        proxy_buffers              4 32k;
        proxy_busy_buffers_size    64k;
        proxy_temp_file_write_size 64k;
    }

}
```

(2) Frontend : package.json

```
{
  "name": "front",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "@popperjs/core": "^2.11.6",
    "axios": "^0.27.2",
    "bootstrap": "^5.2.1",
    "bootstrap-icons": "^1.9.1",
    "bootstrap-vue-3": "^0.3.3",
    "core-js": "^3.8.3",
    "phaser": "^3.55.2",
    "pinia": "^2.0.22",
    "register-service-worker": "^1.7.2",
    "sweetalert2": "^11.4.33",
    "vue": "^3.2.13",
    "vue-router": "^4.0.3",
    "vue3-google-login": "^2.0.12",
    "vue3-popper": "^1.5.0"
  },
  "devDependencies": {
    "@babel/core": "^7.12.16",
    "@babel/eslint-parser": "^7.12.16",
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-eslint": "~5.0.0",
    "@vue/cli-plugin-pwa": "~5.0.0",
    "@vue/cli-plugin-router": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "eslint": "^7.32.0",
    "eslint-config-prettier": "^8.3.0",
    "eslint-plugin-prettier": "^4.0.0",
    "eslint-plugin-vue": "^8.0.3",
    "prettier": "^2.4.1"
  }
}
```

(3) Back-end : src/main/java/resources/application.properties

```
# database
spring.data.mongodb.uri=mongodb+srv://S07P22B301:zFoNN24jV5@ssafy.ngivl.mongod
b.net/S07P22B301?authSource=admin
spring.data.mongodb.database=S07P22B301

# port number
server.port=8081

# url
server.servlet.context-path=/api

# multipart setting
```

```
spring.servlet.multipart.maxFileSize=1000MB
spring.servlet.multipart.maxRequestSize=1000MB
```

3. 도커 파일

1. Front-End

```
FROM node:lts-alpine as build-stage

WORKDIR /app

COPY . .

RUN npm install --save --legacy-peer-deps

RUN npm run build

FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

2. Back-End

```
# Build Stage
FROM openjdk:8-jdk-alpine as build-stage

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

# Deploy Stage
FROM openjdk:8-jdk-alpine

WORKDIR /app
COPY --from=build-stage build/libs/*.jar app.jar
```

```
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/app/app.jar" ]
```

3. AI Server

```
FROM python:3.8.13

WORKDIR /app

COPY requirements.txt .

RUN apt-get update && apt-get upgrade -y
RUN apt-get install libgl1-mesa-glx -y

RUN pip3 install torch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 --extra-index-url https://download.pytorch.org/whl/cpu
# RUN pip install pytorch_lightning kornia omegaconf
RUN pip install --no-cache-dir --upgrade -r requirements.txt

COPY app .
EXPOSE 8082
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8082"]
```

4. Docker Compose

```
version: "3.8"

services:
  spring:
    build: back
    container_name: spring
    restart: always
    expose:
      - "8081"
    volumes:
      - ./data:/app/data
    networks:
      - my_network
  core:
    build: core
    container_name: core
    restart: always
    volumes:
      - "./data:/app/data"
    expose:
      - "8082"
```

```yaml
    networks:
      - my_network
    # entrypoint: "gunicorn -k uvicorn.workers.UvicornWorker main:app --bind
0.0.0.0:8082 --workers 1"
  nginx:
    build: front
    container_name: nginx
    restart: unless-stopped
    volumes:
      - ./front/deploy/nginx:/etc/nginx/conf.d
      - ./front/deploy/certbot/conf:/etc/letsencrypt
      - ./front/deploy/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - core
    networks:
      - my_network
    command: '/bin/sh -c ''while :; do sleep 6h & wait $${!}; nginx -s reload;
done & nginx -g "daemon off;"'''
  certbot:
    image: certbot/certbot
    container_name: certbot
    restart: unless-stopped
    volumes:
      - ./deploy/certbot/conf:/etc/letsencrypt
      - ./deploy/certbot/www:/var/www/certbot
    networks:
      - my_network
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep
12h & wait $${!}; done;'"

networks:
  my_network:
    driver: bridge
```

4. 배포 과정

배포 환경 : AWS EC2

0. EC2 설정

- Ubuntu 업데이트

```
sudo apt-get update && sudo apt-get upgrade -y
```

- Timezone 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

1. SSL 인증서 발급 및 Nginx 설정

- init-letsencrypt.sh 파일에 실행 권한 부여

```
   sudo chmod +x init-letsencrypt.sh
```

- init-letsencrypt.sh 파일을 실행하여 SSL 인증서 발급

```
   ./init-letsencrypt.sh
```

2. MongoDB

- SSAFY에서 제공한 외부 서버 이용

3. 실행

- docker-compose 설치

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

- docker-compose를 활용하여 실행

```
docker-compose up -d
```

5. 주요 속성 정보

1. MongoDB 계정 정보

- Database URL

mongodb+srv://S07P22B301:zFoNN24jV5@ssafy.ngivl.mongodb.net/S07P22B301?authSource=admin

- Username : S07P22B301

- Password : zFoNN24jV5

2. Spring Boot Database 설정

- application.properties

```
spring.data.mongodb.uri=mongodb+srv://S07P22B301:zFoNN24jV5@ssafy.ngivl.mongod
b.net/S07P22B301?authSource=admin
spring.data.mongodb.database=S07P22B301
```

- build.gradle

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'
}
```