빌드 및 배포 가이드

FLODY

삼성 SW 청년 아카데미 7기 대전 공통 1반 B101

진행 기간 : 2022.07.11 ~ 2022.08.19(6주)

장종환 김애리 정교준 정세한 최재현

1.기술 스택 및 버전

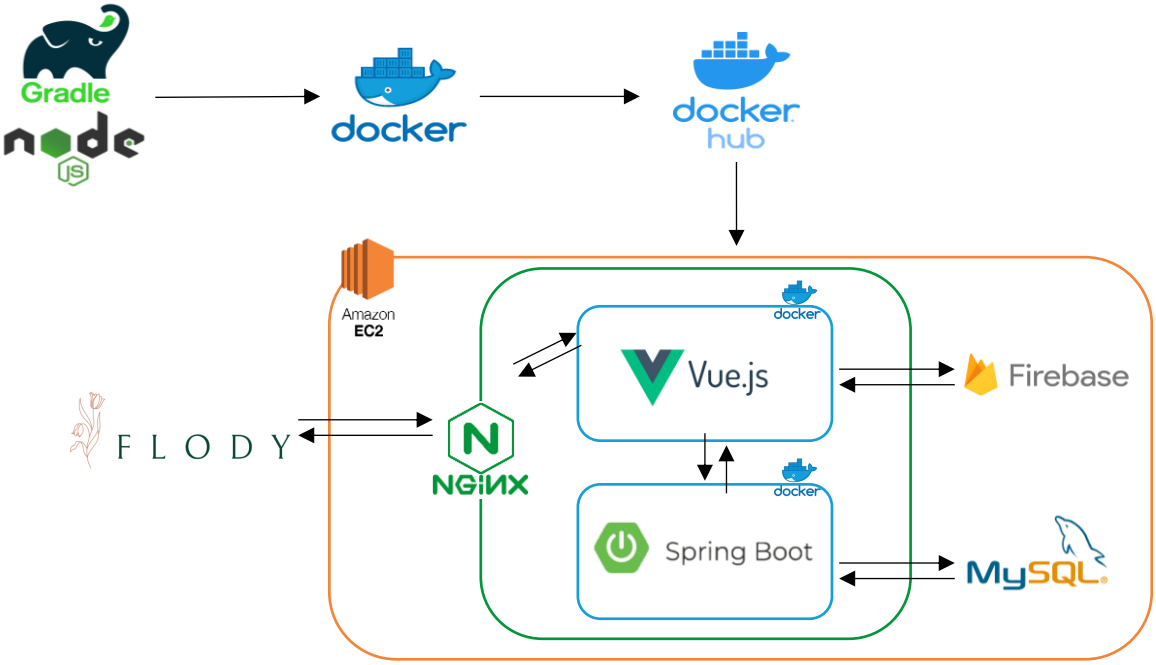
- 기술 스택 상세 내용

구분	사용 목적	사용 기술	기술 스택	버전
협업	형상 관리	GitLab		
	이슈 관리	JIRA		
	커뮤니케이션	Mattermost		
		Notion		
		Zoom-clova		
		Webex		
		Google Sheets		
BackEnd	개발	DBMS	MySQL	8.0.28
		DB API	JPA	2.7.2
		Java	Zulu	1.8
		Framework	Spring Boot	2.7.2
		JWT		0.9.1
		WAS	Tomcat	2.7.2
		Build	Gradle	7.5
front	개발	Javascript	Node.js	16.16.0 LTS
		Framework	Vue	3.2.13
			vuex	4.0.2
			Vue-router	4.0.13
			Bootstrap-vue-3	0.1.21
		DBMS	firebase	9.9.1
Server	배포	OS	Ubuntu	20.04 LTC
		배포	Docker	20.10.17
		서버	AWS EC2	5.4.0
			Nginx	1.18.0

1.상세 내용

1. 배포 흐름

배포 환경 및 배포 흐름은 다음과 같습니다.



각 프로젝트들을 build하여 image로 변환합니다. 이후 hub를 통해 로컬과 서버간 image 파일들을 연동합니다. 배포 서버에서 docker hub에서 pull 받은 이미지를 컨테이너로 실행합니다. 해당 컨테이너는 Nginx의 proxy server로 설정됩니다.

2. 포트 번호

Port Num

No.	포트 번호	이름
1	22	SSH
2	443	HTTPS
3	3306	MySQL
4	8081	Spring Boot Docker Container
5	3000	Vue Docker Container

3. 환경설정

(1) FrontEnd :package.json

```
{
  "name": "flody-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": "^2.11.5",
    "axios": "^0.27.2",
    "bad-words": "^3.0.4",
    "bootstrap": "^5.2.0",
    "bootstrap-vue": "^2.22.0",
    "bootstrap-vue-3": "^0.1.21",
    "core-js": "^3.8.3",
    "firebase": "^9.9.1",
    "firebaseui": "^6.0.1",
    "sweetalert2": "^11.4.27",
    "swiper": "^8.3.2",
    "v-calendar": "^3.0.0-alpha.8",
    "vue": "^3.2.13",
```

```
    "vue-infinite-slide-bar": "^1.1.1",
    "vue-jwt-decode": "^0.1.0",
    "vue-router": "^4.0.13",
    "vue-sweetalert2": "^5.0.5",
    "vuex": "^4.0.2",
    "vuex-persistedstate": "^4.1.0"
  },
  "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-router": "~5.0.0",
    "@vue/cli-plugin-vuex": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "eslint": "^7.32.0",
    "eslint-plugin-vue": "^8.0.3"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/vue3-essential",
      "eslint:recommended"
```

```
    "extends": [
      "plugin:vue/vue3-essential",
      "eslint:recommended"
    ],
    "parserOptions": {
      "parser": "@babel/eslint-parser"
    },
    "rules": {}
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead",
    "not ie 11"
  ]
}
```

```
yarn add v-calendar@next
```

(2) BackEnd : src/main/java/resources/application.properties

```
# Database(MySQL) 통신

spring.datasource.url=jdbc:mysql://i7B101.p.ssafy.io:3306/flody?serverTimezone=Asia/Seou
l

spring.datasource.username=flody

spring.datasource.password=flodydbpass

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver


# JPA 설정

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.properties.hibernate.format_sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true


# Port Num

server.port=8081

server.address = 0.0.0.0
```

3. 배포 과정

배포 환경 : Amazon EC2

1. MySql

-1) Ubuntu 업데이트

```
Sudo apt-get update
```

-2) MySQL 설치

```
Sudo apt install mysql-server
```

-3) MySQL 접속

```
Sudo mysql -u root -p
```

-4) DataBase 변경 및 계정생성, 권한부여

```
Use mysql;
create user 'flody'@'%' identified by 'flodydbpass';
grant all privileges on *.* to 'flody'@'%';
flush privileges;
```

(6) MySQL 접속 해제

```
Exit
```

(7) MySQL 접속 권한 수정 : 경로 이동

```
cd /etc/mysql/mysql.conf.d/
```

(8) MySQL 접속 권한 수정 : 접속

```
sudo nano mysqld.cnf
```

(9) MySQL 접속 권한 수정 : 파일 수정

```
bind-address = 0.0.0.0
```

(10) MySQL을 재시작하여 변경 내용 적용

```
sudo service mysql restart
```

(11) MySQL Workbench에서 접속 확인

```
Hostname : i7b101.p.ssafy.io

Port : 3306

Username : flody

Pasword : flodydbpass
```

2. FrontEnd(Vue3)

(1) [Local] Dockerfile 생성

```
# build stage

FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./
COPY yarn.lock ./
RUN npm install
RUN npm install yarn
COPY . .
RUN npm run build

# production stage
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) [Local] Docker image 생성

```
docker build -t [Docker Hub ID]/[Repository Name]:[Image Tag] .
```

(3) [Local] 생성한 이미지 Dockerhub에 push

```
docker push [Docker Hub ID]/[Repository Name]:[Image Tag]
```

(4) [Server] 도커 로그인

```
sudo docker login
```

(5) [Server] Docker Hub에서 image pull

```
sudo docker pull [Docker Hub ID]/[Repository Name]:[Image Tag]
```

(6) Docker Image Container 실행

```
sudo docker run --name [사용자가 지정한 이름] -d -p [외부 포트]:[expose 포트] [Docker Hub ID]/[Repository Name]:[Image Tag]
```

※수동 배포로, 진행 시 [Server]에서 실행중인 Container를 중지 및 삭제, 해당 서버에 존재하는 Image 삭제 후 (2)-(6)까지의 과정을 반복함.

3) BackEnd(Spring Boot)

(1) [Local] Dockerfile 생성

```
FROM openjdk:8-jdk-alpine
EXPOSE 8081
ARG JAR_FILE=flody-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} back.jar
ENTRYPOINT ["java","-jar","/back.jar"]
```

(2)[Local] 빌드 파일 생성

```
InteliJ > Gradle > Bootjar 실행
```

(3) [Local] Docker image 생성

```
docker build -t [Docker Hub ID]/[Repository Name]:[Image Tag] .
```

(4) [Local] 생성한 이미지 Dockerhub에 push

```
docker push [Docker Hub ID]/[Repository Name]:[Image Tag]
```

(5) [Server] 도커 로그인

```
sudo docker login
```

(6) [Server] Docker Hub에서 image pull

```
sudo docker pull [Docker Hub ID]/[Repository Name]:[Image Tag]
```

(7) Docker Image Container 실행

```
sudo docker run --name [사용자가 지정한 이름] -d -p [외부 포트]:[expose 포트] [Docker Hub ID]/[Repository Name]:[Image Tag]
```

※수동 배포로, 진행 시 [Server]에서 실행중인 Container를 중지 및 삭제, 해당 서버에 존재하는 Image 삭제 후 (2)-(7)까지의 과정을 반복함.

4) ssl 인증서 발급 및 nginx 리버스 프록시 설정

아래 모든 과정은 Amazon EC2에서 진행됩니다.

(0) Ubuntu 업데이트

```
sudo apt-get update sudo apt-get upgrade
```

(1) nignx 설치

```
sudo apt-get install nginx
```

(2) nginx 중단

```
sudo systemctl stop nginx
```

(3) letsencrypt 설치

```
sudo apt-get install letsencrypt
```

(4) ssl 인증서 발급 : 정상적으로 발급되었을 경우

```
/etc/letsencrypt/live/{도메인 네임}안에 인증서 파일이 있음
sudo letsencrypt certonly --standalone -d i7b101.p.ssafy.io
```

(5) 서버 블록 파일 생성(리버스 프록시 설정) : 경로 이동

```
cd /etc/nginx/sites-available
```

(6) 서버 블록 파일 생성(리버스 프록시 설정) : 접속

```
sudo vi /etc/nginx/sites-available/XXX.conf
```

(7) 서버 블록 파일 생성(리버스 프록시 설정) : 파일 수정

```
server {

        server_name i7b101.p.ssafy.io;
        location /{ proxy_pass http://i7b101.p.ssafy.io:3000;

}
location /api {
        proxy_pass http://i7b101.p.ssafy.io:8081/api;
}
listen 443 ssl;
ssl_certificate /etc/letsencrypt/live/i7b101.p.ssafy.io/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/i7b1011.p.ssafy.io/privkey.pem;
}
server {
        if ($host = i7b101.p.ssafy.io) {
            return 301 https://$host$request_uri;
        }
              listen 80 default_server;
              server_name i7b101.p.ssafy.io;
          return 404; # managed by Certbot
}
```

(8) 각 서버 블록에 대한 심볼릭 링크 생성

```
sudo ln -s /etc/nginx/sites-available/xxx.conf /etc/nginx/sites-enabled/xxx.conf
```

(9) nginx 설정 체크

```
sudo nginx -t
```

(10) nginx 재시작

```
sudo systemctl restart nginx
```

5) SSL 인증서 적용

(1) frontend – 포트 변경

```
  https://i7b101.p.ssafy.io
```

4. 주요 속성 정보

1) MySQL 계정 정보

```
Database : flody
username : flody
password : flodydbpass
```

2) Spring Boot Database 설정

```
 # Database

spring.datasource.url=jdbc:mysql://i7B101.p.ssafy.io:3306/flody?serverTimezone=Asia/Seoul
spring.datasource.username=flody
spring.datasource.password=flodydbpass
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```