포팅 메뉴얼

기술 스택 및 버전

프로젝트기술스택

Aa 기술	역할	버전
SpringBoot	BE	2.7.7
QueryDsl	BE	1.0.10
MYSQL	BE	8.0.31
JDK	BE	zulu 11.0.17
react	FE	18.2.0
chakra ui	FE	2.5.0
Docker	Infra	**20.10.22**
docker-compose	Infra	2.16.0
jenkins	Infra	2.390
ngnix	Infra	1.18.0
UBUNTU	Infra	20.04

EC2 세팅

Nginx & Letsencrypt 설치

- Nginx는 웹 서버 프로그램으로 HTTP로서의 역할 및 리버스 프록시 / 로드 밸런서 역할을 수행한다.
- Letsencrypt는 Http > Https가 되기 위해서는 SSL 인증서가 필요한데 기본적으로 SSL은 매우 비싸다. 그렇기에 Letsencrypt라는 무료 SSL 인증서를 통하여 보안을 강화한다.

```
sudo apt-get update
sudo apt-get install nginx
sudo nginx -v
sudo systemctl stop nginx

# SSL 인증
sudo apt-get install letsencrypt
sudo letsencrypt certonly --stadalone -d i8a806.p.ssafy.io
cd /etc/letsencrypt/live/i8a806.p.ssafy.io # root 계정으로..
cd /etx/nginx/sites-available
sudo vim nginx.conf

    location / {
            proxy_pass http://localhost:3000;

            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $host;
            proxy_set_header Origin "";

    }

    location /api {
            #rewrite ^/api(.*)$ $1?$args break;
            proxy_pass http://localhost:8080;
    }

    location /ws {
            proxy_pass http://localhost:8080;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $host;
            proxy_set_header Origin "";
    }

   location /api/ws-stomp {
            proxy_pass http://localhost:8080;
            #rewrite /api/(.*) /$1 break;

            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
            proxy_set_header Host $host;
    }

listen 443 ssl; # managed by Certbot
# 도메인 이름을 써줘야함
ssl_certificate /etc/letsencrypt/live/i8a806.p.ssafy.io/fullchain.pem; # managed by Certbot
# 도메인 이름을 써줘야함
ssl_certificate_key /etc/letsencrypt/live/i8a806.p.ssafy.io/privkey.pem; # managed by Certbot
# include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
# ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {

# 도메인 이름을 입력
if ($host = i8a806.p.ssafy.io) {
    return 301 https://$host$request_uri;
} # managed by Certbot

    client_max_body_size 10M;


# 파일 연동 및 테스트
sudo ln -s /etc/nginx/sites-available/nginx.conf/etc/nginx/sites-enabled/nginx.conf
```

```
sudo nginx -t

# Nginx 재시작
sudo systemctl restart nginx

#Nginx 상태 확인
sudo systemctl status nginx
```

Docker 설치

1. 패키지 업데이트 진행

```
sudo apt-get update
```

2. 필요 패키지 설치

```
sudo apt-get install \
ca-certificates \
curl \
gnupg \
lsb-release
```

3. Docker의 Official GPG Key 를 등록

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. 도커 엔진 설치

```
# 다시 업데이트
sudo apt-get update

# 도커 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. 도커 버전 확인

```
docker --version
```

redis 설치

1 레디스 설치

```
$ docker pull redis
```

2.설치된 이미지 확인

```
$ docker images
```

3. docker network 구성

```
$ docker network create redis-net
```

4. 컨테이너 실행

```
$ docker run --name dingrr -p 6379:6379 --network redis-net -d redis redis-server --appendonly yes
```

5. 레디스 서버 접속

```
$ docker run -it --network redis-net --rm redis redis-cli -h dingrr
```

MySQL 환경 셋팅

root 계정 패스워드 변경

- EC2에서 MySQL에 접속하기

```
sudo docker exec -it mysql /bin/bash
```

- bash에 접속하면 mysql 접속 명령어 입력

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

- root 비밀 번호 변경

```
ALTER user 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '변경하고 싶은 패스워드';
flush privileges;
```

대소문자 구분 해결하기

- 기본적으로 대소문자를 구분하도록 데이터베이스에 설정이 되어있었다
  - 0 : 대소문자 구분
  - 1 : 대소문자 구분 X

```
mysql> show global variables like '%lower_case%';
+------------------------+-------+
| Variable_name | Value |
+------------------------+-------+
| lower_case_table_names | 0 |
+------------------------+-------+
```

- lower_case_table_names 를 바꾸기 위해서는 my.cnf 에서 아래의 한 줄을 추가하면 된다.

```
#my.cnf
[mysqld]
lower_case_table_names=1
```

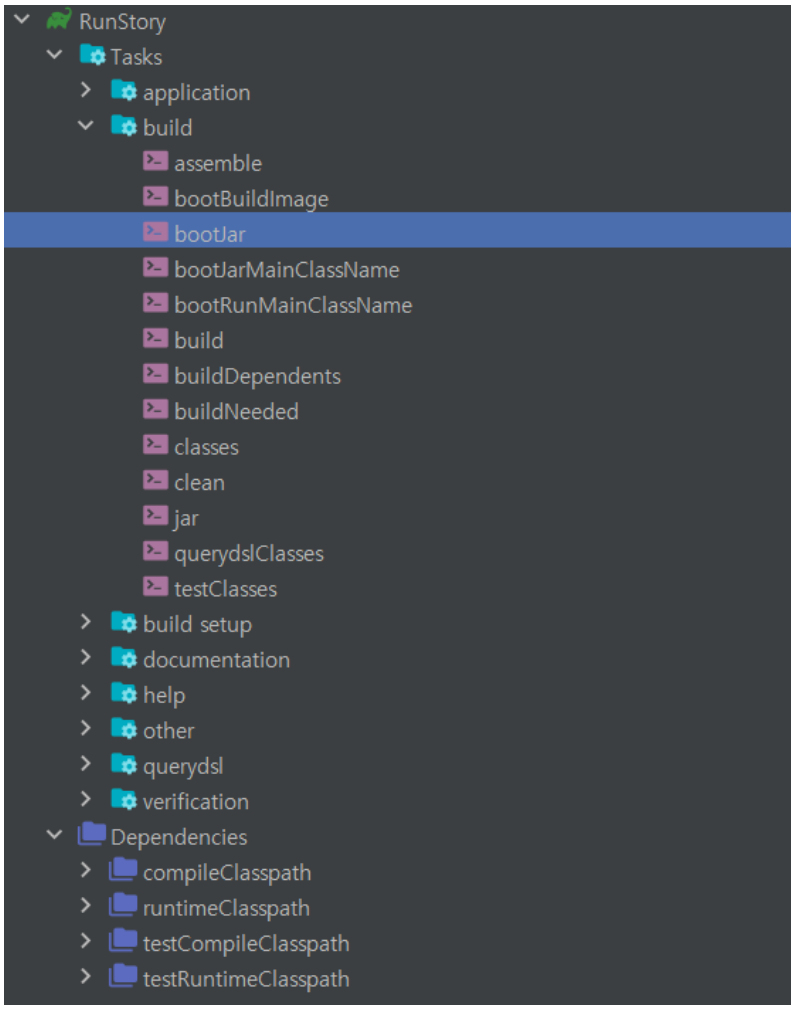
백엔드 빌드

배포용 설정

- EC2 MYSQL 연결 (Applications.properties)
- Application.properties에서 MySQL에 맞는 ip 및 port 연결
- Redis와 Socket 연결
- EC2에 FrontEnd / Backend / Mysql / Jenkins / Redis Container 미리 생성
- Frontend / Backend의 경우 개별 Dockerfile을 build 시켜 Container 생성

jar 파일로 빌드

- CMD > Server 최상단 폴더 > ./gredlew build (or gredlew build)
- 그럼 프로젝트 폴더에 build 라는 폴더가 생기고 build/libs/ 에 jar 파일이 생긴다.



도커 파일로 빌드 (프론트엔드 / 백엔드)

- Backend의 경우에는 Image를 빌드할 곳이 필요한데 보통 jar파일을 사용하므로 위와 같이 Build시켜서 jar파일을 생성한 후에 Build를 시작한다.
- 보통 jar 파일이 2개가 나오는데 1개를 고정으로 사용하는 것이 좋다.

```
FROM openjdk:11-jdk
#ARG JAR_FILE="build/libs/simple-*.jar"
```

```
COPY build/libs/RunStory-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar","/app.jar"]
```

- 프론트엔드의 경우 기본적으로 복사하여 사용하기 때문에 따로 빌드를 할 필요가 없다.
- Docker File에서 특별하게 설치가 필요한 경우에는 추가적으로 다운로드를 받아야 한다.

```
FROM nginx:stable-alpine
FROM node:18-alpine
WORKDIR /runtogether/src/app
COPY package*.json ./ # 해당 폴더에 있는 Library 혹은 의존성을 가져온다.
COPY package*.json ../


# 필요한 라이브러리를 받는 역할
RUN npm install
RUN npm i @fortawesome/free-solid-svg-icons
RUN npm i @fortawesome/free-regular-svg-icons
RUN npm i react-infinite-scroll-component
RUN npm i @fortawesome/free-brands-svg-icons
RUN npm i @fortawesome/react-fontawesome
RUN npm install --save html2canvas
RUN npm install react-slick --save
RUN npm install slick-carousel --save
RUN npm install axios

COPY ./ ./ # 해당 폴더에 포갓한다.
CMD [ "npm", "run","start" ] # 시작한다.
```

백엔드 배포

- Dockerfile이 있는 폴더 (보통 Server 폴더의 최상단)으로 간 후에 Dockerfile을 Build 시켜서 Image를 생성하고 Container를 생성 run한다.
- Jenkins를 사용할 경우에는 백엔드에서 배포가 된지만 보통 GitLab에 Application.properties 나 Email.properties 같은 보안이 중요한 것은 올리지 않는다. 그렇기 때문에 자동배포를 할 경우 Application.properties 등이 없는 에러가 발생하게 되는데 이런 경우를 대비하여 미리 Application.properties를 수동으로 AWS 안에 집어넣고 돌림으로서 자동 배포가 가능하다.
- AWS 서버 내의 로컬 파일과 AWS 도커 서버 컨테이너 간의 연결이 필요하므로 volume 설정을 통하여 연결한다.

```
Backend Container 생성
# 앞에 sudo를 쓰거나 sudo su 사용 후에 진행
docker build -t runstory-server ./
docker run -d --name runstory-server -v /home/ubuntu/images:/var/lib/runstory -p 8080:8080 runstory-server

# 만약 필요 시 docker container에 들어가야 한다면.. (Volume 확인 등등...)
sudo docker exec -it runstory-server

Jenkins Container 생성
  sudo docker run -d --name jenkins -u root --privileged \
  -p '9090:8080' \
  -v '/home/ubuntu/docker-volume/jenkins:/var/jenkins_home' \
  -v '/var/run/docker.sock:/var/run/docker.sock' \
  -v '/usr/bin/docker:/usr/bin/docker' \
  jenkins/jenkins

Jenkins Docker 내부로 들어가기 > 도커 설치 확인

Application.properties를 Jenkins에 강제 주입하기 위한 방식
/home/ubuntu/docker-volume/jenkins/workspace/runstory/server/src/main/resources
```

젠킨스 내부 작업

- 젠킨스 - GitLab Webhook 연결한뒤

```
1. Jenkins 내부 Execute Shell을 사용하기 위해서 Gitlab / Docker 프러그인 설치
2. Manage Credential > Add Credential로 연결
 - GitLab Settings > Access Token > Token 발급
 - GitLab API Token을 통해서 ID 생성
 - (Username with Password 에서 위에있는 ID / Password : Token 집어넣기
 - Build when a change is pushed ... 에서 URL + 고급 > Secret Token 받음
 - GitLab Settings > Web Hook > URL, Secret Token 넣기 (자동 배포를 위한 방식)
3. 새로운 Item > 프리스타일 프로젝트로 생성
 - Git 연결
 - Push Event마다 빌드하기 위한 설정
4. Execute Shell을 통해서 Commend에 필요한 코드 사용
 - 젠킨스는 컨테이너를 삭제하고 다시 Build하는 방식을 사용한다는 것을 생각하고 사용
 - 재빌드 > 기존 컨테이너 삭제 > image 및 Container 생성

cd server
chmod +x gradlew
./gradlew clean build
docker rm -f runstory-server
docker rm -f runstory-client
docker rmi runstory-server
docker rmi runstory-client
docker build -t runstory-server ./
docker run -d --name runstory-server -v /home/ubuntu/images:/var/lib/runstory -p 8080:8080 runstory-server
docker build -t runstory-client ../client/runtogether
docker run -d --name runstory-client -v /home/ubuntu/images:/runtogether/src/app/public/runstory -p 3000:3000 runstory-client
```

프론트엔드 배포

- Dockerfile이 있는 폴더 (보통 Server 폴더의 최상단)으로 간 후에 Dockerfile을 Build 시켜서 Image를 생성하고 Container를 생성 run한다.
- 프론트에서 AWS 로컬에 저장되어 있는 이미지를 확인하기 위해서 Volume 설정을 한다.

```
docker build -t runstory-client ../client/runtogether
docker run -d --name runstory-client -v /home/ubuntu/images:/runtogether/src/app/public/runstory -p 3000:3000 runstory-client
```

외부 서비스 문서

💻소셜로그인

🗨KAKAO

- OAuth기반 소셜 로그인 API 제공
- https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api

🗾지도

- 티맵 기반 지도 API 제공
- https://tmapapi.sktelecom.com/main.html#webv2/guide/webGuide.sample1

🌐AWS

- AWS 기반 웹 서버 구축
- https://aws.amazon.com/ko/

데이터베이스 덤프 파일 최신본

exec폴더 내에 첨부

시연 시나리오

- 챗봇을 통한 사용자 가이드 및 UCC 참조