포팅 매뉴얼

1. 버전 정보

FrontEnd

React : 16.16.0

BackEnd

jdk : 11

SpringBoot : 2.7.5

QueryDSL :

Database

- mariaDB : 10.9.3-MariaDB-1:10.9.3+maria~ubu2204
    - host : k7a301.p.ssafy.io
    - port : 3306
    - username : a301
    - password : ssafy!301*A~7
- redis :
    - host : k7a301.p.ssafy.io
    - port : 6379
    - password : ssafy!301*A~7

Infra

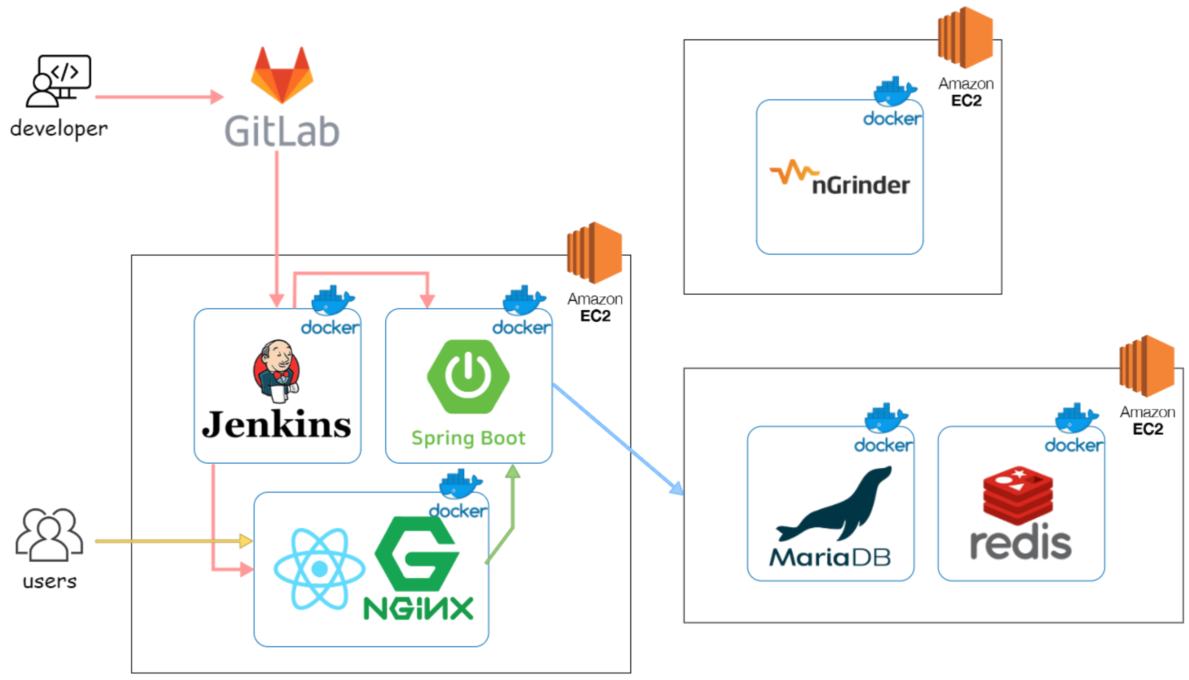
- docker : 20.10.21
- Jenkins : 2.361.2
    - url : http://43.200.196.109:9090/

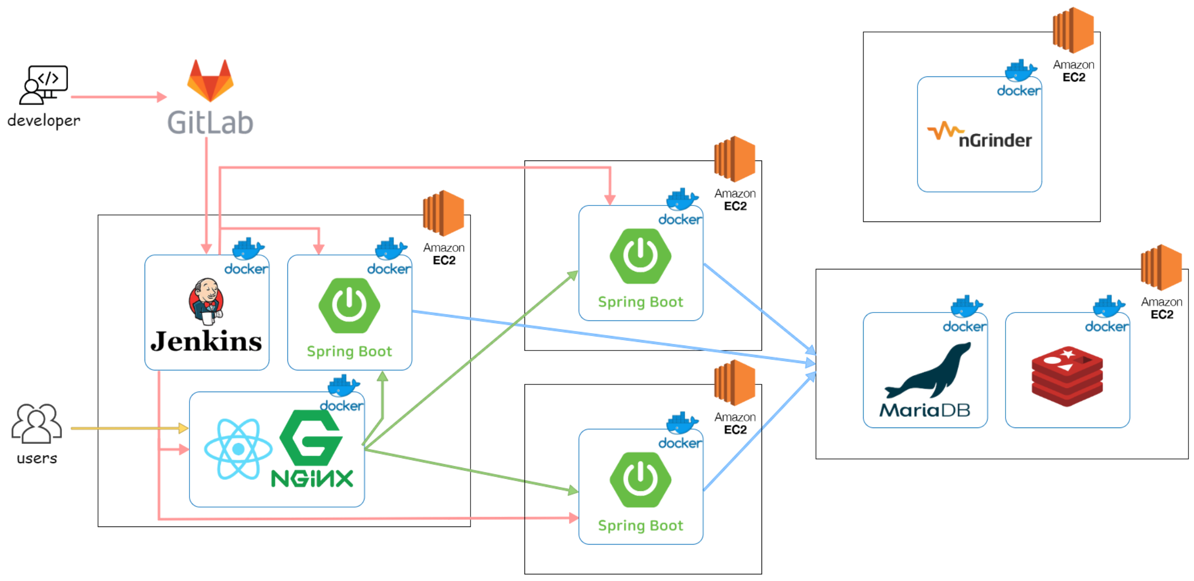
    - id : ahwl8240
    - password : ssafy!301*A~7
- nginx : 1.22.1

IDE

- Intellij
- vscode

2. Architecture





3. 배포 과정

EC2 서버 생성

1. EC2 인스턴스 생성
    a. 인스턴스 유형 : t2.large
    b. AMI 이름 : ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20220912
    c. AMI ID : ami-0e9bfdb247cc8de84
2. 보안그룹
    a. 개방 포트 : HTTP, HTTPS, 9090(Jenkins), 8080(SpringBoot)
3. 각 인스턴스에 탄력적 IP 연결

도커 설치

- 도커 설치를 위한 패키지 설치

```
sudo apt update
sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

- 도커를 설치하기 위해 gpg Key를 다운

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- 도커 설치

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose
```

SSL

- `$ sudo certbot certonly --standalone` - 발급
- `$ sudo ls -al /etc/letsencrypt/live/sword-shield.co.kr` - 확인
- 이후 nginx 설정에 해당 인증서를 포함하면 https 설정 완료

Jenkins 세팅

- docker-componse 파일 생성

`$ vim docker-compose.yml`

```
version: '3'

services:
    jenkins:
        image: jenkins/jenkins:lts
        container_name: jenkins
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - /jenkins:/var/jenkins_home
        ports:
            - "9090:8080"
        privileged: true
        user: root
```

- 젠킨스 컨테이너 생성

`$ sudo docker-compose up -d`

- jenkins 내부에 도커 설치

```
apt update
apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

```
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
apt update
apt install docker-ce docker-ce-cli containerd.io docker-compose
```

React

- Dockerfile

```
FROM node:16.16.0 as build-stage
WORKDIR /var/jenkins_home/workspace/the-knight/frontend
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /var/jenkins_home/workspace/the-knight/frontend/build /usr/share/nginx/html
COPY --from=build-stage /var/jenkins_home/workspace/the-knight/frontend/deploy_conf/nginx.conf /etc/nginx/conf.d/default.conf

# COPY --from=build-stage /var/jenkins_home/ssl/fullchain.pem /etc/letsencrypt/live/sword-shield.co.kr/fullchain.pem
# COPY --from=build-stage /var/jenkins_home/ssl/privkey.pem /etc/letsencrypt/live/sword-shield.co.kr/privkey.pem

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

리액트 빌드 후 nginx에 포함시키는 형태

SpringBoot

- application.yml
- application-auth.yml
- application-server.yml
- Dockerfile

```
FROM openjdk:11

EXPOSE 8080

ARG JAR_FILE=build/libs/the-knight-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java", "-jar", "/app.jar"]

ENV TZ=Asia/Seoul
```

```
RUN apt-get update
RUN apt-get install -y tzdata
```

Nginx

```
upstream backend {
  server 172.31.61.233:8080;
#  server 43.200.122.53:8080;
#  server 43.200.113.29:8080;
}

server {
  listen 80;
  server_name sword-shield.co.kr;
  return 301 https://sword-shield.co.kr$request_uri;
}

server {
  listen 443 ssl http2;
  server_name sword-shield.co.kr;

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;

  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/sword-shield.co.kr/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/sword-shield.co.kr/privkey.pem;

  location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

  location ~ (/api|/pub|/oauth2|/websocket|/swagger|/swagger-resources|/v3) {
        proxy_pass http://backend;
        proxy_set_header Host $http_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-Port $Server_port;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    proxy_buffer_size          128k;
    proxy_buffers              4 256k;
    proxy_busy_buffers_size    256k;
}
server {
    if ($host = sword-shield.co.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen 80;
  server_name sword-shield.co.kr;
    return 404; # managed by Certbot

}
```

Jenkins 프로젝트

- Build Steps - Execute shell : 프로젝트 빌드

```
docker image prune -a --force

cd /var/jenkins_home/workspace/the-knight/frontend/
docker build -t react .

cd /var/jenkins_home/workspace/the-knight/BE/the-knight
chmod +x gradlew
./gradlew clean build -x test --stacktrace
```

```
docker build -t ahwl8240/the-knight-springboot .

docker images

docker push ahwl8240/the-knight-springboot
```

- Build Steps - Execute shell : mm 알림

```
REQUETE="curl -i \
        -X POST \
        -H 'Content-Type: application/json' \
        -d '{ \
                \"channel\": \"$CHANNEL\", \
                \"icon_url\": \"https://www.mattermost.org/wp-content/uploads/2016/04/icon.png\", \
                \"attachments\": [{ \
                        \"fallback\": \"Nouvelle construction Jenkins\", \
                        \"color\": \"#FF8000\", \
                        \"author_name\": \"Jenkins\", \
                        \"author_icon\": \"https://www.jenkins.io/images/logos/jenkins/Jenkins.png\", \
                        \"author_link\": \"https://www.jenkins.io/\", \
                        \"title\": \"배포 완료\", \
                        \"title_link\": \"$BUILD_URL\", \
                        \"text\": \"젠킨스 배포가 완료되었습니다.\", \
                        \"fields\": [{ \
                                  \"short\":true, \
                                  \"title\":\"Branch\", \
                                  \"value\":\"$BRANCH_NAME\" \
                        }, \
                        { \
                                \"short\":false, \
                                \"title\":\"Detail\", \
                                \"value\":\"$BUILD_URL\" \
                        }] \
        }] \
        }'\
        https://meeting.ssafy.com/hooks/a1unr36u3jdwzpm4foj7dmtrbo"
eval $REQUETE
```

- 빌드 후 조치 - Send build artifacts over SSH - Exec command : localhost용

```
if (sudo docker ps -a | grep "react"); then sudo docker rm -f react; fi
if (sudo docker ps -a | grep "springboot"); then sudo docker rm -f springboot; fi

sudo docker run -it -d -p 80:80 -p 443:443 -v /jenkins/ssl:/etc/letsencrypt/live/sword-shield.co.kr --name react react
echo "Run react"
sudo docker run -it -d -p 8080:8080  --name springboot ahwl8240/the-knight-springboot
echo "Run springboot"

sudo docker image prune -a --force
```

- 빌드 후 조치 - Send build artifacts over SSH - Exec command : 다른 서버용

```
sudo docker pull ahwl8240/the-knight-springboot

if (sudo docker ps -a | grep "springboot"); then sudo docker rm -f springboot; fi
sudo docker run -it -d -p 8080:8080  --name springboot ahwl8240/the-knight-springboot

sudo docker image prune -a --force
```

MariaDB

- `$ sudo docker pull mariadb`
- `$ sudo docker run --name mariaDB -p 3306:3306 -e MYSQL_ROOT_PASSWORD=패스워드 -d mariadb`

Redis

- `$ sudo docker pull redis`
- `$ sudo docker run --name redis -d -p 6379:6379 redis`

- redis-cli로 password 설정