Boarda 포팅매뉴얼

프로젝트 기술 스택

- 이슈 관리: Jira
- 형상 관리: Git, Gitlab
- 의사소통, 협업: Notion, Mattermost, Discord
- 개발 환경
  - OS: Windows 10
  - IDE: Intellij, VSCode
  - EC2: Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1051-aws x86_64)
  - Reverse Proxy: Nginx
  - SSH: WSL , MobaXterm
  - SSL: CertBot, Let’s Encrypt
  - Database: MySQL 8.0.35
  - CI/CD: Jenkins

FrontEnd

- React 18.2.0
- react-router-dom 6.21.3
- Node.js 20.10
- Recoil 0.7.7
- Vite 5.0.8
- axios 1.6.7
- styled-components 6.1.8
- @material-tailwind/react 2.1.8
- @mui/material 5.15.7

BackEnd

- Springboot 3.2.2
- Spring Data JPA
- Spring Security

- JWT
- Selenium-java 3.141.59
- Java JDK 17
- QueryDSL 5.0.0
- MySQL 8.0.35
- Amazon S3
- Amazon RDS

EC2 서버 세팅

서버 시간 변경

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

도커 사용

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

docker-compose.yml에 아래 내용 넣기

```
vim docker-compose.yml
```

```
# docker-compose.yml 파일
version: '3'

services:
  jenkins:
    build:
      context: .
      dockerfile: Dockerfile
        # image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home:rw
    # - jenkins_home:/var/jenkins_home:rw
    ports:
      - 9090:8080
      - 50000:50000
        #privileged: true
    user: root
    restart: always
    environment:
      - DOCKER_HOST=unix:///var/run/docker.sock
```

docker-compose.yml 파일 실행

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

아래 명령어를 통해 도커가 잘 작동하고 있는 지 확인하기

```
sudo service docker status
```

```
ubuntu@ip-172-26-15-138:~$ sudo service docker status
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-01-24 21:41:32 KST; 3 weeks 0 days ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 28441 (dockerd)
      Tasks: 82
     Memory: 3.3G
     CGroup: /system.slice/docker.service
             ├─ 28441 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

그래도 안 되면 권한 확인 해보기

```
ubuntu@ip-172-26-15-138:~$ ls -la /var/run/docker.sock
srw-rw---- 1 root docker 0 Jan 24 21:41 /var/run/docker.sock
```

그래도 안 되면 아래 명령어를 통해 유저를 추가하고 서버 재부팅하기

```
sudo usermod -aG docker ${USER}
```

젠킨스 사용

젠킨스 띄우기

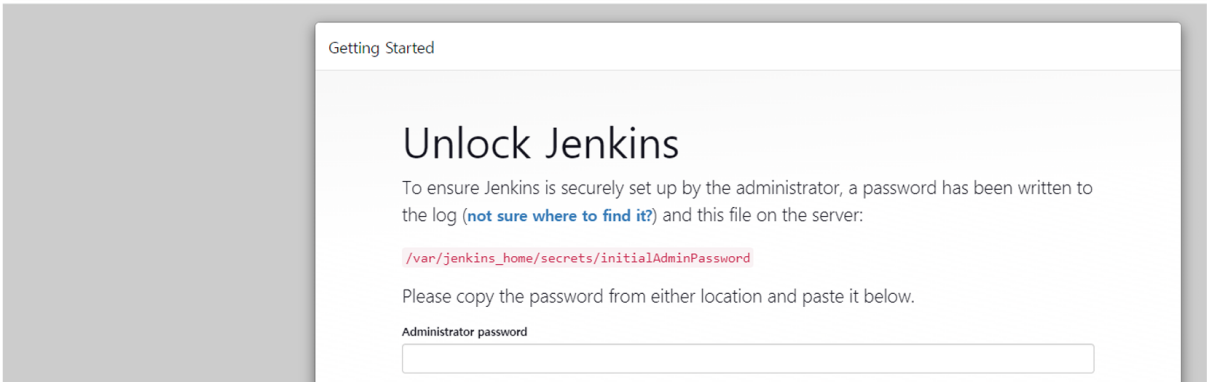
```
sudo docker-compose up -d
```

띄운 젠킨스 컨테이너 확인

```
sudo docker ps
```

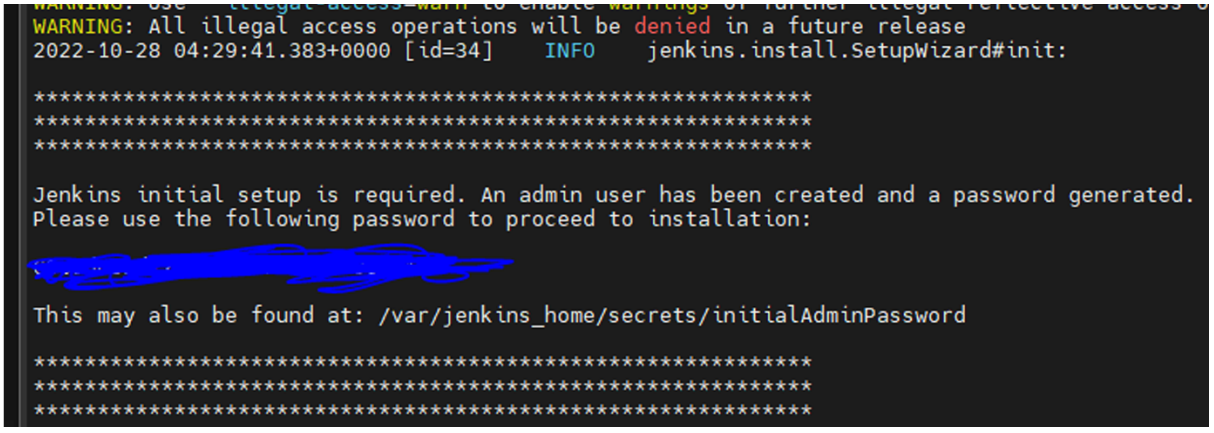
```
2f4a34accb79   ubuntu_jenkins         "/usr/bin/tini -- /u…"   2 weeks ago      Up 2 weeks      0.0.0.0:50000->50000/tcp, :::50000->50000/tcp,
0.0.0.0:9090->8080/tcp, :::9090->8080/tcp   jenkins
```

여기까지 하고 나서 도메인에 접속하면 아래와 같은 화면 확인이 가능



!!! 여기서 입력해야 할 비밀번호는 아래 명령어를 입력하면 볼 수 있습니다

```
sudo docker logs jenkins
```



이것을 다시 젠킨스 암호 입력창에 넣어준다.

암호를 입력하면 Jenkins를 커스터마이즈 하라고 하는데 좌측에 있는 **Install suggested plugins**를 선택해서 설치 해준다.

그 후에 계정을 만들어주고 대시보드의 Jenkins 관리에서 플러그인 매니저에 들어가서 필요한 플러그인(Gitlab, Docker 관련)을 깔아준다.

젠킨스-도커 연결

1. **젠킨스에 도커를 설치하기**

젠킨스 컨테이너에 들어가는 명령어

```
sudo docker exec -it jenkins bash
```

도커 설치

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

2. 프론트엔드와 백엔드에 도커 파일 작성

- **프론트엔드**

```
FROM node:20 as builder

WORKDIR /app

COPY package*.json ./
COPY yarn.lock ./

RUN yarn

COPY . ./

RUN yarn build

FROM nginx:latest

RUN mkdir /app

WORKDIR /app


COPY  --from=builder /app/dist /app/dist

RUN rm /etc/nginx/conf.d/default.conf
```

```
COPY ./default.conf /etc/nginx/conf.d

EXPOSE 3000

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

- **백엔드**

```
FROM openjdk:17-oracle
EXPOSE 8080
ARG JAR_FILE=build/libs/gongtong-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=prod",
```

프론트엔드 프로젝트 폴더에 .dockerignore도 작성

```
/node_modules
```

젠킨스 프리스타일 프로젝트 생성

젠킨스 프리스타일 프로젝트를 프론트엔드용, 백엔드용 2개를 생성하고 Git과 연결해주고, 빌드 유발 조건등을 설정해주고 Secret Token Generate하고 Gitlab 설정

application-secret.yml과 .env 파일

시크릿 키가 포함되어 유출이 되면 안 되는 정보들은 해당 파일에 저장하여

젠킨스 credentials에서 관리

Gitlab Webhook 설정

URL: 도메인/project/아까만든젠킨스프로젝트이름

Secret Token에는 위에서 Generate한 시크릿 토큰을 넣어준다.

웹훅 생성하고 test를 누르고 Push Events를 눌러서 테스트한다.

Jenkins를 확인해보면 빌드가 잘 된 모습을 볼 수 있다.

HTTPS

패키지 업데이트 및 업그레이드

```
sudo apt update
sudo add-apt-repository --remove ppa:certbot/certbot
sudo apt upgrade
```

Nginx 세팅

```
sudo apt install nginx -y
sydo systemctl status nginx
```

SSL 설정

1. letsencrypt 설치

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

2. Certbot 설치

```
sudo apt-get install certbot python3-certbot-nginx
```

3. Certbot Nginx 연결

```
sudo certbot --nginx
```

- 이메일 입력
- 약관 동의 - **Y**
- 이메일 수신 동의
- 도메인 입력 i10{팀코드}.p.ssafy.io
- http 입력시 리다이렉트 여부 - **2**

nginx 환경설정

```
sudo vim /etc/nginx/sites-available/default
```

가비아 도메인 구입

가비아에서 서비스명으로 도메인을 구매했다.

www.boarda.site

DNS 설정

- A타입
- 호스트: www
- 값/위치: {서버 공인 IP}
- TTL: 3600

nginx - default.conf 파일의 내용

```
upstream frontend{
```

```
        server www.boarda.site:3000;
}
upstream backend{
        server www.boarda.site:8080;
}

server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;


        location / {
                # First attempt to serve request as file, the
                # as directory, then fall back to displaying a
                try_files $uri $uri/ =404;
        }

}

server {

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    server_name www.boarda.site; # managed by Certbot


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/www.boarda.site/ful
```

```
    ssl_certificate_key /etc/letsencrypt/live/www.boarda.site
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

    location / {
        proxy_pass http://frontend;
    }

    location /api {
        proxy_pass http://backend;
    }
    location /api/alarm/subscribe{
        proxy_pass http://backend;
        proxy_set_header Connection '';
        proxy_set_header Cache-Control 'no-cache';
        proxy_set_header X-Accel-Buffering 'no';
        proxy_set_header Content_Type 'text/event-stream';
        proxy_buffering off;
        proxy_http_version 1.1;
        chunked_transfer_encoding on;
        proxy_read_timeout 86400s;
    }
}

server {
    if ($host = boarda.site) {
        return 301 https://www.$host$request_uri;
    } # managed by Certbot

        listen 80 ;
        listen [::]:80 ;
    server_name boarda.site;
    return 404; # managed by Certbot


}

server {
```

```
    if ($host = www.boarda.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name www.boarda.site;
    return 404; # managed by Certbot
}
```

외부 서비스

1. kakao API

해당 프로젝트에서는 보드게임카페 정보 API와 카카오맵API를 사용합니다.

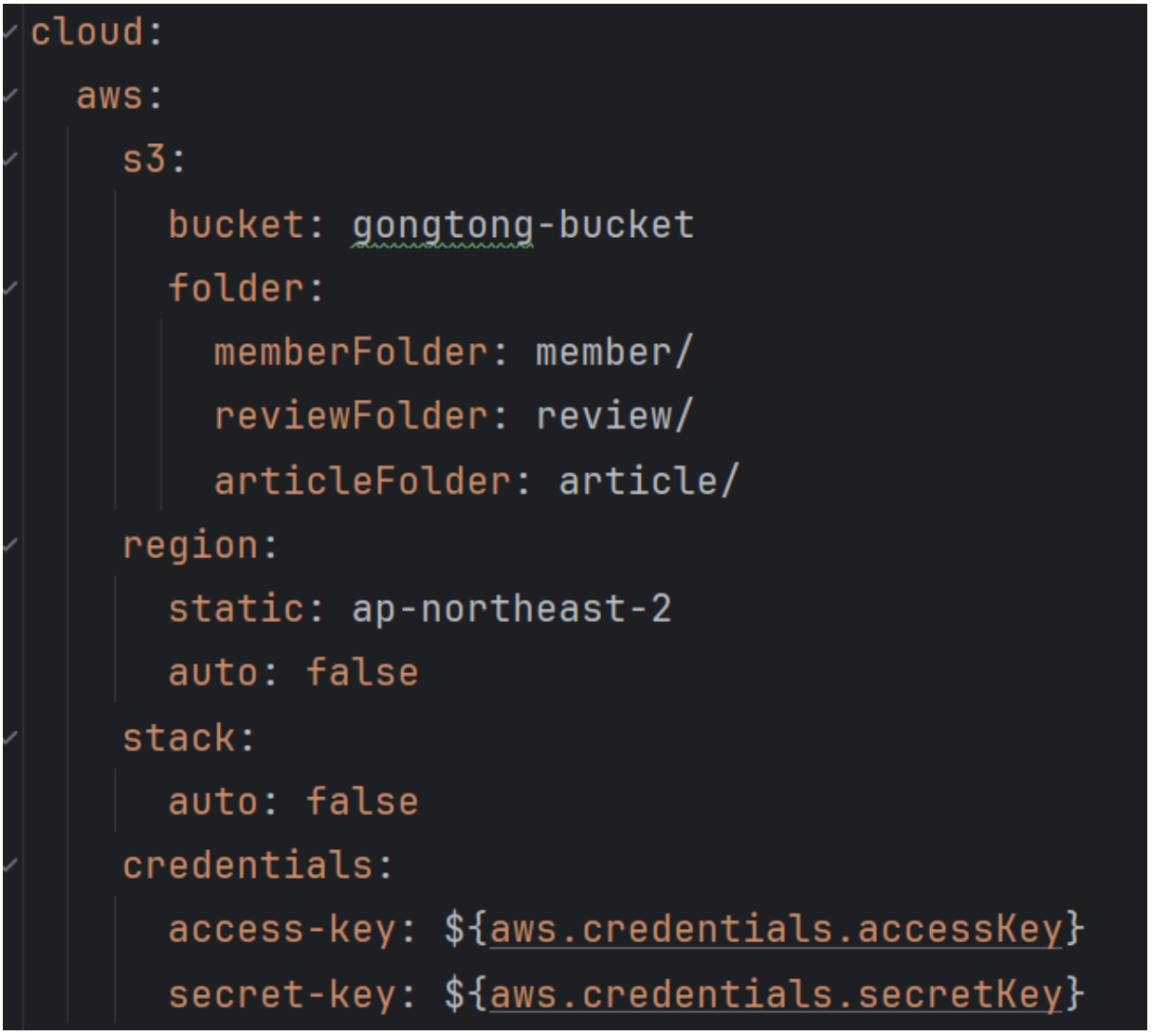
보드게임 매장 위치 표시, 마감임박 모집 지역을 위해 카카오맵 API 를 사용했습니다

application.yml과 .env 환경설정 파일에 kakao api_key를 설정하고
카카오 개발자 도구에서 redirect-uri를 설정하면 사용할 수 있습니다.

2. Amazon S3

AWS에서 제공하는 객체 스토리지 서비스

이미지 파일을 업로드 및 관리하기 위해서 사용합니다.



application.yml과 build.gradle에 의존성을 추가하고 S3 Config 파일을 추가하면 사용할 수 있습니다.

3. Amazon RDS

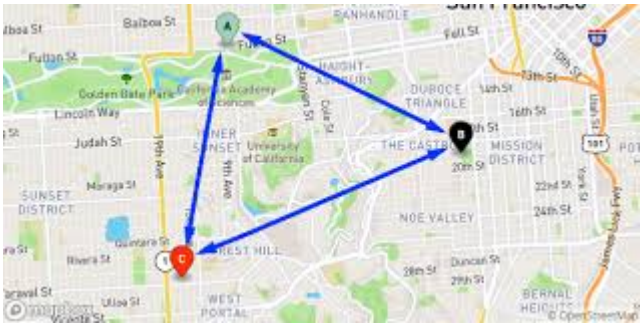
클라우드 환경에서 MySQL 관계형 데이터베이스를 설정하고 운영하기 위하여 사용했습니다.

```yaml
spring:
  profiles:
    include: secret
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${database.prod.url}
    username: ${database.prod.username}
    password: ${database.prod.password}
```

application.yml에 datasource로 명시해주었습니다.

4. Mapbox API

모임 기능에서 지역 선택할 때 화려한 화면 효과와 레이어, 색 채우기 등의 자유로운 커스텀 부가 기능이 용이하여 사용하였습니다.



.env 설정 파일에 Access token key를 설정했습니다.