포팅매뉴얼

1. 개발환경

AWS EC2

- ubuntu : 20.04 LTS
- Docker : 20.10.25
- Nginx : 1.18.0(ubuntu)
- certbot : 0.40.0

Spring boot

- Java : JDK 11.0.19
- Spring boot : 2.7.13
- JPA : 1.0.10
- jwt : 0.11.5
- swagger : 2.9.2
- sockjs : 1.1.2
- websocket : 2.3.3-1
- firebase-admin : 9.2.0

Flask

- Python : 3.7.3
- Flask : 1.0.2

IoT

- Raspberrypi : 5.10.103
- arduino : 1.8.19

React

- react : 18.2.0

- mul-material : 5.14.1
- dayjs : 1.11.9
- stompjs : 2.3.3
- chatscope(chat-ui-kit-react) : 1.10.1
- chatscope(chat-ui-kit-styles) : 1.4.0
- firebase : 10.1.0

Database

- mysql : 8.0.33
- redis : 3.0.504

Certbot SSL Certification

```
$ sudo apt-get install certbot

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

Nginx : 1.18.0

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

```
server {

        server_name i9c107.p.ssafy.io;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                # try_files $uri $uri/ =404;
                proxy_pass http://i9c107.p.ssafy.io:3000;

        }

        location /api {
                proxy_pass http://i9c107.p.ssafy.io:8080;
                proxy_set_header Host $host;
        }

         location /ws/chat {
```

```
                proxy_pass http://i9c107.p.ssafy.io:8080/ws/chat;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }

        location /test500 {
                return 500;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i9c107.p.ssafy.io/fullchain.pem; # managed b
y Certbot
    ssl_certificate_key /etc/letsencrypt/live/i9c107.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 {
    if ($host = i9c107.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 default_server;
        listen [::]:80 default_server;

        server_name i9c107.p.ssafy.io;
    return 404; # managed by Certbot
```

프로젝트 빌드 및 배포

- react docker file :

```
# 이미지 내에서 명령어를 실행할(현 위치로 잡을) 디렉토리 설정
WORKDIR /app

# package.json 워킹 디렉토리에 복사(.은 설정한 워킹 디렉토리를 뜻함)
COPY package.json .

# 이미지 생성 과정에서 실행할 명령어,
# RUN npm install
RUN npm install
```

```
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사
COPY . .

# 3000번 포트 노출
EXPOSE 3000

# 컨테이너 실행 시 실행할 명령어
CMD ["npm", "start"]
```

- spring boot docker file

```
FROM openjdk:11

ARG JAR_FILE=build/libs/*.jar

COPY ${JAR_FILE} app.jar

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

- database docker file

```
FROM mysql:8.0.33

ENV MYSQL_DATABASE juchamsi

ENVMYSQL_USER'user'
ENVMYSQL_PASSWORD'password'
ENVMYSQL_ROOT_PASSWORD'password'
```

- docker-compose

```
version: '3.8'

services:
  frontend:
    image: yuhojae/fe-server-img
    ports:
      - "3000:3000"

  redis:
    image: redis
    ports:
      - 6379:6379

  database:
    container_name: db-server-con
    image: yuhojae/db-server-img
    volumes:
      - juchamsi-db:/var/lib/mysql
```

```
  backend:
    image: yuhojae/be-server-img
    ports:
      - 8080:8080
    depends_on:
      - database
      - redis

volumes:
  juchamsi-db:

networks:
  juchamsi-network:
    driver: bridge
```