포팅 메뉴얼

▼ 목차

▼ 프로젝트 기술 스택

형상 관리

- **GitLab**

이슈 관리

- **Jira**

빌드 및 배포 관리

- **Jenkins 2.375.2**

Server

UX/UI

- **Figma**

Communication

- **Notion**

IDE

- **IntelliJ 2021.2.4**
- **VSCode 1.75.1**

- **AWS EC2**
  - **Docker 20.10.23**

Database

- **MySQL 8.0.31**

프론트엔드

- **React 8.1.2**
  - **Websocket.io**
  - **axios**
  - **node.js 16.13.1**

백엔드

- **Spring 2.7.8**
- **SpringBoot 2.7.8**
  - **openjdk-17-jdk**
  - **Maven**
  - **Swagger2**
  - **Lombok**
  - **spring-data-jpa**

임베디드

- **Raspberry Pi OS (64-bit)**
  - **Websocket.io**
  - **onshape**

▼ Frontend

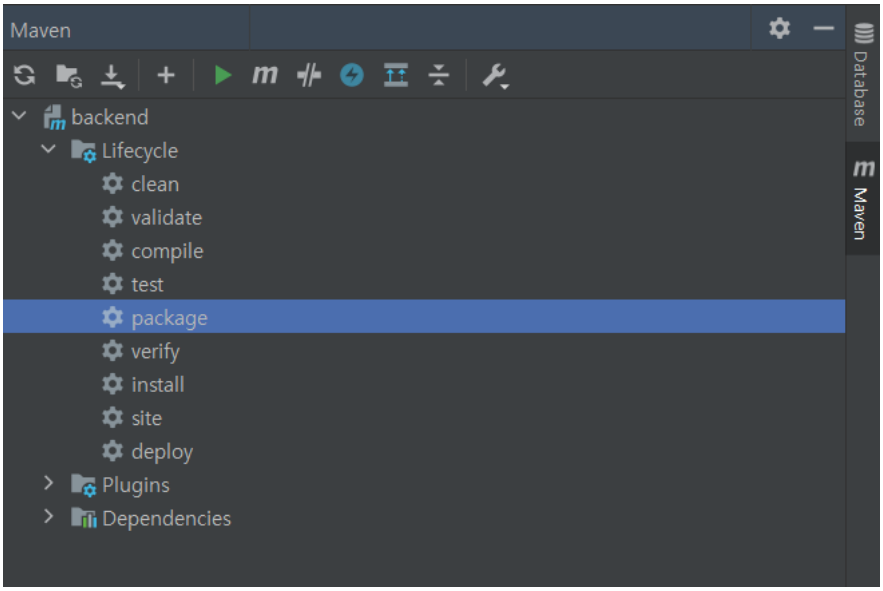
▼ 패키지 설치 및 실행

```
npm install

npm start
```

프토 번호 3000

▼ Backend

우측 Maven 메뉴에서 `Lifecycle` → `package` 를 실행하여 빌드



이후 docker를 이용해 자동 빌드 및 배포

▼ 개발 시 설정

- **Application.properties**

```
spring.datasource.url = jdbc:mysql://localhost:3306/inburger?useSSH=false
spring.datasource.username=root
spring.datasource.password=inburger0@0
spring.jooq.sql-dialect = org.hibernate.dialect.MySQLDialect
spring.jpa.show_sql=true

server.port=8081

spring.jpa.hibernate.ddl-auto=create
spring.sql.init.mode=always
spring.jpa.defer-datasource-initialization=true
spring.mvc.pathmatch.matching-strategy = ant_path_matcher
```

▼ 배포 시 설정

- **Application.properties**

```
spring.datasource.url = jdbc:mysql://<aws 주소>:3306/inburger?useSSH=false
spring.datasource.username=root
spring.datasource.password=inburger0@0
spring.jooq.sql-dialect = org.hibernate.dialect.MySQLDialect
spring.jpa.show_sql=true

server.port=8081

spring.jpa.hibernate.ddl-auto=none
spring.mvc.pathmatch.matching-strategy = ant_path_matcher
```

▼ Raspberry Pi 4.0

▼ 라즈베리 설치 및 초기 설정

라즈베리 파이 SD Card 포맷

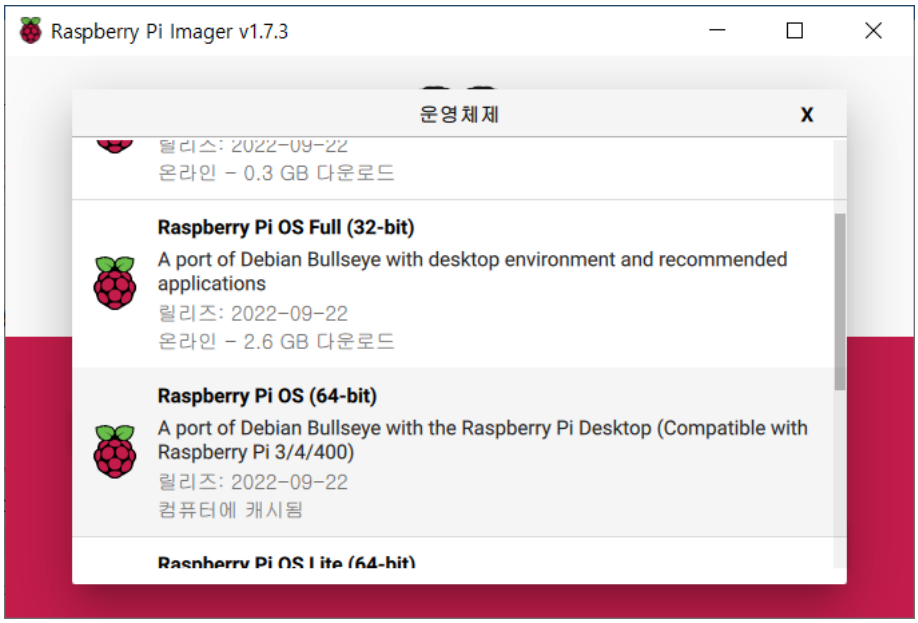
- SD Card Formatter
  - 다운로드

    https://www.sdcard.org/downloads/formatter/sd-memory-card-formatter-for-windows-download/
  - E 드라이브 포맷

🔹 Raspberry Pi OS (64-bit) 설치

- Raspberry Pi Imager
  - 다운로드

    https://www.raspberrypi.com/software/
  - Raspberry Pi Imager 실행 → `운영체제 선택` → `Raspberry Pi OS (Other)` → `Raspberry Pi OS (64-bit)`



🔹 config 설정

- `Preference` → `Raspberry Pi Configuration` → `Interfaces`
  - `SSH`, `VNC`, `SPI`, `I2C`, `Serial Port` 허용
  - 명령창에 `$sudo reboot` 입력

▼ 패키지 설치

🔹 OpenCV

- OpenCV 설치 명령어
  - apt 패키지 설치

```
sudo apt update
sudo apt upgrade -y
sudo apt install build-essential cmake pkg-config -y
sudo apt install libjpeg-dev libtiff5-dev libpng-dev -y
sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev -y
sudo apt install libxvidcore-dev libx264-dev -y
sudo apt install libfontconfig1-dev libcairo2-dev -y
sudo apt install libgdk-pixbuf2.0-dev libpango1.0-dev -y
sudo apt install libgtk2.0-dev libgtk-3-dev -y
sudo apt install libatlas-base-dev gfortran -y
sudo apt install libhdf5-dev libhdf5-serial-dev libhdf5-103 -y
sudo apt install python3-pyqt5 -y
```

  - pip3 패키지 설치

```
pip3 install imutils
pip3 install opencv-contrib-python
```

- 설치 테스트
  - 파일 구조

        - OpenCV_test_dir
          |_ lenna.jpg
          |_ test.py
    - lenna.jpg

    - test.py

```
import cv2

img_color = cv2.imread("./Lenna.jpg", cv2.IMREAD_COLOR)

if img_color is None:
    print("이미지 파일을 읽을 수 없습니다.")
    exit(1)

cv2.imshow("Color", img_color)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

🔹 InsightFace

- 참조
    - 홈페이지

      https://insightface.ai/
    - 깃허브

      https://github.com/deepinsight/insightface
- InsightFace 설치 명령어
    - 설치 전 종속 패키지 버전 확인
        - typing-extensions 최신 버전
            - `pip show typing-extensions` 버전 확인 후
            - `pip install typing-extensions==4.4.0` 최신 버전으로 재설치
        - numpy 1.23 이하 버전
            - `pip show numpy` 버전 확인 후

- `pip install numpy==1.23.5` 다운 버전 재설치

- 설치 명령어

```
pip install -U insightface
pip install onnxruntime
```

- 설치 테스트
  - test.py

```
import cv2
import numpy as np
import insightface
from insightface.app import FaceAnalysis
from insightface.data import get_image as ins_get_image

app = FaceAnalysis(providers=['CUDAExecutionProvider', 'CPUExecutionProvider'])
app.prepare(ctx_id=0, det_size=(640, 640))
img = ins_get_image('t1')
faces = app.get(img)
rimg = app.draw_on(img, faces)
cv2.imwrite("./t1_output.jpg", rimg)
```

Face_recognition

- 참조
  - 깃허브

> https://github.com/ageitgey/face_recognition

https://github.com/ukayzm/opencv/tree/master/face_recognition

- 설치 명령어

```
$ pip install opencv-python
$ pip install opencv-contrib-python
$ pip install dlib
$ pip install face_recognition
$ pip install flask
$ pip install imutils
```

- 소스 코드 다운로드

  https://github.com/ukayzm/opencv/tree/master/face_recognition

- 설치 테스트

  face_recognition 폴더내의 파일 실행

```
$ python camera.py
$ python face_recog.py
$ python live_streaming.py
```

  unknown_face_classifier 폴더내의 파일 실행

```
$ python face_classifier.py 0 -d -S 0.5
```

  옵션

```
$ python face_classifier.py -h
usage: face_classifier.py [-h] [-t THRESHOLD] [-S SECONDS] [-s STOP] [-k SKIP]
                          [-d] [-c CAPTURE]
                          inputfile
```

```
positional arguments:
  inputfile             video file to detect or '0' to detect from web cam

optional arguments:
  -h, --help            show this help message and exit
  -t THRESHOLD, --threshold THRESHOLD
                        threshold of the similarity (default=0.44)
  -S SECONDS, --seconds SECONDS
                        seconds between capture
  -s STOP, --stop STOP  stop detecting after # seconds
  -k SKIP, --skip SKIP  skip detecting for # seconds from the start
  -d, --display         display the frame in real time
  -c CAPTURE, --capture CAPTURE
                        save the frames with face in the CAPTURE directory
```

🔹 GPIO 초음파 센서 테스트

- test.py

```
from gpiozero import DistanceSensor
from time import sleep

sensor = DistanceSensor(21,20) #Echo, Trig

while True:
    print(sensor.distance,"m")
```

Socketio 설치 및 테스트

```
pip install socketio
pip install "python-socketio[client]"
```

- test.py

```
import socketio
from time import sleep

sio = socketio.Client()

sio.connect('접속할 서버 주소')

print('my sid is', sio.sid)

@sio.on('수신할 메세지 tag')
def on_message(data):
    print(data)

sio.emit('발신할 메세지 tag',발신할 data)

sio.disconnect()
```

Raspberry Pi 화면 해상도(1920 X 1080) 및 세로 화면 설정

```
$ sudo vi /boot/config.txt
```

- config.txt

```
display_auto_detect=1

# kiosk mode
display_hdmi_rotate=3
#hdmi_cvt=1920 1080  60 6 0 0 0

# Enable DRM VC4 V3D driver
#dtoverlay=vc4-kms-v3d
max_framebuffers=2

# Disable compensation for displays with overscan
disable_overscan=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
```

```
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[all]

[pi4]
 dtoverlay=vc4-fkms-v3d
# Run as fast as firmware / board allows
arm_boost=1

[all]
enable_uart=0

dtoverlay=rpi-sense
gpu_mem=128
```

화면 터치 좌표 회전 설정

```
$ cd /usr/share/X11/xorg.conf.d/
/usr/share/X11/xorg.conf.d $ ls

# 45와 40 둘중 하나만 있을수 있음
/usr/share/X11/xorg.conf.d $ sudo vi 45-evdev.conf
/usr/share/X11/xorg.conf.d $ sudo vi 40-libinput.conf
```

- 45-evdev.conf 수정

```
Section "InputClass"
        Identifier "evdev touchscreen catchall"
        MatchIsTouchscreen "on"
        MatchDevicePath "/dev/input/event*"
        Driver "evdev"
        Option "TransformationMatrix" "0 -1 1 1 0 0 0 1"
EndSection
```

- 40-libinput.conf 수정

```
Section "InputClass"
        Identifier "libinput touchscreen catchall"
        MatchIsTouchscreen "on"
        MatchDevicePath "/dev/input/event*"
        Driver "libinput"
        Option "TransformationMatrix" "0 -1 1 1 0 0 0 1"
EndSection
```

Raspberry Pi 키오스크 모드 부팅 설정

```
$ sudo apt install unclutter
$ sudo vi /etc/xdg/lxsession/LXDE-pi/autostart
```

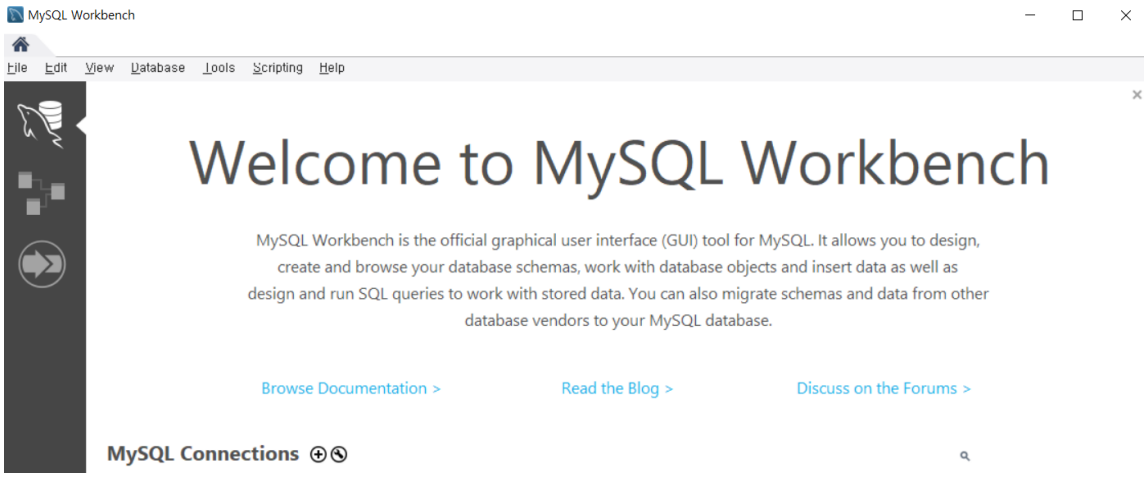
- autostart 수정

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash

# 스크린 세이버 비활성화
@xset s noblank
@xset s off
@xset -dpms

# 마우스 커서 0.1초간 표시후 사라짐
@unclutter -idle 0.1

# 부팅시 chormium을 kiosk모드로 app주소로 접속
/usr/bin/chromium-browser --kiosk --app='접속할 웹 주소'

# 키오스크 프로그램 실행
sudo python 실행할파일.py
```

▼ MySQL

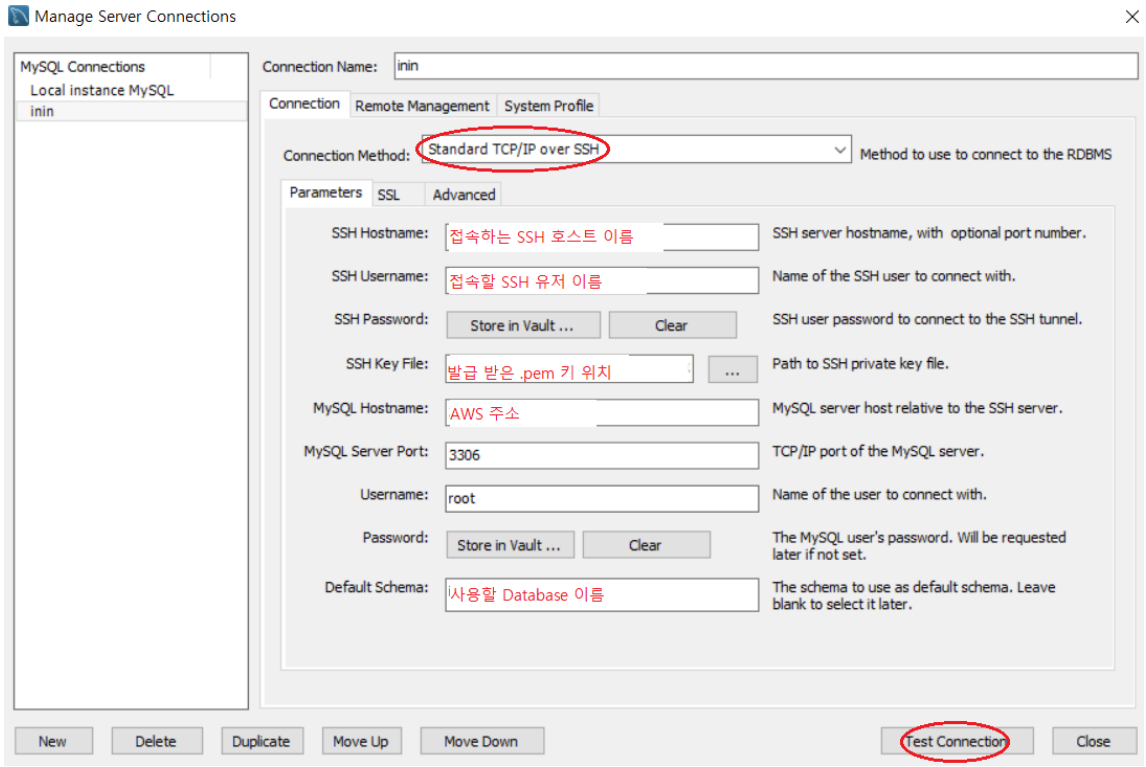
▼ MySQL WorkBench

🔹 DB 연결



MySQL Connections의 ➕ 버튼 누르기

🔹 SSH 연결 설정



Connection Method를 Standard TCP/IP over SSH를 선택하고 정보 입력 후 Test Connection을 눌러 접속 확인

▼ AWS EC2 세팅

▼ SSH

🔹 방화벽 설정(기본)

```
sudo ufw allow ssh
sudo ufw enable
```

▼ Docker

🔹 docker 패키지 최신 버전 확인

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

🔹 apt가 HTTPS를 통해 repository를 이용하는 것을 허용할 수 있도록 해주는 패키지들 설치

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

🔹 docker 공식 GPG key 추가

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

🔹 docker repositoy 등록

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

🔹 docker 설치

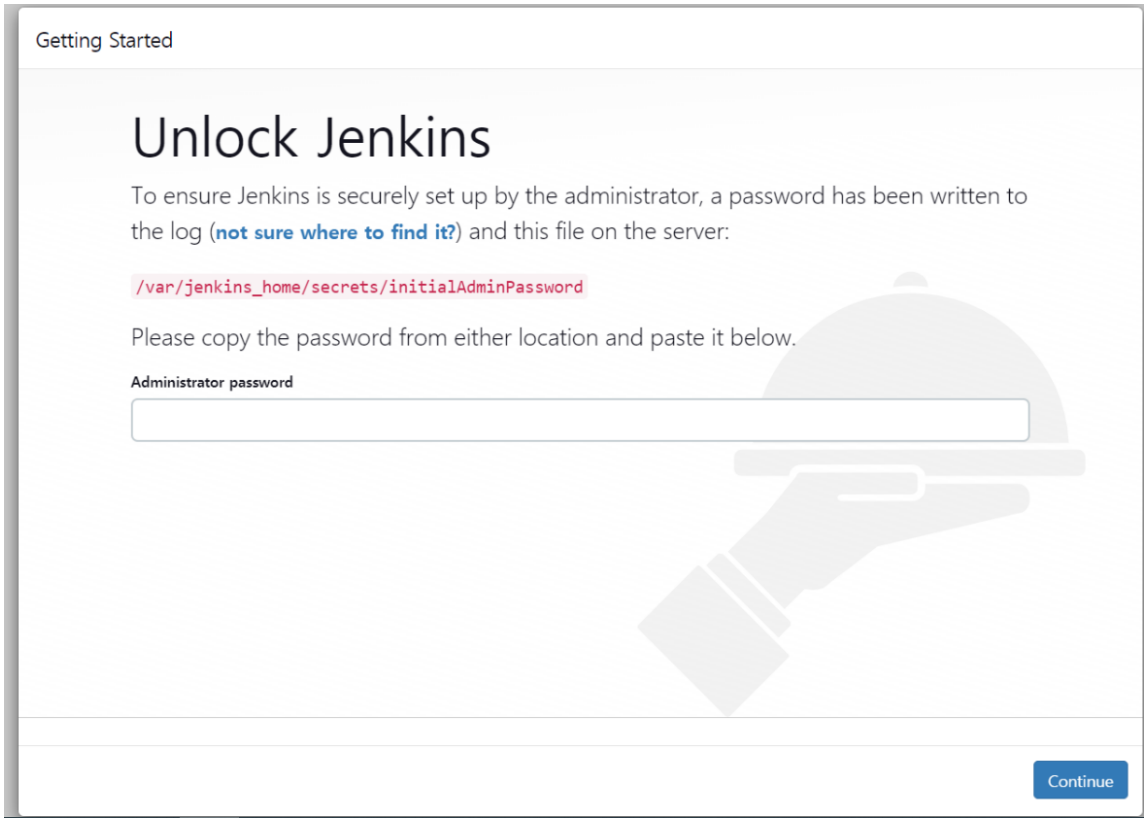
```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

▼ Jenkins

🔹 docker에 jenkins 설치 및 구동

```
docker run -u 0 -d -p 9090:8080 -p 50000:50000 -v /var/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -
```

도커 내부 접속




```
docker exec -it jenkins /bin/bash
```

🔹 **docker old version 제거**

```
apt-get remove docker docker-engine docker.io containerd runc
```

🔹 **password 찾기**

```
apt-get update
apt-get install \
    ca-certificates \
    curl \
    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-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

🔹 **Jenkins에 maven 설치**

```
apt-get update
apt-get install maven

apt-get install openjdk-17-jdk
```

이후 젠킨스 global 설정에서 maven과 jdk 버전 설정

🔹 **password 찾기**

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

▼ MySQL

🔹 docker에 설치 및 구동

```
docker pull mysql
docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=<비밀번호> --name <컨테이너 이름> --character-set-server=utf8mb4 --collation-serve
```

🔹 mysql 접속

```
docker exec -it <컨테이너 이름>
mysql -u root -p
-> <비밀번호 입력>
```

🔹 database 생성

```
create database <database>;
```

- **추가 명령어**

🔹 database 보기

```
show databases;
```

🔹 database 선택

```
use <테이블 명>;
```

🔹 database table 보기

```
show tables;
```

▼ React

🔹 docker에 띄우기

```
docker run -d -p 3000:3000 --name react-frontend --network a203 doodoo3804/react-frontend
```

▼ WebSocket.io Server

🔹 docker 로 Websocket server 띄우기

```
docker run -d -p 4001:4001 --name socket-server --network a203 doodoo3804/socket-server
```

- **App.js 코드**

```
const express = require("express");
const http = require("http");
const sockeIo = require("socket.io");

const port = process.env.PORT || 4001;
const index = require("./routes/index");
const app = express();
app.use(index);

const server = http.createServer(app);

const io = sockeIo(server, {
    cors: {
        origin: ['http://localhost:3000', 'http://3.36.49.220:3000','http://70.12.246.87:3000'],
        megthods: ["GET", "POST"]
    }
});
```

```
let interval;

io.on("connection", (socket) =>{
    console.log("접속");
    let count = 0;
    socket.on("pi", (data) =>{
        console.log(`pi에서 들어온 메시지 수신${count}: ${data}`);
        socket.broadcast.emit('react', data);
        console.log(data);
        count ++;
    });

    socket.on("react", (data) =>{
        console.log(`react에서 들어온 메시지 수신: ${data}`);
        socket.broadcast.emit('pi', data);
    });

    socket.on("disconnect", () => {
        console.log("나감");
        clearInterval(interval);
    });
});

server.listen(port, ()=> console.log(`Listening on port ${port}`));
```

▼ SpringBoot

🔹 **application.properties AWS 설정으로 변경**

```
spring.datasource.url = jdbc:mysql://3.36.49.220:3306/inburger?useSSH=false
```

🔹 **docker에 띄우기**

```
docker run -d -p 8081:8081 --name spring-backend --network a203 doodoo3804/spring-backend
```

▼ Jenkins 세팅

▼ 플러그인 설치

`Jenkins 관리` → `플러그인 관리` → `Available plugins`

- **GitLab**
- **Maven Integration plugin**
- **Publish Over SSH**
- 필요 플러그인 추가 설치

▼ 자동 빌드 설정

🔹 **Credentials 설정**

`Jenkins 관리` → `Credentials`

GitLab 계정으로 생성

🔹 **새 Item 설정**

`새로운 Item` → `Freestyle project` → `Configure` → `General`

- 소스코드 관리
  - 위에서 생성한 Credential 등록
- 빌드 유발
  - GitLab에서 변화 있을 때 webhook 발동 체크
  - `고급` 누르고 Secret token을 Generate로 생성 후 코드 저장

🔹 **GitLab webhook 설정**

`Settings` → `Webhooks`

URL에 Jenkins Item 주소 입력하고 위의 Secret token 기입

▼ 자동 배포 설정

자동 배포는 아래 Docker File 작성 이후 진행

`Configure` → `Build Step` → `Add build step`

🔹 **Websocket server**

Excute shell

```
cd <빌드 경로>
docker login <유저 정보>
docker build -t <이미지 이름>
docker push <이미지 이름>
docker stop socket-server && docker rm socket-server
docker run -d -p 4001:4001 --name socket-server --network a203 <이미지 이름>
```

🔹 **React**

Excute shell

```
cd <빌드 경로>
docker login <유저 정보>
docker build -t <이미지 이름>
docker push <이미지 이름>
docker stop react-frontend && docker rm react-frontend
docker run -d -p 3000:3000 --name react-frontend --network a203 <이미지 이름>
```

이후 docker-compose 파일에 react 부분 작성

🔹 **Invoke top-level Maven targets**

SpringBoot 빌드한 maven 버전과 동일한 버전을 Jenkins에 설치하고 등록

🔹 **SpringBoot & MySQL**

- **doker-compose.yml** 을 작성한 이후 진행

Excute shell

```
cd <빌드 경로>
docker-compose up -d --build
```

▼ Docker 세팅

▼ React

🔹 **/front-end/my-app/Dockerfile**

```
FROM node:14
WORKDIR /app
COPY package.json .
RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

▼ SpringBoot

🔹 **/back-end/Dockerfile**

```
FROM openjdk:17-jdk-alpine
VOLUME /tmp
```

```
EXPOSE 8081
FROM openjdk:11
ARG JAR_FILE=target/inburger.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

▼ docker-compose

- **/back-end/docker-compose.yml**

```
version: "3.3"
services:
  web:
    build: ../front-end/my-app/
    ports:
      - "80:3000"
  database:
    image: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: "1234"
      MYSQL_DATABASE: mydb
    volumes:
      - ./db/data:/var/lib/mysql
      - ./database/conf.d:/etc/mysql/conf.d
      - ./database/initdb.d:/docker-entrypoint-initdb.d
    ports:
      - 3306:3306
  application:
    build: ./backend/
    depends_on:
      - database
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://3.36.49.220:3306/inburger?useSSH=false
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: "1234"
    ports:
      - 8081:8081
    restart: always
```