



포팅 메뉴얼 : 정리 문서

1. 개발환경

1.1 Frontend

- Next.js 15.2.3
- React 19.0.0
- TypeScript ^5
- React Query 5.69.0
- Framer Motion 12.6.2
- MQTT.js 5.10.4
- SweetAlert2

1.2 Backend

- Java 17
- SpringBoot 3.4.4
 - SpringBoot Data JPA
 - Lombok
 - mqttv3
 - mysql-connector-j
- Maven 3.9.9

1.3 ROS2

- python 3.7.5

1.4 AI

- python 3.11.8
- LangChain
- OpenAI tst
- ChatGPT gpt-3.5-turbo
- OpenWeatherMap API
- Google Custom Search API
- Tavily API

1.5 Database

- MySQL
- AWS S3
- ChromaDB

1.6 Server

- Ubuntu 22.04
 - Mosquitto

1.7 IDE

- Visual Studio Code
- IntelliJ

1.8 툴

- Gitlab
 - Jira
 - Mattermost
 - Notion
 - putty
-

2. 인프라 셋팅

2.1 서버 세팅

2.1.1 시간대 변경

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

2.1.2 필요한 포트 개방

```
#ufw 상태 확인  
sudo ufw status  
  
# 사용할 포트 허용하기 (ufw inactive 상태)  
sudo ufw allow 22  
  
# ufw 활성화 하기  
sudo ufw enable
```

2.1.3 Docker 설치

```
# 기존 패키지 목록을 업데이트  
sudo apt-get update  
  
# Docker 설치  
sudo apt-get install -y docker.io  
  
# Docker 서비스 시작  
sudo systemctl start docker  
  
# Docker 서비스 자동 시작 설정  
sudo systemctl enable docker  
  
# Docker 권한 부여 (현재 사용자에게 권한을 부여)  
sudo usermod -aG docker $USER  
# 사용자가 도커 그룹에 추가된 후, 로그아웃한 뒤 다시 로그인해야 권한 변경 적용  
exit
```

```
# Docker 설치 확인
docker --version

# Docker 명령어 실행 확인
docker images

# Docker 시간 및 위치 설정
sudo timedatectl set-timezone Asia/Seoul
# 확인1
timedatectl
# 확인2
date
```

2.1.4 ChromaDB

```
# Docker 이미지 다운로드
docker pull ghcr.io/chroma-core/chroma:latest

# Chroma DB docker 컨테이너 실행
docker run -d -p 8000:8000 ghcr.io/chroma-core/chroma:latest
```

2.1.5 Mosquitto

```
# mosquitto 이미지 다운로드
docker pull eclipse-mosquitto:latest

# 포트 개방
sudo ufw allow 1883/tcp
sudo ufw allow 8083/tcp
sudo ufw reload

# mosquitto 컨테이너 실행
docker run -d -p 8083:8083 -p 1883:1883 --hostname mosquitto --name mosquitto eclipse-mosquitto:latest

# 웹소켓 사용을 위해 config 수정
docker exec -it mosquitto /bin/sh
```

```
cd mosquito/config
vi mosquito.conf
```

1 mosquito.conf

```
listener 8083
protocol websockets

listener 1883
protocol mqtt
```

2 nginx.conf

```
# WebSocket 연결을 위한 설정 (Mosquitto WebSocket 포트)
location /ws/ {
    # SSL 처리 후 내부 ws:// 프로토콜로 전달
    proxy_pass http://mosquitto:8083; # 내부 WebSocket 서버로 전달 (ws
s:// → ws://)

    # WebSocket 헤더 설정
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header Host $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-Proto $scheme;
}
```

2.2 Jenkins

2.2.1 젠킨스 설치 및 실행

```
# apt update
sudo apt update
sudo apt upgrade

# 포트 허용
```

```
sudo ufw allow 8080/tcp
```

```
# jenkins 도커 설치 및 실행
```

```
docker run -d --name jenkins \  
-p 8080:8080 -p 50000:50000 \  
-v jenkins_home:/var/jenkins_home \  
-v /var/run/docker.sock:/var/run/docker.sock \  
-u root \  
jenkins/jenkins:its
```

```
# Jenkins 기본설정하기
```

```
# Gitlab 웹훅 설정
```

```
# gitlab access token 발급
```

```
# gitlab webhook 토큰 발급
```

```
# PipeLine
```

```
# Trigger: 마스터 브랜치가 머지 리퀘스트 승인되면 트리거되도록 설정
```

```
# Dockerhub 푸시
```

- PipeLine Script

```
pipeline {  
  agent any  
  
  environment {  
    // DockerHub 또는 다른 레지스트리 관련 변수 (예시)  
    DOCKER_IMAGE_NAME = ""  
    DOCKER_HUB_CREDENTIALS = credentials('DOCKER_HUB_CREDENTIALS')  
    MYSQL_USER_PASSWORD = credentials('mysql_happie_password')  
    MYSQL_USER = credentials('mysql_user')  
    MYSQL_URL = credentials('mysql_url')  
    MQTT_USERNAME = credentials('mqtt_username')  
    MQTT_PASSWORD = credentials('mqtt_password')  
  }  
  
  tools {  

```

```

    nodejs 'NodeJS 22.13.0'
    maven 'Maven 3.9.9'
  }

  stages {
    stage('clone') {
      steps {
        echo 'Start cloning happie...'
        git branch: 'develop', credentialsId: '75fd6066-d89a-4b17-a4b2-43660139bafd', url: 'https://lab.ssafy.com/s12-mobility-smarthome-sub1/S12P21E103.git'
        echo 'Clone finished'
      }
    }

    stage('Install Dependencies') {
      steps {
        echo 'Installing dependencies...'
        dir('FE') {
          sh 'npm install'
        }
      }
    }

    stage('Build Next.js Application') {
      steps {
        echo 'Building Next.js application...'
        dir('FE') {
          sh 'npm run build'
        }
      }
    }

    stage('Build Backend') {
      steps {
        echo 'Building Spring Boot application with Maven...'
        dir('BE') {
          withCredentials([

```

```

        string(credentialsId: 'mysql_url', variable: 'MYSQL_URL'),
        string(credentialsId: 'mysql_user', variable: 'MYSQL_USER'),
        string(credentialsId: 'mysql_happie_password', variable: 'MYSQL_PASSWORD'),
        string(credentialsId: 'mqtt_username', variable: 'MYSQL_USERNAME'),
        string(credentialsId: 'mqtt_password', variable: 'MYSQL_PASSWORD')
    )}{
        sh '''
            echo "Building with the following credentials..."
            echo "MYSQL_URL=$MYSQL_URL"
            echo "MYSQL_USER=$MYSQL_USER"
            echo "MYSQL_PASSWORD=$MYSQL_PASSWORD" | sed
"s/./*/g" # 비밀번호 감추기

            mvn clean install -f pom.xml
        '''
    }
}

stage('Build Docker Images') {
    steps {
        script {
            sh 'docker build -f FE/Dockerfile -t happie_frontend ./FE'
            sh 'docker build -f BE/Dockerfile -t happie_backend ./BE'
        }
    }
}

stage('Push Docker Images') {
    steps {
        script {
            withCredentials([usernamePassword(credentialsId: 'DOCKER_HUB_CREDENTIALS', passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {

```



```

        sh """
            docker login -u $DOCKER_USERNAME -p $DOCKER_PASS
WORD
            docker tag happie_frontend ${DOCKER_IMAGE_NAME}/ha
ppie_frontend
            docker push ${DOCKER_IMAGE_NAME}/happie_frontend:l
atest
            docker tag happie_backend ${DOCKER_IMAGE_NAME}/ha
ppie_backend
            docker push ${DOCKER_IMAGE_NAME}/happie_backend:l
atest
            docker rmi ${DOCKER_IMAGE_NAME}/happie_frontend:lat
est
            docker rmi ${DOCKER_IMAGE_NAME}/happie_backend:lat
est
            docker rmi happie_frontend
            docker rmi happie_backend
        """
    }
}
}
}

stage('Deploy') {
    steps {
        script {
            echo 'Stopping and removing old containers...'
            sh 'docker-compose -f docker-compose.yml down'

            echo 'Writing environment variables to .env file...'
            withCredentials([
                string(credentialsId: 'mysql_happie_password', variable: 'MYSQL_
PASSWORD'),
                string(credentialsId: 'mysql_user', variable: 'MYSQL_USER'),
                string(credentialsId: 'mysql_url', variable: 'MYSQL_URL'),

```

```

        string(credentialsId: 'mqtt_username', variable: 'MYSQL_USERNAME'),
        string(credentialsId: 'mqtt_password', variable: 'MYSQL_PASSWORD')
    ]) {
        sh '''
        #!/bin/bash
        echo "MYSQL_PASSWORD=${MYSQL_PASSWORD}" > .env
        echo "MYSQL_USER=${MYSQL_USER}" >> .env
        echo "MYSQL_URL=${MYSQL_URL}" >> .env
        '''
    }

    echo 'Starting new containers...'
    sh 'docker-compose --env-file .env -f docker-compose.yml up -d --
build'
    }
}

}

}

post {
    success {
        echo 'Pipeline succeeded!'
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER}
by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
            endpoint: 'https://meeting.ssafy.com/hooks/ucoy',
            channel: 'happie-build'
            )
        }
    }
}

```

```

    }
}
failure {
    echo 'Pipeline failed.'
    script {
        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
        mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
            endpoint: 'https://meeting.ssafy.com/hooks/ucoet5xm5ny',
            channel: 'happie-build'
        )
    }
}
}
}
}

```

2.2.2 젠킨스 컨테이너내 도커, 도커컴포즈 설치

```

# Jenkins 컨테이너 접속
docker exec -it jenkins bash

# Docker CLI 설치
apt-get update
apt-get install -y docker.io

# docker compose 설치
sudo apt-get update
sudo apt-get install -y curl jq
sudo curl -L "https://github.com/docker/compose/releases/download/$(curl -s https://api.github.com/repos/docker/compose/releases/latest | jq -r .tag_name)/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# 실행 권한 부여

```

```
sudo chmod +x /usr/local/bin/docker-compose
```

```
# 설치 확인
```

```
docker-compose --version
```

2.2.3 젠킨스내 한국시간으로 변경

Jenkins 관리 - Users - 설정 icon - Account

Time zone

Select a time zone to use rather than the system default.

Asia/Seoul

Current time on server in Coordinated Universal Time: Mar 26, 2025, 1:59:52 AM; in proposed display zone:
Mar 26, 2025, 10:59:52 AM

Save

Apply

2.3 React

2.3.1 docker-compose

```
version: '3.8'
```

```
services:
```

```
  frontend:
```

```
    build:
```

```
      context: ./FE
```

```
      dockerfile: Dockerfile
```

```
    container_name: my-frontend-container-v1
```

```
    ports:
```

```
      - "80:80"
```

```
      - "443:443"
```

```
    volumes:
```

```
      - /etc/letsencrypt:/etc/letsencrypt:ro
```

```
    networks:
```

```
      - frontend_network
```

```
networks:
  frontend_network:
    driver: bridge
```

2.3.2 dockerfile

```
# 1 빌드 단계 (Next.js 정적 사이트 빌드)
FROM node:22 AS build

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build
#RUN npm run export

# 2 Nginx 서버 단계
FROM nginx:latest

# Nginx 설정 파일 복사
COPY ./nginx.conf /etc/nginx/conf.d/default.conf

# 빌드된 정적 파일을 Nginx에 복사
COPY --from=build /app/out /usr/share/nginx/html

# 포트 설정
EXPOSE 80
EXPOSE 443

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

2.3.3 SSL 인증서 적용

```
# Certbot 설치
sudo apt update
sudo apt install certbot

# Mosquitto용 인증서 발급
sudo certbot certonly --standalone --preferred-challenges http -d j12e103.p.s
```

2.3.4 nginx.conf

```
server {
    listen 80;
    server_name j12e103.p.ssafy.io;

    # HTTP에서 HTTPS로 리다이렉트
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name j12e103.p.ssafy.io;

    # SSL 인증서와 키 설정
    ssl_certificate /etc/letsencrypt/live/j12e103.p.ssafy.io-0001/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j12e103.p.ssafy.io-0001/privkey.pem;

    # SSL 설정 (권장)
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256';
    ssl_prefer_server_ciphers off;

    # 업로드 용량 제한 설정 (100MB)
    client_max_body_size 100M;
```

```

# 리소스 루트 설정
root /usr/share/nginx/html; # 정적 파일 경로 (Next.js 빌드 결과물)
index index.html;

# 기본 페이지 설정
location / {
    try_files $uri $uri/ /index.html; # Next.js의 SPA 처리
}

# 필요한 파일들에 대한 접근을 설정 (예: /_next)
location /_next/ {
    try_files $uri $uri/ =404;
}

# 필요한 파일들에 대한 접근을 설정 (예: /static)
location /static/ {
    try_files $uri $uri/ =404;
}
}

```

2.4 SpringBoot

2.4.1 docker-compose

```

backend:
  build:
    context: ./BE
    dockerfile: Dockerfile
  container_name: happie-backend-v1
  ports:
    - "8085:8080"
  volumes:
    - /etc/localtime:/etc/localtime:ro
  networks:
    - frontend_network
  environment:
    - MYSQL_URL=${MYSQL_URL}
    - MYSQL_USER=${MYSQL_USER}

```

```
- MYSQL_PASSWORD=${MYSQL_PASSWORD}
- MQTT_USERNAME=${MQTT_USERNAME}
- MQTT_PASSWORD=${MQTT_PASSWORD}
```

2.4.2 dockerfile

```
# Docker 이미지의 베이스 이미지를 설정
FROM openjdk:17-jdk

# 작업 디렉토리 설정
WORKDIR /spring-boot

# 호스트 시스템의 target 디렉토리에 있는 JAR 파일을 컨테이너의 /spring-boot/ 디렉토리로 복사
COPY target/happie-0.0.1-SNAPSHOT.jar /spring-boot/app.jar

COPY src/main/resources/application.properties /spring-boot/application.properties

# Spring Boot 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "/spring-boot/app.jar"]
```

2.4.3 nginx.conf

```
location /api/ {
    proxy_pass http://backend:8080;
    proxy_set_header Host $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-Proto $scheme;

    # CORS 문제 해결 (필요 시 추가)
    add_header Access-Control-Allow-Origin *;
    add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS, PUT, DELETE';
    add_header Access-Control-Allow-Headers 'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range';
```



```

        add_header Access-Control-Expose-Headers 'Content-Length,Content-Range';

        # OPTIONS 요청 처리 (CORS Preflight)
        if ($request_method = OPTIONS) {
            return 204;
        }
    }
}

```

2.4.4 application.properties

```

spring.application.name=happie

spring.profiles.active=dev

# mysql
spring.jpa.database=mysql
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=${MYSQL_URL}
spring.datasource.username=${MYSQL_USER}
spring.datasource.password=${MYSQL_PASSWORD}
spring.jpa.properties.hibernate.show_sql=true

# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

# MQTT
mqtt.topic=/robot/destination
mqtt.client-id=happie_spring_pub
mqtt.broker=tcp://j12e103.p.ssafy.io:1883
mqtt.username=${MQTT_USERNAME}
mqtt.password=${MQTT_PASSWORD}

```

3. AI Server

3.1 pip 설치 목록

```
# MQTT 통신을 위한 클라이언트 라이브러리
pip install paho-mqtt
# OpenAI API 사용을 위한 라이브러리
pip install openai
# .env 환경변수 파일 로딩용
pip install python-dotenv
# Chroma 벡터 데이터베이스 사용
pip install chromadb
# 엑셀 데이터 로드 시 사용
pip install pandas
# 벡터화 과정중 사용
pip install numpy
# 텍스트 벡터화나 유사도 측정 시 사용
pip install scikit-learn
# 이미지 처리
pip install pillow
# HTTP 요청 처리
pip install requests
# STT 기능 구현
pip install SpeechRecognition
# 오디오 변환을 위한 라이브러리
pip install pydub
```

3.2 API

3.2.1 ChatGPT

- 임베딩 모델 채택 (OpenAI Text-Embedding-Ada-002-v2)
- LLM 모델 채택 (OpenAI GPT-3.5-turbo)
- <https://platform.openai.com/docs/overview> 에서 키 발급

3.2.2 Weather API

- 날씨 확인
- <https://openweathermap.org/api> 에서 키 발급

3.2.3 Google Custom Search API

- 외부 검색
- <https://console.cloud.google.com/> 에서 키 발급

3.2.4 Tavily API

- 외부 검색
 - <https://app.tavily.com/home> 에서 키 발급
-

4. ROS2

| 스켈레톤 기반

5. Yolo

5.1 pip 설치 목록

5.1.1 pip install -r requirements.txt

```
cd S12P21E103\ROS\yolov5  
pip install -r requirements.txt
```