

포팅 매뉴얼 및 외부 서비스 정보

1. 프로젝트 기술 스택



공통

FrontEnd

BackEnd

IoT

2. Frontend

[패키지 설치 및 실행](#)

[빌드 및 배포](#)

[Dockerfile을 통해 진행](#)

3. Backend

[빌드 및 배포](#)

[Dockerfile을 통해 진행](#)

4. AWS EC2

[Docker](#)

[Jenkins](#)

5. Jenkins

[GitLab PlugIn 설치 및 연동](#)

[자동 빌드 및 배포 설정](#)

[Jenkins Backend execute shell](#)

[Jenkins Frontend execute shell](#)

[Jenkins Pipeline script](#)

6. Docker

[Frontend](#)

[Dockerfile](#)

[nginx.conf](#)

[docker-compose.yml](#)

[Backend](#)

[Dockerfile](#)

[docker-compose.yml](#)

[MySQL](#)

7. Nginx

[/etc/nginx/nginx.conf](#)

8. 외부 서비스 정보

[Simple & Easy Notification Service API by Naver Cloud](#)

1. 프로젝트 기술 스택



공통

상세	내용
GitLab	형상 관리
Jira	일정 및 이슈 관리
Mattermost	커뮤니케이션
Notion	일정 및 문서 관리
IntelliJ	IDE (2022.02)
Visual Studio Code	IDE



FrontEnd

상세	버전
Node.js	18.17.0
NPM	9.6.7
React	5.0.1
Zustand	4.3.9
React-router-dom	6.14.2

TypeScript	5.1.6
------------	-------

BackEnd

상세	버전
JDK (Zulu)	11.0.19
SpringBoot	2.7.13
Mqttv3	1.2.5
JWT	0.9.1
MySQL	8.0.26
Ubuntu	20.04.6 LTS
Nginx	1.18.0
Docker	24.0.5
Docker-compose	2.20.2
Jenkins	2.401.3

IoT

상세	버전
Arduino	Arduino IDE 2.1.1
Raspberry Pi	rasbian os 32bit
paho.mqtt	1.6.1
QT Framework	5.15.2
Flask	1.1.2

2. Frontend

패키지 설치 및 실행

```
npm install
npm start
```

빌드 및 배포

Dockerfile을 통해 진행

3. Backend

빌드 및 배포

Dockerfile을 통해 진행

4. AWS EC2

Docker

```
sudo apt update
sudo apt upgrade
sudo apt install docker-ce
```

Jenkins

- Docker에 Jenkins 설치 및 구동

```
docker run -d -p 5000:5000 -v /var/jenkins:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock --name jenkins-container jenkins/jenkins:lts
```

5. Jenkins

GitLab PlugIn 설치 및 연동

자동 빌드 및 배포 설정

Jenkins Backend execute shell

```
chmod +x backend
cd backend
chmod +x ./gradlew
./gradlew clean build
docker-compose up -d --build
```

Jenkins Frontend execute shell

```
chmod +x frontend
cd frontend
docker-compose up -d --build
```

Jenkins Pipeline script

```
pipeline {
    agent any
    stages {
        stage ('Git clone') {
            steps {
                script {
                    git branch: 'develop', credentialsId: 'ysang10', url: 'https://lab.ssafy.com/s09-webmobile3-sub2/S09P12A101'
                }
            }
        }
        stage ('Backend build') {
            steps {
                script {
                    sh '''
                        chmod +x backend
                        cd backend
                        chmod +x ./gradlew
                        ./gradlew clean build
                        docker-compose up -d --build
                    '''
                }
            }
        }
        stage ('Frontend build') {
            steps {
                script {
                    sh '''
                        chmod +x frontend
                        cd frontend
                        docker-compose up -d --build
                    '''
                }
            }
        }
    }
    post {
        success {
            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: 'ggod',
                    message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(@${Author_Name})\n(<${env.BUILD_URL}|Details>)"
                )
            }
        }
    }
}
```

```

    )
  }
}
failure {
  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>)"
    )
  }
}
}
}
}
}
}

```

6. Docker

Frontend

Dockerfile

```

# node version : 18.16.1
FROM node:18.16.1-alpine as builder
# 작업 위치 지정
WORKDIR /app

# 환경 변수 지정
# 개발 환경에서는 development, 운영(배포) 환경에서는 production
ENV NODE_ENV development
ENV PATH /app/node_modules/.bin:$PATH

# 현재 파일을 복사
COPY . .

# package.json에 명시된 의존성 설치
RUN npm install --force

# 빌드 시작
RUN npm run build

# nginx
FROM nginx:latest
RUN mkdir /app
WORKDIR /app

# 기존 환경 설정 제거
RUN rm /etc/nginx/conf.d/default.conf
# 디렉토리의 conf 파일을 복사
COPY ./nginx.conf /etc/nginx/conf.d
# builder로 부터 /app/build를 복사해옵니다.
COPY --from=builder /app/build /app/build

# 3000번 포트를 노출한다고 써봤지만, 이는 명시하기 위한 것으로 반드시 docker run의 옵션으로 포트 매핑을 해야함.
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]

```

nginx.conf

```

server {
    listen 3000;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}

```

docker-compose.yml

```

version: "3"

# 정의할 서비스들의 목록
services:
  # 서비스 이름 지정

```

```

frontend:
  # 컨테이너 이름 지정 (선택 사항)
  container_name: react-container
  # 이미지 빌드 설정
  build:
    context: . # Dockerfile을 빌드하는 기준 경로 (현재 디렉토리)
    dockerfile: ./Dockerfile # 사용할 Dockerfile의 경로
  # 포트 매핑 (호스트 포트: 컨테이너 포트)
  ports:
    - "3000:3000" # 호스트의 3000 포트와 컨테이너의 3000 포트를 매핑
  networks:
    - A101-network
  restart: always
mosquitto:
  image: "eclipse-mosquitto"
  container_name: mqtt-container
  ports:
    - "1883:1883"
    - "9001:9001"
  volumes:
    - /var/jenkins_home/workspace/A101/frontend/mqtt/config/mosquitto.conf:/mosquitto/config/mosquitto.conf
    - /var/jenkins_home/workspace/A101/frontend/mqtt/data:/mosquitto/data
    - /var/jenkins_home/workspace/A101/frontend/mqtt/log:/mosquitto/log
  networks:
    - A101-network
  restart: always
networks:
  A101-network:
    driver: bridge

```

Backend

Dockerfile

```

# JDK를 설정
FROM azul/zulu-openjdk:11.0.19

COPY /build/libs/*.jar app.jar

ENTRYPOINT java -jar app.jar

# 실행될 포트 설정
EXPOSE 8080

```

docker-compose.yml

```

version: "3"
services:
  spring:
    # 컨테이너 이름
    container_name: spring-container
    build:
      # 경로
      context: .
      # 사용할 docker 파일
      dockerfile: ./Dockerfile
    ports:
      # 지정 포트
      # 호스트의 8080 포트를 컨테이너의 8080 포트와 매핑
      - 8080:8080
    volumes:
      - /var/jenkins_home/workspace/A101/backend/build/libs:/deploy

```

MySQL

```

docker run -d -p 3306:3306 --name mysql-container -e MYSQL_ROOT_PASSWORD=??? mysql:8.0.26

```

7. Nginx

/etc/nginx/nginx.conf

```

server {

    root /var/www/i9a101.p.ssafy.io/html;
    index index.html index.htm index.nginx-debian.html;

    server_name i9a101.p.ssafy.io;

    location /api {
        proxy_pass http://localhost:8000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
    }

    location / {
        #try_files $uri $uri/ =404;
        proxy_pass http://localhost:3000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        # WebSocket 설정 추가
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    # /ws 경로에 대한 WebSocket 프록시 설정
    location /mqtt {
        proxy_pass http://localhost:9001;
        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;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i9a101.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i9a101.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

    # WSS 설정 추가
    #location /mqtt {
    #    proxy_pass http://localhost:9001;
    #    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;
    #}
}

server {

    if ($host = i9a101.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name i9a101.p.ssafy.io;
    return 404; # managed by Certbot

}

```

8. 외부 서비스 정보

Simple & Easy Notification Service API by Naver Cloud

- 회원가입 시 인증번호 및 알림 기능을 위해 NAVER CLOUD PLATFORM에서 지원하는 SMS API 서비스를 사용했습니다.
- <https://www.ncloud.com/>
- <https://console.ncloud.com/dashboard>

