



포팅_메뉴얼

1. 개발 환경
 2. 설정 파일 및 환경 변수 정보
 3. 포트 열기
 4. 시스템 업데이트 및 Docker 설치
 5. Jenkins 설치
 6. 깃랩 웹훅 적용
 7. Docker-compose 파일
 - 1) Mysql 컨테이너
 - 2) Nginx 컨테이너
 - 3) 백엔드 spring boot 컨테이너
 - 4) Redis 컨테이너
 8. 젠킨스 - 메타모스트 연동
 9. 메인 디바이스 - 아대간 통신 구현
 10. jetson orin nano 업로드
-

1. 개발 환경

Server / FE / BE

- **Server** : Ubuntu 22.04.4 LTS
 - **jenkins** : 2.479.3
 - **GitLab**
 - **spring-boot** : 3.4.2
 - **JDK** : openjdk 17.0.14 2025-01-21
 - **node** : v20.18.2
 - **npm** : 10.8.2
 - **mysql** : 8.0.41-1.el9
 - **redis** : 7.4.2
 - **nginx** : 1.26.3
 - **docker** : 27.5.1
 - **docker-compose** : 1.29.2
 - **intelliJ** : 2024.03
 - **vscode** : 1.97.2
 - **React** : 18
 - **Next.js** : 14.2.16
 - **TypeScript** : 5
 - **Tailwind CSS** : 3.4.17
-

AI / EMB

- **boto3** : 1.36.14
- **h5py** : 3.7.0
- **keras** : 2.10.0
- **matplotlib** : 3.7.5
- **numpy** : 1.24.4
- **opencv-python** : 4.11.0.86
- **python** : 3.8.20

- tensorflow : 2.10.0
- torch : 2.1.0a0+41361538.nv23.6

2. 설정 파일 및 환경 변수 정보

application.yaml

```
server:
  port: 8080

spring:
  jwt:
    access:
      expire-time: 1800000
    refresh:
      expire-time: 86400000
    secret: ${SPRING_JWT_SECRET}
  redis:
    host: ${SPRING_REDIS_HOST}
    port: ${SPRING_REDIS_PORT}
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${SPRING_DATASOURCE_URL}
    username: ${SPRING_DATASOURCE_USERNAME}
    password: ${SPRING_DATASOURCE_PASSWORD}
  jpa:
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: false
        show_sql: false
        dialect: org.hibernate.dialect.MySQL8Dialect
    generate-ddl: true
    show-sql: false
    defer-datasource-initialization: true # 하이버네이트가 테이블 만들고 data.sql 실행
```

```
batch:
  jdbc:
    initialize-schema: never

logging:
  level:
    org.hibernate.SQL: debug # SQL 쿼리 로그 출력
    org.hibernate.type: trace # SQL 쿼리 파라미터 로그 출력
    com.your.package: debug
    org.hibernate.tool.hbm2ddl: debug # 스키마 생성 관련 로그
sql:
  init:
    mode: never
# AWS 설정
cloud:
  aws:
    s3:
      region: ${AWS_S3_REGION}
      bucket: ${AWS_BUCKET_NAME}
    stack.auto: false
    credentials:
      accessKey: ${AWS_S3_ACCESS_KEY}
      secretKey: ${AWS_S3_SECRET_KEY}
```

3. 포트 열기

1. 처음 ufw 설정 시 실수로 ssh접속이 안되는 경우를 방지하기 위해 ssh 터미널을 여유있게 2~3개 연결해 놓는다.

2. ufw 상태 확인

```
$ sudo ufw status
```

Status : inactive

3. 사용할 포트 허용하기 (ufw inactive 상태)

```
$ sudo ufw allow 22
```

3-1 등록된 포트 조회하기 (ufw inactive 상태)

```
$ sudo ufw show added
```

Added user rules (see 'ufw status' for running firewall):

```
ufw allow 22
```

4. ufw 활성화 하기

```
$ sudo ufw enable
```

Command may disrupt existing ssh connections. Proceed with operation (y|n)

4.1 ufw 상태 및 등록된 rule 확인하기

```
$ sudo ufw status numbered
```

Status: active

ufw에 허용된 포트 확인

```
sudo ufw status numbered
```

```
sudo ufw allow 80 (80번 포트를 추가로 설정)
```

```
sudo ufw allow 8080 (스프링 부트 기본 포트, 테스트용)
```

```
sudo ufw allow 443 (https)
```

```
sudo ufw allow 9090 (젠킨스)
```

4. 시스템 업데이트 및 Docker 설치

```
sudo apt-get update
```

```
sudo apt-get install apt-transport-https
```

```
sudo apt-get install ca-certificates
```

```
sudo apt-get install curl
```

```
sudo apt-get install software-properties-common
```

도커 공식 GPG(오픈 소스 암호화 도구 GNU Privacy Guard)

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

도커 저장소 추가

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

```
sudo apt-get update
```

도커 설치

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

시스템 실행시 도커 자동 실행

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

도커 버전확인

```
docker --version
```

5. Jenkins 설치

젠킨스 사용을 위한 자바 설치

```
sudo apt update
```

```
sudo apt install openjdk-17-jdk -y
```

```
java -version
```

젠킨스 설치

```
$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install jenkins
```

확인

```
$ sudo systemctl status jenkins
```

jenkins 권한 주기

```
sudo usermod -aG docker jenkins
```

```
$ sudo systemctl start jenkins
```

```
# 초기 비밀번호 확인
```

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

우리의 경우 젠킨스 포트번호가 기존 실행중인 코드와 겹쳐서 오류가 발생하였다.

이를 해결하기 위해 2가지 파일에서의 포트 번호를 변경하였다. (9090)

기본적인 시스템 젠킨스 포트 변경 (이것만 변경했을때는 적용이 되지 않았다)

```
sudo nano /etc/default/jenkins
```

젠킨스 서비스에서 포트 변경

```
sudo nano /usr/lib/systemd/system/jenkins.service
```

젠킨스 재시작(이 명령어 실행전에 데몬 재실행을 하라고 뜬다 해주자 sudo systemctl c
sudo systemctl restart jenkins

확인

```
sudo systemctl status jenkins
```

젠킨스 설정

{서버 주소}:9090 에 접속하여 GUI 환경에서 젠킨스 플러그인을 설치하였다.

추천 플러그인 이외에 설치한 것은 다음과 같다.



[설치 플러그인]

- Generic Webhook Trigger
- Gitlab
- Gitlab API
- Gitlab Authentication
- Mattermost Notification
- Docker pipeline

젠킨스 credential 등록

- gitlab project 연결
- env-file



#env 요소들

```
SPRING_DATASOURCE_PASSWORD
SPRING_DATASOURCE_URL
SPRING_DATASOURCE_USERNAME
SPRING_REDIS_HOST
SPRING_REDIS_PORT
AWS_S3_REGION
AWS_BUCKET_NAME
AWS_S3_ACCESS_KEY
AWS_S3_SECRET_KEY
BACKEND_IMAGE_NAME
SPRING_JWT_SECRET
```

6. 깃랩 웹훅 적용

[1] GitLab Access Token

Token name	Scopes	Created	Last Used ?	Expires	Role	Action
a303token	api, read_api, create_runner, k8s_proxy, read_repository, write_repository, ai_features	Mar 15, 2024	Never	in 4 weeks	Maintainer	

[2] System → GitLab Server

- Jenkins 관리 - Manage Credentials 클릭
- Stores scoped to Jenkins - Domains - (global) - Add credentials 클릭
- Add Credentials 클릭
- 정보 입력 후 **Create** 클릭

- Kind : Gitlab API token 선택
- API tokens : 위에서 발급받은 token 입력
- ID : Credential에 대한 별칭
- GitLab 계정 등록(Username with password)
 - Kind: Username with password
 - GitLab 계정 입력
- Jenkins에서 System → GitLab Server에서 아래와 같이 설정 후 test 연결시 Credentials verified for user project ~~ 나오면 성공
 - credentials는 위에서 만든 credential로 추가

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

☐ Treat username as secret ?

Password ?

ID ?

Create

New credentials

Kind

GitLab API token

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

API token

.....

ID ?

Description ?

Create

GitLab


☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?
A name for the connection

fitter


GitLab host URL ?
The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

https://lab-.com

Credentials ?
API Token for accessing GitLab

GitLab API token

Add ▾

고급 ▾  Edited

Test Connection

7. Docker-compose 파일

1) Mysql 컨테이너

```
mysql:
  image: mysql:8.0
  container_name: mysql-db
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${SPRING_DATASOURCE_USERNAME}
    MYSQL_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
  ports:
    - "3306:3306"
  volumes:
    - /home/ubuntu/mysql_data:/var/lib/mysql
  healthcheck:
    test: [ "CMD", "mysqladmin", "ping", "-h", "localhost" ]
    interval: 10s
    timeout: 5s
    retries: 5
```

2) Nginx 컨테이너

```
nginx:
  image: nginx:stable # Nginx 공식 stable 이미지 사용
  container_name: nginx
  network_mode: "host" # --network host
  volumes:
    - ./cicd/nginx.conf:/etc/nginx/conf.d/default.conf:ro # Nginx 설정 파일 마운트
    - /etc/letsencrypt:/etc/letsencrypt:ro # SSL 인증서 마운트 (읽기 전용)
    - ./cicd/html:/usr/share/nginx/html # 정적 파일 마운트
  restart: always # 컨테이너가 종료되면 자동 재시작
```

3) 백엔드 spring boot 컨테이너

```
app:
  build:
    context: ./backend/Split
    dockerfile: Dockerfile
  image: ${BACKEND_IMAGE_NAME}
  container_name: spring-boot-app
  restart: always
  environment:
    SPRING_DATASOURCE_URL: ${SPRING_DATASOURCE_URL}
    SPRING_DATASOURCE_USERNAME: ${SPRING_DATASOURCE_USERNAME}
    SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
    SPRING_REDIS_HOST: ${SPRING_REDIS_HOST}
    SPRING_REDIS_PORT: ${SPRING_REDIS_PORT}
    AWS_S3_ACCESS_KEY: ${AWS_S3_ACCESS_KEY}
    AWS_S3_SECRET_KEY: ${AWS_S3_SECRET_KEY}
    AWS_S3_REGION: ${AWS_S3_REGION}
    AWS_BUCKET_NAME: ${AWS_BUCKET_NAME}
    SPRING_JWT_SECRET: ${SPRING_JWT_SECRET}
  depends_on:
    mysql:
      condition: service_healthy
    redis:
      condition: service_started
```

```
ports:  
- "8080:8080"
```

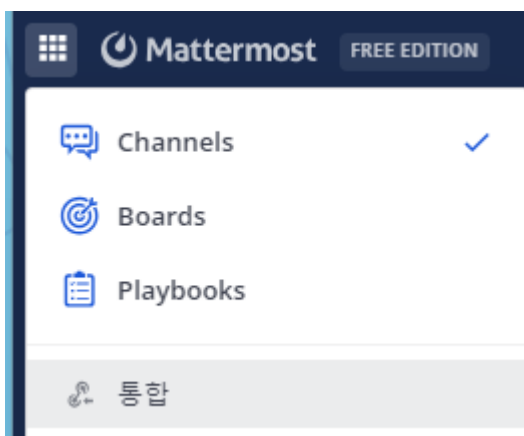
Dockerfile

```
FROM bellsoft/liberica-openjdk-alpine:17  
WORKDIR /app  
COPY build/libs/*.jar app.jar  
EXPOSE 8080  
ENTRYPOINT ["java","-jar","app.jar"]
```

4) Redis 컨테이너

```
redis:  
  image: redis:latest  
  container_name: redis-cache  
  restart: always  
  ports:  
    - "6379:6379"
```

8. 젠킨스 - 메타모스트 연동



앱 디렉터리를 방문하여 Mattermost에 대한 자체 호스팅



전체 Incoming Webhook

Incoming Webhook은 외부 시스템에서 메시지를 받을 수 있게합니다.

- 추가하기 선택
- 제목/설명/수신 받을 채널을 선택

jenkins 빌드 확인 편집 - 삭제

빌드 성공 및 실패에 따른 메시지 반환

URL: <https://meeting.ssafy.com/hooks/97gr1wff1>

sungmin0283(이)가 2025년 2월 7일 금요일에 생성

- 젠킨스 플러그인 설치

Mattermost Notification Plugin 3.1.3

This plugin is a Mattermost notifier that can publish build status to Mattermost channels.
[Report an issue with this plugin](#)

- 젠킨스 관리 - 시스템 설정

Global Mattermost Notifier Settings

Endpoint ?

https://meeting.ssafy.com/hooks/97gr1v

Channel ?

B202

Icon to use ?

endpoint 인풋 혹은

channel - 채널명

테스트 결과 성공

- jenkins 파일에 post 추가

//메타모스트 연동

```
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: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_Name}",
                endpoint: 'https://meeting.ssafy.com/hooks/97gr1wff138tmqum7exw',
                channel: ' B202'
            )
        }
    }
}
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
        endpoint: 'https://meeting.ssafy.com/hooks/97gr1wff138tmqum7exw
        channel: 'B202'
    )
}
}
}

```

9. 메인 디바이스 - 아대간 통신 구현

아대

```

// WiFi 설정
#define WIFI_SSID "DESKTOP-PHKCE21 0954" // 자신의 WiFi SSID로 변경
#define WIFI_PASS "P54446k]" // 자신의 WiFi 비밀번호로 변경

// 서버 설정 (Jetson Orin Nano의 IP와 포트)
#define SERVER_IP "192.168.137.226" // Jetson 보드의 IP 주소로 변경
#define SERVER_PORT 5000 // 사용하는 포트 번호

static const char *TAG = "SOCKET_CLIENT";
static int sock = -1;

```

메인 디바이스 - `avgscore_to_esp32.cpp`

```

const char* esp32_ip = "192.168.137.103"; // ESP32 IP 주소
const int port = 6000; // ESP32 포트

```

10. jetson orin nano 업로드

- jetson에 aws cli 설치 (설치 하고 버전 확인)

```

sudo apt update
sudo apt install awscli

aws -- version

```

- IAM 자격증명 설정 (access key, secret key, region name, output format 순으로 입력)

```
aws configure
```