# 포팅 메뉴얼



## Project stack skill version

- Java : 17
- Spring boot : 3.1.2
- Python : 3.12.2
- MySQL :  8.3.0
- Redis : 7.2.4
- Node.js : 20.11.0
- MongoDB : 7.0.7
- elastic-search : 8.11.4
- monstache : 6.7.17

```
// requirements.txt
APScheduler==3.10.4
```

```
beautifulsoup4==4.12.3
Flask==3.0.2
Flask-Cors==4.0.0
ipykernel==6.29.3
ipython==8.22.2
jupyter==1.0.0
jupyterlab==4.1.4
lxml==5.1.0
matplotlib==3.8.3
nlpaug==1.1.11
nltk==3.8.1
numpy==1.26.4
pandas==2.2.1
py-youtube==1.1.7
PyJWT==2.8.0
pymongo==4.6.2
PyMySQL==1.1.0
requests==2.31.0
scikit-learn==1.4.1.post1
scipy==1.12.0
scrapetube==2.5.1
selenium==4.19.0
tensorflow==2.16.1
tqdm==4.66.2
urllib3==2.2.1
webdriver-manager==4.0.1
youtube-transcript-api==0.6.2
```

## 포트 번호

- Back : 8081
- Front : 5173

- MySQL : 3300
- Redis : 6339
- MongoDB : 8018
- Nginx : 80/443

---

## 환경 변수

- application.properties

  backend/src/main/resources/application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://j10a507.p.ssafy.io:3300/cn
nect?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone
=Asia/Seoul&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=Tkvlxmrghkvmfhwprxm!

spring.jpa.open-in-view=false
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.b
oot.model.naming.PhysicalNamingStrategyStandardImpl
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.dialect=org.hibernate.dialect.MySQL8Inn
oDBDialect
spring.jpa.defer-datasource-initialization=true
spring.sql.init.mode=always
spring.sql.init.encoding=UTF-8

logging.level.org.hibernate.SQL=debug

server.port = 8081
```

```
jwt.token.secret-key=dyAeHubOOc8KaOfYB6XEQoEj1QzRlVgtjNL8PYs1
A1tymZvvqkcEU7L1imkKHeDa
jwt.access-token.expire-length=1209600000
jwt.refresh-token.expire-length=4838400000

spring.data.redis.host=j10a507.p.ssafy.io
spring.data.redis.port=6379
spring.data.redis.password=Tkvlxmrghkvmfhwprxm!

spring.data.mongodb.host=j10a507.p.ssafy.io
spring.data.mongodb.port=8018
spring.data.mongodb.database=cnnect
spring.data.mongodb.username=admin
spring.data.mongodb.password=Tkvlxmrghkvmfhwprxm!
spring.main.allow-bean-definition-overriding=true
spring.data.mongodb.authentication-database=admin

spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=cnnect.user@gmail.com
spring.mail.password=bgyhbooxwbhiqqiu
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=5000
spring.mail.properties.mail.smtp.writetimeout=5000
spring.auth-code-expiration-millis=1800000

spring.data.elasticsearch.host=j10a507.p.ssafy.io
spring.data.elasticsearch.port=9200
```

- .env

/frontend/.env

```
# API URL settings for PJT
VITE_VUE_API_URL=https://j10a507.p.ssafy.io/api
VITE_VUE_API_URL2=https://j10a507.p.ssafy.io/data

# Google Translate API Key
VITE_GT_ACCESS_KEY=AIzaSyAwQaqzrgQv89XtgPun4Vr1gRINB6nrCJA

# ETRI(발음) API Key
VITE_ETRI_ACCESS_KEY=c6646c04-a2c6-4e98-bb89-2ea1fac86d1d

# CLOVA SECRET KEY
VITE_CLOVASPEECH_API_KEY = 125a154a337c459781cab18313a8decf
```

- config.ini

/data/config.ini

```
[DATABASE]
HOST = j10a507.p.ssafy.io
PORT = 3300
DATABASE_NAME = cnnect
USERNAME = root
PASSWORD = Tkvlxmrghkvmfhwprxm!

[MONGODB]
HOST = j10a507.p.ssafy.io
PORT = 8017
DATABASE_NAME = cnnect
USERNAME = admin
PASSWORD = Tkvlxmrghkvmfhwprxm!

[JWT]
SECRET_KEY = Tkvlxmrghkvmfhwprxm!
```

## Dockerfile

- backend

/backend

```
FROM openjdk:17-jdk
ARG JAR_FILE=build/libs/*-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

- frontend

/frontend

```
FROM node:lts-alpine as build-stage

WORKDIR /app

COPY ./package*.json ./

RUN npm install
COPY . .

RUN npm run build


# production stage
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- data

```
FROM python:3.12.2

WORKDIR /app

RUN apt-get update && \
    apt-get install -y wget unzip

RUN wget -q -O - https://dl-ssl.google.com/linux/linux_signing_k
    echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb
    apt-get update && \
    apt-get install -y google-chrome-stable

RUN wget -O /tmp/chromedriver.zip http://chromedriver.storage.go
    mkdir -p /usr/src/chrome && \
    unzip /tmp/chromedriver.zip chromedriver -d /usr/src/chrome

RUN apt-get update && \
    pip install --upgrade pip && \
    pip install flask

COPY requirements.txt /app/
RUN pip install -r requirements.txt

COPY . /app

EXPOSE 5000

CMD ["python3", "-m", "flask", "run", "--host=0.0.0.0"]
```

## Jenkins 쉘 스크립트

- backend

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
        jdk 'jdk-17'
        dockerTool 'Docker'
    }

    stages {
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'develop-be', url: 'https://lab.s
safy.com/s10-bigdata-recom-sub2/S10P22A507.git',
                credentialsId: 'gitlab-personal'
            }
        }
        stage('Apply application.properties files') {
            steps {
                withCredentials([file(credentialsId: 'applica
tion', variable: 'secretFile')]) {
                    script {
                        sh 'mkdir -p backend/src/main/resourc
es/'
                        sh 'cp $secretFile backend/src/main/r
esources/application.properties'
                    }
                }
            }
```

```
        }
        stage('BE-Build') {
            steps {
                dir("./backend") {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stack
trace
                    '''
                }
            }
        }
        stage('Delete existing Docker images and containers')
{
            steps {
                sh'''
                    if docker container inspect cnnect_server
>/dev/null 2>&1; then
                        echo "container exists locally"
                        docker stop cnnect_server
                        docker rm cnnect_server
                    else
                        echo "container does not exist locall
y"
                    fi
                    if docker image inspect server >/dev/null
2>&1; then
                        echo "Image exists locally"
                        docker rmi server
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }
```

```
        stage('Build and Deploy Docker') {
            steps {
                dir('./backend') {
                    sh'''
                        echo [BE] Build Docker Image!
                        docker build -t server .
                        echo [BE] Run Docker Container!
                        docker run -dp 8081:8081 -e TZ=Asia/S
eoul --name cnnect_server server
                    '''
                }
            }
        }
    }
}
```

- frontend

```
pipeline {
    agent any
    tools {
        nodejs 'nodejs'
        dockerTool 'Docker'
    }
    stages {
        stage('Clear current directory') {
            steps {
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git url: 'https://lab.ssafy.com/s10-bigdata-rec
```

```
                branch: 'develop-fe',
                credentialsId: 'gitlab-personal'
            }
        }
        stage('Apply .env files') {
            steps {
                withCredentials([file(credentialsId: 'env', vari
                    script {
                        sh 'mkdir -p frontend'
                        sh 'cp $secretFile frontend/.env'
                    }
                }
            }
        }
        stage('List Contents in Frontend Directory') {
            steps {
                dir('./frontend') {
                    sh 'ls -la'
                }
            }
        }
        stage('FE-Build') {
            steps {
                dir('./frontend') {
                    sh'''
                        if docker container inspect cnnect_clien
                            echo "container exists locally"
                            docker stop cnnect_client
                            docker rm cnnect_client
                        else
                            echo "container does not exist local
                        fi
                        if docker image inspect client >/dev/nul
                            echo "Image exists locally"
                            docker rmi client
                        else
```

```
                            echo "Image does not exist locally"
                        fi
                        docker build -t cnnect_client -f ./Docke
                    '''
                }
            }
        }
        stage('Build and Deploy Docker') {
            steps {
                dir('./frontend') {
                    sh'''
                        echo [FE] Run Docker Container!
                        docker run -dp 5173:80 --name cnnect_cli
                    '''
                }
            }
        }
    }
}
```

- data

```
pipeline {
    agent any
    tools {
        dockerTool 'Docker'
    }
    stages {
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
```

```
        stage('Git Clone') {
            steps {
                git branch: 'develop-data', url: 'https://lab.ss
                credentialsId: 'gitlab-personal'
            }
        }
        stage('Apply config.ini files') {
            steps {
                withCredentials([file(credentialsId: 'config', v
                    script {
                        sh 'mkdir -p data/'
                        sh 'cp $secretFile data/config.ini'
                    }
                }
            }
        }
        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect cnnect_data >/de
                        echo "container exists locally"
                        docker stop cnnect_data
                        docker rm cnnect_data
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect data >/dev/null 2>&1
                        echo "Image exists locally"
                        docker rmi data
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }
        stage('Build and Deploy Docker') {
```

```
            steps {
                dir('./data') {
                    sh'''
                        echo [DATA] Build Docker Image!
                        docker build -t data .
                        echo [DATA] Run Docker Container!
                        docker run -dp 5000:5000 -e TZ=Asia/Seou
                    '''
                }
            }
        }
    }
}
```

## NginX SSL 설정

```
sudo snap install --classic certbot
sudo apt-add-repository -r ppa:certbot/certbot
sudo apt-get -y install python3-certbot-nginx

sudo certbot --nginx -d j10a507.p.ssafy.io
```

## NginX Reverse proxy 설정

/etc/nginx/sites-available/default.conf

```
server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    server_name j10a507.p.ssafy.io;

    client_max_body_size 100M;
```

```
    ssl_certificate /etc/letsencrypt/live/j10a507.p.ssafy.io/ful
    ssl_certificate_key /etc/letsencrypt/live/j10a507.p.ssafy.io
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    location / {
        proxy_pass http://localhost:5173;
        error_page 405 =200 $uri;
    }

    location /api {
        proxy_pass http://localhost:8081;
    }

    location /data {
        proxy_pass http://127.0.0.1:5000;
    }

    location /daum {
        proxy_pass https://dic.daum.net;
        proxy_set_header Host dic.daum.net;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
        rewrite ^/daum(/.*)$ $1 break;
    }

    location /naverapi{
        proxy_pass https://clovaspeech-gw.ncloud.com;
        proxy_set_header Host clovaspeech-gw.ncloud.com;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
        rewrite ^/naverapi(/.*)$ $1 break;
    }
}

server {
    if ($host = j10a507.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }


        listen 80 ;
        listen [::]:80 ;
    server_name j10a507.p.ssafy.io;
    return 404;


}
```

front container 내부 /etc/nginx/conf.d/nginx.conf

```
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri /index.html;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
```

```
    }
}
```

## MySQL

```
docker run -d --name mysql-container -p 3306:3306 -v mysql-vo
lume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=Tklvxmrghkvmfhwprx
m! mysql:latest
```

## Redis

```
docker run -d --name redis-container -p 6379:6379 redis:lates
t redis-server --requirepass "Tkvlxmrghkvmfhwprxm!"
```

## MongoDB ReplicaSet 생성

```
openssl rand -base64 756 > mongodb.key
chmod 400 mongodb.key
```

```
version: '3.8'

services:

  mongo1:
    image: mongo:latest
    hostname: mongo1
    container_name: mongo1
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: Tkvlxmrghkvmfhwprxm!
    ports:
      # 로컬 27017 포트로 요청이 들어오면 컨테이너의 27017 포트로 리다
이렉트되도록 설정
```

```
      - 8018:27017
    volumes:
      - ./data/db/replica/mongo1:/data/db
      - ./mongodb.key:/etc/mongodb.key
    command:
      - '--replSet'
      - 'myReplicaSet'
      - '--keyFile'
      - '/etc/mongodb.key'
      - '--bind_ip_all'
```

## Elastic Search

- Dockerfile (custom-elasticsearch 이미지 생성용)

```
FROM docker.elastic.co/elasticsearch/elasticsearch:8.11.4

USER root
RUN apt-get update && apt-get install -y vim
USER elasticsearch
```

- docker-compose.yml

```
version: '3'

services:
  elasticsearch:
    image: custom-elasticsearch
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      - discovery.type=single-node
    networks:
      - mongoCluster2
```

```
  monstache:
    restart: always
    image: rwynn/monstache:6.7.17
    command: -f ./monstache.config.toml &
    volumes:
      - ./config/monstache.config.toml:/monstache.config.toml
    depends_on:
      - elasticsearch
    links:
      - elasticsearch
    ports:
      - "8082:8082"
    networks:
      - mongoCluster2

networks:
  mongoCluster2:
```

- monstache.config.toml

```
mongo-url = "mongodb://admin:Tkvlxmrghkvmfhwprxm!@mongo1:27017/
elasticsearch-urls = ["http://elasticsearch:9200"]
direct-read-namespaces = [ "cnnect.data"] // mongodb안에 cnnect
dropped-collections = false
dropped-databases = false
resume = false
resume-write-unsafe = true
index-as-update = true
index-oplog-time = true
verbose = true

[[script]]    # ES index명 설정
script = """
module.exports = function(doc, ns) {
```

```
    doc._meta_monstache = { index: ns.replace(".","-") };
    return doc;
}
"""
```