포팅 메뉴얼

목차

1. [사용 도구]

- **이슈 관리 :** Jira
- **형상 관리 :** GitLab, GitHub
- **커뮤니케이션 :** Notion, MatterMost, Discord
- **디자인 :** Figma, ERDCloud, draw.io
- **CI/CD :** Jenkins, Docker

2. [개발 도구]

- **Visual Studio Code :** 1.86.2
- **Intellij :** 2024.1 (Ultimate Edition)
- **Terminus :** 8.12.2

3. [개발 환경]

FrontEnd

- **Node.js :** 20.11.1
- **Vue3 :** 3.4.23
- **Chart.js :** 5.3.1
- **Sock.js :** 1.6.1
- **Stomp.js :** 1.2.6

BackEnd

- **Java :** 17.0.11
- **Spring Boot :** 3.2.4

Server

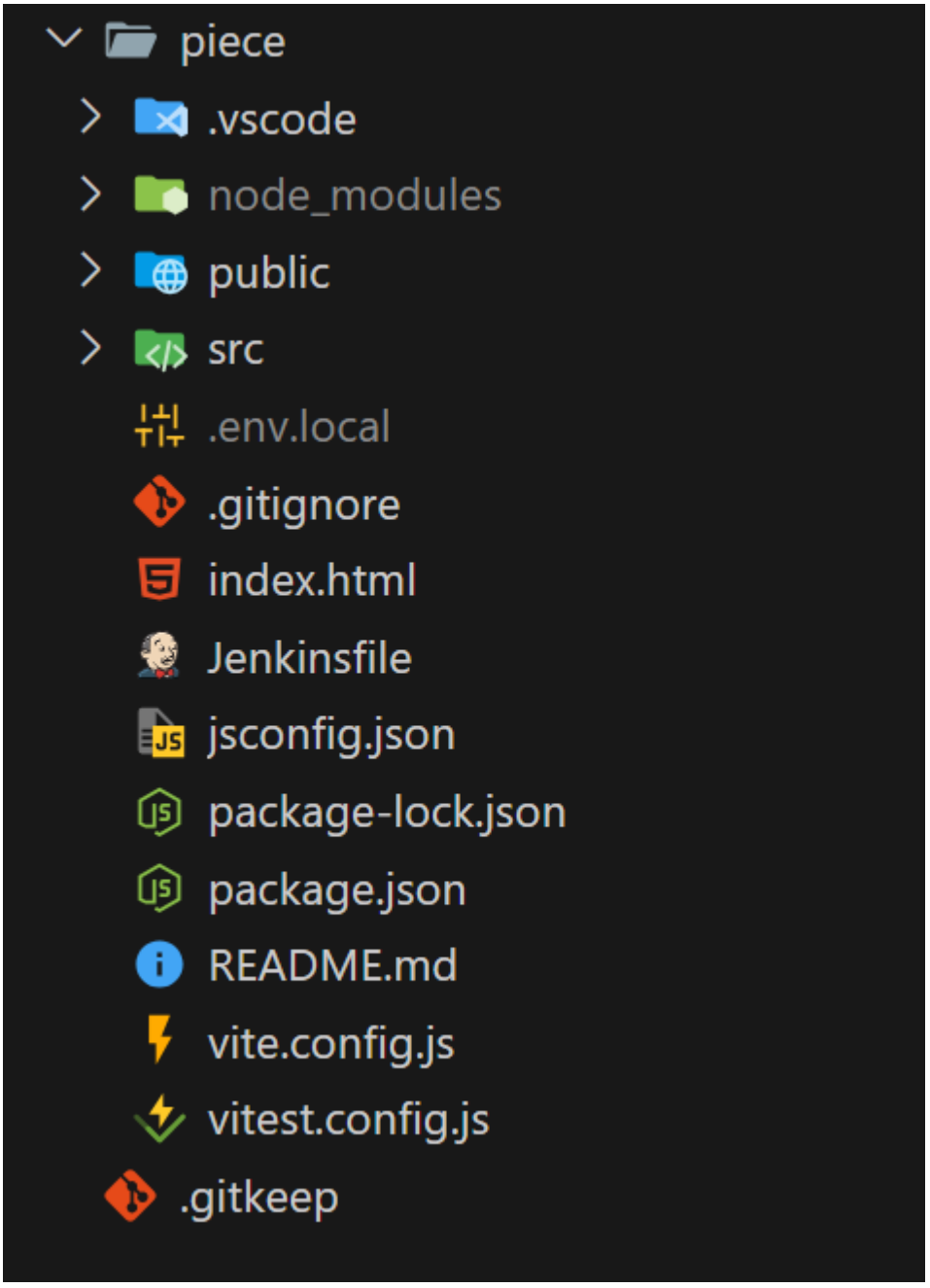
- AWS ec2
- AWS S3
- AWS CloudFront
- AWS Route 53

Service

- **Nginx :** 1.18.0
- **Jenkins :** 2.456
- **Docker :** 26.1.2
- **MongoDB :** 5.0.17
- **MySQL :** 8.0.36

4. [환경변수]

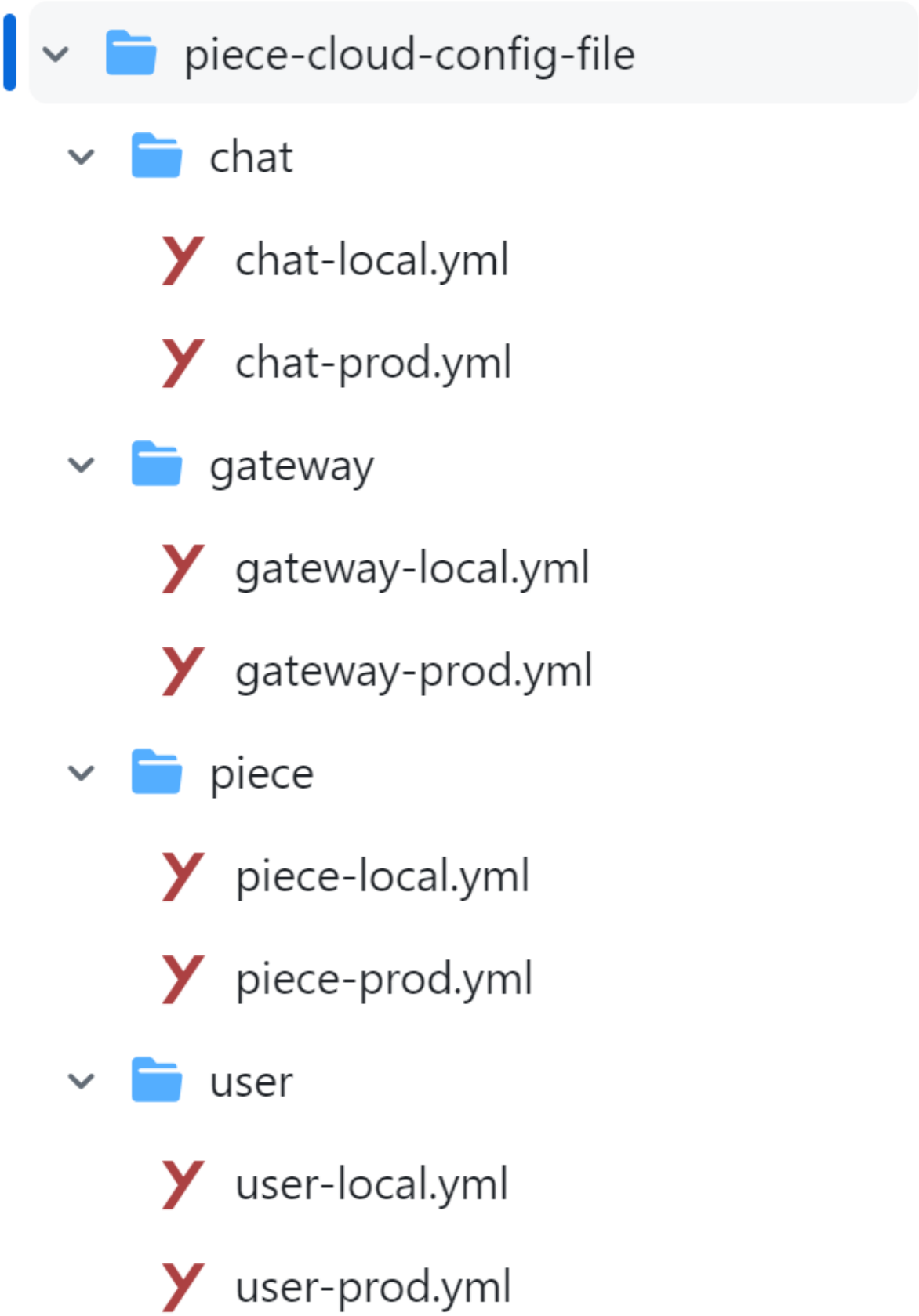
FrontEnd



.env.local

```
# VITE_REST_URL=http://localhost:8000/api
VITE_REST_URL=https://k10b202.p.ssafy.io/api
VITE_REST_PIECE_API=${VITE_REST_URL}/piece
VITE_REST_USER_API=${VITE_REST_URL}/user
VITE_REST_CHAT_API=${VITE_REST_URL}/chat
```

BackEnd



piece-local.yml

```
server:
  port: 0 # Random Port
  servlet:
    context-path: /piece
  baseURL: http://localhost:8000

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/piece?useSSL=false&allow
    username: ssafy
    password: 1234
  jpa:
    hibernate:
      ddl-auto: update
      naming:
        physical-strategy: org.hibernate.boot.model.naming.Ph
    defer-datasource-initialization: true
    properties:
      hibernate.format_sql: true
      dialect: org.hibernate.dialect.MySQL8InnoDBDialect
  application:
    name: piece
eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8761/eureka
  instance:
    instance-id: ${spring.application.name}:${spring.applicat
    hostname: localhost
    prefer-ip-address: true
management:
  endpoints:
    web:
      exposure:
        include: refresh, health, beans, busrefresh, info, me
```

```
logging:
  level:
    org.hibernate.SQL: debug
    org.hibernate.type: trace

# feign client url 세팅
external:
  chat:
    host: ${server.baseURL}/api/chat
  user:
    host: ${server.baseURL}/api/user

# secret key
secret:
  kobis:
    api-key: "{cipher}7f548667e7181d9c28a110380c93bb3d8de1aad
  tmdb:
    api-key: "{cipher}732b759b318ace73151a1f817e5f1361445563b
  kopis:
    api-key: "{cipher}b0a67dff9bfbd50a017021c9ced17805143ea9c

# s3
amazon:
  aws:
    access-key: "{cipher}bd13e0d628912dafe4c7e19a40cf35fe84c0
    secret-key: "{cipher}935080aabc0254b56fc200d65d165928f613
    region: ap-northeast-2
    bucket: piecemaker.kr
    prefixAddress: "{cipher}832a154862b127a501866a2f7ff491493

# OpenAi
openai:
  gpt-model: "{cipher}42b799167ab49394709fbdd89e1e482ad2581e0
  gpt-key: "{cipher}0ca18cce91a3663e556498373a841e255faddc8fa
  gpt-url: "{cipher}351c5cad9df3defdcaccc9606301cffcb5af49d14
  dalle-model: "{cipher}e3cd7af5586a98c89e849a7c10074bd4765ff
  dalle-key: "{cipher}13ae980b4107c720076ba9dce7a348d585ed6b2
```

```
  dalle-url: "{cipher}1cb8eb5ca5bb4571b44f0d17f61a3265e8c902e

base:
  service-url: "${server.baseURL}/api/piece"
```

build.gradle - piece

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.4'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

// docker setting
jar {
    enabled = false
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

ext {
```

```
    set('springCloudVersion', "2023.0.1")
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-star
//    implementation 'org.springframework.boot:spring-boot-st
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-dev
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-
    testImplementation 'org.springframework.security:spring-s

    // spring cloud eureka
    implementation 'org.springframework.cloud:spring-cloud-st

    // spring cloud config
    implementation 'org.springframework.cloud:spring-cloud-st
    implementation 'org.springframework.cloud:spring-cloud-st

    // feign client
    implementation 'org.springframework.cloud:spring-cloud-st
    runtimeOnly 'io.micrometer:micrometer-registry-prometheus

    // s3
    implementation 'com.amazonaws:aws-java-sdk-s3'
    implementation platform('com.amazonaws:aws-java-sdk-bom:1
//    implementation 'io.awspring.cloud:spring-cloud-aws-star
//    implementation 'io.awspring.cloud:spring-cloud-starter-

    // qeurydsl
    implementation 'com.querydsl:querydsl-jpa:5.0.0:jakarta'
    annotationProcessor "com.querydsl:querydsl-apt:5.0.0:jaka
```

```
    annotationProcessor "jakarta.annotation:jakarta.annotatio
    annotationProcessor "jakarta.persistence:jakarta.persiste

    // OpenAI
    implementation group: 'com.theokanning.openai-gpt3-java',
    implementation 'org.springframework.boot:spring-boot-star
    implementation group: 'com.fasterxml.jackson.core', name:
    implementation group: 'com.fasterxml.jackson.core', name:

    // https://mvnrepository.com/artifact/javax.xml.bind/jaxb
    implementation group: 'javax.xml.bind', name: 'jaxb-api',

}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-depe
    }
}

tasks.named('test') {
    useJUnitPlatform()
}

// Querydsl 설정부
def generated = 'src/main/generated'

// querydsl QClass 파일 생성 위치를 지정
tasks.withType(JavaCompile) {
    options.generatedSourceOutputDirectory = file(generated)
}

// java source set에 querydsl QClass 위치 추가
sourceSets {
    main.java.srcDirs += "$projectDir/build/generated"
}

// gradle clean 시에 QClass 디렉토리 삭제
```

```
clean {
    delete file(generated)
}
```

build.gradle - users

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.5'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

repositories {
    mavenCentral()
}

ext {
    set('springCloudVersion', "2023.0.1")
}

dependencies {

    // spring cloud eureka
    implementation 'org.springframework.cloud:spring-cloud-st

    // spring cloud config
    implementation 'org.springframework.cloud:spring-cloud-st
    implementation 'org.springframework.cloud:spring-cloud-st
```

```
    // feign client
    implementation 'org.springframework.cloud:spring-cloud-st

    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-dev
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-
    testImplementation 'org.springframework.security:spring-s
    implementation 'org.springframework.boot:spring-boot-star
//  implementation 'io.jsonwebtoken:jjwt:0.9.1' //JWT를 위한 라
//  implementation 'javax.xml.bind:jaxb-api:2.3.1' // 사용하는
    runtimeOnly 'io.micrometer:micrometer-registry-prometheus


//  implementation 'io.jsonwebtoken:jjwt:0.12.5'
//  implementation 'jakarta.xml.bind:jakarta.xml.bind-api:4.0

    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.5'

    // s3
    implementation 'com.amazonaws:aws-java-sdk-s3'
    implementation platform('com.amazonaws:aws-java-sdk-bom:1
//    implementation 'io.awspring.cloud:spring-cloud-aws-star
//    implementation 'io.awspring.cloud:spring-cloud-starter-

}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-depe
```

```
    }
}

tasks.named('test') {
    useJUnitPlatform()
}
```

chat-local.yml

```
server:
  port: 0 # Random Port
  servlet:
    context-path: /chat
  baseURL: http://localhost:8000
spring:
  data:
    mongodb:
      uri: mongodb://localhost:27017/piecechatlogs
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/piece?useSSL=false&allow
    username: ssafy
    password: 1234
  jpa:
    hibernate:
      ddl-auto: update
      naming:
        physical-strategy: org.hibernate.boot.model.naming.Ph
    defer-datasource-initialization: true
    properties:
      hibernate.format_sql: true
      dialect: org.hibernate.dialect.MySQL8InnoDBDialect
  application:
    name: chat
eureka:
  client:
    fetch-registry: true
```

```
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8761/eureka
  instance:
    instance-id: ${spring.application.name}:${spring.applicat
    hostname: localhost
    prefer-ip-address: true
management:
  endpoints:
    web:
      exposure:
        include: refresh, health, beans, busrefresh, info, me

logging:
  level:
    org.hibernate.SQL: debug
    org.hibernate.type: trace

# feign client url 세팅
external:
  piece:
    host: ${server.baseURL}/api/piece
  user:
    host: ${server.baseURL}/api/user

# s3
amazon:
  aws:
    access-key: "{cipher}bd13e0d628912dafe4c7e19a40cf35fe84c0
    secret-key: "{cipher}935080aabc0254b56fc200d65d165928f613
    region: ap-northeast-2
    bucket: piecemaker.kr
    prefixAddress: "{cipher}832a154862b127a501866a2f7ff491493
```

build.gradle - chats

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.5'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

repositories {
    mavenCentral()
}

ext {
    set('springCloudVersion', "2023.0.1")
}

dependencies {

    // spring cloud eureka
    implementation 'org.springframework.cloud:spring-cloud-st

    // spring cloud config
    implementation 'org.springframework.cloud:spring-cloud-st
    implementation 'org.springframework.cloud:spring-cloud-st

    // feign client
    implementation 'org.springframework.cloud:spring-cloud-st

    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star
```

```
    // chat
    implementation 'org.springframework.boot:spring-boot-star
    implementation 'org.springframework.boot:spring-boot-star

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-dev
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'

    testImplementation 'org.springframework.boot:spring-boot-
    runtimeOnly 'io.micrometer:micrometer-registry-prometheus
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-depe
    }
}

tasks.named('test') {
    useJUnitPlatform()
}
```

Nginx

default.conf

```
server {
    listen 80;
    server_name piecemaker.kr;

    if ($host = piecemaker.kr) {
        return 301 https://$host$request_uri;
    }

    client_max_body_size 20M;

    return 404;
}
```

```
server {
    listen 443 ssl http2;
    server_name piecemaker.kr;

    ssl_certificate /etc/letsencrypt/live/k10b202.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/k10b202.p.ssafy
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        root /home/opendocs/jenkins/workspace/piece-FE/fronte
        index index.html;
        try_files $uri $uri/ /index.html =404;
    //        proxy_pass https://localhost:3000/;
    //        proxy_set_header Host $http_host;
    //        proxy_set_header X-Real-IP $remote_addr;
    //        proxy_set_header X-Forwarded-For $proxy_add_x_f
    //        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api {
        proxy_pass https://localhost:8000/;

        // 05-16 add
        proxy_set_header X-real-IP $remote_addr;
        proxy_set_header HOST $http_host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward

        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy true;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    // Endpoint for web socket connection
    location /api/chat/wss {
```

```
        proxy_pass http://localhost:8000;  // web socket serv
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

default

```
server {
    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/piecemaker.kr/fullc
    ssl_certificate_key /etc/letsencrypt/live/piecemaker.kr/p
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

    root /home/opendocs/jenkins/workspace/piece-FE/frontend/p

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name piecemaker.kr www.piecemaker.kr; # managed by

    client_max_body_size 20M;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html =404;
    }

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

```
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_ssl_server_name on;

        # websocket
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

server {
    listen 80;
    listen [::]:80;
    server_name piecemaker.kr www.piecemaker.kr;
    if ($host = piecemaker.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    # listen 80;
    # listen [::]:80 ;
    return 404; # managed by Certbot
}
```

5. [CI/CD 구축]

Docker

Dockerfile - Backend

```
# jdk17 image start
FROM openjdk:17
# 인자 정리 - jar
ARG JAR_FILE=build/libs/*.jar
# jar file copy
```

```
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-Dspring.profiles.active=docker", "-jar"
```

docker-compose.yml

```
services:
  db:
    image: mysql:8.0
    container_name: mysql
    restart: always
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: "pieceofcake422!"
      MYSQL_DATABASE: "mysql"
      MYSQL_USER: "pieces"
      MYSQL_PASSWORD: "pieceofcake422!"
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - ./database/datadir/:/var/lib/mysql
      - ./database/init/:/docker_entrypoint-initdb.d/
```

Jenkins

Jenkinsfile - FrontEnd

```
pipeline {
    agent any

    parameters{
```

```
        string(name: 'NAME', defaultValue: 'vue')
    }

    tools {
        nodejs 'Node20'
    }

    stages {
        stage('Vue build'){
            steps{
                dir('frontend/piece'){
                    echo 'Vue build...'
                    sh 'npm install'
                    sh 'CI=false npm run build'
                }
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'good',
                message: "##  :velkoz: #${env.BUILD_NUMBER} 빌
                endpoint: 'https://meeting.ssafy.com/hooks/dk
                channel: 'jenkins202'
//              icon: 'https://www.pngfind.com/pngs/m/437-437
                )
            }
        }

        failure {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'danger',
```

```
                message: "##  :badgun: #${env.BUILD_NUMBER} 빌
                endpoint: 'https://meeting.ssafy.com/hooks/dk
                channel: 'jenkins202'
//              icon: 'https://www.iconspng.com/images/sad-pe
                )
            }
        }
    }
}
```

Jenkinsfile - BackEnd

```
pipeline {
    agent any

    parameters{
        string(name: 'NAME', defaultValue: 'piece')
    }

    stages {
        stage('SCM') {
            steps {
                sh "echo 'SCM...'"
                checkout scm
            }
        }

        stage('Springboot build'){
            steps{
                dir('backend/piece'){
                    sh '''
                    echo 'Springboot build...'
                    chmod +x gradlew
                    ./gradlew clean build -x test
                    '''
                }
```

```
            }
        }

        stage('Dockerimage build'){
            steps {
                dir('backend/piece') {
                    script {
                        def containerExists = sh(script: "doc

                        if (containerExists) {
                            sh "docker stop ${params.NAME}"
                            sh "docker rm ${params.NAME}"
                            sh "docker rmi hyunjinius/springb
                        }


                        sh '''
                        echo 'Dockerimage build...'
                        docker build -t hyunjinius/springboot
                        '''
                    }
                }
            }
        }

        stage('Deploy'){
            steps{
                dir('backend/piece'){
                    sh '''
                    echo 'Deploy BE...'
                    docker run -d --env-file ./.env --name pi
                    '''
                }
            }
        }
    }

    post {
```

```
        success {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'good',
                message: "##  :velkoz: #${env.BUILD_NUMBER} 빌
                endpoint: 'https://meeting.ssafy.com/hooks/dk
                channel: 'jenkins202'
                )
            }
        }

        failure {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'danger',
                message: "##  :badgun: #${env.BUILD_NUMBER} 빌
                endpoint: 'https://meeting.ssafy.com/hooks/dk
                channel: 'jenkins202'
                )
            }
        }
    }
}
```

6. [외부 서비스 사용]

kopis

공연예술통합전산망

예술경영지원센터 운영, 공연 예매 정보 집계 및 DB, 예매상황판, 공연 통계 등 제공.

https://www.kopis.or.kr/por/cs/openapi/openApiUseSend.do?menuId=MNU_00074

키 발급신청 후 piece 프로젝트의 application yml에

```
# secret key
secret:
    kopis:
    api-key: {발급받은 키 입력}
```

위와 같이 발급받은 키 입력

tmdb

Getting Started

Welcome to version 3 of The Movie Database (TMDB) API. This is where you will find the definitive list of currently available methods for our movie, tv, actor and image API.

https://developer.themoviedb.org/reference/intro/getting-started

키 발급신청 후 piece 프로젝트의 application yml에

```
# secret key
secret:
    tmdb:
    api-key: {발급받은 키 입력}
```

위와 같이 발급받은 키 입력

GPT4 & Dalle

https://platform.openai.com/api-keys

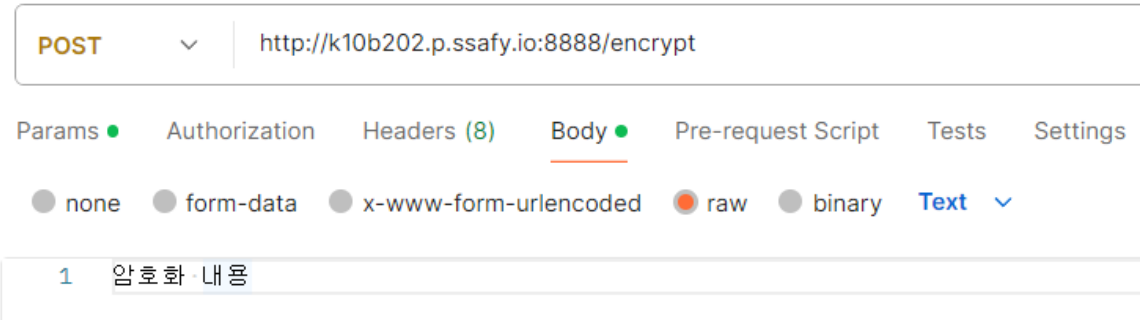
위의 주소에서 API key 발급 후 piece 프로젝트 application.yml에 항목에 맞는 내용 기입

- 프로젝트 사용 내용

- gpt-model : gpt-4-turbo-2024-04-09
- gpt-url : https://api.openai.com/v1/chat/completions
- dalle-model : dall-e-3
- dalle-url : https://api.openai.com/v1/images/generations

```
# OpenAi
openai:
  gpt-model: "{cipher}42b799167ab49394709fbdd89e1e482ad2581e0
  gpt-key: "{cipher}0ca18cce91a3663e556498373a841e255faddc8fa
  gpt-url: "{cipher}351c5cad9df3defdcaccc9606301cffcb5af49d14
  dalle-model: "{cipher}e3cd7af5586a98c89e849a7c10074bd4765ff
  dalle-key: "{cipher}13ae980b4107c720076ba9dce7a348d585ed6b2
  dalle-url: "{cipher}1cb8eb5ca5bb4571b44f0d17f61a3265e8c902e
```

▼ 참고사항



현재 프로젝트에서는 위의 이미지와 같은 방식을 통해 암호화를 한 후, 앞에 {cipher}를 붙여 프로젝트 yml 파일에 기입하였음