👨‍💻

포팅 메뉴얼

2024/10/10 13:45

목차

1. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost, Discord, KaKaoTalk
- 디자인 : Figma
- CI/CD : Jenkins

2. 개발 도구

- Visual Studio Code : 1.94.1
- Intellij : 2024.1.4 (Ultimate Edition)
- MobaXterm

- Postman
- Sourcetree
- MySQL WorkBench

3. 개발 환경

Frontend

React	18.3.1
Zustand	4.5.5
Styled Component	6.1.13

Backend

Java	openjdk version "17.0.11" 2024-04-16
Spring Boot	3.3.3
FastAPI	
Hadoop	
Spark	
Python	

Server

AWS S3	yorijori-bucket
AWS EC2	SSAFY 제공

Service

MySQL	8.0
NginX	1.27.1
Jenkins	2.462.2
Docker	27.2.0
Docker Compose	2.20
Dockerhub	dockerhub/parkgc0504

4. 환경변수 형태

- Backend
    - nginx
        - ngnix.conf

```
server{

    listen 80;
    server_name j11c206.p.ssafy.io;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl;
    server_name j11c206.p.ssafy.io;
    ssl_certificate  /etc/letsencrypt/live/j11c206.p
    ssl_certificate_key /etc/letsencrypt/live/j11c20
     location / {
        proxy_pass http://frontend:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api/v1 {
        proxy_pass http://apigateway-service:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

    - fastapi
        - .env

```
DATABASE_URL=mysql+pymysql://root:*****@j11c206.p.ss

PRICE_CERT_KEY=************
PRICE_CERT_ID=****
PRICE_API_URL=http://www.kamis.or.kr/service/price/x
```

- user_service(네이버 OAuth + S3)
  - application.yml

```
# application.yml

server:
  port: 8081

spring:
  application:
    name: user-service
  config:
    import: application-secrets.yml

  datasource:
    url: jdbc:mysql://j11c206.p.ssafy.io:3306/yorijo
    username: ******
    password: ******
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
        format_sql: true
        default_batch_fetch_size: 500
        naming:
          physical-strategy: org.hibernate.boot.mode
  security:
```

```yaml
    oauth2:
      client:
        registration:
          naver:
            client-id: ${naver.client-id} # applicat
            client-secret: ${naver.client-secret} #
            redirect-uri: http://j11c206.p.ssafy.io:
            authorization-grant-type: authorization_
            client-name: Naver
            scope: id, name, email, profile_image, b
        provider:
          naver:
            authorization-uri: https://nid.naver.com
            token-uri: https://nid.naver.com/oauth2.
            user-info-uri: https://openapi.naver.com
            user-name-attribute: response

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring
    prefer-ip-address: true
    ip-address: j11c206.p.ssafy.io

jwt:
  secretKey: ${jwt.secretKey} # application-secrets.
  access:
    expiration: 86400000
    header: Authorization

  refresh:
    expiration: 1209600000
    header: Authorization-refresh



cloud:
  aws:
    s3:
      bucket: yorijori-bucket
    stack:
      auto: false
    region:
      static: ap-northeast-2
    credentials:
      accessKey: ************
      secretKey: ************
```

- application-secrets.yml

```
jwt:
  secretKey: "**************"

naver:
  client-id: ***********
  client-secret: ***********
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.3'
    id 'io.spring.dependency-management' version '1.
    id 'org.asciidoctor.jvm.convert' version '3.3.2'
}

group = 'com.recipe'
version = '1.0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

ext {
    set('snippetsDir', file("build/generated-snippet
    set('springCloudVersion', "2023.0.3")
}

dependencies {
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-

    // ModelMapper
    implementation 'org.modelmapper:modelmapper:3.2.

    // Spring Cloud Config Client
    implementation 'org.springframework.cloud:spring

    // Spring Cloud Bootstrap
    implementation 'org.springframework.cloud:spring

    // Spring Boot Actuator
    implementation 'org.springframework.boot:spring-

    // Spring Cloud Bus with AMQP (RabbitMQ)
    implementation 'org.springframework.cloud:spring

    // OpenFeign for declarative REST client
    implementation 'org.springframework.cloud:spring

    // Eureka Client for Spring Cloud Netflix
    implementation 'org.springframework.cloud:spring

    // S3
    implementation 'org.springframework.cloud:spring


    runtimeOnly 'com.mysql:mysql-connector-j'
    implementation 'com.auth0:java-jwt:4.2.1'

    implementation 'org.apache.commons:commons-lang3

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spr
    testImplementation 'org.springframework.restdocs
    testRuntimeOnly 'org.junit.platform:junit-platfo
}

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

tasks.named('test') {
    outputs.dir snippetsDir
    useJUnitPlatform()
}

tasks.named('asciidoctor') {
    inputs.dir snippetsDir
    dependsOn test
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar UserService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "UserService.jar"]
```

- Jenkinsfile

```
pipeline {
    agent any


    environment {
        BACKEND_DIR = './user_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/user-serv
        CONFIG_FILE_PATH = '/home/backend_config/app
        APPLICATION_YML_PATH = '/home/backend_config
        TARGET_DIR = 'user_service/src/main/resource
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }

        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "user_service 빌드 시
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }

        stage('Copy config files') {
            steps {
                script {
                    def workspaceDir = pwd()
                    sh "cp ${CONFIG_FILE_PATH} ${wor
                    sh "cp ${APPLICATION_YML_PATH} $
                }
            }
        }

        stage('Build user_service') {
            when {
                changeset "user_service/**"  // back
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }

        stage('Docker Build and Push user_service to
            when {
                changeset "user_service/**"  // back
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }


        stage('Deploy Locally') {
            when {
                changeset "user_service/**"  // back
            }
            steps {
                script {
                    // Stop and remove any running u
                    sh 'docker stop user-service ||

                    // Run the new user-service cont
                    sh '''
                    docker run -d -p 8081:8081 --net
                    -e spring.cloud.config.uri=http:
                    -e spring.rabbitmq.host=rabbitmq
                    -e eureka.client.registerWithEur
                    -e eureka.client.fetchRegistry=t
                    -e eureka.client.serviceUrl.defa
                    -e logging.file=/api-logs/users-
                    ${DOCKER_IMAGE_BACKEND}
                    '''
                }
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'good',
                    message: "user_service 빌드 성공:
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'danger',
                    message: "user_service 빌드 실패:
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
    }
}
```

- social_service
    - application.yml

```
server:
  port: 8083

spring:
  application:
    name: social-service
  datasource:
    url: jdbc:mysql://j11c206.p.ssafy.io:3306/yorijo
    username: *****
```

```yaml
      password: *****
    jpa:
      hibernate:
        ddl-auto: update
      show-sql: true
      generate-ddl: true

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring
    prefer-ip-address: true
    ip-address: j11c206.p.ssafy.io


logging:
  level:
    com.recipe.catalogservice: DEBUG
```

- Jenkinsfile

```groovy
pipeline {
    agent any

    environment {
        BACKEND_DIR = './social_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/social-se
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }
        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "social-service 빌드
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }

        stage('Build social-service') {
            when {
                changeset "social_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }

        stage('Docker Build and Push social_service
            when {
                changeset "social_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy Locally') {
             when {
                   changeset "social_service/**"  //
             }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        // Stop and remove any runni
                        sh 'docker stop social-servi

                        // Run the new social-servic
                        sh '''
                        docker run -d -p 8083:8083 \
                        --network backendnet \
                        --name social-service \
                        -e "eureka.client.registerWi
                        -e "eureka.client.fetchRegis
                        -e "eureka.client.serviceUrl
                        -e "logging.file=/api-logs/s
                        parkgc0504/social-service:la
                        '''
                    }
                }
            }
        }
    }


     post {
        success {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'good',
                    message: "social_service 빌드 성공
```

```
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'danger',
                    message: "social_service 빌드 실패
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
    }
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar SocialService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "SocialService.jar"]
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.recipe'
version = '1,0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

repositories {
    mavenCentral()
}

ext {
    set('snippetsDir', file("build/generated-snippet
    set('springCloudVersion', "2023.0.3")
}

dependencies {
    implementation 'org.springframework.boot:spring-
//  implementation 'org.springframework.boot:spring-
//  implementation 'org.springframework.boot:spring-
//  implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-

    // ModelMapper
    implementation 'org.modelmapper:modelmapper:3.2.

    // Spring Cloud Config Client
    implementation 'org.springframework.cloud:spring

    // Spring Cloud Bootstrap
    implementation 'org.springframework.cloud:spring

    // Spring Boot Actuator
    implementation 'org.springframework.boot:spring-

    // Spring Cloud Bus with AMQP (RabbitMQ)
    implementation 'org.springframework.cloud:spring

    // OpenFeign for declarative REST client
    implementation 'org.springframework.cloud:spring

    // Eureka Client for Spring Cloud Netflix
    implementation 'org.springframework.cloud:spring


    runtimeOnly 'com.mysql:mysql-connector-j'
    implementation 'com.auth0:java-jwt:4.2.1'

    implementation 'org.apache.commons:commons-lang3

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spr
    testImplementation 'org.springframework.restdocs
    testRuntimeOnly 'org.junit.platform:junit-platfo
}

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

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

- ingredient_service
    - application.yml

```
server:
  port: 8084

spring:
  application:
    name: ingredient-service
  datasource:
    url: jdbc:mysql://j11c206.p.ssafy.io:3306/yorijo
    username: *******
    password: *******
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring
    prefer-ip-address: true
    ip-address: j11c206.p.ssafy.io


logging:
  level:
    com.recipe.catalogservice: DEBUG

ocr:
  api:
    key: w8faWtah0sxEHtDEV7MI
    secret: 665FR9208F
```

- Jenkinsfile

```
pipeline {
    agent any

    environment {
        BACKEND_DIR = './ingredient_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/ingredien
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }
        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "ingredient_service
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }

        stage('Build ingredient_service') {
            when {
                changeset "ingredient_service/**"  /
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }
        stage('Docker Build and Push ingredient_serv
            when {
                changeset "ingredient_service/**"  /
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy Locally') {
             when {
                    changeset "ingredient_service/**"
             }
             steps {
                dir(BACKEND_DIR) {
                    script {
                        // Stop and remove any runni
                        sh 'docker stop ingredient-s

                        // Run the new ingredient-se
                        sh '''
                        docker run -d -p 8084:8084 \
                        --network backendnet \
                        --name ingredient-service \
                        -e "eureka.client.registerWi
                        -e "eureka.client.fetchRegis
                        -e "eureka.client.serviceUrl
                        -e "spring.rabbitmq.host=j11
                        -e "logging.file=/api-logs/i
                        parkgc0504/ingredient-servic
            '''
          }
        }
      }
    }
  }


  post {
    success {
      script {
        def Author_ID = sh(script: "git show
        def Author_Name = sh(script: "git sh
        mattermostSend(color: 'good',
          message: "ingredient_service 빌드
          endpoint: 'https://meeting.ssafy
          channel: 'C206_deploy'
        )
      }
    }
    failure {
      script {
        def Author_ID = sh(script: "git show
        def Author_Name = sh(script: "git sh
        mattermostSend(color: 'danger',
          message: "ingredient_service 빌드
          endpoint: 'https://meeting.ssafy
          channel: 'C206_deploy'
        )
      }
    }
  }
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk
```

```
# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar IngredientService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "IngredientService.jar"]
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.recipe'
version = '1.0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

repositories {
    mavenCentral()
}

ext {
    set('snippetsDir', file("build/generated-snippet
    set('springCloudVersion', "2023.0.3")
}

dependencies {
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-

    // ModelMapper
    implementation 'org.modelmapper:modelmapper:3.2.

    // Spring Cloud Config Client
    implementation 'org.springframework.cloud:spring

    // Spring Cloud Bootstrap
    implementation 'org.springframework.cloud:spring

    // Spring Boot Actuator
    implementation 'org.springframework.boot:spring-

    // Spring Cloud Bus with AMQP (RabbitMQ)
    implementation 'org.springframework.cloud:spring

    // OpenFeign for declarative REST client
    implementation 'org.springframework.cloud:spring

    // Eureka Client for Spring Cloud Netflix
    implementation 'org.springframework.cloud:spring

    // Jaro-Winkler 유사도 라이브러리
    implementation 'info.debatty:java-string-similar


    runtimeOnly 'com.mysql:mysql-connector-j'
    implementation 'com.auth0:java-jwt:4.2.1'

    implementation 'org.apache.commons:commons-lang3

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spr
    testImplementation 'org.springframework.restdocs
    testRuntimeOnly 'org.junit.platform:junit-platfo
```

```
}

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

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

- config_service
  - application.yml

```
server:
  port: 8888

spring:
  application:
    name: config-service
  rabbitmq:
    host: 127.0.0.1
    port: 5672
    username: ****
    password: ****
  cloud:
    config:
      server:
        git:
          uri: https://github.com/Nbowow/spring-clou
#           username: [your username] -> repository p
#           password: [your password]
#           uri: file://C:/Users/SSAFY/Desktop/Nam/Sp

management:
  endpoints:
    web:
      exposure:
        include: health, busrefresh
```

- Jenkinsfile

```
pipeline {
    agent any

    environment {
        BACKEND_DIR = './config_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/config-se
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }

        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "config_service 빌드
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }
        stage('Build config_service') {
            when {
                changeset "config_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }

        stage('Docker Build and Push config_service
            when {
                changeset "config_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy Locally') {
            when {
                changeset "config_service/**"  // ba
            }
            steps {
                script {
                    // 기존 컨테이너 중지 및 삭제
                    sh 'docker stop config-service |
                    sh 'docker rm config-service ||
                    // 새로운 컨테이너 실행
                    sh '''
                    docker run -d -p 8888:8888 --net
                    -e "spring.rabbitmq.host=rabbitm
                    -e "spring.profiles.active=defau
                    --name config-service parkgc0504
                    '''
                }
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'good',
                    message: "config_service 빌드 성공
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'danger',
                    message: "config_service 빌드 실패
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
    }
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar ConfigService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "ConfigService.jar"]
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.recipe'
version = '1.0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

repositories {
    mavenCentral()
}

ext {
    set('springCloudVersion', "2023.0.3")
}
```

```
dependencies {
    implementation 'org.springframework.cloud:spring
//  implementation 'org.springframework.cloud:spring
    implementation 'org.springframework.cloud:spring
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.cloud:spring

    testImplementation 'org.springframework.boot:spr
    testRuntimeOnly 'org.junit.platform:junit-platfo
}

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

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

- apigateway_service
  - application.yml

```
server:
  port: 8000

spring:
  application:
    name: apigateway-service
  cloud:
    gateway:
      default-filters:
        - name: GlobalFilter
          args:
            baseMessage: Spring Cloud Gateway Global
            preLogger: true
```

```
              postLogger: true
        routes:
          - id: user-service
            uri: lb://USER-SERVICE
            predicates:
              - Path=/user-service/users
              - Method=POST
            filters:
              - RemoveRequestHeader=Cookie
              - RewritePath=/user-service/(?<segment>.

          - id: user-service
            uri: lb://USER-SERVICE
            predicates:
              - Path=/api/v1/users/**

          - id: recipe-service
            uri: lb://RECIPE-SERVICE
            predicates:
              - Path=/api/v1/recipe/**

          - id: social-service
            uri: lb://SOCIAL-SERVICE
            predicates:
              - Path=/api/v1/social/**

          - id: ingredient-service
            uri: lb://INGREDIENT-SERVICE
            predicates:
              - Path=/api/v1/ingredient/**




management:
  endpoints:
    web:
      exposure:
        include: refresh, health, beans, httpexchang
```

- Jenkinsfile

```
pipeline {
    agent any

    environment {
        BACKEND_DIR = './apigateway_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/apigatewa
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }

        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "apigateway_service
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }

        stage('Build apigateway_service') {
            when {
                changeset "apigateway_service/**"  /
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }

        stage('Docker Build and Push apigateway_serv
            when {
                changeset "apigateway_service/**"  /
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy Locally') {
            when {
                changeset "apigateway_service/**"  /
            }
            steps {
                script {
                    // 기존 컨테이너 중지 및 삭제
                    sh 'docker stop apigateway-servi
                    sh 'docker rm apigateway-service

                    // 새로운 컨테이너 실행
                    sh '''
                    docker run -d -p 8000:8000 --net
                    -e "spring.cloud.config.uri=http
                    -e "spring.rabbitmq.host=rabbitm
                    -e "eureka.client.registerWithEu
                    -e "eureka.client.fetchRegistry=
                    -e "eureka.client.serviceUrl.def
                    --name apigateway-service \
                    parkgc0504/apigateway-service:la
                    '''
                }
            }
        }


    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'good',
                    message: "apigateway_service 빌드
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'danger',
                    message: "apigateway_service 빌드
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
    }
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar ApigatewayService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "ApigatewayService.jar"]
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.recipe'
version = '1.0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

repositories {
    mavenCentral()
}
```

```
ext {
    set('springCloudVersion', "2023.0.3")
}

dependencies {
    implementation 'org.springframework.cloud:spring
    implementation 'org.springframework.cloud:spring
    implementation 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

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

    implementation 'javax.xml.bind:jaxb-api:2.3.1'
    implementation 'org.springframework.cloud:spring
    implementation 'org.springframework.cloud:spring
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.cloud:spring

    implementation 'org.springframework.boot:spring-
    testImplementation 'org.springframework.boot:spr
    testRuntimeOnly 'org.junit.platform:junit-platfo
}

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

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

- recipe_service
  - application.yml

```yaml
server:
  port: 8082

spring:
  application:
    name: recipe-service
  datasource:
    url: jdbc:mysql://j11c206.p.ssafy.io:3306/yorijo
    username: ******
    password: ******
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
        format_sql: true
        default_batch_fetch_size: 500
        naming:
          physical-strategy: org.hibernate.boot.mode
          implicit-strategy: org.hibernate.boot.mode

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring
    prefer-ip-address: true
    ip-address: j11c206.p.ssafy.io


logging:
  level:
    com.recipe.catalogservice: DEBUG

cloud:
  aws:
    s3:
      bucket: yorijori-bucket
    stack:
      auto: false
    region:
      static: ap-northeast-2
    credentials:
      accessKey: *******
      secretKey: *******
```

- Jenkinsfile

```
pipeline {
    agent any

    environment {
        BACKEND_DIR = './recipe_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/recipe-se
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }

        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "recipe-service 빌드
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }

        stage('Build recipe-service') {
            when {
                changeset "recipe_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }

        stage('Docker Build and Push recipe_service
            when {
                changeset "recipe_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy Locally') {
            when {
                changeset "recipe_service/**"  // ba
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        // Stop and remove any runni
                        sh 'docker stop recipe-servi

                        // Run the new recipe-servic
                        sh '''
                        docker run -d -p 8082:8082 \
                        --network backendnet \
                        --name recipe-service \
                        -e "eureka.client.registerWi
                        -e "eureka.client.fetchRegis
                        -e "eureka.client.serviceUrl
                        -e "logging.file=/api-logs/r
                        parkgc0504/recipe-service:la
                        '''
                    }
                }
            }
        }
    }


     post {
        success {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'good',
                    message: "recipe_service 빌드 성공
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'danger',
                    message: "recipe_service 빌드 실패
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
    }
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar RecipeService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "RecipeService.jar"]
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.recipe'
version = '1.0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
```

```
}

repositories {
    mavenCentral()
}

ext {
    set('springCloudVersion', "2023.0.3")
}


dependencies {
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.boot:spring-
    implementation 'org.springframework.cloud:spring
    implementation 'org.modelmapper:modelmapper:3.2.

    implementation 'org.springframework.cloud:spring

    // OpenFeign for declarative REST client
    implementation 'org.springframework.cloud:spring

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spr
    testRuntimeOnly 'org.junit.platform:junit-platfo
}

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

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

- discovery_service
  - application.yml

```
server:
  port: 8761

spring:
  application:
    name: discoveryservice

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
```

  - Jenkinsfile

```
pipeline {
    agent any

    environment {
        BACKEND_DIR = './discovery_service'
        DOCKER_IMAGE_BACKEND = 'parkgc0504/discovery
        GITLAB_CREDENTIALSID = 'gichangssafy'
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentia
    }

    stages {
        stage('Checkout') {
            steps {
                git credentialsId: GITLAB_CREDENTIAL
            }
        }

        stage('Notify Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git
                    def Author_Name = sh(script: "gi
                    mattermostSend(
                        color: 'warning',
                        message: "discovery_service
                        endpoint: 'https://meeting.s
                        channel: 'C206_deploy'
                    )
                }
            }
        }


        stage('Build discovery_service') {
            when {
                changeset "discovery_service/**"  //
            }
            steps {
                dir(BACKEND_DIR) {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                }
            }
        }

        stage('Docker Build and Push discovery_servi
            when {
                changeset "discovery_service/**"  //
            }
            steps {
                dir(BACKEND_DIR) {
                    script {
                        docker.withRegistry('https:/
                            def backendImage = docke
                            backendImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy Locally') {
            when {
                changeset "discovery_service/**"  //
            }
            steps {
                script {
                    // 기존 컨테이너 중지 및 삭제
                    sh 'docker stop discovery-servic
                    sh 'docker rm discovery-service

                    // 새로운 컨테이너 실행
                    sh '''
                    docker run -d -p 8761:8761 --net
                    -e "spring.cloud.config.uri=http
                    --name discovery-service parkgc0
                    '''
                }
            }
        }

    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'good',
                    message: "discovery_service 빌드
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show
                def Author_Name = sh(script: "git sh
                mattermostSend(color: 'danger',
                    message: "discovery_service 빌드
                    endpoint: 'https://meeting.ssafy
                    channel: 'C206_deploy'
                )
            }
        }
    }
}
```

- Dockerfile

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17.0.1-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file to the container
COPY build/libs/*.jar DiscoveryService.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "DiscoveryService.jar"]
```

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.
}

group = 'com.recipe'
```

```
version = '1.0'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

repositories {
    mavenCentral()
}

ext {
    set('springCloudVersion', "2023.0.3")
}

dependencies {
    implementation 'org.springframework.cloud:spring
    implementation 'org.springframework.boot:spring-
    testImplementation 'org.springframework.boot:spr
    testRuntimeOnly 'org.junit.platform:junit-platfo
}

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

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

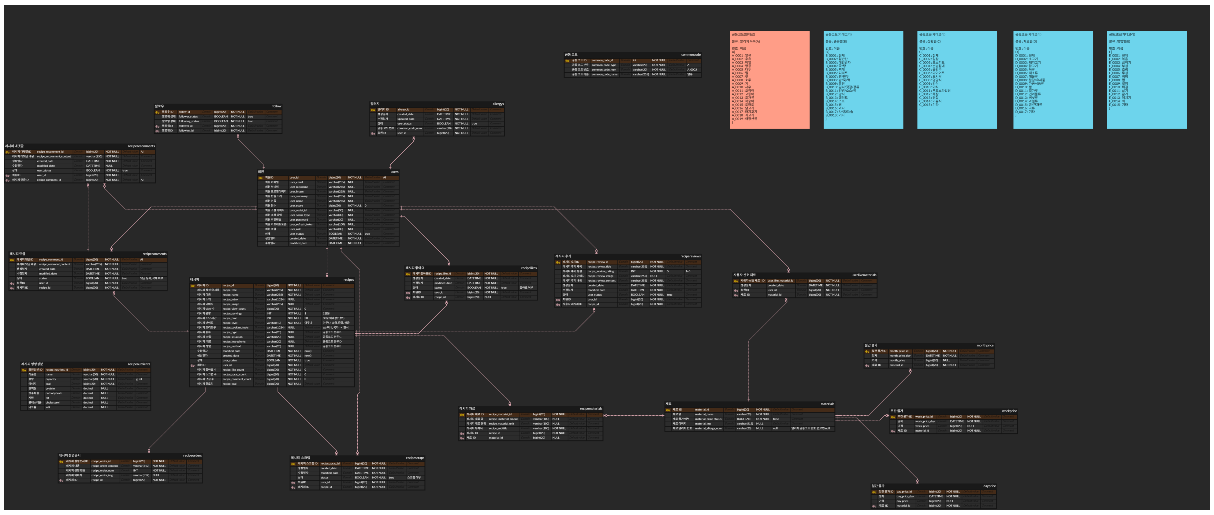
5. 포트 정보

서비스 및 포트 구성

서비스	컨테이너 이름	호스트 포트	컨테이너 포트
Recipe Service	recipe-service	8082	8082
Social Service	social-service	8083	8083
Frontend	frontend	3000	3000
User Service	user-service	8081	8081
Ingredient Service	ingredient-service	8084	8084
Jenkins	jenkins	8080	8080
Nginx	nginx	80, 443	80, 443
API Gateway Service	apigateway-service	8000	8000
Discovery Service	discovery-service	8761	8761
Config Service	config-service	8888	8888
MySQL	mysql	3306	3306
Redis	redis	6379	6379
RabbitMQ	rabbitmq	4369, 5671-5672, 15671-15672	4369, 5671-5672, 15671-15672

6. 산출물

ERD



UCC

PPT

7. 더미데이터

7. 시연시나리오