포팅 매뉴얼

🛠️Stack

IDE

`IntelliJ(2023.3)` `Android Studio(Android Studio Iguana)`

Infra

`Jenkins` `Docker` `Docker-compose` `kafka(Zookeeper)`

Android

`Kotlin(1.9.0)` `Gradle(8.4)` `Jetpack Compose`

Backend

`Java(17)` `Springboot(3.2.5)` `SpringSecurity` `r2dbc` `netty` `springcloud(4.1.1)`

AI

`Python`

Database

`mysql` `redis` `ElasticSearch` `mongodb`

🛠️Build

Android

💡 안드로이드에서 로그인을 하기 위해서는 사전에 서명이 등록된 애플리케이션이어야 합니다.

GitLab에 미리 빌드된 APK를 사용해주시길 바랍니다.

- debug

  ```
  Android Studio > Build > Build Bundle(s) / APK(s) > Build APK
  ```

- release

  ```
  Android Studio > Build > Generate Signed Bundle / APK
  ```

🛠️Deploy

Backend

```
docker-compose up -d
```

엘라스틱 서치

```
version: '7.9'

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.9.3
    container_name: elasticsearch
    environment:
      - node.name=es01
      - cluster.name=es-docker-cluster
      - discovery.type=single-node
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - esdata:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
    networks:
      - elastic

  kibana:
    image: docker.elastic.co/kibana/kibana:7.9.3
    container_name: kibana
    environment:
      ELASTICSEARCH_URL: http://elasticsearch:9200
    ports:
      - 5601:5601
    networks:
      - elastic

volumes:
  esdata:
    driver: local

networks:
  elastic:
    driver: bridge
```

데이터 추가 가이드

```
upload csv data file
[Kibana] {elasticsearch 주소}:5601 접속 > Machine Learning > Data
> csv파일 선택
```

백엔드 환경 구성을 위한 Docker compose

```
version: '3'

services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2181:2181"
    networks:
      - infra
  kafka:
    container_name: kafka
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_LISTENERS: INSIDE://kafka:9092,OUTSIDE://
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLAINTEXT,OU
      KAFKA_LISTENERS: INSIDE://0.0.0.0:9092,OUTSIDE://0.0.0.0:
      KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - zookeeper
    networks:
      - infra

  nginx-proxy:
    image: jwilder/nginx-proxy
    container_name: nginx-proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./docker_volumes/nginx/certs:/etc/nginx/certs:rw
      - ./docker_volumes/nginx/html:/usr/share/nginx/html:rw
      - ./docker_volumes/nginx/vhost.d:/etc/nginx/vhost.d:rw
      - /var/run/docker.sock:/tmp/docker.sock:ro
    networks:
      - infra

  letsencrypt:
    image: jrcs/letsencrypt-nginx-proxy-companion
    container_name: letsencrypt
    environment:
      - NGINX_PROXY_CONTAINER=nginx-proxy
    depends_on:
      - nginx-proxy
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./docker_volumes/nginx/certs:/etc/nginx/certs:rw
      - ./docker_volumes/nginx/vhost.d:/etc/nginx/vhost.d:rw
      - ./docker_volumes/nginx/html:/usr/share/nginx/html:rw
      - ./docker_volumes/acme.sh:/etc/acme.sh
    networks:
      - infra

  host-nginx:
    image: nginx
    container_name: host-nginx
    expose:
      - "80"
    environment:
      - VIRTUAL_HOST=
      - LETSENCRYPT_HOST=
      - LETSENCRYPT_EMAIL=
    networks:
      - infra

  jenkins:
    image: jmg97/custom-jenkins:latest
    container_name: jenkins
    environment:
      - VIRTUAL_PORT=8080
      - JENKINS_OPTS="--prefix=/jenkins"
    volumes:
      - ./docker_volumes/jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - infra

  mysql:
    image: mysql:latest
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: ""
    volumes:
      - ./docker_volumes/mysql:/var/lib/mysql
    ports:
      - "3306:3306"
    networks:
      - infra

  redis:
    image: redis:latest
    container_name: redis
    networks:
      - infra

  mongo:
    image: mongo
    container_name: mongo
    restart: always
    volumes:
      - mongo-data:/data/db
    networks:
      - infra

networks:
  infra:
    external: true

volumes:
  certs:
    driver: local
    driver_opts:
      type: none
      device: $PWD/docker_volumes/nginx/certs
      o: bind
  vhostd:
    driver: local
    driver_opts:
      type: none
      device: $PWD/docker_volumes/nginx/vhost.d
      o: bind
  html:
    driver: local
    driver_opts:
      type: none
      device: $PWD/docker_volumes/nginx/html
      o: bind
  jenkins_home:
    driver: local
    driver_opts:
      type: none
      device: $PWD/docker_volumes/jenkins
      o: bind
  mysql_data:
    driver: local
    driver_opts:
      type: none
      device: $PWD/docker_volumes/mysql
      o: bind
  mongo-data:
    driver: local
```

> ### Android

- 안드로이드 파일 위치: `exec/mealtoyou.apk`

🛠️Environment Variables

💡 기본적으로 Vault 를 통한 환경 변수 주입

Vault

alarm-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "fcm": {
    "project-id": ${FCM_PROJECT_ID},
    "service-account-file": {
      "auth_provider_x509_cert_url": ${FCM_AUTH_PROVIDER_CERT_UI
      "auth_uri": ${FCM_AUTH_URI},
      "client_email": ${FCM_CLIENT_EMAIL},
      "client_id": ${FCM_CLIENT_ID},
      "client_x509_cert_url": ${FCM_CLIENT_CERT_URL},
      "private_key": ${FCM_PRIVATE_KEY},
      "private_key_id": ${FCM_PRIVATE_KEY_ID},
      "project_id": ${FCM_PROJECT_ID},
      "token_uri": ${FCM_TOKEN_URI},
      "type": ${FCM_TYPE},
      "universe_domain": ${FCM_UNIVERSE_DOMAIN}
    },
    "topic-name": ${FCM_TOPIC_NAME}
  },
  "google": {
    "clientId": ${GOOGLE_CLIENT_ID}
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "openai": {
    "apikey": ${OPENAI_APIKEY},
    "use": true
  },
  "server": {
    "port": ${ALARM_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "user-service"
    },
    "cloud": {
      "aws": {
        "credentials": {
          "access-key": ${AWS_ACCESS_KEY},
          "secret-key": ${AWS_SECRET_KEY}
        },
        "s3": {
          "bucket": ${AWS_S3_BUCKET}
        }
      }
    },
    "data": {
      "redis": {
        "host": ${REDIS_HOST},
        "port": ${REDIST_PROT}
      }
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "user-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serialization
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
    "r2dbc": {
      "password": ${R2DBC_PASSWORD},
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    },
    "security": {
      "oauth2": {
        "client": {
          "registration": {
            "google": {
              "client-id": ${OAUTH_GOOGLE_CLIENTID},
              "client-secret": ${OAUTH_CLIENT_SECRET},
              "redirect-uri": ${OAUTH_REDIRECT_URI},
              "scope": [
                "profile",
                "email"
              ]
            }
          }
        }
      }
    }
  }
}
```

chatting-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "server": {
    "port": ${CHATTING_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "chatting-service"
    },
    "data": {
      "mongodb": {
        "uri": ${MONGODB_URI}
      }
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "chatting-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    }
  }
}
```

community-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "server": {
    "port": ${COMMUNITY_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "community-service"
    },
    "data": {
      "redis": {
        "host": ${REDIS_HOST},
        "port": ${REDIS_PORT}
      }
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "community-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serialization
        "value-serializer": "org.apache.kafka.common.serializati
      }
    },
    "r2dbc": {
      "password": ${R2DBC_PASSWORD},
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    }
  }
}
```

diet-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "server": {
    "port": ${DIET_SERVER_PORT}
  },
  "spring": {
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "diet-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
    "r2dbc": {
      "password": ${R2DBC_PASSWORD},
      "pool": {
        "initial-size": 10,
        "max-idle-time": "30m",
        "max-size": 20
      },
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    }
  }
}
```

eureka

```
{
  "eureka": {
    "client": {
      "fetchRegistry": false,
      "registerWithEureka": false,
      "serviceUrl": {
        "defaultZone": "http://${eureka.instance.hostname}:${se
      }
    },
    "instance": {
      "hostname": ${HOSTNAME}
    },
    "server": {
      "enableSelfPreservation": false,
      "evictionIntervalTimerInMs": 5000
    }
  },
  "server": {
    "port": ${EUREKA_SERVER_PORT}
  }
}
```

food-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "server": {
    "port": ${FOOD_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "food-service"
    },
    "data": {
      "elasticsearch": {
        "password": ${ELASTICSEARCH_PASSWORD},
        "repositories": {
          "enabled": true
        },
        "url": ${ELASTICSEARCH_URL},
        "username": ${ELASTICSEARCH_USERNAME}
      }
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "food-group"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
    "r2dbc": {
     "password": ${R2DBC_PASSWORD},
     "pool": {
        "initial-size": 10,
        "max-idle-time": "30m",
        "max-size": 20
      },
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    }
  }
}
```

gateway

```
{
  "eureka": {
    "client": {
      "fetchRegistry": true,
      "registerWithEureka": true,
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "server": {
    "port": ${GATEWAY_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "gateway"
    },
    "cloud": {
      "gateway": {
        "discovery": {
          "locator": {
            "enabled": true,
            "lower-case-service-id": true
          }
        },
        "routes": [
          {
            "filters": [
              "RewritePath=/api/foods/(?<segment>.*), /${segmen
            ],
            "id": "food-service",
            "predicates": [
              "Path=/api/foods/**"
            ],
            "uri": "lb://food-service"
          },
          {
            "filters": [
              "RewritePath=/api/diets/(?<segment>.*), /${segmen
            ],
            "id": "food-service",
            "predicates": [
              "Path=/api/diets/**"
            ],
            "uri": "lb://food-service"
          }
        ]
      }
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "group-chat"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
    "profiles": {
      "active": "dev"
    }
  }
}
```

health-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "server": {
    "port": ${HEALTH_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "health-service"
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "health-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
    "r2dbc": {
      "password": ${R2DBC_PASSWORD},
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    }
  }
}
```

supplement-service

```
{
  "eureka": {
    "client": {
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "jwt": {
    "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "server": {
    "port": ${SUPPLEMENT_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "supplement-service"
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "supplement-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
     "r2dbc": {
      "password": ${R2DBC_PASSWORD},
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    }
  }
}
```

user-service

```
  {
    "eureka": {
      "client": {
```

```
      "serviceUrl": {
        "defaultZone": ${EUREKA_CLIENT_SERVICEURL}
      }
    },
    "instance": {
      "preferIpAddress": true
    }
  },
  "fcm": {
   "project-id": ${FCM_PROJECT_ID},
    "service-account-file": {
      "auth_provider_x509_cert_url": ${FCM_AUTH_PROVIDER_CERT_UR
      "auth_uri": ${FCM_AUTH_URI},
      "client_email": ${FCM_CLIENT_EMAIL},
      "client_id": ${FCM_CLIENT_ID},
      "client_x509_cert_url": ${FCM_CLIENT_CERT_URL},
      "private_key": ${FCM_PRIVATE_KEY},
      "private_key_id": ${FCM_PRIVATE_KEY_ID},
      "project_id": ${FCM_PROJECT_ID},
      "token_uri": ${FCM_TOKEN_URI},
      "type": ${FCM_TYPE},
      "universe_domain": ${FCM_UNIVERSE_DOMAIN}
    },
    "topic-name": ${FCM_TOPIC_NAME}
  },
  "google": {
    "clientId": ${GOOGLE_CLIENT_ID}
  },
  "jwt": {
     "secret": ${JWT_SECRET}
  },
  "logging": {
    "level": {
      "org.springframework.r2dbc.core": "debug"
    }
  },
  "openai": {
    "apikey": ${OPENAI_APIKEY},
    "use": true
  },
  "server": {
    "port": ${USER_SERVER_PORT}
  },
  "spring": {
    "application": {
      "name": "user-service"
    },
    "cloud": {
      "aws": {
        "credentials": {
          "access-key": ${AWS_ACCESS_KEY},
          "secret-key": ${AWS_SECRET_KEY}
        },
        "s3": {
          "bucket": ${AWS_S3_BUCKET}
        }
      }
    },
    "data": {
      "redis": {
        "host": ${REDIS_HOST},
        "port": ${REDIST_PROT}
      }
    },
    "kafka": {
      "bootstrap-servers": ${KAFKA_BOOTSTRAP_SERVER},
      "consumer": {
        "auto-offset-reset": "earliest",
        "group-id": "user-service"
      },
      "producer": {
        "key-serializer": "org.apache.kafka.common.serializatio
        "value-serializer": "org.apache.kafka.common.serializat
      }
    },
    "r2dbc": {
      "password": ${R2DBC_PASSWORD},
      "url": ${R2DBC_URL},
      "username": ${R2DBC_USERNAME}
    },
    "security": {
      "oauth2": {
        "client": {
          "registration": {
            "google": {
              "client-id": ${OAUTH_GOOGLE_CLIENTID},
              "client-secret": ${OAUTH_CLIENT_SECRET},
              "redirect-uri": ${OAUTH_REDIRECT_URI},
              "scope": [
                "profile",
                "email"
              ]
            }
          }
        }
      }
    }
  }
}
```

Backend

application.yml

> config

```yaml
server:
  port: ${SERVER_PORT}

spring:
  application:
    name: config-server
  profiles:
    active: vault
  cloud:
    config:
      server:
        vault:
          host: ${VAULT_HOST}
          port: 443
          scheme: https
          token: ${VAULT_TOKEN}
          authentication: TOKEN
          kv-version: 2
          backend: ${BACKEND}
          profile-separator: '/'
    stream:
      kafka:
        binder:
          brokers: ${BROKER}
      bindings:
        springCloudBusInput:
          destination: springCloudBus
          group: springCloudBus
          consumer:
            max-attempts: 1
        springCloudBusOutput:
          destination: springCloudBus
          producer:
            required-groups: springCloudBus
  kafka:
    listener:
      concurrency: 6

management:
  endpoints:
    web:
      exposure:
        include: '*'
logging:
  level:
    root: INFO
    kafka: trace
```

eureka

```
spring:
  application:
    name: eureka
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      discovery:
        enabled: false
  security:
    user:
      name: ${EUREKA_NAME}
      password: ${EUREKA_PASSWORD}
```

gateway

```
spring:
  application:
    name: gateway
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev

logging:
  level:
    org:
      springframework:
        cloud: DEBUG
        web: DEBUG
```

user-service

```
spring:
  application:
    name: user-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
```

/key/mealtoyou.json

```
{
  "type": "service_account",
  "project_id": "",
  "private_key_id": "",
  "private_key": "",
  "client_email": "",
  "client_id": "",
  "auth_uri": "",
  "token_uri": "",
  "auth_provider_x509_cert_url": "",
  "client_x509_cert_url": "",
  "universe_domain": ""
}
```

alarm-service

```
spring:
  application:
    name: alarm-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
fcm:
  service-account-file: key/mealtoyou.json
  topic-name: testMessage
  project-id: mealtoyou-197d9
```

chatting-service

```
spring:
  application:
    name: chatting-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
```

community-service

```
spring:
  application:
    name: community-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
```

food-service

```
spring:
  application:
    name: food-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
  profiles:
    include: oauth
```

health-service

```
spring:
  application:
    name: health-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
  profiles:
    include: oauth
```

supplement-service

```
spring:
  application:
    name: supplement-service
  config:
    import: optional:configserver:${CONFIG_IMPORT}
  cloud:
    config:
      profile: dev
  profiles:
    include: oauth
```

bootstrap.yml

config

```
spring.application.name: spring-vault-demo
spring.cloud.vault:
  authentication: TOKEN
  token: ${VAULT_TOKEN}
  uri: ${VAULT_URI}
  fail-fast: true
  config.lifecycle.enabled: true
```

AI

.env

```
MODEL_PATH=${MODEL_PATH}
OPENAI_API_KEY=${OPENAI_API_KEY}
DATABASE_URL=${DATABASE_URL}
```

.deploy

```
EUREKA_SERVER=${EUREKA_SERVER}
INSTANCE_HOST=ai-service
ENVIRONMENT=docker
REDIS=redis
```